

Translated from vietnamese and complied to AdobePDF
by **LithiumLi** of *forum.astalavista.ms*.



Introduction

1.1 Introduction

- Dear You, are "in" in your book is **REA_UnPacKing Ebook**, this is a document of the forum **Reverse Engineering Association (REA)** of the articles related to field **Manual unpack** by members in BQT also has a brother in contributions during the time since the establishment of REA to the present. Variety of genres, rich in content, short, and animals with the appropriate "level" is that REA hope will achieve in this document.
- Looking back through time, REA has the step is quite long in Reverse. The success that REA have today are because of the contribution does not stop the rest of the members of the REA. The administrator (BQT) REA Thank and send the best wishes to all members involved and sticking with REA.

1.2 History development process REA

- This is not a document first REA general, before this document REA have the document as **REA-cRaCkErTeAm Tutorials, PE Tutorials, Reverse. Net Software, Tutorials v. IDA. Etc..** And the future expectation is there will be complete **OllyDbg Tutorials J** . With the document **REA_UnPacKing** hope this gives you the benefit of no less what the material above, but with many members participating REA sure many people still wonder REA got what? So I would man allowed on behalf of BQT REA brief summary of a few lines of the formation and development of the REA to the present. History is not to say about yourself that REA purpose here is to help members of the REA have more understanding of the formation of "the family" that you are attending.
- In the early days inceptive, REA is just a box of the **HVA**. Member of the box is not much (I also personally involved HVA since **13/07/2004**). At that time, this box operation brought heavy warez is calculated and a lack of depth technical Reverse, which may then the concept of RE is still doubtful. Activities mainly by box is to provide a crack for software that members of the HVA ask for help. This box to that time there was quite perplexed with text ... Request.
- Until tut series of "**How to become a cracker**" was born Cracking the box was moved to a new step parentheses. Turn other members that are now pillars of REA appear. Can say this is the golden period of the HVA Cracking box. Activities really exciting, tut the series was created across all sectors: patch, crack, keygen, unpack ... to excel in those named as Computer_Angel, Zombie, Moonbaby, RCA, Deux , hacnho, and infinite v. dqtn. v.. (and some other members but I do not remember).
- With developed more quickly requires a playground for Cracking the three members at that time is **Computer_Angel - Zombie - Moonbaby** offline after a meeting has decided to establish a separate forum, a playground of their own and separated from the box really Cracking of HVA. 3 admin REA's sit together and officially established with the domain name forum is **reafareastking** (domain name to remember it too difficult) and the REA was born from that.
- But was separated from HVA (a popular forum at that time) and work on a host's REA but still not so that going down. Confirm it is for the participation of a range of key members such as **RCA, Deux, LittleBoy, QHQCcrker, benina, kienmanowar and the_Lighthouse**. Lao **hacnho** now also participate in discussions but REA aged and another member is **tladn** a separate team get called **Vietnamese Cracking Team**, an independent Web site for Unpacking the domain is <http://tothesky.U.S.>, members dqtn is also a member of good doctors but also biu busy with his team at the time, my memory is not mistaken deciduous (www.phudu.com) J .

- Living is a time and gradually to the stable REA suffered a big loss is **QHQC** members would withdraw from the REA (for personal reasons), member **the_Lighthouse** are living very well with the explosion many disappeared without a goodbye (cause I is not convenient to say here). However this time the REA welcoming a new member is **hoadongnoi**, this may say a woman who is very active and is only made to the REA J mod. Activities for a period of time following the REA to share hands member **Deux** is key, now the site is aged hacnho (tothesky.us) also attention but then it also must be accompanied by the feeding of regret many people, and **hacnho tlandn** of activities at REA as VIP members. Site of deciduous bác **dqtl** also departed, later doctors still do not know any other site again, just remember that the doctors also living in a time REA and Offline.
- REA of my time to join the official has taken a new domain (reaonline.net), I dropped in the box Cracking HVA a few times, I can remember times I reply article by a member with the nick is TQN members answer in this article about his Olly Anti com, just remember that after the items **TQN** reply that he has reg nick (**thangcuem**) to participate in the REA. With REA then really say that no more information about his TQN, but when he benina are wandering on the site outside the country see Nick TQN appear a lot to prove this is not the usual. He Benina immediately discuss with BQT seeking to enlist his TQN about living in REA. He is a member TQN older may equal or under his Benina (two did not know when he has offline age not nữa). Join a time he TQN from Nick thangcuem about TQN and VIP members of the REA. Now he has Benina gradually reduce the REA activities.
- Time activities follow REA welcoming the new members as **Thug4lif3** (forget to take the only time he's writing in this khúc), **Merc, light.phoenix, TrickyBoy, WhyNotBar, takada**, brother and many more and ... etc.. Thug is little writing but have broad knowledge, I have time to work with and Thug to present and future is still good brothers. Merc know my time to do a workshop with Thug, later in the Merc and Thug in the general level 3. At the par Merc REA is the capital of the speed with Soft keygen to dizziness. Light.phoenix is also a member of good, living seriously and have discussed the technical high also on the table by REA more gold in the Promotion admin Moonbaby have initiated. Two members can not follow is not to say that TrickyBoy and WhyNotBar, can say two members of REA emerging very rapidly after a series of tuts about Armadillo's aged in nho.Hien they are mod by REA. After the additional staff, the new mod hoadongnoi by REA start the work of individuals, gradually the regular activities in REA again.
- Computed at the time after lúra trickyboy, whynotbar, the REA takada welcoming some members also have many other factors such as quality **XIANUA, Moth, error, nhc1987**,

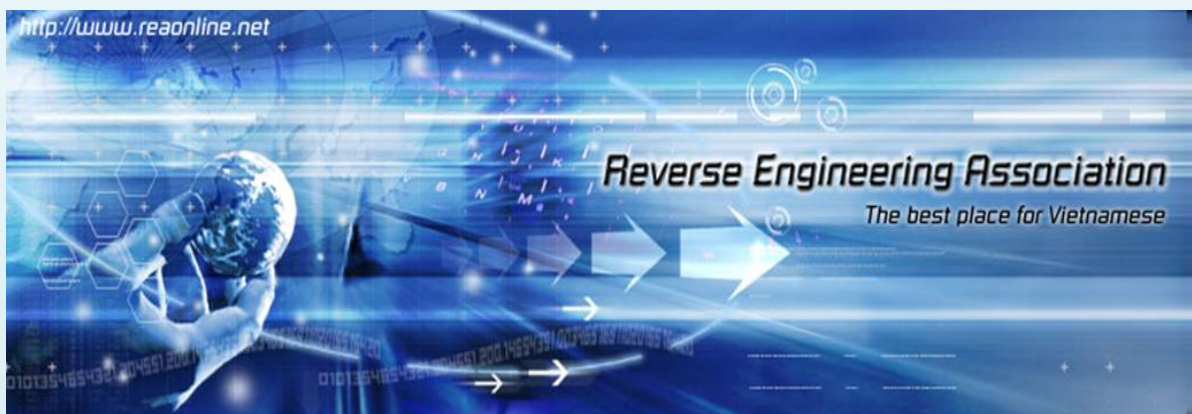
Akira, V. mrangelx. ... And

- After 4 years to work on a separate server, and former Admin Moonbaby Zombie has temporarily withdraw abandon the RCA, and Merc kienmanowar up to management (English Com is demanding he withdraw them not to hehe) to the time the current number of members of the REA has up to about 800 members. Can you ask a forum with about 800 members, the star called forum is exciting? Please answer the REA is now active in a number of rules is:
 - Quality over quantity: The members registered but not active within 1 month will be deleted from the database, as well as messages with chit chat, it will enable not exist in the REA.
 - No existing lessons Request: REA activities based on the sharing of knowledge is key, not the place to resolve the requirements individuals related Cracking / Unpacking. With these messages are the natural request were treated according to laws of the REA.
 - Activities under rules "family": When the REA is whether the level of your "high" in size to do what should probably just be used. "Glass on the franchise," "humble, very thà" rules are invariable by REA.
- Is the brief introduction and history of the establishment and development of REA. The hope was to bring you an overall look and use the referral back here for a little space for limited capital for the posts of technical Unpacking.

Body sent.

On behalf of BQT REA.

kienmanowar



Amardillo 4.xx-Patching Hardware Fingerprint (HWID)

Contents Table:

I. Intro:

HWID for II.Patching packed dll:

HWID for III.Patching packed EXE:

IV.Ending:

Tools: Hide OllyDBG + plugins, WinHex, ArmDetach 1.1

Skill Request: Using Basic Knowledge Olly + manual unpacking Armadillo.

I. Intro:

Hi all of you, after the World Cup flavor concentration specify that all the world enjoy, this is back on regular pace. Work, study, eat and play ... spare seat crack. J

Today tricky introduces a new way to patch Hardware Fingerprint Armadillo 4.xx. Method patch for version 1 is the author is mentioned (Armadillo Memory Patch Trick), this has not been used again.

Here please note that you, this document is only for reference, if someone misused to register illegal software, or other purposes is bad authors ko responsibility where nhé.

The Hardware Fingerprint is wai What? tricky in a tí nhé.

Armadillo just 1 packer has always help create mechanisms to register KEY software. If you try to pack a few files with Armadillo will have seen the creation KEY. With each choosing Option in the Armadillo, will be born an algorithm to create and check key.Nhung general picture is as follows:



As you see HW ID is an important part in the process of registration. But why have it? In the normalization of soft just Name + KEY is enough?

Assuming only Name + KEY, the version 1 software is registered, the software that may be registered in several other machines, which used to simply use the Key Name +. This is the manufacturers do not want to because they need the software sold by the copy on each machine. Thus was born HW ID, including at the Armadillo and protect others. HW ID is calculated from how to get the parameters of the hardware on your computer, can be that is the CPU, RAM, harddisk, ... but overlapping cases in the machine is very small, so does the probability 1 Name + KEY limited, it makes the production more assured.

However, what ko ko flawless. Whether HW ID on each machine born different, but if we have 1 HWID KEY + 1 + 1 Name was valid on 1 May that we can patch them into HWID HWID machines need to register, and of course KEY + name will go under the valid function to check key. In addition, the number 1 or you wonder how to unpack 1 soft pack by armadillo while it require any key? Course is based on the above, it should at least 1 HWID KEY + 1 + 1 Name to register after the registration is valid, the code are completely unpack, we are new to OEP to dump. ... Etc. The format is quite soft as little as how to protect the type of "not see the soft side was demanding KEY" is just the type 1 or just unfinished. Or have in place ko ko KEY then read how Crack was, but because the seats do see how the soft Who is buying it. So a new type 1 KEY Trial is key, it helps us see the face of soft nose how, KEY but will expire within 1 certain number of days, then we must unpack before KEY expired or clear course in the Registry to xài from the beginning .. v.. v..

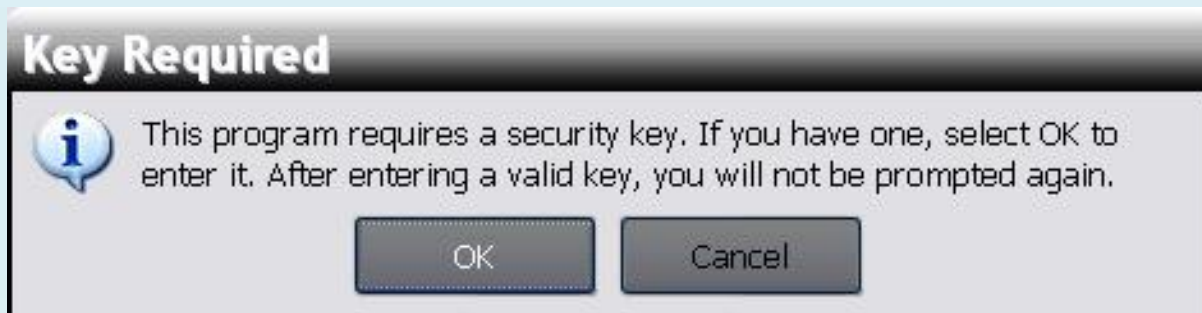
If you do not have the above, you temporarily forget to unpack it. Up to the time now, the computer does not play in the key check of the Armadillo. (As does the level of 10 to key, the brute force it

takes hundreds of years)

Say many things, now we see the method to patch HWID Armadillo 4.xx how nhé.

HWID for II.Patching packed dll:

Dll tricky to select examples before because if the dll lickerish. Exe as easy to do right? In this example, one pack of Ukhook40.dll file Unikey 4 beta. When running unikey.exe, dll load and requires the key:



Okie, dom face each other for fun. Watch list shows the process unikey.exe 1 process, 1 dll UKHook40.dll the load, so this is dll protect standard form (the medical information of why the format ko protect Debug Blocker for the dll).

Okay, close to all, open up Olly <<ke payment of transmission kiếp Armadillo>. Load UKHook40.dll to:

Address	Hex dump	Disassembly
008D80C7	55	PUSH EBP
008D80C8	8BEC	MOV EBP,ESP
008D80CA	53	PUSH EBX
008D80CB	8B5D 08	MOV EBX,DWORD PTR SS:[EBP+8]
008D80CE	56	PUSH ESI
008D80CF	8B75 0C	MOV ESI,DWORD PTR SS:[EBP+C]
008D80D2	57	PUSH EDI
008D80D3	8B7D 10	MOV EDI,DWORD PTR SS:[EBP+10]
008D80D6	85F6	TEST ESI,ESI
008D80D8	75 09	JNZ SHORT UKHook40.008D80E3

Here ko mentioned to Hide Debug or configuration Olly néh What more, you have to find ways IsDebugPresent Hide and OutputDebugStringA, as well as other bypass the Exception. Free of the stars click RUN (F9), Olly run circuit, springiness NAG Armadillo's key demands:

Enter Key

Enter the registration name and key below, exactly as given to you.

Hardware fingerprint: 0425-75DD

Name:

Key:

OK

Cancel

NAG has springiness, find time to see NAG this year in memory, back through Olly, Pause for the F12, Alt-K:

Address	Stack	Procedure / arguments	Called from	Frame
0006851C	7C90D85C	Includes ntdll.KiFastSystemCallRet	ntdll.7C90D85A	00068574
00068520	7C8023ED	ntdll.2wDelayExecution	kernel32.7C8023E7	00068574
00068578	7C802451	? kernel32.SleepEx	kernel32.7C80244C	00068574
0006857C	00000001	Timeout = 1. ms		
00068580	00000000	Alertable = FALSE		
00068588	009E34A5	? kernel32.Sleep	009E349F	00068584
0006858C	00000001	Timeout = 1. ms		
000685C0	009F4840	? 009E33B4	009F483B	000685BC
000691F8	009F9169	009F461E	009F9164	000691F4
0006E944	009F687E	009F6018	009F6879	0006E940

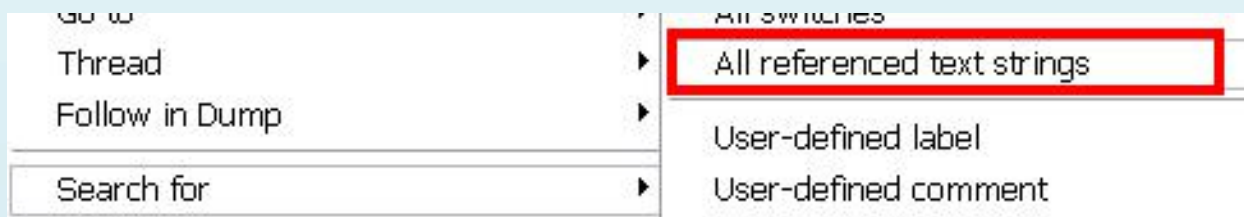
Select 1 of the line is highlighted, as they are under the area code of the key check NAG, double click on:

Address	Hex dump	Disassembly
009F9164	E8 B5B4FFFF	CALL 009F461E
009F9169	83C4 0C	ADD ESP,0C
009F916C	0FB6C0	MOVZX EAX,AL
009F916F	85C0	TEST EAX,EAX
009F9171	75 07	JNZ SHORT 009F917A
009F9173	80A5 F4F9FFFF	AND BYTE PTR SS:[EBP-60C],0
009F917A	0FB685 F4F9FFFF	MOVZX EAX,BYTE PTR SS:[EBP-60C]

Based on Offset (your computer may be different), we see this area code outside the section of the dll files, view memory map (if necessary):

00870000	00116000	UKHook40		Map	R	E	R	E
00870000	00001000	UKHook40	PE header	Imag	R		RWE	
00871000	0001C000	UKHook40	.text	Imag	R		RWE	
00880000	00005000	UKHook40	.rdata	Imag	R		RWE	
00892000	0000A000	UKHook40	.data	Imag	R		RWE	
0089C000	00003000	UKHook40	.reloc	Imag	R		RWE	
0089F000	00040000	UKHook40	.text1	Imag	R		RWE	
008DF000	00010000	UKHook40	.adata	Imag	R		RWE	
008EF000	00010000	UKHook40	.data1	Imag	R		RWE	
008FF000	00010000	UKHook40	.reloc1	Imag	R		RWE	
0090F000	000A0000	UKHook40	.pdata	Imag	R		RWE	
009B0000	00004000			Priv	RW		RW	
009C0000	00002000			Map	R		R	
009D0000	00065000			Priv	RW		RW	
00940000	0000C000			Priv	RW		RW	

Back to this area code, click to select:



When the Search String, 1 string tricky found suspicious, the **form&percent;&percent;** It's like chain HWID we also see the form xxxx-xxxx. After many times BP set to test, was tricky to find out where 1 is very interesting:

00900773	PUSH 0A093C0	ASCII "%u"
0090078C	PUSH 0A098A4	ASCII "%s"
009007AB	PUSH 0A09898	ASCII "DATELASTRUN"
009007D6	PUSH 0A096F0	ASCII "%04X-%04X"
009007EF	PUSH 0A0988C	ASCII "FINGERPRINT"
00900822	PUSH 0A09880	ASCII "????-????"
00900844	PUSH 0A096F0	ASCII "%04X-%04X"

04X-04X appear more than once, but several positions makes it so interesting, you can search quickly by searching **DATELASTRUN** sequence as in the picture. Now Double Click on the series:



On it is 1 function call, we enter into, see:



Do not ask why the tricky bit, as above: "After many times BP set" should be found, the time you enter to function CALL in the image above, to this:



Asked order **XOR EAX,** is at home items, time set on BP Hardware execution to it:

Breakpoint	Toggle	F2
New origin here	Conditional	Shift+F2
Go to	Conditional log	Shift+F4
Thread	Run to selection	F4
Follow in Dump	Memory, on access	
Search for	Memory, on write	
Find references to	Hardware, on execution	

Ctrl-F2, restart again Olly you go. Then F9 to run, you will break at BP has set:

Address	Hex dump	Disassembly
009E5DA3	35 8AC0E665	XOR EAX,65E6C08A
009E5DA8	5F	POP EDI
009E5DA9	5E	POP ESI
009E5DAA	C2 0800	RET 8
009E5DAD	E8 05000000	CALL 009E5DB7
009E5DB2	E9 0C000000	JMP 009E5DC3

So we break here before NAG require ra.Gio F8 KEY springiness of a trace, dom through results EAX XOR command after me:

Registers (FPU)	
EAX	042575DD
ECX	00A710C0
EDX	00000000
EBX	00A0FB00
ESP	0006DA38

EAX = 042575DD. it is the Hardware ID is born on our (ko marked "-"). So we do need to know the specific process of calculating how HWID, only know me XOR command is the last step to make HWID to check the key. Hehe, you emerged the idea of what has not? then it is the right man, just patch me this command to **MOV EAX,** with **xxxxxxxx** xxxxxxxx Hardware ID is attached KEY + Name that we are on another machine. Asked command also take 5 bytes, so do not cause any benefit after patch. Ah, wen for HWID KEY + + Name not valid anymore, it's here:

HWID: 8300-4214

Name: **REA CrAcKerTeAm**

Key: 000003-1V3UNC-4FM85J-EYXFAQ-TCY9F2-GZPP29-BP9TAG-3F7DWR-XA2PJ9-M2BY8W-ATH9N0-392UY0-NTR7XW-H1U92Y

So time is on the patch command to me: **MOV EAX,** so it always returns HWID that we need, then **set new Origin** at me this command to run back and change the value of EAX:

Address	Hex dump	Disassembly
009E5DA3	B8 14420083	MOV EAX,83004214
009E5DA8	5F	POP EDI
009E5DA9	5E	POP ESI
009E5DAA	C2 0800	RET 8
009E5DAD	E8 05000000	CALL 009E5DB7
009E5DB2	E9 0C000000	JMP 009E5DC3

Now, do not delete them farewell BP, subject to press F9 to run, you'll see it break here many times, to see HWID calculated multiple times, so if you do not patch that modify the value EAX hand, sure of you evil hands. Okie, after many times F9, you also to the NAG, and look to see:



Hehe, HWID HWID is now on who should do so in tricky oi (specific Why are doctors). But we just changed the chain and the only HWID for bowel function to check the key do? Income + Name Key is to know now. After entering, we also see it back to me on command several times, under more difficult F9 small, as it demonstrated that the code key, it is always charged to the back several times. Finally, 1 notification unwind:



Is horror movie you have to do. J Hehehe. Now run to run again, it will also require ko key again, because key + name + HWID was stored in the Registry (of course can encrypt). Each run, it dom in the Registry, to see Okie ko han asked what police.

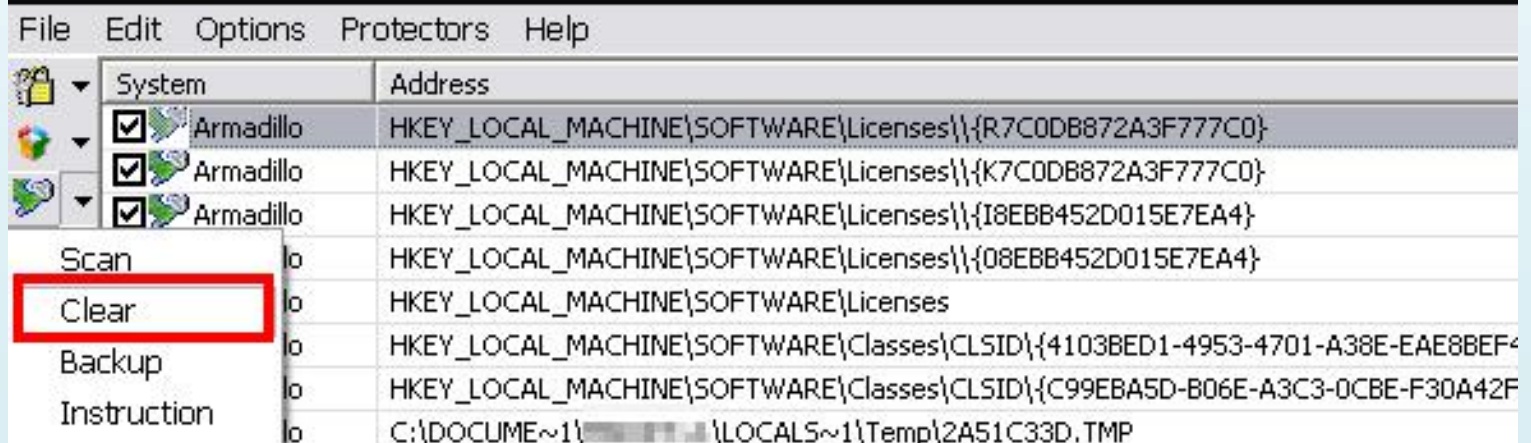
After Patch HWID for valid key, you can open to OEP's dll and unpack conducted as usual. Here we do not talk to unpack. Kĩ time to notice the times in order XOR Break on, the special 1 In particular, after entering valid key, the calculation is still HWID recalculate. This method makes Memory Patch Trick previously failed. Also making 1 cause it failed. You will go to know.

Now you open [Trial Reset 3.0 RC1](#) that tricky attached. Click on the icon of Armadillo Scan to clear

the course and registered in the Registry.

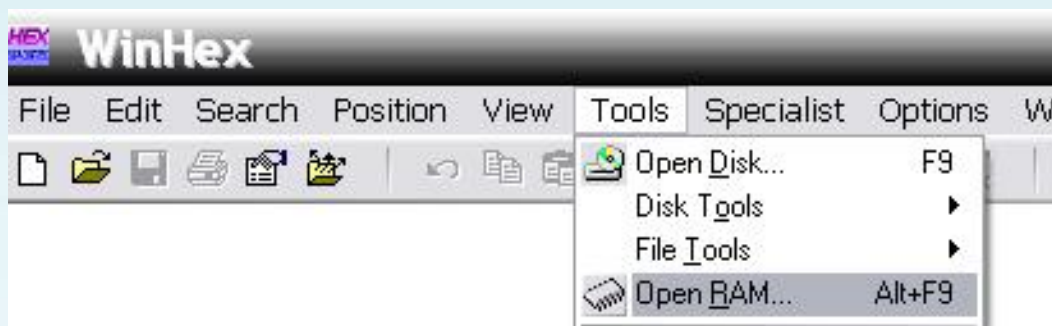
Note: if the computer has 1 software pack by Armadillo and it is registered may be lost after Clear Key. Backup should be the key if necessary Registry (backup based courses Trial Reset the scan)

R Trial-Reset 3.0 RC1



Once clear, the Unikey our return to the status has not registered, and it requires the same HWID KEY old. L No problem, that is what we want, as tricky to introduce the 1 patch more quickly with using Winhex, 1 tool mentioned in Memory Patch Trick.

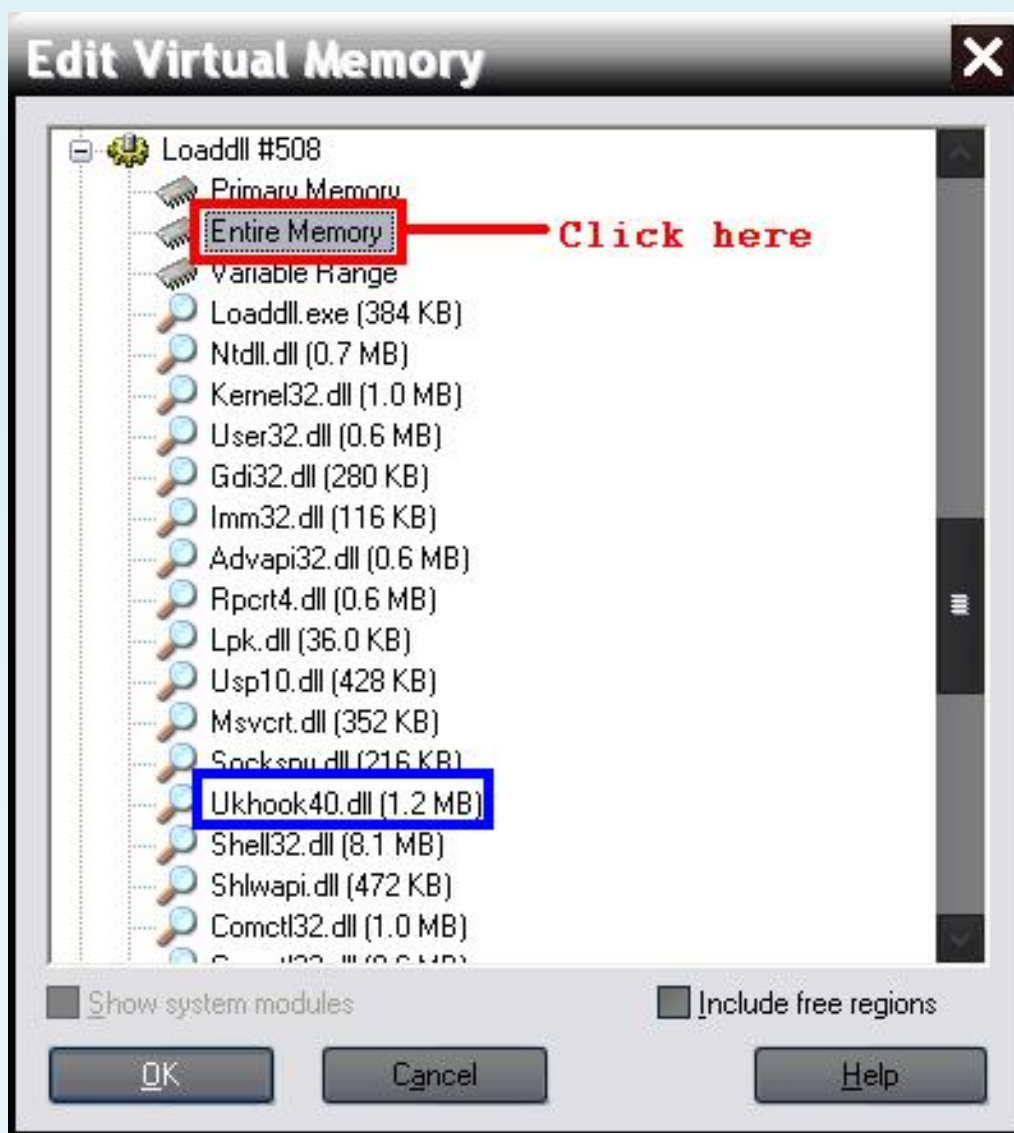
But we still use Olly to run this dll, because some cases, assume we do have main. Exe dll to load out, it can be used to temporarily Olly load. After you load and run, does require KEY NAG. Keep it open Winhex up time. Select Tools à Open RAM (or press Alt + F9):



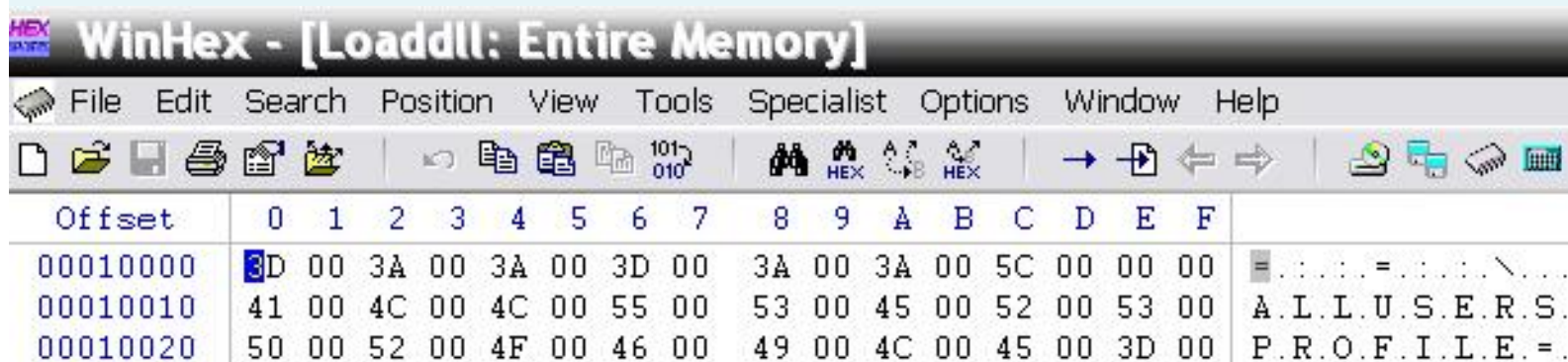
At Process list, select the process LoadDll by Olly:



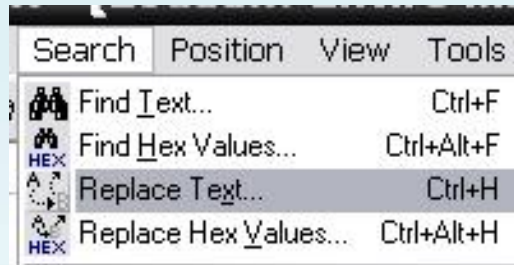
Assuming do here is by Olly LoadDll the Unikey you choose (if the main run. Exe outside Olly). Then click marked "+" next to point out, choose **Entire**



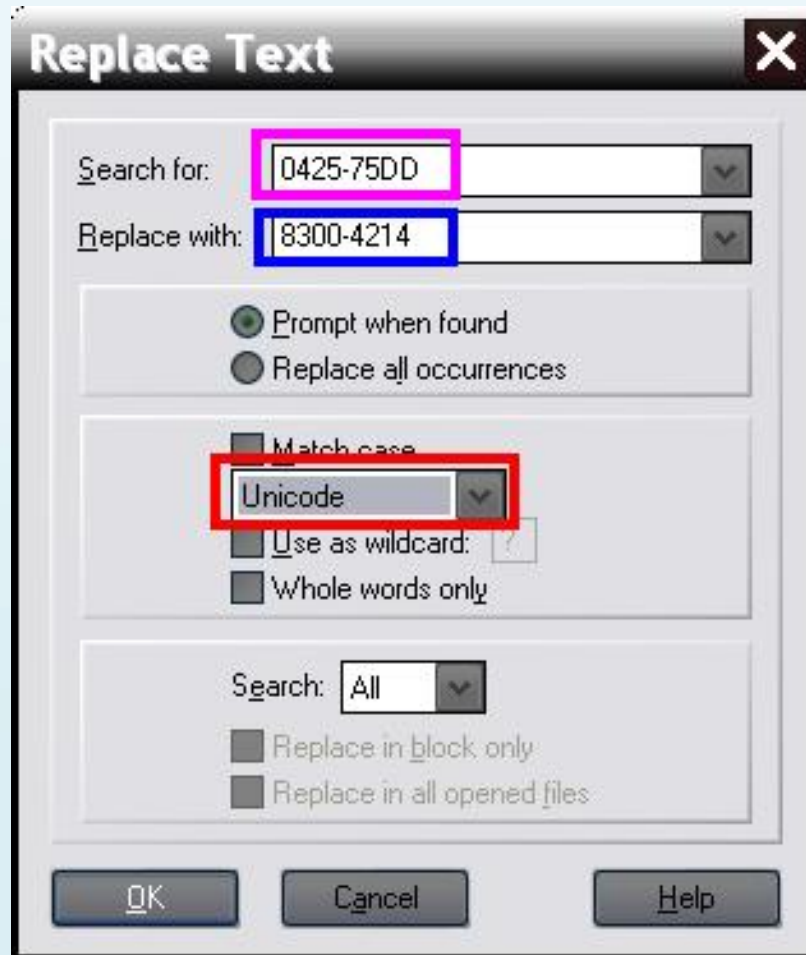
Now you see quite the same tut Memory Patch Trick, but tricky for you will see the difference. Once selected as the one to come:



In the old tut, we will search by Byte reverse HWID and replaced by Byte HWID accompanied by valid key. Ko It is also in ver 4.xx by Armadillo. Because now it Unicode text format but also ko byte Memory in normal again, this is also the reason that tricky 2 above. So choose à Replace Text Search:



Select Text format Unicode Search is needed, then fill HWID on Search box, complete with valid HWID KEY Replace the box:



Click OK to see how many times Replace in memory:



Back found that HWID changed:



One more note, if you use the Winhex not actually registered, the Replace function to do work. At present of the Internet have Winhex keygen attached (ZWT group) but keygen that work effectively completed ko whole. Since when does the function of Winhex, the check will be Fake KEY check 1 again only key Xin really new xài be, or you need to patch more Winhex.

Back to the issue, as above, but change is HWID in memory, but after enter key, the students are still

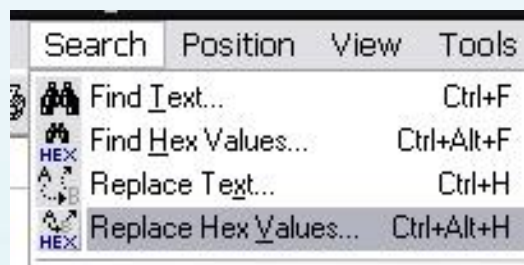
HWID recalculate, so we still have to order me Patch **XOR EAX, 65E6C08A**

There is only one patch in Winhex, hehehe. Forgot to say, the key questions in this command is the only, **only exists only 1 question in this command Check the memory of the key, and more, in ver 4.xx of Armadillo (from 4.0-4.4) , have asked this command.**

Any time that the question of Byte Opcode see this command:

Address	Hex dump	Disassembly
009E5DA3	35 8AC0E665	XOR EAX, 65E6C08A
009E5DA8	5F	POP EDI
009E5DA9	5E	POP ESI
009E5DAA	C2 0800	RET 8
009E5DAD	E8 05000000	CALL 009E5DB7
009E5DB2	E9 0C000000	JNE 009E5DC3

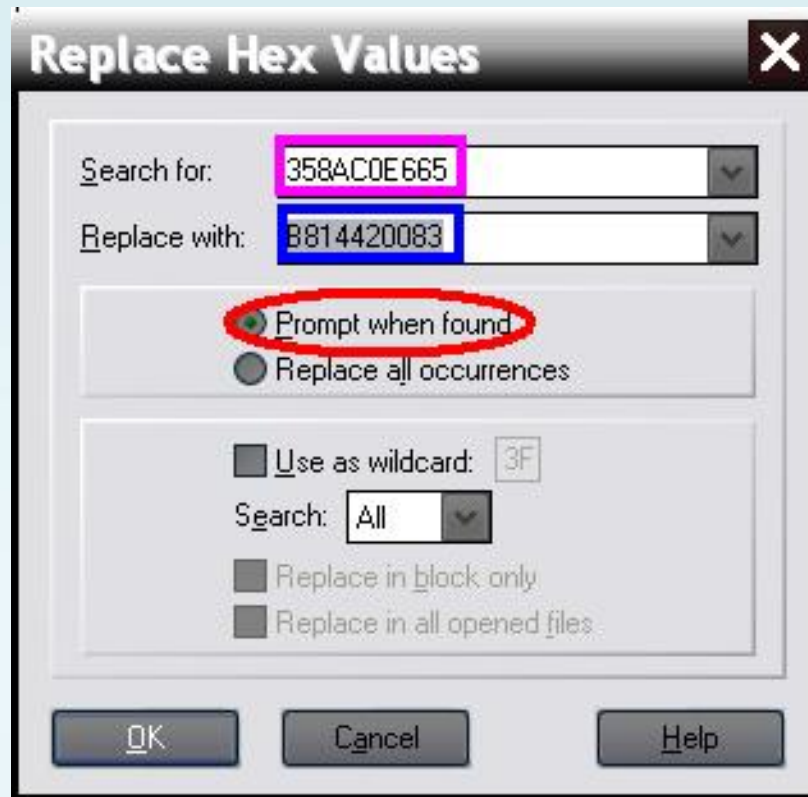
As **358AC0E665**. This is the reverse of bytes in memory. Should we just search in italy chang Winhex. Drag the scroll bar back to the top Memory for the search ko mistake, then select Search à Replace Hex Values:



Search is at the Byte **358AC0E665**. Replace the box fill what? Sure we change the order using XOR
What ko:

Address	Hex dump	Disassembly
009E5DA3	B8 14420083	MOV EAX, 83004214
009E5DA8	5F	POP EDI
009E5DA9	5E	POP ESI
009E5DAA	C2 0800	RET 8
009E5DAD	E8 05000000	CALL 009E5DB7
009E5DB2	F9 0C000000	JNS 009E5DC3

Based on HWID and Opcode of command, and vice versa as in the picture, you fill the box replace the Byte: **B814420083**, remember to select **Prompt when found** to occur less errors:



Click OK, and only 1 time Replace.



Now, enter the name still valid + Key call this:



HWID for III.Patching packed EXE:

Now the session. Exe. Tricky Pack 2 has the form, a Standard for unikey.exe file, the process is similar dll file, the only other time this man meat unikey.exe. **Remember to use the Clean Clear Trial Reset will of course armadillo for the dll before making contact with. nhé exe.**

Here tricky only guide you to the Patch 2: Debug Blocker has (or accompanied CopyMem) on unikey.exe. When run, it does 2 process:

UniKeyNT.exe	2824		
UniKeyNT.exe	1364		
procexp.exe	3384	0.76	Svsii

And also require KEY:



Here again is 1 HWID other, easy to understand, because this is from HWID. Exe. Only to do it khùng or that the form FFFF-FFFF on tricky. On the other you will jog. However it ko important, the main change is to be accompanied by valid HWID KEY. Where we have this:

HWID: **F9D0-F6CE**

Name: **REA CrAcKerTeAm**

Key: **000014-XN0MTU-A92HZE-JYTM6H-ZFX8KE-ZU2N2C-QEZMV8-D9DRDT-GUMU5H-RD83PB**

Okie, close to the end. Because here are 2 of the process Debug Blocker (ko concern is CopyMEM or do), we use 1.1 ArmDetach by RES) (Debug Blocker to defeat the fast. Drag the file to Unikey.exe ArmDetach you go:



Child process PID on your computer may be different. Now open up Olly (ko close ArmDetach) Attach Child Process:

```
00000ABC Un iKeyNT
00000FE8 Un iKeyNT
```

F9 to run, then pause the F12, in order Follow dump - JMP, to correct EB FE 60 E8, we are:

Address	Hex dump	Disassembly
00481000	60	PUSHAD
00481001	E8 00000000	CALL UniKeyNT.00481006
00481006	5D	POP EBP
00481007	50	PUSH EAX
00481008	51	PUSH ECX
00481009	0FCA	BSWAP EDX
0048100B	F7D2	NOT EDX
0048100D	9C	PUSHFD
0048100E	F7D2	NOT EDX
00481010	0FCA	BSWAP EDX
00481012	EB 0F	JNS SHORT UniKeyNT.00481023
00481014	B9 EB0FB8EB	MOV ECX,EBB80FEB
00481019	07	POP ES
0048101A	B9 EB0F90EB	MOV ECX,EB900FEB
0048101F	00	JMP

Address	Hex dump	ASCII
00481000	60 5D 00 00 00 00 5D 50 51 0F CA F7 D2 9C F7 D2JP
00481010	0F CA EB 0F B9 EB 0F B8 EB 07 B9 EB 0F 90 EB 08	#####

Speaking brief because as you unpack the Armadillo. Now F9 run for running a circuit (as was hide and debug exception bypass). Key frames require flying. Back Olly. Pause F12, Alt-K is the same as in the dll, find the code of the key check:

Address	Stack	Procedure / arguments	Called from
000687C8	7C90D85C	Includes ntdll.KiFastSystemCallRet	ntdll.7C90D85A
000687CC	7C8023ED	ntdll.ZwDelayExecution	kernel32.7C8023E7
00068824	7C802451	? kernel32.SleepEx	kernel32.7C80244C
00068828	00000001	Timeout = 1. ms	
0006882C	00000000	Alertable = FALSE	
00068834	009B34A5	? kernel32.Sleep	009B349F
00068838	00000001	Timeout = 1. ms	
0006886C	009C4840	? 009B33B4	009C483B
000694A4	009C9169	009C461E	009C9164
0006EBF0	009C687E	009C6D18	009C6879

Double Click. Here to demonstrate the unique command of XOR above, you first pull up in this area remember 9A0000: (your computer may be different)

Address	Hex dump	Disassembly	Comment
009A0000	4D	DEC EBP	
009A0001	5A	POP EDX	
009A0002	90	NOP	
009A0003	0003	ADD BYTE PTR DS:[EBX],AL	
009A0005	0000	ADD BYTE PTR DS:[EAX],AL	
009A0007	000400	ADD BYTE PTR DS:[EAX+EAX],AL	
009A000A	0000	ADD BYTE PTR DS:[EAX],AL	
009A000C	FFFF	???	Unknown command
009A000E	0000	ADD BYTE PTR DS:[EAX],AL	
009A0010	B8 00000000	MOV EAX,0	
009A0015	0000	ADD BYTE PTR DS:[EAX],AL	
009A0017	0040 00	ADD BYTE PTR DS:[EAX],AL	
009A001A	0000	ADD BYTE PTR DS:[EAX],AL	
009A001C	0000	ADD BYTE PTR DS:[EAX],AL	

Ctrl-F command to search, fill **XOR EAX, 65E6C08A**

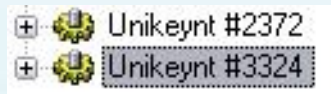


OK, it stops only at 1 time:

Address	Hex dump	Disassembly
009B5DA3	35 8AC0E665	XOR EAX,65E6C08A
009B5DA8	5F	POP EDI
009B5DA9	5E	POP ESI
009B5DAA	C2 0800	RET 8
009B5DAD	E8 05000000	CALL 009B5DB7
009B5DB2	E9 0C000000	JNG 009B5DC3
009B5DB7	68 C8319E00	PUSH 9E31C8

To do this it should say more. Tricky just want to prove the only order of questions. Now, Close up, we'll patch it with Winhex (winhex is the fastest method for this).

Olly Unikey.exe running out, the demands on the key, open Winhex. Open Tools à RAM. There is only 2 in the list Winhex process, and we always choose to process 2 (ko interested need ID)



Because process 2 sure Child process. Now everything was as normal as the dll, even HWID other alternative at this time. The first is for Unicode Text:



OK:



Continue to command XOR, here is the replacement bytes **B8CEF6D0F9**



Load and Valid key Name:



So once the patch method HWID's Armadillo 4.xx (4.0-4.4) Rationalization of the work to unpack the Armadillo dll in this tut, tut because of the hacnho respond before too hasty but forget a few things for Newbie. But consider this target has not found enough to create complete tut 1. See you after that (if possible)

IV.Ending:

Please note the 1 again, this document is only for reference, if someone misused to register illegal software, or other purposes is bad authors ko responsibility where nhé.

Sincere thanks:

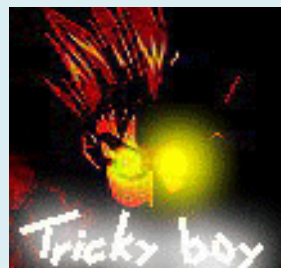
The brother-in REA helped tricky in the study.

Typically hacnho series with tut unpack their super levels.

Uncle Why Not Bar for some tricky help pack the file Unikey

Takada has giùm test on the other bytes.

... And you all again.

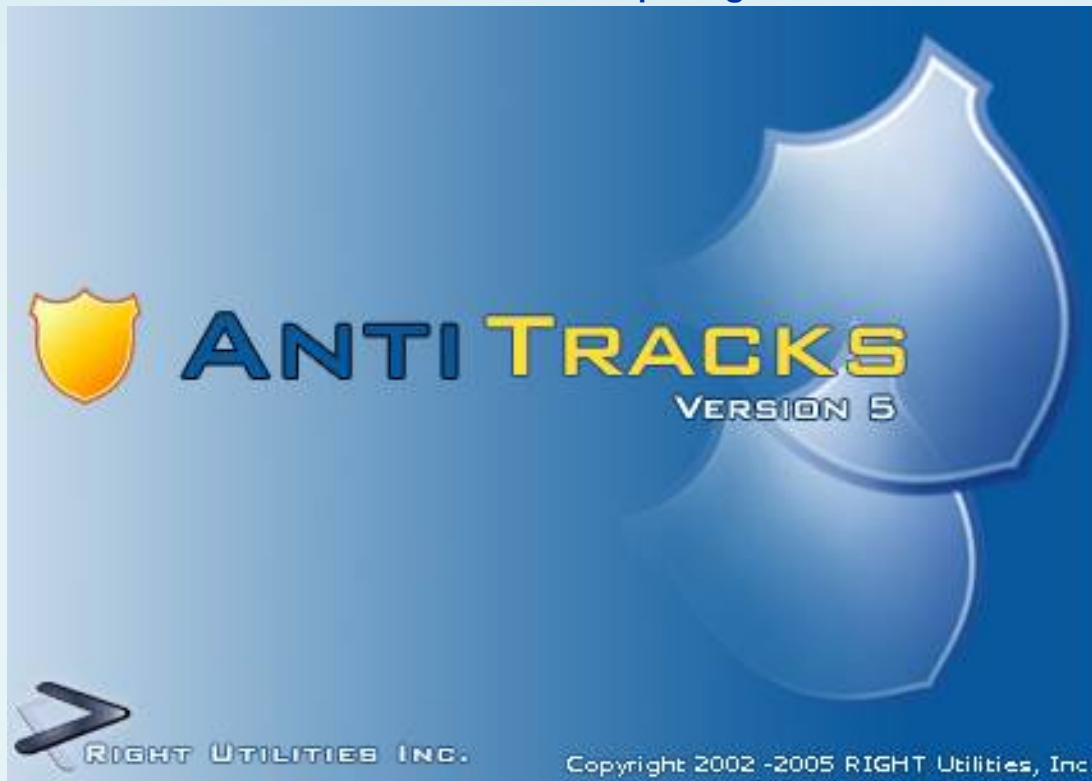


Tàn Ma Kiem
(Written by Trickyboy)

Armadillo collect sand-stone

Armadillo 4.xx-Code Splicing tut!

Target: **Anti Tracks v5.6.1**
ARM 4.xx-Code Splicing



Tools:

- 1.OllyDBG - The best config debugger for ArmMUP by hacnho.
- 2.LordPE 1.4 Deluxe
- 3.Import REConstructor 1.6 Final
- 4.ArmInline 0.6

Unpacking

_Load On target:

Address	Hex dump	Disassembly	Comment
00540323	55	PUSH EBP	
00540324	8BEC	MOV EBP,ESP	
00540326	6A FF	PUSH -1	
00540328	68 20AB5600	PUSH AntiTrac.0056AB20	
0054032D	68 60005400	PUSH AntiTrac.00540060	
00540332	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	SE handler installation
00540338	50	PUSH EAX	
00540339	64:8925 000000	MOV DWORD PTR FS:[0],ESP	
00540340	83EC 58	SUB ESP,58	
00540343	53	PUSH EBX	
00540344	56	PUSH ESI	
00540345	57	PUSH EDI	
00540346	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00540349	FF15 88515600	CALL DWORD PTR DS:[<&KERNEL32.GetVersion	kernel32.GetVersion
0054034F	33D2	XOR EDX,EDX	

_HE GetModuleHandleA, Shift + F9:

Address	Value	Comment
0012FF38	005403EB	CALL to GetModuleHandleA from A
0012FF3C	00000000	pModule = NULL
0012FF40	00000000	
0012FF44	00141F0D	
0012FF48	0000000A	
0012FF4C	00000006	
0012FF50	00000000	
0012FF54	7FFDF000	
0012FF58	00000040	

_Shift + F9 times 2:

Address	Value	Comment
0012CED8	77C959FC	CALL to GetModuleHandleA from m
0012CEDC	77C131AC	pModule = "kernel32.dll"
0012CEE0	77C5CA20	msvcrt.77C5CA20
0012CEE4	00000000	
0012CEE8	77C1E94F	msvcrt.<ModuleEntryPoint>
0012CEEC	77F59A7B	RETURN to ntdll.77F59A7B from nt
0012CEF0	0000E323	
0012CEF4	0012CEE0	
0012CF58	77570172	kernel32.GetEnvironmentVariableA

_Shift + F9 times 3:

Address	Value	Comment
0012CF9C	70A78663	CALL to GetModuleHandleA from S
0012CFA0	70A7F8FC	pModule = "KERNEL32.DLL"
0012CFA4	00000000	
0012CFA8	70A70000	SHLWAPI.70A70000
0012CFAC	0012CFE8	
0012CFB0	00000001	
0012CFB4	77F5166A	RETURN to ntdll.77F5166A from nt
0012CFB8	77F5166A	RETURN to ntdll.77F5166A from nt
0012CFBC	7007825B	RETURN to SHLWAPI.7007825B from

_Lan 4:

Address	Value	Comment
0012CEC4	7712B124	CALL to GetModuleHandleA from O
0012CEC8	771A22E4	pModule = "KERNEL32.DLL"
0012CECC	7712AD56	RETURN to OLEAUT32.7712AD56 from
0012CED0	771A2080	OLEAUT32.771A2080
0012CED4	000003E8	
0012CED8	7712B0CA	RETURN to OLEAUT32.7712B0CA from
0012CEDC	00000001	
0012CEE0	00000001	
0012CF54	7712B000	OLEAUT32.7712B000

_Lan 5:

Address	Value	Comment
0012CEC0	7712B124	CALL to GetModuleHandleA from O
0012CEC4	771A22E4	pModule = "KERNEL32.DLL"
0012CEC8	7712ADAC	RETURN to OLEAUT32.7712ADAC from
0012CECC	771A2064	OLEAUT32.771A2064
0012CED0	000003E8	
0012CED4	7712B0D0	RETURN to OLEAUT32.7712B0D0 from
0012CED8	00000001	
0012CEDC	00000001	
0012CF50	00000001	

_Lan 6:

Address	Value	Comment
00120750	0052B043	CALL to GetModuleHandleA from A
00120754	00000000	pModule = NULL
00120758	00000000	
0012075C	003A7DA3	
00120760	005A9E72	AntiTrac.005A9E72
00120764	00000000	
00120768	00585000	ASCII "PDATA000"
0012076C	77F79000	ntdll.77F79000
00120770	00000000	

_Lan 7:

Address	Value	Comment
00126884	003C35E7	CALL to GetModuleHandleA from 0
00126888	003D7474	pModule = "kernel32.dll"
0012688C	003D8744	ASCII "VirtualAlloc"
00126890	00000001	
00126894	003F6A10	
00126898	00000000	
0012689C	00000000	
001268A0	00000000	
001268A4	00000000	

_Lan 8:

Address	Value	Comment
00126884	003C3604	CALL to GetModuleHandleA from 0
00126888	003D7474	pModule = "kernel32.dll"
0012688C	003D8738	ASCII "VirtualFree"
00126890	00000001	
00126894	003F6A10	
00126898	00000000	
0012689C	00000000	
001268A0	00000000	
001268A4	00000000	

_Lan 9:

Address	Value	Comment
001265F4	003AAB85	CALL to GetModuleHandleA from 0
001265F8	00126738	pModule = "kernel32.dll"
001265FC	00000000	
00126600	CCE00000	
00126604	5E940012	
00126608	003D697C	
0012660C	00000000	
00126610	80000000	
00126614	00000000	

F8 _Nhan trace through RETN order, you will come:

Address	Hex dump	Disassembly	Comment
003AAB85	8B00 E4C93D00	MOV ECX, DWORD PTR DS:[3DC9E43]	
003AAB8B	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003AAB8E	A1 E4C93D00	MOV EAX, DWORD PTR DS:[3DC9E43]	
003AAB93	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
003AAB96	75 16	JNZ SHORT 003AABCE	
003AAB98	8D85 B4FEFFFF	LEA EAX, DWORD PTR SS:[EBP-14C]	
003AAB9E	50	PUSH EAX	
003AABBF	FF15 E0203D00	CALL DWORD PTR DS:[3D20E0]	kernel32.LoadLibraryA
003AABC5	8B00 E4C93D00	MOV ECX, DWORD PTR DS:[3DC9E43]	
003AABC8	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003AABCE	A1 E4C93D00	MOV EAX, DWORD PTR DS:[3DC9E43]	
003AABD3	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
003AABD6	74 F8 31010000	JE 003AAD00	
003AABDC	33C9	XOR ECX,ECX	
003AABDE	8B07	MOV EAX, DWORD PTR DS:[EDI]	
003AABE0	3918	CMP DWORD PTR DS:[EAX],EBX	
003AABE2	74 06	JE SHORT 003AABEA	
003AABE4	41	INC ECX	
003AABE5	83C0 0C	ADD EAX,0C	

Jump _Magic, EB's patch:

Address	Hex dump	Disassembly	Comment
003AAB85	8B00 E4C93D00	MOV ECX,DWORD PTR DS:[3DC9E4]	
003AAB88	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003AAB8E	A1 E4C93D00	MOV EAX,DWORD PTR DS:[3DC9E4]	
003AAB93	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
003AAB96	75 16	JNZ SHORT 003AABCE	
003AAB98	8D85 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
003AAB9E	50	PUSH EAX	
003AABBF	FF15 E0203D00	CALL DWORD PTR DS:[3D20E0]	kernel32.LoadLibraryA
003AABC5	8B00 E4C93D00	MOV ECX,DWORD PTR DS:[3DC9E4]	
003AABC8	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003AABCE	A1 E4C93D00	MOV EAX,DWORD PTR DS:[3DC9E4]	
003AABD3	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
003AABD6	F9 32010000	JMP 003AAD00	
003AABD8	90	NOB	
003AABDC	33C9	XOR ECX,ECX	
003AABDE	8B07	MOV EAX,DWORD PTR DS:[EDI]	
003AABE0	3918	CMP DWORD PTR DS:[EAX],EBX	

_hd GetModuleHandleA, set breakpoint BP CreateThread, Shift + F9, Ctrl + F9, F8, Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
003CC35B	A1 F81E3E00	MOV EAX,DWORD PTR DS:[3E1EF0]	
003CC360	59	POP ECX	
003CC361	8B48 68	MOV ECX,DWORD PTR DS:[EAX+68]	
003CC364	3348 04	XOR ECX,DWORD PTR DS:[EAX+4]	
003CC367	3308	XOR ECX,DWORD PTR DS:[EAX]	
003CC369	F6C1 40	TEST CL,40	
003CC36C	75 08	JNZ SHORT 003CC376	
003CC36E	6A 01	PUSH 1	
003CC370	E8 17B9FDFF	CALL 003A7C8C	
003CC375	59	POP ECX	
003CC376	53	PUSH EBX	

_Cuon Down:

Address	Hex dump	Disassembly	Comment
003CC3D6	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
003CC3D9	FF76 08	PUSH DWORD PTR DS:[ESI+8]	
003CC3DC	53	PUSH EBX	
003CC3DD	E8 74E2FEFF	CALL 003BA656	
003CC3E2	50	PUSH EAX	
003CC3E3	A1 F81E3E00	MOV EAX,DWORD PTR DS:[3E1EF8]	
003CC3E8	8B48 58	MOV ECX,DWORD PTR DS:[EAX+58]	
003CC3EB	3348 30	XOR ECX,DWORD PTR DS:[EAX+30]	
003CC3EE	3308	XOR ECX,DWORD PTR DS:[EAX]	
003CC3F0	2BF9	SUB EDI,ECX	
003CC3F2	FFD7	CALL EDI	
003CC3F4	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
003CC3F7	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
003CC3FA	5F	POP EDI	
003CC3FB	5E	POP ESI	
003CC3FC	5B	POP EBX	
003CC3FD	C9	LEAVE	
003CC3FE	C3	RETN	

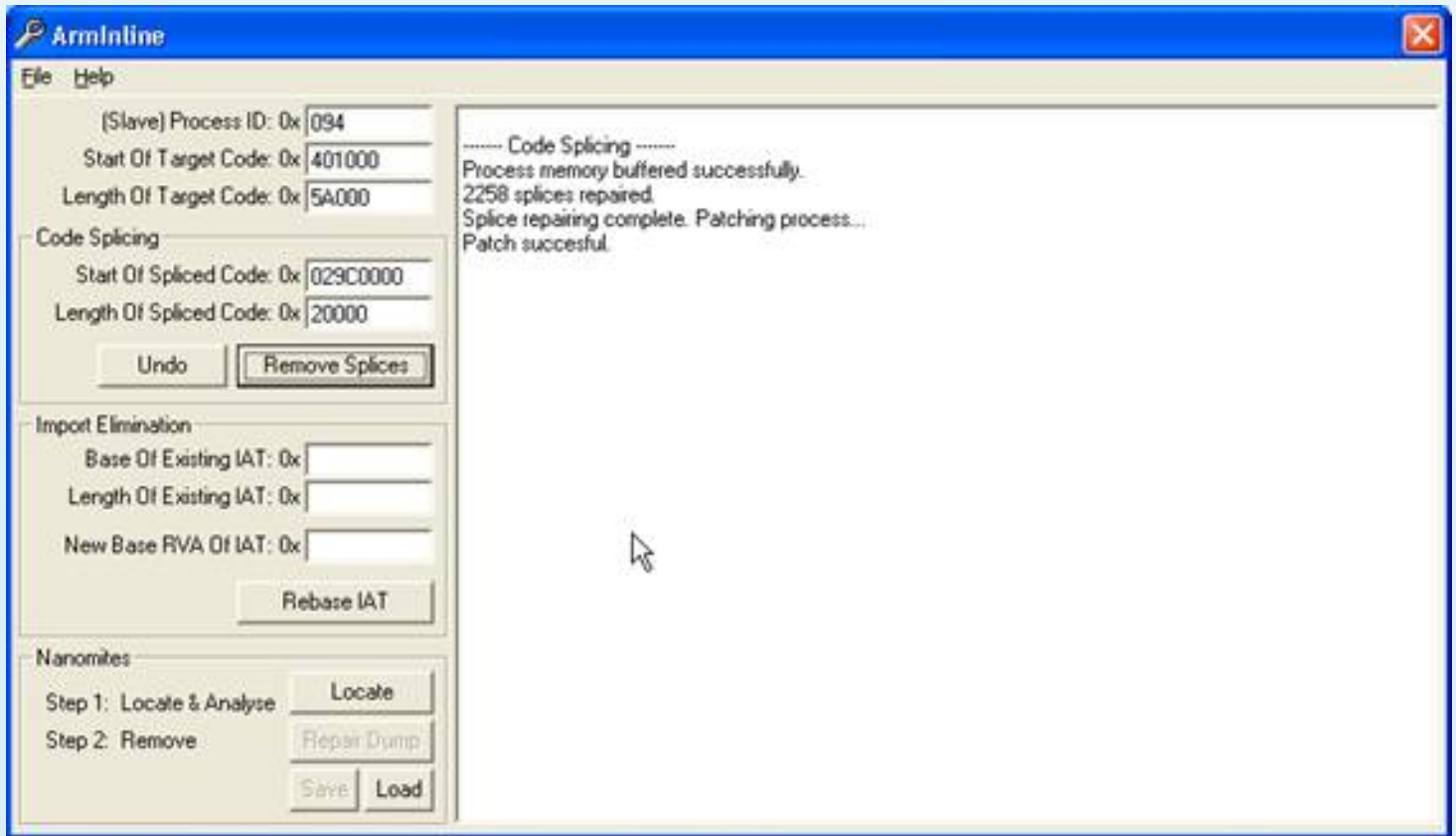
_Nhan Call EDI in F2, F9, F7: OEP!

Address	Hex dump	Disassembly	Comment
00401000	EB 10	JMP SHORT AntiTrac.00401012	
00401002	66:623A	BOUND DI,DWORD PTR DS:[EDX]	
00401005	43	INC EBX	
00401006	2B2B	SUB EBX,DWORD PTR DS:[EBX]	
00401008	48	DEC EAX	
00401009	4F	DEC EDI	
0040100A	4F	DEC EDI	
0040100B	4B	DEC EBX	
0040100C	90	NOP	
0040100D	E9 B8B64500	JMP 0085C6CA	
00401012	A1 ABB64500	MOV EAX,DWORD PTR DS:[45B6AB]	
00401017	C1E0 02	SHL EAX,2	
0040101A	A3 AFB64500	MOV DWORD PTR DS:[45B6AF],EAX	
0040101F	52	PUSH EDX	
00401020	6A 00	PUSH 0	
00401022	E8 F3990500	CALL AntiTrac.0045AA1A	JMP to kernel32.GetModuleHandle
00401027	8BD0	MOV EDX,EAX	
00401029	E8 162F0400	CALL AntiTrac.00443F44	
0040102E	5A	POP EDX	
0040102F	E8 E29B0500	CALL AntiTrac.0045AC16	JMP to CC3250MT.__CRTL_MEM_Us
00401034	E8 4F2F0400	CALL AntiTrac.00443F88	
00401039	6A 00	PUSH 0	
0040103B	E8 88300400	CALL AntiTrac.004440C8	

_Ta Need to find signs of Splicing Code, Alt + M:

Address	Size	Owner	Section	Contains	Type	Ac
00482000	00001000	AntiTrac	.rdata		Image	R
00483000	00009000	AntiTrac	.idata		Image	R
0048C000	00081000	AntiTrac	.edata	exports	Image	R
00500000	00008000	AntiTrac	.reloc		Image	R
00515000	00040000	AntiTrac	.text1	code	Image	R
00555000	00010000	AntiTrac	.adata		Image	R
00565000	00010000	AntiTrac	.data1	data,import	Image	R
00575000	00010000	AntiTrac	.reloc1	relocations	Image	R
00585000	00070000	AntiTrac	.pdata		Image	R
005F5000	00080000	AntiTrac	.rsrc	resources	Image	R
00680000	00006000				Map	R
00740000	00002000				Map	R
00750000	00103000				Map	R
00860000	000AE000				Map	R
00B60000	00001000				Priv RW	
00C60000	0000C000				Priv RW	
00C70000	00002000				Map	R
00C80000	0001A000				Priv RW	
00CA0000	000E7000				Priv RW	
00D64000	00002000				Priv RW	
00D80000	00006000				Priv RW	
00D90000	00003000				Priv RW	
00DD0000	00001000				Map	RW
00DE0000	00001000				Priv RW	
00FE0000	00001000				Priv RW	
00FF0000	00003000				Priv RW	
01000000	00010000				Priv RW	
01400000	00003000				Priv RW	
01440000	00003000				Priv RW	
01450000	00001000	BORLNDMM		PE header	Image	R
01451000	00004000	BORLNDMM	CODE	code	Image	R
01455000	00001000	BORLNDMM	DATA	data	Image	R
01456000	00001000	BORLNDMM	BSS		Image	R
01457000	00001000	BORLNDMM	.idata	imports	Image	R
01458000	00001000	BORLNDMM	.edata	exports	Image	R
01459000	00001000	BORLNDMM	.reloc	relocations	Image	R
0145A000	00001000	BORLNDMM	.rsrc	resources	Image	R
01460000	00002000				Map	R
01470000	00001000				Priv RW	
014F0000	00004000				Priv RW	
015F0000	00004000				Priv RW	
01790000	00002000				Map	R
01890000	00003000				Priv RW	
029C0000	00020000				Priv RW	
32500000	00001000	CC3250MT		PE header	Image	R
32501000	00084000	CC3250MT	.text	code	Image	R
32585000	0002C000	CC3250MT	.data	data	Image	R
325B1000	00003000	CC3250MT	.tls		Image	R
325B4000	00003000	CC3250MT	.idata	imports	Image	R

_Mo ArmInline and complete information as follows:



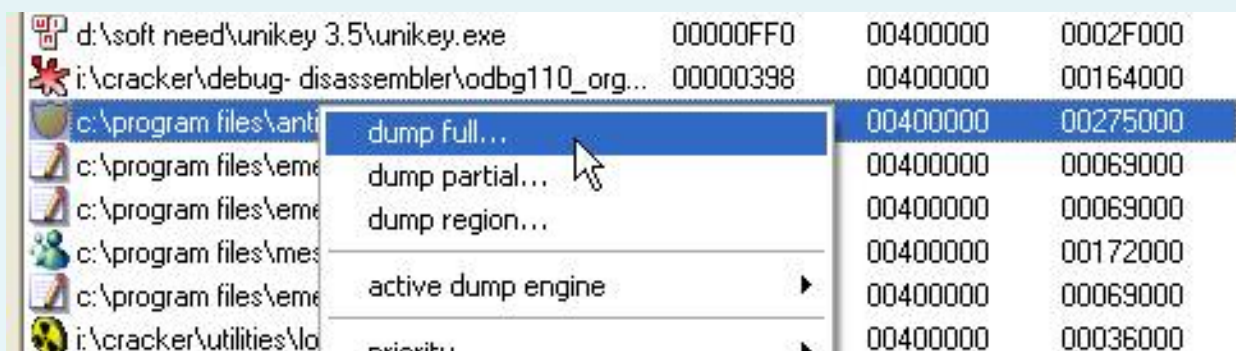
_Code Splicing die! We need to find the IAT. In 00401022, you follow in dump will find information IAT as follows:

IAT Start: 004837FC 4000A314 [VCL50. System @ @ initialization \\$ qqrv](#)

IAT End: 00484FE8 7621F6C6 WININET.FindNextUrlCacheEntryA

IAT Len: 000017EC

Full _Dung LordPE dump:

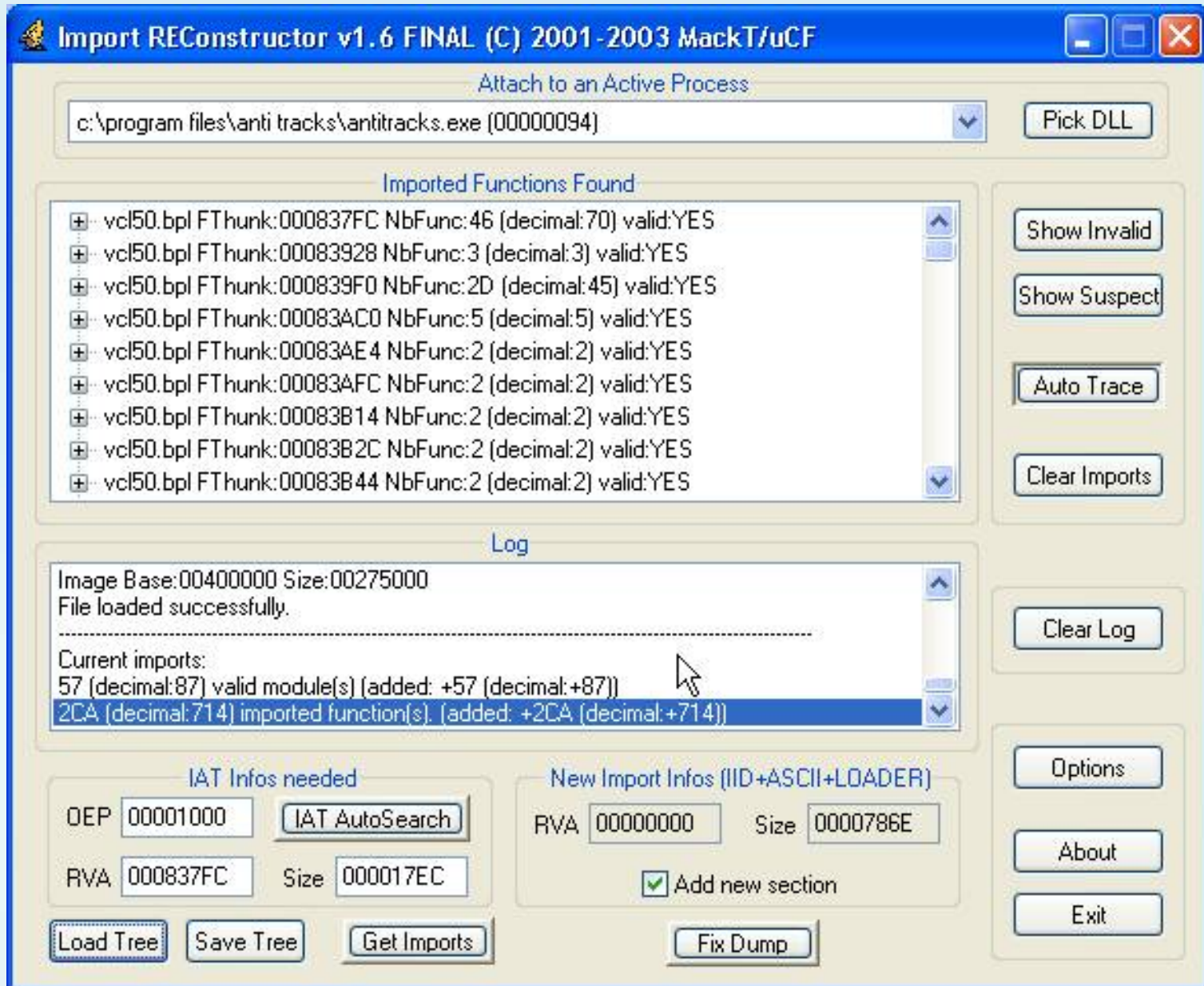


_Mo ImpREC, enter the following information:

OEP: 00001000

IATRVA: 000837FC

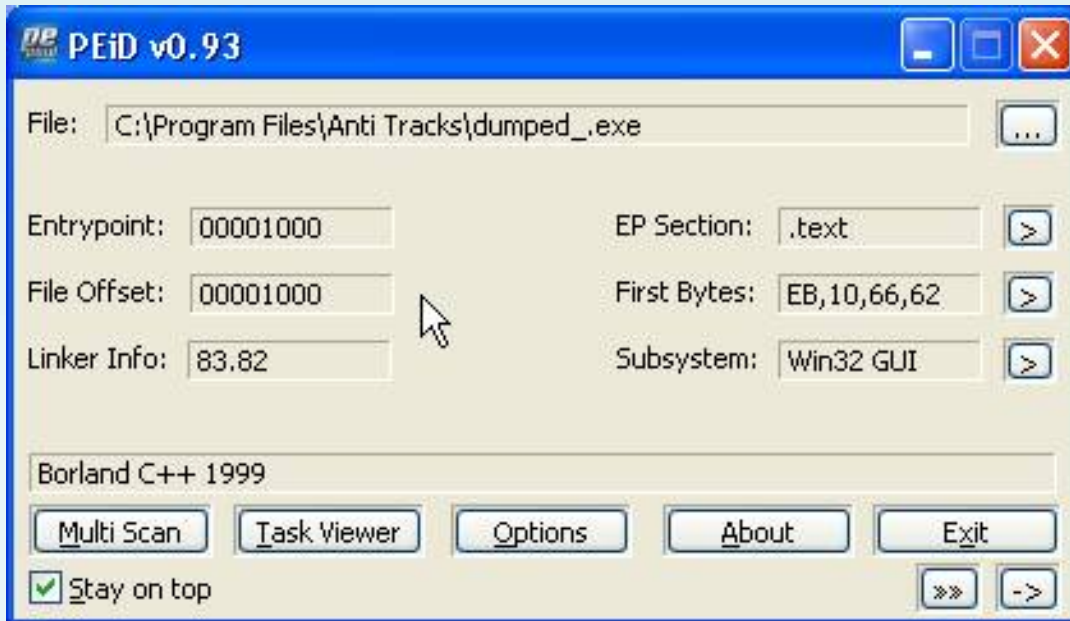
IATSize: 000017EC



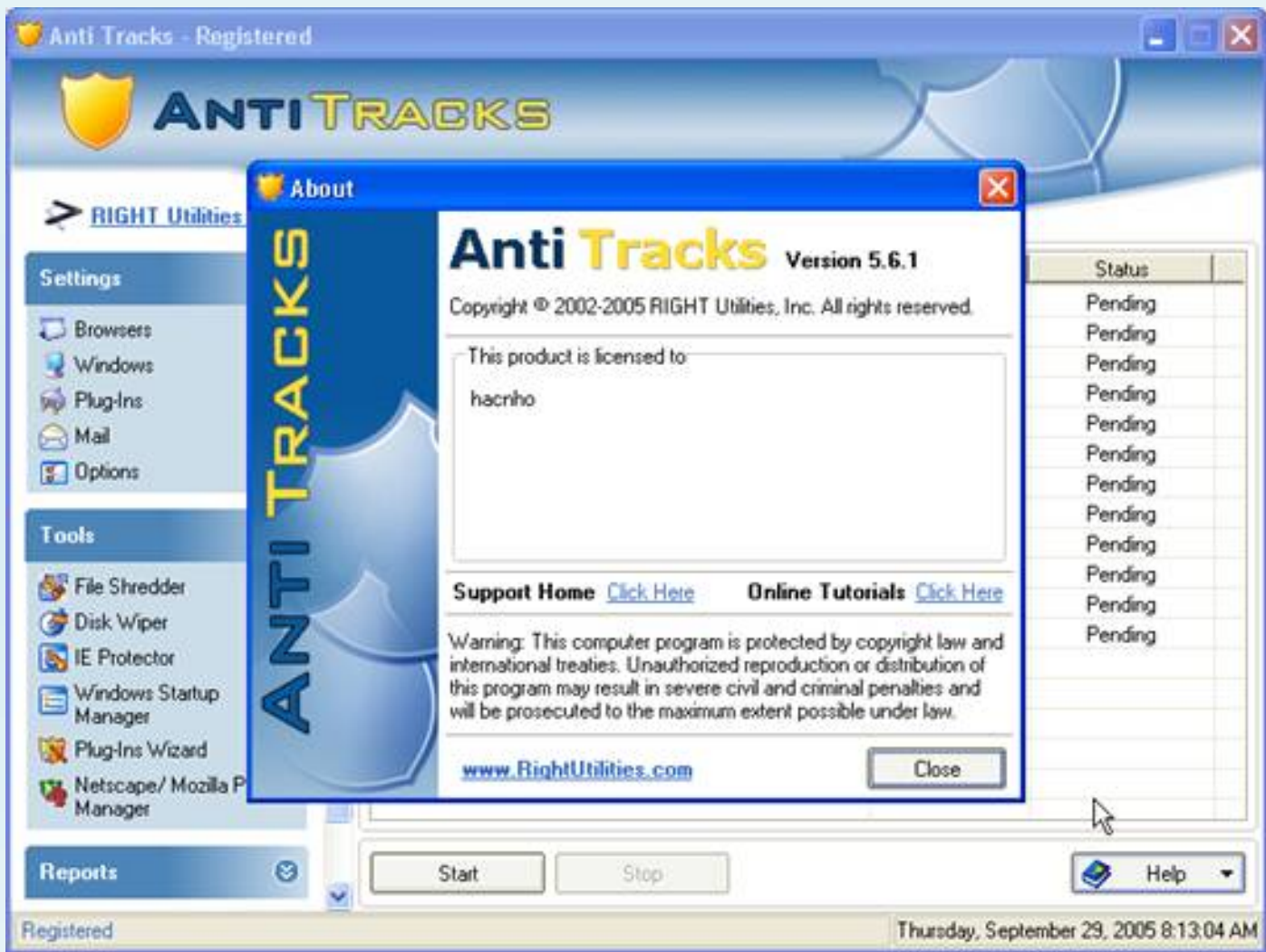
_Fix Dump:



_Detect With PeiD 0.93:



Run Try dumped.exe:



_Unpacked SuccessFul!

_Cracking:

User: hacnho

Serial: 071E-A123-AF7E-4850-ABC1-A3E5-0BF4-BBBD-4C18-A0A3-2498-7720-DCDB-E967

I. Conclusion

_For More tuts, please visit <http://tinicat.de/hacnho>

_Bye!

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtlm, hosiminh, Nilrem, fly, Madman_Hercules, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RiF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx OllyDBG of the authors and ArmInline

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 28/09/2005)



Armadillo & Macromedia Games

Contents Table:

I-Introduction

II Basic Protection

III-Advanced Protection:

IV-Ending

Tools: Hide OllyDBG + plugins, LordPE, ImportREC 1.6, CFF Explorer

Skill Request: Basic Knowledge Using Olly + manual unpacking Armadillo.

I-Introduction:

Hi everybody, who are as like reading this document, I'm very interested in playing with Armadillo. I know that many had for this Tuts packer and greatly appreciate to Authors who wrote those tuts, they're so cool! Today, I just write this doc to share some my small knowledge. Maybe I'm still a Newbie, hope it useful for someone 🙏 . Ok, let's start!

II-Basic-Protection:

In this part, we'll defeat from OberonGames.com games, for example, I choose "**A Series of Unfortunate Events**," you can download it from:

http://www.oberongames.com/exe/unfortunate_events-setup.exe?RefId=&origin=pgame_dl_u&ext=unfortunate_events-setup.exe

OberonMedia always have two files to run game. First is a file **launcher**, the shortcut is created on Desktop (after installed) to run it:



If you click **"Play Demo Now"**, this second launcher will load the file (or **main file**) to run game. But you can run the second without launcher so you just defeat the main. Of course, both files use a same protection is Armadillo - Trial expired and if you will get over time limit.

Load *"Unfortunate.exe"* into Olly, we stop at EP:

File View Debug Plugins Options Window Help Custom Cracked BP LP VB				
Address	Hex	dump	Disassembly	Comment
004D1000	60		PUSHAD	
004D1001	E8 00000000		CALL Unfortun.004D1006	
004D1006	5D		POP EBP	
004D1007	50		PUSH EAX	
004D1008	51		PUSH ECX	
004D1009	0FCA		BSWAP EDX	
004D100A	F7D2		NOT EDX	

Oberon only use Armadillo - Standard Protection to protect their games. I will not explain more detail about this protection (as How to Bypass Anti-Debug?, Why check Ignore All Exceptions ...) because you can find a lot of

good tuts on other Team as ARTeam, snd, CrackLatinos ... And for unpacking - Standard protection, we should use Scripts to make it faster.

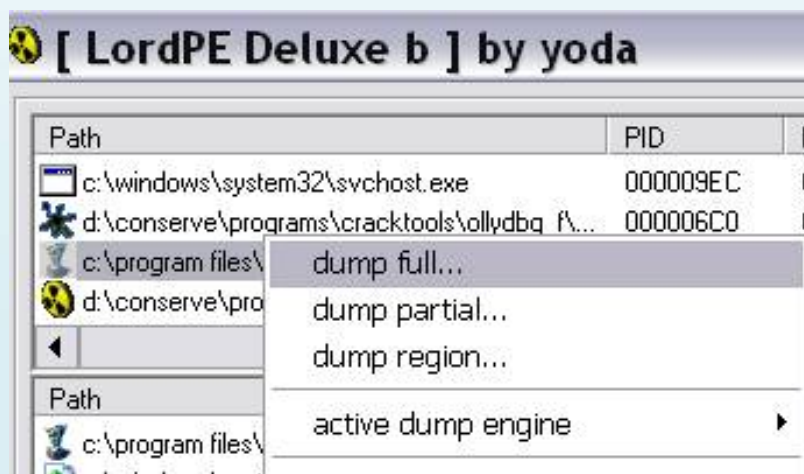
I recommend fly's script: [Armadillo V4.0-V4.4.Standard.Protection.osc](#), It's a one of best scripts to defeat armadillo-standard. Thanks guy for his hard work.

OK, run the script and fly's Stop at OEP:

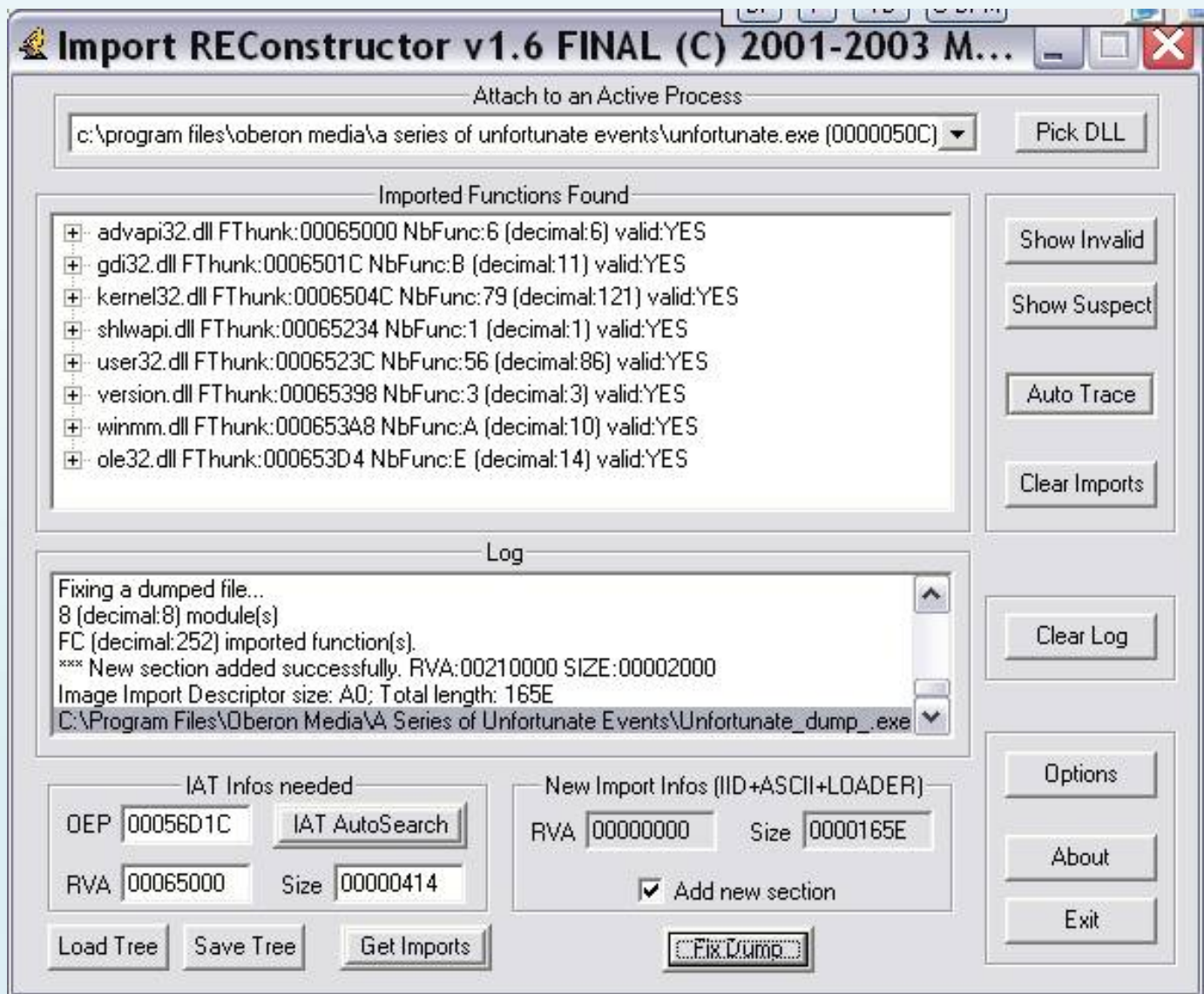


Address	Hex dump	Disassembly	Comment
00456D1C	6A 60	PUSH 60	This is the OEP!
00456D1E	68 40A64600	PUSH Unfortun.0046A640	
00456D23	E8 28E8FFFF	CALL Unfortun.00455550	
00456D28	BF 94000000	MOV EDI,94	
00456D2D	8BC7	MOV EAX,EDI	
00456D2F	E8 7CE9FFFF	CALL Unfortun.004556B0	
00456D34	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00456D37	8BF4	MOV ESI,ESP	
00456D39	893E	MOV DWORD PTR DS:[ESI],EDI	
00456D3B	56	PUSH ESI	
00456D3C	FF15 18524600	CALL DWORD PTR DS:[465218]	kernel32.GetVersio
00456D42	8B4E 10	MOV ECX,DWORD PTR DS:[ESI+10]	
00456D45	89D0 4C9C4800	MOV DWORD PTR DS:[489C4C],ECX	

Open LordPE to dump it:



Then fix IAT ImportREC by:



Reduce the file size by CFF Explorer (Delete useless sections), and do not forget BaseOfCode fixing:

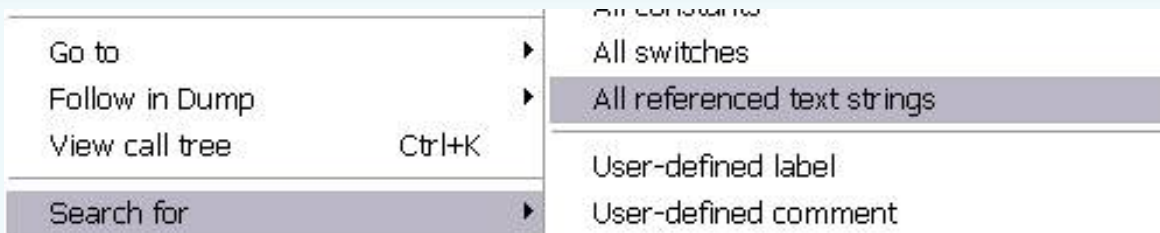
Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address
Byte[8]	Dword	Dword	Dword	Dword
.text	00063B54	00001000	00063B54	00001000
.rdata	0000E45E	00065000	0000E45E	00065000
.data	0002CA48	00074000	0002CA48	00074000
.text1	00030000	000A1000	00030000	000A1000
.adata	00010000	000D1000	00010000	000D1000
.data1	00020000	000E1000	00020000	000E1000
.pdata	00080000	00101000	00080000	00101000
.rsrc	0008F000	00181000	0008F000	00181000
.mact	00002000	00210000	00002000	00210000

AddressOfEntryPoint	00000128	Dword	00056D1C
BaseOfCode	0000012C	Dword	00001000
BaseOfData	00000130	Dword	000E1000

Save the file. But after that, run the file and it show a error message:



Do not afraid of that. We unpacked well. It's just a custom protection. OK, load the file into fixed Olly, stop at OEP, Search String: (Note: If you miss fixing BaseOfCode, you can not find exactly strings in Olly.)



Type:



And go to here:

```

00400031 PUSH Unfortun.004662FC      ASCII "Release"
00400036 PUSH Unfortun.004661F0      ASCII "-----"
0040E329 MOV ESI,Unfortun.00466390  ASCII "VERIFY failure: %s"
0040E330 MOV ESI,Unfortun.0046637C  ASCII "CHECK failure: %s"
0040E368 MOV EAX,Unfortun.0046636C  ASCII "VERIFY failure"
0040E36F MOV EAX,Unfortun.0046635C  ASCII "CHECK failure"
0040E3D0 PUSH Unfortun.004663A4      ASCII "%s: code = 0x%08X, addr = 0x%08X"
0040E3F5 PUSH Unfortun.004661F0

```


Enter it, and stop at:

Address	Hex dump	Disassembly	Comment
0040E2EB	C3	PUSH EBP	Start Function
0040E2EC	55	MOV EBP,ESP	
0040E2ED	8BEC	PUSH ECX	
0040E2EF	51	PUSH EBX	
0040E2F0	53	XOR EBX,EBX	
0040E2F1	330B	CMP BYTE PTR SS:[EBP+14],BL	
0040E2F3	385D 14	JNZ Unfortun.0040E385	
0040E2F6	0F85 89000000	CMP BYTE PTR SS:[EBP+8],BL	
0040E2FC	385D 08	JE SHORT Unfortun.0040E320	
0040E2FF	74 1F	MOV EAX,DWORD PTR DS:[474130]	
0040E301	A1 30414700	INC DWORD PTR DS:[EAX]	
0040E306	FF00	CALL DWORD PTR DS:[<%kernel32.GetLastError	C GetLastError
0040E308	FF15 F0514600	CMP EAX,EBX	
0040E30E	3BC3	JE SHORT Unfortun.0040E31B	
0040E310	74 09	MOV ECX,DWORD PTR DS:[474130]	Unfortun.00474410
0040E312	8B0D 30414700	MOV DWORD PTR DS:[ECX+1C],EAX	
0040E318	8941 1C	CALL Unfortun.00402CC2	
0040E31B	E8 A249FFFF	CMP DWORD PTR SS:[EBP+18],EBX	
0040E320	395D 18	JE SHORT Unfortun.0040E365	
0040E323	74 40	CMP BYTE PTR SS:[EBP+8],BL	
0040E325	385D 08	PUSH ESI	
0040E328	56	MOV ESI,Unfortun.00466390	ASCII "VERIFY failure: %
0040E329	BE 90634600	JNZ SHORT Unfortun.0040E335	
0040E32E	75 05	MOV ESI,Unfortun.0046637C	ASCII "CHECK failure: %s
0040E330	BE 7C634600	PUSH DWORD PTR SS:[EBP+1C]	
0040E335	74 1C	LEA EAX,DWORD PTR SS:[EBP-4]	
0040E338	8D45 FC	PUSH DWORD PTR SS:[EBP+18]	
0040E33B	FF75 18	PUSH EAX	
0040E33E	50	CALL Unfortun.0040D010	
0040E33F	E8 CCECFFFF	PUSH DWORD PTR DS:[EAX]	Arg5
0040E344	FF30	PUSH ESI	Arg4
0040E346	56	PUSH DWORD PTR SS:[EBP+10]	Arg3
0040E347	FF75 10	PUSH DWORD PTR SS:[EBP+C]	Arg2
0040E34A	FF75 0C	PUSH 1	Arg1 = 00000001
0040E34D	6A 01		

Scroll up a bit, set Start BP on this function. When running, it will break on here many times, and I think this is a CHECK verify file. OK, Scroll down and find this command:

Address	Hex dump	Disassembly	Comment
0040E338	8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]	
0040E33B	FF75 18	PUSH DWORD PTR SS:[EBP+18]	
0040E33E	50	PUSH EAX	
0040E33F	E8 CCECFFFF	CALL Unfortun.0040D010	
0040E344	FF30	PUSH DWORD PTR DS:[EAX]	Arg5
0040E346	56	PUSH ESI	Arg4
0040E347	FF75 10	PUSH DWORD PTR SS:[EBP+10]	Arg3
0040E34A	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	Arg2
0040E34D	6A 01	PUSH 1	Arg1 = 00000001
0040E34F	E8 E2FDFFFF	CALL Unfortun.0040E136	Unfortun.0040E136
0040E354	8B4D FC	MOV ECX,DWORD PTR SS:[EBP-4]	
0040E357	83C4 20	ADD ESP,20	
0040E35A	83C1 F0	ADD ECX,-10	
0040E35D	E8 2440FFFF	CALL Unfortun.00402386	
0040E362	5E	POP ESI	
0040E363	EB 20	JNZ SHORT Unfortun.0040E385	
0040E365	385D 08	CMP BYTE PTR SS:[EBP+8],BL	
0040E368	B8 6C634600	MOV EAX,Unfortun.0046636C	ASCII "VERIFY failure"
0040E36D	75 05	JNZ SHORT Unfortun.0040E374	
0040E36F	B8 5C634600	MOV EAX,Unfortun.0046635C	ASCII "CHECK failure"
0040E374	50	PUSH EAX	Arg4
0040E375	FF75 10	PUSH DWORD PTR SS:[EBP+10]	Arg3
0040E378	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	Arg2
0040E37B	6A 01	PUSH 1	Arg1 = 00000001
0040E37D	E8 B4FDFFFF	CALL Unfortun.0040E136	Unfortun.0040E136
0040E382	83C4 10	ADD ESP,10	
0040E385	8A45 14	MOV AL,BYTE PTR SS:[EBP+14]	
0040E388	5B	POP EBX	
0040E389	C9	LEAVE	
0040E38A	C3	RET	

If this function return AL == 0, it will show above error message. So it simple to change:



Then Copy to executable -> Select and Save the file. Now run it, game will start without any NAG or Time Limit.

Good, you did it! This method can be used to defeat many games on Oberon. But it seem not make us excited. Let's go to next part to find more fun!

III-Advanced Protection:

This type protection is used for some games on Oberon and PlayFirst. For example, I choose **Inspector Parker** at Oberon. But now, it's not available on there. You can download it from:

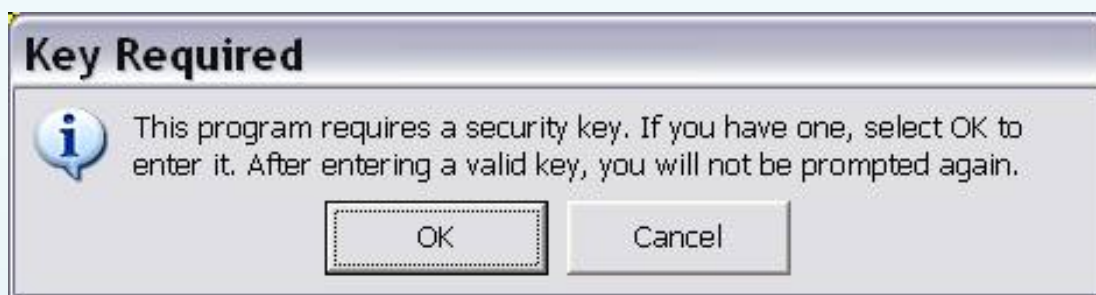
<http://download.shockwave.com/pub/inspectorparker/InstallInspectorParker.exe>

*Note: This is not not Game: **Inspector Parker-Betrapped**. Do not mistake that.*

Parker also Inspector use 2 files to run game. But main file is not same folder with a launcher. It's here: (Default)

C: \ Program Files \ Inspector Parker \ product

We run main file, and it show key request message:





A dialog box titled "Enter Key" with a close button (X) in the top right corner. The text inside says "Enter the registration name and key below, exactly as given to you." Below this text are two input fields: "Name:" and "Key:". At the bottom are "OK" and "Cancel" buttons.

As I known we can not unpack Armadillo app without a key if that request it before running. So close it and uninstall game now! Hehehe, just a joke. We'll open the launcher:



When you click "**Play**", launcher load main file and run game without key request message. Thinking a bit, we see that a trial launcher generated key and import to registry (expired after one runtime), after that, main file can

run without NAG. OK, we have two way to unpack in this case:

1. Leech trial launcher from key when it has generated key. Use it when main file ask and unpack it.
2. When launcher have finished creating and key trial before load main file, we stop it and unpack the main.

A-Fishing Trial Key:

With this game, I'll use first method. Now, load launcher (*Inspector Parker.exe*) into Olly:

Address	Hex	dump	Disassembly	Comment
00477000	60		PUSHAD	
00477001	E8	00000000	CALL Inspecto.00477006	
00477006	5D		POP EBP	
00477007	50		PUSH EAX	
00477008	51		PUSH ECX	
00477009	EB 0F		ATP SHORT Inspecto.0047701A	
0047700B	B9		DB BS	
0047700C	EB 0F		ATP SHORT Inspecto.0047701D	
0047700E	B8 EB07B9EB		MOV EAX,EBB907EB	
00477013	0F90EB		SETQ BL	

Hit Shift-F9 to run it, show the menu launcher same pic above. Come back Olly, Open Memory Map:

00370000	00001000			Priv	RW	RW	
00380000	00001000			Priv	RW	RW	
00390000	00004000			Priv	RW	RW	
003A0000	00049000			Map	RW	RW	
003F0000	0000E000			Priv	RW	RW	
00400000	00001000			Imag	R	RWE	
401000	00026000	Inspecto	.text	Imag	R	RWE	Main code is here.
00427000	00000000	Inspecto	.data	Imag	R	RWE	
00430000	00017000	Inspecto	.data	Imag	R	RWE	
00447000	00030000	Inspecto	.text1	Imag	R	RWE	
00477000	00010000	Inspecto	.adata	Imag	R	RWE	
00487000	00020000	Inspecto	.data1	Imag	R	RWE	
004A7000	00040000	Inspecto	.pdata	Imag	R	RWE	
004E7000	0000A000	Inspecto	.rsrc	Imag	R	RWE	
00500000	00009000			Map	R E	R E	
005C0000	00002000			Map	R E	R E	
005D0000	00103000			Map	R	R	
006E0000	0010A000			Map	R E	R E	
009E0000	00001000			Priv	RW	RW	

The launcher file is packed by Armadillo, after unpack code, main code is stored at section. Text. Ctrl-G 401,000 go to:



We're here:

Address	Hex dump	Disassembly	Comment
00401000	8B4424 08	MOV EAX,DWORD PTR SS:[ESP+8]	
00401004	56	PUSH ESI	
00401005	3D 00040000	CMP EAX,400	
0040100A	74 1A	JE SHORT Inspecto.00401026	
0040100C	8B4C24 14	MOV ECX,DWORD PTR SS:[ESP+14]	
00401010	8B5424 10	MOV EDX,DWORD PTR SS:[ESP+10]	
00401014	51	PUSH ECX	
00401015	52	PUSH EDX	
00401016	50	PUSH EAX	
00401017	8B4424 14	MOV EAX,DWORD PTR SS:[ESP+14]	
00401018	50	PUSH EAX	
0040101C	FF15 D0744200	CALL DWORD PTR DS:[4274D0]	USER32.DefWindowProcA
00401022	5E	POP ESI	

Right-click - Search:

Comment	;	
Breakpoint		All intermodular calls
Run trace		All commands
		All sequences
New origin here	Ctrl+Gray *	All constants
Go to		All switches
Thread		All referenced text strings
Follow in Dump		User-defined label
Search for		User-defined comment

Click "**Destination**" to sort list:

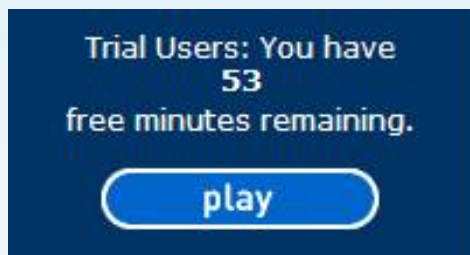
Address	Disassembly	Destination
00401000	MOV EAX,DWORD PTR SS:[ESP+8]	(Initial CPU selection)
004011D6	CALL DWORD PTR DS:[427170]	DS:[00427170]=00AEA101
00401296	CALL DWORD PTR DS:[427174]	DS:[00427174]=00AEA881
0040131F	CALL DWORD PTR DS:[427178]	DS:[00427178]=00AEA8A1
0040139F	CALL DWORD PTR DS:[43460C]	DS:[0043460C]=00B058B0
00401433	CALL DWORD PTR DS:[434614]	DS:[00434614]=00B0580C
00401696	CALL DWORD PTR DS:[434620]	DS:[00434620]=00B056E2
0040172F	CALL DWORD PTR DS:[434610]	DS:[00434610]=00B058EA
00401735	CALL DWORD PTR DS:[434620]	DS:[00434620]=00B056E2
004017A7	CALL DWORD PTR DS:[43460C]	DS:[0043460C]=00B058B0
004017B9	CALL DWORD PTR DS:[43460C]	DS:[0043460C]=00B058B0
00401939	CALL DWORD PTR DS:[434610]	DS:[00434610]=00B058EA
00401A05	CALL DWORD PTR DS:[42717C]	DS:[0042717C]=00AEA90E
00401AED	CALL DWORD PTR DS:[434610]	DS:[00434610]=00B058EA
00401AF3	CALL DWORD PTR DS:[434620]	DS:[00434620]=00B056E2
00402014	CALL DWORD PTR DS:[427178]	DS:[00427178]=00AEA8A1
0040203D	CALL DWORD PTR DS:[43461C]	DS:[0043461C]=00B058A5
00402043	CALL DWORD PTR DS:[434620]	DS:[00434620]=00B056E2
00402207	CALL DWORD PTR DS:[434614]	DS:[00434614]=00B0580C
0040221E	CALL DWORD PTR DS:[434614]	DS:[00434614]=00B0580C
0040222B	CALL DWORD PTR DS:[434614]	DS:[00434614]=00B0580C
004024E4	CALL DWORD PTR DS:[434620]	DS:[00434620]=00B056E2
00402511	CALL DWORD PTR DS:[434624]	DS:[00434624]=00B058BC
0040252D	CALL DWORD PTR DS:[427178]	DS:[00427178]=00AEA8A1
00402544	CALL DWORD PTR DS:[43461C]	DS:[0043461C]=00B058A5
0040254A	CALL DWORD PTR DS:[434620]	DS:[00434620]=00B056E2
0040274D	CALL DWORD PTR DS:[434618]	DS:[00434618]=00B05A73
00402E33	CALL DWORD PTR DS:[427170]	DS:[00427170]=00AEA101
00402F3D	CALL DWORD PTR DS:[427174]	DS:[00427174]=00AEA881
00402F7D	CALL DWORD PTR DS:[427174]	DS:[00427174]=00AEA881
00403849	CALL DWORD PTR DS:[427488]	DS:[00427488]=00AED129
004039EE	CALL DWORD PTR DS:[427488]	DS:[00427488]=00AED129
00403A39	CALL DWORD PTR DS:[427488]	DS:[00427488]=00AED129
00403C5D	CALL DWORD PTR DS:[427170]	DS:[00427170]=00AEA101
00403C6E	CALL DWORD PTR DS:[42716C]	DS:[0042716C]=00AE971D
0040404E	CALL DWORD PTR DS:[42700C]	DS:[0042700C]=00AED94D
004042E6	CALL DWORD PTR DS:[4271CC]	DS:[004271CC]=00AED0B0
004055B3	CALL DWORD PTR DS:[42729C]	DS:[0042729C]=00AED0EB
00405635	CALL DWORD PTR DS:[42718C]	DS:[0042718C]=00AE50B6

We see that many Call to addresses in a section out of range file. This section are created to decrypt code for

Armadillo. And I think it has some key functions generate trial from there. OK, we set all those BP CALL: (hehe, F2 hit more and more ..)

00401000	MOV EAX,DWORD PTR SS:[ESP+0]	(Initial CPU selection)
00401106	CALL DWORD PTR DS:[427170]	DS:[00427170]=00AEA101
00401296	CALL DWORD PTR DS:[427174]	DS:[00427174]=00AEA881
0040131F	CALL DWORD PTR DS:[427178]	DS:[00427178]=00AEA8A1
0040139F	CALL DWORD PTR DS:[43460C]	DS:[0043460C]=00B058B0
00401433	CALL DWORD PTR DS:[434614]	DS:[00434614]=00B0580C
00401696	CALL DWORD PTR DS:[434620]	DS:[00434620]=00B056E2
0040172F	CALL DWORD PTR DS:[434610]	DS:[00434610]=00B058EA
00401735	CALL DWORD PTR DS:[434620]	DS:[00434620]=00B056E2
004017A7	CALL DWORD PTR DS:[43460C]	DS:[0043460C]=00B058B0
004017B9	CALL DWORD PTR DS:[43460C]	DS:[0043460C]=00B058B0
00401939	CALL DWORD PTR DS:[434610]	DS:[00434610]=00B058EA
00401A05	CALL DWORD PTR DS:[42717C]	DS:[0042717C]=00AEA00E
00401AED	CALL DWORD PTR DS:[434610]	DS:[00434610]=00B058EA
00401AF3	CALL DWORD PTR DS:[434620]	DS:[00434620]=00B056E2
00402014	CALL DWORD PTR DS:[427178]	DS:[00427178]=00AEA8A1
0040203D	CALL DWORD PTR DS:[43461C]	DS:[0043461C]=00B058A5
00402043	CALL DWORD PTR DS:[434620]	DS:[00434620]=00B056E2
00402207	CALL DWORD PTR DS:[434614]	DS:[00434614]=00B0580C
0040221E	CALL DWORD PTR DS:[434614]	DS:[00434614]=00B0580C
0040222B	CALL DWORD PTR DS:[434614]	DS:[00434614]=00B0580C
004024E4	CALL DWORD PTR DS:[434620]	DS:[00434620]=00B056E2
00402511	CALL DWORD PTR DS:[434624]	DS:[00434624]=00B058BC
0040252D	CALL DWORD PTR DS:[427178]	DS:[00427178]=00AEA8A1
00402544	CALL DWORD PTR DS:[43461C]	DS:[0043461C]=00B058A5
0040254A	CALL DWORD PTR DS:[434620]	DS:[00434620]=00B056E2
0040274D	CALL DWORD PTR DS:[434618]	DS:[00434618]=00B05A73
00402E33	CALL DWORD PTR DS:[427170]	DS:[00427170]=00AEA101
00402F3D	CALL DWORD PTR DS:[427174]	DS:[00427174]=00AEA881
00402F7D	CALL DWORD PTR DS:[427174]	DS:[00427174]=00AEA881
00403849	CALL DWORD PTR DS:[427488]	DS:[00427488]=00AED129
004039EE	CALL DWORD PTR DS:[427488]	DS:[00427488]=00AED129
00403A39	CALL DWORD PTR DS:[427488]	DS:[00427488]=00AED129
00403C5D	CALL DWORD PTR DS:[427170]	DS:[00427170]=00AEA101
00403C6E	CALL DWORD PTR DS:[42716C]	DS:[0042716C]=00AE971D
0040404E	CALL DWORD PTR DS:[42700C]	DS:[0042700C]=00AED94D
004042E6	CALL DWORD PTR DS:[4271CC]	DS:[004271CC]=00AED0B0
004055B3	CALL DWORD PTR DS:[42729C]	DS:[0042729C]=00AED0EB
0040567E	CALL DWORD PTR DS:[4271BC]	DS:[004271BC]=00AEDB66
00405695	CALL DWORD PTR DS:[4271C0]	DS:[004271C0]=00AED0B0
004056B7	CALL DWORD PTR DS:[42729C]	DS:[0042729C]=00AED0EB

Why do I make that stupid? Because I do not know exactly which CALL can be fishing Key trial. But you'll see this result immediatelly. Switch to Shockwave menu and click "Play":



Olly break here:

Address	Hex dump	Disassembly
00402014	FF15 78714200	CALL DWORD PTR DS:[427178]
0040201A	A1 4C524300	MOV EAX,DWORD PTR DS:[43524C]
0040201F	40	INC EAX
00402020	50	PUSH EAX
00402021	6A 04	PUSH 4
00402023	A3 4C524300	MOV DWORD PTR DS:[43524C],EAX
00402028	E8 C3F3FFFF	CALL Inspecto.004013F0
0040202D	83C4 08	ADD ESP,8
00402030	8D8C24 0401000	LEA ECX,DWORD PTR SS:[ESP+104]
00402037	8D5424 04	LEA EDX,DWORD PTR SS:[ESP+4]
0040203B	51	PUSH ECX

Trace F8 to here: (if it at any other break when call tracing, and trace to continue RETN)

Address	Hex dump	Disassembly
00402013	50	PUSH EAX
00402014	FF15 78714200	CALL DWORD PTR DS:[427178]
0040201A	A1 4C524300	MOV EAX,DWORD PTR DS:[43524C]
0040201F	40	INC EAX
00402020	50	PUSH EAX
00402021	6A 04	PUSH 4
00402023	A3 4C524300	MOV DWORD PTR DS:[43524C],EAX
00402028	E8 C3F3FFFF	CALL Inspecto.004013F0
0040202D	83C4 08	ADD ESP,8
00402030	8D8C24 0401000	LEA ECX,DWORD PTR SS:[ESP+104]
00402037	8D5424 04	LEA EDX,DWORD PTR SS:[ESP+4]
0040203B	51	PUSH ECX
0040203C	52	PUSH EDX
0040203D	FF15 1C464300	CALL DWORD PTR DS:[43461C]
00402043	FF15 20464300	CALL DWORD PTR DS:[434620]
00402049	E8 82F2FFFF	CALL Inspecto.00401200

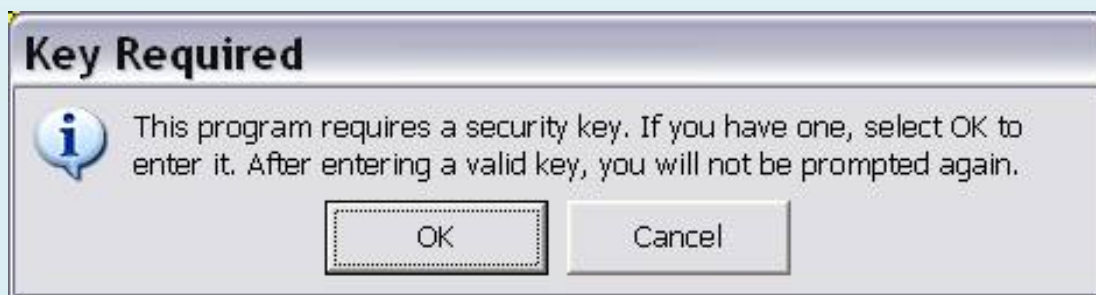
When stop at pic place as above, see Registers Window:

Registers (FPU)		
EAX	7FFDF001	
ECX	0012EBB8	ASCII "C525-FB60-DBFB-FA99-0546-D23C-7175-EB65-0108-3207-F646-B13F-85BE-8CDF-1F0E"
EDX	0012EAB8	ASCII "TRY2"
EBX	00000000	
ESP	0012EAB0	
EBP	0017E5C0	ASCII "try"
ESI	004305C8	ASCII "Error"
EDI	0012ECB8	
EIP	0040203C	Inspecto.0040203C
C 0	ES 0023	32bit 0(FFFFFFFF)
P 1	CS 001B	32bit 0(FFFFFFFF)
A 1	SS 0023	32bit 0(FFFFFFFF)
Z 0	DS 0023	32bit 0(FFFFFFFF)
S 0	FS 003B	32bit 7FFDF000(FFF)
T 0	GS 0000	NULL
D 0		
0 0	LastErr	ERROR_SUCCESS (00000000)

Hehe, EDX and ECX = Name = Trial Key. We'll use it mainly for file 🧐 . OK. Olly other Open, load main file (*Parker.exe*):

Address	Hex dump	Disassembly
2003F000	50	PUSHAD
2003F001	E8 00000000	CALL Parker.2003F006
2003F006	5D	POP EBP
2003F007	50	PUSH EAX
2003F008	51	PUSH ECX
2003F009	EB 0F	MOV SHORT Parker.2003F01A
2003F00B	B9	DB BS
2003F00C	EB 0F	MOV SHORT Parker.2003F01D
2003F00E	B8 E07B9EB	MOV EAX,E07B9EB

Use fly's script, it'll run until show message:



Which input data before you get:

Enter Key

Enter the registration name and key below, exactly as given to you.

Name:

Key:

OK Cancel

OK:


Key Valid

 Key is valid, and has been stored.

OK

Scripts will continue and finished at OEP:

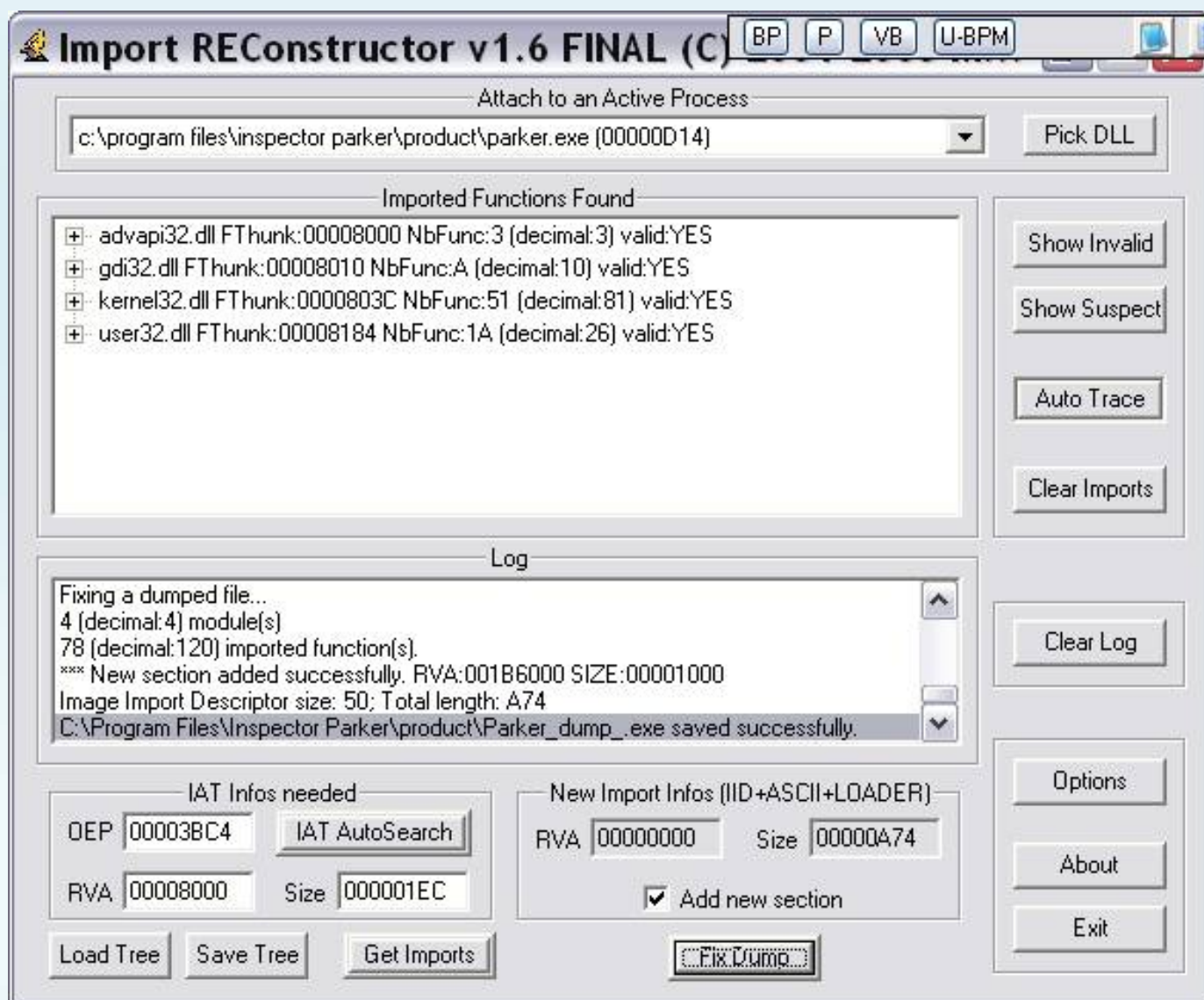
OllyScript

 Just : OEP ! Dump and Fix IAT. Good Luck

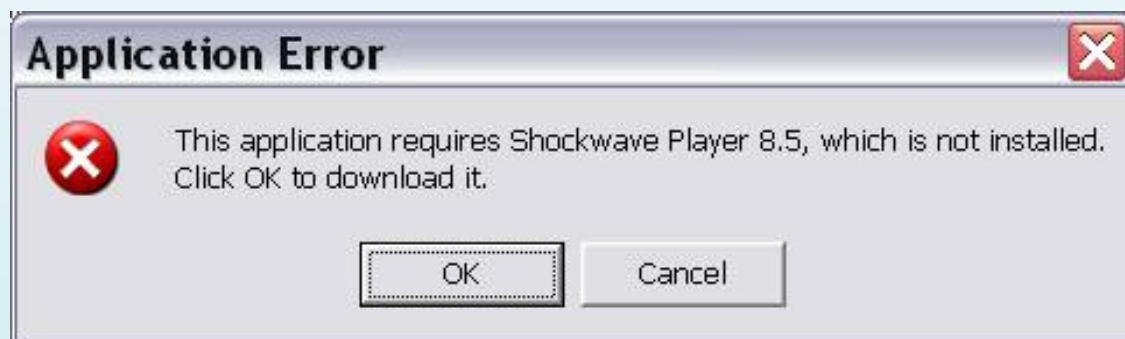
OK

Address	Hex dump	Disassembly	Comment
20003BC4	55	PUSH EBP	This is the OEP! Found
20003BC5	8BEC	MOV EBP,ESP	
20003BC7	6A FF	PUSH -1	
20003BC9	68 F0810020	PUSH Parker.200081F0	
20003BCE	68 204E0020	PUSH Parker.20004E20	
20003BD3	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
20003BD9	50	PUSH EAX	
20003BDA	64:8925 000000	MOV DWORD PTR FS:[0],ESP	
20003BE1	83EC 58	SUB ESP,58	
20003BE4	53	PUSH EBX	
20003BE5	56	PUSH ESI	
20003BE6	57	PUSH EDI	
20003BE7	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
20003BEA	FF15 10810020	CALL DWORD PTR DS:[20008110]	kernel32.GetVersion
20003BF0	33D2	XOR EDX,EDX	
20003BF2	8AD4	MOV DL,AH	
20003BF4	8915 2CCE0020	MOV DWORD PTR DS:[2000CE2C],EDX	
20003BFA	8BC8	MOV ECX,EAX	
20003BFC	81E1 FF000000	AND ECX,0FF	

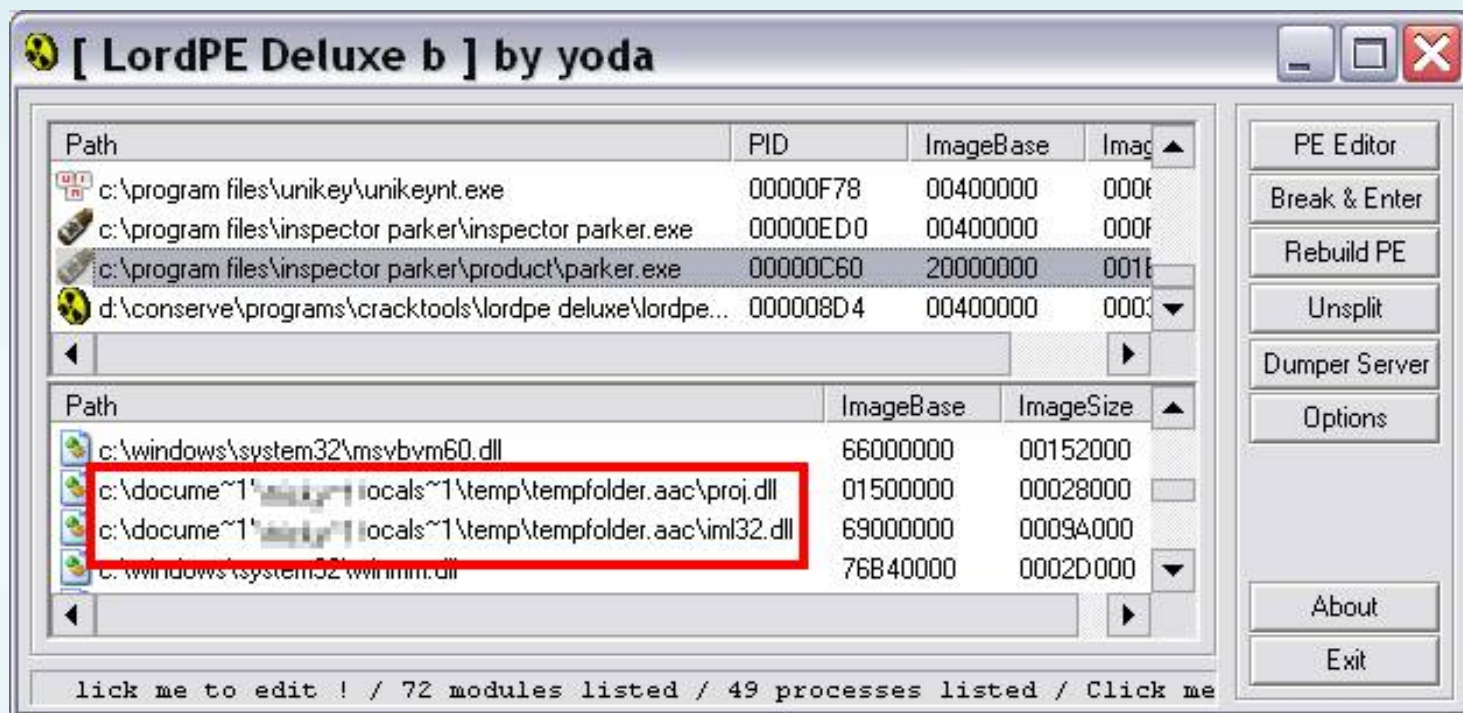
Dump it and fix IAT:



If you do not fix the script successfully, you should try fixing by manual MagicJump. Next step, all close, fixed file and run it show message:



Well, we miss some files when unpacking. Run launcher again and click **"Play"** to run main file. But do not Close Game. Hit Alt-Tab to Switch Out. LordPE open, view Module which are loaded by main file:



OK, we miss some file:

Name	Size
dirapi.dll	1,072 KB
iml32.dll	548 KB
Macromedia.lok	0 KB
msvcrt.dll	261 KB
proj.dll	156 KB

They are 4 modules of Macromedia Director. Copy all to same folder with a main file (do not need Macromedia.lok). After that, run the file again fixed and still show messages:





Hehe, that is the error message of Macromedia Director if you miss or Overlay Data Resource Change in something. In this case, after unpacking, we missed Overlay Data. I do not explain this, you should find some tuts about ActiveMark to undertand what is it.

I also note: Data Overlay of Macromedia Director does not like Overlay Data as of ActiveMark. It's really hard with me. I do not find it easily to paste into end of dump file. And when to write this doc, I still do not know how to find that. ***Hope have any body know about Overlay Director of Data and explain for me.***

But I know a way to fix that. See in main folder of files, you will find a file: "Parker.dcr". It'sa movie which is created by Macromedia Director. If you tried and learnt Director publish a file. Exe, you'll know that *. DCR files is like as Overlay Data. After publish. Exe files, you will not need it more. Hehe, the authors missed removing it. Now, create new *. ini file with Notepad by contents:

```
[Movies]
Movie01= Parker.dcr
```

This *. ini can be added more information but you just need that. Save the file with a name like as name of. Exe files.

Ex: If my file is fixed: *Parker_f.exe* my *. ini files will *Parker_f.ini* is. (Search more information about Director to understand it)

Okie, open the file again fixed, and games run well. Yeah! we solved a Game Director.

But with this method, you only do it if has just **one movie file** (and removing authors missed it). When it has more than one movie file (like as *Diner Dash* from PlayFirst game), you can not exactly add info to *. ini file. So you must know how to find Overlay Data.

Next part, we'll defeat a game on PlayFirst to go to End of this document. Relax a bit and GO GO GO!

B-Stop The launcher:

Now, our victim is **Diner Dash 2** (not Diner Dash 1) on PlayFirst. You can download it from:

http://www.playfirst.com:80/bits/Install_DinerDash2.EXE

It's similar as OberonGames, has two files: one launcher and one main file. Main file is installed at:

C: \ Program Files \ PlayFirst \ Diner Dash 2 \ game

The launcher also generate a Trial Key to load main file. But at this time, we'll stop it before main file is called.

OK, load launcher into Olly:

Address	Hex	dump	Disassembly	Comment
0044F2E3	55		PUSH EBP	
0044F2E4	8BEC		MOV EBP,ESP	
0044F2E6	6A FF		PUSH -1	
0044F2E8	68 209B4700		PUSH dinerdas.00479B20	
0044F2ED	68 20F04400		PUSH dinerdas.0044F020	SE handler installa
0044F2F2	64:A1 00000000		MOV EAX,DWORD PTR FS:[0]	
0044F2F8	50		PUSH EAX	
0044F2F9	64:8925 0000		MOV DWORD PTR FS:[0],ESP	
0044F300	83EC 58		SUB ESP,58	
0044F303	53		PUSH EBX	
0044F304	56		PUSH ESI	
0044F305	57		PUSH EDI	
0044F306	8965 E8		MOV DWORD PTR SS:[EBP-18],ESP	
0044F309	FF15 88414700		CALL DWORD PTR DS:[&KERNEL32.GetVersion	kernel32.GetVersion
0044F30F	33D2		XOR EDI,EDI	

Shift-F9 to run:






Diner Dash 2



Sling grits, earn tips - help Flo save her friends!

Visit PlayFirst.com for more games

Free Trial

Play the game for free during the trial period.

59 minutes, 10 seconds remaining.

PLAY

Buy

Buy Diner Dash 2 and get unlimited play!

BUY IT NOW

Unlock

After you have purchased the game, you can unlock it here.

Unlock key:

OR

Username
or Email:

Password:

UNLOCK

[HELP](#)
[TERMS](#)
[PRIVACY](#)
[close x](#)

Switch to Olly, open Memory Map:

003F0000	0000E000	dinerdas	.text	code	Image	RWE	
00401000	0001A000	dinerdas	.text	code	Image	RWE	
00421000	00003000	dinerdas	.data	data, import	Image	RWE	
00424000	00040000	dinerdas	.text1	code	Image	RWE	
00464000	00010000	dinerdas	.adata	resources	Image	RWE	
00474000	00020000	dinerdas	.data1	data, import	Image	RWE	
00494000	00040000	dinerdas	.pdata	resources	Image	RWE	
004D4000	00008000	dinerdas	.rsrc	resources	Image	RWE	
004E0000	00009000				Map	RWE	
005A0000	00002000				Map	RWE	
005B0000	00103000				Map	RWE	

Goto:



At here:

Address	Hex dump	Disassembly	Comment
00401000	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]	
00401004	6A 00	PUSH 0	
00401006	68 90B24100	PUSH dinerdas.0041B290	ASCII "Error"
00401008	50	PUSH EAX	
0040100C	6A 00	PUSH 0	
0040100E	FF15 14B24100	CALL DWORD PTR DS:[41B214]	
00401014	6A 00	PUSH 0	
00401016	E8 7CD00000	CALL dinerdas.0040E097	
00401018	CC	INT3	
0040101C	CC	INT3	

Right Click, Search:

Comment	;	
Breakpoint		▶ All intermodular calls
Run trace		▶ All commands
		▶ All sequences
New origin here	Ctrl+Gray *	▶ All constants
Go to		▶ All switches
Thread		▶ All referenced text strings
Follow in Dump		▶ User-defined label
Search for		▶ User-defined comment

Address	Disassembly	Destination
004096E9	CALL DWORD PTR DS:[41B0C0]	DS:[0041B0C0]=00AD6796
004097B8	CALL DWORD PTR DS:[41B0C8]	DS:[0041B0C8]=00AD67F0
004097C2	CALL DWORD PTR DS:[41B050]	DS:[0041B050]=00AD770E
0040C8BE	CALL DWORD PTR DS:[41B048]	DS:[0041B048]=00AD5996
0040DF41	CALL DWORD PTR DS:[41B188]	DS:[0041B188]=00AD8C70
0040DF51	CALL DWORD PTR DS:[41B044]	DS:[0041B044]=00AD4DB0
0040DF65	CALL DWORD PTR DS:[41B18C]	DS:[0041B18C]=00AD6692
0040DFF0	CALL DWORD PTR DS:[41B04C]	DS:[0041B04C]=00AD674B
004104C8	CALL DWORD PTR DS:[41B14C]	DS:[0041B14C]=00AD9759
00411C44	CALL DWORD PTR DS:[41B0AC]	DS:[0041B0AC]=00AD7188
00411D0E	CALL DWORD PTR DS:[41B0AC]	DS:[0041B0AC]=00AD7188
00412130	CALL DWORD PTR DS:[41B148]	DS:[0041B148]=00AD759A
0041223D	CALL DWORD PTR DS:[41B050]	DS:[0041B050]=00AD770E
004123AA	CALL DWORD PTR DS:[41B0B0]	DS:[0041B0B0]=00AD73C5
00412419	CALL DWORD PTR DS:[41B0B0]	DS:[0041B0B0]=00AD73C5
004126B5	CALL DWORD PTR DS:[41B138]	DS:[0041B138]=00AD778D
0041271C	CALL DWORD PTR DS:[41B138]	DS:[0041B138]=00AD778D
00415998	CALL DWORD PTR DS:[41B188]	DS:[0041B188]=00AD8C70
004159A8	CALL DWORD PTR DS:[41B044]	DS:[0041B044]=00AD4DB0
00415DA6	CALL DWORD PTR DS:[41B0B0]	DS:[0041B0B0]=00AD73C5
0041637B	CALL DWORD PTR DS:[41B0F0]	DS:[0041B0F0]=00AD62A6
00416498	CALL DWORD PTR DS:[422984]	DS:[00422984]=00AD1878
0041690E	CALL DWORD PTR DS:[41B058]	DS:[0041B058]=00AD975F
00416B87	CALL DWORD PTR DS:[41B18C]	DS:[0041B18C]=00AD6692
00416E51	CALL DWORD PTR DS:[41B148]	DS:[0041B148]=00AD759A
0041713D	CALL DWORD PTR DS:[41B024]	DS:[0041B024]=00AD6CFE
0041714A	CALL DWORD PTR DS:[41B138]	DS:[0041B138]=00AD778D
00417155	CALL DWORD PTR DS:[41B050]	DS:[0041B050]=00AD770E
0041814F	CALL DWORD PTR DS:[41B048]	DS:[0041B048]=00AD5996
0040C947	CALL DWORD PTR DS:[422620]	ATL.AtIAXGetControl
0040C6F8	CALL DWORD PTR DS:[41B1D4]	USER32.BeginPaint
004080D6	CALL DWORD PTR DS:[41B260]	ole32.CoCreateGuid
0040C8B3	CALL DWORD PTR DS:[41B25C]	ole32.CoInitialize

Do not set any F2 by BP, because it has Integrity Check. And as I said that we'll stop it before launcher call main file. Launcher use to call CreateProcessA main file, find it:

004192A0	CALL DWORD PTR DS:[41B164]	kernel32.CreateProcessA
00418F77	CALL DWORD PTR DS:[41B168]	kernel32.CompareStringA
004191E3	CALL DWORD PTR DS:[41B168]	kernel32.CompareStringW
0040C50E	CALL DWORD PTR DS:[41B0DC]	kernel32.CreateMutexA
0040C39D	CALL DWORD PTR DS:[41B0D8]	kernel32.CreateProcessA
0040C6B8	CALL DWORD PTR DS:[41B1CC]	USER32.DefWindowProcA
004078B8	CALL DWORD PTR DS:[41B0A8]	kernel32.DeleteFileA
0040CA02	CALL DWORD PTR DS:[41B1BC]	USER32.DispatchMessageA

Enter this call:

Address	Hex dump	Disassembly	Comment
0040C39A	56	PUSH ESI	
0040C39B	6A 00	PUSH 0	
0040C39D	FF15 D8B04100	CALL DWORD PTR DS:[41B0D8]	kernel32.CreateProcessA
0040C3A3	56	PUSH ESI	
0040C3A4	E8 82220000	CALL dinerdas.0040E62B	
0040C3A9	83C4 04	ADD ESP,4	
0040C3AC	5F	POP EDI	
0040C3AD	5E	POP ESI	
0040C3AE	5D	POP EBP	
0040C3AF	B0 01	MOV AL,1	
0040C3B1	5B	POP EBX	
0040C3B2	83C4 44	ADD ESP,44	
0040C3B5	C3	RETN	

Right Click on this call, use to bypass Hardware Breakpoint Anti-BP of Integrity Check (do it for Safe):

Breakpoint	▶	Toggle	F2
Run trace	▶	Conditional	Shift+F2
		Conditional log	Shift+F4
		Run to selection	F4
Follow	Enter		
New origin here	Ctrl+Gray *		
Go to	▶	Memory, on access	
Thread	▶	Memory, on write	
Follow in Dump	▶		
		Hardware, on execution	

Now, Switch to Game Menu and click "Play":



Bread:

Address	Hex dump	Disassembly	Comment
0040C39A	56	PUSH ESI	
0040C39B	6A 00	PUSH 0	
0040C39D	FF15 08B04100	CALL DWORD PTR DS:[41B0D8]	kernel32.CreateProcessA
0040C3A3	56	PUSH ESI	
0040C3A4	E8 82220000	CALL dinerdas.0040E62B	
0040C3A9	83C4 04	ADD ESP,4	
0040C3AC	5F	POP EDI	
0040C3AD	5E	POP ESI	
0040C3AE	5D	POP EBP	
0040C3AF	B0 01	MOV AL,1	
0040C3B1	5B	POP EBX	
0040C3B2	83C4 44	ADD ESP,44	
0040C3B5	C3	RETN	

Call before CreateProcessA, the launcher has generated Trial Key. Then, we can unpack the main energy. Do not close this Olly, Olly and other open load main file:

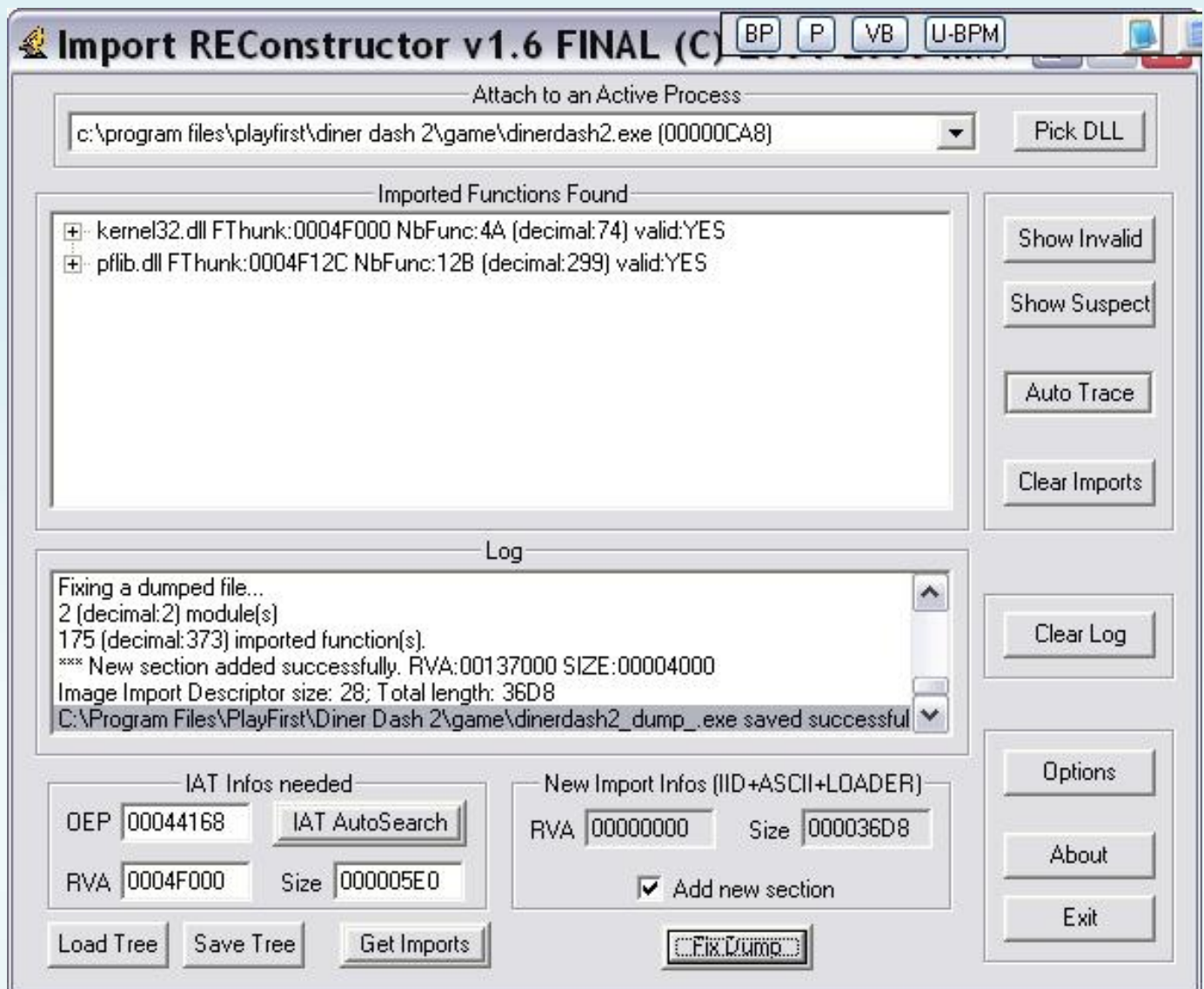
Address	Hex dump	Disassembly	Comment
0048A2E3	55	PUSH EBP	
0048A2E4	8BEC	MOV EBP,ESP	
0048A2E6	6A FF	PUSH -1	
0048A2E8	68 204B4B00	PUSH dinerdas.004B4B20	
0048A2ED	68 20A04800	PUSH dinerdas.0048A020	SE handler installat
0048A2F2	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
0048A2F8	50	PUSH EAX	
0048A2F9	64:8925 0000	MOV DWORD PTR FS:[0],ESP	
0048A300	83EC 58	SUB ESP,58	
0048A303	53	PUSH EBX	
0048A304	56	PUSH ESI	
0048A305	57	PUSH EDI	
0048A306	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0048A309	FF15 88F14A00	CALL DWORD PTR DS:[<&KERNEL32.GetVersion	kernel32.GetVersion
0048A30F	33D2	XOR EDX,EDX	
0048A311	8004	MOV DL,4H	

Use fly's script again (or use manual unpacking), stop it at OEP:



Address	Hex dump	Disassembly	Comment
00444168	6A 60	PUSH 60	This is the OEP! F
0044416A	68 00444500	PUSH dinerdas.00454400	
0044416F	E8 1C110000	CALL dinerdas.00445290	
00444174	BF 94000000	MOV EDI,94	
00444179	8BC7	MOV EAX,EDI	
0044417B	E8 50420000	CALL dinerdas.004483D0	
00444180	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00444183	8BF4	MOV ESI,ESP	
00444185	893E	MOV DWORD PTR DS:[ESI],EDI	
00444187	56	PUSH ESI	
00444188	FF15 04F14400	CALL DWORD PTR DS:[44F104]	kernel32.GetVersio
0044418E	8B4E 10	MOV ECX,DWORD PTR DS:[ESI+10]	
00444191	890D 7CCC4500	MOV DWORD PTR DS:[45CC7C],ECX	

Dump it and fix IAT:



Ok, fixed the file and run game start DONE! (phewww ... 😊 do not need damn Overlay Data)

IV-Ending:

I do not like these games on this doc, just excited to playing with Armadillo. But if you like games, you should buy them. Authors worked hard and they should be paid for their work.

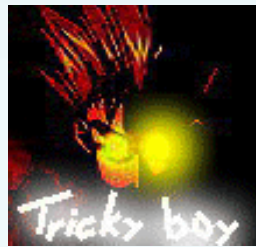
I tried my best but damn sorry for my english 😞 ! See ya next time.

Big thanks to:

All REA's members: Computer_Angel, Moonbaby, Zombie, hacnho, benina, kienmanowar, rongchaua, Deux, Merc, hoadongnoi, the_lighthouse, TQN, light.phoenix, hytkl, tlandn, hurt_heart, dzungltn, Zoi, littleboy84, haule_nth, takada, Why not bar, iamidiot, Akira, dump, thienthandien, [kid], ...

Special thanks to:

fly, stephenteH, Gabri3l, MaDMAAn_H3rCuL3s, CondZero, Ricardo Narvaja + NCR, lena151, hagggar, ARTeam, snd, RES, CrackLatinos, all unpack.cn ... Authors who created tools and you.



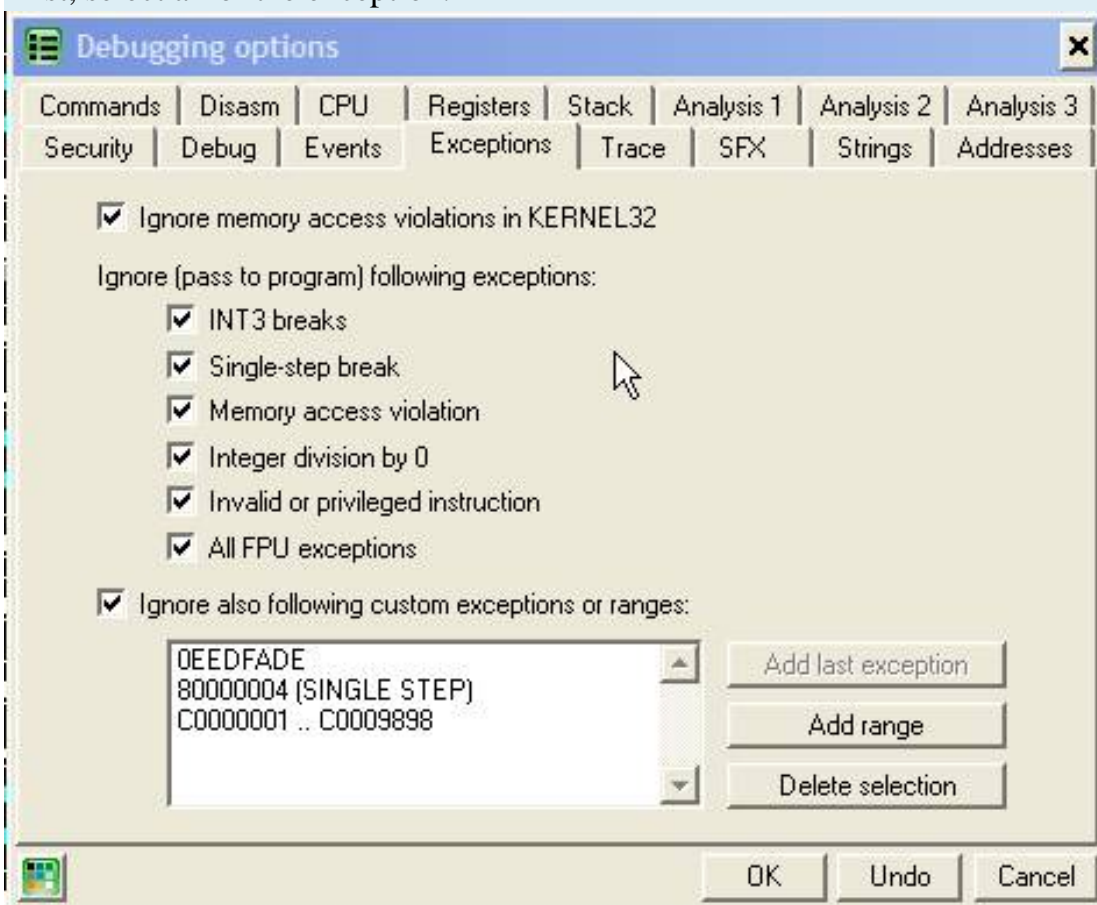
Written by Trickyboy - 2006
Welcome to www.reaonline.net

Target: UnPackMe_Armadillo3.70a.h (included in rar file)

Pack: Armadillo 3.70 with option Standard + Code Splicing + IAT elimination

Tools: Olly hacnho ImportREC + + + LordPE armInline 0.4.1 (included in rar file)

First, select all of the exception.



Start:

Olly _Load to target:

He GetModuleHandleA, Shift + F9, olly break:

_Shift + F9, Shift + F9, ALT + F9, => magic jump

Debugger window showing assembly code and registers. The assembly code is for the CPU - main thread. The registers window shows the state of the CPU registers. The command window shows the command 'He GetModuleHandleA'.

Assembly code (Address | Value | Comment):

Address	Value	Comment
00401000	77E7B5E1	00132: SelectObject
00401004	77E7B5E1	00132: BitBlt
00401008	77E7B5E1	00132: DeleteObject
0040100C	77E7B5E1	00132: CreatePalette
00401010	77E7B5E1	00132: CreateDC
00401014	77E7B5E1	00132: SelectPalette
00401018	77E7B5E1	00132: RealizePalette
0040101C	77E7B5E1	00132: CreateGDIObject
00401020	77E7B5E1	00132: DeleteDC
00401024	77E7B5E1	00132: CreateCompatibleDC
00401028	77E7B5E1	00132: SetThreadPriority
0040102C	77E7B5E1	00132: GetCurrentThread
00401030	77E7B5E1	00132: CreateProcessA

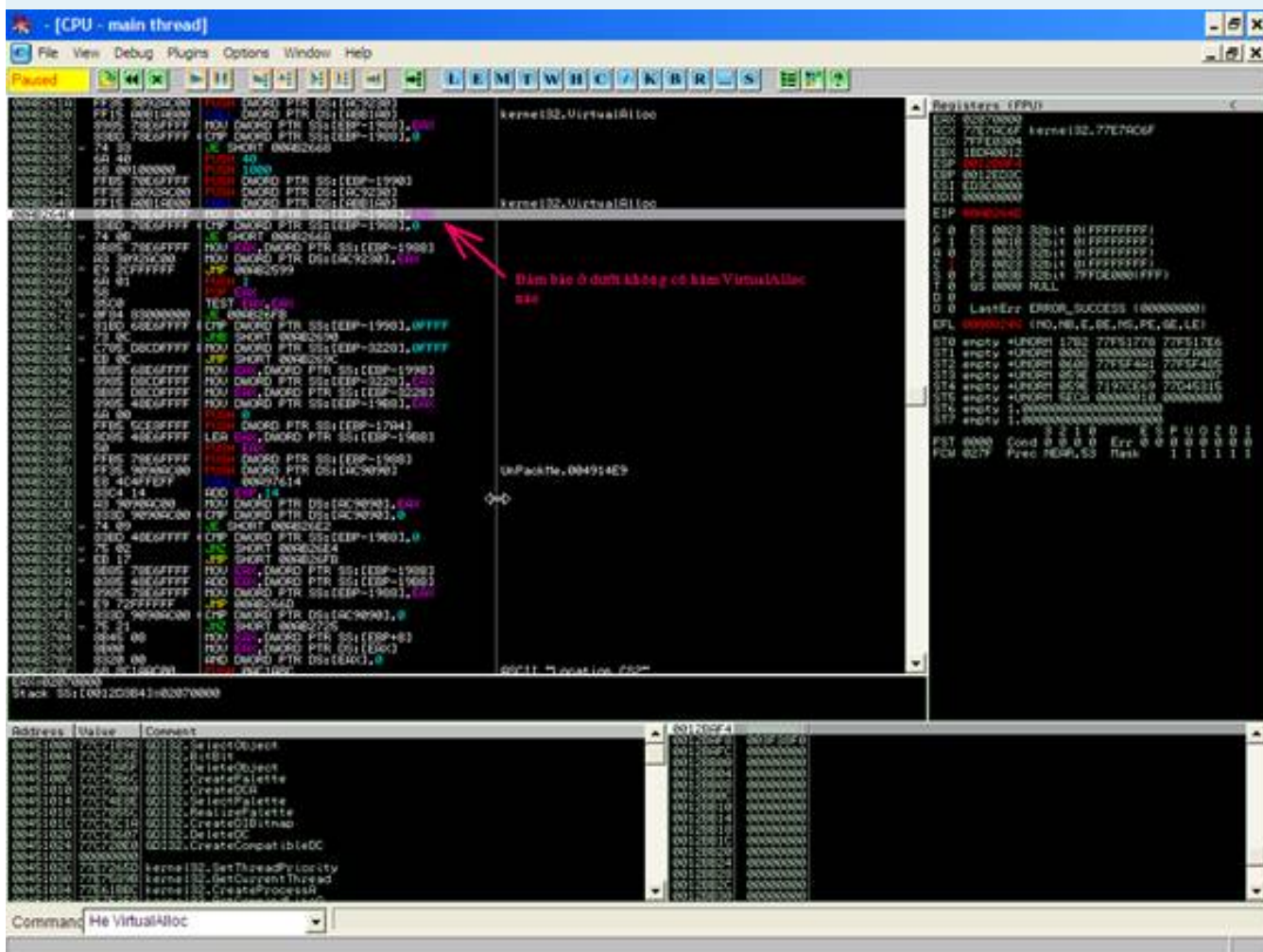
Registers (EAX | ECX | EDI | ESP | ESI | EDI | ESP):

Register	Value
EAX	77E7B5E1
ECX	77E7B5E1
EDI	77E7B5E1
ESP	77E7B5E1
ESI	77E7B5E1
EDI	77E7B5E1
ESP	77E7B5E1

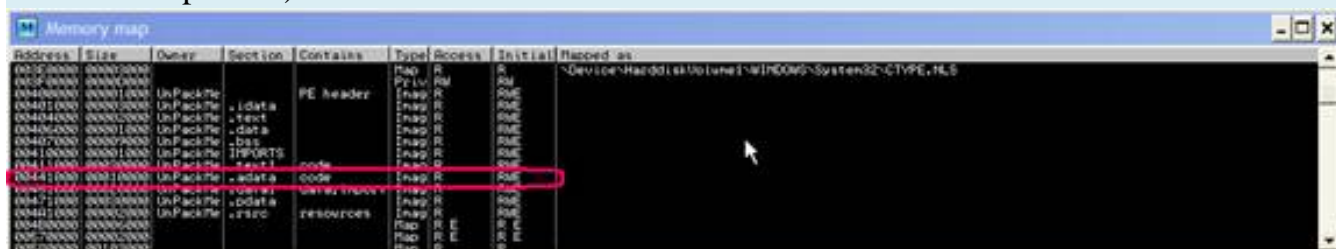
Command: He GetModuleHandleA

_Hd GetModuleHandleA, He VirtualAlloc, Shift + F9, olly break:

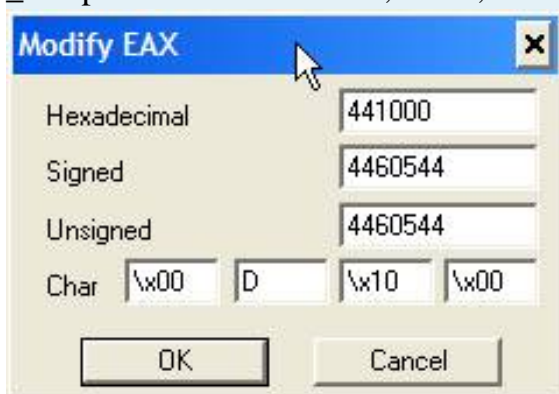
Shift + F9, Shift + F9, Shift + F9, Alt + F9,



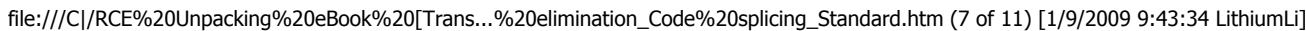
_EAX now contains the address areas that will set arma stolen byte code of the original target, now we will redirect the region of 1 area is available in the file that we used to do (eg region. ADATA, is the area code contains the packer)

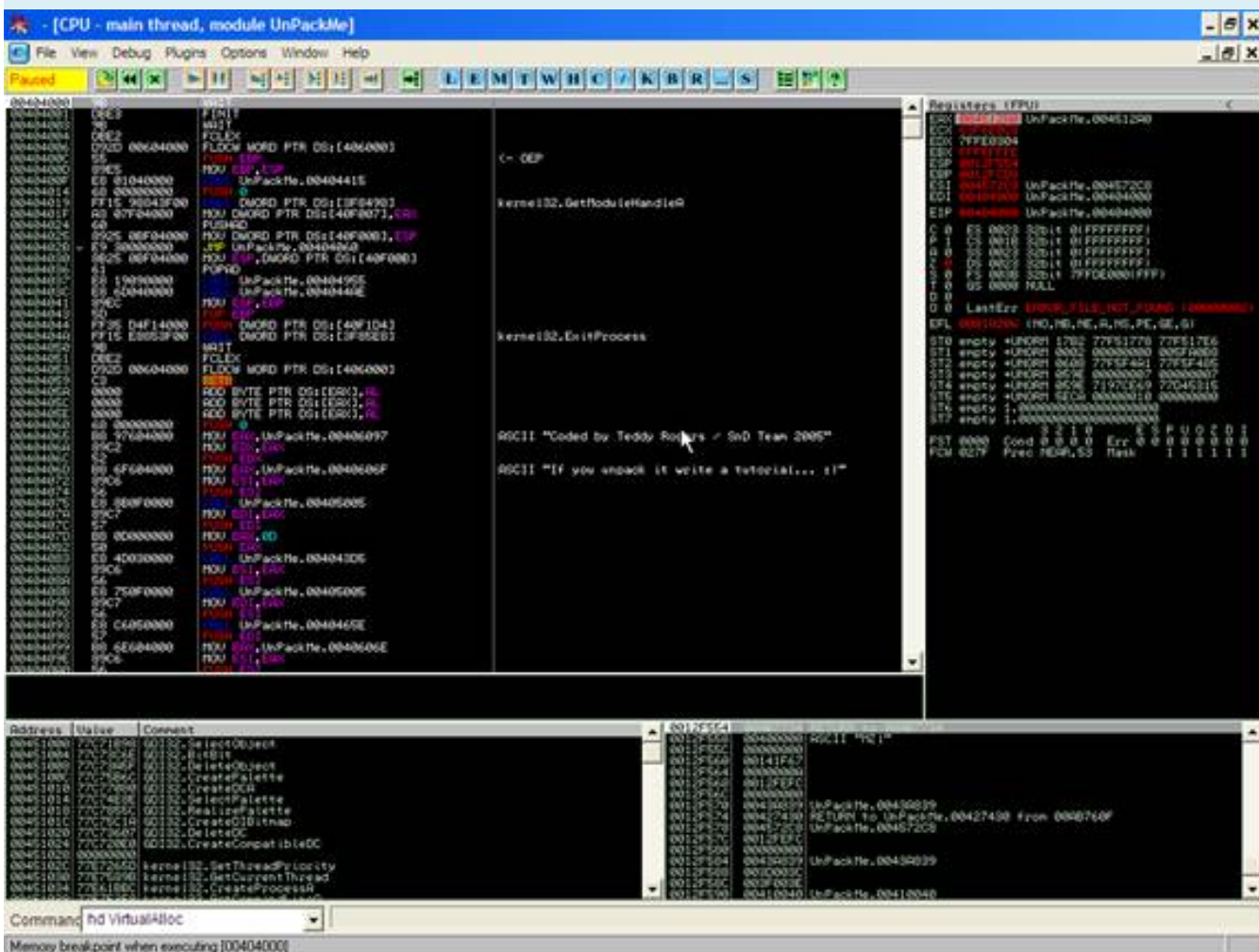


_Nhập Left mouse of EAX, Enter, then enter 441000 invisible, and OK.



_Ta Code Splicing defeat was, now we'll come to OEP. Hd VirtualAlloc, Alt + M, BP on access to the section. Text:





_ OEP: 40400C.

We have magic patch jump to avoid destruction IAT, redirect Code Splicing (this you can always use the tool ArmInline). Now the IAT elimination.

You refer to:

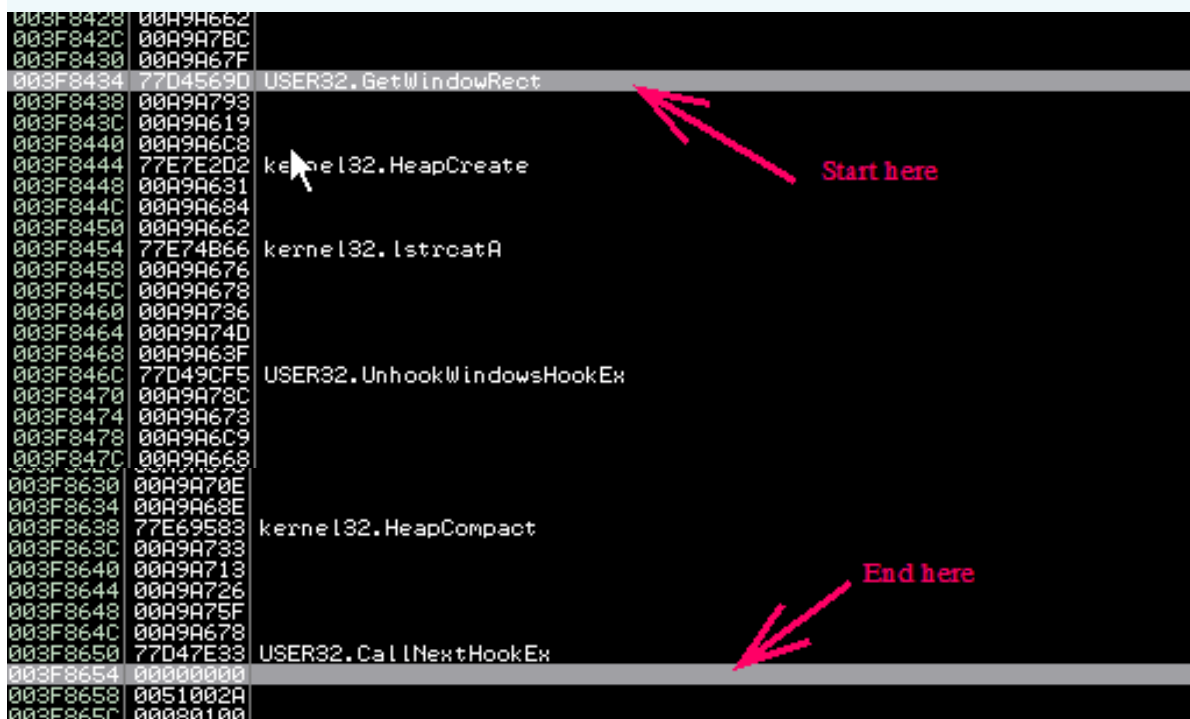
Import Address Table (IAT) elimination means that the protector has erased the IAT nice and clean from the protected exe, meaning all the strings and valid pointer locations and has put the API API elsewhere. Those pointers are not the original calls at the APIs Redirected but in the memory locations that are allocated during the unpacking of the program (prog reaches till the OEP). Those memory locations are most of the time (I can say every time) in a lower memory location that the ImageBase to the exe make it harder to find and then with ImpRec probably use some Plug Reconstruct to them, as they are all invalid. The protector also changes the call dword ptr ds: [xxxxxxx] calls in original code section of the exe (where xxxxxx is the location API where the pointer is valid before the packing, in the clear exe, in the clear meaning IAT). So the protector makes them into call dword ptr ds: [yyyyyyy] where are the new yyyyyyy Redirected pointers of APIs I said earlier. Of course in [yyyyyyy] the pointers are invalid, but the jumps at EIP [yyyyyyy] codes are located where to dynamically re-create the original valid pointers, and then with a call register (where in the register is good valid pointer) the prog jumps at the API. Of course it is not for each API Redirected like that, but is almost the same philosophy for all.

(Source: Armadillo IAT elimination + code by splicing KaGra)

Here:



_ Neu Following in your dump, you will reach IAT:



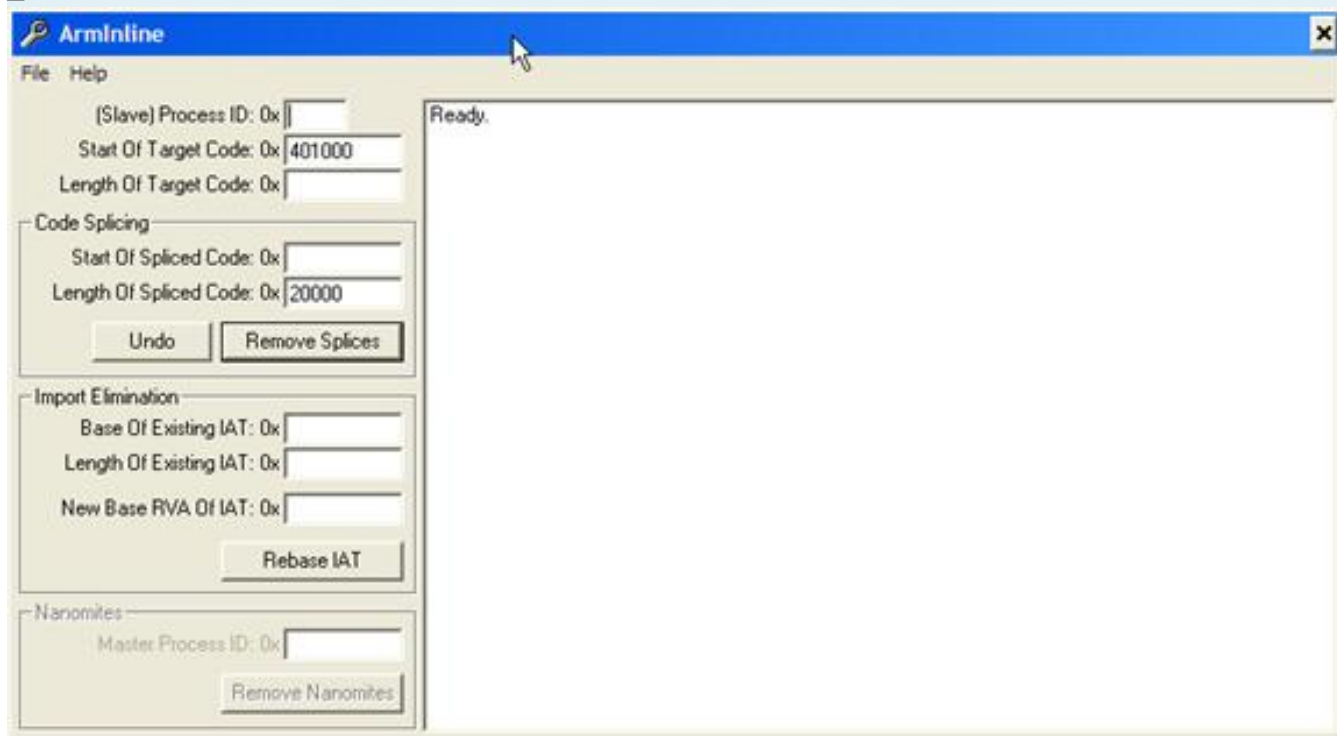
_ Vay IAT:

Start: 003F8434 77D4569D USER32.GetWindowRect

Finish: 003F8654 00000000

Len = 3f8654 - 3f8434 = 220

_ Now run ArmInline:



_ Fill in the boxes:

Process ID: PID your

Start of target code: RVA's section. Text (404000)

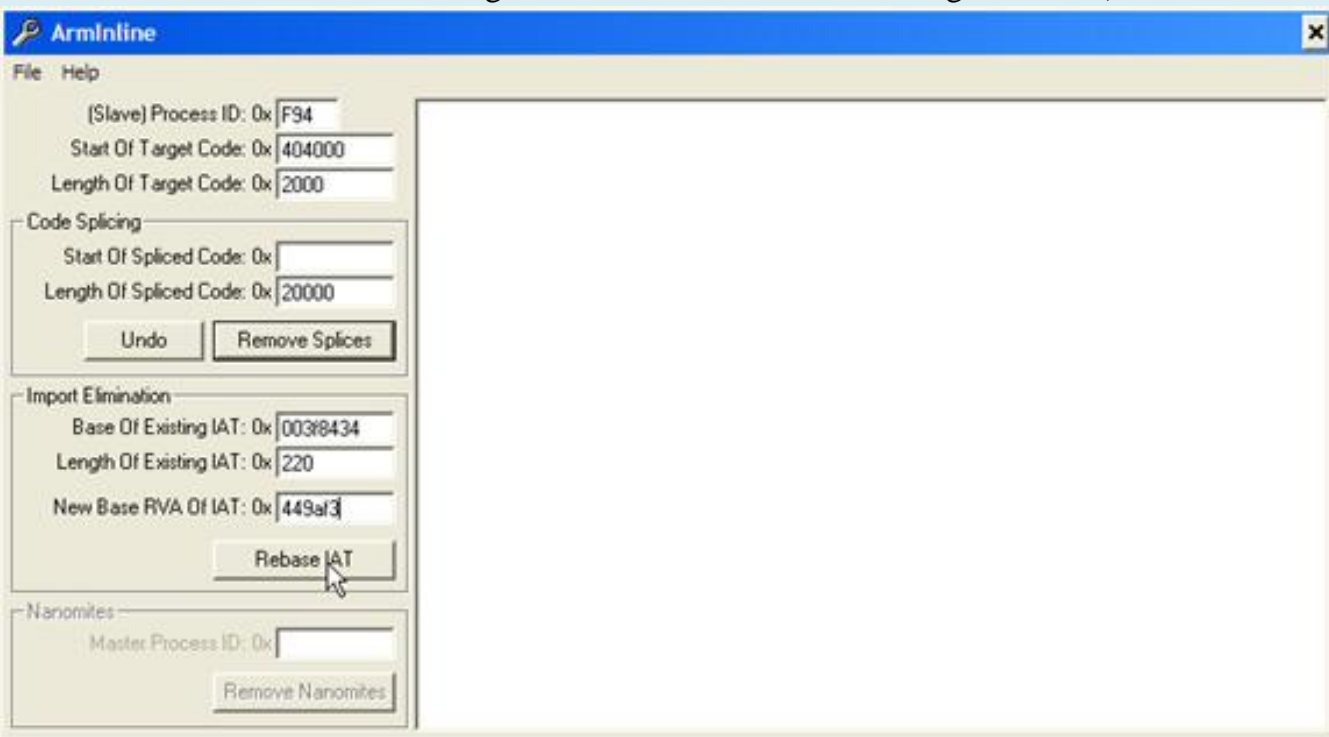
Length of target code: the length of the section. Text (2000)

In the Import elimination:

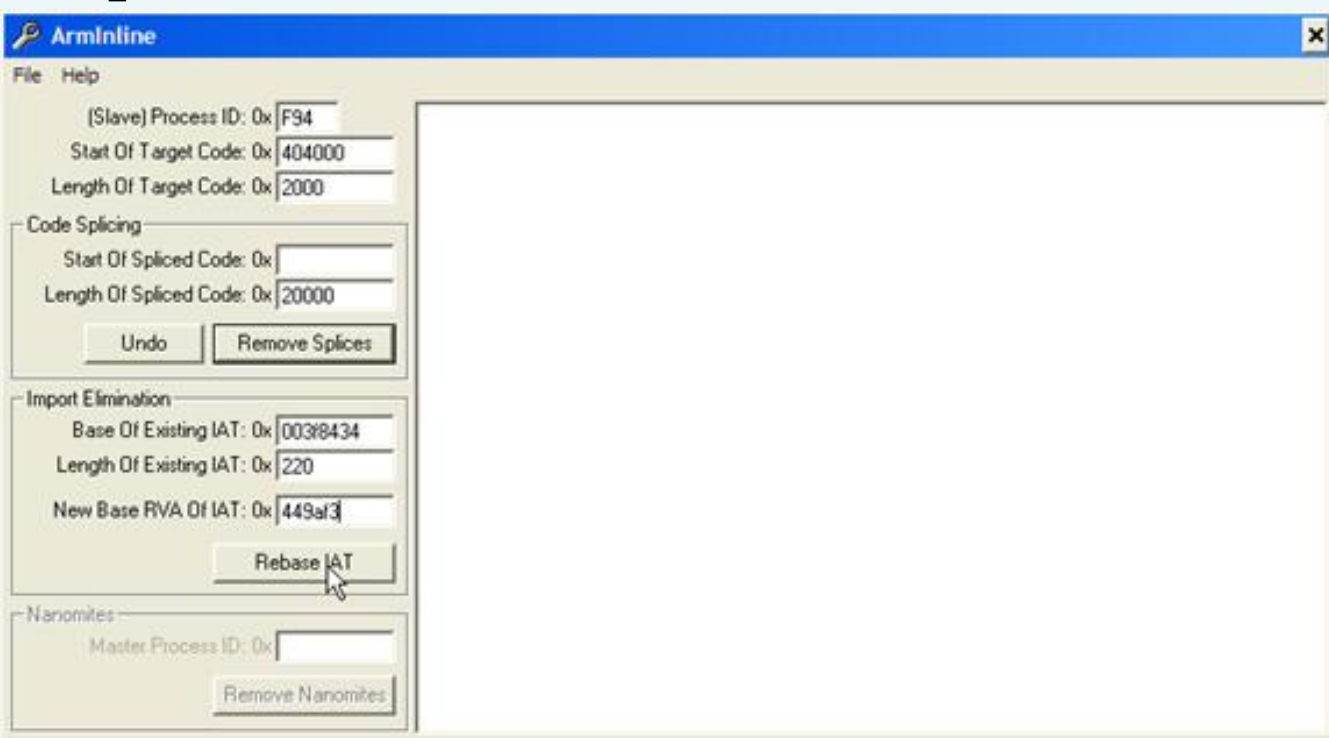
Existing Base of IAT: 003f8434

Length of existing IAT: 220

New base RVA of IAT: This you can find the cave 1 large > 220 bytes, then enter the address of the first cave. (I found section. ADATA about 1 large areas to bottom, corn on the right: 449af3)



Rebase _Nhan IAT:



Now olly:

The screenshot shows the UnPackMe debugger interface. The main window displays assembly code with various instructions and comments. Two specific instructions are highlighted with red boxes and arrows pointing to them from a red text box that says "Các gài simulation quá giờ":

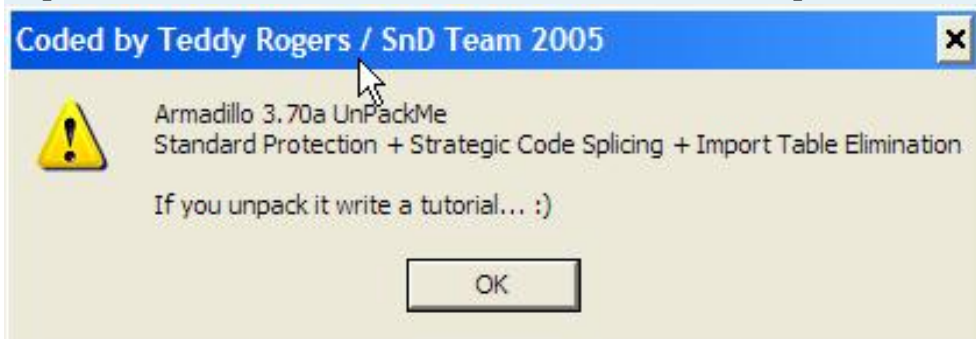
- `kernel32.GetModuleHandle`
- `kernel32.ExitProcess`

The right-hand pane shows the state of the CPU registers, including EAX, ECX, EDI, ESI, ESP, EIP, and others, along with their values and segment registers (CS, DS, SS, ES, FS, GS).

At the bottom, there is a command line with the text "hd VirtualAlloc".

Open lordPE, full dump -> fix IAT:

ImportREC -> OEP: 400C -> IAT auto search -> Get imports -> fix dump -> done



Armadillo 4.xx-Code Splicing (Other Method)

Why Not Bar

Target: RM to MP3 Converter v1.32 (target2 in Armdillo_tuts_6_exp by Hacnho)

Packer Armadillo 4.xx Code Splicing (anti-dump)!

Tools: OllyDbg (by **hacnho**), LordPE 1.4, 1.6 Import REConstructor

This is the first tut they write can be difficult to complete if there is anything wrong xót expect the medical sincere suggestions and to ignore. If you have a track and practice of the "secret Kiếp" unpack Armadillo's Hacnho really get it you know the Method Ma Su Additional HacNho presented. But in the West has its **MaDMAn_H3erCuL3s** of how to use format **Code Splicing** is Ta. No Fix **Code Splicing** What ráo Add 1 run section is delicious healthy. According to how you think how long the run is OK. You should also see the stars. To easily compare their use Target as RM to MP3 Converter v1.32 in Armdillo_tuts_6_exp.

_Nhu Usual we Load Target to Olly:

00496000	60	E8 00000000	CALL RMTOMP3.00496006	
00496001	E8	00000000	CALL RMTOMP3.00496006	
00496006	50		POP EBP	kernel32.77E814C7
00496007	50		PUSH EAX	
00496008	51		PUSH ECX	
00496009	0FCA		BSWAP EDX	
0049600B	F7D2		NOT EDX	
0049600D	9C		BT SHFT	
0049600E	F7D2		NOT EDX	
00496010	0FCA		BSWAP EDX	
00496012	55		IMB SHFT	

_Va To IAT has been completed we need to patch magic jump to quickly you can use the script (with the tut). Running Script finished you here:

003A9B49	8B0D	74B73D00	MOV ECX,DWORD PTR DS:[30B7741]	
003A9B4F	89040E		MOV DWORD PTR DS:[ESI+ECX],EAX	kernel32.77E60000
003A9B52	A1	74B73D00	MOV EAX,DWORD PTR DS:[30B7741]	
003A9B57	391C06		CMP DWORD PTR DS:[ESI+EAX],EBX	
003A9B5A	75	16	JNZ SHORT 003A9B72	
003A9B5C	8D85	B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
003A9B62	50		PUSH EAX	kernel32.77E60000
003A9B63	FF15	DC003D00	CALL DWORD PTR DS:[3000DC]	kernel32.LoadLibraryA
003A9B69	8B0D	74B73D00	MOV ECX,DWORD PTR DS:[30B7741]	
003A9B6F	89040E		MOV DWORD PTR DS:[ESI+ECX],EAX	kernel32.77E60000
003A9B72	A1	74B73D00	MOV EAX,DWORD PTR DS:[30B7741]	
003A9B77	391C06		CMP DWORD PTR DS:[ESI+EAX],EBX	
003A9B7A	75	16	JNZ SHORT 003A9B92	
003A9B80	33C9		XOR ECX,ECX	kernel32.77E7B5E1
003A9B82	0007		MOV EAX,DWORD PTR DS:[EDI]	
003A9B84	3918		CMP DWORD PTR DS:[EAX],EBX	
003A9B86	74	06	JE SHORT 003A9B8E	
003A9B88	41		INC ECX	kernel32.77E7B5E1
003A9B89	83C0	0C	ADD EAX,0C	

_Patch To:

003A9B62	50	CALL EAX	kernel32.77E60000
003A9B63	FF15 DC003D00	CALL DWORD PTR DS:[3D00DC]	kernel32.LoadLibraryA
003A9B69	8B0D 74B73D00	MOV ECX,DWORD PTR DS:[3D8774]	
003A9B6F	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	kernel32.77E60000
003A9B72	A1 74B73D00	MOV EAX,DWORD PTR DS:[3D8774]	
003A9B77	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
003A9B7A	✓ E9 33010000	JMP 003A9CB2	
003A9B7F	90	NOB	
003A9B80	33C9	XOR ECX,ECX	kernel32.77E7B5E1
003A9B82	8B07	MOV EAX,DWORD PTR DS:[EDI]	
003A9B84	3918	CMP DWORD PTR DS:[EAX],EBX	
003A9B86	✓ 74 06	JE SHORT 003A9B8E	
003A9B88	41	INC ECX	kernel32.77E7B5E1
003A9B89	83C0 0C	ADD EAX,0C	
003A9B8C	^ EB F6	JMP SHORT 003A9B84	
003A9B8E	8B09	MOV EBX,ECX	kernel32.77E7B5E1
003A9B90	C1E3 02	SHL EBX,2	

_Nhan Alt-M to window "Memory Map". Set in Breakpoint section. Text:

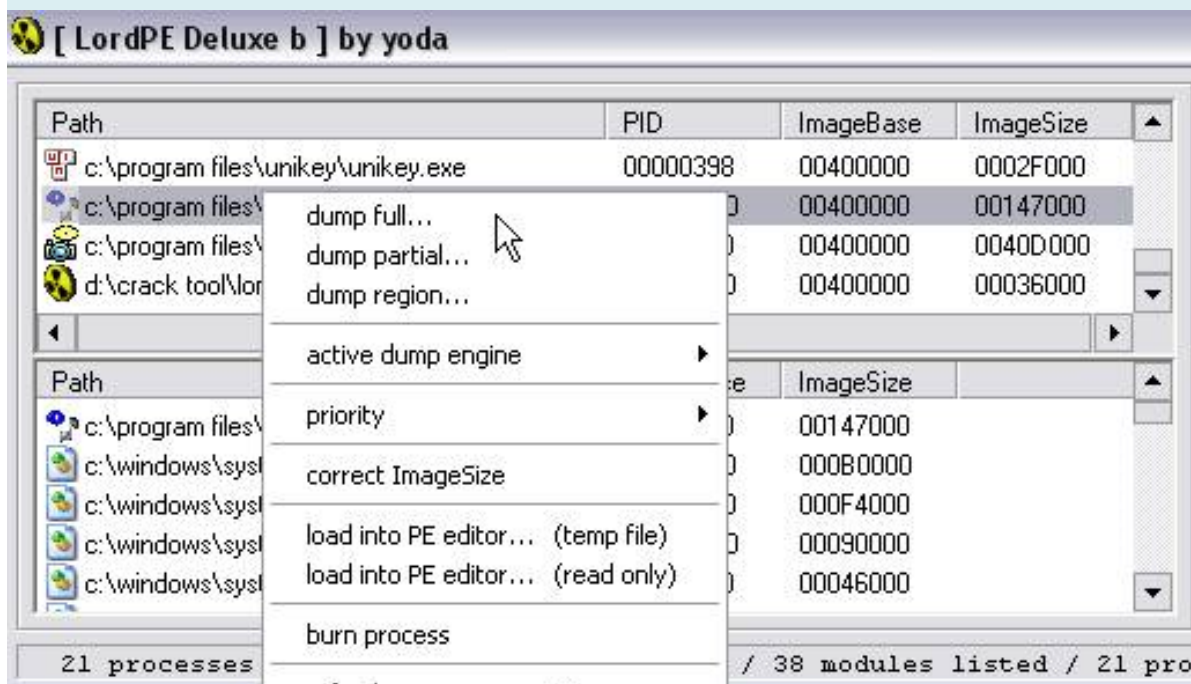
00400000	00001000	RMTOMP3		Imag	R	RWE	
00401000	0004B000	RMTOMP3	.text	Imag	R	RWE	
0044C000	00010000	RMTOMP3	.rda				
0045C000	0000A000	RMTOMP3	.dat				
00466000	00030000	RMTOMP3	.text				
00496000	00010000	RMTOMP3	.ada				
004A6000	00020000	RMTOMP3	.dat				
004C6000	00070000	RMTOMP3	.pda				
00536000	00011000	RMTOMP3	.rsr				
00550000	00003000						
00610000	00002000						
00620000	00103000						
00730000	0007D000						
00A30000	00001000						
00B30000	0000C000						
00B40000	00002000						
00B50000	00018000						
00B70000	000A4000						
00C2A000	0000A000						
00C50000	00006000						
00C60000	00001000						
00C70000	00001000						
00E70000	00001000						
00E80000	00003000						
00E90000	00010000						
01290000	00003000						
012D0000	00004000						
012E0000	00003000						
012F0000	00001000						

_Nhan Shift-F9. We here:

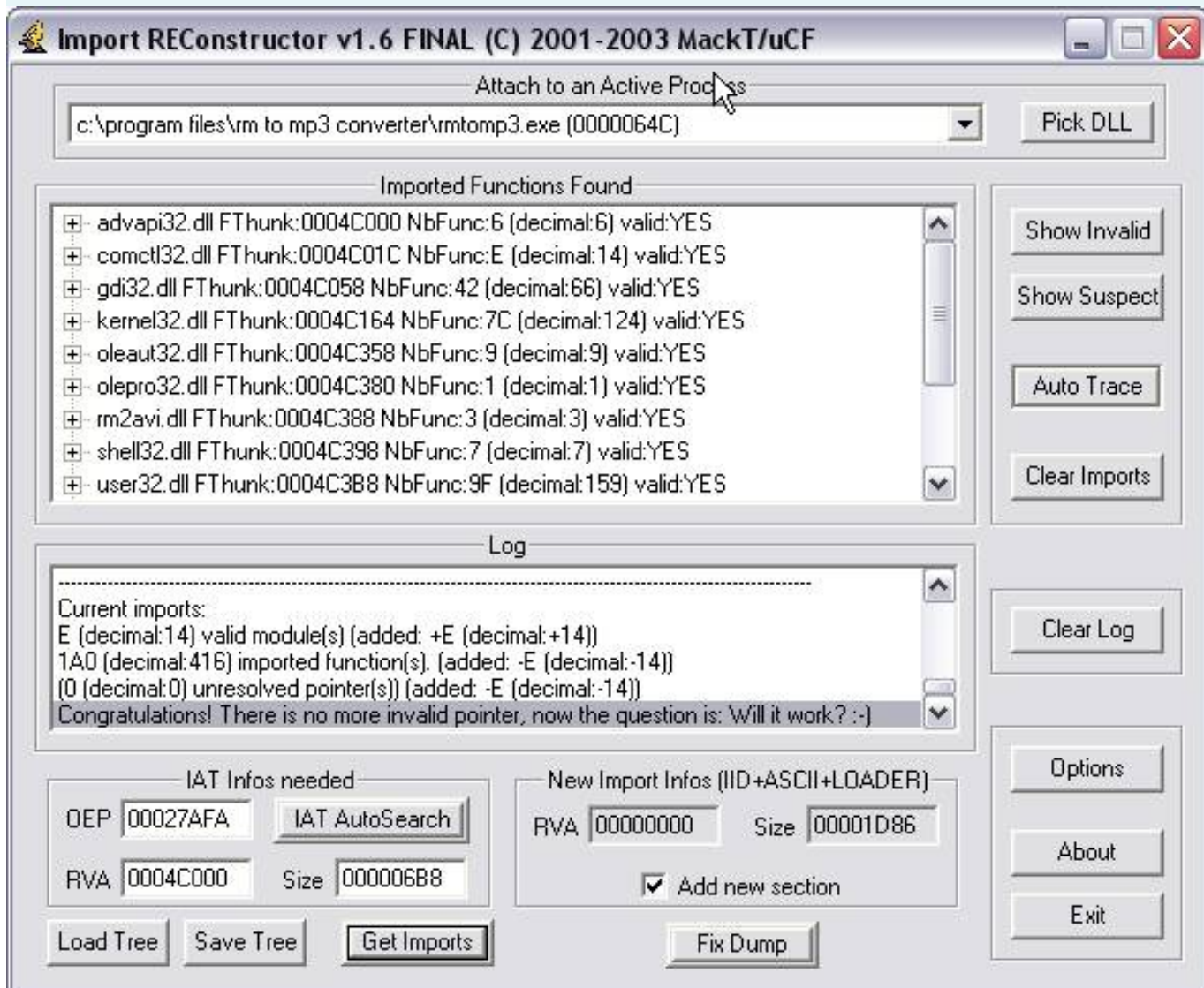
Address	Hex dump	Disassembly	Comment
00427AFA	55	PUSH ESP	
00427AFB	8BEC	MOV EBP,ESP	
00427AFD	6A FF	PUSH -1	
00427AFF	68 881B4500	PUSH RMTOMP3.00451B88	
00427B04	68 A8AF4200	PUSH RMTOMP3.0042AFA8	
00427B09	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00427B0F	50	PUSH EAX	
00427B10	64:8925 000000	MOV DWORD PTR FS:[0],ESP	
00427B17	83EC 58	SUB ESP,58	
00427B1A	53	PUSH EBX	
00427B1B	56	PUSH ESI	
00427B1C	57	PUSH EDI	
00427B1D	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00427B20	FF15 C4C14400	CALL NEAR DWORD PTR DS:[44C1C4]	kernel32.GetVersion
00427B26	33D2	XOR EDX,EDX	
00427B28	8A04	MOV DL,AH	
00427B2A	8915 B8374600	MOV DWORD PTR DS:[4637B8],EDX	
00427B30	8BC8	MOV ECX,EAX	

OEP: 00427AFA

_Bay Hours we used to dump LordPE Full. Open LordPE, dump:



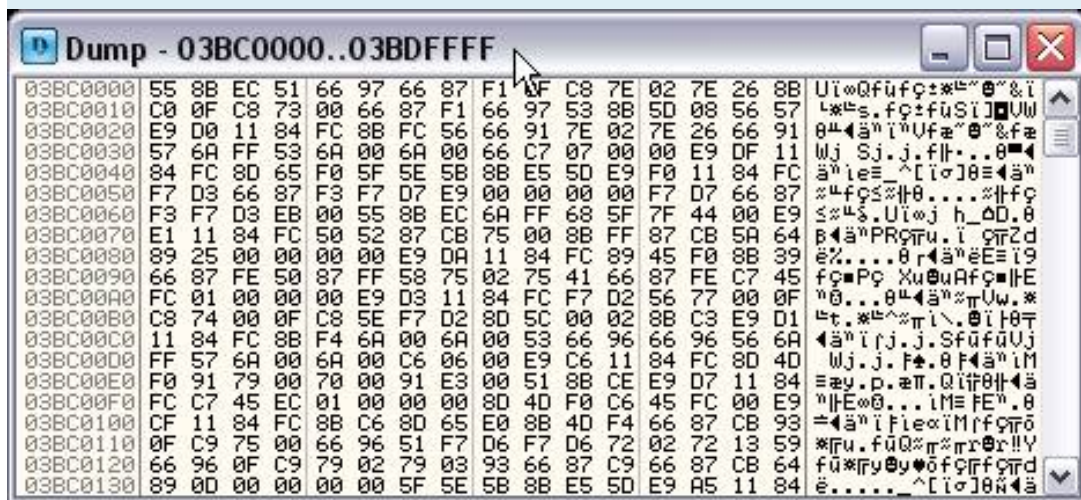
Open Imprec, fill OEP: 27AFA. Click "IAT AutoSearch," OK, CutThunk (s), "Get Import":



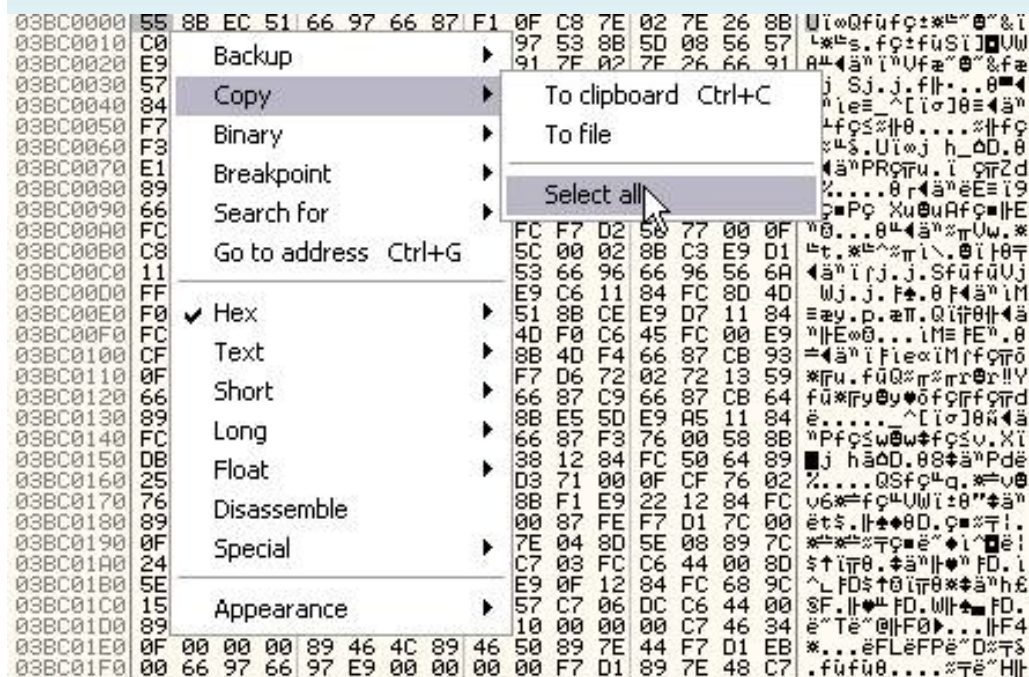
Fix dump, try Run and of course it does not run and I important steps, we need to find a cove Splicing of the Code. Section usually form 0xxx0000. View pictures that you will immediately stop:

00330000	00049000				Priv	RW		
003D7000	00004000				Priv	RW		
003B4000	0000D000				Priv	RW		
003F0000	00002000				Map	R		
00400000	00001000	RMTOMP3			Imag	R		RWE
00401000	0004B000	RMTOMP3	.text		Imag	R		RWE
0044C000	00010000	RMTOMP3	.rdata		Imag	R		RWE
0045C000	0000A000	RMTOMP3	.data		Imag	R		RWE
00466000	00030000	RMTOMP3	.text1	code	Imag	R		RWE
00496000	00010000	RMTOMP3	.adata	code	Imag	R		RWE
004A6000	00020000	RMTOMP3	.data1	data, import	Imag	R		RWE
004C6000	00070000	RMTOMP3	.pdata		Imag	R		RWE
00536000	00011000	RMTOMP3	.rsrc	resources	Imag	R		RWE
00550000	00003000				Map	R E		R E
00610000	00002000				Map	R E		R E
00620000	00103000				Map	R		R
00730000	0007D000				Map	R E		R E
00A30000	00001000				Priv	RW		RW
00B30000	0000C000				Priv	RW		RW
00B40000	00002000				Map	R		R
00B50000	00018000				Priv	RW		RW
00B70000	000A4000				Priv	RW		RW
00C2A000	0000A000				Priv	RW		RW
00C50000	00006000				Priv	RW		RW
00C60000	00001000				Map	RW		RW
00C70000	00001000				Priv	RW		RW
00E70000	00001000				Priv	RW		RW
00E80000	00003000				Priv	RW		RW
00E90000	00010000				Priv	RW		RW
01290000	00003000				Priv	RW		RW
012D0000	00004000				Priv	RW		RW
012E0000	00003000				Priv	RW		RW
012F0000	00001000				Priv	RW		RW
0146E000	00002000				Priv	RW	Gua:	RW
03BC0000	00020000				Priv	RWE		RWE
10000000	00001000	RM2AVI	.text	PE header	Imag	R		RWE
10001000	0000D000	RM2AVI	.code	code	Imag	R		RWE
1000E000	00002000	RM2AVI	.rdata	imports, exp	Imag	R		RWE
10010000	00005000	RM2AVI	.data	data	Imag	R		RWE
10015000	00002000	RM2AVI	.reloc	relocations	Imag	R		RWE

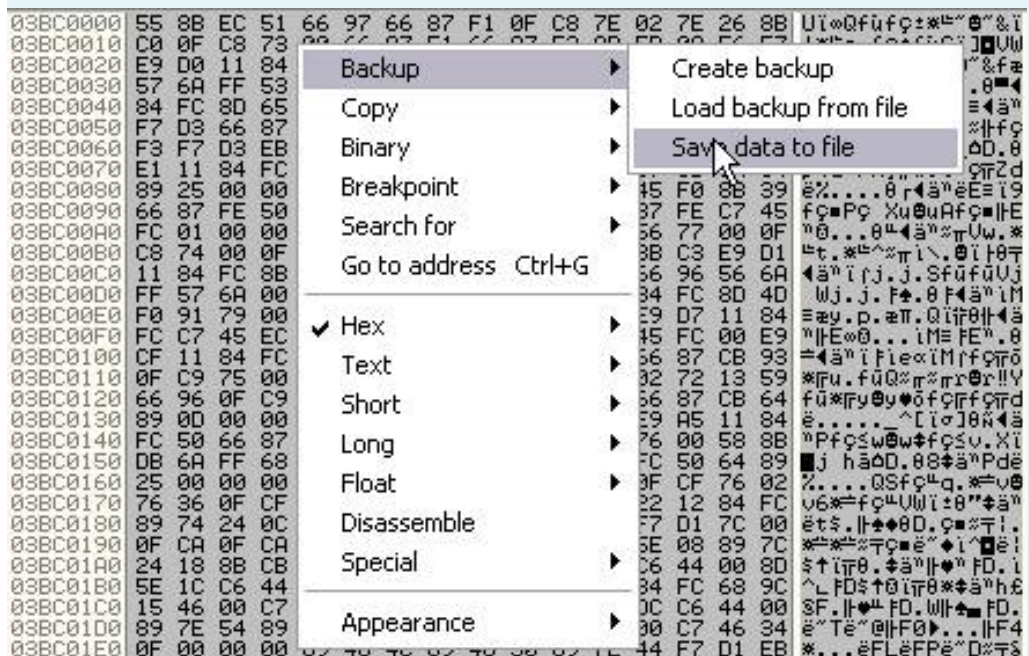
Double click on Address 03BC0000 Size = 00020000 =



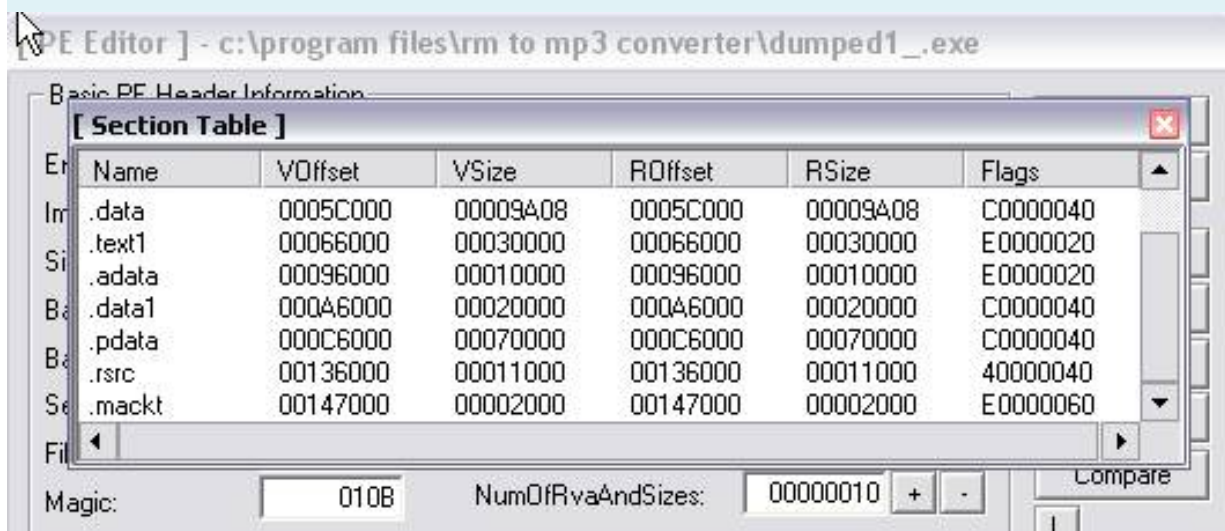
Select Copy / Select All



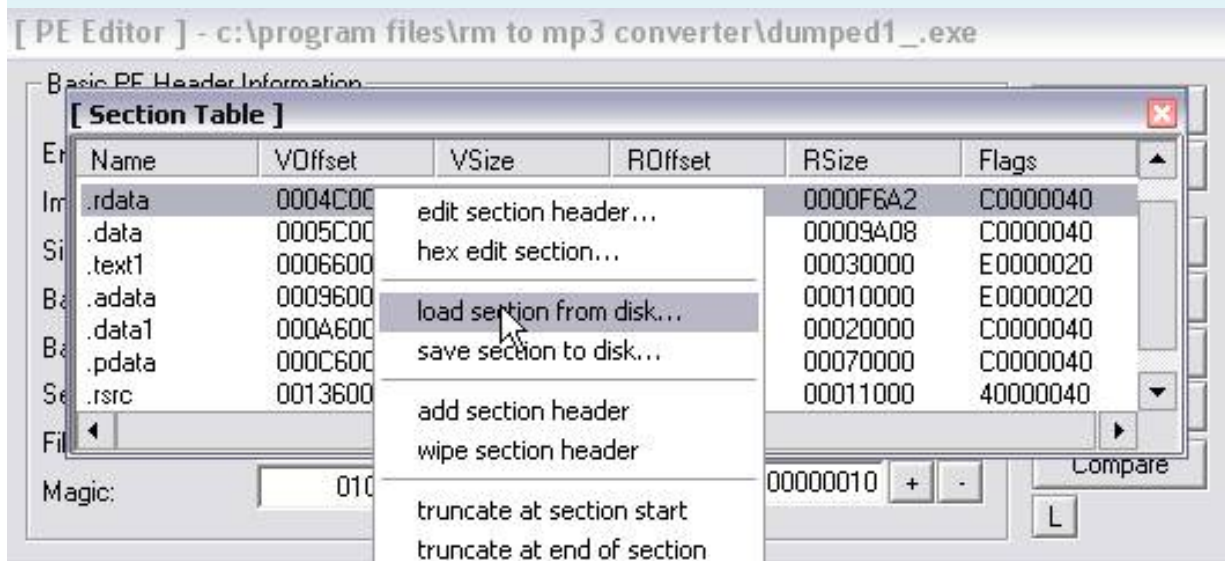
Chon Save data to file, save the file formats are *. mem, the machine is "_03BC0000.mem"



Open a new file dump fix completed by PE Editor's LordPE, click the button next to Section open window Section Table



In Section Table window, click to select Load from disk section



Select _03BC0000.mem (your computer may be different)



[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
.text1	00066000	00030000	00066000	00030000	E0000020
.adata	00096000	00010000	00096000	00010000	E0000020
.data1	000A6000	00020000	000A6000	00020000	C0000040
.pdata	000C6000	00070000	000C6000	00070000	C0000040
.rsrc	00136000	00011000	00136000	00011000	40000040
.mactt	00147000	00002000	00147000	00002000	E0000060
._03BC000	00149000	00020000	00149000	00020000	E00000E0

_Toi You need to source a bit more to the new Section to run lickerish: 03BC0000 - 400,000 = 37C0000. Then Save again. I may be more Newbie question number 400,000 in the first? Simply it is only with Image Base default is 400,000.

[Edit SectionHeader]

Section Header

Name: ._03BC000

VirtualAddress: 37C0000

VirtualSize: 00020000

RawOffset: 00149000

RawSize: 00020000

Flags: E00000E0 ...

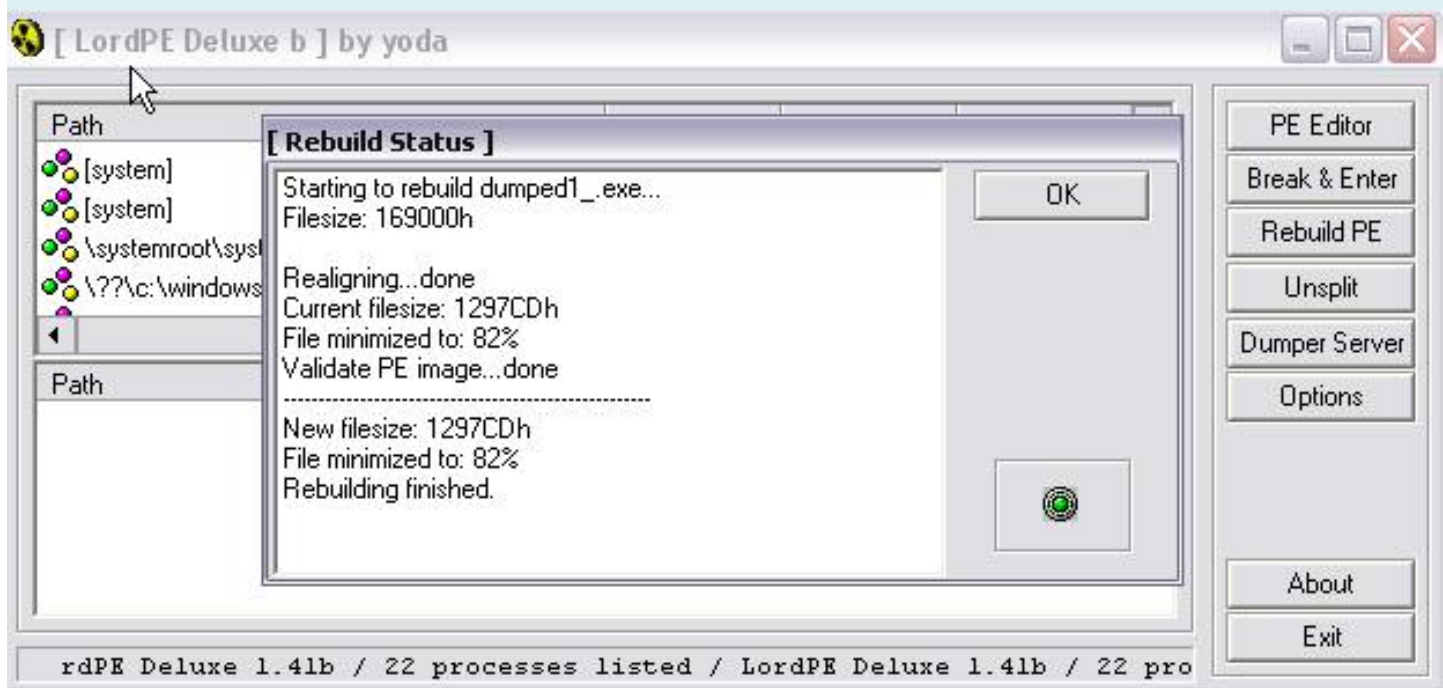
OK

Cancel

Test Run _ considered stars! What heaven up

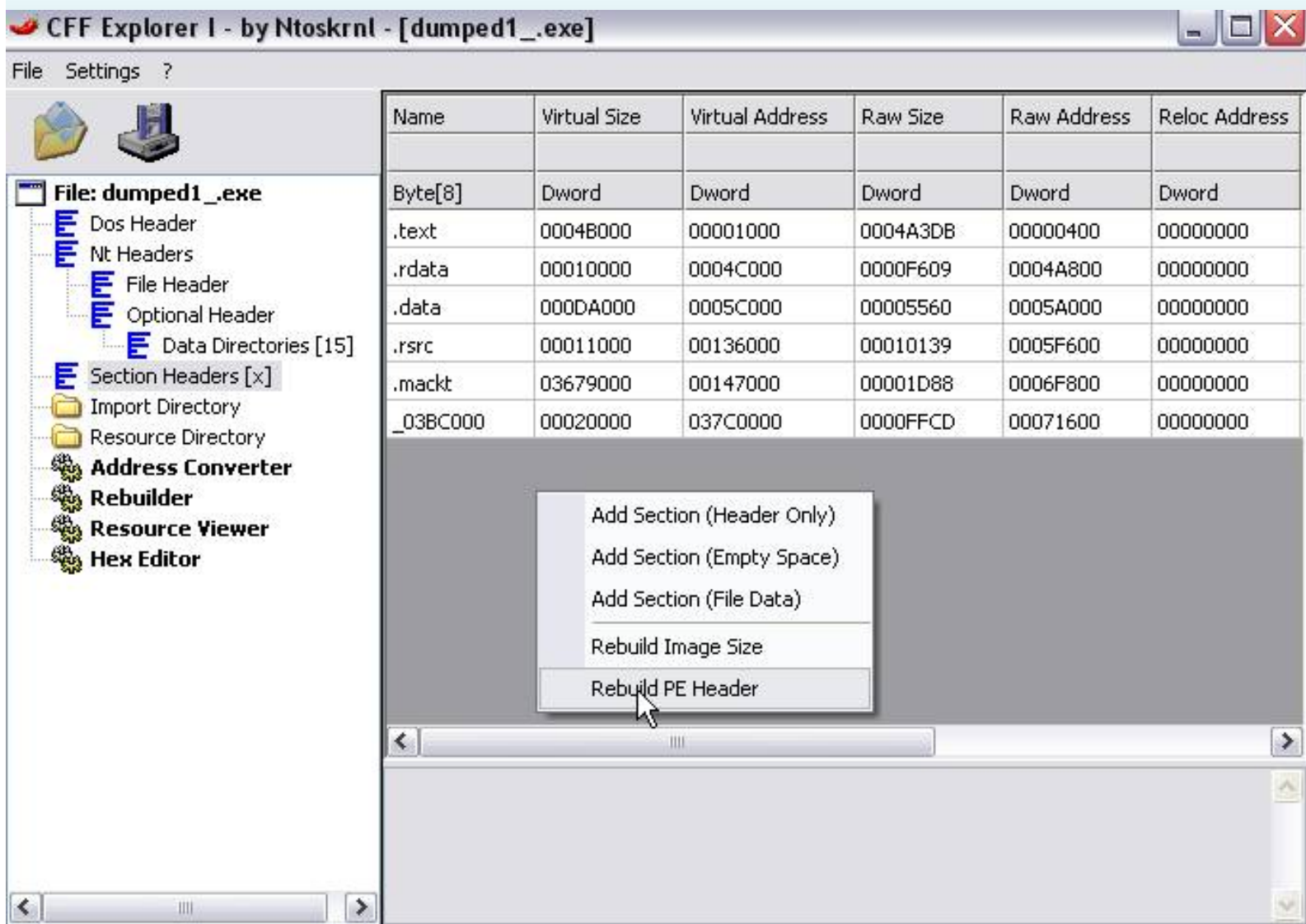


_Uhm! Running LordPE rebuild PE à à select New File Save



_ Run again, hú complete run call. Unpack Done !!!!!!!!!!!!!!!

If you want to collapse it again using the CFF Explorer remove Section legacy. Text1. ADATA. Data1. Pdata. Then click to select rebuild PE Header.



_Save The space is 517 KB

Have Fun!

Thanx To: [Hacnho](#), [MaDMAn_H3erCuL3s ...](#) and [You](#)

Written by Why Not Bar (11-10-2005)

Armadillo 5.x dll - Visual.Assist.X.V10.4.1640 Build 2008.05.22

Author: Computer_Angel

Link: http://www.wholetomato.com/downloads/VA_X_Setup1640.exe

Size: 4:03 MB

Type: unpack / Patch

Protection: Armadillo 5.x dll

OS: Win9x/NT/2000/XP

Date: 2005-12-10

Homepage: <http://www.wholetomato.com/index.html>

Description: Visual Assist X dramatically reduces application development time with key new features and improvements to existing features in Visual Studio

Tools use: WinXP, OllyDBG, PEid, ArmInline, ImportREC, LordPE

1.Chinh in the henh Olly:

Use Phantom Plugin to hide Olly.

2.Find magic point:

Bp in OpenMutexA API, we break in:

```
7C80EC1B> 8BFF mov edi, edi
7C80EC1D 55 push ebp
7C80EC1E 8BEC mov ebp, esp
7C80EC20 51 push ecx
7C80EC21 51 push ecx
7C80EC22 837D 10 00 Cmp dword ptr [ebp +10], 0
7C80EC26 56 push esi
7C80EC27 0F84 7A500300 je 7C843CA7
```

Nhen through the stack:

```
0012E4F4 00B2903B / Call to OpenMutexA from 00B29035
0012E4F8 001F0001 | Access = 1F0001
0012E4FC 00000000 | Inheritable = false
0012E500 0012E7BC \ MutexName = "6D417F01: SIMULATEEXPIRED"
```

Clear bp, set in hwbp API VirtualAlloc.Ta break in:

```
7C809A81> 8BFF mov edi, edi
7C809A83 55 push ebp
```

```

7C809A84 8BEC mov ebp, esp
7C809A86 FF75 14 push dword ptr [ebp +14]
7C809A89 FF75 10 push dword ptr [ebp +10]
7C809A8C FF75 0C push dword ptr [ebp + C]
7C809A8F FF75 08 push dword ptr [ebp +8]
7C809A92 6A FF push -1
7C809A94 E8 09000000 call VirtualAllocEx
7C809A99 5D pop ebp
7C809A9A C2 1000 retn 10

```

When through the stack:

```

001292C4 00B589BD / Call to VirtualAlloc from 00B589B7
001292C8 03260000 | 03260000 = Address
001292CC 0001FFC6 | Size = 1FFC6 (131,014.)
001292D0 00002000 | AllocationType = MEM_RESERVE
001292D4 00000040 \ Protect PAGE_EXECUTE_READWRITE =

```

Remember:

+ Address: 03260000 <- cove containing splicing code

Continue, hwbp clear, and set hwbp in API LoadLibraryA, we break in:

```

7C801D77> 8BFF mov edi, edi
7C801D79 55 push ebp
7C801D7A 8BEC mov ebp, esp
7C801D7C 837D 08 00 Cmp dword ptr [ebp +8], 0
7C801D80 53 push ebx
7C801D81 56 push esi
7C801D82 74 14 je short 7C801D98

```

When through the stack:

```

00B3AB51 8945 FC mov dword ptr [ebp-4], eax; SHLWAPI.77F60000
00B3AB54 6A 01 push 1
FC 00B3AB56 8B45 mov eax, dword ptr [ebp-4]
00B3AB59 50 push eax
00B3AB5A E8 B1EDFFFF call 00B39910
00B3AB5F 83C4 08 add esp, 8
00B3AB62 EB 03 jmp short 00B3AB67

```

Continue Ctrl-F9, we have:

```
00B5A9BE 8985 90D4FFFF mov dword ptr [ebp-2B70], eax; SHLWAPI.77F60000
00B5A9C4 83BD 90D4FFFF 0> Cmp dword ptr [ebp-2B70], 0
00B5A9CB 0F85 A6000000 jnz 00B5AA77
00B5A9D1 83BD 90D4FFFF 0> Cmp dword ptr [ebp-2B70], 0
00B5A9D8 75 60 jnz short 00B5AA3A
```

Here, tem command "PUSH 100":

```
00B5AFAA 68 00010000 push 100
00B5AFAF 8D8D 40C1FFFF Lea ecx, dword ptr [ebp-3EC0]
00B5AFB5 51 push ecx
00B5AFB6 8B95 40C2FFFF mov edx, dword ptr [ebp-3DC0]
00B5AFBC 8B02 mov eax, dword ptr [edx]
00B5AFBE 50 push eax
00B5AFBF E8 4C7AFBFF call 00B12A10 <- magic point
00B5AFC4 83C4 0C add esp, 0C
00B5AFC7 8D8D 40C1FFFF Lea ecx, dword ptr [ebp-3EC0]
00B5AFCD 51 push ecx
00B5AFCE 8D95 50C2FFFF Lea edx, dword ptr [ebp-3DB0]
```

Take vro 00B12A10 call, we have:

```
00B12A10 55 push ebp
00B12A11 8BEC mov ebp, esp
00B12A13 83EC 2C sub esp, 2C
00B12A16 833D 00A6B800 0> Cmp dword ptr [B8A600], 0
00B12A1D 75 59 jnz short 00B12A78
00B12A1F C745 EC A7BC16F> mov dword ptr [ebp-14], F416BCA7
```

Thrnh source code:

```
00B12A10 C3 retn
00B12A11 8BEC mov ebp, esp
00B12A13 83EC 2C sub esp, 2C
00B12A16 833D 00A6B800 0> Cmp dword ptr [B8A600], 0
```

3. Tem OEP:

Back in the EIP, tem command asm "SALC":

```
00B5B3A1 E8 2AC30000 call 00B676D0
00B5B3A6 83C4 04 add esp, 4
00B5B3A9 EB 03 jmp short 00B5B3AE
00B5B3AB D6 salc
00B5B3AC D6 salc
00B5B3AD 8B8B 0D7C4CB9 mov ecx, dword ptr [ebx + B94C7C0D]
00B5B3B3 0089 8D28C1FF add byte ptr [ecx + FFC1288D], cl
```

JMP set break at the top of the SALC, clear hwbp, Shift-F9 to run to it, then Alt-M:

Memory map

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
010A0000	00002000	Map	R	R				
03260000	00020000	Priv	RWE	RE	<-	Splicing	Memory	
1ED00000	00001000	VA_X	PE header	IMAG	R	RWE		
1ED01000	00281000	VA_X.	RWE text	IMAG	R	<-	Set on bpm execute	
1EF82000	00018000	VA_X	CODE	IMAG	R	RWE		
1EF9A000	0009E000	VA_X.	Rdata exports	IMAG	R	RWE		
1F038000	0002A000	VA_X.	IMAG data	data	R	RWE		
1F062000	00001000	VA_X	DATA	IMAG	R	RWE		
1F063000	00055000	VA_X	BSS	IMAG	R	RWE		
1F0B8000	00042000	VA_X.	Reloc	IMAG	R	RWE		
1F0FA000	00050000	VA_X.	Text1 code	IMAG	R	RWE		
1F14A000	00010000	VA_X.	ADATA code	IMAG	R	RWE		
1F15A000	00010000	VA_X.	Data1	IMAG	R	RWE		
1F16A000	00010000	VA_X.	Reloc1 relocations	IMAG	R	RWE		
1F17A000	001B0000	VA_X.	Pdata imports	IMAG	R	RWE		
1F32A000	00054000	VA_X.	Rsrc resources	IMAG	R	RWE		

Set on execute in bpm. Text section, then press Shift-F9, we will stop at OEP:

```
1EF1D944 837C24 08 01 Cmp dword ptr [esp +8], 1
1EF1D949 75 05 jnz short 1EF1D950
1EF1D94B E8 59260100 call 1EF2FFA9
1EF1D950 FF7424 04 push dword ptr [ESP +4]
1EF1D954 8B4C24 10 mov ecx, dword ptr [esp +10]
1EF1D958 8B5424 0C mov edx, dword ptr [esp + C]
```



```

1EF1D95C E8 EDFEFFFF call 1EF1D84E
1EF1D961 59 pop ecx
1EF1D962 C2 0C00 retn 0C

```

4th process Splicing code:

Running Arminline, fill vro processID, the splicing code and memory size of it, click Remove Splices

5. Dump, fix IAT:

Dung LordPE dump, and fix IAT with Imprec:

6. Patch Vissual Assist:

From notification License / Error. Trace the contrary, we have:

```

. text: 1ED480E7 call sub_1ED01730
. text: 1ED480EC test BL, BL
. text: 1ED480EE jz short loc_1ED48130
. text: 1ED480F0 mov ecx, dword_1F04A1E0
. text: 1ED480F6 push 0; uType
. text: 1ED480F8 push offset aLicense; "License"
. text: 1ED480FD push offset aError; "Error"
. text: 1ED48102 push ecx; hWnd
. text: 1ED48103 call ds: __imp_MessageBoxA

```

Since then, the points tem's function check:

```

. text: 1ED47F80 push 0FFFFFFFFh
. text: 1ED47F82 push offset unk_1EF442A0
. text: 1ED47F87 mov eax, large fs: 0
. text: 1ED47F8D push eax
. text: 1ED47F8E sub esp, 34h

```

Patch thrnh:

```

mov eax, 1
retn

```

Game Over!

Armadillo dll - Unpacking and More

Contents Table:

I. Intro &

Preparation:

Magic Jump II.Fix and Find OEP:

III.Find & Fix CodeSplicing:

IV.Find & Fix IAT elimination:

V. rebuild Import:

ImageBase VI.Fix & Relocation:

VII.Reduce Size:

VIII.Ending:

Tools: **Hide**
plugins OllyDBG
++ Scripts,
LordPE,
ImportREC 1.6,
CFF Explorer

Skill Request:

Using Basic
Knowledge Olly +
manual unpacking
Armadillo.

I-Intro & Preparation:

Hi everyone, today I tricky to complete more of tut unpack dll for Armadillo. Can you read each of the hacnho tut, tut of the MaDMaN ARTeam, or by the ICU Whiterat by tricky ... but at the tut then, a few points that do not make many say you're having trouble during unpack dll. It points out this very basic, the majority or be ignored, but i do with this article, hope you can confidently resolve dll's Armadillo. But before you go to the main issues, we need to review the format type Protect Standard, Code Splicing IAT and elimination of the dll, Armadillo applies only to protect this form.

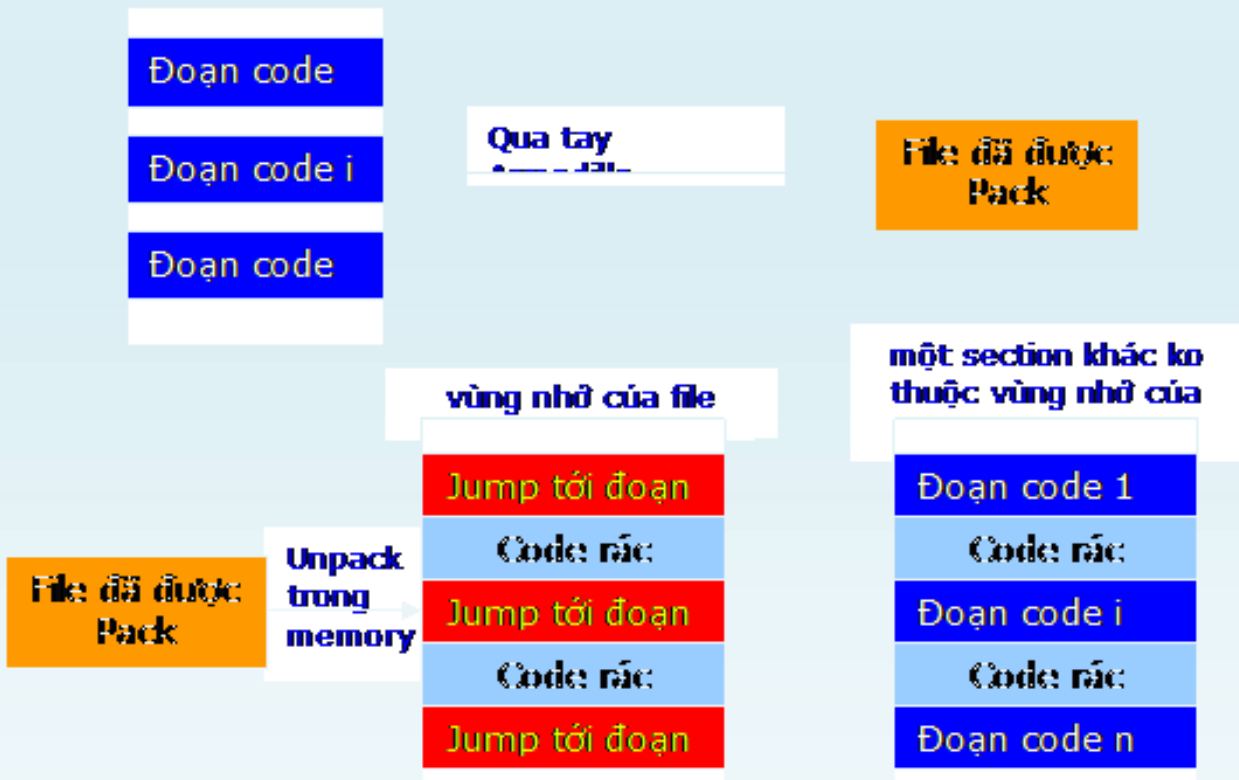
Usually we only unpack the familiar hands of foreign tut, sometimes done in the beginning that you do not know what is. This is easy to change, we have an 1. On the other hand, if you look through the science , sometimes you xài a familiar formula that, solve a series of items that have never thought to demonstrate how to start? Every thinking person is found, the formula must have the xài Yes. However, if i could find himself "the truth", they at least must also understand the way that we have to find out and understand what they are doing. From the back issues: Standard Protect What? Splicing or IAT code elimination is what? Here if you would like to understand how fast and streamlined for not free of them:

Standard Protection



You see any PE file will need to IAT (more structured PE file). When the Pack, Armadillo will Encrypt to unpack time in memory, IAT will be so wrong with the original, some seats were destroyed by the Byte encryption. So when the dump at OEP, we just dump the IAT is the bay, while also ko Armadillo out how to decrypt it again. The IAT is canceled by one (or several) orders that do not jump to perform (or be does). Therefore, it should find Magic Jump to patch it always happens (or do not occur), then we will be the first original IAT.

Code Splicing



From the pictures we see after unpack in memory, the original code in the file, now has rock out another section, accompanied spam code (code meaningless, do not change the value of how to or write variables ...). In the file is replaced by the command Jump to code in that section (plus garbage code). Therefore, if you dump in the OEP, we just dump the command and lack Jump lose some of the very first code.

There are 2 ways to fix. The dump is one more section contains the code is missing, then connect it to the dump file was in OEP. 2 hours, according to our men Jump command, filter out spam code, copy and paste the code actually return in order to jump. In the hours are 1 tool to quickly ArmTool is, in 2 hours there ArmInline . According tricky, 2 hours or more, because it creates the same structure of the file before the pack. However, 1 month and to be less than survival.

IAT elimination

2 models on the protection, you can quickly understand in this section. After unpack in memory, instead of the IAT is in the file, then the stone is out section 1 ko under other file. Should the dump, we are surviving the loss of the IAT. How to fix the same as in Splicing Code, section or dump it out, or bring back the value of 1 IAT section is available in the file, usually selected. ADATA. And candidates Incandescent is ArmInline because it works very fast and quite accurate.

But more needs to say, why is. ADATA? other section can be or do? Ah, the pack again, Armadillo create section for the unpack is. Text1. Reloc1. ADATA. Data1. Pdata. After unpack, the section that i have used more and section. ADATA have relatively large size compared to the original size of the IAT, it is often chosen as the place to redirect to the IAT. You can select any section which to redirect IAT, the free copy of it available to appropriate size of IAT. Bit more tricky for the man you will see.

Okie, Thanksgiving has enough items now xấn arms and carried unpack dll pack by Armadillo. Target kì this file is **UKHook40.DLL's** Unikey but the latest version 4 Unikey RC1.

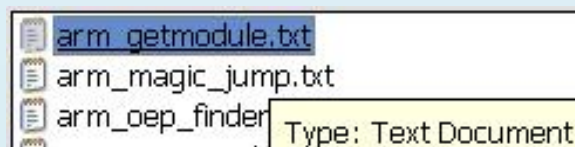
II-Fix and Magic Jump Find OEP:

Dll file is as PE. EXE so you can load to unpack and Olly normal, it will have to borrow through LoadDLL created by Olly 1 Main. Exe to load it into memory.

What we started in EP, you have to Hide the Anti-Debug Armadillo's farewell (IsDebuggerPresent, OutputDebugStringA), check all ... Exception:

Address	Hex dump	Disassembly
00800897	55	PUSH EBP
00800898	8BEC	MOV EBP,ESP
0080089A	53	PUSH EBX
0080089B	8B5D 08	MOV EBX,DWORD PTR SS:[EBP+8]
0080089E	56	PUSH ESI
0080089F	8B75 0C	MOV ESI,DWORD PTR SS:[EBP+C]
008008A2	57	PUSH EDI
008008A3	8B7D 10	MOV EDI,DWORD PTR SS:[EBP+10]
008008A6	85F6	TEST ESI,ESI
008008A8	75 09	JNZ SHORT UKHook40.008008B3
008008AA	833D 0C878F0	CMP DWORD PTR DS:[8F87DC],0
008008B1	EB 26	JNE SHORT UKHook40.008008D9

It can be used "he GetModuleHandleA" or scripts to fix Magic Jump. Here Scripts used for fast (as you are too familiar to you unpack this file. Exe):



After the script is finished running, magic jump found and fix:

Address	Hex dump	Disassembly	Comment
00A05FC9	8B0D AC40A300	MOV ECX,DWORD PTR DS:[A340AC]	
00A05FCF	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00A05FD2	A1 AC40A300	MOV EAX,DWORD PTR DS:[A340AC]	
00A05FD7	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00A05FDA	75 16	JNZ SHORT 00A05FF2	
00A05FDC	8D85 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
00A05FE2	50	PUSH EAX	
00A05FE3	FF15 BC62A200	CALL DWORD PTR DS:[A262BC]	kernel32.LoadLibraryA
00A05FE9	8B0D AC40A300	MOV ECX,DWORD PTR DS:[A340AC]	
00A05FEF	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00A05FF2	A1 AC40A300	MOV EAX,DWORD PTR DS:[A340AC]	
00A05FF7	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00A05FFA	E9 30010000	JMP 00A0612F	<- MagicJump fixed
00A05FFF	0033	ADD BYTE PTR DS:[EBX],DH	
00A06001	C9	LEAVE	
00A06002	8B07	MOV EAX,DWORD PTR DS:[EDI]	
00A06004	3918	CMP DWORD PTR DS:[EAX],EBX	
00A06006	74 06	JE SHORT 00A0600E	

Hours Goto CreateThread to function, Ctrl-G:



To limit the Anti-Debug, we do not set at the top of BP function that should be set at the function that RETN command:

Address	Hex dump	Disassembly
7C81082F	8BFF	MOV EDI,EDI
7C810831	55	PUSH EBP
7C810832	8BEC	MOV EBP,ESP
7C810834	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
7C810837	FF75 18	PUSH DWORD PTR SS:[EBP+18]
7C81083A	FF75 14	PUSH DWORD PTR SS:[EBP+14]
7C81083D	FF75 10	PUSH DWORD PTR SS:[EBP+10]
7C810840	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
7C810843	FF75 08	PUSH DWORD PTR SS:[EBP+8]
7C810846	6A FF	PUSH -1
7C810848	E8 D9FDFFFF	CALL kernel32.CreateRemoteThread
7C81084D	5D	POP EBP
7C81084E	C2 1800	RETN 18
7C810851	90	NOP

F9 to Break there. *If there is any machine running the script and then the crash, the resolution you nhé, can use the Trial Reset Scan file. Tmp the Armadillo and delete it and then try to run the script to see why. Un-BP go, F7 (or F8) to return to:*

Address	Hex dump	Disassembly	Comment
00A0C4E1	50	PUSH EAX	
00A0C4E2	FF15 4C62A200	CALL DWORD PTR DS:[A2624C]	kernel32.CloseHandle
00A0C4E8	6A 0A	PUSH 0A	
00A0C4EA	FF15 B462A200	CALL DWORD PTR DS:[A262B4]	kernel32.Sleep
00A0C4F0	EB 33	JMP SHORT 00A0C525	
00A0C4F2	E8 C9090000	CALL 00A0CEC0	
00A0C4F7	84C0	TEST AL,AL	
00A0C4F9	74 10	JE SHORT 00A0C50B	
00A0C4FB	E8 9367FFFF	CALL 00A02C93	
00A0C500	85C0	TEST EAX,EAX	
00A0C502	75 07	JNZ SHORT 00A0C50B	

This is the area code for decrypt the Armadillo. We do need more attention, it should also do more to trace tired, because the area code from this, the first value that it is access to OEP. Therefore, open to the Memory Map, set [memory breakpoint on access](#) at the section. Text:

Address	Hex dump	Disassembly	Comment
00870000	00001000	UKHook40	
00871000	0001E000	UKHook40	
0088F000	00006000	UKHook40	
00895000	0000C000	UKHook40	
008A1000	00003000	UKHook40	
008A4000	00040000	UKHook40	
008E4000	00010000	UKHook40	
008F4000	00010000	UKHook40	
00904000	00010000	UKHook40	
00914000	000B0000	UKHook40	
009D0000	00004000	UKHook40	
009E0000	00002000	UKHook40	
009F0000	00064000	UKHook40	
00A60000	0000C000	UKHook40	
00A70000	00006000	UKHook40	
00A80000	000F8000	UKHook40	
00BB0000	00001000	UKHook40	
00BB5000	00002000	UKHook40	

F9 a new, right OEP:

Address	Hex dump	Disassembly	Comment
008807E9	6A 0C	PUSH 0C	<-- OEP here
008807EB	68 20FA8900	PUSH UKHook40.0088FA20	
008807F0	E8 232B0000	CALL UKHook40.00883318	
008807F5	33C0	XOR EAX,EAX	
008807F7	40	INC EAX	
008807F8	8945 E4	MOV DWORD PTR SS:[EBP-1C],EAX	
008807FB	8B75 0C	MOV ESI,DWORD PTR SS:[EBP+C]	
008807FE	33FF	XOR EDI,EDI	
00880800	3B57	CMPS ESI,EDI	

So have completed the fix to get Magic Jump Full IAT and OEP fast lê.

III-Find & Fix CodeSplicing:

Now we'll find out this dll protect with CodeSplicing (CP) or ko? How easy to find, *at the OEP, you just enter the command CALL nearby, if you see a command - JMP to offset an outside file ie it will have CP.* Here we enter on the command CALL OEP and found:

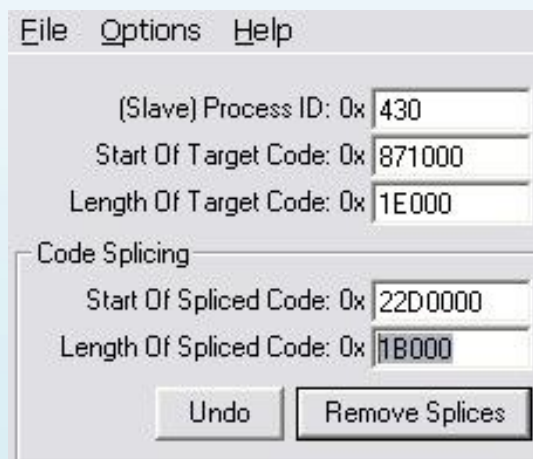
Address	Hex dump	Disassembly
00883318	68 6C338900	PUSH UKHook40.0088336C
0088331D	64:A1 00000000	MOV EAX,PUSHP PTR FS:[0]
00883323	- E9 1E1AA501	JMP 022D4D46
00883328	66:87FE	XCHG EAX,EDX
0088332B	92	XCHG EAX,EDX
0088332C	92	XCHG EAX,EDX
0088332D	66:87FE	XCHG SI,DI
00883330	2BE0	CALL EBX
00883332	- E9 291AA501	JMP 022D4D60
00883337	0FCB	POPCB EBX
00883339	87F2	XCHG EDX,ESI
0088333B	87CB	XCHG EBX,ECX

Two command JMP see in the picture is No. 2 in command of JMP to the outside section of the file containing the code was. We found the address 22Dxxxx form, open the memory map and find the section in this store Memory:

00570000	00118000				Map	R	E	R	E
00870000	00001000	UKHook40			Image	R		RWE	
00871000	0001E000	UKHook40			Image	R		RWE	
0088F000	00006000	UKHook40	.rdata	exports	Image	R		RWE	
00895000	0000C000	UKHook40	.data	data	Image	R		RWE	
008A1000	00003000	UKHook40	.reloc		Image	R		RWE	
008A4000	00040000	UKHook40	.text1	code	Image	R		RWE	
008E4000	00010000	UKHook40	.adata	code	Image	R		RWE	
008F4000	00010000	UKHook40	.data1		Image	R		RWE	
00904000	00010000	UKHook40	.reloc1	relocations	Image	R		RWE	
00914000	000B0000	UKHook40	.pdata	imports	Image	R		RWE	
009D0000	00004000				Priv	RW		RW	
009E0000	00002000				Map	R		R	
009F0000	00064000				Priv	RW		RW	
00A60000	0000C000				Priv	RW		RW	
00A70000	00006000				Priv	RW		RW	
00A80000	000F8000				Priv	RW		RW	
00BB0000	00001000				Priv	RW	Guar	RW	
00BBE000	00002000			stack of th	Priv	RW	Guar	RW	
00BC0000	00007000				Priv	RW		RW	
00C00000	00001000				Map	RW		RW	
00C10000	00001000				Map	RW		RW	
00C20000	00001000				Priv	RW		RW	
00C41000	00001000				Priv	RW		RW	
00C61000	00022000				Priv	RW		RW	
00E20000	00001000				Priv	RW		RW	
00E30000	00001000				Priv	RW		RW	
00E7D000	00001000				Priv	RW	Guar	RW	
00E7E000	00002000			stack of th	Priv	RW	Guar	RW	
22D00000	0001B000				Priv	R	E	RWE	

So address start of this section is 22D00000 and size is 1B000. There are many you understand CodeSplicing machines are always located in the same position and size is always 20,000. Here you know exactly how to find it then. Now open up ArmInline to fix, version 0.95 is tricky to use. Wait ArmInline get process list, select loadll.exe, the modules section, select UKHook40.dll.

Then the parties Stocks get information that we find the comparison with the ArmInline, where it has taken the size of the Code Splicing, but we feel careful again to fix:



Now click the Remove Splices then. Fix lickerish, including any surviving command JMP:

```
Analysing module...
Searching for Code Splicing...
Done (1 potential pages found). Scanning...
Code Splicing section located.
Locating IAT...
Success. Determining IAT hive...
A possible IAT has been located.

----- Code Splicing -----
Process memory buffered successfully.
1182 splices repaired...
1536 splices repaired.
Splice repairing complete. Patching process...
Patch succesful.
```

Okie, in the order we always fix it before the new Government to IAT elimination. Now continue nhé.

IV-Find & Fix IAT elimination:

To check for IAT elimination or do, I mon men under the command CALL near OEP, until an order form found JMP [IAT] or CALL [IAT]. Here, from the OEP, to enter the command CALL 3:



I see a command CALL [IAT]:

00880670	E8 6E0C0000	CALL UKHook40.008812F0	
00880682	8BF4	MOV ESI,ESP	
00880684	56	PUSH ESI	
00880685	C706 94000000	MOV DWORD PTR DS:[ESI],94	
00880688	FF15 9C13C600	CALL DWORD PTR DS:[C6139C]	kernel32.GetVersionExA
00880691	85C0	TEST EAX,EAX	
00880693	7F84 34010000	JE UKHook40.008807CD	

Address contains the IAT is located outside the area that memory to load the file. Memory View to see:

00450000	00103000				Map	R	R
00457000	00133000				Map	R	E
00870000	00001000	UKHook40			Imag	R	RWE
00871000	0001E000	UKHook40	.text		Imag	R	RWE
0088F000	00006000	UKHook40	.rdata	exports	Imag	R	RWE
00895000	0000C000	UKHook40	.data	data	Imag	R	RWE
008A1000	00003000	UKHook40	.reloc		Imag	R	RWE
008A4000	00040000	UKHook40	.text1	code	Imag	R	RWE
008E4000	00010000	UKHook40	.adata	code	Imag	R	RWE
008F4000	00010000	UKHook40	.data1		Imag	R	RWE
00904000	00010000	UKHook40	.reloc1	relocations	Imag	R	RWE
00914000	000B0000	UKHook40	.pdata	imports	Imag	R	RWE
009D0000	00004000				Priv	RW	RW
009E0000	00002000				Map	R	R
009F0000	00064000				Priv	RW	RW
00A60000	0000C000				Priv	RW	RW
00A70000	00006000				Priv	RW	RW
00A80000	000F8000				Priv	RW	RW
00BB0000	00001000				Priv	RW	Guar
00BBE000	00002000			stack of th	Priv	RW	Guar
00BC0000	00007000				Priv	RW	RW
00C00000	00001000				Map	RW	RW
00C10000	00001000				Map	RW	RW
00C20000	00001000				Priv	RW	RW
00C61000	00022000				Priv	RW	RW

Therefore, it may conclude that there IAT elimination. Back on the CALL command, Click right, [Follow dump](#) it in, drag up and down for IAT Start and End:

Address	Value	Comment
00C611E4	021C0704	
00C611E8	7C80CEC4	kernel32.LCMapStringW
00C611EC	00A07499	
00C611F0	7C80CD58	kernel32.FlushFileBuffers
00C611F4	7C812AC6	kernel32.GetSystemInfo
00C611F8	00A07359	
00C611FC	00A07362	
00C61200	7C80C729	kernel32.lstrcpyA
00C61204	7C9179FD	ntdll.RtlReAllocateHeap
00C61208	00A073C7	
00C6120C	7C801755	kernel32.GetSystemTimeAsFileTime

IAT Start

Address	Value	Comment
00C615A8	00A074DC	
00C615AC	77D4C640	USER32.GetFocus
00C615B0	00A07524	
00C615B4	00A074EE	
00C615B8	00A07465	
00C615BC	7C8112E3	kernel32.IsValidCodePage
00C615C0	7C80180E	kernel32.ReadFile
00C615C4	ABABABAB	
00C615C8	ABABABAB	
00C615CC	FFFFFFEE	
00C615D0	00000000	

IAT End

So in C611E8 IAT Start and End in C615C0. But for the length of the code contains IAT, we must get the address below IAT End to go except that $C615C4 = \text{Length} - C611E8 = 3DC$

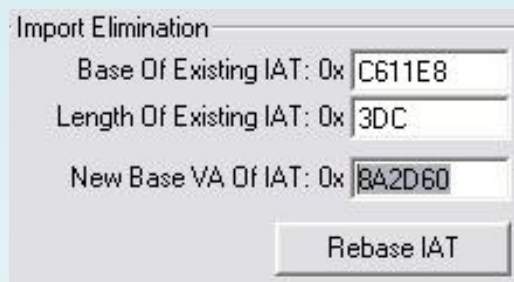
Back ArmInline, to fill the IAT elimination but that it itself has not taken correctly, here is the Base (Start IAT) and Length:

00570000	00148000				Map	R	E	R	E
00870000	00001000	UKHook40			Image	R		RWE	
00871000	0001E000	UKHook40	.text		Image	R		RWE	
0088F000	00006000	UKHook40	.rdata	exports	Image	R		RWE	
00895000	0000C000	UKHook40	.data	data	Image	R		RWE	
008A1000	00003000	UKHook40	.reloc		Image	R		RWE	
008A4000	00048000	UKHook40	.text1	code	Image	R		RWE	
008E4000	00010000	UKHook40	.adata	code	Image	R		RWE	
008F4000	00010000	UKHook40	.data1		Image	R		RWE	
00904000	00010000	UKHook40	.reloc1	relocations	Image	R		RWE	
00914000	00080000	UKHook40	.rdata	imports	Image	R		RWE	
00918000	00008000				EXD	RW		RW	

[illegible]

008A2C80	00	39	04	39	08	39	0C	39	10	39	14	39	20	39	30	39	49	59	69	79	89	90			
008A2C90	34	39	38	39	F4	39	FC	39	04	3A	0C	3A	14	3A	1C	3A	4989	99	99	99	90	90			
008A2CA0	24	3A	2C	3A	34	3A	3C	3A	44	3A	4C	3A	54	3A	5C	3A	\$:	4	<	D	L	T	:	\
008A2CB0	64	3A	6C	3A	74	3A	7C	3A	84	3A	88	3A	8C	3A	90	3A	d	:	t	:	:	:	:	:	:
008A2CC0	94	3A	98	3A	9C	3A	90	3A	A4	3A	B8	3A	C0	3A	C4	3A	0	:	:	:	:	:	:	:	:
008A2CD0	C8	3A	CC	3A	D0	3A	D4	3A	D8	3A	DC	3A	E0	3A	E4	3A	4	:	:	:	:	:	:	:	:
008A2CE0	E8	3A	EC	3A	F0	3A	F4	3A	F8	3A	FC	3A	00	3B	04	3B	4	:	:	:	:	:	:	:	:
008A2CF0	08	3B	0C	3B	10	3B	14	3B	18	3B	1C	3B	20	3B	24	3B	[:	:	:	:	:	:	:	:
008A2D00	28	3B	2C	3B	30	3B	34	3B	38	3B	3C	3B	40	3B	44	3B	(:	:	:	:	:	:	:	:
008A2D10	48	3B	4C	3B	50	3B	54	3B	58	3B	5C	3B	60	3B	64	3B	H	:	:	:	:	:	:	:	:
008A2D20	68	3B	40	3D	44	3D	00	00	00	00	00	00	00	00	00	00	h	:	:	:	:	:	:	:	:
008A2D30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:
008A2D40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:
008A2D50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:
008A2D60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:
008A2D70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:
008A2D80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:
008A2D90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:
008A2DA0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:
008A2DB0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:
008A2DC0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:
008A2DD0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:
008A2DE0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:
008A2DF0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	:	:	:	:	:	:	:	:

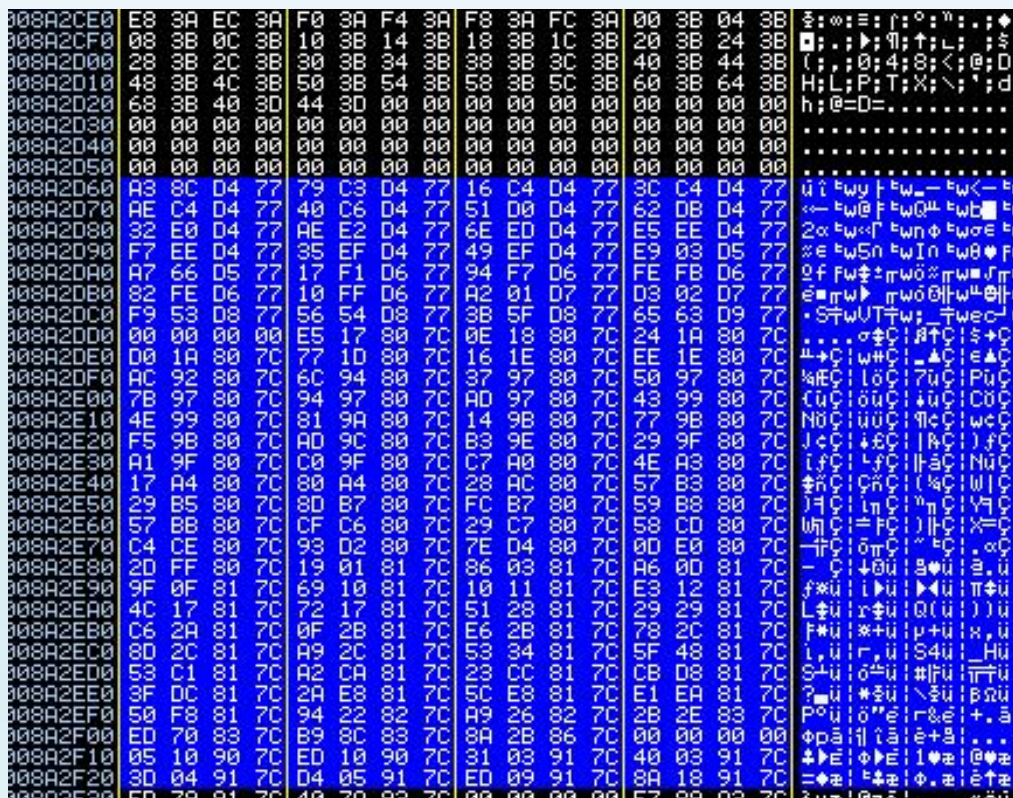
But a lot of space to the end of this section. And of course it larger 3DC. Tricky selection only begins to redirect IAT in 8A2D60. Remember that on you, the hours will offset nay other jog. Back ArmInline, enter New Base have chosen to:



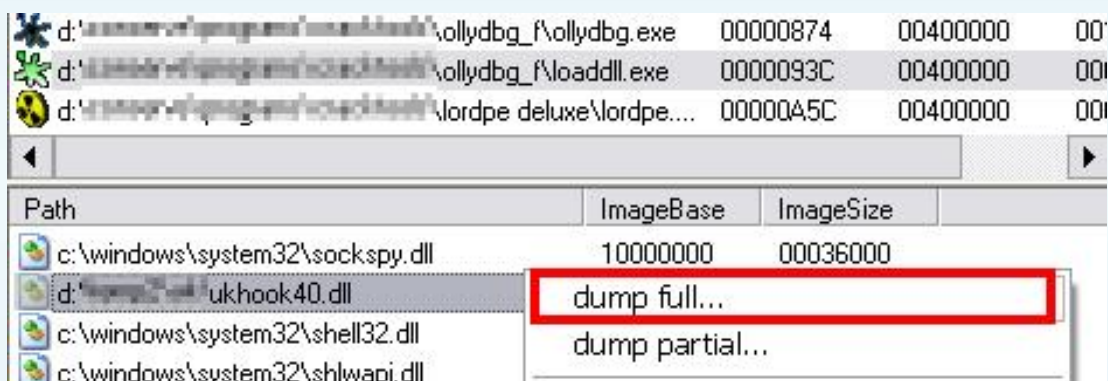
Can click Rebase IAT was then. Lickerish Redirect:

```
----- Rebasing IAT -----
Process memory buffered successfully.
290 DLL calls found total.
Analysing...
117 API functions referenced from 4 DLLs.
Redirecting DLL references:
290 calls redirected total.
Patching process...
Process successfully patched.
```

Close ArmInline go back section of the **dump. Reloc** this, we see Byte by IAT was to redirect:



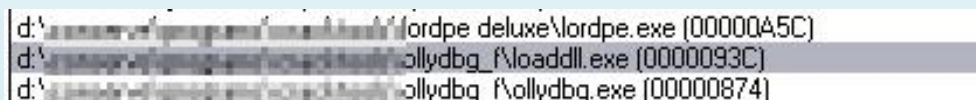
Okie, all healthy appetite, we open up and dump LordPE you go. Loadll.exe Select the process and the modules below, choose UKHook40.dll (similar in choosing ArmInline). Click to full-dump in this module.



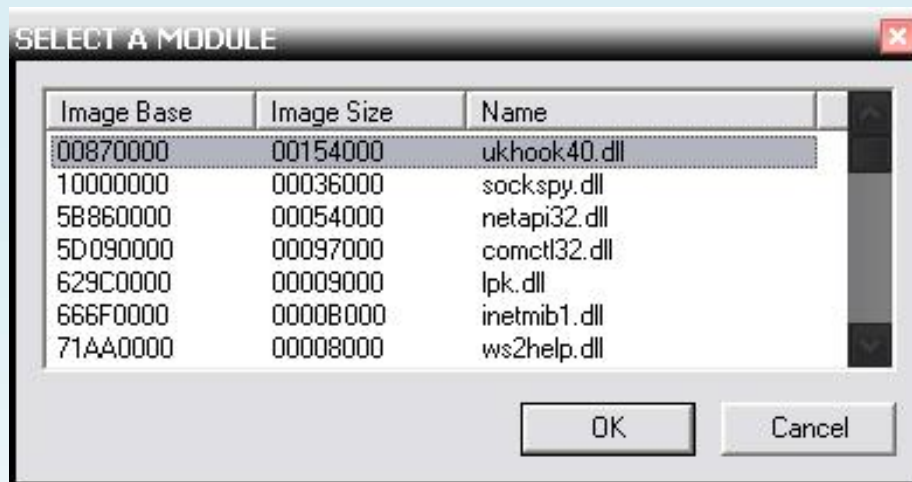
Save the file. (retain the Olly nhé)

V-Import rebuild:

Open up ImportREC. Select load.dll.exe in the process list:



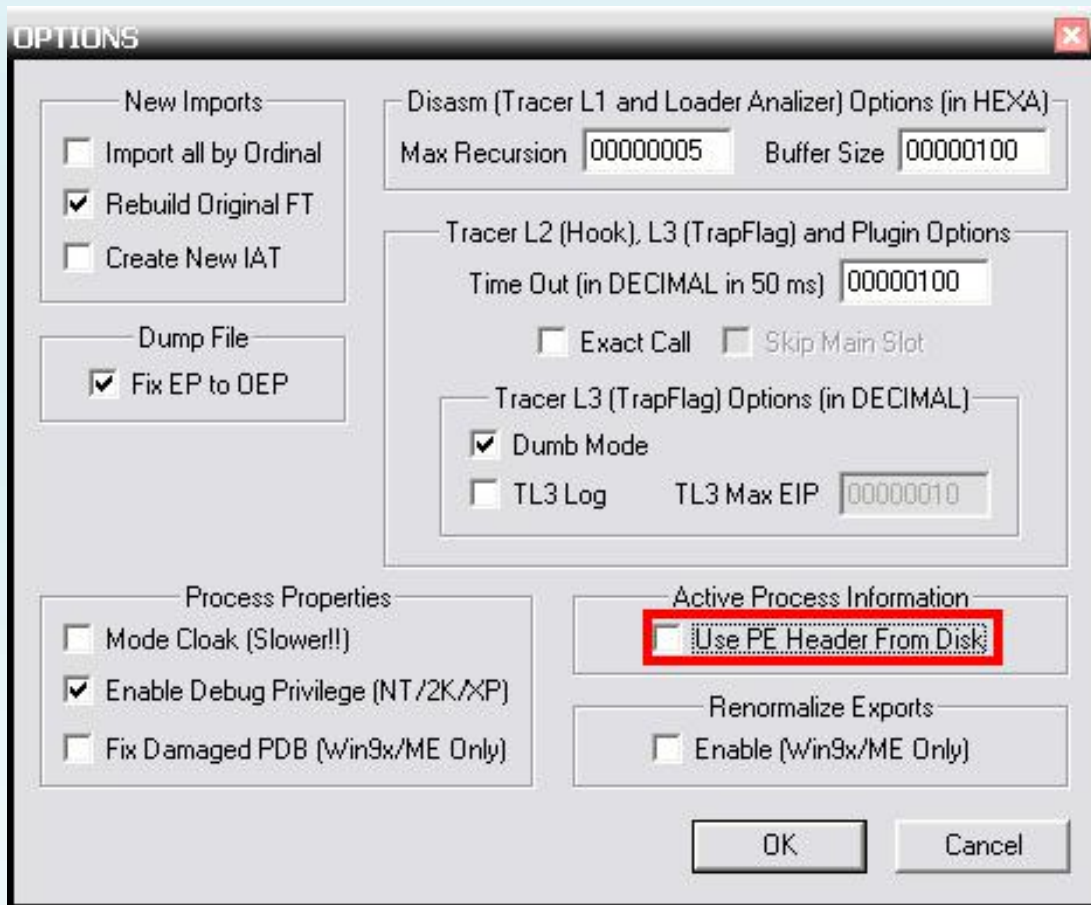
Then Click the button dll Pick:



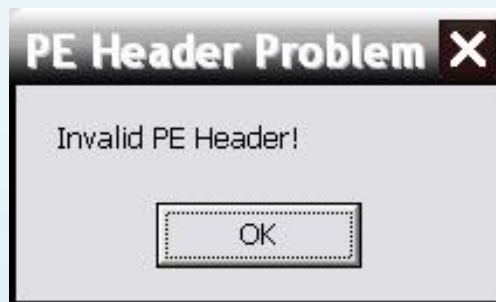
We see this time in memory of ImageBase UKHook40.dll is 870,000. But when it is OK:

Image Base:10000000 Size:00154000

ImageBase = 10000000. Very simply, this is ImageBase original file this. ImportREC has obtained this information from the PE Header file on disk to do so in memory of the modules, so when calculating RVA to Fix Import, it is easy to frame and or error. Often you also having problems or where exactly do? Okie, we open up ImportREC Option:



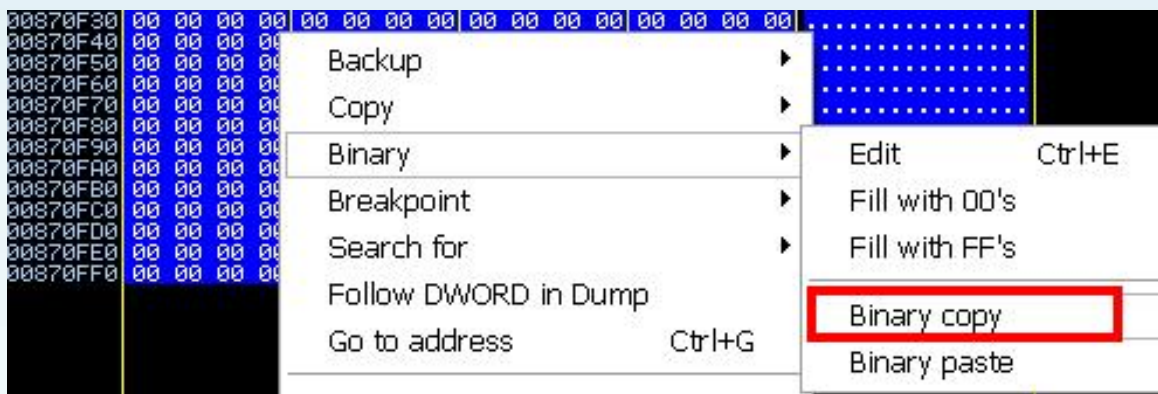
In their place, you **uncheck** it. If you check there, ImportREC always get information from the PE header disk and calculates wrong. I just check in when unpack the file. Exe. Why? Because. Exe usually load into memory properly ImageBase with its original, but the dll ko. OK, one hour Pick dll to do:



Now back to this notice. Easy to understand all, unpack the file. Exe, we check in on the Option and it always get information from PE Header file of the disc, the PE Header time in memory has been destroyed Armadillo when we go to OEP. Therefore ImportREC ko ImageBase get more time to load it into a file UKHook40.dll other Olly, open memory map, select PE Header intact xi:

00870000	00001000	UKHook40	PE header	Imag	R	RWE
00871000	0001E000	UKHook40	.text	Imag	R	RWE
0088F000	00006000	UKHook40	.rdata	Imag	R	RWE
00895000	0000C000	UKHook40	.data	Imag	R	RWE
008A1000	00003000	UKHook40	.reloc	Imag	R	RWE

Double-click on it, to all bytes of this section, click Copy to Binary à :



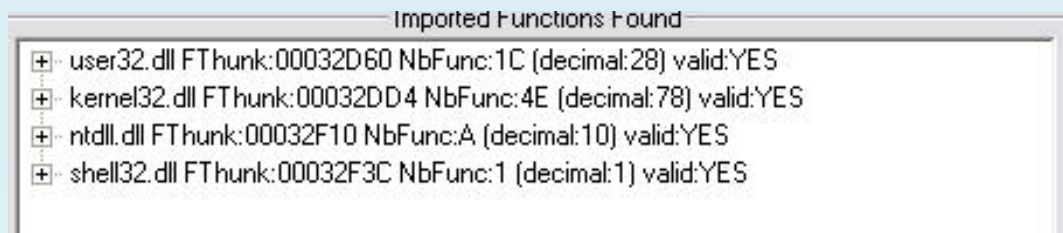
Close this new Olly, Olly back old Open memory map, select the section at 870,000, to end this section and click to paste that into Binary. PE header was back:

00870000	00001000	UKHook40	PE header	Image	R	RWE
00871000	0001E000	UKHook40	.text	Image	R	RWE
0088F000	00006000	UKHook40	.rdata	Image	R	RWE
00895000	0000C000	UKHook40	.data	Image	R	RWE
008A1000	00003000	UKHook40	.reloc	Image	R	RWE
008A4000	00040000	UKHook40	.text1	Image	R	RWE
008E4000	00010000	UKHook40	.adata	Image	R	RWE
008F4000	00010000	UKHook40	.data1	Image	R	RWE
00904000	00010000	UKHook40	.reloc1	Image	R	RWE

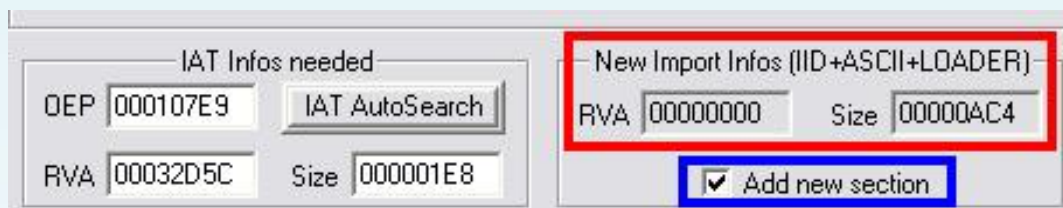
The above are familiar with each considered the beginning of the tut hacnho. Now switch ImportREC, dll Pick again:

Image Base:00870000 Size:00154000

Enter OEP = $8807E9 - 870,000 = 107E9$.Nhan IAT Autostart and Import Get, get it healthy appetite:



Now drilling Fix dump run. for us to this:



ImportREC will add 1 section. mackt to dump file if you check [Add New section](#) here. Section by ImportREC often add size to 4000, the balance of space after the Import and fix the IAT that ArmInline redirect (*if you can check Create New IAT*). But we really do need to check this Option. Now, notice that size ImportREC need to fix the dump file is AC4. We turn over in Olly memory map, select the **section. Reloc** (from the beginning now, Olly this hê Close nhé ko):

[illegible]

New Import Infos (IID+ASCII+LOADER)

RVA Size

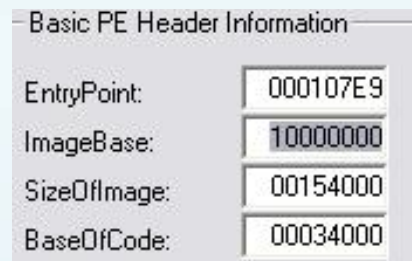
☐ Add new section

LA-Fix ImageBase & Relocation:

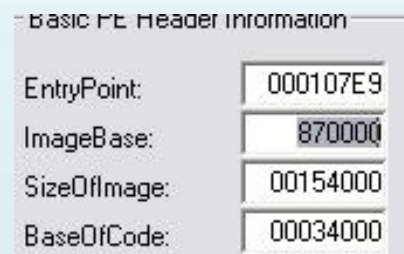
file:///C:/RCE Unpacking eBook [Translated by LithiumLi]/Armadillo DLL – Unpacking and MORE.htm (14 of 22) [1/9/2009 9:43:49 LithiumLi]

ImageBase: địa chỉ nạp "được ưu tiên" cho PE file. Lấy ví dụ : Nếu như giá trị trong trường này là 400000h, PE Loader sẽ cố gắng nạp file vào trong không gian địa chỉ ảo bắt đầu tại 400000h. Từ "được ưu tiên" ở đây có nghĩa là PE Loader không thể nạp file vào địa chỉ đó nếu như có một module nào khác đã chiếm giữ vùng địa chỉ này. 99 % các trường hợp giá trị của ImageBase luôn là 400000h

Remember now we dump in ImageBase 870,000, Import and fix that, should the need for adjustment ImageBase as the dump. Open up LordPE, PE Editor, select the file you just fix Import this:



Edit to 870,000:



Click Save, close LordPE, as a complete fix ImageBase. However, the file has not run well, we come to the concept 2:

Base Relocations Section :

Khi mà trình linker tạo ra một file Exe, nó chuẩn bị một nơi mà tại đó file sẽ được ánh xạ vào trong bộ nhớ. Dựa trên điều này, trình linker sẽ đặt các địa chỉ thật của đoạn mã và những mục dữ liệu vào trong file thực thi. Nếu vì bất cứ lý do gì file thực thi kết thúc quá trình nạp ở một nơi nào đó nếu không trong phạm vi không gian địa chỉ ảo, thì những địa chỉ này sẽ bị trình linker đặt vào trong image không đúng. Thông tin được lưu trong section `.reloc` cho phép trình PE loader fix những địa chỉ này trong loaded image và vậy chúng sẽ lại chính xác. Mặt khác, nếu trình loader có thể nạp file tại những địa chỉ base addresses được thừa nhận bởi trình linker, thì dữ liệu ở section `.reloc` là không cần thiết và bị bỏ đi.

In the packer, most are compress / encrypt Relocation of the file. But at the Armadillo, it do not do it, just replace it with a Relocation to another for the packer. The **section. Reloc1**. Relocation section also contains the **original. Reloc** to do is encrypt anything. When the pack file. Exe, Armadillo Relocation is still to **section.**

Reloc. but the dll, it does **section. Reloc1**. Should we need to properly fix the Relocation of the dll file after unpack. Now use Olly, load the file after the fix. Open memory map:

00870000	00010000	UKHook40	PE header	Map	R	RWE
00871000	0001E000	UKHook40	.text	Imag	R	RWE
0088F000	00006000	UKHook40	.rdata	Imag	R	RWE
00895000	0000C000	UKHook40	exports	Imag	R	RWE
008A1000	00003000	UKHook40	data	Imag	R	RWE
008A4000	00040000	UKHook40	imports	Imag	R	RWE
008E4000	00010000	UKHook40	code	Imag	R	RWE
008F4000	00010000	UKHook40	code	Imag	R	RWE
00904000	00010000	UKHook40	code	Imag	R	RWE
00914000	000B0000	UKHook40	relocations	Imag	R	RWE
009D0000	00004000	UKHook40	.pdata	Priv	RW	RW

We found that, with Olly unpack that we use, this file is always load in ImageBase 870,000. Return to the main UPC. Enter function Call 3 under OEP, we see a call CALL [IAT] at this precise:

00880670	E8 6E0C0000	CALL UKHook40.008812F0	
00880682	8BF4	MOV ESI,ESP	
00880684	56	PUSH ESI	
00880685	C706 94000000	MOV DWORD PTR DS:[ESI],94	
00880688	FF15 A82E8A00	CALL DWORD PTR DS:[&kernel32.GetVersionExA]	kernel32.GetVersionExA
00880691	85C0	TEST EAX,EAX	
00880693	0F84 34010000	JE UKHook40.008807CD	
00880699	8B4E 10	MOV ECX,DWORD PTR DS:[ESI+10]	

Keep the load more files into a "fish and Olly." Here tricky selected by Olly Diablo, you can be home to one of the aged that down on. Why in particular, because it always load dll file to another ImageBase with Olly there:

00870000	00049000			Map	RW	RW
008C0000	00001000	UKHook40	PE header	Imag	R	RWE
008C1000	0001E000	UKHook40	code	Imag	R	RWE
008DF000	00006000	UKHook40	code, exports	Imag	R	RWE
008E5000	0000C000	UKHook40	code, data	Imag	R	RWE
008F1000	00003000	UKHook40	code, imports	Imag	R	RWE
008F4000	00040000	UKHook40	code	Imag	R	RWE
00934000	00010000	UKHook40	code	Imag	R	RWE
00944000	00010000	UKHook40	code	Imag	R	RWE
00954000	00010000	UKHook40	code, relocations	Imag	R	RWE
00964000	000B0000	UKHook40	code	Imag	R	RWE

ImageBase = 8C0000. Since OEP, enter the function call 3, we see CALL [IAT] was wrong on this bet:

008D0668	55	PUSH EBP	
008D0669	8BEC	MOV EBP,ESP	
008D066B	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]	
008D066E	83F8 01	CMP EAX,1	
008D0671	56	PUSH ESI	
008D0672	0F85 E0000000	JNZ 008D0758	
008D0678	B8 94000000	MOV EAX,94	
008D067D	E8 6E0C0000	CALL 008D12F0	
008D0682	8BF4	MOV ESI,ESP	
008D0684	56	PUSH ESI	
008D0685	C706 94000000	MOV DWORD PTR DS:[ESI],94	
008D0688	FF15 A82E8A00	CALL DWORD PTR DS:[8A2E8A8]	
008D0691	85C0	TEST EAX,EAX	
008D0693	0F84 34010000	JE 008D07CD	

Based on explanations of concepts Relocation, sure you understand why? ImageBase now have another, but because Relocation should it be wrong ko load IAT right position. When you run the file that i have Olly, ImageBase then the other, IAT ko load is the right place, so do file will run, or run, but failed. Okie, Olly close of the Diablo. Back Olly there, select Memory in PE Header map:

00870000	4D 5A	ASCII "MZ"	DOS EXE Signature
00870002	9000	DW 0090	DOS_PartPag = 90 (144.)
00870004	0300	DW 0003	DOS_PageCnt = 3
00870006	0000	DW 0000	DOS_ReloCnt = 0
00870008	0400	DW 0004	DOS_HdrSize = 4
0087000A	0000	DW 0000	DOS_MinMem = 0
0087000C	FFFF	DW FFFF	DOS_MaxMem = FFFF (65535.)
0087000E	0000	DW 0000	DOS_ReloSS = 0
00870010	B800	DW 00B8	DOS_ExeSP = B8
00870012	0000	DW 0000	DOS_ChkSum = 0
00870014	0000	DW 0000	DOS_ExeIP = 0
00870016	0000	DW 0000	DOS_ReloCS = 0
00870018	4000	DW 0040	DOS_Tabloff = 40
0087001A	0000	DW 0000	DOS_Overlay = 0
0087001C	00	DB 00	
0087001D	00	DB 00	
0087001E	00	DB 00	

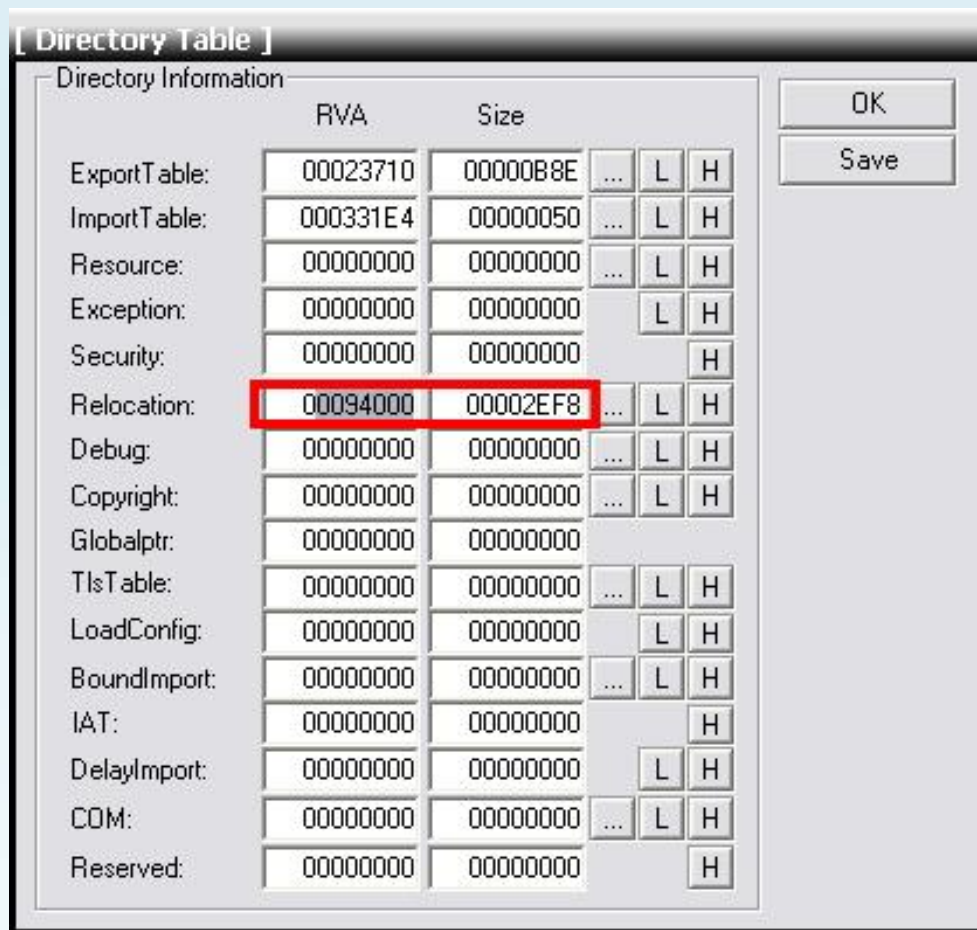
00870163	10570200	00	00002370	Export Table address = 2370
0087016C	8E0B0800	00	0000088E	Export Table size = 8E (2958.)
00870170	E4310300	00	000331E4	Import Table address = 331E4
00870174	50000000	00	00000050	Import Table size = 50 (80.)
00870178	00000000	00	00000000	Resource Table address = 0
0087017C	00000000	00	00000000	Resource Table size = 0
00870180	00000000	00	00000000	Exception Table address = 0
00870184	00000000	00	00000000	Exception Table size = 0
00870188	00000000	00	00000000	Certificate File pointer = 0
0087018C	00000000	00	00000000	Certificate Table size = 0
00870190	00400900	00	00094000	Relocation Table address = 94000
00870194	F82E0000	00	00002EF8	Relocation Table size = 2EF8 (12024.)
00870198	00000000	00	00000000	Debug Data address = 0
0087019C	00000000	00	00000000	Debug Data size = 0
008701A0	00000000	00	00000000	Architecture Data address = 0
008701A4	00000000	00	00000000	Architecture Data size = 0
008701A8	00000000	00	00000000	Global Ptr address = 0
008701AC	00000000	00	00000000	Misc_bv 0.

We found the address of Relocation is $94000 + 870000 = 904000$, which is where the load **section. Reloc1:**

00570000	00120000				Map	R	E	R
00870000	00001000	UKHook40		PE header	Image	R		RWE
00871000	0001E000	UKHook40	.text		Image	R		RWE
0088F000	00006000	UKHook40	.rdata	exports	Image	R		RWE
00895000	0000C000	UKHook40	.data	data	Image	R		RWE
008A1000	00003000	UKHook40	.reloc	imports	Image	R		RWE
008A4000	00040000	UKHook40	.text1	code	Image	R		RWE
008E4000	00010000	UKHook40	.adatas	code	Image	R		RWE
008F4000	00010000	UKHook40	.data1		Image	R		RWE
00904000	00010000	UKHook40	.reloc1	relocations	Image	R		RWE
00914000	00080000	UKHook40	.pdata		Image	R		RWE
00920000	00004000				Image	R		RWE

Now we must fix addresses turn right on Relocation of the original file. Double Click section. Reloc:

[illegible]



You fix the $RVA = 8A1000 - ImageBase: 870000 = 31000$. Size = 1D28. Save the file.

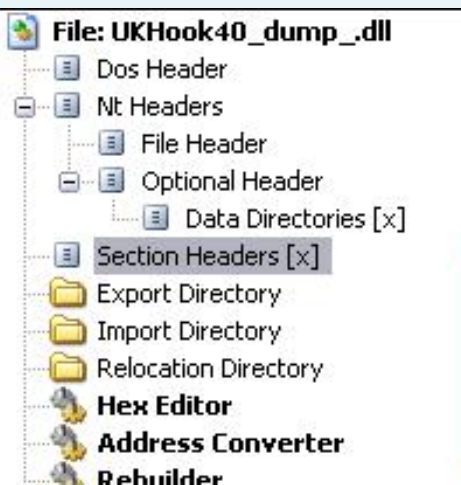
Now see the files have to rename UKHook40.dll not? If it is running Unikey.exe. It lickerish load.

Note: Some of the old Unikey as beta 4, if you do not use that fix Relocation UPX (or some other packer) pack dll file for reduced size. Unikey.exe run, it can load dll suon will. There is no misunderstandings can check the file size. Because UPX accidentally find and correct root Relocation encrypt it. So where do run error. **In most cases, we must always fix the Relocation dll unpack any time, whether any packer.**

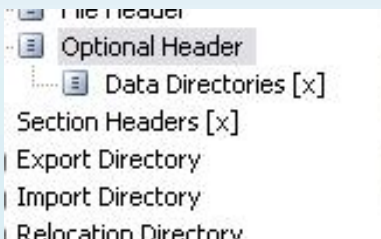
Very fortunate that Armadillo ko hê Relocation should encrypt it easy fix. If the packer will encounter other problems to day. Now to our final stage.

VII Reduce-Size:

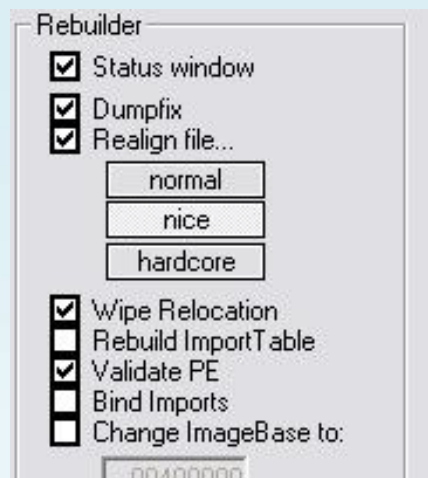
In fact also do what many do, open up CFF Explorer, open the file, delete all section:

File: UKHook40_dump_dll 	Byte[8]	Dword	Dword	Dword	Dword
	.text	0001DAEC	00001000	0001DAEC	00001000
	.rdata	0000529E	0001F000	0000529E	0001F000
	.data	0000B74C	00025000	0000B74C	00025000
	.reloc	0000225E	00031000	0000225E	00031000
	text1	00040000	00034000	00040000	00034000
	adata	00010000	00074000	00010000	00074000
	data1	00010000	00084000	00010000	00084000
	reloc1	00010000	00094000	00010000	00094000
	pdata	000B0000	000A4000	000B0000	000A4000

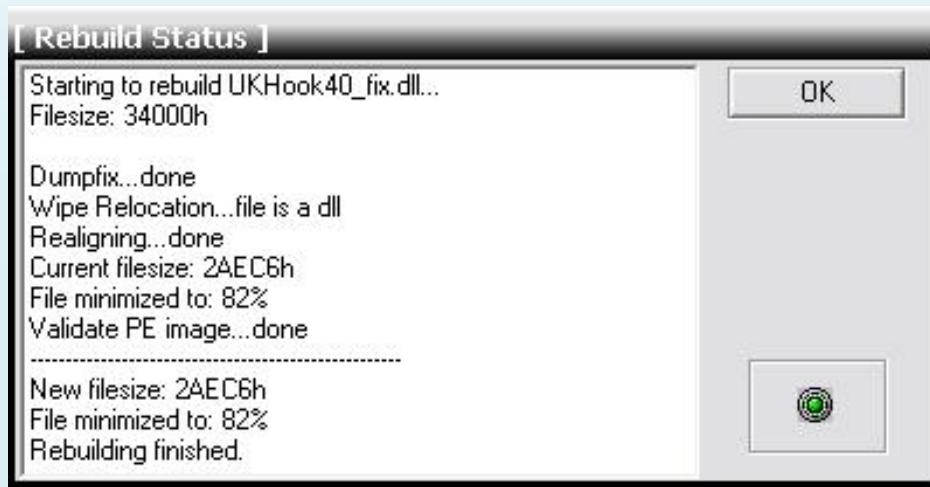
Because. ADATA do not redirect the IAT should delete that we do fear anything. Then edit Of Base Code for the right:

	SizeOfInitializedData	00000110	Dword	000B2000
	SizeOfUninitializedData	00000114	Dword	00000000
	AddressOfEntryPoint	00000118	Dword	000107E9
	BaseOfCode	0000011C	Dword	00001000
	BaseOfData	00000120	Dword	0001F000

Save the file. To open more streamlined LordPE, the Option to rebuild:



Conduct rebuild the file after the fix:



Size after the collapse, but smaller than the original file is not pack:

UKHook40_dump.dll	1,360 KB
UKHook40_dump_.dll	1,360 KB
UKHook40_fix.dll	172 KB
UKHook40_not_Pack.dll	184 KB
UKHook40_packed.dll	1,000 KB
UniKeyNT.exe	212 KB

Rename, file and still be good Unikey.exe Load. Considered as completed for the unpack dll pack by Armadillo.

VIII-Ending:

Indeed unpack ko hề simply copy, Lam Hung huc 1 as the host, we need to master the basic knowledge. Hope you have a lot of things from this article.

Big Thanks to: fly [CUG], RES stephenteH () for helping.

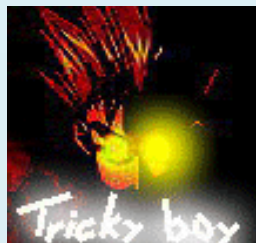
Sincere thanks:

The brother-in REA helped tricky in the study.

Typically hacnho series with tut unpack their super levels.

Uncle Why Not Bar for some tricky help pack the file Unikey

... And you all again.



Tần Ma Kiem

(Written by Trickyboy)

Armadillo Exact Version Location Tutorial

Two methods of manually Finding the exact version of an ARMADiLLO protected file

Target: **TARGET.EXE**

(http://www.absolutelock.de/construction/files/infobase/New/arma_version_location/Target.zip)

Protection ..: **ARMADiLLO v3.?**

Difficulty **Intermediate**:

Tools Needed:

1.) **Olly Debug**

Welcome! :)

This tut was written by: **MEPHiST0** (From Gods Unpacking site)

Translated by: **kienmanowar**

Posts will be next to you just how to know the exact version of the Armadillo packer used to protect files. So first you download target used in this article was the first. Now Let's do it:) Enjoy!

Part 1:

Definition To Armadillo's version information

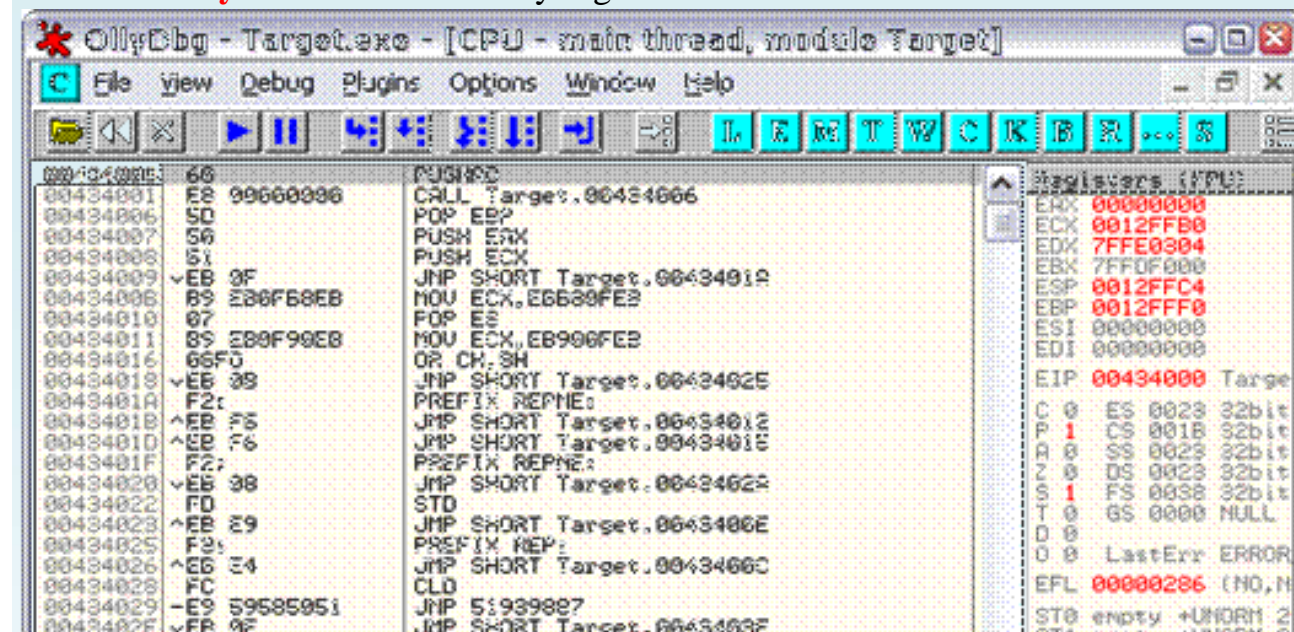
There are many of us (especially children) do not know how to find the exact version of the Armadillo. Some people back then that this can not be performed. This article will explain to you how to find the exact version of the Armadillo (the exact number to each child and not a word is still "x" odious). The method used in This was written to apply to **have Armadillo version 3.xx**, even with **Debug Blocker**, or **CopyMem2**. And also one more to say here is, at least you should know is how to unpack the Armadillo do before you read this article.

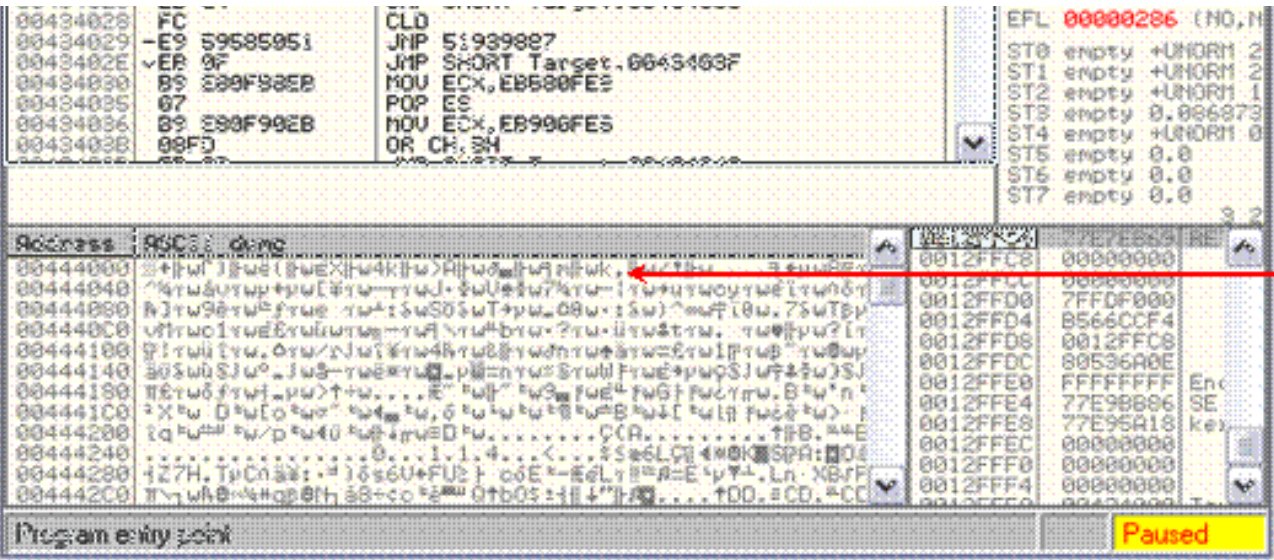
As you see in this article the author has attached a file Target.exe's authors, and this file is Protected by the Armadillo?..? :). After all read this article you must find out which version of the Armadillo's authors used to packed file. Exact number of versions of this (and sometimes build) are encrypted in the Armadillo.

Part 2:

METHOD 1 - Finding Exact Armadillo's version

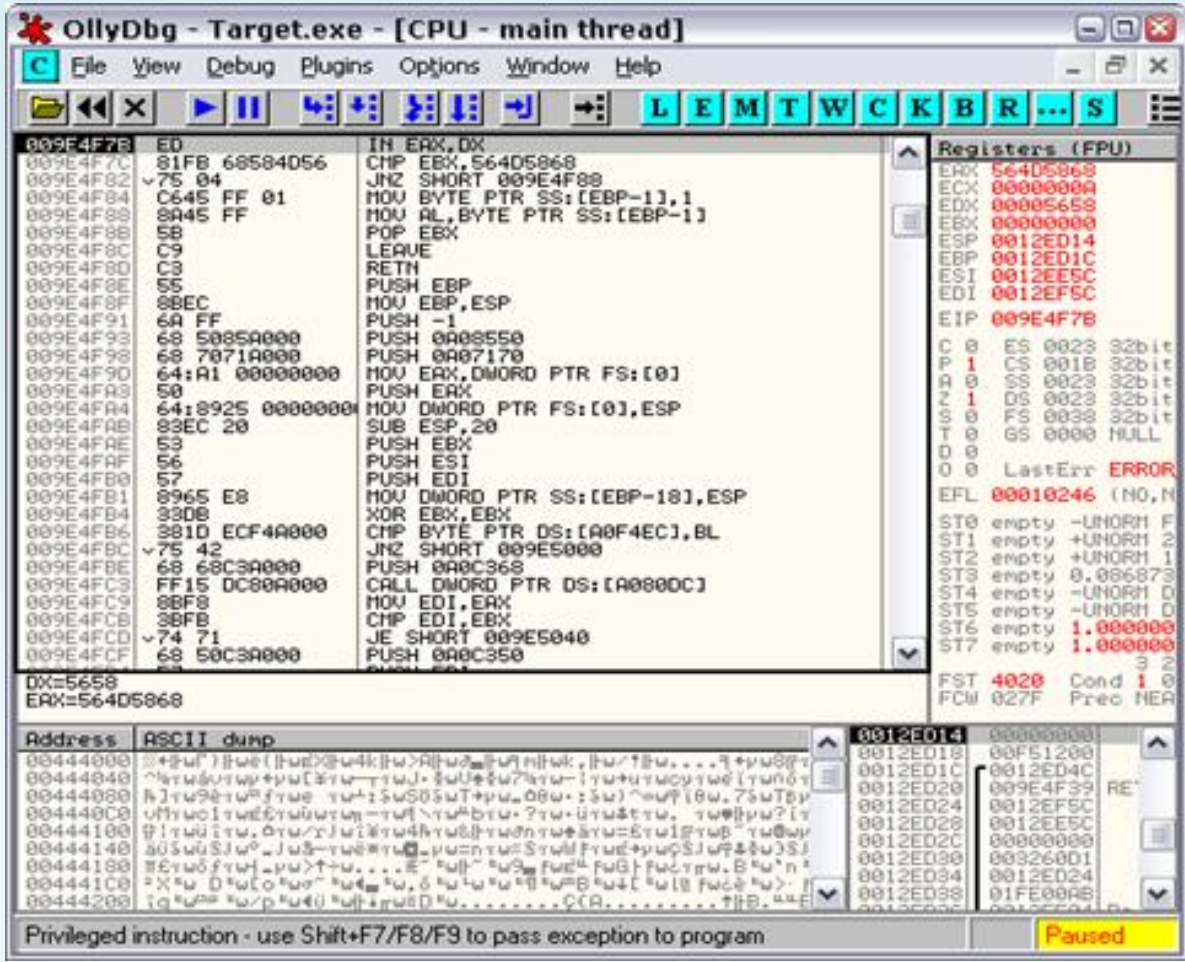
Oki we started nhé. First you load the file in Olly target.exe. After the load is finished we will be in **Point of Entry** Armadillo. Similarly Figure below:





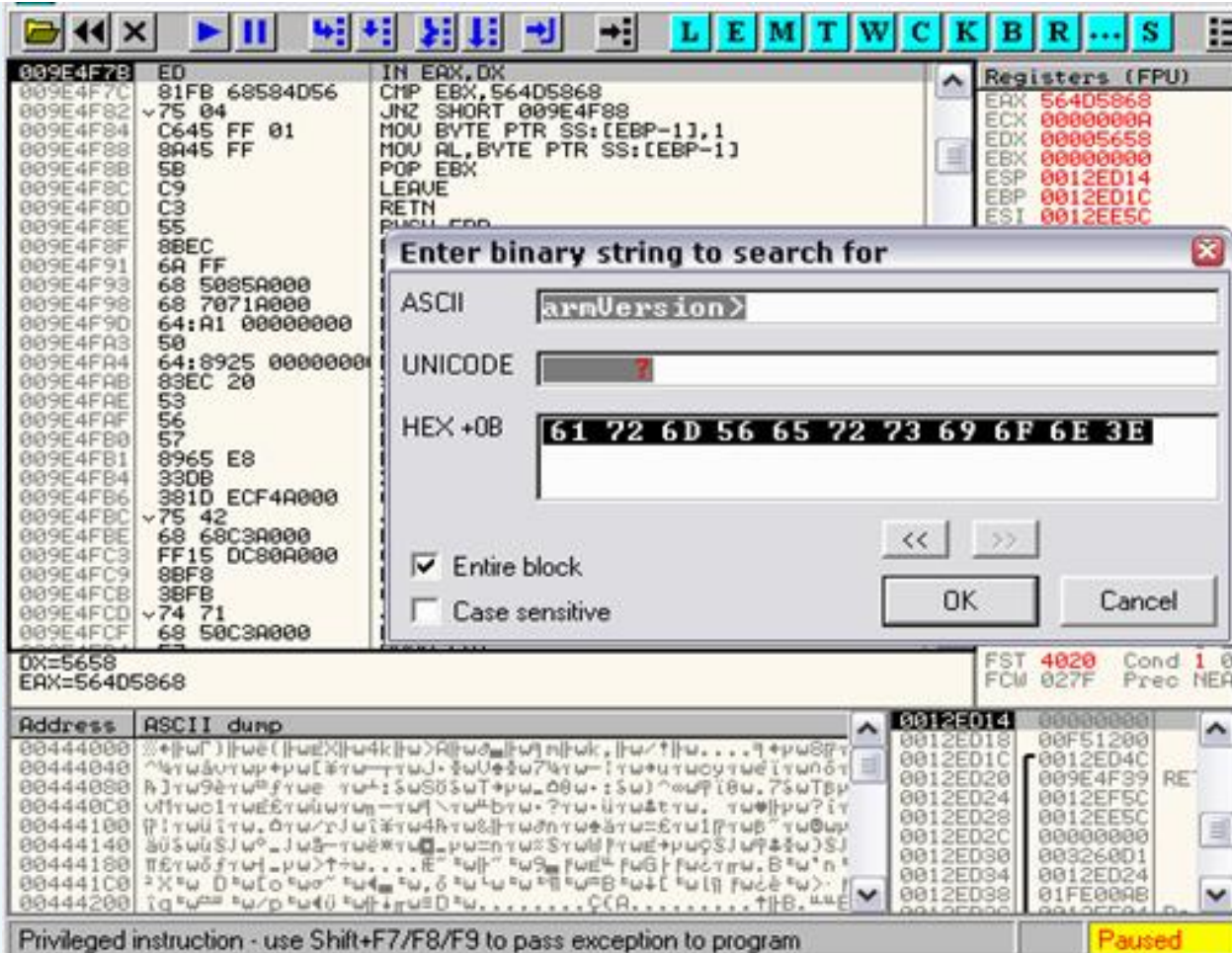
Right Click in
your Dump
and Select
HEX or TEXT

Now after you have selected one of two options is **HEX** or **TEXT** window dump
 Next you press **Shift + F9 3 times** (in this case is 3 times, but she's the only hit 1 time only depending on
 each machine) and you will receive a 'privileged instruction. Now, the code of the file is Decrypted target.
 ext and we can find the exact version of the Armadillo:). Similarly Figure:

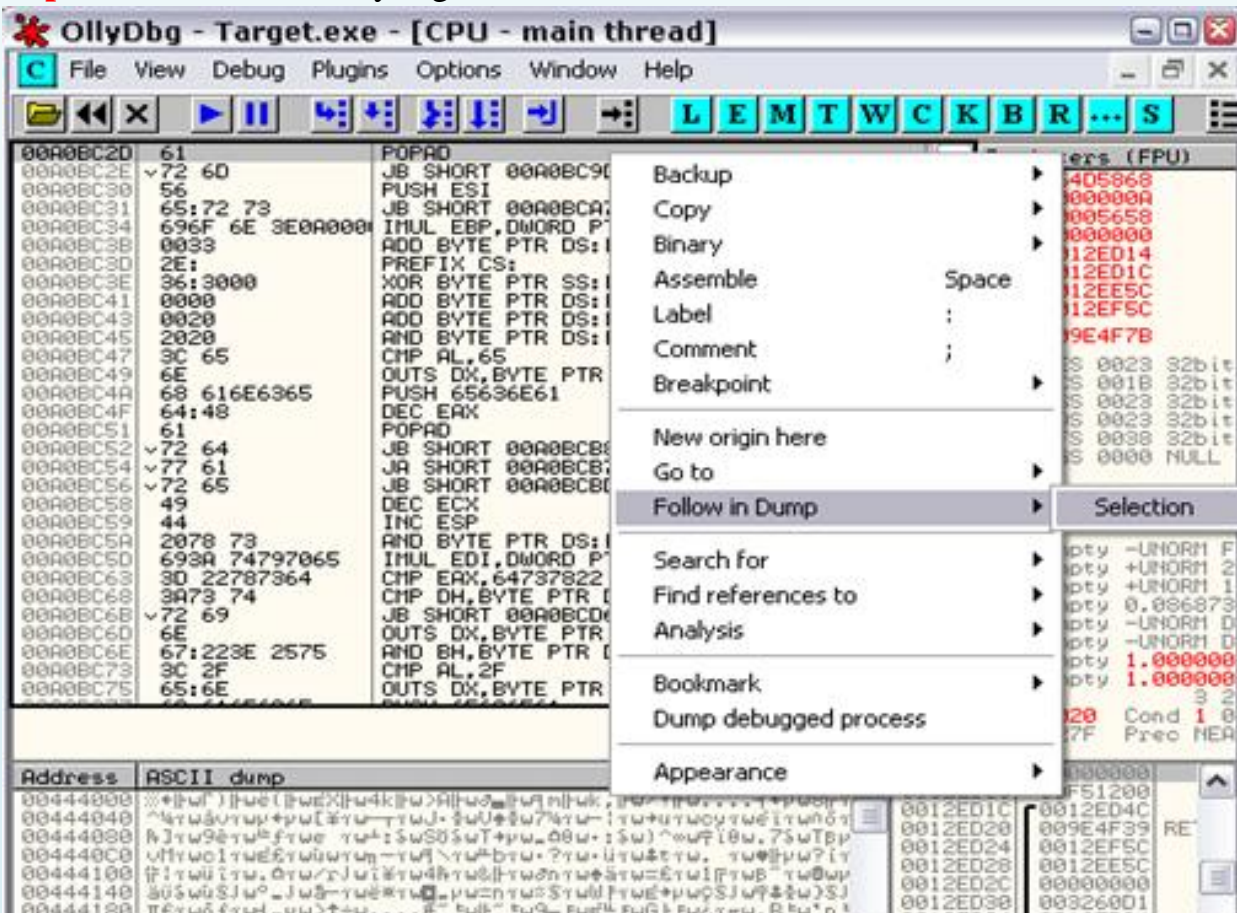


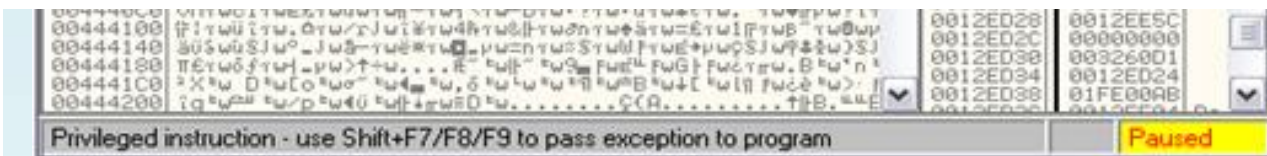
Oki, now at the main screen of you, press **Ctrl + B** to open a dialog box allows us to **search Binary strings** (Search binary sequence). In the ASCII text box you type the following: **armVersion>** similar image illustrated below:



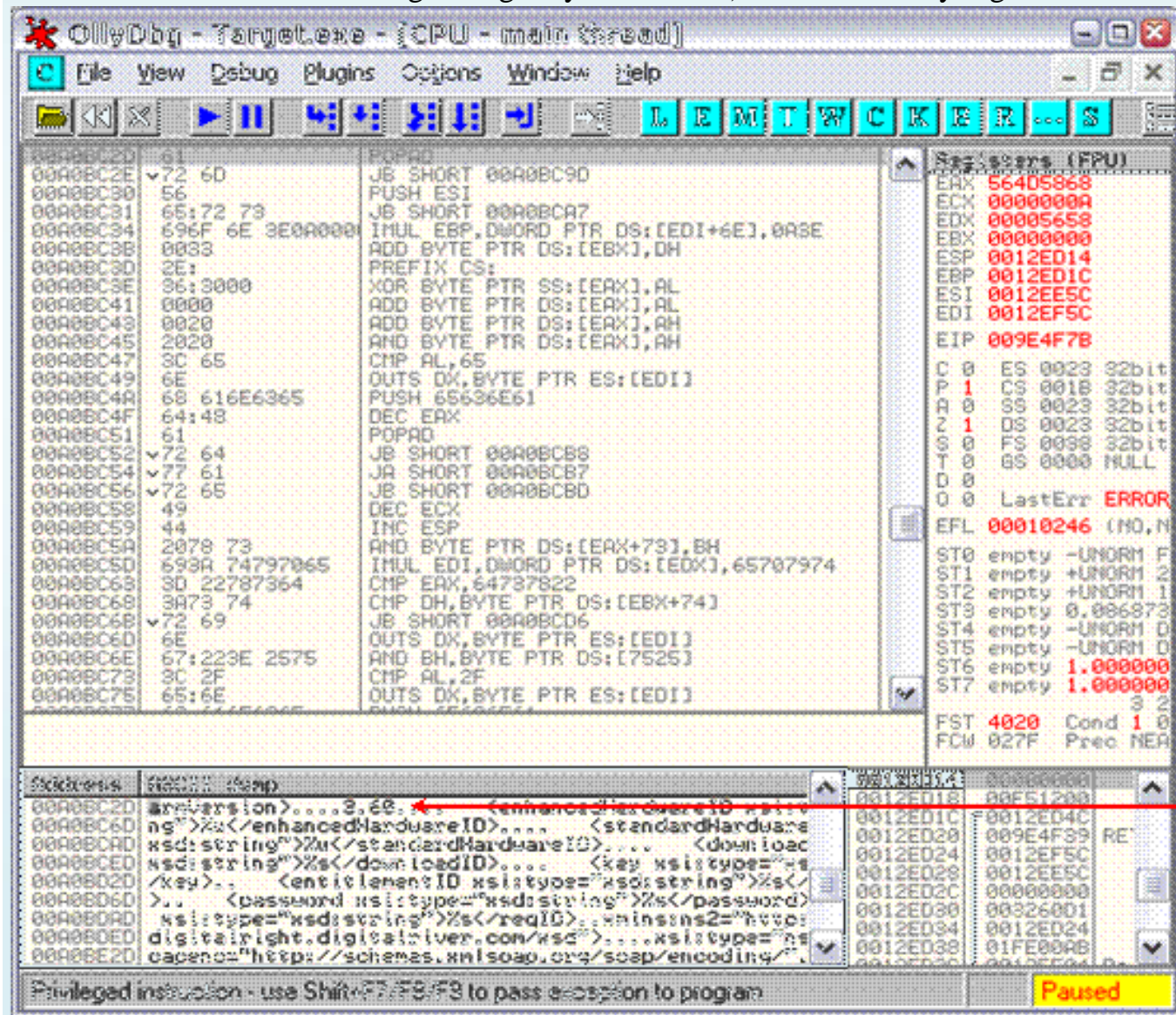


You must enter exactly as on it, after you enter finished click OK button. Hiiii it will take us to the Armadillo's Version. Next to go: the HIGHLIGHT FIRST BYTE> right click on it and choose **Follow in dump> Selection**. Similarly Figure below:





Kekeke It is then the other, now you look down the window dump Window. We will see exactly Version of Armadillo lu lu it is growing very obnoxious:) Yeah! Similarly Figure below:



As you see above, we have the exact version of the Armadillo not appear characters an "x" is more obnoxious. Exe file here was protected by **Armadillo version 3.60**.

Part 3:

METHOD 2 - Finding Exact Armadillo's version

An easy way for you can find the exact version of the Armadillo:

When you find the **OEP**, and you have any **DUMPED Armadillo Protected File v3.xx** We will open the file has been dumped in a HEX editor program (for example Hiew) and search **armVersion** by searching by Ascii. This method will let us know the exact version of the Armadillo dumped in your file: D. But there are also the methods are not very effective. But way or another as long as our purpose is to be ... ENJOY: D

(~~)- Thanx All-oOo-oOo-(~~)



Armadillo v3.xx Manual Unpacking Tutorial for Windows XP

Manual Unpacking Armadillo Standard Protection + DEBUG blocker with OllyDbg then Patching Armadillo, so we can rebuild with imports ImpRec.

Target: FlashFavourite v1.31

Website: [Http://www.pipisoft.com/](http://www.pipisoft.com/)

Protection: Armadillo v3.60 + Debug Blocker.

Difficulty: Intermediate / Difficult (know a little about debugging)

Tool Needed:

1. Olly Dbg v1.08 or better
2. LordPE Deluxe

Welcome! :)

This tut was written by: **MEPHiST0 (From Gods Unpacking site)**

Translated by: **kienmanowar**

This article will explain how a thorough through illustrated with detailed images, how to unpack armadillo v3 with debug blocker feature.

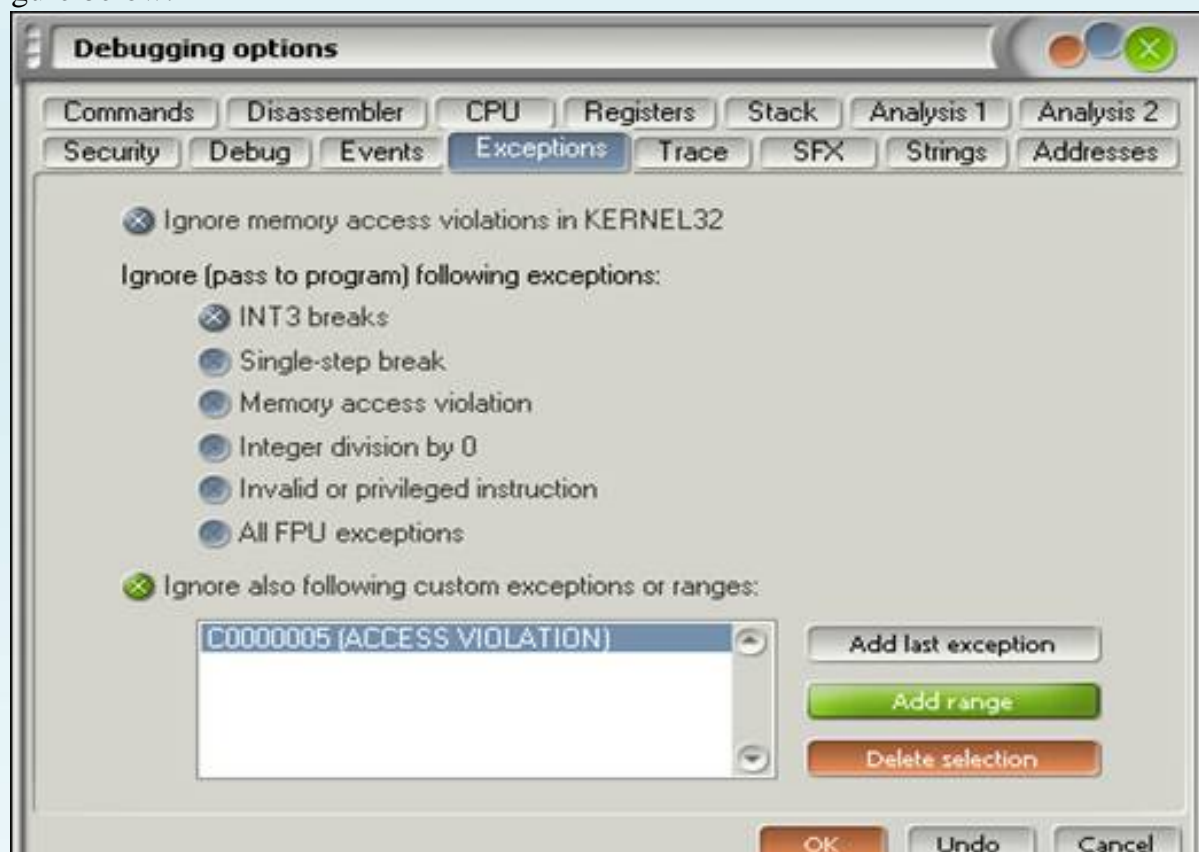
Part I:

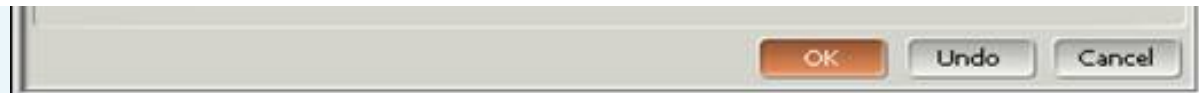
Defeating Armadillo's Debug Blocker and reaching the Original Entry Point, and dumping.

-**** First ... Armadillo with Debug Feature Blocker is the location of two of the processing Father and Child. The word processor is a Father Loader - Process Child process is a normal file is protected by Armadillo.

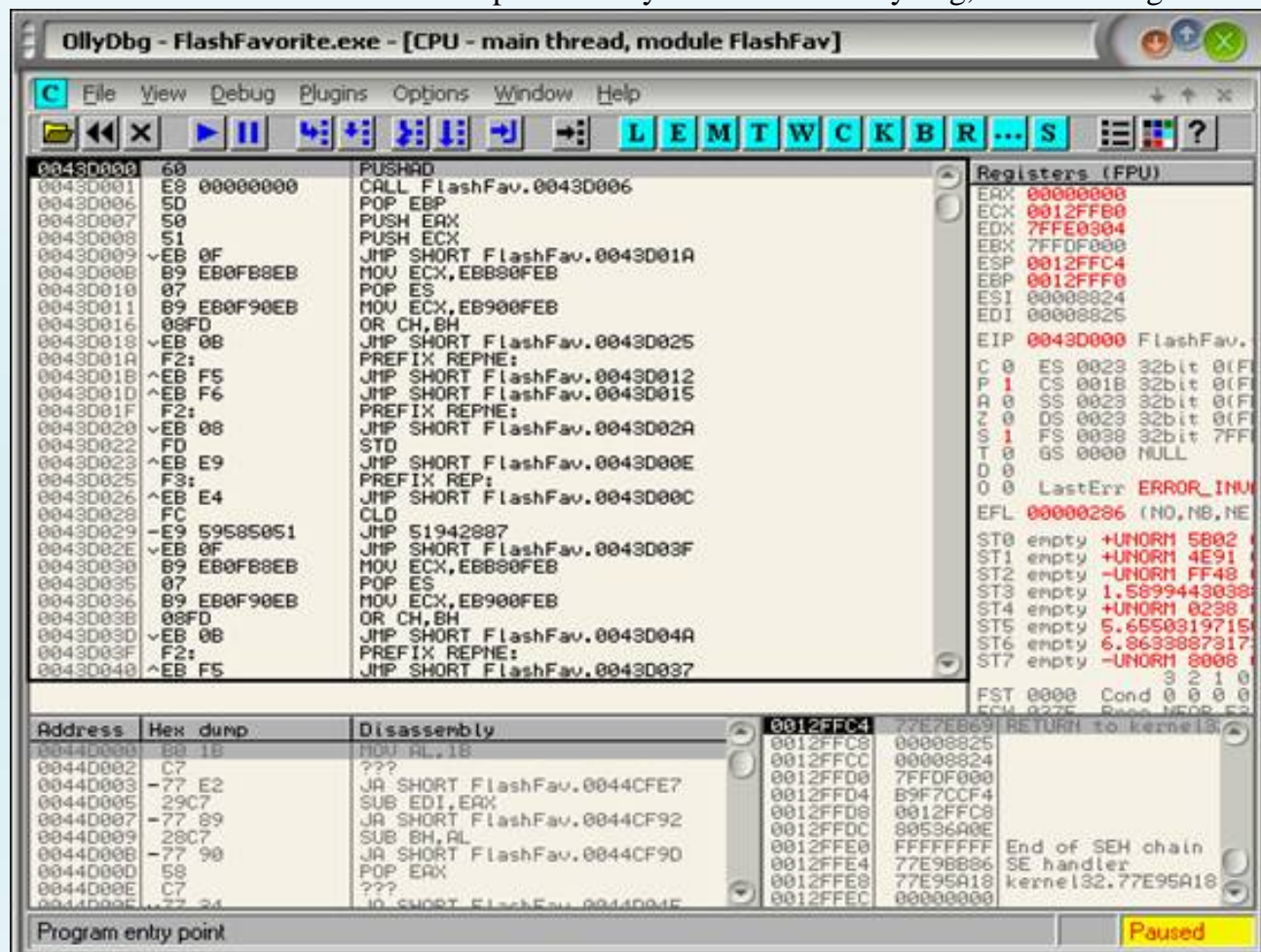
-**** The address that we have been through Olly Debug is not the same. The Armadillo is newer v3.70 check Ollydbg.exe file - so you may want to change the name of Olly for this case. Method used to Unpacking Armadillo will be effective for all **Armadillo 3.xx version**, except **CopyMEM2 Custom** and other **features** used in the Armadillo.

-**** Next Armadillo many Access Violations (region are not allowed access), so you must configure in by Olly add more: **C0000005 (ACCESS VIOLATION)** to the exception list (exclusion list). To achieve this you run Olly up, then select **Options ---> Options debugging**, and select the **Exceptions tab** to configure Olly as in Figure below:



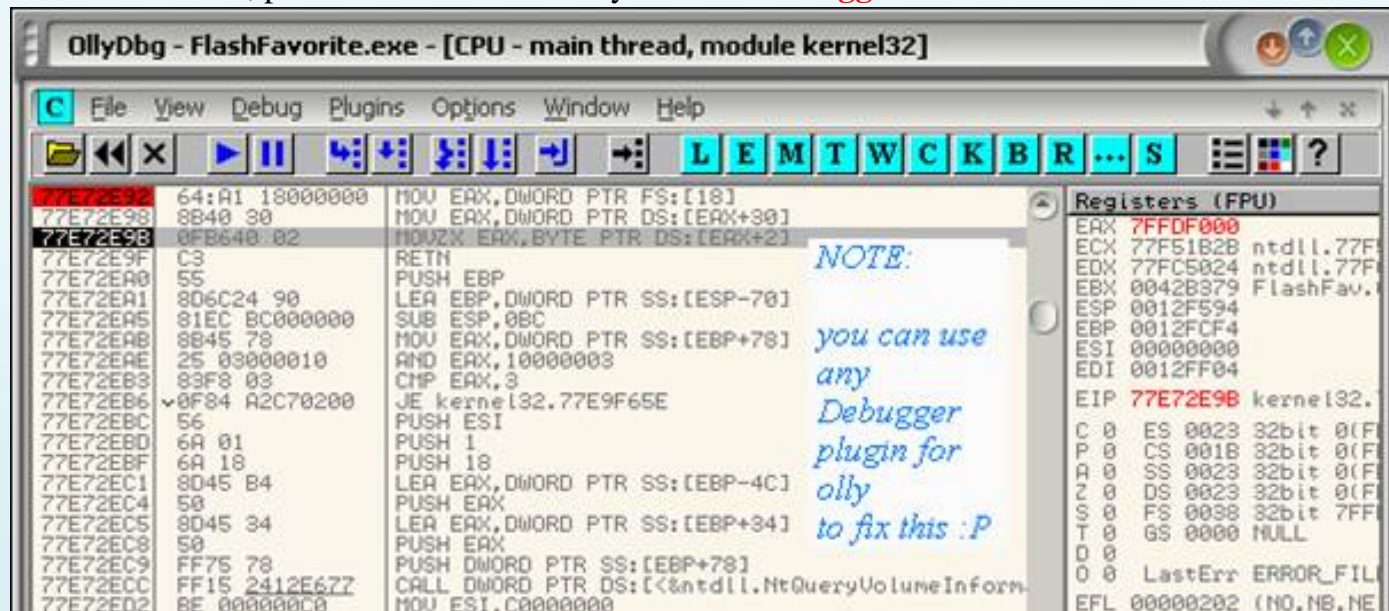


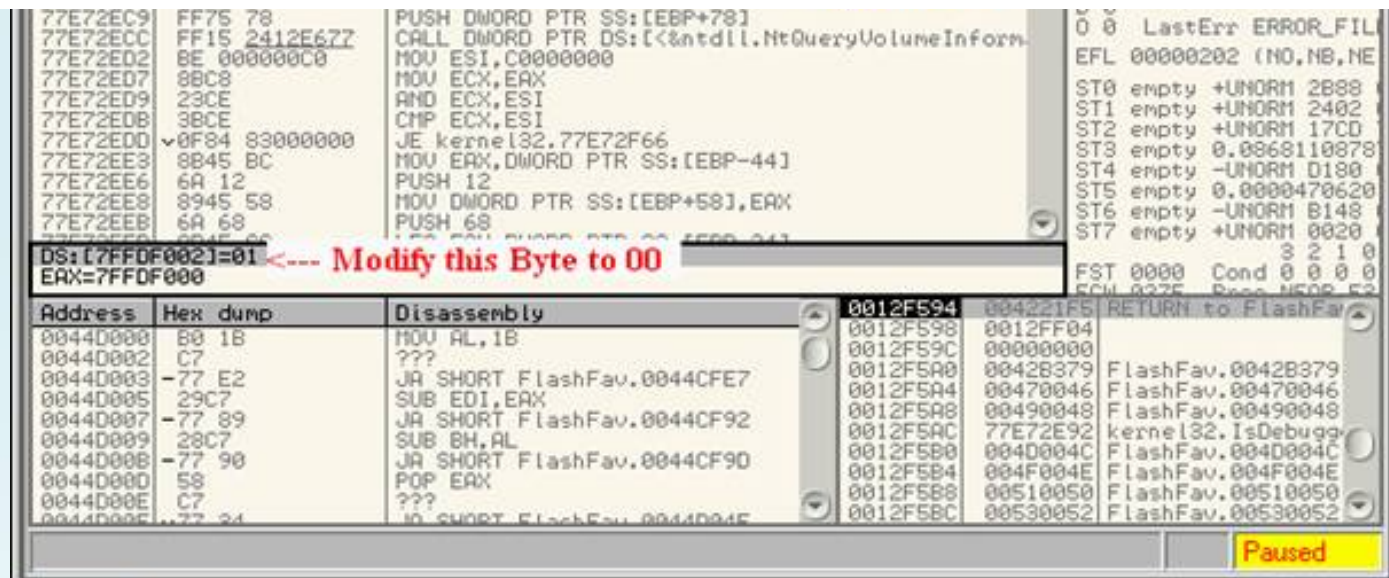
-**** Oki, now we will proceed with work Cracking target above. After such a configuration is done, we will load the file **FlashFavourite.exe** was protected by Armadillo in Olly dbg, similar to Figure below:



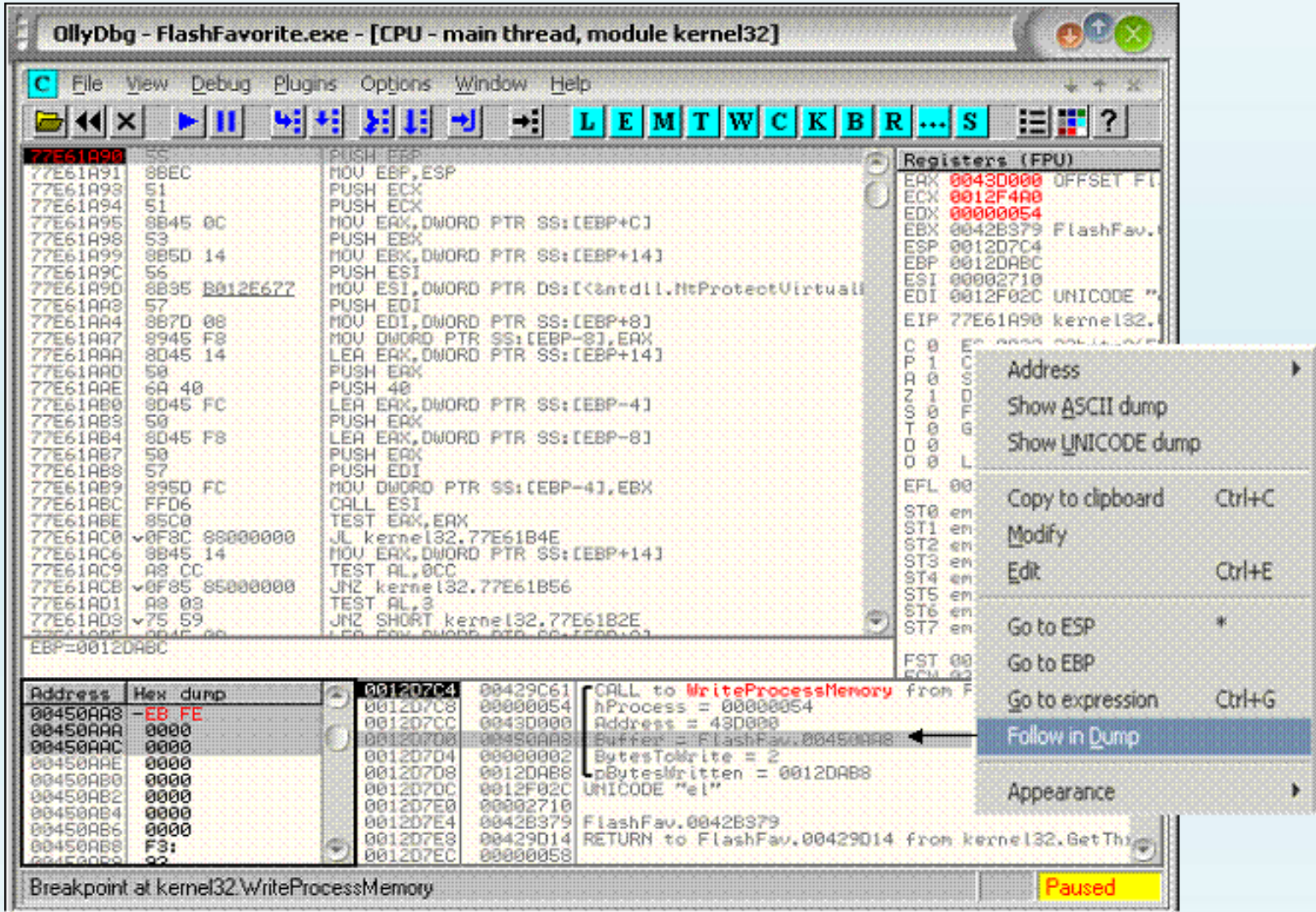
After -**** load complete file Olly will stop at **0043D000** such, this is the **Entry Point** of Entry Point Armadillo ... this actually the same as the Point of Entry when a file is Protected by ASPACK .. and some look like armadillo programs compiled C + + so ... But actually, this is the armadillo. Certainly Armadillo here will check the troubleshooting current (Present Debugger). Therefore, we then set in BP **IsDebuggerPresent** function (or can use the Plugin and skip writing this ..):

-**** After Set BP, press **Shift + F9** 3 times you will **IsDebuggerPresent** at Break:





-**** Oki, now after we fix the test debugger, next we will conduct work to ignore (in other words is to avoid) Armadillo Debug Blocker feature. :) What we should do is make the Child Process Entry Point and endless loop (hex: EB FE). We search an endless loop so we can stop at Point of Entry Process Child:) and we can do this by using the API function is **WriteProcessMemory**.
 -**** We set a break point in **the API** function **WriteProcessMemory** and run by pressing **Shift + F9**, you will receive an instruction (privileged instruction), just press Shift + F9 to ignore it. We'll Break in **WriteProcessMemory**. First, we break, it is no good, Break the next time we will have what we are looking for. Similarly Figure below:



-**** Break time this second, we will stop at the API function **WriteProcessMemory**. Please pay attention to window Stack Window:

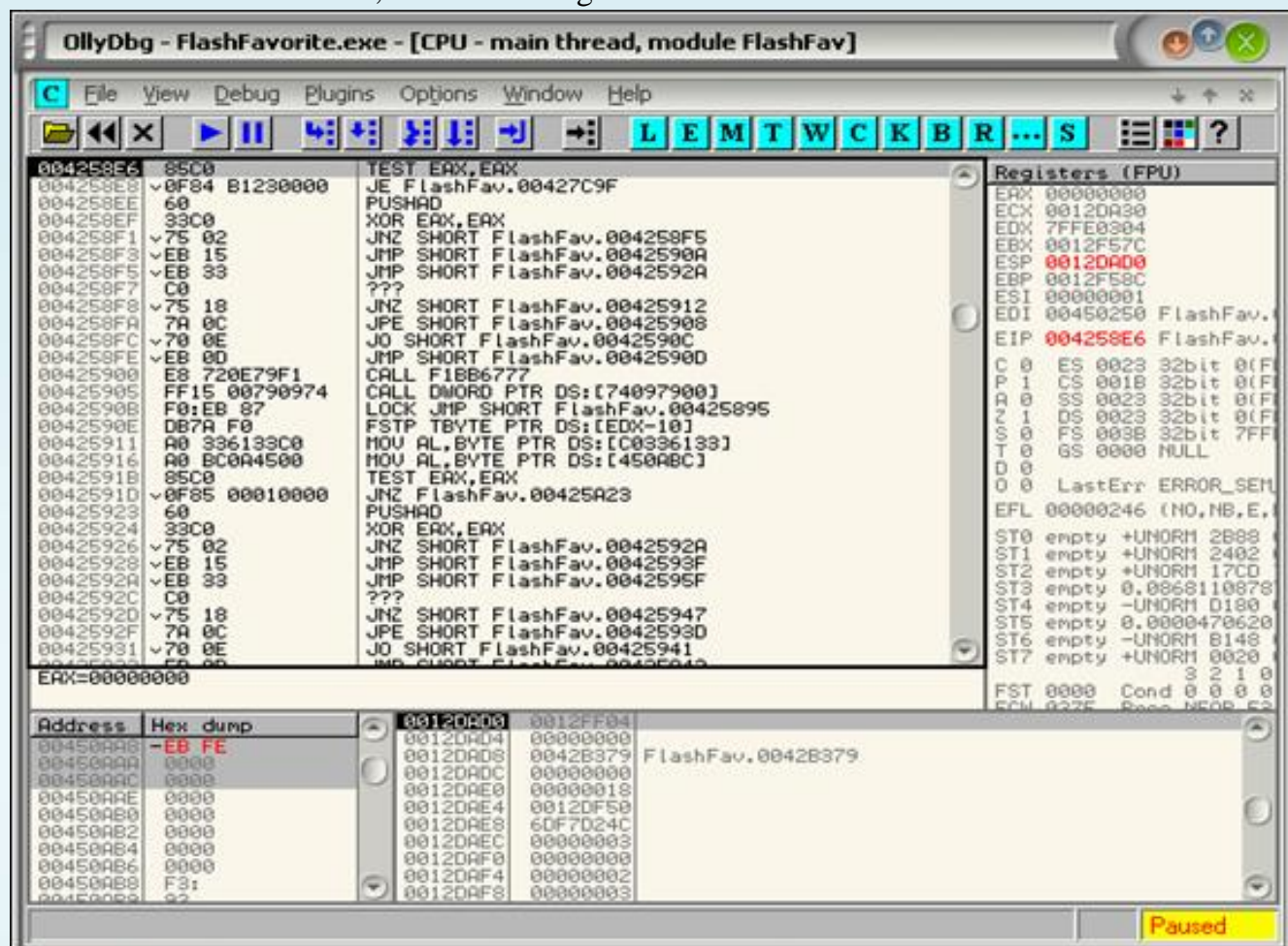
Check out the BUFFER> writing the buffer is 2 bytes

Check out the Address> address is where writing is to buffer

-**** Right Click Buffer similar image illustrated above, then select **Follow in dump**: 2 bytes are written, in my case it is 2 bytes: **60 E8** so keep in mind we get (we will have to write the 2 bytes to return to Child Process before when we want to Attach).

Oki -**** now we will edit 2 bytes **60 E8** window to dump Windows **EB FE (Jump EIP)**. We make Buffer noted the order in dance forever Entry Point, so we can correct at Break Point of Entry Process Child:), like Figure above.

-**** Now, we have Point of Entry Process Child in a loop (loop) ... we will Attach Child Armadillo Process ... But do not allow us to Attach! Therefore we must stop ... Patch. Olly run by pressing the key combination **Shift + F9**, and while running, put in a BreakPoint **WaitForDebugEvent** function. (We will always break at the function **WaitForDebugEvent** because processing is located in a loop). When we Break in **WaitForDebugEvent** function, you press **Ctrl + F9** (Trace until RETN), a new line (new thread) will be created ...: P (it will say on the bottom of olly window.). Now we will use in order RETN from **WaitForDebugEvent** function, press F7 to trace into the RETN. **Make sure to write EAX value = 0** if you RETN from **WaitForDebugEvent** function, you'll come to function TEST EAX, EAX like image illustrated below:



-**** Here we will assemble:

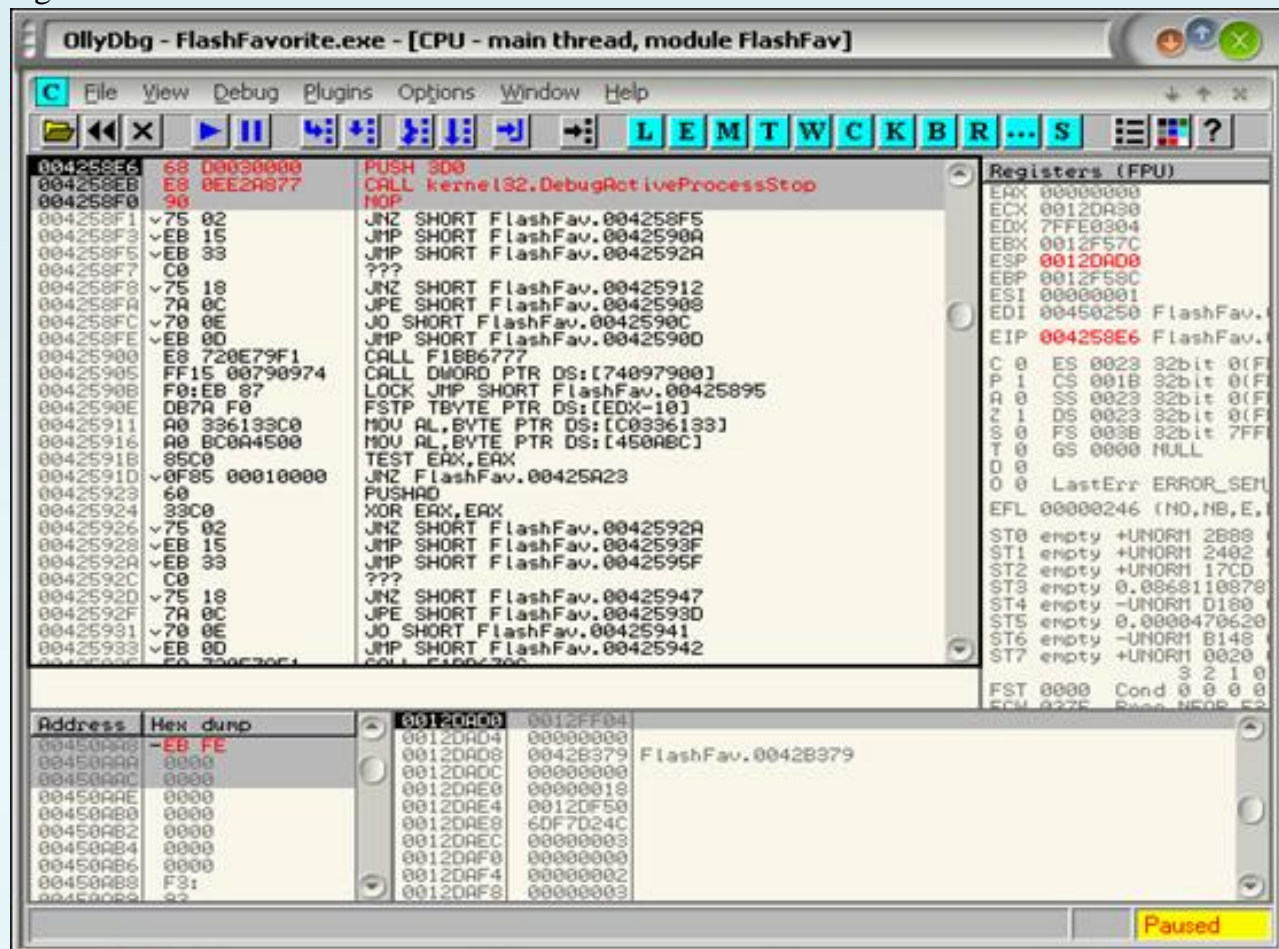
PUSH PID (PID = Process ID)

CALL DebugActiveProcessStop <thx winxp!>

You may have been accurate in **PID** Olly Dbg, by clicking File> Attach>

There we will have 2 files are running FlashFavourite - select get correct process that is not highlighted RED. This PID is that we need .. :)

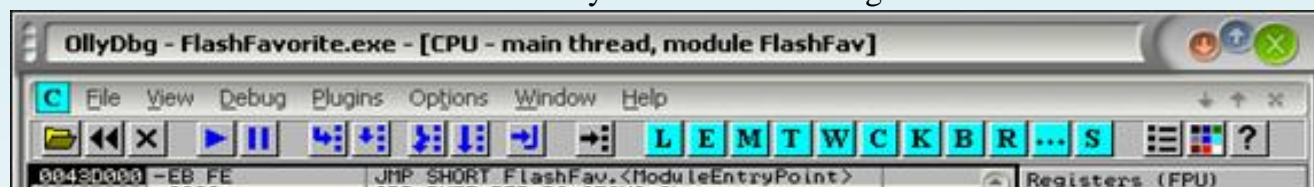
-**** In my case it's correct that I have PID is **03D0**, so assemble order to test the following **PUSH 03D0**. Similar images illustrated below:

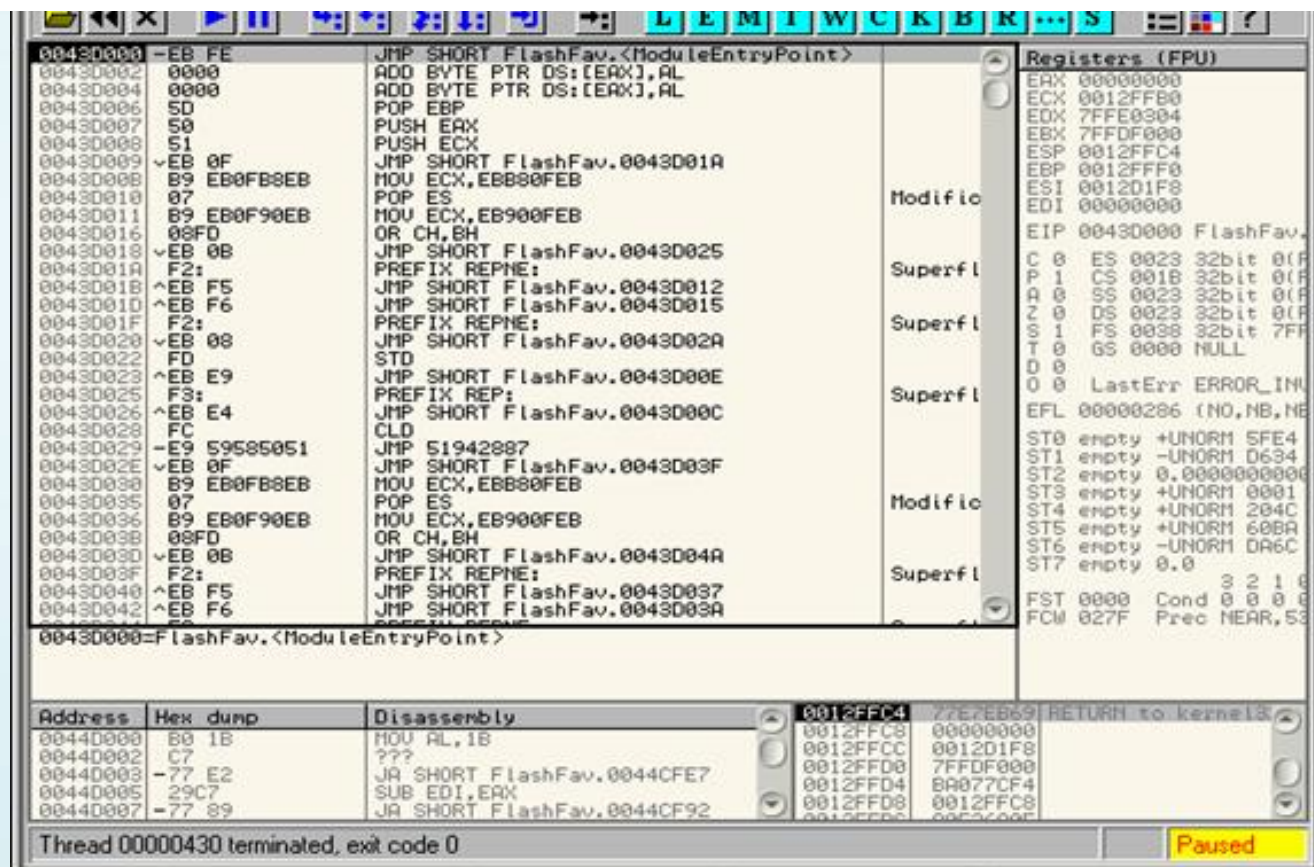


Oki -**** Hmm, now after you edit the code like Figure above, followed by pressing **F8** for me to order **NOP** - and open a another Olly debug while others keep the window first Best (same as we open a screen text editor, while others have a screen is more work). Note the following when you can open your PC will slow the process | Now at the screen Olly we just open it, you choose **File> Attach>** Process and you have to use **PUSH (PID)**. When you Attach the screen Olly's finished, you will here in Olly (may be different in your computer):

```
77F7F571 C3 RETN
77F7F572 8BFF MOV EDI, EDI
77F7F574 CC INT3
77F7F575 C3 RETN
77F7F576 8BFF MOV EDI, EDI
77F7F578 8B4424 04 MOV EAX, DWORD PTR SS: [ESP +4]
77F7F57C CC INT3
77F7F57D C2 0400 RETN 4
77F7F580 64: A1 18000000 MOV EAX, DWORD PTR FS: [18]
```

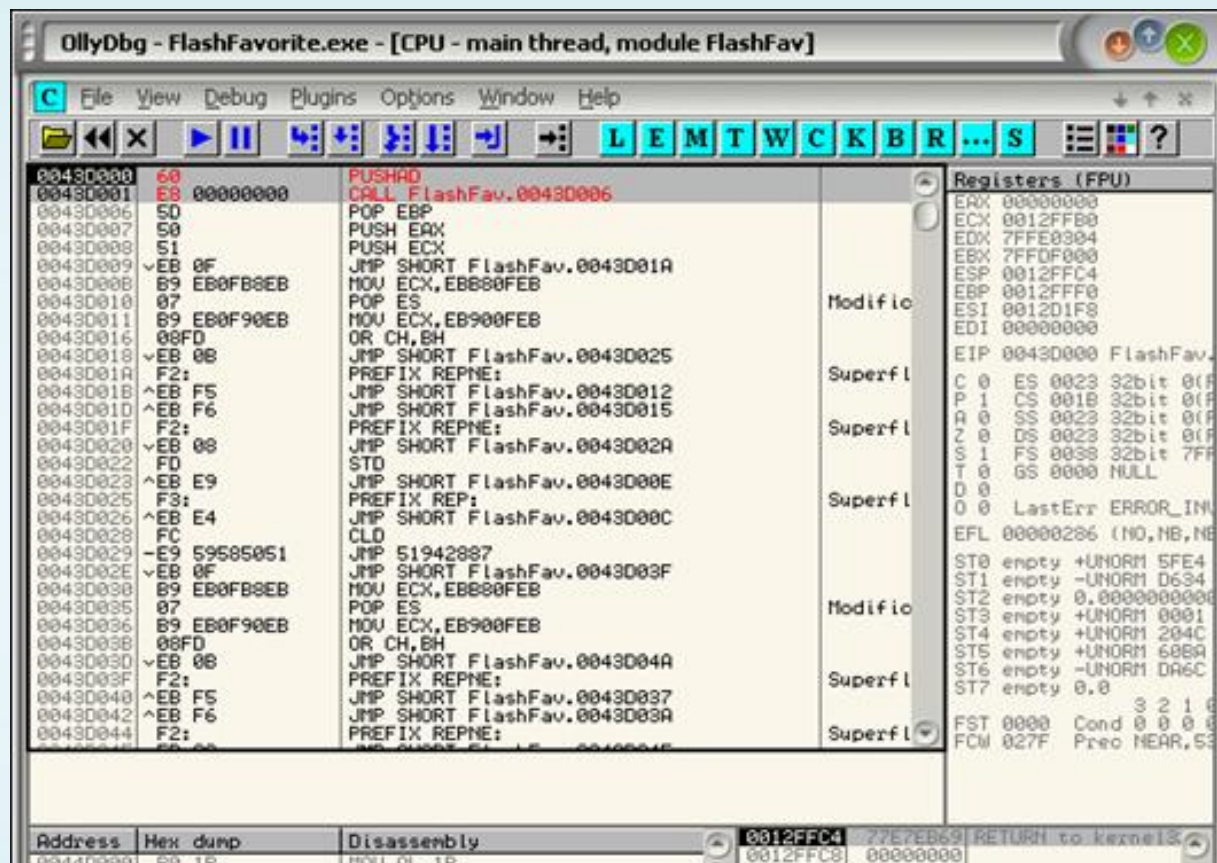
-**** Now you press **F9** to run and then press **F12** to pause, we'll stay in place of endless loop (Infinite Loop) that we set before! : D AKA: Child Process Entry Point. Similar images illustrated below:





-**** If Olly's screen is the same as illustrated by pictures of me on the Congratulations! You have excluded the Armadillo's Debug Feature Blocker. Now we will not do the Child in the process that has created Armadillo. Now it's just like a protection Armadillo normal.

Oki -**** stay in laos a cigarette that has (this author does not smoke them) and as the rest of this article we do with Standard Protection:). All that we we must do is assemble **EB FE** (JMP EIP) return **Original Bytes**. You also remember what it is not? In my case it is **60 E8** .. So we change the **EB FE 60 E8** similar image illustrated below:





-**** Oki, after we change back to EIP JMP Original Bytes .. we will have IS Debugger Check in at certain Child Process. So we will set a BP in IsDebuggerPresent function (or use a plug in). Run by pressing **Shift + F9** until we Break in function, the function to fix that it once more for Child Process.

-**** Next we re-set at a BP function **CreatThread (some armadillo ... you have to break in function SetProcessWorkingSetSize)**. We now run again by pressing **Shift + F9**, occasionally (in this case There is) we will see a Nag Screen and is **Armadillo's Time Trial Nag Screen**), this is a good sign:) So when we dismantle it then the exe file will be cracked:) Pretty Much) (^ _ ^)

-**** Time when the Nag up, they click OK and we'll break in **CreatThread**. When you stop at **CreatThread**, press **Ctrl + F9** once, and you'll come to me lenh **RETN 18**, press **F7** to trace into the RETN

-**** We again return to the Armadillo's Code, and we are very close to the jaw Call ORIGINAL ENTRY POINT! We will stop at a code fiddling follows:

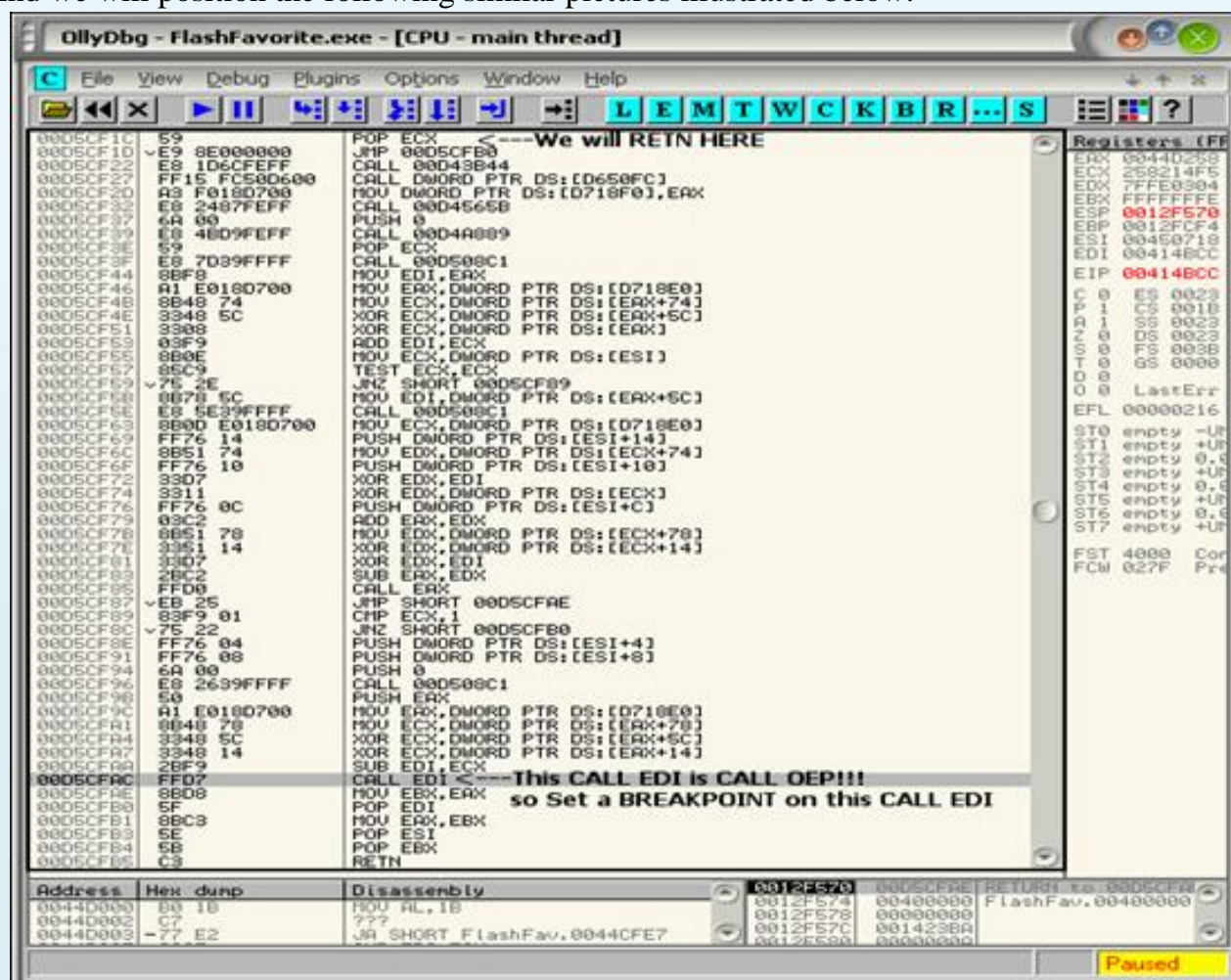
5e POP ESI

C9 LEAVE

C3 RETN

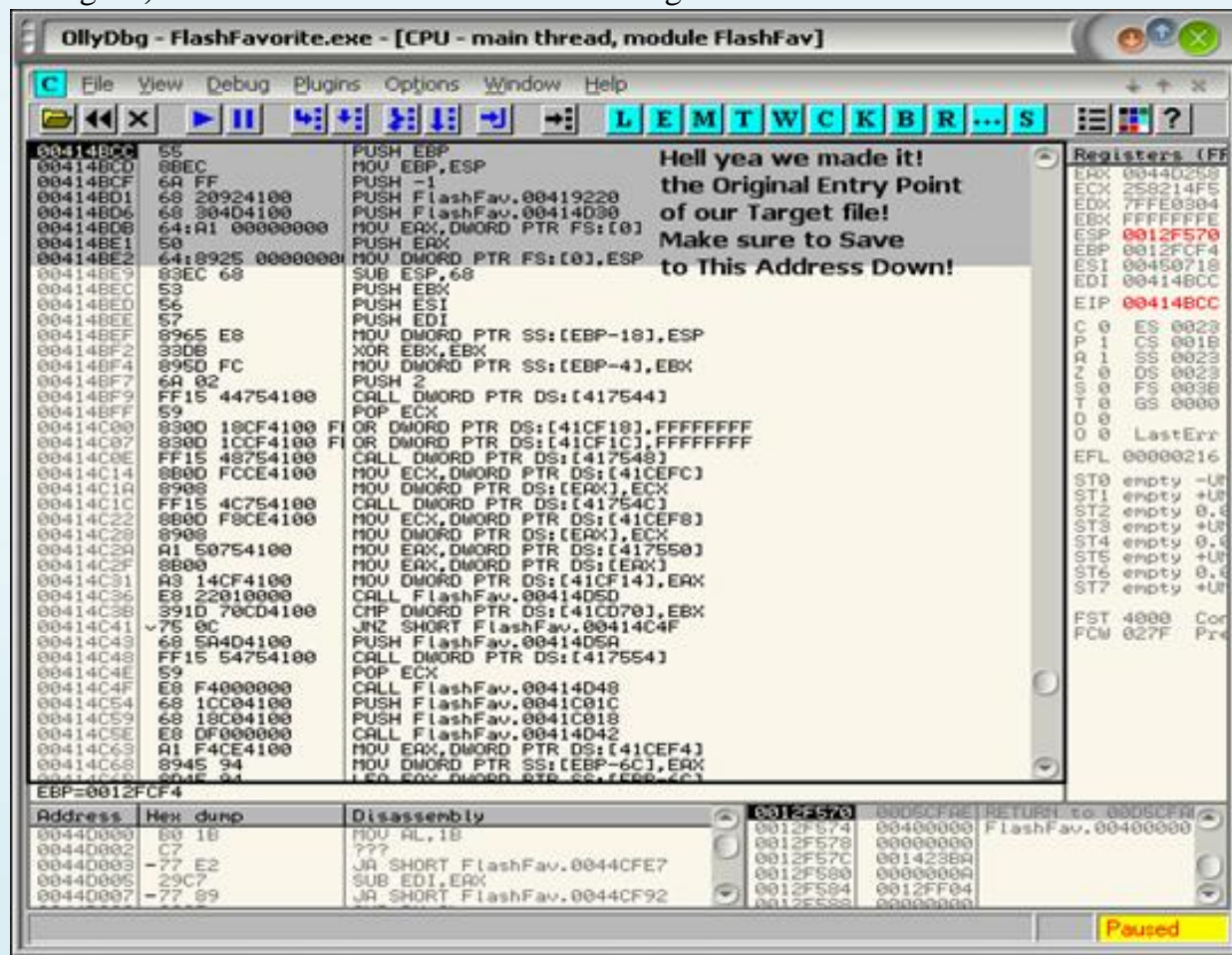
Trace into this RETN --^^^

And we will position the following similar pictures illustrated below:



-**** Set at a BP **Call EDI** functions, then press **F9** 1 times and they will function at Break **Call EDI**. When we break the function at the function that is called OEP function! : D. Now into the Trace Call this function by

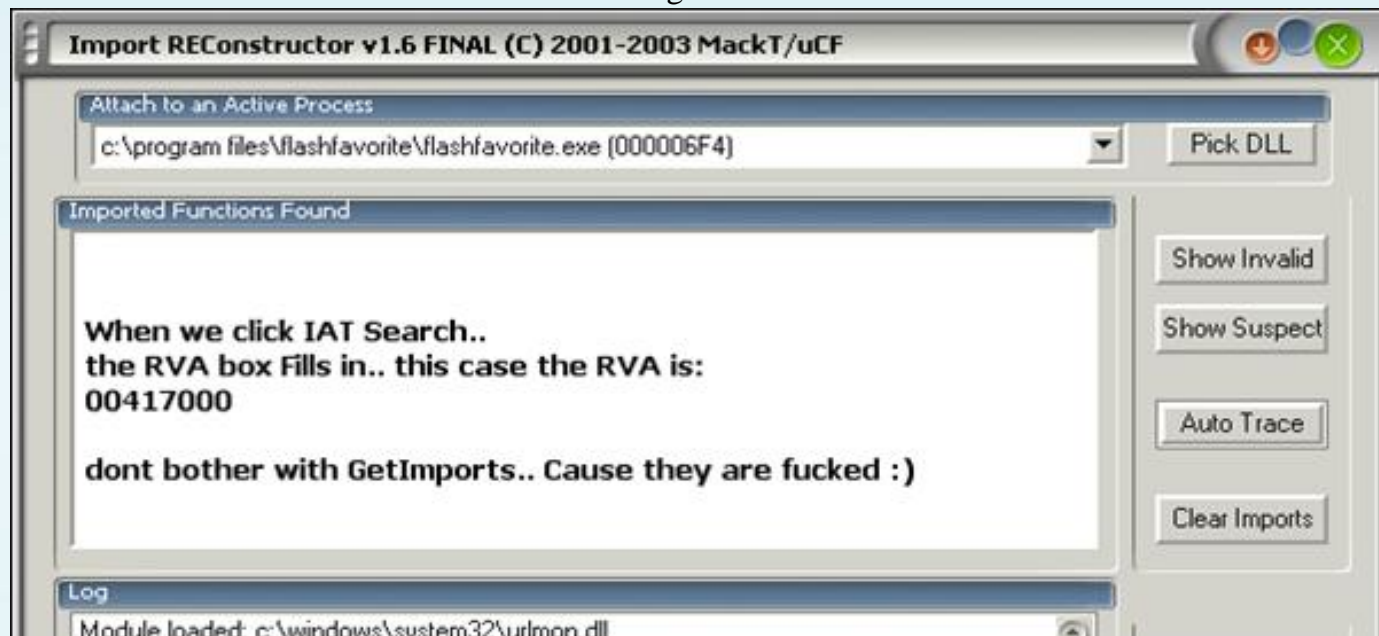
pressing **F7**, we'll come to a different code like Figure below:

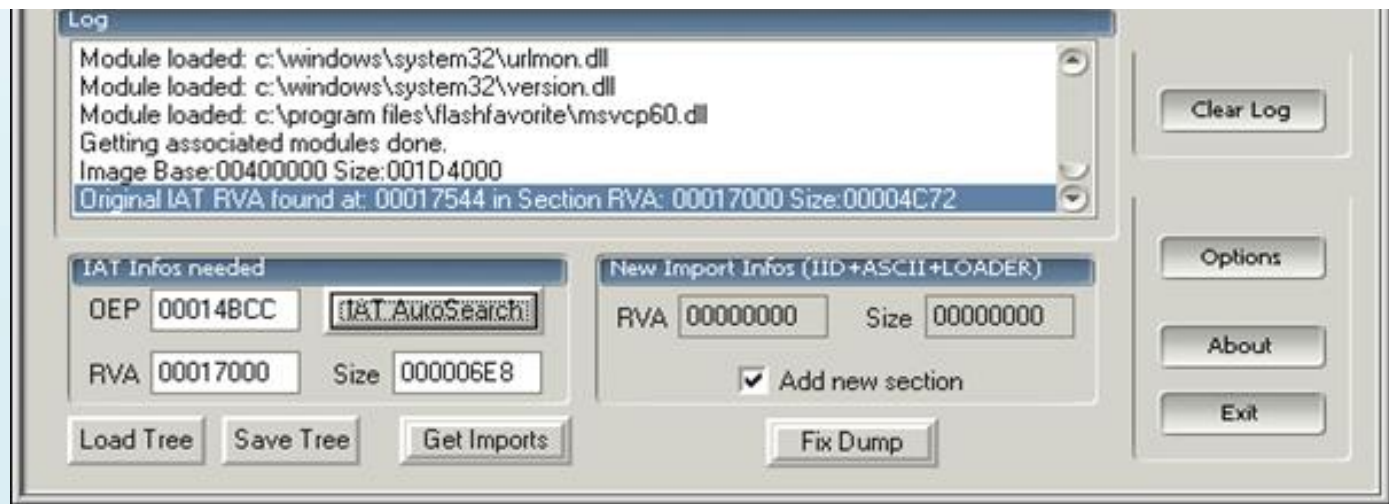


**** YEAH ... that in the imagination: D .. We've been doing that. The Original Entry Point is **00414BCC**. Now we will conduct the next job is with this **dump** file **LordPE!** . We just keep Olly screen (do not close) Open up **LordPE**, Highlight **FlashFavourite.exe** **Correct Process ID** with you in Attach Olly. Right Click on the Correct Process, and **dump (Full)** ... and save the file has been dumped into the installation program.

**** Congratulations! You just have OEP's armadillo + debug blocker rui it; not the currency must not do

**** Hope you Olly still open, the next we run the program ImpRec. At screen ImpRec ... we choose to file **FlashFavourite.exe** **Correct Process ID**. Enter by OEP to measure the real OEP formula: **OEP in Olly - ImageBase 00414BCC = - = 00014BCC ImageBase**. We enter this value to the OEP in ImpRec, then Click to select **Auto Search IAT**. We will be similar to Figure below:





**** Once we click to select IAT AutoSearch, next we will fill in values for RVA this case ... I is **RVA ImageBase + = 00017000 = 00417000**.

/ * Note: Always Save the RVA Address down. After you have the RVA ADDRESS you can close Imprec .. Now, I know its a Bitch ... but we will have to restart .. Crack and the Debug Blocker Again ... Make sure you file you Dumped from olly with LordPE.

Save OEP RVA and address in a text file ... ALL restart and Olly Debugs ...

(dont be confused, we will have to attack the debug blocker again tho (which sucks), in order to crack the Import methods of stealing armadillo.) * /

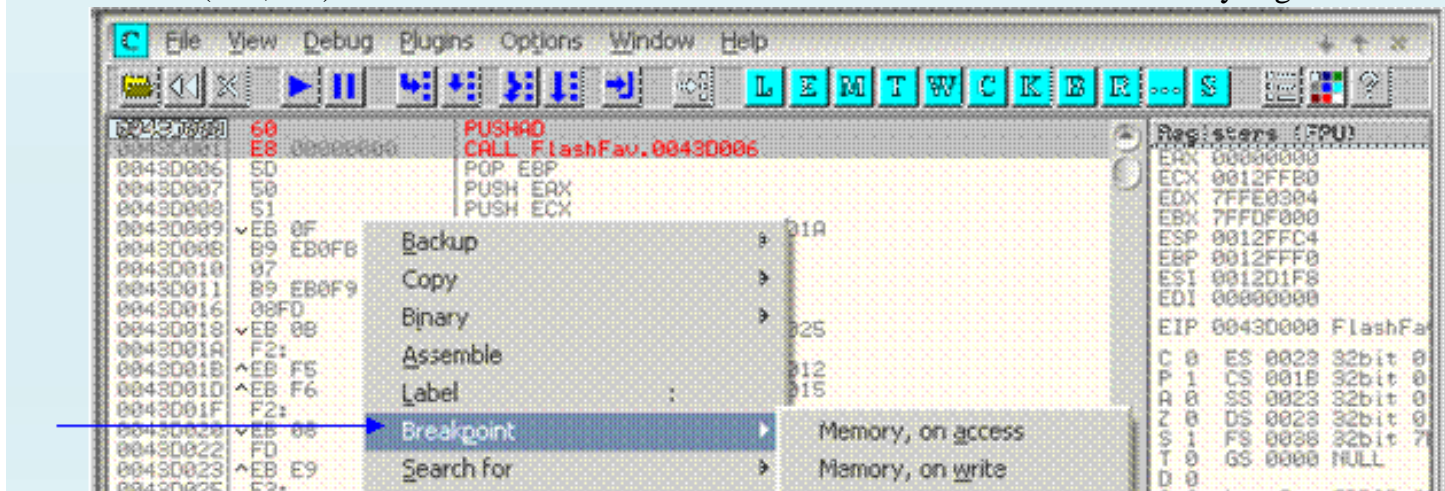
Part II: **Defeating Armadillo's Faking Import Methods and Reconstructing the ImpREC with Imports**

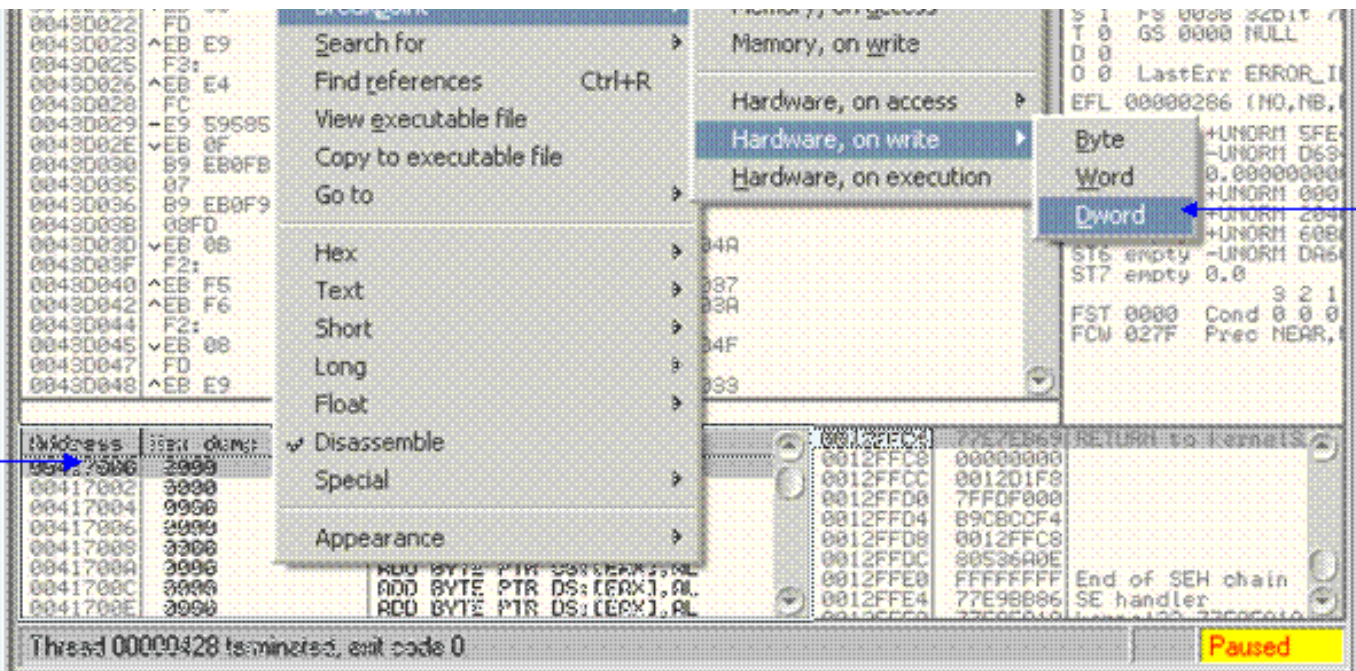
Some **** Armadillo Patch can be easily ... I wrote all the directions for you to understand more clearly about **Armadillo's Import Stealing** .. Sometimes this accounts for very little time but also when you take a long time can find the place to patch.

**** Most Armadillo have the same technical Patching we will discuss in this next section (sometimes with some technical Armadillo Patching quite a bit different), so part 2 of article writing will still be used more in the next period.

**** Oki, in writing this we will need to add **Defeat Debug Blocker** again. So please proceed to the steps as said in part of the message (unless you no longer remember what they have done in the a:)). Now after you Attach to Process ID ... Correct Patch **EB FE E8** to **60** blah blah ...

In **** window dump Windows, and select Rightclick Dissassemble to see in the dissassemble Press **Ctrl + G** in the window to open the dump Window Goto dialog box: type in RVA and we have been since the ImpRec ... before **you write it does not it?** Olly will bring us to address 417,000 Now we are again Right Click on the address RVA (417,000) selected **BeakPoint> Hardware> Write On> Dword**. Similarly Figure below:

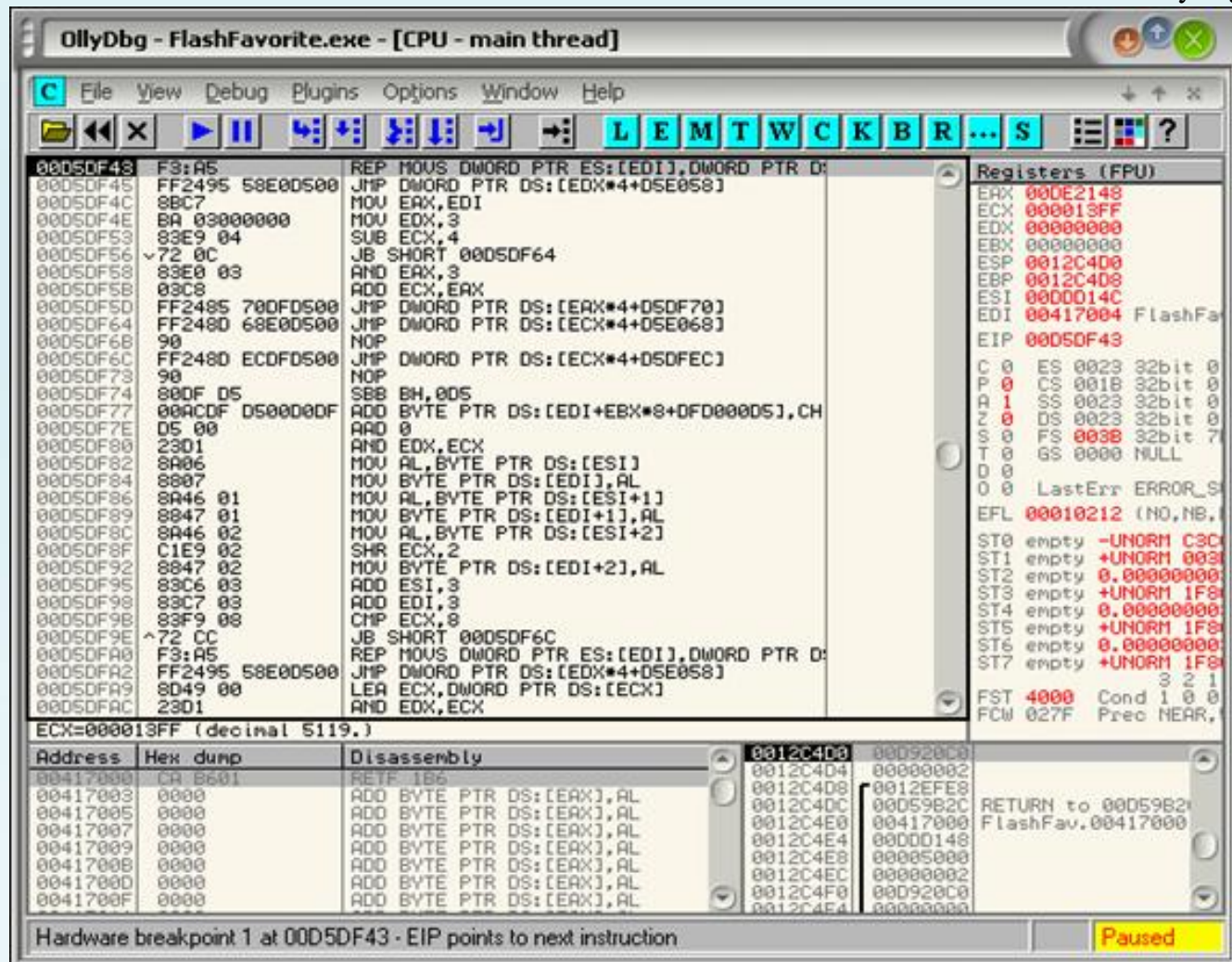




Oki -**** after you have set **hardware BREAKPOINT> on Write> Dword** at RVA (417,000 in this case ..) ... Remember that we still need to Fix function API IsDebuggerPresent So let's set in BP that function .. click **Shift + F9** until we break IsDebuggerPresent and Patch.

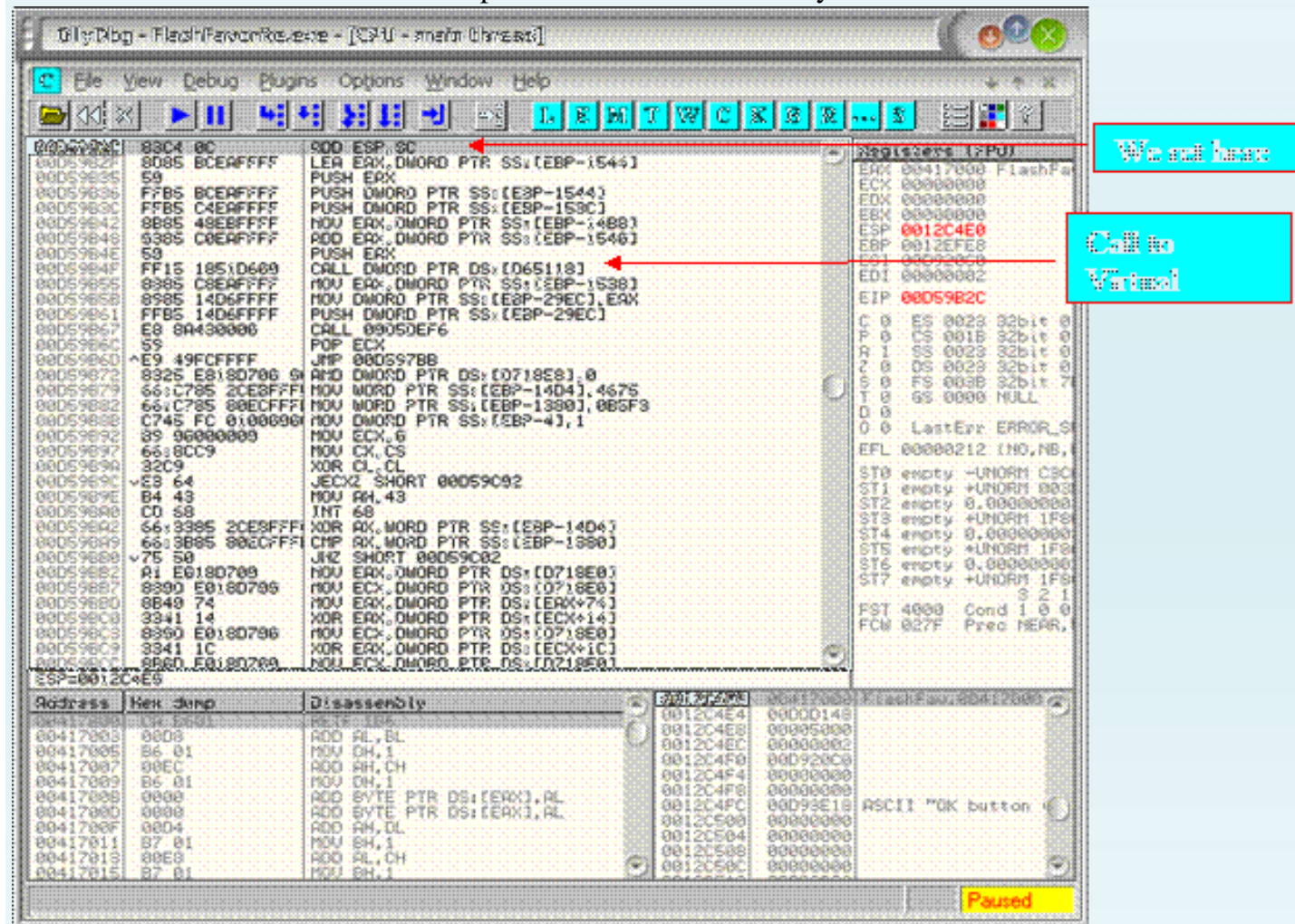
-**** Next press **Shift + F9** until we receive a notice Nag ... Click OK at Nag Screen

YES! We Break: **Hardware BreakPoint 1 at 00xxxxxx <address doesn't matter>**. Similarly Figure below:



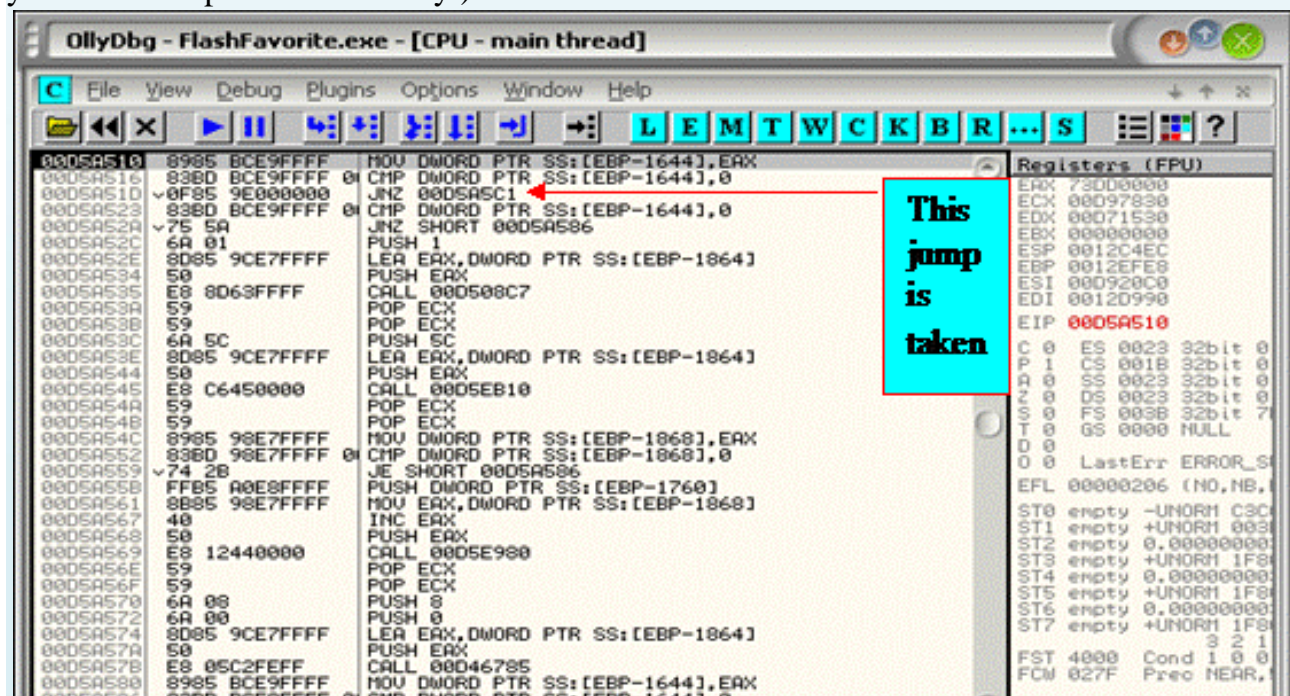
-**** The position that we need ... (not exactly where the armadillo steals imports ..). Press **Ctrl + F9** to Trace to

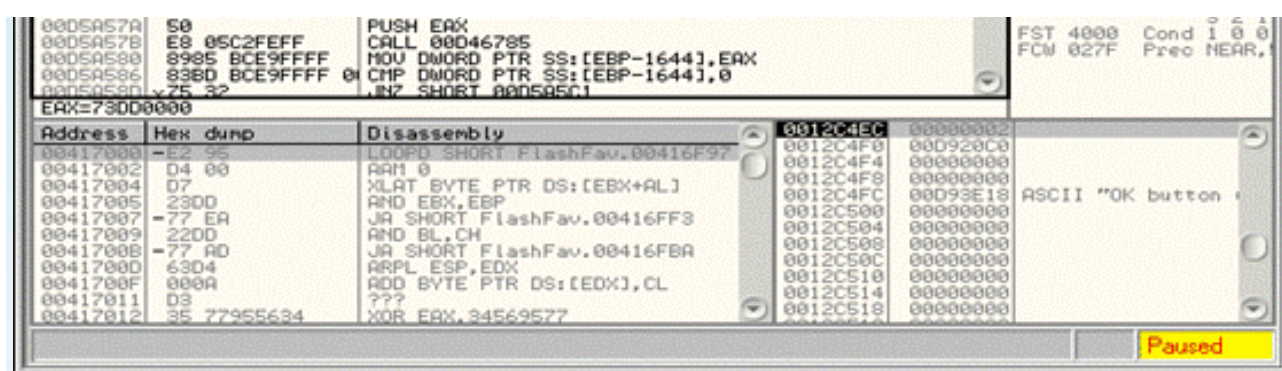
the nearest command **RETN ...** and press **F7** to Trace Into and you will be like illustrations below:



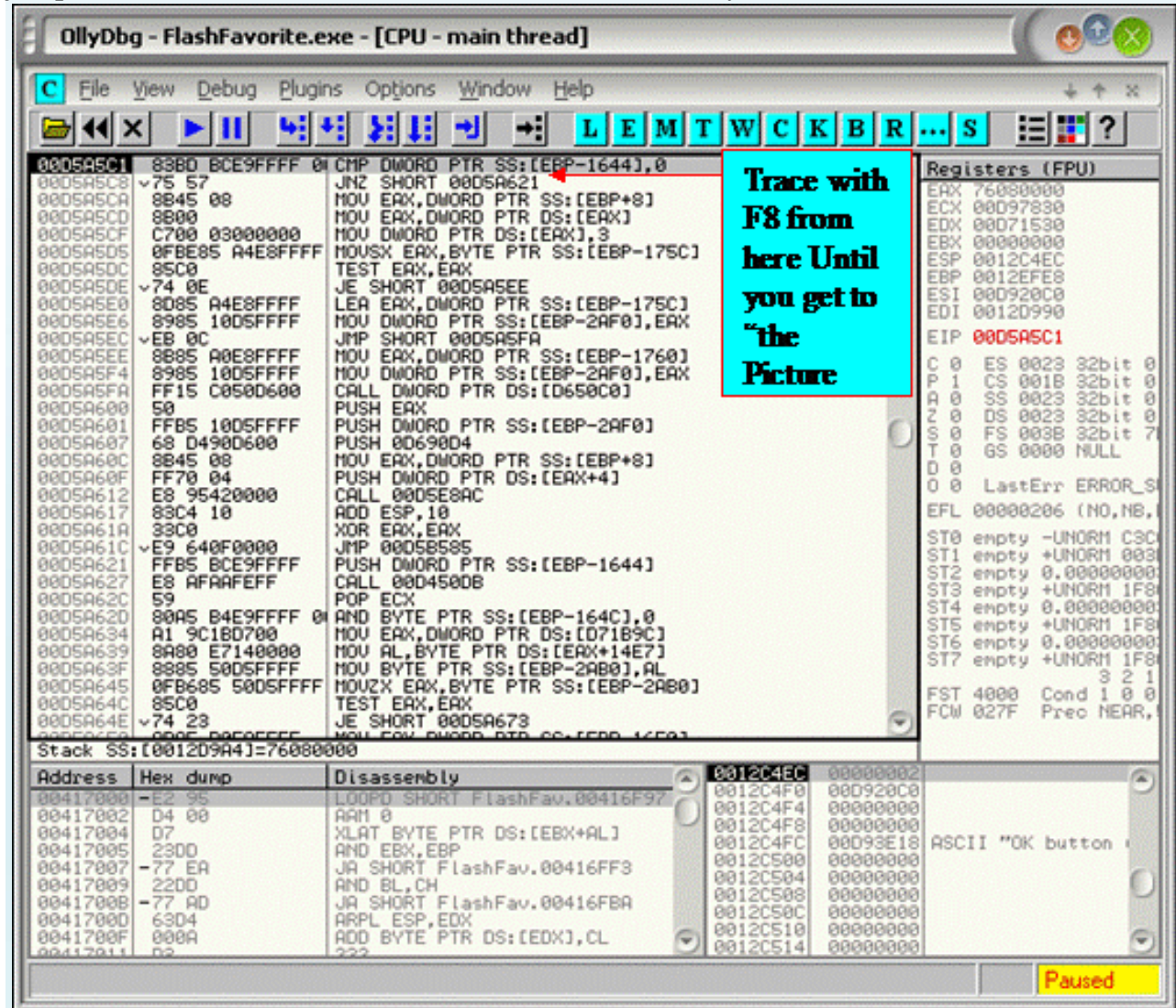
-**** Now ... When you get this in place in Armadillo, remember what the same ... (in most cases you can see a call to order me **VirtualProtect** after ADD ESP, 0C ..) this is easily the best to be recognized. (A lot of these patches are the same ... but some can be slightly differnt.).

-**** When we in the same image as illustrated above: press **Ctrl + F9** you can get access indoctrination or nothing ... Press **F7** if you get **Access Violation**. If you receive a **Hardware BP** is the press **Ctrl + F9** ... how is .. you will end up here: eventually:)

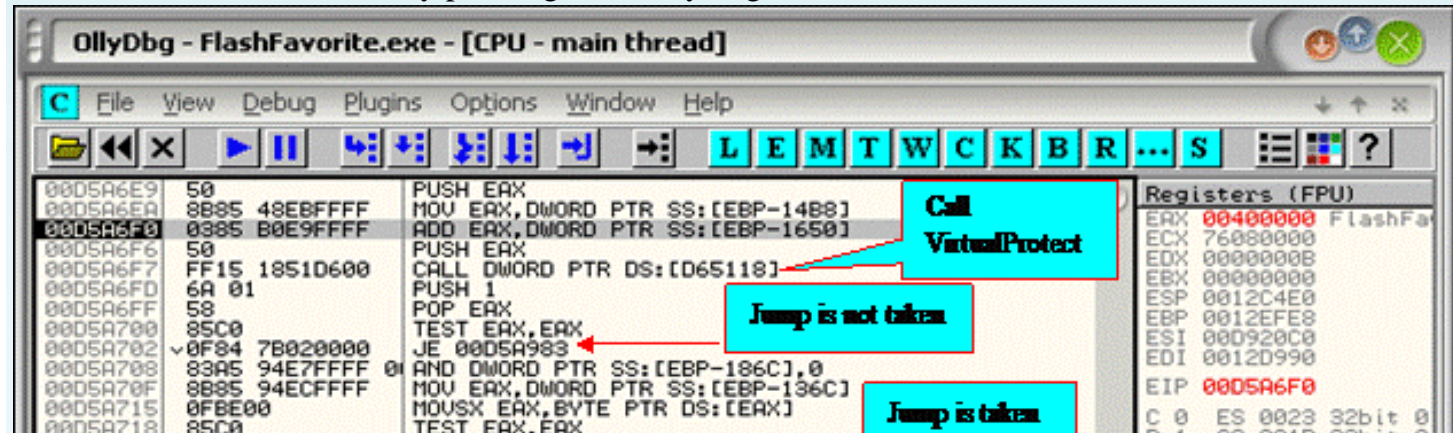




**** We will be like on .. (in my case) you might break .. Further Ahead (1 picture down ..). Trace until having jumped the first order ... and it is taken .. so take it .. and you will be here: z



**** So we continue Trace by pressing F8 until you get the same as here: Just do it:)



Jump is taken

You can Tell when you close. All this move Dword Ptr SS

Address	Hex dump	Disassembly
00D5A70F	8B85 94ECFFFF	MOV EAX,DWORD PTR SS:[EBP-136C]
00D5A715	0FB800	MOVSX EAX,BYTE PTR DS:[EAX]
00D5A718	85C0	TEST EAX,EAX
00D5A71A	0F85 0C010000	JNZ 00D5A82C
00D5A720	C785 44E7FFFF	MOV DWORD PTR SS:[EBP-18BC],0D462B2
00D5A72A	C785 48E7FFFF	MOV DWORD PTR SS:[EBP-18B8],0D46377
00D5A734	C785 4CE7FFFF	MOV DWORD PTR SS:[EBP-18B4],0D4639A
00D5A73E	C785 50E7FFFF	MOV DWORD PTR SS:[EBP-18B0],0D462D8
00D5A748	C785 54E7FFFF	MOV DWORD PTR SS:[EBP-18AC],0D46315
00D5A752	C785 58E7FFFF	MOV DWORD PTR SS:[EBP-18A8],0D4631A
00D5A75C	C785 5CE7FFFF	MOV DWORD PTR SS:[EBP-18A4],0D4631F
00D5A766	C785 60E7FFFF	MOV DWORD PTR SS:[EBP-18A0],0D46324
00D5A770	C785 64E7FFFF	MOV DWORD PTR SS:[EBP-189C],0D4634A
00D5A77A	C785 68E7FFFF	MOV DWORD PTR SS:[EBP-1898],0D46370
00D5A784	C785 6CE7FFFF	MOV DWORD PTR SS:[EBP-1894],0D462B2
00D5A78E	C785 70E7FFFF	MOV DWORD PTR SS:[EBP-1890],0D46377
00D5A798	C785 74E7FFFF	MOV DWORD PTR SS:[EBP-188C],0D4639A
00D5A7A2	C785 78E7FFFF	MOV DWORD PTR SS:[EBP-1888],0D462D8
00D5A7AC	C785 7CE7FFFF	MOV DWORD PTR SS:[EBP-1884],0D463AD
00D5A7B6	C785 80E7FFFF	MOV DWORD PTR SS:[EBP-1880],0D463B2
00D5A7C0	C785 84E7FFFF	MOV DWORD PTR SS:[EBP-187C],0D463B7
00D5A7CA	C785 88E7FFFF	MOV DWORD PTR SS:[EBP-1878],0D46324
00D5A7D4	C785 8CE7FFFF	MOV DWORD PTR SS:[EBP-1874],0D4634A

Stack SS:[0012D998]=000174F0

Address	Hex dump	Disassembly
00417000	-E2 95	LOOPE SHORT FlashFav.00416F97
00417002	D4 00	RAM 0
00417004	D7	XLAT BYTE PTR DS:[EBX+AL]
00417005	230D	AND EBX,EBP
00417007	-77 EA	JA SHORT FlashFav.00416FF3
00417009	220D	AND BL,CH
0041700B	-77 AD	JA SHORT FlashFav.00416FBA
0041700D	63D4	ARPL ESP,EDX
0041700F	000A	ADD BYTE PTR DS:[EDX],CL
00417011	D2	232

Paused

**** Now is the finicky .. not to be mistake if you have previously made a few times and then ... this is as easy to eat cakes:) .. JNZ execute that in the picture above ...

And now you will be here:

This jump is taken

This Call contains the Data which will need to be patched

Address	Hex dump	Disassembly
00D5A82C	8B85 94ECFFFF	MOV EAX,DWORD PTR SS:[EBP-136C]
00D5A832	0FB800	MOVSX EAX,BYTE PTR DS:[EAX]
00D5A835	3D FF000000	CMP EAX,0FF
00D5A83A	0F85 A7000000	JNZ 00D5A8E7
00D5A840	8B85 94ECFFFF	MOV EAX,DWORD PTR SS:[EBP-136C]
00D5A846	40	INC EAX
00D5A847	8B85 94ECFFFF	MOV DWORD PTR SS:[EBP-136C],EAX
00D5A84D	8B85 94ECFFFF	MOV EAX,DWORD PTR SS:[EBP-136C]
00D5A853	66:8B00	MOV AX,WORD PTR DS:[EAX]
00D5A856	66:8985 40E7FFFF	MOV WORD PTR SS:[EBP-18C0],AX
00D5A85D	8B85 94ECFFFF	MOV EAX,DWORD PTR SS:[EBP-136C]
00D5A863	40	INC EAX
00D5A864	40	INC EAX
00D5A865	8B85 94ECFFFF	MOV DWORD PTR SS:[EBP-136C],EAX
00D5A868	0FB785 40E7FFFF	MOVSX EAX,WORD PTR SS:[EBP-18C0]
00D5A872	50	PUSH EAX
00D5A873	FFB5 BCE9FFFF	PUSH DWORD PTR SS:[EBP-1644]
00D5A879	E8 A0B1FEFF	CALL 00D45A1E
00D5A87E	8B85 94E7FFFF	MOV DWORD PTR SS:[EBP-186C],EAX
00D5A884	83B0 94E7FFFF	CMP DWORD PTR SS:[EBP-186C],0
00D5A88B	75 58	JNZ SHORT 00D5A8E5
00D5A88D	FF15 C050D600	CALL DWORD PTR DS:[D650C0]
00D5A893	83F8 32	CMP EAX,32
00D5A896	75 0A	JNZ SHORT 00D5A8A2
00D5A898	C785 94E7FFFF	MOV DWORD PTR SS:[EBP-186C],0D458CA
00D5A8A2	83B0 94E7FFFF	CMP DWORD PTR SS:[EBP-186C],0
00D5A8A9	75 3A	JNZ SHORT 00D5A8E5
00D5A8AB	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]
00D5A8AE	8B00	MOV EAX,DWORD PTR DS:[EAX]
00D5A8B0	C700 03000000	MOV DWORD PTR DS:[EAX],3
00D5A8B6	FF15 C050D600	CALL DWORD PTR DS:[D650C0]
00D5A8BC	50	PUSH EAX
00D5A8BD	0FB785 40E7FFFF	MOVSX EAX,WORD PTR SS:[EBP-18C0]

Stack SS:[0012C7C]=00000083, (ASCII "?0?%\$basic_string@OU?%char_traits@D@std@U?%\$allocat

Address	Hex dump	Disassembly
00417000	-E2 95	LOOPE SHORT FlashFav.00416F97
00417002	D4 00	RAM 0
00417004	D7	XLAT BYTE PTR DS:[EBX+AL]
00417005	230D	AND EBX,EBP
00417007	-77 EA	JA SHORT FlashFav.00416FF3
00417009	220D	AND BL,CH
0041700B	-77 AD	JA SHORT FlashFav.00416FBA
0041700D	63D4	ARPL ESP,EDX
0041700F	000A	ADD BYTE PTR DS:[EDX],CL
00417011	D2	232

Paused



-**** Technical method of Armadillo it is encrypted part of the code (Encrypts Part of this Code) ... and then in code Work on it simply will Armadillo is the code for you .. we'll patch in a dance party in order code is good code. You could observe Call function in Figure in:). You can get out is Decryption Code because it has 3 commands around Call function that, by the order are:

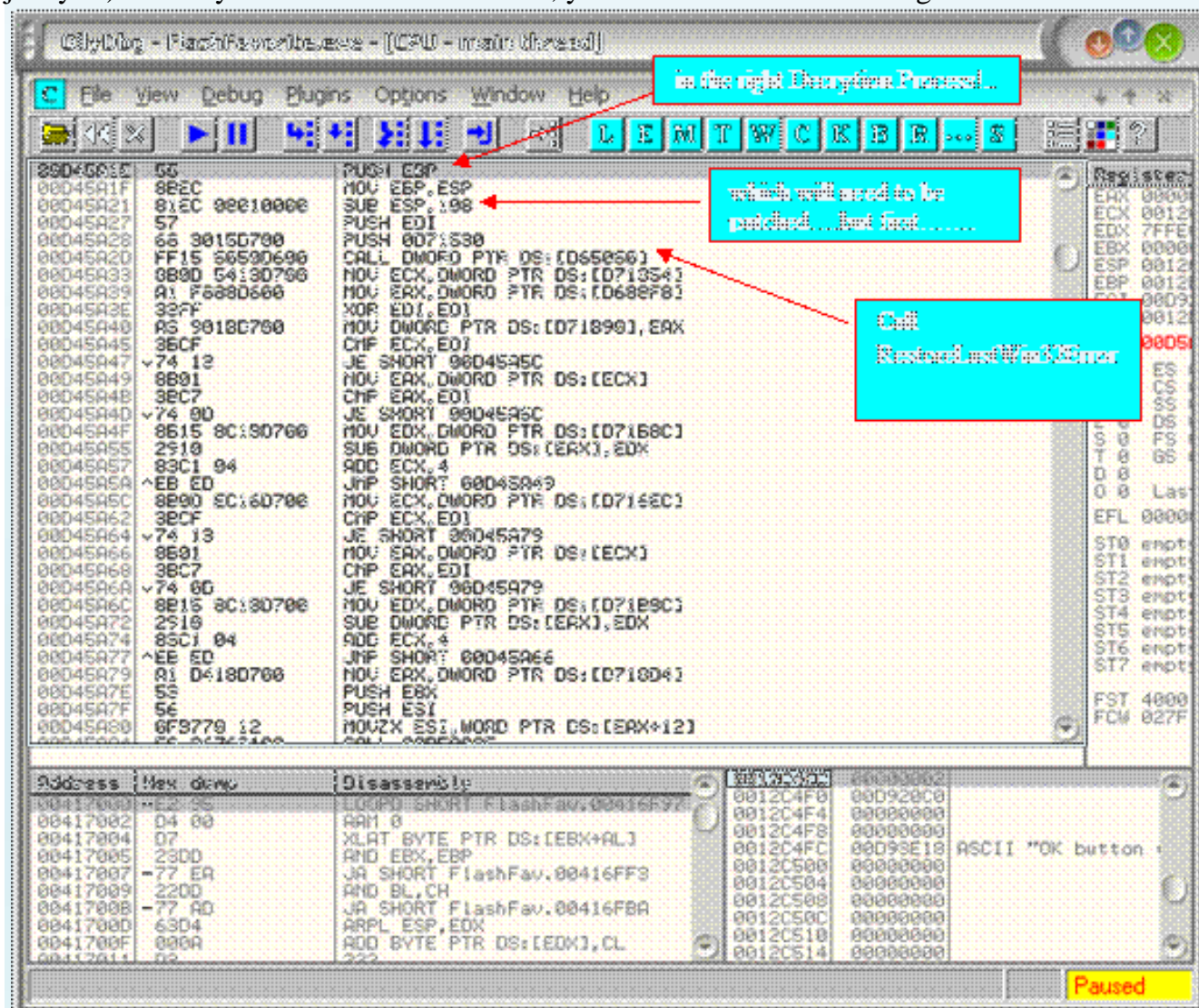
FFB5 BCE9FFFF **PUSH DWORD PTR SS: [EBP-1644]** <--

E8 A0B1FEFF **CALL 00D45A1E**

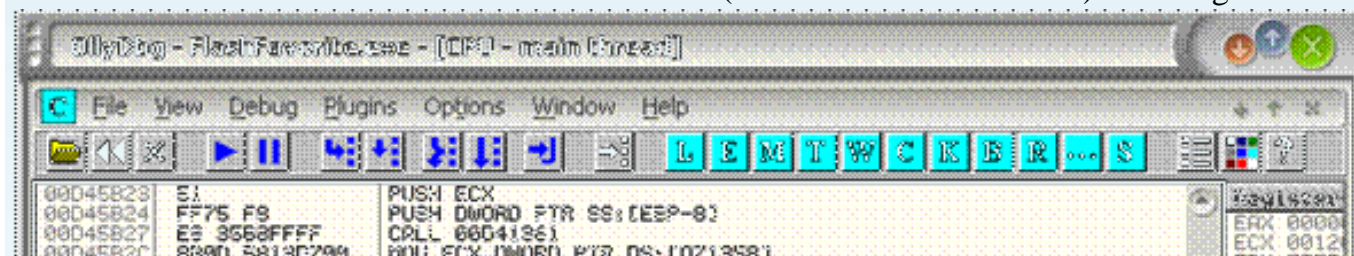
8985 94E7FFFF **MOV DWORD PTR SS: [EBP-186C], EAX** <--

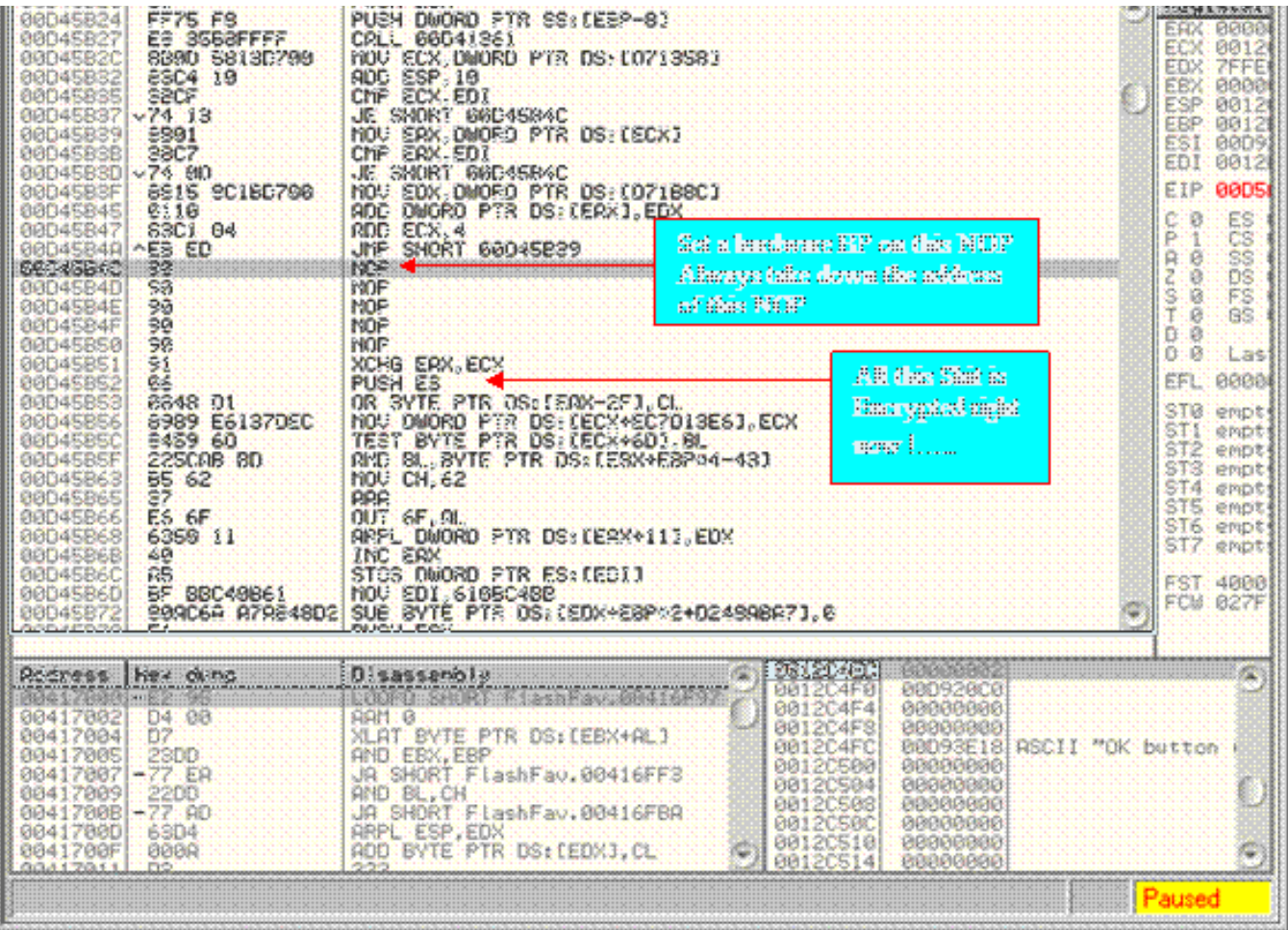
83BD 94E7FFFF **Cmp DWORD PTR SS: [EBP-186C], 0** <--

-**** The Call Decryption where we need to patch Armadillo, so we restore Imports. Highlight the Call to function, and press Enter> Enter will follow the Call Address (because as you can see .. this is not call Executed just yet!). When you follow the instructions, you will here the same as Figure:



-**** Now when we were here ... All these jumps are just shit and shit decrypting .. Therefore, in Olly we scroll down the mouse a bit until we see some order NOP (about 4 or 5 orders NOPs). Like Figure below:

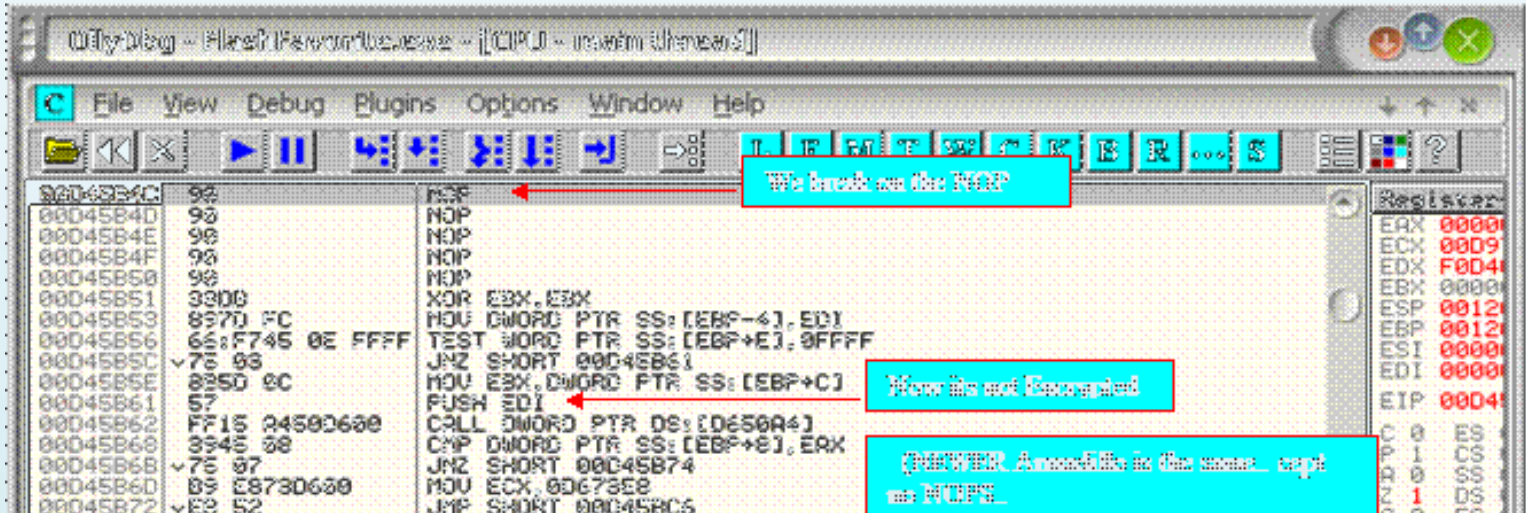




-**** Save the address of this command **NOP** ... (we'll call it: The Decryption NOP). That is the position we Patch Armadillo so we need to rebuild Imports by ImpRec tool. :). But the file that we are in a load that Olly Debug Blocker .. so we must set up a Hardware Child Process in BP, the hardware will not bpx Save ... So we need to save the address of the command by NOP by writing this address a text document. Next we press **Ctrl + F2** to Restart Olly again ..

-**** We must again Defeat Debug Blocker again Attach the child process and v. Now when we Attach Process Right, sure that we still need to Fix IsDebuggerPresent. So in the screen of Olly, we set up at a BP function IsDebuggerPresent (you wont break on it just yet ... if you do, patch it ..)

-**** Press **Shift + F9 2 times**, followed by pressing **Ctrl + G** and enter the address of Decryption Submit we have recorded a text document, then set up a Hardware in BP Decryption Submit this. Finally you press **Shift + F9** you will break in function IsDebuggerPresent ..: Fix IsDebuggerPresent so only the owner chận What else .. Once you Patching IsDebuggerPresent, continue to press **Shift + F9** to Run program, we will receive Nag is a screen, click OK, we'll break in Decryption NOP:). Similar images illustrated below:



The screenshot shows the assembly window of OllyDbg. The assembly code is as follows:

```

00045B68 75 07 JNZ SHORT 00045B74
00045B6D 09 E8730630 MOV ECX, 006735E8
00045B72 E2 52 JMP SHORT 00045B85
00045B74 3930 E0790630 CMP DWORD PTR DS:[0679E0], EDI
00045B7A 09 E0790630 MOV ECX, 00679E0
00045B7F 0F84 93303030 JE 00045C18
00045B85 8335 60803630 MOV ESI, DWORD PTR DS:[060860]
00045B8B A1 E0180720 MOV EAX, DWORD PTR DS:[0718E0]
00045B90 F641 50 01 TEST BYTE PTR DS:[ECX+0], 1
00045B94 74 0E JE SHORT 00045BA4
00045B96 8550 5C MOV EDX, DWORD PTR DS:[EAX+5C]
00045B99 3350 48 XOR EDX, DWORD PTR DS:[EAX+48]
00045B9C 3350 1C XOR EDX, DWORD PTR DS:[EAX+1C]
00045B9F F6C2 90 TEST DL, 00
00045BA2 75 13 JNZ SHORT 00045BB7
00045BA4 8550 5C MOV EDX, DWORD PTR DS:[EAX+5C]
00045BA7 3350 50 XOR EDX, DWORD PTR DS:[EAX+50]
00045BA9 3350 44 XOR EDX, DWORD PTR DS:[EAX+44]
00045BAD 3350 20 XOR EDX, DWORD PTR DS:[EAX+20]
00045BB0 3315 XOR EDX, DWORD PTR DS:[ESI]
00045BB2 3955 08 CMP DWORD PTR SS:[EBP+8], EDX
00045BB5 74 0C JE SHORT 00045BC5
00045BB7 83C1 2C ADD ECX, 2C

```

Annotations on the right side of the assembly window:

- (HARDLY Amardillo at this point... just in NOPS... to break on some random instructions)
- we can navigate where to patch with the test DL, 00
- All we have to do is NOP this jump :)

The registers window on the right shows the following values:

```

EAX 00000000
ECX 00000000
EDX 00000000
EBX 00000000
ESP 00000000
EBP 00000000
ESI 00000000
EDI 00000000
EIP 00045B85
EFL 00000000
ST0 empty
ST1 empty
ST2 empty
ST3 empty
ST4 empty
ST5 empty
ST6 empty
ST7 empty
FST 4000
FCW 027F

```

The disassembly window at the bottom shows the following code:

```

Address 00417000 Dump 02 55 Disassembly LOOPD SHORT FlashFav.00416F9C
00417002 04 00 RAR 0
00417004 07 XLAT BYTE PTR DS:[EBX+AL]
00417005 2300 AND EBX, EBP
00417007 77 EA JA SHORT FlashFav.00416FF3
00417009 2200 AND BL, CH
0041700B 77 AD JA SHORT FlashFav.00416FBA
0041700D 6304 ARPL ESP, EDX
0041700F 000A ADD BYTE PTR DS:[EDX], CL
00417011 03 ???
00417012 35 77955634 XOR EAX, 34569577
00417017 77 EC JA SHORT FlashFav.00417005

```

The hardware breakpoint window at the bottom shows a breakpoint set at address 00045B8C, and the status is Paused.

-**** Now we can see orders jumped after testing ordered by **Test DL, 80**. 2 orders jumped below me Test command. So we or submit orders jumped JE. We conducted Patch dance in order to make Amardillo never interfere Imports:). Leaving us with about 6 to cut thunks

-**** We have to remove me from BP hardware orders submitted. You can make by clicking> Debug> Hardware Breakpoints in Olly.

Next -**** run by pressing Shift + F9 once

(READ FIRST):

(IN MOST CASES: When you make this patch, the EXE will not Run .. you will get a privileged instruction or something like that .. and you will then fix with imprec ..)

But in this case, our exe file will run after we Patch! **SO! We just unpacked Arma-Fucking-DILLO**. You still keep windows Olly đây too? Do not Close:)

-**** Now we open up ImpRec ...

- 1.) Enter the value OEP (00414BCC) - ImageBase = 00014BCC
- 2.) Click to select IAT AutoSearch
- 3.) Next click Get Imports (Now All the imports will be there! FUCKING HELL yea!)
- 4.) Select Show Invalid (Show invalid, select all invalid thunks)
- 5.) Right Click on the Invalid Thunks and select Cut Thunks

the invalid thunks are left by armadillo .. and are just surrounding the IAT ... (so just cut the remaining thunks)

-**** Now all are Thunks Valid! Finally we choose to click and select Fix dump dump file that we dump before!

-**** HMmmm We go to Olly and breathed a very deep .. to relax after a process of stress and fatigue ... then we run the file has Dumped + Import File Fixed up **HOLY fuck! RUNSS it!**

-**** I hope that with what they have trans will be a number of medical problems Armadillo v3! .. Due to limited capabilities should not translate correctly the steps of the author but they have tried hard to translate this article offers a little something for not only myself but for them all Members of the REA and BQT:) **Thanx 4**

All J

oOo **GreetZ** oOo and **Special Thanks:**

Computer_Angel, Moonbaby, Zombie, Benina, QhQcrker, RongChauA, Nini, Hoadongnoi, the_lighthouse, hacnho, Deux and all my friends All REA 's member

Thanx to -****- authors of this tut -****- -:



Armadillo x.xx - collect sand stone

He backpack, I used to do arm as claimed chung steps to MUPing road. The new start-aged lovers of the tart surely not sure this bit and never sold the bags to first, why, because it is too sour, too spicy. I say I fear the thăng SI, but I do thăng Hai Hung is thăng arm, she pretended that I rock eyelash bags, I did not fear in the complete end of IAT, which fix dump bong thăng ImpREC it reports "Can not add any sections. Very interesting, I had to unpack many soft pack with arm but never dare to put pen to write a tut called unpack. I actually said, the manufacturer of the arm as it is insane. Who lives each ver it has more than a dozen options to protect the approximately 30 months to protect. How many it has not been back to New ver. Each home page on its bags blindfold to see dek dare. Hi huc forever with v4.05, khoai too light it up ver 4.10. Each ver it all add more new models that protect the excellent way the people . SecurityTeam parties have announced a bug's OllyDBG. The aged do not believe, as I read lười information about security, should a target load the bags onto Olly Shift + F9 it crash. I edit Options is the size. The following are ver apply bug (OutputDebugStringA API, arm function used to send the string Olly overflow error based caching, summer summer, the crash is) to make an anti-debug the arm. More in <http://www.securiteam.com/windowsntfocus/5ZP0N00DFE.html>

Currently on the NET about 100 tuts written arm, sufficient enough in style, but all originated from Cracks Latinos. Since then many tuts the cover again. Head of the mem ku Fly pier, then Exetools, Woodman, BoadAnticrack, UnpackingGOD, BIW, RET, ICU, ARTEAM, RES is next to last and snd tsRH, month tsrh insurance is not the best ever written only tuts unpack and release. At AVATAR / RES () is a group of emerging professional arm, both employers SoftwarePassport release ver is it release the patch inline line. Good boy! Vietnam also has its three aged write some tut unpack arm typically aged tlandn wrote the tut unpack crackme by VCT, aged trans Com a tut group Cracks Latinos, aged hhphong bags to put together written tut see: D. Today I Decision hear your baby sitting hoadongnoi cover all tut or to a general basic tut from easy to difficult for the aged convenient searching. All the tuts I have written this way to look at the bags, which means that many of the other org tut but reading performance in line, some months it written that there ulterior, dek read out many belgium. When you write a set of hands, I also know chả never complete new again. Each target I lost about 30 'to unpack + + photograph should think chả bít to ever ... ever!

I have a hand in her basement lang 50 tuts should also say ver arm which I was playing well;;). One before started to unpack your arm is to be patient and very privation, as if it was not clear what price you are out of place sitting not only publishing it on the ice for one. Remember to unpack only a passion is not the final points of cracking. If you unpack not only do not have the problem. Bags here, do not unpack the bags soft line to unpack her bags in: D.

To read the tut, asked to select Hide White Space in WinWord.

I. Tools used in this tut portfolio simply have only a few:

Patch-OllyDBG + + 2 hours **AntiDetectOllly_v2.2.4.exe** patch type anti debug I just said above. How to patch attached:

a.Cach 1:

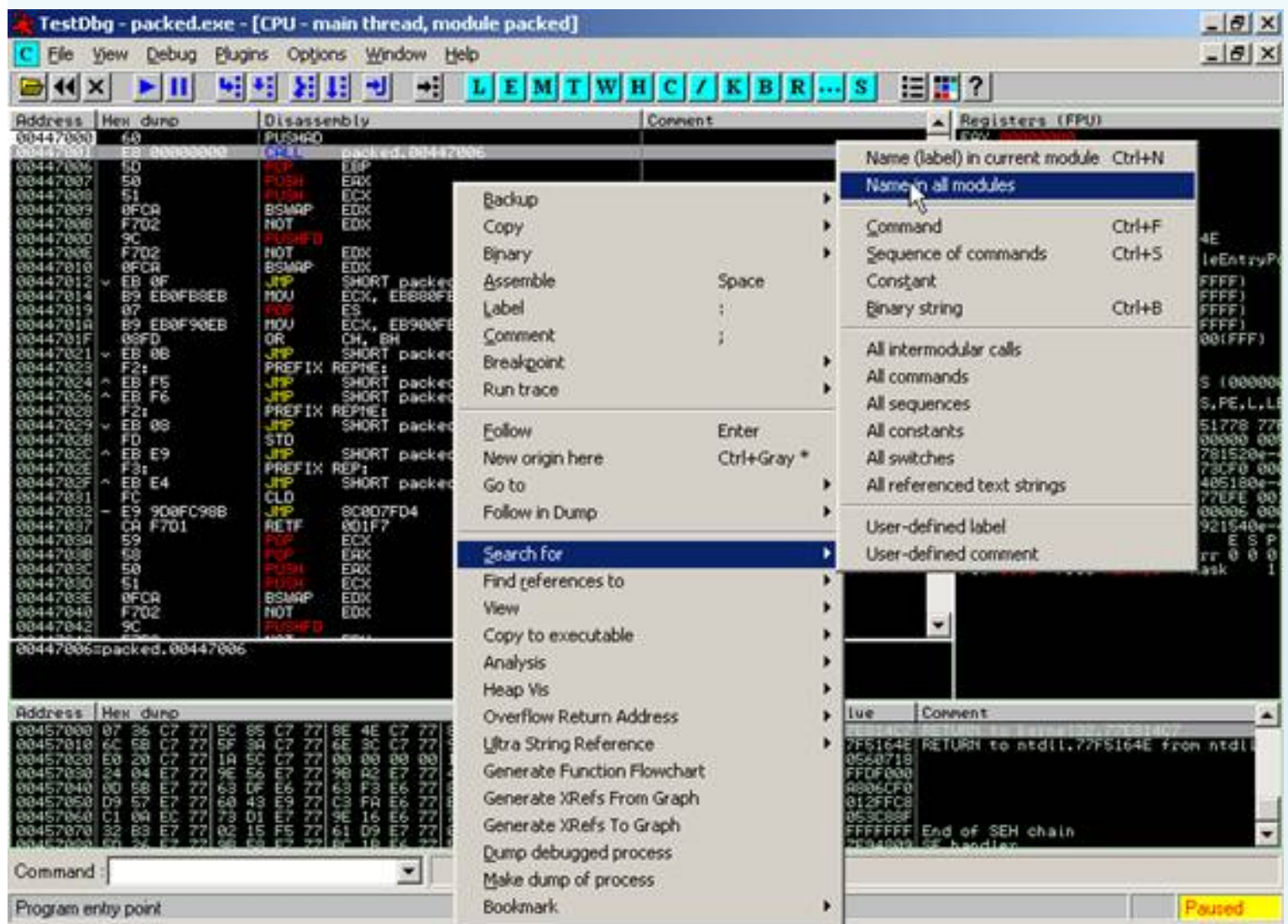
_Load OllyDBG on target. R-Click Search for select-> Names in all modules.

_Cuon Screen down **OutputDebugStringA** search function.

_Double Click on this you will function on the CPU window, you will at this address:

77E949B7> 68 2C020000 PUSH 22C

_Armadillo Will call this function, we must patch, assemble lines from **22C** to **PUSH RETN 4**. Done, see Figure attached:



Address	Module	Section	Type	Name	Comment
77060D98	USER32	.text	Export	OpenIcon	
803CA600	Teerayoo	.idata	Import	USER32.OpenInputDesktop	
77060D43	USER32	.text	Export	OpenInputDesktop	
77EB8AC5	kernel32	.text	Export	OpenJobObjectA	
77EB8974	kernel32	.text	Export	OpenJobObjectW	
803CA494	Teerayoo	.idata	Import	KERNEL32.OpenMutexA	
804578A4	packed	.data	Import	KERNEL32.OpenMutexA	
77E82391	kernel32	.text	Export	OpenMutexA	
803CA498	Teerayoo	.idata	Import	KERNEL32.OpenMutexW	
77E816E1	kernel32	.text	Export	OpenMutexW	
803CA49C	Teerayoo	.idata	Import	KERNEL32.OpenProcess	
77E72E23	kernel32	.text	Export	OpenProcess	
803CA0EC	Teerayoo	.idata	Import	ADVAPI32.OpenProcessToken	
77002FCD	ADVAPI32	.text	Export	OpenProcessToken	
788012B0	RPCRT4	.text	Import	ADVAPI32.OpenProcessToken	
77E69907	kernel32	.text	Export	OpenProfileUserMapping	
77006B90	ADVAPI32	.text	Export	OpenSCManagerA	
77002168	ADVAPI32	.text	Export	OpenSCManagerW	
788012C4	RPCRT4	.text	Import	ADVAPI32.OpenSCManagerW	
77E95768	kernel32	.text	Export	OpenSemaphoreA	
77E6A2AC	kernel32	.text	Export	OpenSemaphoreW	
770E824F	ADVAPI32	.text	Export	OpenServiceA	
770021F1	ADVAPI32	.text	Export	OpenServiceW	
788012C8	RPCRT4	.text	Import	ADVAPI32.OpenServiceW	
77E93DFC	kernel32	.text	Export	OpenThread	
77001EAE	ADVAPI32	.text	Export	OpenThreadToken	
78801330	RPCRT4	.text	Import	ADVAPI32.OpenThreadToken	
77E1B4D6	ADVAPI32	.text	Export	OpenTraceA	
77E1B5CF	ADVAPI32	.text	Export	OpenTraceW	
77EB4B20	kernel32	.text	Export	OpenWaitableTimerA	
77E92372	kernel32	.text	Export	OpenWaitableTimerW	
77D6867B	USER32	.text	Export	OpenWindowStationA	
77D6AD4E	USER32	.text	Export	OpenWindowStationW	
77C20800	MSUCRT	.text	Export	_open_osfhandle	
77C5C9C8	MSUCRT	.data	Export	_osplatform	
77C5C9C4	MSUCRT	.data	Export	_osver	
77C2DF28	MSUCRT	.text	Export	_outp	
77C2DF3A	MSUCRT	.text	Export	_outpd	
803CA4A0	Teerayoo	.idata	Import	KERNEL32.OutputDebugStringA	
77121390	OLEAUT32	.text	Import	KERNEL32.OutputDebugStringA	
77E949B7	kernel32	.text	Export	OutputDebugStringA	
77001510	ADVAPI32	.text	Import	KERNEL32.OutputDebugStringW	
77EAF0F8	kernel32	.text	Export	OutputDebugStringW	
77C2DF2D	MSUCRT	.text	Export	_outpw	
8045726C	packed	.data	Import	USER32.PackDDEIPParam	
77D6A000	USER32	.text	Export	PackDDEIPParam	
77D4BD56	USER32	.text	Export	PaintDesktop	
77D5869F	USER32	.text	Export	PaintMenuBar	
77C8A07D	GDI32	.text	Export	PaintRgn	
7639124C	IMM32	.text	Import	GDI32.PatBlt	

Address	Hex dump	Disassembly	Comment
77E949B7	68 2C020000	PUSH 22C	
77E949BC	68 8853E977	PUSH kernel32.77E95388	
77E949C1	E8 1259FEFF	CALL kernel32.77E7A2D8	
77E949C6	8B65 FC 00	AND DWORD PTR SS:[EBP-4], 0	
77E949CA	8B4D 08	MOV ECX, DWORD PTR SS:[EBP+8]	
77E949CD	8BC1	MOV EAX, ECX	
77E949CF	8D70 01	LEA ESI, DWORD PTR DS:[EAX+1]	
77E949D2	8A10	MOV DL, BYTE PTR DS:[EAX]	
77E949D4	40	INC EAX	
77E949D5	84D2	TEST DL, DL	
77E949D7	75 F9	JNZ SHORT kernel32.77E949D2	
77E949D9	2BC6	SUB EAX, ESI	
77E949DB	40	INC EAX	
77E949DC	8945 E0	MOV DWORD PTR SS:[EBP-20], EAX	
77E949DF	894D E4	MOV DWORD PTR SS:[EBP-1C], ECX	
77E949E2	8D45 E0	LEA EAX, DWORD PTR SS:[EBP-20]	
77E949E5	50	PUSH EAX	
77E949E6	6A 02	PUSH 2	
77E949E8	6A 00	PUSH 0	
77E949EA	68 06000140	PUSH 40010006	
77E949EF	E8 43EEFDFF	CALL kernel32.RaiseException	
77E949F4	834D FC FF	OR DWORD PTR SS:[EBP-4], FFFFFFFF	
77E949F8	E8 A259FEFF	CALL kernel32.77E7A39F	
77E949FD	C2 0400	RETN 4	
77E94A00	6A 57	PUSH 57	
77E94A02	E8 A959FEFF	CALL kernel32.77E7A3B0	
77E94A07	33C0	XOR EAX, EAX	
77E94A09	E9 2C90FEFF	JMP kernel32.77E7DA3A	
77E94A0E	6A FF	PUSH -1	
77E94A10	57	PUSH EDI	
77E94A11	6A FF	PUSH -1	
77E94A13	53	PUSH EBX	
77E94A14	6A 01	PUSH 1	
77E94A16	E8 FB7FFDFF	CALL kernel32.GetSystemDefaultLCID	
77E94A1B	50	PUSH EAX	



Address	Hex dump	Disassembly	Comment
77E949B7	C2 0400	RETN 4	
77E949BA	90	NOP	
77E949BB	90	NOP	
77E949BC	68 8853E977	PUSH kernel32.77E95388	
77E949C1	E8 1259FEFF	CALL kernel32.77E7A2D8	
77E949C6	8365 FC 00	AND DWORD PTR SS:[EBP-4], 0	
77E949CA	8B4D 08	MOV ECX, DWORD PTR SS:[EBP+8]	
77E949CD	8BC1	MOV EAX, ECX	
77E949CE	8D70 01	LEA ESI, DWORD PTR DS:[EAX+1]	

b.Cach 2:

_Buoc Such as 1,2,3

4 _Buoc patch instead of the 22C PUSH RETN4 we do the following:

_Xac The address of the function kernel32.OutputDebugStringA with APIAddress.



_Go To address both found, scroll down to find the screen to order the end of this function form:

77E949BA 33C0 XOR EAX, EAX

77E949BC 40 INC EAX

77E949BD C3 RETN;

3 _Binary copy lines. To top jaw, paste 4 bytes to Push 22C.

ImpREC 1.6-Final

1.0-APIAddress

- Re-Pair v0.6

Some target-(unpackme and soft).

II. Let's Unpacking.

Speaking before one, in the following tuts I will not speak to the anti-antidebug again. The aged know from his duties when kernel patch load target. Under the order, I will write the first easy-tut, tuts hard to write the following. The tuts I have written sucessful unpack, rest assured, only the aged thắng unpack wrong but I do not write wrong.

Tut # 1: Armadillo 4.01a (Public Build-Trial Editon)

OPTIONS: Standard Protection, Enable import elimination, Enable strategic code

splicing, patching enable memory protections.

_Load OllyDBG on target:

Address	Hex dump	Disassembly	Comment
004302E3	55	PUSH EBP	
004302E4	8BEC	MOV EBP, ESP	
004302E6	6A FF	PUSH -1	
004302E8	68 200B4500	PUSH tut1.0045AB20	
004302ED	68 200B4300	PUSH tut1.00430020	SE handler installation
004302F2	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
004302F8	50	PUSH EAX	
004302F9	64:8925 0000	MOV DWORD PTR FS:[0], ESP	
00430300	83EC 58	SUB ESP, 58	
00430303	53	PUSH EBX	
00430304	56	PUSH ESI	
00430305	57	PUSH EDI	

_Chon Options in OllyDBG:

☒ Ignore memory access violations in KERNEL32

Ignore (pass to program) following exceptions:

☒ INT3 breaks

☐ Single-step break

☐ Memory access violation

☐ Integer division by 0

☐ Invalid or privileged instruction

☐ All FPU exceptions

☐ Ignore also following custom exceptions or ranges:

0EEDFADE

80000004 (SINGLE STEP)

C0000005 (ACCESS VIOLATION)

C000001D (ILLEGAL INSTRUCTION)

C000001E (INVALID LOCK SEQUENCE)

00000000 (ILLEGAL INSTRUCTION)

Add last exception

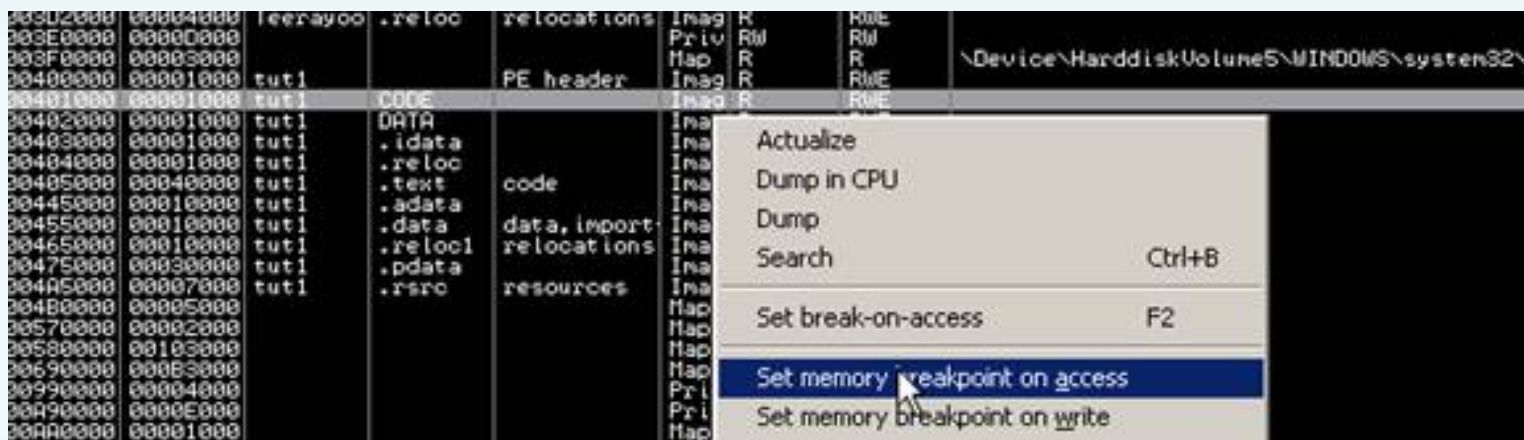
Add range

Delete selection

_Sau Patch the kernel, press Shift + F9 to here 29 times (30 times to jump to dll system, 31 fully run, the trial has informed the press OK to ignore, the regged is similar):

Address	Hex dump	Disassembly	Comment
00CDD265	8900	MOV DWORD PTR DS:[EAX], EAX	
00CDD268	90	NOP	
00CDD269	E9 57010000	JMP 00CDD2C5	
00CDD26E	FF75 EC	PUSH DWORD PTR SS:[EBP-14]	
00CDD271	E8 34F5FFFF	CALL 00CDD27A	
00CDD276	59	POP ECX	
00CDD277	C3	RETN	


_Nhan Alt + M to mem map, select the section CODE set mem brkp access:



_Shift + F9: I OEP:

Address	Hex dump	Disassembly	Comment
00401099	EB 27	JMP SHORT tut1.004010C2	
0040109B	33C0	XOR EAX, EAX	
0040109D	A3 F7204000	MOV DWORD PTR DS:[4020F7], EAX	
004010A2	6A 29	PUSH 29	
004010A4	68 0E204000	PUSH tut1.0040200E	
004010A9	6A 65	PUSH 65	
004010AB	FF75 08	DWORD PTR SS:[EBP+8]	
004010AE	E8 D9010000	CALL tut1.0040128C	
004010B3	A3 F7204000	MOV DWORD PTR DS:[4020F7], EAX	JMP to USER32.GetDlgItemTextA

_Dump Us, stop stop, do not jump. Preview the skin. Alt + M, remove mem breakpoint and see:



00340000	00001000			Priv	RME	RME	
00350000	00010000			Priv	RM	RM	
00360000	00001000			Priv	RM	RM	
00370000	00001000			Priv	RM	RM	
00380000	00001000	Teerayoo	PE header	Imag	R	RME	
00381000	0003E000	Teerayoo	code	Imag	R	RME	
0038F000	0000A000	Teerayoo	data	Imag	R	RME	
003C9000	00001000	Teerayoo	data	Imag	R	RME	
003CA000	00002000	Teerayoo	imports	Imag	R	RME	
003CC000	00004000	Teerayoo	exports	Imag	R	RME	
003D0000	00002000	Teerayoo	resources	Imag	R	RME	
003D2000	00004000	Teerayoo	relocations	Imag	R	RME	
003E0000	00000000			Map	R	RM	
00400000	00001000	tut1		Imag	R	RME	
00401000	00001000	tut1	CODE	Imag	R	RME	
00402000	00001000	tut1	DATA	Imag	R	RME	
00403000	00001000	tut1	.idata	Imag	R	RME	
00404000	00001000	tut1	.reloc	Imag	R	RME	
00405000	00040000	tut1	code	Imag	R	RME	
00445000	00010000	tut1	.adata	Imag	R	RME	
00455000	00010000	tut1	data, import	Imag	R	RME	
00465000	00010000	tut1	relocations	Imag	R	RME	

_Sao Up of coffee, they have set a new mem brkpoint they also see the word PE Header mû. Phone, adding that word to do so belgium hours. Hix, learn the process protect the Dillo, where they have this where I do, they only destroy full table of the PE header after dump completely: D. OK, open a OllyDBG other target tut1.exe load up, press Alt + M, double click on the line:

Memory map, item 28

Address = 00400000

Size = 00001000 (4096.)

Owner tut1 = 00400000 (itself)

Section =

PE header contains =

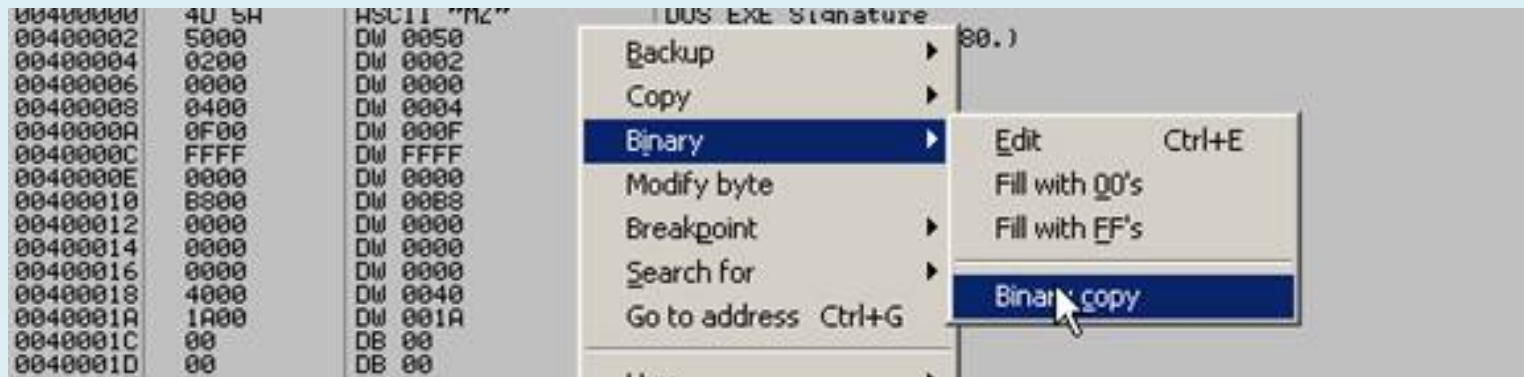
Type = 01001002 IMAG

Access R =

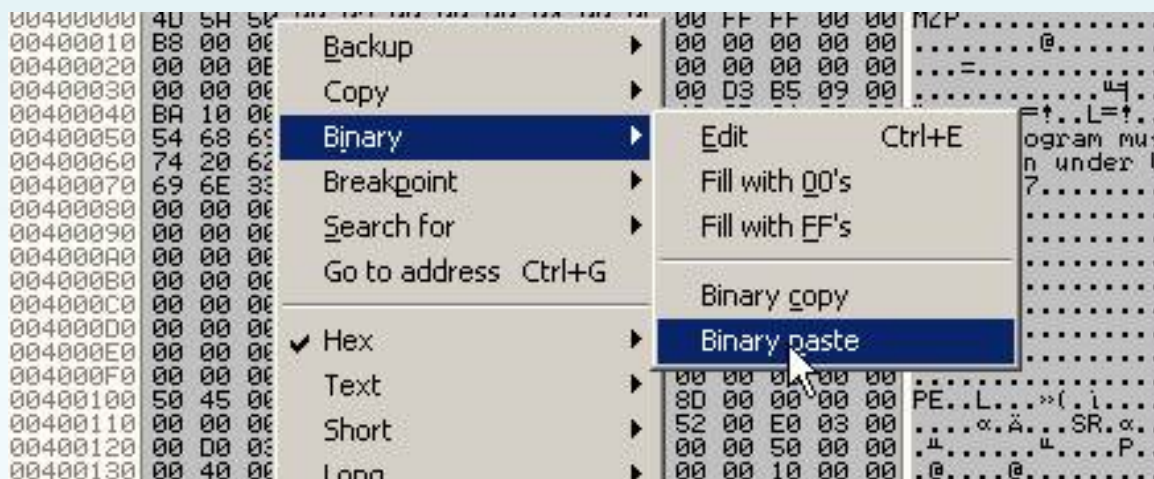
Initial access = RWE

003F0000	00003000			PE header	Map	R	R	\Dev
00400000	00001000	tut1		PE header	Imag	R	RWE	
00401000	00001000	tut1	CODE		Imag	R	RWE	
00402000	00001000	tut1	DATA		Imag	R	RWE	

_Drag Mouse from the beginning to the end (400,000 ... 400FFF), select Copy Binary:



_Tro OllyDBG the original. Also double to 400,000 addresses, and drag the mouse to 400,000 ... 400FFF, select Binary paste:



_Bay Dump hours only, select OllyDump:

OllyDump - tut1.exe

Start Address: Size:

Entry Point: -> Modify:

Base of Code: Base of Data:

☒ Fix Raw Size & Offset of Dump Image

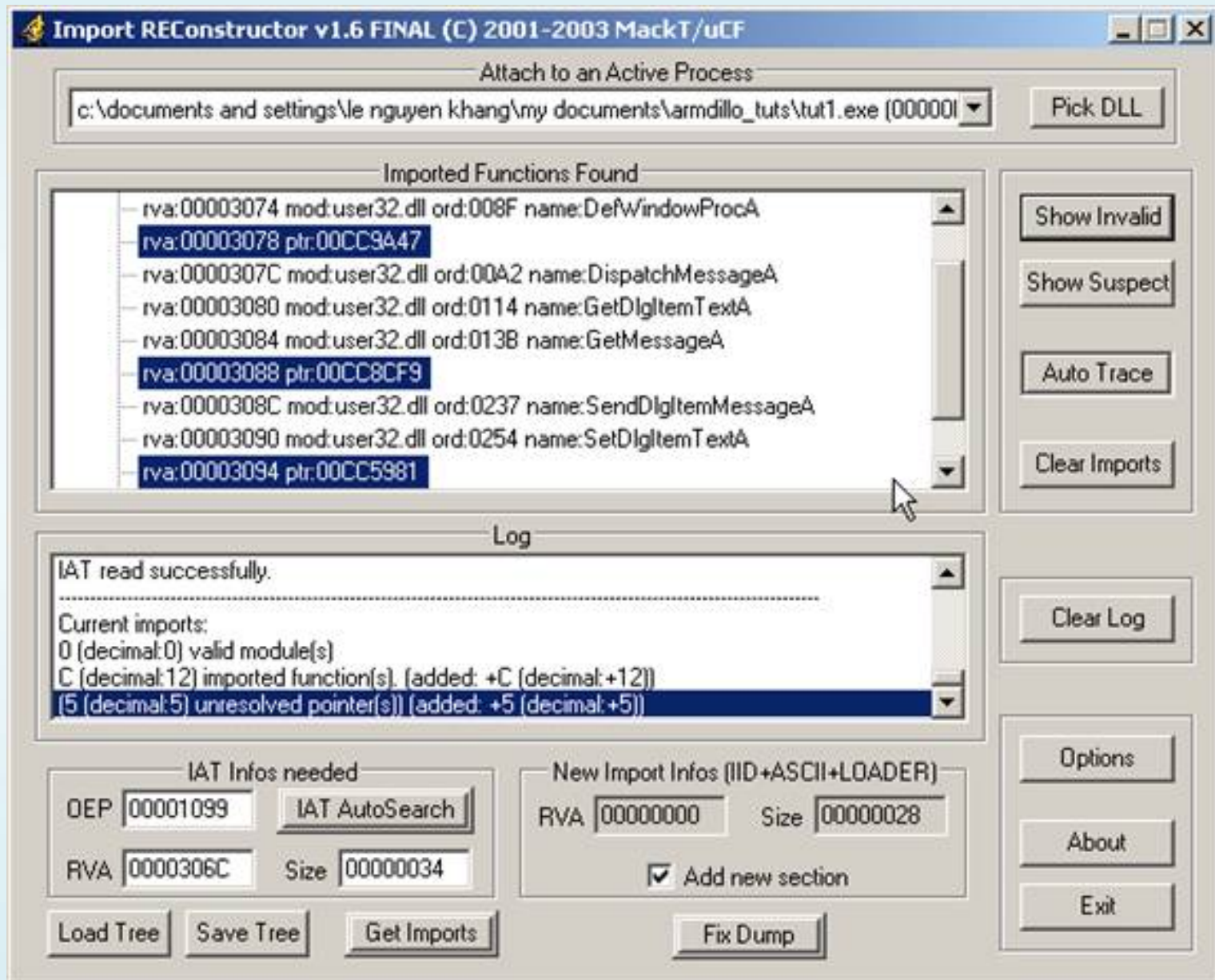
Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristic
CODE	00001000	00001000	00001000	00001000	60000020
DATA	00001000	00002000	00001000	00002000	C0000040
.idata	00001000	00003000	00001000	00003000	C0000040
.reloc	00001000	00004000	00001000	00004000	50000040
.text	00040000	00005000	00040000	00005000	E0000020
.adata	00010000	00045000	00010000	00045000	E0000020
.data	00010000	00055000	00010000	00055000	C0000040

☐ Rebuild Imports

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Fix IAT _Bi hours:



5 for _Co invalid. Cuts do not have records that Thunks bay nhe!

+00003078

+00003088

+00003094

+00003098

+0000309 C

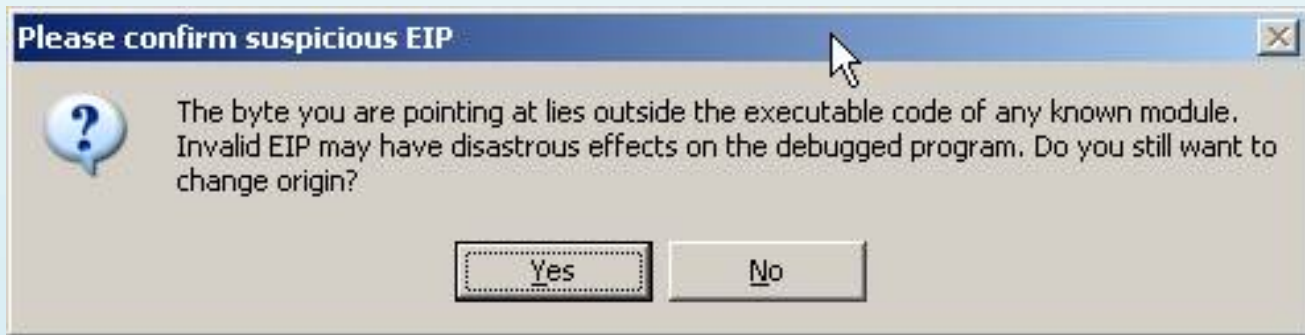
_Tro OllyDBG again, scroll up the screen on the same type, Ctrl + B, enter FF25:

Address	Hex dump	Disassembly	Comment
00401274	- FF25 70304000	JMP NEAR DWORD PTR DS:[403070]	USER32.TranslateMessage
0040127A	- FF25 74304000	JMP NEAR DWORD PTR DS:[403074]	USER32.DefWindowProcA
00401280	- FF25 78304000	JMP NEAR DWORD PTR DS:[403078]	
00401286	- FF25 7C304000	JMP NEAR DWORD PTR DS:[40307C]	USER32.DispatchMessageA
0040128C	- FF25 80304000	JMP NEAR DWORD PTR DS:[403080]	USER32.GetDlgItemTextA
00401292	- FF25 84304000	JMP NEAR DWORD PTR DS:[403084]	USER32.GetMessageA
00401298	- FF25 88304000	JMP NEAR DWORD PTR DS:[403088]	
0040129E	- FF25 8C304000	JMP NEAR DWORD PTR DS:[40308C]	USER32.SendDlgItemMessageA
004012A4	- FF25 90304000	JMP NEAR DWORD PTR DS:[403090]	USER32.SetDlgItemTextA
004012AA	- FF25 94304000	JMP NEAR DWORD PTR DS:[403094]	
004012B0	- FF25 98304000	JMP NEAR DWORD PTR DS:[403098]	
004012B6	- FF25 9C304000	JMP NEAR DWORD PTR DS:[40309C]	

_Ta Beginning JMP NEAR DWORD PTR DS: [403078] (403078-400000 = 3078). R-click, choose:



_Ta To mind the value of **EIP 00401280** tut1.00401280:



_Nhan F7. One is to:

Address	Hex dump	Disassembly	Comment
00CC9A47	55	PUSH EBP	
00CC9A48	8BEC	MOV EBP, ESP	
00CC9A4A	FF75 18	PUSH DWORD PTR SS:[EBP+18]	
00CC9A4D	FF75 14	PUSH DWORD PTR SS:[EBP+14]	
00CC9A50	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
00CC9A53	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]	
00CC9A56	FF75 08	PUSH DWORD PTR SS:[EBP+08]	
00CC9A59	FF15 4024CE00	CALL NEAR DWORD PTR DS:[CE2440]	USER32.ShowDialogParamA
00CC9A5F	5D	POP EBP	
00CC9A60	C2 1400	RETN 14	

_Nhan See from start to finish jaw function, no other function. That function is the need to find 00003078 user32.dll 009F DialogBoxParamA.

_Tro Address to 401,280, the second search function as similar to the one here:
00401298 - FF25 88304000 JMP NEAR DWORD PTR DS: [403088]

Address	Hex dump	Disassembly	Comment
00CC8CF9	55	PUSH EBP	
00CC8CFA	8BEC	MOV EBP, ESP	
00CC8CFC	51	PUSH ECX	
00CC8CFD	53	PUSH EBX	
00CC8CFE	56	PUSH ESI	
00CC8CFF	57	PUSH EDI	
00CC8D00	60	PUSHAD	
00CC8D01	8B15 184ACF00	MOV EDX, DWORD PTR DS:[CF4A18]	kernel32.77E7A237
00CC8D07	83C2 64	ADD EDX, 64	
00CC8D0A	FFD2	CALL NEAR EDX	USER32.77D66412
00CC8D0C	8B15 0449CF00	MOV EDX, DWORD PTR DS:[CF49D4]	
00CC8D12	83C2 64	ADD EDX, 64	
00CC8D15	B9 05000000	MOV ECX, 5	
00CC8D1A	803A CC	CMP BYTE PTR DS:[EDX], 0CC	
00CC8D1D	74 10	JE SHORT 00CC8D2F	
00CC8D1F	52 59	LOAD SHORT 00CC8D10	

_Phew, Do not see anything, do not nản, you press F8 a few times, until the line:

00CC8D12 83C2 64 ADD EDX, 64

00CC8D15 B9 05000000 MOV ECX, 5

_Dom The Register window to see what has EDX is strange:

```
Registers (FPU)
EAX 0082F871
ECX 00401099 tut1.00401099
EDX 77D66476 USER32.MessageBoxA
EBX 00000000
ESP 0012D454
EBP 0012D484
ESI 0045BA30 tut1.0045BA30
EDI 00000001
EIP 00CC8D15
```

_Yeah, Here is the address of MessageBoxA function. Perhaps you ask, it is the function GetTickCount I do not choose, he he, including subliminal say anything, I try to select and then, crash when run.

_Tiep Continue to **00003094**, the hand kô find any, can Cut Thunk.

00003098 _Tiep to make similar, we find it is located at:

004012AA - FF25 98304000 JMP NEAR DWORD PTR DS: [403098]

_Set New origin, F7 at the following code:

Address	Hex dump	Disassembly	Comment
00CC8C70	55	PUSH EBP	
00CC8C71	8BEC	MOV EBP, ESP	
00CC8C73	51	PUSH ECX	
00CC8C74	53	PUSH EBX	
00CC8C75	56	PUSH ESI	
00CC8C76	57	PUSH EDI	
00CC8C77	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
00CC8C7A	E8 1ACBFFFF	CALL 00CC5799	
00CC8C7F	85C0	TEST EAX, EAX	

_Y On the chang, press F8 trace gradually:

00CC8C99 83C2 64 ADD EDX, 64

00CC8C9C B9 05000000 MOV ECX, 5

Cmp CC 00CC8CA1 803A BYTE PTR DS: [EDX], 0CC

_Tuong Automatically find the function: GetModuleHandleA

```
Registers (FPU)
EAX 0080DEAA
ECX 00400000 ASCII "MZP"
EDX 77E7AD86 kernel32.GetModuleHandleA
EBX 00000000
ESP 0012D454
EBP 0012D484
ESI 0045BA30 tut1.0045BA30
EDI 00000001
EIP 00CC8C9C
C 0 ES 0023 32bit 0(FFFFFFFF)
P 0 CS 001B 32bit 0(FFFFFFFF)
```

_Bay Hours last longer function, dom in IAT I see Thieu function exit, if the phang confident now that this function. But it does not find:

004012B0 - FF25 9C304000 JMP NEAR DWORD PTR DS: [40309C]

_Trace Last few lines of code we find the function:

The screenshot displays the OllyDbg interface with the following components:

- Assembly Window:** Shows instructions from address 00CC6692 to 00CC66F8. Key instructions include:
 - 00CC6692: 55 PUSH EBP
 - 00CC6693: 8BEC MOV EBP, ESP
 - 00CC6695: 6A FF PUSH -1
 - 00CC6697: 68 1827CE00 PUSH 0CE2718
 - 00CC669C: 68 4017CE00 PUSH 0CE1740
 - 00CC66A1: 64:01 00000000 MOV EAX, DWORD PTR FS:[0]
 - 00CC66A7: 50 PUSH EAX
 - 00CC66A8: 64:01 00000000 MOV EAX, DWORD PTR FS:[0], ESP
 - 00CC66AF: 51 PUSH ECX
 - 00CC66B0: 51 PUSH ECX
 - 00CC66B1: 53 PUSH EBX
 - 00CC66B2: 56 PUSH ESI
 - 00CC66B3: 57 PUSH EDI
 - 00CC66B4: 8965 E8 MOV DWORD PTR SS:[EBP-18], ESP
 - 00CC66B7: 8965 FC 00 AND DWORD PTR SS:[EBP-4], 0
 - 00CC66B8: 6A 02 PUSH 2
 - 00CC66B9: FF15 A421CE00 CALL NEAR DWORD PTR DS:[CE21A4]
 - 00CC66C3: C605 38F5CE00 MOV BYTE PTR DS:[CEFS38], 1
 - 00CC66C7: E8 BEC30000 CALL 00CCD2A2
 - 00CC66D4: EB 07 JMP SHORT 00CC66D0
 - 00CC66D6: 6A 01 PUSH 1
 - 00CC66D8: 58 POP EAX
 - 00CC66D9: C3 RETN
 - 00CC66DA: 8B65 E8 MOV ESP, DWORD PTR SS:[EBP-18]
 - 00CC66DB: 8340 FC FF OR DWORD PTR SS:[EBP-4], 0
 - 00CC66E1: C745 FC 010000 MOV DWORD PTR SS:[EBP-4], 1
 - 00CC66E8: 60 PUSHAD
 - 00CC66E9: 8B15 E849CF00 MOV EDX, DWORD PTR DS:[CF49E8]
 - 00CC66EF: 89C2 64 ADD EDX, 64
 - 00CC66F2: FF75 00 JMP DWORD PTR SS:[EBP+0]
 - 00CC66F5: FF02 JMP NEAR EDI
 - 00CC66F7: 61 POPAD
 - 00CC66F8: EB 07 JMP SHORT 00CC66F0
 - 00CC66FA: 6A 01 PUSH 1
- Registers Window:** Shows the state of registers. EAX is 00000001, ECX is 00120474, EDI is 77E79899 (circled in red). Other registers like ESI, EDI, and EIP are also visible.
- Stack Window:** Shows the current stack frame. The stack pointer (ESP) is at 00120440. The stack contains data from the function call.

_Fix Dump, all worked. Sucessful unpack.

Tut # 2: Armadillo 4:05 - Registry Mechanic 5.0.0.142 Standard Protection

The meals I post an article on how to unpack only REA with tools and manual. There is a medical AM I try to unpack manually review. OK, today we will unpack this month.

_Chinh In OllyDBG Options:

Commands	Disasm	CPU	Registers	Stack	Analysis 1	Analysis 2	Analysis 3
Security	Debug	Events	Exceptions	Trace	SFX	Strings	Addresses

☒ Ignore memory access violations in KERNEL32

Ignore (pass to program) following exceptions:

☒ INT3 breaks

☐ Single-step break

☐ Memory access violation

☐ Integer division by 0

☐ Invalid or privileged instruction

☐ All FPU exceptions

☒ Ignore also following custom exceptions or ranges:

0EEDFADE	▲
80000004 (SINGLE STEP)	▶
C0000005 (ACCESS VIOLATION)	▶
C000001D (ILLEGAL INSTRUCTION)	▶
C000001E (INVALID LOCK SEQUENCE)	▼
C0000000 (PRIVILEGED INSTRUCTION)	▼

Add last exception

Add range

Delete selection

_Load OllyDBG on target:

Address	Hex dump	Disassembly	Comment
009434C3	55	PUSH EBP	
009434C4	8BEC	MOV EBP, ESP	
009434C6	6A FF	PUSH -1	
009434C8	68 20DB9600	PUSH RegMech.0096DB20	
009434CD	68 00329400	PUSH RegMech.00943200	
009434D2	64:A1 000000	MOV EAX, DWORD PTR FS:[0]	SE handler installation
009434D8	50	PUSH EAX	
009434D9	64:8925 0000	MOV DWORD PTR FS:[0], ESP	
009434E0	83EC 58	SUB ESP, 58	
009434E3	53	PUSH EBX	
009434E4	56	PUSH ESI	

Atl + F1 _Go, BP CreateThread enter:

bp CreateThread

_Nhan Shift + F9:

Address	Hex dump	Disassembly	Comment
77E7BE53	55	PUSH EBP	
77E7BE54	8BEC	MOV EBP, ESP	
77E7BE56	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]	
77E7BE59	FF75 18	PUSH DWORD PTR SS:[EBP+18]	
77E7BE5C	FF75 14	PUSH DWORD PTR SS:[EBP+14]	
77E7BE5F	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
77E7BE62	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]	
77E7BE65	FF75 08	PUSH DWORD PTR SS:[EBP+08]	
77E7BE68	6A FF	PUSH -1	
77E7BE6A	E8 30FEFFFF	CALL kernel32.CreateRemoteThread	
77E7BE6F	5D	POP EBP	
77E7BE70	C2 1800	RETN 18	

_Nhan Ctrl + F9:

77E7BE6F	5D	POP	EBP	
77E7BE70	C2 1800	RETN	18	
77E7BE73	8BC2	MOV	EAX, EDX	
77E7BE75	E9 6DFCFFFF	JMP	kernel32.77E7BAE7	
77E7BE78	55	PUSH	EBP	

_Nhan F8:

Address	Hex dump	Disassembly	Comment
01241102	5E	POP	ESI
01241103	C9	LEAVE	
01241104	C3	RETN	
01241105	55	PUSH	EBP
01241106	8BEC	MOV	EBP, ESP
01241108	81EC 28010000	SUB	ESP, 128
0124110E	56	PUSH	ESI

_Lai Press Ctrl + F9:

01241103	C9	LEAVE	
01241104	C3	RETN	
01241105	55	PUSH	EBP

_Nhan To F8:

Address	Hex dump	Disassembly	Comment
0125BE07	A1 D81E2701	MOV	EAX, DWORD PTR DS:[1271ED8]
0125BE0C	59	POP	ECX
0125BE0D	8B48 68	MOV	ECX, DWORD PTR DS:[EAX+68]
0125BE0E	3348 64	XOR	ECX, DWORD PTR DS:[EAX+64]
0125BE0F	3348 5C	XOR	ECX, DWORD PTR DS:[EAX+5C]
0125BE10	F6C1 40	TEST	CL, 40
0125BE11	75 08	JNZ	SHORT 0125BEE3
0125BE12	6A 01	PUSH	1
0125BE13	E8 15BEF0FF	CALL	01237CF7
0125BE14	59	POP	ECX
0125BE15	53	PUSH	EBX
0125BE16	C705 E0792601	MOV	DWORD PTR DS:[12679E0], 12687A4
0125BE17	F8 22FAE0FF	CALL	0123A915

_Cuon Screen down hairbreadth decency Call for EDI:

0125BF4B	53	PUSH	EBX
0125BF4C	E8 33E7FEFF	CALL	0124A684
0125BF4D	50	PUSH	EAX
0125BF4E	A1 D81E2701	MOV	EAX, DWORD PTR DS:[1271ED8]
0125BF4F	8B48 68	MOV	ECX, DWORD PTR DS:[EAX+68]
0125BF50	3348 34	XOR	ECX, DWORD PTR DS:[EAX+34]
0125BF51	3348 20	XOR	ECX, DWORD PTR DS:[EAX+20]
0125BF52	2BF9	SUB	EDI, ECX
0125BF53	FFD7	CALL	NEAR EDI
0125BF54	8945 FC	MOV	DWORD PTR SS:[EBP-4], EAX
0125BF55	8B45 FC	MOV	EAX, DWORD PTR SS:[EBP-4]
0125BF56	5F	POP	EDI
0125BF57	5E	POP	ESI
0125BF58	5B	POP	EBX
0125BF59	C9	LEAVE	
0125BF5A	C3	RETN	

F2 _Nhan set breakpoint at **0125BF62 FFD7 CALL NEAR EDI**, press F9 to run, will Olly ice in 0125BF62, press F7 to our OEP, yeah!

Address	Hex dump	Disassembly	Comment
00412990	68 D8394100	PUSH RegMech.004139D8	
00412995	E8 F0FFFFFF	CALL RegMech.0041298A	
0041299A	0000	ADD BYTE PTR DS:[EAX], AL	
0041299C	68 00000030	PUSH 30000000	
004129A1	0000	ADD BYTE PTR DS:[EAX], AL	
004129A3	0060 00	ADD BYTE PTR DS:[EAX], AH	
004129A6	0000	ADD BYTE PTR DS:[EAX], AL	
004129A8	48	DEC EAX	
004129A9	0000	ADD BYTE PTR DS:[EAX], AL	
004129AB	00D3	ADD BL, DL	
004129AD	E4 1E	IN AL, 1E	I/O command
004129AF	94	XCHG EAX, ESP	
004129B0	D263 20	SHL BYTE PTR DS:[EBX+20], CL	
004129B3	4F	DEC EDI	
004129B4	8012 EB	ADC BYTE PTR DS:[EDX], 0EB	
004129B7	25 655B46E2	AND EAX, E2465B65	
004129BC	0000	ADD BYTE PTR DS:[EAX], AL	
004129BE	0000	ADD BYTE PTR DS:[EAX], AL	
004129C0	0000	ADD BYTE PTR DS:[EAX], AL	
004129C2	0100	ADD DWORD PTR DS:[EAX], EAX	
004129C4	0000	ADD BYTE PTR DS:[EAX], AL	
004129C6	0000	ADD BYTE PTR DS:[EAX], AL	
004129C8	0300	ADD EAX, DWORD PTR DS:[EAX]	

_Tro As tut on, open a Olly, target load, Binary copy PE header from the 400,000 ... 400FFF and paste. Then dump.

OllyDump - RegMech.exe

Start Address: 400000

Size: 73B000

Dump

Entry Point: 5434C3

-> Modify: 12990

Get EIP as DEP

Cancel

Base of Code: 518000

Base of Data: 568000

☒ Fix Raw Size & Offset of Dump Image

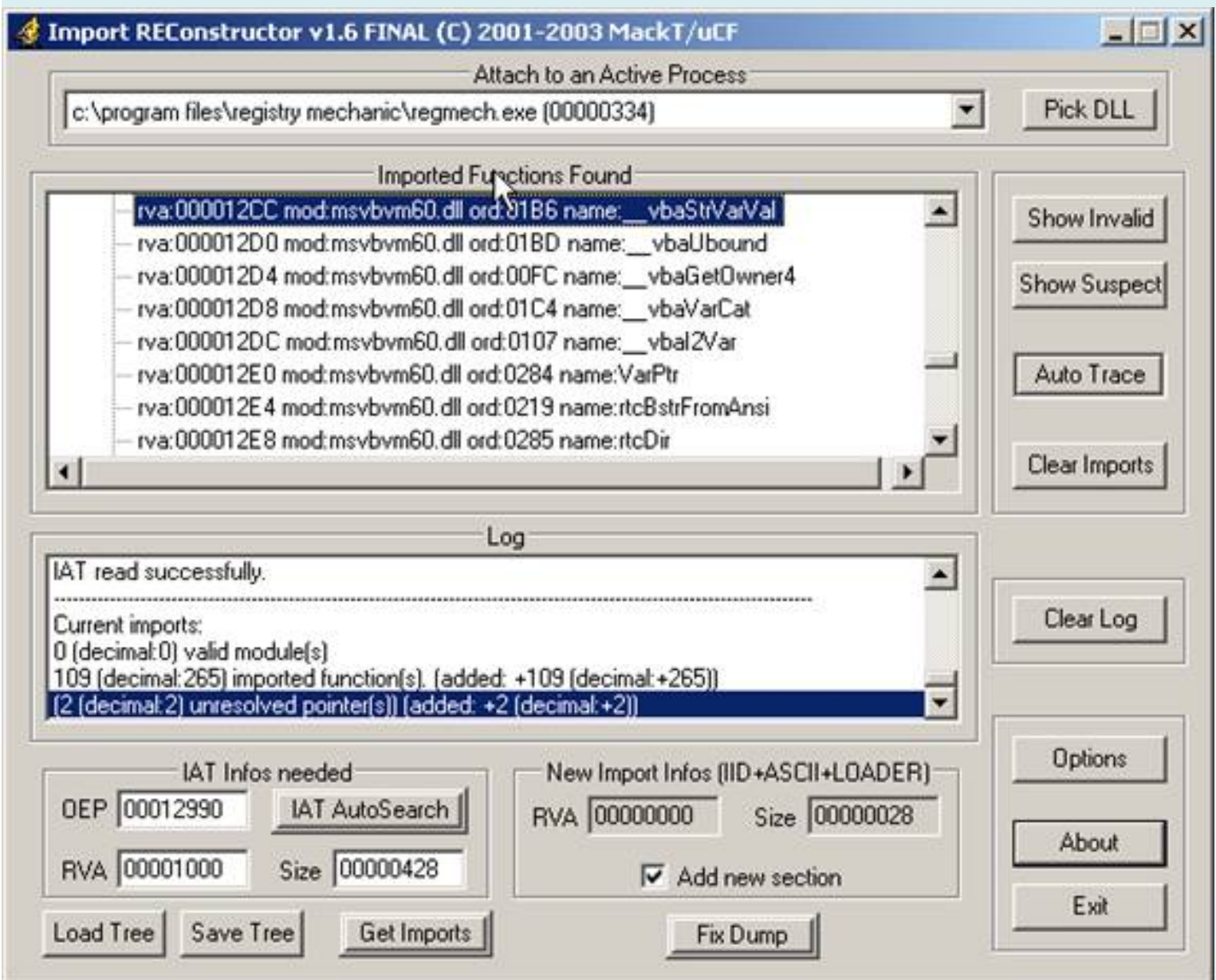
Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
.text	00508BEC	00001000	00508BEC	00001000	60000020
.data	0000DB98	0050A000	0000DB98	0050A000	C0000040
.text1	00040000	00518000	00040000	00518000	E0000020
.adata	00010000	00558000	00010000	00558000	E0000020
.data1	00020000	00568000	00020000	00568000	C0000040
.pdata	00120000	00588000	00120000	00588000	C0000040
.rsrc	00093000	006A8000	00093000	006A8000	40000040

☐ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

_Fix IAT:



_hai invalid children:

JMP NEAR DWORD PTR DS: [401048]

JMP NEAR DWORD PTR DS: [401420]

_Tu OEP, as opposed to end up, type Ctrl + F and enter JMP NEAR DWORD PTR DS: [401048]. We will go to address:

004125B8 - FF25 48104000 JMP NEAR DWORD PTR DS: [401048]

_Tai Ago, R-Click, select New from origine here. Press F7, to this:

Address	Hex dump	Disassembly	Comment
0123F261	55	PUSH EBP	
0123F262	8BEC	MOV EBP, ESP	
0123F264	6A FF	PUSH -1	
0123F266	68 D8262601	PUSH 12626D8	
0123F268	68 500D2601	PUSH 1260D50	JMP to MSUCRT._except_handl
0123F270	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
0123F276	50	PUSH EAX	
0123F277	64:8925 00000001	MOV DWORD PTR FS:[0], ESP	
0123F27E	83EC 10	SUB ESP, 10	
0123F281	53	PUSH EBX	
0123F282	56	PUSH ESI	
0123F283	57	PUSH EDI	
0123F284	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
0123F287	33F6	XOR ESI, ESI	
0123F289	68 64742601	PUSH 1267464	ASCII "MSUBUM60.DLL"
0123F28E	FF15 E8202601	CALL NEAR DWORD PTR DS:[12620E8]	kernel32.GetModuleHandleA
0123F294	85C0	TEST EAX, EAX	
0123F296	74 0E	JE SHORT 0123F2A6	
0123F298	68 58742601	PUSH 1267458	ASCII "__vbaEnd"
0123F29D	50	PUSH EAX	
0123F29E	FF15 EC202601	CALL NEAR DWORD PTR DS:[12620EC]	kernel32.GetProcAddress
0123F2A4	8BF0	MOV ESI, EAX	
0123F2A6	6A 02	PUSH 2	
0123F2A8	FF15 90212601	CALL NEAR DWORD PTR DS:[1262190]	kernel32.SetErrorMode
0123F2AE	8365 FC 00	AND DWORD PTR SS:[EBP-4], 0	
0123F2B2	85F6	TEST ESI, ESI	
0123F2B4	74 16	JE SHORT 0123F2CC	
0123F2B6	FFD6	CALL NEAR ESI	
0123F2B8	EB 12	JMP SHORT 0123F2CC	
0123F2BA	6A 01	PUSH 1	
0123F2BC	58	POP EAX	
0123F2BD	C3	RETN	
0123F2BE	8B65 E8	MOV ESP, DWORD PTR SS:[EBP-18]	
0123F2C1	68 C8000000	PUSH 0C8000000	

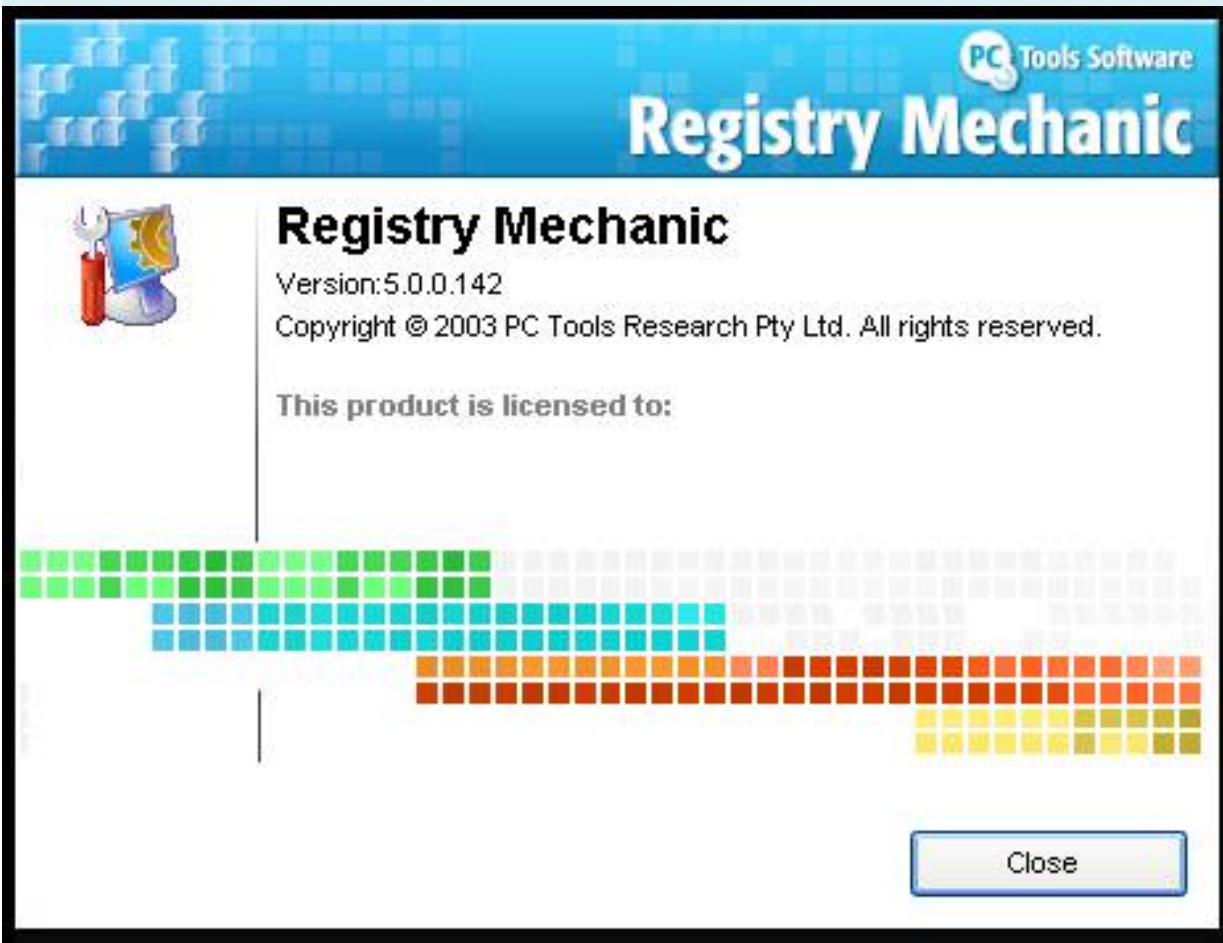
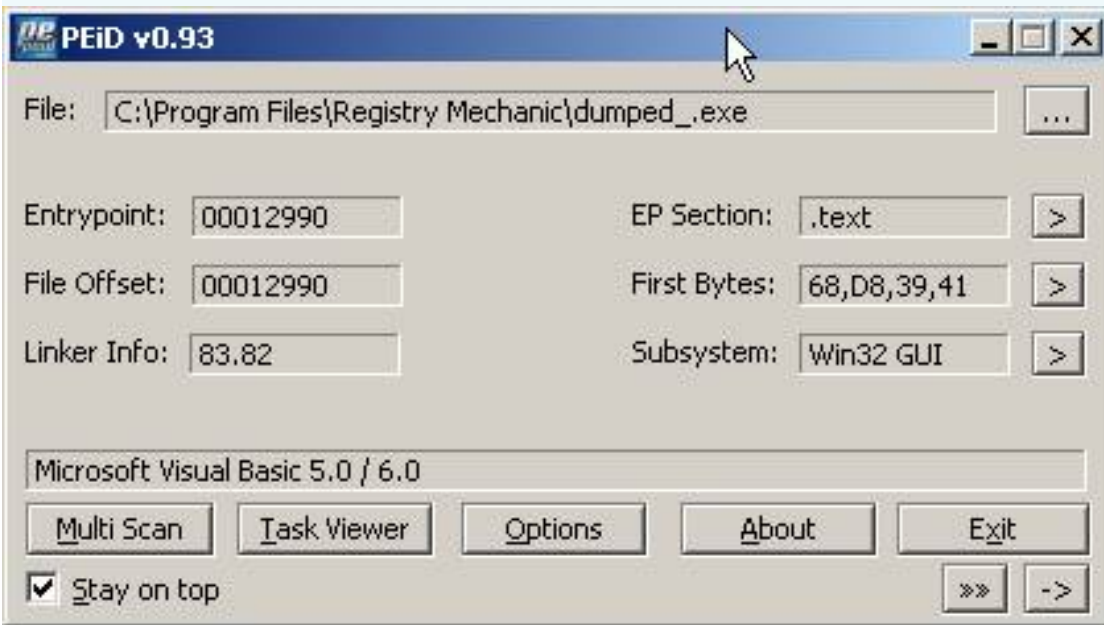
_Trace Down since we see from the beginning and end of the function, the function __vbaEnd (equivalent ExitProcess. Because the code with VB should we choose this function.

_Ham JMP NEAR DWORD PTR DS: [401420] should not exist Cut Thunk.

_Fix Done:



_Detect:



The Summer, after many nights of thinking, I think of new ways to call is accurate manual unpack protect the target with standard protection. Copyright @ hacnho, khoa khoa. The doctors read this tut can practice immediately. But the doctors read by tut ARTEAM also unpack a soft well with Registry Mechanic but even I, try to read that much arm dek also understand many belgium J. Attach target, dumped.exe, and dumped_.exe tree for medical practice. Ah forget a note that when you open a copy to OllyDBG PE Header, complete copy it

to memory, not only is not ImpREC Attach the die.

Tut # 3: UnPackMe_Armadillo4.10.b ***Standard Protection***



This _Cai similar on the tut, it is important which method you choose, be slow and patience to find the OEP and fix IAT (about 24 functions). If successful you will be here:

- [CPU - main thread, module UnPackMe]

File View Debug Plugins Options Window Help

LEMTWHC/KBR...S

Address	Hex dump	Disassembly	Comment
00427180	66	POP ESP	
00427181	88EC	MOV EBP, ESP	
00427183	6A FF	PUSH -1	
00427185	68 50E4500	PUSH UnPackMe.00450E60	
0042718A	60 C0924200	PUSH UnPackMe.004272C8	
0042718F	64A1 00000000	MOV EAX, DWORD PTR FS:[0]	
004271C5	50	POP EAX	
004271C6	64:8925 000000	MOV DWORD PTR FS:[0], ESP	
004271C0	83C4 A8	ADD ESP, -A8	
004271D0	53	PUSH EBX	
004271D1	56	PUSH ESI	
004271D2	57	PUSH EDI	
004271D3	8965 E8	MOV DWORD PTR SS:[EBP-10], ESP	
004271D6	FF15 DC0A4600	NEAR DWORD PTR DS:[468A0C]	
004271DC	3302	XOR EDX, EDX	
004271DE	8D04	MOV DL, AH	
004271E8	8915 34E64500	MOV DWORD PTR DS:[45E634], EDX	
004271E6	8BC8	MOV ECX, EAX	
004271E8	81E1 FF000000	AND ECX, 0FF	
004271EE	8900 30E64500	MOV DWORD PTR DS:[45E630], ECX	
004271F4	C1E1 08	SHL ECX, 8	
004271F7	8BCA	ADD ECX, EDX	
004271F9	8900 2CE64500	MOV DWORD PTR DS:[45E62C], ECX	
004271FF	C1E8 10	SHR EAX, 10	
00427202	A3 28E64500	MOV DWORD PTR DS:[45E628], EAX	
00427207	E8 94210000	CALL UnPackMe.004272A0	
0042720C	85C8	TEST EAX, EAX	
0042720E	75 0A	JNZ SHORT UnPackMe.0042721A	
00427210	6A 1C	PUSH 1C	
00427212	E8 49010000	CALL UnPackMe.00427260	
00427217	83C4 04	ADD ESP, 4	
0042721A	E8 012F0000	CALL UnPackMe.0042A1F0	
0042721F	85C8	TEST EAX, EAX	
00427221	75 0A	JNZ SHORT UnPackMe.0042722D	

EBP=00120F14

Address	Hex dump	ASCII	Address	Value	Comment
004C3000	07 36 C7 77 5C 85 C7 77	6E 4E C7 77 80 70 C7 77	00120794	00000000	RETURN to UnPackMe.00400000
004C3010	6C 58 C7 77 5F 3A C7 77	6E 3C C7 77 98 1B C7 77	00120798	00400000	UnPackMe.00400000
004C3020	E0 20 C7 77 1A 5C C7 77	00 00 00 00 1C 39 E7 77	0012079C	00000000	
004C3030	24 04 E7 77 9E 56 E7 77	9B A2 E7 77 49 A9 E7 77	001207A0	00141F1B	
004C3040	00 5B E7 77 63 DF E6 77	63 FA E6 77 86 A0 E7 77	001207A4	0000000A	
004C3050	D9 57 E7 77 60 43 E9 77	C3 FA E6 77 80 6C E6 77	001207A8	0012FF2C	
004C3060	C1 04 EC 77 73 D1 E7 77	9E 16 E6 77 72 AC E7 77	001207AC	00000000	
004C3070	32 83 E7 77 02 15 F5 77	61 09 E7 77 0C 15 F5 77	001207B0	7FFDF000	
004C3080	50 26 E7 77 98 58 E7 77	BC 1B E6 77 58 E3 E7 77	001207B4	0048B089	RETURN to UnPackMe.0048B089 from 0
004C3090	7E 17 E6 77 68 90 E6 77	4C A4 E7 77 12 AC E7 77	001207B8	004C9E08	UnPackMe.004C9E08

Command: []

Memory breakpoint when executing [00427180]

Paused

Tut # 4: UnPackMe_Armadillo4.20.b

Standard Protection



_De Advanced skills, you unpack entirely in tut4 target.

III. A result:

That's all part of a standard protection. Hopefully, through this tut you can unpack to unpack and me being soft on the NET release. The form of this soft pack bags listed here:

Ace Utilities v2.5.0.4016 (Armadillo Standard)

<http://www.acelogix.com>



(showing the game. he he)

Advanced X Video Converter v3.9.32 (Armadillo Standard + elimination Import)

<http://www.aoamedia.com>

AoA DVD Ripper v3.75 (Armadillo Standard + elimination Import)

<http://www.aoamedia.com>

AVI / MPEG / RM / WMV joiner v4.81 (Armadillo Standard Code Splicing + + Import elimination)

<http://www.boilsoft.com>

BearShare Pro v4.7.1.1 (Armadillo Standard)

<http://www.bearshare.com>

Coding Workshop Ringtone Converter v5.2.4 (Armadillo Standard)

<http://www.codingworkshop.com>

Coding Workshop Polyphonic Wizard v4.0.3 (Armadillo Standard)

<http://www.codingworkshop.com>

File recover 5.0.1.15

<http://www.pctools.com>

HyperCam v2.11.00 (Armadillo Standard + elimination Import)

<http://www.hyperionics.com>

HyperSnap-DX v5.62.05 (Armadillo Standard + elimination Import)

<http://www.hyperionics.com>

Privacy Guardian 4.0.0.11

<http://www.pctools.com>

Registry Mechanic 5.0.0.14

<http://www.pctools.com>

Spyware Doctor 3.2.1

<http://www.pctools.com>

Spam Monitor 2.5

<http://www.pctools.com>

_Mot Tut simple with excess image. He he, sure of your target down on hearing xiu always tired. Wrote almost finished and then Serie # 2, but the ability to target, only for you to read this and then finished tut I write more. Lost 3h for the tut this hix. People write a tut for the cracking Nero you. Sure week after new bags up to the Serie # 2. Her only child that nhe. Belgium reported.

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini, hoadongnoi, hosiminh, Nilrem, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, anh_surprised ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RiF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx to authors of OllyDBG.

To be continued ...

Written by **hacnho** (tutorial date: Saigon 21/08/2005)

Armadillo collect sand-stone

Bug Fix in Series # 1!

QUOTE:

Re: Tuts: Armadillo collect sand-stone

Bó hands and then some .. ői aged children to help.

-I have tried to try multiple times to target 3 (or 4) but not how to find out what function fully as he tlandn.

Here they give 2 in the Invalid ImportREC as examples:

--- 1:

rva: 00060ADC ptr: 00C2A12D

Search I-olly in order:

"CALL DWORD PTR DS: [460ADC]"

-I here:

Code:

00418E4A FF15 DC0A4600 CALL DWORD PTR DS: [460ADC]

Then New Origin here, then to the F7 Trace out:

Code:

00C2A12D A1 D06CC500 MOV EAX, DWORD PTR DS: [C56CD0] 00C2A132 C3 RETN

-So here it RETN tia then, I do find that nao.The function in file IAT.txt by an tlandn found that:

Code:

1 00060ADC kernel32.dll 01C8 GetVersion

--- 2:

rva: 00060B40 ptr: 00C2579D

Ollly search in order:

"CALL DWORD PTR DS: [460B40]"

I here:

Code:

0040962C FF15 400B4600 CALL DWORD PTR DS: [460B40]

Then Trace F7 to go through a loop, then through:

Code:

00C257A8 8B35 5C32C400 MOV ESI, DWORD PTR DS: [C4325C]; ntdll.RtlEnterCriticalSection

then back to:

Code:

00C2593C 83C6 0C ADD ESI, 0C

at this value is to write:

Quote:

EAX 77DD0000 ADVAPI32.77DD0000

EDX F0C10000

Then try to buy Trace F8 to make here:

Code:

00C25A28 FFD7 CALL EDI; ntdll.RtlLeaveCriticalSection

Trace-back and then again to Terminate always.

The file in which IAT.txt to shareholders:

Code:

1 00060B40 kernel32.dll GetProcAddress 018A

-So he asked to do any wrong place? The more he can use any Tool to find the function or not based solely on experience, seeing the lack phac vao.Ma not the only place that Invalid, more seats also can not fix as the IAT's tlandn look forward to ... the answer soon, not sure if the networks have lost, then they dippy

Radio-always the Newbie:

Quote:

If the Search command (Ctrl-F):

Code:

JMP NEAR DWORD PTR DS: [XXXXXX]

Search or binary with String (Ctrl-B):

Code:

FF 25

but still much less Fix Invalid place of the scroll to the top of the Search and the command:

Code:

CALL DWORD PTR DS: [XXXXXX]

Search or Binary with chains:

Code:

FF 15

will find the place to Fix Invalid.

-Would the address is in Invalid ImportREC (Olly in memory plus 400,000) and Search by the corresponding order to avoid mistake.

-In the Search, so check with your Plugin Bookmark Olly.Cach do the following:

-When I was OEP (dump is completed, and prepare fix IAT), we will roll up on the line correct? Now Click to a right line, select Bookmarks -> Insert bookmark 0 to save the position in Bookmark only 0.

-Then start Search, for example with the command:

Code:

CALL DWORD PTR DS: [XXXXXX]

-The command that is, Click to a more Bookmark -> Insert bookmark 1.

-Then, hours New Origin, F7 to Trace, the function to find the appropriate back immediately by Click right, Bookmarks -> Go to bookmark 1 or 0 (depending how your Search).

-To remove the position at that store, Click right, Bookmarks -> Delete bookmark 0 or 1

-This helps you save time when Trace to the depth that want to turn out right.

He trickyboy this nhận real, post messages to the village was also asked directions again; ;).
I have is a mistake in one. This bug is fixed. Read it for fun.

After reading the two of you I sure do not need to read this section but for those like the
writing on bags. Very simple.

Tut # 3: UnPackMe_Armadillo4.10.b ***Standard Protection***



Olly _Load on target, press Alt + M placed on Breakpoint Memory Access. Press Shift + F9, jump to OEP.

004271B0	55	PUSH EBP
004271B1	8BEC	MOV EBP,ESP
004271B3	6A FF	PUSH -1
004271B5	68 600E4500	PUSH UnPackMe.00450E60
004271B8	68 C8924200	PUSH UnPackMe.004292C8
004271BF	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
004271C5	50	PUSH EAX
004271C6	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
004271CD	83C4 A8	ADD ESP,-58
004271D0	53	PUSH EBX
004271D1	56	PUSH ESI
004271D2	57	PUSH EDI
004271D3	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP
004271D6	FF15 DC0A4600	CALL DWORD PTR DS:[460ADC]
004271DC	33D2	XOR EDX,EDX
004271DE	8AD4	MOV DL,AH
004271E0	8915 34E64500	MOV DWORD PTR DS:[45E634],EDX
004271E6	8BC8	MOV ECX,EAX
004271E8	81E1 FF000000	AND ECX,0FF
004271EE	890D 30E64500	MOV DWORD PTR DS:[45E630],ECX
004271F4	C1E1 08	SHL ECX,8
004271F7	03CA	ADD ECX,EDX
004271F9	890D 2CE64500	MOV DWORD PTR DS:[45E62C],ECX
004271FF	C1E8 10	SHR EAX,10
00427202	A3 28E64500	MOV DWORD PTR DS:[45E628],EAX
00427207	E8 94210000	CALL UnPackMe.004293A0
0042720C	85C0	TEST EAX,EAX
0042720E	75 0A	JNZ SHORT UnPackMe.0042721A
00427210	6A 1C	PUSH 1C
00427212	E8 49010000	CALL UnPackMe.00427360
00427217	83C4 04	ADD ESP,4
0042721A	E8 D12F0000	CALL UnPackMe.0042A1F0
0042721F	05C0	TEST EBX,EBX

_Ghi Remember DC0A4600 function 004271D6 FF15 CALL DWORD PTR DS: [460ADC]

_ Ctrl + F2. Click Memory dump Windows, Ctrl + G, enter 460ADC, selected first 4 bytes, placed on HardwareBreakpoint write type DWord. Press Shift + F9:

77C42F43	F3: A5	REP MOVSD WORD PTR ES:[EDI],DWORD PTR DS:[EDI]	
77C42F45	FF2495 5830C477	JMP DWORD PTR DS:[EDX*4+77C43058]	
77C42F4C	88C7	MOV EAX,EDI	
77C42F4E	BA 03000000	MOV EDX,3	
77C42F53	83E9 04	SUB ECX,4	
77C42F56	72 0C	JB SHORT nsvert.77C42F64	
77C42F58	83E0 03	AND EAX,3	
77C42F5B	03C8	ADD ECX,EAX	
77C42F5D	FF2485 702EC477	JMP DWORD PTR DS:[EAX*4+77C42F70]	
77C42F64	FF248D 5830C477	JMP DWORD PTR DS:[ECX*4+77C43068]	
77C42F6B	90	NOP	
77C42F6C	FF248D EC2EC477	JMP DWORD PTR DS:[ECX*4+77C42FEC]	
77C42F73	90	NOP	
77C42F74	802E C4	SUB BYTE PTR DS:[EDI],0C4	
77C42F77	72 AC	JA SHORT nsvert.77C42F25	
77C42F79	2E	DAS	
77C42F7A	C477 D0	LES ESI,FWORD PTR DS:[EDI-30]	Modification of segment register
77C42F7D	2E	DAS	
77C42F7E	C477 23	LES ESI,FWORD PTR DS:[EDI+23]	Modification of segment register
77C42F81	018A 0688078A	ROR DWORD PTR DS:[EDX+8A078806],1	
77C42F87	46	INC ESI	
77C42F88	0188 47018A46	ADD DWORD PTR DS:[EAX+468A0147],ECX	
77C42F8E	02C1	ADD AL,CL	
77C42F90	E9 02884702	JMP 7A0BB797	
77C42F95	83C6 03	ADD ESI,3	
77C42F98	83C7 03	ADD EDI,3	
77C42F9B	83F9 08	CMPL ECX,8	
77C42F9E	72 CC	JB SHORT nsvert.77C42F6C	
77C42FA0	F3: A5	REP MOVSD WORD PTR ES:[EDI],DWORD PTR DS:[EDI]	
77C42FA2	FF2495 5830C477	JMP DWORD PTR DS:[EDX*4+77C43058]	
77C42FA9	8D49 00	LEA ECX,DWORD PTR DS:[ECX]	
77C42FAC	23D1	AND EDX,ECX	
77C42FAD	0000	MOV AL,BYTE PTR DS:[ECX]	

_Shift + F9 again:

00C1A82E	8B85 10D9FFFF	MOV EAX,DWORD PTR SS:[EBP-26F0]	UnPackMe.00460A0C
00C1A834	83C0 04	ADD EAX,4	
00C1A837	8985 10D9FFFF	MOV DWORD PTR SS:[EBP-26F0],EAX	
00C1A83D	E9 4DFCFFFF	JMP 00C1A48F	
00C1A842	FF15 7032C200	CALL DWORD PTR DS:[C23270]	kernel32.GetTickCount
00C1A848	2B85 A0D4FFFF	SUB EAX,DWORD PTR SS:[EBP-2B60]	
00C1A84E	8B8D A0D4FFFF	MOV ECX,DWORD PTR SS:[EBP-2B5C]	
00C1A854	6BC9 32	IMUL ECX,ECX,32	
00C1A857	81C1 D0070000	ADD ECX,700	
00C1A85D	3BC1	CMPL EAX,ECX	
00C1A85F	76 07	JBE SHORT 00C1A868	
00C1A861	C685 34D9FFFF	MOV BYTE PTR SS:[EBP-26CC],1	
00C1A868	83BD E4D7FFFF	CMPL DWORD PTR SS:[EBP-281C],0	
00C1A86F	0F85 8A000000	JNZ 00C1A8FF	
00C1A875	0FB685 90D4FFFF	MOVZX EAX,BYTE PTR SS:[EBP-2B70]	
00C1A87C	85C0	TEST EAX,EAX	
00C1A87E	74 7F	JE SHORT 00C1A8FF	
00C1A880	6A 00	PUSH 0	
00C1A882	8B85 94D4FFFF	MOV EAX,DWORD PTR SS:[EBP-2B6C]	
00C1A888	C1E0 02	SHL EAX,2	
00C1A88B	50	PUSH EAX	
00C1A88C	8B85 0CD8FFFF	MOV EAX,DWORD PTR SS:[EBP-27F4]	
00C1A892	0385 8CD4FFFF	ADD EAX,DWORD PTR SS:[EBP-2B74]	
00C1A898	50	PUSH EAX	
00C1A899	E8 AF1C0000	CALL 00C1C54D	
00C1A89E	83C4 0C	ADD ESP,0C	
00C1A8A1	8B85 94D4FFFF	MOV EAX,DWORD PTR SS:[EBP-2B6C]	
00C1A8A7	C1E0 02	SHL EAX,2	
00C1A8AA	50	PUSH EAX	
00C1A8AB	FFB5 6CD9FFFF	PUSH DWORD PTR SS:[EBP-2694]	
00C1A8B1	8B85 0CD8FFFF	MOV EAX,DWORD PTR SS:[EBP-27F4]	
00C1A8B7	0385 8CD4FFFF	ADD EAX,DWORD PTR SS:[EBP-2B74]	
00C1A8BD	50	PUSH EAX	

_Roi, A little mouse on the functions you see on Call stricmp function, note the address of it.

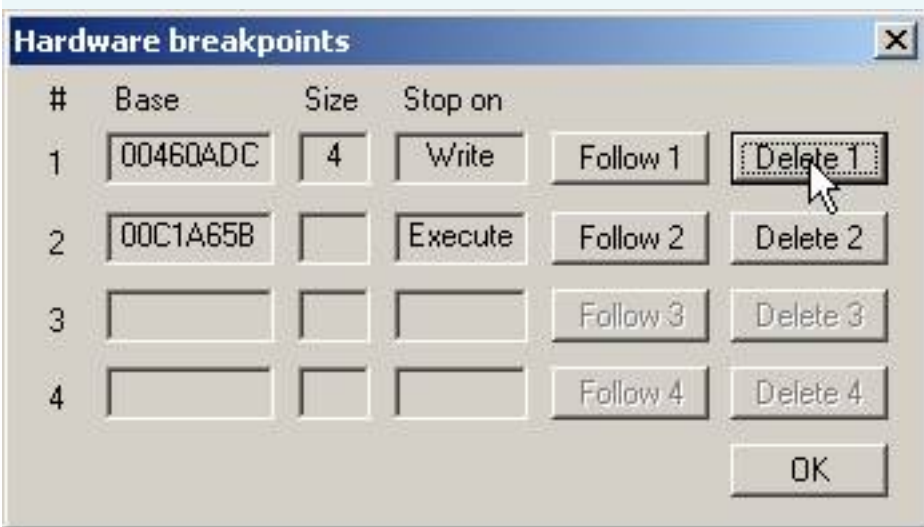
00C1A652	50	PUSH EAX	
00C1A653	8B85 54C2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DAC]	
00C1A659	FF30	PUSH DWORD PTR DS:[EAX]	
00C1A65B	E8 C079F0FF	CALL 00BF2020	
00C1A660	83C4 0C	ADD ESP,0C	
00C1A663	8085 54C1FFFF	LEA EAX,DWORD PTR SS:[EBP-3EAC]	
00C1A669	50	PUSH EAX	
00C1A66A	8085 64C2FFFF	LEA EAX,DWORD PTR SS:[EBP-3D9C]	
00C1A670	50	PUSH EAX	
00C1A671	FF15 7833C200	CALL DWORD PTR DS:[C23378]	msvcrt._stricmp
00C1A677	59	POP ECX	
00C1A678	59	POP ECX	
00C1A679	85C0	TEST EAX,EAX	
00C1A67B	75 11	JNZ SHORT 00C1A68E	
00C1A67D	8B85 54C2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DAC]	
00C1A683	8B40 08	MOV EAX,DWORD PTR DS:[EAX+8]	
00C1A686	8985 64CAFFFF	MOV DWORD PTR SS:[EBP-359C],EAX	
00C1A68C	EB 02	JMP SHORT 00C1A690	
00C1A68E	EB 9C	JMP SHORT 00C1A62C	
00C1A690	8B85 A4D4FFFF	MOV EAX,DWORD PTR SS:[EBP-2B5C]	
00C1A696	40	INC EAX	
00C1A697	8985 A4D4FFFF	MOV DWORD PTR SS:[EBP-2B5C],EAX	
00C1A69D	EB 37	JMP SHORT 00C1A6D6	
00C1A69F	8D8D 38D9FFFF	LEA ECX,DWORD PTR SS:[EBP-5C8]	
00C1A6A5	E8 9669F0FF	CALL 00BF1040	
00C1A6AA	0FB6C0	MOVZX EAX,AL	
00C1A6AD	99	CDB	
00C1A6AE	6A 14	PUSH 14	
00C1A6B0	59	POP ECX	
00C1A6B1	F7F9	IDIV ECX	
00C1A6B3	8B85 10D9FFFF	MOV EAX,DWORD PTR SS:[EBP-26F0]	
00C1A6B9	8B8C95 94D7FFFF	MOV ECX,DWORD PTR SS:[EBP+EDX*4-286C]	
00C1A6C0	0000	MOV DWORD PTR DS:[EAX],ECX	

Olly _Restart again, through your window Memory dump, Ctrl + G, enter 460ADC, selected first 4 bytes, placed on HardwareBreakpoint write Word style. Press Shift + F9, the code window, press Ctrl + G to enter 00C1A65B:

00C1A65B	096A A2	FLDQ WORD PTR DS:[EDX-5E]	
00C1A65E	41	INC ECX	
00C1A65F	8438	TEST BYTE PTR DS:[EAX],BH	
00C1A661	FE0E	DEC BYTE PTR DS:[ESI]	
00C1A663	318E 1E9F1A42	XOR DWORD PTR DS:[ESI+421A9F1E],ECX	
00C1A669	FC	CLO	
00C1A66A	6938 75EDAA85	IMUL EDI,DWORD PTR DS:[EAX],85AAED75	
00C1A670	99	CDB	
00C1A671	C3	RETN	
00C1A672	76 5B	JBE SHORT 00C1A6CF	
00C1A674	7A 4D	JPE SHORT 00C1A6C3	
00C1A676	837D 41 B7	CMPL DWORD PTR SS:[EBP+41],-49	
00C1A67A	31B6 BFFA3119	XOR DWORD PTR DS:[ESI+1931FABF],ESI	
00C1A680	E2 4C	LOOPD SHORT 00C1A6CE	
00C1A682	D0E7	SHL BH,1	
00C1A684	8A6CD9 4C	MOV CH,BYTE PTR DS:[ECX+EBX*8+4C]	
00C1A688	7E A7	JLE SHORT 00C1A631	
00C1A68A	90	NOP	
00C1A68B	90	NOP	
00C1A68C	92	XCHG EAX,EDX	
00C1A68D	F8	CLC	
00C1A68E	8068 55 E9	SUB BYTE PTR DS:[EAX+55],0E9	
00C1A692	6C	INS BYTE PTR ES:[EDI],0X	I/O command
00C1A693	8467 A1	TEST BYTE PTR DS:[EDI-5F],AH	
00C1A696	2B1D 195F7BC2	SUB EBX,DWORD PTR DS:[C27B5F19]	
00C1A69C	25 76E5B0F4	AND EAX,F480E576	
00C1A6A1	E5 FF	IN EAX,0FF	I/O command
00C1A6A3	1F	POP DS	Modification of segment register
00C1A6A4	76 26	JBE SHORT 00C1A6CC	
00C1A6A6	2C D5	SUB AL,0D5	
00C1A6A8	0895 1B9B535B	OR BYTE PTR SS:[EBP+5B539B1B],DL	
00C1A6AE	2E:DC2B	FSUBR QWORD PTR CS:[EBX]	
00C1A6B1	124C 50	OPB AL,BYTE PTR DS:[ESI+20]	

__Tai 00C1A65B address, you must click, set a (HardwareBreakpoint on Excuton):

_Nhu So we have two breakpoint, delete HarwareBreakpoint on Write:



_Nhan F9, you will jump to this code:

<pre> 00C1A65B E8 C079FDFF CALL 00BF2020 00C1A660 83C4 0C ADD ESP,0C 00C1A663 8D85 54C1FFFF LEA EAX,DWORD PTR SS:[EBP-3EAC] 00C1A669 50 PUSH EAX 00C1A66A 8D85 64C2FFFF LEA EAX,DWORD PTR SS:[EBP-3D9C] 00C1A670 50 PUSH EAX 00C1A671 FF15 7833C200 CALL DWORD PTR DS:[C23378] 00C1A677 59 POP ECX 00C1A678 59 POP ECX 00C1A679 85C0 TEST EAX,EAX 00C1A67B 75 11 JNZ SHORT 00C1A68E 00C1A67D 8B85 54C2FFFF MOV EAX,DWORD PTR SS:[EBP-3DAC] 00C1A683 8B40 08 MOV EAX,DWORD PTR DS:[EAX+8] 00C1A686 8985 64CAFFFF MOV DWORD PTR SS:[EBP-359C],EAX 00C1A68C EB 02 JMP SHORT 00C1A690 00C1A68E EB 9C JMP SHORT 00C1A62C 00C1A690 8B85 A4D4FFFF MOV EAX,DWORD PTR SS:[EBP-2B5C] 00C1A696 40 INC EAX 00C1A697 8985 A4D4FFFF MOV DWORD PTR SS:[EBP-2B5C],EAX 00C1A69D EB 37 JMP SHORT 00C1A6D6 00C1A69F 8D8D 38D9FFFF LEA ECX,DWORD PTR SS:[EBP-1C8] 00C1A6A5 E8 9669FDFF CALL 00BF1040 00C1A6AA 0FB6C0 MOVZX EAX,AL 00C1A6AD 99 CDQ 00C1A6AE 6A 14 PUSH 14 00C1A6B0 59 POP ECX 00C1A6B1 F7F9 IDIV ECX 00C1A6B3 8B85 10D9FFFF MOV EAX,DWORD PTR SS:[EBP-26F0] 00C1A6B9 8B8C95 94D7FFFF MOV ECX,DWORD PTR SS:[EBP+EDX*4-286C] 00C1A6C0 8908 MOV DWORD PTR DS:[EAX],ECX 00C1A6C2 8B85 10D9FFFF MOV EAX,DWORD PTR SS:[EBP-26F0] 00C1A6C8 83C0 04 ADD EAX,4 00C1A6CD 8B85 10D9FFFF MOV EAX,DWORD PTR SS:[EBP-26F0] </pre>	<pre> msvcrt._stricmp </pre>
--	------------------------------

_Chung We need to patch this function, enter at 00C1A65B E8 C079FDFF CALL 00BF2020 you jump into the jaw, Ctrl + E, the 55 to C3:

<pre> 00BF2020 55 PUSH EBP 00BF2021 8BEC MOV EBP,ESP 00BF2023 51 PUSH ECX 00BF2024 A1 70B8C200 MOV EAX,DWORD PTR DS:[C2B870] 00BF2029 53 PUSH EBX 00BF202A 56 PUSH ESI 00BF202B 57 PUSH EDI 00BF202C 85C0 TEST EAX,EAX 00BF202E 75 37 JNZ SHORT 00BF2067 00BF2030 68 00010000 PUSH 100 00BF2035 6745 5C 70C00041 MOV EDI,DWORD PTR DS:[5C70C000] </pre>	<pre> </pre>
--	--------------

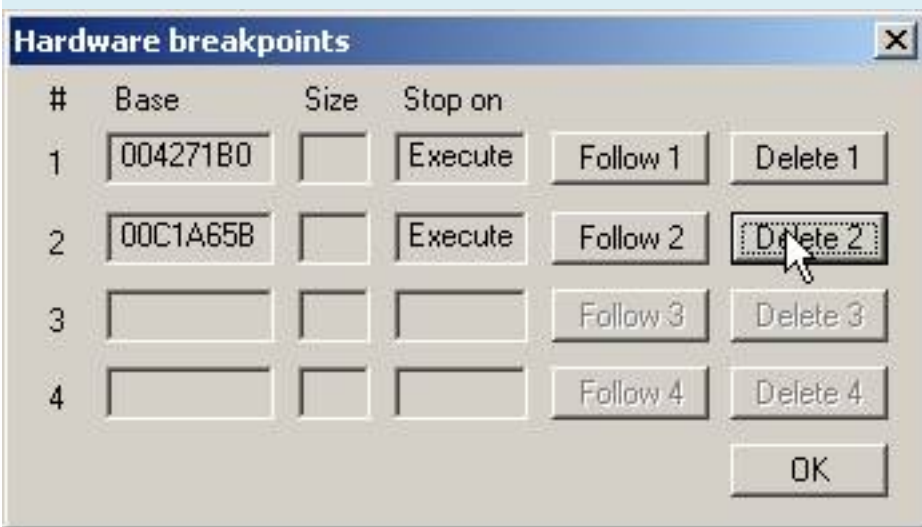
00BF2020	CS	RETN	
00BF2021	8BEC	MOV EBP,ESP	
00BF2023	51	PUSH ECX	
00BF2024	A1 70B8C200	MOV EAX,DWORD PTR DS:[C2B870]	
00BF2029	53	PUSH EBX	
00BF202A	56	PUSH ESI	
00BF202B	57	PUSH EDI	
00BF202C	85C0	TEST EAX,EAX	
00BF202E	75 37	JNZ SHORT 00BF2067	
00BF2030	68 00010000	PUSH 100	

_Bam '-', Press Ctrl + G, enter the address of OEP: 4271B0, we also set a breakpoint HE:

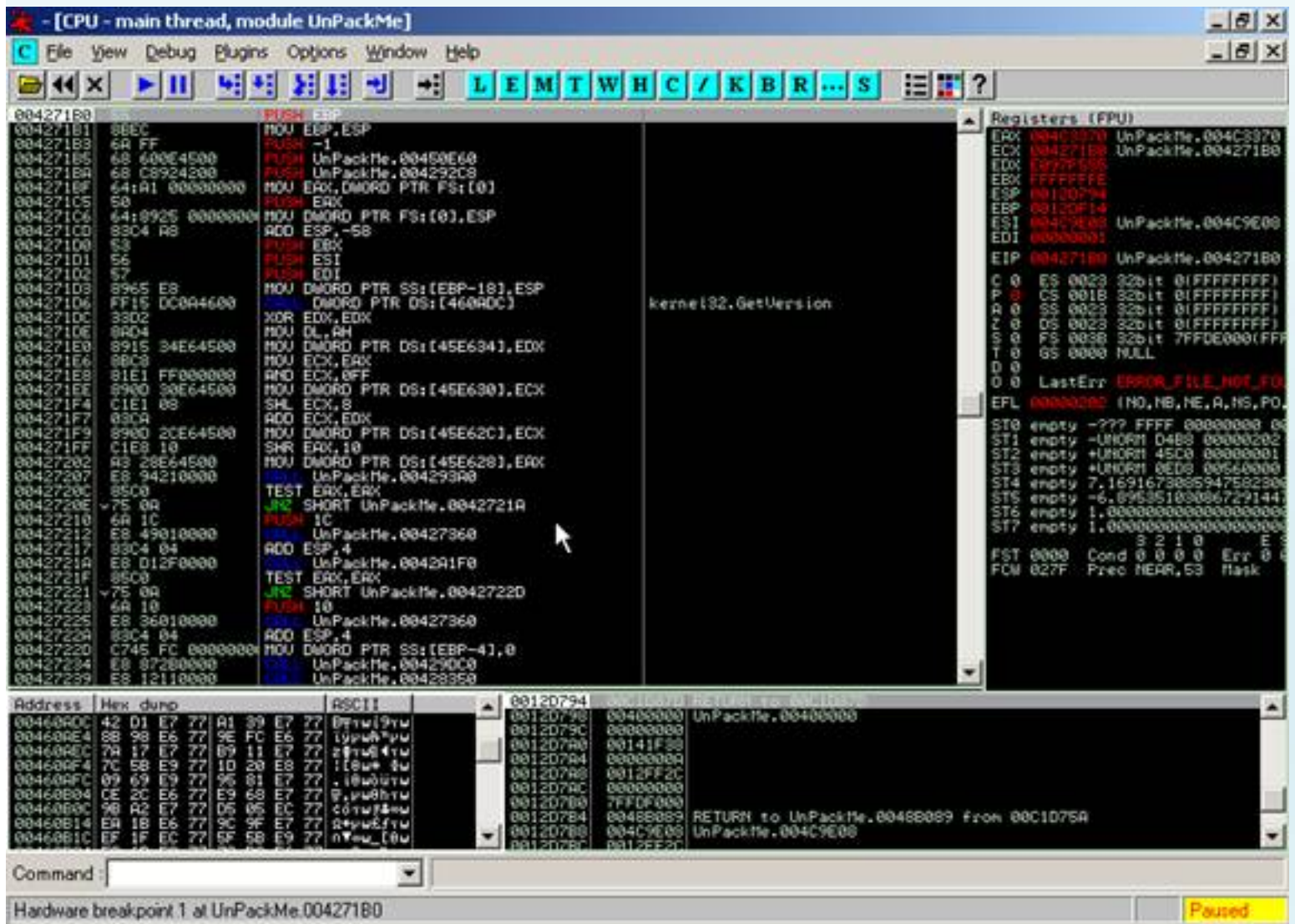
The screenshot shows the OllyDbg interface. The assembly window displays code starting at address 004271B0. A context menu is open over the assembly window, with the 'Breakpoint' option selected, and a sub-menu showing 'Hardware, on execution' highlighted. The registers window on the right shows the current state of CPU registers. The memory dump window at the bottom shows the hex dump of the current memory location.

Address	Hex dump	Unicode
004271B0	55	
004271B1	8BEC	
004271B3	6A FF	
004271B5	68 60E4500	
004271B8	68 C8924200	
004271BF	64:A1 00000000	
004271C5	50	
004271C6	64:8925 00000000	
004271C0	83C4 A8	
004271D0	53	
004271D1	56	
004271D2	57	
004271D3	8965 E8	
004271D6	FF15 DC0A4600	
004271DC	33D2	
004271DE	9AD4	
004271E0	8915 34E64500	
004271E6	8BC8	
004271E8	81E1 FF000000	
004271EE	8900 30E64500	
004271F4	C1E1 00	
004271F7	83C0	
004271F9	8900 2CE64500	
004271FF	C1E8 10	
00427202	A3 20E64500	
00427207	E8 94210000	
0042720C	85C0	
0042720E	75 00	
00427210	6A 1C	
00427212	E8 49010000	
00427217	83C4 04	
0042721A	E8 012F0000	
0042721F	85C0	

_Ban Again removed in HE 00C1A65B:



_Nhan '-', You return to the function 00C1A65B E8 C079FDFF CALL 00BF2020, press F9, you will break in with OEP Full IAT. Full dump with PETools, fix IAT normal thẳng cut out any invalid.

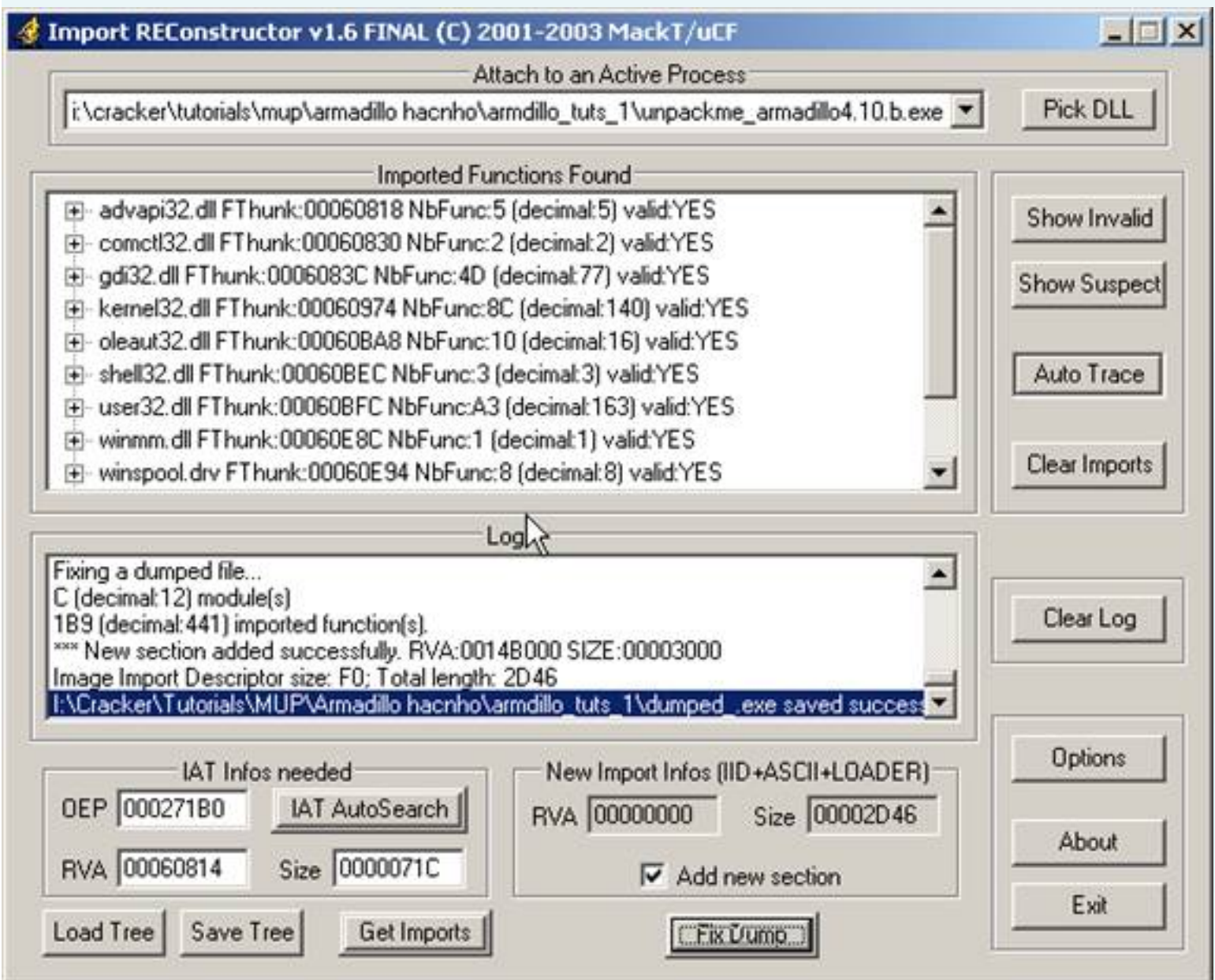


Path	PID	ImageBase	ImageSize
c:\program files\alcohol soft\alcohol 120\star...	00000690	00400000	0003A000
c:\windows\system32\wdmgr.exe	000006A8	01000000	0000C000
c:\windows\system32\fast.exe	00000764	01000000	0000C000
c:\program files\winamp\winamp.exe	00000C00	00400000	0012E000
c:\windows\explorer.exe	00000B9C	01000000	000F8000
c:\windows\system32\wisptis.exe	0000083C	01000000	00031000
c:\program files\unikey\unikey.exe	000005F8	00400000	0002F000
c:\program files\microsoft office\office11\win...	00000A0C	30000000	008AA000
c:\program files\techsmith\snagit 7\snagit32....	00000E50	00400000	00390000
c:\program files\techsmith\snagit 7\tschelp.exe	0000080C	00400000	0000A000
i:\cracker\debug- disassembler\odbg110_org...	0000013C	00400000	00153000
i:\cracker\tutori...	4	00400000	0014B000
i:\cracker\utilitie...	0	00400000	00036000

Path	PID	ImageBase	ImageSize
i:\cracker\tutori...	0	0014B000	
c:\windows\sys...	0	000A7000	
c:\windows\sys...	0	000E6000	
c:\windows\sys...	0	0008C000	
c:\windows\sys...	0	00040000	
c:\windows\sys...	0	0008D000	
c:\windows\sys...	0	00086000	
c:\windows\sys...	0	0001C000	
c:\windows\sys...	0	00008000	
c:\windows\sys...	0	0005A000	
c:\windows\system32\msctf.dll	74720000	00044000	
c:\windows\system32\msvcrt.dll	77C10000	00053000	
c:\windows\system32\msctfime.ime	00A40000	00028000	
c:\windows\system32\msctfime.ime	77100000	00021000	

Context menu options for the selected process:

- dump full...
- dump partial...
- dump region...
- active dump engine
- priority
- correct ImageSize
- load into PE editor... (temp file)
- load into PE editor... (read only)
- burn process
- refresh F5



Kiem Dumped.exe investigation file, run well. Good work, unpacked successful!

_That That we not get all the IAT len = 71C. Full IAT has 460F42-len = 460818 = 72A. Full IAT has attached this tut.

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, Kienmanowar, benina, TQN, the_lighthouse, Nini, hoadongnoi, dqtlm, hosiminh, Nilrem, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, anh_surprised ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RiF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx to authors of OllyDBG.

To be continued ...

Written by [hacnho](#) (tutorial date: Saigon 25/08/2005)

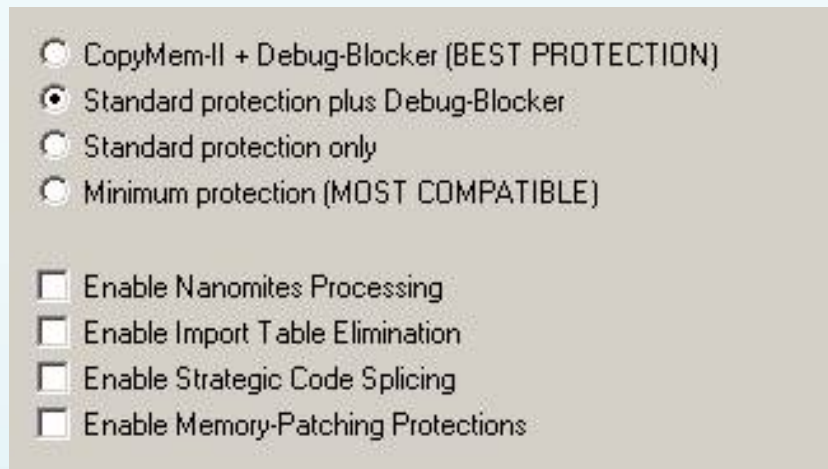
Armadillo collect sand-stone

P han II Identification unpack + Debug Blocker type 1

I. Introduction:

Welcome to the first I only introduced through a standard model unpack Armadillo + dump anti, anti breakpoint, anti patch mem. This current form has two unpack. First is a paste in PE Header tut's bags, the second is to find MAGIC JUMP + patch that tlandn mentioned. In the process can unpack flexible switch. How to unpack this many variables that I do not have enough time to write all. You must choose to unpack a soft, patience is the virtue necessary to unpack armadillo.

In this tut I will guide you unpack a target to protect the Standard model + Debugblocker. Please see the type of Armadillo 4: 30 Protect Pro Full version:



Method to unpack protect this type is known by only one and save people from the two men 1.x. pioneering work unpacker method is ArmKiller and [LUNAR_DUST]. On the NET now has two focuses of the forum trum unpack armadillo is RCE Anticrack board and board. This method is popular on RCE, you can find in Woodmann.net read. Read more at:

<http://www.woodmann.net/forum/forumdisplay.php?f=4>

So DebugBlocker + II CopyMem What? Bít I do not say it correctly, but the experience is probably correct J :

_Khi Option we choose Debug-Blocker, the Armadillo will operate as it will create two on memory processes (process), the two called by the IT process is a role as a server, a process is created and managed by a server called a client. Calls by the cracker will have a divorce, a father is a child or a son. Progress parent will control the children in the debug mode, the load on it will create a Debug Events, then will joker unpack thẳng children, created for the children IAT, PE Header, sections by creating a temporary file (s ver 2.xx that before) or the treatment of memory and run it. We take this process to unpack it using the API are two breakpoint **WaitForDebugEvent** and **WriteProcessMemory**.

BOOL WriteProcessMemory (

HANDLE hProcess, // handle to process whose memory is written to
LPVOID lpBaseAddress, // address to start writing to
LPVOID lpBuffer, // Pointer to buffer to write data to
DWORD nSize, // number of bytes to write
LPDWORD lpNumberOfBytesWritten // actual number of bytes written
);

BOOL WaitForDebugEvent (

LPDEBUG_EVENT lpDebugEvent, // address of structure for event information
DWORD dwMilliseconds // number of milliseconds to wait for event
);

II. Tools

1:10 OllyDBG with plugin: Hide Debugger 1.2.3f, Armadillo Process Detach Plugin v1.0, OllyDBG PE Dumper 3.0.3, Command Bar 3.10.109c has bug patch with RE-PAIR 0.6 + AntiDetectOlly_v2.2.4. It is important that you only use the plugin above, the other plugin please take away place. If you is not never Attach the child process, always hanging in Asia.

PE Tools v1.5.600.2005

Import REConstructor v1.6 FINAL

Debug blocker detect script I write for the media to determine the type protect this. How the script of this is quite simple: Find the signal **0000C085** file to determine. When xài must ignore all execptions.

Address	Hex dump	Disassembly
00419FA0	85C0	TEST EAX, EAX
00419FAF	0F85 74020000	JNZ unpackr...0041A229
00419FB5	6A 01	PUSH 1

// Scripts for OllyScript plugin by SHaG - <http://ollyscript.apsvans.com>

*/ **

////////////////////////////////////

// Armadillo's Debug Feature blocker or signal CopyMEM2 detective

// Author: hacnho from mod MEPHiSTOs - ARMADiLLO Detective v1.00

// Email: hacnho@hotmail.com

// Website: <http://tinicat.de/hacnho>

// OS: WinXP Pro SP1, OllyDbg 1:10 Final, OllyScript v0.92

// ReLeAsE Date: 14 July 2005

////////////////////////////////////

** /*

```
var signalcheck
var mem
var time
var nono

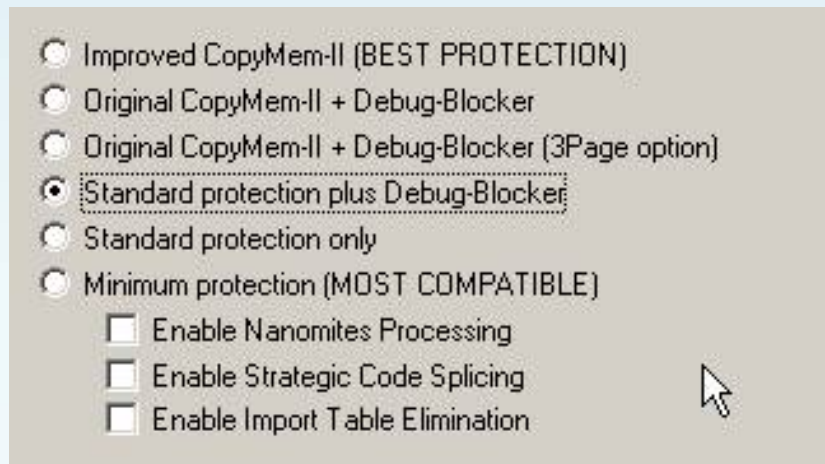
GPA "OpenMutexA", "kernel32.dll"
mov mem, $ RESULT
BP mem
esto
esto
rtr
STI
bc mem
GPA "time", "MSVCRT.dll"
mov time, $ RESULT
BP time
mov signalcheck, [eip]
and signalcheck, 0000FFFF
Cmp signalcheck, 0000C085 / / checking for debug blocker signal
je db

db:
jne nono
msg "This file is protected with Armadillo's Debug Feature blocker or CopyMEM II."
ret

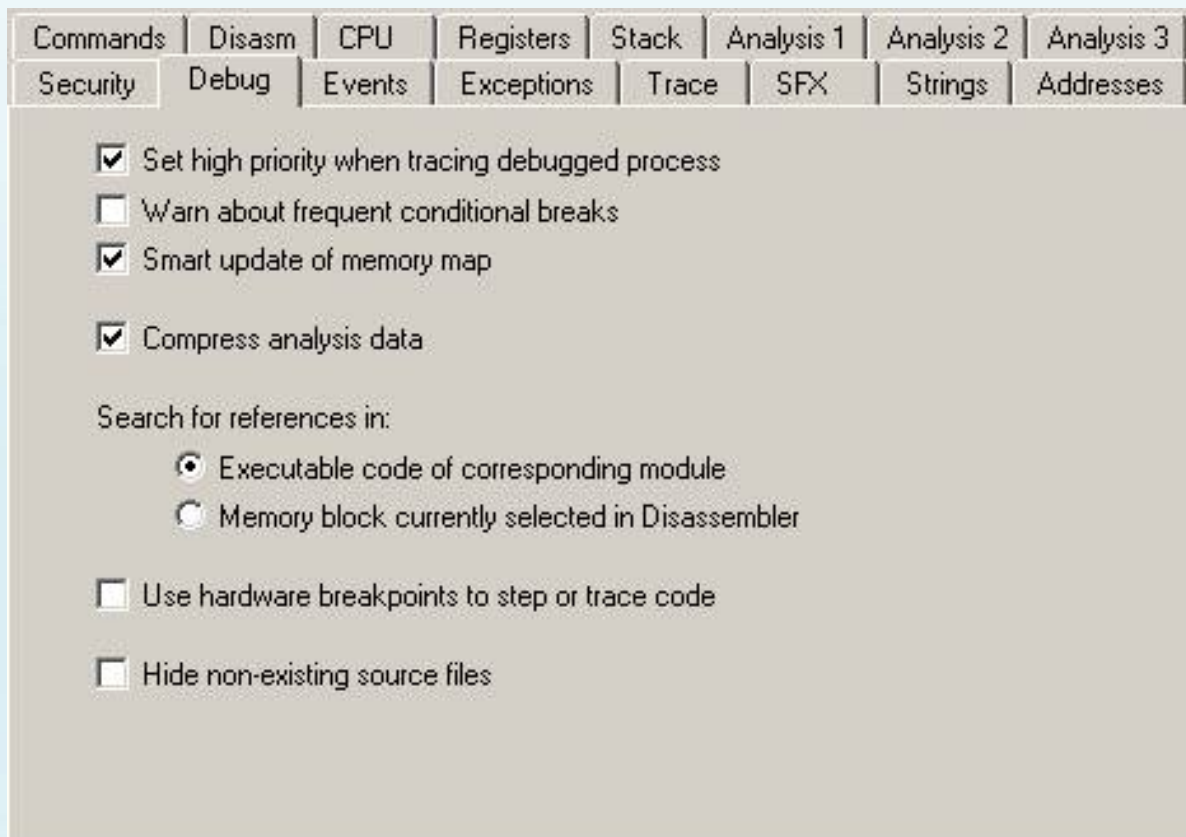
nono:
msg "This file is protected with Armadillo's Debug Feature blocker or CopyMEM II."
ret
```

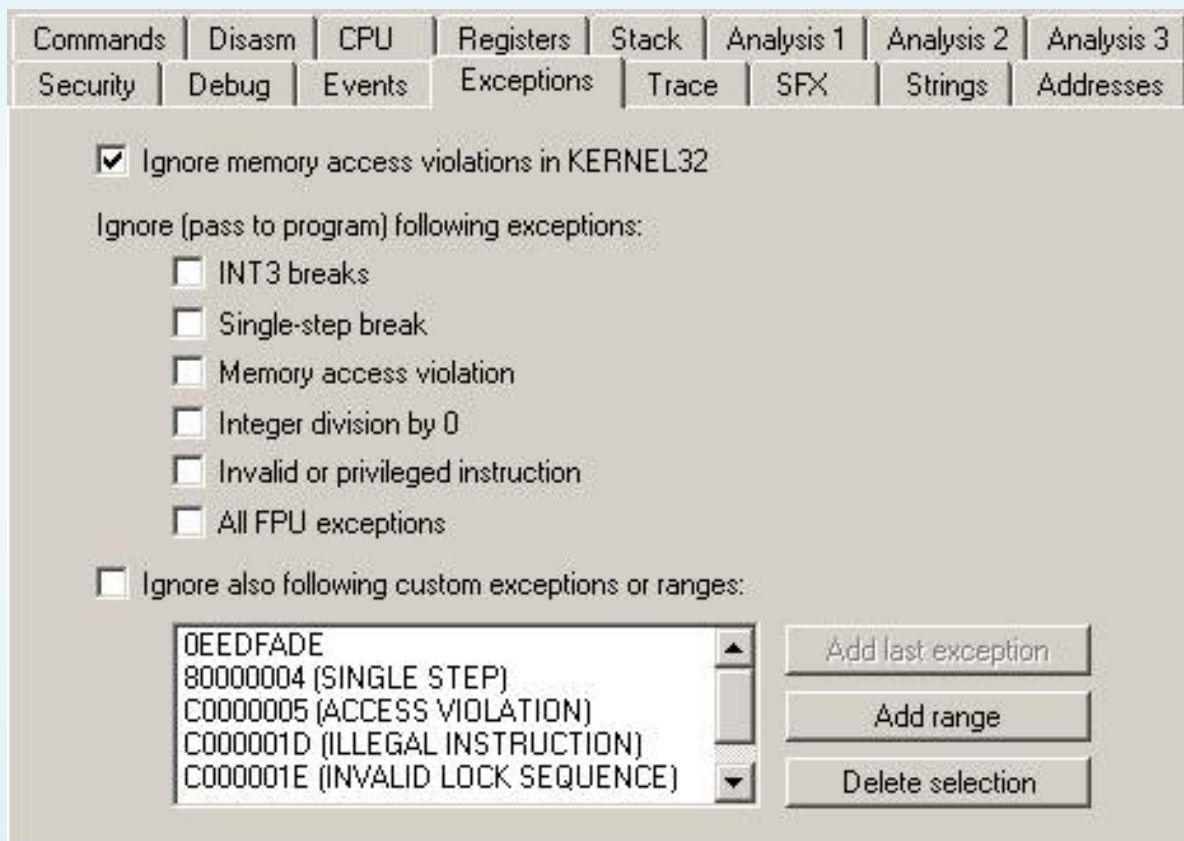
III. Manual Unpacking

TUT # 1: snd-UnPackMe_Armadillo4.00 Standard + Debug-Blocker

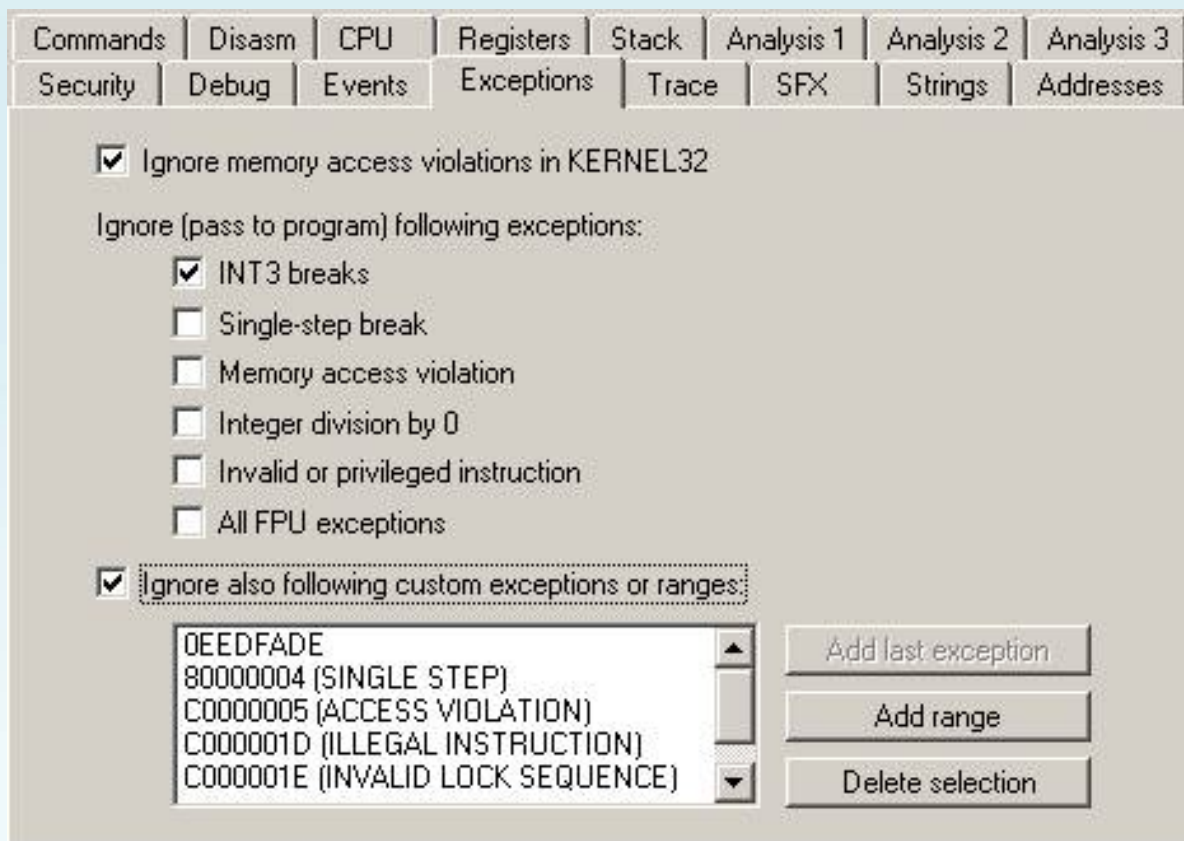


_Truoc All suggest you tune in Options OllyDBG as follows, if not will not Attach the child process:





_Khi We work the process as the father on, when working with children is as follows:



_Co Three method to work on the parent process, use a script **arma_detach.osc**, using hand and Plugin: d. I just you to use your hands. The other two it is done automatically,

chả you need to do anything but crash quite a lot, use your hands to ensure food J .

*. After OllyDBG load used to target:

0048E743	55	PUSH EBP	SE handler installation
0048E744	8B EC	MOV EBP, ESP	
0048E746	6A FF	PUSH -1	
0048E748	68 20 8B 4B 00	PUSH target1.004B8B20	
0048E74D	68 80 E4 48 00	PUSH target1.0048E480	
0048E752	64: A1 00 00 00 00	MOV EAX, DWORD PTR FS:[0]	
0048E758	50	PUSH EAX	
0048E759	64: 89 25 00 00 00	MOV DWORD PTR FS:[0], ESP	
0048E760	83 EC 58	SUB ESP, 58	
0048E763	53	PUSH EBX	
0048E764	56	PUSH ESI	
0048E765	57	PUSH EDI	

*. Set breakpoint WriteProcessMemory, Shift + F9 twice (as long as you target after we saw the window stack visible function WriteProcessMemory have to load our target is OK:

0012B9B8	00482000	CALL to WriteProcessMemory from
0012B9BC	0000004C	hProcess = 0000004C (window)
0012B9C0	0048E743	Address = 48E743
0012B9C4	004B9E34	Buffer = target1.004B9E34
0012B9C8	00000002	BytesToWrite = 2
0012B9CC	0012BCAC	pBytesWritten = 0012BCAC
0012B9D0	0012D248	UNICODE "Kernel32.dll"
0012B9D4	00002710	
0012B9D8	7FFDF000	
0012B9DC	00482E70	RETURN to target1.00482E70 from
0012B9E0	00000050	
0012B9E4	0012B9E8	
0012B9E8	00010001	
0012B9EC	00000000	
0012B9F0	00000038	
0012B9F4	00000020	
0012B9F8	00000000	

*. You pay attention to the Buffer, which will patch it to jump to the EIP. Now you must click on the Buffer, select Follow in dump. Through the window dump, you will see the following:

Address	Hex dump
004B9E34	55 8B 00 00 00 00 00 00
004B9E3C	00 00 00 00 00 00 00 00
004B9E44	00 00 00 00 00 00 00 00
004B9E4C	00 00 00 00 00 00 00 00
004B9E54	00 00 00 00 00 00 00 00
004B9E5C	00 00 00 00 00 00 00 00
004B9E64	00 00 00 00 00 00 00 00
004B9E6C	00 00 00 00 00 00 00 00

*. Two bytes of this block, Ctrl + E to revise **EB FE**:

HEX +00

EB FE

*. Now you add a breakpoint again to hook the process I have said above, parents thẳng process will create a DebugEvent.Ok, BP WaitForDebugEvent, Shift + F9, Ctrl + F9, F7. Too dangerous: d.


```

0012BCB8 0047E8BF CALL to WaitForDebugEvent from target1.
0012BCBC 0012CD90 pDebugEvent = 0012CD90
0012BCC0 000003E8 Timeout = 1000. ms
0012BCC4 0012FF2C
0012BCC8 00000000
0012BCCC 7FFDF000

```

```

0047E8BF . 85C0 TEST EAX,EAX
0047E8C1 .v 0F84 2B270000 JE target1.00480FF2
0047E8C7 . 8B85 FCFDFFFF MOV EAX,DWORD PTR SS:[EBP-204]
0047E8CD . 25 FF000000 AND EAX,0FF
0047E8D2 . 85C0 TEST EAX,EAX
0047E8D4 .v 74 13 JE SHORT target1.0047E8E9
0047E8D6 . 8B0D 449F4B00 MOV ECX,DWORD PTR DS:[4B9F44]
0047E8DC . 8379 20 00 CMP DWORD PTR DS:[ECX+20],0
0047E8E0 .v 74 07 JE SHORT target1.0047E8E9
0047E8E2 . C685 FCFDFFFF MOV BYTE PTR SS:[EBP-204],0
0047E8E9 > 68 389E4B00 PUSH target1.004B9E38
0047E8EE . FF15 A4314B00 CALL DWORD PTR DS:[<&KERNEL32.EnterCrit
0047E8F4 . 60 PUSHAD

```

*. To have two method to assemble:

PUSH PID (PID = Process ID)

CALL DebugActiveProcessStop

NOP

You may have been accurate in PID Olly Dbg, by clicking File> Attach>

There we will have 2 files are running FlashFavourite - select get correct process that is not highlighted RED. This is the PID that we need ...

Or

*/ * * 47E8BF / PUSH EAX*

*/ * * 47E8C0 / CALL kernel32.DebugActiveProcessStop*

*/ * * 47E8C5 / NOP*

(Value of EAX = 1).

```

Registers (FPU)
EAX 00000001
ECX 0012BCA0
EDX 7FFE0304
EBX 7FFDF000
ESP 0012BCBC
EBP 0012D7A4
ESI 00002710
EDI 0012C310

```

```

0047E8B8 . 52 PUSH EDX
0047E8B9 . FF15 E0304B00 CALL DWORD PTR DS:[<&KERNEL32.WaitForDel pDebugEvent.
0047E8BF . 50 PUSH EAX WaitForDebugEvent
0047E8C0 . E8 6E09A377 CALL kernel32.DebugActiveProcessStop
0047E8C5 . 90 NOP
0047E8C6 ? 008B 85FCFDFF ADD BYTE PTR DS:[EBX+FFFD0C85],CL
0047E8CC ? FF25 FF000000 JMP DWORD PTR DS:[FF]
0047E8D2 . 85C0 TEST EAX,EAX
0047E8D4 .v 74 13 JE SHORT target1.0047E8E9

```

Process	Name	Window	Path
00000000	spoolsv		C:\WINDOWS\system32\spoolsv.exe
0000010C	smss		\SystemRoot\System32\smss.exe
000001FC	csrss		\\??C:\WINDOWS\system32\csrss.exe
00000214	winlogon	NetDOE Agent	\\??C:\WINDOWS\system32\winlogon.exe
00000240	services		C:\WINDOWS\system32\services.exe
0000024C	lsass		C:\WINDOWS\system32\lsass.exe
000002F4	svchost		C:\WINDOWS\system32\svchost.exe
00000320	svchost		C:\WINDOWS\system32\svchost.exe
00000390	svchost		C:\WINDOWS\system32\svchost.exe
000003D0	svchost		C:\WINDOWS\system32\svchost.exe
000003F8	target1		C:\Documents and Settings\Le Nguyen Khang\My Documents\Armadillo_tuts_2\target1.exe
00000454	Explorer	SysFader	C:\WINDOWS\Explorer.EXE
00000530	inetinfo		C:\WINDOWS\System32\inetinfo.exe
000005EC	taskswi		C:\WINDOWS\System32\taskswitch.exe
000005F8	fast	Default IHE	C:\WINDOWS\System32\fast.exe
00000598	udfmgr		C:\WINDOWS\System32\udfmgr.exe
00000720	Fast		C:\WINDOWS\System32\Fast.exe
000007E4	otfmon	TF_FloatingLangBar_MndTitl	C:\WINDOWS\System32\otfmon.exe
00000800	dllhost		C:\WINDOWS\System32\dllhost.exe
0000082C	dllhost		C:\WINDOWS\System32\dllhost.exe
0000083C	target1		C:\Documents and Settings\Le Nguyen Khang\My Documents\Armadillo_tuts_2\target1.exe
00000858	WINWORD	CiceroUIWndFrame	C:\Program Files\Microsoft Office\OFFICE11\WINWORD.EXE
00000880	FIREFOX	Reverse Engineering Associ	C:\PROGRAM FILES\FIREFOX\FIREFOX.EXE
000008B8	UniKey	UniKey 3.62	C:\Program Files\UniKey\UniKey.exe
00000C04	Snagit32	Snagit Capture Preview	C:\Program Files\TechSmith\Snagit 7\Snagit32.exe
00000C60	WISPTIS	Default IHE	C:\WINDOWS\System32\WISPTIS.EXE
00000D08	TSCHelp	C:\Program Files\TechSmith	C:\Program Files\TechSmith\Snagit 7\TSCHelp.exe
00000E30	EMEDITOR	arna_detach.txt - EEditor	C:\Program Files\EmEditor\EMEDITOR.EXE
00000FF4	winamp	Winamp Playlist Editor	C:\Program Files\Winamp\winamp.exe

0047E8B8	52	PUSH EDX	pDebugEvent
0047E8B9	FF15 E0304B00	CALL DWORD PTR DS:[&KERNEL32.WaitForDel	WaitForDebugEvent
0047E8BF	60 3C080000	PUSH 83C	
0047E8C4	E8 6A09A377	CALL kernel32.DebugActiveProcessStop	
0047E8C9	90	NOP	
0047E8CA	F0	STO	
0047E8CB	FFFF	???	Unknown command
0047E8CD	25 FF000000	AND EAX,0FF	
0047E8D2	85C0	TEST EAX,EAX	
0047E8D4	74 13	JE SHORT target1.0047E8E9	
0047E8D6	8B0D 449F4B00	MOV ECX,DWORD PTR DS:[4B9F44]	
0047E8DC	8379 20 00	CMPL DWORD PTR DS:[ECX+20],0	
0047E8E0	74 07	JE SHORT target1.0047E8E9	
0047E8E2	C685 FCFDFFFF	MOV BYTE PTR SS:[EBP-204],0	
0047E8E9	68 389E4B00	PUSH target1.004B9E38	pCriticalSection = target1.00
0047E8EE	FF15 A4314B00	CALL DWORD PTR DS:[&KERNEL32.EnterCrit	EnterCriticalSection

*. You press F8 to the file, stop, the process of working on the parent process has been temporarily closed.

_Bay Time you open a window OllyDBG other, choose File-> Attach, select the process with PID like you find in (the bags are 83C). If you configure as above, the Attach a rup.

77F767CE	C3	RETN	
77F767CF	CC	INT3	
77F767D0	C3	RETN	
77F767D1	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]	
77F767D5	CC	INT3	
77F767D6	C2 0400	RETN 4	
77F767D9	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]	
77F767DF	C3	RETN	
77F767E0	57	PUSH EDI	
77F767E1	8B7C24 0C	MOV EDI,DWORD PTR SS:[ESP+C]	
77F767E5	8B5424 08	MOV EDX,DWORD PTR SS:[ESP+8]	
77F767E9	C702 00000000	MOV DWORD PTR DS:[EDX],0	
77F767EF	897A 04	MOV DWORD PTR DS:[EDX+4],EDI	
77F767F2	0BFF	OR EDI,EDI	
77F767F4	74 11	JE SHORT ntddll.77F76807	

_Bay Time you edit the OPTIONS OllyDBG like above, press F9, then press F12.

0048E743	-EB FE	JMP SHORT target1.<ModuleEntryPoint>	
0048E745	EC	IN AL,DX	I/O command
0048E746	6A FF	PUSH -1	
0048E748	68 208B4B00	PUSH target1.004B8B20	
0048E74D	68 80E44800	PUSH target1.0048E480	
0048E752	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
0048E758	50	PUSH EAX	
0048E759	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
0048E760	83EC 58	SUB ESP,58	

_Chac Certainly you remember just two bytes original patch above. If kô remember the last window is OllyDBG father debug process, click the button **L**, to review:

77E61A94	318 loops, 8 switches
77E61A94	Breakpoint at kernel32.WriteProcessMemory
77E61A94	Breakpoint at kernel32.WriteProcessMemory
	*** original OEP bytes :
	[pBuffer] = 00008B55
	*** changed OEP bytes :
	[pBuffer] = 0000FEEB
77EAF13C	Breakpoint at kernel32.WaitForDebugEvent
77EAF13C	Breakpoint at kernel32.WaitForDebugEvent
77EAF13C	Breakpoint at kernel32.WaitForDebugEvent
	pDebugEvent = 0012CD90
	child_ProcID = 000001D4

_OK, Org bytes is 558B. Back Olly debug the process, press Ctrl + E to edit **EB FE 55 8B**.

0048E743	55	PUSH EBP	
0048E744	8B EC	MOV EBP,ESP	
0048E746	6A FF	PUSH -1	
0048E748	68 208B4B00	PUSH target1.004B8B20	
0048E74D	68 80E44800	PUSH target1.0048E480	
0048E752	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
0048E758	50	PUSH EAX	
0048E759	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
0048E760	83EC 58	SUB ESP,58	
0048E763	53	PUSH EBX	
0048E764	56	PUSH ESI	
0048E765	57	PUSH EDI	
0048E766	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0048E769	FF15 88314B00	JMP DWORD PTR DS:[<&KERNEL32.GetVersion	kernel32.GetVersion
0048E76E	33D2	XOR EDX,EDX	

_Bay Time you press Alt + M to Memory Map, set breakpoint on memory access in text sections.

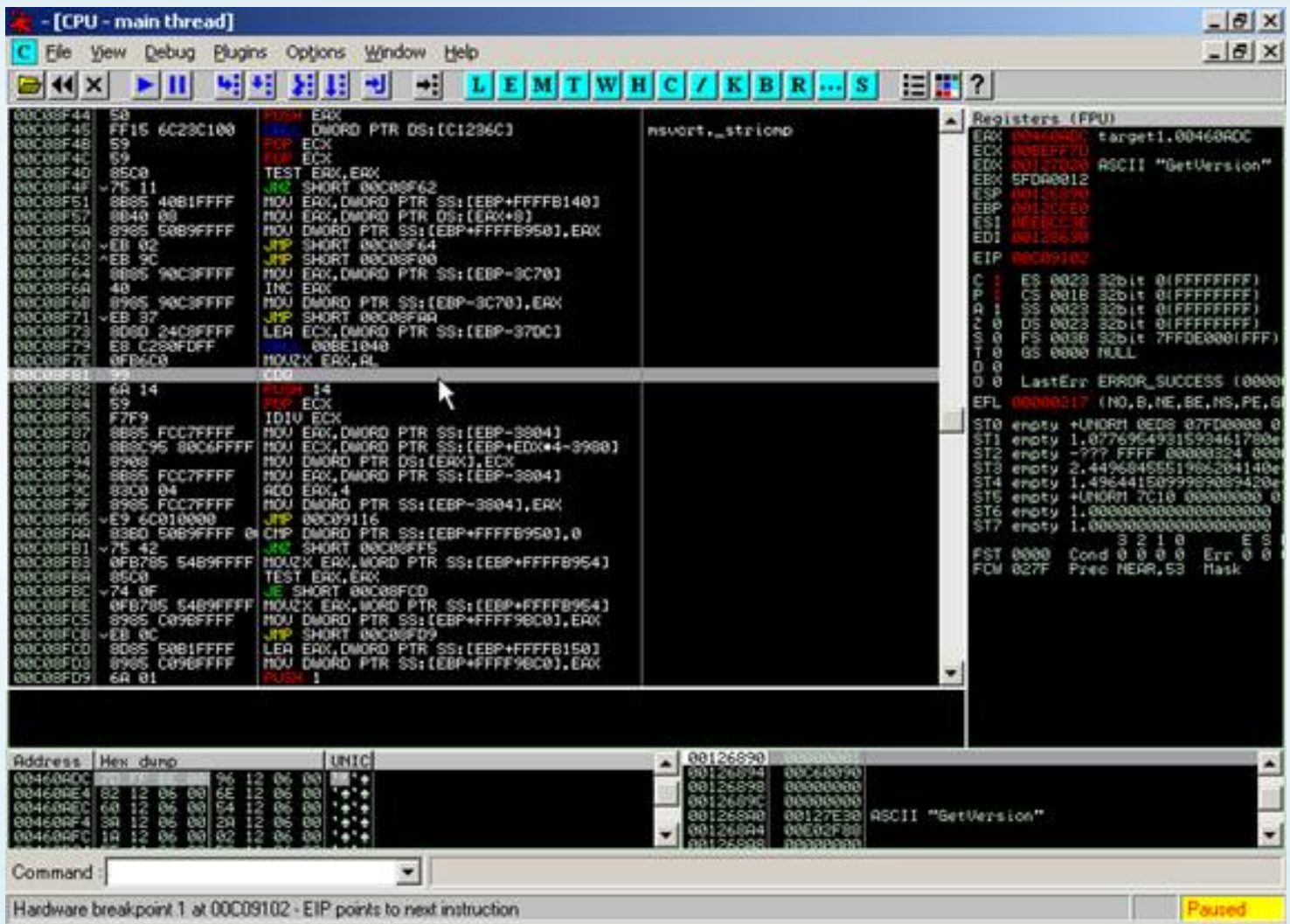
003E0000	00002000				Map	R	E		
003F0000	00008000				Priv	RW			
00400000	00001000	target1		PE header	Imag	R			
00401000	00040000	target1	.text						
0044B000	0000C000	target1	.rdata						
00457000	00009000	target1	.data						
00460000	00003000	target1	.idata						
00463000	00040000	target1	.text1	code					
004A3000	00010000	target1	.adata						
004B3000	00020000	target1	.data1	data, imp					
004D3000	00060000	target1	.pdata						
00533000	00008000	target1	.rsrc	resource					
00540000	00103000								
00650000	0014C000								
00950000	00001000								
00960000	00001000								
00980000	00001000								
00990000	00001000								
009A0000	00001000								
009C0000	00001000								

_Shift + F9, you will jump to OEP. To call out the function.

004271B0	55	PUSH EBP
004271B1	88EC	MOV EBP,ESP
004271B3	6A FF	PUSH -1
004271B5	68 600E4500	PUSH target1.00450E60
004271B9	68 C8924200	PUSH target1.004292C8
004271BF	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
004271C5	50	PUSH EAX
004271C6	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
004271CD	83C4 A8	ADD ESP,-58
004271D0	53	PUSH EBX
004271D1	56	PUSH ESI
004271D2	57	PUSH EDI
004271D3	8965 E0	MOV DWORD PTR SS:[EDP-18],ESP
004271D6	FF15 DC0A4600	CALL DWORD PTR DS:[460ADC]
004271DC	3302	XOR EDX,EDX
004271DE	8AD4	MOV DL,AH
004271E0	8915 34E64500	MOV DWORD PTR DS:[45E634],EDX
004271E6	8BC8	MOV ECX,EAX
004271E8	81E1 FF000000	AND ECX,0FF
004271EE	8900 30E64500	MOV DWORD PTR DS:[45E630],ECX
004271F4	C1E1 08	SHL ECX,8
004271F7	03C9	ADD ECX,EDX

_Khoai Too, OEP to call. This return to fix Magic JUMP as standard version. To this you will have many ways to dump, fix IAT. How to paste PE Header, Find magic jump, plunging even as the parties MP2K, dek thèm ráo fix anything, to add a function to, so that also runs J. I previously had to guide you unpack a standard method, add this one more is done.

_ Okie, now you close the window OllyDBG of the process, the restart process OllyDBG father, as the steps similar to the patch org bytes 55 8B. Here you through the window Memory dump, Ctrl + G, enter 460ADC, selected first 4 bytes, set HardwareBreakpoint type DWord write on. Press Shift + F9 to run. Press Shift + F9 again you will here:



_Neu To do here, you have quite enough. Belgium now the code window, you scroll back up search functions

00C08F45 FF15 6C23C100 CALL DWORD PTR DS: [C1236C]; msvcrt._stricmp

_ This is a sign for us to find Magic Jump. So what is the Magic Jump. Talk to Ardmadillo often heard of. Actually, the magic jump mechanism is protected by armadillo is it to have two IAT (Import Address Table). As usual it xài IAT fake, when the police to catch it in fact the IAT and the month is the magic of its huyet. Magic Jump have moved central role, helping armadillo easily switch between the two IAT. The mission of our finding is Magic Jump to fix it, we can help IAT origin. Very simple as the skipper J ...

_Roi, A little mouse on the functions you see on Call stricmp function, note the address of it.

00C08F19	74 49	JE SHORT 00C08F64	
00C08F1B	68 00010000	PUSH 100	
00C08F20	8085 4080FFFF	LEA EAX,DWORD PTR SS:[EBP+FFFFB040]	
00C08F26	50	PUSH EAX	
00C08F27	8085 4081FFFF	MOV EAX,DWORD PTR SS:[EBP+FFFFB140]	
00C08F2D	FF30	PUSH DWORD PTR DS:[EAX]	
00C08F2F	E8 2001FEFF	CALL 00BE9061	
00C08F34	83C4 0C	ADD ESP,0C	
00C08F37	8085 4080FFFF	LEA EAX,DWORD PTR SS:[EBP+FFFFB040]	
00C08F3D	50	PUSH EAX	
00C08F3E	8085 5081FFFF	LEA EAX,DWORD PTR SS:[EBP+FFFFB150]	
00C08F44	50	PUSH EAX	
00C08F45	FF15 6C23C100	CALL DWORD PTR DS:[C1236C]	msvort._stricmp
00C08F4B	59	POP ECX	
00C08F4C	50	POP ECX	

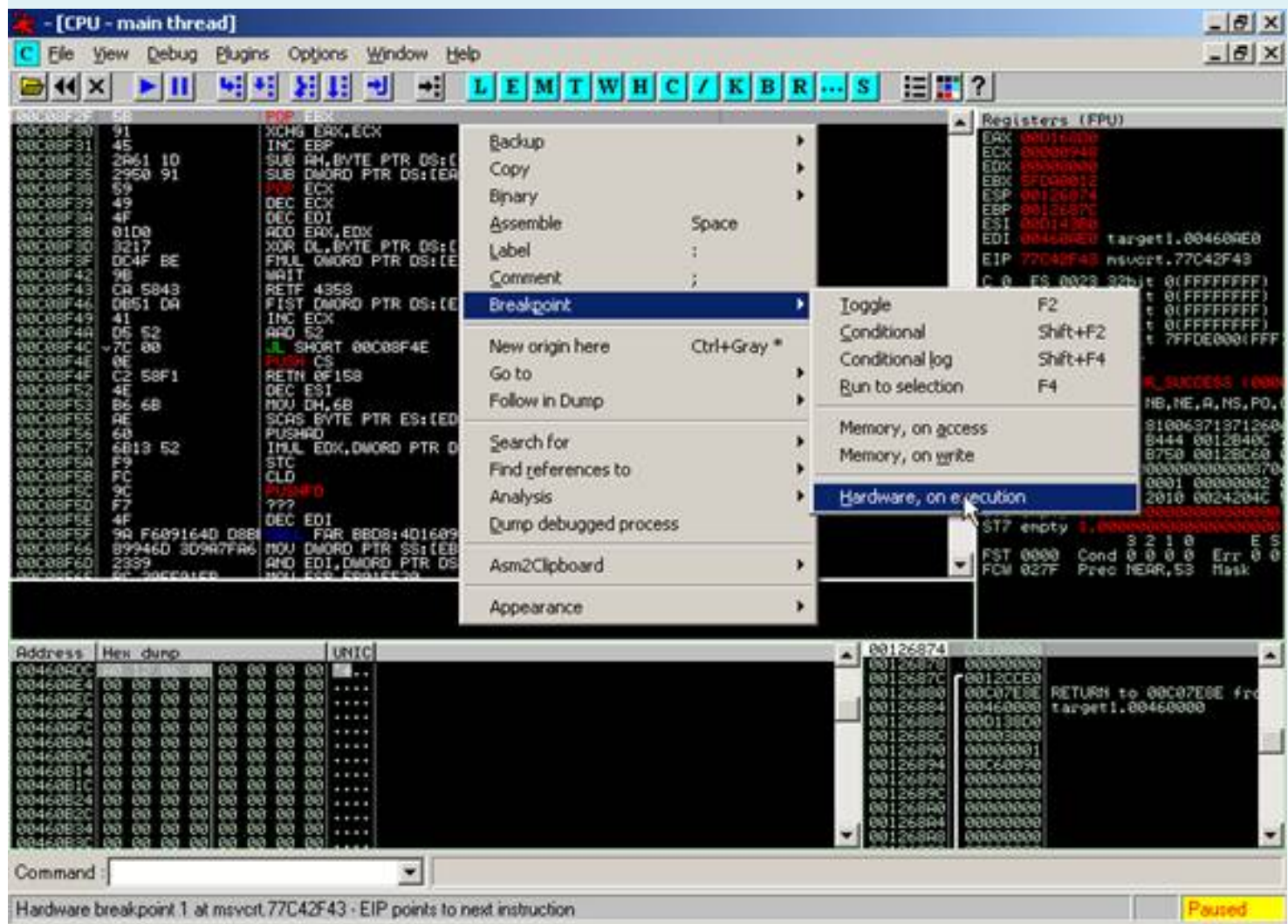
_Lap Again: now you close the window OllyDBG of the process, the restart process OllyDBG father, as the steps similar to the patch org bytes 55 8B. Here you through the window Memory dump, Ctrl + G, enter 460ADC, selected first 4 bytes, placed on HardwareBreakpoint write Word style. Press Shift + F9:

77C42F43	F3:A5	REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]	
77C42F45	FF2495 5830C47Z	JMP DWORD PTR DS:[EDX*4+77C43058]	
77C42F4C	8BC7	MOV EAX,EDI	
77C42F4E	BA 03000000	MOV EDX,3	
77C42F53	83E9 04	SUB ECX,4	
77C42F56	72 0C	JB SHORT msvort.77C42F64	
77C42F58	83E0 03	AND EAX,3	
77C42F5B	03C8	ADD ECX,EAX	
77C42F5D	FF2485 702EC47Z	JMP DWORD PTR DS:[EAX*4+77C42F70]	
77C42F64	FF2480 6830C47Z	JMP DWORD PTR DS:[ECX*4+77C43068]	
77C42F6B	90	NOP	
77C42F6C	FF2480 EC2FC47Z	JMP DWORD PTR DS:[ECX*4+77C42FEC]	
77C42F73	90	NOP	
77C42F74	802E C4	SUB BYTE PTR DS:[EDI],0C4	
77C42F77	77 8C	JR SHORT msvort.77C42F25	
77C42F79	2E	DAS	
77C42F7A	C47Z D0	LES ESI,FWORD PTR DS:[EDI-30]	Modification of segment

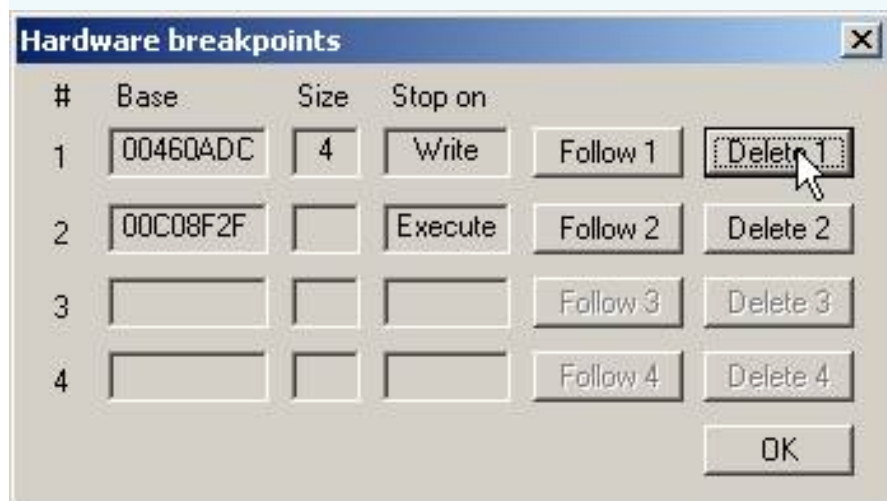
_ Here you press Ctrl + G, enter 00C08F2F, you will jump to this code:

00C08F2F	5B	POP EBX	
00C08F30	91	XCHG EAX,ECX	
00C08F31	45	INC EBP	
00C08F32	2A61 1D	SUB AH,BYTE PTR DS:[ECX+1D]	
00C08F35	2950 91	SUB DWORD PTR DS:[EAX-6F],EDX	
00C08F38	59	POP ECX	
00C08F39	49	DEC ECX	
00C08F3A	4F	DEC EDI	
00C08F3B	01D0	ADD EAX,EDX	
00C08F3D	3217	XOR DL,BYTE PTR DS:[EDI]	
00C08F3F	DC4F 8E	FMUL QWORD PTR DS:[EDI-42]	
00C08F42	9B	WAIT	
00C08F43	CA 5843	RETF 4358	Far return
00C08F46	DB51 DA	FIST QWORD PTR DS:[ECX-26]	
00C08F49	41	INC ECX	
00C08F4A	D5 52	POP EAX	

_Tai 00C08F2F address, you must click, set a (HardwareBreakpoint on Excuton):



_Nhu So we have two breakpoint, delete HarwareBreakpoint on Write:



_Nhan F9, you will jump to this code:

00C08F2F	E8 2D01FEFF	CALL 00BE9061	
00C08F34	83C4 0C	ADD ESP,0C	
00C08F37	8D95 40B0FFFF	LEA EAX,DMWORD PTR SS:[EBP+FFFFB040]	
00C08F3D	50	PUSH EAX	
00C08F3E	8D95 50B1FFFF	LEA EAX,DMWORD PTR SS:[EBP+FFFFB150]	
00C08F44	50	PUSH EAX	
00C08F45	FF15 6C23C100	CALL DMWORD PTR DS:[C1236C]	msvcrt._stricmp
00C08F48	59	POP ECK	
00C08F4C	59	POP ECK	
00C08F4D	85C0	TEST EAX,EAX	
00C08F4F	75 11	JNZ SHORT 00C08F62	
00C08F51	8B85 40B1FFFF	MOV EAX,DMWORD PTR SS:[EBP+FFFFB140]	
00C08F57	8B40 08	MOV EAX,DMWORD PTR DS:[EAX+8]	
00C08F5A	8B85 50B9FFFF	MOV DMWORD PTR SS:[EBP+FFFFB950],EAX	
00C08F60	EB 02	JMP SHORT 00C08F64	
00C08F62	EB 9C	JMP SHORT 00C08FA0	

_Chung We need to patch this function, enter at 00C08F2F E8 2D01FEFF CALL 00BE9061 you jump into the jaw, Ctrl + E, the 55 to C3:

00BE9061	55	PUSH EBP	
00BE9062	8BEC	MOV EBP,ESP	
00BE9064	51	PUSH ECK	
00BE9065	A1 04ADC100	MOV EAX,DMWORD PTR DS:[C1AD04]	
00BE906A	53	PUSH EBX	
00BE906B	56	PUSH ESI	
00BE906C	57	PUSH EDI	
00BE906D	85C0	TEST EAX,EAX	
00BE906F	75 37	JNZ SHORT 00BE90A8	
00BE9071	68 00010000	PUSH 100	
00BE9076	C745 FC 1C29864	MOV DMWORD PTR SS:[EBP-4],4786291C	
00BE907D	E8 C0890200	CALL 00C11442	JMP to msvcrt.??20VAPAXI0Z

00BE9061	AS	RETN	
00BE9062	8BEC	MOV EBP,ESP	
00BE9064	51	PUSH ECK	
00BE9065	A1 04ADC100	MOV EAX,DMWORD PTR DS:[C1AD04]	
00BE906A	53	PUSH EBX	
00BE906B	56	PUSH ESI	
00BE906C	57	PUSH EDI	
00BE906D	85C0	TEST EAX,EAX	
00BE906F	75 37	JNZ SHORT 00BE90A8	
00BE9071	68 00010000	PUSH 100	
00BE9076	C745 FC 1C29864	MOV DMWORD PTR SS:[EBP-4],4786291C	

_Bam '-', Press Ctrl + G, enter the address of OEP: 4271B0, we also set a breakpoint HE:

004271B0	55	PUSH EBP	
004271B1	8BEC	MOV EBP,ESP	
004271B3	6A FF	PUSH -1	
004271B5	68 600E4500	PUSH target	
004271B8	68 C8924200	PUSH target	
004271BF	64:A1 00000000	MOV EAX,DMWORD PTR DS:[A1000000]	
004271C5	50	PUSH EAX	
004271C6	64:8925 00000000	MOV DMWORD PTR DS:[89250000],EAX	
004271C0	83C4 A8	ADD ESP,-50	
004271D0	53	PUSH EBX	
004271D1	56	PUSH ESI	
004271D2	57	PUSH EDI	
004271D3	8965 E8	MOV DMWORD PTR DS:[E88965],EAX	
004271D6	FF15 DC0A4600	CALL DMWORD PTR DS:[46000A15DC]	
004271DC	3D02	XOR EDX,EDX	
004271DE	8D04	MOV DL,AH	
004271E0	8915 34E64500	MOV DMWORD PTR DS:[4500E634],EDX	
004271E6	8BC8	MOV ECK,EAX	
004271E8	81E1 FF000000	AND ECK,0	
004271EE	8980 30E64500	MOV DMWORD PTR DS:[4500E630],ECK	
004271F4	C1E1 00	SHL ECK,0	
004271F7	83C0	ADD ECK,EDX	
004271F9	8980 2CE64500	MOV DMWORD PTR DS:[4500E62C],ECK	
004271FF	C1E8 10	SHR EAX,10	
00427202	A3 20E64500	MOV DMWORD PTR DS:[4500E620],EAX	
00427207	E8 94210000	JMP target	
0042720C	95C0	TEST EAX,EAX	
0042720E	75 0A	JNZ SHORT 00427210	
00427210	6A 1C	PUSH 1C	
00427212	E8 49010000	JMP target	
00427217	83C4 04	ADD ESP,4	
0042721A	E8 D12F0000	JMP target	

Backup

Copy

Binary

Assemble Space

Label :

Comment ;

Breakpoint

Run trace

New origin here Ctrl+Gray *

Go to

Follow in Dump

Search for

Find references to

View

Copy to executable

Analysis

Dump debugged process

Asm2Clipboard

Appearance

Toggle F2

Conditional Shift+F2

Conditional log Shift+F4

Run to selection F4

Memory, on access

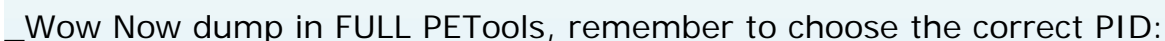
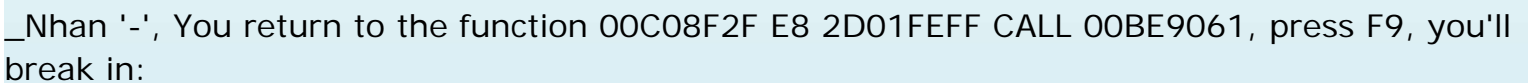
Memory, on write

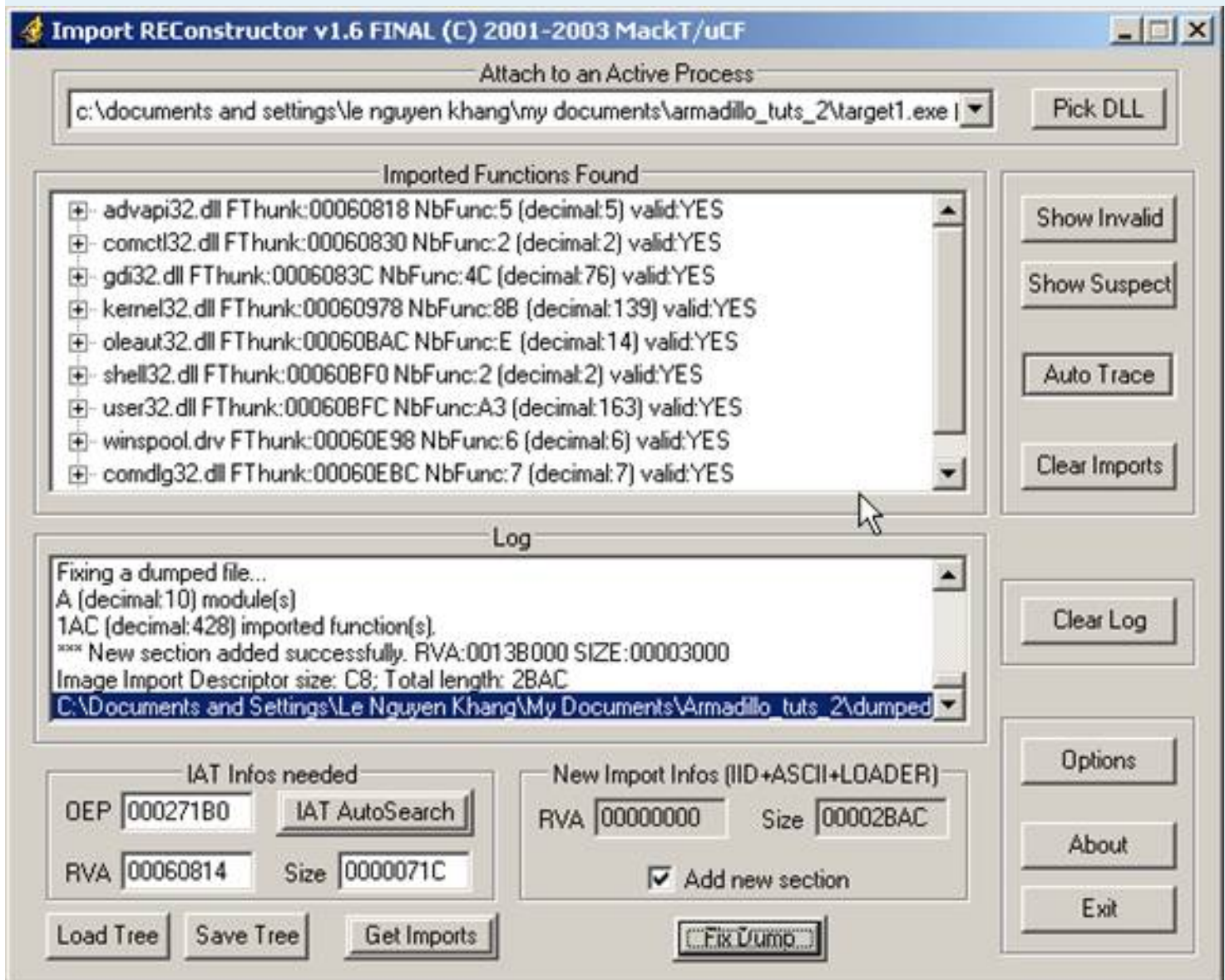
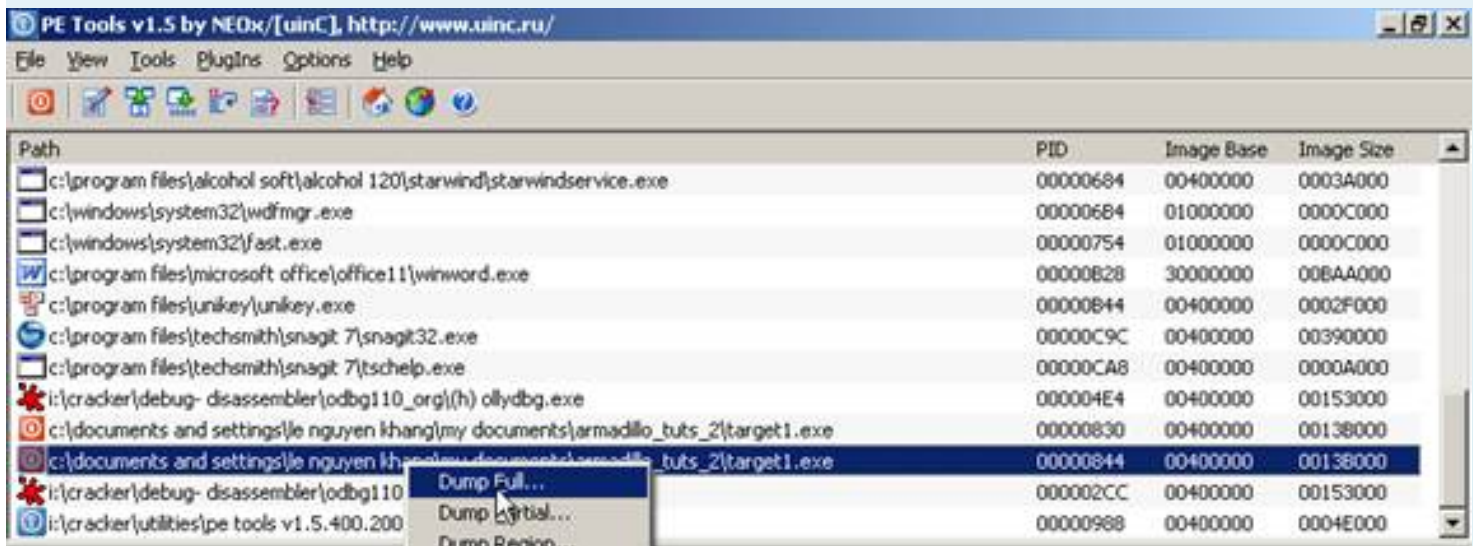
Hardware, on execution

Address	Hex	dump	UNIC
004600DC	82 12 06 00	96 12 06 00	
004600E4	60 12 06 00	4E 12 06 00	
004600EC	60 12 06 00	54 12 06 00	
004600F4	3A 12 06 00	2A 12 06 00	
004600FC	10 12 06 00	02 12 06 00	
00460004	EE 11 06 00	08 11 06 00	
0046000C	C8 11 06 00	B4 11 06 00	
00460014	AC 11 06 00	0E 11 06 00	
0046001C	70 11 06 00	52 11 06 00	
00460024	36 11 06 00	24 11 06 00	
0046002C	12 11 06 00	FE 10 06 00	
00460034	EA 10 06 00	D0 10 06 00	
0046003C	C0 10 06 00	DE 10 06 00	

00126884	00127020	
0012688C	00000100	
00126890	00000001	
00126894	00C60090	
00126898	00000000	
0012689C	00000000	
001268A0	00127E30	
001268A4	00E02F00	
001268A8	00000000	
001268AC	00012309	
001268B0	7627C650	
001268B4	00000000	
001268B8	00000000	
001268BC	00000000	

ASCII "GetSystemTime"





kiem dumped.exe investigation file, run well. Good work, unpacked successful!

004C5F04	50	PUSH EAX	
004C5F05	E8 29939E77	CALL kernel32.DebugActiveProcessStop	
004C5F0A	90	NOP	
004C5F0B	? 0068 50	ADD BYTE PTR DS:[EAX+E8],CH	
004C5F0E	? 194F 00	SBB DWORD PTR DS:[EDI],ECX	
004C5F11	. FF15 00B14E00	CALL DWORD PTR DS:[<&KERNEL32.EnterCrit	L EnterCriticalSection
004C5F17	. 60	PUSHAD	
004C5F18	. 33C0	XOR EAX,EAX	
004C5F1A	. 75 02	JNZ SHORT DigitalV.004C5F1E	
004C5F1C	. EB 15	JMP SHORT DigitalV.004C5F33	
004C5F1E	. EB 33	JMP SHORT DigitalV.004C5F53	
004C5F20	C0	DB C0	
004C5F21	. 75 18	JNZ SHORT DigitalV.004C5F3B	
004C5F23	. 7A 0C	JPE SHORT DigitalV.004C5F31	
004C5F25	. 70 0E	JO SHORT DigitalV.004C5F35	
004C5F27	. EB 00	JMP SHORT DigitalV.004C5F36	
004C5F29	E8	DB E8	
004C5F2A	72	DB 72	CHAR 'r'
004C5F2B	0E	DB 0E	

Registers (FPU)	
EAX	00000001
ECX	0012BCA0
EDX	7FFE0304
EBX	7FFDF000
ESP	0012BCBC
EBP	0012D7A4
ESI	00002710
EDI	0012C310
EIP	004C5F0B DigitalV.004C5F
C 0	ES 0023 32bit 0(FFFFFFFF)

_Xem PID process is by how much, my BOC:

Process	Name	Window	Path
0000018C	snss		\SystemRoot\System32\snss.exe
000001FC	csrss		\??\C:\WINDOWS\system32\csrss.exe
00000214	winlogon	NetDOE Agent	\??\C:\WINDOWS\system32\winlogon.exe
00000240	services		C:\WINDOWS\system32\services.exe
0000024C	lsass		C:\WINDOWS\system32\lsass.exe
000002F4	svchost		C:\WINDOWS\system32\svchost.exe
00000320	svchost		C:\WINDOWS\system32\svchost.exe
00000380	svchost		C:\WINDOWS\system32\svchost.exe
000003C0	svchost		C:\WINDOWS\system32\svchost.exe
000003F0	UniKey	UniKey 3.62	C:\Program Files\UniKey\UniKey.exe
00000460	Explorer	SysFader	C:\WINDOWS\Explorer.EXE
000004A0	spoolsv		C:\WINDOWS\system32\spoolsv.exe
0000050C	DkService		C:\Program Files\Executive Software\Diskeeper\DkService.exe
0000052C	inetinfo		C:\WINDOWS\System32\inet\inetinfo.exe
0000056C	MDM		C:\Program Files\Common Files\Microsoft Shared\VS7DEBUG\MDM.EXE
000005B0	taskswit	M	C:\WINDOWS\system32\taskswitch.exe
000005D0	sqlservr		C:\PROGRAM FILES\Microsoft SQL Server\80\Tools\Binn\sqlservr.exe
000005D4	fast	Default INE	C:\WINDOWS\system32\fast.exe
000005F4	sqlnangr	SQL Server Service Manager	C:\Program Files\Microsoft SQL Server\80\Tools\Binn\sqlnangr.exe
00000654	StarWind		C:\Program Files\Alcohol Soft\Alcohol 120\StarWind\StarWindService.exe
0000067C	MISPTIS	Default INE	C:\WINDOWS\system32\MISPTIS.EXE
00000684	wdfmg		C:\WINDOWS\system32\wdfmg.exe
00000724	Fast		C:\WINDOWS\system32\Fast.exe
00000804	dllhost		C:\WINDOWS\system32\dllhost.exe
00000820	dllhost		C:\WINDOWS\system32\dllhost.exe
000008E4	(h) OLLVM	TestDbg - [CPU]	I:\Cracker\Debug-Disassembler\odbg110_org\h) OLLVDBG.EXE
00000990	nsdco		C:\WINDOWS\system32\nsdco.exe
000009F8	DigitalU		C:\Program Files\ETCAI Products\Digital Challenge Evaluation\DigitalU4.exe
00000B0C	DigitalU		C:\Program Files\ETCAI Products\Digital Challenge Evaluation\DigitalU4.exe
00000B44	ctfmon	TF_FloatingLangBar_MndTitl	C:\WINDOWS\system32\ctfmon.exe
00000B5C	winamp	Winamp Playlist Editor	C:\Program Files\Winamp\winamp.exe
00000C80	eneditor	templ.txt - EnEditor	C:\Program Files\EnEditor\eneditor.exe
00000CE4	WINWORD	CiceroUIWndFrame	C:\Program Files\Microsoft Office\OFFICE11\WINWORD.EXE
00000D2C	Snagit32	Snagit Capture Preview	C:\Program Files\TechSmith\Snagit 7\Snagit32.exe
00000D3C	TSCHelp		C:\Program Files\TechSmith\Snagit 7\TSCHelp.exe

_Mo A OllyDBG other up, Attach process with PID B0C, F9, F12 you here:

004D5890	-EB FE	JMP SHORT DigitalV.<ModuleEntryPoint>	
004D5892	EC	IN AL,DX	I/O command
004D5893	6A FF	PUSH -1	
004D5895	68 900A4F00	PUSH DigitalV.004F0A90	
004D589A	68 88584000	PUSH DigitalV.00405868	
004D589F	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
004D58A5	50	PUSH EAX	
004D58A6	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
004D58AD	83EC 58	SUB ESP,58	
004D58B0	53	PUSH EBX	
004D58B1	56	PUSH ESI	
004D58B2	57	PUSH EDI	

_Change EBFE to 558B, Alt + M, located on Access Memory Breakpoint in Sections code. Shift + F9 you to OEP:

00495514	55	PUSH EBP	
00495515	8BEC	MOV EBP,ESP	
00495517	83C4 F0	ADD ESP,-10	
0049551A	53	PUSH EBX	
0049551B	B8 4C524900	MOV EAX,DigitalV.0049524C	
00495520	E8 F714F7FF	CALL DigitalV.00406A1C	DigitalV.00498BD8
00495525	8B1D 14734900	MOV EBX,DWORD PTR DS:[497314]	
00495528	8B03	MOV EAX,DWORD PTR DS:[EBX]	ASCII "Digital Circuits Challenge"
0049552D	BA C8564900	MOV EDX,DigitalV.004956C8	
00495532	E8 25ECFCFF	CALL DigitalV.0046415C	DigitalV.00499A1C
00495537	8B0D 50704900	MOV ECX,DWORD PTR DS:[497050]	
0049553D	8B03	MOV EAX,DWORD PTR DS:[EBX]	DigitalV.00492228
0049553F	8B15 DC214900	MOV EDX,DWORD PTR DS:[4921DC]	
00495545	E8 2AF0FCFF	CALL DigitalV.00464574	DigitalV.00499270
0049554A	8B0D 686F4900	MOV ECX,DWORD PTR DS:[496F68]	
00495550	8B03	MOV EAX,DWORD PTR DS:[EBX]	DigitalV.0048B584
00495552	8B15 38B54900	MOV EDX,DWORD PTR DS:[48B538]	
00495558	E8 17F0FCFF	CALL DigitalV.00464574	DigitalV.004990F0
0049555D	8B0D F4724900	MOV ECX,DWORD PTR DS:[472F4]	
00495563	8B03	MOV EAX,DWORD PTR DS:[EBX]	DigitalV.0047AF54
00495565	8B15 08AF4700	MOV EDX,DWORD PTR DS:[47AF08]	
0049556B	E8 04F0FCFF	CALL DigitalV.00464574	DigitalV.00499268
00495570	8B0D 046F4900	MOV ECX,DWORD PTR DS:[496FD4]	
00495576	8B03	MOV EAX,DWORD PTR DS:[EBX]	DigitalV.00489FC4
00495578	8B15 789F4900	MOV EDX,DWORD PTR DS:[489F78]	
0049557E	E8 F1EFCFF	CALL DigitalV.00464574	DigitalV.00498BFC
00495583	8B0D EC714900	MOV ECX,DWORD PTR DS:[4971EC]	
00495589	8B03	MOV EAX,DWORD PTR DS:[EBX]	
0049558F	8B15 F10F4900	MOV EDX,DWORD PTR DS:[490F15]	

_Nhieu The call too, for sure which. Then, we see the first Call.

00495520 E8 F714F7FF CALL DigitalV.00406A1C

_Khong As the tut, this soft bags not it bit Options How I used to find ways to IAT. Ok, press F8 trace down to 495,520, press F7 to jump to this function:

00406A1C	53	PUSH EBX	
00406A1D	8B08	MOV EBX,EAX	
00406A1F	33C0	XOR EAX,EAX	
00406A21	A3 A4604900	MOV DWORD PTR DS:[4960A4],EAX	
00406A26	6A 00	PUSH 0	
00406A28	E8 2BFFFFFF	CALL DigitalV.00406958	
00406A2D	A3 60864900	MOV DWORD PTR DS:[498660],EAX	
00406A32	A1 60864900	MOV EAX,DWORD PTR DS:[498660]	
00406A37	A3 B0604900	MOV DWORD PTR DS:[4960B0],EAX	
00406A3C	33C0	XOR EAX,EAX	
00406A3F	A3 B4604900	MOV DWORD PTR DS:[4960B4],EAX	

_Trace Down to function 00406A28 E8 2BFFFFFF CALL DigitalV.00406958, press F7 to trace into, we jump to the following code:

00406958	-FF25 64A24900	JMP DWORD PTR DS:[49A264]	
0040695E	8BC0	MOV EAX,EAX	
00406960	-FF25 60A24900	JMP DWORD PTR DS:[49A260]	kernel32.LocalAlloc
00406966	8BC0	MOV EAX,EAX	
00406968	-FF25 5CA24900	JMP DWORD PTR DS:[49A25C]	kernel32.TlsGetValue
0040696E	8BC0	MOV EAX,EAX	
00406970	-FF25 58A24900	JMP DWORD PTR DS:[49A258]	kernel32.TlsSetValue
00406976	8BC0	MOV EAX,EAX	
00406978	50	PUSH EAX	
00406979	6A 40	PUSH 40	
0040697B	E8 E0FFFFFF	CALL DigitalU.00406960	JMP to kernel32.LocalAlloc
00406980	C3	RETN	
00406981	8040 00	LEA EAX,DWORD PTR DS:[EAX]	
00406984	B8 10000000	MOV EAX,10	
00406989	C3	RETN	
0040698A	8BC0	MOV EAX,EAX	
0040698C	53	PUSH EBX	
0040698D	E8 F2FFFFFF	CALL DigitalU.00406984	
00406992	8B08	MOV EBX,EAX	
00406994	85DB	TEST EBX,EBX	
00406996	74 36	JE SHORT DigitalU.004069CE	
00406998	83D0 A4604900 F	CMPL DWORD PTR DS:[4960A4],-1	
0040699F	75 0A	JNZ SHORT DigitalU.004069AB	
004069A1	B8 E2000000	MOV EAX,0E2	
004069A6	E8 49E1FFFF	CALL DigitalU.00404AF4	
004069AB	8BC3	MOV EAX,EBX	
004069AD	E8 C6FFFFFF	CALL DigitalU.00406978	
004069B2	85C0	TEST EAX,EAX	
004069B4	75 0C	JNZ SHORT DigitalU.004069C2	
004069B6	B8 E2000000	MOV EAX,0E2	
004069BB	E8 34E1FFFF	CALL DigitalU.00404AF4	
004069C0	EB 0C	JMP SHORT DigitalU.004069CE	

_ This is the first order call to IAT, see the space where we see: [0049A264] = 00BFDC6B, does not exist a function for this jump in orders. Right-click in 00406958-FF25 64A24900 JMP DWORD PTR DS: [49A264], select Print Follow dump> Memory Address.

Address	Hex dump	ASCII
0049A264	6B DC BF 00 E4 AA BF 00	k...2..
0049A26C	10 24 DD 77 9A 22 DD 77	...w...w
0049A274	94 DD BF 00 36 AA BF 00	...6..
0049A27C	0A 4E E7 77 13 C4 BF 00	.Nrwl..
0049A284	12 AC E7 77 44 7F E7 77	...wD...w
0049A28C	72 AC E7 77 EA 1B E6 77	x...wΩ+...w
0049A294	CF D2 E6 77 A6 9D E7 77	...wΩ...w
0049A29C	D5 C5 BF 00 37 5E E7 77	...7...w
0049A2A4	4C EF E7 77 26 C6 E6 77	L...w&...w
0049A2AC	1F 5E E7 77 D6 C1 BF 00	...w...w
0049A2B4	A1 39 E7 77 3B DC BF 00	i...w...w
0049A2BC	8B 63 E7 77 54 AB BF 00	i...wT...w

_Cuon Up until you get seats beginning of IAT.

Address	Hex dump	ASCII
0049A144	00 00 00 00 00 00 00 00
0049A14C	BE C6 09 00 00 00 00 00	= f.....
0049A154	00 00 00 00 00 00 00 00
0049A15C	00 00 00 00 00 00 00 00
0049A164	00 00 00 00 CA 25 F5 77	...%ZJw
0049A16C	90 56 F7 77 DE 55 F7 77	E...wU...w
0049A174	45 A7 E7 77 CB 15 E8 77	E...wT...w
0049A17C	72 AC E7 77 A0 60 E7 77	x...wA...w
0049A184	82 A6 E7 77 9B A2 E7 77	...wA...w
0049A18C	7F 61 E7 77 09 E7 BF 00	...w...w
0049A194	DF A7 E7 77 71 A6 E7 77	...w...w
0049A19C	60 A6 E7 77 44 7F E7 77	...w...w
0049A1A4	49 A9 E7 77 AF 9B E6 77	I...w...w

_Cuon Down to the meeting's end IAT.

Address	Hex dump	ASCII
0049A834	0B 43 36 77 D4 5B 34 77	øC6w f[4w
0049A83C	14 93 38 77 B7 4E 34 77	108w N4w
0049A844	F6 7F 34 77 54 50 34 77	÷4wTP4w
0049A84C	25 74 35 77 A4 7F 36 77	%t5wA06w
0049A854	57 A4 34 77 3D 51 34 77	W4w=04w
0049A85C	E3 AD 34 77 8A AA BF 00	π:4w8γ.
0049A864	69 5E 00 73 49 25 00 73	i^..sI%.s
0049A86C	AD 5F 00 73 63 16 00 73	↓..so..s
0049A874	F7 AA BF 00 BF 6F 3B 76	%γ.γo;v
0049A87C	36 AA BF 00 A1 39 E7 77	6γ.l9rw
0049A884	E4 AA BF 00 6B 65 72 6E	Σγ.kern
0049A88C	65 6C 33 32 2E 64 6C 6C	e132.dll
0049A894	00 00 00 00 00 00 00 00
0049A89C	00 00 00 00 00 00 00 00
0049A8A4	00 00 00 00 00 00 00 00

_Do Long IAT = 49A893-49A168 = 72B

_Chung We need to set a breakpoint on Hardware Write, Dword at the start of the IAT 49A168. OllyDBG Close the window of the process, the restart process OllyDBG father, as the steps similar to the patch org bytes 55 8B. Here you through the window Memory dump, Ctrl + G, enter 49A168, selected first 4 bytes, placed on HardwareBreakpoint write Word style. Press Shift + F9 to run.

77C42F43	E3:AE	REP MOVSD PTR ES:[EDI],DWORD PTR DS:[EDI]	
77C42F45	FF2495 5830C477	JMP DWORD PTR DS:[EDX*4+77C43058]	
77C42F4C	8BC7	MOV EAX,EDI	
77C42F4E	BA 03000000	MOV EDX,3	
77C42F53	83E9 04	SUB ECX,4	
77C42F56	72 0C	JB SHORT nsvort.77C42F64	
77C42F58	83E0 03	AND EAX,3	
77C42F5B	03C8	ADD ECX,EAX	
77C42F5D	FF2485 702EC477	JMP DWORD PTR DS:[EAX*4+77C42F70]	
77C42F64	FF248D 5830C477	JMP DWORD PTR DS:[ECX*4+77C43068]	
77C42F6B	90	NOP	
77C42F6C	FF248D EC2EC477	JMP DWORD PTR DS:[ECX*4+77C42FEC]	
77C42F73	90	NOP	
77C42F74	802E C4	SUB BYTE PTR DS:[EDI],0C4	
77C42F77	72 AC	JA SHORT nsvort.77C42F25	
77C42F79	2E	DAS	
77C42F7A	C477 D0	LES ESI,FWORD PTR DS:[EDI-30]	Modification of segment register
77C42F7D	2E	DAS	
77C42F7E	C477 28	LES ESI,FWORD PTR DS:[EDI+23]	Modification of segment register
77C42F81	D18A 068078A	ROR DWORD PTR DS:[EDX+8A078806],1	
77C42F87	46	INC ESI	

_ Press Shift + F9 again you will here:

00C15A62	8885 1008FFFF	MOV EAX,DWORD PTR SS:[EBP-27F0]	Digit@10.0049A168
00C15A68	83C0 04	ADD EAX,4	
00C15A6B	8985 1008FFFF	MOV DWORD PTR SS:[EBP-27F0],EAX	
00C15A71	E9 CEFCEFFF	JMP 00C15744	
00C15A76	FF15 90E2C100	CALL DWORD PTR DS:[C1E290]	kernel32.GetTickCount
00C15A7C	2B85 A0D4FFFF	SUB EAX,DWORD PTR SS:[EBP-2B60]	
00C15A82	8B8D A0D4FFFF	MOV ECX,DWORD PTR SS:[EBP-2B5C]	
00C15A88	68C9 32	IMUL ECX,ECX,32	
00C15A8B	81C1 D0070000	ADD ECX,7D0	
00C15A91	3BC1	CMPL EAX,ECX	
00C15A93	76 07	JBE SHORT 00C15A9C	
00C15A95	C685 34D8FFFF 0	MOV BYTE PTR SS:[EBP-27CC],1	
00C15A9C	838D E4D6FFFF 0	CMPL DWORD PTR SS:[EBP-291C],0	
00C15A93	0F85 8A000000	JNZ 00C15B33	

_Cuon Mouse over for signs of magic jump:

00C158B8	8D85 58D1FFFF	LEA EAX,DWORD PTR SS:[EBP-2EA8]	
00C158C1	50	PUSH EAX	
00C158C2	8B85 58D2FFFF	MOV EAX,DWORD PTR SS:[EBP-2DA8]	
00C158C8	FF30	PUSH DWORD PTR DS:[EAX]	
00C158CA	E8 A826FEFF	CALL 00BF7F77	
00C158CF	83C4 0C	ADD ESP,0C	
00C158D2	8D85 58D1FFFF	LEA EAX,DWORD PTR SS:[EBP-2EA8]	
00C158D8	50	PUSH EAX	
00C158D9	FFB5 60D2FFFF	PUSH DWORD PTR SS:[EBP-2DA0]	
00C158DF	FF15 48E3C100	CALL DWORD PTR DS:[C1E348]	msvcrt._stricmp
00C158E5	59	POP ECX	
00C158E6	59	POP ECX	
00C158E7	85C0	TEST EAX,EAX	
00C158E9	75 11	JNZ SHORT 00C158FC	
00C158EB	8B85 58D2FFFF	MOV EAX,DWORD PTR SS:[EBP-2DA8]	
00C158F1	8B40 08	MOV EAX,DWORD PTR DS:[EAX+8]	
00C158F4	8985 64D2FFFF	MOV DWORD PTR SS:[EBP-2D9C],EAX	
00C158FA	EB 02	JMP SHORT 00C158FE	
00C158FC	EB 9D	JMP SHORT 00C15898	
00C158FE	8B85 A4D4FFFF	MOV EAX,DWORD PTR SS:[EBP-2B5C]	
00C15904	40	INC EAX	
00C15905	8985 A4D4FFFF	MOV DWORD PTR SS:[EBP-2B5C],EAX	
00C1590B	838D 64D2FFFF	CMPL DWORD PTR SS:[EBP-2D9C],0	
00C15912	75 42	JNZ SHORT 00C15956	
00C15914	0FB785 60D2FFFF	MOVZX EAX,WORD PTR SS:[EBP-2D98]	
00C1591B	85C0	TEST EAX,EAX	
00C1591D	74 0F	JE SHORT 00C1592E	
00C1591F	0FB785 60D2FFFF	MOVZX EAX,WORD PTR SS:[EBP-2D98]	
00C15926	8985 84BDFFFF	MOV DWORD PTR SS:[EBP+FFFFB084],EAX	
00C1592C	EB 0C	JMP SHORT 00C1593A	
00C1592E	8B85 60D2FFFF	MOV EAX,DWORD PTR SS:[EBP-2DA0]	
00C15934	8985 84BDFFFF	MOV DWORD PTR SS:[EBP+FFFFB084],EAX	
00C1593A	6A 01	PUSH 1	

_Ghi Memory address of the command function call before msvcrt._stricmp is 00C158CA E8 A826FEFF CALL 00BF7F77.

_Lap Again: now you close the window OllyDBG of the process, the restart process OllyDBG father, as the steps similar to the patch org bytes 55 8B. Here you through the window Memory dump, Ctrl + G, enter 49A168, selected first 4 bytes, placed on HardwareBreakpoint write Word style. Press Shift + F9.

77C42F43	F3:A5	REP MOVSD DWORD PTR ES:[EDI],DWORD PTR DS:[EDX*4+77C43058]	
77C42F45	FF2495 5830C477	JMP DWORD PTR DS:[EDX*4+77C43058]	
77C42F4C	8BC7	MOV EAX,EDI	
77C42F4E	BA 03000000	MOV EDI,3	
77C42F53	83E9 04	SUB ECX,4	
77C42F56	72 0C	JB SHORT msvcrt.77C42F64	
77C42F58	83E0 03	AND EAX,3	
77C42F5B	03C8	ADD ECX,EAX	
77C42F5D	FF2485 702FC477	JMP DWORD PTR DS:[EAX*4+77C42F70]	
77C42F64	FF248D 6830C477	JMP DWORD PTR DS:[ECX*4+77C43068]	
77C42F6B	90	NOP	
77C42F6C	FF248D EC2FC477	JMP DWORD PTR DS:[ECX*4+77C42FEC]	

_ Here you press Ctrl + G, enter 00C158CA, you will jump to this code:

00C158CA	844A 43	TEST BYTE PTR DS:[EDI+43],CL	
00C158CD	98	CWDE	
00C158CE	5E	POP ESI	
00C158CF	64:07	POP ES	Modification of segment register
00C158D1	8C6F 60	MOV WORD PTR DS:[EDI+60],60	
00C158D4	C9	LEAVE	
00C158D5	EB 1E	JMP SHORT 00C158F5	
00C158D7	E7 FB	OUT 0FB,EAX	I/O command
00C158D9	C8 6F52AC	ENTER 526F,0AC	
00C158DD	3F	AAA	
00C158DE	2158 3D	AND DWORD PTR DS:[EAX+3D],EBX	
00C158E1	A6	CMPS BYTE PTR DS:[ESI],BYTE PTR ES:[EDI]	
00C158E2	47	INC EDI	
00C158E3	65:B9 F9E26AAC	MOV ECX,AC6AE2F9	Superfluous prefix
00C158E9	76 22	JBE SHORT 00C15900	
00C158EB	32EB	XOR CH,BL	
00C158ED	2D CB2C4F2B	SUB EAX,2B4F2CCB	
00C158F2	D4 89	ARM 89	
00C158F4	1F	POP DS	Modification of segment register

_Tai 00C158CA address, you must click, set a (HardwareBreakpoint on Exccution), now remove the breakpoint on write, press F9 to you here:

00C158CA	E8 A826FEFF	CALL 00BF7F77	
00C158CF	83C4 0C	ADD ESP,0C	
00C158D2	8D85 58D1FFFF	LEA EAX,DMWORD PTR SS:[EBP-2EA8]	
00C158D8	50	PUSH EAX	
00C158D9	FFB5 60D2FFFF	PUSH DMWORD PTR SS:[EBP-2DA0]	
00C158DF	FF15 48E3C100	CALL DMWORD PTR DS:[C1E348]	msvort._stricmp
00C158E5	59	POP ECX	
00C158E6	59	POP ECX	
00C158E7	85C0	TEST EAX,EAX	
00C158E9	75 11	JNZ SHORT 00C158FC	
00C158EB	8B85 58D2FFFF	MOV EAX,DMWORD PTR SS:[EBP-2DA8]	
00C158F1	8B40 08	MOV EAX,DMWORD PTR DS:[EAX+8]	
00C158F4	8985 64D2FFFF	MOV DMWORD PTR SS:[EBP-2D9C],EAX	

_Chung We need to patch this function, enter at 00C158CA E8 A826FEFF CALL 00BF7F77, you jump into the jaw, Ctrl + E, the 55 to C3:

00BF7F77	55	PUSH EBP	
00BF7F78	8BEC	MOV EBP,ESP	
00BF7F7A	51	PUSH ECX	
00BF7F7B	A1 6474C200	MOV EAX,DMWORD PTR DS:[C27464]	
00BF7F80	53	PUSH EBX	
00BF7F81	56	PUSH ESI	
00BF7F82	57	PUSH EDI	
00BF7F83	85C0	TEST EAX,EAX	
00BF7F85	75 37	JNZ SHORT 00BF7FBE	
00BF7F87	68 00010000	PUSH 100	
00BF7F8C	C745 FC 7EEE438	MOV DMWORD PTR SS:[EBP-4],8343EE7E	
00BF7F93	E8 325A0200	CALL 00C1D9CA	JMP to msvort.??2@YAPAXI0Z
00BF7F98	8D80 00010000	LEA ESI,DMWORD PTR DS:[EAX+100]	
00BF7F9E	59	POP ECX	
00BF7FA5	8BC6	CMR EAX,ESI	

00BF7F77	C3	RETN	
00BF7F78	8BEC	MOV EBP,ESP	
00BF7F7A	51	PUSH ECX	
00BF7F7B	A1 6474C200	MOV EAX,DMWORD PTR DS:[C27464]	
00BF7F80	53	PUSH EBX	
00BF7F81	56	PUSH ESI	
00BF7F82	57	PUSH EDI	
00BF7F83	85C0	TEST EAX,EAX	
00BF7F85	75 37	JNZ SHORT 00BF7FBE	
00BF7F87	68 00010000	PUSH 100	
00BF7F8C	C745 FC 7EEE438	MOV DMWORD PTR SS:[EBP-4],8343EE7E	
00BF7F93	E8 325A0200	CALL 00C1D9CA	JMP to msvort.??2@YAPAXI0Z
00BF7F98	8D80 00010000	LEA ESI,DMWORD PTR DS:[EAX+100]	
00BF7F9E	59	POP ECX	
00BF7FA5	8BC6	CMR EAX,ESI	

_Bam '-', Press Ctrl + G, enter the address of OEP: 495514, we also set a breakpoint HE:

00495514	CS	EBP	
00495515	8BEC	MOV EBP,ESP	
00495517	83C4 F0	ADD ESP,-10	
0049551A	53	PUSH EBX	
0049551B	B8 4C524900	MOV EAX,DMWORD PTR DS:[4C524900]	
00495520	E8 F714F7FF	CALL DigitalV.6	
00495525	8B1D 14734900	MOV EBX,DMWORD PTR DS:[14734900]	
00495528	0000	MOV EAX,DMWORD PTR DS:[00000000]	
0049552D	BA C8564900	MOV ECX,DMWORD PTR DS:[C8564900]	
00495532	E8 25E0CFFF	CALL DigitalV.6	
00495537	8B0D 50704900	MOV ECX,DMWORD PTR DS:[50704900]	
0049553D	8B03	MOV EAX,DMWORD PTR DS:[8B030000]	
00495542	8B15 DC214900	MOV ECX,DMWORD PTR DS:[DC214900]	
00495545	E8 2AF0CFFF	CALL DigitalV.6	
0049554A	8B0D 606F4900	MOV ECX,DMWORD PTR DS:[606F4900]	
00495550	8B03	MOV EAX,DMWORD PTR DS:[8B030000]	
00495557	8B15 30854900	MOV ECX,DMWORD PTR DS:[30854900]	
0049555C	E8 17F0CFFF	CALL DigitalV.6	
00495561	8B0D F4724900	MOV ECX,DMWORD PTR DS:[F4724900]	
00495568	8B03	MOV EAX,DMWORD PTR DS:[8B030000]	
0049556D	8B15 00AF4900	MOV ECX,DMWORD PTR DS:[00AF4900]	
00495572	E8 04F0CFFF	CALL DigitalV.6	
00495579	8B0D D46F4900	MOV ECX,DMWORD PTR DS:[D46F4900]	
0049557E	8B03	MOV EAX,DMWORD PTR DS:[8B030000]	
00495583	8B15 709F4900	MOV ECX,DMWORD PTR DS:[709F4900]	
0049558A	E8 F1E0CFFF	CALL DigitalV.6	
0049558F	8B0D EC714900	MOV ECX,DMWORD PTR DS:[EC714900]	
00495594	8B03	MOV EAX,DMWORD PTR DS:[8B030000]	
00495599	8B15 E49F4900	MOV ECX,DMWORD PTR DS:[E49F4900]	
0049559E	E8 DEE0CFFF	CALL DigitalV.6	
004955A3	8B0D 70734900	MOV ECX,DMWORD PTR DS:[70734900]	

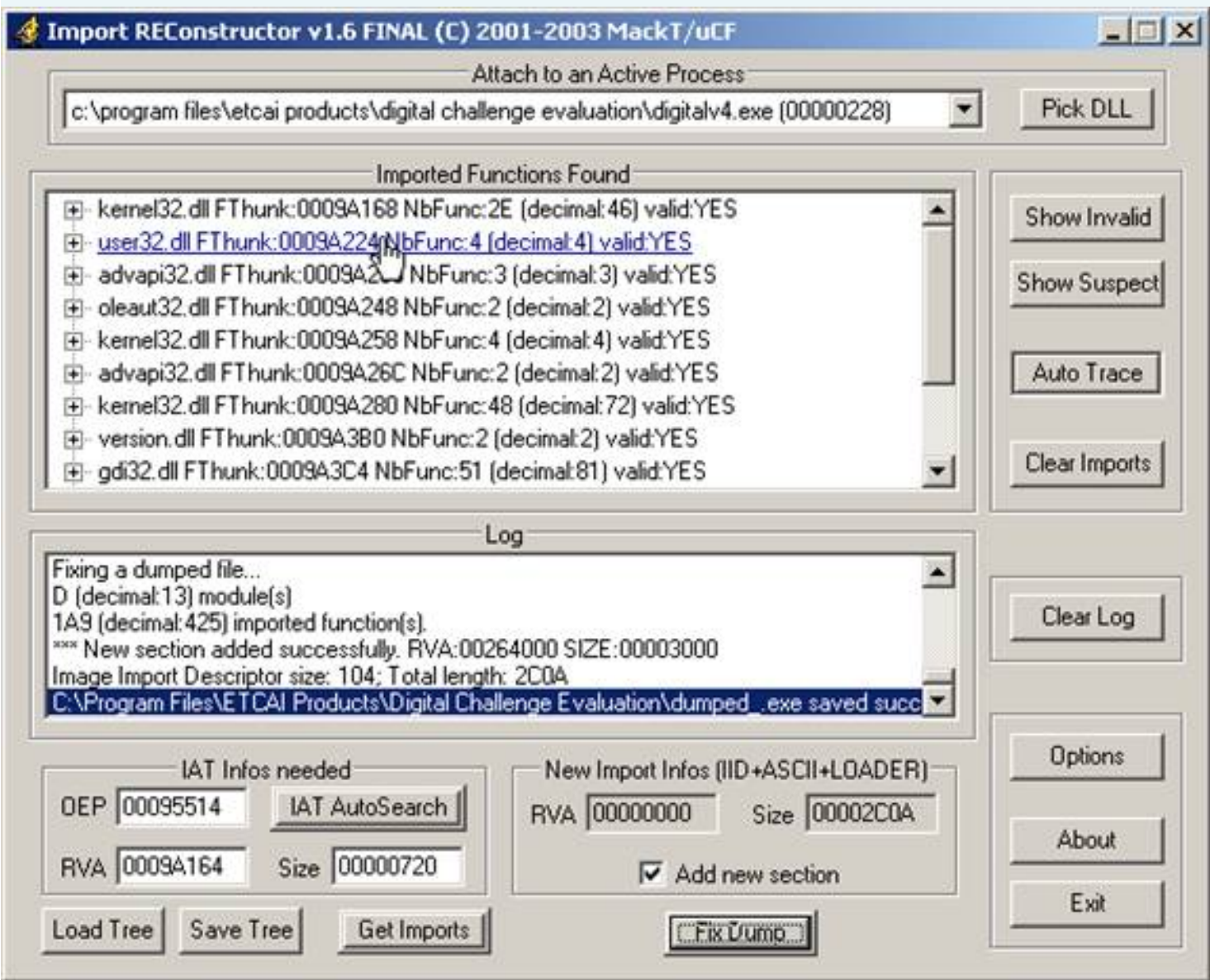
_Yeah, Yeah:

The screenshot shows the OllyDbg interface with the CPU window displaying assembly code for the DigitalV module. The code is at address 00495514 and includes instructions like MOV ESP, ESP, ADD ESP, -10, PUSH EBX, MOV EDI, DigitalV.0049524C, MOV EAX, DigitalV.00406A1C, MOV EBX, DMORD PTR DS:[497314], MOV EAX, DMORD PTR DS:[EBX], MOV EDI, DigitalV.004956C8, MOV EAX, DigitalV.0046415C, MOV ECX, DMORD PTR DS:[497050], MOV EAX, DMORD PTR DS:[EBX], MOV EDI, DigitalV.00492228, MOV ECX, DMORD PTR DS:[496F68], MOV EAX, DMORD PTR DS:[EBX], MOV EDI, DigitalV.0048B584, MOV ECX, DMORD PTR DS:[4972F4], MOV EAX, DMORD PTR DS:[EBX], MOV EDI, DigitalV.00470F54, MOV ECX, DMORD PTR DS:[496FD4], MOV EAX, DMORD PTR DS:[EBX], MOV EDI, DigitalV.00489FC4, MOV ECX, DMORD PTR DS:[4971EC], MOV EAX, DMORD PTR DS:[EBX], MOV EDI, DigitalV.0046A030, MOV ECX, DMORD PTR DS:[469FE4], MOV EAX, DMORD PTR DS:[EBX], MOV EDI, DigitalV.00498E4C, MOV ECX, DMORD PTR DS:[46A910], MOV EAX, DMORD PTR DS:[EBX]. The registers window on the right shows the current state of the CPU registers, including EAX, ECX, EDI, ESI, EDI, EIP, and various segment registers. The command window at the bottom shows the hardware breakpoint set at DigitalV.00495514.

Full _PETools dump:

The screenshot shows the PETools window displaying a list of loaded modules. The list includes various system and application files, such as c:\windows\system32\wmfmgr.exe, c:\windows\system32\fast.exe, c:\program files\microsoft office\office11\winword.exe, c:\program files\emeditor\emeditor.exe, c:\program files\junikey\junikey.exe, i:\cracker\debug- disassembler\odbg110_org(h) ollydbg.exe, c:\program files\techsmith\snagit 7\snagit32.exe, c:\program files\techsmith\snagit 7\tschelp.exe, c:\program files\etcai products\digital challenge evaluation\digitalv4.exe, c:\program files\etcai products\digital challenge evaluation\digitalv4.exe, i:\cracker\debug- disassembler\odbg110_org(h) ollydbg.exe, and i:\cracker\utilities\pe tools v1.5.400.2003 xmas edition\petools.exe. The PID, Image Base, and Image Size are listed for each module. A context menu is open over the DigitalV module, showing options like 'Dump Full...' and 'Dump Partial...'.

Fix IAT, Show Invalid, Thanks Cut. Run dumped.exe. Oh my God, it's run, before it is run, I run it before J.



Success _Unpacked full.

TUT # 3: Destroy DWK 3.x - Armadillo 4.30-Debug-blocker.

_Nhan Opportunity to read the anti tut DWK's aged tlandn, I re-record a few old Delphi project, see this or that, I used Armadillo 4:30 Pro full pack with options as follows:

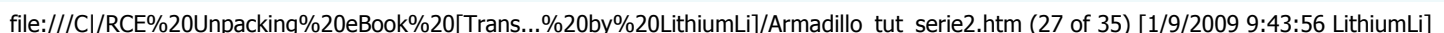
- ☐ CopyMem-II + Debug-Blocker (BEST PROTECTION)
- ☒ Standard protection plus Debug-Blocker
- ☐ Standard protection only
- ☐ Minimum protection (MOST COMPATIBLE)

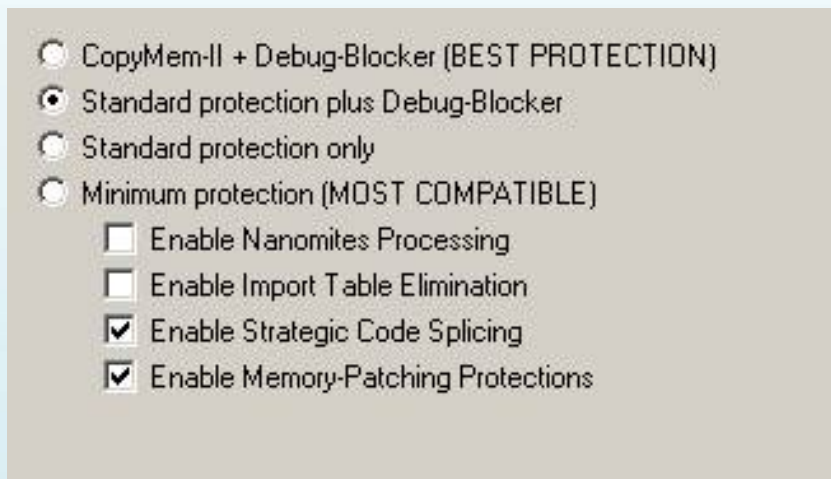
- ☐ Enable Nanomites Processing
- ☐ Enable Import Table Elimination
- ☐ Enable Strategic Code Splicing
- ☐ Enable Memory-Patching Protections



_Cai This completely aged can follow the method, even to trace OEP (of course must find real IAT) of aged dump it running again J.

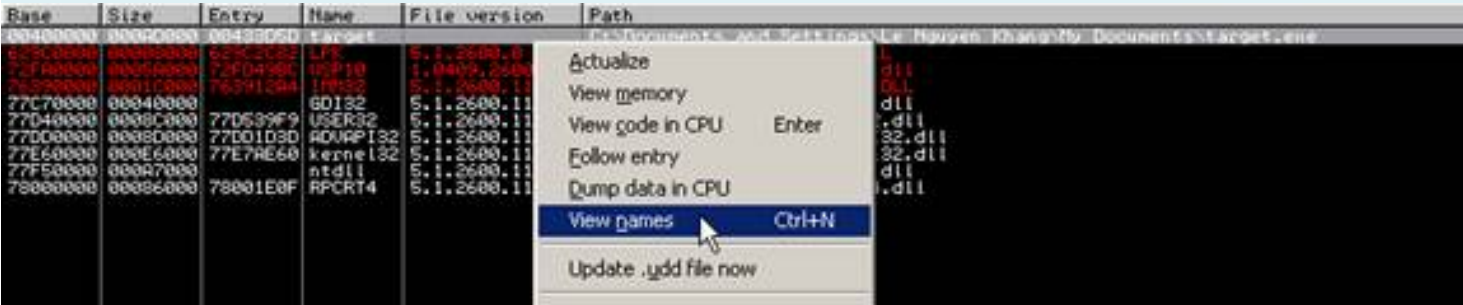
_Cac Aged how that last out this is OK:



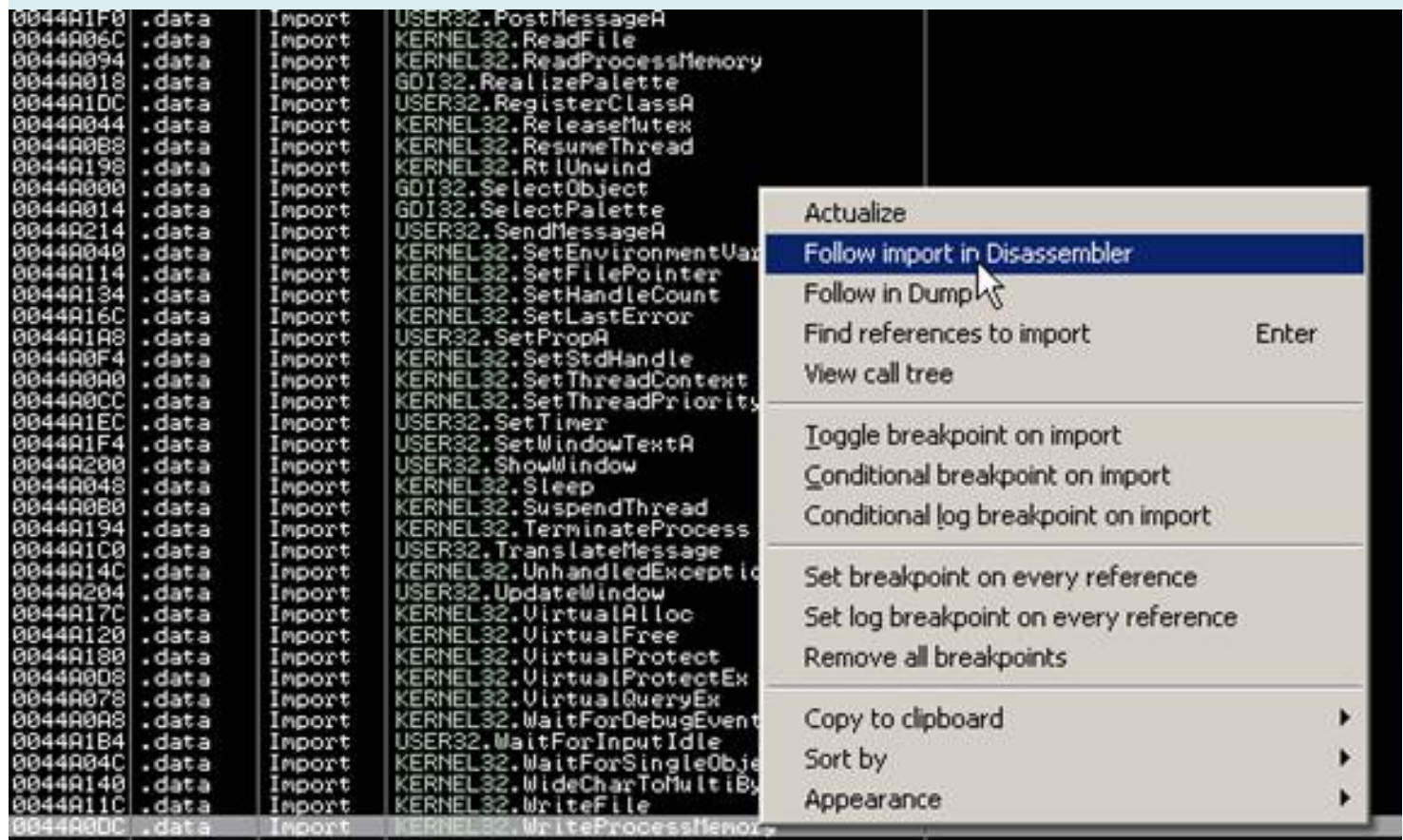


_Cai Target actually unpack this type there is but it is quite trouble for the child. Part father we do completely the same as the tuts, but I want to introduce you to a different method also going to remove the idea is to debug blocker quite as long. If you want to choose the longer this approach.

Olly _Load on target, press Alt + E, right click in modules target.exe select View Names.



_Cuon Screen down WriteProcessMemory search function, right-click to select Import Follow in Disassembler.



_Ban To be here, press F2 set breakpoint at 77E61A95 8BEC MOV EBP, ESP:

77E61A94	55	PUSH EBP
77E61A95	8BEC	MOV EBP, ESP
77E61A97	51	PUSH ECX
77E61A98	51	PUSH ECX
77E61A99	8B45 0C	MOV EAX, DWORD PTR SS:[EBP+C]
77E61A9C	53	PUSH EBX
77E61A9D	8B5D 14	MOV EBX, DWORD PTR SS:[EBP+14]
77E61AA0	56	PUSH ESI
77E61AA1	8B35 B012E677	MOV ESI, DWORD PTR DS:[<&ntdll.NtProtectVirtualMemory]
77E61AA7	57	PUSH EDI
77E61AA8	8B7D 08	MOV EDI, DWORD PTR SS:[EBP+8]
77E61AAB	8945 F8	MOV DWORD PTR SS:[EBP-8], EAX
77E61AAE	8D45 14	LEA EAX, DWORD PTR SS:[EBP+14]
77E61AB1	50	PUSH EAX
77E61AB2	6A 04	PUSH 4

_ Shift + F9, Alt + F9 to you here:

00428543	. 8BC0	MOV EAX,EAX	
00428545	. 8045 FC	LEA EAX,DMWORD PTR SS:[EBP-4]	
00428548	. 50	PUSH EAX	
00428549	. 6A 02	PUSH 2	pBytesWritten BytesToWrite = 2
0042854B	. 804D F8	LEA ECX,DMWORD PTR SS:[EBP-8]	
0042854E	. 51	PUSH ECX	Buffer
0042854F	. 8B55 10	MOV EDX,DMWORD PTR SS:[EBP+10]	
00428552	. 52	PUSH EDX	Address
00428553	. 8B45 08	MOV EAX,DMWORD PTR SS:[EBP+8]	
00428556	. 8B08	MOV ECX,DMWORD PTR DS:[EAX]	
00428558	. 51	PUSH ECX	hProcess
00428559	. FF15 DCA04400	CALL DWORD PTR DS:[<&KERNEL32.WriteProc...	WriteProcessMemory
0042855F	<v7D 07	JMP SHORT target.00428568	
00428561	<v7C 03	JL SHORT target.00428566	
00428563	>EB 05	JMP SHORT target.0042856A	
00428565	. E8	DB E8	
00428566	>74 FB	JE SHORT target.00428563	
00428568	>EB F9	JMP SHORT target.00428563	
0042856A	>EB 5F	JMP SHORT target.004285CB	
0042856C	>8D55 FC	LEA EDX,DMWORD PTR SS:[EBP-4]	
0042856F	. 52	PUSH EDX	
00428570	. 6A 02	PUSH 2	pBytesWritten BytesToWrite = 2
00428572	. 68 CC064500	PUSH target.004506CC	Buffer = target.004506CC
00428577	. 8B45 10	MOV EAX,DMWORD PTR SS:[EBP+10]	
0042857A	. 50	PUSH EAX	Address
0042857B	. 8B4D 08	MOV ECX,DMWORD PTR SS:[EBP+8]	
0042857E	. 8B11	MOV EDX,DMWORD PTR DS:[ECX]	
00428580	. 52	PUSH EDX	hProcess
00428581	. FF15 DCA04400	CALL DWORD PTR DS:[<&KERNEL32.WriteProc...	WriteProcessMemory
00428587	. 50	PUSH EAX	
00428588	. F7D0	NOT EAX	
0042858A	. 0FC8	BSWAP EAX	
0042858C	. 50	PUSH EAX	

_ Click to select at **00428572** Follow in dump> Immediate constant:

_Nhìn Down window dump we see:

Address	Hex dump	ASCII
004506CC	55 8B 00 00 00 00 00 00	i.....
004506D4	00 00 00 00 00 00 00 00
004506DC	00 00 00 00 00 00 00 00
004506E4	00 00 00 00 00 00 00 00
004506EC	00 00 00 00 00 00 00 00
004506F4	00 00 00 00 00 00 00 00
004506FC	00 00 00 00 00 00 00 00
00450704	00 00 00 00 00 00 00 00
0045070C	00 00 00 00 00 00 00 00
00450714	00 00 00 00 00 00 00 00
0045071C	00 00 00 00 00 00 00 00
00450724	00 00 00 00 00 00 00 00
0045072C	00 00 00 00 00 00 00 00
00450734	00 00 00 00 00 00 00 00
0045073C	00 00 00 00 00 00 00 00

_Ta To **55 8B EB FE**,:

Address	Hex dump	ASCII
004506CC	EB FE 00 00 00 00 00 00
004506D4	00 00 00 00 00 00 00 00
004506DC	00 00 00 00 00 00 00 00
004506E4	00 00 00 00 00 00 00 00
004506EC	00 00 00 00 00 00 00 00
004506F4	00 00 00 00 00 00 00 00
004506FC	00 00 00 00 00 00 00 00
00450704	00 00 00 00 00 00 00 00
0045070C	00 00 00 00 00 00 00 00
00450714	00 00 00 00 00 00 00 00
0045071C	00 00 00 00 00 00 00 00
00450724	00 00 00 00 00 00 00 00
0045072C	00 00 00 00 00 00 00 00
00450734	00 00 00 00 00 00 00 00
0045073C	00 00 00 00 00 00 00 00

_Nhan F9 to run, while the program is running, press Alt + F1, enter WaitForDebugEvent:

Command:

_Olly Stop here in WaitForDebugEvent function, and we press Alt + F9, to here:

004240D5	. 85C0	TEST EAX,EAX	
004240D7	.v0F84 EF260000	JE target.004267CC	
004240DD	. 68 00064500	PUSH target.004506D0	
004240E2	. FF15 98A04400	CALL DWORD PTR DS:[<&KERNEL32.EnterCriticalSection = target.004506D0]	
004240E8	. 60	PUSHAD	
004240E9	. 33C0	XOR EAX,EAX	
004240EB	.v75 02	JNZ SHORT target.004240EF	
004240ED	.vEB 15	JMP SHORT target.00424104	
004240EF	.vEB 33	JMP SHORT target.00424124	
004240F1	. C0	DB C0	
004240F2	.v75 18	JNZ SHORT target.0042410C	
004240F4	. 7A 0C	JPE SHORT target.00424102	
004240F6	.v70 0E	JO SHORT target.00424106	
004240F8	.vEB 00	JMP SHORT target.00424107	
004240FA	. E8	DB E8	
004240FB	. 72	DB 72	CHAR 'r'
004240FC	. 0E	DB 0E	
004240FD	. 79	DB 79	CHAR 'y'
004240FE	. F1	DB F1	

_Tro The same, press Space to assemble.

004240D5	. 50	PUSH EAX	
004240D6	. E8 59B1A877	CALL kernel32.DebugActiveProcessStop	
004240DB	. 90	NOP	
004240DC	. ? 0068 D0	ADD BYTE PTR DS:[EAX-30],CH	
004240DF	. ? 06	PUSH ES	
004240E0	. ? 45	INC EBP	
004240E1	. ? 00FF	ADD BH,BH	
004240E3	. ? 15 98A04400	ADC EAX,<&KERNEL32.EnterCriticalSection>	
004240E8	. 60	PUSHAD	
004240E9	. 33C0	XOR EAX,EAX	
004240EB	.v75 02	JNZ SHORT target.004240EF	
004240ED	.vEB 15	JMP SHORT target.00424104	
004240EF	.vEB 33	JMP SHORT target.00424124	
004240F1	. C0	DB C0	
004240F2	.v75 18	JNZ SHORT target.0042410C	

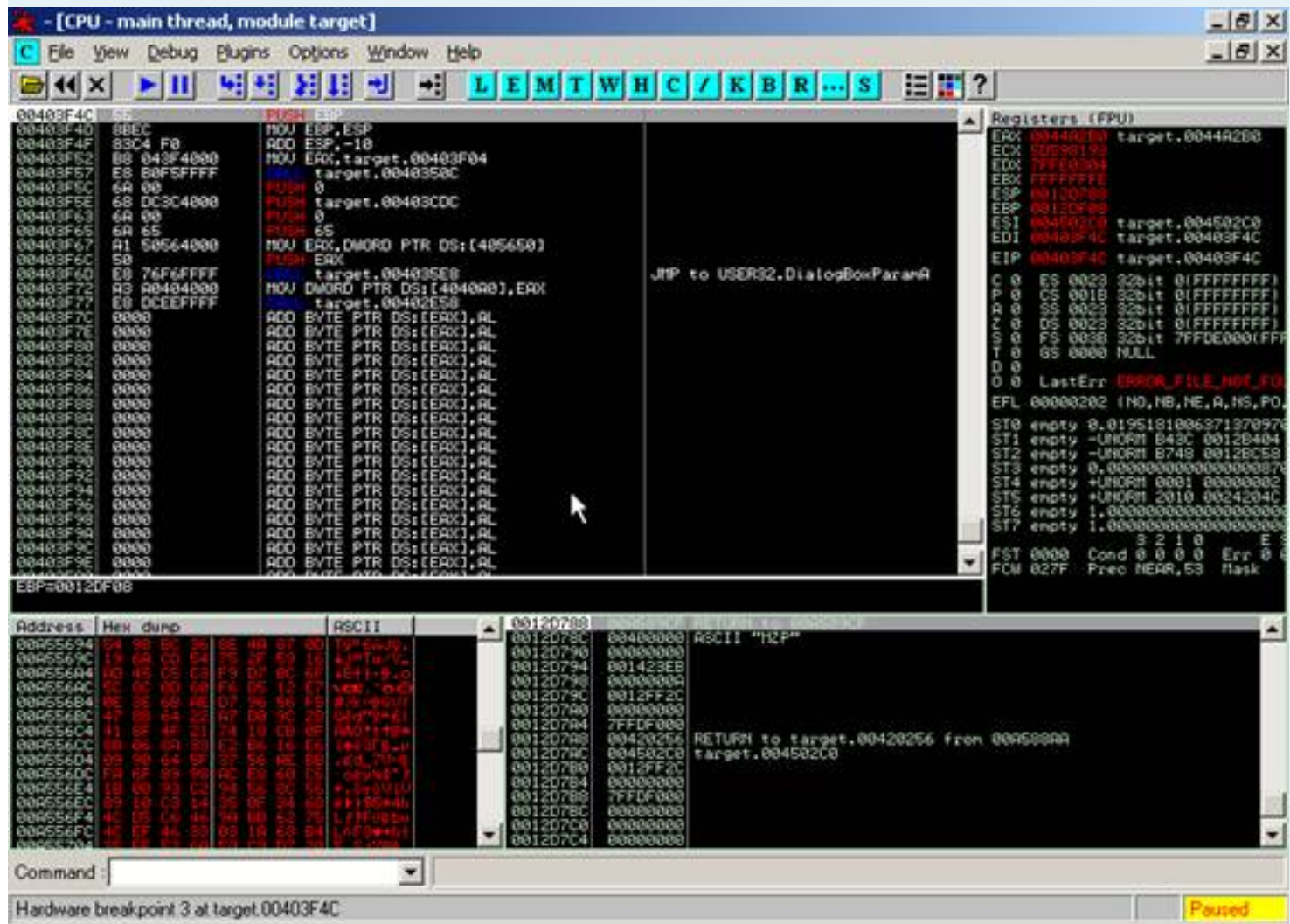
_Xu Complete joker, time directly to children, open up another one Olly, Attach, patch bytes, press Alt + F1 into BP CreateThread, Shift + F9, Ctrl + F9, F7, Ctrl + F9, F7. Done to you here:

00A5894B	6A 00	PUSH 0	
00A5894D	C705 B432A600 A1	MOV DWORD PTR DS:[A632B4],0A63FAC	ASCII "RC"
00A58957	E8 090DFEFF	CALL 00A39665	
00A5895C	59	POP ECX	
00A5895D	59	POP ECX	
00A5895E	E8 DEFAFEFF	CALL 00A48441	
00A58963	8BF8	MOV EDI,EAX	
00A58965	A1 78C5A600	MOV EAX,DWORD PTR DS:[A6C578]	
00A5896A	8B48 34	MOV ECX,DWORD PTR DS:[EAX+34]	
00A5896D	3348 18	XOR ECX,DWORD PTR DS:[EAX+18]	
00A58970	3348 14	XOR ECX,DWORD PTR DS:[EAX+14]	
00A58973	03F9	ADD EDI,ECX	
00A58975	8B0E	MOV ECX,DWORD PTR DS:[ESI]	
00A58977	85C9	TEST ECX,ECX	
00A58979	75 2F	JNZ SHORT 00A589AA	
00A5897B	8B78 34	MOV EDI,DWORD PTR DS:[EAX+34]	
00A5897E	E8 BEFAFEFF	CALL 00A48441	
00A58983	8B00 78C5A600	MOV ECX,DWORD PTR DS:[A6C578]	target.0044A2B0
00A58989	FF76 14	PUSH DWORD PTR DS:[ESI+14]	
00A5898C	8B51 18	MOV EDX,DWORD PTR DS:[ECX+18]	
00A5898F	FF76 10	PUSH DWORD PTR DS:[ESI+10]	
00A58992	3351 14	XOR EDX,DWORD PTR DS:[ECX+14]	
00A58995	FF76 0C	PUSH DWORD PTR DS:[ESI+C]	
00A58998	33D7	XOR EDX,EDI	
00A5899A	03C2	ADD EAX,EDX	
00A5899C	8B51 44	MOV EDX,DWORD PTR DS:[ECX+44]	
00A5899F	3351 40	XOR EDX,DWORD PTR DS:[ECX+40]	
00A589A2	33D7	XOR EDX,EDI	
00A589A4	2BC2	SUB EAX,EDX	
00A589A6	FFD0	CALL EAX	
00A589A8	EB 25	JMP SHORT 00A589CF	
00A589AA	83F9 01	CMPL ECX,1	
00A589AD	75 22	JNZ SHORT 00A58901	
00A589AF	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
00A589B2	FF76 08	PUSH DWORD PTR DS:[ESI+8]	
00A589B5	6A 00	PUSH 0	
00A589B7	E8 85FAFEFF	CALL 00A48441	
00A589BC	50	PUSH EAX	
00A589BD	A1 78C5A600	MOV EAX,DWORD PTR DS:[A6C578]	
00A589C2	8B48 44	MOV ECX,DWORD PTR DS:[EAX+44]	
00A589C5	3348 40	XOR ECX,DWORD PTR DS:[EAX+40]	
00A589C8	3348 34	XOR ECX,DWORD PTR DS:[EAX+34]	
00A589CB	2BF9	SUB EDI,ECX	
00A589CD	FFD7	CALL EDI	
00A589CF	8B08	MOV EBX,EAX	
00A589D1	5F	POP EDI	
00A589D2	8BC3	MOV EAX,EBX	
00A589D4	5E	POP ESI	
00A589D5	5B	POP EBX	

_Call EDI is a signal jump to OEP. Press F2 set breakpoint at 00A589CD. Press Shift + F9, F7. He he, OEP.

00403F4C	55	PUSH EBP	
00403F4D	8BEC	MOV EBP,ESP	
00403F4F	83C4 F0	ADD ESP,-10	
00403F52	B8 043F4000	MOV EAX,target.00403F04	
00403F57	E8 B0F5FFFF	CALL target.0040350C	
00403F5C	6A 00	PUSH 0	
00403F5E	68 DC3C4000	PUSH target.00403C0C	
00403F63	6A 00	PUSH 0	
00403F65	6A 65	PUSH 65	
00403F67	A1 50564000	MOV EAX,DWORD PTR DS:[405650]	
00403F6C	50	PUSH EAX	
00403F6D	E8 76F6FFFF	CALL target.004035E8	
00403F72	A3 A0404000	MOV DWORD PTR DS:[4040A0],EAX	
00403F77	E8 DCEFFFFF	CALL target.00402E58	
00403F7C	0000	ADD BYTE PTR DS:[EAX],AL	
00403F7E	0000	ADD BYTE PTR DS:[EAX],AL	
00403F80	0000	ADD BYTE PTR DS:[EAX],AL	
00403F82	0000	ADD BYTE PTR DS:[EAX],AL	
00403F84	0000	ADD BYTE PTR DS:[EAX],AL	
00403F86	0000	ADD BYTE PTR DS:[EAX],AL	
00403F88	0000	ADD BYTE PTR DS:[EAX],AL	
00403F8A	0000	ADD BYTE PTR DS:[EAX],AL	

IAT _Fix do the same tut 2.



_Ok, Done.

TUT # 5: snd Unpackme - Armadillo 3.70a-Debug-blocker.



_Ban Own writing here!

TUT # 6: hacnho Unpackme - Armadillo 3.75a1-Debug-blocker.



_Ban Own writing here!

TUT # 7: snd Unpackme - Armadillo 4.10-Debug-blocker.

_Ban Own writing here!



_Ban Own writing here!

TUT # 8: snd Unpackme - Armadillo 4.20-Debug-blocker.



_Ban Own writing here!

IV. Conclusion

Hix, the finish is part of the format debug blocker. This format is the same wish, because I only cracked pub should test it all, touching soft Custom will build slightly different. Lost 3h to write this tut. I look after I mentioned to some form has added CopyMEM II. The past have sent you a soft seek help unpack bags, he sent to him this nq, I unpack the complete return both a thank you can not, since it does not receive any request any! Who I send spam report that under L.

Wishing success. Appointment in the series with # CopyMem II.

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini, hoadongnoi, dqtln, hosiminh, Nilrem, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, anh_surprised ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RiF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx to authors of OllyDBG.

To be continued ...

Written by [hacnho](#) (tutorial date: Saigon 24/08/2005)

Armadillo collect sand-stone

Part 3: Unpacking Armadillo x.xx + = cracked!

I. Intro:

_Ta Meet again in this Section 3: In this tut we will handle with soft + upackme goals unpack Once cracked. This is version Expansion of the series 1 and series # # 2! Therefore Series # 4 will discuss CopyMemII. Single!

I do _Khi tut also has the video tut. If you want to be able to check rq the steps of their practice. Specify one that target all in all I test on all operating systems WindowsXP SP1. All are of relatively, hope you do not surprised to see the target does not run on your machine.

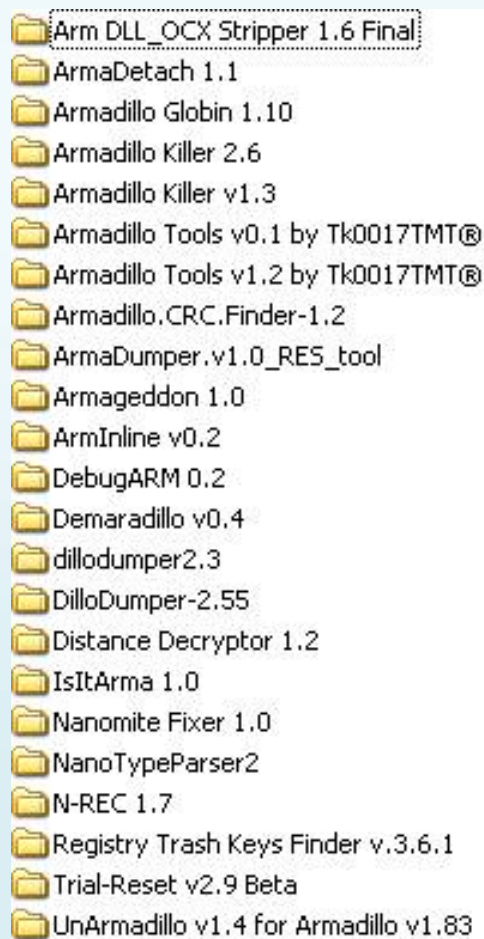
II. Tools:

1:10 OllyDBG with plugin: Hide Debugger 1.2.3f, Armadillo Process Detach Plugin v1.0, OllyDBG PE Dumper 3.0.3, Command Bar 3.10.109c has bug patch with RE-PAIR 0.6 + AntiDetectOlly_v2.2.4.

PE Tools v1.5.RC5.2005

Import REConstructor v1.6 FINAL

You have more questions about the tools I have not armadillo. Hi, please thừa that the Armadillo NET tools on my lot. You see the collection of tools I nhé. Rq like it in this topic, including AM or send YIM.



III. Unpacking

Target # 1: Remote Installer 1.3



Olly _Load on target, set a breakpoint on execution: He WaitForDebugEvent. Press F9, we have:

```

0012DAB0 0040802A CALL to WaitForDebugEvent from
0012DAB4 0012EB60 pDebugEvent = 0012EB60
0012DAB8 000003E8 Timeout = 1000. ms
0012DABC 0012FF04
0012DAC0 00000000
0012DAC4 004EB869 Remote_I.004EB869
0012DAC8 00000000
0012DACC 00000096
0012DAD0 00000000
0012DAD4 83646B1A
0012DAD8 00000000
0012DADC 00000000
0012DAE0 00000000
0012DAE4 00000000

```

_Ghi Memory address 0012EB60. Ctrl + F2 restart, delete He, BP set WriteProcessMemory. Press F9:

```

0012D7B0 004E0054 CALL to WriteProcessMemory from
0012D7B4 0000004C hProcess = 0000004C
0012D7B8 004F3000 Address = 4F3000
0012D7BC 0012DAA0 Buffer = 0012DAA0
0012D7C0 00000002 BytesToWrite = 2
0012D7C4 0012DAA4 pBytesWritten = 0012DAA4
0012D7C8 0012F018 UNICODE "e132.dll"
0012D7CC 00002710
0012D7D0 004EB869 Remote_I.004EB869
0012D7D4 77F6379E RETURN to ntdll.77F6379E from ntdll
0012D7D8 0012D80C
0012D7DC 77F748CB RETURN to ntdll.77F748CB from ntdll
0012D7E0 00150000
0012D7E4 00000000

```

Address	Hex dump	ASCII
0012EB60	00 00 00 00 90 EB 12 00 1D 2F F5 77 90 FE 12 00E\$+.#/JwE=
0012EB70	0E 00 00 00 9C EB 12 00 C0 EB 12 00 1C 00 00 00	...E\$+.L\$+.L...
0012EB80	90 FE 12 00 9C 00 00 00 69 B8 4E 00 00 00 00	E\$+.L\$+.L\$+.L...
0012EB90	D8 EC 12 00 4F DE E7 77 D0 EC 12 00 0E 00 00 00	...0...L\$+.L...
0012EBA0	00 00 00 00 F8 00 E6 77 00 00 00 00 1C 01 00 00	...°pw...L0...
0012EBB0	05 00 00 00 01 00 00 00 28 0A 00 00 02 00 00 00	...0...L\$+.L...
0012EBC0	53 00 65 00 00 00 00 00 00 00 00 00 00 00 00 00	S.e.....
0012EBD0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012EBE0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012EBF0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012EC00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012EC10	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

_F9 Times 2:

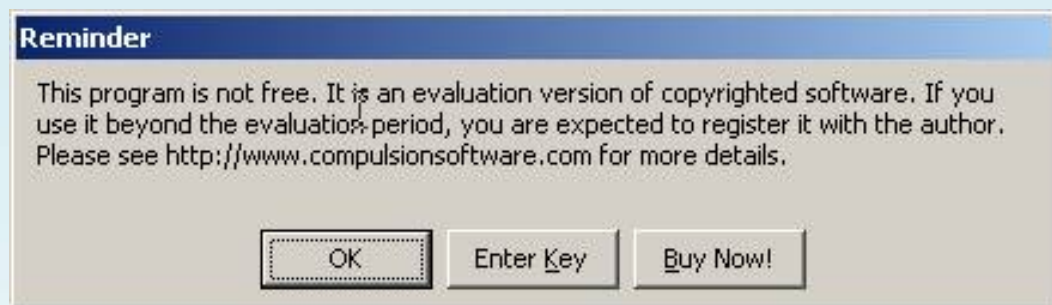
```

001207B0 004F00A5 CALL to WriteProcessMemory
001207B4 0000004C hProcess = 0000004C
001207B8 004F3000 Address = 4F3000
001207BC 00509020 Buffer = Remote_I.00509020
001207C0 00000002 BytesToWrite = 2
001207C4 0012DAA4 pBytesWritten = 0012DAA4
001207C8 0012F018 UNICODE "el32.dll"
001207CC 00002710
001207D0 004EB869 Remote_I.004EB869
001207D4 004E0158 Remote_I.004E0158
001207D8 00000050
001207DC 001207E0
001207E0 00010001
001207E4 00000000

```

Address	Hex dump	ASCII
0012EB60	00 00 00 00 90 EB 12 00 1D 2F F5 77 90 FE 12 00E\$+.#/JwE=
0012EB70	0E 00 00 00 9C EB 12 00 C0 EB 12 00 1C 00 00 00	...E\$+.L\$+.L...
0012EB80	90 FE 12 00 9C 00 00 00 69 B8 4E 00 00 00 00	E\$+.L\$+.L\$+.L...
0012EB90	D8 EC 12 00 4F DE E7 77 D0 EC 12 00 0E 00 00 00	...0...L\$+.L...
0012EBA0	00 00 00 00 F8 00 E6 77 00 00 00 00 1C 01 00 00	...°pw...L0...
0012EBB0	05 00 00 00 01 00 00 00 28 0A 00 00 02 00 00 00	...0...L\$+.L...
0012EBC0	53 00 65 00 00 00 00 00 00 00 00 00 00 00 00 00	S.e.....
0012EBD0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012EBE0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012EBF0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012EC00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0012EC10	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

_F9 Times 3, program run entirely with the registered notice:



_Khua Khua, not listed on a 1000 bytes code has OEP. Yeah! The resolution of the court, not CopyMemII. You can read by tut sLayer/MP2k say this is a soft CopyMemII, I think kô to J ! Restart Olly, load target. Clear all breakpoint. Bp WriteProcessMemory, F9, F9:


```

001207B0 004F00A5 CALL to WriteProcessMemory from R
001207B4 0000004C hProcess = 0000004C
001207B8 004F3000 Address = 4F3000
001207BC 00509020 Buffer = Remote_I.00509020
001207C0 00000002 BytesToWrite = 2
001207C4 0012DAA4 pBytesWritten = 0012DAA4
001207C8 0012F018 UNICODE "e132.dll"
001207CC 00002710
001207D0 004EB869 Remote_I.004EB869
001207D4 004E0158 RETURN to Remote_I.004E0158 from k
001207D8 00000050
001207DC 001207E0
001207E0 00010001
001207E4 00000000

```

_Follow Dump in, change into 60E8 EBFE:

Address	Hex dump	ASCII
00509020	60 E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509030	78 32 79 BF 00 00 00 00 03 01 00 00 00 00 00 00	x2y.....
00509040	20 37 39 00 08 1F 39 00 8C F4 12 00 00 10 40 00	g9.79.i(+.0
00509050	00 00 00 00 CF 00 00 00 00 00 00 00 80 1C 39 00	...=.....CL9

_Bp WaitForDebugEvent, F9, Ctrl + F9, F7:

Address	Hex dump	Disassembly	Comments
004DBD2A	85C0	TEST EAX,EAX	
004DBD2C	0F84 B1230000	JE Remote_I.004DE0E3	
004DBD32	60	PUSHAD	
004DBD33	33C0	XOR EAX,EAX	
004DBD35	75 02	JNZ SHORT Remote_I.004DBD39	
004DBD37	EB 15	JMP SHORT Remote_I.004DBD4E	
004DBD39	EB 33	JMP SHORT Remote_I.004DBD6E	
004DBD3B	C075 18 7A	SAL BYTE PTR SS:[EBP+18],7A	Shift constant out of range 1.
004DBD3F	0C 70	OR AL,70	
004DBD41	0E	PUSH CS	
004DBD42	EB 00	JMP SHORT Remote_I.004DBD51	
004DBD44	E8 720E79F1	CALL F1C6C88B	
004DBD49	FF15 00790974	CALL DWORD PTR DS:[74097900]	
004DBD4F	F0:EB 87	LOCK JMP SHORT Remote_I.004DBCD9	LOCK prefix is not allowed

_Theo Old methods. Push EAX Call DebugActiveProcessStop, submitted. Record of EAX.

Address	Hex dump	Disassembly	Comments	Registers (FPU)
004DBD2A	50	PUSH EAX		EAX 00000001
004DBD2B	90	CALL kernel32.DebugActiveProcessStop		ECX 00153480
004DBD30	90	NOP		EDX 00000000
004DBD31	0060 33	ADD BYTE PTR DS:[EAX+33],AH		EBX 004EB869 Remote_I.004EB869
004DBD34	C075 02 EB	SAL BYTE PTR SS:[EBP+2],0EB	Shift constant out of range 1.	ESP 0012DAA4
004DBD36	15 EB33C075	ADC EAX,75C03EB		EBP 0012F578
004DBD3D	187A 0C	SBB BYTE PTR DS:[EDX+C],BH		ESI 00002710
004DBD40	70 0E	JO SHORT Remote_I.004DBD50		EIP 0012F018 UNICODE "e132.dll"
004DBD42	EB 00	JMP SHORT Remote_I.004DBD51		EIP 004DBD2A Remote_I.004DBD2A
004DBD44	E8 720E79F1	CALL F1C6C88B		C 0 ES 0023 32bit 0(FFFFFFFF)
004DBD49	FF15 00790974	CALL DWORD PTR DS:[74097900]		P 0 CS 001B 32bit 0(FFFFFFFF)
004DBD4F	F0:EB 87	LOCK JMP SHORT Remote_I.004DBCD9	LOCK prefix is not allowed	A 0 SS 0023 32bit 0(FFFFFFFF)
004DBD52	D07A F0	FSTP TBYTE PTR DS:[EDX-10]		Z 0 DS 0023 32bit 0(FFFFFFFF)
004DBD55	A0 336139C0	MOV AL,BYTE PTR DS:[C0336139]		S 0 FS 0038 32bit 7FFDCE000(FFF)
004DBD5A	A0 34905000	MOV AL,BYTE PTR DS:[509034]		T 0 GS 0000 NULL
004DBD5F	85C0	TEST EAX,EAX		D 0
004DBD61	0F85 00010000	JNZ Remote_I.004DBE67		O 0 LastErr ERROR_SUCCESS (0000)
004DBD67	60	PUSHAD		EFL 00000202 (NO,NB,NE,A,NS,PO,GE)
004DBD68	33C0	XOR EAX,EAX		ST0 empty +UNORM 17B2 77F51778 72
004DBD6A	75 02	JNZ SHORT Remote_I.004DBD6E		ST1 empty +UNORM 0002 00000000 00
004DBD6C	EB 15	JMP SHORT Remote_I.004DBD83		ST2 empty +UNORM 0609 77F5F4A1 72
004DBD6E	EB 33	JMP SHORT Remote_I.004DBD83		ST3 empty +UNORM 023E 00000007 00
004DBD70	C075 18 7A	SAL BYTE PTR SS:[EBP+18],7A	Shift constant out of range 1.	ST4 empty +UNORM 023E 7197CE69 72
004DBD74	0C 70	OR AL,70		ST5 empty +UNORM 5E9A 00000010 00
004DBD76	0E	PUSH CS		ST6 empty +UNORM 800E 7197D0B1 00
004DBD77	EB 00	JMP SHORT Remote_I.004DBD86		ST7 empty +UNORM 0002 00000000 00
004DBD79	E8 720E79F1	CALL F1C6C88B		3 2 1 0 E S
004DBD7E	FF15 00790974	CALL DWORD PTR DS:[74097900]		FST 0000 Cond 0 0 0 0 Err 0 0 0
004DBD84	F0:EB 87	LOCK JMP SHORT Remote_I.004DBD8E	LOCK prefix is not allowed	FCM 027F Prec NEAR,53 Mask
004DBD87	D07A F0	FSTP TBYTE PTR DS:[EDX-10]		
004DBD8A	A0 33618B80	MOV AL,BYTE PTR DS:[808B6133]		
004DBD8F	4B	DEC EAX		
004DBD90	F6FF	IDIV BH		
004DBD92	FF81 E1FF0000	INC DWORD PTR DS:[ECX+FFF1]		

Address	Hex dump	ASCII
00509020	EB EB 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509030	78 32 79 BF 00 00 00 00 03 01 00 00 00 00 00 00	x2y.....
00509040	20 37 39 00 08 1F 39 00 8C F4 12 00 00 10 40 00	g9.79.i(+.0
00509050	00 00 00 00 CF 00 00 00 00 00 00 00 80 1C 39 00	...=.....CL9
00509060	68 07 39 00 00 53 39 00 00 57 39 00 01 00 00 00	h.9.39.0.9.0
00509070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005090A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005090B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005090C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005090D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

_Nhan F8, to trace 004DBD30 90 filed, hix review of EAX.

The screenshot displays a debugger interface with three main panels:

- Assembly Panel (Top Left):** Shows a list of instructions starting from address 0040B02A. The instruction at 0040B030 is highlighted: `CALL kernel32.DebugActiveProcessStop`. Other instructions include `MOV EAX, 75C83EB`, `LOCK JMP SHORT Rremote_I.0040B0D9`, and `INC DWORD PTR DS:[ECX+FFF1]`.
- Registers Panel (Top Right):** Displays the state of CPU registers. `EAX` is 00000000, `ECX` is 77F5119F, and `EDX` is 00000000. The `EIP` register is 0040B030. A red circle highlights the `EAX` register.
- Hex Dump Panel (Bottom):** Shows a memory dump starting at address 00509020. The first few bytes are 78 32 79 BF, which correspond to the ASCII string "x2y...".

_Toi Ago, according to doctors lightphoenix EAX = 0 it is still Attach the MSDN's BillGates J. Ok, we see the child Attach:

Process	Name	Window	Path
00000118	spoolsv		C:\WINDOWS\system32\spoolsv.exe
000001BC	smss		SystemRoot\System32\smss.exe
000001FC	csrss		??\C:\WINDOWS\system32\csrss.exe
00000214	winlogon	NetDDE Agent	??\C:\WINDOWS\system32\winlogon.exe
00000240	services		C:\WINDOWS\system32\services.exe
0000024C	lsass		C:\WINDOWS\system32\lsass.exe
000002E8	svchost		C:\WINDOWS\system32\svchost.exe
00000314	svchost		C:\WINDOWS\system32\svchost.exe
00000396	svchost		C:\WINDOWS\system32\svchost.exe
00000408	svchost		C:\WINDOWS\system32\svchost.exe
00000440	Explorer	armadillo_tuts_3	C:\WINDOWS\Explorer.EXE
0000048C	fast	Default INE	C:\WINDOWS\System32\fast.exe
00000558	inetinfo		C:\WINDOWS\System32\inetrv\inetinfo.exe
00000568	CrackerU	Cracker Utils	I:\Project DELPHI7\My Project\CrackerUtils\CrackerUtils.exe
00000574	Snagit32	Paint Tools	C:\Program Files\TechSmith\Snagit 7\Snagit32.exe
00000588	NDM		C:\Program Files\Comm Files\Microsoft Shared\VS7DEBUG\NDM.EXE
0000068C	wdfmgr		C:\WINDOWS\System32\wdfmgr.exe
00000708	Fast		C:\WINDOWS\System32\Fast.exe
00000708	(h) OLLVM	- ECPU - main thread, nod	I:\Cracker\Debug-Disassembler\odbg118_org\{h} OLLVDBG.EXE
00000A98	wmplayer	WMFVideoCtrl\NotifyWndClass	C:\Program Files\Windows Media Player\wmplayer.exe
00000CC4	WINWORD	Can't Redo	C:\Program Files\Microsoft Office\OFFICE11\WINWORD.EXE
00000CF0	ctfmon	TF_FloatingLangBar_MndTitl	C:\WINDOWS\System32\ctfmon.exe
00000D8C	TSCHelp	C:\Program Files\TechSmith	C:\Program Files\TechSmith\Snagit 7\TSCHelp.exe
00000DF0	UniKey	UniKey 3.62	C:\Program Files\UniKey\UniKey.exe
00000E5C	Remote I		C:\Program Files\Compulsion\Remote Installer\Remote Installer.exe
00000E68	Remote I		C:\Program Files\Compulsion\Remote Installer\Remote Installer.exe
00000FE4	eneditor	Untitled * - EnEditor	C:\Program Files\EnEditor\eneditor.exe



Process	Name	Window	Path
00000118	spoolsv		C:\WINDOWS\system32\spoolsv.exe
000001BC	smss		SystemRoot\System32\smss.exe
000001FC	csrss		??\C:\WINDOWS\system32\csrss.exe
00000214	winlogon	NetDDE Agent	??\C:\WINDOWS\system32\winlogon.exe
00000240	services		C:\WINDOWS\system32\services.exe
0000024C	lsass		C:\WINDOWS\system32\lsass.exe
000002E8	svchost		C:\WINDOWS\system32\svchost.exe
00000314	svchost		C:\WINDOWS\system32\svchost.exe
00000396	svchost		C:\WINDOWS\system32\svchost.exe
00000408	svchost		C:\WINDOWS\system32\svchost.exe
00000440	Explorer	armadillo_tuts_3	C:\WINDOWS\Explorer.EXE
0000048C	fast	Default INE	C:\WINDOWS\System32\fast.exe
00000558	inetinfo		C:\WINDOWS\System32\inetrv\inetinfo.exe
00000568	CrackerU	Cracker Utils	I:\Project DELPHI7\My Project\CrackerUtils\CrackerUtils.exe
00000574	Snagit32	Paint Tools	C:\Program Files\TechSmith\Snagit 7\Snagit32.exe
00000588	NDM		C:\Program Files\Comm Files\Microsoft Shared\VS7DEBUG\NDM.EXE
0000068C	wdfmgr		C:\WINDOWS\System32\wdfmgr.exe
00000708	Fast		C:\WINDOWS\System32\Fast.exe
00000708	(h) OLLVM	- ECPU - main thread, nod	I:\Cracker\Debug-Disassembler\odbg118_org\{h} OLLVDBG.EXE
00000A98	wmplayer	WMFVideoCtrl\NotifyWndClass	C:\Program Files\Windows Media Player\wmplayer.exe
00000CC4	WINWORD	Can't Redo	C:\Program Files\Microsoft Office\OFFICE11\WINWORD.EXE
00000CF0	ctfmon	TF_FloatingLangBar_MndTitl	C:\WINDOWS\System32\ctfmon.exe
00000D8C	TSCHelp	C:\Program Files\TechSmith	C:\Program Files\TechSmith\Snagit 7\TSCHelp.exe
00000DF0	UniKey	UniKey 3.62	C:\Program Files\UniKey\UniKey.exe
00000E5C	Remote I		C:\Program Files\Compulsion\Remote Installer\Remote Installer.exe
00000E68	Remote I		C:\Program Files\Compulsion\Remote Installer\Remote Installer.exe
00000FE4	eneditor	Untitled * - EnEditor	C:\Program Files\EnEditor\eneditor.exe



_Bay Time I pass by, Okie, I'll show you a way to pass by! Restart all again. Bp
WriteProcessMemory, F9, F9:

```

001207B0 004E0A85 CALL to WriteProcessMemory from R
001207B4 0000004C hProcess = 0000004C
001207B8 004F3000 Address = 4F3000
001207BC 00509020 Buffer = Remote_I.00509020
001207C0 00000002 BytesToWrite = 2
001207C4 0012DAA4 pBytesWritten = 0012DAA4
001207C8 0012F018 UNICODE "el32.dll"
001207CC 00002710
001207D0 004EB869 Remote_I.004EB869
001207D4 004E0158 RETURN to Remote_I.004E0158 from k
001207D8 00000050
001207DC 001207E0
001207E0 00010001
001207E4 00000000

```


_Follow Dump in, change into 60E8 EBFE:

Address	Hex dump	ASCII
00509020	60 E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00	3.....
00509030	78 52 79 BF 00 00 00 00 03 01 00 00 00 00 00 00	x2y.....0.....
00509040	20 67 39 00 08 1F 39 00 8C F4 12 00 00 10 40 00	g9.79.i(r...e
00509050	00 00 00 00 CF 00 00 00 00 00 00 00 80 1C 39 00	...=...CL9

_O Code Window, Ctrl + G to enter: WaitForDebugEvent. Press F2 to set breakpoint, F9, Alt + F9.

004DBD2A	85C0	TEST EAX,EAX	
004DBD2C	0F84 B1230000	JE Remote_I.004DE0E3	
004DBD32	60	PUSHAD	
004DBD33	33C0	XOR EAX,EAX	
004DBD35	75 02	JNZ SHORT Remote_I.004DBD39	
004DBD37	EB 15	JMP SHORT Remote_I.004DBD4E	
004DBD39	EB 33	JMP SHORT Remote_I.004DBD6E	
004DBD3B	C075 18 7A	SAL BYTE PTR SS:[EBP+18],7A	Shift constant out of range 1.
004DBD3F	0C 70	OR AL,70	
004DBD41	0E	PUSH CS	
004DBD42	EB 00	JMP SHORT Remote_I.004DBD51	

_Assemble To:

004DBD2A	85C0	TEST EAX,EAX	
004DBD2C	0F84 B1230000	JE Remote_I.004DE0E3	
004DBD32	60	PUSHAD	
004DBD33	33C0	XOR EAX,EAX	
004DBD35	75 02	JNZ SHORT Remote_I.004DBD39	
004DBD37	EB 15	JMP SHORT Remote_I.004DBD4E	
004DBD39	EB 33	JMP SHORT Remote_I.004DBD6E	
004DBD3B	C075 18 7A	SAL BYTE PTR SS:[EBP+18],7A	Shift constant out of range 1.
004DBD3F	0C 70	OR AL,70	
004DBD41	0E	PUSH CS	
004DBD42	EB 00	JMP SHORT Remote_I.004DBD51	
004DBD44	E8 720E79F1	CALL FIC6CBEB	
004DBD49	FF15 00790974	LOCK JMP SHORT Remote_I.004DBD0E	LOCK prefix is not allowed
004DBD4F	F01E8 87	FSTP TBYTE PTR DS:[EDX-10]	
004DBD55	A0 336133C0	MOV AL,BYTE PTR DS:[C0336133]	
004DBD5A	A0 34905000	MOV AL,BYTE PTR DS:[509034]	
004DBD5F	85C0	TEST EAX,EAX	
004DBD61	0F85 00010000	JE Remote_I.004DBE67	
004DBD67	60	PUSHAD	
004DBD68	33C0	XOR EAX,EAX	
004DBD6A	75 02	JNZ SHORT Remote_I.004DBD6E	
004DBD6C	EB 15	JMP SHORT Remote_I.004DBD83	
004DBD6E	EB 33	JMP SHORT Remote_I.004DBD0E	
004DBD70	C075 18 7A	SAL BYTE PTR SS:[EBP+18],7A	Shift constant out of range 1.
004DBD74	0C 70	OR AL,70	
004DBD76	0E	PUSH CS	
004DBD77	EB 00	JMP SHORT Remote_I.004DBD86	
004DBD79	E8 720E79F1	CALL FIC6CBEB	
004DBD7E	FF15 00790974	LOCK JMP SHORT Remote_I.004DBD0E	LOCK prefix is not allowed
004DBD84	F01E8 87	FSTP TBYTE PTR DS:[EDX-10]	
004DBD87	0B7A F0	MOV AL,BYTE PTR DS:[0B8B6133]	
004DBD8A	A0 3361B880	MOV AL,BYTE PTR DS:[B88B6133]	
004DBD8F	48	DEC EAX	
004DBD90	E4FF	INT3 BH	

Address	Hex dump	ASCII
00509020	00 00 E6 77 00 00 00 00 00 00 00 00 00 00 00 00	..vw.....
00509030	00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00	0.....
00509040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005090A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005090B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005090C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005090D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005090E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005090F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005091A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005091B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005091C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005091D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005091E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005091F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00509200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

_Den Unpack this process took place normally. Attach child, F9, F12. Change to EBFE 60E8:

004F3000	60	PUSHAD	
004F3001	EB 00000000	CALL Remote_I.004F3006	
004F3006	50	POP EBP	
004F3007	50	PUSH EAX	
004F3008	51	PUSH ECX	
004F3009	EB 0F	JMP SHORT Remote_I.004F301A	
004F300B	B9 E80FB8EB	MOV ECX,E880FEB	
004F3010	07	POP ES	
004F3011	B9 E80F90EB	MOV ECX,E890FEB	Modification of segment regist
004F3016	08FD	OR CH,BH	
004F3018	EB 08	JMP SHORT Remote_I.004F3025	
004F301A	F2:	PREFIX REPNE:	Superfluous prefix
004F301B	EB F5	JMP SHORT Remote_I.004F3012	
004F301D	EB F5	JMP SHORT Remote_I.004F3015	

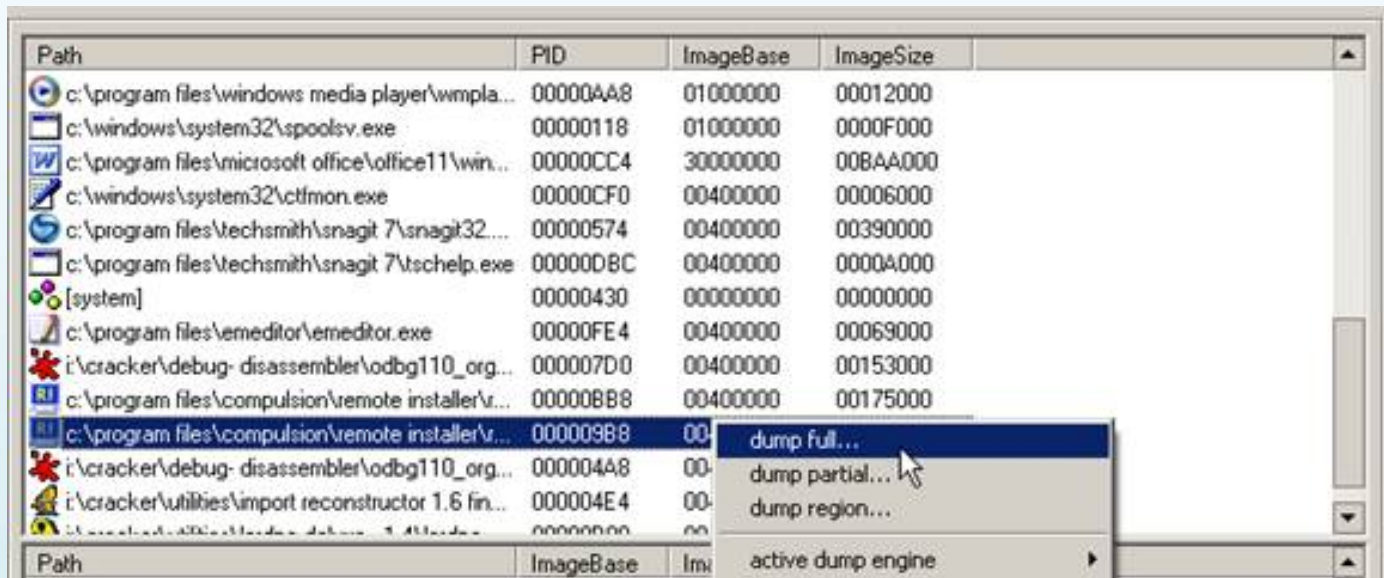
_Alt + M, Breakpoint On Memory Access, F9. OEP.

```

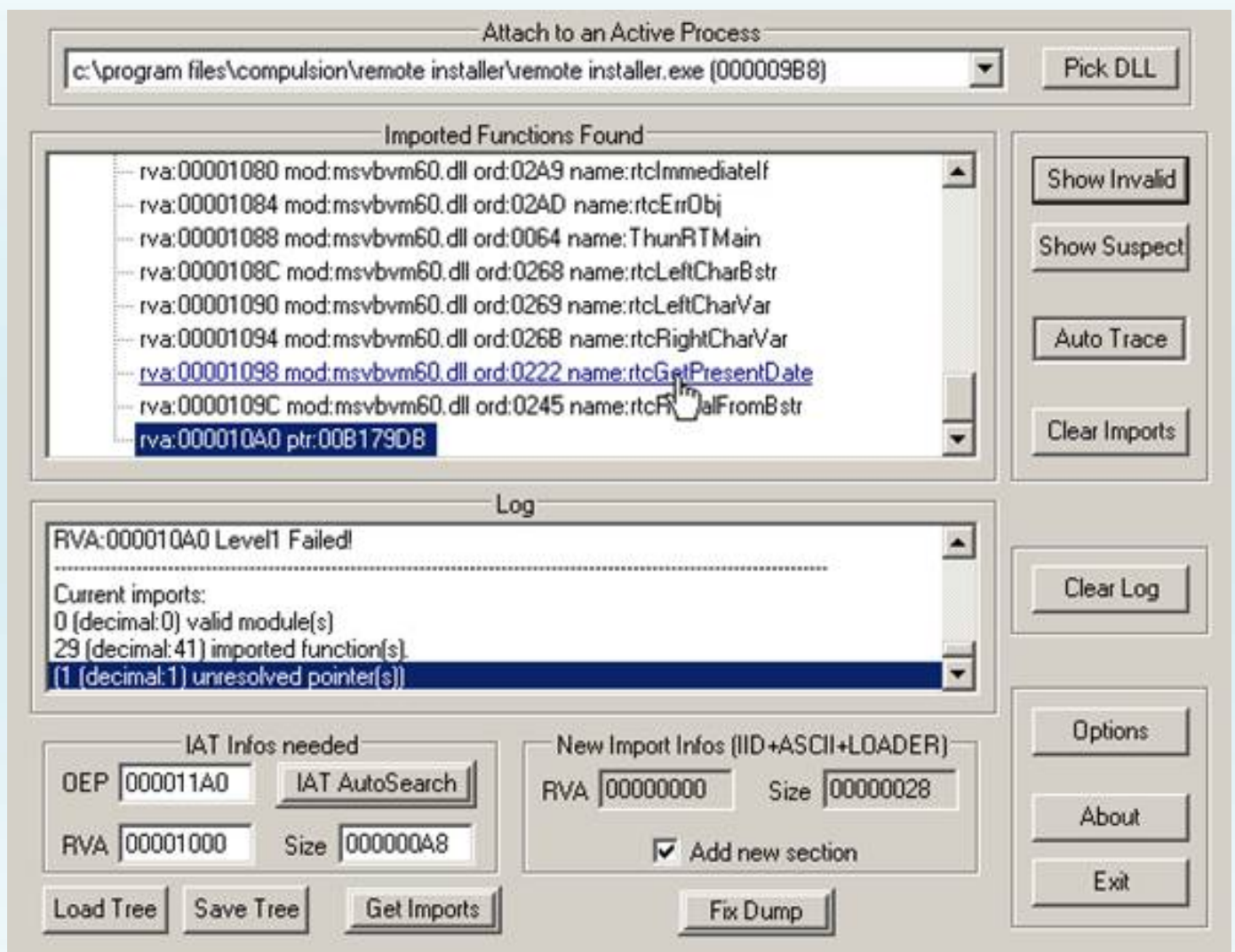
004011A0  68 E4124000  PUSH Remote_I.004012E4
004011A5  E8 F0FFFFFF  CALL Remote_I.0040119A
004011AA  0000        ADD BYTE PTR DS:[EAX],AL
004011AC  0000        ADD BYTE PTR DS:[EAX],AL
004011AE  0000        ADD BYTE PTR DS:[EAX],AL
004011B0  3000        XOR BYTE PTR DS:[EAX],AL
004011B2  0000        ADD BYTE PTR DS:[EAX],AL
004011B4  58          POP EAX
004011B5  0000        ADD BYTE PTR DS:[EAX],AL
004011B7  0040 00     ADD BYTE PTR DS:[EAX],AL

```

Full-LordPE dump:



_IAT Fix:



_1 Function invalid. We see the program in VB. I dump the data, load it on the run, yeah. But close it crash. IAT considered, find missing __vbaend function. Add the invisible. Fix dump. It's good run! No further notice anything. View:



1 _Hang this product is more AssetDB, shared a packer and this is CopyMemII. We will discuss this in soft series # 4. View:

_F9 Hits 1:

Address	Hex dump	ASCII
0012EB60	00 00 00 00 90 EB 12 00E\$#
0012EB68	10 2F F5 77 90 FE 12 00	#/Jug\$#
0012EB70	0E 00 00 00 9C EB 12 00	#....E\$#
0012EB78	C0 EB 12 00 1C 00 00 00	\$\$\$....
0012EB80	90 FE 12 00 9C 00 00 00	\$\$\$E...
0012EB88	69 28 71 00 00 00 00 00	lfg.....
0012EB90	08 EC 12 00 4F DE E7 77	q\$\$.0lw
0012EB98	00 EC 12 00 0E 00 00 00	=\$\$.#...
0012EBA0	00 00 00 00 F8 00 E6 77\$.pw
0012EBB0	00 00 00 00 1C 01 00 00L0...
0012EBC0	05 00 00 00 01 00 00 00	\$....0...
0012EBD0	28 0A 00 00 02 00 00 00	(...0...
0012EBE0	53 00 65 00 00 00 00 00	S.e.....
0012EBF0	00 00 00 00 00 00 00 00

Address	Hex dump	ASCII
001207B8	0000004C	CALL to WriteProcessMemory from CSAssetD.00712869
001207B4	0000004C	hProcess = 0000004C
001207B8	0071A000	Address = 71A000
001207BC	00120AA0	Buffer = 00120AA0
001207C0	00000002	BytesToWrite = 2
001207C4	00120AA4	pBytesWritten = 00120AA4
001207C8	0012F018	UNICODE "e132.dll"
001207CC	00002710	
001207D0	00712869	CSAssetD.00712869
001207D4	77F6379E	RETURN to ntdll.77F6379E from ntdll.77F7804E
001207D8	0012080C	
001207DC	77F748CB	RETURN to ntdll.77F748CB from ntdll.77F7476C
001207E0	00150000	
001207E4	00000000	
001207E8	00153620	
001207EC	00150000	

_F9 Lan2:

Address	Hex dump	ASCII
0012EB60	00 00 00 00 90 EB 12 00E\$#
0012EB68	10 2F F5 77 90 FE 12 00	#/Jug\$#
0012EB70	0E 00 00 00 9C EB 12 00	#....E\$#
0012EB78	C0 EB 12 00 1C 00 00 00	\$\$\$....
0012EB80	90 FE 12 00 9C 00 00 00	\$\$\$E...
0012EB88	69 28 71 00 00 00 00 00	lfg.....
0012EB90	08 EC 12 00 4F DE E7 77	q\$\$.0lw
0012EB98	00 EC 12 00 0E 00 00 00	=\$\$.#...
0012EBA0	00 00 00 00 F8 00 E6 77\$.pw
0012EBB0	00 00 00 00 1C 01 00 00L0...
0012EBC0	05 00 00 00 01 00 00 00	\$....0...
0012EBD0	28 0A 00 00 02 00 00 00	(...0...
0012EBE0	53 00 65 00 00 00 00 00	S.e.....
0012EBF0	00 00 00 00 00 00 00 00

Address	Hex dump	ASCII
001207B8	0000004C	CALL to WriteProcessMemory from CSAssetD.00712869
001207B4	0071A000	hProcess = 0000004C
001207BC	00730020	Address = 71A000
001207C0	00000002	Buffer = CSAssetD.00730020
001207C4	00120AA4	BytesToWrite = 2
001207C8	0012F018	pBytesWritten = 00120AA4
001207CC	00002710	UNICODE "e132.dll"
001207D0	00712869	CSAssetD.00712869
001207D4	00707158	RETURN to CSAssetD.00707158 from kernel32.GetThe
001207D8	00000050	
001207DC	001207E0	
001207E0	00010001	
001207E4	00000000	
001207E8	00000038	
001207EC	00000020	

_F9 Times 3:

Address	Hex dump	ASCII		00120950	00700000	Call to WriteProcessMemory from CSAssetD.00700000
0012EB60	01 00 00 00 F8 00 00 00	0...0...		00120954	0000004C	hProcess = 0000004C
0012EB68	FC 00 00 00 01 00 00 000..C		00120958	0040E000	Address = 40E000
0012EB70	00 00 00 00 00 00 00 00		0012095C	00C00048	Buffer = 00C00048
0012EB78	68 E1 40 00 AC 00 00 00	h8E.0...		00120960	00001000	BytesToWrite = 1000 (4096.)
0012EB80	00 00 00 00 68 E1 40 00h8E.		00120964	00120A6C	pBytesWritten = 00120A6C
0012EB88	68 E1 40 00 00 00 00 000....		00120968	00000001	
0012EB90	01 67 55 E1 00 00 00 00	1...7..		0012096C	00000000	
0012EB98	18 AD BF AA 00 00 00 00	1...r..		00120970	0012F520	
0012EBA0	13 00 00 00 DA 0E 51 00	!...r..		00120974	80000E84	
0012EBA8	64 AD BF AA 5E CC 4D 80	d...r..	OEP	00120978	00000000	
0012EBB0	00 00 00 00 68 E1 40 00h8E.		0012097C	F78040FA	
0012EBB8	01 00 00 00 01 00 00 00	...0....		00120980	E12A3A30	
0012EBC0	01 00 65 00 00 00 00 00	0.e....		00120984	00140012	
0012EBC8	00 00 00 00 00 00 00 00		00120988	F78040E8	
0012EBD0	00 00 00 00 00 00 00 00		0012098C	000000AC	

0040E150	3D BEE34D18	CMP EAX,184DE3BE	
0040E162	69D5 FD0AD6F0	IMUL EDX,EBP,F0D60AFD	
0040E168	65148	DEC EAX	Superfluous prefix
0040E16A	7E B0	JLE SHORT CSAssetD.0040E11C	
0040E16C	0D F0660FF2	OR EAX,F20F66F0	
0040E171	E7 96	OUT 96,EAX	I/O command
0040E173	F0:0D 1896F03D	LOCK OR EAX,3DF09618	LOCK prefix is not allowed
0040E179	1896 F0651896	SBB BYTE PTR DS:[ESI+961865F0],DL	
0040E17F	F0:4D	LOCK DEC EBP	LOCK prefix is not allowed
0040E181	1896 F0FD53DF	SBB BYTE PTR DS:[ESI+DF53FDF0],DL	
0040E187	62C1	BOUND EAX,ECX	Illegal use of register
0040E189	D5 9B	RAD 9B	
0040E18B	B7 A2	MOV BH,0A2	
0040E18D	8D8402	LEA EAX,DWORD PTR DS:[EDX+EAX]	
0040E190	6858 57 11	IMUL EBX,DWORD PTR DS:[EAX+5711]	

Target # 2: Declan's Japanese Dictionary - Armadillo 4.xx



_Load Target:

004AC000	60	PUSHAD	
004AC001	E8 00000000	CALL djdict.004AC006	
004AC006	5D	POP EBP	
004AC007	50	PUSH EAX	
004AC008	51	PUSH ECX	
004AC009	0FCA	BSWAP EDX	
004AC00B	F702	NOT EDX	
004AC00D	9C	PUSHFD	
004AC00E	F702	NOT EDX	
004AC010	0FCA	BSWAP EDX	
004AC012	EB 0F	JMP SHORT djdict.004AC023	

_Dang Standard should have two months to find OEP. A Alt + M, set breakpoint on memory

access. Second BP CreateThread, F9, Ctrl + F9, F7, Ctrl + F9, F7. Set break point in Call ECX, F9, F7:

003CD7F8	59	POP ECX	kernel32.77E7BE2B
003CD7F9	BF D8C83D00	MOV EDI,30C8D8	
003CD7FE	8BCF	MOV ECX,EDI	
003CD800	E8 38ABFDFF	JMP 003A833D	
003CD805	84C0	TEST AL,AL	
003CD807	75 09	JNZ SHORT 003CD812	
003CD809	6A 01	PUSH 1	
003CD80B	8BCF	MOV ECX,EDI	
003CD80D	E8 5AF8FDFF	JMP 003A006C	
003CD812	B9 40BB3D00	MOV ECX,30BB40	
003CD817	C705 70903D00	MOV DWORD PTR DS:[3D9070],309D08	ASCII "RC"
003CD821	E8 28F3FFFF	CALL 003CCB4E	
003CD826	6A 00	PUSH 0	
003CD828	E8 21F3FFFF	CALL 003CCB4E	
003CD82D	A1 20CF3D00	MOV EAX,DWORD PTR DS:[30CF20]	
003CD832	59	POP ECX	
003CD833	8B15 38CF3D00	MOV EDX,DWORD PTR DS:[30CF38]	djdict.00400000
003CD839	8B3E	MOV EDI,DWORD PTR DS:[ESI]	
003CD83B	8B48 64	MOV ECX,DWORD PTR DS:[EAX+64]	
003CD83E	3348 58	XOR ECX,DWORD PTR DS:[EAX+58]	
003CD841	3348 18	XOR ECX,DWORD PTR DS:[EAX+18]	
003CD844	03CA	ADD ECX,EDX	
003CD846	85FF	TEST EDI,EDI	
003CD848	75 18	JNZ SHORT 003CD862	
003CD84A	8B50 64	MOV EDX,DWORD PTR DS:[EAX+64]	
003CD84D	FF76 14	PUSH DWORD PTR DS:[ESI+14]	
003CD850	3350 60	XOR EDX,DWORD PTR DS:[EAX+60]	
003CD853	FF76 10	PUSH DWORD PTR DS:[ESI+10]	
003CD856	3350 10	XOR EDX,DWORD PTR DS:[EAX+10]	
003CD859	FF76 0C	PUSH DWORD PTR DS:[ESI+C]	
003CD85C	2BCA	SUB ECX,EDX	
003CD85E	FFD1	CALL ECX	
003CD860	EB 18	JMP SHORT 003CD87D	
003CD862	83FF 01	CMPI EDI,1	
003CD865	75 18	JNZ SHORT 003CD87F	
003CD867	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
003CD86A	FF76 08	PUSH DWORD PTR DS:[ESI+8]	
003CD86D	6A 00	PUSH 0	
003CD86F	52	PUSH EDX	
003CD870	8B50 64	MOV EDX,DWORD PTR DS:[EAX+64]	
003CD873	3350 60	XOR EDX,DWORD PTR DS:[EAX+60]	
003CD876	3350 10	XOR EDX,DWORD PTR DS:[EAX+10]	
003CD879	2BCA	SUB ECX,EDX	
003CD87B	FFD1	CALL ECX	
003CD87D	8B08	MOV EBX,EAX	
003CD87F	5F	POP EDI	
003CD880	8BC3	MOV EAX,EBX	
003CD882	5E	POP ESI	

_OEP:

00415726	6A 60	PUSH 60	
00415728	68 78234200	PUSH djdict.00422378	
0041572D	E8 BA0E0000	CALL djdict.004165EC	
00415732	BF 94000000	MOV EDI,94	
00415737	8BC7	MOV EAX,EDI	
00415739	E8 62FFFFFF	CALL djdict.004156A0	
0041573E	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00415741	8BF4	MOV ESI,ESP	
00415743	893E	MOV DWORD PTR DS:[ESI],EDI	
00415745	56	PUSH ESI	
00415746	FF15 60F14100	CALL DWORD PTR DS:[41F160]	kernel32.GetVersionExA
0041574C	8B4E 10	MOV ECX,DWORD PTR DS:[ESI+10]	
0041574F	890D 68A64500	MOV DWORD PTR DS:[45A668],ECX	
00415755	8B46 04	MOV EAX,DWORD PTR DS:[ESI+4]	
00415758	A3 74A64500	MOV DWORD PTR DS:[45A674],EAX	
0041575D	8B56 08	MOV EDX,DWORD PTR DS:[ESI+8]	

_Den Fix the IAT. We always like to trace function under Call first to find the OEP Jmp infinity method of Cracks by Latinos or VirtualProtect breakpoint.

_Cach First disable (After trace function to call):

004165EC	68 389E4100	PUSH djdict.00419E38	
004165F1	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
004165F7	50	PUSH EAX	
004165F8	8B4424 10	MOV EAX,DWORD PTR SS:[ESP+10]	
004165FC	896C24 10	MOV DWORD PTR SS:[ESP+10],EBP	
00416600	8D6C24 10	LEA EBP,DWORD PTR SS:[ESP+10]	
00416604	2BE0	SUB ESP,EAX	
00416606	53	PUSH EBX	
00416607	56	PUSH ESI	

_Cach Second using BP VirtualProtect, Shift + F9 40 times more for the **100 J Push**

77E6169E	55	PUSH EBP
77E6169F	8BEC	MOV EBP,ESP
77E616A1	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77E616A4	FF75 10	PUSH DWORD PTR SS:[EBP+10]
77E616A7	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]
77E616AA	FF75 08	PUSH DWORD PTR SS:[EBP+08]
77E616AD	6A FF	PUSH -1
77E616AF	E8 A4B80100	CALL kernel32.VirtualProtectEx
77E616B4	5D	POP EBP
77E616B5	C2 1000	RET 0
77E616B8	837C24 04 00	CMP DWORD PTR SS:[ESP+4],0
77E616BD	74 18	JE SHORT kernel32.77E616D7
77E616BF	FF7424 08	PUSH DWORD PTR SS:[ESP+8]

_Hai Way I finished this way out of Vietnam Male ;;). You notice function: CALL DWORD PTR DS:[41F160]; kernel32.GetVersionExA not? I find it a valuable suspicious: **41F160**. Ok, restart again. Memory to dump Windows, Ctrl + G to enter 41F160, set Hardware Breakpoint Write on the Dword. Shift + F9:

77C42F48	F3:A5	REP MOVSD DWORD PTR ES:[EDI],DWORD PTR DS:[EDI]
77C42F49	FF2495 5830C477	JMP DWORD PTR DS:[EDX+4+77C43058]
77C42F4C	8BC7	MOV EAX,EDI
77C42F4E	BA 03000000	MOV EDI,3
77C42F53	83E9 04	SUB ECX,4
77C42F56	72 0C	JB SHORT msvcrt.77C42F64
77C42F58	83E0 03	AND EAX,3
77C42F5B	03C8	ADD ECX,EAX
77C42F5D	FF2495 782EC477	JMP DWORD PTR DS:[EAX+4+77C42F70]
77C42F64	FF2495 5830C477	JMP DWORD PTR DS:[ECX+4+77C43058]

_Shift + F9 times 2:

003CA82E	8B85 1009FFFF	MOV EAX,DWORD PTR SS:[EBP-26F0]	djdlot.0041F160
003CA834	83C0 04	ADD EAX,4	
003CA837	8985 1009FFFF	MOV DWORD PTR SS:[EBP-26F0],EAX	
003CA83D	E9 40FCFFFF	JMP 003CA48F	
003CA842	FF15 78333D00	CALL DWORD PTR DS:[3D3270]	kernel32.GetTickCount
003CA848	2B85 A004FFFF	SUB EAX,DWORD PTR SS:[EBP-2B60]	
003CA84E	8B8D A004FFFF	MOV ECX,DWORD PTR SS:[EBP-2B5C]	
003CA854	6BC9 32	IMUL ECX,ECX,32	
003CA857	81C1 D0070000	ADD ECX,7D0	
003CA85D	3BC1	CMPL EAX,ECX	
003CA85F	76 07	JBE SHORT 003CA868	
003CA861	C885 34085555 0	MOV BYTE PTR SS:[EBP-26C0],1	

_Cuon Up a bit:

003CA641	8378 08 00	CMP DWORD PTR DS:[EAX+8],0	
003CA645	74 49	JE SHORT 003CA690	
003CA647	68 00010000	PUSH 100	
003CA64C	8D85 54C1FFFF	LEA EAX,DWORD PTR SS:[EBP-3EAC]	
003CA652	50	PUSH EAX	
003CA653	8B85 54C2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DAC]	
003CA659	FF30	PUSH DWORD PTR DS:[EAX]	
003CA65B	E8 C879F0FF	CALL 003A2020	
003CA660	83C4 0C	ADD ESP,0C	
003CA663	8D85 54C1FFFF	LEA EAX,DWORD PTR SS:[EBP-3EAC]	
003CA669	50	PUSH EAX	
003CA66A	8D85 64C2FFFF	LEA EAX,DWORD PTR SS:[EBP-3D9C]	
003CA670	50	PUSH EAX	
003CA671	FF15 78333D00	CALL DWORD PTR DS:[3D3378]	msvcrt._stricmp
003CA677	59	POP ECX	
003CA678	59	POP ECX	
003CA679	85C0	TEST EAX,EAX	
003CA67B	75 11	JNZ SHORT 003CA68E	
003CA67D	8B85 54C2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DAC]	
003CA683	8B40 08	MOV EAX,DWORD PTR DS:[EAX+8]	

_Khua Khua, so how J portfolio. Similarly the tut before you unpack normal. Restart again, Shift + F9, enter 003CA65B, in which a. F9, to enter this function, 55 to C3 patch. Ctrl + G to enter address OEP 415,726, set in a, HE delete old. Press F9. Using LordPE dump, fix IAT, show invalid, Cut Thunks. Run test, cracked yeah!



_Các You all soft meat led to this, the same type!

Target # 3: Ace Utilities 2:50 - Armadillo 4.xx



CRC check _Mot form of Armadillo. We unpacked using the error when run the file has been mod.
Okie, we treat it!

_Load Target:

004C9A40	55	PUSH EBP	
004C9A41	8BEC	MOV EBP,ESP	
004C9A43	6A FF	PUSH -1	
004C9A45	68 A04A4E00	PUSH 0u.004E4AA0	
004C9A4A	68 18974C00	PUSH 0u.004C9718	
004C9A4F	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	SE handler installation
004C9A55	58	PUSH EAX	
004C9A56	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
004C9A5D	8BEC 58	SUB ESP,58	
004C9A60	53	PUSH EBX	
004C9A61	56	PUSH ESI	
004C9A62	57	PUSH EDI	

_ Alt + M, set breakpoint on memory access, F9. OEP:

00467C23	55	PUSH EBP	
00467C24	8BEC	MOV EBP,ESP	
00467C26	6A FF	PUSH -1	
00467C28	68 F0884800	PUSH 0u.004808F8	
00467C2D	68 827A4600	PUSH 0u.00467A82	
00467C32	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	JMP to msvrt._except_handler3
00467C38	58	PUSH EAX	
00467C39	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00467C40	8BEC 68	SUB ESP,68	
00467C43	53	PUSH EBX	
00467C44	56	PUSH ESI	
00467C45	57	PUSH EDI	
00467C46	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00467C49	330B	XOR EBX,EBX	
00467C4B	895D FC	MOV DWORD PTR SS:[EBP-4],EBX	
00467C4E	6A 02	PUSH 2	
00467C50	FF15 F84A4700	CALL DWORD PTR DS:[474AF8]	msvrt.__set_app_type
00467C56	59	POP EAX	
00467C57	8300 80EF4900	FI OR DWORD PTR DS:[49EF80],FFFFFFFF	
00467C5E	8300 84EF4900	FI OR DWORD PTR DS:[49EF84],FFFFFFFF	
00467C65	FF15 FC4A4700	CALL DWORD PTR DS:[474AFC]	msvrt.__p_fnode
00467C6B	8B00 64EF4900	MOV ECX,DWORD PTR DS:[49EF64]	
00467C71	8908	MOV DWORD PTR DS:[EAX],ECX	
00467C73	FF15 004B4700	CALL DWORD PTR DS:[474B00]	msvrt.__p_conmode
00467C79	8B00 60EF4900	MOV ECX,DWORD PTR DS:[49EF60]	
00467C7F	8908	MOV DWORD PTR DS:[EAX],ECX	
00467C81	01 044B4700	MOV ECX,DWORD PTR DS:[474B04]	

_Ghi 474AF8 remember, Ctrl + F2, Memory window, Alt + F1, HW 474AF8, Shift + F9 2 times:

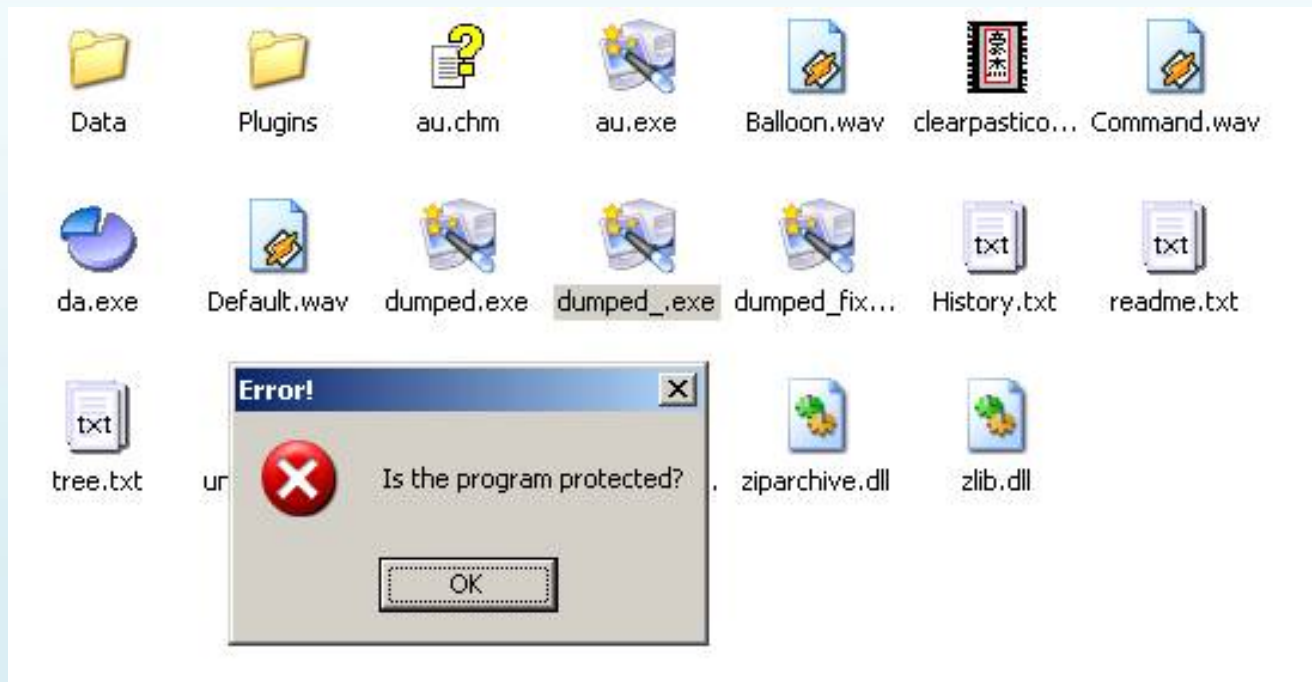
003D6088	8B85 0CC8FFFF	MOV EAX,DWORD PTR SS:[EBP-37F4]	0u.00474AF8
003D6089	83C0 04	ADD EAX,4	
003D608C	8985 0CC8FFFF	MOV DWORD PTR SS:[EBP-37F4],EAX	
003D6092	^E9 CEF0FFFF	JMP 003D6A65	
003D6097	FF15 80F03D00	CALL DWORD PTR DS:[3DF080]	kernel32.GetTickCount
003D609D	2B85 9CC4FFFF	SUB EAX,DWORD PTR SS:[EBP-3B64]	
003D60A3	8B80 A0C4FFFF	MOV ECX,DWORD PTR SS:[EBP-3B60]	
003D60A9	6BC9 32	IMUL ECX,ECX,32	
003D60AC	81C1 D0070000	ADD ECX,700	
003D60B2	3BC1	CMP EAX,ECX	

_Cuon Up:

003D68DC	8085 54C1FFFF	LEA EAX,DWORD PTR SS:[EBP-3EAC]	
003D68E2	50	PUSH EAX	
003D68E3	8B85 54C2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DAC]	
003D68E9	FF30	PUSH DWORD PTR DS:[EAX]	
003D68EB	E8 BB14FEFF	CALL 003B80AB	
003D68F0	83C4 0C	ADD ESP,0C	
003D68F3	8085 54C1FFFF	LEA EAX,DWORD PTR SS:[EBP-3EAC]	
003D68F9	50	PUSH EAX	
003D68FA	FFB5 5CC2FFFF	PUSH DWORD PTR SS:[EBP-3DA4]	
003D68C0	FF15 58F33D00	CALL DWORD PTR DS:[3DF358]	msvrt._stricmp
003D68C6	59	POP ECX	
003D68C7	59	POP ECX	
003D68C8	85C0	TEST EAX,EAX	
003D68C9	75 11	JNZ SHORT 003D6C1D	
003D68CA	8B85 54C2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DAC]	

_Restart Again, Shift + F9, enter 003D6BEB, in which a. F9, to enter this function, 55 to C3 patch.

Ctrl + G to enter OEP 00467C23 address, set in a HE, HE delete old. Press F9. LordPE Using full dump, fix IAT, show invalid, Cut Thunks. Test Run:



Chung We need to pass by CRC check. Load up dumped.exe Olly. Press F9 to notice the error, press F12, press Alt + K.

Address	Stack	Procedure / arguments	Called from	Frame
00120084	77D43C6B	Includes 7FFE0304	USER32.77D43C69	00120088
00120088	77D4B406	USER32.WaitMessage	USER32.77D4B401	00120088
001200BC	77D4D9AA	USER32.77D4B279	USER32.77D4D9A5	00120088
001200E4	77D662F4	USER32.77D4D8F6	USER32.77D662EF	001200E0
0012E09C	77D65D77	? USER32.SoftModalMessageBox	USER32.77D65D72	0012E024
0012E1E4	77D66441	? USER32.77D65C3D	USER32.77D6643C	0012E16C
0012E238	77D66529	USER32.MessageBoxTimeoutW	USER32.77D66524	0012E234
0012E26C	77D66486	? USER32.MessageBoxTimeoutA	USER32.77D66481	0012E268
0012E28C	77D6649A	? USER32.MessageBoxExA	USER32.77D66495	0012E288
0012E290	00000000	hOwner = NULL		
0012E294	00496274	Text = "Is the program protected?"		
0012E298	00496290	Title = "Error!"		
0012E29C	00000010	Style = MB_OK!MB_ICONHAND!MB_APPLMODAL		
0012E2A0	00000000	LanguageID = 0 (LANG_NEUTRAL)		
0012E2A4	004015EE	? USER32.MessageBoxA	dumped_.004015E8	
0012E2A8	00000000	hOwner = NULL		
0012E2AC	00496274	Text = "Is the program protected?"		
0012E2B0	00496290	Title = "Error!"		
0012E2B4	00000010	Style = MB_OK!MB_ICONHAND!MB_APPLMODAL		

We see _Chung nag known since dumped_.004015E8

Call stack of main thread, item 14

Address = 0012E2A4

Stack = 004015EE

Procedure / arguments =? USER32.MessageBoxA

Called from dumped_.004015E8 =

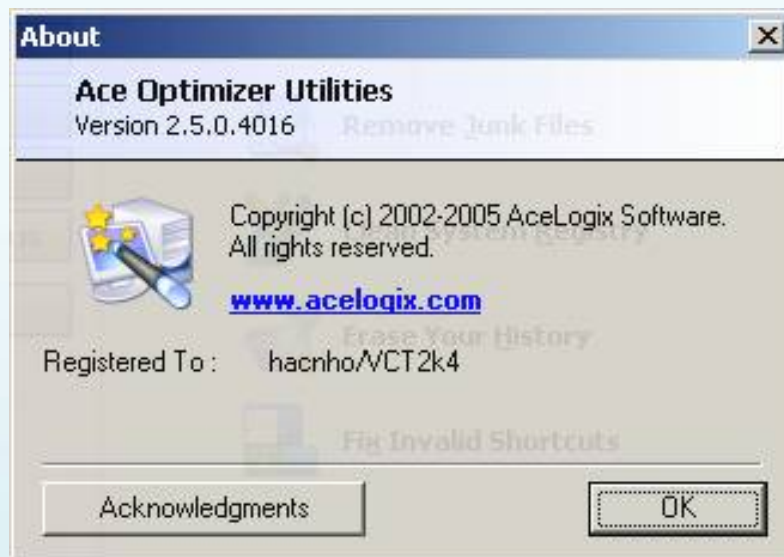
_Tro The CPU, type Ctrl + G to 004015E8:

004015CE	53	PUSH EBX	
004015CF	50	PUSH EAX	
004015D0	68 98624900	PUSH dumped_.00496298	ASCII "CERTIFICATE"
004015D5	FFD6	CALL ESI	
004015D7	85C0	TEST EAX,EAX	
004015D9	75 41	JNZ SHORT dumped_.0040161C	
004015DB	6A 10	PUSH 10	
004015DD	68 98624900	PUSH dumped_.00496298	ASCII "Error!"
004015E2	68 74624900	PUSH dumped_.00496274	ASCII "Is the program protected?"
004015E7	50	PUSH EAX	
004015E8	FF15 604E4700	CALL DWORD PTR DS:[&user32.MessageBoxA	USER32.MessageBoxA
004015EE	6A 00	PUSH 0	
004015F0	8D8D 28FFFFFF	LEA ECX,DWORD PTR SS:[EBP-D8]	
004015F6	E8 375E0000	JMP dumped_.00407432	
004015F8	8D8D 28FFFFFF	LEA ECX,DWORD PTR SS:[EBP-D8]	
00401601	C645 FC 03	MOV BYTE PTR SS:[EBP-41],3	

Patch 004015D9 75 41 JNZ SHORT dumped.0040161C to:

004015CE	53	PUSH EBX	
004015CF	50	PUSH EAX	
004015D0	68 98624900	PUSH dumped_.00496298	ASCII "CERTIFICATE"
004015D5	FFD6	CALL ESI	
004015D7	85C0	TEST EAX,EAX	
004015D9	75 41	JMP SHORT dumped_.0040161C	
004015DB	6A 10	PUSH 10	
004015DD	68 98624900	PUSH dumped_.00496298	ASCII "Error!"
004015E2	68 74624900	PUSH dumped_.00496274	ASCII "Is the program protected?"
004015E7	50	PUSH EAX	
004015E8	FF15 604E4700	CALL DWORD PTR DS:[&user32.MessageBoxA	USER32.MessageBoxA
004015EE	6A 00	PUSH 0	
004015F0	8D8D 28FFFFFF	LEA ECX,DWORD PTR SS:[EBP-D8]	
004015F6	E8 375E0000	JMP dumped_.00407432	

_ Copy to Executable> Selection, Ctrl + E, re-change EB41 to 7541, save the file. Run test, cracked!



Target # 4: File recover 5.0 - Armadillo 4.xx



_Day Is a cracked-unpack other, after they registered but unpack requires a function ArmaAccess.Dll to run. Why do we require this file, when inserted into the pack armadillo soft this function, as we unpack grass function as it should require, not really function this role at all. There are two ways to handle them is to use our library are available on the homepage pub Armadillo to complete or we patch. This format will be unpacked after GetFileSizeA used to check size, if not true, LoadLibraryA ArmaAccess.Dll file to load, via GetProcAddress to handle. We patch the GetProcAddress function.

```
FARPROC GetProcAddress (
HMODULE hModule, // handle to dll modules
LPCSTR lpProcName // name of function
);
```


00408EB9	8BCF	MOV ECX,EDI	
00408EBB	E8 9CF8FFFF	CALL D1_0040875C	
00408EC0	C705 28384200 0	MOV DWORD PTR DS:[423828],1	
00408ECA	FF35 20384200	PUSH DWORD PTR DS:[423820]	
00408ED0	FF15 9CA14100	CALL DWORD PTR DS:[<&kernel32.LoadLibraryA>]	kernel32.LoadLibraryA
00408ED6	8BF0	MOV ESI,EAX	
00408ED8	85F6	TEST ESI,ESI	
00408EDA	5F	POP EDI	
00408EDB	74 11	JE SHORT D1_00408EEE	
00408EDD	FF35 1C384200	PUSH DWORD PTR DS:[42381C]	
00408EE3	56	PUSH ESI	
00408EE4	FF15 90A14100	CALL DWORD PTR DS:[<&kernel32.GetProcAddress>]	kernel32.GetProcAddress
00408EEA	85C0	TEST EAX,EAX	
00408EEC	75 04	JNZ SHORT D1_00408EF2	
00408EEE	32C0	XOR AL,AL	
00408EF0	EB 15	JMP SHORT D1_00408F07	
00408EF2	FF7424 10	PUSH DWORD PTR SS:[ESP+10]	
00408EF6	FF7424 10	PUSH DWORD PTR SS:[ESP+10]	
00408EFA	FFD0	CALL EAX	
00408EFC	56	PUSH ESI	
00408EFD	8AD8	MOV BL,AL	
00408EFF	FF15 8CA14100	CALL DWORD PTR DS:[<&kernel32.FreeLibrary>]	kernel32.FreeLibrary
00408F05	8AC3	MOV AL,BL	
00408F07	5E	POP ESI	
00408F08	5B	POP EBX	
00408F09	C3	RETN	
DS:[0041A19C]=77E7D961 (kernel32.LoadLibraryA)			
0012DEC8	00380648	FileName = "ArmAccess.DLL"	
0012DECC	0012DF00	ASCII "SACOPHARYNX"	
0012DED0	0012E004	ASCII "49815798"	

_Dang This very interesting, appointment of a tut other because it is quite long! Here I can you protect a soft style and we use the library available for cracked.

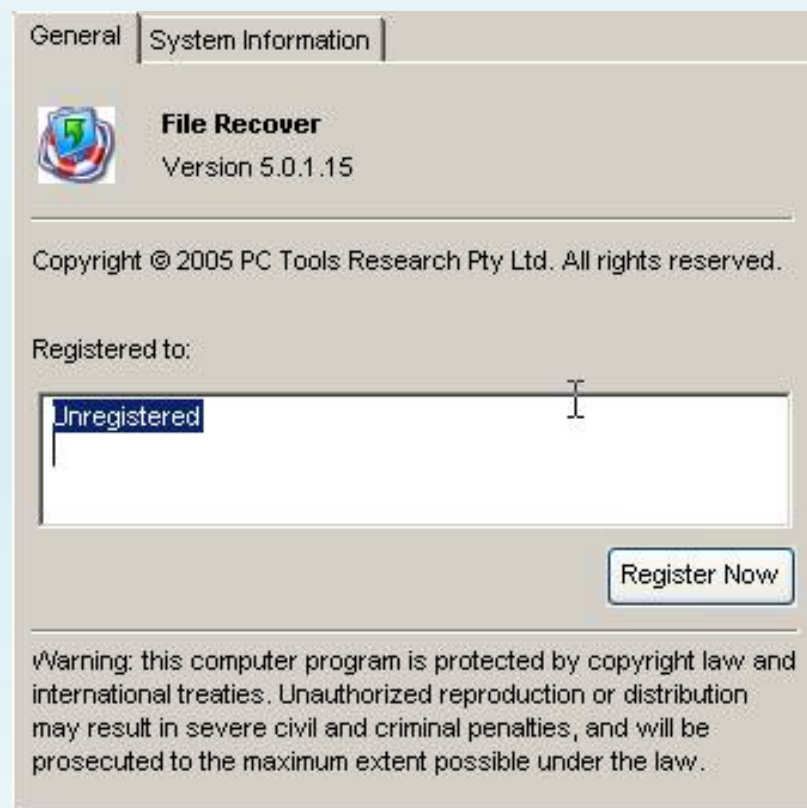
_Load Target:

00469000	60	PUSHAD	
00469001	E8 00000000	CALL FileReco.00469006	
00469006	5D	POP EBP	
00469007	5D	PUSH EAX	
00469008	51	PUSH ECX	
00469009	0FCA	BSWAP EDX	
0046900B	F7D2	NOT EDX	
0046900D	9C	PUSHFD	
0046900E	F7D2	NOT EDX	
00469010	0FCA	BSWAP EDX	
00469012	EB 0F	JMP SHORT FileReco.00469023	
00469014	B9 EB0FB8EB	MOV ECX,EBB80FEB	
00469019	07	POP ES	
0046901A	B9 EB0F90EB	MOV ECX,EB900FEB	Modification of segment regis
0046901F	005D	OR CH,CH	

_Alt + M, set breakpoint in text sections. F9, OEP:

0041FAFD	6A 74	PUSH 74	
0041FAFF	68 20414200	PUSH FileReco.00424120	
0041FB04	E8 BFFDFFFF	CALL FileReco.0041F8C8	
0041FB09	330B	XOR EBX,EBX	
0041FB0B	895D E0	MOV DWORD PTR SS:[EBP-20],EBX	
0041FB0E	53	PUSH EBX	
0041FB0F	8B3D 80114200	MOV EDI,DWORD PTR DS:[421180]	
0041FB15	FFD7	CALL EDI	
0041FB17	66:8138 405A	CMP WORD PTR DS:[EAX],5A40	
0041FB1C	75 1F	JNZ SHORT FileReco.0041FB3D	
0041FB1E	8B48 3C	MOV ECX,DWORD PTR DS:[EAX+3C]	
0041FB21	03C8	ADD ECX,EAX	
0041FB23	8139 50450000	CMP DWORD PTR DS:[ECX],4550	
0041FB29	75 12	JNZ SHORT FileReco.0041FB3D	
0041FB2B	0FB741 18	MOVZX EAX,WORD PTR DS:[ECX+18]	
0041FB2F	3D 0B010000	CMP EAX,10B	
0041FB34	74 1F	JE SHORT FileReco.0041FB55	
0041FB36	3D 0B020000	CMP EAX,20B	
0041FB3B	74 05	JE SHORT FileReco.0041FB42	
0041FB3D	895D E4	MOV DWORD PTR SS:[EBP-1C],EBX	
0041FB40	EB 27	JMP SHORT FileReco.0041FB69	
0041FB42	83B9 84000000	CMP DWORD PTR DS:[ECX+84],0E	
0041FB49	76 F2	JBE SHORT FileReco.0041FB3D	
0041FB4B	33C0	XOR EAX,EAX	
0041FB4D	3999 F8000000	CMP DWORD PTR DS:[ECX+F8],EBX	
0041FB53	EB 0E	JMP SHORT FileReco.0041FB63	
0041FB55	8379 74 0E	CMP DWORD PTR DS:[ECX+74],0E	
0041FB59	76 E2	JBE SHORT FileReco.0041FB3D	
0041FB5B	33C0	XOR EAX,EAX	
0041FB5D	3999 E8000000	CMP DWORD PTR DS:[ECX+E8],EBX	
0041FB63	0F95C0	SETNE AL	
0041FB66	8945 E4	MOV DWORD PTR SS:[EBP-1C],EAX	
0041FB69	895D FC	MOV DWORD PTR SS:[EBP-4],EBX	
0041FB6C	6A 02	PUSH 2	
0041FB6E	FF15 18134200	CALL DWORD PTR DS:[421318]	MSUCR71.__set_app_type
0041FB74	59	POP ECX	
0041FB75	830D 68884200	FI OR DWORD PTR DS:[428868],FFFFFFFF	
0041FB7C	830D 6C884200	FI OR DWORD PTR DS:[42886C],FFFFFFFF	

-At 41FB6E, click to select Follow in dump> Memory Address. Set the HW Dword. Do the same steps above. Dump, Fix IAT. Copy the file to try ArmAccess.Dll Run:



Van Not cracked open HexWorkShop dumped.exe load. Ctrl + F: unregistered

```

000215AC 556E 7265 6769 7374 6572 6564 0000 0000 5553 4552 Unregistered....USER
000215C0 4E41 4D45 0000 0000 5553 4552 4B45 5900 5245 474C NAME....USERKEY.REGL
000215D4 494D 5F4D 4158 4649 4C45 5300 4172 6D41 6363 6573 IM_MAXFILES.ArmAcces
000215E8 732E 646C 6C00 0000 4669 6C65 5265 636F 7665 7200 s.dll...FileRecover.
000215FC 4669 6C65 2052 6563 6F76 6572 0000 0000 5665 7269 File Recover....Veri
00021610 6679 4B65 7900 0000 496E 7374 616C 6C4B 6579 0000 fyKey...InstallKey..
00021624 4578 7069 7265 4375 7272 656E 744B 6579 0000 0000 ExpireCurrentKey....
00021638 7777 772E 7063 746F 6F6C 732E 636F 6D00 7777 772E www.pctools.com.www.
0002164C 7063 746F 6F6C 732E 636F 6D2F 7072 6F64 7563 742F pctools.com/product/
00021660 7375 7070 6F72 7400 7777 772E 7063 746F 6F6C 732E support.www.pctools.

```

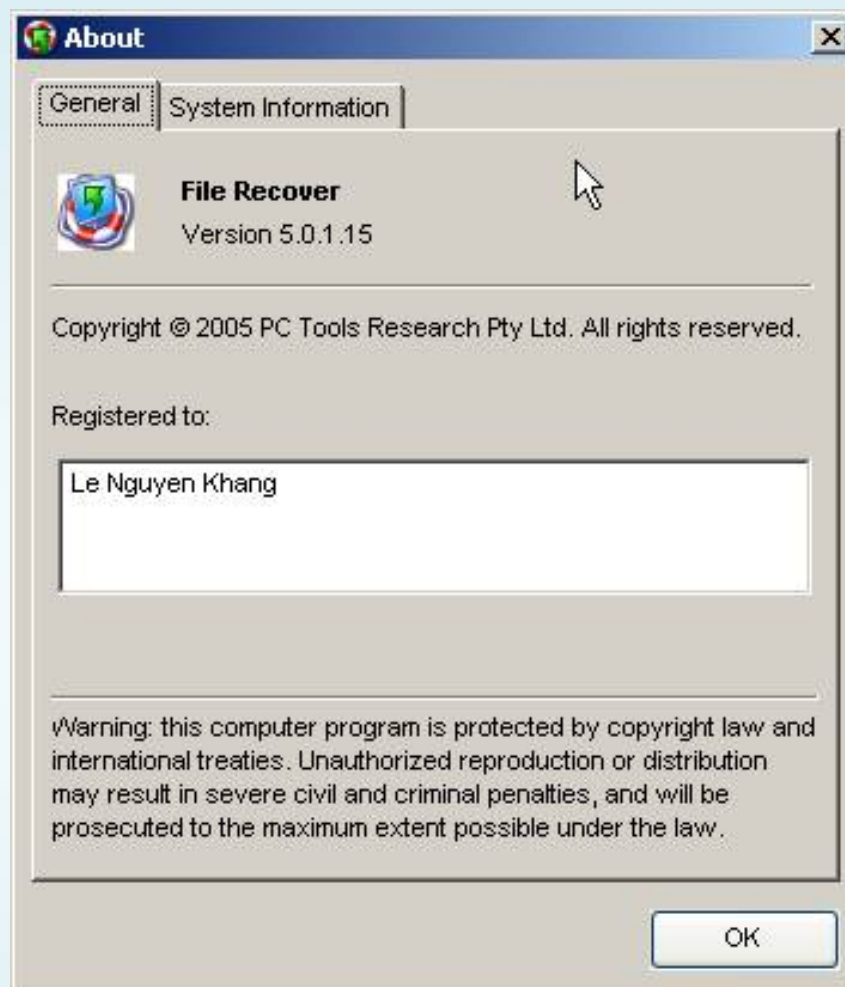
_Edit To:

```

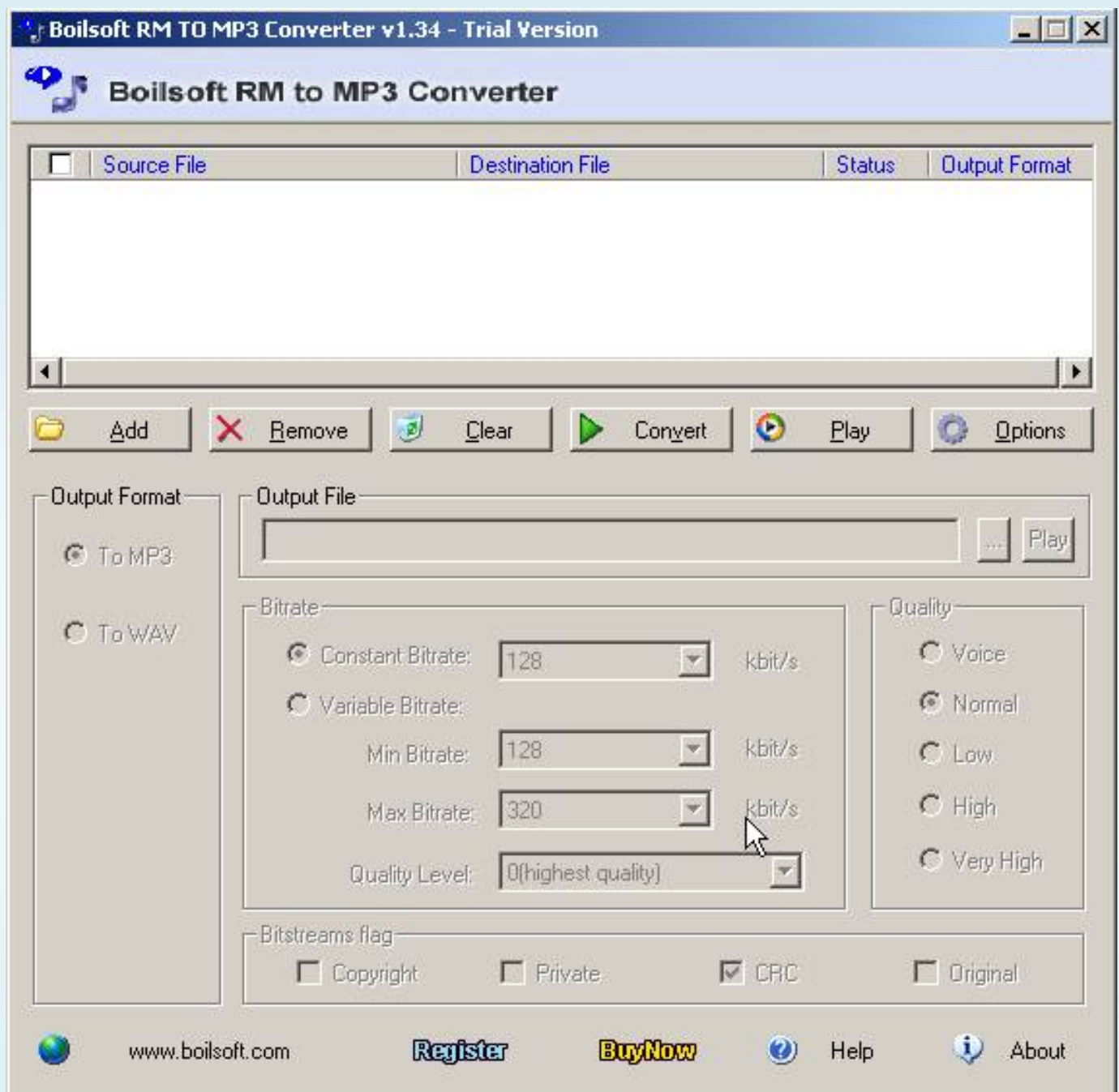
000215AC 0000 0000 0000 0000 0000 0000 0000 0000 5553 4552 .....USER
000215C0 4E41 4D45 0000 0000 5553 4552 4B45 5900 5245 474C NAME....USERKEY.REGL
000215D4 494D 5F4D 4158 4649 4C45 5300 4172 6D41 6363 6573 IM_MAXFILES.ArmAcces
000215E8 732E 646C 6C00 0000 4669 6C65 5265 636F 7665 7200 s.dll...FileRecover.
000215FC 4669 6C65 2052 6563 6F76 6572 0000 0000 5665 7269 File Recover....Veri
00021610 6679 4B65 7900 0000 496E 7374 616C 6C4B 6579 0000 fyKey...InstallKey..
00021624 4578 7069 7265 4375 7272 656E 744B 6579 0000 0000 ExpireCurrentKey....
00021638 7777 772E 7063 746F 6F6C 732E 636F 6D00 7777 772E www.pctools.com.www.
0002164C 7063 746F 6F6C 732E 636F 6D2F 7072 6F64 7563 742F pctools.com/product/
00021660 7375 7070 6F72 7400 7777 772E 7063 746F 6F6C 732E support.www.pctools.
00021674 636F 6D2F 6669 6C65 2D72 6563 6F76 6572 2F70 7572 com/file-recover/pur
00021688 6368 6173 6500 0000 4C69 7665 5570 6461 7465 2E65 chase...LiveUpdate.e

```

_Save, Edited open file:

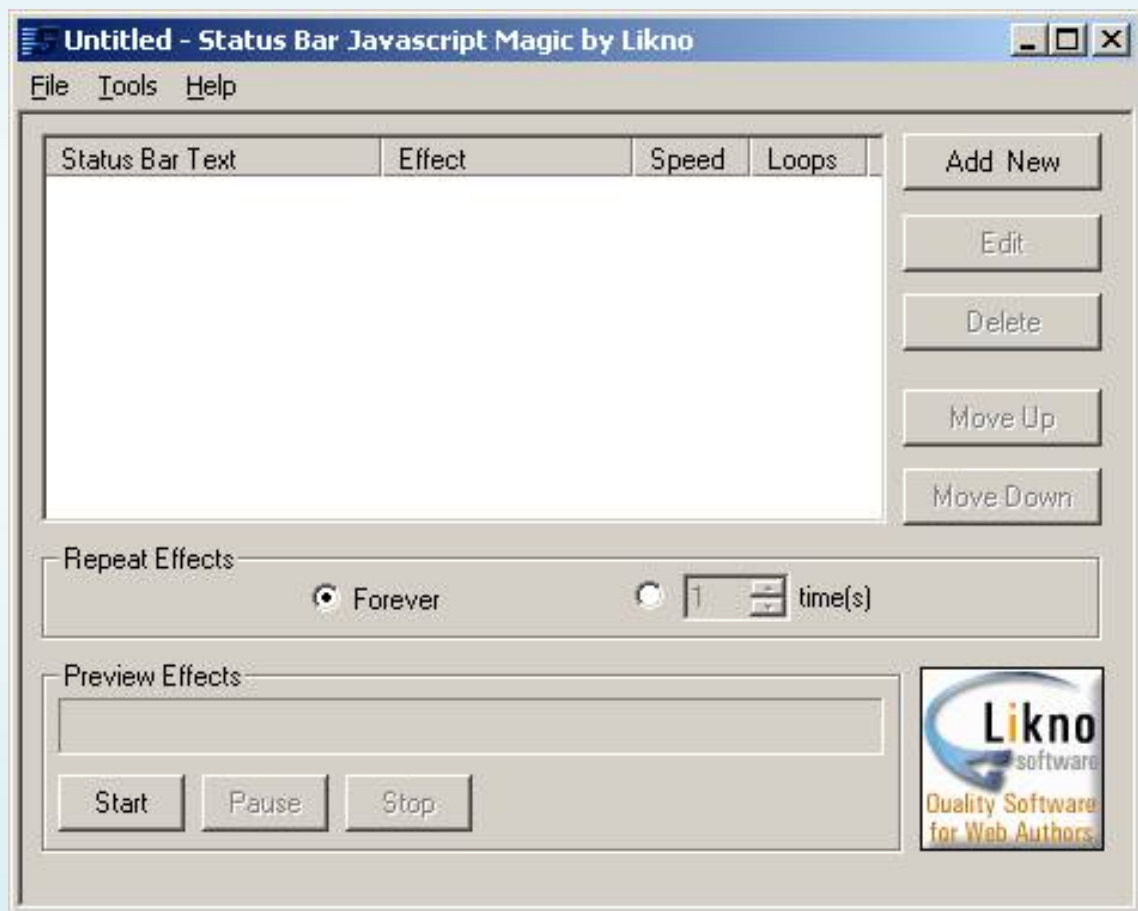


Target # 5: RM to MP3 Converter v1.32 - Armadillo 4.xx



_Ban His own practice.

Target # 6: Status Bar Javascript Magic version 1.0 - Armadillo 4.xx



_Cai Not you run 15 days. Remove it nhé!

II. Conclusion

_ Tut that this heart of some of you that I am, why medical instructions form push ebp without pushad form.

_Neu You have any soft pack with Armadillo please link to lenguyenkhang@gmail.com

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtn, hosiminh, Nilrem, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RIF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx to authors of OllyDBG.

To be continued ...

Written by [hacnho](#) (tutorial date: Tien Giang 1/09/2005)

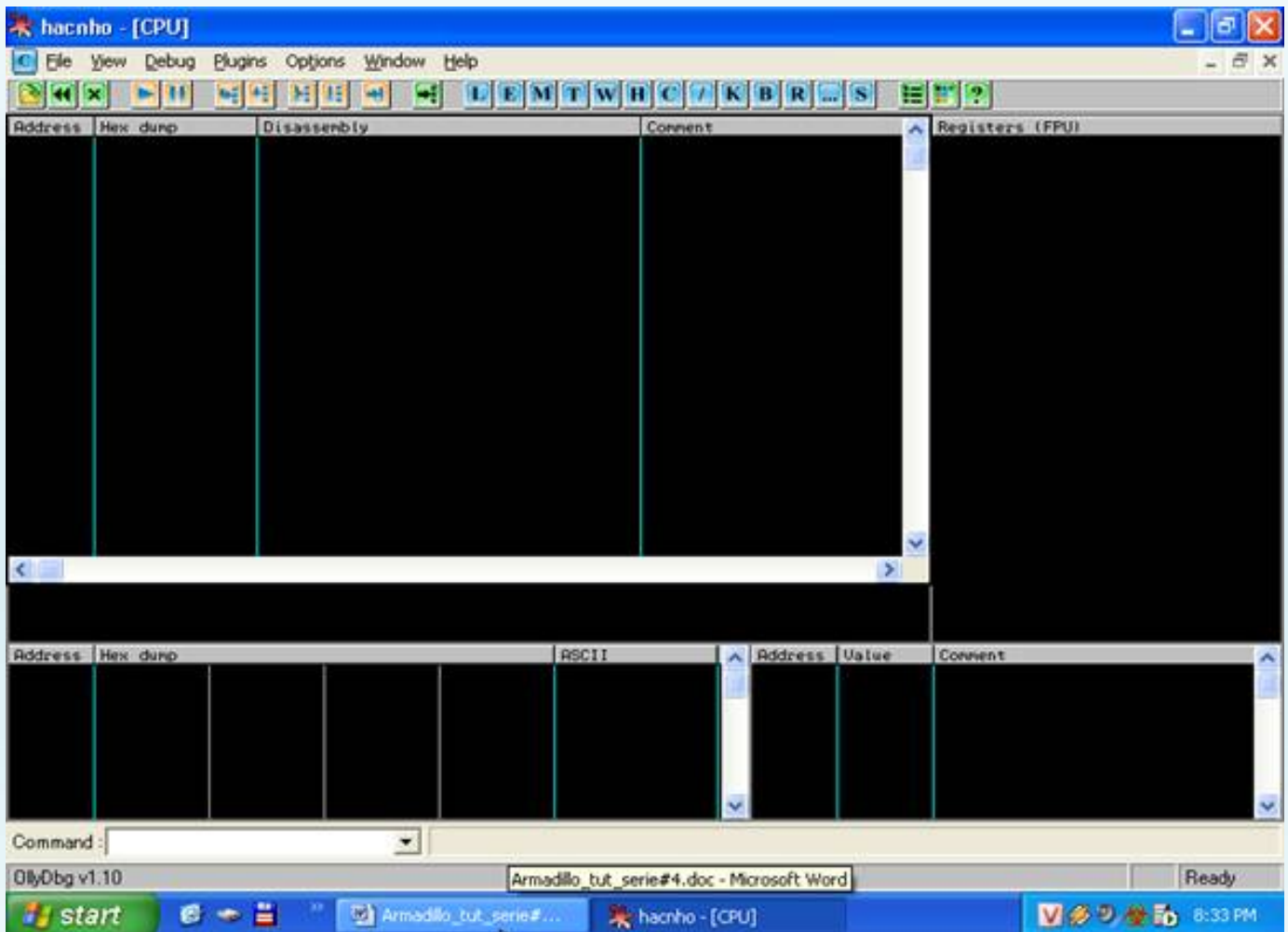
Armadillo collect sand-stone

Section 4: Armadillo x.xx-blocker + Debug CopyMemII

I. Intro

_Tui Written to this the tut, tut by the old loss. In this tut we will learn about CopyMem protection. This is a very top. If you do well with this format you have nearly reached the Upper legacy of Armadillo Unpacking only the last floor is Nanomites. You know then debug blocker, but CopyMem you heard but not to understand what it is. And the fact that you can understand, but never started to unpack it all. In fact you are not good at that we have a psychological y. Play it to the Armadillo jitters, no need for it under any form! Like me that "a cat eating a mouse in 3 seconds, asked 30 children 30 cats eat mouse how many seconds?" Almost all the respondents is 30 seconds! You also see that, through 3 tut Armadillo and not just as hard not to think?

_ You Merc new share for a OllyDBG with the ultra cool mod by tSRH again. I also down about. This is not patch you should not be used to unpack Armadillo. That the author intervention raw bạo to Resource, the ulterior by ASPack pack, can not unpack. I must have made an interface similar to the share of Merce. This smooth running with Armadillo. This is a gift and my gift to you this last tut. After tut offline I have this job, sure to 1 again sit back with a new PC. When that hope to meet again for all you see is pro.



_ What is that so called CopyMEM protection and blocking debugger what does mean?

When an Armadillo protected Application starts executing, it also creates another process. This process is called child, and the first executed exe is called father. So, when you will run it under Olly and check in process list, you will find two processes with the same name, with a different ID's. The red one in process list is the current process Olly debugs, and is the father, and the black one is the child.

So, debugger is actually blocking the Olly incapability of a Ring-3 debugger to debug the child, which is the "real" actual process that we want to make a dump, cause the child is a separate process created during runtime.

Now, when the father decides to create the child process, will allocate memory dynamically. The father will create the child with all the sections similar to him (cause is actually the same program) except two sections (maybe more in other applications): Code section. Of course, IAT section will also not be similar, cause in father is filled with 00's in the child if filled with Redirected calls.

At start, code section of child is filled with 00's. When the child has to start running from OEP, this will cause an exception to the father, cause as I said, in memory (code section) are 00's. The father will realize that and will copy the first 1000h bytes to the child. So, in this starting point, OEP will be between the memory locations that father is filling the original decrypted code. The father will also copy an encryption-decryption routine for this space in memory child, that will be responsible for encrypting during runtime of this code, when EIP of child will not be between the Ranges of just copied bytes. Then, father will copy the original 1000h bytes next to the child, and the next till it fills all of the code section child. Now, the child is to start running independant actually. So alone, runs the first block of 1000h bytes (OEP which is inside these bounds memory), and all other blocks of 1000h bytes are encrypted in child. If EIP passes the limits of that memory space (or a call or just a jmp code execution) the child 1000h bytes this encrypts and decrypts the blocks requested, that EIP now. And it is inside this thing continues till the exit of the program. So, cause memory copies of those that continually happen, and because it is the second version of that is called techik CP2 or Copymem2.

_ Noi Short Defeat CopyMemII when we need to understand that the father will write 1000 bytes to the code. In this code, to buot must contain OEP. By algorithm traces necessary, we will determine the exact OEP. Good code this code. Dump and fix IAT as splicing form code.

II. Tools

_ Công Only need the tools are:

1. OllyDBG - The best config debugger for Armadillo unpacking by hacnho.
2. PETools 1.5RC5
3. Import REConstructor 1.6 Final
4. PUPE Suite 2005 - Universal Process Patcher Pro
5. EditPluss 2:20 Regged (thanx kienmanowar)
6. API Address 1.0

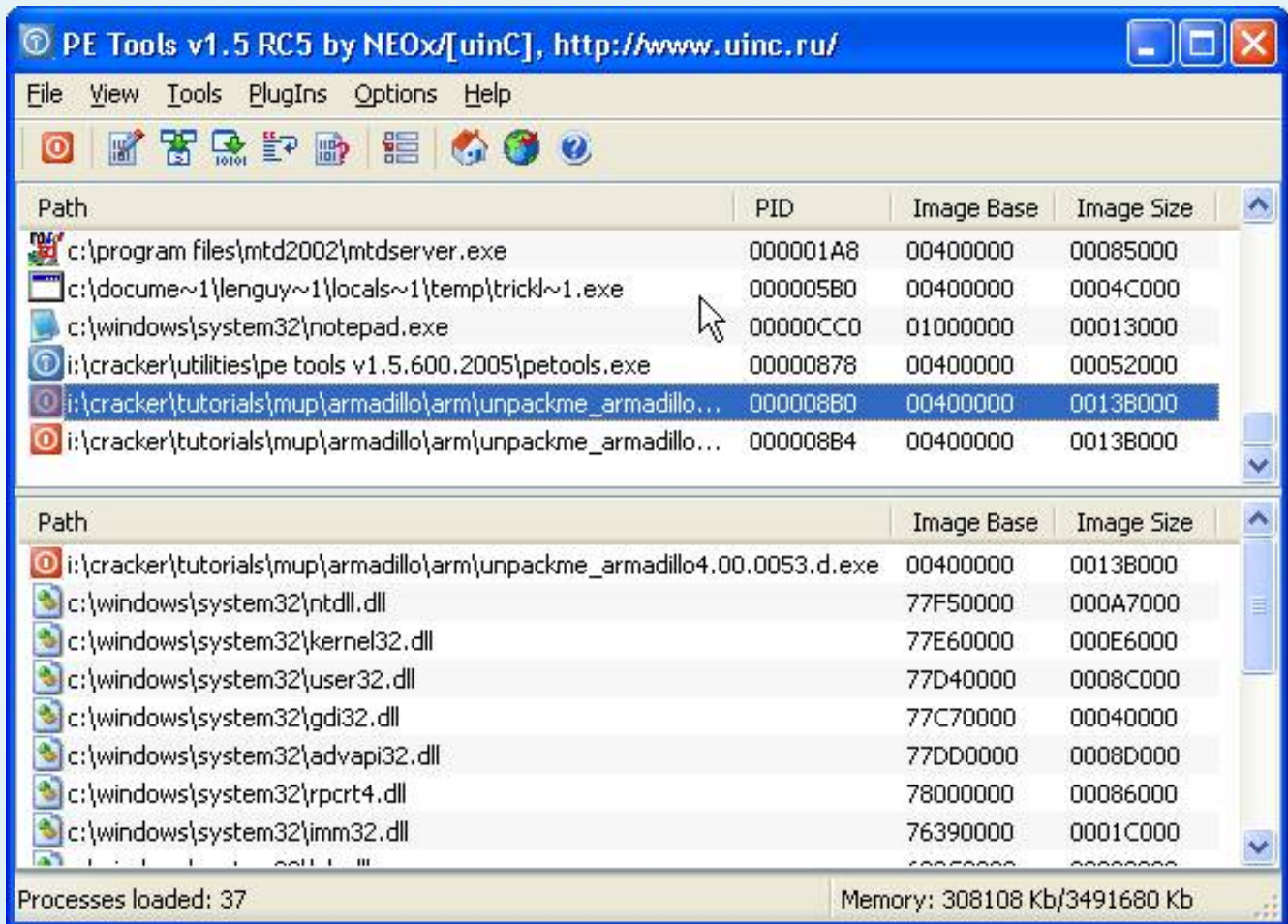
III. Unpacking

_ Truoc Unpack the bags will protect an snd Unpackme this type.

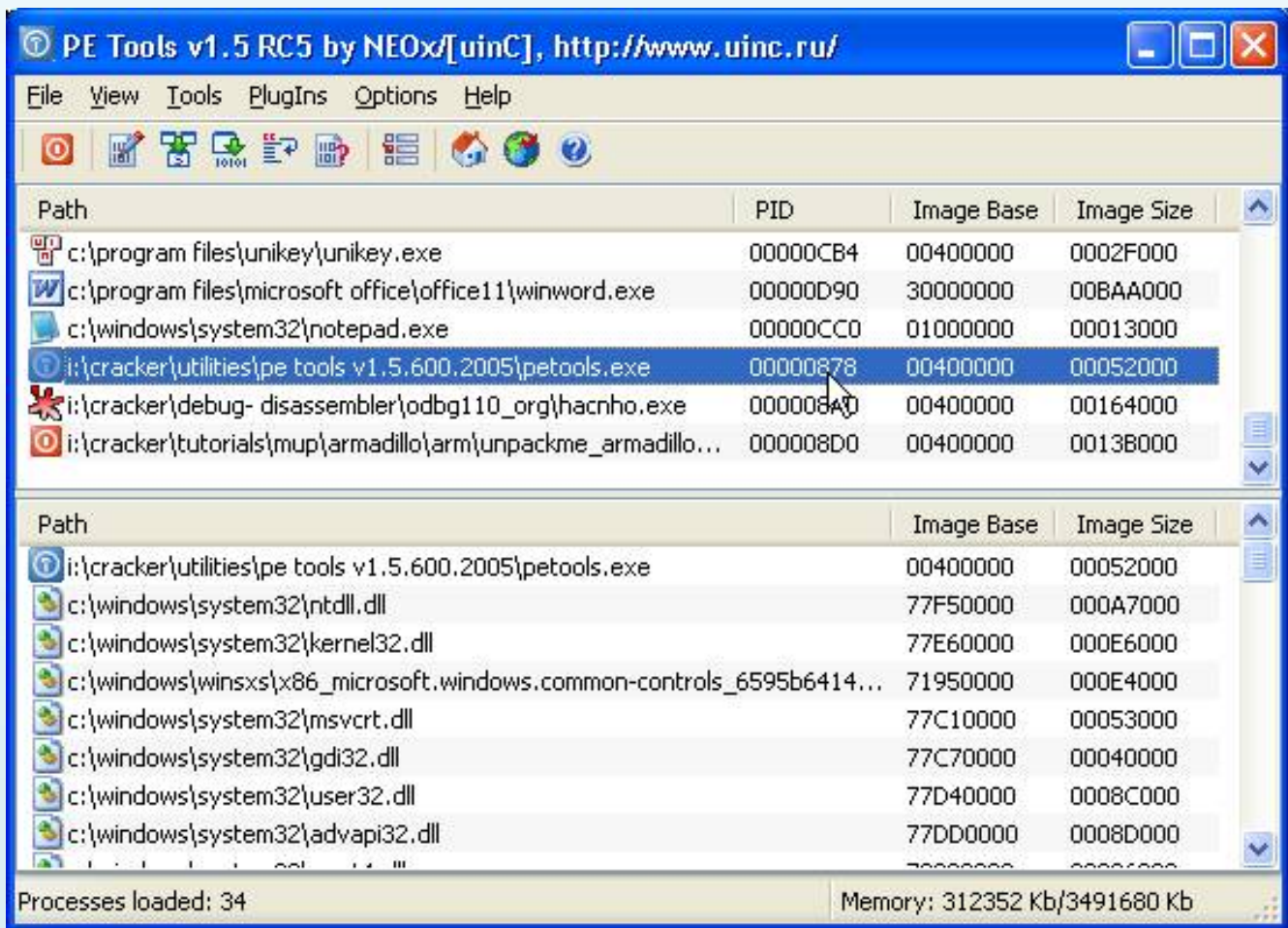
Target # 1: snd-Unpackme Armadillo 4.xx



_Dau The first of CopyMemII is when you run your target EN, it will exist two process. View:



_Khi Olly load up the target about 1 up process:



_Trong OlllyDBG, Attach to the menu:

Process	Name	Window	Path
000001BC	smss		\SystemRoot\System32\smss.exe
000001FC	csrss		\\??\C:\WINDOWS\system32\csrss.exe
00000214	winlogon	NetDDE Agent	\\??\C:\WINDOWS\system32\winlogon.exe
00000240	services		C:\WINDOWS\system32\services.exe
0000024C	lsass		C:\WINDOWS\system32\lsass.exe
000002F4	svchost		C:\WINDOWS\system32\svchost.exe
00000320	svchost		C:\WINDOWS\system32\svchost.exe
00000308	svchost		C:\WINDOWS\system32\svchost.exe
000003AC	svchost		C:\WINDOWS\system32\svchost.exe
00000414	spoolsv		C:\WINDOWS\system32\spoolsv.exe
00000510	Explorer	SysFader	C:\WINDOWS\Explorer.EXE
00000584	DkService		C:\Program Files\Executive Software\Diskeeper\DkService.exe
000005A4	inetinfo		C:\WINDOWS\System32\inetrv\inetinfo.exe
0000056C	MDM		C:\Program Files\Common Files\Microsoft Shared\US7DEBUG\MDM.EXE
000005D8	sqlservr		C:\PROGRAM FILES\Microsoft SQL Server\80\Tools\Binn\sqlservr.exe
0000061C	taskswit		C:\WINDOWS\System32\taskswitch.exe
00000624	fast	Default IME	C:\WINDOWS\System32\fast.exe
0000062C	ctfmon	TF FloatingLangBar_WndTitl	C:\WINDOWS\System32\ctfmon.exe
00000644	sqlnangr	SQL Server Service Manager	C:\Program Files\Microsoft SQL Server\80\Tools\Binn\sqlnangr.exe
00000604	StarWind		C:\Program Files\Alcohol Soft\Alcohol 120\StarWind\StarWindService.exe
0000069C	udfmg		C:\WINDOWS\System32\udfmg.exe
000006FC	Fast		C:\WINDOWS\System32\Fast.exe
00000764	CrackerU	Cracker Utils	I:\Project DELPHI7\My Project\CrackerUtils\CrackerUtils.exe
00000878	PETools	PE Tools v1.5 RC5 by NEOx	I:\CrackerUtilities\PE Tools v1.5.600.2005\PETools.exe
000008D0	UnPackMe		I:\CrackerTutorials\MUP\Armadillo\arm\UnPackMe_Armadillo4.00.00S3.d.exe
000008DC	winamp	Winamp Playlist Editor	C:\Program Files\Winamp\winamp.exe
00000C94	Snagit32	Paint Tools	C:\Program Files\TechSmith\Snagit 7\Snagit32.exe
00000C9C	TSCHelp	C:\Program Files\TechSmith	C:\Program Files\TechSmith\Snagit 7\TSCHelp.exe
00000CB4	UniKey	UniKey 3.62	C:\Program Files\UniKey\UniKey.exe
00000CC0	NOTEPAD	CP2+DebugBlocker.txt - Not	C:\WINDOWS\system32\notepad.exe
00000D90	WINWORD	CiceroUIWndFrame	C:\Program Files\Microsoft Office\OFFICE11\WINWORD.EXE

_Khi Begin to familiarize yourself with Armadillo, right on the Wealth is a soft-Lab Developer 3.0 mad I want to, you clearly see it has two process hat, do not Attach. Why so, why so: D. This is a feature of CopyMemII.

_Dau The second is our considered view it is 1000 bytes to write code that contains the child OEP process or not?

_Load Target to Olly. Alt + F1, BP WaitForDebugEvent. F9:

Address	Value	Comment
0012BCB8	0047E8BF	CALL to WaitForDebugEvent from UnPackMe.
0012BCBC	0012CD90	pDebugEvent = 0012CD90
0012BCC0	000003E8	Timeout = 1000. ms
0012BCC4	0012FF2C	
0012BCC8	00000000	
0012BCCC	7FFDF000	
0012BCD0	00000000	
0012BCD4	00000000	
0012BCD8	00000000	
0012BCDC	6A0B974E	
0012BCE0	00000000	

_Ta See address DebugEvent process by creating a father 0012CD90. Remember take it. Ctrl + F2 restart Olly, Alt + F1, BP WriteProcessMemory, Ctrl + G: 0012CD90. F9 first:

Address	Hex dump	ASCII	Address	Value	Comment
0012CD90	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B988	0048204C	CALL to WriteProcessMemory from UnPackMe.
0012CD94	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B98C	0000004C	hProcess = 0000004C
0012CD98	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B990	0048E743	Address = 48E743
0012CD9C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B994	0012BCA8	Buffer = 0012BCA8
0012CDA0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B998	00000002	BytesToWrite = 2
0012CDA4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B99C	0012BCAC	pBytesWritten = 0012BCAC
0012CDA8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9A0	00120248	Unicode "Kernel32.dll"
0012CDA4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9A4	00002710	
0012CDA8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9A8	7FFDF000	
0012CDB0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9AC	77F6379E	RETURN to ntdll.77F6379E from ntdll.77F78
0012CDB4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9B0	0012B914	
0012CDB8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9B4	77F748CB	RETURN to ntdll.77F748CB from ntdll.77F74
0012CDB4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9B8	00140000	
0012CDB8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9BC	00000000	
0012CDB4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9C0	00143650	

_F9 Second:

Address	Hex dump	ASCII	Address	Value	Comment
0012CD90	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B988	0048204C	CALL to WriteProcessMemory from UnPackMe.
0012CD94	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B98C	0000004C	hProcess = 0000004C
0012CD98	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B990	0048E743	Address = 48E743
0012CDA0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B994	00489E34	Buffer = UnPackMe.00489E34
0012CDA4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B998	00000002	BytesToWrite = 2
0012CDA8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B99C	0012BCAC	pBytesWritten = 0012BCAC
0012CDB0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9A0	0012024C	Unicode "Kernel32.dll"
0012CDB4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9A4	00002710	
0012CDB8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9A8	7FFDF000	
0012CDB4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9AC	00482E70	RETURN to UnPackMe.00482E70 from kernel32
0012CDB8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9B0	00000000	
0012CDB4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9B4	0012B9E8	
0012CDB8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9B8	00010001	
0012CDB4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9BC	00000000	
0012CDB8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9C0	00000038	
0012CDB4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012B9C4	00000020	

_F9 Third (to view more, select Long> Address).

Address	Value	Comment	Address	Value	Comment
0012CD90	00000001		0012B988	0048204C	CALL to WriteProcessMemory from UnPackMe.
0012CD94	00000940		0012B98C	0000004C	hProcess = 0000004C
0012CD98	00000974		0012B990	00427000	Address = 427000
0012CD9C	00000001		0012B994	00383070	Buffer = 00383070
0012CDA0	00000000		0012B998	00001000	BytesToWrite = 1000 (4096.)
0012CDA4	00000000		0012B99C	0012BC74	pBytesWritten = 0012BC74
0012CDA8	00427100	UnPackMe.00427100	0012B9A0	00000000	
0012CDA4	00000002		0012B9A4	00000268	
0012CDB0	00000000		0012B9A8	7FFDF000	
0012CDB4	00427100	UnPackMe.00427100	0012B9AC	00000000	
0012CDB8	00427100	UnPackMe.00427100	0012B9B0	00000024	
0012CDB4	00000000		0012B9B4	00480415	UnPackMe.00480415
0012CDB8	E1539C01		0012B9B8	00000069	
0012CDB4	00000000		0012B9BC	00010909	
0012CDB8	6A0B9720		0012B9C0	03EF8656	
0012CDB4	00000000		0012B9C4	77F755E2	RETURN to ntdll.77F755E2

_Nhan Alt + K to stack we see:

Call stack of main thread			
Address	Stack	Procedure / arguments	Called from
0012BB58	00482857	? kernel32.WriteProcessMemory	UnPackMe.00482851
0012BB5C	0000004C	hProcess = 0000004C	
0012BB60	00427000	Address = 427000	
0012BB64	00383070	Buffer = 00383070	
0012BB68	00001000	BytesToWrite = 1000 (4096.)	
0012BB6C	0012BC74	pBytesWritten = 0012BC74	

What _Thay through this information. OEP we = 4271B0. Process father will write 1000 bytes to process from the start address 427000. Retrieved 4271B0-427000 = 1B0 <4096 to 432 <1000. Clearly OEP lot in about 1000 bytes.


_Vay Function call to write this in 1000 where bytes. Need to know it to submit it: D.

_Tro The main problem! You are here:

Address	Hex dump	Disassembly	Comment
77E61A94	55	PUSH ESP	
77E61A95	8BEC	MOV EBP,ESP	
77E61A97	51	PUSH ECX	
77E61A98	51	PUSH ECX	
77E61A99	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]	
77E61A9C	53	PUSH EBX	
77E61A9D	8B5D 14	MOV EBX,DWORD PTR SS:[EBP+14]	
77E61AA0	56	PUSH ESI	
77E61AA1	8B35 B012E627	MOV ESI,DWORD PTR DS:[<ntdll.NtProtect<ntdll.ZuProtectVirtualMemoc	
77E61AA7	57	PUSH EDI	
77E61AA8	8B7D 08	MOV EDI,DWORD PTR SS:[EBP+8]	
77E61AAB	8945 F8	MOV DWORD PTR SS:[EBP+8],EAX	
77E61AAE	8D45 14	LEA EAX,DWORD PTR SS:[EBP+14]	
77E61AB1	58	PUSH EAX	
77E61AB2	6A 04	PUSH 4	
77E61AB4	8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]	
77E61AB7	58	PUSH EAX	
77E61AB8	8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]	
77E61ABE	58	PUSH EAX	
77E61ABF	57	PUSH EDI	
77E61AC0	895D FC	MOV DWORD PTR SS:[EBP-4],EBX	
77E61AC6	FFD6	CALL ESI	
77E61AC7	85C0	TEST EAX,EAX	
77E61AC8	0FBC 80000000	JNZ kernel32.77E61B52	
77E61ACA	8D45 14	MOV EAX,DWORD PTR SS:[EBP+14]	

Address	Value	Comment
004B3000	77C73607	60132.DeleteDC
004B3004	77C7355C	60132.RealizePalette
004B3008	77C74E8E	60132.SelectPalette
004B300C	77C77000	60132.CreateDCR
004B3010	77C7586C	60132.CreatePalette
004B3014	77C7305F	60132.DeleteObject
004B3018	77C73C6E	60132.BitBit
004B301C	77C71B98	60132.SelectObject
004B3020	77C72008	60132.CreateCompatibleDC
004B3024	77C75C1A	60132.CreateDIBitmap
004B3028	00000000	
004B302C	77E7391C	kernel32.GlobalUnlock
004B3030	77E70424	kernel32.GlobalLock
004B3034	77E7569E	kernel32.GlobalAlloc
004B3038	77E7A29B	kernel32.GetTickCount
004B303C	77E70950	kernel32.NtDeviceIoControl

Address	Value	Comment
0012BB58	0000004C	hProcess = 0000004C
0012BB5C	00427000	Address = 427000
0012BB60	00383070	Buffer = 00383070
0012BB64	00001000	BytesToWrite = 1000 (4096.)
0012BB68	0012BC74	pBytesWritten = 0012BC74
0012BB6C	00000000	
0012BB70	00000000	
0012BB74	77FDF000	
0012BB78	77FDF000	
0012BB7C	00000000	
0012BB80	00000024	
0012BB84	0048D415	UnPackMe.0048D415
0012BB88	00000069	
0012BB8C	00010709	
0012BB90	FFFFFFFF	
0012BB94	77E755E7	RETURN to ntdll.77E755E7

_Xoa Breakpoint WriteProcessMemory go: BC WriteProcessMemory. The purpose we need to find the call, temporarily called Magic Call .

+ F9 _Ctrl execute till return. You will be here:

The screenshot shows a debugger interface with the following components:

- Assembly Window:** Displays instructions at various addresses. Key instructions include:
 - `MOV ESI, 00000000`
 - `XOR EAX, EAX`
 - `LEA ECX, DWORD PTR SS:[EBP+14]`
 - `PUSH ECX`
 - `PUSH EAX`
 - `LEA EAX, DWORD PTR SS:[EBP-4]`
 - `PUSH EAX`
 - `LEA EAX, DWORD PTR SS:[EBP-8]`
 - `PUSH EAX`
 - `PUSH EDI`
 - `CALL ESI`
 - `XOR ESI, ESI`
 - `PUSH 00000000`
 - `CALL kernel32.77E7A466`
 - `MOV EAX, ESI`
 - `SHORT kernel32.77E61B20`
 - `PUSH EAX`
 - `CALL kernel32.77E7A466`
 - `SHORT kernel32.77E61B2E`
 - `LEA ECX, DWORD PTR SS:[EBP+14]`
 - `PUSH ECX`
 - `PUSH EAX`
- Registers (FPU) Window:** Shows the state of registers. EIP is `77E61B24` with comment `kernel32.77E61B24`. Other registers like EAX, ECX, EDI are shown with their current values.
- Stack Area:** Below the assembly window, a stack area is visible with addresses and values. A comment indicates a return to `00482857 (UnPackMe.00482857)`.

Stack area _Trong you roll the mouse to meet the second return from the first return is:
 0012BB58 00482857 Return to UnPackMe.00482857 from kernel32 WriteProcessMemory
 -We see the need to find is:

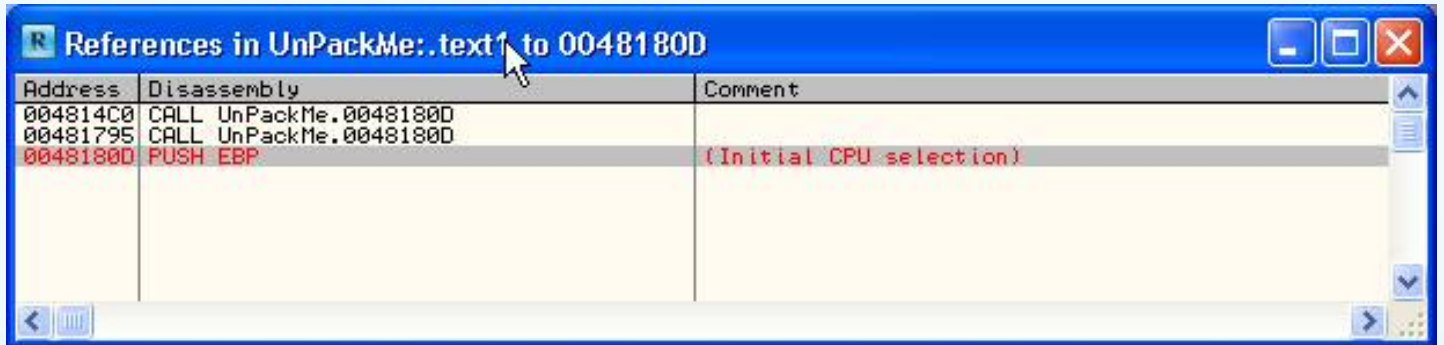
Address	Value	Comment
0012BC70	00384070	
0012BC74	00001000	
0012BC78	00384070	
0012BC7C	0012BCB0	
0012BC80	004814C5	RETURN to UnPackMe.004814C5 from UnPackMe.0048180D
0012BC84	00000026	
0012BC88	00382784	
0012BC8C	00000000	
0012BC90	0000000B	
0012BC94	00000268	
0012BC98	7FFDF000	
0012BC9C	00000000	
0012BCA0	0012C318	
0012BCA4	0012C318	
0012BCA8	00002710	

_Ghi Memory address 48180D. Back to the CPU, Ctrl + G: 48180D.

Address	Hex dump	Disassembly	Comment
00481800	55	PUSH EBP	
0048180E	8BEC	MOV EBP, ESP	
00481810	81EC 00010000	SUB ESP, 100	
00481816	53	PUSH EBX	
00481817	56	PUSH ESI	
00481818	57	PUSH EDI	
00481819	8B45 08	MOV EAX, DWORD PTR SS:[EBP+8]	
0048181C	C1E0 0C	SHL EAX, 0C	
0048181F	8B0D 349F4B00	MOV ECX, DWORD PTR DS:[4B9F34]	UnPackMe.00401000
00481825	03C8	ADD ECX, EAX	
00481827	894D EC	MOV DWORD PTR SS:[EBP-14], ECX	
0048182A	8B15 5C9F4B00	MOV EDX, DWORD PTR DS:[4B9F5C]	
00481830	8955 FC	MOV DWORD PTR SS:[EBP-4], EDX	
00481833	A1 5C9F4B00	MOV EAX, DWORD PTR DS:[4B9F5C]	
00481838	05 00100000	ADD EAX, 1000	
0048183D	8B45 E4	MOV DWORD PTR SS:[EBP-C], EAX	

_Tai Why we do this. Understand that prior to 1000 bytes to write code can encrypt our OEP, WriteProcessMemory function should point to Call a function that I called Magic Call. Why do I Ctrl + F9 because I want the values returned by WriteProcess function. We need value for the first time I was about to write is 1000 bytes call (I F9 3 times, if you have not run any time they choose to return value 3). C'est trop simple: d.

_ After coming here, we need to know who point to address this (particular function call any point to this address, one of which is the magic call). At 48180D you press Ctrl + R:



_Vay Is the second call is in the service. According to the time you submit nhĩ now? NOP bay is to dust immediately. I also know which chả lun: D. Jokes phone, you also remember our return address not just find:

0012BC80 | 004814C5 Return to UnPackMe.004814C5 from UnPackMe.0048180D

4814C5 _Thay have little relevance to what we choose it and what the charge! How can that be selected because it was called then, the need to destroy thẳng is 481,795. Hà too difficult to understand, very simple, like fig tree, the want bóp must springy, the leaderboard role intermediaries. Indeed, the value of only 4814C5 role intermediaries to function WriteProcessMemory called up. Key points are located at 481,795. You need to patch this place functions. Double click on

References in UnPackMe. Text1 to 0048180D, Item 1

Address = 00481795

Disassembly = CALL UnPackMe.0048180D

_Ta Here:

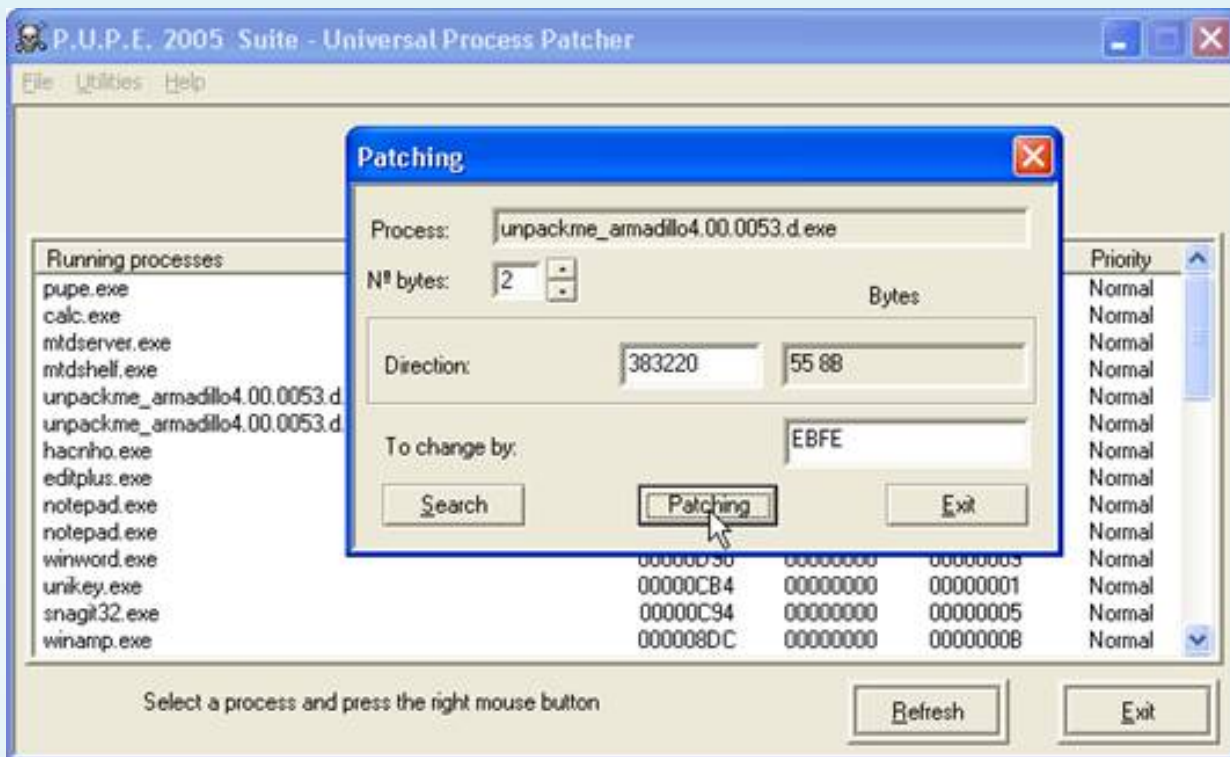
Address	Hex dump	Disassembly	Comment
00481795	. E8 73000000	CALL UnPackMe.0048180D	
0048179A	. 83C4 0C	ADD ESP,0C	
0048179D	. 50	PUSH EAX	
0048179E	. F7D0	NOT EAX	
004817A0	. 0FC8	BSWAP EAX	
004817A2	. 58	POP EAX	
004817A3	. 73 00	JNB SHORT UnPackMe.004817A5	
004817A5	> 9C	PUSHFD	
004817A6	. 60	PUSHAD	
004817A7	. EB 2B	JNB SHORT UnPackMe.004817D4	
004817A9	. D2	DB D2	
004817AA	. 70	DB 70	CHAR 'p'

_Nop It:

Address	Hex dump	Disassembly	Comment
00481795	90	NOP	
00481796	90	NOP	
00481797	90	NOP	
00481798	90	NOP	
00481799	90	NOP	
0048179A	. 83C4 0C	ADD ESP,0C	
0048179D	. 50	PUSH EAX	
0048179E	. F7D0	NOT EAX	
004817A0	. 0FC8	BSWAP EAX	
004817A2	. 58	POP EAX	
004817A3	~ 73 00	JNB SHORT UnPackMe.004817A5	
004817A5	> 9C	PUSHFD	
004817A6	. 60	PUSHAD	
004817A7	~ EB 2B	JNB SHORT UnPackMe.004817D4	

_Viec Need to do next is we sever the relationship between father and child by creating an endless repeats. How, Wow, the question is or. You also remember address Buffer contains bytes to patch the child does not, we will go to the point of entry for Buffer opcode Jump EIP as endless loop on this EP's buffer (ie, for it to jump itself J , jump hoài, jump forever to boot, tolerable). What is important is how to find EP. Fairly simple! You also remember I value except 4271b0-1b0 = 427000 is not? Public value to address Buffer to be one of the EP Buffer to patch. Thus we have 1B0 +383070 = 383220. Ctrl + G to address this patch, No no. No pro tí time. We are REA, we are pro J. Using the Tools to go pro, yeah! I used PUPE Suite 2005 - Universal Process Patcher.

_Mo Pupe up, other than normal when we choose Attach process without red color, but when this type of pro patch we need to know the red color of the PID process. After PID remember, right click in the process Pupe, select num Bytes of 2, enter the address of Buffer EP, click search, we need to patch into EBFE. Note, this is a very powerful tools by Game Cracking!



This _Luc at 383,220 will be replaced by:

Address	Hex dump	Disassembly	Comment
00383220	- EB FE	JMP SHORT 00383220	
00383222	EC	IN AL,DX	I/O command
00383223	6A FF	PUSH -1	
00383225	68 600E4500	PUSH 450E60	
0038322A	68 C8924200	PUSH 4292C8	
0038322F	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00383235	50	PUSH EAX	
00383236	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
0038323D	83C4 A8	ADD ESP,-58	
00383240	53	PUSH EBX	
00383241	56	PUSH ESI	
00383242	57	PUSH EDI	

_Luc You have a breakpoint in the kernel32 WriteProcessMemory. Bc it, then set a breakpoint WaitForDebugEvent BP, F9.Tai address 0012BCBC [in Stack] -> right click> Follow in Disassembler.

The screenshot shows the OllyDbg interface. The main window displays assembly code for kernel32.dll. A context menu is open over the 'Follow in Disassembler' option. The menu includes the following options:

- Address
- Show ASCII dump
- Show UNICODE dump
- Lock stack
- Copy to clipboard (Ctrl+C)
- Modify
- Edit (Ctrl+E)
- Push DWORD
- Pop DWORD
- Search for address
- Search for binary string (Ctrl+B)
- Go to EBP
- Go to expression (Ctrl+G)
- Follow in Disassembler (Enter)**
- Follow in Dump
- Appearance

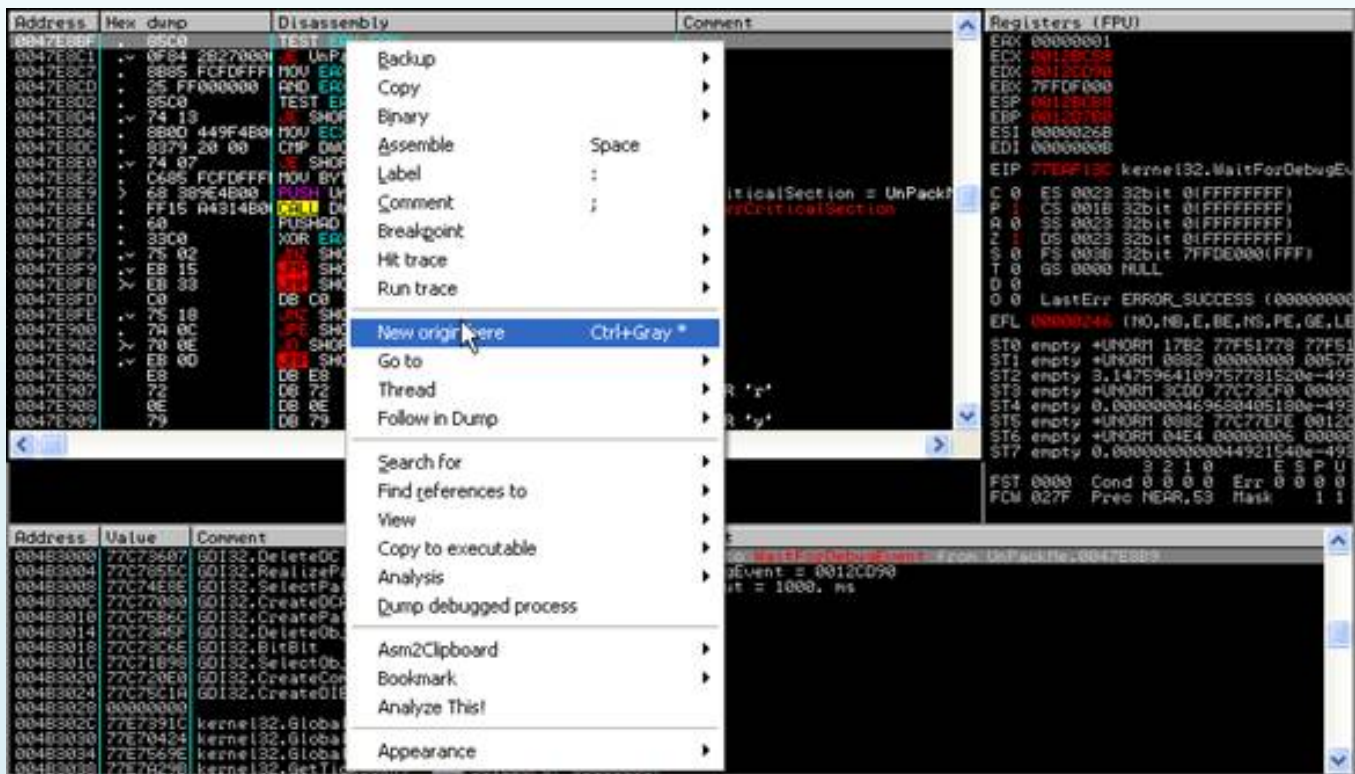
The background shows the following assembly code:

```

77EAF130 8BEC      MOV EBP,ESP
77EAF13F 83EC 68   SUB ESP,68
77EAF142 56        PUSH ESI
77EAF143 FF75 0C   PUSH DWORD PTR SS:[EBP+C]
77EAF146 0D45 F8   LEA EAX,DWORD PTR SS:[EBP-8]
77EAF149 59        PUSH EAX
77EAF14F E9 5B01FCFF CALL kernel32.77E7A2AA
77EAF151 56        PUSH ESI
77EAF152 8D45 98   LEA EAX,DWORD PTR SS:[EBP-68]
77EAF155 59        PUSH EAX
77EAF156 E8 CFB60100 CALL <JMP.&ntdll.ObgUIWaitStateChange>
77EAF158 3D 01010000 CMP EAX,101
77EAF160 74 EF     JE SHORT kernel32.77EAF151
77EAF162 3D C8000000 CMP EAX,0C0
77EAF167 74 E8     JE SHORT kernel32.77EAF151
77EAF169 85C0      TEST EAX,EAX
77EAF16B 7D 07     JGE SHORT kernel32.77EAF174
77EAF16D 3D 02000140 CMP EAX,40010002
77EAF172 75 26     JBE SHORT kernel32.77EAF19A
77EAF174 3D 02010000 CMP EAX,102
77EAF179 75 0E     JBE SHORT kernel32.77EAF189
77EAF17B 6A 79     PUSH 79
77EAF17D E9 2E62FCFF CALL kernel32.77E7A3B0
77EAF182 33C0      XOR EAX,EAX
  
```

The bottom of the screenshot shows a list of memory addresses and their values, along with a command line: `BP WaitForDebugEvent`.

_Tai Here, it should set an origin here:



_Ta Found:

Address	Hex dump	Disassembly	Comment
0047E8A7	DB DB	DB DB	
0047E8A8	7A 7A	DB 7A	CHAR 'z'
0047E8A9	F0 F0	DB F0	
0047E8AA	A0 A0	DB A0	
0047E8AB	> 3361 68	XOR ESP,DWORD PTR DS:[ECX+68]	
0047E8AE	? E8 0300008B	CALL 8B47E8B6	
0047E8B3	? 95	XCHG EAX,EBP	
0047E8B4	? DCF5	FDIUR ST(5),ST	
0047E8B6	? FFFF	Unknown command	
0047E8B8	. 52	PUSH EDX	pDebugEvent
0047E8B9	. FF15 E0304B00	CALL DWORD PTR DS:[<&KERNEL32.WaitForDel	WaitForDebugEvent
0047E8BF	. 85C0	TEST EAX,EAX	
0047E8C1	> 0F84 2B270000	JE UnPackMe.00480FF2	
0047E8C7	. 8B85 FCFDFFFF	MOV EAX,DWORD PTR SS:[EBP-204]	
0047E8CD	. 25 FF000000	AND EAX,0FF	
0047E8D2	. 85C0	TEST EAX,EAX	
0047E8D4	> 74 13	JE SHORT UnPackMe.0047E8E9	
0047E8D6	. 8B00 449F4B00	MOV ECX,DWORD PTR DS:[4B9F44]	
0047E8DC	. 8379 20 00	CMP DWORD PTR DS:[ECX+20],0	
0047E8E0	> 74 07	JE SHORT UnPackMe.0047E8E9	
0047E8E2	. C685 FCFDFFFF	MOV BYTE PTR SS:[EBP-204],0	
0047E8E9	> 68 389E4B00	PUSH UnPackMe.004B9E38	pCriticalSection = UnPackMe
0047E8EE	. FF15 A4314B00	CALL DWORD PTR DS:[<&KERNEL32.EnterCrit	EnterCriticalSection
0047E8F4	. 60	PUSHAD	
0047E8F5	. 33C0	XOR EAX,EAX	
0047E8F7	> 75 02	JNZ SHORT UnPackMe.0047E8FB	

_Ta Areas need to create a memory to decrypt 1000 bytes and 47e8bf we will point to the area we are about to create a memory. And where in belgium nhĩ hours. You remember the image address 48180D Ko, I see an attractive 401,000. Ok, drilling created, we jump to what it was. Need patch đām code here so they make a single jump is to the memory of him. To jump into the grass as we need to clean đām code of WaitForDebugEvent function. You see the command test eax, eax a jump command. I Include a Vip Jump. We will dance to it EP area we will remember (here I get 401,000). OK, we patch as follows:

Address	Hex dump	Disassembly	Comment
0047E8A9	F0	DB F0	
0047E8AA	A0	DB A0	
0047E8AB	3361 90	XOR ESP,DWORD PTR DS:[ECX-70]	
0047E8AE	90	NOP	
0047E8AF	90	NOP	
0047E8B0	90	NOP	
0047E8B1	90	NOP	
0047E8B2	90	NOP	
0047E8B3	90	NOP	
0047E8B4	90	NOP	
0047E8B5	90	NOP	
0047E8B6	90	NOP	
0047E8B7	90	NOP	
0047E8B8	90	NOP	
0047E8B9	90	NOP	
0047E8BA	90	NOP	
0047E8BB	90	NOP	
0047E8BC	90	NOP	
0047E8BD	90	NOP	
0047E8BE	90	NOP	
0047E8BF	85C0	TEST EAX,EAX	
0047E8C1	E9 3A27F8FF	JMP UnPackMe.00401000	
0047E8C6	90	NOP	
0047E8C7	8B85 FCFDFF	MOV EAX,DWORD PTR SS:[EBP-204]	
0047E8CD	25 FF000000	AND EAX,0FF	
0047E8D2	85C0	TEST EAX,EAX	
0047E8D4	74 13	JE SHORT UnPackMe.0047E8E9	
0047E8D6	8B0D 449F4B0	MOV ECX,DWORD PTR DS:[4B9F44]	
0047E8DC	8379 20 00	CMP DWORD PTR DS:[ECX+20],0	
0047E8E0	74 07	JE SHORT UnPackMe.0047E8E9	
0047E8E2	C685 FCFDFF	MOV BYTE PTR SS:[EBP-204],0	
0047E8E9	68 389E4B00	PUSH UnPackMe.004B9E38	
0047E8EE	FF15 A4314B0	CALL DWORD PTR DS:[<&KERNEL32.EnterCrit	pCriticalSection = UnPackMe.EnterCriticalSection

_Ok, Fly to 401,000 working phone: D.

Address	Hex dump	Disassembly	Comment
00401000	0000	ADD BYTE PTR DS:[EAX],AL	
00401002	0000	ADD BYTE PTR DS:[EAX],AL	
00401004	0000	ADD BYTE PTR DS:[EAX],AL	
00401006	0000	ADD BYTE PTR DS:[EAX],AL	
00401008	0000	ADD BYTE PTR DS:[EAX],AL	
0040100A	0000	ADD BYTE PTR DS:[EAX],AL	
0040100C	0000	ADD BYTE PTR DS:[EAX],AL	
0040100E	0000	ADD BYTE PTR DS:[EAX],AL	
00401010	0000	ADD BYTE PTR DS:[EAX],AL	
00401012	0000	ADD BYTE PTR DS:[EAX],AL	
00401014	0000	ADD BYTE PTR DS:[EAX],AL	
00401016	0000	ADD BYTE PTR DS:[EAX],AL	
00401018	0000	ADD BYTE PTR DS:[EAX],AL	
0040101A	0000	ADD BYTE PTR DS:[EAX],AL	

_Uh, Which created what belgium hours, khua khua. Make sure you also fly the bags packages. Default!

_O Here are two ways to create a memory region, hix. It is using OpenMutexA and I just following months of OpenMutexA surely not be because I make lười J mutex.

```

00401000 60 PUSHAD
00401001 9C PUSHFD
00401002 68 C8FB1200 PUSH 12FBC8; ASCII "5D4: AA2FD56DE" ***
00401007 33C0 XOR EAX, EAX
00401009 50 PUSH EAX
0040100A 50 PUSH EAX
0040100B E8 B5A6A577 CALL kernel32.CreateMutexA
00401010 9D POPFD
00401011 61 POPAD
00401012 - E9 7A13A677 JMP kernel32.OpenMutexA
00401017 90 NOP

```

_Dia Only 401,002 is the value of your mutex.

_Cach Second is as follows:

Address	Value	Comment
0012CD90	00000001	
0012CD94	000000A0	
0012CD98	00000054	
0012CD9C	00000001	
0012CDA0	00000000	
0012CDA4	00000000	
0012CDA8	004271B0	UnPackMe.004271B0
0012CDAC	00000002	
0012CDB0	00000000	
0012CDB4	004271B0	UnPackMe.004271B0
0012CDB8	004271B0	UnPackMe.004271B0
0012CDBC	00000000	
0012CDC0	E1539C01	
0012CDC4	00000000	
0012CDC8	00419D18	

_Ban But also remember, Ok we have 3 addresses stored OEP:

0012CDA8 004271B0 UnPackMe.004271B0

0012CDB4 004271B0 UnPackMe.004271B0

0012CDB8 004271B0 UnPackMe.004271B0

_Ta Need a jump command against the address we jump to NOP in order:

0047E8C6 90 NOP

_Ta Command also need a jump Jump if Equal order to jump to the end Submit commands. Ok, I finished analyzing the patch as follows:

Address	Hex dump	Disassembly	Comment
00401000	8105 A8CD1200 00100000	ADD DWORD PTR DS:[12CDA8],1000	
0040100A	8105 B4CD1200 00100000	ADD DWORD PTR DS:[12CDB4],1000	
00401014	8105 B8CD1200 00100000	ADD DWORD PTR DS:[12CDB8],1000	
0040101E	813D A8CD1200 00304400	CMP DWORD PTR DS:[12CDA8],UnPackMe.0044B000	
00401028	74 05	JE SHORT UnPackMe.0040102F	
0040102A	E9 97D80700	JMP UnPackMe.0047E8C6	
0040102F	90	NOP	
00401030	0000	ADD BYTE PTR DS:[EAX],AL	
00401032	0000	ADD BYTE PTR DS:[EAX],AL	
00401034	0000	ADD BYTE PTR DS:[EAX],AL	
00401036	0000	ADD BYTE PTR DS:[EAX],AL	
00401038	0000	ADD BYTE PTR DS:[EAX],AL	

_Cung Have a patch is

ADD DWORD PTR DS: [12CDA8], 1000

ADD DWORD PTR DS: [12CDB4], 1000

ADD DWORD PTR DS: [12CDB8], 1000

Cmp DWORD PTR DS: [12CDA8], UnPackMe.0044B000

JNZ UnPackMe. 0047E8C6

NOP

44B000 _Gia of where to see:

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
003C0000	00001000				Priv	RW	RW	
00400000	00001000	UnPackMe		PE header	Image	R	RWE	
00401000	0004A000	UnPackMe	.text		Image	R	RWE	401000+4A000=44B000
0044B000	0000C000	UnPackMe	.rdata		Image	R	RWE	
00457000	00009000	UnPackMe	.data		Image	R	RWE	
00460000	00003000	UnPackMe	.idata		Image	R	RWE	
00463000	00040000	UnPackMe	.text1	code	Image	R	RWE	
004A3000	00010000	UnPackMe	.adata		Image	R	RWE	
004B3000	00020000	UnPackMe	.data1	data, import	Image	R	RWE	
004D3000	00060000	UnPackMe	.pdata		Image	R	RWE	
00533000	00008000	UnPackMe	.rsrc	resources	Image	R	RWE	
00540000	00006000				Map	R E	R E	
00600000	00002000				Map	R E	R E	
00610000	00103000				Map	R	R	
00700000	0000F000				Map	R	R	

_O Target here is the VC (VB +, ASM, C) we should Cmp text section, individual Delphi we compare the section code.

_Sau The patch, patch continued in 3 addresses containing OEP:

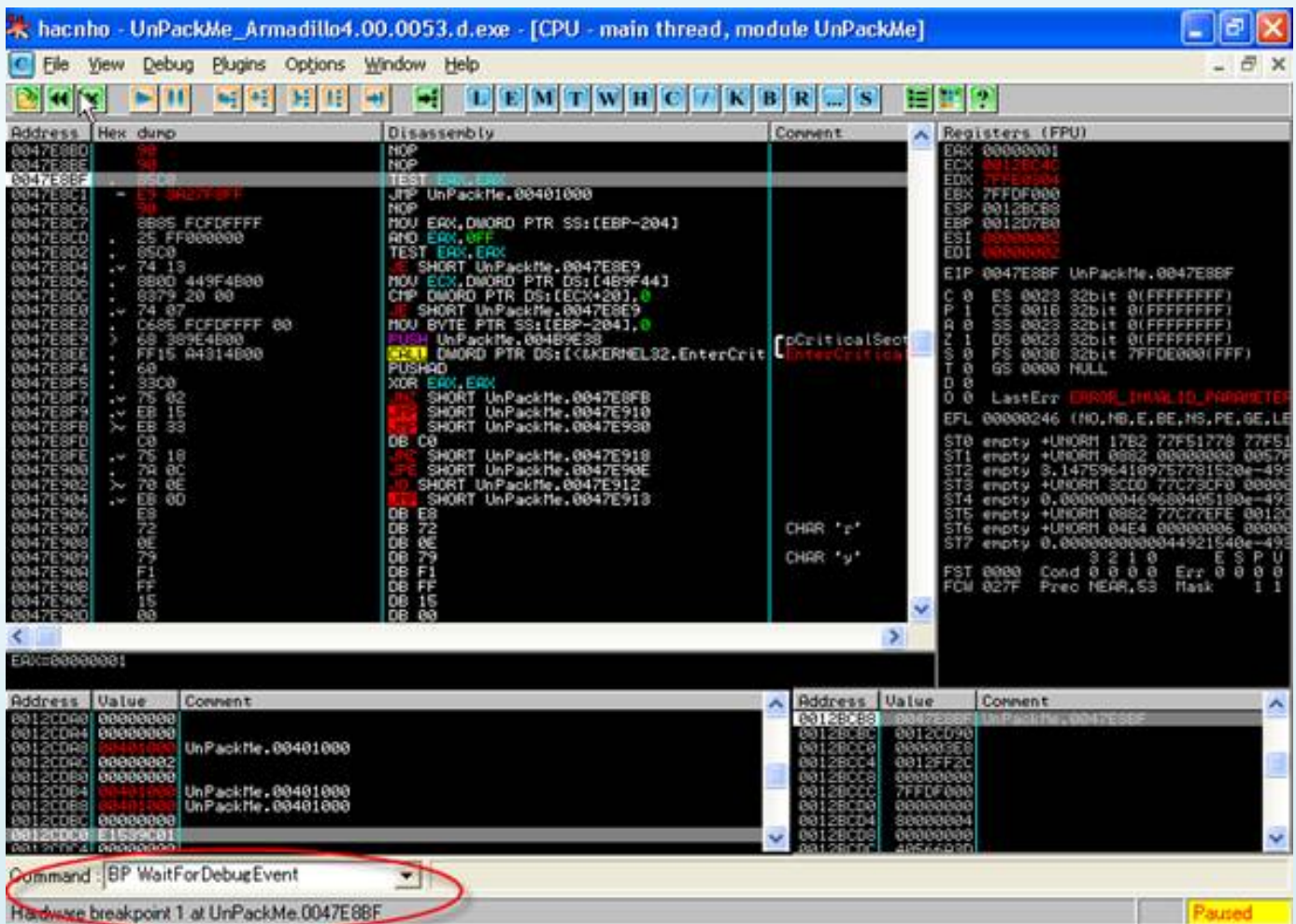
Address	Value	Comment
0012CDA4	00000000	
0012CDA8	004271B0	UnPackMe.004271B0
0012CDAC	00000002	
0012CDB0	00000000	
0012CDB4	004271B0	UnPackMe.004271B0
0012CDB8	004271B0	UnPackMe.004271B0
0012CDBC	00000000	
0012CDC0	E1539C01	
0012CDC4	00000000	
0012CDC8	09F48D18	

Address	Value	Comment
0012CDA4	00000000	
0012CDA8	00400000	UnPackMe.00400000
0012CDAC	00000002	
0012CDB0	00000000	
0012CDB4	00400000	UnPackMe.00400000
0012CDB8	00400000	UnPackMe.00400000
0012CDBC	00000000	
0012CDC0	E1539C01	
0012CDC4	00000000	
0012CDC8	09F48D18	

_Chung One patch to do so to sign MZP:

Address	Value	Comment
0006CD90	00000001	
0006CD94	00000004	
0006CD98	00000000	
0006CD9C	80000001	
0006CDA0	00000000	
0006CDA4	00000000	
0006CDA8	00400000	ASCII "MZP"
0006CDAC	00000002	
0006CDB0	00000000	
0006CDB4	00400000	ASCII "MZP"
0006CDB8	00400000	ASCII "MZP"
0006CDBC	00000000	
0006CDC0	E1538D01	
0006CDC4	00000000	
0006CDC8	AA091D18	
0006CDCC	00000000	
0006CD00	00000013	

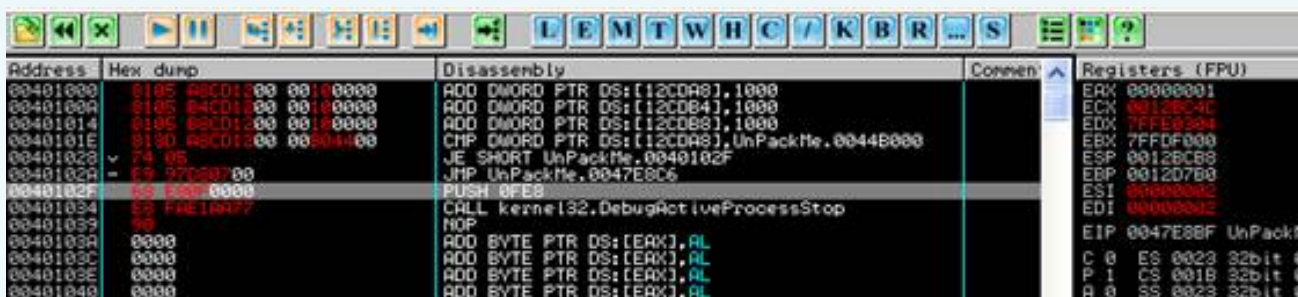
_Yeah Yeah, CopyMemII ôi, they spend calling. Once the patch is finished as you Ctrl + G to address 47E8BF a hardware breakpoint set here. Press F9 to run:



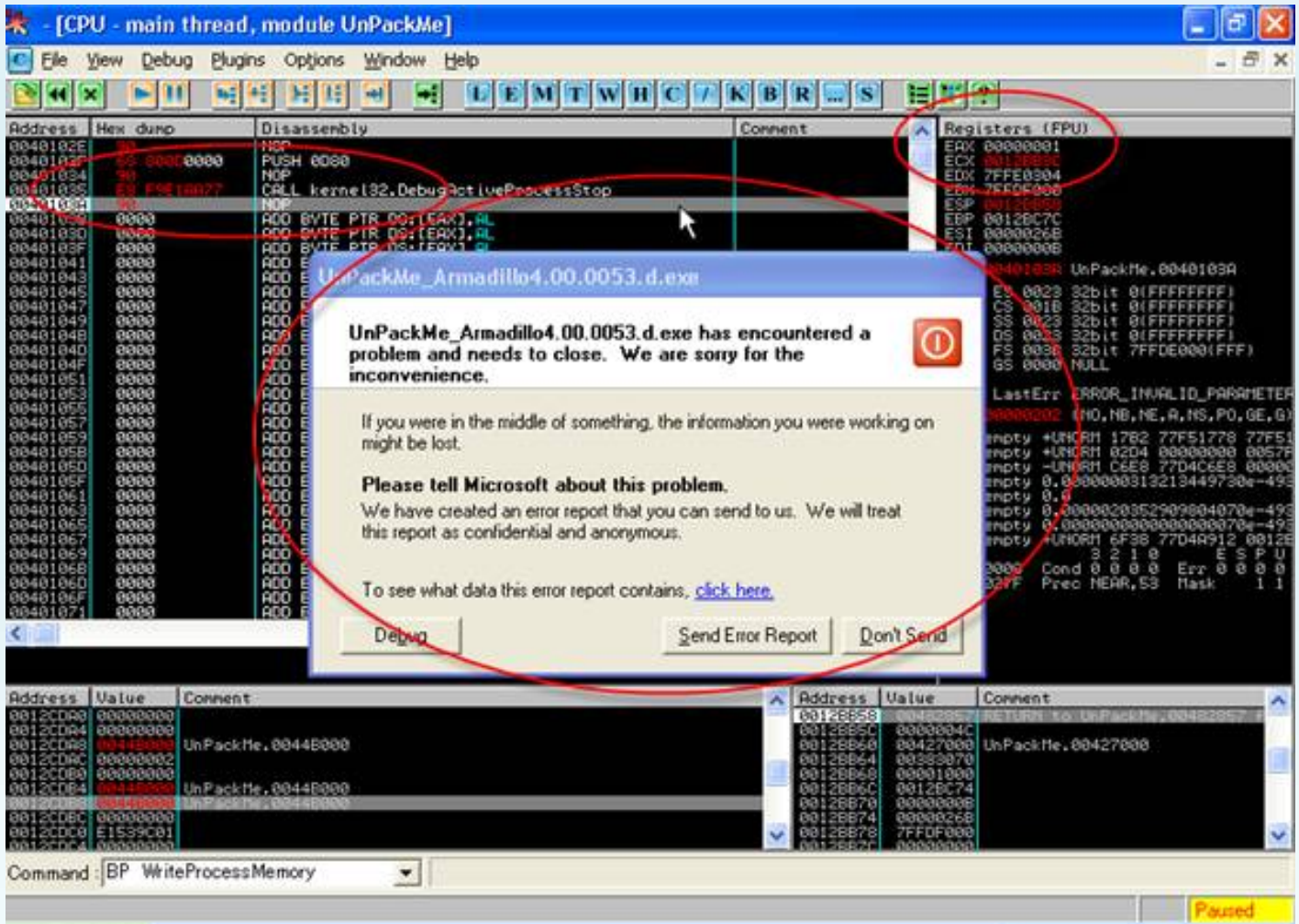
_Quay To 401,000, we pass by debug blocker:



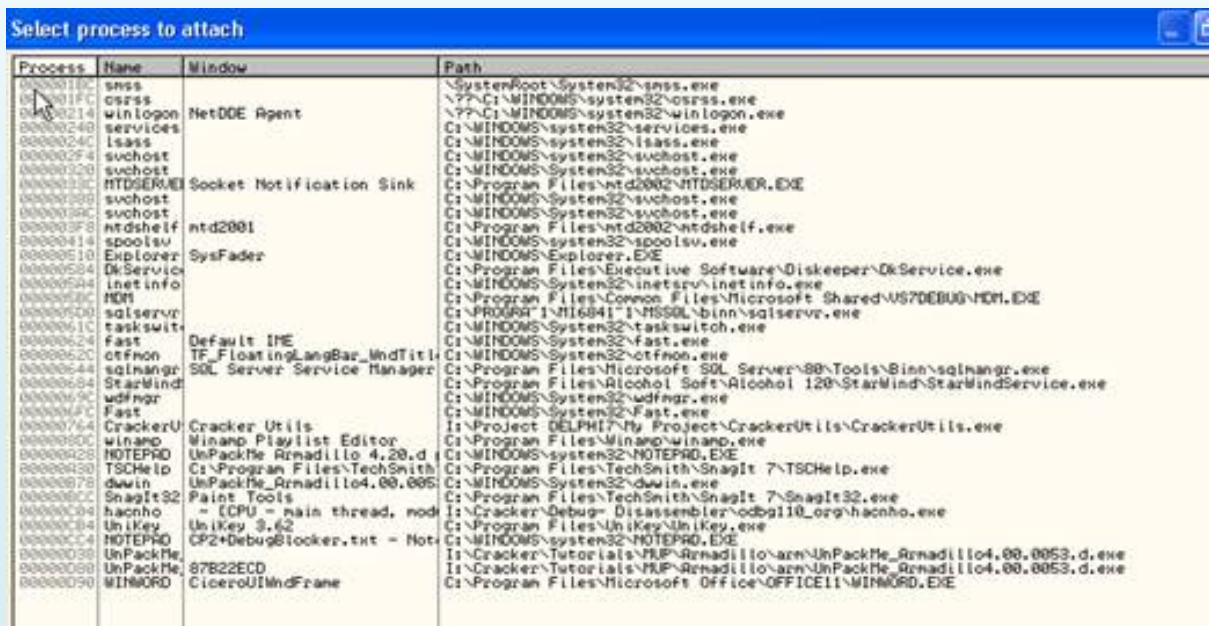
401025 _Tai address, we pass by as usual:



_Nhan F9, you will stop in HE. Press F8 when the last from the following functions nop Call DebugActiveProcessStop, I run on it not eax = 1 is the gateway to, hix, yeah, not by 0. Ka Ka, AC: A crash notification, and then I die:



_Kha Kha, choc you please stop, CopyMem official went to dust. Debug Thằng Blocker also xách package follow. The crash is informed by sharing with our hands đấy! Khua khua. Do not close, forget it. Open a OllyDBG other, Attach.



_Neu You do as I follow the steps above you will see two up process, you would share with mud.

Redo nhé! Who is to do here, the doctors always to the J.

The screenshot shows a debugger window titled "[CPU - thread 000000B4, module ntdll]". The main pane displays assembly code with columns for Address, Hex dump, Disassembly, and Comment. The code is from the ntdll.dll module, specifically around the 77F76700-77F76824 range. The disassembly shows instructions like MOV EAX, DWORD PTR SS:[ESP+4], INT3, MOV EAX, DWORD PTR FS:[10], MOV EDI, DWORD PTR SS:[ESP+C], MOV EDX, DWORD PTR SS:[ESP+8], MOV DWORD PTR DS:[EDX], 0, MOV DWORD PTR DS:[EDX+4], EDI, OR EDI, EDI, and SHOR ntdll.77F76807. The registers pane on the right shows the state of the CPU registers, including EAX, ECX, EDI, ESP, EBP, ESI, and EIP. The command line at the bottom shows "BP WaitFor UnPackMe_Armadillo4.00.0053.d.exe".

Address	Hex dump	Disassembly	Comment
77F76700	CC	INT3	
77F76701	8B4424 04	MOV EAX, DWORD PTR SS:[ESP+4]	
77F76705	CC	INT3	
77F76706	C2 0400	SHOR ntdll.77F76706	
77F76709	64:A1 10000000	MOV EAX, DWORD PTR FS:[10]	
77F7670F	C3	RETN	
77F76710	57	PUSH EDI	
77F76711	8B7C24 0C	MOV EDI, DWORD PTR SS:[ESP+C]	
77F76715	8B5424 08	MOV EDX, DWORD PTR SS:[ESP+8]	
77F76719	C702 00000000	MOV DWORD PTR DS:[EDX], 0	
77F7671F	897A 04	MOV DWORD PTR DS:[EDX+4], EDI	
77F76722	0BFF	OR EDI, EDI	
77F76724	74 11	JE SHORT ntdll.77F76807	
77F76726	83C9 FF	OR ECX, FFFFFFFF	
77F76729	33C0	XOR EAX, EAX	
77F7672B	F2:AE	REPNE SCAS BYTE PTR ES:[EDI]	
77F7672D	F701	NOT ECX	
77F7672F	66:894A 02	MOV WORD PTR DS:[EDX+2], CX	
77F76803	49	DEC ECX	
77F76804	66:890A	MOV WORD PTR DS:[EDX], CX	
77F76807	5F	POP EDI	
77F76808	C2 0800	SHOR ntdll.77F76808	
77F76809	57	PUSH EDI	
77F7680C	8B7C24 0C	MOV EDI, DWORD PTR SS:[ESP+C]	
77F76810	8B5424 08	MOV EDX, DWORD PTR SS:[ESP+8]	
77F76814	C702 00000000	MOV DWORD PTR DS:[EDX], 0	
77F7681A	897A 04	MOV DWORD PTR DS:[EDX+4], EDI	
77F7681D	0BFF	OR EDI, EDI	
77F7681F	74 11	JE SHORT ntdll.77F76832	
77F76821	83C9 FF	OR ECX, FFFFFFFF	
77F76824	33C0	XOR EAX, EAX	

Address	Value	Comment
004B3000	77C73607	60132.DeleteDC
004B3004	77C7355C	60132.RealizePalette
004B3008	77C74E8E	60132.SelectPalette
004B300C	77C77080	60132.CreateDCA
004B3010	77C75B6C	60132.CreatePalette
004B3014	77C7305F	60132.DeleteObject
004B3018	77C7306E	60132.BitBlt
004B301C	77C71B98	60132.SelectObject
004B3020	77C720E0	60132.CreateCompatibleDC
004B3024	77C72C10	60132.CreateOIBitmap

Address	Value	Comment
0149FF00	00000000	0149FF00
0149FF04	00000000	0149FF04
0149FF08	00000000	0149FF08
0149FF0C	0149FF00	0149FF0C
0149FF0E	85000980	0149FF0E
0149FFE4	FFFFFFFF	0149FFE4
0149FFE8	77F79005	0149FFE8
0149FFEC	77F741D8	0149FFEC

Command: BP WaitFor UnPackMe_Armadillo4.00.0053.d.exe

_F9, F12, Oh, we are stopping the process decrypt your father then! Congratulation! OEP was returned form its true!

- [CPU - main thread, module UnPackMe]

File View Debug Plugins Options Window Help

LEMTWHCKBR...S

Address	Hex dump	Disassembly	Comment
004271B0	55	PUSH EBP	
004271B1	8BEC	MOV EBP,ESP	
004271B3	6A FF	PUSH -1	
004271B5	68 600E4500	PUSH UnPackMe.00450E60	
004271B8	68 C0924200	PUSH UnPackMe.004292C0	
004271BF	641A1 00000000	MOV EAX,DWORD PTR FS:[0]	
004271C5	50	PUSH EAX	
004271C6	6418925 000000	MOV DWORD PTR FS:[0],ESP	
004271C0	83C4 A0	ADD ESP,-50	
004271D0	53	PUSH EBX	
004271D1	56	PUSH ESI	
004271D2	57	PUSH EDI	
004271D3	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
004271D6	FF15 DC004600	JMP DWORD PTR DS:[4600DC]	
004271DC	33D2	XOR EDX,EDX	
004271DE	8AD4	MOV DL,AH	
004271E0	8915 34E64500	MOV DWORD PTR DS:[45E694],EDX	
004271E6	8BC8	MOV ECX,EAX	
004271E8	01E1 FF000000	AND ECX,0FF	
004271EE	8900 30E64500	MOV DWORD PTR DS:[45E690],ECX	
004271F4	C1E1 00	SHL ECX,0	
004271F7	8BCA	ADD ECX,EDX	
004271F9	8900 20E64500	MOV DWORD PTR DS:[45E62C],ECX	
004271FF	C1E8 10	SHR EAX,10	
00427202	A0 30E64500	MOV DWORD PTR DS:[45E620],EAX	
00427207	E8 94210000	JMP UnPackMe.004295A0	
0042720C	85C8	TEST EAX,EAX	
0042720E	75 0A	JNZ SHORT UnPackMe.0042721A	
00427210	6A 1C	PUSH 1C	
00427212	E8 49010000	JMP UnPackMe.00427360	
00427217	83C4 04	ADD ESP,4	
0042721A	E8 D12F0000	JMP UnPackMe.0042A1F0	
0042721F	85C8	TEST EAX,EAX	

Registers (FPU)
EAX 015A0000
ECX 00001000
EDX 00000000
EBX 0012B378
ESP 0012B330
EBP 0012B30C
ESI 00000000
EDI 7FFDF000
EIP 7FFE0304
C 0 ES 0023 32bit 0(FFFFFFFF)
P 0 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL
D 0
0 0 LastErr ERROR_SUCCESS (00000000)
EPL 00000202 (NO,NB,NE,A,NS,PO,GE,G)
ST0 empty +UNORM 25C0 000000F5 AA742
ST1 empty 4.4772184561011752590e-493
ST2 empty +UNORM 0001 00120044 0012C
ST3 empty 3.371133554112553100e-493
ST4 empty 5.9205387542319596580e-493
ST5 empty -1.468596124212672770e-42
ST6 empty 1.000000000000000000000000
ST7 empty 1.000000000000000000000000
3 2 1 0 ESP U
FST 0000 Cond 0 0 0 0 Err 0 0 0 0
FCW 027F Prec NEAR,S3 Mask 1 1

Address	Value	Comment
00463000	77C73607	GDI32.DeleteDC
00463004	77C7855C	GDI32.RealizePalette
00463008	77C7485E	GDI32.SelectPalette
0046300C	77C77000	GDI32.CreateDCA
00463010	77C7586C	GDI32.CreatePalette
00463014	77C7306F	GDI32.DeleteObject
00463018	77C78C6E	GDI32.BitBlt
0046301C	77C71B78	GDI32.SelectObject
00463020	77C720E0	GDI32.CreateCompatibleDC
00463024	77C75C16	GDI32.CreateDIBitmap

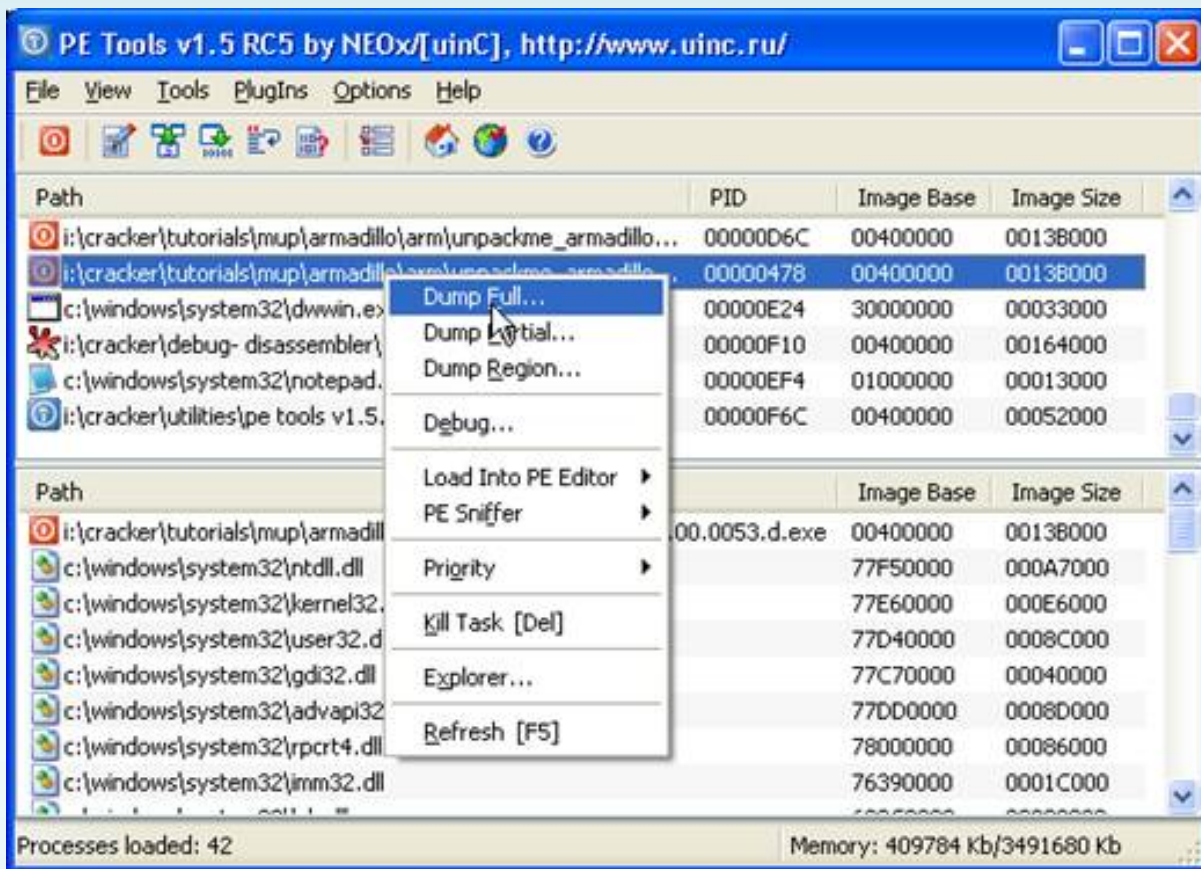
Address	Value	Comment
0012B330	77E75E00	RETURN to stdll.77E76700
0012B334	77E75E00	RETURN to kernel32.77E75EE0
0012B338	00000002	
0012B33C	0012B378	
0012B340	00000001	
0012B344	00000000	
0012B348	0012B364	
0012B34C	00000001	
0012B350	01500000	
0012B354	77E51502	stdll.Br[Get]astWin32Error

Command: BP: WriteProcessMemory

Paused

start [Taskbar icons] 1:39 AM

_Tai Here we dum full.

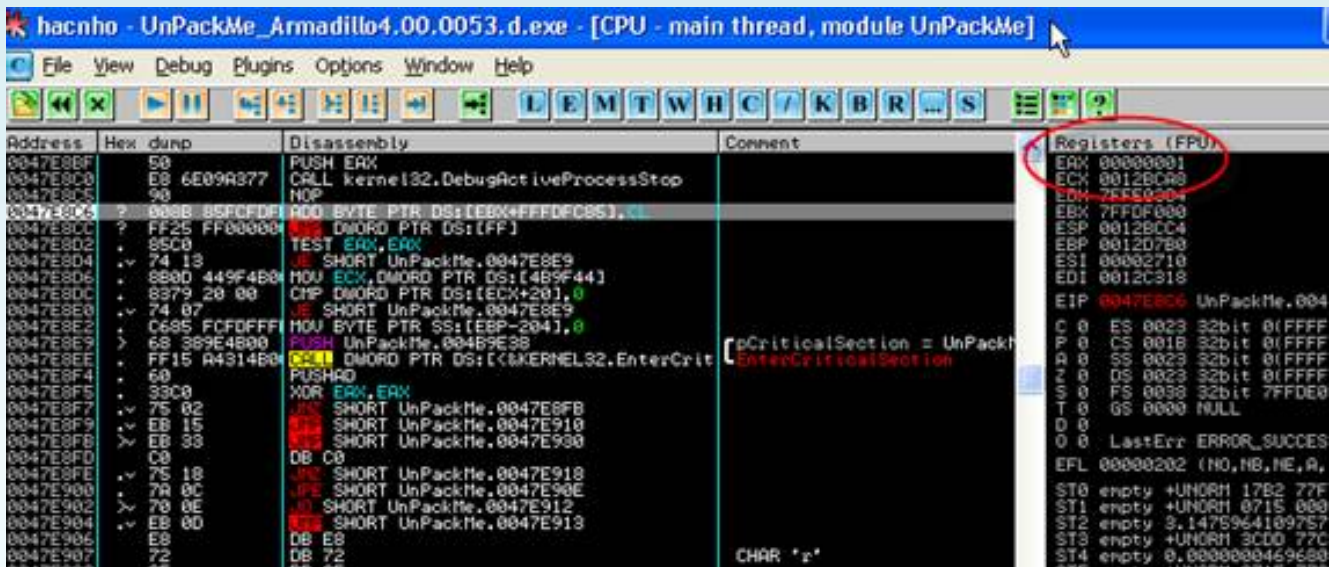


Important not least as we fix IAT.

_Sau When full dump, hix hix, I unpack itself this right unpackme 5 minutes. Write tut 2h the doctors take you. With the children and then finish writing. Overwrite them are too painful, so too lười hours, travel to take the café. 2.30h gòi morning!



Oilly _Dong all, boot machine sure to eat J. Load the target. By pass process as usual (I have used months post or simply use the script).



_Attach, Select the correct PID, F9, F12, change into bytes EBFE 558B.

Address	Hex dump	Disassembly	Comment
0048E743	55	PUSH EBP	
0048E744	8BEC	MOV EBP,ESP	
0048E746	6A FF	PUSH -1	
0048E748	68 208B4B00	PUSH UnPackMe.004B8B20	
0048E74D	68 80E44800	PUSH UnPackMe.0048E480	
0048E752	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
0048E758	50	PUSH EAX	
0048E759	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
0048E760	83EC 58	SUB ESP,58	
0048E763	53	PUSH EBX	
0048E764	56	PUSH ESI	

_Alt + F1: HE GetModuleHandleA. F9 first:

Address	Value	Comment
0012FF38	0048E80B	CALL to GetModuleHandleA from UnPackMe.0048E805
0012FF3C	00000000	pModule = NULL
0012FF40	00000000	
0012FF44	00142462	
0012FF48	0000000A	
0012FF4C	0012B3AC	
0012FF50	00130608	
0012FF54	7FFDF000	
0012FF58	AA28DC8C	
0012FF5C	00142462	

_F9 Times 2:

Address	Value	Comment
0012C804	77C959FC	CALL to GetModuleHandleA from 77C959F6
0012C808	77C131AC	pModule = "kernel32.dll"
0012C80C	77C5CA20	
0012C810	00000000	
0012C814	77C1E94F	
0012C818	77F59A7B	RETURN to ntdll.77F59A7B from ntdll.77F78C4E
0012C81C	0000E323	
0012C820	0012C80C	
0012C824	77E7D173	kernel32.GetEnvironmentVariableA
0012C828	0012C800	Pointer to next SEH record

_F9 Times 3:

Address	Value	Comment
0012C68C	74721BF0	CALL to GetModuleHandleA from 74721BF6
0012C690	0012C694	pModule = "C:\WINDOWS\System32\ntdll.dll"
0012C694	575C9A43	
0012C698	4F444E49	
0012C69C	535C5357	
0012C6A0	65747379	
0012C6A4	5C32336D	
0012C6A8	6C64746E	
0012C6AC	6C642E6C	
0012C6B0	0014006C	

_F9 Times 4:

Address	Value	Comment
0012C698	74721BF0	CALL to GetModuleHandleA from 74721BF6
0012C69C	0012C6A0	pModule = "C:\WINDOWS\System32\imm32.dll"
0012C6A0	575C9A43	
0012C6A4	4F444E49	
0012C6A8	535C5357	
0012C6AC	65747379	
0012C6B0	5C32336D	
0012C6B4	336D6D69	
0012C6B8	6C642E32	
0012C6BC	0014006C	

_F9 Times 5:

Address	Value	Comment
0012C5EC	74721BF0	CALL to GetModuleHandleA from 74721BF6
0012C5F0	0012C5F4	pModule = "C:\WINDOWS\System32\KERNEL32"
0012C5F4	575C3A43	
0012C5F8	4F444E49	
0012C5FC	535C5357	
0012C600	65747379	
0012C604	5C32336D	
0012C608	4E52454B	
0012C60C	32334C45	
0012C610	00000000	

_F9 Times 6:

Address	Value	Comment
0012C728	00AAA990	CALL to GetModuleHandleA from msctfime.00AAA97A
0012C72C	0012C730	pModule = "C:\WINDOWS\System32\ntdll.dll"
0012C730	575C3A43	
0012C734	4F444E49	
0012C738	535C5357	
0012C73C	65747379	
0012C740	5C32336D	
0012C744	6C64746E	
0012C748	6C642E6C	
0012C74C	77F5006C	ASCII "DOS mode mode"

_F9 Times 7:

Address	Value	Comment
0012CF9C	70A78663	CALL to GetModuleHandleA from SHLWAPI.70A7865D
0012CFA0	70A7F8FC	pModule = "KERNEL32.DLL"
0012CFA4	00000000	
0012CFA8	70A70000	SHLWAPI.70A70000
0012CFAC	0012CFE8	
0012CFB0	00000001	
0012CFB4	77F5166A	RETURN to ntdll.77F5166A from ntdll.77F78C4E
0012CFB8	77F5166A	RETURN to ntdll.77F5166A from ntdll.77F78C4E
0012CFBC	70A783FB	RETURN to SHLWAPI.70A783FB from SHLWAPI.70A78643
0012CFC0	00000001	

_F9 Times 8:

Address	Value	Comment
0012CEC4	7712B124	CALL to GetModuleHandleA from 7712B11E
0012CEC8	771A22E4	pModule = "KERNEL32.DLL"
0012CECC	7712AD56	RETURN to 7712AD56 from 7712B119
0012CED0	771A2080	
0012CED4	000003E8	
0012CED8	7712B0CA	RETURN to 7712B0CA from 7712AD34
0012CEDC	00000001	
0012CEE0	00000001	
0012CEE4	77120080	
0012CEE8	00000000	

_F9 Times 9:

Address	Value	Comment
0012CEC0	7712B124	CALL to GetModuleHandleA from 7712B11E
0012CEC4	771A22E4	pModule = "KERNEL32.DLL"
0012CEC8	7712ADAC	RETURN to 7712ADAC from 7712B119
0012CECC	771A2080	
0012CED0	000003E8	
0012CED4	7712B0D0	RETURN to 7712B0D0 from 7712AD8A
0012CED8	00000001	
0012CEDC	00000001	
0012CEE0	00000001	
0012CEE4	77120080	

_F9 Times 10:

Address	Value	Comment
00120750	00479463	CALL to GetModuleHandleA from UnPackMe.00479450
00120754	00000000	pModule = NULL
00120758	00120764	
0012075C	00C47EBE	
00120760	004F69D2	UnPackMe.004F69D2
00120764	00B70476	
00120768	00403000	UnPackMe.00403000
0012076C	00120700	
00120770	00C6C3A8	
00120774	00C61C57	

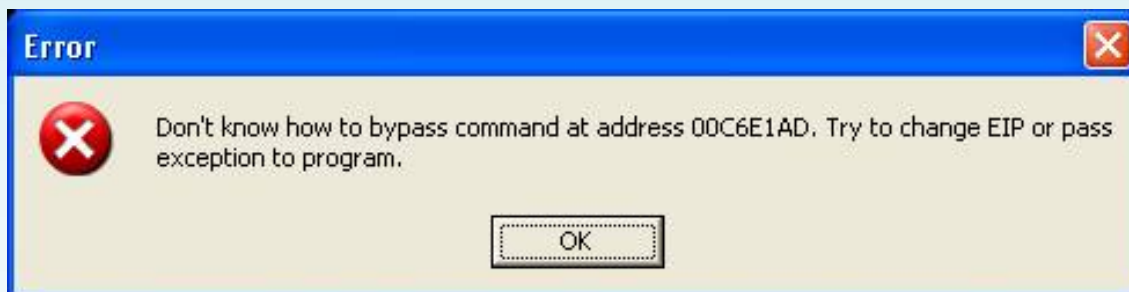
_F9 Times 11:

Address	Value	Comment
00126884	00C6372E	CALL to GetModuleHandleA from 00C63728
00126888	00C77474	pModule = "kernel32.dll"
0012688C	00C78744	ASCII "VirtualAlloc"
00126890	00000001	
00126894	00CC0090	
00126898	00000000	
0012689C	00000000	
001268A0	00000000	
001268A4	00000000	
001268A8	00000000	

_F9 Times 12:

Address	Value	Comment
00126884	00C63748	CALL to GetModuleHandleA from 00C63745
00126888	00C77474	pModule = "kernel32.dll"
0012688C	00C78738	ASCII "VirtualFree"
00126890	00000001	
00126894	00CC0090	
00126898	00000000	
0012689C	00000000	
001268A0	00000000	
001268A4	00000000	
001268A8	00000000	

_F9 Last



_Nhan Shift + F9 you will pass by here:

Address	Hex dump	Disassembly	Comment	Registers (FFU)
77E7AD85	897C24 04 00	CMP DWORD PTR SS:[ESP+4],		EAX 00126738 ASCII "kernel32.dll"
77E7AD88	0F84 37010000	JL kernel32.77E7AEC8		ECX 00126744
77E7AD91	FF7424 04	PUSH DWORD PTR SS:[ESP+4]		EDX 00126738 ASCII "kernel32.dll"
77E7AD95	E8 F8050000	JMP kernel32.77E7B392		EBX 00000000
77E7AD9A	85C0	TEST EAX,EAX		ESP 001267F4
77E7AD9C	74 00	JSHORT kernel32.77E7ADA6		EBP 00126804
77E7AD9E	FF70 04	PUSH DWORD PTR DS:[EAX+4]		ESI 00000000
77E7ADA1	E8 27060000	JMP kernel32.GetModuleHandleW		EDI 00C7697C
77E7ADA6	C2 0400	RETN 4		EIP 77E7AD85 kernel32.GetModuleHandleW
77E7ADA9	55	PUSH EBP		C 0 ES 0023 32bit 0(FFFFFFFF)
77E7ADAA	8BEC	MOV EBP,ESP		P 1 CS 0018 32bit 0(FFFFFFFF)
77E7ADAC	83EC 10	SUB ESP,10		A 0 SS 0023 32bit 0(FFFFFFFF)
77E7ADAF	57	PUSH EDI		Z 1 DS 0023 32bit 0(FFFFFFFF)
77E7ADB0	64:A1 10000000	MOV EAX,DWORD PTR FS:[10]		S 0 FS 0038 32bit 7FDE0000(FFF)
77E7ADB6	8B7D 10	MOV EDI,DWORD PTR SS:[EBP+10]		T 0 GS 0000 NULL
77E7ADB9	8B40 30	MOV EAX,DWORD PTR DS:[EAX+30]		O 0 LastErr ERROR_ALREADY_EXISTS (0)
77E7ADBC	80C3F	LEA ECX,DWORD PTR DS:[EDI+EDI]		EFL 00000246 (NO, NB, E, BE, NS, PE, GE, LE
77E7ADBF	51	PUSH ECX		ST0 empty 4.4731075735452830400e-495
77E7ADC0	FF35 EC27ED27	PUSH DWORD PTR DS:[77ED77EC]		ST1 empty 6.954996135018450410e-285
77E7ADC6	FF70 10	PUSH DWORD PTR DS:[EAX+10]		ST2 empty -UNORM C9FA 00000000 00000
77E7ADC9	FF15 0C10E627	JL DWORD PTR DS:[kernel32.RtlAllocateHeap.RtlAllocateHeap		ST3 empty -3.0093840722362035910e+75
77E7ADCF	85C0	TEST EAX,EAX		ST4 empty +UNORM 2960 00000038 0C407
77E7ADD1	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX		ST5 empty -UNORM B92C 7FFFFFFF 00000
77E7ADD4	0F84 5A300200	JL kernel32.77E90E34		ST6 empty 1.000000000000000000000000
77E7ADD9	56	PUSH ESI		ST7 empty 1.000000000000000000000000
77E7ADD8	57	PUSH EDI		FST 0000 Cond 0 0 0 0 Err 0 0 0 0
77E7ADD0	50	PUSH EAX		FCW 037F Prec NEAR,64 Mask 1 1
77E7ADD0	FF75 00	PUSH DWORD PTR SS:[EBP+0]		
77E7ADE0	E8 C4FEFFFF	JMP kernel32.GetModuleFileNameW		
77E7ADE5	8BF0	MOV ESI,EAX		
77E7ADE7	00436	LEA EAX,DWORD PTR DS:[ESI+ESI]		
77E7ADEA	66:8945 F8	MOV WORD PTR SS:[EBP-0],AX		
77E7ADEE	66:05 0200	ADD AX,2		

Stack SS:[001265F8]=00126738, (ASCII "kernel32.dll")

Address	Value	Comment	Address	Value	Comment
00C4AC00	77C73607	00132.DeleteDC	001265F4	00C4AC01	CALL to 00132.DeleteDC From 00C4AC00
00C4AC04	77C7360C	00132.RealizePalette	001265F8	00126738	Module = "kernel32.dll"
00C4AC08	77C7360E	00132.SelectPalette	001265FC	00000000	
00C4AC0C	77C73610	00132.CreateDC	00126600	00000000	
00C4AC10	77C73612	00132.CreatePalette	00126604	5FD00012	
00C4AC14	77C73614	00132.DeleteObject	00126608	00C7697C	
00C4AC18	77C73616	00132.BitBlt	0012660C	00000000	
00C4AC1C	77C73618	00132.SelectObject	00126610	00000000	
00C4AC20	77C7361A	00132.CreateCompatibleDC	00126614	00003FFF	
00C4AC24	77C7361C	00132.CreateDIBitmap	00126618	00000000	

_Nhan F8 to trace 77E7ADA6 C2 0400 RETN 4: you will go to this code:

Address	Hex dump	Disassembly	Comment
00C4AC01	8B0D E4C9C700	MOV ECX,DWORD PTR DS:[C7C9E4]	
00C4AC07	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00C4AC0A	A1 E4C9C700	MOV EAX,DWORD PTR DS:[C7C9E4]	
00C4AC0F	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00C4ACD2	75 16	JNZ SHORT 00C4ACFA	
00C4ACD4	8085 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
00C4ACDA	50	PUSH EAX	
00C4ACDB	FF15 E020C700	JL DWORD PTR DS:[C720E0]	kernel32.LoadLibraryA
00C4ACE1	8B0D E4C9C700	MOV ECX,DWORD PTR DS:[C7C9E4]	
00C4ACE7	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00C4ACED	A1 E4C9C700	MOV EAX,DWORD PTR DS:[C7C9E4]	
00C4ACF2	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00C4ACF2	0F84 32010000	JE 00C4AE2A	Magic Jump
00C4ACF3	33C9	XOR ECX,ECX	
00C4ACFA	8B07	MOV EAX,DWORD PTR DS:[EDI]	
00C4ACFC	3918	CMP DWORD PTR DS:[EAX],EBX	
00C4ACFE	74 06	JL SHORT 00C4AD06	
00C4AD00	41	INC ECX	
00C4AD01	83C0 0C	ADD EAX,0C	
00C4AD04	EB F6	JSHORT 00C4ACFC	
00C4AD06	8B09	MOV EBX,ECX	
00C4AD08	C1E3 02	SHL EBX,2	
00C4AD0B	53	PUSH EBX	
00C4AD0C	E8 31670200	JMP to nsvert.??2@YAPAXI02	
00C4AD11	8B0D DCC9C700	MOV ECX,DWORD PTR DS:[C7C9DC]	
00C4AD17	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00C4AD1A	53	PUSH EBX	
00C4AD1B	E8 22670200	JMP to nsvert.??2@YAPAXI02	
00C4AD20	59	POP ECX	
00C4AD21	59	POP ECX	
00C4AD22	8B0D E0C9C700	MOV ECX,DWORD PTR DS:[C7C9E0]	
00C4AD28	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00C4AD2B	8B1F	MOV EBX,DWORD PTR DS:[EDI]	

_Nhu So we have a magic jump 00C4ACF2 / 0F84 32010000 JE 00C4AE2A
Patch it 00C4AE2A Jump:

Address	Hex dump	Disassembly	Comment
00C4ACC1	880D E4C9C700	MOV ECX,DWORD PTR DS:[C7C9E4]	
00C4ACC7	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00C4ACCA	A1 E4C9C700	MOV EAX,DWORD PTR DS:[C7C9E4]	
00C4ACCF	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00C4ACD2	75 16	JNZ SHORT 00C4ACEA	
00C4ACD4	8085 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
00C4ACDA	50	PUSH EAX	
00C4ACD8	FF15 E020C700	CALL DWORD PTR DS:[C720E0]	kernel32.LoadLibraryA
00C4ACE1	880D E4C9C700	MOV ECX,DWORD PTR DS:[C7C9E4]	
00C4ACE7	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00C4ACED	A1 E4C9C700	MOV EAX,DWORD PTR DS:[C7C9E4]	
00C4ACEF	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00C4ACF2	E9 23010000	JMP 00C4AE2A	
00C4ACF7	90	NOP	
00C4ACF8	33C9	XOR ECX,ECX	
00C4ACFA	8807	MOV EAX,DWORD PTR DS:[EDI]	
00C4ACFC	3918	CMP DWORD PTR DS:[EAX],EBX	
00C4ACFE	74 06	JE SHORT 00C4AD06	
00C4AD00	41	INC ECX	
00C4AD01	83C0 0C	ADD EAX,0C	
00C4AD04	EB F6	JE SHORT 00C4ACFC	

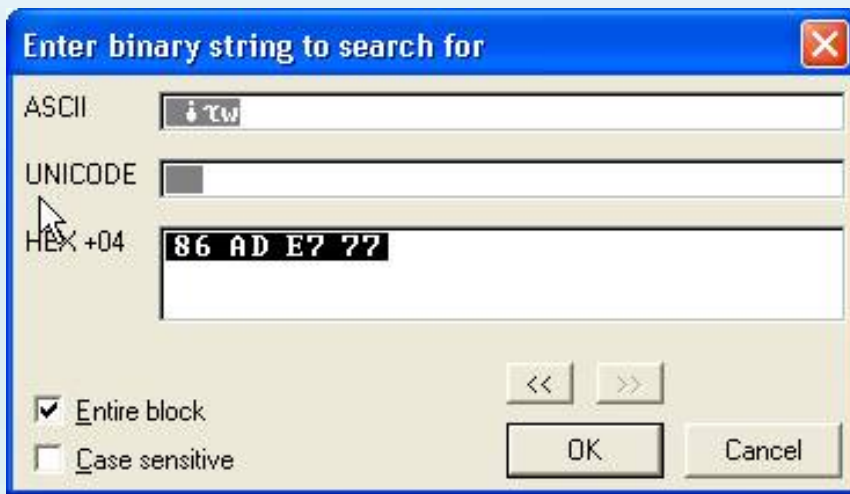
_Alt + M, memory breakpoint on access:

Address	Hex dump	Disassembly	Comment
004271B3	64:89C7	MOV EDI,EAX	Superfluous prefix
004271B3	9E	SAHF	
004271B4	CE	INTO	
004271B5	6A 4B	PUSH 4B	
004271B7	FA	CLI	
004271B8	74 02	JE SHORT UnPackMe.004271BC	
004271BA	43	INC EBX	
004271BB	3C A3	CMP AL,0A3	
004271BD	40	INC EAX	
004271BE	2B90 90022BF4	SUB EDX,DWORD PTR DS:[EAX+F42B0290]	
004271C4	3152 4F	XOR DWORD PTR DS:[EDX+4F],EDX	
004271C7	7D 14	JGE SHORT UnPackMe.004271D0	
004271C9	022B	ADD CH,BYTE PTR DS:[EBX]	
004271CB	F4	HLT	Privileged command
004271CC	3181 EF5C6254	XOR DWORD PTR DS:[ECX+54625CEF],EAX	
004271D2	7C 7D	JL SHORT UnPackMe.00427251	
004271D4	54	PUSH ESP	
004271D5	EA 04E1ED08 6D	JMP FAR F46D:08EDE1D4	Far jump
004271DC	0200	ADD DL,AL	
004271DE	A1 2088171F	MOV EAX,DWORD PTR DS:[1F17B820]	
004271E3	127402 A0	ADC DH,BYTE PTR DS:[EDX+EAX-60]	
004271E7	3C B0	CMP AL,0B0	
004271E9	E3 D4	JECXZ SHORT UnPackMe.004271BF	
004271EB	F4	HLT	Privileged command
004271EC	3102	XOR DWORD PTR DS:[EDX],EAX	
004271EE	A2 F901E46E	MOV BYTE PTR DS:[6EE401F9],AL	
004271F3	F4	HLT	Privileged command
004271F4	F0:E3 23	LOCK JECXZ SHORT UnPackMe.0042721A	LOCK prefix is not allowed
004271F7	F7FB	IDIV EBX	
004271F9	8B26	MOV ESP,DWORD PTR DS:[ESI]	
004271FB	D807	FCDM ST(7)	
004271FD	47	INC EDI	
004271FE	2B35 D91288DC	SUB ESI,DWORD PTR DS:[DC8812D9]	

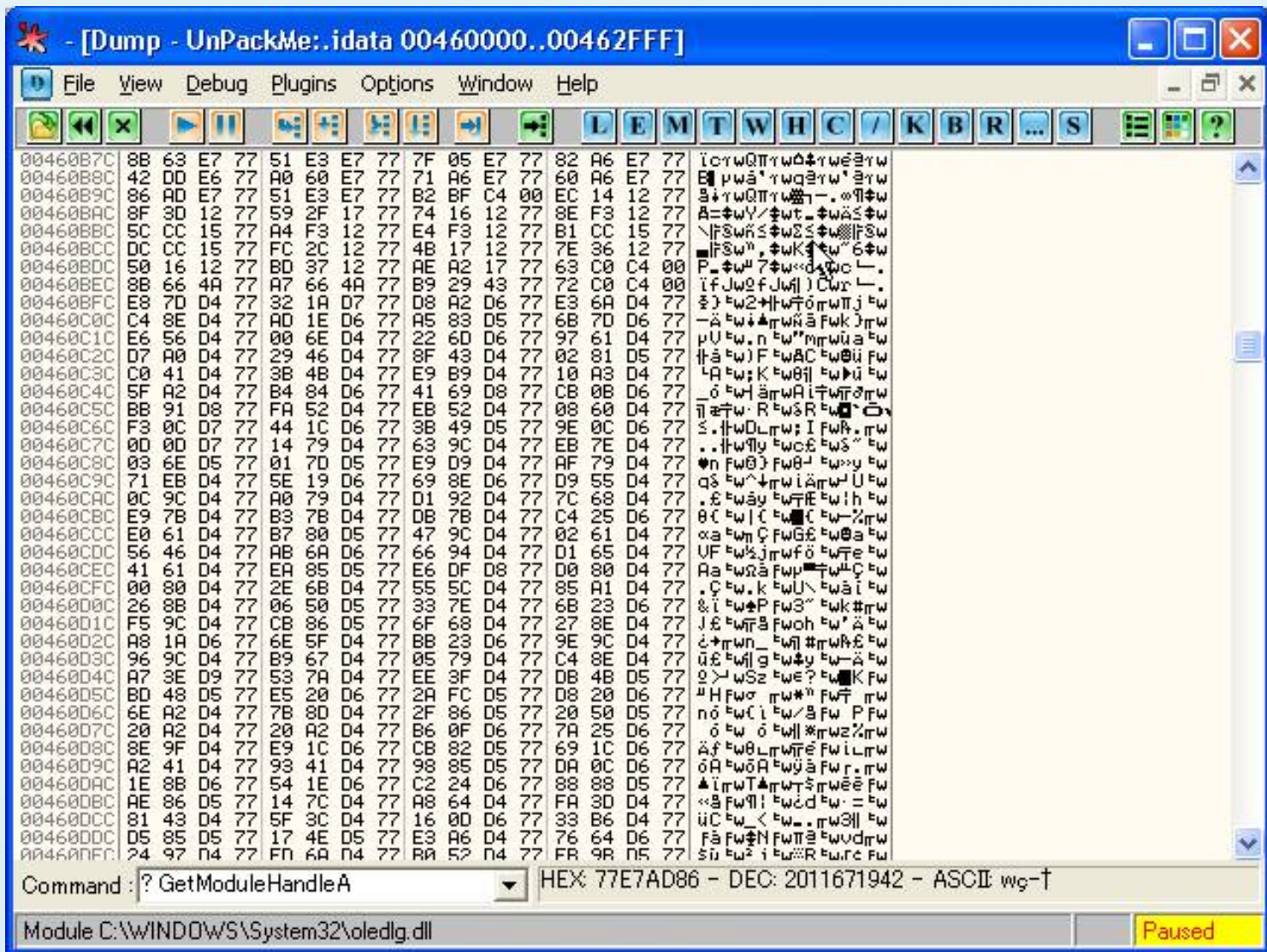
_Chung We are to OEP has been encrypted. Address to use the API function GetModuleHandleA code:

GetModuleHandleA OX77E7AD86 @ KERNEL32.DL Get ? X

_Ok And address of the function 77E7AD86, back OllyDBG, Alt + M, Ctrl + B to enter the address of the function GetModuleHandleA reverse is **86 AD E7 77**



_Ta'll Stop here:



_Vao Dump Windows, Ctrl + G to enter 460000:

Address	Value	Comment
00460000	00000000	
00460004	00000000	
00460008	00000000	
0046000C	00060F3A	
00460010	00000000	
00460014	00000000	
00460018	00000000	
0046001C	00000000	
00460020	00061904	
00460024	00000000	
00460028	00000000	
0046002C	00000000	
00460030	00000000	

_Cuon Down to having been the first function:

Address	Value	Comment
00460810	00000000	
00460814	00000000	
00460818	77DD17D8	ADVAPI32.RegCloseKey
0046081C	77DD229A	ADVAPI32.RegOpenKeyExA
00460820	77DD27D6	ADVAPI32.RegCreateKeyExA
00460824	77DE63B1	ADVAPI32.RegSetValueExA
00460828	77DD2410	ADVAPI32.RegQueryValueExA
0046082C	00C4BFFE	
00460830	773438B1	COMCTL32.InitCommonControls
00460834	7734513D	COMCTL32.ImageList_Destroy
00460838	00C4C099	
0046083C	77C73D02	GDI32.GetClipBox
00460840	77C74200	GDI32.ExcludeClipRect
00460844	77C72945	GDI32.IntersectClipRect
00460848	77C7509D	GDI32.MoveToEx
0046084C	77C70F10	GDI32.LineTo

Start _Vay IAT starting 460818

_Cuon Down to the final function:

Address	Value	Comment
00460F08	771FA25A	ole32.OleFlushClipboard
00460F0C	771DE19F	ole32.CoRevokeClassObject
00460F10	771DEE61	ole32.CoRegisterMessageFilter
00460F14	771C94CF	ole32.CoFreeUnusedLibraries
00460F18	771C48A9	ole32.CoGetClassObject
00460F1C	772090AA	ole32.StgOpenStorageOnILockBytes
00460F20	00C4BFED	
00460F24	74D3F0F3	oledlg.OleUIBusyA
00460F28	00C4BFE8	
00460F2C	00000000	
00460F30	00000000	
00460F34	00000000	
00460F38	49570000	
00460F3C	2E4D4D4E	
00460F40	006C6C64	
00460F44	00000000	

End _Vay IAT is 460F24.

IAT _Vay length = 460F24 - 460818 = 70C

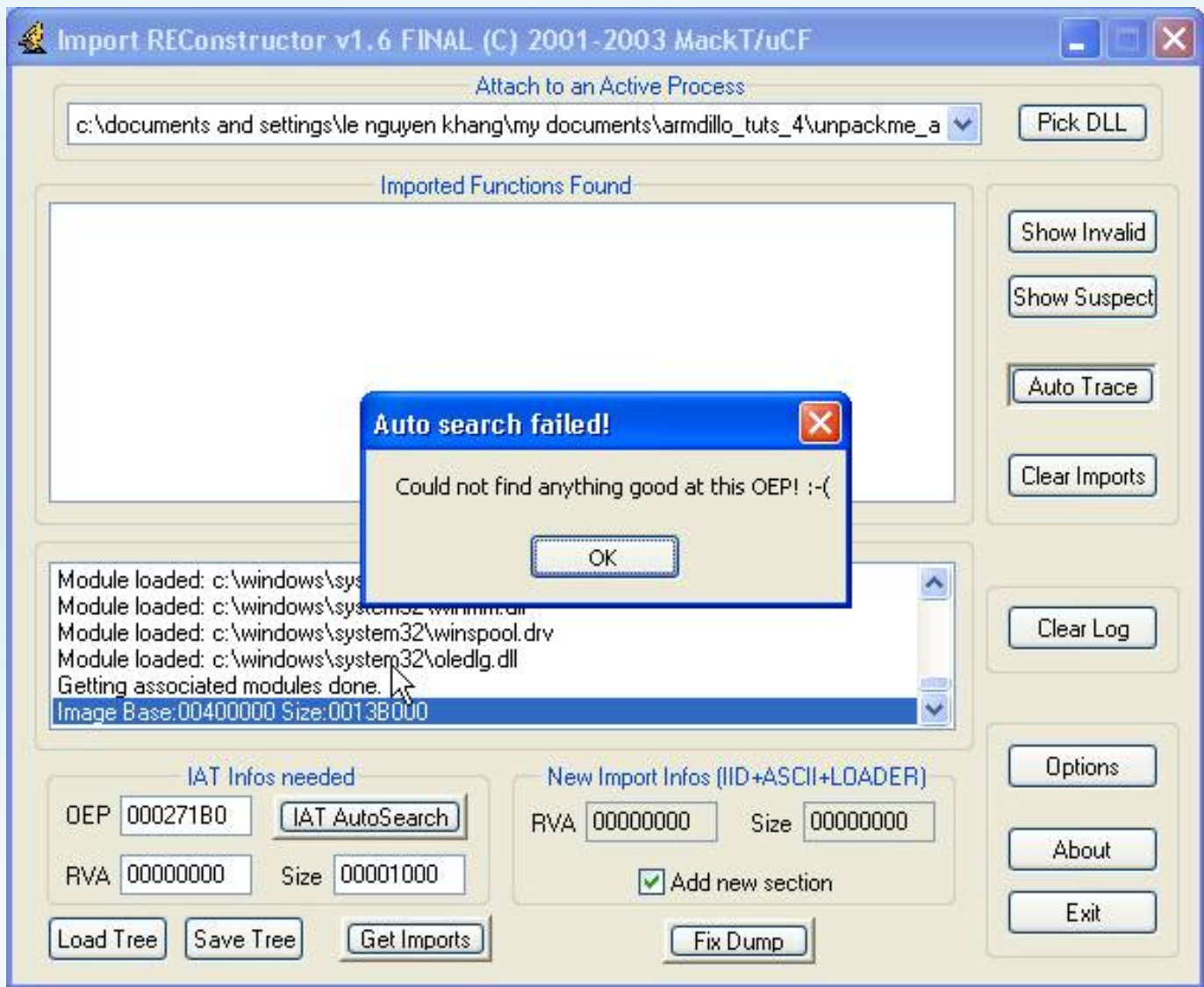
_Nhu So we have the results:

OEP: 000271B0

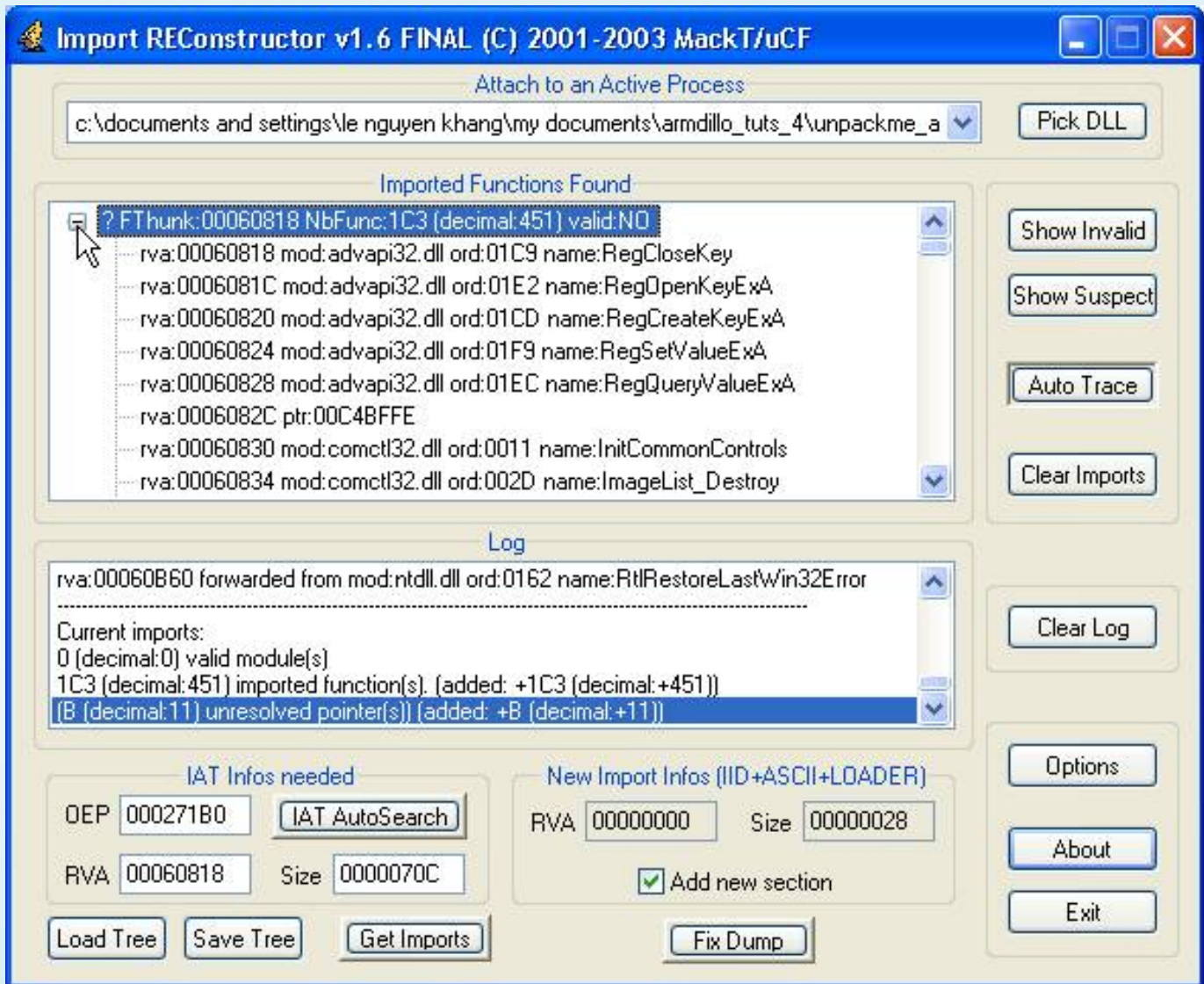
IATRVA: 00060818

IATSize: 0000070C

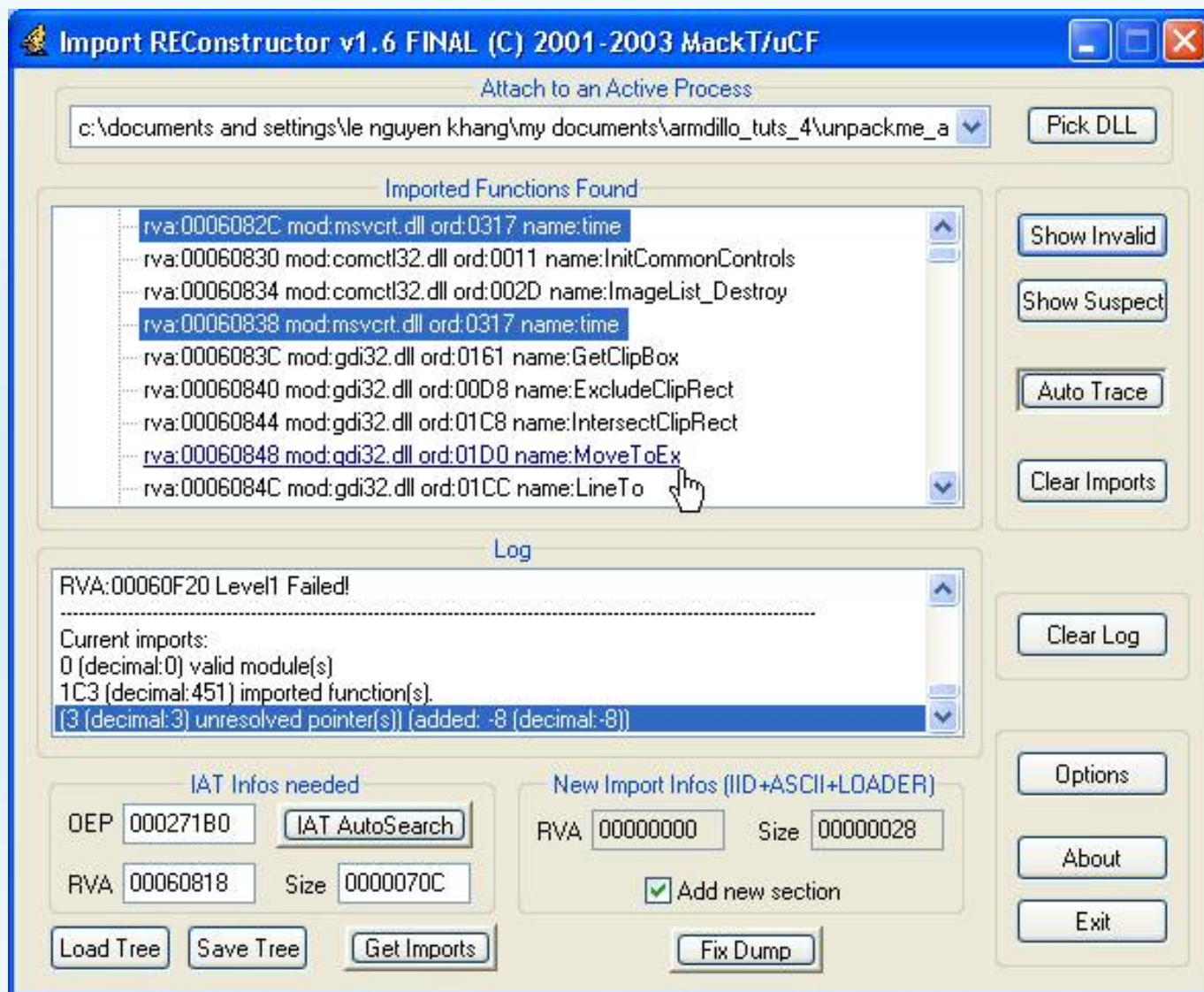
_De OllyDBG resources, open up ImportREC, select PID process coincides with the process, try to hit enter IAT IAT Auto Search see why:

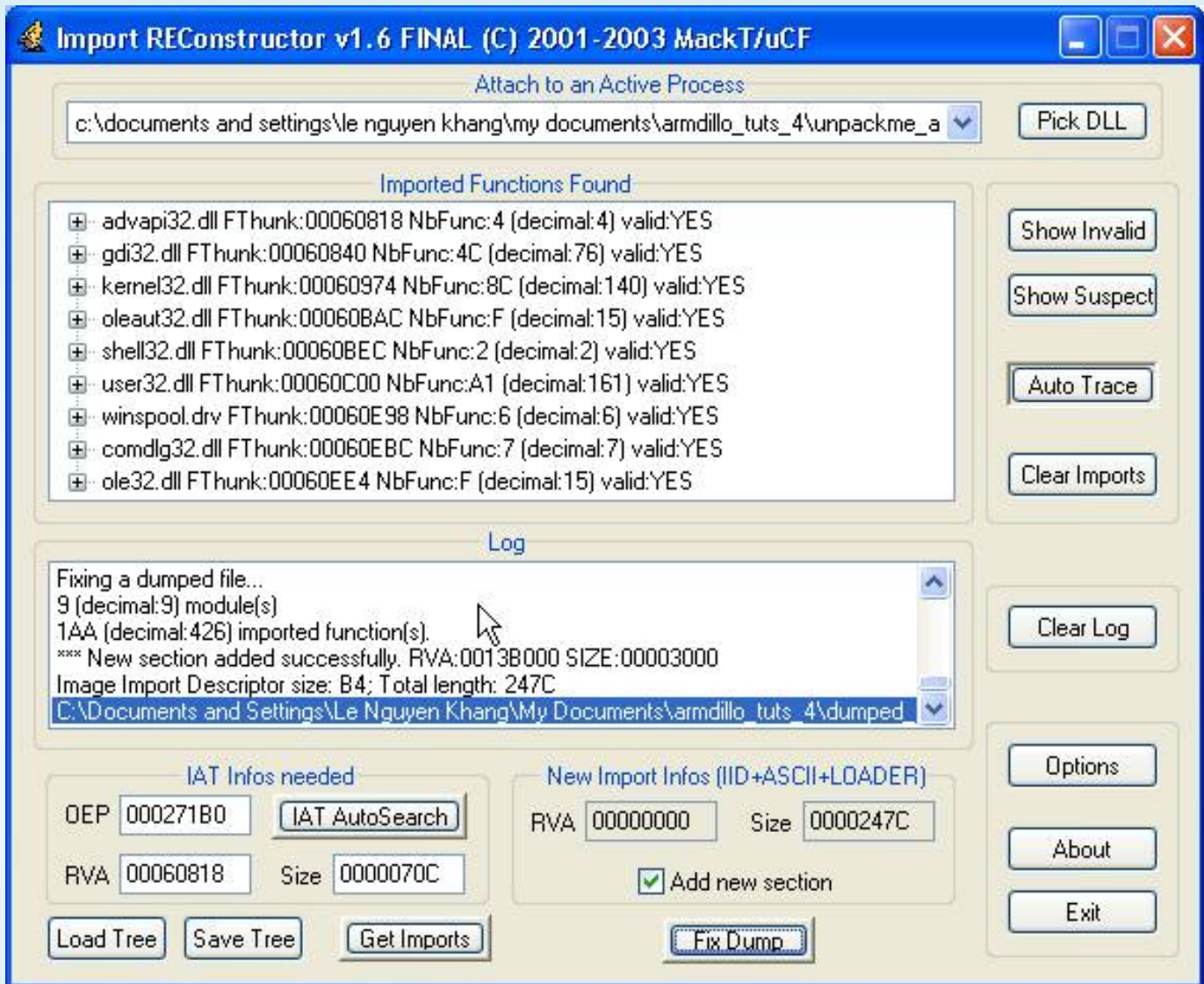


_Duoc Then, enter the information in the press to go Get Imports:

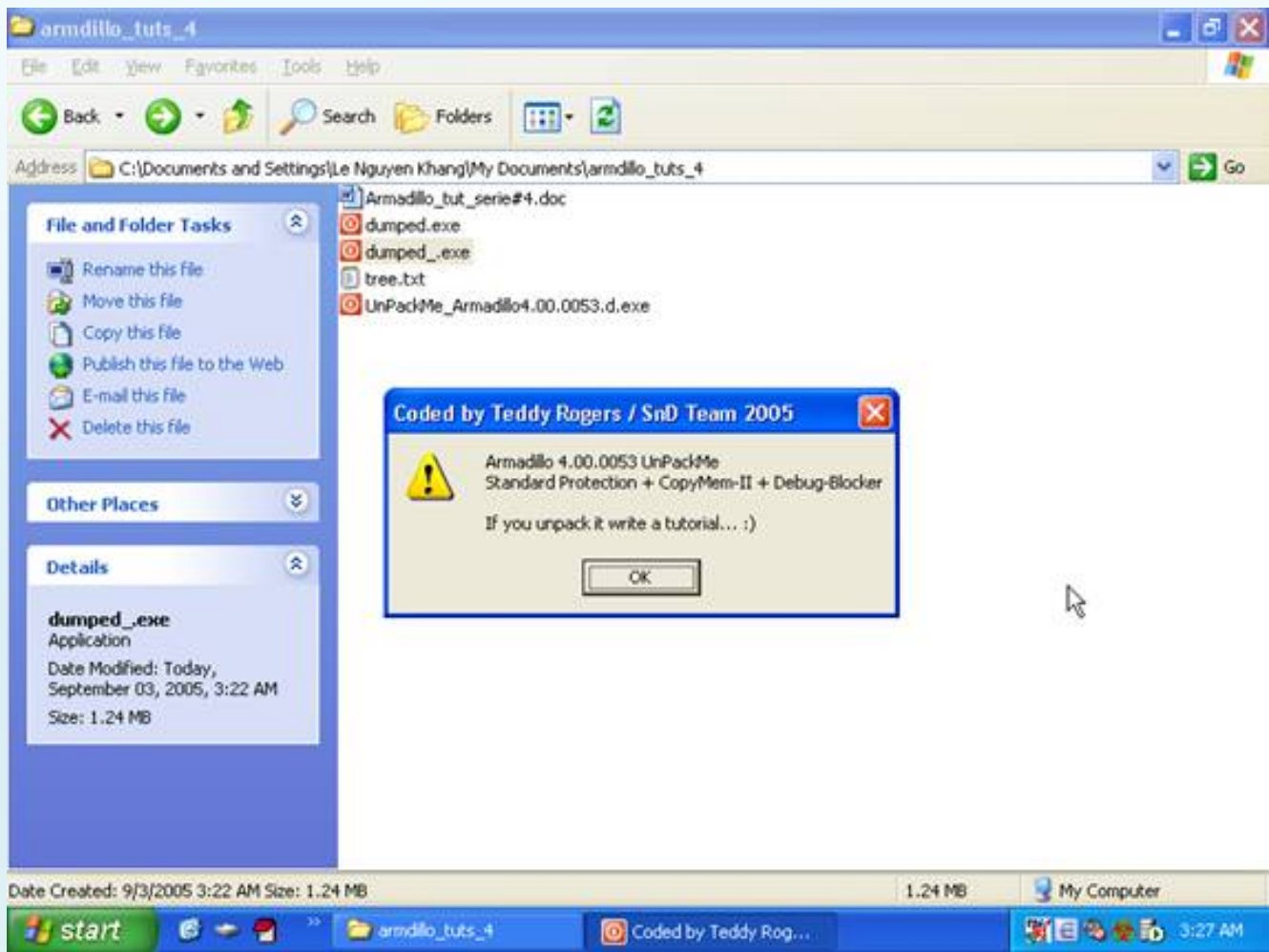


_Show Invalid, Cut thunks, fix dump:

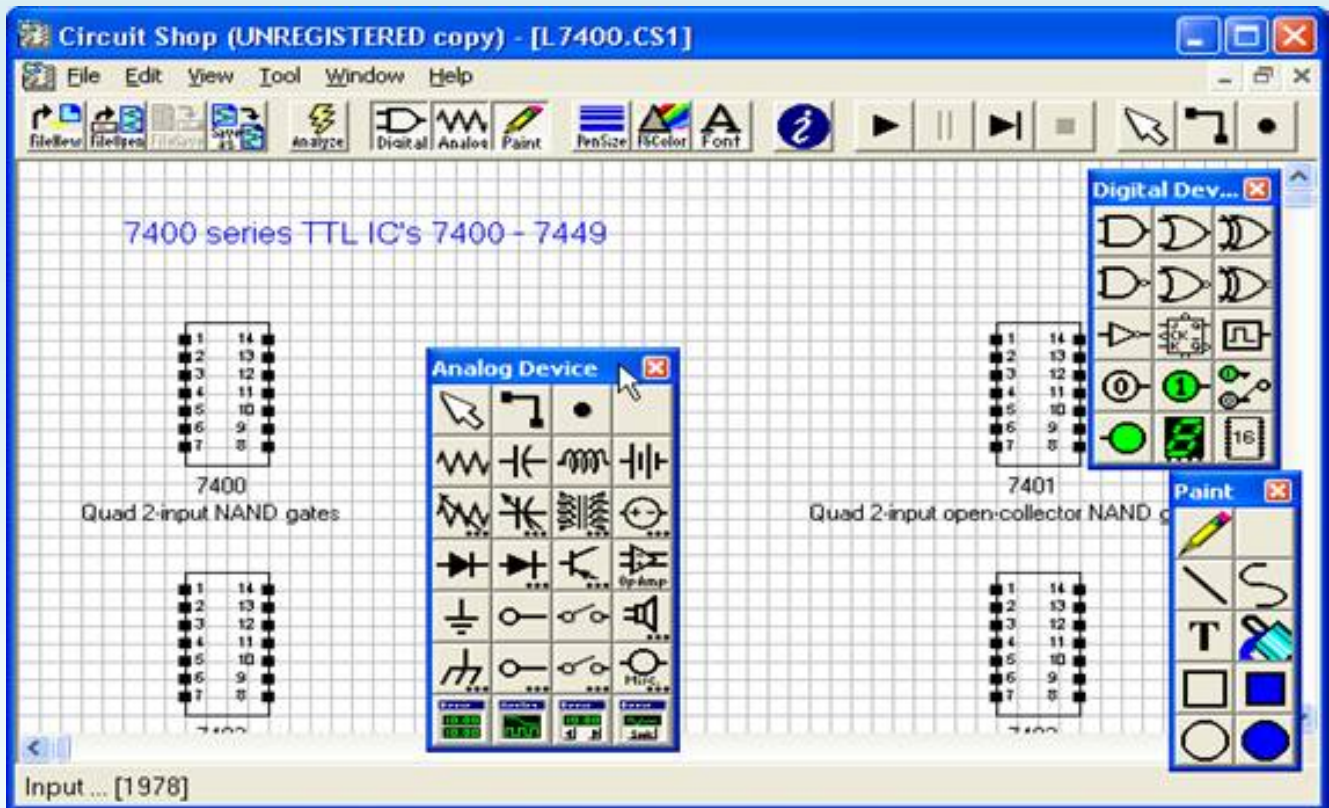




Giai The most important: Run dumped.exe



28 pages _ to protect a headache as the unpacker. But also think, you must not!
Target # 2: Circuit Shop - Armadillo 4.xx



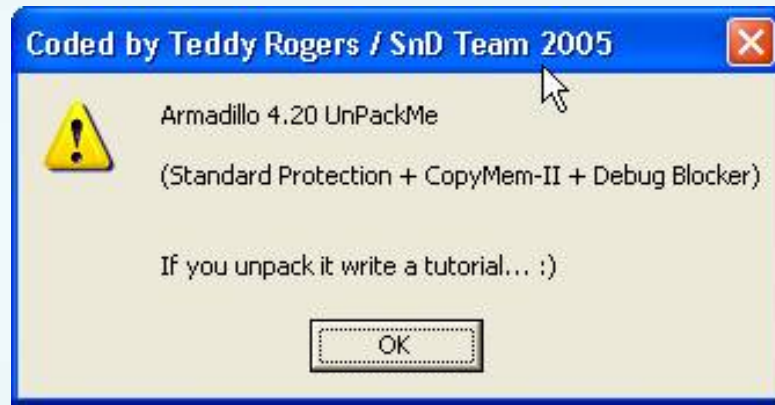
__Ban Own writing, if not involved in this topic.

Target # 3: snd-Unpackme Armadillo 4.10a



_Ban Own writing, if not involved in this topic.

Target # 4: snd-Unpackme Armadillo 4:20



_Ban Own writing, if not involved in this topic.

IV. Conclusion

_Tuan Later I go down to Western markets surveyed call (this sếp few bags of party mut season packages. Khua khua khua) so I offline a few weeks. I re ngộ you Nanomites with her children.

_Tai I do not write much tut only write one, Dear CopyMem type that has only one style and a way like this only. I test call, will post after the expansion of tut, now it's too long not to stand again.

_Trong Time I go, there are also sure to discuss with his children, he Com do not close topic nhé!

_Ba The gambling people want to blindfold them. Yes, they post pictures of children 5 years prior to her baby with drug see: D. Con baby she stands next time it is about to have her husband call, do not ask her children. J of child offenders. He re the baby is a good brother's children. Brother rowlock that; ;).

Mỗi got legs throughout the holiday heaven
Blue eyes see some of the same sea
Places where only see swordsman
Know where to find us a dance

Make Ma General structure Loses

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtlh, hosiminh, Nilrem, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RIF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx OilyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: Tien Giang 3/09/2005)

MUP arm 4.xx Method by other hacnho

I was reading a tut by Madman. So, the tut is greate, but i have an other method for unpack the software of this corpt.

Tools: OllyDBG best config by hacnho, LordPE, and ImpREC WARK

Target: XVideoConverter 3.9.37

_Load Target into OllyDBG:

```
004788C3> / $ 55 PUSH EBP
004788C4 |. 8BEC MOV EBP, ESP
004788C6 |. 6A FF PUSH -1
004788C8 |. 68 88214A00 PUSH XVideoCo.004A2188
004788CD |. 68 00864700 PUSH XVideoCo.00478600; SE handler installation
```

_Bp CreateThread, Shift + F9, Ctrl + F9, F8, Ctrl + F9, F8:

```
003B980F 59 POP ECX; kernel32.7C8107FD
003B9810 BF 10893C00 MOV EDI, 3C8910
003B9815 8BCF MOV ECX, EDI
003B9817 E8 03E9FDFF CALL 0039811F
003B981C 84C0 TEST AL, AL
003B981E 75 09 JNZ SHORT 003B9829
003B9820 6A 01 PUSH 1
003B9822 8BCF MOV ECX, EDI
003B9824 E8 B93BFEFF CALL 0039D3E2
003B9829 B9 F04C3D00 MOV ECX, 3D4CF0
003B982E C705 30423C00 8> MOV DWORD PTR DS: [3C4230], 3C5E84
003B9838 E8 15380000 CALL 003BD052
003B983D 6A 00 PUSH 0
003B983F E8 0E380000 CALL 003BD052
003B9844 59 POP ECX
003B9845 33C9 XOR ECX, ECX
003B9847 380D 348F3C00 Cmp BYTE PTR DS: [3C8F34], CL
003B984D 75 36 JNZ SHORT 003B9885
003B984F A1 5C8F3C00 MOV EAX, DWORD PTR DS: [3C8F5C]
003B9854 53 PUSH EBX
003B9855 8B48 50 MOV ECX, DWORD PTR DS: [EAX +50]
```



```
003B9858 894D 08 MOV DWORD PTR SS: [EBP +8], ECX
003B985B 8B78 54 MOV EDI, DWORD PTR DS: [EAX +54]
003B985E 3378 38 XOR EDI, DWORD PTR DS: [EAX +38]
003B9861 8B58 68 MOV EBX, DWORD PTR DS: [EAX +68]
003B9864 3358 20 XOR EBX, DWORD PTR DS: [EAX +20]
003B9867 8D4D 08 LEA ECX, DWORD PTR SS: [EBP +8]
003B986A 3378 1C XOR EDI, DWORD PTR DS: [EAX +1 C]
003B986D 3358 18 XOR EBX, DWORD PTR DS: [EAX +18]
003B9870 033D 748F3C00 ADD EDI, DWORD PTR DS: [3C8F74]; XVideoCo.00400000
003B9876 E8 8577FDFF CALL 00391000
003B987B 33D2 XOR EDX, EDX
003B987D F7F3 DIV EBX
003B987F 8B0C3A MOV ECX, DWORD PTR DS: [EDX + EDI]
003B9882 5B POP EBX
003B9883 03D7 ADD EDX, EDI
003B9885 A1 5C8F3C00 MOV EAX, DWORD PTR DS: [3C8F5C]
003B988A 3148 50 XOR DWORD PTR DS: [EAX +50], ECX
003B988D A1 5C8F3C00 MOV EAX, DWORD PTR DS: [3C8F5C]
003B9892 3148 50 XOR DWORD PTR DS: [EAX +50], ECX
003B9895 A1 5C8F3C00 MOV EAX, DWORD PTR DS: [3C8F5C]
003B989A 8B16 MOV EDX, DWORD PTR DS: [ESI]
003B989C 8B48 5C MOV ECX, DWORD PTR DS: [EAX +5 C]
003B989F 3348 44 XOR ECX, DWORD PTR DS: [EAX +44]
003B98A2 3348 18 XOR ECX, DWORD PTR DS: [EAX +18]
003B98A5 030D 748F3C00 ADD ECX, DWORD PTR DS: [3C8F74]; XVideoCo.00400000
003B98AB 85D2 TEST EDX, EDX
003B98AD 75 18 JNZ SHORT 003B98C7
003B98AF 8B50 6C MOV EDX, DWORD PTR DS: [EAX +6 C]
003B98B2 FF76 18 PUSH DWORD PTR DS: [ESI +18]
003B98B5 3350 18 XOR EDX, DWORD PTR DS: [EAX +18]
003B98B8 FF76 14 PUSH DWORD PTR DS: [ESI +14]
003B98BB 3350 14 XOR EDX, DWORD PTR DS: [EAX +14]
003B98BE FF76 10 PUSH DWORD PTR DS: [ESI +10]
003B98C1 2BCA SUB ECX, EDX
003B98C3 FFD1 CALL ECX
003B98C5 EB 1D JMP SHORT 003B98E4
003B98C7 83FA 01 Cmp EDX, 1
003B98CA 75 1B JNZ SHORT 003B98E7
003B98CC FF76 04 PUSH DWORD PTR DS: [ESI +4]
003B98CF 8B50 6C MOV EDX, DWORD PTR DS: [EAX +6 C]
```

```
003B98D2 3350 18 XOR EDX, DWORD PTR DS: [EAX +18]
003B98D5 FF76 08 PUSH DWORD PTR DS: [ESI +8]
003B98D8 3350 14 XOR EDX, DWORD PTR DS: [EAX +14]
003B98DB 6A 00 PUSH 0
003B98DD FF76 0C PUSH DWORD PTR DS: [ESI + C]
003B98E0 2BCA SUB ECX, EDX
003B98E2 FFD1 CALL ECX
003B98E4 8945 FC MOV DWORD PTR SS: [EBP-4], EAX
003B98E7 8B45 FC MOV EAX, DWORD PTR SS: [EBP-4]
```

_Set Breakpoint here:

003B98E2 FFD1 CALL ECX

_Press F9, F7: OEP:

```
004274C2 55 PUSH EBP
004274C3 8BEC MOV EBP, ESP
004274C5 6A FF PUSH -1
004274C7 68 08C84200 PUSH XVideoCo.0042C808
004274CC 68 48764200 PUSH XVideoCo.00427648; JMP to msvcrt.__except_handler3
004274D1 64: A1 00000000 MOV EAX, DWORD PTR FS: [0]
004274D7 50 PUSH EAX
004274D8 64:8925 00000000> MOV DWORD PTR FS: [0], ESP
004274DF 83EC 68 SUB ESP, 68
004274E2 53 PUSH EBX
004274E3 56 PUSH ESI
004274E4 57 PUSH EDI
004274E5 8965 E8 MOV DWORD PTR SS: [EBP-18], ESP
004274E8 33DB XOR EBX, EBX
004274EA 895D FC MOV DWORD PTR SS: [EBP-4], EBX
004274ED 6A 02 PUSH 2
004274EF FF15 58C64200 CALL DWORD PTR DS: [42C658]; msvcrt.__set_app_type
004274F5 59 POP ECX
```

_ You see the special signal:

004274EF FF15 58C64200 CALL DWORD PTR DS: [42C658]; msvcrt.__set_app_type

_In Dump Window: Ctrl + G: 42C658, Set breakpoint on write, dword

_Ctrl + F2 to restart: Shift + F9

```
77C46FA3 F3: A5 REP MOVSD WORD PTR ES: [EDI], DWORD PTR DS>
77C46FA5 FF2495 B870C477 JMP DWORD PTR DS: [EDX * 4 +77 C470B8]
77C46FAC 8BC7 MOV EAX, EDI
77C46FAE BA 03000000 MOV EDX, 3
77C46FB3 83E9 04 SUB ECX, 4
77C46FB6 72 0C JB SHORT msvcrt.77C46FC4
```

_Shift + F9 again:

```
003B6C7D 8B85 14DBFFFF MOV EAX, DWORD PTR SS: [EBP-24EC]; XVideoCo.0042C658
003B6C83 83C0 04 ADD EAX, 4
003B6C86 8985 14DBFFFF MOV DWORD PTR SS: [EBP-24EC], EAX
003B6C8C ^ E9 4DFCFFFF JMP 003B68DE
003B6C91 FF15 84E23B00 CALL DWORD PTR DS: [3BE284]; kernel32.GetTickCount
```

_Scroll Down until you see:

```
003B6AA8 FF30 PUSH DWORD PTR DS: [EAX]
003B6AAA E8 26650000 CALL 003BCFD5
003B6AAF 83C4 0C ADD ESP, 0C
003B6AB2 8D85 0CC4FFFF LEA EAX, DWORD PTR SS: [EBP-3BF4]
003B6AB8 50 PUSH EAX
003B6AB9 8D85 1CC5FFFF LEA EAX, DWORD PTR SS: [EBP-3AE4]
003B6ABF 50 PUSH EAX
003B6AC0 FF15 84E33B00 CALL DWORD PTR DS: [3BE384]; msvcrt._stricmp
```

_Do You see the special call

003B6AAA E8 26650000 CALL 003BCFD5

_Ctrl + F2, Shift + F9, in CPU Window: Ctrl + G: 003B6AAA, set a breakpoint here on Execution: HE 003B6AAA. Delete hardware breakpoint on write. F9 you still here:

```
003B6AAA E8 26650000 CALL 003BCFD5
003B6AAF 83C4 0C ADD ESP, 0C
003B6AB2 8D85 0CC4FFFF LEA EAX, DWORD PTR SS: [EBP-3BF4]
003B6AB8 50 PUSH EAX
003B6AB9 8D85 1CC5FFFF LEA EAX, DWORD PTR SS: [EBP-3AE4]
```



```
003B6ABF 50 PUSH EAX
```

Call at the `_Enter` 003B6AAA:

```
003BCFD5 55 PUSH EBP
003BCFD6 8BEC MOV EBP, ESP
003BCFD8 51 PUSH ECX
003BCFD9 A1 204D3D00 MOV EAX, DWORD PTR DS: [3D4D20]
003BCFDE 53 PUSH EBX
003BCFDF 56 PUSH ESI
003BCFE0 57 PUSH EDI
```

`_Change` 55 to C3:

```
003BCFD5 C3 RETN
003BCFD6 8BEC MOV EBP, ESP
003BCFD8 51 PUSH ECX
003BCFD9 A1 204D3D00 MOV EAX, DWORD PTR DS: [3D4D20]
003BCFDE 53 PUSH EBX
003BCFDF 56 PUSH ESI
```

`_Ctrl + G`: Enter the address of OEP:

```
004274C2 55 PUSH EBP
004274C3 8BEC MOV EBP, ESP
004274C5 6A FF PUSH -1
004274C7 68 08C84200 PUSH XVideoCo.0042C808
004274CC 68 48764200 PUSH XVideoCo.00427648; JMP to msvcrt._except_handler3
```

`_Set` A Breakpoint here on execution, delete the breakpoint at 003B6AAA. F9, you break at the OEP. Now, open the LordPE. Choose the process XVideoConverter.exe, Full dump.

`_Open` ImpREC, enter the OEP, IAT Auto Search. Show Invalid, Cut Thunk, fix dump.

`_Now` The most important step is: Open Wark (can found at wasm.ru). Choose Utilities, Stuff PE, PE Header rebuild. Now run the target. Yeah, it Run ...

`_Unpacked` Success

`_The Lesson` for learning: Bro Madman was found a new ways for PE rebuild. This is Wark. Do

not need for change size PE Header

_Best Regards

_PS: Dumped the files can found at <http://tinicat.de> / hacnho

Môi got around the foot of heaven
Doi blue eyes of some high sea
Places where only see swordsman
NRO know where tem for the dance

Make Ma General structure Loses

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtn, hosiminh, Nilrem, Teerayoot, Ferrari, MaDMAN_H3rCuL3s, ThunderPWR, Kruger, Kelvin, Devilz, NXL ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RiF, N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx OllyDBG of the authors.

Armadillo collect sand-stone

Part 6: Armadillo 4.xx-Standard Protection other way

I. Intro

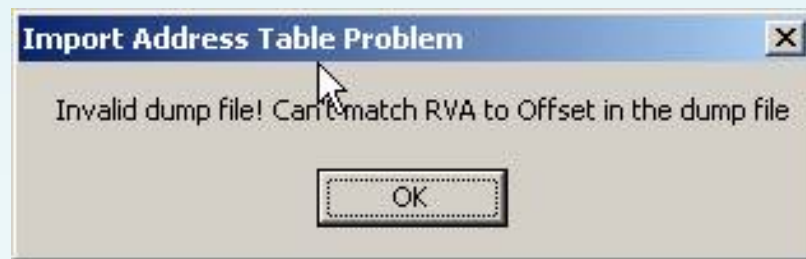
_Toi Through too sad, with all the clothes to wear (a few meals now than under this rain, it is not exposed to dry) bags run about 120 trees Tien Giang get it. Available opportunity to REA Too see the topic is a model to protect that ARTeam also Giò vắ to run the Import Table elimination. Very sour but it is very unpack Khoai. Tomorrow I am running back down, go with the bittorent hgame coming down is completed, I should write a sad paradox that this series # 6. Although when writing this line not imagine I'll write a tut tut like this, but that when finished reading this tut you will answer the questions they post today on this topic, and I will certainly suffering because of the wonders of the L. How to unpack this and the TBN EN nobody likes to why but I know some chả pa China Khoai use. I sinh foreign capital should also do not like the place because the bit TQ few words to the front without reading the charge, just like with the baby from place to stop you because "You do đi, giùm to their children in Austria in the J ... ", you probably did not sit right im not: D. Khua khua, we start all, to discuss how this itch I have such as requisite way down calme, calme down!

_De Identify signs of Armadillo's Splicing Feature Code is:

003C53D5 8985 DCAEFFFF MOV DWORD PTR SS: [EBP + FFFFAEDC], EAX
8985 # 83BD ???????? ???????? 0074? #

_Dau Brand awareness of Armadillo's Import Feature elimination is difficult to identify but the experience when I met you three cases as follows, is a destination market Armadillo's Import and elimination Feature Code Splicing:





II. Tools

_Công Only need the tools are:

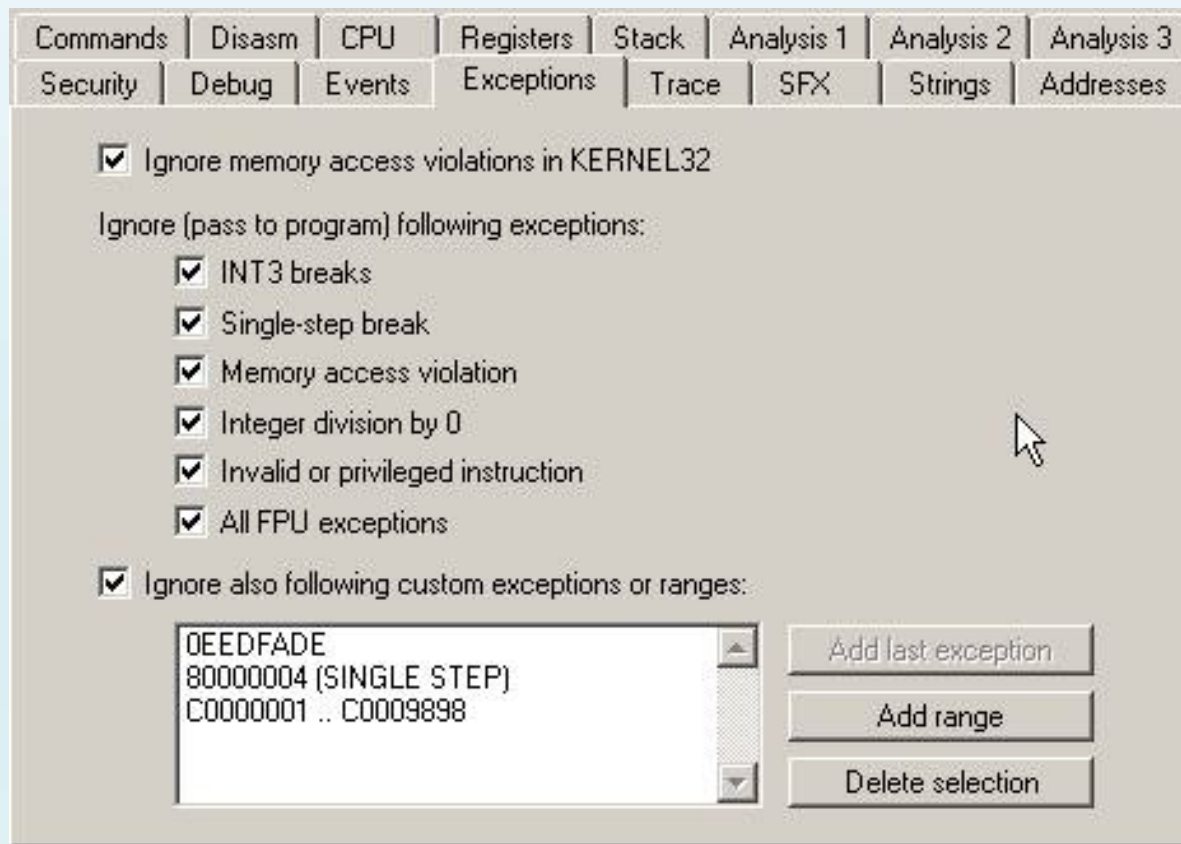
1. OllyDBG - The best config debugger for ArmMUP by hacnho.
2. LordPE 1.4 Deluxe
3. Import REConstructor 1.6 Final
4. API Address 1.0

III. Unpacking

Target # 1: PowerPoint to Flash 1.67-Armadillo 4.xx + IAT elimination



_Config Plugin:





_ Load up OllyDBG target:

00483914	55	PUSH EBP	
00483915	8BEC	MOV EBP,ESP	
00483917	6A FF	PUSH -1	
00483919	68 0054800	PUSH pptFlash.004805D0	
0048391E	68 34344800	PUSH pptFlash.00483434	
00483923	64:A1 0000000	MOV EAX,DWORD PTR FS:[0]	SE handler installation
00483929	50	PUSH EAX	
0048392A	64:8925 00000	MOV DWORD PTR FS:[0],ESP	
00483931	83EC 58	SUB ESP,58	
00483934	53	PUSH EBX	
00483935	56	PUSH ESI	pptFlash.00570718
00483936	57	PUSH EDI	ntdll.77F5164E
00483937	8965 E8	MOV [LOCAL.6],ESP	
0048393A	FF15 68A14800	CALL DWORD PTR DS:[<&KERNEL32.GetVersion>]	kernel32.GetVersion
00483940	33D2	XOR EDX,EDX	
00483942	8AD4	MOV DL,AH	
00483944	8915 840F4800	MOV DWORD PTR DS:[480F84],EDX	
0048394A	8BC8	MOV ECK,EAX	
0048394C	81F1 FF000000	AND ECK,0FF	

_Ta Need for a mutex, set a breakpoint in the function OpenMutexA: BP OpenMutexA, press F9:

0012F5B8	0047E85D	CALL to OpenMutexA from p
0012F5BC	001F0001	Access = 1F0001
0012F5C0	00000000	Inheritable = FALSE
0012F5C4	0012FBF8	MutexName = "3D0::DA3935E
0012F5C8	0012FF2C	
0012F5CC	00000000	
0012F5D0	7FFDF000	
0012F5D4	004B004A	pptFlash.004B004A
0012F5D8	004D004C	pptFlash.004D004C
0012F5DC	004F004E	pptFlash.004F004E
0012F5E0	00510050	pptFlash.00510050
0012F5E4	00530052	pptFlash.00530052

77E82391	55	PUSH EBP	
77E82392	8BEC	MOV EBP,ESP	
77E82394	51	PUSH ECK	
77E82395	51	PUSH ECK	
77E82396	837D 10 00	CMPL DWORD PTR SS:[EBP+10],0	
77E8239A	56	PUSH ESI	
77E8239B	0F84 C2E30100	JE kernel32.77EA0763	
77E823A1	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]	
77E823A7	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
77E823AA	80B0 F80B0000	LEA ESI,DWORD PTR DS:[EAX+BF8]	
77E823B0	8045 F8	LEA EAX,DWORD PTR SS:[EBP-8]	
77E823B3	50	PUSH EAX	
77E823B4	FF15 8C10E677	CALL DWORD PTR DS:[<&ntdll.RtlInitAnsiString	ntdll.RtlInitAnsiString

_Nhu So we have 0012F5C4 0012FBF8 \ MutexName = "3D0: DA3935EA83".

_Ctrl + G to enter the address memory area: **401,000**

00401000	0000	ADD BYTE PTR DS:[EAX],AL
00401002	0000	ADD BYTE PTR DS:[EAX],AL
00401004	0000	ADD BYTE PTR DS:[EAX],AL
00401006	0000	ADD BYTE PTR DS:[EAX],AL
00401008	0000	ADD BYTE PTR DS:[EAX],AL
0040100A	0000	ADD BYTE PTR DS:[EAX],AL
0040100C	0000	ADD BYTE PTR DS:[EAX],AL
0040100E	0000	ADD BYTE PTR DS:[EAX],AL
00401010	0000	ADD BYTE PTR DS:[EAX],AL
00401012	0000	ADD BYTE PTR DS:[EAX],AL
00401014	0000	ADD BYTE PTR DS:[EAX],AL
00401016	0000	ADD BYTE PTR DS:[EAX],AL
00401018	0000	ADD BYTE PTR DS:[EAX],AL

_Ctrl + E to edit as follows:

```

/ * 401000 * / PUSHAD
/ * 401001 * / PUSHFD
/ * 401002 * / PUSH 12FBF8
/ * 401007 * / XOR EAX, EAX
/ * 401009 * / PUSH EAX
/ * * 40100A / PUSH EAX
/ * * 40100B / CALL kernel32.CreateMutexA
/ * 401010 * / POPFD
/ * 401011 * / POPAD
/ * 401012 * / JMP kernel32.OpenMutexA
/ * 401017 * / ADD BYTE PTR DS: [EAX], AL

```

00401000	60	PUSHAD	
00401001	9C	PUSHFD	
00401002	68 F8FB1200	PUSH 12FBF8	ASCII "300::DA3935EA83"
00401007	33C0	XOR EAX,EAX	
00401009	50	PUSH EAX	
0040100A	58	PUSH EAX	
0040100B	E8 B5A6A777	CALL kernel32.CreateMutexA	
00401010	9D	POPFD	
00401011	61	POPAD	
00401012	E9 7A13A877	JMP kernel32.OpenMutexA	
00401017	0000	ADD BYTE PTR DS:[EAX],AL	
00401019	0000	ADD BYTE PTR DS:[EAX],AL	
0040101B	0000	ADD BYTE PTR DS:[EAX],AL	
0040101D	0000	ADD BYTE PTR DS:[EAX],AL	

_ At 401,000 you please press F9 right once! You will return:

77E82391	55	PUSH EBP	
77E82392	8BEC	MOV EBP,ESP	
77E82394	51	PUSH ECK	
77E82395	51	PUSH ECK	
77E82396	837D 10 00	CMP DWORD PTR SS:[EBP+10],0	
77E8239A	56	PUSH ESI	
77E8239B	7E 84 C2E30100	JE kernel32.77EA0763	
77E823A1	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]	
77E823A7	FF75 10	CMPL DWORD PTR SS:[EBP+10]	
77E823AA	8DB0 F80B0000	LEA ESI,DWORD PTR DS:[EAX+BF8]	
77E823B0	8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]	
77E823B3	50	POP EAX	

_Ctrl + G to the address 401000, Ctrl + * set at 401,000 new origin:

```

00401000  00  PUSHAD
00401001  9C  PUSHFD
00401002  68  F6FB1300  PUSH 12FBF8
00401007  33C0  XOR EAX,EAX
00401009  50  PUSH EAX
0040100A  50  PUSH EAX
0040100B  E8  B5A6A777  CALL kernel32.CreateMutexA
00401010  90  POPFD
00401011  61  POPAD
00401012  E9  7A18A677  JMP kernel32.OpenMutexA
00401017  00  RETN 578

```

ASCII "300::0A3935EA83"

_Neu Do not correct you will not be implemented GetModuleHandleA function, Okie, now you press Alt + F1: BC OpenMutexA, BP GetModuleHandleA.

_F9 Time 1:

```

0012ED04  77C359FC  CALL to GetModuleHandleA from msvort.77C359FC
0012ED08  77C131AC  pModule = "kernel32.dll"
0012ED0C  77C5CA20  msvort.77C5CA20
0012ED10  00000000
0012ED14  77C1E94F  msvort.<ModuleEntryPoint>
0012ED18  77F59A7B  RETURN to ntdll.77F59A7B from ntdll.77F78C4E
0012ED1C  0000E323
0012ED20  0012ED0C
0012ED24  77E7D173  kernel32.GetEnvironmentVariableA
0012ED28  0012EF00  Pointer to next SEH record
0012ED2C  77C33EB0  SE handler
0012ED30  77C131C0  msvort.77C131C0

```

_F9 Times 2:

```

0012ED04  77C359FC  CALL to GetModuleHandleA from msvort.77C359FC
0012ED08  77C131AC  pModule = "kernel32.dll"
0012ED0C  77C5CA20  msvort.77C5CA20
0012ED10  00000000
0012ED14  77C1E94F  msvort.<ModuleEntryPoint>
0012ED18  77F59A7B  RETURN to ntdll.77F59A7B from ntdll.77F78C4E
0012ED1C  0000E323
0012ED20  0012ED0C
0012ED24  77E7D173  kernel32.GetEnvironmentVariableA
0012ED28  0012EF00  Pointer to next SEH record
0012ED2C  77C33EB0  SE handler
0012ED30  77C131C0  msvort.77C131C0

```

_F9 Times 3:

```

0012ECF0  7712B124  CALL to GetModuleHandleA from OLEAUT32.7712B124
0012ECF4  771A22E4  pModule = "KERNEL32.DLL"
0012ECF8  7712AD56  RETURN to OLEAUT32.7712AD56 from OLEAUT32.7712AD56
0012ECFC  771A2080  OLEAUT32.771A2080
0012ED00  00000313
0012ED04  7712B0C0  RETURN to OLEAUT32.7712B0C0 from OLEAUT32.7712B0C0
0012ED08  00000001
0012ED0C  00000001
0012ED10  77120080  OLEAUT32.77120080
0012ED14  00000000
0012ED18  00000001
0012ED1C  7712007F  OLEAUT32.7712007F

```

_F9 Times 4:

```

0012ECEC  7712B124  CALL to GetModuleHandleA from OLEAUT32.7712B124
0012ECF0  771A22E4  pModule = "KERNEL32.DLL"
0012ECF4  7712ADAC  RETURN to OLEAUT32.7712ADAC from OLEAUT32.7712ADAC
0012ECF8  771A2064  OLEAUT32.771A2064
0012ECFC  000003E8
0012ED00  7712B0D0  RETURN to OLEAUT32.7712B0D0 from OLEAUT32.7712B0D0
0012ED04  00000001
0012ED08  00000001
0012ED0C  00000001
0012ED10  77120080  OLEAUT32.77120080
0012ED14  00000000
0012ED18  00000001

```

_F9 Times 5:

```

0012F570 00470009 CALL to GetModuleHandleA from pptFlash.00470009
0012F574 00000000 pModule = NULL
0012F578 0012F588
0012F57C 003B47E0
0012F580 0048C694 ASCII 04,"z)"
0012F584 00000000
0012F588 004AA000 ASCII "PDATA000"
0012F58C 7FFDEC00 UNICODE "RPCRT4.dll"
0012F590 0048DCA4 pptFlash.0048DCA4
0012F594 0048DCA8 pptFlash.0048DCA8
0012F598 0048A228 ASCII "fn<"
0012F59C 0048C694 ASCII 04,"z)"

```

_F9 Times 6:

```

0012E00C 003B600B CALL to GetModuleHandleA from 003B600B
0012E010 003CB808 pModule = "kernel32.dll"
0012E014 0048EE94 pptFlash.0048EE94
0012E018 0048EE94 pptFlash.0048EE94
0012E01C 0012F590
0012E020 0012F590
0012E024 0012E048
0012E028 003B86C5 RETURN to 003B86C5 from 003B86EA
0012E02C 0012E014
0012E030 00000008
0012E034 0012F44C Pointer to next SEH record
0012E038 003C3503 SE handler

```

_F9 Times 7:

```

0012E00C 003B600B CALL to GetModuleHandleA from 003B600B
0012E010 003CB808 pModule = "kernel32.dll"
0012E014 0048EE94 pptFlash.0048EE94
0012E018 0048EE94 pptFlash.0048EE94
0012E01C 0012F590
0012E020 0012F590
0012E024 0012E048
0012E028 003B86C5 RETURN to 003B86C5 from 003B86EA
0012E02C 0012E014
0012E030 00000008
0012E034 0012F44C Pointer to next SEH record
0012E038 003C3503 SE handler

```

_F9 times 8:

```

0012E048 003C0375 CALL to GetModuleHandleA from 003C036F
0012E04C 00C42008 pModule = "SHLWAPI.dll"
0012E050 0048EE94 pptFlash.0048EE94
0012E054 0048EE94 pptFlash.0048EE94
0012E058 0012F590
0012E05C 00000000
0012E060 00000151
0012E064 00000000
0012E068 0000007E
0012E06C 00000000
0012E070 00000153
0012E074 00000009

```

_F9 Times 9 (J good number):

```

0012E02C 003B653E CALL to GetModuleHandleA from 003B6538
0012E030 00000000 pModule = NULL
0012E034 0048EE94 pptFlash.0048EE94
0012E038 0048EE94 pptFlash.0048EE94
0012E03C 0012F590
0012E040 0012F44C
0012E044 003C05A3 RETURN to 003C05A3 from 003B6522
0012E048 70A70000 SHLWAPI.70A70000
0012E04C 00C4201C ASCII "PathIsUNC"
0012E050 0048EE94 pptFlash.0048EE94
0012E054 0048EE94 pptFlash.0048EE94
0012E058 0012F590

```

_Khi Press F9 to here, you press Alt + F9 execute Till User Code:

003B6532	75 03	JNZ SHORT 003B6537	
003B6534	8B5D 0C	MOV EBX,DWORD PTR SS:[EBP+C]	
003B6537	57	PUSH EDI	
003B6538	FF15 CC903C00	CALL DWORD PTR DS:[3C90CC]	kernel32.GetModuleHandleA
003B653E	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	SHLWAPI.70A70000
003B6541	3BC8	CMP ECX,EAX	pptFlash.00400000
003B6543	75 07	JNZ SHORT 003B654C	
003B6545	B8 A8B33C00	MOV EAX,3CB3A8	
003B654A	EB 2F	JMP SHORT 003B657B	
003B654C	390D D0B73C00	CMP DWORD PTR DS:[3CB7D8],EDI	
003B6552	B8 D8B73C00	MOV EAX,3CB7D8	
003B6557	74 0C	JE SHORT 003B6565	
003B6559	3B48 08	CMPEB ECX,DWORD PTR DS:[EAX+8]	
003B655C	74 1A	JE SHORT 003B6578	
003B655E	83C0 0C	ADD EAX,0C	
003B6561	3938	CMP DWORD PTR DS:[EAX],EDI	
003B6563	75 F4	JNZ SHORT 003B6559	
003B6565	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	SHLWAPI.70A70000
003B6568	FF75 08	PUSH DWORD PTR SS:[EBP+8]	kernel32.GetProcAddress
003B656B	FF15 F8903C00	CALL DWORD PTR DS:[3C90F8]	pptFlash.0048EE94
003B6571	5F	POP EDI	pptFlash.0048EE94
003B6572	5E	POP ESI	pptFlash.0048EE94
003B6573	5B	POP EBX	pptFlash.0048EE94
003B6574	5D	POP EBP	pptFlash.0048EE94
003B6575	C2 0000	RETN 8	
003B6578	8B4D 04	MOV EAX,DWORD PTR DS:[EAX+4]	
003B657B	3BC7	CMP EAX,EDI	
003B657D	74 E6	JE SHORT 003B6565	
003B657F	3978 08	CMP DWORD PTR DS:[EAX+8],EDI	
003B6582	8BF0	MOV ESI,EAX	
003B6584	74 DF	JE SHORT 003B6565	pptFlash.00400000
003B6586	55	ENDP	

Magic Jump. Change thanh
EB

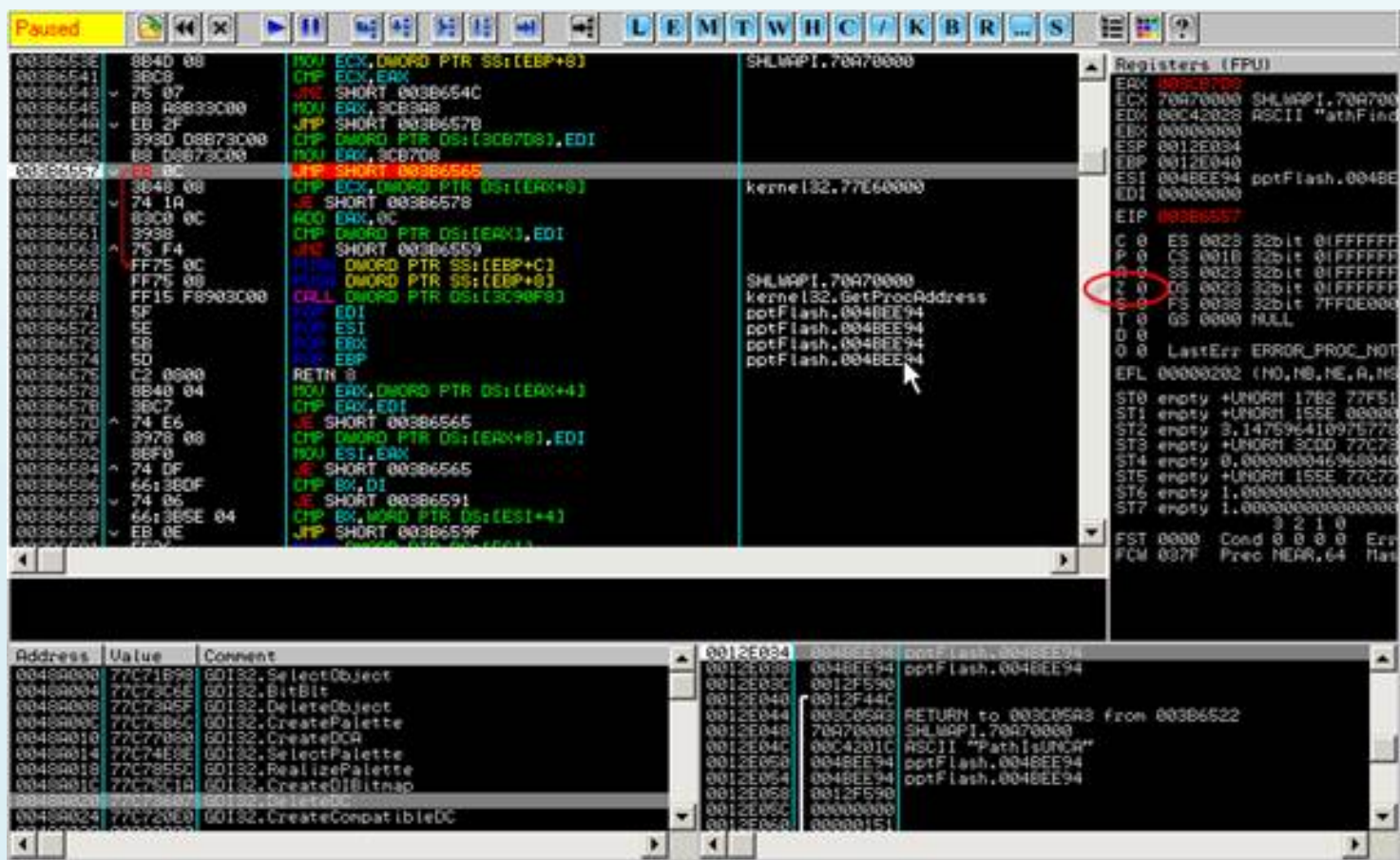
_Yeah, Magic jump!

003B653E	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	SHLWAPI.70A70000
003B6541	3BC8	CMP ECX,EAX	pptFlash.00400000
003B6543	75 07	JNZ SHORT 003B654C	
003B6545	B8 A8B33C00	MOV EAX,3CB3A8	
003B654A	EB 2F	JMP SHORT 003B657B	
003B654C	390D D0B73C00	CMP DWORD PTR DS:[3CB7D8],EDI	
003B6552	B8 D8B73C00	MOV EAX,3CB7D8	
003B6557	74 0C	JE SHORT 003B6565	
003B6559	3B48 08	CMPEB ECX,DWORD PTR DS:[EAX+8]	
003B655C	74 1A	JE SHORT 003B6578	
003B655E	83C0 0C	ADD EAX,0C	

_Ban Look through FPU:

```
Registers (FPU)
EAX 00400000 pptFlash.00400000
ECX 00000000
EDX 00C42028 ASCII "athFind
EBX 00000000
ESP 0012E034
EBP 0012E040
ESI 0048EE94 pptFlash.0048EE94
EDI 00000000
EIP 003B653E
C 0 ES 0023 32bit 0(FFFFFFF
P 1 CS 001B 32bit 0(FFFFFFF
A 0 SS 0023 32bit 0(FFFFFFF
Z 1 DS 0023 32bit 0(FFFFFFF
S 0 FS 0038 32bit 7FFDE000
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_PROC_NOT
EFL 00000246 (NO,NB,E,BE,NS
```

_Co Z is 1, we started to trace magic jump it changed to 0:



_Buoc Important here is: When you jump to the magic, right click at the Z:



_Set:



_Nhan Alt + F1: BC GetModuleHandleA

_Okie, Now you press Alt + M:

003E0000	00027000				Priv	RW		
003E0000	00004000				Priv	RW		
003F0000	00002000				Map	R		
00400000	00001000	pptFlash		PE header	Imag	R		
00401000	00058000	pptFlash	.text					
0045C000	00017000	pptFlash	.rdata					
00473000	00007000	pptFlash	.data					
0047A000	00010000	pptFlash	.text1	code				
0048A000	00020000	pptFlash	.data1	data				
0049A000	00060000	pptFlash	.pdata					
0050A000	00124000	pptFlash	.rsrc	reso				
00630000	00008000							
006F0000	00002000							
00700000	00103000							
00810000	00126000							
00B10000	00001000							
00C10000	00005000							
00C20000	00002000							
00C30000	00024000							
629C0000	00001000	LPK		PE h				
629C1000	00004000	LPK	.text	code				

Actualize
 Dump in CPU
 Dump
 Search Ctrl+B
 Set break-on-access F2
 Set memory breakpoint on access
 Set memory breakpoint on write
 Set access

_Shift + F9: OEP, wow!

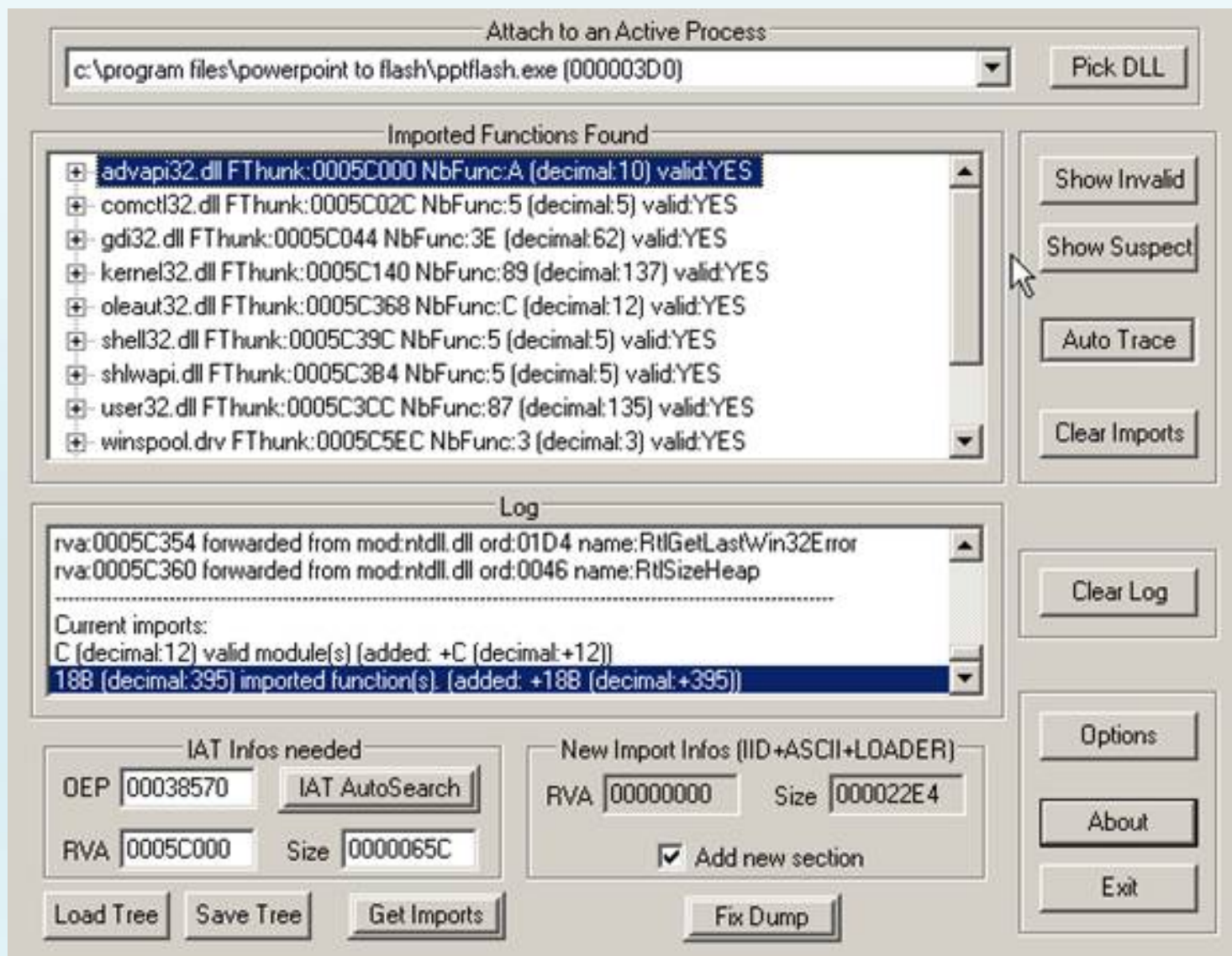
00438570	6A 5D	PUSH 60	
00438572	68 98534600	PUSH pptFlash.00465398	
00438577	E8 30200000	CALL pptFlash.004388AC	
0043857C	8F 94000000	MOV EDI,94	
00438581	88C7	MOV EAX,EDI	pptFlash.00438570
00438583	E8 A8FAFFFF	CALL pptFlash.00438030	
00438588	8965 E8	MOV DWORD PTR SS:[EBP-10],ESP	
0043858B	88F4	MOV ESI,ESP	
0043858D	893E	MOV DWORD PTR DS:[ESI],EDI	pptFlash.00438570
0043858F	56	PUSH ESI	pptFlash.0048DA70
00438590	FF15 B4C14500	CALL DWORD PTR DS:[45C1B4]	kernel32.GetVersionExA
00438596	884E 10	MOV ECX,DWORD PTR DS:[ESI+10]	
00438599	8980 14794700	MOV DWORD PTR DS:[477914],ECX	
0043859F	8846 04	MOV EAX,DWORD PTR DS:[ESI+4]	
004385A2	A3 20794700	MOV DWORD PTR DS:[477920],EAX	pptFlash.00400000
004385A7	8856 08	MOV EDX,DWORD PTR DS:[ESI+8]	
004385AA	8915 24794700	MOV DWORD PTR DS:[477924],EDX	
004385B0	8876 0C	MOV ESI,DWORD PTR DS:[ESI+C]	
004385B3	81E6 FF7F0000	AND ESI,7FFF	
004385B9	8935 18794700	MOV DWORD PTR DS:[477918],ESI	pptFlash.0048DA70
004385BF	83F9 02	CMP ECX,2	
004385C2	74 0C	JE SHORT pptFlash.004385D0	
004385C4	81CE 00000000	OR ESI,0000	
004385CA	8935 18794700	MOV DWORD PTR DS:[477918],ESI	pptFlash.0048DA70
004385D0	C1E0 08	SHL EAX,8	
004385D3	83C2	ADD EAX,EDX	
004385D5	A3 1C794700	MOV DWORD PTR DS:[47791C],EAX	pptFlash.00400000
004385DA	33F6	XOR ESI,ESI	pptFlash.0048DA70
004385DC	56	PUSH ESI	pptFlash.0048DA70
004385DD	883D F8C14500	MOV EDI,DWORD PTR DS:[45C1F8]	kernel32.GetModuleHandleA
004385E3	FFD7	CALL EDI	pptFlash.00438570
004385F5	83C2	ADD EAX,EDX	

_Lord PE DumpFull:

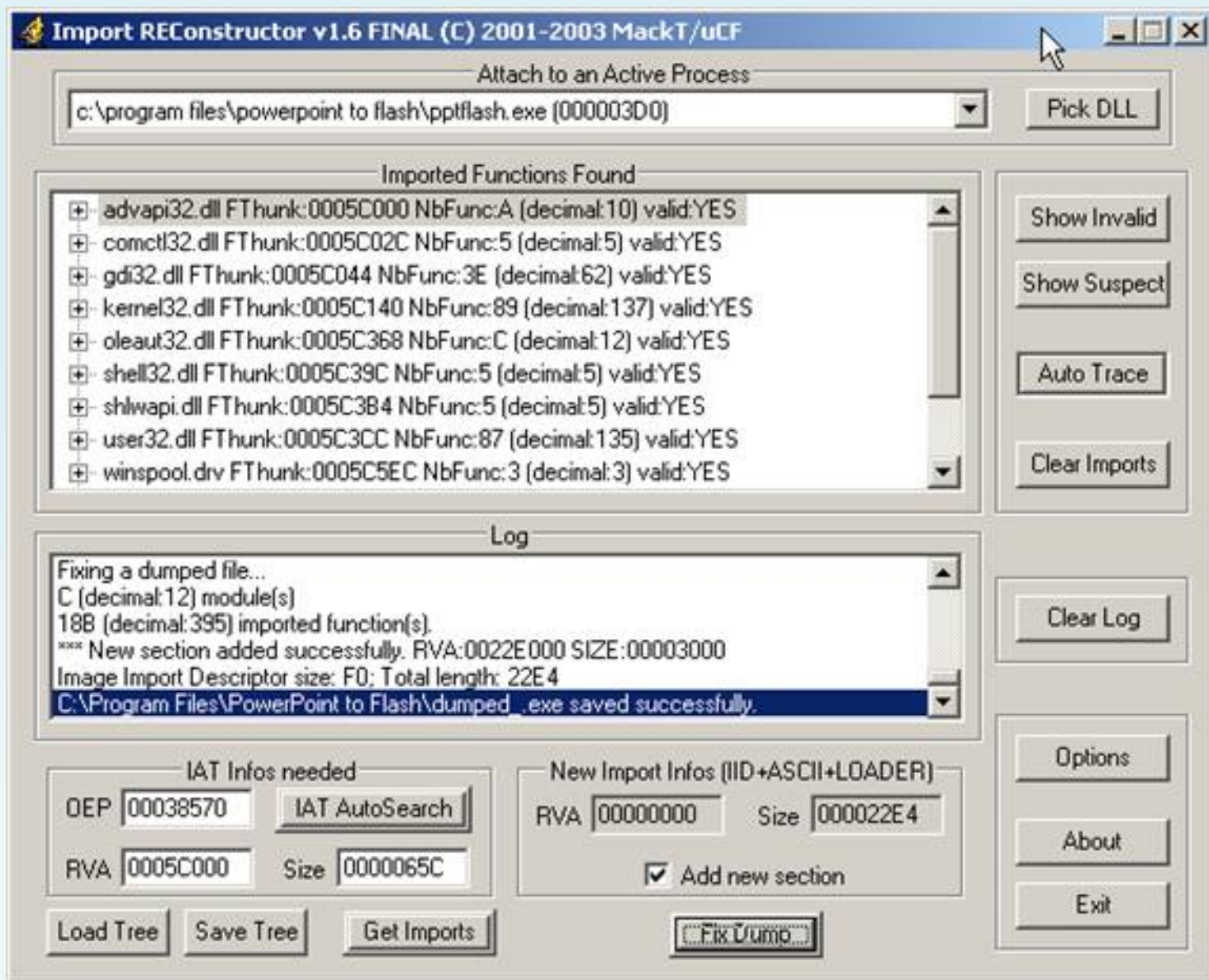
Path	PID	ImageBase	ImageSize
i:\project delphi7\my project\crackerutils\cra...	0000068C	00400000	000F4000
c:\program files\winamp\winamp.exe	000006EC	00400000	0012E000
c:\program files\techsmith\snagit 7\snagit32...	0000045C	00400000	00390000
c:\program files\techsmith\snagit 7\tschelp.exe	00000200	00400000	00004000
c:\program files\emeditor\emeditor.exe	00000428	00400000	00069000
[system]	000006A8	00000000	00000000
c:\program files\internet explorer\iexplore.exe	000000B8	00400000	00019000
[system]	000003D8	00000000	00000000
c:\program files\microsoft office\office11\win...	00000310	30000000	008AA000
c:\program files\unikey\unikey.exe	00000348	00400000	0002F000
i:\cracker\debug-disassembler\odbg110_orq...	00000724	00400000	00169000
c:\program file	000	00400000	0022E000
i:\cracker\utili	000	00400000	00036000
Path	ase	ImageSize	
c:\program file	000	0022E000	
c:\windows\s	000	000A7000	
c:\windows\s	000	000E6000	
c:\windows\s	000	0008C000	

_ImpREC:





Invalid _Nothing, yeah!



Run Try dumped.exe:



Target # 2: FRAPS 2.6.4-Armadillo 4.xx + IAT elimination + Code Splicing



_Load OllyDBG on target:

Address	Hex dump	Disassembly	Comment
00E02433	55	PUSH EBP	
00E02434	8BEC	MOV EBP,ESP	
00E02436	6A FF	PUSH -1	
00E02438	68 88C1E200	PUSH fraps.00E2C188	
00E0243D	68 7021E000	PUSH fraps.00E02170	
00E02442	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	SE handler installation
00E02448	50	PUSH EAX	
00E02449	64:8925 0000	MOV DWORD PTR FS:[0],ESP	
00E02450	83EC 58	SUB ESP,58	
00E02453	53	PUSH EBX	
00E02454	55	PUSH ESI	

_Ta Need for a mutex, set a breakpoint in the function OpenMutexA: BP OpenMutexA, press F9

Address	Hex dump	Disassembly	Comment
77E82391	55	PUSH EBP	
77E82392	8BEC	MOV EBP,ESP	
77E82394	51	PUSH ECX	
77E82395	51	PUSH ECX	
77E82396	837D 10 00	CMP DWORD PTR SS:[EBP+10],0	
77E8239A	56	PUSH ESI	
77E8239B	0F84 C2E30100	JE kernel32.77E80763	
77E823A1	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]	
77E823A7	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
77E823A9	8080 F8000000	LEA ESI,DWORD PTR DS:[EAX+BF8]	

Address	Value	Comment
0012F738	000E0018	CALL to OpenMutexA from fraps.000E0015
0012F73C	001F0001	Access = 1F0001
0012F740	00000000	Inheritable = FALSE
0012F744	0012FDC8	MutexName = "91C::DA802AD515"
0012F748	00000000	
0012F74C	00000000	
0012F750	7FFDF000	
0012F754	00380000	
0012F758	00381AC8	
0012F75C	00380000	

_Ok, So we have: 0012F744 0012FDC8 \ MutexName = "638:: DA802AD515"

_Ctrl + G to enter the address memory area: **401,000**

Address	Hex dump	Disassembly	Comment
00401000	0000	ADD BYTE PTR DS:[EAX],AL	
00401002	0000	ADD BYTE PTR DS:[EAX],AL	
00401004	0000	ADD BYTE PTR DS:[EAX],AL	
00401006	0000	ADD BYTE PTR DS:[EAX],AL	
00401008	0000	ADD BYTE PTR DS:[EAX],AL	
0040100A	0000	ADD BYTE PTR DS:[EAX],AL	
0040100C	0000	ADD BYTE PTR DS:[EAX],AL	
0040100E	0000	ADD BYTE PTR DS:[EAX],AL	
00401010	0000	ADD BYTE PTR DS:[EAX],AL	

_Ctrl + E to edit as follows:

Address	Hex dump	Disassembly	Comment
00401000	50	PUSHAD	
00401001	9C	PUSHFD	
00401002	68 C8FD1200	PUSH 12FDC8	ASCII "91C::DA802AD515"
00401007	33C0	XOR EAX,EAX	
00401009	50	PUSH EAX	
0040100A	50	PUSH EAX	
0040100B	E8 85A6A777	CALL kernel32.CreateMutexA	
00401010	90	POPF	
00401011	51	POPAD	
00401012	E9 7A13A877	JMP kernel32.OpenMutexA	
00401017	90	NOP	
00401018	0000	ADD BYTE PTR DS:[EAX],AL	
0040101A	0000	ADD BYTE PTR DS:[EAX],AL	
0040101C	0000	ADD BYTE PTR DS:[EAX],AL	
0040101E	0000	ADD BYTE PTR DS:[EAX],AL	
00401020	0000	ADD BYTE PTR DS:[EAX],AL	
00401022	0000	ADD BYTE PTR DS:[EAX],AL	

PUSH is _Dong 12FDC8 address Mutex: 0012F744 0012FDC8 \ MutexName = "638:: DA802AD515".

_Nhan F9:

Address	Value	Comment
0012E7AC	003ACCF8	CALL to OpenMutexA from 003ACCF2
0012E7B0	001F0001	Access = 1F0001 Inheritable = FALSE MutexName = "14CEB363:SIMULATEEXPIRED"
0012E7B4	00000000	
0012E7B8	0012E7C8	
0012E7BC	003D8910	
0012E7C0	CEDD71E6	
0012E7C4	00000000	
0012E7C8	45433431	
0012E7CC	33363342	
0012E7D0	4D49533A	

_Xoa OpenMutexA breakpoint. Ctrl + G, enter GetModuleHandleA, press F2 to set a.

_F9 Time 1:

Address	Value	Comment
0012967C	003C30CE	CALL to GetModuleHandleA from 003C30C8
00129680	003D3D6C	pModule = "kernel32.dll"
00129684	003D5D70	
00129688	003D8910	ASCII "VirtualAlloc"
0012968C	CEDD71E6	
00129690	00000000	
00129694	00000000	
00129698	00000000	
0012969C	00000000	
001296A0	00000000	

_Lan 2:

Address	Value	Comment
0012967C	003C30EB	CALL to GetModuleHandleA from 003C30E5
00129680	003D3D6C	pModule = "kernel32.dll"
00129684	003D5D64	
00129688	003D8910	ASCII "VirtualFree"
0012968C	CEDD71E6	
00129690	00000000	
00129694	00000000	
00129698	00000000	
0012969C	00000000	
001296A0	00000000	

_Lan 3:

Address	Value	Comment
0012941C	003B5386	CALL to GetModuleHandleA from 003B5380
00129420	00129558	pModule = "kernel32.dll"
00129424	0012EBA4	
00129428	2CF92316	
0012942C	00000000	
00129430	003D3244	
00129434	00000000	
00129438	00000000	
0012943C	00129488	
00129440	77C37BBE	RETURN to MSVCRT.77C37BBE from ntdll.RtlSetLa

Address	Hex dump	Disassembly	Comment
77E7AD86	837C24 04 00	CMP DWORD PTR SS:[ESP+4],0	
77E7AD88	0F84 37010000	JE kernel32.77E7AEC8	
77E7AD91	FF7424 04	PUSH DWORD PTR SS:[ESP+4]	
77E7AD95	E8 F8050000	CALL kernel32.77E7B992	
77E7AD9A	85C9	TEST EAX,EAX	
77E7AD9C	74 08	JE SHORT kernel32.77E7ADA6	
77E7AD9E	FF70 04	PUSH DWORD PTR DS:[EAX+4]	
77E7ADA1	E8 27060000	CALL kernel32.GetModuleHandleW	
77E7ADA5	C2 0400	SETL E	
77E7ADA9	55	PUSH EBP	
77E7ADAA	8BEC	MOV EBP,EAX	
77E7ADAC	83EC 10	SUB ESP,10	
77E7ADAF	57	PUSH EDI	
77E7ADB0	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]	
77E7ADB6	8B7D 10	MOV EDI,DWORD PTR SS:[EBP+10]	
77E7ADB9	8B40 30	MOV EAX,DWORD PTR DS:[EAX+30]	

_Tro The CPU, press F8 function trace down through RETN4

Address	Hex dump	Disassembly	Comment
003B5385	8B0D 24CF3D00	MOV ECX,DWORD PTR DS:[3DCF24]	
003B538C	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003B538F	A1 24CF3D00	MOV EAX,DWORD PTR DS:[3DCF24]	
003B5394	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
003B5397	75 16	JNZ SHORT 003B53AF	
003B5399	8D85 DCFEFFFF	LEA EAX,DWORD PTR SS:[EBP-124]	
003B539F	58	PUSH EAX	
003B53A0	FF15 90E03C00	CALL DWORD PTR DS:[3CE090]	kernel32.LoadLibraryA
003B53A6	8B0D 24CF3D00	MOV ECX,DWORD PTR DS:[3DCF24]	
003B53AC	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003B53AF	A1 24CF3D00	MOV EAX,DWORD PTR DS:[3DCF24]	
003B53B4	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
003B53B7	0F84 2F010000	JE 003B54EC	
003B53BD	23C9	XOR ECX,ECX	
003B53BF	8B03	MOV EAX,DWORD PTR DS:[EBX]	
003B53C1	3938	CMP DWORD PTR DS:[EAX],EDI	
003B53C3	74 06	JE SHORT 003B53CB	
003B53C5	41	INC ECX	
003B53C6	83C0 0C	ADD EAX,0C	
003B53C9	EB F6	JNE SHORT 003B53C1	
003B53CB	8BF9	MOV EDI,ECX	
003B53CD	C1E7 02	SHL EDI,2	
003B53D0	57	PUSH EDI	
003B53D1	E8 BC7B0100	CALL 003CCF92	JMP to MSUCRT.??20VAPAKI0Z
003B53D6	8B0D 1CCF3D00	MOV ECX,DWORD PTR DS:[3DCF1C]	
003B53DC	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003B53DF	57	PUSH EDI	
003B53E0	E8 AD7B0100	CALL 003CCF92	JMP to MSUCRT.??20VAPAKI0Z
003B53E5	59	POP ECX	
003B53E6	59	POP ECX	

_Patch To 003B54EC Jmp:

Address	Hex dump	Disassembly	Comment
003B5385	8B0D 24CF3D00	MOV ECX,DWORD PTR DS:[3DCF24]	
003B538C	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003B538F	A1 24CF3D00	MOV EAX,DWORD PTR DS:[3DCF24]	
003B5394	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
003B5397	75 16	JNZ SHORT 003B53AF	
003B5399	8D85 DCFEFFFF	LEA EAX,DWORD PTR SS:[EBP-124]	
003B539F	58	PUSH EAX	
003B53A0	FF15 90E03C00	CALL DWORD PTR DS:[3CE090]	kernel32.LoadLibraryA
003B53A6	8B0D 24CF3D00	MOV ECX,DWORD PTR DS:[3DCF24]	
003B53AC	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003B53AF	A1 24CF3D00	MOV EAX,DWORD PTR DS:[3DCF24]	
003B53B4	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
003B53B7	E9 50010000	JMP 003B54EC	
003B53BC	90	NOP	
003B53BD	33C9	XOR ECX,ECX	
003B53BF	8B03	MOV EAX,DWORD PTR DS:[EBX]	
003B53C1	3938	CMP DWORD PTR DS:[EAX],EDI	
003B53C3	74 06	JE SHORT 003B53CB	
003B53C5	41	INC ECX	

_Hd GetModuleHandleA, press Alt + M, set breakpoint on access at 401,000

Memory map, item 22

Address = 00401000

Size = 00012000 (73728.)

Owner = 00400000 fraps

Section = . text

Type = 01001002 IMAG

Access R =

Initial access = RWE

_F9 2 times:

Address	Hex dump	Disassembly	Comment
0040C434	5B	PUSH EBP	
0040C435	8BEC	MOV EBP,ESP	
0040C437	6A FF	PUSH -1	
0040C439	68 30304100	PUSH fraps.00413328	
0040C43E	68 30E94000	PUSH fraps.0040E930	
0040C443	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
0040C449	50	PUSH EAX	
0040C44A	64:9325 00000000	MOV DWORD PTR FS:[0],ESP	
0040C451	8BEC 58	SUB ESP,58	
0040C454	50	PUSH EAX	
0040C455	56	PUSH ESI	
0040C456	57	PUSH EDI	
0040C457	8B65 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0040C45A	FF15 04377001	CALL DWORD PTR DS:[17037B4]	kernel32.GetVersion
0040C460	33D2	XOR EDX,EDX	
0040C462	8AD4	MOV DL,AH	
0040C464	8915 9C000C00	MOV DWORD PTR DS:[DC009C],EDX	
0040C46A	8BCB	MOV ECX,EAX	
0040C46C	81E1 FF000000	AND ECX,0FF	
0040C472	8900 90000C00	MOV DWORD PTR DS:[DC0090],ECX	
0040C478	C1E1 08	SHL ECX,8	
0040C47B	83CB	ADD ECX,EDX	
0040C47D	8900 94000C00	MOV DWORD PTR DS:[DC0094],ECX	
0040C483	C1E8 10	SHR EAX,10	
0040C486	A3 90000C00	MOV DWORD PTR DS:[DC0090],EAX	
0040C48B	33F6	XOR ESI,ESI	
0040C48D	56	PUSH ESI	
0040C48E	E8 47230000	CALL fraps.0040E7DA	
0040C493	59	POP ECX	
0040C494	85C9	TEST EAX,EAX	

_Yeah, OEP! Dump only:

Path	PID	ImageBase	ImageSize
c:\windows\system32\msdtc.exe	000006AC	00400000	00004000
c:\windows\system32\spoolsv.exe	00000120	01000000	0000F000
c:\program files\executive software\diskeepe...	00000528	00400000	000B9000
c:\program files\winamp\winamp.exe	000005CC	00400000	00130000
c:\program files\emeditor\emeditor.exe	00000734	00400000	00069000
c:\program files\microsoft office\office11\win...	0000010C	30000000	008AA000
c:\program files\techsmith\snagit 7\snagit32...	0000019C	00400000	00390000
c:\program files\techsmith\snagit 7\tschelp.exe	00000308	00400000	0000A000
c:\program files\unikey\unikey.exe	000007C8	00400000	0002F000
i:\cracker\debug- disassembler\odbg110_org...	00000874	00400000	00164000
c:\program files\internet explorer\iexplore.exe	00000A80	00400000	00019000
c:\fraps\fraps.exe	dump full...	00400000	
i:\cracker\utilities\lordpe deluxe - 1.4\lordp...	dump partial...	00400000	000036000
	dump region...		



_Tiep To remove the old breakpoint, set breakpoint: BP VirtualProtect. F9:

Address	Value	Comment
0012F5F4	00DEC966	CALL to VirtualProtect from fraps.00DEC960
0012F5F8	003A1000	Address = 003A1000
0012F5FC	0002C3BA	Size = 2C3BA (181178.)
0012F600	00000020	NewProtect = PAGE_EXECUTE_READ
0012F604	0012F658	pOldProtect = 0012F658
0012F608	00000004	
0012F60C	00000000	
0012F610	7FFDF000	
0012F614	77E60000	kernel32.77E60000
0012F618	77E600F8	ASCII "PE"

_F9 To run it, remember the last time F9. My time here is 18:

Address	Value	Comment
00129650	003C8671	CALL to VirtualProtect from 003C866F
00129654	004000F0	Address = fraps.004000F0
00129658	000000F8	Size = F8 (248.)
0012965C	00000002	NewProtect = PAGE_READONLY
00129660	0012967C	pOldProtect = 0012967C
00129664	3FC8768C	
00129668	0012B900	
0012966C	00000000	
00129670	00000238	
00129674	01C563EF	

Address	Hex dump	Disassembly	Comment
77E6169E	55	MOV ESP	
77E6169F	8BEC	MOV EBP,ESP	
77E616A1	FF75 14	PUSH DWORD PTR SS:[EBP+14]	
77E616A4	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
77E616A7	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
77E616AA	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
77E616AD	6A FF	PUSH -1	
77E616AF	E8 A4B00100	CALL kernel32.VirtualProtectEx	
77E616B4	5D	POP EBP	
77E616B5	C2 1000	RETI 10	
77E616B8	837C24 04 00	CMPL DWORD PTR SS:[ESP+4],0	
77E616BD	74 18	JE SHORT kernel32.77E616D7	
77E616BF	FF7424 08	PUSH DWORD PTR SS:[ESP+8]	
77E616C3	FF7424 08	PUSH DWORD PTR SS:[ESP+8]	
77E616C7	FF15 F813E677	CALL DWORD PTR DS:[<&ntdll.NtTerminateP	ntdll.ZwTerminateProcess
77E616CD	85C0	TEST EAX,EAX	
77E616CF	7C 0F	JL SHORT kernel32.77E616E8	
77E616D1	33C0	XOR EAX,EAX	
77E616D3	40	INC EAX	
77E616D4	C2 0000	RETI 0	
77E616D7	6A 06	PUSH 6	
77E616D9	5D	POP EBP	

_Tro The CPU, press F8 trace down the last return:

Address	Hex dump	Disassembly	Comment
003C8671	5E	POP ESI	kernel32.VirtualProtect
003C8672	5F	POP EDI	
003C8673	58	POP EBX	
003C8674	C9	LEAVE	
003C8675	C3	RETN	
003C8676	55	PUSH EBP	
003C8677	8BEC	MOV EBP,ESP	
003C8679	83EC 10	SUB ESP,10	
003C867C	8B4D 10	MOV ECX,DWORD PTR SS:[EBP+10]	
003C867F	8B45 14	MOV EAX,DWORD PTR SS:[EBP+14]	
003C8682	8D1401	LEA EDX,DWORD PTR DS:[ECX+EAX]	
003C8685	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
003C8688	2B45 0C	SUB EAX,DWORD PTR SS:[EBP+C]	
003C868B	8955 F4	MOV DWORD PTR SS:[EBP-C],EDX	
003C868E	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
003C8691	✓ 0F84 C1000000	JE 003C8758	
003C8697	3BCA	CMPL ECX,EDX	
003C8699	✓ 0F83 B9000000	JNB 003C8758	
003C869F	53	PUSH EBX	
003C86A0	56	PUSH ESI	
003C86A1	57	PUSH EDI	
003C86A2	8B3D 24E13C00	MOV EDI,DWORD PTR DS:[3CE124]	kernel32.VirtualProtect
003C86A8	BE 00100000	MOV ESI,1000	
003C86AD	✓ EB 06	CALL SHORT 003C86B5	
003C86AF	8B4D 10	MOV ECX,DWORD PTR SS:[EBP+10]	
003C86B2	8B55 F4	MOV EDX,DWORD PTR SS:[EBP-C]	
003C86B5	8B41 04	MOV EAX,DWORD PTR DS:[ECX+4]	
003C86B8	8B19	MOV EBX,DWORD PTR DS:[ECX]	
003C86BA	03C1	ADD EAX,ECX	
003C86BC	3BC2	CMPL EAX,EDX	

_Co Do not get used to, yeah, defeat successful:

_Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
003C2D9A	S9	POP EBX	0012F704
003C2D9B	8945 E4	MOV DWORD PTR SS:[EBP-1C],EAX	
003C2D9E	C705 30423000	MOV DWORD PTR DS:[3D4230],304B88	
003C2DA8	A1 5C8F3D00	MOV EAX,DWORD PTR DS:[3D8F5C]	
003C2DA9	8B88 94000000	MOV ECX,DWORD PTR DS:[EAX+94]	
003C2DB3	3348 70	XOR ECX,DWORD PTR DS:[EAX+70]	
003C2DB6	3348 5C	XOR ECX,DWORD PTR DS:[EAX+5C]	
003C2DB9	F7C1 04022000	TEST ECX,200204	
003C2DBF	✓ 74 2E	JE SHORT 003C2DEF	
003C2DC1	A1 6C2E3E00	MOV EAX,DWORD PTR DS:[3E2E6C]	
003C2DC6	3BC3	CMPL EAX,EBX	
003C2DC8	✓ 74 25	JE SHORT 003C2DEF	
003C2DCA	8B0D 70313E00	MOV ECX,DWORD PTR DS:[3E3170]	
003C2DD8	8948 08	MOV DWORD PTR DS:[EAX+8],ECX	
003C2DD3	A1 6C2E3E00	MOV EAX,DWORD PTR DS:[3E2E6C]	
003C2DD8	8B0D 74313E00	MOV ECX,DWORD PTR DS:[3E3174]	
003C2DE1	8948 0C	MOV DWORD PTR DS:[EAX+C],ECX	
003C2DE1	A1 6C2E3E00	MOV EAX,DWORD PTR DS:[3E2E6C]	
003C2DE6	8B0D 682E3E00	MOV ECX,DWORD PTR DS:[3E2E68]	
003C2DEC	8948 10	MOV DWORD PTR DS:[EAX+10],ECX	
003C2DEF	C705 30423000	MOV DWORD PTR DS:[3D4230],304B7C	
003C2DF9	6A 01	PUSH 1	
003C2DFB	5F	POP EDI	
003C2DFC	397D E4	CMPL DWORD PTR SS:[EBP-1C],EDI	
003C2DFF	✓ 74 36	JE SHORT 003C2E37	
003C2E01	57	PUSH EDI	
003C2E02	8B0D 58AD3000	MOV ECX,DWORD PTR DS:[3DAD58]	
003C2E08	E8 D28DFEFF	CALL 003ABBD5	
003C2E0D	8B35 58AD3000	MOV ESI,DWORD PTR DS:[3DAD58]	
003C2E13	89B5 EBF6FFFF	MOV DWORD PTR SS:[EBP-918],ESI	

_Tiep The Ctrl + F9, F8 to you here:

Address	Hex dump	Disassembly	Comment
00DEB09A	83C4 04	ADD ESP,4	
00DEB09D	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
00DEB0A0	837D EC FF	CMPL DWORD PTR SS:[EBP-14],-1	
00DEB0A4	74 0B	JL SHORT fraps.00DEB0B1	
00DEB0A6	8B55 EC	MOV EDI,DWORD PTR SS:[EBP-14]	
00DEB0A9	8915 8C05E20	MOV DWORD PTR DS:[E2D58C],EDI	
00DEB0AF	EB 10	JL SHORT fraps.00DEB0C1	
00DEB0B1	837D FC 01	CMPL DWORD PTR SS:[EBP-4],1	
00DEB0B5	74 0A	JL SHORT fraps.00DEB0C1	
00DEB0B7	C705 8C05E20	MOV DWORD PTR DS:[E2D58C],1	
00DEB0C1	B8 01000000	MOV EAX,1	
00DEB0C6	85C0	TEST EAX,EAX	
00DEB0C8	74 09	JL SHORT fraps.00DEB0D3	
00DEB0CA	C745 80 0000	MOV DWORD PTR SS:[EBP-80],0	
00DEB0D1	EB 10	JL SHORT fraps.00DEB0E3	
00DEB0D3	68 0C58E200	PUSH fraps.00E2580C	[Arg1 = 00E2580C ASCII "Back from LoadProgram"]
00DEB0D8	E8 B7B10000	CALL fraps.00DF6F94	fraps.00DF6F94
00DEB0DD	83C4 04	ADD ESP,4	
00DEB0E0	8945 80	MOV DWORD PTR SS:[EBP-80],EAX	
00DEB0E3	837D 80 00	CMPL DWORD PTR SS:[EBP-80],0	
00DEB0E7	74 0A	JL SHORT fraps.00DEB0F3	
00DEB0E9	8B4D B0	MOV ECX,DWORD PTR SS:[EBP-50]	
00DEB0EC	51	PUSH ECX	
00DEB0ED	FF15 2452E20	CALL DWORD PTR DS:[<USER32.DestroyWind	hWnd DestroyWindow
00DEB0F3	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00DEB0F6	8B45	MOV ESP,EBP	
00DEB0F8	5D	POP EBP	
00DEB0F9	C3	RETN	
00DEB0FA	55	PUSH EBP	
00DEB0FB	8B45	MOV EBP,ESP	

_Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
00DED97A	8945 E4	MOV DWORD PTR SS:[EBP-1C],EAX	
00DED97D	6A 00	PUSH 0	[Arg1 = 00000000]
00DED97F	E8 99000000	CALL fraps.000EDA1D	fraps.000EDA1D
00DED984	83C4 04	ADD ESP,4	
00DED987	6A 00	PUSH 0	
00DED989	E8 BE460100	CALL fraps.00E0204C	
00DED98E	83C4 04	ADD ESP,4	
00DED991	837D E4 01	CMPL DWORD PTR SS:[EBP-1C],1	
00DED995	75 11	JNZ SHORT fraps.00DED998	
00DED997	68 98D0E200	PUSH fraps.00E2D098	
00DED99C	FF15 C8D0E20	CALL DWORD PTR DS:[E2D08C]	
00DED9A2	83C4 04	ADD ESP,4	
00DED9A5	8945 F4	MOV DWORD PTR SS:[EBP-1C],EAX	
00DED9A8	B9 01000000	MOV ECX,1	
00DED9AD	85C9	TEST ECX,ECX	
00DED9AF	74 0C	JL SHORT fraps.00DED9B3	Enter vào hàm Call này!
00DED9B1	C785 64F8FFF	MOV DWORD PTR SS:[EBP-79C],0	
00DED9B8	EB 17	JL SHORT fraps.00DED9C4	
00DED9BD	8B55 E4	MOV EDI,DWORD PTR SS:[EBP-1C]	
00DED9C0	52	PUSH EDI	
00DED9C1	68 F85CE200	PUSH fraps.00E25CF8	[Arg2 Arg1 = 00E25CF8 ASCII "RunUserProgram returne
00DED9C6	E8 C9950000	CALL fraps.00DF6F94	fraps.00DF6F94
00DED9CB	83C4 08	ADD ESP,8	
00DED9CE	8985 64F8FFF	MOV DWORD PTR SS:[EBP-79C],EAX	
00DED9D4	68 FC89D0E0	PUSH fraps.00EB9FC	
00DED9D9	E8 6E460100	CALL fraps.00E0204C	
00DED9DE	83C4 04	ADD ESP,4	
00DED9E1	833D 8CD0E20	CMPL DWORD PTR DS:[E2D08C],0	
00DED9E8	74 06	JL SHORT fraps.00DED9F0	
00DED9EA	FF15 8CD0E20	CALL DWORD PTR DS:[E2D08C]	

_Toi Here:

Address	Hex dump	Disassembly	Comment
003C9694	55	PUSH EBP	
003C9695	8BEC	MOV EBP,ESP	
003C9697	51	PUSH ECX	
003C9698	56	PUSH ESI	
003C9699	8B75 08	MOV ESI,DWORD PTR SS:[EBP+8]	
003C969C	C605 5E2E3E00	MOV BYTE PTR DS:[3E2E5E],1	
003C96A3	C705 30423D00	MOV DWORD PTR DS:[3D4230],3D5E94	
003C96A0	833E 02	CMPL DWORD PTR DS:[ESI],2	
003C96B0	57	PUSH EDI	
003C96B1	C745 FC FEFFFF	MOV DWORD PTR SS:[EBP-4],-2	
003C96B8	75 62	JNZ SHORT 003C971C	
003C96BA	6A 00	PUSH 0	

_Cuon Down to see the beautiful signs:

Address	Hex dump	Disassembly	Comment
003C97D9	85D2	TEST EDX,EDX	
003C97DB	75 17	JNZ SHORT 003C97F4	
003C97DD	8B50 5C	MOV EDX,DWORD PTR DS:[EAX+5C]	
003C97E0	FF76 18	PUSH DWORD PTR DS:[ESI+18]	
003C97E3	3350 4C	XOR EDX,DWORD PTR DS:[EAX+4C]	
003C97E6	FF76 14	PUSH DWORD PTR DS:[ESI+14]	
003C97E9	3310	XOR EDX,DWORD PTR DS:[EAX]	
003C97EB	FF76 10	PUSH DWORD PTR DS:[ESI+10]	
003C97EE	2BCA	SUB ECX,EDX	
003C97F0	FFD1	CALL ECX	
003C97F2	EB 1C	JNZ SHORT 003C9810	
003C97F4	83FA 01	CMPL EDX,1	
003C97F7	75 1A	JNZ SHORT 003C9813	
003C97F9	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
003C97FC	8B50 5C	MOV EDX,DWORD PTR DS:[EAX+5C]	
003C97FF	3350 4C	XOR EDX,DWORD PTR DS:[EAX+4C]	
003C9802	FF76 08	PUSH DWORD PTR DS:[ESI+8]	
003C9805	3310	XOR EDX,DWORD PTR DS:[EAX]	
003C9807	6A 00	PUSH 0	
003C9809	FF76 0C	PUSH DWORD PTR DS:[ESI+C]	
003C980C	2BCA	SUB ECX,EDX	
003C980E	FFD1	CALL ECX	
003C9810	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
003C9813	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
003C9816	5F	POP EDI	
003C9817	5E	POP ESI	
003C9818	C9	LEAVE	
003C9819	C3	RET	
003C981A	837C24 08 01	CMPL DWORD PTR SS:[ESP+8],1	
003C981F	75 14	JNZ SHORT 003C9835	
003C9821	68 A82B3E00	PUSH 3E2BA8	
003C9826	FF15 78E23C00	CALL DWORD PTR DS:[3CE278]	kernel32.InitializeCriticalSection
003C982C	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]	
003C9830	A3 708F3D00	MOV DWORD PTR DS:[3D8F70],EAX	

_Yeah Yeah! Now, set the breakpoint at:

Address	Hex dump	Disassembly	Comment
003C97D9	85D2	TEST EDX,EDX	
003C97DB	75 17	JNZ SHORT 003C97F4	
003C97DD	8B50 5C	MOV EDX,DWORD PTR DS:[EAX+5C]	
003C97E0	FF76 18	PUSH DWORD PTR DS:[ESI+18]	
003C97E3	3350 4C	XOR EDX,DWORD PTR DS:[EAX+4C]	
003C97E6	FF76 14	PUSH DWORD PTR DS:[ESI+14]	
003C97E9	3310	XOR EDX,DWORD PTR DS:[EAX]	
003C97EB	FF76 10	PUSH DWORD PTR DS:[ESI+10]	
003C97EE	2BCA	SUB ECX,EDX	
003C97F0	FFD1	CALL ECX	
003C97F2	EB 1C	JNZ SHORT 003C9810	
003C97F4	83FA 01	CMPL EDX,1	
003C97F7	75 1A	JNZ SHORT 003C9813	
003C97F9	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
003C97FC	8B50 5C	MOV EDX,DWORD PTR DS:[EAX+5C]	
003C97FF	3350 4C	XOR EDX,DWORD PTR DS:[EAX+4C]	
003C9802	FF76 08	PUSH DWORD PTR DS:[ESI+8]	
003C9805	3310	XOR EDX,DWORD PTR DS:[EAX]	
003C9807	6A 00	PUSH 0	
003C9809	FF76 0C	PUSH DWORD PTR DS:[ESI+C]	
003C980C	2BCA	SUB ECX,EDX	
003C980E	FFD1	CALL ECX	
003C9810	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
003C9813	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
003C9816	5F	POP EDI	
003C9817	5E	POP ESI	
003C9818	C9	LEAVE	
003C9819	C3	RET	
003C981A	837C24 08 01	CMPL DWORD PTR SS:[ESP+8],1	
003C981F	75 14	JNZ SHORT 003C9835	
003C9821	68 A82B3E00	PUSH 3E2BA8	
003C9826	FF15 78E23C00	CALL DWORD PTR DS:[3CE278]	kernel32.InitializeCriticalSection
003C982C	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]	
003C9830	A3 708F3D00	MOV DWORD PTR DS:[3D8F70],EAX	

Set breakpoint o day!

_F9, F8. Congratulation, has to OEP. Khua khua!.

Address	Hex dump	Disassembly	Comment
0040C434	55	PUSH ESP	
0040C435	8BEC	MOV EBP,ESP	
0040C437	6A FF	PUSH -1	
0040C439	68 28334100	PUSH fraps.00413328	
0040C43E	68 30E94000	PUSH fraps.0040E930	
0040C443	64:41 00000000	MOV EAX,DWORD PTR FS:[0]	
0040C449	58	PUSH EAX	
0040C44A	64:8925 000000	MOV DWORD PTR FS:[0],ESP	
0040C451	83EC 58	SUB ESP,58	
0040C454	53	PUSH EBX	
0040C455	56	PUSH ESI	
0040C456	57	PUSH EDI	
0040C457	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0040C45A	FF15 2C3F7001	JMP DWORD PTR DS:[1703F2C]	
0040C460	33D2	XOR EDX,EDX	
0040C462	8AD4	MOV DL,AH	
0040C464	8915 9C30DC00	MOV DWORD PTR DS:[DC309C],EDX	
0040C46A	8BC8	MOV ECX,EAX	
0040C46C	81E1 FF000000	AND ECX,0FF	
0040C472	8900 9830DC00	MOV DWORD PTR DS:[DC3098],ECX	
0040C478	C1E1 08	SHL ECX,8	
0040C47B	83CA	ADD ECX,EDX	
0040C47D	8900 9430DC00	MOV DWORD PTR DS:[DC3094],ECX	
0040C483	C1E8 10	SHR EAX,10	
0040C486	A3 9830DC00	MOV DWORD PTR DS:[DC3090],EAX	
0040C48B	33F6	XOR ESI,ESI	
0040C48D	56	PUSH ESI	
0040C48E	E8 47230000	JMP fraps.0040E7DA	
0040C493	59	POP ECX	
0040C494	85C0	TEST EAX,EAX	
0040C496	75 08	JNZ SHORT fraps.0040C4A0	
0040C498	6A 1C	PUSH 1C	
0040C49A	E8 B0000000	JMP fraps.0040C54F	
0040C49F	59	POP ECX	

_Lord PE, dump Full, done!

c:\fraps\fraps.exe	dump full...	004000
i:\cracker\utilities\lordpe	dump partial...	0400C
	dump region...	
Path		imageS
c:\fraps\fraps.exe	active dump engine	0C04C
c:\windows\system32\nt	priority	00A7C
c:\windows\system32\k	correct ImageSize	00E6C
c:\windows\system32\u	load into PE editor... (temp file)	008CC
c:\windows\system32\g	load into PE editor... (read only)	0040C
c:\windows\system32\a		008D0
c:\windows\system32\vp	burn process	0086C
c:\windows\system32\in		001CC
c:\windows\system32\lc	refresh	0008C

c:\program files\techsmith\snagit 7\tschelp.exe	00000308	00400000	0000A000
c:\program files\unikey\unikey.exe	000007C8	00400000	0002F000
i:\cracker\debug- disassembler\odbg110_org...	00000874	00400000	00164000
c:\program files\internet explorer\iexplore.exe	00000A80	00400000	00019000
c:\fraps\fraps.exe	00E94	00400000	00C04000
i:\cracker\utilities\lordpe	00F28	00400000	00036000
Path		eBase	ImageSize
c:\fraps\fraps.exe		0000	00C04000
c:\windows\system32\nt		0000	000A7000
c:\windows\system32\k		0000	000E6000
c:\windows\system32\u		0000	0008C000

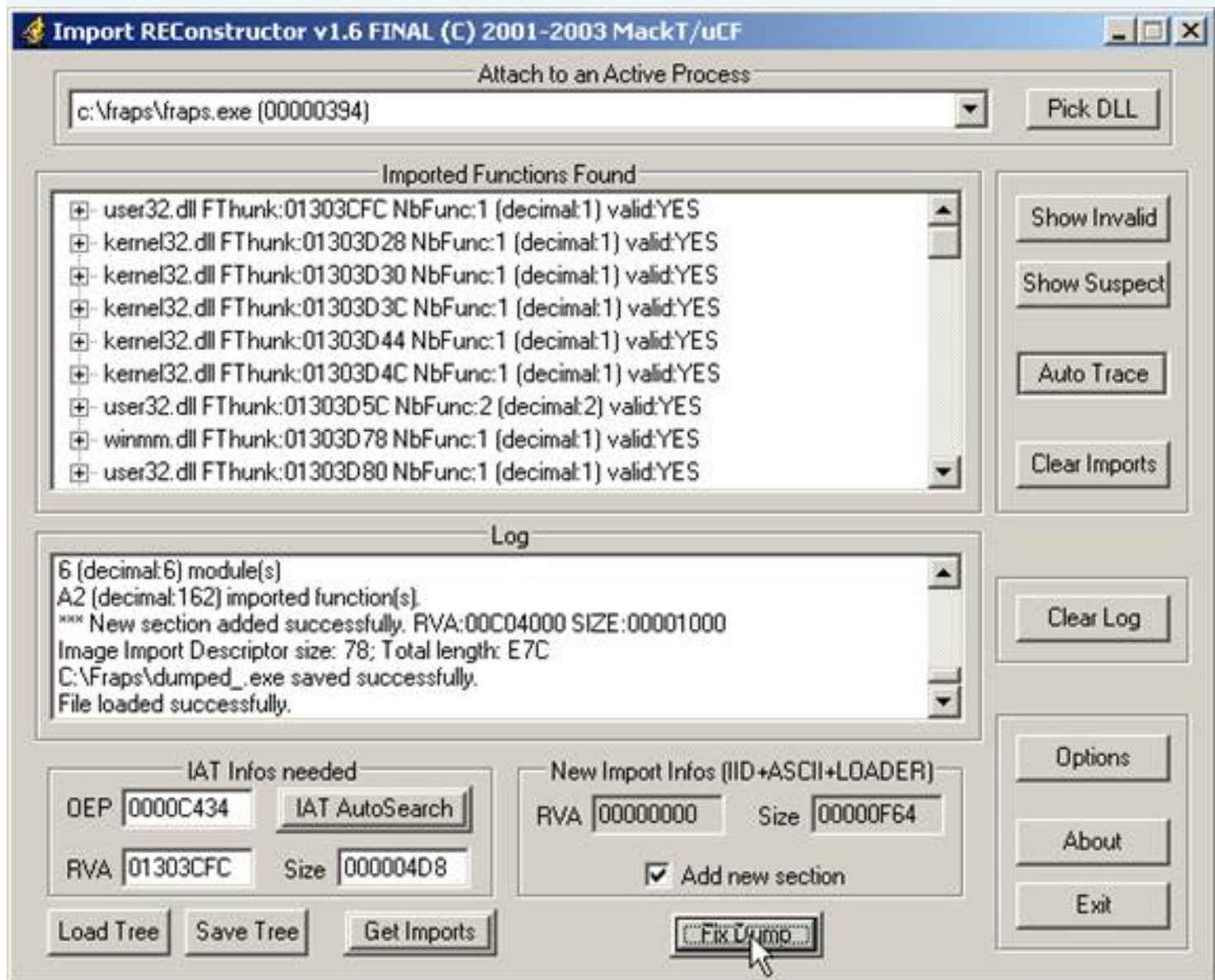
_IAT:

IAT Start: 01703584 77D49724 USER32.LoadIconA

IAT End: 01703A5C 77E7F02E kernel32.SetFilePointer

Len: 4D8

_Show Invalid, Fix dump! Run!



IV. Conclusion

_Hy Expectations you satisfied with this tut! See Series # 7. Is probably not have this series, but because you have not found free of this type should I always do! Bye!

Mỗi got legs throughout the holiday heaven

Blue eyes see some of the same sea

Places where only see swordsman

Know where to find us a dance

Make Ma General structure Loses

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtlm, hosiminh, Nilrem, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RiF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx OllyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: Tien Giang 16/09/2005)

Armadillo collect sand-stone

Series # 6 Exp: Armadillo 4.xx-Code Splicing

I. Intro

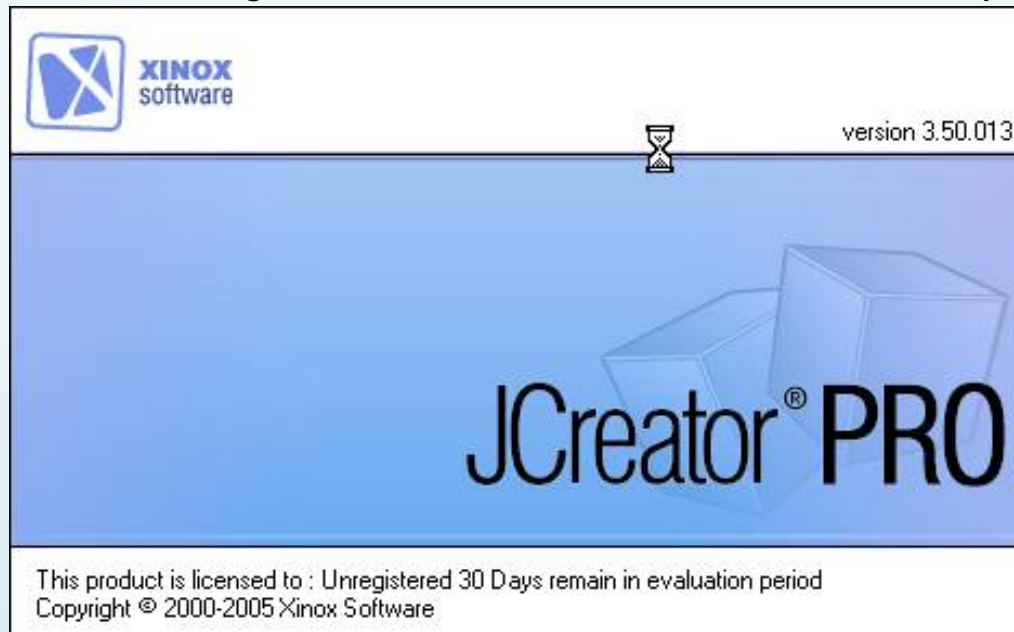
Nry _Phan I will guide you in a different format unpack Code Splicing! I read in the topic have lr number you only ask questions you have not mr NRO post len ge mr you the knowledge through a series of tut nry. You expect the contribution of bri clause written by each person in a while.

II. Tools

- 1.OllyDBG - The best config debugger for ArmMUP by hacnho.
- 2.LordPE 1.4 Deluxe
- 3.Import REConstructor 1.6 Final
- 4.API Address 1.0

III. Unpacking

Target # 1: JCreator Pro TM 3:50 - 3.xx Code Splicing (anti-dump)!



_Load Target:

008E5000	60	PUSHAD	
008E5001	E8 00000000	CALL JCreator.008E5006	
008E5006	5D	POP EBP	
008E5007	50	PUSH EAX	
008E5008	51	PUSH ECX	
008E5009	0FCA	BSWAP EDX	
008E500B	F7D2	NOT EDX	
008E500D	9C	PUSHAD	
008E500E	F7D2	NOT EDX	
008E5010	0FCA	BSWAP EDX	
008E5012	EB 0F	JMP SHORT JCreator.008E5023	
008E5014	B9 F80F00F8	MOV ECX, F80F00F8	kernel32.77E814C7

_Set HE GetModuleHandleA, F9:



_Shift + F9 pass by:

77E7AD86	837C24 04 00	CMP DWORD PTR SS:[ESP+4],0	
77E7AD88	0F84 37010000	JE kernel32.77E7AEC8	
77E7AD91	FF7424 04	PUSH DWORD PTR SS:[ESP+4]	
77E7AD95	E8 F8050000	CALL kernel32.77E7B392	
77E7AD9A	85C0	TEST EAX,EAX	ADVAPI32.77DE65F7
77E7AD9C	74 08	JE SHORT kernel32.77E7ADA6	
77E7AD9E	FF70 04	PUSH DWORD PTR DS:[EAX+4]	
77E7ADA1	E8 27060000	CALL kernel32.GetModuleHandleW	
77E7ADA6	C2 0400	RETN 4	
77E7ADA9	55	PUSH EBP	
77E7ADAA	8BEC	MOV EBP,ESP	
77E7ADAC	83EC 10	SUB ESP,10	
77E7ADAF	57	PUSH EDI	
77E7ADB0	64 01 18000000	MOV EAX,DWORD PTR FS:[18]	

00127A6C	003D51E0	CALL to GetModuleHandleA from 003D51DA
00127A70	003E8BAC	pModule = "kernel32.dll"
00127A74	003E9CC4	ASCII "VirtualAlloc"
00127A78	003EC8D8	
00127A7C	77F75690	ntdll.RtlLeaveCriticalSection
00127A80	00000000	
00127A84	00000000	
00127A88	00000000	
00127A8C	00000000	
00127A90	00000000	

_Shift + F9 times 2:

00127A6C	003D51F0	CALL to GetModuleHandleA from 003D51F7
00127A70	003E8BAC	pModule = "kernel32.dll"
00127A74	003E9CB8	ASCII "VirtualFree"
00127A78	003EC8D8	
00127A7C	77F75690	ntdll.RtlLeaveCriticalSection
00127A80	00000000	
00127A84	00000000	
00127A88	00000000	
00127A8C	00000000	
00127A90	00000000	

_Shift + F9 times 3:

001277D0	003C4E69	CALL to GetModuleHandleA from 003C4E63
001277D4	00127920	pModule = "kernel32.dll"
001277D8	0012CC3C	
001277DC	BDCE3161	
001277E0	00000000	
001277E4	003E809C	
001277E8	CD00004A	
001277EC	00000000	
001277F0	80000000	
001277F4	00003FFF	

_Atl + F9:

003C4E69	8B0D AC0D3F00	MOV ECX,DWORD PTR DS:[3E800C]	
003C4E6F	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	kernel32.77E60000
003C4E72	A1 AC0D3F00	MOV EAX,DWORD PTR DS:[3F0DAC]	
003C4E77	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
003C4E7A	75 16	JNZ SHORT 003C4E92	
003C4E7C	8D85 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
003C4E82	50	PUSH EAX	kernel32.77E60000
003C4E83	FF15 B4323E00	CALL DWORD PTR DS:[3E32B4]	kernel32.LoadLibraryA
003C4E89	8B0D AC0D3F00	MOV ECX,DWORD PTR DS:[3F0DAC]	
003C4E8F	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	kernel32.77E60000
003C4E92	A1 AC0D3F00	MOV EAX,DWORD PTR DS:[3F0DAC]	
003C4E97	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
003C4E9A	0F84 2F010000	JE 003C4FCF	kernel32.77E7B5E1
003C4EA0	33C9	XOR ECX,ECX	
003C4EA2	8B07	MOV EAX,DWORD PTR DS:[EDI]	
003C4EA4	3918	CMP DWORD PTR DS:[EAX],EBX	
003C4EA6	74 06	JE SHORT 003C4EAE	kernel32.77E7B5E1
003C4EA8	41	INC ECX	
003C4EA9	83C0 0C	ADD EAX,0C	kernel32.77E7B5E1
003C4EAC	EB F6	JMP SHORT 003C4EA4	kernel32.77E7B5E1
003C4EAE	8BD9	MOV EBX,ECX	
003C4EB0	C1E3 02	SHL EBX,2	

Magic Jump

_Change JE thrnh JMP!

003C4E69	8B0D AC0D3F00	MOV ECX,DWORD PTR DS:[3F0DAC]	kernel32.77E60000
003C4E6F	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003C4E72	A1 AC0D3F00	MOV EAX,DWORD PTR DS:[3F0DAC]	
003C4E77	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
003C4E7A	75 16	JNZ SHORT 003C4E92	
003C4E7C	8D85 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
003C4E82	50	PUSH EAX	kernel32.77E60000
003C4E83	FF15 B4323E00	CALL DWORD PTR DS:[3E32B4]	kernel32.LoadLibraryA
003C4E89	8B0D AC0D3F00	MOV ECX,DWORD PTR DS:[3F0DAC]	
003C4E8F	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	kernel32.77E60000
003C4E92	A1 AC0D3F00	MOV EAX,DWORD PTR DS:[3F0DAC]	
003C4E97	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
003C4E9A	0F84 2F010000	JE 003C4FCF	
003C4E9F	90	NOOP	
003C4EA0	33C9	XOR ECX,ECX	kernel32.77E7B5E1
003C4EA2	8B07	MOV EAX,DWORD PTR DS:[EDI]	
003C4EA4	3918	CMP DWORD PTR DS:[EAX],EBX	
003C4EA6	74 06	JE SHORT 003C4EAE	

_Xoa Breakpoint: hd GetModuleHandleA

_Set Breakpoint he VirtualAlloc, F9:

77E7AC72	55	PUSH EBP	
77E7AC73	8BEC	MOV EBP,ESP	
77E7AC75	FF75 14	PUSH DWORD PTR SS:[EBP+14]	
77E7AC78	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
77E7AC7B	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
77E7AC7E	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
77E7AC81	6A FF	PUSH -1	
77E7AC83	E8 9CFFFFFF	CALL kernel32.VirtualAllocEx	73421D07
77E7AC88	5D	POP EBP	
77E7AC89	C2 1000	RETN 10	
77E7AC8C	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]	
77E7AC90	85C0	TEST EAX,EAX	

00127084	73421D07	CALL to VirtualAlloc from 73421D05	
00127088	00000000	Address = NULL	
0012708C	00400000	Size = 400000 (4194304.)	
001270C0	00002000	AllocationType = MEM_RESERVE	
001270C4	00000004	Protect = PAGE_READWRITE	
001270C8	00000000		
001270CC	00000001		
001270D0	0011118		
001270D4	73421CCD	RETURN to 73421CCD from 73421CDB	
001270D8	73421BCE	RETURN to 73421BCE from 73421C9C	
001270DC	73421BCE	RETURN to 73421BCE from 73421C9C	

+ F9 _Alt times 1:

73421D07	8BF0	MOV ESI,EAX	
73421D09	85F6	TEST ESI,ESI	
73421D0B	0F84 C5700400	JE 73468DD6	
73421D11	6A 04	PUSH 4	
73421D13	68 00100000	PUSH 1000	
73421D18	68 00000100	PUSH 10000	
73421D1D	56	PUSH ESI	
73421D1E	FFD7	CALL EDI	UNICODE "\0\0\0\0\0\0\0\0"
73421D20	85C0	TEST EAX,EAX	kernel32.VirtualAlloc
73421D22	0F84 A0700400	JE 73468DC8	
73421D28	81FD 30B05273	CMP EBX,7352B030	
73421D2E	0F85 74700400	JNZ 73468DA8	
73421D34	A1 30B05273	MOV EAX,DWORD PTR DS:[7352B030]	

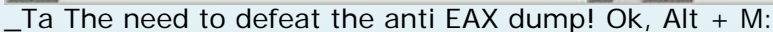
_Shift + F9, Alt + F9 times 2:

73421D20	85C0	TEST EAX,EAX	
73421D22	0F84 A0700400	JE 73468DC8	
73421D28	81FD 30B05273	CMP EBX,7352B030	
73421D2E	0F85 74700400	JNZ 73468DA8	
73421D34	A1 30B05273	MOV EAX,DWORD PTR DS:[7352B030]	
73421D39	85C0	TEST EAX,EAX	
73421D3B	0F84 49700400	JE 73468D8A	
73421D41	A1 34B05273	MOV EAX,DWORD PTR DS:[7352B034]	
73421D46	85C0	TEST EAX,EAX	
73421D48	0F84 4B700400	JE 73468D99	
73421D4E	8D86 00004000	LEA EAX,DWORD PTR DS:[ESI+400000]	
73421D54	8D4D 18	LEA ECX,DWORD PTR SS:[EBP+18]	
73421D57	8D95 98000000	LEA EDX,DWORD PTR SS:[EBP+98]	
73421D5D	8945 14	MOV DWORD PTR SS:[EBP+14],EAX	
73421D60	8B75 10	MOV ESI,DWORD PTR SS:[EBP+10],ESI	

_Shift + F9, Alt + F9 times 3:

003D80D4	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX	
003D80DA	838D 6CD7FFFF	CMP DWORD PTR SS:[EBP-2894],0	
003D80E1	74 64	JE SHORT 003D8147	
003D80E3	6A 40	PUSH 40	
003D80E5	68 00100000	PUSH 1000	
003D80EA	FFB5 64D7FFFF	PUSH DWORD PTR SS:[EBP-289C]	
003D80F0	FF35 F06C3F00	PUSH DWORD PTR DS:[3F6CF0]	
003D80F6	FF15 88313E00	CALL DWORD PTR DS:[3E3188]	kernel32.VirtualAlloc
003D80FC	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX	
003D8102	838D 6CD7FFFF	CMP DWORD PTR SS:[EBP-2894],0	
003D8109	74 3C	JE SHORT 003D8147	
003D810B	8B85 6CD7FFFF	MOV EAX,DWORD PTR SS:[EBP-2894]	
003D8111	A3 F06C3F00	MOV DWORD PTR DS:[3F6CF0],EAX	
003D8116	8B85 6CD7FFFF	MOV EAX,DWORD PTR SS:[EBP-2894]	
003D811C	3B85 0CD8FFFF	CMP EAX,DWORD PTR SS:[EBP-27F4]	JCreator.00400000
003D8122	76 23	JBE SHORT 003D8147	
003D8124	8B85 0CD8FFFF	MOV EAX,DWORD PTR SS:[EBP-27F4]	JCreator.00400000
003D812A	0385 7CD9FFFF	ADD EAX,DWORD PTR SS:[EBP-2684]	JCreator.00969000
003D8130	3985 6CD7FFFF	CMP DWORD PTR SS:[EBP-2894],EAX	
003D8136	73 0F	JNB SHORT 003D8147	
003D8138	8B85 64D7FFFF	MOV EAX,DWORD PTR SS:[EBP-289C]	
003D813E	C1E8 02	SHR EAX,2	
003D8141	8985 64D7FFFF	MOV DWORD PTR SS:[EBP-289C],EAX	
003D8147	E9 FBFFFFFF	JMP 003D8047	
003D814C	E8 F75D0000	CALL 003D0F48	
003D8151	85C0	TEST EAX,EAX	
003D8153	74 0E	JE SHORT 003D8163	
003D8155	8B85 64D7FFFF	MOV EAX,DWORD PTR SS:[EBP-289C]	
003D815B	D1E8	SHR EAX,1	
003D815D	8985 64D7FFFF	MOV DWORD PTR SS:[EBP-289C],EAX	
003D8163	6A 01	PUSH 1	
003D8165	58	POP EAX	003EC8D8
003D8166	85C0	TEST EAX,EAX	

F8 _Trace down to this:



__Ta Will change in the EAX thrnh address ADATA section. Back to the CPU, FPU Double click vro change:

Modify EAX

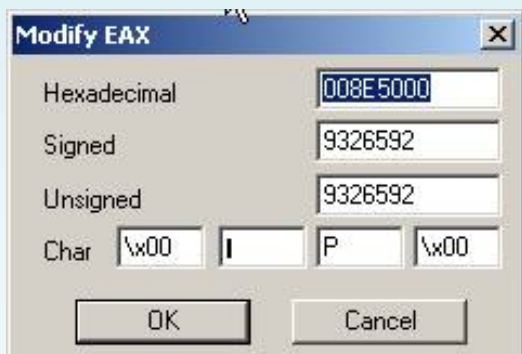
Hexadecimal: 031E0000

Signed: 52297728

Unsigned: 52297728

Char: ☐ \x03 ☐ \x1E ☐ \x00 ☐ \x00

OK Cancel



```
Registers (FPU)
EAX 008E5000 JCreator.<ModuleEntryPoint>
ECX 77E7AC6F kernel32.77E7AC6F
EDX 7FFE0304
EBX 00000000
ESP 00127A78
EBP 0012CD94
ESI BDCE3161
EDI 0012CC3C
EIP 003D80FC

C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty +UNORM 17B2 77F51778 77F517E6
ST1 empty -UNORM CD9C 00000000 0057E618
ST2 empty +UNORM 0003 0012CDD0 0012DAC9
ST3 empty +UNORM 0020 00000020 0012CDD0
ST4 empty -UNORM DAF1 004A6C1C 0012CDC4
ST5 empty -UNORM CDF8 004B1AC4 00000003
ST6 empty 1.000000000000000000000000
ST7 empty 1.000000000000000000000000

3 2 1 0 ES P U O Z D I
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1
```

_Hd VirtualAlloc, Alt + M, set breakpoint on access at 401,000 sections of text, F9, OEP:

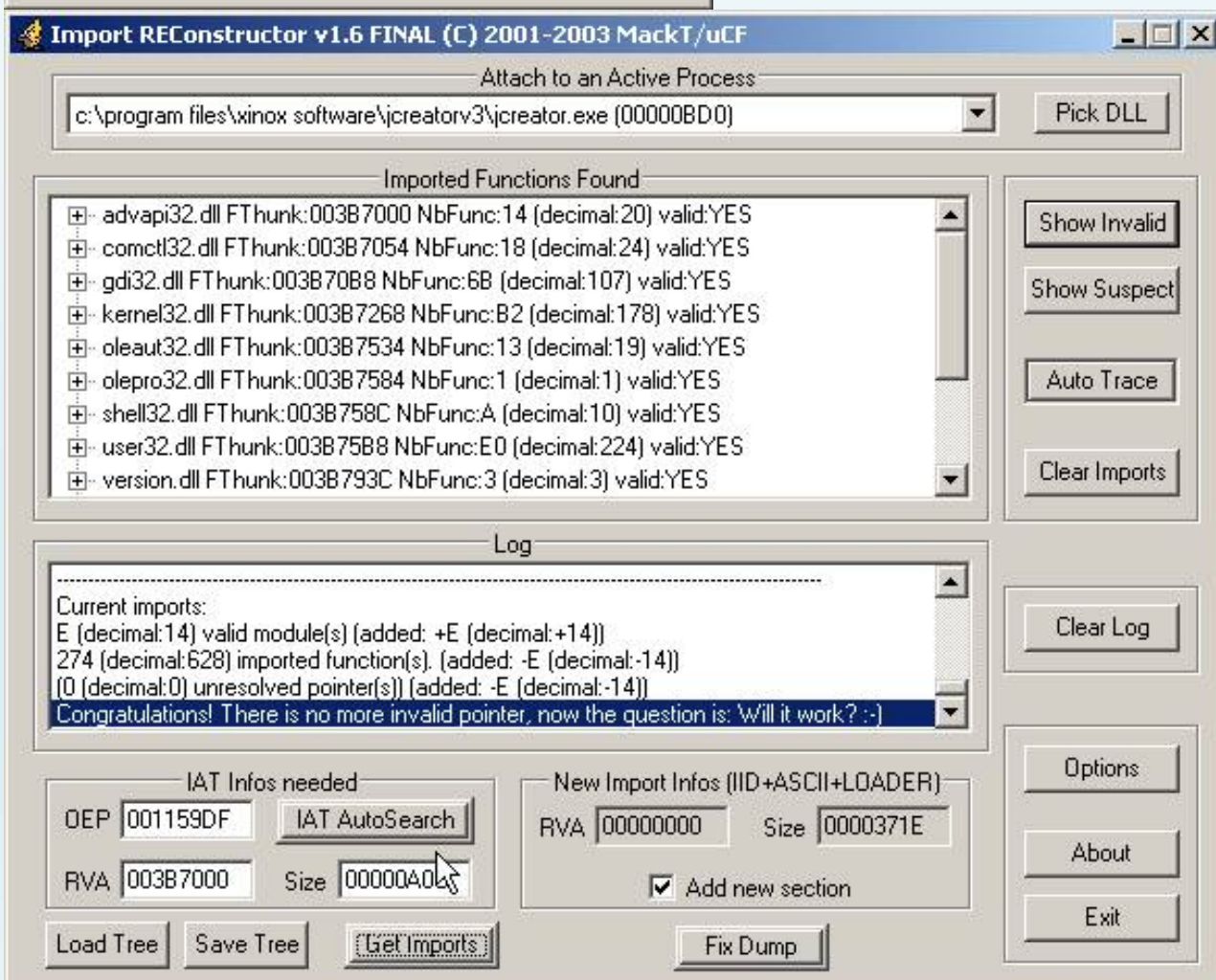
005159DF	55	PUSH EBP	<- OEP
005159E0	8BEC	MOV EBP,SP	
005159E2	6A FF	PUSH -1	
005159E4	68 C0B57D00	PUSH JCreator.007DB5C0	
005159E9	68 24A85100	PUSH JCreator.0051A824	
005159EE	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
005159F4	50	PUSH EAX	JCreator.008F5370
005159F5	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
005159FC	83EC 58	SUB ESP,58	
005159FF	53	PUSH EBX	
00515A00	56	PUSH ESI	JCreator.008FBE08
00515A01	57	PUSH EDI	
00515A02	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00515A05	FF15 30747B00	CALL DWORD PTR DS:[7B7430]	kernel32.GetVersion
00515A08	33D2	XOR EDX,EDX	
00515A0D	8A04	MOV DL,AH	
00515A0F	8915 BCDB8800	MOV DWORD PTR DS:[88DB8C],EDX	
00515A15	8BC8	MOV ECX,EAX	JCreator.008F5370
00515A17	81E1 FF000000	AND ECX,0FF	
00515A1D	890D B8DB8800	MOV DWORD PTR DS:[88DB88],ECX	JCreator.005159DF

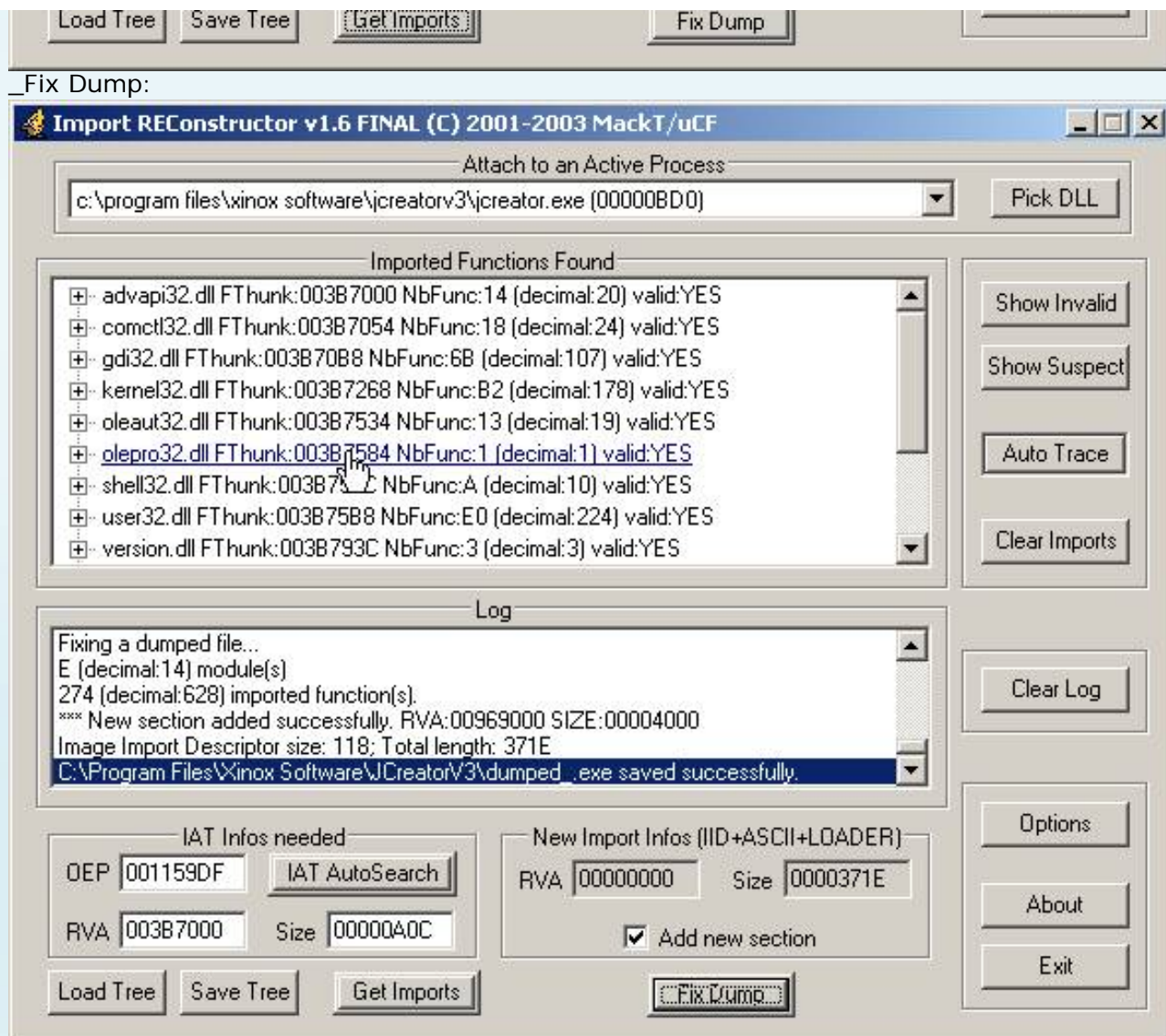
_LordPE DumpFull:

Path	PID	ImageBase	ImageSize
i:\project delphi7\my project\crackerutils\cra...	00000310	00400000	000F4000
c:\windows\system32\ctfmon.exe	00000A54	00400000	00006000
c:\windows\system32\svchost.exe	00000A6C	01000000	00006000
c:\program files\common files\microsoft share...	00000400	00400000	0004D000
c:\program files\winamp\winamp.exe	00000EF4	00400000	00121000
c:\program files\internet explorer\iexplore.exe	00000C80	00400000	00019000
d:\soft need\unikey 3.5\unikey.exe	00000498	00400000	0002F000
c:\program files\microsoft office\office11\win...	000004E8	30000000	008AA000
c:\program files\techsmith\snagit 7\snagit32....	00000D20	00400000	00286000
c:\program files\techsmith\snagit 7\tschelp.exe	00000E18	00400000	0000A000
i:\cracker\debug- disassembler\odba110_ora...	00000F70	00400000	00169000
i:\cracker\utilities	0	00400000	00969000
	C	00400000	00036000

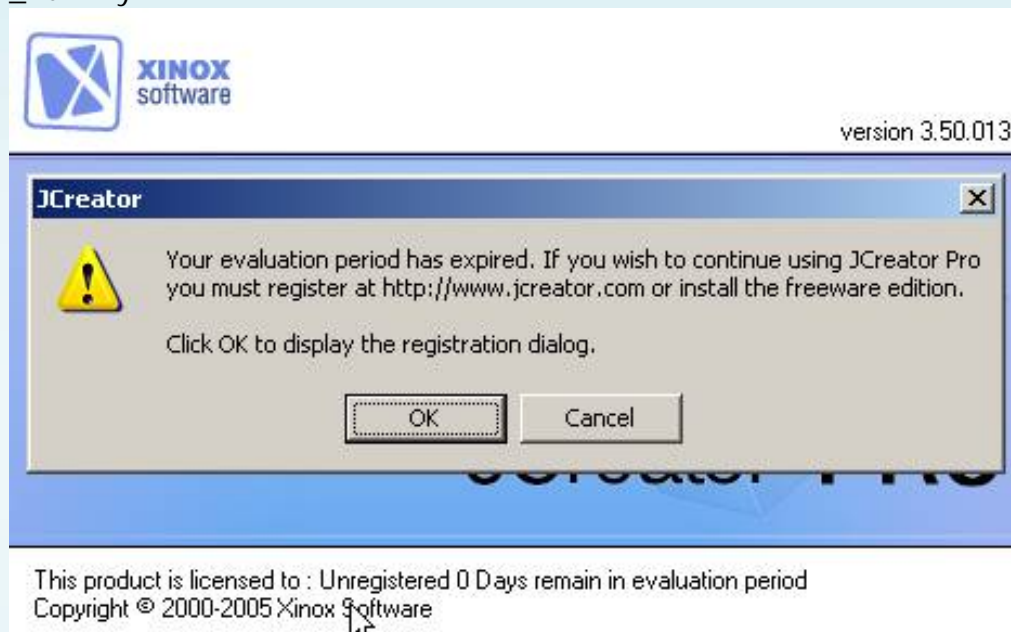
Path	ImageSize
c:\program files\...	00969000
c:\windows\sys...	000A7000

ImpREC. Enter OEP, IAT Auto Search, Get Import, Show Invalid, thanks Cuts:



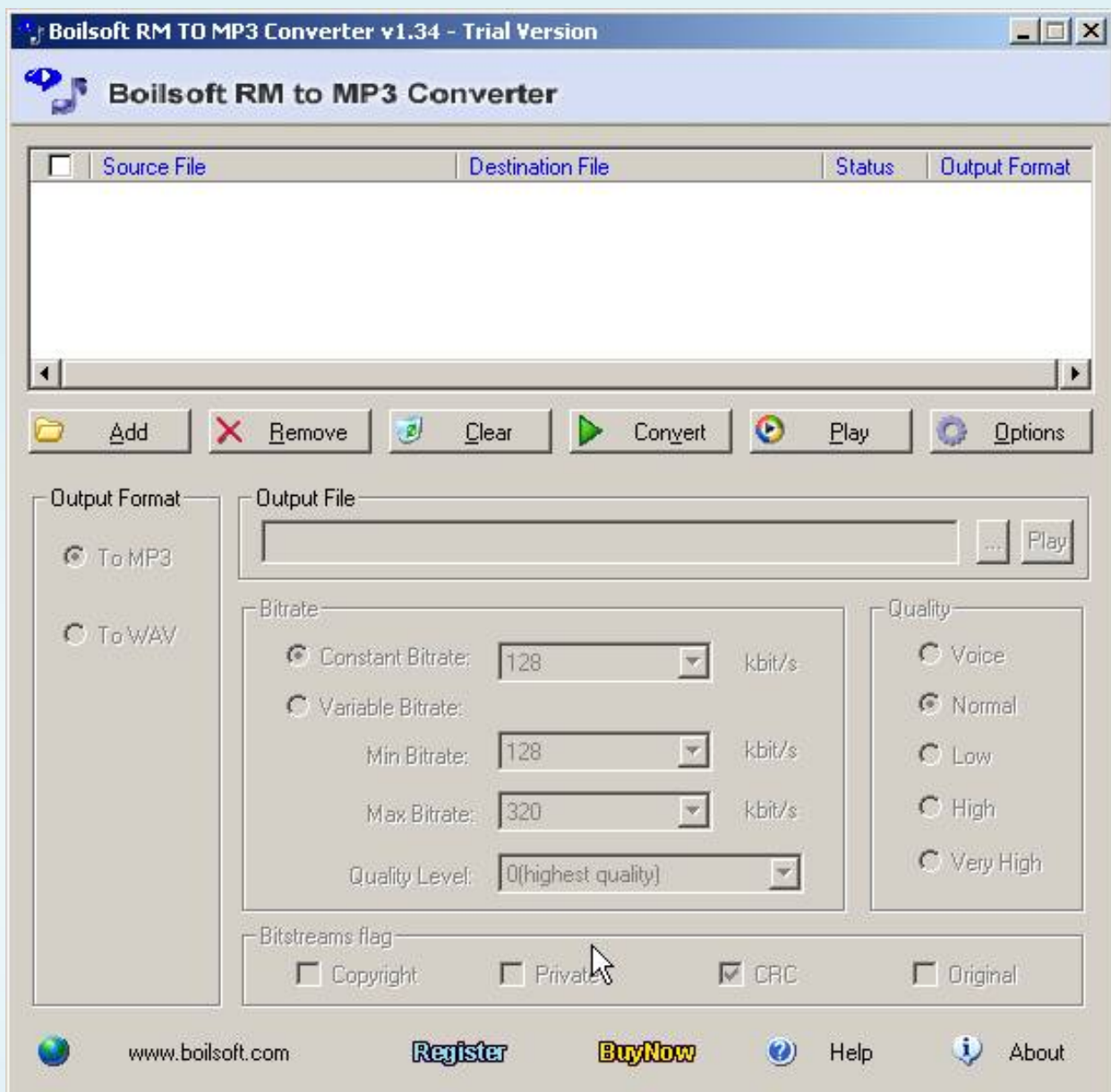


_Run Try:



_Unpack Done!

Target # 2: RM to MP3 Converter v1.32 - 3.xx Code Splicing (anti-dump)!



_Load Target:

00496000	60	PUSH EDI	
00496001	E8 00000000	CALL RMTOMP3.00496006	
00496006	50	POP EBP	
00496007	50	PUSH EAX	
00496008	51	PUSH ECX	
00496009	0FCA	BSWAP EDX	
0049600B	F7D2	NOT EDX	
0049600D	9C	PUSH EDI	
0049600E	F7D2	NOT EDX	
00496010	0FCA	BSWAP EDX	

_he GetModuleHandleA, Shift + F9 3 times:

77E7AD86	837C24 04 00	CMOVDWORD PTR SS:[ESP+4],0	
77E7AD8B	0F84 37010000	JE kernel32.77E7AEC8	
77E7AD91	FF7424 04	PUSH DWORD PTR SS:[ESP+4]	
77E7AD95	E8 F8050000	CALL kernel32.77E7B392	
77E7AD9A	85C0	TEST EAX,EAX	
77E7AD9C	74 08	JE SHORT kernel32.77E7ADA6	
77E7AD9E	FF70 04	PUSH DWORD PTR DS:[EAX+4]	
77E7ADA1	E8 27060000	CALL kernel32.GetModuleHandleW	
77E7ADA6	C2 0400	RETN 4	
77E7ADA9	55	PUSH EBP	

001278E0	000A9B49	CALL to GetModuleHandleA from 000A9B49	
001278E4	00127A24	pModule = "kernel32.dll"	
001278E8	00000000		
001278EC	CE300000		
001278F0	417B0012		
001278F4	003D57C4		
001278F8	00000000		
001278FC	80000000		
00127900	00003FFF		
00127904	00000000		

F8 _Nhan trace through RETN4 hrm:

003A9B49	8B00 74B73D00	MOV ECX, DWORD PTR DS:[ESI+ECX], EAX	
003A9B4F	89040E	MOV DWORD PTR DS:[ESI+ECX], EAX	kernel32.77E60000
003A9B52	A1 74B73D00	MOV EAX, DWORD PTR DS:[30B7741]	
003A9B57	391C06	CMP DWORD PTR DS:[ESI+EAX], EBX	
003A9B5A	75 16	JNZ SHORT 003A9B72	
003A9B5C	8D85 B4FEFFFF	LEA EAX, DWORD PTR SS:[EBP-14C]	
003A9B62	50	PUSH EAX	kernel32.77E60000
003A9B63	FF15 DC003D00	CALL DWORD PTR DS:[3D00DC]	kernel32.LoadLibraryA
003A9B69	8B00 74B73D00	MOV ECX, DWORD PTR DS:[30B7741]	
003A9B6F	89040E	MOV DWORD PTR DS:[ESI+ECX], EAX	kernel32.77E60000
003A9B72	A1 74B73D00	MOV EAX, DWORD PTR DS:[30B7741]	
003A9B77	391C06	CMP DWORD PTR DS:[ESI+EAX], EBX	
003A9B7A	0F84 32010000	JE 003A9CB2	
003A9B80	33C9	XOR ECX, ECX	kernel32.77E7B5E1
003A9B82	8B07	MOV EAX, DWORD PTR DS:[EDI]	
003A9B84	3918	CMP DWORD PTR DS:[EAX], EBX	
003A9B86	74 06	JE SHORT 003A9B8E	
003A9B88	41	INC ECX	kernel32.77E7B5E1
003A9B89	83C0 0C	ADD EAX, 0C	

Patch Thrnh JMP:

003A9B62	50	PUSH EAX	kernel32.77E60000
003A9B63	FF15 DC003D00	CALL DWORD PTR DS:[3D00DC]	kernel32.LoadLibraryA
003A9B69	8B00 74B73D00	MOV ECX, DWORD PTR DS:[30B7741]	
003A9B6F	89040E	MOV DWORD PTR DS:[ESI+ECX], EAX	kernel32.77E60000
003A9B72	A1 74B73D00	MOV EAX, DWORD PTR DS:[30B7741]	
003A9B77	391C06	CMP DWORD PTR DS:[ESI+EAX], EBX	
003A9B7A	E9 33010000	JMP 003A9CB2	
003A9B7E	5B	POP EBX	
003A9B80	33C9	XOR ECX, ECX	kernel32.77E7B5E1
003A9B82	8B07	MOV EAX, DWORD PTR DS:[EDI]	
003A9B84	3918	CMP DWORD PTR DS:[EAX], EBX	
003A9B86	74 06	JE SHORT 003A9B8E	
003A9B88	41	INC ECX	kernel32.77E7B5E1
003A9B89	83C0 0C	ADD EAX, 0C	
003A9B8C	EB F6	JMP SHORT 003A9B84	
003A9B8E	8B09	MOV EBX, ECX	kernel32.77E7B5E1
003A9B90	C1E3 02	SHL EBX, 2	

_hd GetModuleHandleA, he VirtualAlloc:

003C4C25	8985 60C6FFFF	MOV DWORD PTR SS:[EBP-39A0], EAX	
003C4C2B	83B0 60C6FFFF	CMP DWORD PTR SS:[EBP-39A0], 0	
003C4C32	74 33	JE SHORT 003C4C67	
003C4C34	6A 40	PUSH 40	
003C4C36	68 00100000	PUSH 1000	
003C4C38	FFB5 58C6FFFF	PUSH DWORD PTR SS:[EBP-39A8]	
003C4C41	FF35 D0013E00	PUSH DWORD PTR DS:[3E01D0]	
003C4C47	FF15 B0013D00	CALL DWORD PTR DS:[3D01B0]	kernel32.VirtualAlloc
003C4C4D	8985 60C6FFFF	MOV DWORD PTR SS:[EBP-39A0], EAX	
003C4C53	83B0 60C6FFFF	CMP DWORD PTR SS:[EBP-39A0], 0	
003C4C5A	74 0B	JE SHORT 003C4C67	
003C4C5C	8B85 60C6FFFF	MOV EAX, DWORD PTR SS:[EBP-39A0]	
003C4C62	A3 D0013E00	MOV DWORD PTR DS:[3E01D0], EAX	
003C4C67	E9 2CFFFFFF	JMP 003C4B98	
003C4C6C	E8 AASD0000	CALL 003CA1B	
003C4C71	85C0	TEST EAX, EAX	
003C4C73	74 0E	JE SHORT 003C4C83	
003C4C75	8B85 58C6FFFF	MOV EAX, DWORD PTR SS:[EBP-39A8]	
003C4C7B	D1E8	SHR EAX, 1	
003C4C7D	8985 58C6FFFF	MOV DWORD PTR SS:[EBP-39A8], EAX	

_Lrm Similar:

003C4C34	6A 40	PUSH 40	
003C4C36	68 00100000	PUSH 1000	
003C4C38	FFB5 58C6FFFF	PUSH DWORD PTR SS:[EBP-39A8]	
003C4C41	FF35 D0013E00	PUSH DWORD PTR DS:[3E01D0]	
003C4C47	FF15 B0013D00	CALL DWORD PTR DS:[3D01B0]	kernel32.VirtualAlloc
003C4C4D	8985 60C6FFFF	MOV DWORD PTR SS:[EBP-39A0], EAX	
003C4C53	83B0 60C6FFFF	CMP DWORD PTR SS:[EBP-39A0], 0	
003C4C5A	74 0B	JE SHORT 003C4C67	
003C4C5C	8B85 60C6FFFF	MOV EAX, DWORD PTR SS:[EBP-39A0]	
003C4C62	A3 D0013E00	MOV DWORD PTR DS:[3E01D0], EAX	
003C4C67	E9 2CFFFFFF	JMP 003C4B98	
003C4C6C	E8 AASD0000	CALL 003CA1B	
003C4C71	85C0	TEST EAX, EAX	
003C4C73	74 0E	JE SHORT 003C4C83	
003C4C75	8B85 58C6FFFF	MOV EAX, DWORD PTR SS:[EBP-39A8]	
003C4C7B	D1E8	SHR EAX, 1	
003C4C7D	8985 58C6FFFF	MOV DWORD PTR SS:[EBP-39A8], EAX	
003C4C83	6A 01	PUSH 1	
003C4C85	50	PUSH EAX	
003C4C87	0F84 32010000	JE 003C4D11	
003C4C89	81B0 58C6FFFF	CMPL DWORD PTR SS:[EBP-39B0], 0FFFF	
003C4C8B	73 0C	JB SHORT 003C4C95	
003C4C8D	C785 78ADFFFF	MOV EAX, DWORD PTR SS:[EBP+FFFFA078], 0FFFF	
003C4C91	E9 2CFFFFFF	JMP 003C4B98	
003C4C93	8B85 58C6FFFF	MOV EAX, DWORD PTR SS:[EBP-39A8]	
003C4C95	8B85 78ADFFFF	MOV EAX, DWORD PTR SS:[EBP+FFFFA078], EAX	
003C4C97	8B85 78ADFFFF	MOV EAX, DWORD PTR SS:[EBP+FFFFA078]	
003C4C99	8B85 2C66FFFF	MOV EAX, DWORD PTR SS:[EBP-39D4], EAX	
003C4C9B	6A 00	PUSH 0	
003C4C9D	FFB5 58C6FFFF	PUSH DWORD PTR SS:[EBP-39A8]	
003C4CA1	8B85 2C66FFFF	MOV EAX, DWORD PTR SS:[EBP-39D4]	
003C4CA3	50	PUSH EAX	
003C4CA5	FFB5 58C6FFFF	PUSH DWORD PTR SS:[EBP-39A8]	

_hd VirtualAlloc, Alt + M, set breakpoint. F9:

00390000	00003000				Map	R	R	\Device\HarddiskVol
003A0000	00049000				Priv	RW		
003F0000	0000C000				Priv	RW		
00400000	00001000	RMTOMP3	PE header		Image	R	RWE	
00401000	0004B000	RMTOMP3	.text			R	RWE	
0044C000	00010000	RMTOMP3	.rdata					
0045C000	0000A000	RMTOMP3	.data					
00466000	00030000	RMTOMP3	.text1	code				
00496000	00010000	RMTOMP3	.adata	code				
004A6000	00020000	RMTOMP3	.data1	data, impo				
004C6000	00070000	RMTOMP3	.pdata					
00536000	00011000	RMTOMP3	.rsrc	resources				
00550000	00008000							
00610000	00002000							
00620000	00103000							
00730000	00164000							
00A30000	00001000							
00B30000	0000C000							
00B40000	00002000							
00B50000	00018000							

- Actualize
- Dump in CPU
- Dump
- Search Ctrl+B
- Set break-on-access F2
- Set memory breakpoint on access
- Set memory breakpoint on write

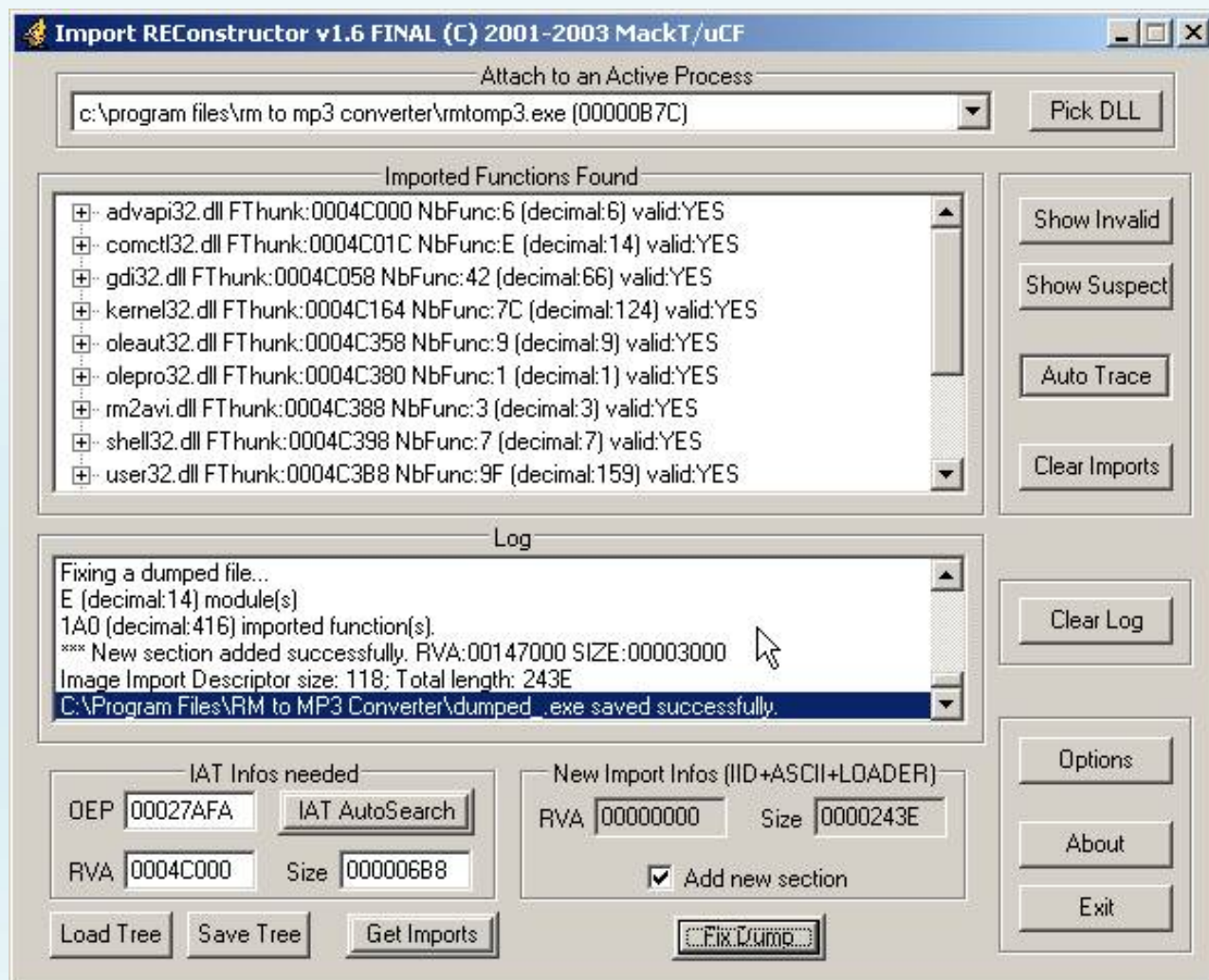
OEP:

00427AFA	55	PUSH EBP	
00427AF8	8BEC	MOV EBP,ESP	
00427AFD	6A FF	PUSH -1	
00427AFF	68 881B4500	PUSH RMTOMP3.00451B88	
00427B04	68 A8AF4200	PUSH RMTOMP3.0042AF8	
00427B09	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00427B0F	50	PUSH EAX	RMTOMP3.004A6310
00427B10	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00427B17	83EC 58	SUB ESP,58	
00427B1A	53	PUSH EBX	
00427B1B	56	PUSH ESI	RMTOMP3.004AC568
00427B1C	57	PUSH EDI	RMTOMP3.00427AFA
00427B1D	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00427B20	FF15 C4C14400	CALL DWORD PTR DS:[44C1C4]	kernel32.GetVersion
00427B26	3302	XOR EDX,EDX	

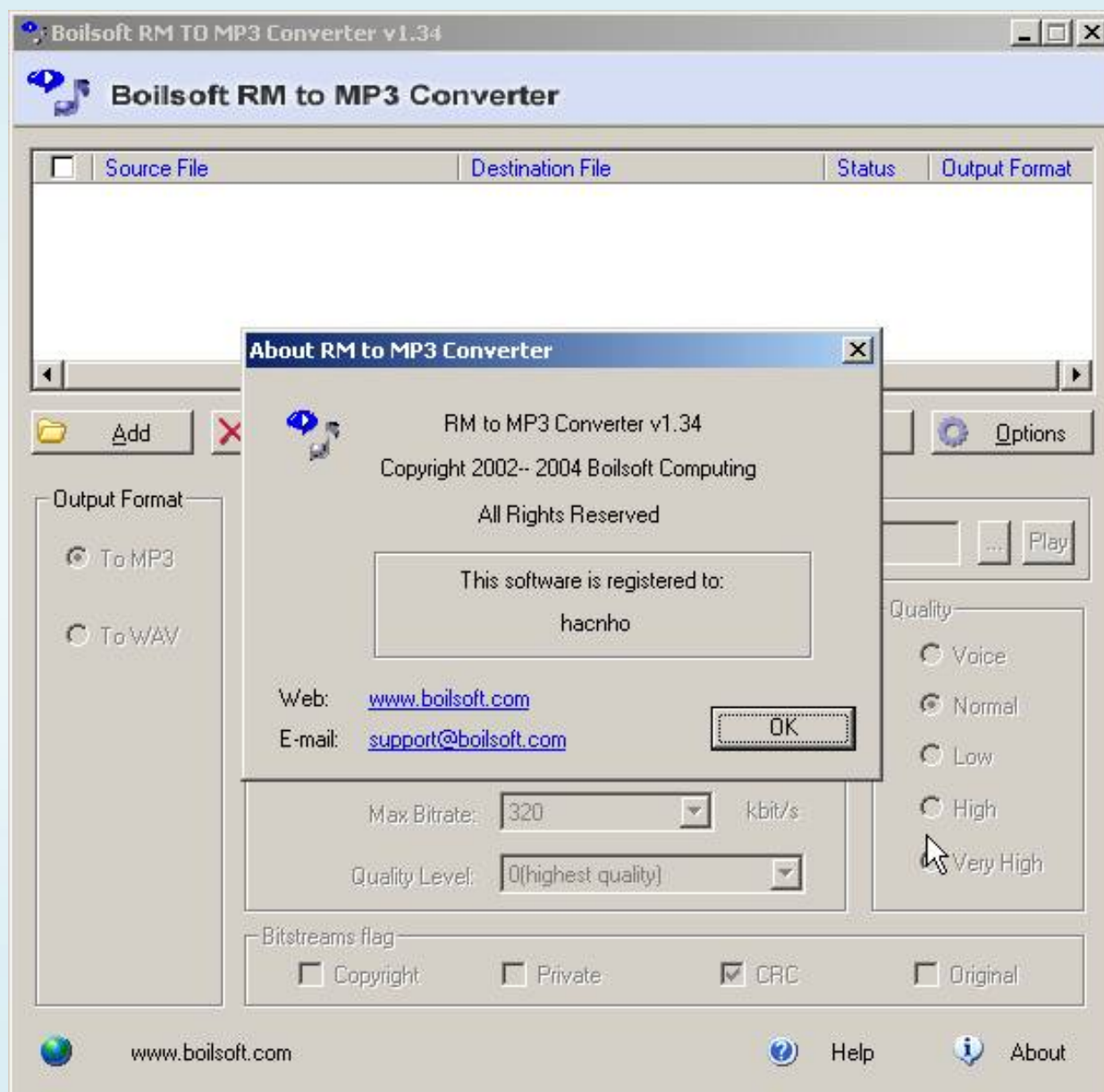
Full _Dump:

c:\program files\winamp\winamp.exe	00000EF4	00400000	00121000
d:\soft need\unikey 3.5\unikey.exe	00000498	00400000	0002F000
c:\program files\techsmith\snagit 7\snagit32....	00000D20	00400000	00286000
c:\program files\techsmith\snagit 7\tschelp.exe	00000E18	00400000	0000A000
i:\cracker\debug- disassembler\odbg110_org...	00000C4C	00400000	00169000
c:\program files\microsoft office\office11\win...	00000CCC	30000000	00BA4000
c:\program files\rm		00400000	00147000
i:\cracker\utilities\l		00400000	00036000
Path		ImageSize	
c:\program files\rm	active dump engine	00147000	
c:\windows\system	priority	000A7000	

_ImpREC:



_Run Try:



_Unpack Done!

IV. Conclusion

After having _Hen tut! Bye ...

_Tang Of the property of ngry month old henh plate. I remember cnn or they forgot!

EeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtlm, hosiminh, Nilrem, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RIF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx OllyDBG of the authors.

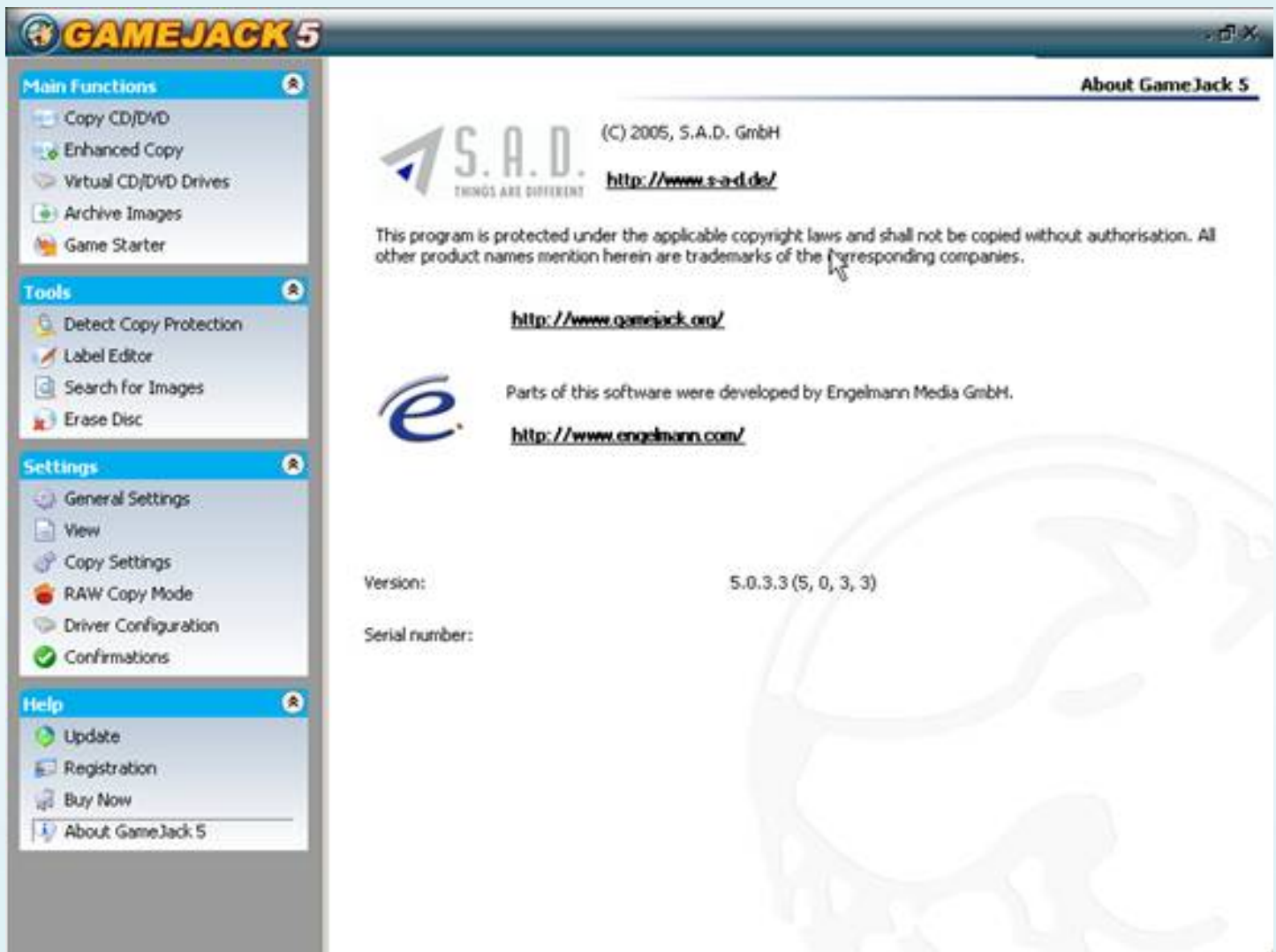
To be continued ...

Written by [hacnho](#) (tutorial date: Hong Source 17/09/2005)

Armadillo collect sand-stone

Series # 7: Armadillo 4.xx-Code Splicing + IAT Redirect

Target: GameJack TM v5.0.3.3



I. Intro

Hello all, in this tut, I'll explaint a way for a target unpack use options Code Splicing Importable and elimination! So easy. Ok, we goooooooooo!

II. Tools

- 1.OllyDBG - The best config debugger for ArmMUP by hacnho.
- 2.LordPE 1.4 Deluxe
- 3.Import REConstructor 1.6 Final
- 4.API Address 1.0
- 5.Arma Inline 0:41
- 6.Wark 1.3

III. Unpacking

_Load Target into OllyDBG:

004F4000	60		PUSHAD
004F4001	E8 00000000		CALL GameJack.004F4006
004F4006	5D		POP EBP
004F4007	50		PUSH EAX
004F4008	51		PUSH ECX
004F4009	0FCA		BSWAP EDX
004F400B	F7D2		NOT EDX
004F400D	9C		PUSHAD
004F400E	F7D2		NOT EDX
004F4010	0FCA		BSWAP EDX
004F4012	EB 0F		JMP SHORT GameJack.004F4023
004F4014	B9 EB0FB8EB		MOV ECX,EBB80FEB
004F4019	07		POP ES
004F401A	B9 EB0F90EB		MOV ECX,EB900FEB
004F401F	08FD		OR CH,BH
004F4021	EB 0B		JMP SHORT GameJack.004F402E
004F4023	E2		PRETIX RETNE

_Set A breakpoint at API GetModuleHandleA. Shift F9. we still here:

77E7AD86	837C24 04 00		CMP DWORD PTR SS:[ESP+4],0
77E7AD8B	0F84 37010000		JE kernel32.77E7AEC8
77E7AD91	FF7424 04		PUSH DWORD PTR SS:[ESP+4]
77E7AD95	E8 F8050000		CALL kernel32.77E7B392
77E7AD9A	85C0		TEST EAX,EAX
77E7AD9C	74 08		JE SHORT kernel32.77E7ADA6
77E7AD9E	FF70 04		PUSH DWORD PTR DS:[EAX+4]
77E7ADA1	E8 27060000		CALL kernel32.GetModuleHandleW
77E7ADA6	C2 0400		RETN 4
77E7ADA9	55		PUSH EBP
77E7ADAA	8BEC		MOV EBP,ESP
77E7ADAC	83EC 10		SUB ESP,10
77E7ADAE	57		PUSH EDI

00127958	00DC5828	[CALL to GetModuleHandleA from 00DC5822
0012795C	000D9BAC	pModule = "kernel32.dll"
00127960	000DAE48	ASCII "VirtualAlloc"
00127964	77F75690	ntdll.RtlLeaveCriticalSection
00127968	000DD968	
0012796C	00000000	
00127970	00000000	
00127974	00000000	
00127978	00000000	
0012797C	00000000	
00127980	00000000	

_Shift + F9:

00127958	00DC5828	[CALL to GetModuleHandleA from 00DC5822
0012795C	000D9BAC	pModule = "kernel32.dll"
00127960	000DAE48	ASCII "VirtualAlloc"
00127964	77F75690	ntdll.RtlLeaveCriticalSection
00127968	000DD968	
0012796C	00000000	
00127970	00000000	
00127974	00000000	
00127978	00000000	
0012797C	00000000	
00127980	00000000	

_Shift + F9:

001276BC	000B5257	[CALL to GetModuleHandleA from 000B5251
001276C0	0012780C	pModule = "kernel32.dll"
001276C4	0012CB28	
001276C8	4B2B1385	
001276CC	00000000	
001276D0	00DD9084	
001276D4	CD8004B	
001276D8	00000000	
001276DC	80000000	
001276E0	00003FFF	
001276E4	00000000	

_That The time for us to say! Now, with Trace down F8 to RETN4

77E7AD86	837C24 04 00		CMP DWORD PTR SS:[ESP+4],0
77E7AD8B	0F84 37010000		JE kernel32.77E7AEC8
77E7AD91	FF7424 04		PUSH DWORD PTR SS:[ESP+4]
77E7AD95	E8 F8050000		CALL kernel32.77E7B392
77E7AD9A	85C0		TEST EAX,EAX
77E7AD9C	74 08		JE SHORT kernel32.77E7ADA6
77E7AD9E	FF70 04		PUSH DWORD PTR DS:[EAX+4]
77E7ADA1	E8 27060000		CALL kernel32.GetModuleHandleW
77E7ADA6	C2 0400		RETN 4
77E7ADA9	55		PUSH EBP
77E7ADAA	8BEC		MOV EBP,ESP
77E7ADAC	83EC 10		SUB ESP,10
77E7ADAF	57		PUSH EDI
77E7ADB0	64 01 18000000		MOV EAX,DWORD PTR ES:[18]

_You Still here:

```

00085257 8B0D 3C1EDE00 MOV ECX,DWORD PTR DS:[DD1E3C]
0008525D 89040E MOV DWORD PTR DS:[ESI+ECX],EAX
00085260 A1 3C1EDE00 MOV EAX,DWORD PTR DS:[DE1E3C]
00085265 391C06 CMP DWORD PTR DS:[ESI+EAX],EBX
00085268 v 75 16 JNZ SHORT 00085280
0008526A 8D85 B4FEFFFF LEA EAX,DWORD PTR SS:[EBP-14C]
00085270 50 PUSH EAX
00085271 FF15 B842DD00 CALL DWORD PTR DS:[DD42B8]
00085277 8B0D 3C1EDE00 MOV ECX,DWORD PTR DS:[DE1E3C]
0008527D 89040E MOV DWORD PTR DS:[ESI+ECX],EAX
00085280 A1 3C1EDE00 MOV EAX,DWORD PTR DS:[DE1E3C]
00085285 391C06 CMP DWORD PTR DS:[ESI+EAX],EBX
00085288 v 0F84 2F010000 JE 000853BD
0008528E 33C9 XOR ECX,ECX
00085290 8B07 MOV EAX,DWORD PTR DS:[EDI]
00085292 3918 CMP DWORD PTR DS:[EAX],EBX
00085294 v 74 06 JE SHORT 0008529C
00085296 41 INC ECX
00085297 83C0 0C ADD EAX,0C
0008529A ^ EB F6 JMP SHORT 00085292
0008529C 8B09 MOV EBX,ECX
0008529E C1E3 02 SHL EBX,2
000852A1 53 PUSH EBX
000852A2 E8 8BDD0100 CALL 00DD3032
000852A7 8B0D 341EDE00 MOV ECX,DWORD PTR DS:[DE1E34]
000852AD 89040E MOV DWORD PTR DS:[ESI+ECX],EAX

```

Magic Jump!

_We Have to patch the JE to JMP:

```

00085277 8B0D 3C1EDE00 MOV ECX,DWORD PTR DS:[DE1E3C]
0008527D 89040E MOV DWORD PTR DS:[ESI+ECX],EAX
00085280 A1 3C1EDE00 MOV EAX,DWORD PTR DS:[DE1E3C]
00085285 391C06 CMP DWORD PTR DS:[ESI+EAX],EBX
00085288 v E9 3B010000 JMP 000853BD
0008528D 90 NOP
0008528E 33C9 XOR ECX,ECX
00085290 8B07 MOV EAX,DWORD PTR DS:[EDI]
00085292 3918 CMP DWORD PTR DS:[EAX],EBX
00085294 v 74 06 JE SHORT 0008529C
00085296 41 INC ECX
00085297 83C0 0C ADD EAX,0C
0008529A ^ EB F6 JMP SHORT 00085292
0008529C 8B09 MOV EBX,ECX

```

_Now, Hd GetModuleHandleA, set a breakpoint at API VirtualAlloc: he VirtualAlloc, Shift + F9:

```

00126FA0 73421D07 CALL to VirtualAlloc from 73421D05
00126FA4 00000000 Address = NULL
00126FA8 00400000 Size = 400000 (4194304.)
00126FAC 00002000 AllocationType = MEM_RESERVE
00126FB0 00000004 Protect = PAGE_READWRITE
00126FB4 00000000
00126FB8 00000001
00126FBC 00127004
00126FC0 73421CCD RETURN to 73421CCD from 73421CDB
00126FC4 73421BCE RETURN to 73421BCE from 73421CAC
00126FC8 73421B50 RETURN to 73421B50 from 73421B7C
77E7AC72 55 PUSH EBP
77E7AC73 8BEC MOV EBP,ESP
77E7AC75 FF75 14 PUSH DWORD PTR SS:[EBP+14]
77E7AC78 FF75 10 PUSH DWORD PTR SS:[EBP+10]
77E7AC7B FF75 0C PUSH DWORD PTR SS:[EBP+0C]
77E7AC7E FF75 08 PUSH DWORD PTR SS:[EBP+08]
77E7AC81 6A FF PUSH -1
77E7AC83 E8 9CFFFFFF CALL kernel32.VirtualAllocEx
77E7AC88 5D POP EBP
77E7AC89 C2 1000 RETN 10
77E7AC8C 8B4424 04 MOV EAX,DWORD PTR SS:[ESP+4]
77E7AC90 85C0 TEST EAX,EAX
77E7AC92 v 0F84 B7010000 JE kernel32.77E7AE4F
77E7AC98 A8 01 TEST AL,1
77E7AC9A ^ 0F85 CAF7FEFF JNZ kernel32.77E6A46A

```

_Alt + F9:

73421D07	8BF0	MOV ESI,EAX
73421D09	85F6	TEST ESI,ESI
73421D0B	0F84 C5700400	JE 73468DD6
73421D11	6A 04	PUSH 4
73421D13	68 00100000	PUSH 1000
73421D18	68 00000100	PUSH 10000
73421D1D	56	PUSH ESI
73421D1E	FFD7	CALL EDI
73421D20	85C0	TEST EAX,EAX
73421D22	0F84 A0700400	JE 73468DC8
73421D28	81FD 30B05273	CMP EBP,7352B030
73421D2E	0F85 74700400	JNZ 73468DA8
73421D34	A1 30B05273	MOV EAX,DWORD PTR DS:[7352B030]
73421D39	85C0	TEST EAX,EAX

_Shift + F9, Alt + F9:

73421D20	85C0	TEST EAX,EAX
73421D22	0F84 A0700400	JE 73468DC8
73421D28	81FD 30B05273	CMP EBP,7352B030
73421D2E	0F85 74700400	JNZ 73468DA8
73421D34	A1 30B05273	MOV EAX,DWORD PTR DS:[7352B030]
73421D39	85C0	TEST EAX,EAX
73421D3B	0F84 49700400	JE 73468D8A
73421D41	A1 34B05273	MOV EAX,DWORD PTR DS:[7352B034]
73421D46	85C0	TEST EAX,EAX
73421D48	0F84 4B700400	JE 73468D99
73421D4E	8D86 00004000	LEA EAX,DWORD PTR DS:[ESI+4000000]
73421D54	8D4D 18	LEA ECX,DWORD PTR SS:[EBP+18]
73421D57	8D95 98000000	LEA EDX,DWORD PTR SS:[EBP+98]
73421D5D	8945 14	MOV DWORD PTR SS:[EBP+14],EAX
73421D60	8975 10	MOV DWORD PTR SS:[EBP+10],ESI

_Shift + F9, Alt + F9:

00DC871C	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX
00DC8722	83B0 6CD7FFFF 00	CMP DWORD PTR SS:[EBP-2894],0
00DC8729	74 64	JE SHORT 00DC878F
00DC872B	6A 40	PUSH 40
00DC872D	68 00100000	PUSH 1000
00DC8732	FFB5 64D7FFFF	PUSH DWORD PTR SS:[EBP-289C]
00DC8738	FF35 807DDE00	PUSH DWORD PTR DS:[DE7D80],EAX
00DC873E	FF15 8C410D00	CALL DWORD PTR DS:[DD418C]
00DC8744	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX
00DC874A	83B0 6CD7FFFF 00	CMP DWORD PTR SS:[EBP-2894],0
00DC8751	74 3C	JE SHORT 00DC878F
00DC8753	8B85 6CD7FFFF	MOV EAX,DWORD PTR SS:[EBP-2894]
00DC8759	A3 807DDE00	MOV DWORD PTR DS:[DE7D80],EAX
00DC875E	8B85 6CD7FFFF	MOV EAX,DWORD PTR SS:[EBP-2894]
00DC8764	3B85 0CD8FFFF	CMP EAX,DWORD PTR SS:[EBP-27F4]
00DC876A	76 23	JBE SHORT 00DC878F
00DC876C	8B85 0CD8FFFF	MOV EAX,DWORD PTR SS:[EBP-27F4]
00DC8773	8B85 7CD8FFFF	MOV EAX,DWORD PTR SS:[EBP-2694]

GaneJack.00400000
kernel32.VirtualAlloc

GaneJack.00400000
GaneJack.00400000

_Now, Trace down with a pass by the F8 Function VirtualAlloc:

00DC871C	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX
00DC8722	83B0 6CD7FFFF 00	CMP DWORD PTR SS:[EBP-2894],0
00DC8729	74 64	JE SHORT 00DC878F
00DC872B	6A 40	PUSH 40
00DC872D	68 00100000	PUSH 1000
00DC8732	FFB5 64D7FFFF	PUSH DWORD PTR SS:[EBP-289C]
00DC8738	FF35 807DDE00	PUSH DWORD PTR DS:[DE7D80],EAX
00DC873E	FF15 8C410D00	CALL DWORD PTR DS:[DD418C]
00DC8744	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX
00DC874A	83B0 6CD7FFFF 00	CMP DWORD PTR SS:[EBP-2894],0
00DC8751	74 3C	JE SHORT 00DC878F
00DC8753	8B85 6CD7FFFF	MOV EAX,DWORD PTR SS:[EBP-2894]
00DC8759	A3 807DDE00	MOV DWORD PTR DS:[DE7D80],EAX
00DC875E	8B85 6CD7FFFF	MOV EAX,DWORD PTR SS:[EBP-2894]
00DC8764	3B85 0CD8FFFF	CMP EAX,DWORD PTR SS:[EBP-27F4]
00DC876A	76 23	JBE SHORT 00DC878F

kernel32.VirtualAlloc

GaneJack.00400000

_Then You see in the FPU:

Registers (FPU)	
EAX	03510000
ECX	77E7AC6F kernel32.77E7AC6F
EDX	7FFE0304
EBX	00000000
ESP	00127964
EBP	0012CC80
ESI	4B2B1385
EDI	0012CB28
EIP	00DC8744
C 0	ES 0023 32bit 0(FFFFFFFF)
P 1	CS 001B 32bit 0(FFFFFFFF)
A 0	SS 0023 32bit 0(FFFFFFFF)
Z 1	DS 0023 32bit 0(FFFFFFFF)
S 0	FS 003B 32bit 7FFDE000(FFF)
T 0	GS 0000 NULL
D 0	
O 0	LastErr ERROR_SUCCESS (00000000)

_the value of EAX contain the code to destroy our IAT. We must change this value to a real memory section of the arm. Now, Alt + M to go memory window:

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
003C0000	00002000				Map	R	R	
003D0000	00002000				Map	R	R	
003E0000	0000C000				Priv	RW	RW	
003F0000	00006000				Priv	RW	RW	
00400000	00002000	GameJack		PE header	Imag	R	RWE	
00402000	00073000	GameJack	.text		Imag	R	RWE	
00475000	00028000	GameJack	.rdata		Imag	R	RWE	
00490000	00007000	GameJack	.data		Imag	R	RWE	
00494000	00050000	GameJack	.text1	code	Imag	R	RWE	
004F4000	00010000	GameJack	.adata	SFX	Imag	R	RWE	
00504000	00020000	GameJack	.data1	data, import	Imag	R	RWE	
00524000	000E0000	GameJack	.pdata		Imag	R	RWE	
00604000	0018C000	GameJack	.rsrc	resources	Imag	R	RWE	
007C0000	00006000				Map	R E	R E	
00880000	00002000				Map	R E	R E	

_Ok, Address 4F4000 we choose to save the code. Go to Windows FPU! Change to EAX 4F4000.

Registers (FPU)		
EAX	004F4000	GameJack.<ModuleEntryPoint>
ECX	77E7AC6F	kernel32.77E7AC6F
EDX	7FFE0304	
EBX	00000000	
ESP	00127964	
EBP	0012CC80	
ESI	4B2B1385	
EDI	0012CB28	
EIP	00DC8744	
C 0	ES 0023 32bit	0(FFFFFFFF)
P 1	CS 001B 32bit	0(FFFFFFFF)

_ Delete the breakpoint: VirtualAlloc hd. Ok, now the code is splicing die. We must find the OEP. Ok, set a breakpoint at API SetProcessWorkingSetSize: he SetProcessWorkingSetSize. F9: OllyDBG ice:

Command: he SetProcessWorkingSetSize HE address -- HW break on execution

Hardware breakpoint 1 at kernel32.SetProcessWorkingSetSize

_Do Not delete breakpoint SetProcessWorkingSetSize. Now set a breakpoint other: he GetCurrentThreadId. F9:

0012D76C	000B5623	CALL to GetCurrentThreadId from 000B561D
0012D770	00000000	
0012D774	00000190	
0012D778	0012DEEC	
0012D77C	000CDD7C	RETURN to 00DCDD7C from 000B55EA
0012D780	0012FF04	
0012D784	00000000	
0012D788	004DFB63	GameJack.004DFB63
0012D78C	004CBE59	RETURN to GameJack.004CBE59 from 00DCDCES
0012D790	0050AE08	GameJack.0050AE08
0012D794	0012FF04	

_Hd SetProcessWorkingSetSize, HD GetCurrentThreadId. Ctrl + F9, F8:

000B5623	50	PUSH EAX
000B5624	FF75 FC	PUSH DWORD PTR SS:[EBP-4]
000B5627	E8 05000000	CALL 000B5631
000B562C	83C4 0C	ADD ESP,0C
000B562F	C9	LEAVE
000B5630	C3	RETN
000B5631	6A 14	PUSH 14
000B5633	E8 FAD90100	CALL 00DD3032
000B5638	85C0	TEST EAX,EAX
000B563A	59	POP ECX
000B563B	74 13	JE SHORT 000B5650

Dow _Trace to RETN with F8:


```

000CDD7C 6A 00          PUSH 0
000CDD7E E8 91D8FEFF    CALL 000BB614
000CDD83 59             POP ECX
000CDD84 BF 68D9DD00    MOV EDI,00DD968
000CDD89 8BCF          MOV ECX,EDI
000CDD8B E8 4FA4FDFF    CALL 00DA81DF
000CDD90 84C0          TEST AL,AL
000CDD92 75 09          JNZ SHORT 00DCDD9D
000CDD94 6A 01          PUSH 1
000CDD96 8BCF          MOV ECX,EDI
000CDD98 E8 7FF3FDFF    CALL 00DA011C
000CDD9D B9 D0CCDD00    MOV ECX,00CCDD0
000CDDA2 C705 70A0DD00 5CAFDD00    MOV DWORD PTR DS:[DDA070],0DDAF5C
000CDDA4 E8 6E52DD00    CALL 00DD301F

```

_Scroll Down until you see:

```

000CDD01 75 18          JNZ SHORT 00DCDDEB
000CDD03 8B50 38        MOV EDX,DWORD PTR DS:[EAX+38]
000CDD06 FF76 18        PUSH DWORD PTR DS:[ESI+18]
000CDD09 3350 0C        XOR EDX,DWORD PTR DS:[EAX+C]
000CDD0C FF76 14        PUSH DWORD PTR DS:[ESI+14]
000CDD0F 3350 04        XOR EDX,DWORD PTR DS:[EAX+4]
000CDD12 FF76 10        PUSH DWORD PTR DS:[ESI+10]
000CDD15 2BCA          SUB ECX,EDX
000CDD17 FFD1          CALL ECX
000CDD19 EB 1D          JMP SHORT 00DCDE08
000CDD1B 83FA 01        CMP EDX,1
000CDD1E 75 1A          JNZ SHORT 00DCDE0A
000CDD21 FF76 04        PUSH DWORD PTR DS:[ESI+4]
000CDD24 8B50 38        MOV EDX,DWORD PTR DS:[EAX+38]
000CDD27 3350 0C        XOR EDX,DWORD PTR DS:[EAX+C]
000CDD2A FF76 08        PUSH DWORD PTR DS:[ESI+8]
000CDD2D 3350 04        XOR EDX,DWORD PTR DS:[EAX+4]
000CDD30 6A 00          PUSH 0
000CDD32 FF76 0C        PUSH DWORD PTR DS:[ESI+C]
000CDD35 2BCA          SUB ECX,EDX
000CDD37 FFD1          CALL ECX
000CDD39 8B08          MOV EBX,EAX
000CDD3B 5F           POP EDI
000CDD3D 8BC3          MOV EAX,EBX
000CDD3F 5E           POP ESI
000CDD41 5B           POP EBX
000CDD43 RETN

```

_What Do you think about this, me, I think J OEP. Now, press F2 at the ECX at DCDE06 Call. F9, and F7: OEP. Congratulation!

```

0046B54E 6A 74          PUSH 74
0046B550 68 E0694700    PUSH GameJack.004769E0
0046B555 E8 1A020000    CALL GameJack.0046B774
0046B55A 330B          XOR EBX,EBX
0046B55C 895D E0        MOV DWORD PTR SS:[EBP-20],EBX
0046B55F 53           PUSH EBX
0046B560 8B3D DC574700  MOV EDI,DWORD PTR DS:[4757DC]
0046B566 FFD7          CALL EDI
0046B568 66:8138 4D5A    CMP WORD PTR DS:[EAX],5A4D
0046B56D 75 1F          JNZ SHORT GameJack.0046B58E
0046B56F 8B48 3C        MOV ECX,DWORD PTR DS:[EAX+3C]
0046B572 03C8          ADD ECX,EAX
0046B574 55           PUSH EBX
0046B576 56           PUSH ESI
0046B578 57           PUSH EDI
0046B57A 58           POP EDI
0046B57C 59           POP ESI
0046B57E 5A           POP EBX
0046B580 RETN

```

_Now The very important step: Defeat Import elimination. We use the Arma Inline 0:41. But, before. We have to find some info to add into Inline arm. Now, Alt + M, click at section:

Memory map, item 26

Address = 00400000

Size = 00002000 (8192.)

Owner = 00400000 GameJack (itself)

Section =

Type = 01001002 IMAG

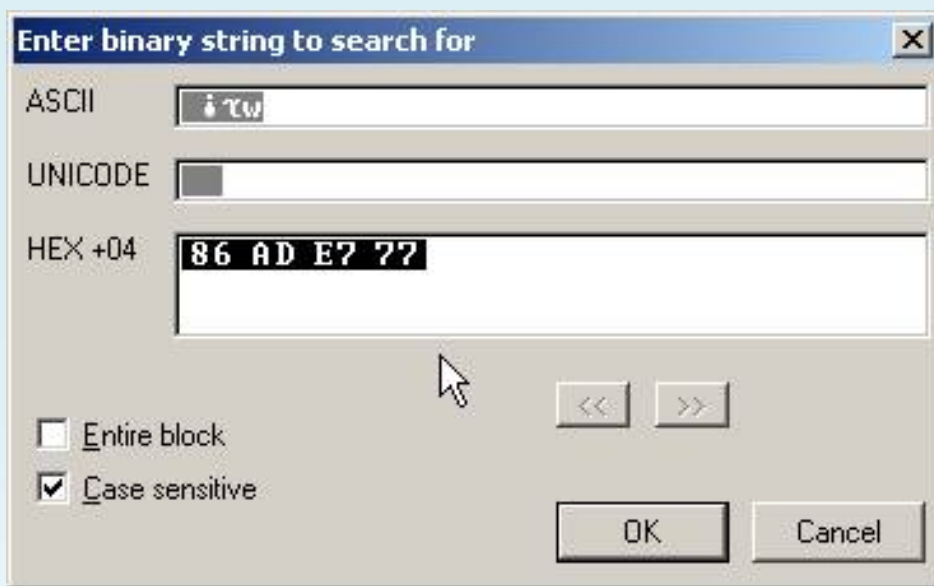
Access R =

Initial access = RWE

_Alt + F1? GetModuleHandleA and we see the Hex of this API is:

Command: HEX: 77E7AD86 - DEC: 2011671942 - ASCII: wq-†

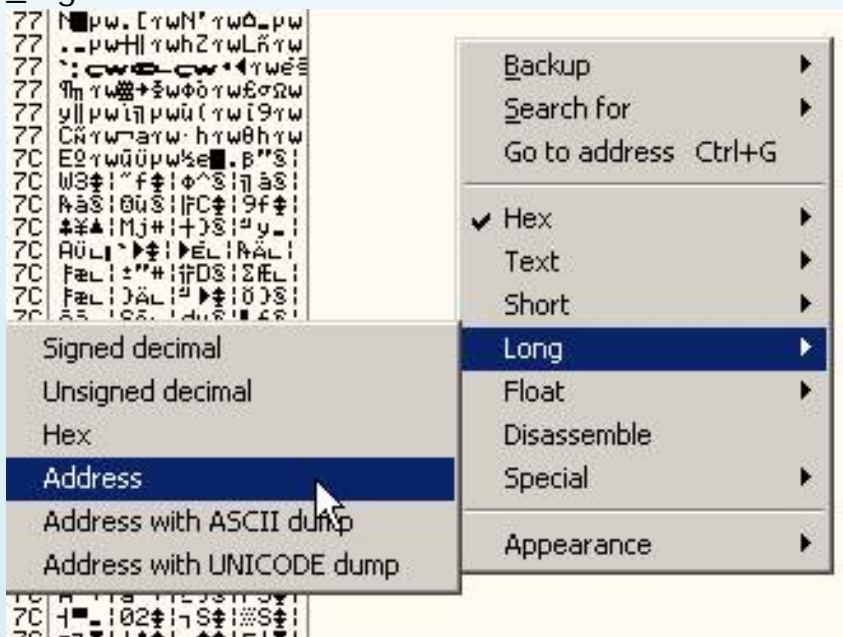
_Ctrl + B: 86ADE777:



_And We found:



_Right Click:



_And IAT we have the table:

00475000	77DD27D6	ADVAPI32.RegCreateKeyExA
00475004	77DD17D8	ADVAPI32.RegCloseKey
00475008	77DE63B1	ADVAPI32.RegSetValueExA
0047500C	77DE6F88	ADVAPI32.RegEnumKeyExA
00475010	77DD229A	ADVAPI32.RegOpenKeyExA
00475014	77DE6A68	ADVAPI32.RegEnumValueA
00475018	77DE6727	ADVAPI32.RegEnumKeyA
0047501C	77DD288B	ADVAPI32.RegCreateKeyA
00475020	77DD3111	ADVAPI32.RegNotifyChangeKeyValue
00475024	77DD2410	ADVAPI32.RegQueryValueExA

IAT Start

_Scroll Down:

004758A0	77E6BB8D	kernel32.GetTimeFormatA
004758A4	77E72897	kernel32.GetPrivateProfileStringA
004758A8	77E739A1	kernel32.MulDiv
004758AC	77E7A543	kernel32.CreateFileMappingA
004758B0	77E761AA	kernel32.MapViewOfFile
004758B4	77E768FA	kernel32.UnmapViewOfFile
004758B8	77E768E9	kernel32.GetCurrentProcessId
004758BC	77E7A745	kernel32.InitializeCriticalSection
004758C0	77E69A96	kernel32.GetShortPathNameA
004758C4	00DB65AB	
004758C8	7C1522E1	MFC71.7C1522E1
004758CC	7C1733E7	MFC71.7C1733E7

IAT End

_Total:

OEP: 46B54E:

IAT Start: 00475000

IAT End: 004758C0

IAT Ireland: 8C0

_Ok, Fire up and fill arm Inline:

1.PiD:

000006C0	wdfmgr	C:\WINDOWS\System32\wdfmgr.exe
000006F4	TSCHelp	C:\Program Files\TechSmith\Snagit 7\TSCHelp.exe
00000CE4	Snagit32	C:\Program Files\TechSmith\Snagit 7\Snagit32.exe
0000008C	WINWORD	C:\Program Files\Microsoft Office\OFFICE11\WINWORD.EXE
000000A8	GameJack	C:\Program Files\GameJack 5\GameJack.exe
00000FD0	emeditor	C:\Program Files\EmEditor\emeditor.exe

2. OEP: 46B54E:

3. IAT Start: 00475000

_Here We have:

ArmInline

File Help

(Slave) Process ID: 0x DA8

Start Of Target Code: 0x 46B54E

Length Of Target Code: 0x 20000

Code Splicing

Start Of Spliced Code: 0x

Length Of Spliced Code: 0x 20000

Undo Remove Splices

Import Elimination

Base Of Existing IAT: 0x 00475000

Length Of Existing IAT: 0x 8C0

New Base RVA Of IAT: 0x

Rebase IAT

Nanomites

Master Process ID: 0x

Remove Nanomites

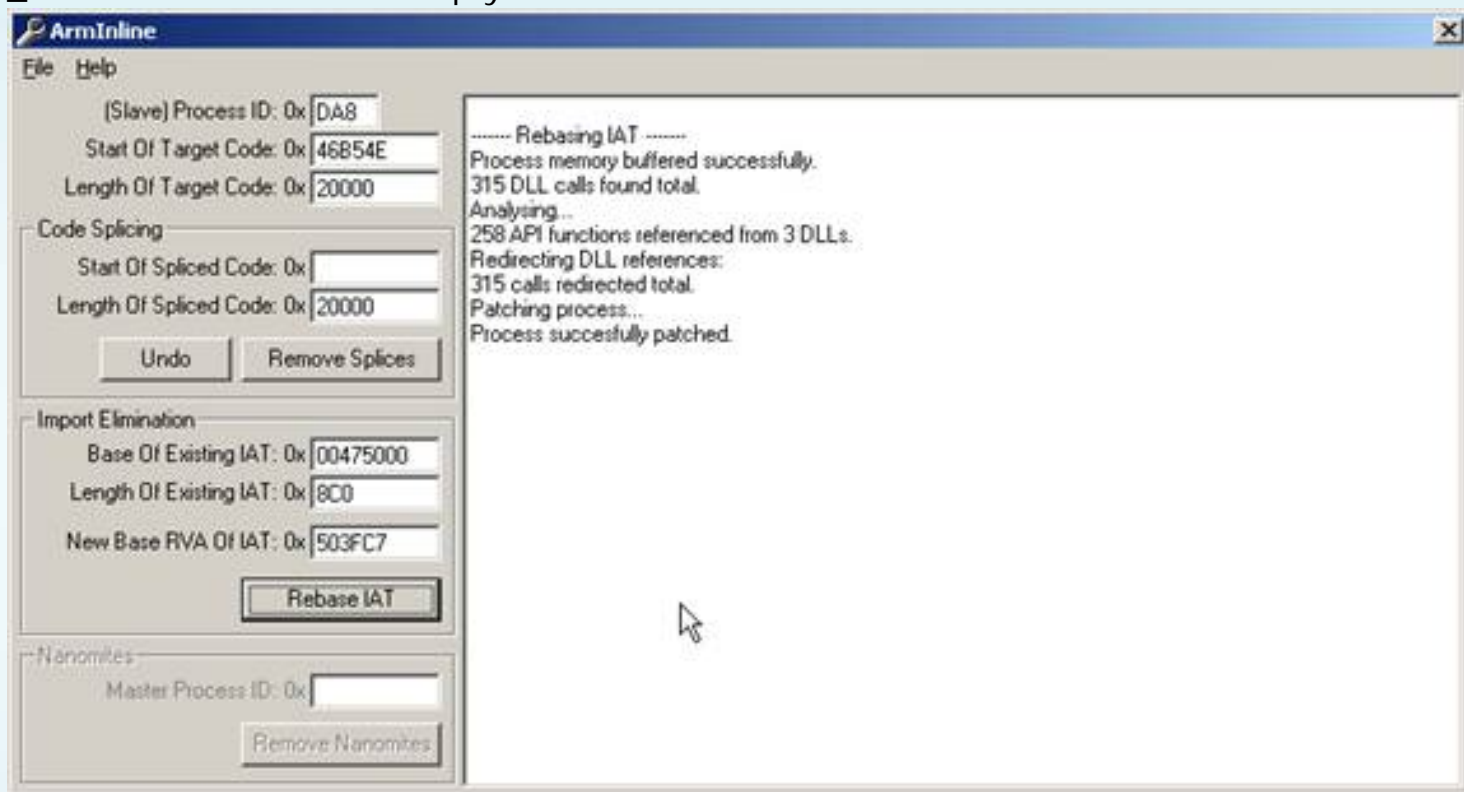
_The Special is: new base to contain our full IAT. We must find a cave memory. Now, back Olly, Alt + M, Double click at sections. ADATA

Address	Hex	Instruction
004F4000	87FB	XCHG EBX,EDI
004F4002	87D3	XCHG EBX,EDX
004F4004	7F 02	JG SHORT GameJack.004F4008
004F4006	7F 31	JG SHORT GameJack.004F4039
004F4008	F7D1	NOT ECX
004F400A	66:92	XCHG AX,DX
004F400C	66:92	XCHG AX,DX
004F400E	F7D1	NOT ECX
004F4010	87D3	XCHG EBX,EDX
004F4012	73 00	JNB SHORT GameJack.004F4014
004F4014	87FB	XCHG EBX,EDI
004F4016	B8 0F4B4700	MOV EAX,GameJack.00474B0F
004F4018	E9 80E8F0FF	JMP GameJack.004028A0
004F4020	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
004F4022	0FC8	BSWAP EAX
004F4024	7F 02	JG SHORT GameJack.004F4029
004F4026	7F 31	JG SHORT GameJack.004F405A

_Scroll Down:

Address	Hex	Instruction
00503FB7	68 F47E4700	PUSH GameJack.00477EF4
00503FBC	50	PUSH EAX
00503FB0	8D4D F0	LEA ECX,DWORD PTR SS:[EBP-10]
00503FC0	E9 22B1F2FF	JMP GameJack.0042F0E7
00503FC5	0000	ADD BYTE PTR DS:[EAX],AL
00503FC7	0000	ADD BYTE PTR DS:[EAX],AL
00503FC9	0000	ADD BYTE PTR DS:[EAX],AL
00503FCB	0000	ADD BYTE PTR DS:[EAX],AL
00503FCD	0000	ADD BYTE PTR DS:[EAX],AL
00503FCF	0000	ADD BYTE PTR DS:[EAX],AL
00503FD1	0000	ADD BYTE PTR DS:[EAX],AL
00503FD3	0000	ADD BYTE PTR DS:[EAX],AL
00503FD5	0000	ADD BYTE PTR DS:[EAX],AL
00503FD7	0000	ADD BYTE PTR DS:[EAX],AL
00503FD9	0000	ADD BYTE PTR DS:[EAX],AL
00503FDB	0000	ADD BYTE PTR DS:[EAX],AL
00503FDD	0000	ADD BYTE PTR DS:[EAX],AL

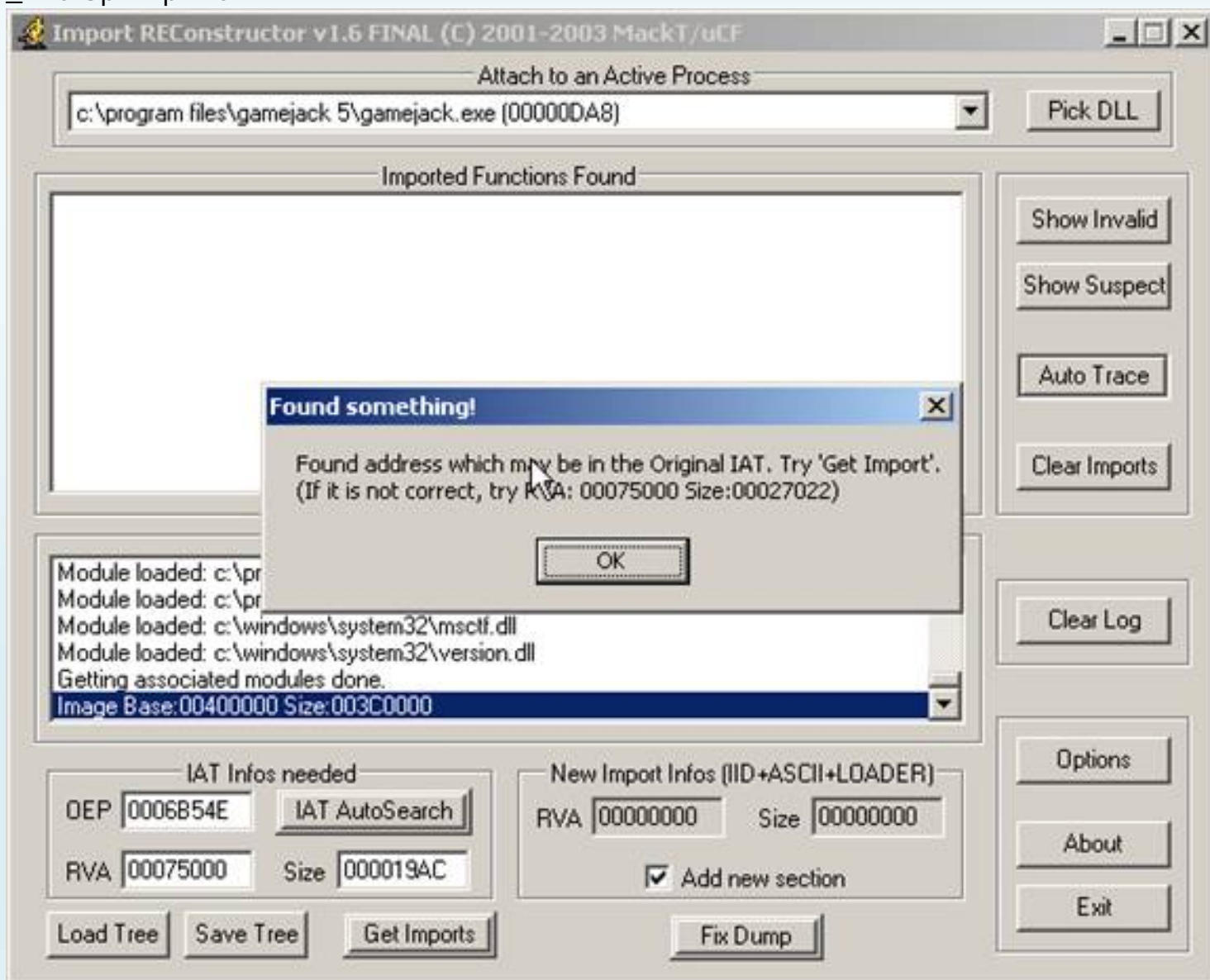
_The 503FC7 address is empty. We choose! Back to inline arm. Fill the address.



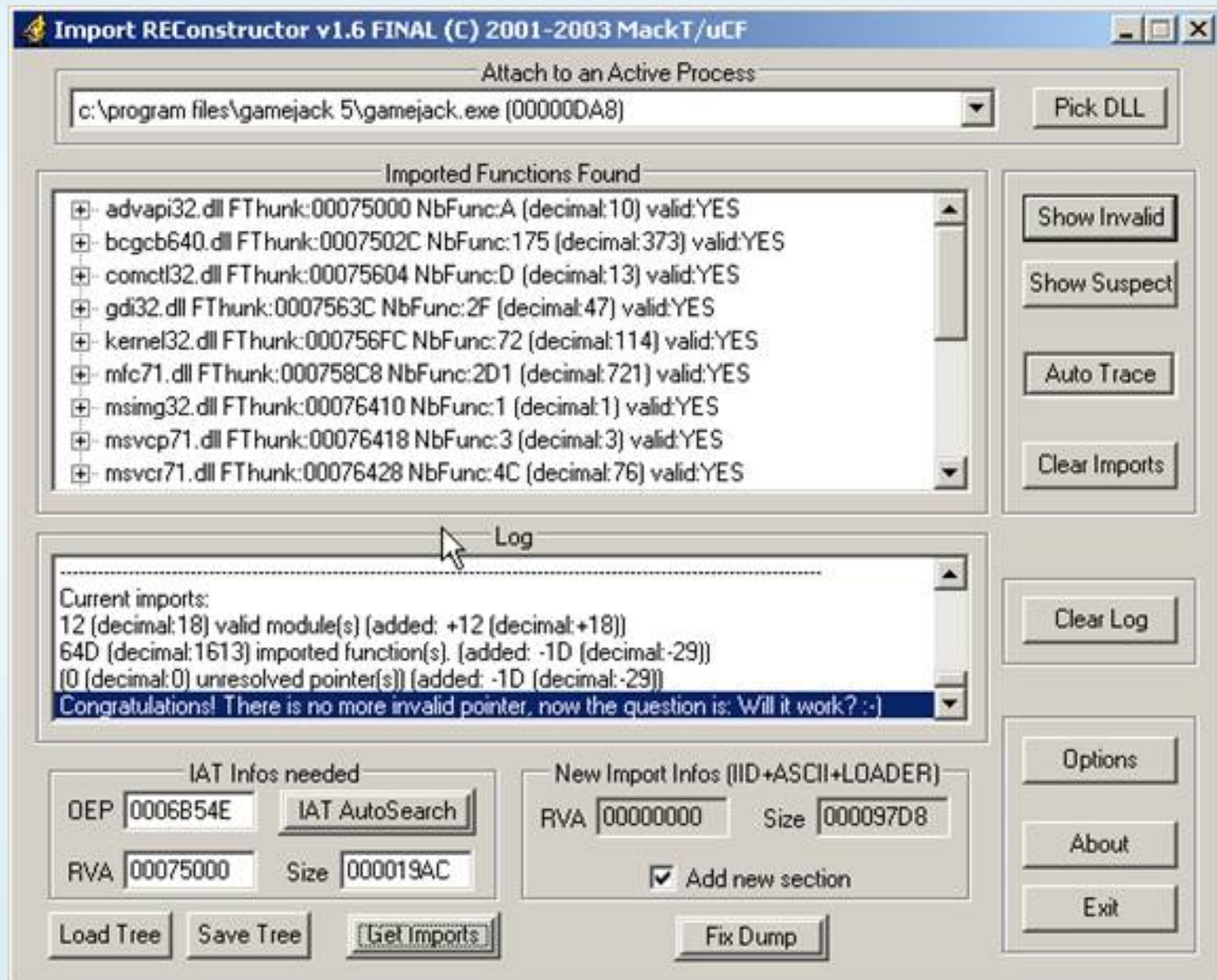
_Lady And Genlement: Import the elimination is defeat. Now use LordPE, Full dump:

Path	PID	ImageBase	ImageSize
c:\program files\alcohol soft\alcohol 120\star...	00000638	00400000	0003A000
i:\project delphi7\my project\crackerutils\cra...	00000680	00400000	000F4000
c:\program files\symantec antivirus\rtvscan.exe	0000068C	00400000	00163000
c:\windows\system32\wdfmgr.exe	000006C0	01000000	0000C000
c:\progra~1\systran\5.0\premium\systra~3.exe	00000104	00400000	00F5A000
c:\program files\techsmith\snagit 7\tschelp.exe	00000BF4	00400000	0000A000
i:\cracker\debug- disassembler\odbg110_org...	00000D64	00400000	00169000
c:\program files\microsoft office\office11\win...	00000D8C	30000000	008AA000
c:\program files\gamejack 5\gamejack...			003C0000
i:\cracker\unpacker\armadillo\arminlin...			00014000
c:\program files\emeditor\emeditor.exe			00069000
c:\program files\techsmith\snagit 7\sn...			00286000
i:\cracker\utilities\lordpe deluxe - 1.4\...			00036000
Path			

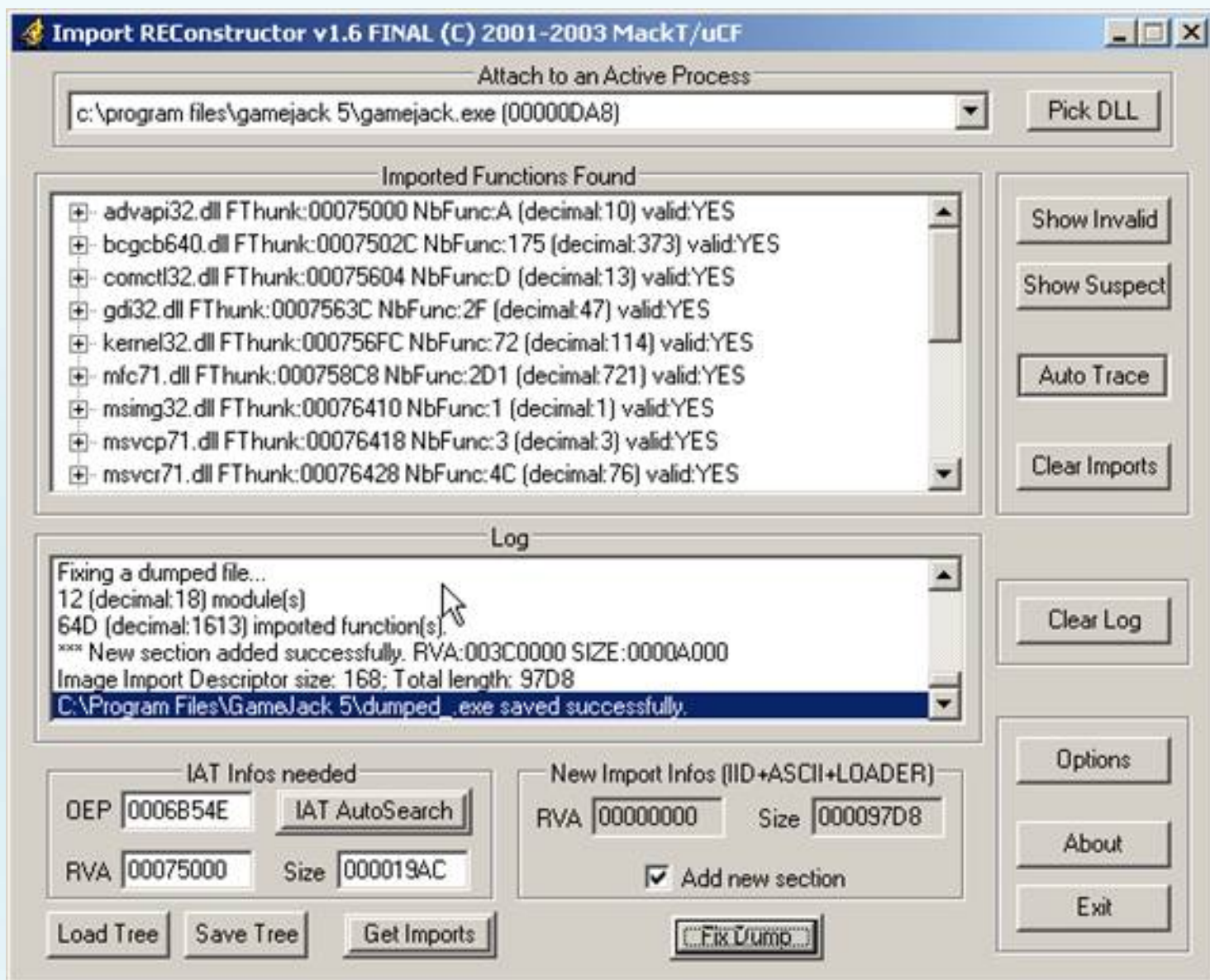
_Fire Up ImpREC:



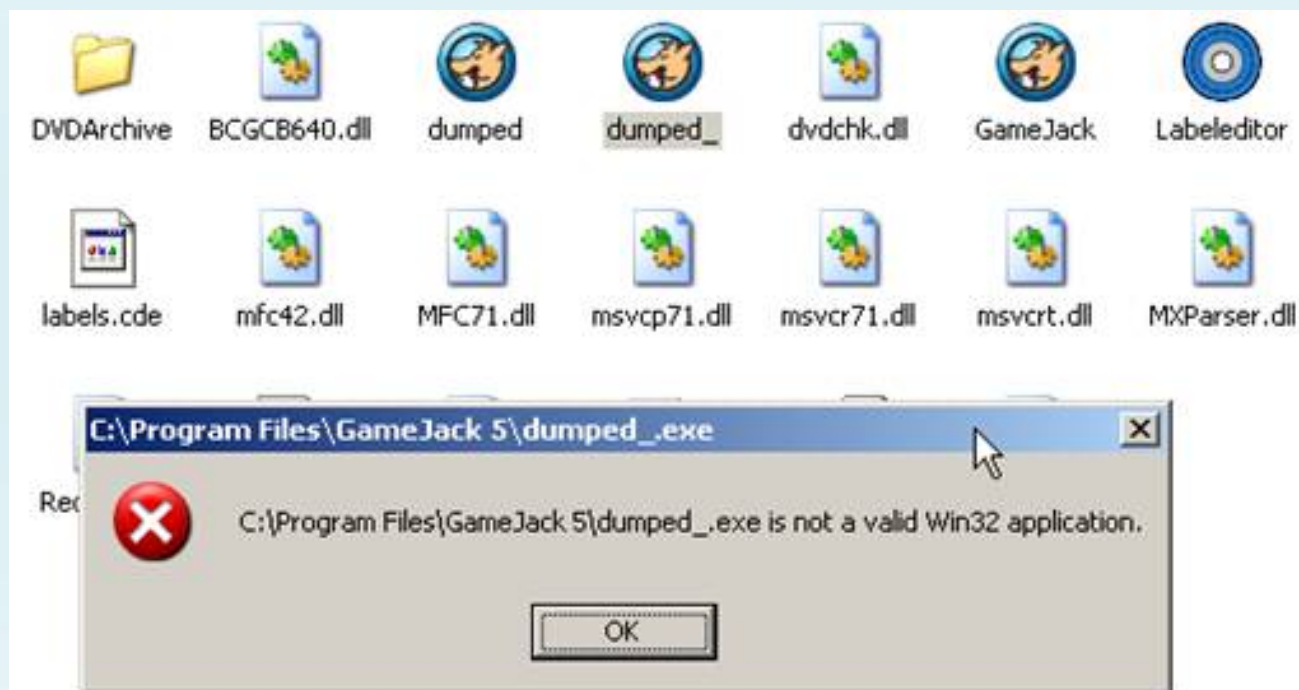
_Show Invalid, thanks Cut:



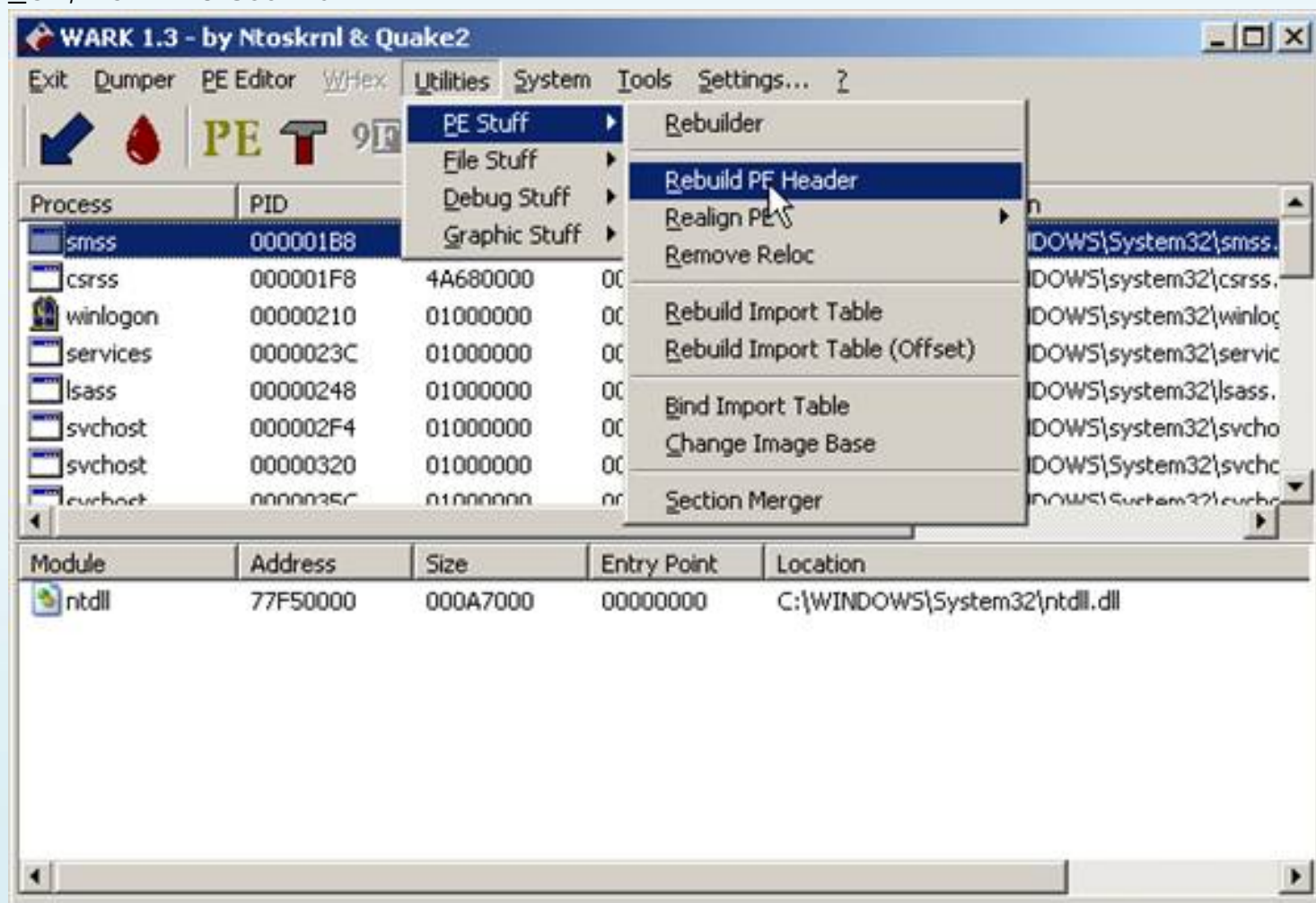
_Fix Dump:



Now, Run the target dumped.exe:

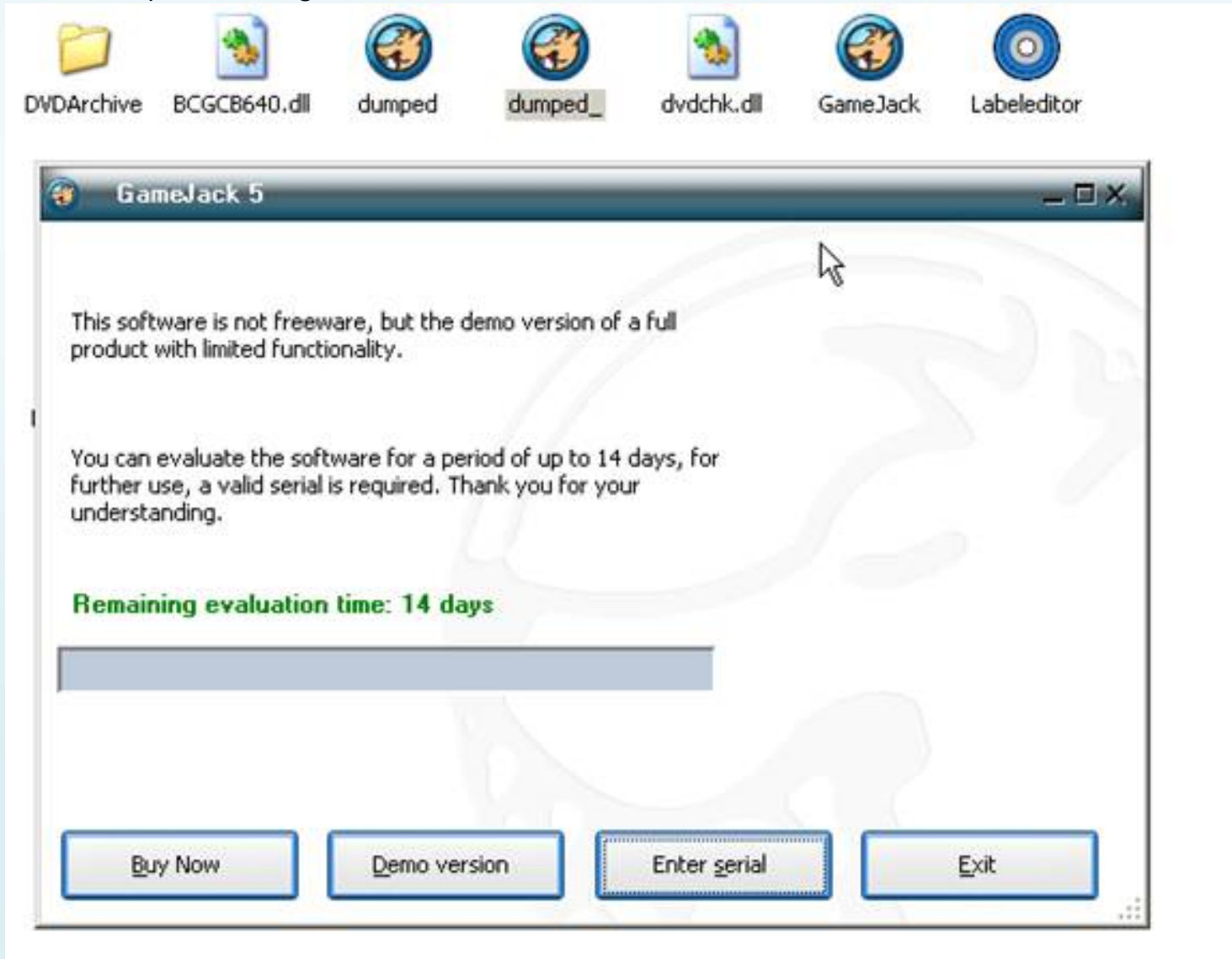


_Oh, Now. 1.3 Use Wark:





Run Dumped.exe again!



_Unpacked SuccessFul! Done ...

_Cracking:

User: Registered

Serial: 4A7SQ5-7K56XM-F2KN05-47ZMD6-32S97C

IV. Conclusion

So easy, huh?

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini, hoadongnoi, dqtn, hosiminh, Nilrem, fly, Madman_Hercules, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RiF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx OllyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 18/09/2005)

Armadillo collect sand-stone

Series # 7 Exp: Armadillo 4.xx-Code Splicing with other Method

I. Intro

All _Hi, some meals you have questions about this type of thing you have to do RM to MP3:

<pre> 0002A241 EB 1B 0002A243 FF35 B0AFD400 0002A249 68 8C050300 0002A24E E8 0A5DFFFF 0002A253 59 0002A254 59 0002A255 8985 10A0FFFF 0002A258 6A 40 0002A25D 69 00100000 0002A262 FF85 64D7FFFF 0002A268 FF35 B0AFD400 0002A26E FF15 8C61D300 0002A274 8985 6C07FFFF 0002A27A 838D 6C07FFFF 00 0002A281 0F84 9C000000 0002A287 838D B0AFD400 00 0002A28E 74 20 0002A290 6A 01 0002A296 59 0002A298 85C0 0002A29E 74 09 0002A29F 83A5 0CA8FFFF 00 0002A29E EB 1B 0002A2A0 A1 B0AFD400 0002A2A5 C1E8 10 0002A2A8 58 0002A2A9 68 6C050300 0002A2AB 6A 40 </pre>	<pre> JMP SHORT 0002A25B MOV DWORD PTR DS:[04AFB0] CALL 003D56C CALL 0001FF5D MOV ECX MOV ECX MOV DWORD PTR SS:[EBP+FFFFA010],EAX MOV 40 MOV 1000 MOV DWORD PTR SS:[EBP-289C] MOV DWORD PTR DS:[04AFB0] CALL DWORD PTR DS:[03618C] MOV DWORD PTR SS:[EBP-2894],EAX CMP DWORD PTR SS:[EBP-2894],0 JE 0002A323 CMP DWORD PTR DS:[04AFB0],0 JE SHORT 0002A26D MOV 1 TEST EAX,EAX JE SHORT 0002A2A0 AND DWORD PTR SS:[EBP+FFFFA00C],0 JMP SHORT 0002A25B MOV EAX,DWORD PTR DS:[04AFB0] SHR EAX,10 MOV EAX CALL 003D56C CALL 0001FF5D </pre>	<pre> ASCII "*** CS reserved %08X" 00040A58 00040A58 kernel32.VirtualAlloc 00040A58 ASCII "*** CS allocation %04X accepted" </pre>
--	---	---

_Tui Less time should be aware network processing to J. Fortunately today handle bags are a target italy chang so. Just 15 seconds, I settled this issue. Mong iamidiot pa satisfy nhé!

I. Tools

- 1.OllyDBG - The best config debugger for ArmMUP by hacnho.
- 2.LordPE 1.4 Deluxe
- 3.Import REConstructor 1.6 Final

II. Unpacking

Target: Zephyrous Keygenme 3



_Load Target into OllyDBG:

<pre> 00609000 60 00609001 E8 00000000 00609006 5D 00609007 50 00609008 51 00609009 0FCA 0060900B F7D2 0060900D 9C 0060900E F7D2 00609010 0FCA 00609012 EB 0F 00609014 B9 EB0FB8EB 00609019 07 0060901D B9 EB0F90EB </pre>	<pre> PUSHAD CALL cwpolywz.00609006 POP EBP PUSH EAX PUSH ECX BSWAP EDX NOT EDX PUSHAD NOT EDX BSWAP EDX JMP SHORT cwpolywz.00609023 MOV ECX,EBB80FEB POP ES MOV ECX,EBB80FEB </pre>	<pre> kernel32.77E814C7 Modification of segment </pre>
--	--	---

_Set Breakpoint at API CreateMutexA: BP CreateMutexA, Shift + F9:

```

77E7B6C5 55          PUSH EBP
77E7B6C6 8BEC       MOV EBP,ESP
77E7B6C8 51        PUSH ECX
77E7B6C9 51        PUSH ECX
77E7B6CA 56        PUSH ESI
77E7B6CB 33F6      XOR ESI,ESI
77E7B6CD 3975 10    CMP DWORD PTR SS:[EBP+10],ESI
77E7B6DE 74 31     JE SHORT kernel32.77E7B703
77E7B6E2 64:A1 18000000 MOV EAX,DWORD PTR FS:[18]

```

```

0012F710 005E1278  CALL to CreateMutexA from cwpolywz.005E1278
0012F714 00000000  pSecurity = NULL
0012F718 00000000  InitialOwner = FALSE
0012F71C 0012FDA0  MutexName = "RNC6E0FEDB"
0012F720 00000004
0012F724 00000000
0012F728 005F6623  cwpolywz.005F6623
0012F72C 00000000
0012F730 00000000
0012F734 00000000

```

_Ok, We have the address of Mutex is: 12FDA0. Now, Ctrl + G: 401000:

```

00401000 0000      ADD BYTE PTR DS:[EAX],AL
00401002 0000      ADD BYTE PTR DS:[EAX],AL
00401004 0000      ADD BYTE PTR DS:[EAX],AL
00401006 0000      ADD BYTE PTR DS:[EAX],AL
00401008 0000      ADD BYTE PTR DS:[EAX],AL
0040100A 0000      ADD BYTE PTR DS:[EAX],AL
0040100C 0000      ADD BYTE PTR DS:[EAX],AL
0040100E 0000      ADD BYTE PTR DS:[EAX],AL
00401010 0000      ADD BYTE PTR DS:[EAX],AL
00401012 0000      ADD BYTE PTR DS:[EAX],AL
00401014 0000      ADD BYTE PTR DS:[EAX],AL

```

_Patch To:

```

00401000 60        PUSHAD
00401001 9C        PUSHFD
00401002 68 A0FD1200 PUSH 12FDA0
00401007 33C0      XOR EAX,EAX
00401009 50        PUSH EAX
0040100A 50        PUSH EAX
0040100B E8 B5A6A777 CALL kernel32.CreateMutexA
00401010 9D        POPFD
00401011 61        POPAD
00401012 - E9 7A19A877 JMP kernel32.OpenMutexA
00401017 0000      ADD BYTE PTR DS:[EAX],AL
00401019 0000      ADD BYTE PTR DS:[EAX],AL

```

_Now, Press F9:

```

77E7B6C5 55          PUSH EBP
77E7B6C6 8BEC       MOV EBP,ESP
77E7B6C8 51        PUSH ECX
77E7B6C9 51        PUSH ECX
77E7B6CA 56        PUSH ESI
77E7B6CB 33F6      XOR ESI,ESI
77E7B6CD 3975 10    CMP DWORD PTR SS:[EBP+10],ESI
77E7B6DE 74 31     JE SHORT kernel32.77E7B703
77E7B6E2 64:A1 18000000 MOV EAX,DWORD PTR FS:[18]
77E7B6E8 FF75 10    PUSH DWORD PTR SS:[EBP+10]
77E7B6EB 8DB0 F80B0000 LEA ESI,DWORD PTR DS:[EAX+BF8]
77E7B6E1 8D45 F8    LEA EAX,DWORD PTR SS:[EBP-8]

```

_Ctrl + G: 401000. Press Ctrl + * to set a new origin:

```

00122564 0002137B  CALL to CreateMutexA from 0002137B
00122568 00049DA8  pSecurity = 00049DA8
0012256C 00000000  InitialOwner = FALSE
00122570 00048850  MutexName = "RNC6E0FEDB"
00122574 000410EC
00122578 00000000
0012257C 0012D650
00122580 00000000
00122584 00000000
00122588 00000000

```

_Ctrl + G: 401000. Press Ctrl + * to set a new origin:

```

00401000 60        PUSHAD
00401001 9C        PUSHFD
00401002 68 A0FD1200 PUSH 12FDA0
00401007 33C0      XOR EAX,EAX
00401009 50        PUSH EAX
0040100A 50        PUSH EAX
0040100B E8 B5A6A777 CALL kernel32.CreateMutexA
00401010 9D        POPFD
00401011 61        POPAD
00401012 - E9 7A19A877 JMP kernel32.OpenMutexA
00401017 0000      ADD BYTE PTR DS:[EAX],AL
00401019 0000      ADD BYTE PTR DS:[EAX],AL

```

_Then, BC CreateMutexA. Now put on BP Hardware ** ** execution on GetModuleHandleA and press F9:

_ 1st break:

```

001292A8 00026B68  CALL to GetModuleHandleA from 00026B68
001292AC 0003BD6C  pModule = "kernel32.dll"
001292B0 0003DDAC  ASCII "VirtualAlloc"
001292B4 00040A58  ntdll.RtlLeaveCriticalSection
001292B8 77F75690
001292BC 00000000
001292C0 00000000
001292C4 00000000
001292C8 00000000
001292CC 00000000

```

_ 2nd break:

3 Rd break:

_4 Break rd:

5 The break:

6 The break:

7 The

8 The break:

9 The break:

10 The break:

```
001286A0 00032985 CALL to GetModuleHandleA from 0003297F
001286A4 00000000 hModule = NULL
001286A8 00000000
001286AC 003A0005
001286B0 00F03008 UNICODE "Thank you for trying the Coding Worksh
001286B4 00000000
001286B8 00000154
001286BC 00000001
001286C0 00000010
001286C4 00000010
```


_11 The break:

A dialog appear. Press OK:

```
001283DC 77123069 CALL to GetModuleHandleA from OLEAUT32.77123069
001283E0 771A2010 pModule = "ole32.dll"
001283E4 0012841C
001283E8 77123021 RETURN to OLEAUT32.77123021 from OLEAUT32.77123069
001283EC 77122D20 OLEAUT32.77122D20
001283F0 00000000
001283F4 00000000
001283F8 00000000
001283FC 00000000
00128400 00000000
```

_12 The break:

```
00128580 74721BF6 CALL to GetModuleHandleA from MSCTF.74721BF6
00128584 00128588 pModule = "C:\\WINDOWS\\System32\\ntdll.dll"
00128588 575C3A43
0012858C 4F444E49
00128590 535C5357
00128594 65747379
00128598 5C32336D
0012859C 6C64746E
001285A0 6C642E6C
001285A4 001285A0
```

_13 The break:

```
00127EF0 10008A88 CALL to GetModuleHandleA from 10008A82
00127EF4 1000A3FC pModule = "kernel32.dll"
00127EF8 1000D688
00127EFC 00000000
00127F00 10004C4F
00127F04 77F59A7B RETURN to ntdll.77F59A7B from ntdll.77F78C4E
00127F08 77E7E323 RETURN to kernel32.77E7E323 from ntdll.RtlCreate
00127F0C 00127EF8
00127F10 00000000
00127F14 00128008 Pointer to next SEH record
```

_14 The break:

```
0012900C 00015A93 CALL to GetModuleHandleA from 00015A90
00129010 0012915C pModule = "kernel32.dll"
00129014 0012EA58
00129018 6F44A848
0012901C 00000000
00129020 0003B244
00129024 C011A940
00129028 00000000
0012902C 00000000
00129030 00000000
```

_That All! Now, in press F8 CPU trace down pass by RETN 4:

00015A93	8B0D 6C50D400	MOV ECX, DWORD PTR DS:[ESI+0A0]	kernel32.77E60000
00015A99	89040E	MOV DWORD PTR DS:[ESI+ECX], EAX	
00015A9C	A1 6C50D400	MOV EAX, DWORD PTR DS:[D4506C]	
00015AB1	391C06	CMP DWORD PTR DS:[ESI+EAX], EBX	
00015AB4	75 16	JNZ SHORT 00015ACC	
00015AB6	8085 B4FEFFFF	LEA EAX, DWORD PTR SS:[EBP-14C]	
00015ABC	50	PUSH EAX	kernel32.77E60000
00015ABD	FF15 B862D300	CALL DWORD PTR DS:[D362B8]	kernel32.LoadLibraryA
00015AC3	8B0D 6C50D400	MOV ECX, DWORD PTR DS:[D4506C]	
00015AC9	89040E	MOV DWORD PTR DS:[ESI+ECX], EAX	kernel32.77E60000
00015ACC	A1 6C50D400	MOV EAX, DWORD PTR DS:[D4506C]	
00015AD1	391C06	CMP DWORD PTR DS:[ESI+EAX], EBX	
00015AD4	75 16	JNZ SHORT 00015ACC	
00015ADA	33C3	XOR ECX, ECX	kernel32.77E7B5E1
00015ADC	8B07	MOV EAX, DWORD PTR DS:[EDI]	
00015ADE	3918	CMP DWORD PTR DS:[EAX], EBX	
00015AE0	74 06	JE SHORT 00015AE8	kernel32.77E7B5E1
00015AE2	41	INC ECX	kernel32.77E7B5E1
00015AE3	83C0 0C	ADD EAX, 0C	
00015AE6	EB F6	JMP SHORT 00015ADE	kernel32.77E7B5E1
00015AE8	8B09	MOV EBX, ECX	
00015AEA	C1E3 02	SHL EBX, 2	
00015AED	53	PUSH EBX	
00015AEE	E8 11FE0100	CALL 00035904	JMP to msvert.operator
00015AF3	8B0D 6450D400	MOV ECX, DWORD PTR DS:[D45064]	kernel32.77E60000
00015AF9	89040E	MOV DWORD PTR DS:[ESI+ECX], EAX	
00015AFC	53	PUSH EBX	
00015AFD	E8 02FE0100	CALL 00035904	JMP to msvert.operator
00015B02	59	POP ECX	0012EA58
00015B03	59	POP ECX	0012EA58
00015B04	8B0D 6850D400	MOV ECX, DWORD PTR DS:[D45068]	
00015B0A	89040E	MOV DWORD PTR DS:[ESI+ECX], EAX	kernel32.77E60000
00015B0D	8B07	MOV EAX, DWORD PTR DS:[EDI]	
00015B0E	8B05 0C555555	MOV DWORD PTR SS:[EBP-154], EAX	kernel32.77E60000

_Delete The breakpoint GetModuleHandleA. Set breakpoint VirtualAlloc, F9:

```
001288F0 73421D07 CALL to VirtualAlloc from 73421D05
001288F4 00000000 Address = NULL
001288F8 00400000 Size = 400000 (4194304.)
001288FC 00002000 AllocationType = MEM_RESERVE
00128900 00000004 Protect = PAGE_READWRITE
00128904 00000000
00128908 00000001
0012890C 00128954
00128910 73421CCD RETURN to 73421CCD from 73421D0B
00128914 73421BCE RETURN to 73421BCE from 73421D0B
```

77E7AC72	55	PUSH EBP	
77E7AC73	8BEC	MOV EBP,ESP	
77E7AC75	FF75 14	PUSH DWORD PTR SS:[EBP+14]	
77E7AC78	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
77E7AC7B	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
77E7AC7E	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
77E7AC81	6A FF	PUSH -1	
77E7AC83	E8 9CFFFFFF	CALL kernel32.VirtualAllocEx	
77E7AC88	5D	POP EBP	73421D07
77E7AC89	C2 1000	RETN 10	
77E7AC8C	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]	
77E7AC90	85C0	TEST EAX,EAX	
77E7AC92	7E 0F84 B7010000	JE kernel32.77E7AE4F	
77E7AC98	A8 01	TEST AL,1	

Shift + F9, Alt + F9 2 times:

0002A214	FF35 B0AFD400	PUSH DWORD PTR DS:[04AFB0]	
0002A21A	FF15 8C61D300	CALL DWORD PTR DS:[03618C]	kernel32.VirtualAlloc
0002A220	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX	
0002A226	83BD 6CD7FFFF 00	CMP DWORD PTR SS:[EBP-2894],0	
0002A22D	7E 0F84 13010000	JE 0002A346	
0002A233	6A 01	PUSH 1	
0002A235	58	POP EAX	00D40A58
0002A236	85C0	TEST EAX,EAX	
0002A238	74 09	JE SHORT 0002A243	
0002A23A	83A5 10A8FFFF 00	AND DWORD PTR SS:[EBP+FFFA810],0	
0002A241	EB 18	JMP SHORT 0002A25B	
0002A243	FF35 B0AFD400	PUSH DWORD PTR DS:[04AFB0]	
0002A249	68 8CD5D300	PUSH 00D5D300	ASCII "*** CS reserved %08X"
0002A24E	E8 0A50FFFF	CALL 00D1FF5D	00D40A58
0002A253	59	POP ECX	00D40A58
0002A254	59	POP ECX	
0002A255	8985 10A8FFFF	MOV DWORD PTR SS:[EBP+FFFA810],EAX	
0002A25B	6A 40	PUSH 40	
0002A25D	68 00100000	PUSH 1000	
0002A262	FFB5 64D7FFFF	PUSH DWORD PTR SS:[EBP-289C]	
0002A268	FF35 B0AFD400	PUSH DWORD PTR DS:[04AFB0]	
0002A26E	FF15 8C61D300	CALL DWORD PTR DS:[03618C]	kernel32.VirtualAlloc
0002A274	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX	
0002A27A	83BD 6CD7FFFF 00	CMP DWORD PTR SS:[EBP-2894],0	
0002A281	7E 0F84 9C000000	JE 0002A323	
0002A287	833D B0AFD400 00	CMP DWORD PTR DS:[04AFB0],0	
0002A28E	74 20	JE SHORT 0002A2B0	
0002A290	6A 01	PUSH 1	
0002A292	58	POP EAX	00D40A58
0002A293	85C0	TEST EAX,EAX	
0002A295	74 09	JE SHORT 0002A2A0	
0002A297	83A5 0CA8FFFF 00	AND DWORD PTR SS:[EBP+FFFA80C],0	
0002A29E	EB 1B	JMP SHORT 0002A2BB	
0002A2A0	A1 B0AFD400	MOV EAX,DWORD PTR DS:[04AFB0]	
0002A2A5	C1E8 10	SHR EAX,10	
0002A2A8	58	POP EAX	
0002A2A9	68 6CD5D300	PUSH 00D5D300	ASCII "*** CS allocation %04X acco
0002A2AE	E8 A8CFFFFF	CALL 00D1FF5D	00D40A58
0002A2B3	59	POP ECX	00D40A58
0002A2B4	59	POP ECX	
0002A2B5	8985 0CA8FFFF	MOV DWORD PTR SS:[EBP+FFFA80C],EAX	
0002A2BB	EB 20	JMP SHORT 0002A2C5	

_Now, If you trace down to pass by the Code Splicing, you will be crash! 100&percent; sure. You must follow my method. Now with trace down F8 to:

83BD 6CD7FFFF 00	CMP DWORD PTR SS:[EBP-2894],0	
7E 0F84 13010000	JE 0002A346	

_The Jump will jump to other place. We want this jump pass by. OK, Alt + F1: HD VirtualAlloc:

0002A214	FF35 B0AFD400	PUSH DWORD PTR DS:[04AFB0]	
0002A21A	FF15 8C61D300	CALL DWORD PTR DS:[03618C]	kernel32.VirtualAlloc
0002A220	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX	
0002A226	83BD 6CD7FFFF 00	CMP DWORD PTR SS:[EBP-2894],0	
0002A22D	7E 0F84 13010000	JE 0002A346	
0002A233	6A 01	PUSH 1	
0002A235	58	POP EAX	00D40A58
0002A236	85C0	TEST EAX,EAX	
0002A238	74 09	JE SHORT 0002A243	
0002A23A	83A5 10A8FFFF 00	AND DWORD PTR SS:[EBP+FFFA810],0	
0002A241	EB 18	JMP SHORT 0002A25B	
0002A243	FF35 B0AFD400	PUSH DWORD PTR DS:[04AFB0]	
0002A249	68 8CD5D300	PUSH 00D5D300	ASCII "*** CS reserved %08X"
0002A24E	E8 0A50FFFF	CALL 00D1FF5D	00D40A58

_By Pass OK. Continue to trace:

0002A26E FF15 8C61D300 CALL DWORD PTR DS: [D3618C]; kernel32.VirtualAlloc

0002A253	59	POP ECX	00D40A58
0002A254	59	POP ECX	00D40A58
0002A255	8985 10A8FFFF	MOV DWORD PTR SS:[EBP+FFFA810],EAX	
0002A25B	6A 40	PUSH 40	
0002A25D	68 00100000	PUSH 1000	
0002A262	FFB5 64D7FFFF	PUSH DWORD PTR SS:[EBP-289C]	
0002A268	FF35 B0AFD400	PUSH DWORD PTR DS:[04AFB0]	
0002A26E	FF15 8C61D300	CALL DWORD PTR DS:[D3618C]	kernel32.VirtualAlloc
0002A274	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX	
0002A27A	83BD 6CD7FFFF 00	CMP DWORD PTR SS:[EBP-2894],0	
0002A281	7E 0F84 9C000000	JE 0002A323	
0002A287	833D B0AFD400 00	CMP DWORD PTR DS:[04AFB0],0	
0002A28E	74 20	JE SHORT 0002A2B0	
0002A290	6A 01	PUSH 1	
0002A292	58	POP EAX	00D40A58
0002A293	85C0	TEST EAX,EAX	
0002A295	74 09	JE SHORT 0002A2A0	
0002A297	83A5 0CA8FFFF 00	AND DWORD PTR SS:[EBP+FFFA80C],0	
0002A29E	EB 1B	JMP SHORT 0002A2BB	

IN _See FPU Register:


```

Registers (FPU)
EAX 03500000
ECX 77E7AC6F kernel32.77E7AC6F
EDX 7FFE0304
EBX 00000000
ESP 001292B4
EBP 0012EBB0
ESI 6F44A848
EDI 0012EA58
EIP 00D2A274
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)

```

_To Pass by, EAX must change the address of the sections. ADATA. Alt + M:

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00300000	00002000				Map	R	R	
003E0000	00006000				Priv	RM	RM	
003F0000	00001000				Map	RM	RM	
00400000	00001000	cupolywz		PE header	Inag	R	RME	
00401000	001AA000	cupolywz	.text		Inag	R	RME	
00500000	0000E000	cupolywz	.data		Inag	R	RME	
00509000	00050000	cupolywz	.text1	code	Inag	R	RME	
00600000	00010000	cupolywz	.adata	SFX	Inag	R	RME	
00619000	00020000	cupolywz	.data1	data, import	Inag	R	RME	
00639000	00000000	cupolywz	.pdata		Inag	R	RME	
00700000	00000000	cupolywz			Map	R	R	
00720000	00006000	cupolywz	.rsrc	resources	Map	R E	R E	
007E0000	00002000				Map	R E	R E	
007FA000	00100000				Map	R	R	

_The Address is 609000. Back to the CPU. Set it:

Modify EAX

Hexadecimal

Signed

Unsigned

Char

Modify EAX

Hexadecimal

Signed

Unsigned

Char

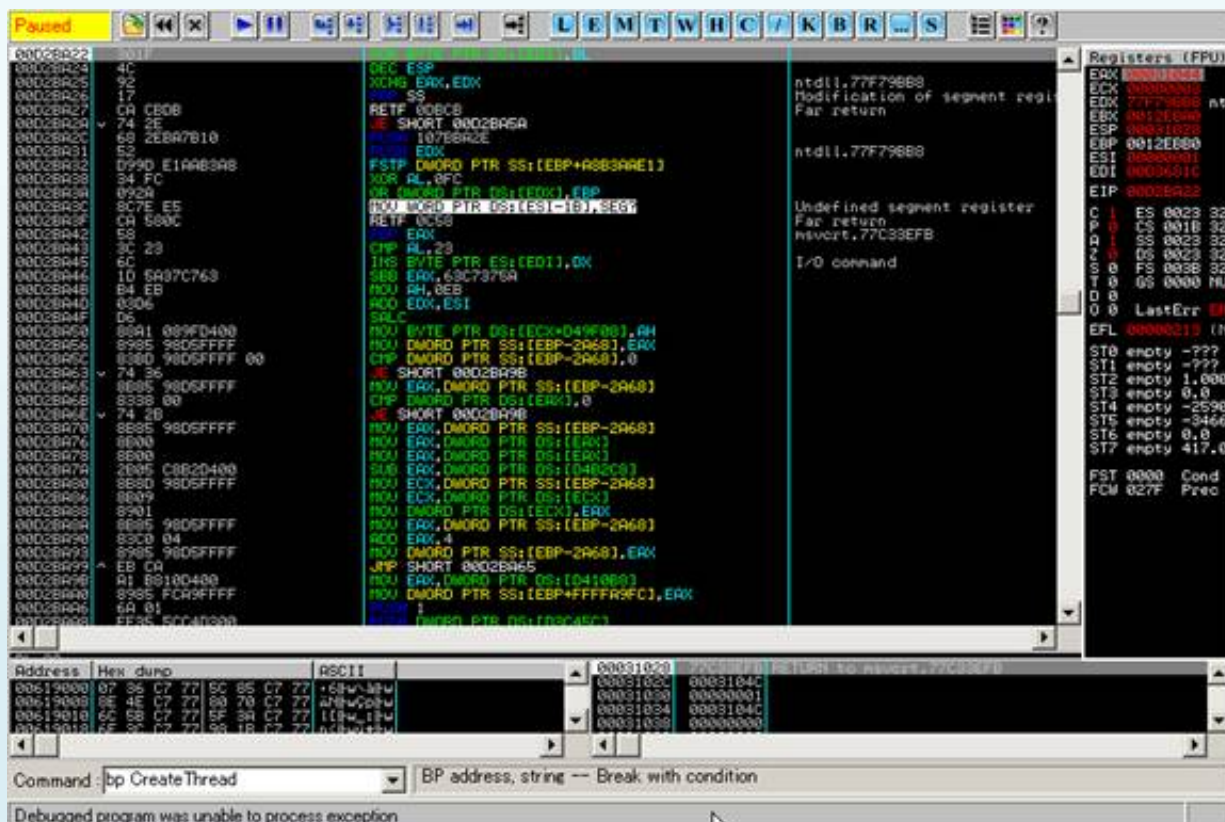
```

Registers (FPU)
EAX 00609000 cwpolywz.<ModuleEntryPoint>
ECX 77E7AC6F kernel32.77E7AC6F
EDX 7FFE0304
EBX 00000000
ESP 001292B4
EBP 0012EBB0
ESI 6F44A848
EDI 0012EA58
EIP 00D2A274
00D2A255 8985 10A8FFFF MOV DWORD PTR SS:[EBP+FFFA810],EAX
00D2A25B 6A 40 PUSH 40
00D2A25D 68 00100000 PUSH 1000
00D2A262 FFB5 6407FFFF PUSH DWORD PTR SS:[EBP-289C]
00D2A268 FF35 B0AFD400 PUSH DWORD PTR DS:[D4AFB0]
00D2A26E FF15 8C61D300 CALL DWORD PTR DS:[D3618C]
00D2A274 8985 6CD7FFFF MOV DWORD PTR SS:[EBP-2894],EAX
00D2A27A 83BD 6CD7FFFF CMP DWORD PTR SS:[EBP-2894],0
00D2A281 0F84 9C000000 JE 00D2A323
00D2A287 83BD B0AFD400 CMP DWORD PTR DS:[D4AFB0],0
00D2A28E 74 20 JE SHORT 00D2A2BD
00D2A290 6A 01 PUSH 1

```

_Now If you press Alt + M set breakpoint section of text or use BP CreateThread. You will see:

Access violation when writing to [00D3681C] - use Shift+F7/F8/F9 to pass exception to program



_Shift + F9:



_Why Crash? Because OllyDBG stack overflow in!

_Too Bad. It's crash. Hix, please Ctrl + F2 restart all task and do it again. You must repeat all step and stop when you set HE GetModuleHandleA. If you do not like press Shift + F9 many times. You can use this script:

/ * Magic Jump Finder Scripts * /

var GetModuleHandleA

GPA "GetModuleHandleA", "kernel32.dll"

mov GetModuleHandleA, \$ RESULT

bphws GetModuleHandleA, "x"

repeat:

esto

rtu

find eip, # 0F84 ?????????????????????? 74 ?????????? EB? #

Cmp \$ result, 0

je repeat

bphwc GetModuleHandleA

ret

_After Shift + F9 or use script to find magic jump, you still here:

00015AA3	8B0D 6C50D400	MOV ECX, DWORD PTR DS:[EAX+D400]	
00015AA9	89040E	MOV DWORD PTR DS:[ESI+ECX], EAX	kernel32.77E60000
00015AAC	A1 6C50D400	MOV EAX, DWORD PTR DS:[D4506C]	
00015AB1	391C06	CMP DWORD PTR DS:[ESI+EAX], EBX	
00015AB4	75 16	JNZ SHORT 00015ACC	
00015AB6	8D85 B4FEFFFF	LEA EAX, DWORD PTR SS:[EBP-14C]	
00015ABC	50	PUSH EAX	kernel32.77E60000
00015ABD	FF15 B862D300	CALL DWORD PTR DS:[D362B8]	kernel32.LoadLibraryA
00015AC3	8B0D 6C50D400	MOV ECX, DWORD PTR DS:[D4506C]	
00015AC9	89040E	MOV DWORD PTR DS:[ESI+ECX], EAX	kernel32.77E60000
00015ACC	A1 6C50D400	MOV EAX, DWORD PTR DS:[D4506C]	
00015AD1	391C06	CMP DWORD PTR DS:[ESI+EAX], EBX	
00015AD4	7E 0F84 2F010000	JE 00015C09	
00015ADA	33C9	XOR ECX, ECX	kernel32.77E7B5E1
00015ADC	8B07	MOV EAX, DWORD PTR DS:[EDI]	
00015ADE	3918	CMP DWORD PTR DS:[EAX], EBX	
00015AE0	74 06	JE SHORT 00015AE8	
00015AE2	41	INC ECX	kernel32.77E7B5E1
00015AE3	83C0 0C	ADD EAX, 0C	
00015AE6	EB F6	JMP SHORT 00015ADE	kernel32.77E7B5E1
00015AE8	8BD9	MOV EBX, ECX	kernel32.77E7B5E1

_Patch Magic jump. Then he Virtual Alloc, Shift + F9, Alt + F9 one time!

77E7AC72	55	PUSH EBP	
77E7AC73	8BEC	MOV EBP, ESP	
77E7AC75	FF75 14	PUSH DWORD PTR SS:[EBP+14]	
77E7AC78	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
77E7AC7B	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]	
77E7AC7E	FF75 08	PUSH DWORD PTR SS:[EBP+08]	
77E7AC81	6A FF	PUSH -1	
77E7AC83	E8 9CFFFFFF	CALL kernel32.VirtualAllocEx	
77E7AC88	5D	POP EBP	73421D07
77E7AC89	C2 1000	RETN 10	
77E7AC8C	8B4424 04	MOV EAX, DWORD PTR SS:[ESP+4]	
77E7AC90	85C0	TEST EAX, EAX	
77E7AC92	7E 0F84 B7010000	JE kernel32.77E7AE4F	
77E7AC98	A8 01	TEST AL, 1	

Registers (FPU)

```

EAX 01390000
ECX 77E7AC6F kernel32.77E7AC6F
EDX 77E7AC64
EBX 73421D07
ESP 00128904
EBP 7352B030
ESI 00000001
EDI 77E7AC72 kernel32.VirtualAllocEx
EIP 73421D07
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)

```

001288F0	73421D07	CALL to VirtualAlloc from 73421D05	
001288F4	00000000	Address = NULL	
001288F8	00400000	Size = 400000 (4194304.)	
001288FC	00002000	AllocationType = MEM_RESERVE	
00128900	00000004	Protect = PAGE_READWRITE	
00128904	00000000		
00128908	00000001		
0012890C	00128954		
00128910	73421CCD	RETURN to 73421CCD from 73421CD8	
00128914	73421BCE	RETURN to 73421BCE from 73421CAC	
00128918	73421B50	RETURN to 73421B50 from 73421B7C	
0012891C	73420000		
00128920	00000001		
00128924	00000000		

Window _Go to register and change the EAX to 609,000.

Modify EAX

Hexadecimal

609000

Signed

6328320

Unsigned

6328320

Char

\x00

\x90

\x00

OK

Cancel

```

Registers (FPU)
EAX 00609900 cwpolywz.<ModuleEntryPoint>
ECX 77F59A7B ntdll.77F59A7B
EDX 00001800
EBX 73420000
ESP 001286F0
EBP 7352B0C0
ESI 00000001
EDI 77E7AC72 kernel32.VirtualAlloc
EIP 77E7AC72 kernel32.VirtualAlloc

C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_PROC_NOT_FOUND (0000007F)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty -??? FFFF 00000000 FF010101
ST1 empty -??? FFFF 00000000 FF010101
ST2 empty -??? FFFF 00000000 00000000
ST3 empty -??? FFFF 00000000 00000000
ST4 empty -NAN FFFF FF010101 FF010101
ST5 empty -??? FFFF 00000000 00000000
ST6 empty -??? FFFF 00000000 00000000
ST7 empty 417.000000000000000000000000

FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 (GT)
FCW 037F Prec NEAR,64 Mask 1 1 1 1 1 1

```

_Hd VirtualAlloc. BP CreateThread. Shift + F9. Oh, yeahhhhhhhhhh, not crash!

77E7BE53	55	PUSH EBP	
77E7BE54	8BEC	MOV EBP,ESP	
77E7BE56	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]	cwpolywz.005E1B9
77E7BE59	FF75 18	PUSH DWORD PTR SS:[EBP+18]	
77E7BE5C	FF75 14	PUSH DWORD PTR SS:[EBP+14]	
77E7BE5F	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
77E7BE62	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]	
77E7BE65	FF75 08	PUSH DWORD PTR SS:[EBP+08]	
77E7BE68	6A FF	PUSH -1	
77E7BE6A	E8 30FEFFFF	CALL kernel32.CreateRemoteThread	
77E7BE6F	5D	POP EBP	00D1BFF9
77E7BE70	C2 1800	RETN 18	

```

0012F6D4 00D1BFF9 CALL to CreateThread from 00D1BFF9
0012F6D8 00000000 pSecurity = NULL
0012F6DC 00000000 StackSize = 0
0012F6E0 00D1C8B9 ThreadFunction = 00D1C8B9
0012F6E4 00000000 pThreadParam = NULL
0012F6E8 00000000 CreationFlags = 0
0012F6EC 0012F6F8 pThreadId = 0012F6F8
0012F6F0 00000004
0012F6F4 00621098 cwpolywz.00621098
0012F6F8 00000001
0012F6FC 0012F714
0012F700 00D30358 RETN to 00D30358 from 00D1BF51
0012F704 00000000
0012F708 00000004

```

_Ctrl + F9, F8:

00D1BFF9	5F	POP EDI	
00D1BFFA	5E	POP ESI	
00D1BFFB	C9	LEAVE	
00D1BFFC	C3	RETN	
00D1BFFD	55	PUSH EBP	
00D1BFFE	8BEC	MOV EBP,ESP	
00D1C000	81EC 28010000	SUB ESP,128	
00D1C006	56	PUSH ESI	
00D1C007	57	PUSH EDI	
00D1C008	BE E4BDD300	MOV ESI,0D3BDE4	ASCII "MainClass"
00D1C00D	8DBD D8FEFFFF	LEA EDI,DWORD PTR SS:[EBP-128]	

_Ctrl + F9, F8 again:

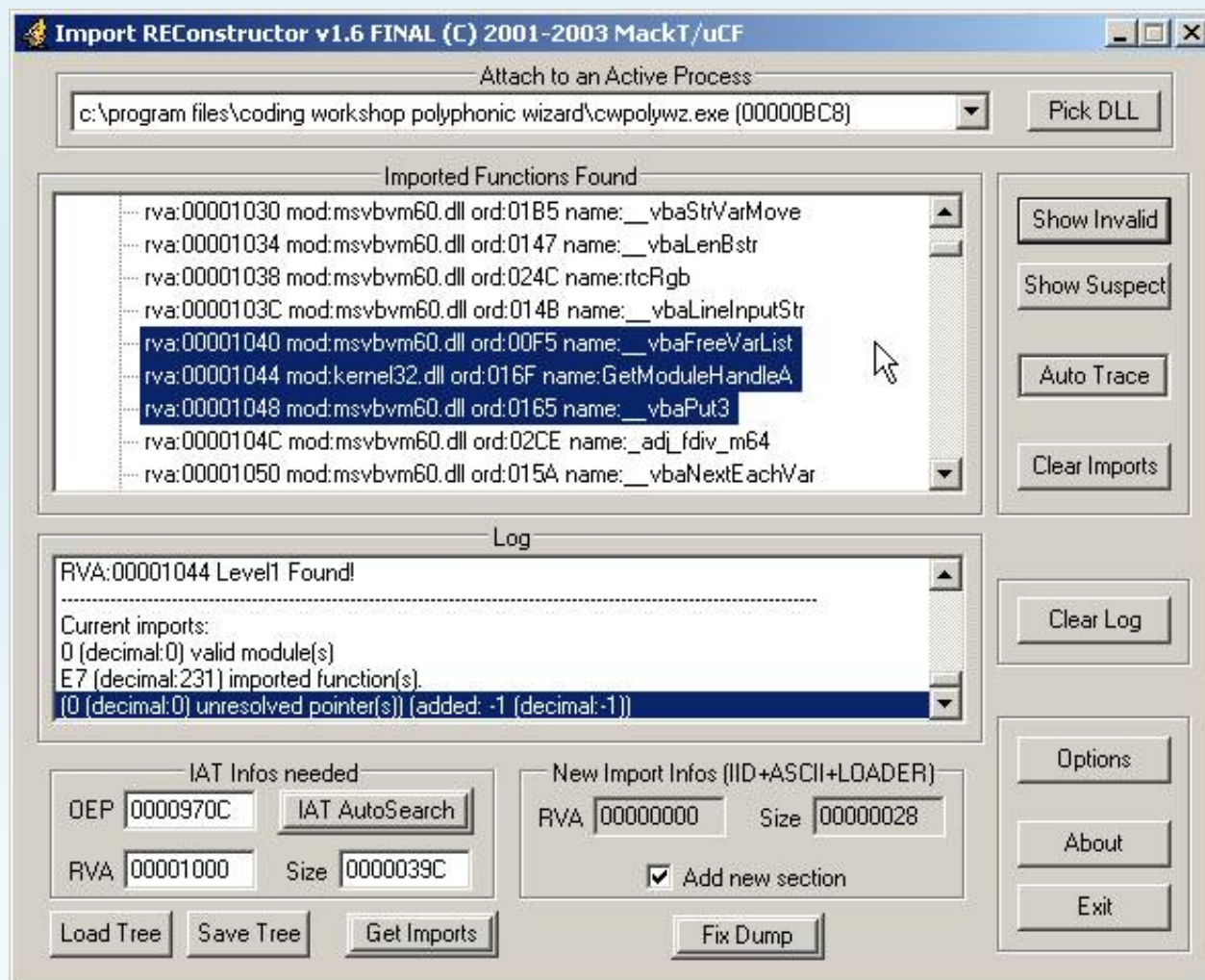
00D30358	59	POP ECX	kernel32.77E7BE2
00D30359	BF 58AD400	MOV EDI,0D40A58	
00D3035E	8BCF	MOV ECX,EDI	
00D30360	E8 9B7DFDFF	CALL 00D08100	
00D30365	84C0	TEST AL,AL	
00D30367	75 09	JNZ SHORT 00D30372	
00D30369	6A 01	PUSH 1	
00D3036B	8BCF	MOV ECX,EDI	
00D3036D	E8 77D3FDFF	CALL 00D0D6E9	

_Scroll Down:

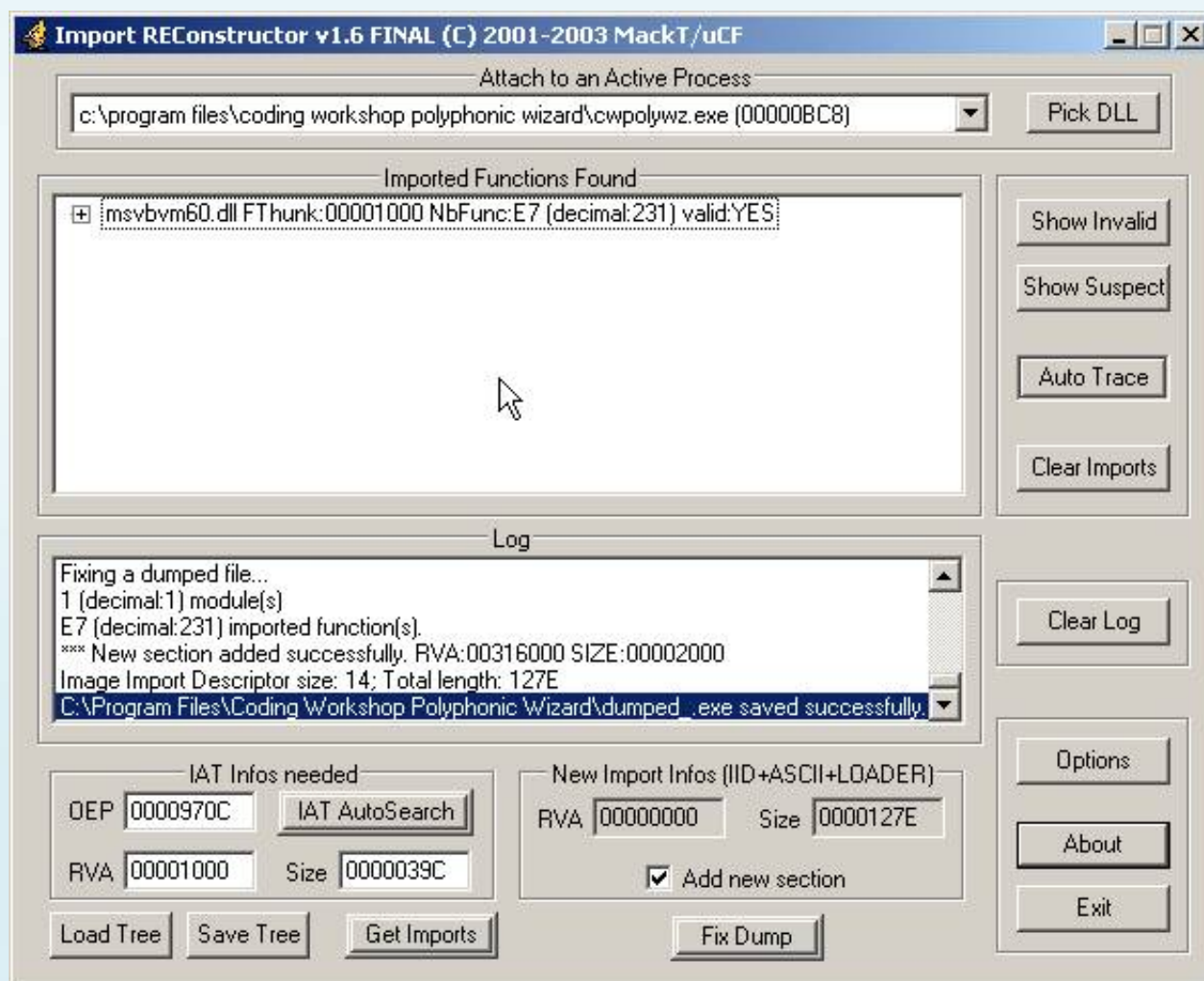
0040970C	68 7C284200	PUSH cwpolywz.0042287C	
00409711	E8 EFFFFFFF	CALL cwpolywz.00409704	
00409716	0000	ADD BYTE PTR DS:[EAX],AL	
00409718	0000	ADD BYTE PTR DS:[EAX],AL	
0040971A	0000	ADD BYTE PTR DS:[EAX],AL	
0040971C	3000	XOR BYTE PTR DS:[EAX],AL	
0040971E	0000	ADD BYTE PTR DS:[EAX],AL	
00409720	68 00000040	PUSH 40000000	
00409725	0000	ADD BYTE PTR DS:[EAX],AL	
00409727	0036	ADD BYTE PTR DS:[ESI],DH	
00409729	5A	POP EDX	00D30436
0040972A	AA	STOS BYTE PTR ES:[EDI]	
0040972B	99	CQ	
0040972C	6E	OUTS DX, BYTE PTR ES:[EDI]	I/O command
0040972D	8399 4B805FD6 A9	SBB DWORD PTR DS:[ECX+D65F8D4B],-57	
00409734	891B	MOV DWORD PTR DS:[EBX],EBX	cwpolywz.005F662
00409736	8063 00	LEA ESP, DWORD PTR DS:[EBX]	
00409739	0000	ADD BYTE PTR DS:[EAX],AL	
0040973B	0000	ADD BYTE PTR DS:[EAX],AL	
0040973D	0001	ADD BYTE PTR DS:[ECX],AL	
0040973F	0000	ADD BYTE PTR DS:[EAX],AL	
00409741	0048 00	ADD BYTE PTR DS:[EAX],CL	
00409744	06	PUSH ES	
00409745	50	PUSH EAX	cwpolywz.0061960
00409746	8302 50	ADD DWORD PTR DS:[EDX],50	
00409749	6F	OUTS DX, DWORD PTR ES:[EDI]	I/O command
0040974A	6C	INS BYTE PTR ES:[EDI],DX	I/O command
0040974B	79 5F	JNS SHORT cwpolywz.004097AC	
0040974D	57	PUSH EDI	cwpolywz.0040100

c:\program files\emeditor\emeditor.exe	0000056C	00400000	00069000
c:\progra~1\systran\5.0\premium\sysra~3.exe	00000868	00400000	00F5A000
c:\program files\microsoft office\office11\win...	00000AF0	30000000	00BA0000
d:\soft need\unikey 3.5\unikey.exe	00000B2C	00400000	0002F000
i:\cracker\debug- disassembler\odbg110_org...	00000BC0	00400000	00169000
c:\program files\coding workshop poly...	dump full...		00316000
c:\program files\techsmith\snagit 7\sn...	dump partial...		00286000
c:\program files\techsmith\snagit 7\ts...	dump region...		0000A000
c:\program files\winamp\winamp.exe			00121000
	active dump engine		00000000
Path	priority		

file:///C:/RCE%20Unpacking%20eBook%20[Tran...%20by%20LithiumLi]/armdillo_tuts_7_exp.htm (10 of 15) [1/9/2009 9:44:05 LithiumLi]



_This Target code in VB6. And the import function GetModuleHandleA is invalid. Change to __vbaEnd. Cuts thunks.



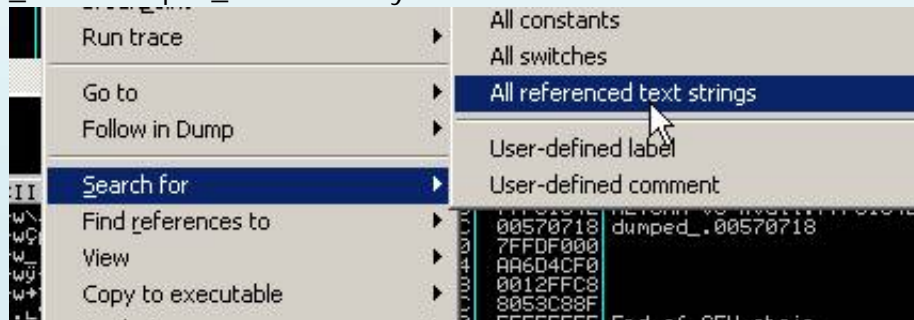
_Run:



_Unpacked Successful!

III. Cracking

Load Dumped.exe into OllyDBG:



_Search Some info needed:

Address	Disassembly	Text string
004F3290	PUSH dumped_.0043BE7C	UNICODE "Options"
004F34B7	PUSH dumped_.0043B8C4	UNICODE "Open"
004F350D	MOV DWORD PTR SS:[EBP-40], dumped_.0043B8C4	UNICODE "Start preview"
004F3563	MOV DWORD PTR SS:[EBP-40], dumped_.0043B8C4	UNICODE "Start preview"
004F396E	MOV DWORD PTR SS:[EBP-DC], dumped_.0043C80C	UNICODE "7601 single developer license"
004F3A37	MOV EDI, dumped_.0043C190	UNICODE "1 September 2005"
004F3F70	MOV EDI, dumped_.0043C7A0	UNICODE "Coding Workshop Polyphonic Wizard v4"
004F3FEA	PUSH dumped_.0043C80C	UNICODE " : Development Copy"
004F406F	PUSH dumped_.0043C838	UNICODE "Version: "
004F416C	PUSH dumped_.0043C850	UNICODE "Build date: "
004F41F7	PUSH dumped_.0043C870	UNICODE "Coding Workshop"
004F424D	PUSH dumped_.0043C894	UNICODE "Polyphonic Wizard"
004F456D	PUSH dumped_.0043C8DC	UNICODE "TM"
004F45C3	PUSH dumped_.0043C8E8	UNICODE "EXPIRED"
004F47E7	PUSH dumped_.0043C948	UNICODE "OK"
004F48D5	PUSH dumped_.0043C954	UNICODE "Registered Software "
004F4988	PUSH dumped_.0043C9B0	UNICODE "USERNAME"
004F49AB	PUSH dumped_.0043C984	UNICODE "Registered copy - "
004F4B4D	MOV DWORD PTR SS:[EBP-DC], dumped_.0043C984	UNICODE "Status: Registered"
004F4D02	PUSH dumped_.0043C9F4	UNICODE "This software has not been registered (trial remaining:"
004F4D2E	PUSH dumped_.0043CA68	UNICODE " days)."
004F4E07	MOV DWORD PTR SS:[EBP-EC], dumped_.0043C9F4	UNICODE "Trial version in use for "
004F4E42	MOV DWORD PTR SS:[EBP-FC], dumped_.0043C9F4	UNICODE " day(s)."
004F4F11	MOV DWORD PTR SS:[EBP-DC], dumped_.0043C9F4	UNICODE "Status: Trial Copy"
004F4FFF	PUSH dumped_.0043CAF4	ASCII 03."α="

Set A breakpoint at function __vbastrcmp. Press F9:

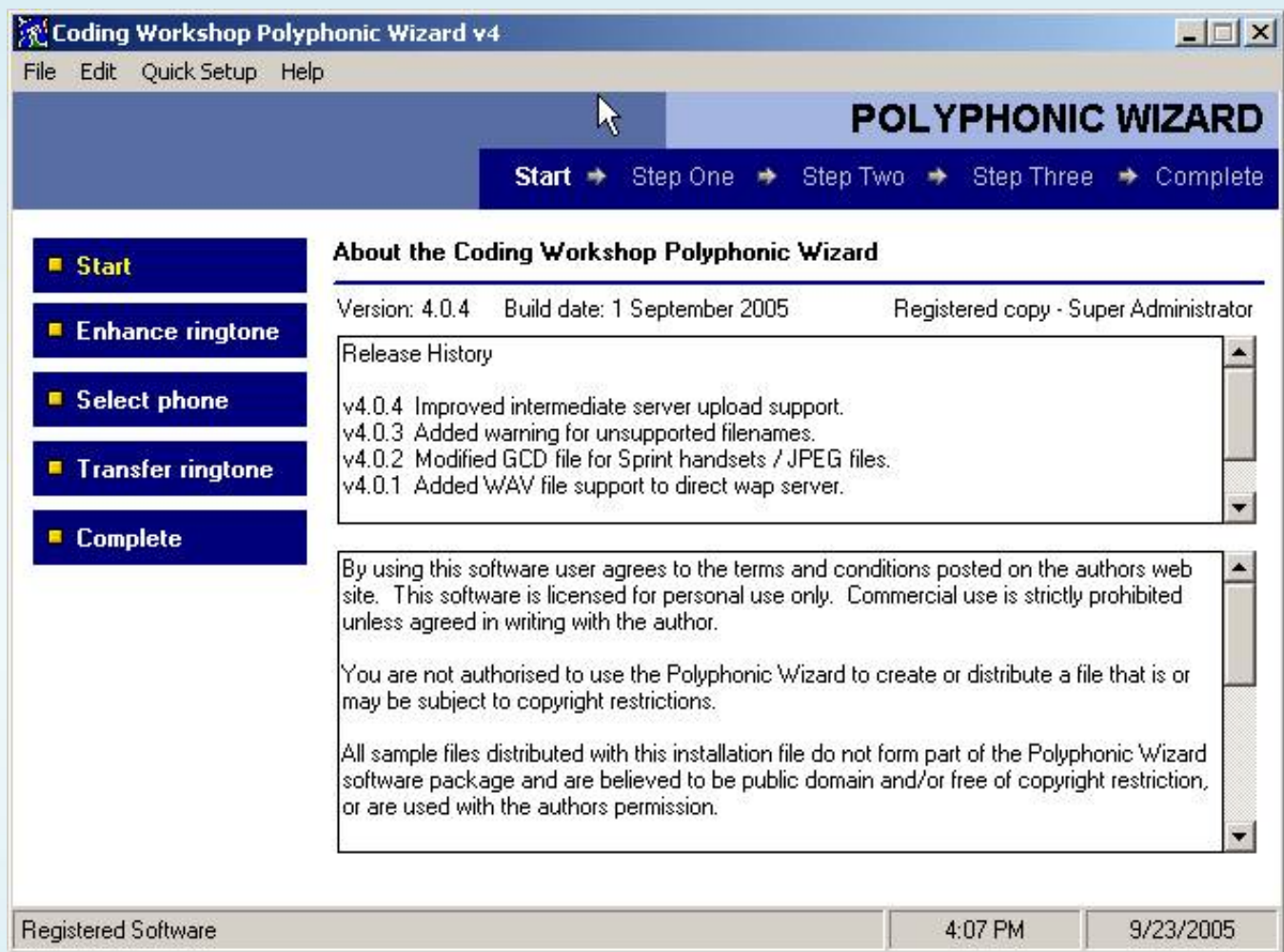
004F47D6	D7	XLAT BYTE PTR DS:[EBX+AL]	
004F47D7	00E8	ADD AL,CH	
004F47D9	6B4D F1 FF	IMUL ECX,DWORD PTR SS:[EBP-F],-1	
004F47DD	C745 FC 28000000	MOV DWORD PTR SS:[EBP-4],28	
004F47E4	FF75 D8	PUSH DWORD PTR SS:[EBP-28]	
004F47E7	68 48C94300	PUSH dumped_.0043C948	UNICODE "OK"
004F47EC	E8 A34CF1FF	CALL <JMP.&msvbvm60.__vbaStrCmp>	
004F47F1	85C0	TEST EAX,EAX	
004F47F3	0F85 FC030000	JNZ dumped_.004F4BF5	
004F47F9	C745 FC 29000000	MOV DWORD PTR SS:[EBP-4],29	
004F4800	90	NOP	
004F4801	90	NOP	
004F4802	90	NOP	
004F4803	90	NOP	
004F4804	90	NOP	
004F4805	E8 3E4DF1FF	CALL <JMP.&msvbvm60.__vbaSetSystemError>	
004F480A	C745 FC 2A000000	MOV DWORD PTR SS:[EBP-4],2A	
004F4811	68 E0BD4300	PUSH dumped_.0043BDE0	

Change JNZ to JZ.

004F47D6	D7	XLAT BYTE PTR DS:[EBX+AL]	
004F47D7	00E8	ADD AL,CH	
004F47D9	6B4D F1 FF	IMUL ECX,DWORD PTR SS:[EBP-F],-1	
004F47DD	C745 FC 28000000	MOV DWORD PTR SS:[EBP-4],28	
004F47E4	FF75 D8	PUSH DWORD PTR SS:[EBP-28]	
004F47E7	68 48C94300	PUSH dumped_.0043C948	UNICODE "OK"
004F47EC	E8 A34CF1FF	CALL <JMP.&msvbvm60.__vbaStrCmp>	
004F47F1	85C0	TEST EAX,EAX	
004F47F3	0F84 FC030000	JZ dumped_.004F4BF5	
004F47F9	C745 FC 29000000	MOV DWORD PTR SS:[EBP-4],29	
004F4800	90	NOP	
004F4801	90	NOP	
004F4802	90	NOP	
004F4803	90	NOP	
004F4804	90	NOP	
004F4805	E8 3E4DF1FF	CALL <JMP.&msvbvm60.__vbaSetSystemError>	
004F480A	C745 FC 2A000000	MOV DWORD PTR SS:[EBP-4],2A	
004F4811	68 E0BD4300	PUSH dumped_.0043BDE0	

Done:





IV. Conclusion

Bye!

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtlN, hosiminh, Nilrem, fly, Madman_Hercules, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RIF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx OllyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 23/09/2005)

Armadillo collect sand-stone

Picture Ripper 3: Armadillo 4.xx-Import + Nanomites Complete elimination tut!

I. Intro:

_Salut Tout le monde, now in the Net, I found some software was packed with options and Import Code Splicing elimination. And we can not unpack because the import was make in memory, I call this IAT is: Virtual IAT. Armadillo make Memory in the IAT and software to redirect. So, we can not use this because when IAT fix dump our dumped file, imprec show a dialog:



_It's A blessing when Admirallo make a very very good tools is: ArmaInline (the newest version is 0.6). ArmaInline appear before, I was found just one method redirect the virtual IAT by **Ricardo Narvaja** in the tuts ([203-ARMADILLO CON DESTRUCCION DE TABLA parte 1](#) R [208-ARMADILLO CON DESTRUCCION DE TABLA Y FINAL PARTE 6](#)) with the target HyperSnap-DX.v5.60! But think in this method only for pro and not for beginner!

_In Exetools Forum, I was published an article called: Code Splicing + IAT elimination. Sorry my blunder! And now, I complete a written tut for this options. I'll do my best to make this tut become very easy for every body! Ok, the end of Intro. Go go go!

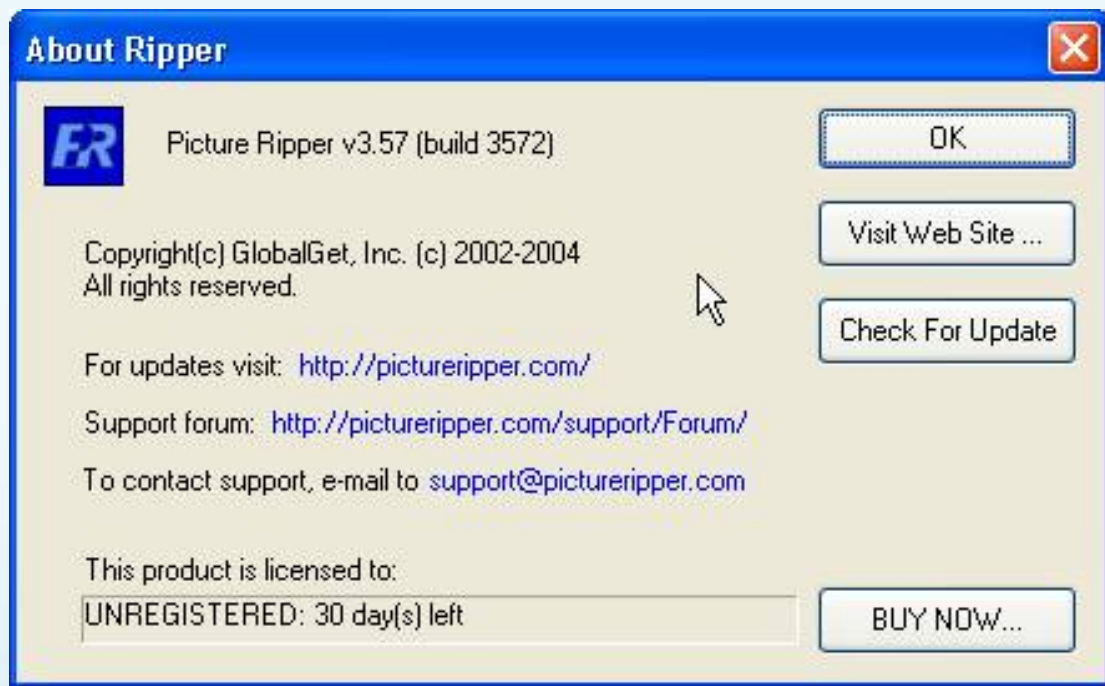
II. Tools

- 1.OllyDBG - The best config debugger for ArmMUP by hacnho.
- 2.LordPE 1.4 Deluxe
- 3.Import REConstructor 1.6 Final
- 4.ArmaInline 0.71

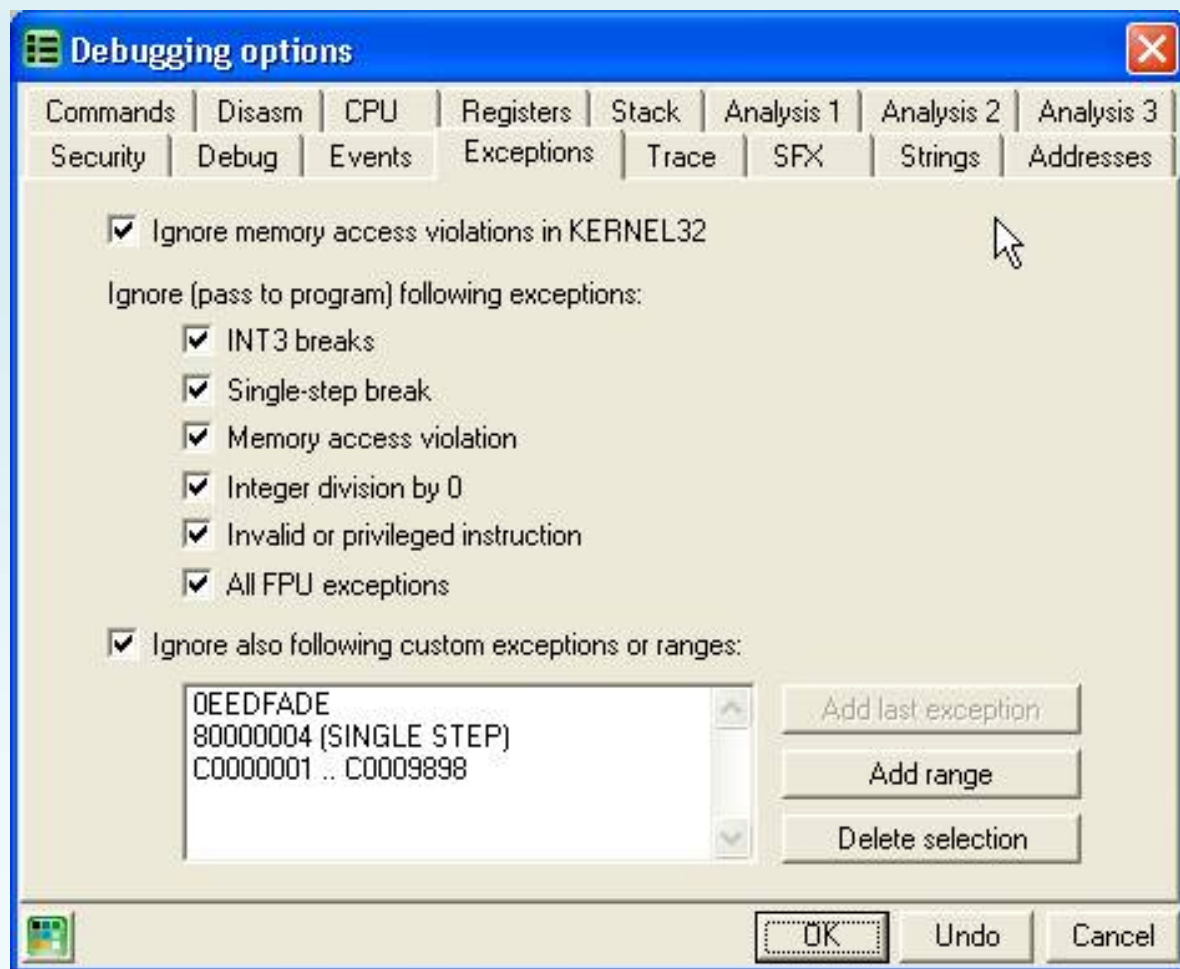
III. Unpacking

Target: Picture Ripper 3:57 build 3572

ARM Debug 4.xx-Blocker IAT elimination + + + Anti Breakpoint Nanomites



_Setting OllyDBG:



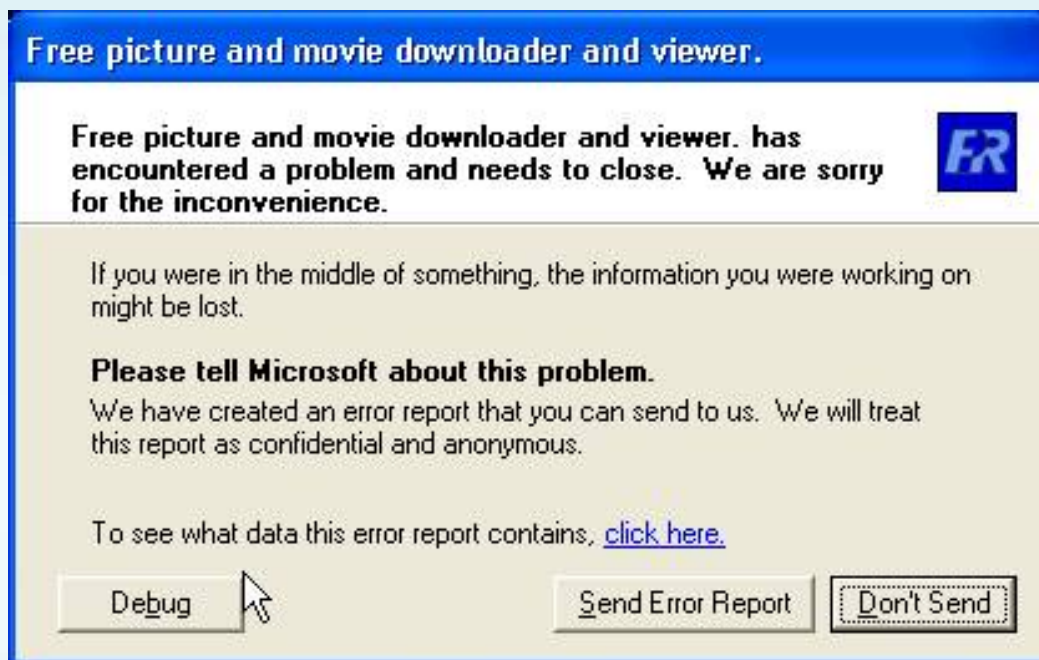
_Detect Target:

	c:\program files\messenger\msmsgs.exe	00000FC4	00400000
	c:\program files\pictureripper 3\pictureripper....	00000724	00400000
	c:\program files\pictureripper 3\pictureripper....	000000EC	00400000
	i:\cracker\utilities\lordpe deluxe - 1.4\lordpe....	00000F24	00400000

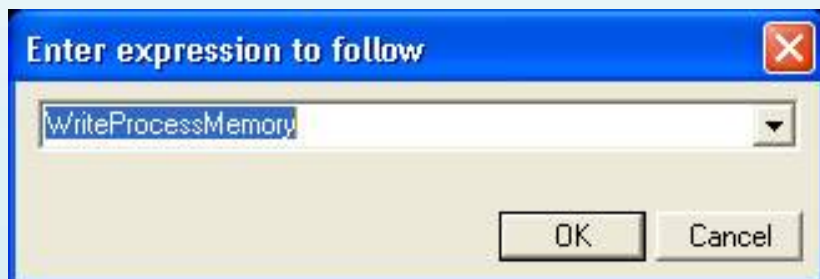
_Ok, Load into OllyDBG target:

Address	Hex dump	Disassembly	Comment
0059B000	60	ENTER	
0059B001	E8 00000000	CALL PictureR.0059B006	
0059B006	50	POP EBP	
0059B007	50	PUSH EAX	
0059B008	51	PUSH ECX	
0059B009	0FCA	BSRMP EDX	
0059B00B	F702	NOT EDX	
0059B00D	9C	PAUSE	
0059B00E	F702	NOT EDX	
0059B010	0FCA	BSRMP EDX	
0059B012	EB 0F	JMP SHORT PictureR.0059B023	
0059B014	B9 EB0FB8EB	MOV ECX,EBB80FEB	
0059B019	07	POP ES	Modification of segment regist

_bp WriteProcessMemory, Shift + F9:



_It's Crash. Oh, I think this target anti-bp! Ok. We must pass by the debug option by hand blocker. Ctrl + F2 restart target. Ctrl + G: WriteProcessMemory:



F2 _Press set a breakpoint at:

Address	Hex dump	Disassembly	Comment
77E61A94	55	PUSH EBP	
77E61A95	88EC	MOV EBP,ESP	
77E61A97	51	PUSH ECK	
77E61A98	51	PUSH ECK	
77E61A99	8845 0C	MOV EAX,DWORD PTR SS:[EBP+C]	
77E61A9C	53	PUSH EBX	
77E61A9D	885D 14	MOV EBX,DWORD PTR SS:[EBP+14]	
77E61AA0	56	PUSH ESI	
77E61AA1	8835 B012E677	MOV ESI,DWORD PTR DS:[&ntdll.NtProtectVirtualMemory]	ntdll.ZwProtectVirtualMemory
77E61AA7	57	PUSH EDI	
77E61AA8	887D 08	MOV EDI,DWORD PTR SS:[EBP+8]	
77E61AAB	8945 F8	MOV DWORD PTR SS:[EBP-8],EAX	

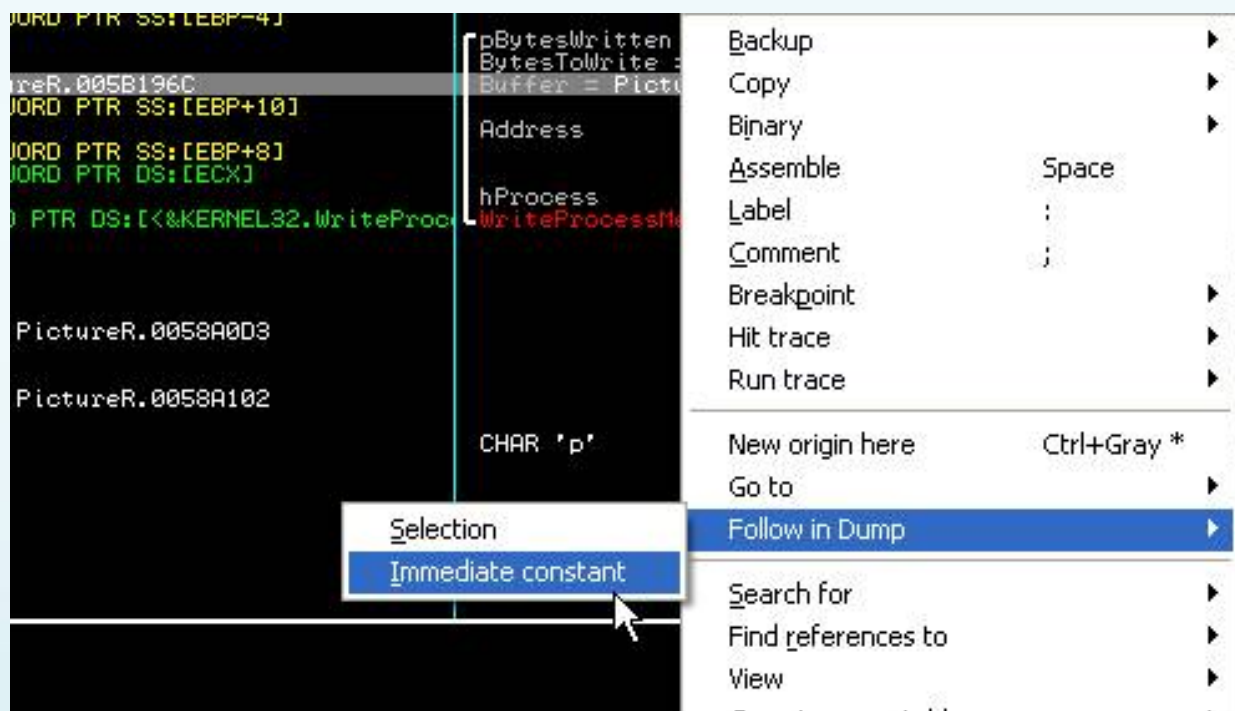
_Shift + F9, yeah. J OllyDBG break. Now, press Alt + F9:

Address	Hex dump	Disassembly	Comment
0058A0A3	<70 07	JL SHORT PictureR.0058A0AC	
0058A0A5	<7C 03	JL SHORT PictureR.0058A0AA	
0058A0A7	<EB 05	JMP SHORT PictureR.0058A0AE	
0058A0A9	E8 74FBEBF9	CALL FA449C22	
0058A0AE	<EB 5F	JMP SHORT PictureR.0058A10F	
0058A0B0	8055 FC	LEA EDX,DWORD PTR SS:[EBP-4]	
0058A0B3	52	PUSH EDX	
0058A0B4	6A 02	PUSH 2	
0058A0B6	68 6C195B00	PUSH PictureR.005B196C	
0058A0B8	8845 10	MOV EAX,DWORD PTR SS:[EBP+10]	
0058A0BE	50	PUSH EAX	
0058A0BF	884D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
0058A0C2	8B11	MOV EDX,DWORD PTR DS:[ECX]	
0058A0C4	52	PUSH EDX	
0058A0C5	FF15 FCB05A00	CALL DWORD PTR DS:[&kernel32.WriteProcessMemory]	kernel32.WriteProcessMemory

_Nothing Interest. Ok, Ctrl + A:

Address	Hex dump	Disassembly	Comment
0058A0A3	<70 07	JL SHORT PictureR.0058A0AC	
0058A0A5	<7C 03	JL SHORT PictureR.0058A0AA	
0058A0A7	>EB 05	JMP SHORT PictureR.0058A0AE	
0058A0A9	E8	DB E8	
0058A0AA	>74 FB	JE SHORT PictureR.0058A0A7	
0058A0AC	>EB F9	JMP SHORT PictureR.0058A0A7	
0058A0AE	>EB 5F	JMP SHORT PictureR.0058A10F	
0058A0B0	> 8055 FC	LEA EDX,DWORD PTR SS:[EBP-4]	
0058A0B3	. 52	PUSH EDX	
0058A0B4	. 6A 02	PUSH 2	
0058A0B6	. 68 6C195B00	PUSH PictureR.005B196C	
0058A0B8	. 8845 10	MOV EAX,DWORD PTR SS:[EBP+10]	
0058A0BE	. 50	PUSH EAX	
0058A0BF	. 884D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
0058A0C2	. 8B11	MOV EDX,DWORD PTR DS:[ECX]	
0058A0C4	. 52	PUSH EDX	
0058A0C5	. FF15 FCB05A00	CALL DWORD PTR DS:[&kernel32.WriteProcessMemory]	WriteProcessMemory
0058A0C8	. 50	PUSH EAX	

_Yeah! Ok, then right click at the buffer and follow in dump> Immediate Constant:



_Look Dump in Windows:

Address	Hex dump	ASCII
005B196C	60 E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00	'.'.....
005B197C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B198C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B199C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19AC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19BC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19CC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19DC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19EC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

_Next, Ctrl + E: 60E8 to change EBFE:

Address	Hex dump	ASCII
005B196C	EB FE 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B197C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B198C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B199C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19AC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19BC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19CC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19DC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19EC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

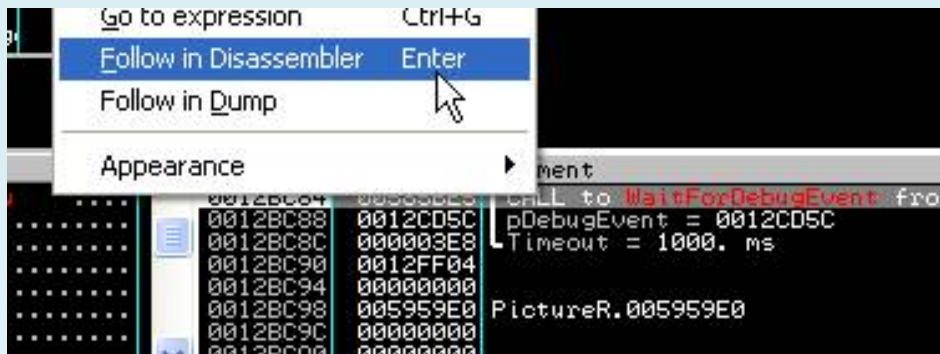
_Following, You set a breakpoint at API WaitForDebugEvent! Alt + F1: BP WaitForDebugEvent (Remember delete the breakpoint at Push ECX)! Attention, set breakpoint before WaitForDebugEvent, you must press F9 one time! If no, the child will be suspend:



_Shift + F9, look in the Stack Window:

Address	Value	Comment
0012BC84	00585BE5	CALL to WaitForDebugEvent from
0012BC88	0012CD5C	pDebugEvent = 0012CD5C
0012BC8C	000003E8	Timeout = 1000. ms
0012BC90	0012FF04	
0012BC94	00000000	
0012BC98	005959E0	PictureR.005959E0
0012BC9C	00000000	
0012BCA0	00000000	
0012BCA4	00000000	

Disassembler _Follow in simple or press Ctrl + F9, F8



_You Still here:

Address	Hex dump	Disassembly	Comment
00585BE5	. 85C0	TEST EAX, EAX	
00585BE7	✓ 0F84 23270000	JE PictureR.00588310	
00585BE0	. 8B95 FCFDFFFF	MOV EDX, DWORD PTR SS:[EBP-204]	
00585BF3	. 81E2 FF000000	AND EDX, 0FF	
00585BF9	. 85D2	TEST EDX, EDX	
00585BFB	✓ 74 12	JE SHORT PictureR.00585C0F	
00585BFD	. A1 781A5B00	MOV EAX, DWORD PTR DS:[5B1A78]	
00585C02	. 8378 20 00	CMPL DWORD PTR DS:[EAX+20], 0	
00585C06	✓ 74 07	JE SHORT PictureR.00585C0F	
00585C08	. C685 FCFDFFFF	MOV BYTE PTR SS:[EBP-204], 0	
00585C0F	> 68 70195B00	PUSH PictureR.005B1970	pCriticalSection = PictureR.0
00585C14	. FF15 A0B15A00	CALL DWORD PTR DS:[&KERNEL32.EnterCrit	EnterCriticalSection
00585C1A	. 60	PUSHAD	
00585C1B	. 33C0	XOR EAX, EAX	

_Look Attach the Window:

00000704	WINWORD	CiceroUIWndFrame	C:\Program Files\Microsoft Office\OFFICE11\WINWORD.E
00000A08	msmsgs	ActiveMovie Window	C:\Program Files\Messenger\msmsgs.exe
00000A20	PictureR		C:\Program Files\PictureRipper 3\PictureRipper.exe
00000A50	PictureR		C:\Program Files\PictureRipper 3\PictureRipper.exe
00000AFC	MTDSEVER	Socket Notification Sink	C:\Program Files\mtd2002\MTDSEVER.EXE
00000B20	PictureR		C:\Program Files\PictureRipper 3\PictureRipper.exe

_Close Attach the Window, to patch:

Address	Hex dump	Disassembly	Comment
00585BE5	68 500A0000	PUSH 0A50	
00585BEA	E8 44969277	CALL kernel32.DebugActiveProcessStop	
00585BEF	90	NOP	
00585BF0	90	NOP	
00585BF1	90	NOP	
00585BF2	90	NOP	
00585BF3	. 81E2 FF000000	AND EDX, 0FF	

_Press F8, trace down:

Address	Hex dump	Disassembly	Comment	Registers (FPU)
00585BE5	58 50000000	PUSH 0058		EAX 00000001
00585BEA	58 44965277	CALL kernel32.DebugActiveProcessStop		ECX 00120074
00585BF7	58	NOP		EDX 7FFE0304
00585BF8	58	NOP		EBX 005959E0 Pl
00585BF1	58	NOP		ESP 00120090
00585BF2	58	NOP		EBP 0012077C
00585BF3	58 81E2 FF000000	AND EDX,0FF		ESI 00002710

_Done! By Debug pass blocker successful! Continued, Fire Up a OllyDBG, Attach the child, F9, F12:

Address	Hex dump	Disassembly	Comment
0059B000	5B FE	JMP SHORT PictureR.<ModuleEntryPoint>	
0059B002	0000	ADD BYTE PTR DS:[EAX],AL	
0059B004	0000	ADD BYTE PTR DS:[EAX],AL	
0059B006	5D	POP EBP	
0059B007	5D	PUSH EAX	
0059B008	51	PUSH ECX	
0059B009	0FCA	BSWAP EDX	
0059B00A	5700	NOT EBP	

_Change EBFE to 60E8:

Address	Hex dump	Disassembly	Comment
0059B000	5B	PUSHAD	
0059B001	E8 00000000	CALL PictureR.0059B006	
0059B006	5D	POP EBP	
0059B007	5D	PUSH EAX	
0059B008	51	PUSH ECX	
0059B009	0FCA	BSWAP EDX	
0059B00A	F702	NOT EDX	

_The Next step is magic patch jump and find OEP. Ok, now, press Alt + F1: He GetModuleHandleA, Shift + F9 first time:

Address	Value	Comment
00127B70	00ED19CC	CALL to GetModuleHandleA from
00127B74	00EE6364	pModule = "kernel32.dll"
00127B78	00EE7588	ASCII "VirtualAlloc"
00127B7C	00000001	
00127B80	00F40090	
00127B84	00000000	
00127B88	00000000	
00127B8C	00000000	

_ 2 nd:

Address	Value	Comment
00127B70	00ED19E9	CALL to GetModuleHandleA from
00127B74	00EE6364	pModule = "kernel32.dll"
00127B78	00EE757C	ASCII "VirtualFree"
00127B7C	00000001	
00127B80	00F40090	
00127B84	00000000	
00127B88	00000000	
00127B8C	00000000	
00127B90	00000000	

_3 Rd:

Address	Value	Comment
001278E0	00EB9BA7	CALL to GetModuleHandleA from
001278E4	00127A24	pModule = "kernel32.dll"
001278E8	00000000	
001278EC	CE300000	
001278F0	41F80012	
001278F4	00EE57C4	
001278F8	00000000	
001278FC	00000000	
00127900	00000000	

_Trace Down F8 with the Return and you here at:

Address	Hex dump	Disassembly	Comment
00EB9BA7	8B0D 74B7EE00	MOV ECX,DWORD PTR DS:[EBB774]	
00EB9BAD	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00EB9BB0	A1 74B7EE00	MOV EAX,DWORD PTR DS:[EBB774]	
00EB9BB5	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00EB9BB8	75 16	JNZ SHORT 00EB9BD0	
00EB9BBA	8D85 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
00EB9BC0	50	PUSH EAX	
00EB9BC1	FF15 DC00EE00	CALL DWORD PTR DS:[EE00DC]	kernel32.LoadLibraryA
00EB9BC7	8B0D 74B7EE00	MOV ECX,DWORD PTR DS:[EBB774]	
00EB9BCD	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00EB9BD0	A1 74B7EE00	MOV EAX,DWORD PTR DS:[EBB774]	
00EB9BD5	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00EB9BD8	75 16	JNZ SHORT 00EB9BD0	
00EB9BDE	33C9	XOR ECX,ECX	
00EB9BE0	8B07	MOV EAX,DWORD PTR DS:[EDI]	
00EB9BE2	3918	CMP DWORD PTR DS:[EAX],EBX	
00EB9BE4	74 06	JE SHORT 00EB9BEC	
00EB9BE6	41	INC ECX	
00EB9BE7	83C0 0C	ADD EAX,0C	
00EB9BEA	EB F6	JMP SHORT 00EB9BE2	The magic jump!

_Change It to EB:

Address	Hex dump	Disassembly	Comment
00EB9BA7	8B0D 74B7EE00	MOV ECX,DWORD PTR DS:[EBB774]	
00EB9BAD	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00EB9BB0	A1 74B7EE00	MOV EAX,DWORD PTR DS:[EBB774]	
00EB9BB5	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00EB9BB8	75 16	JNZ SHORT 00EB9BD0	
00EB9BBA	8D85 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
00EB9BC0	50	PUSH EAX	
00EB9BC1	FF15 DC00EE00	CALL DWORD PTR DS:[EE00DC]	kernel32.LoadLibraryA
00EB9BC7	8B0D 74B7EE00	MOV ECX,DWORD PTR DS:[EBB774]	
00EB9BCD	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00EB9BD0	A1 74B7EE00	MOV EAX,DWORD PTR DS:[EBB774]	
00EB9BD5	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00EB9BD8	E9 39010000	JMP EB 39010000	
00EB9BD0	50	PUSH EAX	
00EB9BDE	33C9	XOR ECX,ECX	
00EB9BE0	8B07	MOV EAX,DWORD PTR DS:[EDI]	
00EB9BE2	3918	CMP DWORD PTR DS:[EAX],EBX	

_Now We find J OEP. Hd GetModuleHandleA, BP CreateThread, Shift + F9:

Address	Value	Comment
0012D744	00EBFEE0	CALL to CreateThread from 00B
0012D748	00000000	pSecurity = NULL
0012D74C	00000000	StackSize = 0
0012D750	00EC070C	ThreadFunction = 00EC070C
0012D754	00000000	pThreadParm = NULL
0012D758	00000000	CreationFlags = 0
0012D75C	0012D764	pThreadId = 0012D764
0012D760	005B1568	PictureR.005B1568

_Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
00EBFEE0	5E	POP ESI	PictureR.005B1568
00EBFEE1	C9	LEAVE	
00EBFEE2	C3	RETN	
00EBFEE3	55	PUSH EBP	
00EBFEE4	8BEC	MOV EBP,ESP	
00EBFEE6	81EC 28010000	SUB ESP,128	
00EBFEEC	56	PUSH ESI	
00EBFEED	57	PUSH EDI	
00EBFEEF	BE F063EE00	MOV ESI,0EE63F0	ASCII "MainClass"
00EBFEF3	8B0D 08FFFFFF	LEA EDI,DWORD PTR SS:[EBP-128]	

_Again, Ctrl + F9, F8, Scroll down the signal of OEP:


```

00EDA2A0 50          PUSH EAX
00EDA2A1 A1 2800EF00 MOV EAX,DWORD PTR DS:[EF0028]
00EDA2A6 8B48 70     MOV ECX,DWORD PTR DS:[EAX+70]
00EDA2A9 3348 4C     XOR ECX,DWORD PTR DS:[EAX+4C]
00EDA2AC 3348 40     XOR ECX,DWORD PTR DS:[EAX+40]
00EDA2AF 2BF9       SUB EDI,ECX
00EDA2B1 FFD7       CALL EDI
00EDA2B3 8BD8       MOV EBX,EAX
00EDA2B5 5F         POP EDI
00EDA2B6 8BC3       MOV EAX,EBX
00EDA2B8 5E         POP ESI
00EDA2B9 5B         POP EBX
00EDA2BA C3         RETN
00EDA2BB 837C24 08 01 CMP DWORD PTR SS:[ESP+8],1

```

_Press Call EDI at F2, F9, F7: OEP J !

Address	Hex dump	Disassembly	Comment
004C80A2	6A 60	PUSH 60	
004C80AC	68 88A35300	PUSH PictureR.0053A388	
004C80B1	E8 CE230000	CALL PictureR.004CA484	
004C80B6	BF 94000000	MOV EDI,94	
004C80BB	8BC7	MOV EAX,EDI	
004C80BD	E8 7EDCFFFF	CALL PictureR.004C5D40	
004C80C2	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
004C80C5	8BF4	MOV ESI,ESP	
004C80C7	893E	MOV DWORD PTR DS:[ESI],EDI	
004C80C9	56	PUSH ESI	
004C80CA	FF15 2C270E01	CALL DWORD PTR DS:[10E272C]	kernel32.GetVersionExA
004C80D0	8B4E 10	MOV ECX,DWORD PTR DS:[ESI+10]	
004C80D3	890D 98895600	MOV DWORD PTR DS:[568998],ECX	
004C80D9	8B46 04	MOV EAX,DWORD PTR DS:[ESI+4]	
004C80DC	A3 A4895600	MOV DWORD PTR DS:[5689A4],EAX	
004C80E1	8B56 08	MOV EDX,DWORD PTR DS:[ESI+8]	
004C80E4	8915 A8895600	MOV DWORD PTR DS:[5689A8],EDX	

_This Is the signal of Import elimination. We must redirect it to a real cave memory! But the first step is to find the IAT. At the API GetVetsionA, Follow in dump> Memory Address, and we have the IAT:



_IAT Start:

Address	Value	Comment
010E24B4	695B6F5C	
010E24B8	57725873	
010E24BC	77637968	SHELL32.77637968
010E24C0	424A5A52	
010E24C4	524E6956	
010E24C8	6D4B7746	
010E24CC	54417170	
010E24D0	7D7D504F	
010E24D4	4F627355	

_IAT End:

Address	Value	Comment
010E2FE0	77C992B2	GDI32.ScaleWindowExtEx
010E2FE4	77E72F4B	kernel32.SetFileTime
010E2FE8	00EBADE9	
010E2FEC	00000000	
010E2FF0	01540002	
010E2FF4	000F1102	
010E2FF8	000B0188	
010E2FFC	000B0188	

_Total We have:

IAT Start: 010E24BC 77637968 SHELL32.77637968

IAT End: 010E2FE4 77E72F4B kernel32.SetFileTime

IAT Len: 010E2FE4-010E24BC = B28

OEP: C80AA

PID: 07F8

_Is Now, we must redirect the IAT to a Virtual Real IAT. ArmInline fire up and fill in:

PID: 0A50

Start Of Target Code: 401000: this is the address sections of text:

003F0000	00002000			Map	R	E	R	E
00400000	00001000	PictureR		Imag	R		RWE	
00401000	00101000	PictureR	.text	Imag	R		RWE	
00502000	0004F000	PictureR	.rdata	Imag	R		RWE	
00551000	00010000	PictureR	.data	Imag	R		RWE	

Length Of Target Code: 101000: Size of text sections:

(Slave) Process ID: 0x	A50
Start Of Target Code: 0x	401000
Length Of Target Code: 0x	101000

_In The table IAT elimination fill in:

Base Of Existing IAT: IAT Start: 010E24BC 77637968 SHELL32.77637968

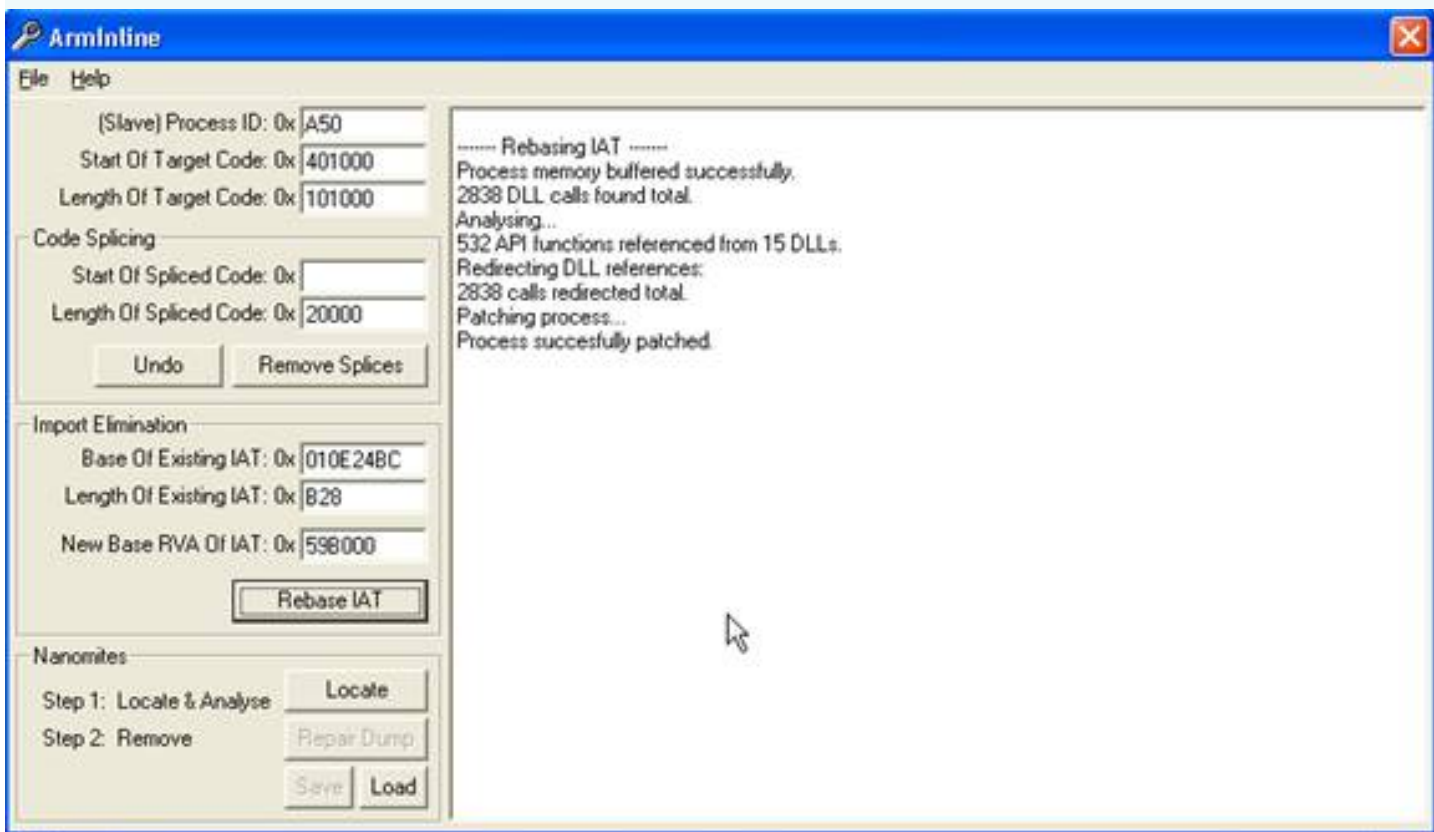
Length Of Existing IAT: IAT Len: 010E2FE4-010E24BC = B28

New Base RVA of IAT: We choose the section. ADATA to redirect the IAT: 59B000

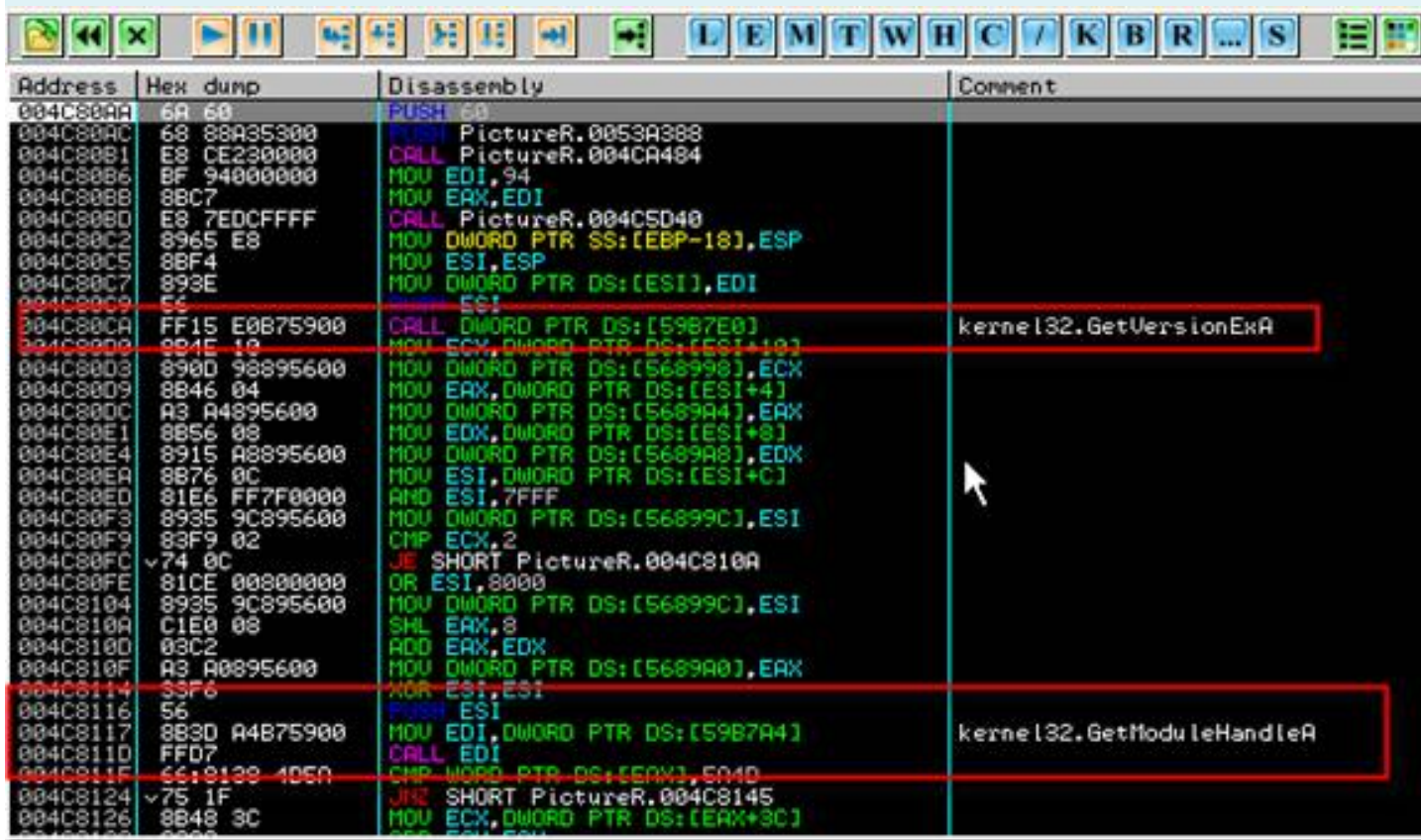
0056B000	00030000	PictureR	.text1	code
0059B000	00010000	PictureR	.adata	code
005AB000	00020000	PictureR	.data1	data, impo

Import Elimination	
Base Of Existing IAT: 0x	010E24BC
Length Of Existing IAT: 0x	B28
New Base RVA Of IAT: 0x	59B000
Rebase IAT	

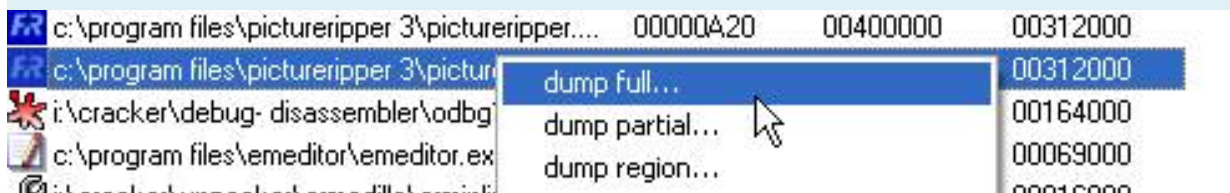
_Ok, Done, click Rebase IAT, and here we are:



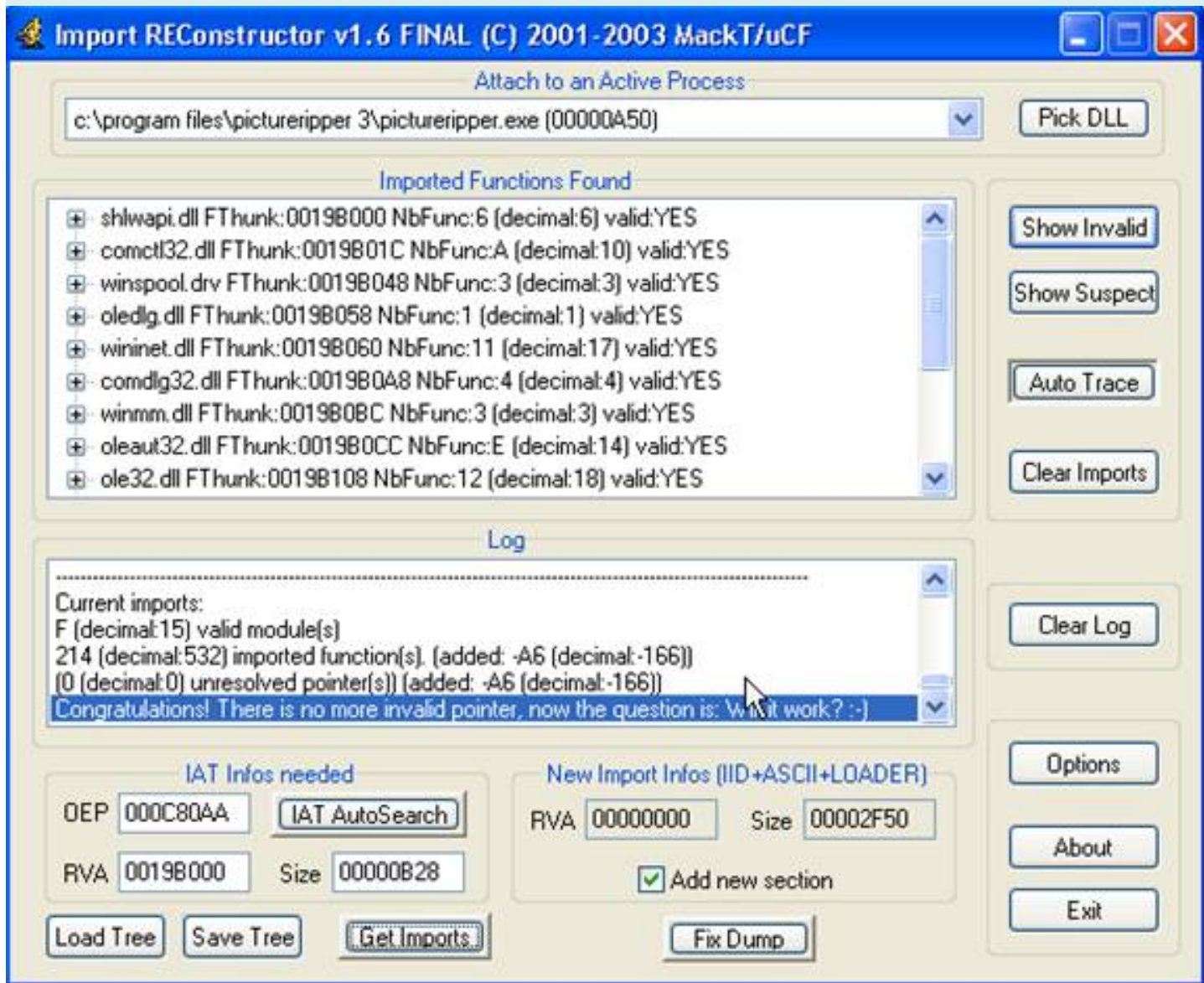
_Go Back to the CPU, and we see the IAT is successful redirect:



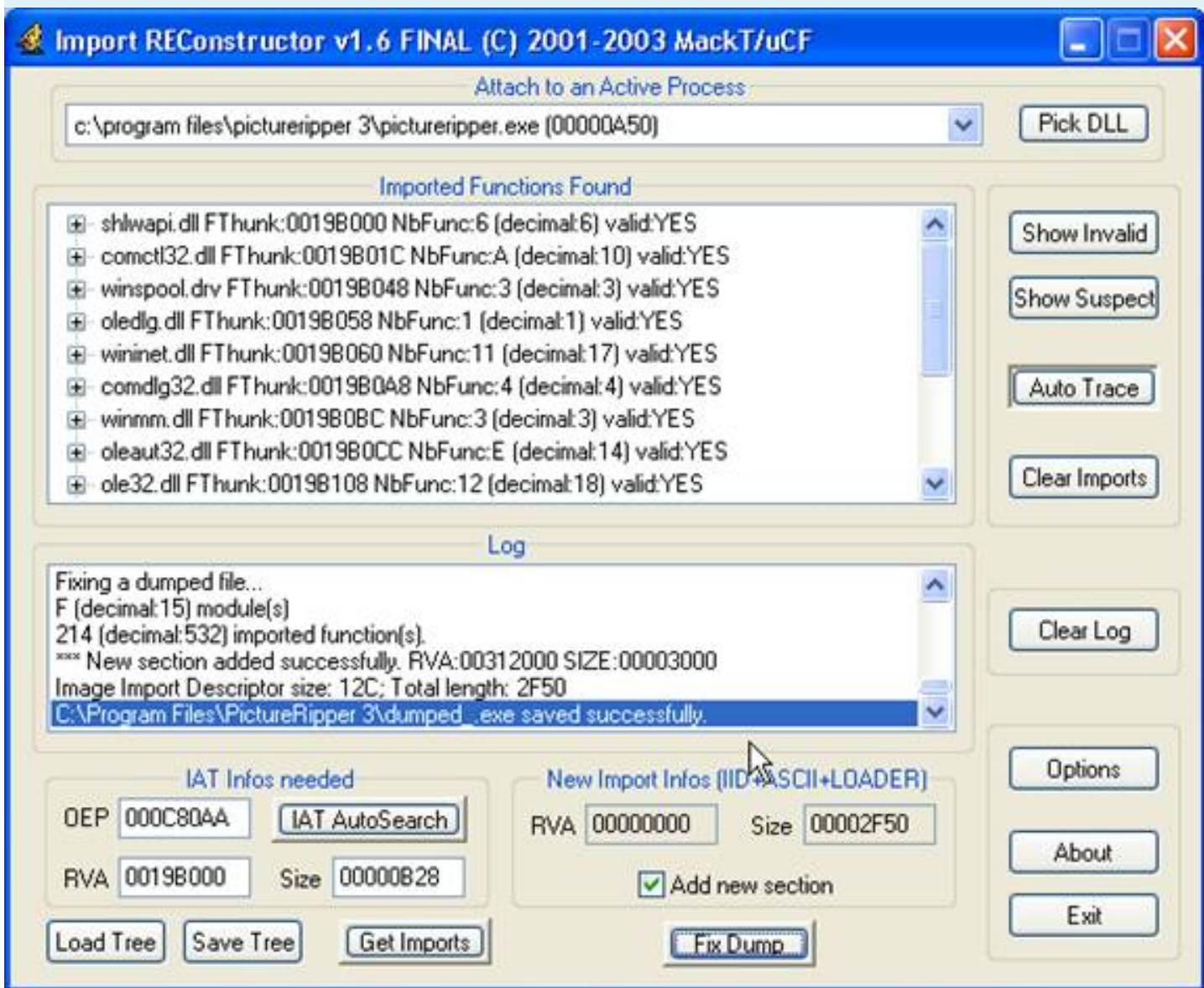
_Use LordPE and Full dump:



_Fire Up Imprec, fill OEP, IAT Auto Search, GetImport, Show Invalid, Thanks Cuts:

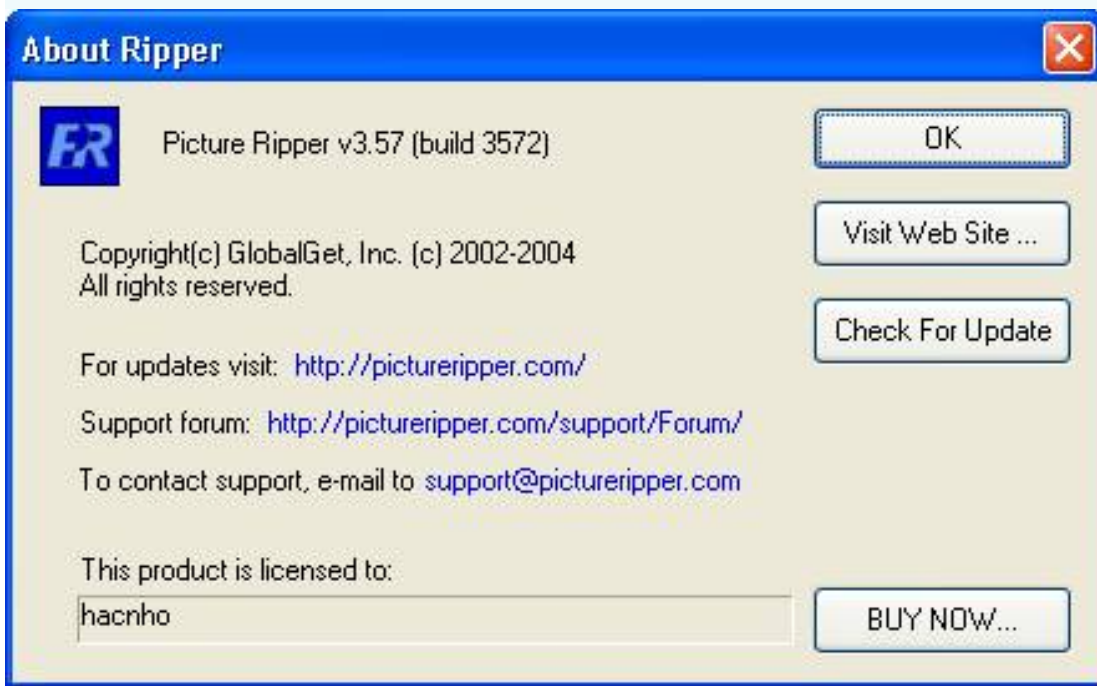


_Fix Dump:

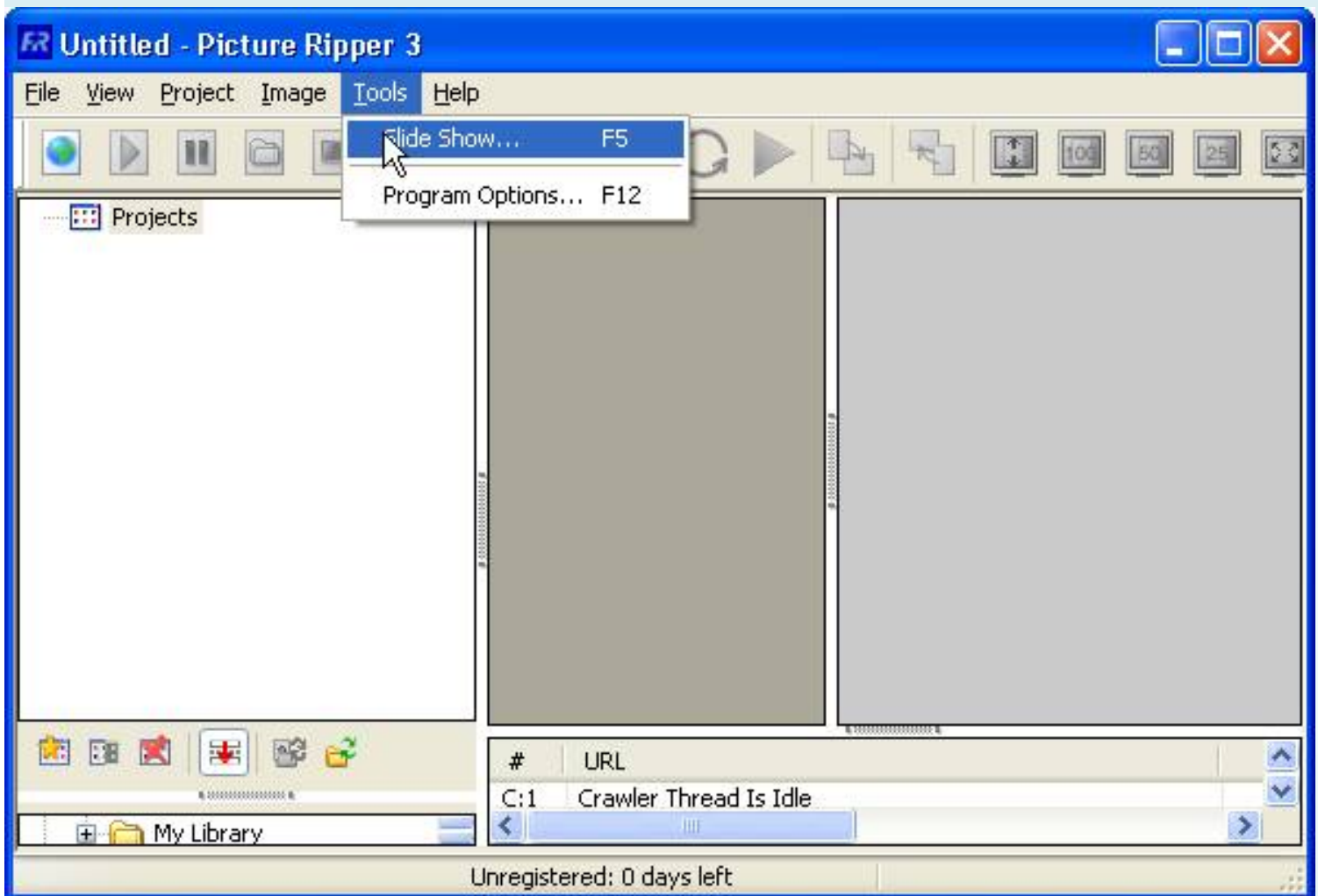


Run Dumped.exe:





_So When we try to open the Options menu:



_We Crash:

Free picture and movie downloader and viewer.

Free picture and movie downloader and viewer. has encountered a problem and needs to close. We are sorry for the inconvenience.



If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.

We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, [click here](#).

Debug

Send Error Report

Don't Send

_Ok, Le analyze this crash. Load N-REC to:

N-Rec 1.7 (c) 2003-2004 inferno / TSRh team

Source file: C:\Program Files\PictureRipper 3\PictureRipper.exe

Ini file: [not needed in this mode]

Open source file

Open ini file

Process

Cancel

☐ be verbose

Work mode

☐ recover by disassembly

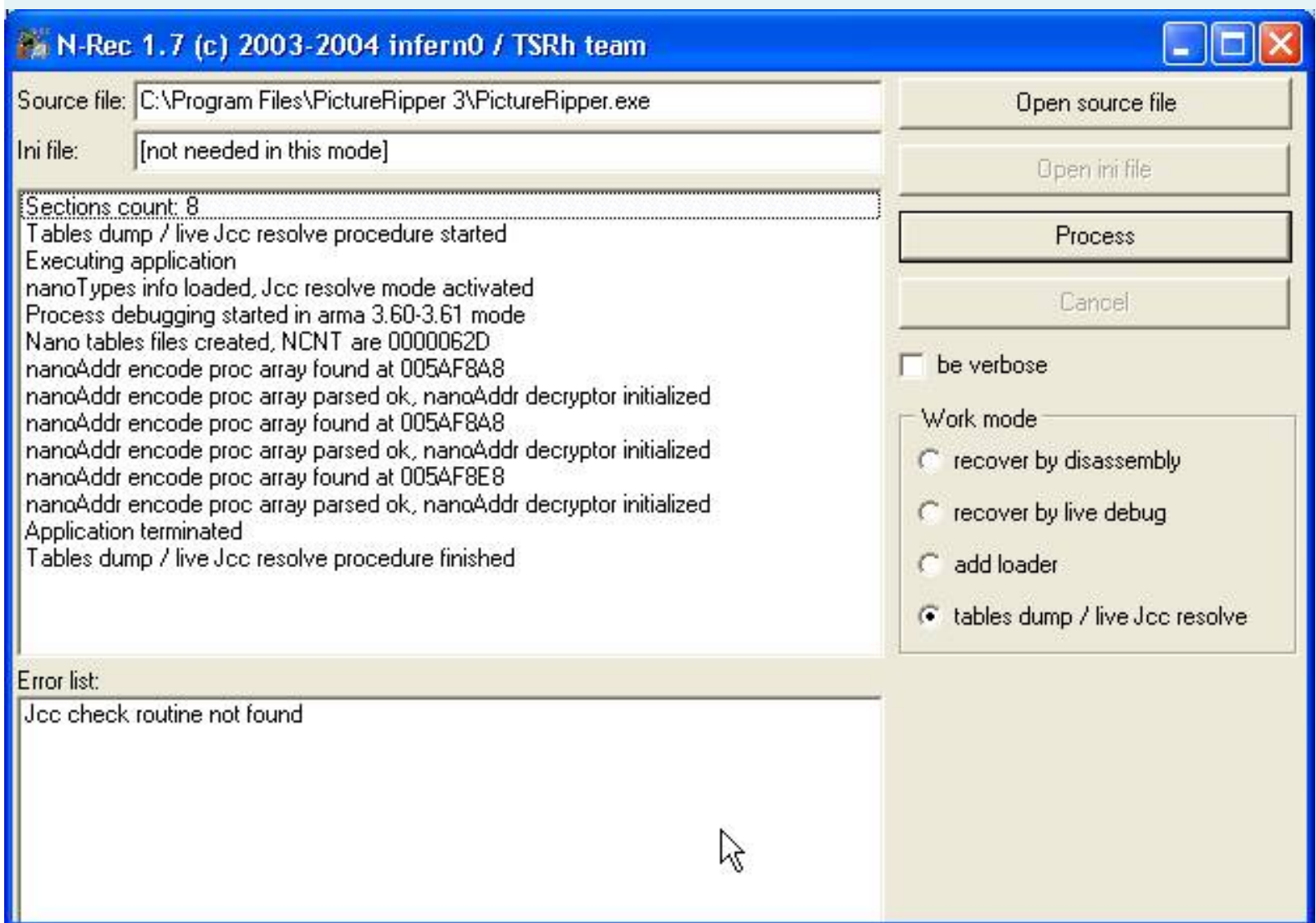
☐ recover by live debug

☐ add loader

☒ tables dump / live Jcc resolve

Error list:

_And We found the CC:



We Use ArmInline 0.71 to fix Nanomites, ok, to load dumped.exe OllyDBG:

hacnho - dumped_.exe - [CPU - main thread, module dumped_]

File View Debug Plugins Options Window Help

Paused

Address	Hex dump	Disassembly	Registers (FPU)
004C80AA	6A 60	push 60	EAX 00000000
004C80AC	68 88A35300	push dumped_.0053A388	ECX 0012FFB0
004C80B1	E8 CE230000	call dumped_.004CA484	EDX 7C90EB94 ntdll.K
004C80B6	BF 94000000	mov edi,94	EBX 7FFDE000
004C80BB	8BC7	mov eax,edi	ESP 0012FFC4
004C80BD	E8 7EDCFFFF	call dumped_.004C5D40	EBP 0012FFFO
004C80C2	8965 E8	mov dword ptr ss:[ebp-18],esp	ESI FFFFFFFF
004C80C5	8BF4	mov esi,esp	EDI 7C910738 ntdll.7
004C80C7	893E	mov dword ptr ds:[esi],edi	EIP 004C80AA dumped_
004C80C9	56	push esi	C 0 ES 0023 32bit 0
004C80CA	FF15 E4B75900	call dword ptr ds:[<kernel32.GetVersio	P 1 CS 001B 32bit 0
004C80D0	8B4E 10	mov ecx,dword ptr ds:[esi+10]	A 0 SS 0023 32bit 0
004C80D3	890D 98895600	mov dword ptr ds:[568998],ecx	Z 1 DS 0023 32bit 0
004C80D9	8B46 04	mov eax,dword ptr ds:[esi+4]	S 0 FS 003B 32bit 7
004C80DC	A3 A4895600	mov dword ptr ds:[5689A4],eax	T 0 GS 0000 NULL
004C80E1	8B56 08	mov edx,dword ptr ds:[esi+8]	D 0 LastErr ERROR_S
			EFL 00000246 (NO,NB,
			ST0 empty -UNORM D1D
			ST1 empty 0.0
			ST2 empty 0.0
			ST3 empty 0.0

Address	Hex dump	ASCII	Address	Value	Comment
005AB000	5F 3A C7 77 80 70 C7 77	_: 惹6p惹	0012FFC4	7C816D4F	RETURN to kernel32.7
005AB008	8E 4E C7 77 5C 85 C7 77	藏惹\吳w	0012FFC8	7C910738	ntdll.7C910738
005AB010	1A 5C C7 77 07 36 C7 77	□\惹□6惹	0012FFCC	FFFFFFFF	
005AB018	6E 3C C7 77 98 1B C7 77	n<惹?惹	0012FFD0	7FFDE000	
005AB020	E0 20 C7 77 6C 5B C7 77	?惹1[惹	0012FFD4	8054B038	
005AB028	00 00 00 00 9B A2 E7 77决壁	0012FFD8	0012FFC8	
005AB030	63 DE E6 77 86 AD E7 77	c咳w喘壁	0012FFDC	85665B40	

Command :

Program entry point

_View PID and text sections:

Select process to attach

Process	Name	Window	Path
00000330	svchost		C:\WINDOWS\system32\svchost.exe
00000358	svchost		C:\WINDOWS\System32\svchost.exe
0000038C	svchost		C:\WINDOWS\system32\svchost.exe
000003E0	svchost		C:\WINDOWS\system32\svchost.exe
00000444	spoolsv		C:\WINDOWS\system32\spoolsv.exe
0000048C	TurboLaunch	TurboLaunch	C:\Program Files\TurboLaunch\TurboLaun
00000540	Explorer	SysFader	C:\WINDOWS\Explorer.EXE
0000058C	UniKeyNT	UniKey 3.62	C:\Program Files\UniKey\UniKeyNT.exe
00000598	ctfmon	TF_FloatingLangBar_WndTitl	C:\WINDOWS\system32\ctfmon.exe
000005A4	sqlmangr	SQL Server Service Manager	C:\Program Files\Microsoft SQL Server\
000005F0	DkService		C:\Program Files\Executive Software\Di
00000630	MDM		C:\Program Files\Common Files\Microsof
000006AC	sqlservr		C:\PROGRAM1\MI6841\1\MSSQL\bin\sqlser
00000708	StarWind		C:\Program Files\Alcohol Soft\Alcohol
00000730	wdfmgr		C:\WINDOWS\system32\wdfmgr.exe
00000990	WINWORD	CiceroUIWndFrame	C:\Program Files\Microsoft Office\OFFI
000009F4	Snagit32	Snagit Capture Preview	C:\Program Files\TechSmith\Snagit 7\Sn
000009FC	TSCHelp	C:\Program Files\TechSmith	C:\Program Files\TechSmith\Snagit 7\TS
00000FF8	dumped_		C:\Program Files\PictureRipper 3\dump

Attach

Cancel

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00150000	00020000				Priv	RW	RW	
00250000	00006000				Priv	RW	RW	
00260000	00003000				Map	RW	RW	
00270000	00016000				Map	R	R	\Device\HarddiskVolume5\WIND
00290000	00030000				Map	R	R	\Device\HarddiskVolume5\WIND
002D0000	00041000				Map	R	R	\Device\HarddiskVolume5\WIND
00320000	00006000				Map	R	R	\Device\HarddiskVolume5\WIND
00330000	00041000				Map	R	R	
00380000	00001000				Priv	RWE	RWE	
00390000	00001000				Priv	RWE	RWE	
003A0000	00004000				Priv	RW	RW	
003B0000	00003000				Map	R	R	\Device\HarddiskVolume5\WIND
003C0000	00008000				Priv	RW	RW	
003D0000	00001000				Priv	RW	RW	
003E0000	00001000				Priv	RW	RW	
003F0000	00003000				Priv	RW	RW	
00400000	00001000	dumped_		PE header	Imag	R	RWE	
00401000	00101000	dumped_	.text		Imag	R	RWE	
00502000	0004F000	dumped_	.rdata		Imag	R	RWE	
00551000	0001A000	dumped_	.data		Imag	R	RWE	
0056B000	00030000	dumped_	.text1	code	Imag	R	RWE	
0059B000	00010000	dumped_	.adata	code	Imag	R	RWE	
005AB000	00020000	dumped_	.data1	data	Imag	R	RWE	
005CB000	000F0000	dumped_	.pdata		Imag	R	RWE	
006BB000	00057000	dumped_	.rsrc	resources	Imag	R	RWE	
00712000	00003000	dumped_	.mactk	imports	Imag	R	RWE	
00720000	00005000				Map	R E	R E	
007E0000	00002000				Map	R E	R E	
007F0000	00103000				Map	R	R	
00900000	000A4000				Map	R E	R E	
00C00000	00002000				Map	R	R	
00C10000	00002000				Map	R	R	
00C20000	00001000				Priv	RW	RW	
00CC0000	00002000				Map	R	R	
629C0000	00001000	LPK		PE header	Imag	R	RWE	
629C1000	00005000	LPK	.text	code, import	Imag	R	RWE	
629C6000	00001000	LPK	.data	data	Imag	R	RWE	
629C7000	00001000	LPK	.rsrc	resources	Imag	R	RWE	
629C8000	00001000	LPK	.reloc	relocations	Imag	R	RWE	
72000000	00001000	LPK		PE header	Imag	R	RWE	

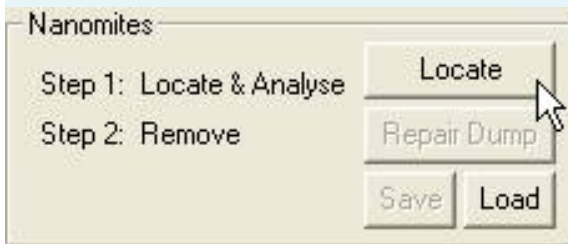
_Fill In ArmInline 0.71:

(Slave) Process ID: 0x FF8

Start Of Target Code: 0x 401000

Length Of Target Code: 0x 101000

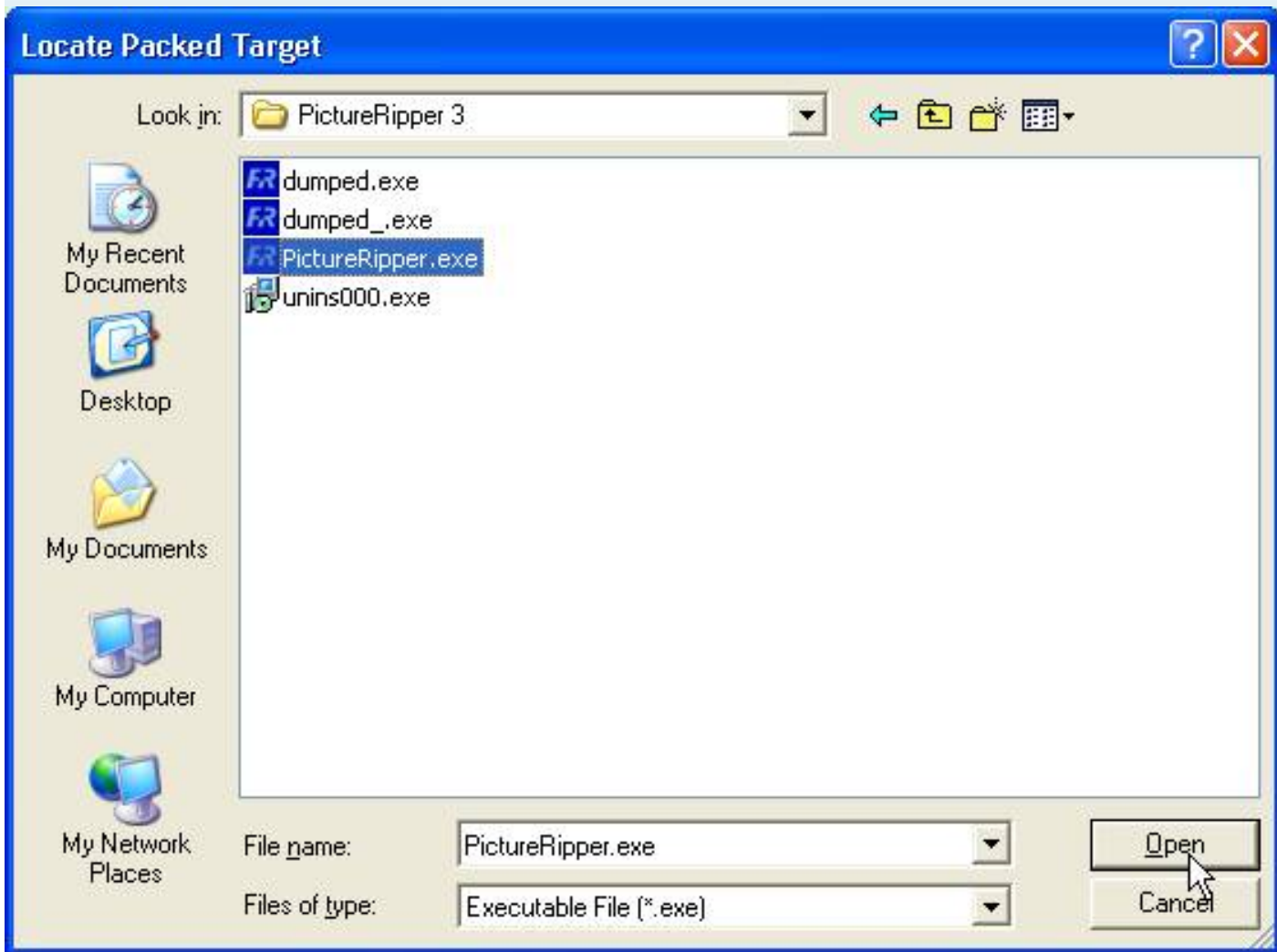
_Click Locate:



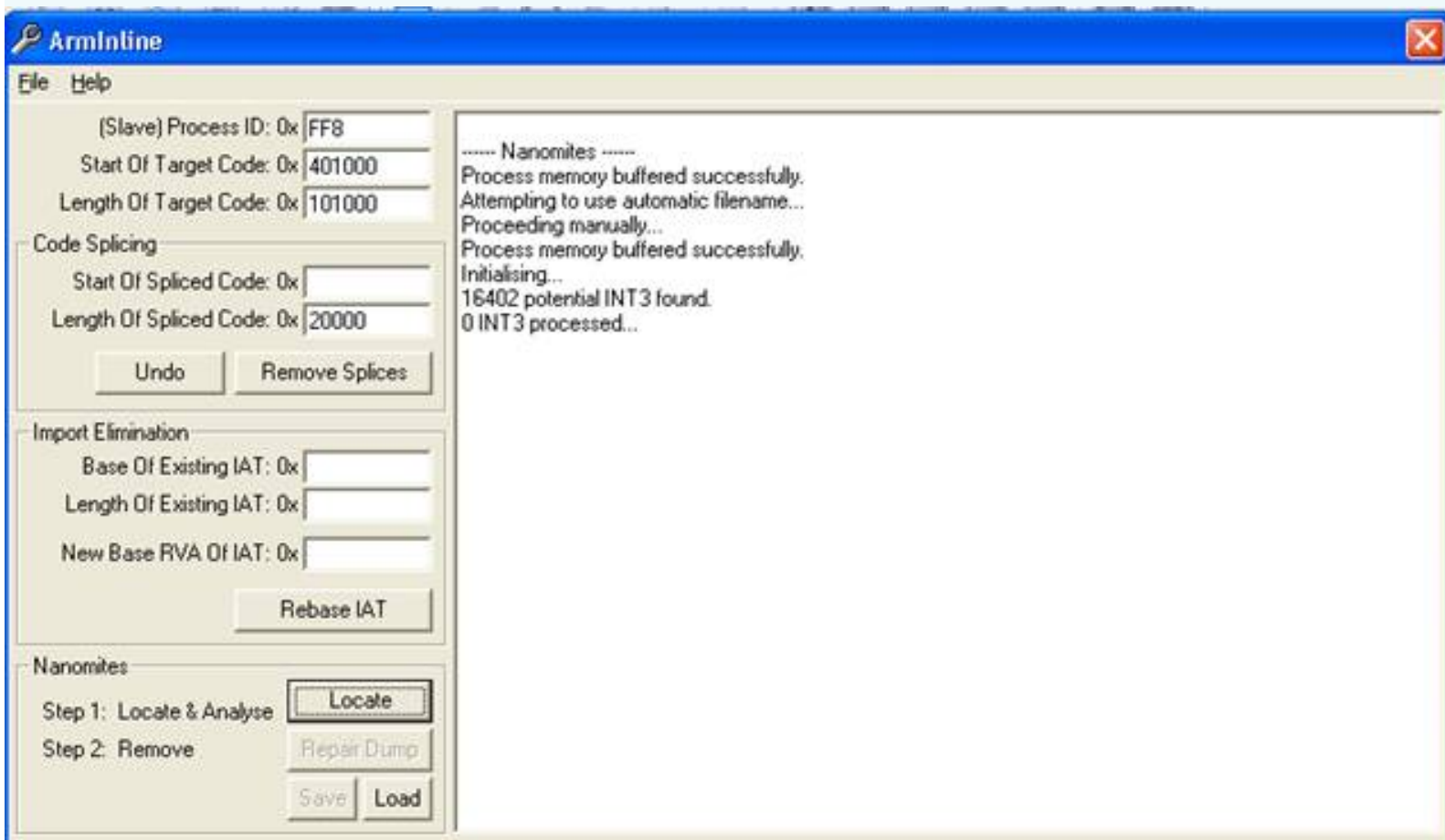
_Then Back to PictureRipper3:



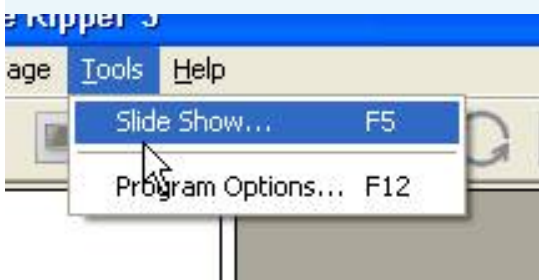
_Now In ArmInline: Load packet target:



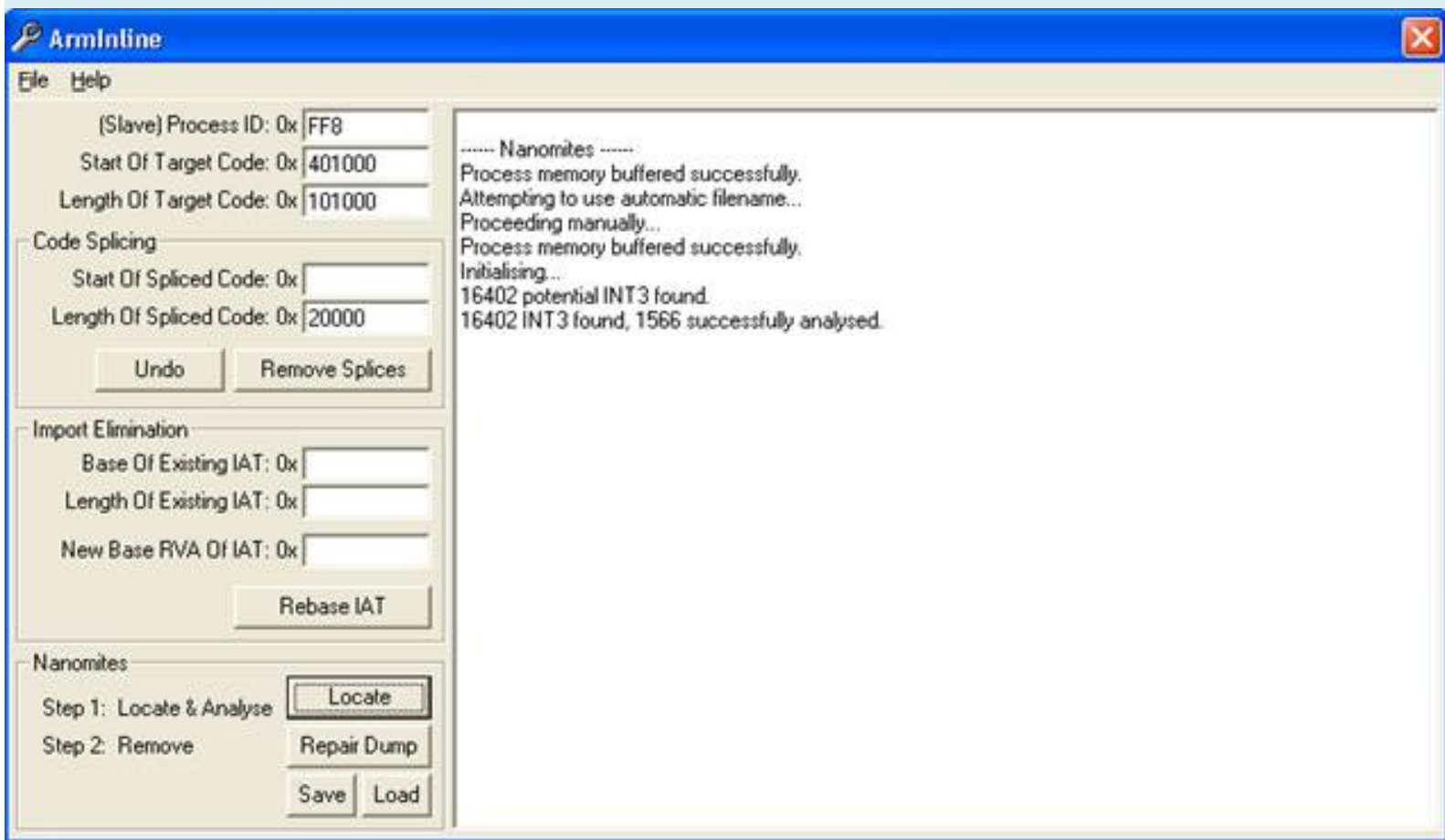
_We Choose the file PictureRipper.exe then:



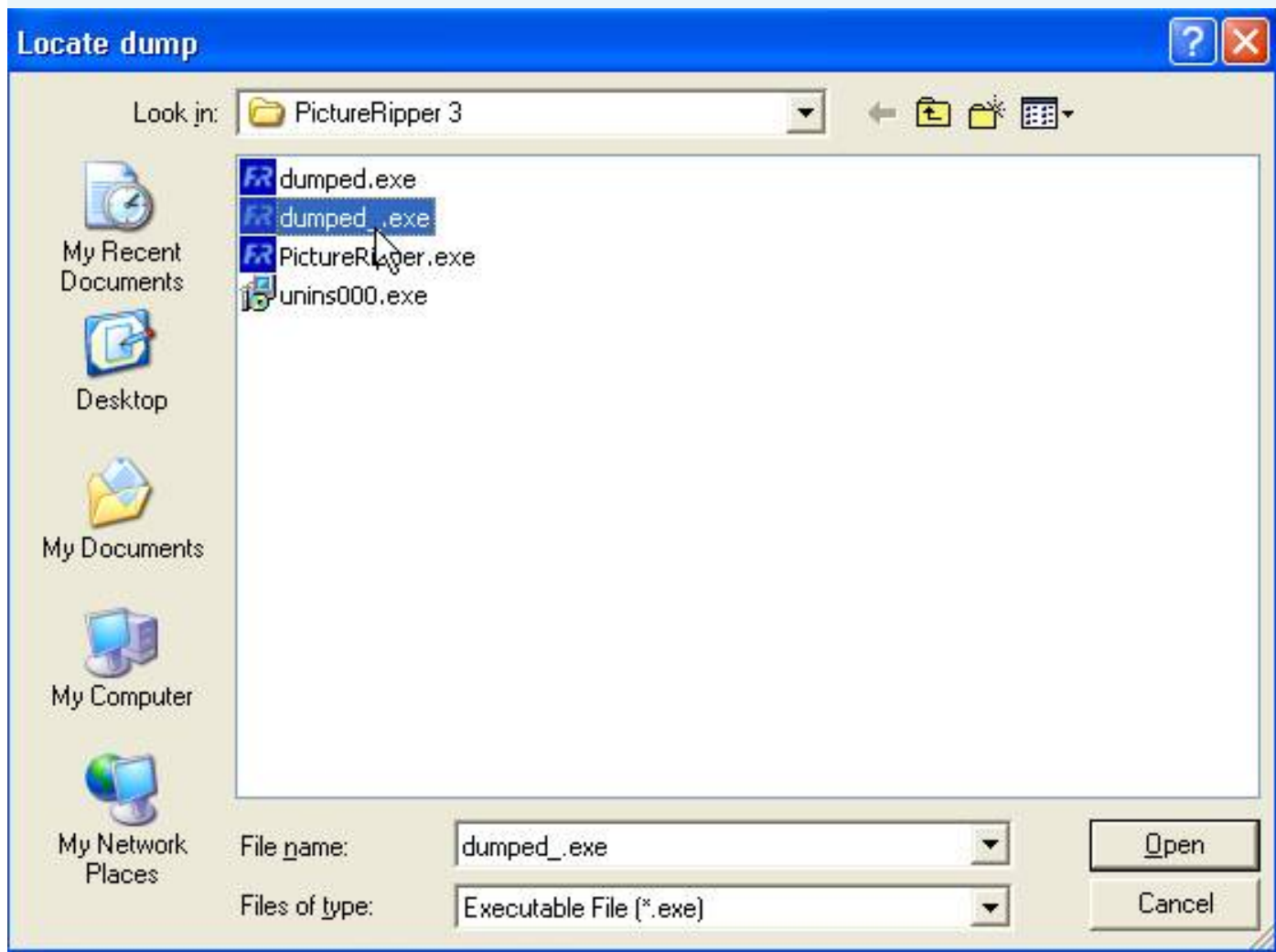
_Back To PictureRipper:



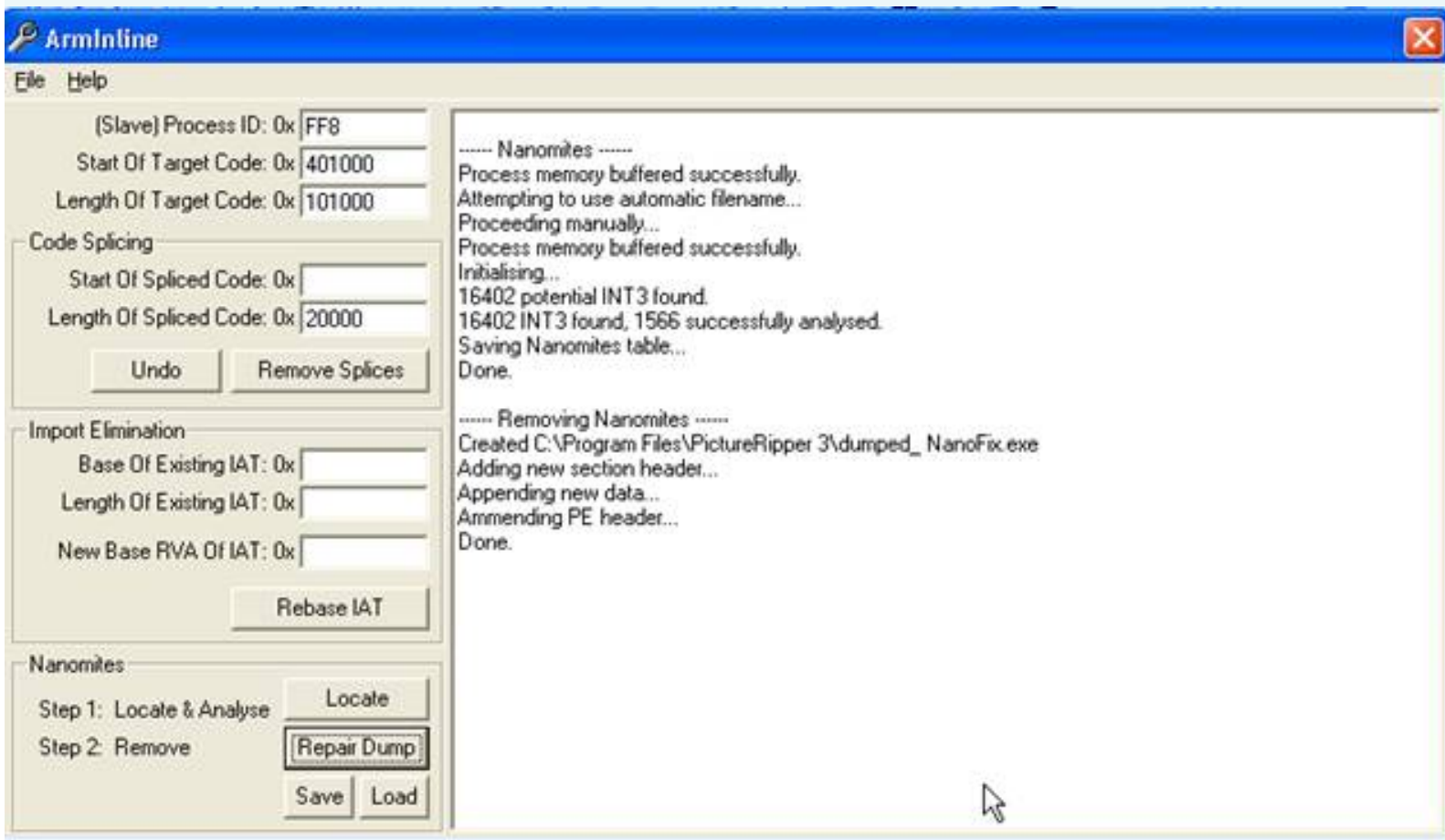
_Now:



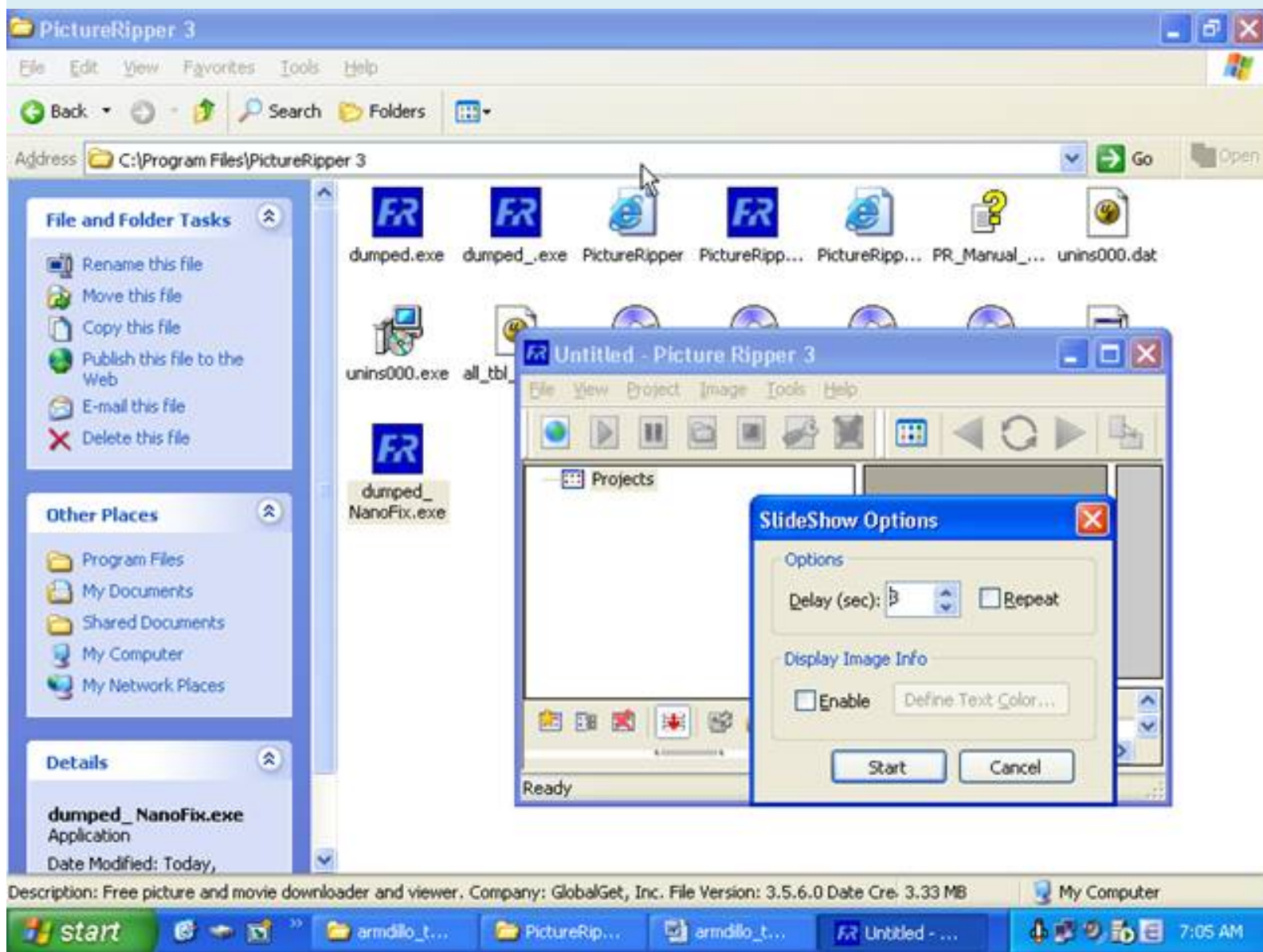
_Now You can save the NanoTable. Next. Click Repair dump:



_Ok:

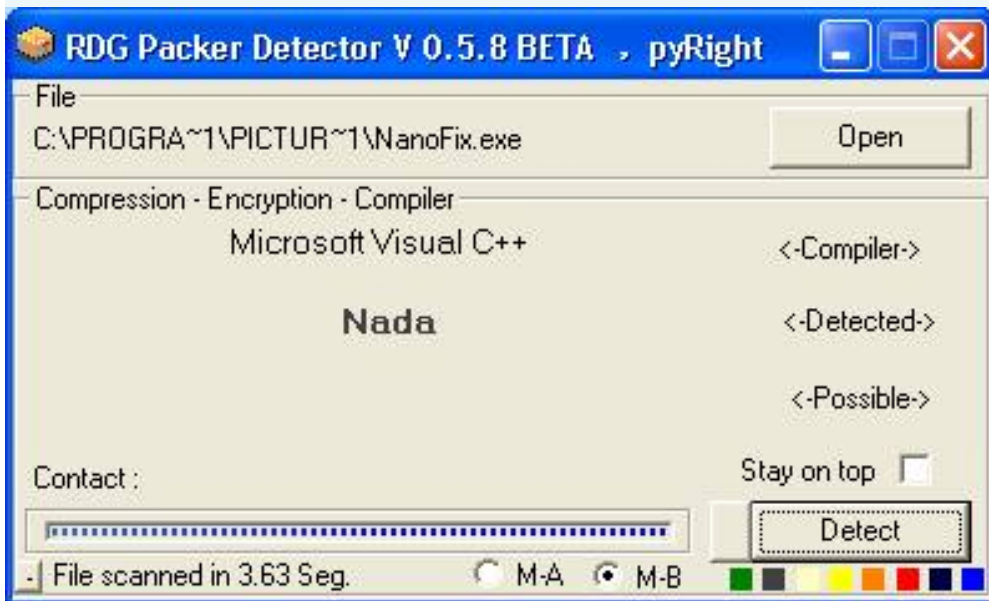


Run Dumped NanoFix.exe file, go to Options menu:



_Unpacked Successful!

Section Viewer					
Name	V. Offset	V. Size	R. Offset	R. Size	Flags
.data	00151000	00019268	00151000	00019268	C0000040
.text1	0016B000	00030000	0016B000	00030000	E0000020
.adata	0019B000	00010000	0019B000	00010000	E0000020
.data1	001AB000	00020000	001AB000	00020000	C0000040
.pdata	001CB000	000F0000	001CB000	000F0000	C0000040
.rsrc	002BB000	00057000	002BB000	00057000	40000040
.mact	00312000	00003000	00312000	00003000	E0000060
.nano	00315000	00041000	00315000	00040777	E00000E0



IV. Conclusion

_For More tuts, please visit <http://tinicat.de/hacnho> or <http://hacnho.exetools.com>

_Bye!

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtn, hosiminh, Nilrem, fly, MaDMAn_H3rCul3s, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light, iamidiot, WhyNotBar, trickyboy, Merc ... and you!

Special Thanx Cracks Latinos.

Thanx OllyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 25/10/2005)

Sa they are a nui

One lost a child

The miraculous Lu

Integration hunger Giang Ho

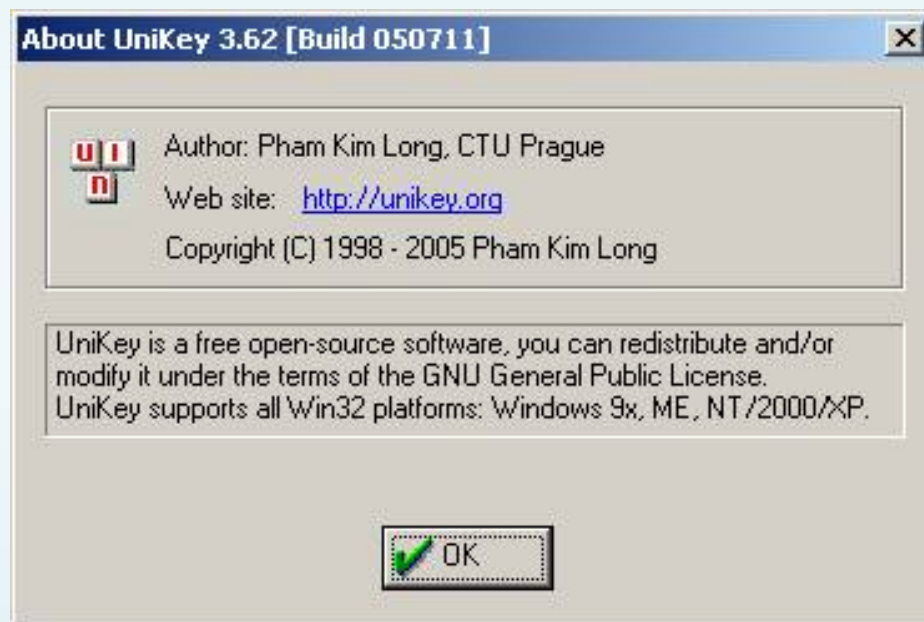
Armadillo collect sand-stone

Series # 9: Armadillo 4.xx-dll files

I. Intro

_Chao Her children, this is a tut tut to the end. In this tut I will discuss a form to protect the dll called na conduct a library of links. Some software does not protect dll protect in the main program. Of course dll can not be run direct to unpack the problem. We can not load up debugger modules that must be through the intermediary debug. Luckily, OllyDBG generated LoadDLL.exe role to a process mapped to the dll that we want to unpack! To illustrate this tut for my use UNIKEY 3.62 latest version not pack (the old versions are PeCompact pack with 1.84 or 2:00 and I have been clean meat J) ! Ok, we only started!





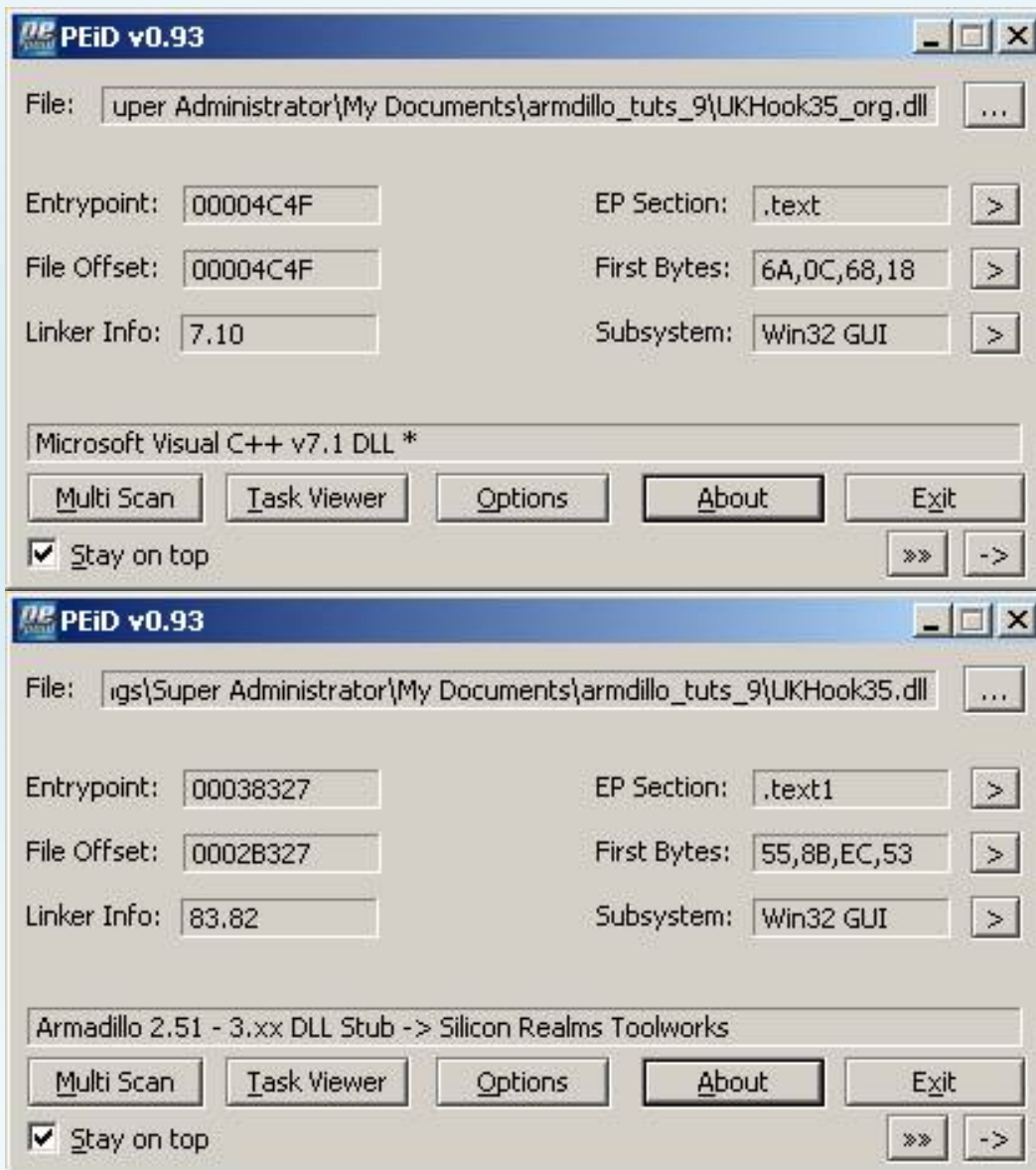
II. Tools:

1. OllyDBG - The best config debugger for ArmMUP by hacnho.
2. LordPE 1.4 Deluxe
3. Import REConstructor 1.6 Final

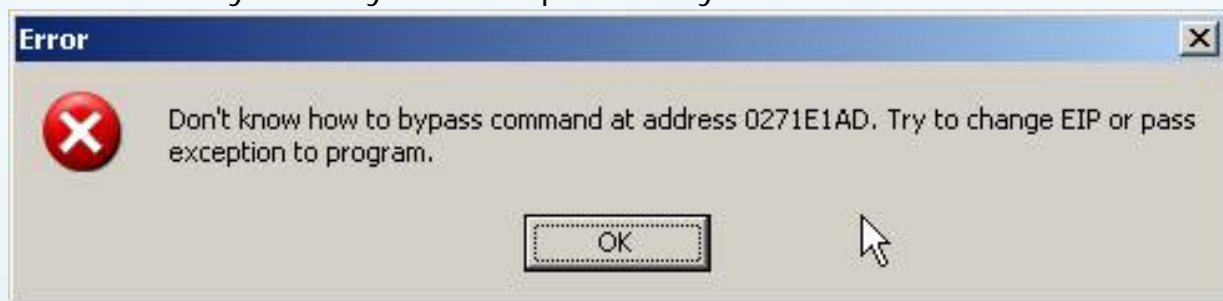
III. Unpacking

_Chuong The main Unikey will reference a dll is UKHook35.dll. I will use the Armadillo 4:20 to Build Public pack with options





_Sau File UKHook35.dll pack when I started the file UniKey.exe run, run good! Ok, you load the file onto Unikey.exe OllyDBG. A report of OllyDBG:



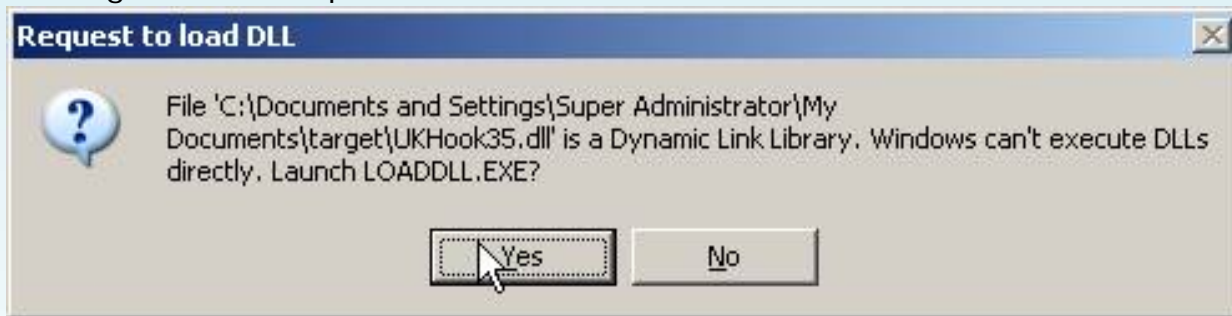
_Nhan Ok, and Shift + F9 me here:

Address	Hex dump	Disassembly	Comment
004108F0	5 6A 60	PUSH 60	
004108F2	68 F8CF4100	PUSH UniKey.0041CFF8	
004108F7	E8 10340000	CALL UniKey.00413D0C	
004108FC	BF 94000000	MOV EDI,94	
00410901	8BC7	MOV EAX,EI	
00410903	E8 C84E0000	CALL Unikey.004157D0	
00410908	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0041090B	8BF4	MOV ESI,ESP	
0041090D	893E	MOV DWORD PTR DS:[ESI],EDI	
0041090F	56	PUSH ESI	
00410910	FF15 6CB04100	CALL DWORD PTR DS:[<&KERNEL32.GetVersion	pVersionInformation
00410916	8B4E 10	MOV ECX,DWORD PTR DS:[ESI+10]	GetVersionExA

_Nhan Alt + E:

Base	Size	Entry	Name	File version	Path
00400000	00000000	004100F0	UnlKey		C:\Documents and Settings\Super Administrator\My Documents\target\UnlKey.exe
00400000	010A1000	0040B164	SHELL32	6.00.2600.1106	C:\WINDOWS\system32\SHELL32.dll
10000000	00000000	10000000	UKHook35		C:\Documents and Settings\Super Administrator\My Documents\target\UKHook35.dll
629C0000	00000000	629C0C32	LFX	5.1.2600.0 (xpc	C:\WINDOWS\System32\LFX.DLL
664F0000	00000000	664F17C3	inetnib1	5.1.2600.0 (xpc	C:\WINDOWS\System32\inetnib1.dll
70A70000	00000000	70A70386	SHLWAPI	6.00.2600.1106	C:\WINDOWS\System32\SHLWAPI.dll
71950000	000E4000	7195E008	comctl32	5.0 (xpapi.0208	C:\WINDOWS\WinSxS\x-ww6_Microsoft.Windows.Common-Controls_6595b64144ccf1df_5.0.1
71A00000	00000000	71A01226	WS2HELP	5.1.2600.0 (xpc	C:\WINDOWS\System32\WS2HELP.dll

_Ok, Close the window OllyDBG, you load the file directly onto UKHook35.dll Olly, a message will show up:



Yes you will _Nhan here:

Address	Hex dump	Disassembly	Comment
10038327	55	PUSH EBP	
10038328	8BEC	MOV EBP,ESP	
1003832A	53	PUSH EBX	
1003832B	8B5D 08	MOV EBX,DWORD PTR SS:[EBP+8]	
1003832E	56	PUSH ESI	
1003832F	8B75 0C	MOV ESI,DWORD PTR SS:[EBP+C]	
10038332	57	PUSH EDI	
10038333	8B7D 10	MOV EDI,DWORD PTR SS:[EBP+10]	

_Nhan Alt + F1 to enter HE GetModuleHandleA, Shift + F9:

Address	Value	Comment
0006ED30	7712B124	CALL to GetModuleHandleA from O
0006ED34	771A22E4	pModule = "KERNEL32.DLL"
0006ED38	7712AD56	RETURN to OLEAUT32.7712AD56 from
0006ED3C	771A2080	OLEAUT32.771A2080
0006ED40	000003E8	
0006ED44	7712B0CA	RETURN to OLEAUT32.7712B0CA from
0006ED48	00000001	
0006ED4C	00000001	
0006ED50	77120080	OLEAUT32.77120080
0006ED54	00000000	

_Shift + F9 times 2:

Address	Value	Comment
0006ED2C	7712B124	CALL to GetModuleHandleA from O
0006ED30	771A22E4	pModule = "KERNEL32.DLL"
0006ED34	7712ADAC	RETURN to OLEAUT32.7712ADAC from
0006ED38	771A2064	OLEAUT32.771A2064
0006ED3C	000003E8	
0006ED40	7712B0D0	RETURN to OLEAUT32.7712B0D0 from
0006ED44	00000001	
0006ED48	00000001	
0006ED4C	00000001	
0006ED50	77120080	OLEAUT32.77120080

_Shift + F9 times 3:

Address	Value	Comment
0006ED2C	7712B124	CALL to GetModuleHandleA from O
0006ED30	771A22E4	pModule = "KERNEL32.DLL"
0006ED34	7712ADAC	RETURN to OLEAUT32.7712ADAC from
0006ED38	771A2064	OLEAUT32.771A2064
0006ED3C	000003E8	
0006ED40	7712B0D0	RETURN to OLEAUT32.7712B0D0 from
0006ED44	00000001	
0006ED48	00000001	
0006ED4C	00000001	
0006ED50	77120080	OLEAUT32.77120080

_Lan 4:

Address	Value	Comment
000686F0	02743748	CALL to GetModuleHandleA from b
000686F4	02757474	pModule = "kernel32.dll"
000686F8	02758738	ASCII "VirtualFree"
000686FC	00000001	
00068700	00396008	
00068704	00000000	
00068708	00000000	
0006870C	00000000	
00068710	00000000	
00068714	00000000	

_Lan 5:

Address	Value	Comment
00068460	0272ACC1	CALL to GetModuleHandleA from 0
00068464	000685A4	pModule = "kernel32.dll"
00068468	00000000	
0006846C	EB4C0000	
00068470	5FDA0006	
00068474	0275697C	
00068478	00000000	
0006847C	80000000	
00068480	00003FFF	
00068484	00000000	

_Nhin Up CPU:

Address	Hex dump	Disassembly	Comment
77E7AD86	837C24 04 00	CMP DWORD PTR SS:[ESP+4],0	
77E7AD8B	0F84 37010000	JE kernel32.77E7AEC8	
77E7AD91	FF7424 04	PUSH DWORD PTR SS:[ESP+4]	
77E7AD95	E8 F8050000	CALL kernel32.77E7B392	
77E7AD9A	85C0	TEST EAX,EAX	
77E7AD9C	74 08	JE SHORT kernel32.77E7ADA6	
77E7AD9E	FF70 04	PUSH DWORD PTR DS:[EAX+4]	
77E7ADA1	E8 27060000	CALL kernel32.GetModuleHandleW	
77E7ADA6	C2 0400	RETN 4	
77E7ADA9	55	PUSH EBP	
77E7ADAA	8BEC	MOV EBP,ESP	
77E7ADAC	83EC 10	SUB ESP,10	
77E7ADAF	57	PUSH EDI	

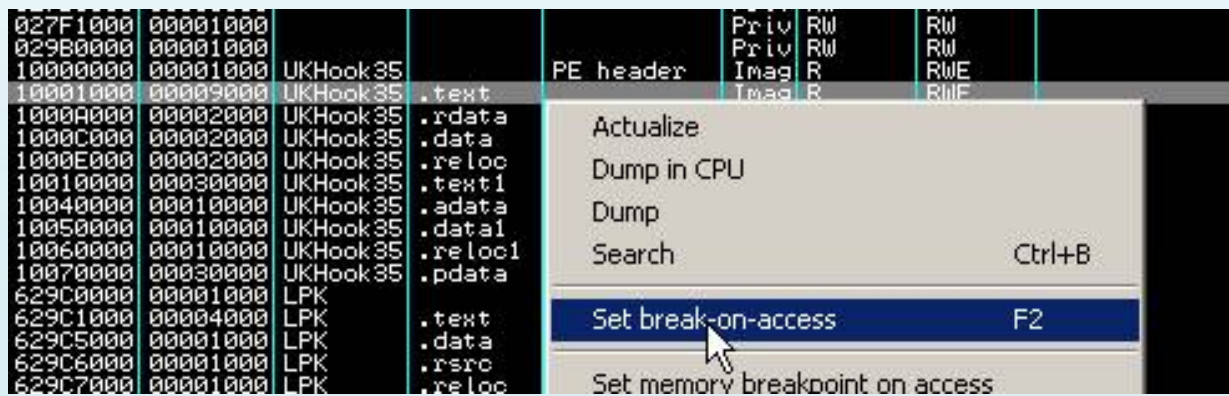
_Bam F8 RETN trace down through 4, you will here:

Address	Hex dump	Disassembly	Comment
0272ACC1	8B00 E4C97502	MOV ECX,DWORD PTR DS:[275C9E4]	
0272ACC7	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
0272ACCA	A1 E4C97502	MOV EAX,DWORD PTR DS:[275C9E4]	
0272ACCF	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
0272ACD2	75 16	JNZ SHORT 0272ACEA	
0272ACD4	8D85 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
0272ACDA	50	PUSH EAX	
0272ACDB	FF15 E0207502	CALL DWORD PTR DS:[27520E0]	kernel32.LoadLibraryA
0272ACE1	8B00 E4C97502	MOV ECX,DWORD PTR DS:[275C9E4]	
0272ACE7	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
0272ACEA	A1 E4C97502	MOV EAX,DWORD PTR DS:[275C9E4]	
0272ACEF	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
0272ACF2	0F84 32010000	JE 0272AE2A	
0272ACF8	33C9	XOR ECX,ECX	
0272ACFA	8B07	MOV EAX,DWORD PTR DS:[EDI]	
0272ACFC	3918	CMP DWORD PTR DS:[EAX],EBX	
0272ACFE	74 06	JE SHORT 0272AD06	
0272AD00	41	INC ECX	
0272AD01	83C0 0C	ADD EAX,0C	

_Patch Magic jump into EB:

Address	Hex dump	Disassembly	Comment
0272ACC1	8B00 E4C97502	MOV ECX,DWORD PTR DS:[275C9E4]	
0272ACC7	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
0272ACCA	A1 E4C97502	MOV EAX,DWORD PTR DS:[275C9E4]	
0272ACCF	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
0272ACD2	75 16	JNZ SHORT 0272ACEA	
0272ACD4	8D85 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
0272ACDA	50	PUSH EAX	
0272ACDB	FF15 E0207502	CALL DWORD PTR DS:[27520E0]	kernel32.LoadLibraryA
0272ACE1	8B00 E4C97502	MOV ECX,DWORD PTR DS:[275C9E4]	
0272ACE7	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
0272ACEA	A1 E4C97502	MOV EAX,DWORD PTR DS:[275C9E4]	
0272ACEF	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
0272ACF2	E9 33010000	JMP 0272AE2A	
0272ACF7	90	NOP	
0272ACF8	33C9	XOR ECX,ECX	
0272ACFA	8B07	MOV EAX,DWORD PTR DS:[EDI]	
0272ACFC	3918	CMP DWORD PTR DS:[EAX],EBX	
0272ACFE	74 06	JE SHORT 0272AD06	

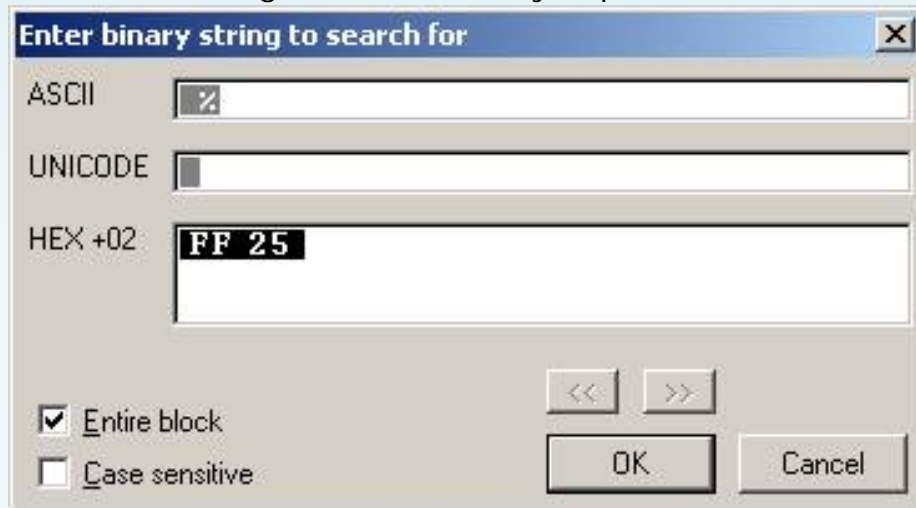
_HD GetModuleHandleA, Alt + M:



_Shift + F9: OEP:

Address	Hex dump	Disassembly	Comment
10004C4F	6A 0C	PUSH 0C	
10004C51	68 18A20010	PUSH UKHook35.1000A218	
10004C56	E8 61160000	CALL UKHook35.100062BC	
10004C58	33C0	XOR EAX,EAX	
10004C5D	40	INC EAX	
10004C5E	8945 E4	MOV DWORD PTR SS:[EBP-1C],EAX	
10004C61	8B75 0C	MOV ESI,DWORD PTR SS:[EBP+C]	
10004C64	33FF	XOR EDI,EDI	
10004C66	3BF7	CMP ESI,EDI	
10004C68	75 0C	JNZ SHORT UKHook35.10004C76	
10004C6A	393D D4D40010	CMP DWORD PTR DS:[1000D4D4],EDI	
10004C70	0F84 B3000000	JE UKHook35.10004D29	

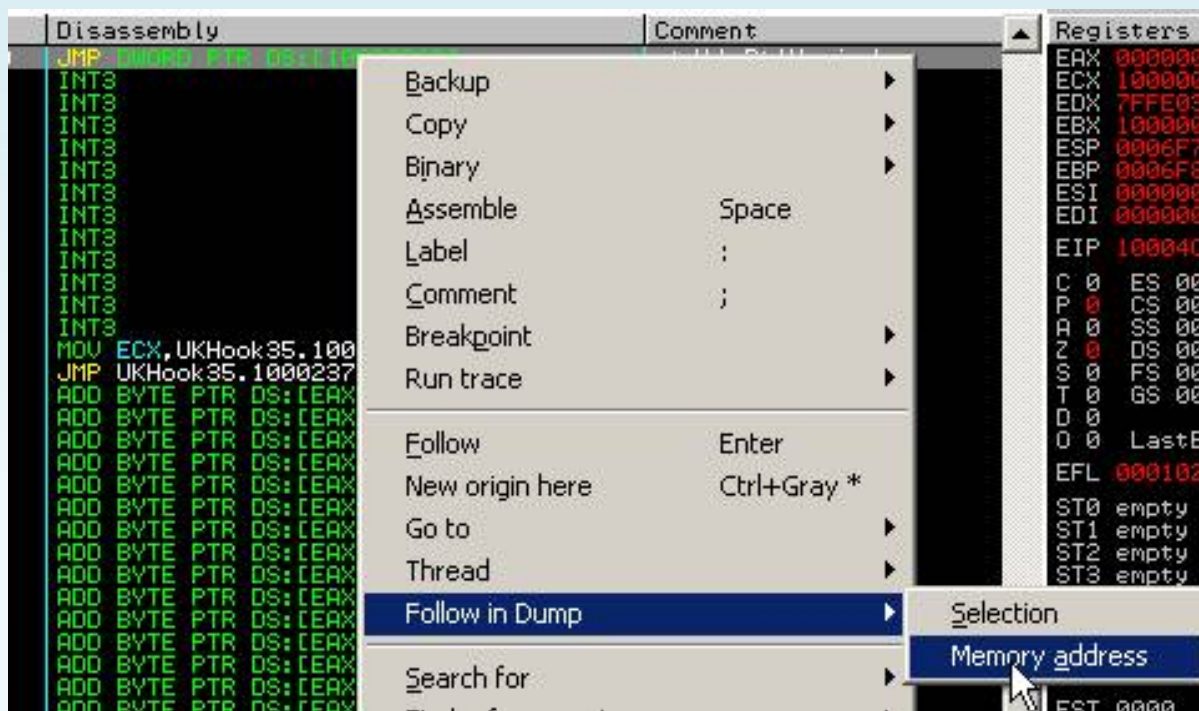
_Ta Start looking for IAT, in OEP you press Ctrl + B:



_Ban To here:

Address	Hex dump	Disassembly	Comment
100094DE	FF25 68A00010	JMP DWORD PTR DS:[1000A068]	ntdll.RtlUnwind
100094E4	CC	INT3	
100094E5	CC	INT3	
100094E6	CC	INT3	
100094E7	CC	INT3	
100094E8	CC	INT3	
100094E9	CC	INT3	
100094EA	CC	INT3	
100094EB	CC	INT3	
100094EC	CC	INT3	
100094ED	CC	INT3	
100094EE	CC	INT3	
100094EF	CC	INT3	
100094F0	B9 78CA0010	MOV ECX,UKHook35.1000CA78	
100094F5	E9 768EFFFF	JMP UKHook35.10002370	
100094FA	0000	ADD BYTE PTR DS:[EAX],AL	
100094FC	0000	ADD BYTE PTR DS:[EAX],AL	
100094FE	0000	ADD BYTE PTR DS:[EAX],AL	

_Right Click:



_Ban To here:

Address	Value	Comment
1000A068	77F60C44	ntdll.RtlUnwind
1000A06C	77E77362	kernel32.GetLocaleInfoA
1000A070	77F755DE	ntdll.RtlEnterCriticalSection
1000A074	77F75690	ntdll.RtlLeaveCriticalSection
1000A078	77E7E35E	kernel32.GetCPInfo
1000A07C	77E7A7DF	kernel32.GetCurrentThreadId
1000A080	77E7E358	kernel32.GetCommandLineA
1000A084	77E7D34A	kernel32.TlsAlloc
1000A088	77F5150C	ntdll.RtlSetLastWin32Error
1000A08C	77E7388C	kernel32.TlsFree

_Cuon Up to make IAT start:

Address	Value	Comment
1000A000	77E74E0A	kernel32.lstrcpyA
1000A004	77E7D0AF	kernel32.GetVersionExA
1000A008	77E7A6F0	kernel32.CloseHandle
1000A00C	77E768FA	kernel32.UnmapViewOfFile
1000A010	77E761AA	kernel32.MapViewOfFile
1000A014	77F51502	ntdll.RtlGetLastWin32Error
1000A018	77E7A543	kernel32.CreateFileMappingA
1000A01C	77E7A949	kernel32.WideCharToMultiByte
1000A020	77E7391C	kernel32.GlobalUnlock
1000A024	77E72024	kernel32.GlobalLock

_Cuon Down search IAT End:

Address	Value	Comment
1000B04C	65656220	
1000B050	6564206E	
1000B054	74636574	
1000B058	77206465	OLE32.77206465
1000B05C	68636968	
1000B060	73616820	
1000B064	726F6320	
1000B068	74707572	
1000B06C	74206465	
1000B070	70206568	

_Tom The information we have:

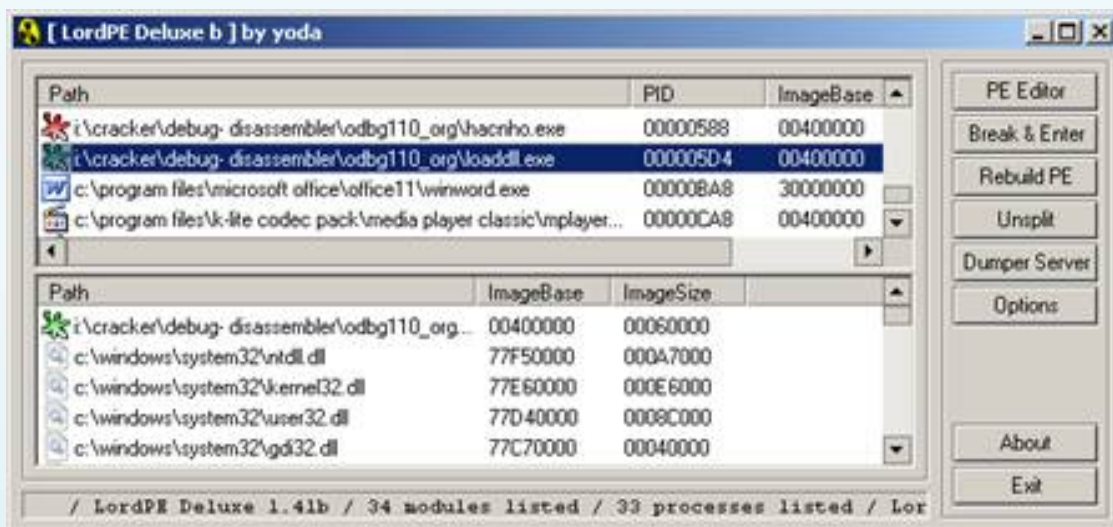
OEP: 4C4F:

IAT Start: 1000A000 77E74E0A kernel32.lstrcpyA

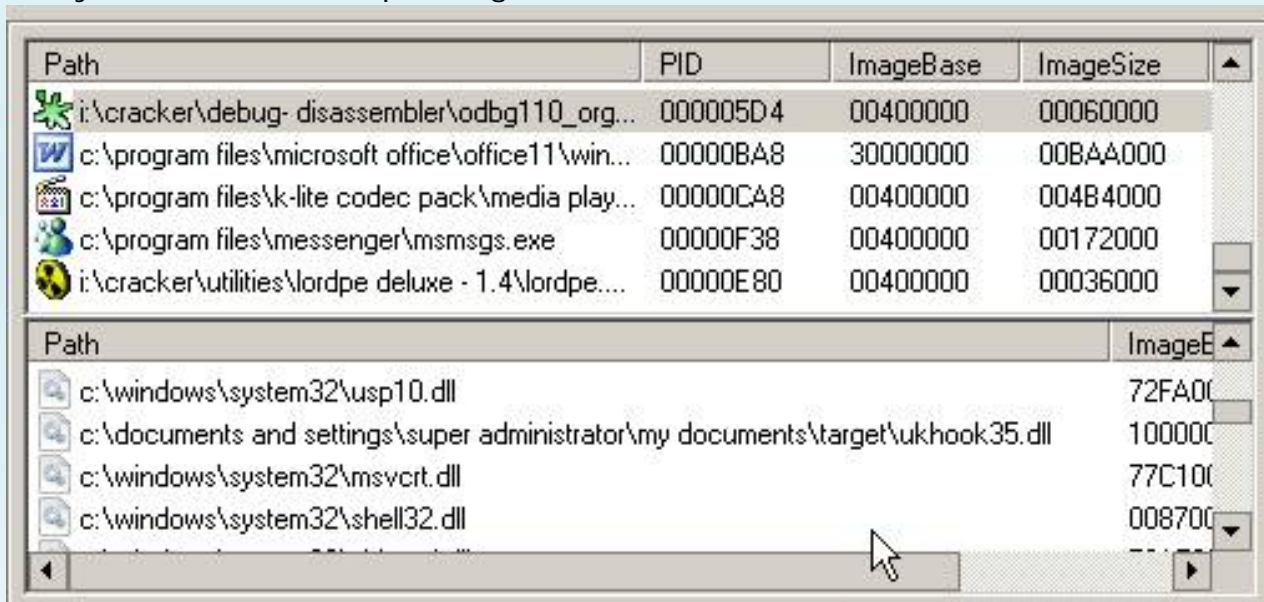
IAT End: 1000B058 77206465 OLE32.77206465

IAT Len: 1058

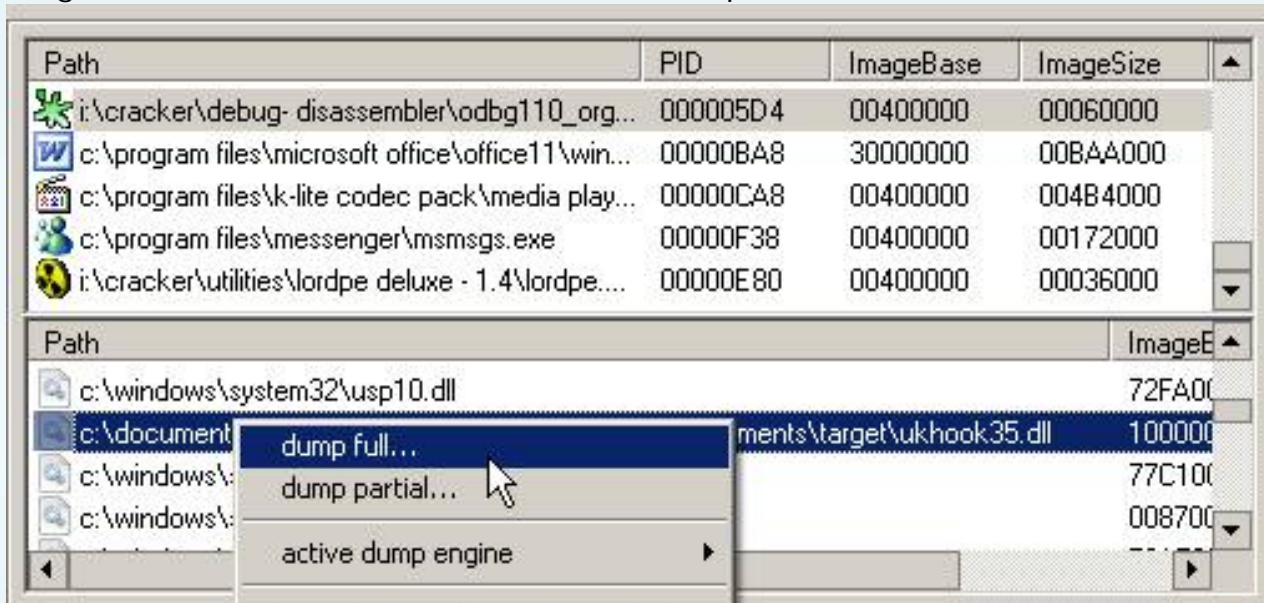
_Bay Hours Alt + C back position OEP, you open LordPE:



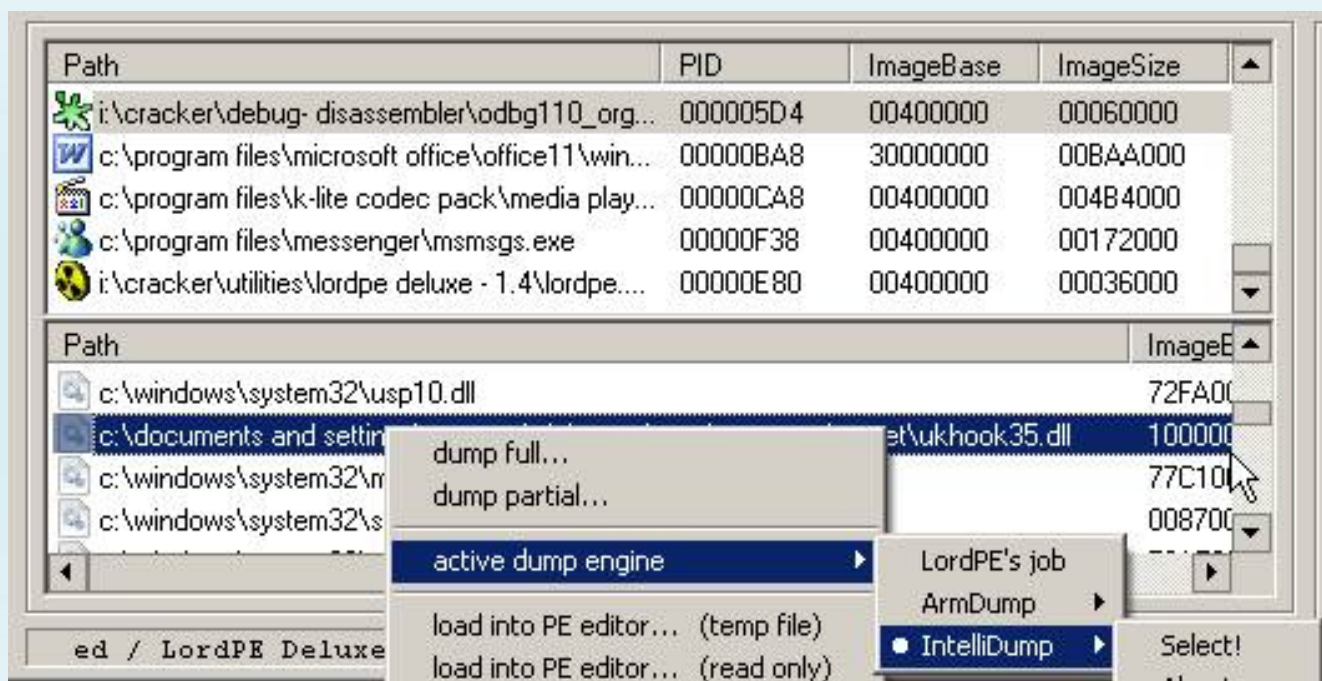
_Di Course, we can not see the process dll list. Do we use LoadDLL made reference to the process UKHook35.dll. You click on load.dll.exe process, look in the module's window to see many modules that are pointing to load.dll.exe, scroll down to see UKHook.dll modules:



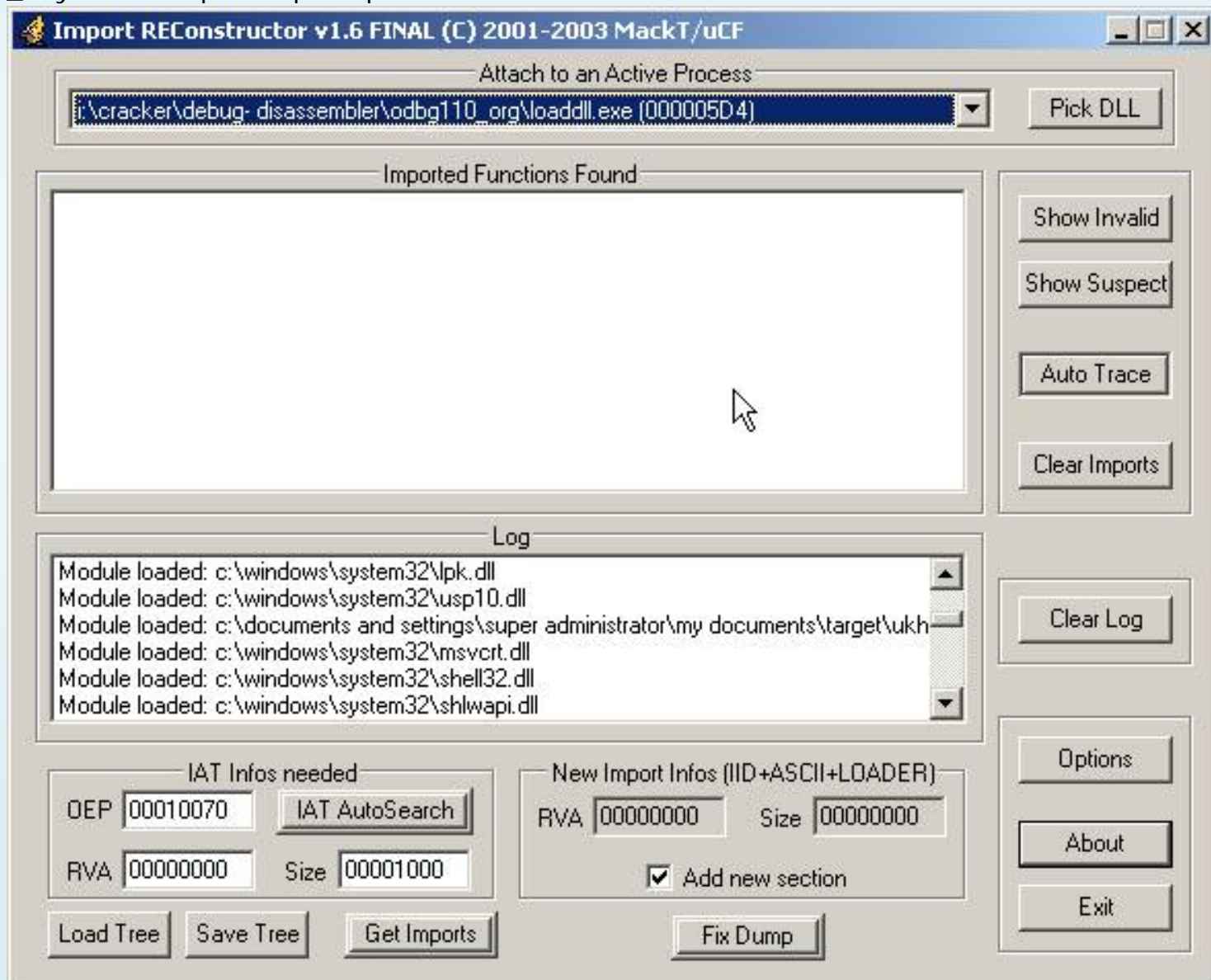
_Right Click in this module, select the full dump:



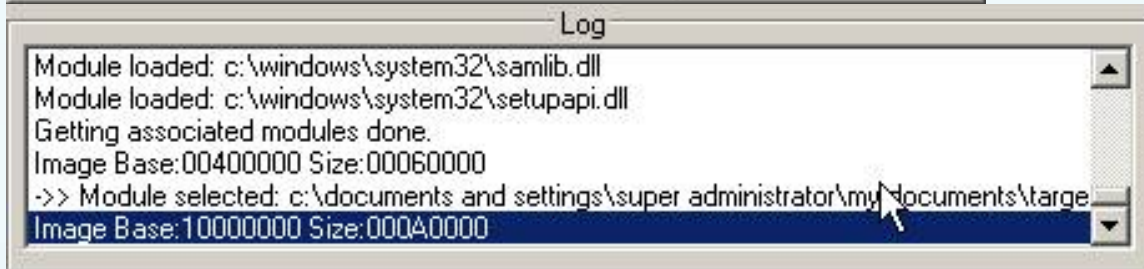
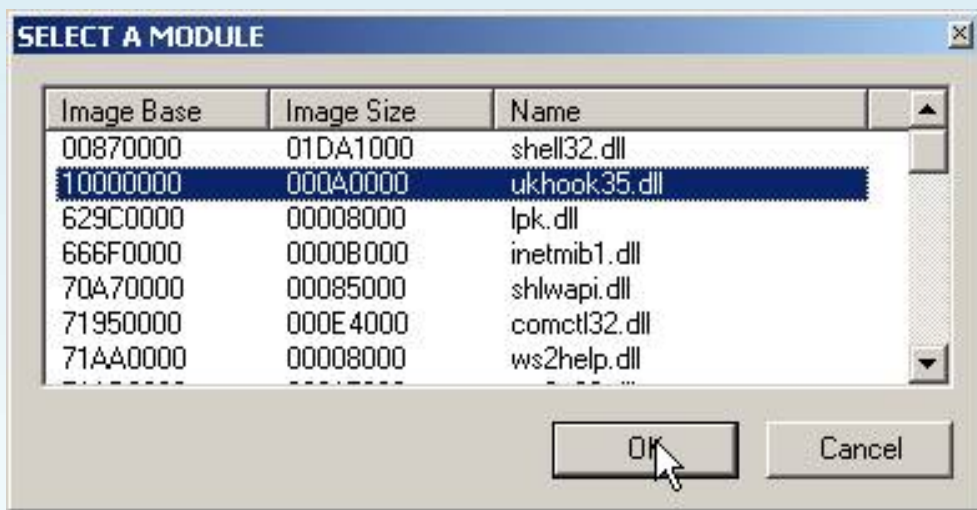
_Nho Select dump dump Intelligent Engine is:



Bay Hours ImpREC open up:



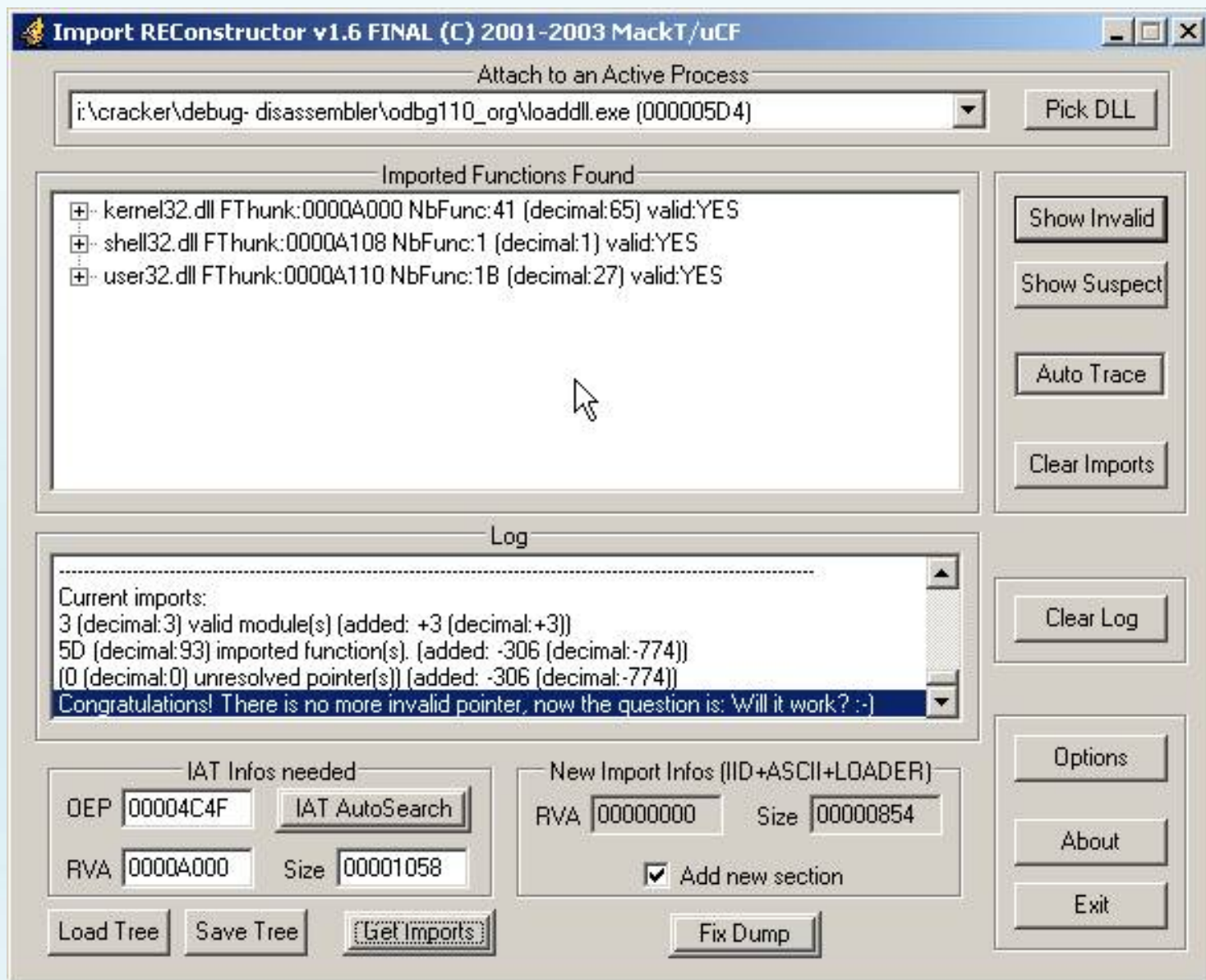
Dll _Click to Pick:



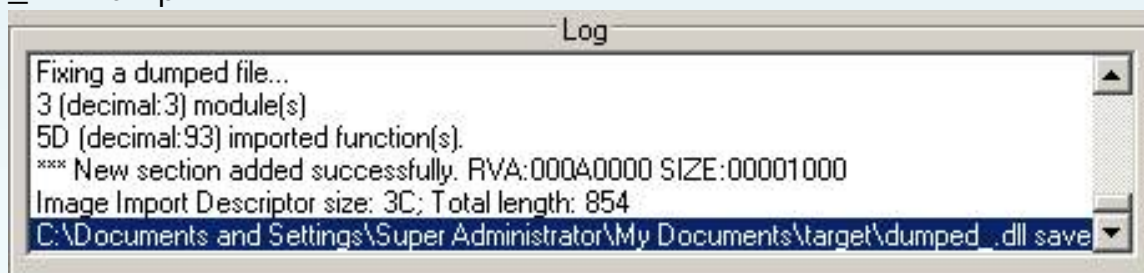
_Dien Information



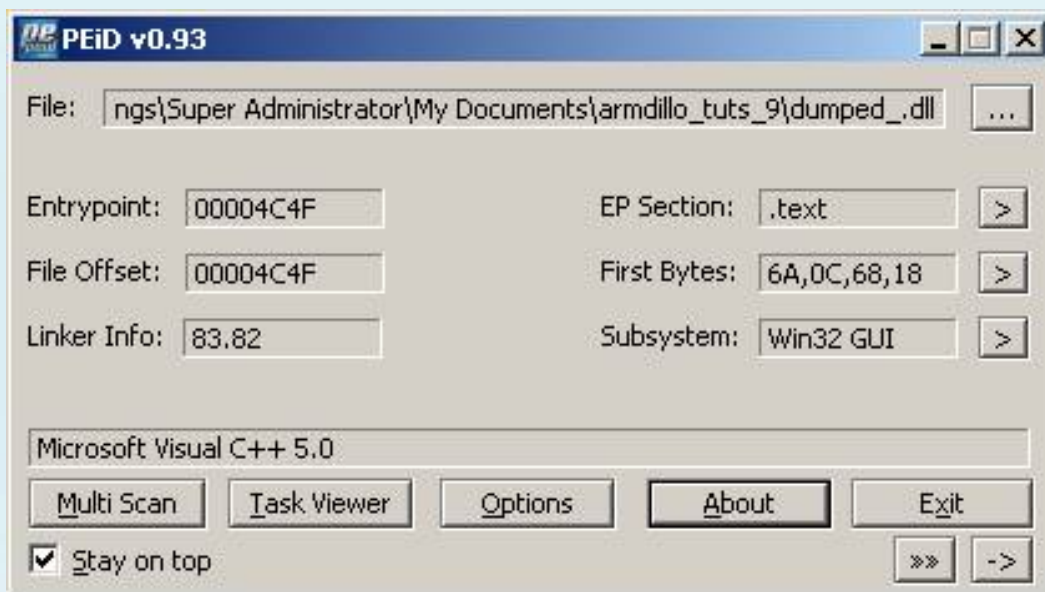
_Get Import:



_Fix Dump:



_Detect:



Rename File dumped.dll to UKHook35.DLL, UNIKEY.EXE run. Run good!



_Unpacked Done!

IV. Conclusion

_For More tuts, please visit <http://tinicat.de/hacnho>

_Bye!

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini, hoadongnoi, dqtn, hosiminh, Nilrem, fly, MaDMAN_H3rCuL3s, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RIF, N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx OillyDBG of the authors and ArmInline

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 29/09/2005)

Reverse Engineering Association

SoftWare

Homepage: www.slysoft.com

Production: SlySoft,

Software: CloneCD v4.3.1.7

Copyright by: Copyright © 2003 SlySoft Inc.. All Rights Reserved.

Type: Time Trial

Packed: 1:23 ASProtect RC4 - 1.3.08.24 -> Alexey Solodovnikov

Language :

Crack Tool: 1:10 OllyDbg, PEiD 0.93, ImportREC v1.6F

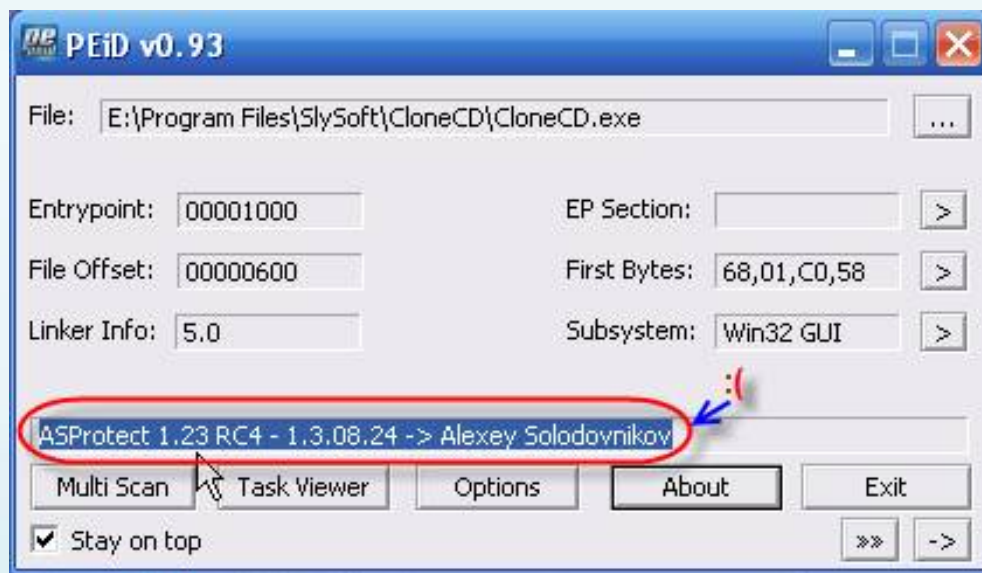
Unpack: Manual

CloneCD v4.3.1.7

CloneCD is the ideal tool to make backup copies of your music-or data CDs, regardless if they are copy protected or not! CloneCD's award-winning user interface copies almost any CD in just a few mouse clicks!

I - Information:

- Hic is soft on this before I have pretty much grace it with it, remember the original thua ngơ ngac after it installed open PEiD to detect them, hic see Asprotect found that both poignant. I replied that SuperNewbie, hiii until now I also still in progress on such teo time :-). Fossick REA in the day that is not reaonline.net as now, the site of grapes and aged on the Net, which is the default Stolen bytes, Anti-Debug, Fix v. dump. V. ..., reading is not available I always close the hiii, and the Soft we also dispose xó not. It did not want to find out what Asprotect, but again the more poignant when later I touched the Armadillo. Is "to avoid having Dura empty coconut shell."
- After several days gác pen today, I earned the tut the CracksLatinos. Riu not dare to dance through the eyes but also for workers mạn written permission for his children are in any situation "tit right eye, the red eye as I left. "
- Hic reinstall programs, use PeiD v0.93 we know the program is packed with ASProtect 1:23 RC4 - 1.3.08.24 -> Alexey Solodovnikov. Hiii they met again later to dispose of many xó but found it difficult to describe lang lang :-). Do not know they love you no longer remember more tolerable.

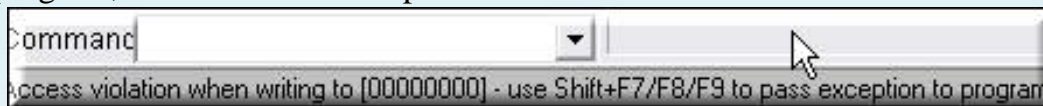


II - Manual Unpacking:

- Load the program in Olly, we will stop here in Olly:

Address	Hex dump	Disassembly	Comment
00401000	68 01C05800	JMPD CloneCD.0058C001	
00401005	E8 01000000	CALL CloneCD.0040100B	
0040100A	C3	RETN	
0040100B	C3	RETN	
0040100C	9F	LAHF	
0040100D	57	PUSH EDI	
0040100E	98	CWDE	
0040100F	FC	CLD	
00401010	AA	STOS BYTE PTR ES:[EDI]	
00401011	A8 BE	TEST AL,0BE	
00401013	F8	CLC	
00401014	05 DDE52305	ADD EAX,523E5D0	
00401019	84D9	TEST CL,BL	

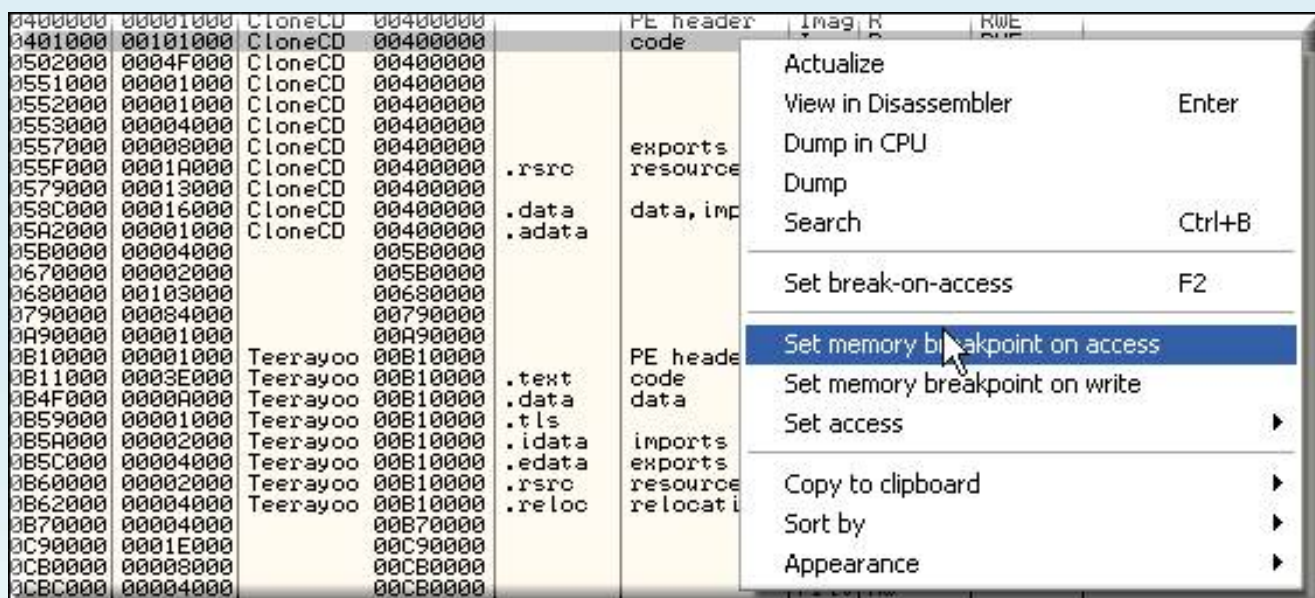
- Press **F9** to Run program, we will Break Exception in first



- Continue to press **Shift + F9** to bypass all Exception until the Run toan.Dong complete the count the number of clicks, in my case is 27 times the Run program completely. Now press **Ctrl + F2** Restart to the program, press **F9** to Run. Then press **Shift + F9** 26 times we here in Olly:

Address	Hex dump	Disassembly	Comment
00CA39EC	3100	XOR DWORD PTR DS:[EAX],EAX	
00CA39EE	64:8F05 000000	POP DWORD PTR FS:[0]	
00CA39F5	58	POP EAX	
00CA39F6	833D B07ECA00	CMP DWORD PTR DS:[CA7EB0],0	
00CA39FD	74 14	JE SHORT 00CA3A13	
00CA39FF	6A 0C	PUSH 0C	
00CA3A01	B9 B07ECA00	MOV ECX,0CA7EB0	
00CA3A06	8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]	
00CA3A09	BA 04000000	MOV EDX,4	
00CA3A0E	E8 2DD1FFFF	CALL 00CA0B40	

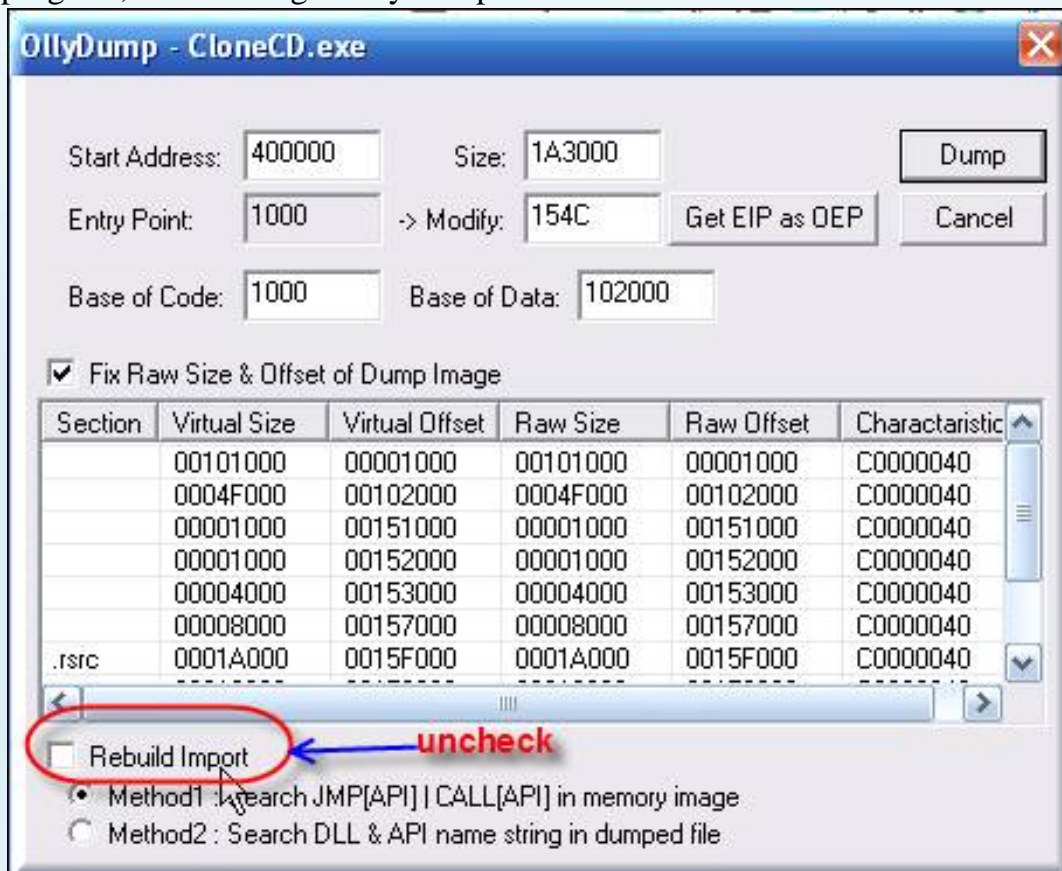
- Press **Shift + F7** 1 times, then press **Alt + M** to open the window Memory. We will set a **Break point** in the Code on Access section as follows:



- Press **F9** to Run program. We will stop at the OEP program.

Address	Hex dump	Disassembly	Comment
0040154C	EB 10	JMP SHORT CloneCD.0040155E	
0040154E	66:623A	BOUND DI,DWORD PTR DS:[EDX]	
00401551	43	INC EBX	
00401552	2B2B	SUB EBP,DWORD PTR DS:[EBX]	
00401554	48	DEC EAX	
00401555	4F	DEC EDI	
00401556	4F	DEC EDI	
00401557	4B	DEC EBX	
00401558	90	NOP	
00401559	E9 98205000	JMP 009035F6	
0040155E	A1 8B205000	MOV EAX,DWORD PTR DS:[50208B]	

- Here, drag up a bit, and observed that the program does not have the missing bytes so we concluded that in the case of this child, we do not need to search for "Stolen Bytes". OEP our time is $40154C - 400,000 = 154C$. Now we will dump the program, use the Plugin OllyDump we are as follows:



- Dump and Save the file with a name any. In my case is "unpacked.exe". Okie, back Olly, press **F7** to Trace a

paragraph until we Trace Call to order the first one to be in position after Olly:

Address	Hex dump	Disassembly
00500FBF	FF 25 C0355500	JMP DWORD PTR DS:[5535C0]
00500FC0	FF 25 C4355500	JMP DWORD PTR DS:[5535C4]
00500FCH	FF 25 C8355500	JMP DWORD PTR DS:[5535C8]
00500FD0	FF 25 CC355500	JMP DWORD PTR DS:[5535CC]
00500FD6	FF 25 D0355500	JMP DWORD PTR DS:[5535D0]
00500FDC	FF 25 D4355500	JMP DWORD PTR DS:[5535D4]
00500FE2	FF 25 D8355500	JMP DWORD PTR DS:[5535D8]
00500FE8	FF 25 DC355500	JMP DWORD PTR DS:[5535DC]
00500FEE	FF 25 E0355500	JMP DWORD PTR DS:[5535E0]
00500FF4	FF 25 E4355500	JMP DWORD PTR DS:[5535E4]
00500FFA	FF 25 E8355500	JMP DWORD PTR DS:[5535E8]
00501000	FF 25 EC355500	JMP DWORD PTR DS:[5535EC]
00501006	FF 25 F0355500	JMP DWORD PTR DS:[5535F0]
0050100C	FF 25 F4355500	JMP DWORD PTR DS:[5535F4]
00501012	FF 25 F8355500	JMP DWORD PTR DS:[5535F8]
00501018	FF 25 FC355500	JMP DWORD PTR DS:[5535FC]
0050101E	FF 25 00365500	JMP DWORD PTR DS:[553600]
00501024	FF 25 04365500	JMP DWORD PTR DS:[553604]

- This order is the first call to IAT, see the space where we see: DS: [005535C0] = 00CA1C64, does not exist a function for this jump in orders. Scroll mouse over until we meet orders **JMP DWORD PTR DS: [55xxxx]** first. In my case are:

00500D8C	FF 25 A8315500	JMP DWORD PTR DS:[5531A8]	elbycdio.ElbyCDIO_CloseTarget
00500D90	FF 25 AC315500	JMP DWORD PTR DS:[5531AC]	elbycdio.ElbyCDIO_DeInitScsi
00500D96	FF 25 B0315500	JMP DWORD PTR DS:[5531B0]	elbycdio.ElbyCDIO_DisablePowerSaving
00500DA0	FF 25 B4315500	JMP DWORD PTR DS:[5531B4]	elbycdio.ElbyCDIO_Eject
00500DA4	FF 25 B8315500	JMP DWORD PTR DS:[5531B8]	elbycdio.ElbyCDIO_EnablePowerSaving
00500DA8	FF 25 BC315500	JMP DWORD PTR DS:[5531BC]	elbycdio.ElbyCDIO_ExDoScsiIO
00500DB0	FF 25 C0315500	JMP DWORD PTR DS:[5531C0]	elbycdio.ElbyCDIO_GetDllVersion
00500DB6	FF 25 C4315500	JMP DWORD PTR DS:[5531C4]	elbycdio.ElbyCDIO_GetDriveName
00500DBC	FF 25 C8315500	JMP DWORD PTR DS:[5531C8]	elbycdio.ElbyCDIO_GetDriverVersion
00500DC2	FF 25 CC315500	JMP DWORD PTR DS:[5531CC]	elbycdio.ElbyCDIO_GetFileVersion
00500DC8	FF 25 D0315500	JMP DWORD PTR DS:[5531D0]	elbycdio.ElbyCDIO_GetMaxTransferSize
00500DCE	FF 25 D4315500	JMP DWORD PTR DS:[5531D4]	elbycdio.ElbyCDIO_GetOSVersion
00500DD4	FF 25 D8315500	JMP DWORD PTR DS:[5531D8]	elbycdio.ElbyCDIO_InitScsiAspi
00500DDA	FF 25 DC315500	JMP DWORD PTR DS:[5531DC]	elbycdio.ElbyCDIO_LockTarget

- At 00500D8C based on, we must click and select "**Follow in dump -> Memory Address.**" In the window dump we will be as follows:

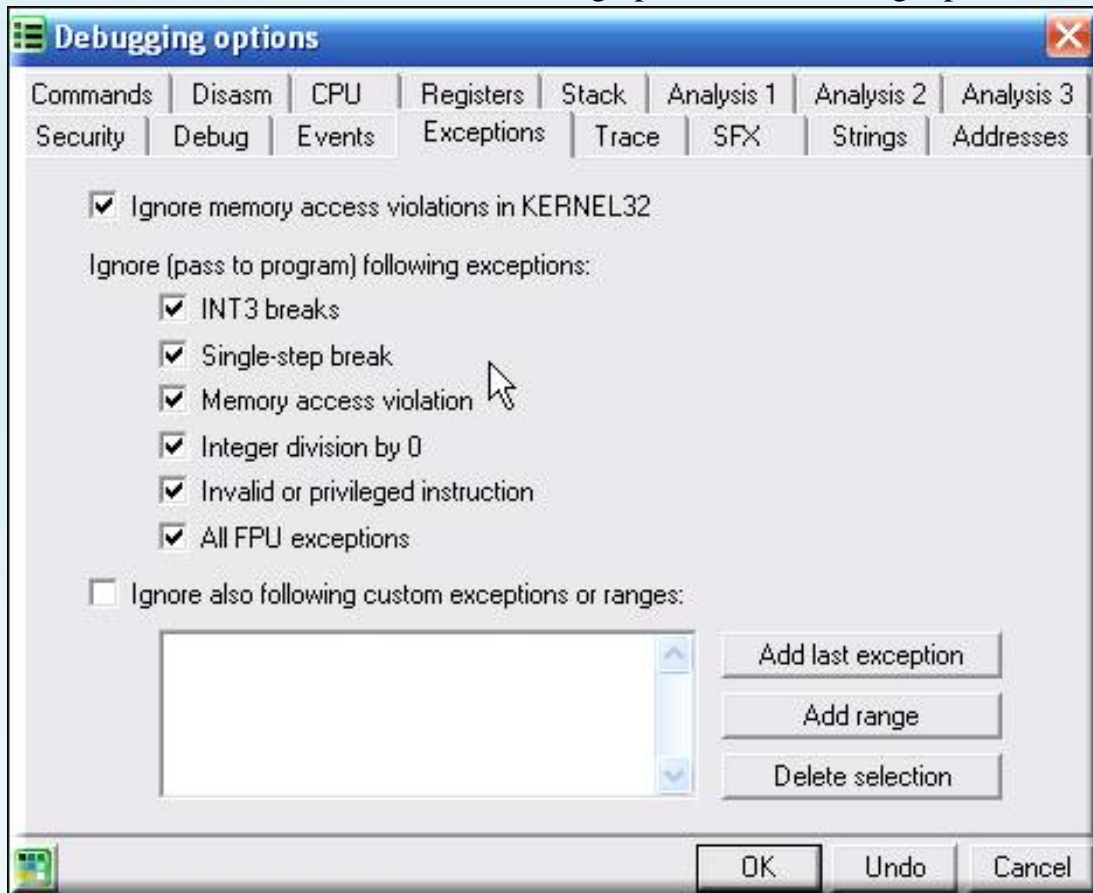
Address	Hex dump	ASCII
00553188	23 21 0E D1 16 5E 00 72	#!&^~.r
00553190	B1 EC 15 26 90 B7 3C FC	~w&En<"
00553198	BF 58 FD 16 A9 78 9D 22	~X?~lX~"
005531A0	BA C8 09 1B F1 8D E1 54	^~+~l~T
005531A8	00 34 00 10 C0 25 00 10	.4.~%~>
005531B0	00 37 00 10 E0 25 00 10	.?~%~>
005531B8	00 37 00 10 40 25 00 10	.?~%~>
005531C0	00 16 00 10 D0 1E 00 10	.~?~%~>
005531C8	30 17 00 10 00 16 00 10	0~?~%~>
005531D0	30 11 00 10 F0 18 00 10	0~?~%~>
005531D8	50 3C 00 10 A0 18 00 10	P<~?~%~>
005531E0	30 22 00 10 90 1E 00 10	0~?~%~>
005531E8	10 26 00 10 00 19 00 10	>~?~%~>
005531F0	90 2B 00 10 60 3C 00 10	E+~?~%~>
005531F8	B1 F3 FD E4 D1 5D 31 87	~&~?~%~>

- To make sure the view 005531A8 have to address is the start of the IAT not we do the following. At 4 bytes 005531A0 selected, press **Ctrl + R** to see this reference to any order, the results I have are empty, similar to 005531A4. So the conclusion is 005531A8 address start of IAT. Continue on the window of CPU Olly drag us down to the end position of IAT. We are as follows:

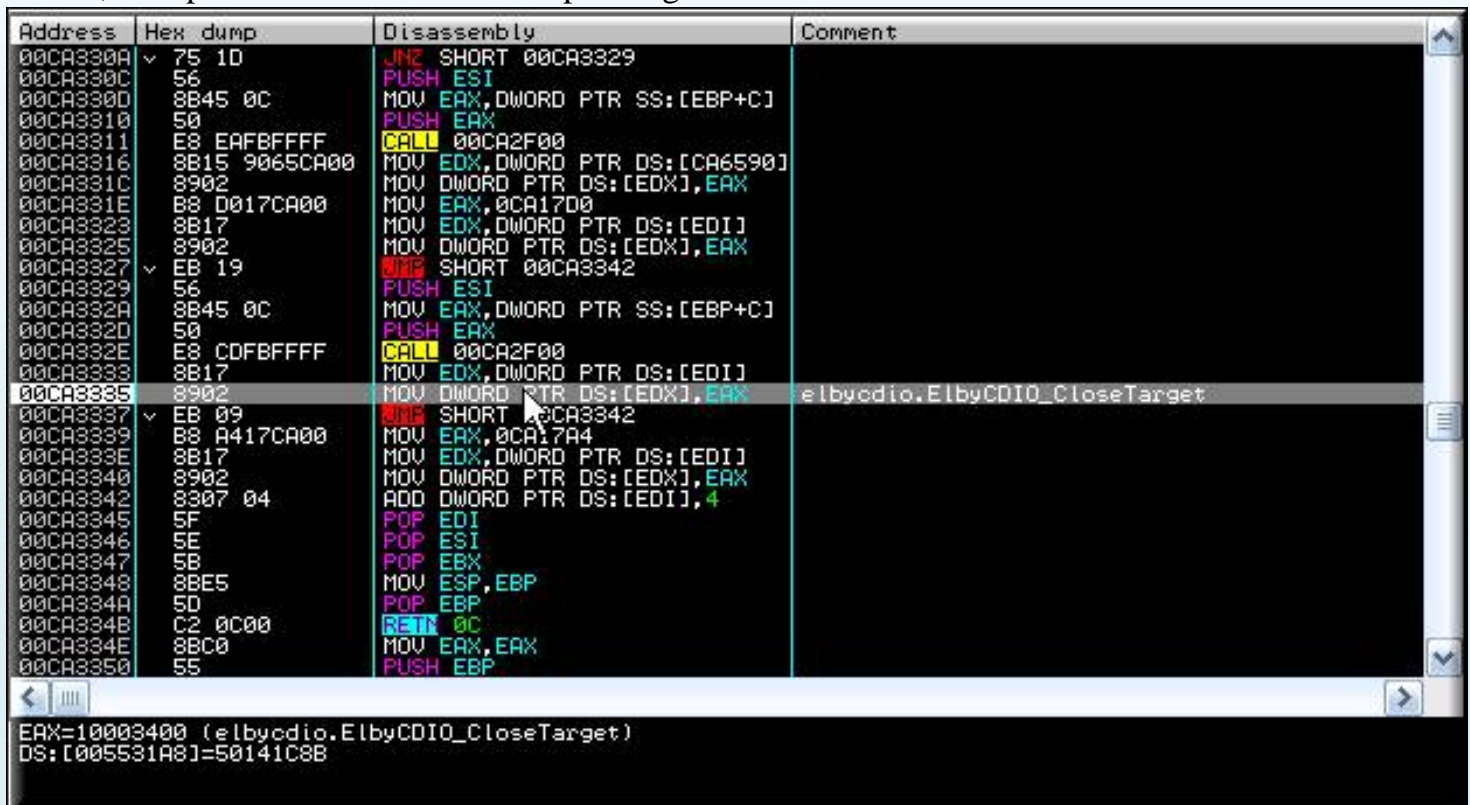
Address	Hex dump	Disassembly	Comment
00501958	FF 25 B0415500	JMP DWORD PTR DS:[5541B0]	oleaut32.SafeArrayGetElement
0050195E	FF 25 B4415500	JMP DWORD PTR DS:[5541B4]	oleaut32.SafeArrayGetLBound
00501964	FF 25 B8415500	JMP DWORD PTR DS:[5541B8]	oleaut32.SafeArrayGetUBound
0050196A	FF 25 BC415500	JMP DWORD PTR DS:[5541BC]	oleaut32.SafeArrayPtrOfIndex
00501970	FF 25 C0415500	JMP DWORD PTR DS:[5541C0]	oleaut32.SafeArrayPutElement
00501976	FF 25 C4415500	JMP DWORD PTR DS:[5541C4]	oleaut32.SafeArrayRedim
0050197C	FF 25 C8415500	JMP DWORD PTR DS:[5541C8]	oleaut32.SysAllocStringLen
00501982	FF 25 CC415500	JMP DWORD PTR DS:[5541CC]	oleaut32.SysFreeString
00501988	FF 25 D0415500	JMP DWORD PTR DS:[5541D0]	oleaut32.SysReAllocStringLen
0050198E	FF 25 D4415500	JMP DWORD PTR DS:[5541D4]	oleaut32.VariantChangeType
00501994	FF 25 D8415500	JMP DWORD PTR DS:[5541D8]	oleaut32.VariantClear
0050199A	FF 25 DC415500	JMP DWORD PTR DS:[5541DC]	oleaut32.VariantCopy
005019A0	FF 25 E0415500	JMP DWORD PTR DS:[5541E0]	oleaut32.VariantCopyInd
005019A6	FF 25 E4415500	JMP DWORD PTR DS:[5541E4]	oleaut32.VariantInit

- Tolerable in 005019A6 Follow me choose **Print dump -> Memory Address.** You can check them 005541E4 see is the address last IAT not. After the process we have been on the length of the IAT is 005541E4 - 005531A8 = 103C. Hic hard too :-)

- Okie Next we must find the Magic Call. Press **Ctrl + F2**, press **F9**, we will stop at the first exception. Here, we too Memory dump window, press **Ctrl + G** to enter **005531A8**. Select the first 4 bytes and set a **Breakpoint on write**. Press **Shift + F7** 1 times, then select the following options in the Debug Options:



- Click OK, then press **F9** to Run. Continue pressing **F9** a few times we will Ice here:



- Scroll mouse over a little to find the location of Magic Call we will be as follows:

Address	Hex dump	Disassembly	Comment
00CA32A4	E8 97D8FFFF	CALL 00CA0B40	
00CA32A9	80B5 FFEFFFFF	LEA ESI,DWORD PTR SS:[EBP-101]	
00CA32AF	56	PUSH ESI	
00CA32B0	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]	
00CA32B3	50	PUSH EAX	
00CA32B4	E8 47FCFFFF	CALL 00CA2F00	
00CA32B9	E8 7EFEFFFF	CALL 00CA317C	magic call
00CA32BE	8B17	MOV EDX,DWORD PTR DS:[EDI]	
00CA32C0	8902	MOV DWORD PTR DS:[EDX],EAX	
00CA32C2	EB 7E	SHORT 00CA3342	
00CA32C4	83FB 06	CMP EBX,6	
00CA32C7	74 05	JE SHORT 00CA32CE	signature
00CA32C9	83FB 03	CMP EBX,3	
00CA32CC	75 37	JNZ SHORT 00CA3305	
00CA32CE	8A06	MOV AL,BYTE PTR DS:[ESI]	
00CA32D0	8B45 FF	MOV BYTE PTR SS:[EBP-1],AL	
00CA32D3	46	INC ESI	
00CA32D4	33C9	XOR ECX,ECX	
00CA32D6	8A4D FF	MOV CL,BYTE PTR SS:[EBP-1]	
00CA32D9	80B5 FFEFFFFF	LEA EAX,DWORD PTR SS:[EBP-101]	
00CA32DF	8BD6	MOV EDI,ESI	
00CA32E1	E8 6A1FFFFF	CALL 00C95250	
00CA32E6	6A 0A	PUSH 0A	

- The address is Magic **00CA32B9** Call, remember this. Now we Restart the Olly, configure the Debug Options -> Exceptions to the 2 first choice, uncheck the first time, press **F9** to run, in CPU window press **Ctrl + G** and enter the address of Magic Call. At the Magic Call BP is set by pressing **F2** and submit Call this command:

Address	Hex dump	Disassembly	Comment
00CA32B9	90	NOP	
00CA32BA	90	NOP	
00CA32BB	90	NOP	
00CA32BC	90	NOP	
00CA32BD	90	NOP	
00CA32BE	8B17	MOV EDX,DWORD PTR DS:[EDI]	
00CA32C0	8902	MOV DWORD PTR DS:[EDX],EAX	
00CA32C2	EB 7E	SHORT 00CA3342	

- Time to configure the Debug Options as I said before, according to press **Shift + F7** and finally press **F9** Ice in our seats that we have set BP.

- Next we Trace with **F7** until we observed that the following signs:

Address	Hex dump	Disassembly	Comment
00CA354D	5B	POP EBX	00CBC672
00CA354E	EB CE	SHORT 00CA351E	
00CA3550	61	POPAD	
00CA3551	E8 3A000000	CALL 00CA3590	
00CA3556	68 5F35CA00	PUSH 0CA355F	
00CA355B	FF0424	INC DWORD PTR SS:[ESP]	
00CA355E	C3	RET	
00CA355F	BC 8B44240C	MOV ESP,0C24448B	

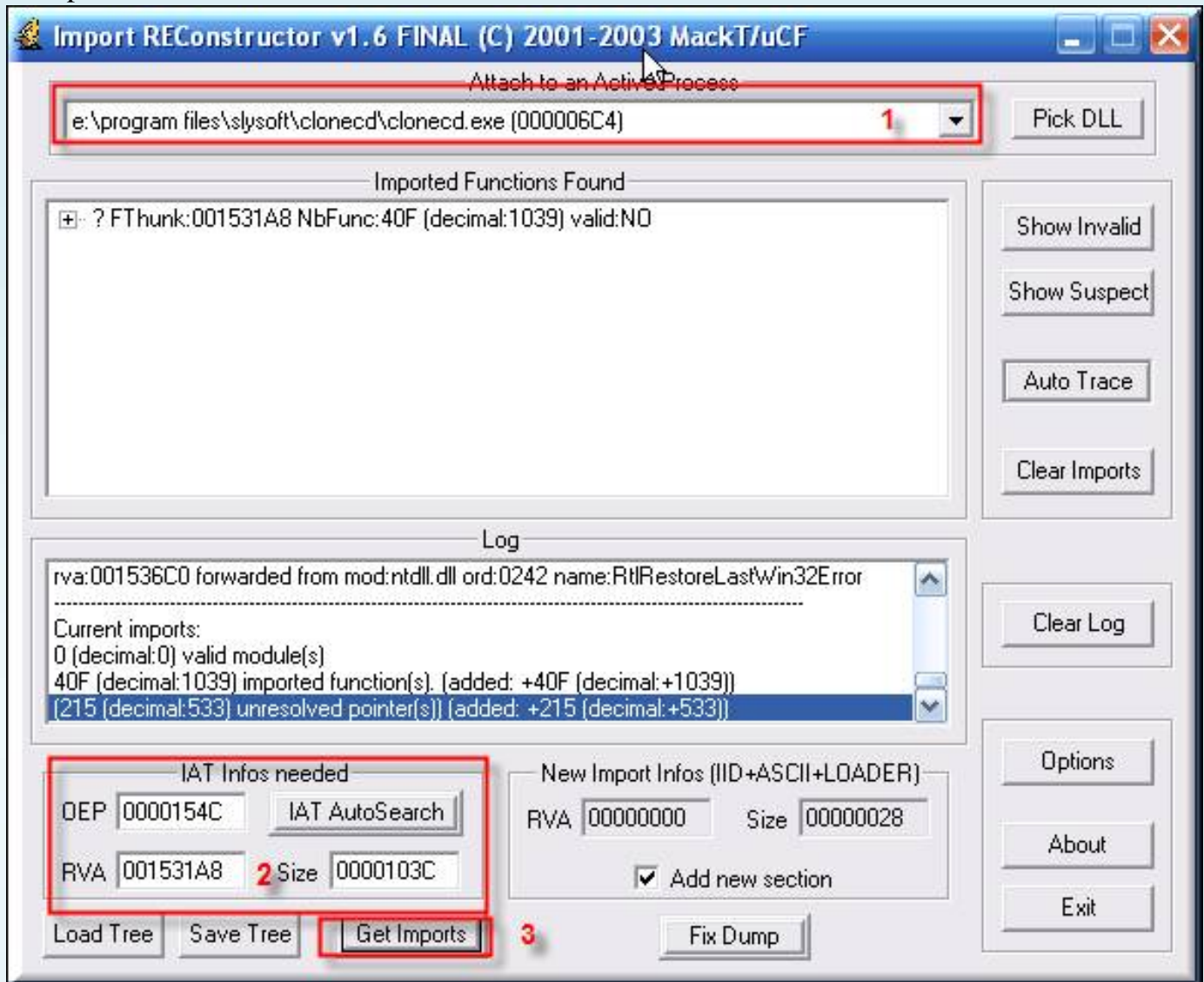
- OK we put in place a BP has ordered **POPAD**. Press **F9** to run, we at Ice Magic's Call, BP delete this and continue to press **F9** we will stop at **POPAD**. Deleted at the BP POPAD, configure the Debug Options -> Exceptions to top 2 choices, click "-" back in place to submit. Here we do the following:

Address	Hex dump	Disassembly	Comment
00CA32B9	90	NOP	
00CA32BA	90	NOP	
00CA32BB	90	NOP	
00CA32BC	90	NOP	
00CA32BD	90	NOP	
00CA32BE	8B17	MOV EDX,DWORD PTR DS:[EDI]	
00CA32C0	8902	MOV DWORD PTR DS:[EDX],EAX	
00CA32C2	EB 7E	SHORT 00CA3342	
00CA32C4	83FB 06	CMP EBX,6	
00CA32C7	74 05	JE SHORT 00CA32CE	
00CA32C9	83FB 03	CMP EBX,3	

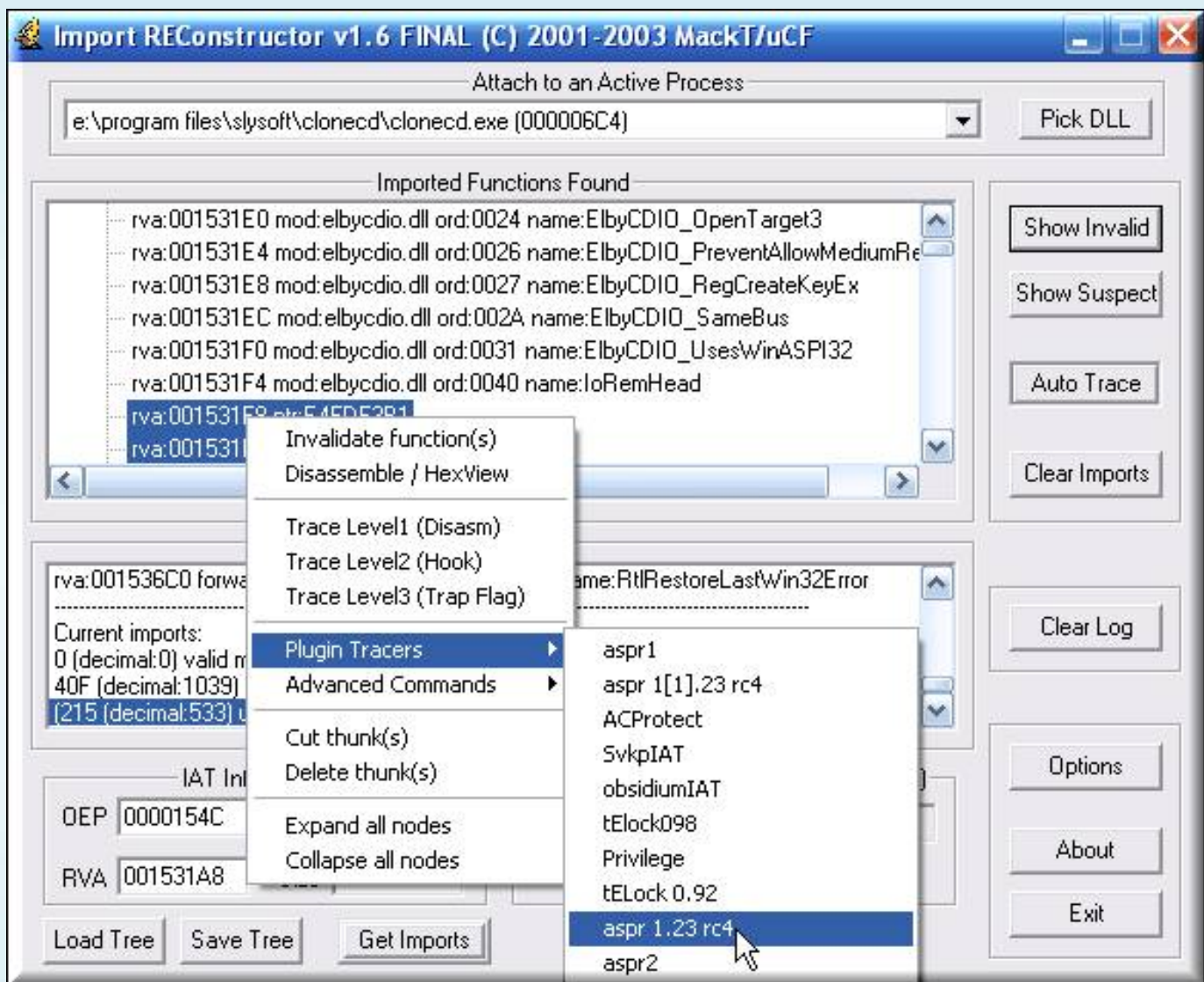
- Press **F9** to Run, then press **Shift + F9** 7 (8 if the Run entirely). Next press **Shift + F7**, **Alt + M** to open the Memory Window, located at BP Code Section. Press **F9**, we stopped at the OEP.

Address	Hex dump	Disassembly	Comment
00401546	EB 10	JMP SHORT CloneCD.0040155E	
00401547	66:623A	BOUND DI,DWORD PTR DS:[EDX]	
00401551	43	INC EBX	
00401552	2B2B	SUB EBP,DWORD PTR DS:[EBX]	
00401554	48	DEC EAX	
00401555	4F	DEC EDI	
00401556	4F	DEC EDI	
00401557	4B	DEC EBX	
00401558	90	NOP	
00401559	E9 98205000	JMP 009035F6	
0040155E	A1 8B205000	MOV EAX,DWORD PTR DS:[50208B]	

- Ac I không coming here this month rồi that, a few more sure I again beating the keyboard. Fortunately we are here to Fix IAT was then. **ImportREC** open up, select Process and enter information like Figure below and click Get Imports:



- Click Show Invalid, we also see many Invalid Thunks. Mouse right at the Invalid and use plug-in:



- Is still a lot Unresolved Pointers. However we Cut Thunks and conducted Fix IAT. Select File unpacked.exe to Fix IAT. Import Rec will save with the name after unpacked_.exe.



- Hic is moments from the best, hiii cam see any mouse that falter:). Close ImportRec and Olly. Run the file to try unpacked_.exe. Passable coffee too, roaring run it:



- Use PeiD to check again, hiii we are as follows:



III - End of tut:

KIENMANOWAR

- Finished - [September 20, 2005](#)

---++---==[**Greatz thanks to**]==---++---

- Thank to my family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHQCrker, the_Lighthouse, Hoadongnoi, Nini, Merc ... all REA's members, HacNho, RongChauA, Deux, tlandn, dqtn, **CracksLatinos**, ARTEAM all my friend, and you.

Special thanks to ---++---==[]==---++---

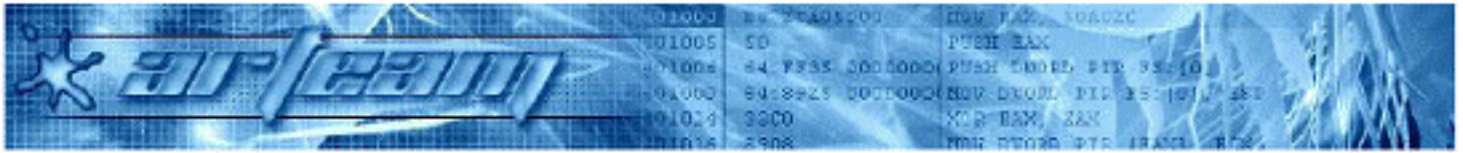
coruso_trac, patmsvn, trm_tr v. .. v.. all brothers in VSEC.

>>>> If you have any suggestions, comments or corrections email me: **kienbigmummy [at] gmail.com**

REVERSE ENGINEERING ASSOCIATION

<http://www.reaonline.net>





Inline patching Asprotect 2.x

ThunderPwr (ARTeam)

0. INTRODUCTION

This article is written just to clarify the steps that must be conducted to make technical Inline Patching for the program compressed by Asprotect; angle from a more general, we can understand more about the target Inline Patching.

Inline patching is not a simple way that can be used to modify behavior of the program, technical because this requires a good knowledge about the protection of the compression, but after gradually coming to have be easily expanded with the compression of the other.

To better understand this technique, you can use a real example, I try to Chord Pickout 1.5. You can download here www.chordpickout.com

All information in this tut to serve in academic research on protection from the protection.

All the information contained in this tutorial does not have to be used in order to use in illegal way copyrighted and protected software. All the info in this tutorial must be used in order to only better know the protection scheme used from this particular protection. It is not encouraged therefore to the use of these information for illicit or various scopes different than the pure study.

1st PACKER Analysis (reversing stage)

You use the scanner to check enforcement program is compressed or not



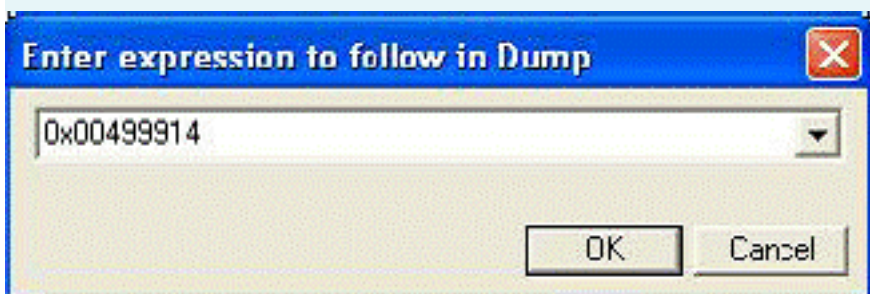
This is often to do as well as to consider a direct Entry point structure similar to each Asprotect target.

00401000	68 01B04E00	PUSH Chordpic.004EB001
00401005	E9 01000000	CALL Chordpic.0040100B
0040100A	C3	RETN
0040100B	C3	RETN
0040100C	AA	DB AA
0040100D	1F	DB 1F
0040100E	5A	DB 5A
0040100F	D0	DB D0
00401010	00	DB 00
00401011	D3	DB D3

We must find the OEP of the application. This depends on the packer. For Asprotect we use methods and exceptions as follows:

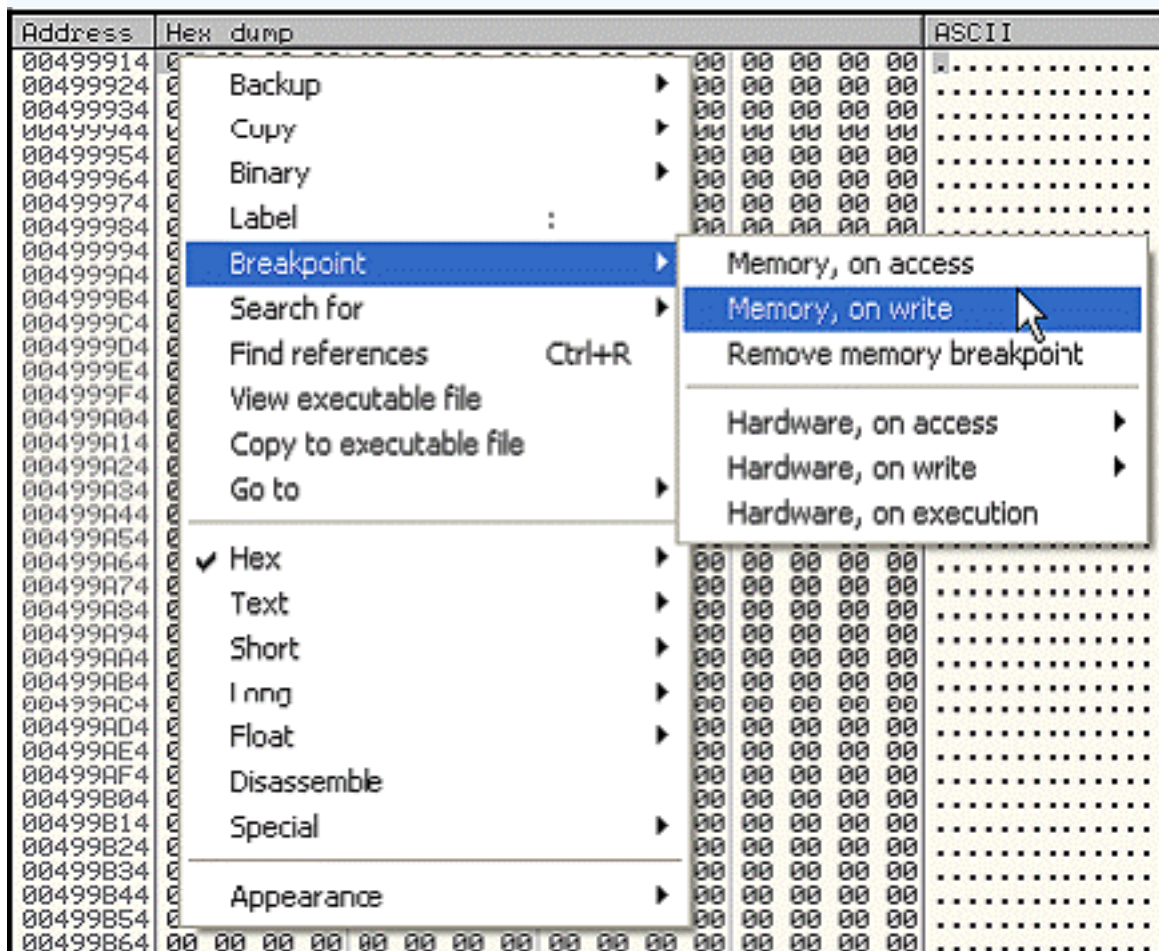
00499914	55	PUSH EBP	OEP
00499915	8BEC	MOV EBP,ESP	
00499917	83C4 E0	ADD ESP,-14	
0049991A	53	PUSH EBX	
0049991B	33C0	XOR EAX,EAX	
0049991D	8945 EC	MOV DWORD PTR SS:[EBP-14],EAX	
00499920	B8 24964900	MOV EAX,chordpic.00499624	
00499925	E8 86CDF6FF	CALL chordpic.00406680	
0049992A	8B1D F0E04900	MOV EBX,DWORD PTR CS:[49E0F0]	
00499930	33C0	XOR EAX,EAX	
00499932	55	PUSH EBP	
00499933	68 4E9A4900	PUSH chordpic.00499A4E	
00499938	64:FF30	PUSH DWORD PTR FS:[EAX]	chordpic.0049FBC0

Now restart target and look at the window dump Olly, press Ctr + G and to address: 0x00499914, we try to review the area remember this:

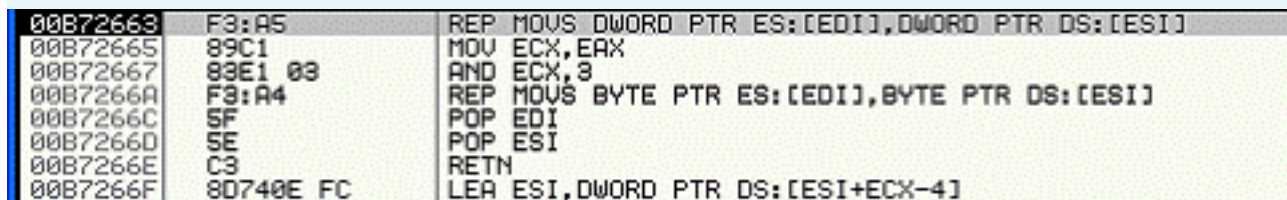


All addresses are from OEP contains 00 BYTE because the code of the program to decompress

from the packer stub, place a breakpoint on memory, to write on the byte at the address identified:



Now press ALT + O to view Options and check all the exceptions (exceptions), then press Shift + F9. After a while Olly will stop at this code:



Press F8 to implement in the loop, then look in the window to dump:

Address	Hex dump	ASCII
00499914	55 8B EC 83 C4 EC 53 33 C0 89 45 EC B8 24 96 49	Uÿā-ÿS3ˆEÿ@šÿI
00499924	00 E8 86 CD F6 FF 8B 1D F0 E0 49 00 33 C0 55 68	.pā÷ i#-0I.3ˆUh
00499934	4E 9A 49 00 64 FF 30 64 89 20 8B 03 E8 5B B1 FC	NŮI.d 0dē iŏp[ŏŏ
00499944	FF 8B 00 B8 DE 49 00 8B 03 8B 15 DC E8 48 00 E8	i.0iI.iŏiSˆpH.p
00499954	60 B1 FC FF 8B 0D 98 E2 49 00 8B 03 8B 15 B4 B4	'ŏŏ i.ŏ0I.iŏiSˆi
00499964	48 00 E8 4D B1 FC FF 8B 0D AC DF 49 00 8B 03 8B	H.pŏŏŏ i.ˆ=I.iŏi
00499974	15 D4 BE 48 00 E8 3A B1 FC FF 8B 0D B0 E1 49 00	SēH.p:ŏŏŏ i.ŏŏpI.
00499984	8B 03 8B 15 D4 B8 48 00 E8 27 B1 FC FF 8B 0D EC	iŏiSē0H.p'ŏŏŏ i.ŏ
00499994	DF 49 00 8B 03 8B 15 28 C1 48 00 E8 14 B1 FC FF	=I.iŏiS(-H.pŏŏŏŏ
004999A4	8B 0D 20 E1 49 00 8B 03 8B 15 D8 CD 48 00 E8 01	i. pI.iŏiSˆi=H.p0
004999B4	B1 FC FF 8B 0D 40 E2 49 00 8B 03 8B 15 78 D0 48	ŏŏŏ i.00I.iŏiSˆx3H
004999C4	00 E8 EE 80 FC FF 8B 0D 98 DE 49 00 8B 03 8B 15	.p-ŏŏŏ i.ŏŏiI.iŏiS
004999D4	8C E3 48 00 E8 DB B0 FC FF E8 86 90 F6 FF 85 C0	#0H.pŏŏŏŏ pāē÷ āˆ
004999E4	7E 4B 8D 55 EC B8 01 00 00 00 E8 D5 90 F6 FF 83	"KlUŏŏŏ...p'ē÷ ā
004999F4	7D EC 00 74 38 A1 B8 DE 49 00 8B 0D 8D 90 58 04	ŏŏ.t8i0iI.i. iēX+
00499A04	00 00 B8 01 00 00 00 E8 B8 90 F6 FF A1 B8 DE 49	..00...p0ē÷ i0iI
00499A14	00 8B 00 E8 BC 62 FF FF 84 C0 75 11 A1 B8 DE 49	.i.pˆb āˆu◀i0iI
00499A24	00 8B 00 05 58 04 00 00 E8 E7 AD F6 FF 8B 03 E8	.i.ˆX+..pŏi÷ iŏp
00499A34	00 B1 FC FF 33 C0 5A 59 59 64 89 10 68 55 9A 49	.ŏŏ 3ˆZYVdēhUŮI
00499A44	00 8D 45 EC E8 CB AD F6 FF C3 E9 45 A7 F6 FF EB	.iEÿŏŏi÷ iŮE2÷ Ů
00499A54	F0 5B E8 49 AC F6 FF 90 00 00 00 00 00 00 00	-[ŏIˆ÷ ē.....
00499A64	00 00 00 00 00 00 00 00 00 00 00 00 00 00

Disassembled view:

Address	Hex dump	ASCII
00499914	55 8B EC 83 C4 EC 53 33 C0 89 45 EC B8 24 96 49	Uÿā-ÿS3ˆEÿ@šÿI
00499924	00 33 C0 55 68	.pā÷ i#-0I.3ˆUh
00499934	03 E8 5B B1 FC	NŮI.d 0dē iŏp[ŏŏ
00499944	DC E8 48 00 E8	i.0iI.iŏiSˆpH.p
00499954	03 8B 15 B4 B4	'ŏŏ i.ŏ0I.iŏiSˆi
00499964	49 00 8B 03 8B	H.pŏŏŏ i.ˆ=I.iŏi
00499974	0D B0 E1 49 00	SēH.p:ŏŏŏ i.ŏŏpI.
00499984	FC FF 8B 0D EC	iŏiSē0H.p'ŏŏŏ i.ŏ
00499994	E8 14 B1 FC FF	=I.iŏiS(-H.pŏŏŏŏ
004999A4	CD 48 00 E8 01	i. pI.iŏiSˆi=H.p0
004999B4	8B 15 78 D0 48	ŏŏŏ i.00I.iŏiSˆx3H
004999C4	00 8B 03 8B 15	.p-ŏŏŏ i.ŏŏiI.iŏiS
004999D4	90 F6 FF 85 C0	#0H.pŏŏŏŏ pāē÷ āˆ
004999E4	D5 90 F6 FF 83	"KlUŏŏŏ...p'ē÷ ā
004999F4	00 8D 90 58 04	ŏŏ.t8i0iI.i. iēX+
00499A04	FF A1 B8 DE 49	..00...p0ē÷ i0iI
00499A14	11 A1 B8 DE 49	.i.pˆb āˆu◀i0iI
00499A24	F6 FF 8B 03 E8	.i.ˆX+..pŏi÷ iŏp
00499A34	10 68 55 9A 49	.ŏŏ 3ˆZYVdēhUŮI
00499A44	45 A7 F6 FF EB	.iEÿŏŏi÷ iŮE2÷ Ů
00499A54	00 00 00 00 00	-[ŏIˆ÷ ē.....
00499A64	00 00 00 00 00
00499A74	00 00 00 00 00
00499A84	00 00 00 00 00
00499A94	00 00 00 00 00
00499AA4	00 00 00 00 00
00499AB4	00 00 00 00 00
00499AC4	00 00 00 00 00
00499AD4	00 00 00 00 00
00499AE4	00 00 00 00 00
00499AF4	00 00 00 00 00
00499B04	00 00 00 00 00
00499B14	00 00 00 00 00
00499B24	00 00 00 00 00
00499B34	00 00 00 00 00
00499B44	00 00 00 00 00
00499B54	00 00 00 00 00
00499B64	00 00 00 00 00
00499B74	00 00 00 00 00

We have:

Address	Hex dump	Disassembly
00499914	55	0B 55
00499915	8B	0B 8B
00499916	EC	0B EC
00499917	83	0B 83
00499918	C4	0B C4
00499919	EC	0B EC
0049991A	53	0B 53
0049991B	33	0B 33
0049991C	C0	0B C0
0049991D	89	0B 89
0049991E	45	0B 45

Code obvious unclear because Olly can not describe (analyze) code from the action of the load the first time because the code is not decompress from packer stub. Therefore, in the code window, click Ctrl + G and the address from where we start describing it (0x00499914) and press Enter, then press Ctrl + A to make the code description:

00499914	55	PUSH EBP
00499915	8BEC	MOV EBP,ESP
00499917	83C4 EC	ADD ESP,-14
0049991A	53	PUSH EBX
0049991B	33C0	XOR EAX,EAX
0049991D	8945 EC	MOV DWORD PTR SS:[EBP-14],EAX
00499920	B8 24964900	MOV EAX,00499624
00499925	E8 86CDF6FF	CALL 004066B0
0049992A	8B1D F0E04900	MOV EBX,DWORD PTR DS:[49E0F0]
00499930	33C0	XOR EAX,EAX
00499932	55	PUSH EBP
00499933	68 4E9A4900	PUSH 00499A4E

Bingo, the code is just decompress and in accordance with which we have found the following steps to use methods of exceptions:

At that point, we know to write code to OEP occur from a region of memory later, press Alt + M:

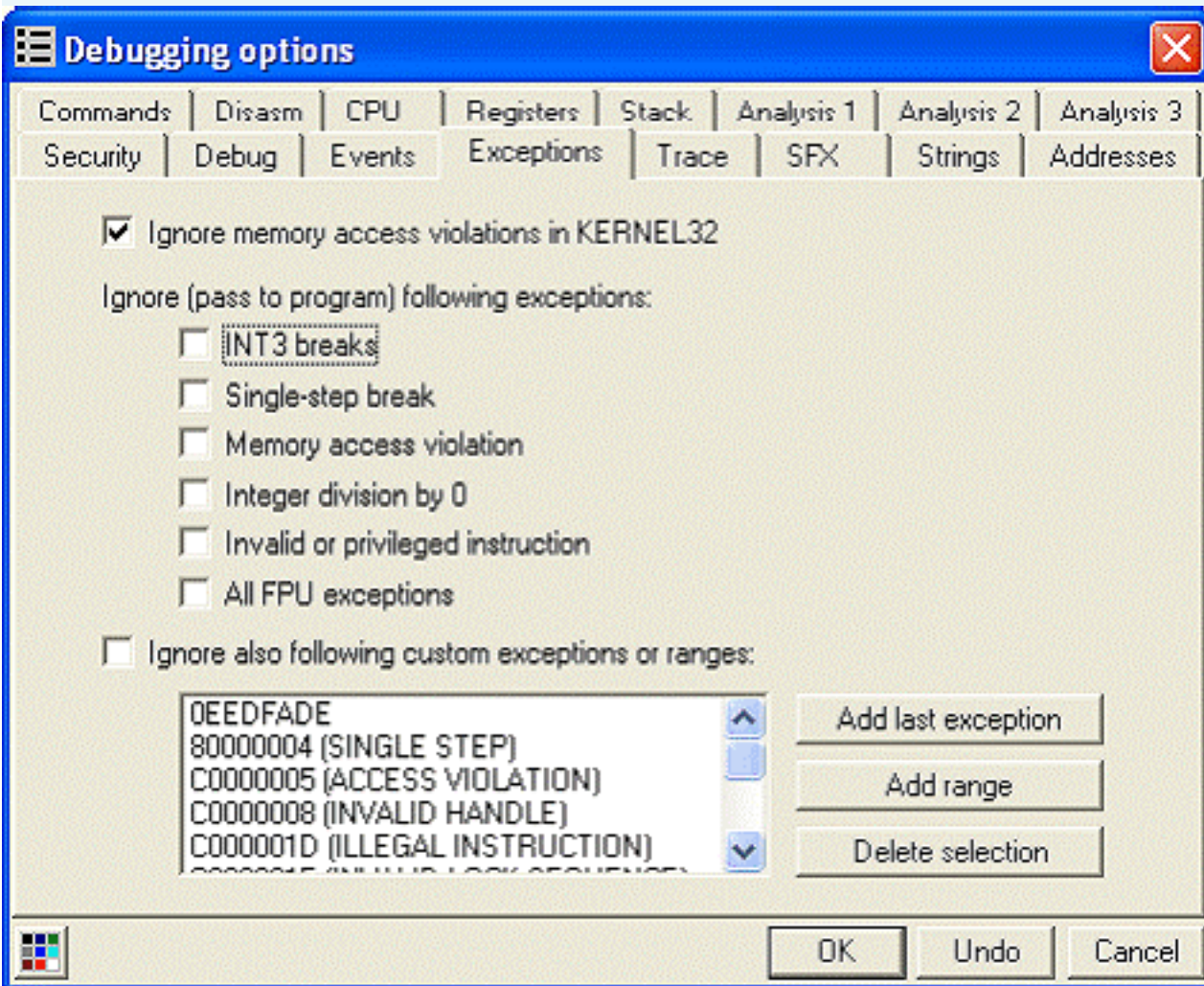
00B40000	00001000				Priv	RWE	RWE	
00B50000	00001000				Priv	RWE	RWE	
00B60000	00001000				Priv	RWE	RWE	
00B70000	00033000				Priv	RWE	RWE	
00B80000	00004000				Priv	RW		
00B88000	0009C000				Priv	RW		
00CB0000	00050000				Map	R	R	
00D00000	00001000				Priv	RWE	RWE	

The region is not dependent on the existing section on the disk because it is not created in the first PE-header, making this the packer demand system memory (active) and then fill code that will be thi.Boi areas because of this memory will be installed at run time (runtime) so we can not find it when the destination (target) to load the Entry Point, but we know when and in time which code is located to explore it through Inline code. To do this, we must know is simply the reverse of the packer.

The first will then search for some of the stub from EntryPoint Asprotect where occur simultaneously to save to memory for the first time to locate.

Restart the destination Ctrl + F2.

Uncheck all exceptions:



Click Ctrl + G and on VirtualAlloc (API function is required levels of memory to the system, the API is used by many the packer and of course that Asprotect. Set breakpoint as follows:

7C809A81	8BFF	MOV EDI,EDI	API Entry Point
7C809A83	55	PUSH EBP	
7C809A84	8BEC	MOV EBP,ESP	
7C809A86	FF75 14	PUSH DWORD PTR SS:[EBP+14]	
7C809A89	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
7C809A8C	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
7C809A8F	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
7C809A92	6A FF	PUSH -1	
7C809A94	E8 09000000	CALL kernel32.VirtualAllocEx	
7C809A99	5D	POP EBP	
7C809A9A	C2 1000	RETN 10	
7C809A9D	90	NOP	
7C809A9E	90	NOP	
7C809A9F	90	NOP	

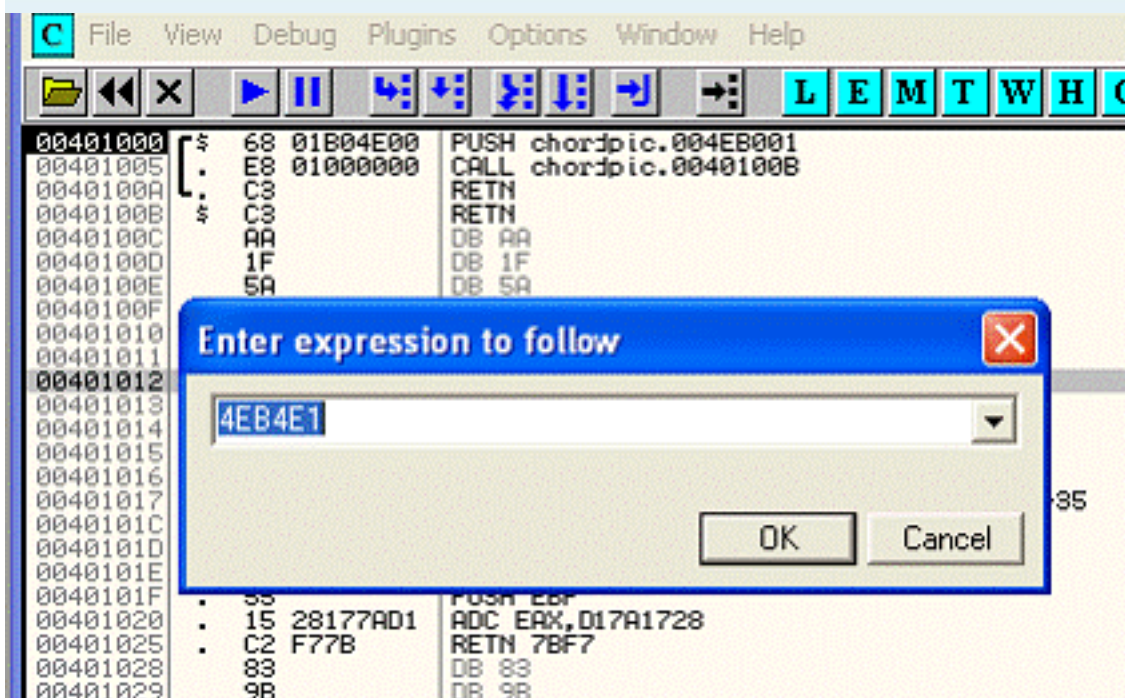
Press Shift + F9 and Olly will break in this API, press ALT + F9 to return a call this API function, and also see the address of memory allocated in the record EAX:

004EB4C1	43	INC EBX	
004EB4C2	BB 3219122A	MOV EBX, 2A121932	
004EB4C7	00B2 5C332589	ADD BYTE PTR DS:[EDX+8925335C], DH	
004EB4CD	A5	MOVS DWORD PTR ES:[EDI], DWORD PTR DS:[ESI]	
004EB4CE	290400	SUB DWORD PTR DS:[EAX+EAX], EAX	
004EB4D1	006A 40	ADD BYTE PTR DS:[EDX+40], CH	
004EB4D4	68 00100000	PUSH 1000	
004EB4D9	FFB5 00040000	PUSH DWORD PTR SS:[EBP+400]	
004EB4DF	6A 00	PUSH 0	
004EB4E1	FF95 F0030000	CALL NEAR DWORD PTR SS:[EBP+3F0]	Richiama VirtualAlloc per la prima volta
004EB4E7	8985 CC010000	MOV DWORD PTR SS:[EBP+1CC], EAX	
004EB4ED	8B9D 00040000	MOV EBX, DWORD PTR SS:[EBP+400]	
004EB4F3	039D 00040000	ADD EBX, DWORD PTR SS:[EBP+400]	
004EB4F9	50	PUSH EAX	
004EB4FA	53	PUSH EBX	

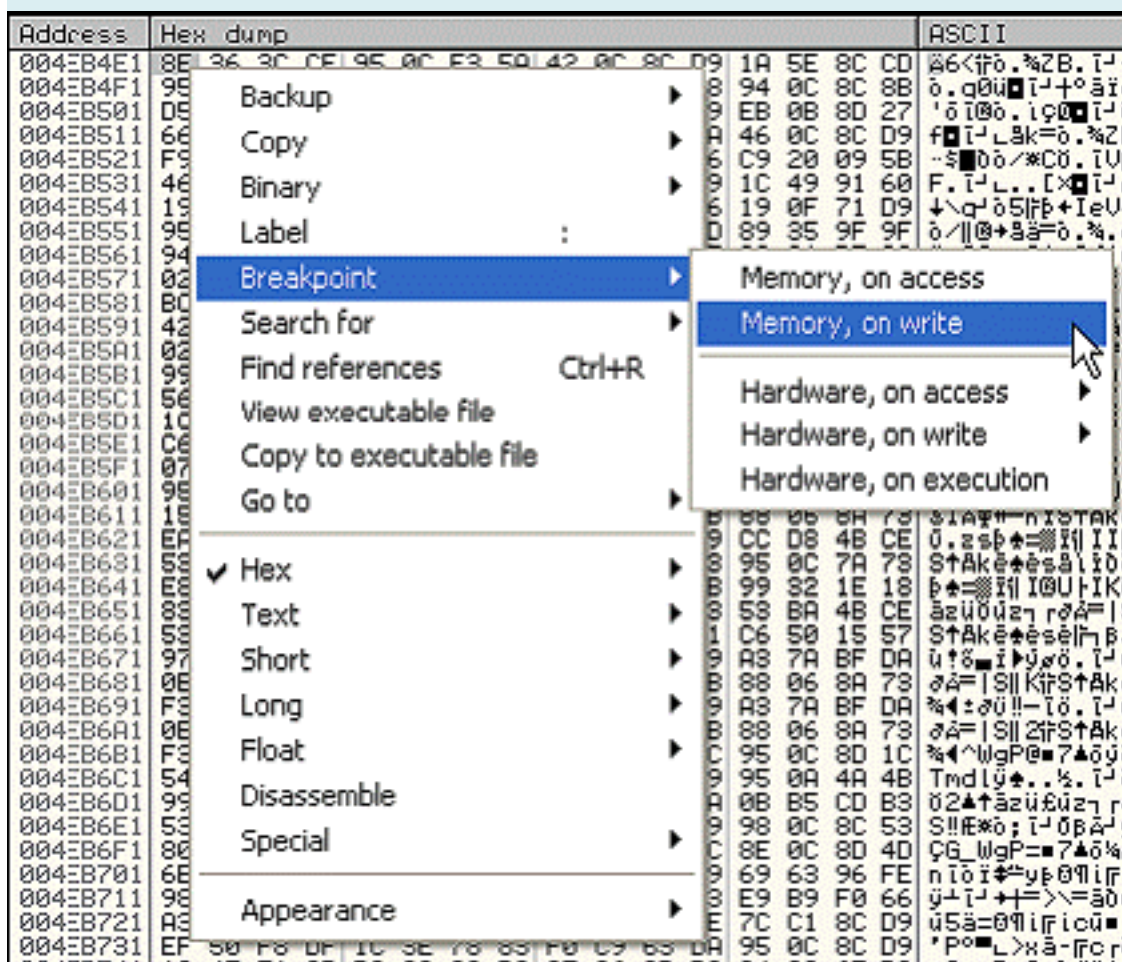
Note the value of EAX may be different in your computer. My 0x00B30000
chart in memory (memory map) of the (ALT + M) we see are as follows:

003F0000	00002000	chordpic		PE header	Map	R	E	R	E	
00400000	00001000	chordpic		code	Inag	R		RWE		
00401000	00099000	chordpic		data	Inag	R		RWE		
0049A000	00005000	chordpic			Inag	R		RWE		
0049F000	00002000	chordpic			Inag	R		RWE		
004A1000	00003000	chordpic			Inag	R		RWE		
004A4000	00001000	chordpic			Inag	R		RWE		
004A5000	00001000	chordpic			Inag	R		RWE		
004A6000	00001000	chordpic			Inag	R		RWE		
004A7000	0000A000	chordpic			Inag	R		RWE		
004B1000	00003000	chordpic	.rsrc	resources	Inag	R		RWE		
004E0000	00022000	chordpic	.data	imports, rel.	Inag	R		RWE		
00500000	00001000	chordpic	.adata		Inag	R		RWE		
00510000	00100000				Map	R		R		
00520000	00001000				Priv	RW		RW		
00530000	00129000				Map	R	E	R	E	
00930000	00001000				Priv	RW		RW		
00940000	00004000				Priv	RW		RW		
00950000	00003000				Map	R		R		
00960000	00003000				Priv	RW		RW		
00970000	00002000				Map	R		R		
00980000	00001000				Priv	RW		RW		
00990000	00004000				Priv	RW		RW		
00B30000	00033000				Priv	RWE		RWE		
011A0000	00002000				Map	R		R		
5A000000	00001000	swpg		PE header	Inag	R		RWE		
5A001000	00012000	swpg	CODE	code	Inag	R		RWE		
5A013000	00001000	swpg	DATA	data	Inag	R		RWE		
5A014000	00001000	swpg	BSS		Inag	R		RWE		
5A015000	00001000	swpg	.idata	imports	Inag	R		RWE		
5A016000	00001000	swpg	.reloc	relocations	Inag	R		RWE		
5A017000	00001000	swpg	.rsrc	resources	Inag	R		RWE		
5F000000	00001000				Priv	RWE		RWE		

We note this code and then contact the OEP decompression time is not present, which means that in mind this area is built with a class code that are not decompress then we must find no.Ta again to see if a function called VirtualAlloc first time this has been available in the program EntryPoint or not? Or is the class through a code and then another to turn a new call this function. Click CRL + F2, on the address at a function called on 0x004EB4E1



We have seen a difference, so Asprotect enforcement decrypt the code before the call VirtualAlloc function, then we must find where this code (ie the code with calls VirtualAlloc) recorded. We do the following: Olly restart the memory and setting breakpoints in a position 0x004EB4E1.



Press Shift + F9 and Olly 0x004EB13F at the end, next to is if a call is to decrypt

004EB13F	9F0413	POP DWORD PTR DS:[EBX+EDX]	
004EB142	E8 0F000000	CALL chordp10.004EB156	Decrittazione di tutto il codice a 0x004EB4E1
004EB147	1B8B 91F6F764	SBB EDI, DWORD PTR DS:[EAX+64F7F691]	
004EB14D	CD 82	INT 82	
004EB14F	93	XCHG EAX, EBX	

Continue to restart Olly and Entry Point to see what we have at 004EB142 (I do not know yet what is so keep), F7 We will see at once that we're in a cycle l p decompression,

004EB0F6	81C3 74080000	ADD EBX,874	
004EB0FC	BF F80F4C62	MOV EDI,624C0FF8	
004EB101	2BD2	SUB EDX,EDX	
004EB103	81EE C29C2260	SUB ESI,60223CC2	
004EB109	FF341A	PUSH DWORD PTR DS:[EDX+EBX]	LOOP START
004EB10C	59	POP ECX	
004EB10D	E8 0D000000	CALL chordpic.004EB11F	
004EB112	41	INC ECX	
004EB113	E6 27	OUT 27,AL	I/O command
004EB115	04 7D	AA 7D	
004EB117	72 C3	JB SHORT chordpic.004EB0DC	
004EB119	40	INC EAX	
004EB11A	79 BE	JNS SHORT chordpic.004EB0DA	
004EB11C	1F	POP DS	Modification of segment register
004EB11D	6C	INS BYTE PTR ES:[EDI],DX	I/O command
004EB11E	90	NOP	
004EB11F	66:8BF9	MOV DI,CX	
004EB122	5E	POP ESI	
004EB123	81E9 7C9A3773	SUB ECX,73379A7C	
004EB129	52	PUSH EDX	
004EB12A	8AE1	MOV AH,CL	
004EB12C	5E	POP ESI	
004EB12D	81E9 05BFA677	SUB ECX,77A6BF05	
004EB133	81E9 5A0B1143	SUB ECX,43110B5A	
004EB139	51	PUSH ECX	
004EB13A	BE FF2D774A	MOV ESI,4A772DFF	
004EB13F	8F0413	POP DWORD PTR DS:[EBX+EDX]	
004EB142	E8 0F000000	CALL chordpic.004EB156	
004EB147	1BB8 91F6F764	SBB EDI,DWORD PTR DS:[EAX+64F7F691]	
004EB14D	CD 82	INT 82	
004EB14F	93	XCHG EAX,EBX	
004EB150	00C9	ROR CL,1	
004EB152	CE	INT0	
004EB153	EF	OUT DX,EAX	I/O command
004EB154	FC	CLD	
004EB155	90	NOP	
004EB156	66:BF A6C8	MOV DI,0C8A6	
004EB15A	5E	POP ESI	
004EB15B	66:BE DF97	MOV SI,97DF	
004EB15F	83EA 03	SUB EDX,3	
004EB162	66:BE 719D	MOV SI,9D71	
004EB166	4A	DEC EDX	
004EB167	66:81DE E2E2	SBB SI,0E2E2	
004EB16C	81FA 78F8FFFF	CMP EDX,-788	
004EB172	0F85 3C000000	JNZ chordpic.004EB184	Non eseguito alla fine del loop
004EB178	E8 13000000	CALL chordpic.004EB190	Primo salto per fine loop
004EB17D	CF	IRETD	
004EB17E	5C	POP ESP	
004EB17F	65:3AEB	CMP CH,BL	Superfluous prefix
004EB182	48	DEC EAX	
004EB183	E1 06	LOOPDE SHORT chordpic.004EB18B	
004EB185	C7	???	Unknown command
004EB186	F4	HLT	Privileged command
004EB187	1D 92636019	SBB EAX,19606392	
004EB18C	90	NOP	
004EB18D	90	NOP	
004EB18E	90	NOP	
004EB18F	90	NOP	
004EB190	E8 0D000000	CALL chordpic.004EB1A2	
004EB195	51	PUSH ECX	
004EB196	86 B7	MOV DH,0B7	
004EB198	24 8D	AND AL,8D	
004EB19A	42	INC EDX	
004EB19B	53	PUSH EBX	
004EB19C	90	NOP	
004EB19D	898E AFBC4568	MOV DWORD PTR DS:[ESI+6845BCAF],ECX	
004EB1A3	A8 20	TEST AL,20	
004EB1A5	E1 35	LOOPDE SHORT chordpic.004EB1DC	
004EB1A7	5E	POP ESI	
004EB1A8	5E	POP ESI	
004EB1A9	5F	POP EDI	
004EB1AA	E9 1B000000	JMP chordpic.004EB1CA	USCITA DAL LOOP
004EB1AF	F9	STC	
004EB1B0	3E:9F	LAHF	Superfluous prefix
004EB1B2	EC	IN AL,DX	I/O command
004EB1B3	90	NOP	
004EB1B4	80D4 F0	ADC AH,0F0	
004EB1B7	E9 40FFFFFF	JMP chordpic.004EB109	LOOP END
004EB1BC	EE	OUT DX,AL	I/O command
004EB1BD	8F	???	Unknown command
004EB1BE	1C 25	SBB AL,25	
004EB1C0	FA	CLI	
004EB1C1	AB	STOS DWORD PTR ES:[EDI]	
004EB1C2	8001 6C074000	??	

Point end loop can be located at 0x004EB1AA. A Set breakpoints at this address and press Shift + F9, we see Olly will stop at that position and we look a bit position that this command will jump jump.

004EB0F6	81C3 74080000	ADD EBX,874	
004EB0FC	BF F8DF4C62	MOV EDI,624C0FF8	
004EB101	2B02	SUB EDX,EDX	
004EB103	81EE C23C2260	SUB ESI,60223CC2	
004EB109	FF341A	PUSH DWORD PTR DS:[EDX+EBX]	LOOP START
004EB10C	59	POP ECX	
004EB10D	E8 0D000000	CALL chordpic.004EB11F	
004EB112	41	INC ECX	
004EB113	E6 27	OUT 27,AL	I/O command
004EB115	D4 7D	ARM 7D	
004EB117	^ 72 C3	JB SHORT chordpic.004EB0DC	
004EB119	40	INC EAX	
004EB11A	^ 79 BE	JNS SHORT chordpic.004EB0DA	
004EB11C	1F	POP DS	Modification of segment register
004EB11D	6C	INS BYTE PTR ES:[EDI],DX	I/O command
004EB11E	90	NOP	
004EB11F	66:0BF9	MOV DI,CX	
004EB122	5E	POP ESI	
004EB123	81E9 7C9A3773	SUB ECX,73379A7C	
004EB129	52	PUSH EDX	
004EB12A	8AE1	MOV AH,CL	
004EB12C	5E	POP ESI	
004EB12D	81E9 058FA677	SUB ECX,77A68F05	
004EB133	81E9 5A0B1143	SUB ECX,43110B5A	
004EB139	51	PUSH ECX	
004EB13A	BE FF2D774A	MOV ESI,4A772DFF	
004EB13F	8F0413	POP DWORD PTR DS:[EBX+EDX]	
004EB142	E8 0F000000	CALL chordpic.004EB156	
004EB147	1B08 91F6F764	SBB EDI,DWORD PTR DS:[EAX+64F7F691]	
004EB14D	CD 82	INT 82	
004EB14F	93	XCHG EAX,EBX	
004EB150	D0C9	ROR CL,1	
004EB152	CE	INTO	
004EB153	EF	OUT DX,EAX	I/O command
004EB154	FC	CLO	
004EB155	90	NOP	
004EB156	66:BF A6C8	MOV DI,0C8A6	
004EB15A	5E	POP ESI	
004EB15B	66:BE DF97	MOV SI,97DF	
004EB15F	83EA 03	SUB EDX,3	
004EB162	66:BE 719D	MOV SI,9D71	
004EB166	4A	DEC EDX	
004EB167	66:81DE E2E2	SBB SI,0E2E2	
004EB16C	81FA 78F8FFFF	CMP EDX,-788	
004EB172	^ 0F85 3C000000	JNZ chordpic.004EB1B4	Non eseguito alla fine del loop
004EB178	E8 13000000	CALL chordpic.004EB190	Primo salto per fine loop
004EB17D	CF	IRETD	
004EB17E	5C	POP ESP	
004EB17F	65:3AEB	CMP CH,BL	Superfluous prefix
004EB182	48	DEC EAX	
004EB183	^ E1 06	LOOPDE SHORT chordpic.004EB18B	
004EB185	C7	???	Unknown command
004EB186	F4	HLT	Privileged command
004EB187	1D 92636019	SBB EAX,19606392	
004EB18C	90	NOP	
004EB18D	90	NOP	
004EB18E	90	NOP	
004EB18F	90	NOP	
004EB190	E8 0D000000	CALL chordpic.004EB1A2	
004EB195	51	PUSH ECX	
004EB196	B6 B7	MOV DH,0B7	
004EB198	24 8D	AND AL,8D	
004EB19A	42	INC EDX	
004EB19B	53	PUSH EBX	
004EB19C	90	NOP	
004EB19D	898E AFBC4568	MOV DWORD PTR DS:[ESI+6845BCAF],ECX	
004EB1A3	A8 20	TEST AL,20	
004EB1A5	^ E1 35	LOOPDE SHORT chordpic.004EB1DC	
004EB1A7	5E	POP ESI	
004EB1A8	5E	POP ESI	
004EB1A9	5F	POP EDI	
004EB1AA	^ E9 1B000000	JMP chordpic.004EB1CA	USCITA DAL LOOP
004EB1AF	F9	STC	
004EB1B0	3E:9F	LAHF	Superfluous prefix
004EB1B2	EC	IN AL,DX	I/O command
004EB1B3	90	NOP	
004EB1B4	80D4 F0	ADC AH,0F0	
004EB1B7	^ E9 4DFFFFFF	JMP chordpic.004EB109	LOOP END
004EB1BC	EE	OUT DX,AL	I/O command

004EB1B4	80D4 F0	ADC AH,0F0	
004EB1B7	E9 4DFFFFFF	JMP chordpic.004EB109	LOOP END
004EB1BC	EE	OUT DX,AL	I/O command
004EB1BD	8F	???	Unknown command
004EB1BE	1C 25	SBB AL,25	
004EB1C0	FA	CLI	
004EB1C1	AB	STOS DWORD PTR ES:[EDI]	
004EB1C2	08A1 C687B4DD	OR BYTE PTR DS:[ECX+00B487C6],AH	
004EB1C8	52	PUSH EDX	
004EB1C9	2366 81	AND ESP,DWORD PTR DS:[ESI-7F]	
004EB1CC	DFC5	FFRECP ST(5)	
004EB1CE	9C	PUSHFD	
004EB1CF	E8 0F000000	CALL chordpic.004EB1E3	

Giio now, Reatart Olly, then implement the position 0x004EB1AA:

004EB1AA	E9 1B000000	JMP chordpic.004EB1CA	USCITA DAL LOOP
004EB1AF	F9	STC	
004EB1B0	3E:9F	LAHF	Superfluous prefix
004EB1B2	EC	IN AL,DX	I/O command
004EB1B3	85 80	MOV CH,80	
004EB1B5	04 F0	RAM 0F0	
004EB1B7	E9 4DFFFFFF	JMP chordpic.004EB109	LOOP END
004EB1BC	EE	OUT DX,AL	I/O command
004EB1BD	8F	???	Unknown command
004EB1BE	1C 25	SBB AL,25	
004EB1C0	FA	CLI	
004EB1C1	AB	STOS DWORD PTR ES:[EDI]	
004EB1C2	08A1 C687B4DD	OR BYTE PTR DS:[ECX+00B487C6],AH	
004EB1C8	52	PUSH EDX	
004EB1C9	2341 E6	AND EAX,DWORD PTR DS:[ECX-1A]	
004EB1CC	CE	INTO	
004EB1CD	F3:	PREFIX REP:	Superfluous prefix
004EB1CE	77 4D	JA SHORT chordpic.004EB21D	
004EB1D0	FF2D 0B643A56	JMP FAR FWORD PTR DS:[563A64DB]	Far jump
004EB1D6	1C 4B	SBB AL,4B	
004EB1D8	17	POP SS	Modification of segment register
004EB1D9	0258 07	ADD BL,BYTE PTR DS:[EAX-29]	
004EB1DC	82 6E	MOV DL,6E	
004EB1DE	54	PUSH ESP	
004EB1DF	230F	AND ECX,DWORD PTR DS:[EDI]	
004EB1E1	9A 10B6D73A DB	CALL FAR 64DB:3AD7B610	Far call
004EB1E8	EF	OUT DX,EAX	I/O command
004EB1E9	31C8	XOR EAX,ECX	

We see the code more clear, this means that the order was not jumping to have the program at EntryPoint, and it also go to the last loop encoding first. This is a heiu very good for us because this is the cursor re-navigate to my area code patching us, then we **recorded this address**.

END OF THE Loop # 1 (redirection # 1): 0x004EB1AA

ORIGINAL instruction: JMP 0x004EB1CA

After implementation loop, we must also check if the code in the address of a VirtualAlloc API first recorded or not.

Then press Ctrl + G and 0x004EB4E1, this code is not decrypt, then we F7, through some of the directives we will face a loop with the new code decrypt disorder.

004EB1B7	E9 4DFFFFFF	JMP chordpic.004EB109	LOOP END
004EB1BC	EE	OUT DX,AL	I/O command
004EB1BD	8F	???	Unknown command
004EB1BE	1C 25	SBB AL,25	
004EB1C0	FA	CLI	
004EB1C1	AB	STOS DWORD PTR ES:[EDI]	
004EB1C2	08A1 C687B4DD	OR BYTE PTR DS:[ECX+00B487C6],AH	
004EB1C8	52	PUSH EDX	
004EB1C9	90	NOP	
004EB1CA	66:81DF C59C	SBB DI,9CC5	
004EB1CF	E8 0F000000	CALL chordpic.004EB1E3	
004EB1D4	4B	DEC EBX	
004EB1D5	2841 E6	SUB BYTE PTR DS:[ECX-1A],AL	
004EB1D8	27	DAA	
004EB1D9	D4 7D	RAM 7D	
004EB1DB	72 C3	JB SHORT chordpic.004EB1A0	
004EB1DD	40	INC EAX	
004EB1DE	70 BF	INC SHORT chordpic.004EB10F	

004EB1D8	72 C3	JB SHORT chordpic.004EB1A0	
004EB1DD	40	INC EAX	
004EB1DE	79 BE	JNS SHORT chordpic.004EB19E	
004EB1E0	1F	POP DS	Modification of segment register
004EB1E1	6C	INS BYTE PTR ES:[EDI],0x	I/O command
004EB1E2	90	NOP	
004EB1E3	51	PUSH ECX	
004EB1E4	E8 0C000000	CALL chordpic.004EB1F5	
004EB1E9	04 ED	ADD AL,0ED	
004EB1EB	22B3 70E96E0F	AND DH,BYTE PTR DS:[EBX+F6EE970]	
004EB1F1	9C	PUSHFD	
004EB1F2	A5	MOVS DWORD PTR ES:[EDI],DWORD PTR	
004EB1F3	7A 2B	JPE SHORT chordpic.004EB220	
004EB1F5	66:8BDE	MOV BX,SI	
004EB1F8	58	POP EAX	
004EB1F9	5B	POP EBX	
004EB1FA	5E	POP ESI	
004EB1FB	B0 7E	MOV AL,7E	
004EB1FD	81C6 7C070000	ADD ESI,77C	
004EB203	80F7 93	XOR BH,93	
004EB206	68 B1010000	PUSH 1B1	
004EB20B	81E3 8516025A	AND EBX,5A021685	
004EB211	59	POP ECX	
004EB212	9ADD	MOVL BL,CH	
004EB214	FF36	PUSH DWORD PTR DS:[ESI]	
004EB216	B0 5A	MOV AL,5A	
004EB218	5A	POP EDX	
004EB219	81C2 5B6F3451	ADD EDX,51346F5B	
004EB21F	0F83 05000000	JNB chordpic.004EB22A	
004EB225	53	PUSH EBX	
004EB226	80C4 30	ADD AH,30	
004EB229	5B	POP EBX	
004EB22A	81F2 F853A348	XOR EDX,48A353F8	
004EB230	81C2 D180F01A	ADD EDX,1AF080D1	
004EB236	E8 0D000000	CALL chordpic.004EB248	
004EB23B	E1 06	LOOPDE SHORT chordpic.004EB243	
004EB23D	C7	???	Unknown command
004EB23E	F4	HLT	Privileged command
004EB23F	1D 92636019	SBB EAX,19606392	
004EB244	90	NOP	
004EB245	90	NOP	
004EB246	90	NOP	
004EB247	90	NOP	
004EB248	8AD8	MOV BL,AL	
004EB24A	5B	POP EBX	
004EB24B	8916	MOV DWORD PTR DS:[ESI],EDX	
004EB24D	83EE 03	SUB ESI,3	
004EB250	BF 661F5A5E	MOV EDI,5E5A1F66	
004EB255	4E	DEC ESI	
004EB256	66:81CB 43C2	OR BX,0C243	
004EB25B	49	DEC ECX	
004EB25C	0F85 19000000	JNZ chordpic.004EB27B	
004EB262	B8 B5D4A229	MOV EAX,29A2D4B5	
004EB267	E9 24000000	JMP chordpic.004EB290	USCITA DAL LOOP#2
004EB26C	BB D8311697	MOV EBX,971631D8	
004EB271	846D A2	TEST BYTE PTR SS:[EBP-5E],CH	
004EB274	33F0	XOR ESI,EAX	
004EB276	90	NOP	
004EB277	90	NOP	
004EB278	90	NOP	
004EB279	90	NOP	
004EB27A	90	NOP	
004EB27B	0F87 02000000	JA chordpic.004EB283	
004EB281	8BC2	MOV EAX,EDX	
004EB283	E9 8CFFFFFF	JMP chordpic.004EB214	
004EB288	20D9	AND CL,BL	
004EB28A	9E	SAHF	
004EB28B	7F 4C	JG SHORT chordpic.004EB2D9	
004EB28D	95	XCHG EAX,EBP	
004EB28E	AA	STOS BYTE PTR ES:[EDI]	
004EB28F	9B	WAIT	
004EB290	BA E5429C47	MOV EDX,479C42E5	
004EB295	C585 5C7CBD1A	LDS EAX,FWORD PTR SS:[EBP+1ABD7C5C]	Modification of segment register

In 004EB267 position is the position loop end, continue to set a breakpoints on this address (F2) and press Shift + F9 to implement all the ttat repeats quickly.

004EB238	✓ E1 0E	LOOPDE SHORT chordpic.004EB243	
004EB23D	C7	???	Unknown command
004EB23E	F4	HLT	Privileged command
004EB23F	1D 92636019	SBB EAX,19606392	
004EB244	90	NOP	
004EB245	90	NOP	
004EB246	90	NOP	
004EB247	90	NOP	
004EB249	9AD9	MOV BL,AL	
004EB24A	5B	POP EBX	
004EB24B	8916	MOV DWORD PTR DS:[ESI],EDX	
004EB24D	83EE 03	SUB ESI,3	
004EB250	BF 661F5A5E	MOV EDI,5E5A1F66	
004EB255	4E	DEC ESI	
004EB256	66:81CB 43C2	OR BX,0C243	
004EB25B	49	DEC ECX	
004EB25C	✓ 0F95 19000000	JNZ chordpic.004EB27B	
004EB262	BB B04A229	MOV EAX,29A2D4B5	
004EB267	✓ E9 24000000	JMP chordpic.004EB290	USCITA DAL LOOP#2
004EB26C	BB D8311697	MOV EBX,971631D8	
004EB271	846D A2	TEST BYTE PTR SS:[EBP-5E],CH	
004EB274	33F0	XOR ESI,EAX	
004EB276	90	NOP	
004EB277	90	NOP	
004EB278	90	NOP	
004EB279	90	NOP	
004EB27A	90	NOP	
004EB27B	✓ 0F87 02000000	JA chordpic.004EB283	
004EB281	8BC2	MOV EAX,EDX	
004EB283	^ E9 8CFFFFFF	JMP chordpic.004EB214	
004EB288	20D9	AND CL,BL	
004EB28A	9E	SAHF	
004EB28B	✓ 7F 4C	JG SHORT chordpic.004EB2D9	
004EB28D	95	XCHG EAX,EBP	
004EB28E	AA	STOS BYTE PTR ES:[EDI]	
004EB28F	9B	WAIT	
004EB290	→ BE 87C4C02B	MOV ESI,2BC0C487	
004EB295	E8 05000000	CALL chordpic.004EB2A3	
004EB299	DD52 23	FST QWORD PTR DS:[EDX+23]	
004EB29D	20D9	AND CL,BL	
004EB29F	9E	SAHF	
004EB2A0	✓ 7F 4C	JG SHORT chordpic.004EB2EE	
004EB2A2	95	XCHG EAX,EBP	
004EB2A3	81E2 77DDF0A	AND EDX,0AFDD077	
004EB2A9	5B	POP EBX	
004EB2AA	68 0221407E	PUSH 7E402102	
004EB2AF	✓ E9 0E000000	JMP chordpic.004EB2C2	
004EB2B4	4E	DEC ESI	
004EB2B5	6F	OUTS DX,DWORD PTR ES:[EDI]	I/O command
004EB2B6	7C 0E	JG SHORT chordpic.004EB2B0	

So we also note recorded the address, where the point to our code.

END OF THE Loop # 2 (redirection # 2): 0x004EB267

Instruction ORIGINATES added: JMP 0x004EB290

Then we also do first time, see the API considered VirtualAlloc Caller appear or not, and did not see it appear, so we continue with the fire again:

We have a loop other decryption:

004EB267	✓ E9 24000000	JMP chordpic.004EB290	USCITA DAL LOOP#2
004EB26C	BB D8311697	MOV EBX,971631D8	
004EB271	846D A2	TEST BYTE PTR SS:[EBP-5E],CH	
004EB274	33F0	XOR ESI,EAX	
004EB276	69EE 8F1C250F	IMUL EBP,ESI,0F251C8F	
004EB27C	8702	XCHG DWORD PTR DS:[EDX],EAX	
004EB27E	0000	ADD BYTE PTR DS:[EAX],AL	
004EB280	008B C2E98CFF	ADD BYTE PTR DS:[EBX+FF8CE9C2],CL	
004EB286	FFFF	???	Unknown command
004EB288	20D9	AND CL,BL	
004EB28A	9E	SAHF	
004EB28B	✓ 7F 4C	JG SHORT chordpic.004EB2D9	
004EB28D	95	XCHG EAX,EBP	
004EB28E	AA	STOS BYTE PTR ES:[EDI]	
004EB28F	9B	WAIT	
004EB290	BE 87C4C023	MOV ESI,2BC0C487	
004EB295	E8 09000000	CALL chordpic.004EB2A3	
004EB29A	DD52 23	FST QWORD PTR DS:[EDX+23]	
004EB29D	20D9	AND CL,BL	
004EB29F	9E	SAHF	
004EB2A0	✓ 7F 4C	JG SHORT chordpic.004EB2EE	
004EB2A2	95	XCHG EAX,EBP	
004EB2A3	81E2 77DDF0A	AND EDX,0AFDD077	
004EB2A9	5B	POP EBX	
004EB2AA	68 0221407E	PUSH 7E402102	
004EB2AF	✓ E9 0E000000	JMP chordpic.004EB2C2	
004EB2B4	4E	DEC ESI	
004EB2B5	6F	OUTS DX,DWORD PTR ES:[EDI]	I/O command
004EB2B6	7C 0E	JG SHORT chordpic.004EB2B0	

004EB2B4	4E	DEC ESI	
004EB2B5	6F	OUTS DX,DWORD PTR ES:[EDI]	I/O command
004EB2B6	7C 05	JL SHORT chordpic.004EB2BD	
004EB2B8	5A	POP EDX	
004EB2B9	8B68 81	MOV EBP,DWORD PTR DS:[EAX-7F]	
004EB2BC	26:67:14 B3	ADC AL,0BD	Superfluous prefix
004EB2C0	B2 03	MOV DL,3	
004EB2C2	58	POP EAX	
004EB2C3	81C3 B5060300	ADD EBX,6B5	
004EB2C9	2BFF	SUB EDI,EDI	
004EB2CB	B8 F170BA61	MOV EAX,61BA70F1	
004EB2D0	8B0C3B	MOV ECX,DWORD PTR DS:[EBX+EDI]	INIZIO DEL LOOP#3
004EB2D3	66:81DE AE38	SBB SI,88AE	
004EB2D8	81F1 F09DD73D	XOR ECX,3DD79DF0	
004EB2DE	81DA 6B18C36F	SBB EDX,6FCC186B	
004EB2E4	81C1 698F831E	ADD ECX,1E8D8F69	
004EB2EA	81E8 128EB370	SUB EAX,70BB8E12	
004EB2F0	81E9 EE3D2E38	SUB ECX,382E3DEE	
004EB2F6	81F6 0C6A4C13	XOR ESI,134C6A0C	
004EB2FC	51	PUSH ECX	
004EB2FD	0FB7F3	MOVZX ESI,BX	
004EB300	8F043B	POP DWORD PTR DS:[EBX+EDI]	
004EB303	66:81E6 1051	AND SI,5110	
004EB308	66:81F2 4B28	XOR DX,284B	
004EB30D	81EF 41BB3515	SUB EDI,1535BB41	
004EB313	0FB7D2	MOVZX EDX,DX	
004EB316	81C7 3DBB3515	ADD EDI,1535BB3D	
004EB31C	0FB7F3	MOVZX ESI,BX	
004EB31F	81FF 20FAF7FF	CMP EDI,-5E0	
004EB325	0F85 19000300	JNZ chordpic.004EB344	
004EB32B	8BC6	MOV EAX,ESI	
004EB32D	E9 41000003	JMP chordpic.004EB373	USCITA DAL LOOP#3
004EB332	22B3 70E96E0F	AND DH,BYTE PTR DS:[EBX+F6EE970]	
004EB338	9C	PUSHFD	
004EB339	A5	MOVS DWORD PTR ES:[EDI],DWORD PTR	
004EB33A	7A 2B	JPE SHORT chordpic.004EB367	
004EB33C	8821	MOV BYTE PTR DS:[ECX],AH	
004EB33E	46	INC ESI	
004EB33F	07	POP ES	Modification of segment register
004EB340	34 5D	XOR AL,5D	
004EB342	90	NOP	
004EB343	90	NOP	
004EB344	0F84 14000300	JE chordpic.004EB35E	
004EB34A	53	PUSH EBX	
004EB34B	E9 0D000003	JMP chordpic.004EB35D	
004EB350	91	XCHG EAX,ECX	
004EB351	F6F7	DIV BH	
004EB353	64:CD 82	INT 82	Superfluous prefix
004EB356	93	XCHG EAX,EBX	
004EB357	D0C9	ROR CL,1	
004EB359	CE	INTO	
004EB35A	EF	OUT DX,EAX	I/O command
004EB35B	FC	CLD	
004EB35C	90	NOP	
004EB35D	58	POP EAX	chordpic.004EB94F
004EB35E	E9 6DFFFFFF	JMP chordpic.004EB2D0	
004EB363	E8 01A6E794	CALL 95365969	
004EB368	3D 32830039	CMP EAX,39008332	
004EB36D	7E DF	JLE SHORT chordpic.004EB34E	
004EB36F	2C F5	SUB AL,0F5	
004EB371	8AFB	MOV BH,BL	
004EB373	9E	SAHF	
004EB374	25 77247595	AND EAX,96752477	
004EB379	A4	MOVS BYTE PTR ES:[EDI],BYTE PTR DS	

Applied as before, a set breakpoints on addresses exit loop and press Shift + F9 to address deen just set breakpoints.

004EB30D	81EF 41BB3515	SUB EDI,15358B41	
004EB313	0FB7D2	MOVZX EDX,DX	
004EB316	81C7 30BB3515	ADD EDI,15358B3D	
004EB31C	0FB7F3	MOVZX ESI,BX	
004EB31F	81FF 20FAFFFF	CMP EDI,-5E0	
004EB325	0F85 19000000	JNZ chordpic.004EB344	
004EB32B	8BC6	MOV EAX,ESI	
004EB32D	E9 41000000	JMP chordpic.004EB373	USCITA DAL _LOOP#3
004EB332	22B3 70E96E0F	AND DH,BYTE PTR DS:[EBX+F6EE970]	
004EB338	9C	PUSHFD	
004EB339	A5	MOVS DWORD PTR ES:[EDI],DWORD PTR	
004EB33H	7A 2B	JPE SHORT chordpic.004EB367	
004EB33C	8821	MOV BYTE PTR DS:[ECX],AH	
004EB33E	46	INC ESI	
004EB33F	07	POP ES	Modification of segment register
004EB340	34 5D	XOR AL,5D	
004EB342	90	NOP	
004EB343	90	NOP	
004EB344	0F84 14000000	JE chordpic.004EB35E	
004EB34A	53	PUSH EBX	
004EB34B	E9 0D000000	JMP chordpic.004EB35D	
004EB350	91	XCHG EAX,ECX	
004EB351	F6F7	DIV BH	
004EB353	64:CD 82	INT 82	Superfluous prefix
004EB356	93	XCHG EAX,EBX	
004EB357	D0C9	ROR CL,1	
004EB359	CE	INTO	
004EB35A	EF	OUT DX,EAX	I/O command
004EB35B	FC	CLD	
004EB35C	90	NOP	
004EB35D	58	POP EAX	
004EB35E	E9 6DFFFFFF	JMP chordpic.004EB2D0	
004EB363	E8 01A6E794	CALL 95365969	
004EB368	3D 32030039	CMP EAX,39000032	
004EB36D	7E DF	JLE SHORT chordpic.004EB34E	
004EB36F	2C F5	SUB AL,0F5	
004EB371	8AFB	MOV BH,BL	
004EB373	E9 09000000	JMP chordpic.004EB381	
004EB378	5D	POP EBP	
004EB379	D2A3 A0591EFF	SHL BYTE PTR DS:[EBX+FF1E59A0],CL	
004EB37F	CC	INT3	
004EB380	15 E80F0000	ADC EAX,0FE8	

We note also recorded the address, where the point to our code.

END OF THE Loop # 3 (redirection # 3): 0x004EB32D

ORIGINAL instruction: JMP 0x004EB373

Continue to look at the view has not found 004EB4E1 appear. we continue to test fire.

We have a loop more

004EB32D	E9 41000000	JMP chordpic.004EB373	USCITA DAL LOOP#3
004EB332	22B3 70E96E0F	AND DH,BYTE PTR DS:[EBX+F6EE970]	
004EB338	9C	PUSHFD	
004EB339	A5	MOVS DWORD PTR ES:[EDI],DWORD PTR	
004EB33A	7A 2B	JPE SHORT chordpic.004EB367	
004EB33C	8821	MOV BYTE PTR DS:[ECX],AH	
004EB33E	46	INC ESI	
004EB33F	07	POP ES	Modification of segment register
004EB340	34 5D	XOR AL,5D	
004EB342	D2A3 0F841400	SHL BYTE PTR DS:[EBX+14840F],CL	
004EB348	0000	ADD BYTE PTR DS:[EAX],AL	
004EB34A	53	PUSH EBX	
004EB34B	E9 0D000000	JMP chordpic.004EB35D	
004EB350	91	XCHG EAX,ECX	
004EB351	F6F7	DIV BH	
004EB353	64:CD 82	INT 82	Superfluous prefix
004EB356	93	XCHG EAX,EBX	
004EB357	D0C9	ROR CL,1	
004EB359	CE	INTO	
004EB35A	EF	OUT DX,EAX	I/O command
004EB35B	FC	CLD	
004EB35C	8558 E9	TEST DWORD PTR DS:[EAX-17],EBX	
004EB35F	6D	INS DWORD PTR ES:[EDI],DX	I/O command
004EB360	FFFF	???	Unknown command
004EB362	FFE8	JMP FAR EAX	Illegal use of register
004EB364	01A6 E7943D32	ADD DWORD PTR DS:[ESI+323D94E7],ES	
004EB36D	0300 20	ADD DWORD PTR DS:[ECX],20	

004EB362	FFE8	JMP FAR EAX	Unknown command
004EB364	01A6 E7943D32	ADD DWORD PTR DS:[ESI+323D94E7],ESI	Illegal use of register
004EB36A	8300 39	ADD DWORD PTR DS:[EAX],39	
004EB36D	^ 7E DF	JLE SHORT chordpic.004EB34E	
004EB36F	2C F5	SUB AL,0F5	
004EB371	8AFB	MOV BH,BL	
004EB373	^ E9 09000000	JMP chordpic.004EB381	
004EB378	5D	POP EBP	
004EB379	D2A3 A0591EFF	SHL BYTE PTR DS:[EBX+FF1E59A0],CL	
004EB37F	CC	INT3	
004EB380	90	NOP	
004EB381	E8 0F000000	CALL chordpic.004EB395	
004EB386	1BB8 91F6F764	SBB EDI,DWORD PTR DS:[EAX+64F7F691]	
004EB38C	CD 82	INT 82	
004EB38E	93	XCHG EAX,EBX	
004EB38F	D0C9	ROR CL,1	
004EB391	CE	INTO	
004EB392	EF	OUT DX,EAX	I/O command
004EB393	FC	CLD	
004EB394	90	NOP	
004EB395	^ 0FAF 03000000	JG chordpic.004EB39F	
004EB39B	0FBFF8	MOVSX EDI,AX	
004EB39E	59	POP ECX	
004EB39F	66:BF 7E9A	MOV DI,9A7E	
004EB3A3	81C1 C9050000	ADD ECX,5C9	
004EB3A9	0F8A 02000000	JPE chordpic.004EB3B1	
004EB3AF	8AF1	MOV DH,CL	
004EB3B1	68 4A010000	PUSH 14A	
004EB3B6	58	POP EAX	
004EB3B7	BE 657B2843	MOV ESI,43287B65	
004EB3BC	8B19	MOV EBX,DWORD PTR DS:[ECX]	INIZIO DEL LOOP#4
004EB3BE	8BD7	MOV EDX,EDI	
004EB3C0	81F3 0F27B513	XOR EBX,13B5270F	
004EB3C6	66:BA 6062	MOV DX,6260	
004EB3CA	81F3 9C63A21A	XOR EBX,1AA2639C	
004EB3D0	66:8BFA	MOV DI,DX	
004EB3D3	81EB A5212678	SUB EBX,782621A5	
004EB3D9	66:81DE B65C	SBB SI,5CB6	
004EB3DE	53	PUSH EBX	
004EB3DF	66:81F7 8934	XOR DI,3489	
004EB3E4	8F01	POP DWORD PTR DS:[ECX]	
004EB3E6	81EE 9AA34F6F	SUB ESI,6F4FA39A	
004EB3EC	81E9 A8D4641F	SUB ECX,1F64D4A8	
004EB3F2	BF 5493861D	MOV EDI,1D869354	
004EB3F7	81C1 A4D4641F	ADD ECX,1F64D4A4	
004EB3FD	68 43026048	PUSH 4B600243	
004EB402	81C7 BB764D47	ADD EDI,474D76BB	
004EB408	5E	POP ESI	
004EB409	48	DEC EAX	
004EB40A	^ 0F85 ACFFFFFF	JNZ chordpic.004EB3BC	
004EB410	E8 12000000	CALL chordpic.004EB427	USCITA DAL LOOP#4
004EB415	A2 33F069EE	MOV BYTE PTR DS:[EE69F033],AL	
004EB41A	8F	???	Unknown command
004EB41B	1C 25	SBB AL,25	
004EB41D	FA	CLI	
004EB41E	AB	STOS DWORD PTR ES:[EDI]	
004EB41F	08A1 C687B4DD	OR BYTE PTR DS:[ECX+DDB487C6],AH	
004EB425	52	PUSH EDX	
004EB426	2366 8B	AND ESP,DWORD PTR DS:[ESI-75]	
004EB429	F1	INT1	
004EB42A	5E	POP ESI	
004EB42B	E8 00000000	CALL chordpic.004EB430	
004EB430	5D	POP EBP	
004EB431	5B	POP EBX	
004EB432	895D 5B	MOV DWORD PTR SS:[EBP+5B],EBX	
004EB435	5B	POP EBX	

How like the F2 at the end loop, the Shift + F9. continue to note:

END OF THE Loop # 4 (redirection # 4): 0x004EB410

ORIGINAL Instruction: 0x004EB427

Finally, it also appears to 004EB4E1 a VirtualAlloc API function calls.

Known, here we laam steps:

1. We are in the position of the packer Entrypoint.
2. Loop decryption 1 implementation, the cursor exit loop in 004EB1AA
3. Loop decryption 2 implementation, the cursor exit loop in 004EB267
4. Loop decryption 3 implementation, the cursor exit loop in 004EB32D
5. Loop decryption 4th implemented, the cursor exit loop in 004EB410
6. We see the appearance cuatoi VirtualAlloc API function calls appear 0x004EB4E1

List breakpoints:

Address	Module	Active	Disassembly	Comment
004EB1AA	chordpic	Always	JMP chordpic.004EB1CA	USCITA DAL LOOP#1
004EB267	chordpic	Always	JMP chordpic.004EB290	USCITA DAL LOOP#2
004EB32D	chordpic	Always	JMP chordpic.004EB373	USCITA DAL LOOP#3
004EB410	chordpic	Always	CALL chordpic.004EB427	USCITA DAL LOOP#4

Then, we will trace the meeting who called VirtualAlloc API function

004EB409	FFB5 00040000	PUSH DWORD PTR SS:[EBP+400]	
004EB40F	6A 00	PUSH 0	
004EB4E1	FF95 F0030000	CALL NEAR DWORD PTR SS:[EBP+3F0]	Richiama VirtualAlloc per la prima volta
004EB4E7	0905 CC010000	MOV DWORD PTR SS:[EBP+1CC],EAX	
004EB4ED	009D 00040000	MOV EBX,DWORD PTR SS:[EBP+400]	
004EB4F3	039D 00040000	ADD EBX,DWORD PTR SS:[EBP+400]	
004EB4F9	50	PUSH EAX	
004EB4FA	53	PUSH EBX	
004EB4FB	E8 04010000	CALL chordpic.004EB504	
004EB500	6A 40	PUSH 40	
004EB502	68 00100000	PUSH 1000	
004EB507	FFB5 00040000	PUSH DWORD PTR SS:[EBP+400]	
004EB50D	6A 00	PUSH 0	
004EB50F	FF95 F0030000	CALL NEAR DWORD PTR SS:[EBP+3F0]	Richiama Virtual Alloc per la seconda volta
004EB515	0905 31040000	MOV DWORD PTR SS:[EBP+431],EAX	EAX = indirizzo sezione a run-time che scrive OEP
004EB51B	0905 D0010000	MOV DWORD PTR SS:[EBP+1D0],EAX	
004EB521	64:67:A1 0000	MOV EAX,DWORD PTR FS:[0]	
004EB526	0905 20040000	MOV DWORD PTR SS:[EBP+42D],EAX	
004EB52C	0055 50	MOV EDX,DWORD PTR SS:[EBP+50]	
004EB52F	0085 D0010000	MOV EAX,DWORD PTR SS:[EBP+1D0]	
004EB535	0902	MOV DWORD PTR DS:[EDX],EAX	
004EB537	0085 00040000	MOV EAX,DWORD PTR SS:[EBP+400]	
004EB53D	0942 04	MOV DWORD PTR DS:[EDX+4],EAX	
004EB540	0085 9F030000	LEA EAX,DWORD PTR SS:[EBP+39F]	
004EB546	0040 55	MOV EAX,DWORD PTR DS:[EAX+55]	
004EB549	0942 08	MOV DWORD PTR DS:[EDX+8],EAX	
004EB54C	0085 EC030000	MOV EAX,DWORD PTR SS:[EBP+3EC]	
004EB552	0942 10	MOV DWORD PTR DS:[EDX+10],EAX	
004EB555	0085 E0030000	MOV EAX,DWORD PTR SS:[EBP+3E0]	
004EB55B	0942 14	MOV DWORD PTR DS:[EDX+14],EAX	
004EB55E	0095 CC010000	MOV EDX,DWORD PTR SS:[EBP+1CC]	
004EB564	00 F0010000	MOV EBX,1F0	
004EB569	007C1A 0C	MOV EDI,DWORD PTR DS:[EDX+EBX+C]	
004EB56D	00FF	OR EDI,EDI	
004EB56F	74 1E	JE SHORT chordpic.004EB58F	
004EB571	004C1A 10	MOV ECX,DWORD PTR DS:[EDX+EBX+10]	
004EB575	00C9	OR ECX,ECX	
004EB577	74 11	JE SHORT chordpic.004EB58A	
004EB579	03BD D0010000	ADD EDI,DWORD PTR SS:[EBP+1D0]	
004EB57F	00741A 14	MOV ESI,DWORD PTR DS:[EDX+EBX+14]	
004EB583	03F2	ADD ESI,EDX	
004EB585	C1F9 02	SAR ECX,2	
004EB588	F3:A5	REP MOVSDWORD PTR ES:[EDI],DWORD PTR DS:[ESI]	
004EB58A	03C3 20	ADD EBX,20	
004EB58D	EB 0A	JMP SHORT chordpic.004EB569	
004EB591	0085 CC010000	MOV EAX,DWORD PTR SS:[EBP+1CC]	Fine loop scrittura area a run-time
004EB595	50	PUSH EAX	
004EB596	0095 D0010000	MOV EDX,DWORD PTR SS:[EBP+1D0]	
004EB59C	52	PUSH EDX	
004EB59D	0018	MOV EBX,DWORD PTR DS:[EAX]	
004EB59F	03DA	ADD EBX,EDX	
004EB5A1	0085 E4030000	MOV EAX,DWORD PTR SS:[EBP+3E4]	
004EB5A7	0903	MOV DWORD PTR DS:[EBX],EAX	
004EB5A9	0085 E0030000	MOV EAX,DWORD PTR SS:[EBP+3E0]	
004EB5AF	0943 04	MOV DWORD PTR DS:[EBX+4],EAX	
004EB5B2	0085 EC030000	MOV EAX,DWORD PTR SS:[EBP+3EC]	
004EB5B8	0943 08	MOV DWORD PTR DS:[EBX+8],EAX	
004EB5BB	5F	POP EDI	
004EB5BC	5E	POP ESI	
004EB5BD	0046 04	MOV EAX,DWORD PTR DS:[ESI+4]	
004EB5C0	03C7	ADD EAX,EDI	
004EB5C2	0905 C7010000	MOV DWORD PTR SS:[EBP+1C7],EAX	
004EB5C8	0055 50	MOV EDX,DWORD PTR SS:[EBP+50]	
004EB5CB	0085 C7010000	MOV EAX,DWORD PTR SS:[EBP+1C7]	
004EB5D1	0942 0C	MOV DWORD PTR DS:[EDX+C],EAX	
004EB5D4	009D 00040000	LEA EBX,DWORD PTR SS:[EBP+400]	
004EB5DA	53	PUSH EBX	
004EB5DB	6A 00	PUSH 0	


```

004EB501  0742 0C      MOV DWORD PTR DS:[EDX*2],EAX
004EB504  8090 00040000 LEA EBX,00040000 PTR SS:[EBP+400]
004EB50A  53          PUSH EBX
004EB50B  6A 00       PUSH 0
004EB50D  6A 00       PUSH 0
004EB50F  6A 01       PUSH 1
004EB5E1  57          PUSH EDI
004EB5E2  8B5E 08     MOV EBX,0008 PTR DS:[ESI+8]
004EB5E6  83DF       ADD EBX,EDI
004EB5E7  53          PUSH EBX
004EB5E9  68 00000000 PUSH 0000
004EB5ED  6A 00       PUSH 0
004EB5EF  56          PUSH ESI
004EB5F0  FF95 F4030000 CALL NEAR DWORD PTR SS:[EBP+3F4]
004EB5F6  68 00100A00 PUSH 00A1000
004EB5FB  C3         RETN
004EB5FC  0000       ADD BYTE PTR DS:[EAX],AL
004EB5FE  B3 00      MOV BL,0
004EB600  0000       ADD BYTE PTR DS:[EAX],AL

```

Salta nell'area allocata a run-time

Order Push 0BA1000 first order RETN, this will navigate the jump to new locations are identified. But there are also interesting points when you look at the command PUSH 8000 because it can recover in EDI address background of new memory areas are located.

You can implement the financial navigate to patching code in our use of the command BYTE PUSH 8000

ABSOLUTE ADDRESS (redirection # 5): 0x004EB5E8

ORIGINAL Instruction: PUSH 8000 -> 0x68 0x00 0x80 0x00 0x00

Now from here I will talk about its offset address 00BA1000.

We begin in code Decompression for OEP.

00BA1000	90	NOP	Entry point
00BA1001	60	PUSHAD	
00BA1002	E8 40060000	CALL 00BA1647	
00BA1007	EB 44	JMP SHORT 00BA104D	
00BA1009	0000	ADD BYTE PTR DS:[EAX],AL	
00BA100B	0000	ADD BYTE PTR DS:[EAX],AL	
00BA100D	0000	ADD BYTE PTR DS:[EAX],AL	
00BA100F	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1011	87DB	XCHG EBX,EBX	
00BA1013	90	NOP	
00BA1014	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1016	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1018	0000	ADD BYTE PTR DS:[EAX],AL	
00BA101A	0000	ADD BYTE PTR DS:[EAX],AL	
00BA101C	0000	ADD BYTE PTR DS:[EAX],AL	
00BA101E	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1020	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1022	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1024	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1026	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1028	0000	ADD BYTE PTR DS:[EAX],AL	
00BA102A	0000	ADD BYTE PTR DS:[EAX],AL	
00BA102C	0000	ADD BYTE PTR DS:[EAX],AL	
00BA102E	1003	ADC BYTE PTR DS:[EBX],AL	
00BA1030	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1032	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1034	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1036	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1038	0000	ADD BYTE PTR DS:[EAX],AL	
00BA103A	0000	ADD BYTE PTR DS:[EAX],AL	
00BA103C	0000	ADD BYTE PTR DS:[EAX],AL	
00BA103E	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1040	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1042	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1044	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1046	0000	ADD BYTE PTR DS:[EAX],AL	
00BA1048	0000	ADD BYTE PTR DS:[EAX],AL	
00BA104A	0000	ADD BYTE PTR DS:[EAX],AL	
00BA104C	90	NOP	
00BA104D	BB 44294400	MOV EBX,442944	
00BA1052	03D0	ADD EBX,EBP	
00BA1054	2B9D 71294400	SUB EBX,DWORD PTR SS:[EBP+442971]	
00BA105A	83BD 08304400	CMP DWORD PTR SS:[EBP+4430D8],0	
00BA1061	899D 2F2E4400	MOV DWORD PTR SS:[EBP+442E2F],EBX	
00BA1067	0F85 3E050000	JNZ 00BA15AB	
00BA106D	8D85 E0304400	LEA EAX,DWORD PTR SS:[EBP+4430E0]	

Looking at the code offset 0x002663, we see the code here is quite different than the code we see the beginning of the description of us, because we must also step by step through several layers before decompression when reaching code unpack all code for the program.

00B72663	0109	ADD ECX,EBX	
00B72665	3D B4944C36	CMP EAX,364C94B4	
00B7266A	48	DEC EAX	
00B7266B	98	CWDE	
00B7266C	2C 44	SUB AL,44	
00B7266E	CB	RET	Far return
00B7266F	E8 3403230A	CALL 00DA29A8	
00B72674	C545 A3	LDS EAX,FWORD PTR SS:[EBP-5D]	Modification of segment register
00B72677	44	INC ESP	
00B72678	74 24	JE 00B7269E	
00B7267A	188D 533CE246	SBB BYTE PTR SS:[EBP+46B23C53],CL	
00B72680	06	PUSH ES	

We continue to describe the code in chi0x00BA1007, we have a function called on VirtualAlloc 0x00BA10C4, then we will have a loop to decrypt noted that other code.

00BA10D6	50	PUSH EAX	
00BA10D7	53	PUSH EBX	
00BA10D8	E8 74050000	CALL 00BA1651	
00BA10DD	80C8	MOV ECX,EAX	
00BA10DF	00BD 452A4400	LEA EDI,DWORD PTR SS:[EBP+442A45]	
00BA10E5	00B5 75294400	MOV ESI,DWORD PTR SS:[EBP+442975]	
00BA10EB	F3:A4	REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	Decritta il codice che segue da 0x00BA1101
00BA10ED	00B5 75294400	MOV EAX,DWORD PTR SS:[EBP+442975]	
00BA10F3	C0 00000000	PUSH 0000	
00BA10F8	6A 00	PUSH 0	
00BA10FA	50	PUSH EAX	
00BA10FB	FF95 70294400	CALL NEAR DWORD PTR SS:[EBP+44297D]	
00BA1101	00B5 512C4400	LEA EAX,DWORD PTR SS:[EBP+442C51]	
00BA1107	50	PUSH EAX	0x00BA1300 -> offset 0x0031300
00BA1108	C3	RETN	
00BA1109	0000	ADD BYTE PTR DS:[EAX],AL	
00BA110B	0000	ADD BYTE PTR DS:[EAX],AL	

Now, we must be coming to a new offset 0x0031303D.

Here we can point to the code patching our

Offset (redirection # 6): 0x00310F3

ORIGINAL Instruction: PUSH 8000 -> 0x68 0x00 0x80 0x00 0x00

Looking a bit through offset 0x002663 window dump Olly's view is considered to decrypt not, of course not, we continue thui.

RETN to F7, F8, teip to trace, is to offset 0x0031343, have found a VirtualAlloc API function calls in offset 0x0031343

00BA130B	0000	ADD BYTE PTR DS:[EAX],AL	
00BA130D	8B9D 552A4400	MOV EBX,DWORD PTR SS:[EBP+442A55]	
00BA1313	0BDB	OR EBX,EBX	
00BA1315	74 0A	JE SHORT 00BA1321	
00BA1317	8B03	MOV EAX,DWORD PTR DS:[EBX]	
00BA1319	8785 592A4400	XCHG DWORD PTR SS:[EBP+442A59],EAX	
00BA131F	8903	MOV DWORD PTR DS:[EBX],EAX	
00BA1321	8DB5 712A4400	LEA ESI,DWORD PTR SS:[EBP+442A71]	
00BA1327	833E 00	CMP DWORD PTR DS:[ESI],0	
00BA132A	0F84 03000000	JE 00BA1403	
00BA1330	8DB5 712A4400	LEA ESI,DWORD PTR SS:[EBP+442A71]	
00BA1336	8B46 04	MOV EAX,DWORD PTR DS:[ESI+4]	
00BA1339	6A 04	PUSH 4	
00BA133B	68 00100000	PUSH 1000	
00BA1340	50	PUSH EAX	
00BA1341	6A 00	PUSH 0	
00BA1343	FF95 79294400	CALL NEAR DWORD PTR SS:[EBP+442979]	call to VirtualAl
00BA1349	8985 75294400	MOV DWORD PTR SS:[EBP+442975],EAX	
00BA134F	56	PUSH ESI	
00BA1350	8B1E	MOV EBX,DWORD PTR DS:[ESI]	
00BA1352	039D 08304400	ADD EBX,DWORD PTR SS:[EBP+4430D8]	
00BA1358	50	PUSH EAX	
00BA1359	53	PUSH EBX	
00BA135A	E8 F2020000	CALL 00BA1651	
00BA135F	00BD 70294400	CMP BYTE PTR SS:[EBP+442970],0	
00BA1366	75 4C	JNZ SHORT 00BA13B4	
00BA1368	FE05 70294400	INC BYTE PTR SS:[EBP+442970]	
00BA136E	8B3E	MOV EDI,DWORD PTR DS:[ESI]	
00BA1370	03BD 08304400	ADD EDI,DWORD PTR SS:[EBP+4430D8]	
00BA1376	FF37	PUSH DWORD PTR DS:[EDI]	
00BA1378	C607 C3	MOV BYTE PTR DS:[EDI],0C3	
00BA137B	FFD7	CALL NEAR EDI	
00BA137D	8F07	POP DWORD PTR DS:[EDI]	
00BA137F	50	PUSH EAX	
00BA1380	51	PUSH ECX	
00BA1381	56	PUSH ESI	
00BA1382	53	PUSH EBX	
00BA1383	8BC8	MOV ECX,EAX	
00BA1385	83E9 06	SUB ECX,6	
00BA1388	00B5 75294400	MOV ESI,DWORD PTR SS:[EBP+442975]	
00BA138E	33DB	XOR EBX,EBX	
00BA1390	0BC9	OR ECX,ECX	
00BA1392	74 1C	JE SHORT 00BA13B0	
00BA1394	78 1A	JS SHORT 00BA13B0	
00BA1396	AC	LODS BYTE PTR DS:[ESI]	
00BA1397	3C E8	CMP AL,0E8	
00BA1399	74 08	JE SHORT 00BA13A3	
00BA139B	3C E9	CMP AL,0E9	
00BA139D	74 04	JF SHORT 00BA13B0	

00BA1399	74 08	JE SHORT 00BA13A3	
00BA139B	3C E9	CMP AL,0E9	
00BA139D	74 04	JE SHORT 00BA13A3	
00BA139F	43	INC EBX	
00BA13A0	49	DEC ECX	
00BA13A1	EB ED	JMP SHORT 00BA1390	
00BA13A3	291E	SUB DWORD PTR DS:[ESI],EBX	
00BA13A5	83C3 05	ADD EBX,5	
00BA13A8	83C6 04	ADD ESI,4	
00BA13AB	83E9 05	SUB ECX,5	
00BA13AE	EB E0	JMP SHORT 00BA1390	
00BA13B0	5B	POP EBX	
00BA13B1	5E	POP ESI	
00BA13B2	59	POP ECX	
00BA13B3	58	POP EAX	
00BA13B4	8BC8	MOV ECX,EAX	
00BA13B6	8B3E	MOV EDI,DWORD PTR DS:[ESI]	
00BA13B8	03BD 08304400	ADD EDI,DWORD PTR SS:[EBP+4430D8]	
00BA13BE	8BB5 75294400	MOV ESI,DWORD PTR SS:[EBP+442975]	
00BA13C4	C1F9 02	SAR ECX,2	
00BA13C7	F3:A5	REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]	Scrittura codice
00BA13C9	8BC8	MOV ECX,EAX	
00BA13CB	83E1 03	AND ECX,3	
00BA13CE	F3:A4	REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	
00BA13D0	5E	POP ESI	
00BA13D1	8BB5 75294400	MOV EAX,DWORD PTR SS:[EBP+442975]	
00BA13D7	68 00800000	PUSH 8000	
00BA13DC	6A 00	PUSH 0	
00BA13DE	50	PUSH EAX	
00BA13DF	FF95 7D294400	CALL NEAR DWORD PTR SS:[EBP+44297D]	
00BA13E5	83C6 08	ADD ESI,8	
00BA13E8	833E 00	CMP DWORD PTR DS:[ESI],0	
00BA13EB	0F85 45FFFFFF	JNZ 00BA1336	
00BA13F1	8B9D 552A4400	MOV EBX,DWORD PTR SS:[EBP+442A55]	
00BA13F7	0BD8	OR EBX,EBX	
00BA13F9	74 08	JE SHORT 00BA1403	
00BA13FB	8B03	MOV EAX,DWORD PTR DS:[EBX]	
00BA13FD	8785 592A4400	XCHG DWORD PTR SS:[EBP+442A59],EAX	
00BA1403	8B95 08304400	MOV EDX,DWORD PTR SS:[EBP+4430D8]	
00BA1409	8BB5 512A4400	MOV EAX,DWORD PTR SS:[EBP+442A51]	
00BA140F	2BD0	SUB EDX,EAX	
00BA1411	74 75	JE SHORT 00BA1488	

Continue F8 to offset 00BA13C9, here we have found a loop decryption.

Again, note the address has ordered PUSH 8000, can point to the code of patching it.

Offset (redirection # 7): 0x00313D7

Instruction **ORIGINATES** added: PUSH 8000 -> 0x68 0x00 0x80 0x00 0x00

Press F8 code will be written

Address	Hex dump	Disassembly	Co
00B72663	F3:A5	REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]	
00B72665	89C1	MOV ECX,EAX	
00B72667	83E1 03	AND ECX,3	
00B7266A	F3:A4	REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	
00B7266C	5F	POP EDI	
00B7266D	5E	POP ESI	
00B7266E	C3	RETN	
00B7266F	8D740E FC	LEA ESI,DWORD PTR DS:[ESI+ECX-4]	

Continue gradually, we will go to the code with the IAT, we have 2 loop mixed.

00BA1494	03F2	ADD ESI,EDX	
00BA1496	8B46 0C	MOV EAX,DWORD PTR DS:[ESI+C]	inizio loop esterno
00BA1499	85C0	TEST EAX,EAX	
00BA149B	0F84 0A010000	JE 00BA15AB	
00BA14A1	03C2	ADD EAX,EDX	
00BA14A3	8BD9	MOV EBX,EAX	
00BA14A5	5B	PUSH EAX	None API
00BA14A6	FF95 EC314400	CALL NEAR DWORD PTR SS:[EBP+4431EC]	call to kernel32.GetModuleHandleA
00BA14AC	85C0	TEST EAX,EAX	
00BA14AE	75 07	JNZ SHORT 00BA14B7	
00BA14B0	53	PUSH EBX	
00BA14B1	FF95 F0314400	CALL NEAR DWORD PTR SS:[EBP+4431F0]	
00BA14B7	8985 4D294400	MOV DWORD PTR SS:[EBP+44294D],EAX	
00BA14B0	C785 51294400	MOV DWORD PTR SS:[EBP+442951],0	
00BA14C7	8B95 D8304400	MOV EDX,DWORD PTR SS:[EBP+4430D8]	inizio loop interno
00BA14C0	8B06	MOV EAX,DWORD PTR DS:[ESI]	
00BA14CF	85C0	TEST EAX,EAX	
00BA14D1	75 03	JNZ SHORT 00BA14D6	
00BA14D3	8B46 10	MOV EAX,DWORD PTR DS:[ESI+10]	
00BA14D6	03C2	ADD EAX,EDX	
00BA14D8	0385 51294400	ADD EAX,DWORD PTR SS:[EBP+442951]	
00BA14DE	8B18	MOV EBX,DWORD PTR DS:[EAX]	
00BA14E0	8B7E 10	MOV EDI,DWORD PTR DS:[ESI+10]	
00BA14E3	03FA	ADD EDI,EDX	
00BA14E5	038D 51294400	ADD EDI,DWORD PTR SS:[EBP+442951]	
00BA14EB	85D8	TEST EBX,EBX	
00BA14ED	0F84 A2000000	JE 00BA1595	
00BA14F3	F7C3 00000000	TEST EBX,00000000	
00BA14F9	75 04	JNZ SHORT 00BA14FF	
00BA14FB	03DA	ADD EBX,EDX	
00BA14FD	43	INC EBX	
00BA14FE	43	INC EBX	
00BA14FF	53	PUSH EBX	
00BA1500	81E3 FFFFFFFF	AND EBX,7FFFFFFF	
00BA1506	53	PUSH EBX	none API di cui si cerca l'indirizzo
00BA1507	FFB5 4D294400	PUSH DWORD PTR SS:[EBP+44294D]	
00BA1500	FF95 E8314400	CALL NEAR DWORD PTR SS:[EBP+4431E8]	call to kernel32.GetProcAddress
00BA1513	85C0	TEST EAX,EAX	
00BA1515	5B	POP EBX	
00BA1516	75 6F	JNZ SHORT 00BA1587	
00BA1518	F7C3 00000000	TEST EBX,00000000	
00BA151E	75 19	JNZ SHORT 00BA1539	
00BA1520	57	PUSH EDI	
00BA1521	8B46 0C	MOV EAX,DWORD PTR DS:[ESI+C]	
00BA1524	0385 D8304400	ADD EAX,DWORD PTR SS:[EBP+4430D8]	
00BA152A	5B	PUSH EAX	
00BA152B	53	PUSH EBX	
00BA152C	8D85 53314400	LEA EAX,DWORD PTR SS:[EBP+443153]	
00BA1532	5B	PUSH EAX	
00BA1533	57	PUSH EDI	
00BA1534	E9 99000000	JMP 00BA15D2	
00BA1539	81E3 FFFFFFFF	AND EBX,7FFFFFFF	
00BA153F	8B85 DC304400	MOV EAX,DWORD PTR SS:[EBP+4430DC]	
00BA1545	3985 4D294400	CMP DWORD PTR SS:[EBP+44294D],EAX	
00BA154B	75 24	JNZ SHORT 00BA1571	
00BA154D	57	PUSH EDI	
00BA154E	8BD3	MOV EDX,EBX	
00BA1550	4A	DEC EDX	
00BA1551	C1E2 02	SHL EDX,2	
00BA1554	8B9D 4D294400	MOV EBX,DWORD PTR SS:[EBP+44294D]	
00BA155A	8B7B 3C	MOV EDI,DWORD PTR DS:[EBX+3C]	
00BA155D	8B7C3B 78	MOV EDI,DWORD PTR DS:[EBX+EDI+78]	
00BA1561	035C3B 1C	ADD EBX,DWORD PTR DS:[EBX+EDI+1C]	
00BA1565	8B0413	MOV EAX,DWORD PTR DS:[EBX+EDX]	
00BA156A	0385 4D294400	ADD EAX,DWORD PTR SS:[EBP+44294D]	
00BA156E	5F	POP EDI	
00BA156F	EB 16	JMP SHORT 00BA1587	
00BA1571	57	PUSH EDI	
00BA1572	8B46 0C	MOV EAX,DWORD PTR DS:[ESI+C]	
00BA1575	0385 D8304400	ADD EAX,DWORD PTR SS:[EBP+4430D8]	
00BA157B	5B	PUSH EAX	
00BA157C	53	PUSH EBX	
00BA157D	8D85 A4314400	LEA EAX,DWORD PTR SS:[EBP+4431A4]	
00BA1583	5B	PUSH EAX	
00BA1584	57	PUSH EDI	
00BA1585	EB 4B	JMP SHORT 00BA15D2	
00BA1587	8907	MOV DWORD PTR DS:[EDI],EAX	Scrive l'indirizzo risolto dell'API
00BA1589	8385 51294400	ADD DWORD PTR SS:[EBP+442951],4	Si sposta nella prossima entry da scrivere
00BA1590	E9 32FFFFFF	JMP 00BA14C7	continua loop interno
00BA1595	8906	MOV DWORD PTR DS:[ESI],EAX	
00BA1597	8946 0C	MOV DWORD PTR DS:[ESI+C],EAX	
00BA159A	8946 10	MOV DWORD PTR DS:[ESI+10],EAX	
00BA159D	03C6 14	ADD ESI,14	
00BA15A0	8B95 D8304400	MOV EDX,DWORD PTR SS:[EBP+4430D8]	
00BA15A6	E9 EBFEFFFF	JMP 00BA1496	continua loop esterno
00BA15AB	8B85 652A4400	MOV EAX,DWORD PTR SS:[EBP+442A65]	
00BA15B1	5B	PUSH EAX	
00BA15B2	0385 D8304400	ADD EAX,DWORD PTR SS:[EBP+4430D8]	

Decode the API will be recorded from 0x002C104 offset, and end position loop is 0x00315AB

00B9C0F8	00000000	
00B9C0FC	00000000	
00B9C100	00000000	
00B9C104	7C809737	kernel32.GetCurrentThreadId
00B9C108	7C92188A	ntdll.RtlDeleteCriticalSection
00B9C10C	7C9110ED	ntdll.RtlLeaveCriticalSection
00B9C110	7C911005	ntdll.RtlEnterCriticalSection
00B9C114	7C809FA1	kernel32.InitializeCriticalSection
00B9C118	7C809B14	kernel32.VirtualFree
00B9C11C	7C809A81	kernel32.VirtualAlloc
00B9C120	7C8099CD	kernel32.LocalFree
00B9C124	7C80998D	kernel32.LocalAlloc
00B9C128	7C80B859	kernel32.VirtualQuery
00B9C12C	7C80A0C7	kernel32.WideCharToMultiByte
00B9C130	7C809CAD	kernel32.MultiByteToWideChar
00B9C134	7C80C6E0	kernel32.lstrlenA
00B9C138	7C810311	kernel32.lstrcpyA
00B9C13C	7C80C729	kernel32.lstrcpyA
00B9C140	7C801D4F	kernel32.LoadLibraryExA
00B9C144	7C80A405	kernel32.GetThreadLocale
00B9C148	7C801EEE	kernel32.GetStartupInfoA

Some next steps we will keep to the process of the classic ASPACK.

00BA1583	50	PUSH EAX	
00BA1584	57	PUSH EDI	
00BA1585	EB 4B	JMP SHORT 00BA15D2	
00BA1587	9907	MOV DWORD PTR DS:[EDI],EAX	Scrive l'indirizzo risolto dell'API
00BA1589	8385 51294400	ADD DWORD PTR SS:[EBP+442951],4	Si sposta nella prossima entry da scrivere
00BA1590	E9 32FFFFFF	JMP 00BA14C7	continua loop interno
00BA1595	8906	MOV DWORD PTR DS:[ESI],EAX	
00BA1597	8946 0C	MOV DWORD PTR DS:[ESI+C],EAX	
00BA159A	8946 10	MOV DWORD PTR DS:[ESI+10],EAX	
00BA159D	83C6 14	ADD ESI,14	
00BA15A0	8B95 08304400	MOV EDX,DWORD PTR SS:[EBP+4430D8]	
00BA15A6	E9 EBFFFFFF	JMP 00BA1496	continua loop esterno
00BA15A8	8B85 652A4400	MOV EAX,DWORD PTR SS:[EBP+442A65]	
00BA15B1	50	PUSH EAX	
00BA15B2	0385 08304400	ADD EAX,DWORD PTR SS:[EBP+4430D8]	
00BA15B8	5B	POP EBX	
00BA15B9	0BD8	OR EBX,EBX	
00BA15BB	8B85 112F4400	MOV DWORD PTR SS:[EBP+442F11],EAX	Scrive l'indirizzo a cui saltare
00BA15C1	61	POPAD	
00BA15C2	75 08	JNZ SHORT 00BA15CC	
00BA15C4	B8 01000000	MOV EAX,1	
00BA15C9	C2 0C00	RETN 0C	
00BA15CC	68 00000000	PUSH 0	
00BA15D1	C3	RETN	
00BA15D2	8B85 DC304400	MOV EAX,DWORD PTR SS:[EBP+4430DC]	

We trace, tase shows deim kahc duoc fold them.

00BA1587	8907	MOV DWORD PTR DS:[EDI],EAX	
00BA1589	8385 51294400	ADD DWORD PTR SS:[EBP+442951],4	Scrive l'indirizzo risolto dell'API
00BA1590	E9 32FFFFFF	JMP 00BA14C7	Si sposta nella prossima entry da scrivere
00BA1595	8906	MOV DWORD PTR DS:[ESI],EAX	continua loop interno
00BA1597	8946 0C	MOV DWORD PTR DS:[ESI+C],EAX	
00BA159A	8946 10	MOV DWORD PTR DS:[ESI+10],EAX	
00BA159D	83C6 14	ADD ESI,14	
00BA15A0	8B95 08304400	MOV EDX,DWORD PTR SS:[EBP+4430D8]	
00BA15A6	E9 EBFFFFFF	JMP 00BA1496	continua loop esterno
00BA15A8	8B85 652A4400	MOV EAX,DWORD PTR SS:[EBP+442A65]	
00BA15B1	50	PUSH EAX	
00BA15B2	0385 08304400	ADD EAX,DWORD PTR SS:[EBP+4430D8]	
00BA15B8	5B	POP EBX	
00BA15B9	0BD8	OR EBX,EBX	
00BA15BB	8B85 112F4400	MOV DWORD PTR SS:[EBP+442F11],EAX	Scrive l'indirizzo a cui saltare
00BA15C1	61	POPAD	
00BA15C2	75 08	JNZ SHORT 00BA15CC	
00BA15C4	B8 01000000	MOV EAX,1	
00BA15C9	C2 0C00	RETN 0C	
00BA15CC	68 DC9B9000	PUSH 0B9B90C	
00BA15D1	C3	RETN	
00BA15D2	8B85 DC304400	MOV EAX,DWORD PTR SS:[EBP+4430DC]	

Navigation to our code is placed here 0x00315C1 offset.

Offset (redirection # 8): 0x00315C1

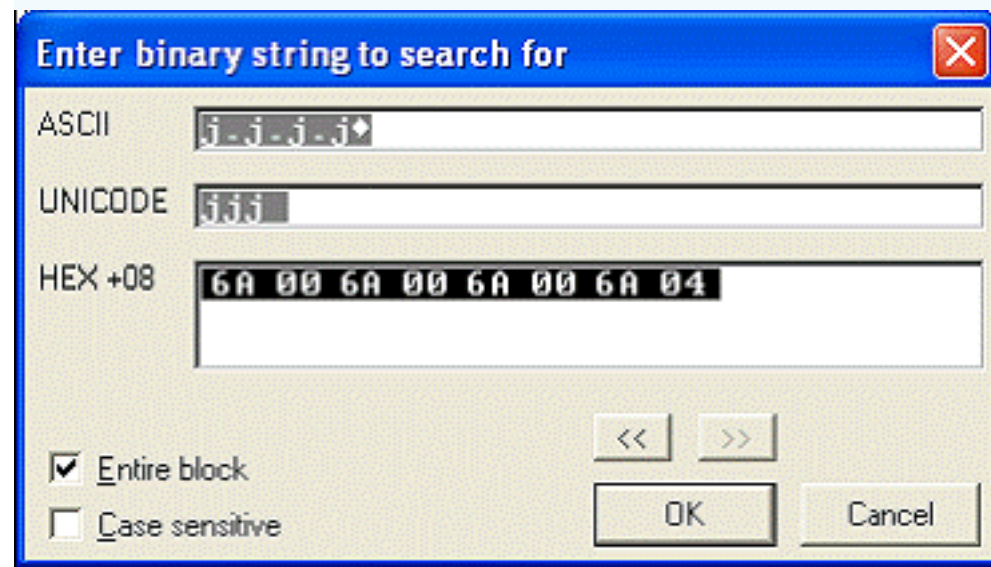
ORIGINAL Instruction: POPAD / JNZ

00B989DC	55	PUSH EBP	chordpic.004EB430
00B989DD	8BEC	MOV EBP,ESP	
00B989DF	83C4 B4	ADD ESP,-4C	
00B989E2	B8 0487B900	MOV EAX,0B987D4	
00B989E7	E9 50CCD0FF	CALL 00B7563C	
00B989EC	E8 0BAFF0FF	CALL 00B734CC	
00B989F1	8D40 00	LEA EAX,DWORD PTR DS:[EAX]	
00B989F4	0000	ADD BYTE PTR DS:[EAX],AL	
00B989F6	0000	ADD BYTE PTR DS:[EAX],AL	
00B989F8	0000	ADD BYTE PTR DS:[EAX],AL	
00B989FA	0000	ADD BYTE PTR DS:[EAX],AL	

I put a break on the breakpoints Hardware offset 0x00289DC will save time when Olly restart.

Now is the time to say CRC checking (check sovereignty on the hard disk).

Click CRL B + on the binary:

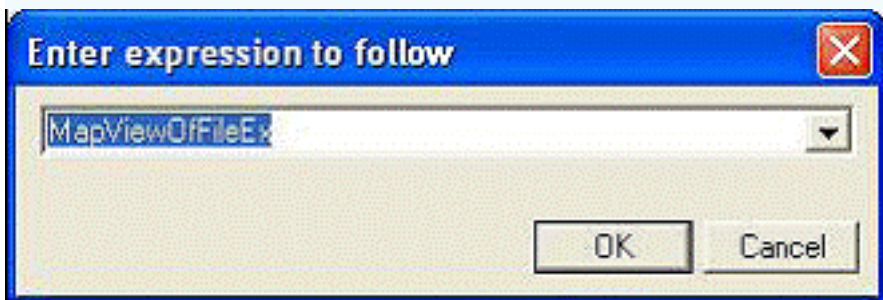


Click OK, we are taught to, offset 0018662.

00B88650	8B40 18	MOV EAX,DWORD PTR DS:[EAX+18]	
00B88653	FFD0	CALL NEAR EAX	
00B88655	833D 14B4B900	CMP DWORD PTR DS:[B9B414],0	
00B8865C	0F84 66040000	JE 00B88AC8	
00B88662	6A 00	PUSH 0	
00B88664	6A 00	PUSH 0	
00B88666	6A 00	PUSH 0	
00B88668	6A 04	PUSH 4	
00B8866A	A1 14B4B900	MOV EAX,DWORD PTR DS:[B9B414]	
00B8866F	50	PUSH EAX	
00B88670	A1 E497B900	MOV EAX,DWORD PTR DS:[B997E4]	
00B88675	8B40 08	MOV EAX,DWORD PTR DS:[EAX+8]	
00B88678	FFD0	CALL NEAR EAX	
00B8867A	8BD8	MOV EBX,EAX	
00B8867C	50	PUSH EAX	
00B8867D	E8 4A010000	CALL 00B887CC	
.....

Code this relationship to the mapped file through MapViewOfFileEx function that is called by CALL EAX directives.

Set break points in MapViewOfFileEx function.



7C80B71E	8BFF	MOV EDI,EDI
7C80B720	55	PUSH EBP
7C80B721	8BEC	MOV EBP,ESP
7C80B723	51	PUSH ECX
7C80B724	51	PUSH ECX
7C80B725	8B45 14	MOV EAX,DWORD PTR SS:[EBP+14]
7C80B728	8945 F8	MOV DWORD PTR SS:[EBP-8],EAX
7C80B72B	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]
7C80B72E	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX
7C80B731	8B45 18	MOV EAX,DWORD PTR SS:[EBP+18]
7C80B734	8945 10	MOV DWORD PTR SS:[EBP+10],EAX
7C80B737	8B45 1C	MOV EAX,DWORD PTR SS:[EBP+1C]
7C80B73H	8945 14	MOV DWORD PTR SS:[EBP+14],EAX
7C80B73D	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
7C80B740	83F8 01	CMP EAX,1
7C80B743	0F84 8F150300	JE kernel32.7C83CCD8
7C80B749	00 00	TEST AL,0

00A20000	8BFF	MOV EDI,EDI
00A20002	55	PUSH EBP
00A20003	8BEC	MOV EBP,ESP
00A20005	6A 00	PUSH 0
00A20007	FF75 18	PUSH DWORD PTR SS:[EBP+18]
00A2000A	FF75 14	PUSH DWORD PTR SS:[EBP+14]
00A2000D	FF75 10	PUSH DWORD PTR SS:[EBP+10]
00A20010	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
00A20013	FF75 08	PUSH DWORD PTR SS:[EBP+8]
00A20016	68 A8B7807C	PUSH 7C80B7A8
00A20018	68 1EB7807C	PUSH kernel32.MapViewOfFileEx
00A20020	C3	RETN
00A20021	0000	ADD BYTE PTR DS:[EAX],AL
00A20023	0000	ADD BYTE PTR DS:[EAX],AL
00A20025	0000	ADD BYTE PTR DS:[EAX],AL

We hold that end, then we repeat the implementation until we meet in the code (using a combination ALT + M break on access on running and then ALT + F9 and Shift + F9), then we can find the key point then we have the image files are mapped into memory, you use procedure nnie search to find the key point.

In Asprotect, MapViewOfFile API used to create image files on the hard drive of us, the test will detect a code that replaced just been recorded by a hardcode way. Check this check the original file.

NOTE for **MapViewOfFile**

From the MSDN it is had:

MapViewOfFile

The **MapViewOfFile** function maps a view of a file mapping into the address space of a calling process.
To specify a suggested base address for the view, use the **MapViewOfFileEx** function. However, this practice is not recommended.

```
LDVOID MapViewOfFile(  
    HANDLE hFileMappingObject,  
    DWORD dwDesiredAccess,  
    DWORD dwFileOffsetHigh,  
    DWORD dwFileOffsetLow,  
    SIZE_T dwNumberOfBytesToMap  
);
```

PUSH 4 sets up the flag for the modalities of access to the area that contains the file image (mapping) to value FILE_MAP_READ, we must have the way to modify the byte inside of the mapped image in order to fake ASProtect and show that the file was not modified. In order to do this we must change the attribute of access to FILE_MAP_COPY that correspond to the value 0x01, therefore we will have to replace the original PUSH 4 with a new PUSH 1 and later restore the original code to avoid other detection from ASProtect.

FILE_MAP_COPY = 0x01
FILE_MAP_WRITE = 0x02
FILE_MAP_READ = 0x04

Value	Meaning
FILE_MAP_WRITE	Read/write access. The mapping object must be created with PAGE_READWRITE protection. A read/write view of the file is mapped.
FILE_MAP_READ	Read-only access. The mapping object must be created with PAGE_READWRITE or PAGE_READONLY protection. A read-only view of the file is mapped.
FILE_MAP_COPY	<p>Copy-on-write access. The mapping object must be created with PAGE_WRITECOPY protection.</p> <p>The system commits physical storage from the paging file at the time that MapViewOfFile is called. The actual physical storage is not used until a thread in the process writes to an address in the view. At that time, the system copies the original page to a new page that is backed by the paging file, maps the page into the process address space, and changes the page protection to PAGE_READWRITE. The threads in the process can access only the local copy of the data, not the original data. If the page is ever trimmed from the working set of the process, it can be written to the paging file storage that is committed when MapViewOfFile is called.</p> <p>This process only allocates physical memory when a virtual address is actually written to. Changes are never written back to the original file, and are freed when the thread in your process unmaps the view.</p> <p>Paging file space for the entire view is committed when copy-on-write access is specified, because the thread in the process can write to every single page. Therefore, enough physical storage space must be obtained at the time MapViewOfFile is called.</p>
FILE_MAP_EXECUTE	<p>Execute access. Code can be run from the mapped memory. The mapping object must be created with PAGE_EXECUTE_READWRITE or PAGE_EXECUTE_READ access.</p> <p>Windows Server 2003 and Windows XP: This feature is not available until Windows XP SP2 and Windows Server 2003 SP1.</p>

Order PUSH 4 position OFFSET is 0x0018669, and we will re-navigate after MaViewOfFile

enforcement.

Offset (redirection # 9): 0x001867A

ORIGINAL instruction: MOV EBX, EAX

Redirection code into the cave we will first use the value which is in EAX since the content of this registry base supplies the address from where it begins the mapping file in memory.

After a few trace what we came here:

00B88EE9	83C5 18	ADD EBP,18	
00B88EEC	6A 04	PUSH 4	
00B88EEE	8D4D E4	LEA ECX,DWORD PTR SS:[EBP-1C]	
00B88EF1	8B15 20B4B900	MOV EDX,DWORD PTR DS:[B9B420]	
00B88EF7	A1 1CB4B900	MOV EAX,DWORD PTR DS:[B9B41C]	
00B88EFC	E8 3389FFFF	CALL 00B81834	
00B88F01	EB 0A	JMP SHORT 00B88F0D	
00B88F03	68 7892B800	PUSH 0B89278	ASCII "617E"
00B88F08	E8 0FC1FFFF	CALL 00B8501C	
00B88F0D	8BC6	MOV EAX,ESI	
00B88F0F	E8 0C9CFE FF	CALL 00B72B20	
00B88F14	52	PUSH EDX	
00B88F15	E8 2B000000	CALL 00B88F45	
00B88F1A	57	PUSH EDI	

We gradually tieop axis after the test and after a few other trace our final round to that code decompression program.

00B890C6	C600 CF	MOV BYTE PTR DS:[EAX],0CF	Inizio loop
00B890C9	EB 71	JMP SHORT 00B8913C	
00B890CB	E8 7494FEFF	CALL 00B72544	
00B890D0	8BF0	MOV ESI,EAX	
00B890D2	8B03	MOV EAX,DWORD PTR DS:[EBX]	
00B890D4	0345 E8	ADD EAX,DWORD PTR SS:[EBP-18]	
00B890D7	8945 FC	MOV DWORD PTR SS:[EEP-4],EAX	
00B890DA	8B4B 04	MOV ECX,DWORD PTR DS:[EBX+4]	
00B890DD	8BD6	MOV EDX,ESI	
00B890DF	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00B890E2	E8 4DEFFFFF	CALL 00B88034	
00B890E7	8B7B 04	MOV EDI,DWORD PTR DS:[EBX+4]	
00B890EA	8A45 FC	MOV AL,BYTE PTR SS:[EBP-4]	
00B890ED	8B15 5497B900	MOV EDX,DWORD PTR DS:[B99754]	
00B890F3	8B02	MOV BYTE PTR DS:[EDX],AL	
00B890F5	8BC7	MOV EAX,EDI	
00B890F7	8B15 6C97B900	MOV EDX,DWORD PTR DS:[B9976C]	
00B890FD	8B02	MOV BYTE PTR DS:[EDX],AL	
00B890FF	007D F7 00	CMP BYTE PTR SS:[EBF-9],0	
00B89103	75 1E	JNZ SHORT 00B89123	
00B89105	C645 F7 01	MOV BYTE PTR SS:[EBF-9],1	Decomprime il codice
00B89109	56	PUSH ESI	
00B8910A	8B75 FC	MOV ESI,DWORD PTR SS:[EBP-4]	
00B8910D	83C6 14	ADD ESI,14	
00B89110	FF36	PUSH DWORD PTR DS:[ESI]	
00B89112	C606 C3	MOV BYTE PTR DS:[ESI],0C3	
00B89115	FFD6	CALL NEAR ESI	
00B89117	8F06	POP DWORD PTR DS:[ESI]	
00B89119	5E	POP ESI	
00B8911A	8BD7	MOV EDX,EDI	
00B8911C	8BC6	MOV EAX,ESI	
00B8911E	E8 75F0FFFF	CALL 00B88198	
00B89123	8BCF	MOV ECX,EDI	
00B89125	8BD6	MOV EDX,ESI	
00B89127	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00B8912A	E8 8DC7FEFF	CALL 00B758BC	
00B8912F	8B53 04	MOV EDX,DWORD PTR DS:[EBX+4]	
00B89132	8BC6	MOV EAX,ESI	
00B89134	E8 2394FEFF	CALL 00B7255C	
00B89139	83C3 0C	ADD EBX,0C	Continua il loop
00B8913C	8B43 04	MOV EAX,DWORD PTR DS:[EBX+4]	
00B8913F	85C0	TEST EAX,EAX	Fine loop
00B89141	77 88	JA SHORT 00B890CB	
00B89143	837D F0 00	CMP DWORD PTR SS:[EEP-10],0	
00B89147	74 08	JE SHORT 00B89151	
00B89149	8B45 F0	MOV EAX,DWORD PTR SS:[EBP-10]	
00B8914C	8B55 EC	MOV EDX,DWORD PTR SS:[EBP-14]	
00B8914F	8910	MOV DWORD PTR DS:[EFX],EDX	
00B89151	B8 03000000	MOV EAX,3	

Address	Hex dump	Disassembly	Comment
00499914	55	DB 55	CHAR 'U'
00499915	8B	DB 8B	
00499916	EC	DB EC	
00499917	83	DB 83	
00499918	C4	DB C4	
00499919	EC	DB EC	
0049991A	53	DB 53	CHAR 'S'
0049991B	33	DB 33	CHAR '3'
0049991C	C0	DB C0	
0049991D	89	DB 89	
0049991E	45	DB 45	CHAR 'E'
0049991F	EC	DB EC	

To see the code of Olly's code window, CRL + G on the address of the OEP, then CRL + A

00499914	55	PUSH EBP	chordpic.0049FBC0
00499915	8BEC	MOV EBP,ESP	
00499917	83C4 EC	ADD ESP,-14	
0049991A	53	PUSH EBX	
0049991B	33C0	XOR EAX,EAX	
0049991D	8945 EC	MOV DWORD PTR SS:[EBP-14],EAX	
00499920	B8 24964900	MOV EAX,chordpic.00499624	
00499925	E8 86CDF6FF	CALL chordpic.004066B0	
0049992A	8B1D F0E04900	MOV EBX,DWORD PTR DS:[49E0F0]	
00499930	33C0	XOR EAX,EAX	
00499932	55	PUSH EBP	
00499933	68 4E9A4900	PUSH chordpic.00499A4E	

Well, to trace axis, ignoring the performance and you can see a loop test to detect memory patching.

00B7CED7	83FB 40	CMP EBX, 40	
00B7CEDA	7C 14	JL SHORT 00B7CEF0	
00B7CEDC	8B07	MOV EDX, EDI	
00B7CEDE	8BC6	MOV EAX, ESI	
00B7CEE0	8B08	MOV ECX, DWORD PTR DS:[EAX]	
00B7CEE2	FF51 14	CALL NEAR DWORD PTR DS:[ECX+14]	
00B7CEES	83C7 40	ADD EDI, 40	
00B7CEE8	83EB 40	SUB EBX, 40	
00B7CEFA	83FA 40	CMP FAX, 40	
00B7CEEE	7D EC	JGE SHORT 00B7CEDC	
00B7CEFF	8B0424	MOV EAX, DWORD PTR SS:[ESP]	Loop lettura codice per verifica CRC
00B7CF03	03C5	ADD EAX, EBP	chordpic.<ModuleEntryPoint>
00B7CF05	8D56 00	LEA EDX, DWORD PTR DS:[ESI+8]	
00B7CF08	8BCB	MOV ECX, EBX	
00B7CF0A	E8 5157FFFF	CALL 00B72650	
00B7CF0C	5A	POP EDX	
00B7CF0E	5D	POP EBP	
00B7CF10	5F	POP EDI	
00B7CF12	5E	POP ESI	
00B7CF14	5B	POP EBX	
00B7CF16	C3	RETN	
00B7CF18	8D40 00	LEA EAX, DWORD PTR DS:[EAX]	

Then a bit we will get a beautiful location other, this check point is 45. The check will check the code is decompress. If you patch the target true this test will occur error.

00B8A32F	8B00	MOV EAX, DWORD PTR DS:[EAX]	
00B8A331	03C7	ADD EAX, EDI	
00B8A333	8B5424 0C	MOV EDX, DWORD PTR SS:[ESP+C]	
00B8A337	E8 D873FFFF	CALL 00B81714	
00B8A33C	8BE8	MOV EBP, EAX	EAX = 5A935349
00B8A33E	892D 3CB4B900	MOV DWORD PTR DS:[B9B43C], EBP	
00B8A344	3B6C24 10	CMP EBP, DWORD PTR SS:[ESP+10]	[ESP+10]=5A935349 (CRC)
00B8A348	74 0C	JE SHORT 00B8A356	
00B8A34A	68 DCA3B800	PUSH 00B8A3DC	ASCII "45JQ"
00B8A34F	E8 C8ACFFFF	CALL 00B8501C	
00B8A354	EB 12	JMP SHORT 00B8A368	
00B8A356	8B4424 0C	MOV EAX, DWORD PTR SS:[ESP+C]	
00B8A35A	A3 38B4B900	MOV DWORD PTR DS:[B9B438], EAX	
00B8A35F	8B4424 08	MOV EAX, DWORD PTR SS:[ESP+8]	
00B8A363	A3 34B4B900	MOV DWORD PTR DS:[B9B434], EAX	

CRC will be stored in a short position in the ratings [ESP +10] and the target is 5A935349.

0012FF50	00000000	
0012FF54	00400000	ASCII "MZP"
0012FF58	00001000	
0012FF5C	00098C00	
0012FF60	5A935349	CRC
0012FF64	0012FF98	
0012FF68	00B70000	
0012FF6C	00B30000	

Now you can patch in position 0x0048CB79 only after a check this, the program will run well and the patch will work.

0048CB67	. 80 EU	MOV AL, 80U	
0048CB69	. CD 45	INT 45	
0048CB6B	> E8 C0F9FFFF	CALL chordpic.0048C530	
0048CB70	. 84CE	TEST AL, AL	
0048CB72	90	NOP	PATCH: NOP
0048CB73	90	NOP	
0048CB74	. A1 ECDD4900	MOV EAX, DWORD PTR DS:[49DDEC]	
0048CB79	C602 00	MOV BYTE PTR DS:[EAX], 0	PATCH: FORCE 0
0048CB7C	> A1 ECDD4900	MOV EAX, DWORD PTR DS:[49DDEC]	
0048CB81	. 803E 00	CMP BYTE PTR DS:[EAX], 0	
0048CB84	~ 0F84 F2000000	JE chordpic.0048CC7C	LOOK AT THIS :-)
0048CB8A	. 33DE	XOR EBX, EBX	
0048CB8C	. 8D5E F8	LEA EDX, DWORD PTR SS:[EBP-8]	
0048CB8F	. A1 4E14900	MOV EAX, DWORD PTR DS:[49E184]	

Next to this, we can use the check after patching, then we write the pointer to the last offset in 0x001A356.

Offset (redirection # 10): 0x001A356

ORIGINAL instruction: MOV EAX, DWORD [ESP + C]

2. Inline PATCHING

Giola now time to write the code for the inline patching and memory areas suitable to all daon the code.

Using. ADATA section are not sure, you do not charge and it will



This is the impact of integrity check. Because when MapViewOfFile enforcement, Asprotect will submit to. ADATA the giaa of zero, so we will have to check the area are Asprotect use and re-directed to the address used when checking Asprotect finished the last byte.

Making this analysis the first free area it turns out from the address 0x0050DCD6, this will be the address to redirect which **MapViewOfFile** after the execution.

We write code patching follows:

004EB1AA JMP chordpic.0050 D100; Redirection from 1 to cave hardcoded jump

0050D100 MOV DWORD PTR DS: [4EB267], 21EA3E9; Cave 1

0050D10A JMP chordpic.004EB1CA

0050D10F MOV DWORD PTR DS: [4EB32D], 21DECE9; Cave 2

0050D119 JMP chordpic.004EB290

0050D11E MOV DWORD PTR DS: [4EB410], 21D18E9; Cave 3

0050D128 JMP chordpic.004EB373

0050D12D MOV DWORD PTR DS: [4EB410], 12E8; Cave 4 (restoration of the original code of the call)

0050D137 MOV DWORD PTR DS: [4EB5E8], 21B59E9; Redirection to the cave 5

0050D141 JMP chordpic.004EB410

ChordPickout ASProtect inline patching tutorial ThunderPwr of ARTeam 05/08/2006

29

0050D146 MOV DWORD PTR DS: [4EB5E8], 800,068; Cave 5 (restoration of the code PUSH 8000)

0050D150 MOV DWORD PTR DS: [50DFFC], EDI; Storage for the base address (at the end of the area)

0050D156 MOV DWORD PTR DS: [EDI +310 F3], 50D16E68; Redirection to the cave 6

0050D160 MOV WORD PTR DS: [EDI +310 F7], 0C300

0050D169 JMP chordpic.004EB5E8
0050D16E PUSHAD; Cave 6 (Save the target context)
0050D16F PUSHFD
0050D170 MOV EAX, DWORD PTR DS: [50DFFC]; Load the base address
0050D175 MOV DWORD PTR DS: [EAX +310 F3], 800,068; Restoration PUSH 8000 (offset 310F3)
0050D17F MOV BYTE PTR DS: [EAX +310 F7], 0
0050D186 MOV BYTE PTR DS: [EAX +310 F8], 6A
0050D18D MOV DWORD PTR DS: [EAX +313 D7], 50D1B268; It goes next to PUSH 8000 (offset 313D7)
for cave 7
0050D197 MOV WORD PTR DS: [EAX +313 DB], 0C300
0050D1A0 ADD EAX, 310F3; Calculate the return address
0050D1A5 MOV DWORD PTR DS: [50D1AD], EAX
0050D1AA POPFD
0050D1AB POPAD
0050D1AC PUSH 0
0050D1B1 RETN
0050D1B2 PUSHAD; Cave 7
0050D1B3 PUSHFD
0050D1B4 MOV EAX, DWORD PTR DS: [50DFFC]
0050D1B9 MOV DWORD PTR DS: [EAX +313 D7], 800,068; Restoration PUSH 8000 to offset one 313D7
0050D1C3 MOV BYTE PTR DS: [EAX +313 DB], 0
0050D1CA MOV BYTE PTR DS: [EAX +313 DC], 6A
0050D1D1 MOV DWORD PTR DS: [EAX +315 C1], 50D1F668; Cave 8 goes to the POPAD (offset 315C1)
for cave 8
0050D1DB MOV WORD PTR DS: [EAX +315 C5], 0C300
0050D1E4 ADD EAX, 313D7; Calculate the return address
0050D1E9 MOV DWORD PTR DS: [50D1F1], EAX
0050D1EE POPFD
0050D1EF POPAD
0050D1F0 PUSH 0
0050D1F5 RETN
0050D1F6 PUSHAD; Cave 8
0050D1F7 PUSHFD
0050D1F8 MOV EAX, DWORD PTR DS: [50DFFC]
0050D1FD MOV DWORD PTR DS: [EAX +315 C1], B8087561; Restoration POPAD / JNZ to offset one
315C1
0050D207 MOV WORD PTR DS: [EAX +315 C5], 1
0050D210 MOV BYTE PTR DS: [EAX +18669], 1; Patch 4 to the PUSH PUSH 1 (offset 18669)
0050D217 MOV DWORD PTR DS: [EAX +1867 A], 50DCD668; It goes to MOV EBX, EAX for cave 9
0050D221 MOV WORD PTR DS: [EAX +1867 E], 0C300
0050D22A ADD EAX, 315C1; Calculates the return address
0050D22F MOV DWORD PTR DS: [50D237], EAX

0050D234 POPFD

0050D235 POPAD

0050D236 PUSH 0BE13D7

0050D23B RETN

Now we have also executed the **MapViewOfFile**, the first within the **area. ADATA** section has been erased from ASProtect, then the cave 9 redirection will have to be made jumping the address

0x0050DCD6.

When we're into the cave 9, since we've EAX in the base address of the file mapping image we've

to restore the image into the RAW for the **SIZE. ADATA** section and restore the code of the first

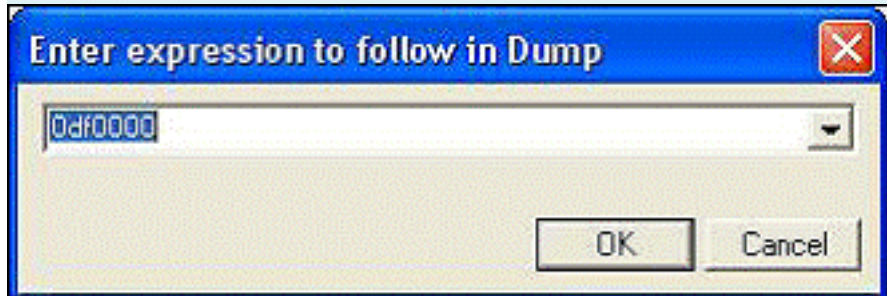
which is hardcoded jump to the address 0x004EB267.

The offset for the first JMP redirection into the image file mapping is easy to find, into the OllyDbg

dump window-press CTRL + G and write the address which is in EAX (in my case 0x00D70000) then

Press OK:

(Note: the remainder is determined to offset the cave patching, khaa also understandable that I be original, the desire for medical information because of fatigue).

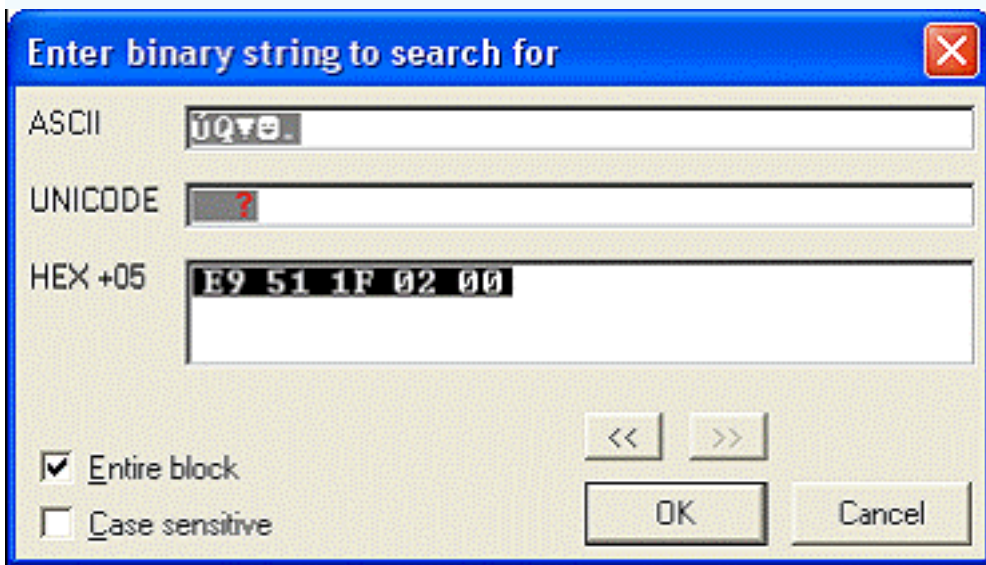


Address	Hex dump	ASCII
00DF0000	4D 5A 50 00 02 00 00 00 04 00 0F 00 FF FF 00 00	MZP.0...*. ..
00DF0010	B8 00 00 00 00 00 00 00 40 00 1A 00 00 00 00 00	@.....@.+.....
00DF0020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00DF0030	00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 000..
00DF0040	BA 10 00 0E 1F B4 09 CD 21 B8 01 4C CD 21 90 90	P.AY+. =100L =1EE
00DF0050	54 68 69 73 20 70 72 6F 67 72 61 6D 20 6D 75 73	This program mus
00DF0060	74 20 62 65 20 72 75 6E 20 75 6E 64 65 72 20 57	t be run under W
00DF0070	69 6E 33 32 0D 0A 24 37 00 00 00 00 00 00 00 00	in32..\$7.....
00DF0080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00DF0090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

then press Ctrl + B and write the pattern that we have to search to looking for the JMP offset

(also remember to check the **entire** **block**):

JMP 0050D100 -> E9 51 1F 02 00



Press OK:

Address	Hex dump	ASCII
00E537AA	E9 51 1F 02 00 F9 3E 9F EC B5 80 D4 F0 E9 40 FF	UQV0.->0ACE-UM
00E537BA	FF FF EE 8F 1C 25 FA AB 08 A1 C6 87 84 D0 52 28	"ALX" 001801IR#
00E537CA	41 E6 CE F3 77 40 FF 2D DB 64 3A 56 1C 48 17 02	ApPswM -d:ULK#0
00E537DA	58 D7 B2 6E 54 23 0F 9A 10 B6 D7 3A DB 64 EF 31	XIenT#*U>Ai: d'1
00E537EA	C8 87 A2 9E C4 D3 FE C9 80 DF 1A 94 66 43 48 89	=p6x-E=IFQ->0fChè

In order to see the code right click -> Disassemble:

Address	Hex dump	Disassembly	Comment
00E537AA	✓ E9 511F0200	JMP 00E75700	
00E537AF	F9	STC	
00E537B0	3E:9F	LAHF	Superfluous prefix
00E537B2	EC	IN AL,DX	I/O command
00E537B3	B5 80	MOV CH,80	
00E537B5	D4 F0	RAM 0F0	
00E537B7	^ E9 40FFFFFF	JMP 00E53709	
00E537BC	EE	OUT DX,AL	I/O command
00E537BD	8F	???	Unknown command
00E537BE	1C 25	SBB AL,25	
00E537C0	FA	CLI	

Well done, this is the code that we're searching for.

004EB1AA	- E9 511F0200	JMP chordpic.0050D100	LOOP#1
004EB1AF	F9	STC	
004EB1B0	3E:9F	LAHF	Superfluous prefix
004EB1B2	EC	IN HL,UX	I/O command
004EB1B3	B5 80	MOV CH,80	
004EB1B5	D4 F0	RAM 0F0	
004EB1B7	^ E9 40FFFFFF	JMP chordpic.004EB109	
004EB1BC	EE	OUT DX,AL	I/O command
004EB1BD	8F	???	Unknown command
004EB1BE	1C 25	SBB AL,25	
004EB1C0	FA	CLI	

To modify the code in order to restore the first jump in the image file is therefore found to offset **0x00637AA**.

Now we can write the code for the first cave 9.

0050DCD6 MOV BYTE PTR DS: [EAX +399], 0; Cave 9 (restores size of given raw)

0050DCDD MOV DWORD PTR DS: [EAX +637 AA], 1BE9; It restores first jump

Now we've restored the image file mapped in memory, remains to put the next redirection

just

after checking the memory.

Below the full code for cave 9:

```
0050DCD6 MOV BYTE PTR DS: [EAX +399], 0; Cave 9 (restores size of given raw)
0050DCDD MOV DWORD PTR DS: [EAX +637 AA], 1BE9; It restores first jump
0050DCE7 PUSHAD
0050DCE8 PUSHFD
0050DCE9 MOV EAX, DWORD PTR DS: [50DFFC]; It loads the base address
0050DCEE MOV BYTE PTR DS: [EAX +18669], 4; PUSH 1 -> 4 PUSH
0050DCF5 MOV DWORD PTR DS: [EAX +1867 A], E850D88B; It restores MOV EBX, EAX
0050DCFF MOV WORD PTR DS: [EAX +1867 E], 14A
0050DD08 MOV DWORD PTR DS: [EAX +1 A356], 50DD2D68; Redirezione to cave 10
0050DD12 MOV WORD PTR DS: [EAX +1 A35A], 0C300
0050DD1B ADD EAX, 1867A; it calculates the return address
0050DD20 MOV DWORD PTR DS: [50DD28], EAX
0050DD25 POPFD
0050DD26 POPAD
0050DD27 PUSH 0
0050DD2C RETN
```

From the previous analysis we know that we have to skip the check before 45 apply our patches

then we can write our last cave code.

```
0050DD2D PUSHAD; Cave 10
0050DD2E PUSHFD
0050DD2F MOV EAX, DWORD PTR DS: [50DFFC]
0050DD34 MOV DWORD PTR DS: [EAX +1 A356], 0C24448B
0050DD3E MOV WORD PTR DS: [EAX +1 A35A], 38A3
0050DD47 MOV WORD PTR DS: [48CB72], 9090; Patch 1
0050DD50 MOV BYTE PTR DS: [48CB7B], 0; Patch 2
0050DD57 ADD EAX, 1A356
0050DD5C MOV DWORD PTR DS: [50DD64], EAX
0050DD61 POPFD
0050DD62 POPAD
0050DD63 PUSH 0
0050DD68 RETN
```

That's all.

ThunderPwr of ARTeam

Thanks to all ARTeam and special thanks goes to John and H3rCul3S Madman Who for

The tutorial on ASProtect inline technique. Also great thanks to Ricardo Narvaja Cracks and all Latinos group. Thanks to that you have read all the tutorial.



Solving ASProtect 2.0 (build 2.63 alpha)

Since the ver Asprotect 1:23 cracker RC4 be the home we reconcile the ear 1 (1 may matter of personality makeup Aspr 1:23 RC4 of the experts as Moonbabe, Zombie ...), the author has asprotect more significant changes in methods of Aspr protect. Some changes, such as:

Protection mechanism OEP better

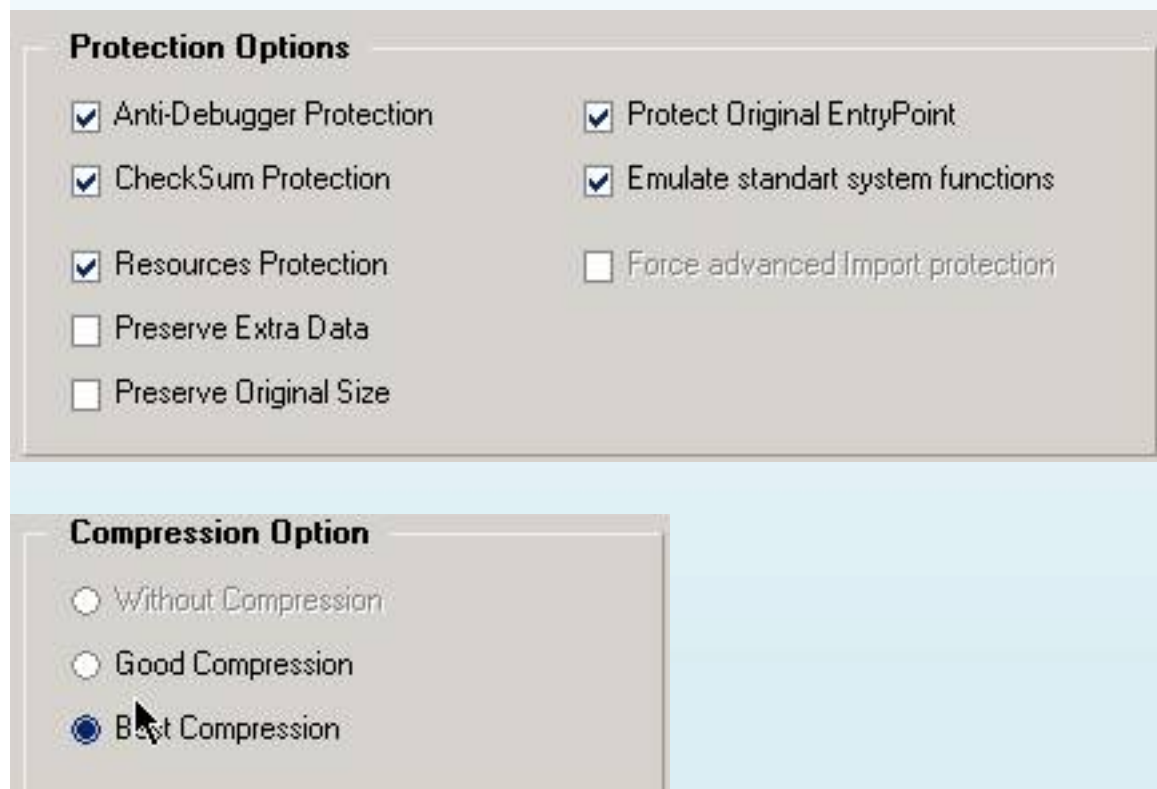
Stolen bytes instructions à Stolen

Change the encryption IAT.

Within the framework of this message, I would slide technical basis to reconcile the encryption of IAT Asprotect 2.0 alpha (while the remaining part is very easy, you or research-based knowledge from existing asprotect 1.23).

1. Preparing:

First, I search the audience, the program 1 crackme of small prdx (crackME 3). Use Aspr 2.0 with the following parameters:



The image shows two windows from the ASProtect 2.0 configuration interface. The top window, titled 'Protection Options', contains a list of checkboxes for various security features. The bottom window, titled 'Compression Option', shows three radio button options for compression levels.

Protection Options

- ☒ Anti-Debugger Protection
- ☒ CheckSum Protection
- ☒ Resources Protection
- ☐ Preserve Extra Data
- ☐ Preserve Original Size
- ☒ Protect Original EntryPoint
- ☒ Emulate standart system functions
- ☐ Force advanced Import protection

Compression Option

- ☐ Without Compression
- ☐ Good Compression
- ☒ Best Compression

and the mode is:

Modes list

Mode ID	Mode Name
1	

New Delete Copy Paste

Common Mode Properties

☒ Is this Mode Active?


Mode Name


☒ Is this Mode registered?

☐ Need password to run Mode

☐ Use Activation Keys

Mode Expires

☐ Expired days after first start 

☐ Expires executions 

☐ Expiration Date 

☐ Reset trial period on new versions

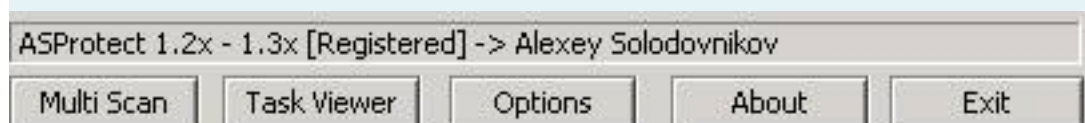
☐ Don't show messages when mode is expired

Reminder / Delay

☒ Use Reminder 

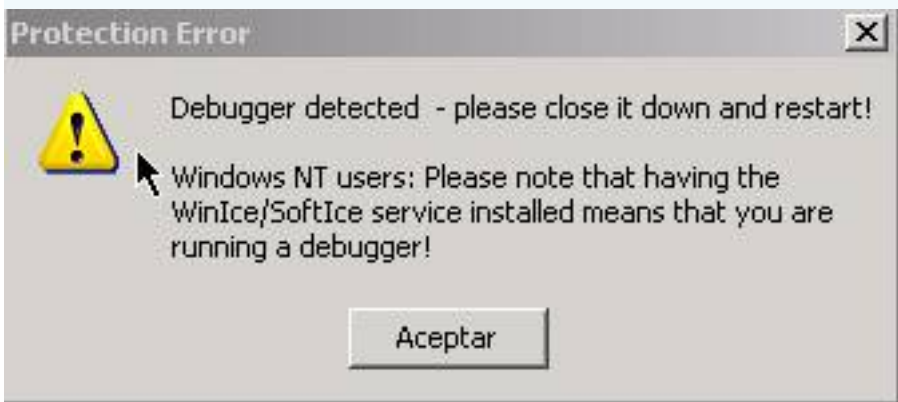
☒ Use Delay

After protect finished, use PeID to check:



2. Beginning on

Crackme3.exe to load in Olly, F9 to run, I see it appear:



Aspr can detect a debugger running, press Ctrl-F2 to load up the program, the plugin IsDebuggerPresent to leave to be detected.

Still the old methods, implementers with Shift-F9, the program will stop, the exception error, the next Shift-F9 to continue, the exception to the end (exception is the position as the stop, then when the next shift-F9, the program will be run, o need to shift the half-F9).

We in this position:

00936807	C700 7F0677B9	MOV DWORD PTR DS:[EAX],B977067F
0093680D	FB	STI
0093680E	2D F8868BEF	SUB EAX,EF8B86F8
00936813	B7 FA	MOV BH,0FA
00936815	EB 01	JMP SHORT 00936818
00936817	6967 64 8F0600	IMUL ESP,DWORD PTR DS:[EDI+64],68F

Now, set memory access breakpoint on the section of the code trinh.Nhan F9 to run it will stop in:

004085A0	55	PUSH EBP
004085A1	8BEC	MOV EBP,ESP
004085A3	83C4 F0	ADD ESP,-10
004085A6	B8 60854000	MOV EAX,crackME.00408560
004085AB	E8 B0C4FFFF	CALL crackME.00404A60

Hooray, this is the original OEP program.

To this step, you use 1 of the dump file as LordPe, Procdum to dump files crackme3.exe dumped. exe.

Programs after unpack current still has not run the IAT are already encrypted, so I need to earn how to resolve it. Principle is original, save asprotect will keep the value of the original IAT, then in the process, it will encrypt this IAT values correspond. And in the process called ham, will through 1 ham process of freedom Asprotect restore the original values.

004085A0	55	PUSH EBP
004085A1	8BEC	MOV EBP,ESP
004085A3	83C4 F0	ADD ESP,-10
004085A6	B8 60854000	MOV EAX,crackME.00408560
004085AB	E8 B0C4FFFF	CALL crackME.00404A60

004085B8	EA 00	CALL crackME.00404BC4	JMP to user32.MessageBoxA
004085C3	E8 40FFFFFF	CALL crackME.00408508	
004085C8	E8 C245FFFF	CALL crackME.00402090	

Scroll down below 1 billion, we see the API MessageBoxA, to address 404BC4 to view the content IAT has already flowers.

00404B13	CS	RET	
00404B14	- FF25 C0B14000	JMP DWORD PTR DS:[40B1C0]	
00404B1A	8BC0	MOV EAX,EAX	
00404B1C	- FF25 BCB14000	JMP DWORD PTR DS:[40B1BC]	
00404B22	8BC0	MOV EAX,EAX	
00404B24	- FF25 B8B14000	JMP DWORD PTR DS:[40B1B8]	
00404B2A	8BC0	MOV EAX,EAX	
00404B2C	- FF25 B4B14000	JMP DWORD PTR DS:[40B1B4]	
00404B32	8BC0	MOV EAX,EAX	
00404B34	- FF25 B0B14000	JMP DWORD PTR DS:[40B1B0]	
00404B3A	8BC0	MOV EAX,EAX	
00404B3C	- FF25 ACB14000	JMP DWORD PTR DS:[40B1AC]	
00404B42	8BC0	MOV EAX,EAX	
00404B44	- FF25 A8B14000	JMP DWORD PTR DS:[40B1A8]	
00404B4A	8BC0	MOV EAX,EAX	
00404B4C	- FF25 A4B14000	JMP DWORD PTR DS:[40B1A4]	
00404B52	8BC0	MOV EAX,EAX	
00404B54	- FF25 A0B14000	JMP DWORD PTR DS:[40B1A0]	
00404B5A	8BC0	MOV EAX,EAX	
00404B5C	- FF25 9CB14000	JMP DWORD PTR DS:[40B19C]	
00404B62	8BC0	MOV EAX,EAX	
00404B64	- FF25 98B14000	JMP DWORD PTR DS:[40B198]	
00404B6A	8BC0	MOV EAX,EAX	
00404B6C	- FF25 94B14000	JMP DWORD PTR DS:[40B194]	
00404B72	8BC0	MOV EAX,EAX	
00404B74	- FF25 90B14000	JMP DWORD PTR DS:[40B190]	
00404B7A	8BC0	MOV EAX,EAX	
00404B7C	- FF25 8CB14000	JMP DWORD PTR DS:[40B18C]	
00404B82	8BC0	MOV EAX,EAX	
00404B84	- FF25 88B14000	JMP DWORD PTR DS:[40B188]	
00404B8A	8BC0	MOV EAX,EAX	
00404B8C	- FF25 84B14000	JMP DWORD PTR DS:[40B184]	
00404B92	8BC0	MOV EAX,EAX	
00404B94	- FF25 80B14000	JMP DWORD PTR DS:[40B180]	
00404B9A	8BC0	MOV EAX,EAX	
00404B9C	- FF25 7CB14000	JMP DWORD PTR DS:[40B17C]	
00404BA2	8BC0	MOV EAX,EAX	
00404BA4	- FF25 D4B14000	JMP DWORD PTR DS:[40B1D4]	user32.CharNextA
00404BAA	8BC0	MOV EAX,EAX	
00404BAC	- FF25 D8B14000	JMP DWORD PTR DS:[40B1D8]	user32.CharToOemA
00404BB2	8BC0	MOV EAX,EAX	
00404BB4	- FF25 D0B14000	JMP DWORD PTR DS:[40B1D0]	user32.GetSystemMetrics
00404BBA	8BC0	MOV EAX,EAX	
00404BBC	- FF25 CCB14000	JMP DWORD PTR DS:[40B1CC]	user32.LoadStringA
00404BC2	8BC0	MOV EAX,EAX	
00404BC4	- FF25 C8B14000	JMP DWORD PTR DS:[40B1C8]	user32.MessageBoxA
00404BCA	8BC0	MOV EAX,EAX	

hm .. we need to find the location of ham IAT encryption. To do that, we need to determine where values, as well as the length of the IAT. Use 1 xiu craftsmanship (please read the post about 1:23 RC4 asprotect old) we find

Beginning: 040B0A0


```
0040B0A0 00 00 AF 00 00 00 B1 00
0040B0A8 00 00 B2 00 00 00 B3 00
0040B0B0 00 00 B4 00 00 00 B5 00
0040B0B8 00 00 B6 00 00 00 B7 00
0040B0C0 00 00 B8 00 00 00 B9 00
```

End: 040B1D8

```
0040B1C8 F0 11 D3 77 8C F2 D1 77
0040B1D0 80 5E D1 77 42 27 D2 77
0040B1D8 83 F1 D1 77 48 D0 3A DF
```

Length: 138

3. IAT solving encrypted:

Ctrl-F2 load up the program, establish parameters of OlIt as follows (check out ignore)

☒ Ignore memory access violations in KERNEL32

Ignore (pass to program) following exceptions:

- ☒ INT3 breaks
- ☒ Single-step break
- ☒ Memory access violation

Now Set hardware breakpoint in the region through the first IAT = 40B0A0, and set breakpoint on memory access in 40B0A0. The reason for .. This is because in fact that, if only 1 set of 2 test program will stop any point that we need both. Then press F9 to run.

```
0092CA6E 8B12 MOV EDX,DWORD PTR DS:[EDX]
0092CA70 8902 MOV DWORD PTR DS:[EDX],EAX
0092CA72 5C EB 5C JMP SHORT 0092CAD0
```

I stop here, at the current location, we have not seen EAX value of element in the IAT has been that and are prepared to be assigned to the region through the IAT.

```
0092CA25 8BCF MOV ECX,EDI
0092CA27 8D95 FCFFFFFF LEA EDX,DWORD PTR SS:[EBP-104]
0092CA2D 92 XCHG EAX,EDX
0092CA2E E8 4595FEFF CALL 00915F78
0092CA33 6A 0A PUSH 0A
0092CA35 8D4E 1A LEA ECX,DWORD PTR DS:[ESI+1A]
0092CA38 8BD1 MOV EDX,EDI
0092CA3A 8D85 FCFFFFFF LEA EAX,DWORD PTR SS:[EBP-104]
0092CA40 E8 8B4BFFFF CALL 00921500
0092CA45 8D85 FCFFFFFF LEA EAX,DWORD PTR SS:[EBP-104]
0092CA4B 50 PUSH EAX
0092CA4C 8B45 10 MOV EAX,DWORD PTR SS:[EBP+10]
0092CA4F 50 PUSH EAX
0092CA50 56 PUSH ESI
0092CA51 E8 8EFDFFFF CALL 0092C7E4
0092CA56 8BD8 MOV EBX,EAX
0092CA58 6A 00 PUSH 0
0092CA5A 68 08BC9200 PUSH 92BC08
0092CA5F 8D4D FC LEA ECX,DWORD PTR SS:[EBP-4]
0092CA62 8BD3 MOV EDX,EBX
0092CA64 8BC6 MOV EAX,ESI
0092CA66 E8 01F9FFFF CALL 0092C360
0092CA6B 8B55 0C MOV EDX,DWORD PTR SS:[EBP+C]
0092CA6E 8B12 MOV EDX,DWORD PTR DS:[EDX]
0092CA70 8902 MOV DWORD PTR DS:[EDX],EAX
0092CA72 5C EB 5C JMP SHORT 0092CAD0
```

Look at the structure segment order, we find completely disagree with how the flowers have the torng asprotect 1:23 RC4 ---> disappear. But not after, while desperate distress, I look at the window registry info:

```
EAX 004F0000
ECX 00000000
EDX 0040B0A0 crackME.0040B0A0
EBX 77F59A22 ntdll.RtlDeleteCriticalSection
ESP 0012FDE8
EBP 0012FEF8
ESI 009604C0
EDI 00000016
EIP 0092CA70
```

EBX not value your API ---> is no. As a result, at the current location, EAX less value after the encryption of EBX. So we modify the 1,

```

0092CA6E 8B12      MOV EDX, DWORD PTR DS:[EDX]
0092CA70 891A      MOV DWORD PTR DS:[EDX], EBX
0092CA72 5B        JMP SHORT 0092CA75

```

to the value recorded on IAT value is always true.

Now we must return to the heart after encryption is completed, the trace segment 1 is to:

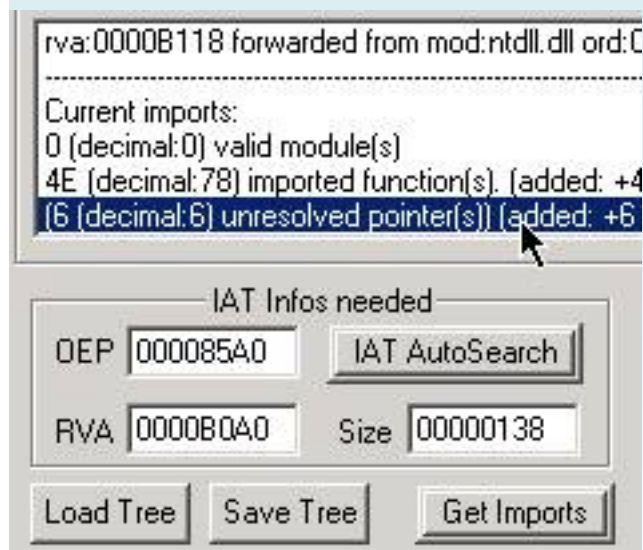
```

0092CB95 E8 36FEFFFF CALL 0092C900
0092CB9A 5B        POP EBX
0092CB9B ^ EB CB    JMP SHORT 0092CB68
0092CB9D 5F        POP EDI
0092CB9E 5E        POP ESI
0092CB9F 5B        POP EBX
0092CBA0 5D        POP EDX

```

Set breakpoint here and press F9 to running the program. After the stop at breakpoint already set, we dump Imprec to restore IAT.

Imprec load up, fill in value in the heart



Human GetImport to IAT. Then Fix the dump, select File dumped.exe to complete the process.

Armadillo collect sand-stone

AutoPlay Media Studio 6.0 <|> ARM 4.xx - Standard Protection + IAT elimination



1. Intro

_Cai This soft, I buy new is down, the USB each time I was online I clean it occupies 512MB disk for a movie so afraid J soft from heavy 20MB up! See also soft happy meat it should be fun to watch her children play! Now, a new professional updatesoft bit bõn contains soft down and release the AllInOne make their product. Mk, I hate to some form of this warez group: D. Only to discuss this in more than hot: P. You only unpack this soft!

2. Tools

- 1.OllyDBG - The best config debugger for ArmMUP by hacnho.
- 2.LordPE 1.4 Deluxe
- 3.Import REConstructor 1.6 Final

3. Unpacking

_Load Target:

Address	Hex	dump	Disassembly	Comment
00CA26D3	55		PUSH EBP	
00CA26D4	8BEC		MOV EBP,ESP	
00CA26D6	6A FF		PUSH -1	
00CA26D8	68 88C1CC00		PUSH ams60.00CCC188	
00CA26DD	68 1024CA00		PUSH ams60.00CA2410	
00CA26E2	64:A1 00000000		MOV EAX,DWORD PTR FS:[0]	SE handler installation
00CA26E8	50		PUSH EAX	
00CA26E9	64:8925 00000000		MOV DWORD PTR FS:[0],ESP	
00CA26F0	83EC 58		SUB ESP,58	
00CA26F3	53		PUSH EBX	
00CA26F4	56		PUSH ESI	ams60.00570718
00CA26F5	57		PUSH EDI	ntdll.77F5164E
00CA26F6	8965 E8		MOV DWORD PTR SS:[EBP-18],ESP	
00CA26F9	FF15 8851CC00		CALL DWORD PTR DS:[<&KERNEL32.GetVersion>]	kernel32.GetVersion
00CA26FF	33D2		XOR EDX,EDX	
00CA2701	8AD4		MOV DL,AH	
00CA2703	8915 14D8CC00		MOV DWORD PTR DS:[CCD814],EDX	
00CA2709	8BC8		MOV ECX,EAX	
00CA270B	81E1 FF000000		AND ECX,0FF	
00CA2711	890D 10D8CC00		MOV DWORD PTR DS:[CCD810],ECX	
00CA2717	C1E1 08		SHL ECX,8	
00CA271A	03CA		ADD ECX,EDX	
00CA271C	890D 0CD8CC00		MOV DWORD PTR DS:[CCD80C],ECX	
00CA2722	C1E8 10		SHR EAX,10	
00CA2725	A3 08D8CC00		MOV DWORD PTR DS:[CCD808],EAX	
00CA272A	33F6		XOR ESI,ESI	ams60.00570718
00CA272C	56		PUSH ESI	ams60.00570718
00CA272D	E8 78160000		CALL ams60.00CA3DAA	
00CA2732	59		POP ECX	kernel32.77E814C7
00CA2733	85C0		TEST EAX,EAX	
00CA2735	75 00		JNZ SHORT ams60.00CA2735	

_Bp CreateThread, F9, Ctrl + F9, F8:

Address	Value	Comment
0012F6FC	016EC025	CALL to CreateThread from 016EC01F
0012F700	00000000	pSecurity = NULL
0012F704	00000000	StackSize = 0
0012F708	016EC8E3	ThreadFunction = 016EC8E3
0012F70C	00000000	pThreadParm = NULL
0012F710	00000000	CreationFlags = 0
0012F714	0012F720	pThreadId = 0012F720
0012F718	77F5164E	RETURN to ntdll.77F5164E from ntdll.77F5164E
0012F71C	00CCD098	ams60.00CCD098
0012F720	00000001	
0012F724	0012F73C	
0012F728	0170036A	RETURN to 0170036A from 016EBF7D
0012F72C	00000000	

Address	Hex dump	Disassembly	Comment
016EC025	5F	POP EDI	ntdll.77F5164E
016EC026	5E	POP ESI	ntdll.77F5164E
016EC027	C9	LEAVE	
016EC028	C3	RETN	
016EC029	55	PUSH EBP	
016EC02A	8BEC	MOV EBP,ESP	
016EC02C	81EC 28010000	SUB ESP,128	
016EC032	56	PUSH ESI	
016EC033	57	PUSH EDI	
016EC034	BE E4BD7001	MOV ESI,170BDE4	ASCII "MainClass"
016EC039	8DBD D8FEFFFF	LEA EDI,DWORD PTR SS:[EBP-128]	
016EC03F	A5	MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]	
016EC040	A5	MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]	
016EC041	66:A5	MOVS WORD PTR ES:[EDI],WORD PTR DS:[ESI]	
016EC043	6A 3D	PUSH 3D	
016EC045	33C0	XOR EAX,EAX	
016EC047	59	POP ECX	ntdll.77F5164E
016EC048	8DBD E2FEFFFF	LEA EDI,DWORD PTR SS:[EBP-11E]	
016EC04E	F3:AB	REP STOS DWORD PTR ES:[EDI]	
016EC050	66:AB	STOS WORD PTR ES:[EDI]	
016EC052	A1 2C567101	MOV EAX,DWORD PTR DS:[171562C]	
016EC057	33FF	XOR EDI,EDI	
016EC059	3BC7	CMP EAX,EDI	
016EC05B	74 20	JE SHORT 016EC07D	
016EC05D	50	PUSH EAX	
016EC05E	68 F8AA7001	PUSH 170AAF8	ASCII "08X"
016EC063	8DBD D8FEFFFF	LEA EAX,DWORD PTR SS:[EBP-128]	
016EC069	57	PUSH EDI	
016EC06A	50	PUSH EAX	
016EC06B	FF15 10637001	CALL DWORD PTR DS:[1706310]	msvcrt.strochr
016EC071	59	POP ECX	ntdll.77F5164E

Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
0170036A	59	POP ECX	kernel32.77E7BE2B
0170036B	BF 580A7101	MOV EDI,1710A58	
01700370	8BCF	MOV ECX,EDI	ntdll.77F5164E
01700372	E8 B77DFDFF	CALL 016D812E	
01700377	84C0	TEST AL,AL	
01700379	75 09	JNZ SHORT 01700384	
0170037B	6A 01	PUSH 1	
0170037D	8BCF	MOV ECX,EDI	ntdll.77F5164E
0170037F	E8 93D3FDFF	CALL 016DD717	
01700384	B9 C0FB7001	MOV ECX,170FBC0	
01700389	C705 30C27001 C0DE7001	MOV DWORD PTR DS:[170C230],170DEC0	
01700393	E8 3EF2FFFF	CALL 016FF5D6	
01700398	6A 00	PUSH 0	
0170039A	E8 37F2FFFF	CALL 016FF5D6	
0170039F	59	POP ECX	kernel32.77E7BE2B
017003A0	33C9	XOR ECX,ECX	kernel32.77E7BE2B
017003A2	380D 7C107101	CMP BYTE PTR DS:[171107C],CL	
017003A8	75 36	JNZ SHORT 017003E0	
017003AA	A1 A4107101	MOV EAX,DWORD PTR DS:[17110A4]	
017003AF	53	PUSH EBX	
017003B0	8B48 08	MOV ECX,DWORD PTR DS:[EAX+8]	
017003B3	894D 08	MOV DWORD PTR SS:[EBP+8],ECX	kernel32.77E7BE2B
017003B6	8B78 44	MOV EDI,DWORD PTR DS:[EAX+44]	
017003B9	3378 3C	XOR EDI,DWORD PTR DS:[EAX+3C]	
017003BC	8B58 70	MOV EBX,DWORD PTR DS:[EAX+70]	
017003BF	3358 58	XOR EBX,DWORD PTR DS:[EAX+58]	
017003C2	8D4D 08	LEA ECX,DWORD PTR SS:[EBP+8]	
017003C5	3378 34	XOR EDI,DWORD PTR DS:[EAX+34]	
017003C8	3358 2C	XOR EBX,DWORD PTR DS:[EAX+2C]	
017003CB	033D BC107101	ADD EDI,DWORD PTR DS:[17110BC]	ams60.00400000
017003D1	59	POP ECX	

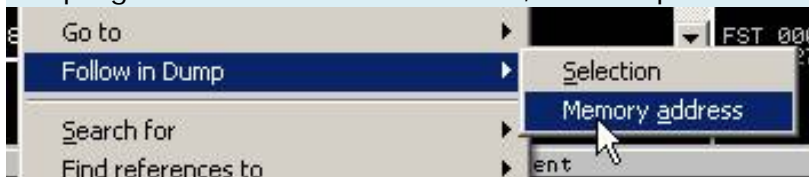
Cuon Down a bit:

Address	Hex dump	Disassembly	Comment
01700409	85D2	TEST EDX,EDX	
0170040B	75 18	JNZ SHORT 01700425	
0170040D	8B50 50	MOV EDX,DWORD PTR DS:[EAX+50]	
01700410	FF76 18	PUSH DWORD PTR DS:[ESI+18]	
01700413	3350 2C	XOR EDX,DWORD PTR DS:[EAX+2C]	
01700416	FF76 14	PUSH DWORD PTR DS:[ESI+14]	
01700419	3350 04	XOR EDX,DWORD PTR DS:[EAX+4]	
0170041C	FF76 10	PUSH DWORD PTR DS:[ESI+10]	
0170041F	2BCA	SUB ECX,EDX	
01700421	FFD1	CALL ECX	kernel32.77E7BE2B
01700423	EB 10	JMP SHORT 01700442	
01700425	83FA 01	CMP EDX,1	
01700428	75 1B	JNZ SHORT 01700445	
0170042A	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
0170042D	8B50 50	MOV EDX,DWORD PTR DS:[EAX+50]	
01700430	3350 2C	XOR EDX,DWORD PTR DS:[EAX+2C]	
01700433	FF76 08	PUSH DWORD PTR DS:[ESI+8]	
01700436	3350 04	XOR EDX,DWORD PTR DS:[EAX+4]	
01700439	6A 00	PUSH 0	
0170043B	FF76 0C	PUSH DWORD PTR DS:[ESI+C]	ams60.00400000
0170043E	2BCA	SUB ECX,EDX	
01700440	FFD1	CALL ECX	kernel32.77E7BE2B
01700442	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
01700445	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
01700448	5F	POP EDI	ntdll.77F5164E
01700449	5E	POP ESI	ams60.00CCD098
0170044A	C9	LEAVE	
0170044B	C3	RETN	
0170044C	837C24 08 01	CMP DWORD PTR SS:[ESP+8],1	
01700451	75 14	JNZ SHORT 01700467	
01700453	68 500C7101	JMP 01710C58	

Dat Call ECX breakpoint in the second, F9, F7 OEP: J

Address	Hex dump	Disassembly	Comment
0076D84D	55	PUSH EBP	
0076D84E	8BEC	MOV EBP,ESP	
0076D850	6A FF	PUSH -1	
0076D852	68 28C29C00	PUSH ams60.009CC228	
0076D857	68 685F7700	PUSH ams60.00775F68	
0076D85C	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
0076D862	50	PUSH EAX	ams60.00CC560C
0076D863	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
0076D86A	83EC 58	SUB ESP,58	
0076D86D	53	PUSH EBX	
0076D86E	56	PUSH ESI	ams60.00CCD098
0076D86F	57	PUSH EDI	ams60.00401000
0076D870	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0076D873	FF15 D89DB200	CALL DWORD PTR DS:[B29DD8]	
0076D879	33D2	XOR EDX,EDX	
0076D87B	8AD4	MOV DL,AH	
0076D87D	8915 C0EFB000	MOV DWORD PTR DS:[B0EFC0],EDX	
0076D883	8BC8	MOV ECX,EAX	ams60.00CC560C
0076D885	81E1 FF000000	AND ECX,0FF	
0076D88B	890D BCEFB000	MOV DWORD PTR DS:[B0EFB0],ECX	ams60.0076D84D
0076D891	C1E1 08	SHL ECX,8	
0076D894	03CA	ADD ECX,EDX	
0076D896	890D B8EFB000	MOV DWORD PTR DS:[B0EFB8],ECX	ams60.0076D84D
0076D89C	C1E8 10	SHR EAX,10	
0076D89F	A3 B4EFB000	MOV DWORD PTR DS:[B0EFB4],EAX	ams60.00CC560C
0076D8A4	6A 01	PUSH 1	
0076D8A6	E8 486D0000	CALL ams60.007745F3	
0076D8AB	59	POP ECX	01700442
0076D8AC	85C0	TEST EAX,EAX	ams60.00CC560C
0076D8AE	75 08	JNZ SHORT ams60.0076D8B8	
0076D8B0	6A 1C	PUSH 1C	

Dom D89DB200 line 0076D873 FF15 CALL DWORD PTR DS: [B29DD8] we now bit IAT elimination J statistics. Wait ti resolve it later, we need to patch magic jump to avoid the damage of the IAT is, to cause the program to crash! At 0076D873, we dump Follow Print> Memory Address:



Trong Window dump Window:

Address	Value	Comment
00B29008	016EAD60	
00B2900C	77F51502	ntdll.RtlGetLastWin32Error
00B290E0	77E75DEC	kernel32.lstrcmpA
00B290E4	77E7A0A9	kernel32.GetModuleFileNameA
00B290E8	77E7A949	kernel32.WideCharToMultiByte
00B290EC	77E7A7FC	kernel32.MultiByteToWideChar
00B290F0	77E760E1	kernel32.lstrlenA
00B290F4	77E7DDAF	kernel32.GetVersionExA
00B290F8	77E712DB	kernel32.GetTempPathA
00B290FC	016E6E52	
00B29E00	00000000	
00B29E04	00000000	
00B29E08	00000000	

_Cuon Up for IAT start:

Address	Value	Comment
00B29654	00000000	
00B29658	00000000	
00B2965C	77DD229A	ADVAPI32.RegOpenKeyExA
00B29660	77DE6504	ADVAPI32.RegDeleteValueA
00B29664	77DE68E2	ADVAPI32.RegDeleteKeyA
00B29668	77DE0820	ADVAPI32.RegGetKeySecurity
00B2966C	77DE0CB8	ADVAPI32.RegSetKeySecurity
00B29670	77DE6960	ADVAPI32.RegFlushKey
00B29674	77DD2410	ADVAPI32.RegQueryValueExA
00B29678	77DE63B1	ADVAPI32.RegSetValueExA
00B2967C	77DE6CF4	ADVAPI32.RegQueryInfoKeyA
00B29680	77DD3111	ADVAPI32.RegNotifyChangeKeyValue
00B29684	77DE6F8B	ADVAPI32.RegEnumKeyExA

_Cuon Down for IAT End:

Address	Value	Comment
00B2AACC	00000000	
00B2AAD0	00000000	
00B2AAD4	00000000	
00B2AAD8	00000000	
00B2AADC	00000000	
00B2AAE0	00000000	
00B2AAE4	761339CB	urlmon.URLDownloadToFileA
00B2AAE8	016E6F60	
00B2AAEC	00000000	
00B2AAF0	00000000	
00B2AAF4	00000000	
00B2AAF8	00000000	
00B2AAFC	00000000	

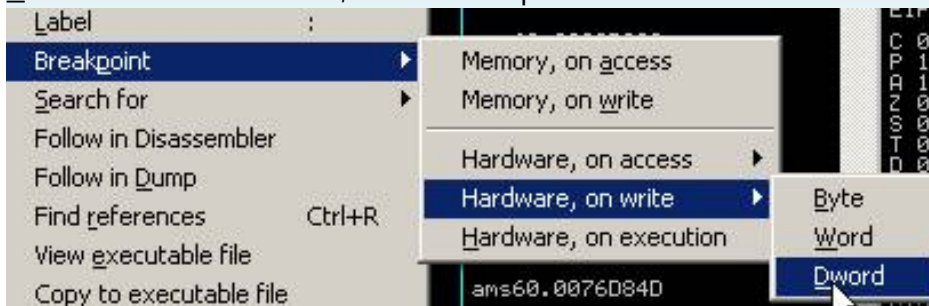
_Tom Again we have basic information:

IAT Start: 00B2965C 77DD229A ADVAPI32.RegOpenKeyExA

IAT End: 0B2AAE4 761339CB urlmon.URLDownloadToFileA

IAT Len: 1488

_Ctrl + G to 00B2965C, set a breakpoint on write:



_Ctrl Restart + F2, Shift + F9:

Address	Hex dump	Disassembly	Comment
77C42F43	F3: A5	REP MOVSD DWORD PTR DS:[EDI],DWORD PTR DS:[EAX]	
77C42F45	FF2495 5830C477	JMP DWORD PTR DS:[EDX*4+77C43058]	msvort.77C43068
77C42F4C	8BC7	MOV EAX,EDI	ams60.00B29660
77C42F4E	BA 03000000	MOV EDI,3	
77C42F53	83E9 04	SUB ECX,4	
77C42F56	72 0C	JB SHORT msvort.77C42F64	
77C42F58	83E0 03	AND EAX,3	
77C42F5B	03C8	ADD ECX,EAX	
77C42F5D	FF2485 702EC477	JMP DWORD PTR DS:[EAX*4+77C42F70]	
77C42F64	FF248D 6830C477	JMP DWORD PTR DS:[ECX*4+77C43068]	
77C42F68	90	NOP	
77C42F6C	FF248D EC2EC477	JMP DWORD PTR DS:[ECX*4+77C42FEC]	
77C42F73	90	NOP	
77C42F74	802E C4	SUB BYTE PTR DS:[EDI],0C4	
77C42F77	77 AC	JA SHORT msvort.77C42F25	
77C42F79	2E	DAS	
77C42F7A	C477 D0	LES ESI,FWORD PTR DS:[EDI-80]	Modification of segment reg
77C42F7D	2E	DAS	
77C42F7E	C477 23	LES ESI,FWORD PTR DS:[EDI+23]	Modification of segment reg
77C42F81	D18A 0688078A	ROR DWORD PTR DS:[EDX+8A078806],1	
77C42F87	46	INC ESI	
77C42F88	0188 47018A46	ADD DWORD PTR DS:[EAX+468A0147],ECX	
77C42F8E	02C1	ADD AL,CL	
77C42F90	E9 02884702	JMP 7A0BB797	
77C42F95	83C6 03	ADD ESI,3	
77C42F98	83C7 03	ADD EDI,3	
77C42F9B	83F9 08	CMP ECX,8	
77C42F9E	72 CC	JB SHORT msvort.77C42F6C	
77C42FA0	F3: A5	REP MOVSD DWORD PTR DS:[EDI],DWORD PTR DS:[EAX]	
77C42FA2	FF2495 5830C477	JMP DWORD PTR DS:[EDX*4+77C43058]	msvort.77C43068
77C42FA3	8048 80	LEA ECX,DWORD PTR DS:[ECX]	

_Shift + F9 times 2:

Address	Hex dump	Disassembly	Comment
016FCCES	8B85 1009FFFF	MOV EAX,DWORD PTR SS:[EBP-26F0]	ams60.00B2965C
016FCCFB	83C0 04	ADD EAX,4	
016FCCFE	9B85 1009FFFF	MOV DWORD PTR SS:[EBP-26F0],EAX	ams60.00B2965C
016FCCFF	E9 4DFCFFFF	JMP 016FC946	
016FCCF9	FF15 78627001	CALL DWORD PTR DS:[1706278]	kernel32.GetTickCount
016FCCFF	2B85 A4D4FFFF	SUB EAX,DWORD PTR SS:[EBP-2B5C]	ams60.00B2E10
016FCD05	8B8D A8D4FFFF	MOV ECX,DWORD PTR SS:[EBP-2B58]	
016FCD0B	6BC9 32	IMUL ECX,ECX,32	ADVAPI32.RegOpenKeyExA
016FCD0E	81C1 D0070000	ADD ECX,7D0	
016FCD14	3BC1	CMP EAX,ECX	ADVAPI32.RegOpenKeyExA
016FCD16	76 28	JBE SHORT 016FCD40	
016FCD18	C685 34D9FFFF 01	MOV BYTE PTR SS:[EBP-26CC],1	
016FCD1F	6A 01	PUSH 1	
016FCD21	58	POP EAX	01710A58
016FCD22	85C0	TEST EAX,EAX	ams60.00B2965C
016FCD24	74 09	JE SHORT 016FCD2F	
016FCD26	83A5 6CA7FFFF 00	AND DWORD PTR SS:[EBP+FFFA76C],0	
016FCD2D	EB 11	JMP SHORT 016FCD40	
016FCD2F	68 24D07001	PUSH 170D024	ASCII " ** IR2"
016FCD34	E8 4E32FFFF	CALL 016EFF87	
016FCD39	59	POP ECX	01710A58
016FCD3A	8B85 6CA7FFFF	MOV DWORD PTR SS:[EBP+FFFA76C],EAX	ams60.00B2965C
016FCD40	83BD E4D7FFFF 00	CMP DWORD PTR SS:[EBP-281C],0	
016FCD47	0F85 8A000000	JNZ 016FCDD7	
016FCD4D	0FB685 94D4FFFF	MOVZX EAX,BYTE PTR SS:[EBP-2B6C]	
016FCD54	85C0	TEST EAX,EAX	ams60.00B2965C
016FCD56	74 7F	JE SHORT 016FCDD7	
016FCD58	6A 00	PUSH 0	
016FCD5A	8B85 98D4FFFF	MOV EAX,DWORD PTR SS:[EBP-2B68]	
016FCD60	C1E0 02	SHL EAX,2	
016FCD62	58	POP EAX	ams60.00B2965C

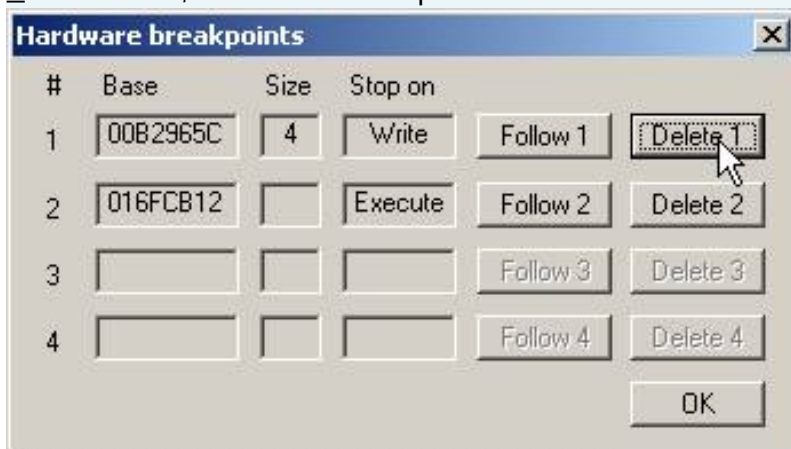
_Cuon Up above a little:

Address	Hex dump	Disassembly	Comment
016FCAEC	8985 58C2FFFF	MOV DWORD PTR SS:[EBP-3DA8],EAX	ams60.00B2965C
016FCAF2	8B85 58C2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DA8]	
016FCAF8	8378 08 00	CMP DWORD PTR DS:[EAX+8],0	
016FC AFC	74 49	JE SHORT 016FCB47	
016FCAFE	68 00010000	PUSH 100	
016FCB03	8D85 58C1FFFF	LEA EAX,DWORD PTR SS:[EBP-3EA8]	
016FCB09	50	PUSH EAX	ams60.00B2965C
016FCB0A	8B85 58C2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DA8]	
016FCB10	FF30	PUSH DWORD PTR DS:[EAX]	ADVAPI32.RegOpenKeyExA
016FCB12	E8 C854FDFF	CALL 016D1FDF	
016FCB17	83C4 0C	ADD ESP,0C	
016FCB1A	8D85 58C1FFFF	LEA EAX,DWORD PTR SS:[EBP-3EA8]	
016FCB20	50	PUSH EAX	ams60.00B2965C
016FCB21	8D85 68C2FFFF	LEA EAX,DWORD PTR SS:[EBP-3D98]	
016FCB27	50	PUSH EAX	ams60.00B2965C
016FCB28	FF15 88637001	CALL DWORD PTR DS:[70018863]	msvcrt.stricmp
016FCB2E	59	POP ECX	01710A58
016FCB2F	59	POP ECX	01710A58
016FCB30	85C0	TEST EAX,EAX	ams60.00B2965C
016FCB32	75 11	JNZ SHORT 016FCB45	
016FCB34	8B85 58C2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DA8]	
016FCB3A	8B40 08	MOV EAX,DWORD PTR DS:[EAX+8]	ams60.0072D36C
016FCB3D	8985 68CAFFFF	MOV DWORD PTR SS:[EBP-3598],EAX	ams60.00B2965C
016FCB43	EB 02	JMP SHORT 016FCB47	
016FCB45	EB 9C	JMP SHORT 016FCAE3	
016FCB47	8B85 A8D4FFFF	MOV EAX,DWORD PTR SS:[EBP-2B58]	
016FCB4D	40	INC EAX	ams60.00B2965C
016FCB4E	8985 A8D4FFFF	MOV DWORD PTR SS:[EBP-2B58],EAX	ams60.00B2965C
016FCB54	EB 37	JMP SHORT 016FCB8D	
016FCB56	8D8D 38D9FFFF	LEA ECX,DWORD PTR SS:[EBP-26C8]	
016FCB5C	58 D5445B55	CALL 016D1840	

_Ghi Memory address functions Magic Call: **016FCB12 E8 C854FDFF CALL 016D1FDF**, Ctrl + F2 to restart, Shift + F9. Ctrl + G: 016FCB12, to here:

Address	Hex dump	Disassembly	Comment
016FCB12	93	XOR EAX,EBX	
016FCB13	6261 5F	BOUND ESP,QWORD PTR DS:[ECX+5F]	
016FCB16	31B7 D6E6CF3A	XOR DWORD PTR DS:[EDI+3ACFE606],ESI	
016FCB1C	4E	DEC ESI	
016FCB1D	C7	???	Unknown command
016FCB1E	A5	MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]	
016FCB1F	E9 08EE8D9F	JMP 80FDB92C	
016FCB24	E6 C4	OUT 0C4,AL	I/O command
016FCB26	01FD	ADD EBP,EDI	ams60.00B29660
016FCB28	7A 3E	JPE SHORT 016FCB68	
016FCB2A	3100	XOR DWORD PTR DS:[EAX],EAX	
016FCB2C	C7	???	Unknown command
016FCB2D	6D	INS DWORD PTR ES:[EDI],DX	I/O command
016FCB2E	58	POP EAX	
016FCB2F	E8 05B0AEB6	CALL B81E7B39	
016FCB34	B9 B0881640	MOV ECX,401688B0	
016FCB39	DF30	FBSTP TBYTE PTR DS:[EAX]	
016FCB3B	332B	XOR EBP,DWORD PTR DS:[EBX]	
016FCB3D	6A E3	PUSH -1D	
016FCB3F	E7 C6	OUT 0C6,EAX	I/O command
016FCB41	37	AAA	
016FCB42	AD	LODS DWORD PTR DS:[ESI]	
016FCB43	B6 5D	MOV DH,5D	
016FCB45	6A 31	PUSH 31	
016FCB47	E1 8A	LOOPE SHORT 016FCAD3	
016FCB49	C106 8D	ROL DWORD PTR DS:[ESI],8D	Shift constant out of range
016FCB4C	0D A18E67C2	OR EAX,C2678EA1	
016FCB51	95	XCHG EAX,EBP	
016FCB52	D813	FCOM DWORD PTR DS:[EBX]	
016FCB54	3F	AAS	
016FCB55	65	OUTS DX,BYTE PTR ES:[EDI]	I/O command

_Dat Here a, removed breakpoint on Write:



_F9, We will stop at Magic Call functions:

Address	Hex dump	Disassembly	Comment
016FCB12	E8 C854FDFF	CALL 016D1FDF	
016FCB17	83C4 0C	ADD ESP,0C	
016FCB1A	8D85 58C1FFFF	LEA EAX,DWORD PTR SS:[EBP-3EA8]	
016FCB20	50	PUSH EAX	
016FCB21	8D85 68C2FFFF	LEA EAX,DWORD PTR SS:[EBP-3D98]	
016FCB27	50	PUSH EAX	
016FCB28	FF15 88637001	CALL DWORD PTR DS:[1706388]	msvcrt._strcmp
016FCB2E	59	POP ECX	0170BA44
016FCB2F	59	POP ECX	0170BA44
016FCB30	85C0	TEST EAX,EAX	
016FCB32	75 11	JNZ SHORT 016FCB45	
016FCB34	8B85 58C2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DA8]	
016FCB3A	8B40 08	MOV EAX,DWORD PTR DS:[EAX+8]	
016FCB3D	8985 68CAFFFF	MOV DWORD PTR SS:[EBP-3598],EAX	
016FCB43	EB 02	JMP SHORT 016FCB47	
016FCB45	EB 9C	JMP SHORT 016FCAE3	
016FCB47	8B85 A8D4FFFF	MOV EAX,DWORD PTR SS:[EBP-2B58]	
016FCB4D	40	INC EAX	
016FCB4E	8985 A8D4FFFF	MOV DWORD PTR SS:[EBP-2B58],EAX	
016FCB54	EB 37	JMP SHORT 016FCB8D	
016FCB56	8D8D 38D9FFFF	LEA ECX,DWORD PTR SS:[EBP-26C8]	
016FCB5C	E8 DF44FDFF	CALL 016D1040	
016FCB61	0FB6C0	MOVZX EAX,AL	
016FCB64	99	CDB	
016FCB65	6A 14	PUSH 14	
016FCB67	59	POP ECX	0170BA44
016FCB68	F7F9	IDIV ECX	
016FCB6A	8B85 10D9FFFF	MOV EAX,DWORD PTR SS:[EBP-26F0]	ams60.00B29AC8
016FCB70	8B8C95 94D7FFFF	MOV ECX,DWORD PTR SS:[EBP+EDX*4-286C]	
016FCB77	8908	MOV DWORD PTR DS:[EAX],ECX	
016FCB79	8B85 10D9FFFF	MOV EAX,DWORD PTR SS:[EBP-26F0]	ams60.00B29AC8

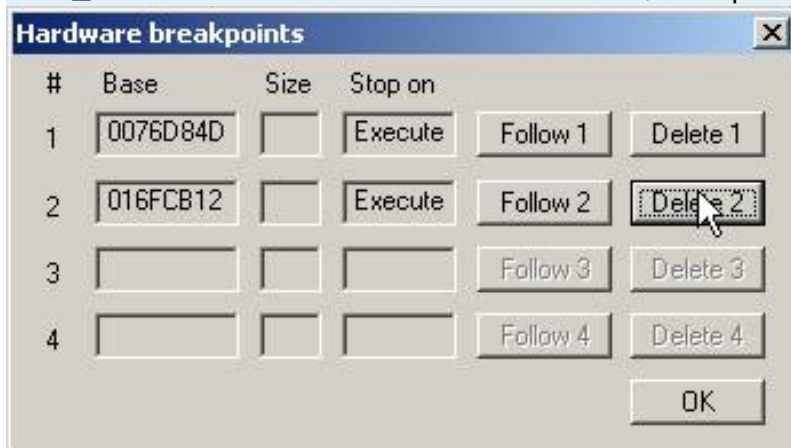
_Enter To call this function:

Address	Hex dump	Disassembly	Comment
016D1FAF	75 F3	JNZ SHORT 016D1FA4	
016D1FB1	5F	POP EDI	0170BA44
016D1FB2	5E	POP ESI	0170BA44
016D1FB3	5B	POP EBX	0170BA44
016D1FB4	C3	RETN	
016D1FB5	E8 05000000	CALL 016D1FBF	
016D1FBA	E9 0A000000	JMP 016D1FC9	
016D1FBF	B9 C0FB7001	MOV ECX,170FBC0	
016D1FC4	E9 90000000	JMP 016D2059	
016D1FC9	68 D51F6D01	PUSH 16D1FD5	
016D1FCE	E8 D8390300	CALL 017059AB	
016D1FD3	59	POP ECX	0170BA44
016D1FD4	C3	RETN	
016D1FD5	B9 C0FB7001	MOV ECX,170FBC0	
016D1FDA	E9 F7D50200	JMP 016FF5D6	
016D1FDF	55	PUSH EBP	
016D1FE0	8BEC	MOV EBP,ESP	
016D1FE2	51	PUSH ECX	
016D1FE3	A1 F0FB7001	MOV EAX,DWORD PTR DS:[170FBF0]	
016D1FE8	53	PUSH EBX	
016D1FE9	56	PUSH ESI	
016D1FEA	57	PUSH EDI	
016D1FEB	85C0	TEST EAX,EAX	
016D1FED	75 37	JNZ SHORT 016D2026	
016D1FEF	68 00010000	PUSH 100	
016D1FF4	C745 FC 029D6798	MOV DWORD PTR SS:[EBP-4],98679D02	
016D1FFB	E8 30390300	CALL 01705930	JMP to msvcrt.??20YAPAXI02
016D2000	8DB0 00010000	LEA ESI,DWORD PTR DS:[EAX+100]	
016D2006	59	POP ECX	0170BA44
016D2007	3BC6	CMP EAX,ESI	
016D2009	02 50552001	MOV DWORD PTR DS:[17055501],EAX	

_Patch To C3:

Address	Hex dump	Disassembly	Comment
016D1FAF	75 F3	JNZ SHORT 016D1FA4	
016D1FB1	5F	POP EDI	0170BA44
016D1FB2	5E	POP ESI	0170BA44
016D1FB3	5B	POP EBX	0170BA44
016D1FB4	C3	RETN	
016D1FB5	E8 05000000	CALL 016D1FBF	
016D1FBA	E9 0A000000	JMP 016D1FC9	
016D1FBF	B9 C0FB7001	MOV ECX,170FBC0	
016D1FC4	E9 90000000	JMP 016D2059	
016D1FC9	68 D51F6D01	PUSH 16D1FD5	
016D1FCE	E8 D8390300	CALL 017059AB	
016D1FD3	59	POP ECX	0170BA44
016D1FD4	C3	RETN	
016D1FD5	B9 C0FB7001	MOV ECX,170FBC0	
016D1FDA	E9 F7D50200	JMP 016FF5D6	
016D1FDF	83	RETN	
016D1FE0	8BEC	MOV EBP,ESP	
016D1FE2	51	PUSH ECX	
016D1FE3	A1 F0FB7001	MOV EAX,DWORD PTR DS:[170FBC0]	
016D1FE8	53	PUSH EBX	
016D1FE9	56	PUSH ESI	
016D1FEA	57	PUSH EDI	
016D1FEB	85C0	TEST EAX,EAX	
016D1FED	75 37	JNZ SHORT 016D2026	
016D1FEF	68 00010000	PUSH 100	
016D1FF4	C745 FC 029D6798	MOV DWORD PTR SS:[EBP-4],98679D02	
016D1FFB	E8 30390300	CALL 01705930	JMP to msvert.??2@YAPAXI02
016D2000	8DB0 00010000	LEA ESI,DWORD PTR DS:[EAX+100]	
016D2006	59	POP ECX	0170BA44
016D2007	3BC6	CMP EAX,ESI	
016D2008	03 F0FB7001	MOV DWORD PTR DS:[170FBC0],EAX	

G + _Ctrl to OEP is 0076D84D 55 PUSH EBP, also placed a clear HE Magic's Call:



_F9 We stopped at the OEP:

Address	Hex dump	Disassembly	Comment
0076D84D	55	PUSH EBP	
0076D84E	8BEC	MOV EBP,ESP	
0076D850	6A FF	PUSH -1	
0076D852	68 28C29C00	PUSH ams60.009CC228	
0076D857	68 685F7700	PUSH ams60.00775F68	
0076D85C	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
0076D862	50	PUSH EAX	ams60.00CC560C
0076D863	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
0076D86A	83EC 58	SUB ESP,58	
0076D86D	53	PUSH EBX	
0076D86E	56	PUSH ESI	ams60.00CCD098
0076D86F	57	PUSH EDI	ams60.00401000
0076D870	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0076D873	FF15 D89DB200	CALL DWORD PTR DS:[B29D08]	kernel32.GetVersion
0076D879	33D2	XOR EDX,EDX	
0076D87B	8AD4	MOV DL,AH	
0076D87D	8915 C0EFB000	MOV DWORD PTR DS:[B0EFC0],EDX	
0076D883	8BC8	MOV ECX,EAX	ams60.00CC560C
0076D885	81E1 FF000000	AND ECX,0FF	ams60.0076D84D
0076D88B	890D BCEFB000	MOV DWORD PTR DS:[B0EFBC],ECX	ams60.0076D84D
0076D891	C1E1 08	SHL ECX,8	ams60.00CC560C
0076D894	03CA	ADD ECX,EDX	ams60.0076D84D
0076D896	890D B8EFB000	MOV DWORD PTR DS:[B0EFB8],ECX	ams60.00CC560C
0076D89C	C1E8 10	SHR EAX,10	
0076D89F	A3 B4EFB000	MOV DWORD PTR DS:[B0EFB4],EAX	
0076D8A4	6A 01	PUSH 1	
0076D8A6	E8 486D0000	CALL ams60.007745F3	
0076D8AB	59	POP ECX	01700442
0076D8AC	85C0	TEST EAX,EAX	ams60.00CC560C
0076D8AE	75 08	JNZ SHORT ams60.0076D8B8	
0076D8B0	50	PUSH EAX	

_Den Redirect the IAT elimination, we need to collect the necessary information:

PID:

Process	Name	Window	Path
00000084	SOUNDMAN	ALSMTray	C:\WINDOWS\SOUNDMAN.EXE
0000009C	ccApp	Norton AntiVirus	C:\Program Files\Common Files\Symantec
000000FC	ctfmon	TF_FloatingLangBar_WndTitl	C:\WINDOWS\System32\ctfmon.exe
000001FC	smss		\SystemRoot\System32\smss.exe
00000240	csrss		??C:\WINDOWS\system32\csrss.exe
00000258	winlogon	NetDDE Agent	??C:\WINDOWS\system32\winlogon.exe
00000284	services		C:\WINDOWS\system32\services.exe
00000290	lsass		C:\WINDOWS\system32\lsass.exe
000002E0	TSCHelp	C:\Program Files\TechSmith	C:\Program Files\TechSmith\SnagIt 7\TS
00000314	EMEDITOR	te.txt * - EmEditor	C:\Program Files\EmEditor\EMEDITOR.EXE
0000032C	svchost		C:\WINDOWS\system32\svchost.exe
00000364	svchost		C:\WINDOWS\System32\svchost.exe
000003A8	svchost		C:\WINDOWS\System32\svchost.exe
000003D4	svchost		C:\WINDOWS\System32\svchost.exe
00000420	ccEvtMgr		C:\Program Files\Common Files\Symantec
0000042C	ams60		C:\Program Files\AutoPlay Media Studio
00000490	spoolsv		C:\WINDOWS\system32\spoolsv.exe

.text. ADATA:

003E0000	0000C000								
003F0000	00006000								
00400000	00001000	ams60							
00401000	00581000	ams60	.text						
00002000	00000000	ams60	.rdata						
00044000	0000B000	ams60	.data						
00B28000	000E6000	ams60	.idata						
00C0E000	00057000	ams60	.reloc						
00C65000	00050000	ams60	.text1	code					
00CB5000	00010000	ams60	.adata						
00CC5000	00010000	ams60	.data1	data, import					
00CD5000	00010000	ams60	.reloc1	relocations					
00CE5000	00320000	ams60	.pdata						
01005000	000DF000	ams60	.rsrc	resources					
010F0000	00006000								
01100000	00000000								

_Mo ArmInline up, enter the information process:

(Slave) Process ID: 0x 42C

Start Of Target Code: 0x 401000

Length Of Target Code: 0x 581000

IAT entry _Trong complete elimination as follows:

Import Elimination

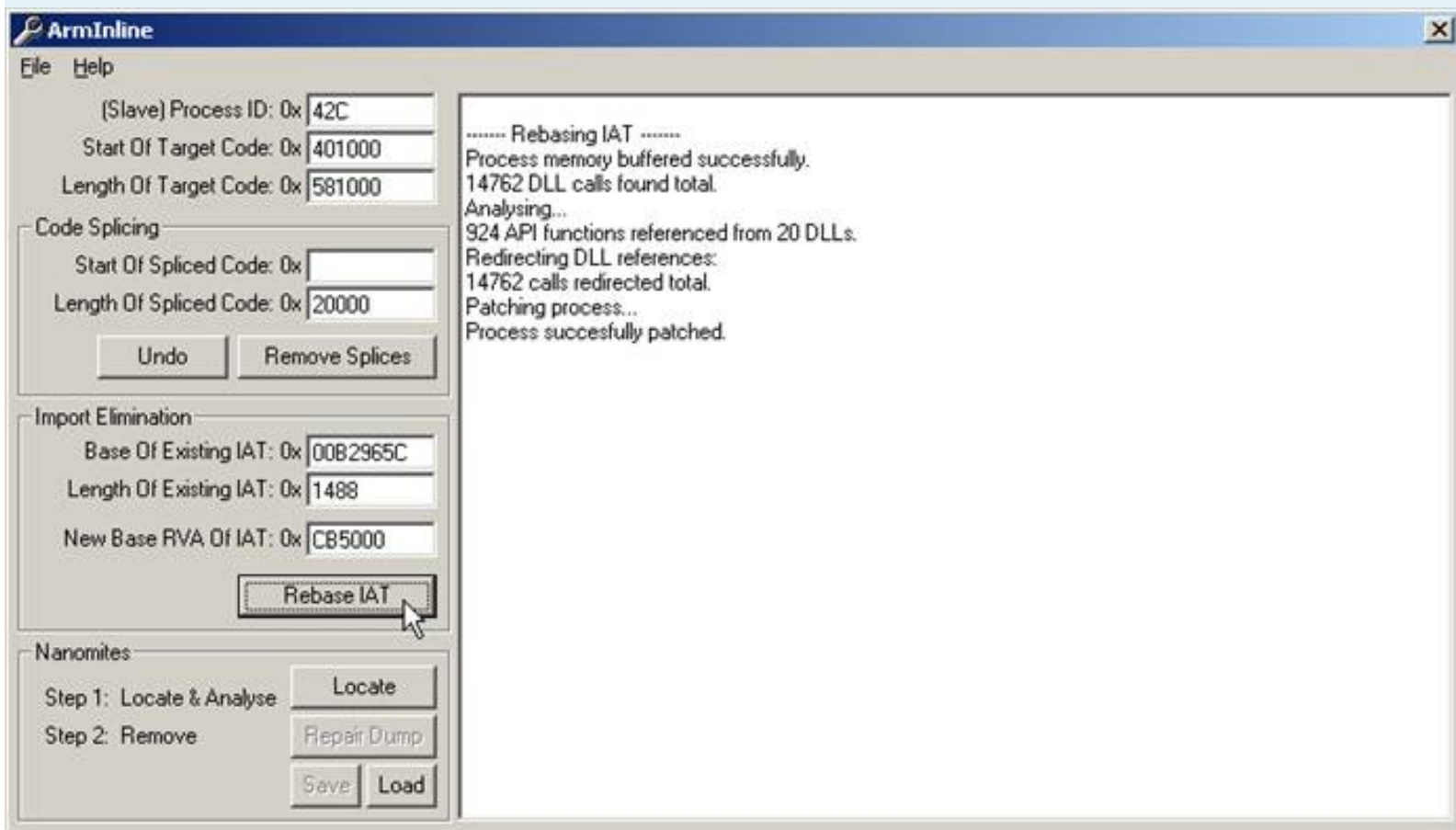
Base Of Existing IAT: 0x 00B2965C

Length Of Existing IAT: 0x 1488

New Base RVA Of IAT: 0x CB5000

Rebase IAT

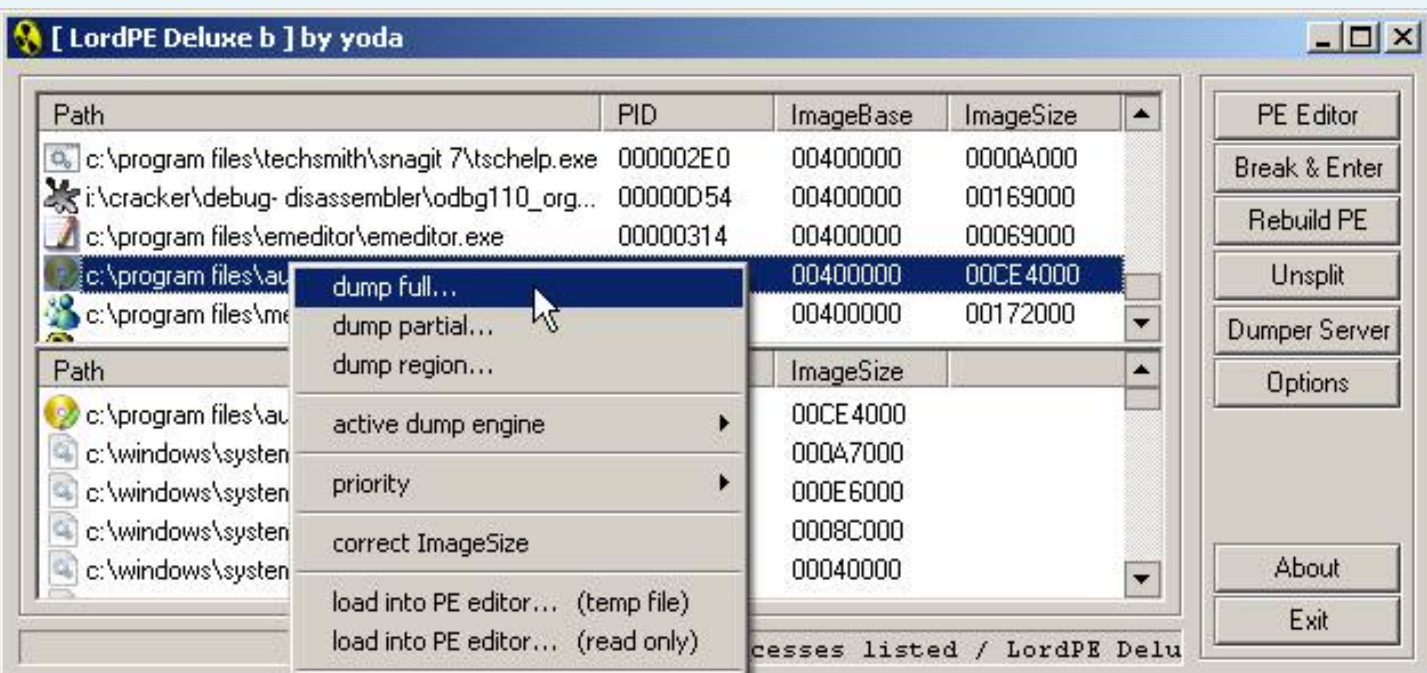
Rebase _Nhan IAT:



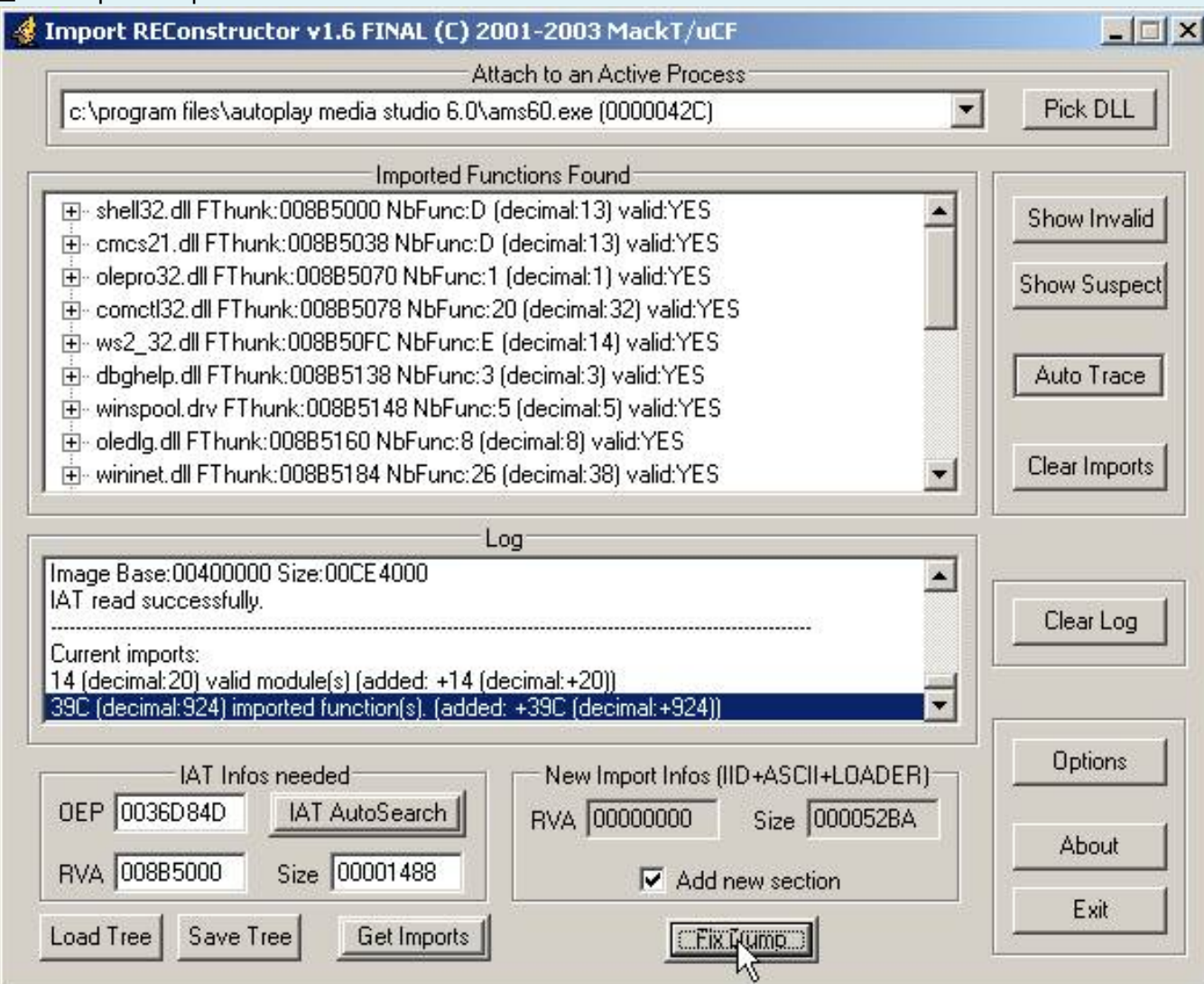
Quay The OllyDBG:

Address	Hex dump	Disassembly	Comment
0076D84D	55	PUSH EBP	
0076D84E	8BEC	MOV EBP,ESP	
0076D850	6A FF	PUSH -1	
0076D852	68 28C29C00	PUSH amd60.009CC228	
0076D857	68 685F7700	PUSH amd60.00775F68	
0076D85C	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
0076D862	50	PUSH EAX	amd60.00CC560C
0076D863	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
0076D86A	83EC 58	SUB ESP,58	
0076D86D	53	PUSH EBX	
0076D86E	56	PUSH ESI	amd60.00CCD098
0076D86F	57	PUSH EDI	amd60.00401000
0076D870	8965 F8	MOV DWORD PTR SS:[EBP-18],ESP	
0076D873	FF15 D45DCB00	CALL DWORD PTR DS:[CB5DD4]	kernel32.GetVersion
0076D879	33D2	XOR EDX,EDX	
0076D87B	8A04	MOV DL,AH	
0076D87D	8915 C0EFB000	MOV DWORD PTR DS:[B0EFC0],EDX	
0076D883	8BC8	MOV ECX,EAX	amd60.00CC560C
0076D885	81E1 FF000000	AND ECX,0FF	
0076D888	890D BCEFB000	MOV DWORD PTR DS:[B0EFBC],ECX	amd60.0076D84D
0076D891	C1E1 08	SHL ECX,8	
0076D894	03CA	ADD ECX,EDX	
0076D896	890D B8EFB000	MOV DWORD PTR DS:[B0EFB8],ECX	amd60.0076D84D
0076D89C	C1E8 10	SHR EAX,10	
0076D89F	A3 B4EFB000	MOV DWORD PTR DS:[B0EFB4],EAX	amd60.00CC560C
0076D8A4	6A 01	PUSH 1	
0076D8A6	E8 486D0000	CALL amd60.007745F3	
0076D8AB	59	POP ECX	01700442
0076D8AC	85C0	TEST EAX,EAX	amd60.00CC560C
0076D8AE	75 08	JNZ SHORT amd60.0076D8B8	
0076D8B0	60 1C	PUSH 1C	

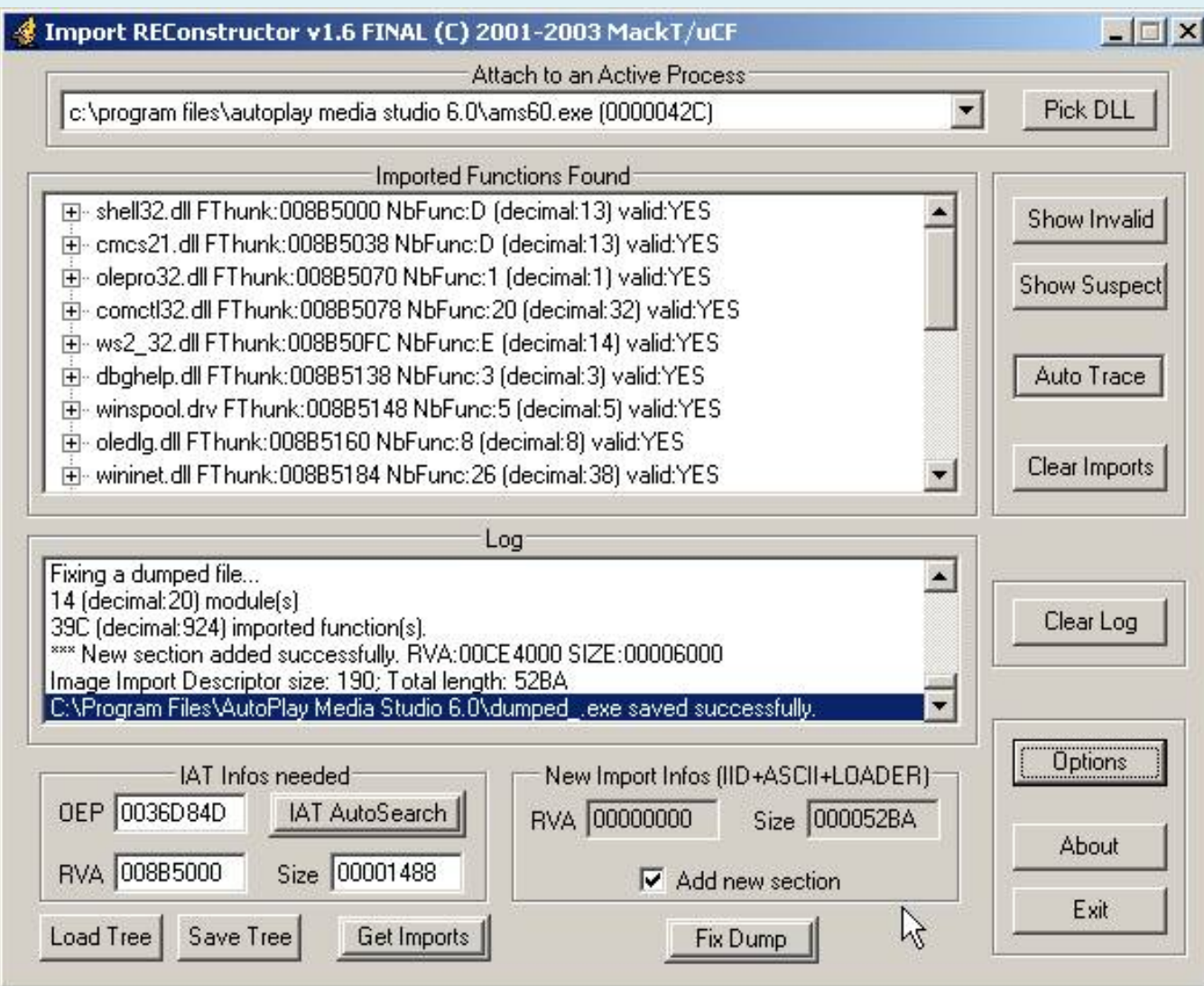
Full _LordPE dump:



Mo ImpREC up:



Fix Dump:



Run Try dumped.exe:

Activation System



Introduction

Please take a few moments to activate your copy of AutoPlay Media Studio. Activation protects Indigo Rose from illegal distribution of its software and helps ensure that you have a genuine copy of the software.

Internet Connection Detected

It only takes a few seconds to activate the product by using the Internet. To activate AutoPlay Media Studio using the available Internet connection, click the Next button.

To activate by email or web browser, or to delay activation for a limited period of time, click the Other Activation Options button.

More Information

To review the License Agreement you have accepted, click the License Agreement button.

[License Agreement](#)

Indigo Rose does not receive or use any personal data during activation. Click the Privacy Statement button for information.

[Privacy Statement](#)



You have 0 days to complete activation in order to continue using this product

[Other Activation Options](#)

[Back](#)

[Next](#)

[Cancel](#)

Ro That you have not unpack, but 7 days, months where good run but it expired. Optimize the only one that has dumped.exe file:

CFF Explorer I - by Ntoskrnl - [dumped_.exe]

File Settings ?

File: dumped_.exe

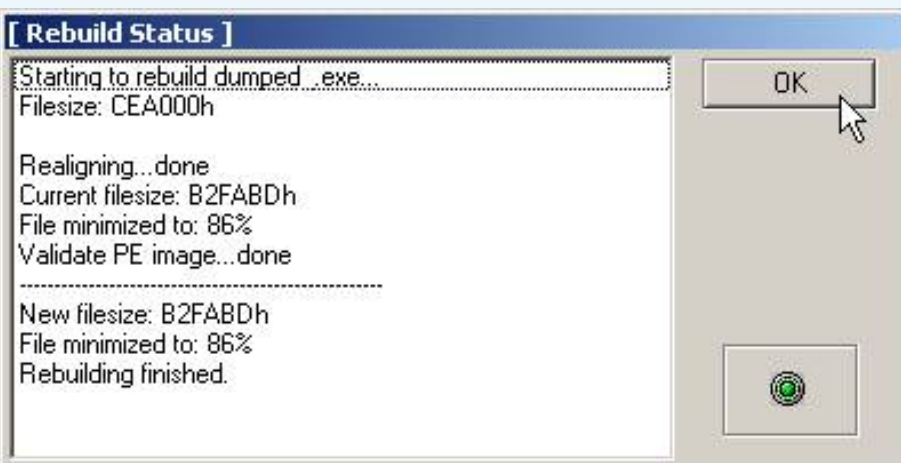
- Dos Header
- Nt Headers
 - File Header
 - Optional Header
 - Data Directories [15]
- Section Headers [x]**
- Import Directory
- Resource Directory
- Address Converter**
- Rebuilder**
- Resource Viewer**
- Hex Editor**

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address
Byte[8]	Dword	Dword	Dword	Dword	Dword
.text	00580CDC	00001000	00580CDC	00001000	00000000
.rdata	000CABFC	00582000	000CABFC	00582000	00000000
.data	000DA908	0064D000	000DA908	0064D000	00000000
.idata	000E5F01	00728000	000E5F01	00728000	00000000
.reloc	000568BD	0080E000	000568BD	0080E000	00000000
.text1	00050000	00865000	00050000	00865000	00000000
.adata	00010000	00885000	00010000	00885000	00000000
.data1	00010000	008C5000	00010000	008C5000	00000000
.reloc1	00010000	008D5000	00010000	008D5000	00000000
.pdata	00320000	008E5000	00320000	008E5000	00000000
.rsrc	000DF000	00C05000	000DF000	00C05000	00000000
.mack	00006000	00CE4000	00006000	00CE4000	00000000

Delete End:

- Change Section Flags
- Add Section (Header Only)
- Add Section (Empty Space)
- Add Section (File Data)
- Delete Section (Header Only)
- Delete Section (Header And Data)**
- Rebuild Image Size
- Rebuild PE Header
- Dump Section

_Sau Rebuild it with LordPE:



_dat called Rebuilt.exe. Open file Rebuilt.exe OllyDBG to search:

Address	Disassembly	Text string
004C478B	PUSH rebuild.00A57080	ASCII "%s" %s"
004C493A	PUSH rebuild.00A57090	ASCII "Invalid activation code."
004C4B13	PUSH rebuild.00A57078	ASCII "Submit failed: %s"
004C5154	PUSH rebuild.00A572CC	ASCII "Elapsed Time: %.2d:%.2d"
004C55B5	PUSH rebuild.00A572E8	ASCII "You have %d days to complete activation in order to continue using this product"
004C57A0	PUSH rebuild.00A57348	ASCII "\\License.txt"
004C57BC	PUSH rebuild.00A4E580	ASCII "open"
004C5900	PUSH rebuild.00A57358	ASCII "\\Data\\Privacy.txt"
004C591C	PUSH rebuild.00A4E580	ASCII "open"
004C5E9F	PUSH rebuild.00A560BC	ASCII "Invalid Activation Code"
004C60E0	PUSH rebuild.00A57370	ASCII "http://www.indigorose.com/route.php?pid=ams60activate"
004C60E5	PUSH rebuild.00A4E580	ASCII "open"
004C6441	PUSH rebuild.00A572E8	ASCII "You have %d days to complete activation in order to continue using this product"
004C6680	PUSH rebuild.00A57348	ASCII "\\License.txt"
004C66CC	PUSH rebuild.00A4E580	ASCII "open"
004C6810	PUSH rebuild.00A57358	ASCII "\\Data\\Privacy.txt"
004C682C	PUSH rebuild.00A4E580	ASCII "open"
004C6985	PUSH rebuild.00A57408	ASCII "Resolving address..."
004C698C	PUSH rebuild.00A574C4	ASCII "Address resolved"
004C6993	PUSH rebuild.00A574A8	ASCII "Connecting to server..."
004C699A	PUSH rebuild.00A5748C	ASCII "Connection established"
004C69A1	PUSH rebuild.00A57474	ASCII "Sending request..."
004C69A8	PUSH rebuild.00A57464	ASCII "Request sent"
004C69AF	PUSH rebuild.00A57448	ASCII "Waiting for response..."
004C69B6	PUSH rebuild.00A57430	ASCII "Response received"
004C69BD	PUSH rebuild.00A57414	ASCII "Closing connection..."
004C69C4	PUSH rebuild.00A573FC	ASCII "Connection closed"
004C69CB	PUSH rebuild.00A573E8	ASCII "Handle created"
004C69D2	PUSH rebuild.00A573D8	ASCII "Handle closed"
004C69D9	PUSH rebuild.00A573C0	ASCII "Operation complete"
004C69E0	PUSH rebuild.00A573B0	ASCII "Working..."
004C6DC8	PUSH rebuild.00A51A54	ASCII "Unknown"
004C6DE2	PUSH rebuild.00A51A54	ASCII "Unknown"
004C6DEE	PUSH rebuild.00A51A54	ASCII "Unknown"
004C6F6E	PUSH rebuild.00A575A0	ASCII "Invalid serial number. Please re-install the product and try again."
004C6F8C	PUSH rebuild.00A57584	ASCII "Invalid Serial Number"
004C7057	PUSH rebuild.00A57574	ASCII "Thank You!"
004C706D	PUSH rebuild.00A57510	ASCII "We appreciate you taking the time to activate this copy of AutoPlay Media Studio."
004C7095	PUSH rebuild.00A574F4	ASCII "Activation Failed"
004C726D	PUSH rebuild.00A57600	ASCII "USERNAME"
004C7282	PUSH rebuild.00A575F4	ASCII "USERKEY"
004C72ED	PUSH rebuild.00A57614	ASCII "IR_CERTIFICATE"
004C7307	PUSH rebuild.00A5760C	ASCII "PERM"
004C736D	PUSH rebuild.00A57650	ASCII "INVALIDKEY"
004C737F	PUSH rebuild.00A57640	ASCII "Unregistered"
004C73F0	PUSH rebuild.00A57634	ASCII "EXPIRED"
004C741B	PUSH rebuild.00A57628	ASCII "DAYSLEFT"
004C759F	PUSH rebuild.00A57660	ASCII "Are you are sure that you want to exit AutoPlay Media Studio?"

_Sau A fuzzy quật J: We run Rebuilt_patched.exe file:

AutoPlay

MEDIA STUDIO 6.0™

Registered to: hacnho

IndigoRose
SOFTWARE DESIGN CORP.

AutoPlay
MEDIA STUDIO 6.0™

ABOUT AUTOPLAY MEDIA STUDIO 6.0

THE PROFESSIONAL CHOICE

Make a great first impression with a professional autorun CD built using AutoPlay Media Studio 6.0, the favorite choice of design professionals. This easy to use software tool allows you to quickly create custom autorun menus, interactive presentations, and custom applications in just minutes.

Use your existing content such as images, music, video, flash, text, and more, and simply drag n' drop your way to amazing menus. Your products will fly off the shelves with a professional front end by AutoPlay Media Studio 6.0. Visit us online at www.indigorose.com for support and more!

Registered to: hacnho



_Day A patch:



_Ban Patch this private not public! Like the mail to send bags!

_Unpacked SuccessFul!

4. Conclusion

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtlN, hosiminh, Nilrem, fly, Madman_Hercules, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RiF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx OllyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 06/10/2005)

Basic steps in the process unpack!

The ray can ask me bac much about this, really did not have 1 set method called for it. The Department, have been able to unpack the new packer test any one must go through these steps:

1.Hoc question, see the high degree of cracker in the world who can not unpack success. If you already have heart test that school ... see how dominant they find OEP, fix IAT ... How would. Then unpack from mo based on the know, do as many hands as familiar, the more experience. (Typically, as packer Asprotect maximum unpack more than 200 soft, full format, all types).

2.Tu seeking packer, pack your own files already know and find the OEP. From that can draw, but common to find OEP, IAT. (eg as to unpack SVKP 1.3, maximum download this packer, self pack file notepad.exe then find your file has OEP pack)

OEP 3.Cac heart is the most fundamental, individual orders trace 1, until I OEP. Additionally level must know the number 1 france antidebug obtain the program or content so that children know that against.

Bac 4.Co number 1 question, how is that biet OEP, which would provide thura is only guesswork. When bac've worked with 1 large number of files, will have 1 bac feelings (which is called also MO) is OEP. In fact, with the current Translation as Delphi, VC + +, BC + + ... OEP commercial competition started with orders balance stack is simple:

```
PUSH EBP
MOV EBP, ESP
...
```

So when traders trace order to meet that test we should save y.

5.Viec fixed IAT. Also do not have the general france not. Currently, you I also use 1 of the tool is 2 Imprec (or ReVirgin). This is a tool to help you fix IAT easily without the need to understand about intimate PE Header.

Here is a simple example to describe the steps that have been mentioned above:

I - Find OEP:

- Use PEiD we know is the program is using *ASPack Pack 2.1 -> Alexey Solodovnikov*.
- Load up the program with Olly. Select No (not Analysis). I will come:

CODE

```
0051F001> 60
PUSHAD <== We
here
0051F002 E8
72050000 CALL
AutoStar.0051F579
```

- Press **F8** one command line to the next. In the book **Registers (FPU)** we have to lick your mouse in the value **ESP** and **Follow Indump**.
- **Hex dump** window will switch to new window. Select **the first 4 bytes** of this window and **lick your mouse** to select **BreakPoint ==> Hardware, on access ==> Dword**. Then press **F9** to us:

CODE

```
0051F4F4 / 75 08
JNZ SHORT
AutoStar.0051F4FE

<== We here
0051F4F6 | b8
01000000 MOV
EAX, 1
0051F4FB | C2
0C00 RETN 0C
0051F4FE \ 68
00104000 PUSH

AutoStar.00401000
0051F503 C3 RETN
<=== Push F8
coming here and
Push F8 again
```

- Then we will come:

CODE

```
00401000 A1
78834900
MOV EAX,
DWORD PTR
DS:
[498378]
<== We here
```

- **Note:** After coming here is not using the mouse or do anything on the screen move up and down
- Then select the next **Plugins ==> Olly dump ==> dump debugged process**. After you select will be completed at the **Entry Point** in the box -> **Modify** is the value of the **OEP**. We save this value. Lick OK and save as you like (the extension is. Exe).

II - Find and RVA RVA size:

- You will find two ways: (1) **IAT Auto Search**, and (2) **Manual**
- You should find using **IAT Auto Search** for more quickly.
- However, in some cases **IAT Auto Search** can not find out **RVA RVA** and **size** so we must be heart by **Manual**.

II.1 - Content IAT Auto Search:

- Stay in the program, the program **ImportREC v1.6** load the file, change the value **OEP** we just find the above.
- The **IAT Auto Search**
- Trade should we change the last two of **RVA to RVA Size 00** and we should increase slightly
- Then **Get** the **Imports**, the next **Show Invalid**, then the **Trace level 1 (disasm)**.
- Finally **Fix** the **dump**. Select the file that you get saved above.
- **Fix** then **dump** (file with the fix is a new file is created above).
- I will be a new file. If nothing special examination FILE has been successfully unpack.

II.2 - Manual:

- Stay in main screen. Press **Ctrl-B** screen in **the CPU**, and go to the **FF 25**, then press **Ctrl-L** continue to find code to the like this:

CODE

```

00497284 -
FF25
34214B00
JMP DWORD
PTR DS:
[4B2134];
advapi32.

GetUserNameA

```

- Press **Ctrl-L** to find out the code Similarly. In the heart, note the largest value **max** and **min** in most small **DS: [4B2134]**.

- So we have:

OEP = value we find the above

RVA = min - 400,000

Size = max - min ==> usually increases slightly

- Use **Import REConstructor v1.6F © 2001-2003** load file.

- Complete the value calculation above the **IAT infos needed**. Then **Get** the **Imports**, the next **Show Invalid**, then the **Trace level 1 (disasm)**.

- After the trace is complete, the next **Show Invalid**, then the next **Show Invalid**, and the **Cut thunks**.

- **Fix** then **dump** (file with the fix is a new file is created above).

- I will be a new file. If nothing special examination FILE has been successfully unpack.

III - How clean and reduce the file size after unpack:

- To improve our conduct over the cleaning and reduce the file size of the file after **Fix dump** (to file as small as possible. However, if you do not like you can ignore).

- To implement this process we use **LordPE Delux v1.4**. Load up the program, select **rebuild PE**. Select File **Fix** we **dump** on, lick the **Open** and we have a complete new file to run CRACK.

- Check the file with **PEiD**, we know the language of the program.

IV - Note:

- The instructions on here to correct **most cases** encountered in the SOFTWARE or CRACKME.

- Some special cases we need to initiate a slightly different take slightly. However the basic steps do not have the difference.

- Address in the instructions may vary, but you should save italy ac me to order. The question this order can not be different.

Bypass Registration EncryptPE V2.2007 (WhynotBar)

_Dùng UltraEdit mở "EncryptPE V2.2007.exe"

```
00000250h: 4550453A20456E637279707450452056 ; EPE: EncryptPE V
00000260h: 322E323030372E31322E312C20436F70 ; 2.2007.12.1, Cop
00000270h: 79726967687420284329205746530000 ; yright (C) WFS..
00000280h: 486F6D65506167653A207777772E656E ; HomePage: www.en
00000290h: 637279707470652E636F6D0000000000 ; cryptpe.com.....
000002a0h: 454D61696C3A2077667323656E637279 ; EMail: wfs#encry
000002b0h: 707470652E636F6D0000000000000000 ; ptpc.com.....
```

_OK...Nó protect bằng EncryptPE V2.2007.12.1. Load vào Olly và Shift+F9

```
00405000 > 60 PUSHAD
00405001 9C PUSHFD
00405002 64:FF35 00000000>PUSH DWORD PTR FS:[0]
00405009 E8 1B020000 CALL EncryptP.00405229
```

_ Shift+F9 Run soft, Alt+F1, BP ExitProcess và press OK

```
0012FA58 711F74A0 /CALL to ExitProcess from V2200712.711F749B
0012FA5C 00000000 \ExitCode = 0
0012FA60 71206ABF RETURN to V2200712.71206ABF from V2200712.711F7488
0012FA64 0012FC58 Pointer to next SEH record
```

_Follow in Disassembler, Search Text String và nhập vào ".Key"

```
1205DAE MOV EAX,V2200712.712060EC ASCII "-UNEPEREG"
1205DE5 MOV EAX,V2200712.71206100 ASCII "/UNEPEREG"
1206205 CMP DWORD PTR DS:[71225718 ASCII "MZP"
1206490 MOV EDX,V2200712.712073E4 ASCII "EncryptPE "
120659C PUSH V2200712.71207400 ASCII "npggnt.des"
1206788 MOV EDX,V2200712.71207414 ASCII "TEncryptPEForm"
1206A89 PUSH V2200712.7120745C ASCII ")<"
1206CBD MOV ECX,V2200712.71207474 ASCII ".key"
1206D5B MOV ECX,V2200712.71207474 ASCII ".key"
1206FC6 MOV EDX,V2200712.71207490 ASCII "NOSYSDLLS"
1207077 MOV EAX,V2200712.712074B0 ASCII ",NOSYSDLLS,"
120755E MOV EDX,V2200712.71207980 ASCII "宋体"
1207A4C MOV EDX,V2200712.71207C30 ASCII "Thank You."
1207ACB MOV EAX,V2200712.71207C50 ASCII "://"
```

_Ok...Set HE 71207077

```
71207077 B8 B0742071 MOV EAX,V2200712.712074B0 ; ASCII ",NOSYSDLLS,"
```

```
7120707C E8 8BE0F1FF CALL V2200712.7112510C
```

_Ctrl+F2, F9 dừng tại HWB vừa mới Set

```
71207077 B8 B0742071 MOV EAX,V2200712.712074B0 ; ASCII ",NOSYSDLLS,"
==>Pause here
```

```
7120707C E8 8BE0F1FF CALL V2200712.7112510C
```

```
71207081 85C0 TEST EAX,EAX
```

```
71207083 0F9FC0 SETG AL
```

```
71207086 8B55 FC MOV EDX,DWORD PTR SS:[EBP-4]
```

```
71207089 8882 52030000 MOV BYTE PTR DS:[EDX+352],AL
```

```
7120708F 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
```

```
71207092 8A90 52030000 MOV DL,BYTE PTR DS:[EAX+352]
```

```
71207098 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
```

```
7120709B 8B80 94030000 MOV EAX,DWORD PTR DS:[EAX+394]
```

```
712070A1 E8 DE9EFFFF CALL V2200712.71200F84
```

```
712070A6 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
```

```
712070A9 8B80 00030000 MOV EAX,DWORD PTR DS:[EAX+300]
```

```
712070AF 8378 0C 01 CMP DWORD PTR DS:[EAX+C],1
```

```
712070B3 75 2E JNZ SHORT V2200712.712070E3 ; ==>Jump Set AL=0
```

```
712070B5 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
```

```
712070B8 8B90 7C030000 MOV EDX,DWORD PTR DS:[EAX+37C]
```

```
712070BE 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
```

```
712070C1 8B80 18030000 MOV EAX,DWORD PTR DS:[EAX+318]
```

```
712070C7 E8 9896F7FF CALL V2200712.71180764 ; ==>Check Key
```

```
712070CC 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
```

```
712070CF 8B90 7C030000 MOV EDX,DWORD PTR DS:[EAX+37C]
```

```
712070D5 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
```

```
712070D8 8B80 20030000 MOV EAX,DWORD PTR DS:[EAX+320]
```

```
712070DE E8 8196F7FF CALL V2200712.71180764 ; ==>Check Key
```

```
712070E3 837D F8 00 CMP DWORD PTR SS:[EBP-8],0
```

```
712070E7 74 29 JE SHORT V2200712.71207112 ; ==>Jump Set AL=1
```

```
712070E9 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
```

```
712070EC 8B80 00030000 MOV EAX,DWORD PTR DS:[EAX+300]
```

```
712070F2 8378 0C 01 CMP DWORD PTR DS:[EAX+C],1
```

```
712070F6 74 0F JE SHORT V2200712.71207107
```

```
712070F8 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
```

```
712070FB 8B80 24030000 MOV EAX,DWORD PTR DS:[EAX+324]
```

```
71207101 8378 30 64 CMP DWORD PTR DS:[EAX+30],64
```

```
71207105 75 0B JNZ SHORT V2200712.71207112
```

```
71207107 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
```

```
7120710A 8B10 MOV EDX,DWORD PTR DS:[EAX]
```

```
7120710C FF92 E8000000 CALL NEAR DWORD PTR DS:[EDX+E8] ; ==>Call NAG Reg
```

```
71207112 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
```

```
71207115 8B80 24030000 MOV EAX,DWORD PTR DS:[EAX+324]
```

```
7120711B 33D2 XOR EDX,EDX
```

```
7120711D E8 8686F5FF CALL V2200712.7115F7A8
```

```
71207122 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
```



```

71207125 8B80 00030000 MOV EAX,DWORD PTR DS:[EAX+300]
7120712B 8378 0C 01 CMP DWORD PTR DS:[EAX+C],1
7120712F 75 24 JNZ SHORT V2200712.71207155 ; ==>Jump Set AL=1
71207131 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
71207134 8B80 24030000 MOV EAX,DWORD PTR DS:[EAX+324]
7120713A BA 01000000 MOV EDX,1
7120713F E8 7486F5FF CALL V2200712.7115F7B8
71207144 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
71207147 8B90 24030000 MOV EDX,DWORD PTR DS:[EAX+324]
7120714D 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
71207150 E8 43F0FFFF CALL V2200712.71206198 ; ==>Check Key
71207155 A1 583F2271 MOV EAX,DWORD PTR DS:[71223F58]

```

_Nếu làm đúng như trên nhấn F9 soft sẽ Run hoàn toàn. Nhấn Ctrl+G nhập vào 401000, **Search All intermodular Calls**

```

00401000 PUSH ES (Initial CPU selection)
0040116D CALL EncryptP.00401160 MSVBVM60.ThunRTMain
00401CC4 CALL FF405CE7
0040238F CALL NEAR DWORD PTR DS:[40 MSVBVM60.rtcMsgBox
004023AE CALL NEAR DWORD PTR DS:[40 MSVBVM60.__vbaVarMove
004023C6 CALL NEAR DWORD PTR DS:[40 MSVBVM60.__vbaFreeVarList
004023EB CALL NEAR DWORD PTR DS:[40 MSVBVM60.__vbaFreeVarList
004023F8 CALL NEAR DWORD PTR DS:[40 MSVBVM60.__vbaFreeVar

```

_Ok, set 1 HWB tại OEP và làm lại các bước trên.

```

00401160 - FF25 6C104000 JMP NEAR DWORD PTR DS:[40106C]
00401166 0000 ADD BYTE PTR DS:[EAX],AL
00401168 68 A81B4000 PUSH EncryptP.00401BA8 ; ==>This is OEP
0040116D E8 EFFFFFFF CALL EncryptP.00401160

```

_Tiến hành Dump và Fixed Dump như bình thường. Test thử thấy Run tốt. Dùng CFF xóa Section **EPE1** đi cho nhẹ và Save lại.

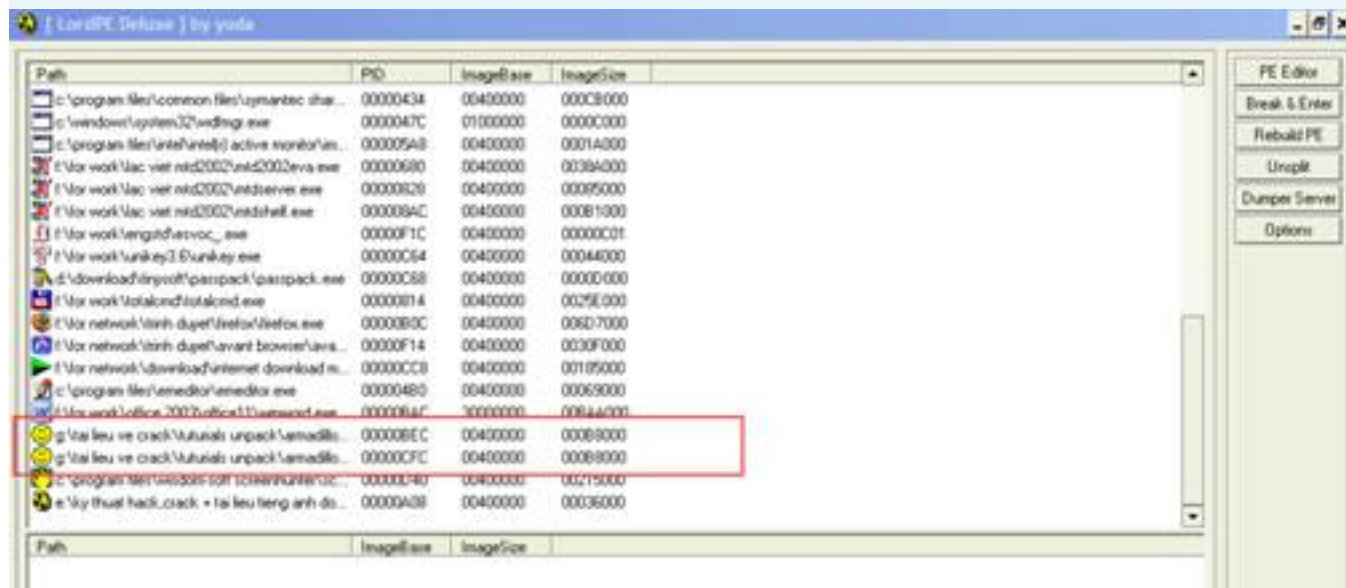
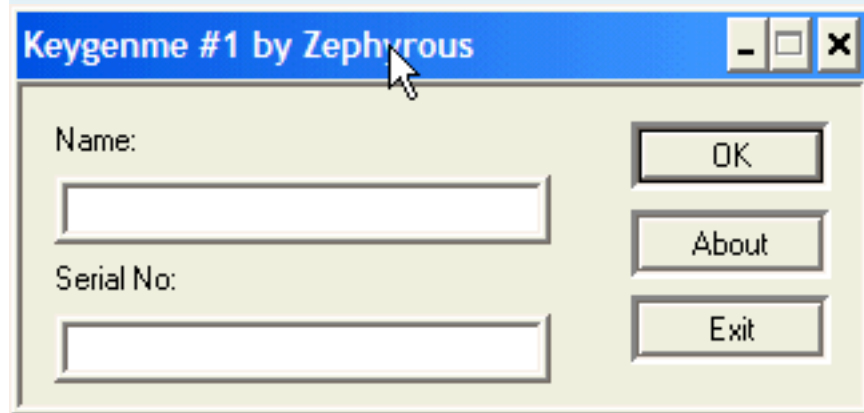
Source: <http://www.reversing.be/forum/viewtopic.php?t=287> (Fisker to thank for sharing this)

Target: Armadillo 4.30a debugblocker.exe (included in rar file)

Tool: Olly (edited by hacnho), lordPE, ImportREC.

Target this pack by 4:30 armadillo standard + debugblocker.

_Chay Normal:



Olly _Load target of:

hacnho - Armadillo 4.30a debugblocker.exe - [CPU - main thread, module Armadill]

File View Debug Plugins Options Window Help

Paused

Registers (FPU)

Register	Value
EAX	00000000
ECX	0012FFD0
EDX	77F5164E
EBX	77F5164E
ESP	0012FFD0
EBP	0012FFD0
EIP	00455000
EFL	00000000
CS	00000000
DS	00000000
SS	00000000
ES	00000000
FS	00000000
GS	00000000
LastErr	ERROR_SUCCESS (00000000)
EPL	00000000 (H0,H0,H0,H0,H0,H0,H0,H0)
ST0	empty
ST1	empty
ST2	empty
ST3	empty
ST4	empty
ST5	empty
ST6	empty
ST7	empty
FST	0000
FCM	0000

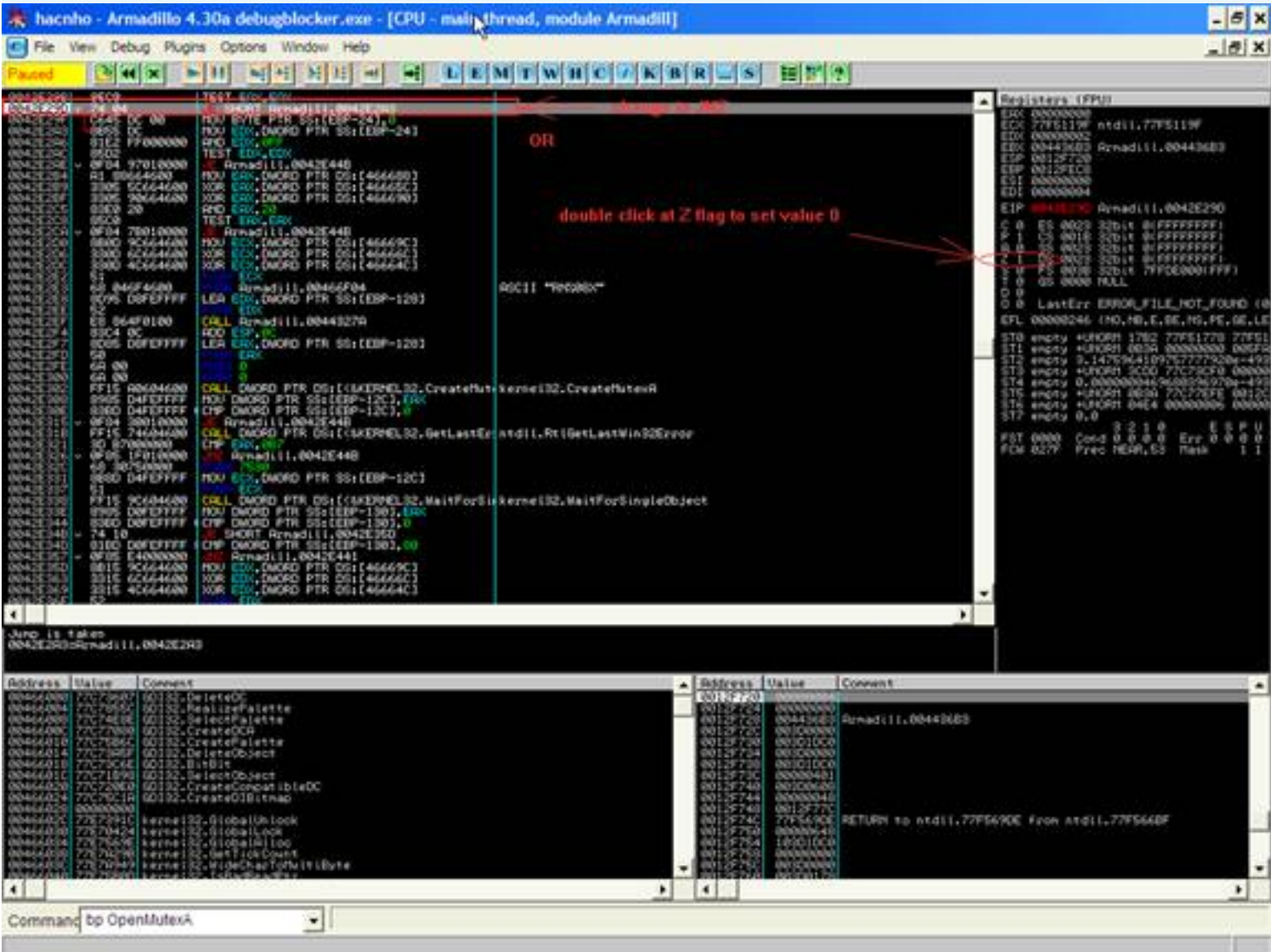
Address Value Comment

Address	Value	Comment
00455000	77C72000	00122: DeleteDC
00455004	77C72000	00122: RealizePalette
00455008	77C72000	00122: SelectPalette
0045500C	77C72000	00122: CreateDC
00455010	77C72000	00122: CreatePalette
00455014	77C72000	00122: DeleteObject
00455018	77C72000	00122: BitBlt
0045501C	77C72000	00122: SelectObject
00455020	77C72000	00122: CreateCompatibleDC
00455024	77C72000	00122: CreateDIBitmap
00455028	00000000	
0045502C	77C72000	kernel32: GlobalUnlock
00455030	77C72000	kernel32: GlobalLock
00455034	77C72000	kernel32: GlobalAlloc
00455038	77C72000	kernel32: GetTickCount
0045503C	77C72000	kernel32: WideCharToMultiByte
00455040	77C72000	kernel32: LocalFree

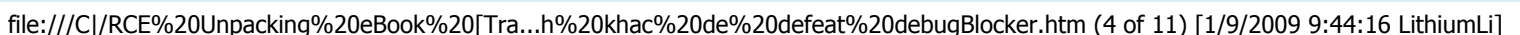
Command: bp OpenMutexA

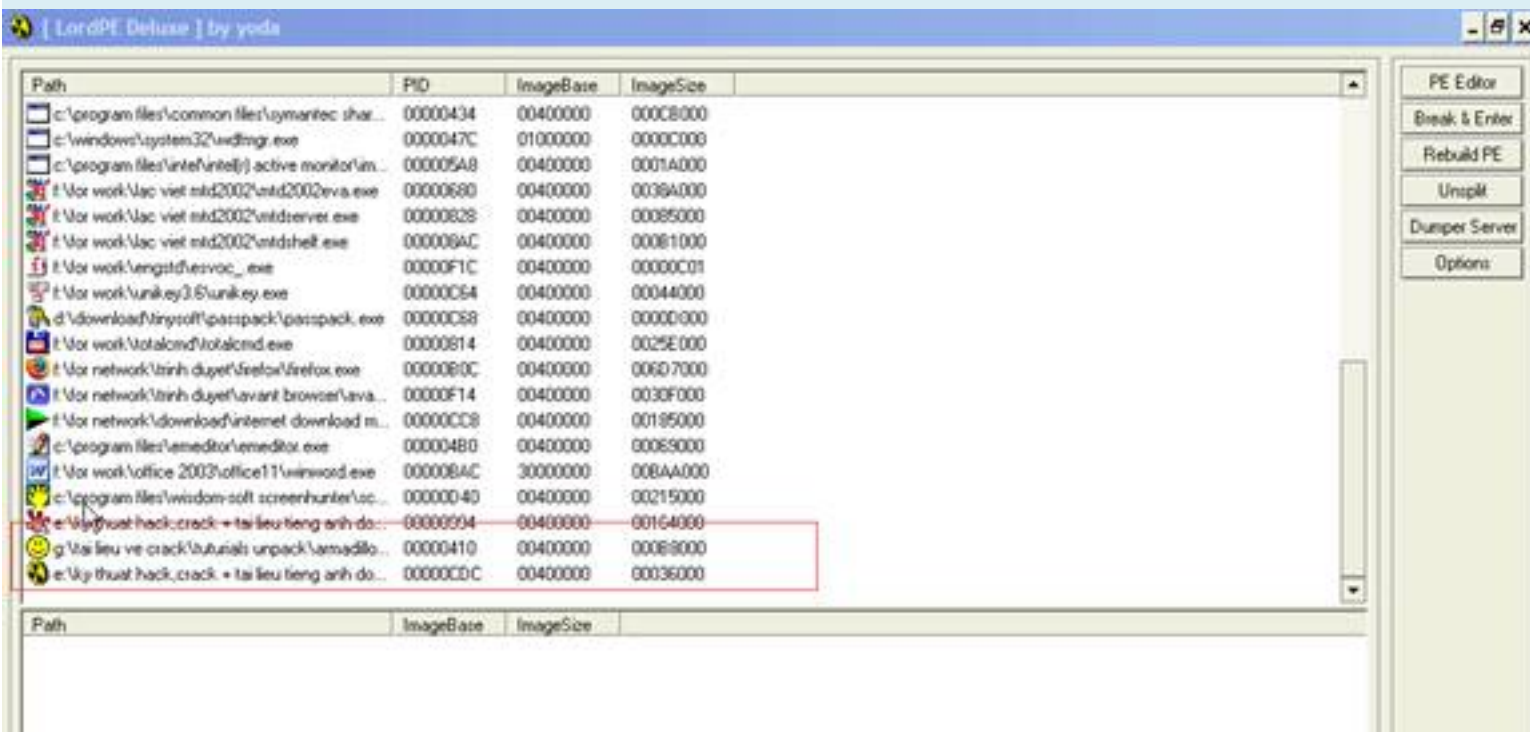
Program entry point

_bp OpenMutexA => F9 => Olly break => Alt + F9, F8:



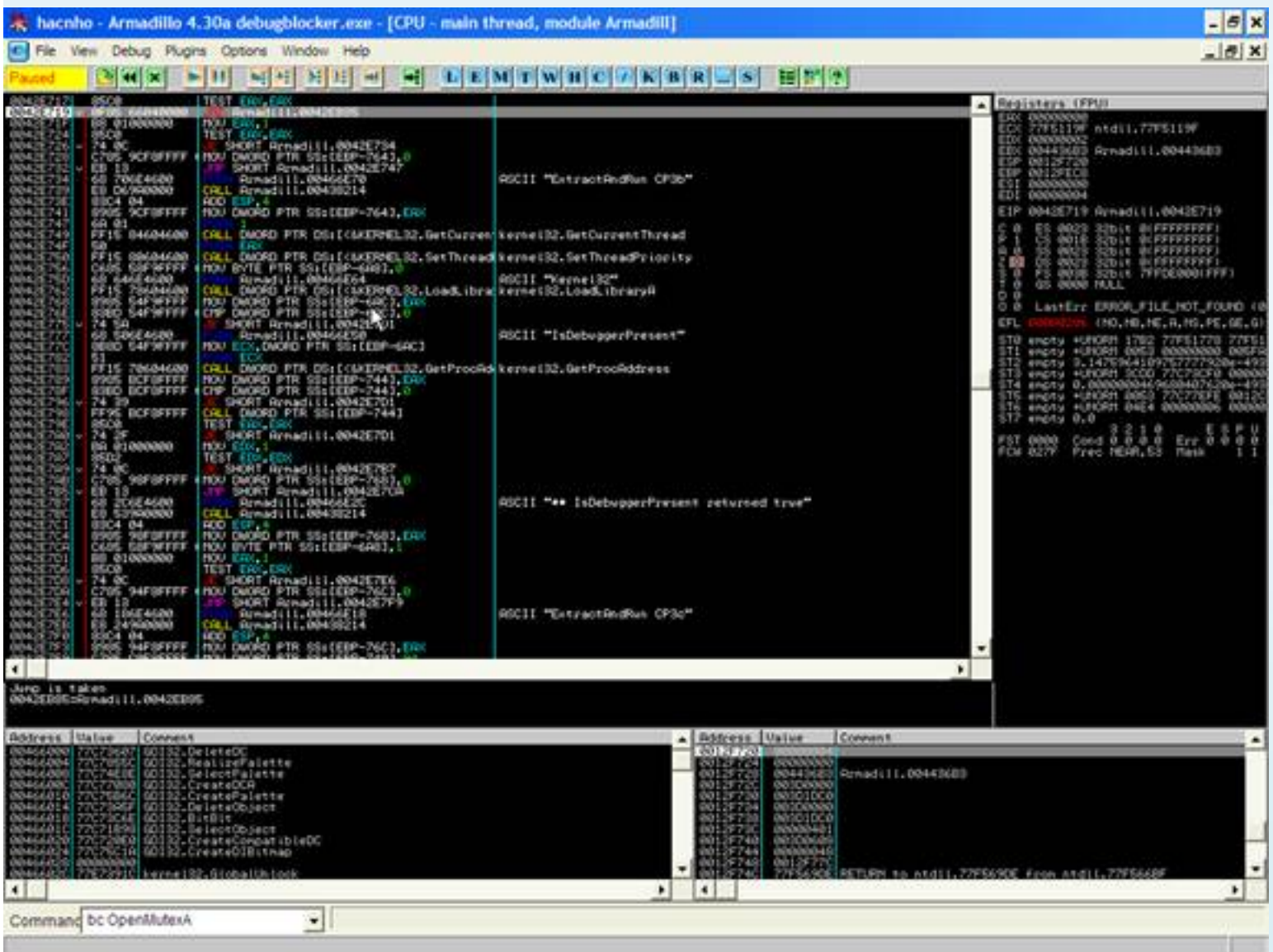
_F9 Play again, olly break, Alt + F9, F8:



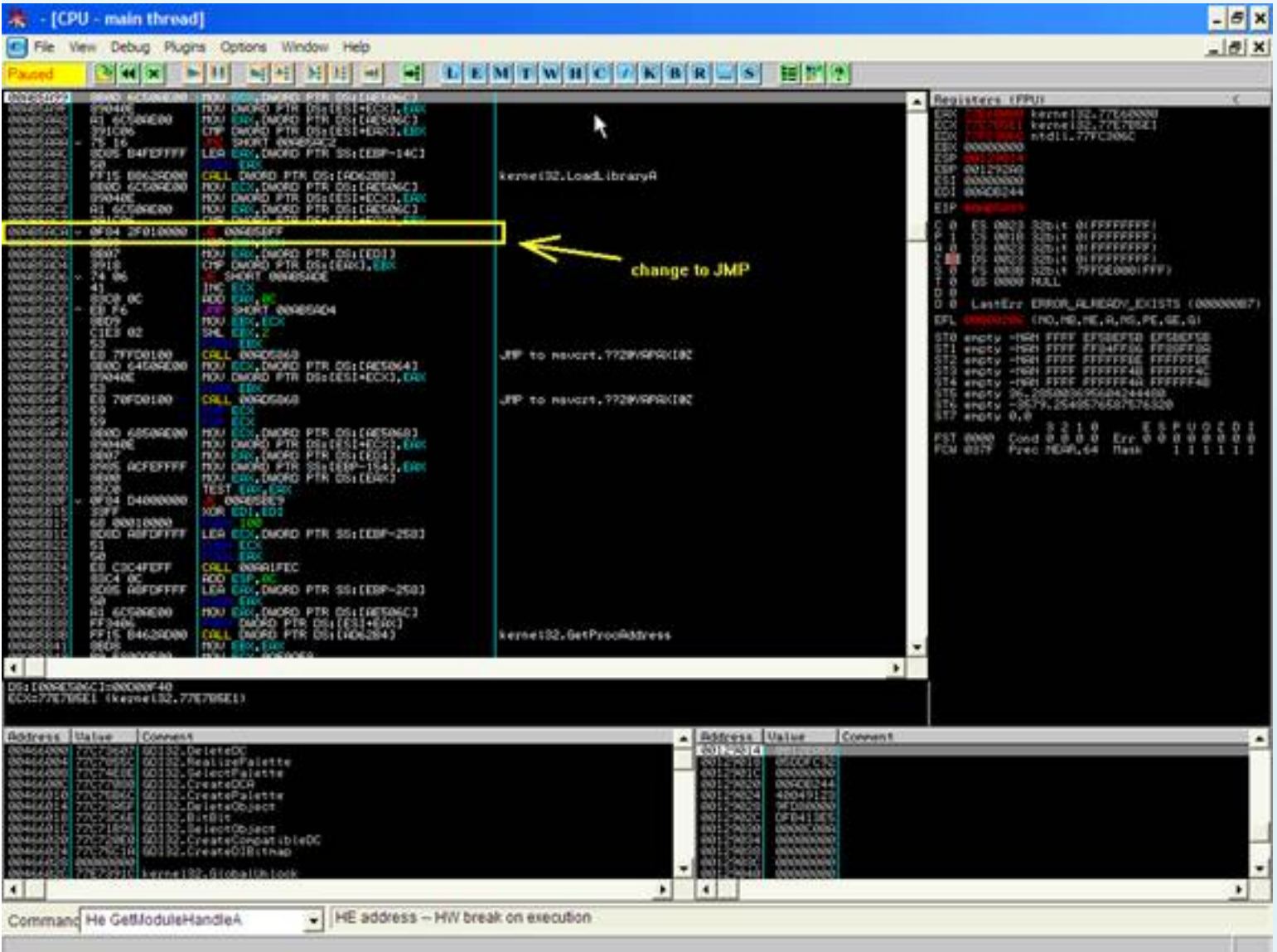


_ As the authors say, the following steps on the patch unpack as usual for the standard. Here are the steps of the child:

_ Load target of olly, BP OpenMutexA, F9, Alt + F9, F8, as the patch, the more F9, Alt + F9, F8, as the patch, BC OpenMutexA. She here:

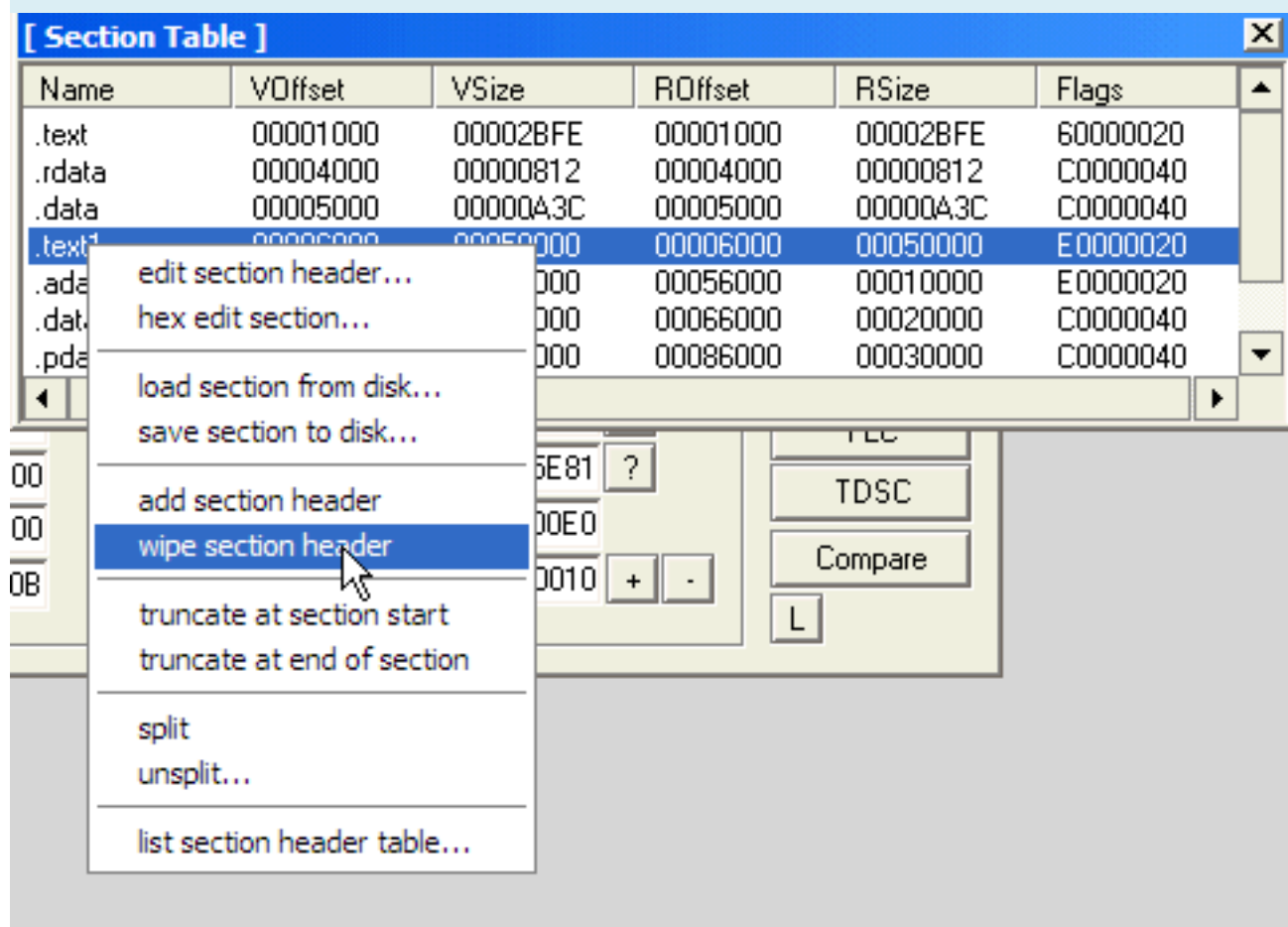


_ Now is the magic fix jump: F9, nag appear, click OK drilling, Olly invisible: He GetModuleHandleA, OK nag, olly break, Alt + F9:



_ After patch magic jump: hd GetModuleHandleA, BP CreateThread, F9, olly break, Ctrl + F9, Ctrl + F9, set breakpoint in order ECX last call, F9, F7 -> OEP:

file:///C:/RCE%20Unpacking%20eBook%20[Tra...h%20khac%20de%20defeat%20debugBlocker.htm (8 of 11) [1/9/2009 9:44:16 LithiumLi]



_Ta Turn wipe the section header section. Text1. ADATA. Data1. Pdata.

[Section Table]					
Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	00002BFE	00001000	00002BFE	60000020
.rdata	00004000	00000812	00004000	00000812	C0000040
.data	00005000	00000A3C	00005000	00000A3C	C0000040
.rsrc	00006000	00002000	00006000	00002000	40000040
.mact	00008000	00001000	00008000	00001000	E0000060

_Dong On the table since we know the value of code base and data base of:

[PE Editor] - g:\tai lieu ve crack\tutorials unpack\armadillo\loat tut ben biw rat t...

Basic PE Header Information

EntryPoint:	000013FB	Subsystem:	0002	...
ImageBase:	00400000	NumberOfSections:	0005	
SizeOfImage:	000B9000	TimeDateStamp:	3E5900E7	
BaseOfCode:	00001000	SizeOfHeaders:	00001000	? +
BaseOfData:	00005000	Characteristics:	010F	...
SectionAlignment:	00001000	Checksum:	00095E81	?
FileAlignment:	00001000	SizeOfOptionalHeader:	00E0	
Magic:	010B	NumOfRvaAndSizes:	00000010	+ -

OK
Save
Sections
Directories
FLC
TDSC
Compare
L

_Save, OK, and the option in lordPE:

[Options]

PEEditor

☒ Section Table: autofix SizeOfImage
☐ Split: ask for file locations
☒ Register shell extension

Task Viewer

☒ Full dump: paste header from disk
☒ Full dump: fix Header
☐ Full dump: force RAW mode
☐ Full dump: rebuild image
☐ Agreement required for terminations
☐ Delete temp files for PE editor

General

☐ Always on top
☒ Restore last directory on startup
☒ Restore last main window position

Break & Enter

☒ Register shell extension

Drag'n'dropped files are...

...for the PE editor
 ...for Break'n'Enter
 ...for the Rebuilder

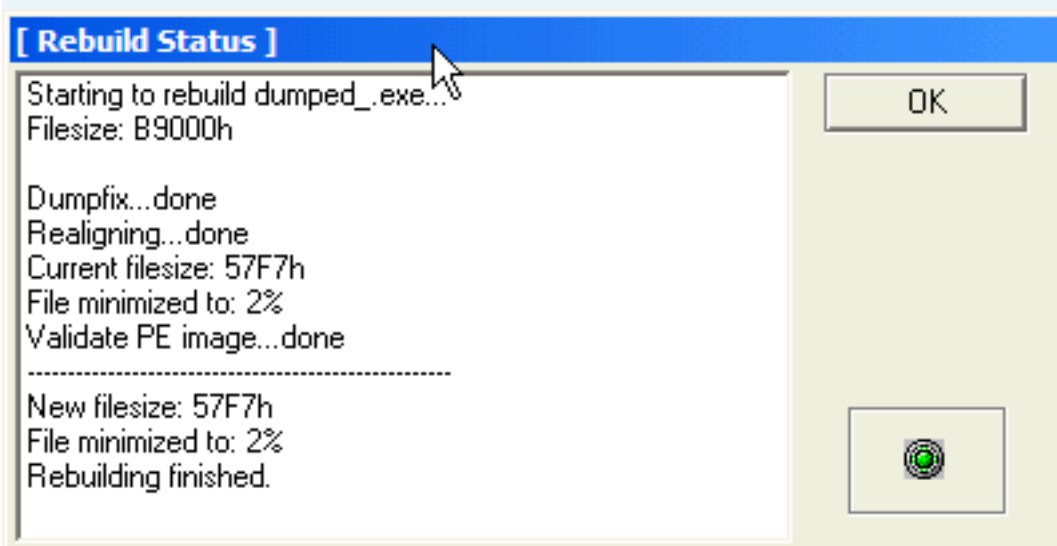
Rebuilder

☒ Status window
☒ Dumpfix
☒ Realign file...
 normal
 nice
 hardcore

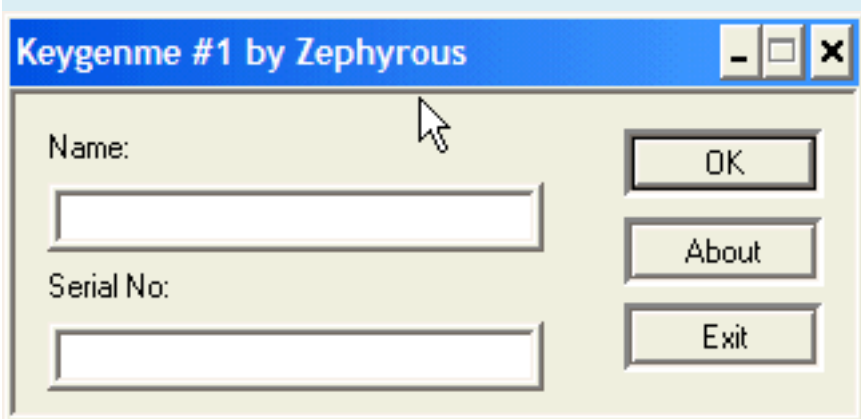
☐ Wipe Relocation
☐ Rebuild ImportTable
☒ Validate PE
☐ Bind Imports
☐ Change ImageBase to:
 00400000

OK
Cancel

_Rebuild PE:



_ File dumped_.exe now have more than 20KB, run test:



_Done

+++ [Armadillo _CodeSplicing the French --- Ta (Evil)]+++ Method

- Welcome Newbie perimeter in REA and aged bay long now enthralled track series jewel "Armadillo - collect sand stone" of the hacnho about unpack Armadillo. Tut Through this series, many people have multiple Also search of new, filled with aged Available for joy, surprise, the man has become crazy wild trains by entering the code, they have special children when they were removing the chiêu **Terminate**. Wiuewi As we have time crying, are laughing at, happy sad confused, and even the mouse is always a dear in the strength of the throw it on the screen. (>..<)

- Now when Tàn per training programs to expand 7 (Tut7_exp) of the hacnho. Tàn Mo was crying 2 times already. Khóc first time after many nights that do not swallow the floating leg of this dish, even when **Coding Workshop Polyphonic Wizard** expiration of training, per Tàn also still not yet been any further steps. Buồn owner of Ah, what is sad when any one is the one to fail and leave behind a Target ridiculous insult ... embracing furniture expectation that Tàn per was wandering for days, not to know on that night, until at the **Cuu wifeless About** ...

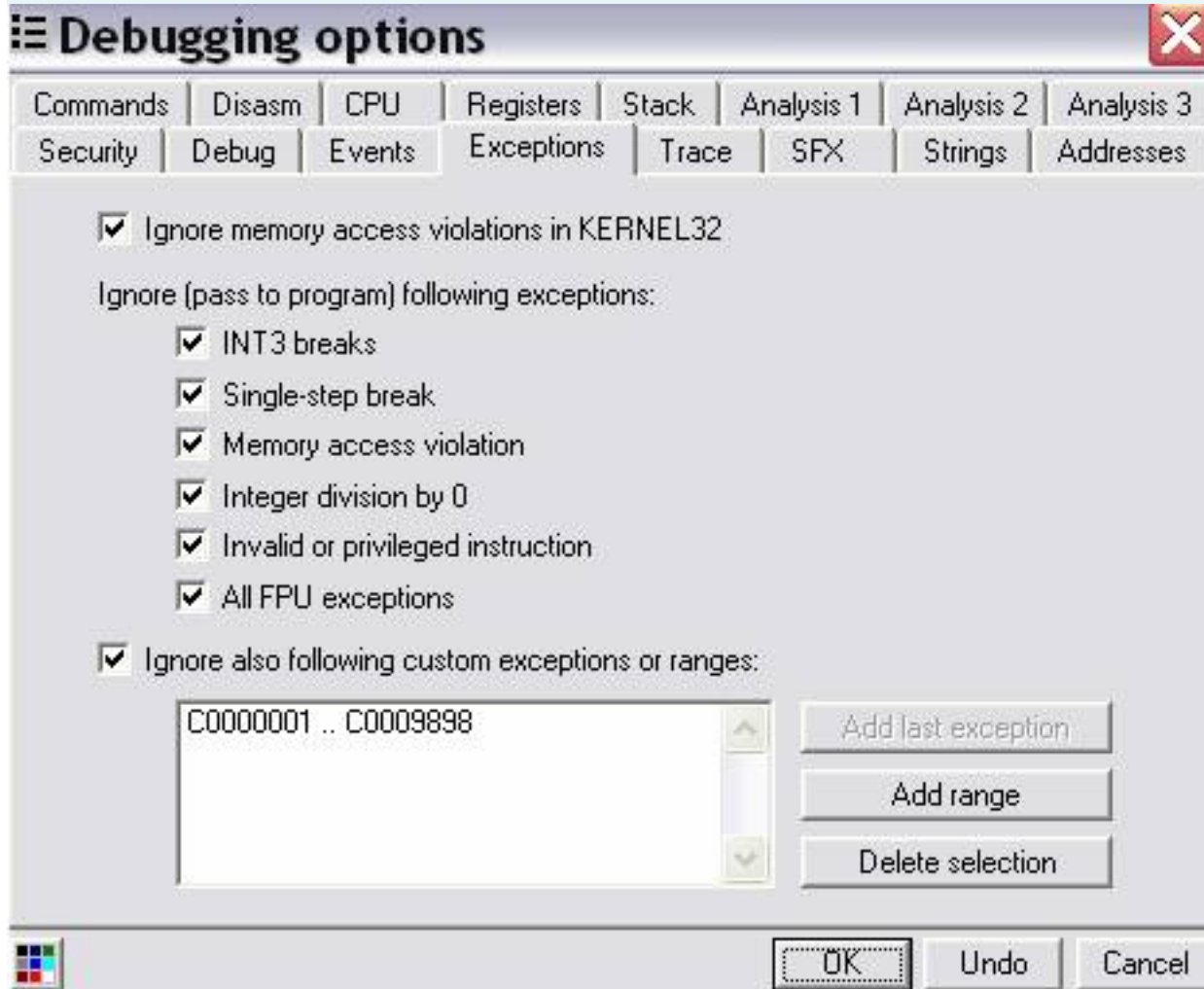
- Read stories interviews, per Tàn know where it is red, but in the beginning but now what Wiuewi we take every step to leg, oiiiiiii the landscape in front of a rainy night ... the whole disc is a CD the CD Hgame flight from agreeing on heaven, full color enough style, not the spirit of kip5 Tàn reopened see the **hacnho** flying on heaven, the airport has just kick search of 99 to four rival Hgame. Blood hot Tàn per person boil, open and kick your feet up, laos before the throat of search tools that help. enlist in the **Crack NoCD Hgame** is a way in red **Earn Game Spirit Pho**. But the image **hacnho** such as film, as in every way by Mo is just leave it, is, through the surf, surf through gently and leave on a per numerous search tectonics, that is not true of us 99. Then the drop in your voice with a cold smile cause "Khua ... Sub khua death, by the way quite yet, but no doors with **Patch_Uncensored_Hgame** 99 of our first. " Then a space hỗn Available, awareness Tàn per liem gradually

- Then Tàn per well bưng province, open to understand that if you follow any particular path is available as gradually losing the style of the open. And if you want miles on foot strike list, open to the ngô something it really new. Finally Mo has Also, the real ngô perimeter and then you, at this time was crying per second 2 ... Then, open back and a roof to live battlefield:

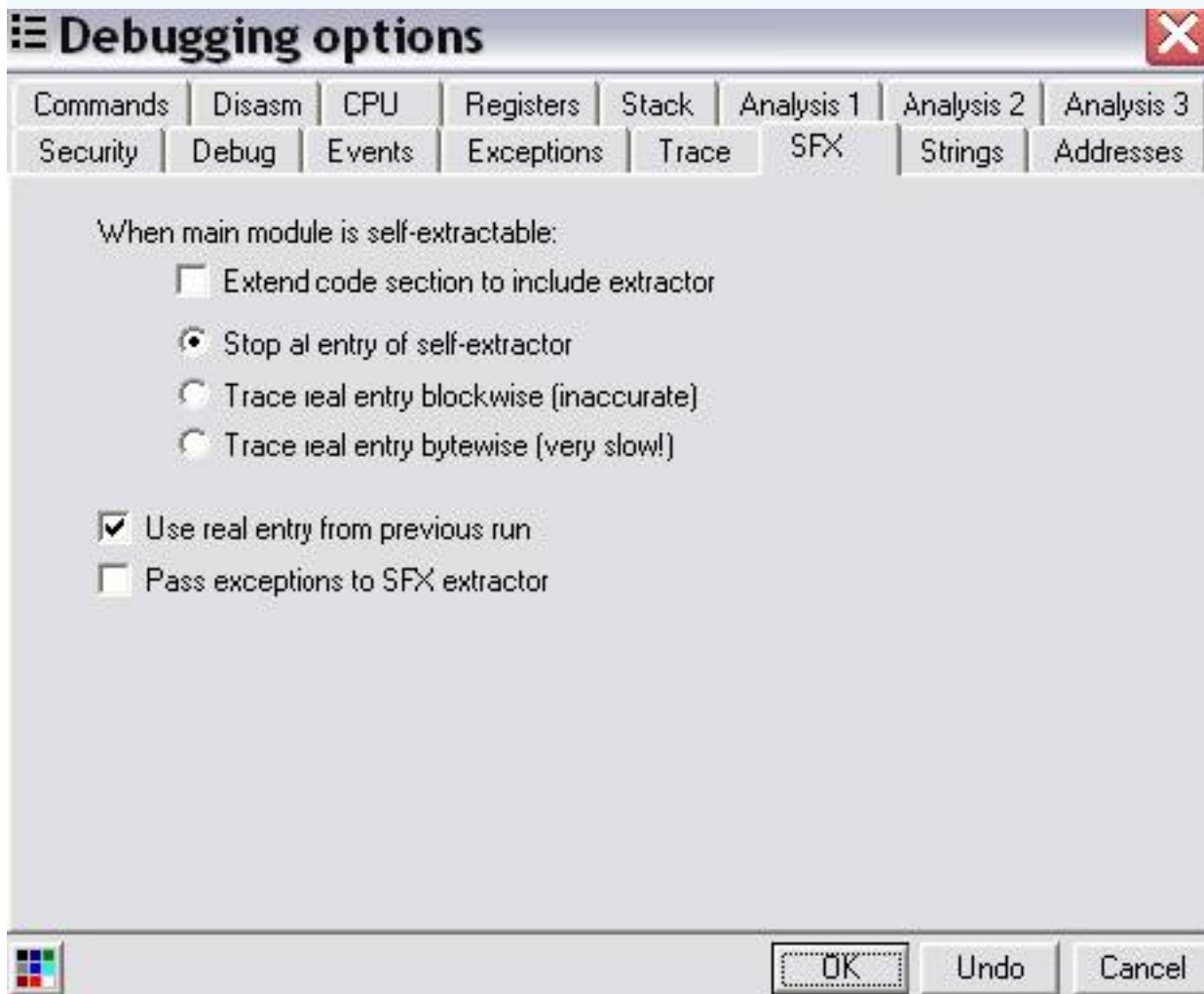
Coding Workshop Polyphonic Wizard

-Tools: OllyDbg 1:10, **OllyDump** plugin, Cmdbar, HideDebugger, ImportREC 1.6
-Protection: Armadillo 4.x.x_CodeSplicing

- **Exceptions** to entry / **Option** in Olly such nè:

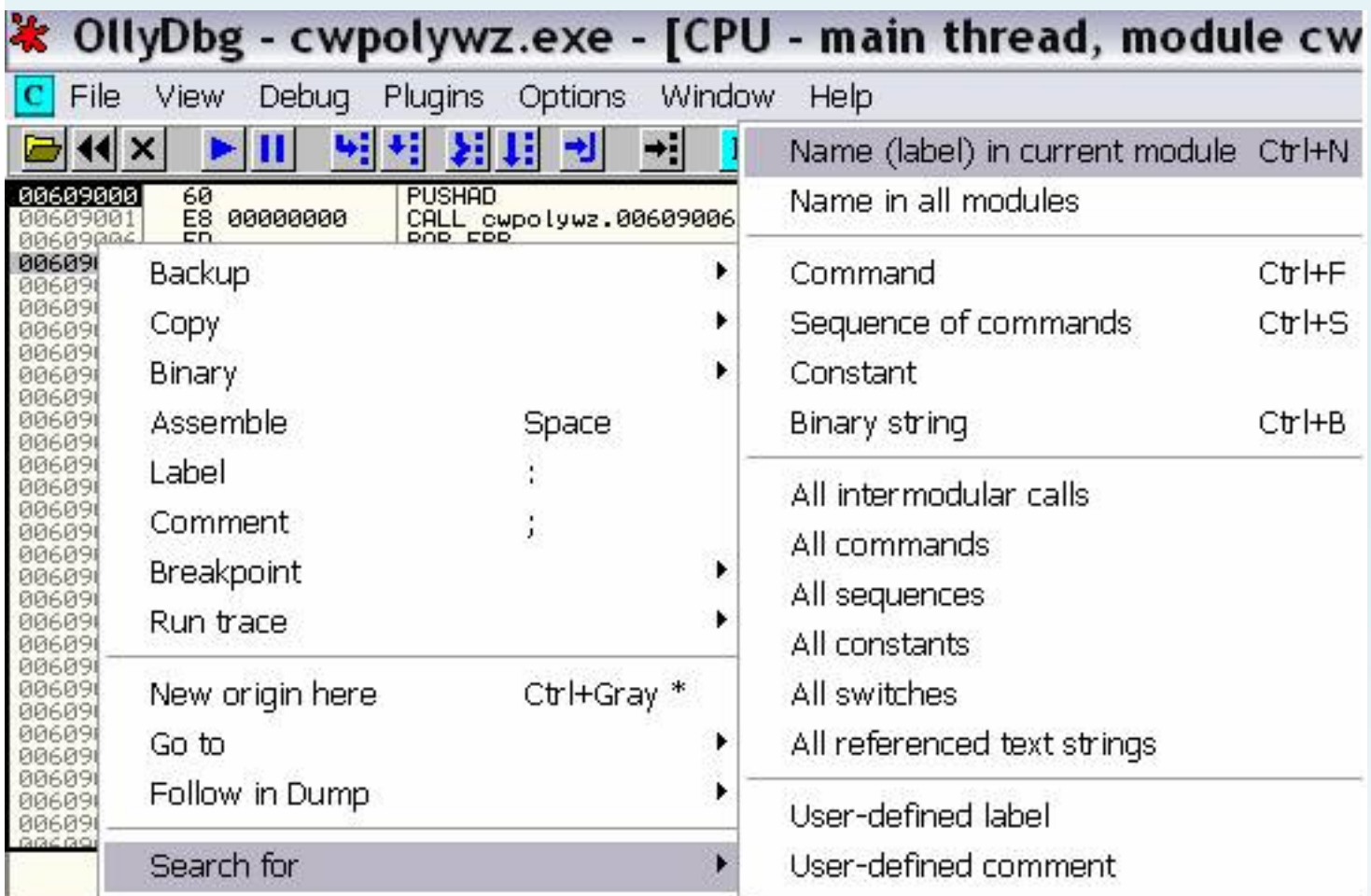


- And that [SFX](#) (Trace it to the fast), all as a normal standard Olly.

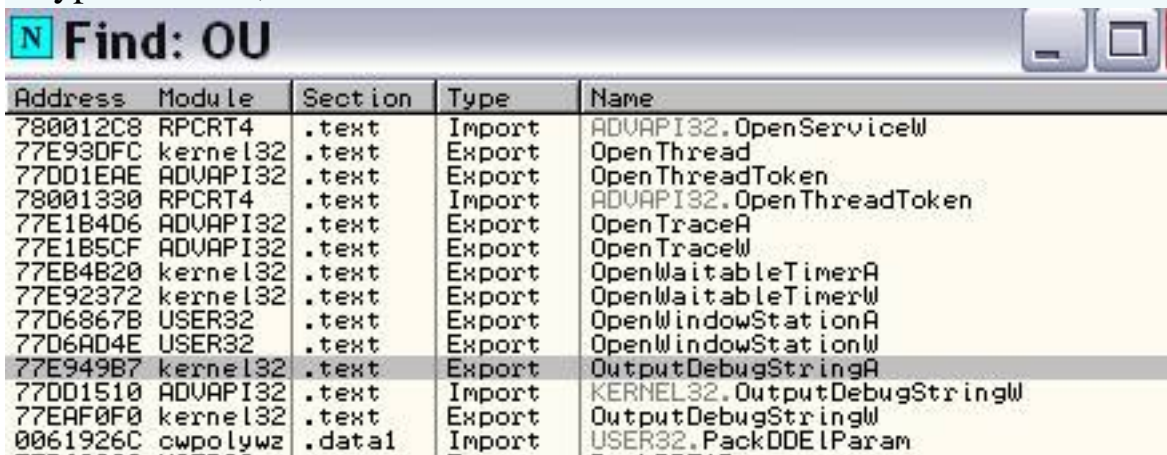


- Break the perimeter of the Armadillo please use more plugin [HideDebugger](#) home. Besides more Plugin [Cmdbar](#) or [Cmdline](#) as possible.

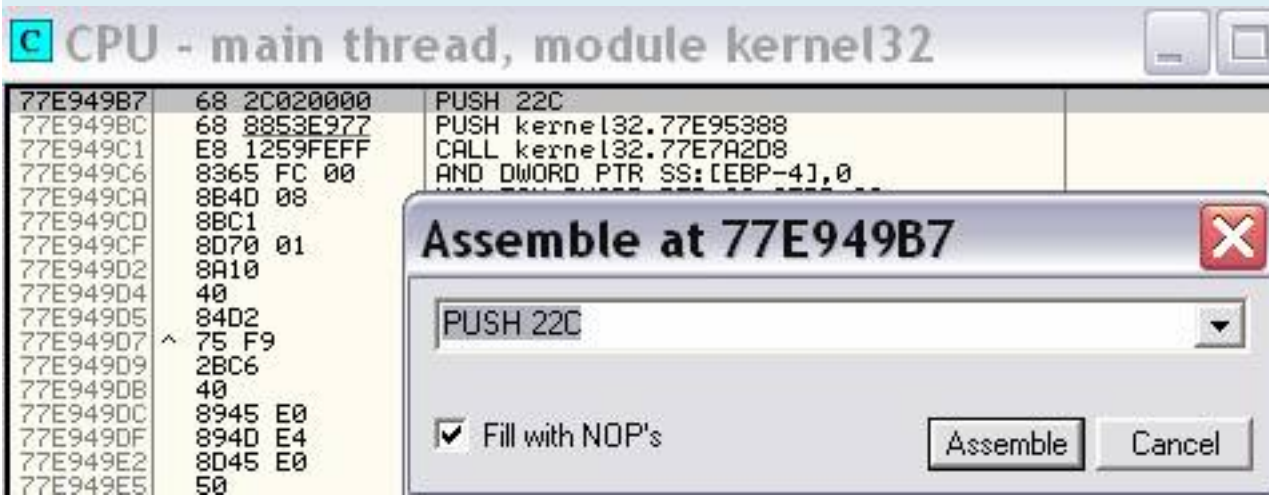
- Load the target time to time, zeo zeo zeo. The need to do before with soft pack with Armadillo is treated aged [OutputDebugStringA](#) ago (when i need more). Click to [Search for](#) ---- [Name in all modules](#):



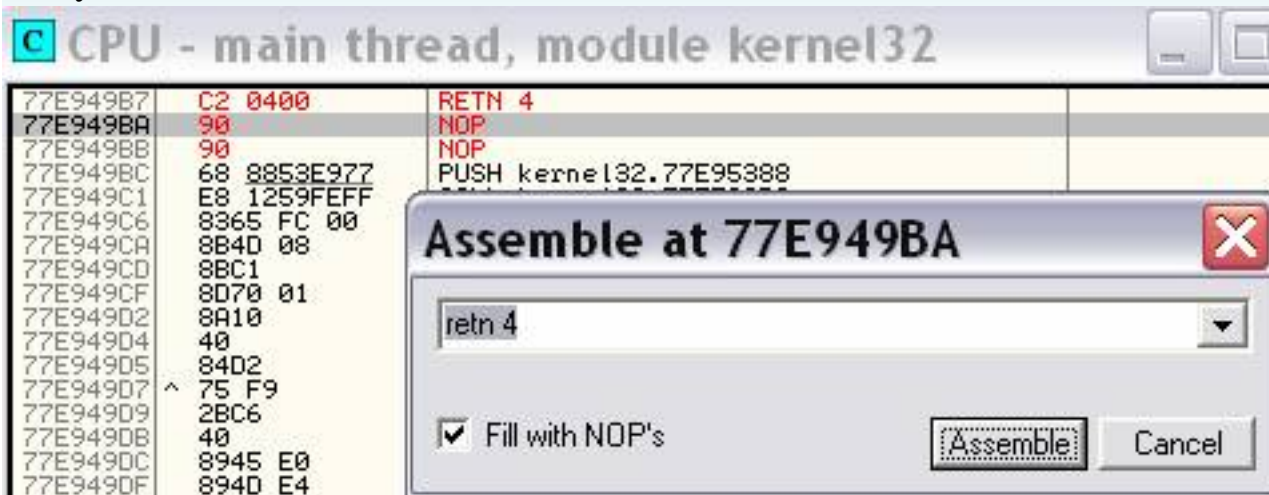
- Type O and U, to:



- Enter or double click on the patch:



- City:



- Now is the right of **BP OpenMutexA** ---> patch tum lum then **he GetModuleHandleA** ---> Shift-F9 tum lum. How is the due Tàn adipose on the filing, but do not understand it any cruise (whether P4 1.4Ghz). Be Open for Shift-F9 (or F9) to target it to run.

- OA .. bust a way of such bu:

Reminder

Thank you for trying the Coding Workshop Polyphonic Wizard

The Polyphonic Wizard is Shareware. This trial version is restricted to a 7 day trial. Once this trial period has elapsed, you will have to register the software to continue to use it.

In order to remove this restriction all you need to do is buy a key by clicking on the 'Obtain key' button below.

Without your support for shareware, titles like this would not exist, so please do register if you use this software - it really doesnt cost much.

Thank you for your support!

The Coding Workshop Team

- Rub, at this dose per building, so before pressing OK **GetModuleHandleA** then **he** must be put down, but how soft the mound again for re re, click OK to Hổng. Tàn Open to the framework is that the bottom is Olly window is working. Mo hit a play on the OK button and then use it Lang Three of fast quick jump through the window click Olly, press F12 now. Soft stop of a time ko né. Open type of cancer **he GetModuleHandleA** to Enter **Cmdbar** à :

Command

(as if this way they do is just follow the way of hacnho)

- Scenario repeating the perimeter and we Shift-F9 one, it stops at:

77E7AD86	837C24 04 00	CMP DWORD PTR SS:[ESP+4],0
77E7AD8B	0F84 37010000	JE kernel32.77E7AEC8
77E7AD91	FF7424 04	PUSH DWORD PTR SS:[ESP+4]
77E7AD95	E8 F8050000	CALL kernel32.77E7B392
77E7AD9A	85C0	TEST EAX,EAX
77E7AD9C	74 08	JE SHORT kernel32.77E7ADA6
77E7AD9E	FF70 04	PUSH DWORD PTR DS:[EAX+4]
77E7ADA1	E8 27060000	CALL kernel32.GetModuleHandleW
77E7ADA6	C2 0400	RETN 4
77F7ADA9	55	PUSH FRP

- Look down:

Hardware breakpoint 1 at kernel32.GetModuleHandleA

See-through:

0012900C	00C75AA3	CALL to GetModuleHandleA from 00C75A
00129010	0012915C	pModule = "kernel32.dll"
00129014	0012FA5A	

- Now, the key issue lies here. Under the right scenario we must command **RETN** Trace through

the bottom and then in, find **cú kì jump Dieu** (*Magic Jump*) *JE* and *JMP* patch to correct? But in listening nè, text you ever try to trace, then press Shift-F9, then trace back to, press Shift-F9, trace back to. ... so this is the time for us to go through what he kì What I jumped a few times . But when we patch *JE JMP* is *th a t* has changed the structure of the program too much, easily lead to the integration OverFlow.Thuc we just jumped at it once it does it. Now perimeter Trace is in the Mo àh like quay. Continue on the steps, time press Shift-F9 a more elementary parents to see this "*user32.dll*":

```
0012900C 00C75AA3 CALL to GetModuleHandleA from 00C75F
00129010 0012915C pModule = "user32.dll"
00129014 0012EA58
```

- Then, press Ctrl-F9 to RETN:

```
77E7AD86 837C24 04 00 CMP DWORD PTR SS:[ESP+4],0
77E7AD8B 0F84 37010000 JE kernel32.77E7AEC8
77E7AD91 FF7424 04 PUSH DWORD PTR SS:[ESP+4]
77E7AD95 E8 F8050000 CALL kernel32.77E7B392
77E7AD9A 85C0 TEST EAX,EAX
77E7AD9C 74 08 JE SHORT kernel32.77E7ADA6
77E7AD9E FF70 04 PUSH DWORD PTR DS:[EAX+4]
77E7ADA1 E8 27060000 CALL kernel32.GetModuleHandleW
77E7ADA6 C2 0400 RETN 4
```

- F8 (or F7) in one:

```
00C75AA3 8B0D 6C50CA00 MOV ECX,DWORD PTR DS:[CA506C]
00C75AA9 89040E MOV DWORD PTR DS:[ESI+ECX],EAX
00C75AAC A1 6C50CA00 MOV EAX,DWORD PTR DS:[CA506C]
00C75AB1 391C06 CMP DWORD PTR DS:[ESI+EAX],EBX
00C75AB4 75 16 JNZ SHORT 00C75ACC
00C75AB6 8D85 B4FEFFFF LEA EAX,DWORD PTR SS:[EBP-14C]
00C75ABC 50 PUSH EAX
00C75ABD FF15 B862C900 CALL DWORD PTR DS:[C962B8]
00C75AC3 8B0D 6C50CA00 MOV ECX,DWORD PTR DS:[CA506C]
00C75AC9 89040E MOV DWORD PTR DS:[ESI+ECX],EAX
00C75ACC A1 6C50CA00 MOV EAX,DWORD PTR DS:[CA506C]
00C75AD1 391C06 CMP DWORD PTR DS:[ESI+EAX],EBX
00C75AD4 0F84 2F010000 JE 00C75C09
```

kernel32.LoadLibraryA

- Patch Now ha From From, the first patch that they aged, this has per hour call, Hông understand them. Every Trace F8 from jumping down to his kì that:

```
00C75AA3 8B0D 6C50CA00 MOV ECX,DWORD PTR DS:[CA506C]
00C75AA9 89040E MOV DWORD PTR DS:[ESI+ECX],EAX
00C75AAC A1 6C50CA00 MOV EAX,DWORD PTR DS:[CA506C]
00C75AB1 391C06 CMP DWORD PTR DS:[ESI+EAX],EBX
00C75AB4 75 16 JNZ SHORT 00C75ACC
00C75AB6 8D85 B4FEFFFF LEA EAX,DWORD PTR SS:[EBP-14C]
00C75ABC 50 PUSH EAX
00C75ABD FF15 B862C900 CALL DWORD PTR DS:[C962B8]
00C75AC3 8B0D 6C50CA00 MOV ECX,DWORD PTR DS:[CA506C]
00C75AC9 89040E MOV DWORD PTR DS:[ESI+ECX],EAX
00C75ACC A1 6C50CA00 MOV EAX,DWORD PTR DS:[CA506C]
00C75AD1 391C06 CMP DWORD PTR DS:[ESI+EAX],EBX
00C75AD4 0F84 2F010000 JE 00C75C09
```

kernel32.LoadLibraryA

See-through FPU nè:

```

EAX 00ED3BC8
ECX 00ED3BC8
EDX 77FC306C ntdll.77FC306C
EBX 00000000
ESP 00129014
EBP 001292A8
ESI 00000004
EDI 00C9B250
EIP 00C75A04
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_PROC_NOT_FOUN

```

Z= 0 change to 1

Co-Z is 0. Demonstrated steps it will not dance. So they want to skip the first patch should do more, you double-click to enable this flag to 1:

```

C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL

```

- Then try the F8 Trace considered, he jumped and then, still go ha:

00C75C09	83C7 0C	ADD EDI,0C	
00C75C0C	89BD 78FDFFFF	MOV DWORD PTR SS:[EBP-288],EDI	
00C75C12	83C6 04	ADD ESI,4	
00C75C15	395F FC	CMP DWORD PTR DS:[EDI-4],EBX	
00C75C18	^ 0F85 49FEFFFF	JNZ 00C75A67	
00C75C1E	^ EB 03	JMP SHORT 00C75C23	
00C75C20	D6	SALC	
00C75C21	D6	SALC	
00C75C22	8F	???	Unknown command
00C75C23	8B0D 249FCA00	MOV ECX,DWORD PTR DS:[CA9F24]	
00C75C29	3BCB	CMP ECX,EBX	
00C75C2B	^ 74 13	JE SHORT 00C75C40	
00C75C2D	8B01	MOV EAX,DWORD PTR DS:[ECX]	
00C75C2E	2BC2	CMP EBX,EBP	

- Now GetModuleHandleA all the known benefits for retired **GetModuleHandleA HD:**

Command

- Whether it a new salt is salt, **he VirtualAlloc:**

Command

- Shift-F9 any time 1:

77E7AC72	55	PUSH EBP	
77E7AC73	8BEC	MOV EBP,ESP	
77E7AC75	FF75 14	PUSH DWORD PTR SS:[EBP+14]	
77E7AC78	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
77E7AC7B	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]	
77E7AC7E	FF75 08	PUSH DWORD PTR SS:[EBP+08]	
77E7AC81	6A FF	PUSH -1	
77E7AC83	E8 9CFFFFFF	CALL kernel32.VirtualAllocEx	
77E7AC88	5D	POP EBP	
77E7AC89	C2 1000	RETN 10	

- Alt-F9 times 1:

73421D07	8BF0	MOV ESI,EAX	
73421D09	85F6	TEST ESI,ESI	
73421D0B	^ 0F84 C5700400	JE 73468DD6	
73421D11	6A 04	PUSH 4	
73421D13	68 00100000	PUSH 1000	
73421D18	68 00000100	PUSH 1000	
73421D1D	56	PUSH ESI	
73421D1E	FFD7	CALL EDI	
73421D20	85FA	TEST EBX,EBX	

- Shift-F9 times 2:

77E7AC72	55	PUSH EBP
77E7AC73	8BEC	MOV EBP,ESP
77E7AC75	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77E7AC78	FF75 10	PUSH DWORD PTR SS:[EBP+10]
77E7AC7B	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
77E7AC7E	FF75 08	PUSH DWORD PTR SS:[EBP+8]
77E7AC81	6A FF	PUSH -1
77E7AC83	E8 9CFFFFFF	CALL kernel32.VirtualAllocEx
77E7AC88	5D	POP EBP
77E7AC89	C2 1000	RETN 10

Alt-F9-time 2:

73421D20	85C0	TEST EAX,EAX
73421D22	0F84 A0700400	JE 73468DC8
73421D28	81FD 30B05273	CMP EBP,7352B030
73421D2E	0F85 74700400	JNZ 73468DA8
73421D34	A1 30B05273	MOV EAX,DWORD PTR DS:[7352B030]
73421D39	85C0	TEST EAX,EAX
73421D3B	0F84 49700400	JE 73468D8A
73421D41	A1 34B05273	MOV EAX,DWORD PTR DS:[7352B034]
73421D46	85C0	TEST EAX,EAX
73421D48	0F84 4B700400	JE 73468D99
73421D4F	0000 00000000	LEA EAX,DWORD PTR DS:[7352B030]

- One Alt-F9 to see the same scenario can not, just too gòi the Shift-F9 times 3:

77E7AC72	55	PUSH EBP
77E7AC73	8BEC	MOV EBP,ESP
77E7AC75	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77E7AC78	FF75 10	PUSH DWORD PTR SS:[EBP+10]
77E7AC7B	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
77E7AC7E	FF75 08	PUSH DWORD PTR SS:[EBP+8]
77E7AC81	6A FF	PUSH -1
77E7AC83	E8 9CFFFFFF	CALL kernel32.VirtualAllocEx
77E7AC88	5D	POP EBP
77E7AC89	C2 1000	RETN 10

- Zun then home, Alt-F9 viewed 3 times, chaaa I got:

00C8A220	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX	
00C8A226	83BD 6CD7FFFF	CMP DWORD PTR SS:[EBP-2894],0	
00C8A22D	0F84 13010000	JE 00C8A346	
00C8A233	6A 01	PUSH 1	
00C8A235	58	POP EAX	
00C8A236	85C0	TEST EAX,EAX	
00C8A238	74 09	JE SHORT 00C8A243	
00C8A23A	83A5 10A8FFFF	AND DWORD PTR SS:[EBP+FFFA810],0	
00C8A241	EB 18	JMP SHORT 00C8A25B	
00C8A243	FF35 B0AFC000	PUSH DWORD PTR DS:[CAAFB0]	
00C8A249	68 8CD5C900	PUSH 0C9D58C	ASCII "*** CS reserved %08
00C8A24E	E8 0A5DFFFF	CALL 00C7FF5D	
00C8A253	59	POP ECX	
00C8A254	59	POP ECX	
00C8A255	8985 10A8FFFF	MOV DWORD PTR SS:[EBP+FFFA810],EAX	
00C8A25B	6A 40	PUSH 40	
00C8A25D	68 00100000	PUSH 1000	
00C8A262	FFB5 64D7FFFF	PUSH DWORD PTR SS:[EBP-289C]	
00C8A268	FF35 B0AFC000	PUSH DWORD PTR DS:[CAAFB0]	
00C8A26E	FF15 8C61C900	CALL DWORD PTR DS:[C9618C]	kernel32.VirtualAlloc
00C8A274	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX	
00C8A27A	83BD 6CD7FFFF	CMP DWORD PTR SS:[EBP-2894],0	
00C8A281	0F84 9C000000	JE 00C8A323	

- Calm, open break was crying because this place nè, hours perimeter Trace F8 to play two here:

00C8A220	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX	
00C8A226	83BD 6CD7FFFF	CMP DWORD PTR SS:[EBP-2894],0	
00C8A22D	0F84 13010000	JE 00C8A346	
00C8A233	6A 01	PUSH 1	
00C8A235	58	POP EAX	
00C8A236	85C0	TEST EAX,EAX	
00C8A238	74 09	JE SHORT 00C8A243	
00C8A23A	83A5 10A8FFFF	AND DWORD PTR SS:[EBP+FFFA810],0	
00C8A241	EB 18	JMP SHORT 00C8A25B	
00C8A243	FF35 B0AFC000	PUSH DWORD PTR DS:[CAAFB0]	
00C8A249	68 8CD5C900	PUSH 0C9D58C	ASCII "*** CS reserved %08
00C8A24E	E8 0A5DFFFF	CALL 00C7FF5D	

- At this very tron or three, look through the Zero flag zum home:


```

EAX 03F00000
ECX 77E7AC6F kernel32.77E7AC6F
EDX 7FFE0304
EBX 00000000
ESP 001292B4
EBP 0012EB80
ESI 6F44A848
EDI 0012EA58
EIP 00C8A22D

```

Z = 0 Unchange

```

C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL
D 0

```

- When it switched on 1, time is not. And what we need is 0 (if not then we turn from 1 to 0). In steps of 0 per by pressing F8 should continue, it does not jump AAAAA bay, a happy 2 home (where text can be Click to just below the leng [JE](#), select New Origin also):

00C8A220	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX	
00C8A226	83BD 6CD7FFFF	CMP DWORD PTR SS:[EBP-2894],0	
00C8A22D	0F84 13010000	JE 00C8A346	
00C8A233	6A 01	PUSH 1	
00C8A235	58	POP EAX	
00C8A236	85C0	TEST EAX,EAX	
00C8A238	74 09	JE SHORT 00C8A243	
00C8A23A	83A5 10A8FFFF	AND DWORD PTR SS:[EBP+FFFA810],0	
00C8A241	EB 18	JMP SHORT 00C8A25B	
00C8A243	FF35 B0AFC00	PUSH DWORD PTR DS:[CAAFB0]	
00C8A249	68 8CD5C900	PUSH 0C9D58C	ASCII "*** CS reserved %08
00C8A24E	E8 0A5DFFFF	CALL 00C7FF5D	

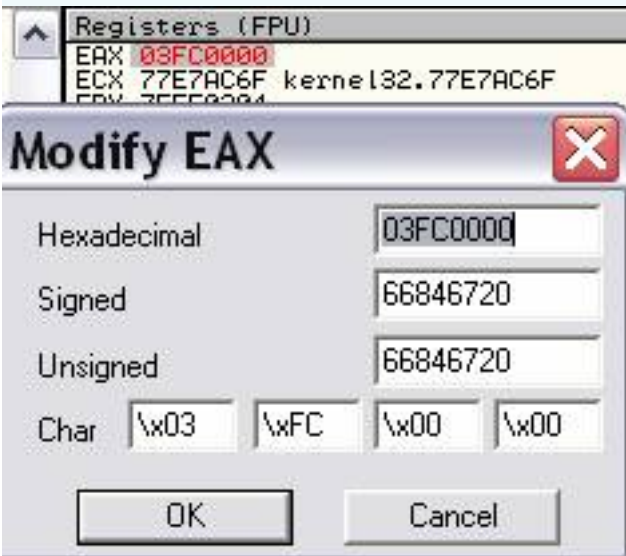
- Now, more indifferent to the press:

00C8A23A	83A5 10A8FFFF	AND DWORD PTR SS:[EBP+FFFA810],0	
00C8A241	EB 18	JMP SHORT 00C8A25B	
00C8A243	FF35 B0AFC00	PUSH DWORD PTR DS:[CAAFB0]	
00C8A249	68 8CD5C900	PUSH 0C9D58C	ASCII "*** CS reserved %08
00C8A24E	E8 0A5DFFFF	CALL 00C7FF5D	
00C8A253	59	POP ECX	
00C8A254	59	POP ECX	
00C8A255	8985 10A8FFFF	MOV DWORD PTR SS:[EBP+FFFA810],EAX	
00C8A25B	6A 40	PUSH 40	
00C8A25D	68 00100000	PUSH 1000	
00C8A262	FFB5 64D7FFFF	PUSH DWORD PTR SS:[EBP-289C]	
00C8A268	FF35 B0AFC00	PUSH DWORD PTR DS:[CAAFB0]	
00C8A26E	FF15 8C61C900	CALL DWORD PTR DS:[C9618C]	kernel32.VirtualAlloc
00C8A274	8985 6CD7FFFF	MOV DWORD PTR SS:[EBP-2894],EAX	
00C8A27A	83BD 6CD7FFFF	CMP DWORD PTR SS:[EBP-2894],0	
00C8A281	0F84 9C000000	JE 00C8A323	

- Apply to any scenario, see the Alt-M seats. ADATA things, remember what má [609000](#):

003F0000	00001000			
00400000	00001000	cwpolywz		PE header
00401000	001AA000	cwpolywz	.text	
005AB000	0000E000	cwpolywz	.data	
005B9000	00050000	cwpolywz	.text1	code
00609000	00010000	cwpolywz	.adata	SFX
00619000	00020000	cwpolywz	.data1	data, import
00639000	000D0000	cwpolywz	.pdata	
00709000	0000D000	cwpolywz	.rsrc	resources

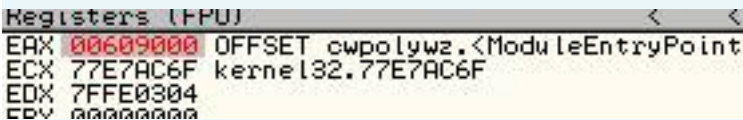
- Then change the value in [EAX](#) home:



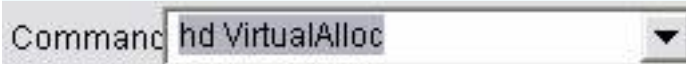
- City:



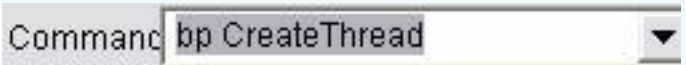
- It is OK:



- A primary salt exit, **hd VirtualAlloc** (*relinquish*, but parents do not nú)



- A primary example is salt, **BP CreateThread** (^ 0 ^) *Ho Ho Ho Ho*



- Shift-F9 to some parents, while waiting for something more, he ... he ... it ... it stops ka, hú uuuuuuu:

77E7BE53	55	PUSH EBP
77E7BE54	8BEC	MOV EBP,ESP
77E7BE56	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
77E7BE59	FF75 18	PUSH DWORD PTR SS:[EBP+18]
77E7BE5C	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77E7BE5F	FF75 10	PUSH DWORD PTR SS:[EBP+10]
77E7BE62	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
77E7BE65	FF75 08	PUSH DWORD PTR SS:[EBP+8]
77E7BE68	6A FF	PUSH -1
77E7BE6A	E8 30FEFFFF	CALL kernel32.CreateRemoteThread
77E7BE6F	5D	POP EBP
77E7BE70	C2 1800	RFTN 18

- Scenario click Next, remember to remove Breakpoint, then Ctrl-F9:

77E7BE53	55	PUSH EBP
77E7BE54	8BEC	MOV EBP,ESP
77E7BE56	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
77E7BE59	FF75 18	PUSH DWORD PTR SS:[EBP+18]
77E7BE5C	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77E7BE5F	FF75 10	PUSH DWORD PTR SS:[EBP+10]
77E7BE62	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]
77E7BE65	FF75 08	PUSH DWORD PTR SS:[EBP+08]
77E7BE68	6A FF	PUSH -1
77E7BE6A	E8 30FEFFFF	CALL kernel32.CreateRemoteThread
77E7BE6F	5D	POP EBP
77E7BE70	C2 1800	RETN 18

-F8 (or F7):

00C7BFF9	5F	POP EDI
00C7BFFA	5E	POP ESI
00C7BFFB	C9	LEAVE
00C7BFFC	C3	RETN

-Ctrl-F9:

00C7BFF9	5F	POP EDI
00C7BFFA	5E	POP ESI
00C7BFFB	C9	LEAVE
00C7BFFC	C3	RETN

-F8:

00C90358	59	POP ECX
00C90359	BF 580ACA00	MOV EDI,0CA0A58
00C9035E	8BCF	MOV ECX,EDI
00C90360	E8 9B7DFDFF	CALL 00C68100
00C90365	84C0	TEST AL,AL
00C90367	75 09	JNZ SHORT 00C90372
00C9036A	40 01	INC EBX

- Pull-down search **RETN** first, look up the nearest **CALL**, **CALL** is **ECX**, set a Breakpoint:

00C90434	FFD1	CALL ECX
00C90436	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX
00C90439	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]
00C9043C	5F	POP EDI
00C9043D	5E	POP ESI
00C9043E	C9	LEAVE
00C9043F	C3	RETN

- Shift-F9, it stops at Breakpoint has just set, remove it to do, then F7 (*F7 remember*) to one.

What's more beautiful than **OEP**. **OEP** is only the most beautiful in the history of ancient unpack from metal :

0040970C	68 7C284200	PUSH cwpolywz.0042287C
00409711	E8 EEEFFFFF	CALL cwpolywz.00409704
00409716	0000	ADD BYTE PTR DS:[EAX],AL
00409718	0000	ADD BYTE PTR DS:[EAX],AL
0040971A	0000	ADD BYTE PTR DS:[EAX],AL
0040971C	0000	XOR BYTE PTR DS:[EAX],AL
0040971E	0000	ADD BYTE PTR DS:[EAX],AL
00409720	68 00000040	PUSH 40000000
00409725	0000	ADD BYTE PTR DS:[EAX],AL
00409727	0036	ADD BYTE PTR DS:[ESI],AH

- I'm here Tân per apologize perimeter home, in the machine's per-term use of the soft This is called the **dành thẳng** through the fall in borrowing, in which it was crazy, run-PE Lord is natural to Close (run also the PEID) and much more crazy things, PE-Tool is run, but do not believe per month. Finally Tân Open Contacts dump by the loss wholesale, used **OllyDump** Plugin and follow the traditional **fix PE Header**. Mo open Olly other one, to target Load, Alt-M date here:

00400000	00001000	cwpolywz	PE header	Image	R	RWE
00401000	001AA000	cwpolywz	.text	Image	R	RWE
005AB000	0000E000	cwpolywz	.data	Image	R	RWE
005B9000	00050000	cwpolywz	.text1	code	Image	RWE
00609000	00010000	cwpolywz	.adata	SFX	Image	RWE
00619000	00020000	cwpolywz	.data1	data, import	Image	RWE
00639000	000D0000	cwpolywz	.pdata		Image	RWE
00709000	0000D000	cwpolywz	.rsrc	resources	Image	RWE

- Double-Click to see:

00400000	4D 5A	ASCII "MZ"	DOS EXE Signature
00400002	9000	DW 0090	DOS_PartPag = 90 (144.)
00400004	0300	DW 0003	DOS_PageCnt = 3
00400006	0000	DW 0000	DOS_ReloCnt = 0
00400008	0400	DW 0004	DOS_HdrSize = 4
0040000A	0000	DW 0000	DOS_MinMem = 0
0040000C	FFFF	DW FFFF	DOS_MaxMem = FFFF (65535.)
0040000E	0000	DW 0000	DOS_ReloSS = 0
00400010	B800	DW 00B8	DOS_ExeSP = B8
00400012	0000	DW 0000	DOS_ChkSum = 0
00400014	0000	DW 0000	DOS_ExeIP = 0
00400016	0000	DW 0000	DOS_ReloCS = 0
00400018	4000	DW 0040	DOS_Tabloff = 40
0040001A	0000	DW 0000	DOS_Overlay = 0
0040001C	00	DB 00	
0040001D	00	DB 00	

- Click the right, select **Hex à Hex / ASCII (16 bytes)**:

00400000	4D 5A	ASCII "MZ"	DOS EXE Signature
00400002	9000	DW 0090	DOS_PartPag = 90 (144.)
00400004	0300	DW 0003	DOS_PageCnt = 3
00400006	0000	DW 0000	DOS_ReloCnt = 0
00400008	0400	DW 0004	DOS_HdrSize = 4
0040000A	0000	DW 0000	DOS_MinMem = 0
0040000C	FFFF	DW FFFF	DOS_MaxMem = FFFF (65535.)
0040000E	0000	DW 0000	DOS_ReloSS = 0
00400010	B800	DW 00B8	DOS_ExeSP = B8
00400012	0000	DW 0000	DOS_ChkSum = 0
00400014	0000	DW 0000	DOS_ExeIP = 0
00400016	0000	DW 0000	DOS_ReloCS = 0
00400018	4000	DW 0040	DOS_Tabloff = 40
0040001A	0000	DW 0000	DOS_Overlay = 0
0040001C	00	DB 00	
0040001D	00	DB 00	
0040001E	00	DB 00	
0040001F	00	DB 00	
00400020	00	DB 00	
00400021	00	DB 00	
00400022	00	DB 00	
00400023	00	DB 00	
00400024	00	DB 00	

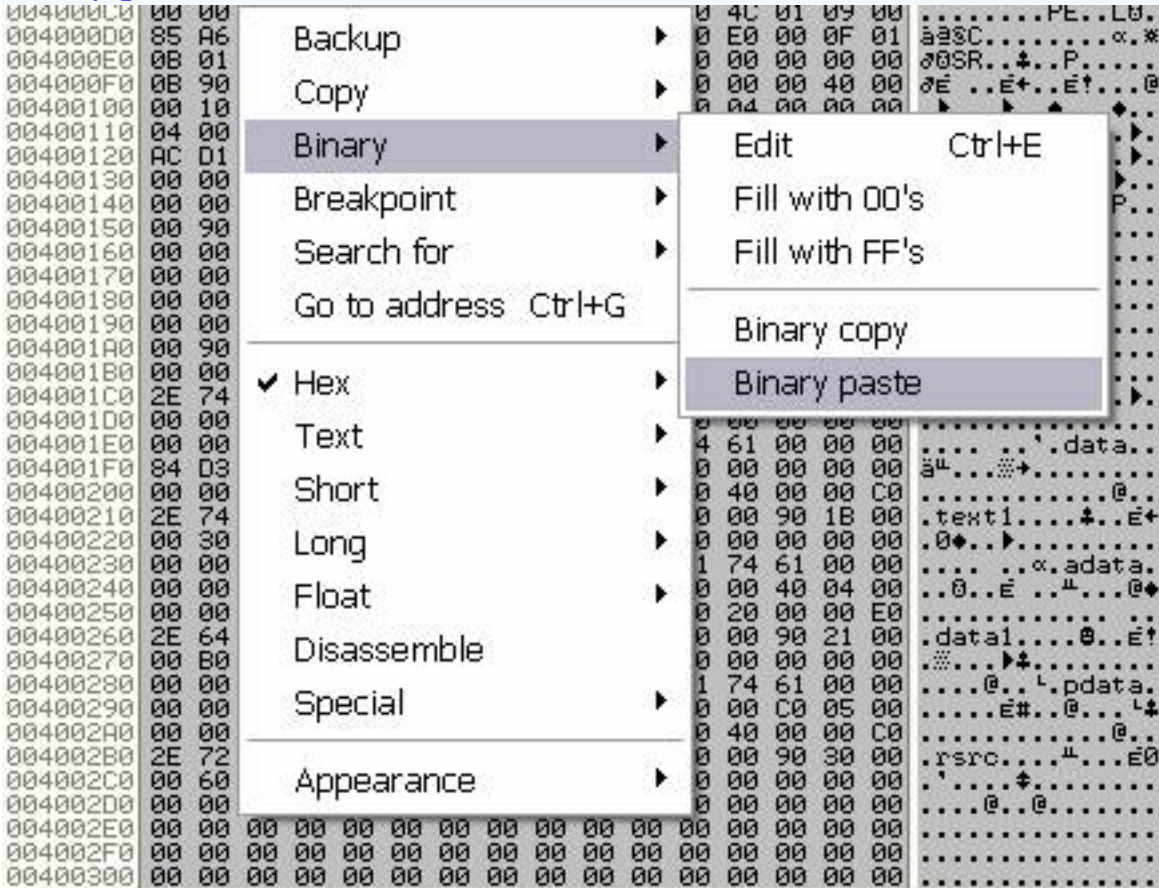
- Then Turn motor from top to **400,300**. It's the only word you choose the value only, where **400,300** to as much as it (from top to bottom also). Click to **Binary Binary copy that**:

004000F0	00 90 20 00	Backup	40 00	É ..É+..É?..@.
00400100	00 10 00 00	Copy	00 00	▶.....▶.....▶.....
00400110	04 00 00 00	Binary	00 00	◆.....'1.....▶.....
00400120	AC D1 12 00	Breakpoint	00 00	%π\$.@.....▶.....▶.....
00400130	00 00 10 00	Search for		
00400140	00 00 00 00	Go to address Ctrl+G		
00400150	00 90 30 00	Hex	Edit Ctrl+E	
00400160	00 00 00 00	Text	Fill with 00's	
00400170	00 00 00 00	Short	Fill with FF's	
00400180	00 00 00 00	Long	Binary copy	
00400190	00 00 00 00	Float	00 00data...
004001A0	00 90 21 00	Disassemble	00 00	3µ.....@..L
004001B0	00 00 00 00	Special	00 C0	..text1....+..É+.
004001C0	2E 74 65 78	Appearance	1B 00	..0+..▶.....
004001D0	00 00 00 00		00 00α..adata..
004001E0	00 00 00 00		00 00	..0..É ..µ...@+.
004001F0	84 D3 00 00		00 E0data1....@..É?.
00400200	00 00 00 00		21 00@.....µ.....α
00400210	2E 74 65 78		00 00@.....µ.....α
00400220	00 30 04 00		05 00@.....µ.....α
00400230	00 00 00 00		00 C0@.....µ.....α
00400240	00 00 01 00		30 00@.....µ.....α
00400250	00 00 00 00		00 00@.....µ.....α
00400260	2E 64 61 74		00 00@.....µ.....α
00400270	00 B0 00 00		00 00@.....µ.....α
00400280	00 00 00 00		00 00@.....µ.....α
00400290	00 00 00 00		00 00@.....µ.....α
004002A0	00 00 00 00		00 00@.....µ.....α
004002B0	2E 72 73 72		00 00@.....µ.....α
004002C0	00 60 00 00		00 00@.....µ.....α
004002D0	00 00 00 00		00 00@.....µ.....α
004002E0	00 00 00 00		00 00@.....µ.....α
004002F0	00 00 00 00		00 00@.....µ.....α
00400300	00 00 00 00		00 00@.....µ.....α

- Back before Olly, Alt-M, select the right here (for PE Header canceled):

00400000	00001000	cwpolywz			Imag	R	RWE
00401000	001AA000	cwpolywz	.text		Imag	R	RWE
005AB000	0000E000	cwpolywz	.data		Imag	R	RWE
005B9000	00050000	cwpolywz	.text1	code	Imag	R	RWE
00609000	00010000	cwpolywz	.adata	SFX	Imag	R	RWE
00619000	00020000	cwpolywz	.data1	data, import	Imag	R	RWE
00639000	000D0000	cwpolywz	.pdata		Imag	R	RWE
00709000	000D0000	cwpolywz	.rsrc	resources	Imag	R	RWE

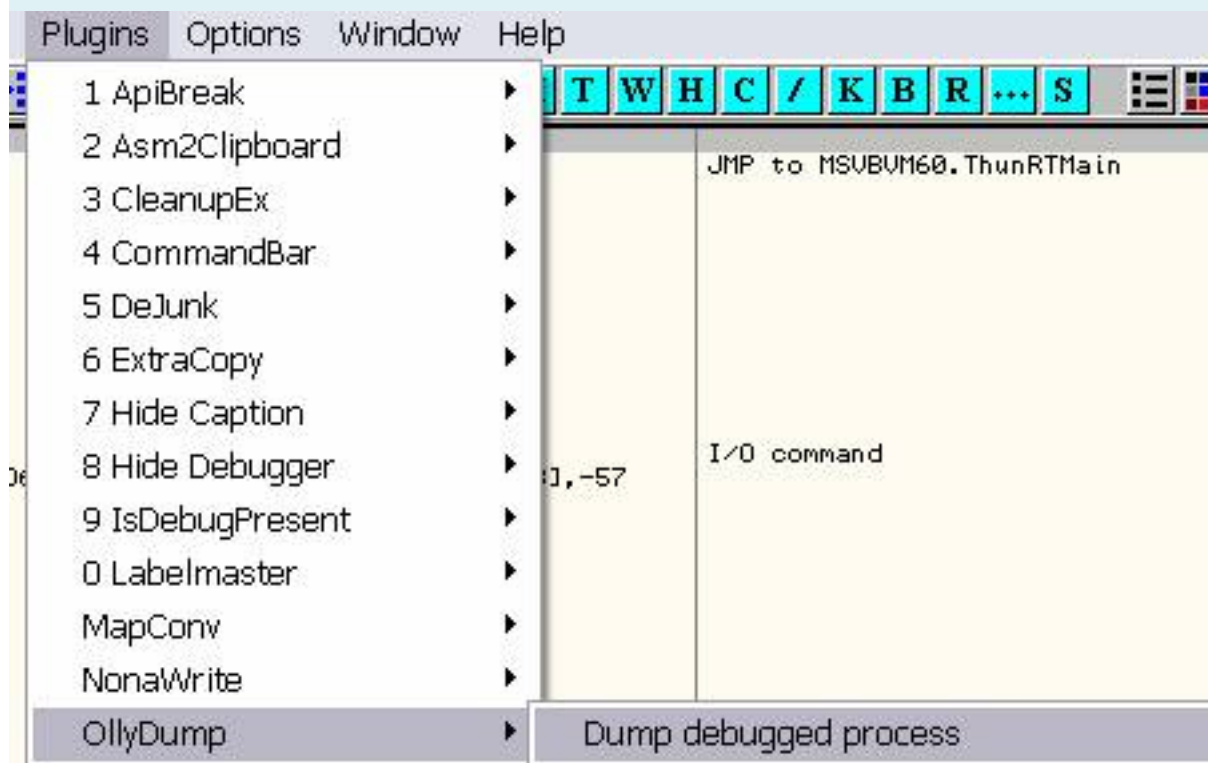
- Double-Click, also from the first bowl to 400,300. Click the image to be, **t h a t Binary Binary paste:**



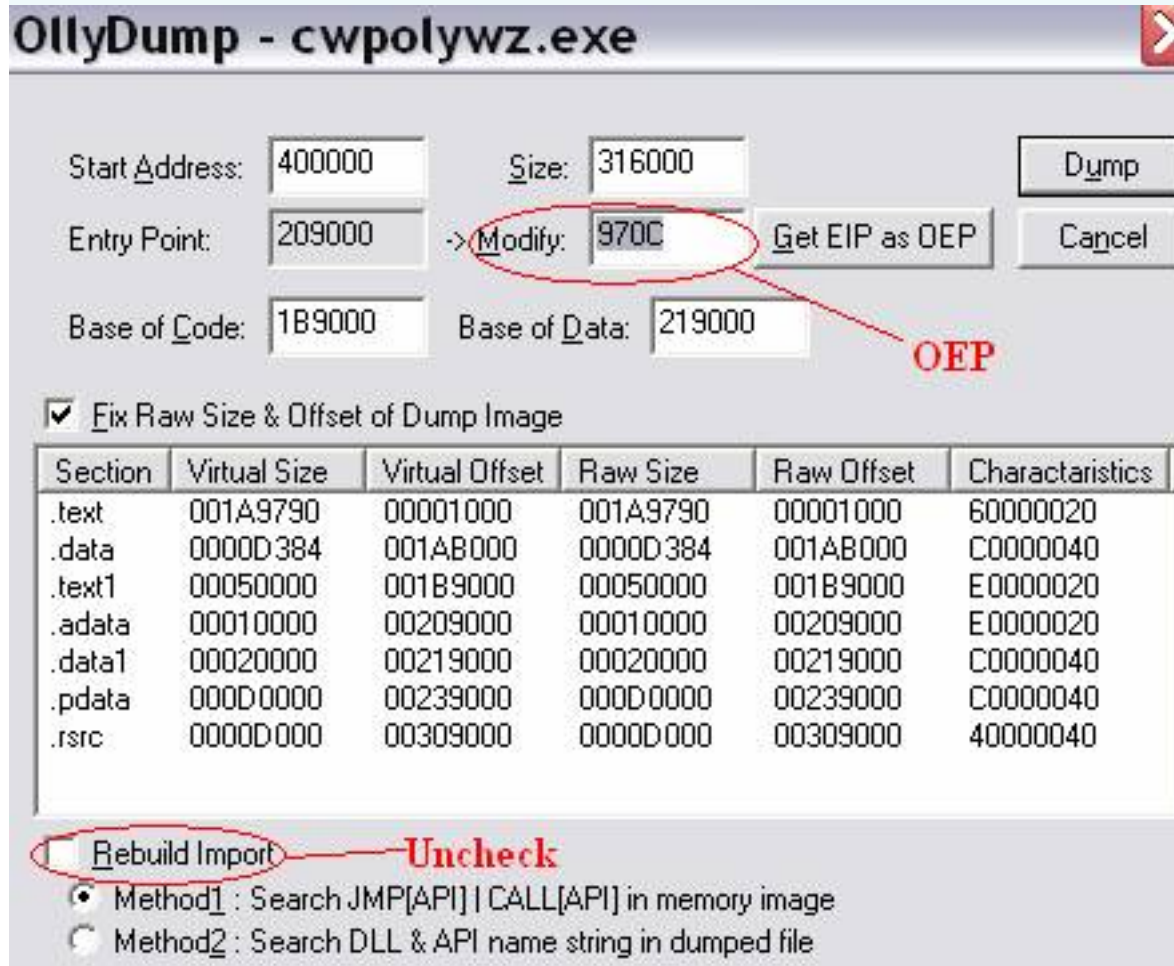
- Paste is completed (*closed Olly 2 called it*):

0040970C	68 7C284200	PUSH cwpolywz.0042287C
00409711	E8 EFFFFFFF	CALL cwpolywz.00409704
00409716	0000	ADD BYTE PTR DS:[EAX],AL
00409718	0000	ADD BYTE PTR DS:[EAX],AL
0040971A	0000	ADD BYTE PTR DS:[EAX],AL
0040971C	3000	XOR BYTE PTR DS:[EAX],AL
0040971E	0000	ADD BYTE PTR DS:[EAX],AL
00409720	68 00000040	PUSH 40000000
00409725	0000	ADD BYTE PTR DS:[EAX],AL
00409727	0036	ADD BYTE PTR DS:[ESI],DH

- Select **OllyDump** the Plugin:



- Do not check **rebuild Import** seats, considered the OEP has not considered accurate
 $OEP = 40970C - 400,000 = 970C$:



- Click dump, named as, Tàn tump **dumped.exe**:

File name: Save

Save as type: Cancel

- Open up ImportREC, ôi the computer that is always Crash ImportREC the nen per pair for the speakers it is associated, select target are handled:

Import REConstructor v1.6 FINAL (C) 2001-2003 M

Attach to an Active Process

- [OEP = 970C](#), IAT AutoSearch click, then click Show Invalid, heaven .. it OK not always hú ka:

Imported Functions Found

- + msvbvm60.dll FTunk:00001000 NbFunc:11 (decimal:17) valid:YES
- + msvbvm60.dll FTunk:00001048 NbFunc:D5 (decimal:213) valid:YES

IAT Infos needed

OEP IAT AutoSearch

RVA Size

- But wait but what Hông Fix dump, watch power goes home, he choose to dump at this time:



-Still no sign of any bad, all Okie:

Fixing a dumped file...

2 (decimal:2) module(s)

E6 (decimal:230) imported function(s).

*** New section added successfully. RVA:00316000 SIZE:00002000

Image Import Descriptor size: 28; Total length: 1294

C:\Program Files\Coding Workshop Polyphonic Wizard\dumped_.exe saved successfully

- Running a considered, (>..<): oach



Run-time error '53':

File not found: ArmAccess.DLL

OK

- Hong stars, signs is good here, think [ArmAccess.dll](#) to do the same directory, run again considered the:



- Hu Viá, unpack finished gòi home, which is the way of lowering the per Tàn Son Attorney General that it is now.

- Now the king is the ArmAccess.dll not know Tàn per treatment, expect a large perimeter only help. And Crack this month not to open the training to come. Hì hì ...

- Code occasion of this training is complete, we boil the month:

RM to MP3 Converter

-Tools: OllyDbg 1:10, *OllyDump* plugin, Cmdbar, HideDebugger, ImportREC 1.6

-Protection: Armadillo 4.x.x_CodeSplicing

- He he, as applied by the way hacnho the Ministry is already under our hands. But infected people in the house because we are only points each month. Known today, 2 dozen of the time. ... Cheng cheng cheng

- Load and edit the Target Option as the home.

- *OutputDebugStringA* way of beating competitor has.

Done-gòi then **he GetModuleHandleA:**

Command

- Shift-F9 things. Lão this is quite a run, then stop:

00127B70	00AC194E	CALL to GetModuleHandleA
00127B74	00AD6364	pModule = "kernel32.dll"
00127B78	00AD7588	ASCII "VirtualAlloc"

- Press Shift-F9 again see the father:

00127B70	00AC196B	CALL to GetModuleHandleA
00127B74	00AD6364	pModule = "kernel32.dll"
00127B78	00AD757C	ASCII "VirtualFree"

- Just in the leg and then in the Shift-F9 and more:

00127B70	00AA9B49	CALL to GetModuleHandleA
00127B74	00127A24	pModule = "kernel32.dll"

- Okie call, here Tàn per nose look to go home, Trace F8 to RETN:

77E7AD86	837C24 04 00	CMP DWORD PTR SS:[ESP+4],0
77E7AD8B	0F84 37010000	JE kernel32.77E7AEC8
77E7AD91	FF7424 04	PUSH DWORD PTR SS:[ESP+4]
77E7AD95	E8 F8050000	CALL kernel32.77E7B392
77E7AD9A	85C0	TEST EAX,EAX
77E7AD9C	74 08	JE SHORT kernel32.77E7ADA6
77E7AD9E	FF70 04	PUSH DWORD PTR DS:[EAX+4]
77E7ADA1	E8 27060000	CALL kernel32.GetModuleHandleW
77E7ADA6	C2 0400	RETN 4

- Always call in, F8. Then F8 to trace, to this:

00AA9B62	50	PUSH EAX	
00AA9B63	FF15 DC00AD00	CALL DWORD PTR DS:[AD00DC]	kernel32.LoadLibraryA
00AA9B69	8B0D 74B7AD00	MOV ECX,DWORD PTR DS:[ADB774]	
00AA9B6F	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00AA9B72	A1 74B7AD00	MOV EAX,DWORD PTR DS:[ADB774]	
00AA9B77	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00AA9B7A	0F84 32010000	JE 00AA9CB2	
00AA9B80	33C9	XOR ECX,ECX	

- Look through the help FPU per:

C 0	E	C 0
P 1	C	P 1
A 0	S	A 0
Z 0	L	Z 1
S 0	F	S 0
T 0	G	T 0

- Flag Z has not enabled us to 1, the pop-up. Then **he GetModuleHandleA removed**. Then **he salt** to salt **VirtualAlloc**:

Command

- Shift-F9 3 and always do, and then Alt-F9 to go. Then F8 to trace here:

00AC4C25	8985 60C6FFFF	MOV DWORD PTR SS:[EBP-39A0],EAX	
00AC4C2B	83BD 60C6FFFF	CMP DWORD PTR SS:[EBP-39A0],0	
00AC4C32	74 33	JE SHORT 00AC4C67	
00AC4C34	6A 40	PUSH 40	
00AC4C36	68 00100000	PUSH 1000	
00AC4C3B	FFB5 58C6FFFF	PUSH DWORD PTR SS:[EBP-39A8]	
00AC4C41	FF35 D001AE00	PUSH DWORD PTR DS:[AE01D0]	
00AC4C47	FF15 B001AD00	CALL DWORD PTR DS:[AD01B0]	kernel32.VirtualAlloc
00AC4C4D	8985 60C6FFFF	MOV DWORD PTR SS:[EBP-39A0],EAX	

- Look through the vision through:

C 0	ES	I
P 1	CS	I
A 0	SS	I
Z 0	DS	I
S 0	FS	I
T 0	GS	I
D 0		
O 0	Las	

- Unlock the Z's of 0 to re health, secure home to trace, to:

00AC4C25	8985 60C6FFFF	MOV DWORD PTR SS:[EBP-39A0],EAX	
00AC4C2B	83BD 60C6FFFF	CMP DWORD PTR SS:[EBP-39A0],0	
00AC4C32	74 33	JE SHORT 00AC4C67	
00AC4C34	6A 40	PUSH 40	
00AC4C36	68 00100000	PUSH 1000	
00AC4C3B	FFB5 58C6FFFF	PUSH DWORD PTR SS:[EBP-39A8]	
00AC4C41	FF35 D001AE00	PUSH DWORD PTR DS:[AE01D0]	
00AC4C47	FF15 B001AD00	CALL DWORD PTR DS:[AD01B0]	kernel32.VirtualAlloc
00AC4C4D	8985 60C6FFFF	MOV DWORD PTR SS:[EBP-39A0],EAX	
00AC4C53	83BD 60C6FFFF	CMP DWORD PTR SS:[EBP-39A0],0	
00AC4C5A	74 0B	JE SHORT 00AC4C67	

- Edit the value in **EAX** to do (remember Alt-M to consider home), is **496,000**:

Registers (FPU)

EAX 02190000
ECX 77E7AC6F kernel32.77E7AC6F
EDX 7FFE0304

Modify EAX

Hexadecimal
Signed
Unsigned
Char

OK Cancel

- Yes:

EAX	00496000	RMTOMP3.<ModuleEntryPo
ECX	77E7AC6F	kernel32.77E7AC6F
EDX	7FFE0304	
EBX	417B0012	
ESP	00127B7C	
EBP	0012CE30	
ESI	CE300000	
EDI	00000000	
EIP	00AC4C4D	

- Delete **HD VirtualAlloc**. Now click **BP CreateThread** to do. Then Shift-F9 and father ko nhé

Terminate:

77E7BE53	55	PUSH EBP
77E7BE54	8BEC	MOV EBP,ESP
77E7BE56	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
77E7BE59	FF75 18	PUSH DWORD PTR SS:[EBP+18]
77E7BE5C	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77E7BE5F	FF75 10	PUSH DWORD PTR SS:[EBP+10]
77E7BE62	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
77E7BE65	FF75 08	PUSH DWORD PTR SS:[EBP+8]
77E7BE68	6A FF	PUSH -1
77E7BE6A	E8 30FEFFFF	CALL kernel32.CreateRemoteThread
77E7BE6F	5D	POP EBP
77E7BE70	C2 1800	RETN 18
77E7BE73	8AC2	MOV EAX,EDX

Every as-usual, BP chỗ delete this, Ctrl-F9:

77E7BE53	55	PUSH EBP
77E7BE54	8BEC	MOV EBP,ESP
77E7BE56	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
77E7BE59	FF75 18	PUSH DWORD PTR SS:[EBP+18]
77E7BE5C	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77E7BE5F	FF75 10	PUSH DWORD PTR SS:[EBP+10]
77E7BE62	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
77E7BE65	FF75 08	PUSH DWORD PTR SS:[EBP+8]
77E7BE68	6A FF	PUSH -1
77E7BE6A	E8 30FEFFFF	CALL kernel32.CreateRemoteThread
77E7BE6F	5D	POP EBP
77E7BE70	C2 1800	RETN 18
77E7BE73	8AC2	MOV EAX,EDX

- F8:

00AAFE76	5E	POP ESI
00AAFE77	C9	LEAVE
00AAFE78	C3	RETN
00AAFE79	55	PUSH EBP

- Ctrl-F9:

00AAFE76	5E	POP ESI
00AAFE77	C9	LEAVE
00AAFE78	C3	RETN
00AAFE79	55	PUSH EBP

- F8:

00ACA186	A1 2800AE00	MOV EAX,DWORD PTR DS:[AE0028]
00ACA18B	59	POP ECX
00ACA18C	8B48 50	MOV ECX,DWORD PTR DS:[EAX+50]
00ACA18F	3348 38	XOR ECX,DWORD PTR DS:[EAX+38]
00ACA192	3348 34	XOR ECX,DWORD PTR DS:[EAX+34]
00ACA195	F6C1 40	TEST CL,40
00ACA198	75 00	JNZ 00ACA19C

- Pull-down search commands nearly **thắng** CALL RETN first. It set for a BP:

00ACA221	2BF9	SUB EDI,ECX
00ACA223	FFD7	CALL EDI
00ACA225	8BD8	MOV EBX,EAX
00ACA227	5F	POP EDI
00ACA228	8BC3	MOV EAX,EBX
00ACA22A	5E	POP ESI
00ACA22B	5B	POP EBX
00ACA22C	C3	RETN

- Shift-F9, stop it right:

00ACA221	2BF9	SUB EDI,ECX
00ACA223	FFD7	CALL EDI
00ACA225	8BD8	MOV EBX,EAX
00ACA227	5F	POP EDI
00ACA228	8BC3	MOV EAX,EBX
00ACA22A	5E	POP ESI
00ACA22B	5B	POP EBX
00ACA22C	C3	RETN

- Delete BP seats that go, then to F7, Bummmmm:

00427AFA	55	PUSH EBP
00427AFB	8BEC	MOV EBP,ESP
00427AFD	6A FF	PUSH -1
00427AFF	68 881B4500	PUSH RNTOMP3.00451B88
00427B04	68 A8AF4200	PUSH RNTOMP3.0042AFA8
00427B09	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
00427B0F	50	PUSH EAX
00427B10	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
00427B17	83EC 58	SUB ESP,58
00427B1A	53	PUSH EBX
00427B1B	56	PUSH ESI
00427B1C	57	PUSH EDI
00427B1D	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP
00427B20	FF15 C4C14400	CALL DWORD PTR DS:44C1C4
00427B26	33D2	XOR EDX,EDX

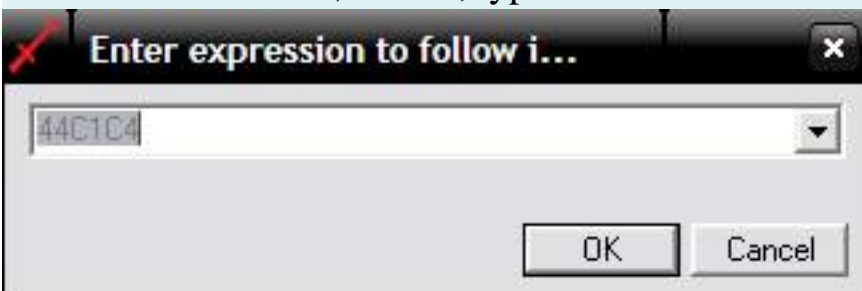
kernel32.GetVersion

- The OEP ship, Lord hours if used full-PE and then dump Fix IAT with ImportREC when the crash will be run. Why crash? I think she is also a jump Magic is not processed. So to that time under the command CALL OEP kia ([kernel32.GetVersion](#)), small home **44C1C4** perimeter. Also remember to always OEP home.

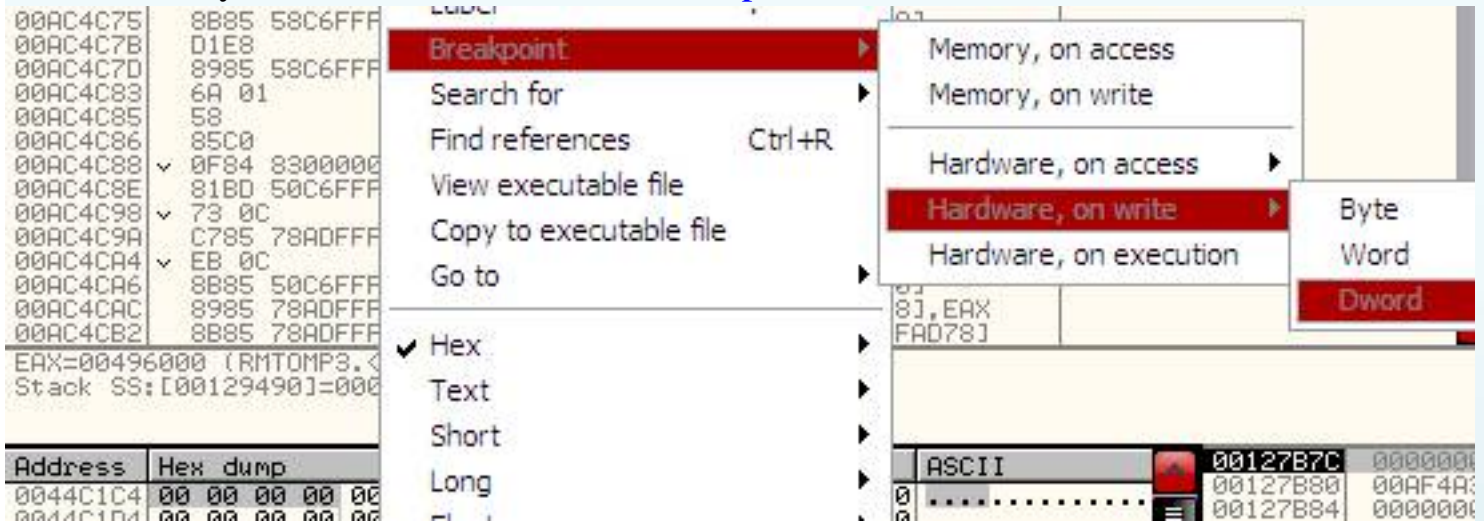
- Restart the Olly, they do not have labels, training martial but suffered extreme aiiiiii

- Follow from top to bottom, until the correct value in EAX completed, **hd VirtualAlloc**.

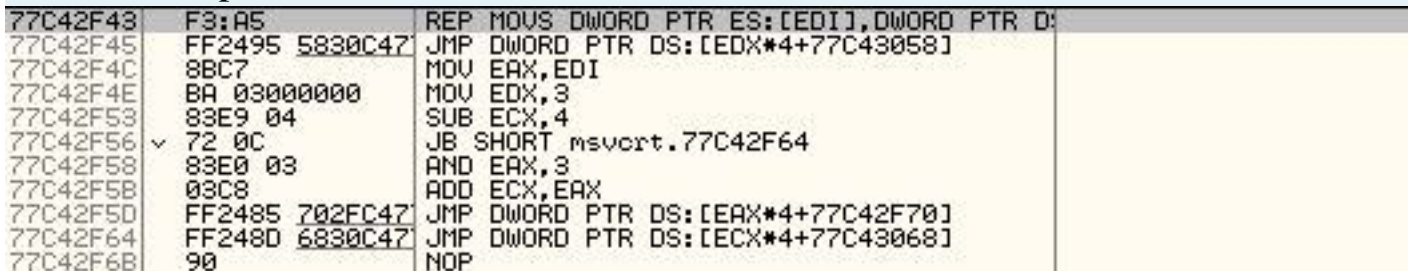
- In the window Hex, Ctrl-G, type in the address:



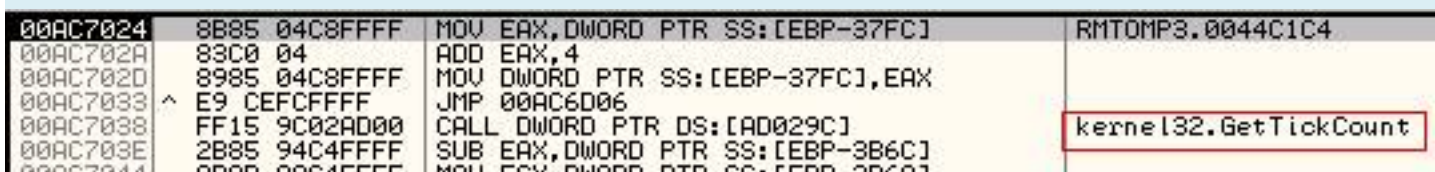
- Then to 4 Byte first and then to Click, **Breakpoint à Hardware, on Write a Dword:**



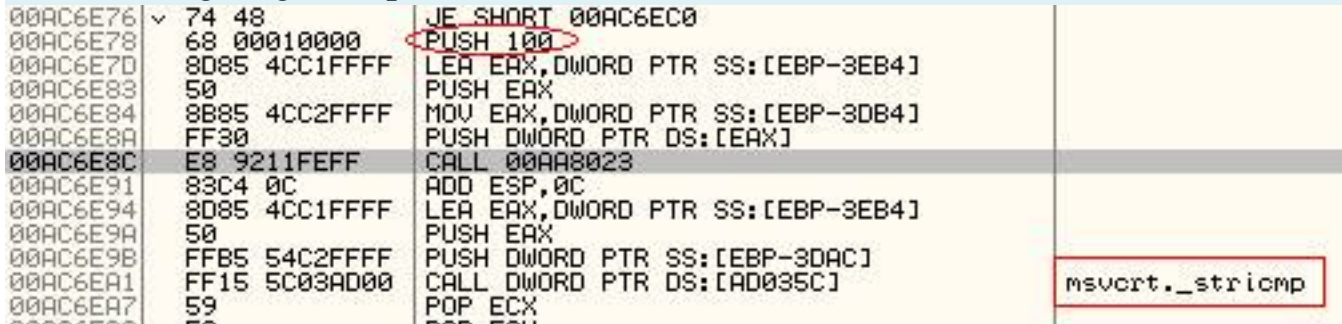
- Shift-F9 stop it:



- Shift-F9 is 2 times:



- Look GetTickCount ko, if read Chapter 1 of the hacnho then it will be very aged very happy there. Hi hi. Hoik find the **msvcrt._stricmp**, PUSH see above 100, then under the command CALL enough signs ship.



- The time that memory address of this CALL command, and then Restart the Tàn per maroon WA. From here, open to look down to see the little ni:

00AC6E91	83C4 0C	ADD ESP,0C	
00AC6E94	8D85 4CC1FFFF	LEA EAX,DWORD PTR SS:[EBP-3EB4]	
00AC6E9A	50	PUSH EAX	
00AC6E9B	FFB5 54C2FFFF	PUSH DWORD PTR SS:[EBP-3DAC]	
00AC6EA1	FF15 5C03AD00	CALL DWORD PTR DS:[AD035C]	msvcrt._stricmp
00AC6EA7	59	POP ECX	
00AC6EA8	59	POP ECX	
00AC6EA9	85C0	TEST EAX,EAX	
00AC6EAB	75 11	JNZ SHORT 00AC6EBE	
00AC6EAD	8B85 4CC2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DB4]	
00AC6EB3	8B40 08	MOV EAX,DWORD PTR DS:[EAX+8]	
00AC6EB6	8985 58C2FFFF	MOV DWORD PTR SS:[EBP-3DA8],EAX	
00AC6EBC	EB 02	JMP SHORT 00AC6EC0	
00AC6EBE	EB 9D	JMP SHORT 00AC6E5D	
00AC6EC0	8B85 98C4FFFF	MOV EAX,DWORD PTR SS:[EBP-3B68]	
00AC6EC6	40	INC EAX	
00AC6EC7	8985 98C4FFFF	MOV DWORD PTR SS:[EBP-3B68],EAX	
00AC6ECD	83BD 58C2FFFF	CMP DWORD PTR SS:[EBP-3DA8],0	
00AC6ED4	75 42	JNZ SHORT 00AC6F18	
00AC6ED6	0FB785 5CC2FFFF	MOVZX EAX,WORD PTR SS:[EBP-3DA4]	

- Select the command JNZ 2 under the [msvcrt._stricmp](#), then Click right, [Breakpoint à Hardware](#), on execution:

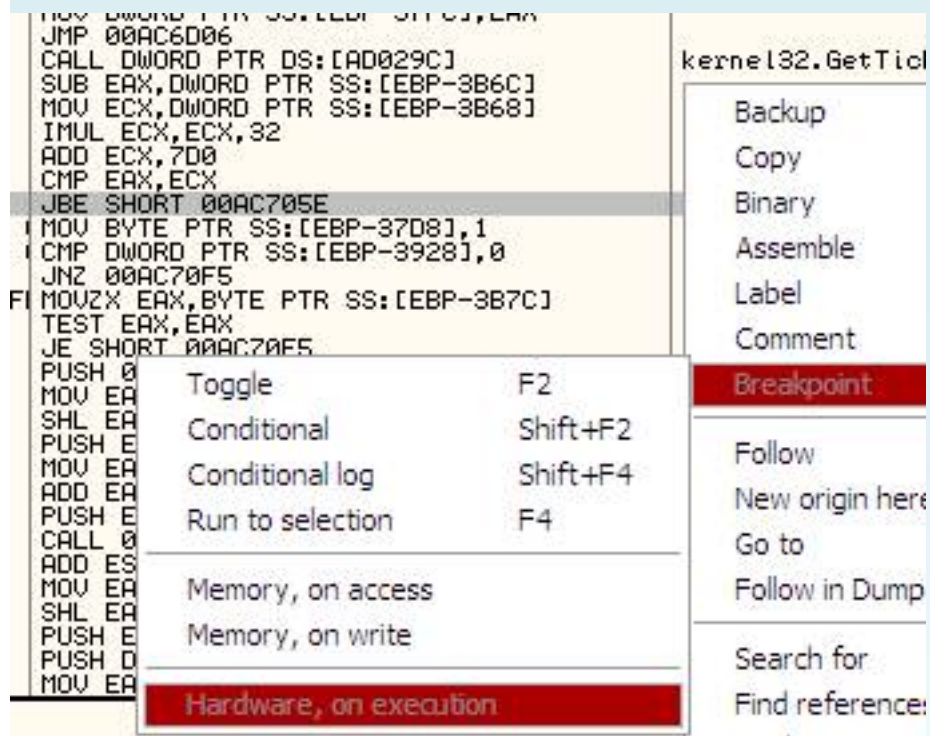
00AC6E9A	50	PUSH EAX	
00AC6E9B	FFB5 54C2FFFF	PUSH DWORD PTR SS:[EBP-3DAC]	
00AC6EA1	FF15 5C03AD00	CALL DWORD PTR DS:[AD035C]	msvcrt._stricmp
00AC6EA7	59	POP ECX	
00AC6EA8	59	POP ECX	
00AC6EA9	85C0	TEST EAX,EAX	
00AC6EAB	75 11	JNZ SHORT 00AC6EBE	
00AC6EAD	8B85 4CC2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DB4]	
00AC6EB3	8B40 08	MOV EAX,DWORD PTR DS:[EAX+8]	
00AC6EB6	8985 58C2FFFF	MOV DWORD PTR SS:[EBP-3DA8],EAX	
00AC6EBC	EB 02	JMP SHORT 00AC6EC0	
00AC6EBE	EB 9D	JMP SHORT 00AC6E5D	
00AC6EC0	8B85 98C4FFFF	MOV EAX,DWORD PTR SS:[EBP-3B68]	
00AC6EC6	40	INC EAX	
00AC6EC7	8985 98C4FFFF	MOV DWORD PTR SS:[EBP-3B68],EAX	
00AC6ECD	83BD 58C2FFFF	CMP DWORD PTR SS:[EBP-3DA8],0	
00AC6ED4	75 42	JNZ SHORT 00AC6F18	
00AC6ED6	0FB785 5CC2FFFF	MOVZX EAX,WORD PTR SS:[EBP-3DA4]	
00AC6EDD	85C0	TEST EAX,EAX	
00AC6EDF	74 0F	JE SHORT 00AC6EE1	
00AC6EE1	0FB785 5CC2FFFF	MOVZX EAX,WORD PTR SS:[EBP-3DA4]	
00AC6EE8	8985 5CADFFFF	MOV DWORD PTR SS:[EBP-3B68],EAX	
00AC6EEE	EB 0C	JMP SHORT 00AC6EF0	
00AC6EF0	8B85 54C2FFFF	MOV EAX,DWORD PTR SS:[EBP-3B68]	
00AC6EF6	8985 5CADFFFF	MOV DWORD PTR SS:[EBP-3B68],EAX	
00AC6EFC	6A 01	PUSH 1	

Hardware, on execution

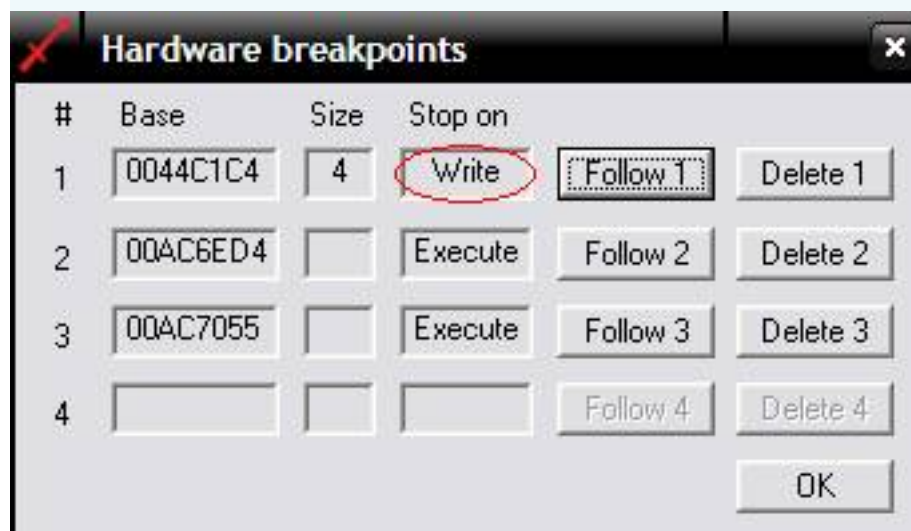
- Then click marked "-" back to front seats, under dom [thắng GetTickCount](#), see command JBE ko:

00AC7024	8B85 04C8FFFF	MOV EAX,DWORD PTR SS:[EBP-37FC]	RMTOMP3.0044C1C4
00AC702A	83C0 04	ADD EAX,4	
00AC702D	8985 04C8FFFF	MOV DWORD PTR SS:[EBP-37FC],EAX	
00AC7033	E9 CEFCEFFF	JMP 00AC6D06	
00AC7038	FF15 9C02AD00	CALL DWORD PTR DS:[AD029C]	kernel32.GetTickCount
00AC703E	2B85 94C4FFFF	SUB EAX,DWORD PTR SS:[EBP-3B6C]	
00AC7044	8B8D 98C4FFFF	MOV ECX,DWORD PTR SS:[EBP-3B68]	
00AC704A	6BC9 32	IMUL ECX,ECX,32	
00AC704D	81C1 D0070000	ADD ECX,7D0	
00AC7053	3BC1	CMP EAX,ECX	
00AC7055	76 07	JBE SHORT 00AC705E	
00AC7057	0C0F 0000FFFF	MOV BYTE PTR SS:[EBP-37D1],AL	

- Click Select, then it must, [that the hardware Breakpoint](#), on execution:



- Delete HW BP admits to:



- Clear Finish:



- Now Shift_F9 to:

00AC6ED4	75 42	JNZ SHORT 00AC6F18
00AC6ED6	0FB785 5CC2FFFF	MOVZX EAX,WORD PTR SS:[EBP-3DA4]
00AC6EDD	85C0	TEST EAX,EAX

- Edit it as follows:

00AC6ED4	75 42	JNZ SHORT 00AC6F18
00AC6ED6	0FB785 5CC2FFFF	MOVZX EAX,WORD PTR SS:[EBP-3DA4]
00AC6EDD	85C0	TEST EAX,EAX

Backup	EAX
Copy	00AC6EF0
Binary	X,WORD PTR SS:[EBP-3DA4]
Assemble	Space
Label	:
	Fill with 00's
	Fill with NOPs

- Done a salt:

00AC6ED4	90	NOP
00AC6ED5	90	NOP

- Remove HW BP place this (to the NOP small line 2):

Toggle	F2	Breakpoint
Conditional	Shift+F2	Go to
Conditional log	Shift+F4	Follow in D
Memory, on access		Search for
Memory, on write		Find refere
Hardware, on execution		Analysis
Remove hardware breakpoint		Asm?Clinh

- Shift-F9 to primary salt 2:

00AC7055	76 07	JBE SHORT 00AC705E
00AC7057	C685 28C8FFFF	MOV BYTE PTR SS:[EBP-37D8],1

- Patch it:

00AC7055	EB 07	JMP SHORT 00AC705E
00AC7057	C685 28C8FFFF	MOV BYTE PTR SS:[EBP-37D8],1
00AC705E	83BD D8C6FFFF	CMP DWORD PTR SS:[EBP-3928],0

- Ha. Cú this is the same jump kî is cú jump final is always small, remove HW BP place this:

Toggle	F2	Breakpoint
Conditional	Shift+F2	Follow
Conditional log	Shift+F4	Go to
Memory, on access		Follow in Dump
Memory, on write		Search for
Hardware, on execution		Find reference
Remove hardware breakpoint		Analysis

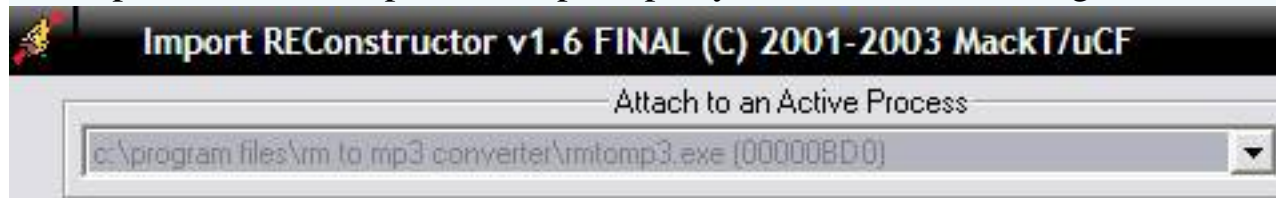
- Now F9 him to run, it failed to place it, oach crash:

00AC75FC	CD CC	INT 0CC	
00AC75FE	EE	OUT DX,AL	I/O command
00AC75FF	31E9	XOR ECX,EBP	
00AC7601	90	NOP	
00AC7602	51	PUSH ECX	
00AC7603	EB 0A	JMP SHORT 00AC760F	
00AC7605	56	PUSH ESI	
00AC7606	06	SAL C	

- No problem, calm. Olly resources for the Lord to dump full-Pe to do. Now Mo is with the

computer should be used this month. I also better perimeter nhi?

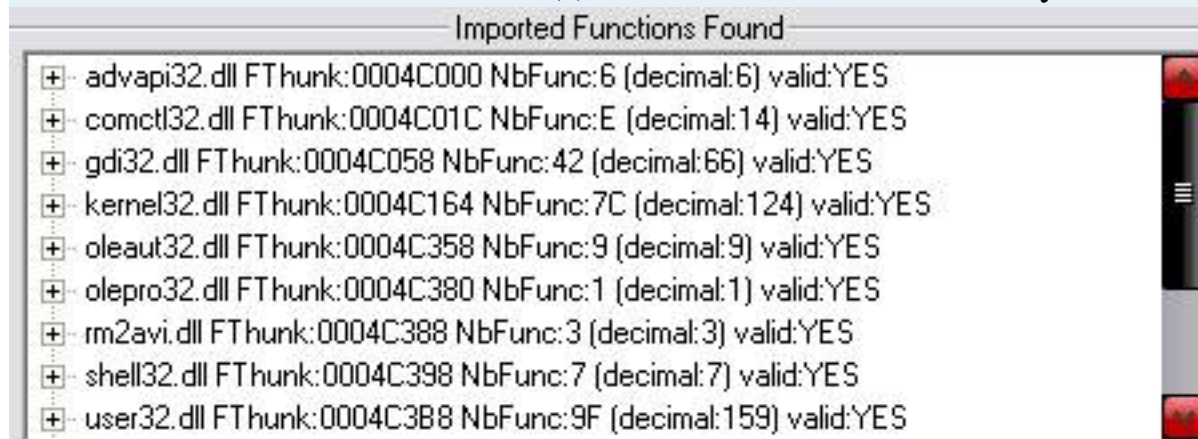
- Dump the finished ImportREC open up any, should select the target A:



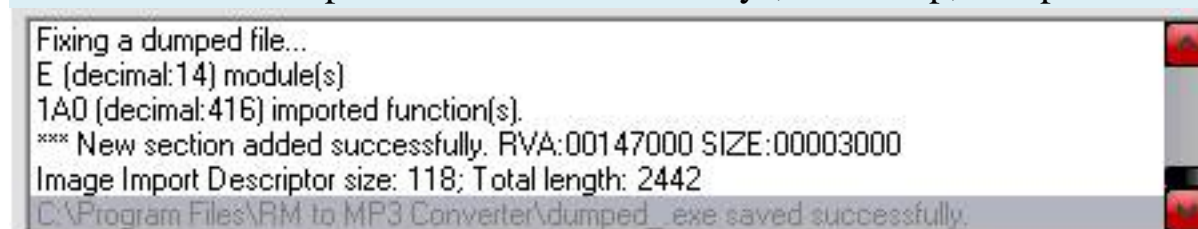
- The OEP not remember, is 400,000 less 27AFA, then fill IAT Auto Search:



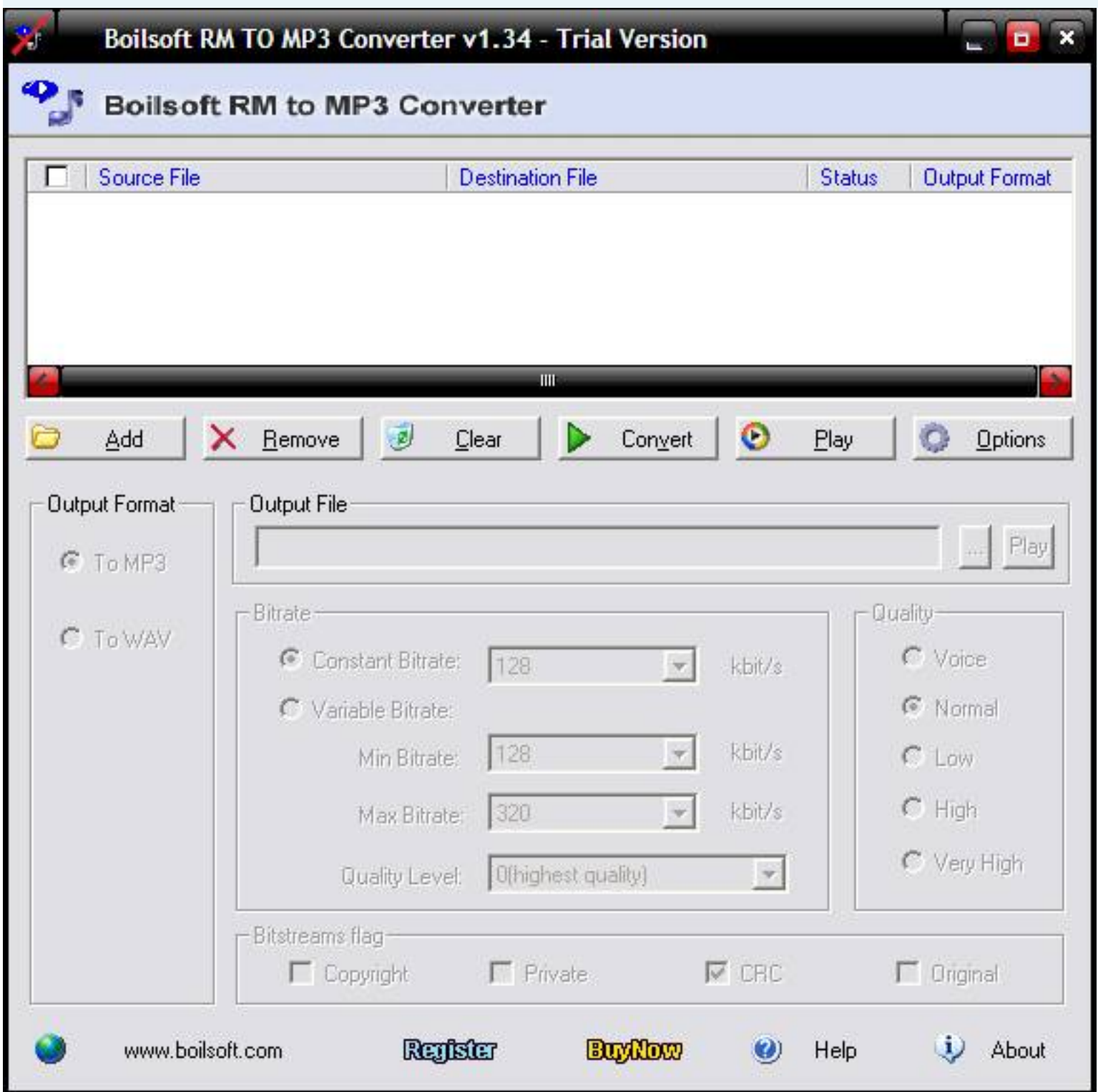
- The Show Invalid à thunk Cut (s), then, salt is salt of their way:



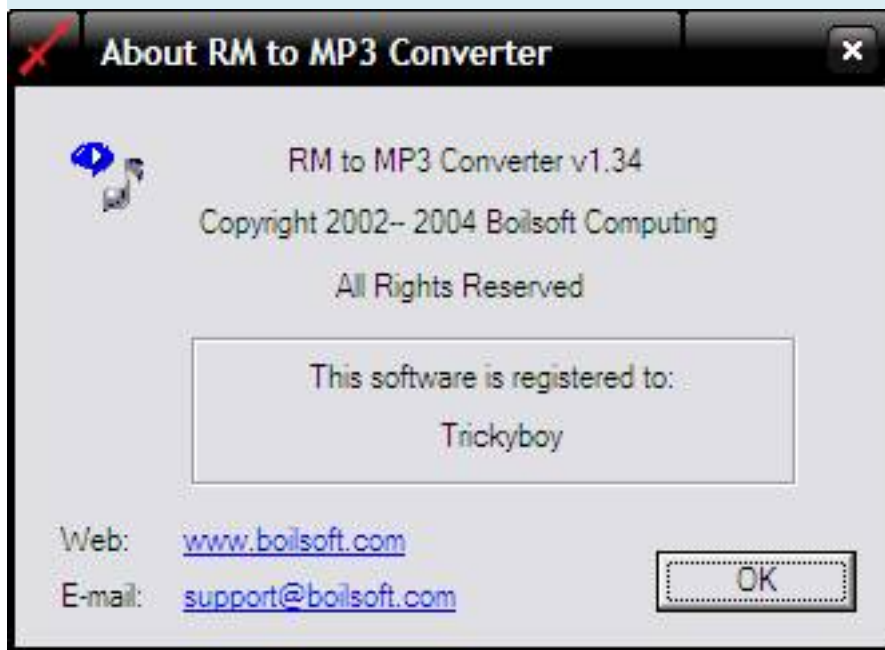
- The afternoon we put her into the river always, Fix dump, dump him choose at this time:



- From Ủ, any newspaper is to go **successfully**, a thẳng any food Crash chairs the line Ah, Dang Cam chairs nè. RUN things considered, close your eyes Yawn only one:



- Number see open, running gọi hì hi. Quay through a back, naturally this is the table, AC:



- What is the per Tàn completed in time know Armadillo_Code --- Ta Splicing of Justice. The Ta it real. When you open a photograph on file of (at xuc Polyphonic), Pink is filled with machines that "monitor" the the file. doc up by 11 per MB. Who code down on hix hix. chiêu Although this is not in Ta Kiem Blood epiphyte's natural for anonymous stars but he needs to open holes for feedback, chán chairs. Thôi on the home, Capture the few pictures for the aged that home.

- Ta lesson this time the legal focus key way in which per temporarily called "Jump by Z". The day, per well is expected to be Newbie "to" more of the new Armadillo thắng this, then remember to know per home. ^ 0 ^ Respiratory Ho Ho! From from ...

*- Sincerely thank
you for Admin, Mod,
and all members of
the REA.*

*- Especially as he
hacnho (in particular
he called).*

*- Thanx to:
- MaDMAn_H3rCuL3s (I
hate this guy)
- Ricardo Narvaja (I
like this guy)*

- And you ...



Search by ngoay is (written by): Trickyboy (self Wiuewi)

Import elimination Debugblocker + + Nanomites

SoftWare : **Picture Ripper 3**

Packed : **Armadillo 4.xx - Debugblocker elimination + + Import Nanomites**

Crack Tool **1.** **OllyDBG by hacnho.**

2. LordPE Deluxe 1.4-by yoda

3. Import REConstructor 1.6 Final

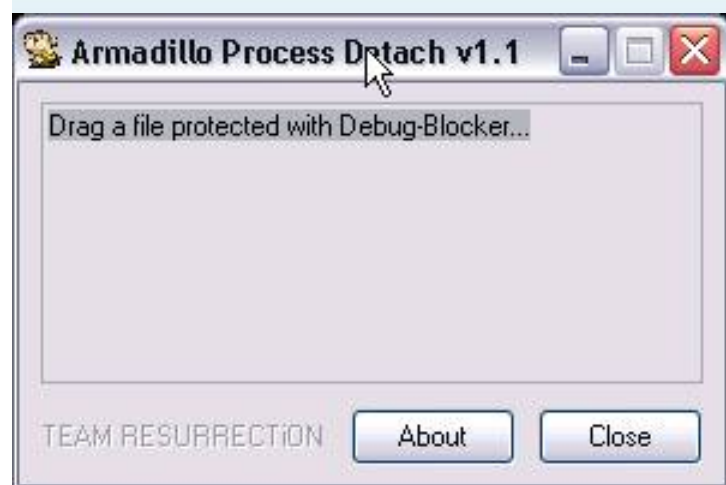
4. ArmInline 0.71

5. ArmaDetach 1.1

Author : **Why Not Bar**

_Tut Before the target **Movie Collector 4.4** can not do much Uncle Done! Not know where the errors started? But they are all Done. Bua this, they conducted with thằng **Picture Ripper 3** Hacnho he has meat and then write a tut. I write TUT with the purpose to Uncle reference and comparison only. By To pass **blocker** they use **Debug** Tool for fast **ArmaDetach 1.1**.

_Chay **ArmaDetach 1.1**:



_Keo File "PictureRipper.exe" and released into the window

PictureRipper

Internet Shortcut
1 KB



PictureRipper.exe
Free picture and movie downl...
GlobalGet, Inc.


Armadillo Process Detach v1.1





DONE!
 |Child process ID: 0000019C|
 |Entry point: 0059B000|
 |Original bytes: 60E8

TEAM RESURRECTION

About

Close


Attach _Chay olly and Child, F9, F12 and patch as follows:

00598000	60	PUSHAD	
00598001	E8 00000000	CALL PictureR.00598006	
00598006	50	POP EBP	
00598007	50	PUSH EAX	
00598008	51	PUSH ECX	
00598009	0FCA	BSWAP EDX	
0059800B	F7D2	NOT EDX	
0059800D	9C	PUSHFD	
0059800E	F7D2	NOT EDX	
00598010	0FCA	BSWAP EDX	
00598012	EB 0F	JMP SHORT PictureR.00598023	
00598014	B9 EB0FB8EB	MOV ECX,EBB80FEB	
00598019	07	POP ES	Modification of segment register
0059801A	B9 EB0F90EB	MOV ECX,EB900FEB	
0059801F	08FD	OR CH,BH	
00598021	EB 0B	JMP SHORT PictureR.0059802E	
00598023	F2:	PREFIX REPNE:	Superfluous prefix
00598024	EB F5	JMP SHORT PictureR.0059801B	
00598026	EB F2	JMP SHORT PictureR.0059801F	

_Chay Scripts "Armadillo Standard unpack"

004C80AA	EA 60	PUSH 60	<- OEP
004C80AC	68 88A35300	PUSH PictureR.0053A388	
004C80B1	E8 CE230000	CALL PictureR.004CA484	
004C80B6	BF 94000000	MOV EDI,94	
004C80B7	00C7	MOV ECX,ECI	

OllyScript



You're at the OEP, now dump with LordPE and fix the IAT with ImpRec. =)

OK

004C80ED	81E8 FF7F0000	AND ESI,7FFF	
004C80F3	8935 9C895600	MOV DWORD PTR DS:[56899C],ESI	
004C80F9	83F9 02	CMP ECX,2	
004C80FC	74 0C	JF SHORT PictureR.004C8010	

Import _Bay hours to *elimination*. Use ArmInline 0.71 to Fix but first we need to find IAT
IAT start: **00F4D038**

```

00F4D030 00000000
00F4D034 00000000
00F4D038 14010154
00F4D03C 010C01BB
00F4D040 77D4D16F USER32.GetTopWindow
00F4D044 7C937A40 ntdll.RtlUnwind
00F4D048 77F19012 GDI32.ExtTextOutA
00F4D04C 7C826F4B kernel32.GetTimeFormatA
00F4D050 77D53A2F USER32.GetScrollInfo
00F4D054 77D4902C USER32.SetScrollInfo
00F4D058 77D52316 USER32.RegisterClassA
00F4D05C 77D4C48A USER32.IsIconic
00F4D060 00F10E29

```

IAT end: 00F4DAD0

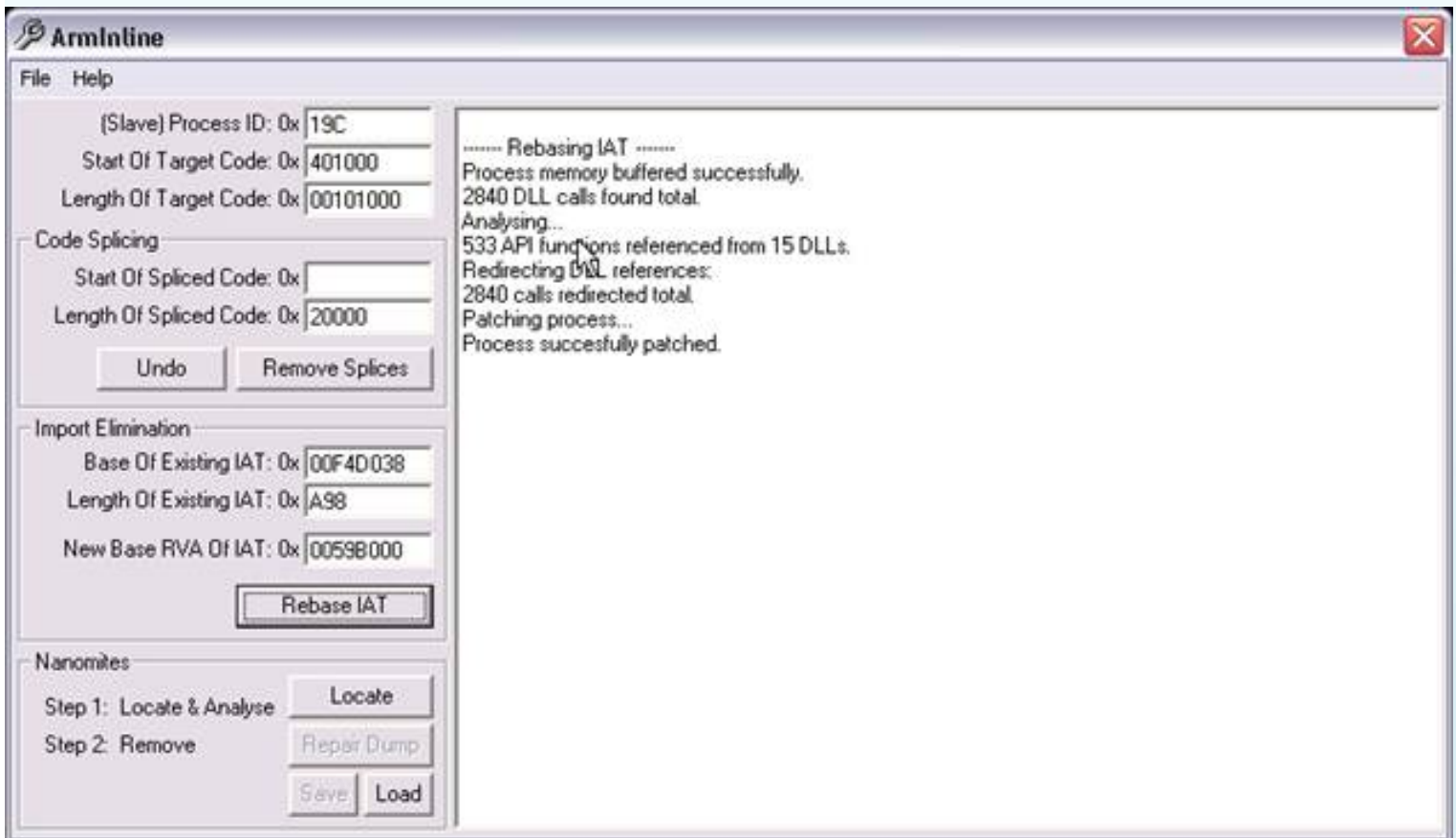
```

00F4DAD0 00E1AD19
00F4DAD4 00000000
00F4DAD8 015400A9
00F4DADC 010C10E7
00F4DAE0 00FB0910
00F4DAE4 00DA0178
00F4DAE8 00001281

```

Len: A98

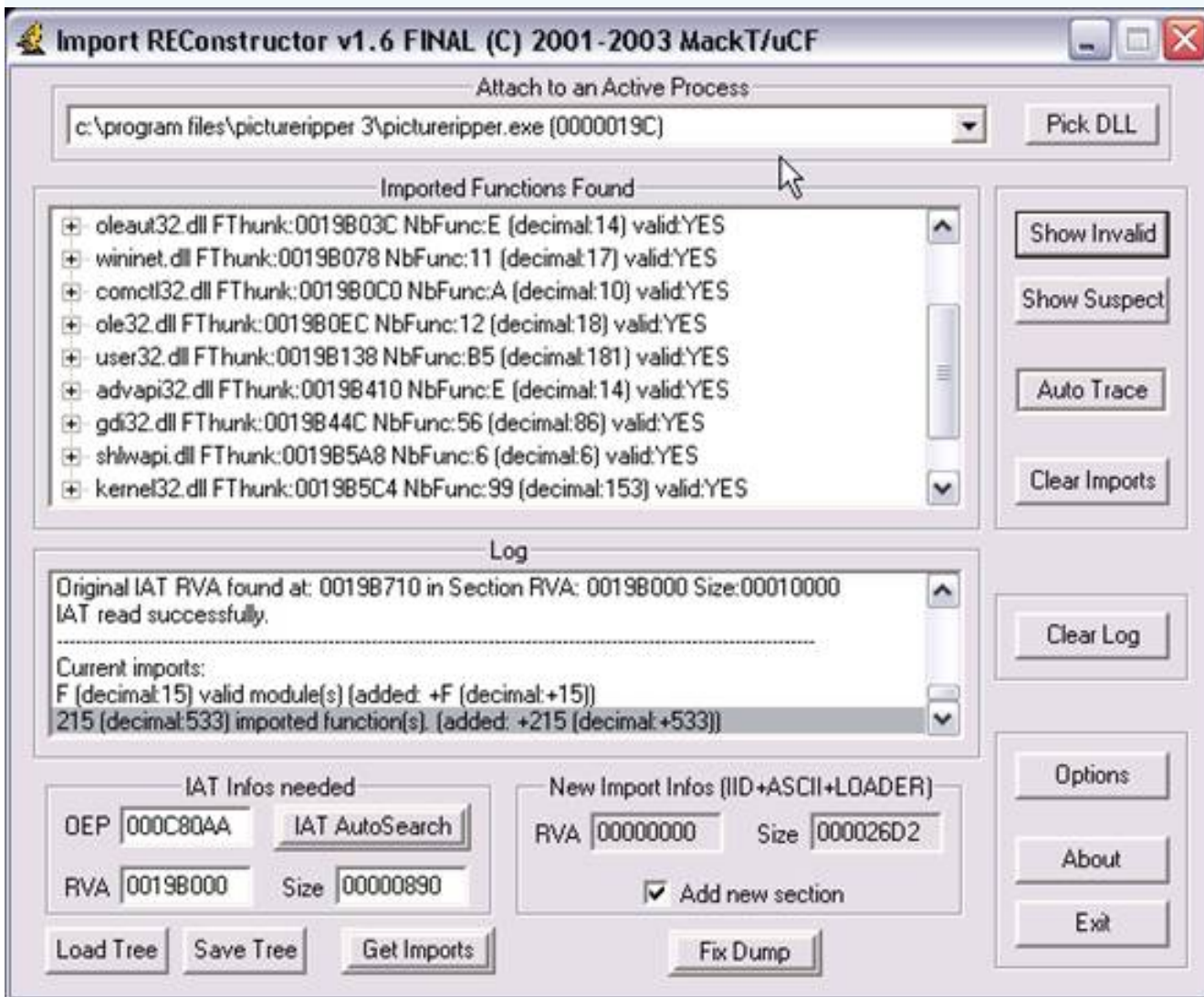
_Chay ArmInline and enter the number and press



_Dung LordPE Full dump

Path	PID	ImageBase	ImageSize
d:\crack tool\armadillo tool\armadetach.exe	00000670	00400000	00008000
c:\program files\hypersnap-dx 5\hprsnap5.exe	000007D8	00400000	0040D000
c:\program files\pictureripper 3\pictureripper.exe	00000290	00400000	00312000
c:\program files\...		00400000	00312000
<div> <div>dump full...</div> <div>dump partial...</div> <div>dump region...</div> </div>			
Path		ImageSize	

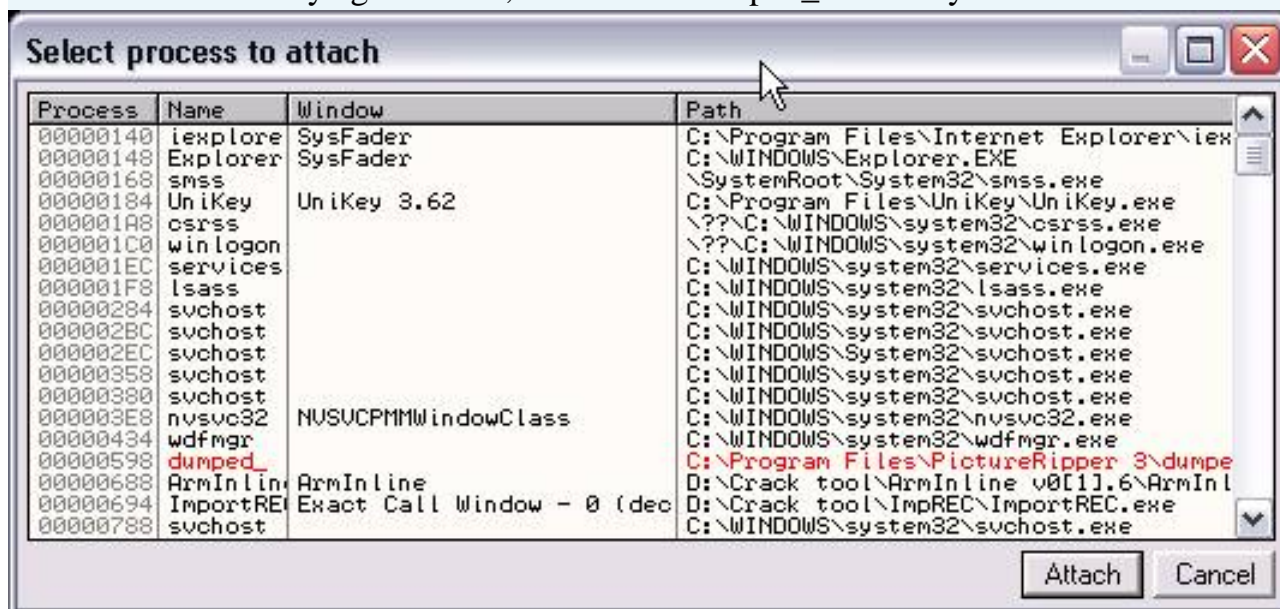
_Bay Hours and fill open ImportREC parameters:



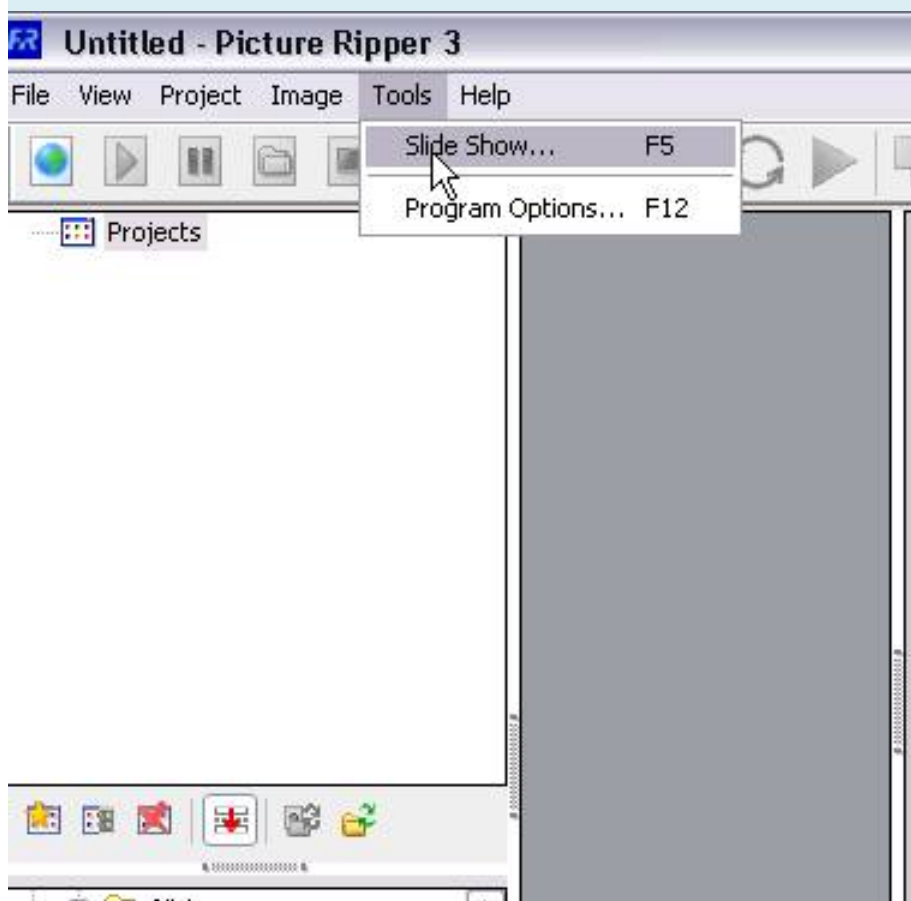
_Fix Dump to try and Run



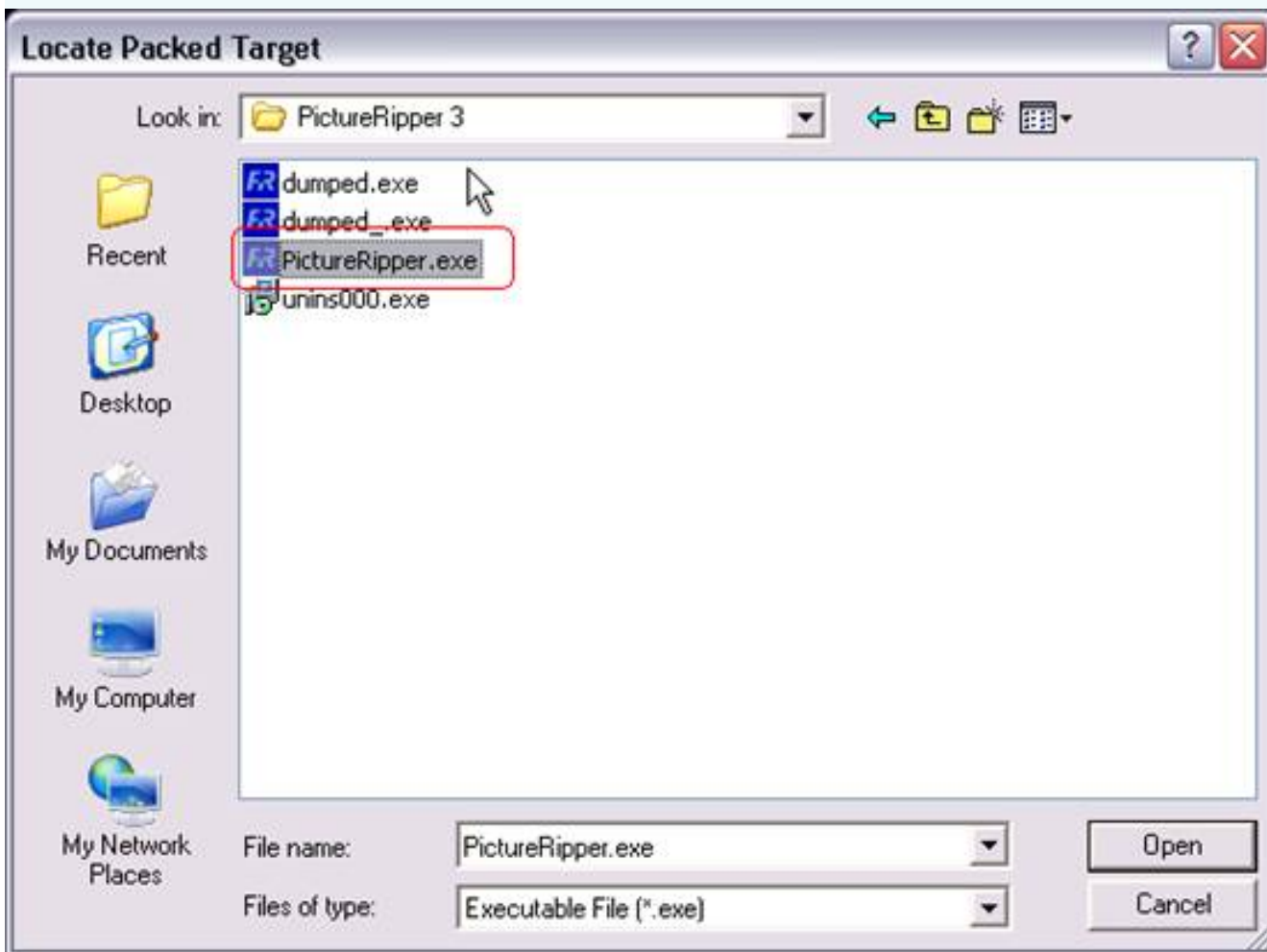
OK Gòi! Now to *Nanomites*. The doctors then they do not know the children but this was! Now, we close ArmaDetach and Olly again. Then, Load File "dumped.exe" Olly to see PID is how much.



_Dien PID to ArmInline, click  , Programs you choose and run as follows:



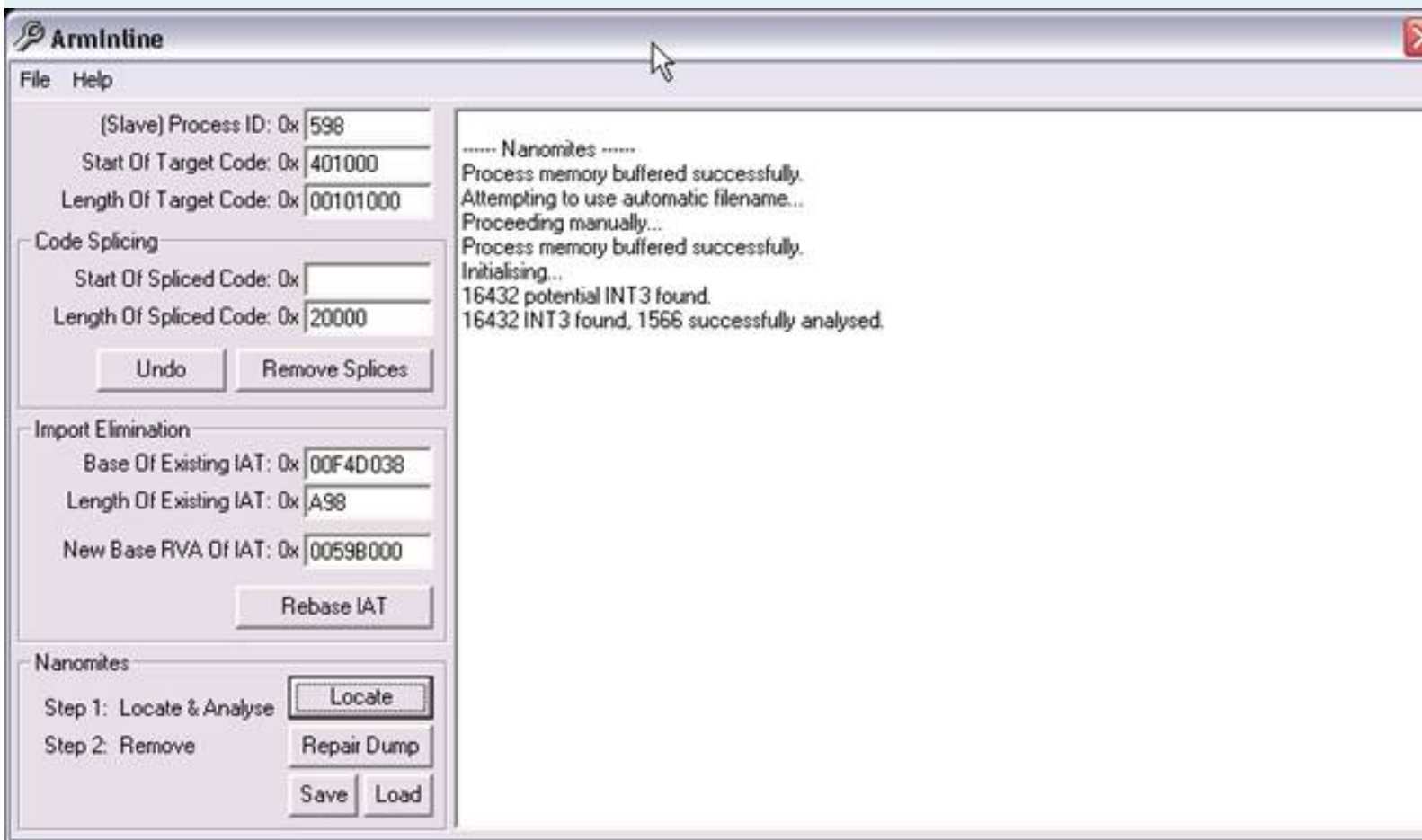
_tip by selecting "PictureRipper.exe"

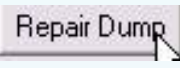


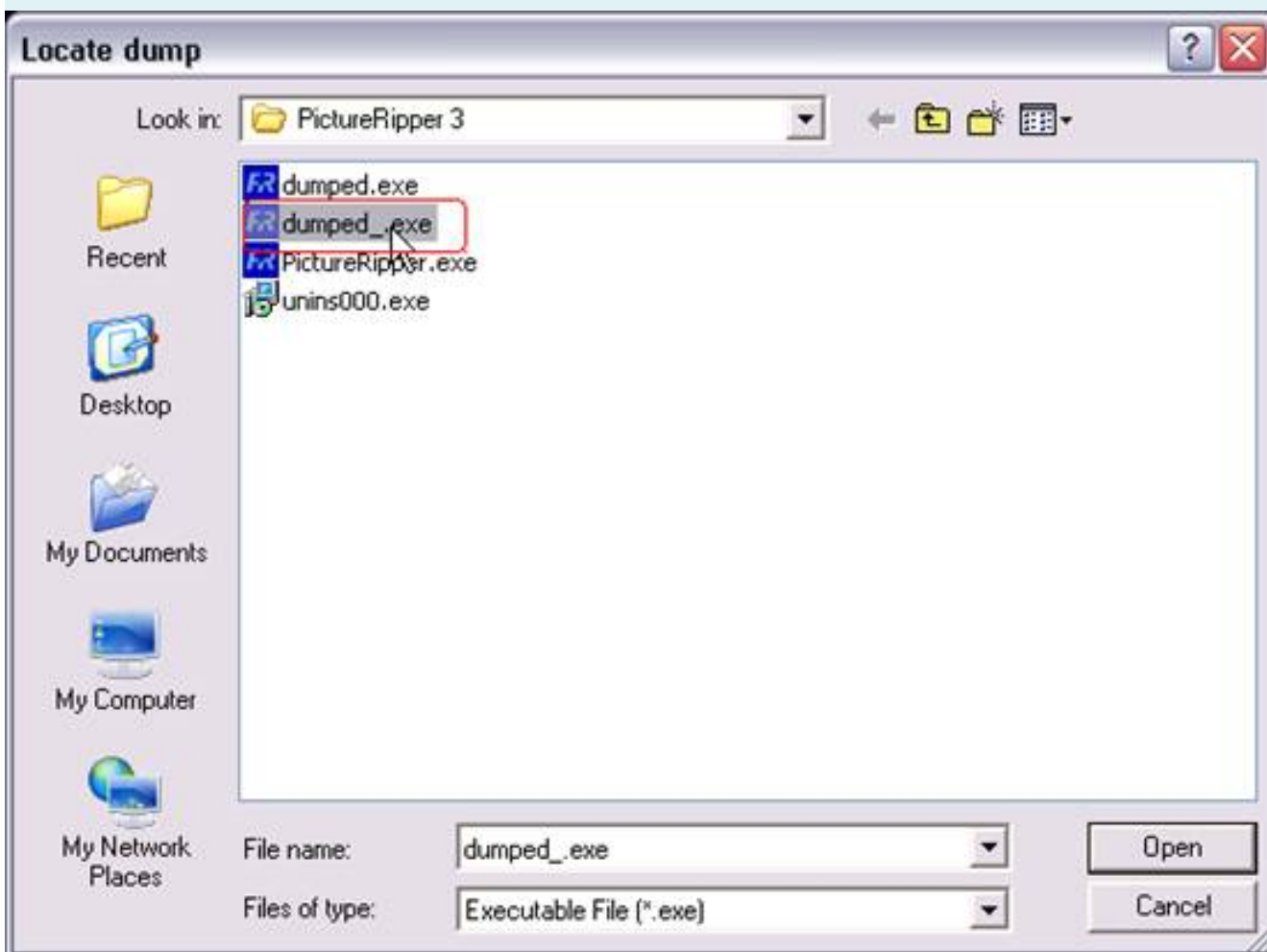
Run to the _Chuong 1 again and choose as you now:



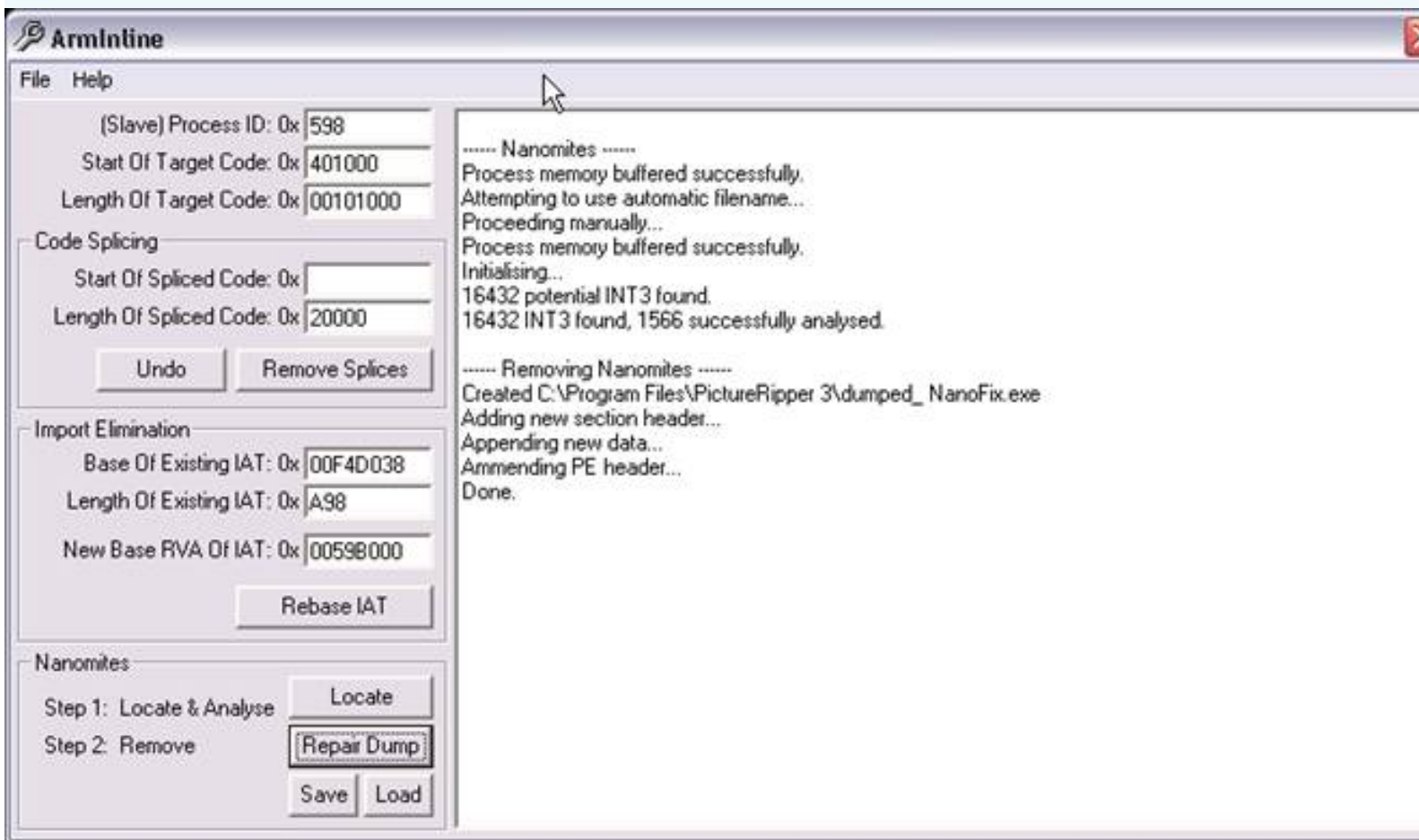
_Khi This, the window we see ArmInline as follows



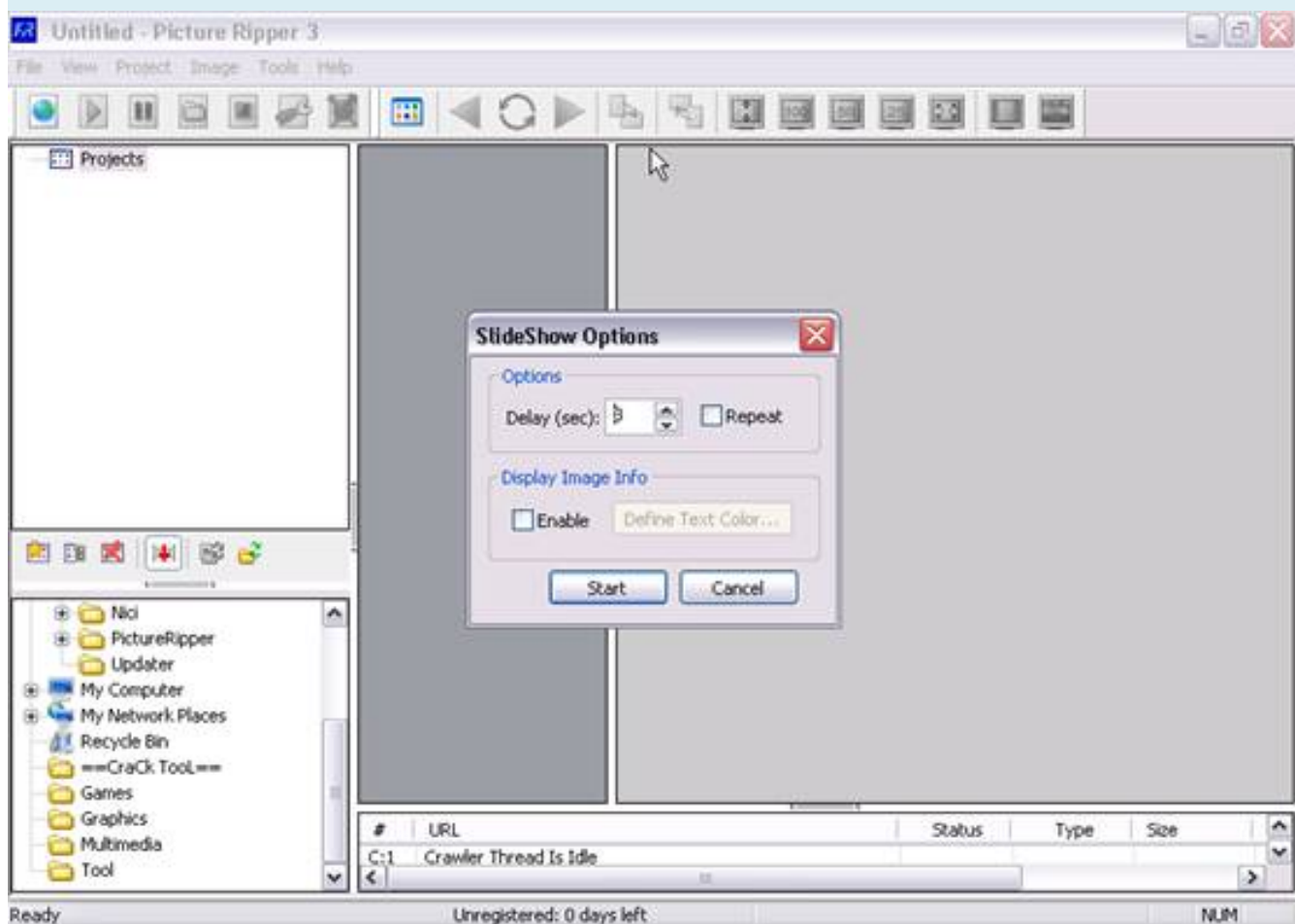
_Tiep To hit us  and select the file "dumped_.exe"



_ArmInline Was then kìa Done!



Nhu The doctors found it, very fast right? Test Run file "dumped NanoFix.exe" see the stars.



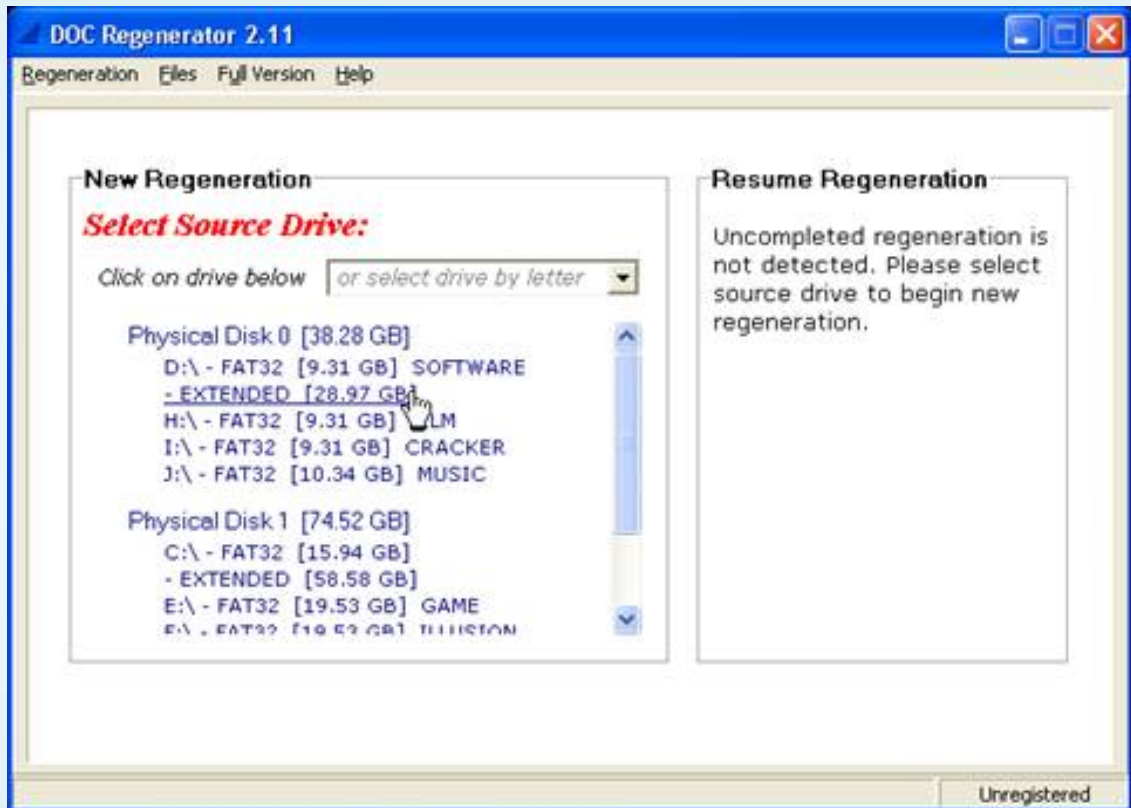
_He He! Done unpack. Chúc of Uncle successful home!

Written by Why Not Bar

Armadillo collect sand-stone

Target: [DOC Regenerator v2.11](#)

Debug Blocker + Hardware Finger Print



I. Tools

- 1.OllyDBG - The best config debugger for ArmMUP by hacnho.
- 2.LordPE 1.4 Deluxe
- 3.Import REConstructor 1.6 Final

II. Unpacking

_Kiem Investigation:

Image Name	PID	Description	User Name
XTM.exe	3872	Warecase eXtended Task Mana...	VIP\Super Administrator
DOC Regenerat...	3828		VIP\Super Administrator
DOC Regenerat...	3808		VIP\Super Administrator
TSCHelp.exe	3060	TechSmith HTML Help Helper	VIP\Super Administrator
Snagit32.exe	3052	Snagit Screen Capture for Win...	VIP\Super Administrator
UniKey.exe	2392		VIP\Super Administrator
SystranServer....	2360	SYSTRAN Server	VIP\Super Administrator
WINWORD.EXE	2304	Microsoft Office Word	VIP\Super Administrator
WISPTIS.EXE	1880	Microsoft Tablet PC Platform Co...	VIP\Super Administrator

_Load On target:

Address	Hex dump	Disassembly	Comment
00487BB0	55	PUSH EBP	
00487BB1	8BEC	MOV EBP,ESP	
00487BB3	6A FF	PUSH -1	
00487BB5	68 A02A4A00	PUSH DOC_Rege.004A2AA0	
00487BB8	68 88784800	PUSH DOC_Rege.00487888	
00487BBF	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	SE handler installation
00487BC5	50	PUSH EAX	
00487BC6	64:8925 000000	MOV DWORD PTR FS:[0],ESP	
00487BCD	83EC 58	SUB ESP,58	
00487BD0	53	PUSH EBX	
00487BD1	56	PUSH ESI	
00487BD2	57	PUSH EDI	
00487BD3	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00487BD6	FF15 78D14900	CALL DWORD PTR DS:[<&KERNEL32.GetVersion	kernel32.GetVersion
00487BDC	33D2	XOR EDX,EDX	
00487BDE	8A04	MOV DL,AH	

_Dat Breakpoint: BP WriteProcessMemory. F9:

Address	Value	Comment
0012B9AC	0047C273	CALL to WriteProcessMemory from
0012B9B0	0000004C	hProcess = 0000004C (window)
0012B9B4	00487BB0	Address = 487BB0
0012B9B8	0012BC9C	Buffer = 0012BC9C
0012B9BC	00000002	BytesToWrite = 2
0012B9C0	0012BCA0	pBytesWritten = 0012BCA0
0012B9C4	0012D23C	UNICODE "Kernel32.dll"
0012B9C8	00002710	

_F9 Continue:

Address	Value	Comment
0012B9AC	0047C29B	CALL to WriteProcessMemory from
0012B9B0	0000004C	hProcess = 0000004C (window)
0012B9B4	00487BB0	Address = 487BB0
0012B9B8	004A396C	Buffer = DOC_Rege.004A396C
0012B9BC	00000002	BytesToWrite = 2
0012B9C0	0012BCA0	pBytesWritten = 0012BCA0
0012B9C4	0012D23C	UNICODE "Kernel32.dll"
0012B9C8	00002710	

_F9 Third: run completely. Not CopyMEMII. Right click:

		Go to expression	Ctrl+G
		Follow in Disassembler	Enter
		Follow in Dump	
		Appearance	

_Patch To:

Address	Hex dump	ASCII
00487BB0	55 8B EC 6A FF 68 A0 2A 4A 00 68 88 78 48 00 64	U!wJ hã*J.hëxH.d
00487BB1	A1 00 00 00 00 50 64 89 25 00 00 00 83 EC 58	i....Pdë%....ãwX
00487BB3	53 51 57 89 65 E8 FF 15 78 D1 49 00 33 D2 8A D4	SUMãë\$ 3wTI.3πë
00487BB5	89 1A E0 3C 4A 00 8B C8 81 E1 FF 00 00 00 89 0D	ë\$α<J.i"üþ ...ë.
00487BB7	DC 3C 4A 00 C1 E1 08 03 CA 89 0D 08 3C 4A 00 C1	«<J.+þ...ë.†<J.+
00487BB9	E8 10 A3 D4 3C 4A 00 33 F6 56 E8 6B 16 00 00 59	3ü"ë<J.3+Ü3k...Y
00487BBB	85 C0 75 08 6A 1C E8 B0 00 00 00 59 89 75 FC E8	ã"üjL...Yëu"ë
00487BBD	36 13 00 00 FF 15 74 D0 49 00 A3 E4 52 4A 00 E8	6!!...3t"i.ü2RJ.ë

Address	Hex dump	ASCII
00487BB0	EB FE EC 6A FF 68 A0 2A 4A 00 68 88 78 48 00 64	...j hã*J.hëxH.d
00487BC0	A1 00 00 00 00 50 64 89 25 00 00 00 83 EC 58	i....Pdë%....ãxX
00487BD0	53 56 57 89 65 E8 FF 15 78 D1 49 00 33 D2 8A D4	SUMëë\$ \$wT1.3πë
00487BE0	89 15 E0 3C 4A 00 8B C8 81 E1 FF 00 00 00 89 0D	ë\$<J.i"üp ...ë.
00487BF0	DC 3C 4A 00 C1 E1 08 03 CA 89 07 D8 3C 4A 00 C1	■<J.±p"ë.†<J.±
00487C00	E8 10 A3 04 3C 4A 00 33 F6 56 EA 68 16 00 00 59	ë"ü"ë<J.3+Uëk...Y
00487C10	85 C0 75 08 6A 1C E8 B0 00 00 00 59 89 75 FC E8	ã"ü"jLë...Yëu"ë
00487C20	36 13 00 00 FF 15 74 00 49 00 A3 E4 52 4A 00 E8	6!!... \$t" I.ü2RJ.ë

_ Set WaitForDebugEvent bp breakpoint. F9:

Address	Value	Comment
0012BCAC	00477DB5	CALL to WaitForDebugEvent from I
0012BCB0	0012CD84	pDebugEvent = 0012CD84
0012BCB4	000003E8	Timeout = 1000. ms
0012BCB8	0012FF2C	
0012BCBC	00000000	
0012BCC0	7FFDF000	
0012BCC4	00000000	
0012BCC8	00000000	

_Right Click in 12BCAC:

_Toi Here:

Address	Hex dump	Disassembly	Comment
00477DB5	. 85C0	TEST EAX,EAX	
00477DB7	..0F84 23270000	JE DOC_Rege.0047A4E0	
00477DBD	. 8B95 FCFDFFFF	MOV EDX,DWORD PTR SS:[EBP-204]	
00477DC3	. 81E2 FF000000	AND EDX,0FF	
00477DC9	. 85D2	TEST EDX,EDX	
00477DCB	..74 12	JE SHORT DOC_Rege.00477DDF	
00477DCD	. A1 783A4A00	MOV EAX,DWORD PTR DS:[4A3A78]	
00477DD2	. 8378 20 00	CMP DWORD PTR DS:[EAX+20],0	
00477DD6	..74 07	JE SHORT DOC_Rege.00477DDF	
00477DD8	. C685 FCFDFFFF	MOV BYTE PTR SS:[EBP-204],0	
00477DDF	> 68 70324A00	PUSH DOC_Rege.004A3970	pCriticalSection = DOC_Rege.0
00477DE4	. FF15 A0D14900	CALL DWORD PTR DS:[<&KERNEL32.EnterCrit	EnterCriticalSection

_Patch To:

Address	Hex dump	Disassembly	Comment
00477DB5	50	PUSH EAX	
00477DB6	E8 7874A377	CALL kernel32.DebugActiveProcessStop	
00477DB8	90	NOP	
00477DBC	? 008B 95FCFDFF	ADD BYTE PTR DS:[EBP+FFFC95],CL	
00477DC2	? FF81 E2FF0000	INC DWORD PTR DS:[ECX+FFE2]	
00477DC8	? 0085 D27412A1	ADD BYTE PTR SS:[EBP+A11274D2],AL	
00477DCE	? 78 3A	JS SHORT DOC_Rege.00477E0A	
00477DD0	? 4A	DEC EDX	

_Nhin Through window FPU:

Registers (FPU)
EAX 00000001
ECX 0012BC9C
EDX 7FFE0304
EBX 7FFDF000
ESP 0012BCB8
EBP 0012D7A4
ESI 00002710
EDI 0012C30C
EIP 00477DBC DOC_Rege.00477DBC

_Ok, Eax = 1, detach success! Open a second OllyDBG: Attach.

Address	Hex dump	Disassembly	Comment
77F767CE	C3	RETN	
77F767CF	CC	INT3	
77F767D0	C3	RETN	
77F767D1	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]	
77F767D5	CC	INT3	
77F767D6	C2 0400	RETN 4	
77F767D9	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]	
77F767DF	C3	RETN	
77F767E0	57	PUSH EDI	
77F767E1	8B7C24 0C	MOV EDI,DWORD PTR SS:[ESP+C]	
77F767E5	8B5424 08	MOV EDX,DWORD PTR SS:[ESP+8]	
77F767E9	C702 00000000	MOV DWORD PTR DS:[EDX],0	
77F767EF	897A 04	MOV DWORD PTR DS:[EDX+4],EDI	

_F9, F12, Ctrl + E to edit EBFE 558B:

Address	Hex dump	Disassembly	Comment
00487BB0	55	PUSH EBP	
00487BB1	56EC	MOV EBP,ESP	
00487BB3	6A FF	PUSH -1	
00487BB5	68 00204000	PUSH DOC_Rege.004A2AA0	
00487BB8	68 88784800	PUSH DOC_Rege.00487888	
00487BBF	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00487BC5	50	PUSH EAX	
00487BC6	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00487BCD	83EC 58	SUB ESP,58	
00487BD0	53	PUSH EBX	
00487BD1	56	PUSH ESI	
00487BD2	57	PUSH EDI	
00487BD3	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00487BD6	FF15 78014900	CALL DWORD PTR DS:[<&KERNEL32.GetVersion	kernel32.GetVersion
00487BDC	33D2	XOR EDX,EDX	
00487BDE	8AD4	MOV DL,AH	

_Tim OEP. CreateThread bp Set, Shift + F9:

Address	Value	Comment
0012D76C	00AAFBF2	CALL to CreateThread from 00AAFBEC
0012D770	00000000	pSecurity = NULL
0012D774	00000000	StackSize = 0
0012D778	00AB041E	ThreadFunction = 00AB041E
0012D77C	00000000	pThreadParm = NULL
0012D780	00000000	CreationFlags = 0
0012D784	0012D78C	pThreadId = 0012D78C
0012D788	004A3568	DOC_Rege.004A3568

_Ctrl + F9, F8, Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
00AC9A70	A1 F8E8AD00	MOV EAX,DWORD PTR DS:[AD529C]	
00AC9A75	59	POP ECK	
00AC9A76	8B48 74	MOV ECK,DWORD PTR DS:[EAX+74]	
00AC9A79	3348 58	XOR ECK,DWORD PTR DS:[EAX+58]	
00AC9A7C	3348 4C	XOR ECK,DWORD PTR DS:[EAX+4C]	
00AC9A7F	F6C1 40	TEST CL,40	
00AC9A82	75 08	JNZ SHORT 00AC9A8C	
00AC9A84	6A 01	PUSH 1	
00AC9A86	E8 C1D1FDFF	CALL 00AA6C4C	
00AC9A88	59	POP ECK	
00AC9A8C	6A 00	PUSH 0	
00AC9A8E	C705 9C52AD00 D	MOV DWORD PTR DS:[AD529C],0AD5FD0	ASCII "RC"
00AC9A98	E8 DAFCFDFF	CALL 00AA9777	
00AC9A9D	59	POP ECK	

_Cuon Down:

Address	Hex dump	Disassembly	Comment
00AC9B02	8B48 5C	MOV ECK,DWORD PTR DS:[EAX+5C]	
00AC9B05	3348 58	XOR ECK,DWORD PTR DS:[EAX+58]	
00AC9B08	3348 24	XOR ECK,DWORD PTR DS:[EAX+24]	
00AC9B0B	2BF9	SUB EDI,ECK	
00AC9B0D	FFD7	CALL EDI	
00AC9B0F	8BD8	MOV EBX,EAX	
00AC9B11	5F	POP EDI	
00AC9B12	8BC3	MOV EAX,EBX	
00AC9B14	5E	POP ESI	
00AC9B15	5B	POP EBX	
00AC9B16	C3	RETN	

_Nhan Here F2, F9, F7: OEP:

Address	Hex dump	Disassembly	Comment
00401378	EB 10	JMP SHORT DOC_Rege.0040138A	
0040137A	66:623A	BOUND DI,DWORD PTR DS:[EDX]	
0040137D	43	INC EBX	
0040137E	2B2B	SUB EBP,DWORD PTR DS:[EBX]	
00401380	48	DEC EAX	
00401381	4F	DEC EDI	
00401382	4F	DEC EDI	
00401383	4B	DEC EBX	
00401384	90	NOP	
00401385	E9 98104300	JMP 00832422	
0040138A	A1 8B104300	MOV EAX,DWORD PTR DS:[43108B]	
0040138F	C1E0 02	SHL EAX,2	
00401392	A3 8F104300	MOV DWORD PTR DS:[43108F],EAX	
00401397	52	PUSH EDX	
00401398	6A 00	PUSH 0	
0040139A	E8 01ED0200	CALL DOC_Rege.004300A0	
0040139F	8B00	MOV EDX,EAX	
004013A1	E8 CE960100	CALL DOC_Rege.0041AA74	
004013A6	5A	POP EDX	
004013A7	E8 FCE00200	CALL DOC_Rege.004301A8	
004013AC	E8 07970100	CALL DOC_Rege.0041AAB8	JMP to CC3260MT.__CRTL_MEM_Us

_Den Find the IAT: The OEP: Ctrl + B to enter FF25. Click OK to us here:

Address	Hex dump	Disassembly	Comment
0042F32C	FF25 D0F44300	JMP DWORD PTR DS:[43F42D]	vc160.@Consts@Initialization\$qqrv
0042F332	FF25 D4F44300	JMP DWORD PTR DS:[43F4D4]	vc160.@Consts@Finalization\$qqrv
0042F338	FF25 6CF54300	JMP DWORD PTR DS:[43F56C]	vc160.@Graphics@Initialization\$qqrv
0042F33E	FF25 70F54300	JMP DWORD PTR DS:[43F570]	vc160.@Graphics@Finalization\$qqrv
0042F344	FF25 74F54300	JMP DWORD PTR DS:[43F574]	vc160.@Graphics@TBitmap@SetPixelF
0042F34A	FF25 78F54300	JMP DWORD PTR DS:[43F578]	vc160.@Graphics@TBitmap@SetHandle
0042F350	FF25 7CF54300	JMP DWORD PTR DS:[43F57C]	vc160.@Graphics@TBitmap@GetScanl
0042F356	FF25 80F54300	JMP DWORD PTR DS:[43F580]	vc160.@Graphics@TBitmap@GetPixelF
0042F35C	FF25 84F54300	JMP DWORD PTR DS:[43F584]	vc160.@Graphics@TBitmap@GetPixelF
0042F362	FF25 88F54300	JMP DWORD PTR DS:[43F588]	vc160.@Graphics@TBitmap@GetPixelF

IAT _Du guess is in the mind 43FXXX. Here Follow Print dump> Memory Address, we obtained information as follows:

IAT Start:

Address	Value	Comment
0043F4D0	400B1EDC	vc160.@Consts@Initialization\$qqrv
0043F4D4	400B1EAC	vc160.@Consts@Finalization\$qqrv
0043F4D8	00AAACC7	
0043F4DC	00040DC9	
0043F4E0	00040DE9	
0043F4E4	00040E07	
0043F4E8	00040E45	
0043F4EC	00040E69	

_IAT End:

Address	Value	Comment
00440AFC	01681518	WINREGP.InitializeDll
00440B00	01684D8C	WINREGP.ReadSector
00440B04	016818F8	WINREGP.GetLogicalDiskGeometry
00440B08	00AAAC23	
00440B0C	366C6376	
00440B10	70622E30	
00440B14	6376006C	
00440B18	2E30366C	

_Dong Window OllyDBG 2. Restart OllyDBG 1, as similar steps to debug by pass blocker. Open OllyDBG 2, In dump window: Ctrl + G 0043F4D0, Set breakpoint on write. Shift + F9 2 times to me here:

Address	Hex dump	Disassembly	Comment
00AC6900	8B85 10C8FFFF	MOV EAX,DWORD PTR SS:[EBP-37F0]	DOC_Rege.0043F4D0
00AC6913	83C0 04	ADD EAX,4	
00AC6916	8985 10C8FFFF	MOV DWORD PTR SS:[EBP-37F0],EAX	
00AC691C	^E9 CFCFFFFF	JMP 00AC65EF	
00AC6921	FF15 9CF2AC00	CALL DWORD PTR DS:[ACF29C]	kernel32.GetTickCount
00AC6927	2B85 A0C4FFFF	SUB EAX,DWORD PTR SS:[EBP-3B60]	
00AC692D	8B8D A4C4FFFF	MOV ECX,DWORD PTR SS:[EBP-3B5C]	
00AC6933	6BC9 32	IMUL ECX,ECX,32	

_Cuon Up. Remember 0040139A.

Address	Hex dump	Disassembly	Comment
00AC676D	8B85 58C2FFFF	MOV EAX,DWORD PTR SS:[EBP-3DA8]	
00AC6773	FF30	PUSH DWORD PTR DS:[EAX]	
00AC6775	E8 8E17FEFF	CALL 00AA7F08	
00AC677A	83C4 0C	ADD ESP,0C	
00AC677D	8D85 58C1FFFF	LEA EAX,DWORD PTR SS:[EBP-3EA8]	
00AC6783	50	PUSH EAX	
00AC6784	FFB5 60C2FFFF	PUSH DWORD PTR SS:[EBP-3DA0]	
00AC678A	FF15 50F3AC00	CALL DWORD PTR DS:[ACF350]	msvcrt._stricmp
00AC6790	59	POP ECX	
00AC6791	59	POP ECX	

_Lap The steps. After opening up the OllyDBG 2, set breakpoint at 0043F4D0, Shift + F9. Ctrl + G 00AC6775, he set one here, F9, break, go to the function call, edit 55 to C3. Ctrl + G to OEP. Set

in a he. F9.

Address	Hex dump	Disassembly	Comment
00401378	EB 10	JMP SHORT DOC_Regenerator.0040138A	
0040137A	66:623A	BOUND DI,DWORD PTR DS:[EDX]	
0040137D	43	INC EBX	
0040137E	2B2B	SUB EBP,DWORD PTR DS:[EBX]	
00401380	48	DEC EAX	
00401381	4F	DEC EDI	
00401382	4F	DEC EDI	
00401383	4B	DEC EBX	
00401384	90	NOP	
00401385	E9 98104300	JMP 00832422	

_ LordPE, Full dump. ImpREC. Enter information OEP, IAT auto search, you receive:



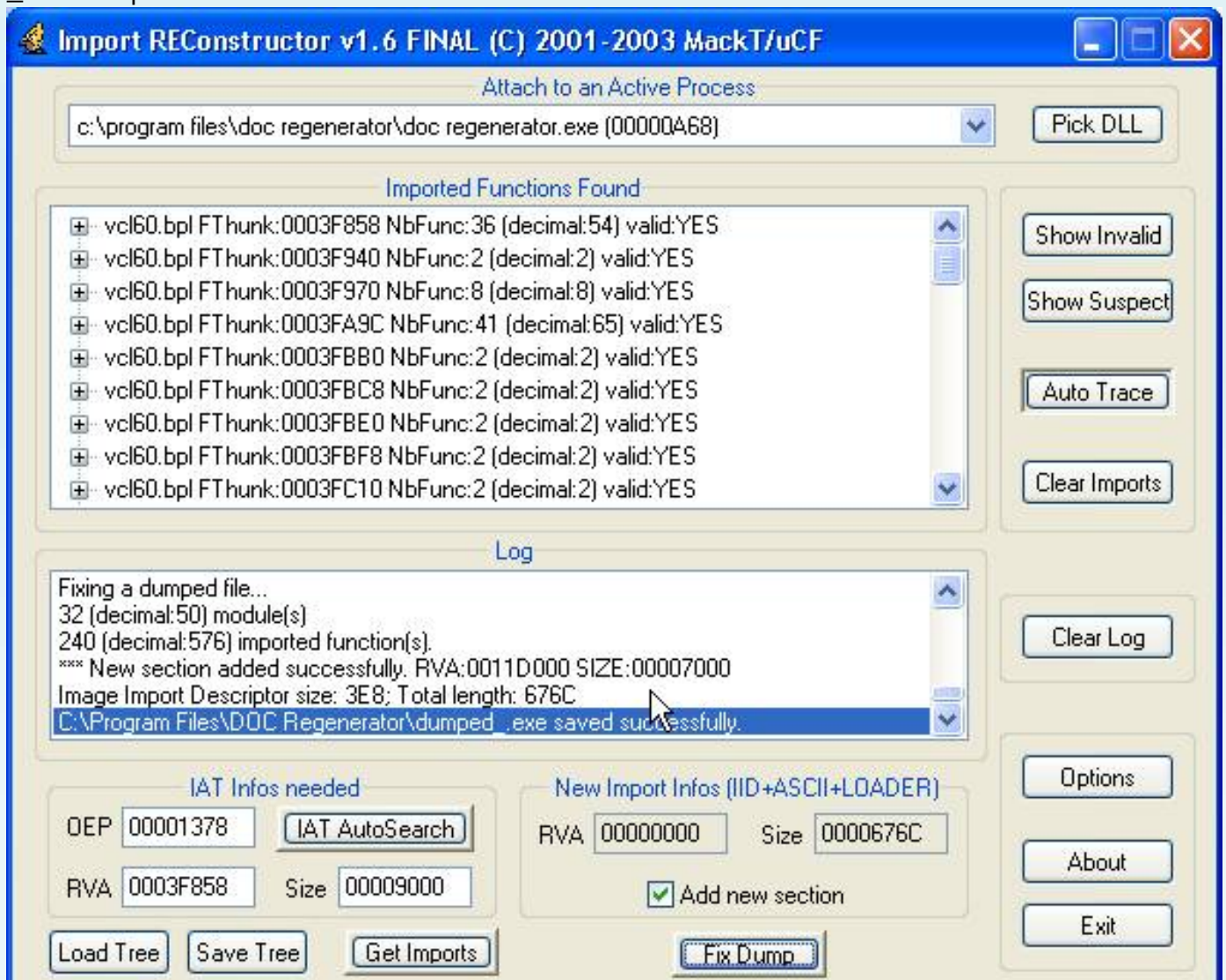
_Bo Last Show Invalid, Thanks Cut. Fix Dum try and run, Crash! Lack of function. ImpREC Open again, enter information such as table notification requirements or enter exactly as follows:

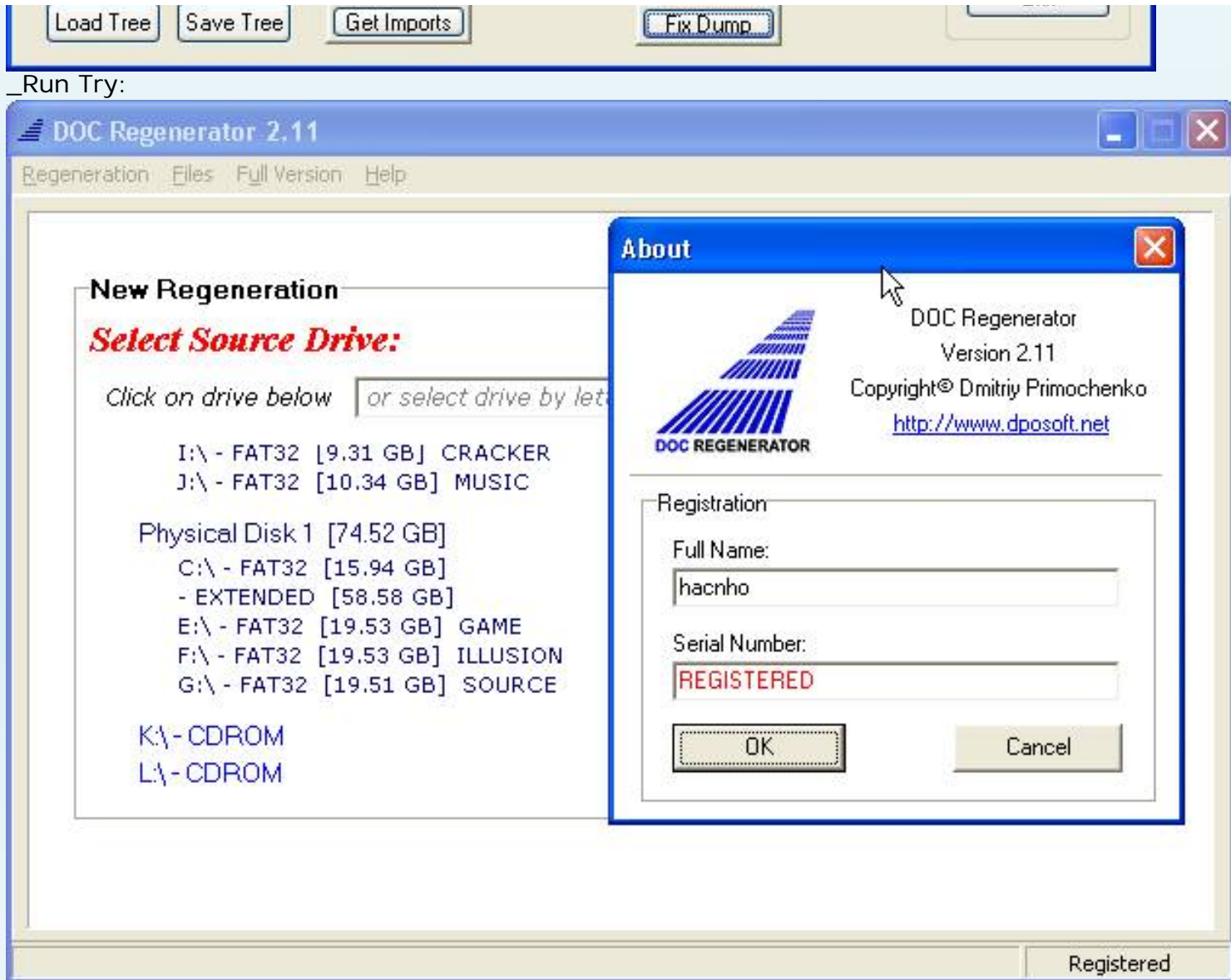
OEP: 00001378

IATRVA: 0003F4D0

IATSize: 000012AC

_Fix Dump:





File Dumped.exe have great size. You can do the following month to the small size by using CFF Explorer:

CFF Explorer 1 - by Ntoskrnl - [dumped_.exe]

File Settings ?

File: dumped_.exe

- Dos Header
- Nt Headers
- File Header
- Optional Header
- Data Directories [15]
- Section Headers [x]
- Export Directory
- Import Directory
- Resource Directory
- Address Converter
- Rebuilder
- Resource Viewer
- Hex Editor

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Addr
Byte[8]	Dword	Dword	Dword	Dword	Dword
.data	0000C000	00031000	0000C000	00031000	00000000
.tls	00001000	0003D000	00001000	0003D000	00000000
.rdata	00001000	0003E000	00001000	0003E000	00000000
.idata	00009000	0003F000	00009000	0003F000	00000000
.edata	00011000	00048000	00011000	00048000	00000000
.reloc	00004000	00059000	00004000	00059000	00000000
.text1	00030000	0005D000	00030000	0005D000	00000000
.addata	00013000	0008D000	00010000	0008D000	00000000
.data1	00010000	0009D000	00010000	0009D000	00000000
.reloc1	00010000	000AD000	00010000	000AD000	00000000
.pdata	00050000	000BD000	00050000	000BD000	00000000
.rsrc	00010000	0010D000	00010000	0010D000	00000000
.mach	00007000	0011D000	00007000	0011D000	00000000

_Done!

MUP: Follow the ExeCryptor_2.2.x_2.3.x (tut's Evolution)

kienmanowar

Thread RLPack Com's exciting too, but the king is *thằng* EC Test try for the aged by tut Evolution at the time not. But then have the Moth try try, hehe Moth they should not kill thread that still run delicious:)). But mainly discuss this article is follow the steps in the Evolution of the aged. Introduction to the EC and its features, the tools attached to perform the MUP brother read in tut.Thread only focus on how the aged Evolution made. Brothers have ideas what the comment nhé!

1. Find information about the Compiler:

This is the first step and is also a very important step to open to OEP based on standard procedures of the compiler, so the need to understand the information needed before further implementation.

Run the file by ExeCryptor pack, then open LordPE process to select and choose **Imagesize Correct**. Note recorded information that LordPE provided. If lack not achieved this step will not be able to create a raw dump. Next LordPE used to dump full program, then use the detector to search for information about the target. RDG will use the results are quite accurate. Through detect this we will know the information about the compiler is more accurate. Then will learn about the structure of the EP (Delphi, VC + +) to find the OEP.

2. Search OEP and OEP near procedure:

There are many ways to find the OEP the best way is to analyze the procedure in the target and to nearly EP. This can be applied to all the protectors. Before the start I wanted to make everything very clearly, so I will try to explain how how to find the OEP for the compilers different ... we are currently working with Delphi Compiler , is the task we are trying to get to the OEP. To be able to find the OEP of the other compilers more the *Appendix A*.

Standard Dephi OEP:

```

PUSH EBP ← OEP
MOV EBP, ESP
ADD ESP, -10
MOV EAX, address ← this instruction is the first really needed to make program works
CALL Sysinit::initexe Procedure ← this is the first procedure of ANY Delphi program
MOV EAX, DWORD PTR DS: [dword]
MOV EAX, DWORD PTR DS: [EAX]
CALL procedure1
MOV ECX, DWORD PTR DS: [dword]
MOV EAX, DWORD PTR DS: [dword]
MOV EAX, DWORD PTR DS: [EAX]
MOV EDX, DWORD PTR DS: [dword]
CALL procedure2
MOV EAX, DWORD PTR DS: [dword]
MOV EAX, DWORD PTR DS: [EAX]
CALL procedure3
CALL procedure4

```

Image is a typical example for Delphi EP, noted that all the Delphi have at least one function CALL first nhau. Nhu same way that we are sure that the function **sysinit: initexe** will be implemented in the program, so if you find it in our target then we will find the OEP quickly. If we go deeper into the function **Sysinit: 2 initexe** the ability that we found as follows:

For the target form Delphi 4-5

```

PUSH EAX
PUSH 0
CALL GetModuleHandleA ← this call to GetModuleHandleA will be important
MOV EDX,dword1
PUSH EDX
MOV DWORD PTR DS:[dword2],EAX
MOV DWORD PTR DS:[EDX+4],EAX
MOV DWORD PTR DS:[EDX+8],0
MOV DWORD PTR DS:[EDX+C],0
CALL procedure1
POP EDX
POP EAX
CALL procedure2
RETN

```

For the target form Delphi 6-7

```

PUSH EBX
MOV EBX,EAX
XOR EAX,EAX
MOV DWORD PTR DS:[dword1],EAX
PUSH 0
CALL GetModuleHandleA ← this call to GetModuleHandleA will be important
MOV DWORD PTR DS:[dword2],EAX
MOV EAX,DWORD PTR DS:[dword3]
MOV DWORD PTR DS:[dword4],EAX
XOR EAX,EAX
MOV DWORD PTR DS:[dword5],EAX
XOR EAX,EAX
MOV DWORD PTR DS:[dword6],EAX
CALL procedure1
MOV EDX,dword7
MOV EAX,EBX
CALL procedure2
POP EBX
RETN

```

Depending on the type of target that we rely on to search through **GetModuleHandleA** function. Load the file has been in full dump Olly, press *Ctrl + G*, type in 401000 then analyze code starting with the address. Next find all intermodular call, sorted by name and find all functions GetModuleHandleA (as it usually refers 2). If the function Follow Call will have code as follows:

```

00406EBC. 50 PUSH EAX
00406EBD. 6A 00 PUSH 0; / pModule = null
00406EBF. E8 F8FEFFFF CALL dumped.00406DBC \ GetModuleHandleA
00406EC4. THREE 08414D00 MOV EDX, dumped.004D4108
00406EC9. 52 PUSH EDX
00406ECA. 8905 DC744D00 MOV DWORD PTR DS: [4D74DC], EAX
00406ED0. 8942 04 MOV DWORD PTR DS: [EDX +4], EAX
00406ED3. C742 08 00000> MOV DWORD PTR DS: [EDX +8], 0
00406EDA. C742 0C 00000> MOV DWORD PTR DS: [EDX + C], 0
00406EE1. E8 8AFFFFFF CALL dumped.00406E70
00406EE6. 5A POP EDX
00406EE7. 58 POP EAX
00406EE8. E8 3FCBFFFF CALL dumped.00403A2C
00406EED. C3 RETN

```

If the form on the right as it is are the functions **Sysinit: initexe Procedure**. To be able to find the OEP, then we

must see the OEP in the form of a target not protect. Almost 95% are similar to the illustrations above and it is usually near the end of the code section! How to find the OEP in our target? Very simply, in our Olly mouse scroll down the code and find the section to the string "ExeCryptor" (press Ctrl + B and sample: 45 91 3D 6F 43 52 5F 35). After finding out this is translated to 1 billion by the OEP program.

```
004D2AAC \ 20274D00 DD dumped.004D2720
004D2AB0 00 DB 00
004D2AB1 00 DB 00
004D2AB2 00 DB 00
004D2AB3 00 DB 00
004D2AB4 70274D00 DD dumped.004D2770
```

3. Dump program:

As we have on the 2 address important program that is, OEP and Sysinit: initexe procedure. Work next week will automatically as what we did when unpack a target that is at the dump near OEP. To do this, there are a lot of different methods, for example, we get the program loaded into Olly and try to remove the anti-debug. Some Scripts help us to bypass the anti-debug but when they work is not always accurate. Normally I use another way to the dump site near the OEP, and that is the simplest way that I want to present in this article.

Because we know the location of the functions need to know (Sysinit: initexe), and we also make sure it was implemented at the same time it is also close to OEP, so we can change a few bytes in the position where it starts and then Attach process, implementation and dump v.. v.. In the case of our target, the procedure Sysinit: Initexe start address is: 0x00406EBC and we have information about it as follows:

```
00406EBC. 50 PUSH EAX
00406EBD. 6A 00 PUSH 0; / pModule = null
00406EBF. E8 F8FEFFFF CALL dumped.00406DBC \ GetModuleHandleA
00406EC4. THREE 08414D00 MOV EDX, dumped.004D4108
```

The work to be done now is what?

Very simply, we need a loader so that it will wait until the program is completely unpacked in memory and then conducted two bytes in patch 0x00406EBC: 50 6A ---> EB FE.

When we conducted this month patch program will fall on state repeats endlessly at start of Sysinit: initexe, and so we can conduct dump file is close to places of OEP. To be able to create a loader is generally the best way is that you have to code for yourself, but remember that if you decide to create a separate tool you need to pay attention to the EC anti-debug tricks, because if you make a simple way the program you can terminate within a few seconds. ***More and six B to better understand how to break the anti-debug and run by the EC Debugger in the program (in particular in Olly)***

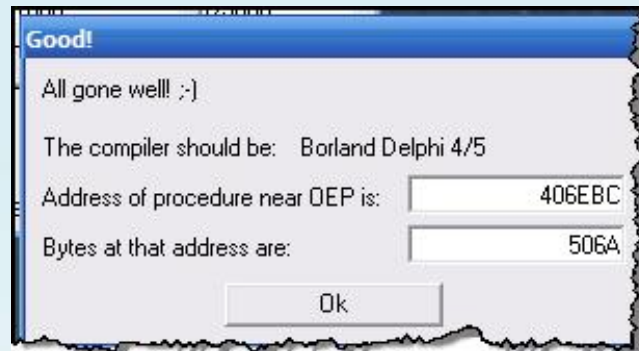
The creation of loader different each time we perform a file unpack any really is not possible and take a long time so I wrote 2 program allows us to find OEP easily that ko difficult time. It is "EC Procedure Finder 1.0" and "EC 1.0 OEP Break." Part I will be more guidance to use this tool 2.

First open EC 1.0 Procedure Finder, click Browse and select our target. The two requirements address the "Base of Code" and "Size of Code". For the two values we open this file is in LordPE protect and search for information in the Sections. Image Base because of the program is 0x00400000, we have 2 values that we need for the EC is Procedure Finder, for example:

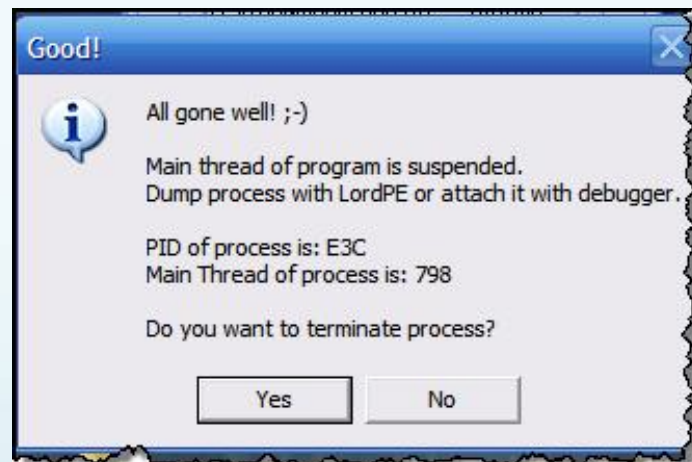
Base of Code (VA) = 401000

Size of Code = D3000

Enter two values on to, then press Go, do not worry if you open the program, this completely thuong.Sau that average (depending on the processor's speed and size of code) you will receive the following :



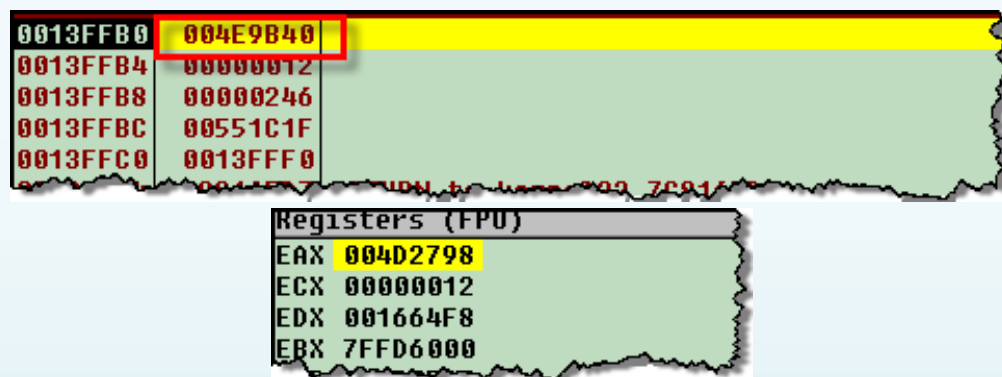
Through analysis of the above, we already know this, but because this is a message and I have to explain what I've learned in the process of unpacking this program in other cases you can always use the tool to find without having to step analysis manually as above. Now we want to how to dump the program from 0x00406EBC, this fully achieved if you follow Appendix B but not really necessary because we have a second to make this job for us. Now open the "EC 1.0 OEP Break" and select the target, enter the value that we are on, press Go and wait a while until the notice:



Now programs we stopped at the position close to OEP and this can be completed without using the debugger. Thong to report on the show to see our information on PID and Main Thread of the process, the this value is really do need however they can still be useful for me Now we click on it because I want at the dump before I close it:). Attach the program to Olly, the Attach fully implemented correctly! Now you will stop me in command after JMP Attach completed, the next press Shift + F9 to program implementation by the toan.Tiep press Pause and return to Main Thread which now has been suspend (resume again this thread), then open the window seat CPU main thread, we here in Olly:

```
00406EBC 50 PUSH EAX
00406EBD 6A 00 PUSH 0
00406EBF E8 F8FEFFFF CALL 00406DBC
00406EC4 THREE 08414D00 MOV EDX, 4D4108
00406EC9 52 PUSH EDX
```

Wow! We have the right where we need to, will now conduct target dump site is close to OEP! Dump programs use memory OllyDump plugin and select "Import rebuild," save as optional. Ok, we've implemented in the dump file is near the most OEP, the next record is about to write and EAX Stack value, then we will need it:



Address on the Stack is necessary because, based on which we know the return address of Sysinit: initexe. The value of EAX to write is very important because it will be used to later if we want to rebuild the code at the OEP, and it also helps you find the address of the OEP of the program. Summarized again in this step will be:

Return from Sysinit Address: initexe: 0x004E9B40

Value of EAX: 0x004D2798

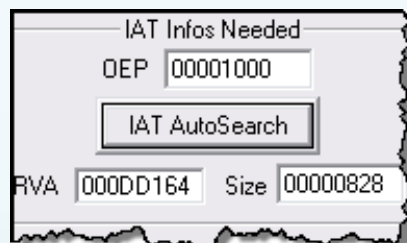
Ok, so we have a dump file is ... due as expected. Section next to rebuild the import, but not close it for Olly stop at on.

4. Rebuild Imports

Previously we had to dump positions near OEP, but clearly it will not work until we fix what some thu. Viec first, like any unpack any steps that we need to build the table of the IAT. To do this, there are 2 different methods are:

1. Use of Deroko plugin (this plugin is used in ImpRec)
2. Use of Scripts PEKill

Both are on the way very well, but to make a quick I choose the second, also because of the reasons is our Olly is still open and hold the first step so we can complete the that does not encounter problems. However, before you can use this script is the one to provide information about the IAT IAT start and end. For this value is completely can read directly from the memory through Olly or use for a ImpRec automatically. Here I use ImpRec, open process in place and ImpRec 0x1000 as OEP RVA (may also include real found OEP), click Auto Search IAT we are as follows:



VA Start of IAT is: 004DD164

VA End of IAT is: 004DD164 + 828 = 004DD98C

Ok now open PEKill's script through an editor and change the two values that we've found in the script. After the edit is done, we go back and make Olly run this script to recover the whole of the IAT. Wait a while to finish the complete script (depending on the speed of your computer). After the IAT are full, we will fix the dumped file. ImportRec open up, select Process and complete information about OEP. Next click Get Imports ... all imports were completely resolve including GetProcAddress function to emulate by the EC. Click Fix and dumped dump file to fix IAT.

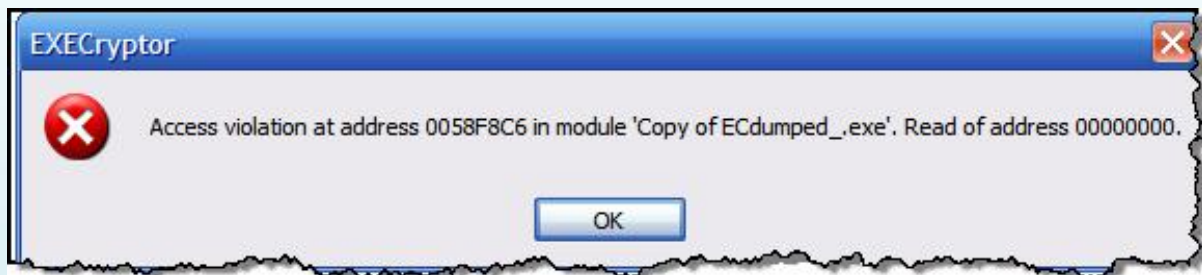
5. Fix TLS PE and Directory

Ok, we fix almost all programs and almost completely unpack and can be implemented (but is only on our servers), but before you take the next step we need to adjust a few prices LordPE therapy. Open the dump file import fix in the PE Editor. Value must correct here is BaseOfCode (0x1000 into edit and save again). Next to the Sections and record information. "Rdata" section. You interested in it because of the value of Delphi this is exactly the value of RVA TLS Directory. The reason that we open the Directories and change the value of the TLS RVA value of that. Rdata we have above. Finally Save the file, so that we have completed the fix PE.

Remember that fix the exact value of the TLS is very important, but not all programs are using it, but for the product / program code with Borland TLS is always used. Because this reason, if you unpack a program code with Borland C++ or Borland Delphi, please remember to fix the value of TLS with the value of RVA. Rdata section. But if you unpack the program is not how to fix Borland the best is set the entire value of the TLS (RVA and size) to 0 ... remember this is very important, because if you do not fix TLS by the EC will be implemented before the OEP and programs to our enforcement is not right and therefore we will have difficulty in debug the program

6. Set the correct OEP or rebuild a piece of it

Remember back in the first one found by the OEP, which is in step 2 we have found is 0x004D2AB8. Therefore we need to revise the value of the Entry Point 0x000D2AB8 with LordPE, the program will run immediately but: | This time we will encounter an error as follows:



If in the process of implementation that you encounter this error, please do not worry too, is a mechanism of anti-debug EC we will find ways to bypass them in the next. The most important part is we have run the file is dumped. As said in the introduction, they do not need Moth defeat the anti-debug thẳng that fix is in addr 0x0058F8C6 program will run as normal. However, the objective of the dicuss as mentioned is to follow the steps of aged Evolution so I will not go off as Moth: D.

I want to show you another way, because in the EC to unpack, we do not want is to find the OEP of the program by simply want the best, but also restore a part of the original OEP. This result is not hard to do but need to observe the fine plus the appendix A to solve the problem. In this article, the EP by the Delphi compiler as we know Delphi 4 / 5 of that OEP will form similar to the following:

```
PUSH EBP ← OEP
MOV EBP,ESP
ADD ESP,-0C
MOV EAX, address ← this instruction is the first really needed to make program works
CALL SysinitProcedure ← this is the first procedure of ANY Delphi program
...
```

Observations on the Figure, you notice that these values need to find we have found in Section 3. Based on these values as we rebuild the following:


```

PUSH EBP ← OEP
MOV EBP,ESP
ADD ESP,-0C
MOV EAX, 004D2798 ← the value of EAX when entering the SysInit::initexe procedure
CALL 00406EBC ← call to SysInit::initexe

```

OK next to what? We need to rebuild all the other parts of the OEP? If you enough patience and time is completely possible, but it is not necessary and even the very time, can be difficult ... the best method is to order a dance Return directly to the functions Sysinit Address: initexe. Finally we OEP will be as follows:

```

PUSH EBP
MOV ESP, EBP
ADD ESP, -10
MOV EAX, 004D2798
CALL 00406EBC
JMP 004E9B40

```

So to be able to rebuild the code this we need a space just enough to perform. Here I choose 0x004D2B48 and fix them.

004D2B34	. FFFFFFFF	DD FFFFFFFF
004D2B38	. 0A000000	DD 0000000A
004D2B3C	. 45 58 45 43 7;	ASCII "EXECryptor",0
004D2B47	. 00	DB 00
004D2B48	. 55	PUSH EBP
004D2B49	. 8BEC	MOV EBP,ESP
004D2B4B	. 83C4 F0	ADD ESP,-10
004D2B4E	. B8 98274D00	MOV EAX,dumped_.004D2798
004D2B53	. E8 6443F3FF	CALL dumped_.00406EBC
004D2B58	. -E9 E36F0100	JMP dumped_.004E9B40
004D2B5D	. 00	DB 00
004D2B5E	. 00	DB 00

After you edit the code as the one to save what has changed, then use LordPE and fix the value of EP D2B48. Then try to run the program.

In fact I see no need to fix this khúc, older Moth straight addr fix that indoctrination is sticky. However, a stranger is not to fix them is not break in CreateThread API to kill the thread. I try try to stay by the EP D2AB8, then Bp in CreateThread not.

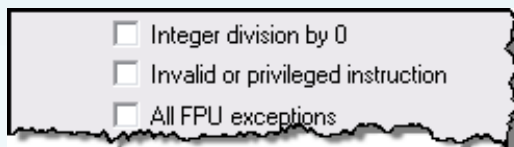
7. Defeating Anti-Debug:

Usually when a file was unpack, you will not have any one of any anti-debug and fix has the file will run normally without problems. But the EC is the other, even when you completely remove the packing layer, but when the program will still fail and do not work properly.

The reason this problem is whether we unpack the target, but even how to go when the EC code is implemented in different places in the program which means that EC can create protection thread, search for the debuggers and the implementation of which we never expected. : (

Naturally, we must completely leave out the mechanism to protect this to complete the unpack as well as to have a perfect file.

As I said above, the protection that EC use is to create threads of protection so if we find out how it is we can run the program normally, at least on Our machine. To do the first configuration we have the Olly, select the Exceptions follows:



However, before the file to load dumped Olly, we need to make some changes in the section of the file. Open file dump fix was in LordPE, to the sections and drag down to the last:

[Section Table]					
Name	VOffset	VSize	ROffset	RSize	Flags
.tls	000E1000	00001000	000E1000	00001000	C0000000
.rdata	000E2000	00001000	000E2000	00001000	C0000040
bsqsutju	000E3000	0000E000	000E3000	0000E000	E0000060
.rsrc	000F1000	0001A000	000F1000	0001A000	C0000040
tzqpqpqs	0010B000	00107000	0010B000	00107000	E0000020
t4umo1p6	00212000	0015D000	00212000	0015D000	E0000060
29wy8q2h	0036F000	00001000	0036F000	00001000	40000080
.mact	00370000	00003000	00370000	00003000	E0000060

We must identify the sections added by the EC, the EC will contain code. If map many of you will see to that. "Rsrc" section is the final section of the Delphi, and ". Mact" section is by ImportRec added. So clearly that the 3 sections in the two section. "Rsrc" and ". Mact" are the main sections of the EC more vao.Nhiem that we need to do is move them into 3 sections 1, this is too simple. You get a total of Vsize of 3 sections, then correct the value that we have to Vsize Rsize and sections of the first, we wipe the last 2 sections below it. Ok, after merge the sections into one, we open up and Olly programs to load. I will stop at the EP, now we will kill the threads, so we need to know the API function that EC use:

CreateThread → used to create protection threads
 WaitForSingleObject → used to control if threads are alive

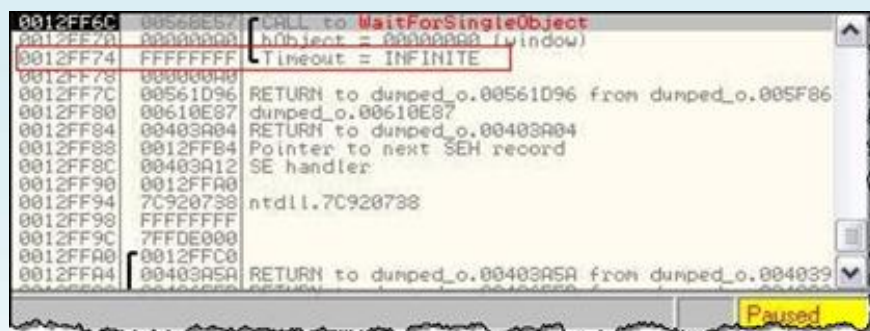
First we put a memory in me BP's first order CreateThread function and press F9 to implement the program. To that Olly will stop a number of memory when reading, you just press F9 to see when the Status bar by Olly appeared the "Memory breakpoint when executing" the time to look at the Stack window:



To fold for red, the value of this function are described by thread implementation that will be created, so if we stop it, the thread will not do anything. Ok, through CPU window and press Ctrl + G , followed by entering the value we find the will to get to the code. So there how do we kill duocc thread.Rat simply found the order is first order with a CALL command RETN (better is the change in RETN 4).

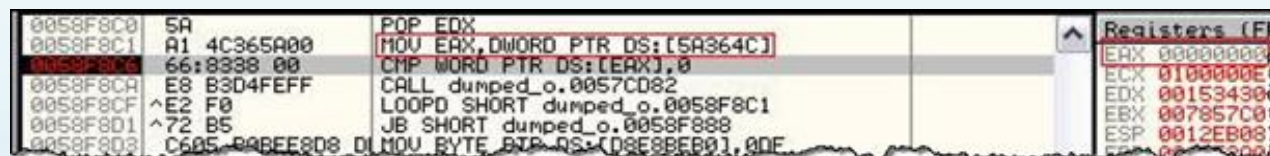
After the patch is finished, save the file and the file to load Olly.Neu we try to implement the program at this time it will not implement itself and it is "close by." Why? To find out the cause is set to a Memory BP

in WaitForSingleObject function and press F9. To appear when "Memory executing breakpoint when it stopped, looked through the window Stack:



The value of Timeout this time is infinite, so the program will wait for an object .. but this object is created by thread protection. Do we kill thread protection, object will never be created and therefore the program will not start, so we must make way? Tolerable Evolution has aged a day of heaven and the aged we discovered that the value Timeout nau was set by me Push command -1. Therefore we need to find all the commands Push -1. Open Memory Map window, follow the EC section and find all the commands Push-1. Tiep theo set a BP on every command in order to restart and Olly.

Press F9 to implement the program, then stop when Olly only change the -1 to Push Push 0. This is not fixed it depends on the different target. After you have finished editing the save a file to another file to run the test. Immediately we failed to meet the Access indoctrination, recorded address that notification was sold. Load program to set and Olly bp at finding and implementing programs to stop in place when we set BP.



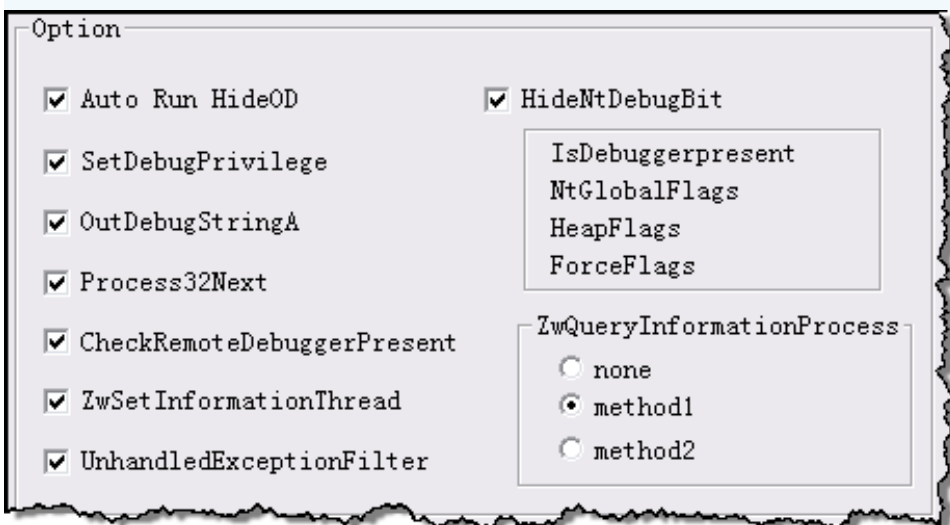
Note red seats fold, read the fix in tut's aged Evolution fix. Sau to complete the fix, try the test file was run fix it lickerish !!!!!!! Finish this stage is then unpack the files can only be implemented on our computer but can not run on other machines.

Appendix: Run on target Debugger

In the first I did not count for much to EC anti-debug tricks is because there have been many articles explaining how to bypass, even had to have 2, 3 scripts (which I especially like the script of Hagggar) can achieve this. One important than all that in about 90&percent; of cases we do not need to run target in the debugger, simply by creating a loader to break in or use OEP 2 tools were above ... this will save much time, simply and quickly.

As I said in the previous section, the protection of the EC is based very much on protection threads that have been implemented before and during implementation of the program. So the only thing we need to do to be able to run the target of a break in olly threads. There are some other tips that small EC to use anti-debug, but we will use a plugin to overcome the present procedure nay. Dau first we use a program through Olly may face as the EC or OllyIce Olly ExeCryptor.

Olly Now we open up and refine the options of HideOD plugin as follows:



Next edit Olly to stop at "Breakpoint System" rather than in "Entry point of main module" (This is because if you try to stop at the packer's EP EC TLS function will be implemented and the anti - debug will detect Olly, so we need action before implementation Function TLS). Now, you can follow the steps to run different programs. I will explain how the fastest point of the load toi.Dau Olly to target and then remove the BP System (If you do not remove the program to be close less than 1 second):

Address	Module	Active	Disassembly
0076E427	EXECrypt	One-shot	CALL EXECrypt.0076E323

Then we observe and think a bit about threads There are two questions need to set answer:

1. A Thread is created like?
2. How to prevent it created?

The questions really easy answer, a thread is created using "CreateThread" API "or" CreateRemoteThread "API or" zwCreateThread "API. It is fortunate I absolutely sure that the EC uses only function "CreateThread" to create mechanisms to protect its (Protection threads).

To hand over the EC and not to create threads it is very simple, completely made by me to hook the first order function "CreateThread". In Olly we observe the code in CreateThread addr, we found as follows:

7C810637	8BFF	MOV EDI,EDI
7C810639	55	PUSH EBP
7C81063A	8BEC	MOV EBP,ESP
7C81063C	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
7C81063F	FF75 18	PUSH DWORD PTR SS:[EBP+18]
7C810642	FF75 14	PUSH DWORD PTR SS:[EBP+14]
7C810645	FF75 10	PUSH DWORD PTR SS:[EBP+10]
7C810648	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]
7C81064B	FF75 08	PUSH DWORD PTR SS:[EBP+08]
7C81064E	6A FF	PUSH -1
7C810650	E8 D7FDFFFF	CALL kernel32.CreateRemoteThread
7C810655	5D	POP EBP
7C810656	C2 1800	RET 18

I found this function from the start mov edi, edi -> 18 Ret. What we need to do to stop the implementation of the code is modified MOV EDI, EDI's RET 18 and so EC will not be creating more threads. If the target is to protect equal version to version 2.2.9, the following steps to edit this is absolutely can run programs and unpack without problems.

However the version we are working as 2.3.x and this is the final so we will encounter some problems occur suddenly during the run target. Yes that's right, if you try to implement the program now it will stop after a few seconds and code enforcement is interrupted ... Why?

This is because there is a procedure that EC used to control the threads are alive, and if the threads have been stopped or terminated, the implementation process will stop.

The reason this is quite the gian. Phien the last EC use some "events" in the implementation of the program, and for each (not all) thread is creating a new event will also be created. So the problem is the EC would like to control if the thread is 1 exist, will wait event has been created ... and simply thread if not created, the event is not generated and therefore the will wait. waiting in endless time: (. Fortunately in this case we can make a hook procedure simple to solve this problem. It functions as a "WaitForSingleObject" will be used to life event. View code is as follows:

7C802520	8BFF	MOV EDI,EDI
7C802522	55	PUSH EBP
7C802523	8BEC	MOV EBP,ESP
7C802525	6A 00	PUSH 0
7C802527	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
7C80252A	FF75 08	PUSH DWORD PTR SS:[EBP+8]
7C80252D	E8 0E000000	CALL kernel32.WaitForSingleObjectEx
7C802532	5D	POP EBP
7C802533	C2 0800	RETN 8

Here is the correct MOV EDI, EDI to RETN 8. Too simple:).

CopyMemII + Debugblocker

Target : **Game Editor 1.3.2**

Homepage: <http://game-editor.com>

Crack Tool : **1.OllyDBG**

2005

2. LordPE

Deluxe 1.4-by

yoda

3.Import

REConstructor

1.6 Final

Author : **Why Not Bar**

With **Game Editor 1.3.2** this looks like any gambling problems must unpack! Uncle children because they have written tut guide. With this Target only **CopyMemII + Debugblocker** no **Nano** but the difficulty here lalam how to earn the IAT is OK Full stop! With how common it is difficult to succeed. Therefore we must use as a dose of the network to search for the Full IAT. Uncle The practice will only understand ...

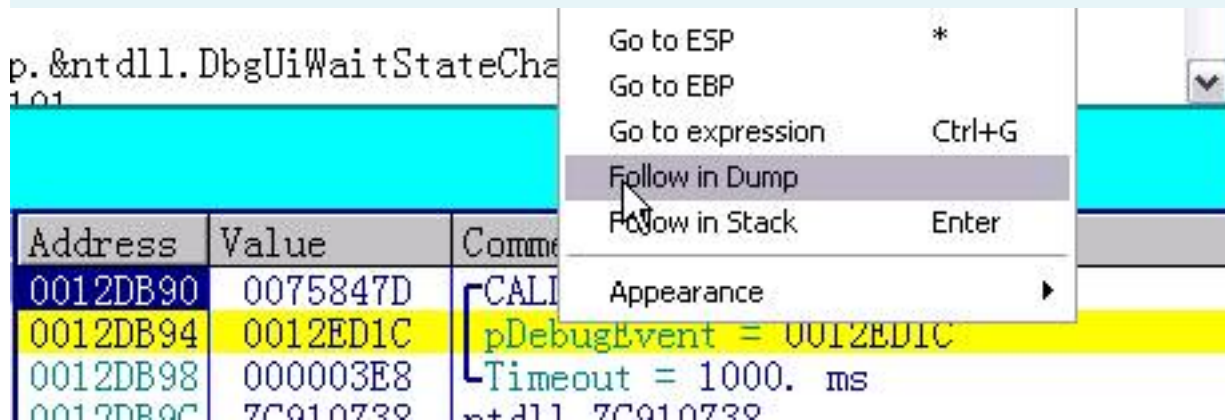
_ Target Load:

Address	Hex	Decomp	Disassembly
007688A3	\$ 55		push ebp
007688A4	. 8BEC		mov ebp, esp
007688A6	. 6A FF		push -1
007688A8	. 68 88217900		push gameEdit.00792188
007688AD	. 68 E0857600		push gameEdit.007685E0
007688B2	. 64:A1 00000000		mov eax, dword ptr fs:[0]
007688B8	. 50		push eax

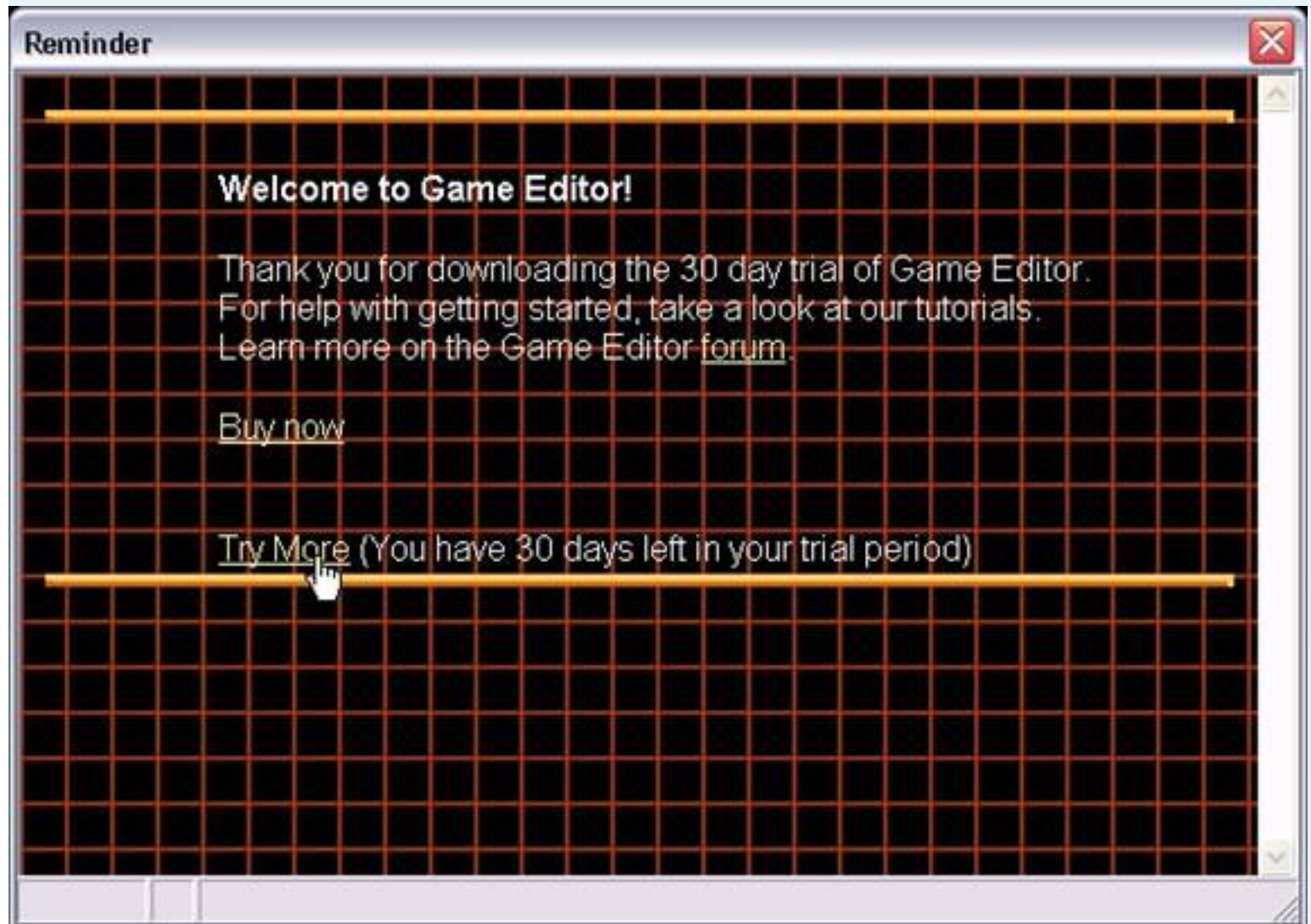
_Alt + F1, BP WaitForDebugEvent and press F9 Stack window we see as follows:

0012DB90	0075847D	CALL to WaitForDebugEvent
0012DB94	0012ED1C	pDebugEvent = 0012ED1C
0012DB98	000003E8	Timeout = 1000. ms
0012DB9C	7C910738	ntdll.7C910738
0012DBA0	00000000	

_Chon Like



Next BC WaitForDebugEvent, BP WriteProcessMemory, press F9, 1 Nag prompt appears



_Nhan "More Try" and you will see the following:

Address	Value	Comment	Address	Value	Comment
0012ED30	00000000		0012D9FC	0075C947	CALL to WriteProcessMemory from gameE
0012ED34	005D6105	gameEdit.005D6105	0012DA00	00000048	hProcess = 00000048 (window)
0012ED38	00000002		0012DA04	005D6000	Address = 5D6000
0012ED3C	00000000		0012DA08	0036D008	Buffer = 0036D008
0012ED40	005D6105	gameEdit.005D6105	0012DA0C	00001000	BytesToWrite = 1000 (4096.)
0012ED44	005D6105	gameEdit.005D6105	0012DA10	0012DB40	pBytesWritten = 0012DB40
0012ED48	00000001		0012DA14	00000008	

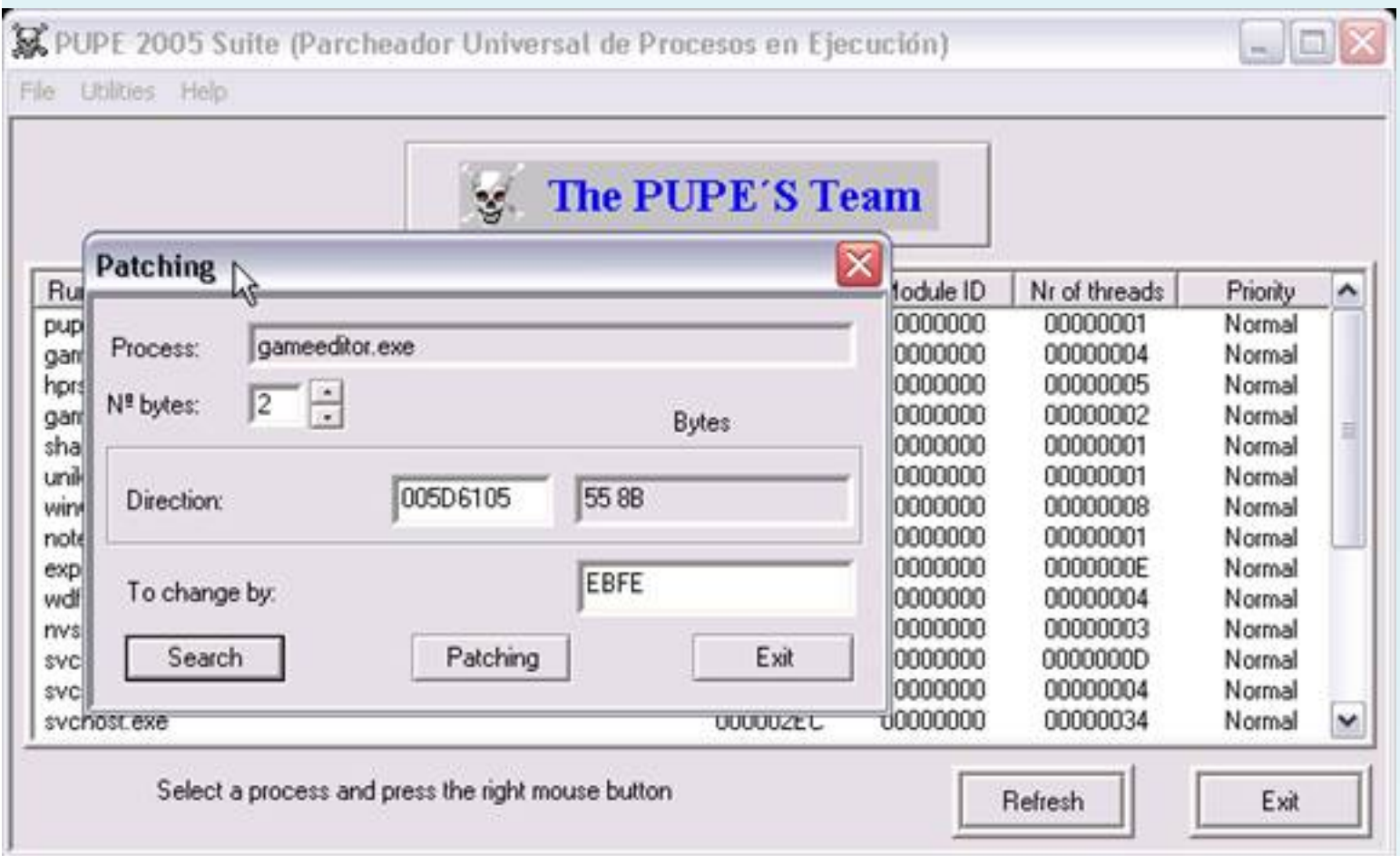
_ Ctrl + F9, in the Stack you roll the mouse to meet the second return from the return first and we are

0012DB40	00001000	
0012DB44	0036E008	
0012DB48	0012DB88	
0012DB4C	0075B3F2	RETURN to gameEdit.0075B3F2 from gameEdit.0075B778
0012DB50	000001D5	
0012DB54	0036B744	
0012DB58	00000000	

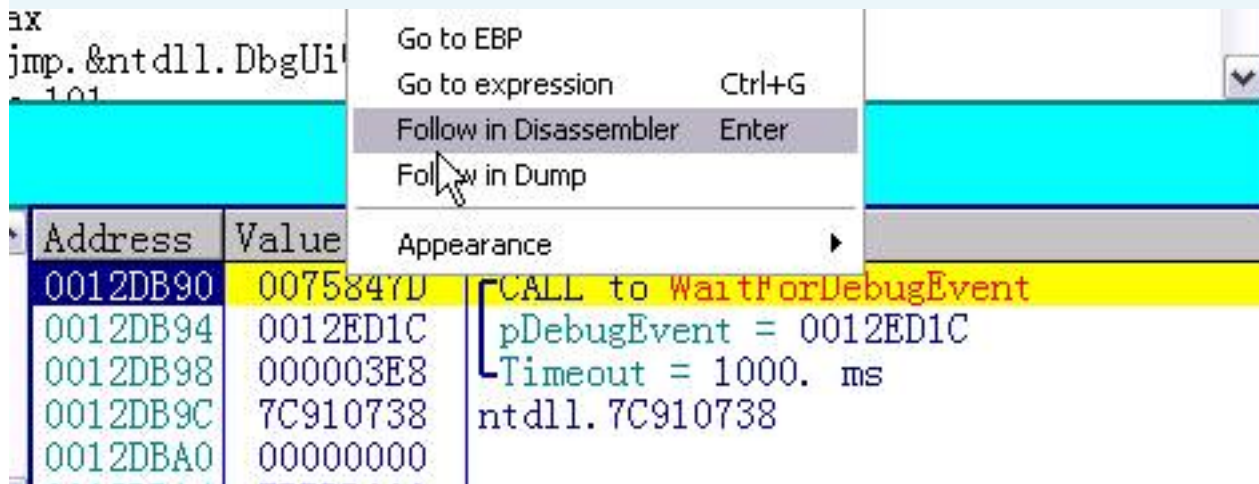
_ Back to the CPU, Ctrl + G: 0075B778, Ctrl + R, select the command Call 2 Double-click on it and submit lenh Call this as follows:

0075B6FF	. 50	push eax	
0075B700	90	nop	
0075B701	90	nop	
0075B702	90	nop	
0075B703	90	nop	
0075B704	90	nop	
0075B705	. 83C4 0C	add esp, 0C	
0075B708	. 50	push eax	
0075B709	. F7D0	not eax	

_ Open PuPe and enter as follows:



_ Click Patching, you have a breakpoint in the kernel32 **WriteProcessMemory**. Bc it, then set a breakpoint bp **WaitForDebugEvent**, F9. At 0012DB90 [in Stack] -> right click> **Follow in Disassembler**.



_ Need to set an origin here:

0075847D	. 85C0	test eax, eax	
0075847F	0F84 102A000	Backup	
00758485	8B85 FCFDFFF	Copy	p-204]
0075848B	25 FF000000	Binary	
00758490	85C0	Assemble	Space
00758492	74 13	Label	A7
00758494	8B0D B435790	Comment	35B4]
0075849A	8379 20 00	Breakpoint], 0
0075849E	74 07	Hit trace	A7
007584A0	C685 FCFDFFF	Run trace], 0
007584A7	> 68 A8347900		
007584AC	. FF15 A4B1780	New origin here	Ctrl+Gray * NEL32.EnterCriti
007584B2	60	Go to	

_ And patch as follows:

00758468	A0	db A0	
00758469	90	nop	
0075846A	90	nop	
0075846B	90	nop	
0075846C	90	nop	
0075846D	90	nop	
0075846E	90	nop	
0075846F	90	nop	
00758470	90	nop	
00758471	90	nop	
00758472	90	nop	
00758473	90	nop	
00758474	90	nop	
00758475	90	nop	
00758476	90	nop	
00758477	90	nop	
00758478	90	nop	
00758479	90	nop	
0075847A	90	nop	
0075847B	90	nop	
0075847C	90	nop	
0075847D	. 85C0	test eax, eax	
0075847F	- E9 7C8BCAFF	jmp gameEdit.00401000	
00758484	90	nop	
00758485	. 8B85 FCFDFFF	mov eax, dword ptr ss:[ebp-204]	
0075848B	25 FF000000	and eax, 0FF	

_ Jump to 401,000 and the patch:

00401000	8105 34ED1200 00100000	add dword ptr ds:[12ED34], 1000	
0040100A	8105 40ED1200 00100000	add dword ptr ds:[12ED40], 1000	
00401014	8105 44ED1200 00100000	add dword ptr ds:[12ED44], 1000	
0040101E	813D 34ED1200 00A05F00	cmp dword ptr ds:[12ED34], gameEdit.005FA000	
00401028	- 0F85 56743500	jnz gameEdit.00758484	
0040102E	90	nop	
0040102F	90	nop	
00401030	0000	add byte ptr ds:[eax], al	

_ Continue in 3 patch addresses containing **OEP**:

0012ED30	00000000	
0012ED34	00400000	gameEdit.00400000
0012ED38	00000002	
0012ED3C	00000000	
0012ED40	00400000	gameEdit.00400000
0012ED44	00400000	gameEdit.00400000

_ Set 1 breakpoint in **0040102E** Press F9 to run

00401000	8105	34ED1200	00100000	add dword ptr ds:[12ED34],1000
0040100A	8105	40ED1200	00100000	add dword ptr ds:[12ED40],1000
00401014	8105	44ED1200	00100000	add dword ptr ds:[12ED44],1000
0040101E	813D	34ED1200	00A05F00	cmp dword ptr ds:[12ED34],gameEdit.005FA000
00401028	-0F85			t.00758484
0040102E	90			
0040102F	90			
00401030	0000			r ds:[eax],al
00401032	0000			r ds:[eax],al
00401034	0000			r ds:[eax],al
00401036	0000			r ds:[eax],al
00401038	0000			r ds:[eax],al
0040103A	0000			
0040103C	0000			
0040103E	0000			

Backup
 Copy
 Binary
 Undo selection Alt+BkSp
 Assemble Space
 Label :
 Comment ;
 Breakpoint
 Run trace

Toggle F2
 Conditional Shift+F2
 Conditional log Shift+F4

_ Nhan F9 to run

Command bp WaitForDebugEvent

Breakpoint at gameEdit.0040102E

_ Now we pass by **debug blocker**

00401000	8105 34ED1200 00100000	add dword ptr ds:[12ED34],1000
0040100A	8105 40ED1200 00100000	add dword ptr ds:[12ED40],1000
00401014	8105 44ED1200 00100000	add dword ptr ds:[12ED44],1000
0040101E	813D 34ED1200 00A05F00	cmp dword ptr ds:[12ED34],gameEdit.005FA000
00401028	- 0F85 56743500	jnz gameEdit.00758484
0040102E	68 C0070000	push 7C0
00401033	E8 5993457C	call kernel32.DebugActiveProcessStop
00401038	90	nop
00401039	90	nop
0040103A	0000	add byte ptr ds:[eax],al
0040103C	0000	add byte ptr ds:[eax],al

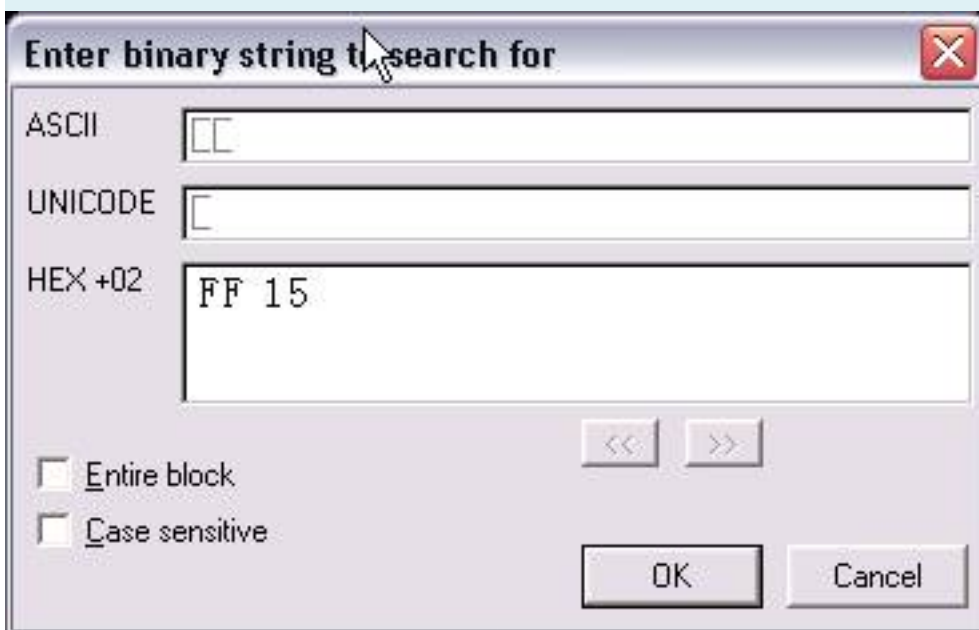
_ Press F8 when past nop after the function **Call debugActiveProcessStop**

Address	Hex dump	Disassembly	Com	Registers (FP)
00401000	8105 34ED1200 00100000	add dword ptr ds:[12ED34],1000		EAX 00000001
0040100A	8105 40ED1200 00100000	add dword ptr ds:[12ED40],1000		ECX 00129C5C
00401014	8105 44ED1200 00100000	add dword ptr ds:[12ED44],1000		EDX 7C90EB94
0040101E	813D 34ED1200 00A05F00	cmp dword ptr ds:[12ED34],gameEdit.005FA000		EBX 7FFDB000
00401028	- 0F85 56743500	jnz gameEdit.00758484		ESP 00129C7C
0040102E	68 C0070000	push 7C0		EBP 0012F73C
00401033	E8 5993457C	call kernel32.DebugActiveProcessStop		ESI 00001D5F
00401038	90	nop		EDI 0000000F
00401039	90	nop		EBI 00401039

_Nhin See **EAX = 00000001** is please send! Open a OllyDBG other, Attach. F9, F12 and patch as follows:

005D6105	55	push ebp
005D6106	8BEC	mov ebp, esp
005D6108	6A FF	push -1
005D610A	68 00076200	push gameEdit.00620700
005D610F	68 307C5D00	push gameEdit.005D7C30
005D6114	64:A1 00000000	mov eax, dword ptr fs:[0]

_ Now we determine the start and IAT IAT End. Scroll mouse on top of the screen, press Ctrl + B and type FF15:



1 _Ok you to play here

0040D726	C605 2C966800 00	mov byte ptr ds:[68962C],0	
0040D72D	FF15 E4A15F00	call dword ptr ds:[5FA1E4]	kernel32.GetWindowsDirectoryA
0040D733	85C0	test eax, eax	
0040D735	0F84 D9000000	je gameEdit.0040D814	
0040D73B	BF 2C986800	mov edi, gameEdit.0068982C	

_Các You should remember this API function to little more we find IAT full easily. Remember completed as selected pictures:

0040D726	C605 2C966800 00	mov byte p	Breakpoint	
0040D72D	FF15 E4A15F00	call dword	Run trace	
0040D733	85C0	test eax, e	Follow	Enter
0040D735	0F84 D9000000	je gameE	New origin here	Ctrl+Gray *
0040D73B	BF 2C986800	mov edi, ga	Go to	
0040D740	83C0 FF	cmov reg, reg	Thread	
ds:[005FA1E4]=7C82293B (kernel32.GetWin			Follow in Dump	
Address	Value	Comment	Search for	Selection
				Memory address

_Trong Window dump Scroll to find IAT start:

Address	Value	Comment
005FA000	77F18DD7	GDI32.CreatePalette
005FA004	77F16A3B	GDI32.DeleteObject
005FA008	77F19FC5	GDI32.GetDIBits
005FA00C	77F16E51	GDI32.CreateCompatibleBitmap
005FA010	77F19610	GDI32.CreateDIBSection
005FA014	77F16CA6	GDI32.DeleteDC
005FA018	77F16DC0	GDI32.DeleteDC

_Cuon To find IAT end:

Address	Value	Comment
005FA3C0	71AB4FD4	ws2_32.gethostname
005FA3C4	00EB6F18	
005FA3C8	763B311E	comdlg32.GetOpenFileNameA
005FA3CC	763C7CD8	comdlg32.GetSaveFileNameA
005FA3D0	00EB6F2B	
005FA3D4	00000000	
005FA3D8	00000000	

_ The next step we find IAT full and paste into the IAT is only OK. **ArmaDetach 1.1** Open and Drag "**gameEditor.exe**" released into the window to see the program as follows:



_ Open 1 Olly again and select the **process ID** and **Child Attach**. F9, F12, to patch **558B**

007688A3	55	push ebp
007688A4	8BEC	mov ebp, esp
007688A6	6A FF	push -1
007688A8	68 88217900	push gameEdit.00792188
007688AD	68 E0857600	push gameEdit.007685E0

_ Use OllyScript 0.92 Script run "**Magic Jump Finder Script.txt**." Running the 1 to the NAG Reminder, click "**More try**" to run more. But if you wait for it to run more to ensure that tomorrow morning. But it does the as follows:

Address	Hex dump	Disassembly	Comments
7C80B529	8BFF	mov edi,edi	
7C80B52B	55	push ebp	
7C80B52C	8BEC	mov ebp,esp	
7C80B52E	837D 08 00	cmp dword ptr ss:[ebp+8],0	

_nen chance network press F12 to 1. Ha ha



I finished _Ok Patch Magic Jump as usual:

00ED5B81	75 16	jnz short 00ED5B99	
00ED5B83	8D85 B4FEFFFF	lea eax,dword ptr ss:[ebp-14C]	
00ED5B89	50	push eax	
00ED5B8A	FF15 B862EF00	call dword ptr ds:[EF62B8]	kernel32.LoadLibraryA
00ED5B90	8B0D 6C50F000	mov ecx,dword ptr ds:[F0506C]	
00ED5B96	89040E	mov dword ptr ds:[esi+ecx],eax	
00ED5B99	A1 6C50F000	mov eax,dword ptr ds:[F0506C]	
00ED5B9E	391C06	cmp dword ptr ds:[esi+eax],ebx	
00ED5BA1	E9 30010000	jmp 00ED5CD6	
00ED5BA6	90	nop	
00ED5BA7	33C9	xor ecx,ecx	
00ED5BA9	8B07	mov eax,dword ptr ds:[edi]	

_Alt + M, and select the image:

00400000	00001000	gameEdit	PK header	Image	R	RWE
00401000	001F9000	gameEdit	text	Image	D	RWE
005FA000	0003D000	gameEdit	Actualize			RWE
00637000	000D8000	gameEdit	Dump in CPU			RWE
0070F000	0001C000	gameEdit	Dump			RWE
0072B000	00050000	gameEdit	Search	Ctrl+B		RWE
0077B000	00010000	gameEdit	Search next	Ctrl+L		RWE
0079B000	00010000	gameEdit	Set break-on-access	F2		RWE
007AB000	00150000	gameEdit				RWE
008FB000	00058000	gameEdit	Set memory breakpoint on access			RWE
00960000	00103000		Set memory breakpoint on write			R
00A70000	00099000					R E

_Shift + F9, hic hic, Pause it here:

Address	Hex dump	Disassembly	Comment
00EEBB45	AF	scas dword ptr es:[edi]	
00EEBB46	C4B5 99261F9B	les esi, fword ptr ss:[ebp+9B1F2699]	Modification of segment r
00EEBB4C	2B82 B4AF0247	sub eax, dword ptr ds:[edx+4702AFB4]	
00EEBB52	DE7F 6E	fdivr word ptr ds:[edi+6E]	
00EEBB55	C8 D84A7C	enter 4AD8, 7C	
00EEBB59	E3 ED	jecxz short 00EEBB48	

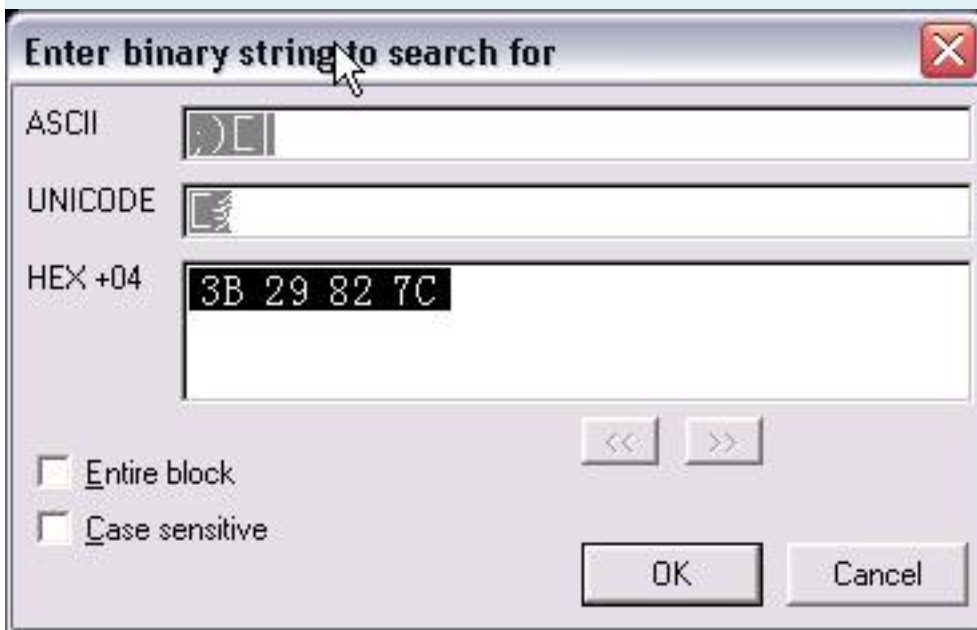
2 _Lieu network of press Shift + F9 and 1 more

Address	Hex dump	Disassembly	Comment
00EEBB45	AF	scas dword ptr es:[edi]	
00EEBB46	C4B5 99261F9B	les esi, fword ptr ss:[ebp+9B1F2699]	Modification of segment r
00EEBB4C	2B82 B4AF0247	sub eax, dword ptr ds:[edx+4702AFB4]	
00EEBB52	DE7F 6E	fdivr word ptr ds:[edi+6E]	
00EEBB55	C8 D84A7C	enter 4AD8, 7C	
00EEBB59	E3 ED	jecxz short 00EEBB48	

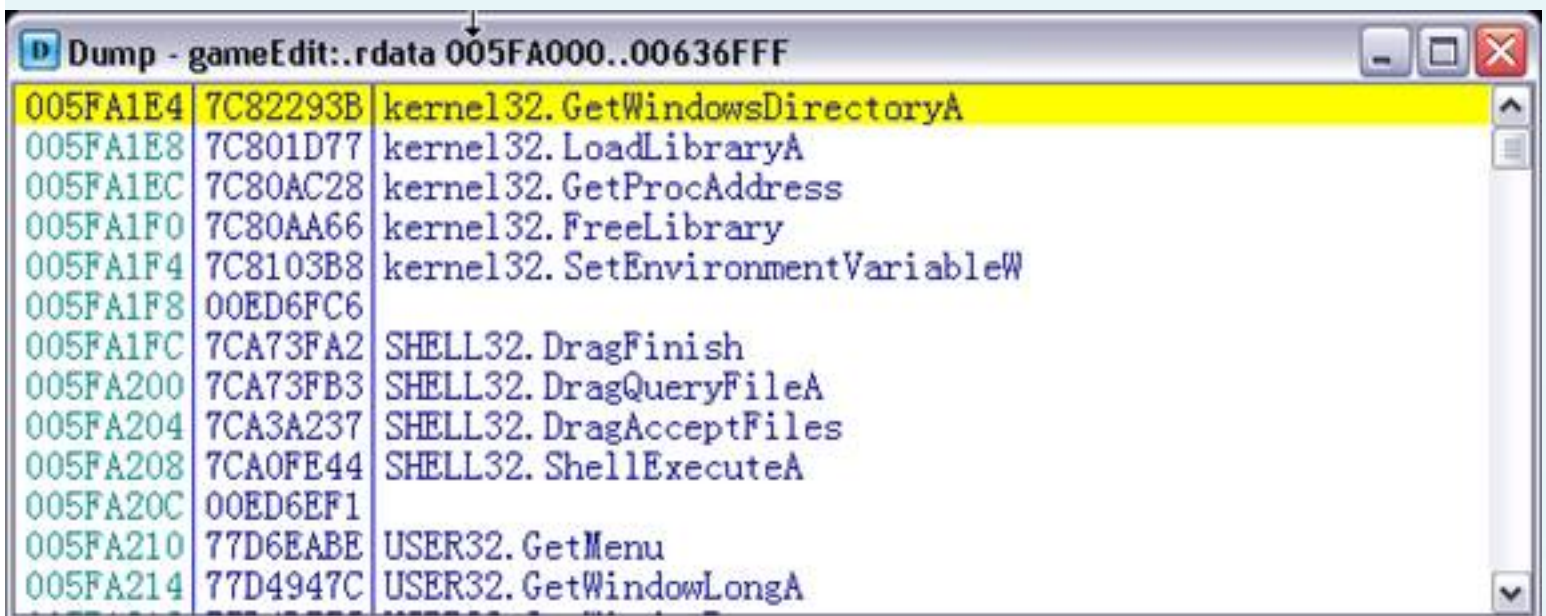
_Dung Hoàng. Grandparent shelves it, because we only need to find the IAT full stop to the OEP
Crash or have anything important first, you should not. You also remember Ham API "**kernel32.
GetWindowsDirectoryA**" they remind you remember when this? OK, Alt + F1 and enter the
following:

Command	? kernel32.GetWindowsDirecto	HEX: 7C82293B - DEC: 2088905019 - ASCII: ,);
Process terminated, exit code C0000005 (-1073741819.)		

_Alt + M, Ctrl + B, reverse the number again as follows:



_ OK to you here:



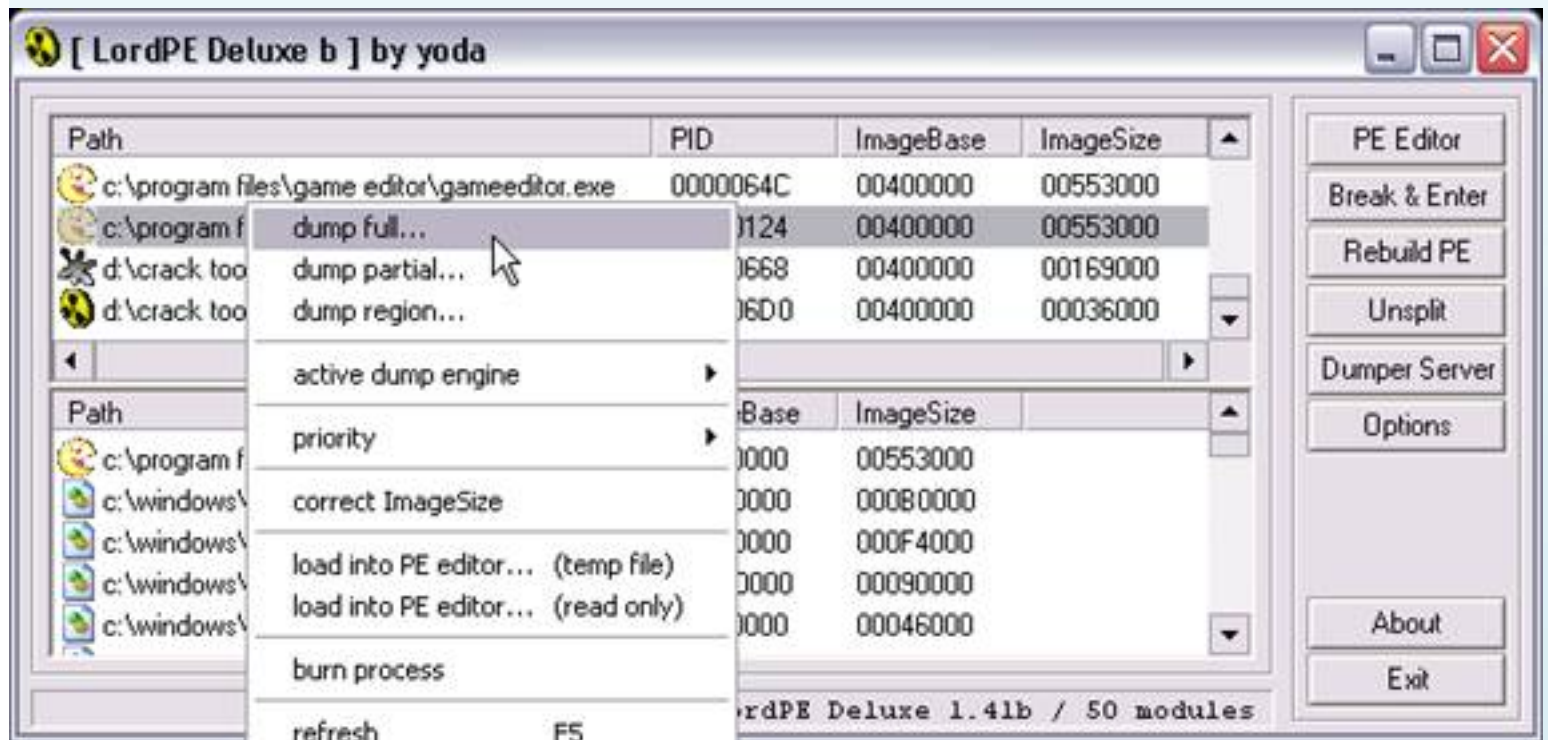
_ **Alt + C**, in the window dump press **Ctrl + G** and enter **005FA1E4**

Address	Value	Comment
005FA1DC	7C809B77	kernel32.CloseHandle
005FA1E0	7C81486A	kernel32.GetEnvironmentVariableA
005FA1E4	7C82293B	kernel32.GetWindowsDirectoryA
005FA1E8	7C801D77	kernel32.LoadLibraryA
005FA1EC	7C80AC28	kernel32.GetProcAddress
005FA1F0	7C80AA66	kernel32.FreeLibrary
005FA1F4	7C8103B8	kernel32.SetEnvironmentVariableW

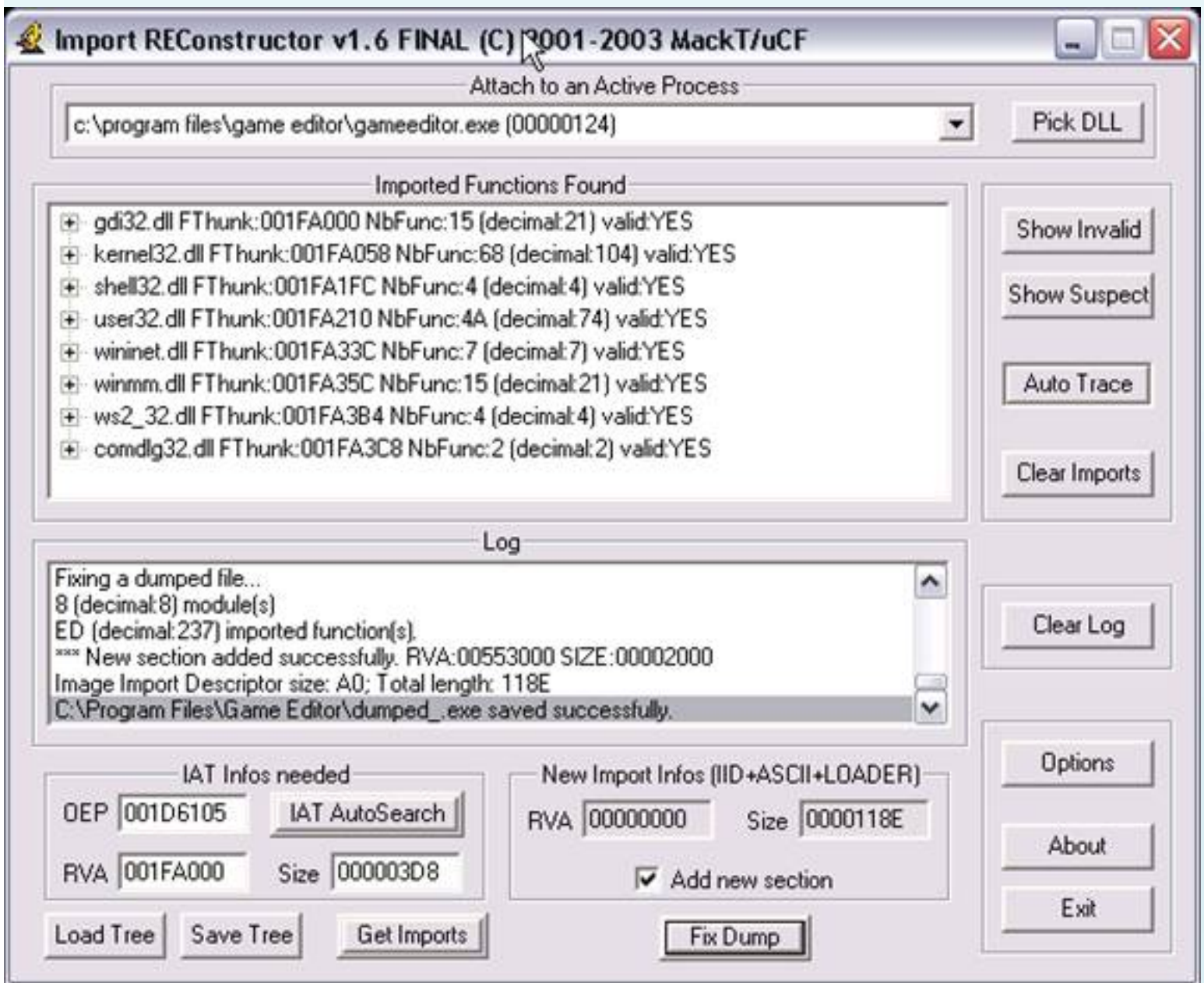
_ Of the IAT from start to IAT End then click **Binany \ Binany Copy** and **Paste** on the IAT is not yet fully complete. After Paste you will find the No. 1 function:

Address	Value	Comment
005FA070	7C80C6E0	kernel32.lstrlenA
005FA074	7C80C729	kernel32.lstrcpyA
005FA078	7C801A24	kernel32.CreateFileA
005FA07C	7C80B529	kernel32.GetModuleHandleA
005FA080	7C80B357	kernel32.GetModuleFileNameA
005FA084	7C80CC67	kernel32.SetThreadPriority
005FA088	7C800010	kernel32.SetCurrentThread...

_Nhu So we are plunging the right direction and has been FIX IAT. Now, dump Full stop! Use LordPE. Remember to choose the correct PID home!



_ Open ImportREC fill and dump Fix



_ Run the file to try "dumped_.exe". Ohh, lickish Run, no more NAG Reminder



This _Toi as unpack Done! Crack and the franchise for you. Target is also very easy Crack! File cracked themselves with the tut



Bye! I look back!

Written by Why Not Bar

Armadillo collect sand-stone

GetRight - 5.0 Final <|> ARM 2.xx-3.xx - Debug Blocker + CopyMem



1. Intro:

_Hi All, the process unpack this soft nothing difficult, just the place to make the target, Hung huc buffaloes as I was a new error target this midstream disk in 1000 near my CD!

2. Tools:

- 1.OllyDBG - The best config debugger for ArmMUP by hacnho.
- 2.LordPE 1.4 Deluxe
- 3.Import REConstructor 1.6 Final

3. Unpacking

_Truoc All we need to identify signs of Debug Blocker:

Image Name	PID	Description	User Name	CPU	Mem Usage
winamp.exe	3224	Winamp	VIP\Super Administrator	1	3,300 K
getright.exe	2940	GetRight@ www.getright.com	VIP\Super Administrator	0	9,432 K
getright.exe	2932	GetRight@ www.getright.com	VIP\Super Administrator	0	1,816 K
UniKey.exe	2920		VIP\Super Administrator	0	3,508 K
TSCHelp.exe	2900	TechSmith HTML Help Helper	VIP\Super Administrator	0	3,024 K
SnagIt32.exe	2892	SnagIt Screen Capture for Win...	VIP\Super Administrator	2	6,536 K
WINWORD.EXE	2836	Microsoft Office Word	VIP\Super Administrator	0	35,016 K
emeditor.exe	2732	EmEditor	VIP\Super Administrator	0	2,900 K
TurboLaunch.exe	1908	TurboLaunch	VIP\Super Administrator	0	6,172 K

_Sau Detect signs of CopyMEM. First, load up OllyDBG target:

Address	Hex dump	Disassembly	Comment
005F80B9	55	PUSH EBP	
005F80BA	8BEC	MOV EBP,ESP	
005F80BC	6A FF	PUSH -1	
005F80BE	68 68026100	PUSH getright.00610268	
005F80C3	68 007B5F00	PUSH getright.005F7B00	
005F80C8	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	SE handler installation
005F80CE	50	PUSH EAX	
005F80CF	64:8925 000000	MOV DWORD PTR FS:[0],ESP	
005F80D6	83EC 58	SUB ESP,58	
005F80D9	53	PUSH EBX	
005F80DA	56	PUSH ESI	
005F80DB	57	PUSH EDI	
005F80DC	8B65 E8	MOV DWORD PTR SS:[EBP-18],ESP	
005F80DF	FF15 28D16000	CALL DWORD PTR DS:[<&KERNEL32.GetVersion	kernel32.GetVersion
005F80E5	3302	XOR EDI,EDI	

_Dat Breakpoint WriteProcessMemory, Shift + F9:

Address	Value	Comment
0012E038	005F6A29	CALL to WriteProcessMemory from
0012E03C	0000004C	hProcess = 0000004C (window)
0012E040	005F80B9	Address = 5F80B9
0012E044	0012E31C	Buffer = 0012E31C
0012E048	00000002	BytesToWrite = 2
0012E04C	0012E320	pBytesWritten = 0012E320
0012E050	77F79005	ntdll.77F79005
0012E054	77FB1BB0	ntdll.77FB1BB0
0012E058	5F5F5F5F	

_Lan 2:

Address	Value	Comment
0012E038	005F6A46	CALL to WriteProcessMemory from
0012E03C	0000004C	hProcess = 0000004C (window)
0012E040	005F80B9	Address = 5F80B9
0012E044	00610A94	Buffer = getright.00610A94
0012E048	00000002	BytesToWrite = 2
0012E04C	0012E320	pBytesWritten = 0012E320
0012E050	005F6AC2	RETURN to getright.005F6AC2 from
0012E054	00000050	
0012E058	0012E05C	

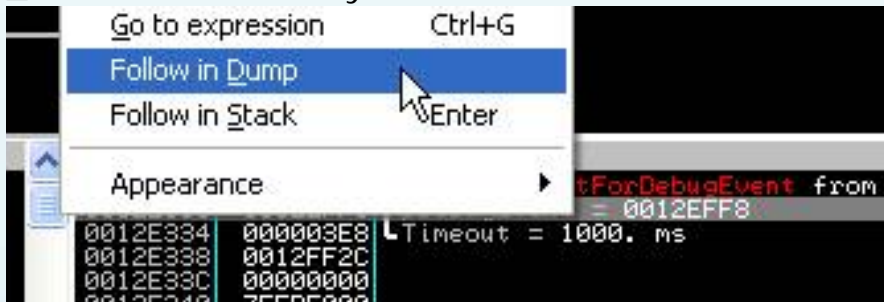
_Lan 3:

Address	Value	Comment
0012E108	005F6404	CALL to WriteProcessMemory from
0012E10C	0000004C	hProcess = 0000004C (window)
0012E1E0	00534000	Address = 534000
0012E1E4	00398F50	Buffer = 00398F50
0012E1E8	00001000	BytesToWrite = 1000 (4096.)
0012E1EC	0012E2EC	pBytesWritten = 0012E2EC
0012E1F0	00001339	
0012E1F4	000403EC	
0012E1F8	0012E574	

_Yeap, J CopyMEM. We need to determine OEP! Set breakpoint at API WaitForDebugEvent! Bp WaitForDebugEvent, F9:

Address	Value	Comment
0012E32C	005F4510	CALL to WaitForDebugEvent from
0012E330	0012EFF8	pDebugEvent = 0012EFF8
0012E334	000003E8	Timeout = 1000. ms
0012E338	0012FF2C	
0012E33C	00000000	
0012E340	7FFDF000	
0012E344	0012E3B8	
0012E348	C0150008	
0012E34C	00000000	

_Ghi 12EFF8 memory. F2 restart Ctrl, Alt + F1: BP WaitForDebugEvent, Follow in dump:



_Xoa Breakpoint WaitForDebugEvent, set breakpoint WriteProcessMemory, F9! We have OEP = 534E90!

Address	Value	Comment
0012F008	00000000	
0012F00C	00000000	
0012F010	00534E90	getright.00534E90
0012F014	00000002	
0012F018	00000000	
0012F01C	00534E90	getright.00534E90
0012F020	00534E90	getright.00534E90
0012F024	00000000	
0012F028	51585001	

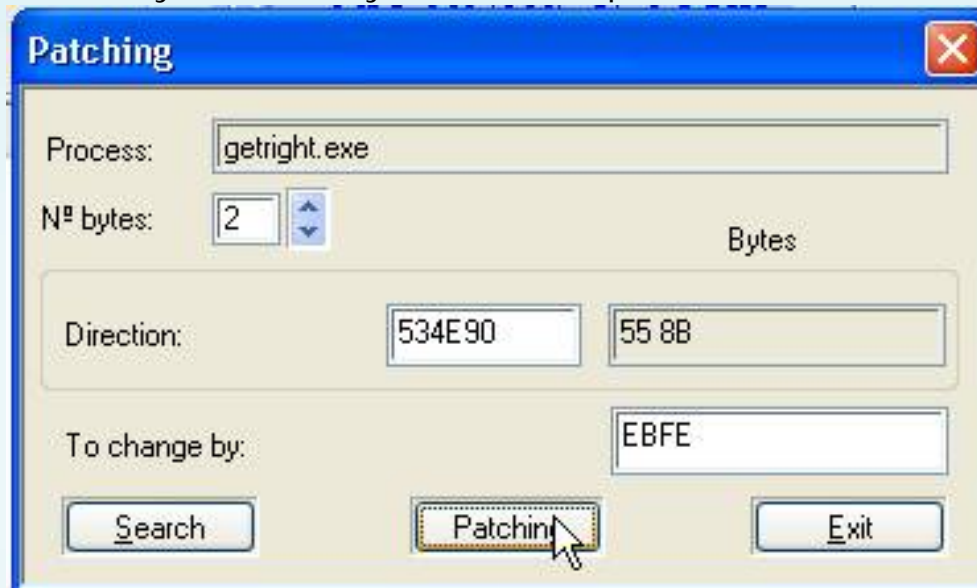
_Chung We need to find MagicCall, but first we need to patch EP's Child Process again, note the stack:

Address	Value	Comment
0012E1D8	005F6404	CALL to WriteProcessMemory from
0012E1DC	0000004C	hProcess = 0000004C (window)
0012E1E0	00534000	Address = 534000
0012E1E4	00398F50	Buffer = 00398F50
0012E1E8	00001000	BytesToWrite = 1000 (4096.)
0012E1EC	0012E2EC	pBytesWritten = 0012E2EC
0012E1F0	00001339	
0012E1F4	00000001	
0012E1F8	00534000	getright.00534000

_Ta Information need attention are:

_OEP = 534E90

_O Here you can use your hands to patch, but I khoái used for pro PUPE J .



_OK Have time to find MagicCall, Alt + K Stack to see why:

Address	Stack	Procedure / arguments	Called from	Frame
0012E1D8	005F64A4	? kernel32.WriteProcessMemory	getright.005F649E	
0012E1DC	0000004C	hProcess = 0000004C (window)		
0012E1E0	00534000	Address = 534000		
0012E1E4	00398F50	Buffer = 00398F50		
0012E1E8	00001000	BytesToWrite = 1000 (4096.)		
0012E1EC	0012E2EC	pBytesWritten = 0012E2EC		
0012E2F8	005F58D6	? getright.005F5A4C	getright.005F58D1	
0012E2FC	00000133	Arg1 = 00000133		
0012E300	00396E6C	Arg2 = 00396E6C		
0012E304	00000000	Arg3 = 00000000		
0012E328	005F4836	getright.005F57A8	getright.005F4831	0012E324
0012E32C	00000133	Arg1 = 00000133		
0012E330	00396E6C	Arg2 = 00396E6C		
0012E334	00000000	Arg3 = 00000000		
0012F5C0	005F1E42	getright.005F36E0	getright.005F1E3D	0012F5BC
0012FD20	005F25C9	getright.005F1897	getright.005F25C4	0012FD1C
0012FF38	005F8187	getright.005F2340	getright.<ModuleEntryPoint>	0012FF34
0012FF3C	00400000	Arg1 = 00400000		
0012FF40	00000000	Arg2 = 00000000		
0012FF44	00151F08	Arg3 = 00151F08		
0012FF48	0000000A	Arg4 = 0000000A		

_Double Click here:

Address	Hex dump	Disassembly	Comment
005F58D1	. E8 76010000	CALL getright.005F5A4C	getright.005F5A4C
005F58D6	. 83C4 0C	ADD ESP,0C	
005F58D9	. 25 FF000000	AND EAX,0FF	
005F58DE	. 85C0	TEST EAX,EAX	
005F58E0	. 75 07	JNZ SHORT getright.005F58E9	
005F58E2	. 32C0	XOR AL,AL	

_Enter Call to function:

Address	Hex dump	Disassembly	Comment
005F5A4C	. 55	PUSH EBP	
005F5A4D	. 8BEC	MOV EBP,ESP	
005F5A4F	. 81EC 00010000	SUB ESP,100	
005F5A55	. 56	PUSH ESI	
005F5A56	. 8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
005F5A59	. C1E0 0C	SHL EAX,0C	
005F5A5C	. 8B0D BC0A6100	MOV ECX,DWORD PTR DS:[610ABC]	getright.00401000
005F5A62	. 03C8	ADD ECX,EAX	
005F5A64	. 8B4D EC	MOV DWORD PTR SS:[EBP-14],ECX	
005F5A67	. 8B15 D80A6100	MOV EDX,DWORD PTR DS:[610AD8]	
005F5A6D	. 8B55 FC	MOV DWORD PTR SS:[EBP-4],EDX	
005F5A70	. 91 D80A6100	MOV EAX,DWORD PTR DS:[610AD8]	

_Ctrl + R:

Address	Disassembly	Comment
005F58D1	CALL getright.005F5A4C	
005F5A24	CALL getright.005F5A4C	
005F5A4C	PUSH EBP	(Initial CPU selection)

_Ta See magic call is:

References in getright. Text1 to 005F5A4C, Item 1

Address = 005F5A24

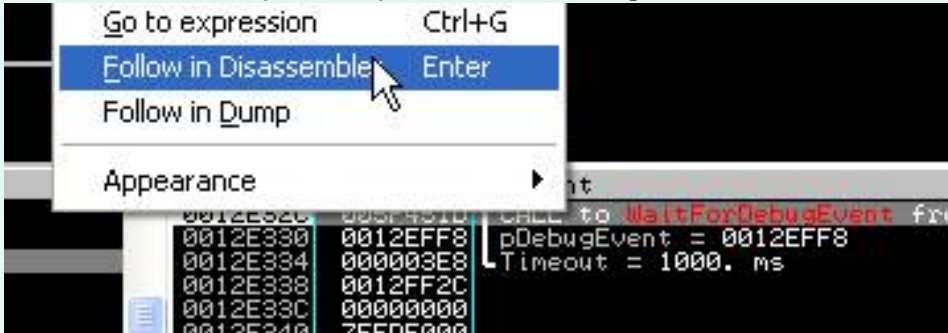
Disassembly = CALL getright.005F5A4C

_Vi Stars? View tut 4! You need to submit it again:

Address	Hex dump	Disassembly	Comment
005F5A24	. E8 23000000	CALL getright.005F5A4C	getright.005F5A4C
005F5A29	. 83C4 0C	ADD ESP,0C	
005F5A2C	. 8B0D C80A6100	MOV ECX,DWORD PTR DS:[610AC8]	
005F5A32	. 8B15 CC0A6100	MOV EDX,DWORD PTR DS:[610ACC]	
005F5A38	. C7048A FFFFFFFF	MOV DWORD PTR DS:[EDX+ECX*4],-1	
005F5A3F	. ^E9 F4FEFFFF	JMP getright.005F5938	
005F5A44	. > B0 01	MOV AL,1	
005F5A46	. > FF	CALL ESI	

Address	Hex dump	Disassembly	Comment
005F5A24	90	NOP	getright.005F5A4C
005F5A25	90	NOP	
005F5A26	90	NOP	
005F5A27	90	NOP	
005F5A28	90	NOP	
005F5A29	83C4 0C	ADD ESP,0C	
005F5A2C	8B00 C80A6100	MOV ECX,DWORD PTR DS:[610AC8]	
005F5A32	8B15 CC0A6100	MOV EDX,DWORD PTR DS:[610ACC]	
005F5A38	C7048A FFFFFFFF	MOV DWORD PTR DS:[EDX+ECX*4],-1	
005F5A3F	E9 F4FEFFFF	JMP getright.005F5938	

_Dat The breakpoint bp WaitForDebugEvent! F9:



_Ta To here:

Address	Hex dump	Disassembly	Comment
005F451D	85C0	TEST EAX,EAX	
005F451F	0F84 33120000	JE getright.005F5758	
005F4525	33D2	XOR EDX,EDX	
005F4527	8A15 A40A6100	MOV DL,BYTE PTR DS:[610AA4]	
005F452D	85D2	TEST EDI,EDI	
005F452F	75 6D	JNZ SHORT getright.005F459E	
005F4531	8B85 9CFAFFFF	MOV EAX,DWORD PTR SS:[EBP-564]	
005F4537	25 FF000000	AND EAX,0FF	
005F453C	85C0	TEST EAX,EAX	
005F453E	74 5E	JE SHORT getright.005F459E	
005F4540	C685 9CFAFFFF	MOV BYTE PTR SS:[EBP-564],0	
005F4547	C745 FC 010000	MOV DWORD PTR SS:[EBP-4],1	
005F454F	66:9C	PUSHFQ	

_Set New origin in here: Ctrl + *:

Address	Hex dump	Disassembly	Comment
005F451D	85C0	TEST EAX,EAX	
005F451F	0F84 33120000	JE getright.005F5758	
005F4525	33D2	XOR EDX,EDX	
005F4527	8A15 A40A6100	MOV DL,BYTE PTR DS:[610AA4]	
005F452D	85D2	TEST EDI,EDI	
005F452F	75 6D	JNZ SHORT getright.005F459E	
005F4531	8B85 9CFAFFFF	MOV EAX,DWORD PTR SS:[EBP-564]	
005F4537	25 FF000000	AND EAX,0FF	
005F453C	85C0	TEST EAX,EAX	
005F453E	74 5E	JE SHORT getright.005F459E	
005F4540	C685 9CFAFFFF	MOV BYTE PTR SS:[EBP-564],0	
005F4547	C745 FC 010000	MOV DWORD PTR SS:[EBP-4],1	
005F454E	66:9C	PUSHFQ	
005F4550	66:58	POPFQ	
005F4552	66:05 6200	ADD AX,6200	
005F4554	66:05 9E00	ADD AX,9E00	
005F4556	66:50	PUSH AX	
005F4558	66:9D	POPFQ	
005F455A	66:05 0100	ADD AX,0100	
005F455C	C745 FC 000000	MOV DWORD PTR SS:[EBP-4],0	
005F455E	EB 33	JMP SHORT getright.005F459E	
005F4560	8B4D EC	MOV ECX,DWORD PTR DS:[610AEC]	
005F4562	8B11	MOV EDI,DWORD PTR DS:[610AEC]	
005F4564	8B02	MOV ESI,DWORD PTR DS:[610AEC]	

_Truoc The patch:

Address	Hex dump	Disassembly	Comment
005F4503	. 85C0	TEST EAX,EAX	
005F4505	..0F84 52120000	JE getright.005F575D	
005F4508	. 68 E8030000	PUSH 3E8	Timeout = 1000. ms pDebugEvent WaitForDebugEvent
005F4510	. 8B8D 38FAFFFF	MOV ECK,DWORD PTR SS:[EBP-5C8]	
005F4516	. 51	PUSH ECK	
005F4517	. FF15 8CD06000	JMP 00000000	
005F451D	. 85C0	TEST EAX,EAX	
005F451F	..0F84 33120000	JE getright.005F5758	
005F4525	. 33D2	XOR EDX,EDX	
005F4527	. 8A15 A40A6100	MOV DL,BYTE PTR DS:[610AA4]	
005F452D	. 85D2	TEST EDX,EDX	

_Sau The patch:

Address	Hex dump	Disassembly	Comment
005F4503	. 85C0	TEST EAX,EAX	
005F4505	..0F84 52120000	JE getright.005F575D	
005F4508	90	NOP	Timeout
005F450C	90	NOP	
005F450D	90	NOP	
005F450E	90	NOP	
005F450F	90	NOP	
005F4510	90	NOP	
005F4511	90	NOP	
005F4512	90	NOP	
005F4513	90	NOP	
005F4514	90	NOP	
005F4515	90	NOP	pDebugEvent WaitForDebugEvent
005F4516	90	NOP	
005F4517	90	NOP	
005F4518	90	NOP	
005F4519	90	NOP	
005F451A	90	NOP	
005F451B	90	NOP	
005F451C	90	NOP	
005F451D	. 85C0	TEST EAX,EAX	
005F451F	-E9 DCCAE0FF	JMP getright.00401000	
005F4524	90	NOP	
005F4525	. 33D2	XOR EDX,EDX	
005F4527	. 8A15 A40A6100	MOV DL,BYTE PTR DS:[610AA4]	
005F452D	. 85D2	TEST EDX,EDX	
005F452F	..75 6D	JNZ SHORT getright.005F459E	

G + _Ctrl 401,000 to patch. Before the patch:

Address	Hex dump	Disassembly	Comment
00401000	0000	ADD BYTE PTR DS:[EAX],AL	
00401002	0000	ADD BYTE PTR DS:[EAX],AL	
00401004	0000	ADD BYTE PTR DS:[EAX],AL	
00401006	0000	ADD BYTE PTR DS:[EAX],AL	
00401008	0000	ADD BYTE PTR DS:[EAX],AL	
0040100A	0000	ADD BYTE PTR DS:[EAX],AL	
0040100C	0000	ADD BYTE PTR DS:[EAX],AL	
0040100E	0000	ADD BYTE PTR DS:[EAX],AL	
00401010	0000	ADD BYTE PTR DS:[EAX],AL	
00401012	0000	ADD BYTE PTR DS:[EAX],AL	

_Sau The patch:

Address	Hex dump	Disassembly	Comment
00401000	8105 10F01200 01	ADD DWORD PTR DS:[12F010],1000	
0040100A	8105 1CF01200 01	ADD DWORD PTR DS:[12F01C],1000	
00401014	8105 20F01200 01	ADD DWORD PTR DS:[12F020],1000	
0040101E	8130 20F01200 01	CMP DWORD PTR DS:[12F020],getright.00586000	
00401028	-0F85 F6341F00	JNZ getright.005F4524	
0040102E	90	NOP	
0040102F	90	NOP	
00401030	0000	ADD BYTE PTR DS:[EAX],AL	

_Tai Stars have the exact number 12F010 stack, and the 586,000:

Address	Value	Comment
0012F008	00000000	
0012F00C	00000000	
0012F010	00534E90	getright.00534E90
0012F014	00000002	
0012F018	00000000	
0012F01C	00534E90	getright.00534E90
0012F020	00534E90	getright.00534E90
0012F024	00000000	
0012F028	51585001	

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
003C0000	0000E000				Map	RW	RW	
003D0000	00001000				Priv	RW	RW	
00400000	00001000	getright		PE header	Imag	R	RWE	
00401000	00185000	getright	.text		Imag	R	RWE	
00586000	00048000	getright	.rdata	exports	Imag	R	RWE	
005D1000	00018000	getright	.data	data	Imag	R	RWE	
005E9000	00004000	getright	.idata		Imag	R	RWE	
005ED000	00020000	getright	.text1	code	Imag	R	RWE	
0060D000	00020000	getright	.data1	imports	Imag	R	RWE	
0062D000	00100000	getright	.pdata		Imag	R	RWE	
0072D000	000CE000	getright	.rsrc	resources	Imag	R	RWE	

_Patch The stack:

Address	Value	Comment
0012F008	00000000	
0012F00C	00000000	
0012F010	00400000	getright.00400000
0012F014	00000002	
0012F018	00000000	
0012F01C	00400000	getright.00400000
0012F020	00400000	getright.00400000
0012F024	00000000	
0012F028	51525001	

_Dat A breakpoint at 40102E, F9:

Breakpoint at getright.0040102E

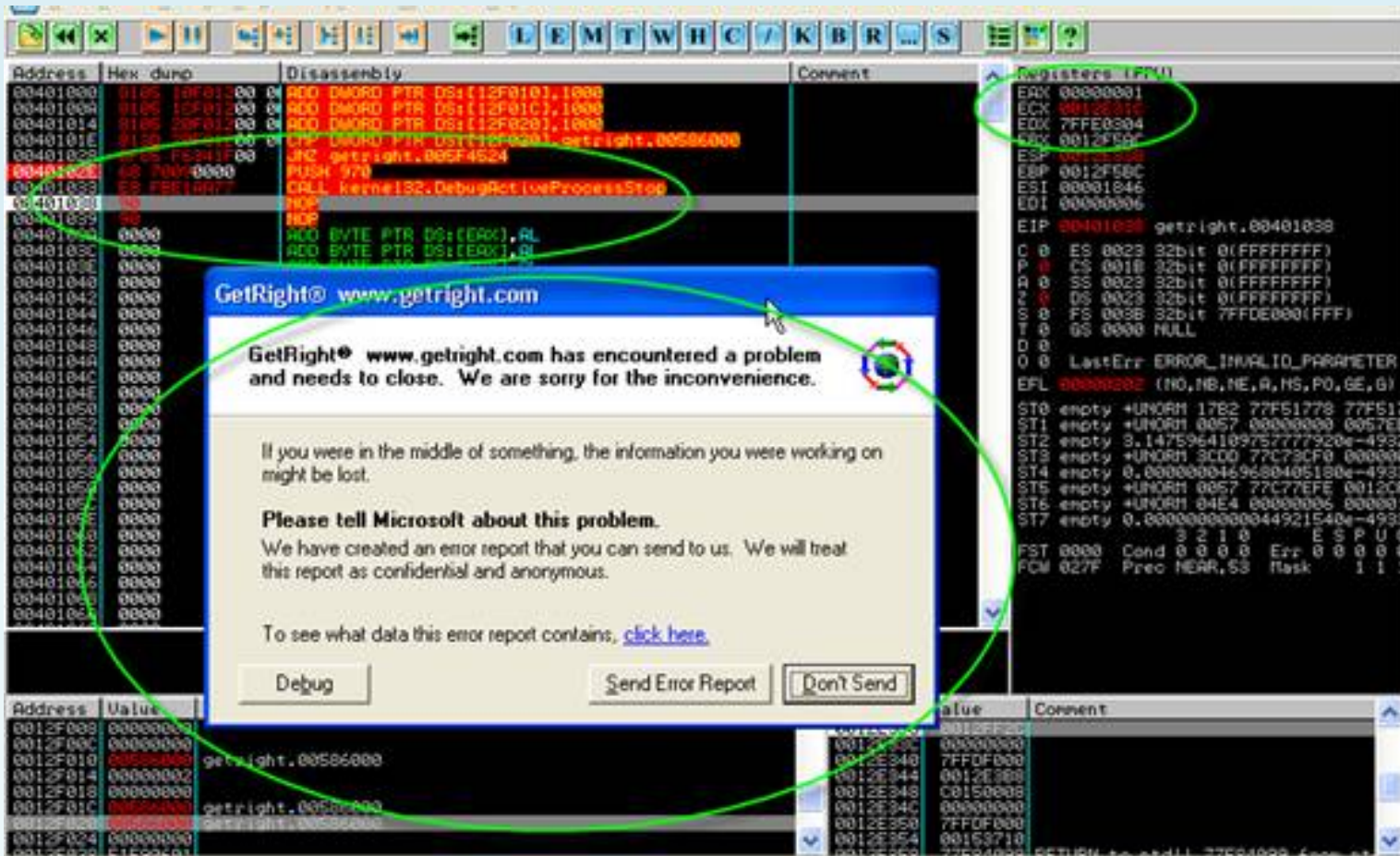
_Toi Here:

Address	Hex dump	Disassembly	Comment
00401000	8105 10F01200 01	ADD DWORD PTR DS:[12F010],1000	
0040100A	8105 1CF01200 01	ADD DWORD PTR DS:[12F01C],1000	
00401014	8105 20F01200 01	ADD DWORD PTR DS:[12F020],1000	
0040101E	8130 20F01200 01	CMP DWORD PTR DS:[12F020],getright.00586000	
00401028	-0F85 F6341F00	JNZ getright.005F4524	
0040102E	9B	NOP	
0040102F	9B	NOP	
00401030	0000	ADD BYTE PTR DS:[EAX],AL	

_Chung We continue to remove the debug blocker!

000006E8	wdfmgr		C:\WINDOWS\System32\wdfmgr.exe
000008B4	TurboLaunch	TurboLaunch	C:\Program Files\TurboLaunch\TurboLaunch.exe
00000964	getright		C:\Program Files\GetRight\getright.exe
00000970	getright	43916AFA	C:\Program Files\GetRight\getright.exe
00000A8C	ieexplore	SysFader	C:\Program Files\Internet Explorer\ieexplore.
00000B94	WINWORD	CiceroUIWndFrame	C:\Program Files\Microsoft Office\OFFICE11\W

_Tiep Continue to patch:



_Done! Open a OllyDBG other, Attach!

Address	Hex dump	Disassembly	Comment
77F767CE	C3	RETN	
77F767CF	CC	INT3	
77F767D0	C3	RETN	
77F767D1	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]	
77F767D5	CC	INT3	
77F767D6	C2 0400	RETN 4	
77F767D9	64:A1 18000000	MOV EAX,WORD PTR FS:[18]	
77F767DF	C3	RETN	
77F767E0	57	PUSH EDI	

_F9, F12:

Address	Hex dump	Disassembly	Comment
00534E90	-EB FE	JMP SHORT getright.00534E90	
00534E92	EC	IN AL,DX	I/O command
00534E93	6A FF	PUSH -1	
00534E95	68 18CF5900	PUSH getright.0059CF18	
00534E9A	68 9C8C5300	PUSH getright.00538C9C	
00534E9F	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00534EA5	50	PUSH EAX	
00534EA6	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00534EAD	83C4 A8	ADD ESP,-58	
00534EB0	53	PUSH EBX	
00534EB1	56	PUSH ESI	
00534EB2	57	PUSH EDI	
00534EB3	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00534EB6	FF15 949D5E00	CALL DWORD PTR DS:[5E9D94]	kernel32.GetVersion
00534FBC	33D2	XOR EDX,EDX	

_Patch To 558B:

Address	Hex dump	Disassembly	Comment
00534E90	55	PUSH EBP	
00534E91	8BEC	MOV EBP,ESP	
00534E93	6A FF	PUSH -1	
00534E95	68 18CF5900	PUSH getright.0059CF18	
00534E9A	68 9C8C5300	PUSH getright.00538C9C	
00534E9F	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00534EA5	50	PUSH EAX	
00534EA6	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00534EAD	83C4 A8	ADD ESP,-58	
00534EB0	53	PUSH EBX	
00534EB1	56	PUSH ESI	
00534EB2	57	PUSH EDI	
00534EB3	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00534EB6	FF15 949D5E00	CALL DWORD PTR DS:[5E9D94]	kernel32.GetVersion
00534EBC	33D2	XOR EDX,EDX	
00534EBF	8AD4	MOV DL,AH	

Full _Dump:

Path	PID	ImageBase	ImageSize
c:\program files\turbolaunch\turbolaunch.exe	000008B4	00400000	001A0000
i:\cracker\debug-disassembler\odbg110_org...	0000095C	00400000	00164000
c:\program files\getright\getright.exe	00000964	00400000	003FB000
c:\...	00000970	00400000	003FB000
c:\...	00000A8C	00400000	00019000
c:\...	00000B94	30000000	00BA0000
c:\...	00000BF4	00400000	00286000
c:\...	00000BFC	00400000	0000A000
d:\...	00000CBC	00400000	0002F000

_Sau Dump when finished, we started the process of finding and Fix IAT dump! At: 00534EB6 FF15 949D5E00 CALL DWORD PTR DS: [5E9D94]; kernel32.GetVersion

_Ban Follow in dump, Memory Address, we will have IAT Info as follows:

_IAT Start:

Address	Value	Comment
005E99E0	00000000	
005E99E4	00000000	
005E99E8	00000000	
005E99EC	77DD31FD	ADVAPI32.AdjustTokenPrivileges
005E99F0	77DD17D8	ADVAPI32.RegCloseKey
005E99F4	77DE6A68	ADVAPI32.RegEnumValueA
005E99F8	77DD229A	ADVAPI32.RegOpenKeyExA
005E99FC	77DE68E2	ADVAPI32.RegDeleteKeyA
005E9A00	77DD3410	ADVAPI32.RegOpenKeyExA

_IAT End:

Address	Value	Comment
005EA2B8	74D3F0F3	ole32.OleUIBusyA
005EA2BC	00000000	
005EA2C0	00000000	
005EA2C4	00000000	
005EA2C8	00000000	
005EA2CC	49570000	
005EA2D0	2E4D4D4E	
005EA2D4	006C6C64	getright.006C6C64

_IAT Len = 8CC! OK, Ctrl + F2 Restart all!

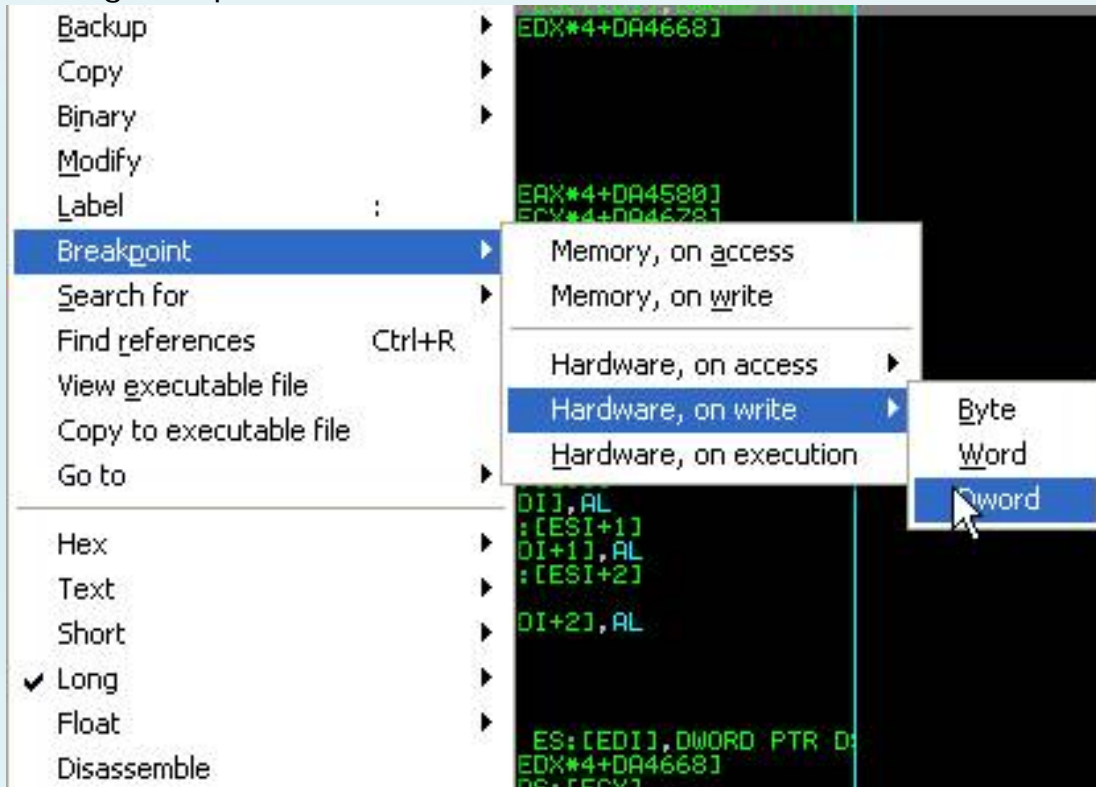
_Detach Father:

Address	Hex dump	Disassembly	Cor
005F451D	50	PUSH EAX	
005F451E	E8 10AD8B77	CALL kernel32.DebugActiveProcessStop	
005F4523	90	NOP	
005F4524	0033	ADD BYTE PTR DS:[EBX],DH	
005F4526	? D28A 15A40A61	ROR BYTE PTR DS:[EDX+610AA415],CL	
005F452C	? 0085 0275608B	ADD BYTE PTR SS:[EBP+8B607502],AL	

_Attach Child:

Address	Hex dump	Disassembly	Comment
005F80B9	55	PUSH EBP	
005F80BA	80EC	MOV EBP,ESP	
005F80BC	6A FF	PUSH -1	
005F80BE	68 68026100	PUSH getright.00610268	
005F80C3	68 007B5F00	PUSH getright.005F7B00	SE handler installation
005F80C8	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
005F80CE	50	PUSH EAX	
005F80CF	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
005F80D6	83EC 58	SUB ESP,58	
005F80D9	53	PUSH EBX	
005F80DA	56	PUSH ESI	
005F80DB	57	PUSH EDI	
005F80DC	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
005F80DF	FF15 28D16000	CALL DWORD PTR DS:[<&KERNEL32.GetVersion	kernel32.GetVersion
005F80E5	33D2	XOR EDX,EDX	
005F80E7	8004	MOV DL,AH	

_Trong Dump window, Ctrl + G: 005E99EC:



_Shift + F9 2 times:

Address	Hex dump	Disassembly	Comment
00DA2278	8B85 74FCFFFF	MOV EAX,DWORD PTR SS:[EBP-38C]	getright.005E99EC
00DA227E	83C0 04	ADD EAX,4	
00DA2281	8985 74FCFFFF	MOV DWORD PTR SS:[EBP-38C],EAX	
00DA2287	E9 78FEFFFF	JMP 00DA2104	
00DA228C	0FB685 80FCFFFF	MOVZX EAX,BYTE PTR SS:[EBP-380]	
00DA2293	85C0	TEST EAX,EAX	
00DA2295	74 7F	JE SHORT 00DA2316	
00DA2297	6A 00	PUSH 0	
00DA2299	8B85 84FCFFFF	MOV EAX,DWORD PTR SS:[EBP-37C]	

_Cuon Upturn for signs of MagicJump:

Address	Hex dump	Disassembly	Comment
00DA21FC	6A 00	PUSH 0	
00DA21FE	FFB5 9CFEFFFF	PUSH DWORD PTR SS:[EBP-164]	
00DA2204	E8 F72B0000	CALL 00DA4E00	
00DA2209	59	POP ECX	
00DA220A	59	POP ECX	
00DA220B	40	INC EAX	
00DA220C	8985 9CFEFFFF	MOV DWORD PTR SS:[EBP-164],EAX	
00DA2212	FFB5 64FCFFFF	PUSH DWORD PTR SS:[EBP-39C]	
00DA2218	FFB5 88FCFFFF	PUSH DWORD PTR SS:[EBP-378]	
00DA221E	E8 3529FFFF	CALL 00D94B58	
00DA2223	8985 6CFCFFFF	MOV DWORD PTR SS:[EBP-394],EAX	
00DA2229	83B0 6CFCFFFF	CMP DWORD PTR SS:[EBP-394],0	
00DA2230	75 38	JNZ SHORT 00DA226A	
00DA2232	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
00DA2235	8B00	MOV EAX,DWORD PTR DS:[EAX]	
00DA2237	C700 03000000	MOV DWORD PTR DS:[EAX],3	
00DA223D	FF15 C4B0DA00	CALL DWORD PTR DS:[DAB0C4]	ntdll.RtlGetLastWin32Error
00DA2243	50	PUSH EAX	
00DA2244	FFB5 64FCFFFF	PUSH DWORD PTR SS:[EBP-39C]	
00DA224A	FFB5 70FCFFFF	PUSH DWORD PTR SS:[EBP-390]	
00DA2250	68 80E5DA00	PUSH 0DAE580	ASCII "File \"%s", function
00DA2255	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]	
00DA2258	FF70 04	PUSH DWORD PTR DS:[EAX+4]	
00DA225B	E8 5C2C0000	CALL 00DA4EB	
00DA2260	83C4 14	ADD ESP,14	
00DA2263	33C0	XOR EAX,EAX	
00DA2265	E9 DB040000	JMP 00DA2745	
00DA226A	8B85 74FCFFFF	MOV EAX,DWORD PTR SS:[EBP-38C]	
00DA2270	8B80 6CFCFFFF	MOV ECX,DWORD PTR SS:[EBP-394]	
00DA2276	8908	MOV DWORD PTR DS:[EAX],ECX	
00DA2278	8B85 74FCFFFF	MOV EAX,DWORD PTR SS:[EBP-38C]	getright.005E99EC
00DA227E	83C0 04	ADD EAX,4	
00DA2281	8985 74FCFFFF	MOV DWORD PTR SS:[EBP-38C],EAX	
00DA2287	E9 78FEFFFF	JMP 00DA2104	

F2 _Nhan set a breakpoint in this function Call!

Address	Hex dump	Disassembly	Comment
00D94B49	5E	POP ESI	
00D94B4A	5B	POP EBX	
00D94B4B	C9	LEAVE	
00D94B4C	C3	RETN	
00D94B4D	6A 78	PUSH 78	
00D94B4F	FF15 00B1DA00	CALL DWORD PTR DS:[DAB100]	ntdll.RtlSetLastWin32Error
00D94B55	33C0	XOR EAX,EAX	
00D94B57	C3	RETN	
00D94B58	55	PUSH EBP	
00D94B59	8BEC	MOV EBP,ESP	
00D94B5B	53	PUSH EBX	
00D94B5C	56	PUSH ESI	
00D94B5D	57	PUSH EDI	
00D94B5E	33FF	XOR EDI,EDI	
00D94B60	33DB	XOR EBX,EBX	
00D94B62	66:F745 0E FFFF	TEST WORD PTR SS:[EBP+E],0FFFF	
00D94B68	75 03	JNZ SHORT 00D94B6D	
00D94B6A	8B5D 0C	MOV EBX,DWORD PTR SS:[EBP+C]	
00D94B6D	57	PUSH EDI	
00D94B6E	FF15 A4B0DA00	CALL DWORD PTR DS:[DAB0A4]	kernel32.GetModuleHandleA
00D94B74	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
00D94B77	3BC8	CMP ECX,EAX	
00D94B79	75 07	JNZ SHORT 00D94B82	
00D94B7B	B8 18D3DA00	MOV EAX,0DA0318	
00D94B80	EB 30	JMP SHORT 00D94BB2	
00D94B82	393D 08D7DA00	CMP DWORD PTR DS:[DAD7D8],EDI	
00D94B88	B8 08D7DA00	MOV EAX,0DA07D8	
00D94B8D	74 0C	JE SHORT 00D94B9B	
00D94B8F	3B48 08	CMP ECX,DWORD PTR DS:[EAX+8]	
00D94B92	74 1B	JE SHORT 00D94BAF	
00D94B94	83C0 0C	ADD EAX,0C	
00D94B97	3938	CMP DWORD PTR DS:[EAX],EDI	
00D94B99	75 F4	JNZ SHORT 00D94B8F	
00D94B9B	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	

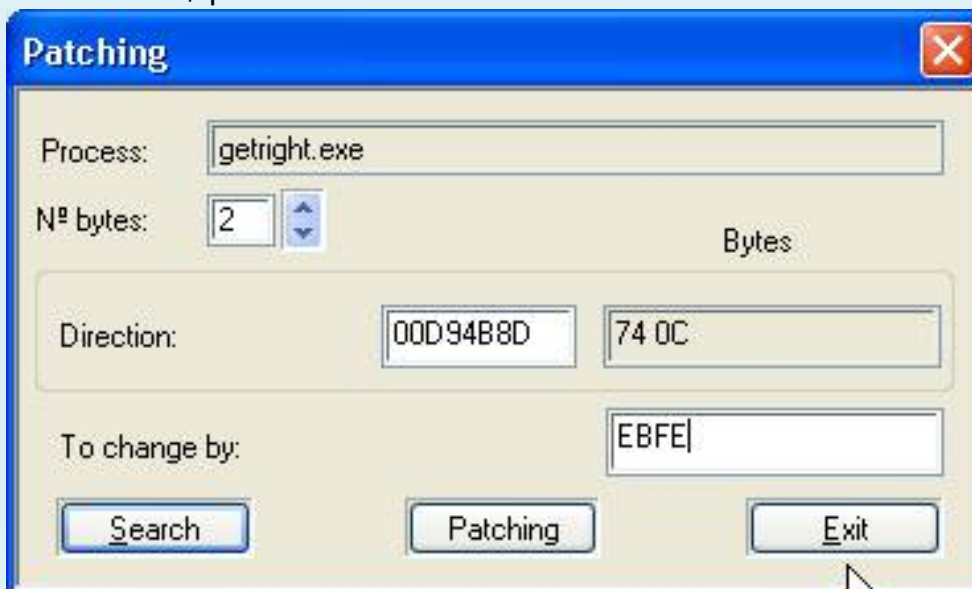
_Patch To:

Address	Hex dump	Disassembly	Comment
00D94B58	55	PUSH EBP	
00D94B59	8BEC	MOV EBP,ESP	
00D94B5B	53	PUSH EBX	
00D94B5C	56	PUSH ESI	
00D94B5D	57	PUSH EDI	
00D94B5E	33FF	XOR EDI,EDI	
00D94B60	33DB	XOR EBX,EBX	
00D94B62	66:F745 0E FFFF	TEST WORD PTR SS:[EBP+E],0FFFF	
00D94B68	75 03	JNZ SHORT 00D94B6D	
00D94B6A	8B5D 0C	MOV EBX,DWORD PTR SS:[EBP+C]	
00D94B6D	57	PUSH EDI	
00D94B6E	FF15 A4B0DA00	CALL DWORD PTR DS:[0AB0DA00]	kernel32.GetModuleHandleA
00D94B74	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
00D94B77	3BC8	CMP ECX,EAX	
00D94B79	75 07	JNZ SHORT 00D94B82	
00D94B7B	B8 18D3DA00	MOV EAX,0DAD318	
00D94B80	EB 30	JMP SHORT 00D94B82	
00D94B82	393D 08D7DA00	CMP DWORD PTR DS:[0AD7D8],EDI	
00D94B88	B8 08D7DA00	MOV EAX,0DAD7D8	
00D94B8D	EB 0C	JMP SHORT 00D94B9B	
00D94B8F	3B48 08	CMP ECX,DWORD PTR DS:[EBP+8]	
00D94B92	74 18	JE SHORT 00D94BAF	
00D94B94	83C0 0C	ADD EAX,0C	
00D94B97	3938	CMP DWORD PTR DS:[EAX],EDI	
00D94B99	75 F4	JNZ SHORT 00D94B8F	
00D94B9B	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
00D94B9E	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
00D94BA1	E8 41000000	CALL 00D94BE7	
00D94BA6	59	POP ECX	
00D94BA7	59	POP ECX	
00D94BA8	5F	POP EDI	
00D94BA9	5E	POP ESI	
00D94BAA	5B	POP EBX	
00D94BAB	5D	POP EBP	

_Delete All breakpoint, F9, the OEP has been encrypted!

Address	Hex dump	Disassembly	Comment
00534E90	44	INC ESP	
00534E91	B8 A504EE5B	MOV EAX,5BEE04A5	
00534E96	51	PUSH ECX	
00534E97	A1 483321F2	MOV EAX,DWORD PTR DS:[F2213348]	
00534E9C	9D	POPF	
00534E9D	60	PUSHAD	
00534E9E	49	DEC ECX	
00534E9F	0AB0 33496E11	OR DH,BYTE PTR DS:[EAX+116E4933]	
00534EA5	632D E7343349	ARPL WORD PTR DS:[493334E7],BP	
00534EAB	6E	OUTS DX,BYTE PTR ES:[EDI]	I/O command
00534EAC	11B0 8DC64265	ADC DWORD PTR DS:[EAX+6542C68D],ESI	
00534EB2	1E	PUSH DS	
00534EB3	E7 74	OUT 74,EAX	I/O command

F2 + _Ctrl restart all! After set breakpoint at start IAT, Shift + F9 2 times you Ctrl + G: 00D94B8D, patch or use the EBFPUPE:



_Nhan F2 in the call DA221E, F9, F7! Now at the magic jump is:

Address	Hex dump	Disassembly	Comment
00D94B79	✓75 07	JNZ SHORT 00D94B82	
00D94B7B	B8 18D3DA00	MOV EAX,0DAD318	
00D94B80	✓EB 30	JMP SHORT 00D94BB2	
00D94B82	393D D8D7DA00	CMP DWORD PTR DS:[DAD7D8],EDI	
00D94B88	B8 D8D7DA00	MOV EAX,0DAD7D8	
00D94B8D	✓EB FE	JMP SHORT 00D94B8D	
00D94B8F	3B48 08	CMP ECX,DWORD PTR DS:[EAX+8]	
00D94B92	✓74 1B	JE SHORT 00D94BAF	
00D94B94	83C0 0C	ADD EAX,0C	
00D94B97	3938	CMP DWORD PTR DS:[EAX],EDI	
00D94B99	^75 F4	JNZ SHORT 00D94B8F	
00D94B9B	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
00D94B9E	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
00D94BA1	E8 41000000	CALL 00D94BE7	
00D94BA6	59	POP ECX	
00D94BA7	59	POP ECX	
00D94BA8	5F	POP EDI	
00D94BA9	5E	POP ESI	
00D94BAA	5B	POP EBX	
00D94BAB	5D	POP EBP	
00D94BAC	C2 0800	RETN 8	
00D94BAF	8B40 04	MOV EAX,DWORD PTR DS:[EAX+4]	
00D94BB2	3BC7	CMP EAX,EDI	
00D94BB4	^74 E5	JE SHORT 00D94B9B	
00D94BB6	3978 08	CMP DWORD PTR DS:[EAX+8],EDI	
00D94BB9	8BF0	MOV ESI,EAX	
00D94BBB	^74 DE	JE SHORT 00D94B9B	
00D94BBD	66:3BDF	CMP BX,DI	
00D94BC0	✓74 06	JE SHORT 00D94BC8	
00D94BC2	66:3B5E 04	CMP BX,WORD PTR DS:[ESI+4]	
00D94BC6	✓EB 0E	JMP SHORT 00D94BD6	
00D94BC8	FF36	PUSH DWORD PTR DS:[ESI]	
00D94BCA	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
00D94BCD	E8 0E5D0100	CALL 00D9A8E0	

_Ta The change is EB0C:

Address	Hex dump	Disassembly	Comment
00D94B59	8BEC	MOV EBP,ESP	
00D94B5B	53	PUSH EBX	
00D94B5C	56	PUSH ESI	
00D94B5D	57	PUSH EDI	
00D94B5E	33FF	XOR EDI,EDI	
00D94B60	330B	XOR EBX,EBX	
00D94B62	66:F745 0E FFFF	TEST WORD PTR SS:[EBP+E],0FFFF	
00D94B68	✓75 03	JNZ SHORT 00D94B6D	
00D94B6A	8B5D 0C	MOV EBX,DWORD PTR SS:[EBP+C]	
00D94B6D	57	PUSH EDI	
00D94B6E	FF15 A4B0DA00	CALL DWORD PTR DS:[DAB0A4]	kernel32.GetModuleHandleA
00D94B74	8B40 08	MOV ECX,DWORD PTR SS:[EBP+8]	
00D94B77	3BC8	CMP ECX,EAX	
00D94B79	✓75 07	JNZ SHORT 00D94B82	
00D94B7B	B8 18D3DA00	MOV EAX,0DAD318	
00D94B80	✓EB 30	JMP SHORT 00D94BB2	
00D94B82	393D D8D7DA00	CMP DWORD PTR DS:[DAD7D8],EDI	
00D94B88	B8 D8D7DA00	MOV EAX,0DAD7D8	
00D94B8D	✓EB 0C	JMP SHORT 00D94B9B	
00D94B8F	3B48 08	CMP ECX,DWORD PTR DS:[EAX+8]	
00D94B92	✓74 1B	JE SHORT 00D94BAF	
00D94B94	83C0 0C	ADD EAX,0C	
00D94B97	3938	CMP DWORD PTR DS:[EAX],EDI	
00D94B99	^75 F4	JNZ SHORT 00D94B8F	
00D94B9B	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
00D94B9E	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
00D94BA1	E8 41000000	CALL 00D94BE7	
00D94BA6	59	POP ECX	
00D94BA7	59	POP ECX	
00D94BA8	5F	POP EDI	
00D94BA9	5E	POP ESI	
00D94BAA	5B	POP EBX	
00D94BAB	5D	POP EBP	
00D94BAC	C2 0800	RETN 8	

_Magic Patched Jump! Now open up ImpREC, enter the following information, Get Import, Show Invalid, Thanks Cuts, Fix dump:

Import REConstructor v1.6 FINAL (C) 2001-2003 MackT/uCF

Attach to an Active Process:

c:\program files\getright\getright.exe (00000D08) Pick DLL

Imported Functions Found

<input checked="" type="checkbox"/> ? FThunk:001E99EC NbFunc:11 (decimal:17) valid:NO
<input checked="" type="checkbox"/> ? FThunk:001E9A34 NbFunc:E (decimal:14) valid:NO
<input checked="" type="checkbox"/> gdi32.dll FThunk:001E9A70 NbFunc:41 (decimal:65) valid:YES
<input checked="" type="checkbox"/> ? FThunk:001E9B78 NbFunc:A0 (decimal:160) valid:NO
<input checked="" type="checkbox"/> ? FThunk:001E9DFC NbFunc:9 (decimal:9) valid:NO
<input checked="" type="checkbox"/> ? FThunk:001E9E24 NbFunc:E (decimal:14) valid:NO
<input checked="" type="checkbox"/> ? FThunk:001E9E60 NbFunc:D0 (decimal:208) valid:NO
<input checked="" type="checkbox"/> winmm.dll FThunk:001EA1A4 NbFunc:1 (decimal:1) valid:YES
<input checked="" type="checkbox"/> winspool.drv FThunk:001EA1AC NbFunc:3 (decimal:3) valid:YES

Show Invalid
Show Suspect
Auto Trace
Clear Imports

Log

rva:001E9DD4 forwarded from mod:ntdll.dll ord:0202 name:RtlAllocateHeap

Current imports:

4 (decimal:4) valid module(s) (added: +4 (decimal:+4))

218 (decimal:536) imported function(s). (added: +218 (decimal:+536))

70 (decimal:112) unresolved pointer(s) (added: +70 (decimal:+112))

Clear Log

IAT Infos needed

OEP IAT AutoSearch

RVA Size

Load Tree Save Tree Get Imports

New Import Infos (IID+ASCII+LOADER)

RVA Size

☒ Add new section

Fix Dump

Options
About
Exit

Attach to an Active Process

c:\program files\getright\getright.exe (00000D08) Pick DLL

Imported Functions Found

- ⊕ advapi32.dll FTThunk:001E99EC NbFunc:1 (decimal:1) valid:YES
- ⊕ gdi32.dll FTThunk:001E9A70 NbFunc:41 (decimal:65) valid:YES
- ⊕ kernel32.dll FTThunk:001E9B88 NbFunc:11 (decimal:17) valid:YES
- ⊕ kernel32.dll FTThunk:001E9BD0 NbFunc:19 (decimal:25) valid:YES
- ⊕ kernel32.dll FTThunk:001E9C38 NbFunc:8 (decimal:11) valid:YES
- ⊕ kernel32.dll FTThunk:001E9C78 NbFunc:8 (decimal:8) valid:YES
- ⊕ kernel32.dll FTThunk:001E9C9C NbFunc:24 (decimal:36) valid:YES
- ⊕ kernel32.dll FTThunk:001E9D30 NbFunc:2 (decimal:2) valid:YES
- ⊕ kernel32.dll FTThunk:001E9D44 NbFunc:5 (decimal:5) valid:YES

Show Invalid
Show Suspect
Auto Trace
Clear Imports

Log

Fixing a dumped file...
 10 (decimal:16) module(s)
 1A8 (decimal:424) imported function(s).
 *** New section added successfully. RVA:005B0000 SIZE:00003000
 Image Import Descriptor size: 140; Total length: 2524
 C:\Program Files\GetRight\dumped_.exe saved successfully.

Clear Log

IAT Infos needed

OEP IAT AutoSearch

RVA Size

Load Tree
Save Tree
Get Imports

New Import Infos (IID+ASCII+LOADER)

RVA Size

☒ Add new section

Fix Dump

Options
About
Exit

_Done: Run the test considered! : P!

GetRight® www.getright.com

GetRight® www.getright.com has encountered a problem and needs to close. We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.
 We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, [click here](#).

Ac, Load the file onto dumped.exe OllyDBG:

Address	Hex dump	Disassembly	Comment
00534E90	55	PUSH EBP	
00534E91	8BEC	MOV EBP,ESP	
00534E93	6A FF	PUSH -1	
00534E95	68 18CF5900	PUSH Dropped_.0059CF18	
00534E9A	68 9C8C5300	PUSH Dropped_.00538C9C	
00534E9F	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00534EA5	50	PUSH EAX	
00534EA6	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00534EAD	83C4 A8	ADD ESP,-58	
00534EB0	53	PUSH EBX	
00534EB1	56	PUSH ESI	
00534EB2	57	PUSH EDI	
00534EB3	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00534EB6	FF15 949D5E00	CALL DWORD PTR DS:[<&kernel32.GetVersion	kernel32.GetVersion
00534EBC	33D2	XOR EDX,EDX	
00534EBE	8AD4	MOV DL,AH	
00534EC0	8915 C0605E00	MOV DWORD PTR DS:[5E60C0],EDX	

_Ctrl + G:



_Toi And set a breakpoint:

Address	Hex dump	Disassembly	Comment
77E7D173	55	PUSH EBP	
77E7D174	8BEC	MOV EBP,ESP	
77E7D176	83EC 20	SUB ESP,20	
77E7D179	53	PUSH EBX	
77E7D17A	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
77E7D17D	8D45 E0	LEA EAX,DWORD PTR SS:[EBP-20]	
77E7D180	50	PUSH EAX	
77E7D181	FF15 8012E677	CALL DWORD PTR DS:[<&ntdll.RtlInitString	ntdll.RtlInitString
77E7D187	6A 01	PUSH 1	
77E7D189	8D45 E0	LEA EAX,DWORD PTR SS:[EBP-20]	
77E7D18C	50	PUSH EAX	
77E7D18D	8D45 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
77E7D190	50	PUSH EAX	
77E7D191	FF15 8012E677	CALL DWORD PTR DS:[<&ntdll.RtlInitString	ntdll.RtlInitString

_F9, Ctrl + F9, F7, and to this:

Address	Value	Comment
0012E594	005132A4	CALL to GetEnvironmentVariableA
0012E598	00D609EC	VarName = "DATELASTRUN"
0012E59C	00D60A0C	Buffer = 00D60A0C
0012E5A0	000000FF	BufSize = FF (255.)
0012E5A4	00000001	
0012E5A8	00D609EC	ASCII "DATELASTRUN"
0012E5AC	00D60A0C	
0012E5B0	0012FF04	Pointer to next SEH record

Address	Hex dump	Disassembly	Comment
005132A4	85C0	TEST EAX,EAX	
005132A6	6A FF	PUSH -1	
005132A8	8D4D F0	LEA ECX,DWORD PTR SS:[EBP-10]	
005132AB	75 21	JNZ SHORT Dropped_.005132CE	
005132AD	E8 DE830300	CALL Dropped_.0054B690	
005132B2	8065 FC 00	AND BYTE PTR SS:[EBP-4],0	
005132B6	8D4D EC	LEA ECX,DWORD PTR SS:[EBP-14]	
005132B9	E8 917F0300	CALL Dropped_.0054B24F	
005132BE	834D FC FF	OR DWORD PTR SS:[EBP-4],FFFFFFFF	

_Patch 7521 to EB21. Continue F9, Ctrl + F9, F7:

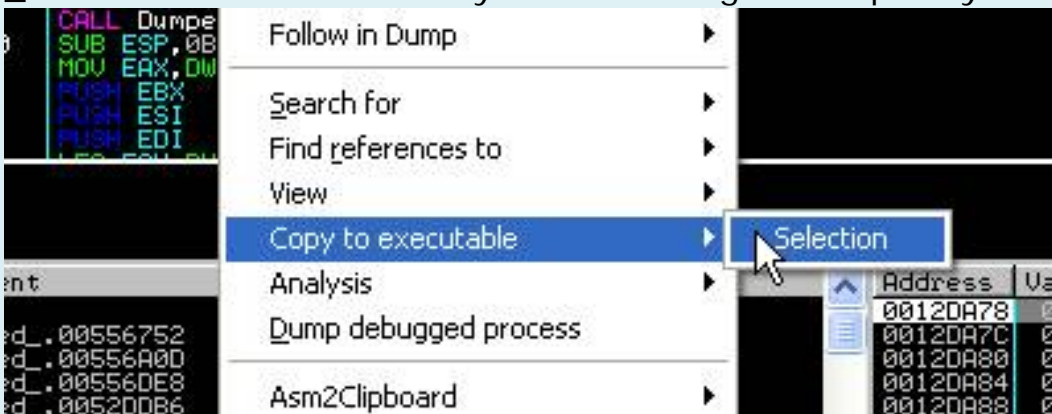
Address	Value	Comment
0012E5B4	0047280B	CALL to GetEnvironmentVariableA
0012E5B8	00D6993C	VarName = "GR_PROTECTED"
0012E5BC	00D6995C	Buffer = 00D6995C
0012E5C0	000000FF	BufSize = FF (255.)
0012E5C4	005E3410	Dumped_.005E3410
0012E5C8	0058FA48	ASCII "RuU"
0012E5CC	FFFFFFFF	
0012E5D0	77F6379E	RETURN to ntdll.77F6379E from nt
0012E5D4	00150000	

Address	Hex dump	Disassembly	Comment
0047280B	85C0	TEST EAX,EAX	
0047280D	75 37	JNZ SHORT Dumped_.00472846	
0047280F	8D4D BC	LEA ECX,DWORD PTR SS:[EBP-44]	
00472812	C645 FC 15	MOV BYTE PTR SS:[EBP-41],15	
00472816	E8 348A0000	CALL Dumped_.0054B24F	
00472818	8D4D E0	LEA ECX,DWORD PTR SS:[EBP-20]	
0047281E	C645 FC 0E	MOV BYTE PTR SS:[EBP-41],0E	
00472822	E8 288A0000	CALL Dumped_.0054B24F	

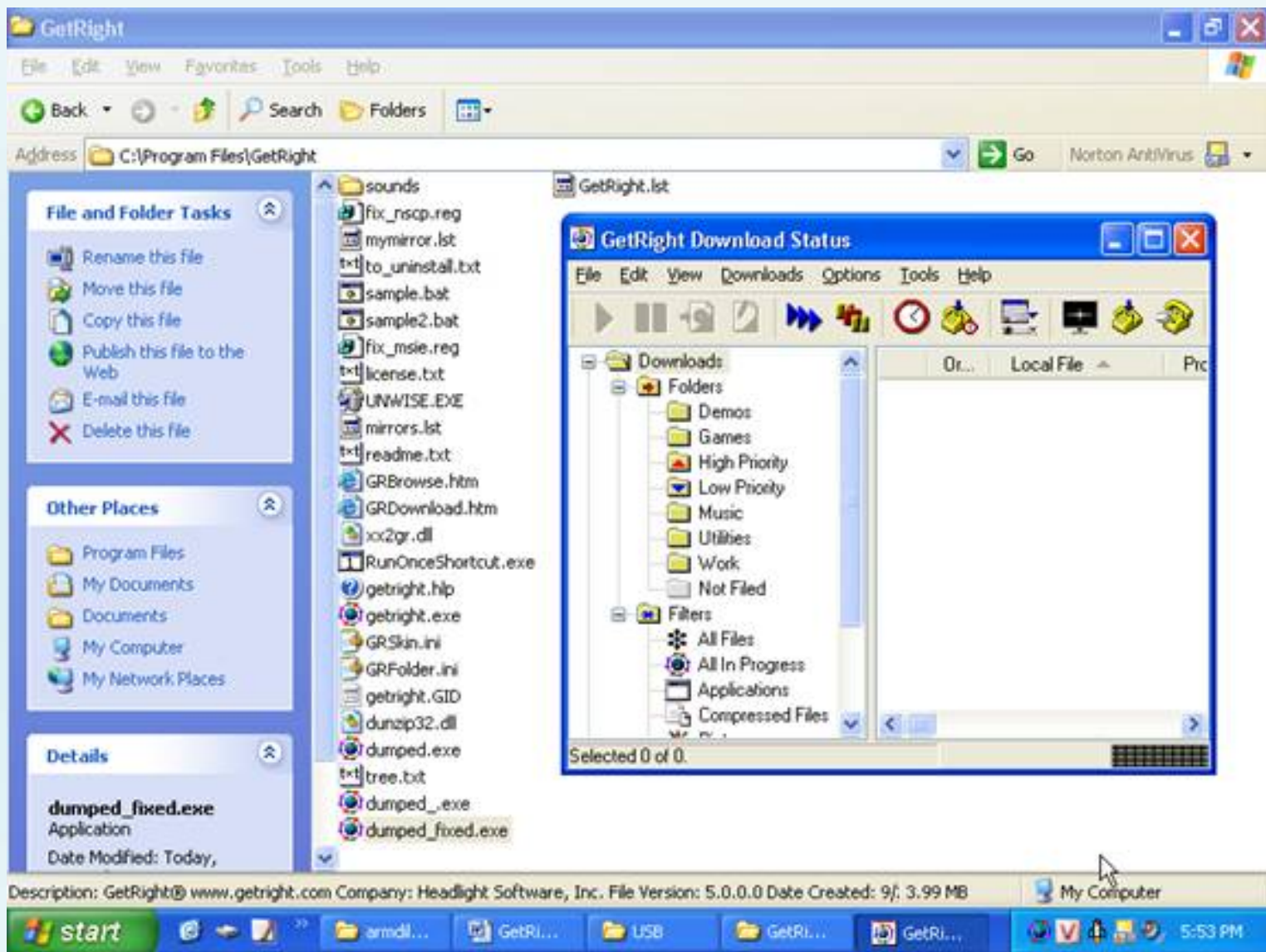
_Patch 7537 to EB37. Continue, F9, Ctrl + F9, F7:

Address	Hex dump	Disassembly	Comment
0043AA39	85C0	TEST EAX,EAX	
0043AA3B	6A FF	PUSH -1	
0043AA3D	8D4D E8	LEA ECX,DWORD PTR SS:[EBP-1]	
0043AA40	75 36	JNZ SHORT Dumped_.0043AA78	
0043AA42	E8 490C1100	CALL Dumped_.0054B690	
0043AA47	6A 01	PUSH 1	
0043AA49	FF15 6CA05E00	CALL DWORD PTR DS:[<&user32.MessageBeep	USER32.Messa
0043AA4F	53	PUSH EBX	
0043AA50	53	PUSH EBX	
0043AA51	6A 10	PUSH 10	

_Lai Patch 7536 to EB36! If you do run it again completely. Ok:



_Save The dumped_fixed.exe



_Unpacked Done!
Conclusion I.
Bye!

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini, hoadongnoi, dqtlm, hosiminh, Nilrem, fly, Madman_Hercules, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RiF, N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx OillyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 28/09/2005)

Armadillo collect sand-stone

GetRight - 6.0 Beta 3 <|> ARM 4.xx - Debug Blocker + Anti BP IAT elimination + + +
Code Splicing Nanomites



I. Intro

_Sau To do about GetRight5 tut, you can ask me to do tut GetRight 6.0 beta 3. Ok, as your mind!

II. Tools

- 1.OllyDBG - The best config debugger for ArmMUP by hacnho.
- 2.LordPE 1.4 Deluxe
- 3.Import REConstructor 1.6 Final

III. Unpacking

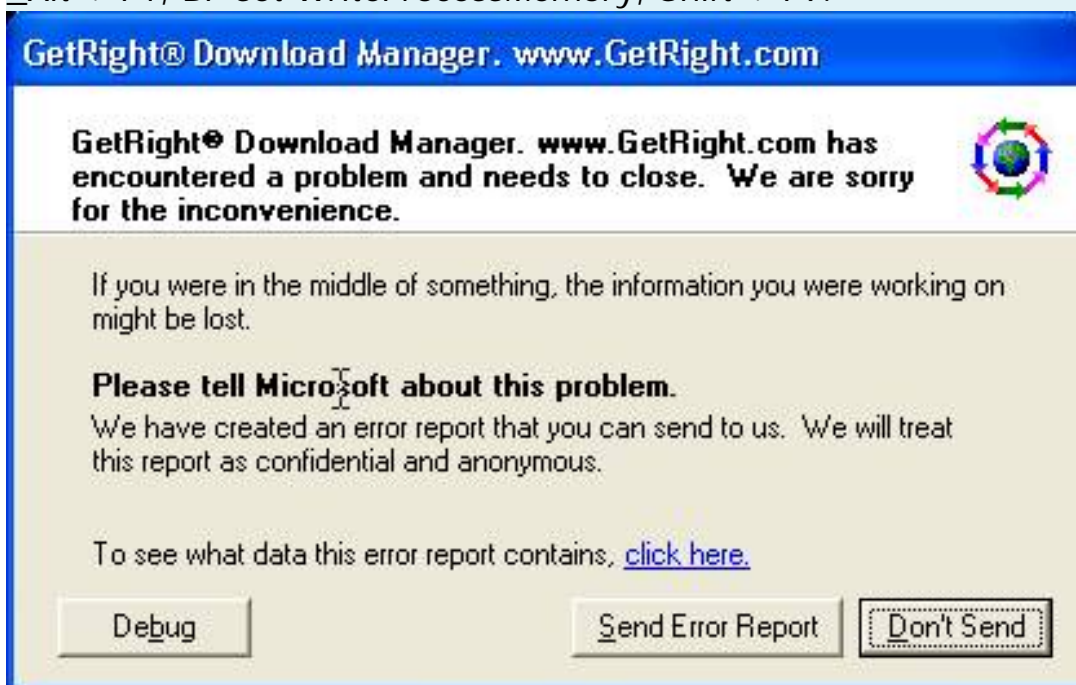
_Detect Target:

Image Name	PID	Description	User Name	CPU	Mem Usage	Handles
DkService.exe	1196	DKSERVICE.EXE	NT AUTHORITY\SYSTEM	0	8,872 K	194
spoolsv.exe	1092	Spooler SubSystem App	NT AUTHORITY\SYSTEM	0	3,556 K	119
getright.exe	1044	GetRight® Download Manager....	VIP\Super Administrator	0	9,532 K	157
svchost.exe	1004	Generic Host Process for Win32...	NT AUTHORITY\LOCAL SERVICE	0	5,540 K	131
svchost.exe	964	Generic Host Process for Win32...	NT AUTHORITY\NETWORK SER...	0	1,628 K	45
svchost.exe	856	Generic Host Process for Win32...	NT AUTHORITY\SYSTEM	0	18,272 K	1,095
svchost.exe	812	Generic Host Process for Win32...	NT AUTHORITY\SYSTEM	0	2,916 K	224
getright.exe	756	GetRight® Download Manager....	VIP\Super Administrator	0	1,620 K	39
lsass.exe	656	LSA Shell (Export Version)	NT AUTHORITY\SYSTEM	0	768 K	297
services.exe	644	Services and Controller app	NT AUTHORITY\SYSTEM	0	2,856 K	280

_Load Target:

Address	Hex dump	Disassembly	Comment
006A7000	60	PUSHAD	
006A7001	E8 00000000	CALL getright.006A7006	
006A7006	50	POP EBP	kernel32.77E814C7
006A7007	50	PUSH EAX	
006A7008	51	PUSH ECX	
006A7009	0FCA	BSWAP EDX	
006A700B	F7D2	NOT EDX	
006A700D	9C	PUSHFD	
006A700E	F7D2	NOT EDX	
006A7010	0FCA	BSWAP EDX	
006A7012	EB 0F	JMP SHORT getright.006A7023	
006A7014	B9 EB0FB8EB	MOV ECX, EBB80FEB	
006A7019	07	POP ES	Modification of se

_Alt + F1, BP set WriteProcessMemory, Shift + F9:



Restart _Ctrl + F2, Ctrl + G: WriteProcessMemory:

Address	Hex dump	Disassembly	Comment
77E61A94	55	PUSH EBP	
77E61A95	8BEC	MOV EBP, ESP	
77E61A97	51	PUSH ECX	
77E61A98	51	PUSH ECX	
77E61A99	8B45 0C	MOV ECX, DWORD PTR SS:[EBP+C]	
77E61A9C	53	PUSH EBX	
77E61A9D	8B5D 14	MOV EBX, DWORD PTR SS:[EBP+14]	
77E61AA0	56	PUSH ESI	getright.00570718
77E61AA1	8B35 B012E677	MOV ESI, DWORD PTR DS:[<&ntdll.NtProtectVi	ntdll.ZwProtectVirtu
77E61AA7	57	PUSH EDI	ntdll.77F5164E
77E61AA8	8B7D 08	MOV EDI, DWORD PTR SS:[EBP+8]	getright.<ModuleEntr
77E61AAB	8945 F8	MOV DWORD PTR SS:[EBP-8], EAX	
77E61AAE	8D45 14	LEA EAX, DWORD PTR SS:[EBP+14]	

_Shift + F9, olly break:

Address	Hex dump	Disassembly	Comment
00687338	7C 03	JE SHORT getright.0068733D	
0068733A	EB 05	JMP SHORT getright.00687341	
0068733C	E8	DB EB	
0068733D	74 FB	JE SHORT getright.0068733A	
0068733F	EB F9	JMP SHORT getright.0068733A	
00687341	EB 5F	JMP SHORT getright.006873A2	
00687343	8D55 FC	LEA EDX, DWORD PTR SS:[EBP-4]	
00687346	52	PUSH EDX	
00687347	6A 02	PUSH 2	
00687349	68 14E26800	PUSH getright.0068E214	
0068734E	8B45 10	MOV EAX, DWORD PTR SS:[EBP+10]	
00687351	50	PUSH EAX	
00687352	8B4D 08	MOV ECX, DWORD PTR SS:[EBP+8]	
00687355	8B11	MOV EDX, DWORD PTR DS:[ECX]	
00687357	52	PUSH EDX	
00687358	FF15 10716800	CALL NEAR DWORD PTR DS:[<&KERNEL32.WriteProcessMemory	pBytesWritten = 7FFE0304 BytesToWrite = 2 Buffer = getright.0068E214 getright.<ModuleEntryPoint> Address = 1 ntdll.77F75801 hProcess = 7FFE0304 WriteProcessMemory
0068735E	50	PUSH EAX	
0068735F	F7D0	NOT EAX	
00687361	0FC8	BSWAP EAX	

_Follow In dump> Immediate Constant:

getright.0068E214
DWORD PTR SS:[EBP+10]
getright.
Address =
ntdll.77F75801
hProcess = 7FFE0304
WriteProcessMemory

getright.00687366
getright.00687395

CHAR 'p'

Selection
Immediate constant
Implicit stack address

Backup
Copy
Binary
Assemble
Label
Comment
Breakpoint
Hit trace
Run trace
New origin here
Go to
Follow in Dump
Search for
Find references to

Space
:
;
Ctrl+Gr

Address Value

_Trong Window dump:

Address	Hex dump	ASCII
0068E214	60 E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00	'.'.....
0068E224	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E234	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E244	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E254	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E264	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E274	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E284	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E294	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E2A4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E2B4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E2C4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

_Change The EBFE:

Address	Hex dump	ASCII
0068E214	EB FE 00 00 00 00 00 00 00 00 00 00 00 00 00 00	'.'.....
0068E224	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E234	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E244	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E254	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E264	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E274	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E284	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E294	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E2A4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E2B4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0068E2C4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

_F9, BP WaitForDebugEvent:

New thread with ID 000005F4 created

Address	Value	Comment
0012BC94	00682EEF	CALL to WaitForDebugEvent from getright.00
0012BC98	0012CD68	pDebugEvent = 0012CD68
0012BC9C	000003E8	Timeout = 1000. ms
0012BCA0	0012FF04	
0012BCA4	00000000	
0012BCA8	00692CF3	getright.00692CF3
0012BCAC	00000000	
0012BCB0	00000000	
0012BCB4	00000000	
0012BCB8	877BAA34	
0012BCBC	00000000	
0012BCC0	00000000	

_Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
00682EEF	. 85C0	TEST EAX, EAX	
00682EF1	. 0F84 AC260000	JE getright.006855A3	
00682EF7	. 8B85 FCFDFFFF	MOV EAX, DWORD PTR SS:[EBP-204]	ntdll.77F63701
00682EFD	. 25 FF000000	AND EAX, 0FF	
00682F02	. 85C0	TEST EAX, EAX	
00682F04	. 74 13	JE SHORT getright.00682F19	
00682F06	. 8B0D 24E36B00	MOV ECX, DWORD PTR DS:[6BE324]	
00682F08	. 8B0D 24E36B00	MOV ECX, DWORD PTR DS:[6BE324]	

_Patch To:

Address	Hex dump	Disassembly	Comment
00682EEF	68 EC040000	PUSH 4EC	
00682EF4	E8 3AC38277	CALL kernel32.DebugActiveProcessStop	
00682EF9	90	NOP	
00682EFA	90	NOP	
00682EFB	90	NOP	
00682EFC	90	NOP	
00682EFD	. 25 FF000000	AND EAX, 0FF	
00682F02	. 85C0	TEST EAX, EAX	
00682F04	. 74 13	JE SHORT getright.00682F19	
00682F06	. 8B0D 24E36B00	MOV ECX, DWORD PTR DS:[6BE324]	

_F8 Trace through:

Address	Hex dump	Disassembly	Registers (FPU)
00682EEF	68 EC040000	PUSH 4EC	EAX 00000001
00682EF4	E8 3AC38277	CALL kernel32.DebugActiveProcessStop	ECX 0012BC84
00682EF9	90	NOP	EDX 7FFE0304
00682EFA	90	NOP	EBX 00692CF3 getright.00692CF3
00682EFB	90	NOP	ESP 0012BCA0
00682EFC	90	NOP	EBP 0012D788
00682EFD	. 25 FF000000	AND EAX, 0FF	ESI 00002710
00682F02	. 85C0	TEST EAX, EAX	EDI 0012C2F4
00682F04	. 74 13	JE SHORT getright.00682F19	EIP 00682F06

_Mo A window OllyDBG other, Attach, F9, F12:

Address	Hex dump	Disassembly	Comment
006A7000	- EB FE	JMP SHORT getright.<ModuleEntryPoint>	
006A7002	0000	ADD BYTE PTR DS:[EAX], AL	
006A7004	0000	ADD BYTE PTR DS:[EAX], AL	
006A7006	5D	POP EBP	kernel32.77E814C7
006A7007	50	PUSH EAX	
006A7008	51	PUSH ECX	
006A7009	0FCA	BSWAP EDX	
006A700B	F7D2	NOT EDX	

_Patch To:

Address	Hex dump	Disassembly	Comment
006A7000	60	PUSHAD	
006A7001	E8 00000000	CALL getright.006A7006	
006A7006	5D	POP EBP	kernel32.77E814C7
006A7007	50	PUSH EAX	
006A7008	51	PUSH ECX	
006A7009	0FCA	BSWAP EDX	

_Den Find at Magic Jump, as many things, I chán then, should write the script automatically find and fix Magic Jump always right is neat bu J

```
var GetModuleHandleA
var AddressOfMagicJump
var LenOfMagicJump
```

GPA "GetModuleHandleA", "kernel32.dll"

```
mov GetModuleHandleA, $ RESULT
```

```
bphws GetModuleHandleA, "x"
```

```
repeat:
```

```
esto
```

```
rtu
```

```
find eip, # 0F84 ???????????????????? 74 ?????????? EB? #
```

```
Cmp $ result, 0
```

```
je repeat
```

```
bphwc GetModuleHandleA
```

```
mov AddressOfMagicJump, $ RESULT
```

```
mov LenOfMagicJump, AddressOfMagicJump
```

```
add LenOfMagicJump, 2
```

```
mov LenOfMagicJump, [LenOfMagicJump]
```

```
inc LenOfMagicJump
```

```
mov [AddressOfMagicJump], 0E9
```

```
inc AddressOfMagicJump
```

```
mov [AddressOfMagicJump], LenOfMagicJump
```

```
CMT $ result, "<- MagicJump fixed"
```

```
msg "MagicJump fixed! Now, the next step is yours: OEP Finder!"
```

```
ret
```

_Sau When running the script:

Address	Hex dump	Disassembly	Comment
01175207	8B0D 3C1E1A01	MOV ECX, DWORD PTR DS:[11A1E3C]	
0117520D	89040E	MOV DWORD PTR DS:[ESI+ECX], EAX	kernel32.77E60000
011752E0	A1 3C1E1A01	MOV EAX, DWORD PTR DS:[11A1E3C]	
011752E5	391C06	CMP DWORD PTR DS:[ESI+EAX], EBX	
011752E8	75 16	JNZ SHORT 01175300	
011752EA	8D85 B4FEFFFF	LEA EAX, DWORD PTR SS:[EBP-14C]	
011752F0	50	PUSH EAX	kernel32.77E60000
011752F1	FF15 B8421901	CALL NEAR DWORD PTR DS:[11942B8]	kernel32.LoadLibrary
011752F7	8B0D 3C1E1A01	MOV ECX, DWORD PTR DS:[11A1E3C]	
011752FD	89040E	MOV DWORD PTR DS:[ESI+ECX], EAX	kernel32.77E60000
01175300	A1 3C1E1A01	MOV EAX, DWORD PTR DS:[11A1E3C]	
01175305	391C06	CMP DWORD PTR DS:[ESI+EAX], EBX	
01175308	E9 30010000	JMP 0117543D	<- MagicJump fixed
0117530D	0033	ADD BYTE PTR DS:[EBX], DH	
0117530F	C9	LEAVE	
01175310	8B07	MOV EAX, DWORD PTR DS:[EDI]	
01175312	3918	CMP DWORD PTR DS:[EAX], EBX	
01175314	74 06	JE SHORT 0117531C	
01175316	41	INC ECX	kernel32.77E7B5E1
01175317	83C0 0C	ADD ECX, 0C	

CreateThread _Dat breakpoint:

Address	Value	Comment
0012D74C	0117B723	CALL to CreateThread from 0117B71D
0012D750	00000000	pSecurity = NULL
0012D754	00000000	StackSize = 0
0012D758	0117BF33	ThreadFunction = 0117BF33
0012D75C	00000000	pThreadParm = NULL
0012D760	00000000	CreationFlags = 0
0012D764	0012D770	pThreadId = 0012D770
0012D768	0012FF04	
0012D76C	006BDE08	getright.006BDE08
0012D770	00000001	
0012D774	0012DEEC	
0012D778	0118DE3A	RETURN to 0118DE3A from 0117B694

_Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
0117B723	5F	POP EDI	0012FF04
0117B724	5E	POP ESI	0012FF04
0117B725	C9	LEAVE	
0117B726	C3	RETN	
0117B727	55	PUSH EBP	
0117B728	8BEC	MOV EBP, ESP	
0117B72A	81EC 28010000	SUB ESP, 128	
0117B730	56	PUSH ESI	
0117B731	57	PUSH EDI	
0117B732	BE 249C1901	MOV ESI, 1199C24	ASCII "MainClass"
0117B737	80BD D8FEFFFF	LEA EDI, DWORD PTR SS:[EBP-128]	
0117B73D	A5	MOVS DWORD PTR ES:[EDI], DWORD PTR DS:[ESI]	

_Ctrl + F9, F8. Scroll down a little:

Address	Hex dump	Disassembly	Comment
0118DE83	030D C8DF1901	ADD ECX, DWORD PTR DS:[119DFC8]	getright.
0118DE89	85D2	TEST EDX, EDX	
0118DE8B	75 18	JNZ SHORT 0118DEA5	
0118DE8D	8B50 74	MOV EDX, DWORD PTR DS:[EAX+74]	
0118DE90	FF76 18	PUSH DWORD PTR DS:[ESI+18]	
0118DE93	3350 70	XOR EDX, DWORD PTR DS:[EAX+70]	
0118DE96	FF76 14	PUSH DWORD PTR DS:[ESI+14]	
0118DE99	3350 0C	XOR EDX, DWORD PTR DS:[EAX+C]	
0118DE9C	FF76 10	PUSH DWORD PTR DS:[ESI+10]	
0118DE9F	2BCA	SUB ECX, EDX	
0118DEA1	FFD1	CALL NEAR ECX	kernel32.
0118DEA3	EB 1D	JMP SHORT 0118DEC2	
0118DEA5	83FA 01	CMP EDX, 1	
0118DEA8	75 1A	JNZ SHORT 0118DEC4	
0118DEAA	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
0118DEAD	8B50 74	MOV EDX, DWORD PTR DS:[EAX+74]	
0118DEB0	3350 70	XOR EDX, DWORD PTR DS:[EAX+70]	
0118DEB3	FF76 08	PUSH DWORD PTR DS:[ESI+8]	
0118DEB6	3350 0C	XOR EDX, DWORD PTR DS:[EAX+C]	
0118DEB9	6A 00	PUSH 0	
0118DEBB	FF76 0C	PUSH DWORD PTR DS:[ESI+C]	getright.
0118DEBE	2BCA	SUB ECX, EDX	
0118DEC0	FFD1	CALL NEAR ECX	kernel32.
0118DEC2	8BD8	MOV EBX, EAX	
0118DEC4	5F	POP EDI	
0118DEC5	8BC3	MOV EAX, EBX	
0118DEC7	5E	POP ESI	getright.
0118DEC9	5B	POP EBX	

_F9, F7: OEP:

Address	Hex dump	Disassembly	Comment
00585E90	55	PUSH EBP	
00585E91	8BEC	MOV EBP, ESP	
00585E93	6A FF	PUSH -1	
00585E95	68 98B05F00	PUSH getright.005FB098	
00585E9A	68 189E5800	PUSH getright.00589E18	
00585E9F	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
00585EA5	50	PUSH EAX	getright.006B7370
00585EA6	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
00585EAD	83C4 A8	ADD ESP, -58	
00585EB0	53	PUSH EBX	
00585EB1	56	PUSH ESI	getright.006BDE08
00585EB2	57	PUSH EDI	
00585EB3	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
00585EB6	FF15 B8CD0803	CALL NEAR DWORD PTR DS:[308CDB8]	kernel32.GetVersion
00585EBC	33D2	XOR EDX, EDX	
00585EBE	8AD4	MOV DL, AH	
00585EC0	8915 E8416500	MOV DWORD PTR DS:[6541E8], EDX	

_Dom The address of the function GetVersion bit immediately IAT elimination J. Phone, so it is quiet there, fix what has Splicing Code. Alt + M to find signs of Splicing Code:

02FA0000	000CE000				Priv	RW	RW	
03084000	0000A000				Priv	RW	RW	
030A0000	00001000				Map	RW	RW	
030B0000	0000C000				Priv	RW	RW	
030C0000	00006000				Priv	RW	RW	
030D0000	00003000				Priv	RW	RW	
03110000	00001000				Map	RW	RW	
03120000	00001000				Map	RW	RW	
03130000	0001A000				Priv	RW	RW	
03171000	00002000				Priv	RW	RW	
03330000	00003000				Priv	RW	RW	
03340000	00010000				Priv	RW	RW	
03740000	00003000				Priv	RW	RW	
03780000	00001000				Priv	RW	RW	
03800000	00003000				Priv	RW	RW	
03850000	00020000				Priv	RWE	RWE	
0396E000	00002000				Priv	RW	Gua: RW	
629C0000	00001000	LPK		PE header	Imag	R	RWE	
629C1000	00004000	LPK	.text	code, import	Imag	R	RWE	
629C5000	00001000	LPK	.data	data	Imag	R	RWE	
629C6000	00001000	LPK	.rsrc	resources	Imag	R	RWE	
629C7000	00001000	LPK	.reloc	relocations	Imag	R	RWE	
666F0000	00001000	inetnib1		PE header	Imag	R	RWE	

He he!

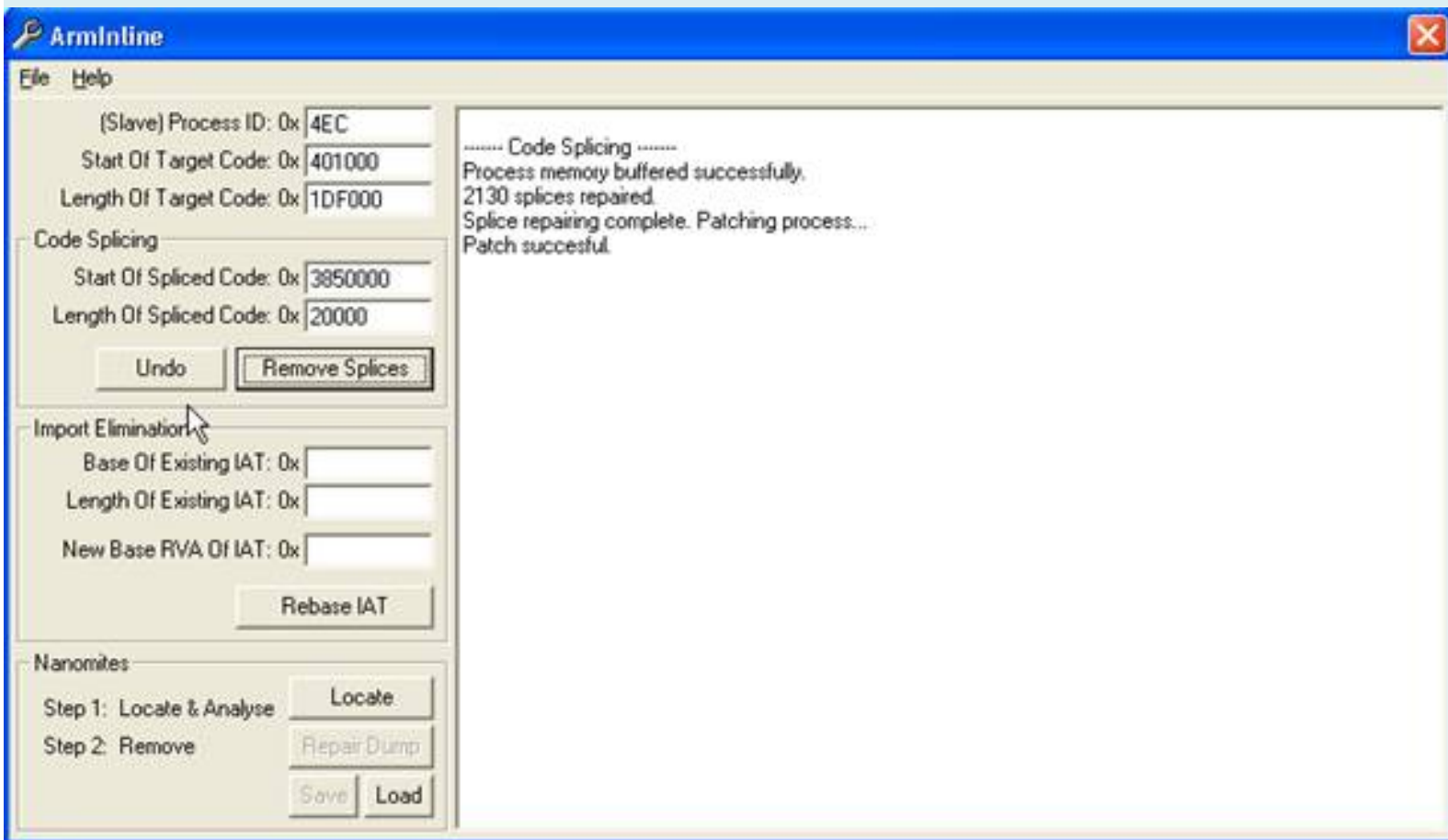
Address	Size	Owner	Section	Contains	Type	Access
00290000	00034000				Map	R
002D0000	00041000				Map	R
00320000	00006000				Map	R
00330000	00005000				Map	R E
003F0000	00002000				Map	R E
00400000	00001000	getright			Imag	R
00401000	001DF000	getright	.text		Imag	R
005E0000	0005C000	getright	.rda	exports	Imag	R
0063C000	0001B000	getright	.data	data	Imag	R
00657000	00050000	getright	.text1	code	Imag	R
006A7000	00010000	getright	.adata	SFX	Imag	R
006B7000	00020000	getright	.data1	imports	Imag	R
006D7000	00240000	getright	.pdata		Imag	R
00917000	002B5000	getright	.rsrc	resources	Imag	R

0000044C	UKService		C:\Program f
000004D0	MDM		C:\Program f
000004EC	getright		C:\Program f
00000508	TSCHelp	C:\Program Files\TechSmith	C:\Program f
0000050C	MTNMINRN	Arrow Suite	C:\Program f

_Mo The ArmInline out, fill in information as follows:

(Slave) Process ID: 0x	4EC
Start Of Target Code: 0x	401000
Length Of Target Code: 0x	1DF000
Code Splicing	
Start Of Spliced Code: 0x	3850000
Length Of Spliced Code: 0x	20000
<input type="button" value="Undo"/> <input type="button" value="Remove Splices"/>	

Remove _Click Splices:



_Tim IAT Info:

IAT Start:

Address	Value	Comment
03084020	00040009	
03084024	00080101	
03084028	76B40000	WINMM.76B40000
0308402C	77E60000	kernel32.77E60000
03084030	77D40000	USER32.77D40000
03084034	77C70000	GDI32.77C70000
03084038	763B0000	comdlg32.763B0000
0308403C	73000000	WINSPOOL.73000000
03084040	77DD0000	ADVAPI32.77DD0000
03084044	011F0000	OFFSET SHELL32.#584
03084048	71950000	OFFSET COMCTL32.#237
0308404C	74D30000	oledlg.74D30000

IAT End:

Address	Value	Comment
0308D408	77E7E358	kernel32.GetCommandLineA
0308D40C	77E7B332	kernel32.GetProcAddress
0308D410	011766B1	
0308D414	00000000	
0308D418	0160017D	SHELL32.0160017D
0308D41C	000D1001	
0308D420	011C6BC8	
0308D424	03149620	
0308D428	9F161F77	
0308D42C	E48B1D48	
0308D430	C361E292	
0308D434	F9CE1FC7	

_Ta Are:

IAT Start: 03084028 76B40000 WINMM.76B40000
IAT End: 0308D418 0160017D SHELL32.0160017D
IAT Len: 93F0

Address	Size	Owner	Section	Contains	Type	Access	Initi
00290000	00034000				Map	R	R
002D0000	00041000				Map	R	R
00320000	00006000				Map	R	R
00330000	00005000				Map	R E	R E
003F0000	00002000				Map	R E	R E
00400000	00001000	getright			Imag	R	RWE
00401000	0010F000	getright	.text		Imag	R	RWE
005E0000	0005C000	getright	.rdata	exports	Imag	R	RWE
0063C000	0001B000	getright	.data	data	Imag	R	RWE
00657000	00050000	getright	.text1	code	Imag	R	RWE
006A7000	00010000	getright	.adata	SFX	Imag	R	RWE
006B7000	00020000	getright	.data1	imports	Imag	R	RWE
006D7000	00240000	getright	.pdata		Imag	R	RWE
00917000	002B5000	getright	.rsrc	resources	Imag	R	RWE
00B00000	00103000				Map	R	R
00CF0000	00008000				Priv	RW	RW

_Dien To ArmInline:

Import Elimination

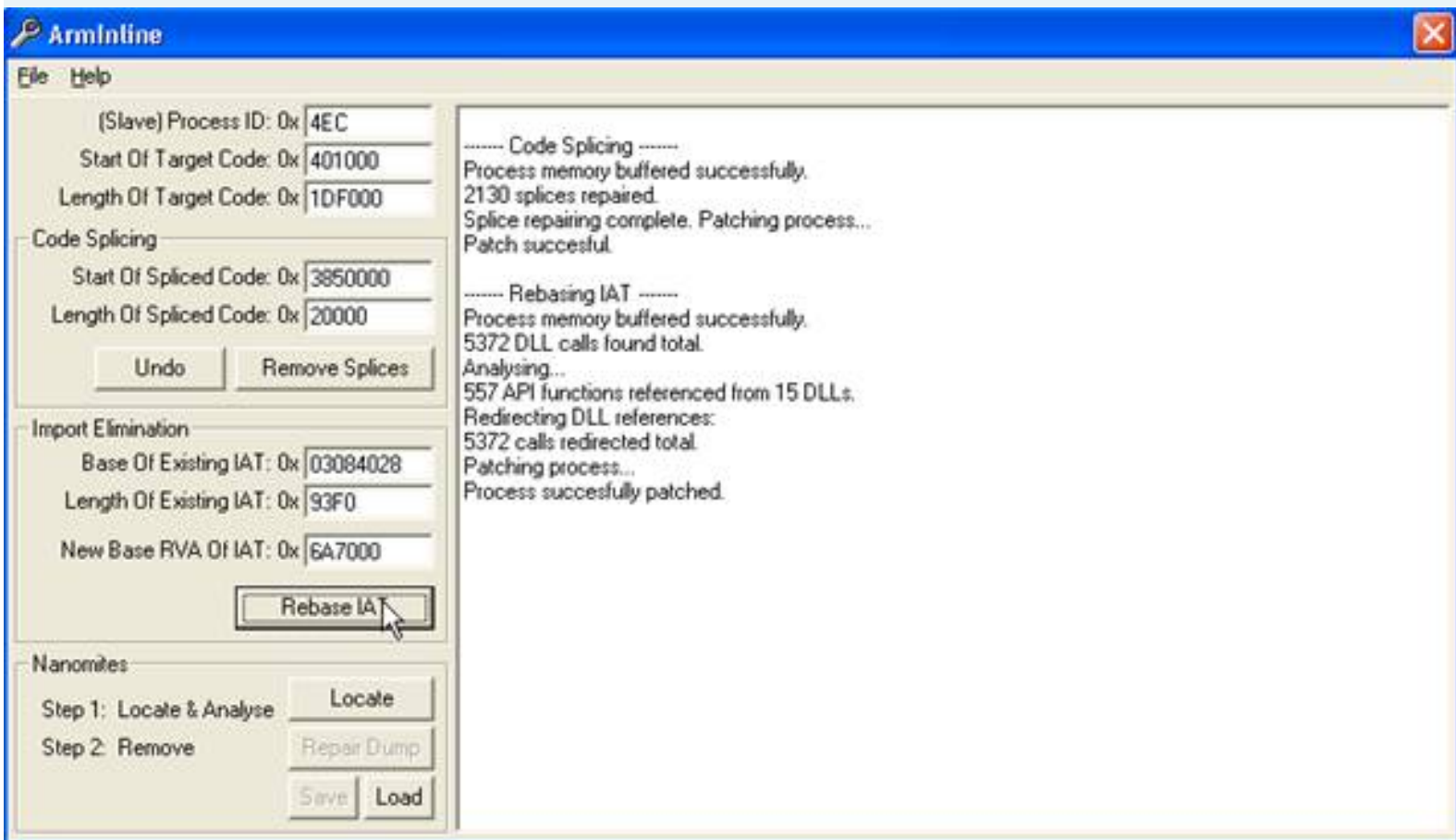
Base Of Existing IAT: 0x03084028

Length Of Existing IAT: 0x93F0

New Base RVA Of IAT: 0x6A7000

Rebase IAT

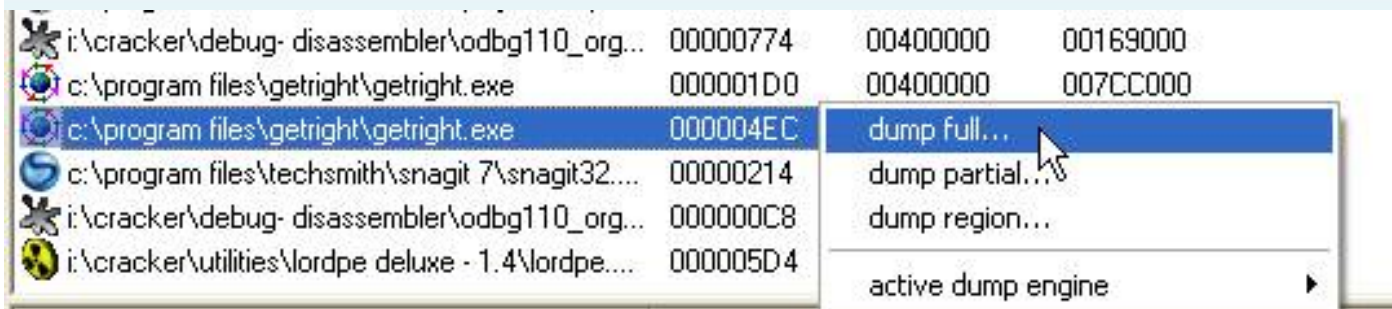
Rebase _Click IAT:



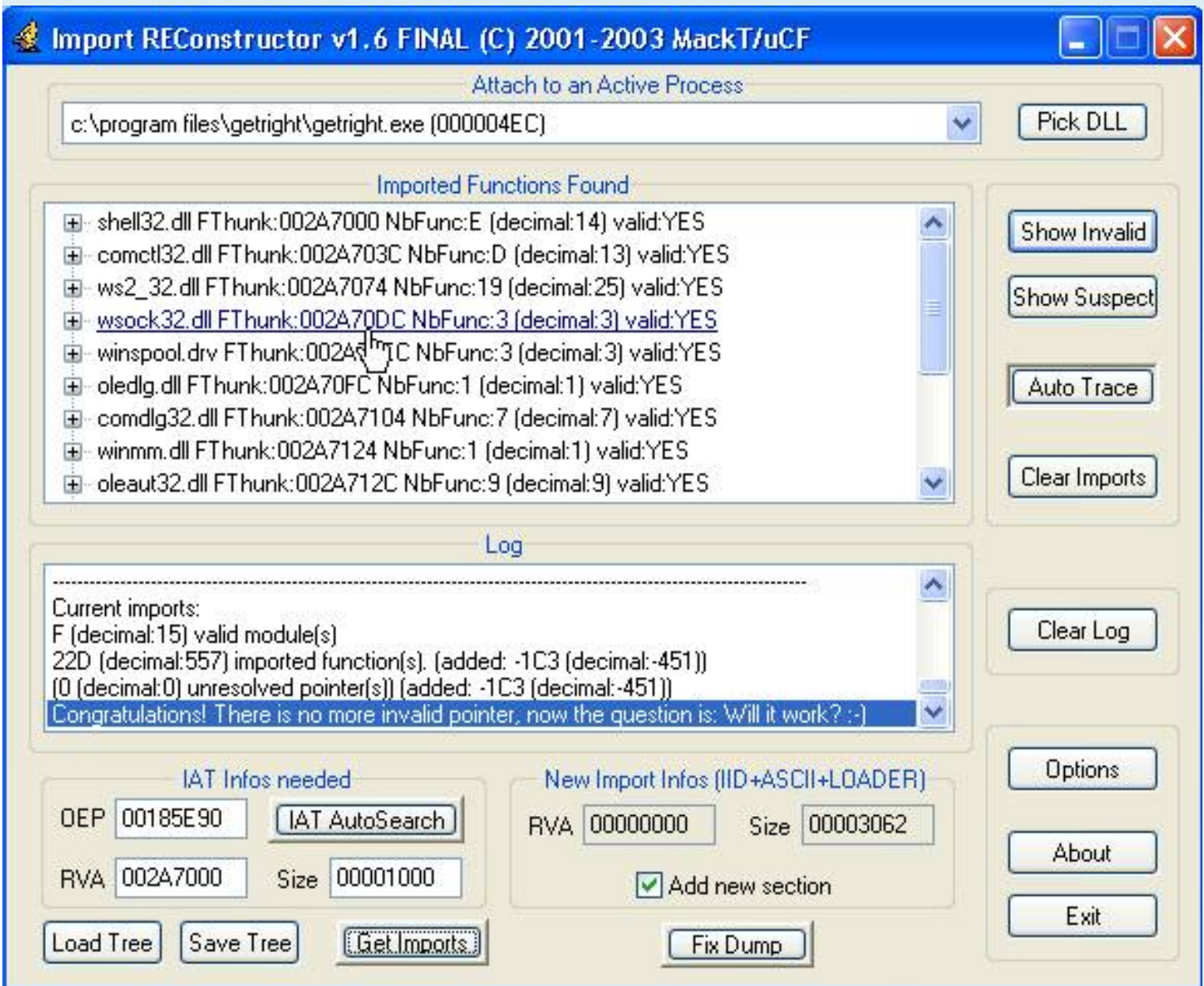
_Quay Back to the CPU:

Address	Hex dump	Disassembly	Comment
00585E90	55	PUSH EBP	
00585E91	8BEC	MOV EBP, ESP	
00585E93	6A FF	PUSH -1	
00585E95	68 98B05F00	PUSH getright.005FB098	
00585E9A	68 189E5800	PUSH getright.00589E18	
00585E9F	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
00585EA5	50	PUSH EAX	getright.006B7370
00585EA6	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
00585EAD	83C4 A8	ADD ESP, -58	
00585EB0	53	PUSH EBX	
00585EB1	56	PUSH ESI	getright.006BDE08
00585EB2	57	PUSH EDI	
00585EB3	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
00585EB6	FF15 20786A00	CALL NEAR DWORD PTR DS:[6A7820]	kernel32.GetVersion
00585EBC	33D2	XOR EDX, EDX	
00585EBE	8AD4	MOV DI, AH	
00585EC0	8915 F8416500	MOV DWORD PTR DS:[6541F8], EAX	

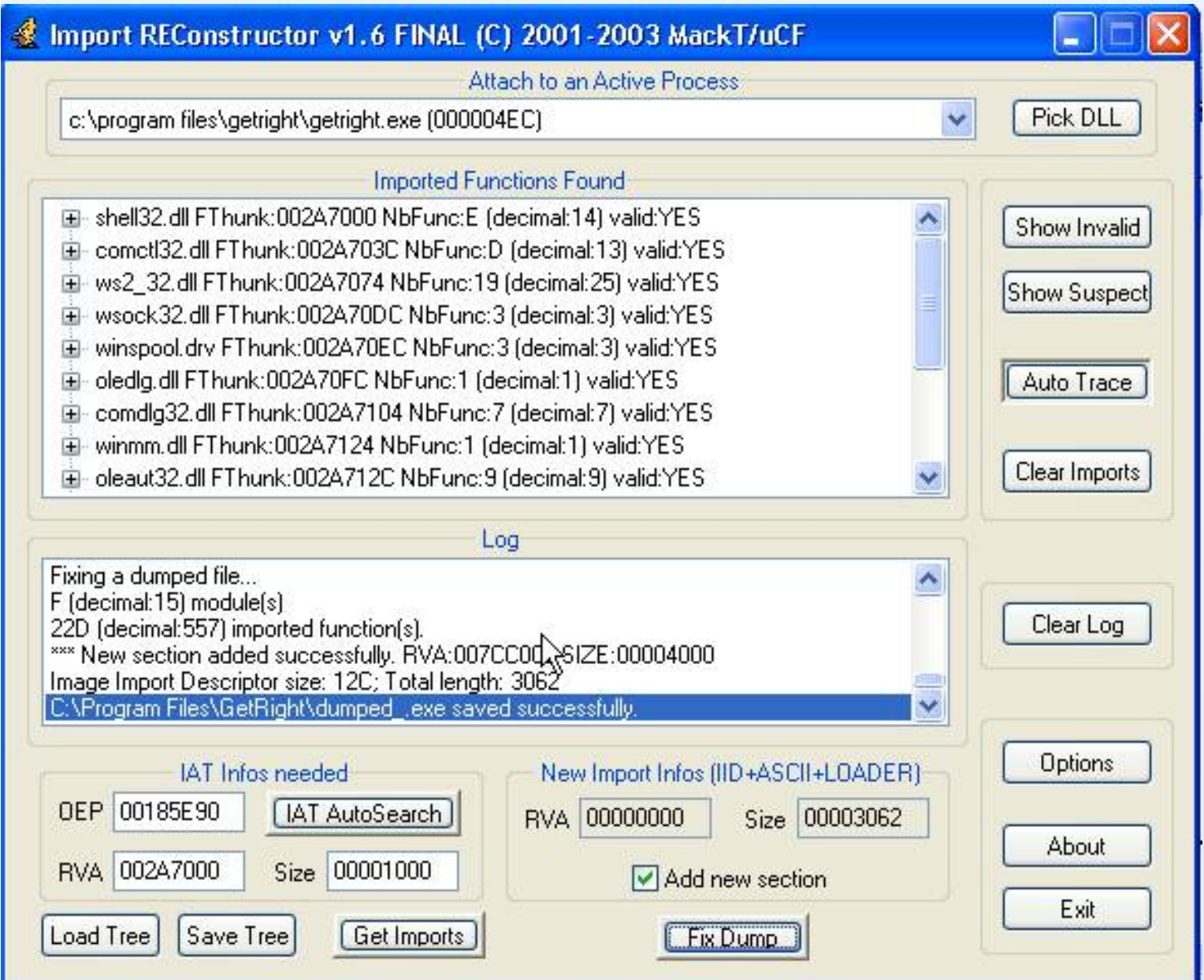
Full _LordPE dump:



_Mo ImpREC and filled as follows:



_Fix Dump:



Run Try dumped.exe:

GetRight® Download Manager. www.GetRight.com

GetRight® Download Manager. www.GetRight.com has encountered a problem and needs to close. We are sorry for the inconvenience.



If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.

We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, [click here](#).

Merde, Italy chang he thắng it is 5.0. OK, dumped.exe Load up, Ctrl + G:

Enter expression to follow

_Toi Here:

Address	Hex dump	Disassembly	Comment
77E7D173	55	PUSH EBP	
77E7D174	8BEC	MOV EBP, ESP	
77E7D176	83EC 20	SUB ESP, 20	
77E7D179	53	PUSH EBX	
77E7D17A	FF75 08	DWORD PTR SS:[EBP+8]	
77E7D17D	8D45 E0	LEA EAX, DWORD PTR SS:[EBP-20]	
77E7D180	50	PUSH EAX	
77E7D181	FF15 8012E677	CALL NEAR DWORD PTR DS:[<ntdll.RtlInitString	ntdll.RtlInitString
77E7D187	6A 01	PUSH 1	
77E7D189	8D45 E0	LEA EAX, DWORD PTR SS:[EBP-20]	

_Sau When pressing F2 set breakpoint here, F9, Ctrl + F9, F7:

Address	Hex dump	Disassembly	Comment
00551B7F	85C0	TEST EAX, EAX	
00551B81	6A FF	PUSH -1	
00551B83	8D4D F0	LEA ECX, DWORD PTR SS:[EBP-10]	
00551B86	75 21	JNZ SHORT dumped_.00551BA9	
00551B88	E8 0DA70400	CALL dumped_.0059C29A	
00551B8D	8065 FC 00	AND BYTE PTR SS:[EBP-4], 0	
00551B91	8D4D EC	LEA ECX, DWORD PTR SS:[EBP-14]	
00551B94	E8 74A20400	CALL dumped_.0059BE00	
00551B99	834D FC FF	OR DWORD PTR SS:[EBP-4], FFFFFFFF	

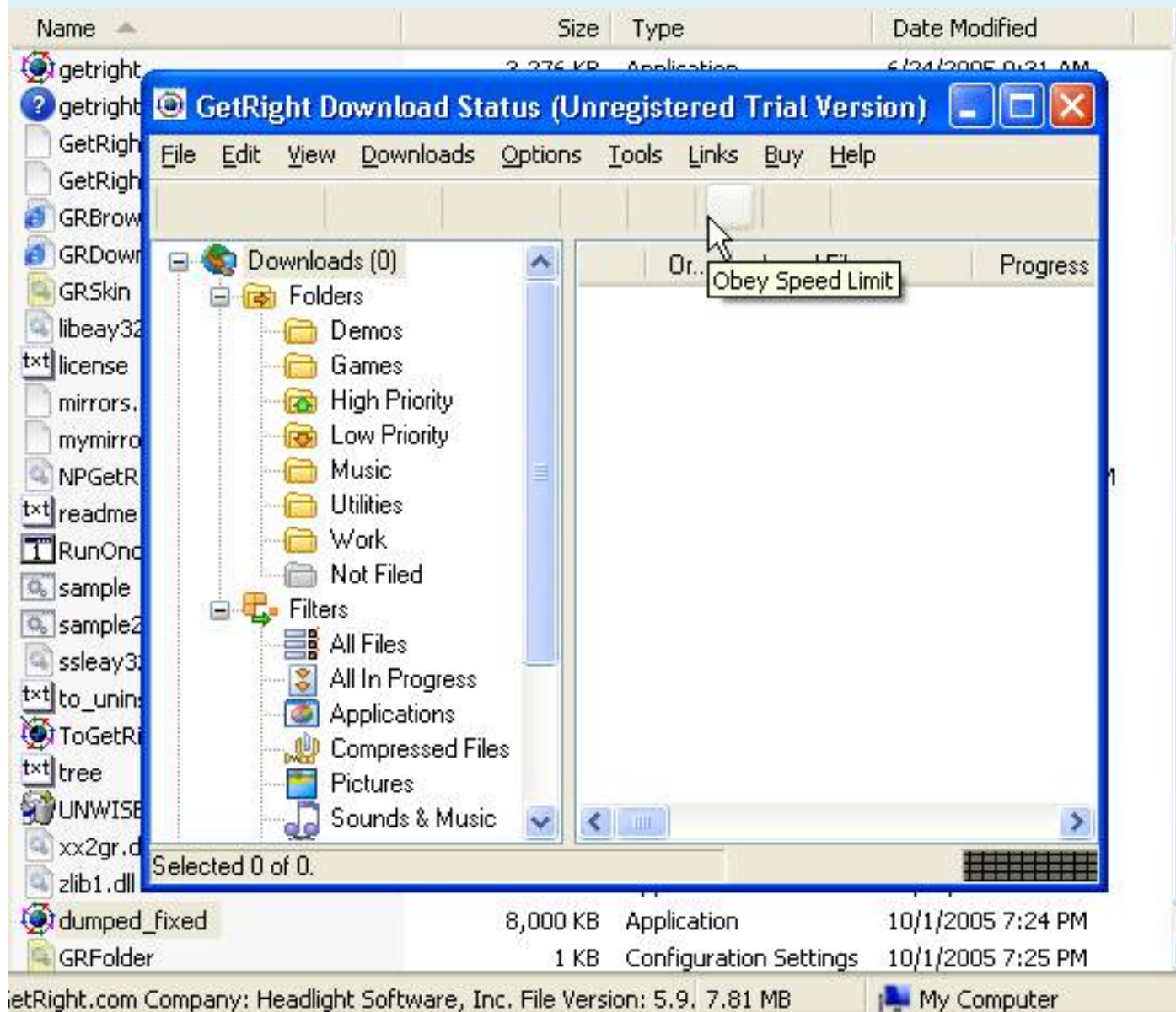
_Patch To:

The screenshot shows a debugger window with assembly code on the left and a context menu open over it. The assembly code lists addresses from 00551B83 to 00551BE0, with instructions like LEA, JMP, CALL, AND, OR, XOR, PUSH, POP, MOV, RETN, and SUB. The context menu includes options like Backup, Copy, Binary, Undo selection, Assemble, Label, Comment, Breakpoint, Run trace, Follow, New origin here, Go to, Follow in Dump, Search for, Find references to, View, Copy to executable, and Analysis. The 'Copy to executable' option is highlighted. Below the assembly code, there is a table with columns 'Address' and 'Hex dump'.

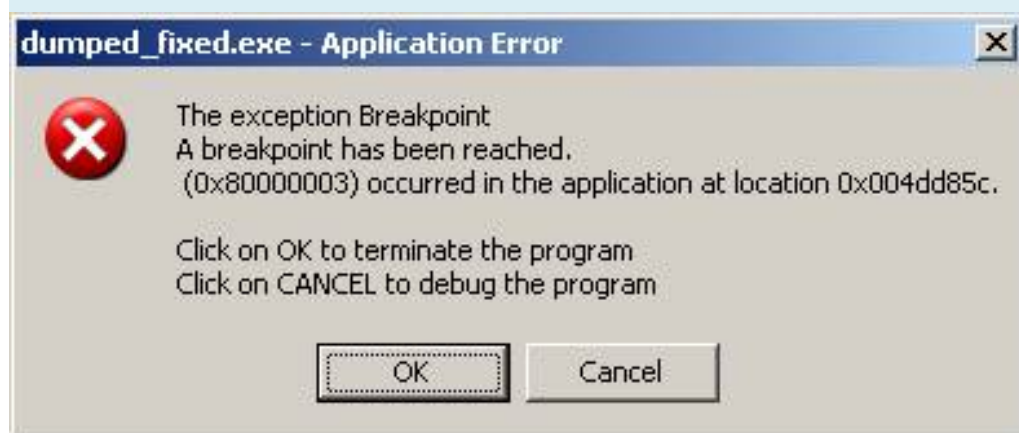
Address	Hex dump
0063C000	00 00 00 00 24 74 5A 00 DA
0063C010	1F E0 57 00 94 82 5A 00 BE
0063C020	62 91 5A 00 BB 91 5A 00 BA
0063C030	99 C8 40 00 BC B4 41 00 B7

_Save Again, run, still crash. Continue load just save the file, do the steps similar to the run completely! Save as dumped_fixed! You try to run after 5 times dumped_fixed.exe patch file:





_Sau When running, close the crash:



_ You should check with Nano or not? EtRight.exe G Load up OllyDBG. DebugBlocker defeat by hand:

- [CPU - main thread, module getright]

File View Debug Plugins Options Window Help

Paused

Address	Hex dump	Disassembly	Comment	Registers (FPU)
00687336	70 07	je short getright.0068733F		EAX 00000001
00687338	7C 03	jle short getright.0068733D		ECX 0012B968
0068733A	EB 05	jmp short getright.00687341		EDX 7C90EB94 ntdll.KiFa
0068733C	E8	db E8		EBX 00692CF3 getright.0
0068733D	74 FB	je short getright.0068733A		ESP 0012B9AC
0068733F	EB F9	jmp short getright.0068733A		EBP 0012BC8C
00687341	EB 5F	jmp short getright.006873A2		ESI 00002710
00687343	8D55 FC	lea edx,dword ptr ss:[ebp-4]		EDI 00120220 UNICODE "3
00687346	52	push edx	pBytesWritten	EIP 00687336 getright.0
00687347	6A 02	push 2	BytesToWrite = 2	C 0 ES 0023 32bit 0(FF)
00687349	68 14E26B00	push getright.006BE214	Buffer = getright.006BE	P 0 CS 001B 32bit 0(FF)
0068734E	8B45 10	mov eax,dword ptr ss:[ebp+10]	Address	A 0 SS 0023 32bit 0(FF)
00687351	50	push eax	hProcess	Z 0 DS 0023 32bit 0(FF)
00687352	8B4D 08	mov ecx,dword ptr ss:[ebp+8]	WriteProcessMemory	S 0 FS 003B 32bit 7FFD
00687355	8B11	mov edx,dword ptr ds:[ecx]		T 0 GS 0000 NULL
00687357	52	push edx		O 0 LastErr ERROR_SUCC
00687358	FF15 10716B00	call dword ptr ds:[<4KERNEL32.WriteProc		EFL 00000202 (NO,NB,NE,
0068735E	50	push eax		ST0 empty -UNORM DOAS 0
0068735F	F7D0	not eax		ST1 empty 0.0
00687361	0FC8	bswap eax		ST2 empty 0.0
				ST3 empty 0.0
				ST4 empty 0.0
				ST5 empty 0.0
				ST6 empty 0.0
				ST7 empty 0.0

Address	Value	Comment	Address	Value	Comment
006BE214	0000FEEB		0012B9AC	0012D220	UNICODE "32.dll"
006BE218	00000000		0012B9B0	00002710	
006BE21C	00000000		0012B9B4	00692CF3	getright.00692CF3
006BE220	00000000		0012B9B8	00000064	
006BE224	00000000		0012B9BC	00000064	
006BE228	00000000		0012B9C0	01080009	
006BE22C	00000000		0012B9C4	02080000	
006BE230	00000000		0012B9C8	0012B9D8	
006BE234	00000000		0012B9CC	005C0000	getright.005C0000
006BE238	00000000		0012B9D0	001527B0	

Command: bp WaitForDebugEvent

Analysing getright: 804 heuristical procedures, 450 calls to known, 1183 calls to guessed functions

- [CPU - main thread, module getright]

File View Debug Plugins Options Window Help

Address Hex dump Disassembly Comment Registers (FPU)

00682EF7	68 B0020000	push 2B0		EAX 00000001
00682EF4	E8 98741D7C	call kernel32.DebugActiveProcessStop		ECX 0012BC80
00682EF9	90	nop		EDX 7C90EB94 ntdll.KiFa
00682EFA	90	nop		EBX 00692CF3 getright.0
00682EFB	90	nop		ESP 0012BCA0
00682EFC	90	nop		EBP 0012D788
00682EFD	25 FF000000	and eax,0FF		ESI 00002710
00682F02	85C0	test eax,eax		EDI 0012C2F4
00682F04	74 13	je short getright.00682F19		EIP 00682EF9 getright.0
00682F06	8B0D 24E36B0	mov ecx,dword ptr ds:[6BE324]		C 0 ES 0023 32bit 0(FF)
00682F0C	8379 20 00	cmp dword ptr ds:[ecx+20],0		P 0 CS 001B 32bit 0(FF)
00682F10	74 07	je short getright.00682F19		A 0 SS 0023 32bit 0(FF)
00682F12	C685 FCFDFFF	mov byte ptr ss:[ebp-204],0		Z 0 DS 0023 32bit 0(FF)
00682F19	68 18E26B00	push getright.006BE218	[pCriticalSection = getr: EnterCriticalSection	S 0 FS 003B 32bit 7FFD
00682F1E	FF15 A4716B0	call dword ptr ds:[<4KERNEL32.EnterCrit:		T 0 GS 0000 NULL
00682F24	60	pushad		O 0
00682F25	33C0	xor eax,eax		O 0 LastErr ERROR_SEM_
00682F27	75 02	jnz short getright.00682F2B		EFL 00000202 (NO,NB,NE,
00682F29	EB 15	jmp short getright.00682F40		ST0 empty -UNORM DOAS 0
00682F2B	EB 33	jmp short getright.00682F60		ST1 empty 0.0

Address	Value	Comment	Address	Value	Comment
006BE214	0000F8EB		0012BCA0	0012FF04	
006BE218	00155808		0012BCA4	00000000	
006BE21C	FFFFFFFF		0012BCA8	00692CF3	getright.00692CF3
006BE220	00000000		0012BCAC	00000000	
006BE224	00000000		0012BCB0	00000000	
006BE228	00000000		0012BCB4	00000000	
006BE22C	00000000		0012BCB8	A563D236	
006BE230	00000000		0012BCBC	00000000	
006BE234	00000000		0012BCC0	00000000	
006BE238	00000000		0012BCC4	00000000	

Command: bp WaitForDebugEvent

_Can Find signs of INT3:

Address	Hex dump	Disassembly	Comment	Registers (FPU)
00682EF7	68 B0020000	push 2B0		EAX 00000001
00682EF4	ES 98741D7C	call kernel32.DebugActiveProcessStop		EAX 00000001
00682EF9	90	nop		EAX 00000001
00682EFA	90	nop		EAX 00000001
00682EFB	90	nop		EAX 00000001
00682EFC	90	nop		EAX 00000001
00682EFD	25 FF000000	and eax,0FF		EAX 00000001
00682F02	85C0	test eax,eax		EAX 00000001
00682F04	74 13	js short getr:		EAX 00000001
00682F06	8B0D 24E36B00	mov ecx,dword ptr [eax]		EAX 00000001
00682F0C	8379 20 00	cmp dword ptr [ecx],0		EAX 00000001
00682F10	74 07	js short getr:		EAX 00000001
00682F12	C685 FCFDFFFF	mov byte ptr [ecx],0		EAX 00000001
00682F19	68 18E26B00	push getright		EAX 00000001
00682F1E	FF15 A4716B00	call dword ptr [ecx]		EAX 00000001
00682F24	60	pushad		EAX 00000001
00682F25	33C0	xor eax,eax		EAX 00000001
00682F27	75 02	jnz short getr:		EAX 00000001
00682F29	EB 15	jmp short getr:		EAX 00000001
00682F2B	EB 33	jmp short getr:		EAX 00000001

Address	Value	Comment
006BE214	0000FEEB	
006BE218	00155808	
006BE21C	FFFFFFFF	
006BE220	00000000	
006BE224	00000000	
006BE228	00000000	
006BE22C	00000000	
006BE230	00000000	
006BE234	00000000	
006BE238	00000000	

Address	Value	Comment
0012BCA0	0012FF04	
0012BCA4	00000000	
0012BCA8	00692CF3	getright.00692CF3
0012BCAC	00000000	
0012BCB0	00000000	
0012BCB4	00000000	
0012BCB8	A563D236	
0012BCBC	00000000	
0012BCC0	00000000	
0012BCC4	00000000	

Enter constant to search for [X]

Hexadecimal

Signed

Unsigned

☐ Entire block

OK Cancel

Address	Hex dump	Disassembly	Comment
00683BB4	. E8 01ED0000	call getright.006928BA	
00683BB9	. 83C4 14	add esp,14	
00683BBC	. 68 00000500	push 50000	Style = MB_OK MB_APPLMOI
00683BC1	. 6A 00	push 0	Title = NULL
00683BC3	. 8D8D D0EEFFFD	lea ecx,dword ptr ss:[ebp-1130]	
00683BC9	. 51	push ecx	Text
00683BCA	. 6A 00	push 0	hOwner = NULL
00683BCC	. FF15 28726B00	call dword ptr ds:[<4USER32.MessageBoxA	MessageBoxA
00683BD2	> E9 ABOB0000	jmp getright.00684782	
00683BD7	> 8B15 D8736B00	mov edx,dword ptr ds:[6B73D8]	
00683BDD	. 81F2 03000000	xor edx,80000003	0xCC
00683BE3	. 3995 D4F5FFFF	cmp dword ptr ss:[ebp-A2C],edx	
00683BE9	> 0F85 3D0B0000	jnz getright.0068472C	
00683BEF	. C785 88EEFFFD	mov dword ptr ss:[ebp-1178],10	
00683BF9	. A1 D8736B00	mov eax,dword ptr ds:[6B73D8]	
00683BFE	. 3305 7C736B00	xor eax,dword ptr ds:[6B737C]	
00683C04	. 3305 AC736B00	xor eax,dword ptr ds:[6B73AC]	
00683C0A	. 8985 8CEEFFFD	mov dword ptr ss:[ebp-1174],eax	
00683C10	. 8B0D DC736B00	mov ecx,dword ptr ds:[6B73DC]	
00683C16	. 330D A8736B00	xor ecx,dword ptr ds:[6B73A8]	

_Load Dumped_fixed.exe up OllyDBG, find information about PID, Sections text:

Paused

Select process to attach

Process	Name	Window	Path
000003C0	svchost		C:\WINDOWS\system32\svchost.exe
00000410	ccEvtMgr		C:\Program Files\Common Files\Synantec
00000430	AcroRd32	Adobe Reader - [Playboys L	C:\Program Files\Adobe\Acrobat 7.0\Rea
00000460	alg		C:\WINDOWS\System32\alg.exe
00000480	TSCHelp		C:\Program Files\TechSmith\Snagit 7\TS
00000510	TurboLau	TurboLaunch	C:\Program Files\TurboLaunch\TurboLau
00000524	Explorer	GetRight	C:\WINDOWS\Explorer.EXE
00000544	spoolsv		C:\WINDOWS\system32\spoolsv.exe
00000580	ccService		C:\Program Files\Executive Software\Di
000005E0	ccApp	Norton AntiVirus	C:\Program Files\Executive Software\Di
00000620	UnKeyNT	UnKey 3.62	C:\Program Files\UnKey\UnKeyNT.exe
00000630	ctfnon	TF_FloatingLangBar_WndTitl	C:\WINDOWS\system32\ctfnon.exe
00000650	MDM		C:\Program Files\Common Files\Microsof
00000680	navapsvc		C:\Program Files\Norton AntiVirus\nav
00000710	StarWind		C:\Program Files\Alcohol Soft\Alcohol
00000904	dumped_f		C:\Program Files\GetRight\dumped_fixed
0000090C	nsnsgs	ODE Server Window	C:\Program Files\Messenger\nsnsgs.exe
00000930	winamp	Winamp Playlist Editor	C:\Program Files\Winamp\Winamp.exe
00000944	conime		C:\WINDOWS\system32\conime.exe

Attach Cancel

Registers (FPU)

EAX	00000000
ECX	0012FFB0
EDX	7C90EB94 ntdll.KiFa
EBX	7FFD0000
ESP	0012FFC4
EBP	0012FFFF
ESI	FFFFFFFF
EDI	7C910738 ntdll.7C910
EIP	00585E90 dumped_f.<
CS	001B 32bit 0(FF)
DS	001B 32bit 0(FF)
SS	0023 32bit 0(FF)
ES	0023 32bit 0(FF)
FS	003B 32bit 7FFD
GS	0000 NULL
IOPL	0000
LastErr	ERROR_SUCC
EFL	00000246 (NO,NB,E,8)
ST0	empty -UNORM 0108 00
ST1	empty 0.0
ST2	empty 0.0
ST3	empty 0.0
ST4	empty 0.0
ST5	empty 0.0
ST6	empty 1.0000000000000000
ST7	empty 1.0000000000000000

Address Value Comment

0063C000	00000000	
0063C004	005A7424	dumped_f.005A7424
0063C008	005A76DA	dumped_f.005A76DA
0063C00C	005A7AEC	dumped_f.005A7AEC
0063C010	0057E01F	dumped_f.0057E01F
0063C014	005A8294	dumped_f.005A8294
0063C018	005A8CBE	dumped_f.005A8CBE
0063C01C	005A8CDF	dumped_f.005A8CDF
0063C020	005A9162	dumped_f.005A9162
0063C024	005A91E0	dumped_f.005A91E0

Address Value Comment

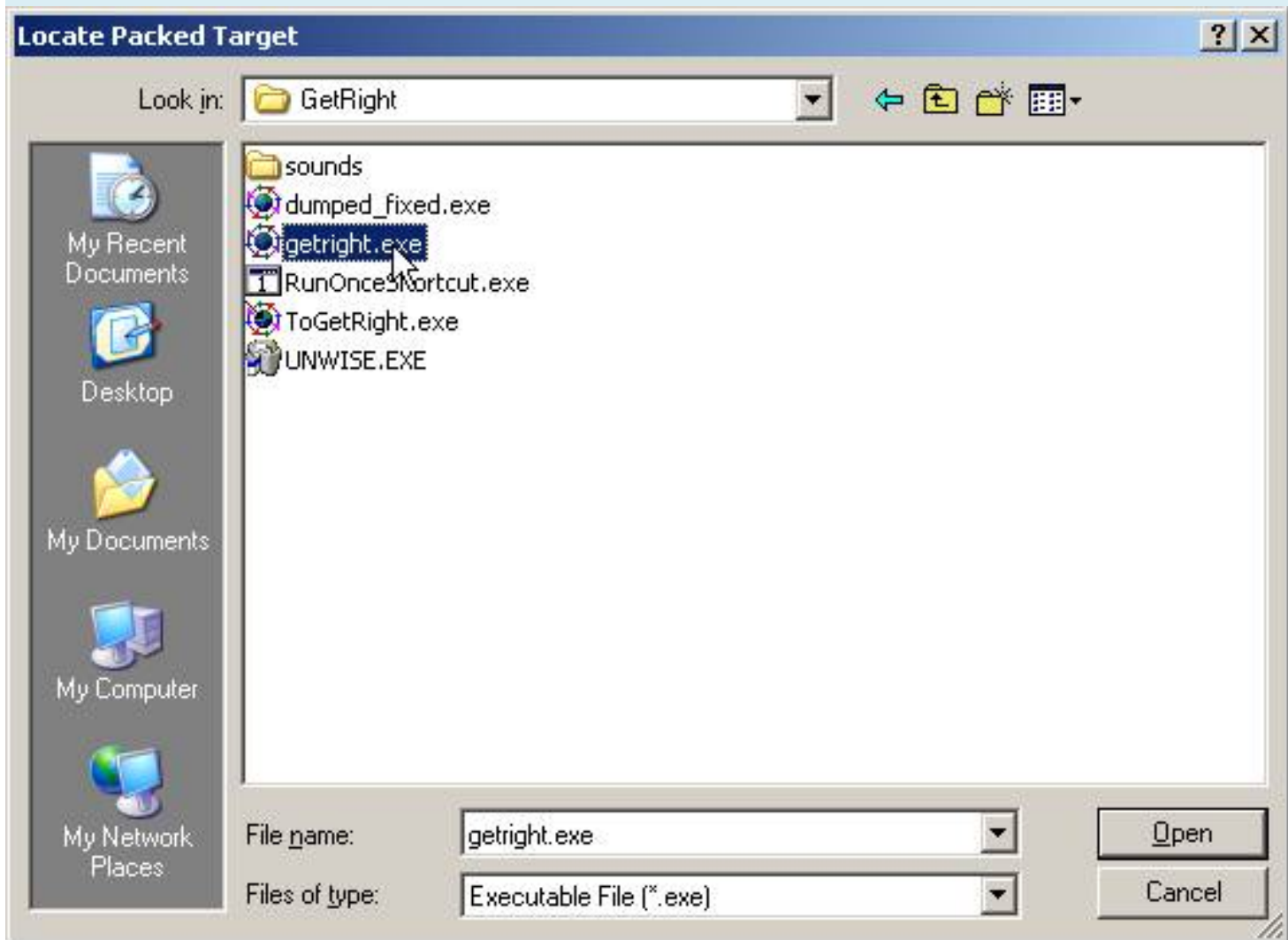
0012FFC4	7C816D4F	RETURN to kernel32
0012FFC8	7C910738	ntdll.7C910738
0012FFCC	FFFFFFFF	
0012FFD0	7FFD0000	
0012FFD4	8054B038	
0012FFD8	0012FFC8	
0012FFDC	851A47F8	
0012FFE0	FFFFFFFF	End of SEH chain
0012FFE4	7C9399F3	SE handler
0012FFE8	7C816D50	kernel32.7C816D50

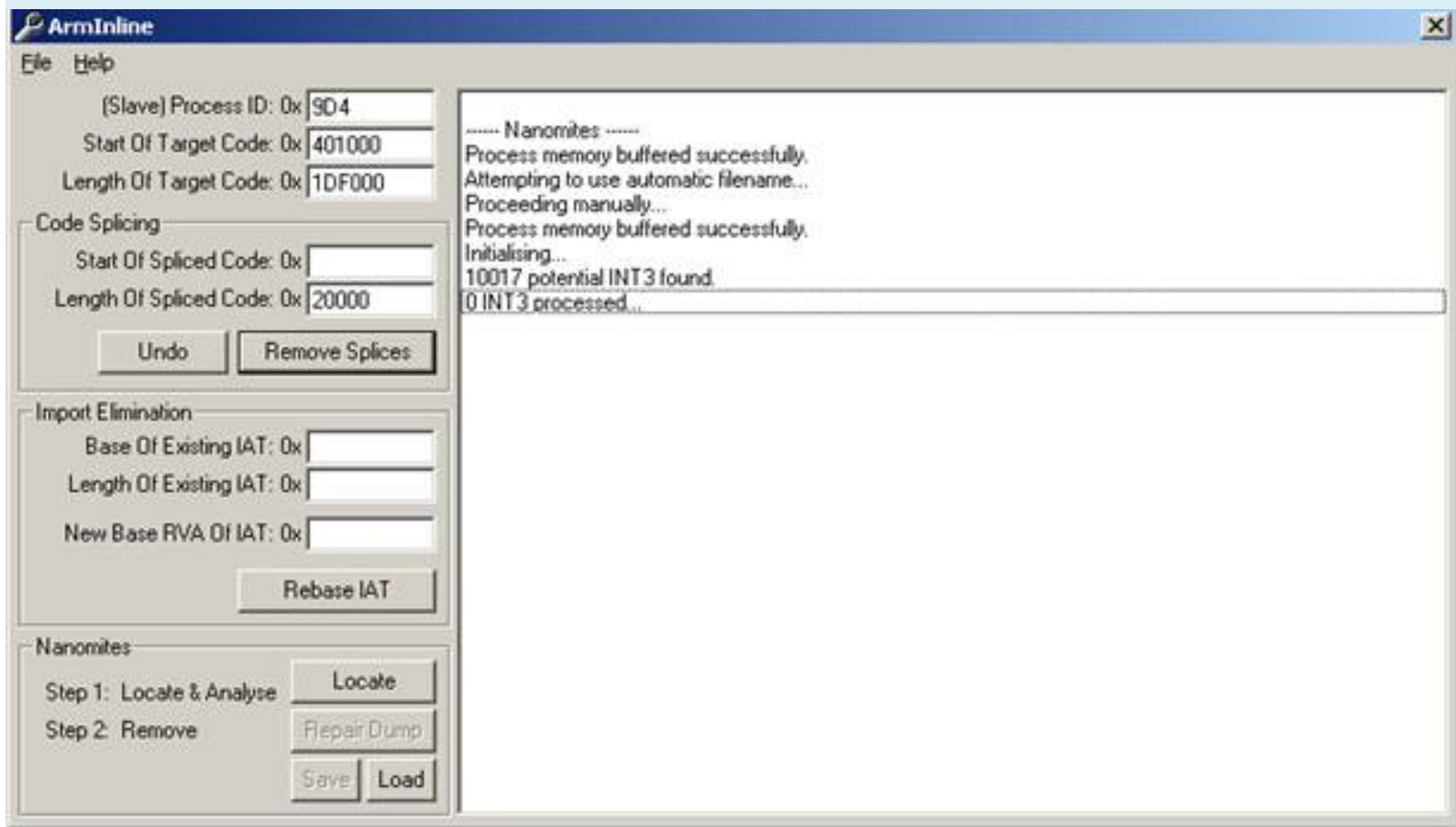
Command :

Program entry point

Memory map						
Address	Size	Owner	Section	Contains	Type	Access
003C0000	00008000				Priv	RW
003D0000	00001000				Priv	RW
003E0000	00001000				Priv	RW
003F0000	00002000				Map	R
00400000	00001000	dumped_f		PE header	Imag	R
00401000	0010F000	dumped_f	.text		Imag	R
005E0000	0005C000	dumped_f	.rdata	export	Imag	R
0063C000	0001B000	dumped_f	.data	data	Imag	R
00657000	00050000	dumped_f	.text1	code	Imag	R
006A7000	00010000	dumped_f	.adata		Imag	R
006B7000	00020000	dumped_f	.data1		Imag	R

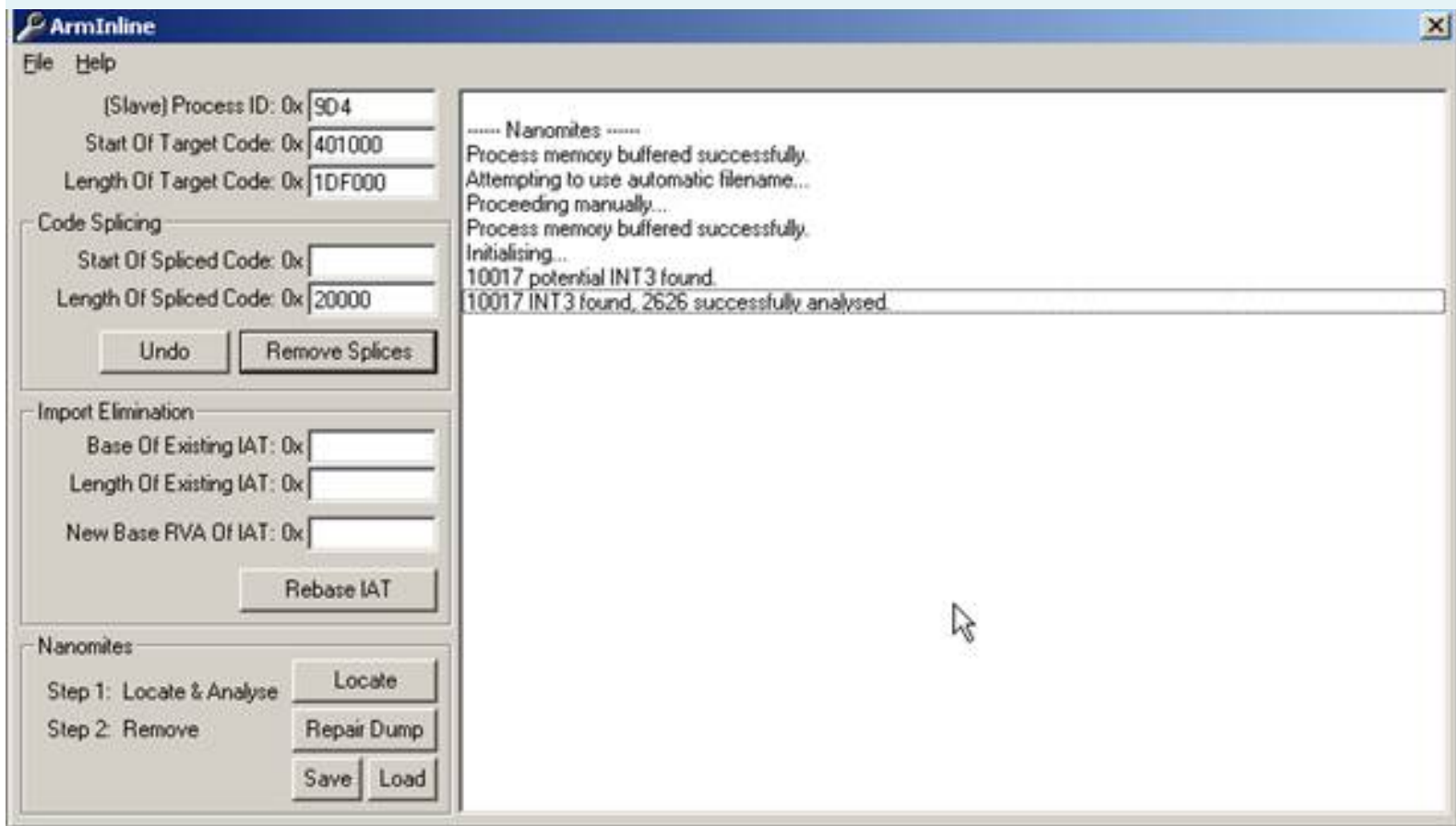
_Mo ArmInline up, enter the number you need. Click locate, and then close the file to the GetRight ArmInline load up, load packed target:



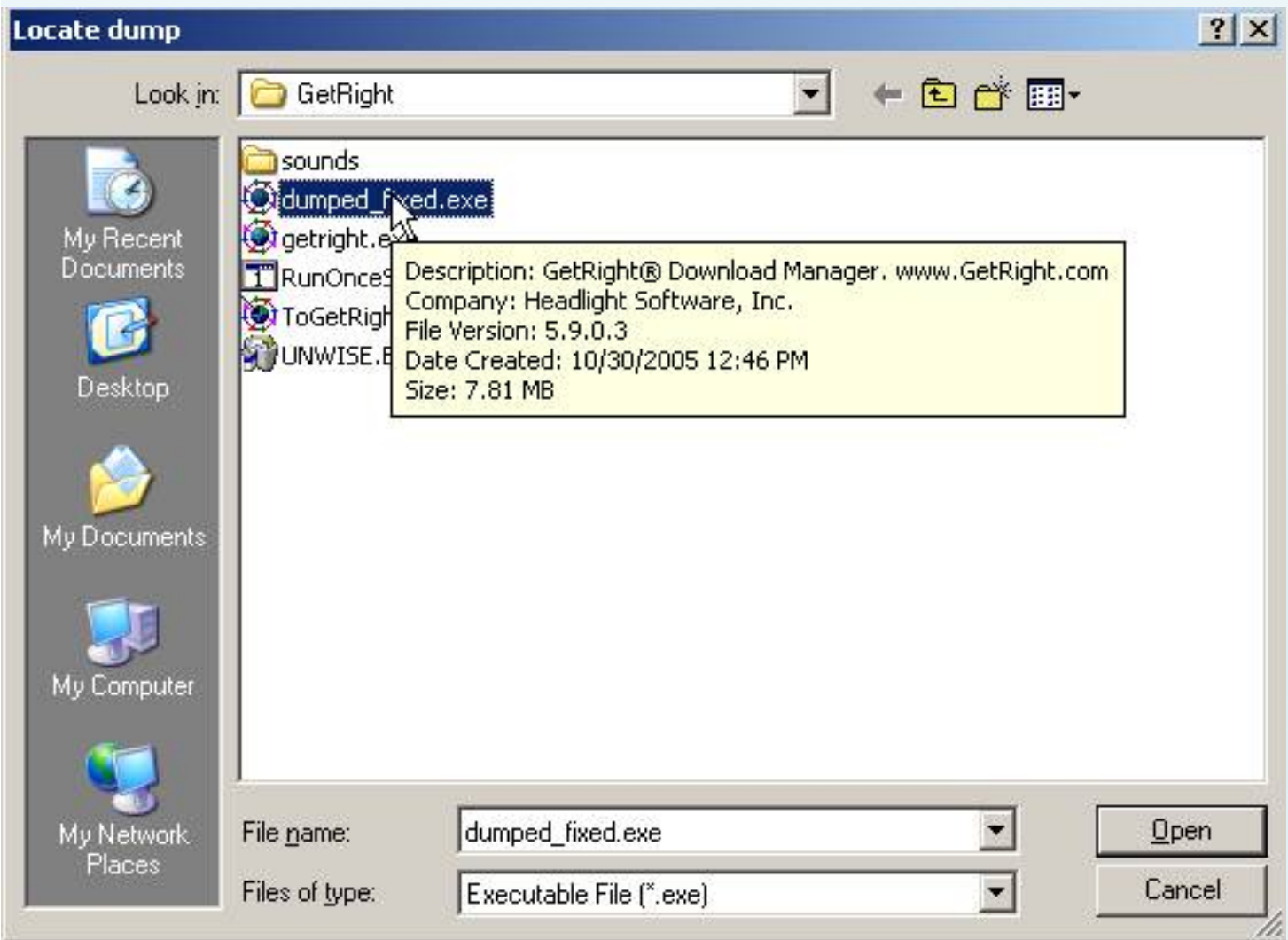


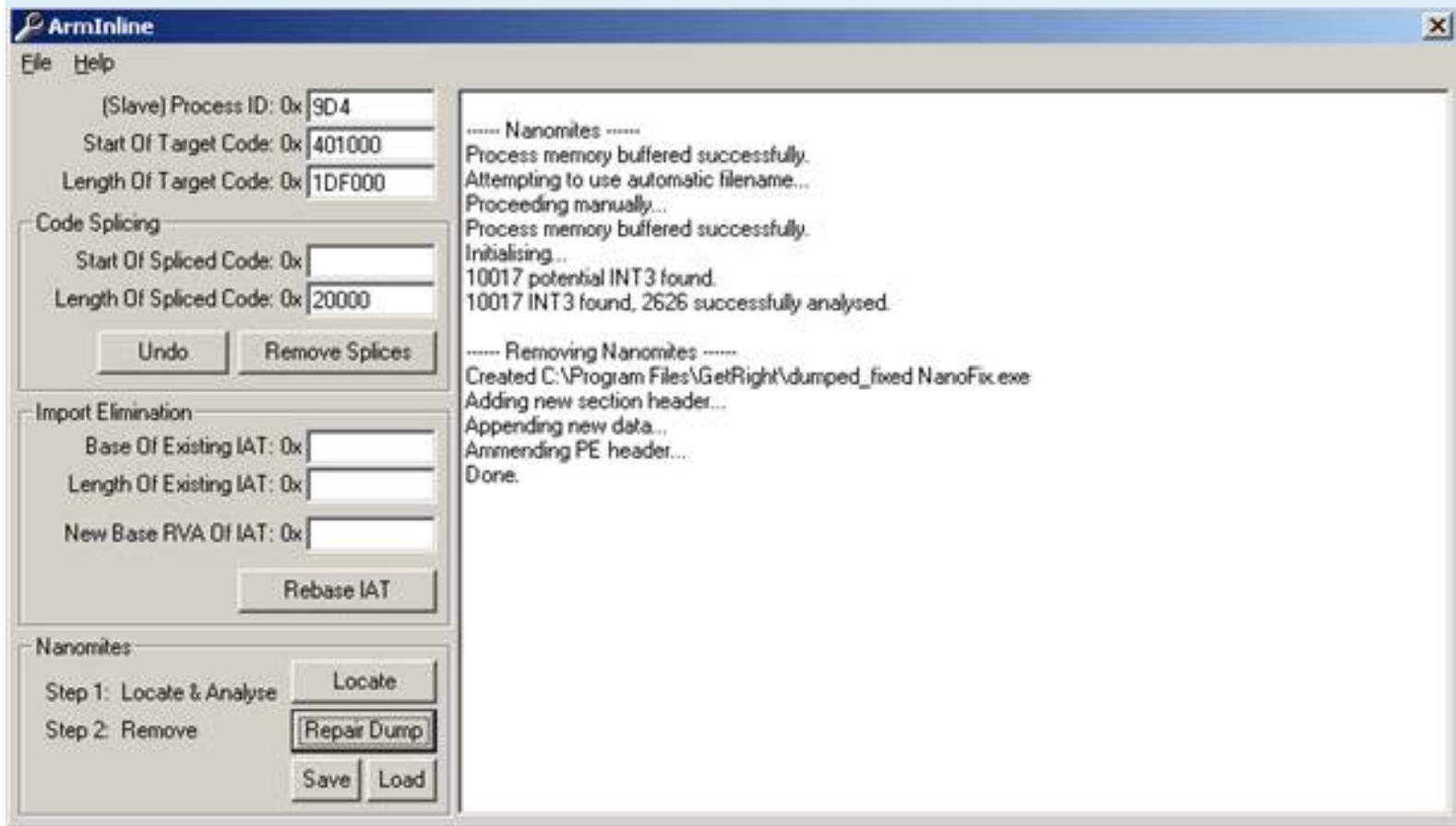
GetRight _Lai close again.



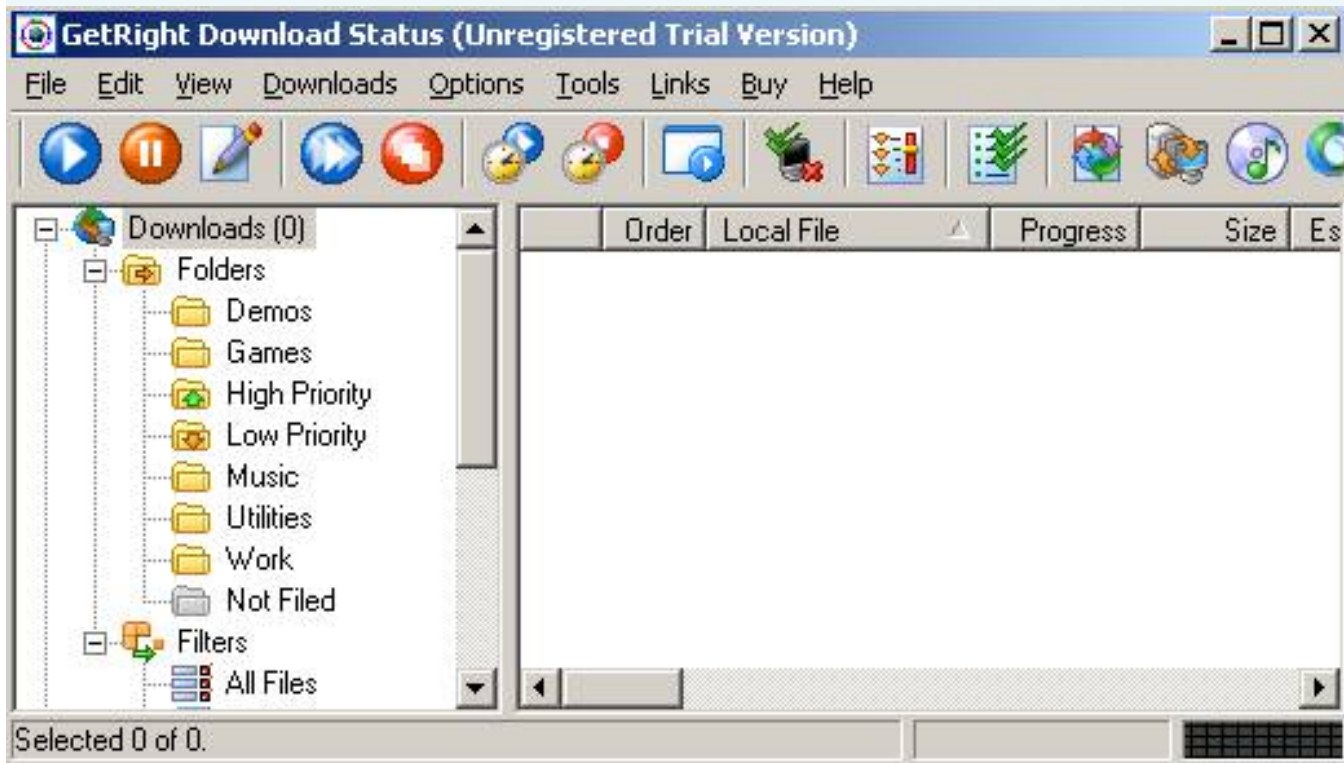


_Repair Dump:





_Run Dumped_fixed NanoFix.exe:



_Unpacked Successful!

IV. Conclusion

_For More tuts, please visit <http://tinicat.de/hacnho> or <http://hacnho.exetools.com>

_Bye!

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtn, hosiminh, Nilrem, fly, MaDMAn_H3rCul3s, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light, iamidiot,

WhyNotBar, trickyboy, Merc ... and you!

Special Thanx Cracks Latinos.

Thanx OllyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 25/10/2005)

How to unpack AHTeam EP Protector 0.3 Private Special Build

by dqtn from Phudu Cracker Team
Vietnam 2004
<http://www.phudu.com>

Victim: unpackme10 packed with 3.1 PEncrypt Final, **AHTeam EP Protector 0.3 Private Special Build**

Homepage: <http://www.tothesky.us>

Tools: 1:10 OllyDbg, PEiD 0.92, LordPE Deluxe, 0.3 unPEncrypt by snaker [UG2002]

Unpack the file: unpackme10.exe

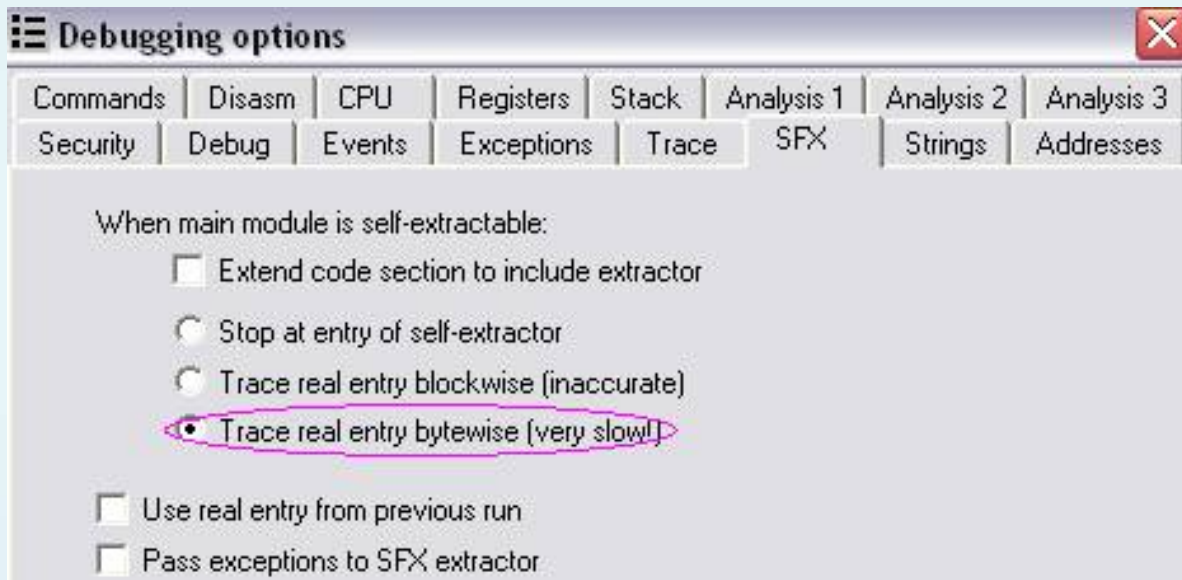
Unpacked by dqtn

PEiD used to check, receive and Nothing found OEP = 401029 ... PEiD because it does not know not to believe that it OEP found ... unpackme load up in OllyDbg will stop at the Entry Point EP = 405417 ... there are many ways to recognize the program is protected by AHTeam EP Protector ... EP is now in bytes of 90, where it commands a xor bytes, as soon as the command xor loop back to the command xor

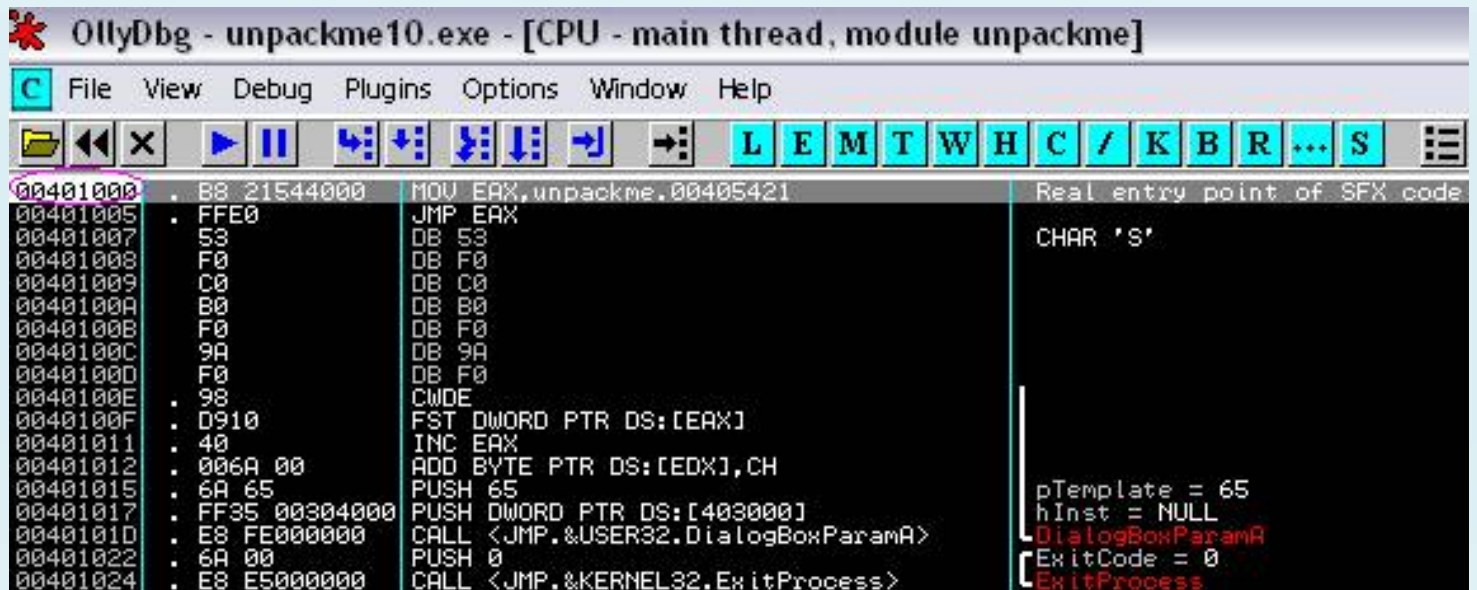
```

OllyDbg - unpackme10.exe - [CPU - main thread, module unpackme]
File View Debug Plugins Options Window Help
[Icons] [L] [E] [M] [T] [W] [H] [C] [/] [K] [B] [R] [...] [S]
00405400 B8 FF0F4000 MOV EAX,unpackme.00400FFF
00405405 B9 10000000 MOV ECX,10
0040540A 803408 F0 XOR BYTE PTR DS:[EAX+ECX],0F0
0040540E ^E2 FA LOOPD SHORT unpackme.0040540A
00405410 B8 00104000 MOV EAX,unpackme.00401000
00405415 FFE0 JMP EAX
00405417 90 NOP
00405418 90 NOP
00405419 90 NOP
0040541A B8 00104000 MOV EAX,unpackme.00401000
0040541F FFE0 JMP EAX
00405421 BA 00104000 MOV EDX,unpackme.00401000
00405426 90 NOP
00405427 90 NOP
00405428 90 NOP
00405429 90 NOP
0040542A B8 9AF018FD MOV EAX,FD18F09A
0040542F 8902 MOV DWORD PTR DS:[EDX],EAX
00405431 83C2 04 ADD EDX,4
00405434 B8 F1F0F053 MOV EAX,53F0F0F1
00405439 8902 MOV DWORD PTR DS:[EDX],EAX
0040543B B8 00544000 MOV EAX,unpackme.00405400
00405440 90 NOP
00405441 90 NOP
00405442 90 NOP
00405443 90 NOP
00405444 90 NOP
00405445 90 NOP
00405446 90 NOP
00405447 FFE0 JMP EAX
  
```

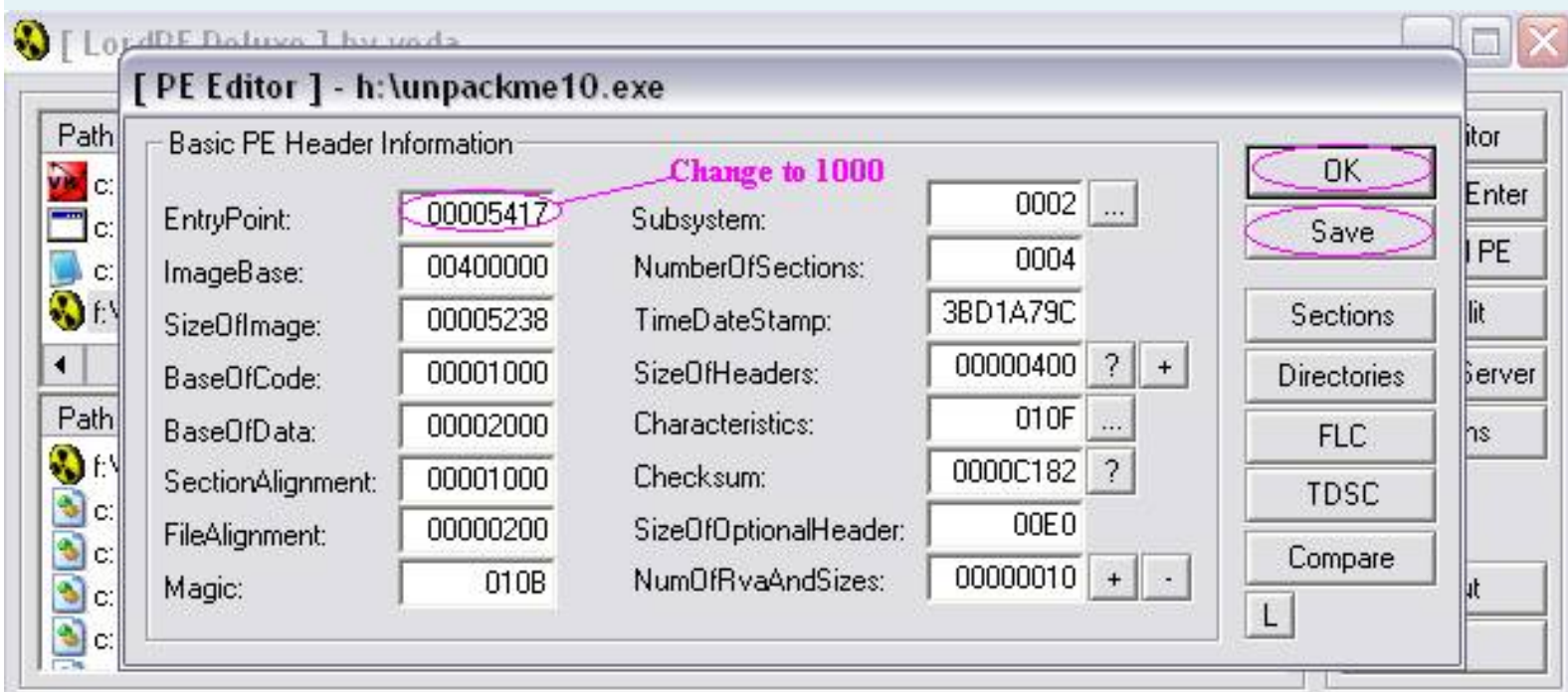
configuration as OllyDbg following



load back in OllyDbg unpackme we will stop at OEP = 401000



here if you dump, fix IAT ... dumped_.exe click the file, nothing happens all ... use PEiD view the file dumped_.exe know pack PEncrypt 3.1 Final -> junkcode ... lol, unpack fail if OEP find itself by setting breakpoint, and then to 401,000 ... by trace down through some of the 401,002 now in the resolution refers to the code call kernel32.GetModuleHandleA ... dump, fix IAT, dumped_.exe click, nothing happens ... use PEiD view the file dumped_.exe TASM32/MASM32 know ... lol, we unpack fail notice that the code started from the 401,000 PEiD entry point to know the program is compressed using PEncrypt 3.1 Deluxe Final LordPE use change Entry Point to 401,000, click Save, click OK



unPEnCrypt used by 0.3 snaker [UG2002] unpackme10 to unpack the file ... new file to run well, PEiD TASM32/MASM32 said ... thành AHTeamEPProtector also often only, or in that it is encrypted by EP packer, then encryption is always OEP of the program if the packer in its list

.....

Victim: Minesweeper 5.1 packed with ASPack 2:12, EP AHTeam private Protector 0.3 Special Build

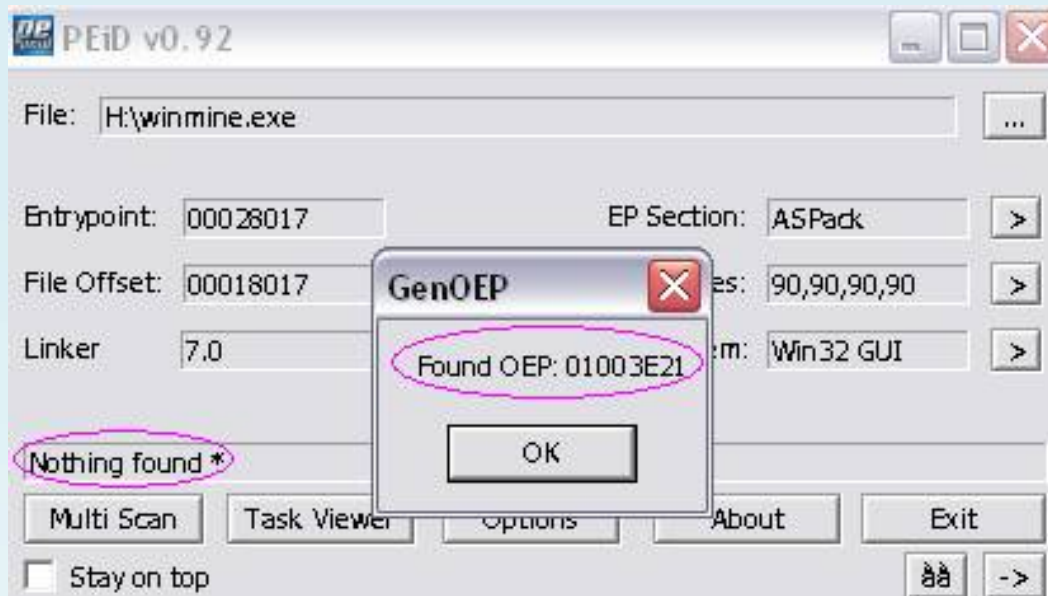
Homepage: Copyright © 1981-2001 Microsoft Corporation by Robert Donner and Curt Johnson

Tools: 1:10 OllyDbg, PEiD 0.92, LordPE Deluxe, Import REConstructor 1.6 Final

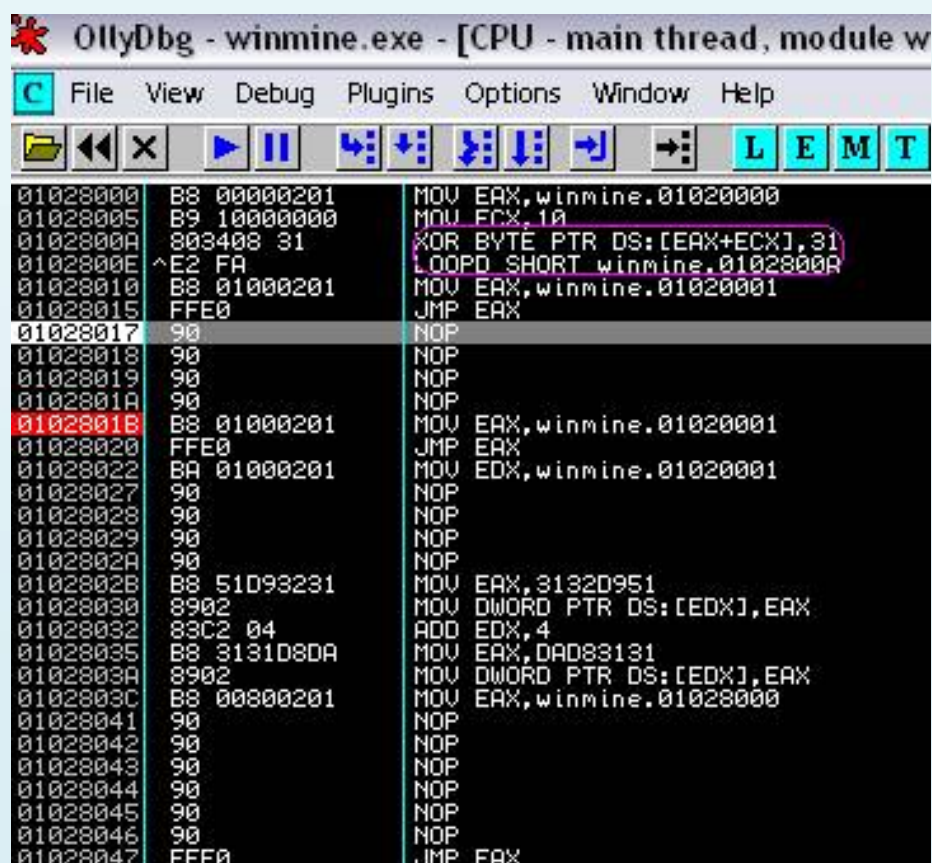
Unpack the file: winmine.exe

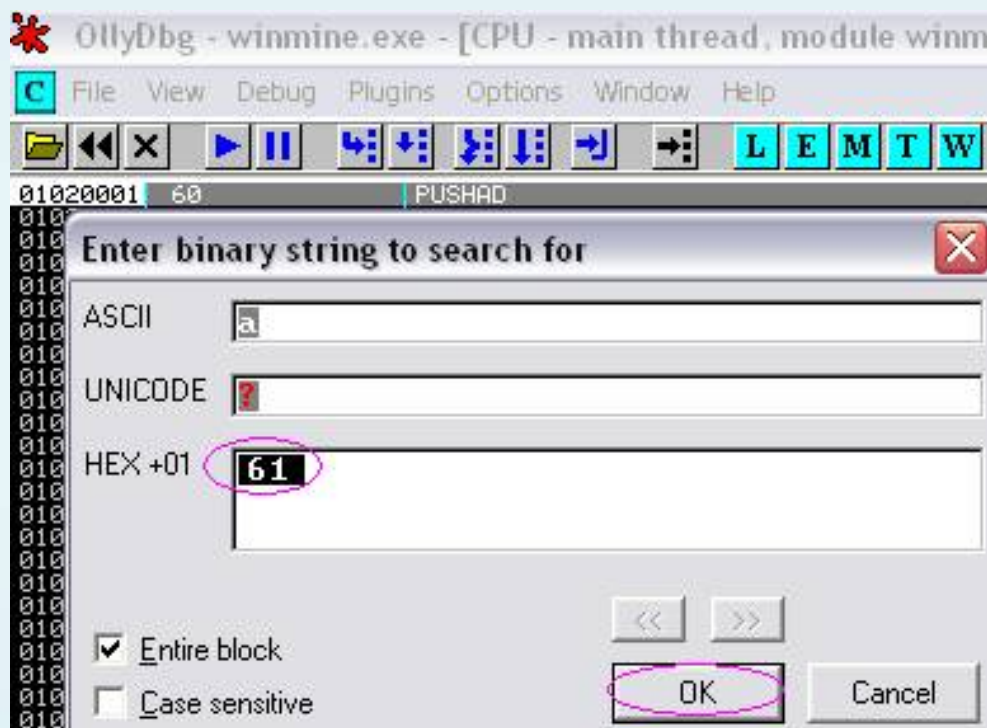
Unpacked by dqtn

WinXP SP1 winmine.exe file is compressed using ASPack dqtn 2:12 and then used to AHTeam EP Protector 0.3 Private Special Build ... PEiD for OEP = 10003E21

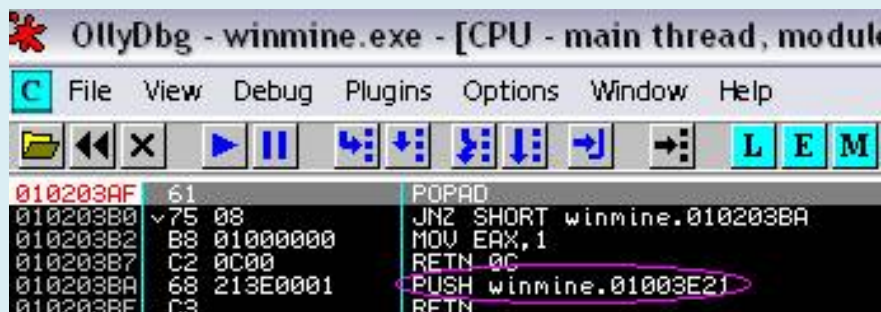


OllyDbg winmine load in, we stopped at the Entry Point 1028017 ... set breakpoint at 102801B, F9 to run, to step over PUSHAD at 1,020,001 ... press Ctrl + B to enter 61, Ctrl + L for POPAD

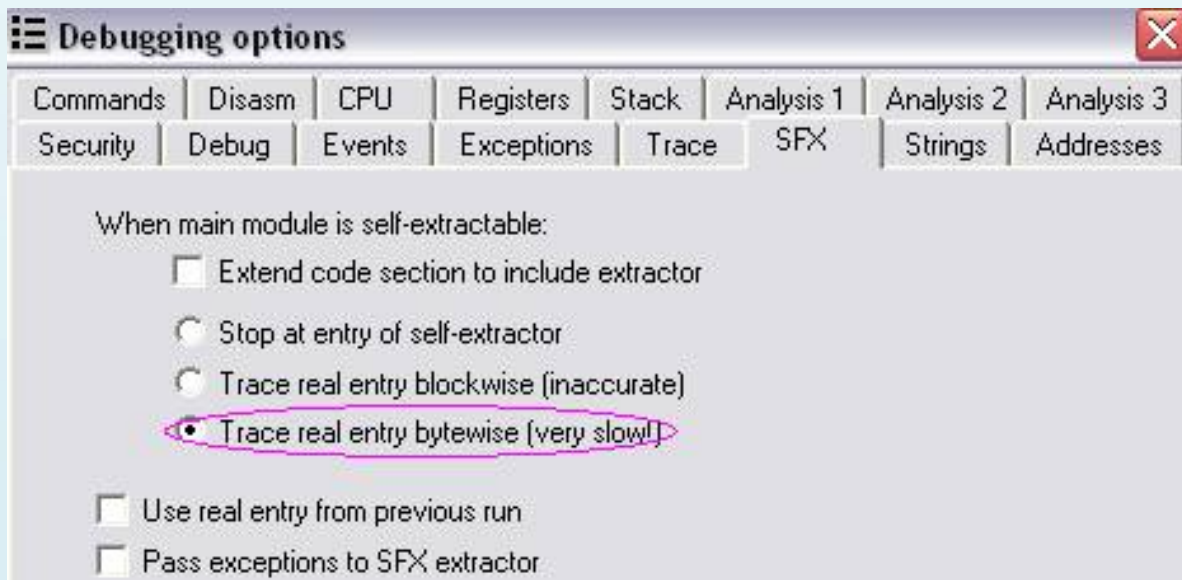




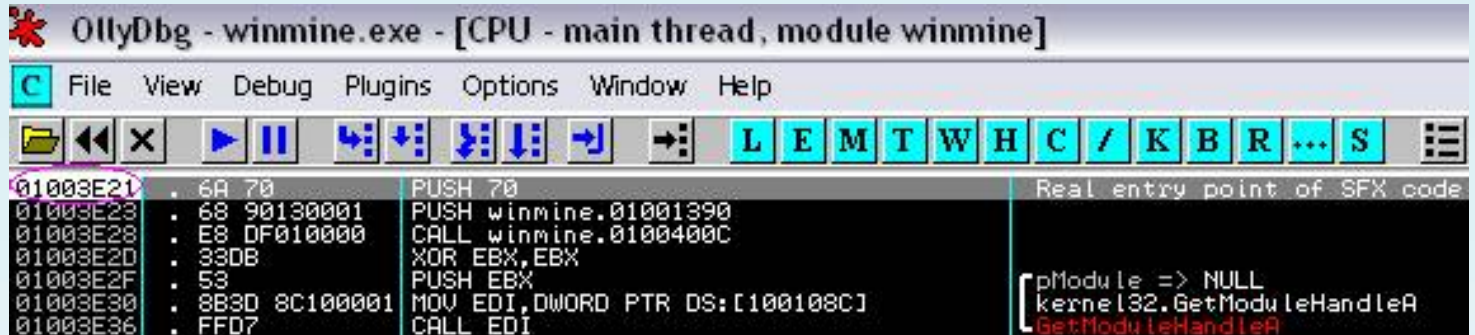
set breakpoint at POPAD in 10203AF, press F9 to run, over the last step we will RETN to OEP in 1003E21



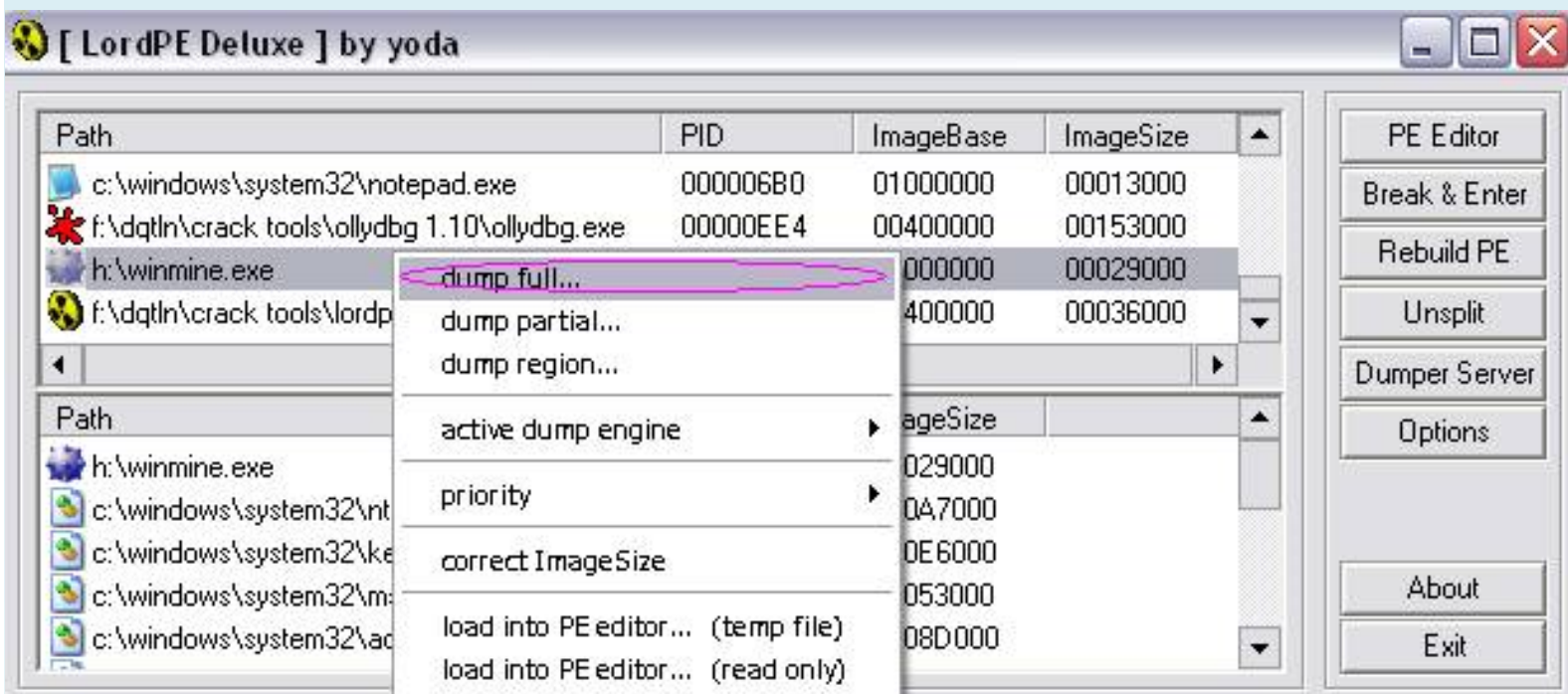
configuration as OllyDbg following



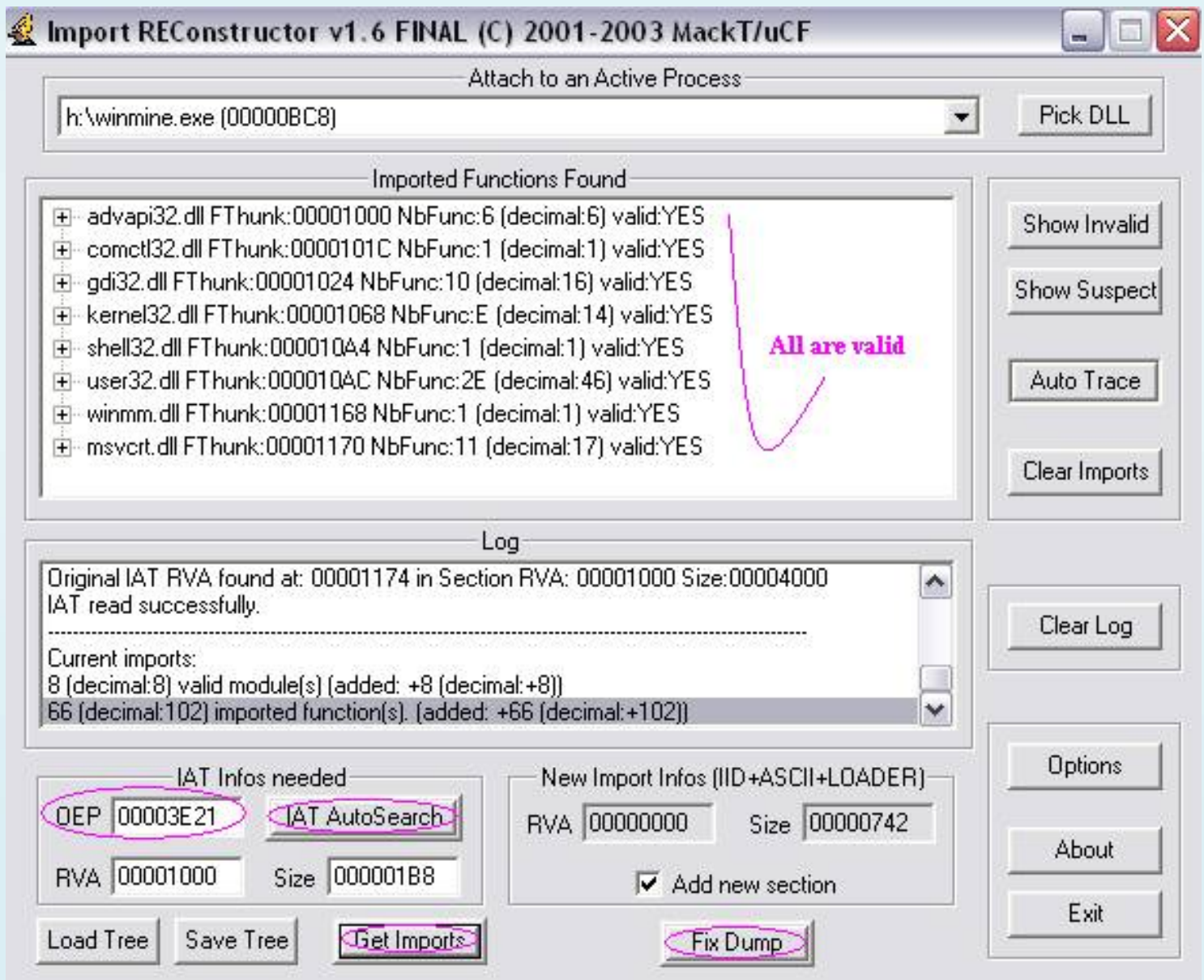
winmine load in OllyDbg and wait a bit to the right will OEP = 1003E21



generally have more ways to find out ... OEP OEP at LordPE Deluxe is used to create files dumped.
exe



we need to use Import REConstructor to fix errors ... changes OEP = 3E21 ... IAT AutoSearch click, click Get more Imports and finally click Fix dump, select File dumped.exe to create the file dumped_.exe



create a new file to run well, can use LordPE Deluxe press rebuild PE to reduce the file size ... italy do chang as ASPack unpack ... problem for the packer in not thành AHTeamEPProtector ... so through this tutorial we learn how to unpack AHTeamEPProtector, a program change or quite EP

greetings

**If you have questions, Remarks about this tutorial, mail me
dqtlm@phudu.com**

How to unpack AntiCrack Protector

by dqtlN from Phudu Team

Vietnam 2005

www.phudu.com

Victim: Cool Color Picker 1.1 packed with **AntiCrack Protector 1.0x**

www.color-picker.net

Homepage:

Tools: 1:10 OllyDbg, PEiD 0.93, LordPE Deluxe, Final Import REConstructor 1.6, 1.4

IsDebuggerPresent OllyDbgPlugin

Unpack the file: ColorPicker.exe

10h57 AM Thursday 17 February 2005

PEiD know we use the compression with **AntiCrack Protector 1.0x -> RISCO Software Inc..** ,

Click Generic OEP Finder exit PEiD always, lol

load the program in OllyDbg, press F9 to run, press Shift + F9 a few times we get "Process terminated, exit code 80", the program does not run anymore ... load the program in OllyDbg, select Plugins / IsDebugPresent / Hide, press F9 to run, press Alt + M to enable Memory map window, set breakpoint ... perform a basic task for the OEP OllyDbg always out, lol, months where it is more anti than it is to use CreateToolhelp32Snapshot API to do this, see the MSDN for more details ... we need to do is not to use the API CreateToolhelp32Snapshot configuration OllyDbg: Options / debugging options/Exceptions/INT3 breaks ... load the program in OllyDbg, right click / Search for / name in all modules, find CreateToolhelp32Snapshot API, double-click on it, we found as follows

```
77E92ED1> 55 PUSH
EBP
77E92ED2 8BEC MOV
EBP, ESP
..... here
are some
lines .....
77E92F33 8BF0 MOV
ESI, EAX
77E92F35 85F6
TEST ESI, ESI
77E92F37 0F8C
F4E50000 JL
kernel32.77EA1531
77E92F3D 8B45 0C
MOV EAX, DWORD
PTR SS: [EBP + C]
77E92F40 5e POP
ESI
77E92F41 C9 LEAVE
```

```
77E92F42 C2 0800
RETN 8
```

we need to kill this API, the following changes

```
77E92ED1>
55 PUSH
EBP
77E92ED2
5D POP
EBP
77E92ED3
33C0 XOR
EAX, EAX
77E92ED5
C2 0800
RETN 8
```

select Plugins / IsDebugPresent / Hide, press F9 to run, press Alt + M to enable Memory map window ... Search with the Owner is ColorPic, Contains the code, right-click to select Set memory breakpoint on access ... close window again OllyDbg, press Shift + F9 one time only one day to OEP = 41A4C9

```
0041A4C9 68 28F34300 PUSH ColorPic.0043F328 => OEP
0041A4CE 68 58DC4100 PUSH ColorPic.0041DC58
..... here are some lines .....
0041A4E7 8965 E8 MOV DWORD PTR SS: [EBP-18], ESP
0041A4EA FF15 50B24300 CALL DWORD PTR DS: [43B250]; kernel32.GetVersion
```

IAT dump & fix, it is easy

dqtl write this article by article AntiCrack Protector 1.0x manually unpack tutorial by KaGra thanks KaGra

greetings

If you have questions, Remarks about this tutorial, mail me

dqtlncrk@gmail.com

How to unpack ASPack

by dqtn from Phudu Cracker Team
Vietnam 2004
<http://www.phudu.com>

Victim: Minesweeper 5.1 packed with ASPack 2:12

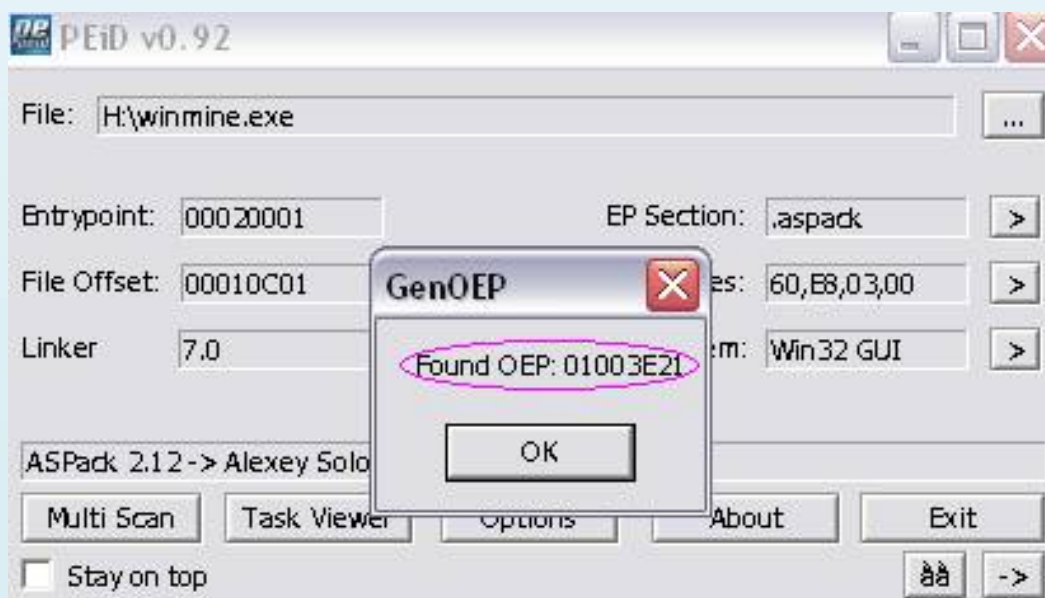
Homepage: Copyright © 1981-2001 Microsoft Corporation by Robert Donner and Curt Johnson

Tools: 1:10 OllyDbg, PEiD 0.92, LordPE Deluxe, Import REConstructor 1.6 Final

Unpack the file: winmine.exe

Unpacked by dqtn

Previous dqtn has a lot of tutorials unpack but has not attached, make it difficult to understand for beginners ... the time is now free to write back dqtn all accompanied by pictures and programs have been compressed you to easily practice ... mine-clearing game WindowXP SP1 was dqtn itself again with ASPack 2:12 -> Alexey Solodovnikov ... to unpack it is very easy, as appropriate similar unpack UPX is Okie ... PEiD does see some information, we know OEP = 01003E21



load up the program in OllyDbg receive compressed code? , Select it ... we stopped at ENTRY POINT at 1020001


```

OllyDbg - winmine.exe - [CPU - main thread, module w
File View Debug Plugins Options Window Help
[Icons] [L] [E] [M] [T]
01020011 60          PUSHAD
01020012 E8 03000000 CALL winmine.0102000A
01020017 -E9 EB045045 JMP 465F04F7
0102001C 55          PUSH EBP
0102001D C3          RETN
0102001E E8 01000000 CALL winmine.01020014
01020013 vEB 5D      JMP SHORT winmine.01020072
01020015 BB EDFFFFFF MOV EBX,-13
0102001A 0300      ADD EBX,EBP

```

move down the order to find POPAD, CtrlB enter 61 for faster ... set break point at which, F9 to run, trace down through the command will RETN to OEP

```

OllyDbg - winmine.exe - [CPU - main thread, module w
File View Debug Plugins Options Window Help
[Icons] [L] [E] [M] [T]
0102039A B8 213E0000 MOV EAX,3E21
0102039F 50          PUSH EAX
010203A0 0385 22040000 ADD EAX,DWORD PTR SS:[EBP+422]
010203A6 59          POP ECX
010203A7 0BC9      OR ECX,ECX
010203A9 8985 A8030000 MOV DWORD PTR SS:[EBP+3A8],EAX
010203AF 61          POPAD
010203B0 v75 08      JNZ SHORT winmine.010203BA
010203B2 B8 01000000 MOV EAX,1
010203B7 C2 0C00      RETN 0C
010203BA 68 00000000 PUSH 0
010203BF C3          RETN

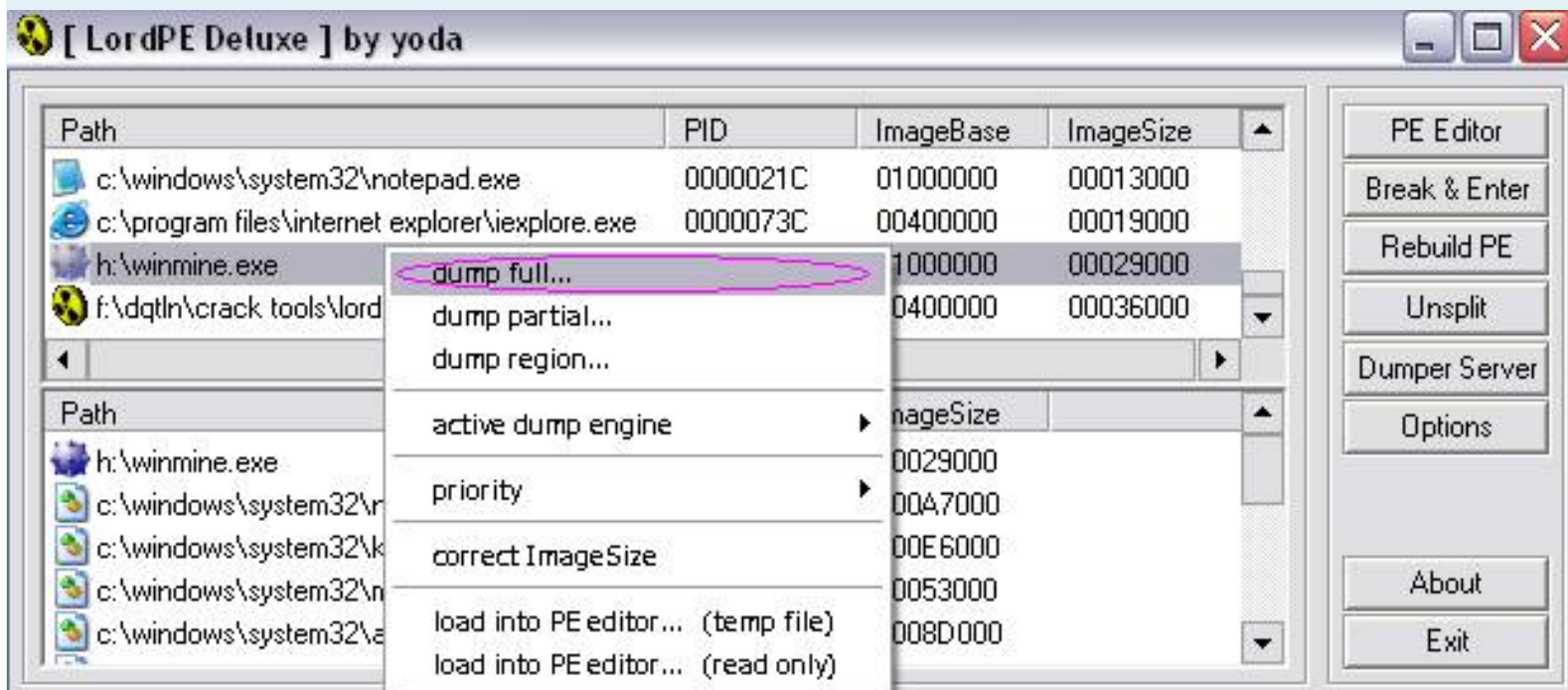
```

```

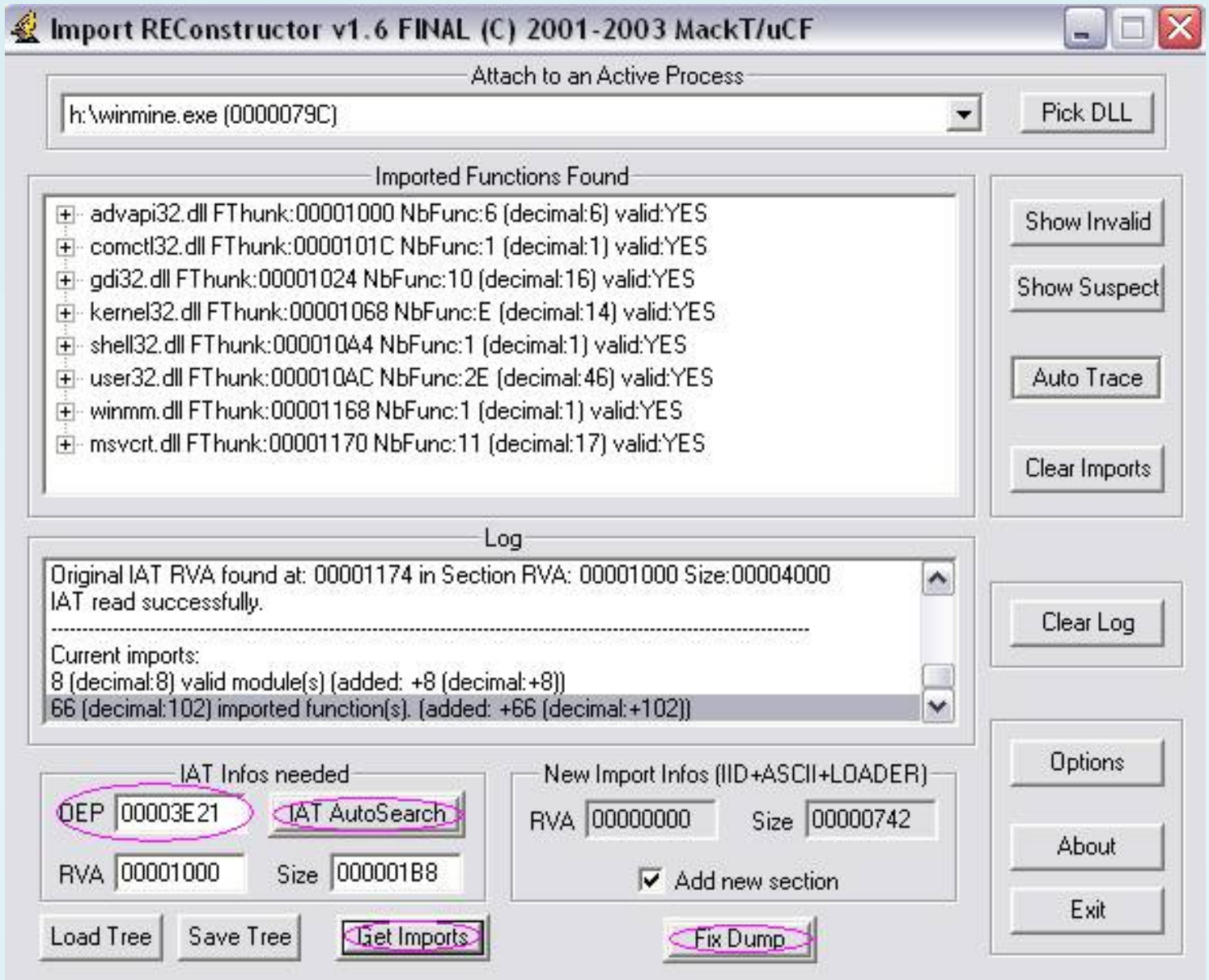
OllyDbg - winmine.exe - [CPU - main thread, module w
File View Debug Plugins Options Window Help
[Icons] [L] [E] [M] [T]
01003E21 6A 70      PUSH 70
01003E23 68 90130001 PUSH winmine.01001390
01003E28 E8 DF010000 CALL winmine.0100400C
01003E2D 33DB      XOR EBX,EBX
01003E2F 53          PUSH EBX
01003E30 8B3D 8C100001 MOV EDI,DWORD PTR DS:[100108C]
01003E36 FFD7      CALL EDI
01003E38 66:8138 4D5A CMP WORD PTR DS:[EAX],5A4D
01003E3D v75 1F      JNZ SHORT winmine.01003E5E

```

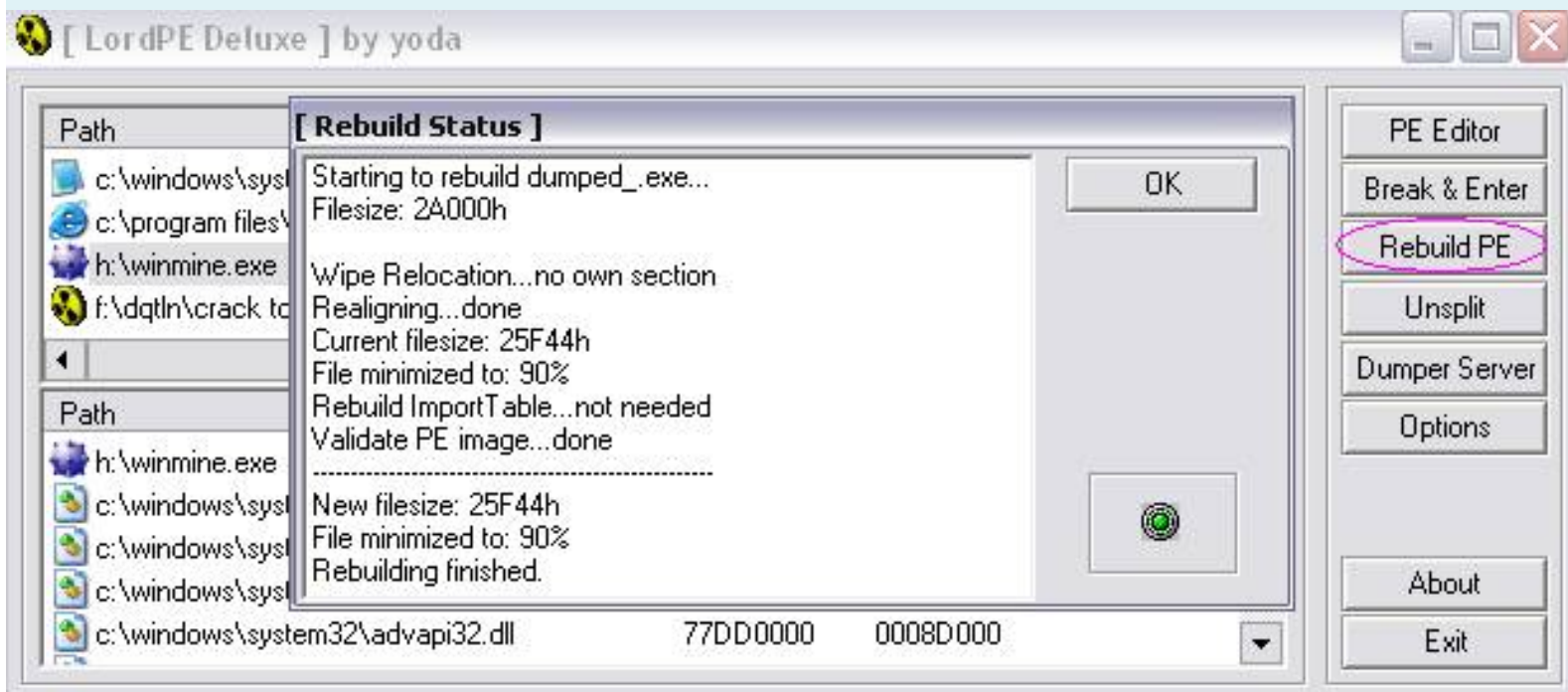
we find the OEP = 1003E21 same PEiD ... here used to LordPE Deluxe memory dump file created as dumped.exe following



run file dumped.exe drilling because it will crash ... we need to use Import REConstructor to fix errors, usually a fix Import API's Table ... changes OEP = 3E21 ... IAT AutoSearch click, click Get more Imports and finally click Fix dump, select File dumped.exe to create the file dumped_.exe



test file dumped_.exe see good work ... normal after the dump file typically has the code of garbage packer, we can use Deluxe LordPE to reduce the size of the file



so we have learned how to unpack ASPack finished ... This way you can unpack all ASPack version of the current ... future is not known because employers write ASPack has ASProtect quite formidable that the

greetings

**If you have questions, Remarks about this tutorial, mail me
dqtln@phudu.com**

How to unpack ASProtect by dqtn from Phudu Team Vietnam 2004 <http://www.phudu.com>

Victim: Advanced RAR Password Recovery 1:50 build packed with 7 **ASProtect 1:22 to 1:23 Beta 21 -> Alexey Solodovnikov**

<http://www.elcomsoft.com/arpr.html>

Homepage: <http://www.aspack.com>

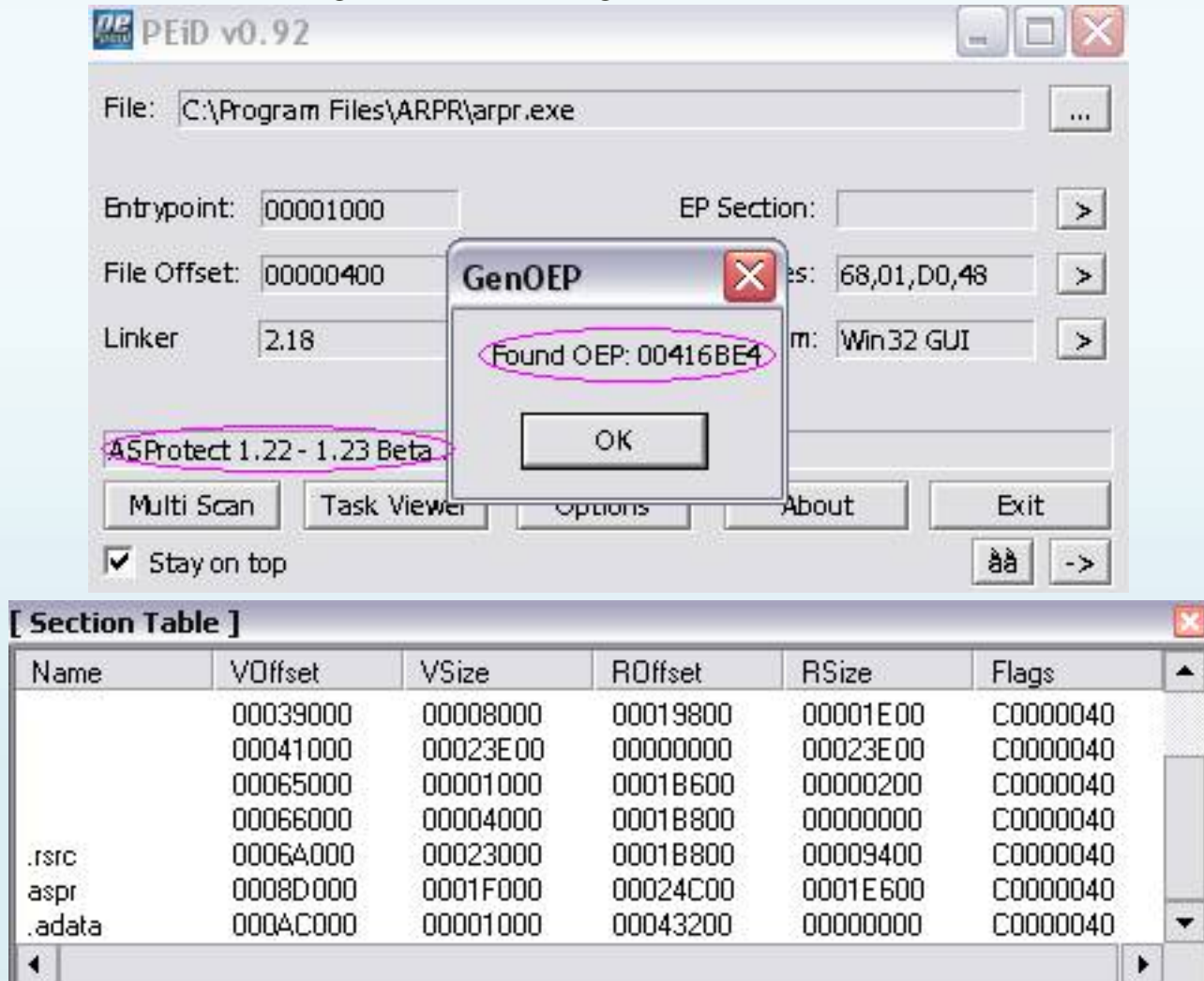
Tools: 1:10 OllyDbg, PEiD 0.92, LordPE Deluxe, Import REConstructor 1.6 Final

Unpack the file: arpr.exe

Unpacked by dqtn

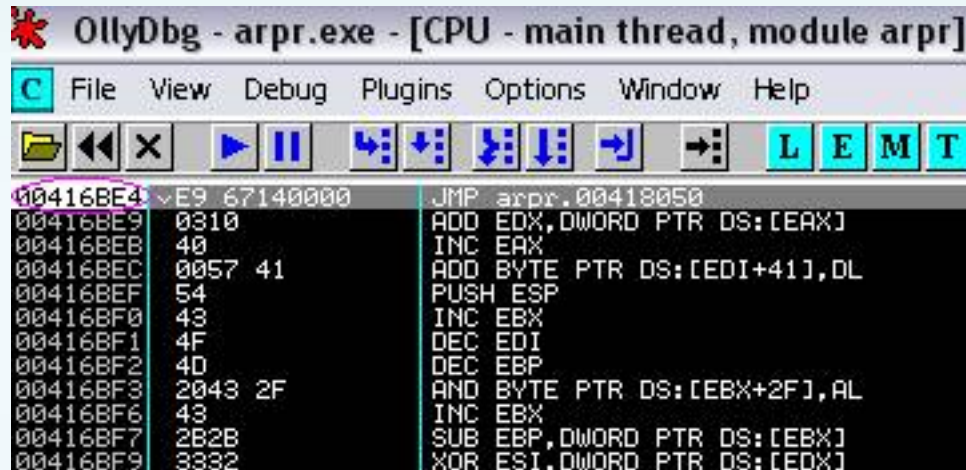
12h40 am Thursday December 30 2004

PEiD we use programs that are compressed using ASProtect 1:22 to 1:23 Beta 21, OEP = 47B920
LordPE and Deluxe for the Flags of the following



load the program in OllyDbg, press the Shift + F9 until the program runs ... load it in OllyDbg, press Shift + F9 to-1 times before running the program, press Alt + M to enable Memory map window ... Search with the Owner is arpr, Contains the code, right-click to select Set memory breakpoint on access ... close window again OllyDbg, press Shift + F7, press F9, we stopped at

the OEP = 416BE4 ... here is the need to find OEP, above all do not have 00 bytes ... this month without Stolen Bytes



here used LordPE Deluxe memory dump file created dumped.exe ... Import using IAT REConstructor fix, but most will not file to run as good as we expected ... may file dumped_.exe run but click on the Register, the program will error ... we need to find accurate RVA and Size in OllyDbg, we are stopped at the OEP = 416BE4, press Ctrl + B to enter FF 25, press Ctrl + L several times we came here

find JMP DWORD PTR DS: [smallest] and JMP DWORD PTR DS: [most] ... RVA think the start and Size

RVA = 3751C, Size = 437964 = 448-43751c

IAT fix the following



greetings

**If you have questions, Remarks about this tutorial, mail me
dqtlncrk@gmail.com**

How to unpack Asprotect 1:23 rc4 series # 1

Name soft: Bad Copy Pro ver 3.72

Cracker: Computer_Angel

Level: average

I. Tool needed:

- _ Ollydbg (debugger tool)
- _ Lord-Pe (tool to dump file)
- _ Imprec 1.6

II. Cach initiate:

Find OEP:

1. Olly Badcopy.exe to load, the shift-F9 to run the program, reality will automatically be stopped again by the point by exception asprotect generated. The number of night-shift F9 from the original time to time for the message that xxx program run entirely. Xxx number that I received record here is 27, ie 27 times after the shift-F9 test program will run.
2. Load up Indeed, the shift-F9 exactly 26 times, then press Shift F7. Press Alt-M to open the window on the **Memory**. Earn memory segment is available Owner "Badcopy" and contains the "codes". Set **on break point** at which **memory access**.
3. Close window memory again, continue to return to the CPU window, press F9, we stop at the following code:

```
00406EE0-FF25 78B2B700 JMP DWORD PTR DS: [B7B278] <----- here
00406EE6 8BC0 MOV EAX, EAX
00406EE8-FF25 74B2B700 JMP DWORD PTR DS: [B7B274]
00406EEE 8BC0 MOV EAX, EAX
00406EF0-FF25 70B2B700 JMP DWORD PTR DS: [B7B270]
00406EF6 8BC0 MOV EAX, EAX
00406EF8-FF25 6CB2B700 JMP DWORD PTR DS: [B7B26C]
```

visible on the top 1 xiu, we see

```
00406ECA E8 09A5FFFF CALL BadCopy.004013D8
00406ECF A3 38705000 MOV DWORD PTR DS: [507038], EAX
00406ED4 E8 F7A4FFFF CALL BadCopy.004013D0
00406ED9 A3 30705000 MOV DWORD PTR DS: [507030], EAX
```

```
00406EDE C3 RETN
00406EDF 90 NOP
```

so sure this can o is OEP ct.Cau question of why so predicted would later I will specify.

Human F8 to jump out of the address **00406EE0**, we again address 15xxxxxxx, then press F9 we will stop at **00406FB5**

```
00406FA7 33C0 XOR EAX, EAX
00406FA9 A3 0C775000 MOV DWORD PTR DS: [50770C], EAX
00406FAE 6A 00 PUSH 0
00406FB0 E8 2BFFFFFF CALL BadCopy.00406EE0
00406FB5 A3 14775000 MOV DWORD PTR DS: [507714], EAX;
BadCopy.00400000 here <-----
00406FBA A1 14775000 MOV EAX, DWORD PTR DS: [507714]
00406FBF A3 B4105000 MOV DWORD PTR DS: [5010B4], EAX
00406FC4 33C0 XOR EAX, EAX
00406FC6 A3 B8105000 MOV DWORD PTR DS: [5010B8], EAX
00406FCB 33C0 XOR EAX, EAX
00406FCD A3 BC105000 MOV DWORD PTR DS: [5010BC], EAX
00406FD2 E8 C1FFFFFF CALL BadCopy.00406F98
00406FD7 THREE B0105000 MOV EDX, BadCopy.005010B0
00406FDC 8BC3 MOV EAX, EBX
00406FDE E8 65D7FFFF CALL BadCopy.00404748
00406FE3 5B POP EBX
00406FE4 C3 RETN
```

similar to above, look up 1 xiu we again confirm that this is o OEP. Use F8 to trace to order **RETN**. When return from our RET at **005008CC**

```
005008CC B0 01 MOV AL, 1 <----- here
005008CE 84C0 TEST AL, AL
005008D0 74 59 JE SHORT BadCopy.0050092B
005008D2 A1 64605000 MOV EAX, DWORD PTR DS: [506064]
005008D7 8B00 MOV EAX, DWORD PTR DS: [EAX]
005008D9 E8 DA0DF7FF CALL BadCopy.004716B8
005008DE A1 64605000 MOV EAX, DWORD PTR DS: [506064]
005008E3 8B00 MOV EAX, DWORD PTR DS: [EAX]
```



```
005008E5 BA 38095000 MOV EDX, BadCopy.00500938; ASCII "BadCopy
Pro"
```

1 xiu look up, we have:

```
005008B3 0000 ADD BYTE PTR DS: [EAX], AL
005008B5 0000 ADD BYTE PTR DS: [EAX], AL
005008B7 003405 50000000 ADD BYTE PTR DS: [EAX +50], DH
005008BE 0000 ADD BYTE PTR DS: [EAX], AL
005008C0 0000 ADD BYTE PTR DS: [EAX], AL
005008C2 0000 ADD BYTE PTR DS: [EAX], AL
005008C4 0000 ADD BYTE PTR DS: [EAX], AL
005008C6 00?
005008C7 E8 D866F0FF CALL BadCopy.00406FA4 <----- OEP
```

OEP is so **5008C7**. Asprotect characteristics is that it will be able to remove 1 of the first byte of the main program and replace it with the value 00, so when you look at the top of the characters have 00 consecutive, I guess that was the exact OEP CT of the original.

Dump file and fix IAT:

1. In the reality that at the one to initiate the dump file, the lordpe, select process and full dump.
2. Continue with Olly, use the Search String Binary, search string of binary "FF 25", we find to be interrupted

```
00401268-FF25 2CB2B700 JMP DWORD PTR DS: [B7B22C]; kernel32.
CreateFileA here <-----
0040126E 8BC0 MOV EAX, EAX
00401270-FF25 28B2B700 JMP DWORD PTR DS: [B7B228]
00401276 8BC0 MOV EAX, EAX
00401278-FF25 24B2B700 JMP DWORD PTR DS: [B7B224]
0040127E 8BC0 MOV EAX, EAX
00401280-FF25 20B2B700 JMP DWORD PTR DS: [B7B220]
00401286 8BC0 MOV EAX, EAX
00401288-FF25 1CB2B700 JMP DWORD PTR DS: [B7B21C]
0040128E 8BC0 MOV EAX, EAX
```

As a result, we predicted the IAT program starts B7Bxxx

3. Running **Imprec.exe** start to recover IAT, selected process is Badcopy,
 RVA enter value = 400000 = B7B000-77B000
 OEP = 400000 = 5008C7-1008C7
 People Get Imports button to load the values in which IAT Imprec find out.

4. Click on the button "Show Invalid", the invalid import function will display in the window functions. Nhan imported mouse to measure the function, select TraceLevel 1
 Imprec will find out the number 1 is the basic API, but still remaining 1 so that the API level Trace Level 1 is not resolved. One roster select individual desire, the right mouse, choose Disassembler / Hex View

With that function when selecting this function, we received notification "Read error" This means that this is o function which Imprec message nhâm, we use the cut thunk (s) to delete it. With the view that function is, we will use the Plugins tracer to identify ham.

This self-compliance on the steps until completely fix all the ham in the table.

(Note, have 1 ham position **0077B3A0** that function tracer in imprec not get out. FreeResource That's ham, but the reason why I know this? You could see the conditions Disassemble the function of this .

```
01551CF0 push ebp
01551CF1 mov ebp, esp
01551CF3 mov eax, [1557E24] / / DWORD value: 00132F68
01551CF9 pop ebp
01551CFA retn 4
```

push on, well done again pop out, and liver ebp = esp stack that balance, and outside the liver only 1 value that is to eax ----> FreeResource ham.)

Once completed, the button "Fix dump", select the file is the file that you just dump out on the steps.

After completing this step, we already have 1 file.exe complete. However, reality can to run need to take the next step.

Find Stolen bytes:

1. Run the file just fix above (dumped_.exe), CT was crash -> program to deliver olly (please call window is (1)), as well as open 1 window program olly half (this is called window (2)), load the file badcopy not to unpack.

2. In the window (2), made the operation similar to the heart OEP, we will stop at locations before OEP xiu 1, which is:

```

00406FB5 A3 14775000 MOV DWORD PTR DS: [507714], EAX;
BadCopy.00400000 here <-----
00406FBA A1 14775000 MOV EAX, DWORD PTR DS: [507714]
00406FBF A3 B4105000 MOV DWORD PTR DS: [5010B4], EAX
00406FC4 33C0 XOR EAX, EAX
00406FC6 A3 B8105000 MOV DWORD PTR DS: [5010B8], EAX
00406FCB 33C0 XOR EAX, EAX
00406FCD A3 BC105000 MOV DWORD PTR DS: [5010BC], EAX
00406FD2 E8 C1FFFFFF CALL BadCopy.00406F98
00406FD7 THREE B0105000 MOV EDX, BadCopy.005010B0
00406FDC 8BC3 MOV EAX, EBX
00406FDE E8 65D7FFFF CALL BadCopy.00404748
00406FE3 5B POP EBX
00406FE4 C3 RETN

```

continue to trace **00406FDC**.

3. Back to the window (1), trace the function first, and keep until the similar in the window (2), at locations **00406FDC**. I noticed the value in (2) EBX time is **50055C**, and in (1) is 0. Look up above the function where we see

```

00406FA4 / $ 53 PUSH EBX
00406FA5 | . 8BD8 MOV EBX, EAX <----- look here
00406FA7 | . 33C0 XOR EAX, EAX
00406FA9 | . A3 0C775000 MOV DWORD PTR DS: [50770C], EAX
00406FAE | . 6A 00 PUSH 0; / pModule = null
00406FB0 | . E8 2BFFFFFF CALL <JMP.&kernel32.GetModuleHandleA> \
GetModuleHandleA
00406FB5 | . A3 14775000 MOV DWORD PTR DS: [507714], EAX
00406FBA | . A1 14775000 MOV EAX, DWORD PTR DS: [507714]
00406FBF | . A3 B4105000 MOV DWORD PTR DS: [5010B4], EAX
00406FC4 | . 33C0 XOR EAX, EAX
00406FC6 | . A3 B8105000 MOV DWORD PTR DS: [5010B8], EAX
00406FCB | . 33C0 XOR EAX, EAX
00406FCD | . A3 BC105000 MOV DWORD PTR DS: [5010BC], EAX
00406FD2 | . E8 C1FFFFFF CALL dumped_.00406F98
00406FD7 | . THREE B0105000 MOV EDX, dumped_.005010B0

```



```

00406FDC | . 8BC3 MOV EAX, EBX
00406FDE | . E8 65D7FFFF CALL dumped_.00404748
00406FE3 | . 5B POP EBX
00406FE4 \. C3 RETN

```

we see in **00406FA5** is **MOV EBX, EAX**, as a result, the initial value is the EAX. Dong window (2) away.

4. Quay back to reality and have fixed dump. (window (1))

```

005008B8 34055000 DD dumped_.00500534
005008BC 0000 ADD BYTE PTR DS: [EAX], AL
005008BE 0000 ADD BYTE PTR DS: [EAX], AL
005008C0 0000 ADD BYTE PTR DS: [EAX], AL
005008C2 0000 ADD BYTE PTR DS: [EAX], AL
005008C4 0000 ADD BYTE PTR DS: [EAX], AL
005008C6 00?
005008C7> E8 D866F0FF CALL dumped_.00406FA4 current <---- OEP
005008CC. B0 01 MOV AL, 1
005008CE. 84C0 TEST AL, AL

```

OEP on the current is 11 bytes 0, asprotect characteristics is to remove the first byte of the program and replace the bytes = 0, then in the process create, asprotect will restore the bytes and implementers no. So we called the byte is lost **stolen bytes**.

The program was compile by VC ++, VB, Delphi ... are start using

```

PUSH EBP
MOV EBP, ESP

```

We move to the vet to **005008BC**, asm order go to:

```

PUSH EBP
MOV EBP, ESP
ADD ESP, 0C-<---- number of stolen bytes
MOV EAX, 50055C

```

we see the order on just khit with byte 0, o children at all about balance.

5. Must click, select "Copy to executable", "All modification"

Then select Copy all to notepad dialog. In just a new window opens, right click, select Save the file, the file is named dumped_2.exe

6. Now, in addition to the functions of PEEditor LordPe, choose File dumped_2.exe, at entrypoint, revise **OEP is 005008BC -400000 = = 001008BC**. Save again.

7. Run the file has unpack (dumped_2.exe).... File run better but still not crack crack. Each step would automatically make you the next.

How to unpack Asprotect 1:23 rc4 Series # 2

Name soft: Chronograph 3.1

Cracker: Computer_Angel

Level: DE

I. Tool needed:

- _ Ollydbg (debugger tool)
- _ Lord-Pe (tool to dump file)
- _ Imprec 1.6

II.Cach initiate:

Find OEP:

1. Olly Chrono.exe to load, the **shift-F9** to run the program, reality will automatically be stopped again by the point by exception asprotect generated. The number of **night-shift F9** from the original time to time for the message that xxx program run entirely. Xxx number that I received record here is 17, ie 17 times after the shift-F9 test program will run.

2. Load up Indeed, the **shift-F9** exactly 16 times, then press **Shift F7**. Press **Alt-M** to open the window on the **Memory**. Earn memory segment is available Owner "Chrono" and contains the "codes". Set **on break point** at which **memory access**.

3. Close window memory again, continue to return to the CPU window, press **F9**, we stop at the following code:

```
00401989 0000 ADD BYTE PTR DS: [EAX], AL
0040198B 0000 ADD BYTE PTR DS: [EAX], AL
0040198D 0000 ADD BYTE PTR DS: [EAX], AL
0040198F 00?
00401990 EB 10 JMP SHORT chrono.004019A2 here <----
00401992 66:623 A BOUND DI, DWORD PTR DS: [EDX]
00401995 43 INC EBX
00401996 2B2B SUB EBP, DWORD PTR DS: [EBX]
00401998 48 DEC EAX
00401999 4F DEC EDI
0040199A 4F DEC EDI
0040199B 4B DEC EBX
```

so OEP is 00401990. Asprotect characteristics is that it will be able to remove 1 of the first byte

of the main program and replace it with the value 00, so when you look at the top of the characters have 00 consecutive, I guess that was the exact OEP CT of the original.

Dump and fix IAT:

1. Content Lordpe, full dump process chrono files dumped.exe.
2. Then the next Imprec to fix IAT. Select chrono process is, go to **OEP 00401990-400000 == 1990**. The button "Autosearch IAT, imprec will automatically find and RVA area size for us. Here, I get results is RVA = 001FF414, size = 248
3. However, we do not choose RVA = 001FF414 that is selected 001FF000 for sure, and size is selected in 2000. Normally size = 1000 is very large but when the functions IAT Autosearch "imprec how you know the maximum size is 4000, so we spread the size 2000 for sure. Human button "Get Imports".
4. With that mind function **RVA: xxxxxxxx ptr: yyyyyyyyyyy, with interest yyyyyyy** too large ----> disassembler can not we can be cut thunk (s) to leave it because this is data "RAC" is not function.
5. It has heart function **RVA: 001FF138 ptr: 4BFE24** disassembler when we have the following code:

```
004BFE24 dec eax
004BFE25 or al, EB
004BFE27 pop edi
004BFE28 mov eax, [ebp-C]
004BFE2B inc dword ptr [eax + 8]
004BFE2E mov eax, [ebp-C]
004BFE31 inc dword ptr [eax + C]
004BFE34 jmp short 004BFE87
Cmp 004BFE36 byte ptr [ebp-5], 0
004BFE3A je short 004BFE44
004BFE3C mov eax, [ebp-C]
```

----> Here is that the code of the function 1 asprotect ---> unpack when the will o be used half ---> may be removed by using cut thunk (s)

6. The remaining function, we roster use the **tracer level 1 to find the API**, if tracer level 1 in the heart, is the next **tracer** plugin \ **asprotect 1:22** to find the API. If so, how the jam you'll also find the desire to be with 2 technical now.

Italy 7.Luu case following extras:

a) It has heart function **RVA: 001FF47C ptr: D31CF0**, when we are dissasmble

```
00D31CF0 push ebp
00D31CF1 mov ebp, esp
00D31CF3 mov eax, [D37E24] / / DWORD value: 00151EE0
00D31CF9 pop ebp
00D31CFA retn 4
```

This is ham FreeResource API, you DoubleClick to establish a name for it.

b) It has heart function **RVA: 001FF494 and RVA: 001FF498** when identifying are paid about GetCurrentProcessId is, we consider ham 2 code:

```
RVA: 001FF494:
00D31CB8 mov eax, [D37E18] / / DWORD value: FFFFFFFF
00D31CBD retn
```

```
RVA: 001FF498:
00D31CC0 mov eax, [D37E20] / / DWORD value: 00000200
00D31CC5 retn
```

It is potentially 1 ham is ham and 1 GetCurrentProcess is GetCurrentProcessId, the france try Tolerance is selected above and is GetCurrentProcess next GetCurrentProcessId 🏠

After you have finished repairing all ham in the table import functions, the button "fix dump" file and select dumped.exe created above to fix.

-> Imprec program will create a new file name 1 is dumped_.exe

Results:

Just a test file fix, you see reality better run, the PEID identify test to see how is Borland C + + 1999 🏠

-> Unpack our successful reality. Here as in the asprotect stolen bytes as in # 1 so we ignored the interruption recovery stolen bytes 🏠 .

How to unpack ASProtect by dqtl from Phudu Cracker Team Vietnam 2004

<http://www.phudu.com>

Victim: 3wGet 1.5 build 151 packed with **ASProtect 1:23 RC4 - 1.3.08.24 -> Alexey Solodovnikov [Overlay]**

from <http://www.3wget.com>

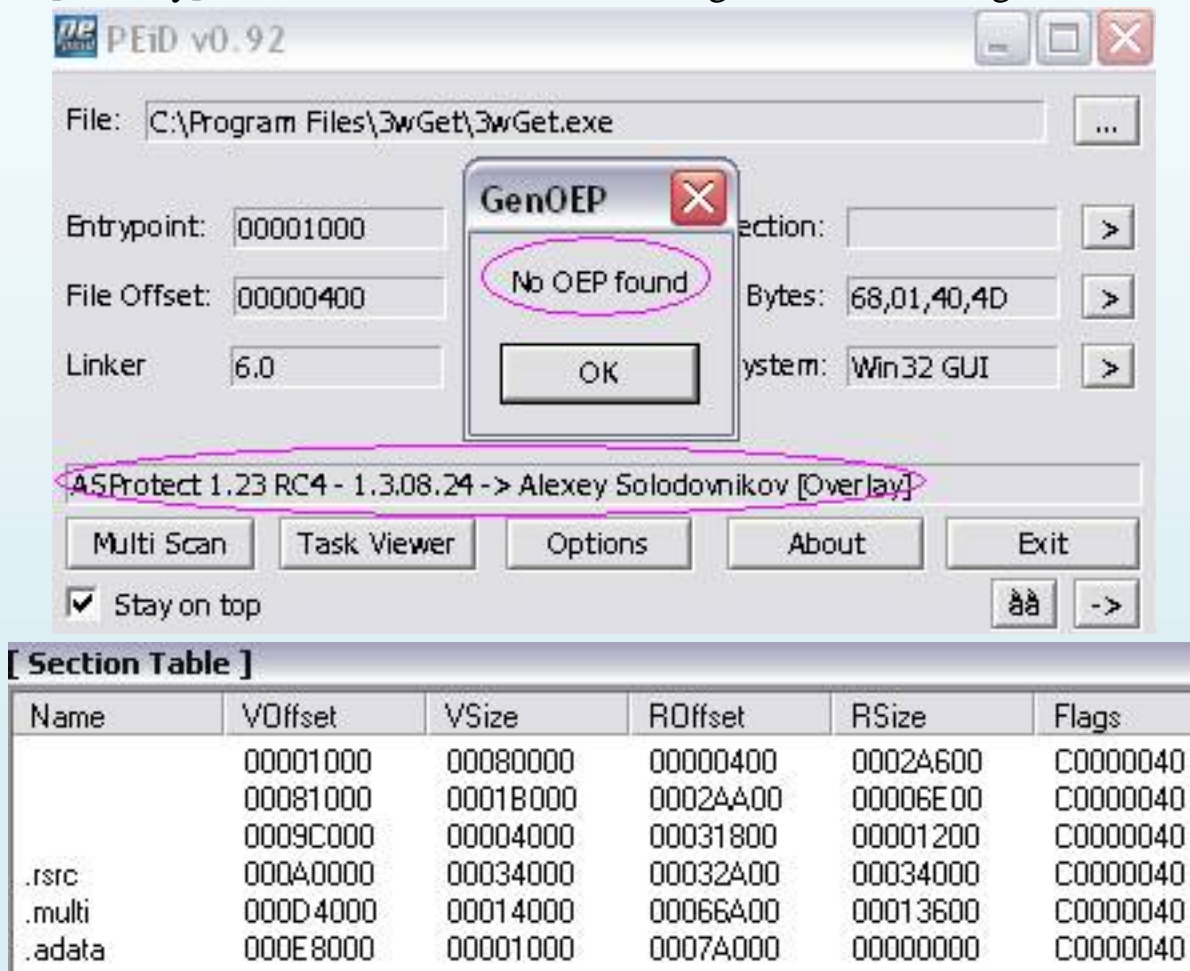
Homepage: <http://www.aspack.com>

Tools: 1:10 OllyDbg, PEiD 0.92, LordPE Deluxe, Import REConstructor 1.6 Final

Unpack the file: 3wGet.exe

Unpacked by dqtl

improved cup divertissement as deepen this month ... alcohol seems he is too excited packer this quite formidable đây ... now the network has many mup ASProtect, however the problem is finding Stolen Bytes is not clear ... dqtl will help you resolve this issue PEiD we use programs that are compressed using ASProtect 1:23 RC4 - 1.3.08.24 -> Alexey Solodovnikov [Overlay] and LordPE Deluxe for the Flags of the following



Finding the OEP

load the program in OllyDbg, press the Shift + F9 until the program runs ... load it in OllyDbg, press Shift + F9 to-1 times, before the run, we stop at the following code


```

00AA39EC 3100 XOR DWORD
PTR DS: [EAX], EAX => you
are here
00AA39EE 64:8 F05 00000000
POP DWORD PTR FS: [0]
..... here
are some
lines .....
00AA3A23 FF75 F0 PUSH
DWORD PTR SS: [EBP-10]
00AA3A26 FF75 EC PUSH
DWORD PTR SS: [EBP-14]
C3 RETN 00AA3A29 => set
breakpoint here

```

set break points at RETN order, press Shift + F9, more pressing Alt + F1, command TC EIP <900000, press ENTER and wait a little will to OEP = 4752A5

```

00475278 0000 ADD BYTE PTR
DS: [EAX], AL
0047527A 0000 ADD BYTE PTR
DS: [EAX], AL
..... here
are some
lines .....
004752A0 0000 ADD BYTE PTR
DS: [EAX], AL
004752A2 0000 ADD BYTE PTR
DS: [EAX], AL
004752A4 00
004752A5 FF15 DC174800
CALL DWORD PTR DS:
[4817DC]; msvcrt.
__set_app_type
004752AB 59 POP ECX
004752AC 830D 70FA4900 FF
OR DWORD PTR DS: [49FA70],
FFFFFFFF
004752B3 830D 74FA4900 FF
OR DWORD PTR DS: [49FA74],

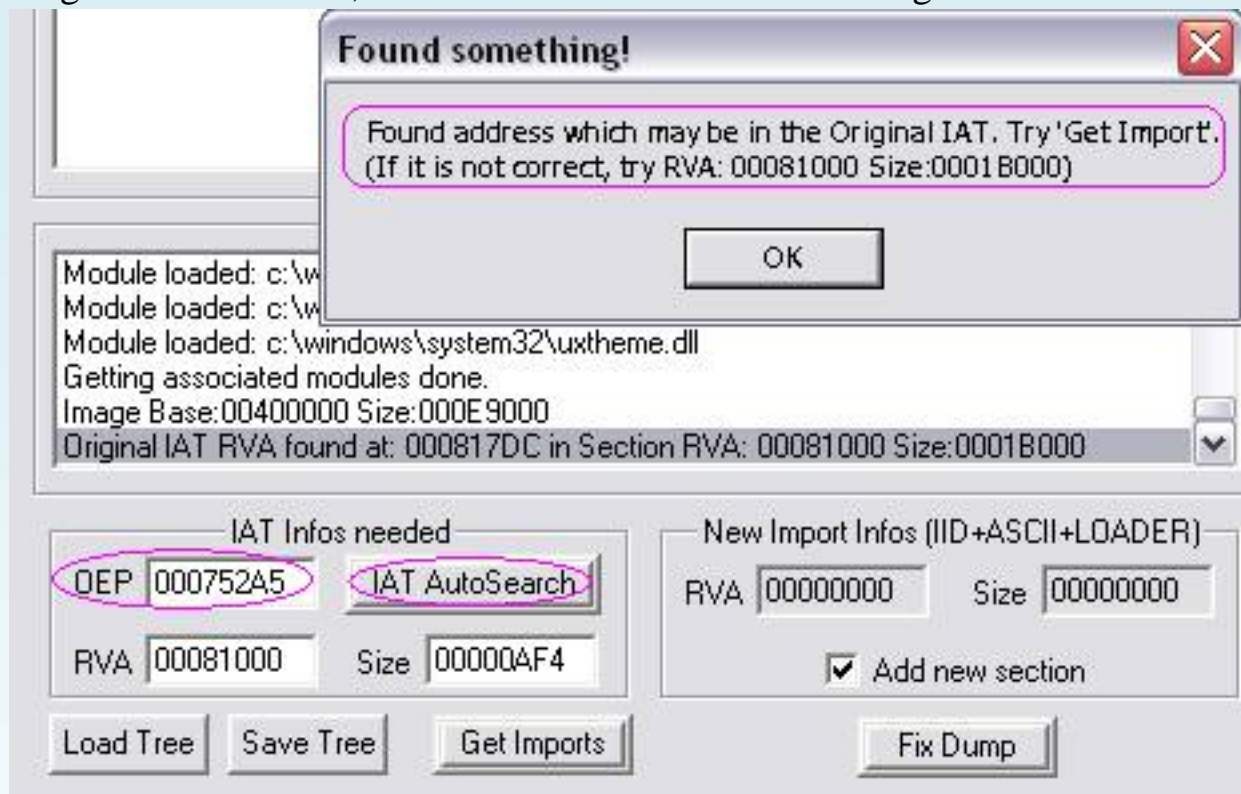
```

```

FFFFFFFF
004752BA FF15 E0174800
CALL DWORD PTR DS:
[4817E0]; msvcrt.
__p__fmode => you are here
OEP

```

pretty much have to find ways to OEP, dqtn will present more hours more
load the program in OllyDbg, press the Shift + F9 until the program runs ... load it in OllyDbg,
press Shift + F9 to-1 times, before running the program, press Shift + F7 and then Alt + M to
enable Memory map window ... Search with the Owner is 3wGet, Contains the code, right-click
to select Set memory breakpoint on access ... close window again OllyDbg, press F9, we stopped
at the OEP = 4752A5 ... count of bytes on the OEP we have seen all 45 bytes 00
here used to create LordPE Deluxe dumped.exe file ... Import REConstructor to use to fix
errors ... changes OEP = 752A5, click IAT AutoSearch as following



Import REConstructor we try to RVA: 81000 Size: 1B000 ... Size: 1B000 is too large, if you like
it just try ... here dqtn get Size = E90 ... then click Get Imports ... Functions are pretty much not
to fix ... Show Invalid clicks, right-click on the rva not fix, select Trace Level1, Import
REConstructor will fix some basic Funtions ... Show more Invalid clicks, right-click on the rva
not fix, select Plugin Tracers / ASProtect or Plugin Tracers 1:22 / 1:23 ASProtect RC4 depending
on what you are ok

Functions of the Import REConstructor not fix is the code of the garbage packer, right click select
Cut thunk (s) to remove it and then click Next Fix dump, select File dumped.exe to create the file

dumped_.exe ... create a new file is not run, we need to find to fix Stolen Bytes

Stolen Bytes

load the program in OllyDbg, press Shift + F9 to-1 times before running the program, set break points at RETN order, press Shift + F9 ... to press Alt + M, set the following breakpoint

The screenshot shows the OllyDbg interface for the file 3wGet.exe. The assembly window displays the following code:

```

00AA39EC 3100 XOR DWORD PTR DS:[EAX],EAX
00AA39EE 64:8F05 00000000 POP DWORD PTR FS:[0]
00AA39F5 58 POP EAX
00AA39F6 833D B07EAA00 0 CMP DWORD PTR DS:[AA7EB0],0
00AA39FD 74 14 JE SHORT 00AA3A13
00AA39FF 6A 0C PUSH 0C
00AA3A01 B9 B07EAA00 MOV ECX,0AA7EB0
00AA3A06 8D45 F8 LEA EAX,DWORD PTR SS:[EBP-8]
00AA3A09 BA 04000000 MOV EDX,4
00AA3A0E E8 20D1FFFF CALL 00AA0B40
00AA3A13 FF75 FC PUSH DWORD PTR SS:[EBP-4]
00AA3A16 FF75 F8 PUSH DWORD PTR SS:[EBP-8]
00AA3A19 8B45 F4 MOV EAX,DWORD PTR SS:[EBP-C]
00AA3A1C 8338 00 CMP DWORD PTR DS:[EAX],0
00AA3A1F 74 02 JE SHORT 00AA3A23
00AA3A21 FF30 PUSH DWORD PTR DS:[EAX]
00AA3A23 FF75 F0 PUSH DWORD PTR SS:[EBP-10]
00AA3A26 FF75 EC PUSH DWORD PTR SS:[EBP-14]
00AA3A29 C3 RETN
  
```

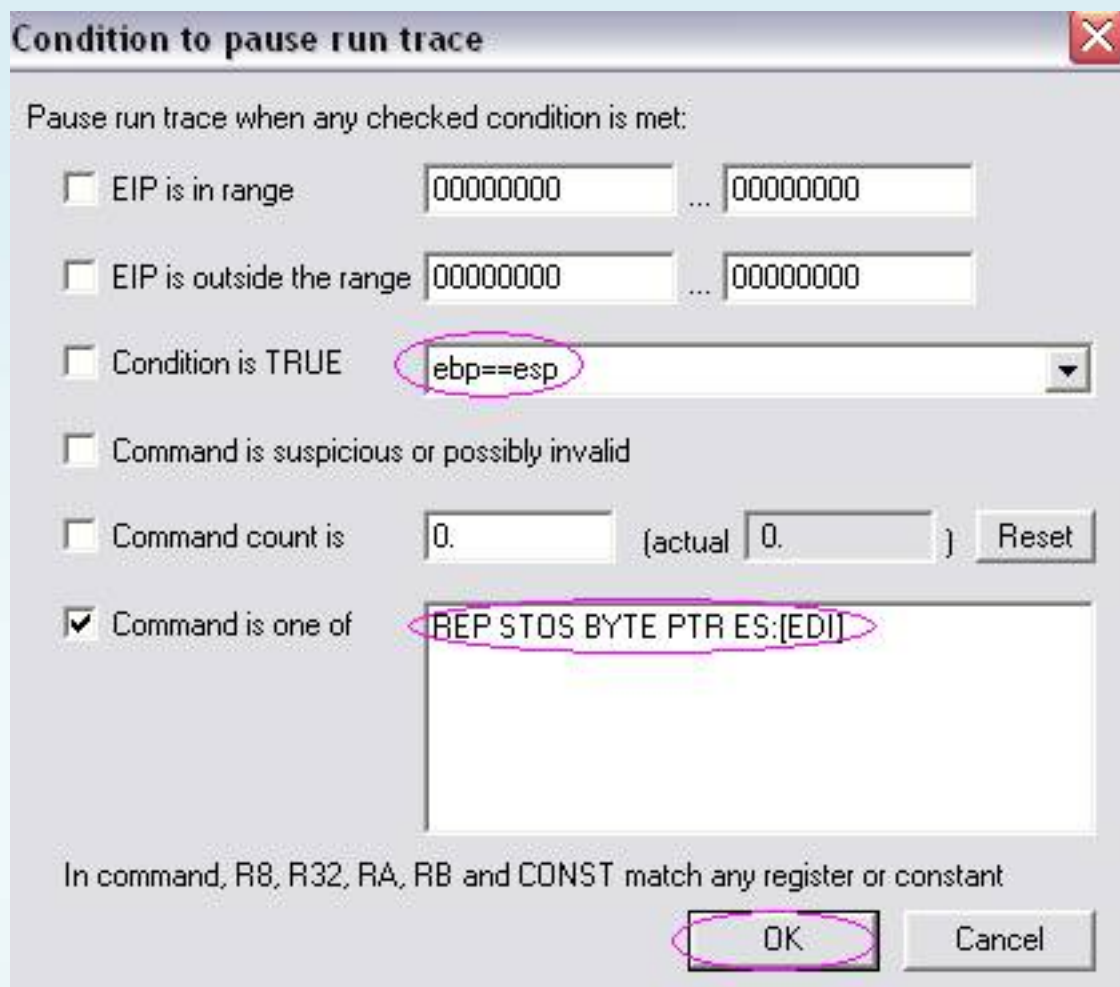
The Memory Map window is open, showing a list of memory segments. A context menu is displayed over the list, with the option "Set memory breakpoint on access" highlighted.

Address	Size	Name	Attributes	Map	R	R
003F0000	00002000	3wGet		PE header	Imag R	RWE
00400000	00001000	3wGet		code	Imag R	RWE
00401000	00008000	3wGet		data	Imag R	RWE
00481000	0001B000	3wGet		resources		
0049C000	00004000	3wGet		imports, x		
004A0000	00034000	3wGet	.rsrc			
004D4000	00014000	3wGet	.multi			
004E8000	00001000	3wGet	.adata			
004F0000	00005000					
005B0000	00002000					
005C0000	00103000					
006D0000	0009F000					
009D0000	00001000					
00A60000	00002000					
00A90000	0001E000					
00AB0000	0000C000					
00BB0000	00051000					
10000000	00001000	sockspy		PE header		
10001000	00009000	sockspy	.text	code		
1000A000	00001000	sockspy	.rdata	imports, e		
1000B000	00029000	sockspy	.data	data		
10034000	00002000	sockspy	.reloc	relocatio		
5AD70000	00001000	uxtheme		PE header		
5AD71000	0002C000	uxtheme	.text	code, impo		
5AD9D000	00001000	uxtheme	.data	data		
5AD9E000	00004000	uxtheme	.rsrc	resources		
5ADA2000	00002000	uxtheme	.reloc	relocatio		
70A70000	00001000	SHLWAPI		PE header		
70A71000	0005B000	SHLWAPI	.text	code, impo		
70ACC000	00001000	SHLWAPI	.data	data		
70ACD000	00002000	SHLWAPI	.rsrc	resources		

The context menu options are:

- Actualize
- View in Disassembler (Enter)
- Dump in CPU
- Dump
- Search (Ctrl+B)
- Set break-on-access (F2)
- Set memory breakpoint on access**
- Set memory breakpoint on write
- Set access
- Copy to clipboard
- Sort by
- Appearance

close window Memory map, press Ctrl + T command REP STOS BYTE PTR ES: [EDI] like



click OK, press Ctrl + F11 to wait a bit and we code to the

```

00AB5CEB
F3: AA
REP STOS
BYTE PTR
ES:
[EDI] =>
you are
here
00AB5CED
9D POPFD
00AB5CEE
5F POP
EDI
00AB5CEF
59 POP
ECX
00AB5CF0

```

C3 RETN

here select View / Run trace, dragging down the bottom, note the value ebp = esp

00AB78C7	ADC ESI,9320CB10	ESI=5F562A59
00AB78CD	PREFIX REPNE:	
00AB78D1	SUB ESI,DWORD PTR SS:[ESP+38]	ESI=5F161A59
00AB78D5	POP ESI	ESP=0012FFC4, ESI=77F5166A
00AB78D6	PREFIX REP:	
00AB78DB	PUSH EBP	ESP=0012FFC0
00AB78DC	MOV EBP,ESP	EBP=0012FFC0
00AB78DE	PUSH -1	ESP=0012FFBC
00AB78E0	PUSH 48BE58	ESP=0012FFB8
00AB78E5	PUSH 475046	ESP=0012FFB4
00AB78EA	MOV EAX,DWORD PTR FS:[0]	EAX=0012FFE0
00AB78F0	PREFIX REP:	
00AB78F5	PUSH EAX	ESP=0012FFB0
00AB78F6	MOV DWORD PTR FS:[0],ESP	
00AB78FD	SUB ESP,68	ESP=0012FF48
00AB7900	PREFIX REP:	
00AB7905	PUSH EBX	ESP=0012FF44
00AB7906	PREFIX REP:	
00AB790B	PUSH ESI	ESP=0012FF40
00AB790C	PREFIX REP:	
00AB7911	PUSH EDI	ESP=0012FF3C
00AB7912	MOV DWORD PTR SS:[EBP-18],ESP	
00AB7915	XOR EBX,EBX	EBX=00000000
00AB7917	MOV DWORD PTR SS:[EBP-4],EBX	
00AB791A	PUSH 2	ESP=0012FF38

edit the file as follows dumped_.exe

```

* OllyDbg - dumped_.exe - [CPU - main thread, module dumped_]
File View Debug Plugins Options Window Help
[Icons] [L] [E] [M] [T] [W] [H] [C] [/] [K] [B] [R] [...] [S]
00475278 55 PUSH EBP
00475279 8BEC MOV EBP,ESP
0047527B 6AFF PUSH -1
0047527D 68 58BE4800 PUSH dumped_.0048BE58
00475282 68 46504700 PUSH <JMP.&msvrt._except_handler3>
00475287 64:A1 00000000 MOV EAX,DWORD PTR FS:[0]
0047528D 50 PUSH EAX
0047528E 64:8925 000000 MOV DWORD PTR FS:[0],ESP
00475295 83EC 68 SUB ESP,68
00475298 53 PUSH EBX
00475299 56 PUSH ESI
0047529A 57 PUSH EDI
0047529B 8965 E8 MOV DWORD PTR SS:[EBP-18],ESP
0047529E 33DB XOR EBX,EBX
004752A0 895D FC MOV DWORD PTR SS:[EBP-4],EBX
004752A3 6A 02 PUSH 2
004752A5 $ FF15 DC174800 CALL DWORD PTR DS:[<&msvrt.__set_app_t msvcrt.__set_app_type
004752AB . 59 POP ECX
004752AC . 8300 70FA4900 OR DWORD PTR DS:[49FA70],FFFFFFFF
004752B3 . 8300 74FA4900 OR DWORD PTR DS:[49FA74],FFFFFFFF
004752BA . FF15 E0174800 CALL DWORD PTR DS:[<&msvrt._p__fmode> msvcrt._p__fmode
004752C0 . 8B00 50FA4900 MOV ECX,DWORD PTR DS:[49FA50]
004752C6 . 8908 MOV DWORD PTR DS:[EAX],ECX
004752C8 . FF15 E4174800 CALL DWORD PTR DS:[<&msvrt._p__commode msvcrt._p__commode

```

Deluxe LordPE use change Entry Point EP = EP = 752A5 to 75,278 ... save your changes ... PE rebuild to reduce the file size ... belgium hours dumped_.exe file running good conduct can crack greetings

If you have questions, Remarks about this tutorial, mail me
dqtlncrk@gmail.com

How to unpack ASProtect by dqtn from Phudu Cracker Team Vietnam 2004 <http://www.phudu.com>

Victim: BlazeDVD 3.5 Professional packed with ASProtect 1:23 RC4 - 1.3.08.24 -> Alexey Solodovnikov from <http://www.blazevideo.com>

Homepage: <http://www.aspack.com>

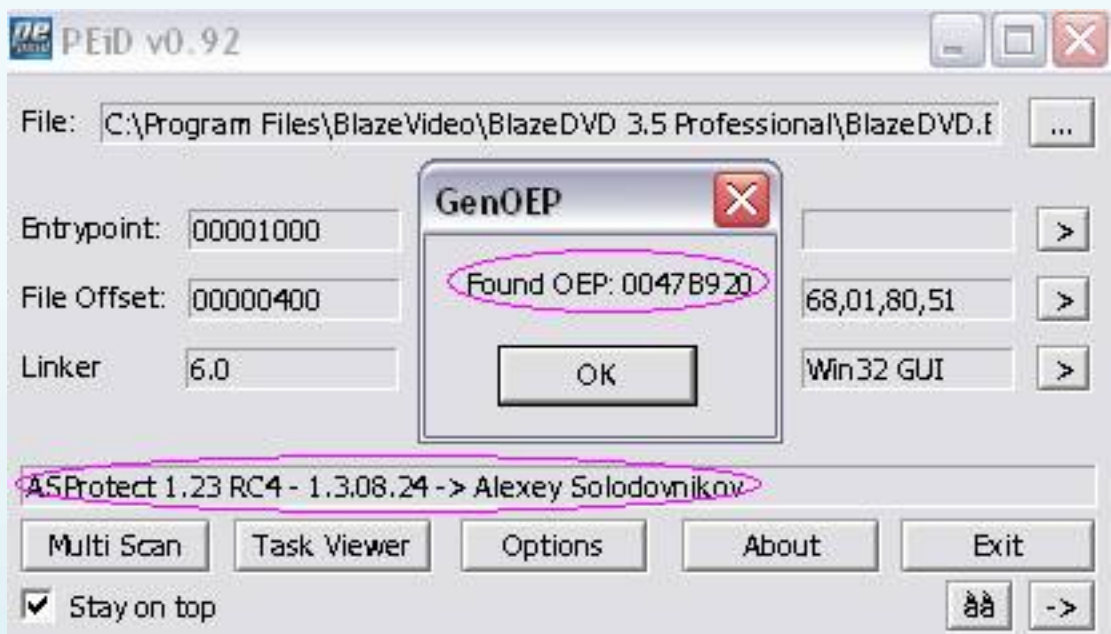
Tools: 1:10 OllyDbg, PEiD 0.92, LordPE Deluxe, Import REConstructor 1.6 Final

Unpack the file: BlazeDVD.EXE

Unpacked by dqtn

11h37 AM Thursday December 19 2004

PEiD we use programs that are compressed using ASProtect 1:23 RC4 - 1.3.08.24 -> Alexey Solodovnikov, OEP = 47B920 (which the flight) and LordPE Deluxe for the Flags of the following



PEiD v0.92

File: C:\Program Files\BlazeVideo\BlazeDVD 3.5 Professional\BlazeDVD.E

Entrypoint: 00001000

File Offset: 00000400

Linker: 6.0

GenOEP

Found OEP: 0047B920

OK

ASProtect 1.23 RC4 - 1.3.08.24 -> Alexey Solodovnikov

Multi Scan Task Viewer Options About Exit

☒ Stay on top

[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
	00001000	000A2000	00000400	00040E00	C0000040
	000A3000	00003000	00041200	00001800	C0000040
	000A6000	0001F000	00042A00	00008E00	C0000040
	000C5000	0001A000	0004B800	00004600	C0000040
.rsrc	000DF000	00039000	0004FE00	00008200	C0000040
.data	00118000	0001F000	00058000	0001EA00	C0000040
.adata	00137000	00001000	00076A00	00000000	C0000040

load the program in OllyDbg, press the Shift + F9 until the program runs about 29 times the OllyDbg notification header problems, not the stars ... load it in OllyDbg, press Shift + F9 to-1 times, before the run, we stop at the following code


```
00B839EC 3100 XOR DWORD
PTR DS: [EAX], EAX => are
here
00B839EE 64:8 F05 00000000
POP DWORD PTR FS: [0]
..... here
are some
lines .....
00B83A26 FF75 EC PUSH
DWORD PTR SS: [EBP-14]
C3 RETN 00B83A29 => set
breakpoint here
```

set break points at RETN order, press Shift + F9, press Alt + F1 command all eip <900,000, hit enter and wait a bit to the right will OEP = 4787E1 ... above 00 is 38 bytes, 38 stolen bytes

```
004787BB 0000 ADD BYTE PTR
DS: [EAX], AL
004787BD 0000 ADD BYTE PTR
DS: [EAX], AL
..... here
are some
lines .....
004787DF 0000 ADD BYTE PTR
DS: [EAX], AL
004787E1 FF15 44634A00
CALL DWORD PTR DS:
[4A6344] => are here OEP
004787E7 33D2 XOR EDX, EDX
004787E9 8AD4 MOV DL, AH
004787EB 8915 38BF4D00 MOV
DWORD PTR DS: [4DBF38],
EDX
```

Stolen find Bytes tutorial as 12, see pictures

Condition to pause run trace ✕

Pause run trace when any checked condition is met:

☐ EIP is in range ...

☐ EIP is outside the range ...

☐ Condition is TRUE

☐ Command is suspicious or possibly invalid

☐ Command count is (actual)

☒ Command is one of

In command, R8, R32, RA, RB and CONST match any register or constant

00B9699F	POP DWORD PTR SS:[ESP]	ESP=0012FFC0	
00B969A3	MOV EBP,ESP	EBP=0012FFC0	EBP=ESP
00B969A5	PUSH -1	ESP=0012FFBC	
00B969A7	PUSH 4B21F8	ESP=0012FFB8	
00B969AC	PUSH 4772F8	ESP=0012FFB4	
00B969B1	MOV EAX,DWORD PTR FS:[0]	EAX=0012FFE0	
00B969B7	ADD WORD PTR DS:[B969C1],0CE9		
00B969C0	JMP SHORT 00B969C5		
00B969C5	PREFIX REP:		
00B969CA	LEA ESP,DWORD PTR SS:[ESP+EBP-2]	ESP=0025FF72	PUSH EBP
00B969CE	SUB ESP,EBP	ESP=0012FFB2	
00B969D0	SUB ESP,2	ESP=0012FFB0	
00B969D6	SUB WORD PTR DS:[B969E0],6CAD		
00B969DF	JMP SHORT 00B969E3		
00B969E3	PUSH EAX	ESP=0012FFAC	
00B969E4	ADD WORD PTR DS:[B969ED],0E508		
00B969ED	JMP SHORT 00B969F2		
00B969F2	POP DWORD PTR SS:[ESP]	ESP=0012FFB0	
00B969F6	MOV DWORD PTR FS:[0],ESP		
00B969FD	SUB ESP,58	ESP=0012FF58	
00B96A00	ADD WORD PTR DS:[B96A0A],0CE9		
00B96A09	JMP SHORT 00B96A0E		
00B96A0E	PREFIX REP:		
00B96A13	LEA ESP,DWORD PTR SS:[ESP+EBP-2]	ESP=0025FF16	
00B96A17	SUB ESP,EBP	ESP=0012FF56	
00B96A19	SUB ESP,2	ESP=0012FF54	
00B96A1F	SUB WORD PTR DS:[B96A29],6CAD		
00B96A28	JMP SHORT 00B96A2C		
00B96A2C	PUSH EBX	ESP=0012FF50	
00B96A2D	ADD WORD PTR DS:[B96A36],0E508		
00B96A36	JMP SHORT 00B96A3B		
00B96A3B	POP DWORD PTR SS:[ESP]	ESP=0012FF54	
00B96A3F	ADD WORD PTR DS:[B96A49],0CE9		
00B96A48	JMP SHORT 00B96A4D		
00B96A4D	PREFIX REP:		
00B96A52	LEA ESP,DWORD PTR SS:[ESP+EBP-2]	ESP=0025FF12	
00B96A56	SUB ESP,EBP	ESP=0012FF52	
00B96A58	SUB ESP,2	ESP=0012FF50	
00B96A5E	SUB WORD PTR DS:[B96A68],6CAD		
00B96A67	JMP SHORT 00B96A6B		
00B96A6B	PUSH ESI	ESP=0012FF4C	
00B96A6C	ADD WORD PTR DS:[B96A75],0E508		
00B96A75	JMP SHORT 00B96A7A		
00B96A7A	POP DWORD PTR SS:[ESP]	ESP=0012FF50	
00B96A7E	ADD WORD PTR DS:[B96A88],0CE9		
00B96A87	JMP SHORT 00B96A8C		
00B96A8C	PREFIX REP:		
00B96A91	LEA ESP,DWORD PTR SS:[ESP+EBP-2]	ESP=0025FF0E	
00B96A95	SUB ESP,EBP	ESP=0012FF4E	
00B96A97	SUB ESP,2	ESP=0012FF4C	
00B96A9D	SUB WORD PTR DS:[B96AA7],6CAD		
00B96AA6	JMP SHORT 00B96AAA		
00B96AAA	PUSH EDI	ESP=0012FF48	
00B96AAB	ADD WORD PTR DS:[B96AB4],0E508		
00B96AB4	JMP SHORT 00B96AB9		
00B96AB9	POP DWORD PTR SS:[ESP]	ESP=0012FF4C	
00B96ABD	MOV DWORD PTR SS:[EBP-18],ESP		
00B96AC0	JMP SHORT 00B96AC3		
00B96AC3	PREFIX REP:		
00B96AC8	PUSH 4787E1	ESP=0012FF48	
00B96ACD	PUSH 0B96861	ESP=0012FF44	
00B96AD2	RETN	ESP=0012FF48	

after dumped and fixed IAT, edit image files as follows dumed_.exe and Deluxe LordPE change Entry Point EP = EP = 787E1 to 787BB ... save your changes ... PE rebuild to reduce the file size ... belgium hours dumped_.exe file running good conduct can crack

IAT Infos needed

IEP 000787E1

RVA 000A6000

IAT AutoSearch

Size 000007CC

New Import Infos (IID+ASCII+LOADER)

RVA 00000000

☒ Add new section

Size 000023B6

Load Tree
Save Tree
Get Imports
Fix Dump

OllyDbg - dumped_.exe - [CPU - main thread, module dumped_]

File View Debug Plugins Options Window Help

L
E
M
T
W
H
C
/
K
B

<div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787B8</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787BC</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787BE</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787C0</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787C5</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787CA</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787D0</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787D1</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787D8</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787DB</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787DC</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787DD</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787DE</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787E1</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787E7</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787E9</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">004787EB</div>	<div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">55</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">88EC</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">6A FF</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">68 F8214B00</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">68 F8724700</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">64:A1 00000000</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">50</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">64:8925 00000000</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">83EC 58</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">53</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">56</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">57</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">8965 E8</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">FF15 44634A00</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">. 33D2</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">. 8AD4</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">. 8915 38BF4D00</div>	<div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">PUSH EBP</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">MOV EBP,ESP</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">PUSH -1</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">PUSH dumped_.004821F8</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">PUSH dumped_.004772F8</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">MOV EAX,DWORD PTR FS:[0]</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">PUSH EAX</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">MOV DWORD PTR FS:[0],ESP</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">SUB ESP,58</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">PUSH EBX</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">PUSH ESI</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">PUSH EDI</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">MOV DWORD PTR SS:[EBP-18],ESP</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">CALL DWORD PTR DS:[&kernel32.GetVersion]</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">XOR EDX,EDX</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">MOV DL,AH</div> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">MOV DWORD PTR DS:[4DBF38],EDX</div>	<div style="border: 1px solid gray; padding: 2px; margin-bottom: 2px;">Entry address</div>
---	--	---	--

greetings

If you have questions, Remarks about this tutorial, mail me

dqtlncrk@gmail.com

How to unpack exe32pack by dqtn from Phudu Team Vietnam 2005 <http://www.phudu.com>

Victim: exe32pack 1:42

Homepage: <http://www.steelbytes.com>

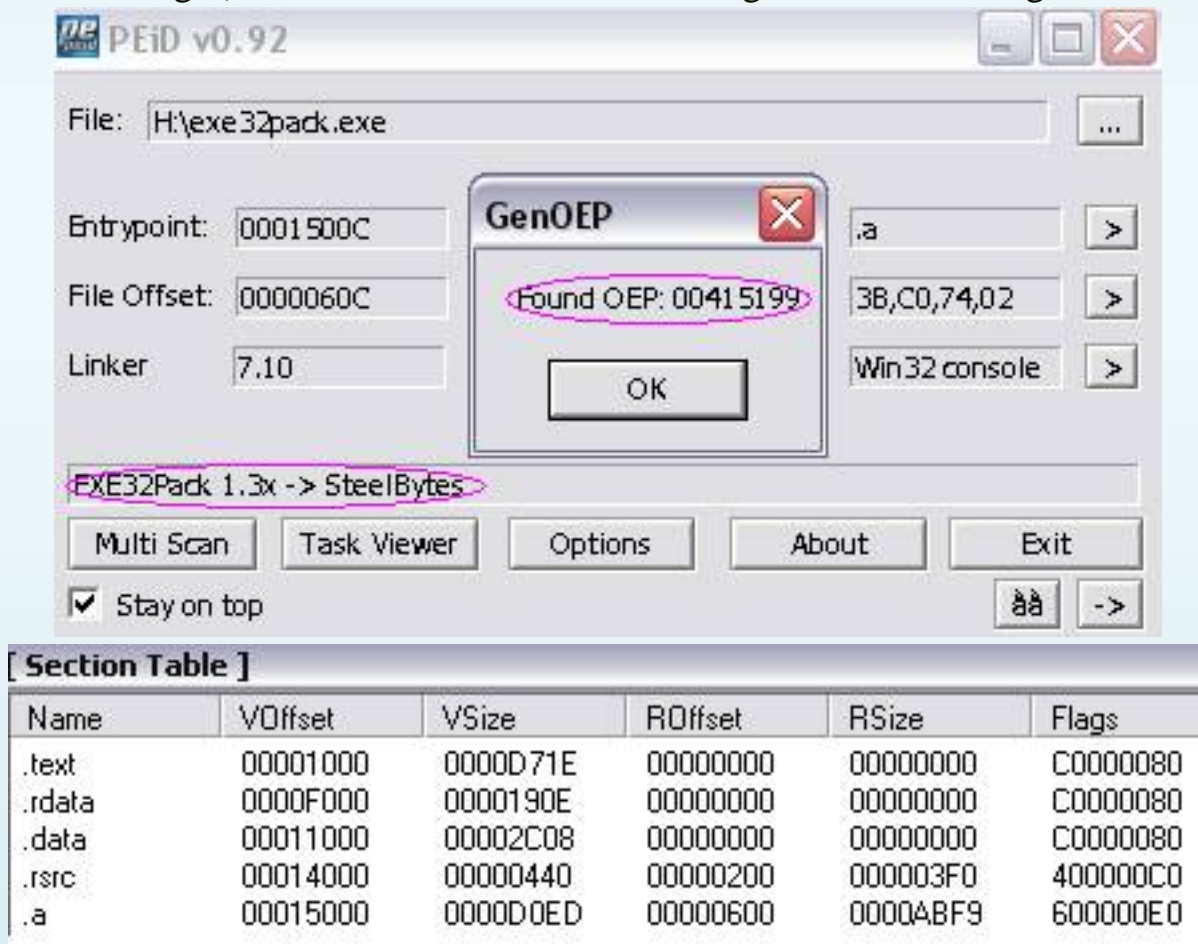
Tools: 1:10 OllyDbg, PEiD 0.92, LordPE Deluxe, Import REConstructor 1.6 Final

Unpack the file: exe32pack.exe

Unpacked by dqtn

12h30 am Thursday January 03 2005

PEiD we use programs that are compressed using EXE32Pack 1.3x -> SteelBytes, OEP = 415,199 (which the flight) and LordPE Deluxe for the Flags of the following



load the program in OllyDbg, we must press F8 mỗi hand to run new programs ... setting breakpoint on JE orders, if any orders jumped repeating the breakpoint is set under the command ... after a very long time, we will take command JMP EAX finally ... it jump to OEP đây

however dqtn have found ways to help OEP faster hehe ... after stops in EP, press Ctrl + B, enter 8F 85, we came here

```
004150DF 8F85
273C4000 POP
DWORD PTR SS:
[EBP +403 C27] =>
are here, set
breakpoint here
004150E5 3BF6 Cmp
ESI, ESI
004150E7 74 02 JE
SHORT
exe32pac.004150EB
```

breakpoint in the set, press F9 ... remove breakpoint ... press Ctrl + B, enter FF E0, we came here

```
0041513A 83FF E0
Cmp EDI, -20 =>
are here
0041513D 0050 01
ADD BYTE PTR
DS: [EAX +1], DL
00415140 000E
ADD BYTE PTR
DS: [ESI], CL
00415142 7F 00 JG
SHORT
exe32pac.00415144
```

press Ctrl + G, enter the 41513B, an increase the value that we see as follows

```
0041513B FFE0
JMP EAX => are
here, set breakpoint
here
0041513D 0050 01
ADD BYTE PTR
DS: [EAX +1], DL
00415140 000E
ADD BYTE PTR
DS: [ESI], CL
00415142 7F 00 JG
SHORT
exe32pac.00415144
```

breakpoint in the set, press F9 ... remove breakpoint ... press F8 one time only ... press Ctrl + B, enter FF E0, press Ctrl + L until OllyDbg report found no more, we found as follows


```
0042015A
BF
FFE0B801
MOV EDI,
1B8E0FF
=> are here
0042015F
0000 ADD
BYTE PTR
DS: [EAX],
AL
00420161
003B ADD
BYTE PTR
DS: [EBX],
BH
C9 LEAVE
00420163
```

press Ctrl + G, enter the 42015B, an increase the value that we see as follows

```
0042015B FFE0
JMP EAX => are
here, set breakpoint
here
0042015D b8
01000000 MOV
EAX, 1
00420162 3BC9
Cmp ECX, ECX
00420164 74 02 JE
SHORT
exe32pac.00420168
```

breakpoint in the set, press F9 ... remove breakpoint ... press F8 one time only one day to OEP = 407F0E ... & dumped fixed the IAT as easy tutorials before ... unpack dqtlncr tried many programs written in MASM, TASM, Visual C++ is compressed using exe32pack 1:42 by this are all ok ... seem not exe32pack 1:42 program should be written in VB, not very clear

greetings

If you have questions, Remarks about this tutorial, mail me

dqtlncrk@gmail.com

How to unpack FSG

by dqtn from Phudu Crocker Team



Original Vietnamese text:

Tools : OllyDbg 1.10 , PEiD 0.92 , LordPE Deluxe , Import REConstructor 1.6 Final

View [Suggest a better translation](#)

Homepage: <http://www.phudu.com>

Tools: 1:10 OllyDbg, PEiD 0.92, LordPE Deluxe, Import REConstructor 1.6 Final


Unpack the file: unpackme FSG 1.33.exe

Unpacked by dqtn

PEiD we use programs that are compressed with FSG 1:33 -> dulek / XT and OEP = 401128



OllyDbg configuration as follows

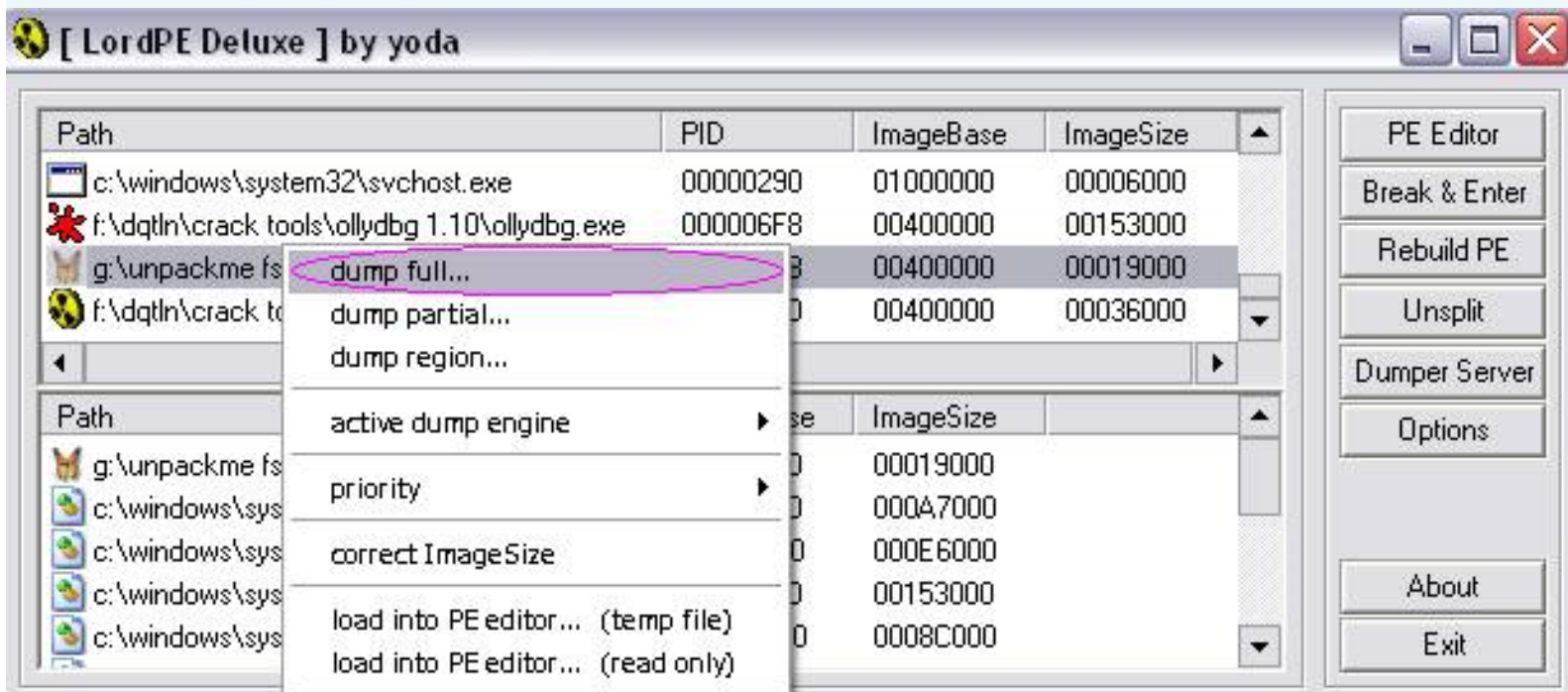


OllyDbg - unpackme FSG 1.33.exe - [CPU - main thread, module unpackme]

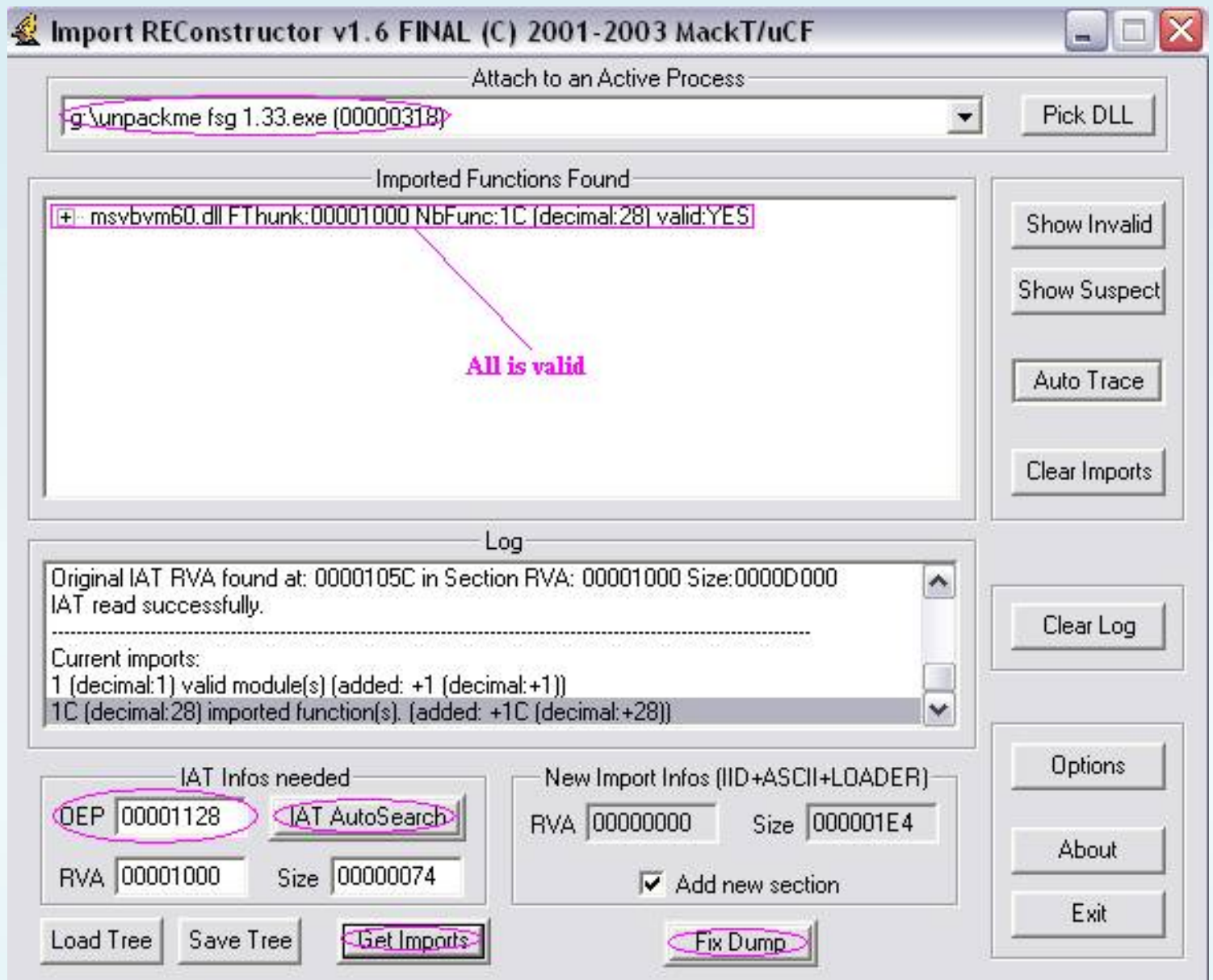
File View Debug Plugins Options Window Help

Address	Disassembly	Comment
00401128	PUSH unpackme.0040A824	Real entry point of SFX code
0040112D	CALL unpackme.00401122	JMP to MSUBUM60.ThunRTMain
00401132	ADD BYTE PTR DS:[EAX],AL	
00401134	ADD BYTE PTR DS:[EAX],AL	
00401136	ADD BYTE PTR DS:[EAX],AL	
00401138	XOR BYTE PTR DS:[EAX],AL	
0040113A	ADD BYTE PTR DS:[EAX],AL	

file:///C:/RCE%20Unpacking%20eBook%20[Tra...umLi]/How%20to%20unpack%20FSG%20v1.33.htm (2 of 5) [1/9/2009 9:44:32 LithiumLi]



run file dumped.exe drilling because it will crash ... we need to use Import REConstructor to fix errors ... changes OEP = 1128 ... IAT AutoSearch click, click Get more Imports and finally click Fix dump, select File dumped.exe to create the file dumped_.exe



test file dumped_.exe see good work ... normal after the dump file typically has the code of garbage packer, we can use Deluxe LordPE to reduce the size of the file ... however when clicking rebuild PE program will crash ... we need to change all of the Flags

Tweaking the dump

IMAGE_SCN_CNT_CODE 0x00000020 Section contains executable code

IMAGE_SCN_CNT_INITIALIZED_DATA 0x00000040 Section contains initialized data

IMAGE_SCN_CNT_UNINITIALIZED_DATA 0x00000080 Section contains uninitialized data

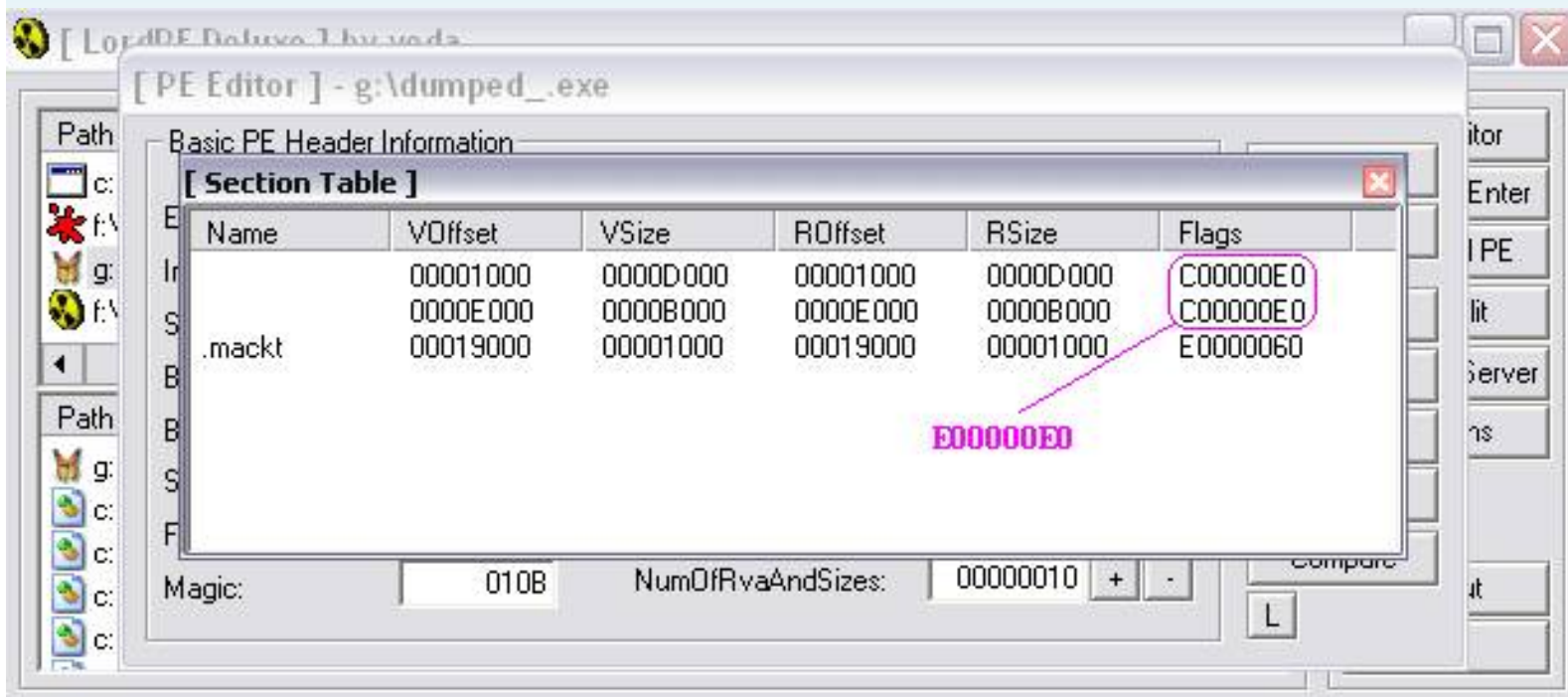
IMAGE_SCN_MEM_EXECUTE 0x20000000 Section can be executed as code

IMAGE_SCN_MEM_READ 0x40000000 Section can be read

IMAGE_SCN_MEM_WRITE 0x80000000 Section can be written to

$0x20 + 0x40 + 0x80 + 0x20000000 + 0x40000000 + 0x80000000 = 0xE00000E0$

Deluxe uses LordPE press PE Editor, select File dumped_.exe, click the button to change the Sections and the Flags, click Save, OK, such as following



here you can click rebuild PE to reduce the file size ... Actually, the steps are not needed for our program just run and debug can proceed to crack ... however, learn some knowledge is better: D ... so through our tutorials to learn how to unpack FSG 1:33, a file compression program exe good and free ... FSG - F [AST] S [mall] G [ood]

greetings

If you have questions, Remarks about this tutorial, mail me

dqtln@phudu.com

How to unpack FSG

by dqtl from Phudu Cracker Team
Vietnam 2004
<http://www.phudu.com>

Victim: Minesweeper 5.1 packed with FSG 2.0

Homepage: Copyright © 1981-2001 Microsoft Corporation by Robert Donner and Curt Johnson

Tools: 1:10 OllyDbg, PEiD 0.92, LordPE Deluxe, Import REConstructor 1.6 Final

Unpack the file: winmine.exe

Unpacked by dqtl

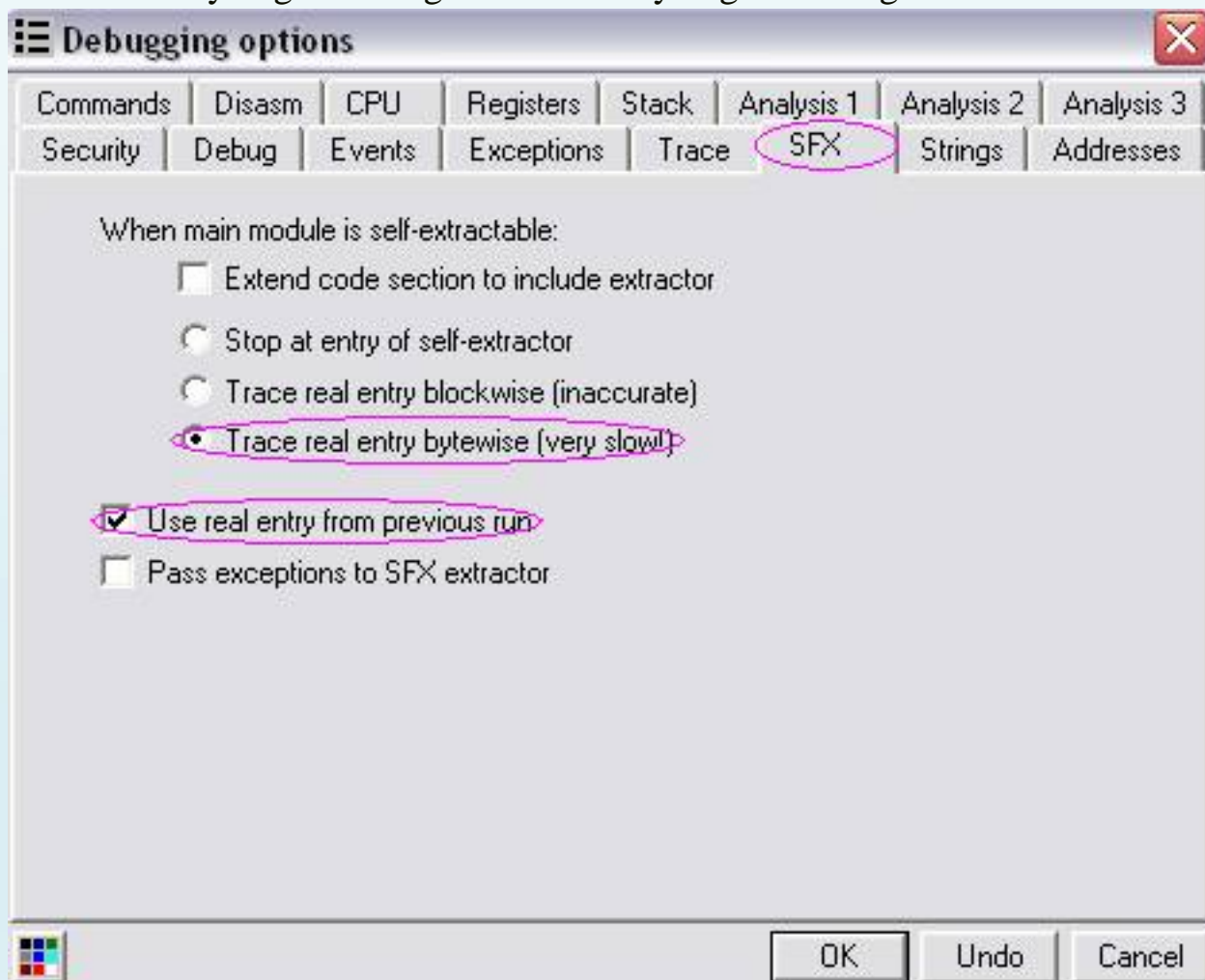
first we need to add the following code into the file of PEiD 0.92 userdb.txt to identify it FSG 2.0

```
[FSG 2.0 -> bart / XT]
```

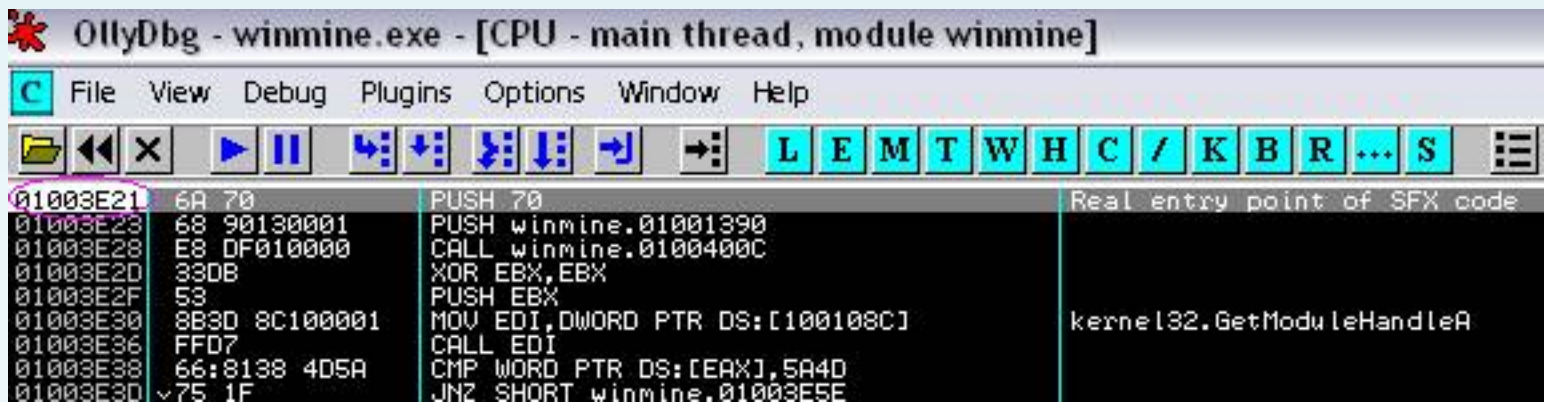
```
signature = 87 25? ? ? 61 94 55 A4 B6 80 FF 13 73 F9 33 C9 FF 13 73 16 33 C0  
FF 13 73 1F B6 80 41 B0
```

```
ep_only = true
```

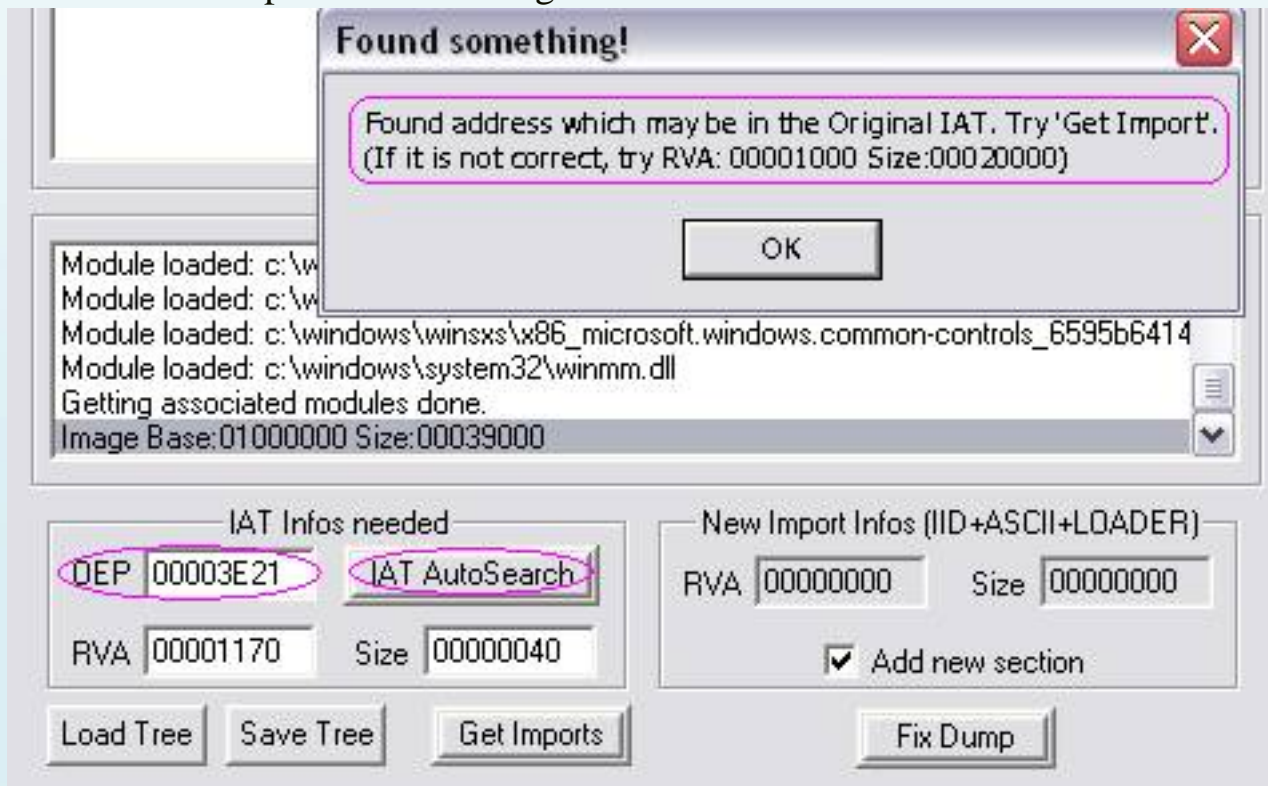
PEiD we use programs that are compressed with FSG 2.0 -> bart / XT and receive not find OEP
so we will find it in OllyDbg ... configuration as OllyDbg following



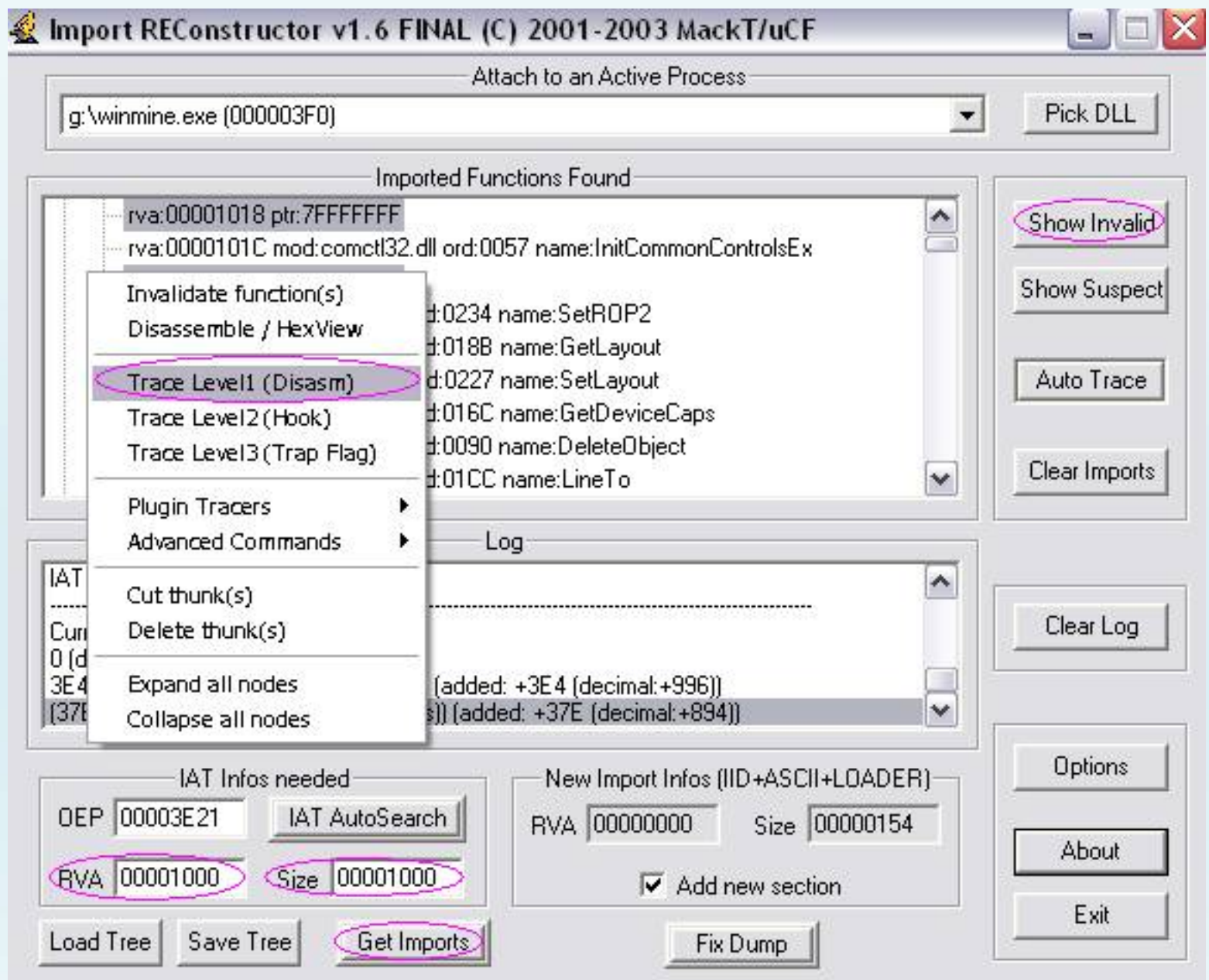
winmine load in OllyDbg and wait a little, we will stop at OEP = 1003E21



here used LordPE Deluxe memory dump file created dumped.exe ... run file dumped.exe drilling because it will crash ... we need to use Import REConstructor to fix errors ... changes OEP = 3E21 ... IAT AutoSearch press as following



if you still do before the tutorial is not ... Import REConstructor we try to RVA: 1000 Size: 20000 ... Size: 20000 is too large, if you like it just try ... here dqtn get Size = 1000 ... then click Get Imports ... Functions are pretty much not to fix ... click ... Show Invalid right click on the rva not fix, select Trace Level1 as following



Import of functions that are REConstructor not fix the code of the garbage packer, right click select Cut thunk (s) to remove it and then click Next Fix dump, select File dumped.exe to create the file dumped_.exe ... check file dumped_.exe running well, but quite large in size by the lot code refuse ... can use LordPE Deluxe, rebuild PE reduce the file size ... so through this tutorial we learn how to unpack FSG 2.0, a file compression program exe good and free ... FSG - F [AST] S [mall] G [ood]

greetings

If you have questions, Remarks about this tutorial, mail me

dqtl@phudu.com

How to unpack PELock by dqtn from Phudu Team Vietnam 2005 www.phudu.com

Victim: softwares packed with [1.0x PELock](#)

Homepage: www.pelock.prv.pl

Tools: 1:10 OllyDbg, PEiD 0.93, Final Import REConstructor 1.6, 1.4 IsDebuggerPresent OllyDbgPlugin

Unpack the file: unpackme1.exe

8h06 am Thursday 26 February 2005

PEiD we use programs that are compressed using [PELock 1.0x -> Bartosz Wojcik](#) ... load the program in OllyDbg press F9, then Shift + F9, try clicking the dqtn 1h also not eat none, including the removal of much of OllyDbgOptions ... Actually, the API IsDebuggerPresent use it to check for program debug running or not, if it has not escaped notice or such other programs that make us always receive each hand ah IsDebuggerPresent used for anti OllyDbgPlugin 1.4 if you have it ... if not do the following load the program in OllyDbg, Right Click / Search for / name in all modules, to find the API IsDebuggerPresent, double click on it, we will come here

```
77E72740> 64: A1 18000000 MOV EAX, DWORD PTR FS: [18] => you are here, set breakpoint here
77E72746 8B40 30 MOV EAX, DWORD PTR DS: [EAX +30]
77E72749 0FB640 02 MOVZX EAX, BYTE PTR DS: [EAX +2] => eax = 0 if not debug the program, now in its 1
77E7274D C3 RETN
```

after the set break points in 77E72740, press F9, OllyDbg will end ... remove breakpoint, press F8 to see it 77E72749 will bring value to 1 to record eax, ie programs are run debug ... change the value 0, then press F9 ... continue to press the Shift + F9 until the program runs ... do the above steps, press Shift + F9 to-1 before running the program, we found as follows

```
00324986
8900 MOV
DWORD
PTR DS:
[EAX],
EAX =>
you are
here
00324988
EB 03 JMP
SHORT
0032498D
0032498A
```

```

DBE3
FINIT
0032498C
61 POPAD
0032498D
EB 02 JMP
SHORT
00324991
0032498F
CD 20 INT
20
00324991
EB 02 JMP
SHORT
00324995

```

here, press Alt + M, set the memory breakpoint tutorials before ... Return to the main window OllyDbg, press Shift + F9 one time only one to OEP = 401002, this is only a temporary stop OEP, OEP has actually encrypted

```

00401000 05 82 =>
OEP has actually
encrypted ... 2 Stolen
Bytes
00401002? E8
CF040000 CALL
unpackme.004014D6
=> OEP
00401007? A3
CA204000 MOV
DWORD PTR DS:
[4020CA], EAX
0040100C. 6A 00
PUSH 0
0040100E. 68
E3204000 PUSH
unpackme.004020E3;
ASCII "No need to
disasm the code!"
00,401,013th E8
76040000 CALL

```

unpackme.0040148E

OEP at press Ctrl + B to enter FF 25 for rva start and size as the previous tutorials, we are results

OEP: 401002

IATRVA: 403184

IATSize: 104

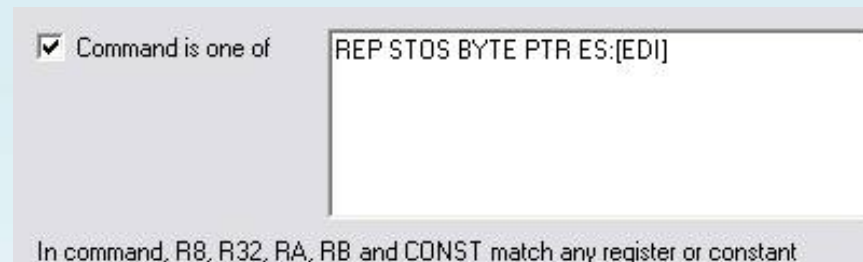
OEP at OllyDump 2.21.108 OllyDbgPlugin used to create a memory dump file dumped.exe ... here if you use Import REConstructor IAT to fix the many that IAT Import REConstructor not fix the ... there are several ways fix manually but will be quite long, but what is applicable to many REConstructor Import packer that does not fix because you do not trace through this API in OllyDbg, dqtn will be presented later this month

Next we find Stolen Bytes, this step is very important

implementation of the above steps, press Shift + F9 to-1 before running the program, we found as follows

```
00324986 8900 MOV DWORD PTR DS: [EAX], EAX => you are here
00324988 EB 03 JMP SHORT 0032498D
0032498A DBE3 FINIT
0032498C 61 POPAD
0032498D EB 02 JMP SHORT 00324991 => set breakpoint here
0032498F CD 20 INT 20
00324991 EB 02 JMP SHORT 00324995
```

set break points in 32498D, press Shift + F9, delete breakpoint ... press Ctrl + T and then enter REP STOS BYTE PTR ES: [EDI] as the following



click OK again OllyDbg main window, press Ctrl + F11 and wait a bit ... OllyDbg when stopped, click View / Run trace we see as follows

1135.	Main		00324C57	PUSH 0	ESP=0012FFB8
1134.	Main		00324C59	JMP SHORT 00324C5C	
1133.	Main		00324C5C	SAL EBX,0	
1132.	Main		00324C5F	JMP SHORT 00324C63	
1131.	Main		00324C63	JMP SHORT 00324C66	
1130.	Main		00324C66	JMP SHORT 00324C6A	
1129.	Main		00324C6A	JMP SHORT 00324C6D	
1128.	Main		00324C6D	JMP SHORT 00324C71	
1127.	Main		00324C71	JMP SHORT 00324C74	
1126.	Main		00324C74	JMP SHORT 00324C77	
1125.	Main		00324C77	SAL EDX,0	
1124.	Main		00324C7A	JMP SHORT 00324C7E	
1123.	Main		00324C7E	SAL ESP,0	
1122.	Main		00324C81	SAL EBP,0	
1121.	Main		00324C84	JMP SHORT 00324C87	
1120.	Main		00324C87	SAL EAX,0	
1119.	Main		00324C8A	JMP SHORT 00324C8E	
1118.	Main		00324C8E	JMP SHORT 00324C92	
1117.	Main		00324C92	JMP SHORT 00324C96	
1116.	Main		00324C96	JMP SHORT 00324C9A	
1115.	Main		00324C9A	JMP SHORT 00324C9E	
1114.	Main		00324C9E	PUSH 401002	ESP=0012FFB4
1113.	Main		00324CA3	JMP SHORT 00324CA7	
1112.	Main		00324CA7	JMP SHORT 00324CAB	
1111.	Main		00324CAB	JMP SHORT 00324CAF	
1110.	Main		00324CAF	JMP SHORT 00324CB3	
1109.	Main		00324CB3	JMP SHORT 00324CB6	
1108.	Main		00324CB6	JMP SHORT 00324CB9	
1107.	Main		00324CB9	JMP SHORT 00324CBC	
1106.	Main		00324CBC	JMP SHORT 00324CC0	
1105.	Main		00324CC0	JMP SHORT 00324CC3	
1104.	Main		00324CC3	JMP SHORT 00324CC6	
1103.	Main		00324CC6	RETN	ESP=0012FFB8 ESP=0012FFB4
1102.	Main	unpackme	00401002	CALL unpackme.004014D6	
1101.	Main	unpackme	004014D6	JMP DWORD PTR DS:[403238]	
1100.	Main		00A904D6	JMP kernel32.GetModuleHandleA	
1099.	Main	kernel32	77E7AD86	CMP DWORD PTR SS:[ESP+4],0	
1098.	Main	kernel32	77E7AD88	JE kernel32.77E7AEC8	
1097.	Main	kernel32	77E7AEC8	MOV EAX,DWORD PTR FS:[18]	EAX=7FFDE000
1096.	Main	kernel32	77E7AECB	MOV EAX,DWORD PTR DS:[EAX+30]	EAX=7FFDF000
1095.	Main	kernel32	77E7AED1	MOV EAX,DWORD PTR DS:[EAX+8]	EAX=00400000
1094.	Main	kernel32	77E7AED4	JMP kernel32.77E7ADA6	
1093.	Main	kernel32	77E7ADA6	RETN 4	ESP=0012FFB8 ESP=0012FFB4 ESP=0012FFB0
1092.	Main	unpackme	00401007	MOV DWORD PTR DS:[4020CA],EAX	
1091.	Main	unpackme	0040100C	PUSH 0	
1090.	Main	unpackme	0040100E	PUSH unpackme.004020E3	
1089.	Main	unpackme	00401013	CALL unpackme.0040148E	
1088.	Main	unpackme	0040148E	JMP DWORD PTR DS:[403204]	

Stolen Bytes luôn bắt
đầu tại esp=xxFFB8

OEP tạm thời ... nhờ đó
mà ta tìm được Stolen
Bytes

so we found Stolen Bytes: PUSH 0 ... dqtn try to do with programs written in MASM, TASM the Stolen Bytes is always PUSH 0

Next we fix IAT using OllyDbg trace through the API to Import REConstructor be identified

configuration OllyDbg: Options / Exceptions / Memory Access indoctrination ... load the program in OllyDbg, anti API IsDebuggerPresent memory, press Shift + F9 to-1 times before to run the program, we found as follows

```
00322EC5
F7F1 DIV
ECX => you
are here
00322EC7
202B AND
BYTE PTR
DS: [EBX],
CH
00322EC9
C9 LEAVE
..... here
are some
lines .....
00322ED5
EB 02 JMP
SHORT
00322ED9
=> set
breakpoint
here
00322ED7
E5 00 IN
EAX, 0; I /
O command
```

set break points in 322ED5, press Shift + F9, delete breakpoint ... press Ctrl + F6 enter C1 B 80, we see the following

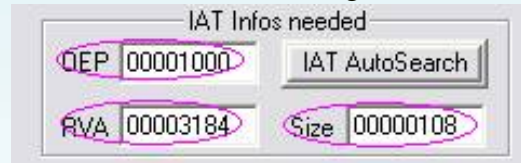
```
00323B5B
F6C1 80
TEST CL,
80 => are
here
00323B5E
EB 02 JMP
SHORT
00323B62
00323B60
CD 20 INT
20
```

```
00323B62
EB 02 JMP
SHORT
00323B66
```

at 323B5B, Right Click / Breakpoint / Hardware, on execution ... press F9 and then change the value to record the CL 80



continue to do so until the program runs ... Import REConstructor use the following fix



why IATSize = 108 while the one found IATSize = 104 ... the fix is done, load the fix was in the fix we find OllyDbg RVA final fix should not we increase 4 units for more REConstructor Import fix all

greetings

If you have questions, Remarks about this tutorial, mail me

dqtlncrk@gmail.com

How to unpack PESpin by dqtn from Phudu Team Vietnam 2005 www.phudu.com

Victim: softwares packed with **0.3 PESpin**

Homepage: pespin.w.interia.pl

Tools: OllyDbg 1:10, OllyDump 2.21.108 OllyDbgPlugin, Import REConstructor 1.6 Final

Unpack the file: winmine.exe XP SP1

12h58 am Tuesday 18 April 2005

[PESpin 0.3 -> cyberbob]

signature = EB 01 68 60 E8 00 00 00 00 8B 1C 24 83 C3 12 81 2B E8 B1 06 00 FE 4B FD 82
2C 24 B7 CD 46 00 0B

ep_only = true

configuration OllyDbg: Options / debugging options/Exceptions/INT3 breaks ... load the program
in OllyDbg, press F8 2 times we came here

```
0102008A 60  
PUSHAD  
0102008B E8  
00000000 CALL  
winmine.01020090  
=> you are here  
01020090 8B1C24  
MOV EBX,  
DWORD PTR SS:  
[ESP]  
01020093 83C3 12  
ADD EBX, 12  
01020096 812B  
E8B10600 SUB  
DWORD PTR DS:  
[EBX], 6B1E8
```

look through the window Registers (FPU), ESP = 6FFA4 see ... note this value, we will use it ...
press Shift + F9 until you see the following code

```
010001D9
FFFF? ;
Unknown
command
=> you are
here
010001DB
FFFF? ;
Unknown
command
010001DD
FFFF? ;
Unknown
command
010001DF
FFFF? ;
Unknown
command
```

press Alt + F1, BP LoadLibraryA command, Enter ... OllyDbg again, press Shift + F9, we found as follows

```
77E7D961> 837C24
04 00 Cmp DWORD
PTR SS: [ESP +4], 0
=> you are here
77E7D966 53 PUSH
EBX
77E7D967 56 PUSH
ESI
77E7D968 74 19 JE
SHORT
kernel32.77E7D983
77E7D96A 68
443EE877 PUSH
kernel32.77E83E44;
ASCII "twain_32.
dll"
77E7D96F FF7424
10 PUSH DWORD
PTR SS: [ESP +10]
```

```
77E7D973 FF15
9813E677 CALL
DWORD PTR DS:
[<& ntdll._strcmpi>];
ntdll._strcmp
..... here are some
lines .....
77E7D98B E8
B1FFFFFF CALL
kernel32.
LoadLibraryExA
77E7D990 5e POP
ESI
77E7D991 5B POP
EBX
77E7D992 C2 0400
RETN 4
```

remove breakpoint, press F8 over command RETN 4, we found as follows

```
01020A4D 85C0
TEST EAX, EAX;
msvcrt.77C10000
=> you are here
01020A4F 0F84
28060000 JE
winmine.0102107D
01020A55 50 PUSH
EAX
01020A56 E8
C5FCFFFF CALL
winmine.01020720
```

the code in the API will take place ... you find the command JMP DWORD PTR SS: [ESP-4] by pressing Ctrl + B to enter FF6424FC, more pressing Ctrl + L ... order to find the 3rd stop, under the command will order the FAR JMP EAX, we found as follows


```
01020B53 FF6424
FC JMP DWORD
PTR SS: [ESP-4] =>
you are here
01020B57 FFE8
FAR JMP EAX;
Illegal use of
register
01020B59 0300
ADD EAX,
DWORD PTR DS:
[EAX]
01020B5B 0000
ADD BYTE PTR
DS: [EAX], AL
01020B5D EB 04
JMP SHORT
winmine.01020B63
```

press F2 set points in the end, press F9, remove break points (Hardware can put on in the execution) ... move to the morning to order FAR JMP EAX, press Ctrl + E, enter 90EB19 (this change is the default for PESpin 0.3), we found as follows

```
01020B53 FF6424
FC JMP DWORD
PTR SS: [ESP-4];
winmine.01020B73
=> you are here ...
after the break point
set, this command
will jump to
1020B73 ... it can
jump to 1020B58,
but with this file is
not packed with
file ... if it jumps to
the 1020B58 we
change it to jump on
the correct order by
FAR JMP EAX to
order new OpCodes
```

```

90EB19
01020B57 90 NOP
01020B58 EB 19
JMP SHORT
winmine.01020B73
01020B5A 0000
ADD BYTE PTR
DS: [EAX], AL
01020B5C 00EB
ADD BL, CH
01020B5E 04 0F
ADD AL, 0F
01020B60 ^ EB FB
JMP SHORT
winmine.01020B5D
    
```

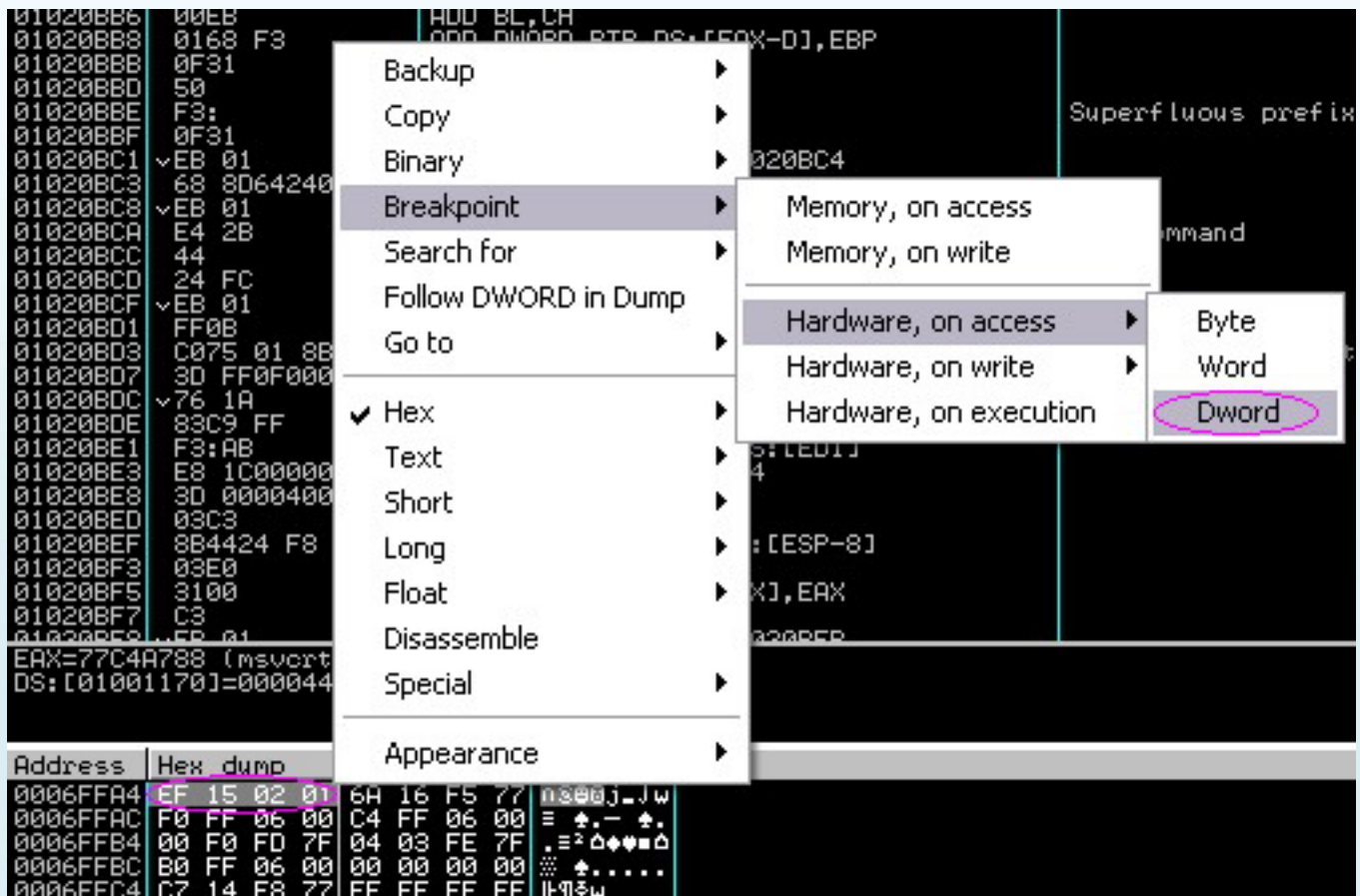
press F8 to see the following ... command attention MOV DWORD PTR DS: [EDX], EAX ...
OpCodes: 8902

```

01020B73 8902
MOV DWORD PTR
DS: [EDX], EAX;
msvcrt._controlfp =>
you are here ... with
some other files are
compressed, there is
order Call, please
change it to
8902909090
01020B75 EB 01
JMP SHORT
winmine.01020B78
4B DEC EBX
01020B77
01020B78 83C2 04
ADD EDX, 4
01020B7B ^ E9
26FFFFFF JMP
winmine.01020AA6
=> jump back to
continue in the API
    
```

```
code
01020B80 E8
83661000 CALL
01127208
01020B85 74 01 JE
SHORT
winmine.01020B88
01020B87 C483
C614E987 LES
EAX, FWORD PTR
DS: [EBX +87
E914C6];
modification of
segment register
01020B8D FE? ;
Unknown command
01020B8E FFFF? ;
Unknown command
01020B90 C9
LEAVE
```

you also remember that the value ESP dqtln above ... but the window Hex dump (bottom left), press Ctrl + G to enter and set the breakpoint 6FFA4 as following



after set breakpoint, press F9 will come

```

01020C0D 6A 70
PUSH 70 => you are
here, Stolen Bytes
first such
01020C0F EB 01
JMP SHORT
winmine.01020C12
01020C11 90 NOP
01020C12 68
90130001 PUSH
winmine.01001390
01020C17 EB 01
JMP SHORT
winmine.01020C1A
01020C19 ^ E2 E8
LOOPD SHORT
winmine.01020C03
01020C1B ED IN
EAX, DX; I / O
    
```

```

command
01020C1C 33FE
XOR EDI, ESI
01020C1E FF33
PUSH DWORD
PTR DS: [EBX]
01020C20 DBEB
FUCOMI ST, ST (3)
01020C22 01B2
53EB0115 ADD
DWORD PTR DS:
[EDX +1501 EB53],
ESI
01020C28-E9
0332FEFF JMP
winmine.01003E30
=> jump to OEP
temporary, 1003E30
first byte is 00,
started at 1003E21

```

here OllyDump 2.21.108 OllyDbgPlugin used to dump the memory ... Import open
 REConstructor fix IAT, easy because the API Open code
 OEP REConstructor = Import 1020C0D - 20C0D = 1000000
 RVA = 1000

Size = 1B8

this packed with files Import REConstructor know we have to RVA and Size is correct ... Other
 files are packed we will have to find Rva and Size in a new (old way to find FF 25 as the first
 tutorials were no longer useful packer with this) ... please wait in the other tutorials

after IAT fix, run the file, unpack the whole ... however 1020C0D OEP OEP is not original, if
 you want to edit the original OEP (in fact do not need the steps below) to find the following
 Stolen Bytes

at 1020C0D PUSH has ordered 70 Stolen Bytes is the first ... JMP command the next (with other
 OpCodes E9), or submit your 3 bytes as follows

```
01020C0D 6A 70
PUSH 70 => Stolen
Byte
01020C0F 90 NOP
01020C10 90 NOP
01020C11 90 NOP
01020C12 68
90130001 PUSH
winmine.01001390
=> Stolen Byte
01020C17 90 NOP
01020C18 90 NOP
01020C19 90 NOP
01020C1A E8
ED33FEFF CALL
winmine.0100400C
=> Stolen Byte
01020C1F 33DB
XOR EBX, EBX =>
Stolen Byte
01020C21 90 NOP
01020C22 90 NOP
01020C23 90 NOP
01020C24 53 PUSH
EBX => Stolen Byte
01020C25 90 NOP
01020C26 90 NOP
01020C27 90 NOP
01020C28-E9
0332FEFF JMP
winmine.01003E30
=> jump to OEP
temporary, the
original OEP
1003E21
```

1003E21 to change the Stolen Bytes we see the same match ... dump & fix IAT as the
This is a packer to unpack ... now many packer has the encryption APIs are in
thanks **fly & David**

greetings

If you have questions, Remarks about this tutorial, mail me

dqtlncrk@yahoo.com

How to unpack Petite

by dqtn from Phudu Team
Vietnam 2005
<http://www.phudu.com>

Victim: Petite 2.2

Homepage: <http://www.un4seen.com/petite>

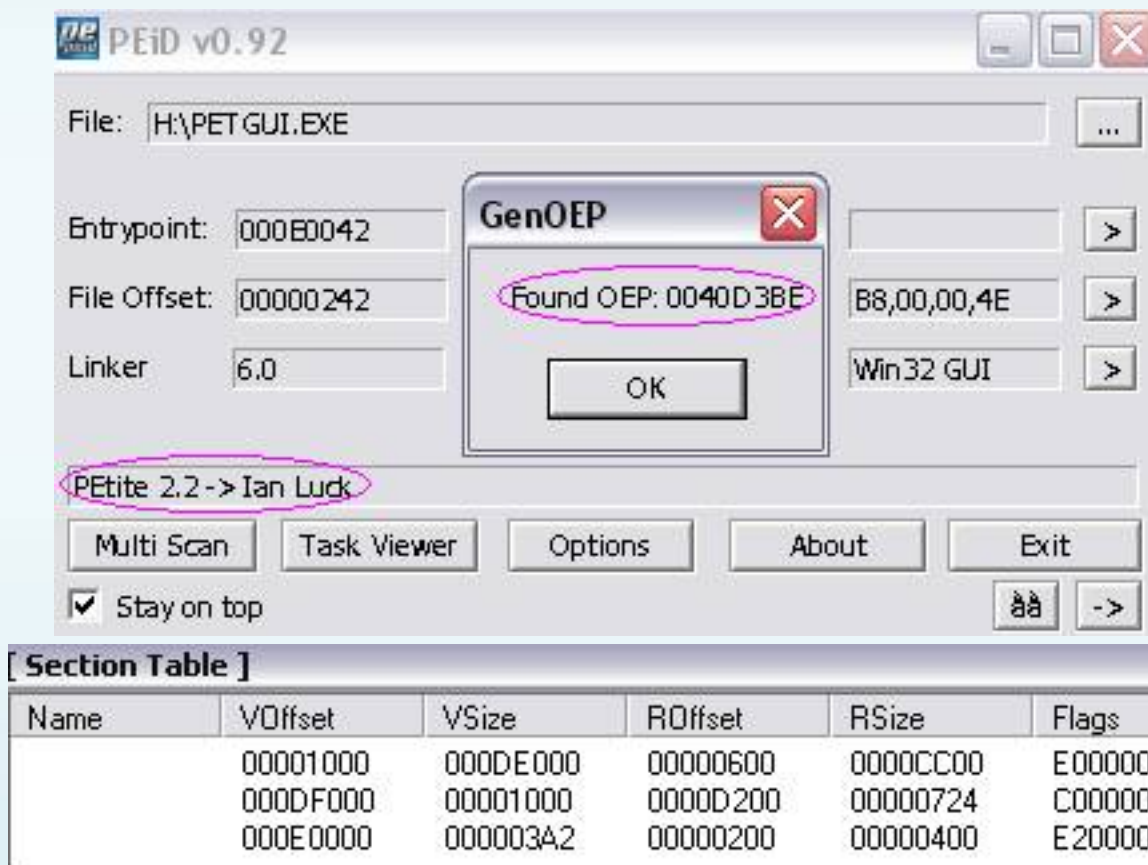
Tools: 1:10 OllyDbg, PEiD 0.92, LordPE Deluxe, Import REConstructor 1.6 Final

Unpack the file: PETGUI.EXE

Unpacked by dqtn

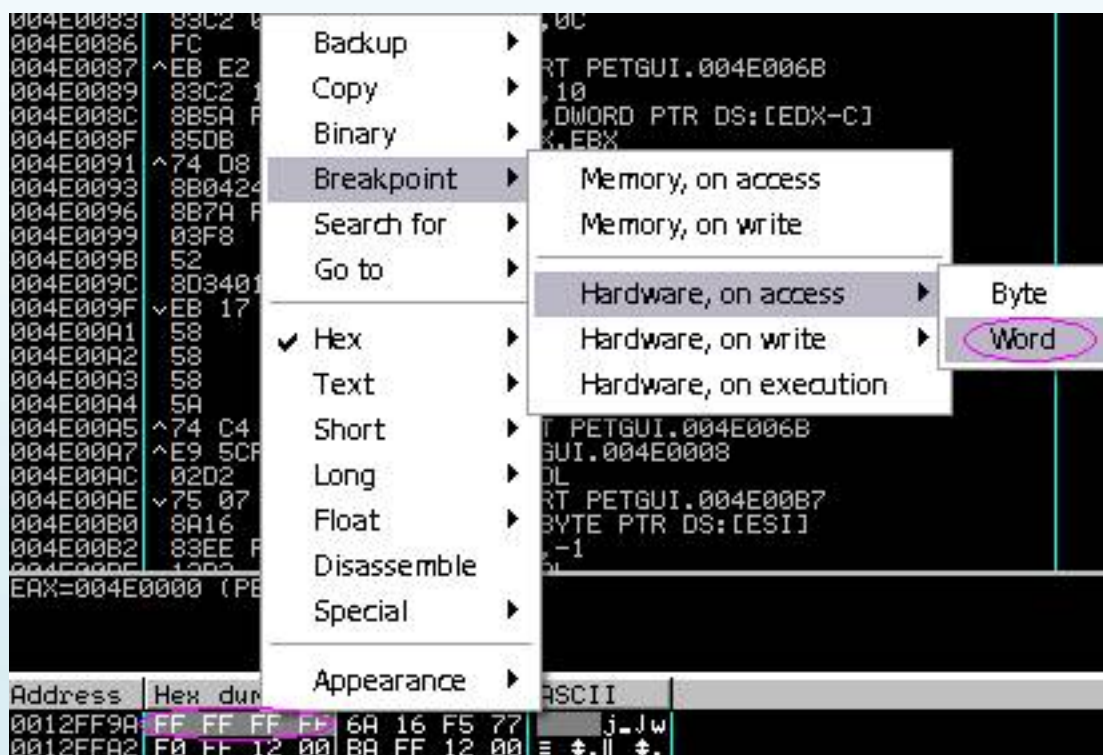
3h06 AM Tuesday 10 January 2005

PEiD use and LordPE Deluxe informations know ... OEP = 40D3BE - incorrect



load program in OllyDbg, the land at Entry Point EP = 4E0042 ... set breakpoint at PUSHAD, run F9, F8 PUSH EAX arrive ... Right Click ESP / Follow in dump ... Set Breakpoint / Hardware, on access / Word

```
004E0042> b8
00004E00 MOV
EAX,
PETGUI.004E0000
=> land here
004E0047 68
F3534100 PUSH
PETGUI.004153F3
004E004C 64: FF35
00000000 PUSH
DWORD PTR FS:
[0]
004E0053 64:8925
00000000 MOV
DWORD PTR FS:
[0], ESP
004E005A 66:9 C
PUSHFW
004E005C 60
PUSHAD => set
breakpoint here
004E005D 50 PUSH
EAX => F8 arrive
here
004E005E 33DB
XOR EBX, EBX
```

Press F9, Shift + F9, Shift + F9 ... u arrive here

```

004E003D 66:9 D POPFW => land here
004E003F 83C4 08 ADD ESP, 8
004E0042>-E9 31D5F2FF JMP
PETGUI.0040D578 => jump OEP
.....
0040D578 55 PUSH EBP => = OEP 40D578
0040D579 8BEC MOV EBP, ESP
0040D57B 6A FF PUSH -1
0040D57D 68 80434100 PUSH
PETGUI.00414380
0040D582 68 78FF4000 PUSH
PETGUI.0040FF78
0040D587 64: A1 00000000 MOV EAX, DWORD
PTR FS: [0]

```

IAT dump & fix now, it's very easy

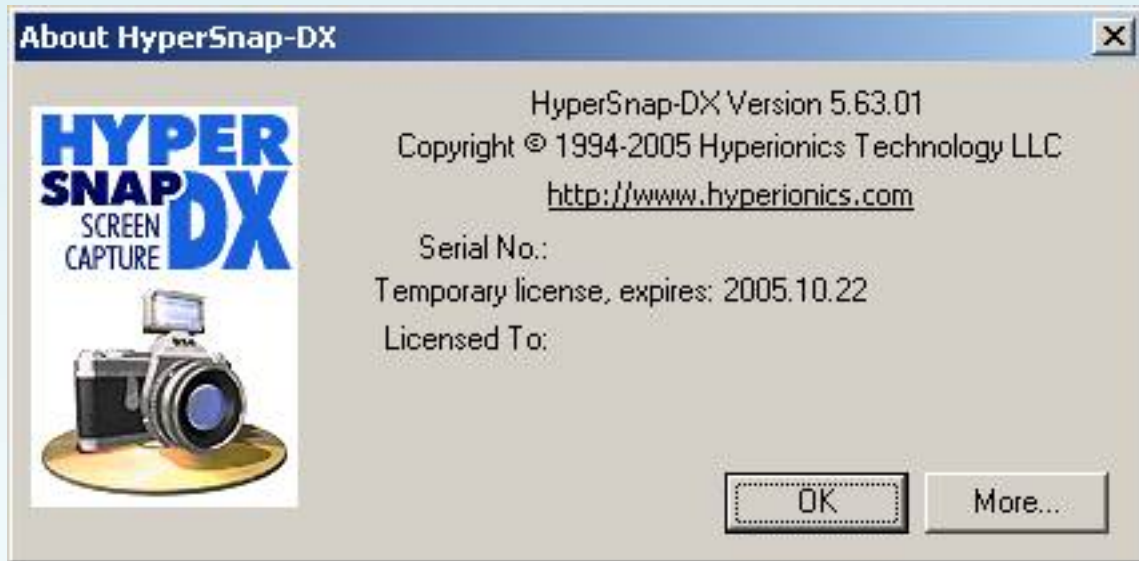
greetings

If you have questions, Remarks about this tutorial, mail me

dqtlncrk@gmail.com

Armadillo collect sand-stone

Hyperionics.HyperSnap-DX.v5.63.01.ENG <|> ARM 4.xx - Standard Protection + IAT elimination + Code Splicing



_Hi All, the soft nry to capture henh, I tenh flags lum it. Go try cri, forked arm Ir children. He he, hearing it all!

_Load Target:

Address	Hex dump	Disassembly	Comment
00699000	60	JMP SHORT HprSnap5.00699006	
00699001	E8 00000000	CALL HprSnap5.00699006	
00699006	50	POP EBP	
00699007	50	PUSH EAX	
00699008	51	PUSH ECX	
00699009	0FCA	BSWAP EDX	
0069900B	F7D2	NOT EDX	
0069900D	9C	PUSHFD	
0069900E	F7D2	NOT EDX	
00699010	0FCA	BSWAP EDX	
00699012	EB 0F	JMP SHORT HprSnap5.00699023	
00699014	B9 EB0FB8EB	MOV ECX,EBB80FEB	
00699019	07	POP ES	
0069901A	B9 EB0F90EB	MOV ECX,EB900FEB	Modification of segment register
0069901F	08FD	OR CH,BH	
00699021	EB 08	JMP SHORT HprSnap5.0069902E	
00699023	F2:	PREFIX REPNE:	Superfluous prefix
00699024	EB F5	JMP SHORT HprSnap5.0069901B	
00699026	EB F6	JMP SHORT HprSnap5.0069901E	
00699028	F2:	PREFIX REPNE:	Superfluous prefix
00699029	EB 08	JMP SHORT HprSnap5.00699033	
0069902B	FD	STD	
0069902C	EB E9	JMP SHORT HprSnap5.00699017	
0069902E	F3:	PREFIX REP:	Superfluous prefix
0069902F	EB E4	JMP SHORT HprSnap5.00699015	
00699031	FC	CLD	
00699032	E9 9D0FC98B	JMP 8C329FD4	
00699037	CA F7D1	RETF 0D1F7	Far return
0069903A	59	POP ECX	
0069903B	58	POP EAX	
0069903C	50	PUSH EAX	
0069903D	51	PUSH ECX	
0069903E	0FCA	BSWAP EDX	
00699040	F7D2	NOT EDX	

Nry _Chay scripts to fix tem and Magic Jump to l :

```

var GetModuleHandleA
var AddressOfMagicJump
var LenOfMagicJump

GPA "GetModuleHandleA", "kernel32.dll"
mov GetModuleHandleA, $ RESULT

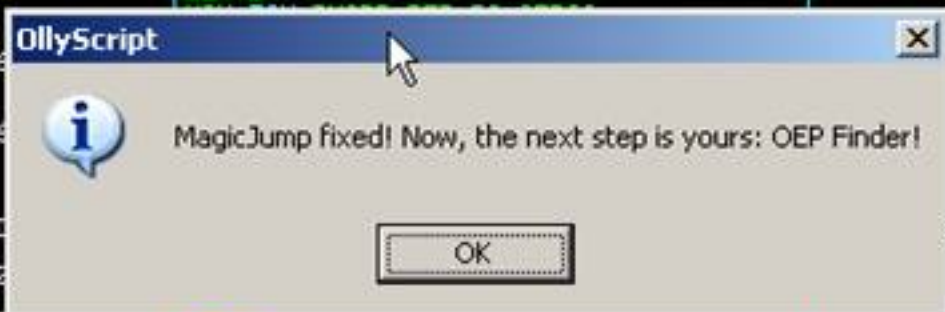
bphws GetModuleHandleA, "x"
repeat:
esto
rtu
find eip, # 0F84 ?????????????????????? 74 ?????????? EB? #
Cmp $ result, 0
je repeat
bphwc GetModuleHandleA

mov AddressOfMagicJump, $ RESULT
mov LenOfMagicJump, AddressOfMagicJump
add LenOfMagicJump, 2
mov LenOfMagicJump, [LenOfMagicJump]
inc LenOfMagicJump
mov [AddressOfMagicJump], 0E9
inc AddressOfMagicJump
mov [AddressOfMagicJump], LenOfMagicJump
CMT $ result, "<- MagicJump fixed"
msg "MagicJump fixed! Now, the next step is yours: OEP Finder!"
ret

```

_Sau When running the script:

Address	Hex dump	Disassembly	Comment
00E159A0	8800 6C50E400	MOV ECX,DWORD PTR DS:[E4006C]	
00E159A2	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00E159A5	A1 6C50E400	MOV EAX,DWORD PTR DS:[E4506C]	
00E159A8	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00E159AB	75 16	JNZ SHORT 00E15905	
00E159AF	8085 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
00E159C5	50	PUSH EAX	
00E159C6	FF15 B862E300	CALL DWORD PTR DS:[E362B8]	kernel32.LoadLibraryA
00E159CC	8800 6C50E400	MOV ECX,DWORD PTR DS:[E4506C]	
00E159D2	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00E159D5	A1 6C50E400	MOV EAX,DWORD PTR DS:[E4506C]	
00E159DA	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00E159DD	E9 30010000	JMP 00E15B12	<- MagicJump fixed
00E159E2	0033	ADD BYTE PTR DS:[EBX],DH	
00E159E4	C9	LEAVE	
00E159E5	8807		
00E159E7	3918		
00E159E9	74 06		
00E159EB	41		
00E159EC	83C0		
00E159EF	EB F6		
00E159F1	88D9		
00E159F3	C1E3		
00E159F6	53		
00E159F7	E8 D0		
00E159FC	880D		
00E15A02	89040E		
00E15A05	53		
00E15A06	E8 CDF00100	CALL 00E357D6	jmp to msvert.??2@YAPAXI0Z
00E15A08	59	POP ECX	
00E15A0C	59	POP ECX	
00E15A0D	8800 6850E400	MOV ECX,DWORD PTR DS:[E45068]	
00E15A13	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00E15A16	8807	MOV EAX,DWORD PTR DS:[EDI]	
00E15A19	8807		



CreateThread _Dat breakpoint, Shift + F9, Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
00E1BEFD	5F	POP EDI	
00E1BEFE	5E	POP ESI	
00E1BEFF	C9	LEAVE	
00E1BF00	C3	RETN	
00E1BF01	55	PUSH EBP	
00E1BF02	88EC	MOV EBP,ESP	
00E1BF04	81EC 28010000	SUB ESP,128	
00E1BF0A	56	PUSH ESI	
00E1BF0B	57	PUSH EDI	
00E1BF0C	BE E48DE300	MOV ESI,0E38DE4	ASCII "MainClass"
00E1BF11	808D D8FEFFFF	LEA EDI,DWORD PTR SS:[EBP-128]	
00E1BF17	A5	MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[E	
00E1BF18	A5	MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[E	
00E1BF19	66:A5	MOVS WORD PTR ES:[EDI],WORD PTR DS:[ESI	
00E1BF1B	6A 3D	PUSH 3D	
00E1BF1D	33C0	XOR EAX,EAX	
00E1BF1F	59	POP ECX	
00E1BF20	808D E2FEFFFF	LEA EDI,DWORD PTR SS:[EBP-11E]	
00E1BF26	F3:AB	REP STOS DWORD PTR ES:[EDI]	
00E1BF28	66:AB	STOS WORD PTR ES:[EDI]	
00E1BF2A	A1 2C56E400	MOV EAX,DWORD PTR DS:[E4562C]	
00E1BF2F	33FF	XOR EDI,EDI	
00E1BF31	38C7	CMP EAX,EDI	
00E1BF33	74 20	JE SHORT 00E1BF55	
00E1BF35	50	PUSH EAX	
00E1BF36	68 F8AAE300	PUSH 0E3AAF8	ASCII "%08X"
00E1BF3B	8085 D8FEFFFF	LEA EAX,DWORD PTR SS:[EBP-128]	
00E1BF41	57	PUSH EDI	
00E1BF42	50	PUSH EAX	
00E1BF43	FF15 1063E300	CALL DWORD PTR DS:[E36310]	msvert.stchr
00E1BF49	59	POP ECX	
00E1BF4A	59	POP ECX	
00E1BF4B	50	PUSH EAX	
00E1BF4C	FF15 0C63E300	CALL DWORD PTR DS:[E3630C]	msvert.sprintf

_Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
00E30232	59	POP ECX	kernel32.77E7BE2B
00E30233	BF 580AE400	MOV EDI,0E40A58	
00E30238	8BCF	MOV ECX,EDI	
00E3023A	E8 D07DFDFF	CALL 00E0800F	
00E3023F	84C0	TEST AL,AL	
00E30241	75 09	JNZ SHORT 00E3024C	
00E30243	6A 01	PUSH 1	
00E30245	8BCF	MOV ECX,EDI	
00E30247	E8 A6D3FDFF	CALL 00E0D5F2	
00E3024C	B9 C0FBE300	MOV ECX,0E3FBC0	
00E30251	C705 30C2E300 C	MOV DWORD PTR DS:[E3C230],0E3DEC0	
00E30258	E8 65550000	CALL 00E357C5	
00E30260	6A 00	PUSH 0	
00E30262	E8 5E550000	CALL 00E357C5	
00E30267	59	POP ECX	
00E30268	33C9	XOR ECX,ECX	
00E3026A	380D 7C10E400	CMP BYTE PTR DS:[E4107C],CL	
00E30270	75 39	JNZ SHORT 00E302AB	
00E30272	A1 A410E400	MOV EAX,DWORD PTR DS:[E410A4]	
00E30277	53	PUSH EBX	
00E30278	8B48 64	MOV ECX,DWORD PTR DS:[EAX+64]	
00E3027B	894D 08	MOV DWORD PTR SS:[EBP+8],ECX	
00E3027E	8B78 5C	MOV EDI,DWORD PTR DS:[EAX+5C]	
00E30281	3378 1C	XOR EDI,DWORD PTR DS:[EAX+1C]	
00E30284	8B98 8C000000	MOV EBX,DWORD PTR DS:[EAX+8C]	
00E3028A	3358 74	XOR EBX,DWORD PTR DS:[EAX+74]	
00E3028D	8D4D 08	LEA ECX,DWORD PTR SS:[EBP+8]	
00E30290	3378 10	XOR EDI,DWORD PTR DS:[EAX+10]	
00E30293	3358 34	XOR EBX,DWORD PTR DS:[EAX+34]	
00E30296	033D BC10E400	ADD EDI,DWORD PTR DS:[E410BC]	
00E3029C	E8 5F0DFDFF	CALL 00E01000	HprSnap5.00400000
00E302A1	33D2	XOR EDX,EDX	
00E302A3	F7F3	DIV EBX	
00E302A5	8B0C3A	MOV ECX,DWORD PTR DS:[EDX+ED1]	

_Dau Of OEP:

Address	Hex dump	Disassembly	Comment
00E302D5	8B5D 74	MOV EDX,DWORD PTR DS:[EAX+74]	
00E302D8	FF76 18	PUSH DWORD PTR DS:[ESI+18]	
00E302DB	335D 4C	XOR EDX,DWORD PTR DS:[EAX+4C]	
00E302DE	FF76 14	PUSH DWORD PTR DS:[ESI+14]	
00E302E1	335D 0C	XOR EDX,DWORD PTR DS:[EAX+0C]	
00E302E4	FF76 10	PUSH DWORD PTR DS:[ESI+10]	
00E302E7	2BCA	SUB ECX,EDX	
00E302E9	FFD1	CALL ECX	
00E302EB	EB 1D	JMP SHORT 00E3030A	
00E302ED	83FA 01	CMPL EDX,1	
00E302F0	75 1B	JNZ SHORT 00E3030D	
00E302F2	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
00E302F5	8B5D 74	MOV EDX,DWORD PTR DS:[EAX+74]	
00E302F8	335D 4C	XOR EDX,DWORD PTR DS:[EAX+4C]	
00E302FB	FF76 08	PUSH DWORD PTR DS:[ESI+8]	
00E302FE	335D 0C	XOR EDX,DWORD PTR DS:[EAX+0C]	
00E30301	6A 00	PUSH 0	
00E30303	FF76 0C	PUSH DWORD PTR DS:[ESI+0C]	
00E30306	2BCA	SUB ECX,EDX	
00E30308	FFD1	CALL ECX	
00E3030A	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
00E3030D	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00E30310	5F	POP EDI	
00E30311	5E	POP ESI	
00E30312	C9	LEAVE	
00E30313	C3	RETN	
00E30314	837C24 08 01	CMP DWORD PTR SS:[ESP+8],1	
00E30319	75 14	JNZ SHORT 00E3032F	
00E3031B	68 F0ACE400	PUSH 0E4ACF0	
00E30320	FF15 6C62E300	CALL DWORD PTR DS:[E3621C]	kernel32.InitializeCriticalSection
00E30326	8B4424 04	MOV EAX,DWORD PTR SS:[EAX+4]	
00E3032A	A3 B810E400	MOV DWORD PTR DS:[E410B8],EAX	
00E3032F	6A 01	PUSH 1	
00E30331	58	POP EAX	

_Dat Breakpoint in 00E30308 FFD1 CALL ECX, F9, F7: J OEP

Address	Hex dump	Disassembly	Comment
004EBF80	55	PUSH EBP	
004EBF81	8BEC	MOV EBP,ESP	
004EBF83	6A FF	PUSH -1	
004EBF85	68 78195E00	PUSH HprSnap5.005E1978	
004EBF8A	68 6CFE4E00	PUSH HprSnap5.004EFE6C	
004EBF8F	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
004EBF95	50	PUSH EAX	
004EBF96	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
004EBF9D	83EC 58	SUB ESP,58	
004EBFA0	53	PUSH EBX	
004EBFA1	56	PUSH ESI	
004EBFA2	57	PUSH EDI	
004EBFA3	9945 50	MOV DWORD PTR SS:[EBP-10],ESP	
004EBFA6	FF15 EC1CDA02	CALL DWORD PTR DS:[2DA1CEC]	kernel32.GetVersion
004EBFA7	33D2	XOR EDI,EDI	
004EBFAE	8A04	MOV DL,04	
004EBFB0	8915 0CBB6300	MOV DWORD PTR DS:[63BB0C],EDI	
004EBFB6	8BC8	MOV ECX,EAX	
004EBFB8	81E1 FF000000	AND ECX,0FF	
004EBFBE	8900 08BB6300	MOV DWORD PTR DS:[63BB08],ECX	
004EBFC4	C1E1 08	SHL ECX,8	
004EBFC7	03CA	ADD ECX,EDX	
004EBFC9	8900 04BB6300	MOV DWORD PTR DS:[63BB04],ECX	
004EBFCF	C1E8 10	SHR EAX,10	
004EBFD2	A3 00BB6300	MOV DWORD PTR DS:[63BB00],EAX	
004EBFD7	6A 01	PUSH 1	
004EBFD9	E8 A9530000	CALL HprSnap5.004F1387	
004EBFDE	59	POP ECX	
004EBFDF	85C0	TEST EAX,EAX	
004EBFE1	75 08	JNZ SHORT HprSnap5.004EBFEB	
004EBFE3	6A 1C	PUSH 1C	
004EBFES	E8 C3000000	CALL HprSnap5.004EC0AD	
004EBFEA	59	POP ECX	
004EBFEF	E8 4C3C0000	CALL HprSnap5.004EFC3C	

IAT Elimination

_Ta Tem need signs of Splicing Code, Alt + M:

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped a
00E70000	00001000	SHELL32		PE header	Image	R	RWE	
00E71000	001E0000	SHELL32	.text	code,import	Image	R	RWE	
01051000	0001C000	SHELL32	.data	data	Image	R	RWE	
0106D000	01B8A000	SHELL32	.rsrc	resources	Image	R	RWE	
02BF7000	0001A000	SHELL32	.reloc	relocations	Image	R	RWE	
02C20000	0001C000				Priv	RW	RW	
02C40000	000A4000				Priv	RW	RW	
02D03000	00002000				Priv	RW	RW	
02D1C000	00001000				Priv	RW	RW	
02D20000	00003000				Priv	RW	RW	
02D60000	00001000				Map	RW	RW	
02D70000	00001000				Map	RW	RW	
02D80000	00001000				Priv	RW	RW	
02DA1000	00001000				Priv	RW	RW	
02F80000	00003000				Priv	RW	RW	
02F90000	00010000				Priv	RW	RW	
03390000	00003000				Priv	RW	RW	
033D0000	00001000				Priv	RW	RW	
03450000	00003000				Priv	RW	RW	
03460000	00003000				Priv	RW	RW	
03470000	00003000				Priv	RW	RW	
03480000	00020000				Priv	RWE	RWE	
034A0000	00010000				Priv	RW	RW	
038A0000	00010000				Priv	RW	RW	
03CA0000	00010000				Priv	RW	RW	
040A0000	00001000				Priv	RW	RW	
040B0000	00010000				Priv	RW	RW	
044B0000	00003000				Priv	RW	RW	
044C0000	00010000				Priv	RW	RW	
048C0000	00003000				Priv	RW	RW	
048D0000	00010000				Priv	RW	RW	
04CD0000	00003000				Priv	RW	RW	
04CE0000	00010000				Priv	RW	RW	
050E0000	00003000				Priv	RW	RW	
051EE000	00002000				Priv	RW	RW	
1FBE0000	00001000	LTDLG10N		PE header	Image	R	RWE	
1FBE1000	00026000	LTDLG10N	.text	code	Image	R	RWE	
1FC07000	00001000	LTDLG10N	.rdata	exports	Image	R	RWE	
1FC08000	00009000	LTDLG10N	.data	data	Image	R	RWE	
1FC11000	00002000	LTDLG10N	.idata	imports	Image	R	RWE	

_Mo ArmInline and complete information about the Code Splicing:

00000598	ccApp	Norton AntiVirus	C:\Program Files\Common Files\Symantec Shared\c
000005C0	StarWind		C:\Program Files\Alcohol Soft\Alcohol 120\StarW
000005E8	wdfmgr		C:\WINDOWS\System32\wdfmgr.exe
00000724	ctfmon	TF_FloatingLangBar_WndTitl	C:\WINDOWS\System32\ctfmon.exe
00000778	UniKey	UniKey 3.62	D:\Soft need\Unikey 3.5\UniKey.exe
00000878	WINWORD	Zoom	C:\Program Files\Microsoft Office\OFFICE11\WINW
0000088C	SOUNDMAN	ALSMTray	C:\WINDOWS\SOUNDMAN.EXE
00000914	Explorer	SysFader	C:\WINDOWS\Explorer.EXE
00000978	TurboLau	TurboLaunch	C:\Program Files\TurboLaunch\TurboLaunch.exe
0000099C	Snagit32	Snagit Capture Preview	C:\Program Files\TechSmith\Snagit 7\Snagit32.ex
00000B00	HprSnap5		C:\Program Files\HyperSnap-DX 5\HprSnap5.exe
00000B64	msmsgs	ActiveMovieWindow	C:\Program Files\Messenger\msmsgs.exe

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped
003C0000	00002000				Map	R	R	
003D0000	00002000				Map	R	R	
003E0000	0000C000				Priv	RW	RW	
003F0000	00006000				Priv	RW	RW	
00400000	00001000	HprSnap5			Imag	R	RWE	
00401000	001CD000	HprSnap5	.text		Imag	R	RWE	
005CE000	00041000	HprSnap5	.rdata	exports	Imag	R	RWE	
0060F000	00039000	HprSnap5	.data	data	Imag	R	RWE	
00648000	00001000	HprSnap5	MY_DATA1		Imag	R	RWE	
00649000	00050000	HprSnap5	.text1	code	Imag	R	RWE	
00699000	00010000	HprSnap5	.adata	SFX	Imag	R	RWE	
006A9000	00020000	HprSnap5	.data1	imports	Imag	R	RWE	
006C9000	00140000	HprSnap5	.pdata		Imag	R	RWE	

(Slave) Process ID: 0x B00

Start Of Target Code: 0x 401000

Length Of Target Code: 0x 1CD000

_Dien Vro Slicing the Code:

Code Splicing

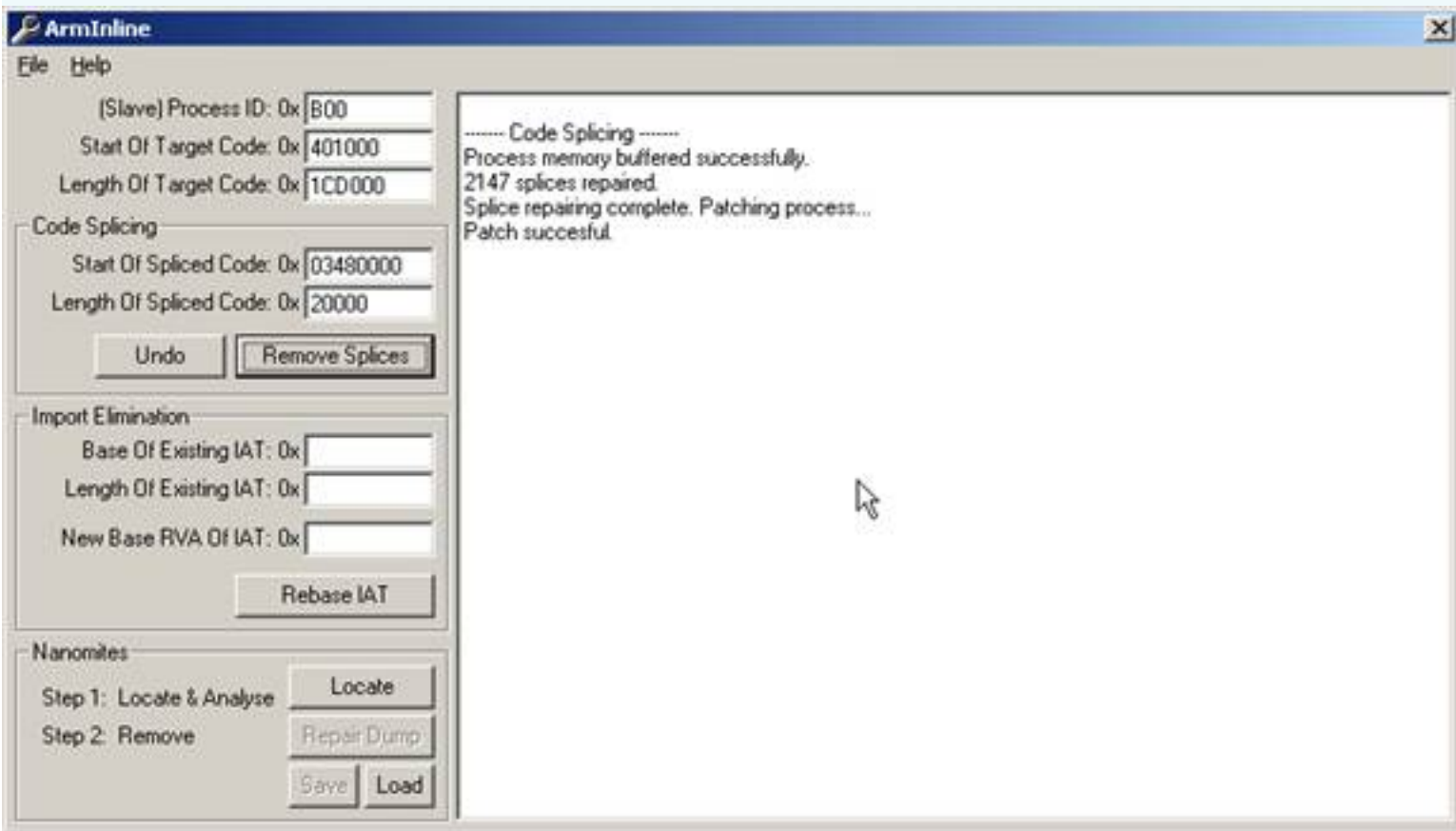
Start Of Spliced Code: 0x 03480000

Length Of Spliced Code: 0x 20000

Undo

Remove Splices

Remove _Nhan Splices:



IAT_Den the elimination:

IAT Start:

Address	Value	Comment
02DA1088	000701B5	
02DA108C	001C0702	
02DA1090	77D57828	USER32.CreateIconFromResourceEx
02DA1094	77E7E351	kernel32.SetHandleCount
02DA1098	77D8743F	USER32.CreateIconFromResource
02DA109C	77E6F767	kernel32.GlobalReAlloc
02DA10A0	77D6973C	USER32.MessageBeep
02DA10A4	77D48000	USER32.GetDlgItem
02DA10A8	77D59D1A	USER32.DefMDIChildProcA
02DA10AC	1FFF1910	USER32.ReleaseBitMap

IAT End:

Address	Value	Comment
02DA1F18	00000000	
02DA1F1C	00000000	
02DA1F20	00130013	
02DA1F24	00180702	
02DA1F28	74D3150E	oledlg.74D3150E
02DA1F2C	BAADF00D	
02DA1F30	BAADF00D	
02DA1F34	BAADF00D	
02DA1F38	BAADF00D	
02DA1F3C	BAADF00D	

_Ta Are:

OEP: 000EBF80

IATRVA: 00299000

IATSize: 00000E98

_Sections. ADATA:

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
003A0000	00003000				Map	R	R	\Device\Ha
003B0000	0000E000				Priv	RW	RW	
003C0000	00002000				Map	R	R	
003D0000	00002000				Map	R	R	
003E0000	0000C000				Priv	RW	RW	
003F0000	00006000				Priv	RW	RW	
00400000	00001000	HprSnap5			Imag	R	RWE	
00401000	001CD000	HprSnap5	.text		Imag	R	RWE	
005CE000	00041000	HprSnap5	.rdata	exports	Imag	R	RWE	
0060F000	00039000	HprSnap5	.data	data	Imag	R	RWE	
00648000	00001000	HprSnap5	MY_DATA1		Imag	R	RWE	
00649000	00050000	HprSnap5	.text1	code	Imag	R	RWE	
00699000	00010000	HprSnap5	.adatas	SFX	Imag	R	RWE	
006A9000	00020000	HprSnap5	.data1	imports	Imag	R	RWE	
006C9000	00140000	HprSnap5	.pdata		Imag	R	RWE	
00809000	00008000	HprSnap5	.rsrc	resources	Imag	R	RWE	
00820000	00005000				Map	R E	R E	
008E0000	00002000				Map	R E	R E	
008F0000	00103000				Map	R	R	

_Dien Vro ArmInline:

Import Elimination

Base Of Existing IAT: 0x 02DA1090

Length Of Existing IAT: 0x E98

New Base RVA Of IAT: 0x 00699000

Rebase IAT

_Nhan RebaseIAT:

ArmInline

File Help

(Slave) Process ID: 0x B00

Start Of Target Code: 0x 401000

Length Of Target Code: 0x 1CD000

Code Splicing

Start Of Spliced Code: 0x 03480000

Length Of Spliced Code: 0x 20000

Undo Remove Splices

Import Elimination

Base Of Existing IAT: 0x 02DA1090

Length Of Existing IAT: 0x E98

New Base RVA Of IAT: 0x 00699000

Rebase IAT

Nanomites

Step 1: Locate & Analyse Locate

Step 2: Remove Repair Dump

Save Load

----- Code Splicing -----

Process memory buffered successfully.

2147 splices repaired.

Splice repairing complete. Patching process...

Patch successful

----- Rebasing IAT -----

Process memory buffered successfully.

7590 DLL calls found total.

Analysing...

715 API functions referenced from 23 DLLs.

Redirecting DLL references:

7590 calls redirected total.

Patching process...

Process successfully patched.

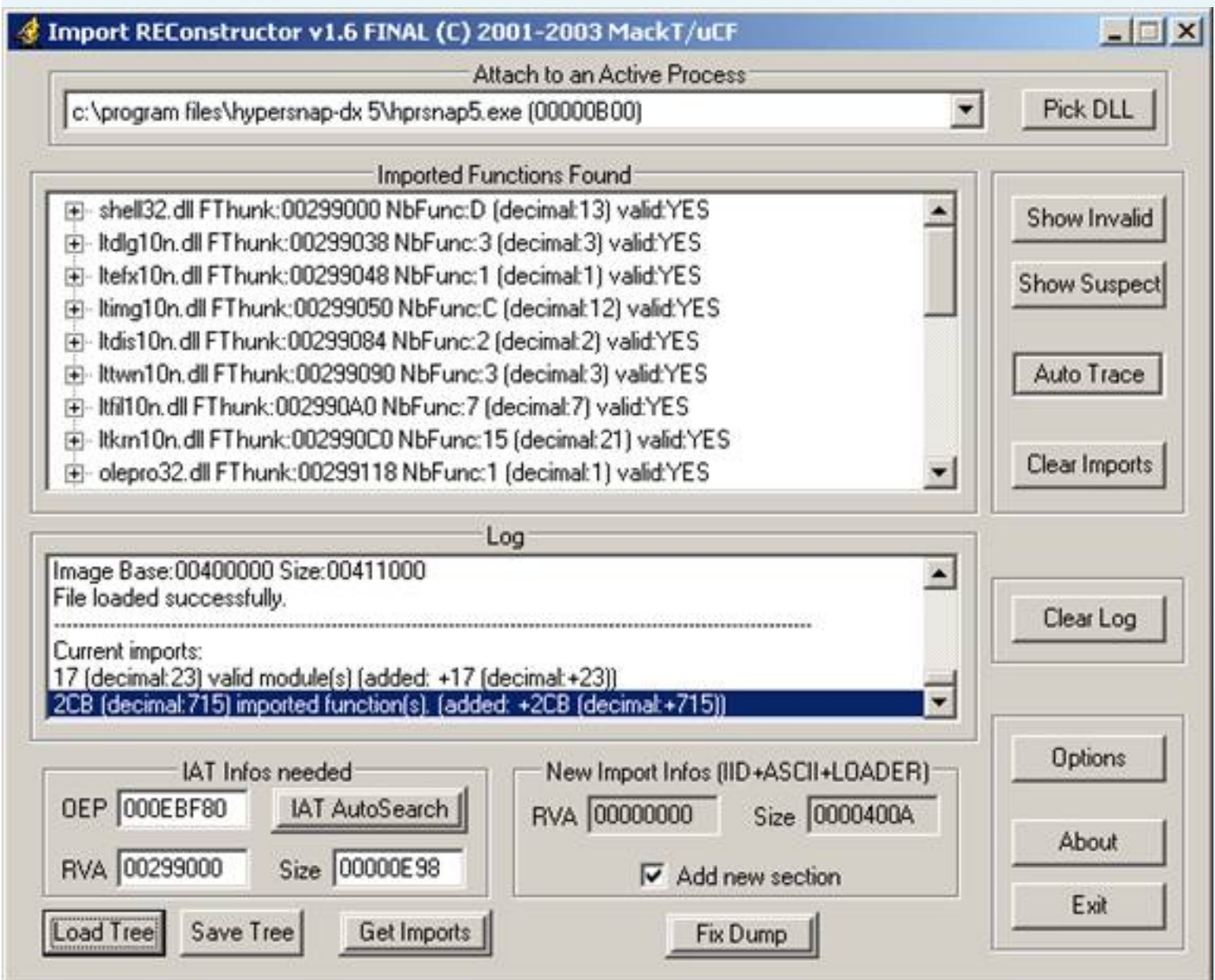
_Quay Back to the CPU:

Address	Hex dump	Disassembly	Comment
004EBF80	55	PUSH EBP	
004EBF81	88EC	MOV EBP,ESP	
004EBF83	6A FF	PUSH -1	
004EBF85	68 78195E00	PUSH HprSnap5.005E1978	
004EBF8A	68 6CFE4E00	PUSH HprSnap5.004EFE6C	
004EBF8F	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
004EBF95	50	PUSH EAX	
004EBF96	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
004EBF9D	83EC 58	SUB ESP,58	
004EBFA0	53	PUSH EBX	
004EBFA1	56	PUSH ESI	
004EBFA2	57	PUSH EDI	
004EBFA3	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
004EBFA6	FF15 8C9A6900	CALL DWORD PTR DS:[699A8C]	kernel32.GetVersion
004EBFA7	33D2	XOR EDX,EDX	
004EBFAE	8AD4	MOV DL,AH	
004EBFB0	8915 0CBB6300	MOV DWORD PTR DS:[63BB0C],EDX	
004EBFB6	8BC8	MOV ECX,EAX	
004EBFB8	81E1 FF000000	AND ECX,0FF	
004EBFBE	8900 08BB6300	MOV DWORD PTR DS:[63BB08],ECX	
004EBFC4	C1E1 08	SHL ECX,8	
004EBFC7	03CA	ADD ECX,EDX	
004EBFC9	8900 04BB6300	MOV DWORD PTR DS:[63BB04],ECX	
004EBFCF	C1E8 10	SHR EAX,10	
004EBFD2	A3 08BB6300	MOV DWORD PTR DS:[63BB00],EAX	
004EBFD7	6A 01	PUSH 1	
004EBFD9	E8 A9530000	CALL HprSnap5.004F1387	
004EBFDE	59	POP ECX	
004EBFDF	85C0	TEST EAX,EAX	
004EBFE1	75 08	JNZ SHORT HprSnap5.004EBFEB	
004EBFE3	6A 1C	PUSH 1C	
004EBFES	E8 C3000000	CALL HprSnap5.004EC0AD	
004EBFEB	59	POP ECX	
004EBFEB	E8 4C3C0000	CALL HprSnap5.004EFC3C	

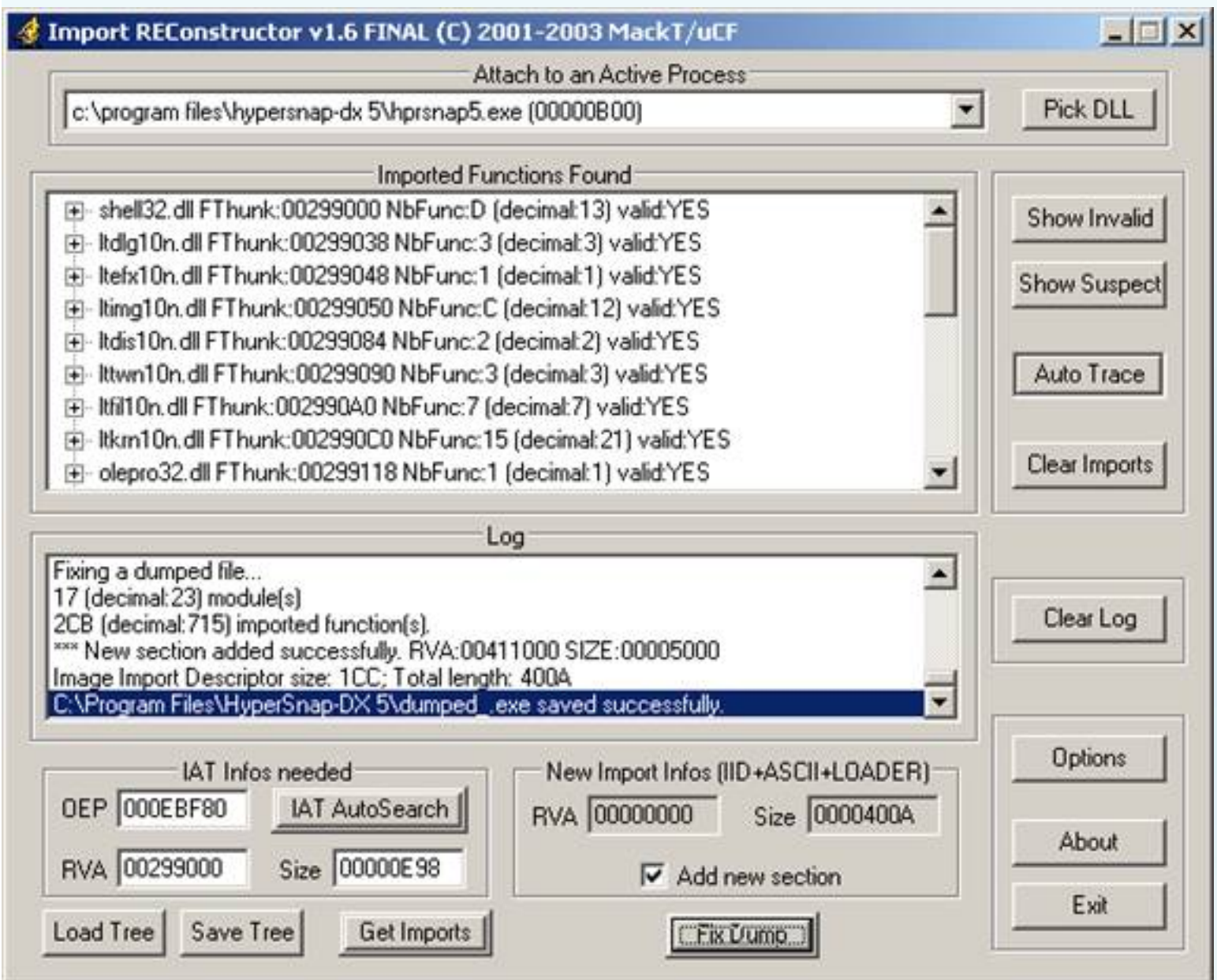
Full _LordPE dump:

c:\program files\hypersnap-d	dump full...	000	00411000
c:\program files\hypersnap-d	dump partial...	000	0001C000
Path	dump region...	Size	
c:\program files\hypersnap-d		000	
c:\windows\system32\ntdll.d	active dump engine	000	

_Mo ImpREC:



_FixDump:



Chay Try Dumped.exe



_OK, Has remove the time trial!

_Unpacked Successful!

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini, hoadongnoi, dqtn, hosiminh, Nilrem, fly, MaDMAAn_H3rCul3s, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RiF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx OllyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 07/10/2005)

Inline Patching Ap Document to PDF Converter v3.0.0

I. Software Introduction:

Homepage: <http://www.adultpdf.com/products/doctopdf/index.html>

Software: Ap Document to PDF Converter v3.0.0

Copyright by: Copyright 2000-2007 by adultpdf.com Inc. All rights reserved

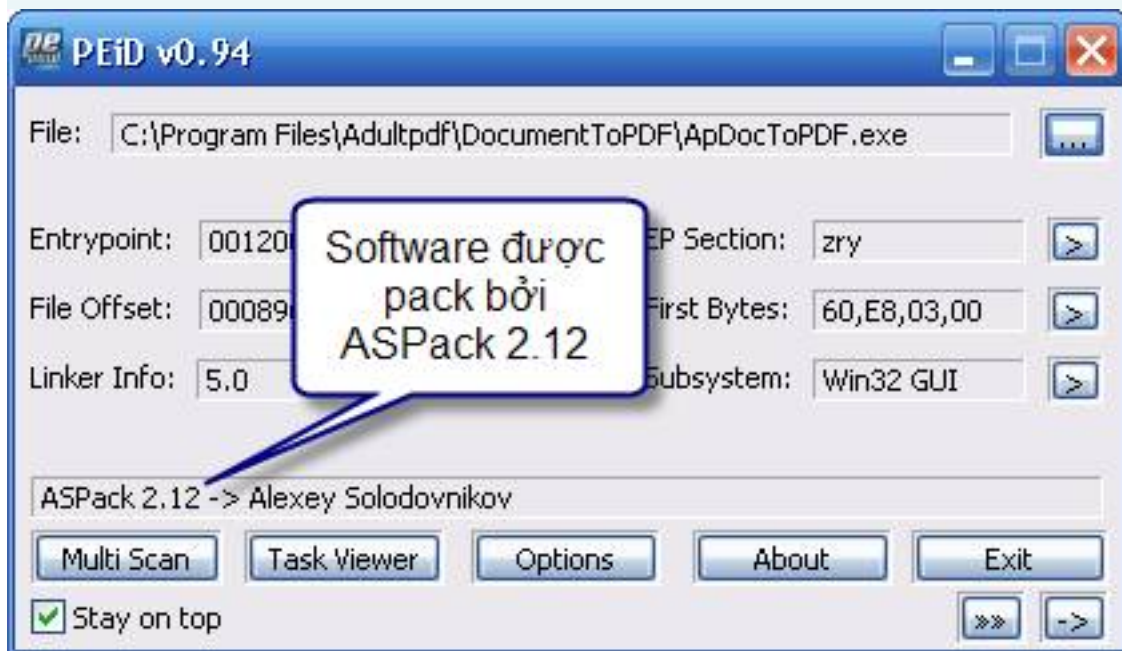
Crack tools: 1:10 OllyDbg, PEiD v 0.94

Information:

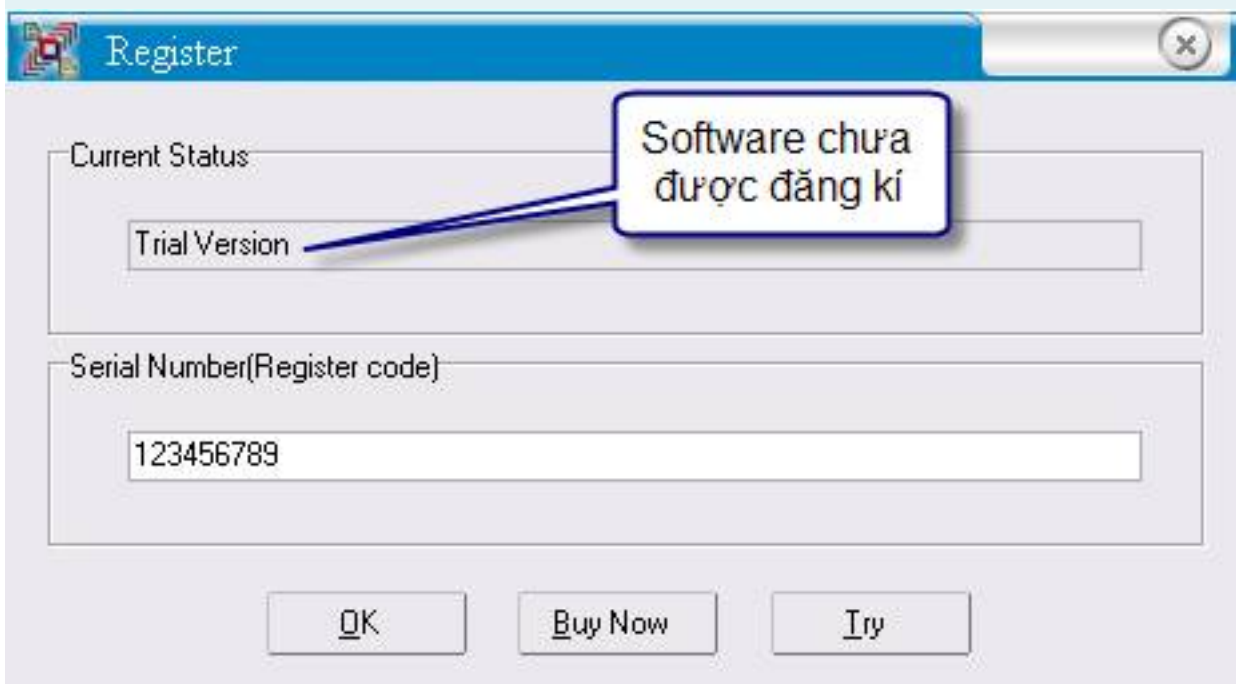
AP toPDF Document Converter is a powerful document to PDF converter, the converter allows you to convert over hundreds of windows printable documents, such as doc, xls, ppt, txt, pdf, html etc, to searchable PDF document. The converter can be used as a PDF writer or PDF creator from any application as soon as it supports printing. It does not need any software support, such as Adobe Acrobat, Acrobat Reader, etc..

II. Information:

We all know before you crack the software, you need to know which software is written in any language, there is not ... to pack from which to crack the method appropriate to help the crack is convenient and fast . For example, a software is written in Visual Basic SmartCheck we should use to debug or software that is written in .NET is not used OllyDbg to be treated. The software knows that pack / protect with packer is the information very important because it helps us Newbie orientation in the Manual Unpacking. There are many tools as possible as this task: PEiD, RDG Packer Detector. In this tut I use to scan files PEiD ApDocToPDF.exe:



Test program and Fake Serial: 123456789



Click OK, our results are:



III. Unpacking:

The unpack ASPack quite easily because it is a Packer, not a Protector. That means that in addition to the pack to make it smaller in size, ASPack not accompanied by the code or anti-debug destroyed IAT (Import Address Table) of the original. Now we use OllyDbg ApDocToPDF file to load, we encountered notified by Olly:



Click OK to ignore, we again encountered a message again:

Compressed code?



Quick statistical test of module 'ApDocToP' reports that its code section is either compressed, encrypted, or contains large amount of embedded data. Results of code analysis can be very unreliable or simply wrong. Do you want to continue analysis?

Yes

No

Click Yes to continue, we will come EP (Entry Point) program:

Address	Hex dump	Disassembly	Comment
00520001	60	PUSHAD	
00520002	E8 03000000	CALL ApDocToP.0052000A	
00520007	E9 EB045D45	JMP 45AF04F7	
0052000C	55	PUSH EBP	
0052000D	C3	RETN	
0052000E	E8 01000000	CALL ApDocToP.00520014	
00520013	EB 5D	JMP SHORT ApDocToP.00520072	
00520015	BB EDFFFFFF	MOV EBX,-13	
0052001A	03DD	ADD EBX,EBP	
0052001C	81EB 00001200	SUB EBX,120000	
00520022	83BD 22040000	CMP DWORD PTR SS:[EBP+422],0	
00520029	899D 22040000	MOV DWORD PTR SS:[EBP+422],EBX	
0052002F	0F85 65030000	JNZ ApDocToP.0052039A	
00520035	8D85 2E040000	LEA EAX,DWORD PTR SS:[EBP+42E]	
0052003B	50	PUSH EAX	
0052003C	FF95 4D0F0000	CALL DWORD PTR SS:[EBP+F4D]	
00520042	8985 26040000	MOV DWORD PTR SS:[EBP+426],EAX	
00520048	8BF8	MOV EDI,EAX	
0052004A	8D5D 5E	LEA EBX,DWORD PTR SS:[EBP+5E]	
0052004D	53	PUSH EBX	
0052004E	50	PUSH EAX	
0052004F	FF95 490F0000	CALL DWORD PTR SS:[EBP+F49]	
00520055	8985 4D050000	MOV DWORD PTR SS:[EBP+54D],EAX	
0052005B	8D5D 6B	LEA EBX,DWORD PTR SS:[EBP+6B]	
0052005F	53	PUSH EBX	

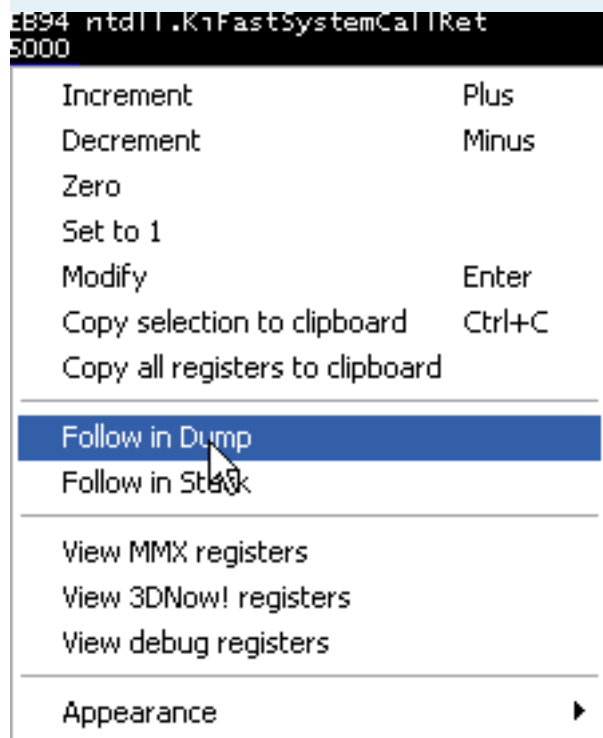
Press F8 and look at the window Registers:

Registers (FPU)	
EAX	00000000
ECX	0012FFB0
EDX	7C90EB94 ntdll.KiFastSystemCallRet
EBX	7FFD5000
ESP	0012FFA4
EBP	0012FFFO
ESI	FFFFFFFF
EDI	7C910738 ntdll.7C910738
EIP	00520002 ApDocToP.00520002

And Stack window:

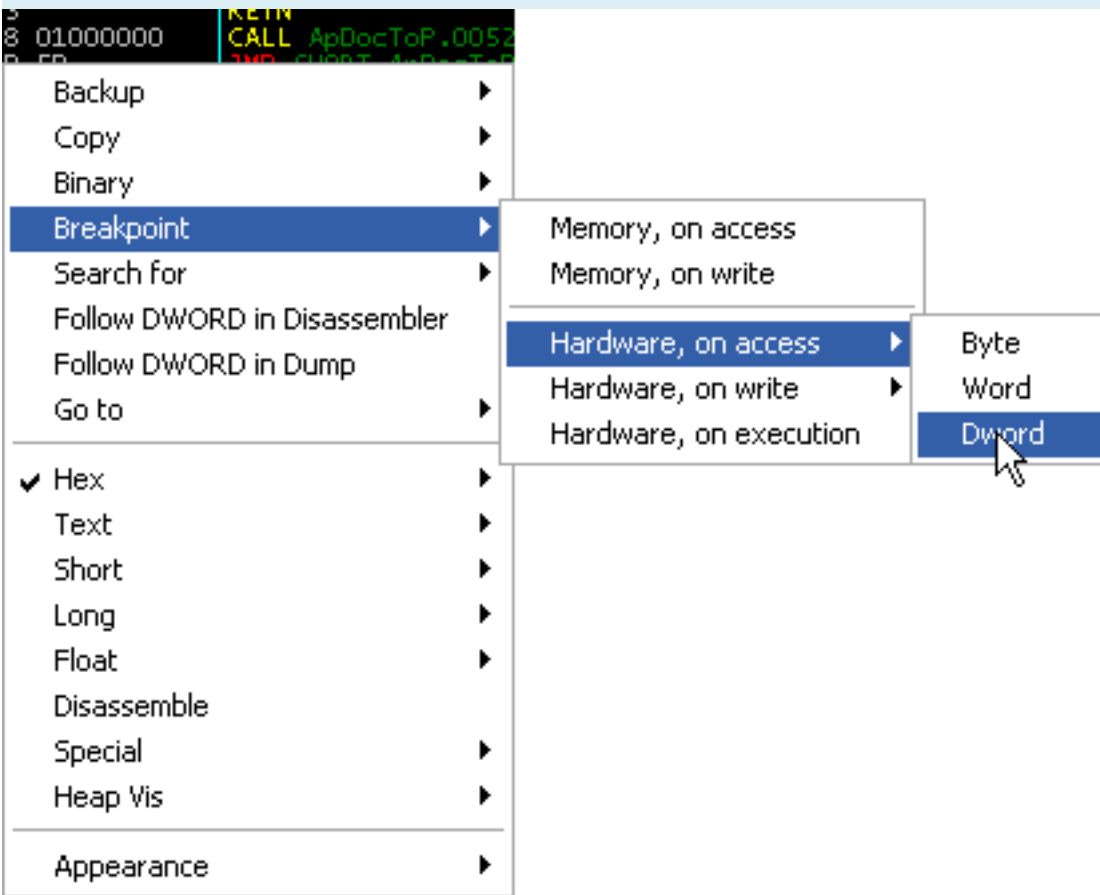
Address	Value	Comment
0012FFA4	7C910738	ntdll.7C910738
0012FFA8	FFFFFFFF	
0012FFAC	0012FFF0	
0012FFB0	0012FFC4	
0012FFB4	7FFD5000	
0012FFB8	7C90EB94	ntdll.KiFastSystemCallRet
0012FFBC	0012FFB0	
0012FFC0	00000000	
0012FFC4	7C816D4F	RETURN to kernel32.7C816D4F
0012FFC8	7C910738	ntdll.7C910738
0012FFCC	FFFFFFFF	
0012FFD0	7FFD5000	
0012FFD4	8054B038	
0012FFD8	0012FFC8	

Before the implementation of the code shall unpack the program in memory, the push all written to the stack for the purpose of preserving the bar record. Therefore, when done to unpack the program will restore all to write this. So to the OEP (Original Entry Point) we do the following: Click on the right to write ESP, Select Follow in dump:



At the dump window, highlight the first 4 bytes and Right Click, select the Hardware Breakpoint à , à On Access Dword:

38	07	91	7C	FF	FF	FF	FF
F0	FF	12	00	C4	FF	12	00
00	50	FD	7F	94	EB	90	7C
B0	FF	12	00	00	00	00	00



Press F9 to run, Olly break here:

Address	Hex dump	Disassembly	Comment
005203B0	75 08	JNZ SHORT ApDocToP.005203BA	
005203B2	B8 01000000	MOV EAX,1	
005203B7	C2 0C00	RETN 0C	
005203BA	68 88144000	PUSH ApDocToP.00401488	
005203BF	C3	RETN	
005203C0	8B85 26040000	MOV EAX,DWORD PTR SS:[EBP+426]	
005203C6	8D8D 3B040000	LEA ECX,DWORD PTR SS:[EBP+43B]	
005203CC	51	PUSH ECX	
005203CD	50	PUSH EAX	
005203CE	FF95 490F0000	CALL DWORD PTR SS:[EBP+F49]	
005203D4	8985 55050000	MOV DWORD PTR SS:[EBP+555],EAX	
005203DA	8D85 47040000	LEA EAX,DWORD PTR SS:[EBP+447]	
005203E0	50	PUSH EAX	
005203E1	FF95 510F0000	CALL DWORD PTR SS:[EBP+F51]	
005203E7	8985 2A040000	MOV DWORD PTR SS:[EBP+42A],EAX	
005203ED	8D8D 52040000	LEA ECX,DWORD PTR SS:[EBP+452]	
005203F3	51	PUSH ECX	

Continue pressing F8 three times more to be one of the OEP:

Address	Hex dump	Disassembly	Comment
00401488	EB 10	JMP SHORT ApDocToP.0040149A	
0040148A	66	DB 66	CHAR 'f'
0040148B	62	DB 62	CHAR 'b'
0040148C	3A	DB 3A	CHAR ':'
0040148D	43	DB 43	CHAR 'C'
0040148E	2B	DB 2B	CHAR '+'
0040148F	2B	DB 2B	CHAR '+'
00401490	48	DB 48	CHAR 'H'
00401491	4F	DB 4F	CHAR 'O'
00401492	4F	DB 4F	CHAR 'O'
00401493	4B	DB 4B	CHAR 'K'
00401494	90	NOP	
00401495	E9	DB E9	
00401496	98304C00	DD OFFSET ApDocToP.___CPPdebugHook	
0040149A	> A1 8B304C00	MOV EAX,DWORD PTR DS:[4C308B]	
0040149F	. C1E0 02	SHL EAX,2	
004014A2	. A3 8F304C00	MOV DWORD PTR DS:[4C308F],EAX	
004014A7	. 52	PUSH EDX	
004014A8	. 6A 00	PUSH 0	
004014AA	. E8 81060C00	CALL ApDocToP.004C1B30	[pModule = NULL GetModuleHandleA
004014AF	. 8BD0	MOV EDX,EAX	
004014B1	. E8 06250B00	CALL ApDocToP.004B39BC	
004014B6	. 5A	POP EDX	

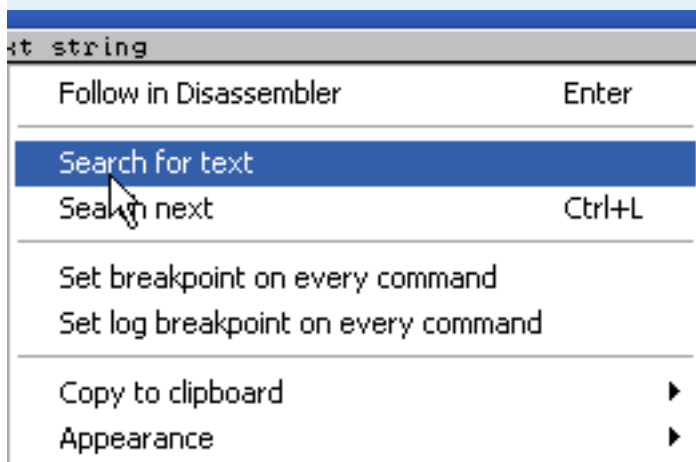
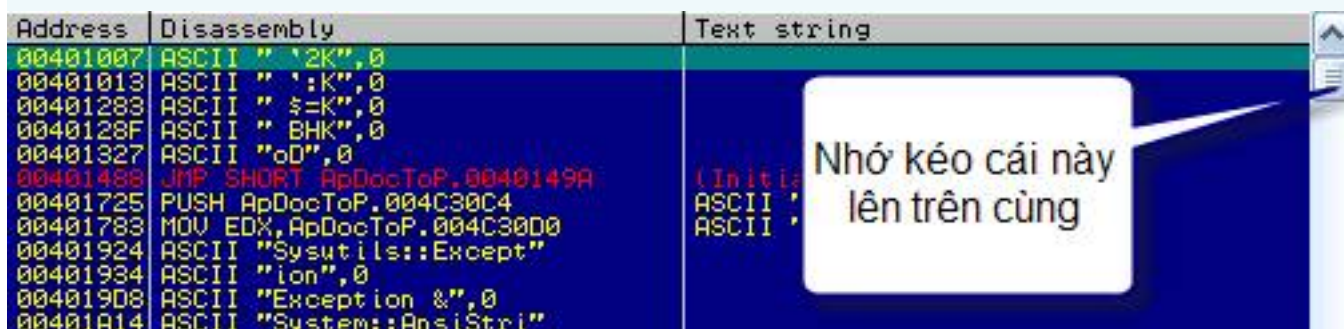
Now we can dump the file and fix IAT Hoanh to unpack the process. However, this article is about technical Inline Patching so I will not do so.

IV. Finding where to patch:

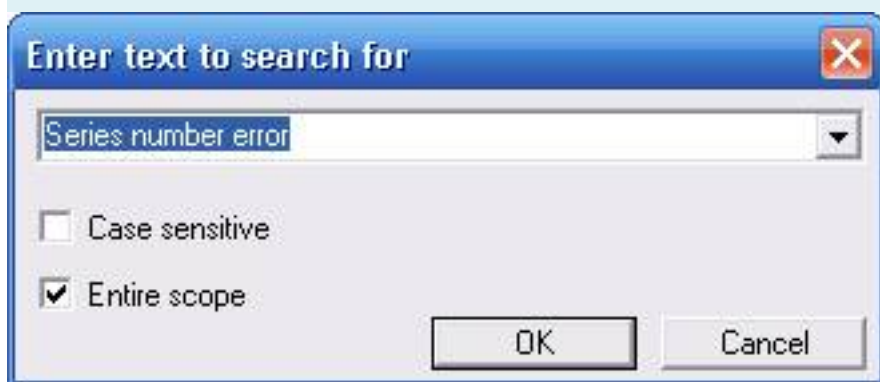
Continue, we will need to find patch. Right-click in the Disassembly window and select Search for All referenced text strings:

Follow in Dump	
Search for	Name (label) in current module Ctrl+N
Find references to	Name in all modules
View	
Copy to executable	Command Ctrl+F
Analysis	Sequence of commands Ctrl+S
Analyze This!	Constant
	Binary string Ctrl+B
Asm2Clipboard	All intermodular calls
Bookmark	All commands
ExtraCopy	All sequences
Dump debugged process	All constants
Heap Vis	All switches
Make dump of process	All referenced text strings
Ultra String Reference	User-defined label
Appearance	User-defined comment

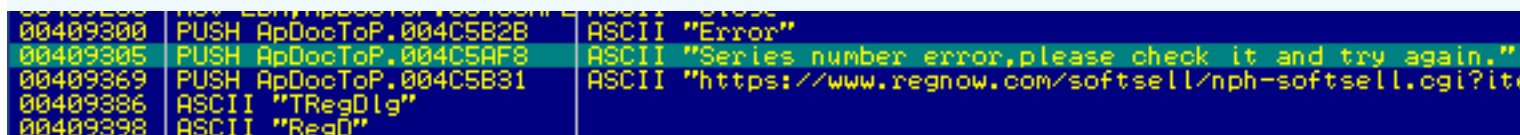
In the window string Text: Right Click, select Search for text:



Enter: "Series number error"



Click OK to me for this:



Double click on the line is to highlight it here:

004092FC	EB 35	JMP SHORT ApDocToP.00409333	
004092FE	6A 10	PUSH 10	
00409300	68 2B5B4C00	PUSH ApDocToP.004C5B2B	ASCII "Error"
00409305	68 F85A4C00	PUSH ApDocToP.004C5AF8	ASCII "Series number error,"
0040930A	8BC3	MOV EAX,EBX	
0040930C	E8 8B4A0800	CALL ApDocToP.0048DD9C	
00409311	50	PUSH EAX	hOwner
00409312	E8 E18F0B00	CALL ApDocToP.004C22F8	MessageBoxA
00409317	FF4D F0	DEC [LOCAL.4]	

Scroll to the top is a bit far:

Gọi hàm check Serial		PUSH ECX	Arg2
		PUSH EBX	Arg1
		CALL ApDocToP.00409188	ApDocToP.00409188
		ADD ESP,8	
		CMP AL,1	
		JNZ ApDocToP.00409188	
		PUSH 40	
		PUSH ApDocToP.004C5B2B	ASCII "Registered Version"
		PUSH ApDocToP.004C5B65	ASCII "Thank you register Ap"
		MOV EAX,EBX	
		CALL ApDocToP.0048DD9C	
		PUSH EAX	hOwner
		CALL ApDocToP.004C22F8	MessageBoxA
		LEA EBX,[LOCAL.127]	

Nhảy tới badboy

Address của hàm
check serial

Good boy

We see here if the value AL other 1, the command will jump to JNZ badboy, conversely will call goodboy. So, make sure check function Serial address at 00,409,188 (on my computer, your computer may be different). To check the serial functions we highlight the "Call ApDocToP.00409188" and press Enter or click the keyboard Ctrl + G and then enter the address of the function check serial is 00409188, we will be here to:

Address	Hex dump	Disassembly	Comment
00409188	55	PUSH EBP	
00409189	8BEC	MOV EBP,ESP	
0040918B	53	PUSH EBX	
0040918C	56	PUSH ESI	
0040918D	57	PUSH EDI	
0040918E	8B5D 0C	MOV EBX,[ARG.2]	
00409191	85DB	TEST EBX,EBX	
00409193	74 0C	JE SHORT ApDocToP.004091A1	
00409195	53	PUSH EBX	
00409196	E8 E5AD0A00	CALL ApDocToP.004B3F80	

What we need is the implementation of this function, the value of AL must be 1. There are many different patch to achieve the purpose, for I will be streamlined patch at the beginning of this function as follows:

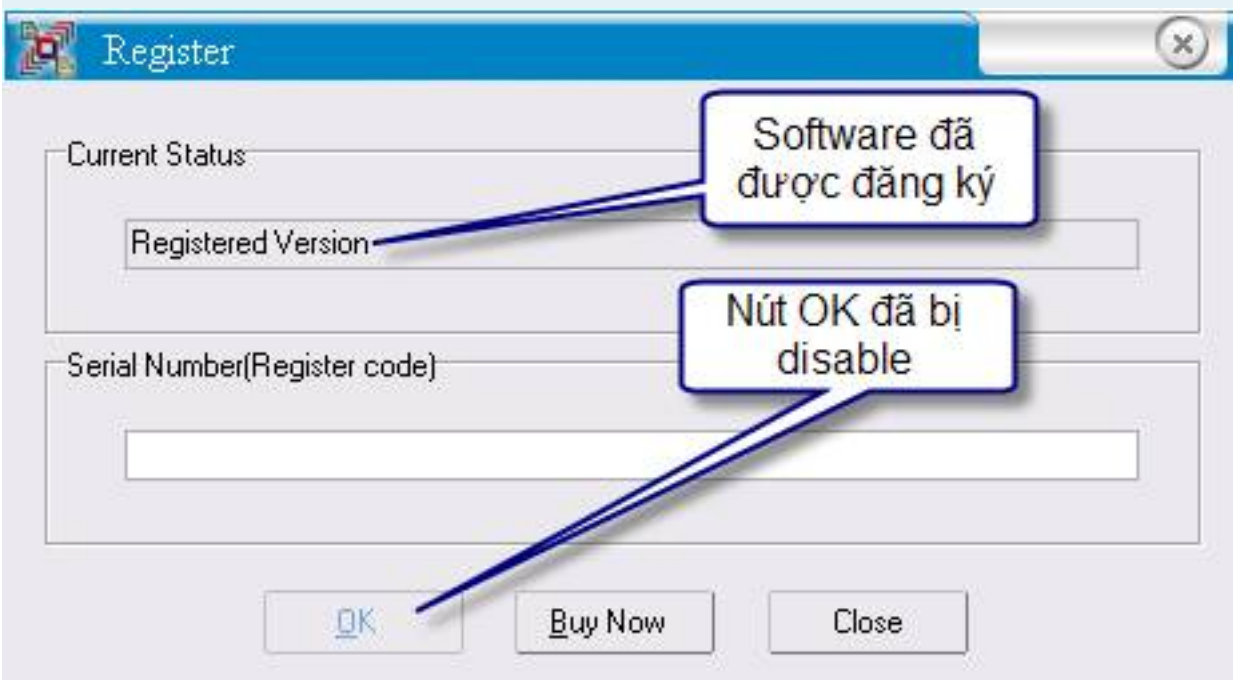
```
MOV AL, 1
RETN
```


00409187	58	NOT
00409188	B0 01	MOV AL,1
0040918A	C3	RETN
0040918B	53	PUSH EBX
0040918C	56	PUSH ESI
0040918D	57	PUSH EDI

Note: To edit the code you want to highlight the command line and click edit spacebar or can use a plugin is very useful NonaWrite. However, tut this is quite simply that I do not use it. Now we run the test program to try, in OllyDbg to Debug menu and then select the Hardware breakpoint:



After deleting Hardware Breakpoint, press F9 to run the program. The program will skip the window to register. On the Help menu, select register ...



Therefore, we need to inline patch three bytes at address 00409188 from 55 to 8B EC B0 01 C3.

You remember the value of this little will to use it in the Inline patching.

V. Inline Patching:

Press Ctrl + F2 to restart the program. Steps Unpacking the same until coming here:

Address	Hex dump	Disassembly	Comment
005203B0	75 08	JNZ SHORT ApDocToP.005203BA	
005203B2	B8 01000000	MOV EAX,1	
005203B7	C2 0C00	RETN 0C	
005203BA	68 88144000	PUSH ApDocToP.00401488	
005203BF	C3	RETN	
005203C0	8B85 26040000	MOV EAX,DWORD PTR SS:[EBP+426]	
005203C6	8D8D 3B040000	LEA ECX,DWORD PTR SS:[EBP+43B]	
005203CC	51	PUSH ECX	
005203CD	50	PUSH EAX	
005203CE	FF95 490F0000	CALL DWORD PTR SS:[EBP+F49]	
005203D4	8985 55050000	MOV DWORD PTR SS:[EBP+555],EAX	
005203DA	8D85 47040000	LEA EAX,DWORD PTR SS:[EBP+447]	
005203E0	50	PUSH EAX	
005203E1	FF95 510F0000	CALL DWORD PTR SS:[EBP+F51]	
005203E7	8985 2A040000	MOV DWORD PTR SS:[EBP+42A],EAX	
005203ED	8D8D 52040000	LEA ECX,DWORD PTR SS:[EBP+452]	
005203F3	51	PUSH ECX	

To now unpack the code has completed the process in memory unpack. For two command line that follows:

PUSH ApDocToP.00401488

RETN

They mean equivalent to the command:

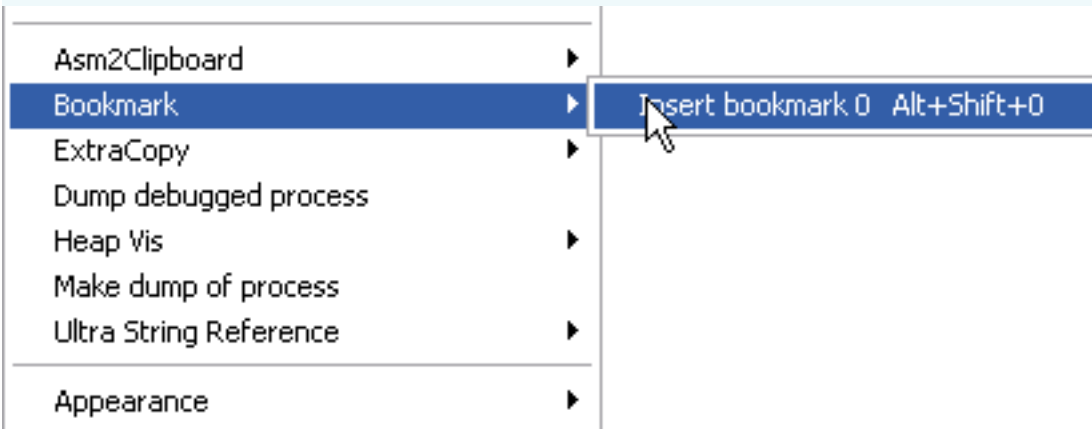
JMP ApDocToP.00401488

That is to jump to the OEP program. Address of the command line:

PUSH ApDocToP.00401488 is **005203BA**. You remember the bit to little more we will use to inject the code. Before you jump to OEP of the program will perform inline patch. First, we need a code space to work, so the code is called code cave. In the window Disassembly we move down the search until the code is appropriate, I choose to start code at **00521235**:

00521231	0000	ADD BYTE PTR DS:[EAX],AL	
00521233	0000	ADD BYTE PTR DS:[EAX],AL	
00521235	0000	ADD BYTE PTR DS:[EAX],AL	
00521237	0000	ADD BYTE PTR DS:[EAX],AL	
00521239	0000	ADD BYTE PTR DS:[EAX],AL	
0052123B	0000	ADD BYTE PTR DS:[EAX],AL	
0052123D	0000	ADD BYTE PTR DS:[EAX],AL	
0052123F	0000	ADD BYTE PTR DS:[EAX],AL	
00521241	0000	ADD BYTE PTR DS:[EAX],AL	
00521243	0000	ADD BYTE PTR DS:[EAX],AL	
00521245	0000	ADD BYTE PTR DS:[EAX],AL	
00521247	0000	ADD BYTE PTR DS:[EAX],AL	
00521249	0000	ADD BYTE PTR DS:[EAX],AL	
0052124B	0000	ADD BYTE PTR DS:[EAX],AL	
0052124D	0000	ADD BYTE PTR DS:[EAX],AL	

Check back for sure, right click on the desired address mark and then select Insert bookmark
Bookmark à 0

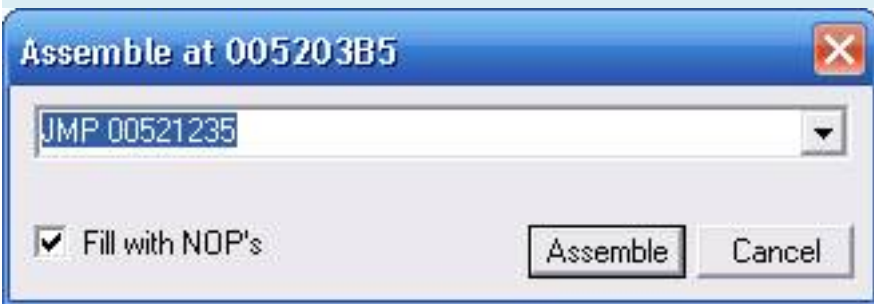


Note: you need a plugin Bookmark to perform this function.

Press * to return to where the break is, from the patch

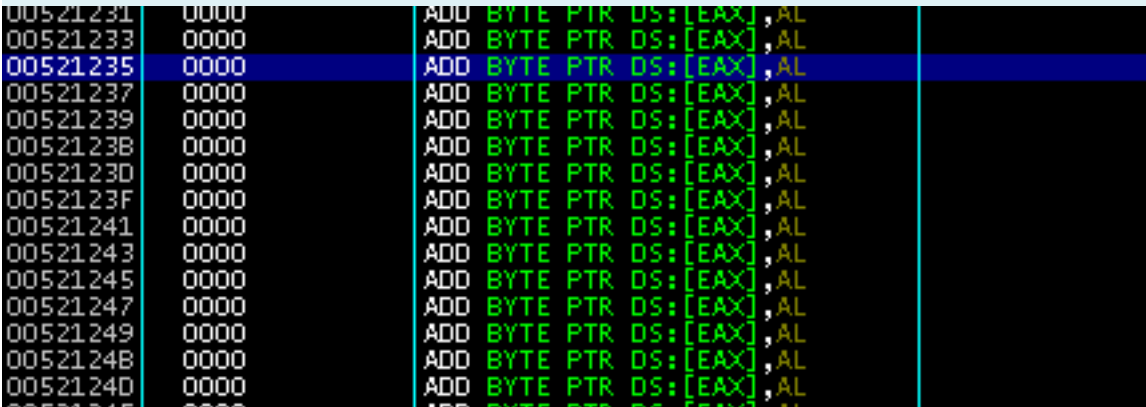
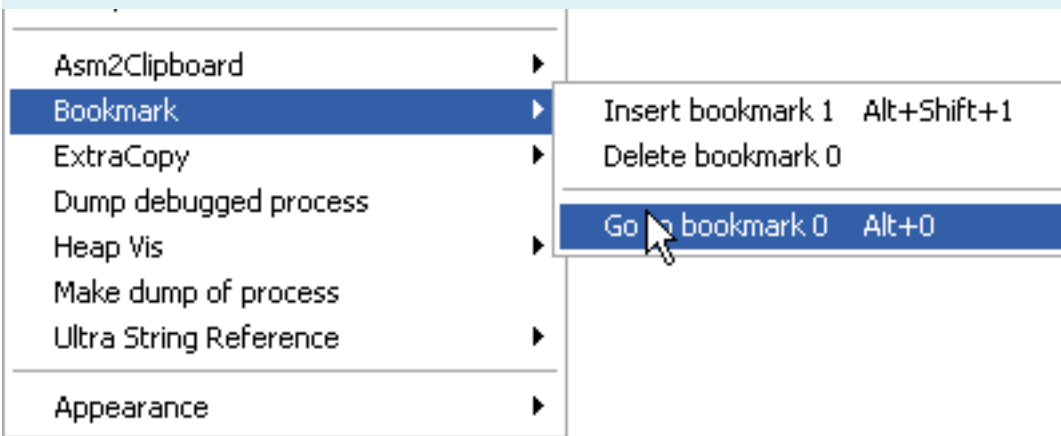
JNZ SHORT 005203BA

the **JMP 00521235**

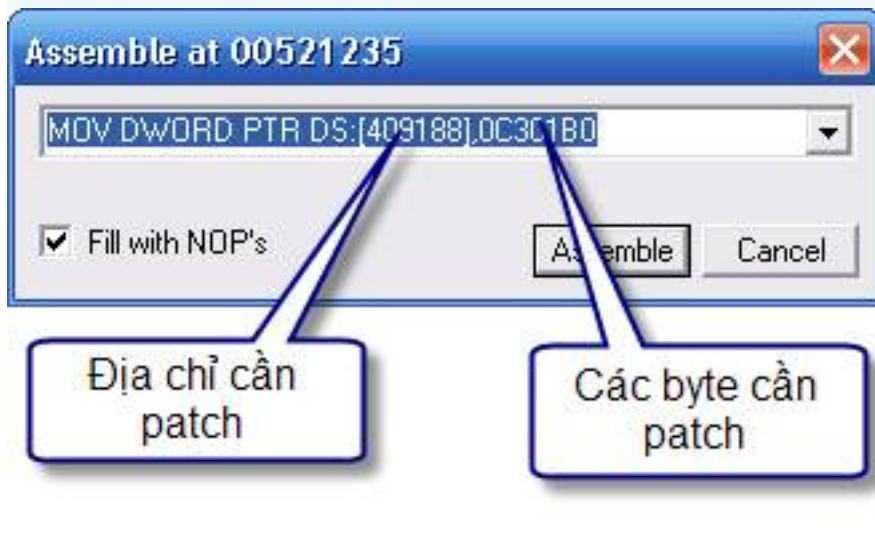


Sau khi patch

Back code our cave by: Disassembly window, right click and choose Bookmark à Go to bookmark 0



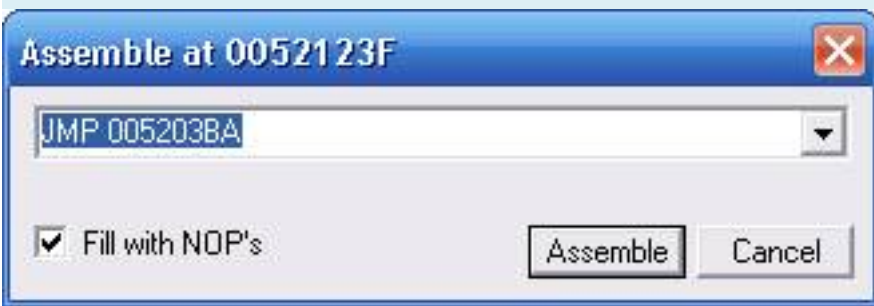
Press Spacebar to edit code, we start to inject code:



You also remember the values that I ask all of you remember it? First is the address of the function and check the serial **00409188** bytes we want to overwrite in this address is **B0 01 C3**. To overwrite the value in this address **00409188** we use:

MOV DWORD PTR DS: [409188], 0C301B0

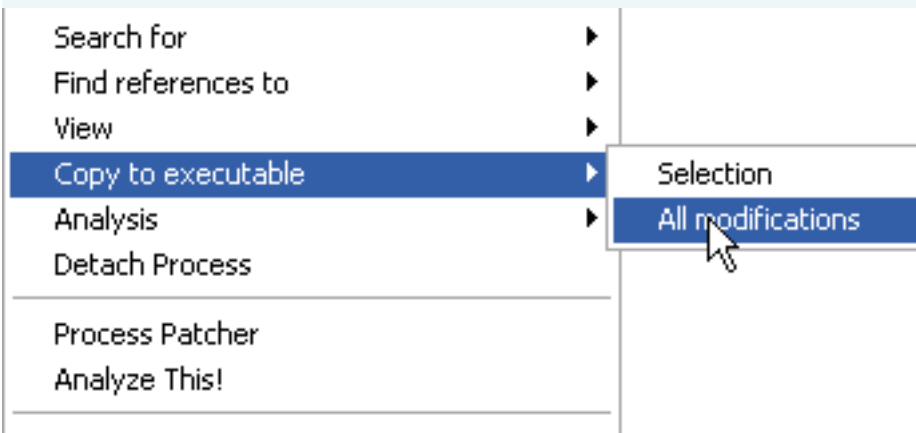
After the implementation of inline patching we need to address **005203BA** from which to jump to the OEP and perform tasks that it must do:



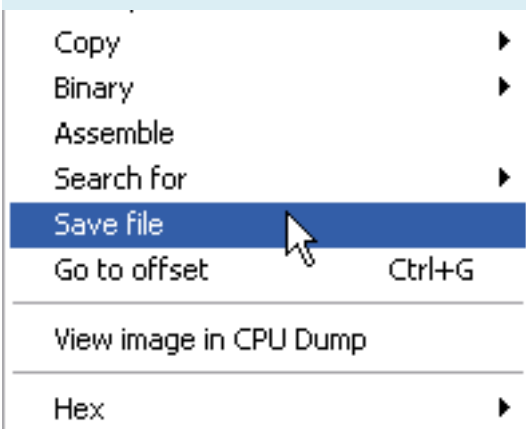
Results we have:

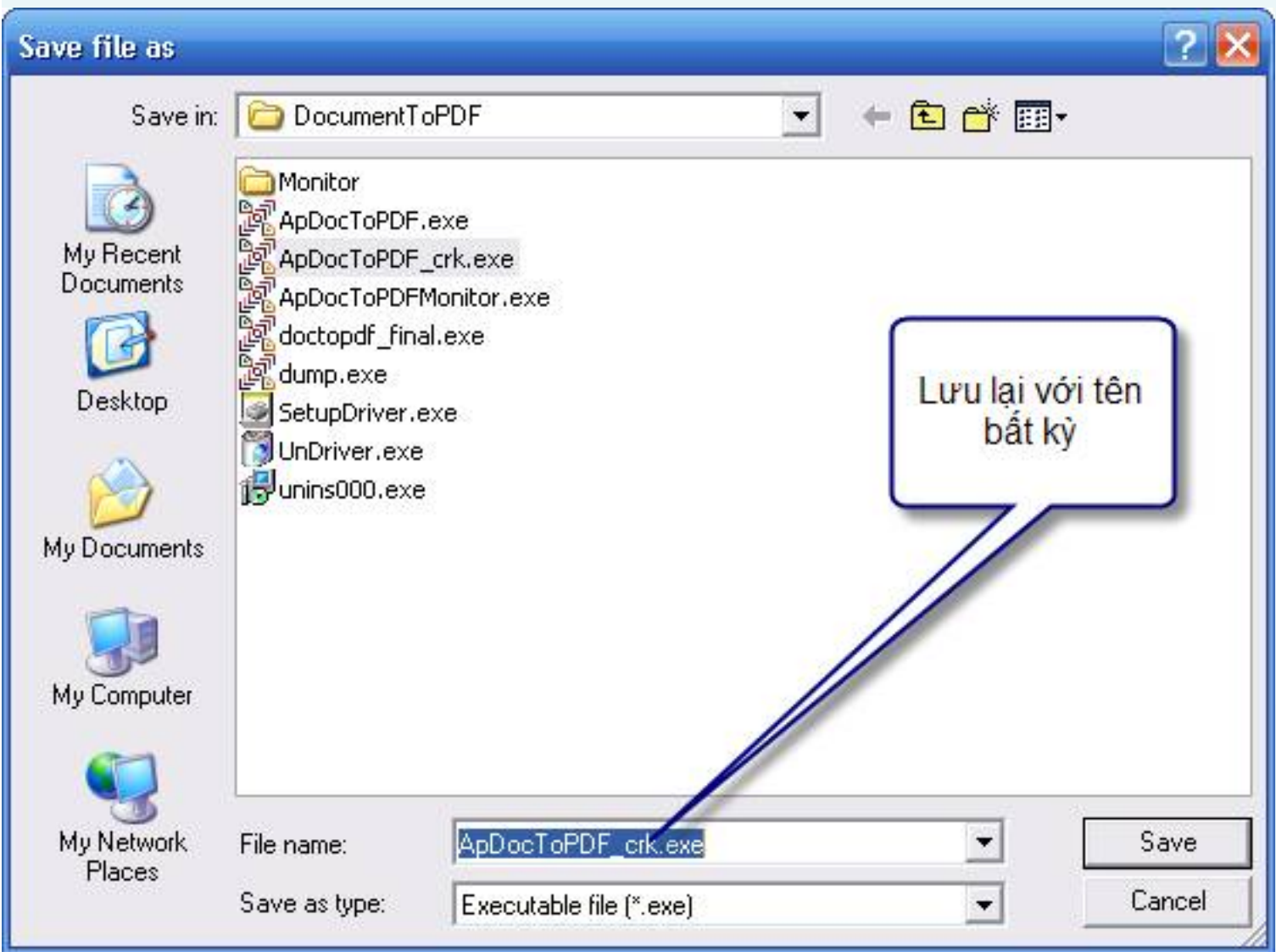
00521233	0000	ADD BYTE PTR DS:[EAX],AL
00521235	C705 88914000	MOV DWORD PTR DS:[409188],0C301B0
0052123F	E9 76F1FFFF	JMP AnDocToP,005203BA
00521244	90	NOP
00521245	0000	ADD BYTE PTR DS:[EAX],AL
00521247	0000	ADD BYTE PTR DS:[EAX],AL
00521249	0000	ADD BYTE PTR DS:[EAX],AL

Save the changes in our ways: in the Disassembly window, Right Click and select Copy to All modifications à excutable



In the new window on the Right Click, select Save File





VII. Testing:

Back to Olly, press F8 twice will come:

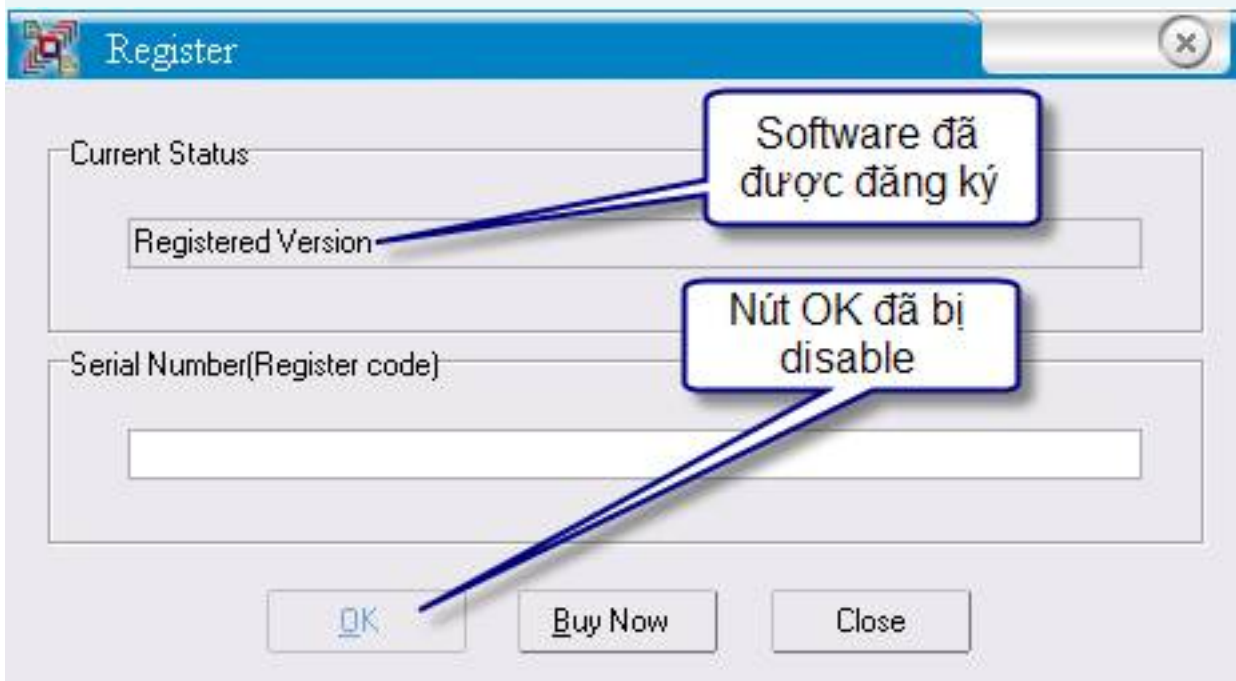
00521235	C705 88914000	MOV DWORD PTR DS:[409188],0C301B0
0052123F	E9 76F1FFFF	JMP ApDocToP.005203BA
00521244	90	NOP
00521245	0000	ADD BYTE PTR DS:[EAX],AL
00521247	0000	ADD BYTE PTR DS:[EAX],AL

That is just done command: **MOV DWORD PTR DS: [409188], 0C301B0**

Press the key combination Ctrl + G to type address **00409188** function is to check serial:

Address	Hex dump	Disassembly
00409188	B0 01	MOV AL,1
0040918A	C3	RETN
0040918B	0056 57	ADD BYTE PTR DS:[ESI+57],0C
0040918E	8B5D 0C	MOV EBX,DWORD PTR SS:[EBP+C]
00409191	85DB	TEST EBX,EBX
00409193	74 0C	JE SHORT ApDocToP.004091A1

We see results as you like. Press F9 to run (remember to remove Hardware Breakpoint), then on the Help menu, select the register:



Now we've done technical Inline Patching then that. Wishing you happy.

VIII. The end:

This is the first hand-tut's what they should have errors, please ignore all the while only in the wrong place.

Special thanks to Hacnho, MoonBaby, Benina, Kienmanowar and RongChauA who've written tuts very good for newbies (Continue in the future?).

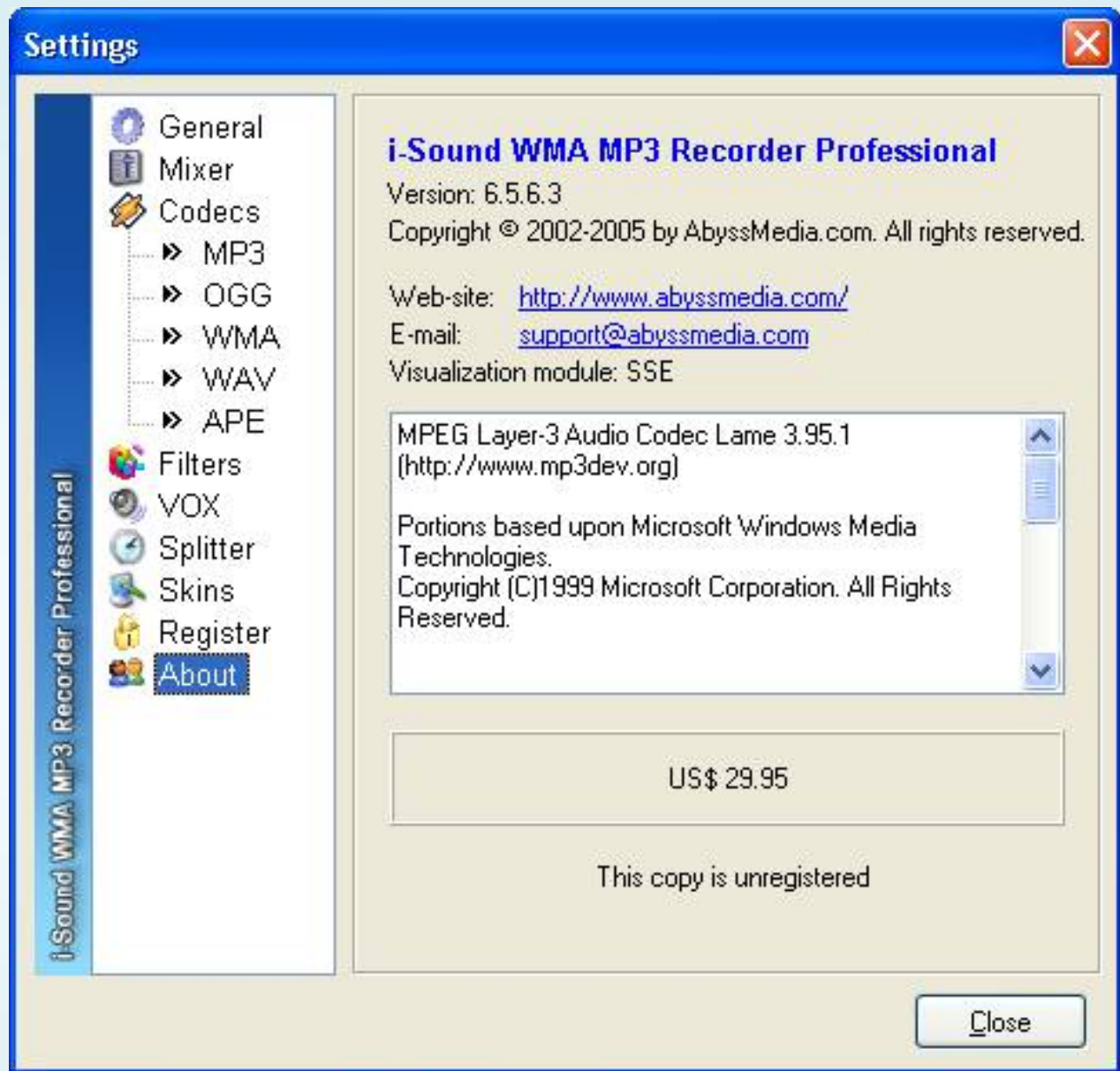
Thanks to all REA's member.

And of course I want to thank you for read it.

ERROR

7 / 6 / 2007

Inline Patching use APE + Dup 2:10 for UPX



_He He, greeting relatives, I'm walking this voice training for the company going I have a karaoke contest in the hands of gold should I make a soft sound to get the mic with the trees

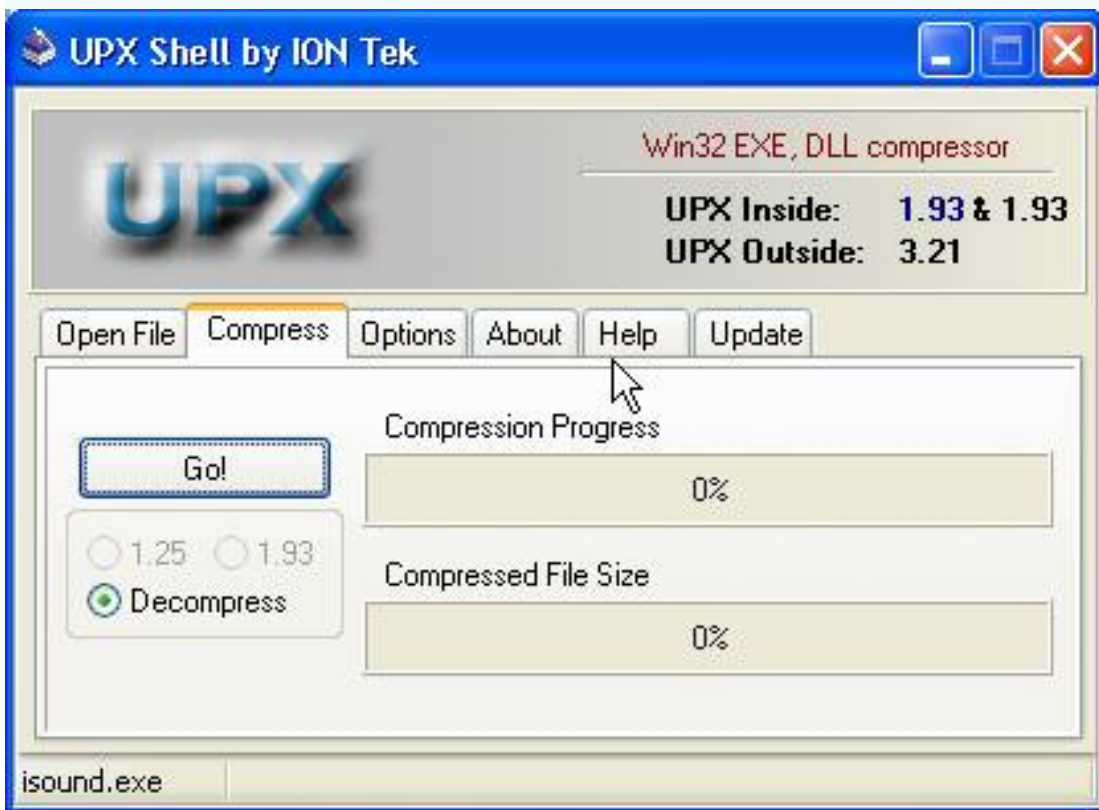
and out guitare team's 3 J . When this soft down, I forget beng finding lech crack the glass cracking it. Seeing this is a problem that little brother, we should mention I do mạn allowed the tut this. Make sure her child is also clear is what inline patching. Unlike normal patching files on the file. Packed files have three months to crack, one we unpack it and then finished it meat, is it we MUP to OEP, looking for signs of patch, then we create a loader, the third is how we live patch to file packed. In this way the three months to patch directly packed file is the most pro. Why? First, we avoid the large size release accompanied by unpacked.cracked. file, as the second we avoid the situation Cmp J and a reason it is ... lười J .

_Nguyen Principles of patching Inline (or a term is Code Injection redirection or Code) is we need to find placements patch, then we create a partition on the memory process. Remember this region include several billion as CRC check patch, patch code, several orders jump to OEP. Then we redirect to the real OEP program a jump command. From here we proceed patch target, then we redirect the area of memory. Since the area is sure to jump to the EP's packer. Argentina is conduct, and many other cases. Today health than many we had Ape inline quite a patcher or Ap0x support a variety packer, regret that not only support for ASProtect L and Armadillo. In this article I will illustrate how to use the patcher this target with a simple packer is UPX 1.93w!

_Truoc All we detect target:



_Unpack:



_Chuong The written Borlan Delphi7:



_Cracking:

```

:004D0A0A A1E01D4E00      mov eax, dword ptr [004E1DE0]
:004D0A0F 8B00                mov eax, dword ptr [eax]
:004D0A11 E816530000          call 004D5D2C
:004D0A16 84C0                test al, al
:004D0A18 7512                jne 004D0A2C

* Possible StringData Ref from Code Obj ->"This copy is unregistered"
|
:004D0A1A BA740E4D00          mov edx, 004D0E74
:004D0A1F 8B83E4040000        mov eax, dword ptr [ebx+000004E4]
:004D0A25 E8E2A2F9FF          call 0046AD0C
:004D0A2A EB6E                jmp 004D0A9A

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:004D0A18(C)
|
-y:004D0A2C 8B0D141D4E00      mov ecx, dword ptr [004E1D14]
:004D0A32 8B490D                mov ecx, dword ptr [ecx+0D]
:004D0A35 8D8598FEFFFF      lea eax, dword ptr [ebp+FFFFFFE98]

* Possible StringData Ref from Code Obj ->"Registered to "
|
:004D0A3B BA980E4D00          mov edx, 004D0E98
:004D0A40 E8AB49F3FF          call 004053F0
:004D0A45 8B9598FEFFFF      mov edx, dword ptr [ebp+FFFFFFE98]
:004D0A4B 8B83E4040000        mov eax, dword ptr [ebx+000004E4]
:004D0A51 E8B6A2F9FF          call 0046AD0C
:004D0A56 33D2                xor edx, edx
:004D0A58 8B83AC040000        mov eax, dword ptr [ebx+000004AC]
:004D0A5E E899A1F9FF          call 0046ABFC
:004D0A63 33D2                xor edx, edx
:004D0A65 8B83C4040000        mov eax, dword ptr [ebx+000004C4]
:004D0A6B E88CA1F9FF          call 0046ABFC

```

_Patch To:

```

:004D0A0A A1E01D4E00      mov eax, dword ptr [004E1DE0]
:004D0A0F 8B00                mov eax, dword ptr [eax]
:004D0A11 E816530000          call 004D5D2C
:004D0A16 84C0                test al, al
:004D0A18 EB12                jmp 004D0A2C

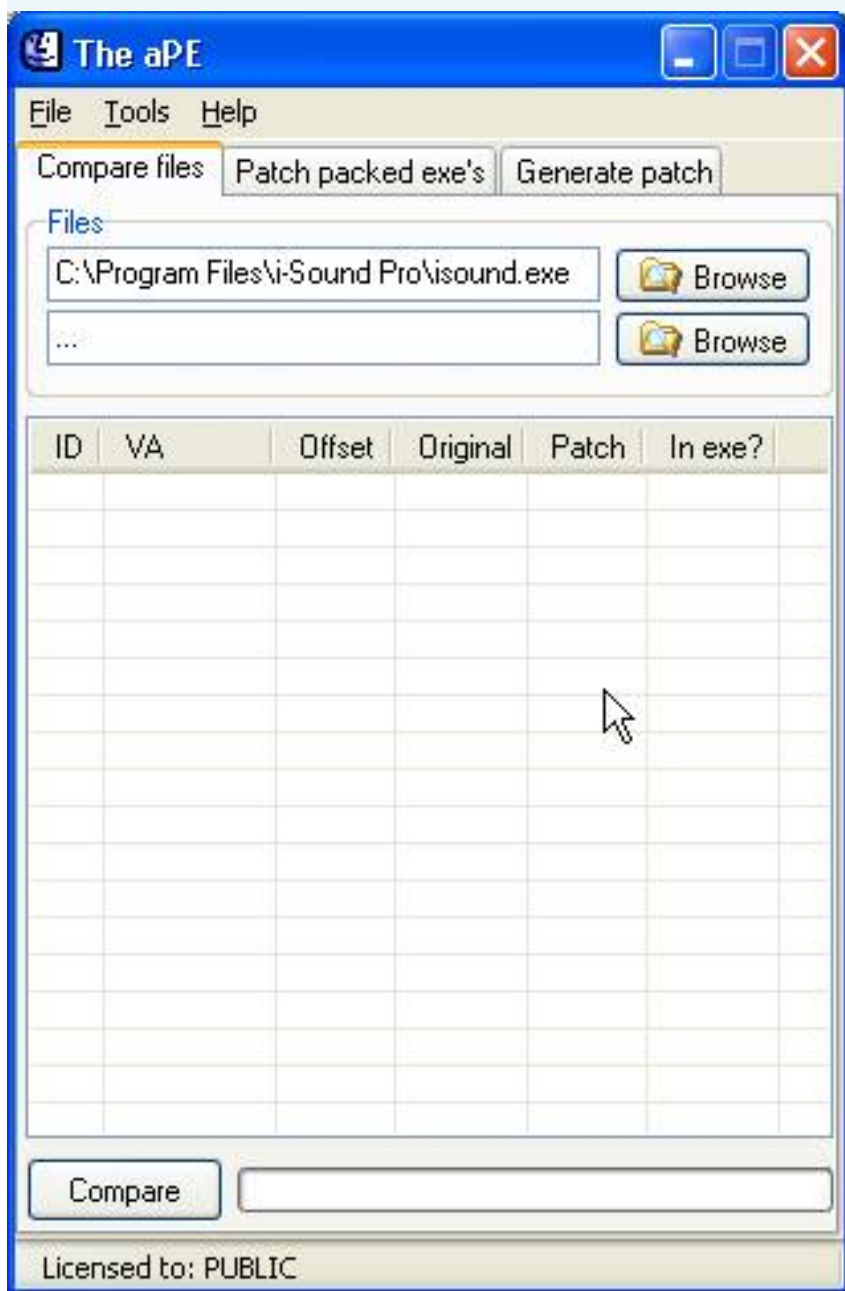
* Possible StringData Ref from Code Obj ->"This copy is unregistered"
|
:004D0A1A BA740E4D00          mov edx, 004D0E74
:004D0A1F 8B83E4040000        mov eax, dword ptr [ebx+000004E4]
:004D0A25 E8E2A2F9FF          call 0046AD0C
:004D0A2A EB6E                jmp 004D0A9A

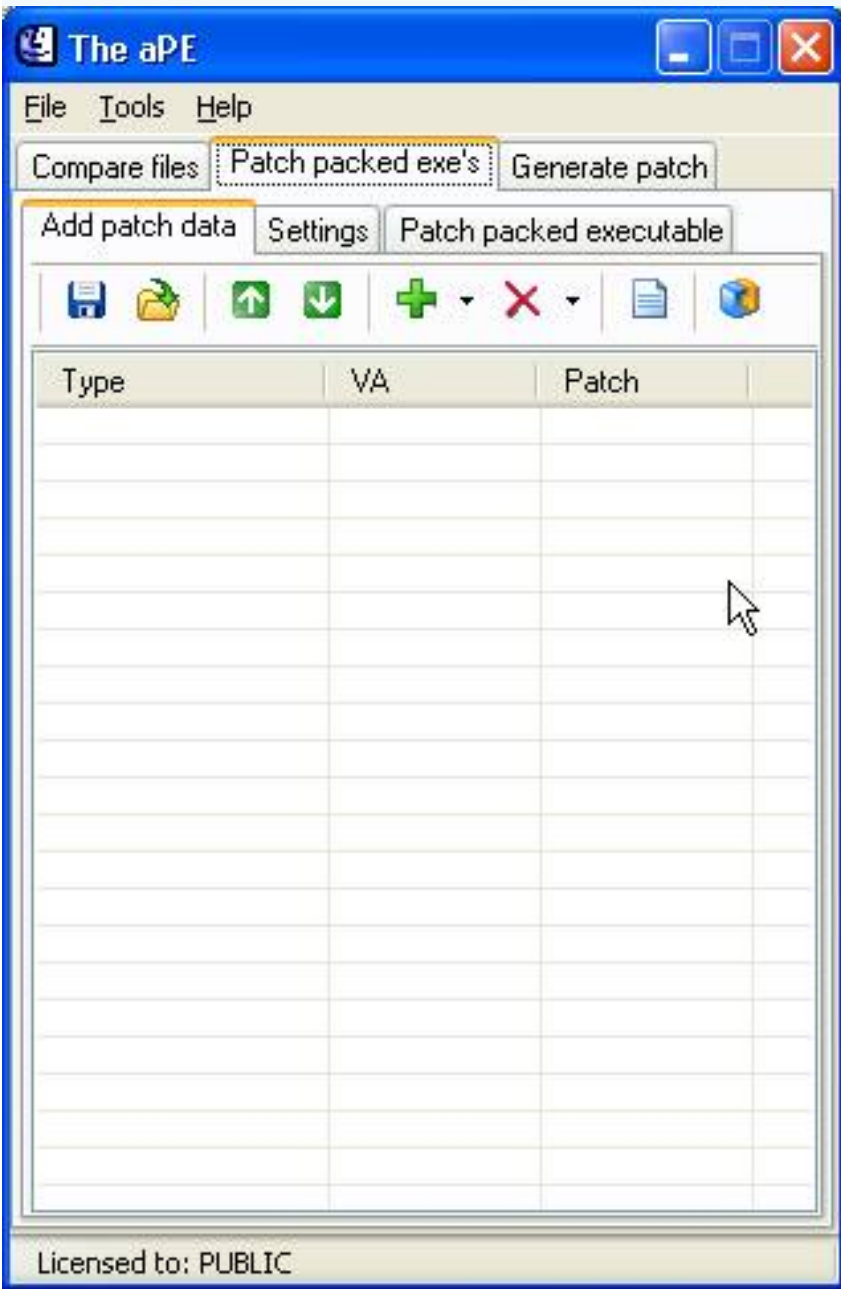
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:004D0A18(U)
|
-y:004D0A2C 8B0D141D4E00      mov ecx, dword ptr [004E1D14]
:004D0A32 8B490D                mov ecx, dword ptr [ecx+0D]
:004D0A35 8D8598FEFFFF      lea eax, dword ptr [ebp+FFFFFFE98]

* Possible StringData Ref from Code Obj ->"Registered to "
|
:004D0A3B BA980E4D00          mov edx, 004D0E98
:004D0A40 E8AB49F3FF          call 004053F0
:004D0A45 8B9598FEFFFF      mov edx, dword ptr [ebp+FFFFFFE98]
:004D0A4B 8B83E4040000        mov eax, dword ptr [ebx+000004E4]
:004D0A51 E8B6A2F9FF          call 0046AD0C
:004D0A56 33D2                xor edx, edx
:004D0A58 8B83AC040000        mov eax, dword ptr [ebx+000004AC]
:004D0A5E E899A1F9FF          call 0046ABFC
:004D0A63 33D2                xor edx, edx
:004D0A65 8B83C4040000        mov eax, dword ptr [ebx+000004C4]
:004D0A6B E88CA1F9FF          call 0046ABFC

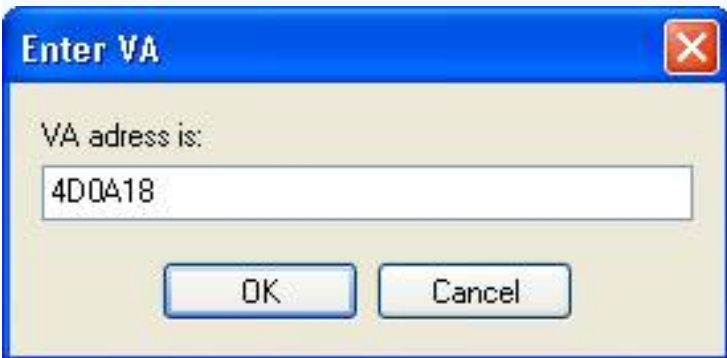
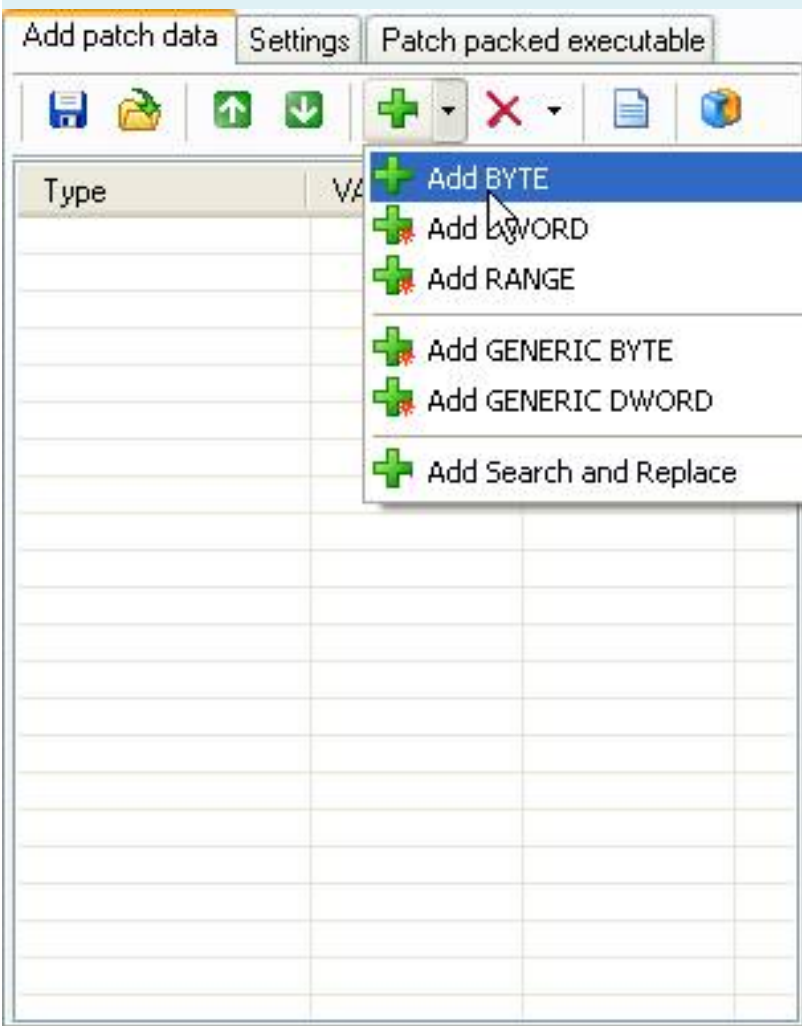
```

_Do File is unpacked. The problem is we need to patch the packed file. Ok, let's open up aPE, the tab packed exe's Patch:

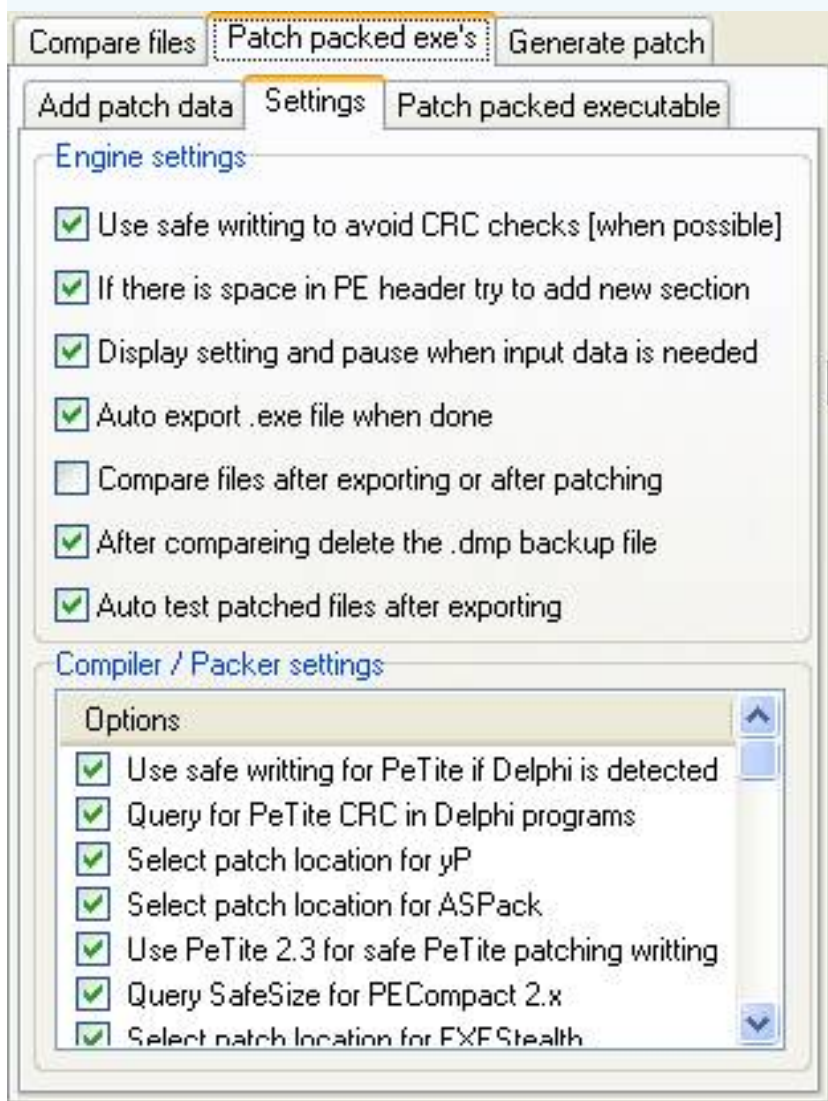




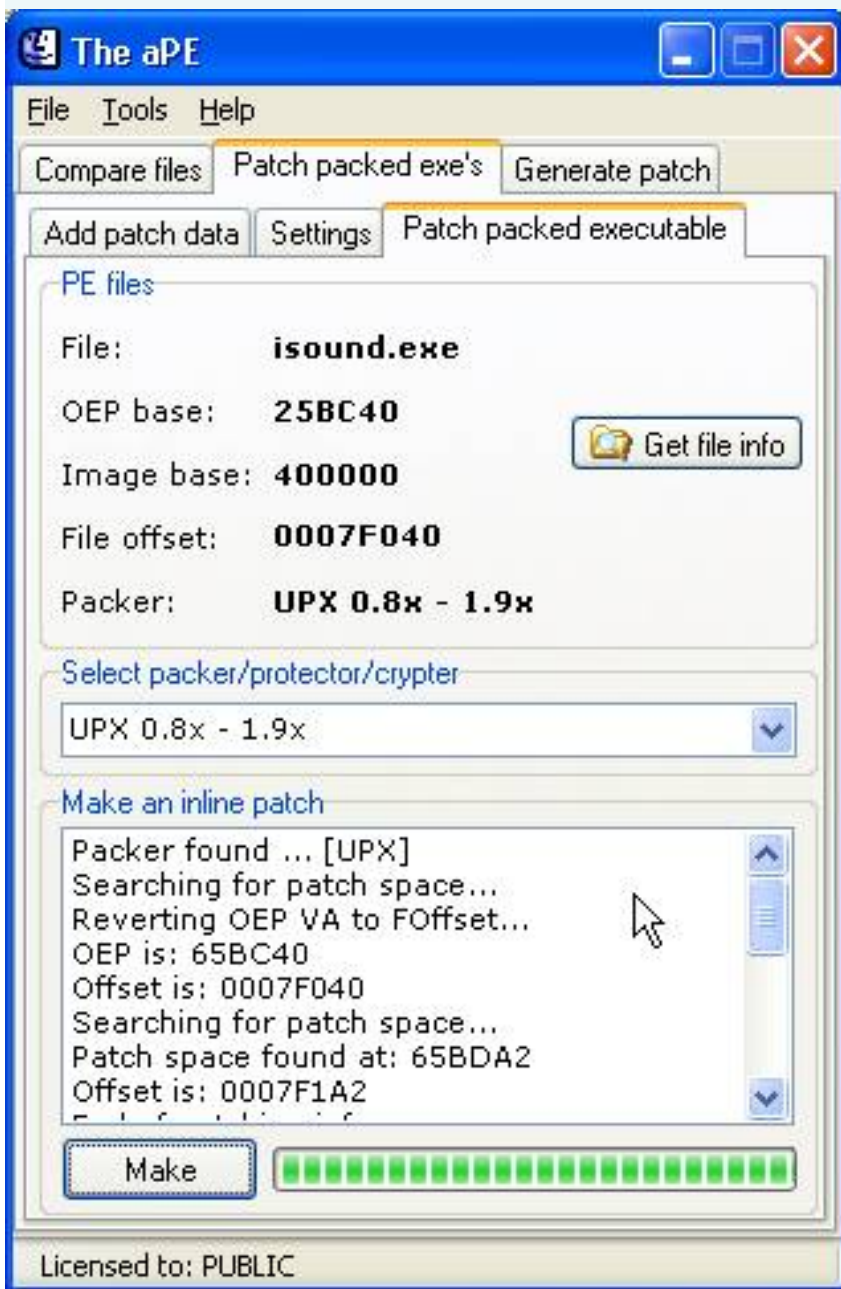
[_Add Bytes:](#)



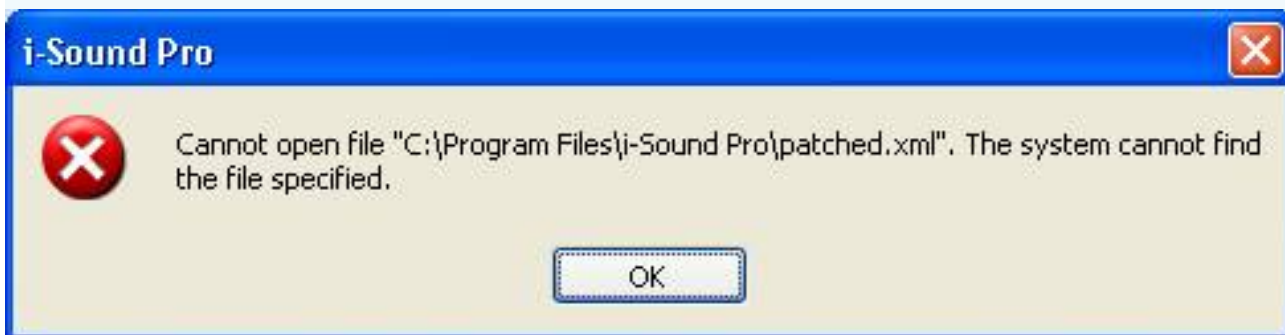
file:///C:/RCE%20Unpacking%20eBook%20[Trans...0LithiumLi]/Inline_Patching%20for%20UPX.htm (9 of 20) [1/9/2009 9:44:38 LithiumLi]



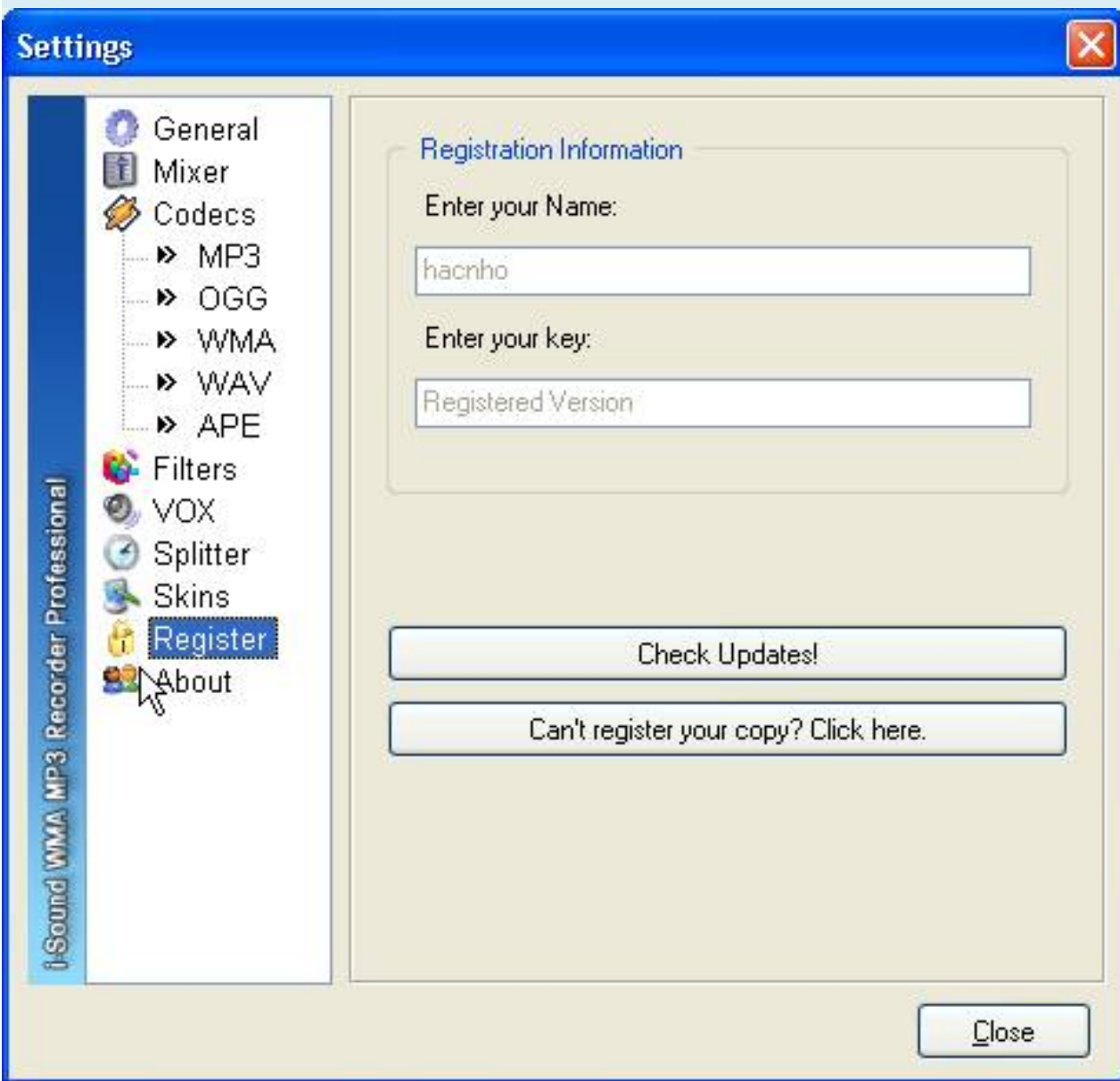
Patch tab _Qua packed executable, Get Info File, select the UPX packer, make click, save again:

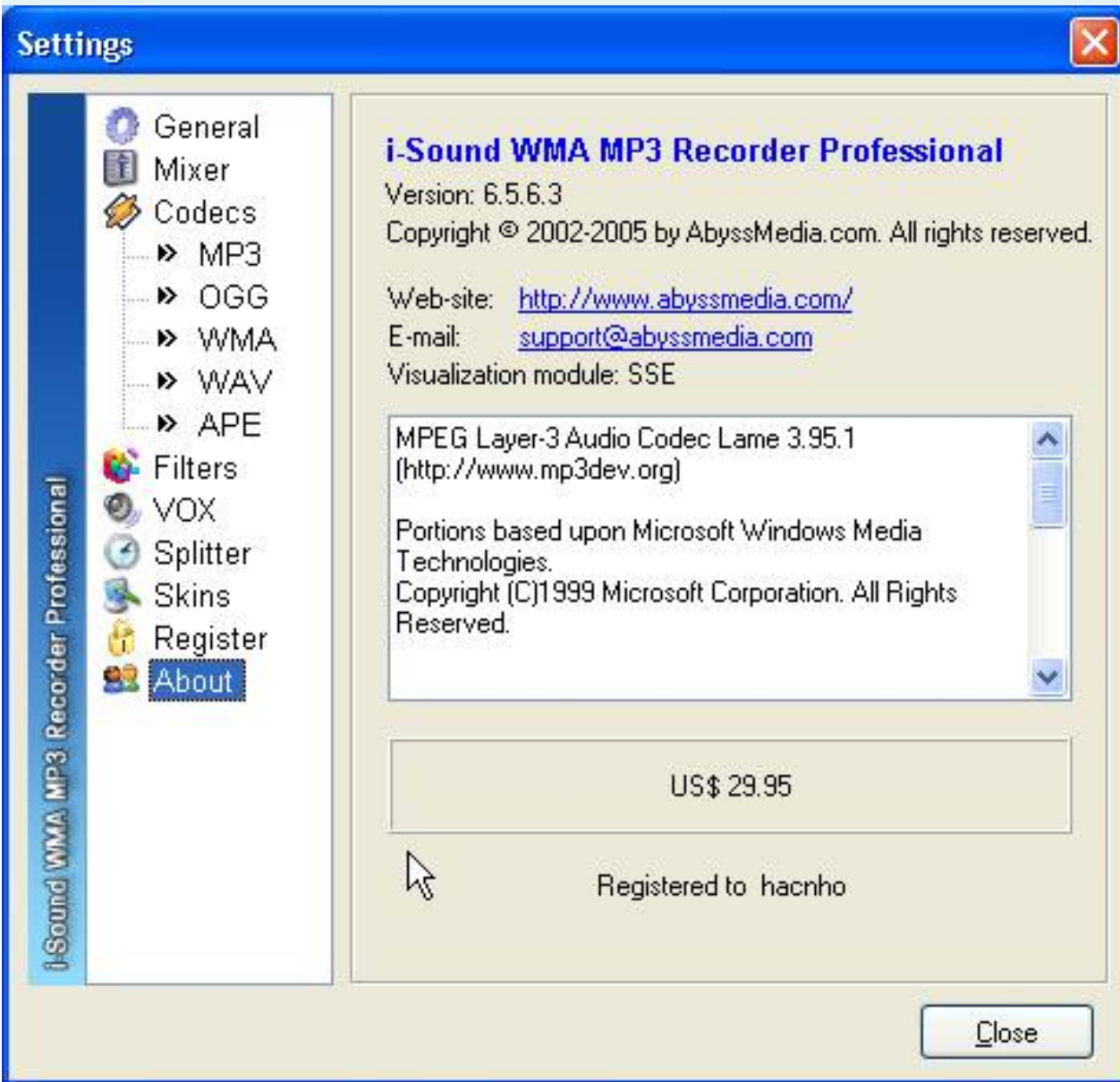


_Run Patched.exe will be error message:



_Ta Patched.exe to rename isound.exe. Test Run:





_Xem Log:

```
Packer found ... [UPX]
Searching for space patch ...
OEP VA reverting to FOffset ...
OEP is: 65BC40
Offset is: 0007F040
Searching for space patch ...
Patch space found at: 65BDA2
Offset is: 0007F1A2
End of patching info ...
Begin patching ...
Make backup files ...
Patching at JMP offset: 0007F1A2
Loading ... JMP to OEP
OEP JMP is: FFE815AC
OEP is: 004DD354
```

Inserting patch code ...
Writing new JMP OEP to ...
BOEP temp: 004DD354
Closing files ...
Compareing files ...

_Xem Script:

```
echo.: The official aPE. asf script:.  
echo  
Packer echo ..... UPX  
echo Version (s) ..... 1.2x - 1.9x  
Written by echo ..... ap0x  
Web-site echo ..... http://ap0x.headcoders.net  
echo Date ..... 6/17/2005  
echo  
check UPX 0.8x - 1.9x  
jne @ end1  
@ begin  
reopen  
Seek 61  
jne @ end2  
save eip, sdata1  
add eip, 1  
Goto eip  
readd dword  
Invert dword  
add eip, dword  
Goto sdata1  
save eip, sdata2  
add sdata2, 4  
echov OEP is sdata2!  
save sdata3, sdata1  
add sdata1, 7  
Goto sdata1  
save sdata1, 00  
save sdata4, 00  
@ doread  
add sdata1, 1  
readb dword  
Cmp sdata1, 30  
je @ dowrite3  
Cmp dword, sdata4  
je @ doread  
jmp @ dowrite2  
@ dowrite  
patch  
writeb 68  
writed sdata2
```



```
writeb C3
exit
@ dwrite2
save sdata1, sdata3
add sdata3, 12
Goto sdata1
writeb E9
wjmp sdata1, sdata3
Goto sdata3
jmp @ dwrite
@ dwrite3
save sdata1, sdata3
Goto sdata1
jmp @ dwrite
@ end1
msgyn selected file might not be UPX packed. Do you want to continue?
je @ begin
exit
@ end2
noexport
echo
echo but not detected UPX version 1.2x - 1.9x!
echo
exit
```

_Dung DUP 2:10 to patcher:

Patch Data

Settings

[illegible]

New Project

Open Project

Save Project

Close Project

Add

EditRemoveCreate Loader

Create Patch

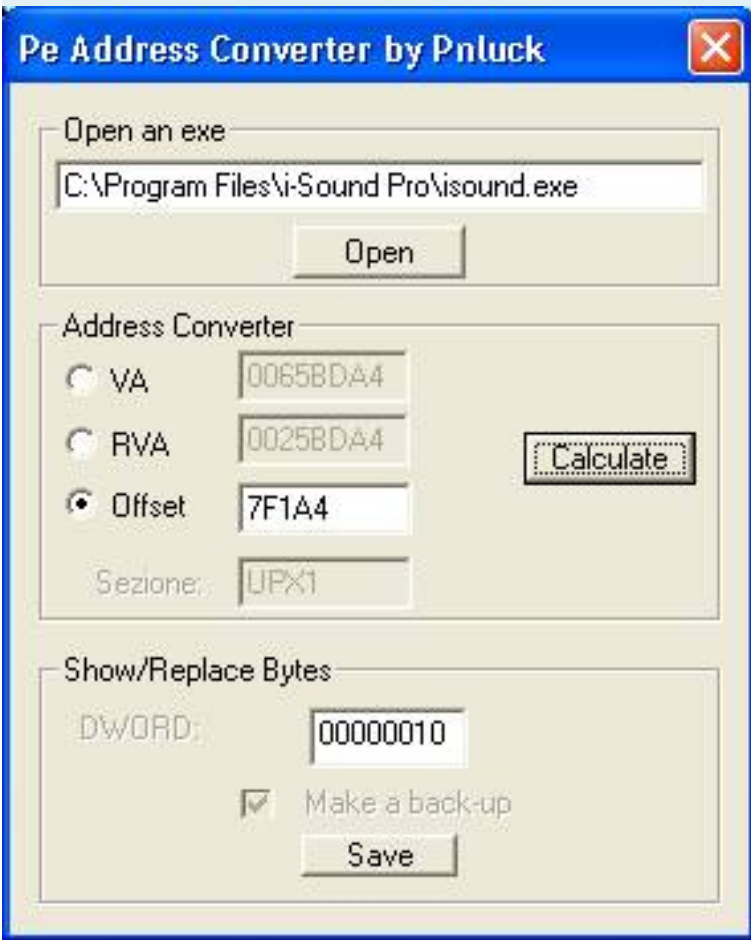
Help

[About](#)

Exit



Xem Table by Dup compare the two files:



_Gia Of VA is 0065BDA4. Ok, load the file has two patch up org and Olly we compare:

Address	Hex dump	Disassembly	Comment
0065BDA4	AC	ADD BYTE PTR DS:[EAX], 1	
0065BDA5	15 E8FFC0BD	ADC EAX, BDC0FFE8	
0065BDA9	65:00D0	ADD AL, DL	Superfluous prefix
0065BDA0	BD 6500F0E0	MOV EBP, E0F00065	
0065BDB2	4D	DEC EBP	
0065BDB3	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDB5	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDB7	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDB9	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDBB	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDBD	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDBF	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDC1	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDC3	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDC5	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDC7	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDC9	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDCB	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDCD	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDCE	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDD1	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDD3	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDD5	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDD7	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDD9	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDDB	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDDD	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDDF	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDE1	0000	ADD BYTE PTR DS:[EAX], AL	
0065BDE3	0000	ADD BYTE PTR DS:[EAX], AL	

Address	Hex dump	Disassembly	Comment
00658DA4	1000	ADD BYTE PTR DS:[EAX],AL	
00658DA6	0000	ADD BYTE PTR DS:[EAX],AL	
00658DA8	C0BD 6500D0BD 65	SAR BYTE PTR SS:[EBP+BDD00065],65	Shift constant out of range 1..31
00658DAF	00F0	ADD AL,DH	
00658DB1	E0 4D	LOOPDNE SHORT isound.0065BE00	
00658DB3	0000	ADD BYTE PTR DS:[EAX],AL	
00658DB5	0000	ADD BYTE PTR DS:[EAX],AL	
00658DB7	00C6	ADD DH,AL	
00658DB9	05 180A4D00	ADD EAX,isound.004D0A18	
00658DBE	EB E9	JMP SHORT isound.0065BDA9	
00658DC0	90	NOP	
00658DC1	15 E8FF0000	ADC EAX,0FFEB	
00658DC6	0000	ADD BYTE PTR DS:[EAX],AL	
00658DC8	0000	ADD BYTE PTR DS:[EAX],AL	
00658DCA	0000	ADD BYTE PTR DS:[EAX],AL	
00658DCC	0000	ADD BYTE PTR DS:[EAX],AL	
00658DCE	0000	ADD BYTE PTR DS:[EAX],AL	
00658DD0	0000	ADD BYTE PTR DS:[EAX],AL	
00658DD2	0000	ADD BYTE PTR DS:[EAX],AL	
00658DD4	0000	ADD BYTE PTR DS:[EAX],AL	
00658DD6	0000	ADD BYTE PTR DS:[EAX],AL	
00658DD8	0000	ADD BYTE PTR DS:[EAX],AL	
00658DDA	0000	ADD BYTE PTR DS:[EAX],AL	
00658DDC	0000	ADD BYTE PTR DS:[EAX],AL	
00658DDE	0000	ADD BYTE PTR DS:[EAX],AL	
00658DE0	0000	ADD BYTE PTR DS:[EAX],AL	
00658DE2	0000	ADD BYTE PTR DS:[EAX],AL	
00658DE4	0000	ADD BYTE PTR DS:[EAX],AL	
00658DE6	0000	ADD BYTE PTR DS:[EAX],AL	
00658DE8	0000	ADD BYTE PTR DS:[EAX],AL	

[_Unpacked.Cracked.by.hacnho](#)

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtlm, hosiminh, Nilrem, fly, MaDMAn_H3rCul3s, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light ... and you!

Special Thanx Cracks Latinos.

Thanx OllyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 12/10/2005)

InsaneFIDO's UnWrapME

Author: Computer_Angel

Target: http://crackmes.de/users/insanefido/insanefidos_unwrapme/

Using similar concept to the wrapper Reflexive wrapper used on games but I hope it is a little more difficult to reverse than that. On the Windows XP unwrapped file should display the XP theme. Coded on XPSP2, may not work with W2000 but I hope it does.

Difficulty: 3 - Getting harder

Platform: Windows 2000/XP only

Language: Unspecified / other

Target includes 2 files: Insane.exe and F1D0.dat. No signs are packed.

Insane.exe: File Implementation

F1D0.dat: File seems to be implemented, however, completely wrong PEHeader (may be due to intentionally sabotage the)

Test Insane.exe, and observations in the Task Manager, found out Insane.exe have 1 new process has been created in the "My Documents \ Angel \ Local Settings \ Temp \ ~ FD33C. Tmp. -> Predicting: Insane.exe 1 will create a new file based on F1D0.dat (can be encrypted). And then Create a new process. Load Insane.exe in olly and observations. The program will break at TLS. OEP's program is at 401,000.

```
00401000> E8 660D0000 CALL Insane.00401D6B
```

```
00401005 46 INC ESI
```

We try to find Intermodular All calls by Insane.exe and found IAT deleted 0

```
00401D72 - FF25 00204000 JMP DWORD PTR DS: [402000]
```

```
00401D78 - FF25 04204000 JMP DWORD PTR DS: [402004]
```

```
00401D7E - FF25 08204000 JMP DWORD PTR DS: [402008]
```

```
00401D84 - FF25 0C204000 JMP DWORD PTR DS: [40200C]
```

```
00401D8A - FF25 10204000 JMP DWORD PTR DS: [402010]
```

```
00401D90 - FF25 14204000 JMP DWORD PTR DS: [402014]
```

```
00401D96 - FF25 18204000 JMP DWORD PTR DS: [402018]
```

```
00401D9C FF25 1C204000 JMP DWORD PTR DS: [40201C]
```

```
00401DA2 - FF25 20204000 JMP DWORD PTR DS: [402020]
```

```
00401DA8 - FF25 24204000 JMP DWORD PTR DS: [402024]
```

```
00401DAE - FF25 28204000 JMP DWORD PTR DS: [402028]
```

```
00401DB4 FF25 2C204000 JMP DWORD PTR DS: [40202C]
```

```
00401DBA - FF25 30204000 JMP DWORD PTR DS: [402030]
```

```
00401DC0 - FF25 34204000 JMP DWORD PTR DS: [402034]
```

```
00401DC6 - FF25 38204000 JMP DWORD PTR DS: [402038]
```

```
00401DCC FF25 3C204000 JMP DWORD PTR DS: [40203C]
00401DD2 - FF25 40204000 JMP DWORD PTR DS: [402040]
00401DD8 - FF25 44204000 JMP DWORD PTR DS: [402044]
00401DDE - FF25 48204000 JMP DWORD PTR DS: [402048]
00401DE4 - FF25 4C204000 JMP DWORD PTR DS: [40204C]
00401DEA FF25 50204000 JMP DWORD PTR DS: [402050]
00401DF0 FF25 54204000 JMP DWORD PTR DS: [402054]
00401DF6 FF25 58204000 JMP DWORD PTR DS: [402058]
00401DFC FF25 5C204000 JMP DWORD PTR DS: [40205C]
00401E02 - FF25 60204000 JMP DWORD PTR DS: [402060]
00401E08 - FF25 64204000 JMP DWORD PTR DS: [402064]
00401E0E - FF25 68204000 JMP DWORD PTR DS: [402068]
00401E14 FF25 6C204000 JMP DWORD PTR DS: [40206C]
00401E1A FF25 70204000 JMP DWORD PTR DS: [402070]
00401E20 FF25 74204000 JMP DWORD PTR DS: [402074]
00401E26 FF25 78204000 JMP DWORD PTR DS: [402078]
00401E2C FF25 7C204000 JMP DWORD PTR DS: [40207C]
00401E32 FF25 80204000 JMP DWORD PTR DS: [402080]
00401E38 FF25 84204000 JMP DWORD PTR DS: [402084]
00401E3E FF25 8C204000 JMP DWORD PTR DS: [40208C]
```

If you run the program entirely in F9, and review Intermodular calls, you'll see:

```
00401D72 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401D78 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401D7E FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401D84 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401D8A FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401D90 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401D96 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401D9C FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DA2 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DA8 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DAE FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DB4 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DBA FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DC0 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DC6 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DCC FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DD2 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DD8 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DDE FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DE4 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
```

```

00401DEA FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DF0 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DF6 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401DFC FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401E02 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401E08 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401E0E FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401E14 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401E1A FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401E20 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401E26 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401E2C FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401E32 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401E38 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
00401E3E FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E

```

So you need to find where Insane.exe start implementing the action changes. Okie, set hw break on memory write at 00401D72 (DWORD). And run the program again, we will:

```

00401CE5> BD 0E1D4000 MOV EBP, Insane.00401D0E; Write Viatualize IAT
00401CEA 892D 40454000 MOV DWORD PTR DS: [404540], EBP
00401CF0 BD 40454000 MOV EBP, Insane.00404540
00401CF5 8B35 50394000 MOV ESI, DWORD PTR DS: [403950]
00401CFB 8A06 MOV AL, BYTE PTR DS: [ESI]
00401CFD 3C FF Cmp AL, 0FF
00401CFF 75 0C JNZ SHORT Insane.00401D0D
00401D01 46 INC ESI
00401D02 C606 15 MOV BYTE PTR DS: [ESI], 15
00401D05 46 INC ESI
00401D06 892E MOV DWORD PTR DS: [ESI], EBP
00401D08 83C6 04 ADD ESI, 4
^ EE 00401D0B EB JMP SHORT Insane.00401CFB
00401D0D C3 RETN

```

This is func implement the above changes. Also, see the code of

```

00401DBA FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E

```

We have:

```

00401D0E 64:8 B2D 0800000> MOV EBP, DWORD PTR FS: [8]
00401D15 5e POP ESI
00401D16 83EE 06 SUB ESI, 6

```



```

00401D19 A1 50394000 MOV EAX, DWORD PTR DS: [403950]
00401D1E 2BF0 SUB ESI, EAX
00401D20 8BC6 MOV EAX, ESI
00401D22 25 FF000000 AND EAX, 0FF
00401D27 33DB XOR EBX, EBX
00401D29 B3 06 MOV BL, 6
00401D2B F6F3 DIV BL
00401D2D C0E0 02 SHL AL, 2
00401D30 8BF0 MOV ESI, EAX
00401D32 81FE 88000000 Cmp ESI, 88
00401D38 7C 1A JL SHORT Insane.00401D54
00401D3A 83C6 04 ADD ESI, 4
00401D3D 8B1D 58454000 MOV EBX, DWORD PTR DS: [404558]; USER32.7E410000
00401D43 0335 60454000 ADD ESI, DWORD PTR DS: [404560]
00401D49 8B36 MOV ESI, DWORD PTR DS: [ESI]
00401D4B 8975 10 MOV DWORD PTR SS: [EBP +10], ESI
00401D4E 015D 10 ADD DWORD PTR SS: [EBP +10], EBX
00401D51 FF65 10 JMP DWORD PTR SS: [EBP +10]
00401D54 0335 60454000 ADD ESI, DWORD PTR DS: [404560]
00401D5A 8B06 MOV EAX, DWORD PTR DS: [ESI]
00401D5C 8945 10 MOV DWORD PTR SS: [EBP +10], EAX
00401D5F A1 50454000 MOV EAX, DWORD PTR DS: [404550]
00401D64 0145 10 ADD DWORD PTR SS: [EBP +10], EAX
00401D67 FF65 10 JMP DWORD PTR SS: [EBP +10]
00401D6A C3 RETN

```

This can be called by the API emulation Insane.exe. So difficult points when analyzing Insane.exe the API is hidden loss, instead it is a call to the API function Emulation.Tuy course, if the notice:

```
00401D51 FF65 10 JMP DWORD PTR SS: [EBP +10]
```

and:

```
00401D67 FF65 10 JMP DWORD PTR SS: [EBP +10]
```

Then restore the IAT for easy analysis is not difficult. Try to change EIP 1 in emulation of the API, for example, we choose:

```
00401D78 FF15 40454000 CALL DWORD PTR DS: [404540]; Insane.00401D0E
```

Menu select "New origin here to set EIP in 00401D78, from which to trace through the" JMP DWORD PTR SS: [EBP +10] "the value [EBP +10] will actually contain the API we need to find.

Thus, to restore the IAT, we do the following:

1st Set 0040101C break in (the function of the changing IAT).

2. When the break, we turn the EIP to execute commands emulation API.

3rd to Trace, to meet the instruction "JMP DWORD PTR SS: [EBP +10]," read value [EBP +10], will be the original API

4. Implementation of editing the IAT, the original point to the API.

To hand it is very tired, so I recommend that you use the script (OllyDBGScript) to help implement the action repeated this.

```
var cur_eip
var top_vir_iat
var iat_top
```

```
init:
bphwcall
bc
```

```
start:
BP 00401D51
bpgoto 00401D51, iat_addr_found_handler
BP 00401D67 //
bpgoto 00401D67, iat_addr_found_handler
BP 0040101C // after_write_virtualize_handle
esto
mov cur_eip, eip
mov top_vir_iat, [403950]
alloc 1000
mov iat_top, $RESULT
```

```
iat_begin:
mov eip, top_vir_iat
Cmp [top_vir_iat], 0
je the_end
```

```
find_real_api:
exec
pushad
pushfd
ende
esto
```

```
iat_addr_found_handler:
mov tmp, ebp
```

```

add tmp, 10
mov [iat_top], [tmp]
mov [top_vir_iat + 1], 25.1
mov [top_vir_iat + 2], iat_top
add iat_top, 4
add top_vir_iat, 6
exec
popfd
popad
ende
jmp iat_begin

```

```

the_end:
mov eip, cur_eip
bc
ret

```

Thus, when run this script is completed, IAT has been restored, this will help very much to see and analyze code.

Dumped and learn.

After IAT fix, I would use the IDA Hexrays +. Talk about a Hexrays 1 plugin and help against asm code -> C + +, in 1 cases it helps our code easier to read, not only is really the C + + code completely. Insane_fix_iat.exe Load to IDA, and hexrays, we will have an overview of the program. First, Insane will create 1 files in temp folder as above, then run this executable file through the API CreateProcessA:

```

StartupInfo.cbReserved2 = 512;
StartupInfo.lpReserved2 = (LPBYTE) lpAddress;
CreateProcessA (ApplicationName, 0, 0, 0, 1, 3U, 0, 0, & StartupInfo, & ProcessInformation);

```

Note 1: CreateProcessA known to CreationFlags = DEBUG_PROCESS | DEBUG_ONLY_THIS_PROCESS (3U).

And then create 1 loop:

```

while (1)
(
WaitForDebugEvent (& DebugEvent, Infinite);
if (DebugEvent.dwDebugEventCode == EXIT_PROCESS_DEBUG_EVENT)
(
CloseHandle (handle_process_dword_4038F4);
UnmapViewOfFile (lpBaseAddress);
VirtualFree (lpAddress, 0x100u, 0x4000u);
GetExitCodeProcess (ProcessInformation.hProcess, & ExitCode);

```



```
GetExitCodeThread (ProcessInformation.hThread, & dword_404548);
TerminateThread (ProcessInformation.hThread, dword_404548);
TerminateProcess (ProcessInformation.hProcess, ExitCode);
CloseHandle (ProcessInformation.hThread);
CloseHandle (ProcessInformation.hProcess);
lstrcpy (CmdLine, PathName);
lstrcat (CmdLine, "winstat.bat");
dword_404568 = CreateFileA (CmdLine, 0x40000000u, 0, 0, 2u, 0x80u, 0);
sub_401E3E (& unk_404204, "del&percent; s \ r \ .. nmove&percent; s \ r \ ndel. \ \&percent;
s", ApplicationName, CmdLine, "winstat.bat");
v8 = lstrlen (& unk_404204);
WriteFile (dword_404568, & unk_404204, v8, & NumberOfBytesWritten, 0);
CloseHandle (dword_404568);
WinExec (CmdLine, 0);
ExitProcess (0);
)
if (DebugEvent.dwDebugEventCode == CREATE_PROCESS_DEBUG_EVENT)
(
handle_process_dword_4038F4 = (HANDLE) DebugEvent.u.Exception.ExceptionRecord.ExceptionFlags;
hThread = DebugEvent.u.Exception.ExceptionRecord.ExceptionRecord;
process_id_dword_4038F8 = DebugEvent.dwProcessId;
thread_id_dword_4038FC = DebugEvent.dwThreadId;
Goto continue_dbg;
)
if (DebugEvent.dwDebugEventCode == EXCEPTION_DEBUG_EVENT)
(
if (DebugEvent.u.Exception.ExceptionRecord.ExceptionCode == EXCEPTION_BREAKPOINT)
(
ContinueDebugEvent (DebugEvent.dwProcessId, DebugEvent.dwThreadId, DBG_CONTINUE);
)
else
(
if (DebugEvent.u.Exception.ExceptionRecord.ExceptionCode! = EXCEPTION_SINGLE_STEP)
Goto continue_dbg;
Context.ContextFlags = 65537;
GetThreadContext (hThread, & Context);
if (Context.Eip == 0x40100B)
(
WriteProcessMemory (handle_process_dword_4038F4, off_40391C, "jFjh | | IH, 1u,
& NumberOfBytesWritten);
dword_403914 = (LPCVOID) dword_40390C;
* (__DWORD *) (* MK_FP (__FS__, 8) + 80) = v7ffe0000;
by
```

```
(
ReadProcessMemory (handle_process_dword_4038F4, dword_403914, & byte_40394C, 1u,
& NumberOfBytesWritten);
byte_40394C = (stru_4030C0.anonymous_3 ^ (unsigned __int8) ((stru_4030C0.anonymous_1
^ (unsigned __int8) ((stru_4030C0.gap_0 [0] ^ (unsigned __int8) byte_40394C) -
stru_4030C0.anonymous_0))
- Stru_4030C0.anonymous_2))
- Stru_4030C0.anonymous_4;
WriteProcessMemory (
handle_process_dword_4038F4,
(LPVOID) dword_403914 + +,
& byte_40394C,
1u,
& NumberOfBytesWritten);
)
while ((signed int) dword_403914 <(signed int) 0x401380u);
Context.Eip = 0x40100Au;
SetThreadContext (hThread, & Context);
ReadProcessMemory (handle_process_dword_4038F4, off_403934, & dword_404534, 4u,
& NumberOfBytesWritten);
V2 = (char *) dword_403904 + dword_403930;
while (1)
(
v3 = v2;
ReadProcessMemory (handle_process_dword_4038F4, v2, & unk_404D88, 1u, & NumberOfBytesWritten);
if (unk_404D88! = -1)
break;
v9 = (char *) v3 + 2;
V10 = (char *) v3 + 2;
ReadProcessMemory (handle_process_dword_4038F4, v9, & unk_404D70, 4u, & NumberOfBytesWritten);
dword_404D80 = dword_404534 + unk_404D70;
WriteProcessMemory (handle_process_dword_4038F4, V10, & dword_404D80, 4u,
& NumberOfBytesWritten);
V2 = (char *) V10 + 4;
)
dword_4034E4 = (int) lpBaseAddress;
v4 = dword_4034E4;
v5 = 0;
by
(
* (* _BYTE) V4 = (stru_4030BA.anonymous_3 ^ (unsigned __int8) ((stru_4030BA.anonymous_1
^ (unsigned __int8) ((stru_4030BA.gap_0 [0] ^ * (* _BYTE) v4) - stru_4030BA.anonymous_0))
- Stru_4030BA.anonymous_2))
- Stru_4030BA.anonymous_4;
```

```
+ + v4;
+ + v5;
)
while (v5 <= dword_4034EC);
FlushViewOfFile (lpBaseAddress, 0);
)
else
(
if (Context.Eip == 0x40102B)
(
- Context.Eip;
SetThreadContext (hThread, & Context);
WriteProcessMemory (handle_process_dword_4038F4, off_403918, & aJsjhLh [1], 1u,
& NumberOfBytesWritten);
WriteProcessMemory (handle_process_dword_4038F4, off_40391C, "jFjh | | IH, 0x1Eu,
& NumberOfBytesWritten);
WriteProcessMemory (handle_process_dword_4038F4, off_403908, "jFjh | | IH, 0x250u,
& NumberOfBytesWritten);
)
else
(
if (Context.Eip == (_DWORD) loc_4010C5)
(
- Context.Eip;
SetThreadContext (hThread, & Context);
WriteProcessMemory (handle_process_dword_4038F4, off_403920, & aJsjhLh [7], 1u,
& NumberOfBytesWritten);
)
else
(
if (Context.Eip == 0x4010EF)
(
- Context.Eip;
SetThreadContext (hThread, & Context);
WriteProcessMemory (handle_process_dword_4038F4, off_403924, & aJsjhLh [4], 1u,
& NumberOfBytesWritten);
VirtualProtectEx (handle_process_dword_4038F4, dword_403904, 0x400u, 1u, & flOldProtect);
VirtualProtectEx (handle_process_dword_4038F4, off_403928, 0x200u, 0x10u, & flOldProtect);
)
)
)
)
dword_4034D4 = v7ffe0000;
ContinueDebugEvent (DebugEvent.dwProcessId, DebugEvent.dwThreadId, DBG_CONTINUE);
```



```

)
)
else
(
continue_dbg:
dword_4034D4 = v7ffe0000;
ContinueDebugEvent (DebugEvent.dwProcessId, DebugEvent.dwThreadId, DBG_EXCEPTION_NOT_HANDLED);
)
)

```

View a code in the loop, we can say as follows:

Insane.exe (The father) called --- (debug mode) -> ~ DFxxx.tmp (the child), the exception from the child will be born Insane.exe (CT father) process.

Observations code, we see the exception that the program will create:

CREATE_PROCESS_DEBUG_EVENT, EXCEPTION_DEBUG_EVENT, EXIT_PROCESS_DEBUG_EVENT.

And the process is in the EXCEPTION_DEBUG_EVENT.

If you have used to nanomites armadillo in the format will see the process is very familiar. It can be understood as follows:

1. Chuong the children will be removed to No. 1 instruction, instead of instruction is able to create debug exception (INT1, INT3, etc. ...)

2. Khi the children run to the instructions this exception will occur again and pause

3. Chuong the parents will take exception from the child and handling: instruction to edit -> new instruction, eip change of the child, ignore errors, etc. ..

So with the way the parents always have the + running parallel each other.

Back to the dump file is, Insane.exe Load to Olly, the first one will break in CreateProcessA and run the program, we break in:

```

0012FF98 00401153 / Call to CreateProcessA from Insane.0040114E
0012FF9C 00403F3C | ModuleFileName = "C: \ DOCUME ~ 1 \ ANGEL ~ 1 \ locals ~ 1 \ Temp \
~ DF3BF9.tmp"
0012FFA0 00000000 | CommandLine = null
0012FFA4 00000000 | pProcessSecurity = null
0012FFA8 00000000 | pThreadSecurity = null
0012FFAC 00000001 | InheritHandles = TRUE
0012FFB0 00000003 | CreationFlags = DEBUG_PROCESS | DEBUG_ONLY_THIS_PROCESS
0012FFB4 00000000 | pEnvironment = null
0012FFB8 00000000 | CurrentDir = null
0012FFBC 00403570 | pStartupInfo = Insane.00403570
0012FFC0 004035B4 \ pProcessInfo = Insane.004035B4
0012FFC4 7C816FD7 Return to kernel32.7C816FD7

```

Therefore, we only need to "C: \ DOCUME ~ 1 \ ANGEL ~ 1 \ locals ~ 1 \ Temp \ ~ DF3BF9.tmp" and copy this file, rename it to unwrap.exe. This is the original file unwrap.exe. Try running unwrap.exe -> Exception error: D. From here, we find out, the father (Insane.exe) to record what the children. Since the source code, we must know breakpoint at API WriteProcessMemory. Running back Insane.exe, we have:

```
0012FFAC 00401358 / Call to WriteProcessMemory from Insane.00401353
0012FFB0 00000034 | 00000034 hProcess = (Window)
0012FFB4 0040100A | Address = 40100A
0012FFB8 004034C5 | Buffer = Insane.004034C5
0012FFBC 00000001 | BytesToWrite = 1
0012FFC0 00404508 \ pBytesWritten = Insane.00404508
0012FFC4 7C816FD7 Return to kernel32.7C816FD7
```

With:

Address: Address contains almost write in the child

BytesToWrite: number of bytes to write

Buffer: contains bytes to write.

...

And will break many times more so.

Again, as dozens of times hands as of 1 positive image, I will still use scripts to monitor the implementation, as well as using scripts to create 1 to fix the scripts of children.

```
var first
```

```
init:
```

```
bc
GPA "WriteProcessMemory", "kernel32.dll"
bp $ RESULT
bpgoto $ result, writemem_handler
wrt "fix_insane_dump.txt Auto ","// gen demo script \ r \ n"
mov first, 0
esto
```

```
writemem_handler:
```

```
Cmp first, 0
jne next
mov first, 1
GPA "WriteProcessMemory", "kernel32.dll"
bc $ RESULT
alloc 1000
mov hmem, $ RESULT
mov tmp, hmem
add tmp, 22
mov [hmem], 609C6A006A016800000000FF351C394000FF35F4384000E80A1EB4FF9D610000000EBFE # #
mov [hmem +7], tmp, 4
```

```

mov tmp_eip, eip
mov tmp, hmem
add tmp, 01e
BP tmp
mov eip, hmem
esto
bc eip
mov eip, tmp_eip
free hmem, 1000
GPA "WriteProcessMemory", "kernel32.dll"
bp $ RESULT
bpgoto $ result, writemem_handler

```

```

Next:
mov addr, [ESP +8]
mov buf, [esp +0 C]
mov size, [esp +10]
Cmp size, 4
jbe type_1
jmp type_2

```

```

type_1:
mov value, [buf], size
Cmp value, 0F1
je skip_type_1
eval "mov [addr ()], (value), (size)"
wrt a "fix_insane_dump.txt, $ RESULT
esto

```

```

skip_type_1:
esto

```

```

type_2:
eval "addr (). bin"
dm buf, size, $ RESULT
eval "lm (addr) (size), (addr). bin"
wrt a "fix_insane_dump.txt, $ RESULT
esto

```

When you run this script, the information recorded in the child will be recorded by 2 ways:

1st Record <= 4 bytes, perform assigned command script = normal.

2nd Writing > 4 bytes, the buffer dump file, then load and use the cmd script.

Summary of what has been achieved in this section:

1. Mechanism of the father-child Insane.exe

2. Unwrap.exe File

3rd script to monitor memory write operation to the children

4th Scripts perform tasks write to the children (3 to gen).

Unwrap Fix, fix iat.

From the article, we have unwrap.exe, however, to implement fully the need to edit.

View code from the 2 above, we know:

a. Program to implement the 0x40100B -> create exception.

b. The father will write mem to 1 series of the program, track the genetic script, we see particular since 40100A-40137F,

Then, continue to overwrite iat jmp address from 40131E-401390.

c. Write to 0x1E bytes 40100A <- works to remove, break old code after going through -> complicate the dump.

d. Write to 402,060 bytes 0x250 <-? unknown reasons

e. Write 4010C4 and 68 at C7 in 4010EE.

Therefore, load unwrap.exe to Olly, we have:

```
00401000> 68 3C304000 PUSH unwrap.0040303C
00401005 E8 8E030000 CALL <JMP.&kernel32.GetStartupInfoA>
0040100A F1 INT1
0040100B 93 XCHG EAX, EBX
0040100C 5B POP EBX
0040100D BE 24 AND AL, 0BE
0040100F 93 XCHG EAX, EBX
00401010 93 XCHG EAX, EBX
00401011 1E PUSH DS
00401012 9F LAHF
00401013 - E3 D3 JECXZ SHORT unwrap.00400FE8
00401015 93 XCHG EAX, EBX
00401016 C593 DBC883D3 LDS EDX, FWORD PTR DS: [EBX + D383C8DB]; modification of
segment register
0040101C 93 XCHG EAX, EBX
0040101D C593 DB93E3D3 LDS EDX, FWORD PTR DS: [EBX + D3E393DB]; modification of
segment register
00401023 93 XCHG EAX, EBX
C89FE3D3 BA 00401024 MOV EDX, D3E39FC8
00401029 93 XCHG EAX, EBX
0040102A A4 MOVS BYTE PTR ES: [EDI], BYTE PTR DS: [ESI]
```

at

```
0040100A F1 INT1
```

instruction has been changed to create exception.Nhu so we will run the script from the gen 2 (Appendix 1).

However only one run to the end b), because khúc after quite short and can manually duoc.Run script, copy the already write -> new file: Unwrap_1.exe

Load Unwrap_1.exe, to olly, view intermodular calls:

```
0040131C FF25 94070200 JMP DWORD PTR DS: [20794]
00401322 FF25 88070200 JMP DWORD PTR DS: [20788]
00401328 FF25 8C070200 JMP DWORD PTR DS: [2078C]
0040132E FF25 90070200 JMP DWORD PTR DS: [20790]
00401334 FF25 D8070200 JMP DWORD PTR DS: [207D8]
0040133A FF25 D4070200 JMP DWORD PTR DS: [207D4]
00401340 FF25 D0070200 JMP DWORD PTR DS: [207D0]
00401346 FF25 CC070200 JMP DWORD PTR DS: [207CC]
0040134C FF25 B4070200 JMP DWORD PTR DS: [207B4]
00401352 FF25 B0070200 JMP DWORD PTR DS: [207B0]
00401358 FF25 DC070200 JMP DWORD PTR DS: [207DC]
0040135E FF25 B8070200 JMP DWORD PTR DS: [207B8]
00401364 FF25 BC070200 JMP DWORD PTR DS: [207BC]
0040136A FF25 C0070200 JMP DWORD PTR DS: [207C0]
00401370 FF25 C4070200 JMP DWORD PTR DS: [207C4]
00401376 FF25 C8070200 JMP DWORD PTR DS: [207C8]
0040137C FF25 A8070200 JMP DWORD PTR DS: [207A8]
00401382 FF25 A4070200 JMP DWORD PTR DS: [207A4]
00401388 FF25 A0070200 JMP DWORD PTR DS: [207A0]
0040138E FF25 9C070200 JMP DWORD PTR DS: [2079C]
```

I come off as completely ko signs have anything to open, and the original unwrap.exe can not run that i have the father. And if the father, then how do debug, Attach ... be.

I say that the secret is completely in the difficult things lo wise, I think now 1 to technical for the ancient armadillo shake lờ past, which is the API DebugActiveProcessStop.

My mind is so: when the program run entirely, since that is when the father, I will call the API "DebugActiveProcessStop" with a processid parameters of the child. So then we can Attach to see .. and some interesting information in the process.

Conducted as follows:

1. Load Insane.exe to olly

2nd F9 to run completely.

3rd to

```
0040115F 833D 90384000 0> Cmp DWORD PTR DS: [403890], 5
00401166 0F85 26010000 JNZ Insane.00401292
```

Break at 0040115F, wait 1 billion, will see olly break. Even this position, you type in asm:

```
push processid
call DebugActiveProcessStop
```

Then press F8 to step through the order 2, so from now Insane.exe and the process of it do what is the relationship with each other again. So we gently Attach to the children, and over 401,000. From here, we make the calls intermodular find:

```
0040131C - FF25 94070200 JMP DWORD PTR DS: [20,794]; gdi32.BitBlt
00401322 - FF25 88070200 JMP DWORD PTR DS: [20,788]; gdi32.CreateCompatibleDC
00401328 - FF25 8C070200 JMP DWORD PTR DS: [2078C]; gdi32.DeleteDC
0040132E - FF25 90070200 JMP DWORD PTR DS: [20,790]; gdi32.SelectObject
00401334 - FF25 D8070200 JMP DWORD PTR DS: [207D8]; USER32.BeginPaint
0040133A - FF25 D4070200 JMP DWORD PTR DS: [207D4]; USER32.CreateDialogParamA
00401340 - FF25 D0070200 JMP DWORD PTR DS: [207D0]; USER32.DialogBoxParamA
00401346 - FF25 CC070200 JMP DWORD PTR DS: [207CC]; USER32.EndDialog
0040134C - FF25 B4070200 JMP DWORD PTR DS: [207B4]; USER32.EndPaint
00401352 - FF25 B0070200 JMP DWORD PTR DS: [207B0]; USER32.GetClientRect
00401358 - FF25 DC070200 JMP DWORD PTR DS: [207DC]; USER32.InvalidateRect
0040135E - FF25 B8070200 JMP DWORD PTR DS: [207B8]; USER32.LoadBitmapA
00401364 - FF25 BC070200 JMP DWORD PTR DS: [207BC]; USER32.ReleaseCapture
0040136A - FF25 C0070200 JMP DWORD PTR DS: [207C0]; USER32.SendMessageA
00401370 - FF25 C4070200 JMP DWORD PTR DS: [207C4]; USER32.SetCapture
00401376 - FF25 C8070200 JMP DWORD PTR DS: [207C8]; USER32.UpdateWindow
0040137C - FF25 A8070200 JMP DWORD PTR DS: [207A8]; kernel32.ExitProcess
00401382 - FF25 A4070200 JMP DWORD PTR DS: [207A4]; kernel32.GetModuleHandleA
00401388 - FF25 A0070200 JMP DWORD PTR DS: [207A0]; kernel32.GetStartupInfoA
0040138E - FF25 9C070200 JMP DWORD PTR DS: [2079C]; kernel32.Sleep
```

Too healthy, it was available in the region is to remember 20788c and copy this entire IAT (binary).

```
F0 5F F1 77 6F 6E F1 77 80 5B F1 77 89 6F F1 77 00 00 00 00 42 24 80 7C EE 1E 80 7C A1 B6 80 7C
DA CD 81 7C 00 00 00 00 AE B6 41 7E 1D B6 41 7E F0 54 42 7E EA D6 41 7E 83 F3 42 7E CE D6 41 7E
F9 D7 41 7E C9 59 42 7E 0C B1 43 7E A3 C7 43 7E 09 B6 41 7E F5 B5 41 7E
```

Now back with unwrap_1.exe, we have the IAT, to fix, can be used manually, but I calculate capital Grid, Khoai automatically so the script to a script to fix the iat.

Ideas such as the following:

1. Create a new memory area (eg 870000)
2. Write table IAT get in on how to begin to offset a 88
3. Read iat jmp addr old.
4. And value to the FF (only get first byte) and then calculate the position of IAT addr.
5. Write iat jmp addr.

Script as follows:

```
init:
alloc 1000
mov top_iat, $ RESULT
mov top_jump, 0040131C

start:
mov [top_iat +88]
#
F05FF1776F6EF177805BF177896FF177000000004224807CEE1E807CA1B6807CDACD817C00000000AEB6417E1DB6417EF054427EEAD6417E83F3427ECED6417EF9D7417
EC959427E0CB1437EA3C7437E09B6417EF5B5417E #
process_jump_iat:
Cmp [top_jump], 0
je the_end
mov tmp_addr, [top_jump +2], 4
and tmp_addr, FF
add tmp_addr, top_iat
mov [top_jump +2], tmp_addr
add top_jump, 6
jmp process_jump_iat

the_end:
ret
```

To this, the work is nearly complete, 1 fix only a few more seats (in the scripts that have gen), which I also work full time then, so ... the rest own people handling small. What can post questions: D
Wishing you happy 1 day ... hehehehe

Game Over!

Translated and written by: **hoadongnoi**

Author: *Marcus*

Information: Unpacking for Newbie

Target: **Hexworkshop Base Converter**

Available: <http://www.reaonline.net>

Tools: Ollydbg, PEID, Import Reconstructor

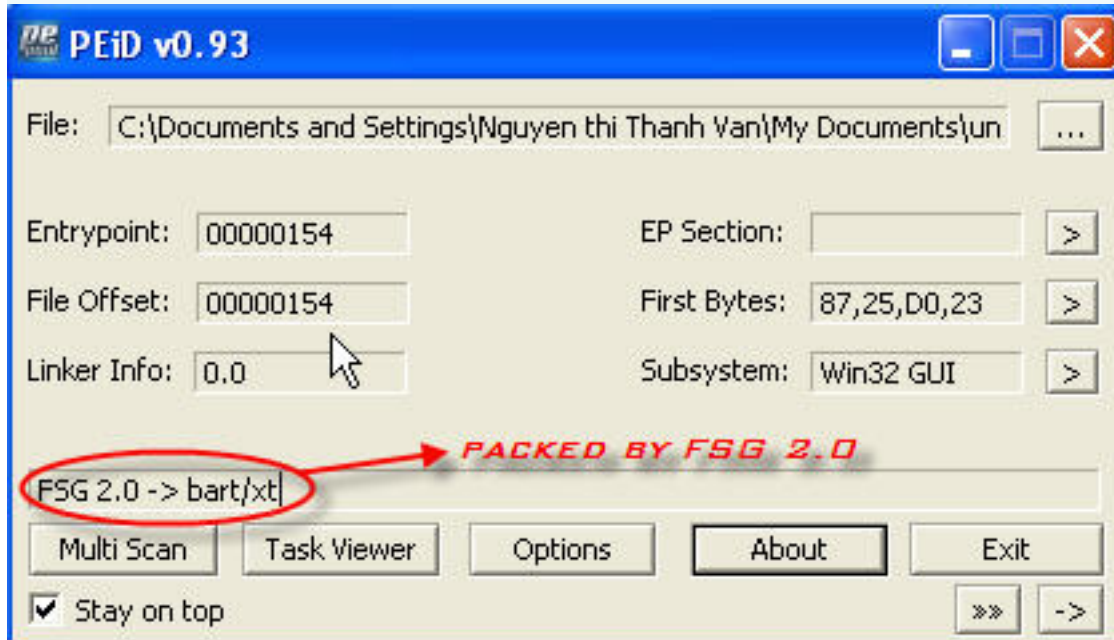
Protection: **FSG 2.0**

Level: Beginner

Category: Manual unpacking

Introduction:

As its name **Hexworkshop Base converter** is 1 converter tools for the data types such as byte, unsigned byte, short, unsigned short, long, Converter between us Hex, decimal, Binary. Original file not packed, but Target was packed with **FSG** to **2.0** for this tut. Now! Let's go! Used to Detect **v0.93 Peid** we have the following information:



The target was the author packed with **FSG 2.0 -> bar / XT**. Load it in Olly, Olly if any notice, just click OK. Olly will stop at EntryPoint:



Normally we will have 2 hours to unpack FSG. That is to use tools to unpack & unpack manually. In this tut we will only consider how to unpack manually. Now if you use F8 to trace you'll see it has 1 loop forever. At the last place where the end of the loop, looking down slightly less than 1 you will find:

004001C9	8B07	MOV	EAX, DWORD PTR DS:[EDI]	
004001CC	40	INC	EAX	
004001CD	78 F3	JS	SHORT bconv32[.004001C2]	
004001CF	75 03	JNZ	SHORT bconv32[.004001D4]	
004001D1	FF63 0C	JMP	NEAR DWORD PTR DS:[EBX+C]	<i>Always jumps here</i>
004001D4	50	PUSH	EAX	
004001D5	55	PUSH	EBP	
004001D6	FF53 14	CALL	NEAR DWORD PTR DS:[EBX+14]	
004001D9	AB	STOS	DWORD PTR ES:[EDI]	
004001DA	EB EE	JMP	SHORT bconv32[.004001CA]	
004001DC	33C9	XOR	ECX, ECX	
004001DE	41	INC	ECX	
004001DF	FF13	CALL	NEAR DWORD PTR DS:[EBX]	
004001E1	13C9	ADC	ECX, ECX	
004001E3	FF13	CALL	NEAR DWORD PTR DS:[EBX]	
004001E5	72 F8	JB	SHORT bconv32[.004001DF]	
004001E7	C3	RETN		

You will have the question is why I know that, Well, now you set BP at JMP this order:

004001DA ^ \ EE EB JMP SHORT bconv32 [.004001 CA ==> Set BP here

Then press F9 to Run, Olly ice will set points in the BP. Press F8 to see you will have 1 loop forever, and always command JNZ SHORT jump to 1, it ignored orders JMP before it

004001D1 FF63 0C JMP NEAR DWORD PTR DS: [EBX + C]

JMP command and this seems to do is never done. So we try to think it always perform JNZ orders to 1 address to check what it is & what to do but never implemented the command JMP 004001CF / 75 03 JNZ SHORT bconv32 [.004001 D4 ==> **checks is all done? If yes then dont jump and go to next address**

004001D1 | FF63 0C JMP NEAR DWORD PTR DS: [EBX + C] ==> **Jump to the OEP**

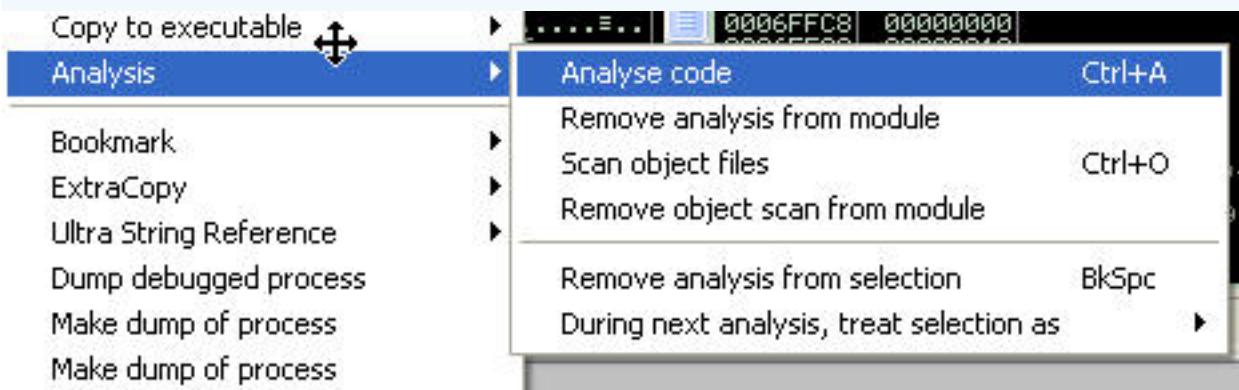
Since then we understand that address: "004001D1" is the address to jump to OEP ==> Set BP at JMP lenh this:

004001CC	40	INC	EAX	
004001CD	78 F3	JS	SHORT bconv32[.004001C2]	
004001CF	75 03	JNZ	SHORT bconv32[.004001D4]	
004001D1	FF63 0C	JMP	NEAR DWORD PTR DS:[EBX+C]	<i>jump to OEP ==> Set BP here</i>
004001D4	50	PUSH	EAX	
004001D5	55	PUSH	EBP	
004001D6	FF53 14	CALL	NEAR DWORD PTR DS:[EBX+14]	
004001D9	AB	STOS	DWORD PTR ES:[EDI]	
004001DA	EB EE	JMP	SHORT bconv32[.004001CA]	
004001DC	33C9	XOR	ECX, ECX	
004001DE	41	INC	ECX	

Press F9 (Run), Olly will be ice at the BP. Using F8 (1 time) to trace more. Wow, they will have to be OEP

0040C865	55	DB	55	CHAR 'U'
0040C866	8B	DB	8B	
0040C867	EC	DB	EC	
0040C868	6A	DB	6A	CHAR 'j'
0040C869	FF	DB	FF	
0040C86A	68	DB	68	CHAR 'h'
0040C86B	68	DB	68	CHAR 'h'
0040C86C	27	DB	27	CHAR ''
0040C86D	43	DB	43	CHAR 'C'
0040C86E	00	DB	00	
0040C86F	68	DB	68	CHAR 'h'
0040C870	74	DB	74	CHAR 't'

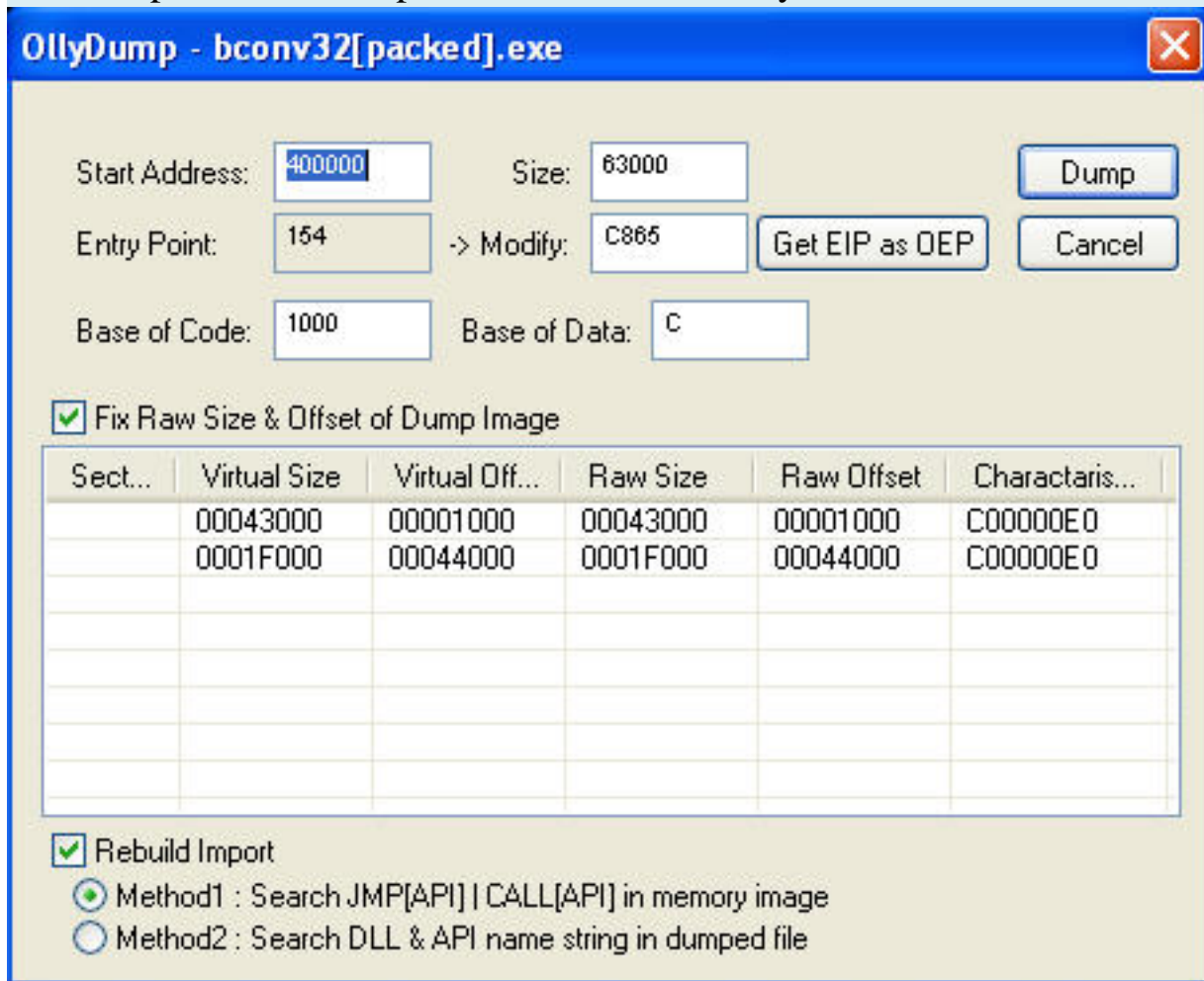
Click to select Analysis / Analyse code:



You will see the real code in it:

0040C865	. 55	PUSH	ESP	OLEAUT32.#392
0040C866	. 8BEC	MOV	EBP, ESP	
0040C868	. 6A FF	PUSH	-1	
0040C86A	. 68 68274300	PUSH	bconv32[.00432768	
0040C86F	. 68 741D4100	PUSH	bconv32[.00411D74	SE handler instal
0040C874	. 64:A1 00000000	MOV	EAX, DWORD PTR FS:[0]	
0040C87A	. 50	PUSH	EAX	
0040C87B	. 64:8925 00000000	MOV	DWORD PTR FS:[0], ESP	
0040C882	. 83EC 58	SUB	ESP, 58	
0040C885	. 53	PUSH	EBX	
0040C886	. 56	PUSH	ESI	
0040C887	. 57	PUSH	EDI	
0040C888	. 8965 E8	MOV	DWORD PTR SS:[EBP-18], ESP	
0040C88B	. FF15 FC024300	CALL	NEAR DWORD PTR DS:[4302FC]	kernel32.GetVersi

Now we proceed to dump the value in the Modify is $0040C865 - 400,000 = C865$

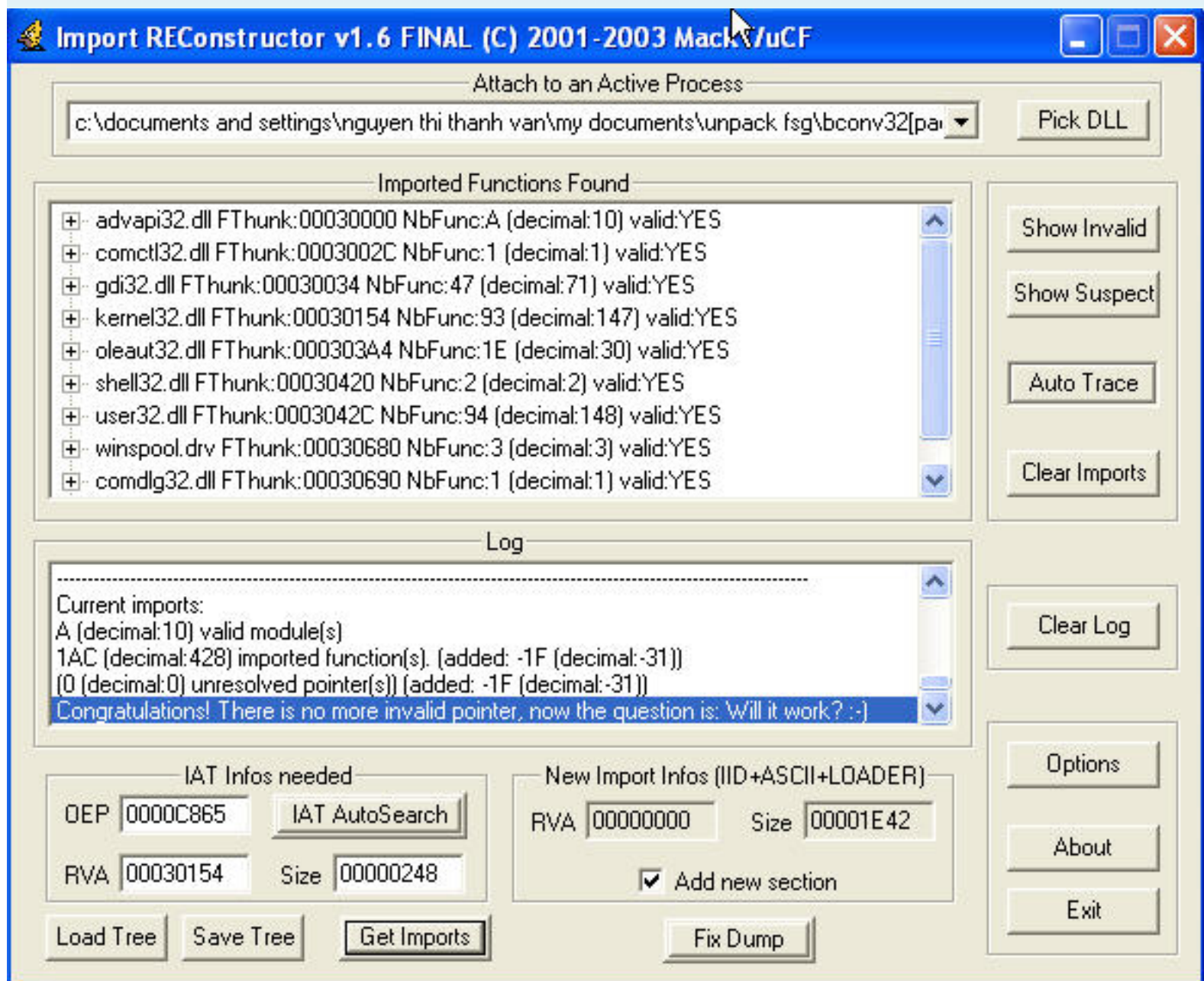


Do not jump to Olly, now we will rebuild Import table with Imprec:

IAT autosearch Click, click Get Imports. Rollover to highlight, click to select commands

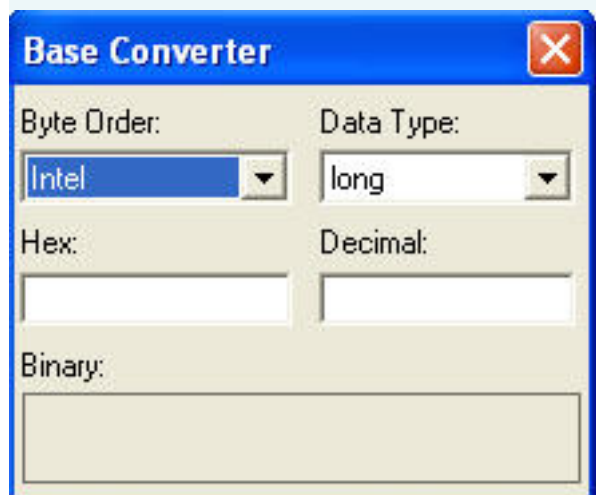
Advance / Get Api calls, click Show Invalid, more mouse click to select Cut Thunk (s). As

follows:



Click Fix to dump, then select the file you dump it with Olly

Done



Have fun! J

Greetz to thank: My family, Computer_Angel, Moonbaby, Zombie_Deathman, HacNho, Benina, RongChauA, Kienmanowar, TQN, QHQCkrker, Littleboy, The_lighthouse, dqtlN, tlandn, ectlong, Nini and ARTeam, ExeTools all my friend, and YOU!

Reverse Engineering Association

<http://www.REAonline.net>

hoadongnoi from REA

17/5/2004

Translated and written by: hoadongnoi

Author: *Marcus*

Information: Unpacking for Newbie

Target: **Hexworkshop Base Converter**

Available: <http://www.reaonline.net>

Tools: Ollydbg, PEID, Import Reconstructor

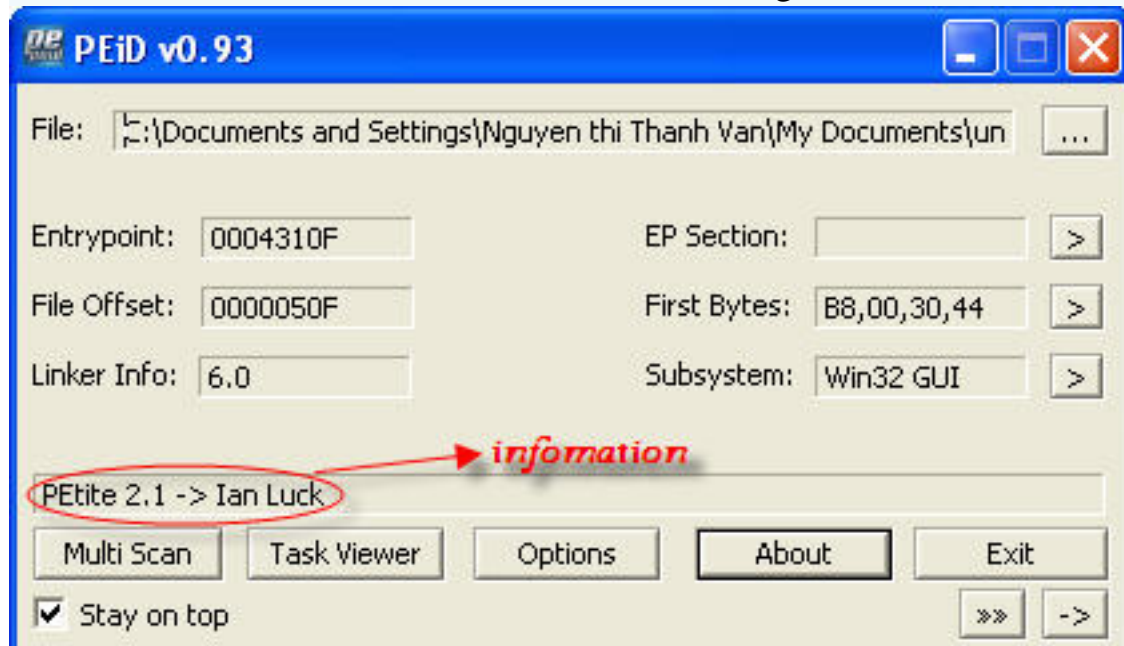
Protection: **Petite 2.3**

Level: Beginner

Category: Manual unpacking

Introduction:

As its name **Hexworkshop Base converter** is 1 converter tools for the data types such as byte, unsigned byte, short, unsigned short, long, Converter between us Hex, decimal, Binary. But the important this is tut we learned how to unpack Petite 2.3 with Manual how. Now, let's go! Used to Detect **v0.93 Peid** we have the following information:



The target was the author packed with **Petite 2.3** Peid but then the report is **Petite 2.1**. oh! ko problems go away, it is still meat as usual. Load it in Olly, Olly if any notice, just click OK. Select No Analysis, Olly will stop at EntryPoint:

Address	Hex dump	Disassembly	Comment
0044310F	B8 00304400	MOV EAX, bconv32[.00443000]	
00443114	6A 00	PUSH 0	
00443116	68 A3884300	PUSH bconv32[.004388A3]	
0044311B	64:FF35 00000000	PUSH DWORD PTR FS:[0]	
00443122	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
00443129	66:9C	PUSHFW	
0044312B	60	PUSHAD	
0044312C	50	PUSH EAX	
0044312D	8B08	MOV EBX, EAX	
0044312F	0300	ADD EAX, DWORD PTR DS:[EAX]	
00443131	68 A0820100	PUSH 182A0	
00443136	6A 00	PUSH 0	
00443138	FF50 1C	CALL NEAR DWORD PTR DS:[EAX+1C]	
0044313B	8943 08	MOV DWORD PTR DS:[EBX+8], EAX	
0044313E	8BC3	MOV EAX, EBX	
00443140	0300	ADD EAX, DWORD PTR DS:[EAX]	
00443142	68 70BC0000	PUSH 0BC70	
00443147	6A 00	PUSH 0	
00443149	FF50 1C	CALL NEAR DWORD PTR DS:[EAX+1C]	
0044314C	8BCC	MOV ECX, ESP	
0044314E	8DA0 70BC0000	LEA ESP, DWORD PTR DS:[EAX+BC70]	
00443154	8961 2E	MOV DWORD PTR DS:[ECX+2E], ESP	
00443157	53	PUSH EBX	
00443158	68 00004000	PUSH bconv32[.00400000]	ASCII "MZ"
0044315D	51	PUSH ECX	

EntryPoint

Now, our Strick is here, now we will have to find Decryption. We hit the F8 until we see 1 code as follows:

004431B9	80424 BF	ADD BYTE PTR SS:[ESP], 0BF	
004431BD	8B0B	MOV ECX, DWORD PTR DS:[EBX]	
004431BF	83C3 14	ADD EAX, 14	
004431C2	8B53 F0	MOV ECX, DWORD PTR DS:[EBX-10]	
004431C5	85D2	TEST EDX, EDX	
004431C7	74 F4	JE SHORT bconv32[.004431BD]	We're here
004431C9	8B4424 18	MOV EAX, DWORD PTR SS:[ESP+18]	
004431CD	8D3401	LEA ESI, DWORD PTR DS:[ECX+EAX]	
004431D0	8B6C24 1C	MOV EBP, DWORD PTR SS:[ESP+1C]	
004431D4	8B6D 08	MOV EBP, DWORD PTR SS:[EBP+8]	
004431D7	8B4B FC	MOV ECX, DWORD PTR DS:[EBX-4]	
004431DA	8BFD	MOV EDI, EBP	
004431DC	F3:A5	REP MOVSD DWORD PTR ES:[EDI], DWORD PTR DS:[ECX]	
004431DE	8BF5	MOV ESI, EBP	
004431E0	8B7B F4	MOV EDI, DWORD PTR DS:[EBX-C]	
004431E3	03F8	ADD EDI, EAX	
004431E5	53	PUSH EBX	
004431E6	52	PUSH EDX	
004431E7	57	PUSH EDI	
004431E8	55	PUSH EBP	
004431E9	E8 2D000000	CALL bconv32[.0044321B]	
004431EE	85C0	TEST EAX, EAX	
004431F0	74 F4	JE SHORT bconv32[.004430D2]	
004431F6	58	POP EAX	
004431F7	5B	POP EBX	
004431F8	59	POP ECX	
004431F9	5B	POP EBX	
004431FA	EB C1	JMP SHORT bconv32[.004431BD]	
004431FC	56	PUSH ESI	

Press F8 again a few more we will come here:

004431E8	55	PUSH EBP	
004431E9	E8 2D000000	CALL bconv32[.0044321B]	
004431FF	85C0	TEST EAX, EAX	
004431F0	74 F4	JE SHORT bconv32[.004430D2]	
004431F6	58	POP EAX	
004431F7	5B	POP EBX	
004431F8	59	POP ECX	
004431F9	5B	POP EBX	
004431FA	EB C1	JMP SHORT bconv32[.004431BD]	If checking done then Jump

JE commands it will Jump to the following code:

004430D2	8B6424 24	MOV	ESP, DWORD PTR SS:[ESP+24]	
004430D6	8B0424	MOV	EAX, DWORD PTR SS:[ESP]	
004430D9	83C4 26	ADD	ESP, 26	
004430DC	8D90 01010000	LEA	EDX, DWORD PTR DS:[EAX+101]	
004430E2	83C4 0C	ADD	ESP, 0C	
004430E5	6A 10	PUSH	10	
004430E7	8BD8	MOV	EBX, EAX	
004430E9	66:05 FA00	ADD	AX, 0FA	
004430ED	50	PUSH	EAX	
004430EE	52	PUSH	EDX	
004430EF	6A 00	PUSH	0	
004430F1	8B1B	MOV	EBX, DWORD PTR DS:[EBX]	
004430F3	FF13	CALL	NEAR DWORD PTR DS:[EBX]	
004430F5	6A FF	PUSH	-1	
004430F7	FF53 0C	CALL	NEAR DWORD PTR DS:[EBX+C]	
004430FA	45	INC	EBP	
004430FB	52	PUSH	EDX	
004430FC	52	PUSH	EDX	
004430FD	4F	DEC	EDI	
004430FE	52	PUSH	EDX	
004430FF	2100	AND	DWORD PTR DS:[EAX], EAX	
00443101	43	INC	EBX	
00443102	6F	OUTS	DX, DWORD PTR ES:[EDI]	I/O command
00443103	✓ 72 72	JB	SHORT bconv32[.00443177]	
00443105	✓ 75 70	JNZ	SHORT bconv32[.00443177]	
00443107	✓ 74 20	JE	SHORT bconv32[.00443129]	
00443109	44	INC	ESP	
0044310A	61	POPAD		
0044310B	✓ 74 61	JE	SHORT bconv32[.0044316E]	
0044310D	2100	AND	DWORD PTR DS:[EAX], EAX	
0044310F	B8 00304400	MOV	EAX, bconv32[.00443000]	
00443114	6A 00	PUSH	0	
00443116	68 A3884300	PUSH	bconv32[.004388A3]	
00443118	64:FF35 00000000	PUSH	DWORD PTR FS:[0]	
00443122	64:8925 00000000	MOV	DWORD PTR FS:[0], ESP	
00443129	66:9C	PUSHFW		
0044312B	60	PUSHAD		
0044312C	50	PUSH	EAX	
0044312D	8BD8	MOV	EBX, EAX	
0044312F	0300	ADD	EAX, DWORD PTR DS:[EAX]	

Here we see 2 orders PUSHFW and PUSHAD, such proven record has to be loaded, so we will question whether it will get out how. If you try to run the app you'll see it reported error:



Press Ctrl + F2 to load the target again to Olly, press F9 to run, press Shift + F9 to 2 times, then look back up a little over 1, we will see the following code:

004430EE	52	PUSH	EDX	
004430EF	6A 00	PUSH	0	
004430F1	8B1B	MOV	EBX, DWORD PTR DS:[EBX]	
004430F3	FF13	CALL	NEAR DWORD PTR DS:[EBX]	
004430F5	6A FF	PUSH	-1	
004430F7	FF53 0C	CALL	NEAR DWORD PTR DS:[EBX+C]	
004430FA	45	INC	EBP	
004430FB	52	PUSH	EDX	
004430FC	52	PUSH	EDX	
004430FD	4F	DEC	EDI	
004430FE	52	PUSH	EDX	
004430FF	2100	AND	DWORD PTR DS:[EAX], EAX	
00443101	43	INC	EBX	
00443102	6F	OUTS	DX, DWORD PTR ES:[EDI]	I/O command
00443103	72 FB	JB	SHORT bconv32[.00443100]	Modification of se
00443105	07	POP	ES	
00443106	5F	POP	EDI	
00443107	F3:AA	REP	STOS BYTE PTR ES:[EDI]	
00443109	61	POPAD		
0044310A	66:9D	POPFW		
0044310C	83C4 0C	ADD	ESP, 0C	
0044310F	- E9 5197FCFF	JMP	bconv32[.0040C865]	
00443114	- E9 61E6A277	JMP	kernel32.DeleteFileA	
00443119	- E9 16F1A377	JMP	kernel32.GetFullPathNameA	
0044311E	- E9 28FEA277	JMP	kernel32.SetFileTime	
00443123	- E9 8995A277	JMP	kernel32.UnlockFile	
00443128	- E9 D067A377	JMP	kernel32.ExitProcess	
0044312D	- E9 0507A377	JMP	kernel32.RaiseException	
00443132	- E9 3129A377	JMP	kernel32.GetLocalTime	
00443137	- E9 7CE5A177	JMP	kernel32.TerminateProcess	

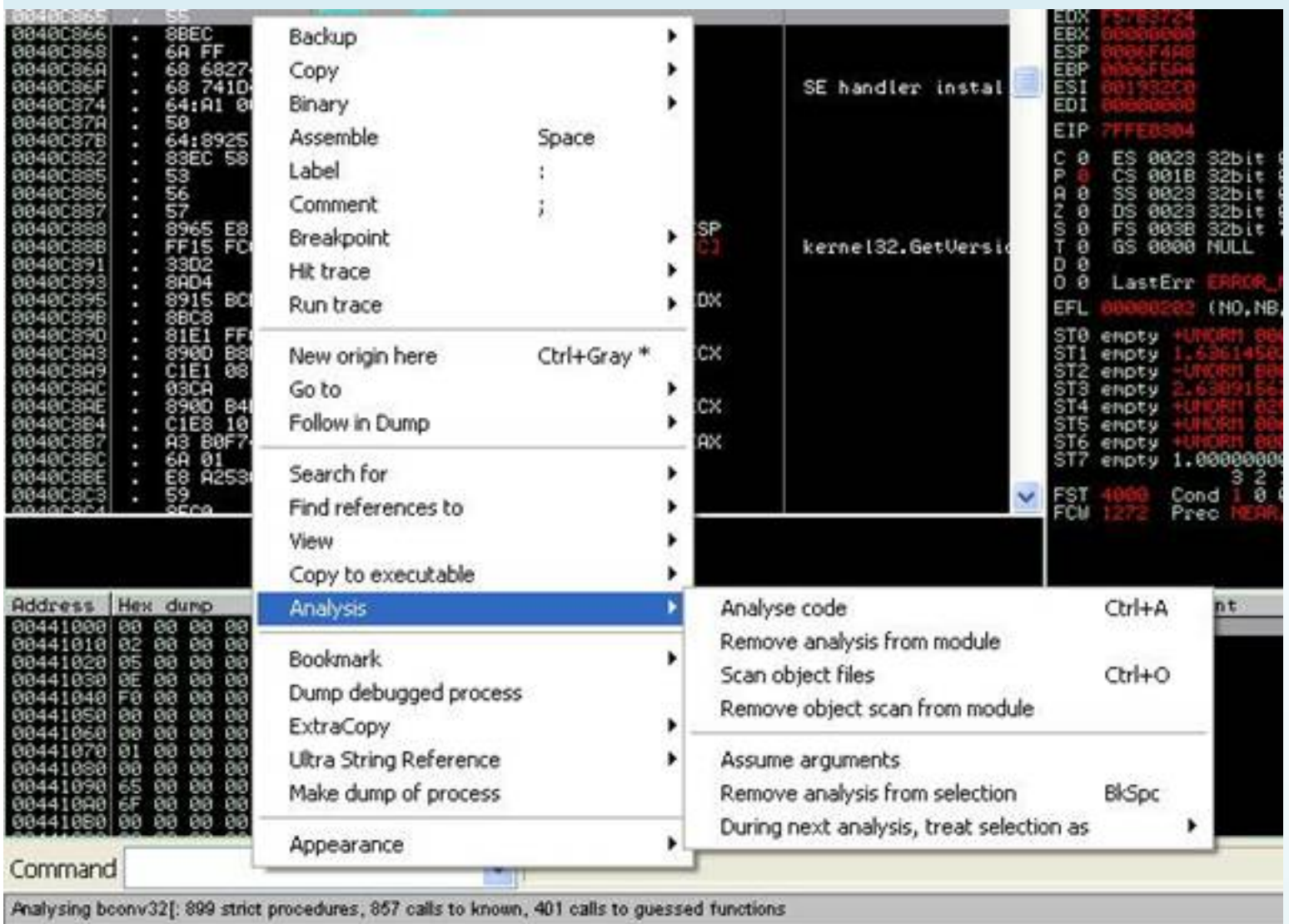
Wow here we found that 2 commands PUSHFW PUSHAD and has been converted into 2 commands POPFW and POPAD. And here we see the code it was like Pop:

00443109	61	POPAD		
0044310A	66:9D	POPFW		
0044310C	83C4 0C	ADD	ESP, 0C	
0044310F	- E9 5197FCFF	JMP	bconv32[.0040C865]	

Now set to vet the morning at the Jump command and press Enter, we will come here, real code:

0040C864	. C3	RETN		
0040C865	. 55	PUSH	EBP	
0040C866	. 8BEC	MOV	EBP, ESP	OEP
0040C868	. 6A FF	PUSH	-1	
0040C86A	. 68 68274300	PUSH	bconv32[.00432768]	

The OEP. Click to select Analysis / Analyse code:



Now we run the dump, in Olly we choose OllyDump / dump debugged process and dump it.

Then we rebuild the Imprec:

Click IAT autosearch

Click Get Imports

Go to the window highlight the text show

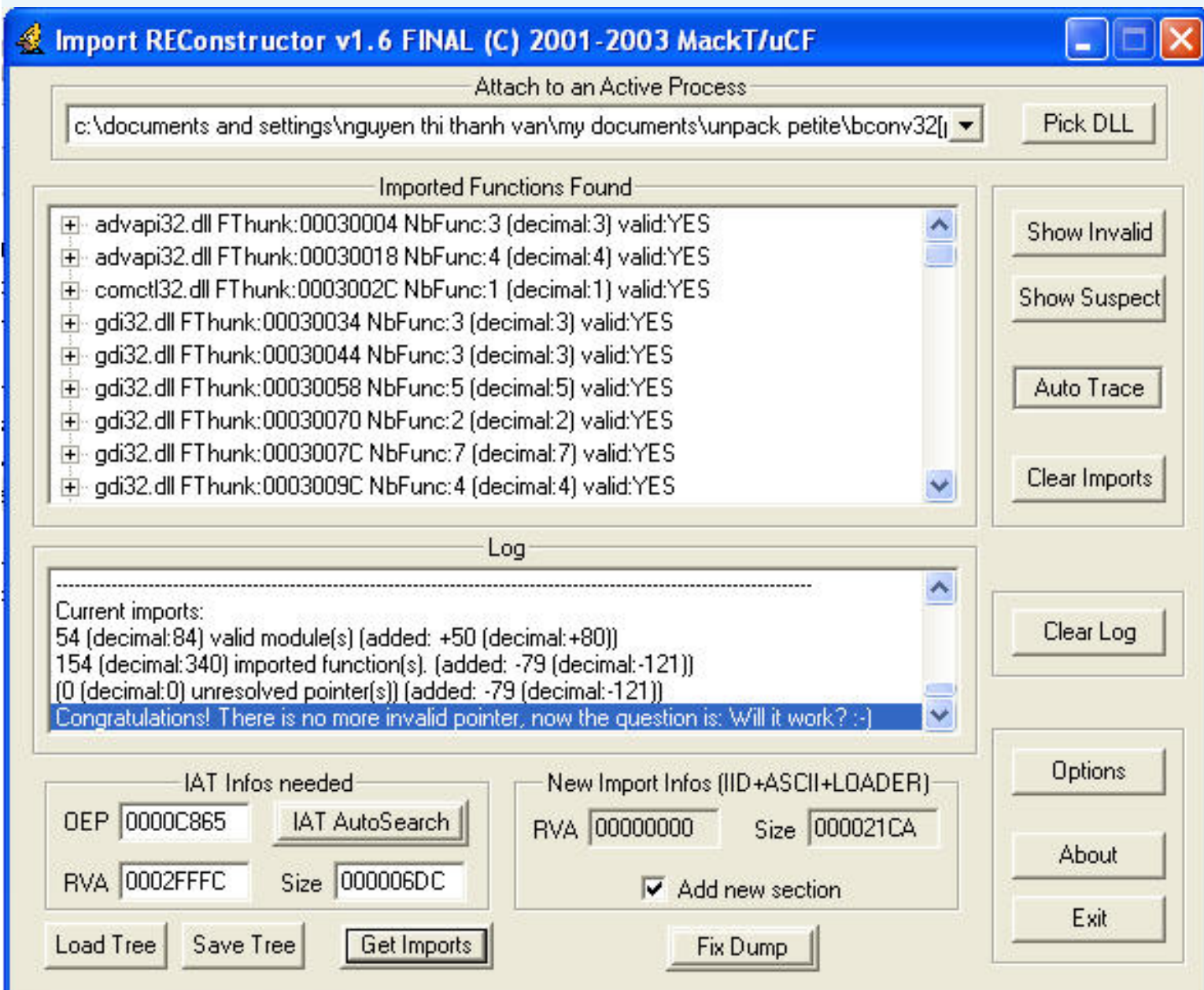
Right click>>>> Advance commands>>>> get the API calls

Click Show Invalid

Go to the window

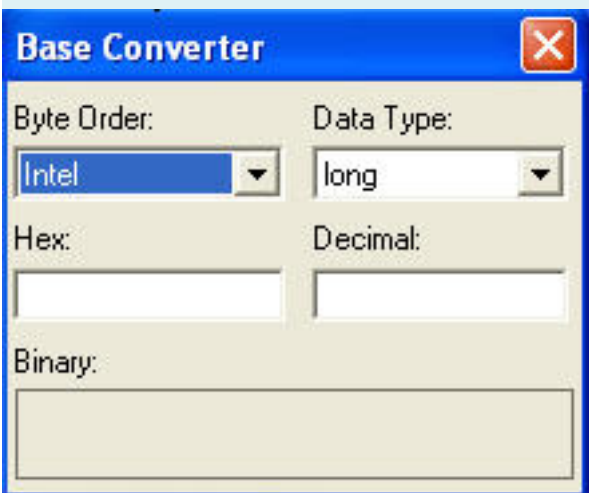
Right click>>> Cut chunk (s).

One result is like the image below



Then choose to Fixdump, select save as to just dump file.

Done!



Have fun! J

Greetz to thank: My family, Computer_Angel, Moonbaby, Zombie_Deathman, HacNho, Benina,

RongChauA, Kienmanowar, TQN, QHQCrker, Littleboy, The_lighthouse, dqtln, tlandn, ectlong,
Nini and ARTeam, ExeTools all my friend, and YOU!

Reverse Engineering Association

<http://www.REAonline.net>

hoadongnoi from REA

12/5/2004

Translated and written by: **hoadongnoi**

Author: *Marcus*

Information: Unpacking for Newbie

Target: **Hexworkshop Base Converter**

Available: <http://www.reaonline.net>

Tools: Ollydbg, PEID, Import Reconstructor

Protection: **Mew 11 SE v1.2**

Level: Beginner

Category: Manual unpacking

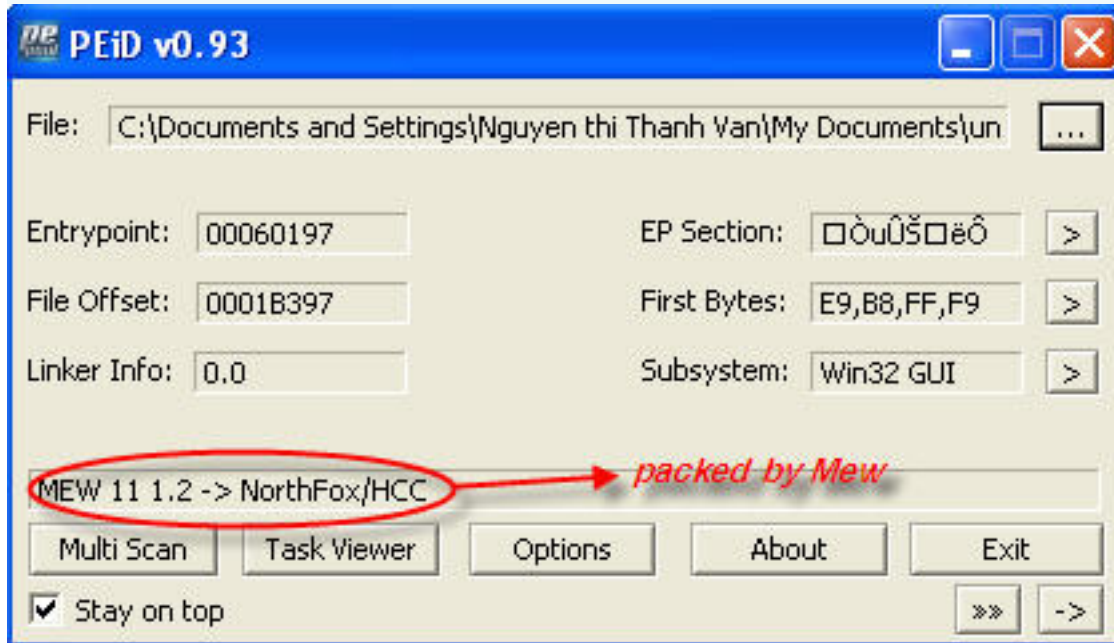
Introduction:

As its name **Hexworkshop Base converter** is 1 converter tools for the data types such as byte, unsigned byte, short, unsigned short, long, Converter between us Hex, decimal, Binary.

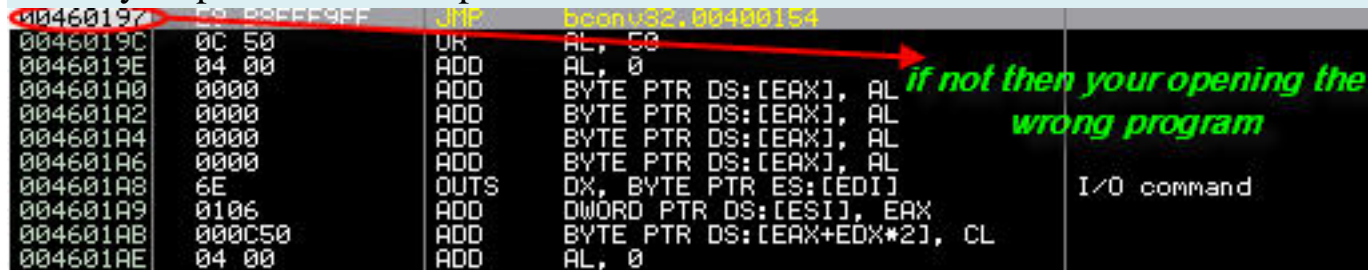
Original file not packed, but Target was packed with Mew 11 SE v1.2 is to serve for this tut.

Now! Let's go!

Used to Detect **v0.93 Peid** we have the following information:



The target was the author packed with **Mew 11 1.2 -> NorthFox / HCC**. Load it in Olly, you'll see Olly stop order in 1 Jump



To view the command Jump this it will jump to where we go press F8, then we will see the following code:

00400154	BE 1C504400	MOV	ESI, bconv32.0044501C	
00400159	8B0E	MOV	EBX, ESI	
0040015B	AD	LODS	DWORD PTR DS:[ESI]	
0040015C	AD	LODS	DWORD PTR DS:[ESI]	
0040015D	50	PUSH	EAX	
0040015E	AD	LODS	DWORD PTR DS:[ESI]	
0040015F	97	XCHG	EAX, EDI	
00400160	B2 80	MOV	DL, 80	
00400162	A4	MOVS	BYTE PTR ES:[EDI], BYTE PTR DS:	
00400163	B6 80	MOV	DH, 80	
00400165	FF13	CALL	NEAR DWORD PTR DS:[EBX]	
00400167	73 F9	JNB	SHORT bconv32.00400162	

Drag the mouse pointer to the little under 1 will order Return to me by this code. Set in the BP

004001F8	AB	STOS	DWORD PTR ES:[EDI]	
004001F9	85C0	TEST	EAX, EAX	
004001FB	75 E5	JNZ	SHORT bconv32.004001E2	
004001FD	C3	RETN		
004001FE	0000	ADD	BYTE PTR DS:[EAX], AL	Set BP here
00400200	0000	ADD	BYTE PTR DS:[EAX], AL	
00400202	0000	ADD	BYTE PTR DS:[EAX], AL	
00400204	0000	ADD	BYTE PTR DS:[EAX], AL	
00400206	0000	ADD	BYTE PTR DS:[EAX], AL	
00400208	0000	ADD	BYTE PTR DS:[EAX], AL	

Press F9 to run the program. Olly will ice at BP we've set. Use F8 to trace, we will be up to code follows:

0040C865	55	DB	55	CHAR 'U'
0040C866	8B	DB	8B	
0040C867	EC	DB	EC	
0040C868	6A	DB	6A	CHAR 'j'
0040C869	FF	DB	FF	
0040C86A	68	DB	68	CHAR 'h'
0040C86B	68	DB	68	CHAR 'h'
0040C86C	27	DB	27	CHAR '''
0040C86D	43	DB	43	CHAR 'C'
0040C86E	00	DB	00	
0040C86F	68	DB	68	CHAR 'h'
0040C870	74	DB	74	CHAR 't'

Click to vet in the morning, select Analysis / Analyse code, it will be to the real code that:

Copy to executable				..u.....	0006FFCC	00000010
Analysis				Analyse code	Ctrl+A	
Bookmark				Remove analysis from module		
ExtraCopy				Scan object files	Ctrl+O	
Ultra String Reference				Remove object scan from module		
Make dump of process				Remove analysis from selection	BkSpC	
0040C865	55	PUSH	ESP	OLEAUT32.#392		
0040C866	8BEC	MOV	EBP, ESP			
0040C868	6A FF	PUSH	-1			
0040C86A	68 68274300	PUSH	bconv32.00432768			
0040C86F	68 741D4100	PUSH	bconv32.00411D74			
0040C874	64:A1 00000000	MOV	EAX, DWORD PTR FS:[0]	SE handler instal		
0040C87A	50	PUSH	EAX			
0040C87B	64:8925 00000000	MOV	DWORD PTR FS:[0], ESP			
0040C882	83EC 58	SUB	ESP, 58			

Now we proceed to dump the value in the Modify is 0040C865 - 400,000 = C865

OllyDump - bconv32[packed].exe

Start Address: Size:

Entry Point: -> Modify:

Base of Code: Base of Data:

☒ Fix Raw Size & Offset of Dump Image

Sect...	Virtual Size	Virtual Off...	Raw Size	Raw Offset	Charactaris...
	00043000	00001000	00043000	00001000	C00000E0
	0001F000	00044000	0001F000	00044000	C00000E0

☒ Rebuild Import

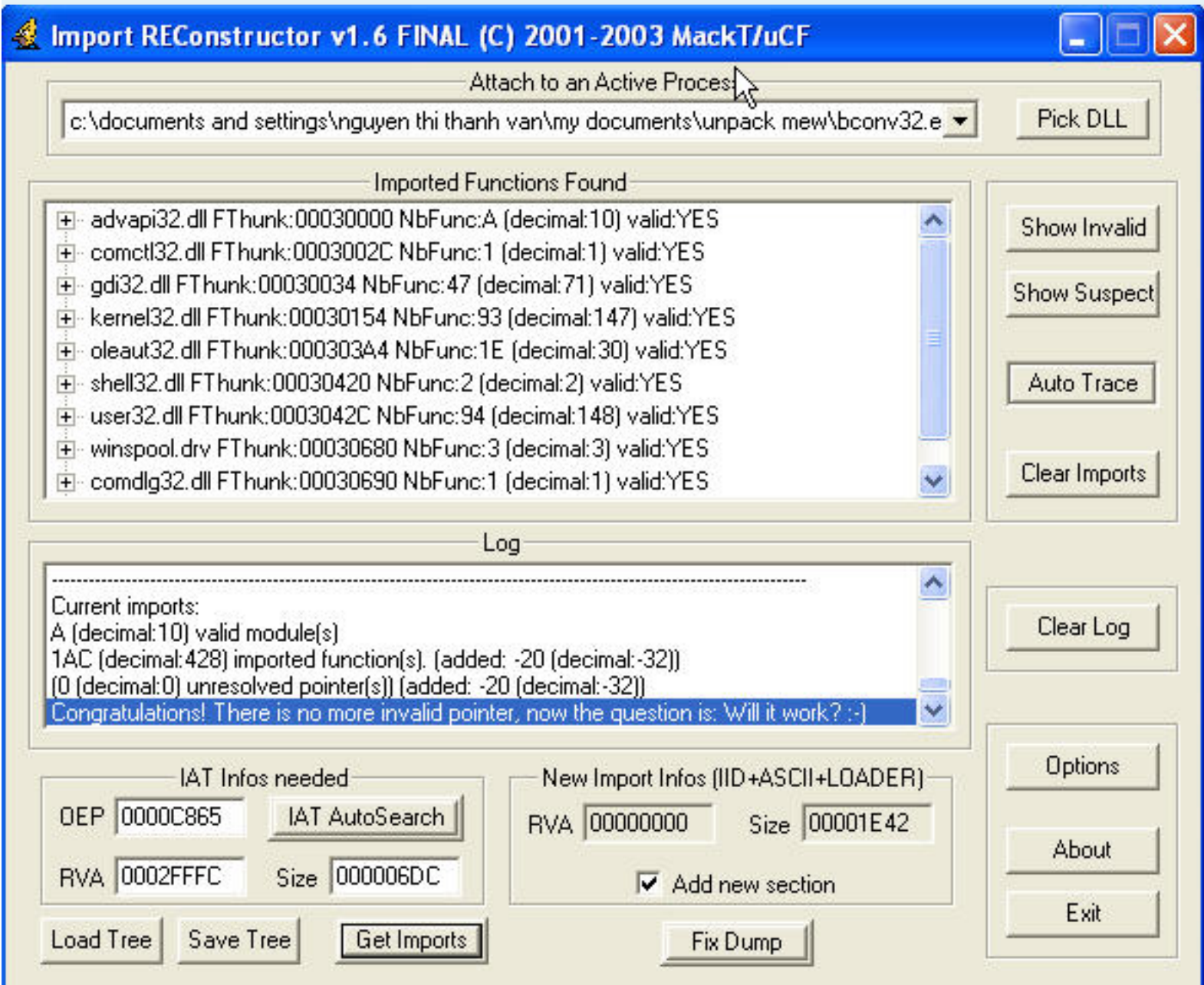
☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

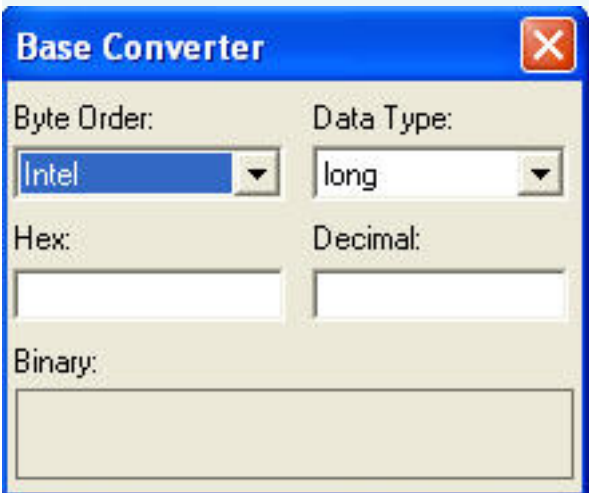
Do not jump to Olly, now we will rebuild Import table with Imprec:

IAT autosearch Click, click Get Imports. Rollover to highlight, click to select commands

Advance / Get Api calls, click Show Invalid, more mouse click to select Cut Thunk (s). As follows:



Click Fix to dump, then select the file you dump it with Olly
 Done



Have fun! J

Greetz to thank: My family, Computer_Angel, Moonbaby, Zombie_Deathman, HacNho, Benina, RongChauA, Kienmanowar, TQN, QHQCkrker, Littleboy, The_lighthouse, dqtn, tlandn, ectlong, Nini and ARTeam, ExeTools all my friend, and YOU!

Reverse Engineering Association

<http://www.REAonline.net>

hoadongnoi from REA

17/5/2004



Manual Fixing IAT - NTKRNL Packer

- I. Introduction:
- Olly II. Config:
- OEP III. Find:
- IV. Fix IAT:
 - 1. Analyze:
 - 2nd Code:
- V. Ending

I. Introduction:

Welcome kOn pa, a long time but they do not interfere reverse tool. Dom Banner on the other is the brain child bik first Mụ mam muss, school ko lo school, worry sweetheart ko ah, hix hix. Natural tut just read his opinion, hand itch day chả they are also a few stories not done with the packer NTKRNL, tut the reversing.be children read but also each time that the drone can do as follow tut ... Now Fortunately tut by playing him in the work. But to say they want to do is dump file that is IAT's packer fix this today, brothers and we do not fix that xài plugin nhé: D.

Target is unpackme code and have the pack, down here:

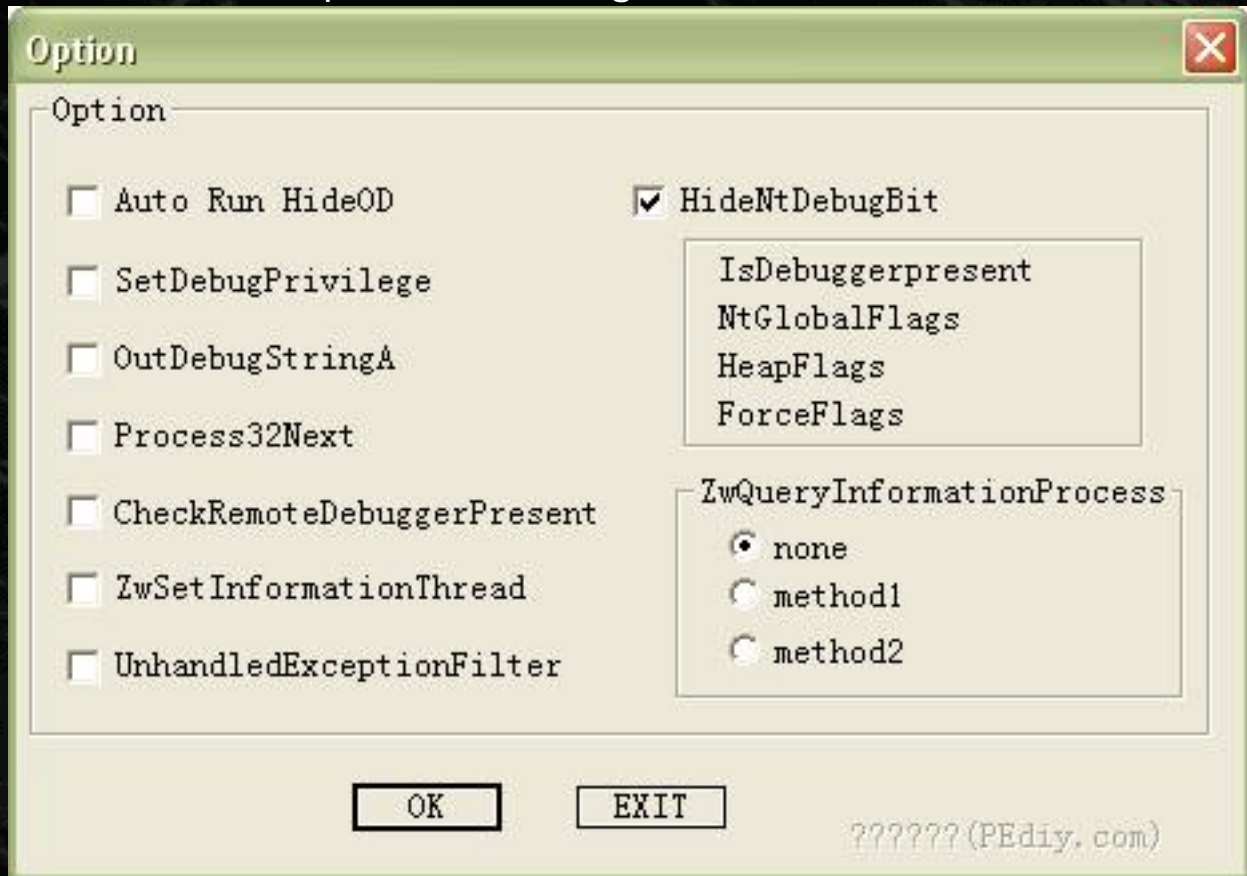
<http://reaonline.net/forum/showthread.php?t=4626>

Now the GOOOOOOOOO

II. Config Olly:

English children do not remember what the desire xài mod Olly is too much nhé. Mod multiple sources to make it before this packer. She new test through Olly OllyShadow often are both. Option for Olly as he is presented

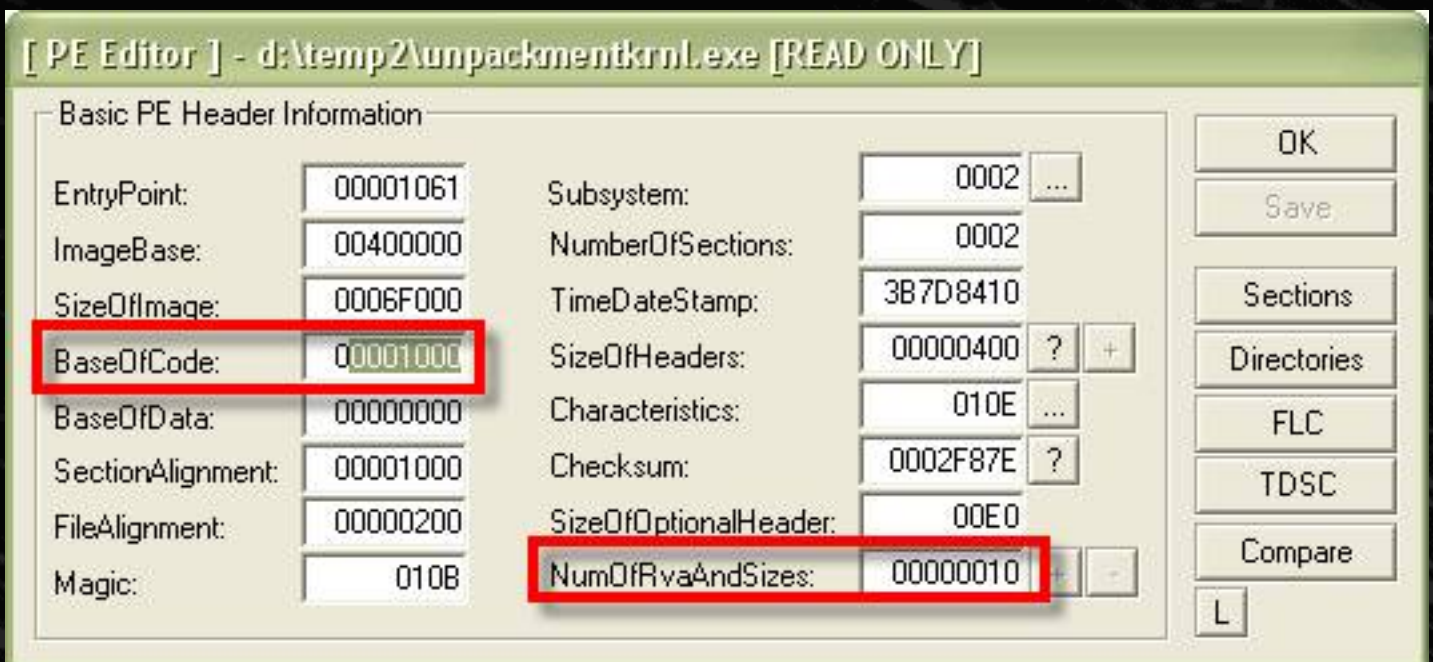
in, check all exception, add more to HideOD plugin. That's enough, and how to fix IAT also thanks 1 part HideOD again.



III. Find OEP:

This also does not need to say much because they only focus on only fix IAT. This is the summary:

1. Fix the PE header:



Using LordPE fix the image above. (If you use often Olly), Olly has the

mod is not necessary.

2. Target load, stop at EP:

Address	Hex dump	Disassembly
00401061 U	68 79CB4100	PUSH UnpackMe.0041CB79
00401066	E8 01000000	CALL UnpackMe.0040106C
0040106B	C3	RETN
0040106C	C3	RETN
0040106D	8C36	MOV WORD PTR DS:[ESI],SEG?
0040106F	5B	POP EBX
00401070	0AEF	OR CH,BH

3. Hide in HideOD Plugin.

4. F9 Run, break:

Address	Hex dump	Disassembly
00330000	C3	RETN
00330001	0000	ADD BYTE PTR DS:[EAX],AL
00330003	0000	ADD BYTE PTR DS:[EAX],AL

5. Change C3 à CC. RETN Because this is the exception to the packer, to detect debugger, INT3 to change to more normal 1 exception, the trick is how packer 1 delicious fresh, and the unpack continue impartiality.

6. After the change, BP LoadLibraryA, F9, break:

Address	Hex dump	Disassembly
7C801D77 k	8BFF	MOV EDI,EDI
7C801D79	55	PUSH EBP
7C801D7A	8BEC	MOV EBP,ESP
7C801D7C	837D 08 00	CMP DWORD PTR SS:[EBP+8],0
7C801D80	53	PUSH EBX
7C801D81	56	PUSH ESI
7C801D82	74 14	JE SHORT kernel32.7C801D98
7C801D84	68 C0E0807C	PUSH kernel32.7C80E0C0
7C801D89	FF75 0A	PUSH DWORD PTR SS:[EBP+8]

Images break on Win XP SP1 and SP2 will vary jog.

7. Leave here to BP, Alt-F9 to return to the program code:

Address	Hex dump	Disassembly
0041F5F7	FF75 F4	PUSH DWORD PTR SS:[EBP-C]
0041F5FA	E8 FA000000	CALL UnpackMe.0041F6F9
0041F5FF	85C0	TEST EAX,EAX
0041F601	74 77	JE SHORT UnpackMe.0041F67A
0041F603	8BF8	MOV EDI,EAX
0041F605	8B4D F8	MOV ECX,DWORD PTR SS:[EBP-8]
0041F608	8B11	MOV EDX,DWORD PTR DS:[ECX]
0041F60A	A5D2	TEST EDX,EDX

8. Pull down to find RETN 4, BP set, F9, break:

0041F65H	C783 84000000 00000000	MOV DWORD PTR DS:[EBX+84],0
0041F694	8B45 EC	MOV EAX,DWORD PTR SS:[EBP-14]
0041F697	8BE5	MOV ESP,EBP
0041F699	5D	POP EBP
0041F69A	C2 0400	RETN 4
0041F69D	55	PUSH EBP
0041F69E	8BEC	MOV EBP,ESP
0041F6A0	83C4 B0	ADD ESP,-50
0041F6A2	FF75 0A	PUSH DWORD PTR SS:[EBP+8]

9. Leave BP, F7 (or F8) to return to

10. Search CALL EAX just below:

Address	Hex dump	Disassembly
0041F3D9	8945 C4	MOV DWORD PTR SS:[EBP-3C],EAX
0041F3DC	8B45 F4	MOV EAX,DWORD PTR SS:[EBP-C]
0041F3DF	0340 3C	ADD EAX,DWORD PTR DS:[EAX+3C]
0041F3E2	8D58 28	LEA EBX,DWORD PTR DS:[EAX+28]
0041F3E5	8B40 28	MOV EAX,DWORD PTR DS:[EAX+28]
0041F3E8	0345 F4	ADD EAX,DWORD PTR SS:[EBP-C]
0041F3EB	C703 00000000	MOV DWORD PTR DS:[EBX],0
0041F3F1	8B5D F4	MOV EBX,DWORD PTR SS:[EBP-C]
0041F3F4	035B 3C	ADD EBX,DWORD PTR DS:[EBX+3C]
0041F3F7	66:F743 16 0020	TEST WORD PTR DS:[EBX+16],2000
0041F3FD	74 10	JE SHORT UnpackMe.0041F40F
0041F3FF	6A 00	PUSH 0
0041F401	6A 01	PUSH 1
0041F403	FF75 F4	PUSH DWORD PTR SS:[EBP-C]
0041F406	FFD0	CALL EAX
0041F408	74 03	JE SHORT UnpackMe.0041F40F
0041F40C	8B45 F4	MOV EAX,DWORD PTR SS:[EBP-C]

11. Running to CALL EAX, F7 go invisible:

Address	Hex dump	Disassembly
003A1E4C	6A 0C	PUSH 0C
003A1E4E	68 C06B3900	PUSH 396BC0
003A1E53	E8 742B0000	CALL 003A49CC
003A1E58	33C0	XOR EAX,EAX
003A1E5A	40	INC EAX
003A1E5B	8945 E4	MOV DWORD PTR SS:[EBP-1C],EAX
003A1E5E	8B75 0C	MOV ESI,DWORD PTR SS:[EBP+C]

12. Ctrl-B command find JMP EAX through byte group: 61 FF E0

13. Search times to 2:

Address	Hex dump	Disassembly
0039B353	83C4 3C	ADD ESP,3C
0039B356	A1 68983A00	MOV EAX,DWORD PTR DS:[3A9868]
0039B35B	894424 1C	MOV DWORD PTR SS:[ESP+1C],EAX
0039B35F	61	POPAD
0039B360	FFE0	JMP EAX
0039B362	5F	POP EDI
0039B363	5E	POP ESI
0039B364	5B	POP EBX
0039B365	5D	POP EBP
0039B366	C3	RET

14. BP set at JMP EAX, F9, the flash screen, break. F7 to OEP:

Address	Hex dump	Disassembly
004017A0	55	PUSH EBP
004017A1	8BEC	MOV EBP,ESP
004017A3	6A FF	PUSH -1
004017A5	68 30214200	PUSH UnpackMe.00422130
004017AA	68 A0364000	PUSH UnpackMe.004036A0
004017AF	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
004017B5	50	PUSH EAX
004017B6	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
004017BD	83C4 A4	ADD ESP,-5C
004017C0	53	PUSH EBX
004017C1	56	PUSH ESI
004017C2	57	PUSH EDI
004017C3	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP
004017C6	FF15 48A24200	CALL DWORD PTR DS:[42A248]
004017CC	A3 607D4200	MOV DWORD PTR DS:[427D60],EAX
004017D1	A1 607D4200	MOV EAX,DWORD PTR DS:[427D60]
004017D6	C1E8 08	SHR EAX,8

OEP

Done. Start time fix IAT minutes.

IV.Fix IAT:

1. Analyze:

There are many ways to find places to save IAT we know. FF 25 hours of 1 very quickly (or go to the right under CALL OEP):

Address	Hex dump	Disassembly
0040165D	CC	INT3
0040165E	CC	INT3
0040165F	CC	INT3
00401660	- FF25 54A34200	JMP DWORD PTR DS:[42A354]
00401666	- FF25 50A34200	JMP DWORD PTR DS:[42A350]
0040166C	- FF25 4CA34200	JMP DWORD PTR DS:[42A34C]
00401672	- FF25 48A34200	JMP DWORD PTR DS:[42A348]
00401678	- FF25 44A34200	JMP DWORD PTR DS:[42A344]
0040167E	- FF25 40A34200	JMP DWORD PTR DS:[42A340]
00401684	- FF25 3CA34200	JMP DWORD PTR DS:[42A33C]
0040168A	- FF25 38A34200	JMP DWORD PTR DS:[42A338]
00401690	- FF25 34A34200	JMP DWORD PTR DS:[42A334]
00401696	- FF25 30A34200	JMP DWORD PTR DS:[42A330]
0040169C	- FF25 2CA34200	JMP DWORD PTR DS:[42A32C]
004016A2	- FF25 28A34200	JMP DWORD PTR DS:[42A328]
004016A8	- FF25 24A34200	JMP DWORD PTR DS:[42A324]
004016AE	- FF25 20A34200	JMP DWORD PTR DS:[42A320]
004016B4	- FF25 1CA34200	JMP DWORD PTR DS:[42A31C]
004016BA	- FF25 18A34200	JMP DWORD PTR DS:[42A318]
004016C0	- FF25 14A34200	JMP DWORD PTR DS:[42A314]
004016C6	- FF25 10A34200	JMP DWORD PTR DS:[42A310]
004016CC	- FF25 0CA34200	JMP DWORD PTR DS:[42A30C]
004016D2	- FF25 08A34200	JMP DWORD PTR DS:[42A308]
004016D8	CC	INT3

Follow in dump à Memory Address:

Address	Value	Comment
0042A1DC	00000000	
0042A1E0	00000000	
0042A1E4	00000000	
0042A1E8	00F40A50	IAT Start
0042A1EC	00F40AD4	
0042A1F0	00F40B58	
0042A1F4	00F40BDC	
0042A1F8	00F40C60	
0042A1FC	00F40CE4	
0042A200	00F40D68	
0042A204	00F40DEC	
0042A208	00F40E70	
0042A20C	00F40EF4	
0042A210	00F40F78	
0042A214	00F40FFC	
0042A218	00F41080	
0042A21C	00F41104	
0042A220	00F41188	
0042A224	00F4120C	
0042A228	00F41290	
0042A22C	00F41314	
0042A230	00F41398	
0042A234	00F4141C	
0042A30C	00F40084	
0042A310	00F40108	
0042A314	00F4018C	
0042A318	00F40210	
0042A31C	00F40294	
0042A320	00F40318	
0042A324	00F4039C	
0042A328	00F40420	
0042A32C	00F404A4	
0042A330	00F40528	
0042A334	00F405AC	
0042A338	00F40630	
0042A33C	00F406B4	
0042A340	00F40738	
0042A344	00F407BC	
0042A348	00F40840	
0042A34C	00F408C4	
0042A350	00F40948	
0042A354	00F409CC	IAT End
0042A358	00000000	
0042A35C	00000000	
0042A360	00000000	
0042A364	00000000	

Tentatively identified as such. Next time we will find exactly through NTKRNL always.

To find out, the IAT has been encrypt both father mistake again any function L. Now we return to OEP, see CALL function just below:

Address	Hex dump	Disassembly
004017A0	55	PUSH EBP
004017A1	8BEC	MOV EBP,ESP
004017A3	6A FF	PUSH -1
004017A5	68 30214200	PUSH UnpackMe.00422130
004017AA	68 A0364000	PUSH UnpackMe.004036A0
004017AF	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
004017B5	50	PUSH EAX
004017B6	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
004017BD	83C4 A4	ADD ESP,-5C
004017C0	53	PUSH EBX
004017C1	56	PUSH ESI
004017C2	57	PUSH EDI
004017C3	8965 E8	MOV DWORD PTR SS:[EBP+18],ESP
004017C6	FF15 48A24200	CALL DWORD PTR DS:[42A248]
004017C8	02 607D4200	MOVL DWORD PTR DS:[427D60],EAX
004017D1	A1 607D4200	MOV EAX,DWORD PTR DS:[427D60]
004017D6	C1E8 08	SHR EAX,8
004017D9	25 FF000000	AND EAX,0FF
004017DE	A3 607D4200	MOV DWORD PTR DS:[427D6C],EAX

I trace this to CALL F7 and go invisible, is the IAT decrypt:

Address	Hex dump	Disassembly
00F416B0	60	PUSHAD
00F416B1	E8 00000000	CALL 00F416B6
00F416B6	5D	POP EBP
00F416B7	81ED 8F8F3900	SUB EBP,398F8F
00F416B8	FFB5 01903900	PUSH DWORD PTR SS:[EBP+399001]
00F416C3	FFB5 05903900	PUSH DWORD PTR SS:[EBP+399005]
00F416C9	FFB5 09903900	PUSH DWORD PTR SS:[EBP+399009]
00F416CF	FFB5 F98F3900	PUSH DWORD PTR SS:[EBP+398FF9]
00F416D5	FFB5 FD8F3900	PUSH DWORD PTR SS:[EBP+398FFD]
00F416DB	E8 08000000	CALL 00F416E8
00F416E0	894424 1C	MOV DWORD PTR SS:[ESP+1C],EAX
00F416E4	61	POPAD
00F416E5	- FFE0	INC EAX
00F416E7	C3	RETN
00F416E8	55	PUSH EBP
00F416E9	8BEC	MOV EBP,ESP
00F416EB	83C4 FC	ADD ESP,-4
00F416EE	2B65 14	SUB ESP,DWORD PTR SS:[EBP+14]
00F416F1	8965 FC	MOV DWORD PTR SS:[EBP+4],ESP
00F416F4	8B4D 14	MOV ECX,DWORD PTR SS:[EBP+14]

This section arising randomly in memory, each machine can do the same offset

Trace from F7 to immediately order SUB:

Address	Hex dump	Disassembly
00F416B0	60	PUSHAD
00F416B1	E8 00000000	CALL 00F416B6
00F416B6	5D	POP EBP
00F416B7	81ED 8F8F3900	SUB EBP,398F8F
00F416B8	FFB5 01903900	PUSH DWORD PTR SS:[EBP+399001]
00F416C3	FFB5 05903900	PUSH DWORD PTR SS:[EBP+399005]
00F416C9	FFB5 09903900	PUSH DWORD PTR SS:[EBP+399009]
00F416CF	FFB5 F98F3900	PUSH DWORD PTR SS:[EBP+398FF9]

Dom through register, see the value in EBP:


```

Registers (FPU)
EAX 00396BA0
ECX 0012FFB0
EDX 7C90EB94 ntdll.KiFastSystemC
EBX 7FFD9000
EBP 00F416B6
ESI 00000000
EDI 7C910738 ntdll.7C910738
EIP 00F416B7

C 1 ES 0023 32bit 0(FFFFFFFF)
P 0 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
O 0
0 0 LastErr ERROR_INVALID_HANDLE

```

After analysis, try try again, get that trick see value here, except to the 6 to 1 in value encrypt IAT: F416B6 - 6 = F416B0

Address	Value	Comment
0042A234	00F4141C	
0042A238	00F414A0	
0042A23C	00F41524	
0042A240	00F415A8	
0042A244	00F416B0	
0042A248	00F417B8	
0042A250	00F4183C	
0042A254	00F418C0	
0042A258	00F41944	
0042A25C	00F419C8	
0042A260	00F41A4C	
0042A264	00F41AD0	
0042A268	00F41B54	
0042A26C	00F41BD8	
0042A270	00F41C5C	
0042A274	00F41CE0	
0042A278	00F41D64	
0042A27C	00F41DE8	

And we also observed kī lines under the code:

Address	Hex dump	Disassembly
00F416B6	5D	POP EBP
00F416B7	81ED 8F8F3900	SUB EBP,398F8F
00F416B8	FFB5 01903900	PUSH DWORD PTR SS:[EBP+399001]
00F416C3	FFB5 05903900	PUSH DWORD PTR SS:[EBP+399005]
00F416C9	FFB5 09903900	PUSH DWORD PTR SS:[EBP+399009]
00F416CF	FFB5 F98F3900	PUSH DWORD PTR SS:[EBP+398FF9]
00F416D5	FFB5 FD8F3900	PUSH DWORD PTR SS:[EBP+398FFD]
00F416DB	E8 00000000	CALL 00F416E8
00F416E0	894424 1C	MOV DWORD PTR SS:[ESP+1C],EAX
00F416E4	61	POPAD
00F416E5	- FFE0	INC EAX

British children get the value at which nhé. Now continue to Trace CALL immediately below, we go F7 invisible:

Address	Hex dump	Disassembly
00F416E7	C3	RET
00F416E8	55	PUSH EBP
00F416E9	8BEC	MOV EBP,ESP
00F416EB	83C4 FC	ADD ESP,-4
00F416EE	2B65 14	SUB ESP,DWORD PTR SS:[EBP+14]
00F416F1	8965 FC	MOV DWORD PTR SS:[EBP-4],ESP
00F416F4	8B4D 14	MOV ECX,DWORD PTR SS:[EBP+14]

They go down immediately under the code, trace back to here:

00F416E8	55	PUSH EBP
00F416E9	8BEC	MOV EBP,ESP
00F416EB	83C4 FC	ADD ESP,-4
00F416EE	2B65 14	SUB ESP,DWORD PTR SS:[EBP+14]
00F416F1	8965 FC	MOV DWORD PTR SS:[EBP-4],ESP
00F416F4	8B4D 14	MOV ECX,DWORD PTR SS:[EBP+14]
00F416F7	8B75 10	MOV ESI,DWORD PTR SS:[EBP+10]
00F416FA	8B7D FC	MOV EDI,DWORD PTR SS:[EBP-4]
00F416FD	F3:A4	REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00F416FF	FF75 FC	PUSH DWORD PTR SS:[EBP-4]
00F41702	FF55 18	CALL DWORD PTR SS:[EBP+18]
00F41705	FF75 FC	PUSH DWORD PTR SS:[EBP-4]
00F41708	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
00F4170B	FF55 08	CALL DWORD PTR SS:[EBP+8]
00F4170E	50	PUSH EAX
00F4170F	8B7D FC	MOV EDI,DWORD PTR SS:[EBP-4]
00F41712	8B4D 14	MOV ECX,DWORD PTR SS:[EBP+14]
00F41715	32C0	XOR AL,AL
00F41717	F3:AA	REP STOSB BYTE PTR ES:[EDI]
00F41719	58	POP EAX

Meet REP must wa F8 to trace the ceremony, and then again several F8 wa CALL, stop at:

Address	Hex dump	Disassembly
00F416FA	8B7D FC	MOV EDI,DWORD PTR SS:[EBP-4]
00F416FD	F3:A4	REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00F416FF	FF75 FC	PUSH DWORD PTR SS:[EBP-4]
00F41702	FF55 18	CALL DWORD PTR SS:[EBP+18]
00F41705	FF75 FC	PUSH DWORD PTR SS:[EBP-4]
00F41708	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
00F4170B	FF55 08	CALL DWORD PTR SS:[EBP+8]
00F4170E	50	PUSH EAX
00F4170F	8B7D FC	MOV EDI,DWORD PTR SS:[EBP-4]
00F41712	8B4D 14	MOV ECX,DWORD PTR SS:[EBP+14]
00F41715	32C0	XOR AL,AL
00F41717	F3:AA	REP STOSB BYTE PTR ES:[EDI]
00F41719	58	POP EAX

Dom through register (or content at the line):

Registers (EIP)	
EAX	7C8111DA kernel32.GetVersion
EDX	7C97C008 ntdll.7C97C008
EBX	7FFD8000
ESP	0012FEF8 ASCII "GetVersion"
EBP	0012FF08
ESI	00F502BC
EDI	0012FF04
EIP	00F4170E
C 0	ES 0023 32bit 0(FFFFFFFF)
P 0	CS 001B 32bit 0(FFFFFFFF)
A 0	SS 0023 32bit 0(FFFFFFFF)
Z 0	DS 0023 32bit 0(FFFFFFFF)
S 0	FS 003B 32bit 7FFDF000(FFF)
T 0	GS 0000 NULL
O 0	
O 0	LastErr ERROR_INVALID_HANDLE
EFL	00000202 (NO,NB,NE,A,NS,PO,GE

Hehe, this is already EAX 1 contains real value in IAT (Ham GetVersion often called the first of OEP's program code in VC). Offset contain it before the program is offset pack contains encrypt value that we found above. Okie, so ask questions, this is the only decrypt IAT value for this? To test our back seats near OEP CALL:

Address	Hex dump	Disassembly
004017C1	56	PUSH ESI
004017C2	57	PUSH EDI
004017C3	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP
004017C6	FF15 48A24200	CALL DWORD PTR DS:[42A248]
004017CC	A3 607D4200	MOV DWORD PTR DS:[427D60],EAX
004017D1	A1 607D4200	MOV EAX,DWORD PTR DS:[427D60]
004017D6	C1E8 08	SHR EAX,8
004017D9	25 FF000000	AND EAX,0FF
004017DE	A3 6C7D4200	MOV DWORD PTR DS:[427D6C],EAX
004017E3	8B0D 607D4200	MOV ECX,DWORD PTR DS:[427D60]
004017E9	01E1 F0000000	AND ECX,0FF

BP set just below this CALL, F9 to exit from the other end to decrypt l . Now, right next to trace CALL immediately below:

Address	Hex dump	Disassembly
004017FE	0315 6C7D4200	ADD EDX,DWORD PTR DS:[427D6C]
00401804	8915 647D4200	MOV DWORD PTR DS:[427D64],EDX
0040180A	A1 607D4200	MOV EAX,DWORD PTR DS:[427D60]
0040180F	C1E8 10	SHR EAX,10
00401812	25 FFFF0000	AND EAX,0FFFF
00401817	A3 607D4200	MOV DWORD PTR DS:[427D60],EAX
0040181C	6A 00	PUSH 0
0040181E	E8 001C0000	CALL UnpackMe.00403430
00401823	83C4 04	ADD ESP,4
00401826	85C0	TEST EAX,EAX
00401828	75 0A	JNZ SHORT UnpackMe.00401834
0040182A	6A 1C	PUSH 1C
0040182C	E8 FF000000	CALL UnpackMe.00401930
00401831	A3C4 04	ADD ESP,4

F7 go invisible:

Address	Hex dump	Disassembly
00403430	55	PUSH EBP
00403431	8BEC	MOV EBP,ESP
00403433	6A 00	PUSH 0
00403435	68 00100000	PUSH 1000
0040343A	33C0	XOR EAX,EAX
0040343C	837D 08 00	CMP DWORD PTR SS:[EBP+8],0
00403440	0F94C0	SETL AL
00403443	50	PUSH EAX
00403444	FF15 A4A24200	CALL DWORD PTR DS:[42A2A4]
0040344F	833D 54954200 00	CMP DWORD PTR DS:[429554],0
00403456	75 04	JNZ SHORT UnpackMe.0040345C

1 CALL IAT again just below, then trace back F7 to go ...:

Address	Hex dump	Disassembly
00F4228C	60	PUSHAD
00F4228D	E8 00000000	CALL 00F42292
00F42292	5D	POP EBP
00F42293	81ED 8F8F3900	SUB EBP,398F8F
00F42299	FFB5 01903900	PUSH DWORD PTR SS:[EBP+399001]
00F4229F	FFB5 05903900	PUSH DWORD PTR SS:[EBP+399005]
00F422A5	FFB5 09903900	PUSH DWORD PTR SS:[EBP+399009]
00F422AB	FFB5 F98F3900	PUSH DWORD PTR SS:[EBP+398FF9]
00F422B1	FFB5 FD8F3900	PUSH DWORD PTR SS:[EBP+398FFD]
00F422B7	E8 08000000	CALL 00F422C4
00F422BC	894424 1C	MOV DWORD PTR SS:[ESP+1C],EAX
00F422C0	61	POPAD
00F422C1	FFE0	JMP EAX
00F422C3	C3	RET

Haha, still remember the value that trick to make you say, here is to see that the offset decrypt the other, but also the content on the change italy, similar to the value, and the CALL to code under the same again. So can form ideas decrypt the code 1 and đã.

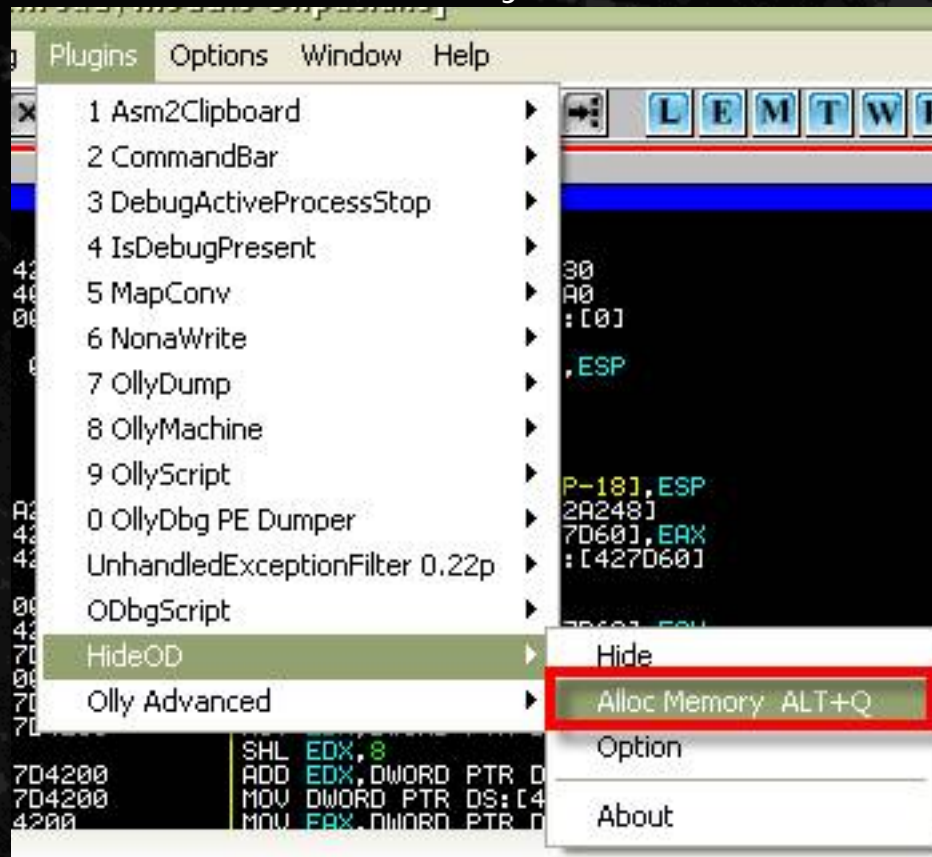
2. Code:

The trick for you to imagine before:

1. First, the value of EBP order to trace the SUB 1 is equal in value encrypt IAT + 6 more.
2. The value of the decrypt all the same should be able to use it to decrypt the entire IAT
3. Encrypt the value in increasing IAT (+84 h), offset increased IAT (+4 h). Enough information to create a loop.
4. Valid Ham will draw me in order PUSH EAX. So the code to the legs is sufficient.

Now we go into the code of this. Suppose we just break in and find

OEP. 1 Create a blank area in memory HideOD:



Memory area is created on each machine each other, each time is different. And now, on a trick F80000 (default size by creating a HideOD 3000h). Goto One to:

Address	Hex dump	Disassembly
00F80000	0000	ADD BYTE PTR DS:[EAX],AL
00F80002	0000	ADD BYTE PTR DS:[EAX],AL
00F80004	0000	ADD BYTE PTR DS:[EAX],AL
00F80006	0000	ADD BYTE PTR DS:[EAX],AL
00F80008	0000	ADD BYTE PTR DS:[EAX],AL
00F8000A	0000	ADD BYTE PTR DS:[EAX],AL
00F8000C	0000	ADD BYTE PTR DS:[EAX],AL
00F8000E	0000	ADD BYTE PTR DS:[EAX],AL
00F80010	0000	ADD BYTE PTR DS:[EAX],AL
00F80012	0000	ADD BYTE PTR DS:[EAX],AL
00F80014	0000	ADD BYTE PTR DS:[EAX],AL
00F80016	0000	ADD BYTE PTR DS:[EAX],AL
00F80018	0000	ADD BYTE PTR DS:[EAX],AL
00F8001A	0000	ADD BYTE PTR DS:[EAX],AL
00F8001C	0000	ADD BYTE PTR DS:[EAX],AL
00F8001E	0000	ADD BYTE PTR DS:[EAX],AL

The first is to record the status of several key wan to write:

00F80000 PUSHAD

Next is to save the status of existing stack, as in the decrypt, Stack been changed quite nhiều (ko rules are clear):

00F80001 MOV DWORD PTR DS:[F81000], ESP

ESP keep the stack pointer to save it is saved with the status of stack. Cove HideOD generated 3000h size should F81000 also in that, do touch someone, we use it instead for 1 Byte 4 variables to store value of ESP.

Now move offset by IAT Start of EBX, began creating loop:

```
00F80007 MOV EBX, 42A1E8
```

42A1E8 the IAT đã Start. Need he considered them self again. The value is important here, because we need it for the EBP:

```
00F8000C MOV EBP, DWORD PTR DS: [EBX]
```

```
00F8000E ADD EBP, 6
```

The more like a 6, and start from the legs decrypt code:

```
00F80011 SUB EBP, 398F8F
```

```
00F80017 PUSH DWORD PTR SS: [EBP + 399001]
```

```
00F8001D PUSH DWORD PTR SS: [EBP + 399005]
```

```
00F80023 PUSH DWORD PTR SS: [EBP + 399009]
```

```
00F80029 PUSH DWORD PTR SS: [EBP + 398 FF9]
```

```
00F8002F PUSH DWORD PTR SS: [EBP + 398 FFD]
```

```
00F80035 CALL 00F8003A
```

```
00F8003A PUSH EBP
```

```
00F8003B MOV EBP, ESP
```

```
00F8003D ADD ESP, -4
```

```
00F80040 SUB ESP, DWORD PTR SS: [EBP + 14]
```

```
00F80043 MOV DWORD PTR SS: [EBP-4], ESP
```

```
00F80046 MOV ECX, DWORD PTR SS: [EBP + 14]
```

```
00F80049 MOV ESI, DWORD PTR SS: [EBP + 10]
```

```
00F8004C MOV EDI, DWORD PTR SS: [EBP-4]
```

```
00F8004F REP MOVSB BYTE PTR ES: [EDI], BYTE PTR DS: [ESI]
```

```
00F80051 PUSH DWORD PTR SS: [EBP-4]
```

```
00F80054 CALL DWORD PTR SS: [EBP + 18]
```

```
00F80057 PUSH DWORD PTR SS: [EBP-4]
```

```
00F8005A PUSH DWORD PTR SS: [EBP + C]
```

```
00F8005D CALL DWORD PTR SS: [EBP + 8]
```

Place **00F80035 CALL 00F8003A call** the code immediately below it, because the copy is made for the same decrypt, it must not modify, balanced stack very tired: D

Complete a call CALL bottom is the original value of 1 IAT transfer of EAX we should not need to copy the code below khúc. Now the conditions to create loop, repeating the code for all IAT.

```
00F80060 MOV DWORD PTR DS: [EBX], EAX
```

```
00F80062 ADD EBX, 4
```

```
Cmp 00F80065 EBX, 42A358
```

```
00F8006B JE SHORT 00F80085
```

```
Cmp 00F8006D DWORD PTR DS: [EBX], 0F40000
```

```
00F80073 JL SHORT 00F80062
```

```
00F80075 Cmp DWORD PTR DS: [EBX], 0F43000
```



```

JG 00F8007B SHORT 00F80062
00F8007D MOV ESP, DWORD PTR DS: [F81000]
00F80083 JMP SHORT 00F8000C
00F80085 MOV ESP, DWORD PTR DS: [F81000]
00F8008B POPAD

```

We saved by Import Value to correct location of its origin (EBX now still hold offset in IAT). Then gradually increase to the EBX. Compare see used IAT End or not? If this is the end to decrypt IAT, if not consider the content viewed here are valid or not, not necessarily because of any increase is offset to the right place to save the IAT. Dom again:

Address	Value	Comment
0042A2A4	00F4228C	
0042A2A8	00F42310	
0042A2AC	00F42394	
0042A2B0	00F42418	
0042A2B4	00000000	
0042A2B8	00000000	
0042A2BC	00000000	
0042A2C0	00000000	
0042A2C4	00000000	
0042A2C8	00000000	
0042A2CC	00000000	
0042A2D0	00000000	
0042A2D4	00000000	
0042A2D8	00000000	
0042A2DC	00000000	
0042A2E0	00000000	
0042A2E4	00000000	
0042A2E8	00000000	
0042A2EC	00000000	
0042A2F0	00000000	
0042A2F4	00000000	
0042A2F8	00000000	
0042A2FC	00000000	
0042A300	00000000	
0042A304	00000000	
0042A308	00F40000	
0042A30C	00F40084	
0042A310	00F40108	
0042A314	00F4018C	
0042A318	00F40210	
0042A31C	00F40294	
0042A320	00F40318	

Region IAT ko continuously between can be valuable refuse or null. So how many consider valid. The first code of the trick is b m conditions encrypt IAT + 84h increased, but that is offset IAT to increase the rate of value to encrypt IAT. Dom's still on, we see the offset in the IAT has value smaller than offset the above.

Eg offset 42A308 > offset 42A2B0. But value is F40000 < F42418.

Should be used if conditions +84 h increases, you must run the code several times with each region IAT. Here trick improvement, considering the value of the other

```

Cmp 00F8006D DWORD PTR DS: [EBX], 0F40000
00F80073 JL SHORT 00F80062
00F80075 Cmp DWORD PTR DS: [EBX], 0F43000
JG 00F8007B SHORT 00F80062

```


Why are 2 of the gold bowl? View area remember Alt-M:

30400000	00001000	UnpackMe		PE header	Imag	R
30401000	0006D000	UnpackMe	.text	code, data	Imag	R
3046E000	00001000	UnpackMe	.rsrc	resources	Imag	R
30470000	00011000				Map	R
30530000	00002000				Map	R
30540000	00103000				Map	R
30650000	001CD000				Map	R
30950000	00001000				Priv	R
30960000	00001000				Priv	R
30970000	00007000				Priv	R
30980000	00001000				Map	R
30990000	00010000				Map	R
309D0000	0000E000				Map	R
309E0000	00001000				Priv	R
309F0000	00010000				Map	R
30A30000	00001000				Priv	R
30AB0000	00006000				Map	R
30FB0000	00010000				Map	R
0F30000	00001000				Map	R
0F40000	00003000				Priv	R
0F50000	00017000				Priv	R
30F70000	00001000				Priv	R
30F80000	00003000				Priv	R
3FFD0000	00001000	rsaenh		PE header	Imag	R

This is a small region decrypt the IAT's packer. Should encrypt the contents of the IAT also located in this area: D. So 2 values on the offset and the end of it, hehe. There are many ways and is only 1 month only. If the value in IAT lot from this region is one that is touching garbage or null (the value problems of this region is quite low 0i), and any further increase offset IAT up until valid, loop started again. When completed, the return POPAD content than the original record. This is the result of the following code:

Address	Hex dump	Disassembly
00F80000	60	PUSHAD
00F80001	8925 0010F800	MOV DWORD PTR DS:[F81000],ESP
00F80007	BB E8A14200	MOV EBX,42A1E8
00F8000C	8B2B	MOV EBP,DWORD PTR DS:[EBX]
00F8000E	83C5 06	ADD EBP,6
00F80011	81ED 8F8F3900	SUB EBP,398F8F
00F80017	FFB5 01903900	PUSH DWORD PTR SS:[EBP+399001]
00F8001D	FFB5 05903900	PUSH DWORD PTR SS:[EBP+399005]
00F80023	FFB5 09903900	PUSH DWORD PTR SS:[EBP+399009]
00F80029	FFB5 F98F3900	PUSH DWORD PTR SS:[EBP+398FF9]
00F8002F	FFB5 FD8F3900	PUSH DWORD PTR SS:[EBP+398FFD]
00F80035	E8 00000000	CALL 00F8003A
00F8003A	55	PUSH EBP
00F8003B	8BEC	MOV EBP,ESP
00F8003D	83C4 FC	ADD ESP,-4
00F80040	2B65 14	SUB ESP,DWORD PTR SS:[EBP+14]
00F80043	8965 FC	MOV DWORD PTR SS:[EBP-4],ESP
00F80046	8B4D 14	MOV ECX,DWORD PTR SS:[EBP+14]
00F80049	8B75 10	MOV ESI,DWORD PTR SS:[EBP+10]
00F8004C	8B7D FC	MOV EDI,DWORD PTR SS:[EBP-4]
00F8004F	F3A4	REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00F80051	FF75 FC	PUSH DWORD PTR SS:[EBP-4]
00F80054	FF55 18	CALL DWORD PTR SS:[EBP+18]
00F80057	FF75 FC	PUSH DWORD PTR SS:[EBP-4]
00F8005A	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
00F8005D	FF55 08	CALL DWORD PTR SS:[EBP+8]
00F80060	8903	MOV DWORD PTR DS:[EBX],EAX
00F80062	83C3 04	ADD EBX,4
00F80065	81FB 58A34200	CMP EBX,42A358
00F8006B	74 18	JE SHORT 00F80085
00F8006D	813B 0000F400	CMP DWORD PTR DS:[EBX],0F40000
00F80073	7C ED	JL SHORT 00F80062
00F80075	813B 0000F400	CMP DWORD PTR DS:[EBX],0F43000
00F8007B	7F E5	JG SHORT 00F80062
00F8007D	8B25 0010F800	MOV ESP,DWORD PTR DS:[F81000]
00F80083	EB 87	JMP SHORT 00F8000C
00F80085	8B25 0010F800	MOV ESP,DWORD PTR DS:[F81000]
00F8008B	61	POPAD

Remember that being in the OEP, we Goto come to code. Now set in New Origin first code, and set in order BP NOP after POPAD. F9 to run this code. Break in the filed, the IAT results decrypt:

Address	Value	Comment
0042A1DC	00000000	
0042A1E0	00000000	
0042A1E4	00000000	
0042A1E8	7C80DDF5	kernel32.GetCurrentProcess
0042A1EC	7C809B47	kernel32.CloseHandle
0042A1F0	7C812641	kernel32.FlushFileBuffers
0042A1F4	7C81D2CB	kernel32.SetStdHandle
0042A1F8	7C810B9E	kernel32.SetFilePointer
0042A1FC	7C80A490	kernel32.GetStringTypeW
0042A200	7C838974	kernel32.GetStringTypeA
0042A204	7C80CCA8	kernel32.LCMapStringW
0042A208	7C838D50	kernel32.LCMapStringA
0042A20C	7C809BF8	kernel32.MultiByteToWideChar
0042A210	7C9179FD	ntdll.RtlReAllocateHeap
0042A214	7C809A51	kernel32.VirtualAlloc
0042A218	7C9105D4	ntdll.RtlAllocateHeap
0042A21C	7C8127A7	kernel32.GetOEMCP
0042A220	7C809915	kernel32.GetACP
0042A224	7C812E76	kernel32.GetCPInfo
0042A228	7C85EB9B	kernel32.HeapValidate
0042A22C	7C809E01	kernel32.IsBadReadPtr
0042A230	7C809E79	kernel32.IsBadWritePtr

Region 1

Address	Value	Comment
0042A2AC	7C809AE4	kernel32.VirtualFree
0042A2B0	7C937A40	ntdll.RtlUnwind
0042A2B4	00000000	
0042A2B8	00000000	
0042A2BC	00000000	
0042A2C0	00000000	
0042A2C4	00000000	
0042A2C8	00000000	
0042A2CC	00000000	
0042A2D0	00000000	
0042A2D4	00000000	
0042A2D8	00000000	
0042A2DC	00000000	
0042A2E0	00000000	
0042A2E4	00000000	
0042A2E8	00000000	
0042A2EC	00000000	
0042A2F0	00000000	
0042A2F4	00000000	
0042A2F8	00000000	
0042A2FC	00000000	
0042A300	00000000	
0042A304	00000000	
0042A308	7E4263F9	USER32.EndDialog
0042A30C	7E43B12C	USER32.DialogBoxParamA
0042A310	7E41DAD2	USER32.DestroyWindow
0042A314	7E41D4D6	USER32.DefWindowProcA
0042A318	7E41B5F9	USER32.BeginPaint
0042A31C	7E41B69E	USER32.GetClientRect
0042A320	7E43C6EA	USER32.DrawTextA
0042A324	7F41B600	USER32.EndPaint

Region 2

Them. What to do next is anyone know, write more redundant and the trick has đũa ... hehe

V. Ending:

The code on the trick has improved while writing this article, because the old code to the long ời, time found errors, hehe. But this type of improvement is quite cui corn, who are in the high security would only help. Let the items

that also more than 4 hours, including time again failed the test, not important in the training run training, fix dump, which is we understand each line of code. Thank he was in the recall tí inspiration for the children this. Now back to trick Đồng project with vẫn's kì ago, hix hix, nhằm is the writing, do report back on, pa kOn chào nhé.

Big thanks to:

All REA's members: Computer_Angel, Moonbaby, Zombie, hacnho, benina, kienmanowar, rongchaua, Deux, Merc, hoadongnoi, the_lighthouse, TQN, light.phoenix, hytkl, tlandn, hurt_heart, dzungltn, Zoi, littleboy84, haule_nth, takada, Why not bar, iamidiot, Akira, dump, thienthandien, [kid], ...

Special thanks to:

fly, stephenteh, Gabri3l, MaDMAn_H3rCuL3s, CondZero, Ricardo Narvaja + NCR, lena151, haggard, ARTeam, snd, RES, CrackLatinos, all unpack. cn ... Authors who created tools and you.



Trick Xi News - 2007

Manual Removing Visual Protect 3.5.4



I -- Print d o t r u c t i o n :

Hi everyone, today our brothers together **Manual Removing Visual Protect Version 3.5.4**. Protect this type now has a lot to use new protection for their Soft. And of course want to be Soft Crack is using **Visual Protect Protect** the first you must remove the protection of this ... we started ...

I I -- Tools & T a g e r t :

• T o o L and P l u g i n c a n d ử n g :

- O L L Y D B G 1.10
- L o r d P E 1.4
- I m p O R T R E C 1.6f
- R D G P A C K D e r e c t e r t o v0.6.4

• O n e g e r t : *ual Vi s e P r o t c t 3.5.4*

[h t t p : / / a g e s w w w . v i s o f t . c o m](http://ageswww.visoft.com)

I I I - F i n d O E P & D u m p F I L E :

_As usual use **RDG Packer Detector v0.6.4** scan target



_OK, Load **OillyDBG** on target and we stop here:

006BCF90	55	PUSH EBP	
006BCF91	8BEC	MOV EBP,ESP	
006BCF93	51	PUSH ECX	
006BCF94	53	PUSH EBX	
006BCF95	56	PUSH ESI	
006BCF96	57	PUSH EDI	
006BCF97	C705 003E6C00	MOV DWORD PTR DS:[6C3E90],0	
006BCFA1	68 48206C00	PUSH VisualPr.006C2048	ASCII "kernel32.dll"
006BCFA6	FF15 00006C00	CALL NEAR DWORD PTR DS:[&KERNEL32.LoadLibraryA	kernel32.LoadLibraryA
006BCFAC	A3 0C3F6C00	MOV DWORD PTR DS:[6C3F0C],EAX	
006BCFB1	68 58206C00	PUSH VisualPr.006C2058	ASCII "GetModuleHandleA"
006BCFB6	A1 0C3F6C00	MOV EAX,DWORD PTR DS:[6C3F0C]	
006BCFBB	50	PUSH EAX	
006BCFBC	FF15 04006C00	CALL NEAR DWORD PTR DS:[&KERNEL32.GetProcAddress	kernel32.GetProcAddress
006BCFC2	A3 903E6C00	MOV DWORD PTR DS:[6C3E90],EAX	

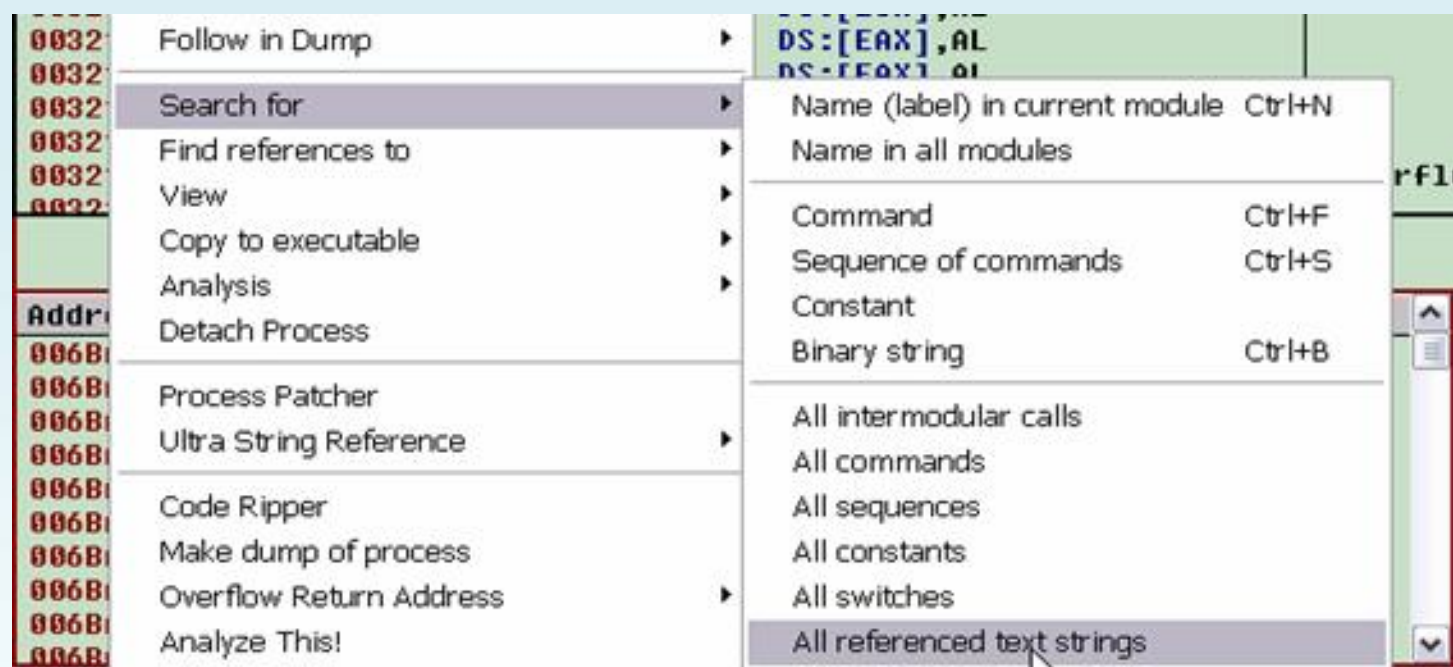
_Press **Alt + E**

Entry	Name	File version	Path
003E2944	UP	3.5.3.0	C:\Program Files\Visage\Visual Protect\VP.DLL
006BCF90	VisualPr	3.5.4.0	C:\Program Files\Visage\Visual Protect\VisualProtect.exe
00F9570B	HSTxtCap	1, 0, 2, 0	C:\Program Files\HyperSnap 6\HSTxtCap.dll
1000721F	dxsnap		C:\Program Files\HyperSnap 6\dxsnap.dll
5AD71626	uxtheme	6.00.2900.2180	C:\WINDOWS\system32\uxtheme.dll
5B0A10B0	undnxfm	5.1.2600.0 (xpc	C:\WINDOWS\system32\undnxfm.dll
5CD714D3	servwdrv	5.1.2600.0 (xpc	C:\WINDOWS\system32\servwdrv.dll
5D0932DA	COMCTL32	5.82 (xpsp_sp2	C:\WINDOWS\system32\COMCTL32.DLL
5EDDEE78	OLEPRO32	5.1.2600.2180	C:\WINDOWS\system32\OLEPRO32.DLL
72000000	WINSP001	5.1.2600.2180	C:\WINDOWS\system32\WINSP001.DLL

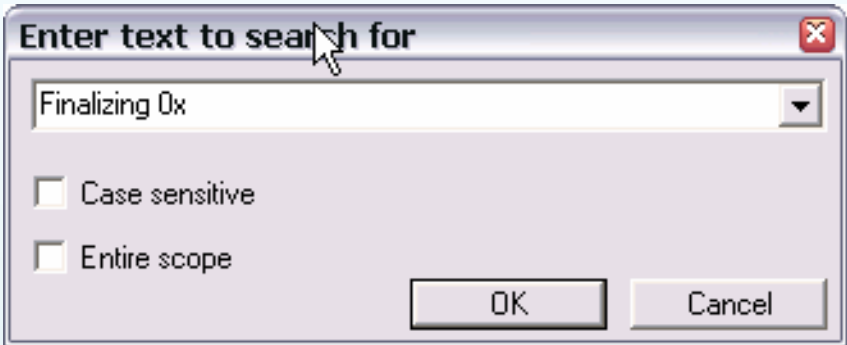
_ Double click **VP.DLL**

00321000	04 10	ADD AL,10	
00321002	3200	XOR AL,BYTE PTR DS:[EAX]	
00321004	0307	ADD EAX,DWORD PTR DS:[EDI]	
00321006	42	INC EDX	
00321007	6F	OUTS DX,DWORD PTR ES:[EDI]	I/O command
00321008	6F	OUTS DX,DWORD PTR ES:[EDI]	I/O command
00321009	6C	INS BYTE PTR ES:[EDI],DX	I/O command
0032100A	65:61	POPAD	Superfluous prefix
0032100C	6E	OUTS DX,BYTE PTR ES:[EDI]	I/O command
0032100D	0100	ADD DWORD PTR DS:[EAX],EAX	
0032100F	0000	ADD BYTE PTR DS:[EAX],AL	
00321011	0001	ADD BYTE PTR DS:[ECX],AL	
00321013	0000	ADD BYTE PTR DS:[EAX],AL	

_ Press **F10** and **Search String**



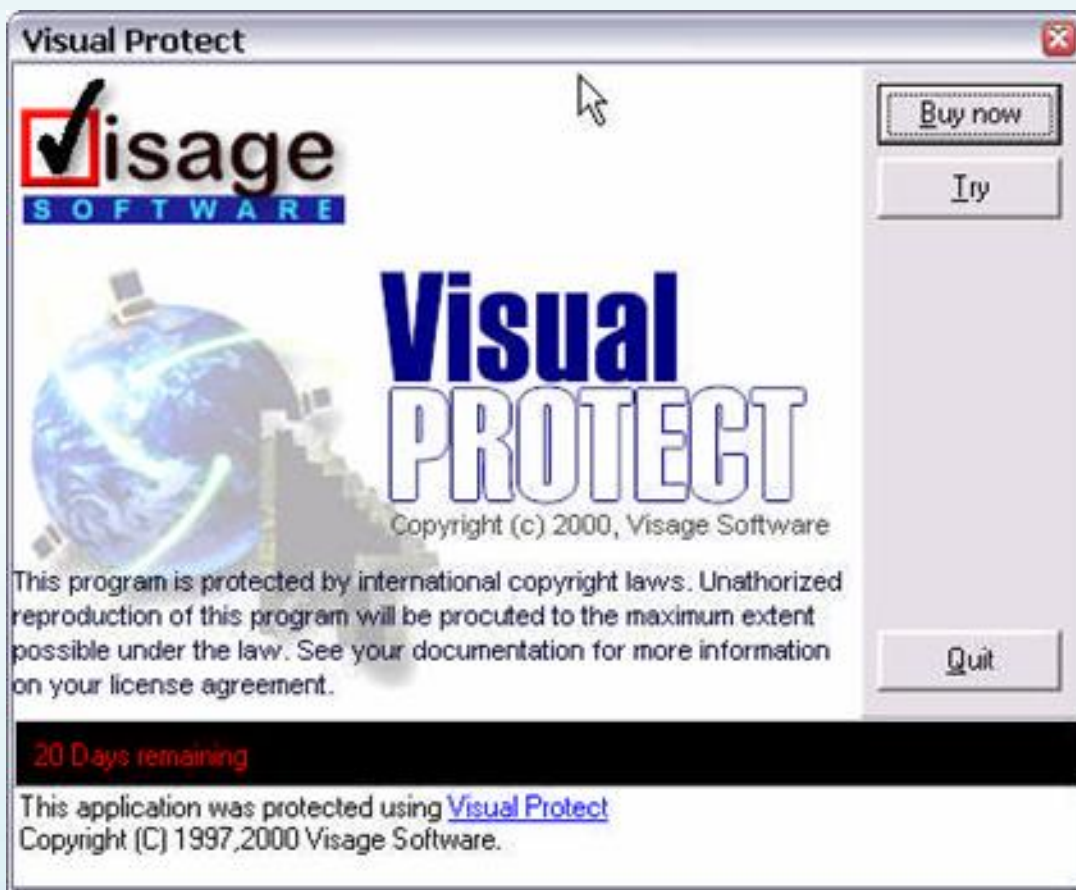
_Nhap To "**Finalizing 0x**" (you complete framework for the Arab world and the Soft Protect this form)



_ Click OK and press **F2** to **003C5FDD** Set 1 BP

003C5D75	MOV EAX,UP.003C6A8C	ASCII "SetEnvironment"
003C5E0F	MOV EDX,UP.003C6AA8	ASCII "ExitProcess UPCRCMatch=False CrashMeDate<>0 "
003C5E42	MOV EAX,UP.003C6AE0	ASCII "Uncompress Sections"
003C5E6E	MOV EAX,UP.003C6AFC	ASCII "Rellocation"
003C5EA6	MOV EAX,UP.003C6B10	ASCII "VC ImportTable Win NT 4.0/Windows 9.x"
003C5EB8	MOV EAX,UP.003C6B40	ASCII "Delphi ImportTable"
003C5F18	MOV EDX,UP.003C6B5C	ASCII "CodeCRCMatch False "
003C5F9B	MOV EDX,UP.003C6B78	ASCII "CodeCRCMatch False"
003C5FDD	MOV EDX,UP.003C6B94	ASCII "Finalizing 0x"
003C6033	MOV EDX,UP.003C6B94	ASCII "Finalizing 0x"
003C6087	MOV EDX,UP.003C6C1C	ASCII "'#13'"
003C60E3	MOV EDX,UP.003C6C1C	ASCII "'#13'"
003C60F3	MOV EDX,UP.003C6C90	ASCII " (0)"
003C614F	MOV EDX,UP.003C6C1C	ASCII "'#13'"

_ Press **F9**, and Soft Run appear **NAG**



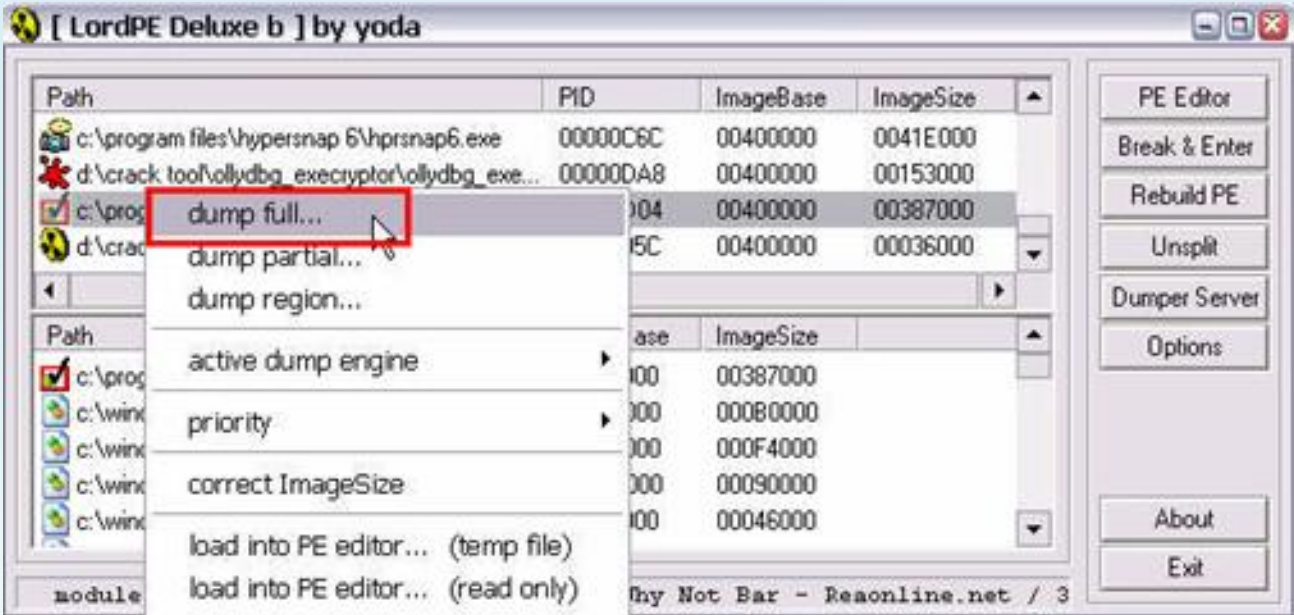
_ Click the **"Try"** and we will stop at just **Set BP**

003C5FC7	B8 08000000	MOV EAX,8	
003C5FCC	E8 AF2FF6FF	CALL UP.00328F80	
003C5FD1	8B8D 58FAFFFF	MOV ECX,DWORD PTR SS:[EBP-5A8]	
003C5FD7	8D85 5CFAFFFF	LEA EAX,DWORD PTR SS:[EBP-5A4]	
003C5FD0	BA 946B3C00	MOV EDX,UP.003C6B94	ASCII "Finalizing 0x"
003C5FE2	E8 B5E8F5FF	CALL UP.0032489C	
003C5FE7	8B85 5CFAFFFF	MOV EAX,DWORD PTR SS:[EBP-5A4]	
003C5FED	E8 46BCFFFF	CALL UP.003C1C38	
003C5FF2	FF65 FC	JMP NEAR DWORD PTR SS:[EBP-4]	
003C5FF5	6A 00	PUSH 0	
003C5FF7	E8 C00CF6FF	CALL UP.00326CBC	
003C5FFC	E9 C6040000	JMP UP.003C64C7	

_Trace **F8** to 5 times you will be to **OEP**

0066B508	55	PUSH EBP	==>OEP
0066B509	8BEC	MOV EBP,ESP	
0066B50B	83C4 F0	ADD ESP,-10	
0066B50E	B8 D0AC6600	MOV EAX,VisualPr.0066ACD0	
0066B513	E8 E8BCD9FF	CALL VisualPr.00407200	
0066B518	A1 34D86700	MOV EAX,DWORD PTR DS:[67D834]	
0066B51D	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0066B51F	E8 301EE2FF	CALL VisualPr.0048D354	
0066B524	A1 34D86700	MOV EAX,DWORD PTR DS:[67D834]	
0066B529	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0066B52B	BA B0B56600	MOV EDX,VisualPr.0066B5B0	ASCII "Visual Protect"
0066B530	E8 D318E2FF	CALL VisualPr.0048CE08	
0066B535	8B0D 50D56700	MOV ECX,DWORD PTR DS:[67D550]	VisualPr.0068DA78

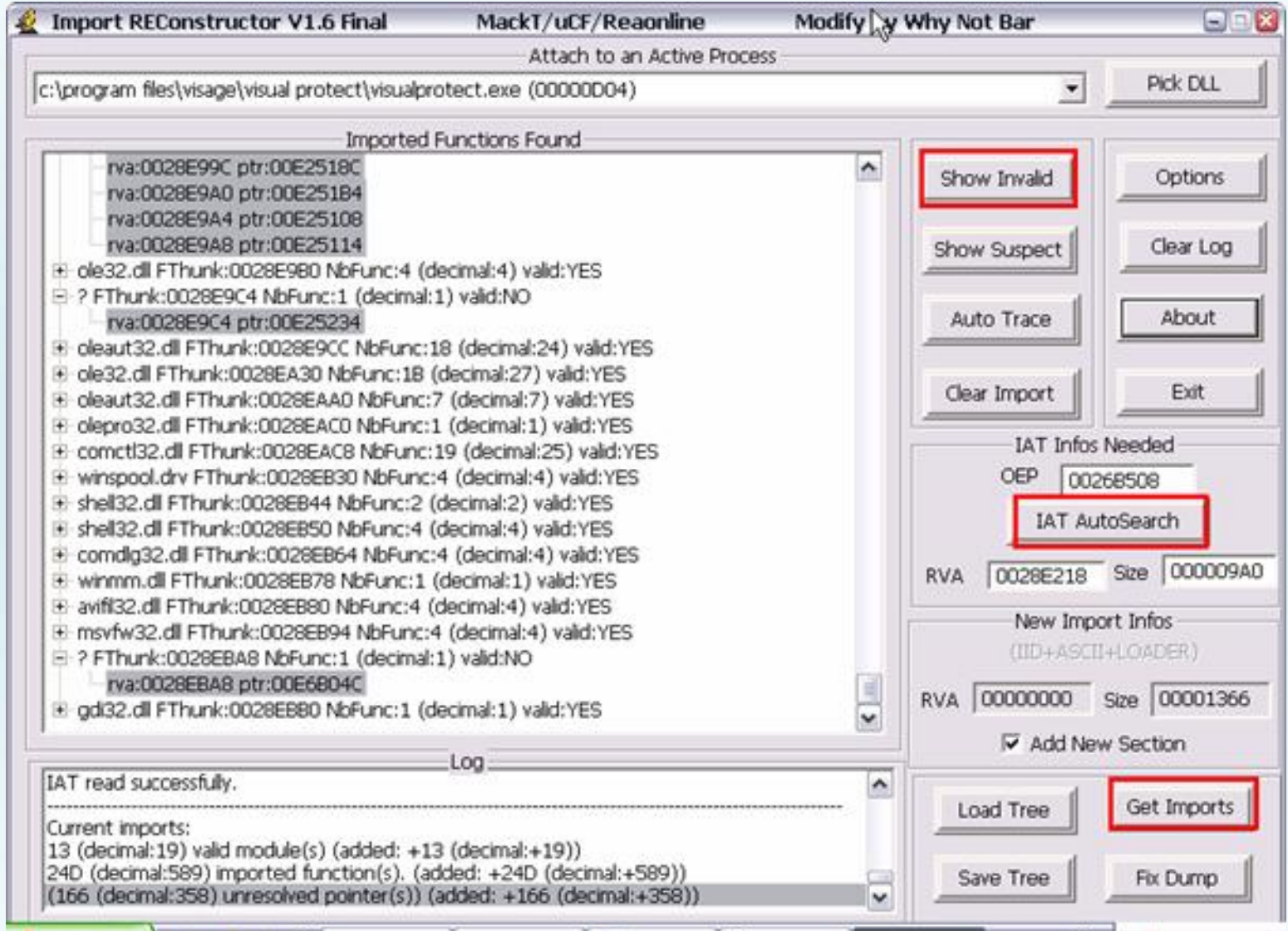
_M in **Lor dPE** and **Full dump**



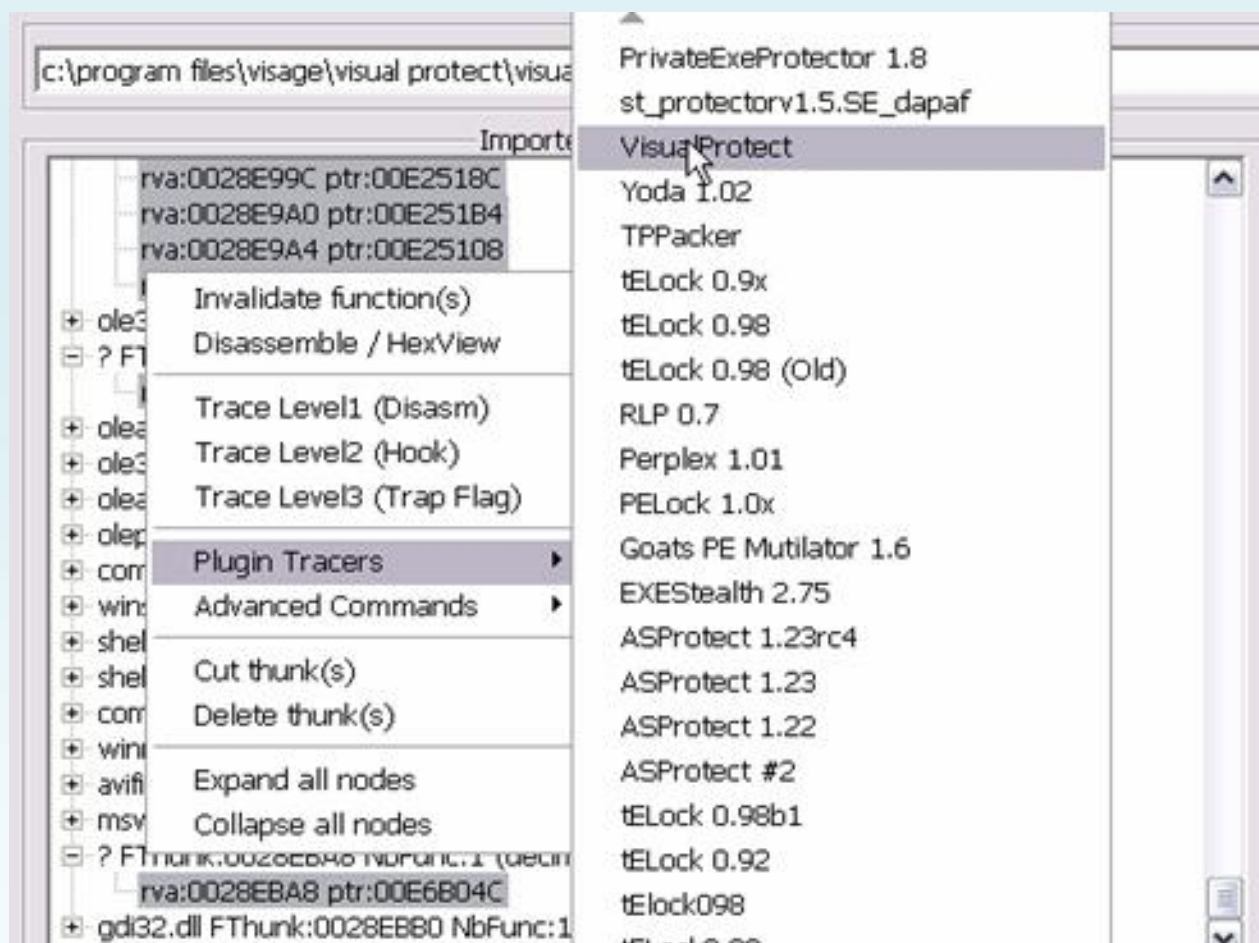
I-R E V I build mp ort:

_ Open **ImportREC** up. Select the list **VisualProtect.exe** process. **OEP** =

Enter **0066B508 - 00,400,000 (Imagebase) = 26B508, IAT AutoSearch**
 Click -> **Get Imports** -> **Show Invalid**



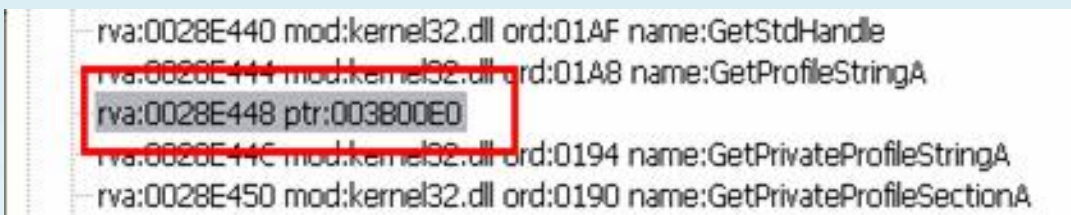
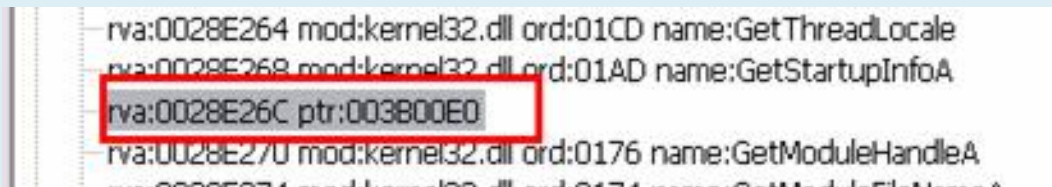
_Hichic A lot ... Fix **Invalid** function manually không always sure to use **Plugin** ... In the window select **ImportREC** nhusau:



_C c h t h ú í x u t a c ó



_ Click the **Show Invalid** 2 children Invalid



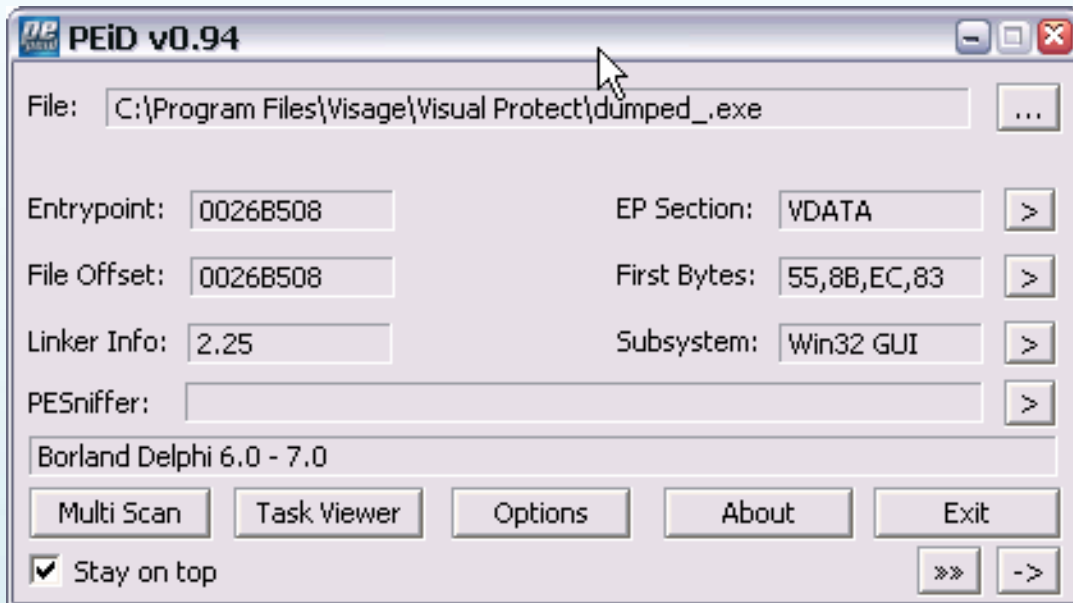
_ Fix manual thui ... Double-click on the first function **Invalid**



2 _ I also modified to "**GetProcAddress**" **Show Invalid** Click again and do not function any longer **Invalid** _ OK, Click **Fix dump** select File and Run **Dumped.exe** thuFile **Dumped.exe**



_Hehehe.... Unpack one D!!



G r I Ee TsF italy Ou the *Compu t e r A _ o f e I, e mbi Z o, M A B oo nb*

italy, H o acnh, Nina B e, k i e o w ar nman, Z o i, D e ux, M e r c, the

light o f nix, T r o ickyb italy, Takad a iamidi ot, of the e n t han e n d i,

o ... and italy u!

The N h a n a g, Day 2 9 t h a n g 8 years 200 6

W hot italyN Bar

MANUAL unpack ACTIVEMARK 5:31

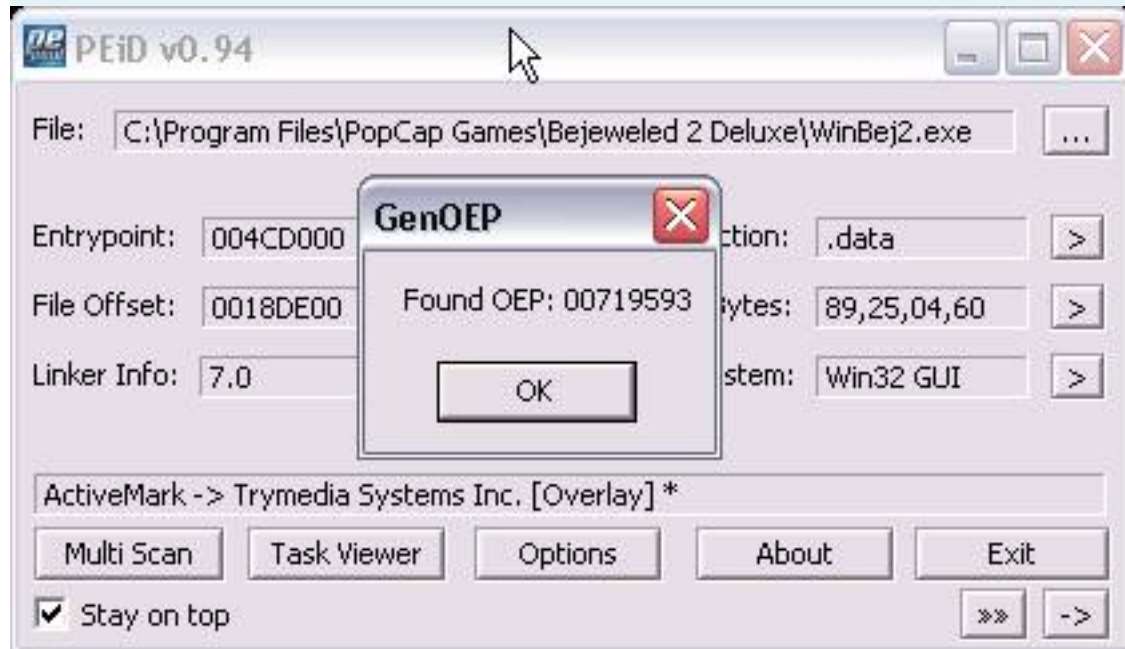
Target: Bejeweled 2 Deluxe (www.popcap.com)

Packer: ActiveMark 5:31

Tools: Ollydbg 1:10, Imprec 1.6, LordPe 1:41, PeiD 0.94, ActiveMark.Version

Bejeweled 2 Deluxe with the new version, the game's graphics and great game but 1 point lrm we do not get hri lnnng lr we buy it! Ha ha, where mr purchase money, treat it only!

_Scan File target with PeiD 0.94 and dung Plugin Finder Generic OEP we have:



_De Know the exact version ActiveMark you can use Tool "ActiveMark.Version"



_Nhu So you can guess the OEP in Section. "Text" (005D4000). All You should note the nry quite important. Run "WinBej2.exe" will have 1 Nag reminder register. Ok, so that open Olly Attach it and and you will here.

7C901231 C3 RETN


```

7C901232 8BFF MOV EDI, EDI
7C901234 90 NOP
7C901235 90 NOP

```

_bay time we jump to the first Address Section. In text **005D4000** we have as follows:

```

7C901231 - E9 CA2DCD83 JMP WinBej2.005D4000
7C901236 90 NOP
7C901237 90 NOP

```

_Nhan F7 to you here:

```

005D4000 6E OUTS DX, BYTE PTR ES: [EDI], I / O command
005D4001 6E OUTS DX, BYTE PTR ES: [EDI], I / O command
005D4002 16 PUSH SS
005D4003 0082 6E160094 ADD BYTE PTR DS: [EDX +9400166 E], AL

```

_nhap mouse to search for and select \ All Calls Intermodular and Go **"GetCommandLineA"** and double click view **"GetCommandLineA"** to you here:

```

00719614 E8 C05F0000 CALL WinBej2.0071F5D9
00719619 FF15 74D17300 CALL NEAR DWORD PTR DS: [73D174]; kernel32.
GetCommandLineA
0071961F A3 44967300 MOV DWORD PTR DS: [739644], EAX
00719624 E8 7E5E0000 CALL WinBej2.0071F4A7

```

_cuon mouse over the top and set the breakpoint 1 Hardware, on Execution in **00719593**:

```

00719593 55 PUSH EBP
00719594 8BEC MOV EBP, ESP
00719596 6A FF PUSH -1
00719598 68 C8EB6300 PUSH WinBej2.0063EBC8
0071959D 68 70F97100 PUSH WinBej2.0071F970

```

_nhap Alt + F2, F3 select "WinBej2.exe, Atl + O card in Events marked" **Break on new module (dll)** "and press F9 until at **00,719,593 (OEP)**

```
00719593 55 PUSH EBP <== OEP
```

```
00719594 8BEC MOV EBP, ESP
```

```
00719596 6A FF PUSH -1
```

```
00719598 68 C8EB6300 PUSH WinBej2.0063EBC8
```

```
0071959D 68 70F97100 PUSH WinBej2.0071F970
```

```
007195A2 64: A1 00000000 MOV EAX, DWORD PTR FS: [0]
```

Dung Plugin OllyDump dump the memory to select the items rebuild Import, import ImportREC open OEP = [319593](#), Click "IAT AutoSearch", "Get Imports", "Show Invalid", "Cut Thunks" and "Fix dump." Try and Run File not run! haha! Now Load File "Dumped.exe" vro Olly, but you must select "[Break on new module \(dll\)](#)" in the Events tab. We are here.

```
00719593> / $ 55 PUSH EBP
```

```
00719594 |. 8BEC MOV EBP, ESP
```

```
00719596 |. 6A FF PUSH -1
```

```
00719598 |. 68 C8EB6300 PUSH dumped_.0063EBC8
```

```
0071959D |. 68 70F97100 PUSH dumped_.0071F970; SE handler installation
```

_nhap mouse to search for and select \ All Calls Intermodular and Go "[GetModuleHandleA](#)" and double click vro dnng "[kernel32.GetModuleHandleA](#)"
First, and we come.

```
006407FA. 837D D8 00 Cmp DWORD PTR SS: [EBP-28], 0
```

```
006407FE. 74 49 JE SHORT dumped_.00640849
```

```
00640800. 813D B0706300> Cmp DWORD PTR DS: [6370B0], 416E6454
```

```
0064080A. 75 2F JNZ SHORT dumped_.0064083B
```

```
0064080C. 6A 00 PUSH 0; / pModule = null
```

```
0064080E. FF15 E8D17300 CALL NEAR DWORD PTR DS: [<& kernel32.  
GetModuleHandleA>] \ GetModuleHandleA
```

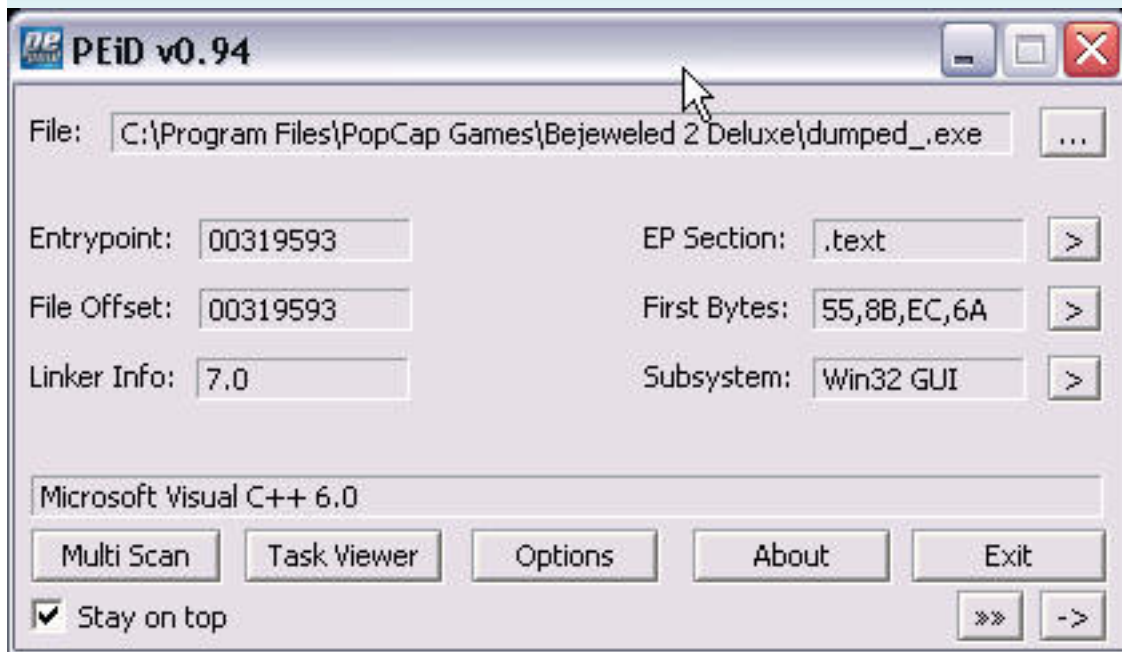
```
00640814. 8B0D BC706300 MOV ECX, DWORD PTR DS: [6370BC]
```

```
0064081A. 03C8 ADD ECX, EAX
```

```
0064081C. 890D E8485D00 MOV DWORD PTR DS: [5D48E8], ECX
```

_Dua Mouse pointer to [006407FE](#) and JE Patch thrnh JNZ. Then click to select Copy to executable \ Section \ Save File (1 Name platform for other file). Run mortar try File Save, Oh, no cnn Nag reminder register more as we've always gni Crack! He he

_Dung PeiD scan shows the Code trenh be the "Microsoft Visual C + + 6.0"



_Unpack Done! Bye.

Written by WhyNotBar.

MANUAL unpack ACTIVEMARK 5.x

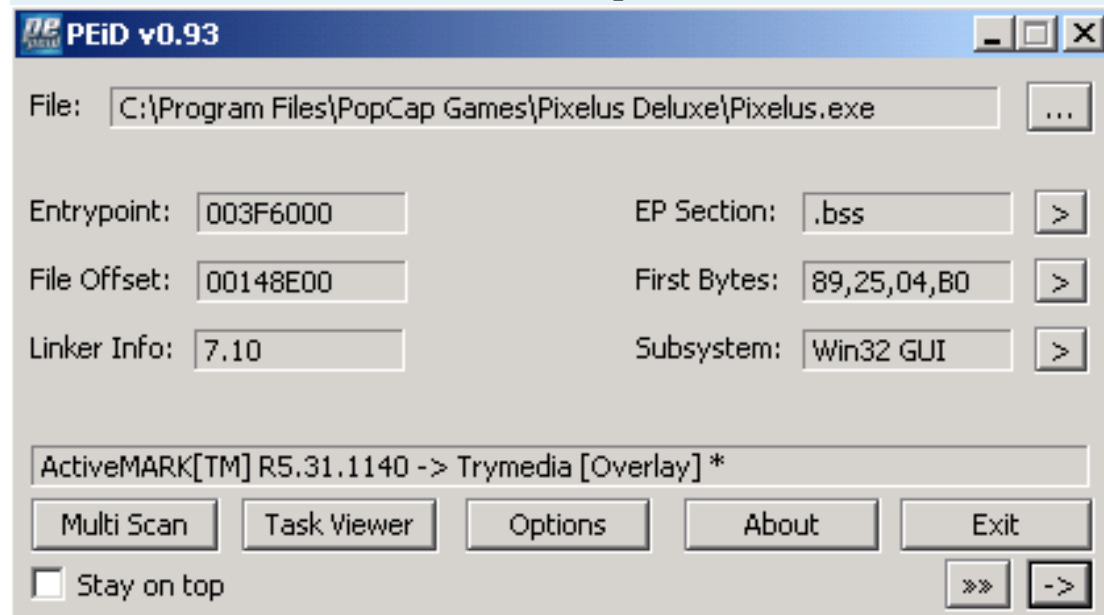
tlandn

Target: Pixelus Delux (www.popcap.com)

Packer: ActiveMark 5.x

Tools: Ollydbg, Imprec, LordPe

As usual, we will use PEID to see the pack in what is



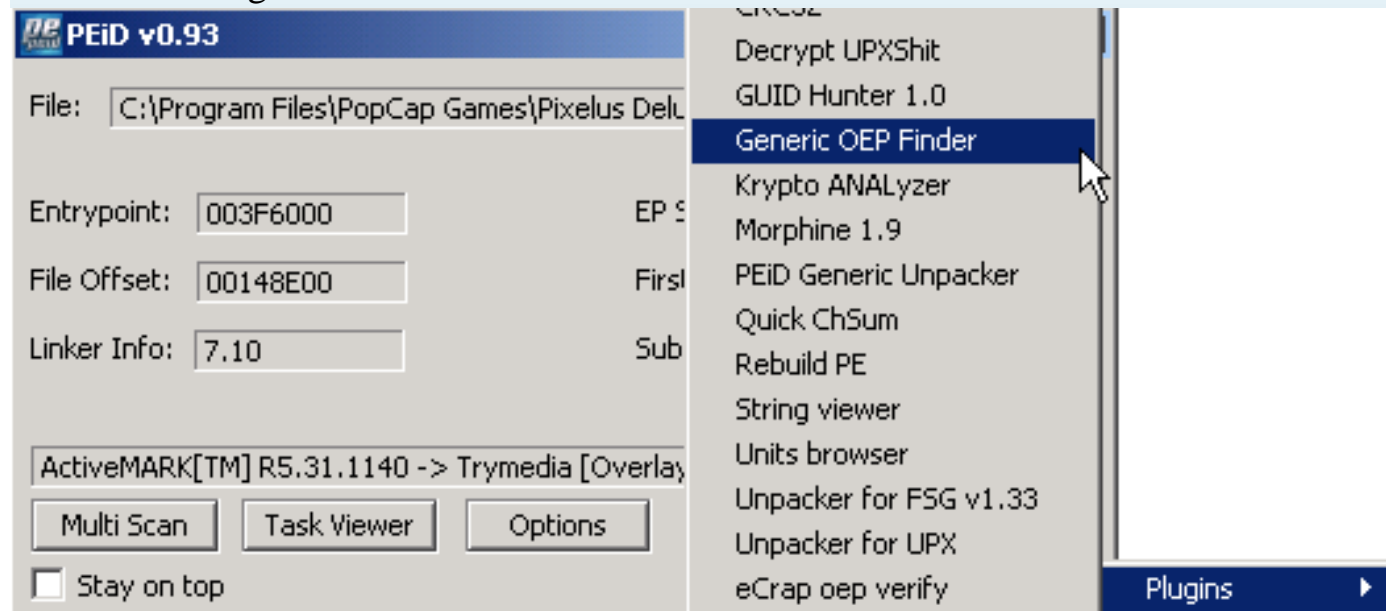
If your PEID not identify ActiveMark, you add the file in the directory userdb.txt PEID the following lines:

[ActiveMARK [TM] R5.31.1140 -> Trymedia]

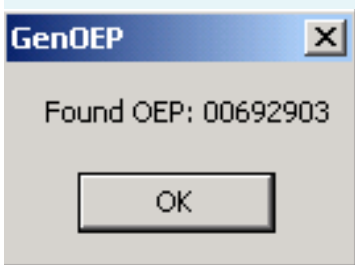
signature = 79117fab9a4a83b5c96b1a48f927b425

ep_only = true

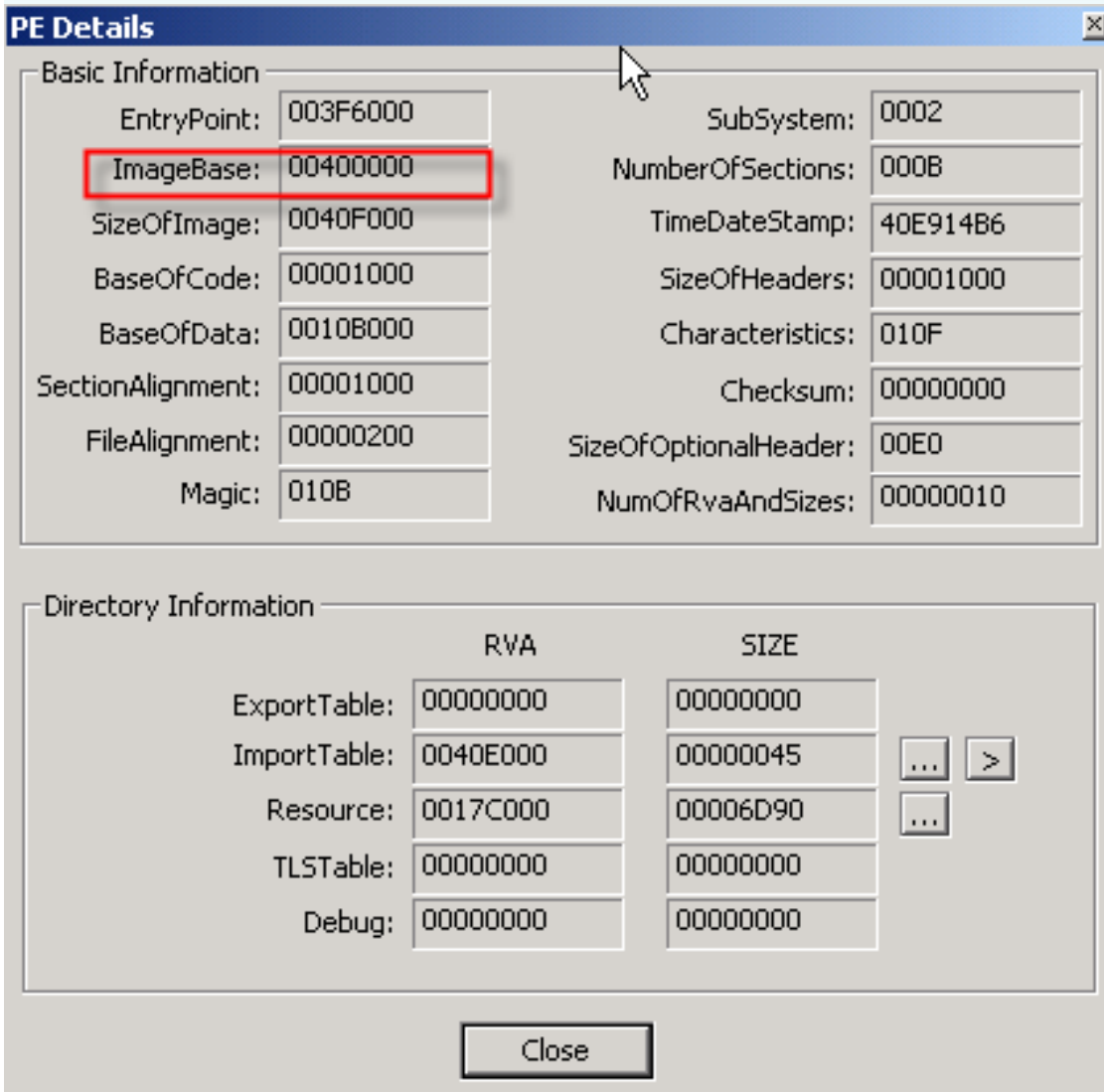
We use the Plugin Finder's Generic OEP PEID.



The result is 00692903:

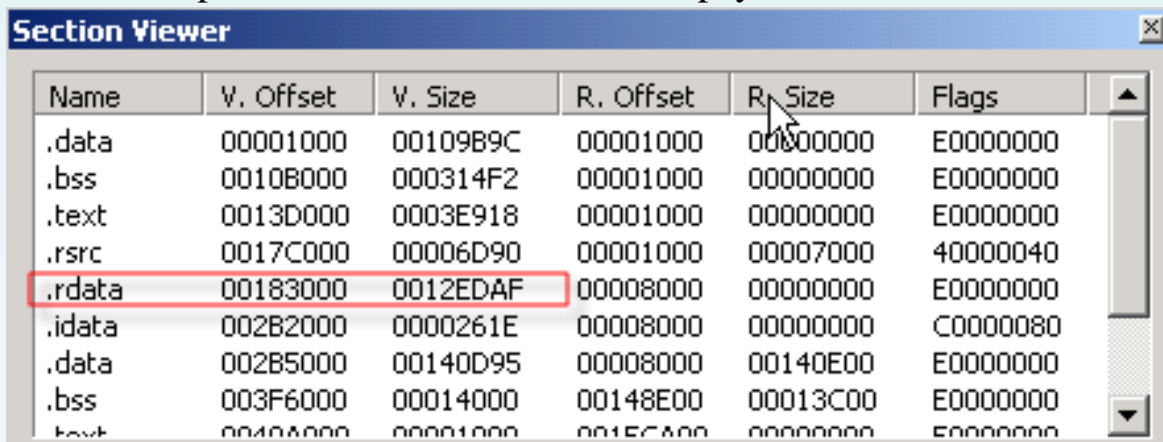


In PEID click button> in the Subsystem. We are:



You note Imagebase is 00400000. Retrieved 00692903 - 00400000 = 00292903. We will consider the address 00292903 is located in any section. This is the section we need to dump there.

Also in PEID press> in the "Section EP. You pay attention to their section below:



.bss	003F6000	00014000	00148E00	00013C00	E0000000
text	0040A000	00001000	0015C800	00000000	E0000000

Close

Why do we choose the section. Rdata this? Simply, you notice V. Offset by this section is 00183000, and V. Size is 0012EDAF. How a community of simple, this section:

Start: 00183000

Finish: 00183000 + 0012EDAF = 002B1DAF

We have: 00183000 < 00292903 < 002B1DAF this section should be selected.

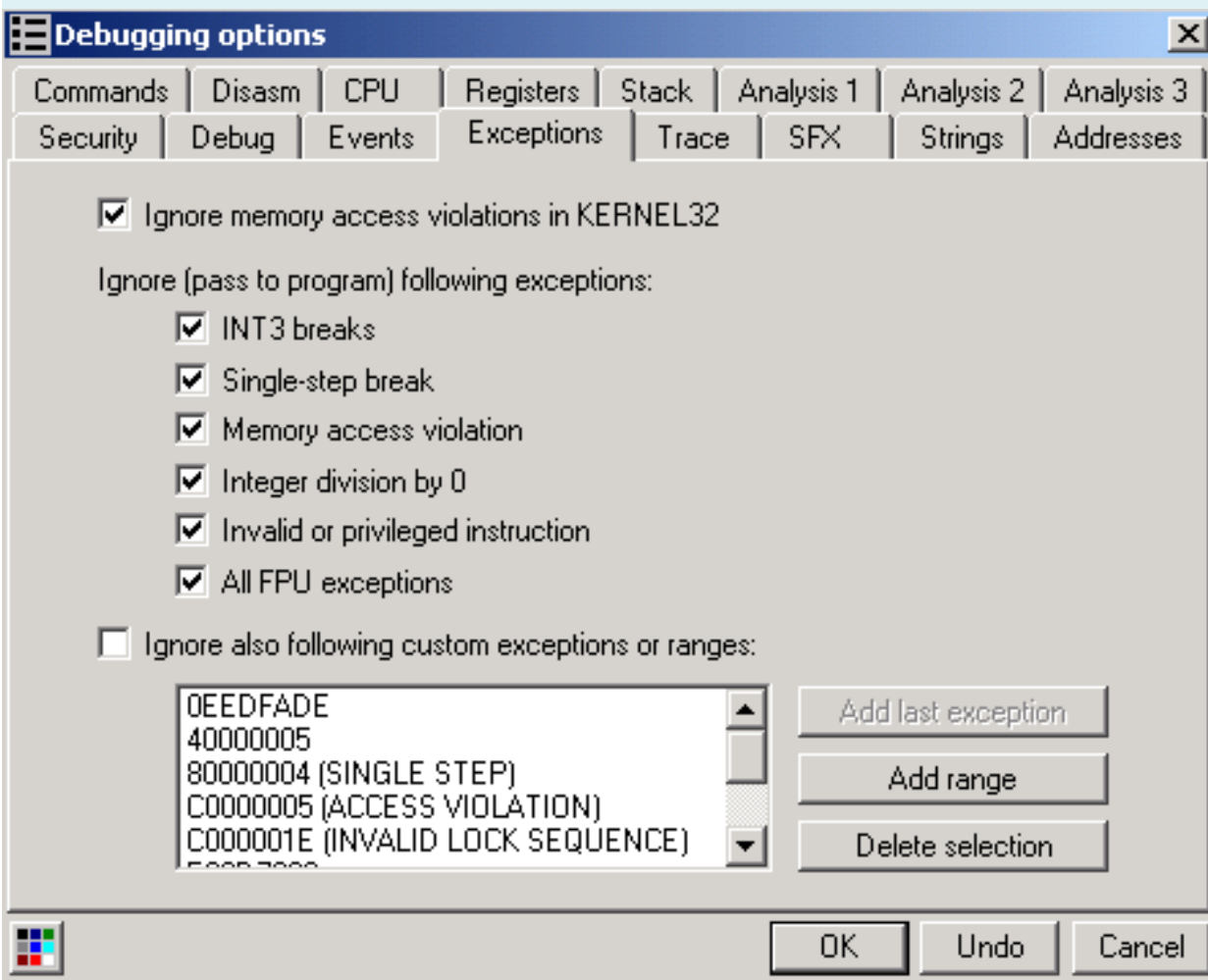
Local first section was considered as completed.

Now we start the program Pixelus. To screen "Welcome to Pixelus it stopped.

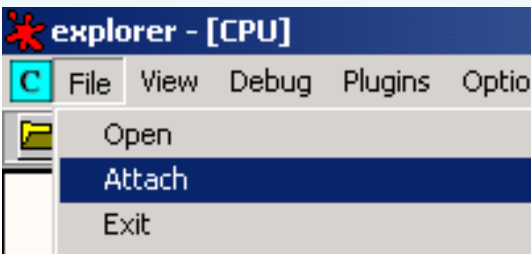


Now we will use OllyDbg Attach to this program.

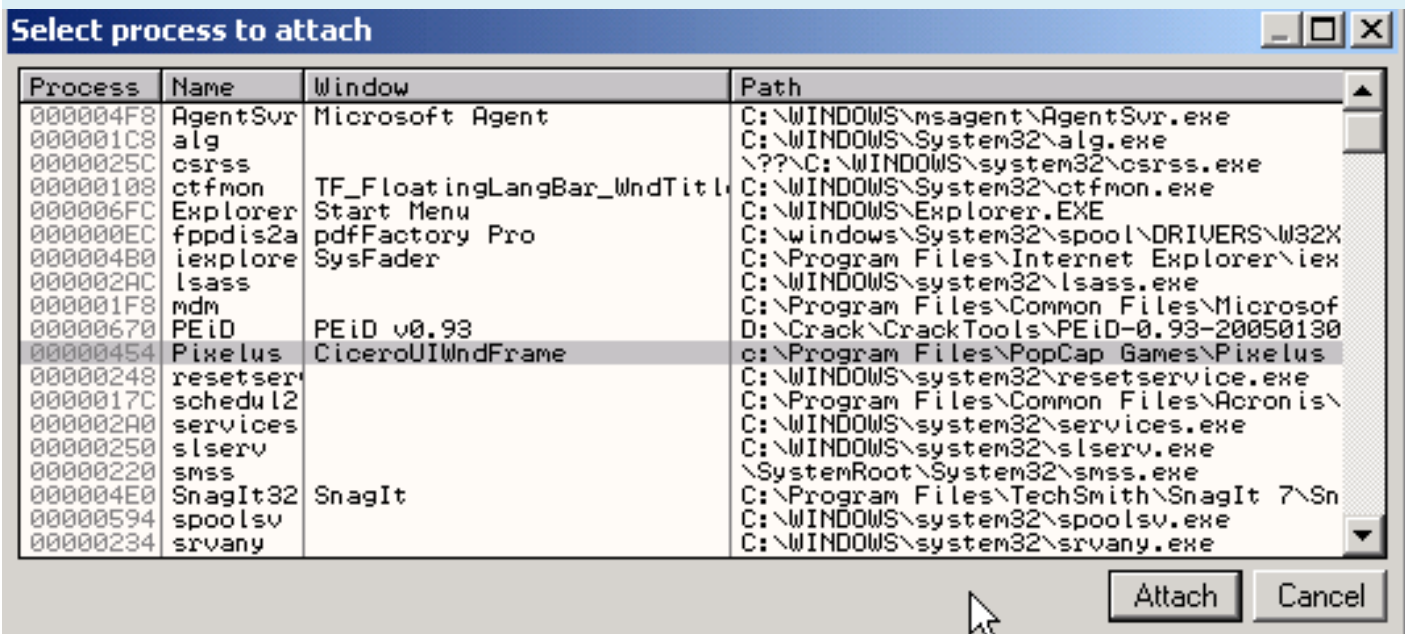
Open OllyDbg. First the original Option Exceptions in the same image:



In File menu select Attach:



Select the program and click the button Pixelus Attach



We stop here:

77F7F571	C3	RETN
77F7F572	8BFF	MOV EDI,EDI
77F7F574	CC	INT3

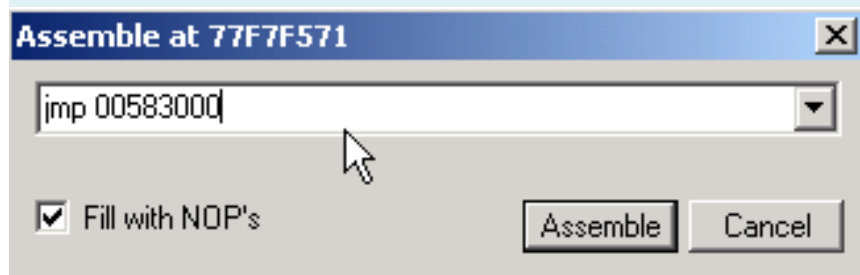
Now we will jump to the section. Rdata we chose above. Note section. Rdata have V. Offset is 00183000. We must add to Imagebase:

$00183000 + 00400000 = 00583000$.

In OllyDbg. Click your mouse to select assemble (or press the Space also)

77F7F571	C3	RETN	
77F7F572	8BFF	MOV EDI,E	Backup
77F7F574	CC	INT3	Copy
77F7F575	C3	RETN	Binary
77F7F576	8BFF	MOV EDI,E	
77F7F578	8B4424 04	MOV EAX,D	
77F7F57C	CC	INT3	Assemble
77F7F57D	C2 0400	RETN 4	Space

Enter JMP 00583000. Click assemble



In the window of the Code Olly would like the following:

77F7F571	- E9 8A3A6088	JMP Pixelus.00583000
77F7F576	8BFF	MOV EDI,EDI
77F7F578	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]
77F7F57C	CC	INT3

Press F7 to jump to 00583000.

00583000	3E:DE12	FICOM WORD PTR DS:[EDX]
00583003	001CDE	ADD BYTE PTR DS:[ESI+EBX*8],BL
00583006	1200	ADC AL,BYTE PTR DS:[EAX]
00583008	08DE	OR DH,BL

Now click the mouse to select the image in

Address	Disassembly	Comment
00583000	SE:DE12	FICOM WORD PTR DS:[EDX]
00583003	001CDE	ADD BYTE PTR DS:[ESI+EBX*8],BL
00583006	1200	ADC AL,BYTE PTR DS:[EAX]
00583008	08DE	OR DH,BL
0058300A	1200	ADC AL,BYTE PTR DS:[EAX]
0058300C	FA	CLI
0058300D	DD12	FST QWORD PTR DS:[EDX]
0058300F	00E8	ADD AL,CH
00583011	DD12	FST QWORD PTR DS:[EDX]
00583013	082E	ADD BYTE PTR DS:[ESI],CH
00583015	DE12	FICOM WORD PTR DS:[EDX]
00583017	00E2 DD1200C6	ADD BYTE PTR DS:[EDX+C6001200],DH
0058301D	DD12	FST QWORD PTR DS:[EDX]
0058301F	00D6	ADD DH,DL
00583021	DD12	FST QWORD PTR DS:[EDX]
00583023	00A4DD 12000000	ADD BYTE PTR SS:[EBP+EBX*8+12],AH
0058302A	0000	ADD BYTE PTR DS:[EAX],AL
0058302C	1100	ADC QWORD PTR DS:[EAX],EAX
0058302E	0000 00000000	ADD BYTE PTR DS:[EAX],AL
00583034	82ED 12	SUB CH,12
00583037	00E8	ADD AL,CH
00583039	ED	IN EAX,DX
0058303A	1200	ADC AL,BYTE PTR DS:[EAX]
0058303C	0C EE	OR AL,0EE
0058303E	1200	ADC AL,BYTE PTR DS:[EAX]
00583040	2C EE	SUB AL,0EE
00583042	1200	ADC AL,BYTE PTR DS:[EAX]
00583044	A0 ED1200C4	MOV AL,BYTE PTR DS:[C40012ED]
00583049	ED	IN EAX,DX
0058304A	1200	ADC AL,BYTE PTR DS:[EAX]
0058304C	F8	CLC
0058304D	ED	IN EAX,DX
0058304E	1200	ADC AL,BYTE PTR DS:[EAX]
00583050	70 ED	JO SHORT Pixelus.0058303F
00583052	1200	ADC AL,BYTE PTR DS:[EAX]
00583054	0000	ADD BYTE PTR DS:[EAX],AL
00583056	0000	ADD BYTE PTR DS:[EAX],AL
00583058	52	PUSH EDX

Registers (FPU)

EAX	7FFDF000
ECX	00000002
EDX	00000003
EBX	00000001
ESP	022FFFFC
EBP	022FFFF4
ESI	00000004
EDI	00000005
EIP	00583000 Pixelus
C 0	ES 0023 32b
P 1	CS 001B 32b
A 0	SS 0023 32b
Z 1	DS 0023 32b
S 0	FS 0038 32b
T 0	GS 0000 NULL
O 0	LastErr ERR0
EFL	00000246 (NO)
ST0	empty +UNORM

Search for

- Find references to
- View
- Copy to executable
- Analysis
- ExtraCopy
- Dump debugged process
- Make dump of process

We will find the API GetCommandLineA (hint: you entered GetCommandLineA, Olly automatically for us).

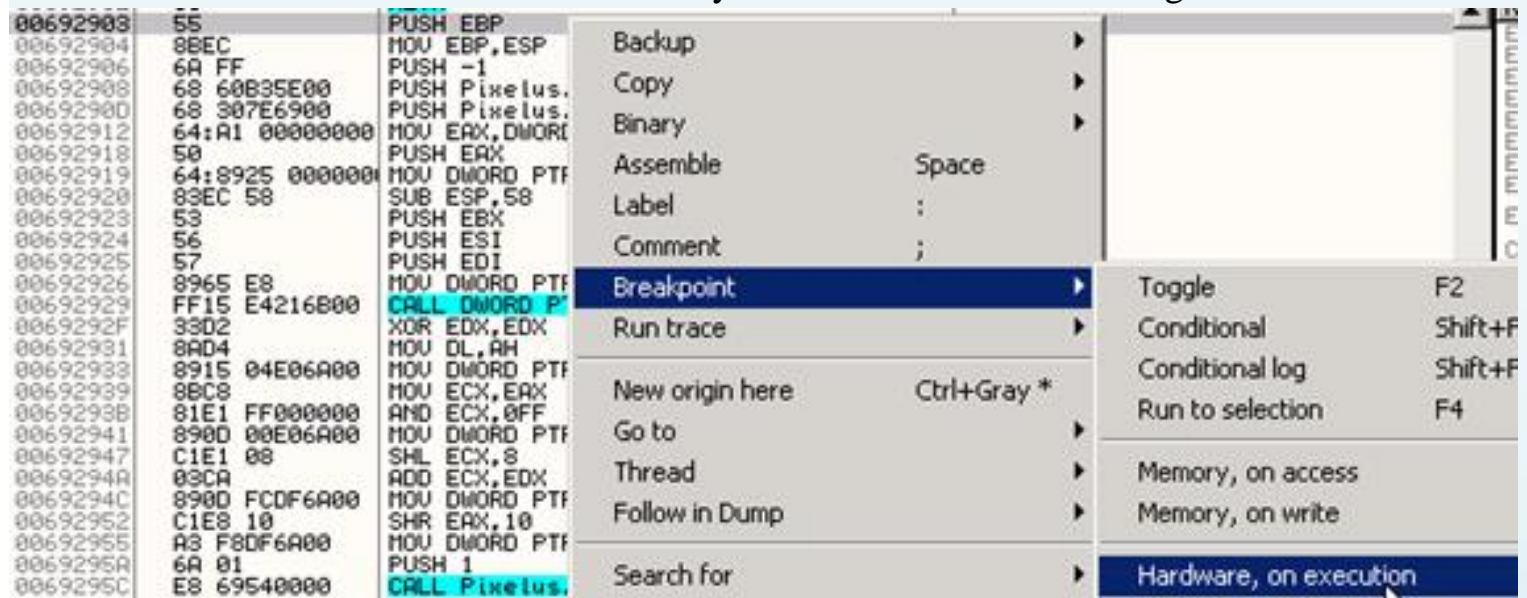
006924B7	CALL Pixelus.006A1DCC	ntdll.RtlUnwind
00692929	CALL DWORD PTR DS:[6B21E4]	kernel32.GetVersion
00692989	CALL DWORD PTR DS:[6B211C]	kernel32.GetCommandLineA
006929B4	CALL DWORD PTR DS:[6B21B0]	kernel32.GetStartupInfoA
006929D7	CALL DWORD PTR DS:[6B218C]	kernel32.GetModuleHandleA

Double-click on that line. We will be here 00692989:

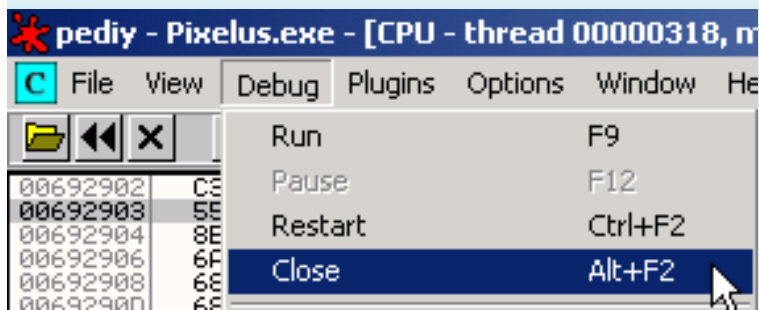
00692903	55	PUSH EBP	
00692904	8BEC	MOV EBP,ESP	
00692906	6A FF	PUSH -1	
00692908	68 60B35E00	PUSH Pixelus.005EB360	
0069290D	68 307E6900	PUSH Pixelus.00697E30	
00692912	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00692918	50	PUSH EAX	
00692919	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00692920	83EC 58	SUB ESP,58	
00692923	53	PUSH EBX	
00692924	56	PUSH ESI	
00692925	57	PUSH EDI	
00692926	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00692929	FF15 E4216B00	CALL DWORD PTR DS:[6B21E4]	kernel32.GetVersion
0069292F	33D2	XOR EDX,EDX	
00692931	8AD4	MOV DL,AH	
00692933	8915 04E06A00	MOV DWORD PTR DS:[6AE004],EDX	
00692939	8BC8	MOV ECX,EAX	
0069293B	81E1 FF000000	AND ECX,0FF	
00692941	890D 00E06A00	MOV DWORD PTR DS:[6AE000],ECX	
00692947	C1E1 08	SHL ECX,8	
0069294A	03CA	ADD ECX,EDX	
0069294C	890D FCD6A00	MOV DWORD PTR DS:[6ADFFC],ECX	
00692952	C1E8 10	SHR EAX,10	
00692955	A3 F8DF6A00	MOV DWORD PTR DS:[6ADFF8],EAX	
0069295A	6A 01	PUSH 1	
0069295C	E8 69540000	CALL Pixelus.00697DCA	
00692961	59	POP ECX	
00692962	85C0	TEST EAX,EAX	
00692964	75 08	JNZ SHORT Pixelus.0069296E	
00692966	6A 1C	PUSH 1C	
00692968	E8 C3000000	CALL Pixelus.00692A30	
0069296D	59	POP ECX	
0069296E	E8 99490000	CALL Pixelus.0069730C	
00692973	85C0	TEST EAX,EAX	
00692975	75 08	JNZ SHORT Pixelus.0069297F	
00692977	6A 10	PUSH 10	
00692979	E8 B2000000	CALL Pixelus.00692A30	
0069297E	59	POP ECX	
0069297F	33F6	XOR ESI,ESI	
00692981	8975 FC	MOV DWORD PTR SS:[EBP-4],ESI	
00692984	E8 10510000	CALL Pixelus.00697A99	
00692989	FF15 1C216B00	CALL DWORD PTR DS:[6B211C]	kernel32.GetCommandLineA
0069298F	A3 84F76A00	MOV DWORD PTR DS:[6AF784],EAX	

Scroll window Olly up points to start the program. You note that in 00692903 PUSH EBP line is we need to find.

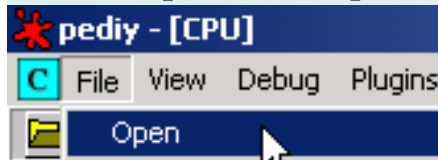
Click the cursor to the line 00692903. Click your mouse to select the image as:



Now we will close the program. Debug menu select Close



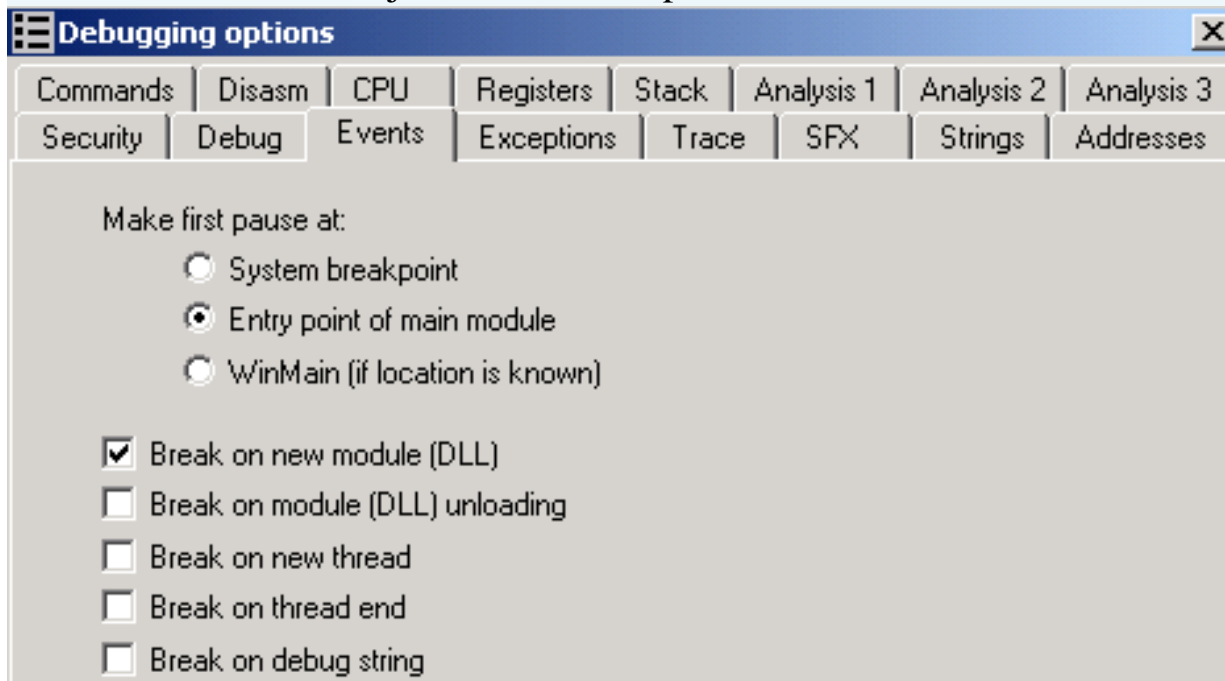
We will open the file pixelus.exe. In the menu choose Open File



And select the file pixelus.exe. We will stop here:

007F6000	8925 04B08000	MOV DWORD PTR DS:[80B004],ESP	
007F6006	68 30607F00	PUSH Pixelus.007F6030	
007F600B	EB 03	JMP SHORT Pixelus.007F6010	
007F600D	D822	FSUB DWORD PTR DS:[EDX]	
007F600F	74 64	JE SHORT Pixelus.007F6075	
007F6011	FF35 00000000	PUSH DWORD PTR DS:[0]	
007F6017	EB 02	JMP SHORT Pixelus.007F601B	
007F6019	FF15 64892500	CALL DWORD PTR DS:[258964]	

Before we run a little adjustment in the Option. Press Alt-O and the like in the picture:



Click OK. Now press F9. We found out the window "executable Modules"

The screenshot shows the 'Executable modules' window. It displays a list of loaded modules with columns for Base, Size, Entry, Name, File version, and Path.

Base	Size	Entry	Name	File version	Path
00400000	0040F000	007F6000	Pixelus	1. 0. 0. 1	C:\Program Files\PopCap Games\Pixelus Deluxe\Pixelus.exe
77E60000	000E5000	77E7A241	kernel32	5.1.2600.0 (xpo	C:\WINDOWS\system32\kernel32.dll
77F50000	000A9000		ntdll	5.1.2600.0 (xpo	C:\WINDOWS\System32\ntdll.dll

Continue pressing F9 until we break at 00,692,903.

00692903	55	PUSH EBP	
00692904	8BEC	MOV EBP,ESP	
00692906	6A FF	PUSH -1	
00692908	68 60B35E00	PUSH Pixelus.005EB360	
0069290D	68 307E6900	PUSH Pixelus.00697E30	
00692912	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00692918	50	PUSH EAX	

This will is our OEP. Using OllyDump to dump the program:

Entry Point: -> Modify:

Base of Code: Base of Data:

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Charactaristic
.data	00109B9C	00001000	00109B9C	00001000	E0000000
.bss	000314F2	0010B000	000314F2	0010B000	E0000000
.text	0003E918	0013D000	0003E918	0013D000	E0000000
.rsrc	00006D90	0017C000	00006D90	0017C000	40000040
.rdata	0012EDAF	00183000	0012EDAF	00183000	E0000000
.idata	0000261E	002B2000	0000261E	002B2000	C0000080
.data	00140D95	002B5000	00140D95	002B5000	E0000000

☐ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

I named the file dump is a.exe.

To make the file smaller in size than. We will remove the last section 4 a.exe file.

Using LordPE open file a.exe. Click to Sections in the Section Table

[PE Editor] - c:\program files\popcap games\pixelus deluxe\a.exe

Basic PE Header Information

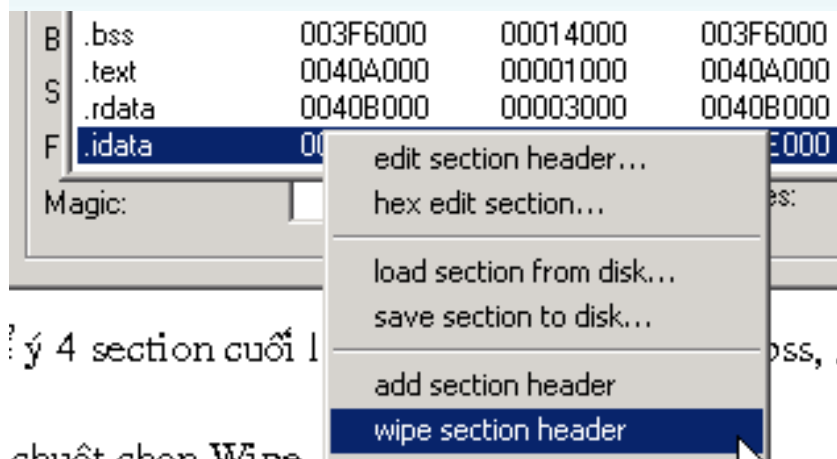
[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
.rsrc	0017C000	00006D90	0017C000	00006D90	40000040
.rdata	00183000	0012EDAF	00183000	0012EDAF	E0000000
.idata	002B2000	0000261E	002B2000	0000261E	C0000080
.data	002B5000	00140D95	002B5000	00140D95	E0000000
.bss	003F6000	00014000	003F6000	00014000	E0000000
.text	0040A000	00001000	0040A000	00001000	E0000000
.rdata	0040B000	00003000	0040B000	00003000	E0000000
.idata	0040E000	00000045	0040E000	00000045	C0000040

Magic: NumOfRvaAndSizes:

To the last section 4 that in turn (on down) is. Bss. Text. Rdata. Idata

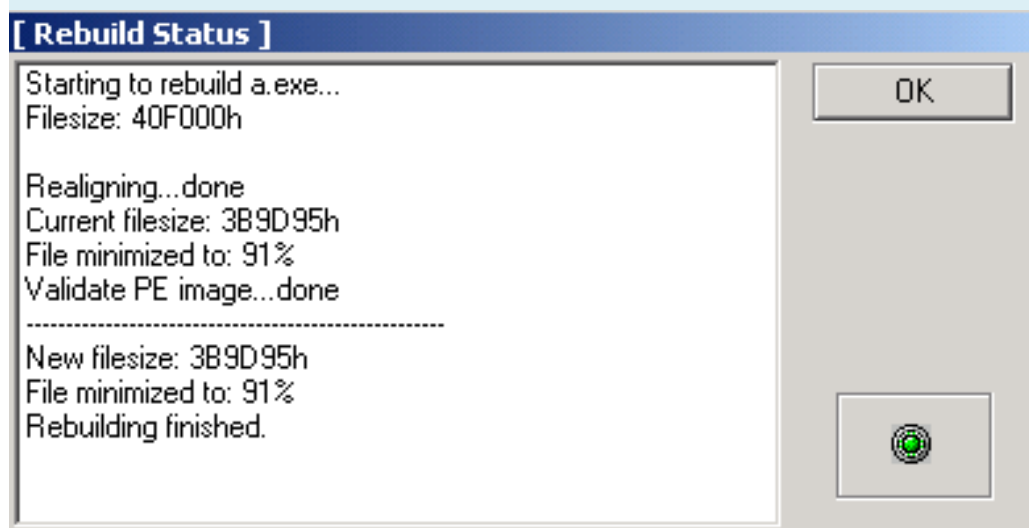
Click your mouse to select "Wipe section header" on each section in 4 on the section.



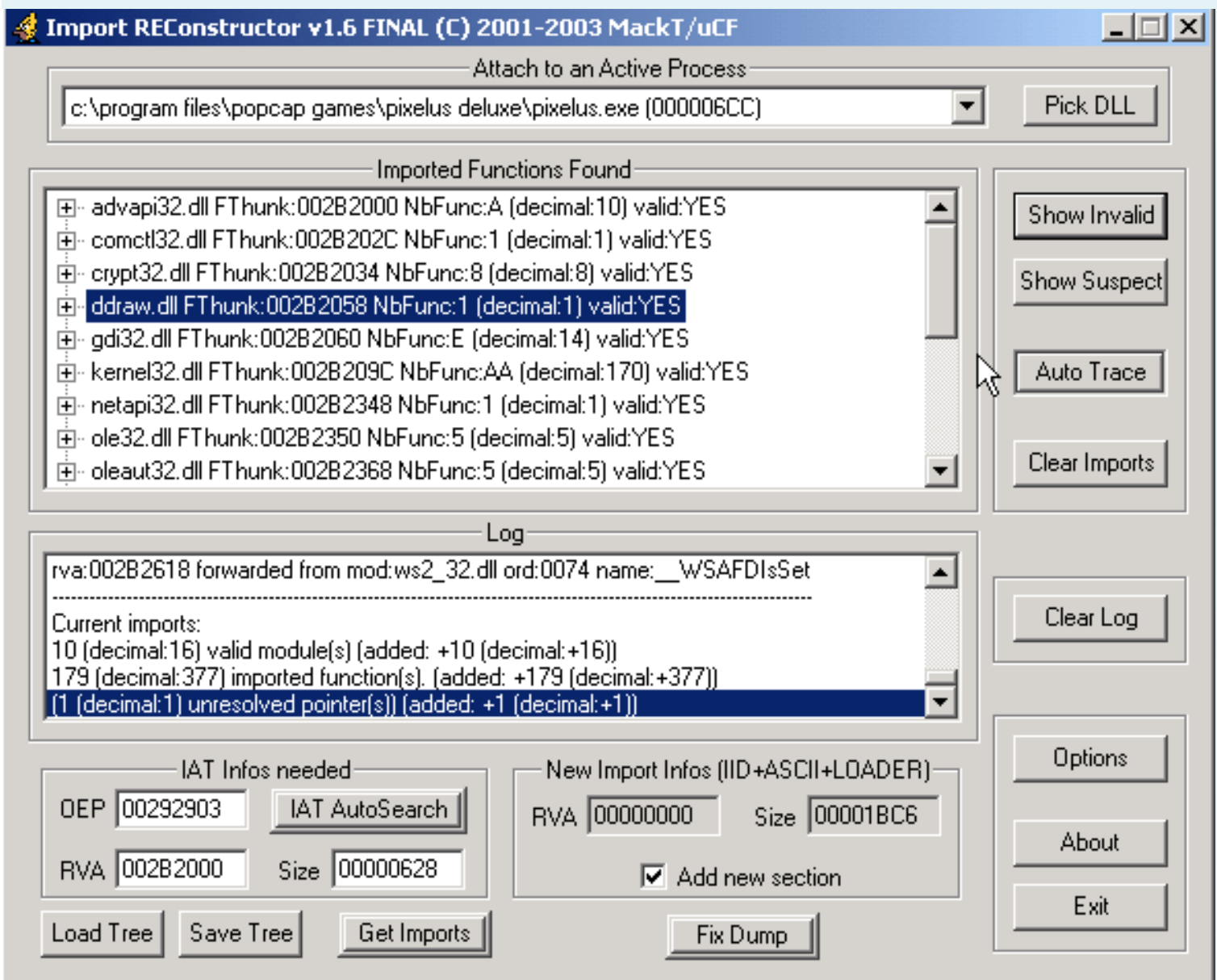
Once removed, we complete the remaining section of the following:

[Section Table]						
Name	VOffset	VSize	ROffset	RSize	Flags	
.data	00001000	00109B9C	00001000	00109B9C	E0000000	
.bss	0010B000	000314F2	0010B000	000314F2	E0000000	
.text	0013D000	0003E918	0013D000	0003E918	E0000000	
.rsrc	0017C000	00006D90	0017C000	00006D90	40000040	
.rdata	00183000	0012EDAF	00183000	0012EDAF	E0000000	
.idata	002B2000	0000261E	002B2000	0000261E	C0000080	
.data	002B5000	00140D95	002B5000	00140D95	E0000000	

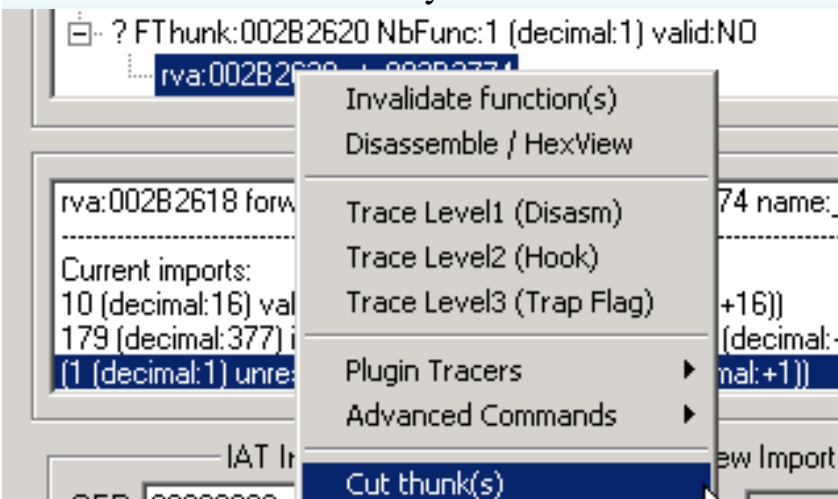
Close window Section Table. Click Save. Click OK. We will rebuild the file a.exe. Also use LordPE. Click rebuild PE. Select File a.exe.



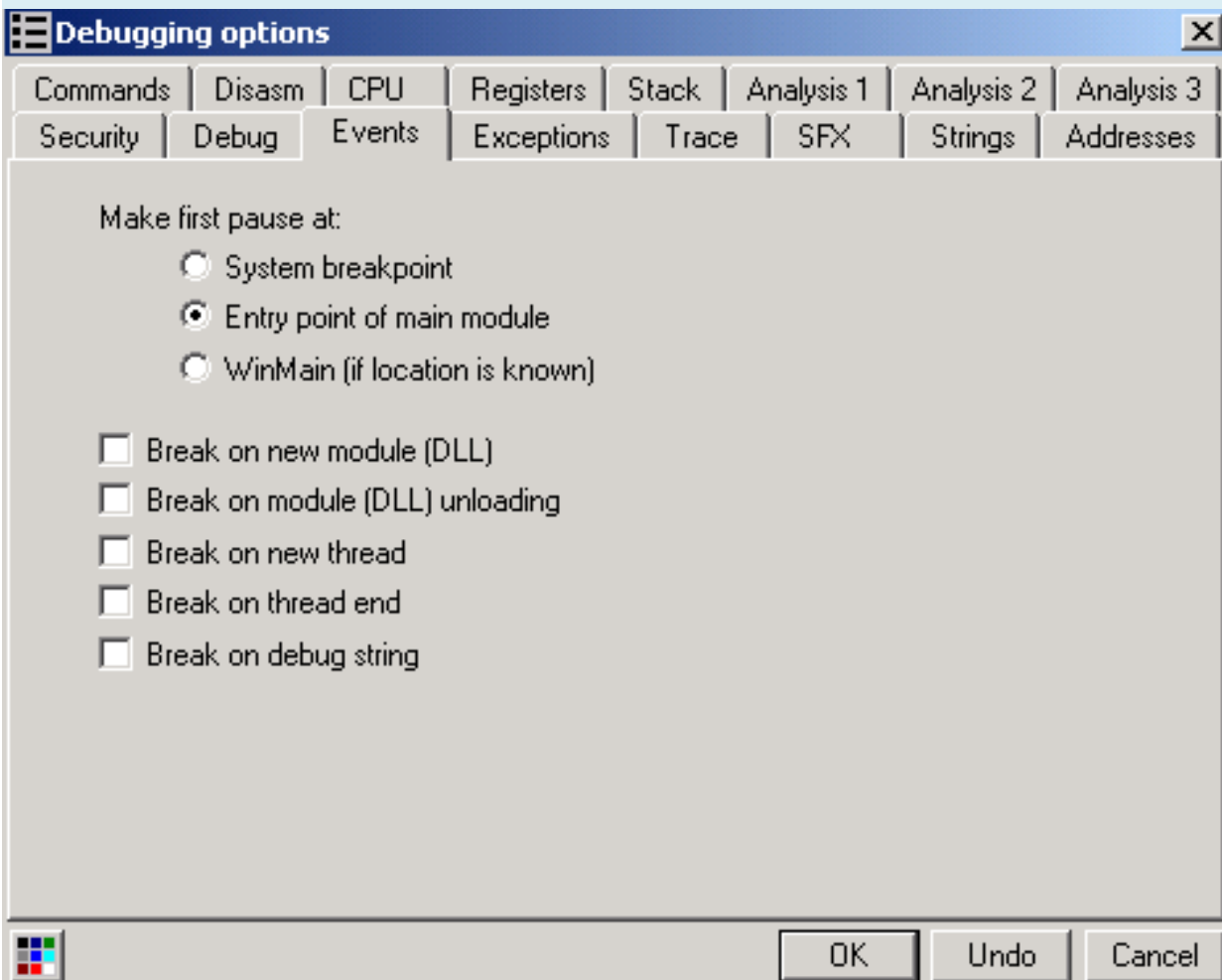
Next we will fix IAT. Open Imprec, OEP enter 00292903, IAT AutoSearch click, click OK, click Get Imports, we are:



Click Show Invalid. Click your mouse to select "Cut Thunks"



Now we have complete IAT. Click "Fix dump", select the file a.exe. We will file a_.exe Test file a_.exe. We do not see any at all. Next we will fix a few seats to exe files can run. Open Olly. Edit Option parameters as following:



Click OK. Using OllyDbg open file a_.exe. Click your mouse to select the image in

00692903	55	PUSH EBP		
00692904	8BEC	MOV EBP,ESP	Backup	
00692906	6A FF	PUSH -1	Copy	
00692908	68 60B35E00	PUSH a_.005	Binary	
0069290D	68 307E6900	PUSH a_.006	Assemble	Space
00692912	64:A1 00000000	MOV EAX,DWC	Label	:
00692918	50	PUSH EAX	Comment	;
00692919	64:8925 00000000	MOV DWORD F	Breakpoint	
00692920	83EC 58	SUB ESP,58	Run trace	
00692923	53	PUSH EBX	Go to	
00692924	56	PUSH ESI	Follow in Dump	
00692925	57	PUSH EDI		
00692926	8965 E8	MOV DWORD F	Search for	Name (label) in current mod
00692929	FF15 E4216B00	CALL DWORD	Find references to	Name in all modules
0069292F	33D2	XOR EDX,EDX	View	Command
00692931	8AD4	MOV DL,AH	Copy to executable	Sequence of commands
00692933	8915 04E06A00	MOV DWORD F	Analysis	Constant
00692939	8BC8	MOV ECX,EA	ExtraCopy	Binary string
0069293B	81E1 FF000000	AND ECX,0FF	Dump debugged process	
00692941	890D 00E06A00	MOV DWORD F	Make dump of process	
00692947	C1E1 08	SHL ECX,8		
0069294A	03CA	ADD ECX,ED		
0069294C	890D FCD6A00	MOV DWORD F		
00692952	C1E8 10	SHR EAX,10		
00692955	A3 F8DF6A00	MOV DWORD F		
0069295A	6A 01	PUSH 1		
0069295C	E8 69540000	CALL a_.006		
00692961	59	POP ECX		
00692962	85C0	TEST EAX,EA		
00692964	75 08	JNZ SHORT a		
00692966	6A 1C	PUSH 1C		
00692968	E8 C3000000	CALL a_.006		
0069296D	59	POP ECX		
0069296E	E8 99490000	CALL a_.006		
00692973	85C0	TEST EAX,EA		
00692975	75 08	JNZ SHORT a		
00692977	6A 10	PUSH 10		

Click the Destination column to sort by name for easy search.

Address	Disassembly	Destination
0068F439	CALL 00D1EA08	
0068F482	CALL 00D1E5FA	

We will find the API GetModuleHandleA. Enter GetModuleHandleA, Olly will automatically go to the API for us.

00697D36	CALL	DWORD	PTR	DS:[<&kernel32.GetModulele	kernel32.GetModuleFileNameA
00697FAE	CALL	DWORD	PTR	DS:[<&kernel32.GetModulele	kernel32.GetModuleFileNameA
005EC7A3	CALL	DWORD	PTR	DS:[<&kernel32.GetModulele	kernel32.GetModuleHandleA
005FD6D2	CALL	DWORD	PTR	DS:[<&kernel32.GetModulele	kernel32.GetModuleHandleA
0060150D	CALL	DWORD	PTR	DS:[<&kernel32.GetModulele	kernel32.GetModuleHandleA
0065C51E	CALL	DWORD	PTR	DS:[<&kernel32.GetModulele	kernel32.GetModuleHandleA
0065D985	CALL	DWORD	PTR	DS:[<&kernel32.GetModulele	kernel32.GetModuleHandleA
00677C69	CALL	DWORD	PTR	DS:[<&kernel32.GetModulele	kernel32.GetModuleHandleA
00677F9C	CALL	DWORD	PTR	DS:[<&kernel32.GetModulele	kernel32.GetModuleHandleA
006929D7	CALL	DWORD	PTR	DS:[<&kernel32.GetModulele	kernel32.GetModuleHandleA
00697C5F	CALL	DWORD	PTR	DS:[<&kernel32.GetModulele	kernel32.GetModuleHandleA
0069B5CE	CALL	DWORD	PTR	DS:[<&kernel32.GetModulele	kernel32.GetModuleHandleA

We have a lot GetModuleHandleA results. The problem is we will select results. Generally, the results first and the second will be the first we need to find. Double-click the mouse on the first results 5EC7A3.

005EC79B	^	0F85 75FFFFFF	JNZ a_.005EC716	
005EC7A1		6A 00	PUSH 0	
005EC7A3		FF15 8C216B00	CALL DWORD PTR DS:[<&kernel32.GetModulele	kernel32
005EC7A9		8B0D 9C635E00	MOV ECX,DWORD PTR DS:[5E639C]	
005EC7AF		03C8	ADD ECX,EAX	

You note that in line 2:

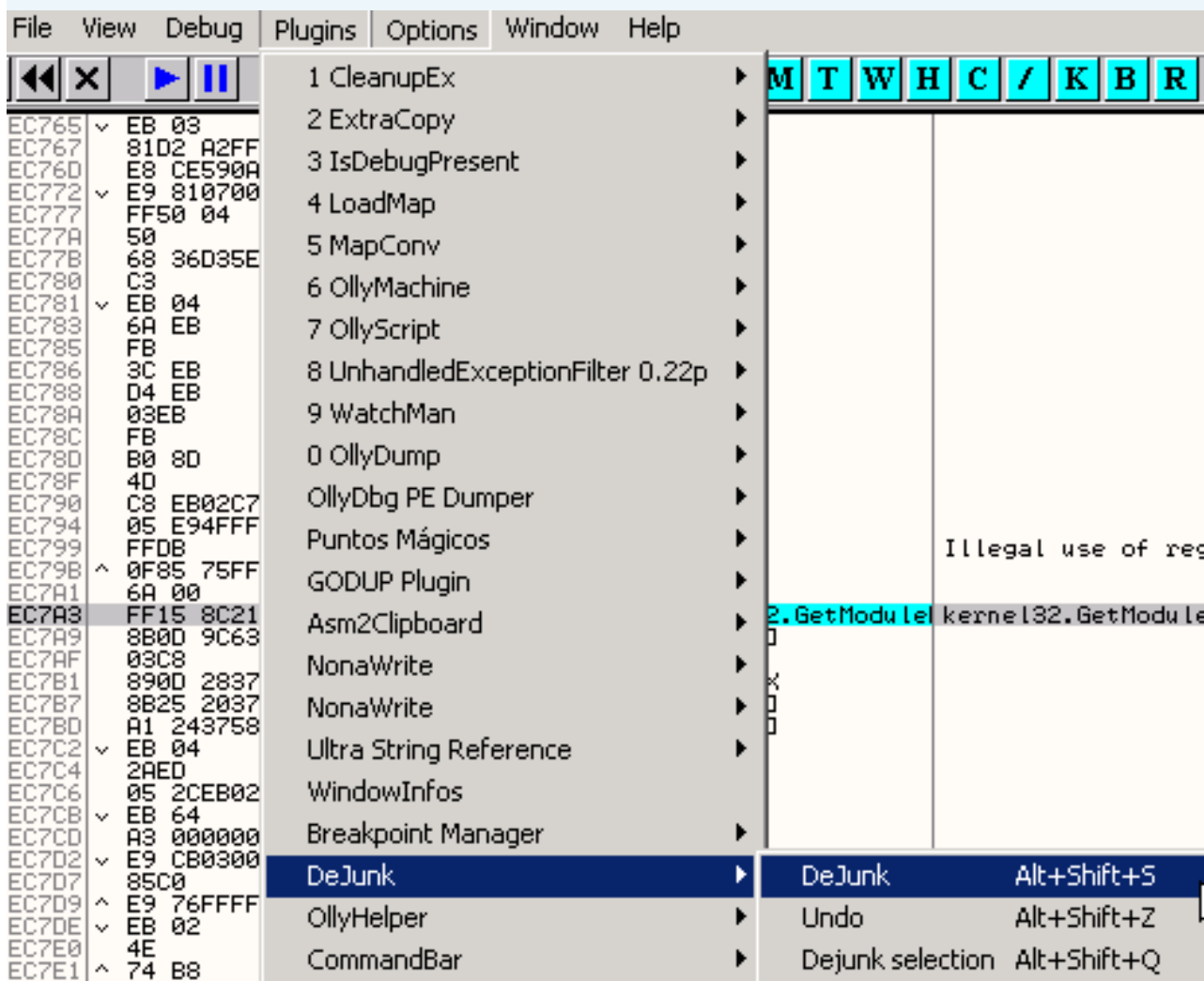
005EC7AF ADD ECX, EAX

If you see this line then you can conclude this is the result we need to find. If not you see the double-click on each result and find the ADD ECX, EAX.

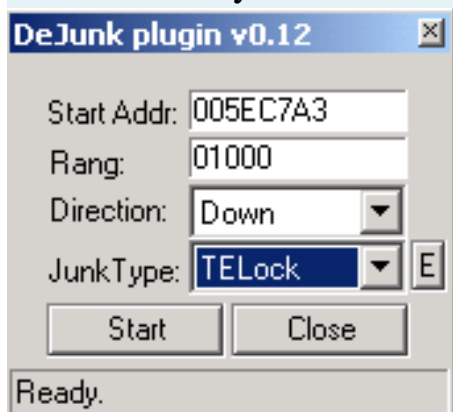
We are **5EC7A3**. We know this is GetModuleHandleA functions that we need. Above the function GetModuleHandleA we will have 2 orders jumped JE and JNZ. Order JNZ we have seen in 005EC79B

005EC79B	^	0F85 75FFFFFF	JNZ a_.005EC716	
005EC7A1		6A 00	PUSH 0	
005EC7A3		FF15 8C216B00	CALL DWORD PTR DS:[<&kernel32.GetModulele	kernel32
005EC7A9		8B0D 9C635E00	MOV ECX,DWORD PTR DS:[5E639C]	
005EC7AF		03C8	ADD ECX,EAX	

We will find the commands on the command JE JNZ. First we will use DeJunk plugin. Select the pictures in



Start addr entry into **005EC7A3**. Select the image as:



Click Start. Click OK.

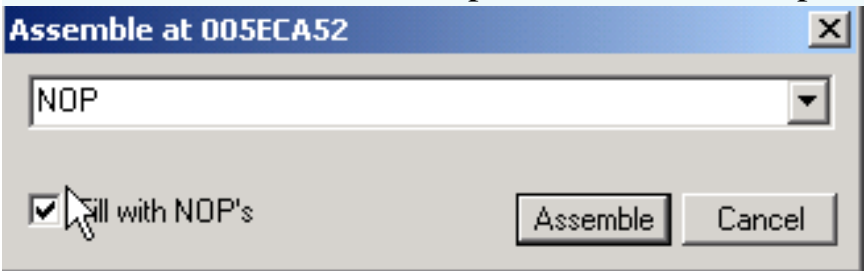
Click the 005EC79B. Click your mouse to select the image in

005EC79B	JNZ a_.005EC716	(Initial CPU selection)
005ECA69	JMP a_.005EC79B	

005ECA4D	✓ E9 0D060000	JMP a_005ED05F
005ECA52	✓ 0F84 58050000	JE a_005ECFB0
005ECA58	813D 90635E00	CMP DWORD PTR DS:[5E6390],416E6454
005ECA62	90	NOP
005ECA63	90	NOP
005ECA64	90	NOP
005ECA65	90	NOP
005ECA66	90	NOP
005ECA67	90	NOP
005ECA68	90	NOP
005ECA69	^ E9 2DFDFFFF	JMP a_005EC79B
005ECA6E	85C0	TEST EAX,EAX

005ECA52 JE a .005ECFB0

Click the 005ECA52. Press Space. Enter as in the picture



005ECA4D	^	E9 00060000	JMP	Comment	
005ECA52		90	NOP	Breakpoint	▶
005ECA53		90	NOP		
005ECA54		90	NOP	Run trace	▶
005ECA55		90	NOP		
005ECA56		90	NOP		
005ECA57		90	NOP	New origin here	Ctrl+Gray *
005ECA58		813D 90635E00	CMP	Go to	▶
005ECA62		90	NOP		
005ECA63		90	NOP	Follow in Dump	▶
005ECA64		90	NOP		
005ECA65		90	NOP		
005ECA66		90	NOP	Search for	▶
005ECA67		90	NOP	Find references to	▶
005ECA68		90	NOP	View	▶
005ECA69	^	E9 2DFDFFFF	JMP		
005ECA6E		85C0	TEST		
005ECA70		90	NOP		
005ECA71		90	NOP	Copy to executable	▶
005ECA72		90	NOP		

file:///C:/RCE%20Unpacking%20eBook%20[Tra...i]/Manual%20Unpack%20ActiveMark%205.x.htm (15 of 19) [1/9/2009 9:44:47 LithiumLi]

001B1C52	90	NOP	
001B1C53	90	NOP	
001B1C54	90	NOP	
001B1C55	90	NOP	
001B1C56	90	NOP	
001B1C57	90	NOP	
001B1C58	813D 90635E00	CMP DWORD PTR DS	
001B1C62	EB 02	JMP SHORT 001B1C	
001B1C64	EB 02	JMP SHORT 001B1C	
001B1C66	EB 01	JMP SHORT 001B1C	
001B1C68	EB E9	JMP SHORT 001B1C	
001B1C6A	2D F0FFFF85	SUB EAX, 85FFFFFFD	
001B1C6F	C0EB 02	SHR BL, 2	
001B1C72	49	DEC ECX	
001B1C73	74 0F	JE SHORT 001B1C8	

Backup
Copy
Binary
Assemble
Search for
Save file

Named a_1.exe. Test program:



We do not see a window up to us to purchase the program again. To this we can be pleased. However, there is a problem here. When we close the window. An error message appear:

Pixelus Game

Pixelus Game has encountered a problem and needs to close. We are sorry for the inconvenience.



If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.

We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, [click here](#).

Send Error Report

Don't Send

We will fix this error.

Using Olly a_1.exe open the file. Press Shift-F9 to run the program. When the window is closing on the window again (press X button on the right corner). Olly will stop here:

Windows code:

014EC2D4	0023	ADD BYTE PTR DS:[EBX],AH
014EC2D6	04 00	ADD AL,0
014EC2D8	0B00	OR EAX,DWORD PTR DS:[EAX]
014EC2DA	0000	ADD BYTE PTR DS:[EAX],AL
014EC2DC	00E0	ADD AL,AH

Registers window:

Registers (FPU)		
EAX	0050ABD1	a_1.0050ABD1
ECX	0057A059	a_1.0057A059
EDX	00DBB108	
EBX	7FFD0000	ASCII "????????????????????"
ESP	0012FE6C	UNICODE "z0"
EBP	0012FE94	
ESI	00000001	
EDI	00000000	
EIP	014EC2D4	

Stack window:

0012FE6C	004F007A	RETURN to a_1.004F007A
0012FE70	00000000	
0012FE74	00000000	
0012FE78	7FFD0000	ASCII "????????????????????"
0012FE7C	0012FE70	
0012FE80	0012FA94	

You note that we will return 004F007A (to red in the image above).

In the window code. G.Nhap Press Ctrl-to 004f007a.

Enter expression to follow

4f007a

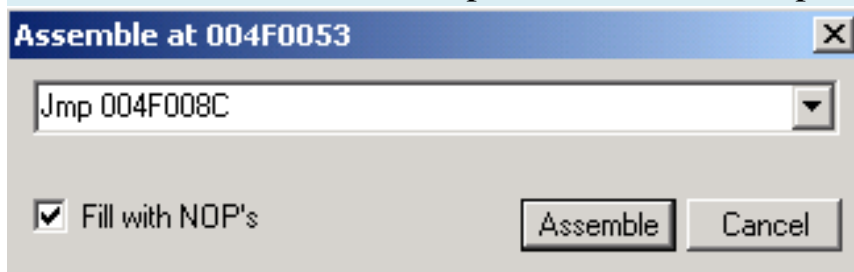
OK Cancel

Click OK.

004F0042	8935 B8A05700	MOV DWORD PTR DS:[57A0B8],ESI
004F0048	8A45 10	MOV AL,BYTE PTR SS:[EBP+10]
004F004B	A2 B4A05700	MOV BYTE PTR DS:[57A0B4],AL
004F0050	397D 0C	CMP DWORD PTR SS:[EBP+C],EDI
004F0053	75 37	JNZ SHORT a_1.004F008C
004F0055	393D 0CB95700	CMP DWORD PTR DS:[57B90C],EDI
004F005B	74 1F	JE SHORT a_1.004F007C
004F005D	A1 08B95700	MOV EAX,DWORD PTR DS:[57B908]
004F0062	83E8 04	SUB EAX,4
004F0065	A3 08B95700	MOV DWORD PTR DS:[57B908],EAX
004F006A	3B05 0CB95700	CMP EAX,DWORD PTR DS:[57B90C]
004F0070	72 0A	JB SHORT a_1.004F007C
004F0072	8B00	MOV EAX,DWORD PTR DS:[EAX]
004F0074	3BC7	CMP EAX,EDI
004F0076	74 E5	JE SHORT a_1.004F005D
004F0078	FFD0	CALL EAX
004F007A	EB E1	JMP SHORT a_1.004F005D

In line 004F0078 CALL EAX will cause problems. There are many ways to solve this problem. Here I choose at the 004F0053 JNZ SHORT a_1.004F008C we will revise the order to always jump JMP overcome command CALL EAX -> do not cause problems.

Click 004F0078 line. Press Space. Enter as in the picture:



Click assemble. Click your mouse to select the image as:

The screenshot displays the Immunity Debugger interface. The assembly pane shows the following code:

```

004F0053  EB 37      JMP SHORT a_1.004F008C
004F0055  393D 0CB95700 CMP DWORD PTR DS:[57B90C],EDI
004F005B  74 1F      JE SHORT a_1.004F007C
004F005D  A1 08B95700 MOV EAX,DWORD PTR DS:[57B908]
004F0062  83E8 04    SUB EAX,4
004F0065  A3 08B95700 MOV DWORD PTR DS:[57B908],EAX
004F006A  3B05 0CB95700 CMP EAX,DWORD PTR DS:[57B90C]
004F0070  72 0A      JB SHORT a_1.004F007C
004F0072  8B00      MOV EAX,DWORD PTR DS:[EAX]
004F0074  3BC7      CMP EAX,EDI
004F0076  74 E5      JE SHORT a_1.004F005D
004F0078  FFD0      CALL EAX
004F007A  EB E1      JMP SHORT a_1.004F005D
004F007C  68 D0D05300 PUSH a_1.0053D0D0
004F0081  B8 C8D05300 MOV EAX,a_1.0053D0C8
004F0086  E8 01FFFFFF CALL a_1.004EFF8C
004F008B  59        POP ECX
004F008C  68 DCD05300 PUSH a_1.0053D0DC
004F0091  B8 D4D05300 MOV EAX,a_1.0053D0D4
004F0096  E8 F1FFFFFF CALL a_1.004EFF8C
004F009B  59        POP ECX
004F009C  834D FC FF OR DWORD PTR SS:[EBP-4],FF
004F00A0  F8 18000000 CALL a_1.004F00BD
004F00A5  397D 10    CMP DWORD PTR SS:[EBP+10],EDI

```

The hex dump pane shows the following data:

```

Address Hex dump
006B2000 9A 18 D0 77 08 59 D0 77 9F 83 D0 77 55 5C C
006B2010 E3 81 D0 77 3E 7F D0 77 EA 22 D0 77 23 AE C
006B2020 D7 23 D0 77 F8 59 D0 77 08 00 00 00 19 52 C
006B2030 08 00 00 00 D7 57 2C 76 A4 4E 2C 76 B7 57 C
006B2040 57 E0 2C 76 FF D5 2C 76 C5 40 2C 76 17 64 C
006B2050 08 6F 2C 76 00 00 00 00 FD CD 01 51 00 00 C
006B2060 B8 6D C7 77 5A F8 C7 77 7C D4 C7 77 B8 20 C
006B2070 89 28 C7 77 1D 53 C7 77 6D 58 C7 77 CC D2 C
006B2080 45 D8 C7 77 00 31 C7 77 B8 18 C7 77 FF 1E C
006B2090 83 1D C7 77 3E E7 C7 77 08 00 00 00 63 79 C
006B20A0 D8 62 E7 77 2E 7F E7 77 8F 80 E6 77 DE 37 C
006B20B0 37 A8 E7 77 97 77 E7 77 D3 76 E7 77 B1 79 C

```

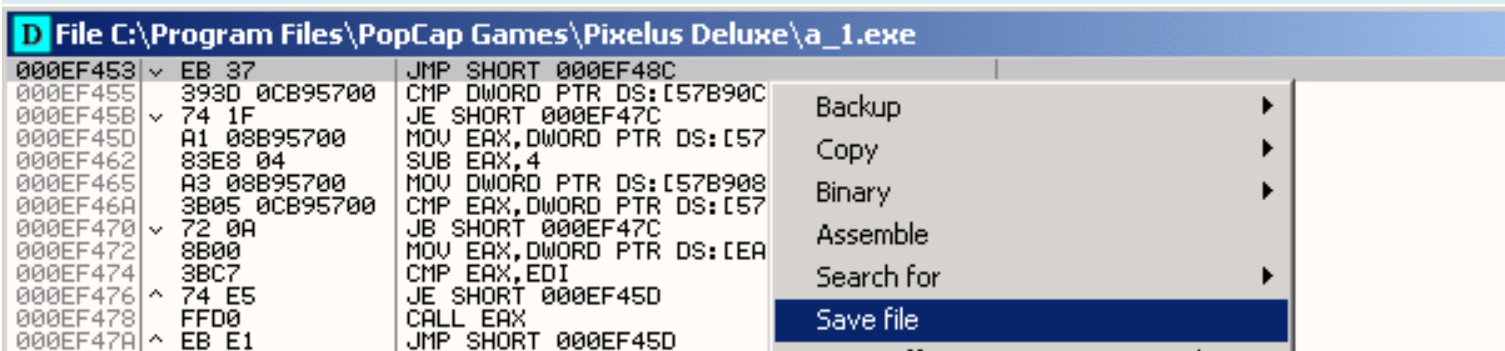
The list of memory addresses pane shows the following data:

```

004F007A RETURN
00000000
00000000
7FFD0000 ASCII "
0012FE70
0012FA94
0012FFB0 Pointer
004F2448 SE hanc
005293F0 a_1.005

```

A dialog box up. Select Copy All. A window up. Click your mouse to select Save File:



Named a_2.exe.

Test file a_2.exe. When you exit the program, we will not encounter the error message again. Them. Happy happy.

Greetingz: All members VCT, Condzero, and you.

tlandn

Manual unpack Armadillo v4.62_all protections

Author: kienmanowar

1. COPYMEM II Defeat

Olly configuration and load the target, set a BP WriteProcessMemory. Press F9 to run the program.

Observations in Olly:

_Break First:

0013D9D8	006E6312	CALL to WriteProcessMemory from thumb.006E630C
0013D9DC	0000007C	hProcess = 0000007C
0013D9E0	006F1B43	Address = 6F1B43
0013D9E4	0013DCC8	Buffer = 0013DCC8
0013D9E8	00000002	BytesToWrite = 2
0013D9EC	0013DCCC	pBytesWritten = 0013DCCC

_Follow At Buffer above we see here bytes will be written to address 0x6F1B43, first place will

infinite loop on EP's son process.

0013DCC8	0000FEEB
----------	----------

_Nhan F9 to run, you break a second time. This time we have the results as follows:

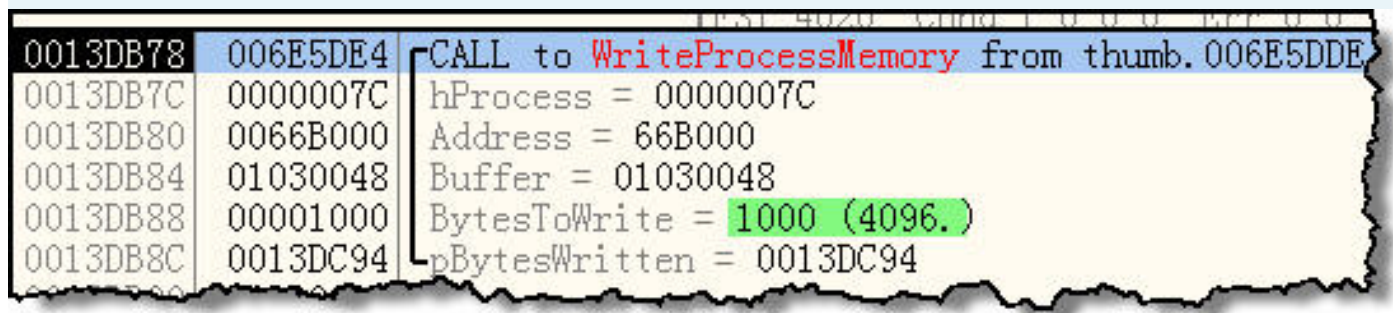
0013D9D8	006E633A	CALL to WriteProcessMemory from thumb.006E6334
0013D9DC	0000007C	hProcess = 0000007C
0013D9E0	006F1B43	Address = 6F1B43
0013D9E4	0071C384	Buffer = thumb.0071C384
0013D9E8	00000002	BytesToWrite = 2
0013D9EC	0013DCCC	pBytesWritten = 0013DCCC
0013D9F0	0013F229	

0071C384	00008B55
0071C388	00000000
0071C38C	00000000

_Nhu Results illustrate the image, this also is 2 bytes from the buffer area will be recorded in this time

but have slightly different times is to replace the infinite loop by 0x558B.

_Tiep To press F9 to run, we function at the break WriteProcessMemory:



_Chung We are stopped at the seat his father will Process 1000 bytes to process from the start address

0x66B000.Bay time we must find Magic Call, press **Ctrl + F9** and press **F8** we'll stop here (with

Ollyice ko need to press F8):

Address	Hex dump	Disassembly	Comment
006E5DDD	. 50	push eax	hProcess
006E5DDE	. FF15 1451710	call dword ptr [&KERNEL32.WriteProc	WriteProcessMemory
006E5DE4	. 85C0	test eax, eax	
006E5DE6	. 75 4B	jnz short 006E5E33	
006E5DE8	. 50	push eax	
006E5DE9	. F7D0	not eax	
006E5DEB	. 0FC8	bswap eax	
006E5DED	. 58	pop eax	
006E5DEE	. 73 00	jnb short 006E5DF0	
006E5DF0	. 9C	pushfd	
006E5DF1	. 60	pushad	
006E5DF2	. 8B 2B	mov ebx, [eax]	

_Tiep By pressing Ctrl + F9 to again, we will stop me in order RETN:

Address	Hex dump	Disassembly
006E5ED9	. 015F 5E	ADD DWORD PTR DS:[EDI+5E], EBX
006E5EDC	. 5B	POP EBX
006E5EDD	. 8BE5	MOV ESP, EBP
006E5EDF	. 5D	POP EBP
006E5EE0	. C3	RETN
006E5EE1	. 55	PUSH EBP
006E5EE2	. 8BEC	MOV ECX, [EBP]

_Nhan Alt + K to open the Call Stack window:

Address	Stack	Procedure / arguments	Called from	Frame
0013DCA0	006E4A52	thumb. 006E4D9A	thumb. 006E4A4D	0013DCD0

_Day Is Deencrypt Call can.Nhap we double-click Call this function to our code:

Address	Hex dump	Disassembly
006E4A48	. 51	PUSH ECX
006E4A49	. 8B55 08	MOV EDX, DWORD PTR SS:[EBP+8]
006E4A4C	. 52	PUSH EDX
006E4A4D	. E8 48030000	CALL thumb.006E4D9A
006E4A52	> 83C4 0C	ADD ESP, 0C
006E4A55	. 25 FF000000	AND EAX, 0FF
006E4A5A	. 85C0	TEST EAX, EAX
006E4A5C	. 75 07	JNZ SHORT thumb.006E4A65

_Nhan Enter to follow the Call this function, we will come here:

Address	Hex dump	Disassembly	Comment
006E4D9A	\$ 55	PUSH EBP	
006E4D9B	. 8BEC	MOV EBP, ESP	
006E4D9D	. 81EC 00010000	SUB ESP, 100	
006E4DA3	. 53	PUSH EBX	
006E4DA4	. 56	PUSH ESI	
006E4DA5	. 57	PUSH EDI	
006E4DA6	. 8B45 08	MOV EAX, DWORD PTR SS:[EBP+8]	
006E4DA9	. C1E0 0C	SHL EAX, 0C	
006E4DAC	. 8B0D 20C57100	MOV ECX, DWORD PTR DS:[71C520]	thumb.00401000
006E4DAF	. 8B48	MOV ECX, EAX	

_Tai Ago we observed the window Tip:



_Chung We have two function call in two different addresses, the first month is thăng and decrypt it

was known then as we observed that in some pictures illustrated above, only the second function is the

function that Encrypt we need to treat ly.Vay is we have an important information for the next step.

2nd Search OEP's Son process.

_Chung We have found Encrypt function in the previous section, next we need to find OEP's son

process. Hold the Olly not need to function in restart.Xoa bp WriteProcessMemory go, we put a BP in

WaitForDebugEvent function. Press F9 to run, we will stop here:

Address	Value	Comment
0013DCD8	006E218F	CALL to WaitForDebugEvent from thumb.006E2189
0013DCDC	0013ED70	pDebugEvent = 0013ED70
0013DCE0	000003E8	Timeout = 1000. ms
0013DCE4	7C910738	ntdll.7C910738
0013DCE8	00000000	

Follow In dump at 0x0013ED70:

Address	Value	Comment
0013ED70	00000001	
0013ED74	00000D00	
0013ED78	00000AAC	
0013ED7C	80000001	
0013ED80	00000000	
0013ED84	00000000	
0013ED88	0066B92C	thumb.0066B92C
0013ED8C	00000002	
0013ED90	00000008	
0013ED94	0066B92C	thumb.0066B92C
0013ED98	0066B92C	thumb.0066B92C
0013ED9C	00000001	
0013EDA0	85D1DC10	
0013EDA4	805BAED5	
0013EDA8	00000694	
0013EDAC	00000000	

Address	Hex dump	ASCII	
0013ED70	01 00 00 00 00 0D 00 00	
0013ED78	AC 0A 00 00 01 00 00 80	?.. r..€	
0013ED80	00 00 00 00 00 00 00 00	
0013ED88	2C B9 66 00 02 00 00 00	, 箴. 7...	
0013ED90	08 00 00 00 2C B9 66 00	□... , 箴.	
0013ED98	2C B9 66 00 01 00 00 00	, 箴. 7...	
0013EDA0	10 DC D1 85 D5 AE 5B 80	+三呎顛€	
0013EDA8	94 06 00 00 00 00 00 00	?.....	
0013EDB0	13 00 00 00 64 4D 00 A8	" ..	

_O Ago we observed that the value 0x0066B92C OEP is that we need to find. As in previous pictures

we remember where that will be recorded in 1000 bytes starting at 0x0066B000, so this will always

include all of our OEP in it.

_Ok Continue to work, we again collect some more important information is encrypt the call, OEP's

son process. The next section we need to do is detach the process from his father's no. De do this the

first we must set an infinite loop in OEP's son process. Ok, restart again Olly, BP set at

WriteProcessMemory, press F9 3 times we stopped in preparation for record 1000 bytes:

Address	Value	Comment
0013D878	006E5DE4	CALL to WriteProcessMemory from thumb.006E5DDE
0013D87C	0000007C	hProcess = 0000007C
0013D880	0066B000	Address = 66B000
0013D884	01050048	Buffer = 01050048
0013D888	00001000	BytesToWrite = 1000 (4096.)
0013D88C	0013DC94	pBytesWritten = 0013DC94
0013D890	0000000C	
0013D894	0000364F	

_Xoa Bp WriteProcessMemory go in, press Ctrl + G and enter the address of the function call is

Encrypt 0x006E4D22. Submit call this function J:

Address	Hex dump	Disassembly
006E4D22	90	NOP
006E4D23	90	NOP
006E4D24	90	NOP
006E4D25	90	NOP
006E4D26	90	NOP
006E4D27	. 83C4 0C	ADD ESP,0C

_Buoc Next infinite loop is placed in the month OEP con.Lam to how we do this, we have a formula

as follows:

OEP - BASE ZONE = Displacement

$66B92C - 66B000 = 92C$

BUFFER ZONE = + displacement of bytes of OEP

$1050048 + 92C = 1050974$

_Chuyen To dump window, press Ctrl + G and enter the address that we have calculated the above:

Address	Hex dump	ASCII
01050974	55 8B EC 83 C4 F4 53 B8	要 齋 齋
0105097C	D4 AE 66 00 E8 D3 B7 D9	援 f. 杓 焚
01050984	FF 8B 1D 54 1B 68 00 8B	齋 ?T+h.
0105098C	03 E8 4A AA DE FF 8B 03	鍋 齋 齋 ?
01050994	BA EC B9 66 00 E8 42 A6	紅 箴. 鋏
0105099C	DE FF 8B 03 83 C0 40 BA	?? 齋 @
010509A4	00 BA 66 00 E8 BF 84 D9	. 箴. 杓 勝

_Thay 2 bytes with 55 8B EB FE:

Address	Hex dump	ASCII
01050974	EB FE EC 83 C4 F4 53 B8	齋 報 齋 S
0105097C	D4 AE 66 00 E8 D3 B7 D9	援 f. 杓 焚
01050984	FF 8B 1D 54 1B 68 00 8B	齋 ?T+h.
0105098C	03 E8 4A AA DE FF 8B 03	鍋 齋 齋 ?
01050994	BA EC B9 66 00 E8 42 A6	紅 箴. 鋏

_Ok, Followed by BP at WaitForDebugEvent.Nhan set to run F9, Olly will break:

Address	Value	Comment
0013DCD8	006E218F	CALL to WaitForDebugEvent from thumb.006E2189
0013DCDC	0013ED70	pDebugEvent = 0013ED70
0013DCE0	000003E8	Timeout = 1000. ms
0013DC54	7C910738	ntdll.7C910738

_Follow In dump at 0x13ED70:

Address	Hex dump	ASCII
0013ED70	01 00 00 00 50 0C 00 00	...P...
0013ED78	1C 06 00 00 01 00 00 80	...r...€
0013ED80	00 00 00 00 00 00 00 00
0013ED88	2C B9 66 00 02 00 00 00	,箴.r...
0013ED90	08 00 00 00 2C B9 66 00	□...箴.
0013ED98	2C B9 66 00 01 00 00 00	,箴.r...
0013EDA0	C8 58 8B 85 D5 AE 5B 80	菟嬢債[€
0013EDA8	94 06 00 00 00 00 00 00	?.....
0013EDB0	13 00 00 00 64 4D 00 A9	!!...dM.

_Tai Here we are OEP's son process and the region marked yellow is very important. We're stopping

at WaitForDebugEvent function, until now we submitted the encrypt function, located at EB FE OEP's

son process. Press Ctrl + F9, F8 and then we will come here:

Address	Hex dump	Disassembly	Comment
006E217B	> 3361 68	XOR ESP,DWORD PTR DS:[ECX+68]	
006E217E	? E8 0300008B	CALL 8B6E2186	
006E2183	? 95	XCHG EAX,EBP	
006E2184	? D0F5	SAL CH,1	
006E2186	? FFFF	???	Unknown command
006E2188	. 52	PUSH EDX	pDebugEvent
006E2189	. FF15 E4507100	CALL NEAR DWORD PTR DS:[<&KERNEL32.WaitForDebugEvent>	WaitForDebugEvent
006E218F	. 85C0	TEST EAX,EAX	
006E2191	. 0F84 BB230000	JB thumb.006E4552	

_Bay Time we change E8 03 with 01 00:

Address	Hex dump	Disassembly	Comment
006E217B	3361 68	XOR ESP,DWORD PTR DS:[ECX+68]	
006E217E	0100	ADD DWORD PTR DS:[EAX],EAX	
006E2180	0000	ADD BYTE PTR DS:[EAX],AL	
006E2182	. 8B95 D0F5FFFF	MOV EDX,DWORD PTR SS:[EBP-A30]	
006E2188	. 52	PUSH EDX	pDebugEvent
006E2189	. FF15 E4507100	CALL NEAR DWORD PTR DS:[<&KERNEL32.WaitForDebugEvent>	WaitForDebugEvent
006E218F	. 85C0		

3. Create code to patch

_Chung We are stopping me in order TEST EAX, EAX. Space click here to assemble and replace me

in order JMP 00401000. Ok we have been as follows:

Address	Hex dump	Disassembly	Comment
006E2188	. 52	PUSH EDX	pDebugEvent
006E2189	. FF15 E4507100	CALL NEAR DWORD PTR DS:[<&KERNEL32.WaitForDebugEvent>	WaitForDebugEvent
006E218F	- E9 6CEED1FF	JMP thumb.00401000	
006E2194	90	NOP	
006E2195	90	NOP	
006E2196	90	NOP	

_Tiep As we press Ctrl + G and to address 0x401000. Here we will edit the following:

Address	Hex dump	Disassembly	Comment
00401000	90	NOP	
00401001	C705 70ED1300 01000	MOV DWORD PTR DS:[13ED70], 1	
00401008	C705 74ED1300 500C0	MOV DWORD PTR DS:[13ED74], 0C50	
00401015	C705 78ED1300 1C060	MOV DWORD PTR DS:[13ED78], 61C	
0040101F	C705 7CED1300 01000	MOV DWORD PTR DS:[13ED7C], 80000001	
00401029	C705 80ED1300 00000	MOV DWORD PTR DS:[13ED80], 0	
00401033	C705 84ED1300 00000	MOV DWORD PTR DS:[13ED84], 0	
0040103D	8105 88ED1300 00100	ADD DWORD PTR DS:[13ED88], 1000	
00401047	8105 94ED1300 00100	ADD DWORD PTR DS:[13ED94], 1000	
00401051	8105 98ED1300 00100	ADD DWORD PTR DS:[13ED98], 1000	
00401058	813D 98ED1300 00B06	CMP DWORD PTR DS:[13ED98], thumb.0066B000	
00401065	^ 74 D6	JE SHORT thumb.0040103D	
00401067	813D 98ED1300 00C06	CMP DWORD PTR DS:[13ED98], thumb.0066C000	
00401071	- 0F8C 1D112E00	JL thumb.006E2194	
00401077	90	NOP	
00401078	0000	ADD BYTE PTR DS:[EAX], AL	

_Ok, Prepared to run the program L but before we run to set a bp at 0x401077 and put in Bp

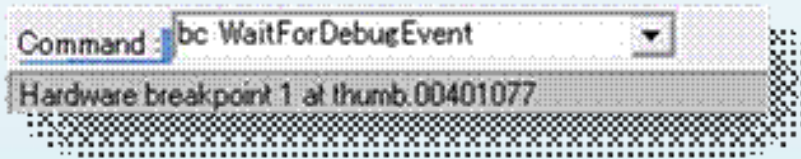
WaitForDebugEvent go. At the same time we also need to correct the value in 3 OEP address. We

revised as follows:

Address	Hex dump	ASCII	
0013ED70	02 00 00 00 50 0C 00 00	..P...	
0013ED78	0C 01 00 00 94 00 00 00	.r..?..	
0013ED80	00 E0 FD 7F 56 08 81 7C	. 贐 v	
0013ED88	00 00 40 00 02 00 00 00	..@.r...	
0013ED90	08 00 00 00 00 00 40 00@.	
0013ED98	00 00 40 00 01 00 00 00	..@.r...	
0013EDA0	C8 58 8B 85 D5 AE 5B 80	菟嬢贐[€	
0013EDA8	94 06 00 00 00 00 00 00	?.....	

_Bay Hours, the press F9 to run only passable, wait a few seconds we will break in the address that we

have set HE bp above:



Task Explorer _Mo up will see two process, the process is under process J child.

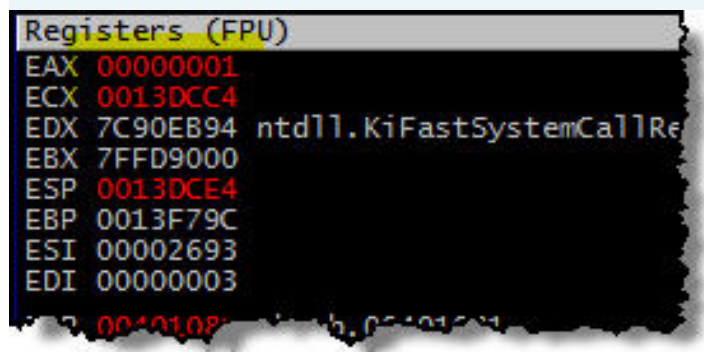
flyDBG	00000CD0	00400000	001EA000	C:\Rapidshare\Reverser_Tools\OllyDbg_Engines\fl...	FlyODbg - 32 ??????
thumb	00000EEC	00400000	0054B000	C:\Program Files\stg\thumb\thumb.exe	
thumb	00000204	00400000	0054B000	C:\Program Files\stg\thumb\thumb.exe	

Olly _Quay again, removing HE bp and 2 more on order follows:

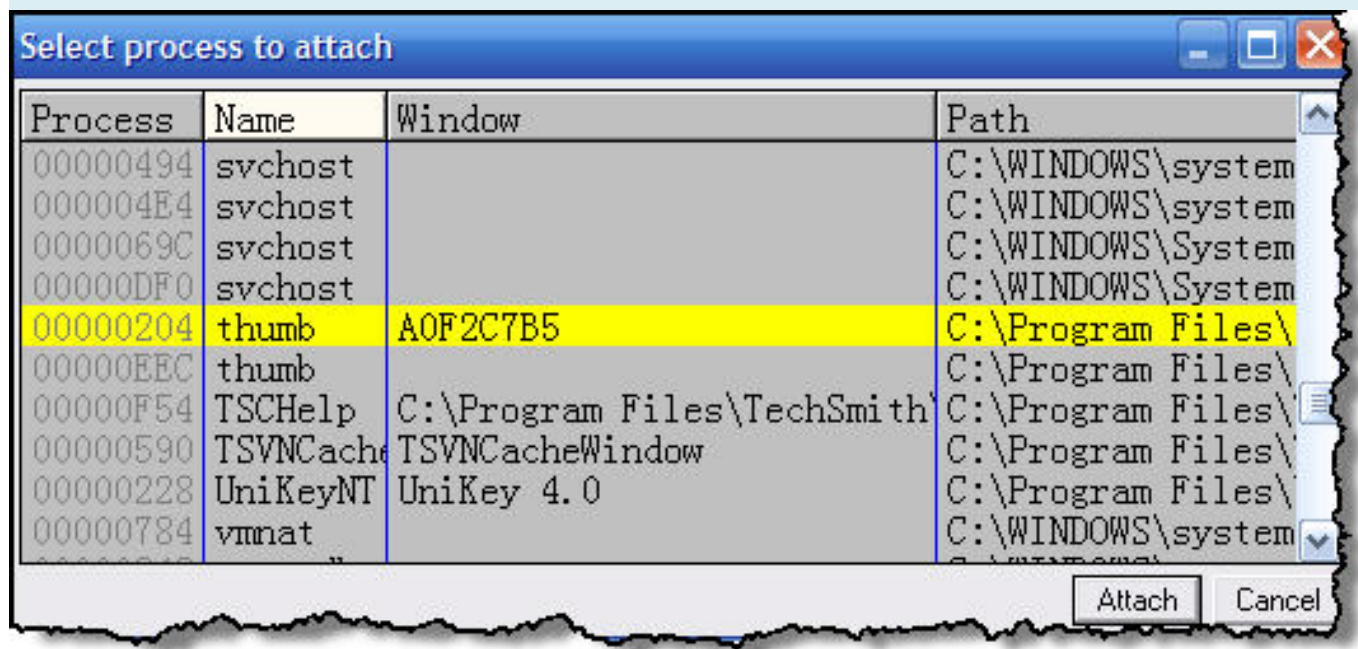
00401077	68 04020000	PUSH 204
0040107C	E8 1093457C	CALL kernel32.DebugActiveProcessStop
00401081	90	NOP

_Nhan F8 to trace the two commands, the last trace function API DebugActiveProcessStop we

recorded the results of EAX is 1. This means that we have made successful J:



There _Dung Olly current we are working now to open a new and Olly Attach child process to:



_Chung We will stop here, in a different modules:

Address	Hex dump	Disassembly
7C901231	C3	RETN
7C901232	8BFF	MOV EDI, EDI

Command :

Attached process paused at ntdll.DbgBreakPoint

_Nhan F9 to run and hit the F12 to pause the program again. We will be here in Olly, tolerable in

infinite loop:

Address	Hex dump	Disassembly	Comment
0066B92C	- EB FE	JMP SHORT thumb.0066B92C	
0066B92E	EC	IN AL, DX	
0066B92F	83C4 F4	ADD ESP, -0C	
0066B932	53	PUSH EBX	
0066B933	B8 D4AE6600	MOV EAX, thumb.0066AED4	
0066B934	FF 7F 00 00	JMP thumb.0066B934	

_Chung We stopped at the OEP, it's about values instead of the original EB FE as above:

Address	Hex dump	Disassembly	Com
0066B92C	55	PUSH EBP	
0066B92D	8BEC	MOV EBP, ESP	
0066B92F	83C4 F4	ADD ESP, -0C	
0066B932	53	PUSH EBX	

4th Search IAT

_Buoc Next we are going to find out IAT.De at the OEP we see a function call. Move the cursor to the

function and press Enter to Follow the functions that we will go to here:

address	hex dump	disassembly	comment
00407110	50	PUSH EAX	
00407111	6A 00	PUSH 0	
00407113	E8 F8FEFFFF	CALL thumb.00407010	
00407118	BA 10C16600	MOV EDX, thumb.0066C110	
0040711D	52	PUSH EDX	
0040711E	8905 DC246800	MOV DWORD PTR DS:[6824DC], EAX	
00407124	E9 68BE9B02	JMP 02DC2F94	
00407129	0FC9	BSWAP ECX	

_Lenh Dance are the highlight of the region's SPLICES also called CODE SPLICING.Nhan Alt + M

to open the Memory Map window, it starts from 0x2DC0000 and length is 20,000:

0173B000	00001000			stack of th	Priv	RW	Guar	RW
0173C000	00004000			stack of th	Priv	RW	Guar	RW
02DC0000	00020000				Priv	R E		RWE
10000000	00001000	sockspy		PE header	Imag	R		RWE
10001000	00009000	sockspy	.text	code	Imag	R		RWE
1000A000	00001000	sockspy	.rdata	imports, exp	Imag	R		RWE

_Tiep Continue to search IAT, we're in first Call function and the function we call this function to find

a different call. Follow the function call that we will see the following:

Address	Hex dump	Disassembly	Comment
0040700E	8BC0	MOV EAX, EAX	
00407010	- FF25 28073101	JMP NEAR DWORD PTR DS:[1310728]	
00407016	8BC0	MOV EAX, EAX	
00407018	- FF25 280B3101	JMP NEAR DWORD PTR DS:[1310B28]	kernel32.LocalAlloc
0040701E	8BC0	MOV EAX, EAX	
00407020	- FF25 D00A3101	JMP NEAR DWORD PTR DS:[1310AD0]	kernel32.TlsGetValue
00407026	8BC0	MOV EAX, EAX	
00407028	- FF25 C8073101	JMP NEAR DWORD PTR DS:[13107C8]	kernel32.TlsSetValue
0040702E	8BC0	MOV EAX, EAX	
00407030	- E9 13BF9B02	JMP 02DC2F48	
00407032	8BC0	MOV EAX, EAX	

_Day Is part of the IAT through indirect jump beyond the external memory area of exe files, this is

IAT Scramble format. We will find places to start end of IAT table, the final results we have are as

follows:

IAT Start: 01310048 7C80E9EC kernel32.FileTimeToSystemTime

IAT End: 01310B64 01137C70

Length: B20

_De Repair the IAT we must perform in child process when it stopped at the EP. To do this we use a

script by fly is *Armadillo V4.0-V4.4.Standard.Protection.osc*.

_De Do this, we re-open a further Olly and load the original exe files not Attach from the process are

currently co.Sau place at a BP WriteProcessMemory and press F9 to run:

Address	Value	Comment
0013D9D8	006E6312	CALL to WriteProcessMemory from thumb.006E630C
0013D9DC	0000007C	hProcess = 0000007C
0013D9E0	006F1B43	Address = 6F1B43
0013D9E4	0013DCC8	Buffer = 0013DCC8
0013D9E8	00000002	BytesToWrite = 2
0013D9EC	0013DCCC	pBytesWritten = 0013DCCC

_Follow At Buffer:

0013DCC8 **EB FE** 00 00 02 00 00 00 ep

_Tiep To press F9 to run and follow the buffer address:

0071C384 **55 8B** 00 00 00 00 00 00 U <.....

EB FE _Oh has been clear and instead of 2 bytes of the original dau.Vay we change back into EB FE:

0071C384 **EB FE** 00 00 00 00 00 00 ep

_Bay Hours we put in BP WriteProcessMemory go, then press F9 to run and open a window Memory

map, and set at a BP section. Text. First we will stop here:

Address	Hex dump	Disassembly	Comment
006DFD75	. 3D 02010000	CMP EAX, 102	
006DFD7A	. 75 02	JNZ SHORT thumb.006DFD7E	
006DFD7C	. EB D3	JMP SHORT thumb.006DFD51	
006DFD7E	> 68 FA000000	PUSH 0FA	[Timeout = 250. ms
006DFD83	. FF15 A4517100	CALL NEAR DWORD PTR DS:[<&KERNEL32.Sleep>]	[Sleep
006DFD89	> 8B45 DC	MOV EAX, DWORD PTR SS:[EBP-24]	
006DFD8C	. 50	PUSH EAX	[Arg1
006DFD8D	. E8 65030000	CALL thumb.006E00F7	[thumb.006E00F7

_Tiep To press F9 to run again and set a BP is the same as above, we will stop here in Olly:

Address	Hex dump	Disassembly	Comment
006E218F	85C0	TEST EAX, EAX	
006E2191	0F84 BB230000	JE thumb.006E4552	
006E2197	8B85 F0FDFFFF	MOV EAX, DWORD PTR SS:[EBP-210]	
006E219D	25 FF000000	AND EAX, 0FF	
006E21A2	85C0	TEST EAX, EAX	
006E21A4	74 13	JE SHORT thumb.006E21B9	

_Ta Find PID of child process, do not ask to be confusion:

Select process to attach			
Process	Name	Window	Path
00000DF0	svchost		C:\WINDOWS\System
00000204	thumb	A0F2C7B5	C:\Program Files\
00000CA0	thumb		C:\Program Files\
00000CB8	thumb		C:\Program Files\
00000EEC	thumb		C:\Program Files\
00000F54	TSCHelp	C:\Program Files\TechSmith\	C:\Program Files\
00000590	TSVNCache	TSVNCacheWindow	C:\Program Files\
00000228	UniKeyNT	UniKey 4.0	C:\Program Files\
00000784	vmnat		C:\WINDOWS\system
00000348	vmnetdhcp		C:\WINDOWS\system

_Ok We have 0CB8 PID is, in our Olly edit as follows:

Address	Hex dump	Disassembly	Comment
006E218F	68 B80C0000	PUSH 0CB8	
006E2194	E8 F881177C	CALL kernel32.DebugActiveProcessStop	
006E2199	90	NOP	

_Nhan F8 to trace the two commands and remember to observe the results of EAX, if it returns the

value 1 is ok, we have done right then, but it does not do it again ặc ặc L.

_Tiep To a more open and more Olly Attach this new child process vao.Nhan F9 to run and press F12

to pause the program. I will here in Olly:

Address	Hex dump	Disassembly	Comment
006F1B43	5- EB FE	JMP SHORT thumb.<ModuleEntryPoint>	
006F1B45	? EC	IN AL, DX	
006F1B46	. 6A FF	PUSH -1	
006F1B48	. 68 00B07100	PUSH thumb.0071B000	
006F1B4D	. 68 80186F00	PUSH thumb.006F1880	SE handler installation

_Thay The EB FE 8B to 55, then Plugins ODBScript used to run the script that I have said above.

Results last one is as follows:

Address	Hex dump	Disassembly	Comment
0066B92C	A6	CMPS BYTE PTR DS:[ESI], BYTE PTR ES:[EDI]	This is the OEP! Found By: fly
0066B92D	15 F47B376A	ADC EAX, EAX	
0066B932	4B	DEC EB	
0066B933	40	INC EA	
0066B934	27	DAA	
0066B935	307E F8	XOR BY	
0066B938	1B4D AF	SBB EC	
0066B93B	210C15 05ACE8F6	AND DW	
0066B942	1873 F0	SBB BY	
0066B945	76 52	JBE SH	
0066B947	52	PUSH E	
0066B948	2D 6193FB49	SUB EA	

_Chung We stopped at OEP's Child process, though the code has been encrypt but not important. tasks

we are looking for IAT.

_De Find exactly IAT, we open the window Memory Map, and to the top. Then press Ctrl + B to

perform the search function 7C80E9EC kernel32.FileTimeToSystemTime because this is the start of

the function IAT. (Remember to find the address against the EC E9 80 7C. Results first is not correct,

press Next Ctrl + L to find and we are as follows:

01350048	7C80E9EC	kernel32.FileTimeToSystemTime
0135004C	77D6ECF2	USER32.OemToCharA
01350050	7C8114AB	kernel32.GetVersion
01350054	77D51DE0	USER32.SetWindowRgn
01350058	7C80B529	kernel32.GetModuleHandleA
0135005C	77D4E2AE	USER32.SendMessageA
01350060	77D4C4AE	USER32.GetForegroundWindow
01350064	77F22D36	GDI32.CloseEnhMetaFile
01350068	7C839166	kernel32.GlobalSize
0135006C	77C01A50	VERSION.GetFileVersionInfoA
01350070	7C801A24	kernel32.CreateFileA
01350074	77DDEBE7	ADVAPI32.RegSetValueExA
01350078	77D4C4D4	USER32.EnableWindow
0135007C	77D97BAD	USER32.EnableScrollBar
01350080	77F266EA	GDI32.SetEnhMetaFileBits
01350084	77D5760B	USER32.IsCharAlphaA
01350088	77F49AA9	GDI32.GetEnhMetaFileBits

_Dem Results compare with what we have been on, we concluded this is IAT Table for our ta.Thuc

Copy from the Binary table in the region to which we need to fix IAT table above, we have been as

follows:

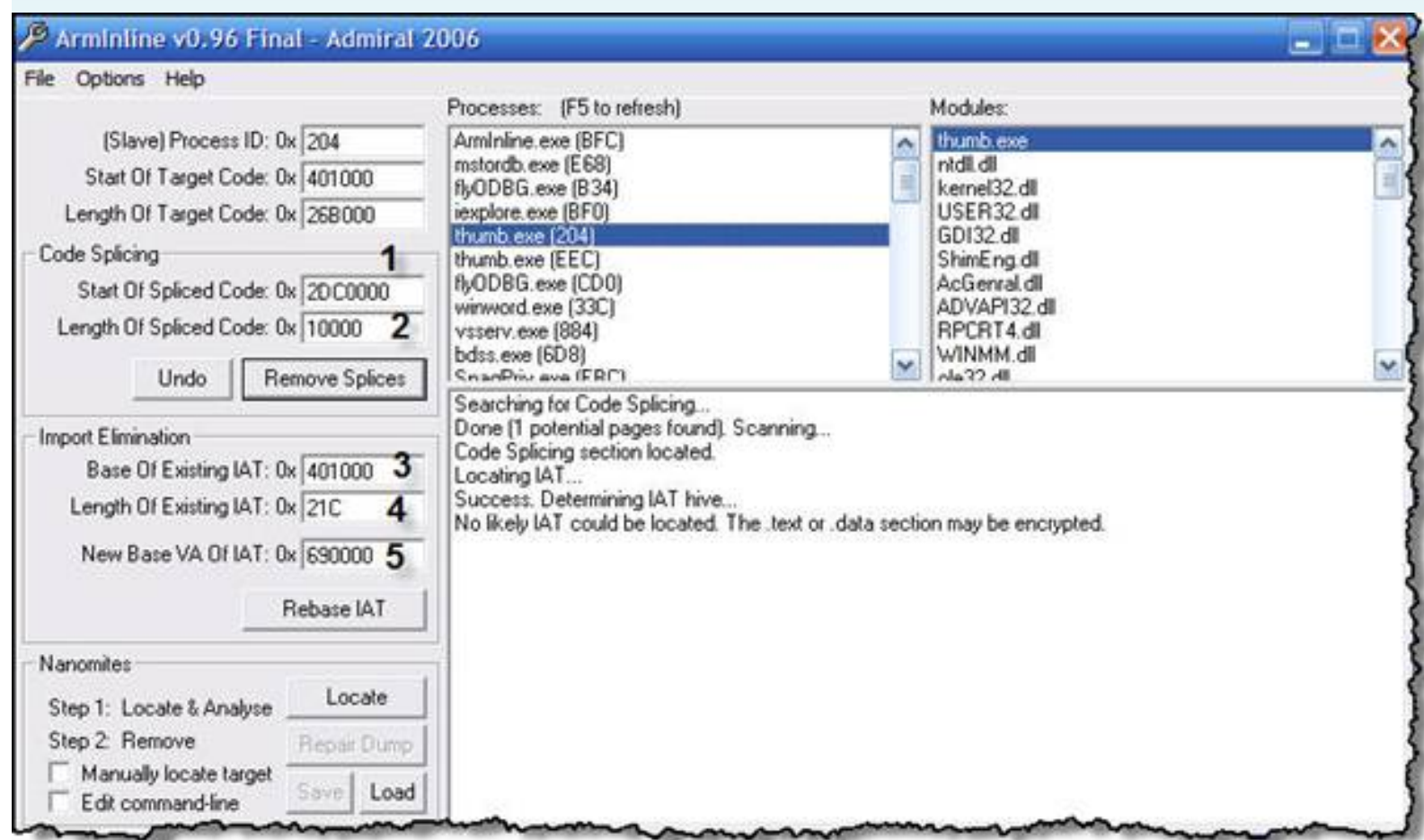
Address	Value	Comment
01310048	7C80E9EC	kernel32.FileTimeToSystemTime
0131004C	77D6ECF2	USER32.OemToCharA
01310050	7C8114AB	kernel32.GetVersion
01310054	77D51DE0	USER32.SetWindowRgn
01310058	7C80B529	kernel32.GetModuleHandleA
0131005C	77D4E2AE	USER32.SendMessageA
01310060	77D4C4AE	USER32.GetForegroundWindow
01310064	77F22D36	GDI32.CloseEnhMetaFile
01310068	7C839166	kernel32.GlobalSize
0131006C	77C01A50	VERSION.GetFileVersionInfoA
01310070	7C801A24	kernel32.CreateFileA
01310074	77DDEBE7	ADVAPI32.RegSetValueExA
01310078	77D4C4D4	USER32.EnableWindow
0131007C	77D97BAD	USER32.EnableScrollBar
01310080	77F266EA	GDI32.SetEnhMetaFileBits
01310084	77D5760B	USER32.IsCharAlphaA
01310088	77F49AA8	GDI32.GetEnhMetaFileDescriptionA

_Vay The IAT is completed, we can close the new open Olly was then (remember retain Olly 2 we

make the first).

_Viec Need to do next is to remove Splicing Code, to the region IAT. To do this we use a tool that is

famous ArmInline. Select the child process in the list of programs that process found



1. The address where the start of the region SPLICES.

2. As the length of the regional Code Splicing.

3. As the starting area of child IAT process.

4. The length of the IAT.

5. Is where we decided to set the IAT.

_Chung We will find areas to set IAT Table, open the memory map and select the following section:

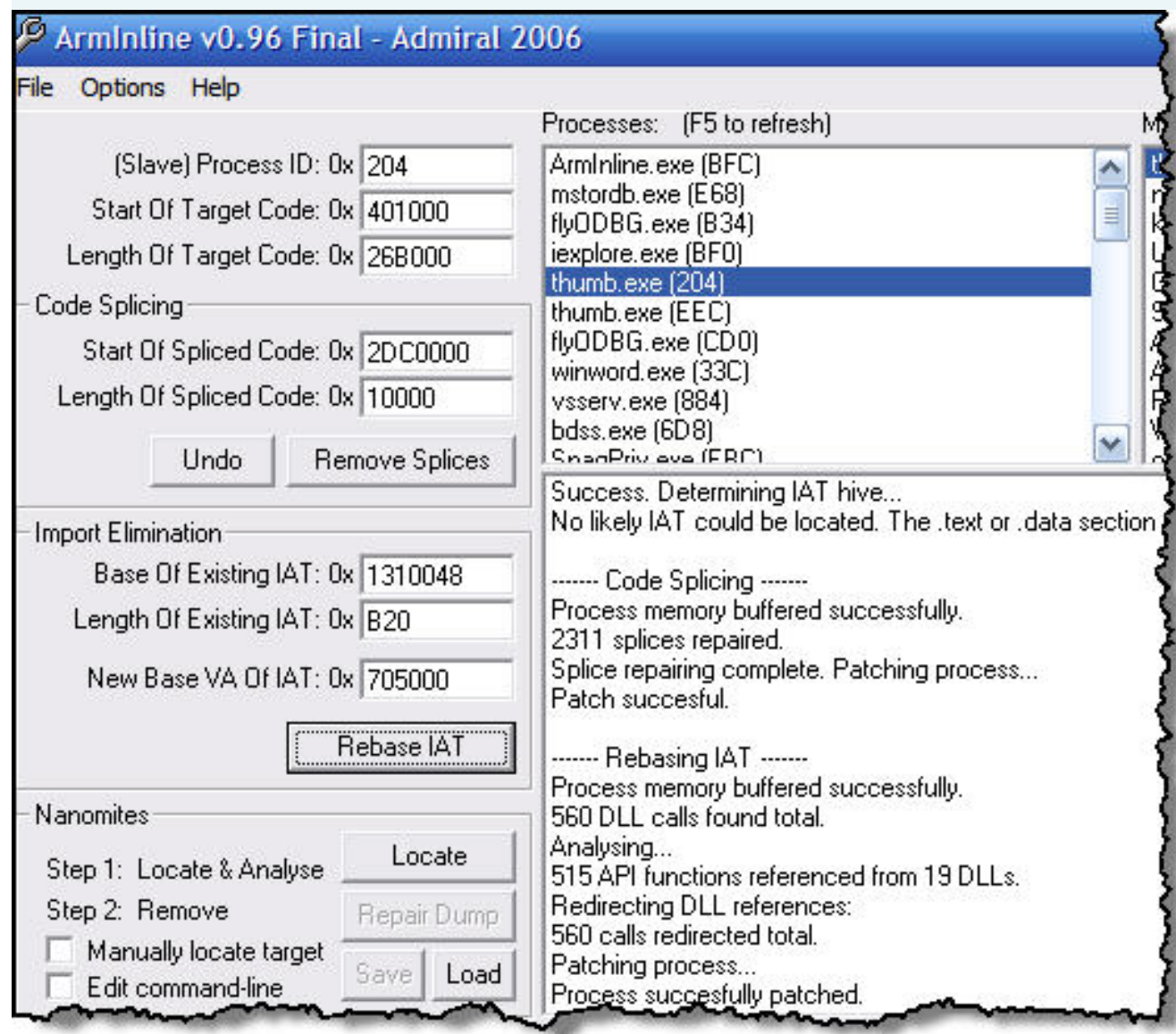
006B5000	00050000	thumb	.text	code	Imag	R	RWE
00705000	00010000	thumb	.adata		Imag	R	RWE
00715000	00010000	thumb	.data	data, imports	Imag	R	RWE

_Day An empty section and the length of 10,000, is very suitable place for us to set IAT.

_Truoc We remove the code splicing:

```
----- Code Splicing -----
Process memory buffered successfully.
2311 splices repaired.
Splice repairing complete. Patching process...
Patch succesful.
```


Rebase _Tiep by the IAT:

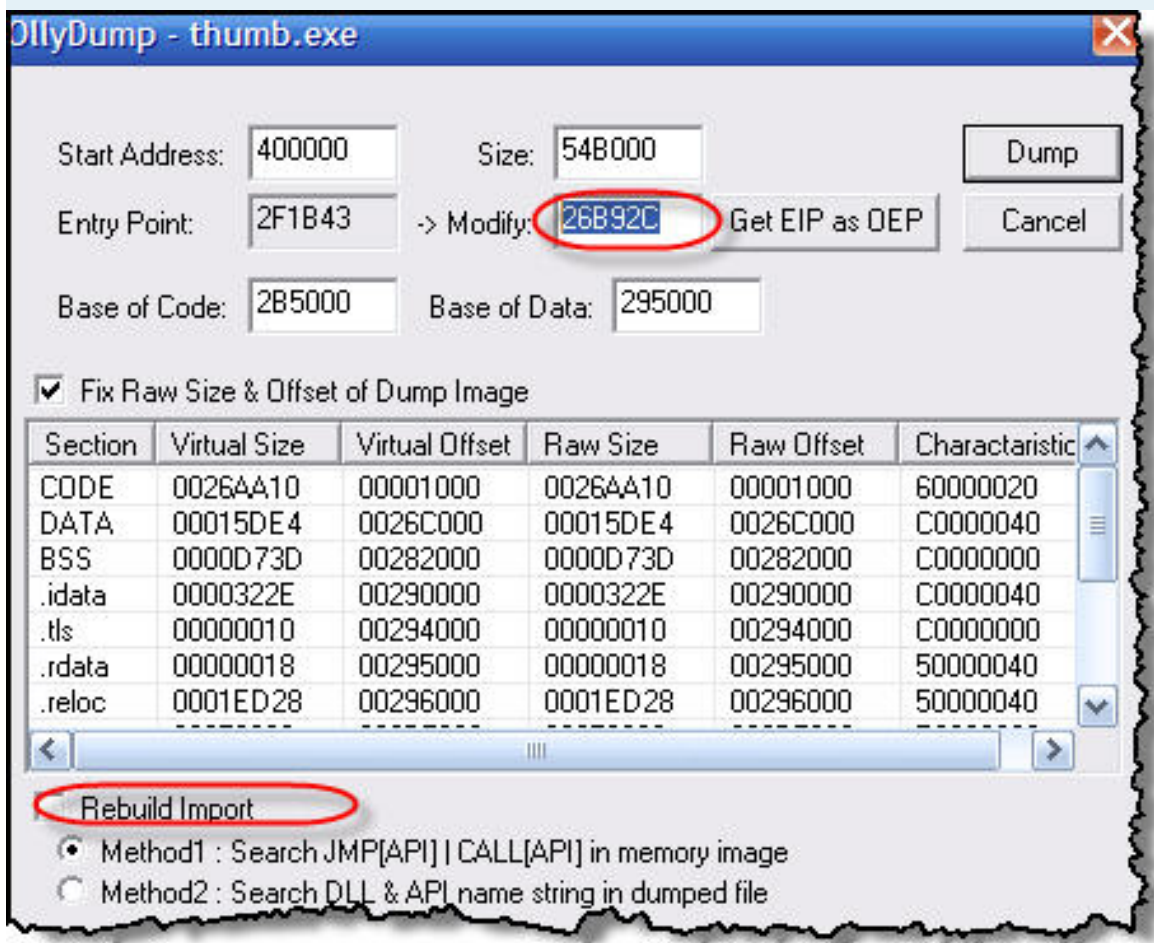


_Kiem To see the correct one was found as follows:

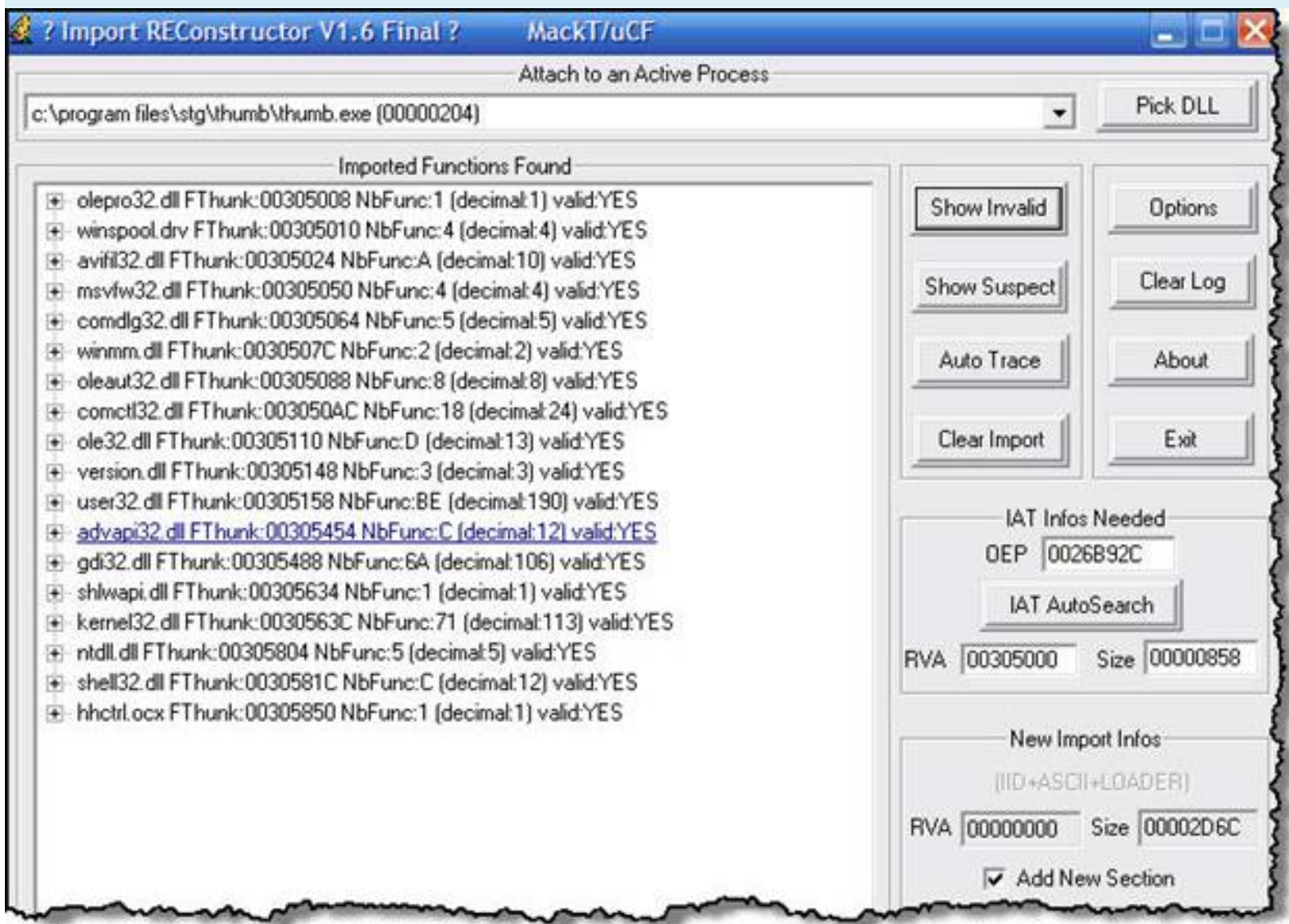
00407110	50	PUSH EAX	
00407111	6A 00	PUSH 0	
00407113	E8 F8FEFFFF	CALL thumb.00407010	JMP to kernel32.GetModuleHandleA
00407118	BA 10C16600	MOV EDX, thumb.0066C110	
0040711D	52	PUSH EDX	
0040711E	8905 DC246800	MOV DWORD PTR DS:[6824DC], EAX	

_Hoan All accurate! Next conduct dump file, but before you dump it will copy the PE Headers for

child process. Then use the dump to dump Olly target:

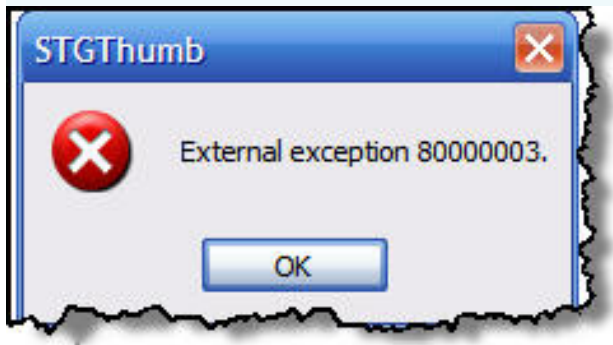


_Tien To rebuild the IAT, remember to select the right child in the process list processes:



_Cut Thunks and fix the IAT has dump.Chay file file has to fix IAT test we received the following

error message:



Day Are signs of Nanomites. Olly Open a new file dumped.exe load. Then in the process of

selecting ArmInline dumped_.exe. Click locate and select the original file.



Repair _Cui select the dump:



_Test Try to fix the file has Nanomites oh works fine!

_Phu Has finally done! J

hacnho's tutorials # 16

Guide to unpack ASProtect 1:23 RC 4

(Support ver ASProtect 1:23 RC4 - 1.3.08.24, RC4 Registered ASProtect 1:23)

I. Introduction:

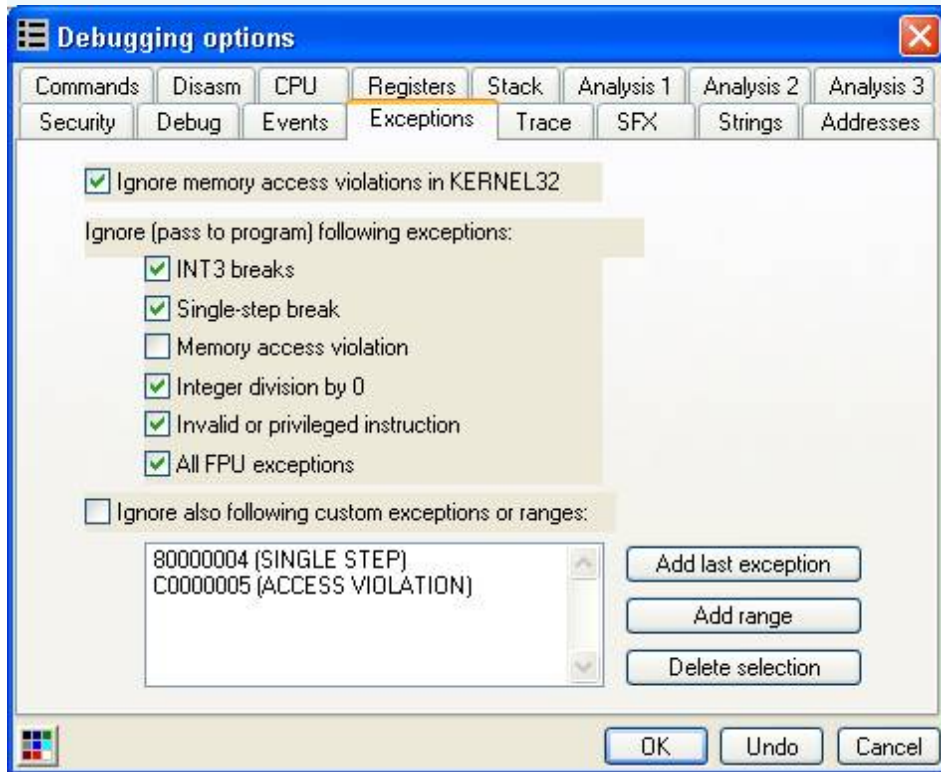
There are many on the NET tut and methods to unpack two ver this ASProtect. But reading to read the chả a tut do as I pleased both. Now write this as I tut is understandable (for bags) best.

II. Tools:

1:10 Final OllyDBG with OllyDump, Hide 1.2 debugger, Command bar 3.1, 0.92 PeiD, Import REConstructor 1.6 plugin with Final ASPR_1.23-ImpREC_Plugin.dll

III. Objective: DVDFab v2.0 (Case 45 bytes).

-Step 1: Edit Option Exception by Olly as follows:



-Step 2: Load up DVDFab.exe Olly.

Address	Hex dump	Disassembly	Comment	Registers (FPU)
00401000	68 01004C00	PUSH DUDFab.004C6001		EAX 00000000
00401005	E8 01000000	JMP DUDFab.00401008		ECX 0012FFB0
0040100A	C3	RETN		EDX 7FFDF000
0040100B	C3	RETN		EBX 7FFDF000
0040100C	55	LOOPNE SHORT DUDFab.00400FB1		ESP 0012FFC4
0040100E	7F F4	PUSH EBP		EBP 0012FFB0
00401011	E8 069892D6	JMP SHORT DUDFab.00401005		ESI 00000000
00401016	3E CA 2870	RETJ D602A81C		EDI 400508B3
0040101A	98	RETJ 7020	Far return	EIP 00401000 DUDFab.<ModuleEntryF
0040101B	34 C7	CNOE		C 0 ES 0023 32bit 0(FFFFFFFF)
0040101D	0C 3D	XOR AL, 3D		P 1 CS 0018 32bit 0(FFFFFFFF)
0040101F	27	DAA		A 0 SS 0023 32bit 0(FFFFFFFF)
00401020	AD	LODS DWORD PTR DS:[ESI]		Z 0 DS 0023 32bit 0(FFFFFFFF)
00401021	F3	PREFIX REP:	Superfluous prefix	S 1 FS 0038 32bit 7FFDE000(FFF)
00401022	9CE7	MOV DI, FS		T 0 GS 0000 NULL
00401024	98	WAIT		D 0
00401025	4B	INC EBP		O 0 LastErr ERROR_SUCCESS (00000000)
00401026	E4 70	IN AL, 70	I/O command	EFL 00000286 (NO,NB,NE,A,S,PE,L,L)
00401028	1C E6	SBB AL, 0E6		ST0 empty +UNORM 2000 00560000 00
0040102A	53	PUSH EBX		ST1 empty +UNORM 2402 00120000 40
0040102B	C4A6 715917C1	LES ESP, FWORD PTR DS:[ESI+C1175971]	Modification of segment register	ST2 empty +UNORM 17CD 77F516F5 FF
00401031	0E	POP CS		ST3 empty 0,00000000 00000000 00
00401032	CF	IRETD		ST4 empty +UNORM CEE6 0012CF14 00
00401033	3F	RAS		ST5 empty +UNORM 0006 000001FC 00
00401034	72 3E	JB SHORT DUDFab.00401074		ST6 empty 1.00000000000000000000
00401036	E1 4A	LOOPNE SHORT DUDFab.00401082		ST7 empty 1.00000000000000000000
004C6001=DUDFab.004C6001				FST 4020 Cond 1 0 0 0 Err 0 0 1
				FCW 027F Prec NEAR, 53 Mask 1

Address	Hex dump	ASCII	Address	Value	Comment
004C6000	90 60 E3 03 00 00 00 E9 EB 04 5D 45 55 C3 E8 01	...%...0\$.IEU+.	0012FFC4	77E95A18	RETURN to kernel32.77E95A18
004C6010	00 00 00 00 50 00 00 FF FF FF 03 00 01 E8 00 60	...\$m0...l0s.	0012FFC8	400508B3	
004C6020	0C 00 00 00 40 01 75 0C 88 74 24 28 83 FE 01 89	..C)M.u.it3(a.e	0012FFCC	00000000	
004C6030	50 4E 75 31 80 45 53 50 53 FF 85 05 09 00 00 80	3NullESP\$.f.r...l	0012FFD0	7FFDF000	
004C6040	45 35 50 E9 82 00 00 00 00 00 00 00 00 00 00	ESP00.....	0012FFD4	F3F4ACF4	
004C6050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012FFD8	0012FFC8	
004C6060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	0012FFDC	80536A0E	
004C6070	85 C0 74 1C E8 01 E8 81 FB F8 C0 A5 23 74 35 33	3it\$.0u0\$4tt53	0012FFE0	FFFFFFFF	End of SEH chain
004C6080	02 56 6A 00 56 FF 75 4E FF D0 5E 83 FE 00 75 24	UJ.U.uN.4\$u\$	0012FFE4	77E98806	SE handler
004C6090	33 D2 88 45 41 85 C0 74 07 52 52 FF 75 35 FF D0	3IEA0\$tr.RR.uS.	0012FFE8	77E95A18	kernel32.77E95A18
004C60A0	88 45 35 05 C0 74 00 60 00 00 00 00 00 00 00	IE5\$tr.h.C..j.u	0012FFEC	00000000	
004C60B0	33 0F FF 00 58 00 00 61 75 06 6A 01 58 C3 0C 00	S.U=I.0aw.j.Xr..	0012FFF0	00000000	
004C60C0	E3 03 71 30 1B C0 40 C2 0C 00 00 00 00 00 00	3=7.0w.j.0...	0012FFF4	00000000	
004C60D0	33 65 60 C3 61 86 47 74 90 12 00 00 00 00 00	0jk=a3Gt\$.r00^?	0012FFF8	00401000	DUDFab.<ModuleEntryPoint>
004C60E0	33 65 60 C3 61 86 47 74 90 12 00 00 00 00 00	UjI\$....767\$.T.	0012FFFC	00000000	

Command

Program entrypoint Paused

-Step 3: Press Shift + F9 (by pass exeception) 26 times, you will go to last exception. Click again to run the program completely.

file:///C:/RCE%20Unpacking%20eBook%20[Tr...SProtect%201.23%20RC%204_by%20hacnho.htm (3 of 26) [1/9/2009 9:44:51 LithiumLi]

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00010000	00001000				Priv	RW	RW	
00020000	00001000				Priv	RW	RW	
00012C000	00001000				Priv	RW	Gua: RW	
00012D000	00003000			stack of ma	Priv	RW	Gua: RW	
000130000	00001000				Map	R	R	
000140000	00008000				Priv	RW	RW	
000240000	00006000				Priv	RW	RW	
000250000	00001000				Map	RW	RW	
000260000	00016000				Map	R	R	
000280000	00034000				Map	R	R	
0002C0000	00041000				Map	R	R	
000310000	00006000				Map	R	R	
000320000	00001000				Priv	RWE	RWE	
000330000	00005000				Priv	RW	RW	
000340000	00003000				Map	R	R	
000350000	00001000				Priv	RW	RW	
000360000	00001000				Priv	RW	RW	
000370000	00001000				Priv	RW	RW	
0003E0000	00002000				Map	R	R	
000400000	00001000	DVDFab		PE header	Imag	R	RWE	
000401000	00058000	DVDFab		code	Imag	R	RWE	
000450000	00008000	DVDFab		exports				
000464000	0001C000	DVDFab						
000480000	00046000	DVDFab	.rsrc	resources				
0004C0000	00016000	DVDFab	.data	data, impo				
0004DC000	00001000	DVDFab	.adata					
0004E0000	00005000							
0005A0000	00002000							
0005B0000	000103000							
0006C0000	0000A3000							
0009D0000	00002000							
000A00000	0001E000							
000A20000	0000C000							
000B20000	00051000							
5A070000	00001000	uxtheme		PE header				
5A071000	0002C000	uxtheme	.text	code, impo				
5A090000	00001000	uxtheme	.data	data				
5A09E000	00004000	uxtheme	.rsrc	resources				
5A0A2000	00002000	uxtheme	.reloc	relocatio				
71950000	00001000	comctl		PE header				
71951000	00038000	comctl	.text	code, impo				
719D0000	00001000	comctl	.data	data				
719DA000	00054000	comctl	.rsrc	resources				
71A2E000	00006000	comctl	.reloc	relocatio				
73D00000	00001000	mfc42		PE header				
73D01000	0009A000	mfc42	.text	code				
73E6B000	00035000	mfc42	.rdata	imports, e				
73EA0000	00008000	mfc42	.data	data				
73EA8000	0000B000	mfc42	.rsrc	resources	Imag	R	RWE	

Actualize
View in Disassembler Enter
Dump in CPU
Dump
Search Ctrl+B
Set break-on-access F2
Set memory breakpoint on access
Set memory breakpoint on write
Set access
Copy to clipboard
Sort by
Appearance

-Step 6: Press Ctrl + F11 to trace over. Wait Olly trace completed, you will come:

Address	Hex dump	Disassembly	Comment
00454F5C	FF15 64974500	CALL NEAR DWORD PTR DS:[459764]	MSUCRT.__set_app_type
00454F62	59	POP ECX	
00454F63	830D 84FD4700	OR DWORD PTR DS:[47FD84], FFFFFFFF	
00454F6A	830D 88FD4700	OR DWORD PTR DS:[47FD88], FFFFFFFF	
00454F71	FF15 68974500	CALL NEAR DWORD PTR DS:[459768]	MSUCRT.__p_fnode
00454F77	8B0D 04F24700	MOV ECX, DWORD PTR DS:[47F2D4]	
00454F7D	8908	MOV DWORD PTR DS:[EAX], ECX	
00454F7F	FF15 6C974500	CALL NEAR DWORD PTR DS:[45976C]	MSUCRT.__p_commode
00454F85	8B0D 00F24700	MOV ECX, DWORD PTR DS:[47F2D0]	
00454F8B	8908	MOV DWORD PTR DS:[EAX], ECX	
00454F8D	A1 70974500	MOV EAX, DWORD PTR DS:[459770]	
00454F92	8B0D	MOV EAX, DWORD PTR DS:[EAX]	
00454F94	A3 80FD4700	MOV DWORD PTR DS:[47FD80], EAX	
00454F99	E8 0B040000	CALL DVDFab.004553A9	
00454F9E	391D 00A14600	CMP DWORD PTR DS:[46A1B0], EBX	
00454FA4	75 0C	JNZ SHORT DVDFab.00454FB2	
00454FA6	68 A6534500	PUSH DVDFab.004553A6	
00454FAB	FF15 74974500	CALL NEAR DWORD PTR DS:[459774]	MSUCRT.__setusermatherr
00454FB1	59	POP ECX	
00454FB2	E8 D7030000	CALL DVDFab.0045538E	
00454FB7	68 60404600	PUSH DVDFab.00464060	
00454FBC	68 5C404600	PUSH DVDFab.0046405C	
00454FC1	E8 C2030000	CALL DVDFab.00455388	JMP to MSUCRT._initterm
00454FC6	A1 CCF24700	MOV EAX, DWORD PTR DS:[47F2CC]	
00454FCB	8945 94	MOV DWORD PTR SS:[EBP-6C], EAX	
00454FCE	8D45 94	LEA EAX, DWORD PTR SS:[EBP-6C]	
00454FD1	50	PUSH EAX	
00454FD3	50	PUSH EAX	

Step-7: scroll mouse over some line, you'll see the code form 00.

Address	Hex dump	Disassembly	Comment
00454F2E	C3	RETN	
00454F2F	0000	ADD BYTE PTR DS:[EAX], AL	
00454F31	0000	ADD BYTE PTR DS:[EAX], AL	
00454F33	0000	ADD BYTE PTR DS:[EAX], AL	
00454F35	0000	ADD BYTE PTR DS:[EAX], AL	
00454F37	0000	ADD BYTE PTR DS:[EAX], AL	
00454F39	0000	ADD BYTE PTR DS:[EAX], AL	
00454F3B	0000	ADD BYTE PTR DS:[EAX], AL	
00454F3D	0000	ADD BYTE PTR DS:[EAX], AL	
00454F3F	0000	ADD BYTE PTR DS:[EAX], AL	
00454F41	0000	ADD BYTE PTR DS:[EAX], AL	
00454F43	0000	ADD BYTE PTR DS:[EAX], AL	
00454F45	0000	ADD BYTE PTR DS:[EAX], AL	
00454F47	0000	ADD BYTE PTR DS:[EAX], AL	
00454F49	0000	ADD BYTE PTR DS:[EAX], AL	
00454F4B	0000	ADD BYTE PTR DS:[EAX], AL	
00454F4D	0000	ADD BYTE PTR DS:[EAX], AL	
00454F4F	0000	ADD BYTE PTR DS:[EAX], AL	
00454F51	0000	ADD BYTE PTR DS:[EAX], AL	
00454F53	0000	ADD BYTE PTR DS:[EAX], AL	
00454F55	0000	ADD BYTE PTR DS:[EAX], AL	
00454F57	0000	ADD BYTE PTR DS:[EAX], AL	
00454F59	0000	ADD BYTE PTR DS:[EAX], AL	
00454F5B	00FF	ADD BH, BH	
00454F5D	15 64974500	ADC ECX, DWORD PTR DS:[459764]	
00454F62	59	POP ECX	
00454F63	830D 84FD4700	OR DWORD PTR DS:[47FD84], FFFFFFFF	

-Step 8: press Ctrl + A to re-analyze:

Address	Hex dump	Disassembly	Comment
00454F2E	C3	RETN	
00454F2F	00	DB 00	
00454F30	00	DB 00	
00454F31	00	DB 00	
00454F32	00	DB 00	
00454F33	00	DB 00	
00454F34	00	DB 00	
00454F35	00	DB 00	
00454F36	00	DB 00	
00454F37	00	DB 00	
00454F38	00	DB 00	
00454F39	00	DB 00	
00454F3A	00	DB 00	
00454F3B	00	DB 00	
00454F3C	00	DB 00	
00454F3D	00	DB 00	
00454F3E	00	DB 00	
00454F3F	00	DB 00	
00454F40	00	DB 00	
00454F41	00	DB 00	
00454F42	00	DB 00	
00454F43	00	DB 00	
00454F44	00	DB 00	
00454F45	00	DB 00	
00454F46	00	DB 00	
00454F47	00	DB 00	
00454F48	00	DB 00	
00454F49	00	DB 00	
00454F4A	00	DB 00	
00454F4B	00	DB 00	
00454F4C	00	DB 00	
00454F4D	00	DB 00	
00454F4E	00	DB 00	
00454F4F	00	DB 00	
00454F50	00	DB 00	
00454F51	00	DB 00	
00454F52	00	DB 00	
00454F53	00	DB 00	
00454F54	00	DB 00	
00454F55	00	DB 00	
00454F56	00	DB 00	
00454F57	00	DB 00	
00454F58	00	DB 00	
00454F59	00	DB 00	
00454F5A	00	DB 00	
00454F5B	00	DB 00	
00454F5C	00	DB 00	
00454F5D	15 64974500	CALL NEAR DWORD PTR DS:[459764]	MSUCRT.____set_app_type

-Step 9: count is 45 bytes. So this case is 45 bytes. Now the task is, we fix the bytes.

Temporary OllyDBG to open again. We will open a window onto other Olly. Press Shift + F9 26 times, then set breakpoint at RETN, press Alt + M, located in memory breakpoint code section.

The most important step for both the push of this case is 45 bytes, we must set conditions balanced stack **ebp** = **esp**. OK, now press Alt + F1 and enter the **TC ebp == esp** or press Ctrl + T and set conditions as follows:

Condition to pause run trace

Pause run trace when any checked condition is met:

☐ EIP is in range 00000000 ... 00000000

☐ EIP is outside the range 00000000 ... 00000000

☒ Condition is TRUE ebp == esp

☐ Command is suspicious or possibly invalid

☐ Command count is 0. (actual 0.)

☐ Command is one of

In command, R8, R32, RA, RB and CONST match any register or constant

-Buoc10: Then press Ctrl + F11 to trace over. You will stop at places where EBP = ESP.

The screenshot shows the Immunity Debugger interface. The assembly window displays the following instructions:

Address	Hex dump	Disassembly	Comment
00A25B03	53	PUSH EBX	
00A25B04	56	PUSH ESI	
00A25B05	8B75 0C	MOV ESI, DWORD PTR SS:[EBP+C]	
00A25B08	8B5D 08	MOV EBX, DWORD PTR SS:[EBP+8]	
00A25B0B	EB 11	JMP SHORT 00A25BEE	
00A25B0D	0FB703	MOVZX EAX, WORD PTR DS:[EBX]	
00A25B0E	03C6	ADD EAX, ESI	
00A25B12	83C3 02	ADD EBX, 2	
00A25B15	8BD0	MOV EDX, EAX	
00A25B18	8BC6	MOV EAX, ESI	
00A25B1B	E8 0C000000	CALL 00A25BFA	
00A25B1E	66:833B 00	CMP WORD PTR DS:[EBX], 0	
00A25B21	75 E9	JNZ SHORT 00A25BDD	
00A25B24	5E	POP ESI	
00A25B27	5B	POP EBX	
00A25B2A	5D	POP EBP	
00A25B2D	C2 0800	RETN 8	
00A25B30	0102	ADD DWORD PTR DS:[EDX], EAX	
00A25B33	C3	RETN	
00A25B36	03C3	ADD EAX, EBX	
00A25B39	BB C1000000	MOV EBX, 0C1	
00A25B3C	0BDB	OR EBX, EBX	
00A25B3F	75 07	JNZ SHORT 00A25C0F	
00A25B42	894424 1C	MOV DWORD PTR SS:[ESP+1C], EAX	
00A25B45	61	POPAD	
00A25B48	50	PUSH EAX	
00A25B4B	C3	RETN	

The registers window shows the following values:

Register	Value	Comment
EAX	00A26E5B	UNICODE "JS"
ECX	00A25C5B	
EDX	00A25C5B	
EBX	000000C1	
ESP	0012FF94	ASCII "zU"
EBP	0012FF94	ASCII "zU"
ESI	00000000	
EDI	00000000	
EIP	00A25B03	
C 0	ES 0023	32bit 0(FFFFFFFF)
P 0	CS 001B	32bit 0(FFFFFFFF)
A 0	DS 0023	32bit 0(FFFFFFFF)
Z 0	DS 0023	32bit 0(FFFFFFFF)
S 0	FS 0038	32bit 7FDE000(FFF)
T 0	GS 0000	NULL
D 0		
O 0		
LastErr	ERROR_SUCCESS (00000000)	
EFL	00000202	(NO,NB,NE,A,NS,PO,GE)
ST0	empty	2.7157716140221984660e-
ST1	empty	+UNORM 76A0 00001000 00
ST2	empty	7.0668009261801862520e+
ST3	empty	1.4445980367382153660e+
ST4	empty	1.0481952052990497930e-
ST5	empty	1.5288070711298957980e+
ST6	empty	-UNORM EA0C 00000000 F3
ST7	empty	96.00000000000000000000
FST 0020	Cond 0 0 0 0	Err 0 0 1
FCW 027F	Prec NEAR,53	Mask 1

The ASCII window shows the string "UNICODE \"JS\"".

-Step 11: Scroll down the screen a few lines. You will see two values need to push to find. Waaaaa.

Address	Hex dump	Disassembly	Comment
00A25CA9	5F	POP EDI	
00A25CAA	FF57 34	CALL NEAR DWORD PTR DS:[EDI+34]	
00A25CAD	✓ E9 5C000000	JMP 00A25D0E	
00A25CB2	9A 0FF069F0 5F	CALL FAR EBSF:F069F00F	Far call
00A25CB9	02CD	ADD CL, CH	
00A25CBB	2064EB 01	AND BYTE PTR DS:[EBX+EBP*8+1], AH	
00A25CBF	E8 80BDA936	CALL 374C1A51	
00A25CC4	A5	MOV DWORD PTR ES:[EDI], DWORD PTR DS:[EDI]	
00A25CC5	87F2	XCHG EDX, ESI	
00A25CC7	✓ EB 01	JMP SHORT 00A25CD0	
00A25CC9	69C1 EF8F5FEB	IMUL EAX, ECX, EB5F8FEE	
00A25CCF	02CD	ADD CL, CH	
00A25CD1	2055 8B	AND BYTE PTR SS:[EBP-75], DL	
00A25CD4	EC	IN AL, DX	I/O command
00A25CD5	6A FF	PUSH -1	
00A25CD7	68 F0D94500	PUSH 45D9F0	
00A25CDC	68 0A4F4500	PUSH 454F0A	
00A25CE1	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	JMP to MSUCRT._except_handler3
00A25CE7	✓ EB 02	JMP SHORT 00A25CEB	
00A25CE9	CD 20	INT 20	
00A25CEB	5A	PUSH EAX	
00A25CEC	64:8725 00000000	MOV DWORD PTR FS:[0], ESP	
00A25CF3	83EC 68	SUB ESP, 68	
00A25CF6	✓ EB 02	JMP SHORT 00A25CFA	
00A25CF8	CD 20	INT 20	
00A25CFA	53	PUSH EBX	
00A25CFB	✓ EB 02	JMP SHORT 00A25CFF	
00A25CFD	5A	PUSH EAX	

-Step 12: Return Olly open it. We will edit the 00 bytes. First you need to know the case 45 bytes. The more you read the tut will understand more about the case of the stolen bytes.

The form of Case 45 bytes:

```
PUSH EBP
MOV EBP, ESP
PUSH -1
PUSH XXXXXX
PUSH XXXXXX
MOV EAX, DWORD PTR FS: [0]
PUSH EAX
MOV DWORD PTR FS: [0], ESP
SUB ESP, 68
PUSH EBX
PUSH ESI
PUSH EDI
MOV DWORD PTR SS: [EBP-18], ESP
XOR EBX, EBX
MOV DWORD PTR SS: [EBP-4], EBX
PUSH 2
```

OK, we will change the value by XXXXXX

PUSH 45D9F0

PUSH 454FOA

Step-13: We of the entire 00 bytes this. Then press Ctrl + E and type the following:

* before:

* follows:

Address	Hex	dump	Disassembly	Comment
00454F27	.	D3E8	SHR EAX, CL	
00454F29	.	C3	RETN	
00454F2A	>	33C0	XOR EAX, EAX	
00454F2C	.	33D2	XOR EDX, EDX	
00454F2E	.	C3	RETN	
00454F2F	.	55	PUSH EBP	
00454F30	.	8BEC	MOV EBP, ESP	
00454F32	.	6A FF	PUSH -1	
00454F34	.	68 F0D94500	PUSH DVDFab.0045D9F0	
00454F39	.	68 0A4F4500	PUSH DVDFab.00454F0A	JMP to MSVCRT._except_handler3
00454F3E	.	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
00454F44	.	50	PUSH EAX	
00454F45	.	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
00454F4C	.	83EC 58	SUB ESP, 58	
00454F4F	.	53	PUSH EBX	
00454F50	.	56	PUSH ESI	
00454F51	.	57	PUSH EDI	
00454F52	.	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
00454F55	.	33DB	XOR EBX, EBX	
00454F57	.	895D FC	MOV DWORD PTR SS:[EBP-4], EBX	
00454F5A	.	6A 02	PUSH 2	
00454F5C	.	FF15 64974500	CALL NEAR DWORD PTR DS:[459764]	MSVCRT.__set_app_type
00454F62	.	59	POP ECX	
00454F63	.	830D 84FD4700	OR DWORD PTR DS:[47FD84], FFFFFFFF	
00454F6A	.	830D 88FD4700	OR DWORD PTR DS:[47FD88], FFFFFFFF	
00454F71	.	FF15 68974500	CALL NEAR DWORD PTR DS:[459768]	MSVCRT._p__fmode

You found it absolutely khit with just 00 bytes. Good job!

Step-14: Right at the Push EBP image as you choose:

OllyDbg - DVDFab.exe - [CPU - main thread, module DVDFab]

File View Debug Plugins Options Window Help

Address Hex dump Disassembly Comment

00454F29 . C3 RETN

00454F2A > 33C0 XOR EAX, EAX

00454F2C . 33D2 XOR EDX, EDX

00454F2E . C3 RETN

00454F2F . 55 PUSH EBP

00454F30 . 8BEC MOV EBP, ESP

00454F32 . 6A FF PUSH -1

00454F34 . 68 F0D94500 PUSH DVDFab.0

00454F39 . 68 0A4F4500 PUSH DVDFab.0

00454F3E . 64:A1 00000000 MOV EAX, DWO

00454F44 . 50 PUSH EAX

00454F45 . 64:8925 00000000 MOV DWORD PT

00454F4C . 83EC 58 SUB ESP, 58

00454F4F . 53 PUSH EBX

00454F50 . 56 PUSH ESI

00454F51 . 57 PUSH EDI

00454F52 . 8965 E8 MOV DWORD PT

00454F55 . 33DB XOR EBX, EBX

00454F57 . 895D FC MOV DWORD PTR

00454F5A . 6A 02 PUSH 2

00454F5C . FF15 64974500 CALL NEAR DWO

00454F62 . 59 POP ECX

00454F63 . 830D 84FD4700 OR DWORD PTR

00454F6A . 830D 88FD4700 OR DWORD PTR

00454F71 . FF15 68974500 CALL NEAR DWO

00454F72 . 830D 84FD4700 OR DWORD PTR

Registers (FPU)

EAX 00A2ABF4 ASCII "BKatyADG"

ECX 0012FFB0 ASCII "r60"

EDX 7FFE0304

EBX 00000000

ESP 0012FF38

EBP 0012FFC0

ESI 00000000

EDI 52050A33

EIP 00454F5C DVDFab.00454F5C

C 0 ES 0023 32bit 0(FFFFFFFF)

P 1 CS 001B 32bit 0(FFFFFFFF)

A 0 SS 0023 32bit 0(FFFFFFFF)

Z 1 DS 0023 32bit 0(FFFFFFFF)

S 0 FS 0038 32bit 7FDE0000

T 0 GS 0000 NULL

D 0

O 0 LastErr ERROR_SUCCESS

EFL 00010246 (NO,NB,E,BE,NS,

ST0 empty 2.7157716140221984

ST1 empty -UNORM F6A0 000010

ST2 empty -7.066800926180186

ST3 empty -8.750816098429019

ST4 empty 1.0408650498949149

ST5 empty -1.518115939569194

ST6 empty -UNORM 9A0C 000000

ST7 empty 96.000000000000000

EST 0020 Cond 0 0 0 0 Err

Address Value Comment

012FF38 00000002

012FF3C 52050A33

012FF40 00000000

012FF44 7FFDF000

012FF48 0000000C

012FF4C 00A2AC08

012FF50 00A28843

012FF54 00A0C5B0

012FF58 00A27359

012FF5C 224687A0

012FF60 00400000

012FF64 40F37012

012FF68 0012FFA4

012FF6C 0012FF90

012FF70 00A00000

012FF74 009E0000

012FF78 00A2AC08

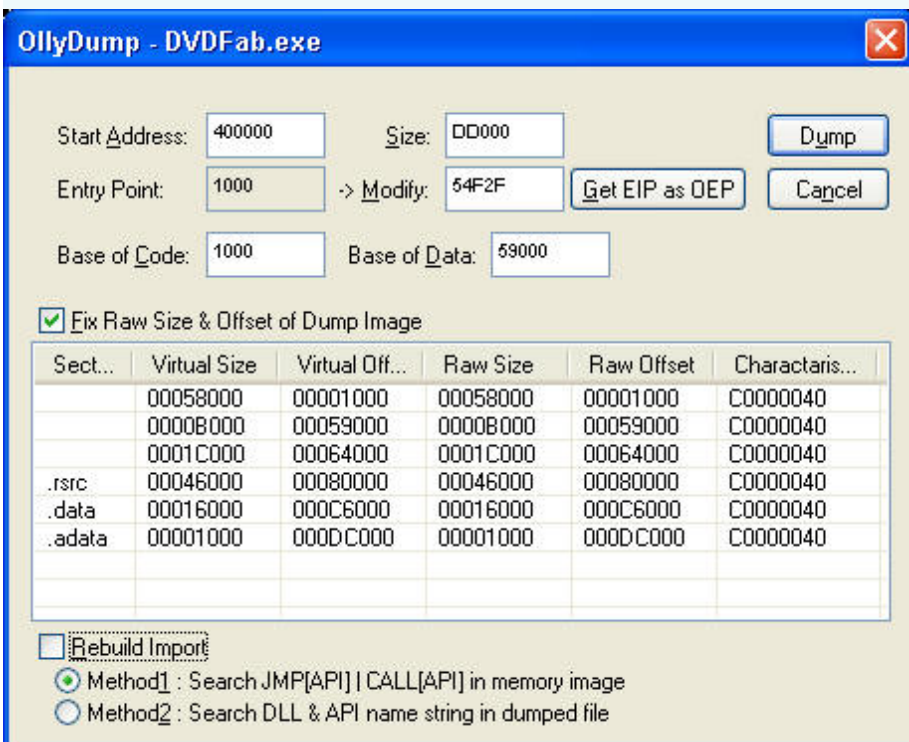
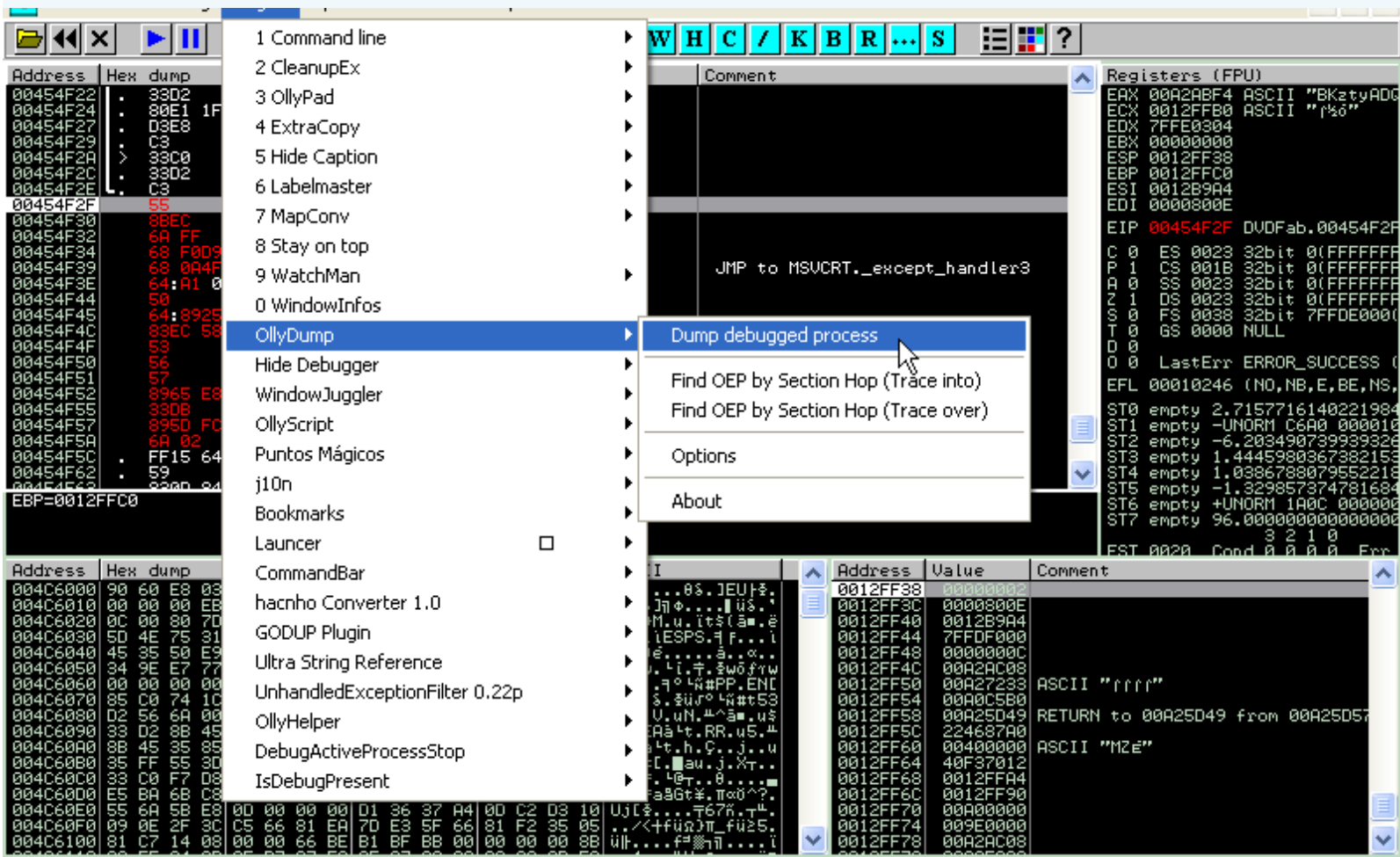
012FF7C 00A2AC08

Command

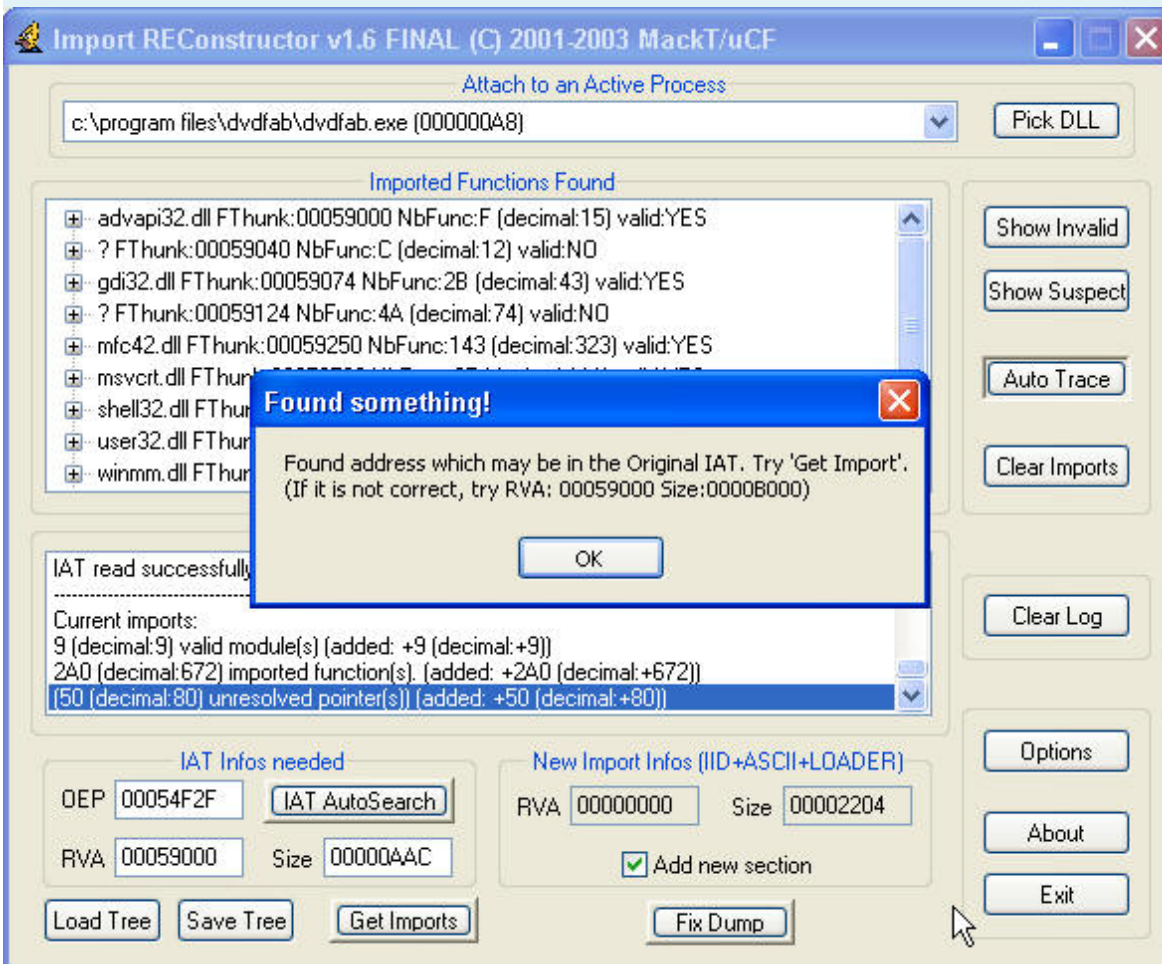
Analysing DVDFab: 1164 strict procedures, 2861 calls to known, 294 calls to guessed functions

start USB... Unp... Micr... ASP... My ... Olly... 6:28 AM

-Step 15: At OllyDUMP, select dump debugged process!



-Step 16: Once dumped, we fix IAT ban ImpREC 1.6



OEP to enter, click IAT AutoSearch. A message for us that if not properly collected by the new size ka B000. This is quite large, should we choose for streamlined in 1000.

Attach to an Active Process

c:\program files\dvd fab\dvd fab.exe (000000A8)

Pick DLL

Imported Functions Found

☒ advapi32.dll FTThunk:00059000 NbFunc:F (decimal:15) valid:YES
☒ ? FTThunk:00059040 NbFunc:C (decimal:12) valid:NO
☒ gdi32.dll FTThunk:00059074 NbFunc:2B (decimal:43) valid:YES
☒ ? FTThunk:00059124 NbFunc:4A (decimal:74) valid:NO
☒ mfc42.dll FTThunk:00059250 NbFunc:143 (decimal:323) valid:YES
☒ msvcrt.dll FTThunk:00059760 NbFunc:6F (decimal:111) valid:YES
☒ shell32.dll FTThunk:00059920 NbFunc:5 (decimal:5) valid:YES
☒ user32.dll FTThunk:00059938 NbFunc:54 (decimal:84) valid:YES
☒ winmm.dll FTThunk:00059A8C NbFunc:1 (decimal:1) valid:YES

Show Invalid

Show Suspect

Auto Trace

Clear Imports

Log

IAT read successfully.

Current imports:

9 (decimal:9) valid module(s)

3C7 (decimal:967) imported function(s). (added: +127 (decimal:+295))

177 (decimal:375) unresolved pointer(s) (added: +127 (decimal:+295))

Clear Log

IAT Infos needed

OEP 00054F2F

IAT AutoSearch

RVA 00059000

Size 00001000

New Import Infos (IID+ASCII+LOADER)

RVA 00000000

Size 00002434

☒ Add new section

Load Tree

Save Tree

Get Imports

Fix Dump

Options

About

Exit

Then click Get Imports. Show Invalid click Next. Click to select trace level1:

Attach to an Active Process

c:\program files\dvd fab\dvd fab.exe (000000A8)

Pick DLL

Imported Functions Found

☒ advapi32.dll FTThunk:00059000 NbFunc:F (decimal:15) valid:YES
☒ ? FTThunk:00059040 NbFunc:C (decimal:12) valid:NO
☒ gdi32.dll FTThunk:00059074 NbFunc:2B (decimal:43) valid:YES
☒ ? FTThunk:00059124 NbFunc:4A (decimal:74) valid:NO
☒ mfc42.dll FTThunk:00059250 NbFunc:143 (decimal:323) valid:YES
☒ msvcrt.dll FTThunk:00059760 NbFunc:6F (decimal:111) valid:YES
☒ shell32.dll FTThunk:00059920 NbFunc:5 (decimal:5) valid:YES
☒ user32.dll FTThunk:00059938 NbFunc:54 (decimal:84) valid:YES
☒ winmm.dll FTThunk:00059A8C NbFunc:1 (decimal:1) valid:YES

Show Invalid

Show Suspect

Auto Trace

Clear Imports

Log

IAT read successfully.

Current imports:

9 (decimal:9) valid module(s)

3C7 (decimal:967) imported function(s). (added: +127 (decimal:+295))

177 (decimal:375) unresolved pointer(s) (added: +127 (decimal:+295))

Clear Log

IAT Infos needed

OEP 00054F2F

IAT AutoSearch

RVA 00059000

Size 00001000

New Import Infos (IID+ASCII+LOADER)

RVA 00000000

Size 00002434

☒ Add new section

Load Tree

Save Tree

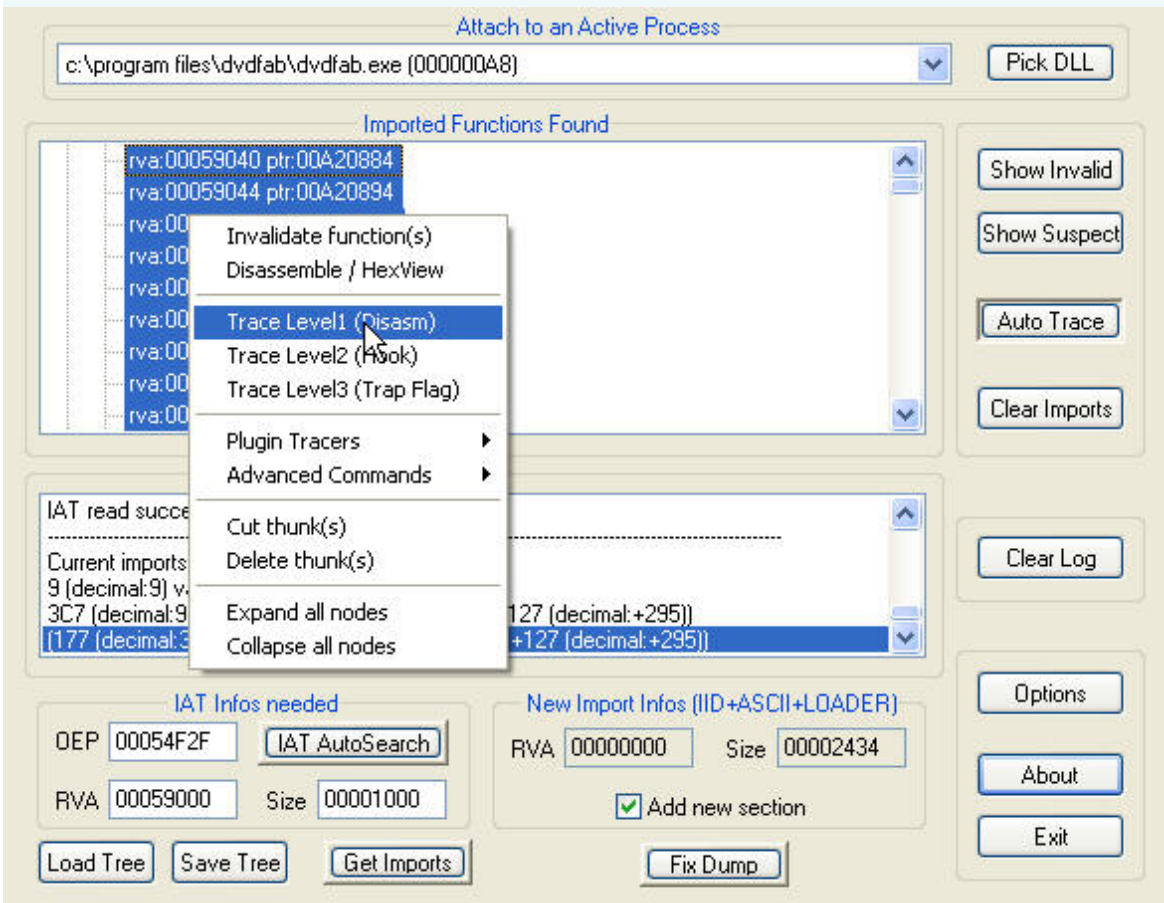
Get Imports

Fix Dump

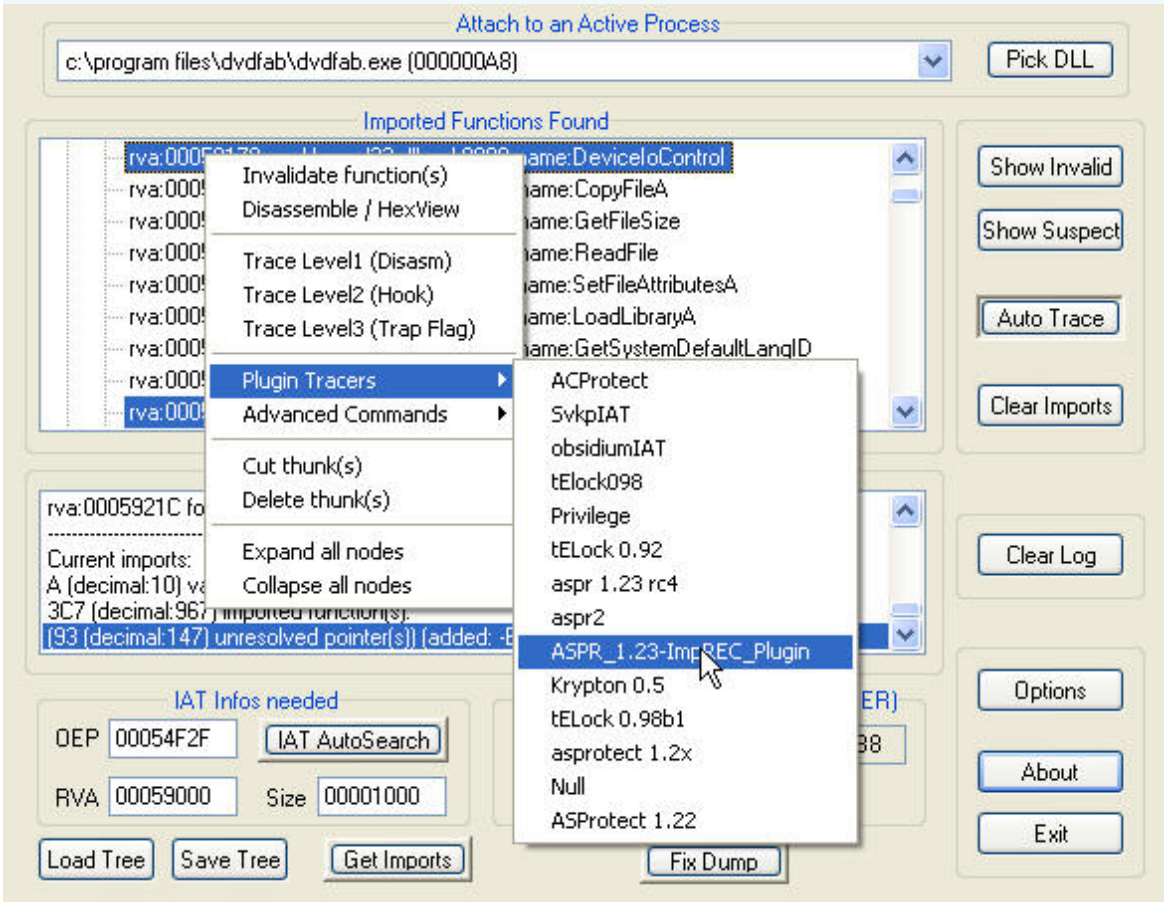
Options

About

Exit



-Then click Show Invalid more, and click to select the image:



-Continue Show Invalid clicks again. And this time, only thunks Cut: d.

OllyDbg - DVDFab.exe - [CPU - main thread, module DVDFab]

File View Debug Plugins Options Window Help

Import REConstructor v1.6 FINAL (C) 2001-2003 MackT/uCF

Attach to an Active Process

c:\program files\dvdfab\dvdfab.exe (00000604) Pick DLL

Imported Functions Found

Function	Thunk	NbFunc	Valid
mfc42.dll	FTThunk:00059250	NbFunc:143 (decimal:323)	valid:YES
msvcrt.dll	FTThunk:00059760	NbFunc:6F (decimal:111)	valid:YES
shell32.dll	FTThunk:00059920	NbFunc:5 (decimal:5)	valid:YES
user32.dll	FTThunk:00059938	NbFunc:54 (decimal:84)	valid:YES
winmm.dll	FTThunk:00059A8C	NbFunc:1 (decimal:1)	valid:YES
comdlg32.dll	FTThunk:00059A94	NbFunc:2 (decimal:2)	valid:YES
ole32.dll	FTThunk:00059AA0	NbFunc:2 (decimal:2)	valid:YES
? FTThunk:00059AB4	NbFunc:1 (decimal:1)	valid:NO	

Show Invalid Show Suspect Auto Trace Clear Imports

Log

RVA:00000000 decimal:+1))
Current i 3C7 (dec
B (dec
[8D (dec
added: -6 (decimal:-6))

Options About Exit

New Import Infos (IID+ASCII+LOADER)
RVA 00000000 Size 00002D32
☒ Add new section
Fix Dump

Invalidate function(s)
Disassemble / HexView
Trace Level1 (Disasm)
Trace Level2 (Hook)
Trace Level3 (Trap Flag)
Plugin Tracers
Advanced Commands
Cut thunk(s)
Delete thunk(s)
Expand all nodes
Collapse all nodes

EBP=0012FFC0

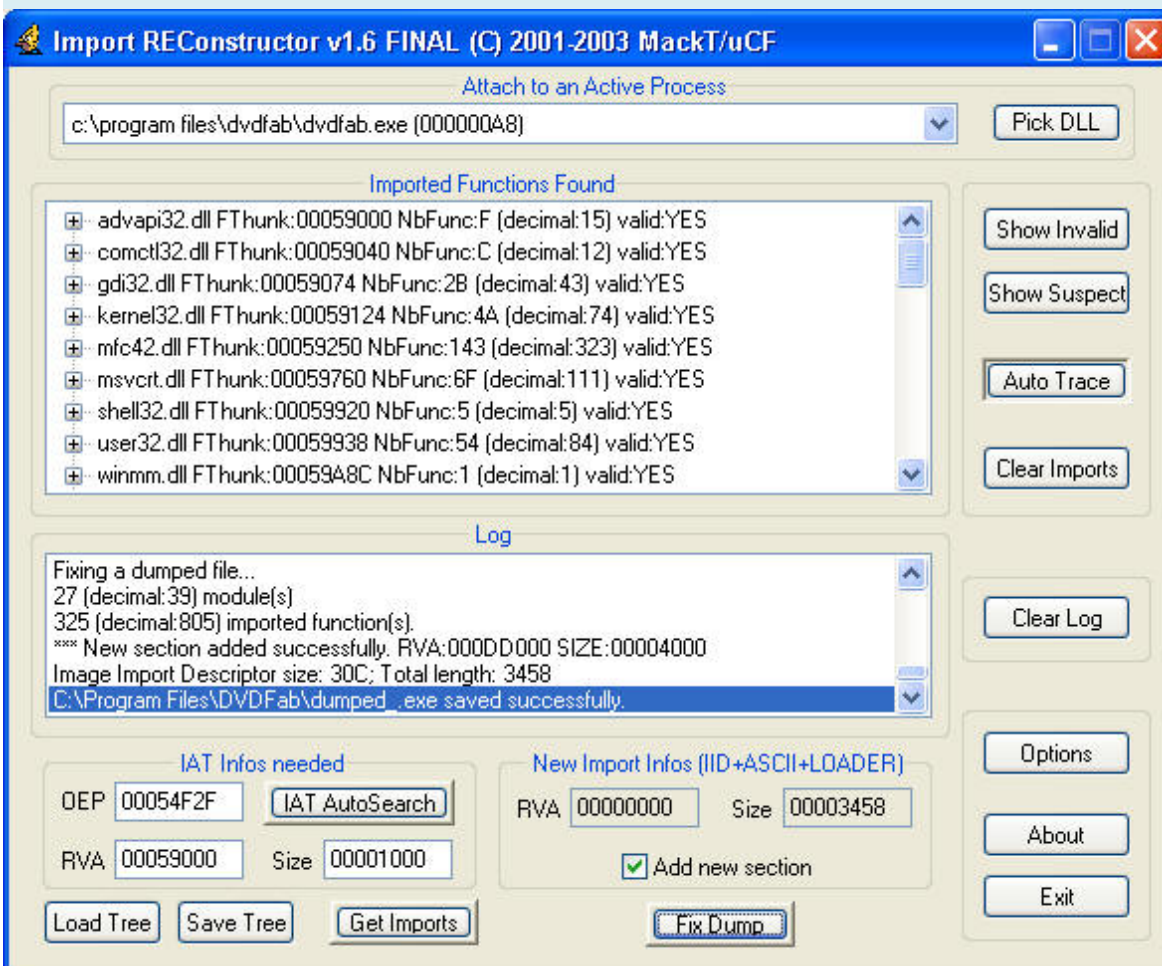
Address	Hex	dump
004C6000	90 60 E6	
004C6010	00 00 00	
004C6020	0C 00 80	
004C6030	50 4E 75	
004C6040	45 35 50	
004C6050	34 9E E7	
004C6060	00 00 00	
004C6070	85 C0 74	
004C6080	02 56 60	
004C6090	33 D2 8E	
004C60A0	8B 45 35	
004C60B0	35 FF 55	
004C60C0	33 C0 F7	
004C60D0	E5 BA 6E	
004C60E0	55 6A 5E	
004C60F0	09 0E 2F	
004C6100	81 C7 14	

Command

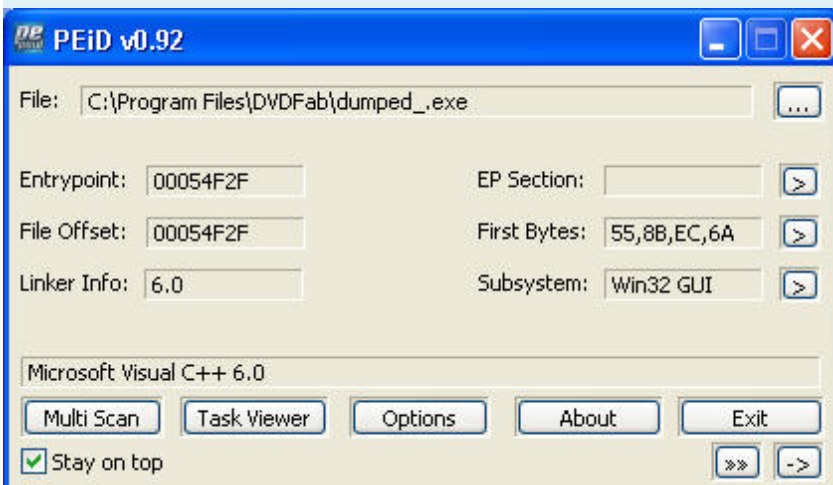
Analysing DVDFab:1164 strict procedures,2861 calls to known,294 calls to guessed functions

Paused

start U... U... M... A... M... O... I... 6:37 AM



Step-17: Test file dumped_.exe see stars? OK, very good run, not crash. Using the considered view PEid ;-).



The goal is complete with a case 45 bytes. The other type of cases, you can apply similar Advanced Windows Password Recovery (Case 38 bytes), NetworkSearcher (Case 38 bytes) ...

Bonus: you raise your table the case that I have a script for the OEP 1.23RC4 version. Very good, there is no guarantee on the homepage Ollycrypts.

```

////////////////////////////////////
// Asprotect //
// Date: 19/10/2004 //
// //
var cebp //////////////////////////////////
var cesp

```

```
var addra
var addra2
var addrb
var addrclast
var count
var test
var addrc
var addrc2
var Valid
var valid2
var csize
var msize
var popc
eoe checklast
eob checklast
GMI 401000, CODESIZE
mov csize, $ RESULT
var sizet
mov sizet, csize
add sizet, 400,000
GMI 401000, MODULESIZE
mov msize, $ RESULT
add msize, 400,000
```

esto

```
checklast:
dbh
Cmp edx, 4
jne f
mov popc, eip
add popc, 4
mov popc, [popc]
Cmp popc, 0000068f
jne f
find eip, # 74? E8 #
mov popc, $ RESULT
sub popc, 5
mov popc, [popc]
mov [popc], 1
Cmp $ result, 0
```

```
je f:
bprm 401,000, csize
eob oep
eoe oep
esto
f:
find eip, 85c00f85 # #
Cmp $ result, 0
je cntlast
mov valid, $ RESULT
Valid sub, 3e
Cmp [valid], 00001fb8
```

```
jne cntlast
mov valid2, $ RESULT
sub valid2, eip
Cmp valid2, Off
```



```
ja cntlast
eob bypass
bp $ RESULT
esto

Bypass:
mov eax, 0
bc $ RESULT
esto
cntlast:
eoe checklast
eob checklast
mov addra, ebp
mov addrc, ebp
sub addra, 10
mov addra2, addra
mov addrc2, addra
mov cesp, esp
mov cebp, ebp
and cesp, 00ff0000
and cebp, 00ff0000
Cmp cesp, cebp
jne false

mov addra, [addra]

Cmp addra, 400,000

jne false1
add addra2, 4
mov addra2, [addra2]
Cmp addra2, msize
jb foundlast
false1:
sub addrc, 20
mov addrc2, addrc
mov cesp, esp
mov cebp, ebp
and cesp, 00ff0000
and cebp, 00ff0000
Cmp cesp, cebp

jne false

mov addrc, [addrc]

Cmp addrc, 400,000
jne false
add addrc2, 4
mov addrc2, [addrc2]
Cmp addrc2, msize

ja test1
Cmp addrc2, 401,000
ja foundlast
jmp false
test1:
mov addrc2, edi
and addrc2, 0000ffff
```

Cmp addrc2, 0

je foundlast

false:

esto

ret

foundlast:

MSGYN "this is the last exception, do you want to continue to the OEP?"

Cmp \$ result, 0

je last

jmp oepn

oep:

Cmp eip, sizet

jb oepf

esto

oepn:

bprm 401,000, csize

cob

CoE

esto

oepf:

msg "this is the oep if it stolen, Thanks for using my scripts; BriteDream"

bpmc

ret

Last:

msg "This is the last exception, Thank you for using my scripts; BriteDream"

ret

8 bytes:

Code:

005B8F74 <Module> / \$ 55 PUSH EBP

005B8F75 |. 8BEC MOV EBP, ESP

005B8F77 |. B9 06000000 MOV ECX, 6

11 bytes (Case I)

Code:

00495778 <Module> \$ 55 PUSH EBP

00495779. 8BEC MOV EBP, ESP

0049577B. 83EC 0C SUB ESP, 0C

0049577E. B8 E8544900 MOV EAX, dumped_1.004954E8

11 bytes (Case II)

Code:

005008BC <Module> \$ 55 PUSH EBP

005008BD. 8BEC MOV EBP, ESP

005008BF. B8 5C055000 MOV EAX, dumped_1.0050055C

005008C4. 83C4 F4 ADD ESP, 0C -

12 bytes:

Code:

004B38C0> \$ 55 PUSH EBP

```
004B38C1. 8BEC MOV EBP, ESP
004B38C3. 83C4 F4 ADD ESP, 0C -
004B38C6. 53 PUSH EBX
004B38C7. B8 58334B00 MOV EAX, dumped_.004B3358
16 bytes:
```

Code:

```
004FF904 <Module> / $ 55 PUSH EBP
004FF905 |. 8BEC MOV EBP, ESP
004FF907 |. 83EC 14 SUB ESP, 14
004FF90A |. 33C0 XOR EAX, EAX
004FF90C |. 8945 EC MOV DWORD PTR SS: [EBP-14], EAX
004FF90F |. B8 54F34F00 MOV EAX, dumped_1.004FF354
20 bytes:
```

Code:

```
00553710 00 DB 00
00553711 00 DB 00
00553712 00 DB 00
00553713 00 DB 00
00553714 00 DB 00
00553715> / $ 55 PUSH EBP
00553716 |. 8BEC MOV EBP, ESP
00553718 |. 51 PUSH ECX
00553719 |. 52 PUSH EDX
0055371A |. 53 PUSH EBX
0055371B |. 50 PUSH EAX
0055371C |. 6A 00 PUSH 0
0055371E |. 53 PUSH EBX
0055371F |. B8 A0305500 MOV EAX, SOunpack.005530A0
22 bytes:
```

Code:

```
0057B0F8 <Module> $ 55 PUSH EBP
0057B0F9. 8BEC MOV EBP, ESP
0057B0FB. 83EC 18 SUB ESP, 18
0057B0FE. 53 PUSH EBX
0057B0FF. 56 PUSH ESI
0057B100. 57 PUSH EDI
0057B101. 33C0 XOR EAX, EAX
0057B103. 8945 E8 MOV DWORD PTR SS: [EBP-18], EAX
0057B106. 8945 EC MOV DWORD PTR SS: [EBP-14], EAX
0057B109. B8 78A95700 MOV EAX, a.0057A978
38 bytes:
```

Code:

```
0066B131 55 PUSH EBP
0066B132 8BEC MOV EBP, ESP
0066B134 6A FF PUSH -1
0066B136 68 C0716C00 PUSH opera.006C71C0
0066B13B 68 D8B96600 PUSH opera.0066B9D8
0066B140 64: A1 00000000 MOV EAX, DWORD PTR FS: [0]
0066B146 50 PUSH EAX
0066B147 64: 8925 00000000 MOV DWORD PTR FS: [0], ESP
0066B14E 83EC 58 SUB ESP, 58
0066B151 53 PUSH EBX
0066B152 56 PUSH ESI
```



```
0066B153 57 PUSH EDI
0066B154 8965 E8 MOV DWORD PTR SS: [EBP-18], ESP
45 bytes:

Code:

00475278> 55 PUSH EBP
00475279 8BEC MOV EBP, ESP
0047527B 6A FF PUSH -1
0047527D 68 58BE4800 PUSH dumped_.0048BE58
00475282 68 46504700 PUSH <JMP.&msvcrt.#206>
00475287 64: A1 00000000 MOV EAX, DWORD PTR FS: [0]
0047528D 50 PUSH EAX
0047528E 64:8925 00000000 MOV DWORD PTR FS: [0], ESP
00475295 83EC 68 SUB ESP, 68
00475298 53 PUSH EBX
00475299 56 PUSH ESI
0047529A 57 PUSH EDI
0047529B 8965 E8 MOV DWORD PTR SS: [EBP-18], ESP
0047529E 33DB XOR EBX, EBX
004752A0 895D FC MOV DWORD PTR SS: [EBP-4], EBX
004752A3 6A 02 PUSH 2
```

Stolen Asprotect Bytes [By Computer_Angel)

Quote:
Case 38 bytes:

```
0066B131 55 PUSH EBP
0066B132 8BEC MOV EBP, ESP
0066B134 6A FF PUSH -1
0066B136 68 C0716C00 PUSH opera.006C71C0
0066B13B 68 D8B96600 PUSH opera.0066B9D8
0066B140 64: A1 00000000 MOV EAX, DWORD PTR FS: [0]
0066B146 50 PUSH EAX
0066B147 64:8925 00000000 MOV DWORD PTR FS: [0], ESP
0066B14E 83EC 58 SUB ESP, 58
0066B151 53 PUSH EBX
0066B152 56 PUSH ESI
0066B153 57 PUSH EDI
0066B154 8965 E8 MOV DWORD PTR SS: [EBP-18], ESP
```

Quote:
11 bytes:

```
005008BC <Module> $ 55 PUSH EBP
005008BD. 8BEC MOV EBP, ESP
005008BF. B8 5C055000 MOV EAX, dumped_1.0050055C
005008C4. 83C4 F4 ADD ESP, 0C -
```

Quote:
22 bytes:

```
0057B0F8 <Module> $ 55 PUSH EBP
0057B0F9. 8BEC MOV EBP, ESP
```

```
0057B0FB. 83EC 18 SUB ESP, 18
0057B0FE. 53 PUSH EBX
0057B0FF. 56 PUSH ESI
0057B100. 57 PUSH EDI
0057B101. 33C0 XOR EAX, EAX
0057B103. 8945 E8 MOV DWORD PTR SS: [EBP-18], EAX
0057B106. 8945 EC MOV DWORD PTR SS: [EBP-14], EAX
0057B109. B8 78A95700 MOV EAX, a.0057A978
```

Quote:

16 bytes:

```
004FF904 <Module> / $ 55 PUSH EBP
004FF905 |. 8BEC MOV EBP, ESP
004FF907 |. 83EC 14 SUB ESP, 14
004FF90A |. 33C0 XOR EAX, EAX
004FF90C |. 8945 EC MOV DWORD PTR SS: [EBP-14], EAX
004FF90F |. B8 54F34F00 MOV EAX, dumped_1.004FF354
```

11 bytes:

Quote:

```
00495778 <Module> $ 55 PUSH EBP
00495779th 8BEC MOV EBP, ESP
0049577B. 83EC 0C SUB ESP, 0C
0049577E. B8 E8544900 MOV EAX, dumped_1.004954E8
```

8 bytes:

Quote:

```
005B8F74 <Module> / $ 55 PUSH EBP
005B8F75 |. 8BEC MOV EBP, ESP
005B8F77 |. B9 06000000 MOV ECX, 6
```

12 bytes:

Quote:

```
004B38C0> $ 55 PUSH EBP
004B38C1. 8BEC MOV EBP, ESP
004B38C3. 83C4 F4 ADD ESP, 0C -
004B38C6. 53 PUSH EBX
004B38C7. B8 58334B00 MOV EAX, dumped_.004B3358
```

20 bytes:

Quote:

```
00553710 00 DB 00
00553711 00 DB 00
00553712 00 DB 00
00553713 00 DB 00
```

```

00553714 00 DB 00
00553715> / $ 55 PUSH EBP
00553716 |. 8BEC MOV EBP, ESP
00553718 |. 51 PUSH ECX
00553719 |. 52 PUSH EDX
0055371A |. 53 PUSH EBX
0055371B |. 50 PUSH EAX
0055371C |. 6A 00 PUSH 0
0055371E |. 53 PUSH EBX
0055371F |. B8 A0305500 MOV EAX, SOunpack.005530A0
-----

```

==== Microsoft Visual C + + 6.0 ====

```

55 PUSH EBP
8BEC MOV EBP, ESP
6AFF PUSH -1
6800000000 PUSH 00000000; à mettre à jour
6800000000 PUSH 00000000; à mettre à jour
64A100000000 MOV EAX, DWORD PTR FS: [0]
50 PUSH EAX
64892500000000 MOV DWORD PTR FS: [0], ESP
83EC58 SUB ESP, 58
53 PUSH EBX
56 PUSH ESI
57 PUSH EDI
8965E8 MOV DWORD PTR [EBP-18], ESP

```

```

558BEC6AFF6800000000680000000064A10000
0000506489250000000083EC585356578965E8

```

=> 38 stolen bytes

=====

==== Microsoft Visual C + + 7.0 ====

```

6A60 PUSH 60
6800000000 PUSH 00000000; à mettre à jour

```

```

6A606800000000

```

=> 7 stolen bytes

=====

==== ===== Borland Delphi

```

55 PUSH EBP
8BEC MOV EBP, ESP
83C4F4 ADD ESP, 0C -
B800000000 MOV EAX, 00000000; à mettre à jour

```

```

558BEC83C4F4B800000000

```

=> 11 stolen bytes

```

55 PUSH EBP

```


8BEC MOV EBP, ESP
 83EC10 SUB ESP, 10
 B800000000 MOV EAX, 00000000; à mettre à jour

558BEC83EC10B800000000

=> 11 stolen bytes

55 PUSH EBP
 8BEC MOV EBP, ESP
 83C4F4 ADD ESP, 0C -
 53 PUSH EBX
 B800000000 MOV EAX, 00000000; à mettre à jour

558BEC83C4F453B800000000

= 12 bytes stolen

55 PUSH EBP
 8BEC MOV EBP, ESP
 83C4F0 ADD ESP, -10
 53 PUSH EBX
 B800000000 MOV EAX, 00000000; à mettre à jour

558BEC83C4F053B800000000

= 12 bytes stolen

=====

Autres cas possibles:

22 bytes:

55 PUSH EBP
 8BEC MOV EBP, ESP
 83EC18 SUB ESP, 18
 53 PUSH EBX
 56 PUSH ESI
 57 PUSH EDI
 33C0 XOR EAX, EAX
 8945E8 MOV DWORD PTR SS: [EBP-18], EAX
 8945EC MOV DWORD PTR SS: [EBP-14], EAX
 B800000000 MOV EAX, 00000000; à mettre à jour

16 bytes:

55 PUSH EBP
 8BEC MOV EBP, ESP
 83EC14 SUB ESP, 14
 33C0 XOR EAX, EAX
 8945EC MOV DWORD PTR SS: [EBP-14], EAX

B800000000 MOV EAX, 00000000; à mettre à jour

8 bytes:

55 PUSH EBP

8BEC MOV EBP, ESP

B900000000 MOV ECX, 00000000; à mettre à jour

00 DB 00

00 DB 00

00 DB 00

00 DB 00

00 DB 00

55 PUSH EBP

8BEC MOV EBP, ESP

51 PUSH ECX

52 PUSH EDX

53 PUSH EBX

50 PUSH EAX

6A00 PUSH 0

B800000000 MOV EAX, 00000000; à mettre à jour

(C'est un cas special).

25 bytes:

55 PUSH EBP

8BEC MOV EBP, ESP

83C4E4 ADD ESP,-1C

53 PUSH EBX

56 PUSH ESI

57 PUSH EDI

33C0 XOR EAX, EAX

8945E4 MOV DWORD PTR SS: [EBP-1C], EAX

8945E8 MOV DWORD PTR SS: [EBP-18], EAX

8945EC MOV DWORD PTR SS: [EBP-14], EAX

B800000000 MOV EAX, 00000000; à mettre à jour

45 bytes:

PUSH EBP

MOV EBP, ESP

ADD ESP,-12C

PUSH EBX

XOR EAX, EAX

MOV [EBP] [-120], EAX

MOV [EBP] [-124], EAX

```
MOV [EBP] [-118], EAX
MOV [EBP] [-11C], EAX
MOV [EBP] [-14], EAX
MOV EAX, 00333333
CALL ...
```

32 bytes:

```
PUSH EBP
MOV EBP, ESP
ADD ESP, -1C
PUSH EBX
XOR EAX, EAX
MOV [EBP] [-1C], EAX
MOV [EBP] [-18], EAX
MOV [EBP] [-14], EAX
MOV EAX, [00407C14]
MOV B, [EAX], 001
MOV EAX, 00333333
CALL ...
```

Possible Stolen bytes: ASProtect

1) M \$ Visual C + +:

```
55 PUSH EBP
8BEC MOV EBP, ESP
6A FF PUSH -1
```

```
68 xxxxxx00 PUSH target_name.00xxxxxx | Look in stack window for "Pointer to next SEH record"
68 xxxxxx00 PUSH target_name.00xxxxxx | Besides the below will see Handler 2 Pushed addresses
```

```
64: A1 00000000 MOV EAX, DWORD PTR FS: [0]
50 PUSH EAX
64: 8925 00000000 MOV DWORD PTR FS: [0], ESP
83EC 58 SUB ESP, 58
53 PUSH EBX
56 PUSH ESI
57 PUSH EDI
8965 E8 MOV DWORD PTR SS: [EBP-18], ESP
```

OR MFC application:

```
55 PUSH EBP
8BEC MOV EBP, ESP
6A FF PUSH -1
```

```
68 xxxxxx00 PUSH target_name.00xxxxxx | Look in stack window for "Pointer to next SEH record"
68 xxxxxx00 PUSH target_name.00xxxxxx | Besides the below will see Handler 2 Pushed addresses
```

```
64: A1 00000000 MOV EAX, DWORD PTR FS: [0]
50 PUSH EAX
```



```

64:8925 00000000 MOV DWORD PTR FS: [0], ESP
83EC 68 SUB ESP, 68
53 PUSH EBX
56 PUSH ESI
57 PUSH EDI
8965 E8 MOV DWORD PTR SS: [EBP-18], ESP
33DB XOR EBX, EBX
895D FC MOV DWORD PTR SS: [EBP-4], EBX
6A 02 PUSH 2

```

2) Borland Delphi:

Stolen 8 Bytes

```

55 PUSH EBP
8BEC MOV EBP, ESP
B9 0x000000 MOV ECX, x <--- Check ECX in value when the land at fake OEP

```

Stolen 11 Bytes

```

55 PUSH EBP
8BEC MOV EBP, ESP
83C4 F0 ADD ESP, -10
B8 xxxxxx00 MOV EAX, target_name.00xxxxxx <- execute or POP EAX MOV EAX, EBX with F7 when you
land after tracing from last RETN value and then check in EAX and then continue on land and F7 Fake OEP

```

Stolen 12 Bytes

```

55 PUSH EBP
8BEC MOV EBP, ESP
83C4 F4 ADD ESP, 0C -
53 PUSH EBX
B8 xxxxxx00 MOV EAX, target_name.00xxxxxx <- execute or POP EAX MOV EAX, EBX with F7 when you
land after tracing from last RETN value and then check in EAX and then continue on land and F7 Fake OEP

```

Stolen 14 Bytes

```

55 PUSH EBP
8BEC MOV EBP, ESP
83C4 F4 ADD ESP, 0C -
53 PUSH EBX
56 PUSH ESI
57 PUSH EDI
B8 xxxxxx00 MOV EAX, target_name.00xxxxxx <- execute or POP EAX MOV EAX, EBX with F7 when you
land after tracing from last RETN value and then check in EAX and then continue on land and F7 Fake OEP

```

Stolen 19 Bytes

```

55 PUSH EBP
8BEC MOV EBP, ESP
33C9 XOR ECX, ECX
51 PUSH ECX
51 PUSH ECX

```

```
51 PUSH ECX
51 PUSH ECX
51 PUSH ECX
51 PUSH ECX
51 PUSH ECX
53 PUSH EBX
56 PUSH ESI
B8 xxxxxx00 MOV EAX, target_name.00xxxxxx <- Check EAX
```

See in other tut! I will have the opportunity to write a tut about ASProtect 1.3x and 2.x. Wait for me!

hacnho [VCT2k4]

<http://tothesky.us>

lenguyenkhang@gmail.com

GrEeTs Fly Out: Deux, infinite, luucorp, Aaron, Canterwood, hhphong, R @ dier, tlandn, Computer_Angel, Moonbaby, RCA, CTL, Zombie, Nilrem, Ferrari, hosiminh, Govindah, elooo, Seven, diablo2oo2's, anh_surprised ... And you ;-)!

Thanx to authors of OllyDBG, ImpREC, LordPE, OllyScript, ASProtect.
To be continued ...

Written by hacnho (tutorial date: Sai Gon 10/01/2004)

L i t t l e t t o u r i a l A u t o b e x e s t e a l t h 2. 7 3 b i t a l y i t a l y N o. W h t

B a r

T a g e r t: a G I C M A S C I S T I u d i o 2 . 2

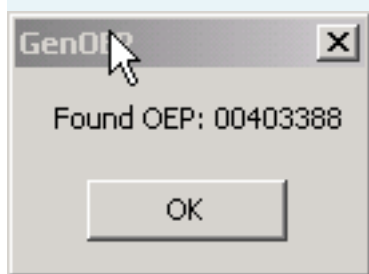
H o m e P a g e : [H_t_t_p:/_/_w_w_w._I_t_a_l_y_x_o_o_s_o_f_t._C_o_m/](http://www.Italy_x_o_o_s_o_f_t._C_o_m/)

Unkpac Tool : **OllitalyBDG 1.10** and **ItalyllOMUDP** **puling**

_Using **PeiD 0.94** **fordetect** **ESTEXealth 2.72 - 2.73** **->** **WebToolMaster**



_ Usingpuling "GenericePOFinder" ForSearchPOE



_ O K, Loadtagertinto O ll italy B D G italy and ou will pause here

Why Not Bar - [UPC= main thread, module Magic_AS]

File View Debug Plugins Options Window Help Tools CrackTools Languages

Paused

00582060	EB 00	jmp	short 00582062
00582062	EB 2F	jmp	short 00582093
00582064	53	push	ebx
00582065	68 61726577	push	77657261
0058206A	61	popad	
0058206B	72 65	jbs	short 005820D2
0058206D	202D 20457865	and	[65784520], ch
00582073	53	push	ebx
00582074	74 65	je	short 005820DB
00582076	61	popad	
00582077	6C	ins	byte ptr es:[edi], dx
00582078	74 68	je	short 005820E2
0058207A	00EB	add	bl, ch
0058207C	16	push	ss
0058207D	77 77	ja	short 005820F6

_No w, e s s P l a t + F 1, "H e 403,338 a n d P r e s t e r s e n

Address	Value			
		▲	0012FFC4	7C816D4F RETURN to kernel32.7C8
0057E000	00000000		0012FFC8	7C910738 ntdll.7C910738
0057E004	41AC5A8C		0012FFCC	FFFFFFFF
0057E008	0014BC68		0012FFD0	7FFDE000
0057E00C	00030000		0012FFD4	8054B038
0057E010	00000003		0012FFD8	0012FFC8
0057E014	80000028		0012FFDC	815BF898
0057E018	0000000E		0012FFE0	FFFFFFFF End of SEH chain
0057E01C	80000068	▼	0012FFE4	7C8399F3 SE handler
			0012FFE8	7C816D58 kernel32.7C816D58
Command <input type="text" value="he 403338"/> <input type="button" value="HE address -- HW break on execution"/>				
Program entry point				

_Press F9, You will lander here and this is OEP

*** - [UPC= main thread, module Magic_AS]**

File View Debug Plugins Options Window Help Tools CrackTools Languages

Paused

00403368	- FF25 A4104000	jmp	[4010A4]	msvbvm60.GetMemStr
0040336E	- FF25 C4104000	jmp	[4010C4]	msvbvm60.PutMemStr
00403374	- FF25 8C104000	jmp	[40108C]	msvbvm60.GetMem4
0040337A	- FF25 C0104000	jmp	[4010C0]	msvbvm60.PutMem4
00403380	- FF25 E4114000	jmp	[4011E4]	msvbvm60.ThunRTMain
00403386	0000	add	[eax], al	
00403388	68 10384000	push	00403810	<== Day 1a OEP
0040338D	E8 EFFFFFFF	call	00403380	jmp to msvbvm60.ThunRTMain
00403392	0000	add	[eax], al	
00403394	0000	add	[eax], al	
00403396	0000	add	[eax], al	
00403398	3000	xor	[eax], al	

_ Go to `melpuifign`, choose `litaly O D U M P` and check **"Rebuild Import"**

OllyDump - Magic ASCII Studio.exe

Start Address: Size:

Entry Point: -> Modify:

Base of Code: Base of Data:

☒ Fix Raw Size & Offset of Dump Image

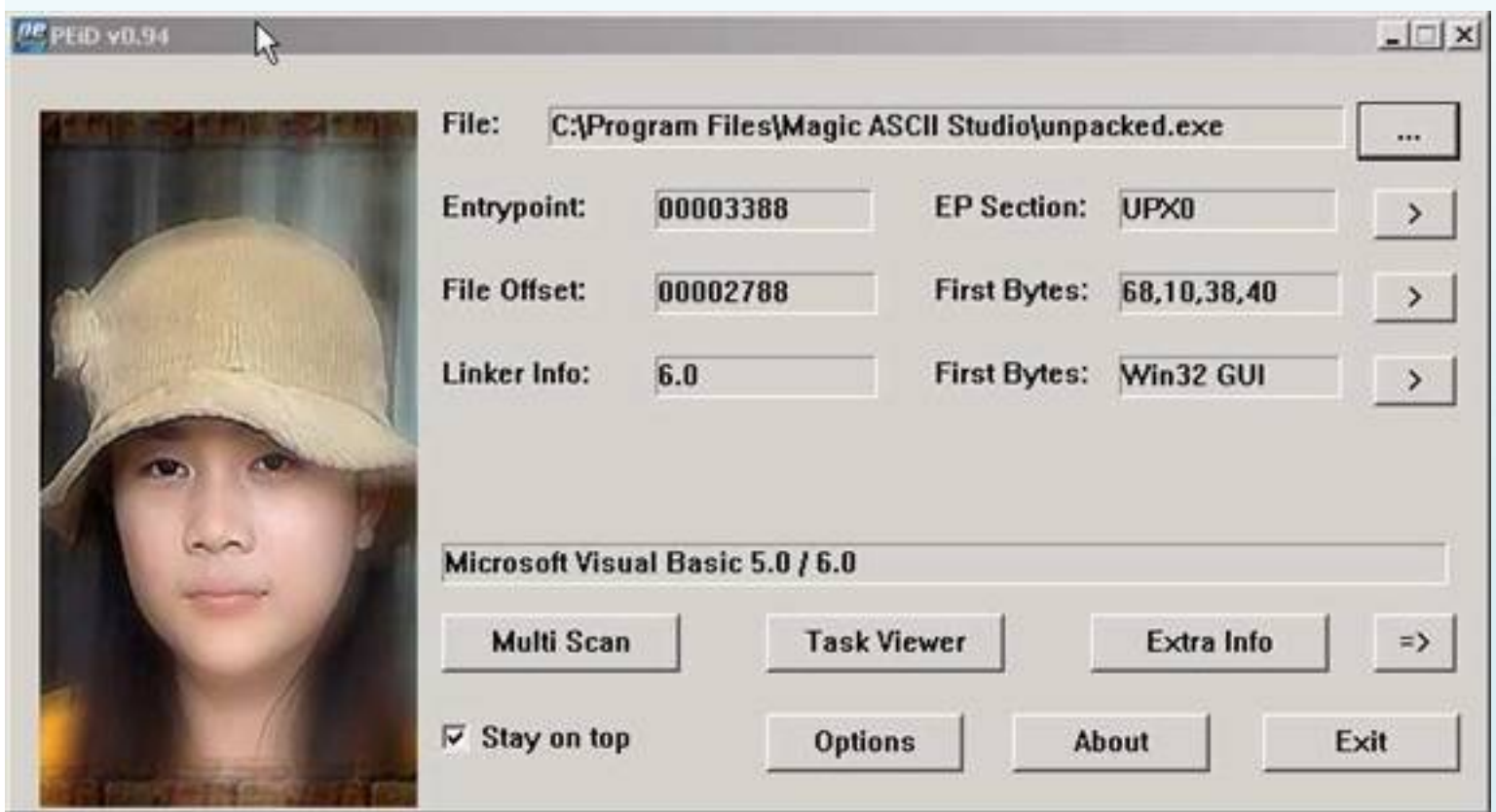
Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
UPX0	00135000	00001000	00135000	00001000	C0000040
UPX1	00048000	00136000	00048000	00136000	C0000040
.rsrc	00002000	0017E000	00002000	0017E000	C0000040
.aspack	00001000	00180000	00001000	00180000	C0000040
.adata	00001000	00181000	00001000	00181000	C0000040
ExeS	00002000	00182000	00002000	00182000	E00000E0

☒ Rebuild Import

- ☒ Method1 : Search JMP[API] | CALL[API] in memory image
- ☐ Method2 : Search DLL & API name string in dumped file

_No w r un heu of the PA c k e e dfil. I t 'sOk a italy ... ingPEiD0.94f U s e d o t r e c t: **M i c r o**

s o f t V i s u a l i s a B 5.0 / 6.0



Okie, [exestealth](#) is on wunpackede succssful! Th a t e s a italy! H uh.

Italy N W h t o a r B

G r e s e T F O L i t a l y u t C o m p u t e r a n _ l e g , Z o m b i e , M o o n B a b i t a l y , H a o f t h e o , B e a

nin , k i n e m a n a r o w , i o Z , D x u e , M e r c , l i g h p o H T e w h e r e x , T r i c o b u p i t a l y , T A

K A d a , a i m i i d t o , t h e H T p g e h o n i x , t o f t h e n a n d e a n ... n d y o u !

Homepage: <http://www.magictweak.com>

Production: Efreesky Software

Software: **MagicTweak v2.70**

Copyright by: Copyright (C) 2000 - 2003 Efreesky Software. All Rights Reserved.

Type: N / S

Packed: **PECompact 1.68 - 1.84 -> Jeremy Collake**

Language: Microsoft Visual C + + 6.0

Crack Tool: 1:10 OllyDbg, PEiD 0.92, Import REConstructor v1.6F, LordPE v1.4

Unpack: Manual

Request: Correct Serial

Comments:

MagicTweak v2.70

MagicTweak is a special program designed to optimize and personalize Microsoft Windows. It provides one-stop, instant access to a variety of Windows settings that can be altered for a friendlier Windows environment. This unique software makes it easy to tweak hundreds of hidden settings in Windows XP/2000/Me/98, so there is no longer any need to dig through the registry looking for that specific setting (from Start Menu, Desktop, IE skin, System Icon to System Security) that just does not seem to be there. With the ability to customize almost any aspect of Windows, you can become a Windows expert almost instantly!

I - Information:

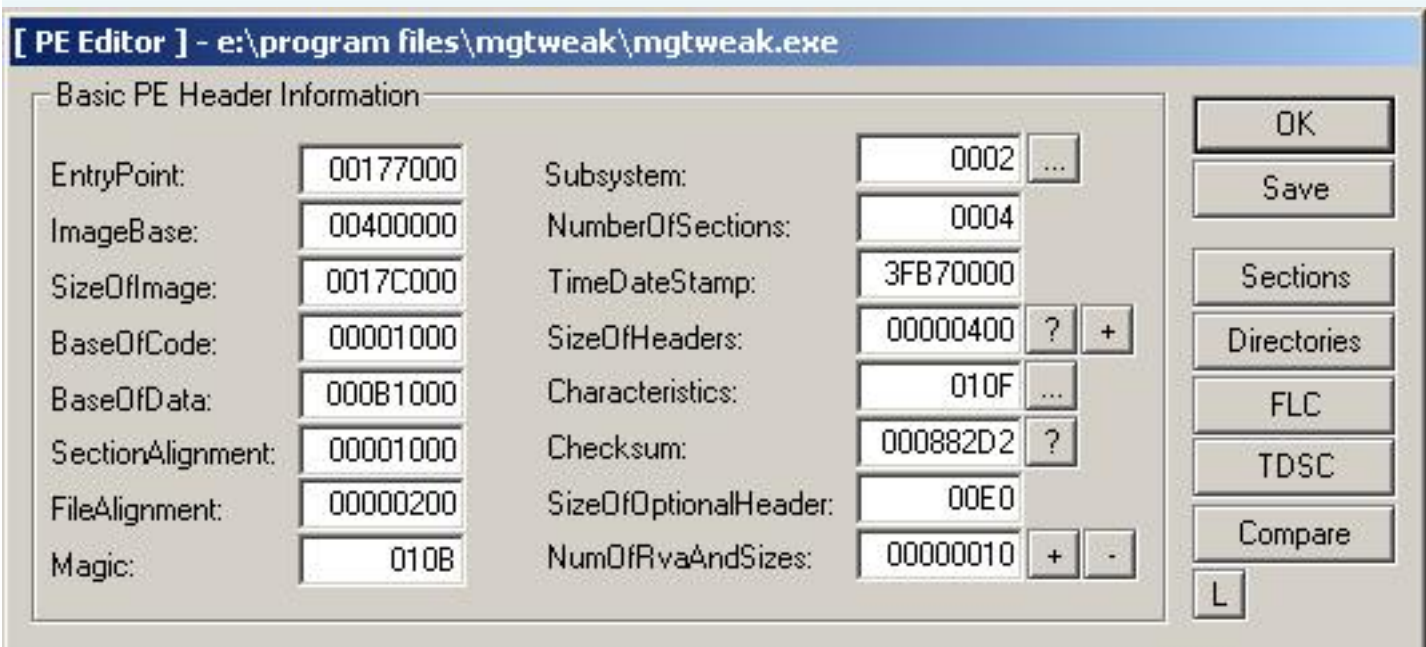
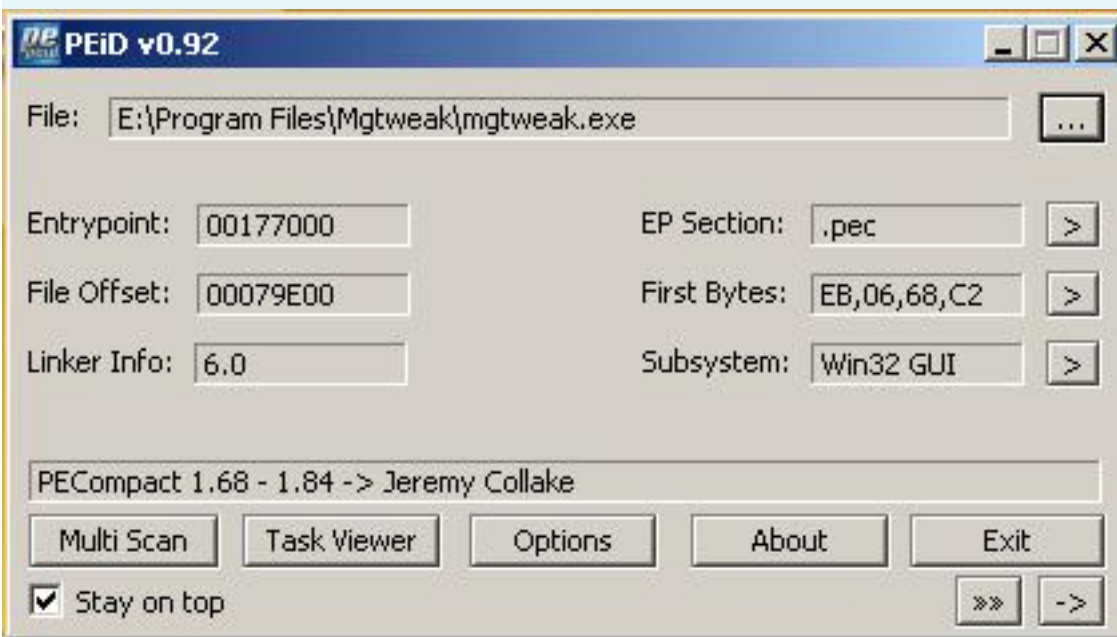
-**** **PeiD** Used to Detect **v0.92**, we know this program was the author with the Pack **PECompact 1.68 - 1.84 -> Jeremy Collake**. So it is with other Pack **thắng Magic Utilities**, but do not all problems have solutions. We will try to **unpack** this month to see why.

-**** Run test program to see what's special, immediately a Nag Screen shot out: **"Enter the registration here"** Nag Screen in this we will see two textbox to enter **User Name** and **Registration Code**. So how is the protection of the program is N / S. If you enter the correct Oki. At this time we should not have been temporarily **đánh** click on **Continue Button Evaluation** for more information.

-**** Once we click to the main screen of the program, the title bar, we see a line as follows: **Magic Tweak unregistered - Day 1 of 15**. That means if we are not legally registered we will only be used to try the features of the program within 15 days only. Too date is "he ời stay hiiiiiii them farewell." One more thing to want to say is this program of the User Interface to look very eyes looking good so it's been quite nice **\$ 29.95**.

II - UnPacking:

-**** **PeiD** Used to Detect and **LordPE**, we have some information as follows:



1. Search OEP:

-**** As we know in a PeiD Plugin "**Generic OEP Finder.**" For the Pack is in **PECompact 1.68 - 1.84 -> Jeremy Collake** this is a good plugin to find the **OEP**, use This plugin we OEP will be as follows:



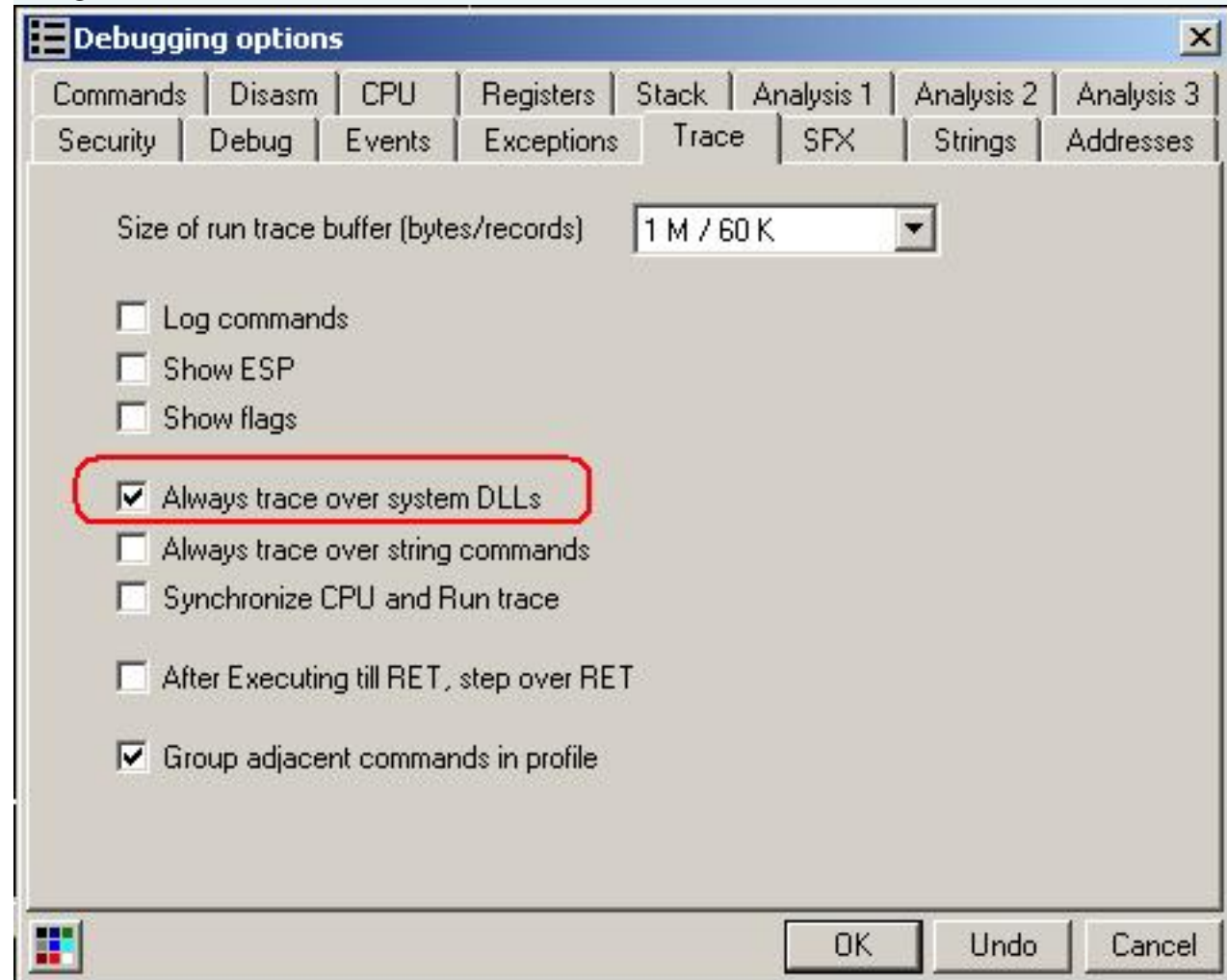
-**** Ke Ke Ke, OEP PeiD which is found **0047A6C2**. Oki, so the old formula we can find **real**

OEP as follows:

Real OEP = OEP find in PEiD-Image-47A6C2 Base = 400000 = **7A6C2**.

2.Dump file:

To -**** can dumping, we must configure the following Olly. At the main screen of Olly, click on the **Option** menu ---> **Options debugging (Alt + O)**. Transfer to **Trace** tab. Revising like the image below:



-**** Then Load files need to unpack Olly (**mgtweak.exe**), select it (**not Analysis**). We will come here:

00577000> / EB 06 JMP SHORT mgtweak.00577008; <== We're here

00577002 | 68 C2A60700 PUSH 7A6C2

00577007 | C3 RETN

00577008 \ 9C PUSHFD

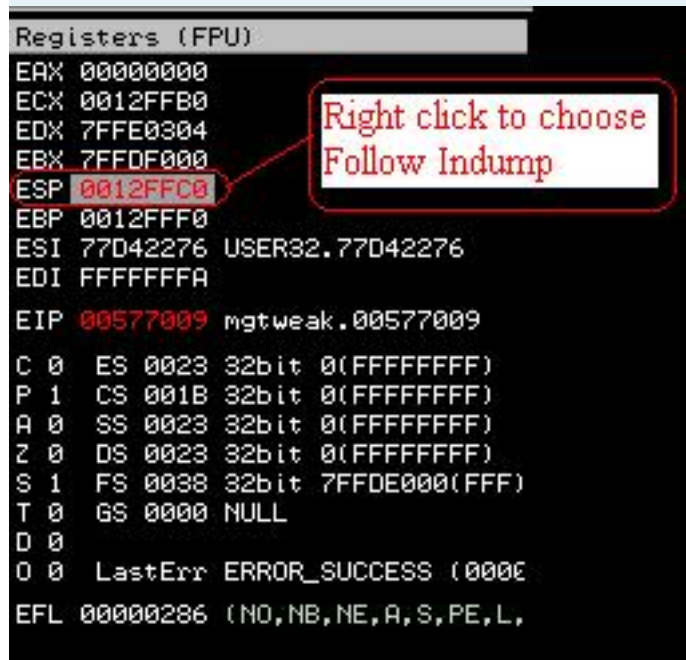
00577009 60 PUSHAD; <== **Push F8 to trace here**

0057700A E8 02000000 CALL mgtweak.00577011

0057700F 33C0 XOR EAX, EAX

-**** Press F8 to Trace to **PUSHAD** in order, after which to trace you to that window to **register**

(FPU). Click to write in to choose **ESP Follow in dump** as follows:



***** **Hex dump** window will switch to new window, select the first **4 bytes** of this window and click to select **Break ==> Hardware, on acces ==> Word**:

Address	Hex dump	ASCII
0012FFC0	86 02 00 00 69 EB E7 77 FA FF FF FF 76 22 D4 77	..isrw v"bw
0012FFD0	00 FD FD 7F 45 00 58 00 C8 FF 12 00 7B 66 53 80	.E.X. .(fSQ
0012FFE0	FF FF FF FF 86 8B E9 77 18 5A E9 77 00 00 00 00	0wZ0w....
0012FFF0	00 00 00 00 00 00 00 00 00 70 57 00 00 00 00 00pl.....

Right Click to Choose
Breakpoint---> Hardware, on
access--> Word

***** Then press **F9**, we will come here:

00578550 50 PUSH EAX; <=== We're here
00578551 68 C2A64700 PUSH mgtweak.0047A6C2
00578556 C2 0400 RETN 4

***** Press **F8** to **Trace** through **RETN4** order, we will come **OEP**:

0047A6C2 55 PUSH EBP; <=== We're here: OEP
0047A6C3 8BEC MOV EBP, ESP

0047A6C5 6A FF PUSH -1

OEP What we found is **47A6C2**. Now we will calculate the real find OEP of the program by the following formula:

Real OEP = OEP find in Olly-Image-Base 47A6C2 = 400000 = **7A6C2**. (OEP finding coincides with the PEid)

2. Dump file:

-**** At **0047A6C2** we **Plugin** to menu -> **OllyDump** -> **debugged process dump** or click to select **dump debugged process**. We will monitor to **OllyDump**:

OllyDump - mgtweak.exe

Start Address: Size:

Entry Point: -> Modify:

Base of Code: Base of Data:

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
pec1	000E1000	00001000	000E1000	00001000	E0000020
.rsrc	00095000	000E2000	00095000	000E2000	C0000040
.pec	00004000	00177000	00004000	00177000	E0000020
.rsrc	00001000	0017B000	00001000	0017B000	C0000040

☒ Rebuild Import

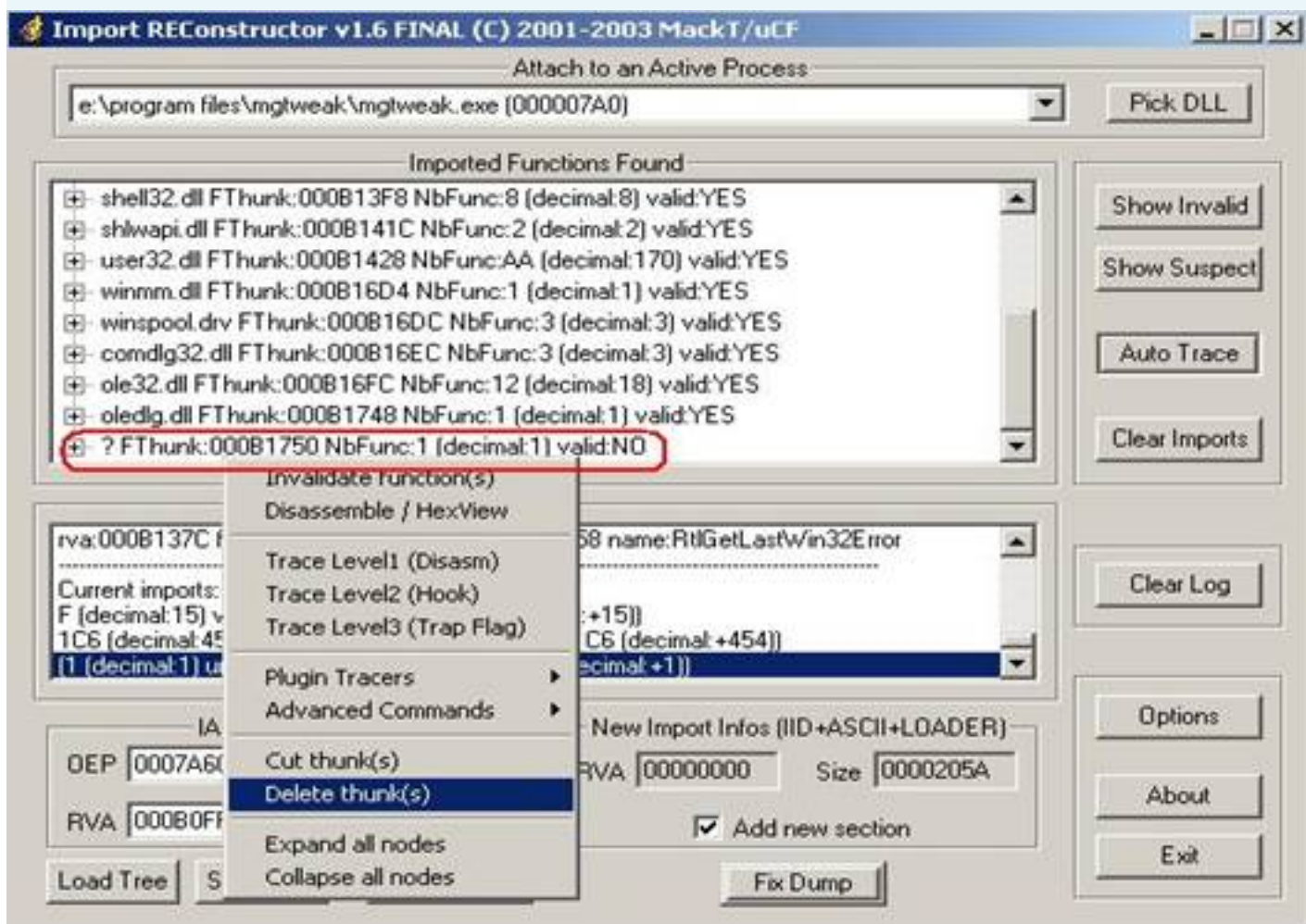
☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Check **Import** -**** **rebuild**, in the Modify the **real OEP** we have calculated according to the formula above. Click **dump**, Save us with a name any. Here you save is **dumped.exe**.

3. Find and Edit IAT (Import Adress Table):

-**** Retain the program, open the **Import REConstructor v1.6F** load file **mgtweak.exe**. Instead of using the OEP in the value we have found and calculated in the (**7A6C2**) then select **IAT AutoSearch** and then click **Get Imports**. We will be as follows:

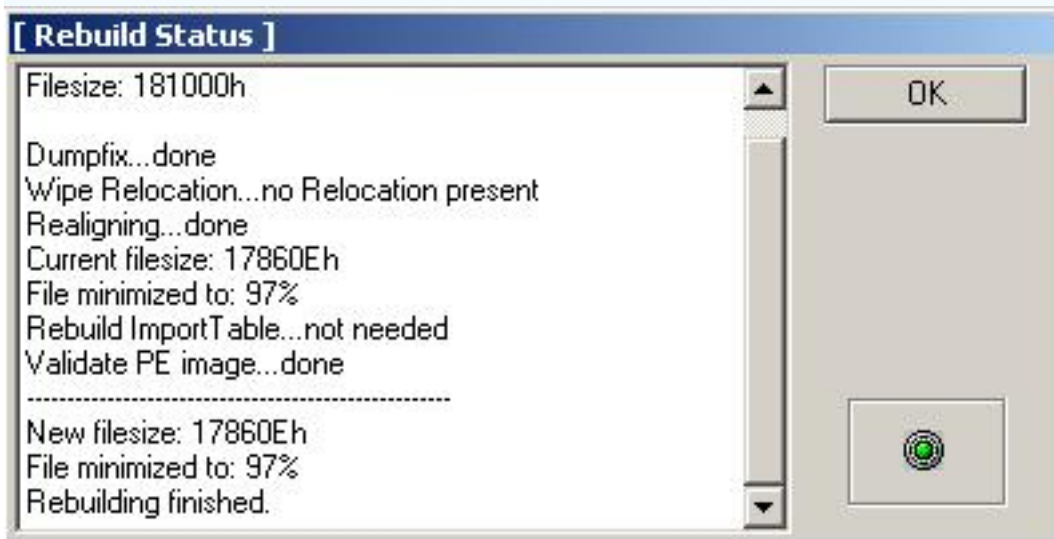


A -**** **Import Function Invalid**, right click in the **Function** and select **Delete Thunk (s)**. Now we press **Fix Fix IAT** to **dump** file for **dumped.exe** that we save on.

4. Cleaning and reduce the size of the file after unpack:

To improve -**** than we conducted the process of cleaning and reduce the file size of the file after **Fix dump** (the purpose of making the file as small as possible). If you do not like you can step over.

Using -**** **LordPE v1.4**. Load up this program, select **Rebuilt PE**. Then select the file that we **Fix dump** (the file **dumped_.exe**). The we have a complete new file which you then. Ặ ặ too tired but ultimately that's the next, we will become easier.



Using -**** **PeiD v0.92** detect again we know this program is the author code with **Microsoft Visual C + + 6.0**. See that the eyes of the right, no child left eye too hiiiiiiiiiii

Homepage: <http://www.magictweak.com>

Production: Efreesky Software

Software: **Magic Utilities 2004 v3.10**

Copyright by: Copyright (C) 2000 - 2004 Efreesky Software. All Rights Reserved.

Type: N / S

Packed: **PECompact 2.x -> Jeremy Collake**

Language: Microsoft Visual C + + 6.0

Crack Tool: 1:10 OllyDbg, PEiD 0.92, Import REConstructor v1.6F, LordPE v1.4

Unpack: N / A

Request: Correct Serial / KeyGen

Comments:

Magic Utilities 2004 v3.10

Magic Utilities is a cute program designed to make your computer clean and more stable. These utilities include Uninstller Plus, Startup Organizer, Process Killer. Magic Utilities enables you to easily and safely uninstall programs, inspect and manage the programs that start automatically when you turn on or logon to your computer, list and control all currently running processes (system and hidden processes are also shown). With a cool and user-friendly interface makes it easy for anyone to use Magic Utilities.

I - Information:

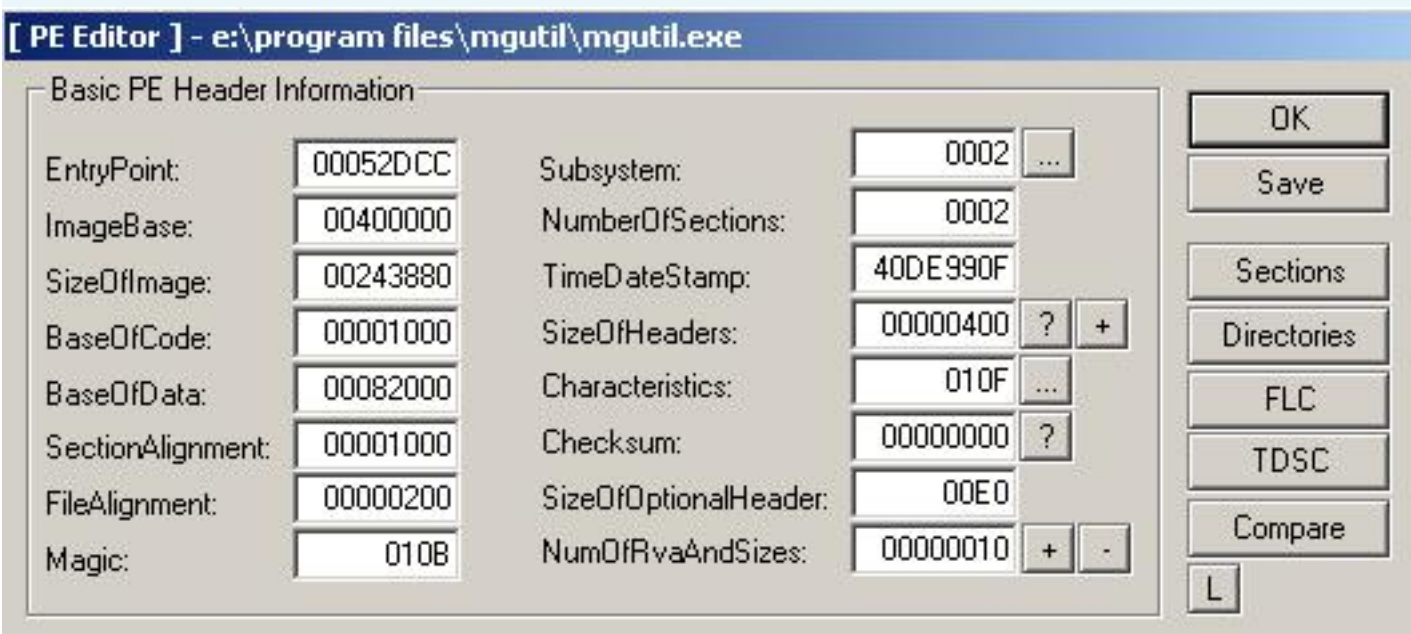
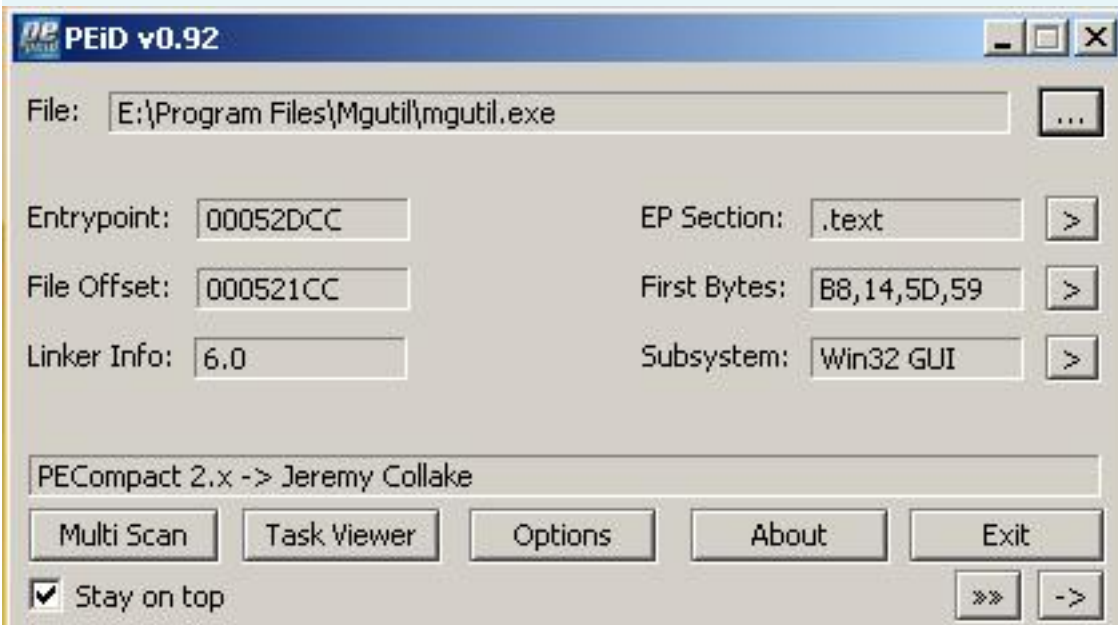
-**** **PeiD** Used to Detect **v0.92**, we know this program was the author with the Pack **PECompact 2.x -> Jeremy Collake**. What is difficult for us then, but do not have a hard new lo the wise. We will try to unpack this month to see why.

-**** Run test program to see what's special, immediately a Nag Screen shot out: **"Enter the registration here"** Nag Screen in this we will see two textbox to enter **User Name** and **Registration Code**. So how is the protection of the program is N / S. If you enter the correct Oki. At this time we should not have been temporarily đành click on **Continue** Button **Evaluation** for more information.

-**** Once we click to the main screen of the program, the title bar, we see a line as follows: **unregistered Magic Utilities 2004 - Day 1 of 15**. That means if we do not register legally we will only be used to try the features of the program within 15 days only. Too date is "he oi stay hiiiiiii them farewell." One more thing to want to say is this program of the User Interface to look very eyes looking good so it's been quite nice **\$ 29.95**.

II - UnPacking:

-**** **PeiD** Used to Detect and **LordPE**, we have some information as follows:



1. Search OEP:

-**** Load program in Olly, select it (not Analysis). We will come here:

00452DCC> b8 145D5900 MOV EAX, mgutil.00595D14; <== We're here

00452DD1 50 PUSH EAX

00452DD2 64: FF35 00000000> PUSH DWORD PTR FS: [0]

00452DD9 64:8925 00000000> MOV DWORD PTR FS: [0], ESP

00452DE0 33C0 XOR EAX, EAX

00452DE2 8908 MOV DWORD PTR DS: [EAX], ECX

00452DE4 50 PUSH EAX

45 INC EBP 00452DE5

-**** Press F9 twice, we'll come to the following order:

00452DE2 8908 MOV DWORD PTR DS: [EAX], ECX ;<=== We're here

00452DE4 50 PUSH EAX

45 INC EBP 00452DE5

00452DE6 43 INC EBX

00452DE7 6F OUTS DX, DWORD PTR ES: [EDI], I / O command

-**** Here we hit the combination **Alt + M** to open the window in Olly **Memory Map**:

Memory map

Address Size Owner Section Contains Type Access Initial Mapped as

00360000 00001000 Priv RW RW

003E0000 00004000 Priv RW RW

003F0000 00002000 Map R R

00400000 00001000 mgutil PE header IMAG R RWE

00401000 0016B000 mgutil. IMAG text code R RWE

0056C000 000D8000 mgutil. Rsrc imports, IMAG reso R RWE

00650000 00003000 Map RE RE

00710000 00002000 Map RE RE

At -**** red line above, we have to click and then select **Set breakpoint on memory access**.

-**** Then press **Shift + F9** we will code to the following:

00595D29 C602 E9 MOV BYTE PTR DS: [EDX], 0E9 ;<=== We're here

00595D2C 83C2 05 ADD EDX, 5

00595D2F 2BCA SUB ECX, EDX

00595D31 894A FC MOV DWORD PTR DS: [EDX-4], ECX

00595D34 33C0 XOR EAX, EAX

00595D36 C3 RETN

-**** Next they hit **Shift + F9** to 3 times more, we'll come to a different code as follows:

00595D37 b8 BD4CCDFF MOV EAX, FFCD4CBD; <=== We're here

00595D3C 64:8 F05 00000000> POP DWORD PTR FS: [0]

00595D43 83C4 04 ADD ESP, 4

00595D46 55 PUSH EBP

00595D47 53 PUSH EBX

-**** Here we press **Ctrl + F12**, we'll see the following code:

00B50289 F3: A4 REP MOVSB BYTE PTR ES: [EDI], BYTE PTR DS: [> ;<=== We're here, very close to OEP rùi you ời.

00B5028B 8BF3 MOV ESI, EBX

00B5028D 8D8D D7128C00 LEA ECX, DWORD PTR SS: [EBP +8 C12D7]

00B50293 51 PUSH ECX

00B50294 E8 4D020000 CALL 00B504E6

00B50299 8B4E 2C MOV ECX, DWORD PTR DS: [ESI +2 C]

00B5029C 8B56 24 MOV EDX, DWORD PTR DS: [ESI +24]

00B5029F 0356 08 ADD EDX, DWORD PTR DS: [ESI +8]

-**** Finally we press **Ctrl + F12** again, OEP will appear right before our eyes:

00452DCC> 55 PUSH EBP <=== Our OEP is 00452DCC

00452DCD 8BEC MOV EBP, ESP

00452DCF 6A FF PUSH -1

00452DD1 68 908D4800 PUSH mgutil.00488D90

00452DD6 68 488D4500 PUSH mgutil.00458D48

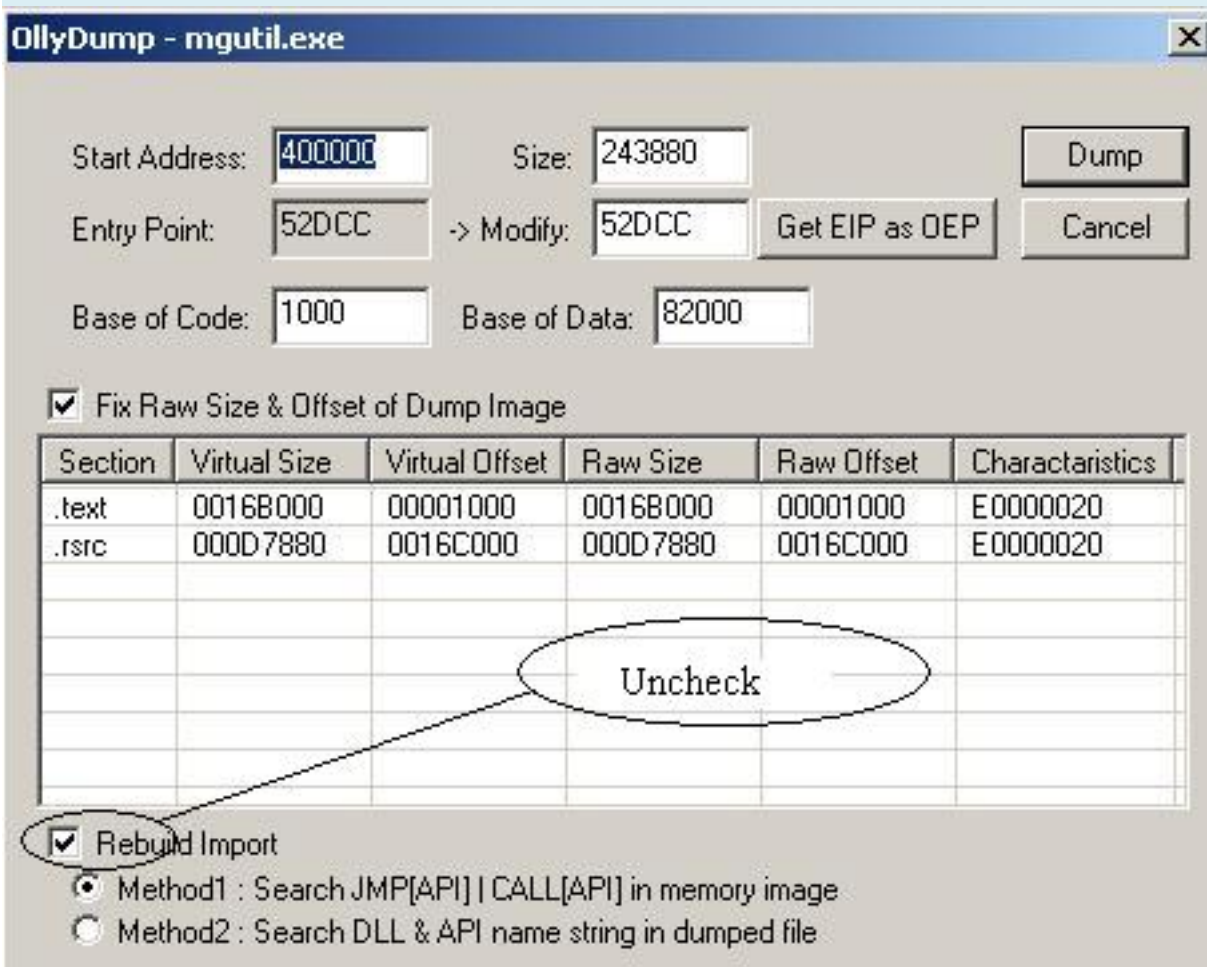
00452DDB 64: A1 00000000 MOV EAX, DWORD PTR FS: [0]

OEP What we found is **452DCC**. Now we will calculate the real find OEP of the program by the following formula:

Real OEP = OEP find in Olly-Image-Base 452DCC = 400000 = **52DCC**.

2.Dump file:

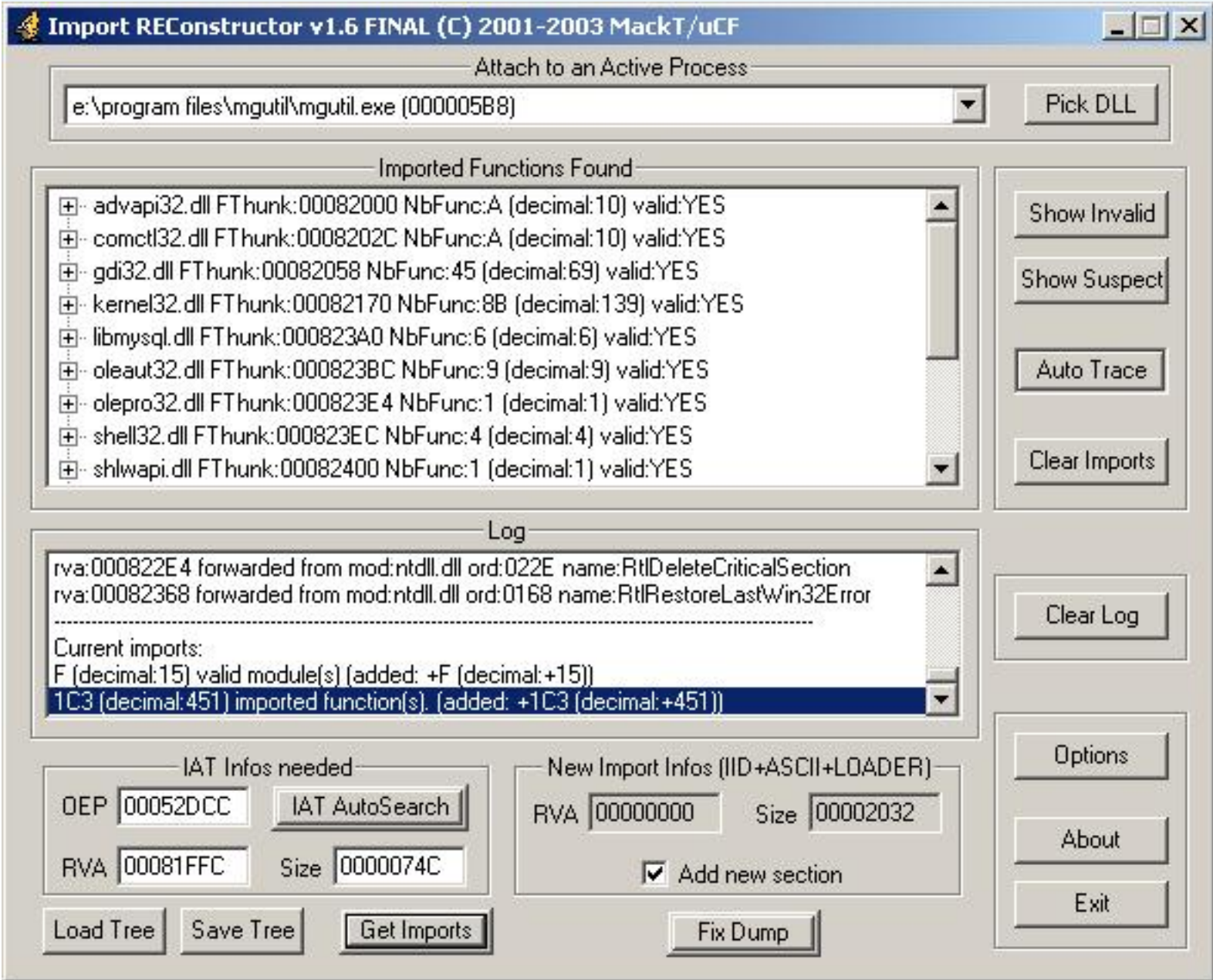
-**** At **00452DCC** we **Plugin** to menu -> **OllyDump -> debugged process dump** or click to select **dump debugged process**. We will monitor to **OllyDump**:



Uncheck **Import -**** rebuild**, in the Modify the real OEP we have calculated according to the formula above. Click dump, Save us with a name any. Here you save is **dumped.exe**.

3. Find and Edit IAT (Import Adress Table):

-**** Retain the program, open the **Import REConstructor v1.6F** load file **mgutil.exe**. Instead of using the OEP in the value we have found and calculated in the (52DCC) then select **IAT AutoSearch** and then click **Get Imports**. We will be as follows:

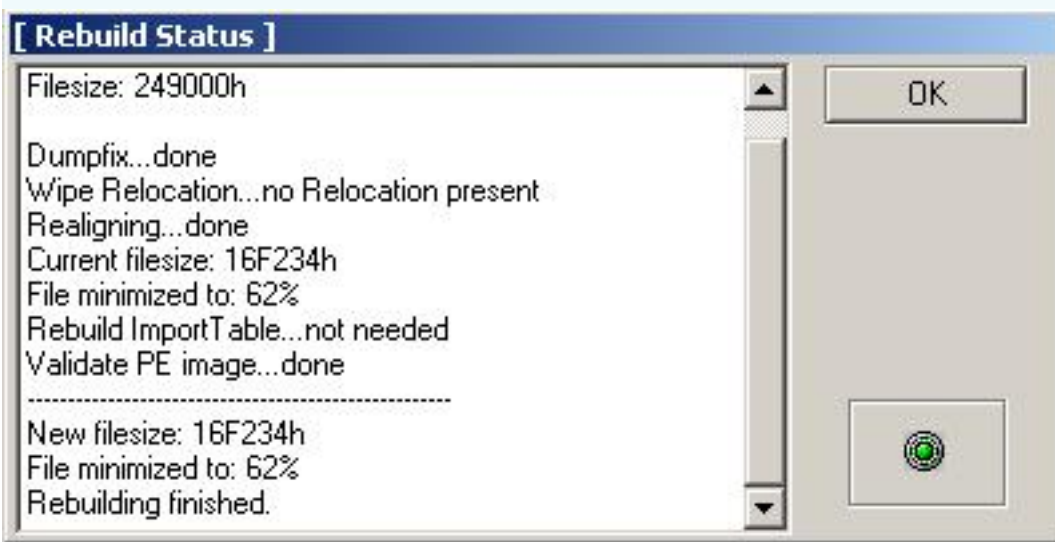


****** All Import Function Valid** all too has run. Now we press **Fix Fix IAT** to **dump** file for **dumped.exe** that we save on.

4. Cleaning and reduce the size of the file after unpack:

To improve -**** than we conducted the process of cleaning and reduce the file size of the file after **Fix dump** (the purpose of making the file as small as possible). If you do not like you can step over.

Using -**** **LordPE v1.4**. Load up this program, select **Rebuilt PE**. Then select the file that we **Fix dump** (the file **dumped.exe**). The we have a complete new file which you then.Ặặặ too tired



Using -**** **PeiD v0.92** detect again we know this program is the author code with **Microsoft Visual C + + 6.0**. See that the eyes of the right, no child left eye too hiiiiiiiiiii

Manual unpack PESpin by 0.7 Kagra

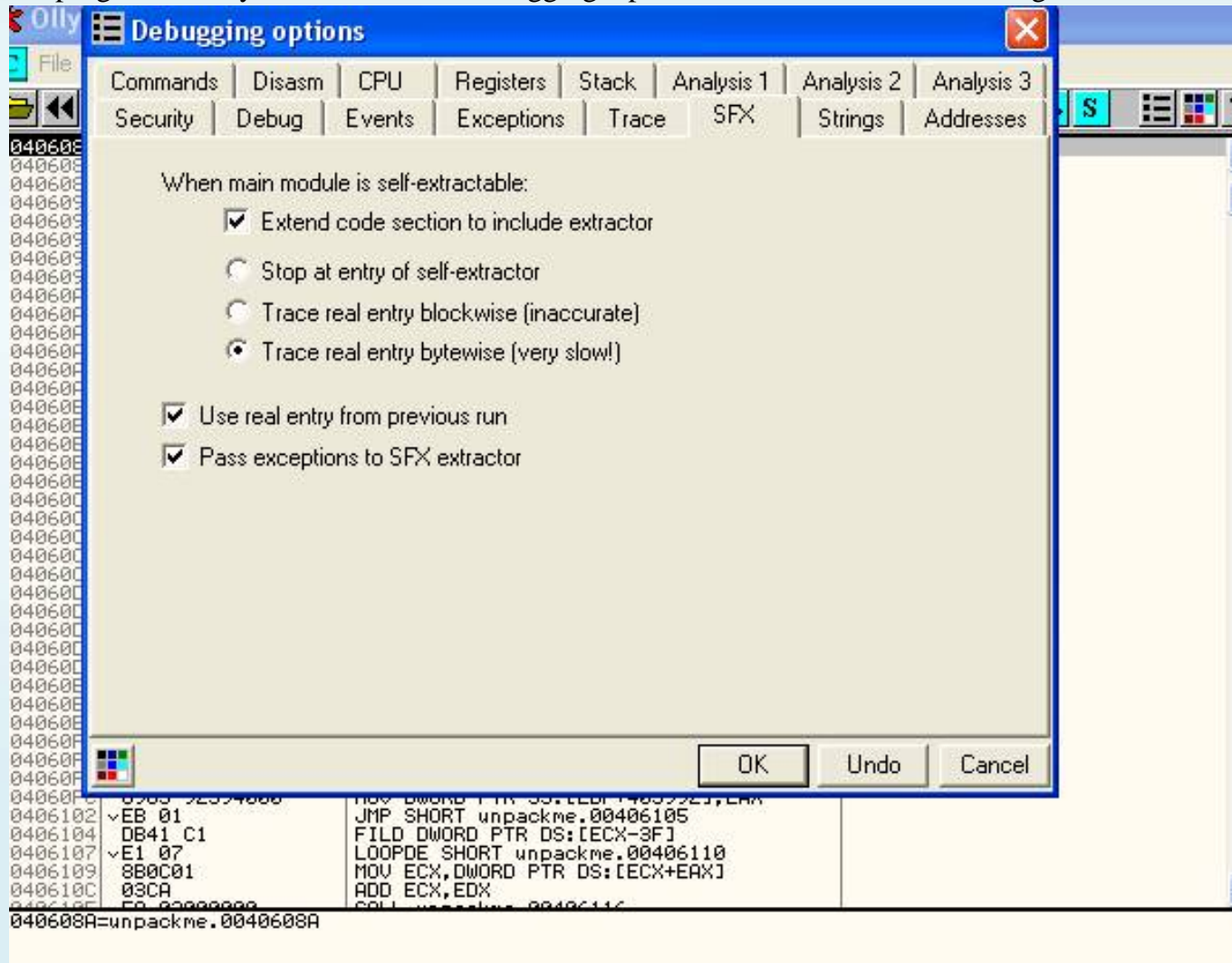
Translate and Edit: tlandn

Tut this Kagra your bags. Tut Kagra's short and does not explain clearly quite difficult to understand. I add images, and notes for you to understand more.

Target: Unpackme? by hacnho (enclosed in the file)

First you have to go hide OllyDbg. Kagra use HideOlly Plugin but you can also use the plugin IsDebugPresent.

Load program to Olly. Press Alt-O to "debugging Options" and the same as following:

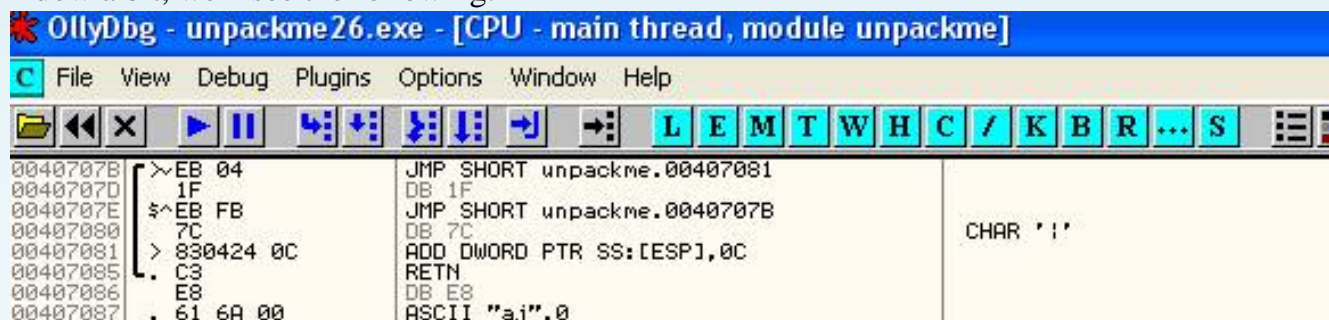


Now load the program by pressing Ctrl-F2. The error message just click on the OK.

Press Ctrl-F9. The program will stop at 00401029. Press Ctrl-A to "analyze ...". A notice on the select YES. Now look down below the window will see the following:

```
EBP=0012FD8C
Jump from 004070DC
```

That is we have to jump 00401029 from 004070DC. Press Ctrl-G. Type in 004070DC. Click OK. Now roll up the window a bit, we'll see the following:

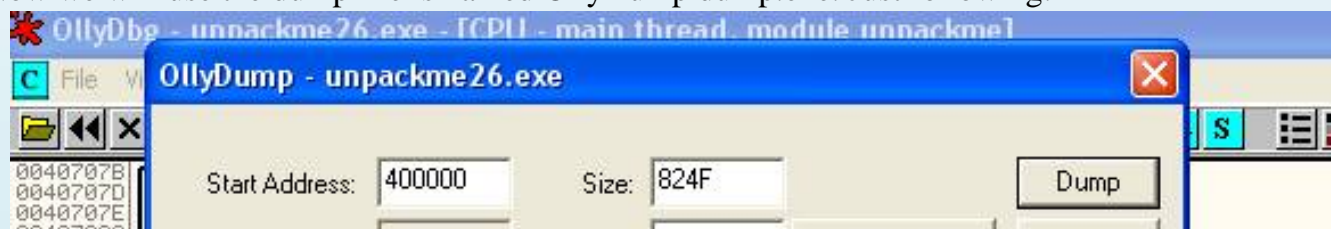


00407085	L. C3	RETN	
00407086	E8	DB E8	
00407087	. 61 6A 00	ASCII "aj",0	
0040708A	EB	DB EB	
0040708B	> 01E3	ADD EBX,ESP	
0040708D	. 68 97704000	PUSH unpackme.00407097	
00407092	.-E9 7DA0FFFF	JMP unpackme.00401114	
00407097	.A3 00304000	MOV DWORD PTR DS:[403000],EAX	
0040709C	.-EB 01	JMP SHORT unpackme.0040709F	
0040709E	. F1	INT1	
0040709F	> 6A 00	PUSH 0	
004070A1	.-EB 01	JMP SHORT unpackme.004070A4	
004070A3	. 45	INC EBP	
004070A4	> 68 F5EED209	PUSH 9D2EEF5	
004070A9	. 812C24 CCDE9209	SUB DWORD PTR SS:[ESP],992DECC	
004070B0	. 6A 00	PUSH 0	
004070B2	.-EB 01	JMP SHORT unpackme.004070B5	
004070B4	. 1C	DB 1C	
004070B5	> 6A 65	PUSH 65	
004070B7	.-EB 01	JMP SHORT unpackme.004070BA	
004070B9	. 81	DB 81	
004070BA	> FF35 00304000	PUSH DWORD PTR DS:[403000]	unpackme.00400000
004070C0	.-EB 01	JMP SHORT unpackme.004070C3	
004070C2	. DC	DB DC	
004070C3	> 68 CD704000	PUSH unpackme.004070CD	
004070C8	.-E9 53A0FFFF	JMP unpackme.00401120	
004070CD	. 6A 00	PUSH 0	
004070CF	.-EB 01	JMP SHORT unpackme.004070D2	
004070D1	. 27	DAA	
004070D2	> 68 DC704000	PUSH unpackme.004070DC	
004070D7	.-E9 32A0FFFF	JMP unpackme.0040110E	
004070DC	.-E9 489FFFFF	JMP unpackme.00401029	
004070E1	. 5C	DB 5C	CHAR '\'
004070E2	. 47	DB 47	CHAR 'G'
004070E3	. 0C	DB 0C	

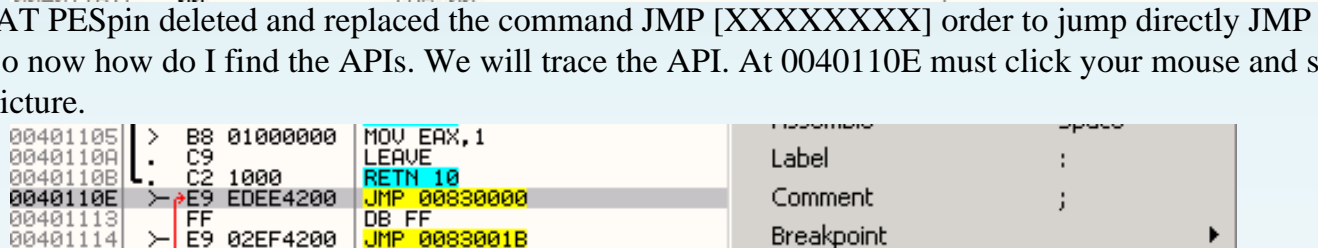
File we start with **PUSH 0 (6A 00)**. We found a little under 00407085 in two **00407088** bytes (6A 00). This is our OEP. Press Ctrl-G. Enter **00407088**. Click OK.

Address	Disassembly	Comment
00407088	PUSH 0	
0040708A	JMP SHORT dumped_.0040708D	
0040708C	JECXZ SHORT dumped_.004070F6	
0040708E	XCHG EAX,EDI	
0040708F	JO SHORT dumped_.004070D1	
00407091	ADD CL,CH	
00407093	JGE SHORT dumped_.00407035	
00407095	???	Unknown command
00407097	MOV DWORD PTR DS:[403000],EAX	
0040709C	JMP SHORT dumped_.0040709F	
0040709E	INT1	
0040709F	PUSH 0	
004070A1	JMP SHORT dumped_.004070A4	
004070A3	INC EBP	
004070A4	PUSH 9D2EEF5	
004070A9	SUB DWORD PTR SS:[ESP],992DECC	
004070B0	PUSH 0	
004070B2	JMP SHORT dumped_.004070B5	
004070B4	SBB AL,6A	
004070B6	JMP SHORT dumped_.004070BA	Superfluous prefix
004070B9	CMP EDI,40300035	
004070BF	ADD BL,CH	
004070C1	ADD ESP,EBX	
004070C3	PUSH dumped_.004070CD	
004070C8	JMP <JMP.&user32.DialogBoxParamA>	
004070CD	PUSH 0	
004070CF	JMP SHORT dumped_.004070D2	
004070D1	DAA	
004070D2	PUSH dumped_.004070DC	
004070D7	JMP <JMP.&kernel32.ExitProcess>	
004070DC	JMP dumped_.00401029	
004070E1	POP ESP	
004070E2	INC EDI	
004070E3	MOV WORD PTR DS:[EAX],DS	Modification of segment register
004070E5	MOV GS,WORD PTR DS:[EBX]	
004070E7	CALL 276EE4E4	
004070EC	INC ESP	
004070ED	PUSH ESI	
004070EE	OUT DX,EAX	I/O command
004070FF	FIPOD 000000 PTR DS:[00000000]	

Now we will use the dump file is named OllyDump dump.exe. Just following:

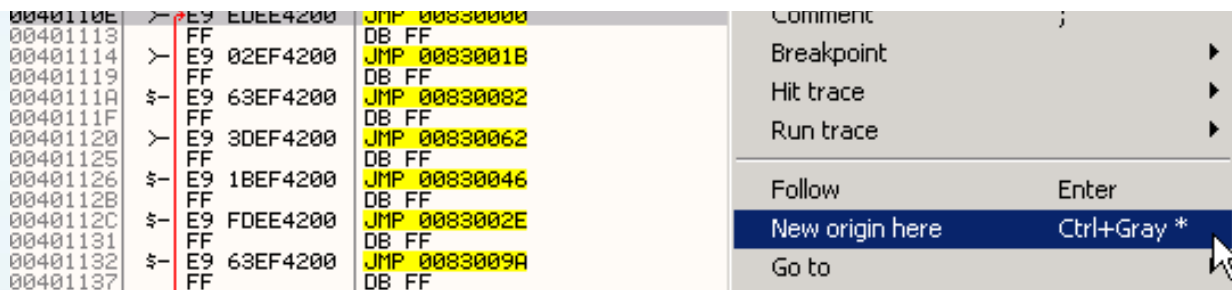


5	32A0FFFF	JMP unpackme.0040110E	
9	489FFFFF	JMP unpackme.00401029	
C		DB 5C	CHAR '\'
7		DB 47	CHAR 'G'

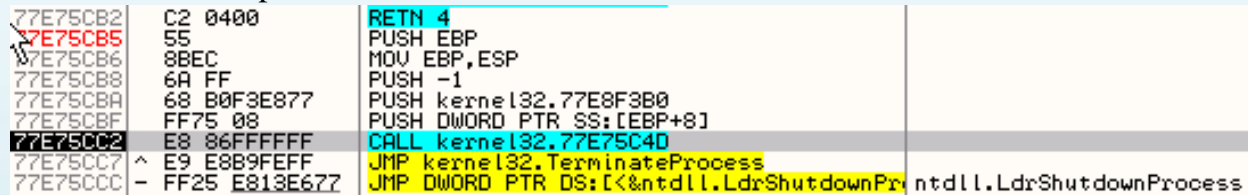


XXXXXXXXXX.
lect as in the

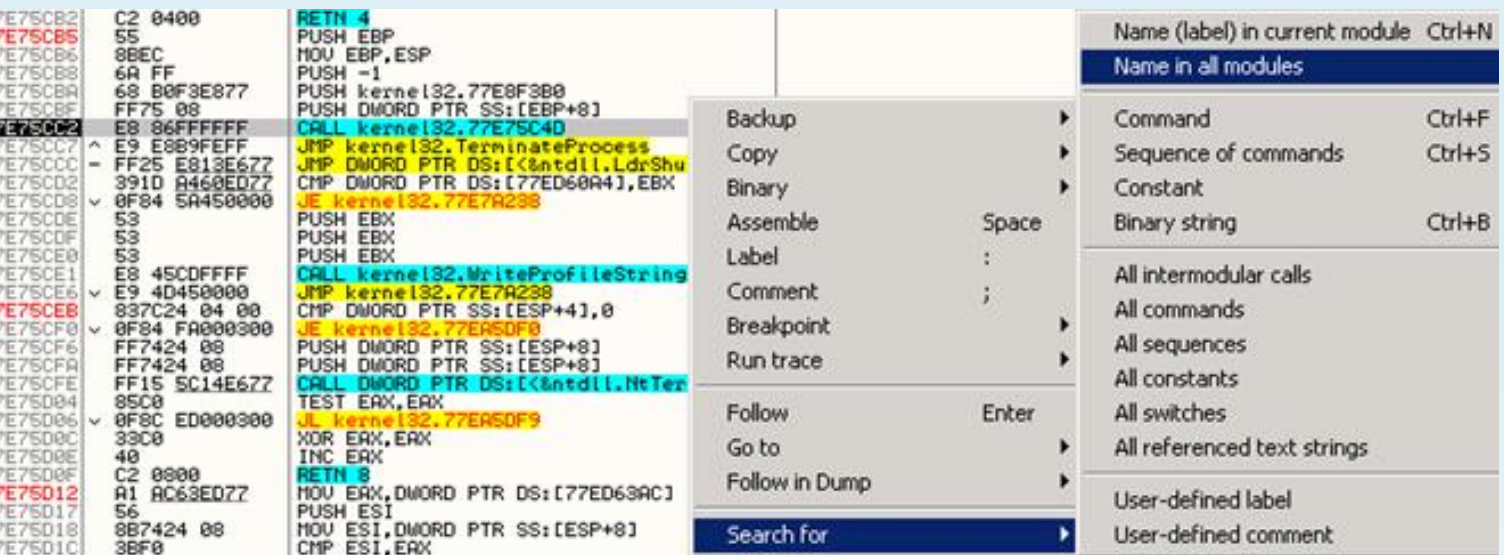
Address	Disassembly	Comment	Breakpoint
00401105	> B8 01000000 MOV EAX,1		
0040110A	: C9 LEAVE		
0040110B	: C2 1000 RETN 10		
0040110E	Y E9 EDEE4200 JMP 00830000		
00401113	: FF DB FF		
00401114	Y E9 02EF4200 JMP 0083001B		



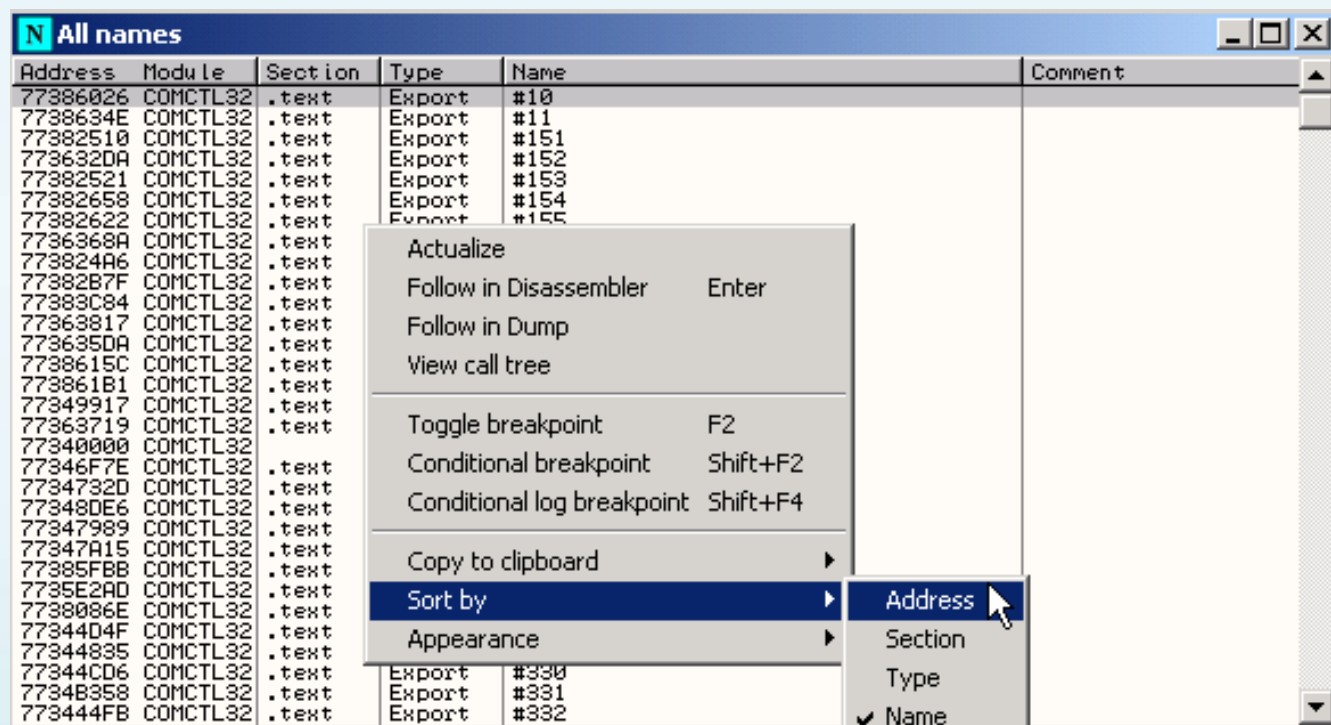
We will in 0040110E. Now trace the F7 until we address in the form 77XXXXXX. On bags, I will stop at 77E75CC2. Scroll up a screen hairbreadth. We found as follows:



Note the starting point is the API 77E75CB5 (red in the picture above). Now click the mouse to select as in the picture.



In the window "All Names". Click your mouse to select the image in order to organize the API in order to address to search.



773444FB COMCTL32|.text |Export |#332

Name

Now scroll down window below the search **77E75CB5** address.

77E752B8	kernel32 .text	Export	IsValidLocale
77E75CB5	kernel32 .text	Export	ExitProcess
77E75CEB	kernel32 .text	Export	TerminateThread

Also, so we are aware of this API is ExitProcess in Kernel32.dll.

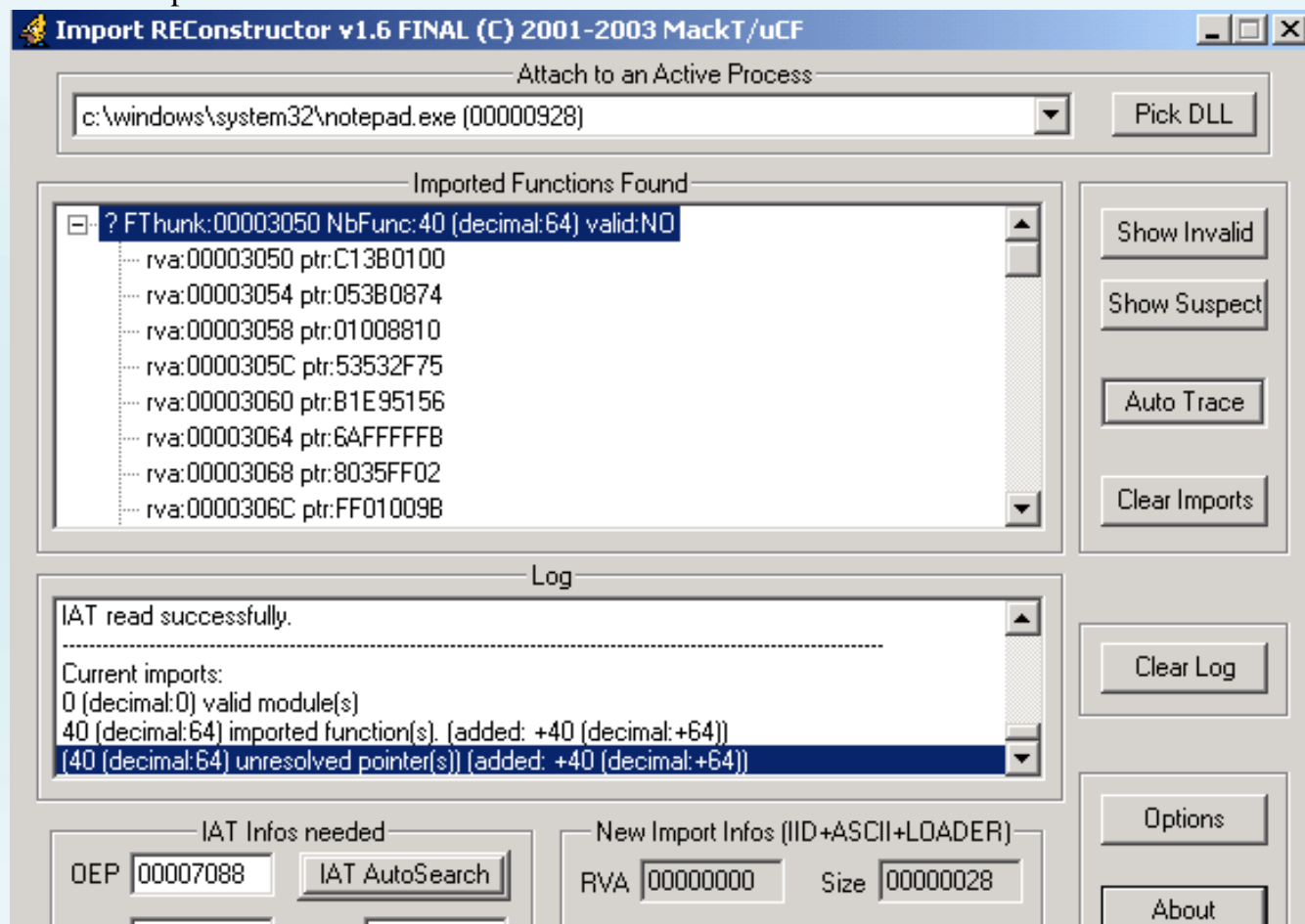
For other APIs, we also do the same (00401114, 0040111A,, 00401156). Also select "New Origin Here" to address remaining and trace with F7.

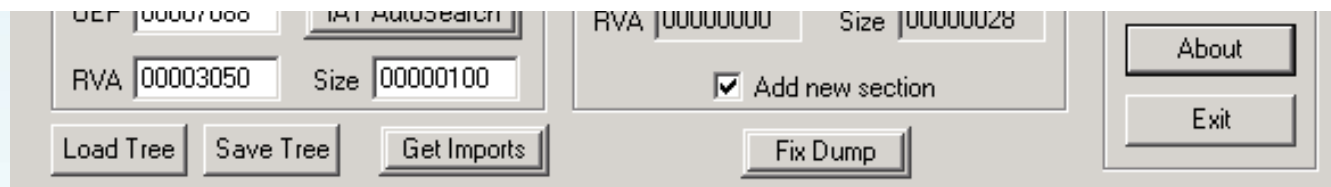
Finally we have a table summarizing the following:

0040110E	ExitProcess	Kernel32.dll
00401114	GetModuleHandleA	Kernel32.dll
0040111A	BeginPaint	User32.dll
00401120	DialogBoxParamA	User32.dll
00401126	EndDialog	User32.dll
0040112C	EndPaint	User32.dll
00401132	LoadBitmapA	User32.dll
00401138	SendMessageA	User32.dll
0040113E	BitBlt	Gdi32.dll
00401144	CreateCompatibleDC	Gdi32.dll
0040114A	DeleteDC	Gdi32.dll
00401150	DeleteObject	Gdi32.dll
00401156	SelectObject	Gdi32.dll

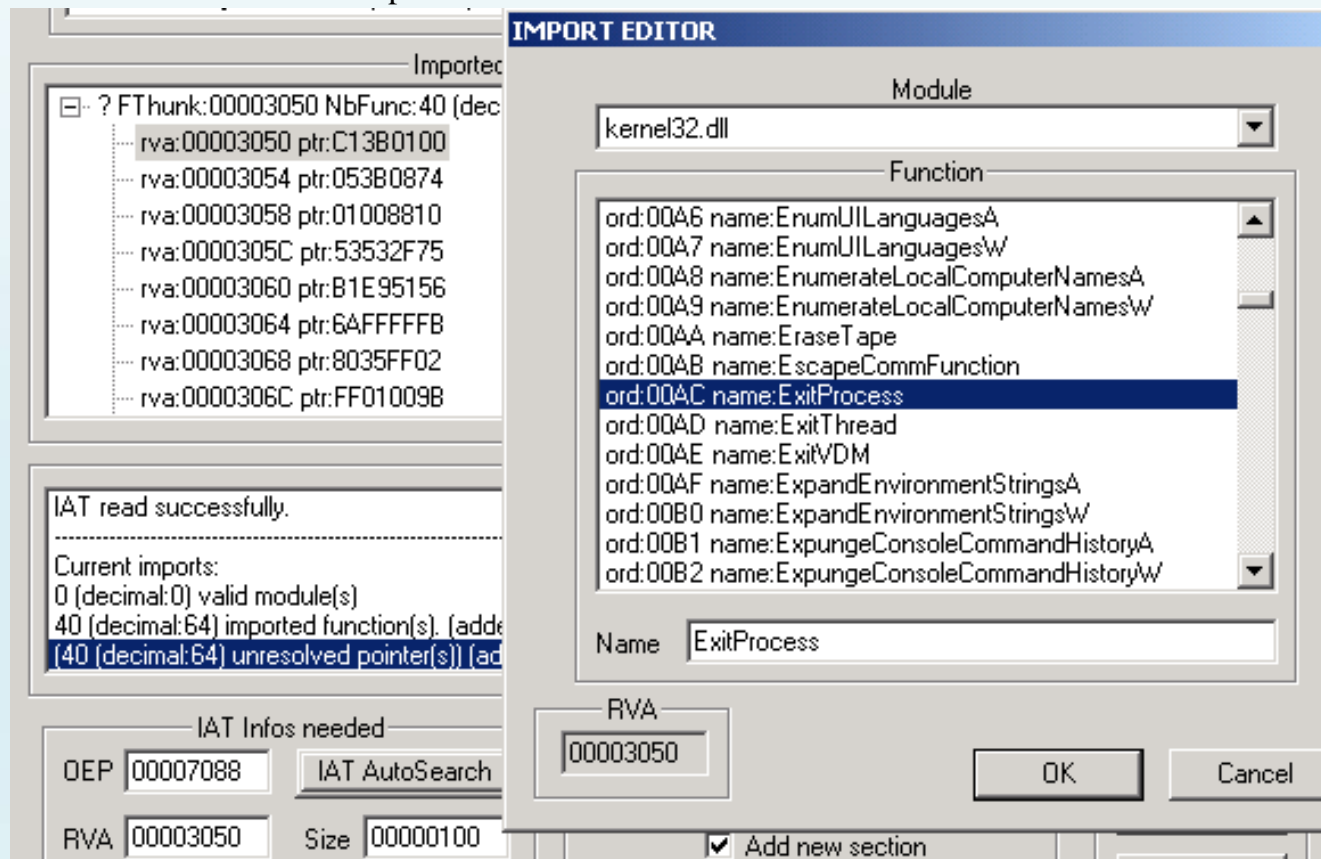
Now we will make a space in the file dump.exe to insert this in the API. Using any program Hex Edit any open files and search dump.exe space (multiple-byte 00). Here I found **00403050** many empty seats. We will insert it in the API.

Running Imprec. Select any of the process will be. Here I choose Notepad. Enter the same number in the picture. Click "Get Imports". We are:



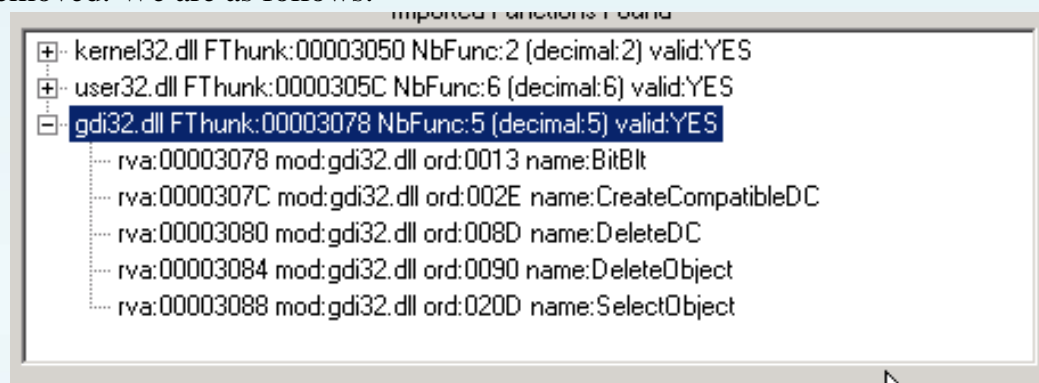


Now we will enter into the API. Double-click the mouse on the "rva: 00003050 ptr: C13B0100". And to complete the API ExitProcess as in the picture:



At a function rva 00003054 GetModuleHandleA. In rva 00003058 we will cut it out (by the end of the API in kernel32.dll by mouse click to select "Cut Thunk (s)")

Do the same for the rest of the API (at 00003074 removed). The rva remaining after complete all API also removed. We are as follows:



Click "Fix dump". Dump.exe Select File, we are dump_.exe file. IAT will be formatted as follows:

Target: C: \ WINDOWS \ system32 \ NOTEPAD.EXE

OEP: 00007088 IAT RVA: 00003050 IAT Size: 00000100

FThunk: 00003050 NbFunc: 00000002

1 00003050 kernel32.dll ExitProcess 00AC

1 00003054 kernel32.dll 0168 GetModuleHandleA

FThunk: 0000305C NbFunc: 00000006

1 0000305C user32.dll 000E BeginPaint

1 00003060 user32.dll 009F DialogBoxParamA

1 00003064 user32.dll 00C7 EndDialog

1 00003068 user32.dll 00C9 EndPaint

1 0000306C user32.dll 01B6 LoadBitmapA

1 00003070 user32.dll 023C SendMessageA

FThunk: 00003078 NbFunc: 00000005

1 00003078 gdi32.dll 0013 BitBlt

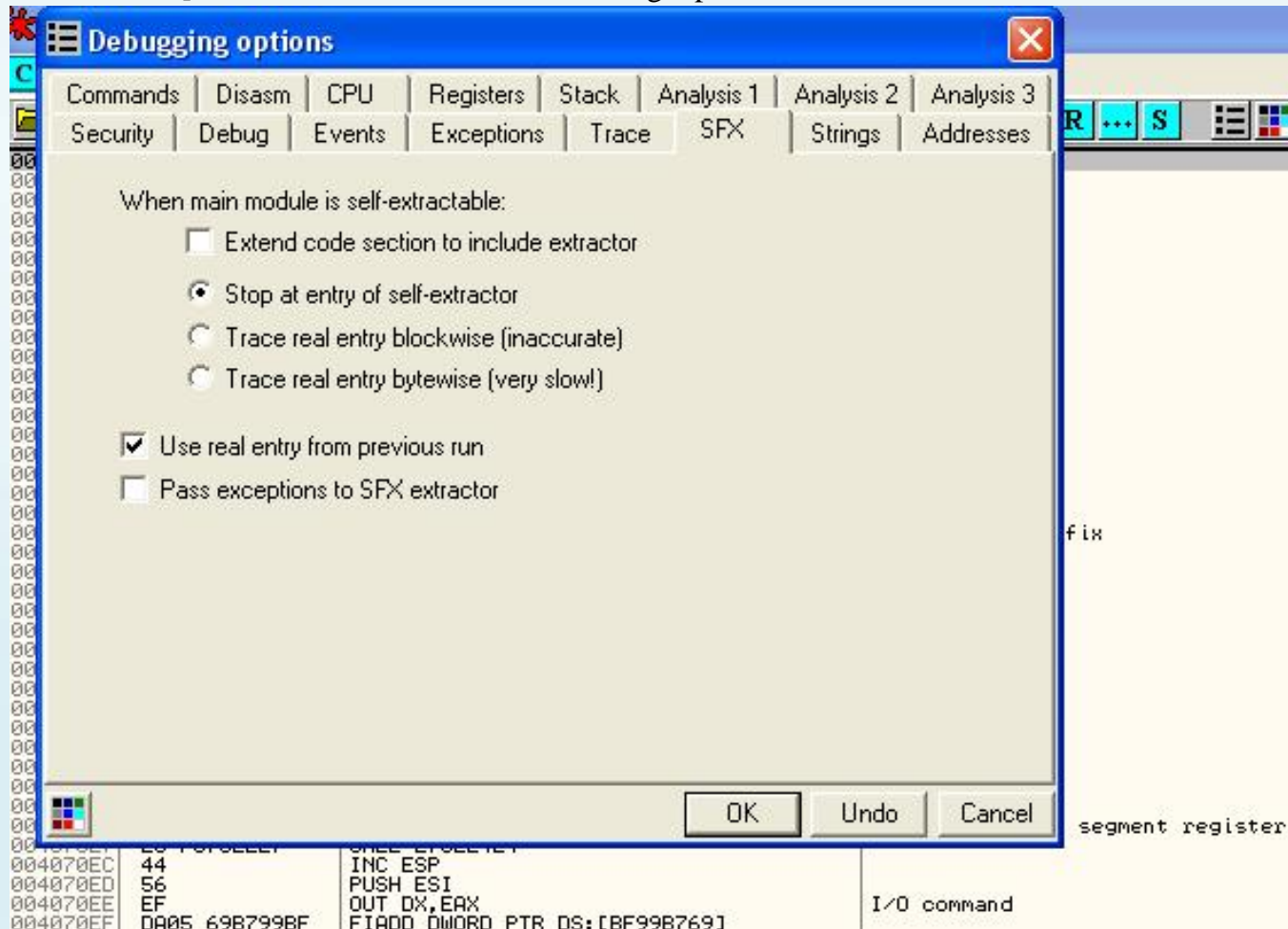
1 0000307C gdi32.dll 002E CreateCompatibleDC

1 00003080 gdi32.dll 008D DeleteDC

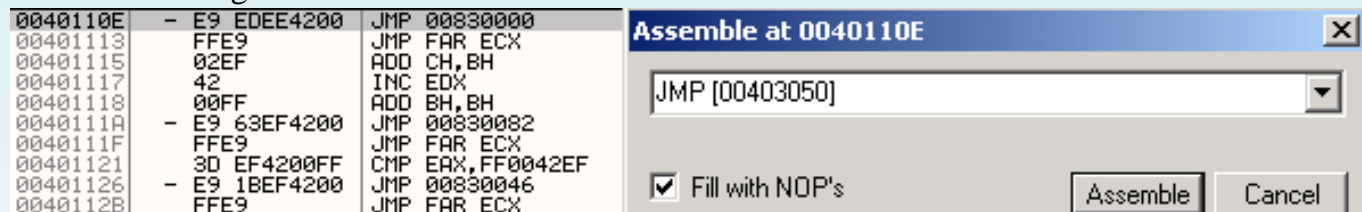
1 00003084 gdi32.dll 0090 DeleteObject

1 00003088 gdi32.dll 020D SelectObject

We need a more minor changes to IAT work. We will address the JMP YYYYYYYYYY and fix the JMP [XXXXXXXX]. But first we will revise the "Debug Options" as follows



Open file dump_.exe. Jump to 0040110E, we will revise the JMP 00830000 by pressing keys in Space 0040110E. Enter the following:




```

00401121 3D EF4200FF CMP EAX,FF0042EF
00401126 - E9 1BEF4200 JMP 00830046
0040112B FFE9 JMP FAR ECX
0040112D FD STD
0040112F FF

```

☒ Fill with NOP's

Assemble

Cancel

Click "assemble" (Note the above is 00403050 is the address of ExitProcess function in our IAT). The API also do the same. Finally we are:

Address	Disassembly	Comment
0040110B	C2 1000	RETN 10
0040110E	FF25 50304000	JMP DWORD PTR DS:[<&kernel32.ExitProcess>]
00401114	FF25 54304000	JMP DWORD PTR DS:[<&kernel32.GetModuleHandleA>]
0040111A	FF25 5C304000	JMP DWORD PTR DS:[<&user32.BeginPaint>]
00401120	FF25 60304000	JMP DWORD PTR DS:[<&user32.ShowDialogParamA>]
00401126	FF25 64304000	JMP DWORD PTR DS:[<&user32.EndDialog>]
0040112C	FF25 68304000	JMP DWORD PTR DS:[<&user32.EndPaint>]
00401132	FF25 6C304000	JMP DWORD PTR DS:[<&user32.LoadBitmapA>]
00401138	FF25 70304000	JMP DWORD PTR DS:[<&user32.SendMessageA>]
0040113E	FF25 78304000	JMP DWORD PTR DS:[<&gdi32.BitBlt>]
00401144	FF25 7C304000	JMP DWORD PTR DS:[<&gdi32.CreateCompatibleDC>]
0040114A	FF25 80304000	JMP DWORD PTR DS:[<&gdi32.DeleteDC>]
00401150	FF25 84304000	JMP DWORD PTR DS:[<&gdi32.DeleteObject>]
00401156	FF25 88304000	JMP DWORD PTR DS:[<&gdi32.SelectObject>]
0040115C	90	NOP

Now we will save the file to edit by clicking the mouse to select the image in

Address	Disassembly	Comment
0040110B	C2 1000	RETN 10
0040110E	FF25 50304000	JMP DWORD PTR DS:[<&kernel32.ExitProcess>]
00401114	FF25 54304000	JMP DWORD PTR DS:[<&kernel32.GetModuleHandleA>]
0040111A	FF25 5C304000	JMP DWORD PTR DS:[<&user32.BeginPaint>]
00401120	FF25 60304000	JMP DWORD PTR DS:[<&user32.ShowDialogParamA>]
00401126	FF25 64304000	JMP DWORD PTR DS:[<&user32.EndDialog>]
0040112C	FF25 68304000	JMP DWORD PTR DS:[<&user32.EndPaint>]
00401132	FF25 6C304000	JMP DWORD PTR DS:[<&user32.LoadBitmapA>]
00401138	FF25 70304000	JMP DWORD PTR DS:[<&user32.SendMessageA>]
0040113E	FF25 78304000	JMP DWORD PTR DS:[<&gdi32.BitBlt>]
00401144	FF25 7C304000	JMP DWORD PTR DS:[<&gdi32.CreateCompatibleDC>]
0040114A	FF25 80304000	JMP DWORD PTR DS:[<&gdi32.DeleteDC>]
00401150	FF25 84304000	JMP DWORD PTR DS:[<&gdi32.DeleteObject>]
00401156	FF25 88304000	JMP DWORD PTR DS:[<&gdi32.SelectObject>]
0040115C	90	NOP
0040115D	00	DB 00
0040115E	00	DB 00
0040115F	00	DB 00
00401160	00	DB 00
00401161	00	DB 00
00401162	00	DB 00
00401163	00	DB 00
00401164	00	DB 00
00401165	00	DB 00
00401166	00	DB 00
00401167	00	DB 00
00401168	00	DB 00
00401169	00	DB 00
0040116A	00	DB 00
0040116B	00	DB 00
0040116C	00	DB 00
0040116D	00	DB 00
0040116E	00	DB 00

Backup

Copy

Binary

Assemble

Label

Comment

Breakpoint

Hit trace

Run trace

New origin here

Go to

Follow in Dump

Search for

Find references to

View

Copy to executable

Analysis

Space

:

;

Ctrl+Gray *

Selection

All modifications

A dialog box to select the "Copy All". Click your mouse to select "Save File" named unpacked.exe

D File C:\PESpin07MUP\dump_.exe

Address	Disassembly	Comment
0000110E	FF25 50304000	JMP DWORD PTR DS:[403050]
00001114	FF25 54304000	JMP DWORD PTR DS:[403054]
0000111A	FF25 5C304000	JMP DWORD PTR DS:[40305C]
00001120	FF25 60304000	JMP DWORD PTR DS:[403060]
00001126	FF25 64304000	JMP DWORD PTR DS:[403064]
0000112C	FF25 68304000	JMP DWORD PTR DS:[403068]
00001132	FF25 6C304000	JMP DWORD PTR DS:[40306C]
00001138	FF25 70304000	JMP DWORD PTR DS:[403070]
0000113E	FF25 78304000	JMP DWORD PTR DS:[403078]
00001144	FF25 7C304000	JMP DWORD PTR DS:[40307C]
0000114A	FF25 80304000	JMP DWORD PTR DS:[403080]
00001150	FF25 84304000	JMP DWORD PTR DS:[403084]
00001156	FF25 88304000	JMP DWORD PTR DS:[403088]
0000115C	90	NOP

Backup

Copy

Binary

Assemble

Search for

Save file

Test file unpacked.exe. Well. Then. Although quite elaborately but finally we were successful in unpack the J
unpackme this. You can use the PE Tools LordPE or to "rebuild PE will reduce the file size unpacked.exe from 40
to 17 Kb Kb J

Greetz: All members VCT, Crusader, and Ricardo Kagra ...

14-06-2005

tlandn

Manual unpack tElock



Welcome all brothers, this is the first tutorial on 3 of them unpack basis for Newbie. Before writing this tut he thought many tut should write this or not, because the target is 1 soft Vietnamese, but the purpose of them is only a guide for how to unpack it.

First they apologize for the medical Bkis, and 2 are sorry medical NhatPhuongLe (hix, are warning 1 time because I unpack việt soft and that does not contain)!!!!

Target: BKAV2006.exe (included in tut)

Tools: Ollydbg,

ImportREC, PeiD

Protect: tElock 0.90 -> in!

Level: Normal

Scan target with Peid we pack it with **tElock 0.90** -> in!, OEP see any of it:



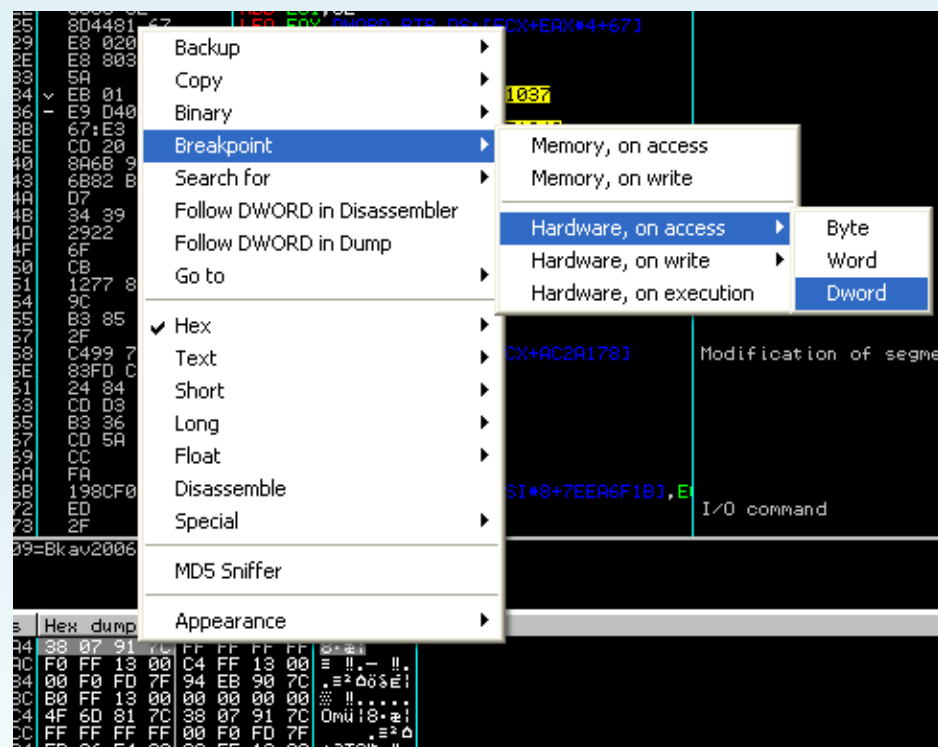
Made, so that we know is 1 some information then. Load Ollydbg to target: the table to inform you select it

```

010F1000 90 NOP
010F1001 60 PUSHAD
010F1002 E8 02000000 CALL Bkav2006.010F1003
010F1007 E8 00E80000 CALL 010FF80C
010F100C 0000 ADD BYTE PTR DS:[EAX],AL
010F100E 5E POP ESI
010F100F 2BC9 SUB ECX,ECX
010F1011 58 POP EAX
010F1012 74 02 JE SHORT Bkav2006.010F1016
010F1014 CD 20 INT 20
010F1016 B9 FF100000 MOV ECX,10FF
010F1018 8BC1 MOV EAX,ECX
010F101D F8 CLC
010F101E 73 02 JNE SHORT Bkav2006.010F1022
010F1020 CD 20 INT 20
010F1023 83C6 32 ADD ESI,32
010F1025 804481 67 LEA EAX,DWORD PTR DS:[ECX+EAX*4+67]
010F1029 E8 02000000 CALL Bkav2006.010F1030
010F102E E8 80300646 CALL 471540B8
010F1033 5A POP EDX
010F1034 EB 01 JMP SHORT Bkav2006.010F1037
010F1036 E9 D409E2EA JMP EBF11A0F
010F103B 67:E3 02 JCXZ SHORT Bkav2006.010F1040
010F103E CD 20 INT 20
010F1040 8A68 92 MOV CH,BYTE PTR DS:[EBX-6E]

```

Press F8 2 times, to look at the screen FPU, **ESP 0013FFA4**, right click -> Follow in dump. Highlight first 4 bytes -> Breakpoint -> Hardware on access -> Dword:



Now press the Shift + F9 8 times (in the child is 8 times), you stand in this place:

CPU - main thread, module Bkav2006

```

010F213A - FF6424 D0 JMP DWORD PTR SS:[ESP-30]
010F213E C8 F97202 ENTER 72F9,2
010F2142 CD 20 INT 20
010F2144 F5 CMC
010F2145 03C7 ADD EAX,EDI
010F2147 E8 41000000 CALL Bkav2006.010F218D
010F214C 5B POP EBX
010F214D E8 00000000 CALL Bkav2006.010F2152
010F2152 0000 ADD BYTE PTR DS:[EAX],AL
010F2154 0000 ADD BYTE PTR DS:[EAX],AL
010F2156 0000 ADD BYTE PTR DS:[EAX],AL
010F2158 0000 ADD BYTE PTR DS:[EAX],AL
010F215A 0000 ADD BYTE PTR DS:[EAX],AL
010F215C 0000 ADD BYTE PTR DS:[EAX],AL
010F215E 0000 ADD BYTE PTR DS:[EAX],AL
010F2160 0000 ADD BYTE PTR DS:[EAX],AL
010F2162 0000 ADD BYTE PTR DS:[EAX],AL
010F2164 0000 ADD BYTE PTR DS:[EAX],AL
010F2166 0000 ADD BYTE PTR DS:[EAX],AL
010F2168 0000 ADD BYTE PTR DS:[EAX],AL
010F216A 0000 ADD BYTE PTR DS:[EAX],AL
010F216C 0000 ADD BYTE PTR DS:[EAX],AL
010F216E 0000 ADD BYTE PTR DS:[EAX],AL
010F2170 0000 ADD BYTE PTR DS:[EAX],AL
010F2172 0000 ADD BYTE PTR DS:[EAX],AL
010F2174 0000 ADD BYTE PTR DS:[EAX],AL
010F2176 0000 ADD BYTE PTR DS:[EAX],AL

```

Just click the star to which the order JMP is OK then! , Which F8 1, Oh ho, we are to OEP and its kakakaka ...

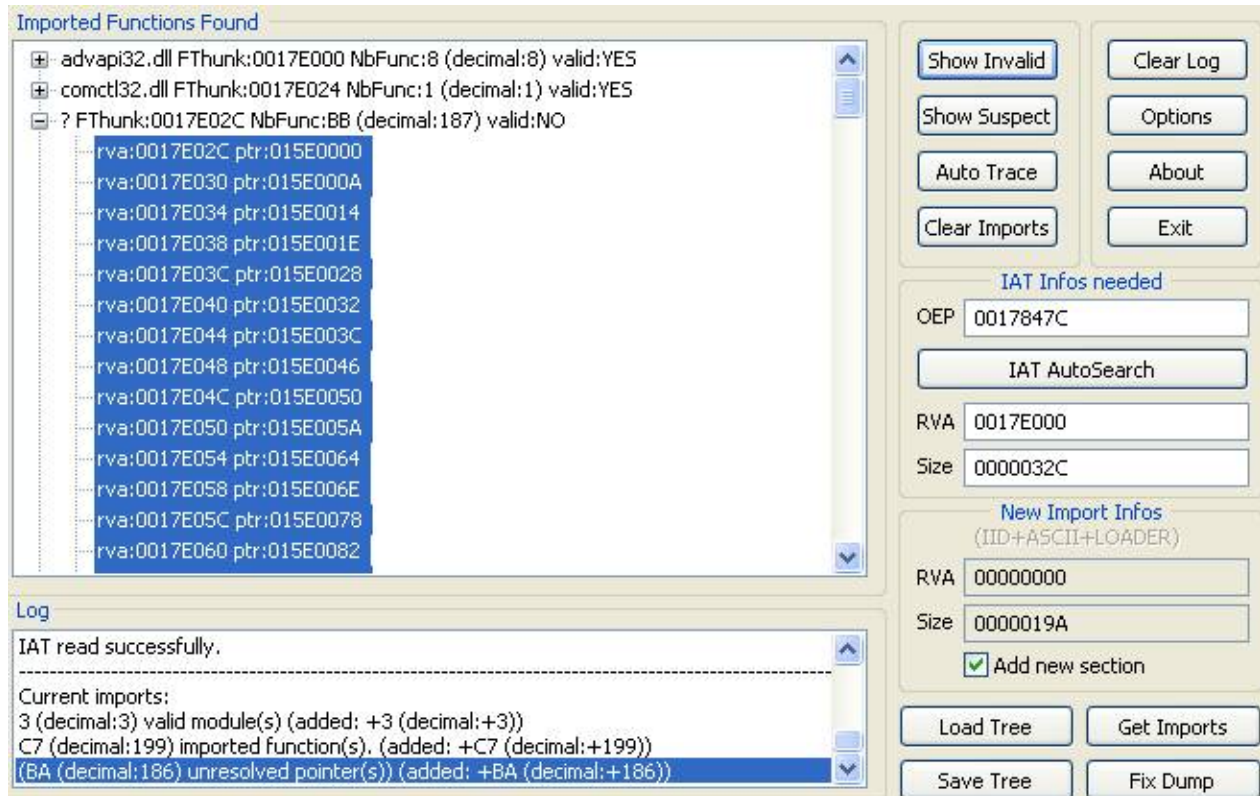
CPU - main thread, module Bkav2006

```

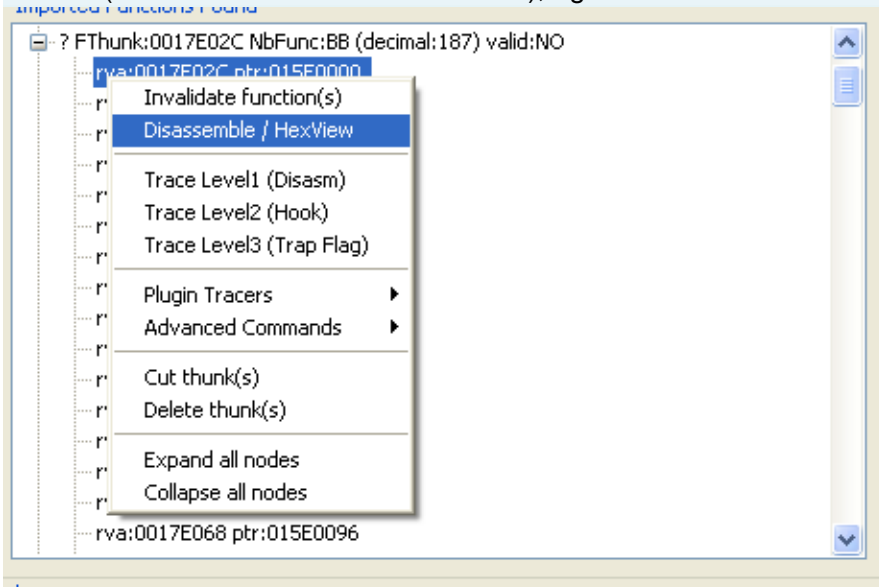
0057847C 55 PUSH EBP
0057847D 8BEC MOV EBP,ESP
0057847F 6A FF PUSH -1
00578481 68 40307D00 PUSH Bkav2006.007D3040
00578486 68 68BA5700 PUSH Bkav2006.0057BA68
0057848B 64:A1 00000000 MOV EAX,DWORD PTR FS:[0]
00578491 50 PUSH EAX
00578492 64:8925 00000000 MOV DWORD PTR FS:[0],ESP
00578499 8BEC 58 SUB ESP,58
0057849C 53 PUSH EBX
0057849D 56 PUSH ESI
0057849E 57 PUSH EDI
0057849F 8965 E8 MOV DWORD PTR SS:[EBP-18],ESP
005784A2 FF15 B0E15700 CALL DWORD PTR DS:[E157B0E1]
005784A8 33D2 XOR EDX,EDX
005784AA 8AD4 MOV DL,AH
005784AC 8915 980A8000 MOV DWORD PTR DS:[800A98],EDX
005784B2 8BC8 MOV ECX,EAX
005784B4 81E1 FF000000 AND ECX,0FF
005784BA 8900 940A8000 MOV DWORD PTR DS:[800A94],ECX
005784C0 C1E1 08 SHL ECX,8
005784C3 03CA ADD ECX,EDX
005784C5 8900 900A8000 MOV DWORD PTR DS:[800A90],ECX
005784CB C1E8 10 SHR EAX,10
005784CE A3 8C0A8000 MOV DWORD PTR DS:[800A8C],EAX
005784D3 33F6 XOR ESI,ESI
005784D5 56 PUSH ESI
005784D6 E8 55010000 CALL Bkav2006.00578630
005784D8 59 POP ECX
005784DC 85C0 TEST EAX,EAX
005784DE 75 08 JNZ SHORT Bkav2006.005784E8
005784E0 6A 1C PUSH 1C
005784E2 E8 B0000000 CALL Bkav2006.00578597
005784E7 59 POP ECX
005784E8 8975 FC MOV DWORD PTR SS:[EBP-4],ESI
005784EB E8 F0220000 CALL Bkav2006.0057A7E0
005784F0 FF15 34E15700 CALL DWORD PTR DS:[E157E134]
005784F6 A3 40208000 MOV DWORD PTR DS:[402080],EAX
005784F8 59 POP ECX

```

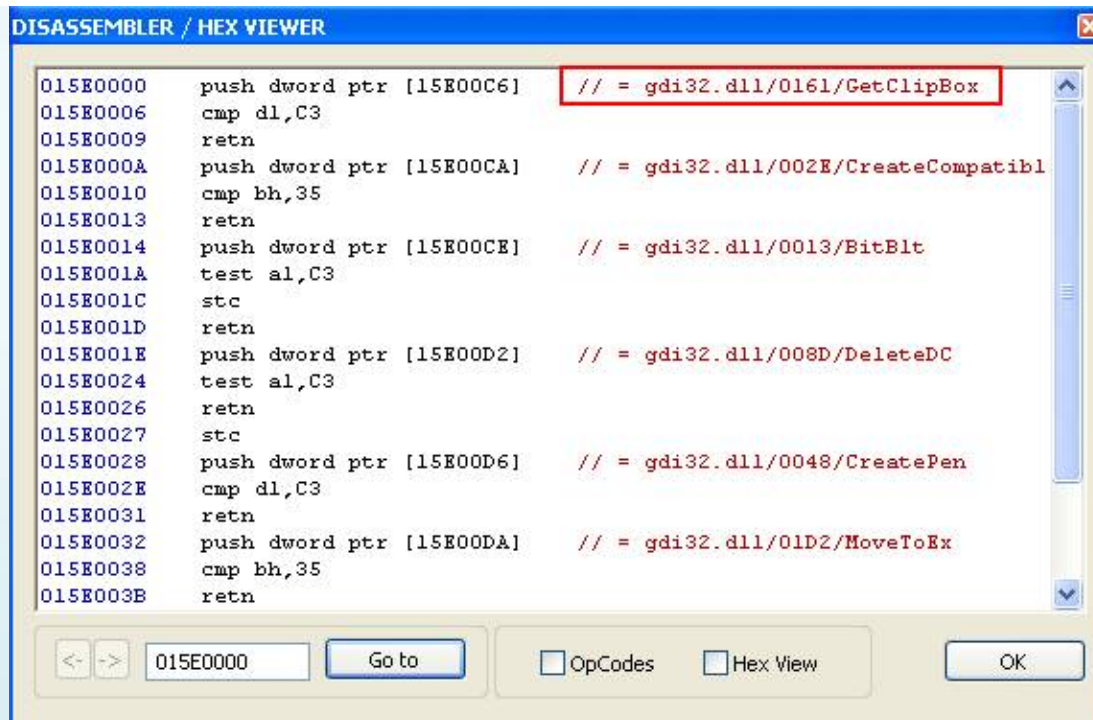
Dump file is, Plugins -> OllyDump -> dump Debugged process, this process they have specific instructions in 1 and tut tut 2 should then they will not do again. After the dump file, load up any ImportREC, then fill OEP IAT AutoSearch, Invalid Show:



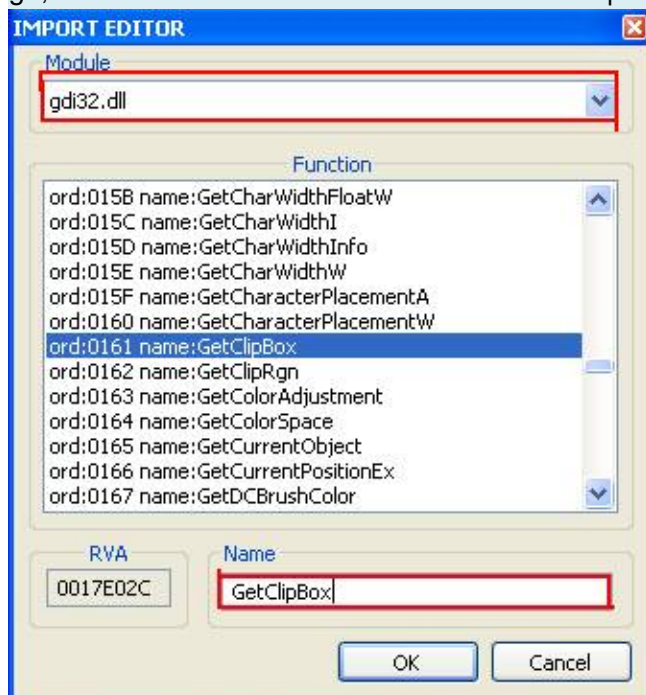
Fix IAT (not that there is not more correct), right-click the invalid first select Disassemble / HexView



Even the first line in the table you'll see: gdi32.dll / 0161 / GetClipBox



Ok remember this from 2 and gdi32.dll GetClipBox, off the table Disassemble / HexView go, double-click the line this rva: 0017E02C ptr: 015E0000 selected as follows:



and
imag
None
Skip
And
> Fix
End



es like the

t Cut thunk (s) -

Manual Unpacking & Cracking

ActiveMark 5. X x



I-- Print d o t r u c t i o n :

ActiveMark 1 Protect is quite famous for the Games feel they own it both difficult and easy to fit. Difficult if we

do not know how to kill and of course extremely easy when we understand the methods of meat do it. The body

then they have done almost the entire game meat is the Protect **ActiveMark** by **bigfishgames** and **Yahoo!** For

Games so they also have little experience in this type of decision and write everything they know about

ActiveMark to share with all brothers ...

II -- Tools T&gerat:

• Tool and Plug in c a n d n g s:

- *O l l i t a l y D B G 1. 1 0*
- *L o r D P E 1 s t 4*
- *I m p o r t h e E C R 1. 6 F*
- *U l t A E d r i t - 3 2*
- *A c t i v e r M a k e r D u m p*
- *A c t i v e r M a k O v e r a y W l i z a r d*
- *A c t i v e r M a k a d e L o r*
- *P r o t i c e t o n i D v 5 t h 1 E*

• One g e r t: *the UK B S u d u o k*

H o m e P a g e: [h_t_t_p://hgwww.bigfi_s_a_m.es.C_o_m/](http://hgwww.bigfi_s_a_m.es.C_o_m/)

II I - U n p a c k i n g

And unpack **ActiveMark** we have 2 ways:

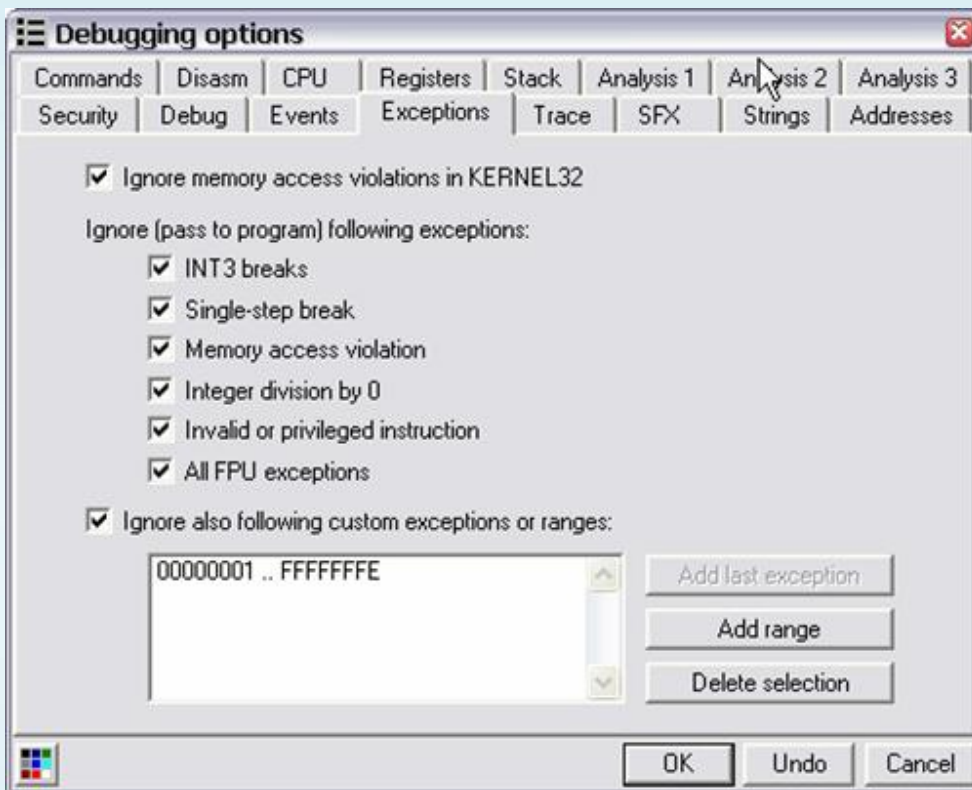
- **Unpacking Tools for support** (this fast and efficient)
- **Manual Unpacking** (this is extremely standard and the professional higher)

1st M a n u a l U n p a c k i n g:

_ Need to **detect** when the **Games** have been Protect Pack or form do they recommend the use Uncle **Protection**
Scan **ID 5.1f** to work because it is quite accurate



_ OK, Open OllyDBG and press **Alt + O** and select the following:



_ Lo ad **in a g e r t** to **OllyDBG** and a stop here:

00748000	8925 04107600	MOV DWORD PTR DS:[761004], ESP	
00748006	68 2B807400	PUSH BUKUSudo.0074802B	
0074800B	EB 03	JMP SHORT BUKUSudo.00748010	
0074800D	EB FB	JMP SHORT BUKUSudo.0074800A	
0074800F	EB EB	JMP SHORT BUKUSudo.00747FFC	
00748011	01EB	ADD EBX, EBP	
00748013	64:FF35 00000000	PUSH DWORD PTR FS:[0]	
0074801A	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
00748021	EB 03	JMP SHORT BUKUSudo.00748026	
00748023	9C	PUSHFD	
00748024	EE	OUT DX, AL	I/O command
00748025	15 E9CD0400	ADC EAX, 4CDE9	

_ Press **Shif t + F9** **G a m e s u n R'll**



_ Now we need to determine the Section contains the **OEP**. In **OllyDBG** window press **Alt + M** and drag down and see

003E0000	00003000			Map	R	R
003F0000	00004000			Priv	RW	RW
00400000	00001000	BUKUSudo	PE header	Imag	RW	Copi RWE
00401000	00008000	BUKUSudo	code	Imag	RW	Copi RWE
00409000	00014000	BUKUSudo	.bss	Imag	RW	Copi RWE
0040D000	00013000	BUKUSudo	.text	Imag	RW	Copi RWE
0040E000	00013000	BUKUSudo	.rdata	Imag	RW	Copi RWE
0040F000	00013000	BUKUSudo	.rsrc	Imag	RW	Copi RWE
00410000	00016E000	BUKUSudo	.data	Imag	RW	Copi RWE
00620000	00003000	BUKUSudo	.idata	Imag	RW	Copi RWE
00623000	00003000	BUKUSudo	.rsrc	Imag	RW	Copi RWE
00626000	0011F000	BUKUSudo	.bss	Imag	RW	Copi RWE
00745000	00001000	BUKUSudo	.idata	Imag	RW	Copi RWE
00746000	00002000	BUKUSudo	.rsrc	Imag	RW	Copi RWE
00748000	00018000	BUKUSudo	.text	Imag	RW	Copi RWE
00749000	00001000	BUKUSudo	.rdata	Imag	RW	Copi RWE
0074A000	0000A000	BUKUSudo	.data	Imag	RW	Copi RWE
0074B000	00001000	BUKUSudo	.idata	Imag	RW	Copi RWE

file:///C:/RCE%20Unpacking%20eBook%20[Tra...ng%20&%20Cracking%20ActiveMark%205.xx.htm (5 of 22) [1/9/2009 9:44:58 LithiumLi]

_ G G o t e d C o m m a n i n E A L

005FC0CD	CALL NEAR DWORD PTR DS:[620288]	kernel32.InterlockedIncrement
005FC1A4	CALL NEAR DWORD PTR DS:[6202D0]	kernel32.MultiByteToWideChar
005FC200	CALL NEAR DWORD PTR DS:[6202D0]	kernel32.MultiByteToWideChar
005FC227	CALL NEAR DWORD PTR DS:[6202D0]	kernel32.MultiByteToWideChar
005FC423	CALL BUKUSudo.0060DF26	ntdll.RtlUnwind
005FC6B7	CALL BUKUSudo.0060DF26	ntdll.RtlUnwind
005FC7D9	CALL NEAR DWORD PTR DS:[620240]	kernel32.GetVersion
005FC839	CALL NEAR DWORD PTR DS:[620180]	kernel32.GetCommandLineA
005FC864	CALL NEAR DWORD PTR DS:[620210]	kernel32.GetStartupInfoA
005FC887	CALL NEAR DWORD PTR DS:[6201E8]	kernel32.GetModuleHandleA
005FC8FD	CALL NEAR DWORD PTR DS:[620144]	kernel32.ExitProcess
005FCCD3	CALL NEAR DWORD PTR DS:[6202F0]	kernel32.RaiseException
005FDF17	CALL NEAR DWORD PTR DS:[620264]	ntdll.RtlAllocateHeap
005FDF6E	CALL NEAR DWORD PTR DS:[620274]	ntdll.RtlReAllocateHeap
005FE07F	CALL NEAR DWORD PTR DS:[620264]	ntdll.RtlAllocateHeap
005FE0C0	CALL NEAR DWORD PTR DS:[620274]	ntdll.RtlReAllocateHeap

N _ steamed double training and am G e h t o C i n m m a n d L E A

005FC825	75 08	JNZ SHORT BUKUSudo.005FC82F	
005FC827	6A 10	PUSH 10	
005FC829	E8 B2000000	CALL BUKUSudo.005FC8E0	
005FC82C	59	POP ECX	
005FC82F	33F6	XOR ESI, ESI	
005FC831	8975 FC	MOV DWORD PTR SS:[EBP-4], ESI	
005FC834	E8 C05F0000	CALL BUKUSudo.006027F9	
005FC839	FF15 80016200	CALL NEAR DWORD PTR DS:[620180]	kernel32.GetCommandLineA
005FC83F	A3 84CB6100	MOV DWORD PTR DS:[61CB84], EAX	
005FC844	E8 7E5E0000	CALL BUKUSudo.006026C7	
005FC849	A3 E0B36100	MOV DWORD PTR DS:[61B3E0], EAX	
005FC84E	E8 275C0000	CALL BUKUSudo.0060247A	
005FC853	E8 695B0000	CALL BUKUSudo.006023C1	

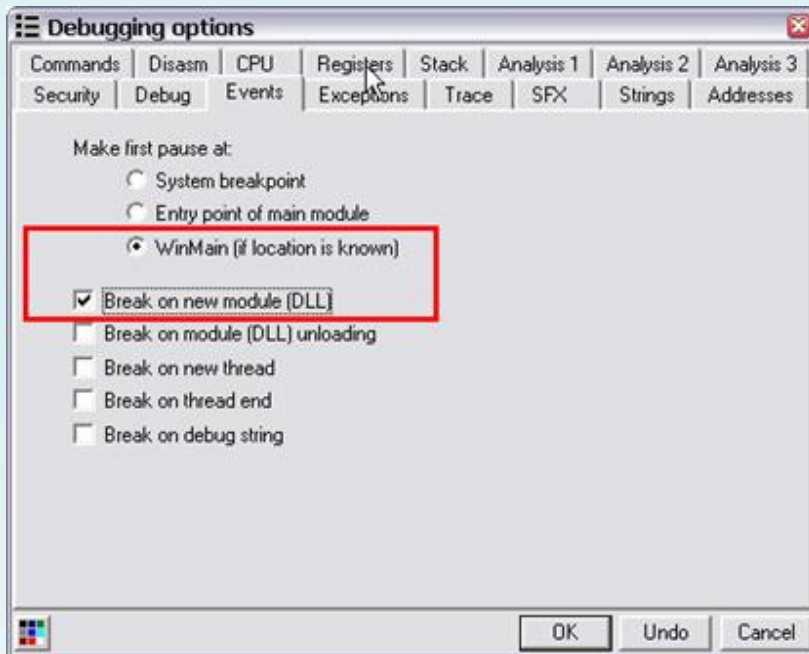
_ Scroll back to the top of this code and is OEP

201607AD	8D6C24 0C	LEA EBP, DWORD PTR SS:[ESP+C]	
201607B1	50	PUSH EAX	
201607B2	C3	RETN	
201607B3	55	PUSH EBP	<=== OEP
201607B4	8BEC	MOV EBP, ESP	
201607B6	6A FF	PUSH -1	
201607B8	68 08170820	PUSH PrimeSus.20081708	
201607BD	68 906B1620	PUSH PrimeSus.20166B90	
201607C2	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
201607C8	50	PUSH EAX	
201607C9	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
201607D0	83EC 58	SUB ESP, 58	
201607D3	53	PUSH EBX	
201607D4	56	PUSH ESI	
201607D5	57	PUSH EDI	
201607D6	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
201607D9	FF15 28421820	CALL NEAR DWORD PTR DS:[20184228]	kernel32.GetVersion
201607DF	33D2	XOR EDX, EDX	
201607E1	8AD4	MOV DL, AH	

_ B at below g a t S e t 1 WBP in H O P E

005FC7A0	8D6C24 0C	LEA EBP, DWORD PTR SS:[ESP+C]	
005FC7B1	50	PUSH EAX	
005FC7B2	C3	RETN	
005FC7B3	55	PUSH EBP	<=== OEP
005FC7B4	8BEC	MOV EBP, ESP	
005FC7B6	6A FF	PUSH -1	
005FC7B8	68 08D75100	PUSH BUKUSudo.0051D708	
005FC7BD	68 902B6000	PUSH BUKUSudo.00602B90	
005FC7C2	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
005FC7C8	50	PUSH EAX	
005FC7C9	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
005FC7D0	83EC 58	SUB ESP, 58	
005FC7D3	53	PUSH EBX	
005FC7D4	56	PUSH ESI	
005FC7D5	57	PUSH EDI	
005FC7D6	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
005FC7D9	FF15 40026200	CALL NEAR DWORD PTR DS:[620240]	kernel32.GetVersion
005FC7DF	33D2	XOR EDX, EDX	
005FC7E1	8AD4	MOV DL, AH	

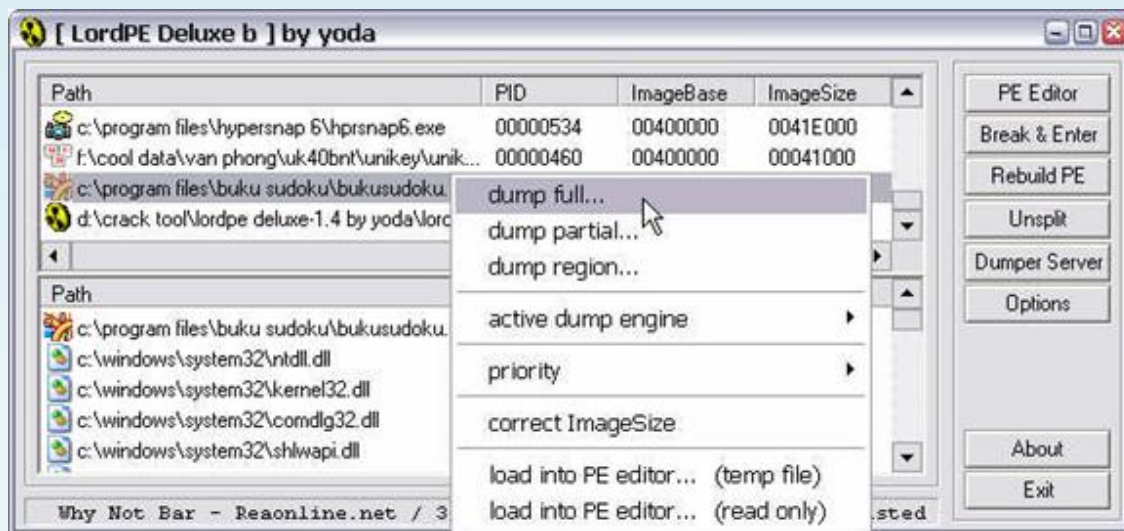
_Ok, n a l t han + O and the more the usau h



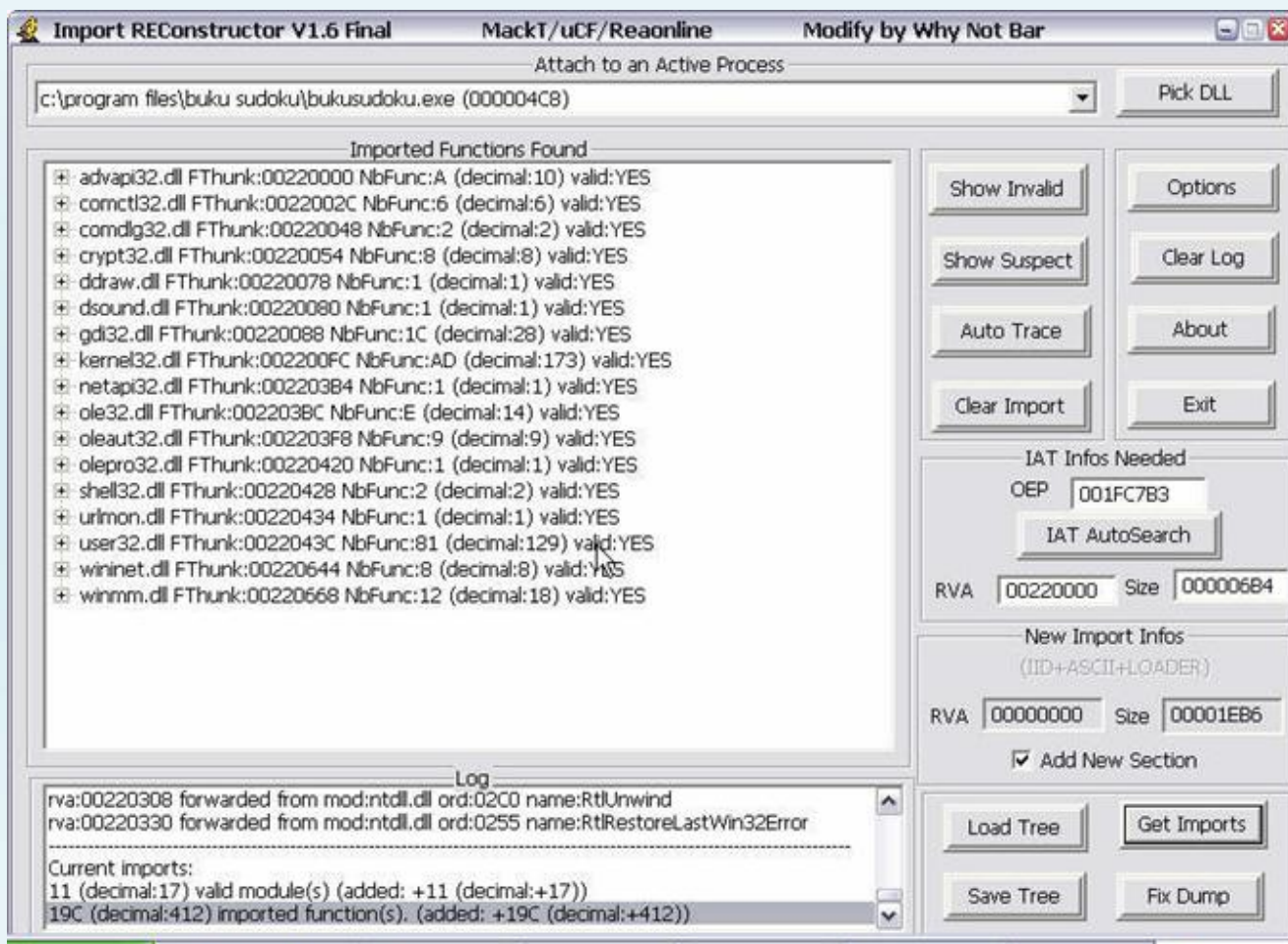
_ Press **Ctrl + F2**, press **F9** to stop the **OEP** (machine they hit 17 times) that we've set **HWBP**

005FC7B3	55	PUSH EBP	<=== OEP
005FC7B4	8BEC	MOV EBP, ESP	
005FC7B6	6A FF	PUSH -1	
005FC7B8	68 08D75100	PUSH BUKUSudo.0051D708	
005FC7BD	68 902B6000	PUSH BUKUSudo.00602B90	
005FC7C2	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
005FC7C8	50	PUSH EAX	
005FC7C9	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
005FC7D0	83EC 58	SUB ESP, 58	
005FC7D3	53	PUSH EBX	
005FC7D4	56	PUSH ESI	
005FC7D5	57	PUSH EDI	
005FC7D6	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
005FC7D9	FF15 40026200	CALL NEAR DWORD PTR DS:[620240]	kernel32.GetVersion
005FC7DF	33D2	XOR EDX, EDX	

No _A K h à, Enough Lor n g p um dPE D F u l l

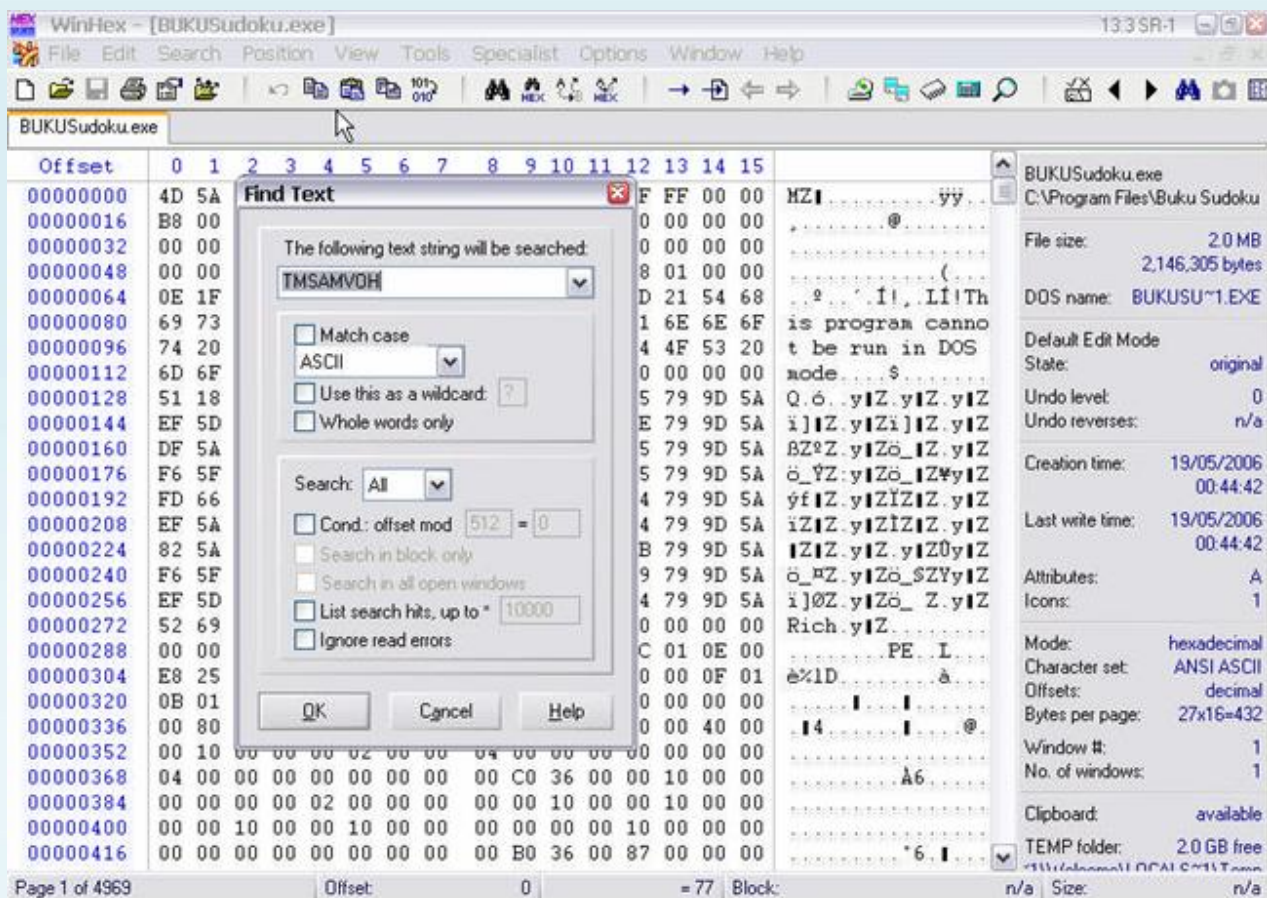


_ Open **ImportREC** up. Select the list **BUKUSudoku.exe** process. **OEP** = Enter **005FC7B3** - **0,040,000** (**Imagebase**) = **001FC7B3**, **IAT AutoSearch** Click -> **Get Imports** -> **Show Invalid**



_ Hu hon, I do not have any of Invalid, click **Fix dump** select File "**Dumped.exe**" Run and try File "**Dumped_.exe**" Run chit they are in line ... The reasons are due **ActiveMark 5.xx** have to add 1 or function that is quite **Overlay Data**, this will encrypt the Data 1 and will be the resolution ActiveMark code for game use. But **ActiveMark** we have **Kill Encrypt Data** and the authors he is writing **ActiveMark** ong star lem or even dump Full card of the prices are still more loss of this data. In the original file still contains the **Encrypt Data** and tasks we need to identify and copy to be in the **Encrypt Data** from File "**BUKUSudoku.exe**" and Paste to File "**Dumped_.exe**" Signs identified as the **Encrypt Data** always start with. **TMSAMVOH** and last address to the final. Using a tool Hex Edit any to us, and the Copy **Data Encrypt** here they use **WinHex** (If using **Hex Workshop**

4.2 with game with space when copying large files sebi Crash best use **WinHex**). Load File "**BUKUSudoku.exe**" to **WinHex**, press **Ctrl + F** and type in **TMSAMVOH**

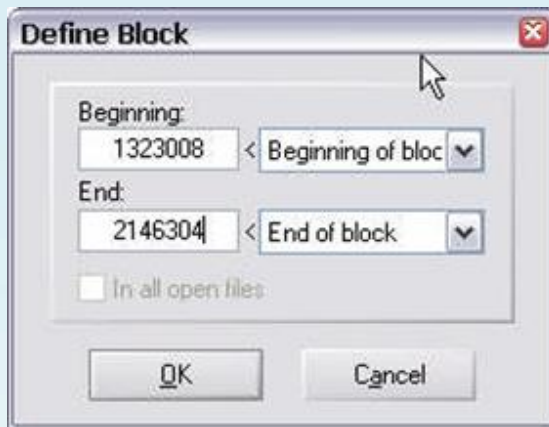


_ Select **01323008** to address final

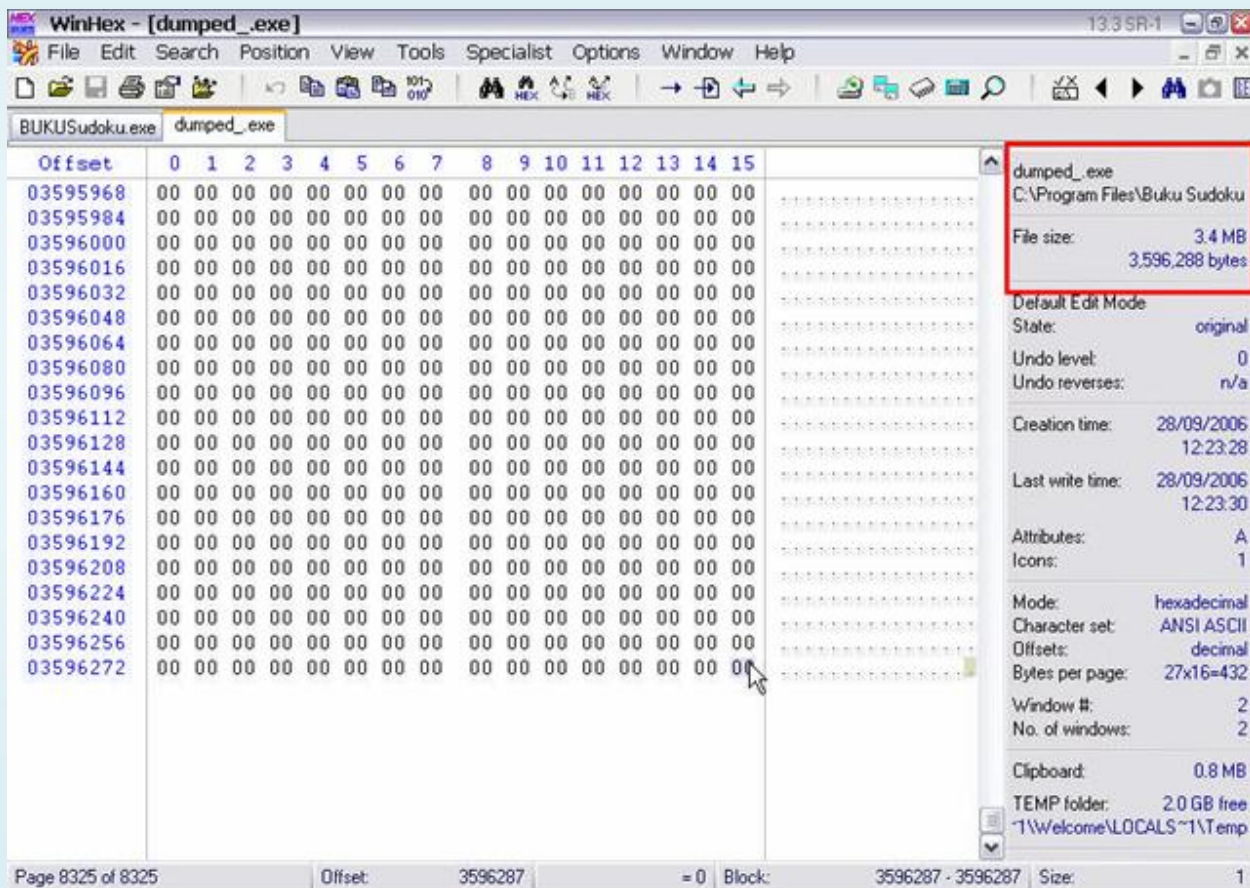
01322960	27 00 00 00 00 00 00 00	6A 01 00 00 6D 00 00 00j...m...
Start 76	01 00 00 00 03 00 4E 20	2E 74 65 78 81 02 00 00N...tex ...
01323008	B0 BF 14 08 48 13 13 42	00 00 00 00 00 00 00 00	*c...H...B.....
01323024	00 54 4D 53 41 4D 56 4F	48 A4 9B FD FF 26 24 E9	.TMSAMVOH* yy&\$e
01323040	D7 F1 D6 F0 D6 AE BE FC	D6 DF B5 C1 D0 1F 07 CE	xR0S00%u0BpAD...I
01323056	EF EE DD DE 4F F1 D1 AE	BE A4 9B FD FF 26 21 EC	iiYp0Rn0% yy&li
01323072	CE F1 D6 F0 D6 AE BE 01	00 14 00 6F 48 9C B3 37	IR0S00%...oH ?7
01323088	8B C7 4E 61 43 33 41 B1	F2 0F B9 2C 89 F1 CD 01	cNaC3Ato... Kf
01323104	00 00 00 01 E4 01 00 00	00 00 01 00 21 04 01 00	...a... ...
01323120	7F 74 DA EE 7C 51 DB C8	EF B2 16 B5 48 32 5E 8A	tUi Q0Ei...pH2^
01323136	99 84 5E 1C 85 AB B1 FF	61 9D 09 18 32 F1 A7 2C	^... <tya ...2ES
01323152	1E 72 CF BA BB 40 51 EE	39 88 E9 44 AB 9C 6E 03	.rI?>@Q19 eD<< n
01323168	2D E6 5E 31 A1 F3 86 76	71 9A 8B C3 8B B7 BD DB	-e^1 o vq IA ...%0
01323184	ED F1 4B E6 D9 1E 58 51	2D 61 78 D7 FB 0F 73 A0	inKa0.XQ-axxu.s
01323200	7A 8D BA 35 22 CE BF EF	03 B0 F4 18 D2 50 A2 00	ya#^2' .0 EJm
			z ?5"i0i...'o.OPe@

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
02146080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
02146096	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
02146112	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
02146128	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
02146144	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
02146160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
02146176	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
02146192	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
02146208	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
02146224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
02146240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
02146256	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
02146272	00	00	00	00	00	00	00	54	4D	53	41	4D	56	4F	46	01	TMSAMVOF
02146288	00	00	00	00	00	E6	6F	0C	00	30	10	0C	00	84	82	18	00 00 00 00
02146304	00																

_ You should use the **Block Define WinHex** to block the fast



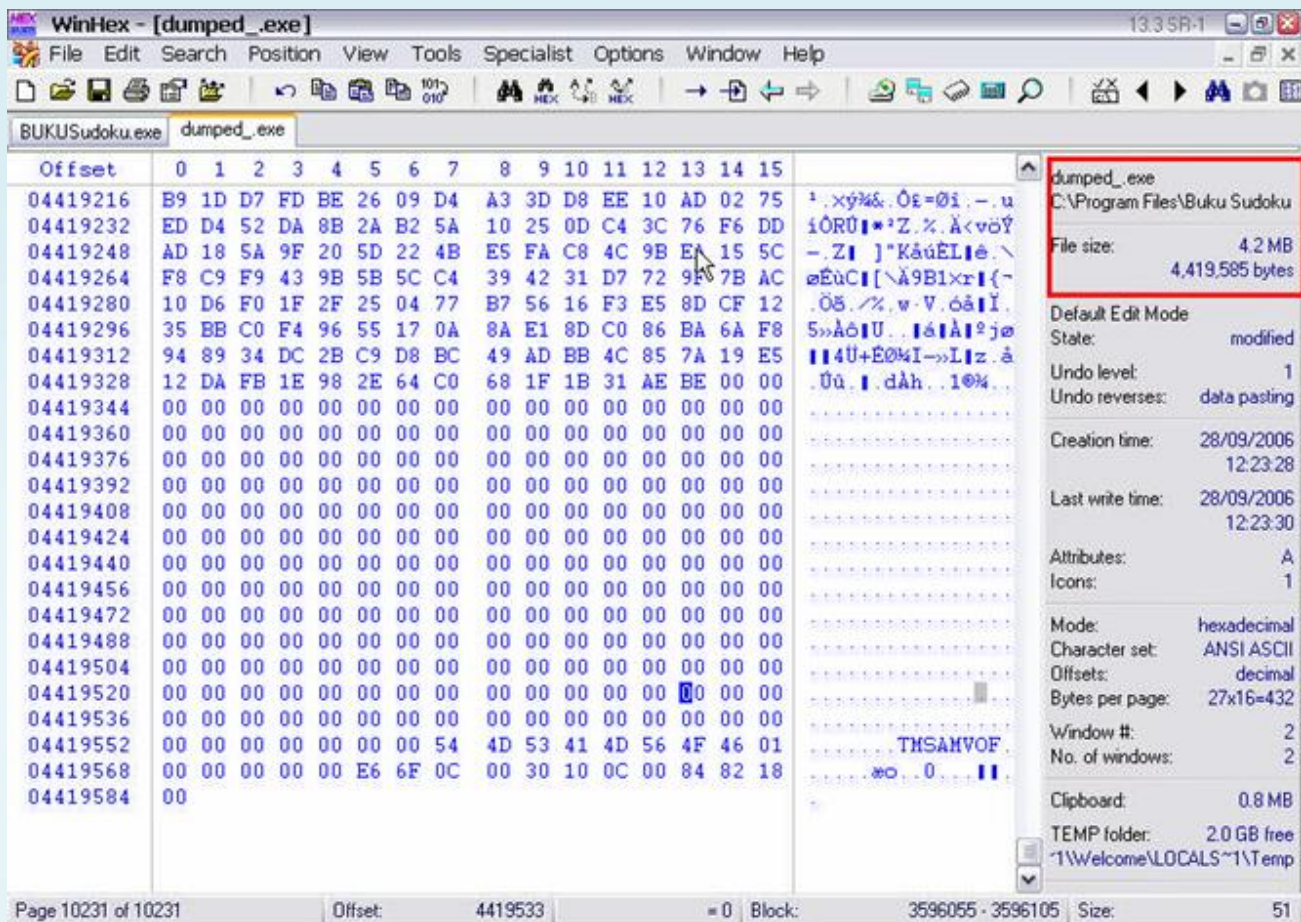
_ Press **Ctrl + C** to copy the **Data Encrypt** this, press **Ctrl + O** select File "**dumped_.exe**" and scroll down to the last address



_ Press **Ctrl + V** to paste and the **Encrypt Data** Copy both to address the end of File "**Dumped_exe**"



_ Click **Yes** we are



_ Passable ... **Save** the name of what is, they placed "**Unpacked.exe**" easy to remember ... **Run** ... khua try khua both numbness run lickerish unpack done!

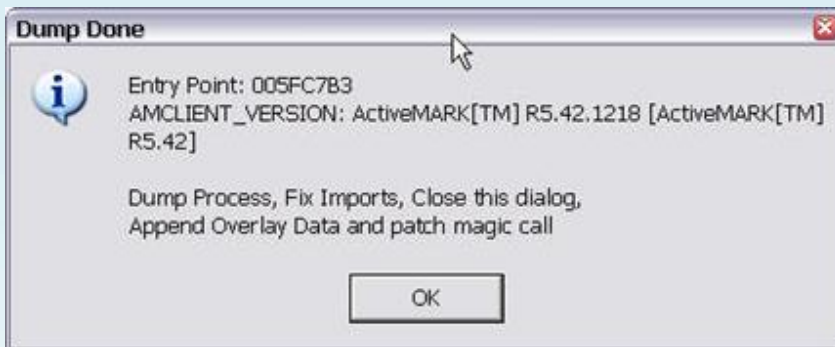
2nd U n pa ck with T oo l s

_ MUP the long line nhuvay tools are used but less than 2 minutes. Open **Amdumper** This tool helps you find

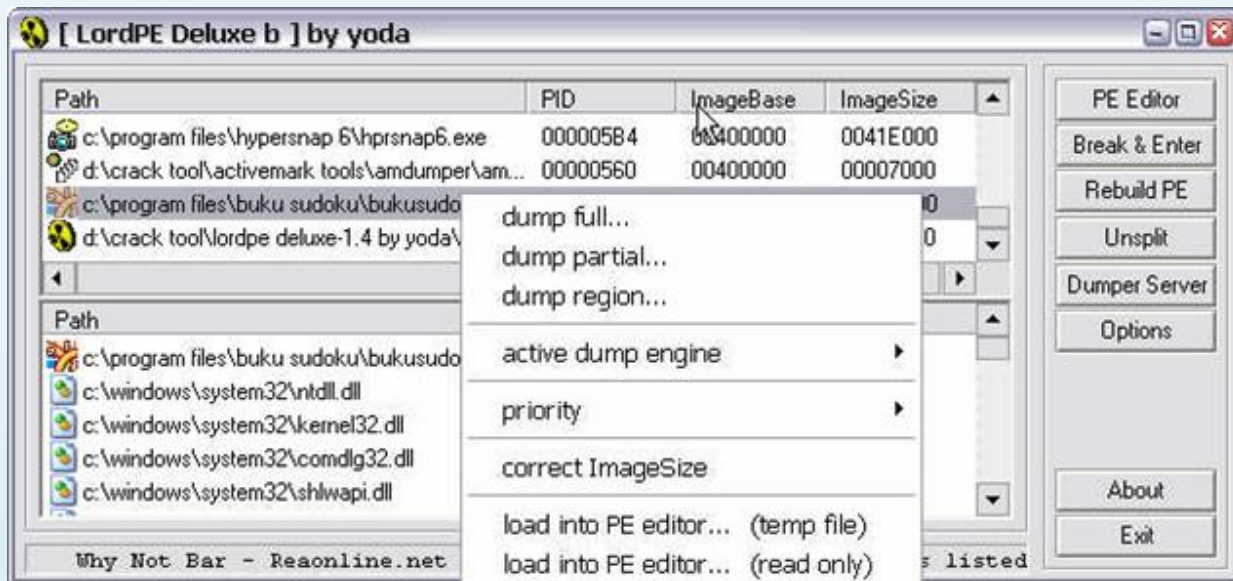
the OEP and you **dump Full Import** and **rebuild** with **ImportREC**



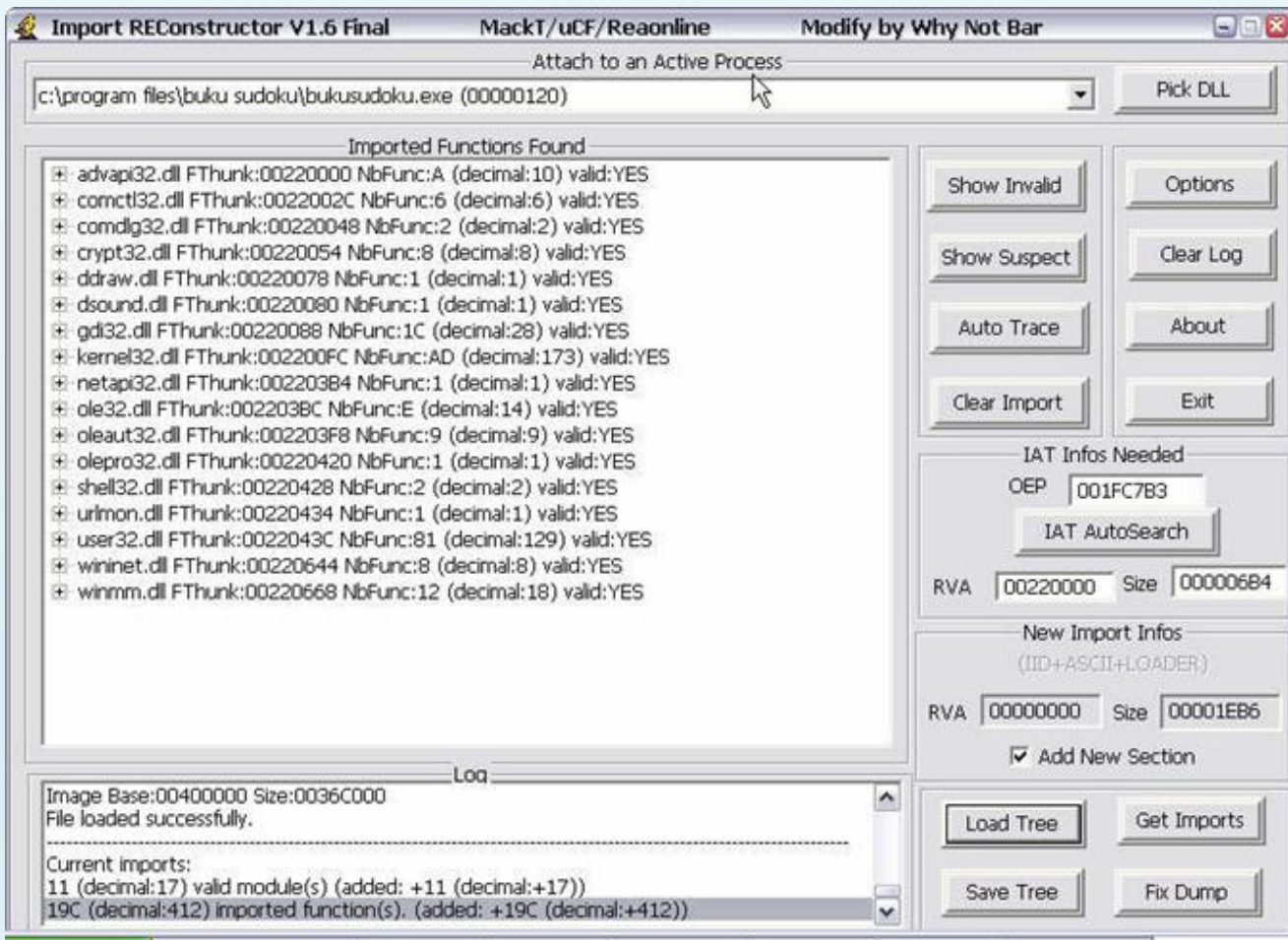
_Nhan "Open ActiveMARK protected target" choose file "BUKUSudoku.exe"



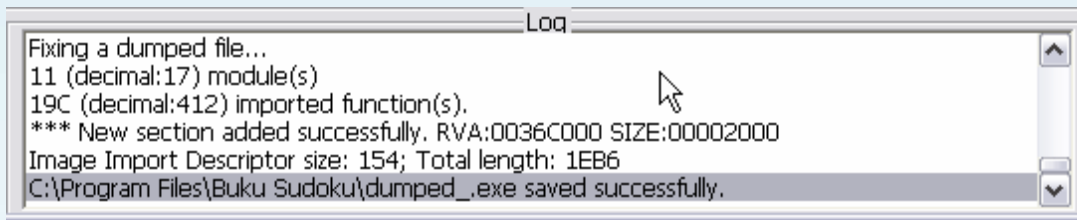
_ Open LordPE choose the correct PID and dump Full



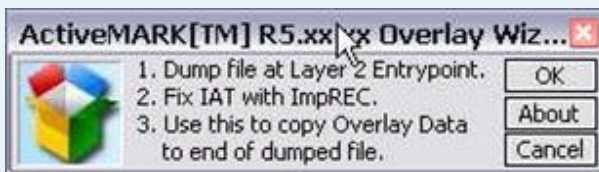
_ Open ImportREC up. Select the list process "BUKUSudoku.exe." Fill 005FC7B3 OEP = (Entry Point)
 - 0,040,000 (Imagebase) = 001FC7B3, IAT AutoSearch Click -> Get Imports -> Show Invalid



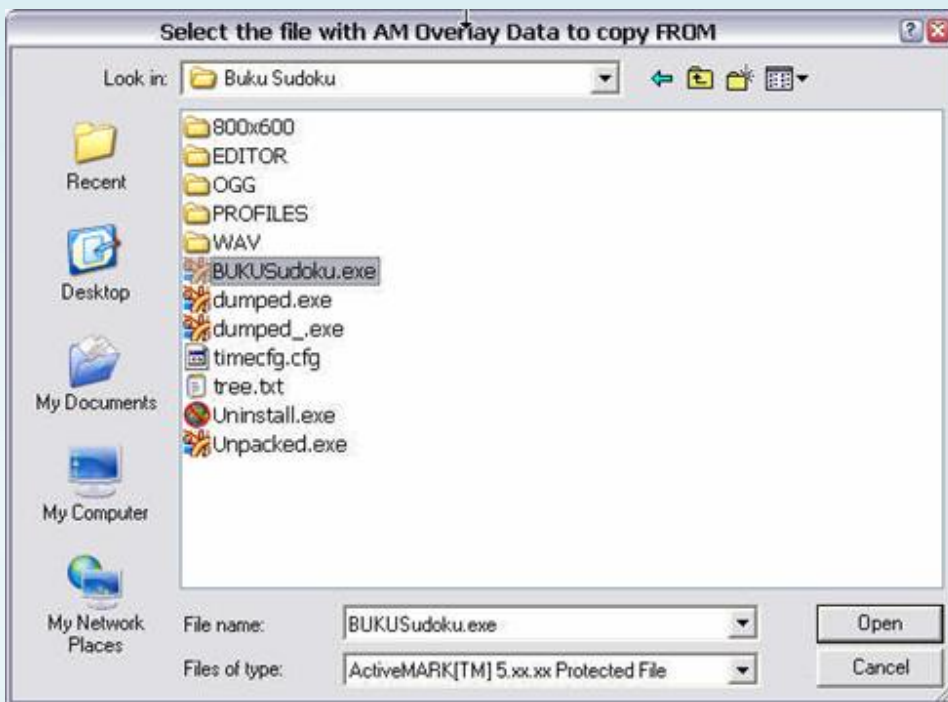
_ Click **Fix** selected **dump** file "**Dumped.exe**"



_ **Dark Overlay Wizard** tool will help you automatically **Encrypt Data** Copy files from the original file to "**dumped_.exe**" quickly but do need to use **WinHex**



_ Click **OK** to select the file "**BUKUSudoku.exe**"



_ Select "**dumped_.exe**." Test Run ... khua khua also run lickerish

I V - C r a c k i n g

_ C h u n g a t n e e d i t a l y B P s c a s A s i a i n a G t o c o n s i d e r i n t h e



_Lo Ad file Unpack de a car on Oll yD B G

005FC7B3	55	PUSH EBP	
005FC7B4	8BEC	MOV EBP, ESP	
005FC7B6	6A FF	PUSH -1	
005FC7B8	68 00D75100	PUSH Unpacked.0051D708	
005FC7BD	68 902B6000	PUSH Unpacked.00602B90	
005FC7C2	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
005FC7C8	50	PUSH EAX	
005FC7C9	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
005FC7D0	83EC 58	SUB ESP, 58	
005FC7D3	53	PUSH EBX	
005FC7D4	56	PUSH ESI	
005FC7D5	57	PUSH EDI	
005FC7D6	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
005FC7D9	FF15 40026200	CALL NEAR DWORD PTR DS:[<&kernel32.GetVersion	kernel32.GetVersion

_D ùn g ug Pl i n "Ul of a S tr ing R e f e r e n c e"

_L E N Scroll first that the ì n h, n han C t r l + F and G o and o "was br o r e, h a n n h p í MN four times

005206A7	MOV EDI, Unpacked.004B2AA0	hooklib_delayed
00520730	MOV EDI, Unpacked.004B2AA0	ignore_key
00520752	MOV EDI, Unpacked.004B2A98	alt_tab
0052080A	MOV EDI, Unpacked.004B2A90	alt_f4
0052083E	PUSH Unpacked.004B2D1C	,\n\r
005208BF	MOV ESI, Unpacked.004B2A88	browser
00520995	MOV DWORD PTR SS:[EBP+C], Unpacked.004B2A88	unknown exception code
00520A18	MOV DWORD PTR DS:[ESI], Unpacked.00515B20	m\nr
00520A35	MOV DWORD PTR DS:[ESI], Unpacked.00515B20	m\nr
00520A81	MOV DWORD PTR DS:[ESI], Unpacked.00515B20	m\nr

_In the other game of the crack and you do not need to hit **the** Arab world campus **4** times that you find how the double click on the "**browser**" will jump to address code distant nhuben the same as

00520897	C2 1000	RETN 10	
0052089A	B8 18F66000	MOV EAX, Unpacked.0060F618	
0052089F	E8 F0BE0D00	CALL Unpacked.005FC794	
005208A4	83EC 5C	SUB ESP, 5C	
005208A7	8A45 0B	MOV AL, BYTE PTR SS:[EBP+B]	
005208AA	53	PUSH EBX	
005208AB	56	PUSH ESI	
005208AC	57	PUSH EDI	
005208AD	8BF9	MOV EDI, ECX	
005208AF	6A 00	PUSH 0	
005208B1	8D4D DC	LEA ECX, DWORD PTR SS:[EBP-24]	
005208B4	897D EC	MOV DWORD PTR SS:[EBP-14], EDI	
005208B7	8B45 DC	MOV BYTE PTR SS:[EBP-24], AL	
005208BA	E8 94260000	CALL Unpacked.00522F53	
005208BF	BE 882A4B00	MOV ESI, Unpacked.004B2A88	browser
005208C4	56	PUSH ESI	
005208C5	E8 46C00D00	CALL Unpacked.005FC910	
005208CA	59	POP ECX	

_ Scroll to the top of **the** code and **Patch**

_ Save and Run Run ... try considered straight game to do is remind NAG



_ To ensure that the food you find this more 4 String "dialog", "timeout", "timer", "execute" and Patch it

_ Ser of a "dial o g"

00520B86	C9	LEAVE	
00520B87	C2 0400	RETN 4	
00520B8A	B8 88F66000	MOV EAX, Unpacked.0060F688	
00520B8F	E8 00BC0D00	CALL Unpacked.005FC794	
00520B94	83EC 5C	SUB ESP, 5C	
00520B97	8A45 13	MOV AL, BYTE PTR SS:[EBP+13]	
00520B9A	53	PUSH EBX	
00520B9B	56	PUSH ESI	
00520B9C	57	PUSH EDI	
00520B9D	8BF9	MOV EDI, ECX	
00520B9F	6A 00	PUSH 0	
00520BA1	8D4D DC	LEA ECX, DWORD PTR SS:[EBP-24]	
00520BA4	897D EC	MOV DWORD PTR SS:[EBP-14], EDI	
00520BA7	8845 DC	MOV BYTE PTR SS:[EBP-24], AL	
00520BAA	E8 A4230000	CALL Unpacked.00522F53	
00520BAF	BE 802A4B00	MOV ESI, Unpacked.004B2A80	dialog
00520BB4	56	PUSH ESI	
00520BB5	E8 56BD0D00	CALL Unpacked.005FC910	

Patch to:

00520B86	C9	LEAVE	
00520B87	C2 0400	RETN 4	
00520B8A	C3	RETN	
00520B8B	90	NOP	
00520B8C	90	NOP	
00520B8D	90	NOP	
00520B8E	90	NOP	
00520B8F	E8 00BC0D00	CALL Unpacked.005FC794	
00520B94	83EC 5C	SUB ESP, 5C	
00520B97	8A45 13	MOV AL, BYTE PTR SS:[EBP+13]	
00520B9A	53	PUSH EBX	
00520B9B	56	PUSH ESI	

00520B9A	53	PUSH EBX	
00520B9B	56	PUSH ESI	
00520B9C	57	PUSH EDI	
00520B9D	8BF9	MOV EDI, ECX	
00520B9F	6A 00	PUSH 0	
00520BA1	8D4D DC	LEA ECX, DWORD PTR SS:[EBP-24]	
00520BA4	897D EC	MOV DWORD PTR SS:[EBP-14], EDI	
00520BA7	8845 DC	MOV BYTE PTR SS:[EBP-24], AL	
00520BAA	E8 A4230000	CALL Unpacked.00522F53	
00520BAF	BE 802A4B00	MOV ESI, Unpacked.004B2A80	dialog
00520BB4	56	PUSH ESI	
00520BB5	E8 56BD0D00	CALL Unpacked.005FC910	
00520BBA	59	POP ECX	

_Ser of a "t i m e t o u"

00520C43	5B	POP EBX	
00520C44	C9	LEAVE	
00520C45	C2 0C00	RETN 0C	
00520C48	B8 9CF66000	MOV EAX, Unpacked.0060F69C	
00520C4D	E8 42B80D00	CALL Unpacked.005FC794	
00520C52	83EC 5C	SUB ESP, 5C	
00520C55	8A45 0B	MOV AL, BYTE PTR SS:[EBP+B]	
00520C58	53	PUSH EBX	
00520C59	56	PUSH ESI	
00520C5A	57	PUSH EDI	
00520C5B	8BF9	MOV EDI, ECX	
00520C5D	6A 00	PUSH 0	
00520C5F	8D4D DC	LEA ECX, DWORD PTR SS:[EBP-24]	
00520C62	897D EC	MOV DWORD PTR SS:[EBP-14], EDI	
00520C65	8845 DC	MOV BYTE PTR SS:[EBP-24], AL	
00520C68	E8 E6220000	CALL Unpacked.00522F53	
00520C6D	BE 702A4B00	MOV ESI, Unpacked.004B2A70	timeout
00520C72	56	PUSH ESI	
00520C73	E8 98BC0D00	CALL Unpacked.005FC910	
00520C78	59	POP ECX	

P a t a n of the h

00520C43	5B	POP EBX	
00520C44	C9	LEAVE	
00520C45	C2 0C00	RETN 0C	
00520C48	C3	RETN	
00520C49	90	NOP	
00520C4A	90	NOP	
00520C4B	90	NOP	
00520C4C	90	NOP	
00520C4D	E8 42B80D00	CALL Unpacked.005FC794	
00520C52	83EC 5C	SUB ESP, 5C	
00520C55	8A45 0B	MOV AL, BYTE PTR SS:[EBP+B]	
00520C58	53	PUSH EBX	
00520C59	56	PUSH ESI	
00520C5A	57	PUSH EDI	
00520C5B	8BF9	MOV EDI, ECX	
00520C5D	6A 00	PUSH 0	
00520C5F	8D4D DC	LEA ECX, DWORD PTR SS:[EBP-24]	

00520C5B	8BF9	MOV EDI, ECX	
00520C5D	6A 00	PUSH 0	
00520C5F	8D4D DC	LEA ECX, DWORD PTR SS:[EBP-24]	
00520C62	897D EC	MOV DWORD PTR SS:[EBP-14], EDI	
00520C65	8845 DC	MOV BYTE PTR SS:[EBP-24], AL	
00520C68	E8 E6220000	CALL Unpacked.00522F53	
00520C6D	BE 702A4B00	MOV ESI, Unpacked.004B2A70	timeout
00520C72	56	PUSH ESI	

_Ser of a "T im e r"

00520D2C	C9	LEAVE	
00520D2D	C2 0400	RETN 4	
00520D30	B8 00F66000	MOV EAX, Unpacked.0060F6B0	,È\ra
00520D35	E8 5ABA0D00	CALL Unpacked.005FC794	
00520D3A	83EC 5C	SUB ESP, 5C	
00520D3D	8A45 0B	MOV AL, BYTE PTR SS:[EBP+8]	
00520D40	53	PUSH EBX	
00520D41	56	PUSH ESI	
00520D42	57	PUSH EDI	
00520D43	8BF9	MOV EDI, ECX	
00520D45	6A 00	PUSH 0	
00520D47	8D4D DC	LEA ECX, DWORD PTR SS:[EBP-24]	
00520D4A	897D EC	MOV DWORD PTR SS:[EBP-14], EDI	
00520D4D	8845 DC	MOV BYTE PTR SS:[EBP-24], AL	
00520D50	E8 FE210000	CALL Unpacked.00522F53	
00520D55	BE 782A4B00	MOV ESI, Unpacked.004B2A78	timer
00520D5A	56	PUSH ESI	

P a t a n of the h

00520D2B	5E	POP ESI	
00520D2C	C9	LEAVE	
00520D2D	C2 0400	RETN 4	
00520D30	C3	RETN	,È\ra
00520D31	90	NOP	
00520D32	90	NOP	
00520D33	90	NOP	
00520D34	90	NOP	
00520D35	E8 5ABA0D00	CALL Unpacked.005FC794	
00520D3A	83EC 5C	SUB ESP, 5C	
00520D3D	8A45 0B	MOV AL, BYTE PTR SS:[EBP+8]	
00520D40	53	PUSH EBX	
00520D41	56	PUSH ESI	
00520D42	57	PUSH EDI	
00520D43	8BF9	MOV EDI, ECX	
00520D45	6A 00	PUSH 0	
00520D47	8D4D DC	LEA ECX, DWORD PTR SS:[EBP-24]	
00520D4A	897D EC	MOV DWORD PTR SS:[EBP-14], EDI	
00520D4D	8845 DC	MOV BYTE PTR SS:[EBP-24], AL	
00520D50	E8 FE210000	CALL Unpacked.00522F53	
00520D55	BE 782A4B00	MOV ESI, Unpacked.004B2A78	timer
00520D5A	56	PUSH ESI	

_Ser of a "E x E cu te"

00520E40	C2 0400	RETN 4	
00520E43	B8 C4F66000	MOV EAX, Unpacked.0060F6C4	.δ\ra
00520E48	E8 47B90D00	CALL Unpacked.005FC794	
00520E4D	83EC 5C	SUB ESP, 5C	
00520E50	8A45 0B	MOV AL, BYTE PTR SS:[EBP+8]	
00520E53	53	PUSH EBX	
00520E54	56	PUSH ESI	
00520E55	57	PUSH EDI	
00520E56	8BF9	MOV EDI, ECX	
00520E58	6A 00	PUSH 0	
00520E5A	8D4D DC	LEA ECX, DWORD PTR SS:[EBP-24]	
00520E5D	897D EC	MOV DWORD PTR SS:[EBP-14], EDI	
00520E60	8845 DC	MOV BYTE PTR SS:[EBP-24], AL	
00520E63	E8 EB200000	CALL Unpacked.00522F53	
00520E68	BE 682A4B00	MOV ESI, Unpacked.004B2A68	execute
00520E6D	56	PUSH ESI	
00520E6E	E8 9DBA0D00	CALL Unpacked.005FC910	


Pat a n of the h

00520E40	C2 0400	RETN 4	
00520E43	C3	RETN	.δ\ra
00520E44	90	NOP	
00520E45	90	NOP	
00520E46	90	NOP	
00520E47	90	NOP	
00520E48	E8 47B90D00	CALL Unpacked.005FC794	
00520E4D	83EC 5C	SUB ESP, 5C	
00520E50	8A45 0B	MOV AL, BYTE PTR SS:[EBP+8]	
00520E53	53	PUSH EBX	
00520E54	56	PUSH ESI	
00520E55	57	PUSH EDI	
00520E56	8BF9	MOV EDI, ECX	
00520E58	6A 00	PUSH 0	
00520E5A	8D4D DC	LEA ECX, DWORD PTR SS:[EBP-24]	
00520E5D	897D EC	MOV DWORD PTR SS:[EBP-14], EDI	
00520E60	8845 DC	MOV BYTE PTR SS:[EBP-24], AL	
00520E63	E8 EB200000	CALL Unpacked.00522F53	
00520E68	BE 682A4B00	MOV ESI, Unpacked.004B2A68	execute
00520E6D	56	PUSH ESI	
00520E6E	E8 9DBA0D00	CALL Unpacked.005FC910	

_ **Save all** ... i also ensure any limit 1 more time ... If this ko Crack use is No. 1 for the game you should consult tut "**Unpacking ActiveMark level 2 entry point**" to see how to find the string "**setkey**" and think that the **Magic Call**.

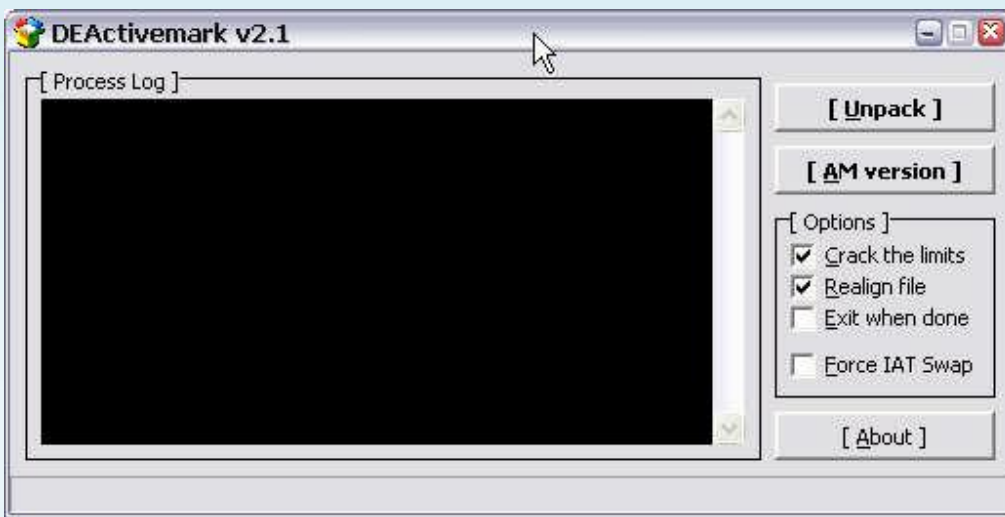
_ Tolerable if it is still lười ko unpack or Crack need for tools to use tired **ActiveMark 5.x Generic Loader**



_ N t han nu  Protect the selected file in **ActiveMark** ... Load into the game i

have always reminded **NAG** spent. But it only supports a game only.

_Moi Have more than **2.1 DeActivemark**



The _Lam also good but it is still only tools but should also do support all games ... some cases it is still the **MUP** solution

G r l Ee TsF italy Ou the Co mpu t e r A _ o f e l, e mbi Z o, M A B oo nb italy, H o acnh, Nina B e, e ki nman o

w ar, Z o i D e ux, M e r c, l o e ight to nix, T r o b icky italy, Takad a iamidi ot, of the handi e n e n t, C o r o

f ndZ ... and italy o u !

N ha that T, N g à 29 italy hang9 the click of 2 006

Wh o t italy N Bar



[MUP & Cracking] MoleBox Pro 2.6 Trial - Volume 1 - <Unpacking>

Contents Table:

I-Introduction

II-Asprotect Layer Bypass

III-Pro Unpacking MoleBox Layer:

Find-A OEP, Fix IAT Hide & Extract 2 modules

B-Fix dump File

C-dump 2 & Find Hidden Files stub

Ending Volume IV-1

Tools: OllyDBG + plugins, LordPE, ImportREC 1.6

Target: MoleBox Pro 2.6 Trial

Link: www.molebox.com

Skill Request: Basic Using Olly and some tools.

I. Introduction

Welcome you, the doctors and the aged. Giang's walking this day song too many, the trick of letting go đành should make "gut" off. But due to the 1 gang high in the trace, trick forced to dance for self-protection. 1 hope to make this way can help resolve the trick giang a blood debt. Đành not subject to the death only.

Today we will practice 1 half-dozen dishes half district. It is em "MoleBox Pro 2.6.0.2375 Trial." Because of the inheritance from that of British đăm 2.5x Mole should destroy them to do this will be suon yet. hope that through this article, we can say with MoleBox DONE. Let the promotion of take up, the battle will be quite long so long in one place is new. I, beginning ...

II.Bypass ASProtect Layer

Viewer, even the developers do Mole also confident with the Packer's aged. Layer we face is ASProtect 2.2 SKE:



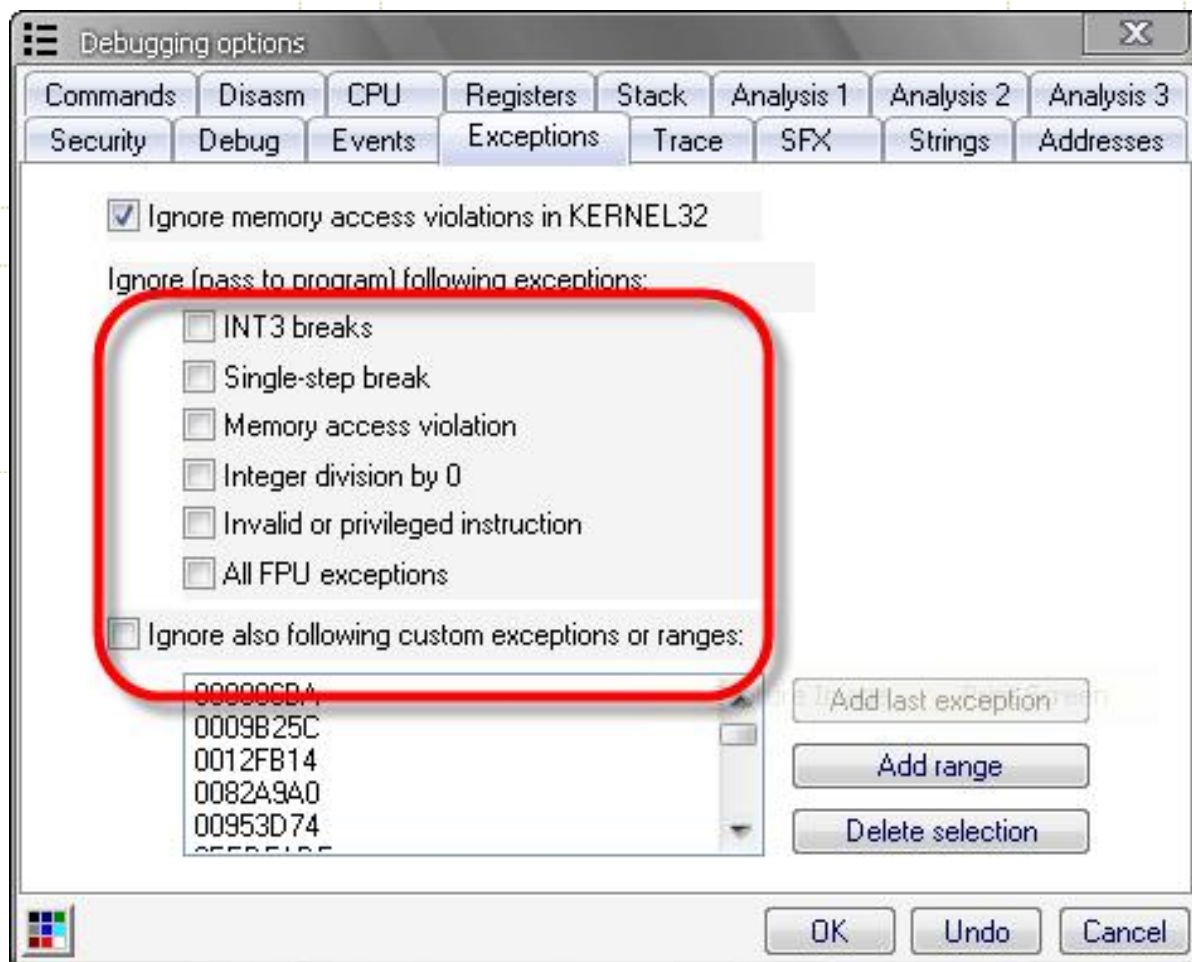
He way seats with carrying wild, calm hear presentations tí trick.

**Việc pack nhiều layer có thể xem là hay ,đồng thời cũng bộc lộ nhiều cái dở .Hay ở chỗ làm run tay các bác yếu tim và hạn chế các Tool unpack chuyên dụng .
Còn dở thì lại nhiều hơn cả cái hay .Ta hãy xét lại các tính năng cơ bản của đa số các packer là:**

- Encrypt Code
- Hide/Eliminate IAT
- Destroy Relocation (DLL/OCX)
- Compress Resource

Khi pack nhiều layer ,các công việc trên đều do layer đầu tiên kiểm nhiệm.Các layer sau chỉ đơn giản là encrypt "nhe" thêm 1 section code của loại packer đó (nếu có) nhằm hide EP của các layer trước. Và tận dụng các Anti-Debug của chính layer đó .

Meanwhile, SKE ASProtect most famous place in: [Hide / eliminate IAT](#). If it is in layer 2 (or $n! = 1$) Nothing is worth the fear. Experiment shows Mole always pack by itself. Therefore ASProtect sure to do the first layer. So, we take it with meat.
Of the familiar with the Exception in Olly when ASProtect 2.x:



Many of you or questions is "why should any Uncheck Ignore the Exception? Machines to take over. "Yes, we are almost in many tut, if only method but not hề have to issue" why ". Even writers sometimes Cuc know it is what ear, can they explain lười or simply 1 to follow the script kiddy train at. So, we pause to 1 walking through a few definitions:

Exception là các lỗi phát sinh trong quá trình Runtime. Để ngăn ngừa chương trình bị crash khi gặp các lỗi này, người lập trình cố gắng ghi lại để xử lý. Nếu xử lý thành công thì chương trình có thể tiếp tục chạy bình thường. Ngược lại thì sẽ gửi thông báo loại lỗi gì và gọi Exit Process để thoát chương trình. Cách làm những thao tác trên chính Handle Exception. (tạm dịch là điều khiển lỗi)

Có nhiều phương pháp để handle exception, phổ biến là SEH (Structured exception handling). SEH có thể dùng để xử lý cả 2 loại lỗi: exception cứng và mềm. Exception cứng thường phát sinh trong quá trình tính toán như lỗi chia không, tràn bộ đếm... Exception mềm như lỗi truy xuất con trỏ null, các tham số ko hợp lệ...

SEH ban đầu chỉ để dùng trong các chương trình code bằng C, nhưng về sau thì cho cả C++, MFC mặc dù theo lời khuyên của các chuyên gia thì: C++ exception handling nên dùng cho C++ và MFC exception handling nên dùng cho MFC. (các bạn tự tìm hiểu thêm nhé)

Với ASProtect, tác giả quá cẩn thận nên thêm nhiều SEH trong quá trình Unpack Code để kiểm soát mọi lỗi xảy ra, nếu ta Uncheck Ignore Exception thì trình debug sẽ bị ngắt lại mỗi lần gặp 1 đoạn SEH, và sau đoạn SEH cuối cùng cũng là lúc ASProtect đã unpack xong code, chuẩn bị access tới code chính và chúng ta thường lại

Now load the file you come in, we stopped at the EP's ASProtect 2.x:

Address	Hex dump	Disassembly
00401000	68 01004000	PUSH mbox2w.00440001
00401005	E8 01000000	CALL mbox2w.0040100B
0040100A	C3	RET
0040100B	C3	RET
0040100C	F9	STC
0040100D	9F	LAHF
0040100E	8B78 A2	MOV EDI,DWORD PTR DS:[EBX-5E]
00401011

Shift-F9 of times to run the target completely. It's best to do is detect that you do is set any memory nhé for any BP will create ASProtect Thread to check the BP here, do it again each time, so check in BP window view is that they remove all the previous and then run. On the computer trick 31 times Shift-F9 is run target, the Ctrl-F2 and to restart the implementation of 30 times to come:

Address	Hex dump	Disassembly
00BB9BF7	0156 00	ADD DWORD PTR DS:[ESI],EDX
00BB9BFA	CF	IRET
00BB9BFB	67:3D D20AFDA7	CMP EAX,A7FD0AD2
00BB9C01	66:67:64:8F06	POP WORD PTR FS:[0]
00BB9C08	83C4 04	ADD ESP,4
00BB9C0B	C1CE B9	ROR ESI,0B9
00BB9C0E	BE 6AEF4600	MOV ESI,46EF6A
00BB9C13	5E	POP ESI
00BB9C14	A1 0488BC00	MOV EAX,DWORD PTR DS:[BC8804]
00BB9C19	8B00	MOV EAX,DWORD PTR DS:[EAX]
00BB9C1B	8B68 1C	MOV EBP,DWORD PTR DS:[EAX+1C]
00BB9C1E	A1 0488BC00	MOV EAX,DWORD PTR DS:[BC8804]
00BB9C23	8B00	MOV EAX,DWORD PTR DS:[EAX]

We are in the code generated by ASProtect, Alt-M open Memory Map, "set breakpoint on memory access (MBOA) to 432,000 in the code section:

Address	Hex dump	Disassembly
00390000	00004000	
003A0000	00003000	
003B0000	000049000	
00400000	00001000	mbox2w
00401000	000023000	mbox2w
00424000	00001000	mbox2w
00425000	00005000	mbox2w
0042A000	00004000	mbox2w
0042E000	00001000	mbox2w
0042F000	00003000	mbox2w
00432000	00012000	mbox2w
00444000	00001000	mbox2w
00445000	00008000	mbox2w
0044D000	00010000	mbox2w
00467000	00001000	mbox2w

Continue Shift-F9 final, and we at Break:

Address	Hex dump	Disassembly
00433B23	E8 00000000	CALL mbox2w.00433B28
00433B28	60	PUSHAD
00433B29	E8 4F000000	CALL mbox2w.00433B7D
00433B2E	868CE2 3E810F7	XCHG BYTE PTR DS:[EDX+740F813E],CL
00433B35	57	PUSH EDI
00433B36	CE	INTO
00433B37		INC ESI

This trick will hustler from this point is, you remove the MBOA néh. Now you can use methods in ASPack unpack UPX or to find the OEP. That Trace F7 through me orders PUSHAD à Follow in dump to record ESP à break on the access Byte ...

But Mole is quite easy to find the OEP 1 otherwise. I enter the command CALL 2:

Address	Hex dump	Disassembly
00433B23	E8 00000000	CALL mbox2w.00433B28
00433B28	60	PUSHAD
00433B29	E8 4F000000	CALL mbox2w.00433B7D
00433B2E	868CE2 3E810F7	XCHG BYTE PTR DS:[EDX+740F813E],CL
00433B35	57	PUSH EDI
00433B36	CE	INT0
00433B37	46	INC ESI
00433B37	57	CALL EBP

Will come:

00433B66	CF	TEST EAX,2454155C
00433B67	A9 5C155424	OUT 9B,AL
00433B6C	E6 9B	OUT 9B,AL
00433B6E	✓ E9 2A690000	JMP mbox2w.0043A49D
00433B73	✓ E9 3E690000	JMP mbox2w.0043A4B6
00433B78	✓ E9 39690000	JMP mbox2w.0043A4B6
00433B7D	E8 6EFBFFFF	CALL mbox2w.00433BF0
00433B82	✓ 7E 04	JLE SHORT mbox2w.00433B88
00433B84	0100	ADD DWORD PTR DS:[EAX],EAX
00433B8C	0E	SCAS BYTE PTR ES:[EDI]

Continue to enter it, is to:

004336EB	5D	POP EBP
004336EC	C3	RET
004336ED	CC	INT3
004336EE	CC	INT3
004336EF	CC	INT3
004336F0	E8 EBF8FFFF	CALL mbox2w.004332E0
004336F5	58	POP EAX
004336F6	E8 55070000	CALL mbox2w.00433E50
004336FB	58	POP EAX
004336FC	894424 24	MOV DWORD PTR SS:[ESP+24],EAX
00433700	61	POPAD
00433701	58	POP EAX
00433702	58	POP EAX
00433703	FFD0	CALL EAX
00433705	CC	INT3
0043370A	CC	INT3
0043370B	CC	INT3

Seats fold in the region, we set a BP on it (F2). When Mole unpack code will complete the questions in order Break CALL EAX before calling it to OEP. And here is where to start Inline Patch if you like.

Through the MaDMAN_H3rCuL3s tut of the Mole ARTeam 2.5x will Extract from Module 2 in the unpack code. And also in the 2.6. We used to function CreateFileA summary of this module. BP set CreateFileA to do:

Command : bp CreateFileA

Shift-F9 first a break, corn down the window stack:

Address	Value	Comment
0012F90C	0043941D	CALL to CreateFileA from mbox2w.00439417
0012F910	00042108	FileName = "D:\TEMP2\MOLEBOXPRO_2.6\MBOX2W.EXE"
0012F914	80000000	Access = GENERIC_READ
0012F918	00000001	ShareMode = FILE_SHARE_READ
0012F91C	00000000	pSecurity = NULL
0012F920	00000003	Mode = OPEN_EXISTING
0012F924	00000000	Attributes = 0
0012F928	00000000	hTemplateFile = NULL

Still not at the Mole extract module, continue Shift-F9 a break:

Address	Value	Comment
0012FB98	0043638E	CALL to CreateFileA from mbox2w.00436388
0012FB9C	00042438	FileName = "D:\TEMP2\MOLEBOXPRO_2.6\MBOX2W.EXE"
0012FBA0	80000000	Access = GENERIC_READ
0012FBA4	00000001	ShareMode = FILE_SHARE_READ
0012FBA8	00000000	pSecurity = NULL
0012FBAC	00000003	Mode = OPEN_EXISTING
0012FBB0	00000000	Attributes = 0
0012FBB4	00000000	hTemplateFile = NULL

The one is the right Shift-F9 a break. If not, add 1 time Shift-F9 a break again:

Address	Value	Comment
0012FA4C	0043C8A4	CALL to CreateFileA from mbox2w.0043C89E
0012FA50	00042888	FileName = "C:\DOCUMENTS\TRICKY\1\LOCALS\1\TEMP\MBX@E64@D42668.###"
0012FA54	40000000	Access = GENERIC_WRITE
0012FA58	00000000	ShareMode = 0
0012FA5C	00000000	pSecurity = NULL
0012FA60	00000002	Mode = CREATE_ALWAYS
0012FA64	00000000	Attributes = 0
0012FA68	00000000	hTemplateFile = NULL

Ah ha, this is the Mole prepare extract the first module to TEMP directory here. Click on the right lines and FileName Follow dump in it:



Looking through the window of a dump Byte Name of file has been encrypted:

Hex dump	ASCII
43 3A 5C 44 4F 43 55 4D 45 7E 31 5C 54 52 49 43	C:\DOCUMENTS\TRICKY\1\LOCALS\1\TEMP\MBX@E64@D42668.###
4B 59 7E 31 5C 4C 4F 43 41 4C 53 7E 31 5C 54 45	
4D 50 5C 4D 42 58 40 45 36 34 40 44 34 32 36 36	
38 2E 23 23 23 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

From here we pull up to see if:

Hex dump	ASCII
06 00 45 00 83 01 00 00 44 3A 5C 54 45 4D 50 32	..E..D:\TEMP2\MOLEBOXPRO_2.6\MBOX2W.EXE.....
5C 4D 4F 4C 45 42 4F 58 50 52 4F 5F 32 2E 36 5C	
4D 42 4F 58 32 57 2E 45 58 45 00 00 00 00 00	
17 00 06 00 89 01 08 00 4D 42 4F 58 32 5F 42 4C	..E..MBOX2_BLACKLIST.TXT.MBOX2_BOOTUPDBGLTDEM
41 43 4B 4C 49 53 54 2E 54 58 54 00 4D 42 4F 58	
32 5F 42 4F 4F 54 55 50 44 42 47 4C 54 44 45 4D	
4F 00 4D 42 4F 58 32 5F 42 4F 4F 54 55 50 4C 54	
44 45 4D 4F 00 4D 53 4B 49 4E 43 4F 52 45 2E 44	
4C 4C 00 4D 53 56 43 58 36 30 2E 44 4C 4C 00 50	
04 00 04 E4 03 00 00 00 00 00 00 00 00 00 00	

That list of files that may Mole Extract is:

- newcp60.dll
- MSkinCore.dll
- mbox2_bootupltdemo
- mbox2_bootupdbgltdemo
- mbox2_blacklist.txt

Address	Hex dump	Disassembly
7C80B529	8BFF	MOV EDI,EDI
7C80B52B	55	PUSH EBP
7C80B52C	8BEC	MOV EBP,ESP
7C80B52E	837D 08 00	CMP DWORD PTR SS:[EBP+8],0
7C80B532	74 18	JE SHORT kernel32.7C80B54C
7C80B534	FF75 08	PUSH DWORD PTR SS:[EBP+8]
7C80B537	E8 682D0000	CALL kernel32.7C80E2A4
7C80B53C	85C0	TEST EAX,EAX
7C80B53E	74 08	JE SHORT kernel32.7C80B548
7C80B540	FF70 04	PUSH DWORD PTR DS:[EAX+4]
7C80B543	E8 F4300000	CALL kernel32.GetModuleHandleW
7C80B548	5D	POP EBP
7C80B549	C2 0400	RET 4
7C80B54C	64:A1 180000	MOV EAX,DWORD PTR FS:[18]
7C80B550	8B40 30	MOV EAX,DWORD PTR DS:[EAX+30]
7C80B553	8B40 08	MOV EAX,DWORD PTR DS:[EAX+8]

Un-Break Point here to go (for Bread F2). Open memory map will be notified 1 trick used by the plugin to ignore OllyAdvanced notice that it should do is consult more. I find the section of the module msvcp60.dll:

77EFC000	00005000	RPCRT4	.reloc	code	Set Break on Access
77F10000	00001000	GDI32		PE h	
77F11000	00042000	GDI32	.text	code	Set memory breakpoint on access
77F53000	00001000	GDI32	.data	code	Set memory breakpoint on write
77F54000	00001000	GDI32	.rsrc	code	Set access
77F55000	00002000	GDI32	.reloc	code	
780C0000	00001000	msvcp60		PE h	
780C1000	0002A000	msvcp60	.text	code	Allocate Memory
780EB000	00030000	msvcp60	.rdata	code	
78118000	00002000	msvcp60	.data	code	
7811D000	00001000	msvcp60	.rsrc	code	
7811F000	00001000	msvcp60			

Set in MBOA section. Text as in the picture. Áy, not with F9 or Shift-F9. Not understand at Olly on trick run too fast or by OEP is now beginning section. Text but when you press F9 now it ào ào This is the last module, including pieces Break any time. What you are doing is like this, usually when the code is unpack, or borrow more to function, in dõ em VirtualProtect or used. Try it to delay the process the implementation, may be the break in the OEP's modules. BP VirtualProtect we go:

Command : bp VirtualProtect

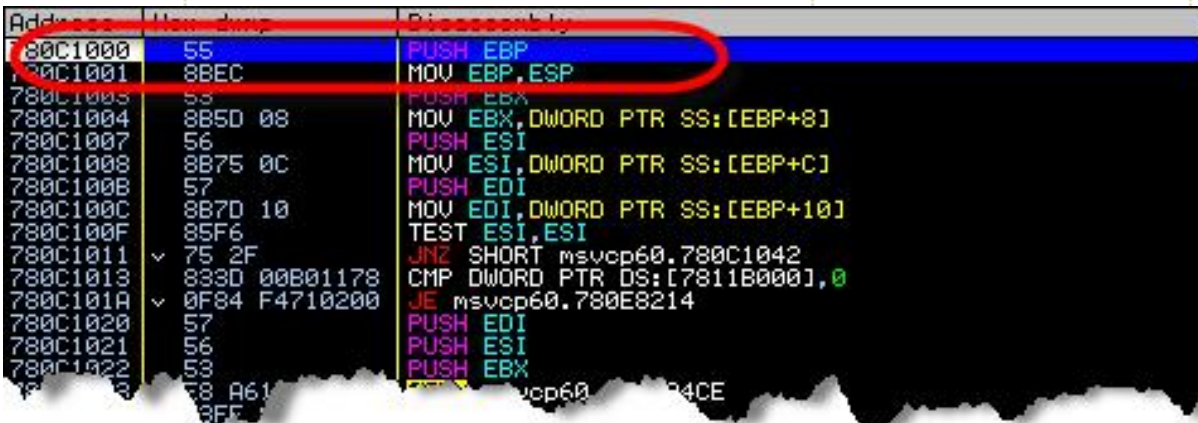
Now, the Shift-F9 to break from at this function, first, we dom down window stack:

Address	Value	Comment
0012F1CC	0043B899	CALL to VirtualProtect from mbox2w.0043B893
0012F1D0	780C01D8	Address = msvc60.780C01D8
0012F1D4	00000005	Size = 5
0012F1D8	00000004	NewProtect = PAGE_READWRITE
0012F1DC	0012F228	pOldProtect = 0012F228
0012F1E0	00000001	
0012F1E4	0012F29C	
0012F1E8	78121000	OFFSET msvc60.<ModuleEntryPoint>

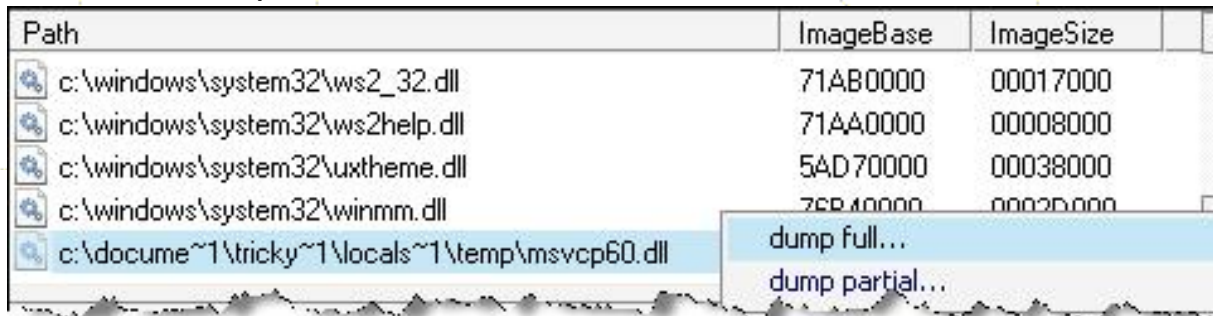
Ah, it is set up under the READWRITE PE Header section contains the modules, continue to Shift-F9:

Address	Value	Comment
0012F1CC	0043B951	CALL to VirtualProtect from mbox2w.0043B94B
0012F1D0	780C1000	Address = msvc60.780C1000
0012F1D4	0002A000	Size = 2A000 (172032.)
0012F1D8	00000020	NewProtect = PAGE_EXECUTE_READ
0012F1DC	0012F220	pOldProtect = 0012F220
0012F1E0	00000001	
0012F1E4	0012F29C	
0012F1E8	78121000	OFFSET msvc60.<ModuleEntryPoint>

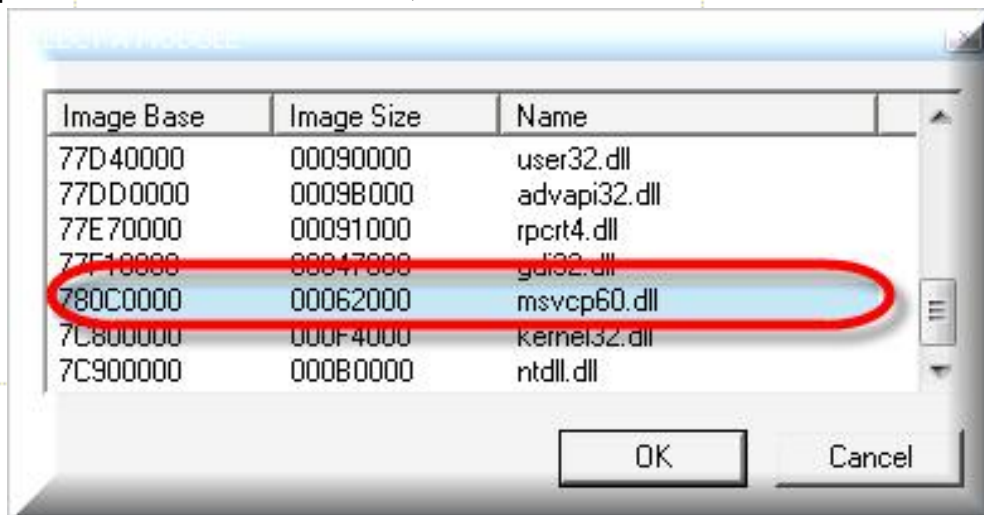
Set up under the EXECUTE_READ section. Text of the module. Keep Shift-F9 until it set up all of the attributes of the section will be at the OEP Break:



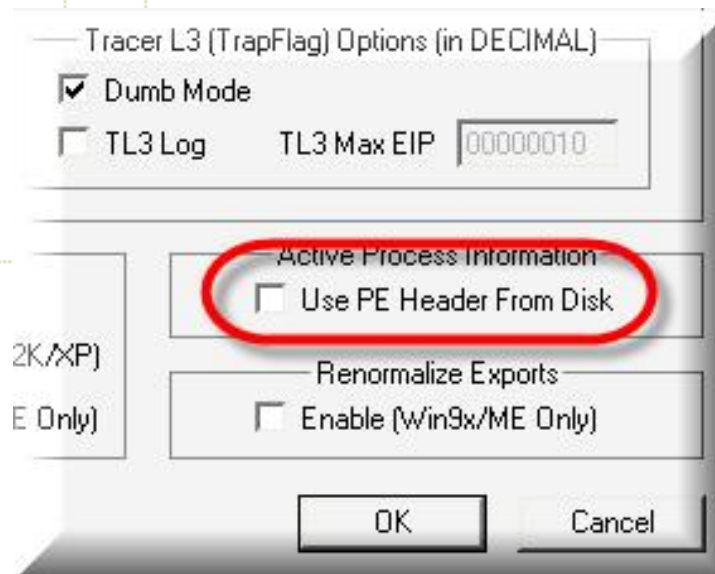
So OEP by this module is now top of the right section. RVA text = 780C1000, hen or what it crawled too fast now I leave this section MBOA in farewell. Then they dump out this module, LordPE trick used to dump:



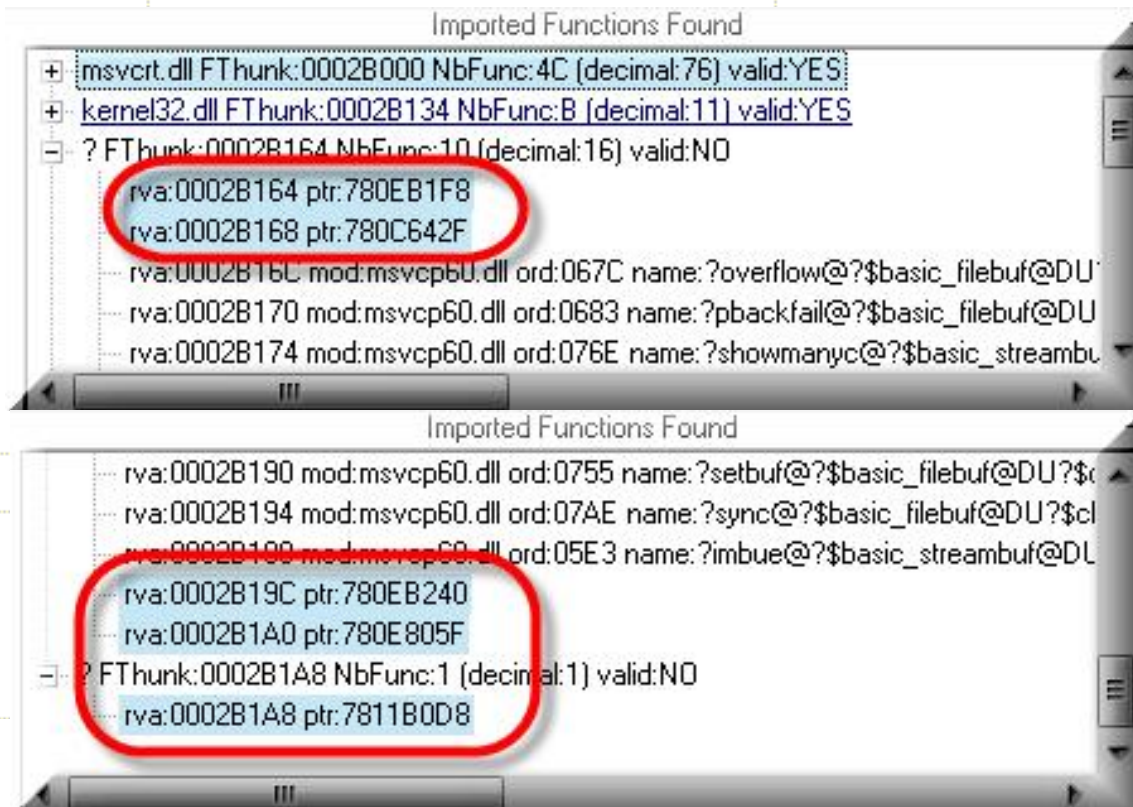
Save with the name: msvc60_dump.dll han. Sau not fix it with ImportREC IAT. ImportREC Open, select the process needs to handle, dll Pick it:



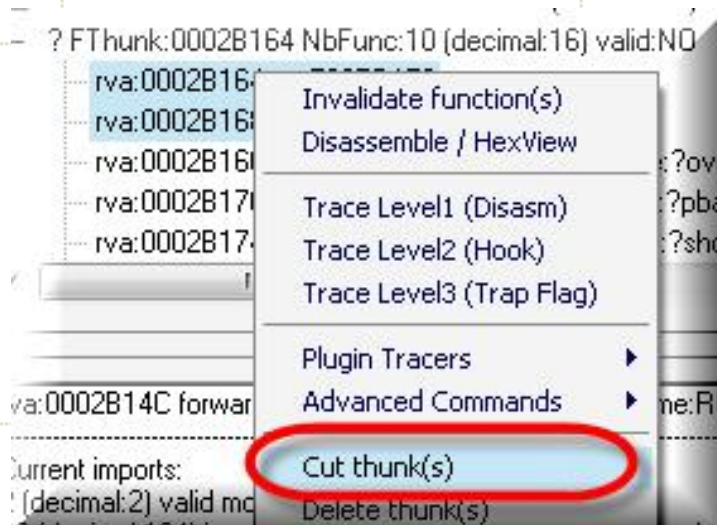
OK play, remember that previously, in the Import Option to uncheck this nhé:



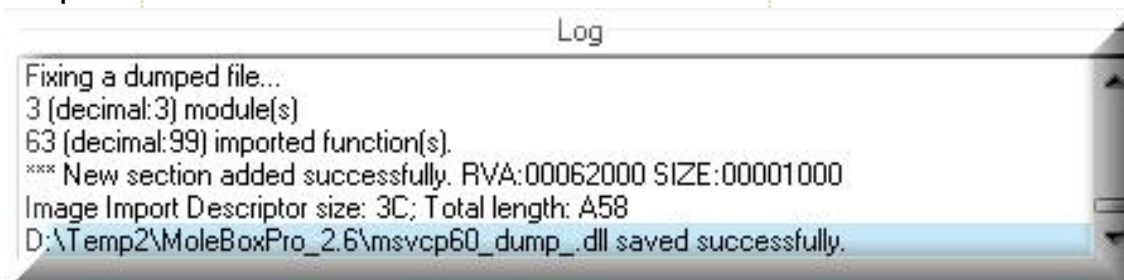
Why must the tut in the Armadillo_DLL_Unpacking_n_More was discussed. OEP = Enter 1000, click Search IAT à Get Auto Import. Continue pressing Invalid Show, here are some invalid value:



If the tut of the MaDMAN_H3rCuL3s we should choose Dissamble / Hex View in the value of Invalid module to find exactly the IAT is Hide. But do know we have aged lăm or ko, where almost Mole still do not interfere to Hide / eliminate the IAT. And to ensure that food, you can check by a search msvc60.dll other (to do this module is designed for Mole, it is a module for the app code in C + +). Version 2.6 uses the Mole is 6.0.8972.0, but only in comparison with the version 6.0.8168.0 I also found that i have more time in the IAT position on. So that an Cut Thunk (s) all:



Then fix the dump is then:

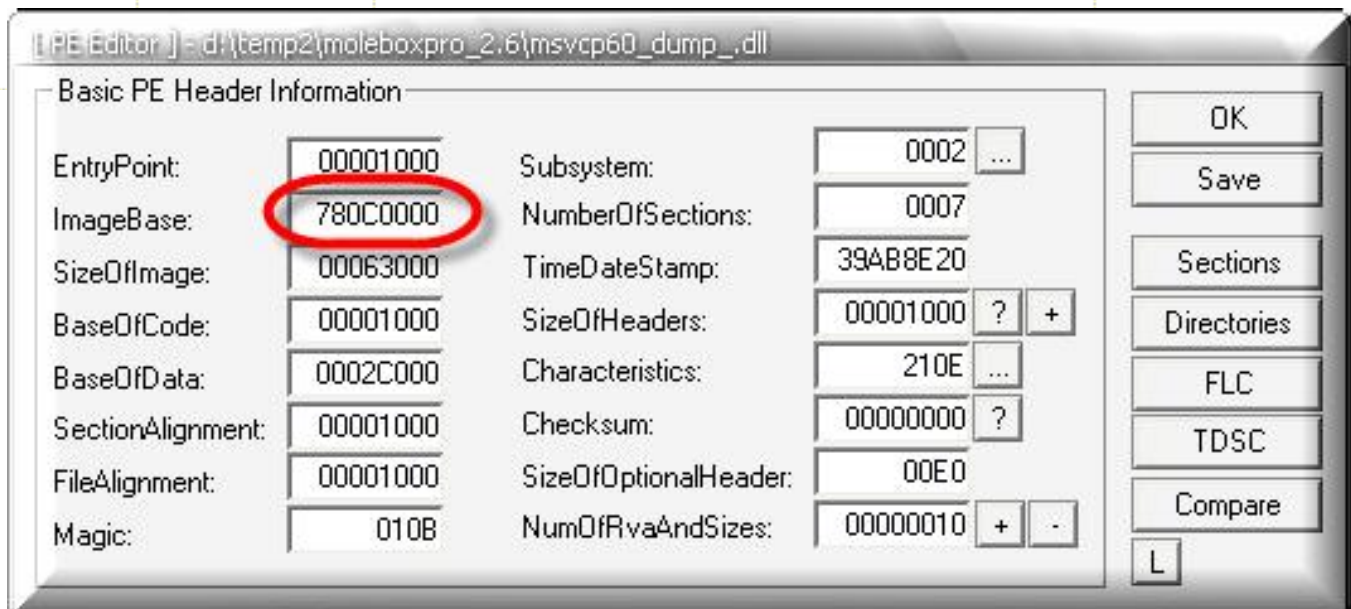


The still have not finished with this first module, remember when unpack 1 dll outside the OEP, IAT, the third important is the Relocation and ImageBase. Why do you consider Armadillo_DLL_Unpacking_n_More tut, the trick also presented several concepts of 1 to unpack the dll file.

First is Imagebase, we see the dump file has at any ImageBase, back in the memory map Olly:

77F54000	00001000	GDI32	.rsrc	resources	Imag	R	RWE
77F55000	00002000	GDI32	.reloc	relocations	Imag	R	RWE
780C0000	00001000	msvc60		PE header	Imag	R	RWE
780C1000	0002A000	msvc60	.text	code	Imag	R	RWE
780EB000	00030000	msvc60	.rdata	exports	Imag	R	RWE
7811B000	00002000	msvc60	.data		Imag	R	RWE
7811D000	00001000	msvc60	.rsrc	resources	Imag	R	RWE
7811E000	00003000	msvc60	.reloc		Imag	R	RWE
78121000	00001000	msvc60	_BOX_	SFX,relocat	Imag	R	RWE
7C900000	00001000	kernel32		PE header	Imag	R	RWE

Add the first load of PE Header modules ImageBase is also the need to find, so IM = 780C0000. Use PE Editor's LordPE file to open the dump has fix this:



So i need to edit, IM is also in memory IM default file. But still Relocation, click on the button right now Sections:

Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	00029BBC	00001000	00029BBC	E0000080
.rdata	0002B000	0002F030	0002B000	0002F030	E0000080
.data	0005B000	00001788	0005B000	00001788	E0000080
.rsrc	0005D000	000003A8	0005D000	000003A8	E0000080
.reloc	0005E000	00002CF0	0005E000	00002CF0	E0000080
__BOX__	00061000	00001000	00061000	00001000	E0000020
.mact	00062000	00001000	00062000	00001000	E0000060

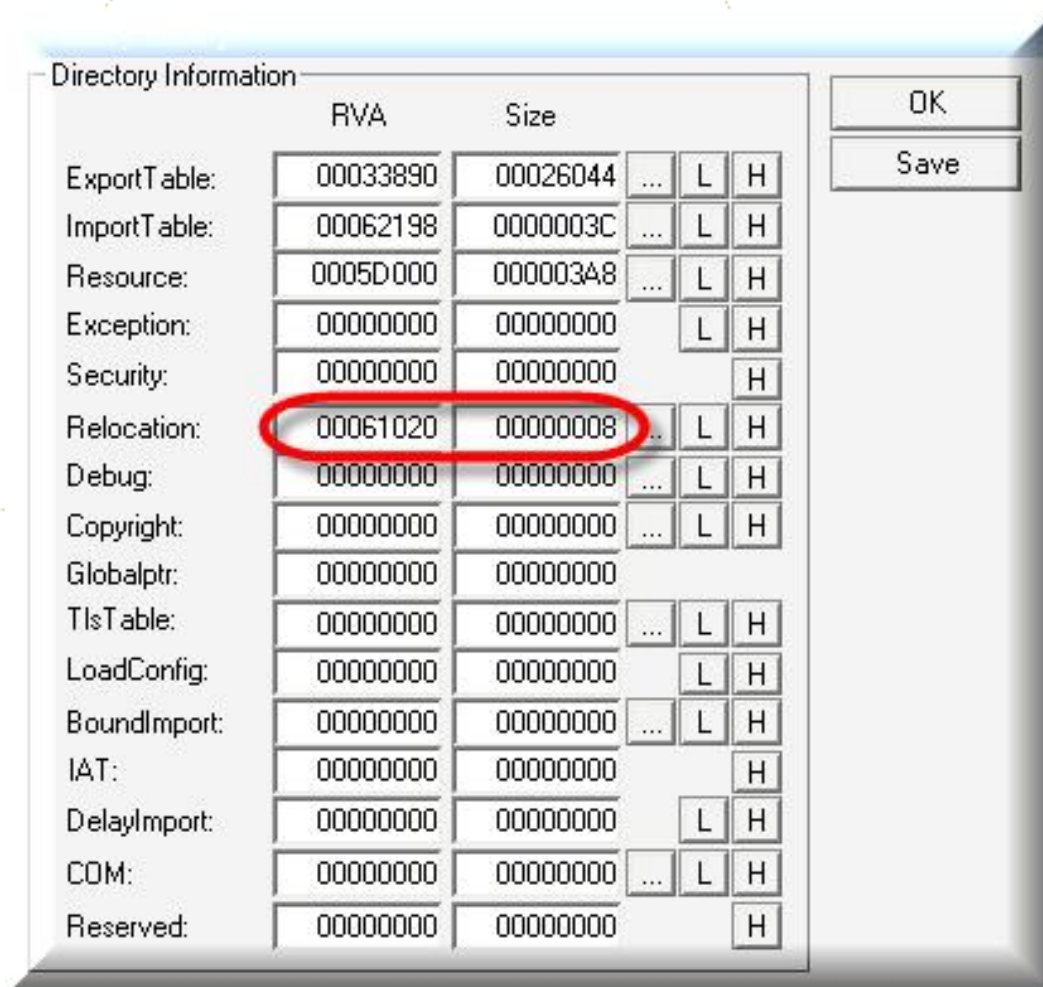
5E000 that start section contains Relocation, next to the actual size of it. Be careful then should we define the memory of food for sure. We see Relocation start here:

7811E000	00 10 00 00	B8 01 00 00	15 30 4C 30	83 30 9B 30	..f0..30L050<0
7811E010	A7 30 AD 30	B8 30 C5 30	CD 30 DB 30	E0 30 E5 30	00:0A0+0=0000000
7811E020	EA 30 F5 30	06 31 10 31	25 31 31 31	37 31 42 31	00J0+1>1%1117B1
7811E030	4B 31 53 31	5E 31 66 31	71 31 79 31	84 31 8C 31	K1S1^1f1q1y1a1i1
7811E040	97 31 9F 31	AA 31 B2 31	B0 31 C5 31	D0 31 D8 31	01f1-1#1^1+1^1

7811E000 - IM: 780C0000 = 5E000. And in the end:

78120C70	E0 3B 38 3C	90 3C E0 3C	30 3D 70 3D	B0 3D F0 3D	α;8<ε<α<0=p=≡≡≡
78120C80	30 3E 70 3E	B0 3E D0 3E	F0 3E 10 3F	30 3F 50 3F	0>p>≡>≡>≡?0?P?
78120C90	70 3F 90 3F	E8 3F 00 00	00 C0 05 00	58 00 00 00	p?ε?ε?...L.X...
78120CA0	40 30 98 30	F0 30 10 31	30 31 58 31	80 31 A8 31	00Y0=0>101X1C1i1
78120CB0	00 31 28 32	80 32 D8 32	30 33 88 33	E0 33 38 34	1(202+203ε3α384
78120CC0	90 34 B0 34	D0 34 F0 34	10 35 14 35	18 35 1C 35	ε4:4^4=4>5^5+5L5
78120CD0	20 35 24 35	28 35 2C 35	30 35 34 35	38 35 3C 35	5ε5(5,5054585<5
78120CE0	40 35 44 35	50 35 70 35	90 35 B0 35	D0 35 00 00	@5D5P5p5ε55^5..
78120CF0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
78120D00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
78120D10	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

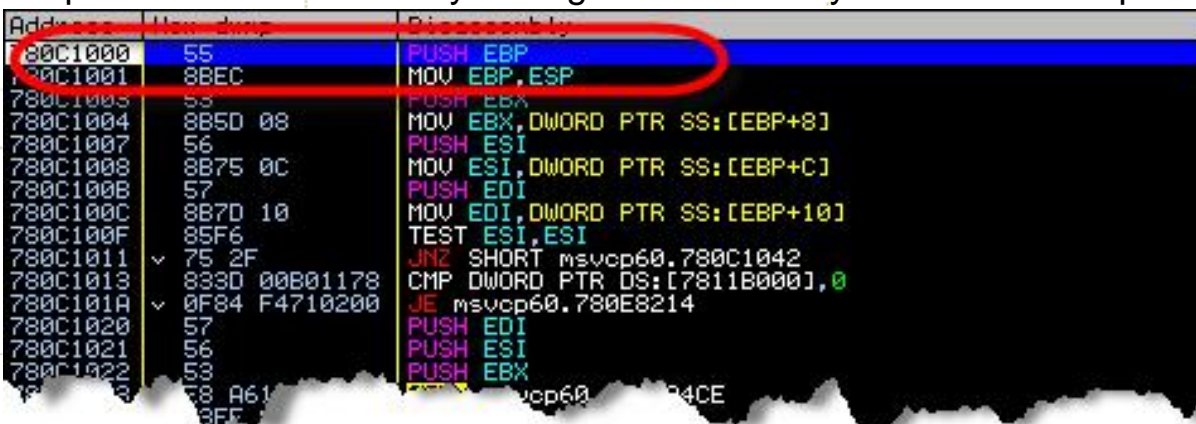
78120CF0 - 7811E000 = 2CF0. Thus the starting address and Size of Relocation are correct. Back PE Editor, close the section, choose to Directories:



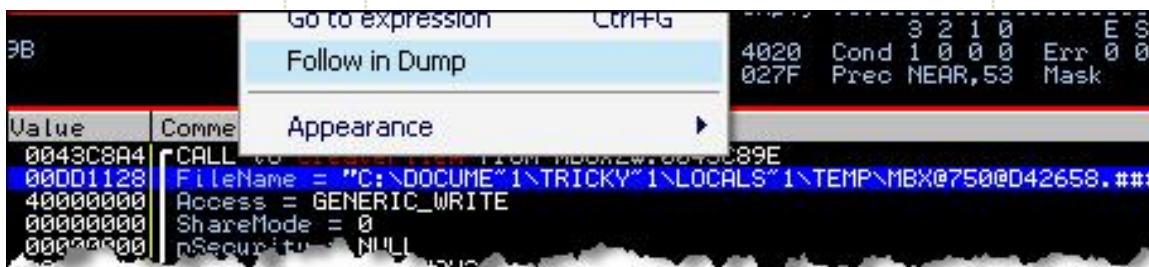
Where fold region must correct, and loaded Offset Size has to be on:



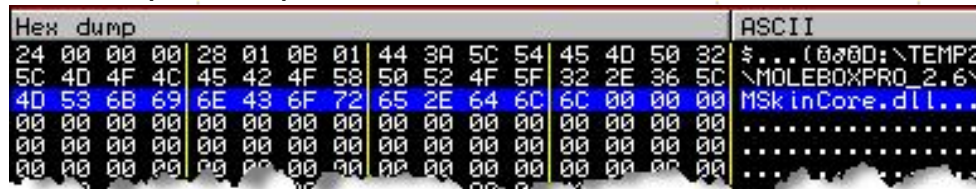
Save and Save again. Rename the file Fix dump into its original name: msvcp60.dll. The module is completed first and then đây. You go back and Olly continued from previous steps:



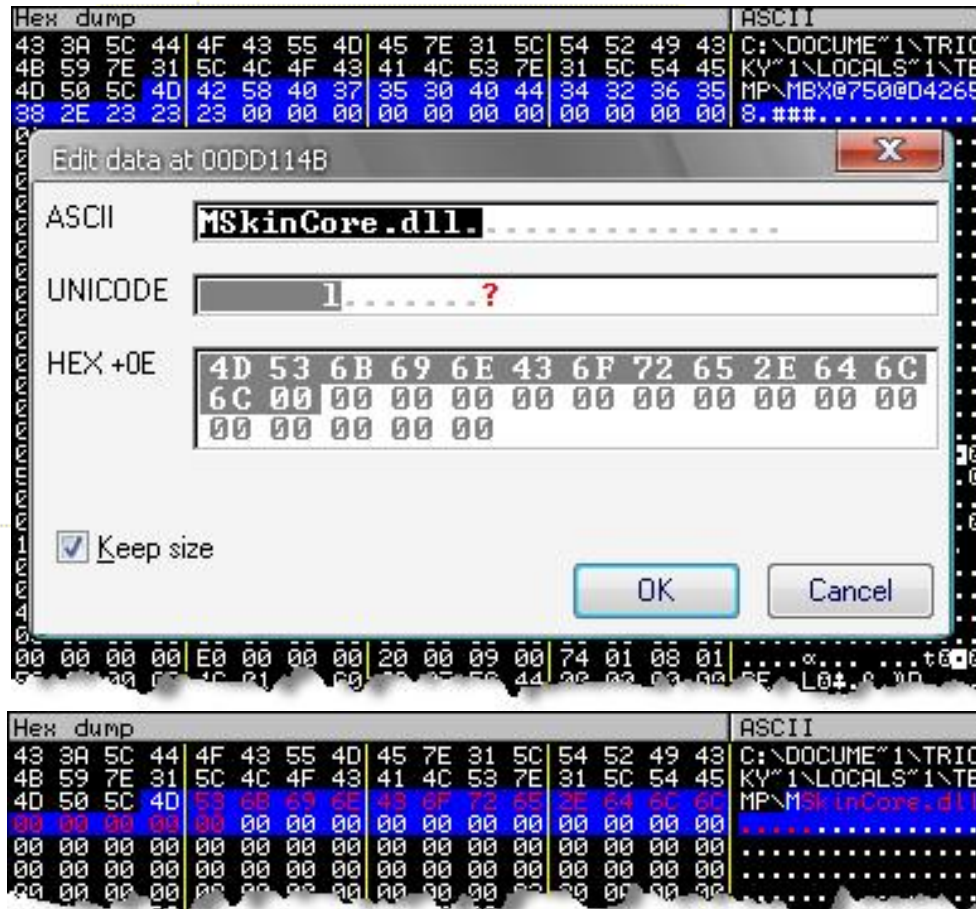
Remember that BP has set VirtualProtect before, time we put it in BC VirtualProtect. If BP initially set at CreateFileA still make you press Shift-F9, Olly will break down and dom Stack for summary module 2:



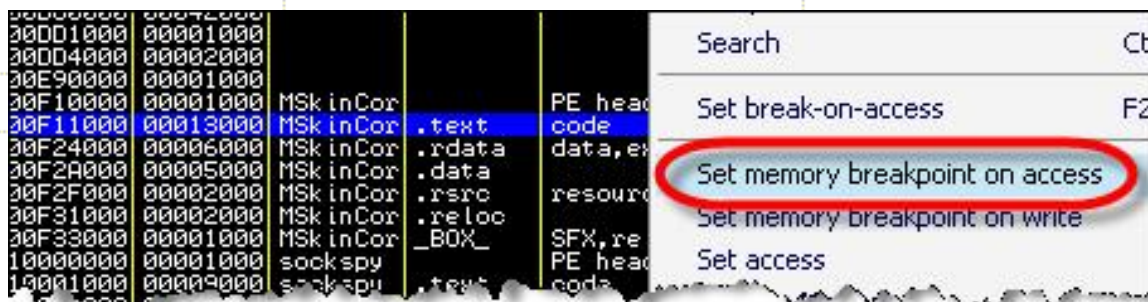
Follow also in line at the dump Filename (as in the picture). Dom wa window dump will see encrypted file name, then pulled up on the 1 billion:



May too, Byte of the original file name has not been overwritten, so fix the encrypted file back to the previous name:



Leave CreateFileA in BP and BP GetModuleHandleA similar to break immediately after the file load into memory. Even then, put in BP GetModuleHandleA go to the memory map and set in MBOA section. Text of this module:



However, in this module, month Male ... flipped out, Mole will conduct Hide / eliminate IAT. Tut by Hec aged Cu Lec to do only one way to find the accuracy of the IAT Hide / eliminate this. The new security aged lăng nhach, the module will do is what they go mò the function of the bay add, still aged only the need to do only. Teu real!

There is a trick used by + NCR tip. Before Mole conducted Hide IAT, it will call VirtualProtect function, so we should set the BP VirtualProtect to find accommodation will destroy our IAT. Once set, the Shift-F9 to run at the top and Break functions:

Address	Hex dump	Disassembly
7C801AD0	8BFF	MOV EDI,EDI
7C801AD2	55	PUSH EBP
7C801AD3	8BEC	MOV EBP,ESP
7C801AD5	FF75 14	PUSH DWORD PTR SS:[EBP+14]
7C801AD8	FF75 10	PUSH DWORD PTR SS:[EBP+10]
7C801ADB	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
7C801ADE	FF75 08	PUSH DWORD PTR SS:[EBP+8]
7C801AE1	6A FF	PUSH -1
7C801AE3	E8 75FFFFFF	CALL kernel32.VirtualProtectEx
7C801AE8	5D	POP EBP
7C801AE9	C2 1000	RET 10
7C801AEC	90	NOP
7C801AF0	90	NOP

Dom down the stack, will see 1 address:

Address	Value	Comment
0012F2AC	0043AFEC	CALL to VirtualProtect from mbox2w.0043AFEC
0012F2B0	00F24018	Address = MSkinCor.00F24018
0012F2B4	00000004	Size = 4
0012F2B8	00000004	NewProtect = PAGE_READWRITE
0012F2BC	0012F2C0	OldProtect = 0012F2C0
0012F2C0	00000000	
0012F2C4	00000000	

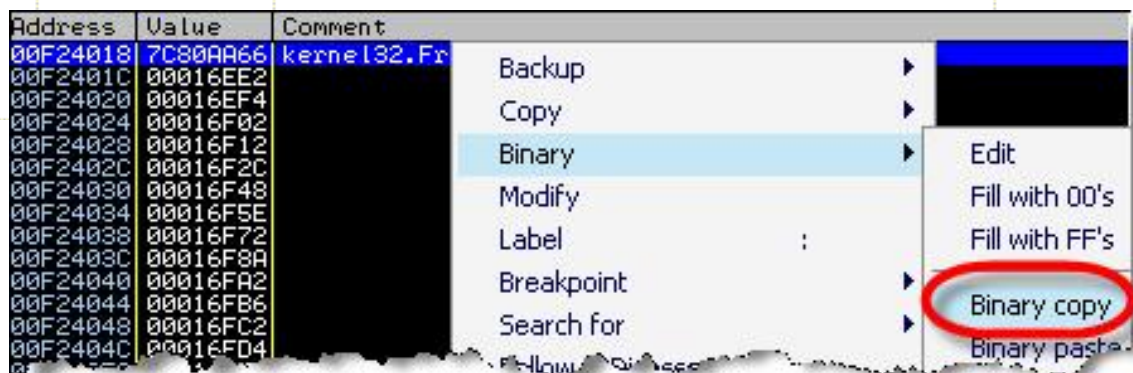
Follow in dump in this line, will see the value IAT first move was to memory:

Address	Hex dump
00F24018	66 AA 80 7C E2 6E 01 00 F4 6E 01 00 02 6F 01 00
00F24028	12 6F 01 00 2C 6F 01 00 48 6F 01 00 5E 6F 01 00
00F24038	72 6F 01 00 8A 6F 01 00 A2 6F 01 00 B6 6F 01 00
00F24048	00 6F 01 00 00 6F 01 00 00 6F 01 00 00 6F 01 00

Click right, select Address Long to see that more clearly:

Address	Value	Comment
00F24018	7C80AA66	kernel32.FreeLibrary
00F2401C	00016EE2	
00F24020	00016EF4	
00F24024	00016F02	
00F24028	00016F12	

This is the value before IAT canceled, Click to time in this line, choose:



How to Backup the value of this. Now we trace to the function of RETN, to return to it, we come:

Address	Hex dump	Disassembly
0043AFEC	85C0	TEST EAX,EAX
0043AFEE	75 0A	JNZ SHORT mbox2w.0043AFFA
0043AFF0	B9 0B0000EF	MOV ECX,EF00000B
0043AFF5	E8 5F2D0000	CALL mbox2w.0043DD59
0043AFFA	8B40 08	MOV ECX,DWORD PTR SS:[EBP+8]
0043AFFD	8B55 F8	MOV EDX,DWORD PTR SS:[EBP-8]
0043B000	8B02	MOV EAX,DWORD PTR DS:[EDX]
0043B002	8901	MOV DWORD PTR DS:[ECX],EAX
0043B004	8D4D F4	LEA ECX,DWORD PTR SS:[EBP-C]
0043B007	51	PUSH ECX
0043B008	8B55 F0	MOV EDX,DWORD PTR SS:[EBP-10]
0043B00B	52	PUSH EDX
0043B00C	6A 04	PUSH 4
0043B00E	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]
0043B011	50	PUSH EAX

Now continue to trace me to this command:

Address	Hex dump	Disassembly
0043AFEC	85C0	TEST EAX,EAX
0043AFEE	75 0A	JNZ SHORT mbox2w.0043AFFA
0043AFF0	B9 0B0000EF	MOV ECX,EF00000B
0043AFF5	E8 5F2D0000	CALL mbox2w.0043DD59
0043AFFA	8B40 08	MOV ECX,DWORD PTR SS:[EBP+8]
0043AFFD	8B55 F8	MOV EDX,DWORD PTR SS:[EBP-8]
0043B000	8B02	MOV EAX,DWORD PTR DS:[EDX]
0043B002	8901	MOV DWORD PTR DS:[ECX],EAX
0043B004	8D4D F4	LEA ECX,DWORD PTR SS:[EBP-C]
0043B007	51	PUSH ECX
0043B008	8B55 F0	MOV EDX,DWORD PTR SS:[EBP-10]
0043B00B	52	PUSH EDX
0043B00C	6A 04	PUSH 4
0043B00E	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]
0043B011	50	PUSH EAX

Observations IAT value at this time, it is still raw:

Address	Value	Comment
00F24018	7C80AA66	kernel32.FreeLibrary
00F2401C	00016EE2	
00F24020	00016EF4	
00F24024	00016F02	
00F24028	00016F12	

But after Trace me through this command are:

Address	Hex dump	Disassembly
0043AFEC	85C0	TEST EAX,EAX
0043AFEE	75 0A	JNZ SHORT mbox2w.0043AFFA
0043AFF0	B9 0B0000EF	MOV ECX,EF00000B
0043AFF5	E8 5F2D0000	CALL mbox2w.0043DD59
0043AFFA	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
0043AFFD	8B55 F8	MOV EDX,DWORD PTR SS:[EBP-8]
0043B000	8B02	MOV EAX,DWORD PTR DS:[EDX]
0043B002	8901	MOV DWORD PTR DS:[ECX],EAX
0043B004	8D4D F4	LEA ECX,DWORD PTR SS:[EBP-C]
0043B007	51	PUSH ECX
0043B008	8B55 F0	MOV EDX,DWORD PTR SS:[EBP-10]
0043B00B	52	PUSH EDX
0043B00C	6A 04	PUSH 4
0043B00E	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]
0043B011	50	PUSH EAX

Address	Value	Comment
00F24018	004408E7	mbox2w.004408E7
00F2401C	00016EE2	
00F24020	00016EF4	
00F24024	00016F02	
00F24028	00016F12	
00F2402C	00016F2C	
00F24030	00016F48	
00F24034	00016F5E	
00F24038	00016F72	

Asia Ah, this old Male dare cancel IAT front of us there the other. Note concept "Cancel" here ko means completely destroyed. It covers always notice encrypted again. For example, in above, refers to the value CALL IAT original has been replaced by a CALL to address 1 (4408E7) in code decrypt by Mole.

Now Click right at the IAT canceled Paste the values that we have now to Backup:

Address	Value	Comment
00F24018	7C80A966	kernel32.FreeLibrary
00F2401C	00016EE2	
00F24020	00016EF4	
00F24024	00016F02	
00F24028	00016F12	
00F2402C	00016F2C	

How color mè nay hour only for you to see Mole IAT cancel any place, when? But if you continue to work with such a long thoong IAT loong the only country disappear. Observations by the orders we've Trace through, every time IAT canceled orders on the question is enforcement. But the value in ECX day will then be replaced by me orders: LEA ECX, for the other. So the wish to prevent the cancellation IAT simply NOP he was going on but do not fear the influence of the ECX:

Address	Hex dump	Disassembly
0043AFEC	85C0	TEST EAX,EAX
0043AFEE	75 0A	JNZ SHORT mbox2w.0043AFFA
0043AFF0	B9 0B0000EF	MOV ECX,EF00000B
0043AFF5	E8 5F2D0000	CALL mbox2w.0043DD59
0043AFFA	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
0043AFFD	8B55 F8	MOV EDX,DWORD PTR SS:[EBP-8]
0043B000	8B02	MOV EAX,DWORD PTR DS:[EDX]
0043B002	90	NOP
0043B003	90	NOP
0043B004	8D4D F4	LEA ECX,DWORD PTR SS:[EBP-C]
0043B007	51	PUSH ECX
0043B008	8B55 F0	MOV EDX,DWORD PTR SS:[EBP-10]
0043B00B	52	PUSH EDX
0043B00C	6A 04	PUSH 4
0043B00E	8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]
0043B011	50	PUSH EAX

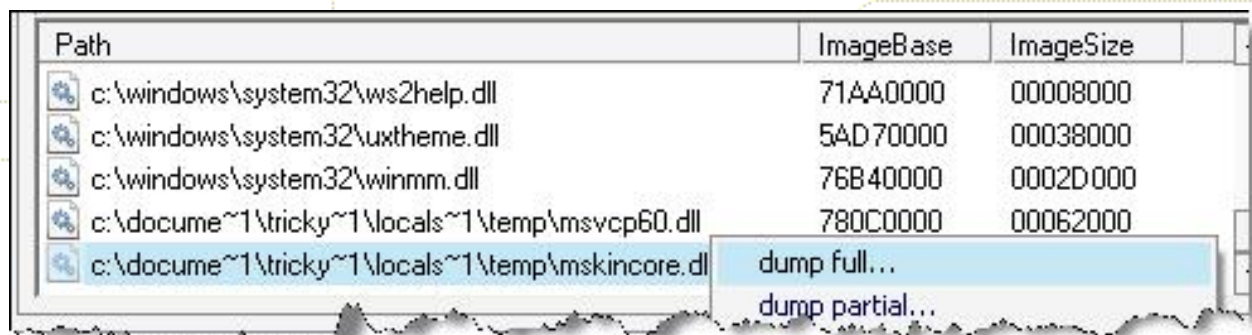
This is what the script unpack MoleBox 2.x or do. Now it continues to Shift-F9 to Break in VirtualProtect, dom wa dump window will see the IAT in turn is to move that do hẽ cancel any function is:

Address	Value	Comment
00F24000	7C809FA1	kernel32.InitializeCriticalSection
00F24004	7C91188A	ntdll.RtlDeleteCriticalSection
00F24008	7C9010ED	ntdll.RtlLeaveCriticalSection
00F2400C	7C801E16	kernel32.TerminateProcess
00F24010	7C80E00D	kernel32.GetCurrentProcess
00F24014	7C901005	ntdll.RtlEnterCriticalSection
00F24018	7C80AA66	kernel32.FreeLibrary
00F2401C	7C80AC28	kernel32.GetProcAddress
00F24020	7C81E85C	kernel32.DeleteFileA
00F24024	7C801077	kernel32.LoadLibraryA
00F24028	7C80A417	kernel32.QueryPerformanceCounter
00F2402C	7C8256DA	kernel32.QueryPerformanceFrequency
00F24030	7C80B357	kernel32.GetModuleFileNameA
00F24034	7C80B529	kernel32.GetModuleHandleA
00F24038	7C8397A1	kernel32.GetCurrentDirectoryA
00F2403C	7C823053	kernel32.SetCurrentDirectoryA
00F24040	7C81367C	kernel32.GetFullPathNameA
00F24044	7C80EFD7	kernel32.FindClose
00F24048	7C813559	kernel32.FindFirstFileA
00F2404C	7C80994E	kernel32.GetCurrentProcessId
00F24050	7C8221CF	kernel32.GetTempPathA
00F24054	7C9105D4	ntdll.RtlAllocateHeap
00F24058	7C80AA49	kernel32.GetProcessHeap
00F2405C	7C85E79B	kernel32.HeapValidate
00F24060	7C91043D	ntdll.RtlFreeHeap
00F24064	7C812851	kernel32.GetVersionExA
00F24068	7C812C8D	kernel32.GetCommandLineA
00F2406C	7C810386	kernel32.SetUnhandledExceptionFilter
00F24070	7C8099BD	kernel32.LocalAlloc
00F24074	7C802442	kernel32.Sleep
00F24078	7C80995D	kernel32.LocalFree
00F2407C	7C825F62	kernel32.FormatMessageA
00F24080	7C910331	ntdll.RtlGetLastWin32Error
00F24084	7C80B859	kernel32.VirtualQuery
00F24088	7C81EAE1	kernel32.RaiseException
00F2408C	7C812929	kernel32.HeapCreate
00F24090	7C811110	kernel32.HeapDestroy
00F24094	00000000	
00F24098	780C2D16	msvc60.std::_Winit::_Winit
00F2409C	780C1E94	msvc60.std::_Winit::_Winit
00F240A0	780C2D5E	msvc60.std::ios_base::Init::_Init
00F240A4	780C171C	msvc60.std::ios_base::Init::_Init
00F240A8	780C15A2	msvc60.std::lockit::Lockit

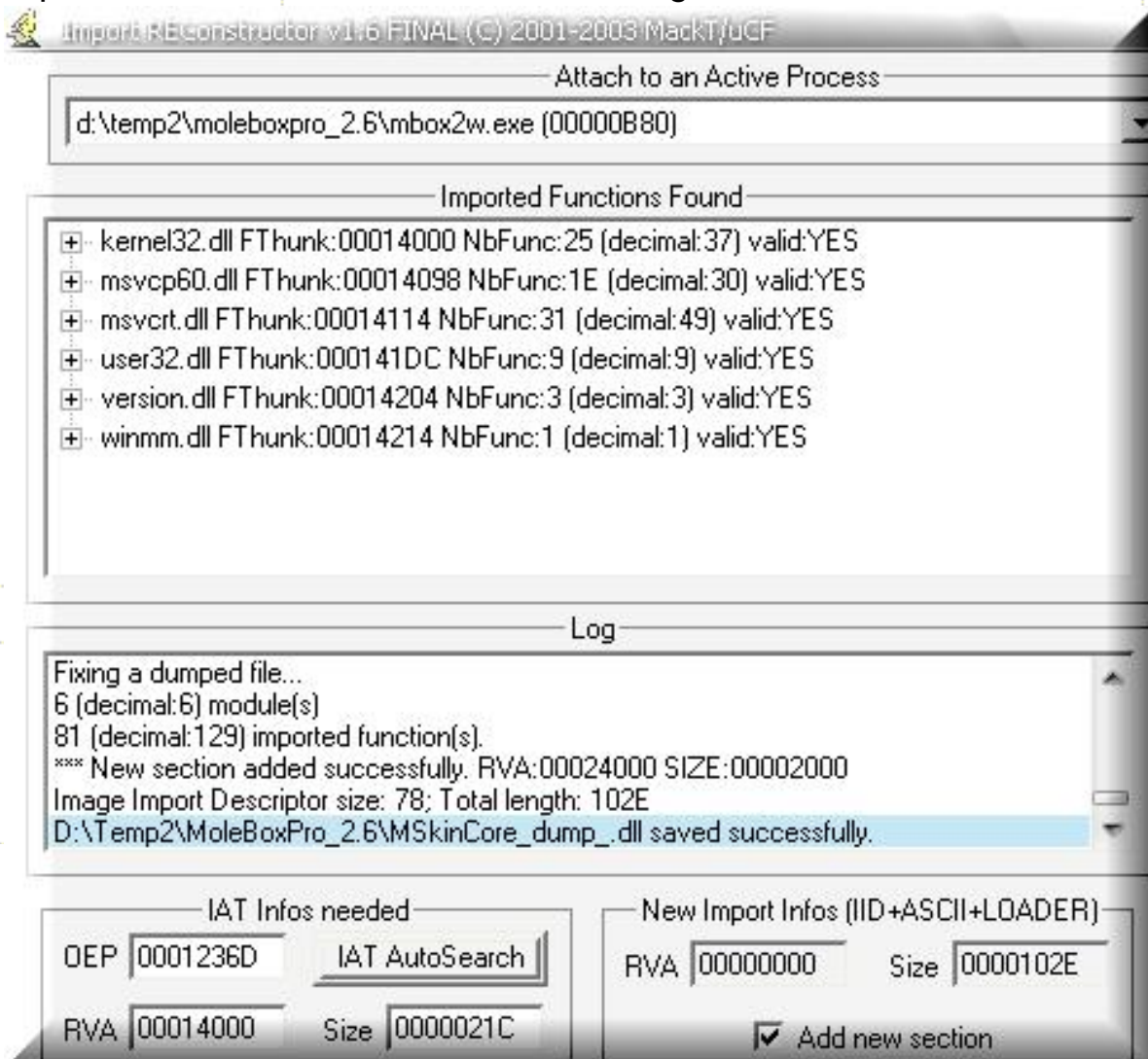
And then a break in the module by OEP MBOA we set in section. Previous text:

Address	Hex Dump	Disassembly
00F2236D	55	PUSH EBP
00F22370	00EC	MOV EBP, EBP
00F22370	53	PUSH EBX
00F22371	8B5D 08	MOV EBX, DWORD PTR SS:[EBP+8]
00F22374	56	PUSH ESI
00F22375	8B75 0C	MOV ESI, DWORD PTR SS:[EBP+C]
00F22378	57	PUSH EDI
00F22379	8B7D 10	MOV EDI, DWORD PTR SS:[EBP+10]
00F2237C	85F6	TEST ESI, ESI
00F2237E	75 09	JNZ SHORT MSkinCor.00F22389
00F22380	833D 14E9F20	CMP DWORD PTR DS:[F2E914], 0
00F22387	EB 26	JNZ SHORT MSkinCor.00F223AF
00F22389	83FE 01	CMP ESI, 1
00F2238C	74 05	JE SHORT MSkinCor.00F22393
00F2238E	83FE 02	CMP ESI, 2
00F22391	75 22	JNZ SHORT MSkinCor.00F223B5
00F22393	A1 1CE9F200	MOV EAX, DWORD PTR DS:[F2E91C]
00F22398	85C0	TEST EAX, EAX
00F2239A	74 09	JE SHORT MSkinCor.00F223A5
00F2239C	57	PUSH EDI
00F2239D	56	PUSH ESI

Delete BP VirtualProtect in farewell, the MBOA are in this section. Using LordPE to dump:



Next to the ImportREC due IAT has been canceled again do so of course the All Valid:



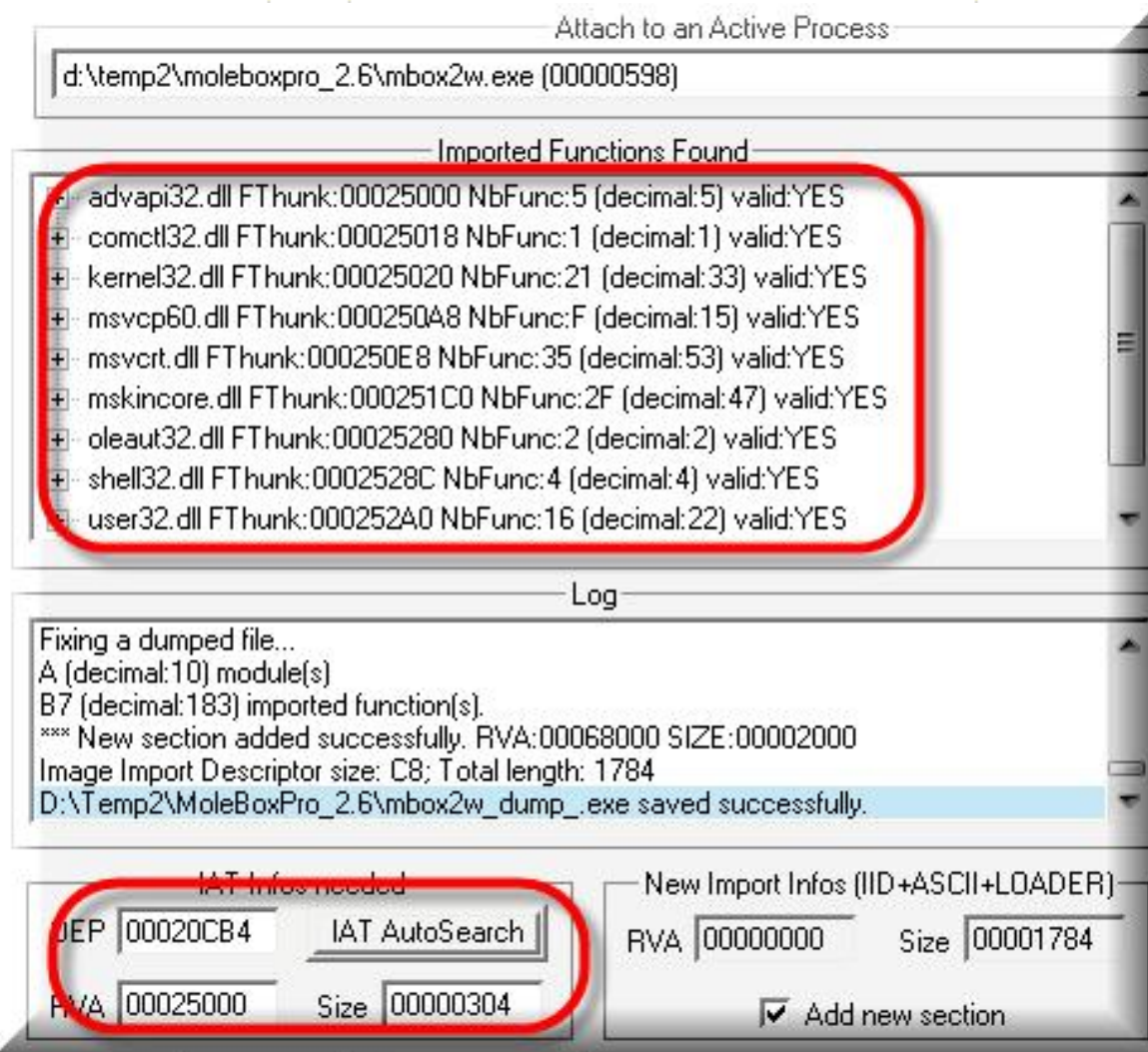
Similarly to the first module, we must conduct revised ImageBase and Relocation.

00004000	00002000				Priv	RW	RW
00E90000	00001000				Priv	RW	RW
00F10000	00001000	MSkinCor		PE header	Imag	R	RWE
00F11000	00013000	MSkinCor	.text	code	Imag	R	RWE
00F24000	00006000	MSkinCor	.rdata	data,export	Imag	R	RWE
00F2A000	00005000	MSkinCor	.data		Imag	R	RWE
00F2F000	00002000	MSkinCor	.rsrc	resources	Imag	R	RWE
00F31000	00002000	MSkinCor	.reloc		Imag	R	RWE
00F33000	00001000	MSkinCor	_BOX_	SFX,relocat	Imag	R	RWE
00000000	00001000	backsoy		PE header	Imag	R	RWE

So Imagebase = F10000 (in the other you can nhé)

Address	Hex dump	Disassembly
00420CB4	55	PUSH EBP
00420CB5	8BEC	MOV EBP,ESP
00420CB7	6A FF	PUSH -1
00420CB9	68 70544200	PUSH mbox2w.00425470
00420CBE	68 6A0E4200	PUSH mbox2w.00420E6A
00420CC3	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
00420CC9	50	PUSH EAX
00420CCA	64:8925 000000	MOV DWORD PTR FS:[0],ESP
00420CD1	83EC 68	SUB ESP,68
00420CD1	F2	REP

LordPE dump. Then Open ImportREC to fix IAT:



I do not see that there Invalid ko nhé concerns, it may also cancel the IAT in here, but also cancel the order with 1 as we fix the above. So obvious Valid All.

B - Fix dump file:

Now remains the same window when Olly OEP at last. Running fix dump file has the same folder with 2 module has extract out why:

MoleSkin Error Message



Program terminated by internal error
Please send file
D:\Temp2\MoleBoxPro_2.6\mbox2w_dump_-log.txt
to support@mail

Reason

```
SEH exception 0xc0000005
Code address 0x7c901010, access to address 0xd41ea4

CS :0x0000001B SS :0x00000023 DS :0x00000023
ES :0x00000023 FS :0x0000003B GS :0x00000000
EAX:0x00433710 EDX:0x00D41E90 ECX:0x7FFDF000
ESP:0x0012FDA0 EBP:0x0012FDE8 EIP:0x7C901010
ESI:0x0015233B EDI:0x00320032

0x0012fda0: 0x0043a3f8 0x00d41e90 0x00320032 0x0015233b
0x0012fdb0: 0x00000000 0x00000000 0x0012fe24 0x00150000
0x0012fdc0: 0x0012ff24 0x00f12736 0x0012fe10 0x00403f9a
0x0012fdd0: 0x0012fe24 0x0012f9c4 0x0012ff18 0x0043254c
```

MoleSkin framework (c) MoleStudio.com, 2002

Close

Send bugreport

This is everyone knows what type of error is correct, then do? Report on the 1 of the SEH in mole. If during unpack, we have done all the right steps when having this type of error, only minutes from Trace to find out where errors arise. Close notification and load the file to fix this dump 1 Olly other:

Address	Hex dump	Disassembly
00420CB4	55	PUSH EBP
00420CB5	8BEC	MOV EBP,ESP
00420CB7	6A FF	PUSH -1
00420CB9	68 70544200	PUSH mbox2w_d.00425470
00420CBE	68 6A0E4200	PUSH <JMP.&msvort._except_handler3>
00420CC3	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
00420CC9	50	PUSH EAX
00420CCA	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
00420CB1	8B7C	MOV ESI,EBP

Also at this time we still Olly 1 before being stopped at the OEP's original file. Temporarily called Olly is Olly (1) and Olly open the dump file fix Olly (2) nhé. How to find a trace errors in Olly (2), the position of having failed, the one back through Olly (1) trace to the same position to see the value in it like? From the comparison and find out how to overcome. How often does a trace Ctrl-F8 function to search for any CALL call to cause the error, press 1 to go yet, we will have to go to 1 loop:

```

00420088 803E 22 CMP BYTE PTR DS:[ESI],22
0042008E 75 3A JNZ SHORT mbox2w_d.004200CA
00420090 46 INC ESI
00420091 8975 8C MOV DWORD PTR SS:[EBP-74],ESI
00420094 8A06 MOV AL,BYTE PTR DS:[ESI]
00420096 3AC3 CMP AL,BL
00420098 74 04 JE SHORT mbox2w_d.0042009E
0042009A 3C 22 CMP AL,22
0042009C 75 F2 JNZ SHORT mbox2w_d.00420090
0042009E 803E 22 CMP BYTE PTR DS:[ESI],22
004200A1 75 04 JNZ SHORT mbox2w_d.004200A7
004200A3 46 INC ESI
004200A4 8975 8C MOV DWORD PTR SS:[EBP-74],ESI
004200A7 8A06 MOV AL,BYTE PTR DS:[ESI]
004200A9 3AC3 CMP AL,BL
004200AB 74 04 JE SHORT mbox2w_d.004200B1
004200AD 3C 20 CMP AL,20
004200AF 76 F2 JBE SHORT mbox2w_d.004200A3
004200B1 895D D0 MOV DWORD PTR SS:[EBP-30],EBX
004200B4 8D45 A4 LEA EAX,DWORD PTR SS:[EBP-5C]
004200B7 50 PUSH EAX
004200B8 FF15 98504200 CALL DWORD PTR DS:[<&kernel32.GetStartupInfoA>]
004200BE F645 D0 01 TEST BYTE PTR SS:[EBP-30],1

```

Immediately press F12 to Pause, find positions end loop, pulling down 1 billion, set in a BP:

```

004200B7 50 PUSH EAX
004200B8 FF15 98504200 CALL DWORD PTR DS:[<&kernel32.GetStartupInfoA>]
004200BE F645 D0 01 TEST BYTE PTR SS:[EBP-30],1
004200C2 74 11 JE SHORT mbox2w_d.004200D5
004200C4 0FB745 D4 MOVZX EAX,WORD PTR SS:[EBP-2C]
004200C8 EB 0E JEB SHORT mbox2w_d.004200D8
004200CA 803E 20 CMP BYTE PTR DS:[ESI],20
004200CD 76 D8 JBE SHORT mbox2w_d.004200A7
004200CF 46 INC ESI
004200D0 8975 8C MOV DWORD PTR SS:[EBP-74],ESI
004200D3 EB F5 JEB SHORT mbox2w_d.004200CA
004200D5 6A 0A PUSH 0A
004200D7 58 POP EAX
004200D8 50 PUSH EAX
004200D9 56 PUSH ESI
004200DA 53 PUSH EBX
004200DB 53 PUSH EBX
004200DC FF15 80504200 CALL DWORD PTR DS:[<&kernel32.GetModuleHandleA>]
004200DE 50 PUSH EAX
004200E3 E8 48AFFFFF CALL mbox2w_d.0041BC30

```

F9 to run and in the Break, BP put it. Continue Ctrl-F8 to find itself and Olly Break met by Exception:

Address	Hex dump	Disassembly
7C901010	837A 14 00	CMP DWORD PTR DS:[EDX+14],0
7C901014	75 4F	JNZ SHORT ntdll.7C901065
7C901016	F0:FF42 04	LOCK INC DWORD PTR DS:[EDX+4]
7C90101A	75 19	JNZ SHORT ntdll.7C901035
7C90101C	8B41 24	MOV EAX,DWORD PTR DS:[ECX+24]
7C90101F	8942 0C	MOV DWORD PTR DS:[EDX+C],EAX
7C901022	C742 08 010000	MOV DWORD PTR DS:[EDX+8],1
7C901029	33C0	XOR EAX,EAX
7C90102B	C2 0400	RET 4
7C90102E	8DA424 00000000	LEA ESP,DWORD PTR SS:[FSF]

Click "-" to step back before looking for new CALL function just go to:

Address	Hex dump	Disassembly
004200E3	E8 48AFFFFF	CALL mbox2w_d.0041BC30
004200E8	8945 98	MOV DWORD PTR SS:[EBP-68],EAX
004200EB	50	PUSH EAX
004200EC	FF15 74514200	CALL DWORD PTR DS:[<&msvcrt.exit>]
004200F2	8B45 EC	MOV EAX,DWORD PTR SS:[EBP-14]
004200F5	8B08	MOV ECX,DWORD PTR DS:[EAX]
004200F7	8B09	MOV ECX,DWORD PTR DS:[ECX]

So go on living here is an error, we enter into the first function, BP set 1. Ctrl-F2 restart again, F9 to Break BP has set in:

Address	Hex dump	Disassembly
0041BC30	55	PUSH EBP
0041BC31	8BEC	MOV EBP,ESP
0041BC33	6A FF	PUSH -1
0041BC35	68 C3344200	PUSH mbox2w_d.004234C3
0041BC3A	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
0041BC40	50	PUSH EAX
0041BC41	64:8925 000000	MOV DWORD PTR FS:[0],ESP
0041BC48	81EC DC000000	SUB ESP,0DC
0041BC4E	53	PUSH EBX
0041BC4F	56	PUSH ESI
0041BC50	57	PUSH EDI

Delete BP go, do the same, find the function CALL cause errors, BP set it right, Ctrl-F2, F9 and Bread:

0041BC59	6A 2E	PUSH 2E
0041BC5B	68 70CF4200	PUSH mbox2w_d.0042CF70
0041BC60	FF15 28514200	CALL DWORD PTR DS:[<&msvcrt.strchr>]
0041BC66	C600 00	MOV BYTE PTR DS:[EAX],0
0041BC69	68 00006200	PUSH 620000
0041BC6E	E8 FB7E0100	CALL mbox2w_d.00433B6E
0041BC73	A6	CMPS BYTE PTR DS:[ESI],BYTE PTR ES:[EDI]
0041BC74	D338	SAR DWORD PTR DS:[EAX],CL
0041BC76	50	POP EBP
0041BC77	6A 5F	PUSH 5F
0041BC79	8479 71	TEST BYTE PTR DS:[ECX+71],BH
0041BC7C	D5 B9	ADD 0B9
0041BC7E	55 90 90 000000	CALL 55909000

This refers to 1 CALL address is 433B6E, this address is located in the section:

003B0000	00049000					
00400000	00001000	mbox2w_d	PE header	Imag	R	RWE
00401000	00023000	mbox2w_d		Imag	R	RWE
00424000	00001000	mbox2w_d		Imag	R	RWE
00425000	00005000	mbox2w_d	data	Imag	R	RWE
0042A000	00004000	mbox2w_d		Imag	R	RWE
0042E000	00001000	mbox2w_d		Imag	R	RWE
0042F000	00003000	mbox2w_d	resources	Imag	R	RWE
00432000	00012000	mbox2w_d	code	Imag	R	RWE
00444000	00001000	mbox2w_d		Imag	R	RWE
00445000	00008000	mbox2w_d		Imag	R	RWE
0044D000	0001A000	mbox2w_d	.data	Imag	R	RWE
00467000	00001000	mbox2w_d	relocations	Imag	R	RWE
00468000	00002000	mbox2w_d	imports	Imag	R	RWE
00470000	00009000	mbox2w_d		Map	R	RWE

This is the old section of the Mole, so is the dump file, we still rely on Mole, unpack News is not complete. Continue to do the same as above, and find out exactly where that error here:

0043A30E	83EC 1C	SUB ESP,1C	
0043A3E1	53	PUSH EBX	
0043A3E2	56	PUSH ESI	
0043A3E3	57	PUSH EDI	
0043A3E4	8955 CC	MOV DWORD PTR SS:[EBP-34],EDX	ntdll.7C910738
0043A3E7	624D 00	MOV DWORD PTR SS:[EBP-30],EAX	
0043A3EA	B8 10374300	MOV EAX,mbox2w_d.00433710	
0043A3EF	FF70 04	PUSH DWORD PTR DS:[EAX+4]	
0043A3F2	FF15 94764400	CALL DWORD PTR DS:[447694]	ntdll.RtlEnterCriticalSection
0043A3F5	8365 FC 00	AND DWORD PTR SS:[EBP-4],0	
0043A3FC	8B45 D0	MOV EAX,DWORD PTR SS:[EBP-30]	mbox2w_d.0043A4A
0043A3FF	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0043A401	8945 DC	MOV DWORD PTR SS:[EBP-24],EAX	mbox2w_d.0042CF7
0043A404	8945 DC	MOV DWORD PTR SS:[EBP-24],EAX	mbox2w_d.0042CF7

BP set immediately ordered PUSH DWORD PTR DS: [EAX +4]. Ctrl-F2, F9 and Bread:

Address	Hex dump	Disassembly
0043A3EF	FF70 04	PUSH DWORD PTR DS:[EAX+4]
0043A3F2	FF15 94764400	CALL DWORD PTR DS:[447694]
0043A3F8	8365 FC 00	AND DWORD PTR SS:[EBP-4],0
0043A3FC	8B45 D0	MOV EAX,DWORD PTR SS:[EBP-30]
0043A3FF	8B00	MOV EAX,DWORD PTR DS:[EAX]
0043A401	8945 DC	MOV DWORD PTR SS:[EBP-24],EAX

Dom down in value [EAX +4] at this time:

DS:[00433714]=00D41E90

Khuc khúc, why go to the address D41E90 funds do exist in memory at this time:

003B0000	00049000					
00400000	00001000	mbox2w_d		PE header	Map	RW
00401000	00023000	mbox2w_d			Imag	R
00424000	00001000	mbox2w_d			Imag	R
00425000	00005000	mbox2w_d		data	Imag	R
0042A000	00004000	mbox2w_d			Imag	R
0042E000	00001000	mbox2w_d			Imag	R
0042F000	00003000	mbox2w_d		resources	Imag	R
00432000	00012000	mbox2w_d		code	Imag	R
00444000	00001000	mbox2w_d			Imag	R
00445000	00008000	mbox2w_d			Imag	R
0044D000	0001A000	mbox2w_d	.data	relocations	Imag	R
00467000	00001000	.adata			Imag	R
00468000	00002000	mbox2w_d	.mact	imports	Imag	R
00470000	00009000				Map	R
00530000	00002000				Map	R
00540000	00103000				Map	R
00550000	00116000				Map	R
00550000	00002000				Map	R
00560000	00002000				Map	R
00570000	00001000				Priv	RW
005F0000	00001000				Priv	RWE
00F00000	00001000	msk incor		PE header	Imag	R
00F11000	00013000	msk incor	.text	code	Imag	R
00F24000	00006000	msk incor	.rdata	data, export	Imag	R
00F30000	00005000	msk incor	.data	data	Imag	R

Clearly ko ko exist right? That is why there notified of this exception:

Access violation when reading [00D41EA4] - use Shift+F7/F8/F9 to pass exception to program

Keep the school, that they Olly (1) is open ko, turn it over. Goto to the same address in the command PUSH 43A3EF, BP set 1, and F9 Break:

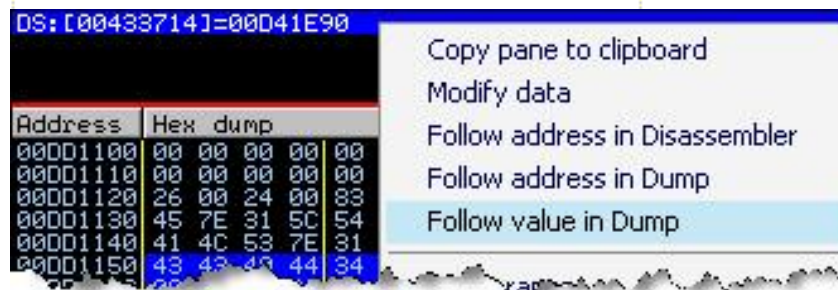
Address	Hex dump	Disassembly
0043A3EF	. FF70 04	PUSH DWORD PTR DS:[EAX+4]
0043A3F2	. FF15 9476440	CALL DWORD PTR DS:[447694]
0043A3F8	. 8365 FC 00	AND DWORD PTR SS:[EBP-4], 0
0043A3FC	. 8B45 D0	MOV EAX, DWORD PTR SS:[EBP-30]
0043A3FF	. 8B00	MOV EAX, DWORD PTR DS:[EAX]
0043A401	. 8B45 D0	MOV EAX, DWORD PTR SS:[EBP-30]

DS:[00433714]=00D41E90

Chang Y, others do anything (remember that address D41E90 this on your other nhé, it is the area section do have a fixed offset). But in Olly (1), the region address these shortcomings:

00CF0000	00001000				Priv	RWE	RWE
00D00000	00001000				Priv	RWE	RWE
00D10000	00001000				Priv	RWE	RWE
00D20000	00001000				Priv	RWE	RWE
00D30000	00001000				Priv	RWE	RWE
00D40000	00003000				Priv	RW	RW
00D50000	00001000				Priv	RW	RW
00D70000	00001000				Priv	RW	RW
00D80000	00042000				Priv	RW	RW
00DD1000	00001000				Priv	RW	RW
00DD4000	00002000				Priv	RW	RW
00E80000	00002000				Map	R	R
00E90000	00001000				Priv	RW	RW
00F10000	00001000	MSK incor		PE header	Imag	R	RWE
00F11000	00001000	MSK incor	.text	code	Imag	R	RWE

This is due Mole Packer created during unpack code. But in the dump file fix will of course do not have. Frequently this case, or we dump the whole section to find the Olly (1) and then add the file to fix dump. But after checking the trick i need to see how. Simply edit the value for the match is. Back of the book's code Olly (1), Follow dump value in value in D41E90 see value in that What:



Ah há:

Address	Hex dump	ASCII
00041E90	A0 85 15 00 FF FF FF FF 00 00 00 00 00 00 00 00	...
00041EA0	00 00 00 00 00 00 00 00 C8 85 15 00 FF FF FF FF	...
00041EB0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	...
00041EC0	E0 85 15 00 FF FF FF FF 00 00 00 00 00 00 00 00	...

Remember to get DWORD 2 here: **FF FF FF FF**. Quay through Olly (2), Follow in dump address:



We found:

Address	Hex dump
00433714	90 1E 04 00 27 48 45 52 45 49 53 42
00433724	4F 44 45 27 00 00 00 00 00 00 00 00
00433734	2E 74 65 78 74 00 00 00 5F 2A 02 00

So DWORD **00D41E90** saved at 433,714. Ko To access must address do actually have this, we modified it to address in the file, then paste DWORD 2 found in Olly (1) to the Okie . Hour 1 place to find enough space contains DWORD 1, section often empty or are used in these cases was ImportREC add to fix the IAT:

Address	Offset	Symbol	Section	Map	RW	RW
003B0000	00049000		PE header	Imag	R	RWE
00400000	00001000	mbox2w_d		Imag	R	RWE
00401000	00023000	mbox2w_d		Imag	R	RWE
00424000	00001000	mbox2w_d		Imag	R	RWE
00425000	00005000	mbox2w_d	data	Imag	R	RWE
0042A000	00004000	mbox2w_d		Imag	R	RWE
0042E000	00001000	mbox2w_d		Imag	R	RWE
0042F000	00003000	mbox2w_d	resources	Imag	R	RWE
00432000	00012000	mbox2w_d	code	Imag	R	RWE
00444000	00001000	mbox2w_d		Imag	R	RWE
00445000	00008000	mbox2w_d		Imag	R	RWE
00446000	00010000	mbox2w_d	data	Imag	R	RWE
00467000	00001000	mbox2w_d	.adata	Imag	R	RWE
00468000	00002000	mbox2w_d	.nackt	Imag	R	RWE
00469000	00000000			Map	R E	R E
00530000	00002000			Map	R E	R E

Size of memory in this section is 2000, and it just use:


```

30469710 49 74 65 60 54 65 78 74 41 00 57 02 53 65 74 46 ItemTextA.W@SetF
30469720 6F 63 75 73 00 00 93 02 53 68 6F 77 57 69 6E 64 ocus...@ShowWind
30469730 6F 77 00 00 9F 00 44 69 61 6C 6F 67 42 6F 78 50 ow...f.DialogBoxF
30469740 61 72 61 60 41 00 3C 02 53 65 6E 64 40 65 73 73 aramA.<@SendMesse
30469750 61 67 65 41 00 00 BC 01 4C 6F 61 64 49 63 6F 6E ageA..@LoadIcon
30469760 41 00 63 6F 60 64 6C 67 33 32 2E 64 6C 6C 00 00 A.comdlg32.dll..
30469770 6E 00 47 65 74 4F 70 65 6E 46 69 6C 65 4E 61 6D n.GetOpenFileNam
30469780 65 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 eA.....
30469790 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
304697A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
304697B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
304697C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
304697D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

469788 - 468000 = 1788 bytes. The balance is still a lot Byte for us. But to consider the Byte is empty there is real or just a virtual:

Name	VOffset	VSize	ROffset	RSize	Flags
	0002F000	00003000	0002F000	00003000	E0000040
	00032000	00012000	00032000	00012000	E0000040
	00044000	00001000	00044000	00001000	E0000040
	00045000	00008000	00045000	00008000	E0000040
.data	0004D000	0001A000	0004D000	0001A000	E0000040
.adata	00067000	00001000	00067000	00001000	E0000040
.mackt	00068000	00002000	00068000	00002000	E0000060

Recall that the Windows memory management areas by 1000h bytes. Size should sometimes in memory rounded up to the actual size than the baby. And if you Save the file with Olly in the memory do exist in the actual file will be error: "Unable locate memory" now. There really Size: Raw Offset is 2000 we should secure the Byte memory space in the other may use and be saved.

Back window memory dump from the map:

```

00469740 61 72 61 60 41 00 3C 02 53 65 6E 64 40 65 73 73 aramH.<@Sendmess
00469750 61 67 65 41 00 00 BC 01 4C 6F 61 64 49 63 6F 6E ageA..@LoadIcon
00469760 41 00 63 6F 60 64 6C 67 33 32 2E 64 6C 6C 00 00 A.comdlg32.dll..
00469770 6E 00 47 65 74 4F 70 65 6E 46 69 6C 65 4E 61 6D n.GetOpenFileNam
00469780 65 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 eA.....
00469790 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
004697A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
004697B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
004697C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
004697D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

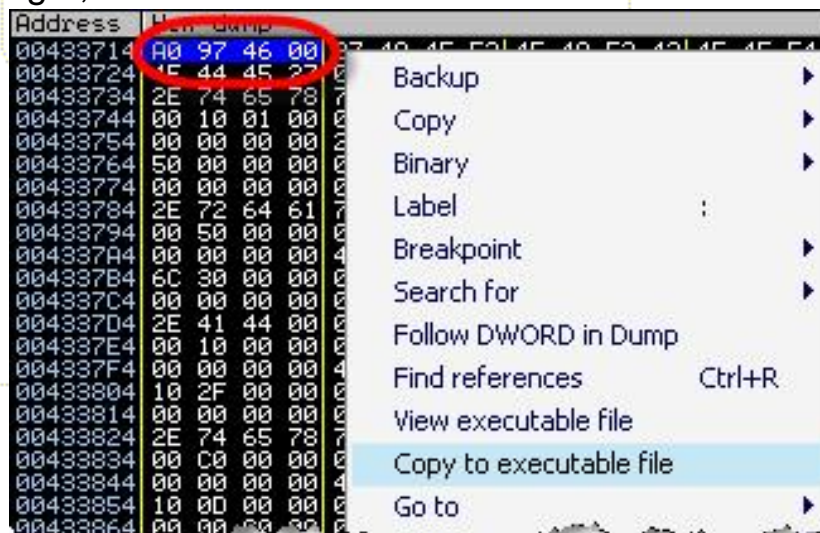
You can use any position is available, the trick is to select 4697A0. Back window Code Olly (2), fix the value D41E90:





à

To DWORD area just right, or Select:

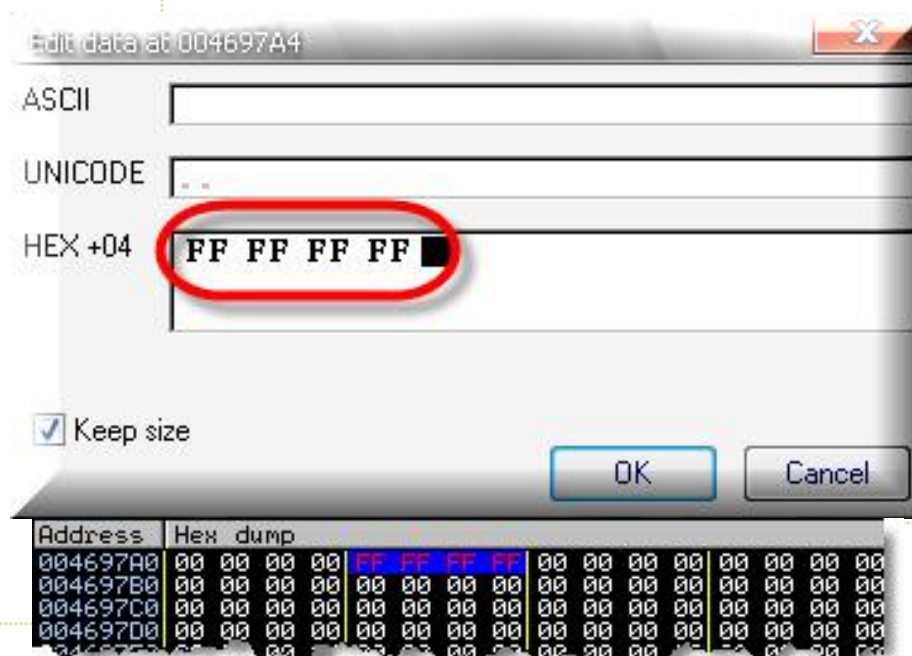


Save overwrite files with the same name fix dump. Remember task of editing we include 2 parts, to address regional D41E90 into 1 address in the file, and edit content at the address for this match DWORD we found in Olly (1) . But only Olly Save the amendment in section 1 so we have to turn 2.

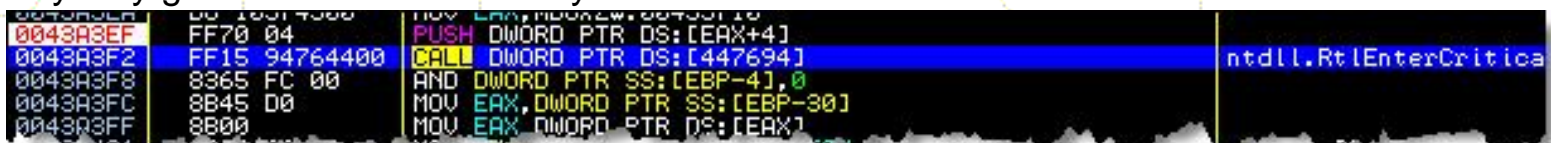
The use Olly (2) open the dump file fix has been saved for, the window dump, Goto to address both fixed 4697A0:



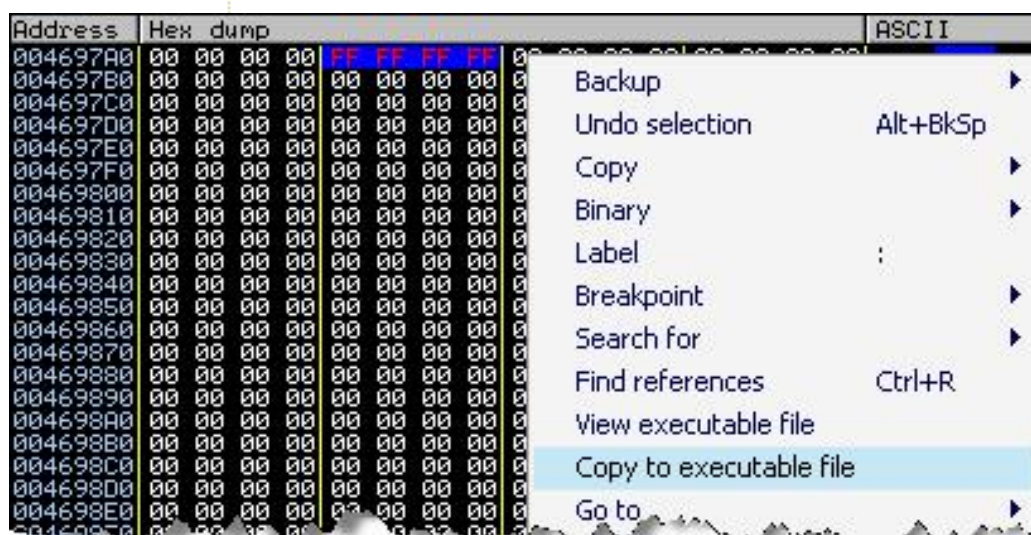
DWORD need to be fixed is the value found in Olly (1) but we must move up from 1 DWORD 4697A0, then the Edit:



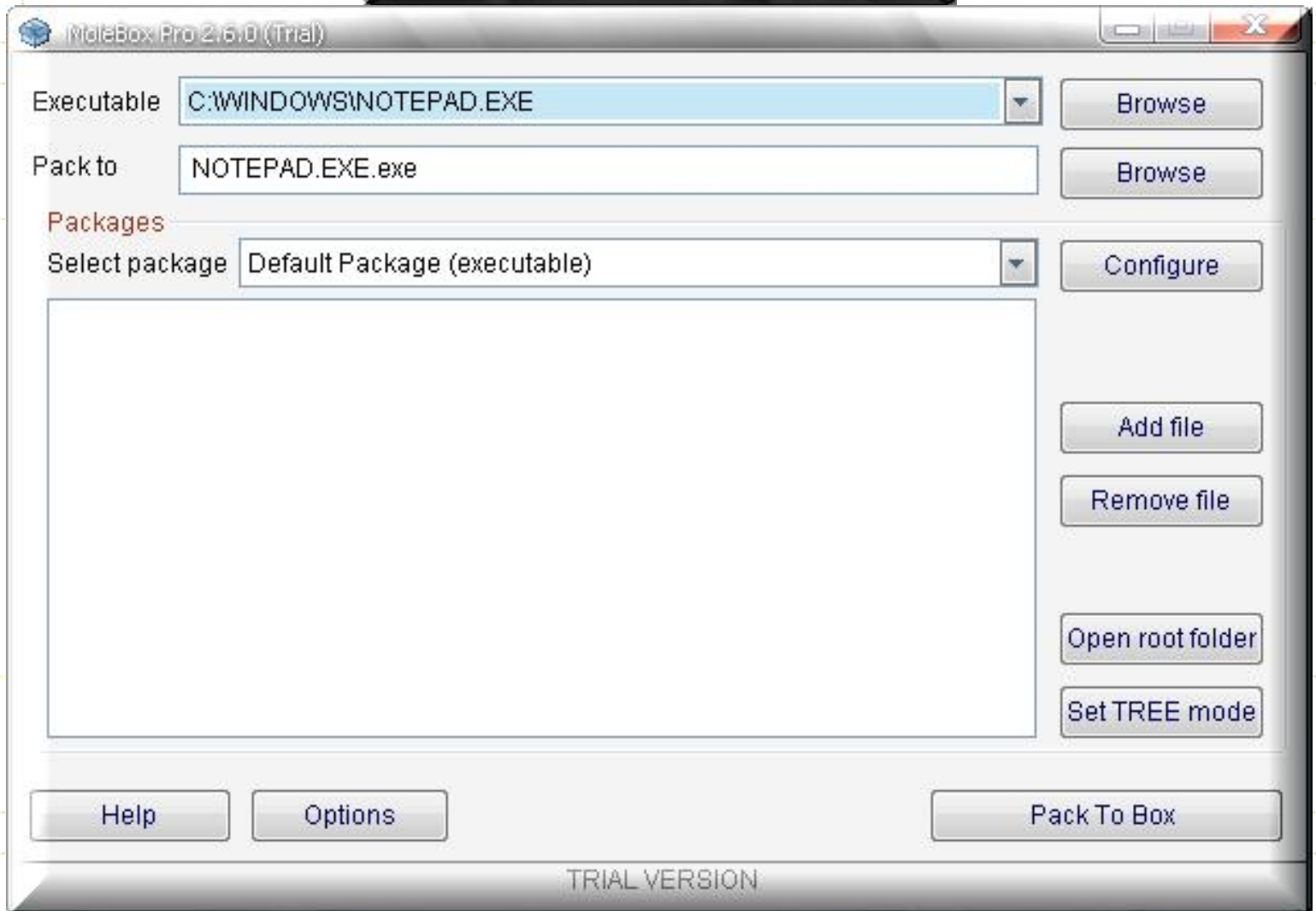
Why only get DWORD 2 found? If you're hard to trace this function:



It will only access to the value in D41E90 News + 4 + 4 is 4697A0 now. So save the file is then this:

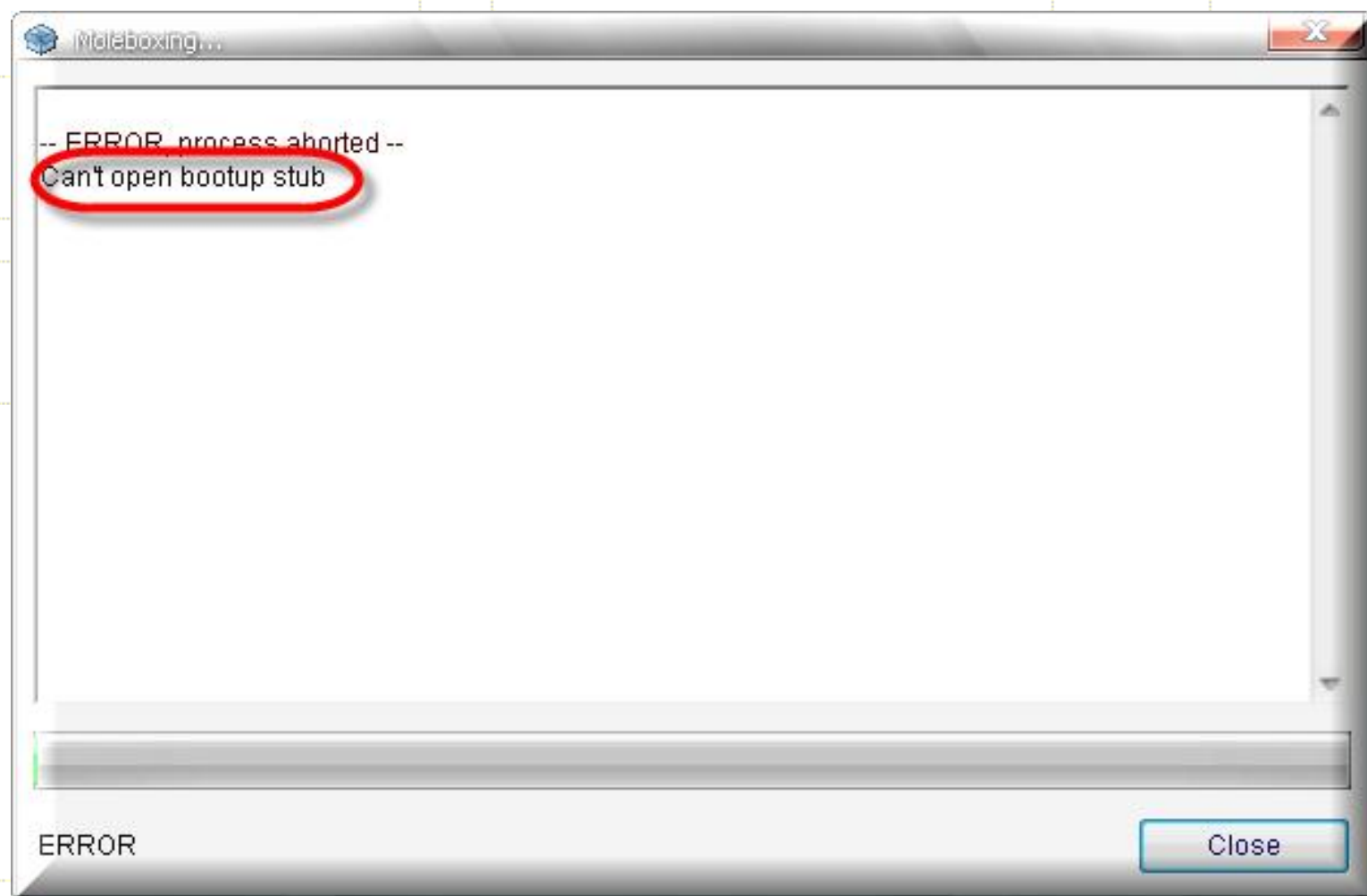


The best is saved for the file name always fix dump. Olly (1) has not Close nhé. Now run fix dump file to see how (the same folder other small module 2), rub rub:



C - Find & dump 2 stub Hidden Files:

File "RUN NGON" do right? That way, let drilling from "NGON" from the early mòm, select 1 files PE get any, as in the picture is NOTEPAD.exe, then press the button "Pack To Box":



So from "NGON" always. Remember the list that we see at first:

- mrcp60.dll
- MSkinCore.dll
- mbox2_bootupldemo
- mbox2_bootupdbgldemo
- mbox2_blacklist.txt

According to the list and the error message I guess it is now the 2 surviving children: [mbox2_bootupldemo](#) and [mbox2_bootupdbgldemo](#). [Mbox2_blacklist.txt](#) the file but i do need to find. If you ever run the 1 Retail 2.5x will meet demands NAG 1 License . As the black list file can only list as Key to compare with the imported variety Key. Therefore, the file can do this in 1 of Trial ko hê requires the key. Now we find the other 2 files bootup nhê. Cu Lao Hec Lec use 1 tip for that is quite trick or using BP CreateFileA associated with BP GetSystemTimeAsFileTime to the return of this function will immediately create files on bootup. Now back Olly (1) is stopped at PUSH commands me ...:

Address	Hex dump	Disassembly
0043A3EF	FF70 04	PUSH DWORD PTR DS:[EAX+4]
0043A3F2	FF15 9476440	CALL DWORD PTR DS:[447694]
0043A3F8	8365 FC 00	AND DWORD PTR SS:[EBP-4],0
0043A3FC	8B45 D0	MOV EAX,DWORD PTR SS:[EBP-30]
0043A3FF	8B00	MOV EAX,DWORD PTR DS:[EAX]
0043A401	8945 7C	MOV DWORD PTR SS:[EBP-24],EAX

Un BP go right here. Click to run for the target to complete the program. Before the Pack File, back Olly (1), BP set CreateFileA:

Command : bp CreateFileA

Now click the new "Pack To Box" and Bread

```

0012F594 00436354 mbox2w.00436354
0012F598 0015FDEC ASCII "C:\WINDOWS\notepad.exe.mboxfg"
0012F59C 0012F5B8
0012F5A0 0012F5B0
0012F5A4 00000000
0012F5A8 00150000
0012F5AC 00000000
0012F5B0 00150000
0012F5B4 000006F1
0012F5B8 00440A56 CALL to CreateFileA from mbox2w.00440A56
0012F5BC 0015FDEC FileName = "C:\WINDOWS\notepad.exe.mboxfg"
0012F5C0 C0000000 Access = GENERIC_READ|GENERIC_WRITE
0012F5C4 00000003 ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
0012F5C8 0012F60C pSecurity = 0012F60C
0012F5CC 00000002 Mode = CREATE_ALWAYS
0012F5D0 00000000 Attributes = NORMAL
0012F5D4 00000000 hTemplateFile = NULL
0012F5D8 FFFFFFFF
0012F5DC 0012F5B8

```

Not need to find the file, continue to press Shift-F9 until you see:

```

0136F4C0 00000005
0136F4C4 00D42224 ASCII "MBOX2_BOOTUPLTDEMO"
0136F4C8 0000002A
0136F4CC 00D42224 ASCII "MBOX2_BOOTUPLTDEMO"
0136F4D0 0136FE04
0136F4D4 0136FE04
0136F4D8 0136FED8
0136F4DC 0043254C mbox2w.0043254C
0136F4E0 0043638E CALL to CreateFileA from mbox2w.0043638E
0136F4E4 00D42438 FileName = "D:\TEMP2\MOLEBOXPRO_2.6\MBOX2W.EXE"
0136F4E8 80000000 Access = GENERIC_READ
0136F4EC 00000001 ShareMode = FILE_SHARE_READ
0136F4F0 00000000 pSecurity = NULL
0136F4F4 00000003 Mode = OPEN_EXISTING
0136F4F8 00000000 Attributes = 0
0136F4FC 00000000 hTemplateFile = NULL
0136F500 00000000
0136F504 00000000

```

Fairness, it is her child. CreateFileA Put in to BP. GetSystemTimeAsFileTime bp Set:

Command : bp GetSystemTimeAsFileTime

Shift-F9 to 1, Bread, and press 1 and, more Bread also function at the top:

Address	Hex dump	Disassembly
7C8017E5	8BFF	MOV EDI,EDI
7C8017E7	55	PUSH EBP
7C8017E8	8BEC	MOV EBP,ESP
7C8017EA	A1 1800FE7F	MOV EAX,DWORD PTR DS:[7FFE0018]
7C8017EF	8B15 1400FE7F	MOV EDX,DWORD PTR DS:[7FFE0014]
7C8017F5	3B05 1C00FE7F	CMP EAX,DWORD PTR DS:[7FFE001C]
7C8017F8	75 ED	JNZ SHORT kernel!32.7C8017EA
7C8017FD	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
7C801800	8911	MOV DWORD PTR DS:[ECX],EDX
7C801802	8941 04	MOV DWORD PTR DS:[ECX+4],EAX
7C801805	5D	POP EBP
7C801806	C2 0400	RETN 4
7C801809	90	NOP

Do not take BP. Ctrl-F9 to trace RETN 4 and Return from this function, we come:

Address	Hex dump	Disassembly
00436FE7	8B45 C4	MOV EAX,DWORD PTR SS:[EBP-3C]
00436FEA	8B4D F0	MOV ECX,DWORD PTR SS:[EBP-10]
00436FED	64:890D 000000	MOV DWORD PTR FS:[0],ECX
00436FF4	5F	POP EDI
00436FF5	5E	POP ESI
00436FF6	5B	POP EBX
00436FF7	C9	LEAVE
00436FF8	C2 0800	RETN 8

Follow dump in value here:



Remember that address D90090 on to another trick on you nhé.

Address	Hex dump	ASCII
00D90090	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZÉ.♦...♦... ..
00D900A0	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	7.....@.....
00D900B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00D900C0	00 00 00 00 00 00 00 00 00 00 00 00 08 00 00 00T...
00D900D0	0E 1F BA 0E 00 84 09 CD 21 B8 01 4C CD 21 54 68	8V 8.-.=†q@L=†Th
00D900E0	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program cannot
00D900F0	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	be run in DOS
00D90100	6D 6F 64 65 2E 0D 0A 0A 00 00 00 00 00 00 00 00	mode: \$....

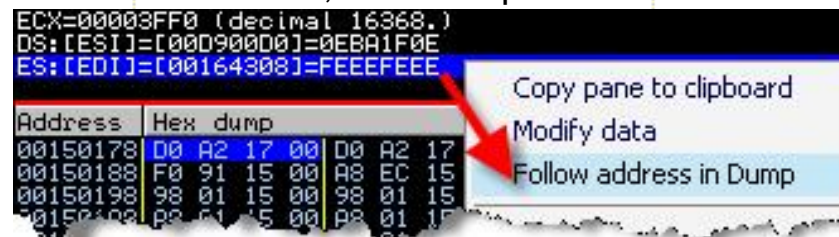
Hum hum, no matter how this same PE header so wá play. Vậy bootup file by this reasoning it has header header as of 1 PE file. But this is only 1 small section header only, not to save the memory area File mbox2_bootupltdemo. One replied that it will decrypt the Byte full file 1 through this region in mind other. Now we set 1 BP access to memory on Byte first



Press Shift-F9 to Break right to access it on Byte during bootup decrypt the file:

Address	Hex dump	Disassembly
00437080	F3:A5	REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
00437082	8BC8	MOV ECX,EAX
00437084	83E1 03	AND ECX,3
00437087	F3:A4	REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00437089	B8 00000100	MOV EAX,10000
0043708E	2B45 F8	SUB EAX,DWORD PTR SS:[EBP-8]
00437091	3945 10	CMP DWORD PTR SS:[EBP+10],EAX
00437094	73 08	JNB SHORT mbox2w.0043709E
00437096	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]
00437099	8945 E4	MOV DWORD PTR SS:[EBP-1C],EAX
0043709C	EB 08	JNB SHORT mbox2w.004370A9
0043709E	B8 00000100	MOV EAX,10000
004370A3	2B45 F8	SUB EAX,DWORD PTR SS:[EBP-8]
004370A6	8945 E4	MOV DWORD PTR SS:[EBP-1C],EAX

Implementing a Trace F7 in order on me, then dump in Follow address here:



Not necessary but must also consider the address on your computer at the other it will nhé, but this is the Byte dump in memory:

Address	Hex dump	ASCII
00164308	EE FE EE FE EE FE EE FE EE FE EE FE EE FE	EEEEEEEEEEEEEEEE
00164318	EE FE EE FE EE FE EE FE EE FE EE FE EE FE	EEEEEEEEEEEEEEEE
00164328	EE FE EE FE EE FE EE FE EE FE EE FE EE FE	EEEEEEEEEEEEEEEE
00164338	EE FE EE FE EE FE EE FE EE FE EE FE EE FE	EEEEEEEEEEEEEEEE
00164348	EE FE EE FE EE FE EE FE EE FE EE FE EE FE	EEEEEEEEEEEEEEEE
00164358	EE FE EE FE EE FE EE FE EE FE EE FE EE FE	EEEEEEEEEEEEEEEE
00164368	EE FE EE FE EE FE EE FE EE FE EE FE EE FE	EEEEEEEEEEEEEEEE
00164378	EE FE EE FE EE FE EE FE EE FE EE FE EE FE	EEEEEEEEEEEEEEEE

The implementation of 1 Trace F7 to see it:

Address	Hex dump	ASCII
001642C8	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZÉ.♦...♦... ..
001642D8	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	7.....@.....
001642E8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001642F8	00 00 00 00 00 00 00 00 00 00 00 00 08 00 00 007...
00164308	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	8V 8.+. =?90L=?Th
00164318	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program cannot
00164328	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	be run in DOS
00164338	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....\$.
00164348	EE FE EE FE EE FE EE FE EE FE EE FE EE FE	EEEEEEEEEEEEEEEE
00164358	EE FE EE FE EE FE EE FE EE FE EE FE EE FE	EEEEEEEEEEEEEEEE

Yes it is the move from Byte D90090 to 1642C8, if any pressing F7 will see more clearly the process. But we do take the time useless, Trace F8 1 for slavery:

Address	Hex dump	Disassembly
00437080	F3:A5	REP MOVSD DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
00437082	8BC8	MOV ECX,EAX
00437084	83E1 03	AND ECX,3
00437087	F3:A4	REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00437089	B8 00000100	MOV EAX,10000
0043708E	2B45 F8	SUB EAX,DWORD PTR SS:[EBP-8]

The Byte has to move a lot:

Address	Hex dump	ASCII
001642C8	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZÉ.♦...♦... ..
001642D8	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	7.....@.....
001642E8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001642F8	00 00 00 00 00 00 00 00 00 00 00 00 08 00 00 007...
00164308	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	8V 8.+. =?90L=?Th
00164318	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program cannot
00164328	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	be run in DOS
00164338	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....\$.
00164348	8B AA FA 42 CF CB 94 11 CF CB 94 11 CF CB 94 11	in BTR0<TR0<TR0<
00164358	4C 07 9A 11 C5 CB 94 11 DC C3 C9 11 CD CB 94 11	LH0<TR0<TR0<TR0<
00164368	4C C3 C9 11 CC CB 94 11 CF CB 95 11 07 CB 94 11	LH0<TR0<TR0<TR0<
00164378	F9 ED 9E 11 DE CB 94 11 F9 ED 9F 11 EF CB 94 11	0A<TR0<0f<0TR0<
00164388	52 69 63 68 CF CB 94 11 00 00 00 00 00 00 00 00	RichTR0<
00164398	00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00PE..L0♦
001643A8	B2 FC 01 45 00 00 00 00 00 00 00 00 E0 00 0E 21	880E.....α.8?
001643B8	0B 01 06 00 00 14 01 00 00 92 00 00 00 00 00 00	80+. 900. E.....
001643C8	10 27 00 00 00 10 00 00 00 30 01 00 00 00 00 10	7.....00.....
001643D8	00 10 00 00 00 02 00 00 04 00 00 00 00 00 00 00	7.....♦.....
001643E8	04 00 00 00 00 00 00 00 00 E0 01 00 00 04 00 00	♦.....α0.♦.....
001643F8	00 00 00 00 02 00 00 00 00 00 10 00 00 10 00 000.....
00164408	00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 007.....
00164418	00 00 00 00 00 00 00 00 CC 3A 01 00 3C 00 00 00 :0.<.....
00164428	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00164438	00 00 00 00 00 00 00 00 00 C0 01 00 88 0F 00 000.é*.....
00164448	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00164458	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00164468	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00164478	00 30 01 00 60 00 00 00 00 00 00 00 00 00 00 00	..00.....
00164488	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00164498	2E 74 65 78 74 00 00 00 FF 12 01 00 00 10 00 00	..text... 70...7
001644A8	00 14 01 00 00 04 00 00 00 00 00 00 00 00 00 00	70.....♦.....
001644B8	00 00 00 00 20 00 00 50 2E 72 64 61 74 61 00 00ed00.....

But how to know the exact length Byte has to move, looking to see is right here:

EAX=00010000, (UNICODE ":::::\")
ECX=00000000

In the address ends of the move is $1642C8 + 10000 = 1742C8$, I Go to it to the dump:

Address	Hex dump	ASCII
00174268	04 8B 45 F0 8B C8 2B 0E 89 06 01 4E 08 89 53 34	♦iE=i+8e+0Nes4
00174278	FF 75 10 E9 B2 01 00 00 8B 45 08 8B CF 89 43 20	u>000..iEi=eC
00174288	8B 45 FC 89 43 1C 8B 45 0C 2B 08 83 60 04 00 89	iEeCLiE.+>♦.e
00174298	38 01 48 08 8B 4D F4 FF 75 10 89 4B 34 E9 DF 01	80HIMr u>EK40H
001742A8	00 00 8B 45 08 6A FC 89 43 20 8B 45 FC 89 43 1C	..iEjneC iEeCL
001742B8	8B 45 F8 89 46 04 8B C7 2B 06 89 3E 01 46 08 8B	iEeF+i+e>0F0i
001742C8	EE FE EE FE EE FE EE FE EE FE EE FE EE FE FE EE FE	eeeeeeeeeeeeeeee
001742D8	EE FE EE FE EE FE EE FE EE FE EE FE EE FE FE EE FE	eeeeeeeeeeeeeeee
001742E8	EE FE EE FE EE FE EE FE EE FE EE FE EE FE FE EE FE	eeeeeeeeeeeeeeee
001742F8	EE FE EE FE EE FE EE FE EE FE EE FE EE FE FE EE FE	eeeeeeeeeeeeeeee
00174308	EE FE EE FE EE FE EE FE EE FE EE FE EE FE FE EE FE	eeeeeeeeeeeeeeee
00174318	EE FE EE FE EE FE EE FE EE FE EE FE EE FE FE EE FE	eeeeeeeeeeeeeeee
00174328	EE FE EE FE EE FE EE FE EE FE EE FE EE FE FE EE FE	eeeeeeeeeeeeeeee
00174338	EE FE EE FE EE FE EE FE EE FE EE FE EE FE FE EE FE	eeeeeeeeeeeeeeee
00174348	EE FE EE FE EE FE EE FE EE FE EE FE EE FE FE EE FE	eeeeeeeeeeeeeeee
00174358	EE FE EE FE EE FE EE FE EE FE EE FE EE FE FE EE FE	eeeeeeeeeeeeeeee
00174368	EE FE EE FE EE FE EE FE EE FE EE FE EE FE FE EE FE	eeeeeeeeeeeeeeee
00174378	EE FE EE FE EE FE EE FE EE FE EE FE EE FE FE EE FE	eeeeeeeeeeeeeeee

1 is still below the garbage. There appears to move this process is not complete. Continue Trace F8 to here:

Address	Hex dump	Disassembly
00437080	F3:A5	REP MOVSD PTR ES:[EDI],DWORD PTR DS:[ESI]
00437082	8BC8	MOV ECX,EAX
00437084	83E1 03	AND ECX,3
00437087	F3:A4	REP MOVSB PTR ES:[EDI],BYTE PTR DS:[ESI]
00437089	B8 00000100	MOV EAX,10000
0043708E	2B45 F8	SUB EAX,DWORD PTR SS:[EBP-8]
00437091	3945 10	CMP DWORD PTR SS:[EBP+10],EAX
00437094	73 08	JNB SHORT mbox2w.0043709E
00437096	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]
00437099	8945 E4	MOV DWORD PTR SS:[EBP-1C],EAX
0043709C	EB 08	MOV SHORT mbox2w.004370A9
0043709E	B8 00000100	MOV EAX,10000
004370A3	2B45 F8	SUB EAX,DWORD PTR SS:[EBP-8]

He he:

```
EAX=00010000, (UNICODE "::::\")
Stack SS:[0136F4B4]=00016000
```

16000 is the new size of the actual file this bootup. In the 6000 Byte not handle it. Shift-F9 to 2 of 2 times to break in GetSystemTimeAsFileTime. Then trace wa RETN 4 to return to here:

Address	Hex dump	Disassembly
00436FE7	8B45 C4	MOV EAX,DWORD PTR SS:[EBP-3C]
00436FEA	8B4D F0	MOV ECX,DWORD PTR SS:[EBP-10]
00436FED	64:890D 000000	MOV DWORD PTR FS:[0],ECX
00436FF4	5F	POP EDI
00436FF5	5E	POP ESI
00436FF6	5B	POP EBX
00436FF7	C9	LEAVE
00436FF9	C2 0800	RETN 8

Same as above, we find the next move preparation Byte:

Stack SS:[0136F42C]=00DA0098	
EAX=01C6E543	
Address	Hex dump
00174268	04 8B 45 F0 8B C8 2B 0E 89 06 01 4E 08 89 53 34
00174278	FF 75 10 E9 B2 01 00 00 8B 45 08 8B CF 89 43 20
00174288	8B 45 FC 89 43 1C 8B 45 0C 2B 08 83 60 04 00 89
00174298	38 01 48 08 8B 4D F4 FF 75 10 89 4B 34 E9 DF 01
001742A8	00 00 8B 45 08 6A FC 89 43 20 8B 45 FC 89 43 1C

Copy pane to clipboard
Modify data
Follow address in Dump
Follow value in Dump

up access memory on Byte's region:

Address	new dump
---------	----------

Address

```
ECX=00001800 (decimal 6144.)
DS:[ESI]=00000000-4389F445
```

move to the end before. Thôi Trace F8 to move it through to the end.

Address	Hex dump	Disassembly
---------	----------	-------------

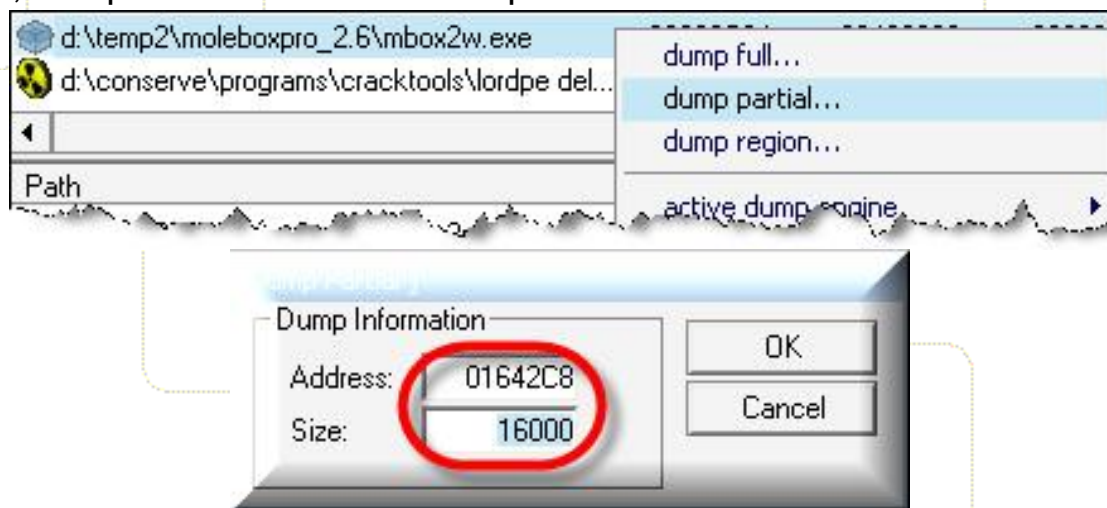
EAX=00006000

.....

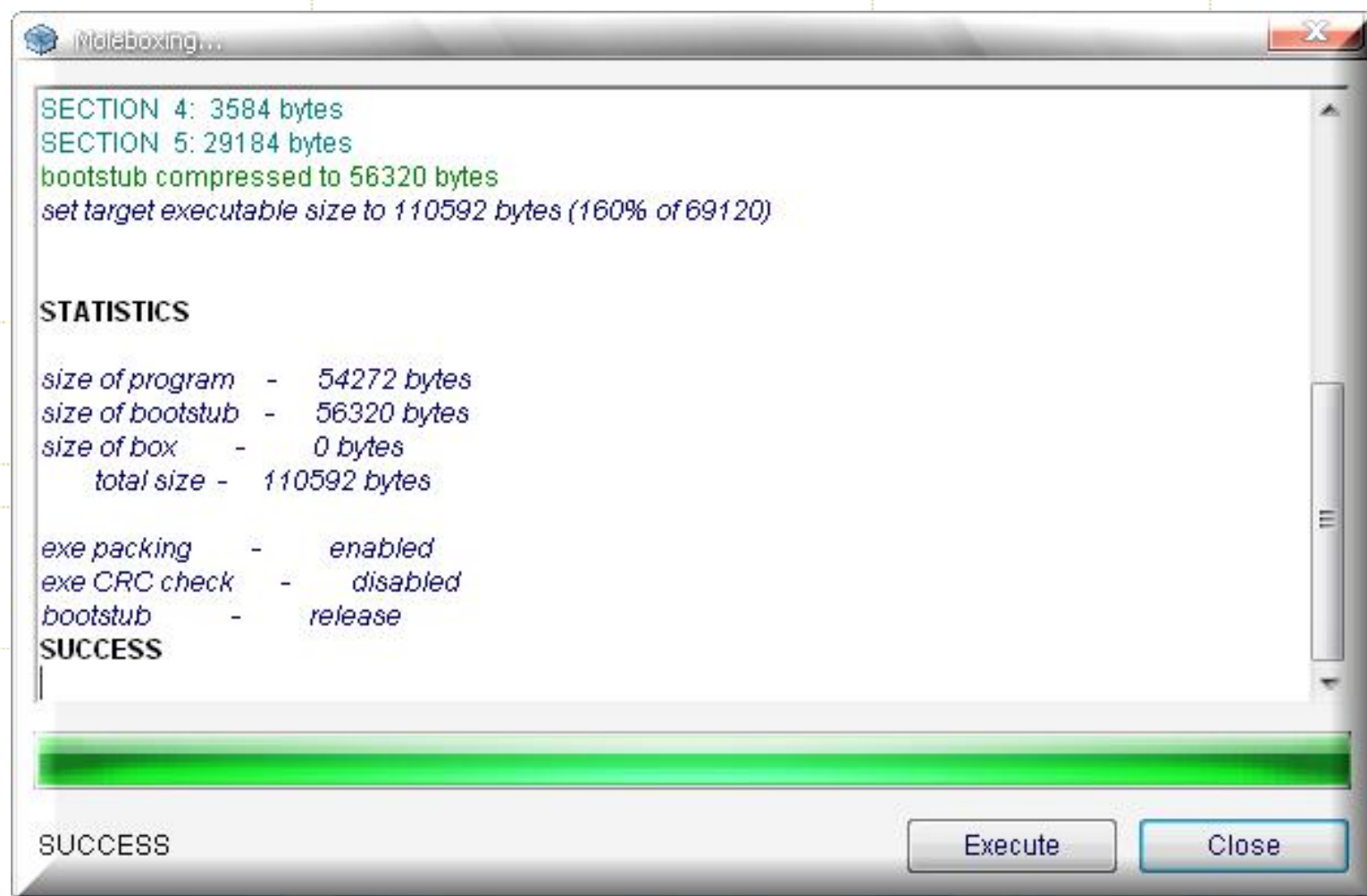
Address	Hex dump	ASCII
001742C8	45 F4 89 43 34 E9 60 01 00 00 8B 45 0C 8B 4D 08	E 0% C 4 0 . IE
001742D8	C7 03 09 00 00 00 C7 40 18 1C 40 01 10 89 4B 20	l . . . l 0 l 0 0 k
001742E8	8B 40 FC 89 4B 1C 8B 40 F8 89 48 04 8B CF 2B 08	i M e K L i M e H i +
001742F8	89 38 01 48 08 E9 54 01 00 00 83 7D EC FD 75 11	e 80 H 0 T 0 . . a j w u
00174308	FF 73 0C FF 76 28 FF 56 24 59 C7 03 09 00 00 00	s . v (U s V l . . .
00174318	59 8B 45 08 FF 75 EC 89 43 20 8B 45 FC 89 43 1C	Y i E u w e C i E n e C L
00174328	8B 45 F8 89 46 04 8B C7 2B 06 89 3E 01 46 08 8B	i E e F i l l + e > 0 F i
00174338	45 F4 89 43 34 E9 F0 00 00 00 8B 45 08 89 43 20	E r e C 4 0 = . . i E e C
00174348	8B 45 FC 89 43 1C 83 66 04 00 8B C7 2B 06 89 3E	i E n e C L a f . i l l + e >
00174358	01 46 08 8B 45 F4 89 43 34 E9 12 FF FF FF 73	0 F i E r e C 4 0 s
00174368	0C FF 76 28 FF 56 24 8B 45 08 C7 03 09 00 00 00	. v (U s i E l l . . .
00174378	C7 46 18 00 40 01 10 89 43 20 8B 45 FC 6A FD 89	l F l . 0 0 e C i E n j e
00174388	43 1C 8B 45 F8 89 46 04 8B C7 2B 06 56 89 3E 53	C L i E e F i l l + e > S
00174398	01 46 08 8B 45 F4 89 43 34 E8 48 14 00 00 83 C4	0 F i E r e C 4 0 H . . a
001743A8	14 E9 E5 00 00 00 83 7D EC FD E9 4F FF FF FF C7	q 0 s . . . a j w 0 l l
001743B8	03 07 00 00 00 EB 09 8B 7D F0 8B 75 0C 8B 4D F4 s . i j = i u . i M r
001743C8	FF 75 10 89 4B 34 56 53 E8 19 14 00 00 8B 4B 34	u j e K 4 U S s l q . . i K 4
001743D8	83 C4 0C 39 4B 30 74 21 8B 55 08 89 53 20 8B 55	a . . 9 K 0 t f i l U e S i U
001743E8	FC 89 53 1C 8B 55 F8 89 56 04 8B D7 2B 16 89 3E	" e S L i U e U i l l + e >
001743F8	01 56 08 89 4B 34 50 8B 31 C7 03 08 00 00 00 EB	0 U e K 4 P s l l . . . s
00174408	09 8B 7D F0 8B 75 0C 8B 4D F4 8B 45 08 6A 01 89	. i j = i u . i M r i E j 0 e
00174418	43 20 8B 45 FC 89 43 1C 8B 45 F8 89 46 04 8B C7	C i E n e C L i E e F i l l
00174428	2B 06 89 3E 01 46 08 89 4B 34 56 EB 55 8B 45 08	+ e > 0 F e K 4 U s U i E
00174438	8B 40 F8 89 43 20 8B 45 FC 89 43 1C 8B 45 0C 89	i M e C i E n e C L i E . e
00174448	48 04 8B 4D F0 8B D1 2B 10 89 08 01 50 08 8B 4D	H i M e i t + e 0 F i M
00174458	F4 6A FD 89 4B 34 EB 29 8B 45 08 8B 4D F8 89 43	(j e K 4 s) i E i M e C
00174468	20 8B 45 FC 89 43 1C 8B 45 0C 6A FE 89 48 04 8B	i E n e C L i E . j e H i
00174478	4D F0 8B D1 2B 10 89 08 8B 4D F4 01 50 08 89 4B	M e i t + e i M r 0 F e K
00174488	34 50 53 E8 5E 13 00 00 83 C4 0C 5F 5E 5B C9 C3	4 P S s ^ ! . . a . . ^ l r t
00174498	55 04 01 00 00 05 01 10 26 06 00 00 00 00 00	v 0 l l a s a . . 0 l

So determined is the address start saving the file is in the bootup Offset = 1642C8 with Size = 16000

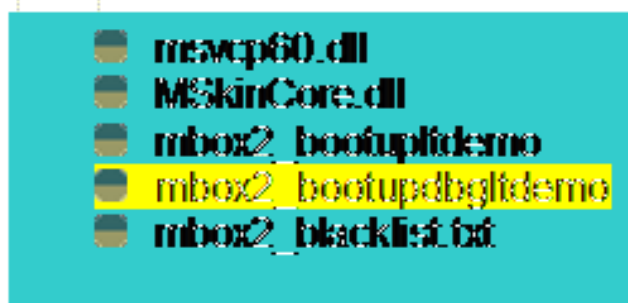
Using LordPE, dump Partial remember this particular area:



Save with its original name: mbox2_bootupltdemo. No extensions nhé. Ko Still Close Olly (1). BP temporarily put in GetSystemTimeAsFileTime go. Shift-F9 to run completely, packing process is complete, close the window pack Mole this close, but not always Mole this nhé. Now back to fix dump file, run and pack NotePad to see why:



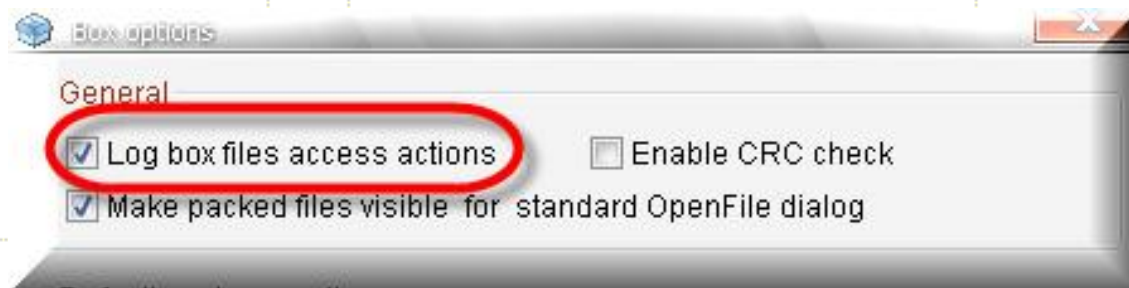
Ko also correct errors ko? But that 1 again, clearly in the list:



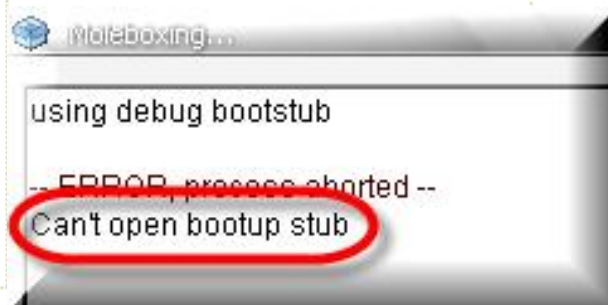
That is missing 1 files mbox2_bootupdbgldemo. Will have to file or do this? Open Option Mole's fix dump file:



Check this box:



Then again NotePad the Pack:



That is still need to file this. Word "dbg" in the file name is probably the Debug, help for the Log file when packing. Okie, Back through the Mole Olly (1), as well as on select Options, click the button before you pack and BP CreateFileA:

Command : bp CreateFileA

Click "Pack To Box," Bread. Continue Shift-F9 until you see:

Address	Value	Comment
0135F4C0	00000002	
0135F4C4	00042224	ASCII "MBOX2_BOOTUPDBGLTDEMO"
0135F4C8	00000020	
0135F4CC	00042224	ASCII "MBOX2_BOOTUPDBGLTDEMO"
0135F4D0	0135FE04	
0135F4D4	0135FE04	
0135F4D8	0135FED8	
0135F4DC	0043254C	mbox2w.0043254C
0135F4E0	00436388	CALL to CreateFileA from mbox2w.00436388
0135F4E4	00042438	FileName = "D:\TEMP2\MOLEBOXPRO_2.6\MBOX2W.EXE"
0135F4E8	00000000	Access = GENERIC_READ

Un-BP in CreateFileA go. BP set GetSystemTimeAsFileTime. And then do you know more then đây. Similar steps in finding a number of Byte initial move to be in:


Address	Hex dump	ASCII
00164280	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZÉ.♦...♦... ..
00164290	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00@..... ..
001642A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001642B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
001642C0	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	ø¶ ø.+. =¶øL=¶Th
001642D0	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program cannot
001642E0	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	be run in DOS
001642F0	6D 6F 64 65 2E 00 0D 0A 24 00 00 00 00 00 00 00	mode....\$.
00164300	8B AF FA 42 CF CE 94 11 CF CE 94 11 CF CE 94 11	i> B¶ø¶¶ø¶¶ø¶¶ø
00164310	4C D2 9A 11 C5 CE 94 11 DC C6 C9 11 CD CE 94 11	L¶¶ø¶¶ø¶¶ø¶¶ø¶¶ø
00164320	4C C6 C9 11 CC CE 94 11 CF CE 95 11 D5 CE 94 11	L¶¶ø¶¶ø¶¶ø¶¶ø¶¶ø
00164330	55 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Address	Hex dump	Disassembly
00437080	F3:A5	REP MOVSD DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
00437082	8BC8	MOV ECX,EAX
00437084	83E1 03	AND ECX,3
00437087	F3:A4	REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00437089	B8 00000100	MOV EAX,10000
0043708E	2B45 F8	SUB EAX,DWORD PTR SS:[EBP-8]
00437091	3945 10	CMP DWORD PTR SS:[EBP+10],EAX
00437094	73 08	JNB SHORT mbox2w.0043709E
00437096	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]
00437098	8945 10	MOV DWORD PTR SS:[EBP+10],EAX

Address	Hex dump	Disassembly
00437080	F3:A5	REP MOVSD WORD PTR ES:[EDI],WORD PTR DS:[ESI]
00437082	8BC8	MOV ECX,EAX
00437084	83E1 03	AND ECX,3
00437087	F3:A4	REP MOVSD BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00437089	B8 00000100	MOV EAX,10000
0043708E	2B45 F8	SUB EAX,DWORD PTR SS:[EBP-8]
00437091	3945 10	CMP DWORD PTR SS:[EBP+10],EAX
00437094	73 08	JNB SHORT mbox2w.0043709E
00437096	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]
00437098	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]

```
EAX=00010000, (UNICODE "=":::\")
Stack SS:[0135F4B4]=00019E00
```

Address	Hex dump	ASCII
00174280	62 FF FF 85 C0 74 1E 6A 00 FF 75 10 FF 75 0C FF	b ä't△j. u u.
00174290	75 08 6A 00 6A 00 6A 00 FF 82 65 FF FF 83 C4 1C	u j. j. j. 端 e-ä
001742A0	89 45 FC EB 12 FF 75 10 FF 75 0C FF 75 08 FF 15	ēEñ\$ u u u. u
001742B0	24 99 01 10 89 45 FC 8B 45 FC C9 C2 0C 00 55 8B	\$00>ēEñ'Eñr..U
001742C0	EC 51 FF 75 10 FF 75 08 FF 75 14 68 74 90 01 10	u u u u qñtē0
001742D0	E8 31 03 FF FF 83 C4 10 6A 00 33 02 8B 40 14 E8	\$14 ä- j. 3ñIñ9
001742E0	79 62 FF FF 85 C0 74 1F 6A 01 FF 75 14 FF 75 10	y b ä'tv j0 uñ u
001742F0	FF 75 0C 6A 00 6A 00 FF 75 08 E8 50 65 FF FF 83	u. j. j. u \$P e
00174300	C4 1C 89 45 FC EB 15 FF 75 14 FF 75 10 FF 75 0C	-LēEñ\$ uñ u u.
00174310	FF 75 08 FF 15 40 40 01 10 89 45 FC 8B 45 FC C9	u \$000>ēEñ'Eñr
00174320	C2 10 00 55 8B EC 51 51 83 7D 08 00 75 0F 6A 7E	T. U. i00Qā30. u* j
00174330	FF 15 90 99 01 10 33 C0 E9 9F 00 00 00 83 7D 08	\$ē00>34 f..ā30
00174340	00 74 08 8B 45 08 89 45 F8 EB 07 C7 45 F8 2C 91	. t0IēEñ\$. lEñ. a
00174350	01 10 FF 75 0C FF 75 F8 68 08 91 01 10 E8 A4 D2	0 u. u h0ā0>3ññ
00174360	FF FF 83 C4 0C 8B 40 08 E8 C2 A4 FF FF 89 45 FC	ä-. iñ0-ñ ēEñ
00174370	83 7D FC 00 75 00 FF 75 08 68 EC 90 01 10 E8 83	ā)ñ. uP uñ h0ē0>ēEñ
00174380	D2 FF FF 59 59 8B 55 0C 8B 40 08 E8 7A AC FF FF	π VYIU. iñ\$ā
00174390	89 45 FC 83 7D FC 00 75 20 FF 15 14 99 01 10 50	ēEñā)ñ. u- 3ñ00>F
001743A0	FF 75 08 68 C8 90 01 10 E8 59 D2 FF FF 83 C4 0C	uñ h0ē0>3ññ ä-
001743B0	FF 15 14 99 01 10 85 C0 75 08 6A 7E FF 15 90 99	3ñ00>ä uñ j \$ē0
001743C0	01 10 33 C0 EB 16 FF 75 FC FF 75 08 68 9C 90 01	0>34. uñ uñ h0ē0
001743D0	10 E8 30 D2 FF FF 83 C4 0C 8B 45 FC C9 C3 55 8B	3ñ0ñ ä-. iEñrUñ
001743E0	EC 6A 00 FF 75 08 E8 38 FF FF FF 59 59 5D C2 04	u j. uñ \$8 VYI+>
001743F0	00 55 8B EC FF 75 10 FF 75 08 68 34 91 01 10 E8	. U. i u uñ h4ē0>3



Dump Information

Address: 164280

Size: 19E00

OK

Cancel

file:///C:/RCE%20Unpacking%20eBook%20Tr...ox%20Pro%202.6%20Trial%20-Volume%201.htm (41 of 43) [1/9/2009 9:45:00 LithiumLi]

Name	Size
readme.txt	2 KB
msvc60.dll	396 KB
MSkinCore.dll	152 KB
molebox.chm	121 KB
mbox2w-log.txt	1 KB
mbox2w_layer1.exe	416 KB
mbox2w_dump -log.txt	1 KB
mbox2w_dump_.exe	424 KB
mbox2w_dump.exe	416 KB
mbox2w.exe-up.txt	2 KB
mbox2w.exe	509 KB
mbox2w	1 KB
mbox2c.exe	19 KB
mbox2_bootupltdemo	88 KB
mbox2_bootupdbgltdemo	104 KB
license.txt	6 KB

IV.Ending Volume 1

Of course the process of packing the Option which will take place will soon. May be called or not unpack DONE? Khoan you set many, the nay-hour unpack trick as the Win XP SP 2. Take the set of files that last 1 Win XP SP1 running dump file and fix:



Program terminated by internal error
Please send file
C:\Sub Local\MoleBoxPro_2.6\mbox2w_dump_-log.txt
to support@mail

Reason

```
SEH exception 0xc0000005
Code address 0x7c901005, access to address 0x7c901005

CS :0x0000001B SS :0x00000023 DS :0x00000023
ES :0x00000023 FS :0x00000038 GS :0x00000000
EAX:0x00433710 EDX:0x00000000 ECX:0x0012FE24
ESP:0x0012FDA0 EBP:0x0012FDE8 EIP:0x7C901005
ESI:0x0015233F EDI:0x70A71A29

0x0012fda0: 0x0043a3f8 0x004697a0 0x70a71a29 0x0015233f
0x0012fdb0: 0x00000000 0x00000000 0x0012fe24 0x00150000
0x0012fdc0: 0x0012ff24 0x00f12736 0x0012fe10 0x00403f9a
0x0012fdd0: 0x0012fe24 0x0012f9c8 0x0012ff18 0x0043254c
```

MoleSkin framework (c) MoleStudio.com, 2002

Close

Send bugreport

It is informed by children and SEH mouse obnoxious. The DONE, do not say what the

monkey house. A file unpack / crack can only run "NGON" on you or on 1 few machines, or just run on several OS ko means you've completed the process Unpacking / Cracking start. Do not say "I test on 10 May in which always oi", must do all OS machines are the same. You should of course also have the machine will do file is run, even for the same OS as SP2. Here is more specific SP1 i can run. So do excavation to fix errors but hí hung with the result: "Only my work system", you just hug the files that do not carry gas for anyone with a bad hand. Ah, of course is 1 month deal is hot: unpack / crack 1 file on SP1, but then the file and do this work on SP2.

Therefore, to say done, all you have to fix that may work All OS, (can add Win 2000, Win 98 is also sometimes by another name, the API function is private). And this trick will presented in Volume 2 if you have time. Volume 1 to the end is here! Hi all.

Big thanks to:

All REA's members: Computer_Angel, Moonbaby, Zombie, hacnho, benina, kienmanowar, rongchaua, Deux, Merc, hoadongnoi, the_lighthouse, TQN, light.phoenix, hytkl, tlandn, hurt_heart, dzungltvn, Zoi, littleboy84, haule_nth, takada, Why not bar, iamidiot, Akira, dump, thienthandien, [kid], ...

Special thanks to:

fly, stephenteh, Gabri3l, MaDMAAn_H3rCuL3s, CondZero, Ricardo Narvaja + NCR, lena151, haggag, ARTeam, snd, RES, CrackLatinos, all unpack.cn ... Authors who created tools and you.



Written by Trickyboy - 2006

Import elimination Debugblocker + + Nanomites

SoftWare : **AoA DVD Ripper 3.81**

Homepage: <http://www.aoamedia.com/>

Packed : *Debugblocker + +
Code Splicing Nanomites*

Crack Tool **1.** *OllyDBG by
hacnho.*

2. *LordPE Deluxe*

1.4-by yoda

3. *Import
REConstructor 1.6
Final*

4. *ArmInline 0.71*

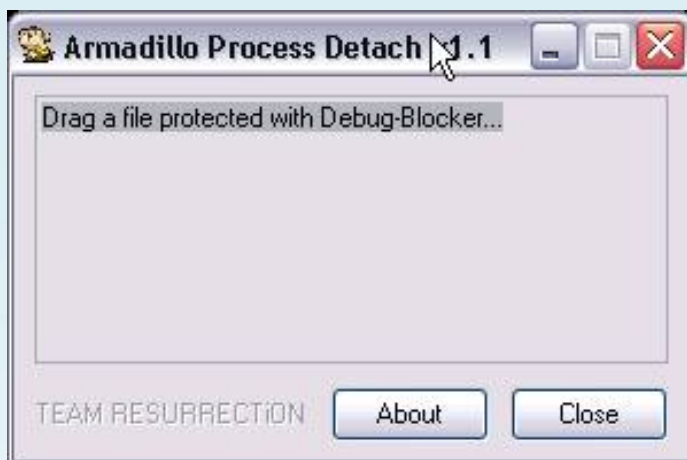
5. *ArmaDetach 1.1*

6. *CFF ExplorerII*

Author : **Why Not Bar**

This target children meat orders! Hehe ... not speechify working days.

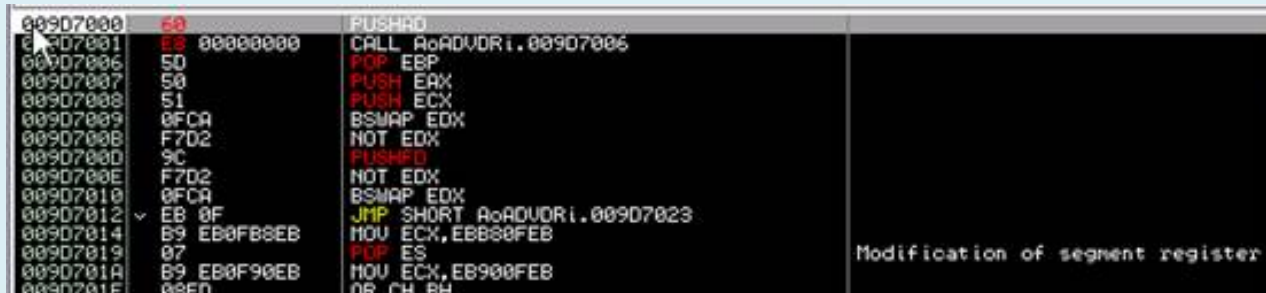
1. Running ArmaDetach 1.1



2. Drag the file "AoADVDRIpper.exe" released into the window of ArmaDetach



3. Select Open Olly Child process ID and Attach. F9, F12 and Edit to 60E8



4. Running Scripts "Armadillo Standard unpack"



5. IAT and now find we are as follows:

IAT Start: 011CB378

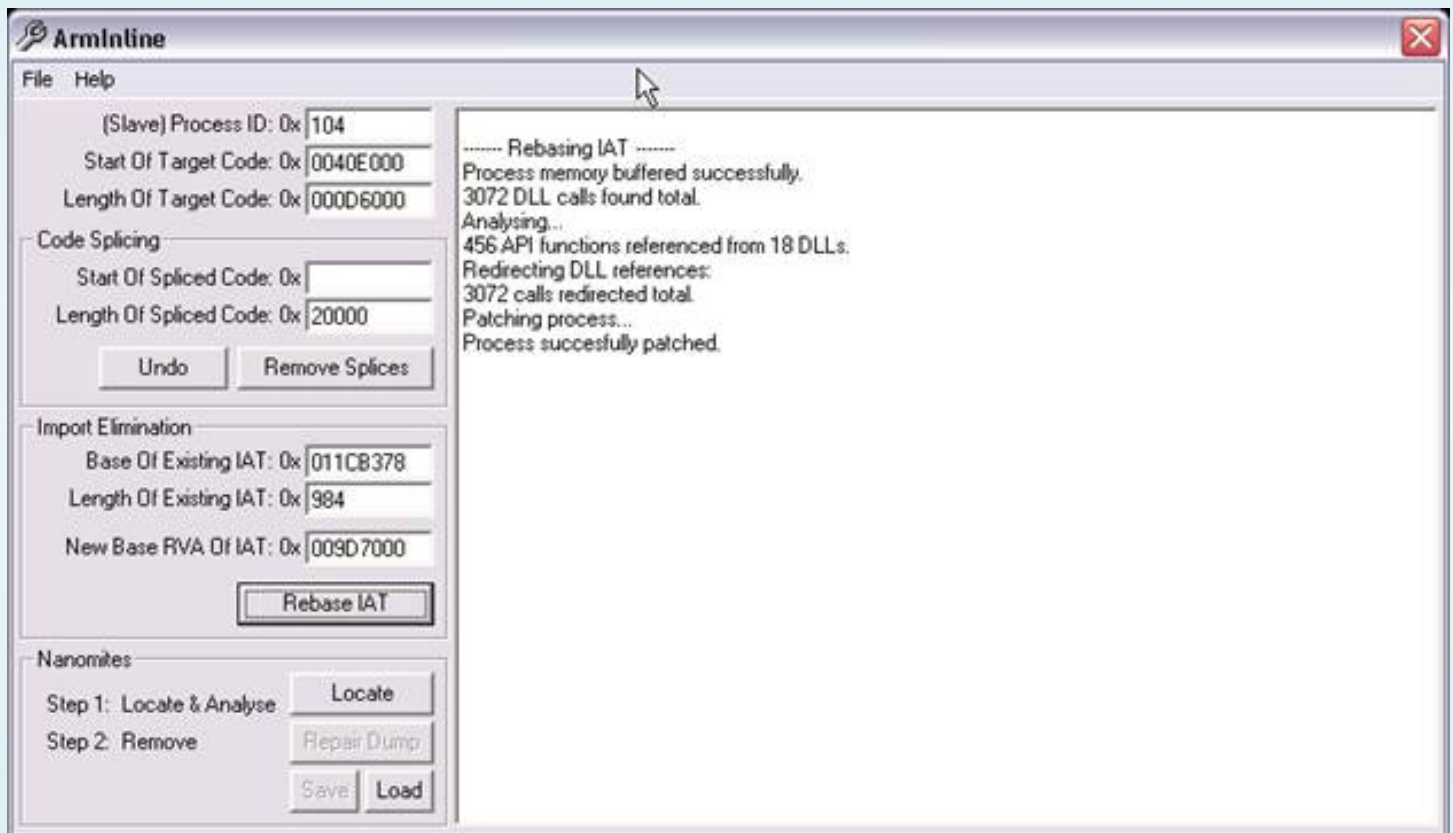
Address	Value	Comment
011CB374	00000000	
011CB378	0003612F	
011CB37C	010801E4	
011CB380	7C801A24	kernel32.CreateFileA
011CB384	77D4B46E	USER32.SetRect
011CB388	77BE5908	MSACM32.acmDriverEnum
011CB38C	77D8050B	USER32.MessageBoxA
011CB390	77D4C57E	USER32.WindowFromPoint
011CB394	7C80B529	kernel32.GetModuleHandleA
011CB398	77D64E3E	USER32.GetLastActivePopup

IAT End: 011CBCFC

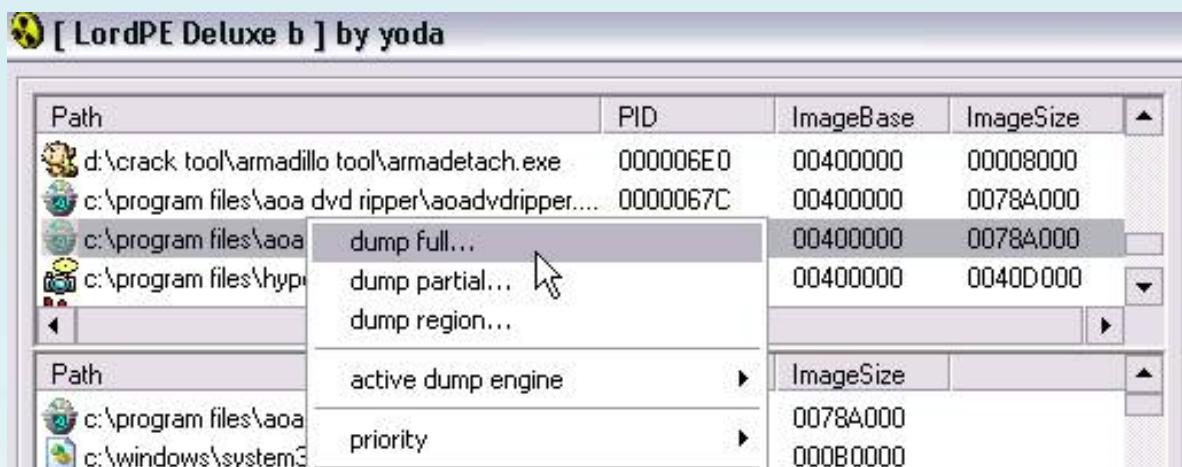
Address	Value	Comment
011CBCE8	01126F04	kernel32.WriteFile
011CBCEC	7C810F9F	
011CBCE0	012F0004	
011CBCE4	010C0115	
011CBCE8	011CB380	
011CBCEC	00000000	
011CB000	00000000	
011CB004	00000001	
011CB008	00000000	
011CB00C	00000000	

IAT len: 984

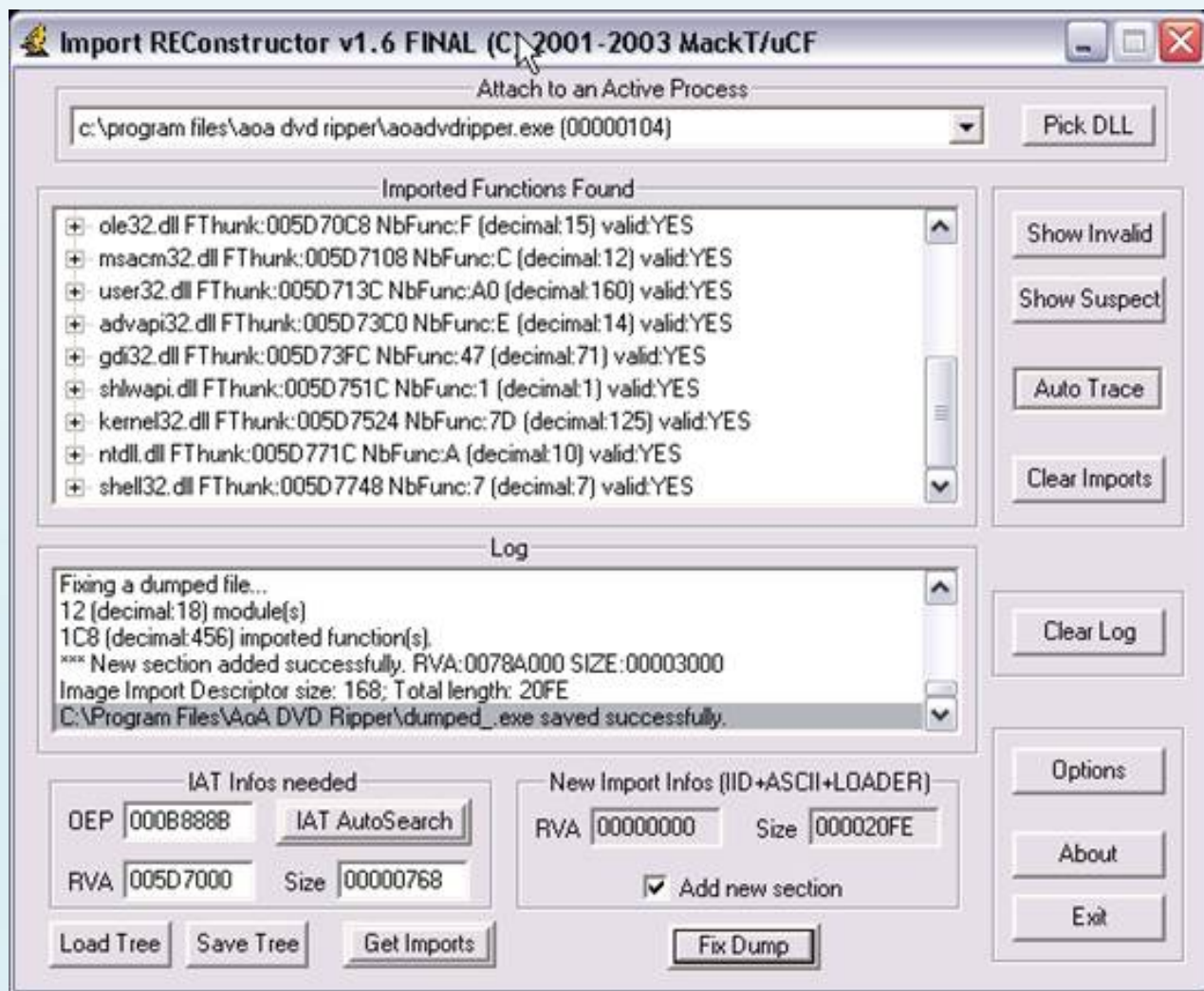
6. Now Fix **Import elimination**. ArmInline Open and complete information (



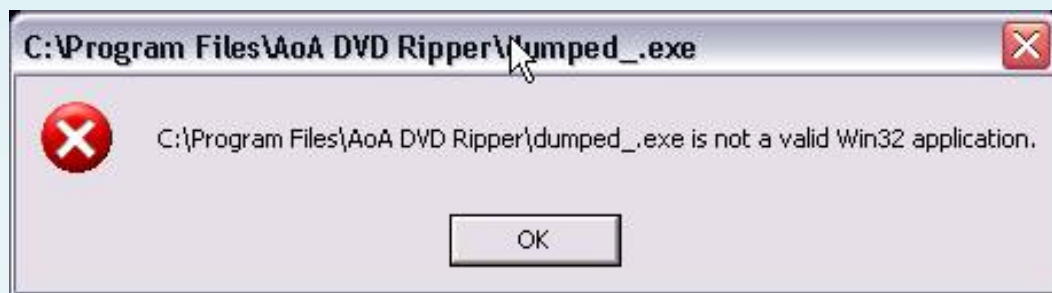
7. Using LordPE dump Full stop!



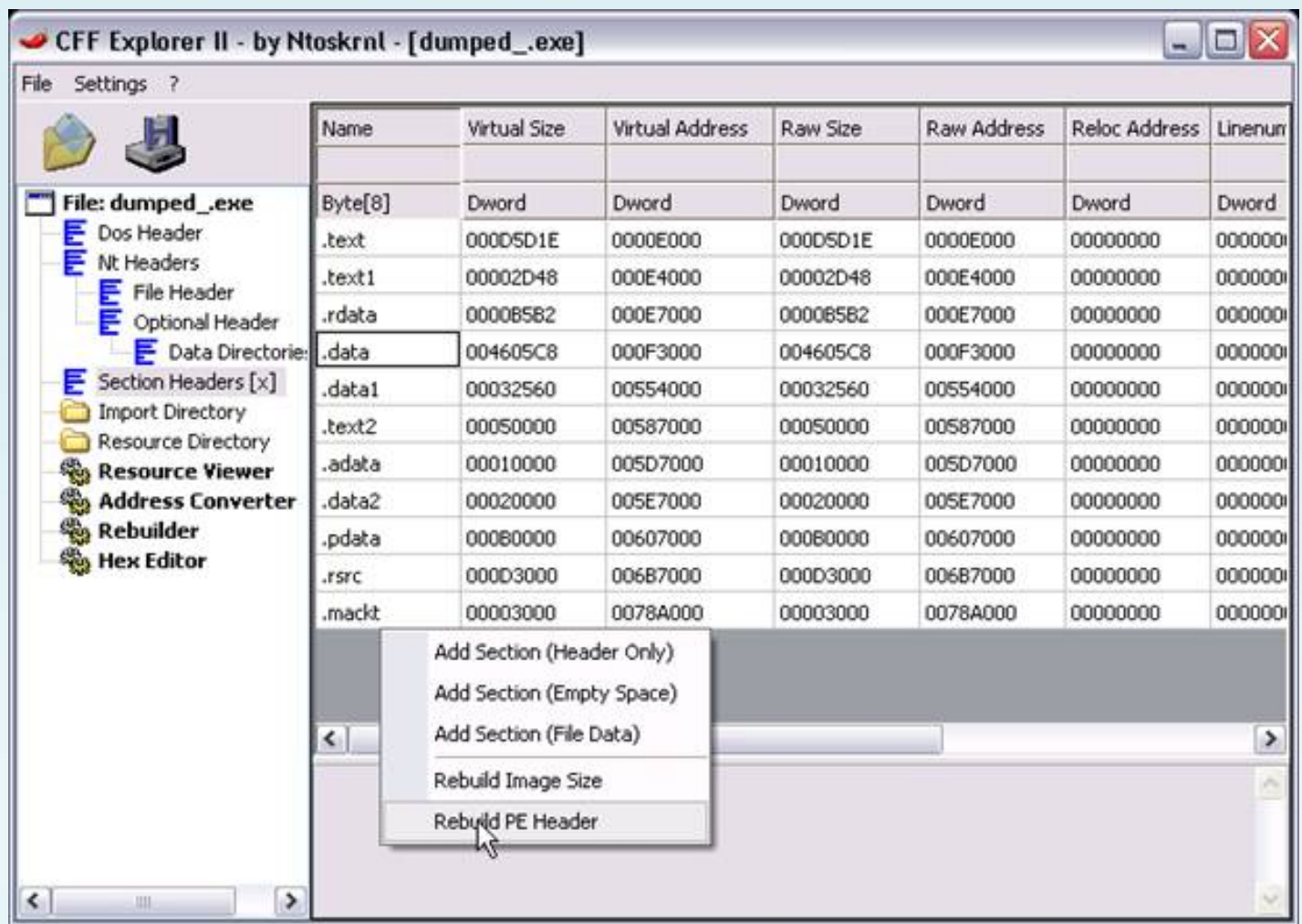
8. M ImportREC fill in the information and dump Fix



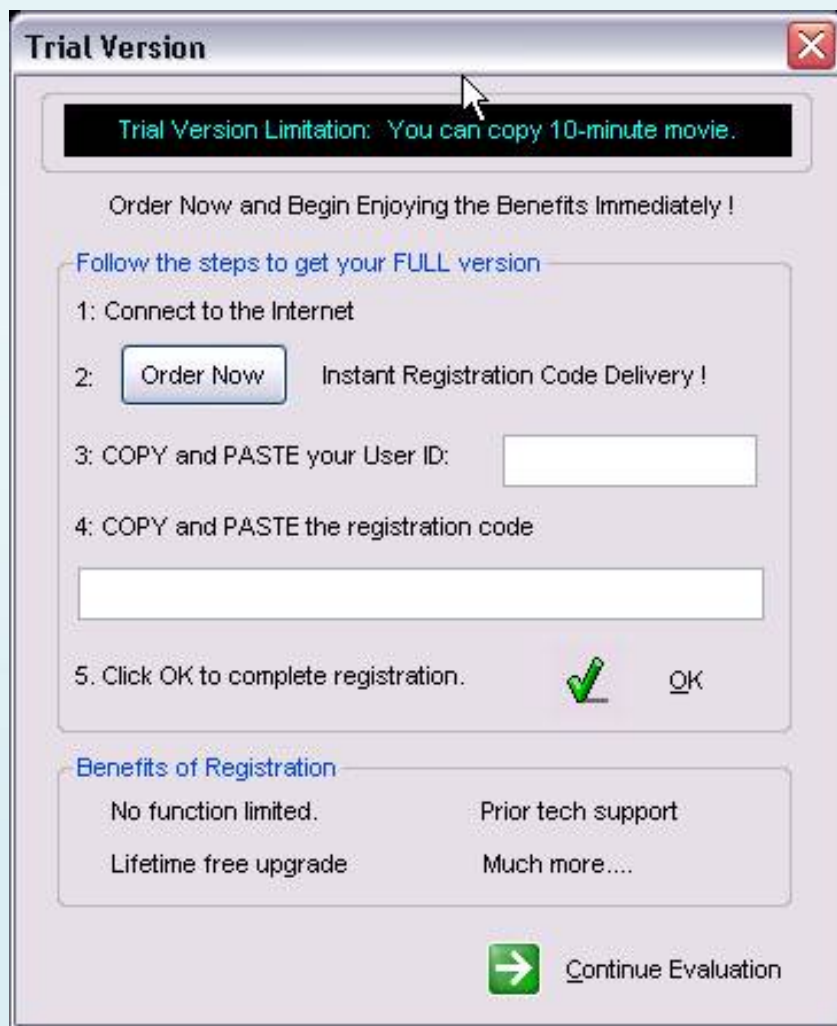
9. Test-File "dumped_.exe"



10. What more here! According sure they think it is PE Header gòi tua te! Fix it all. Here they use ExplorerII CFF. You choose like home



11. Save then run back and try to see why.



12. Hu gòi complete run! Now we make meat baby [Nanomites](#). Click  [Continue Evaluation](#) to the primary.



13. Nice ha! Next click the button  Register, And type the following:



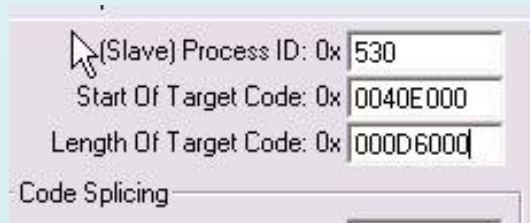
14. Click   we found as follows:




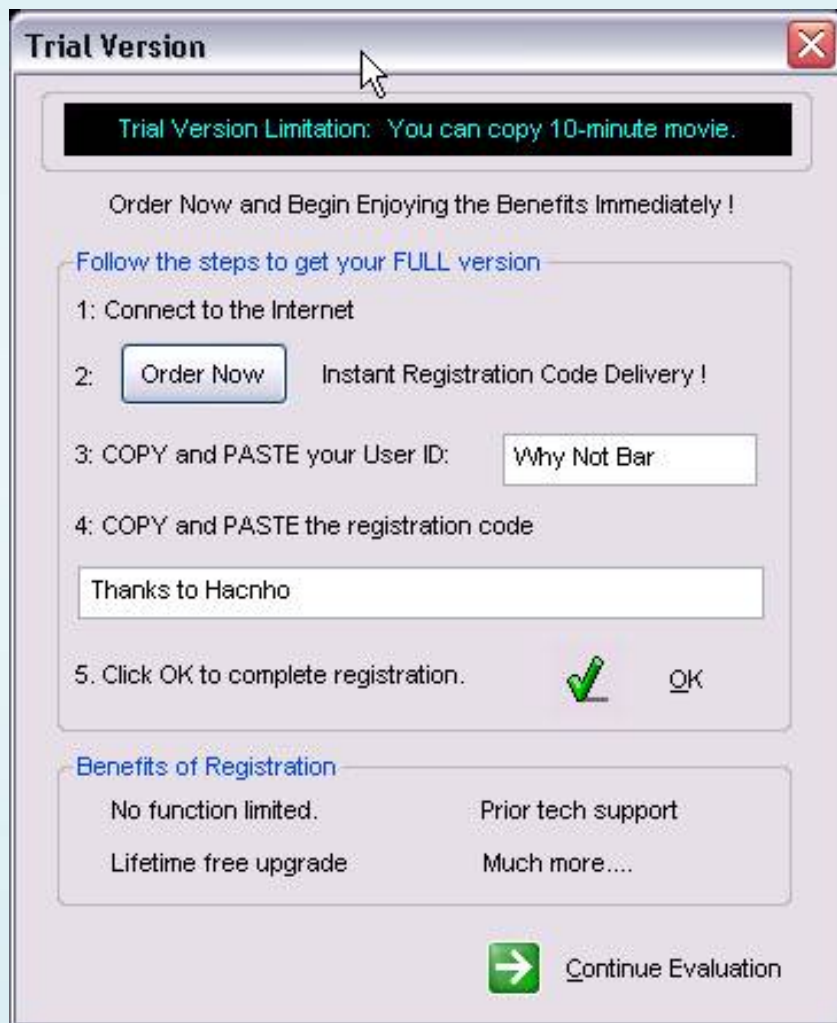
15. Never mind! Ministry that the Nano! Cute ha! Load File "dumped_.exe" to olly and PID

Process	Name	Window	Path
00000168	smss		\SystemRoot\System32\smss.exe
00000178	HprSnap5	HyperSnap-DX - [Snap17]	C:\Program Files\HyperSnap-DX 5\HprSna
000001A8	csrss		\\?C:\WINDOWS\system32\csrss.exe
000001C0	winlogon		\\?C:\WINDOWS\system32\winlogon.exe
000001EC	services		C:\WINDOWS\system32\services.exe
000001F8	lsass		C:\WINDOWS\system32\lsass.exe
00000284	svchost		C:\WINDOWS\system32\svchost.exe
000002BC	svchost		C:\WINDOWS\system32\svchost.exe
000002EC	svchost		C:\WINDOWS\System32\svchost.exe
00000354	svchost		C:\WINDOWS\system32\svchost.exe
00000360	WINWORD	Highlight (Yellow)	C:\Program Files\Microsoft Office\OFFI
0000037C	svchost		C:\WINDOWS\system32\svchost.exe
0000046C	wdfmgr		C:\WINDOWS\system32\wdfmgr.exe
000004E8	alg		C:\WINDOWS\System32\alg.exe
00000530	dumped_		C:\Program Files\AoA DVD Ripper\dumped
00000564	NOTEPAD	API for crack.txt - Notepad	C:\WINDOWS\system32\notepad.exe
000006C0	Explorer	SysFader	C:\WINDOWS\Explorer.EXE
000006D0	dumped_	Trial Version	C:\Program Files\AoA DVD Ripper\dumped
000007AC	UniKey	UniKey 3.62	C:\Program Files\UniKey\UniKey.exe

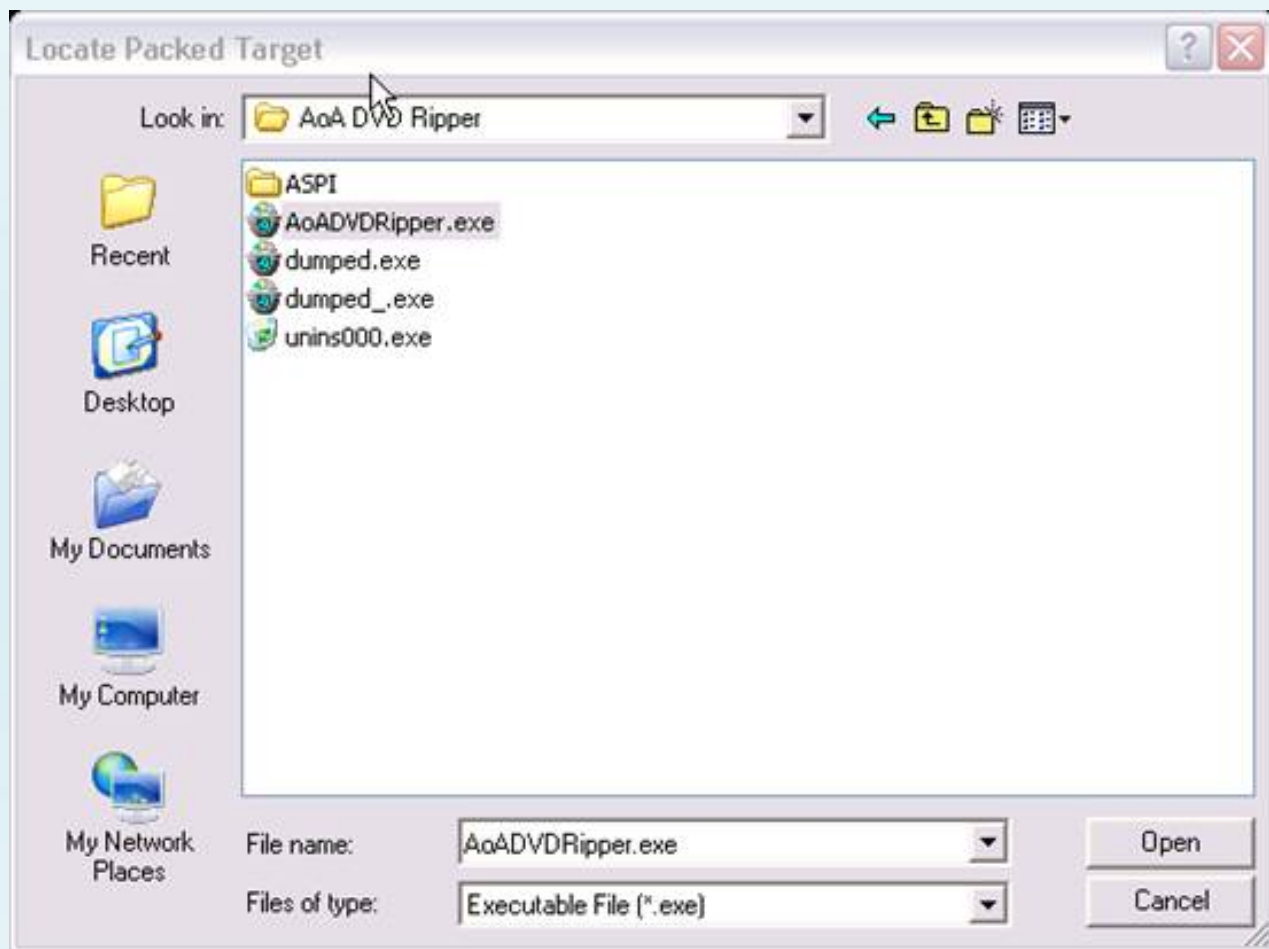
16. ArmInline open and filled as follows:



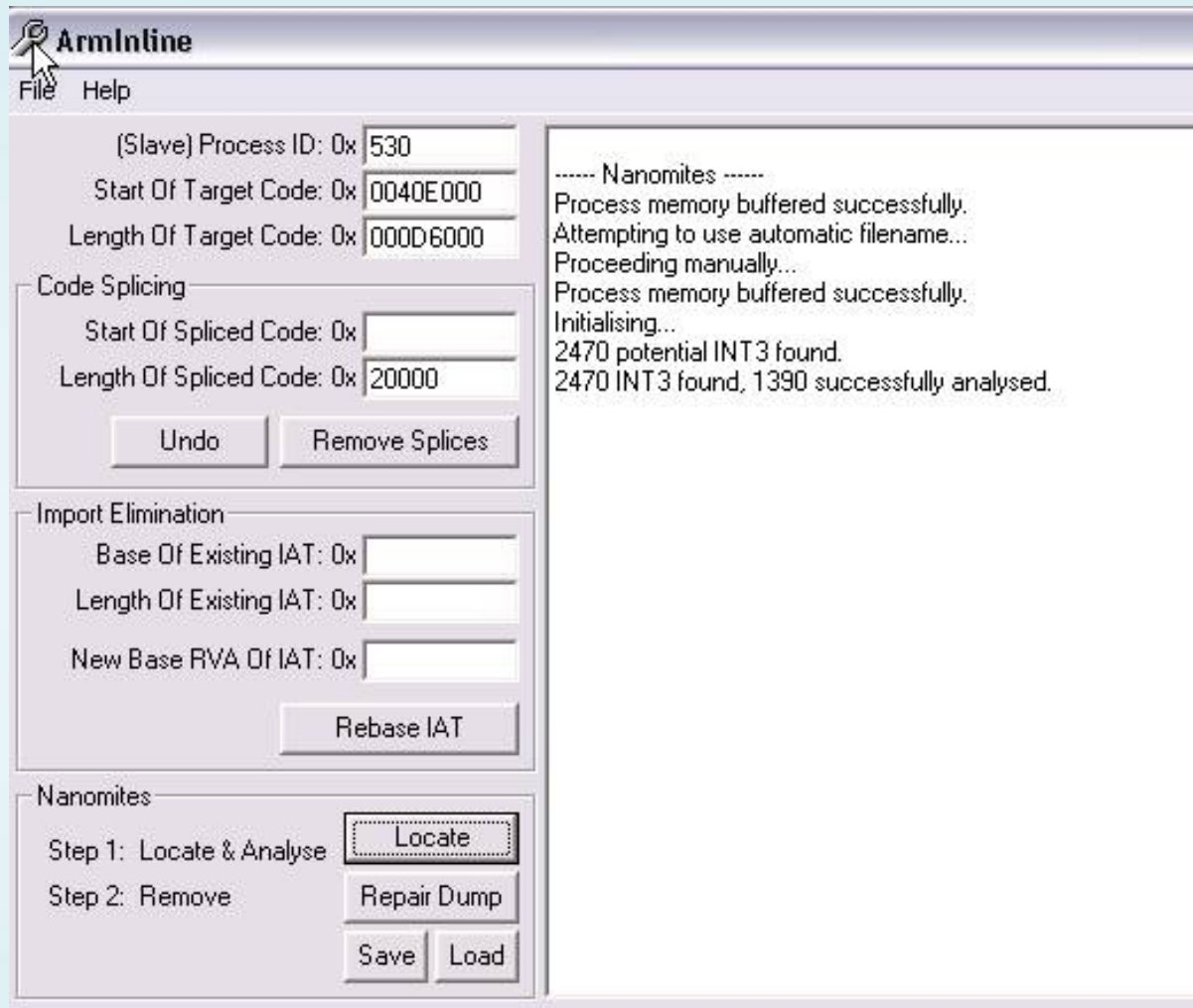
17. Click  , Enter the following



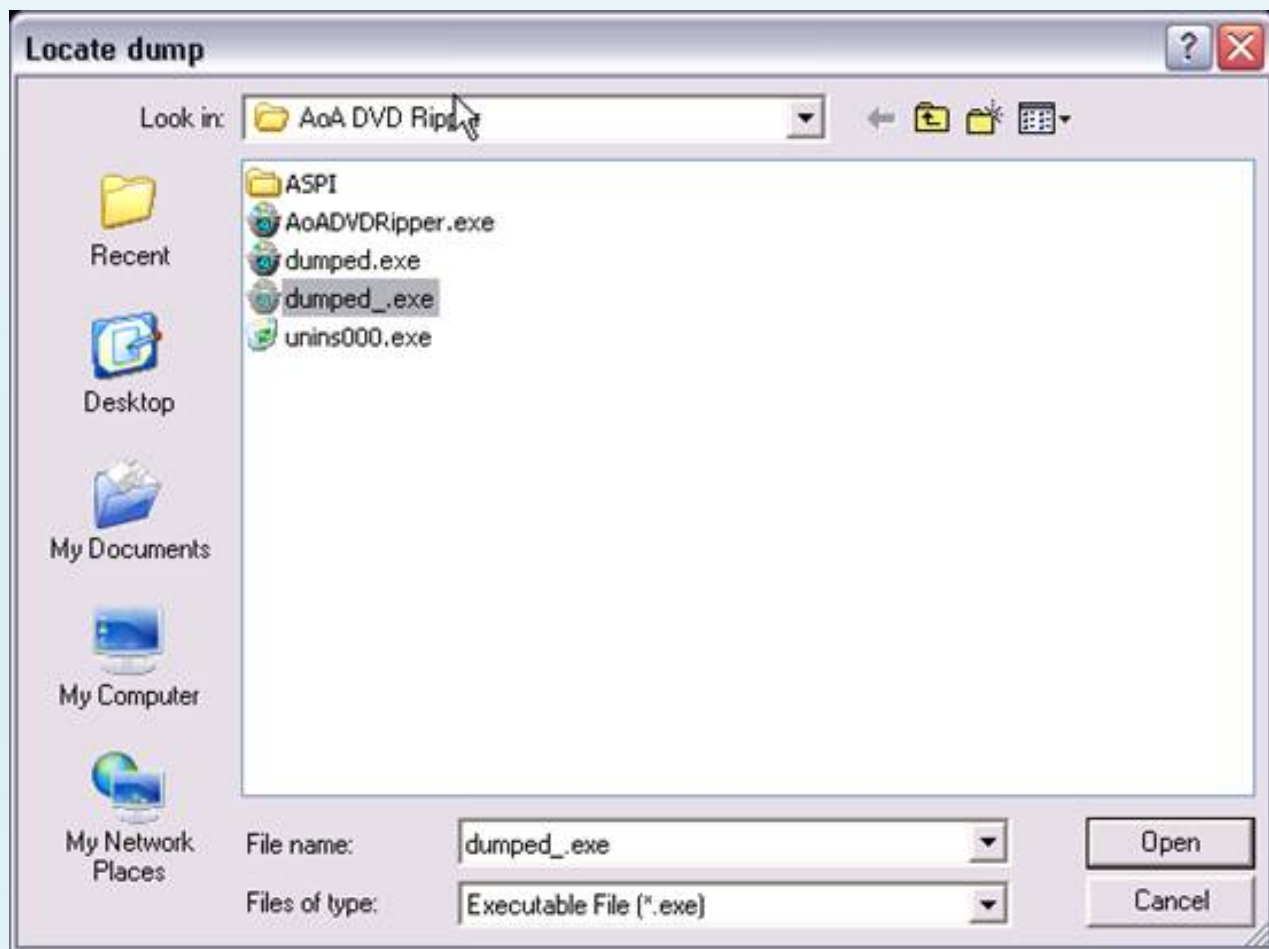
18. Click OK to 1, next select the file "AoADVDRipper.exe"



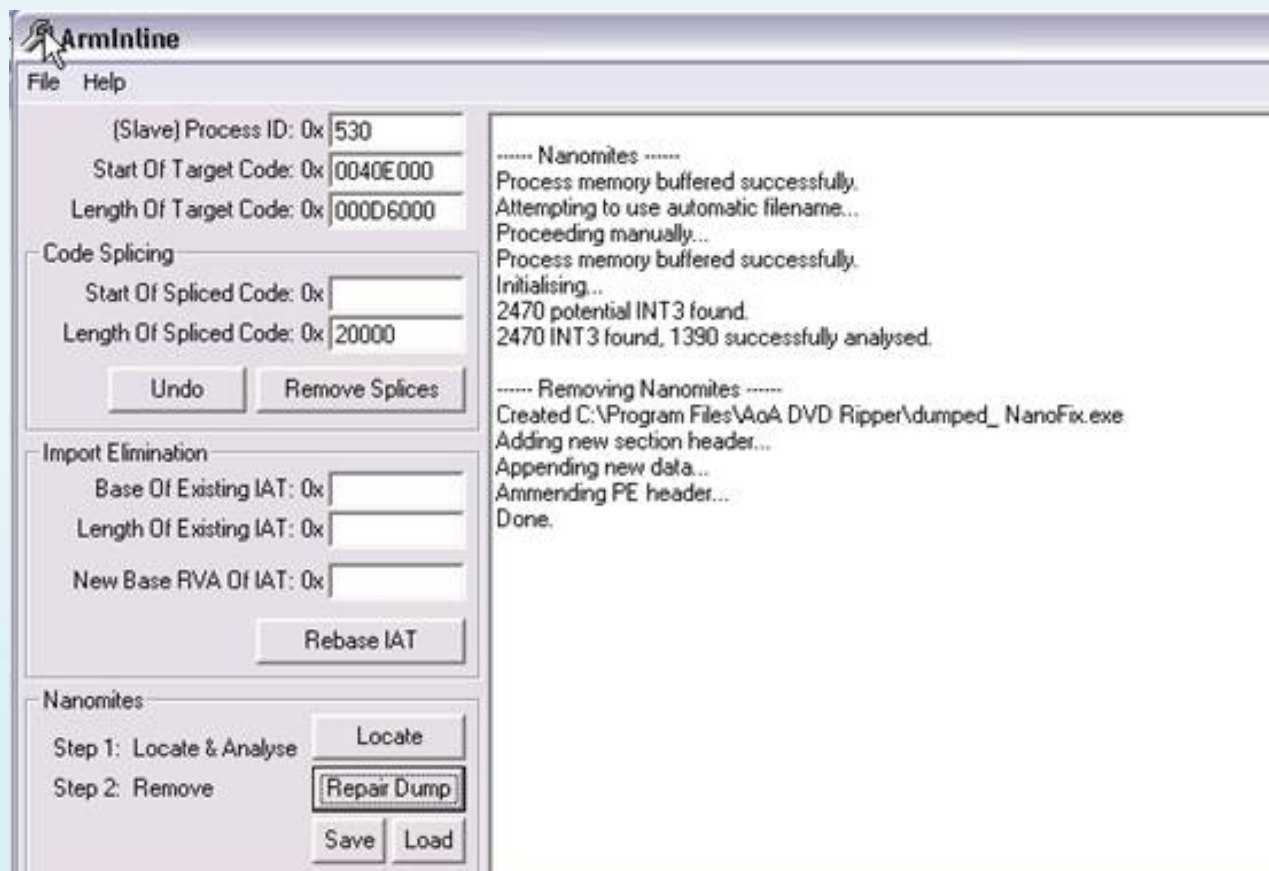
19. At Nag as the type and then click OK, back window ArmInline you see the following:



20. Click **Repair Dump** and select File "dumped_.exe"



21. Done gọi là you! See



22. Run test considered stars



23. Oh! Yeah. Unpack Done !!!!!!!!!!!!!!!!!!!!!!! The Crack ha!'s Not for a few songs! Was also easy to treat it.

Written by

Not

Bar

Why

ASF Converter v2.68: Cracking unpacking Armadillo 3.78 Standard Protection Manual + + Reduce filesize.

[Target]: **ASF Converter v2.68**

[Website]: <http://www.Boilsoft.com>

[Author]: **LightPhoenix**

[Email]: [light\(dot\)Phoenix\(at\)gmail\(dot\)com](mailto:light(dot)Phoenix(at)gmail(dot)com)

[At Contact]: [www.reaonline.net / forum](http://www.reaonline.net/forum)

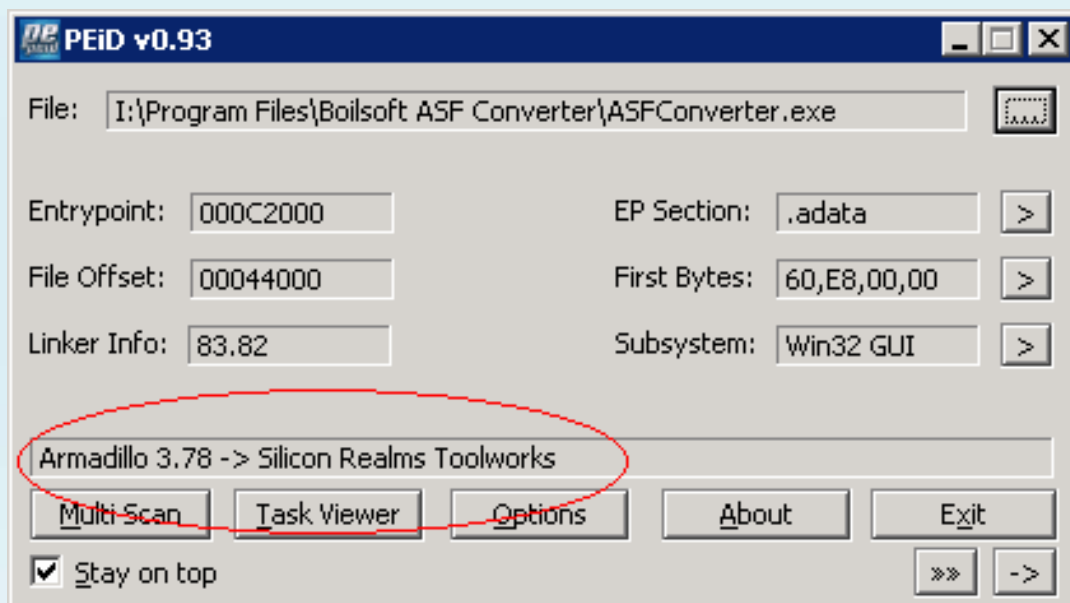
The settings should be

- 1.) Olly Debug v1.1
- 2.) LordPE
- 3.) Import Reconstructor v1.6 Final
- 4) PEID 0.93

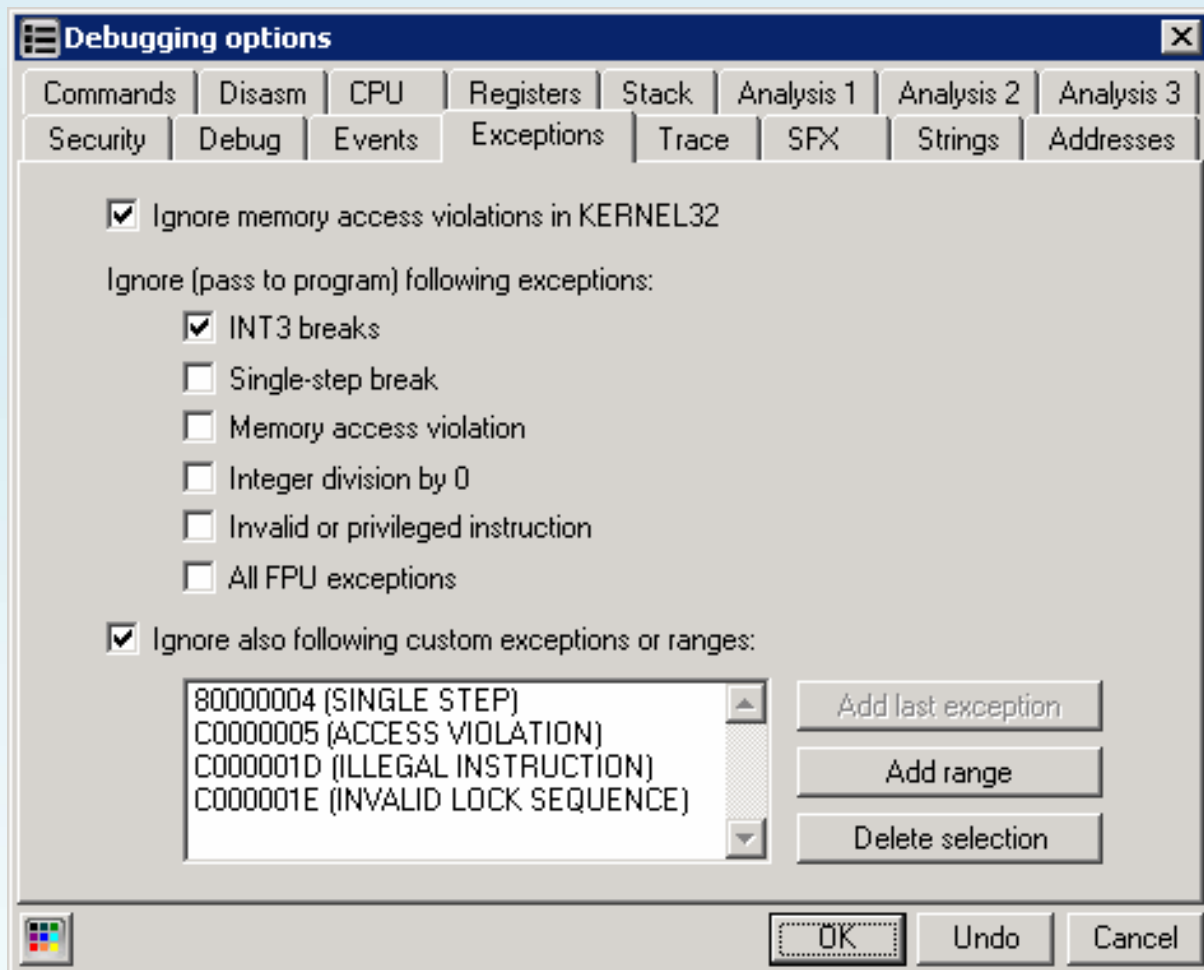
(You culay the medical tool in Hacnho gioithieu series tut about unpack a r m e adillo the tone Ok! J)

PART 1: MUP Armadillo

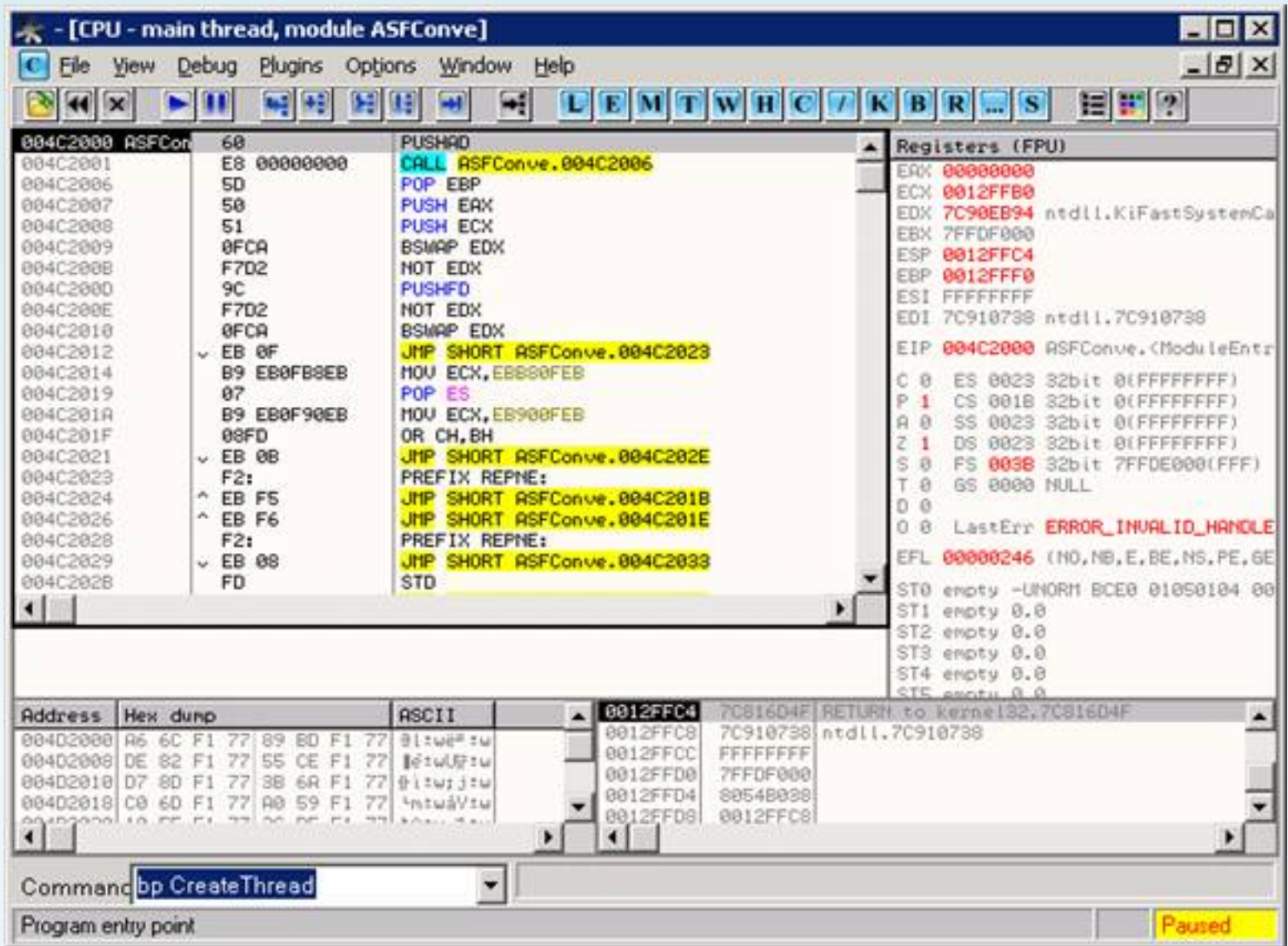
Except in the first, we survey this name. L click italy PEID test, the q u a:



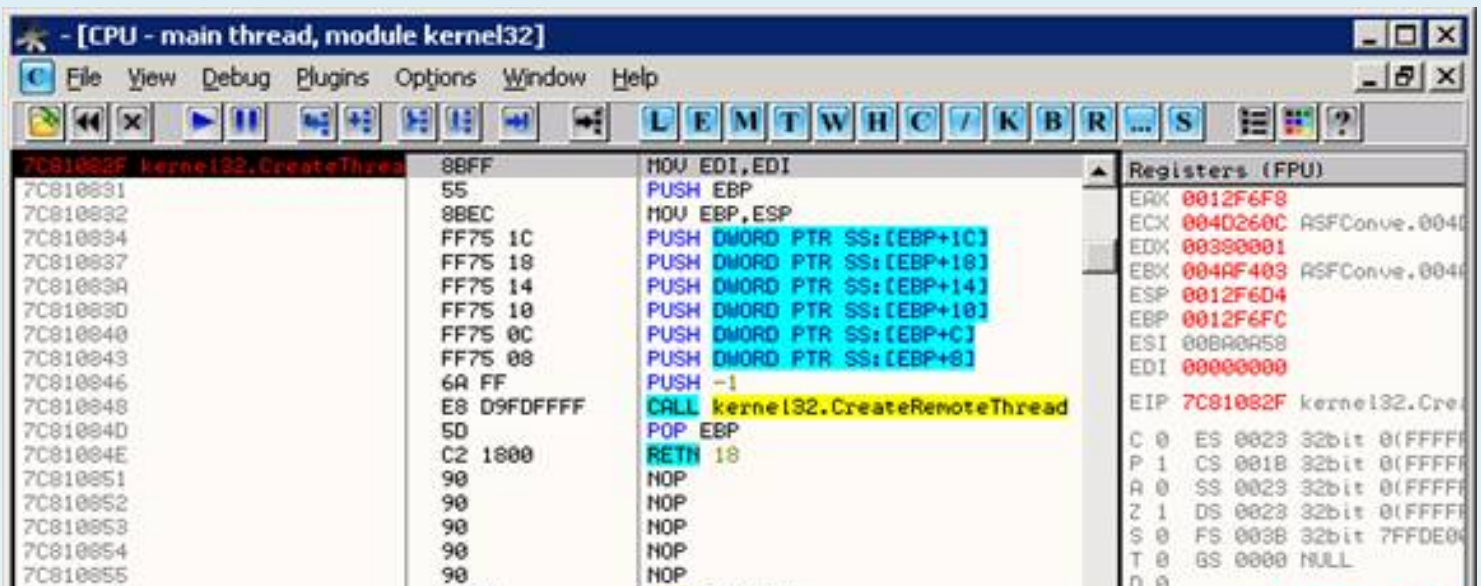
Chaythunophat, otask m m anagerlen, t h m otprocessASFConverter.exeduyhat way -> hehe, nokhongbiDebug-blocker, quaonroi, mupthoi.MoOllyDbgra (odayemdung OllyShadow), load targ e tlen, put options exception as follows:



A place in the BP reateThread C:



Running time: F9: hienrathongbao it: "Don 't know how to ...", m devilish, you OK, just a italy the IEP, Shift + F 9. Again having "privileged instruction", Shift + F9 lannua. Ha, break it at CreateThread rui:



7C810853	90	NOP	S 0 FS 0038 32bit 7FFDE0
7C810854	90	NOP	T 0 GS 0000 NULL
7C810855	90	NOP	D 0
7C810856	33ED	XOR EBP,EBP	O 0 LastErr ERROR_FILE_NO
7C810858	53	PUSH EBX	EFL 00200246 (NO,NB,E,BE,
7C810859	50	PUSH EAX	ST0 empty -UNORM BCE0 0105
7C81085A	6A 00	PUSH 0	ST1 empty 0.0
			ST2 empty 0.0
			ST3 empty 0.0
			ST4 empty 0.0
			ST5 empty 0.0

EDI=00000000

Address	Hex dump	ASCII			
004D2000	A6 6C F1 77 89 ED F1 77	01:we" :u	0012F6D4	00B7BF65	CALL to CreateThread from 00B7BF5F
004D2008	DE 82 F1 77 55 CE F1 77	02:u0F :u	0012F6D8	00000000	pSecurity = NULL
004D2010	D7 8D F1 77 38 6A F1 77	03:u;j :u	0012F6DC	00000000	StackSize = 0
004D2018	C0 6D F1 77 A0 59 F1 77	4ntu3V :u	0012F6E0	00B7C826	ThreadFunction = 00B7C826
004D2020	10 7F F1 77 7C 8F F1 77	04:u :u	0012F6E4	00000000	pThreadParam = NULL
004D2028	10 7F F1 77 7C 8F F1 77	05:u :u	0012F6E8	00000000	CreationFlags = 0

Command: bp CreateThread BP address, string - Break with condition

Breakpoint at kernel32.CreateThread

Paused

Ctrl + F9 to denhay ret, F7. L ai Ctrl + F9, F7 more. Now I'm here:

Address	Hex dump	Disassembly
00B90259	59	POP ECX
00B9025A	BF 580ABA00	MOV EDI,0BA0A58
00B9025F	8BCF	MOV ECX,EDI
00B90261	E8 1D7EFDFF	CALL 00B68083
00B90266	84C0	TEST AL,AL
00B90268	75 09	JNZ SHORT 00B90273
00B9026A	6A 01	PUSH 1
00B9026C	8BCF	MOV ECX,EDI
00B9026F	F8 F7D3F0FF	CALL 00B60650

00B90264	64 01	PUSH 1
00B9026C	8BCF	MOV ECX,EDI
00B9026E	E8 E7D3F0FF	CALL 00B6D65A
00B90273	B9 C0FBB900	MOV ECX,0B9FBC0
00B90278	C705 30C2B900 C	MOV DWORD PTR DS:[B9C230],0B9DEC0
00B90282	E8 62550000	CALL 00B957E9
00B90287	6A 00	PUSH 0
00B90289	E8 5B550000	CALL 00B957E9
00B9029C	EB	POP ECX

Scroll down to balance're a tí, a 00B9032C of you, that we timt h:

- [CPU - main thread]

Address	Hex dump	Assembly
00B90308	FF76 10	PUSH DWORD PTR DS:[ESI+10]
00B9030B	2BCA	SUB ECX,EDX
00B9030D	FFD1	CALL ECX
00B9030F	EB 10	JMP SHORT 00B9032E
00B90311	83FA 01	CMP EDX,1
00B90314	75 1B	JNZ SHORT 00B90331
00B90316	FF76 04	PUSH DWORD PTR DS:[ESI+4]
00B90319	8B50 54	MOV EDX,DWORD PTR DS:[EAX+54]
00B9031C	3350 4C	XOR EDX,DWORD PTR DS:[EAX+4C]
00B9031F	FF76 08	PUSH DWORD PTR DS:[ESI+8]
00B90322	3350 2C	XOR EDX,DWORD PTR DS:[EAX+2C]
00B90325	6A 00	PUSH 0
00B90327	FF76 0C	PUSH DWORD PTR DS:[ESI+C]
00B9032A	2BCA	SUB ECX,EDX
00B9032C	FFD1	CALL ECX
00B9032E	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX
00B90331	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]
00B90334	5F	POP EDI
00B90335	5E	POP ESI
00B90336	C9	LEAVE
00B90337	C3	RETN
00B90338	837C24 08 01	CMP DWORD PTR SS:[ESP+8],1

Registers (FPU)

Register	Value
EAX	00000150
ECX	7C8107FD kernel32.7C8107FD
EDX	7C90EB94 ntdll.KiFastS
EBX	004AF403 ASFConve.004AF403
ESP	0012F704
EBP	0012F714
ESI	004DA098 ASFConve.004DA098
EDI	00000004
EIP	00B90259

Command bp CreateThread

BP address, string – Break with condition

Paused

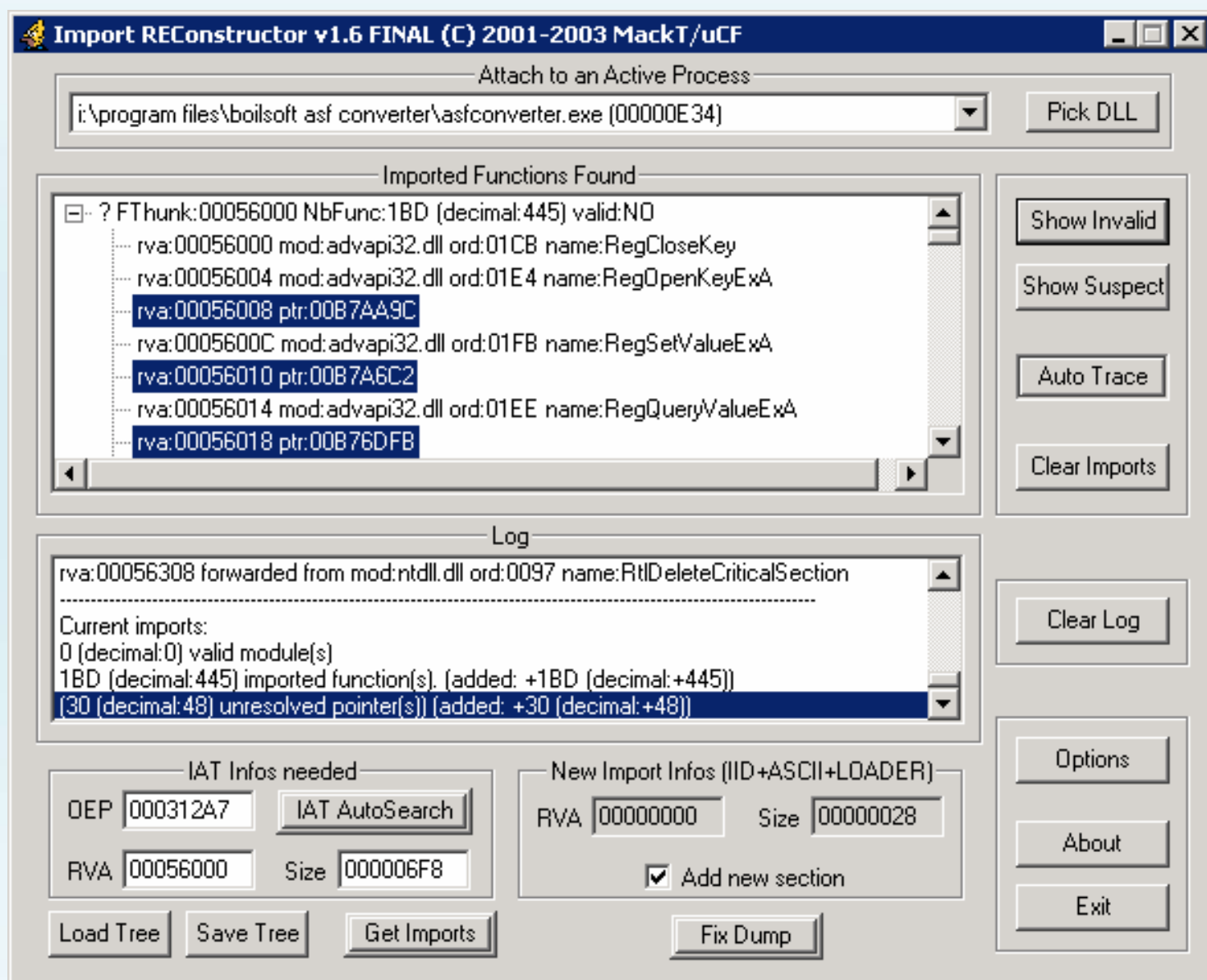
DaylalenhnhayveOEP (dungconhamvoicallecxt you i00B9030Dnha). Tadattbpo ago, F9, and F7. O, we

then select the OEP in:

The screenshot shows a debugger window titled "[CPU - main thread, module ASFConve]". The assembly list on the left shows instructions starting at address 004312A7. The instruction at 004312D5 is highlighted in blue: `CALL DWORD PTR DS:[456354]`. The registers window on the right shows the EIP register at 004312A7. The command window at the bottom shows "bp CreateThread" and "BP address, string - Break with condition".

Address	Hex dump	ASCII
004D2000	A6 6C F1 77 89 8D F1 77	01tw0#tw
004D2008	DE 82 F1 77 55 CE F1 77	01tw0Utw
004D2010	D7 8D F1 77 38 6A F1 77	01tw;jtw
004D2018	C0 6D F1 77 A0 59 F1 77	01tw0Vtw
004D2020	10 5F F1 77 00 5F F1 77	01tw;tw

Go to LordPE, then import process ASFConverter, and profile the imported.exe. Then, I get the import table, load process ASFConverter, go to 0x4312A7 (0x4312A7-0x312A7 = 0x400000) and "IAT AutoSearch", "Get Import table. Click "Show Invalid":



Then, I find that the import table is not correct. I find that the import table is not correct.

(MUP tut thanx mcuabac hacnho A r). Im Nhintren portREC, tat ayRVAcuaIATla0x56000 h -> V Acuanola: 0x400000 + 0x56000 = 0x456000. ObatdaucuaIAT A aylavungn h, is the AA oi m m l r h oadcachamt be vienvaghid i achivao.Tacandatbponwrite m e modaydechanlucnobatdauthuc does because I created the E AT. Resta tOlly & r o l a strange ad, clickvaovungdu m p, Ctrl + G, enter456000.Rightclick, than the breakpoint-> hardware, onwrite-> italy DWORD.F9decha: L a iOk, Shi t f + F9, Shi t f + F9. O, it was BP and then another, but the stars aio m m odule svcr -> F9 to run.

- [CPU - main thread, module msvcr]

File View Debug Plugins Options Window Help

LEMTW H C / K B R ... S

Address	Disassembly	Comment
77C46FA3	F3:AS	REP MOVSD DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
77C46FA5	FF2495 B870C477	JMP DWORD PTR DS:[EDX*4+77C470B8]
77C46FAC	8BC7	MOV EAX,EDI
77C46FAE	BA 03000000	MOV ECX,3
77C46FB3	83E9 04	SUB ECX,4
77C46FB6	72 0C	JB SHORT msvcr.77C46FC4
77C46FB8	83E0 03	AND EAX,3
77C46FB8	03C8	ADD ECX,EAX
77C46FBD	FF2485 D06FC477	JMP DWORD PTR DS:[EAX*4+77C46FD8]
77C46FC4	FF248D C870C477	JMP DWORD PTR DS:[ECX*4+77C470C8]
77C46FCB	90	NOP
77C46FCC	FF248D 4C70C477	JMP DWORD PTR DS:[ECX*4+77C4704C]
77C46FD3	90	NOP
77C46FD4	E0 6F	LOOPNE SHORT msvcr.77C47045
77C46FD6	C477 0C	LES ESI,FWORD PTR DS:[EDI+C]
77C46FD9	70 C4	J0 SHORT msvcr.77C46F9F
77C46FDB	77 30	JA SHORT msvcr.77C47000
77C46FDD	70 C4	J0 SHORT msvcr.77C46FA3
77C46FDF	77 23	JA SHORT msvcr.77C47004
77C46FE1	D18A 0688078A	ROR DWORD PTR DS:[EDX+8A078806],1
77C46FE7	46	INC ESI
77C46FEB	0188 47018A46	ADD DWORD PTR DS:[EAX+468A0147],ECX

Registers (FPU)

EAX	00078750
ECX	000047F0
EDX	00000000
EBX	00000000
ESP	00129298
EBP	001292A0
ESI	00066790
EDI	00456040 ASFConve.00456000
EIP	77C46FA3 msvcr.77C46FA3
C 0	ES 0023 32bit 0(FFFFF)
P 0	CS 001B 32bit 0(FFFFF)
A 1	SS 0023 32bit 0(FFFFF)
Z 0	DS 0023 32bit 0(FFFFF)
S 0	FS 003B 32bit 7FFDF000
T 0	GS 0000 NULL
D 0	
O 0	LastErr ERROR_SUCCESS
EFL	00210212 (NO,NB,NE,A,I)
ST0	empty -3.7266563491021
ST1	empty 0.000000000000000
ST2	empty -6.5928153957391
ST3	empty -UNGRH B7C0 0000
ST4	empty 5.65525129227322
ST5	empty -3.0041082223551

ECX=000047F0 (decimal 18416.)
DS:[ESI]=00066790=00067142
ES:[EDI]=00456040=00000000

Address	Hex dump	ASCII
00456000	72 6F 06 00 94 6F 06 00	not a text
00456008	A4 6F 06 00 B6 6F 06 00	not a text
00456010	C8 6F 06 00 80 6F 06 00	not a text
00456018	00 00 00 00 E4 70 06 00Zpt
00456020	CC 70 06 00 FD 70 06 00

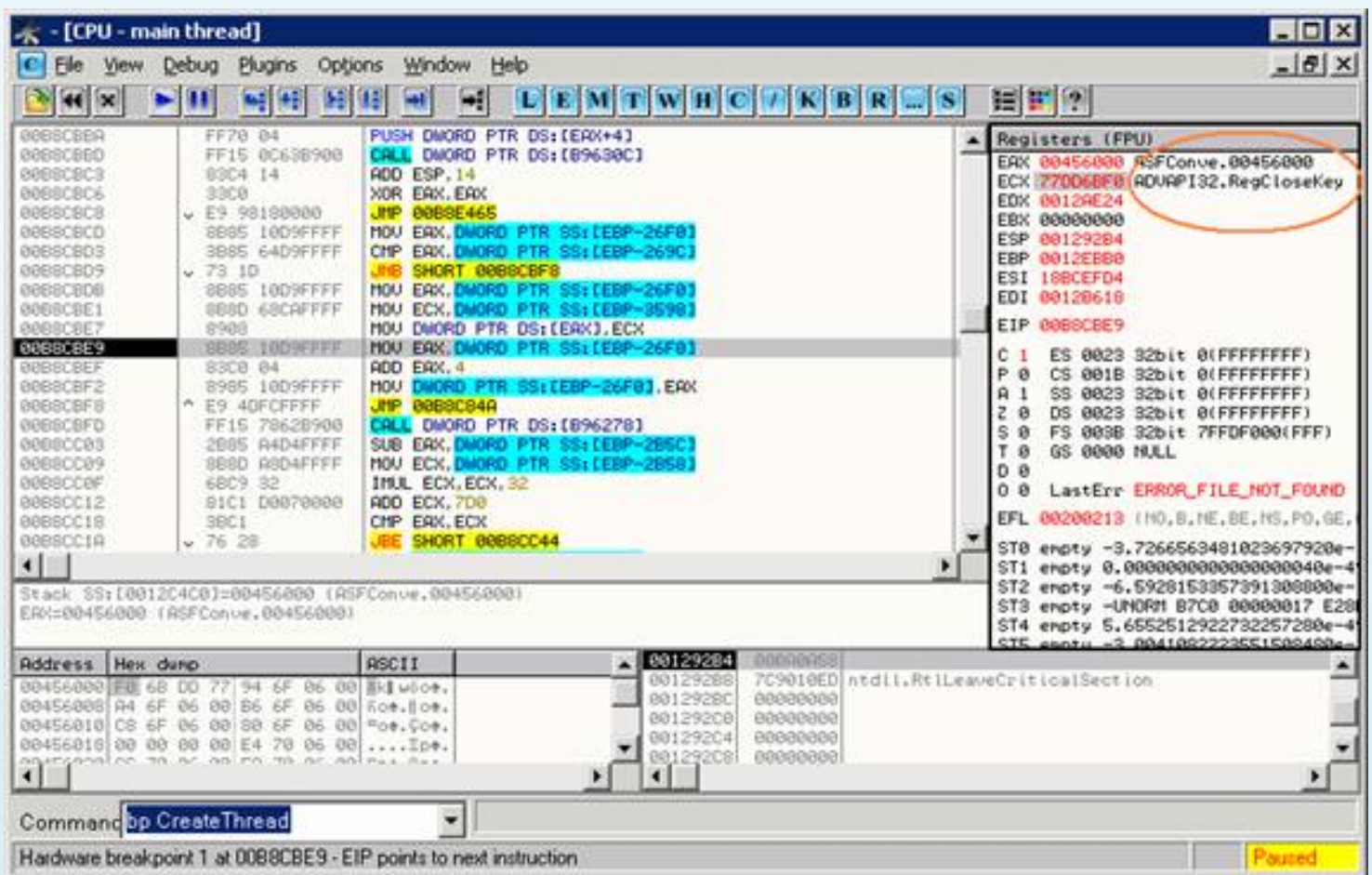
00129298 C9A2EDC3
0012929C 0012EA58
001292A0 0012EB00
001292A4 00B8B7C8 RETURN to 00B8B7C8 from 00B957F6
001292A8 00456000 ASFConve.00456000
001292AC 00066750

Command **bp CreateThread**

Hardware breakpoint 1 at msvcr.77C46FA3 - EIP points to next instruction

Paused

At BP h could lant two, we have a user noican the T-score:



To this, there are two ways to build ung laibang IAT. How the h be the box, you scroll up on a line, will see the command:


```
00B8CBE78908MOV  DWORD PTR DS:[EAX], ECX
```

Nhomsangcuasoregister, tat h ayecxchuadiachihamRegCloseKey, eax = Tuclacu 456,000 b ptai enhnay l, i timnhungentrynaoconth eutrong b angIAT adienvao m (using the BP o ndition chodo to the F9 hieu J) . Cachthuhaihocdu debit ctrongtutcuabachacno, PageUplen5 L-6 An timt h after aydauhieu :

The screenshot shows a debugger window titled "[CPU - main thread]". The assembly window displays the following instructions:

```

00B8C9DF 8B85 58C2FFFF MOV DWORD PTR SS:[EBP-3DA8],EAX
00B8C9E5 EB 0F JNP SHORT 00B8C9F6
00B8C9E7 8B85 58C2FFFF MOV EAX, DWORD PTR SS:[EBP-3DA8]
00B8C9ED 83C0 0C ADD EAX, 0C
00B8C9F0 8B85 58C2FFFF MOV DWORD PTR SS:[EBP-3DA8],EAX
00B8C9F6 8B85 58C2FFFF MOV EAX, DWORD PTR SS:[EBP-3DA8]
00B8C9FC 8378 08 00 CMP DWORD PTR DS:[EAX+8], 0
00B8CA00 74 49 JZ SHORT 00B8CA48
00B8CA02 68 00100000 PUSH 100
00B8CA07 8B85 58C1FFFF LEA EAX, DWORD PTR SS:[EBP-3EAB]
00B8CA0D 50 PUSH EAX
00B8CA0E 8B85 58C2FFFF MOV EAX, DWORD PTR SS:[EBP-3DA8]
00B8CA14 FF30 PUSH DWORD PTR DS:[EAX]
00B8CA16 E8 1F55F0FF CALL 00B61F3A
00B8CA18 83C4 0C ADD ESP, 0C
00B8CA1E 8B85 58C1FFFF LEA EAX, DWORD PTR SS:[EBP-3EAB]
00B8CA24 50 PUSH EAX
00B8CA26 8B85 68C2FFFF LEA EAX, DWORD PTR SS:[EBP-3D98]
00B8CA2B 50 PUSH EAX
00B8CA2C FF15 8B63B900 CALL DWORD PTR DS:[8B63B900]
00B8CA32 59 POP ECX
00B8CA33 59 POP ECX
  
```

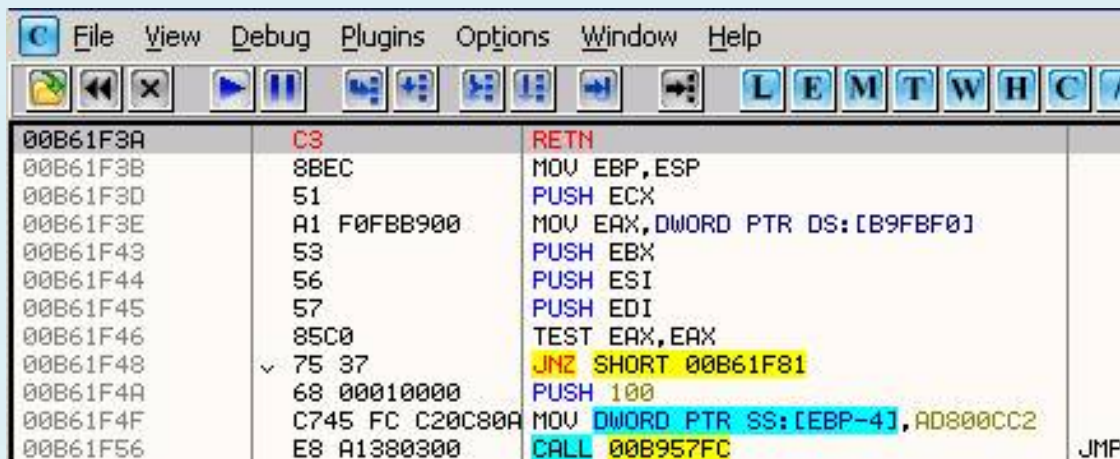
The registers window shows the following values:

```

Registers (FPU)
EAX 00456000 ASFC
ECX 77000000 AQUA
EDX 0012AE24
EBX 00000000
ESP 00129284
EBP 0012EBB0
ESI 188CEFD4
EDI 0012B618
EIP 00B8CBE9
  
```

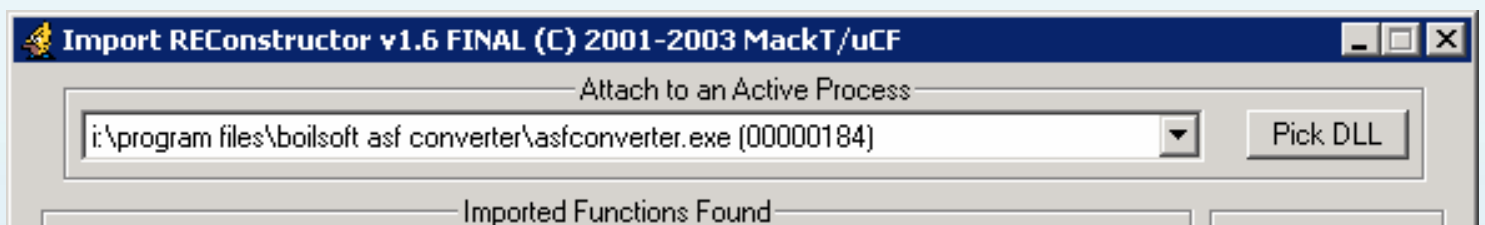
The command window at the bottom shows the command "bp CreateThread".

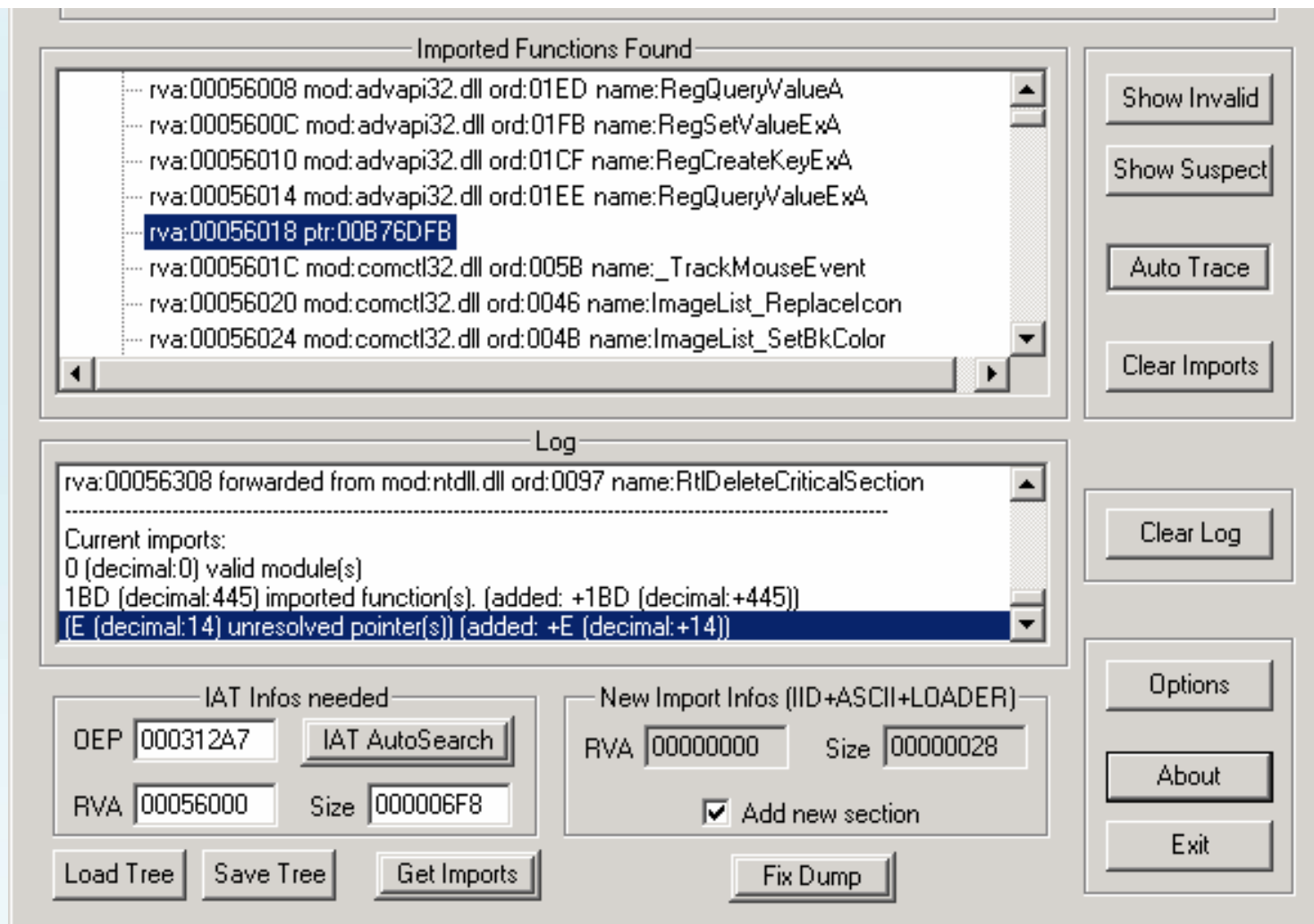
Tailenhcalkhoanh oten, tan h yd ol click i ac h ai l ila0x0B8CA16.Restart all catuda are set hw on write at 0x456000, F9 ... At times HW break head First, we or to e. A only a 0x0B8CA16 vadat hw on excecution. F9 dechay of the Ep. Khidapause, taentervahamcall, thaybytes55-> C3 (h o acanspace, type vaOk **ret**). Mucdichc ử aviectrenlavohieuhoa g oihamnay, ngaykhi and rove the uacallxongnose always, not if a.



Address	Hex	Instruction
00B61F3A	C3	RETN
00B61F3B	8BEC	MOV EBP,ESP
00B61F3D	51	PUSH ECX
00B61F3E	A1 F0FBB900	MOV EAX,DWORD PTR DS:[B9FBBF0]
00B61F43	53	PUSH EBX
00B61F44	56	PUSH ESI
00B61F45	57	PUSH EDI
00B61F46	85C0	TEST EAX,EAX
00B61F48	75 37	JNZ SHORT 00B61F81
00B61F4A	68 00010000	PUSH 100
00B61F4F	C745 FC C20C80A	MOV DWORD PTR SS:[EBP-4],AD800CC2
00B61F56	E8 A1380300	CALL 00B957FC

G at C & L earhetcachw bp.Nhaydend i achi0x4312A7 (OEP), set H Wonexcecution taidodedunglaitaiOEP.F9de the haytiep.Okie, nodadungroi.G olayI i m portRECra, the current huc italy are all b o C above, we đur C:



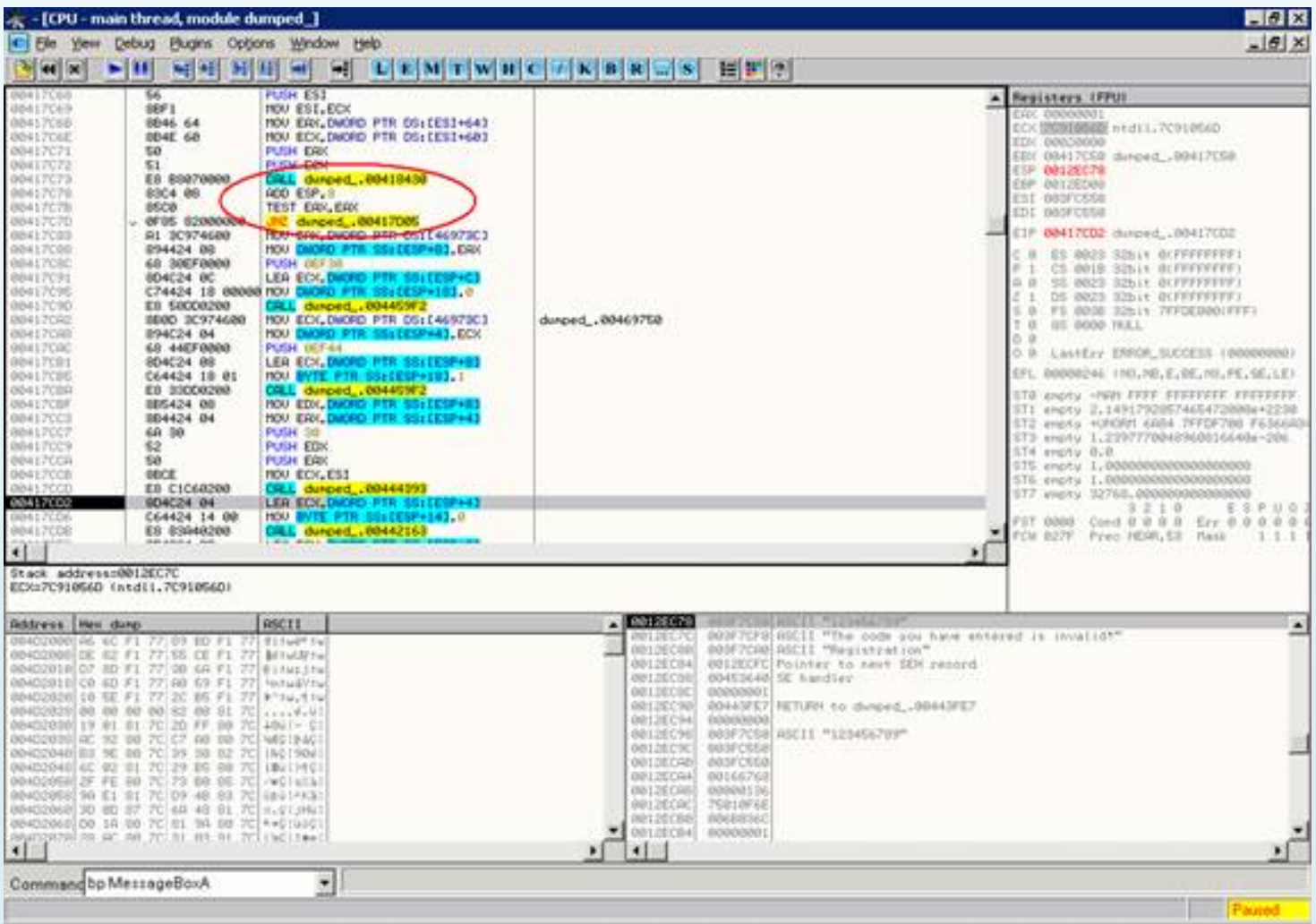


Well rùi, b i hours c u t If the NK then fix du m p. We đư C File du m ped_.exe. Run t h using it play. Ohlala, then success! N h like it up the nag this:



PART 2: Crack

Tiepthep antru h o C, P annaytat h h m uchiencrackfiled ped_.exe.LoadnovaoOlly, F9. At register, Enter: N a m e: REA; Code: 123456789. Back that in the l ai Olly, e place b p I run the MessageBoxA. Done back to reg, click O K. Again BP, veOlly. Ctrl + F9, to c h as Mr. Trinh click OK close m e s sagebox, veOlly edition F7. Trace live cra to 0x00417CD2



Chuyvungkhoanh, daylachocheckreg.Noreturneax = 1neuregok, conkhongfalse

vah i en m essageboxinvalid.Entervaoaham0x00418430. A occodecuano, click tanhan t h i v italy dautiennoloadthu E nArmAccess.dll, timhamInstallK yva g e d oidecheckco eNeu khongtimt h aythu and iện, who khongtimduochamInstallKeyhaycheck s, n ot POSTPAY l ai r eax = 0 . However, if the patch hamnay (at 0x418492):

```
00418490 33C0 XOR EAX, EAX
```

```
00418492 5B POP EBX
```

```
00418493          C3          RETN
```

```
00418494          90          NOP
```

```
00418495          90          NOP
```

replace with:

```
00418490          33C0          XOR EAX, EAX
```

```
00418492          40          INC EAX
```

```
00418493          5B          POP EBX
```

```

00418494 C3 RETN
00418495 90 NOP

```

Ketquavankhongon, vichikhicheckkeyno m h oiok.Conkhirestartc're not ngtrinhvan chuareg. Uclakhik T h oidong, noconco m othamcheck the ua.Oday, the aiham00418430, tacuon a whimper en l m otchutse find t h ayngayhamki to mtranay, a run that i0x4183C0.Hamn italy cungloadArmAcesss, nhunggoihamVerifyK y.Tathuchienpatchhamnaytaigiat e r i returns:

```

00418420          33C0          XOR EAX ,
                                EAX
00418422          5B          POP EBX
00418423 C3 RETN
00418424          90          NOP
00418425          90          NOP

```

Become :

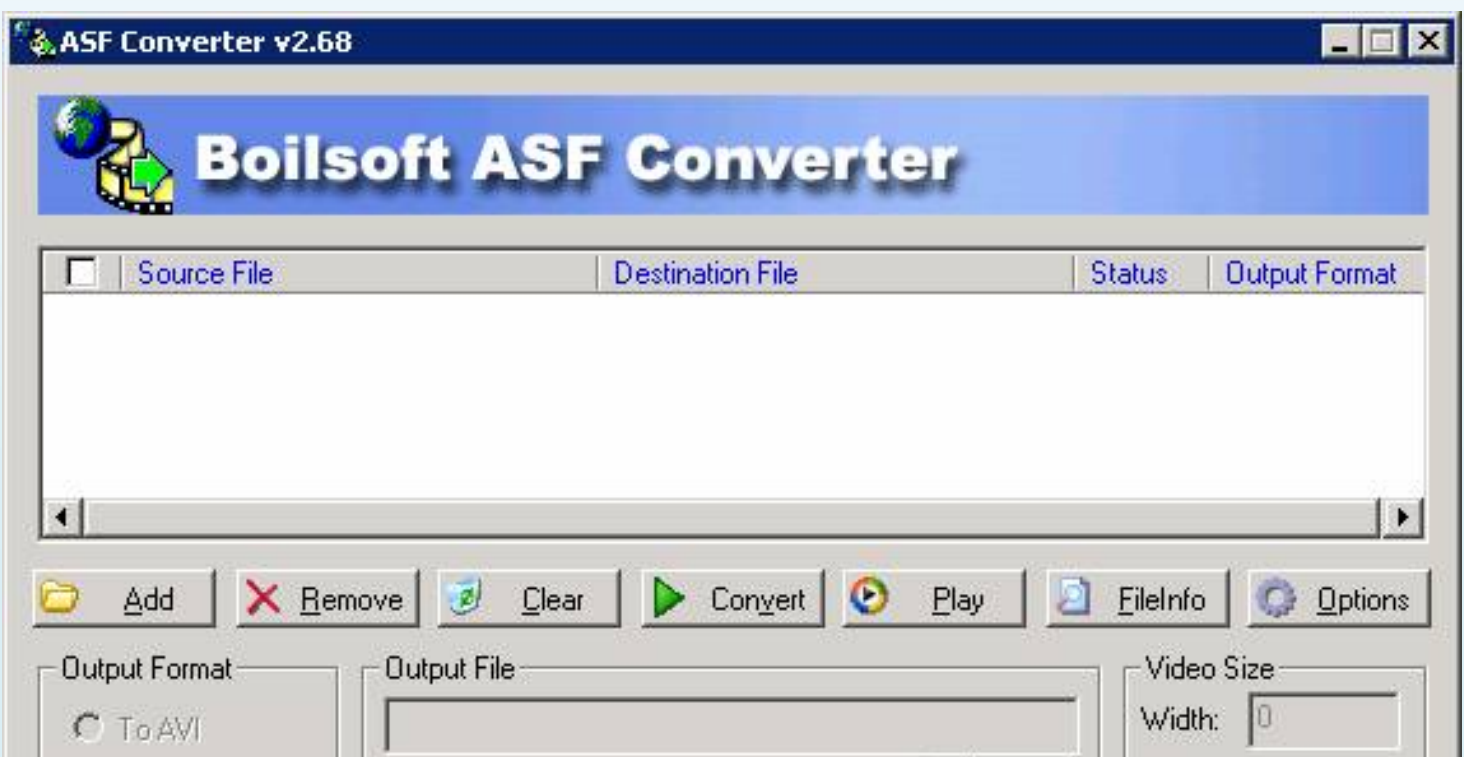
```

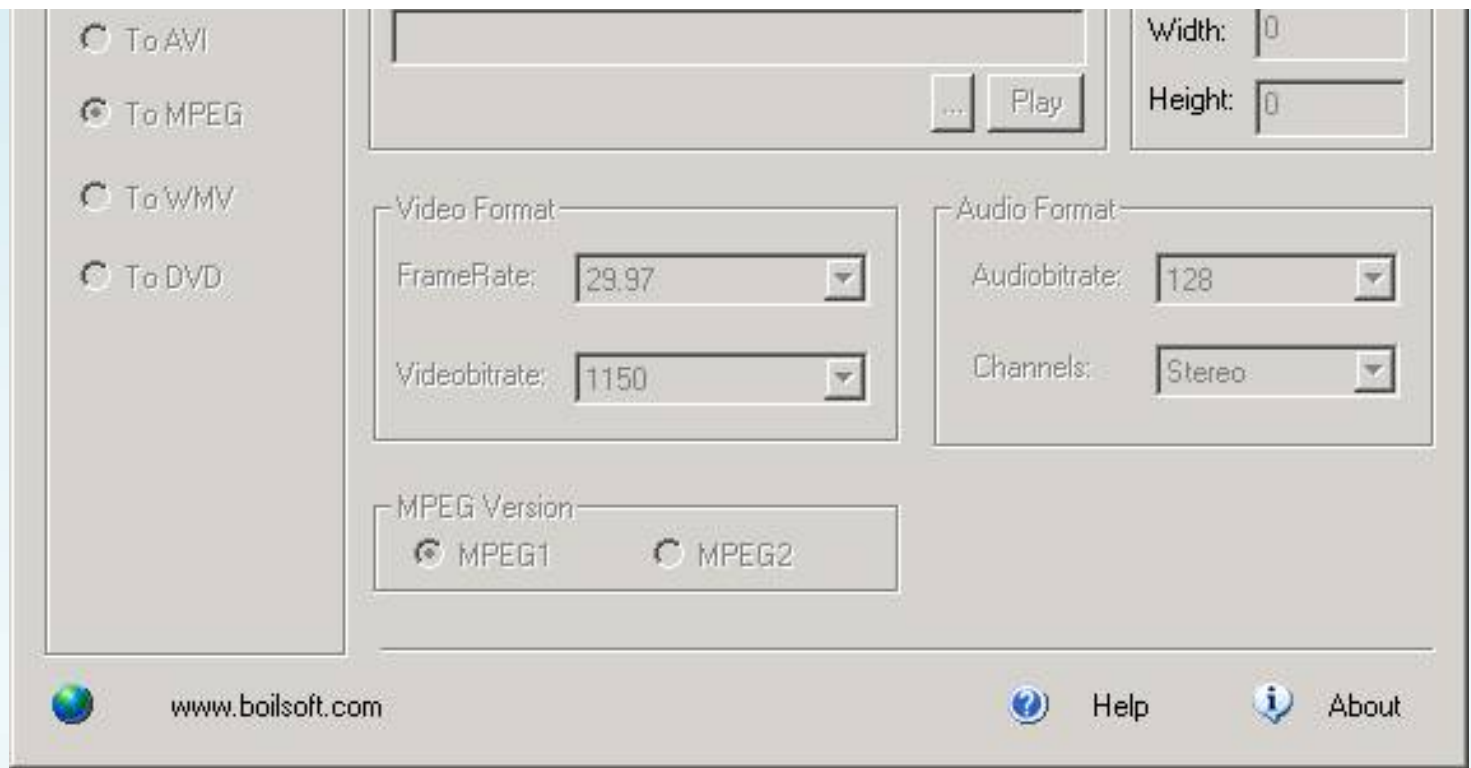
00418420          33C0          XOR EAX , EAX
00418422          40          INC EAX
00418423          5B          POP EBX
00418424 C3 RETN
00418425 90 NOP

```

Select area just patch (from 418,422 to 418,424), right click -> write to executal b e-> selection. Dongwindowvua m m o.Ollyseh ie nlen SG h oi, clickYes, save the file patch.exe.

OK, time to run the file patch.exe, we đư C:





He, daxong! G h i oneut interests, the cacbancudefilenaychaycu gdu debit c.Tuynhienneux the k m e h t h countries estimated that f ilenaytat h i s zec ử anoco1500KB L. Dayladodu m praphandu the promise vatrongdococaphancodecuaarmkhongcandungtoi.Tacothed fish cdoanodecua armdi to giamkich uo of the f ile:

PART 3: Reduce file size

Copy patch.exe to a more f ile, have called pact.exe m, m ofilenay table Editor of PE not LordPE, in the section, we have:

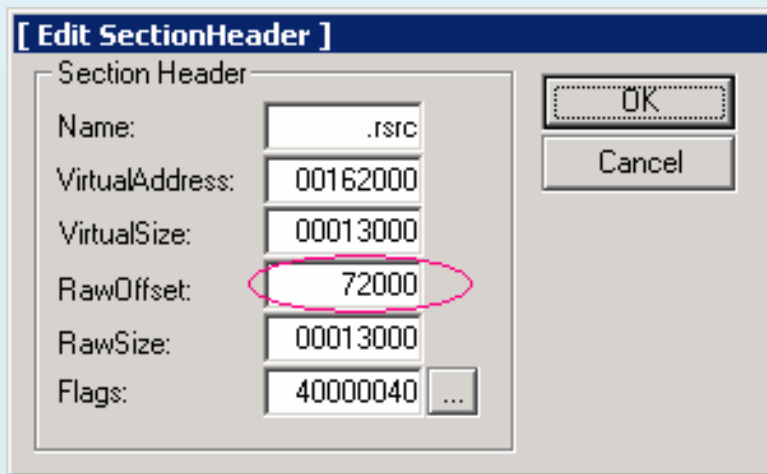
[Section Table]					
Name	VOffset	VSize	ROffset	RSize	Flags
.rdata	00056000	0001135E	00056000	0001135E	C0000040
.data	00068000	00009C68	00068000	00009C68	C0000040
.text1	00072000	00050000	00072000	00050000	E0000020
.adata	000C2000	00010000	000C2000	00010000	E0000020
.data1	000D2000	00020000	000D2000	00020000	C0000040
.pdata	000F2000	00070000	000F2000	00070000	C0000040
.rsrc	00162000	00013000	00162000	00013000	40000040
.mackt	00175000	00002000	00175000	00002000	E0000060

Cacsection.text1. ADATA. Data1. PdatalacuaA r m, cothetodi.Tuynhien, xoakhong dungcachselamcrashfileexedantoikhongc h aydu debit c.Tase huchientungbu mind cnhu the following:

- 1) Record POSTPAY all the information section of the (default if they do not have to t h m in the f ile patch.exera to xemlai if necessary).
 - 2) Comments italy 2 section .rsc (Resource CPC file) & .Ackt m (section add to I m your portRECdefixIAT). XemROffsetcua.rsc (laoffsettrenfileexe), C tadu 0x162000. By ROffset. Text is 0x72000.
- 3) Open Hexworkshop the (hoachexeditornaytuythich), goto0x162000, f ile selectden end and cut more than
- 4) Tiepdengotodenoffset0x72000, selectdenc the ôi f ile h ondelete and evil (whether aylax that a code share your ar m)
- 5) Now paste the datada cut in step 3 at the location 0x72000 and then save the file ái.

X e mlaiki the hthu mind cfileco pact.exe m: J nocon540KB. Tuynhieniconcua ilebi f m ráo at all, the

file also bicrash always. : Ddung worry, we will re-ua s PE Header t h ai l ì run it delicious. DungPEEditorc ù aLordPE, m in the hansection, **xaahetcacsectionheader. Text1. ADATA. Data1. Pdata** go (rightclick, than the "wipesectionheader"). Audoeitsectionheadercua S. Rsc as follows:



Vitadadoidu ieucua.rsrcve0x72000nentacanch the only part of hlaiRawOffsetcuano.Cac to other resources.

're More of you. Ackt m:

GiatriRawOffsetbandaucua. Cktla0x175000.Giat r m a I. Acktb m a ndaucua.rsrcla 0x162000. Dotadichuyencahaikhoi iennhau l, t r chonenvi ITU ơ n g gdoi iuachung not change. Khối. Rsrc 0x72000 in the en-> block. Ackt m will have a new RawOffset: $0x175000 - 0x162000 + 0x72000 = 0x85000$

[Edit SectionHeader]

Section Header

Name: .mact

VirtualAddress: 00175000

VirtualSize: 00002000

RawOffset: 00085000

RawSize: 00002000

Flags: E0000060 ...

OK

Cancel

Tiepdien, simply isolate nhlai now chthu of mind cvirtual uasection.data, dovungn than the ua.datava. rsrc not enough o C continuously. We have: VOffset available data is 0x68000, VSize is 0x9C68-> đ A uoi the only area of memory. Data is $0x68000 + 0x9C68 = 0x71C68$ -> rounding is 0x72000 (because Windows Management the the page size mco m e 4K = 0x1000). Tacanch only to the hlaiVSizeaocho i achicu number ù a.datatrungvoidiac ic h idaucua. Rsrc is 0x162000, so the need to fix is: $0x162000 - 0x68000 = 0xFA000$. N so damaged, the section of the same ôi as follows:

[Section Table]					
Name	VOffset	VSize	ROffset	RSize	Flags
text	00001000	00054757	00001000	00054757	60000020
.rdata	00056000	0001135E	00056000	0001135E	C0000040
.data	00068000	000FA000	00068000	00009C68	C0000040
.rsrc	00162000	00013000	00072000	00013000	40000040
.mact	00175000	00002000	00085000	00002000	E0000060

Save the file you have i.Chay file pact.exe, Okie, it has a number of activities, the treatment with k t h o C as a 540K J If there is the press table aspack ai, h is the size C is the remaining 231K!. To the end is guided MUP and crack a SFConverter v2.68!

Thanx to:

Hacnho, Deux, has puter_Angel, Z o m bie, Moonbaby, RCA, kien m anowar, benina, TQN, the_lighthouse, Nini, hoadongnoi, ... and you!

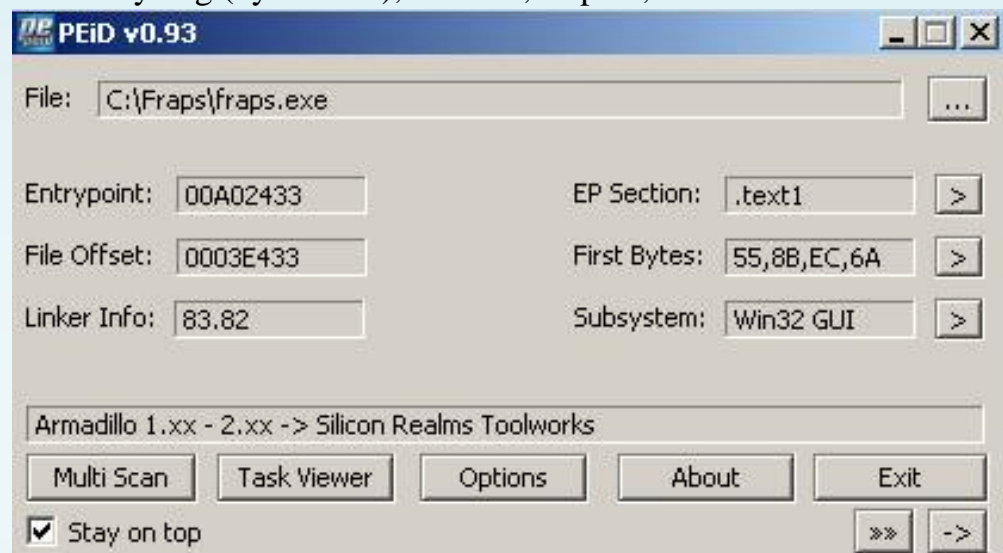
Written by LightPhoenix (date: Sept., 5th 2005)

MUP_Armadillo_Fraps_Code_Splicing_ + _IAT_Elimination tlandn

Target: Fraps (included in tut)

Packer: Armadillo

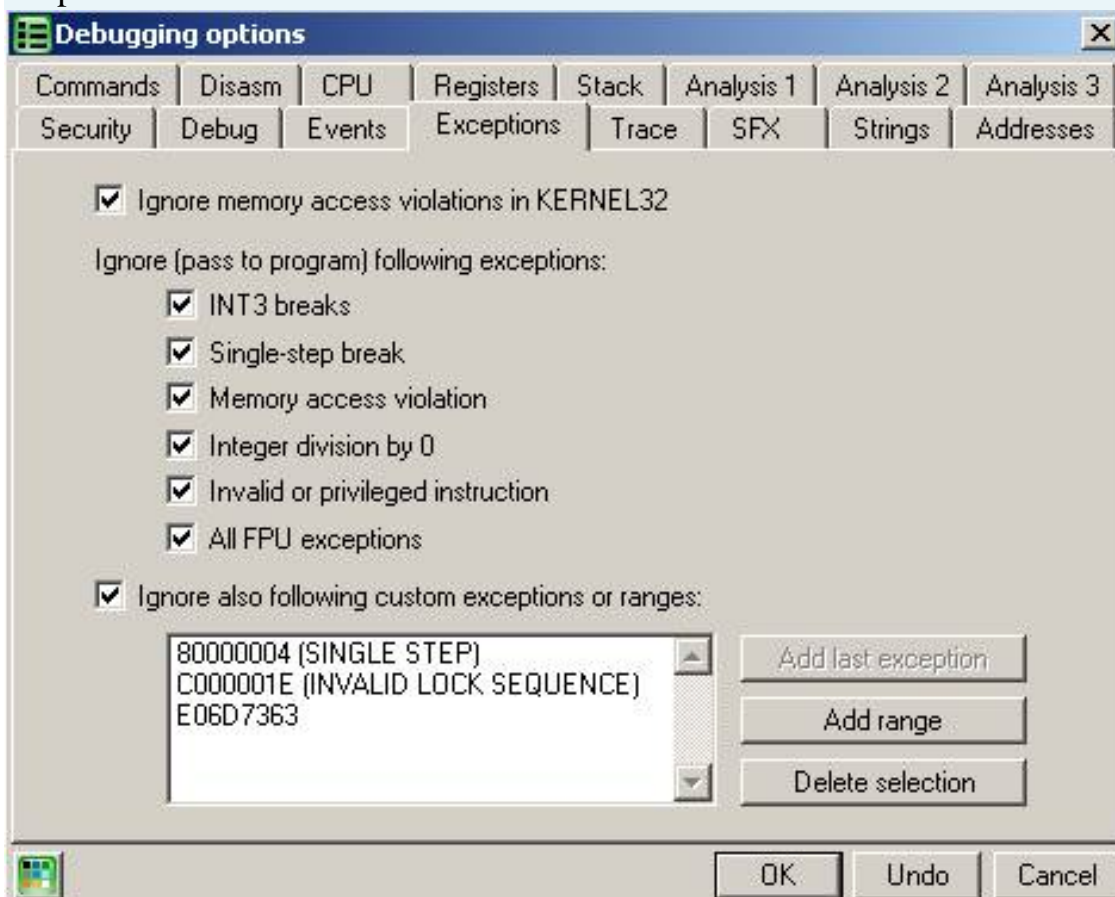
Tools: OllyDbg (by hacnho), LordPE, Imprec, ArmInline 0.4



First we check the pack with what is. Using PEID:

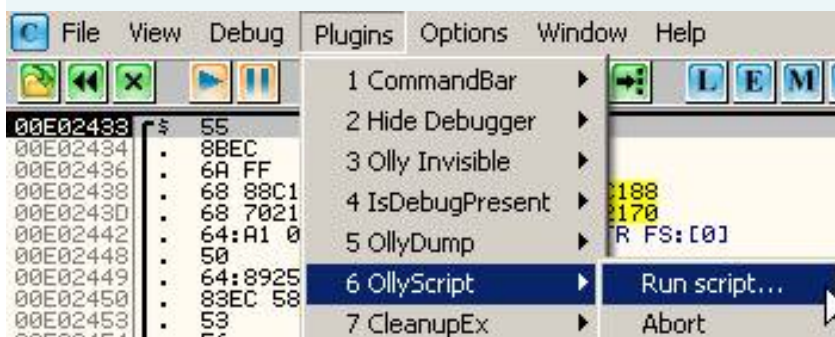
We see the pack in the Armadillo.

Using OllyDbg (hacnho by using a very good J) fraps.exe open the file. Press Alt-O. Edit the number as in the picture:



First we will jump to find magic patch. The goal is to make our IAT will complete.

For the fast, I will use the script by hacnho for magic jump (included in the tut). Using OllyScript plugin:

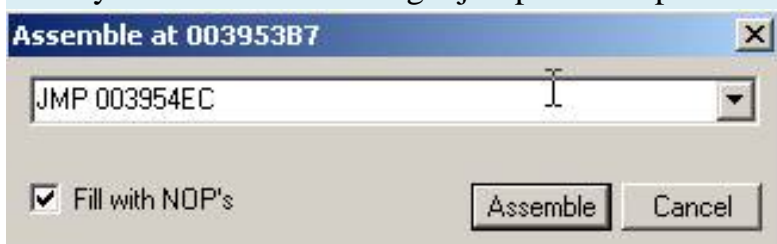


After plugin finished running. We will be here:

00395386	8B0D 24CF3B00	MOV ECX,DWORD PTR DS:[3BCF24]	
0039538C	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
0039538F	A1 24CF3B00	MOV EAX,DWORD PTR DS:[3BCF24]	
00395394	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
00395397	75 16	JNZ SHORT 003953AF	
00395399	8085 DCFEFFFF	LEA EAX,DWORD PTR SS:[EBP-124]	
0039539F	50	PUSH EAX	
003953A0	FF15 90E03A00	CALL DWORD PTR DS:[3AE090]	kernel32.LoadLibraryA
003953A6	8B0D 24CF3B00	MOV ECX,DWORD PTR DS:[3BCF24]	
003953AC	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003953AF	A1 24CF3B00	MOV EAX,DWORD PTR DS:[3BCF24]	
003953B4	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
003953B7	0F84 2F010000	JE 003954EC	
003953BD	33C9	XOR ECX,ECX	
003953BF	8B03	MOV EAX,DWORD PTR DS:[EBX]	
003953C1	3938	CMP DWORD PTR DS:[EAX],EDI	
003953C3	74 06	JE SHORT 003953CB	
003953C5	41	INC ECX	
003953C6	83C0 0C	ADD EAX,0C	
003953C9	EB F6	JMP SHORT 003953C1	

You note of Red's jump on the magic of us. We will revise the JMP 003954EC.

Click your cursor on the magic jump. Press Space. Enter JMP 003954EC:



Click Assemble. We are:

00395386	8B0D 24CF3B00	MOV ECX,DWORD PTR DS:[3BCF24]	
0039538C	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
0039538F	A1 24CF3B00	MOV EAX,DWORD PTR DS:[3BCF24]	
00395394	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
00395397	75 16	JNZ SHORT 003953AF	
00395399	8085 DCFEFFFF	LEA EAX,DWORD PTR SS:[EBP-124]	
0039539F	50	PUSH EAX	
003953A0	FF15 90E03A00	CALL DWORD PTR DS:[3AE090]	kernel32.LoadLibraryA
003953A6	8B0D 24CF3B00	MOV ECX,DWORD PTR DS:[3BCF24]	
003953AC	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003953AF	A1 24CF3B00	MOV EAX,DWORD PTR DS:[3BCF24]	
003953B4	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
003953B7	E9 30010000	JMP 003954EC	
003953BC	90	NOP	
003953BD	33C9	XOR ECX,ECX	
003953BF	8B03	MOV EAX,DWORD PTR DS:[EBX]	
003953C1	3938	CMP DWORD PTR DS:[EAX],EDI	
003953C3	74 06	JE SHORT 003953CB	
003953C5	41	INC ECX	
003953C6	83C0 0C	ADD EAX,0C	
003953C9	EB F6	JMP SHORT 003953C1	

So once the Magic Jump.

Press Alt-M to window "Memory Map". Set in Breakpoint section. Text:

00380000	0005C000				Priv	Rw	Rw	
003E0000	00004000				Priv	Rw	Rw	
003F0000	00010000				Priv	Rw	Rw	
00400000	00001000	fraps		PE header	Image	R	RwE	
00401000	00012000	fraps	.text					
00413000	00002000	fraps	.rdata					
00415000	009B0000	fraps	.data					
00DC5000	00050000	fraps	.text1	code				
00E15000	00010000	fraps	.adata					
00E25000	00020000	fraps	.data1	data, impo				
00E45000	00060000	fraps	.pdata					
00EA5000	0015F000	fraps	.rsrc	resources				
01010000	00005000							
01000000	00002000							
010E0000	00103000							
011F0000	000AE000							
014F0000	0000A000							
015F0000	0000C000							
01600000	00003000							

Actualize

Dump in CPU

Dump

Search

Ctrl+B

Set break-on-access

F2

Set memory breakpoint on access

Press Shift-F9. We here:

003A97A0	8B0C3A	MOV ECX,DWORD PTR DS:[EDX+EDI]	
003A97B0	5B	POP EBX	
003A97B1	03D7	ADD EDX,EDI	
003A97B3	A1 5C8F3B00	MOV EAX,DWORD PTR DS:[3B8F5C]	
003A97B8	3148 30	XOR DWORD PTR DS:[EAX+30],ECX	
003A97BB	A1 5C8F3B00	MOV EAX,DWORD PTR DS:[3B8F5C]	
003A97C0	3148 30	XOR DWORD PTR DS:[EAX+30],ECX	
003A97C3	A1 5C8F3B00	MOV EAX,DWORD PTR DS:[3B8F5C]	
003A97C8	8B16	MOV EDX,DWORD PTR DS:[ESI]	
003A97CA	8B48 64	MOV ECX,DWORD PTR DS:[EAX+64]	
003A97CD	3348 5C	XOR ECX,DWORD PTR DS:[EAX+5C]	
003A97D0	3348 24	XOR ECX,DWORD PTR DS:[EAX+24]	
003A97D3	030D 748F3B00	ADD ECX,DWORD PTR DS:[3B8F74]	
003A97D9	85D2	TEST EDX,EDX	
003A97DB	75 17	JNZ SHORT 003A97F4	
003A97DD	8B50 5C	MOV EDX,DWORD PTR DS:[EAX+5C]	
003A97E0	FF76 18	PUSH DWORD PTR DS:[ESI+18]	
003A97E3	3350 4C	XOR EDX,DWORD PTR DS:[EAX+4C]	
003A97E6	FF76 14	PUSH DWORD PTR DS:[ESI+14]	
003A97E9	3310	XOR EDX,DWORD PTR DS:[EAX]	
003A97EB	FF76 10	PUSH DWORD PTR DS:[ESI+10]	
003A97EE	2BCA	SUB ECX,EDX	
003A97F0	FFD1	CALL ECX	
003A97F2	EB 1C	JMP SHORT 003A9810	
003A97F4	83FA 01	CMP EDX,1	
003A97F7	75 1A	JNZ SHORT 003A9813	
003A97F9	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
003A97FC	8B50 5C	MOV EDX,DWORD PTR DS:[EAX+5C]	
003A97FF	3350 4C	XOR EDX,DWORD PTR DS:[EAX+4C]	
003A9802	FF76 08	PUSH DWORD PTR DS:[ESI+8]	
003A9805	3310	XOR EDX,DWORD PTR DS:[EAX]	
003A9807	6A 00	PUSH 0	
003A9809	FF76 0C	PUSH DWORD PTR DS:[ESI+C]	
003A980C	2BCA	SUB ECX,EDX	
003A980E	FFD1	CALL ECX	
003A9810	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
003A9813	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
003A9816	5F	POP EDI	
003A9817	5E	POP ESI	
003A9818	C9	LEAVE	
003A9819	C3	RETN	

fraps.00400000

You note of red in the image above. Set a breakpoint in it (double-click the mouse on the line). Press F9. We in the line:

003A97E9	3310	XOR EDX,DWORD PTR DS:[EAX]	
003A97EB	FF76 10	PUSH DWORD PTR DS:[ESI+10]	
003A97EE	2BCA	SUB ECX,EDX	
003A97F0	FFD1	CALL ECX	
003A97F2	EB 1C	JMP SHORT 003A9810	
003A97F4	83FA 01	CMP EDX,1	
003A97F7	75 1A	JNZ SHORT 003A9813	
003A97F9	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
003A97FC	8B50 5C	MOV EDX,DWORD PTR DS:[EAX+5C]	
003A97FF	3350 4C	XOR EDX,DWORD PTR DS:[EAX+4C]	
003A9802	FF76 08	PUSH DWORD PTR DS:[ESI+8]	
003A9805	3310	XOR EDX,DWORD PTR DS:[EAX]	
003A9807	6A 00	PUSH 0	
003A9809	FF76 0C	PUSH DWORD PTR DS:[ESI+C]	
003A980C	2BCA	SUB ECX,EDX	
003A980E	FFD1	CALL ECX	
003A9810	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
003A9813	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
003A9816	5F	POP EDI	
003A9817	5E	POP ESI	
003A9818	C9	LEAVE	
003A9819	C3	RETN	

fraps.0040C434

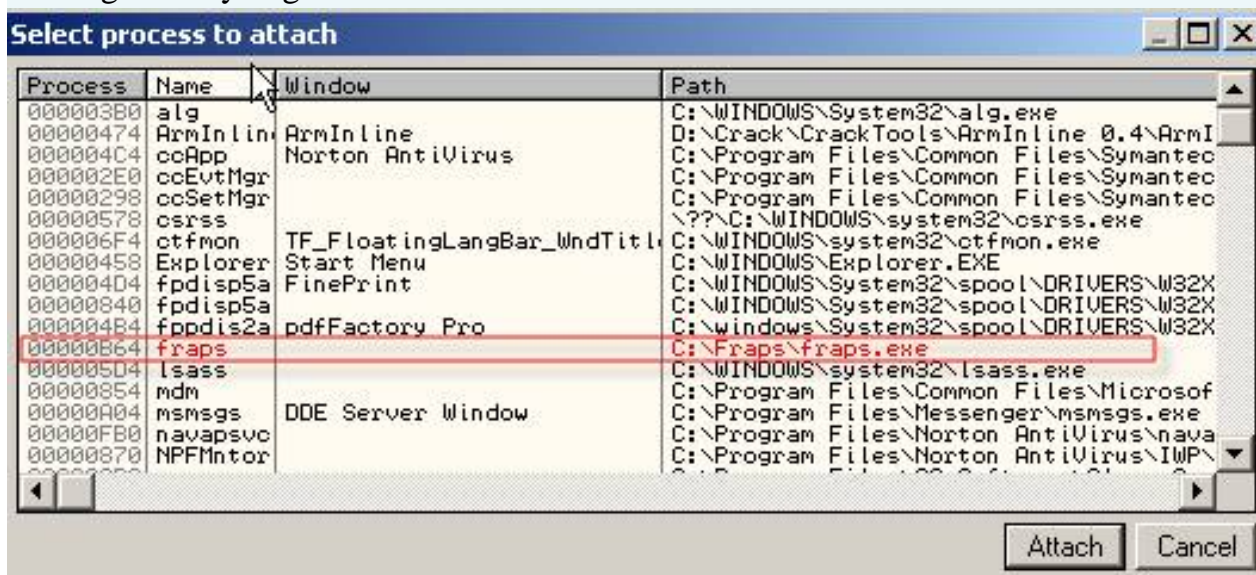
Press F7. We at J OEP

0040C434	55	PUSH EBP	
0040C435	8BEC	MOV EBP,ESP	
0040C437	6A FF	PUSH -1	
0040C439	68 28334100	PUSH fraps.00413328	
0040C43E	68 30E94000	PUSH fraps.0040E930	
0040C443	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
0040C449	50	PUSH EAX	
0040C44A	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
0040C451	83EC 58	SUB ESP,58	
0040C454	53	PUSH EBX	
0040C455	56	PUSH ESI	
0040C456	57	PUSH EDI	
0040C457	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0040C45A	FF15 744B6F01	CALL DWORD PTR DS:[16F4B74]	kernel32.GetVersion

OEP: 0040C434

The next step we will fix Code Splicing and IAT elimination.

The first is the Code Splicing. Booting ArmInline 0.4. We need to find a process by PID Fraps we are running. In OllyDbg select File -> Attach. Make a red line:



Here I'm on B64 (your computer may be different). I B64 to enter the "Process ID" in ArmInline.

In OllyDbg click Cancel to close the window Attach. Press Alt-M. We will find the area of memory that are Splicing Code. Very easy. Section usually form 0xxx0000 and the last section under Fraps. Image to more easily understand:

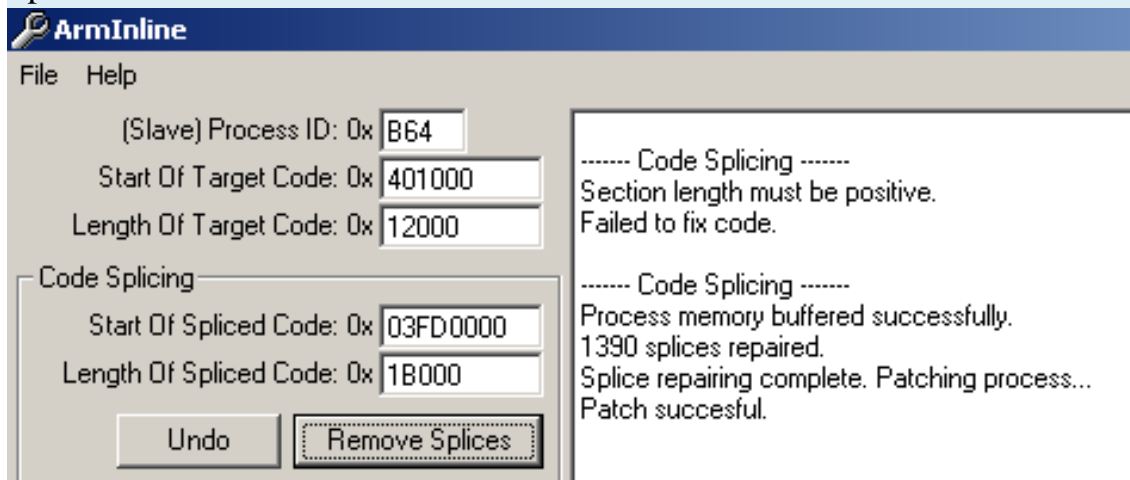
00400000	00001000	fraps			Imag	R	RWE
00401000	00012000	fraps	.text		Imag	R	RWE
00413000	00002000	fraps	.rdata		Imag	R	RWE
00415000	009B0000	fraps	.data		Imag	R	RWE
00DC5000	00050000	fraps	.text1	code	Imag	R	RWE
00E15000	00010000	fraps	.adata		Imag	R	RWE
00E25000	00020000	fraps	.data1	data,import	Imag	R	RWE
00E45000	00060000	fraps	.pdata		Imag	R	RWE
00EA5000	0015F000	fraps	.rsrc	resources	Imag	R	RWE
01010000	00005000				Map	R E	R E
010D0000	00002000				Map	R E	R E
010E0000	00103000				Map	R	R
011F0000	000B3000				Map	R E	R E
014F0000	0000A000				Priv	RW	RW
015F0000	0000C000				Priv	RW	RW
01600000	00002000				Map	R	R
01610000	0001D000				Priv	RW	RW
01630000	000AF000				Priv	RW	RW
016F3000	00002000				Priv	RW	RW
01710000	00006000				Priv	RW	RW
01720000	00001000				Map	RW	RW
01730000	00001000				Map	RW	RW
01740000	00039000				Priv	RW	RW
01940000	00001000				Priv	RW	RW
01950000	00004000				Priv	RW	RW
01960000	00010000				Priv	RW	RW
01D60000	00003000				Priv	RW	RW
01DF0000	00002000				Map	R	R
01E00000	00001000				Priv	RW	RW
01F7E000	00002000				Priv	RW	RW
03FD0000	0001B000				Priv	R E	RWE
5B0A0000	00001000	umdmxfrm	PE header		Imag	R	RWE

Here I'm on the computer section and 03FD0000 length is 1B000. Enter the number on the item in the Code

Splicing ArmInline. Also, we must find the section. Text and length of this section:

00400000	00001000	fraps			Image	R	RWE	
00401000	00012000	fraps	.text		Image	R	RWE	
00413000	00002000	fraps	.rdata		Image	R	RWE	
00415000	009B0000	fraps	.data		Image	R	RWE	
00DC5000	00050000	fraps	.text1	code	Image	R	RWE	
00E15000	00010000	fraps	.adata		Image	R	RWE	
00E25000	00020000	fraps	.data1	data, import	Image	R	RWE	
00E45000	00060000	fraps	.pdata		Image	R	RWE	
00EA5000	0015F000	fraps	.rsrc	resources	Image	R	RWE	

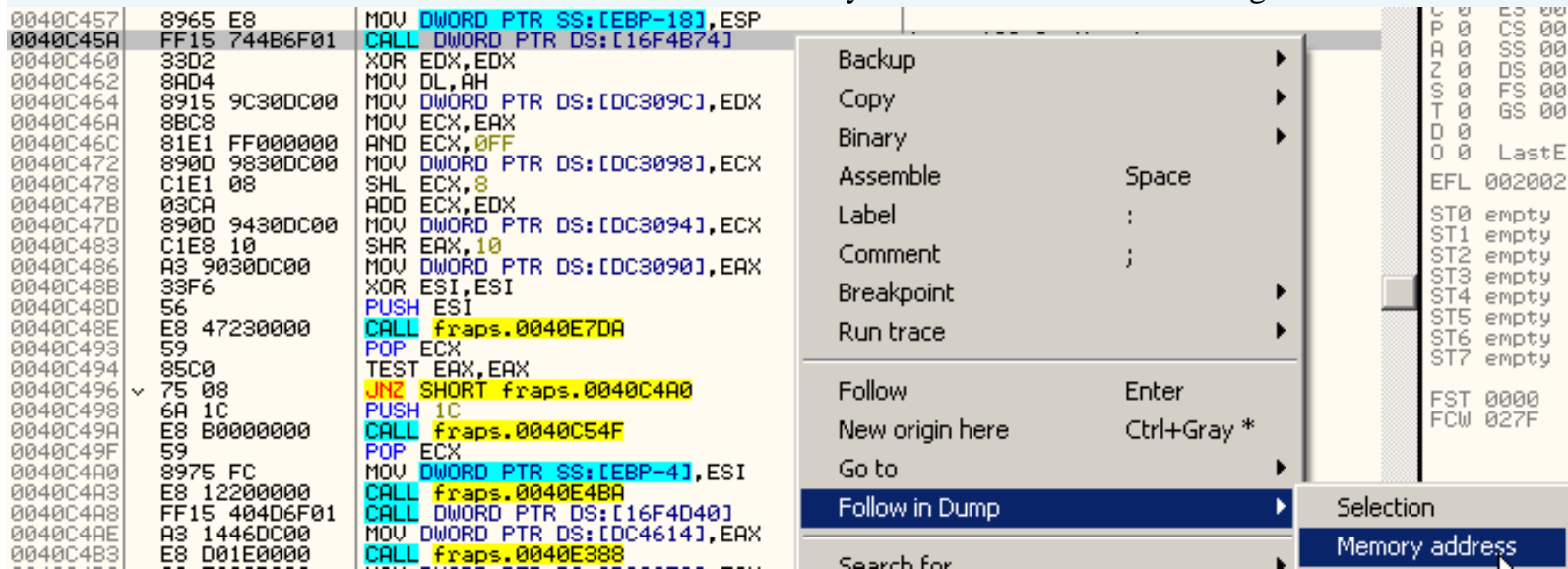
We have a section. Text: 401,000, length of 12,000. Enter the number on the ArmInline, click "Remove Splices":



Done Code Splicing. Now to IAT elimination. Close window "Memory Map" again. We are in OEP:

0040C434	55	PUSH EBP	
0040C435	8BEC	MOV EBP,ESP	
0040C437	6A FF	PUSH -1	
0040C439	68 28334100	PUSH fraps.00413328	
0040C43E	68 30E94000	PUSH fraps.0040E930	
0040C443	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
0040C449	50	PUSH EAX	
0040C44A	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
0040C451	83EC 58	SUB ESP,58	
0040C454	53	PUSH EBX	
0040C455	56	PUSH ESI	
0040C456	57	PUSH EDI	
0040C457	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0040C45A	FF15 744B6F01	CALL DWORD PTR DS:[16F4B74]	kernel32.GetVersion

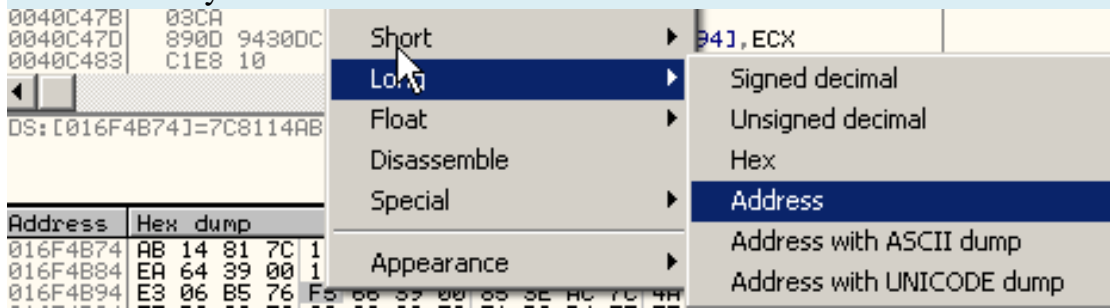
You note in 0040C45A we call GetVersion function. Click your mouse to select the image as:



In the window dump:

Address	Hex dump	ASCII
016F4B74	AB 14 81 7C 16 4F D9 77 7A 66 39 00 72 65 39 00	%Tui!_Opwzf9.re9.
016F4B84	EA 64 39 00 19 99 80 7C C3 65 39 00 26 66 39 00	nd9.+0Cite9.&f9.
016F4B94	E3 06 B5 76 F5 66 39 00 85 3E AC 7C 4A BC D4 77	π41vjf9.ā>%iJ ēw
016F4BA4	57 B3 80 7C 80 38 82 7C 51 D0 D4 77 FE 65 39 00	W Ç Ç8ē!Q^ ēwme9.

To more easily visible. You must click the mouse to the window and select the dump in the picture:



We are:

Address	Hex dump	Comment
016F4B74	AB 14 81 7C 1	kernel32.GetVersion
016F4B78	77D94F16	USER32.TrackPopupMenu
016F4B7C	0039667A	
016F4B80	00396572	
016F4B84	003964EA	
016F4B88	7C809919	kernel32.GetCurrentThread
016F4B8C	003965C3	
016F4B90	00396626	
016F4B94	76B506E3	WINMM.mixerGetLineControlsA
016F4B98	003966F5	
016F4B9C	7CAC3E85	SHELL32.SHBrowseForFolderA
016F4BA0	77D4BC4A	USER32.MsgWaitForMultipleObjectsEx
016F4BA4	7C80B357	kernel32.GetModuleFileNameA

Here, that it must not over. In the window dump pulled up to the starting point IAT, we have:

016F4934	00000000	
016F4938	0003009D	
016F493C	01080192	
016F4940	003966F5	
016F4944	77D521AE	USER32.LoadIconA
016F4948	00396520	
016F494C	7C809943	kernel32.GetACP
016F4950	76B5C3FD	WINMM.waveInGetPosition
016F4954	77D4C5B8	USER32.ScreenToClient
016F4958	77D4CEFD	USER32.PeekMessageA
016F495C	7C811110	kernel32.HeapDestroy
016F4960	00396605	
016F4964	7C80EB3F	kernel32.CreateMutexA

The starting point is: 16F4944 (gray line, LoadIconA address on your computer may be different).

In the window dump drag down to the end points of the IAT, we have:

016F4DF8	77DD7883	ADVAPI32.RegQueryValueExA
016F4DFC	0039651E	
016F4E00	7C80ACB2	kernel32.IsProcessorFeaturePresent
016F4E04	003965E6	
016F4E08	77D4C55A	USER32.GetSubMenu
016F4E0C	00396624	
016F4E10	003965BA	
016F4E14	00396672	
016F4E18	0039666C	
016F4E1C	7C810DA6	kernel32.SetFilePointer
016F4E20	009D0004	fraps.009D0004

End points: 16F4E1C (gray line, SetFilePointer).

In summary:

IAT

The starting point: 16F4944

End points: 16F4E1C

Length: 16F4E1C - 16F4944 = 4D8

Enter the number on the ArmInline (just starting point and length).

We also need a number where more ArmInline to move to the IAT, we will select that section. ADATA. In

OllyDbg press Alt-M

00400000	00001000	fraps		
00401000	00012000	fraps	.text	
00413000	00002000	fraps	.rdata	
00415000	009B0000	fraps	.data	
00DC5000	00050000	fraps	.text1	code
00E15000	00010000	fraps	.adata	
00E25000	00020000	fraps	.datal	data, import
00E45000	00060000	fraps	.pdata	
00EA5000	0015F000	fraps	.rsrc	resources

Section. ADATA (gray line): 00E15000. Enter parameters to ArmInline as following, click Rebase IAT:

ArmInline
File Help

(Slave) Process ID: 0x

Start Of Target Code: 0x

Length Of Target Code: 0x

Code Splicing

Start Of Spliced Code: 0x

Length Of Spliced Code: 0x

Import Elimination

Base Of Existing IAT: 0x

Length Of Existing IAT: 0x

New Base RVA Of IAT: 0x

Nanomites

Master Process ID: 0x

----- Code Splicing -----
Section length must be positive.
Failed to fix code.

----- Code Splicing -----
Process memory buffered successfully.
1390 splices repaired.
Splice repairing complete. Patching process...
Patch succesful.

----- Rebasing IAT -----
IAT length must be a positive multiple of 4.
Failed to rebase IAT.

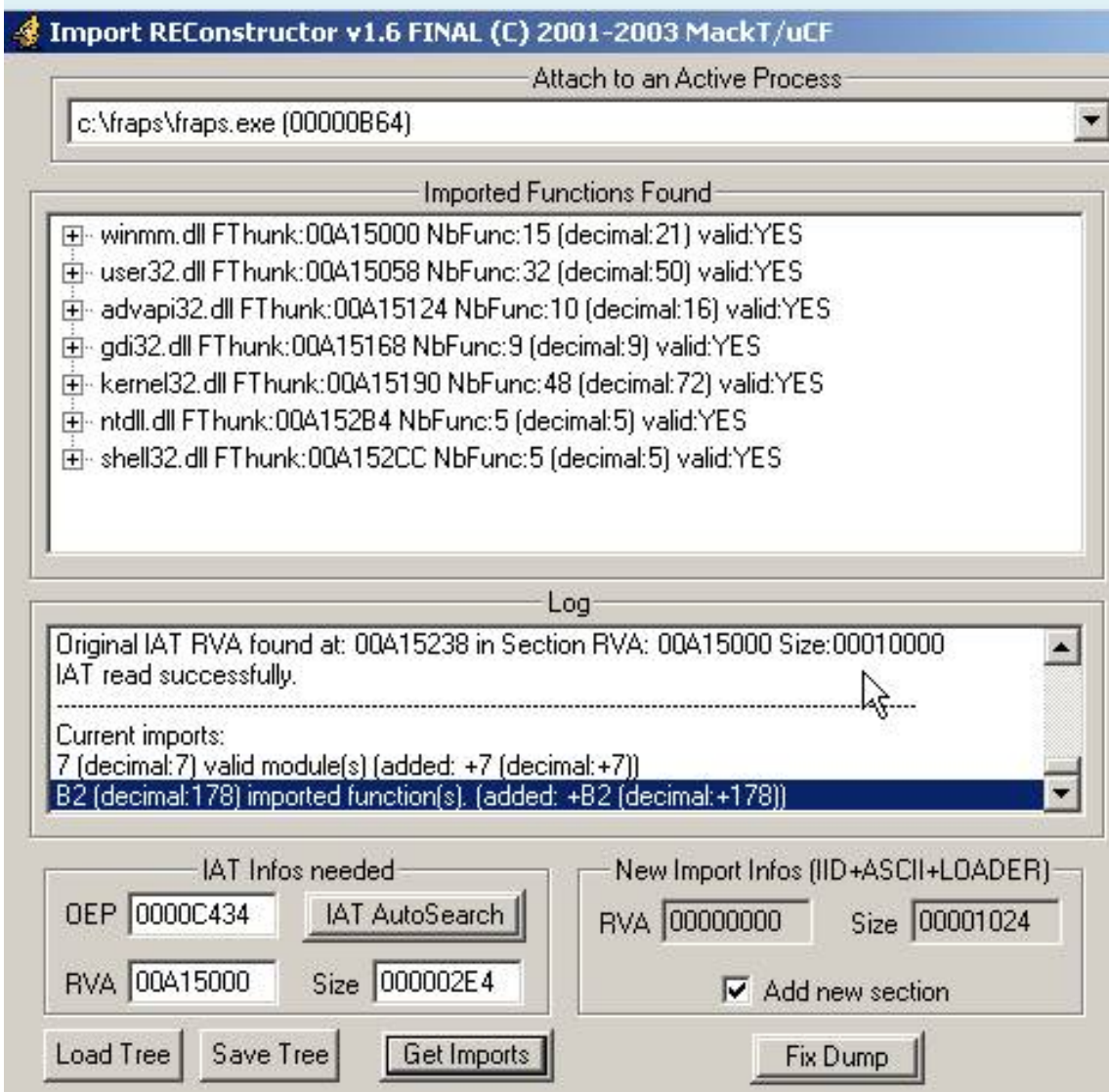
----- Rebasing IAT -----
Process memory buffered successfully.
592 DLL calls found total.
Analysing...
178 API functions referenced from 7 DLLs.
Redirecting DLL references:
592 calls redirected total.
Patching process...
Process succesfully patched.

Now we will use to LordPE dump file. Open LordPE, dump:

[LordPE Deluxe] by yoda

Path	PID	ImageBase
c:\fraps\fraps.exe	dump full...	00400000
d:\crack\cracktools\...	dump partial...	00400000
c:\windows\system32\...	dump region...	01000000
c:\program files\mess...	active dump engine	01000000
d:\crack\cracktools\...	priority	00400000
Path		ImageSize

Name the file is dump.exe. Open Imprec, fill OEP: C434. Click "IAT AutoSearch," OK, "Get Import":



Click "Fix dump" file dump.exe choose. We are dump_.exe file. Test J



Have Fun!

tlandn

Greetz: All VCT memberz, and you ... J

2008

[MUP Armadillo v4.64 Small Case]



www.reasonline.net

kienmanowar

www.reaonline.net

6/19/2008

Contents

[I.](#)

[II. The sudung](#)

[III. Unpack Armadillo v4.64](#)

[1st find information about Target.](#)

[2. CopyMemII extravagant and DebugBlocker](#)

[3. Nanomites](#)

[IV.](#)

I. Introduction

Welcome brother, is the second article in my post about how to unpack Armadillo. This is because the reeve tuts4you, they have found on Armadillo v4.64 which is the right target which I replied in the previous version 4.62 arm. Down to try to try and found it still used as the Protect first, but after mup complete the run is not as soft for too. After you ask the brothers to their new memory forgotten one Add to that the promise to be tricky should not be disappointed. This article is nothing but the subliminal hope it provides some useful information for the brothers! For the brother and the higher the firm is also pleased to read J

II. The use Tools

Posts using the following:

1. Ollylce (ver **2008.1.1**): **This was introduced in the REA, include it available to plug can help brothers bypass the mechanism of anti-debug Armadillo**
2. Armadillo Find Protected V1.8
3. ArmaDetach v1.31
4. ArmInline v0.96f (Eng)
5. Task LordPE or Explorer.
6. ImportRec v1.7c Final.
7. CFF Explorer.

III. Manual unpack Armadillo v4.64

1. Seeking information on Target.

thumb.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00319210	00001000	00000000	00000000	00000000	00000000	0000	0000	60000020
.itext	00001908	0031B000	00000000	00000000	00000000	00000000	0000	0000	60000020
.data	0001FFAC	0031D000	00000000	00000000	00000000	00000000	0000	0000	C0000040
.bss	00016850	0033D000	00000000	00000000	00000000	00000000	0000	0000	C0000000
.idata	00004890	00334000	00000000	00000000	00000000	00000000	0000	0000	C0000040
.edata	000002DC	00333000	00000000	00001000	00000000	00000000	0000	0000	40000040
.tls	0000016C	00333000	00000000	00000000	00000000	00000000	0000	0000	C0000000
.rdata	00000018	00333000	00001000	00002C00	00000000	00000000	0000	0000	40000040
.reloc	00028108	00333000	00000000	00000000	00000000	00000000	0000	0000	42000040
.text1	00050000	00335000	00043000	00004000	00000000	00000000	0000	0000	E0000020
.adata	00010000	003D5000	0000D000	00047000	00000000	00000000	0000	0000	E0000020
.data1	00020000	003E5000	0000A000	00054000	00000000	00000000	0000	0000	C0000040
.pdata	00230000	00405000	00230000	0005E000	00000000	00000000	0000	0000	C0000040
.rsrc	000EC000	00635000	00026000	0028E000	00000000	00000000	0000	0000	40000040

Next we need to check to see this program pack in as many Armadillo version and protect the mechanism is applied to it to the power of that deal. Open Armadillo Find Protect v1.8, then drag & drop target to be one of the following information:

17-06-2008 14:19:01 <----->

C: \ Program Files \ stg \ thumb \ thumb.exe

- Protected Armadillo

Protection system (Professional)

- <Protection Options>

Debug-Blocker

CopyMem-II

Enable elimination Import Table

Enable Strategic Code Splicing

Enable Nanomites Processing

- Key <Backup Options>

Variable Backup Keys

- <Compression Options>

Best / Slowest Compression

- <Other Options>

- Version 4.64 22January2007

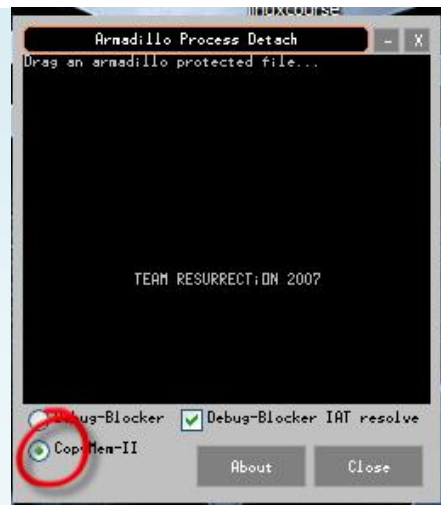
- Elapsed Time 00h 00m 02s 093ms

Ui chà enough of his face from Debug-blocker to the floor last Nanomites, so here are tired. Ok this is the collection of information is finished, we carried out any work.

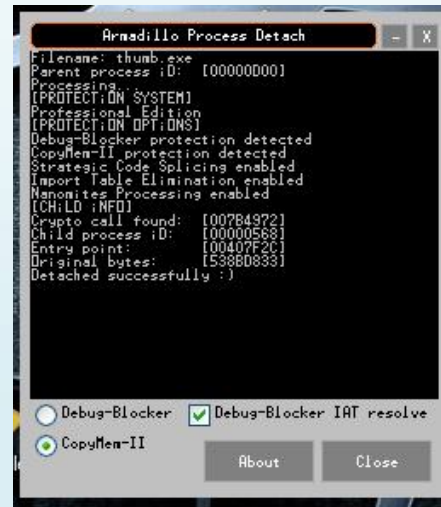
2. Through face CopyMemII and DebugBlocker

If his children have read messages before the MUP Armadillo 5:42 will see if we play the manual is very tired and oài. So in this Tutor I applied the tools available to reduce the work load already have the order before.

First load the program and select ArmaDetach v1.31 as follows:



Then drag & drop target by us to:



[CHILD Info]

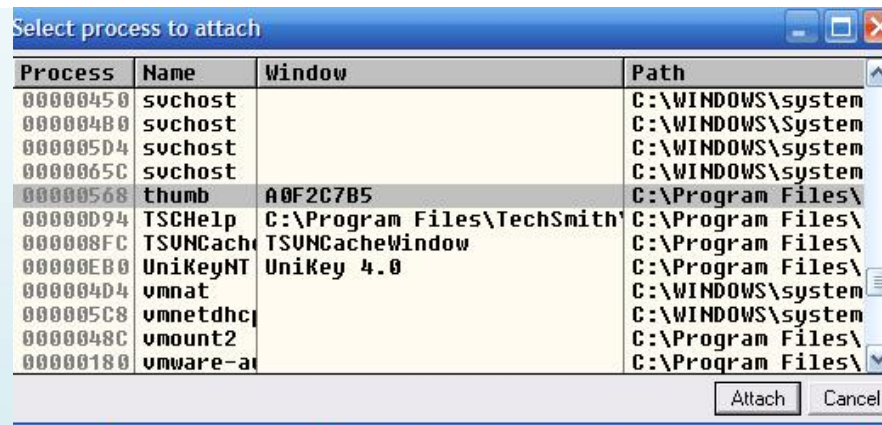
Crypto call found: [007B4972]

Child process ID: [00000568]

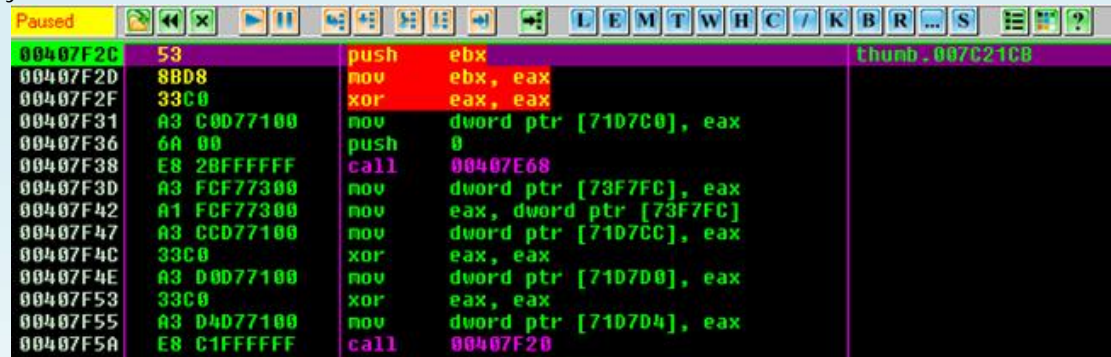
Entry point: [00407F2C] to EP

Original bytes: [538BD833]

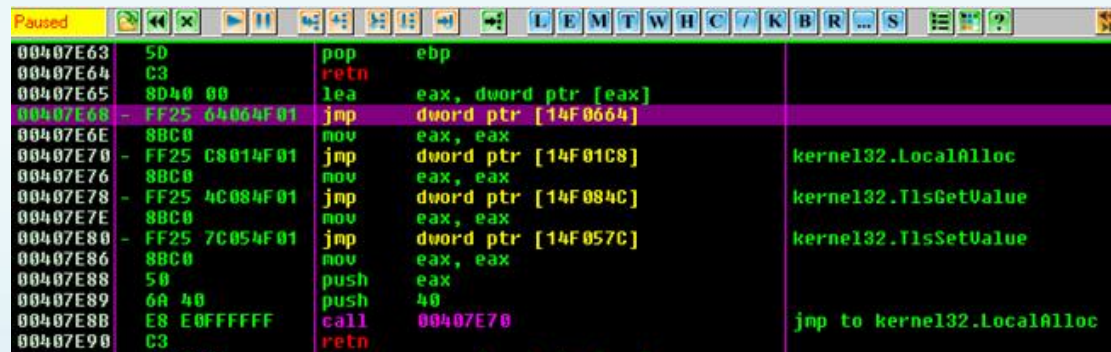
After implementation of the above is officially over we face **Debug-blocker** and then **CopyMem2**. Now the load and Olly Attach child process to:



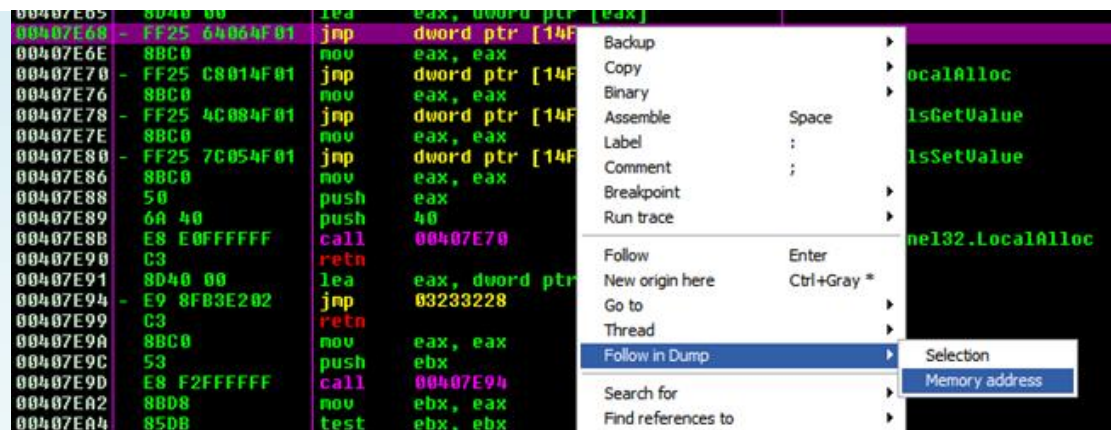
F9 and F12, and then edit the EP in bytes:



Khúc them and defeat CopyMem2 DebugBlocker very gently but do not have to waste any dew J. Khuc next restore the IAT to target, we must first locate the **IAT Start**, **End IAT** and the **IAT** has **Size**. You have to notice under orders Call EP not: 00407F38 E8 2BFFFFFF call 00407E68. Select it and press Enter to follow, to me here:



Ola something there, no doubt what this is more a call to one of the functions of the API, that function is what we will not know. One of the things have, and mouse to select the following:



At window dump scroll mouse over we have been IAT start:

014F0040	000801A5	
014F0044	02080103	
014F0048	77F44308	GDI32.Arc ← IAT Start
014F004C	01327BF1	
014F0050	7E456002	USER32.LoadKeyboardLayoutA
014F0054	7E41EF3D	USER32.DestroyIcon
014F0058	77C018BA	VERSION.VerQueryValueA
014F005C	77C019FF	VERSION.GetFileVersionInfoSizeA
014F0060	77F17C11	GDI32.GetViewportOrgEx
014F0064	77F17D01	GDI32.GetViewportExtEx
014F0068	7E41C236	USER32.DestroyCaret
014F006C	7E43147A	USER32.GetMenu
014F0070	7E42F2B3	USER32.ShowScrollBar
014F0074	77C01A50	VERSION.GetFileVersionInfoA

Scroll down we have IAT End:

014F0D38	01327B97	
014F0D3C	77F1BBDC	GDI32.TextOutA
014F0D40	01327CDA	
014F0D44	77F161FF	GDI32.CreateBitmap
014F0D48	77F1FE86	GDI32.DeleteEnhMetaFile
014F0D4C	01327CE2	
014F0D50	01327C06	
014F0D54	01327CEE	
014F0D58	01327BD5	
014F0D5C	7E41CB85	USER32.PostMessageA ← IAT End

Summarized the end we have:

014F0048 77F44308 GDI32.Arc <== IAT Start

014F004C 01317BF1

014F0050 7E456002 USER32.LoadKeyboardLayoutA

014F0054 7E41EF3D USER32.DestroyIcon

... ..

... ..

014F0D5C 7E41CB85 USER32.PostMessageA

014F0D60 00000000

014F0D64 00000000 <== End IAT (2 back down position to ensure ArmInline find the right)

IAT IAT Size = End - Start = IAT 0xD1C

With information such as the time when we are looking for **Full IAT**. Open the ArmaDetach v1.31

but this time only select Debug-blocker and Debug-Blocker IAT resolve it. Then drop into the target. Next use a new Olly to Attach child proc, fix the bytes.

007D5000	60	pushad
007D5001	E8 00000000	call 007D5006
007D5006	5D	pop ebp
007D5007	50	push eax
007D5008	51	push ecx
007D5009	0FCA	bswap edx
007D500B	F7D2	not edx
007D500D	9C	pushfd
007D500E	F7D2	not edx
007D5010	0FCA	bswap edx
007D5012	EB 0F	jmp short 007D5009

After recover bytes in the EP, set in a hwbp GetModuleHandleA (our purpose is to find and fix

Magic Jump-for versions arm use under 5 magic jump to cancel our IAT). Once in place hwbp GetModuleHandleA, press Shift + F9 to implement the program until you see signs in the window, the Stack stop:

0013920C	0135676D	CALL to GetModuleHandleA from 01356767
00139210	0013935C	pModule = "kernel32.dll"
00139214	00000000	
00139218	EBEC0000	
0013921C	A8390013	
00139220	0137C1EC	

Then press Ctrl + F9 to us here:

01356788	E8 C40BFFFF	call 01340354	
01356790	84C0	test al, al	
01356792	0F84 53010000	je 013568E8	
01356798	0D05 B4FEFFFF	lea eax, dword ptr [ebp-14C]	
0135679E	50	push eax	
0135679F	FF15 E4713701	call dword ptr [13771E4]	kernel32.LoadLibraryA
013567A5	8B00 AC553801	mov ecx, dword ptr [13855AC]	
013567AB	89040E	mov dword ptr [esi+ecx], eax	
013567AE	A1 AC553801	mov eax, dword ptr [13855AC]	
013567B3	391C06	cmp dword ptr [esi+eax], ebx	
013567B6	E9 30010000	jmp 013568E8	Magic jump
013567BB	0033	add byte ptr [ebx], dh	
013567BD	C9	leave	
013567BE	8B07	mov eax, dword ptr [edi]	
013567C0	3918	cmp dword ptr [eax], ebx	
013567C2	74 06	je short 013567CA	
013567C4	41	inc ecx	
013567C5	83C0 0C	add eax, 0C	
013567C8	EB F6	jmp short 013567C0	

According to the pictures we see on the Magic Jump has always jmp is so then we have to fix, and from the ability of the IAT program have not been canceled J. Next is to find OEP by placing in a hwbp CreateThread (remember to remove hwbp in GetModuleHandleA farewell).

Note: If your still want to try any way to find and fix 100 Push is set in a hwbp

VirtualProtect, then Shift + F9 to implement the program until found signs similar to the window at the stop Stack again:

0013947C	0136FA75	CALL to VirtualProtect from 0136FA73
00139480	00400000	Address = thumb.00400000
00139484	00000040	Size = 40 (64.)
00139488	00000004	NewProtect = PAGE_READWRITE
0013948C	001394A8	pOldProtect = 001394A8
00139490	076174FB	
00139494	0013B654	
00139498	A8390013	
0013949C	00000328	

Then press Ctrl + F9 to turn back to main code, all in the Search command is push 100. Next select the first month, follow it and fix the following:

```

013426D5 59          pop     ecx
013426D6 C3          ret
013426D7 B9 00FF3701 mov     ecx, 137FF00
013426DC E9 DA3D0100 jmp     0135648B
013426E1 C3          ret
013426E2 8BEC       mov     ebp, esp
013426E4 51          push    ecx
013426E5 A1 30FF3701 mov     eax, dword ptr [137FF30]
013426EA 53          push    ebx
013426EB 56          push    esi
013426EC 57          push    edi
013426ED 85C0       test     eax, eax
013426EF 75 37      jnz     short 01342728
013426F1 68 00010000 push    100
013426F6 C745 FC 878F09 mov     dword ptr [ebp-4], 04098F87
013426FD E8 36420300 call    01376938
01342702 8DB0 00010000 lea     esi, dword ptr [eax+100]
01342708 50          pop     eax

```

After the fix is on leave as hwpb VirtualProtect go in, put in a hwp CreateThread purpose we are going to find OEP original program.

Implementation of similar Tutor Armadillo v.5.42 to us here:

```

0071C818 55          push    ebp
0071C819 8BEC       mov     ebp, esp
0071C81B 83C4 F0    add     esp, -10
0071C81E 53          push    ebx
0071C81F B8 18957100 mov     eax, 00719518
0071C824 E8 03B7CEFF call    00407F2C
0071C829 8B1D C4CC7300 mov     ebx, dword ptr [73CCC4]
0071C82F 8B03       mov     eax, dword ptr [ebx]
0071C831 E8 52A60BFF call    00406E88
0071C836 8B03       mov     eax, dword ptr [ebx]
0071C838 B2 01      mov     dl, 1
0071C83A E8 21C50BFF call    00408D60
0071C83F 8B03       mov     eax, dword ptr [ebx]
0071C841 BA E4C87100 mov     edx, 0071C8E4
0071C846 E8 F5A00BFF call    00406940
0071C84B 8B03       mov     eax, dword ptr [ebx]
0071C84D 83C0 50    add     eax, 50
0071C850 BA F8C87100 mov     edx, 0071C8F8
0071C855 E8 40A60BFF call    00406940

```

Hi hi dom down a bit and you do not see: Call 00407F2C <- this is the Call function with the address of the EP we find on the bypass and DebugBlocker CopyMem2. Now is the time for full IAT, press Alt + M and roll the mouse on the top, then press Ctrl + B to enter: 08 43 F4 77

77 (address GDI32.Arc).

Address	Size	Owner	Section	Contains	Type	Access	Initial
00010000	00002000				Priv	RW	RW
00020000	00001000				Priv	RW	RW
00030000	0000E000				Priv	RW	RW
00123000	00001000				Priv	RW	RW
00124000	0001C000						
00140000	00003000						
00150000	00002000						
00160000	00016000						
00260000	00006000						
00270000	00003000						
00280000	00016000						
002A0000	0003D000						
002E0000	00041000						
00330000	00006000						
00340000	00008000						
00350000	00004000						
00360000	00003000						
00370000	00003000						
00380000	00008000						

Enter binary string to search for

ASCII

UNICODE

HEX +04

☐ Entire block

☐ Case sensitive

OK Cancel

Click OK, we find the IAT Full here:

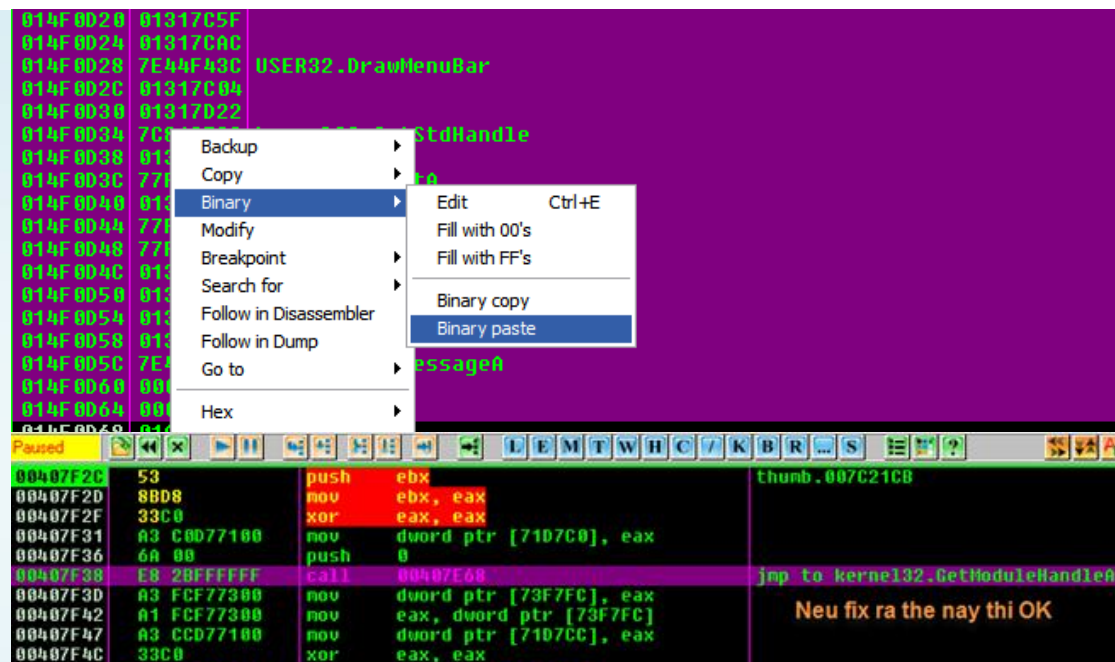
01530048	77F44308	GDI32. Arc
0153004C	01357BF1	
01530050	7E456002	USER32. LoadKeyboardLayoutA
01530054	7E41EF3D	USER32. DestroyIcon
01530058	77C018BA	VERSION. VerQueryValueA
0153005C	77C019FF	VERSION. GetFileVersionInfoSizeA
01530060	77F17C11	GDI32. GetViewportOrgEx
01530064	77F17D01	GDI32. GetViewportExtEx
01530068	7E41C236	USER32. DestroyCaret
0153006C	7E43147A	USER32. GetMenu
01530070	7E42F2B3	USER32. ShowScrollBar
01530074	77C01A50	VERSION. GetFileVersionInfoA
01530078	01357B15	
0153007C	7E42FE31	USER32. SetClassLongA
01530080	77F17AB0	GDI32. SelectClipRgn
01530084	7C80BE01	kernel32. lstrcpyA
01530088	7C80977A	kernel32. InterlockedDecrement
0153008C	7C838AE7	kernel32. _lwrite
01530090	7C8353CE	kernel32. _lread
01530094	7C812A09	kernel32. RaiseException

Check the information found on the same IAT match what we collected in the previous step.

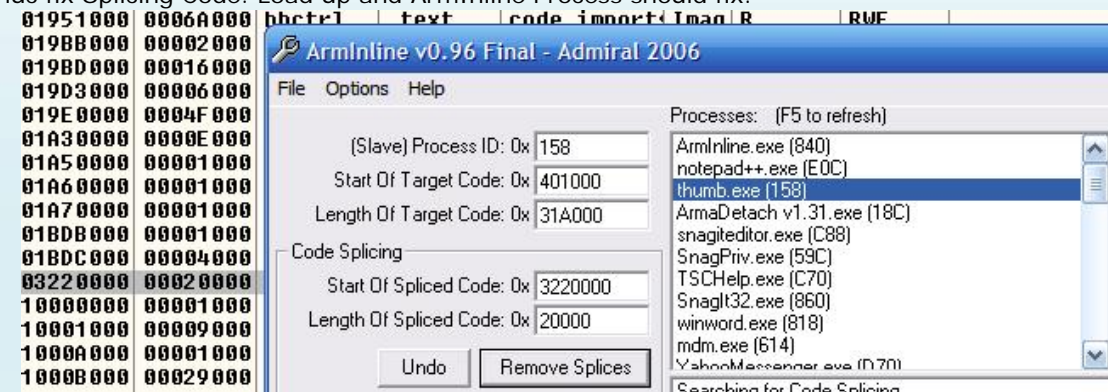
Locate the entire block Full IAT this, right click and select:

01530D28	7E44F43C	USER32. DrawMenuBar
01530D2C	01357C04	
01530D30	01357D22	
01530D34	7C812F39	kernel32. GetStdHandle
01530D38	01357B97	
01530D3C	77F1BBDC	GDI32. TextOutA
01530D40	01357CDA	
01530D44	77F161FF	GDI32. CreateBitmap
01530D48	77F1FE86	GDI32. DeleteEnhMetaFile
01530D4C	01357CE2	
01530D50	01357D00	
01530D54	01357D00	
01530D58	01357D00	
01530D5C	7E41E73D	
01530D60	00000000	
01530D64	00000000	
01530D68	01A50000	
01530D6C	02080000	
01530D70	00000000	

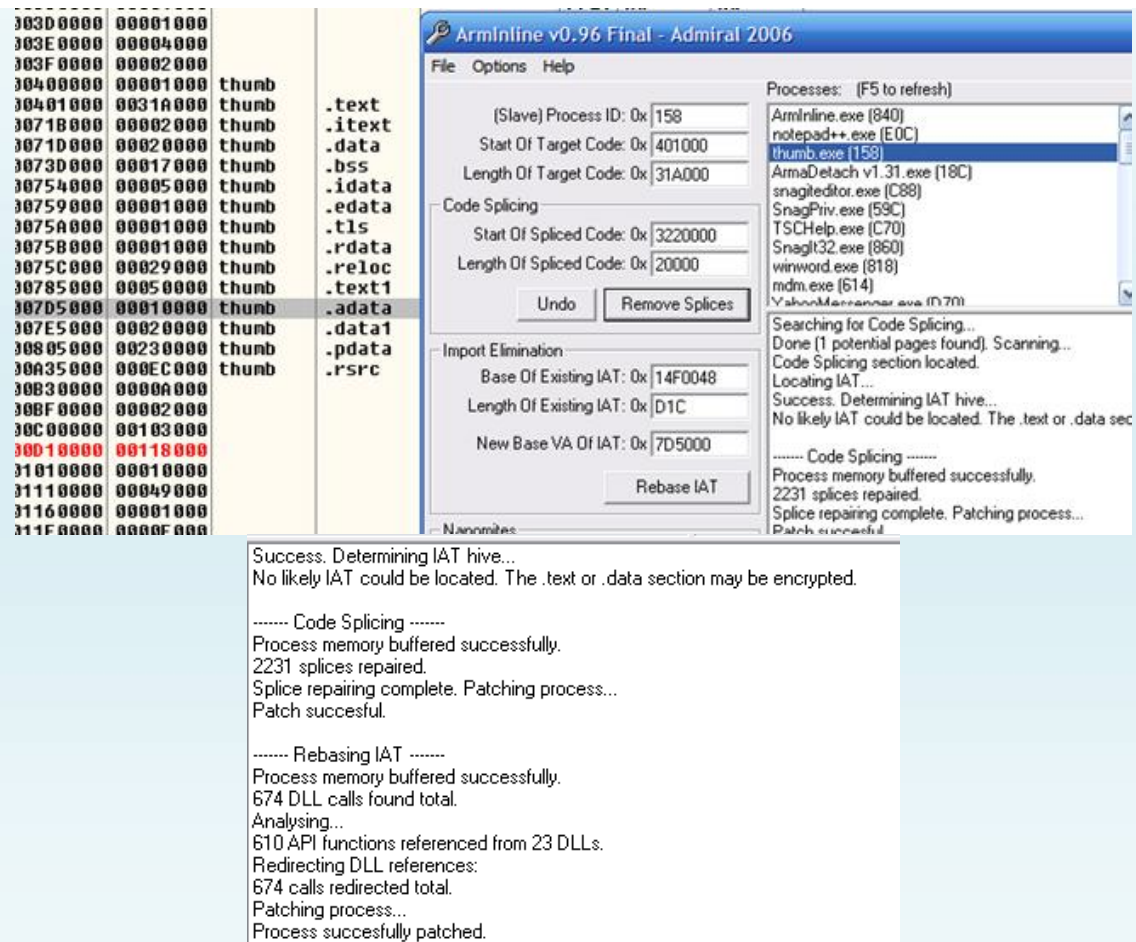
So as we have been Full IAT then, is the Olly IAT used to find and return to the first Olly to continue the process of positioning IAT. To fix IAT need to fix, then right click and select :



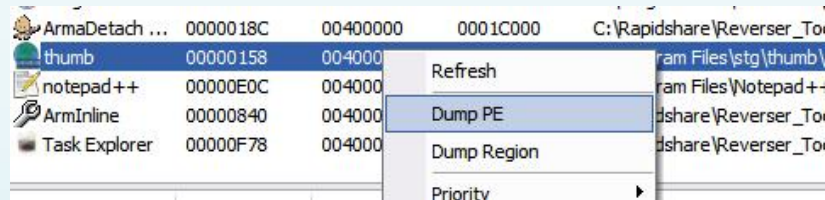
Khuc them rebuild the IAT. Next is khuc fix Splicing Code. Load up and ArmInline Process should fix:



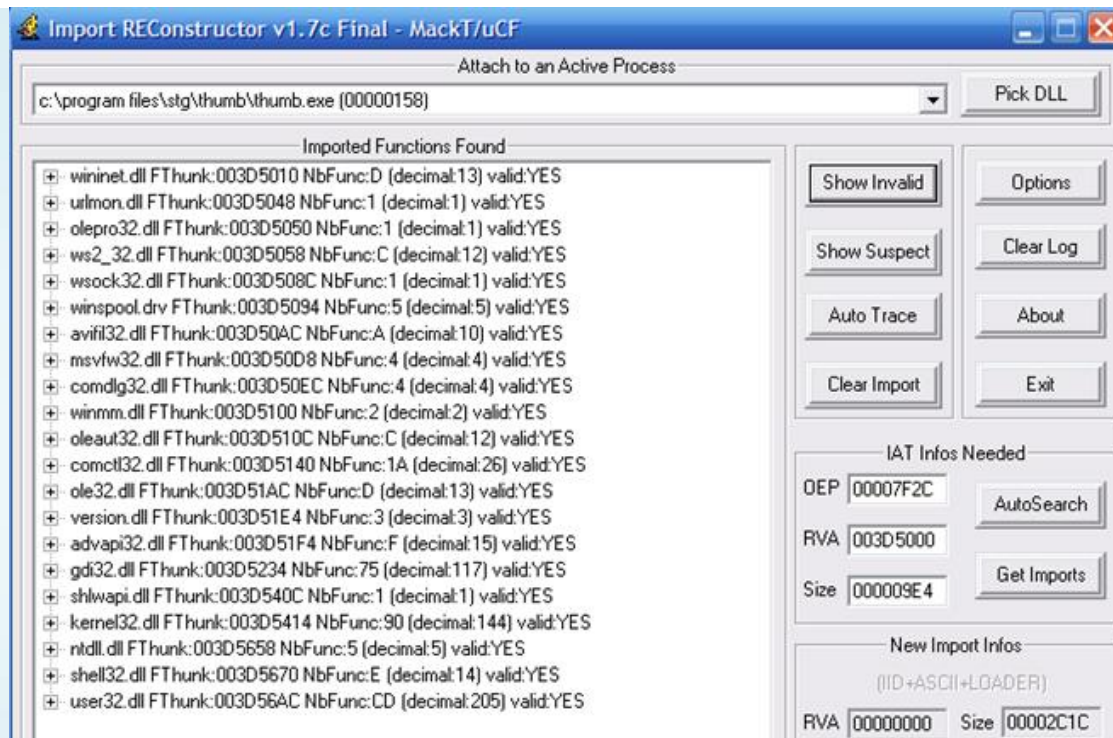
Remove Splices Rebase to complete the IAT are:



After such a fix is completed, open up Task Explorer to dump and save the file with a name thumb_Dumped.exe:



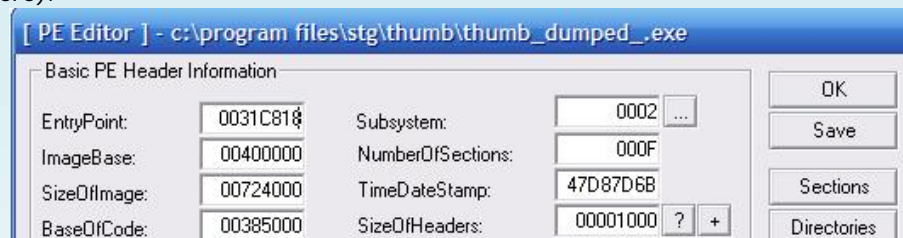
ImportRec continue to open up 1.7c to fix dumped file (cut remove invalid if any):



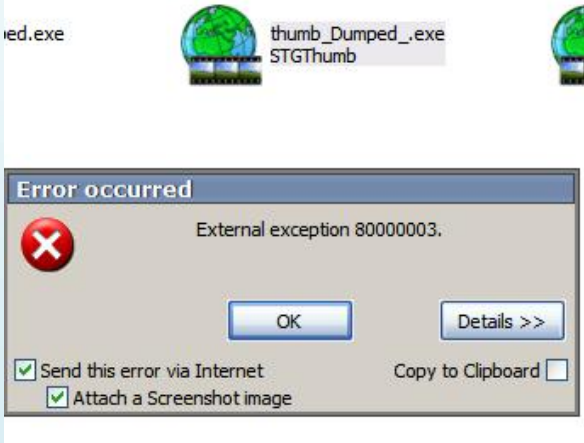
After we complete fix is a new file after the fix is thumb_Dumped_.exe. Now try to run ... see how default ăc do not see all too strange, have only seen each of its process:

SnagitEditor.exe	3208	Snagit Editor 9	TechSmith Corporation
ArmaDetach v1.31.exe	396	Armadillo Process Detach	TEAM RESURREC...
notepad++.exe	3596	Notepad++ : a free (GNU) source code editor	Don HO don.h@free.fr
thumb_Dumped_.exe	336		STGThumb

It looks like I forgot what it must be a right and then to edit the original file OEP on lolz.Cac you remember the original OEP but that 0071C818. Now open the file to load in LordPE and fix the following (remember to kill process of it here):



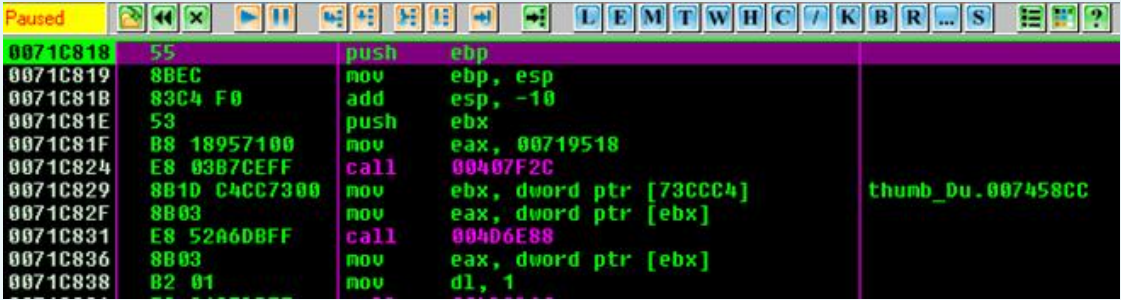
After the fix is completed a test to see how the J , Nag sắc shot kĩa:



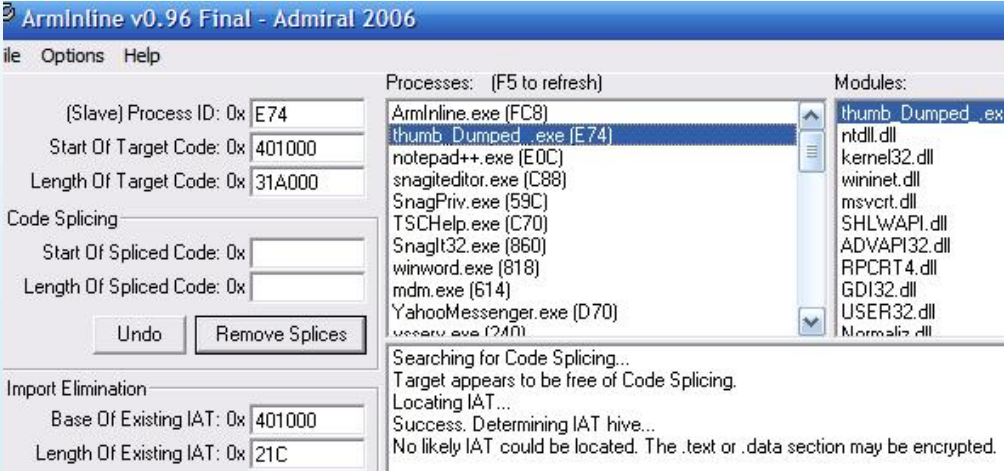
Nag see what this is all you should be happy here, I am also happy because this is more a sign of Nanomites. Now is the time for us to bypass the final floor it.

3. Fix Nanomites

As said before when we see the Nag on file, that is our dump was very good and fix things, only the last obstacle is the only Nanomites again. To bypass this mechanism is used ArmInline is the fastest. The fix what type? I will just for you right here, open up Olly and load the file has to fix the above:

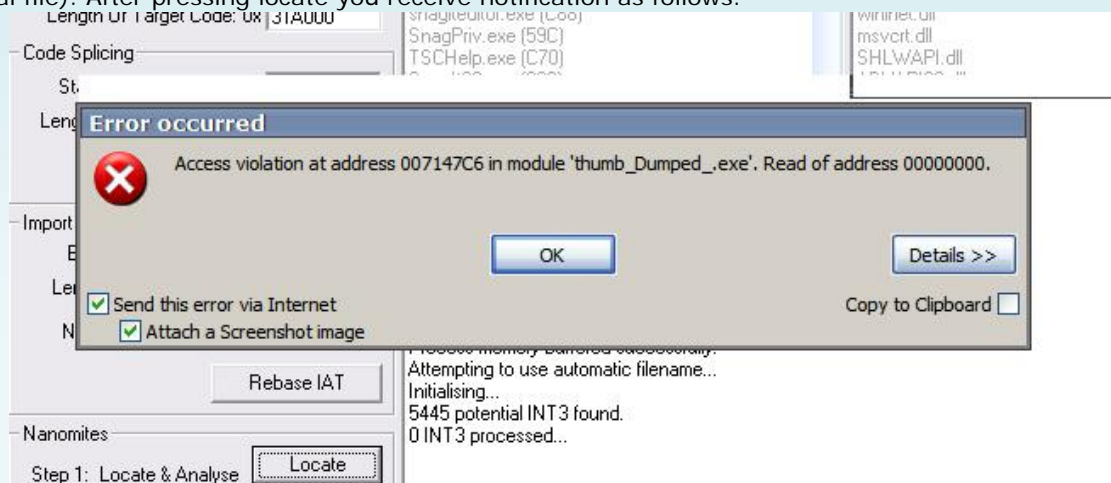


Hold the Olly like, open up ArmInline and process of the program:

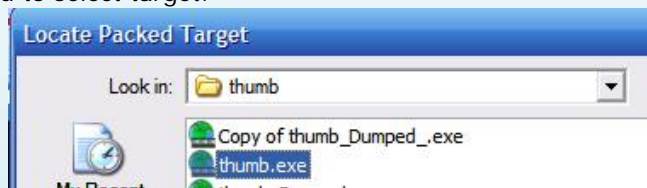


Now you down to that part of Nanomites ArmInline, where you choose to locate (Note: in some cases ArmInline not automatically be to locate the original file you should select the Target to

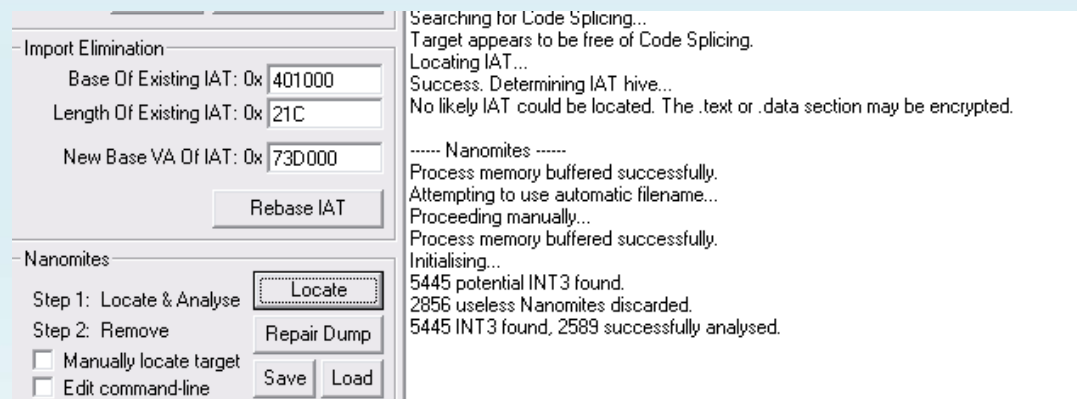
locate manually browse to the original file). After pressing locate you receive notification as follows:



Do not forget it all, uncheck the Send this error via the Internet and click OK. After you click OK the program will be implemented, you close the program will continue to receive notification similar tren.Nhan OK to ignore it, ArmInline will allow you to select target:



Select the original file and click Open, ArmInline automatically analyze and fix Nano us. Results finally obtained as follows:



So is the life tolerable Nanomites then, we select Repair hours and dump file thumb_Dumped_.exe to repair. If Repair Ok brother will receive the following information:

```

----- Nanomites -----
Process memory buffered successfully.
Attempting to use automatic filename...
Proceeding manually...
Process memory buffered successfully.
Initialising...
5445 potential INT3 found.
2856 useless Nanomites discarded.
5445 INT3 found, 2589 successfully analysed.

----- Removing Nanomites -----
Created C:\Program Files\stg\thumb\thumb_Dumped_ NanoFix.exe
Adding new section header...
Appending new data...
Amending PE header...
Done.
Success. Determining IAT hive...
A possible IAT has been located.

```

Now brother just run the file and test results only J . However, when I write this article and Repair the dump file is a sticky this (before the write is not the K) :

```

Locating IAT...
Success. Determining IAT hive...
No likely IAT could be located. The .text or .data section may be encrypted.

----- Nanomites -----
Process memory buffered successfully.
Attempting to use automatic filename...
Proceeding manually...
Process memory buffered successfully.
Initialising...
5445 potential INT3 found.
2856 useless Nanomites discarded.
5445 INT3 found, 2589 successfully analysed.

----- Removing Nanomites -----
Created C:\Program Files\stg\thumb\thumb_Dumped_ NanoFix.exe
Unable to add a section to the dump.
Try removing some of Armadillo's residual sections.

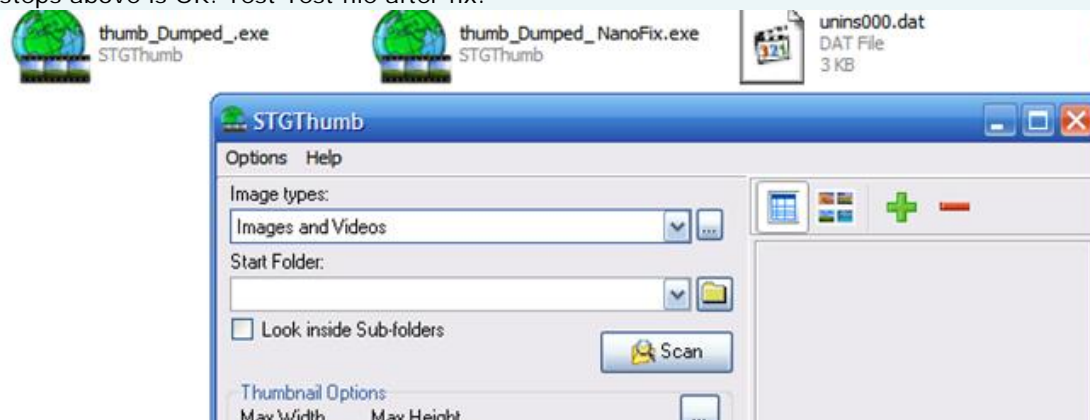
```

If you stick to this case you are not worried at all, just load the file in Explorer and CFF reduce the cut sections of the Armadillo:

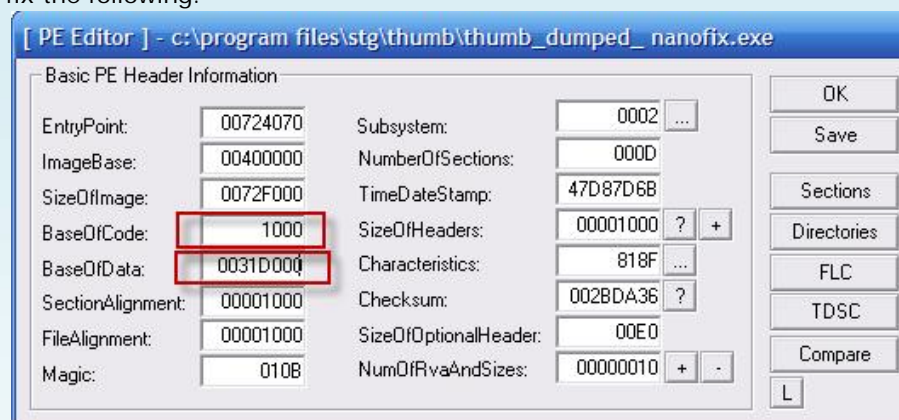
thumb_Dumped_exe									
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumbers ...	Characteristics
00000360	00000368	0000036C	00000370	00000374	00000378	0000037C	00000380	00000382	00000384
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.itext	00001908	0031B000	00001908	0031B000	00000000	00000000	0000	0000	60000020
.data	0001FFAC	0031D000	0001FFAC	0031D000	00000000	00000000	0000	0000	C0000040
.bss	00016850	0033D000	00016850	0033D000	00000000	00000000	0000	0000	C0000000
.idata	00004890	00354000	00004890	00354000	00000000	00000000	0000	0000	C0000040
.edata	000002DC	00359000	000002DC	00359000	00000000	00000000	0000	0000	40000040
.tls	0000016C	0035A000	0000016C	0035A000	00000000	00000000	0000	0000	C0000000
.rdata	00000018	0035B000	00000018	0035B000	00000000	00000000	0000	0000	40000040
.reloc	00028108	0035C000	00028108	0035C000	00000000	00000000	0000	0000	42000040
.text1	00050000	00385000	00050000	00385000	00000000	00000000	0000	0000	E0000020
.adata	00010000	003D5000	00010000	003D5000	00000000	00000000	0000	0000	E0000020
.data1	00020000	003E5000	00020000	003E5000	00000000	00000000	0000	0000	C0000040
.pdata	00230000	00405000	00230000	00405000	00000000	00000000	0000	0000	C0000040
.rsrc	000EC000	00635000	000EC000	00635000	00000000	00000000	0000	0000	40000040
.mact	00003000	00721000	00003000	00721000	00000000	00000000	0000	0000	E0000060

thumb_Dumped_exe									
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	0031A000	00001000	00319210	00001000	00000000	00000000	0000	0000	60000020
.itext	00002000	0031B000	00001908	0031B000	00000000	00000000	0000	0000	60000020
.data	00020000	0031D000	0001FFAC	0031D000	00000000	00000000	0000	0000	C0000040
.bss	00017000	0033D000	00016850	0033D000	00000000	00000000	0000	0000	C0000000
.idata	00005000	00354000	00004890	00354000	00000000	00000000	0000	0000	C0000040
.edata	00001000	00359000	000002DC	00359000	00000000	00000000	0000	0000	40000040
.tls	00001000	0035A000	0000016C	0035A000	00000000	00000000	0000	0000	C0000000
.rdata	00001000	0035B000	00000018	0035B000	00000000	00000000	0000	0000	40000040
.reloc	00079000	0035C000	00028108	0035C000	00000000	00000000	0000	0000	42000040
.adata	00260000	003D5000	00010000	00385000	00000000	00000000	0000	0000	E0000020
.rsrc	000EC000	00635000	000EC000	00395000	00000000	00000000	0000	0000	40000040
.mact	00003000	00721000	00003000	00481000	00000000	00000000	0000	0000	E0000060

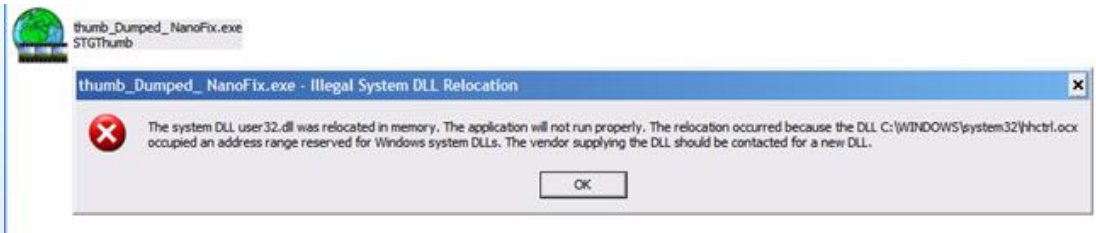
Then proceed to Fix Nanomites as the steps above is OK. Test Test file after fix:



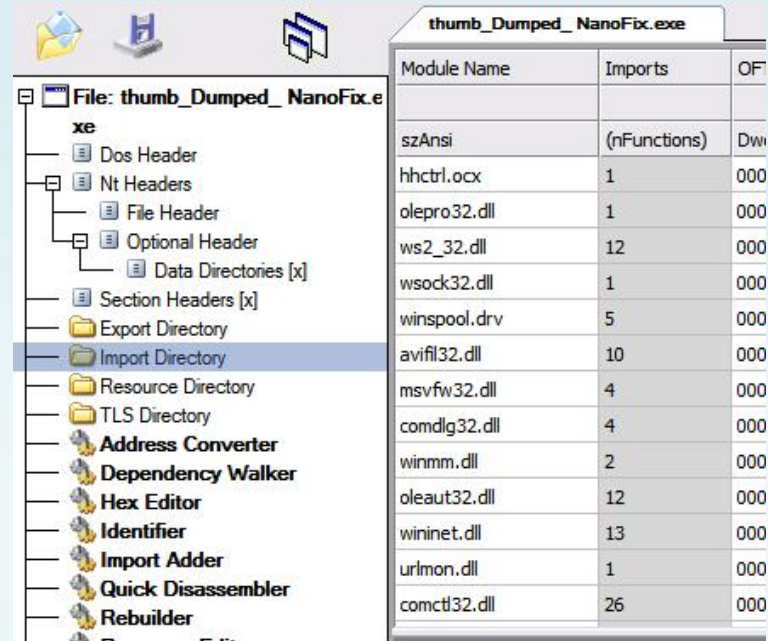
File to run OK, more like beautiful brothers to fix the following:



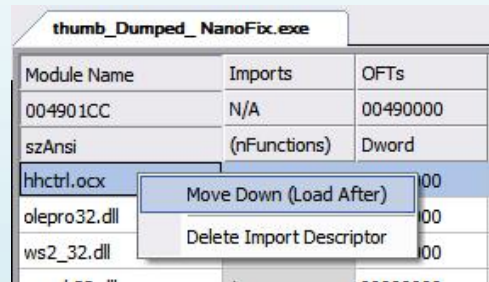
Note: In some cases the very useful new experience, after you finish all fix everything and run the file you receive the Nag as follows:



Read the notice on sure you also have to understand is notorious and the J . You do not worry too in this case, the files are still performing well but do not run it normally is because thăng hhctrl.ocx has occupied an area that would address the region must for System DLLs. So How do I run files without the sticky nag, very simple one is to load it follows that the system dll on Nag announced. Are there any tool does not support, there is J CFF Explorer. OK time to load exe files CFF:



Select the Import as the Directory, select hhctrl.ocx and click to select:



Move it down user32.dll is OK.

thumb_Dumped_NanoFix.exe				
Module Name	Imports	OFTs	TimeDateStamp	Forward
004901CC	N/A	0049012C	00490130	004901
szAnsi	(nFunctions)	Dword	Dword	Dword
wininet.dll	13	00000000	00000000	000000
urlmon.dll	1	00000000	00000000	000000
comctl32.dll	26	00000000	00000000	000000
ole32.dll	13	00000000	00000000	000000
version.dll	3	00000000	00000000	000000
user32.dll	205	00000000	00000000	000000
hhctrl.ocx	1	00000000	00000000	000000

Finally Save the file and test our. File enforcement normal! Phu

IV. Conclusion

Finally, the complete message, this article I wrote for the purpose of storage and only the basic points to unpack Armadillo but I do not go more deeply into each section of these have too many resources no. Them's talk about the record more knowledge you are with me it is a pleasure, a feeling for their own Refresh your knowledge of minh. Rat thank him and you for taking the time valuable to you to read this document.

PS: This document is only a reference, the author is not responsible if the reader uses it to any goal.

Best Regards

 [Kienmanowar]



---+---==[Greatz thanks to]=---+---

My family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHQCrker, the_Lighthouse, Merc, Hoadongnoi, Nini ... all REA's members, TQN, HacNho, RongChauA, Deux, tlandn, light.phoenix, dqtlN, ARTEAM all my friend, and you.

Thanks to ---+---==[]=---+---

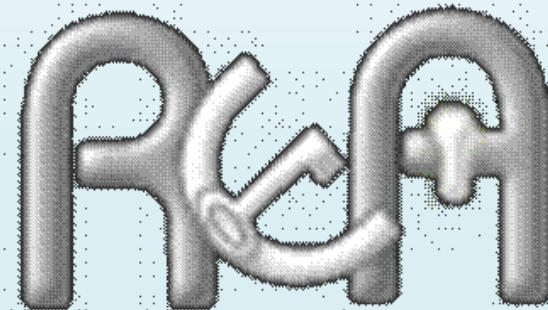
iamidiot, WhyNotBar, trickyboy, dzungltvn, takada, hurt_heart, haule_nth, hytkl, Moth, XIANUA, nhc1987, v. Oxdie. v.. You have contributed greatly to the REA. Hope you will continue to promote J

I want to thank **Teddy Roggers** for his great site, Reversing.be folks (especially **haggar**), Arteam folks (**Shub-Nigurrath**, **MaDMAn_H3rCuL3s**) and all folks on crackmes.de, thank to all members of **unpack.cn** (especially **fly** and **linhanshi**). Great thanks to **lena151** (I like your tutorials). And finally, thanks to **Ricardo NARVAJA** and all members on **CRACKSLATINOS**.

>>>> If you have any suggestions, comments or corrections email me: **kienmanowar [at] reaonline.net**

2008

[MUP Armadillo v5.42 Case Study]



www.reasonline.net

kienmanowar

www.reaonline.net

6/14/2008

Contents

[I.](#)

[II. The sudung](#)

[III. Unpack Armadillo v5.42](#)

[1st find information about Unpackme.](#)

[2. CopyMemII extravagant and DebugBlocker](#)

[3rd Fix IAT - bypass CodeSplicing - Rebase IAT:](#)

[4. Fix DUMPED test file and the file after the fix IAT:](#)

[IV. Unpack using the tools](#)

[V.](#)

I. Introduction

Welcome brother, you see walking this increasingly too slow. Look brother RlPack exchange, EncryptPE, ExEcRyptor, PECompact, v. WinLic. V.. just as not, you still do not know is just outside ngo K. Take the one to be the lượm unpackme the pack with Armadillo v.5.42 chance to try to try, who are easy-to-use real currency. Try to try again just as often fail ^ ^. Finally, it must also seek to help his children be mup. Tutor of the full Armadillo Ray on the network, even small aged (hacnho) has certainly show a series of it (no one should read). This article I wrote the main purpose is to document storage, then the old error reading for pleasure. I do not go deep more, want more for reference by tut-aged small and Tutor mang.Now on, let's go

II. The Tools use

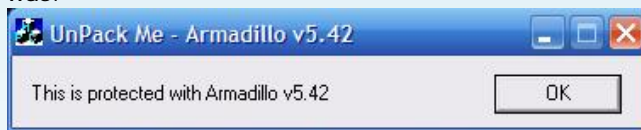
Posts using the following:

1. Ollylce (ver **2008.1.1**): **This was introduced in the REA, include it available to plug can help brothers bypass the mechanism of anti-debug Armadillo**
2. Armadillo Find Protected V1.8
3. ArmaDetach v1.31
4. ArmInline v0.96f (Eng)
5. Task LordPE or Explorer.
6. ImportRec v1.7c Final.
7. CFF Explorer.
8. Suite 2005 PUPE-Universal Process Patcher
9. Arma 5.x Fix Magic Call.osc

III. Manual unpack Armadillo v5.42

1. Seeking information on Unpackme.

Before treatment unpackme we face test run nose how it was:



Dom try the section of it:

Armadillo_UnPack Me.exe									
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00011B8E	00001000	00000000	00000000	00000000	00000000	0000	0000	60000020
.rdata	0000453E	00013000	00000000	00000000	00000000	00000000	0000	0000	40000040
.data	00006348	00006000	00000000	00000000	00000000	00000000	0000	0000	C0000040
text1	00050000	00010000	00000000	00000000	00000000	00000000	0000	0000	E0000020
adata	00010000	0006F000	0000D000	00043000	00000000	00000000	0000	0000	E0000020
data1	00020000	0007F000	0000B000	00050000	00000000	00000000	0000	0000	C0000040
pdata	00060000	0009F000	00057000	0005B000	00000000	00000000	0000	0000	C0000040
.rsrc	00003000	000FF000	00001000	000B2000	00000000	00000000	0000	0000	40000040

Next we need to check it has been used as pack with Armadillo ver 5:42 or not? And protect the mechanism is applied to it. Fortunately, doctors have vel "and" a tool to perform this job is Armadillo Find Protect. Armadillo Run Find Protect v1.8, then drag & drop to unpackme we obtained the following information:

15-06-2008 15:13:20 <----->

C: \ Documents and Settings \ m4n0w4r \ Desktop \ Armadillo_UnPack Me.exe

- Protected Armadillo

Protection system (Professional)

- <Protection Options>

Debug-Blocker

CopyMem-II

Enable elimination Import Table

Enable Strategic Code Splicing

Enable Memory-Patching Protections

- Key <Backup Options>

Fixed Backup Keys

- <Compression Options>

Best / Slowest Compression

- <Other Options>

Allow Only One Copy

- **Version 5.42 20-02-2008 to exact**

- Elapsed Time 00h 00m 01s 812ms

Rub it is almost all aspects of the mechanism by Protect Armadillo, just missing the end of each level is more Nanomites thoi.Ok so is the collection of information is finished, we carried out any work.

2. Through face CopyMemII and DebugBlocker

First, we target to load Olly was then calculated to:

0046F000	Armadill.<ModuleEntryPoint>	60	pushad
0046F001		E8 00000000	call 0046F006
0046F006		5D	pop ebp
0046F007		50	push eax
0046F008		51	push ecx
0046F009		0FCA	bswap edx
0046F00B		F7D2	not edx
0046F00D		9C	pushfd
0046F00E		F7D2	not edx
0046F010		0FCA	bswap edx
0046F012		EB 0F	jmp short 0046F023

Set in a hwbp API is WaitForDebugEvent:

Command	he WaitForDebugEvent	HE address -- HW break on execution
Start:47F000 End:47EFFF Value:77F1B221		

Armadillo do it playing the **Memory Patching Protection**, so it will detect the BP we set at any

time API. So to bypass this mechanism we use hwbp.Ok, after BP put on as we press Shift + F9 to run unpackme, the program will break in our BP. Now dom through window Stack we obtained the following information:

0013DD80	00446208	CALL to WaitForDebugEvent
0013DD84	0013ECB4	pDebugEvent = 0013ECB4
0013DD88	000003E8	Timeout = 1000. ms
0013DD8C	2DA95FDF	

You save this information to:

0013DD80 00446208 / Call to WaitForDebugEvent

0013DD84 0013ECB4 | pDebugEvent = 0013ECB4

0013DD88 000003E8 \ Timeout = 1000. ms

Now we press Ctrl + F2 to restart Olly, put in hwbp WaitForDebugEvent go. Set in a hwbp WriteProcessMemory, press Shift + F9 to see when it is preparing to write 1000 bytes is stopped (usually hit 3 times is found):

0013DCF0	00449440	CALL to WriteProcessMemory from Armadill.0044943A
0013DCF4	00000078	hProcess = 00000078
0013DCF8	00401000	Address = 401000
0013DCFC	00BF6430	Buffer = 00BF6430
0013DD00	00001000	BytesToWrite = 1000 (4096.)
0013DD04	0013DD3C	pBytesWritten = 0013DD3C
0013DD08	0013DD14	

Record information on:

0013DCF0 00449440 / Call to WriteProcessMemory from Armadill.0044943A

0013DCF4 00000078 | 00000078 = hProcess

0013DCF8 00401000 | Address = 401000

0013DCFC 00BF6430 | Buffer = 00BF6430

0013DD00 00001000 | BytesToWrite = 1000 (4096).

0013DD04 0013DD3C \ = pBytesWritten 0013DD3C

Through the window we dump press Ctrl + G and enter the address of pDebugEvent that we find the above:

0013ECB4	00000001	
0013ECB8	00000058	
0013ECBC	00000120	
0013ECC0	80000001	
0013ECC4	00000000	
0013ECC8	00000000	
0013ECCC	00401E6E	Armadill.00401E6E
0013ECD0	00000002	OEP
0013ECD4	00000008	
0013ECD8	00401E6E	Armadill.00401E6E
0013ECDC	00401E6E	Armadill.00401E6E
0013ECE0	00AC7C28	
0013ECE4	86420020	
0013ECE8	805B00C7	
0013ECEC	000000B8	
0013ECE8	00000000	

According to what we have above, we know our OEP is 0x00401E6E. Process parents will write the 1000 bytes to process from the start address is 0x401000. So we need to do next is to find and "destroy" function call to 1000 bytes. To find it we do the following, dom through window Stack at this time we're stopped the hwbp, mouse scroll down a bit until you see the first return:

0013DD30	00BF7430	
0013DD34	00000020	
0013DD38	00401000	Armadill.00401000
0013DD3C	00001000	
0013DD40	00000020	
0013DD44	00BF7430	
0013DD48	0013DD78	
0013DD4C	00408755	RETURN to Armadill.00408755 from Armadill.00408AA0
0013DD50	00000000	
0013DD54	00BF5FC4	
0013DD58	00000000	
0013DD5C	00000007	

Address 0x00448AA0 second is the address that we need to concern because it is based on, we will find a specific function or is MagicCall EncryptCall or that related to CopyMem function. Ok, time flying through the window CPU and press Ctrl + G, type in the address 0x00448AA0 will come:

00448AA0	\$ 55	push	ebp	
00448AA1	. 8BEC	mov	ebp, esp	
00448AA3	. 83EC 40	sub	esp, 40	
00448AA6	. 8B45 08	mov	eax, dword ptr [ebp+8]	
00448AA9	. C1E0 0C	shl	eax, 0C	
00448AAC	. 0305 B863480	add	eax, dword ptr [486308]	Armadill.00401000
00448AB2	. 8945 F0	mov	dword ptr [ebp-10], eax	
00448AB5	. 8B00 E063480	mov	ecx, dword ptr [4863E0]	
00448AB8	. 8940 FC	mov	dword ptr [ebp-4], ecx	
00448ABF	. 8B45 F0	mov	edx, dword ptr [ebp-10]	

Pull back down we have information:

0044941E	> 8055 F4	lea	edx, dword ptr [ebp-c]	
00449421	. 52	push	edx	
00449422	. 68 00100000	push	1000	pBytesWritten BytesToWrite = 1000 (4096..)
00449427	. A1 E0634800	mov	eax, dword ptr [4863E0]	Buffer => 00BF6430
0044942C	. 50	push	eax	Address
0044942D	. 8B40 F0	mov	ecx, dword ptr [ebp-10]	
00449430	. 51	push	ecx	
00449431	. 8B15 B463480	mov	edx, dword ptr [4863B4]	
00449437	. 8B02	mov	eax, dword ptr [edx]	
00449439	. 50	push	eax	hProcess
0044943A	. FF15 0CF1470	call	dword ptr [<KERNEL32.WriteProc	WriteProcessMemory
00449440	. 85C0	test	eax, eax	

At which we press Ctrl + G, we choose ebp push and press Ctrl + R to find all references to function at this:

Address	Disassembly	Comment
00448750	call 00448AA0	
00448A1B	call 00448AA0	
00448AA0	push ebp	(Initial CPU selection)

According to the context of my money has to go before any analysis of the function that the second Call meat, so double click function second call we will call this to function. NOP conducted it is finished!

00448A17	. 8B048A	mov	eax, dword ptr [edx+ecx*4]
00448A1A	. 50	push	eax
00448A1B	. 90	nop	
00448A1C	. 90	nop	
00448A1D	. 90	nop	
00448A1E	. 90	nop	
00448A1F	. 90	nop	
00448A20	. 83C4 0C	add	esp, 0C
00448A23	. 50	push	eax

The work of the next one is sever relations hold both father and child process by creating a loop of tan.Sau that the next time we OEP month child process. I press Ctrl + F9 to out from the API is WriteProcessMemory:

00449440	85C0	test	eax, eax
00449442	75 4B	jnz	short 0044948F
00449444	50	push	eax
00449445	F7D0	not	eax

You also remember window dump in the air, not sure if you find directly to the `pDebugEvent` :

013ECC0	00401E6E	Armadillo.00401E6E
013ECD0	00000002	
013ECD4	00000008	
013ECD8	00401E6E	Armadillo.00401E6E
013ECDC	00401E6E	Armadillo.00401E6E
013ECE0	000C7C28	

As above, we have been OEP's child is 0x00401E6E process. We need information about them later, so save it again:

0013ECB4 00000001

0013ECB8 00000A10 child to process ID

0013ECBC 00000D2C

0013ECC0 80000001

0013ECC4 00000000

0013ECC8 00000000

0013ECCC 00401E6E Armadillo.00401E6E

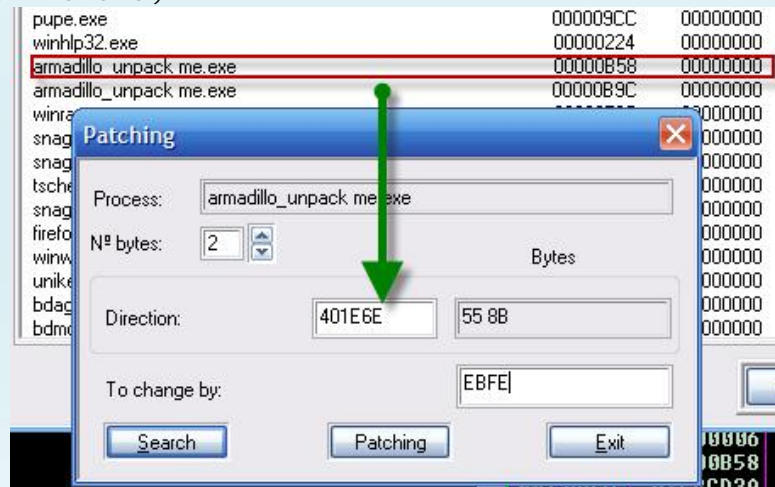
0013ECD0 00000002

0013ECD4 00000008

0013ECD8 00401E6E Armadillo.00401E6E

0013E CDC 00401E6E Armadillo.00401E6E

Now we'll patch child process at the OEP its infinite loop. PUPE Open up and fill in as follows (remember to find the right child process by PID small J) :



Remember the original two bytes are 55 8B because we need them to restore OEP simultaneously PUPE close again. After patch back Ollly finished, remove `hwbp` in `WriteProcessMemory` and reset `hwbp` in `WaitForDebugEvent`. Press Shift + F9 to run the program, we will break in `hwbp`:

0013DD80	00446208	CALL to WaitForDebugEvent from Armadillo.00446202
0013DD84	0013ECB4	pDebugEvent = 0013ECB4
0013DD88	000003E8	Timeout = 1000, ms
0013DD8C	86F010EB	

Click the first line and select Follow in Disassembler, we come:

```

004461F3  > 3361 68      xor     esp, dword ptr [ecx+68]
004461F7  ? E8 0300008B  call    8B4461FF
004461FC  ? 95          xchg    eax, ebp
004461FD  ? B0 F5       mov     al, 0F5
004461FF  ? FFFF       ???
00446201  - 52          push    edx
00446202  - FF15 DCF0478 call    dword ptr [ &KERNEL32.WaitForDe
00446208  - 85C0       test    eax, eax
0044620A  ~ 0F84 3120000 je      00448241
00446210  ~ 0FB645 D3   movzx   eax, byte ptr [ebp-20]

```

Unknown command
pDebugEvent
WaitForDebugEvent
here

Submit your entire code related to WaitForDebugEvent, and set new origin in order test eax, eax:

```

004461F3  > 90          nop
004461F4  > 90          nop
004461F5  > 90          nop
004461F6  > 90          nop
004461F7  > 90          nop
004461F8  > 90          nop
004461F9  > 90          nop
004461FA  > 90          nop
004461FB  > 90          nop
004461FC  > 90          nop
004461FD  > 90          nop
004461FE  > 90          nop
004461FF  > 90          nop
00446200  > 90          nop
00446201  > 90          nop
00446202  > 90          nop
00446203  > 90          nop
00446204  > 90          nop
00446205  > 90          nop
00446206  > 90          nop
00446207  > 90          nop
00446208  - 85C0       test    eax, eax

```

pDebugEvent
WaitForDebugEvent

```

00446207  > 90          nop
00446208  - 85C0       test    eax, eax
0044620A  ~ 0F84 3120000 je      00448241
00446210  - 0FB645 D3   movzx   eax, byte ptr [ebp-20]
00446214  - 85C0       test    eax, eax
00446216  ~ 74 10      je      short 00446228
00446218  - 8B0D C863480 mov     ecx, dword ptr [48
0044621E  - 8379 20 00   cmp     dword ptr [ecx+20]
00446222  ~ 74 04      je      short 00446228
00446224  - C645 D3 00   mov     byte ptr [ebp-20],
00446228  > 68 F4614800 push    004861F4
0044622D  - FF15 00F1478 call    dword ptr [ &KERN

```

Backup
Copy
Binary
Assemble Space
Label :
Comment ;
Breakpoint
Hit trace
Run trace
New origin here Ctrl+Gray *

Now we need a code to decrypt 1000 bytes, where we choose section. Text as a starting place for this code. So section. Text is where, press Alt + M to open the window Memory:

00010000	00001000	Armadill		PF header	Image	R	RWE
00401000	00012000	Armadill	.text		Image	R	RWE
00413000	00005000	Armadill	.rdata		Image	R	RWE
00418000	00007000	Armadill	.data		Image	R	RWE
0041F000	00050000	Armadill	.text1	code	Image	R	RWE

OK, so section. Text begins at 0x00401000. I edit the command test eax, eax it jumps to the region this section:

```

Paused
00446208  - E9 F3ADF0FF jmp     00401000
0044620D  > 90          nop
0044620E  > 90          nop
0044620F  > 90          nop

```

Now we press Enter `jmp 00401000` in order to follow the areas that we need to create code. One patch as follows:

```

Paused
00401000 C705 B4EC1300 mov dword ptr [13ECB4], 1
0040100A C705 B8EC1300 mov dword ptr [13ECB8], 0A10
00401014 C705 BCEC1300 mov dword ptr [13ECBC], 002C
0040101E C705 C0EC1300 mov dword ptr [13ECC0], 80000001
00401028 C705 C4EC1300 mov dword ptr [13ECC4], 0
00401032 C705 C8EC1300 mov dword ptr [13ECC8], 0
0040103C 8105 CCEC1300 add dword ptr [13ECCC], 1000
00401046 8105 D8EC1300 add dword ptr [13ECD8], 1000
00401050 8105 DCEC1300 add dword ptr [13ECD0], 1000
0040105A 813D DCEC1300 cmp dword ptr [13ECD0], 00401000
00401064 ^ 74 D6 je short 0040103C
00401066 813D DCEC1300 cmp dword ptr [13ECD0], 00413000
00401070 - 0F85 97510400 jnz 00446200
00401076 68 100A0000 push 0A10
0040107B E8 2995457C call kernel32.DebugActiveProcessStop
00401080 90 nop
00401081 90 nop

```

Next we follow the `pDebugEvent` in the window and dump 3 patch addresses containing OEP:

```

0013ECB4 00000001
0013ECB8 00000A10
0013ECBC 00000D2C
0013ECC0 80000001
0013ECC4 00000000
0013ECC8 00000000
0013ECCC 00400000 Armadill.00400000
0013ECD0 00000002
0013ECD4 00000008
0013ECD8 00400000 Armadill.00400000
0013ECC0 00400000 Armadill.00400000

```

OK, after such a path is set in a hwbp `0040107B E8 2995457C call kernel32.DebugActiveProcessStop` and press Shift + F9 to run the program:

```

00401000 C705 B4EC1300 mov dword ptr [13ECB4], 1
0040100A C705 B8EC1300 mov dword ptr [13ECB8], 0A10
00401014 C705 BCEC1300 mov dword ptr [13ECBC], 002C
0040101E C705 C0EC1300 mov dword ptr [13ECC0], 80000001
00401028 C705 C4EC1300 mov dword ptr [13ECC4], 0
00401032 C705 C8EC1300 mov dword ptr [13ECC8], 0
0040103C 8105 CCEC1300 add dword ptr [13ECCC], 1000
00401046 8105 D8EC1300 add dword ptr [13ECD8], 1000
00401050 8105 DCEC1300 add dword ptr [13ECD0], 1000
0040105A 813D DCEC1300 cmp dword ptr [13ECD0], 00401000
00401064 ^ 74 D6 je short 0040103C
00401066 813D DCEC1300 cmp dword ptr [13ECD0], 00413000
00401070 - 0F85 97510400 jnz 00446200
00401076 68 100A0000 push 0A10
0040107B E8 2995457C call kernel32.DebugActiveProcessStop
00401080 90 nop

```

Registers (FPU)

```

EAX 00000000
ECX 00000001
EDX 7C90E894 ntdll.KiFastSystemCallRet
EBX 00000001
ESP 0013D05C
EBP 0013F6FC
ESI 00000113
EDI 00000003
EIP 0040107B Armadill.0040107B
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 0018 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDF000(FFF)

```

Tolerable, we are stopped at King hwbp set, dom Registers through the window you see the value of EAX is 0. If we trace through `DebugActiveProcessStop` function that returns the value in EAX is not a non-zero, the as we go and dust from the beginning K.

DebugActiveProcessStop Function

Stops the debugger from debugging the specified process.

```
BOOL WINAPI DebugActiveProcessStop(
    __in DWORD dwProcessId
);
```

Parameters

dwProcessId

The identifier of the process to stop debugging.

Return Value

If the function succeeds, the return value is nonzero.

Now we press F8 to trace through this function and that the value of EAX to write:

Address	Disassembly
00401080	mov dword ptr [13ECB4], 1
0040108A	mov dword ptr [13ECB8], 0A10
00401094	mov dword ptr [13ECBC], 002C
0040109E	mov dword ptr [13ECC0], 80000001
004010A8	mov dword ptr [13ECC4], 0
004010B2	mov dword ptr [13ECC8], 0
004010BC	add dword ptr [13ECC0], 1000
004010C6	add dword ptr [13ECC4], 1000
004010D0	add dword ptr [13ECC8], 1000
004010DA	cmp dword ptr [13ECC0], 00401000
004010E4	je 74 D6
004010EE	cmp dword ptr [13ECC0], 00413000
004010F8	jnz 0F85 97510400
00401102	push 0A10
0040110C	call kernel32.DebugActiveProcessStop
00401116	nop
00401120	nop

Register	Value
EAX	00000001
ECX	00130840
EDX	7C90EB94 ntdll
EBX	00000001
ESP	00130860
EBP	0013F6FC
ESI	00000113
EDI	00000003
EIP	00401080 Armadillo

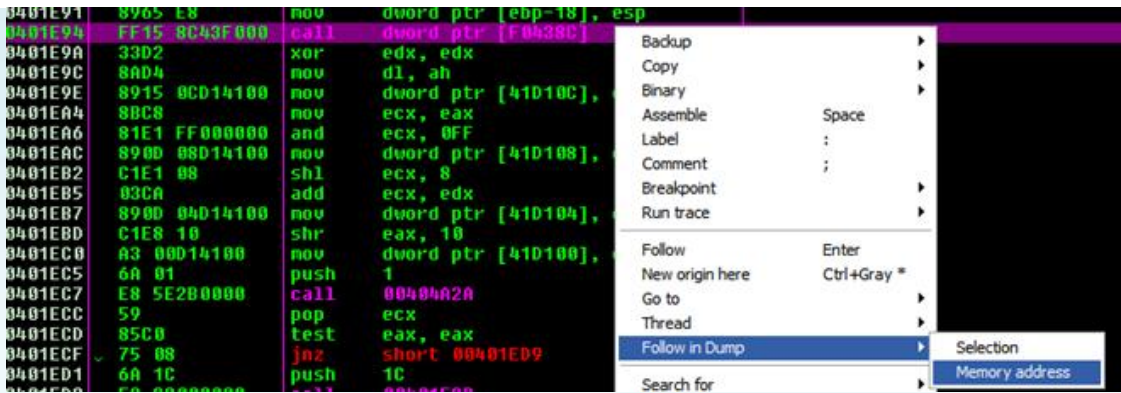
Ola, it is CopyMem2 and DebugBlocker have to dust not we go J dust. Now we open a more Olly and child Attach to process, run it and press F12 and recover the OEP in bytes (bytes of this we are at one with the patch PUPE):

Address	Disassembly
00401E6E	push ebp
00401E6F	mov ebp, esp
00401E71	push -1
00401E73	push 00414558
00401E78	push 00404A90
00401E7D	mov eax, dword ptr fs:[0]
00401E83	push eax
00401E84	mov dword ptr fs:[0], esp
00401E8B	sub esp, 58

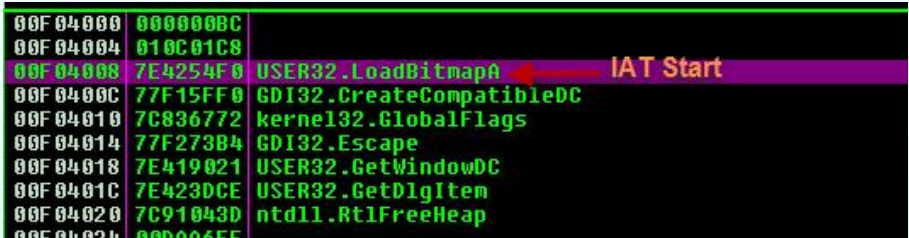
So is temporary finished! Next we fix IAT, fix CodeSplicing and rebase the IAT.

3. Fix IAT - bypass CodeSplicing - Rebase IAT:

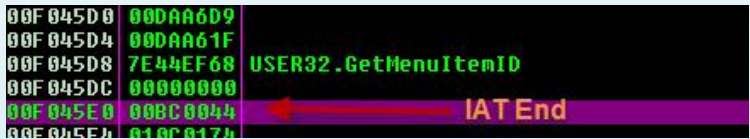
At the screen when Olly Attach one child process, pulling down a bit to see Call command as follows: 00401E94 FF15 8C43F000 call dword ptr [F0438C]. Mouse in order to Call and select as follows:



At window dump we'll see many API functions, so the one here is not the IAT program. Now you roll your mouse over the top for IAT start:



Continue to roll the mouse down for IAT End:



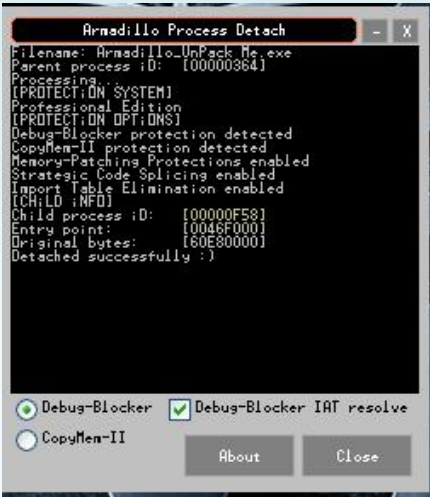
Summarized again:

IAT Start: 00F04008 7E4254F0 USER32.LoadBitmapA

IAT End: 00F045E0 00BC0044

IAT Size: 5D8

Now we go looking for Full IAT. Search by any now? Search by "read a" respectable to do so, time is running ArmaDetach v1.31, then drag our target to:



You open an Olly Attach another child and that the process has to detach. F9 and F12 to run

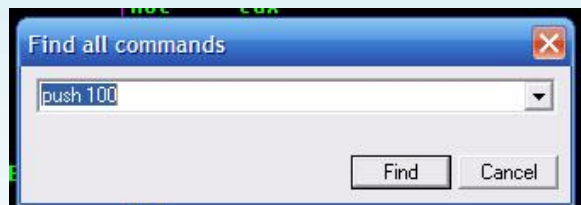
the program to pause, and then recover the bytes:

0046F000	60	pushad
0046F001	E8 00000000	call 0046F006
0046F006	5D	pop ebp
0046F007	50	push eax
0046F008	51	push ecx
0046F009	0FCA	bswap edx
0046F00B	F7D2	not edx
0046F00D	9C	pushfd
0046F00E	F7D2	not edx
0046F010	0FCA	bswap edx
0046F012	EB 0F	jmp short 0046F023

Next we set a hwbp he VirtualProtect (remember to remove the hwbp has before). Then press Shift + F9 to run the program until you see signs as follows (try this only for the many, they do not ask him J) :

00139360	00E5DEDA	CALL to VirtualProtect from 00E5DEDA
00139364	00400000	Address = Armadillo.00400000
00139368	00000040	Size = 40 (64.)
0013936C	00000004	NewProtect = PAGE_READWRITE
00139370	0013943C	pOldProtect = 0013943C
00139374	00000000	
00139378	00000000	

After the break in place are such signs, press Ctrl + F9 to return to the main code. Then right click and select Search For> All commands:



Now you will see many orders PUSH 100, the experience is a giang PUSH 100 orders for the first meat. So scroll mouse thẳng to the top, select the first command and press Enter to us here:

00E1264F	CC	int3
00E12650	55	push ebp
00E12651	8BEC	mov ebp, esp
00E12653	83EC 2C	sub esp, 2C
00E12656	833D 00B6E800	cmp dword ptr [E8B6E800], 0
00E1265D	75 59	jnz short 00E126B8
00E1265F	C745 EC 16944B	mov dword ptr [ebp-14], 1C4B9416
00E12666	68 00010000	push 100
00E1266B	E8 86580500	call 00E67EF6
00E12670	83C4 04	add esp, 4

Conduct patch command PUSH EBP into Ret.

00E1264F	CC	int3
00E12650	C3	retn
00E12651	8BEC	mov ebp, esp
00E12653	83EC 2C	sub esp, 2C
00E12656	833D 00B6E800	cmp dword ptr [E8B6E800], 0
00E1265D	75 59	jnz short 00E126B8
00E1265F	C745 EC 16944B	mov dword ptr [ebp-14], 1C4B9416
00E12666	68 00010000	push 100
00E1266B	E8 86580500	call 00E67EF6

OK them the IAT, wind traditional way to OEP. We put in a hwbp CreateThread (delete hwbp set), then press Shift + F9 we stopped at hwbp:

0013F6A0	00E437E5	CALL to CreateThread from 00E437DF
0013F6A4	00000000	pSecurity = NULL
0013F6A8	00000000	StackSize = 0
0013F6AC	00E44750	ThreadFunction = 00E44750
0013F6B0	00000000	pThreadParm = NULL
0013F6B4	00000000	CreationFlags = 0
0013F6B8	0013F6C4	pThreadId = 0013F6C4
0013F6BC	00000002	

Press Ctr + F9 2 times we return to Armadillo code, dragging down the search Call edx second, press F2 to set it at BP:

00E5FA3C	31	push	ecx
00E5FA3D	8B55 F4	mov	edx, dword ptr [ebp-C]
00E5FA40	2B55 DC	sub	edx, dword ptr [ebp-24]
00E5FA43	FFD2	call	edx
00E5FA45	8945 FC	mov	dword ptr [ebp-4], eax
00E5FA48	8B45 FC	mov	eax, dword ptr [ebp-4]
00E5FA4B	5E	pop	esi
00E5FA4C	8BE5	mov	esp, ebp
00E5FA4E	5D	pop	ebp
00E5FA4F	C3	ret	

Press F9 to run it stopped at Call edx, press F7 to trace into one of the OEP in the original program:

00401E6E	73 C6	jmp	short 00401E66
00401E70	AE	scas	byte ptr es:[edi]
00401E71	A3 D9251A8C	mov	dword ptr [8C1A25D9], eax
00401E76	67:4D	dec	ebp
00401E78	2A59 6C	sub	bl, byte ptr [ecx+6C]
00401E7B	0D 42AD874D	or	eax, 4D87AD42
00401E80	42	inc	edx
00401E81	C9	leave	
00401E82	26:1D 2640034D	sbb	eax, 4D034026
00401E88	42	inc	edx

Hehe time we go to find any clean IAT, to find how nh? You fly through the window memory: Alt + M. You remember the address but IAT Start, which is the function IAT Start: 00F04008 7E4254F0 USER32.LoadBitmapA. You will find this function, in memory window to drag it on and press Ctrl + B:

Enter binary string to search for

ASCII

8TB

UNICODE

8TB

HEX +04

F0 54 42 7E

Address of LoadBitMapA

☐ Entire block

☐ Case sensitive

<<

>>

OK

Cancel

Click OK to search until we have the following information:

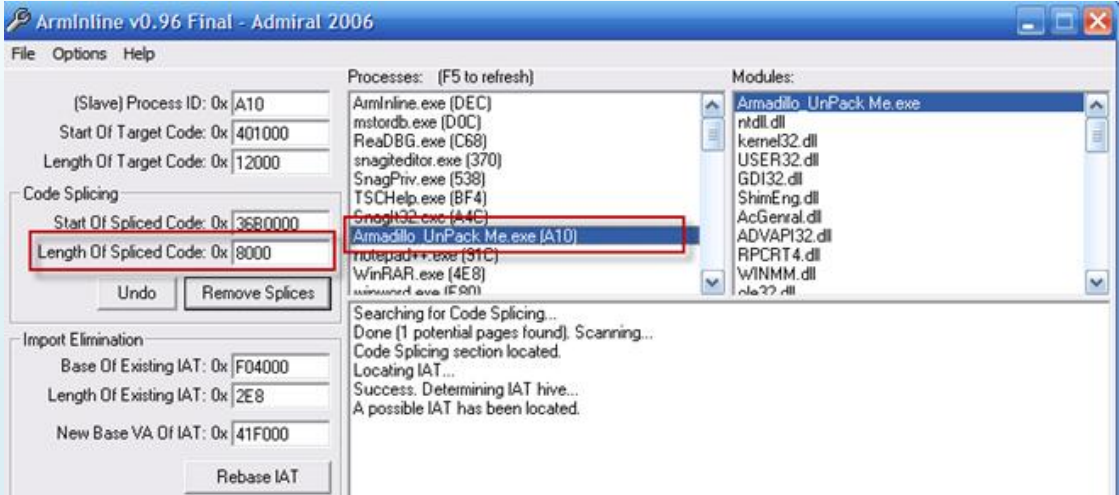
00F94000	000000BC	
00F94004	010C01B2	
00F94008	7E4254F0	USER32.LoadBitmapA
00F9400C	77F15FF0	GDI32.CreateCompatibleDC
00F94010	7C836772	kernel32.GlobalFlags
00F94014	77F273B4	GDI32.Escape

00F945C4	7E42FFC0	USER32.SetPropA
00F945C8	7C80B4CF	kernel32.GetModuleFileNameA
00F945CC	7E430002	USER32.GetPropA
00F945D0	00E3A6D9	
00F945D4	00E3A61F	
00F945D8	7E44EF68	USER32.GetMenuItemID
00F945DC	00000000	
00F945E0	00BC0145	

Very exactly matches the information that we learn about IAT. To locate all of the IAT IAT Start and End then click to select Binary> Binary Copy. Now we return to Olly thng we defeat CopyMem2 and DebugBlocker, locate areas IAT IAT Start and end in the dump window, right click and select Binary> Binary Paste to paste the entire full to IAT (while you save the huddle Binary again because we need for the following):

00F04000	000000BC	
00F04004	010C01C8	
00F04008	7E4254F0	USER32.LoadBitmap
00F0400C	77F15FF0	GDI32.CreateCompatibleDC
00F04010	7C836772	kernel32.GlobalFlags
00F04014	77F27304	GDI32.Escape
00F04018	7E419021	USER32.GetWindowDC
00F0401C	7E4230CE	USER32.GetDlgItem
00F04020	7C91043D	ntdll.RtlFreeHeap
00F04024	00E3A6E5	
00F04028	00E3A6F2	
00F0402C	7E43212B	USER32.GetWindowTextA
00F04030	7E41B72F	USER32.GetParent
00F04034	77F161FF	GDI32.CreateBitmap
00F04038	7E43C6CA	USER32.DrawTextA
00F0403C	7E43E0C7	USER32.GetClassInfoA
00F04040	00E3A550	
00F04044	7C80AC0F	kernel32.SetErrorMode
00F04048	7C81CDDA	kernel32.ExitProcess
00F0404C	00E3A638	
00F04050	7C80CC97	kernel32.SetHandleCount
00F04054	77F16E6F	GDI32.DeleteDC

So once the IAT. Now we will fix Code splicing, time we do not need to use that Olly thng for full IAT more so it is close to the eye and then J. Running the program ArmInline v0.96f (Eng), straighten this fix is the most fast.



The program automatically detect splice area code contains, but we should check the information on length of splicing code for many when they detect non-standard. Press Alt + M and search:

ArmInline v0.96 Final - Admiral 2006

File Options Help

(Slave) Process ID: 0x A10

Start Of Target Code: 0x 401000

Length Of Target Code: 0x 12000

Code Splicing

Start Of Spliced Code: 0x 3680000

Length Of Spliced Code: 0x 18000

Undo Remove Splices

Import Elimination

Base Of Existing IAT: 0x F04000

Length Of Existing IAT: 0x 2E8

New Base VA Of IAT: 0x 41F000

Rebase IAT

Processes: (F5 to refresh)

Armlnline.exe (DEC)

mstordb.exe (D0C)

ReaDBG.exe (C68)

snagiteditor.exe (370)

SnagPriv.exe (538)

TSCHelp.exe (BF4)

Snagit32.exe (A4C)

Armadillo_UnPack Me.exe (A10)

notepad++.exe (91C)

WinRAR.exe (4E8)

winmmr.exe (F80)

Searching for Code Splicing...

Done (1 potential pages found). Scanning...

Code Splicing section located.

Locating IAT...

Success. Determining IAT hive...

A possible IAT has been located.

----- Code Splicing -----

Process memory buffered successfully.

1019 splices repaired.

Splice repairing complete. Patching process...

Patch successful.

Start Of Spliced Code: 0x368000
Length Of Spliced Code: 0x18000

Undo Remove Splices

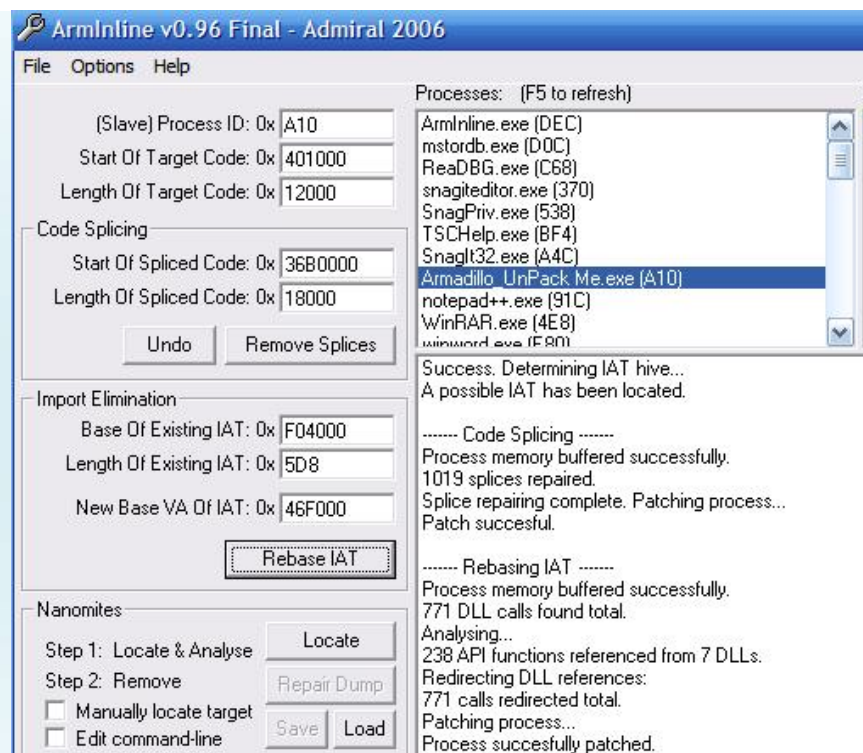
Import Elimination
Base Of Existing IAT: 0xF04000
Length Of Existing IAT: 508
New Base VA Of IAT: 0x46F000

Rebase IAT

code
SFX
data,import:
resources

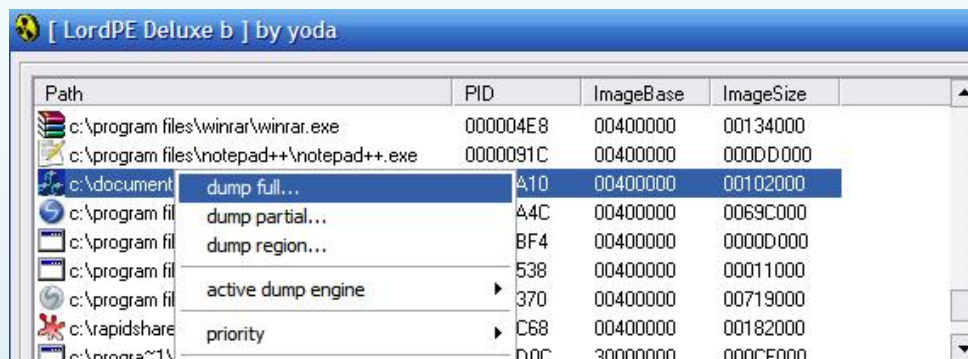
00400000 00001000 Armaddill
00401000 00002000 Armaddill .text
00403000 00005000 Armaddill .rdata
00408000 00007000 Armaddill .data
0040F000 00005000 Armaddill .text1
0046F000 00010000 Armaddill .data
0047F000 00002000 Armaddill .data1
0049F000 00006000 Armaddill .pdata
004FF000 00003000 Armaddill .rsrc
00510000 0000C000
005D0000 00002000
005E0000 00103000
006F0000 0015F000
009F0000 00008000
00A50000 00000000

file:///C:/RCE%20Unpacking%20eBook%20[Translated%20by%20Lith...i]/Manual%20Unpacking%20Armadillo%20v5.42%20Case%20Study.htm (15 of 21) [1/9/2009 9:45:08 LithiumLi]

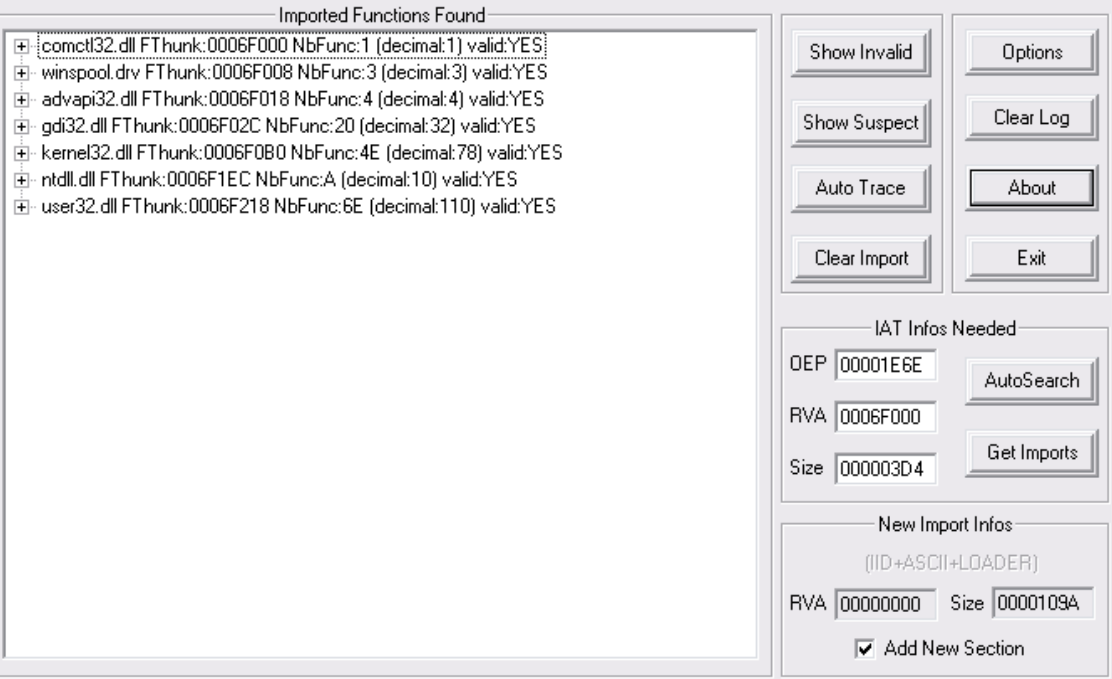


4. Fix DUMPED test file and the file after the fix IAT:

Once done the job on, time is the time we conducted Fix IAT dump file to file dumped. Open up and dump LordPE full:



1.7c ImpREC open up, select the edit process OEP and AutoSearch. Next click Get Imports:

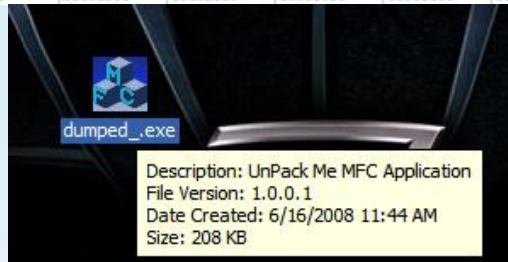


Tolerable without any invalid, it is fortunate. Now it's time to fix dump, select Fix dump dumped and browse to the file. After we have completed fix dumped_.exe file. Run the file to try to see the results:



To file for lessening our use CFF Explorer VII del to reduce the section of Arma:

dumped_.exe									
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00011B8E	00001000	00011B8E	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000453E	00013000	0000453E	00013000	00000000	00000000	0000	0000	40000040
.data	00006348	00018000	00006348	00018000	00000000	00000000	0000	0000	C0000040
.text1	00050000	0001F000	00050000	0001F000	00000000	00000000	0000	0000	E0000020
.adata	00010000	0006F000	00010000	0006F000	00000000	00000000	0000	0000	E0000020
.data1	00020000	0007F000	00020000	0007F000	00000000	00000000	0000	0000	C0000040
.pdata	00060000	0009F000	00060000	0009F000	00000000	00000000	0000	0000	C0000040
.rsrc	00003000	000FF000	00003000	000FF000	00000000	00000000	0000	0000	40000040
.mact	00002000	00102000	00002000	00102000	00000000	00000000	0000	0000	E0000060



So is the MUP is finished, if you follow the above is that you have very good then. But if you encounter any target that also play this style, the only sure water died. The next section we will use the tool available to bypass much faster.

IV. Unpack using the tools available

This section will guide you unpack with Arma tool available with this tool will reduce the load for us to reduce the work must be performed manually very tired. First load the program and select ArmaDetach v1.31 as follows:



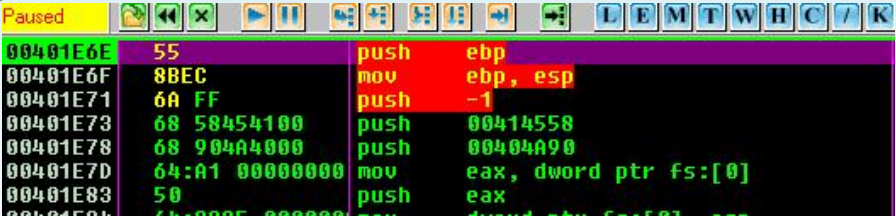
Then drag & drop target by us to:



You go to that section is found Crypto Call: 0x00448A1B. Look not familiar, is the function that we found when implemented by the Manual. Remember child process ID and Original bytes program that provides for us. Now the load and Olly Attach child process to:



F9 and F12, and then edit the original OEP in bytes:



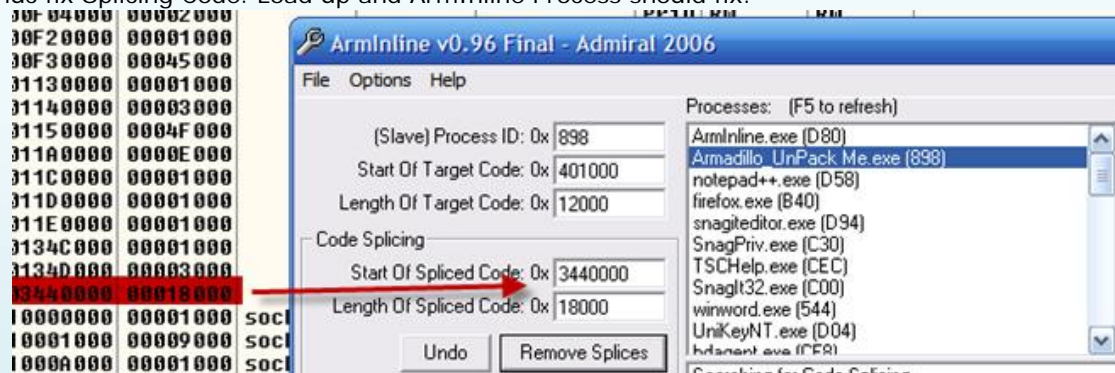
Khúc them and defeat CopyMem2 DebugBlocker very gently but do not have to waste any dew J. Khúc next restore the IAT to target, if you have the binary's IAT that I retain the first step in any time to bring that xài, but if you forget it again ArmaDetach used v1.31, but this time only Debug-blocker and Debug-Blocker IAT resolve it. Then drop into the target. Next use a new Olly to Attach child proc, fix the bytes. Next you run the script *Arma 5.x Magic Fix Call.osc* to fix it Call Magic lost the support of the search. Finally, do as the first to find out Full IAT. I saved the IAT available full and that I always paste into J

```

00F04010 7E419021 USER32.GetWindowDC
00F0401C 7E4230CE USER32.GetDlgItem
00F04020 7C91043D ntdll.RtlFreeHeap
00F04024 00E3A6E5
00F04028 00E3A6F2
00F0402C 7E43212B USER32.GetWindowTextA
00F04030 7E41B72F USER32.GetParent
00F04034 77F161FF GDI32.CreateBitmap
00F04038 7E43C6CA USER32.DrawTextA
00F0403C 7E43EBC7 USER32.GetClassInfoA
00F04040 00E3A550
00F04044 7C80AC0F kernel32.SetErrorMode
00F04048 7C81CDDA kernel32.ExitProcess
00F0404C 00E3A63B
00F04050 7C80CC97 kernel32.SetHandleCount
00F04054 77F16E6F GDI32.DeleteDC
00F04058 00E3A66F
00F0405C 7E41B604 USER32.GetWindowRect
00F04060 00E3A5E9
00F04064 7E44F852 USER32.SetMenuItemBitmaps
00F04068 7E41D7F9 USER32.UpdateWindow
00F0406C 00E3A5A6
00F04070 77DD761B ADVAPI32.RegOpenKeyExA
00F04074 7E42E002 USER32.GetMessageA
00F04078 77F16C0A GDI32.DeleteObject
00F0407C 7C84467D kernel32.SetUnhandledExceptionFilter
00F04080 7C809847 kernel32.CloseHandle
00F04084 00E3A6D9
00F04088 00E3A582
00F0408C 00E3A4D4
00F04090 7E41D8A4 USER32.ShowWindow
00F04094 7C812B9F kernel32.TlsAlloc
00F04098 00E3A5A0
00F0409C 7C81CF5B kernel32.GetEnvironmentStringsA
00F040A0 00E3A4A4
00F040A4 77F15B00 GDI32.SelectObject
00F040A8 7E41BDC8 USER32.ScreenToClient

```

Khuc them rebuild the IAT. Next is khuc fix Splicing Code. Load up and ArmInline Process should fix:

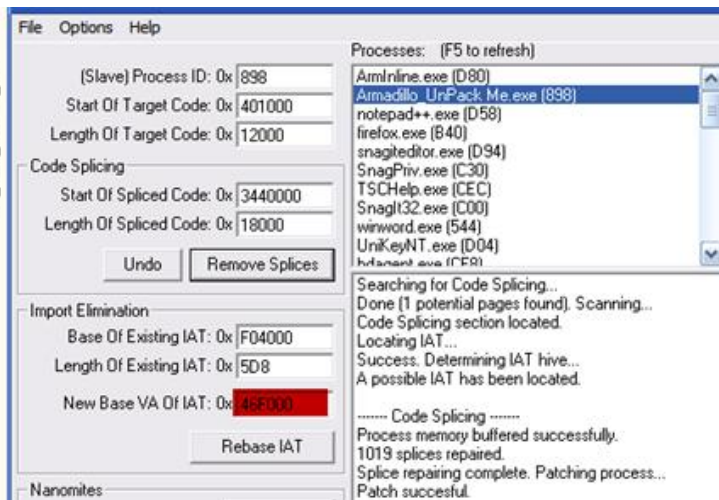


Remove Splices Rebase to complete the IAT are:

```

003E0000 00002000
003F0000 00004000
00400000 00001000
00401000 00012000 Armadill .text
00413000 00005000 Armadill .rdata
00418000 00007000 Armadill .data
0041F000 00050000 Armadill .text1
00461000 00010000 Armadill .adata
0047F000 00020000 Armadill .data1
0049F000 00060000 Armadill .pdata
004FF000 00003000 Armadill .rsrc
00510000 00007000
005D0000 00002000
005E0000 00103000
006F0000 000C0000
009F0000 00008000
00AF0000 00049000
00B40000 00001000
00BC0000 00002000
00BD0000 0000A000
00BE0000 0000A000
00EE0000 00002000
00BF0000 00001000
00C00000 00008000
00C40000 0000C000

```



Finally a full dump with LordPE and fix IAT with ImpRec once again! Faster step in the many and J. Dumped_ To file size of the small del reduce the section of the Armadillo is complete, try the test file was found fix mượt run as a way to do Manual.

V. Conclusion

Finally, the complete message, takes 25 pages for a topic has been said a lot of hope you're not bored. This article I wrote for the purpose of storage and only the basic points to unpack Armadillo but I do not go more deeply into each section of these have too much material about the no. Them more recorded knowledge you are with me it is a pleasure, a feeling for their own Refresh your knowledge of minh.Rat thank him and you for taking your valuable time to read this document.

PS: This document is only a reference, the author is not responsible if the reader uses it to any goal.

Best Regards

_ [Kienmanowar] _



--+ +--==[Greatz thanks to]==--+ +--

My family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHQCrk, the_Lighthouse, Merc, Hoadongnoi, Nini ... all REA's members, TQN, HacNho, RongChauA, Deux, tlandn, light.phoenix, dqtl, ARTEAM all my friend, and you.

Thanks to --+ +--==[]==--+ +--

iamidiot, WhyNotBar, trickyboy, dzungltvn, takada, hurt_heart, haule_nth, hytkl, Moth, XIANUA, nhc1987, v. Oxdie. v.. You have contributed greatly to the REA. Hope you will continue to promote J

I want to thank **Teddy Roggers** for his great site, Reversing.be folks (especially **haggar**), Arteam folks (**Shub-Nigurrath**, **MaDMAn_H3rCuL3s**) and all folks on crackmes.de, thank to all members of **unpack.cn** (especially **fly** and **linhanshi**). Great thanks to **lena151** (I like your tutorials). And finally, thanks to **Ricardo NARVAJA** and all members on **CRACKSLATINOS**.

>>>> If you have any suggestions, comments or corrections email me: **kienmanowar**

[at] reaonline.net

2008

[MUP Armadillo v6.00]



Why Not Bar

www.reaonline.net

7/14/2008

I. Introduction

ArmaDillo today, to the new Version 6.0 ... With the new tools have been put **Armageddon_v1.3.3** Blacklist and the course will do is unpack. After MUP Done should write to the tut is to store for later the party forgotten that draw readers. In this tut, beyond the present way unpack Armadillo V6.0, he also presented to the "**Retouch**" the dump file after unpack it look better ... Pro only do work

II. The Tools use

Posts using the following:

1. **READBG 1.1 (version OllyDBG MOD) or OllyIce (ver 2008.1.1)**
2. **Armadillo Find Protected V1.8**
3. **ArmaDetach v1.31**
4. **Fixed Armlnline v0.96f by Trickky Boy**
5. **Task LordPE or Explorer.**
6. **CHimpREC or 1.0 ImportRec v1.7c Final.**
7. **CFF Explorer.**
8. **0.94 PEiD and RDG Packer Detector v0.6.5**

III. Manual unpack Armadillo v4.64

1. Seeking information on Target.

Armadillo Find Protect v1.8 check to see the pack in this version **Armadillo** how much and which protect the Option. Open Armadillo Find Protect v1.8, then **drag & drop** target to be one of the following information:

12-07-2008 00:36:35 <----->

C: \ Documents and Settings \ REAteam \ Desktop \ Testlab \ UnPackMe_Armadillo_v6 \ UnPackMe.exe

- Protected Armadillo

Protection system (Professional)

- **<Protection Options>**

Debug-Blocker

CopyMem-II

Enable elimination Import Table

Enable Strategic Code Splicing

Enable Nanomites Processing

Enable Memory-Patching Protections

- Key <Backup Options>

Variable Backup Keys

- <Compression Options>

Best / Slowest Compression

- <Other Options>

Allow Only One Copy

- **Version 6.00 08-07-2008**

- Elapsed Time 00h 00m 07s 578ms

You should add Add **Signature.txt** by **Armadillo Find Protect v1.8** to identify properly Version

"4872AE00 Version 6.00 08-07-2008"

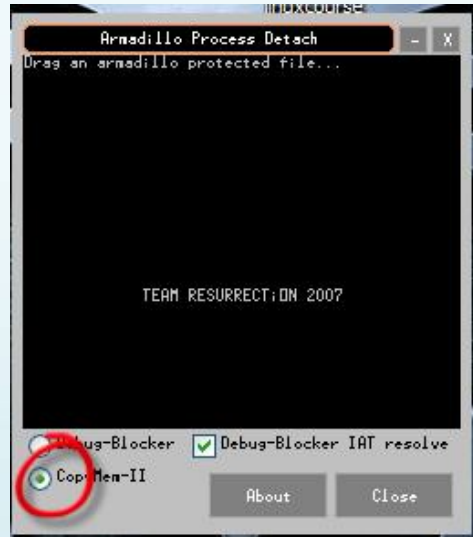
Fairness, full Protect the essence of the **Armadillo**. Heard in this Custom Build ...

2. CopyMemII defeat, DebugBlocker

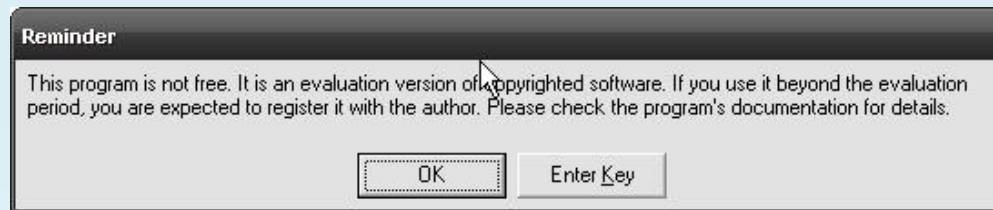
If he does want children **Manual Detach** or find out more you can see tut **MUP Armadillo v5.42 Case Study** by **Kien** or read the old tut on REA. So in this Tutor She used the tools

available for fast results that same

First load the program and select **ArmaDetach v1.31** as follows:



Then **drag & drop** target by us to:



Click OK



Now the load and Olly Attach Select **Child Process ID** for accuracy, press **F9** and **F12**, then based on the Info **ArmaDetach** edit by the **EP** in bytes:

00402DA4	55	PUSH EBP	
00402DA5	8BEC	MOV EBP,ESP	
00402DA7	6A FF	PUSH -1	
00402DA9	68 88484000	PUSH UnPackMe.00404888	
00402DAE	68 2A2F4000	PUSH UnPackMe.00402F2A	JMP to msvcrt._except_handler3; SE
00402DB3	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00402DB9	50	PUSH EAX	
00402DBA	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00402DC1	83EC 68	SUB ESP,68	
00402DC4	53	PUSH EBX	
00402DC5	56	PUSH ESI	
00402DC6	57	PUSH EDI	
00402DC7	8965 E8	MOV [LOCAL.6],ESP	
00402DCA	33DB	XOR EBX,EBX	
00402DCC	895D FC	MOV [LOCAL.1],EBX	
00402DCF	6A 02	PUSH 2	
00402DD1	FF15 0C547201	CALL NEAR DWORD PTR DS:[172540C]	msvcrt.__set_app_type
00402DD7	59	POP ECX	
00402DD8	830D 50634000	FOR DWORD PTR DS:[406350],FFFFFFFF	
00402DDF	830D 54634000	FOR DWORD PTR DS:[406354],FFFFFFFF	
00402DE6	FF15 BC517201	CALL NEAR DWORD PTR DS:[17251BC]	msvcrt.__p_fmode
00402DEC	8B0D 44634000	MOV ECX,DWORD PTR DS:[406344]	
00402DF2	8908	MOV DWORD PTR DS:[EAX],ECX	
00402DF4	FF15 18547201	CALL NEAR DWORD PTR DS:[1725418]	msvcrt.__p_commode

OK! defeat and was **CopyMemII DebugBlocker**. To wait is to find the peace of the **IAT** to **Full Paste**. We need to determine the **IAT Start, End** and **IAT IAT Size**. From the command Call **EP** below, select it, right click and select as follows:

00402DAE . 68 2A2F4000 PUSH UnPackMe.00402F2A
00402DB3 . 64:A1 00000000 MOV EAX,DWORD PTR FS:[0]
00402DB9 . 50 PUSH EAX
00402DBA . 64:8925 00000000 MOV DWORD PTR FS:[0],ESP
00402DC1 . 83EC 68 SUB ESP,68
00402DC4 . 53 PUSH EBX
00402DC5 . 56 PUSH ESI
00402DC6 . 57 PUSH EDI
00402DC7 . 8965 E8 MOV [LOCAL.6],ESP
00402DCA . 33DB XOR EBX,EBX
00402DCC . 895D FC MOV [LOCAL.1],EBX
00402DCF . 6A 02 PUSH 2
00402DD1 . FF15 0C547201 CALL NEAR DWORD PTR D
00402DD7 . 59 POP ECX
00402DD8 . 830D 50634000 FOR DWORD PTR DS:[4063
00402DDF . 830D 54634000 FOR DWORD PTR DS:[4063
00402DE6 . FF15 BC517201 CALL NEAR DWORD PTR D
00402DEC . 8B0D 44634000 MOV ECX,DWORD PTR DS:
00402DF2 . 8908 MOV DWORD PTR DS:[EAX
00402DF4 . FF15 18547201 CALL NEAR DWORD PTR D
00402DFA . 8B0D 40634000 MOV ECX,DWORD PTR DS:
00402E00 . 8908 MOV DWORD PTR DS:[EAX
00402E02 . A1 2C527201 MOV EAX,DWORD PTR DS:
00402E07 . 8B00 MOV EAX,DWORD PTR DS:

Backup
Copy
Binary
Assemble
Label
Comment
Breakpoint
Hit trace
Run trace
Follow
New origin here
Go to
Thread
Follow in Dump
View call tree
Search for

Space
:
;

Enter
Ctrl+Gray*

Ctrl+K

set_app_type

p__fmode

p__commode

Selection
Memory address

At window dump scroll mouse over we have been **IAT start:**

Address	Value	Comment
01724F4C	00000000	
01724F50	002400A1	
01724F54	010C01A2	
01724F58	77F3AD80	GDI32.CreateFontA ==>IAT Start
01724F5C	015CC5D8	
01724F60	015CC665	
01724F64	73E6CBFA	MFC42.#3738
01724F68	73DD2415	MFC42.#567
01724F6C	73DE1CF0	MFC42.#1089
01724F70	015CC778	
01724F74	73DD6879	MFC42.#641

Scroll down we have **IAT End:**

Address	Value	Comment
01725434	015CC76E	
01725438	015CC6CF	
0172543C	015CC760	
01725440	73DD5EF1	MFC42.#2554
01725444	73DD3CFA	MFC42.#3874
01725448	73DE092D	MFC42.#5875
0172544C	77F1C83B	GDI32.Ellipse
01725450	77C018BA	VERSION.VerQueryValueA
01725454	00000000	==>IAT End
01725458	00A10004	UnPackMe.00000001
0172545C	010C01C3	

How summation lay gently we have **IAT Size**

IAT IAT Size = End - Start = IAT 0x4FC

For Full **IAT**, open again **ArmaDetach v1.31** but this time only select **Debug-blocker**. Then drop the target.



Next use a new Olly Attach to **Process Child ID** (remember to choose the exact) and fix the bytes.

00447000	60	PUSHAD	
00447001	E8 00000000	CALL UnPackMe.00447006	
00447006	5D	POP EBP	
00447007	50	PUSH EAX	
00447008	51	PUSH ECX	
00447009	0FCA	BSWAP EDX	
0044700B	F7D2	NOT EDX	
0044700D	9C	PUSHFD	
0044700E	F7D2	NOT EDX	
00447010	0FCA	BSWAP EDX	
00447012	EB 0F	JMP SHORT UnPackMe.00447023	
00447014	B9 EBF8EB	MOV ECX,EBB80FEB	
00447019	07	POP ES	
0044701A	B9 EBF90EB	MOV ECX,EB900FEB	
0044701F	00FD	OP CU BU	

Modification of segment register

Continue pressing **Alt + F1**, **BP CreateFileMappingA**, **Shift + F9**, and will stop here

Registers (FPU)

EAX	01661CD4
ECX	00000004
EDX	00000000
EBX	00000000
ESP	0012E204
EBP	0012E640
ESI	BC350C7C
EDI	88CFD9F2
EIP	7C80946C kernel32.CreateFileMap
C 0	ES 0023 32bit 0(FFFFFFFF)
P 0	CS 001B 32bit 0(FFFFFFFF)
A 0	SS 0023 32bit 0(FFFFFFFF)
Z 0	DS 0023 32bit 0(FFFFFFFF)
S 0	FS 003B 32bit 7FFDF000(FFF)

Address	Value	Comment
00457000	77F1CE55	GDI32.CreateDCA
00457004	77F1B52C	GDI32.CreateDIBitmap
00457008	77F15E10	GDI32.CreateCompatibleDC
0045700C	77F159A0	GDI32.SelectObject
00457010	77F182DE	GDI32.SelectPalette
00457014	77F1BD89	GDI32.RealizePalette
00457018	77F16DC0	GDI32.BitBlt
0045701C	77F16CA6	GDI32.DeleteDC
00457020	77F16A3B	GDI32.DeleteObject
00457024	77F18DD7	GDI32.CreatePalette

1 time unpack Armadillo v5.xx and 1 draw is the Opcode can quickly get the Full IAT performance with 98% account. Press **Alt + M**, **Ctrl + B** search the following **Opcode**

"55 8B EC 83 EC 2C 83 3D"

00DDCD10 55 push ebp

00DDCD11 8BEC mov ebp, esp

00DDCD13 83EC 2C sub esp, 2C

00DDCD16 833D 7CDCE000 0> Cmp dword ptr [E0DC7C], 0

00DDCD1D 75 59 jnz short 00DDCD78

Enter binary string to search for

ASCII

UNICODE

HEX +08

☐ Entire block

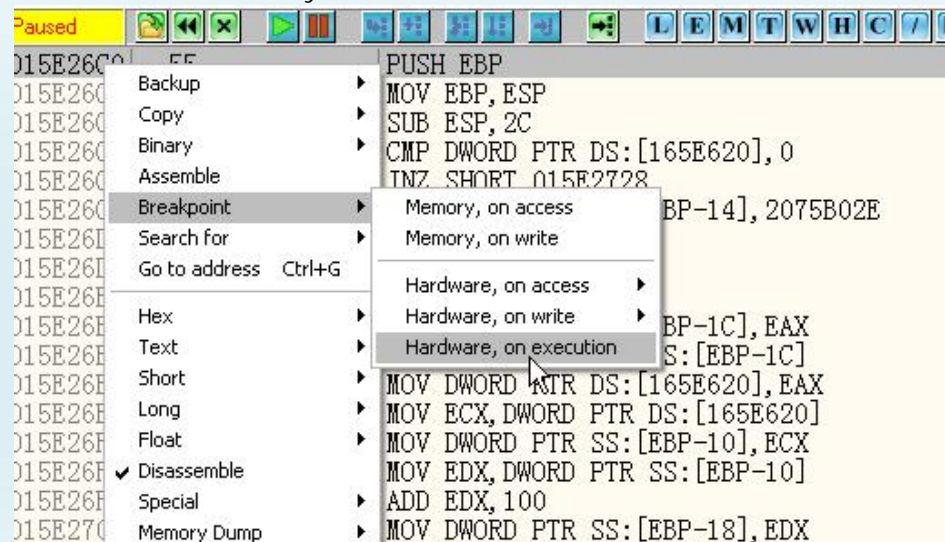
☐ Case sensitive

<< >>

OK Cancel

015E26C0	55	8B	EC	83	EC	2C	83	3D	20	E6	65	01	00	75	59	C7	U? 媛波, ? 鐳 r.uY
015E26D0	45	EC	2E	B0	75	20	68	00	01	00	00	E8	A6	85	05	00	E? 昂 h. r. . 瑕?
015E26E0	83	C4	04	89	45	E4	8B	45	E4	A3	20	E6	65	01	8B	0D	煥+ 境鏡E列 鐳 r?
015E26F0	20	E6	65	01	89	4D	F0	8B	55	F0	81	C2	00	01	00	00	鐳 r 容饗U饋? r. .
015E2700	89	55	E8	8B	45	F0	3B	45	E8	73	1D	68	00	01	00	00	塢鐳E?E鐳h. r. .
015E2710	8D	4D	EC	E8	88	00	00	00	8B	4D	F0	88	01	8B	55	F0	峇寔?.. 菱饒 煥
015E2720	83	C2	01	89	55	F0	EB	DB	8B	45	0C	89	45	F4	8B	4D	煥 r 境蹕E. 境蹕M
015E2730	F4	03	4D	10	89	4D	F8	8B	15	20	E6	65	01	89	55	FC	?M+ 塔鵠+ 鐳 r 塢
015E2740	8B	45	F4	3B	45	F8	73	3C	8B	4D	08	0F	BE	11	8B	45	煥?E鵠< 菱Q?煥
015E2750	FC	0F	BE	08	33	D1	8B	45	F4	88	10	8B	4D	FC	83	C1	??? 鐳E鵠+ 菱鵠
015E2760	01	89	4D	FC	8B	55	08	83	C2	01	89	55	08	8B	45	F4	r 塔鵠U 煥 r 塢Q煥
015E2770	0F	BE	08	85	C9	75	02	EB	0B	8B	55	F4	83	C2	01	89	Q? 央u?煥鵠?
015E2780	55	F4	EB	BC	8B	45	F4	2B	45	F8	F7	D8	1B	C0	23	45	U 基紘E?E ??E
015E2790	0C	8B	E5	5D	C3	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	. 煥J 锰烫烫烫烫烫

And with only 1 address this, if someone do it for information on the test are soft Pack with Arma v5.xx **up. Disassemble** out for the **set** and **1 HWBP** there always



bc CreateFileMappingA, press **Shift + F9** until **NAG Reminder** up, click OK Play stopped at **HWBP Set**

Address	Disassembly	Comment
015E26C0	55	PUSH EBP
015E26C1	8BEC	MOV EBP, ESP
015E26C3	83EC 2C	SUB ESP, 2C
015E26C6	833D 20E66501	CMP DWORD PTR DS:[165E620], 0
015E26CD	75 59	JNZ SHORT 015E2728
015E26CF	C745 EC 2EB0752	MOV DWORD PTR SS:[EBP-14], 2075B02E
015E26D6	68 00010000	PUSH 100
015E26DB	E8 A6850500	CALL 0163AC86
015E26E0	83C4 04	ADD ESP, 4
015E26E3	8945 E4	MOV DWORD PTR SS:[EBP-1C], EAX
015E26E6	8B45 E4	MOV EAX, DWORD PTR SS:[EBP-1C]
015E26E9	A3 20E66501	MOV DWORD PTR DS:[165E620], EAX
015E26EE	8B0D 20E66501	MOV ECX, DWORD PTR DS:[165E620]
015E26F4	894D F0	MOV DWORD PTR SS:[EBP-10], ECX
015E26F7	8B55 F0	MOV EDI, DWORD PTR SS:[EBP-10]

Ctrl + E to patch C3

Address	Disassembly	Comment
015E26C0	C3	RETN
015E26C1	8BEC	MOV EBP, ESP
015E26C3	83EC 2C	SUB ESP, 2C
015E26C6	833D 20E66501	CMP DWORD PTR DS:[165E620], 0
015E26CD	75 59	JNZ SHORT 015E2728
015E26CF	C745 EC 2EB0752	MOV DWORD PTR SS:[EBP-14], 2075B02E
015E26D6	68 00010000	PUSH 100
015E26DB	E8 A6850500	CALL 0163AC86
015E26E0	83C4 04	ADD ESP, 4
015E26E3	8945 E4	MOV DWORD PTR SS:[EBP-1C], EAX
015E26E6	8B45 E4	MOV EAX, DWORD PTR SS:[EBP-1C]
015E26E9	A3 20E66501	MOV DWORD PTR DS:[165E620], EAX
015E26EE	8B0D 20E66501	MOV ECX, DWORD PTR DS:[165E620]
015E26F4	894D F0	MOV DWORD PTR SS:[EBP-10], ECX
015E26F7	8B55 F0	MOV EDI, DWORD PTR SS:[EBP-10]

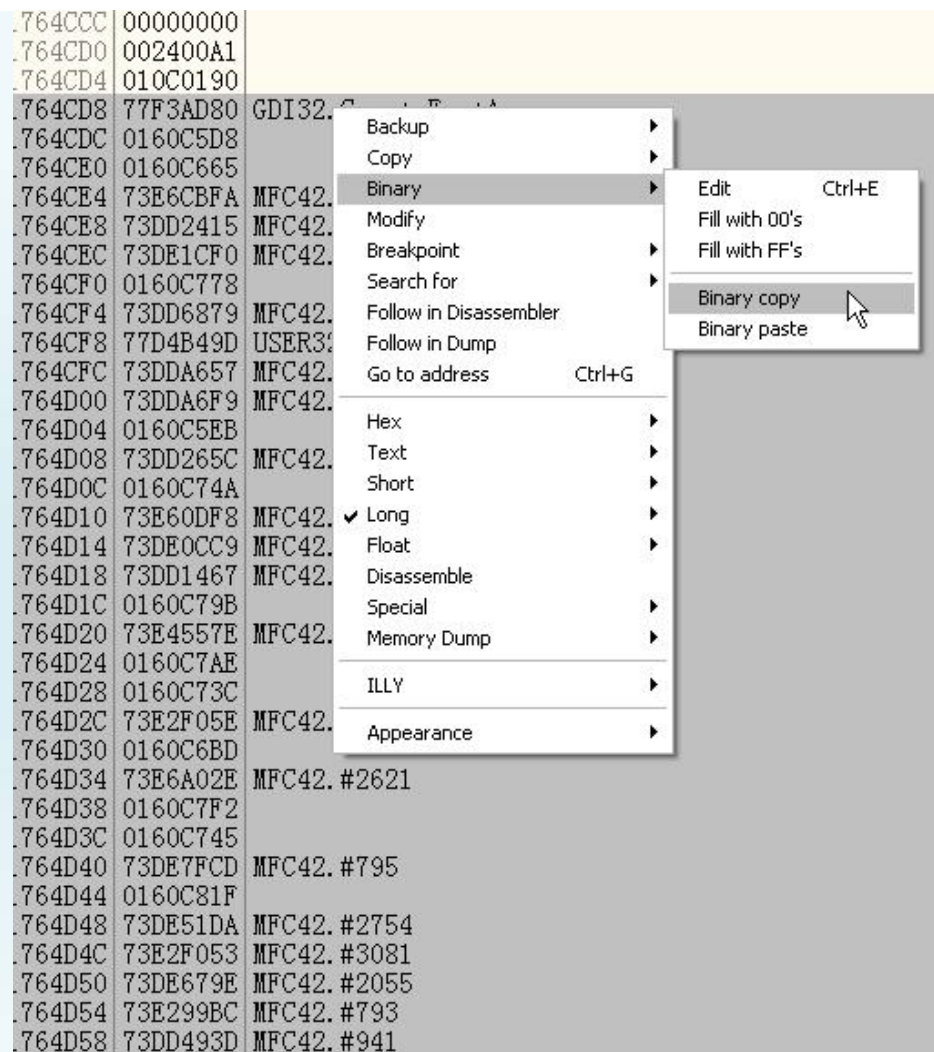
Remove HWBP, Alt + M and Memory Set BP under the same image

02C0000	00041000				Map	R	R	\Device\N
0310000	00006000				Map	R	R	\Device\N
0320000	00005000				Map	R E	R E	
03E0000	00002000				Map	R E	R E	
03F0000	00008000				Priv	RW	RW	
0400000	00001000	UnPackMe		PE header	Imag	R	RWE	
0401000	00003000	UnPackMe	.kadx	code	Imag	R	RWE	
0404000	00002000	UnPackMe	.gl	Actualize			RWE	
0406000	00001000	UnPackMe	.fr	View in Disassembler	Enter		RWE	
0407000	00040000	UnPackMe	.zk	Dump in CPU			RWE	
0447000	00010000	UnPackMe	.qv	Dump			RWE	
0457000	00020000	UnPackMe	.go	Search	Ctrl+B		RWE	
0477000	00490000	UnPackMe	.sz	Search next	Ctrl+L		RWE	
0907000	0046C000	UnPackMe	.jy				RWE	
0D80000	00103000			Set break-on-access	F2		R	
0E90000	000A5000			Set memory breakpoint on access			R E	
1190000	00003000			Set memory breakpoint on write			R	\Device\N
11A0000	00010000			Set access			RW	
11B0000	00001000						RW	
11C0000	00001000			Allocate Memory			RW	
11D0000	00001000			Free Memory			R	
11E0000	00001000			Zero Memory			RWE	
11F0000	00001000			Dump Memory-Area			RWE	
1200000	00001000			Load dumped memory			RWE	
1210000	00004000						RW	
1220000	0004A000			Copy to clipboard			R	
1270000	00001000			Sort by			R	
1280000	00004000			Appearance			dm	

Press **Shift + F9** to **OEP** is the ball, but i do for purposes of our search is the only IAT Full Crash ...
If there is not anything to nho ...

00402DA4	- 74 FE	JE SHORT UnPackMe.00402DA4	==>OEP
00402DA6	11E0	ADC EAX, ESP	
00402DA8	DE1D 75C26175	FICOMP WORD PTR DS:[7561C275]	
00402DAE	95	XCHG EAX, EBP	
00402DAF	A0 0E35FDEE	MOV AL, BYTE PTR DS:[EEFD350E]	
00402DB4	8075 FD 8A	XOR BYTE PTR SS:[EBP-3], 8A	
00402DB8	2125 99030475	AND DWORD PTR DS:[75040399], ESP	
00402DBE	FD	STD	
00402DBF	8A21	MOV AH, BYTE PTR DS:[ECX]	
00402DC1	F611	NOT BYTE PTR DS:[ECX]	
00402DC3	E2 72	LOOPD SHORT UnPackMe.00402E37	
00402DC5	23AA 03449DCE	AND EBP, DWORD PTR DS:[EDX+CE9D4403]	
00402DCB	51	PUSH ECX	
00402DCC	A8 28	TEST AL, 28	
00402DCF	01E0	ADD EAX, EBP	

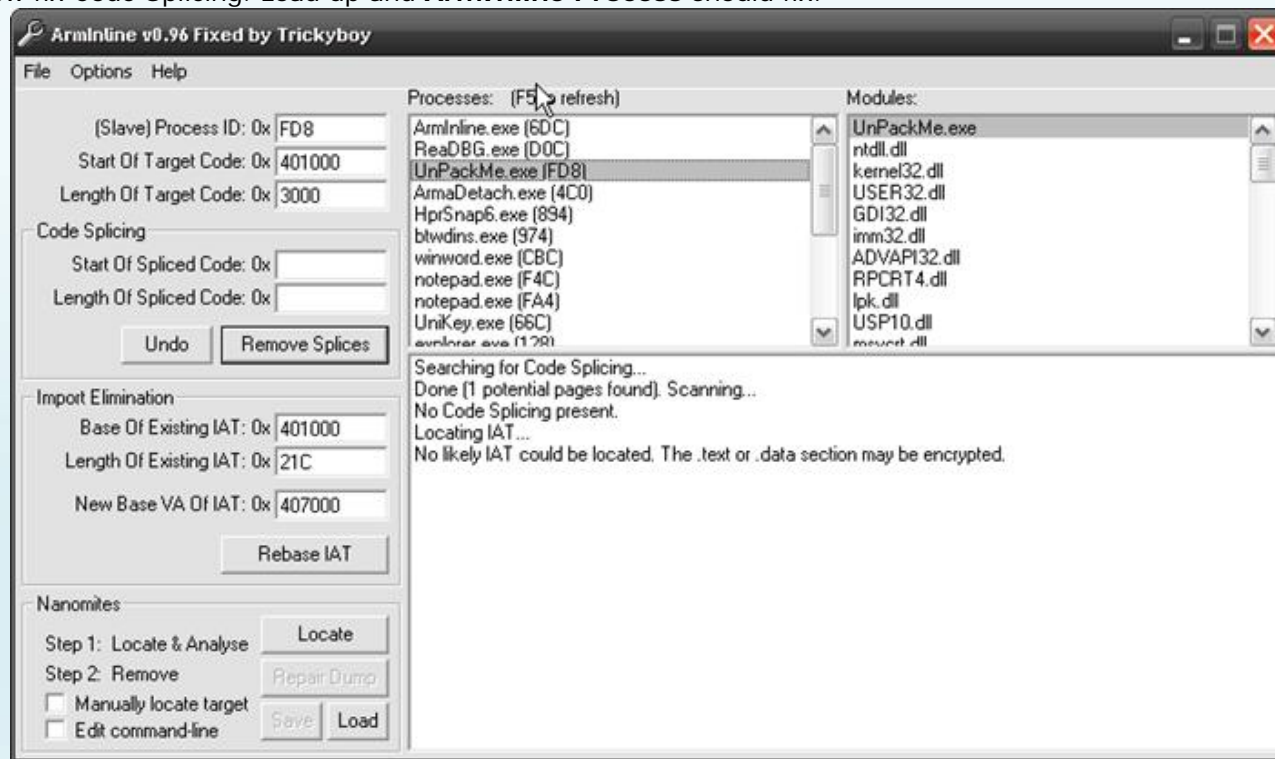
Now is the time for full IAT, press **Alt + M** and roll the mouse on the top, then press **Ctrl + B**
to enter: **80 AD F3 77** Address by **GDI32.CreateFontA (IAT Start)**. Click OK, after seeing
Check IAT information that we find at the same match we COPY Full IAT block this, right click
and select:



Included Full IAT then, is the Olly IAT used to find and return to the first Olly. One of the need to fix the IAT and **Binary Paste** to see us have more API functions but at first i have if we are successful in Fix IAT

Address	Value	Comment
01725370	0160C6CA	
01725374	0160C76C	
01725378	0160C7B3	
0172537C	7C80AA66	kernel32.FreeLibrary
01725380	0160C68C	
01725384	77D4E2AE	USER32.SendMessageA
01725388	0160C75D	
0172538C	77D48C06	USER32.SetTimer
01725390	77C39D67	msvcrt._initterm
01725394	73E2F314	MFC42.#5302
01725398	7C80B529	kernel32.GetModuleHandleA
0172539C	77C1EEEE	msvcrt.__getmainargs
017253A0	73DDCF2B	MFC42.#1576
017253A4	77C39E9A	msvcrt._exit
017253A8	0160C658	
017253AC	77C34E51	msvcrt.__dllonexit
017253B0	7C801EEE	kernel32.GetStartupInfoA
017253B4	73DD6BE1	MFC42.#2446
017253B8	73DD1265	MFC42.#4079
017253BC	73E2F314	MFC42.#5302

So complete rebuild IAT. Now fix Code Splicing. Load up and **Armlnline Process** should fix:

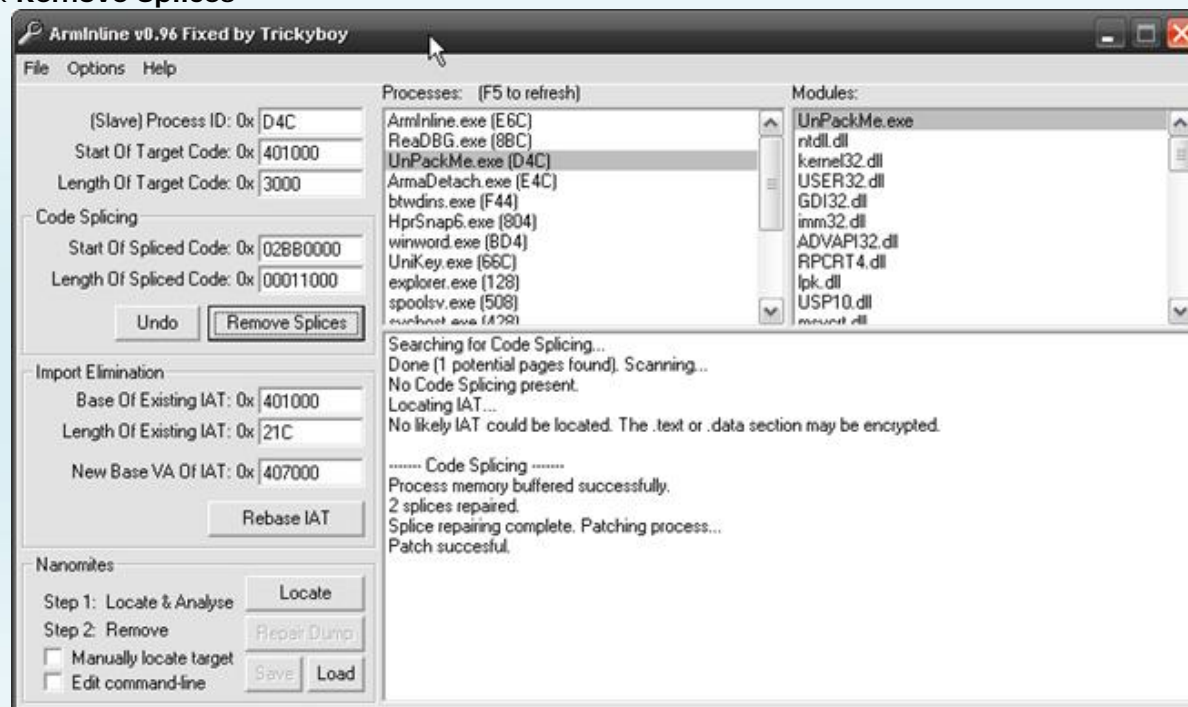


With this target ko Tool automatically find the address **Code Splicing** as always so we should have to go find ... **Alt + M** and scroll down to see the mouse signs **Splicing** of the **Code**, each and every time they do it will address Others should note that you do not block the Arab world crash that affected Asia

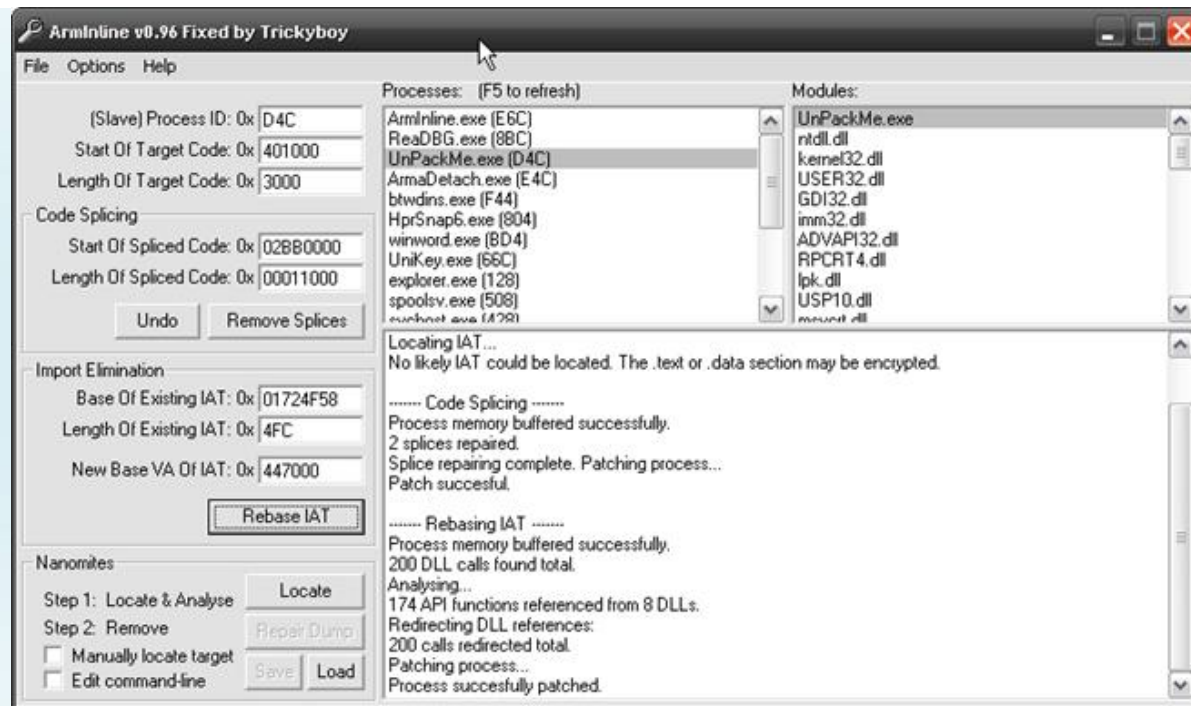
01931000	00001000			stack of the	Priv	RW	Guar	RW	
01A3E000	00001000				Priv	RW	Guar	RW	
01A3F000	00001000			stack of the	Priv	RW	Guar	RW	
01A40000	00001000				Priv	RW		RW	
01A60000	00044000				Priv	RW		RW	
01C40000	00003000				Priv	RW		RW	
01F3D000	00001000				Priv	RW	Guar	RW	
01F3E000	00002000			stack of the	Priv	RW	Guar	RW	
02BB0000	00011000				Priv	R	E	RWE	
10000000	00001000	HSTxtCap		PE header	Imag	R		RWE	
10001000	0000F000	HSTxtCap	.text	code	Imag	R		RWE	
10010000	00003000	HSTxtCap	.rdata	imports, exports	Imag	R		RWE	
10013000	00003000	HSTxtCap	.data	data	Imag	R		RWE	
10016000	00082000	HSTxtCap	.shared		Imag	R		RWE	

Code Splicing

OK, back to fill **Armline** click **Remove Splices**



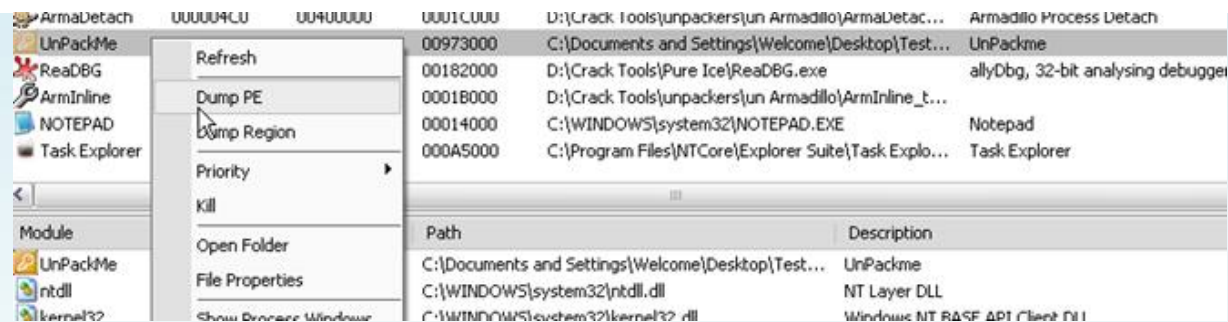
Time to **Import Table elimination**, complete address and **IAT IAT Start to End Armline**, click **Rebase IAT**



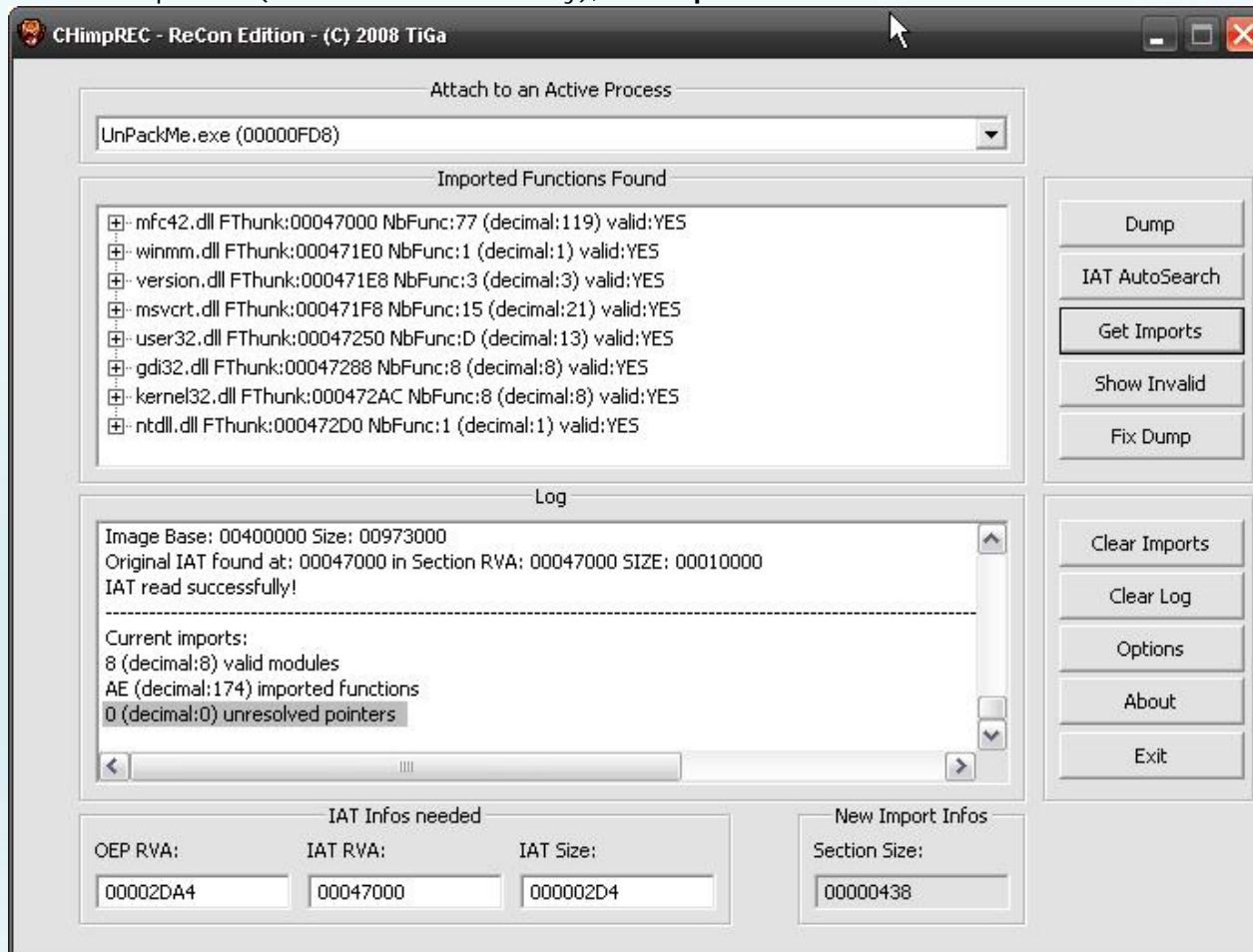
In Olly has a change

00402D9D	L	C3	RETN	
00402D9E	\$-	FF25 40724400	JMP NEAR DWORD PTR DS:[447240]	msvcrt._ftol
00402DA4	.	55	PUSH EBP	
00402DA5	.	8BEC	MOV EBP, ESP	
00402DA7	.	6A FF	PUSH -1	
00402DA9	.	68 88484000	PUSH UnPackMe.00404888	
00402DAE	.	68 2A2F4000	PUSH UnPackMe.00402F2A	JMP to msvcrt._except_handler3; SE
00402DB3	.	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
00402DB9	.	50	PUSH EAX	
00402DBA	.	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
00402DC1	.	83EC 68	SUB ESP, 68	
00402DC4	.	53	PUSH EBX	
00402DC5	.	56	PUSH ESI	
00402DC6	.	57	PUSH EDI	
00402DC7	.	8965 E8	MOV [LOCAL.6], ESP	
00402DCA	.	33DB	XOR EBX, EBX	
00402DCC	.	895D FC	MOV [LOCAL.1], EBX	
00402DCF	.	6A 02	PUSH 2	
00402DD1	.	FF15 18724400	CALL NEAR DWORD PTR DS:[447218]	msvcrt.__set_app_type
00402DD7	.	59	POP ECX	
00402DD8	.	830D 50634000	OR DWORD PTR DS:[406350], FFFFFFFF	
00402DDF	.	830D 54634000	OR DWORD PTR DS:[406354], FFFFFFFF	
00402DE6	.	FF15 00724400	CALL NEAR DWORD PTR DS:[447200]	msvcrt.__p__fmode
00402DEC	.	8B0D 44634000	MOV ECX, DWORD PTR DS:[406344]	
00402DF2	.	8908	MOV DWORD PTR DS:[EAX], ECX	

OK, open up **Task Explorer** to dump and save the file:



Continue to open up **CHimpREC** fix dumped file (cut remove invalid if any), xài **ImportREC** also:

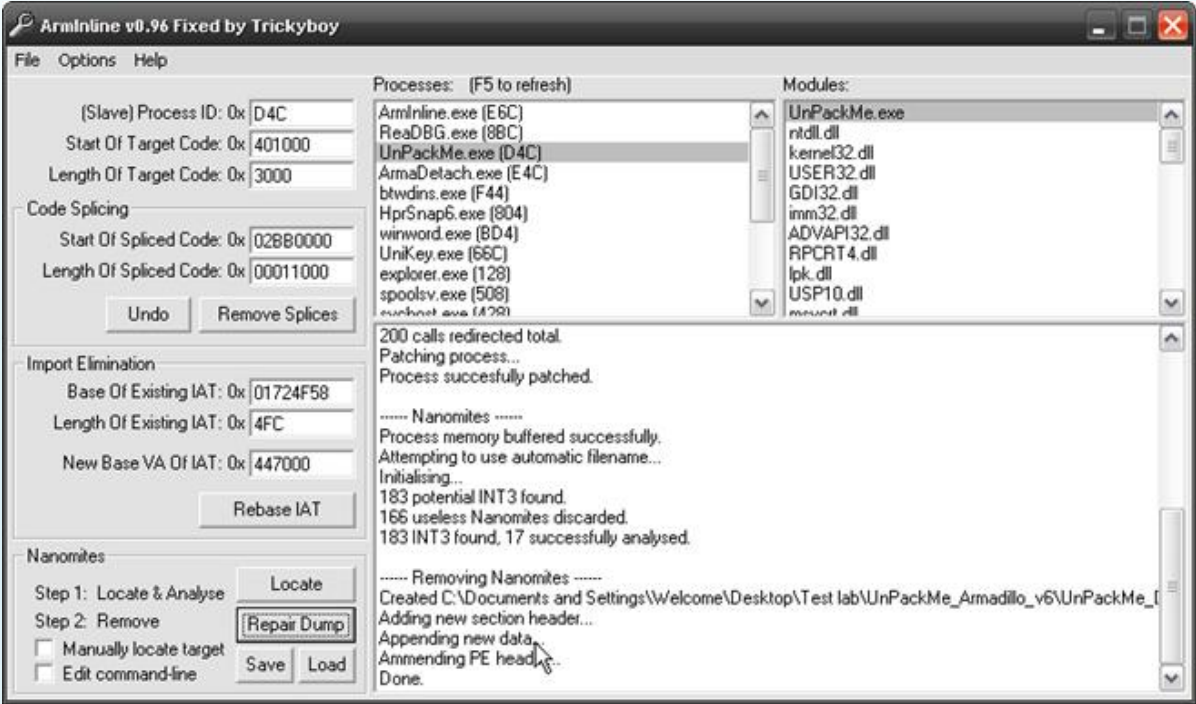


After the fix is done, run another test ... another crash in line because **Nanomites Processing** muh enough time ... do ... I invited him over the 3, then to lower resolution.

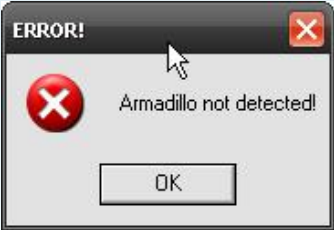
3. Fix Nanomites

To **Fix Nanomites** they use **Armlnline fixed by TrickyBoy** ... 1 tool has been fixed by the insurance Trick's songs. Tool can fix the original crash that when. Back **Armlnline** running, click **locate**, it does NAG Reminder, click OK and select **Repair dump** dump file to **repair**. If

Ok **Repair** will receive the following information:



It's time to try Run ... File dump ăăă country flush the NAG in line ...



This has 2 months ... Fix **Fixed** Load **File Nano** to **Olly**, press **F9** it loi the **NAG Reminder**, press **F12**, **Alt + K**, Stack method that is traditional

Address	Stack	Procedure / arguments	Called from	Frame
0012E8B0	77D493F5	Includes ntdll.KiFastSystemCallRet	USER32. 77D493F3	0012E8E4
0012E8B4	77D6EA24	USER32.WaitMessage	USER32. 77D6EA1F	0012E8E4
0012E8E8	77D5688A	USER32. 77D6E895	USER32. 77D56885	0012E8E4
0012E910	77D6B7C5	USER32. 77D567D4	USER32. 77D6B7C0	0012E90C
0012EBD0	77D6B12B	USER32. SoftModalMessageBox	USER32. 77D6B126	0012EBCC
0012ED20	77D95FDF	USER32. 77D6AFB6	USER32. 77D95FDA	0012ED1C
0012ED78	77D96084	USER32. MessageBoxTimeoutW	USER32. 77D9607F	0012ED74
0012EDAC	77D80598	? USER32. MessageBoxTimeoutA	USER32. 77D80593	0012EDA8
0012EDCC	77D80550	? USER32. MessageBoxExA	USER32. 77D8054B	0012EDC8
0012EDD0	00000000	hOwner = NULL		
0012EDD4	004060AC	Text = "Armadillo not detected!"		
0012EDD8	004060C4	Title = "ERROR!"		
0012EDDC	00000010	Style = MB_OK MB_ICONHAND MB_APPLMODAL		
0012EDE0	00000000	LanguageID = 0 (LANG_NEUTRAL)		
0012EDE8	0040152A	? USER32. MessageBoxA	UnPackMe. 00401524	0012EDE4
0012EDEC	00000000	hOwner = NULL		
0012EDF0	004060AC	Text = "Armadillo not detected!"		
0012EDF4	004060C4	Title = "ERROR!"		
0012EDF8	00000010	Style = MB_OK MB_ICONHAND MB_APPLMODAL		
0012F204	0040169F	? UnPackMe. 004014C0	UnPackMe. 0040169A	

Set 1 BP 0040150C, Ctrl + F2, F9 stop at BP

00401506	68 FF000000	PUSH OFF	
0040150B	50	PUSH EAX	
0040150C	68 CC604000	PUSH UnPackMe. 004060CC	ASCII "ALTUSERNAME"
00401511	FFD6	CALL NEAR ESI	
00401513	85C0	TEST EAX, EAX	
00401515	75 1E	JNZ SHORT UnPackMe. 00401535	
00401517	6A 10	PUSH 10	
00401519	68 C4604000	PUSH UnPackMe. 004060C4	ASCII "ERROR!"
0040151E	68 AC604000	PUSH UnPackMe. 004060AC	ASCII "Armadillo not detected!"
00401523	50	PUSH EAX	
00401524	FF15 80724400	CALL NEAR DWORD PTR DS:[<&user32. Message	USER32. MessageBoxA
0040152A	5F	POP EDI	
0040152B	32C0	XOR AL, AL	
0040152D	5E	POP ESI	
0040152E	81C4 00040000	ADD ESP, 400	
00401534	C3	RETN	
00401535	8D8C24 08010000	LEA ECX, DWORD PTR SS:[ESP+108]	
0040153C	68 FF000000	PUSH OFF	
00401541	51	PUSH ECX	
00401542	68 A4604000	PUSH UnPackMe. 004060A4	ASCII "USERKEY"

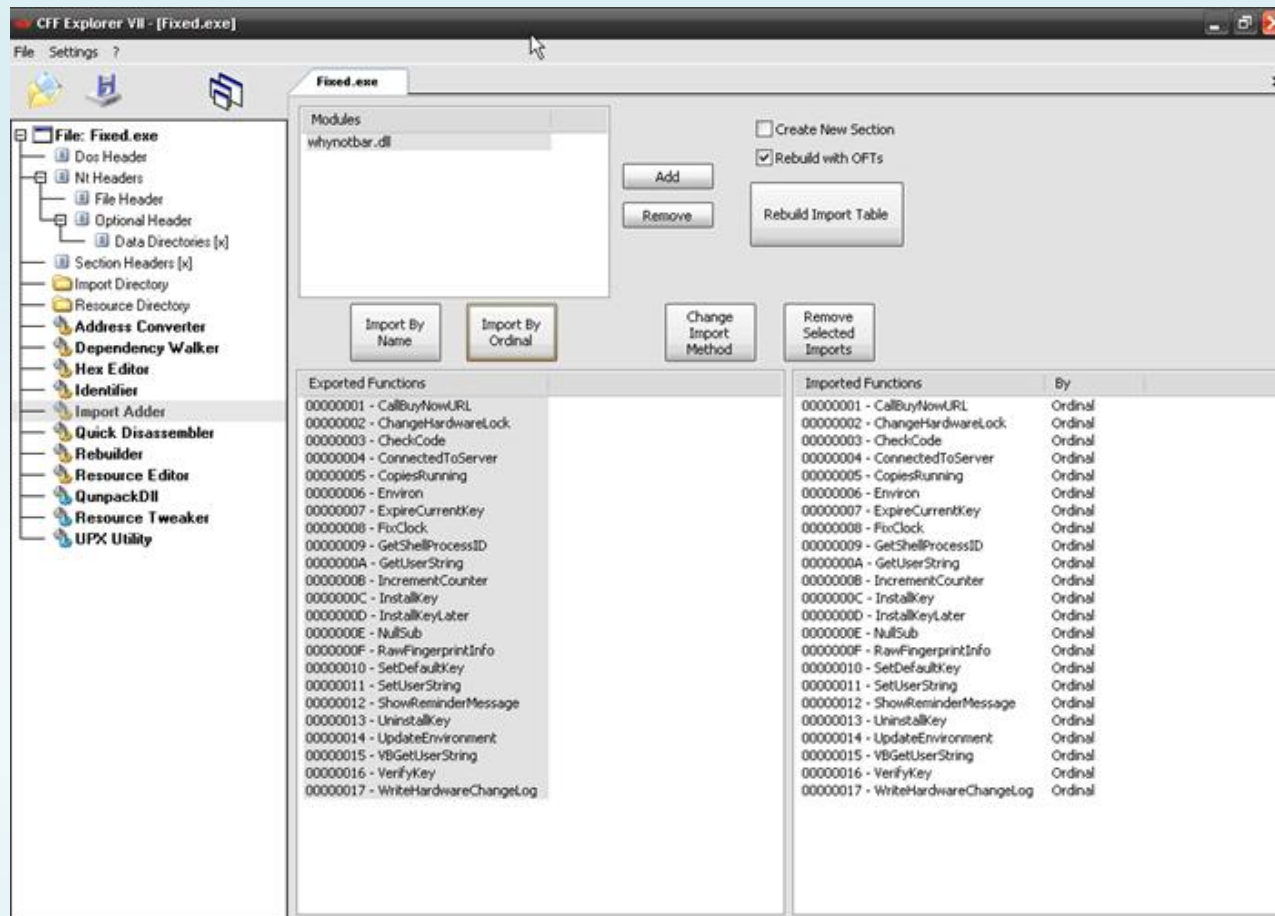
Patch as the image

00401506	68 FF000000	PUSH OFF	
0040150B	50	PUSH EAX	
0040150C	68 CC604000	PUSH UnPackMe.004060CC	ASCII "ALTUSERNAME"
00401511	FFD6	CALL NEAR ESI	
00401513	85C0	TEST EAX, EAX	
00401515	75 1E	JNZ SHORT UnPackMe.00401535	===>Patch to JMP
00401517	6A 10	PUSH 10	
00401519	68 C4604000	PUSH UnPackMe.004060C4	ASCII "ERROR!"
0040151E	68 AC604000	PUSH UnPackMe.004060AC	ASCII "Armadillo not detected!"
00401523	50	PUSH EAX	
00401524	FF15 80724400	CALL NEAR DWORD PTR DS:[<&user32.Message]	USER32.MessageBoxA
0040152A	5F	POP EDI	
0040152B	32C0	XOR AL, AL	
0040152D	5E	POP ESI	
0040152E	81C4 00040000	ADD ESP, 400	
00401534	C3	RETN	
00401535	8D8C24 08010000	LEA ECX, DWORD PTR SS:[ESP+108]	
0040153C	68 FF000000	PUSH OFF	
00401541	51	PUSH ECX	
00401542	68 A4604000	PUSH UnPackMe.004060A4	ASCII "USERKEY"
00401547	FFD6	CALL NEAR ESI	
00401549	85C0	TEST EAX, EAX	
0040154B	75 27	JNZ SHORT UnPackMe.00401574	===>Patch to JMP
0040154D	BF 9C604000	MOV EDI, UnPackMe.0040609C	ASCII "No key!"
00401552	83C9 FF	OR ECX, FFFFFFFF	

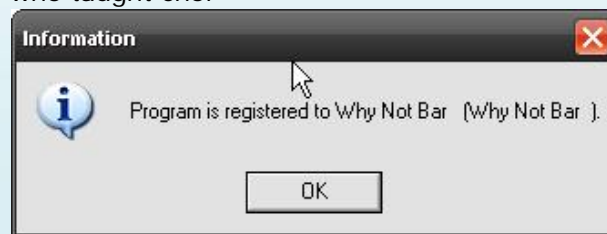
Save the Run tolerable.



Method 2, you can Add More **User Name**, **User Key** ... can patch directly in the dump file to bypass NAG, but here they play add 1 mod dll files from the File ... **ArmAccess.dll** from the infected patch fix the above . Most of the soft pack with Armadillo unpack after they finished 1 and add to the cracked always take from the hands of the foot. Tolerable ... File this name very private and very Pro "**whynotbar.dll**" ... ha ha ... with **CFF** Open File, select **Import Adapter**, select all the API function below, click **Import** and **Ordinal By** pressing **rebuild Import Tabale**



Save and run the test considered Khua khua name who taught choi





From the infected hen ... If patch file unpack he was OK to invite them through the "Retouch"

3. Retouch File dump

After Fix nano completed, it **ArmaInline** Add 1 code to EP Fix Nano should look very nice ko ...

Paused		L E M T W H C / K B R ... S
00D74070	60	PUSHAD
00D74071	9C	PUSHFD
00D74072	E8 00000000	CALL Fixed_.00D74077
00D74077	5A	POP EDX
00D74078	83EA 77	SUB EDX, 77
00D7407B	52	PUSH EDX
00D7407C	64:A1 30000000	MOV EAX, DWORD PTR FS:[30]
00D74082	8B40 0C	MOV EAX, DWORD PTR DS:[EAX+C]
00D74085	8B70 1C	MOV ESI, DWORD PTR DS:[EAX+1C]
00D74088	8B5E 08	MOV EBX, DWORD PTR DS:[ESI+8]
00D7408B	59	POP ECX
00D7408C	8959 14	MOV DWORD PTR DS:[ECX+14], EBX
00D7408F	51	PUSH ECX
00D74090	AD	LODS DWORD PTR DS:[ESI]
00D74091	8B40 08	MOV EAX, DWORD PTR DS:[EAX+8]
00D74094	8942 50	MOV DWORD PTR DS:[EDX+50], EAX
00D74097	33ED	XOR EBP, EBP
00D74099	8B58 3C	MOV EBX, DWORD PTR DS:[EAX+3C]

The code is responsible for the use of Nano **INT3** it will AutoFix. So we just need it to run, check all Funtion (if any) and click Exit. How such a code to help one another out Fix a few bugs **INT3** created by Nano. Xót To do what they should always Exit soft for sure. Exit When finished typing **Ctrl + G** to **401,000** and to conduct **Binary Copy** from the beginning to end. Then load dump file that we have not **Fix Nano** with **Binary ArmaInline** and **Paste** it. We will see in Olly **INT3** is a change in order to **Jump** respectively.

00401663	6A FF	PUSH -1	
00401665	68 B0304000	PUSH UnPackMe.004030B0	
0040166A	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00401670	50	PUSH EAX	
00401671	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00401678	83EC 1C	SUB ESP,1C	
0040167B	57	PUSH EDI	
0040167C	8BF9	MOV EDI,ECX	
0040167E	E8 0B150000	CALL <JMP.&mfc42.#4710>	
00401683	EB 03	JMP SHORT UnPackMe.00401688	
00401685	A5	MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]	
00401686	889C68 E0614000	MOV BYTE PTR DS:[EAX+EBP*2+4061E0],BL	
0040168D	68 D8614000	PUSH UnPackMe.004061D8	ASCII "john"
00401692	E8 69FFFFFF	CALL UnPackMe.00401600	
00401697	83C4 08	ADD ESP,8	
0040169A	E8 21FEFFFF	CALL UnPackMe.004014C0	
0040169F	84C0	TEST AL,AL	
004016A1	75 08	JNZ SHORT UnPackMe.004016AB	
004016A3	6A 00	PUSH 0	
004016A5	FF15 28724400	CALL NEAR DWORD PTR DS:[<&msvcrt.exit>]	msvcrt.exit
004016AB	56	PUSH ESI	
004016AC	90	NOP	
004016AD	90	NOP	
004016AE	90	NOP	

OK ... Save File and Run ... try running File and delicious EP returned correctly and reduce the Section 1. "Nano" which ArmInline Add to Fix Nano

00402DA4	55	PUSH EBP	==>EP
00402DA5	8BEC	MOV EBP,ESP	
00402DA7	6A FF	PUSH -1	
00402DA9	68 88484000	PUSH UnPackMd.00404888	
00402DAE	68 2A2F4000	PUSH <JMP.&msvcrt._except_handler3>	
00402DB3	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00402DB9	50	PUSH EAX	
00402DBA	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00402DC1	83EC 68	SUB ESP,68	
00402DC4	53	PUSH EBX	
00402DC5	56	PUSH ESI	
00402DC6	57	PUSH EDI	
00402DC7	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00402DCA	33DB	XOR EBX,EBX	
00402DCC	895D FC	MOV DWORD PTR SS:[EBP-4],EBX	
00402DCF	6A 02	PUSH 2	
00402DD1	FF15 18724400	CALL NEAR DWORD PTR DS:[<&msvcrt.__set_app_type	

Conducted to remove the balance of the Section of ArmaDillo

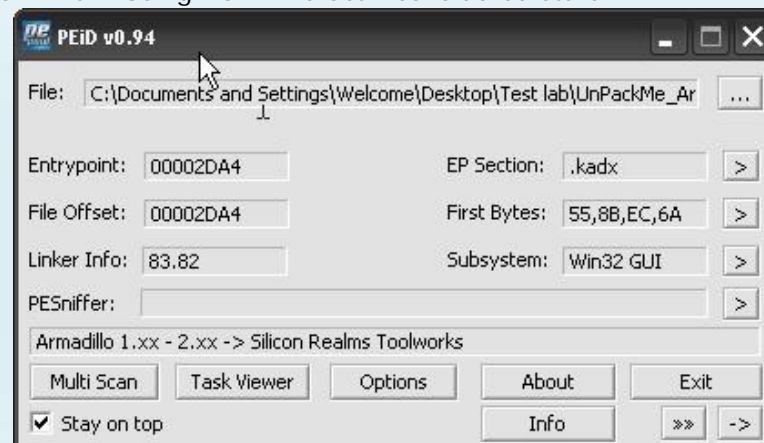
UnPackMe_Del_section.exe					
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address
Byte[8]	Dword	Dword	Dword	Dword	Dword
.kadx	000022F2	00001000	000022F2	00001000	00000000
.glsu	000013C8	00004000	000013C8	00004000	00000000
.frmp	00000358	00006000	00000358	00006000	00000000
zkjfv	00040000	00007000	00040000	00007000	00000000
.qvmhe	00010000	00007000	00010000	00007000	00000000
.gofq	00020000	00007000	00020000	00007000	00000000
.szet	00490000	00077000	00490000	00077000	00000000
.ygm	0046C000	00507000	0046C000	00507000	00000000
.TiGa	00001000	00973000	00001000	00973000	00000000

Xóa các
Section này

Rename the name of the **Section** for aesthetic slightly

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address
00000210	00000218	0000021C	00000220	00000224	00000228
Byte[8]	Dword	Dword	Dword	Dword	Dword
.text	00003000	00001000	000022F2	00000400	00000000
.rdata	00002000	00004000	000013AF	00002800	00000000
.data	00041000	00006000	00000261	00003C00	00000000
.idata	0092C000	00047000	0000C005	00004000	00000000
.idata	00001000	00973000	0000073A	00010200	00000000
.rsrc	0046C000	00974000	0046B8C6	00010400	00000000

After deleting the **Section**, File storage dump down half. Using **PeiD** File Scan considered stars



Cause PeiD that still recognize the value of the Armadillo's **MajorLinkerVersion** and **MenorLinkerVersion**. I know that this is also the seat choc ngoay Peid players who doubt it Scan 2 of this ... accidentally lượm know in time ...

Member	Offset	Size	Value	Meaning
Magic	00000110	Word	010B	PE32
MajorLinkerVersion	00000112	Byte	53	
MinorLinkerVersion	00000113	Byte	52	
SizeOfCode	00000114	Dword	00049000	
SizeOfInitializedData	00000118	Dword	0049A000	
SizeOfUninitializedData	0000011C	Dword	00000000	
AddressOfEntryPoint	00000120	Dword	00002DA4	.text
BaseOfCode	00000124	Dword	00007000	
BaseOfData	00000128	Dword	00057000	
ImageBase	0000012C	Dword	00400000	
SectionAlignment	00000130	Dword	00001000	
FileAlignment	00000134	Dword	00001000	
MajorOperatingSystemVersion	00000138	Word	0004	
MinorOperatingSystemVersion	0000013A	Word	0000	

According to experience your own children to look at the **EP** that it is **Microsoft Visual C + + 6.0**, if you do not have experience using the **RDG Packer Detector v0.6.5** considered to Detect File dump code with anything ... and compare with the **MajorLinkerVersion** and that they **MenorLinkerVersion** have listed below and use the **CFF** to Fix

Code MajorLinkerVersion MenorLinkerVersion

Microsoft Visual C + + 7.0 - v7.1 07 0A

Microsoft Visual C + + 7.0 dll Method 3 07 00

Microsoft Visual C + + 6.0 dll 06 00

Microsoft Visual C + + 6.0 [Debug] 06 00

Borland C + + 1999 05 00

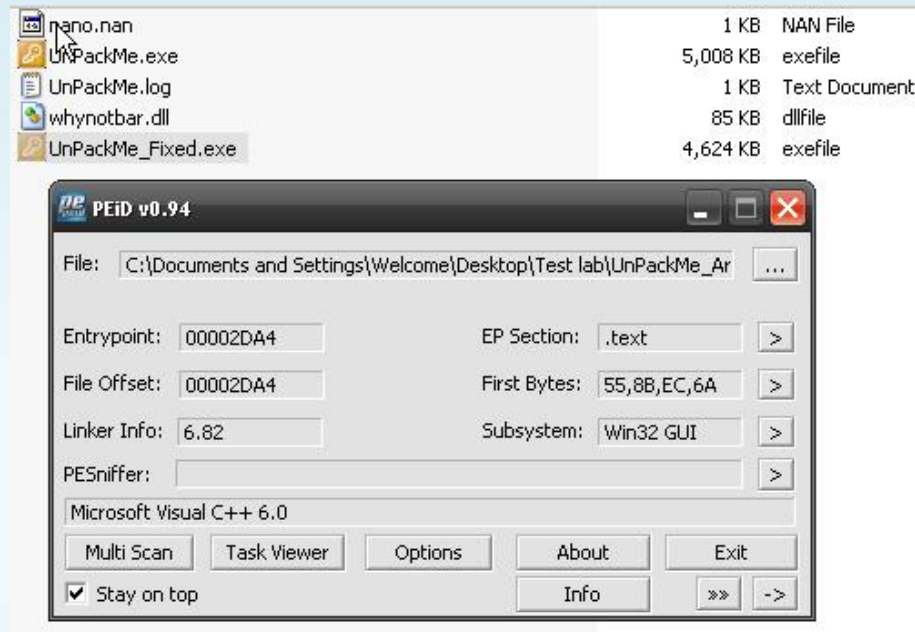
Borland Delphi v6.0 - v7.0 02 19

Microsoft Visual Basic 5.0 / 6.0 06 00

MASM32 / TASM32 05 0C

Member	Offset	Size	Value	Meaning
MajorLinkerVersion	00000112	Byte	06	
MinorLinkerVersion	00000113	Byte	00	
SizeOfCode	00000114	Dword	00049000	
SizeOfInitializedData	00000118	Dword	0049A000	
SizeOfUninitializedData	0000011C	Dword	00000000	
AddressOfEntryPoint	00000120	Dword	00002DA4	.kadx

Save and use the scan to see **Peid**



Ok ... too tired to unpack this ... 5 minutes without sitting writing copy that is a long long time ...
 oi hungry, they remember and add iu party ... brothers we pause here hen ...

IV. Conclusion

Finally, the complete message, this article I wrote for the purpose of storage and only the basic points to unpack Armadillo but I do not go more deeply into each section of these have too many resources no. Them's talk about the record more knowledge you are with me it is a pleasure, a feeling for their own Refresh your knowledge of minh.Rat thank him and you for taking the time valuable to you to read this document.

PS: This document is only a reference, the author is not responsible if the reader uses it to any goal.

Best Regards

_ [Why Not Bar] _

---+ +---==[Greatz thanks to]==---+ +---

My family, Computer_Angel, Moonbaby, Kienmanowar, Zombie_Deathman, Littleboy, Benina, QHQCrkcr, the_Lighthouse, Merc, Hoadongnoi, Nini ... all REA's members, TQN, HacNho, RongChauA, Deux, tlandn, light.phoenix, dqtlN, ARTEAM all my friend, and you.

Thanks to ---+ +---==[]=---+ +---

iamidiot, trickyboy, dzungltn, takada, hurt_heart, haule_nth, hytkl, Moth, XIANUA, nhc1987, v. Oxdie. v.. You have contributed greatly to the REA. Hope you will continue to promote J I want to thank **Teddy Rogers** for his great site, Reversing.be folks (especially **haggar**), Arteam folks (**Shub-Nigurrath**, **MaDMAn_H3rCuL3s**) and all folks on crackmes.de, thank to all members of **unpack.cn** (especially **fly** and **linhanshi**). Great thanks to **lena151** (I like your tutorials). And finally, thanks to **Ricardo NARVAJA** and all members on **CRACKSLATINOS**.

>>>> If you have any suggestions, comments or corrections email me: **WhyNotBar [at] reaonline.net**

**+ + CopyMemII
Debugblocker
IAT elimination
+ Code Splicing
+ Nanomites**

Target : DiaryOne 5.6

Crack Tool : 1.OllyDBG

2005

2. LordPE

Deluxe 1.4-by

yoda

3.Import

REConstructor

1.6 Final

4. ArmlInline

0.71

Author : Why Not Bar

Before they start to say a pair! Please Dear Uncle Em with 1 lính new acceptance of children with written tut if the goal is how to how to unpack Done 1 Soft Pack with Arma is not going into the resolution of the copy. so should the quality of articles in English can not Hachho, Benina member or the other ... hope for the Uncle! Uncle Funnynet suggestions for a message they do not have the quality they should also not dare to write more! hic hic! I do not Uncle Funnynet sad and I also hope that many people comment so they have to be progress. Meals before any gambling say they should wait tut they write. Again I just want this article is the view that Uncle not only dare to be first TUT. Thôi start time ...

_ Target Load:

006F6793	55	PUSH EBP	
006F6794	8BEC	MOV EBP,ESP	
006F6796	6A FF	PUSH -1	
006F6798	68 88017200	PUSH DiaryOne.00720188	
006F679D	68 00646F00	PUSH DiaryOne.006F64D0	SE handler installation
006F67A2	64:R1 00000001	MOV EAX,DWORD PTR FS:[0]	
006F67A8	50	PUSH EAX	
006F67A9	64:8925 0000	MOV DWORD PTR FS:[0],ESP	
006F67B0	83EC 58	SUB ESP,58	
006F67B3	F3	REPNE SCASB	

_ **Alt + F1, BP WaitForDebugEvent** and press **F9** Stack window we see as follows:


```

0012DB90  006E636D  CALL to WaitForDebugEvent from DiaryOne.006E6367
0012DB94  0012ED1C  pDebugEvent = 0012ED1C
0012DB98  000003E8  Timeout = 1000. ms
0012DB9C  7C910738  ntdll.7C910738
0012DBA0  00000000

```

_Chon Like



_ Next **WaitForDebugEvent** BC, BP WriteProcessMemory, press F9, 1 Nag appears click OK you'll see the following:

```

5012D9FC  006E6837  CALL to WriteProcessMemory from DiaryOne.006E6831
0012DA00  00000048  hProcess = 00000048 (window)
0012DA04  0067A000  Address = 67A000
0012DA08  00E50048  Buffer = 00E50048
0012DA0C  00001000  BytesToWrite = 1000 (4096.)
0012DA10  0012DB40  pBytesWritten = 0012DB40
0012DA14  00000000
0012DA18  00000000

```

_ **Ctrl + F9** execute till return. You will be here:

```

7C802294  C2 1400  RETN 14
7C802297  50      PUSH EAX
7C802298  E8 DE700000  CALL kernel32.7C80937B
7C802299  E9 84000000  JMP kernel32.7C802326
7C8022A2  8D45 14     LEA EAX,DWORD PTR SS:[EBP+14]
7C8022A5  50      PUSH EAX
7C8022A6  6A 04     PUSH 4
7C8022A8  8D45 FC     LEA EAX,DWORD PTR SS:[EBP-4]
7C8022AB  50      PUSH EAX
7C8022AC  8D45 F8     LEA EAX,DWORD PTR SS:[EBP-8]
7C8022AF  50      PUSH EAX
7C8022B0  57      PUSH EDI
7C8022B1  FF06     JMP NEAR ESI

```

_ In areas Stack you roll the mouse to meet the second return from the return first and we are

```

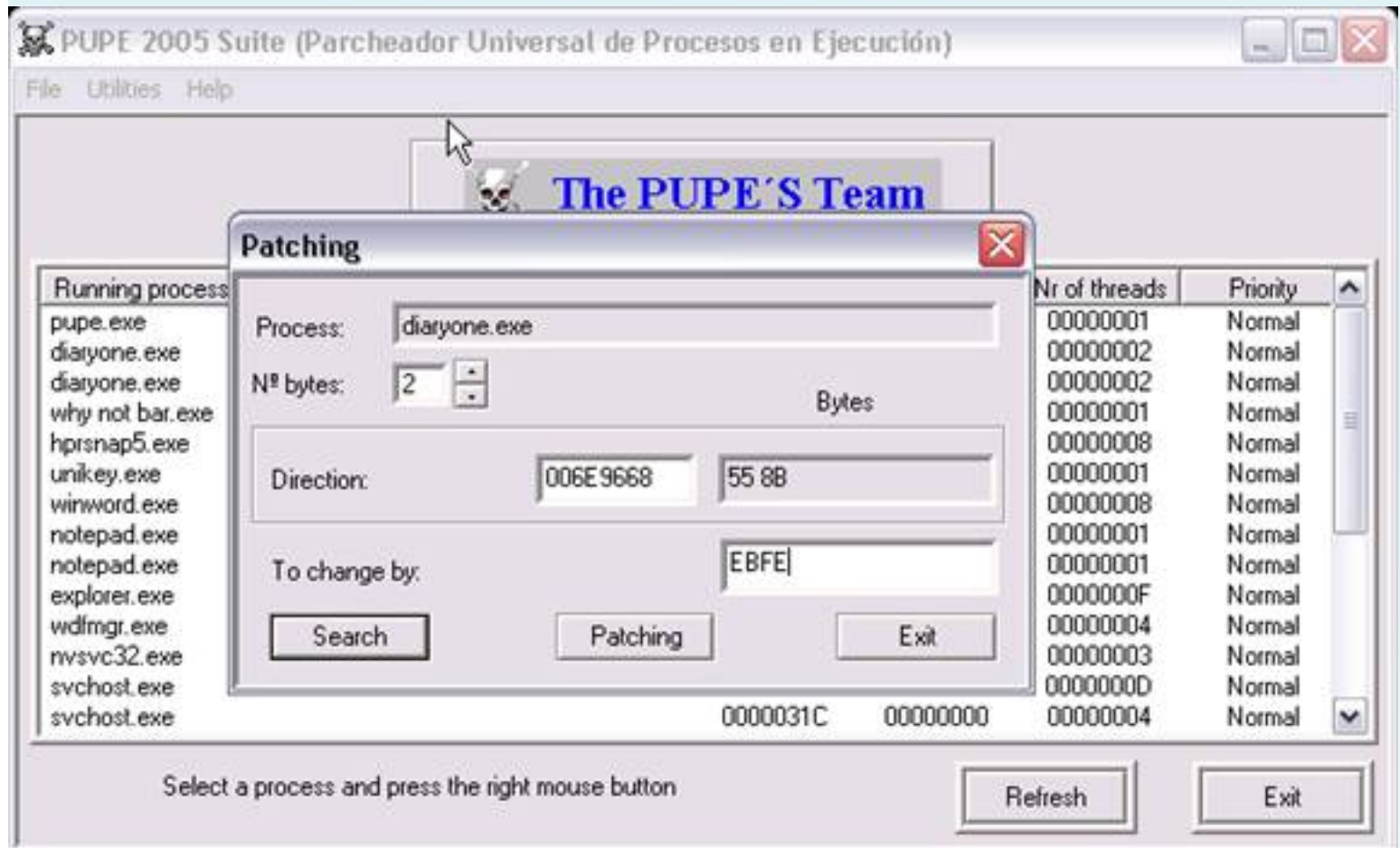
0012DB44  00E51048
0012DB48  0012DB88
0012DB4C  006E92E2  RETURN to DiaryOne.006E92E2 from DiaryOne.006E9668
0012DB50  00000279
0012DB54  0037E200
0012DB58  00000000
0012DB5C  00000000
0012DB60  00002790

```

_ Back to the CPU, Ctrl + G: 006E9668, Ctrl + R, select the command Call 2 Double-click on it and submit lenh Call this as follows:

006E95EF	50	PUSH EAX
006E95F0	90	NOP
006E95F1	90	NOP
006E95F2	90	NOP
006E95F3	90	NOP
006E95F4	90	NOP
006E95F5	83C4 0C	ADD ESP,0C
006E95F8	50	PUSH EAX
006E95F9	F7D0	NOT EAX

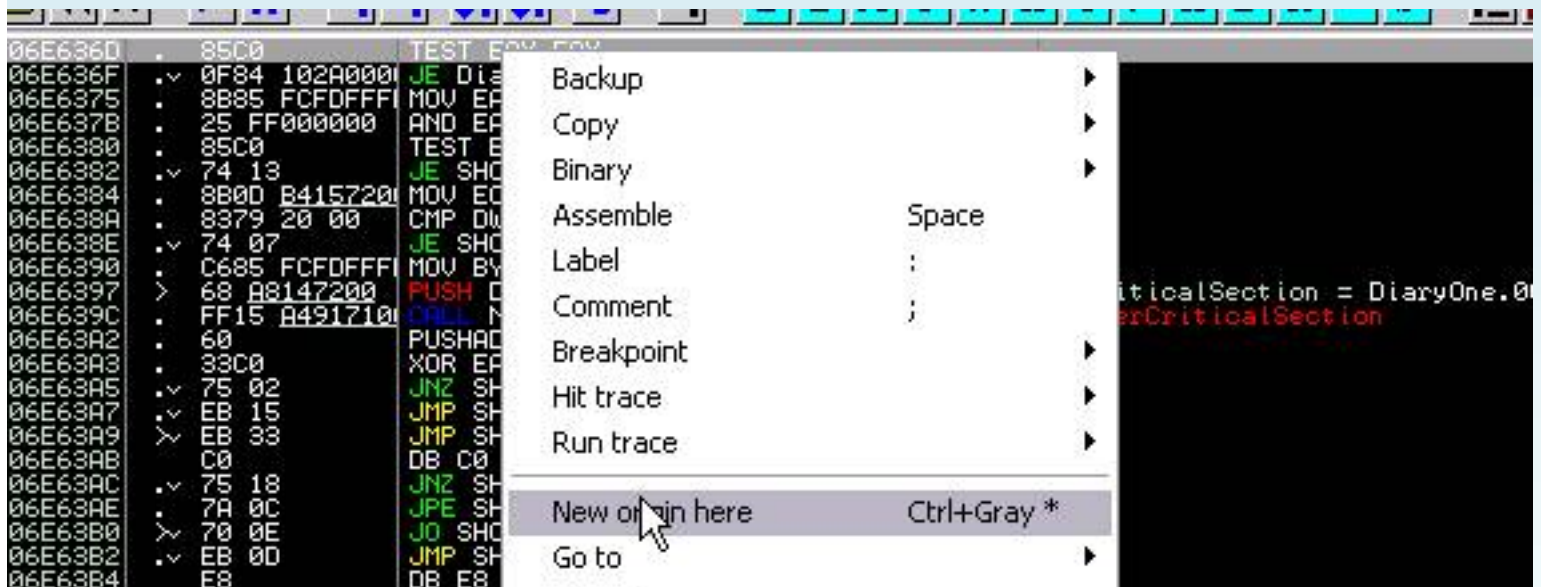
_Mo PuPe and enter as follows:



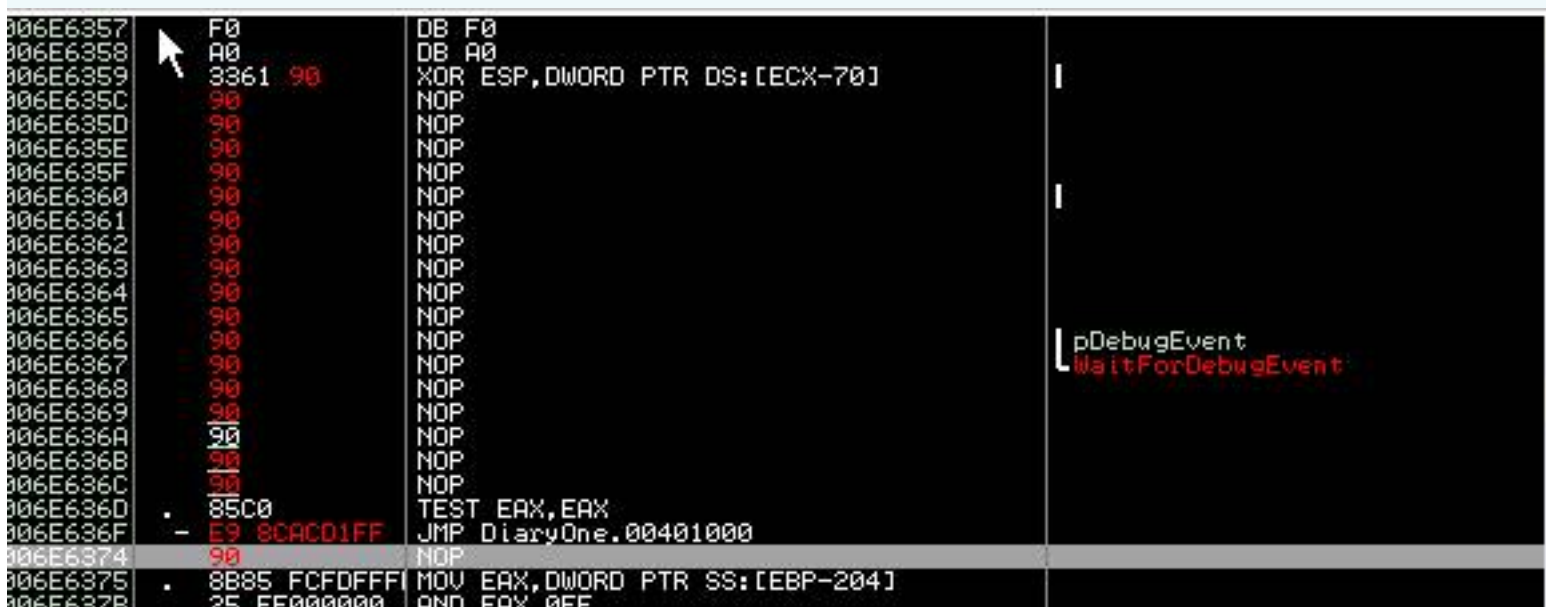
Patching _Nhan, you are a breakpoint in the kernel32 WriteProcessMemory. Bc it, then set a breakpoint bp WaitForDebugEvent, F9. At 0012DB90 [in Stack] -> right click> Follow in Disassembler.



_ Need to set an origin here:



_ And patch as follows:



_ Jump to 401,000 and the patch:


```

00401000 8105 34ED1200 ADD DWORD PTR DS:[12ED34],1000
0040100A 8105 40ED1200 ADD DWORD PTR DS:[12ED40],1000
00401014 8105 44ED1200 ADD DWORD PTR DS:[12ED44],1000
0040101E 813D 34ED1200 CMP DWORD PTR DS:[12ED34],DiaryOne.0067
00401028 0F85 46532E00 JNZ DiaryOne.006E6374
0040102E 90 NOP
0040102F 90 NOP
00401030 0000 ADD BYTE PTR DS:[EAX],AL
00401032 0000 ADD BYTE PTR DS:[EAX],AL
00401034 0000 ADD BYTE PTR DS:[EAX],AL
00401036 0000 ADD BYTE PTR DS:[EAX],AL
00401038 0000 ADD BYTE PTR DS:[EAX],AL

```

_ Continue in 3 patch addresses containing OEP:

```

0012ED2C 00000000
0012ED30 00000000
0012ED34 00400000 ASCII "MZP"
0012ED38 00000002
0012ED3C 00000000
0012ED40 00400000 ASCII "MZP"
0012ED44 00400000 ASCII "MZP"
0012ED48 00000001

```

1 _Dat breakpoint in 0040102E Press F9 to run

Command: bp WaitForDebugEvent
Breakpoint at DiaryOne.0040102E

_Bay Time we pass by debug blocker

```

00401000 8105 34ED1200 ADD DWORD PTR DS:[12ED34],1000
0040100A 8105 40ED1200 ADD DWORD PTR DS:[12ED40],1000
00401014 8105 44ED1200 ADD DWORD PTR DS:[12ED44],1000
0040101E 813D 34ED1200 CMP DWORD PTR DS:[12ED34],DiaryOne.0067
00401028 0F85 46532E00 JNZ DiaryOne.006E6374
0040102E 68 44060000 PUSH 644
00401033 E8 5999457C CALL kernel32.DebugActiveProcessStop
00401038 90 NOP
00401039 0000 ADD BYTE PTR DS:[EAX],AL
0040103B 0000 ADD BYTE PTR DS:[EAX],AL
0040103D 0000 ADD BYTE PTR DS:[EAX],AL

```

_ Press F8 when the last from the following functions nop Call DebugActiveProcessStop

```

Registers (FPU)
EAX 00000001
ECX 0012DB5C
EDX 7C90EB94 ntdll.KiFastSystemC
EBX 7FFD0000
ESP 0012DB7C
EBP 0012DB88
ESI 00000044

```

_ Open a **OllyDBG** other, Attach. F9, F12 and patch as follows:

0067ABD0	55	PUSH EBP
0067ABD1	8BEC	MOV EBP,ESP
0067ABD3	83C4 F0	ADD ESP,-10
0067ABD6	B8 28A26700	MOV EAX,DiaryOne.0067A228
0067ABD8	E8 F8CB08FF	CALL DiaryOne.004077D8
0067ABE0	E8 E7A9FFFF	CALL DiaryOne.006755CC
0067ABE5	E8 B2A208FF	CALL DiaryOne.00404E9C
0067ABEA	8BC0	MOV EAX,EAX
0067ABEC	0000	ADD BYTE PTR DS:[EAX],AL
0067ABEE	0000	ADD BYTE PTR DS:[EAX],AL
0067ABF0	0000	ADD BYTE PTR DS:[EAX],AL
0067ABF2	0000	ADD BYTE PTR DS:[EAX],AL
0067ABF4	0000	ADD BYTE PTR DS:[EAX],AL

_Bay Time we start identifying IAT and IAT End

00407713	90	NOP	
00407714	FF25 E8170E01	JMP NEAR DWORD PTR DS:[10E17E8]	
0040771A	8BC0	MOV EAX,EAX	
0040771C	FF25 6C1B0E01	JMP NEAR DWORD PTR DS:[10E1B6C]	kernel32.LocalAlloc
00407722	8BC0	MOV EAX,EAX	
00407724	FF25 D0120E01	JMP NEAR DWORD PTR DS:[10E12D0]	kernel32.TlsGetValue
0040772A	8BC0	MOV EAX,EAX	
0040772C	FF25 64170E01	JMP NEAR DWORD PTR DS:[10E1764]	kernel32.TlsSetValue
00407732	8BC0	MOV EAX,EAX	
00407734	50	PUSH EAX	
00407735	6A 40	PUSH 40	
00407737	E8 E0FFFFFF	CALL DiaryOne.0040771C	JMP to kernel32.LocalAlloc
0040773C	C3	RETN	

_Voi Steps familiar operation and we have:

IAT start:

010E12B8	006D0074	DiaryOne.006D0074
010E12BC	00000000	
010E12C0	004E0184	DiaryOne.004E0184
010E12C4	020C0106	
010E12C8	77D57608	USER32.IsCharAlphaA
010E12CC	77D4EF49	USER32.SetKeyboardState
010E12D0	7C809750	kernel32.TlsGetValue
010E12D4	77DD761B	ADVAPI32.RegOpenKeyExA
010E12D8	77D4B49D	USER32.InvalidRect
010E12DC	77D4B3E7	USER32.IntersectRect

IAT end:

010E12E0	7C80A34E	kernel32.CompareStringW
010E12E4	7C80D293	kernel32.CompareStringA
010E12E8	00EC6EA8	
010E12EC	00EC6FB0	
010E12F0	00EC7059	
010E12F4	00EC6FEE	
010E12F8	00EC6F02	
010E12FC	77DD7883	ADVAPI32.RegQueryValueExA
010E1300	00000000	
010E1304	01844224	
010E1308	02081032	

_buoc next IAT we find complete and paste into the IAT is only OK. **ArmaDetach 1.1** Open and Drag "DiaryOne.EXE" released into the window to see the program as follows



_ Open 1 Olly again and select the [process ID](#) and [Child](#) Attach. F9, F12, to patch 558B

006F6793	55	push ebp	
006F6794	8BEC	mov ebp, esp	
006F6796	6A FF	push -1	
006F6798	68 88017200	push DiaryOne.00720188	
006F679D	68 D0646F00	push DiaryOne.006F64D0	
006F67A2	64:A1 000000	mov eax, dword ptr fs:[0]	

_ Use OllyScript 0.92 Script run "[Armadillo V4.0-V4.4.Standard.Protection.osc](#)." Wait script you run to finish here

地址	十六进制	反汇编	注释
0067ABD0	5A	pop edx	This is the OEP!
0067ABD1	A9 5366CBD2	test eax, D2CB6653	
0067ABD6	07		
0067ABD7	CD A		
0067ABD9	45		
0067ABDA	BF C		
0067ABDF	1AE7		
0067ABE1	C516		
0067ABE3	1AF0		
0067ABE5	CA C		
0067ABE8	D7	xlslat byte ptr ds:[ebx+al]	
0067ABE9	DD3425 0F22B	fsave (108-byte) ptr ds:[E5BF220F]	
0067ABF0	0F22	???	未知命令



_ Alt + F1 and enter the following:

Command	? kernel32.LocalAlloc	HEX: 7C8099BD - DEC: 2088802749 - ASCII: €™½
---------	-----------------------	---

_ Alt + M, Ctrl + B, reverse the number again as follows:

ASCII	CCC
UNICODE	唇
HEX +04	BD 99 80 7C

☐ Entire block
 ☐ Case sensitive

_ OK to you here:

唇 - 01111000..01132FFF		
011114B4	77F19D07	GDI32.LineTo
011114B8	77D6F7D0	USER32.FindWindowExA
011114BC	7C8099BD	kernel32.LocalAlloc
011114C0	7C801EEE	kernel32.GetStartupInfoA
011114C4	77D4B46E	USER32.SetRect
011114C8	77D6F3C6	USER32.FindWindowA
011114CC	77D4D3C5	USER32.FillRect
011114D0	77D4BDD1	USER32.EqualRect
011114D4	7C810311	kernel32.lstrcpynA
011114D8	77D6152F	USER32.SendDlgItemMessageA
011114DC	7C81E4BD	kernel32.CreateEventA
011114E0	77D4D935	USER32.EnumWindows
011114E4	77D4FACD	USER32.EnumThreadWindows
011114E8	7C80C9C1	kernel32.GetLocalTime
011114EC	77D6DA71	USER32.EnumClipboardFormats
011114F0	77D4BC8E	USER32.MsgWaitForMultipleObjects
011114F4	77D4B4C5	USER32.EndPaint
011114F8	77F1DCC0	GDI32.DeleteEnhMetaFile
011114FC	77D4C4D4	USER32.EnableWindow
01111500	77D97BAD	USER32.EnableScrollBar

_ Alt + C, in the window dump press Ctrl + G and enter 11114BC

011114D4	77F19D07	GDI32.LineTo
011114B8	77D6F7D0	USER32.FindWindowExA
011114BC	7C8099BD	kernel32.LocalAlloc
011114C0	7C801EEE	kernel32.GetStartupInfoA
011114C4	77D4B46E	USER32.SetRect
011114C8	77D6F3C6	USER32.FindWindowA

_ Roll back to determine the IAT start:

011112B8	006D0074	DiaryOne.006D0074
011112BC	00000000	
011112C0	004E0184	DiaryOne.004E0184
011112C4	020C013C	
011112C8	77D5760B	USER32.IsCharAlphaA
011112CC	77D4EF49	USER32.SetKeyboardState
011112D0	7C809750	kernel32.TlsGetValue
011112D4	77DD761B	ADVAPI32.RegOpenKeyExA

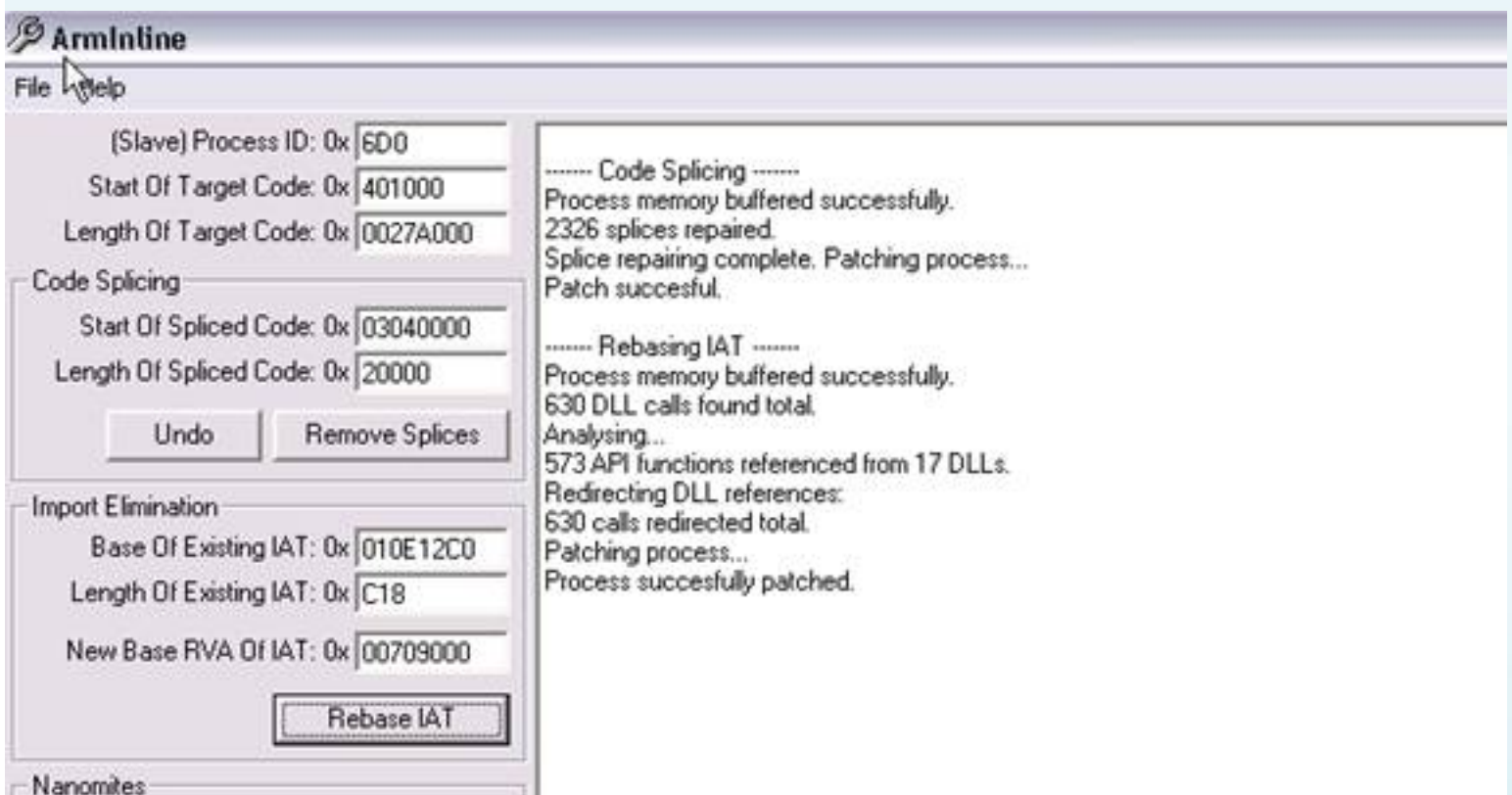
Scroll down _ identified IAT end:

01111EC8	7C801D77	kernel32.LoadLibraryA
01111ECC	00EF7059	
01111ED0	00EF6FEE	
01111ED4	00EF6F02	
01111ED8	77DD7883	ADVAPI32.RegQueryValueExA
01111EDC	00000000	

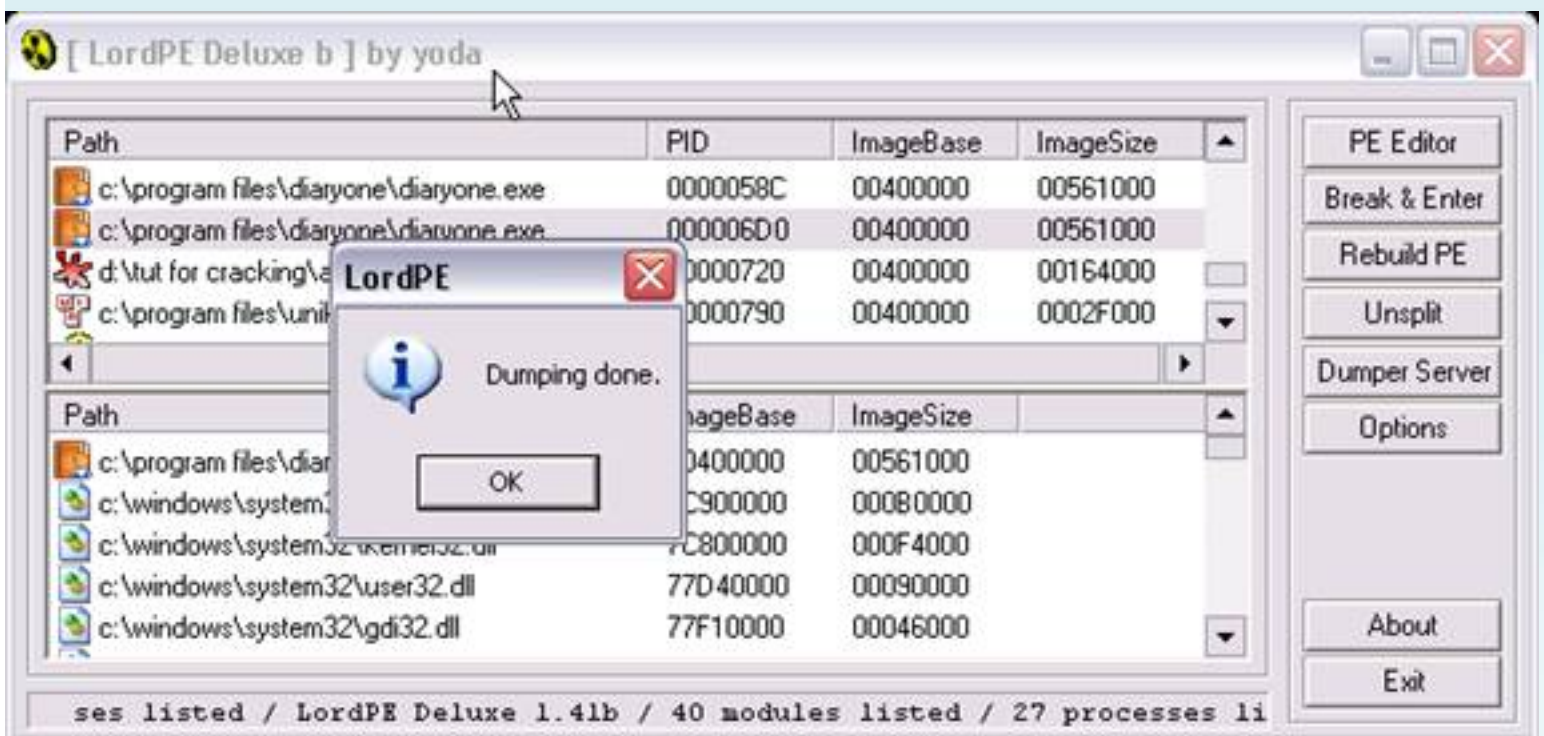
_ Now Binany \ Binany Copy and Paste on the IAT is not yet fully complete. After Paste you will find the No. 1 function:

Address	Value	Comment
010E1E98	77D6F29F	USER32.UnhookWindowsHookEx
010E1E9C	7C825F62	kernel32.FormatMessageA
010E1EA0	77D7F116	USER32.SetMenu
010E1EA4	00EF709D	
010E1EA8	7C839308	kernel32._lclose
010E1EAC	77F178DC	GDI32.SelectClipRgn
010E1EB0	7C809CAD	kernel32.MultiByteToWideChar
010E1EB4	77F18834	GDI32.SetBrushOrgEx
010E1EB8	00EF6F15	
010E1EBC	7C80A34E	kernel32.CompareStringW
010E1EC0	7C80D293	kernel32.CompareStringA
010E1EC4	00EF6E98	
010E1EC8	7C801D77	kernel32.LoadLibraryA
010E1ECC	00EF7059	
010E1ED0	00EF6FEE	
010E1ED4	00EF6F02	

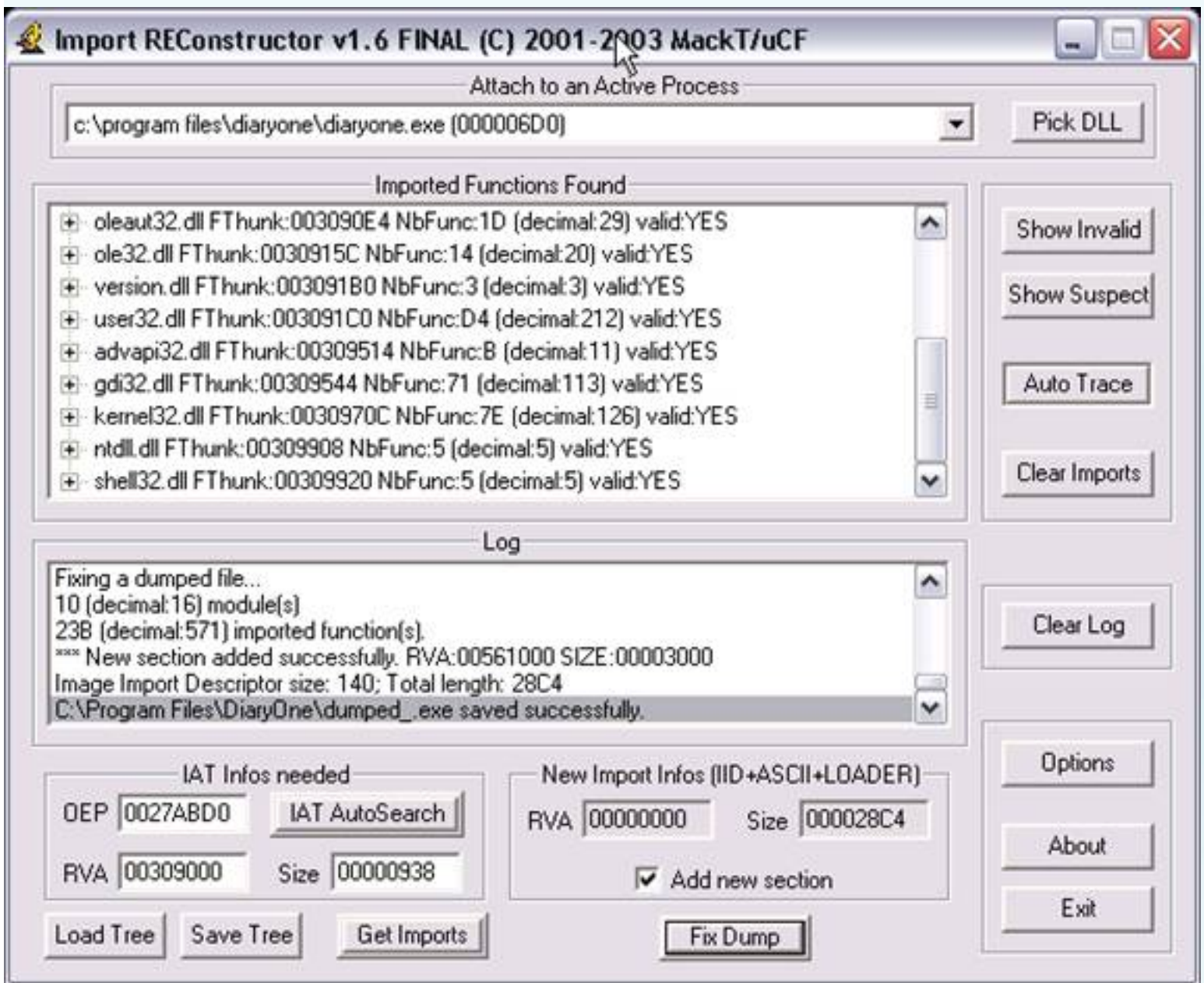
_ So is our Fix the IAT. ArmaDetach now closed and olly there again. Next we Fix **IAT elimination, Code Splicing**. ArmaInline M in to fill up as follows:



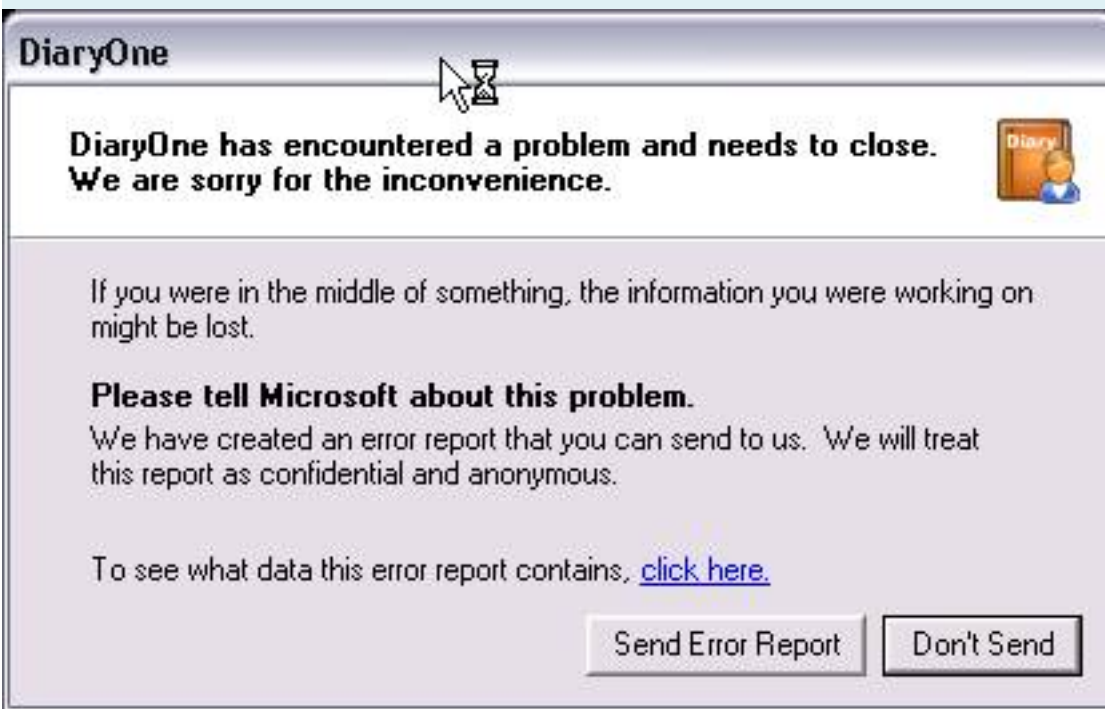
_ Good call, dump Full stop! Use LordPE



_ Open ImportREC fill and dump Fix



_ Run the file to try "dumped_.exe"



Em That the NAG as the Uncle confused. Here they also say more if gambling is done by way of Uncle Tomo to this month, the meat is not so! That the signs of NANO now! She also not know why more? But in this case he m  y open and found 1 Fix it, see Uncle n   ... Load File "dumped.exe" Olly to be Crash

Address	Disassembly	Comment
0067ABD0	55	MOV EBP, ESP
0067ABD1	8BEC	MOV EBP, ESP
0067ABD3	83C4 F0	ADD ESP, -10
0067ABD6	B8 28A26700	MOV EAX, dumped_.0067A228
0067ABD8	E8 F8CB08FF	CALL dumped_.004077D8
0067ABE0	E8 E7A9FFFF	CALL dumped_.0067550C
0067ABE5	E8 B2A208FF	CALL dumped_.00404E9C
0067ABEA	8BC0	MOV EAX, EAX
0067ABEC	0000	ADD BYTE PTR DS:[EAX], AL
0067ABEE	0000	ADD BYTE PTR DS:[EAX], AL
0067ABF0	0000	ADD BYTE PTR DS:[EAX], AL

_ Press Shift + F9 to Run, and it does the error NAG

DiaryOne



DiaryOne has encountered a problem and needs to close.
We are sorry for the inconvenience.



If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.

We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, [click here](#).

Send Error Report

Don't Send

_Nhan F12, Alt + K

Address	Stack	Procedure	Called from	Frame
0012DE44	7C90E9AB	Includes ntdll.KiFastSystemCallRet	ntdll.7C90E9A9	0012DEE0
0012DE48	7C8094F2	ntdll.ZwWaitForMultipleObjects	kernel32.7C8094EC	0012DEE0
0012DEE4	7C809C86	? kernel32.WaitForMultipleObjectsEx	kernel32.7C809C81	0012DEE0
0012DF00	6945763C	? kernel32.WaitForMultipleObjects	faultrep.69457636	0012DEFC
0012E894	694582B1	? faultrep.6945705D	faultrep.694582AC	0012E890
0012F908	7C863059	Includes faultrep.694582B1	kernel32.7C863053	0012F904
0012FB7C	00404B34	? <JMP.&kernel32.UnhandledException	dumped_.00404B2F	0012FB78

_va selected as type:

0012DE44	7C90E9AB	Includes ntdll.KiFastSystemCallRet	ntdll.7C90E9A9
0012DE48	7C8094F2	ntdll.ZwWaitForMultipleObjects	kernel32.7C8094EC
0012DEE4	7C809C86	? kernel32.WaitForMultipleObjectsEx	kernel32.7C809C81
0012DF00	6945763C	? kernel32.WaitForMultipleObjects	faultrep.69457636
0012E894	694582B1	? faultrep.6945705D	faultrep.694582AC
0012F908	7C863059	Includes faultrep.694582B1	kernel32.7C863053
0012FB7C	00404B34	? <JMP.&kernel32.Unha	dumped_.00404B2F

Actualize

Show arguments

Space

Follow address in stack

Show procedure

Enter

Show call

Execute to return

F4

Copy to clipboard

Appearance

_ta to here:


```

00404B00  C2 0400      RETN 4
00404B10  8B4424 04    MOV EAX,DWORD PTR SS:[ESP+4]
00404B14  F740 04 060000 TEST DWORD PTR DS:[EAX+4],6
00404B1B  0F85 89000000 JNZ dumped_.00404BAA
00404B21  803D 18B06700 CMP BYTE PTR DS:[67B018],0
00404B28  77 0F      JA SHORT dumped_.00404B39
00404B2A  8D4424 04    LEA EAX,DWORD PTR SS:[ESP+4]
00404B2E  50        PUSH EAX
00404B2F  E8 00C8FFFF CALL <JMP.&kernel32.UnhandledExceptionF
00404B34  83F8 00     CMP EAX,0
00404B37  74 71      JE SHORT dumped_.00404BAA
00404B39  8B4424 04    MOV EAX,DWORD PTR SS:[ESP+4]
00404B3D  FC        CLD
00404B3E  E8 C9F5FFFF CALL dumped_.0040410C

```

Funtion _cuon to top this and like



_Ctrl + F2, Shift + F9 we stopped at the BP Set nay

```

00404B00  C2 0400      RETN 4
00404B10  8B4424 04    MOV EAX,DWORD PTR SS:[ESP+4]
00404B14  F740 04 060000 TEST DWORD PTR DS:[EAX+4],6
00404B1B  0F85 89000000 JNZ dumped_.00404BAA
00404B21  803D 18B06700 CMP BYTE PTR DS:[67B018],0
00404B28  77 0F      JA SHORT dumped_.00404B39
00404B2A  8D4424 04    LEA EAX,DWORD PTR SS:[ESP+4]
00404B2E  50        PUSH EAX
00404B2F  E8 00C8FFFF CALL <JMP.&kernel32.UnhandledExceptionF
00404B34  83F8 00     CMP EAX,0
00404B37  74 71      JE SHORT dumped_.00404BAA

```

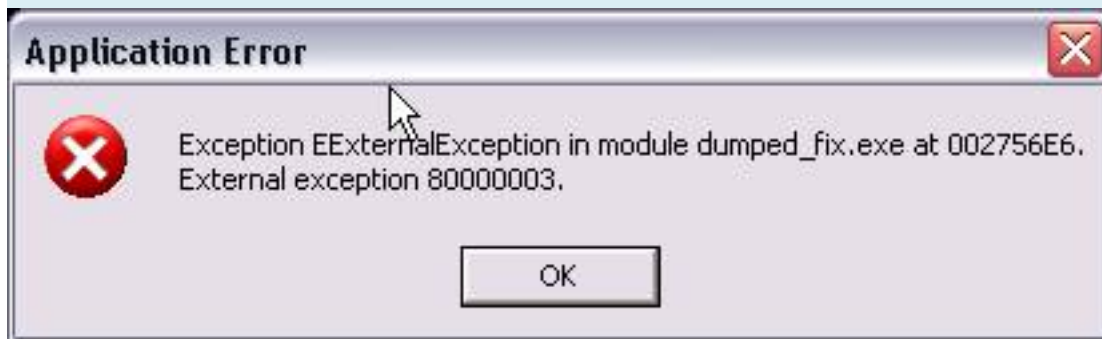
_ta need to patch at:

00404B28 / 77 0F JA SHORT dumped_.00404B39

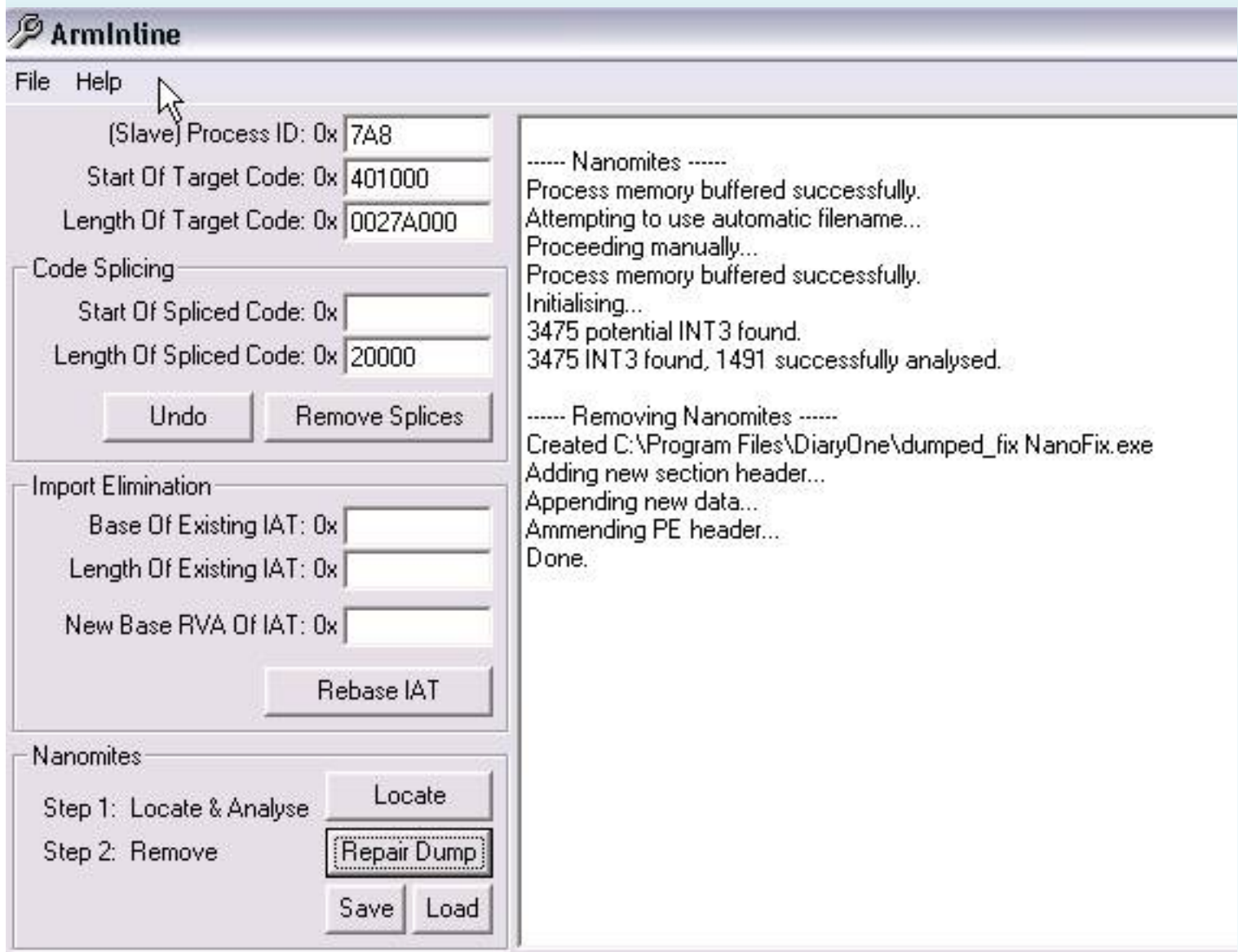
And the patch as follows:

00404B17	FF 48 84 000000	TEST DWORD PTR DS:[EAX+7],0
00404B1B	0F85 89000000	JNZ dumped_.00404BAA
00404B21	803D 18B06700	CMP BYTE PTR DS:[67B018],0
00404B28	EB 0F	JMP SHORT dumped_.00404B39
00404B2A	8D4424 04	LEA EAX,DWORD PTR SS:[ESP+4]
00404B2E	50	PUSH EAX
00404B2F	E8 00C8FFFF	CALL <JMP.&kernel32.UnhandledExceptionF
00404B34	83F8 00	CMP EAX,0
00404B37	74 71	JE SHORT dumped_.00404BAA
00404B39	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]
00404B3D	FC	CLD
00404B3E	E8 C9F5FFFF	CALL dumped_.0040410C
00404B43	8B5424 08	MOV EDX,DWORD PTR SS:[ESP+8]
00404B47	6A 00	PUSH 0

_ Then click to select Copy to executable \ Section \ Save File (1 should put the file name). Run just try File Save



_Ha Ha Nano appear. Load mortar save the file with signs Nano to Olly, open ArmInline enter parameters and to conduct Fix Nano as usual



_Run Test, run too!



_Em On this too! Miss hearing gòi But the place to do well. Again olly to load it. Here we pay attention to our **002757B9** plus **400,000** (Image Base). So in the CPU press Ctrl + G and enter **6757B9** and we come

006757B4	E8 F3F1D8FF	CALL dumped_f.004049AC	
006757B5	E9 F6000000	JMP dumped_f.006758B4	
006757BE	90	NOP	
006757BF	90	NOP	
006757C0	90	NOP	
006757C1	90	NOP	
006757C2	90	NOP	
006757C3	8D45 CC	LEA EAX,DWORD PTR SS:[EBP-34]	
006757C6	E8 45EAF6FF	CALL dumped_f.005E4210	
006757CB	8D45 CC	LEA EAX,DWORD PTR SS:[EBP-34]	
006757CE	50	PUSH EAX	
006757CF	8D45 C8	LEA EAX,DWORD PTR SS:[EBP-38]	
006757D2	50	PUSH EAX	
006757D3	B1 01	MOV CL,1	
006757D5	BA B0596700	MOV EDX,dumped_f.006759B0	ASCII "DiaryOne"
006757DA	B8 FC596700	MOV EAX,dumped_f.006759FC	ASCII "N+Ks0jLEncQeedBzD4gYz"
006757DF	E8 1459F7FF	CALL dumped_f.005EB0F8	

_cuon mouse over 1 billion is to come:

00675775	E8 7212E0FF	CALL dumped_f.004769EC	
0067577A	8B45 D4	MOV EAX,DWORD PTR SS:[EBP-2C]	
0067577D	E8 CE94F8FF	CALL dumped_f.005FEC50	
00675782	3D 20402C00	CMPL EAX,2C4020	
00675787	7D 07	JGE SHORT dumped_f.00675790	
00675789	3D E0322900	CMPL EAX,2932E0	UNICODE "anana"
0067578E	7F 2E	JS SHORT dumped_f.006757BE	
00675790	8D45 D0	LEA EAX,DWORD PTR SS:[EBP-30]	
00675793	50	PUSH EAX	
00675794	B1 01	MOV CL,1	
00675796	BA B0596700	MOV EDX,dumped_f.006759B0	ASCII "DiaryOne"
00675798	B8 C4596700	MOV EAX,dumped_f.006759C4	ASCII "qj0UK+yjsrtw2RclU0rYqU4wWxTy"
006757A0	E8 5359F7FF	CALL dumped_f.005EB0F8	
006757A5	8B4D D0	MOV ECX,DWORD PTR SS:[EBP-30]	
006757A8	B2 01	MOV DL,1	
006757AA	A1 7C8E4000	MOV EAX,DWORD PTR DS:[408E7C]	
006757AF	E8 9491D9FF	CALL dumped_f.0040E948	
006757B4	E8 F3F1D8FF	CALL dumped_f.004049AC	
006757B9	E9 F6000000	JMP dumped_f.006758B4	
006757BE	90	NOP	

_dia simply Patch

00675787 / 7D 07 JGE SHORT dumped_f.00675790

We NOP it

0067577A	8B45 D4	MOV EAX,DWORD PTR SS:[EBP-2C]	
0067577D	E8 CE94F8FF	CALL dumped_f.005FEC50	
00675782	3D 20402C00	CMPL EAX,2C4020	
00675787	90	NOP	
00675788	90	NOP	
00675789	3D E0322900	CMPL EAX,2932E0	UNICODE "anana"
0067578E	7F 2E	JS SHORT dumped_f.006757BE	
00675790	8D45 D0	LEA EAX,DWORD PTR SS:[EBP-30]	
00675793	50	PUSH EAX	
00675794	B1 01	MOV CL,1	
00675796	BA B0596700	MOV EDX,dumped_f.006759B0	ASCII "DiaryOne"
00675798	B8 C4596700	MOV EAX,dumped_f.006759C4	ASCII "qj0UK+yjsrtw2RclU0rYqU4wWxTykaz lh9"
006757A0	E8 5359F7FF	CALL dumped_f.005EB0F8	
006757A5	8B4D D0	MOV ECX,DWORD PTR SS:[EBP-30]	
006757A8	B2 01	MOV DL,1	
006757AA	A1 7C8E4000	MOV EAX,DWORD PTR DS:[408E7C]	
006757AF	E8 9491D9FF	CALL dumped_f.0040E948	
006757B4	E8 F3F1D8FF	CALL dumped_f.004049AC	
006757B9	E9 F6000000	JMP dumped_f.006758B4	
006757BE	90	NOP	

_Save Again, Run test stars. But before Olly Run closed again.

DiaryOne

Diary to **msn** Spaces



why not bar



create account

Sign in

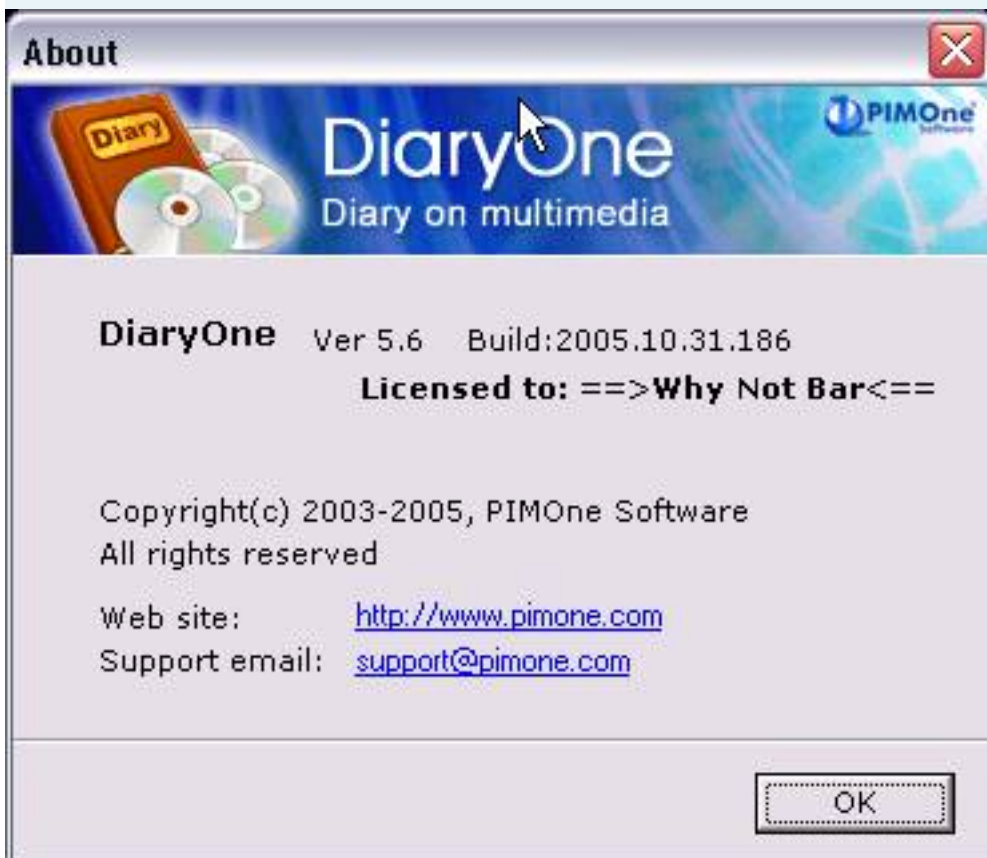
☐

Auto sign in next time

Done _Ha hectares packages. But here it's just a reminder only Nag but still unregistered



_Crack It simple, but they not only discussed! Because this area of more children called Uncle!



P / S: if the Uncle support of restraining the child would do some form of this Target again! Sour than it was this much! We believe some Uncle ...

Why Not Bar

Tutorials hacnho # 8

Manual unpacking EXE Shield v0.5 -> SMOKE

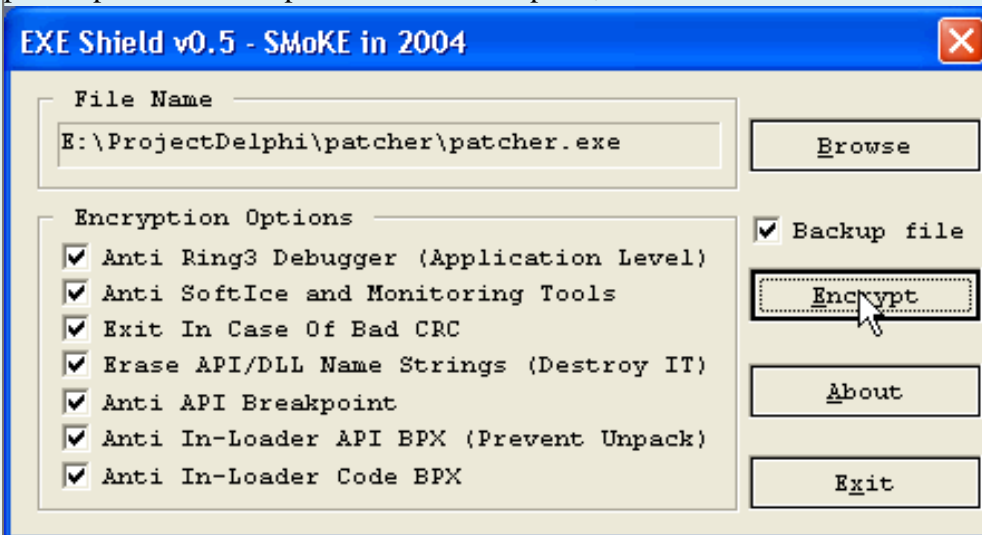
Information	Unpacking for Newbie's
Target	patcher.rar
Available	http://nhandan.info/hacnho/tuts/unpackme8_tuts.zip
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, Stud_PE 1.8, PEiD 0.92, 1.6 ImportREC Final.
Protection	EXE Shield v0.5 -> SMOKE
L Evel	Very Easy!
Category	Manual unpacking

1. Introduction

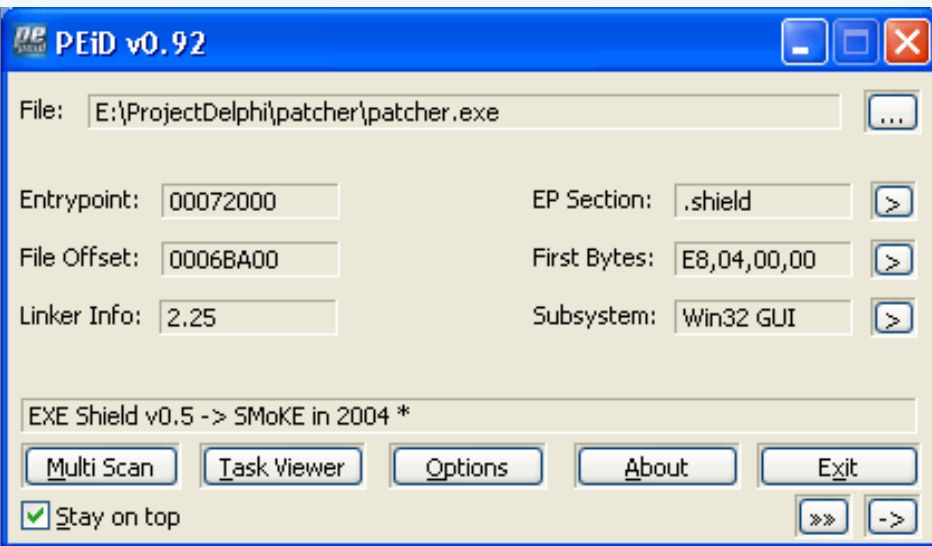
In a topic as Exetools Forum, when I wrote "EXE Shield v0.5 is very easy for unpack," A lot of memberz were not belive. And they were a request for tut prove my paroles ... He he, okay. I will explain to you the ways for this unpacker packer.

2. Getting Started

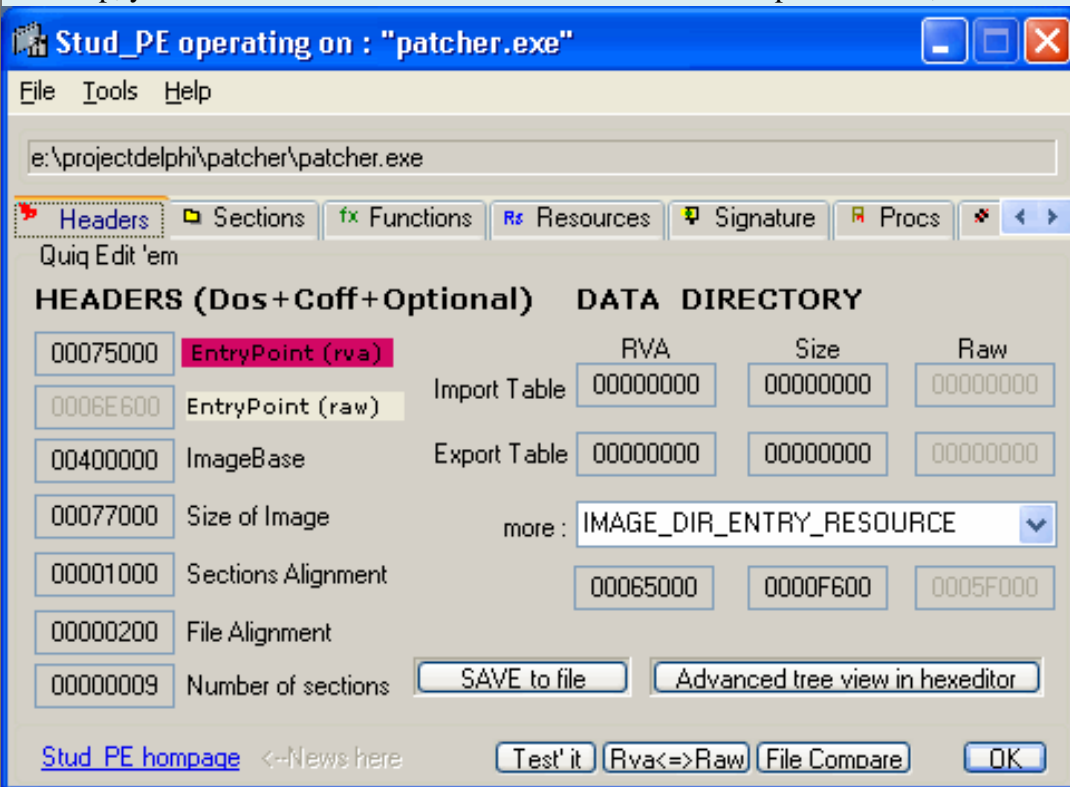
Before unpack this packer. I have to pack some application. I choose my packer code in Delphi. I choose all options for this packer pack. Par example: Anti API Breakpoint, Anti Si ... See here:

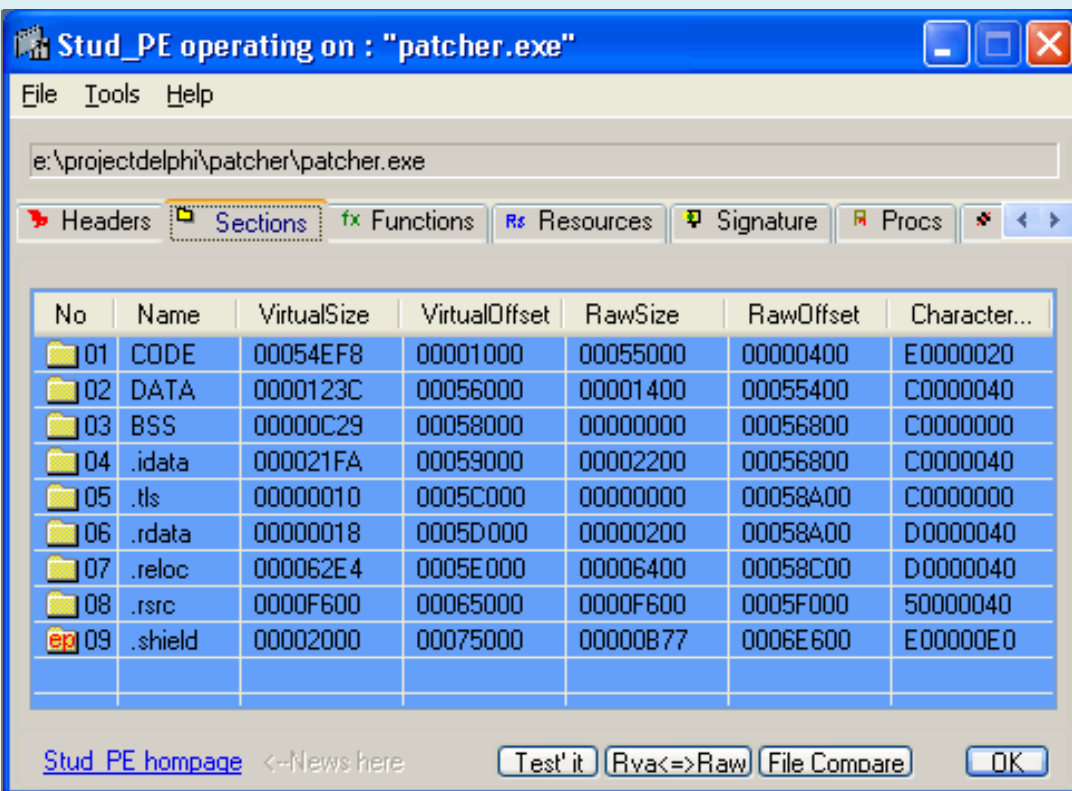


Now, try to detect with PEiD 0.92 :-).



First step, you have to find some info from this PE software. Open Lord PE, PE Editor choose. And we have:





EP: 75000, flags The value of this case is not needed, Image Base is always 400000, Import Table: 0000 and size is 00.

3. Finding the OEP

Load **unpackme.exe** into OllyDBG. And you still here:

```

00472000  E8 04000000  CALL patcher.00472009
00472005  8360 EB 0C   AND DWORD PTR DS:[EAX-15],0C
00472009  5D          POP EBP
0047200A  75 05       JMP SHORT patcher.00472011
0047200C  45         INC EBP
0047200D  55         PUSH EBP
0047200E  75 04       JMP SHORT patcher.00472014

```

Then, you press **F7** until you see as follows:

```

0047200A  75 05       JMP SHORT patcher.00472011

```

And then, you press **Shift + F9 1 time**. Now you still here:

```

00472B52  8B00       MOV EAX,DWORD PTR DS:[EAX]
00472B54  75 01       JMP SHORT patcher.00472B57
00472B56  845414 1E     TEST BYTE PTR SS:[ESP+EDX+1E],DL
00472B5A  0000      ADD BYTE PTR DS:[EAX],AL
00472B5C  C045 00 10  ROL BYTE PTR SS:[EBP],10
00472B60  C045 00 C0  ROL BYTE PTR SS:[EBP],0C0
00472B64  60        PUSHAD
00472B65  45        INC EBP

```

Shift constant out of range 1..31

Continued, press **F9**.

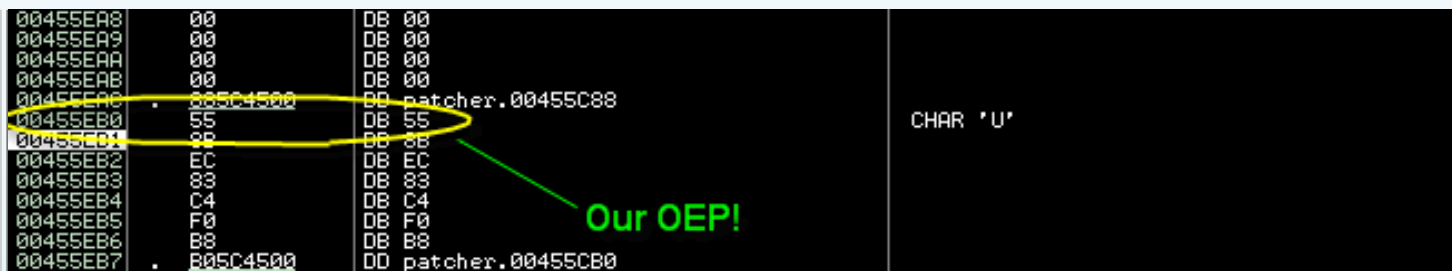
And then press **Shift + F9**. You still here

```

00455EB1  8BEC       MOV EBP,ESP
00455EB3  83C4 F0    ADD ESP,-10
00455EB6  B8 005C4500 MOV EAX,patcher.00455CB0
00455EB8  E8 0403FBFF CALL patcher.004061C4
00455EC0  A1 C8704500 MOV EAX,DWORD PTR DS:[4570C8]
00455EC5  8B00       MOV EAX,DWORD PTR DS:[EAX]
00455EC7  E8 24DDFFFF CALL patcher.00453BF0
00455EC9  0000 00714500 MOV EBX,DWORD PTR DS:[457100]
00455ECB  75 00       JMP SHORT patcher.00455ECB

```

Next, press **CTRL + A** analyze the code for:

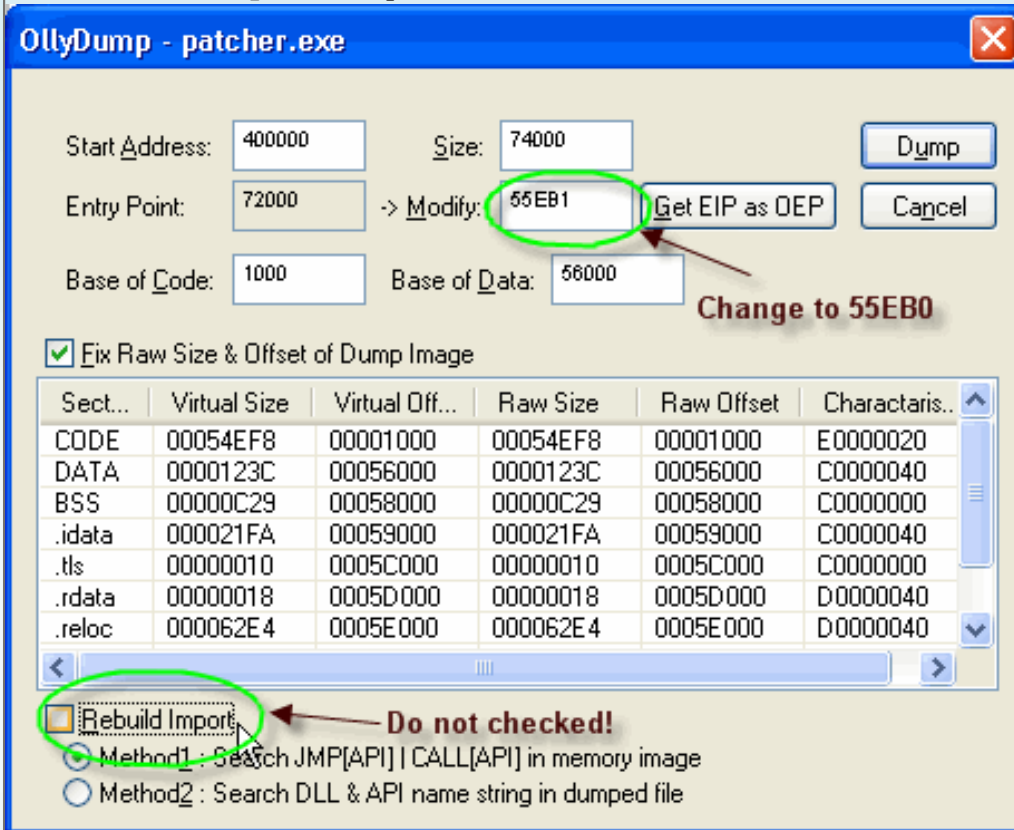


Congratulations! According OEP we found is **455EB0**. And now we Calculate the real OEP of this unpackme by the formula:

Real OEP = OEP find in Olly-Image Base = 400000 = **455EB0-55EB0**.

4th dumping

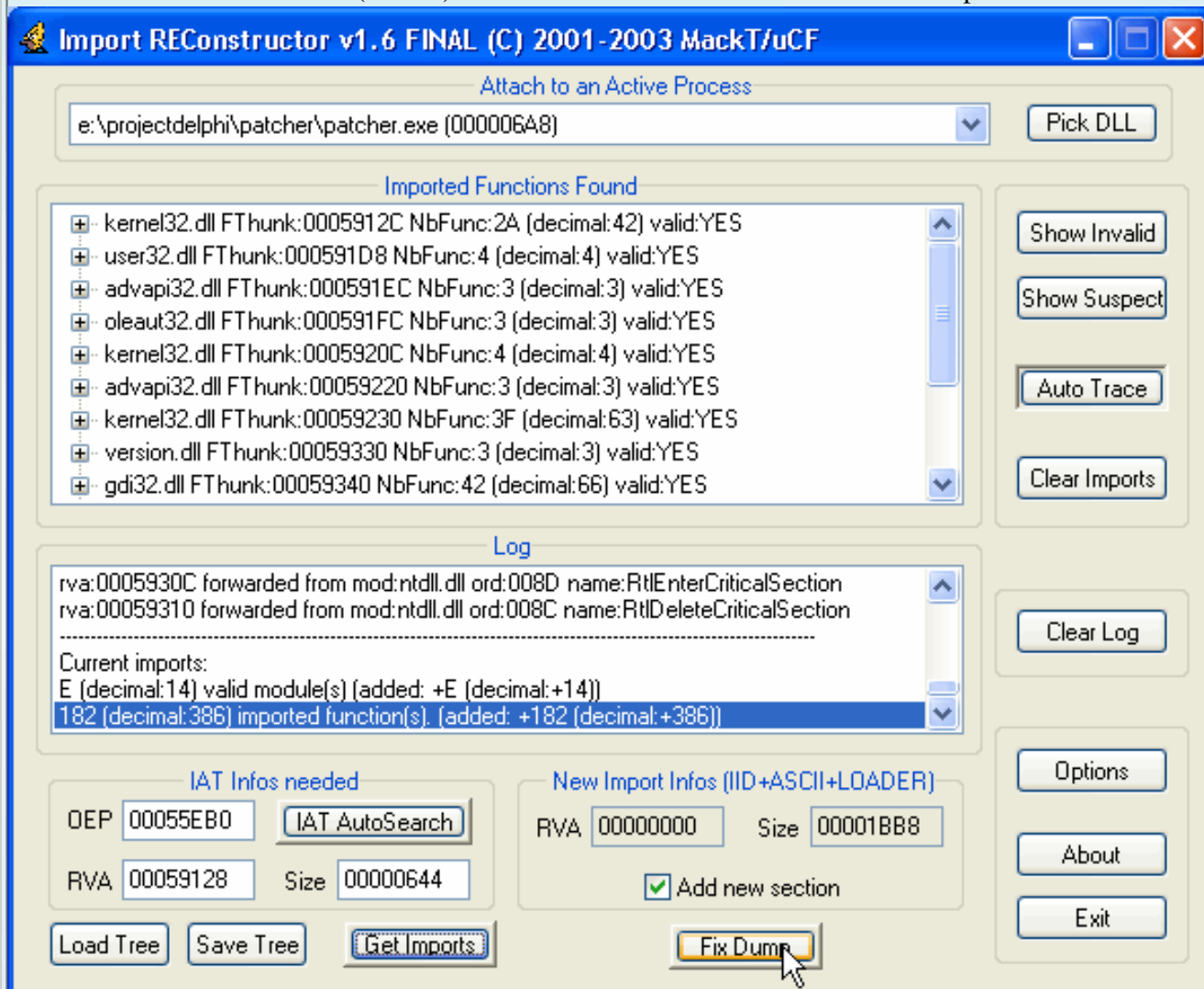
At **00455EB0** address, we go to the menu Plugin -> OllyDump -> dump debugged process. And then, just press dump, save the file at **dumped.exe** unpacked.



Do not run **dumped.exe** now, will be a crash ... It must fix IAT.

5. Finding and Fixing the Adress Import Table

And select Open ImpREC attached to active process and choose unpackme.exe. Change the value in the OEP window to the one we wrote down earlier (55EB0) then select IATAutosearch then click Get Imports.



All Import Functions valid.

Now, click fix dump to fix the IAT **dumped.exe** file.

Use LordPE 1.4 by Y0da for rebuild our Dumped_.exe

6. Testing Our Unpacked file

Now run unpacked files. Wow, not crash.

7. Conclusion

Special thanx to **koncool** et **R @ dier** for this template.

My Greetz to: Deux, NVH (c), luucorp, Maip0301, R @ dier, tlandn, Computer_Angel, Zombie, RCA, CTL, JAL, LeVuHoang, 777, LeonHart, Bin ... and you ;-)

To be continued ...

Written by [hacnho](#) (tutorial date: Sai Gon 17/4/2004)

If a code of Unipack and ECr exp italy to R 2. 2. 5 0

Why Not Bar

TGE ar t : UnPackMe_ ECr exp italy to R 2. 2. 50. J. E x e

P a c k e r E C r e x p i t a l y t o R 2. 2. 5 0 (A l l P r o t e c t i o n s a n b e l e d + 100&percent; With ual r t i z a t-ion)

T o o l s: O l l i t a l y B D X G_ E c r i t a l y t o p r, O l l D i t a l y u m 2. 2 1, O l l A d i t a l y v a n c e d 1 s t 2 6, P r o t e c t i o n I D_ t h e 5 t h 1 E, F C F E x p l o r e R V

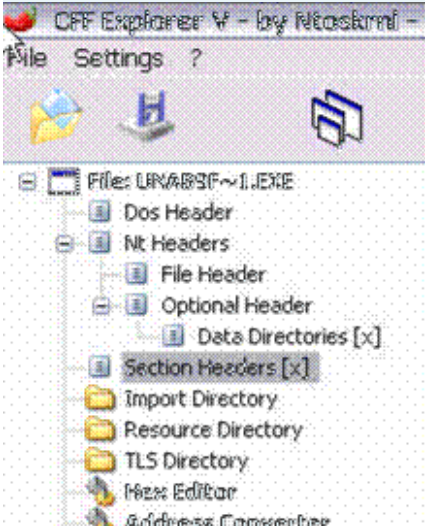
Sis has found the range for **MUP tut Newbie**, Download read about or see h ... ay Consider also find themselves **Newbie** but also want to contribute in some subjects to people with learning ...

_ Ung D o t P r e c t i o n I D _ 5.1 e S c v a n F i l e



OFF Explorer V - by Nicosini - [UNABSF~1.EXE]

File Settings ?

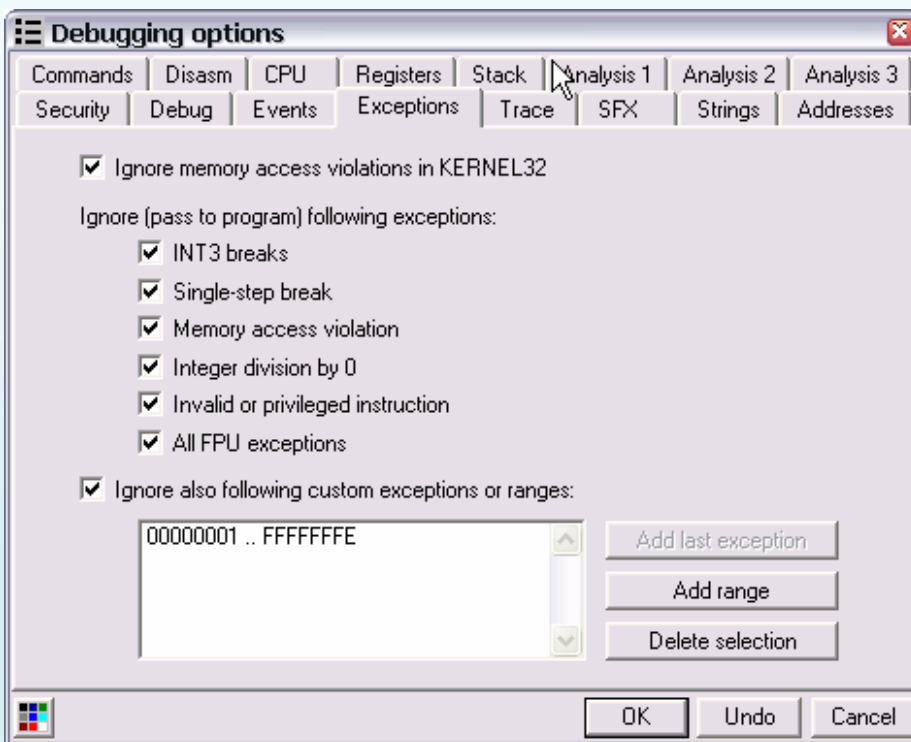


The screenshot shows the OFF Explorer V interface. On the left, a tree view displays the file structure for 'File: UNABSF~1.EXE'. The 'Section Headers [x]' are expanded, showing various sections. On the right, a table lists these sections with their names, virtual sizes, virtual addresses, raw sizes, raw addresses, and relative addresses. Four sections are highlighted with red boxes: 'eebvb3vn', 'yqiu.kuw', 'zwm09o0o', and 'e3.pv3qt'.

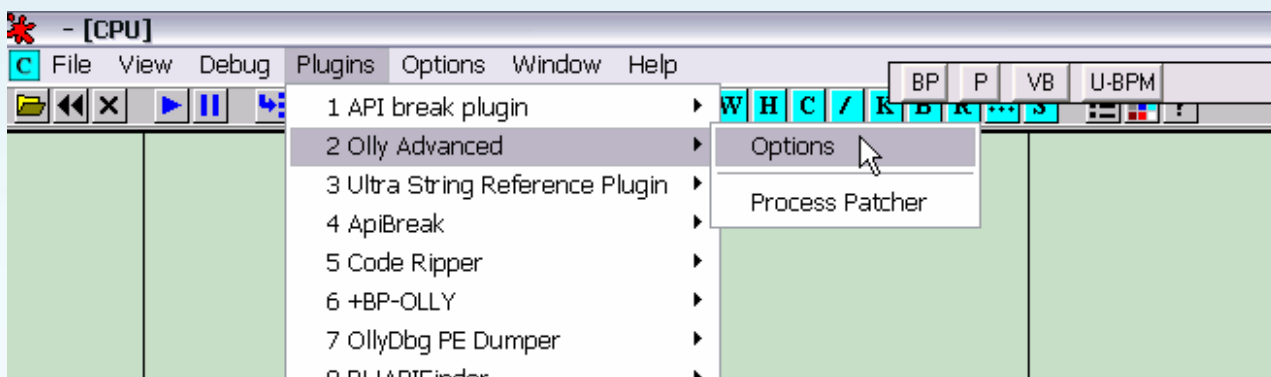
Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Rela
Bytef81	Dword	Dword	Dword	Dword	Dwc
.text	0004A000	00001000	00000000	00000400	000
.rdata	0000C000	00048000	00000000	00000400	000
.data	00009000	00057000	00000000	00000400	000
eebvb3vn	00003000	00060000	00000000	00000400	000
.rsrc	00008000	00063000	00007A00	00000400	000
yqiu.kuw	00001000	00068000	00000000	00007E00	000
zwm09o0o	0004A000	0006C000	00000000	00007E00	000
e3.pv3qt	00071000	00086000	00070930	00007E00	000

_ That is, addition **ExeCryptor** easily identify the observations of **Section** ... it's very similar ko lang ad nhucac other Packer

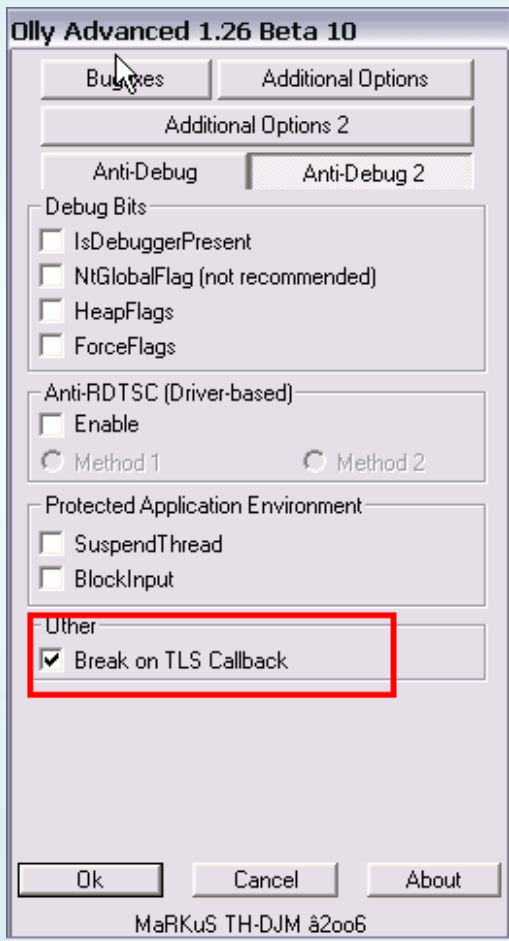
_ OK OllyDBG_ **ExeCryptor** open, press **Alt + O** and the following



_ Select **Olly Advanced** plugin and like



_ To select as follows:



_ Now Load "*UnPackMe_ExeCryptor2.2.50.j.exe*" to **OllyDBG** and you will stop here

00526911	05 8B660000	ADD EAX,668B	
00526916	FFE0	JMP NEAR EAX	
00526918	E8 EBF0FFFF	CALL UnPackMe.00526808	
0052691D	05 E55E0000	ADD EAX,5EE5	
00526922	FFE0	JMP NEAR EAX	
00526924	E8 04000000	CALL UnPackMe.0052692D	
00526929	FFFF	???	Unknown command
0052692B	FFFF	???	Unknown command
0052692D	5E	POP ESI	
0052692E	C3	RETN	
0052692F	0000	ADD BYTE PTR DS:[EAX],AL	
00526931	0000	ADD BYTE PTR DS:[EAX],AL	
00526933	0000	ADD BYTE PTR DS:[EAX],AL	
00526935	0000	ADD BYTE PTR DS:[EAX],AL	

_ Press **Alt + B** 1 Breakpoint you and delete it

Address	Module	Active	Disassembly
0052690C	UnPackMe	One-shot	:26808

Remove Del
Follow in Disassembler Enter
Copy to clipboard
Appearance

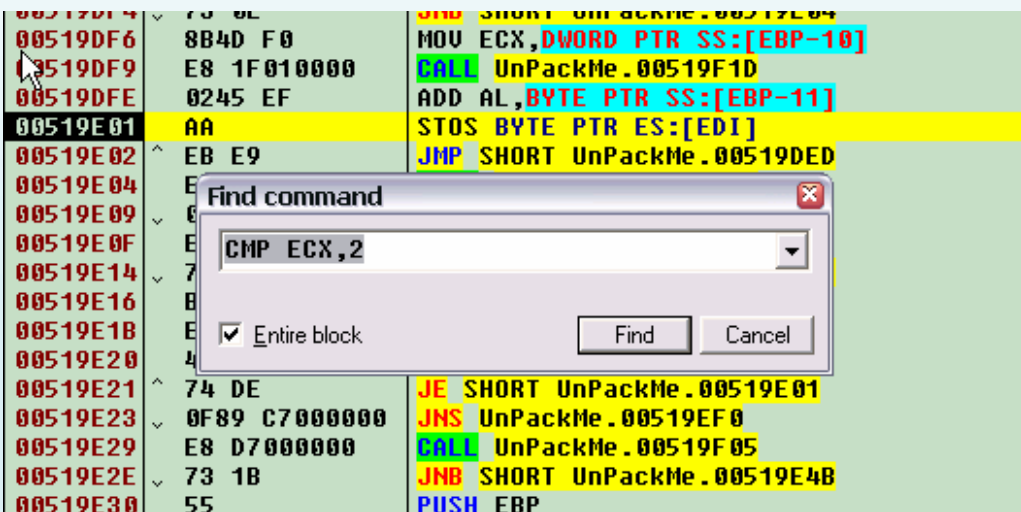
_ Press **Alt + M** and BP Set in **Section 1. Text**

00020000	00001000		Search	Ctrl+B	RW	
0012D000	00001000				RW	
0012E000	00002000		Set break-on-access	F2	RW	
00130000	00003000				R	
00140000	00004000		Set memory breakpoint on access		RW	
00240000	00006000		Set memory breakpoint on write		RW	
00250000	00003000		Set access		RW	
00260000	00016000		Allocate Memory		R	\Device\HarddiskVol1
00280000	0003D000		Free Memory		R	\Device\HarddiskVol1
002C0000	00041000		Zero Memory		R	\Device\HarddiskVol1
00310000	00006000		Dump Memory-Area		R	\Device\HarddiskVol1
00320000	00001000		Load dumped memory		RWE	
00330000	00001000				RWE	
00340000	00001000				RWE	
00350000	00001000				RWE	
00360000	00001000		Copy to clipboard		RWE	
00370000	00001000		Sort by		RW	
00380000	00001000		Appearance		RW	
00400000	00001000	UnPackMe			RWE	
00401000	0004A000	UnPackMe	.text		RWE	
0044B000	0000C000	UnPackMe	.rdata		RWE	
00457000	00009000	UnPackMe	.data	data	RWE	

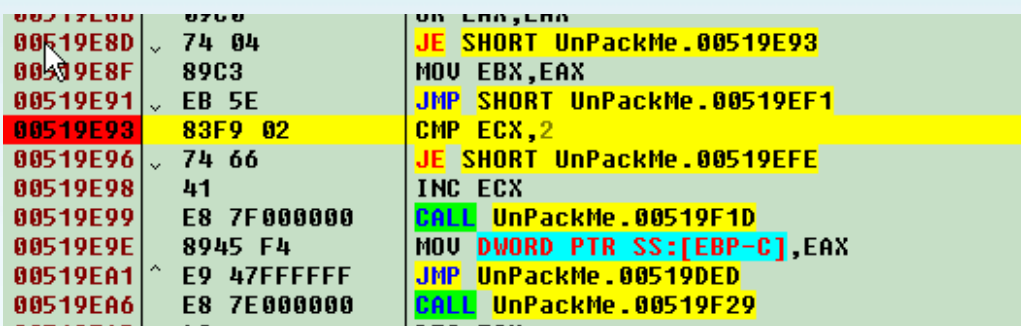
_ Press **Shift + F9** you stop here

00519DF6	8B4D F0	MOV ECX,DWORD PTR SS:[EBP-10]	
00519DF9	E8 1F010000	CALL UnPackMe.00519F1D	
00519DFE	0245 EF	ADD AL,BYTE PTR SS:[EBP-11]	
00519E01	AA	STOS BYTE PTR ES:[EDI]	
00519E02	EB E9	JMP SHORT UnPackMe.00519DED	
00519E04	E8 FC000000	CALL UnPackMe.00519F05	
00519E09	0F82 97000000	JB UnPackMe.00519EA6	
00519E0F	E8 F1000000	CALL UnPackMe.00519F05	
00519E14	73 5B	JNB SHORT UnPackMe.00519E71	
00519E16	B9 04000000	MOV ECX,4	
00519E1B	E8 FD000000	CALL UnPackMe.00519F1D	
00519E20	48	DEC EAX	
00519E21	74 DE	JE SHORT UnPackMe.00519E01	
00519E23	0F89 C7000000	JNS UnPackMe.00519EF0	
00519E29	E8 D7000000	CALL UnPackMe.00519F05	
00519E2E	73 1B	JNB SHORT UnPackMe.00519E4B	
00519F30	55	PIUSH FRP	

_ Press **Ctrl + F** to enter **Cmp ECX, 2**



_ Click **Find**, press **F2** to set it at BP



_ Press **Shift + F9 10 times** (If you hit 11 times will be Crash hichic if you do not believe can receive messages) Soft time code has been completely on the **F2** nho.Nhan BP to remove this and I find to **OEP**, press **Alt + M** to Set and BP in **Section 1. Text**

00350000	00001000				Priv	RWE	RWE
00360000	00001000				Priv	RWE	RWE
00370000	00001000				Priv	RW	RW
00380000	00001000				Priv	RW	RW
00400000	00001000	UnPackMe		PE header	Imag	R	RWE
00401000	0004A000	UnPackMe	.text		Imag	R	RWE
0044B000	0000C000	UnPackMe	.rdata		Imag	R	RWE
00457000	00009000	UnPackMe	.data	data	Imag	R	RWE
00460000	00003000	UnPackMe	eebvb3vn		Imag	R	RWE
00463000	00008000	UnPackMe	.rsrc	resources	Imag	R	RWE
0046B000	00001000	UnPackMe	ygiu.kuw		Imag	R	RWE

_ Press **Shift + F9** to you here

004271B0	- E9 6CD10500	JMP UnPackMe.00484321	
004271B5	58	POP EAX	
004271B6	C1C0 18	ROL EAX,18	
004271B9	- E9 424E0400	JMP UnPackMe.0046C000	
004271BE	70 12	JO SHORT UnPackMe.004271D2	
004271C0	6C	INS BYTE PTR ES:[EDI],DX	I/O command
004271C1	90	NOP	
004271C2	321F	XOR BL,BYTE PTR DS:[EDI]	
004271C4	ED	IN EAX,DX	I/O command
004271C5	50	PUSH EAX	
004271C6	64:8925 000000	MOV DWORD PTR FS:[0],ESP	
004271CD	83C4 A8	ADD ESP,-58	
004271D0	53	PUSH EBX	
004271D1	FA	PUSH ESI	

Hic hic _ have **Stolen Code**, but in this case do not need very banana Fix What rảo ... Dumped 1 and is run as **Horses**, use and remember **OlllyDump** plugin selected nhuhinh under house! If selected nhuday from the use **ImportREC**

Entry Point: 12690C -> Modify: 271C5 Get EIP as OEP Cancel

Base of Code: B6000 Base of Data: 4B000

☒ Fix Row Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
.text	0004A000	00001000	0004A000	00001000	E0000020
.rdata	0000C000	0004B000	0000C000	0004B000	C0000040
.data	00009000	00057000	00009000	00057000	C0000040
eebvb...	00003000	00060000	00003000	00060000	E0000060
.rsrc	00008000	00063000	00008000	00063000	40000040
ygiu.kuw	00001000	0006B000	00001000	0006B000	C0000040
zwm09...	0004A000	0006C000	0004A000	0006C000	E0000020
e3.pv3qt	00071000	000B6000	00071000	000B6000	E0000060

☒ Rebuild Import

- ☒ Method1 : Search JMP[API] | CALL[API] in memory image
- ☐ Method2 : Search DLL & API name string in dumped file

_ Click **Save** and **enter** the name, such as **Unpacked_Full** such and wait in seconds. Run test run unpack lickerish
Done !!!!!!!

W r i t e n t W h y b y t h e B N o a r (2 6 - 0 8 - 2 0 0 6)

::: [MUP EXECryptor v.2.2.6 with target: PowerArchiver 2007]:::

(_kienmanowar_)



I. Proem

Walking this brother BQT REA movement are looking for is soft to protect by EXECryptor voc try. REA also have a lot of articles on this Protector, the election is that the message of **WhyNotBar, tlandn, trickyboy** not matter in which the professional soft kill without written tut that's the J Com. People have presented different methods to MUP Execryptor, common goals are 3 methods as medical tlandn mentioned that manual manual for OEP use **OEPfinder vX.YZ** deroko by using scripts **AntiDBG Bypass OEP . txt** and add a more of the trick is to use 2 bytes is famous for **EB FE** MUP. However in the later version of the EXECryptor OEPfinder was working more and deroko are also not aware of this tool to update, use the script **AntiDBG OEP.txt Bypass** at the time was not (in general is not stable), so only the remaining two months is completely manual to find the OEP, is used **EB FE**. In this article I do not have the super-high, my brother learned to write and what I do only. I will present both for target PowerArchiver. *A reminder of what I do not do the following purposes outside academic and research so I will not bear any responsibility if you do use it to the not a good idea!* Ok13! L3t's R0ck w1th m3 J

II. Target and Tools

Target:

Name: **PowerArchiver 2007, Version 10.0**

Home site: <http://www.powerarchiver.com>

Tools:

Debugger: flyODBG

PE Tools: ProtectionID, RDG Packer Detector, CFF Explorer, LordPE, ImpRec

III. Manual Unpacking

1. MUP usual way:

_Dau First we used the program to detect find information about the target we are about to work:

_Thong Information has been passed **Protection ID v5.2c:**



_Thong Through **RDG Packer Detector**:



_Thong Of the section collected through **CFF Explorer**:

CFF Explorer V - by Ntoskrnl - [POWERARC.EXE]

File Settings ?

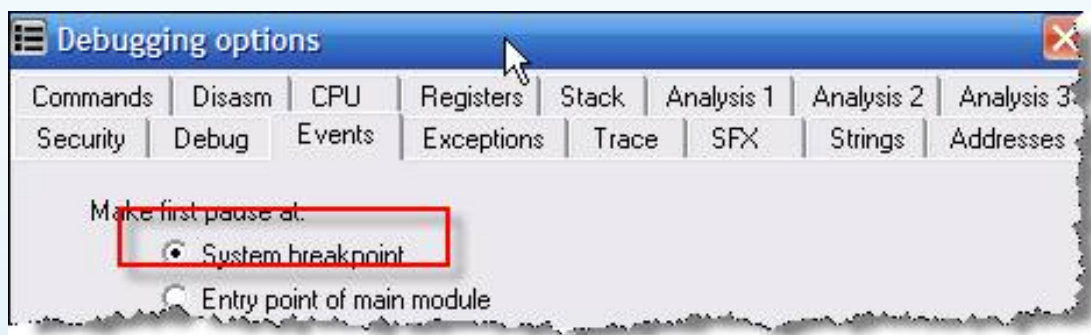
File: POWERARC.EXE

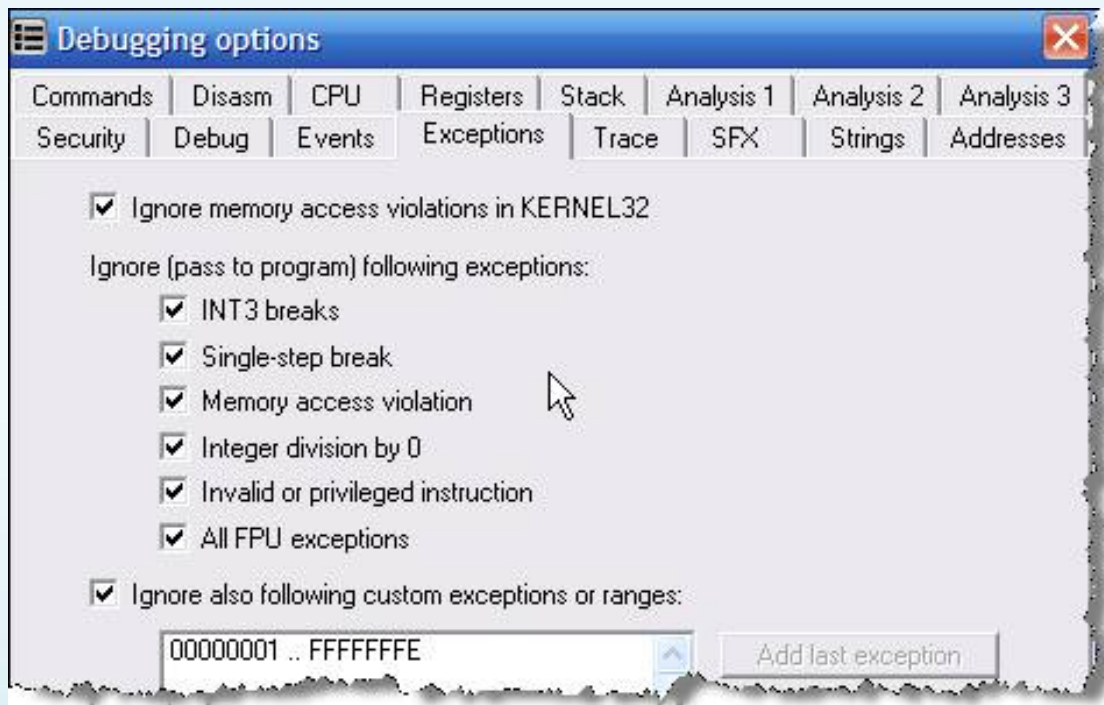
- Dos Header
- Nt Headers
 - File Header
 - Optional Header
 - Data Directories [x]
- Section Headers [x]
 - Import Directory
 - Resource Directory
 - TLS Directory
 - Hex Editor
 - Address Converter
 - Resource Viewer
 - Rebuilder
 - Import Adder

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations Number	Linenumbers Number	Characteristics
00000320	00000328	0000032C	00000330	00000334	00000338	0000033C	00000340	00000342	00000344
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
CODE	004F1000	00001000	004F0E00	00000400	00000000	00000000	0000	0000	E0000020
DATA	0001F000	004F2000	0001E600	004F1200	00000000	00000000	0000	0000	C0000040
BSS	0027A000	00511000	00000000	0050F800	00000000	00000000	0000	0000	C0000000
.idata	00005000	00788000	00004400	0050F800	00000000	00000000	0000	0000	C0000040
.dry7fyu3	00001000	00790000	00001000	00514200	00000000	00000000	0000	0000	E0000060
.ls	00001000	00791000	00000000	00515200	00000000	00000000	0000	0000	C0000000
.rdata	00001000	00792000	00000200	00515200	00000000	00000000	0000	0000	50000040
.35hdt6f3	0004A000	00793000	0004A000	00515400	00000000	00000000	0000	0000	E0000060
.rsrc	0024D000	007DD000	0000B000	005F400	00000000	00000000	0000	0000	C0000040
.r8n1dsx	000AF000	00A2A000	000AE200	0056A400	00000000	00000000	0000	0000	40000040
.gkcs6s8	0000D000	00AD9000	0000C5B8	00618600	00000000	00000000	0000	0000	E0000020
.8tzzvktu	00001000	00AE6000	00000200	00624C00	00000000	00000000	0000	0000	40000080

Through this we can conclude the PowerArchvler protected by EXEcryptor, and we also know this is the code with Borland Delphi (sometimes we do not trust 100&percent; to the detector, but the target I have this crack and then 1 time I should make sure it is using Borland Delphi code, more most of the program code with Borland has section. Code).

_Ok Collect information such as full, time we Olly configured as follows:





Olly's `_Cau` complete our target to Olly load after load programs into Olly we stop here:

Address	Hex dump	Disassembly	Comment
7C901231	C3	ret	
7C901232	8BFF	mov edi, edi	
7C901234	90	nop	
7C901235	90	nop	
7C901236	90	nop	
7C901237	90	nop	

`_Nhan` **Alt + B** to open the window BreakPoint, then remove the BP system.

Address	Module	Active	Disassembly
00BD62C8	POWERARC	One	cmp POWERARC.00B90C29

Remove Del

Follow in Disassembler Enter

Copy to clipboard

Appearance

`_Tiep` By pressing **Alt + M** to open the Memory window, you select the **section's CODE** and set a target BP as follows:

00400000	00001000	POWERARC		PE header	Imag	R	RWE
00401000	004F1000	POWERARC	COD				RWE
008F2000	0001F000	POWERARC	DAT				RWE
00911000	0027A000	POWERARC	BSS				RWE
00B8B000	00005000	POWERARC	.id				RWE
00B90000	00001000	POWERARC	dry			Ctrl+B	RWE
00B91000	00001000	POWERARC	.tl				RWE
00B92000	00001000	POWERARC	.rd			F2	RWE

Actualize

Dump in CPU

Dump

Search

Set break-on-access

_Nhan **F9** to run, we will break here in Olly:

Address	Hex dump	Disassembly	Comment
7C928515	8B06	mov eax, dword ptr ds:[esi]	POWERARC.00B939D4
7C928517	3BC3	cmp eax, ebx	
7C928519	0F85 3A2F0100	jnz ntdll.7C93B459	
7C92851F	834D FC FF	or dword ptr ss:[ebp-4], FFFFFFFF	
7C928523	E8 DA68FEFF	call ntdll.7C90EE02	
7C928528	C2 0800	ret 8	

_Nhin Down slightly you will see a command **RETN 8**, vet am moving down this command and press **F2** to set BP 1. Next you press **Shift + F9**, we will break me in order RETN.

Address	Hex dump	Disassembly
7C928515	8B06	mov eax, dword ptr ds:[esi]
7C928517	3BC3	cmp eax, ebx
7C928519	0F85 3A2F0100	jnz ntdll.7C93B459
7C92851F	834D FC FF	or dword ptr ss:[ebp-4], FFFFFFFF
7C928523	E8 DA68FEFF	call ntdll.7C90EE02
7C928528	C2 0800	ret 8

_Nhan **F2** BP to remove this, then continue to press **Alt + M** and set to a similar BP made in the above **section. Code**. Once you've set at the BP **section. Code**, press F9 to run the program, we will break here in Olly:

Address	Hex dump	Disassembly	Comment
008A23D8	9D	popfd	
008A23DC	E8 D1182F00	call POWERARC.00B93CB2	
008A23E1	E8 98903300	call POWERARC.00BD847E	
008A23E6	53	push ebx	
008A23E7	08F3	or bl, dh	
008A23E9	3F	aas	
008A23EA	93	xchg eax, ebx	
008A23EB	0BD0	or edx, eax	
008A23ED	58	pop eax	
008A23EE	F8 0B1E3000	call POWERARC.00BA41FE	

_Do The program code with **Borland Delphi 6.0 - 7.0 GetModuleHandleA** function is below. Based on this we will time to OEP of trinh.Trong Olly, in screen CPU you press **Ctrl + G** and enter:



_Nhan OK we will go to here:

Address	Hex dump	Disassembly	Comment
00401000	04 10	add al,10	
00401002	40	inc eax	
00401003	0003	add byte ptr ds:[ebx],al	
00401005	07	pop es	
00401006	42	inc edx	

_Tai Here you must click and select the image below:

Address	Hex dump	Disassembly	Comment
00401000	04 10	add al,10	
00401002	40	inc eax	
00401003	0003	add byte ptr ds:[ebx],al	
00401005	07	pop es	
00401006	42	inc edx	
00401007	6F	outs	
00401008	6F	outs	
00401009	6C	ins	
0040100A	65:61	popa	
0040100C	6E	outs	
0040100D	0100	add	
0040100F	0000	add	
00401011	0001	add	
00401013	0000	add	
00401015	0000	add	
00401017	1040 00	adc	
0040101A	05 46616C73	add	
0040101F	65:04 54	add	
00401022	72 75	jb s	
00401024	65:8D40 00	lea	
00401028	2C 10	sub	
0040102A	40	inc	
0040102B	0009	add	
0040102D	0857 69	or b	
00401030	64:	pref	
00401031	65:43	inc	
00401033	68 61720300	push	
00401038	0000	add	
0040103A	00FF	add	
0040103C	FF00	inc	
0040103E	0000 31040000	add	

_Ta Function will select second, and mouse to select the image below:

004A6DA1	call <jmp.&kernel32.GetModuleFileName@>	kernel32.GetModuleFileName@	
00406C01	call <jmp.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA	
00407934	call <jmp.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA	
00410F46	call <jmp.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA	
00411F51	call <jmp.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA	
004356CE	call <jmp.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA	
00438857	call <jmp.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA	
00467970	call <jmp.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA	

_Chung Will be in here in Olly:

Address	Hex dump	Disassembly	Comment
00407919	74 DB	je short POWERARC.004078F6	
0040791B	C3	retn	
0040791C	B8 CC208F00	mov eax, POWERARC.008F20CC	
00407921	E8 06F8FFFF	call POWERARC.0040712C	
00407926	C3	retn	
00407927	90	nop	
00407928	53	push ebx	
00407929	8BD8	mov ebx, eax	
0040792B	33C0	xor eax, eax	
0040792D	A3 C4208F00	mov dword ptr ds:[8F20C4], eax	
00407932	6A 00	push 0	
00407934	E8 2BFFFFFF	call <jmp.&kernel32.GetModuleHandleA>	
00407939	A3 68169100	mov dword ptr ds:[911668], eax	
0040793E	A1 68169100	mov eax, dword ptr ds:[911668]	
00407943	A3 D0208F00	mov dword ptr ds:[8F20D0], eax	

_Gio We will find out which function to call this code, and to vet the morning address 0x00407928. You press **Ctrl + F** and enter **Call 407,928**, similar to Figure below:

Address	Hex dump	Disassembly
00407927	90	nop
00407928	53	push ebx
00407929	8BD8	mov ebx, eax
0040792B	33C0	xor eax, eax
0040792D	A3 C4208F00	mov dword ptr ds:[8F20C4], eax
00407932	6A 00	push 0
00407934	E8 2BFFFFFF	call <jmp.&kernel32.GetModuleHandleA>
00407939	A3 68169100	mov dword ptr ds:[911668], eax
0040793E	A1 68169100	mov eax, dword ptr ds:[911668]
00407943	A3 D0208F00	mov dword ptr ds:[8F20D0], eax
00407948	33C0	xor eax, eax

Find command

call 407928

☐ Entire block

Find Cancel

Find _Nhan, Olly will take you to come up and hairbreadth you will see OEP program but is in Stolen OEP:

Address	Hex dump	Disassembly	Comment
008F18F2	8F00	pop dword ptr ds:[eax]	
008F18F4	E9 CF492E00	jmp POWERARC.<ModuleEntryPoint>	<== this is OEP
008F18F9	75 E5	jnz short POWERARC.008F18E0	
008F18FB	AB	stos dword ptr es:[edi]	
008F18FC	6A 00	push 0	
008F18FE	6A 00	push 0	
008F1900	49	dec ecx	
008F1901	75 F9	jnz short POWERARC.008F18FC	
008F1903	51	push ecx	
008F1904	B8 2C0C8F00	mov eax, POWERARC.008F0C2C	
008F1909	E8 1A6081FF	call POWERARC.00407928	
008F190E	33C0	xor eax, eax	

_Ok Is so we know the OEP of the program where then, the fix Stolen bytes we will later. Now we need to see the interest of the IAT has canceled what was not. Very easy to find the position you start and end of the OEP. According to the trick in the hip, I do have the super-high, pulling down a pull-up and conclusion IAT has not been canceled tolerable. What is healthy and J .

Address	Value	Comment
00B8B268	00000000	
00B8B26C	7C91188A	ntdll.RtlDeleteCriticalSection ← IAT Start
00B8B270	7C9010ED	ntdll.RtlLeaveCriticalSection
00B8B274	7C901005	ntdll.RtlEnterCriticalSection
00B8B278	7C809FA1	kernel32.InitializeCriticalSection
00B8B27C	7C809B14	kernel32.VirtualFree
00B8B280	7C809A81	kernel32.VirtualAlloc
00B8B284	7C80995D	kernel32.LocalFree
00B8B288	7C8099BD	kernel32.LocalAlloc
00B8B28C	7C8092AC	kernel32.GetTickCount
00B8B290	7C80A417	kernel32.QueryPerformanceCounter
00B8B294	7C8114AB	kernel32.GetVersion
00B8B298	7C809737	kernel32.GetCurrentThreadId
00B8B29C	7C809794	kernel32.InterlockedDecrement

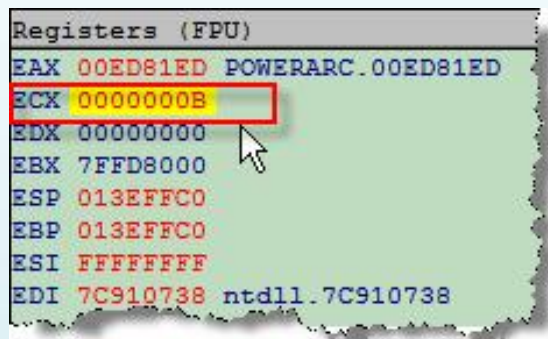
Address	Value	Comment
00B8BF38	77A9FC1F	crypt32.CertEnumCertificatesInStore
00B8BF3C	77AB8140	crypt32.CertDeleteCertificateFromStore
00B8BF40	77A9085A	crypt32.CertFreeCertificateContext
00B8BF44	77AB8A1A	crypt32.CertAddEncodedCertificateToStore
00B8BF48	77A8C19E	crypt32.CertCloseStore
00B8BF4C	77A94F78	crypt32.CertFindCertificateInStore
00B8BF50	77AB9E0A	crypt32.CertOpenSystemStoreA
00B8BF54	00000000	
00B8BF58	77DF1BD1	advapi32.CryptGetProvParam
00B8BF5C	77DEA879	advapi32.CryptImportKey
00B8BF60	77E11861	advapi32.CryptExportKey
00B8BF64	77DE8546	advapi32.CryptReleaseContext
00B8BF68	77DEA544	advapi32.CryptDestroyKey
00B8BF6C	77E11789	advapi32.CryptGetUserKey
00B8BF70	77DE7F96	advapi32.CryptAcquireContextA
00B8BF74	00000000	
00B8BF78	774F42A9	ole32.CoCreateGuid ← IAT End
00B8BF7C	00000000	

_Van On IAT is finished, we do not bother rolling tan it again. Now we go back to fix Stolen bytes. Remember that you are still in the media have not OEP nhé. Now we set a similar BP's illustrated below:

008F18F2	8F00	pop dword ptr ds:[eax]	
008F18F4	- E9 CF492E00	jmp POWERARC.<ModuleEntryPoint>	<== this is OEP
008F18F9	^ 75 E5	jnz short POWERARC.008F18E0	
008F18FB	AB	stos dword ptr es:[edi]	
008F18FC	6A 00	push 0	
008F18FE	6A 00	push 0	
008F1900	49	dec ecx	

_Tiep As you press **F9** to run the program, we will break at the place that we've set this BP. Luc observing the Registers window, you will see an information important for the fix stolen bytes

our (my way of learning lớn Why only).



F2 _Nhan to leave BP we've set, then proceed as revised picture below:

Address	Hex dump	Disassembly	Comment
008F18E8	FC	cld	
008F18E9	A2 8E000000	mov byte ptr ds:[8E], al	
008F18EE	0000	add byte ptr ds:[eax], al	
008F18F0	04 0C	add al, 0C	
008F18F2	8E00	pop dword ptr ds:[eax]	
008F18F4	55	push ebp	<== this is OEP
008F18F5	8BEC	mov ebp, esp	
008F18F7	B9 0B000000	mov ecx, 0B	
008F18FC	6A 00	push 0	
008F18FE	6A 00	push 0	

_Chuot Must address **0x008F18F4** and selected as follows:

008F18F4	55	push ebp	<== this is OEP
008F18F5	8BEC	mov ebp, esp	
008F18F7	B9 0B000000	mov ecx, 0B	
008F18FC	6A 00	push 0	
008F18FE	6A 00	push 0	
008F1900	49	dec ecx	
008F1901	75 F9	jnz short POWERARC.008F18FC	
008F1903	51	push ecx	
008F1904	B8 2C0C8F00	mov eax, POWERARC.008F0C2C	
008F1909	E8 1A60B1FF	call POWERARC.00407928	
008F190E	33C0	xor eax, eax	
008F1910	55	push ebp	
008F1911	68 631C8F00	push POWERARC.008F1C63	
008F1916	64:FF30	push dword ptr fs:[eax]	
008F1919	64:8920	mov dword ptr fs:[eax], esp	
008F191C	8D45 EC	lea eax, dword ptr ss:[ebp-14]	

_Bay Hours conducting dump file and Fix IAT thoi. Toi xài always quick to Ollydump plugin. You remember select functions rebuild Import nhé, then conducted dump and save the file with a name as you like, I put a **dumped.exe**.

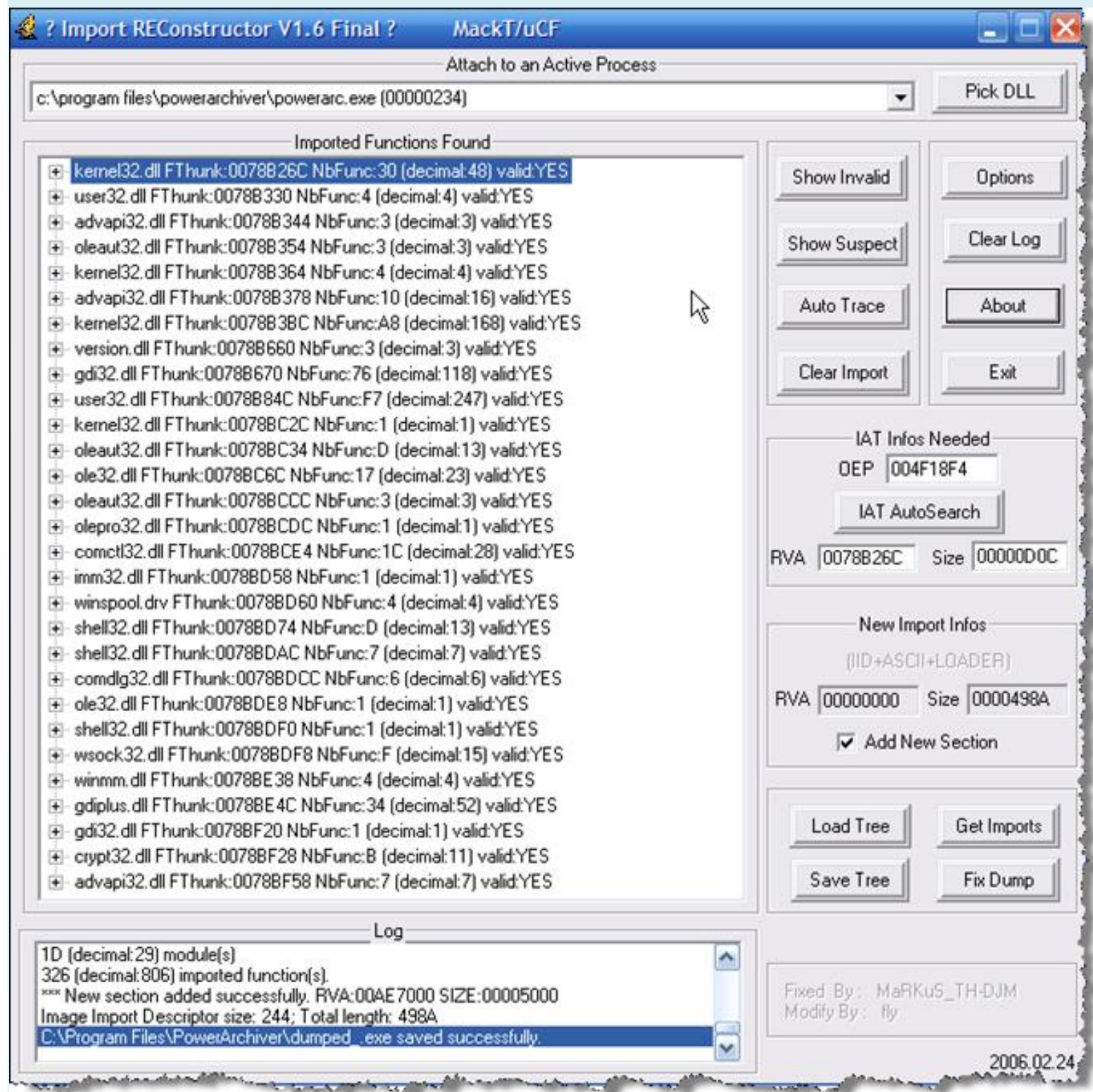
_Ok Dump file after it completed the last step is fix IAT. Before we open ImportRec calculation has little 1:

Real OEP = 0x008F18F4 - 0x00400000 = 0x004F18F4

RVA = IAT Start - 0x0078B26C = 0x00400000

Size = IAT End - IAT Start = 0x00000D0C

_Bay Open ImportRec out, select Process and complete the information on to. Then click Get Imports. Wow, so cool is not an invalid any func. Tolerable fix any dump J. Click Fix dump file and dumped.



_Bay Time we try to test the file has to fix it has not run normally nhé. Oh well run, so that we have completed the MUP J



2. Properly EB FE:

_Cach This trick is presented and that I will not go into the analysis that will always applied. The OEP find you can do or can follow the same instructions in the article by trick, but here are a particular need not fix Stolen bytes that the program is still running J. First, the load has to Olly, then remove the BP system, followed by pressing **Ctrl + G** and enter the address of OEP:

Address	Hex dump	Disassembly	Comment
7C901231	C3	ret	
7C901232	8BFF	mov edi, edi	
7C901234	90		
7C901235	90		
7C901236	90		
7C901237	90		
7C901238	90		
7C901239	CC		

Enter expression to follow

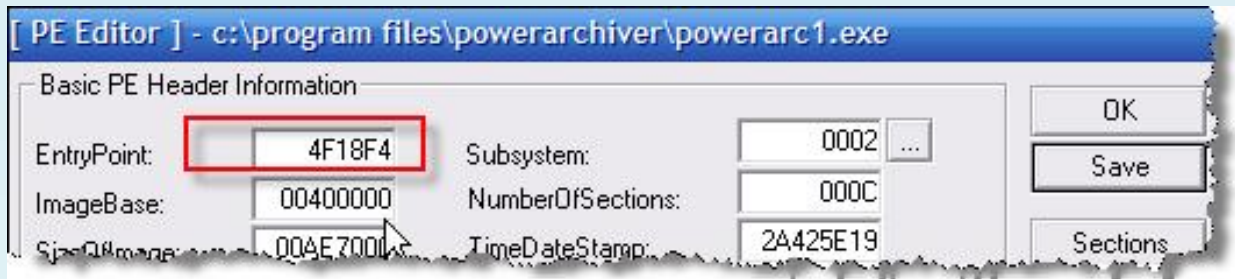
_Nhan OK, we will stay in the position of OEP:

Address	Hex dump	Disassembly	Comment
008F18F4	- E9 CF492E00	jmp POWERARC.<ModuleEntryPoint>	
008F18F9	^ 75 E5	jnz short POWERARC.008F18E0	
008F18FB	AB	stos dword ptr es:[edi]	
008F18FC	6A 80	mov byte ptr [edi], al	

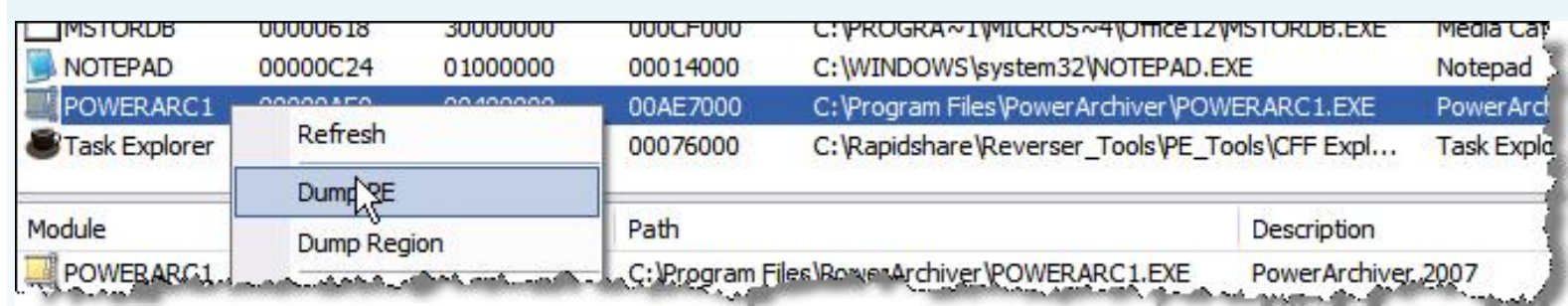
_Tai This press **Ctrl + E**, and to correct **EB FE** and remember two bytes have been replaced:

Address	Hex dump	Disassembly	Comment
008F18F4	- EB FE	jmp short POWERARC.008F18F4	
008F18F6	49	dec ecx	

_Save Things have changed again, here's my Save **POWERARC1.EXE**. Then load this file on LordPE EP and revised to correct OEP we have again and Save:



_Bay Time double click the file to run **POWERARC1.EXE** file. Now the program is falling on the status repeats endlessly, we conducted the full dump. Here I use Task Explorer to Save the dump and into **POWERARC1_Dumped.EXE**:



_Sau Dump when finished using ImportRec to fix IAT. Simply select the Process, then IAT Auto Search, the next Get Imports and finally J Fix dump. Hì hì do not jump gold fix dump file to run with, hours POWERARC1 kill the process and load the dump file fix Olly to edit the bytes have been Edit.Cuoi together with Save the name is **POWERARC_fix.EXE**.

Address	Hex dump	Disassembly	Comment
008F18F4	E9 CF492E00	jmp POWERARC.00BD62C8	
008F18F9	75 E5	jnz short POWERARC.008F18E0	
008F18FB	AB	stos dword ptr es:[edi]	
008F18FC	6A 00	push 0	
008F18FE	6A 00	push 0	

_Run Try to view the file **POWERARC_fix.EXE** nao.Kha kha has, it also run mượt J



_Phu Them, too tired, but not khiếp with the IAT cancel it: D. Over here I have presented to you 2 hours and as you see two ways this work is quite good. Now it is tired, appointment to the posts in the other. Thank you very much brother wrote the tuts to share experiences!

Best Regards

[Kienmanowar]



--+ +---==[**Greatz thanks to**]=---+ +--

My family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHQCcrker, the_Lighthouse, Merc, Hoadongnoi, Nini ... all REA's members, TQN, HacNho, RongChauA, Deux, tlandn, light.phoenix, dqtlIn, ARTEAM all my friend, and you.

Thanks to --+ +---==[]=---+ +--

iamidiot, WhyNotBar, trickyboy, dzungltn, takada, hurt_heart, haule_nth, v. hytkl. v.. You have contributed greatly to the REA. Hope you will continue to promote J

I want to thank **Teddy Roggers** for his great site, Reversing.be folks (especially **haggar**), Arteam folks (**Shub-Nigurrath**, **MaDMAN_H3rCuL3s**) and all folks on crackmes.de, thank to all members of **unpack.cn** (especially **fly** and **linhanshi**). Great thanks to **lena151** (I like your tutorials). And finally, thanks to **Ricardo NARVAJA** and all members on **CRACKSLATINOS**.

>>>> If you have any suggestions, comments or corrections email me: **kienmanowar [at] reaonline.net**

computer_angel unpacking series

Manual unpacking EZIP 1.0 -> Jonathan Clark [Overlay]

Information	Unpacking series for Newbie
Target	VBReformer
Available	None
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, PEiD 0.92.
Protection	EZIP 1.0 -> Jonathan Clark [Overlay]
L Evel	Easy
Category	Manual unpacking

1. Introduction

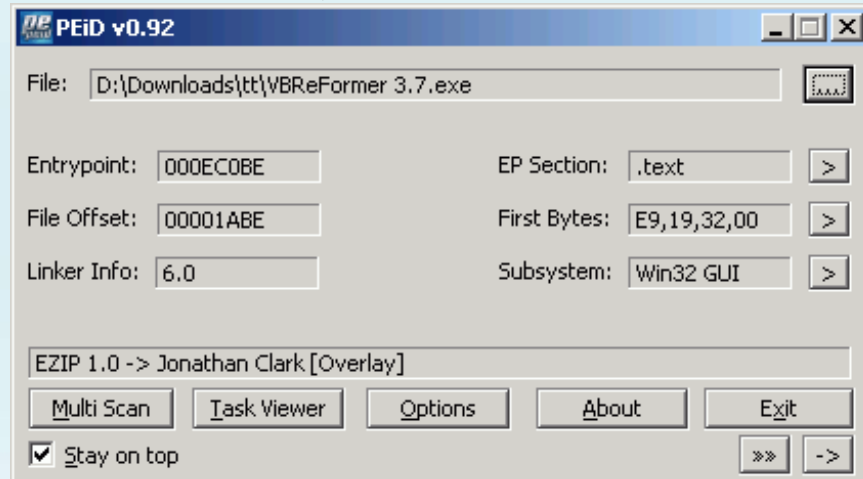
EZIP 1.0 is a free packer with some features:

- Compressed EXEs are typically **30-50%** their original size.
- Compressed EXEs run as normal. No special files or drivers need.
- Compressed programs are more difficult to reverse engineer
- Compressed programs load faster from network drives or CDROM
- Ezip is easy to install and uninstall.
- Ezip is completely free! Other similar programs cost from \$ 50 to \$ 200.

Today, I will show you the way to unpack for this packer. It is very easy! (The GUI of this packer is nice too.)

2. Getting Started

Use PEiD to detect the packer of VBReformer, now you see it said: EZIP 1.0 -> Jonathan Clark [Overlay]

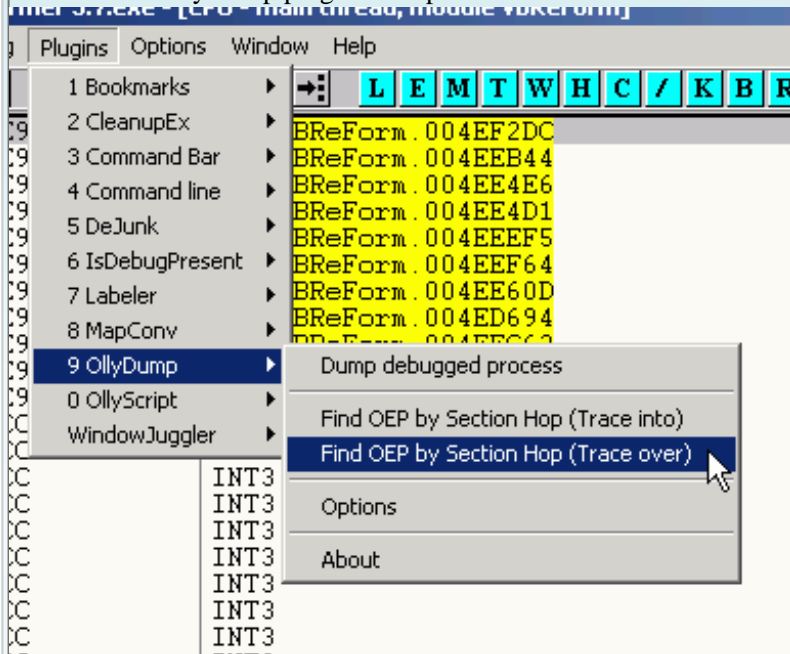


3. Finding the OEP

First load **VBReformer.exe** into OllyDBG. And you see a lot of instruction JUMP:

004EC0BE	\$	E9 19320000	JMP VBReForm.004EF2DC
004EC0C3	.	E9 7C2A0000	JMP VBReForm.004EEB44
004EC0C8	\$	E9 19240000	JMP VBReForm.004EE4E6
004EC0CD	\$	E9 FF230000	JMP VBReForm.004EE4D1
004EC0D2	.	E9 1E2E0000	JMP VBReForm.004EEEF5
004EC0D7	\$	E9 882E0000	JMP VBReForm.004EEF64
004EC0DC	\$	E9 2C250000	JMP VBReForm.004EE60D
004EC0E1	\$	E9 AE150000	JMP VBReForm.004ED694
004EC0E6	\$	E9 772B0000	JMP VBReForm.004EEC62
004EC0EB	\$	E9 87020000	JMP VBReForm.004EC377
004EC0F0	\$	E9 702E0000	JMP VBReForm.004EEF65
004EC0F5	CC		INT3
004EC0F6	CC		INT3
004EC0F7	CC		INT3
004EC0F8	CC		INT3
004EC0F9	CC		INT3
004EC0FA	CC		INT3

Now use the OllyDump plugin to help us find the real OEP:



This will take us to PUSHAD instruction nicely. Now we will have to use the trick of the PUSHAD. For that it does not know becomes one since I explain it: In the first place you must step in and press the F8 PUSHAD so that it is this executed. After you will see that ESP (in the window of registry) becomes color from red , compared with beams click the right button menu and select "Follow in dump"

The screenshot shows the 'Breakpoint' menu in OllyDbg. The 'Breakpoint' option is selected, opening a submenu. In this submenu, 'Hardware, on access' is selected, which has opened another submenu. In this third-level submenu, 'Dword' is selected, highlighted in blue. The background shows assembly code and a hex dump.

file:///C:/RCE%20Unpacking%20eBook%20[Tran...iumLi]/Manual%20unpacking%20EZIP%201.0.htm (3 of 5) [1/9/2009 9:45:14 LithiumLi]

00402BAC	68 4C8D4000	PUSH VBRForm.00408D4C	
00402BB1	E8 F0FFFFFF	CALL VBRForm.00402BA6	JMP to MSVBVM60.ThunRTMai
00402BB6	0000	ADD BYTE PTR DS:[EAX],AL	
00402BB8	48	DEC EAX	
00402BB9	0000	ADD BYTE PTR DS:[EAX],AL	
00402BBB	0030	ADD BYTE PTR DS:[EAX],DH	
00402BBD	0000	ADD BYTE PTR DS:[EAX],AL	
00402BBF	0040 00	ADD BYTE PTR DS:[EAX],AL	
00402BC2	0000	ADD BYTE PTR DS:[EAX],AL	
00402BC4	0000	ADD BYTE PTR DS:[EAX],AL	
00402BC6	0000	ADD BYTE PTR DS:[EAX],AL	
00402BC8	99	CDQ	
00402BC9	1981 F0E1D4BC	SBB DWORD PTR DS:[ECX+BCD4E1F0],EAX	
00402BCF	4A	DEC EDX	
00402BD0	AD	LODS DWORD PTR DS:[ESI]	
00402BD1	F9	STC	
00402BD2	02D3	ADD DL,BL	

4th dumping

At address **00402BAC** we go to the menu Plugin -> OllyDump -> dump debugged process. And then, just press dump, save the file at **dumped.exe** unpacked.

OllyDump - VBRFormer 3.7.exe

Start Address: 400000 Size: F6000 Dump

Entry Point: ECOBE -> Modify: 2BAC Get EIP as OEP Cancel

Base of Code: EC000 Base of Data: F1000

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
UPX0	000A4000	00001000	000A4000	00001000	E0000080
UPX1	00045000	000A5000	00045000	000A5000	E0000040
.rsrc	00002000	000EA000	00002000	000EA000	C0000040
.text	00004706	000EC000	00004706	000EC000	60000020
.rdata	000003CC	000F1000	000003CC	000F1000	40000040
.data	00004000	000F2000	00004000	000F2000	C0000040

☒ Rebuild Import

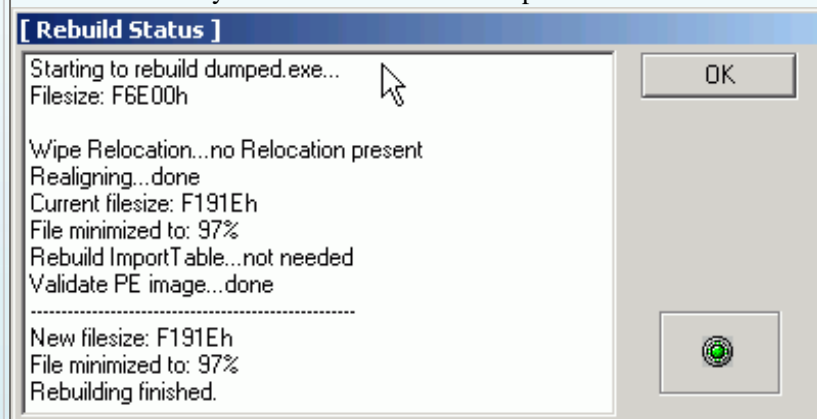
☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

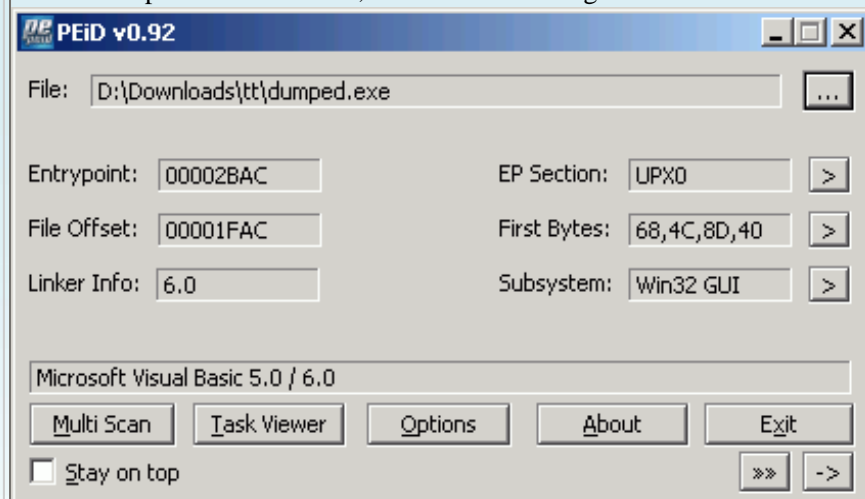
Because we fix IAT was automatically on OllyDump. So, not needed to use ImportREC!

5. Rebuild PE

Use LordPE 1.4 by Y0da for rebuild our dumped.exe



Now run unpacked files. Wow, not crash. Detect again:



6. Conclusion

Special thanx to **R @ dier** for this template.

My Greetz to: Deux, infinite, R @ dier, tlandn, Computer_Angel, Zombie, RCA, Moonbaby, ... and you ;-)!

I'll be back ...

Written by [Computer_Angel](#) (tutorial date: TPHCM 30/5/2004)

hacnho Tutorials # 1

Manual unpacking FSG 1.0 template by R @ dier.

Information	Unpacking for Newbie's
Target	unpackme.exe
Available	http://nhandan.info/hacnho/tuts/unpackme1_tuts.zip
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, ImpRec 1.6 for XP, Lord PE 1.4.
Protection	FSG by 1.0 dulek / XT
L Evel	Easy
Category	Manual unpacking

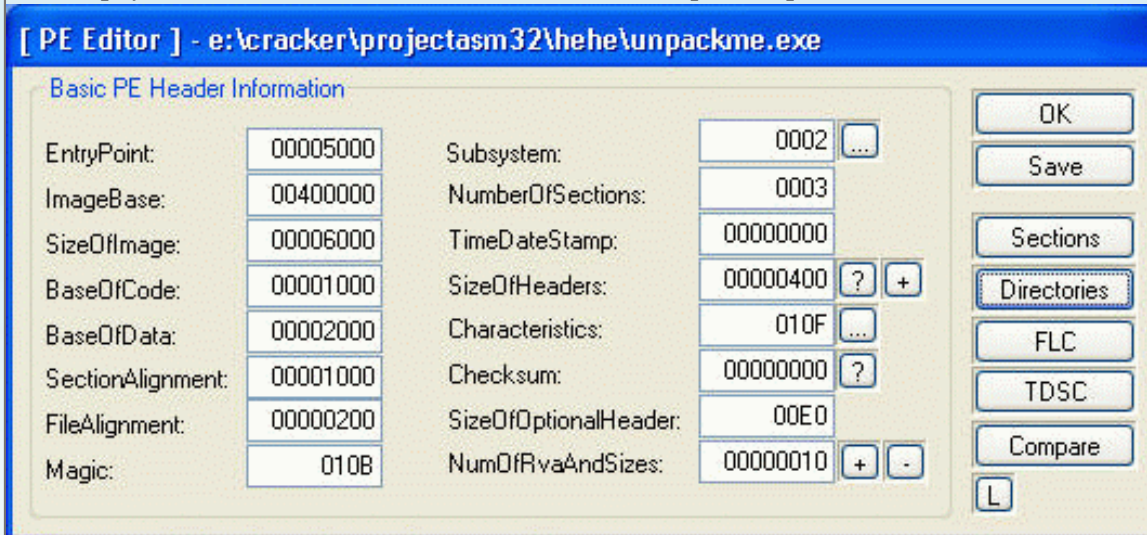
1. Introduction

Salut tout le monde!

This is my first tut for manual Unpacking. In this tut, I will introduction to manual unpacking FSG 1.0. A packer for easy learning unpack Tech.

2. Getting Started

First step, you have to find some info from this PE me unpack. Open Lord PE, PE Editor choose. And we have:



[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
	00001000	00003000	00000000	00000000	C00000E0
	00004000	00001000	00000400	000000BA	C00000E0
	00005000	00001000	00000200	00000200	C00000E0

and we have:

OEP: 5000, All Flags is C00000E0, Image Base is always 400000, Import Table: 50F4 and size is 6B.

3. Finding the OEP

We have to configure the options in SFX Olly. Press Alt + O and modify:

Commands	Disasm	CPU	Registers	Stack	Analysis 1	Analysis 2	Analysis 3
Security	Debug	Events	Exceptions	Trace	SFX	Strings	Addresses

When main module is self-extractable:

☐ Extend code section to include extractor

☐ Stop at entry of self-extractor

☐ Trace real entry blockwise (inaccurate)

☒ Trace real entry bitwise (very slow!)

☒ Use real entry from previous run

☐ Pass exceptions to SFX extractor

Now, this load unpackme in to Olly. And we still here:

<pre> 00401000 . 6A 00 PUSH 0 00401002 . 68 00304000 PUSH unpackme.00403000 00401007 . 68 1E304000 PUSH unpackme.0040301E 0040100C . 6A 00 PUSH 0 0040100E . E8 0D000000 CALL unpackme.00401020 00401013 . 6A 00 PUSH 0 00401015 . E8 0D000000 CALL unpackme.0040101A 0040101A . FF25 00204000 JMP DWORD PTR DS:[402000] 00401020 \$- FF25 08204000 JMP DWORD PTR DS:[402008] 00401026 . 00 DB 00 00401027 . 00 DB 00 00401028 . 00 DB 00 </pre>	<pre> Style = MB_OK!MB_APPLMODAL Title = "VCT-Vietnamese Crackers TEAM!" Text = "Try to Unpack me! Packed with " hOwner = NULL MessageBoxA ExitCode = 0 ExitProcess kernel32.ExitProcess user32.MessageBoxA </pre>	Registers (F) EAX: 00000000 ECX: 77F51B2B EDX: 77FC5024 EBX: 00405128 ESP: 0012FFC4 EBP: 77D40000 ESI: 0040200C EDI: 004020A4 EIP: 00401000
--	--	---

Wow, I see the OEP unpackme of this! This is:

Real OEP = OEP find in Olly-Image Base = 401000-400000 = **1000**.

4th dumping

Print Line 00401000. 6A 00 PUSH 0. Press F2 for set a breakpoint. Then, press F9 to run unpackme. When unpackme running, we go to the menu Plugin -> OllyDump -> dump debugged process.

OllyDump - unpackme.exe

Start Address: Size:

Entry Point: -> Modify:

Base of Code: Base of Data:

☒ Fix Raw Size & Offset of Dump Image

Sect...	Virtual Size	Virtual Off...	Raw Size	Raw Offset	Characteris...
	00003000	00001000	00003000	00001000	C00000E0
	00001000	00004000	00001000	00004000	C00000E0
	00001000	00005000	00001000	00005000	C00000E0

change to 1000

change to C0000040 (aply to 3 flags)

☒ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Raw Size	Raw Offset	Characteris...
00003000	00001000	C00000E0
00001000	00004000	C00000E0
00001000	00005000	C00000E0

Edit Section

Section Values

Name:

Virtual Size:

Virtual Offset:

Raw Size:

Raw Offset:

Characteristics:

Characteristics of the Section

☒ contains code

☒ contains initialized data

☒ contains uninitialized data

☐ shareable

☐ executable

☒ readable

☒ writeable

Change modify the Entry Point to **1000**. And change all flags (characteristics in Olly) to **C0000040**. And then, just press dump, save the unpacked files. Now, do not shut down OllyDbg just yet, we need to get the import table and fix our exe.

5. Finding and Fixing the Address Import Table

ImpREC open, select Print ImpRec attached to active process and choose our target program.
Change the value in the OEP window to the one we wrote down earlier (**1000**)
then select IAT Autosearch then click Get Imports.

The screenshot shows the ImpREC application interface. At the top, there's a section titled "Attach to an Active Process" with a dropdown menu showing "e:\cracker\tutocrack>manual unpacking\heh\unpackme.exe (00000634)" and a "Pick DLL" button. Below this is the "Imported Functions Found" section, which lists two entries: "kernel32.dll FTunk:00002000 NbFunc:1 (decimal:1) valid:YES" and "user32.dll FTunk:00002008 NbFunc:1 (decimal:1) valid:YES". To the right of this list are buttons for "Show Invalid", "Show Suspect", "Auto Trace", and "Clear Imports". Below the imported functions is a "Log" section with a text area showing "Original IAT RVA found at: 00002008 in Section RVA: 00001000 Size:00003000" and "IAT read successfully.". To the right of the log is a "Clear Log" button. Below the log is the "IAT Infos needed" section with input fields for "OEP" (00001000), "RVA" (00001FFC), and "Size" (00000014), along with an "IAT AutoSearch" button. To the right of this is the "New Import Infos (IID+ASCII+LOADER)" section with input fields for "RVA" (00000000) and "Size" (00000072), a checked "Add new section" checkbox, and a "Fix Dump" button. At the bottom left are buttons for "Load Tree", "Save Tree", and "Get Imports". At the bottom right are buttons for "Options", "About", and "Exit".

Ohh, all imports is valid. We have to do now is fix our exe click on **Fix dump** and select our unpacked.exe and we are done :-) our dump will be saved as unpacked_.exe and will now run.

6. Testing Our Unpacked file

Now run unpackme.exe. Wow, not crash. Using PEiD 0.92 detect: **MASM32 / TASM32**. Okie, FSG 1.0 is now unpacked successful!

7. Conclusion

Special thanx to [R @ dier](#) for this template.

My Greetz to: Deux, RCA, Moonbaby, Computer_Angel, tlandn, Zombie, Maip0301, tykhung, softcracker_vn, CTL, LeVuHoang ...

To be continued ...

Written by [hacnho](#) (tutorial date: Sai Gon 15/2/2004)

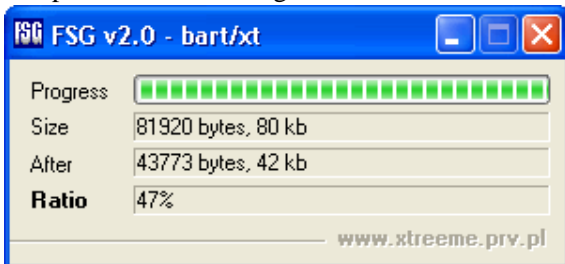
hacnho Tutorials # 14

Manual unpacking FSG 2.0 - > bart / XT

Information	Unpacking for Newbie's
Target	target1 and target2
Available	http://nhandan.info/hacnho/tuts/unpackme14_tuts.zip
Tools	1:10 OllyDbg plugin with Final OllyDump 2.21.108, OllyScript, Lord PE 1.4, 0.92 and PEiD EM Editor for 4:04 write scripts.
Protection	FSG 2.0 -> bart / XT
Level	Standard
Category	Manual unpacking

1.Introduction

Most Crackers, was used keygenner FSG 1.x, 2.x protect for the keygen or patcher. I was posted for a tut unpack this packer but not enough for all case. Now, let try another method for unpacking FSG 2.0



In this tutorials, I have two files: one: just fix IAT with OllyDump, two: Can not fix auto IAT, we must fix IAT by hand with ImpREC.

2. Target1

Step 1: Find OEP

Load-1 into **target** OllyDBG

Address	Hex dump	Disassembly	Comment
00400154	8725 BCA24100	XCHG DWORD PTR DS:[41A2BC], ESP	
00400159	61	POPAD	
0040015B	94	XCHG EAX, ESP	
0040015C	55	PUSH EBP	
0040015D	A4	MOVSB BYTE PTR ES:[EDI], BYTE PTR DS:	
0040015E	B6 80	MOV DH, 80	
00400160	FF13	CALL NEAR DWORD PTR DS:[EBX]	
00400162	73 F9	JNB SHORT target1.0040015D	
00400164	33C9	XOR ECX, ECX	
00400166	FF13	CALL NEAR DWORD PTR DS:[EBX]	
00400168	73 16	JNB SHORT target1.00400180	
0040016A	33C0	XOR EAX, EAX	
0040016C	FF13	CALL NEAR DWORD PTR DS:[EBX]	
0040016E	73 1F	JNB SHORT target1.0040018F	
00400170	B6 80	MOV DH, 80	
00400172	41	INC ECX	
00400173	B0 10	MOV AL, 10	
00400175	FF13	CALL NEAR DWORD PTR DS:[EBX]	
00400177	12C0	ADC AL, AL	
00400178	75 3A	JNZ SHORT target1.00400175	
0040017D	AA	STOSB BYTE PTR ES:[EDI]	
0040017E	EB E0	JMP SHORT target1.00400160	
00400180	FF53 08	CALL NEAR DWORD PTR DS:[EBX+8]	
00400183	02F6	ADD DH, DH	
00400185	83D9 01	SBB ECX, 1	
00400188	75 0E	JNZ SHORT target1.00400198	

Scroll down until you see:

004001C5	50	PUSH EAX	
004001C6	FF53 10	CALL NEAR DWORD PTR DS:[EBX+10]	
004001C9	95	XCHG EAX, EBP	
004001CA	8B07	MOV EAX, DWORD PTR DS:[EDI]	
004001CC	40	INC EAX	
004001CD	78 F3	JS SHORT target1.004001C2	
004001CF	75 03	JNZ SHORT target1.004001D4	
004001D1	FF53 0C	JMP NEAR DWORD PTR DS:[EBX+C]	
004001D4	50	PUSH EAX	
004001D5	55	PUSH EBP	
004001D6	FF53 14	CALL NEAR DWORD PTR DS:[EBX+14]	
004001D9	AB	STOSD DWORD PTR ES:[EDI]	
004001DA	EB EE	JMP SHORT target1.004001CA	
004001DC	33C9	XOR ECX, ECX	

three jump is
the special
signal of FSG

Toggle a breakpoint at **004001D1** (press F2). Press F9, Olly will be break at this **breakpoint!** Then, you can clear our breakpoint. Next step, press **F7** and you will still at OEP:

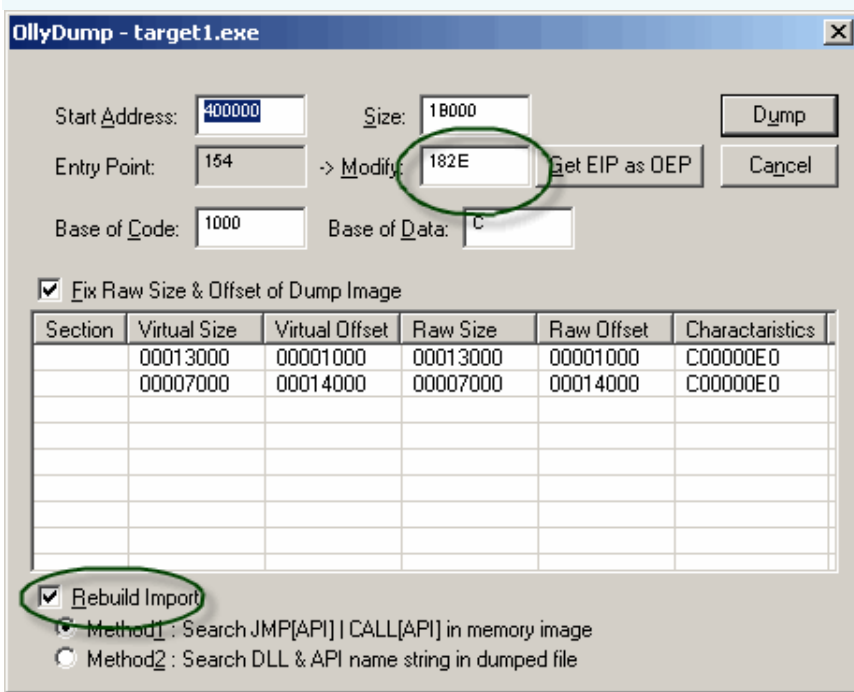
Address	Hex dump	Disassembly	Comment
0040182E	6A 00	DB 6A	CHAR 'j'
0040182F	00	DB 00	
00401830	E8	DB E8	
00401831	CD	DB CD	
00401832	01	DB 01	
00401833	00	DB 00	
00401834	00	DB 00	
00401835	A3	DB A3	
00401836	40	DB 40	CHAR '@'

Press Ctrl + A for analyze:

Address	Hex dump	Disassembly	Comment
0040182E	6A 00	PUSH 0	Module = NULL
00401830	E8 CD010000	CALL target1.00401A02	GetModuleHandleA
00401835	A3 40314000	MOV DWORD PTR DS:[403140], EAX	
0040183A	6A 00	PUSH 0	
0040183C	68 55184000	PUSH target1.00401855	
00401841	6A 00	PUSH 0	
00401843	68 C8000000	PUSH 0C8	
00401848	50	PUSH EAX	
00401849	E8 6C010000	CALL target1.004019BA	DialogBoxParamA
0040184E	6A 00	PUSH 0	
00401850	E8 A7010000	CALL target1.004019FC	ExitProcess
00401855	55	EBP	
00401856	8BEC	MOV EBP, ESP	

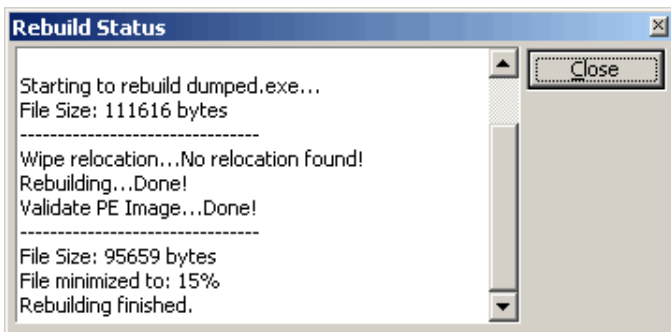
Step 2: dump and auto fix IAT

- Go to the menu plugin, choose OllyDUMP:



Step 3: rebuild PE

Use LordPE 1.4 by Y0da for rebuild our **dumped.exe**



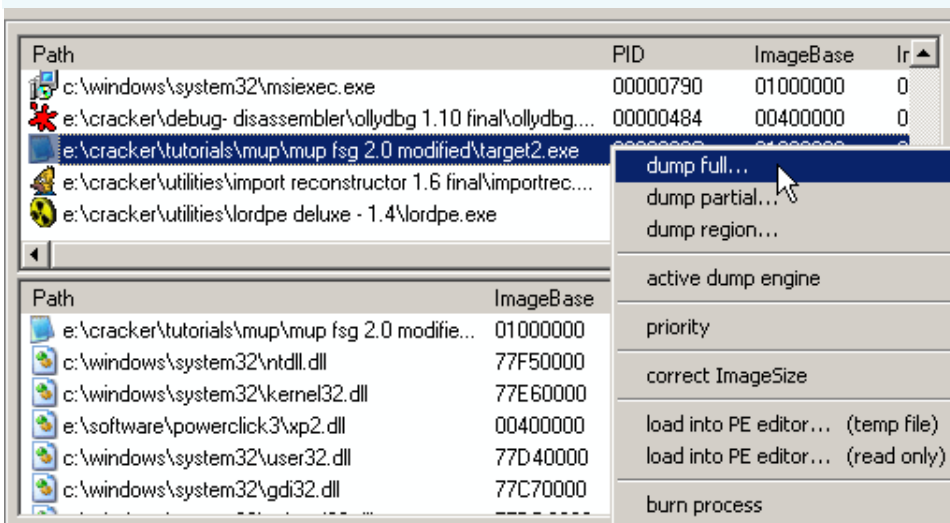
- Unpacked successful! Done ...

3.Target2

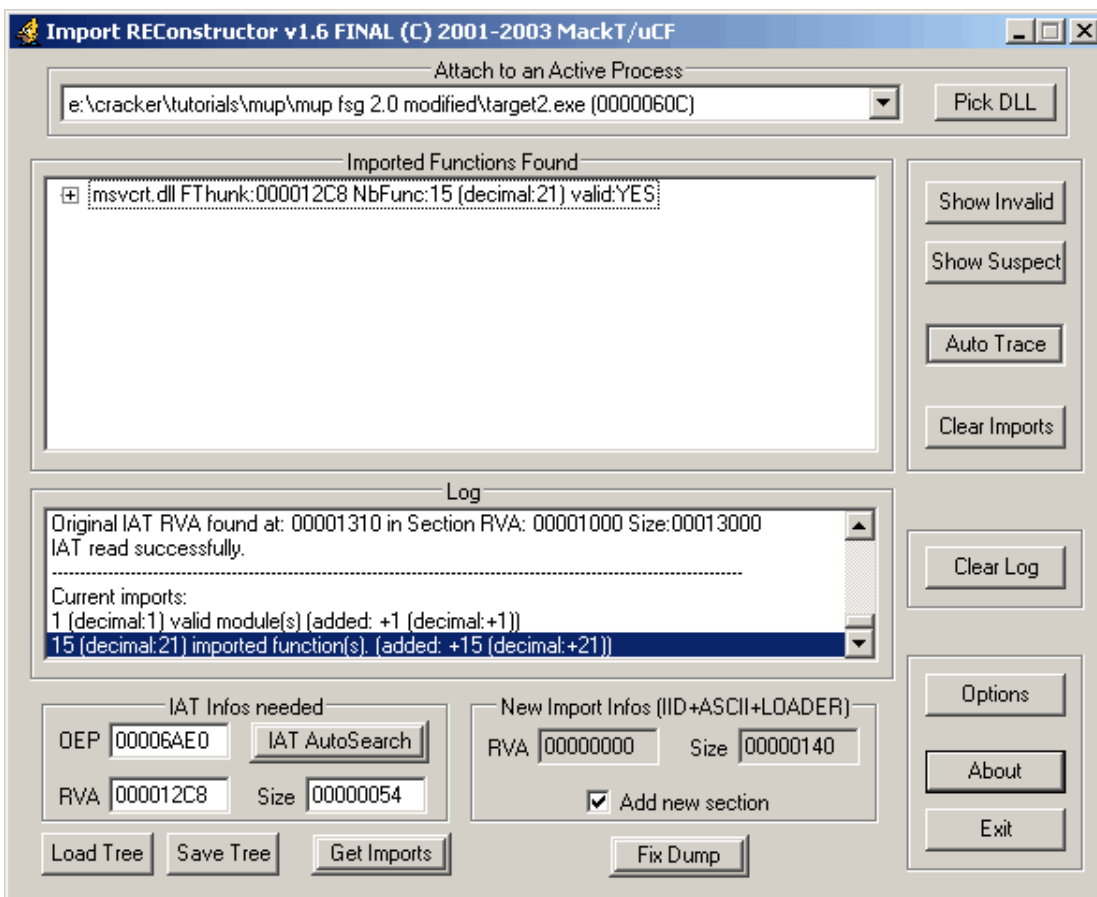
Step 1 is the same **target1**: Find the special signal, set breakpoint, jump to OEP, analyze ...

Step 2: dump and fix IAT by hand with ImpREC

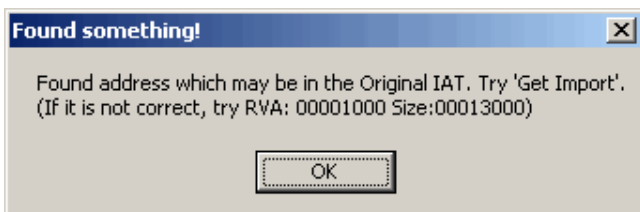
-When you still at OEP, LordPE open, right click at targer2. Choose **Full dump** ...



- Do not close Olly, now in select ImpRec attached to active process and choose the **target2.exe**. Change the value in the box OEP (OEP for the **target2.exe** is **6AE0**), then select IATAutosearch then click Get Imports.



As we will, ImpREC was found only **msvcrt.dll**. So, we know, this target was Coded in VC++ and have to some lib, par exp: user32.dll, kernel.dll, ntdll.dll, gdi32.dll ... etc! Remembers, after we press IAT AutoSearch, a dialog was appeared.



IAT Infos needed

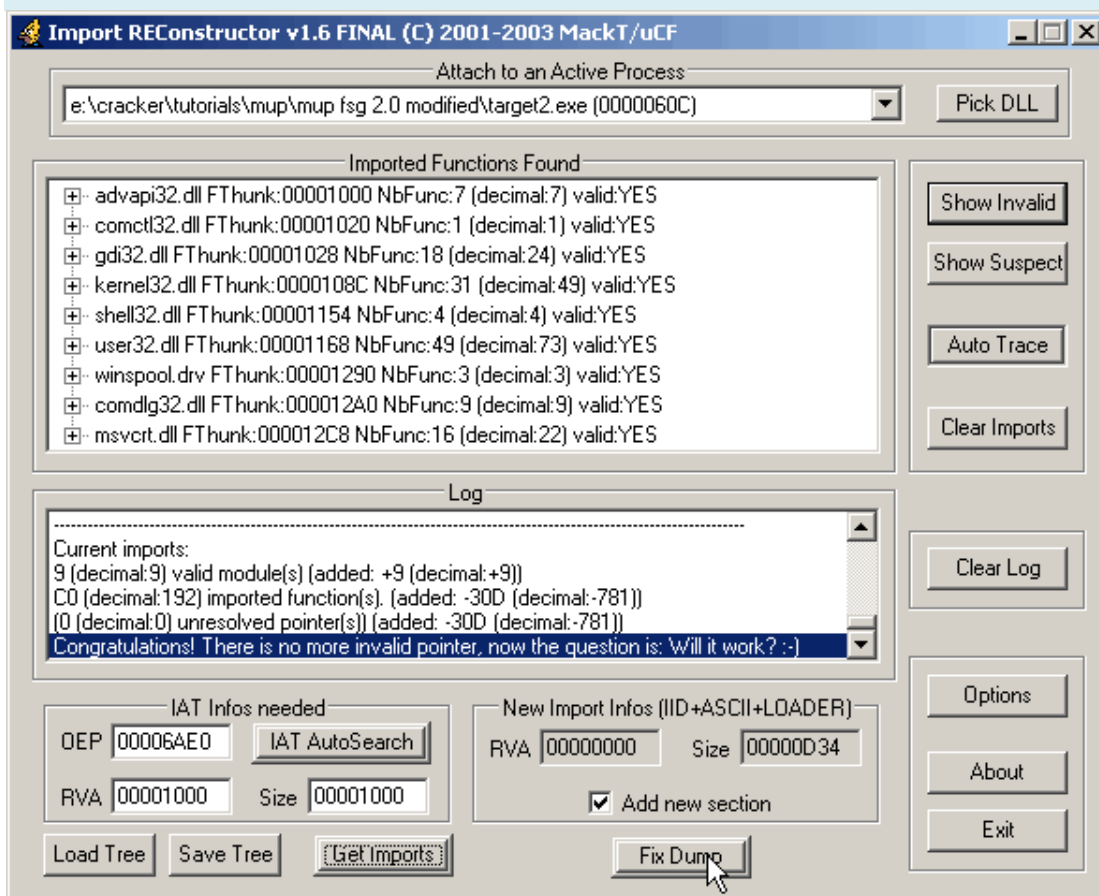
OE	00006AE0	IAT AutoSearch
RVA	00001000	Size 00001000

All import is invalid: ((.

Then, click the button **Show Invalid**. Right click at a function invalid, choose **Cut** **thunk (s)!**

The screenshot shows the 'Import REConstructor v1.6 FINAL (C) 2001-2003 MackT/uCF' application window. The main area displays a list of imported functions, including 'rva:0000101C ptr:7FFFFFFF' and 'rva:00001024 ptr:7FFFFFFF'. A context menu is open over the list, with 'Cut thunk(s)' selected. The window also features a 'Pick DLL' button, 'Show Invalid', 'Show Suspect', 'Auto Trace', 'Clear Imports', 'Clear Log', 'Options', 'About', and 'Exit' buttons. The 'IAT Info needed' section shows 'OEP' as '00006AE0' and 'RVA' as '00001000'. The 'New Import Info' section shows 'RVA' as '00000000' and 'Size' as '000002D0'. The 'Add new section' checkbox is checked.

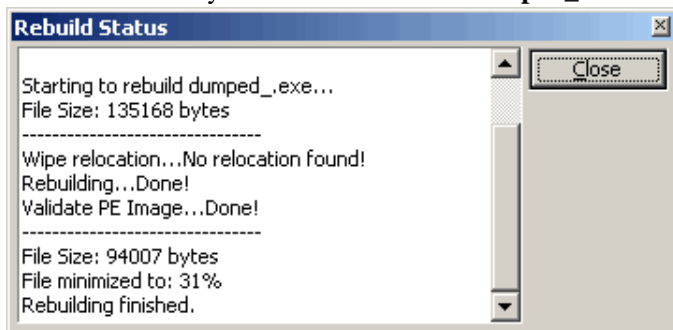
Waiting for few second ImpREC cut thanks. Wow, all is valid, good job: D.



Now, click Fix dump fix for our target!

Step 3: rebuild PE

Use LordPE 1.4 by Y0da for rebuild our **dumped_.exe**



- Unpacked successful! Done ...

4. Testing Our Unpacked file

Now run 2 unpacked files. It's Okay!

Using **PEiD 0.92** for detect: **MASM32 / TASM32** and **Micorsoft Visual C + + 7.0 Method2**. Okie, FSG 2.0is now unpacked successful!

5. Create OllyScript

After find OEP in Olly, we create a need for auto OllyScript find OEP next time! Remember! We have found three step for OEP:

- 1st First: Find the special signal of FSG for breakpoint set!
 - 2.Second: Set breakpoint, press F9 to run, clear breakpoint!
 - 3.Final: When was OllyDBG break, step into for jump to **OEP**
- Okay!**, Write in our step langue OllyScript

Cut here -----

```

/ *
//////////////////////////////// //

```

```

/ / FSG 2.0 OEP finder
/ / Author: hacnho/VCT2k4
/ / Email: hacnho@hotmail.com
/ / Website: http://nhandan.info/hacnho
/ / OS: WinXP Pro, OllyDbg 1:10 Final, OllyScript v0.85
//////////////////////////////// //
* /

```

oeb Break

```

findop eip, # FF63? # // Find the special signal, this is the JUMP
bphws $ result, "x" // Set a breakpoint on memory cute e x
run // Run the program

```

```

Bread:
bphwc $ RESULT // Clear memory breakpoint
STI // Step into (F7)
Police eip // Ctrl + A for Analyze
log eip // Logs to source OllyDbg log window.
CMT eip, "This is the OEP! Found by hacnho/VCT2k4" // Write a comment
Msg "Dumped and IAT fix now! Thanx for using my script ...!" // Show a message

```

ret // Exits script

Cut here -----

6. Conclusion

GrEeTs Fly Out: Deux, infinite, NVH (c), softcracker_vn, luucorp, Aaron, Canterwood, hhphong, R @ dier, tlandn, Computer_Angel, Zombie, RCA, CTL, Moonbaby, Nilrem, Teerayoot, Ferrari, Kruger, Devilz, anh_surprised ;-) ... and you!
 Thanx to authors of OllyDBG, ImpREC, LordPE, OllyScript, FSG, for the Canterwood tagert1 and Bill Gates for the target2;).

To be continued ...

Written by [hacnho](#) (tutorial date: Saigon 30/07/2004)

FRIENDS SITE

[\[Exetools Forum\]](#) | [\[HVAOnline\]](#) | [\[Vncracking Group\]](#) | [\[REA Forum\]](#) | [\[hacnho's homepage\]](#) | [\[AR Team\]](#)
 | [\[Vicki's Fan\]](#) | [\[Devilz Crack\]](#) |

...: Copyright © 2004 by hacnho <[=-=]> VCT-Vietnamese Cracking Team 2k4:: ..

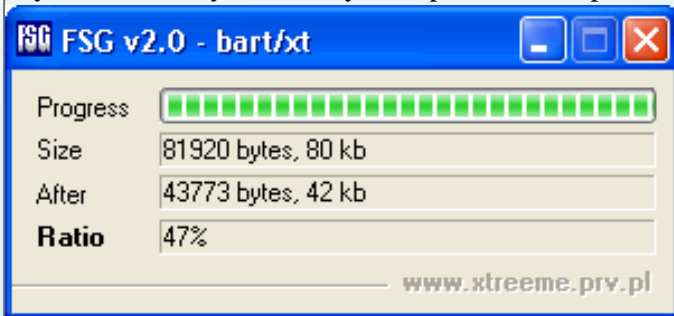
hacnho Tutorials # 12

Manual unpacking FSG v2.0 -> bart / XT

Information	Unpacking for Newbie's
Target	unpackme.exe
Available	http://nhandan.info/hacnho/tuts/unpackme12_tuts.zip
Tools	OllyDbg plugin 1.10c with OllyDump 2.21.108, LordPE 1.4, PEiD 0.92.
Protection	FSG v2.0 -> bart / XT
L Evel	Easy
Category	Manual unpacking

1. Introduction

FSG v2.0 -> bart / XT is a good pack packer for viruses: d. It was Anti Virus tools detect this virus is a packer. To day, I will show you the way to unpack for this packer. It is very easy! The GUI of this packer is nice.



2. Getting Started

Use PEiD and get some LordPE for PE Info. If your PEiD can not detect my unpackme. You can add this to your sign **userdb.txt**

;-----

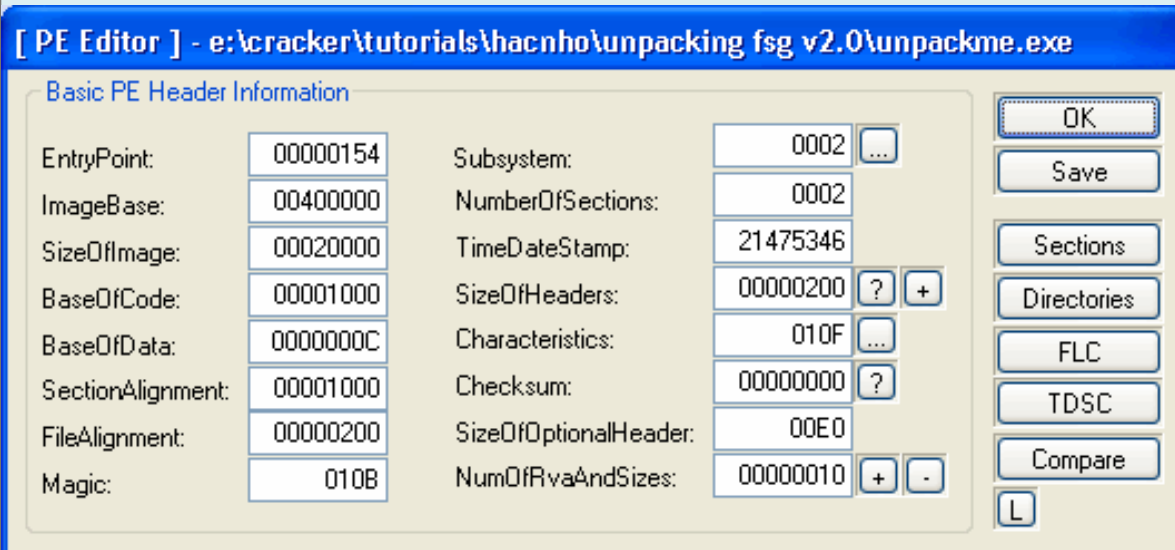
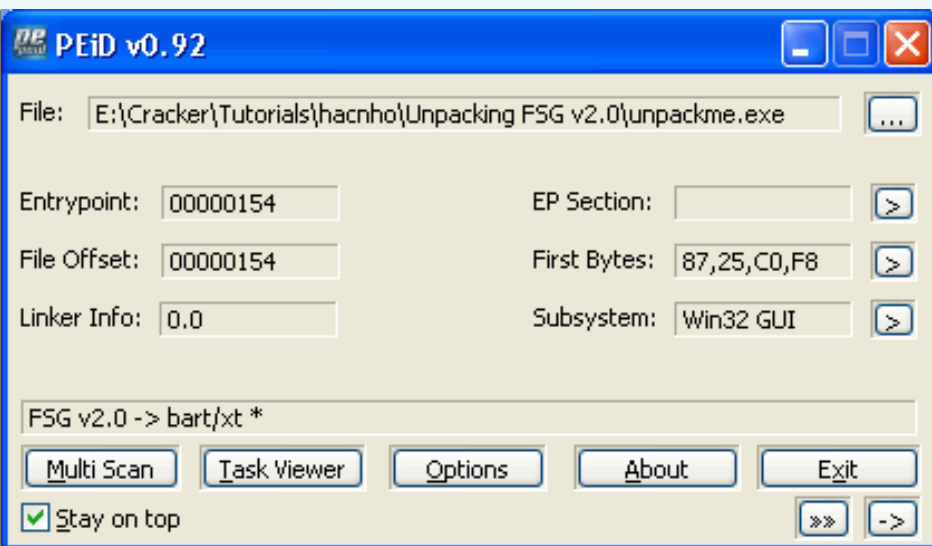
[FSG v2.0 -> bart / XT]

signature = 87 25? ? ? 00 61 94 55 A4 B6 80 FF 13

ep_only = true

;-----

or copy the files on your **userdb.txt** of PEiD folder and overwrite the old. (Included in zip file).



[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
	00001000	00014000	00000000	00000000	C00000E0
	00015000	00008000	00000200	0000A8FD	C00000E0

EP: 154, flags The value of this case is not needed, Image Base is always 400000, Import Table: 0000 and size is 00.

3. Finding the OEP

First, you may edit your OllyDBG Options:

Commands	Disasm	CPU	Registers	Stack	Analysis 1	Analysis 2	Analysis 3
Security	Debug	Events	Exceptions	Trace	SFX	Strings	Addresses

When main module is self-extractable:

☐ Extend code section to include extractor

☐ Stop at entry of self-extractor

☐ Trace real entry blockwise (inaccurate)

☒ Trace real entry bitwise (very slow!)

☒ Use real entry from previous run

☐ Pass exceptions to SFX extractor

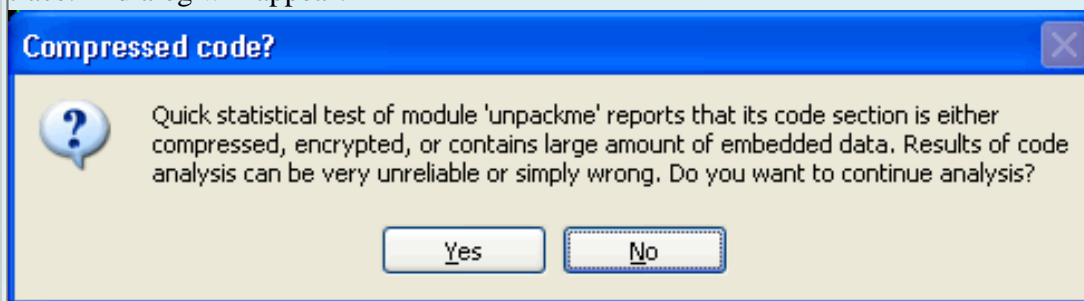
Load **unpackme.exe** into OllyDBG. And you still here:

00400154	8725 C0F84100	XCHG DWORD PTR DS:[41F8C0],ESP	
0040015A	61	POPAD	
0040015B	94	XCHG EAX,ESP	
0040015C	55	PUSH EBP	
0040015D	A4	MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	
0040015E	B6 80	MOV DH,80	
00400160	FF13	CALL DWORD PTR DS:[EBX]	
00400162	^ 73 F9	JNB SHORT unpackme.0040015D	
00400164	33C9	XOR ECX,ECX	
00400166	FF13	CALL DWORD PTR DS:[EBX]	
00400168	^ 73 16	JNB SHORT unpackme.00400180	
0040016A	33C0	XOR EAX,EAX	
0040016C	FF13	CALL DWORD PTR DS:[EBX]	
0040016E	^ 73 1F	JNB SHORT unpackme.0040018F	
00400170	B6 80	MOV DH,80	
00400172	41	INC ECX	
00400173	B0 10	MOV AL,10	
00400175	FF13	CALL DWORD PTR DS:[EBX]	
00400177	12C0	ADC AL,AL	
00400179	^ 73 FA	JNB SHORT unpackme.00400175	
0040017B	^ 75 3A	JNZ SHORT unpackme.004001B7	
0040017D	AA	STOS BYTE PTR ES:[EDI]	
0040017E	^ EB E0	JMP SHORT unpackme.00400160	
00400180	FF53 08	CALL DWORD PTR DS:[EBX+8]	
00400183	02F6	ADD DH,DH	

Now, attention! See Olly on status bar:

Tracing SFX: read=00403000

Wait Olly is tracing SFX. If your CPU speed is high, very fast Olly trace. (My CPU 2.8Ghz trace in 3S). When Olly finish trace. A dialog will appear!



Click **Yes** and we will still here:

Address	Disassembly	Comment
00401B76	55	PUSH EBP
00401B77	8BEC	MOV EBP,ESP
00401B79	6A FF	PUSH -1
00401B7B	68 88334000	PUSH unpackme.00403388
00401B7D	68 FC1C4000	PUSH unpackme.00401CFC
00401B7F	64:A1 000000	MOV EAX,DWORD PTR FS:[0]
00401B81	50	PUSH EAX
00401B83	64:8925 0000	MOV DWORD PTR FS:[0],ESP
00401B85	83EC 68	SUB ESP,68
00401B87	53	PUSH EBX
00401B89	56	PUSH ESI
00401B8B	57	PUSH EDI
00401B8D	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP
00401B8F	33DB	XOR EBX,EBX
00401B91	895D FC	MOV DWORD PTR SS:[EBP-4],EBX
00401B93	6A 02	PUSH 2
00401B95	FF15 B8314000	CALL DWORD PTR DS:[4031B8]
00401B97	59	POP ECX
00401B99	830D 40414000	OR DWORD PTR DS:[404140],FFFFFFFF
00401B9B	830D 44414000	OR DWORD PTR DS:[404144],FFFFFFFF
00401B9D	FF15 B4314000	CALL DWORD PTR DS:[4031B4]
00401B9F	8B0D 34414000	MOV ECX,DWORD PTR DS:[404134]
00401BA1	8908	MOV DWORD PTR DS:[EAX],ECX
00401BA3	FF15 B0314000	CALL DWORD PTR DS:[4031B0]
00401BA5	8B0D 30414000	MOV ECX,DWORD PTR DS:[404130]
00401BA7	8908	MOV DWORD PTR DS:[EAX],ECX

Real entry point of SFX code

Our OEP

MSVCRT.__set_app_type

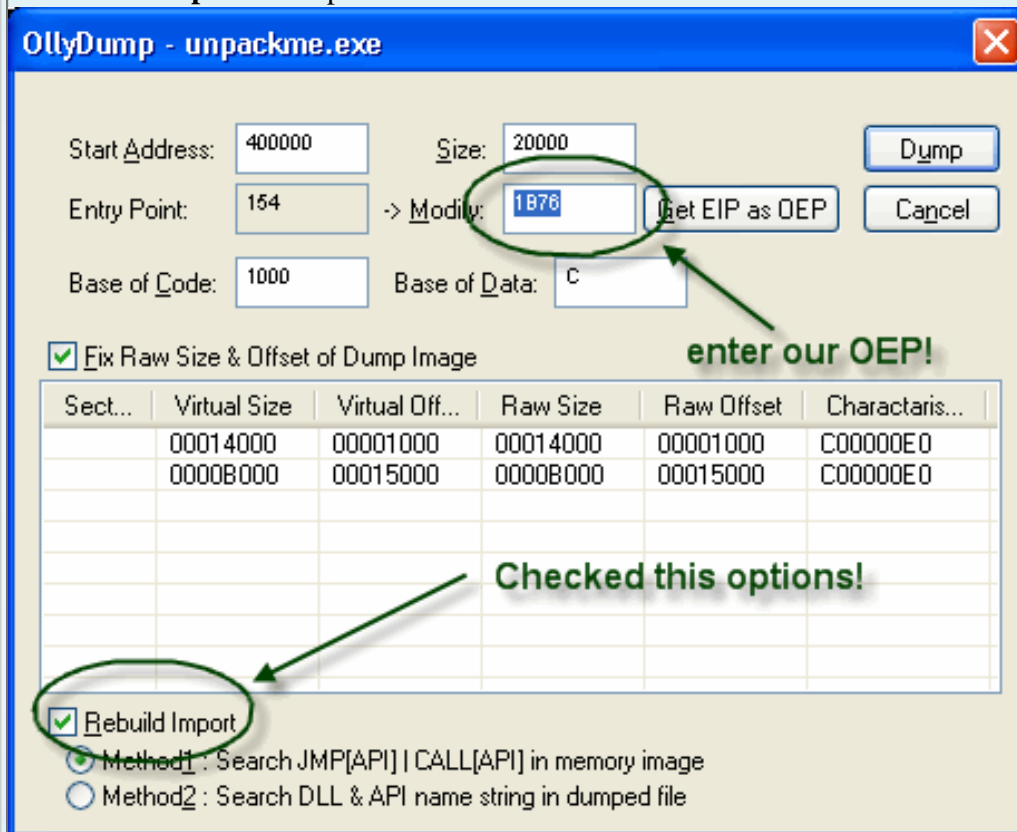
MSVCRT.__p__fmode

MSVCRT.__p__commode

Congratulations! According OEP we found is **401B76** And now we Calculate the real OEP of this unpackme by the formula:
 Real OEP = OEP Olly find in Image-Base-401B76 = 400000 = **1B76**.

4th dumping

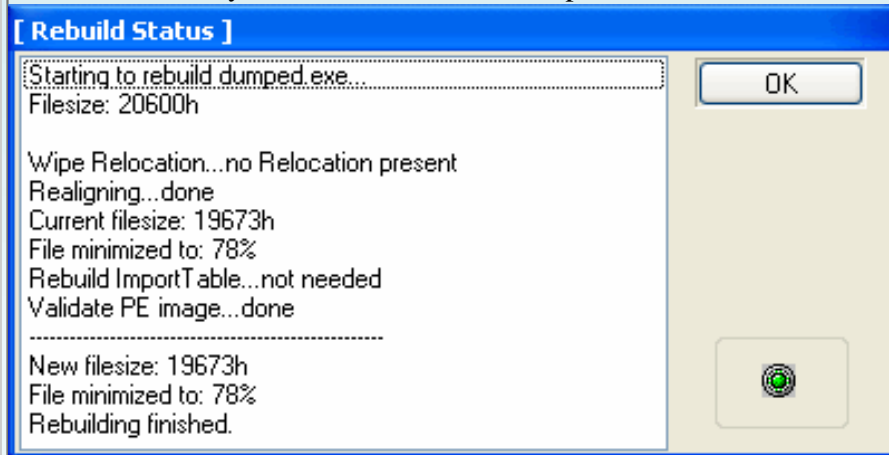
At address **00401B76** we go to the menu Plugin -> OllyDump -> dump debugged process. And then, just press dump, save the file at **dumped.exe** unpacked.



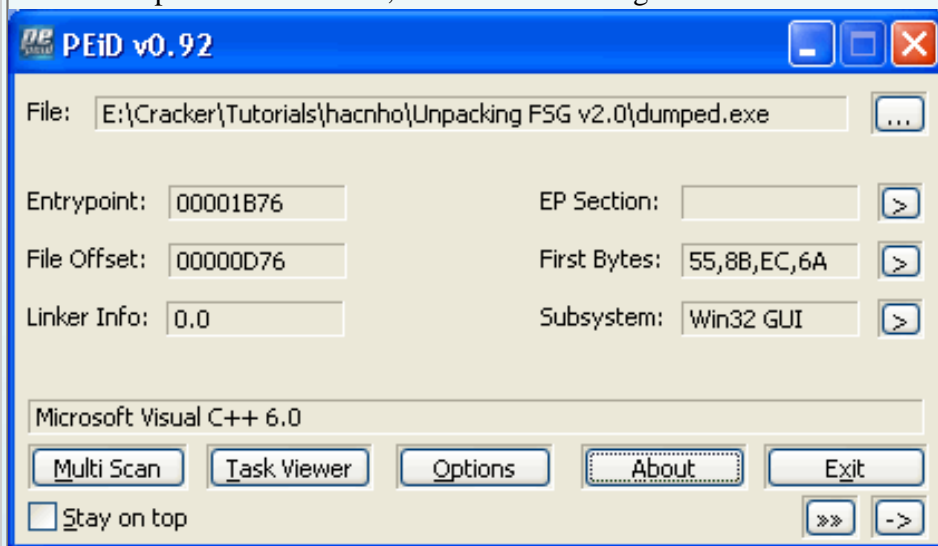
Because we fix IAT was on OllyDump. So, not needed to use ImportREC!

5. Rebuild PE

Use LordPE 1.4 by Y0da for rebuild our dumped.exe



Now run unpacked files. Wow, not crash. Detect again:



6. Conclusion

Special thanx to **R @ dier** for this template.

My Greetz to: Deux, infinite, NVH (c), softcracker_vn, luucorp, Aaron, hhphong, R @ dier, tlandn, Computer_Angel, Zombie, RCA, CTL, Moonbaby, Ferrari, Devilz, Neitsa, anh_surprise ... and you; -)!

To be continued ...

Written by [hacnho](#) (tutorial date: Sai Gon 25/5/2004)

Manual Unpacking of the ki himms italy - Packer 1st0



I -- Print d o t r u c t i o n :

Seeing the www.unpack.cn have introduced Packer 1 "is the tree garden" is their **hmimys-Packer 1.0**. Products brothers together MUP we try to see it do?

I I -- Tools T & g e r a t :

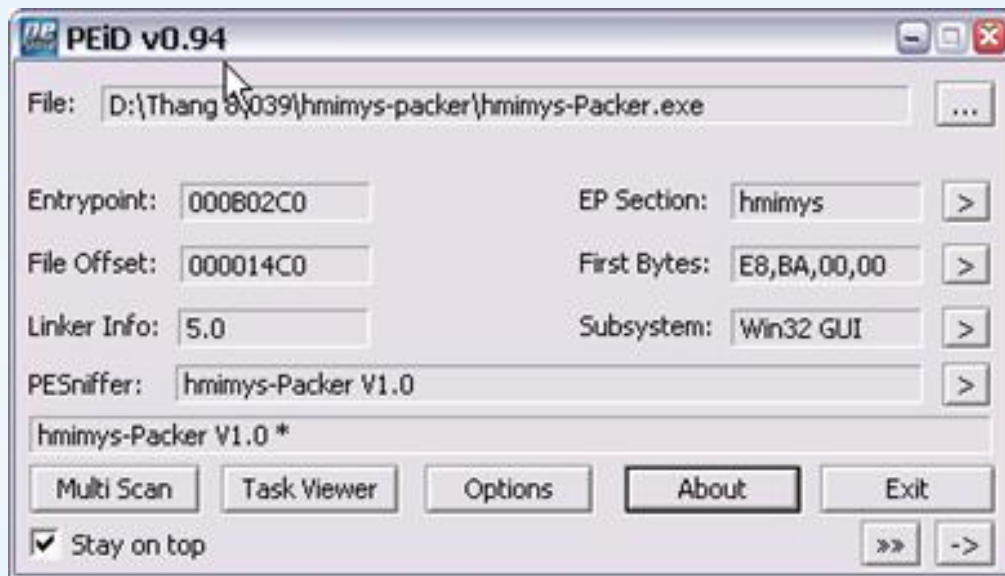
• T o o l P and the need to lug a d ứ ng:

- O L L Y D B G 1:10
- OllyDump 2.2 p l u g i n
- I m p O R T R E C 1.6f
- P E I D 0.94

• One g e r t : *hmimys - P e r 1.0 ACK*

I I I - F i n d O E P & D u m p F i l e :

_As usual for **PE** can **iD0.94** S t a g e r t



PEiD _ If you do identify the **hmimys-Packer 1.0** Add the **signature** to add **"userdb.txt"**

[S hmimy - ACK P e r V 1.0]

ignasture = 8 e BA00 00 00? 00 0 0 00

p_e = o nly will fal

_OK, Lo ad T a g e r tvao **OillyDBG**

Address	Hex dump	Disassembly	Comment
004B02C0	E8 BA000000	call hmimys-P.004B037F	
004B02C5	0300	add eax, dword ptr ds:[eax]	
004B02C7	0000	add byte ptr ds:[eax], al	
004B02C9	00E0	add al, ah	
004B02CB	0A00	or al, byte ptr ds:[eax]	
004B02CD	0010	add byte ptr ds:[eax], dl	
004B02CF	40	inc eax	
004B02D0	0069 96	add byte ptr ds:[ecx-6A], ch	
004B02D3	0300	add eax, dword ptr ds:[eax]	
004B02D5	42	inc edx	
004B02D6	0C 4B	or al, 4B	

_N han **F 7**

Address	Hex dump	Disassembly	Comment
004B037F	5E	pop esi	hmimys-P.004B02C5
004B0380	83C6 64	add esi,64	
004B0383	AD	lods dword ptr ds:[esi]	
004B0384	50	push eax	
004B0385	AD	lods dword ptr ds:[esi]	
004B0386	50	push eax	
004B0387	83EE 6C	sub esi,6C	
004B038A	AD	lods dword ptr ds:[esi]	
004B038B	50	push eax	
004B038C	AD	lods dword ptr ds:[esi]	
004B038D	50	push eax	

_ Mouse Scroll down to position the end of this Code and press F2 to Set 1 in which BP

Address	Hex dump	Disassembly	Comment
004B03C5	FF53 04	call near dword ptr ds:[ebx+4]	
004B03C8	AB	stos dword ptr es:[edi]	
004B03C9	EB E2	jmp short hmimys-P.004B03AD	
004B03CB	25 FFFFFFFF	and eax,7FFFFFFF	
004B03D0	50	push eax	
004B03D1	55	push ebp	
004B03D2	FF53 04	call near dword ptr ds:[ebx+4]	
004B03D5	AB	stos dword ptr es:[edi]	
004B03D6	EB DA	jmp short hmimys-P.004B03B2	
004B03D8	83EB 44	sub ebx,44	
004B03DB	8BF3	mov esi,ebx	
004B03DD	AD	lods dword ptr ds:[esi]	
004B03DE	50	push eax	
004B03DF	C3	ret	
004B03E0	55	push ebp	
004B03E1	8BFC	mov ebp,esp	

_ Press **F9** will stop at the point we've set BP

004B03DD	AD	lods dword ptr ds:[esi]	
004B03DE	50	push eax	
004B03DF	C3	ret	
004B03E0	55	push ebp	
004B03E1	8BEC	mov ebp, esp	
004B03E3	83EC 10	sub esp, 10	
004B03E6	8B4D 0C	mov ecx, dword ptr ds:[ebp+C]	
Return to 00401000 (hmimys-P.00401000)			

Address	Value	Address	Value	Comment
0048A000	56E02C	0012FFC0	00401000	hmimys-P.00401000
0048A004	200000	0012FFC4	7C816D4F	RETURN to kernel32.7C816D4F
0048A008	004057	0012FFC8	7C910738	ntdll.7C910738
0048A00C	B58C2C	0012FFCC	FFFFFFFF	
0048A010	1C0000	0012FFD0	7FFD4000	
0048A014	004892	0012FFD4	8054B038	
0048A018	E1302C	0012FFD8	0012FFC8	

Breakpoint at hmimys-P.004B03DF

_ Press the **F8** will be **OEP**

Address	Hex dump	Disassembly	Comment
00401000	EB 10	jmp short hmimys-P.00401012	==>OEP
00401002	66:623A	bound di, dword ptr ds:[edx]	
00401005	43	inc ebx	
00401006	2B2B	sub ebp, dword ptr ds:[ebx]	
00401008	48	dec eax	
00401009	4F	dec edi	
0040100A	4F	dec edi	
0040100B	4B	dec ebx	
0040100C	90	nop	
0040100D	E9 18A44800	jmp 0088B42A	
00401012	A1 0BA44800	mov eax, dword ptr ds:[48A40B]	
00401017	C1E0 02	shl eax, 2	
0040101A	A3 0FA44800	mov dword ptr ds:[48A40F], eax	
0040101F	52	push edx	
00401020	6A 00	push 0	
00401022	E8 0B840800	call hmimys-P.00489432	jmp to kernel32.GetModuleHandleA
00401027	8BD0	mov edx, eax	

_ Bay Time we need to determine the position start and end of the **IAT**.

Address	Value	Comment
004990E0	00099E14	
004990E4	00099E24	
004990E8	00000000	
004990EC	77DD6BF0	ADVAPI32. RegCloseKey
004990F0	77DD761B	ADVAPI32. RegOpenKeyExA
004990F4	77DD7883	ADVAPI32. RegQueryValueExA
004990F8	00000000	
004990FC	00099E38	
00499100	00099E46	
00499104	00099E58	
00499108	00099E64	
0049910C	00099E74	
00499110	00099E82	

IAT Start

Scroll down to find **_ IAT End**

Address	Value	Comment
00499D60	77132122	OLEAUT32. VarR8FromStr
00499D64	771265C4	OLEAUT32. VariantChangeTypeEx
00499D68	771248C0	OLEAUT32. VariantClear
00499D6C	7714D295	OLEAUT32. VariantCopy
00499D70	7714D348	OLEAUT32. VariantCopyInd
00499D74	77124920	OLEAUT32. VariantInit
00499D78	00000000	
00499D7C	0009B7DE	
00499D80	00000000	
00499D84	76381180	
00499D88	00000000	
00499D8C	41564441	
00499D90	32334950	

IAT End

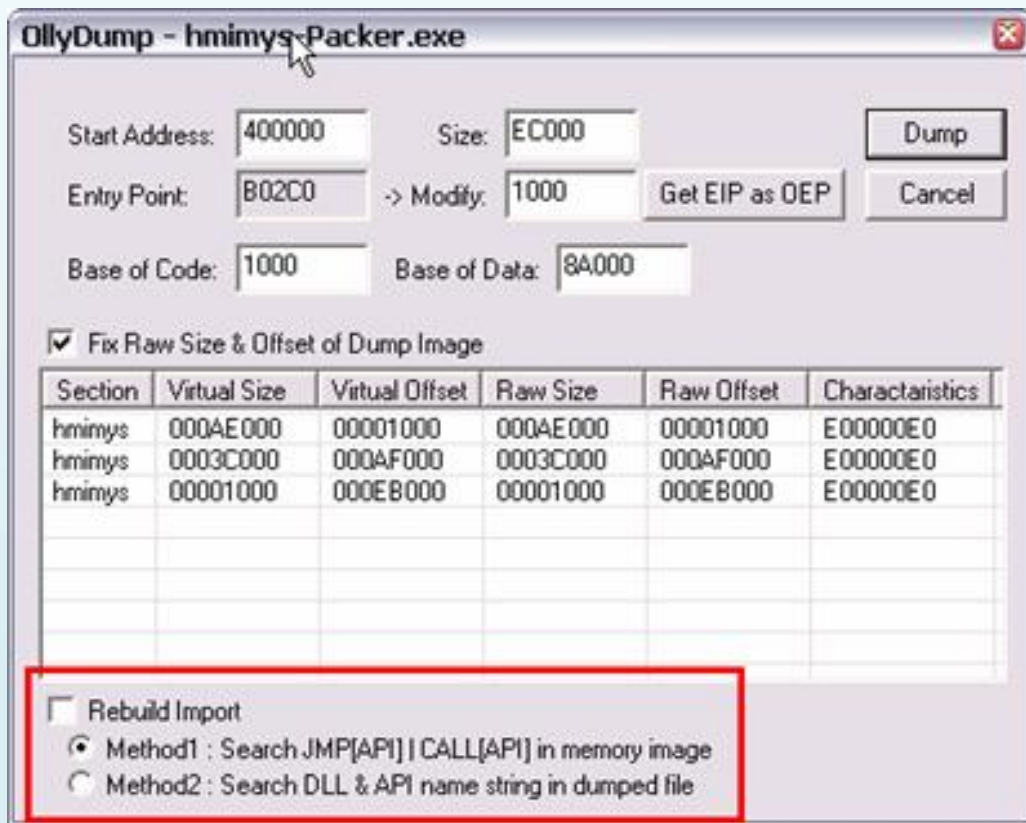
_T Ú m back of a C n h to the usa:

Dr. IAt r a t: 00 4 9 9 0 EC 77 DD 6 B 0 A D F I VAP 3 2. Re g C
o l e s e K italy

IAT E nd: 0049 9 D 7 4 7 7 1 2 49 20 O L E A U - T 32. And a n t r i n i t
I

IAT and L e h t: 00 0 0 0 C 8 8

Good q _ not called, D u m p i try ... with a S T E N any k y h n n g n as they
choose long n g h is h h ì n h a n below.



Th _ see a P r e c k to this for a q u a o n g h could write to the cells of the Asia S i r c i t p

```

/ / = ===== = = = = = = = = = =====
// F i l e m e      : h m i m i t a l y s - P r d e v i l i s h 1 s t E P
   N a              0 0 F i n d S c r i p t
// A u t h o r      : I t a l y N W h t o B a r
// W e b S i t e    : h _ t t _ p : / / _ r e _ o n a l i n e . n e t
   e
/ / ===== = = = = = = = = = =====

S T I / F 7
F D O I N P e i p , 5 0 # 3 # / / S e r o f a " 5 0 C 3 "
g o $ U R E S T L
s t o / F 8
s t o / F 8

c e t i m p , " O P E ! F o u n d b i t a l y y N W h o

```


t r a B!" M S G " P l e m p A S E d u d e d i n o u r

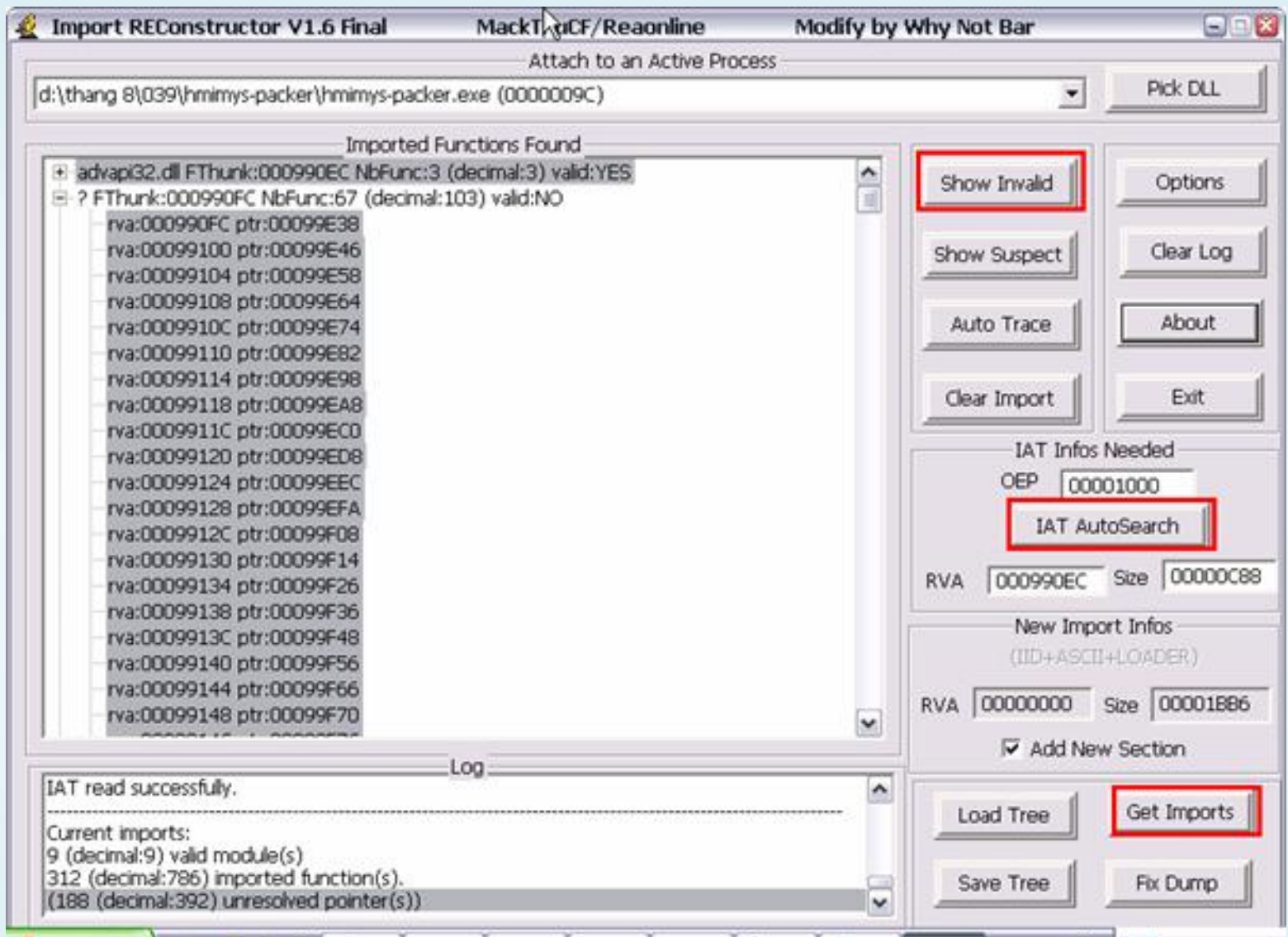
x i F i A T"

R e t

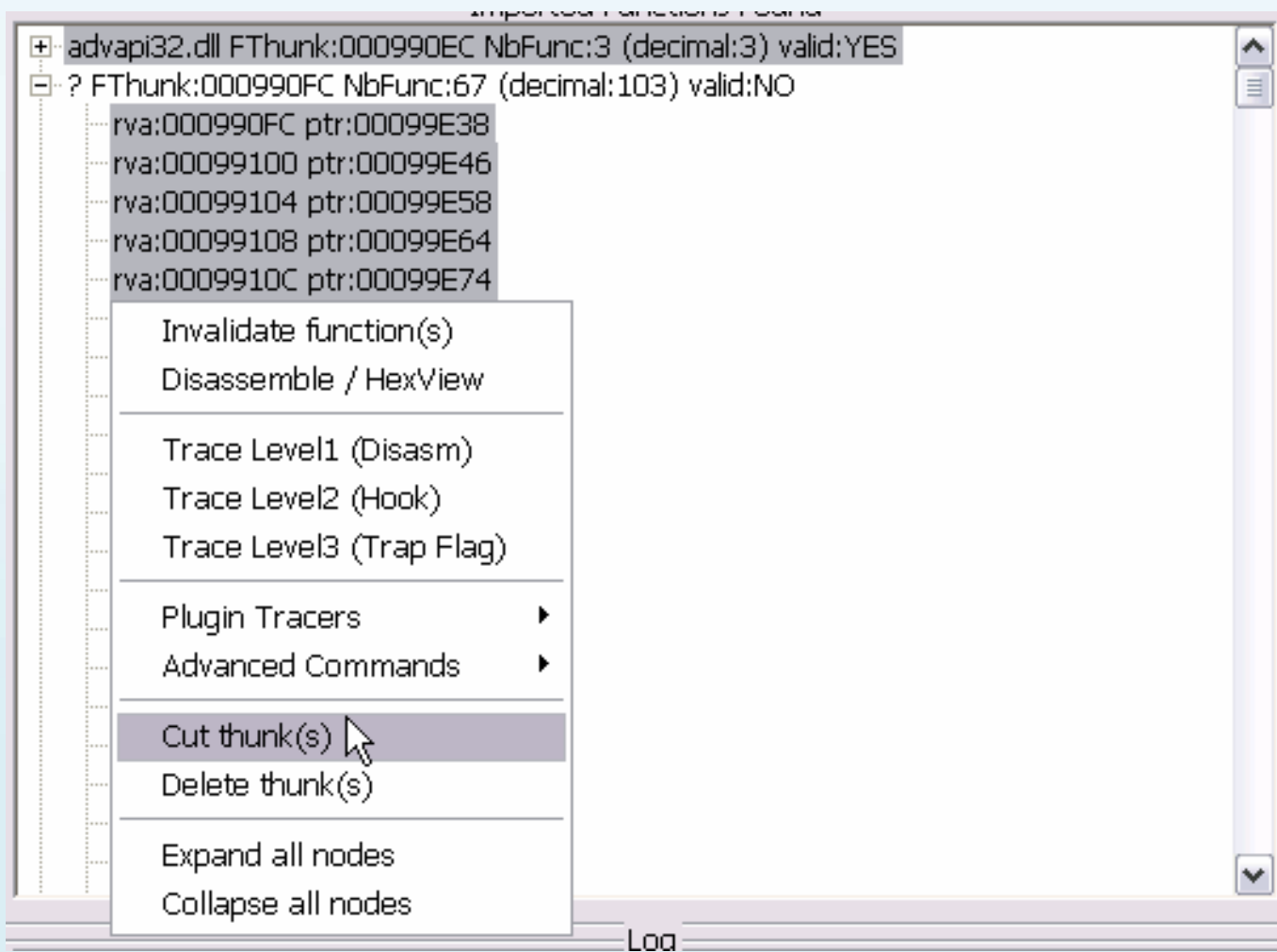
I-R E V I build mp ort:

_ Open **ImportREC** up. Select list **in-process hmimys Packer.exe**. Enter **O**
P E = 0 0 40 1 000-004000 0 0 (I m a g e s a b e) = 0 0 001000
R V = A 0 0 4 9 9 0 E-C 0 0 400,000 (I m a g e s a b e) = E 0 0
0990 C Si z e = 0000 0 CC 8 (I A T and L e t h)

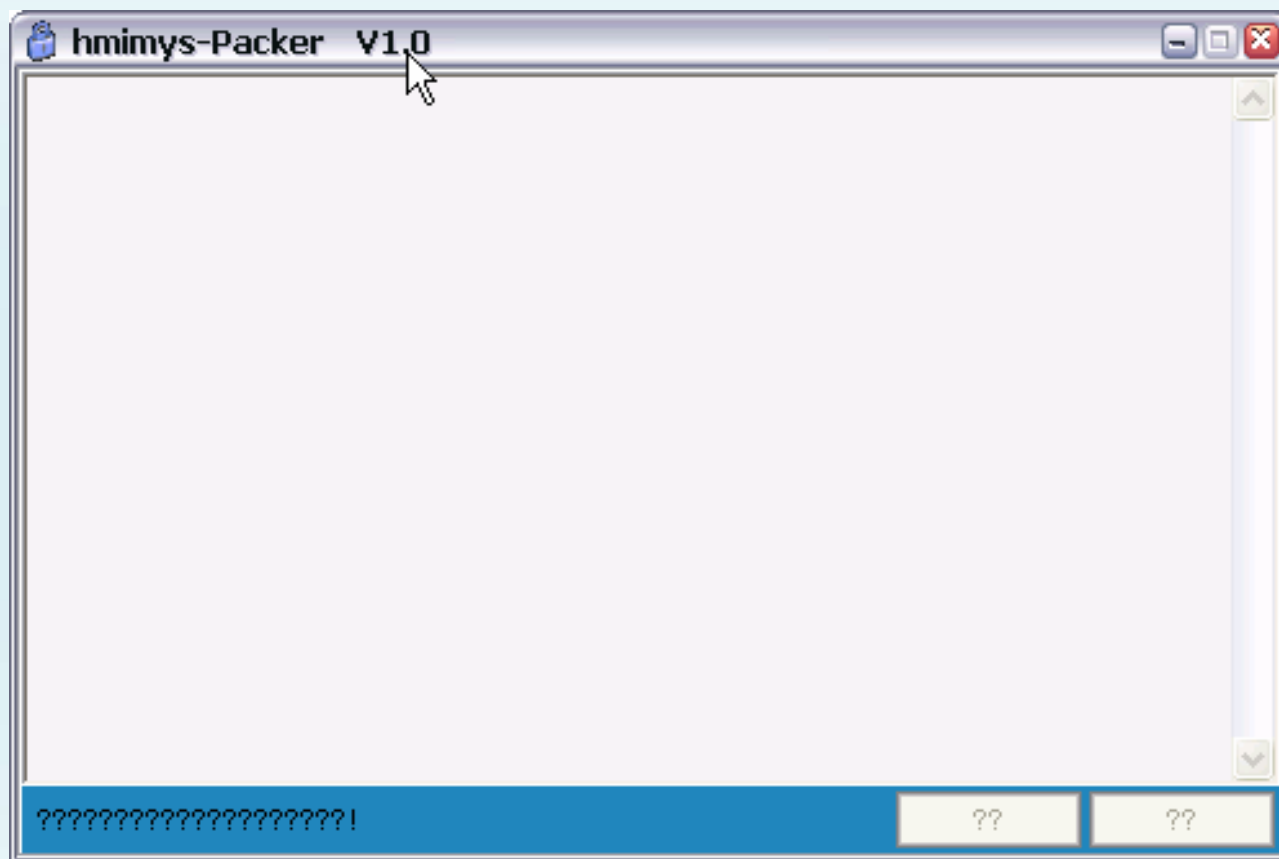
Click *I A T to the A S E A R and-> G e t I m p t o r s-> S h o w I n v a l i d*



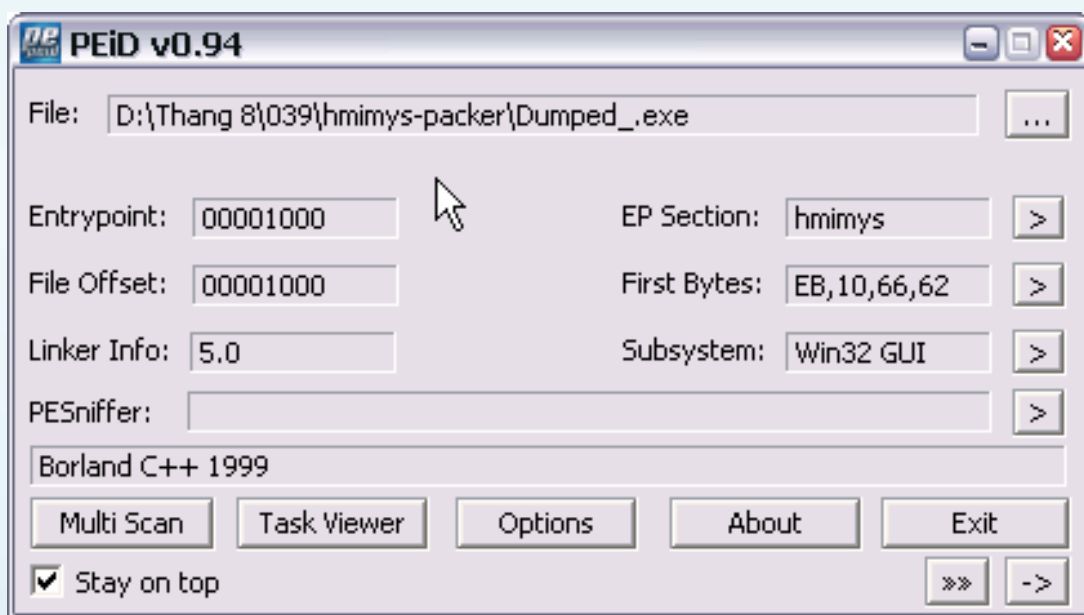
_Hichic A lot function **Invalid Thunks** ... **Cut** out the window and select **ImportREC** nhusau:



_ Show Invalid Click again and do not function any longer **Invalid**, Click **Fix Dumped.exe dump** file and select Run Test File **Dumped_.exe**



_Hehehe.... **Unp a ck D o n e!!**



**G r I Ee TsF italy Ou the Compu t e r A _ of e l, e mbi Z o, M A B oo nb
italy, H o acnh, Nina B e, k i nman o w e r a, Z o i, D e ux, M e r c, the
light o f nix, T r o ickyb italy, Takad a iamidi ot, of the e n t e n handi ...
and italy o u!**

The N h a n a g, 3 1 t h a n G8 year 200 6

Whot italyN Bar

::: [MUP ID Application Protector 1.2]::.

([_kienmanowar_](#))



I. Foreword

Today, I downloaded UnpackMe (PE32bit): **ID Application Protector 1.2** from Teddy's site and try to manual unpacking it. 0k13! L3t's R0ck w1th m3 J

II. Target and Tools

Target:

Name: **UnPackMe_ID Application Protector 1.2.e.exe**

Home site: <http://www.tuts4you.com/>

Tools:

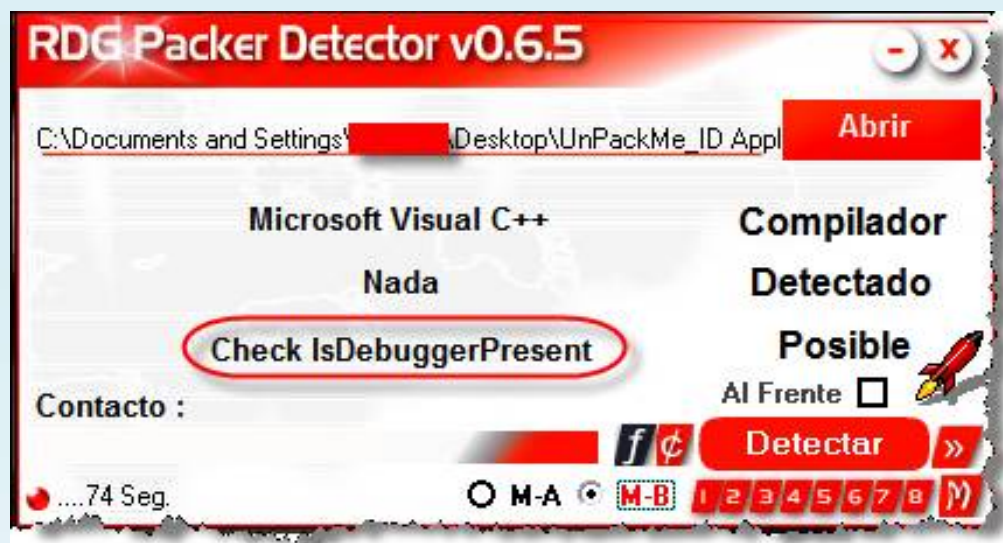
Debugger: Ollylce (16/2/07)

PE Tools: RDG Packer Detector, PEiD, CFF Explorer, LordPE, ImpRec

III. Manual Unpacking

_First, Dectectors we use to find information about the target we'll work with:

_ **RDG Packer Detector:**

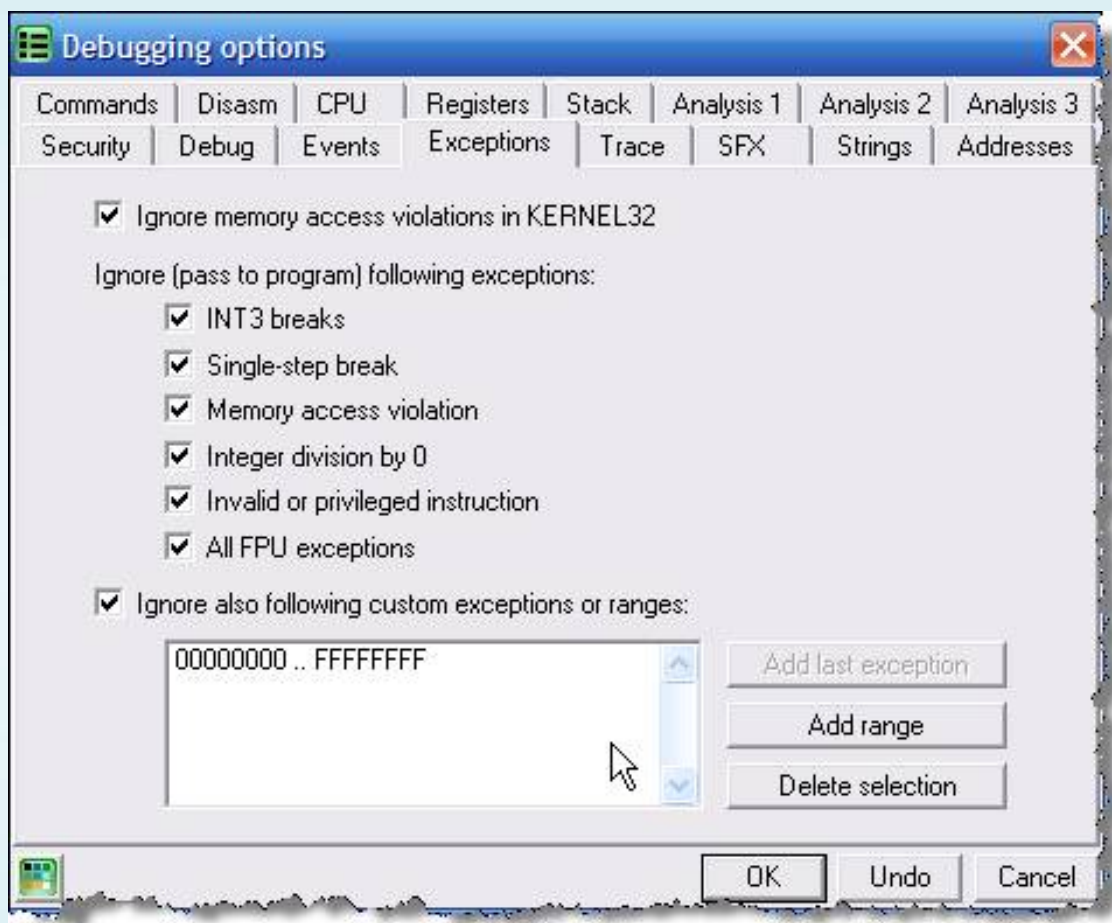


_ **PeiD:**



RDG packer can not detect this but it gives me some useful information about target. My PeiD gives me exactly the result because it uses the signatures of J fly.

_Ok That's enough, now open Olly and configure it like that:



_This Uses the target *to IsDebuggerPresent function determines whether the calling process is running under the context of a debugger.* We can bypass this function manually or use the **Hide Debugger** plugin to defeat it.



Olly _After configured like above picture, we will load into Olly target, after loading successful we stop here in Olly:

Address	Hex dump	Disassembly	Comment
0046B060	60	pushad	
0046B061	E8 00000000	call 0046B066	
0046B066	5D	pop ebp	
0046B067	81ED F20B4700	sub ebp, 00470BF2	
0046B06D	B9 19224700	mov ecx, 00472219	
0046B070	81ED F20B4700	sub ebp, 00470BF2	

_Aha, **Pushad and Popad method.** May be we do it like MUP MUP UPX method, it was first that I tried but I can not go to the OEP of this target. Ok, we try another method. Press **Alt + M** to open the Memory window:

003F0000	00002000				Map	R	R
00400000	00001000	UnPackMe	PE header	Imag	R	RWE	
00401000	0004A000	UnPackMe	.text code	Imag	R	RWE	
0044B000	0000C000	UnPackMe	.rdata	Imag	R	RWE	
00457000	00009000	UnPackMe	.data data	Imag	R	RWE	
00460000	00003000	UnPackMe	.idata	Imag	R	RWE	
00463000	00008000	UnPackMe	.rsrc resources	Imag	R	RWE	
0046B000	0002F000	UnPackMe	.Prt SFX, imports	Imag	R	RWE	
004A0000	00008000			Map	R E	R E	

_Next, **Select. Text** section and set a BP like the following picture:

00400000	00001000	UnPackMe	PE header	Imag	R	RWE	
00401000	0004A000	UnPackMe	.text code	Imag	R	RWE	
0044B000	0000C000	UnPackMe	.rdata	Imag	R	RWE	
00457000	00009000	UnPackMe	.data dat	Imag	R	RWE	
00460000	00003000	UnPackMe	.idata	Imag	R	RWE	
00463000	00008000	UnPackMe	.rsrc res	Imag	R	RWE	
0046B000	0002F000	UnPackMe	.Prt SFX	Imag	R	RWE	
004A0000	00008000			Map	R E	R E	
00560000	00002000						
00570000	00103000						
00580000	00005000						

Actualize

View in Disassembler Enter

Dump in CPU

Dump

Search Ctrl+B

Set break-on-access F2

Set memory breakpoint on access

_Press **Shift + F9** to run, we'll break in here Olly:

Address	Hex dump	Disassembly	Comment
0046BF13	3E:AC	lods byte ptr [esi]	
0046BF15	26:AA	stos byte ptr es:[edi]	
0046BF17	E2 FA	loopd short 0046BF13	
0046BF19	5F	pop edi	
0046BF1A	5E	pop esi	
0046BF1B	58	pop eax	
0046BF1C	59	pop ecx	
0046BF1D	89EC	mov esp, ebp	
0046BF1F	5D	pop ebp	
0046BF20	C3	ret	

BP _Remove memory and see a little code below, you'll find at **RETN 0x0046BF20**, select this command and press **F2** to set a BP. Next, we'll break press **F9**:

0046BF1D	89EC	mov esp, ebp
0046BF1F	5D	pop ebp
0046BF20	C3	ret
0046BF21	55	push ebp

_Remove This BP, and then press **Alt + M**, set a BP as we did at **above. Text** section. After that, press **Shift + F9** to run our target. Olly'll break here:

Address	Hex dump	Disassembly	Comment
0046BCBB	3E:8917	mov dword ptr [edi], edx	
0046BCBE	83C7 04	add edi, 4	
0046BCC1	49	dec ecx	
0046BCC2	75 F1	jnz short 0046BCB5	
0046BCC4	29C6	sub esi, eax	
0046BCC6	29C7	sub edi, eax	
0046BCC8	3E:8A06	mov al, byte ptr [esi]	

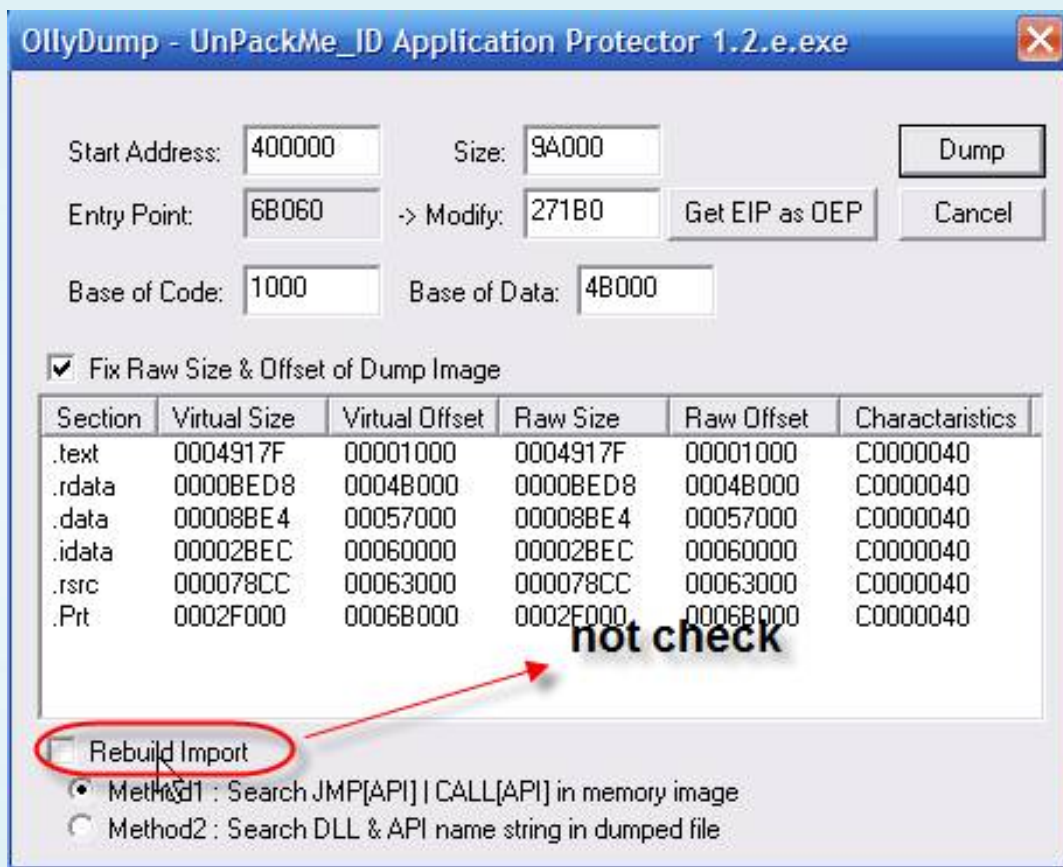
_OK, Do the same, remove memory breakpoint and scroll down to find the **RETN** command. I found it as the following picture:

Address	Hex dump	Disassembly	Comment
0046BECB	F7D8	neg eax	
0046BECF	83C4 0C	add esp, 0C	
0046BED2	5A	pop edx	
0046BED3	59	pop ecx	
0046BED4	5B	pop ebx	
0046BED5	5E	pop esi	
0046BED6	5F	pop edi	
0046BED7	5D	pop ebp	
0046BED8	C2 0800	retn 8	
0046BEDB	B8 01000000	mov eax, 1	

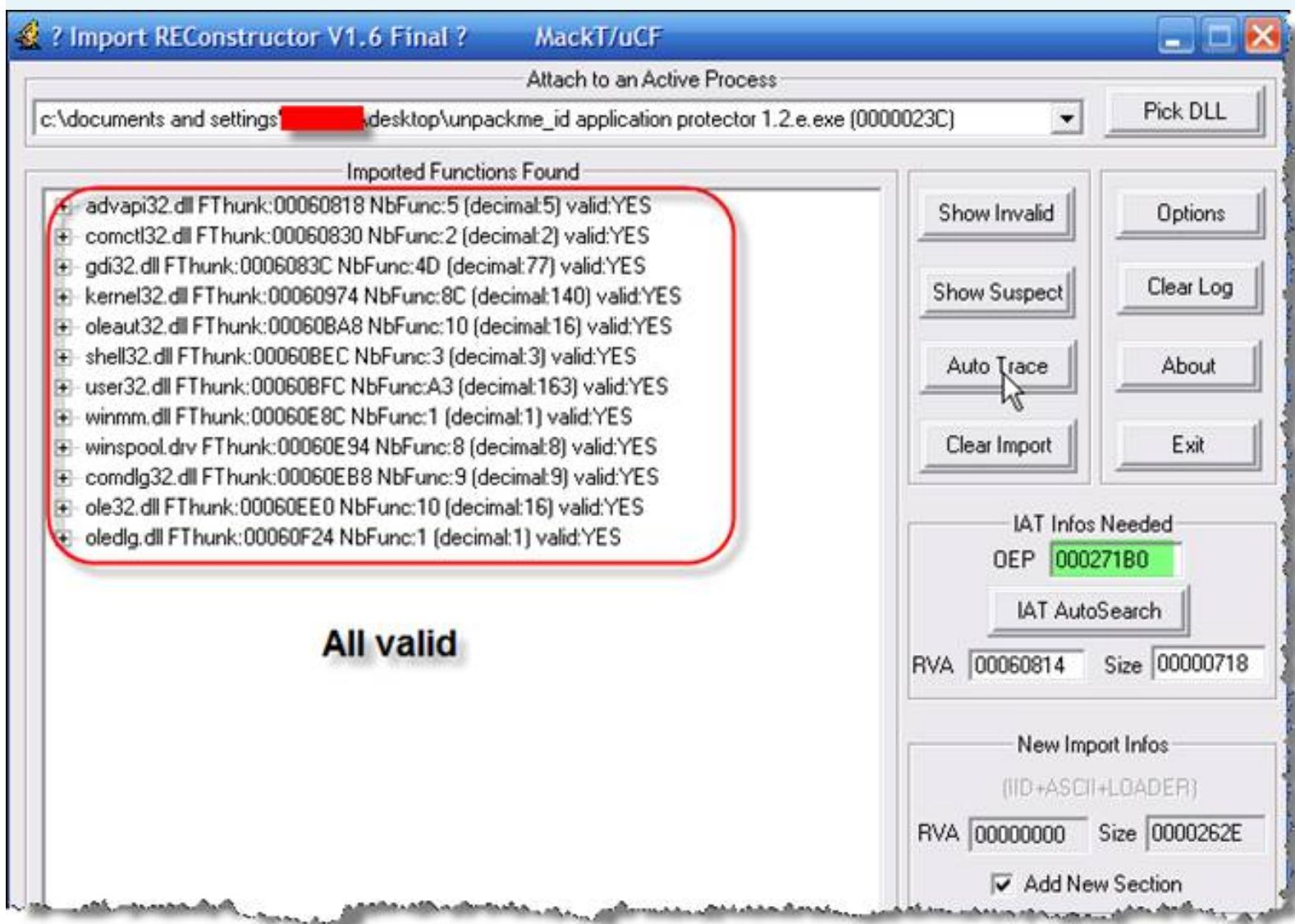
_Nothing Specially, press **F2** to place at a BP **RETN 8** command and then press **F9** to run. Olly'll break at the BP that we set. This clear and BP Press **Alt + M** to open the memory window to set the BP as we did above. After that, press **Shift + F9** to run, kaka Olly OEP at the break of the target J .

Address	Hex dump	Disassembly	Comment
004271B0	55	push ebp	OEP
004271B1	8BEC	mov ebp, esp	
004271B3	6A FF	push -1	
004271B5	68 600E4500	push 00450E60	
004271BA	68 C8924200	push 004292C8	
004271BF	64:A1 00000000	mov eax, dword ptr fs:[0]	

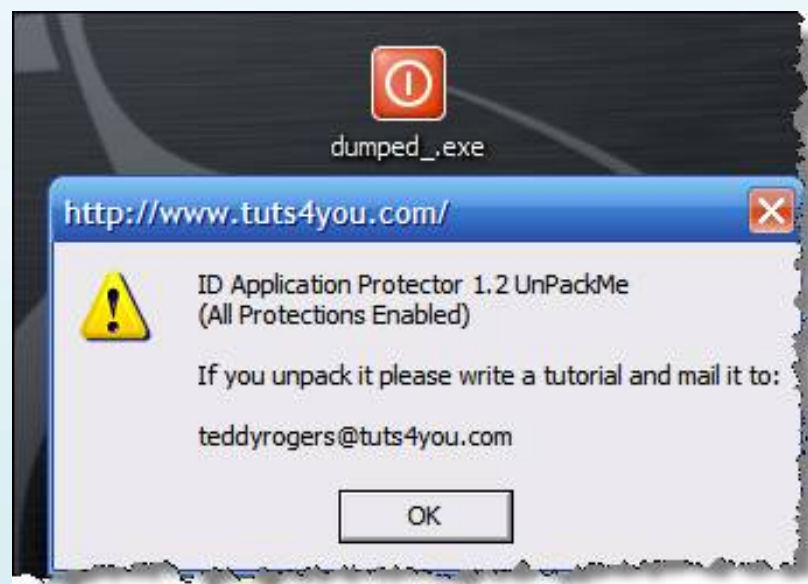
_Let's Dump and Fix IAT. Use **OllyDump** to dump the target and save the file dumped as any name you like:



_Open ImpRec and chooses process, fill the right OEP. **IAT AutoSearch** Press and then press **Get Imports**.

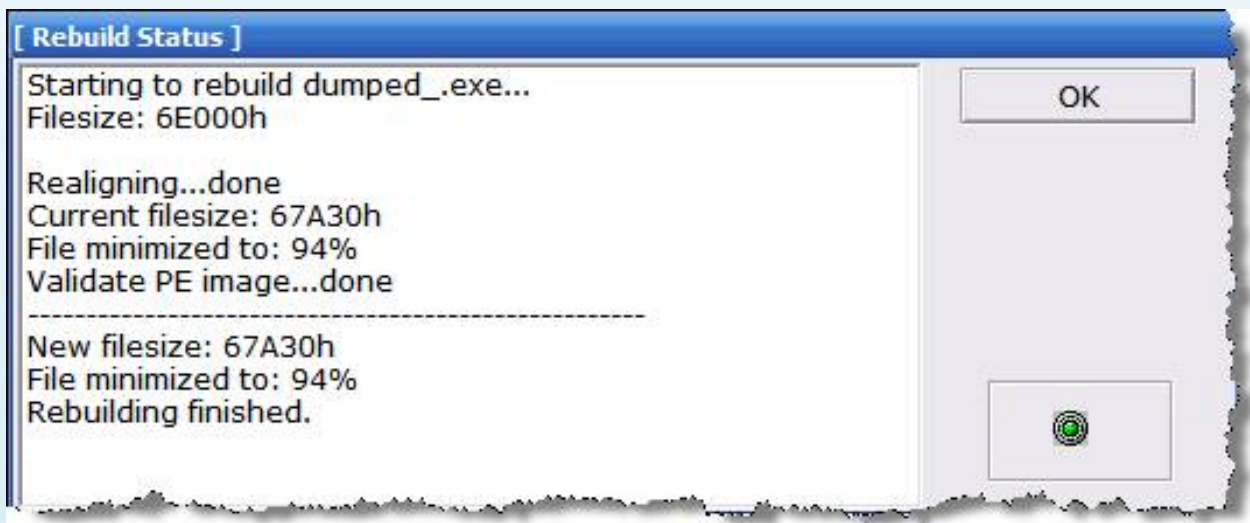
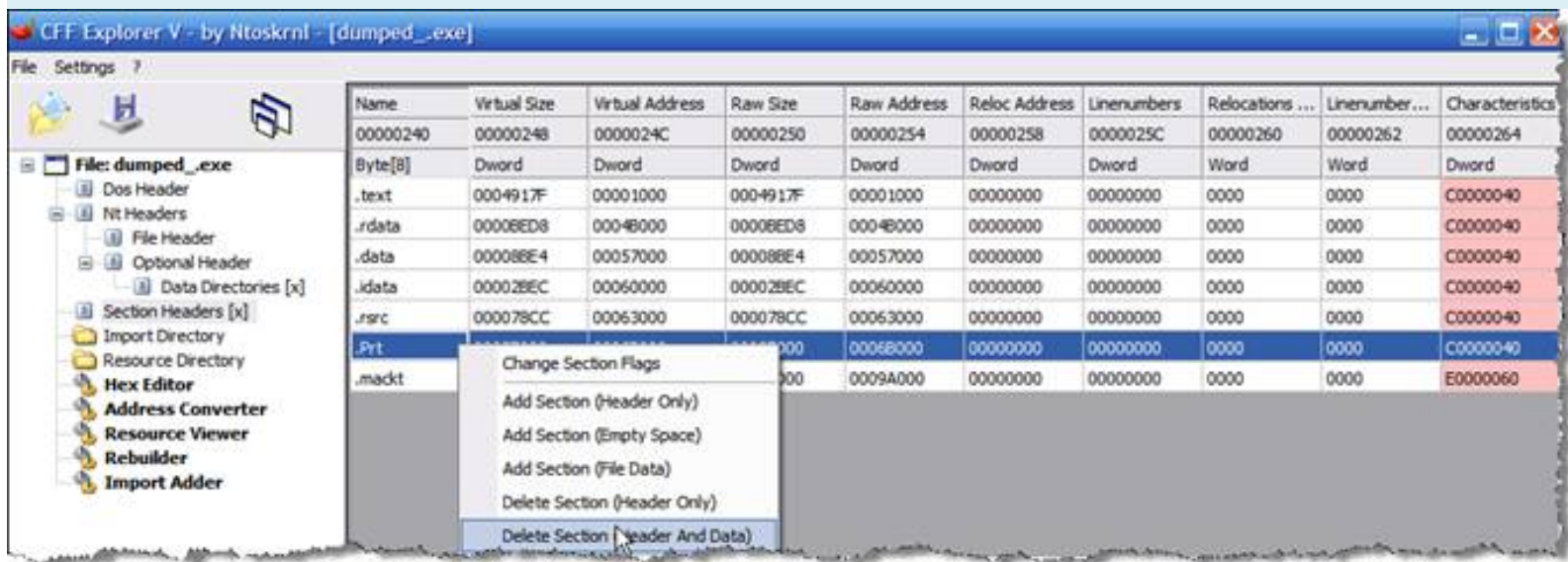


_Finally Press **Fix dump** dumped and choose the file to fix IAT. Run our fixed file to test, it runs ok normally J



_Open CFF Explorer and load the file into fixed it. Delete the section of the packer and use the

LordPE fixed to rebuild the file.



_The End. I hope my poor English with all of you can understand what I write. See you in another tutorials.

Best Regards

_ [Kienmanowar] _



--+ +---==[Greatz thanks to]=---+ +--

My family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHOCrker,

the_Lighthouse, Merc, Hoadongnoi, Nini ... all REA's members, TQN, HacNho, RongChauA, Deux, tlandn, light.phoenix, dqtlN, ARTEAM all my friend, and you.

Thanks to --+ +--==[]==--+ +--

iamidiot, WhyNotBar, trickyboy, dzungltvn, takada, hurt_heart, haule_nth, v. hytkl. v.. You have contributed greatly to the REA. Hope you will continue to promote J

I want to thank **Teddy Roggers** for his great site, Reversing.be folks (especially **haggar**), Arteam folks (**Shub-Nigurrath**, **MaDMan_H3rCuL3s**) and all folks on crackmes.de, thank to all members of **unpack.cn** (especially **fly** and **linhanshi**). Great thanks to **lena151** (I like your tutorials). And finally, thanks to **Ricardo NARVAJA** and all members on **CRACKSLATINOS**.

>>> If you have any suggestions, comments or corrections email me: **kienmanowar [at] reaonline.net**

hacnho Tutorials # 11

Manual unpacking Mew 10 exe-coder 1.0 -> Northfox [HCC]

Information	Unpacking for Newbie's
Target	unpackme.exe
Available	http://nhandan.info/hacnho/tuts/unpackme11_tuts.zip
Tools	OllyDbg plugin 1.10c with OllyDump 2.21.108, ImportREC Final 1.6, 1.4 LordPE, PEiD 0.92.
Protection	Exe mew 10-coder 1.0 -> Northfox [HCC]
L Evel	Standard
Category	Manual unpacking

1. Introduction

Mew 10 v1.0 is a powerful packer for compress your soft. Some case, it's good pack UPX or FSG. See bellow:

exe name	unpacked	FSG 1.33	UPX 1.22	Mew 9 beta 3	Mew 10 v1.0
test.exe	1024	can't pack	can't pack	717	542
server.exe	45056	18576	19456	can't pack	18518
opengl.exe	6144	2784	4096	2929	2711
mew5.exe	45056	18928	19968	19096	18880
main.exe	90624	38576	39424	38723	38477
keygen.exe	7680	4352	5632	4479	4272
imtoo.exe	2560	1088	can't pack	1234	1039
crack.exe	5120	2976	can't pack	3097	2894
bytekiller.exe	61440	22640	23040	22556	22321

He he, this is good for packer optimize size for your software, but if you do not use it for anti-cracker. Why? Because, very easy for unpack this packer. This is the newest packer was found on k3nny's website (hxxp: // k3nny.wz.cz). Okay, I will explain the ways for manual unpacking for you and anyone like unpacking tech; --).

2. Getting Started

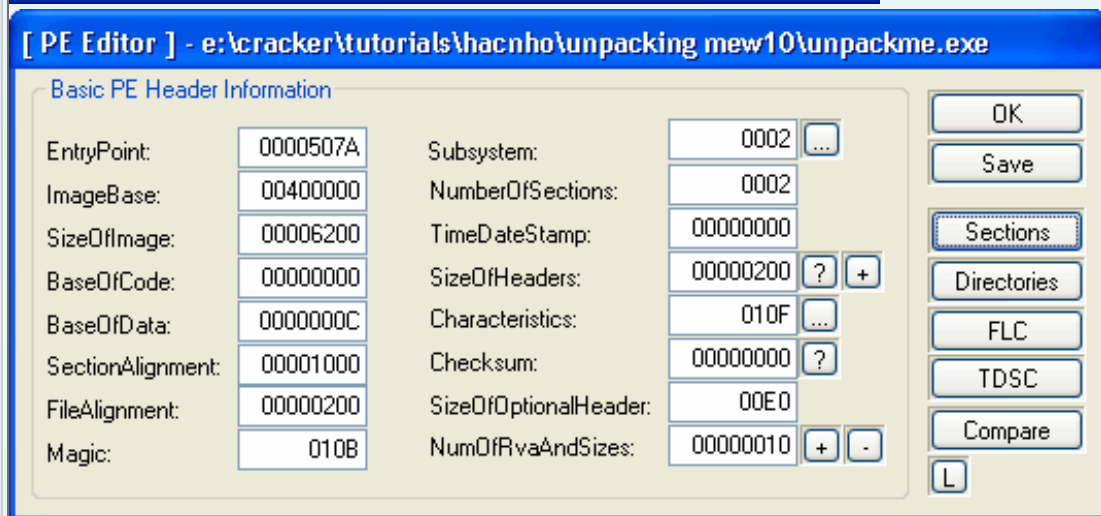
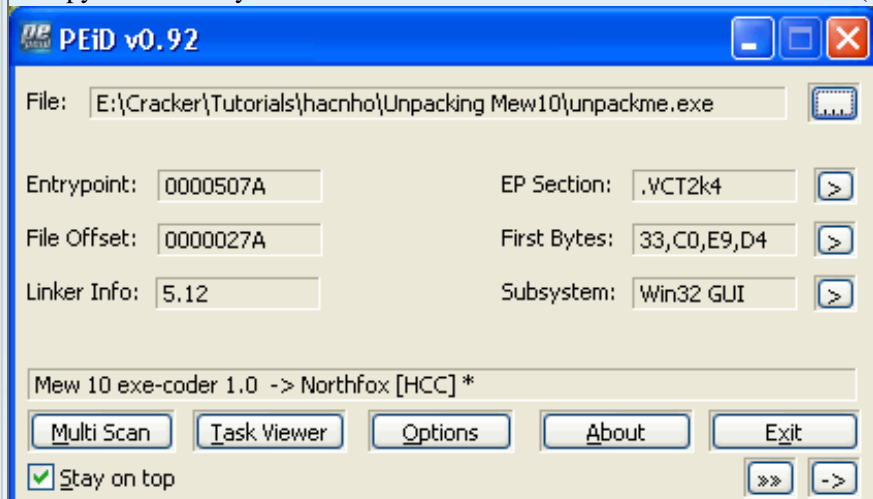
Use PEiD and get some LordPE for PE Info. If your PEiD can not detect my unpackme. You can add this to your sign **userdb.txt**

[Mew 10 exe-coder 1.0 -> Northfox [HCC]]

signature = 33 C0 E9 ? ? FF FF 6A ? ? ? ? 70

ep_only = true

or copy the files on your **userdb.txt** of PEiD folder and overwrite the old. (Included in zip file).



[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
.hacnho	00001000	00004000	00000000	00000000	E00000C0
.VCT2k4	00005000	00001200	00000200	00000FC3	E0000020

EP: 507A, flags The value of this case is not needed, Image Base is always 400000, Import Table: 0000 and size is 00.

3. Finding the OEP

Load **unpackme.exe** into OllyDBG. And you still here:

```

0040507A 33C0 XOR EAX,EAX
0040507C E9 04B0FFFF JMP keygen.00400155
00405081 6A 38 PUSH 38
00405083 40 INC EAX
00405084 68 3A3070EA PUSH EA70303A
00405089 33DF XOR EBX,EDI
0040508B 0B38 OR EDI,DWORD PTR DS:[EAX]
0040508D 0E PUSH CS
0040508E E8 7D021407 CALL 07545310
00405093 214C1E 01 AND DWORD PTR DS:[ESI+EBX+1],ECX
00405097 73 17 JNB SHORT keygen.004050B0
00405099 50 PUSH EAX
0040509A E2 68 LOOPD SHORT keygen.00405104
0040509C 0E PUSH CS
0040509D 184D F5 SBB BYTE PTR SS:[EBP-B],CL
004050A0 3188 04328645 XOR DWORD PTR DS:[EAX+45863204],ECX

```

Then, you press **F7** two times and you see as follows:

```

00400155 ? BE 61504000 MOV ESI,keygen.00405061 Real entry point of SFX code
0040015A AC LODS BYTE PTR DS:[ESI]
0040015B 91 XCHG EAX,ECX
0040015C AD LODS DWORD PTR DS:[ESI]
0040015D 95 XCHG EAX,EBP
0040015E AD LODS DWORD PTR DS:[ESI]
0040015F 92 XCHG EAX,EDX
00400160 AD LODS DWORD PTR DS:[ESI]
00400161 51 PUSH ECX
00400162 56 PUSH ESI
00400163 87F2 XCHG EDX,ESI
00400165 97 XCHG EAX,EDI
00400166 FC CLD

```

Now, press **F7** to trace to address **400,162**. At this line, you see in the **Registers (FPU)** table. The value of **ESP** is **0012FFC0**, then you right click on and choose **ESP Follow in the dump**.

```

00400155 ? BE 61504000 MOV ESI,keygen.00405061 Real entry point of SFX code
0040015A AC LODS BYTE PTR DS:[ESI]
0040015B 91 XCHG EAX,ECX
0040015C AD LODS DWORD PTR DS:[ESI]
0040015D 95 XCHG EAX,EBP
0040015E AD LODS DWORD PTR DS:[ESI]
0040015F 92 XCHG EAX,EDX
00400160 AD LODS DWORD PTR DS:[ESI]
00400161 51 PUSH ECX
00400162 56 PUSH ESI
00400163 87F2 XCHG EDX,ESI
00400165 97 XCHG EAX,EDI
00400166 FC CLD
00400167 B2 80 MOV DL,80
00400169 33DB XOR EBX,EBX
0040016B A4 MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
0040016C B3 02 MOV BL,2
0040016E FF55 04 CALL DWORD PTR SS:[EBP+4]
00400171 73 F8 JNB SHORT keygen.0040016B
00400173 33C9 XOR ECX,ECX
00400175 FF55 04 CALL DWORD PTR SS:[EBP+4]
00400178 73 18 JNB SHORT keygen.00400192
0040017A 33C0 XOR EAX,EAX
0040017C FF55 04 CALL DWORD PTR SS:[EBP+4]
0040017F 73 1F JNB SHORT keygen.004001A0
00400181 B3 02 MOV BL,2
00400183 41 INC ECX
00400184 B0 10 MOV AL,10
00400186 FF55 04 CALL DWORD PTR SS:[EBP+4]
00400189 12C0 ADC AL,AL
0040018B 73 F8 JNB SHORT keygen.0040019C

```

Registers (FPU)	
EAX	00401000 keys
ECX	00000003 keys
EDX	00405081 keys
EBX	7FFDF000 keys
ESP	0012FFC0 keys
ESI	00405040 keys
EDI	0000800E keys
EIP	00400162 keys
ES	0023 32bit
CS	001B 32bit
SS	0023 32bit
DS	0023 32bit
FS	0038 32bit
GS	0000 NULL
LastErr ERROR	
00000246 (NO, ...)	
empty	-UNORM
empty	+UNORM
empty	+UNORM
empty	0.08881
empty	-UNORM
empty	+UNORM
empty	+UNORM
empty	+UNORM
empty	3
0000	Cond 0
027F	Prec NE

ESI=0040506E (keygen.0040506E)

Increment Plus
 Decrement Minus
 Zero
 Set to 1
 Modify Enter
 Copy selection to clipboard Ctrl+C
 Copy all registers to clipboard
Follow in Dump
 Follow in Stack
 View MMX registers
 View 3DNow! registers
 View debug registers
 Appearance

Address	Hex dump	ASCII
0012FFC0	00 00 00 00 69 EB E7 77 0E 80 00 00 68 CD 12 00i\$rwB\$.h=.
0012FFD0	00 F0 FD 7F F4 DC EE F9 C8 FF 12 00 0E 6A 53 80	..=20f_e-4+.AjS0
0012FFE0	FF FF FF FF 86 B8 E9 77 18 5A E9 77 00 00 00 00	..0w+20w....
0012FFF0	00 00 00 00 00 00 00 00 7A 50 40 00 00 00 00 00zPe.....

Address	Hex dump	Comment
0012FFC0	00000003	
0012FFC4	77E7EB69	RETURN to kernel32.77E7
0012FFC8	0000800E	
0012FFCC	0012CD68	
0012FFD0	7FFDF000	
0012FFD4	F9EEDCF4	
0012FFD8	0012FFC8	
0012FFDC	80536A0E	
0012FFE0	FFFFFFFF	End of SEH chain
0012FFE4	77E9BB86	SE handler
0012FFE8	00000000	

Then you go to the Hex dump window. Then right click on the value **0012FFC0** and select **Breakpoint -> Hardware, on Access -> Word**. Our breakpoint is now set.

Now, when our breakpoint was set. We press **F9** 8 times. And we still here:

00401000	6A 40	PUSH 40	
00401002	68 00304000	PUSH keygen.00403000	ASCII "===[UCT-Viet Cracking Team
00401004	68 33304000	PUSH keygen.00403033	ASCII " --+=[SoFtWare iFORMAT
00401006	6A 00	PUSH 0	
00401008	E8 70020000	CALL keygen.00401290	JMP to user32.MessageBoxA
0040100A	6A 00	PUSH 0	
0040100C	E8 4C020000	CALL keygen.00401266	JMP to kernel32.GetModuleHandleA
0040100E	6A 01	PUSH 1	
00401010	6A 00	PUSH 0	
00401012	6A 00	PUSH 0	
00401014	50	PUSH EAX	
00401016	E8 68000000	CALL keygen.0040108E	
00401018	6A 40	PUSH 40	
0040101A	68 F5314000	PUSH keygen.004031F5	ASCII "Final paroles!"
0040101C	68 04324000	PUSH keygen.00403204	ASCII "More cracks, please contac
0040101E	6A 00	PUSH 0	
00401020	E8 57020000	CALL keygen.00401290	JMP to user32.MessageBoxA
00401022	50	PUSH EAX	
00401024	E8 21020000	CALL keygen.00401266	JMP to kernel32.ExitProcess
00401026	55	PUSH EBP	
00401028	8BEC	MOV EBP,ESP	
0040102A	837D 0C 10	CMP DWORD PTR SS:[EBP+C],10	
0040102C	75 0D	JNZ SHORT keygen.00401055	
0040102E	6A 00	PUSH 0	
00401030	E8 47020000	CALL keygen.00401296	JMP to user32.PostQuitMessage
00401032	33C0	XOR EAX,EAX	
00401034	C9	LEAVE	
00401036	C2 1000	RETN 10	

Congratulations! According OEP we found is **401000** And now we Calculate the real OEP of this unpackme by the formula:
Real OEP = OEP find in Olly-Image Base = 401000-400000 = **1000**.

4th dumping

At address **0041000** we go to the menu Plugin -> OllyDump -> dump debugged process. And then, just press dump, save the file at **dumped.exe** unpacked.

00401000	6A 40	PUSH 40	
00401002	68 00304000	PUSH unpackme.00403000	ASCII "===[UCT-Viet Cracking Team
00401004	68 33304000	PUSH unpackme.00403033	ASCII " --+=[SoFtWare iFORMAT
00401006	6A 00	PUSH 0	
00401008	E8 70020000	CALL unpackme.00401290	JMP to user32.MessageBoxA
0040100A	6A 00	PUSH 0	
0040100C	E8 4C020000	CALL unpackme.00401266	JMP to kernel32.GetModuleHandleA
0040100E	6A 01	PUSH 1	
00401010	6A 00	PUSH 0	
00401012	6A 00	PUSH 0	
00401014	50	PUSH EAX	
00401016	E8 68000000	CALL unpackme.0040108E	
00401018	6A 40	PUSH 40	
0040101A	68 F5314000	PUSH unpackme.004031F5	ASCII "Final paroles!"
0040101C	68 04324000	PUSH unpackme.00403204	ASCII "More cracks, please contac
0040101E	6A 00	PUSH 0	
00401020	E8 57020000	CALL unpackme.00401290	JMP to user32.MessageBoxA
00401022	50	PUSH EAX	
00401024	E8 21020000	CALL unpackme.00401266	JMP to kernel32.ExitProcess
00401026	55	PUSH EBP	
00401028	8BEC	MOV EBP,ESP	
0040102A	837D 0C 10	CMP DWORD PTR SS:[EBP+C],10	
0040102C	75 0D	JNZ SHORT unpackme.00401055	
0040102E	6A 00	PUSH 0	
00401030	E8 47020000	CALL unpackme.00401296	JMP to user32.PostQuitMessage
00401032	33C0	XOR EAX,EAX	
00401034	C9	LEAVE	
00401036	C2 1000	RETN 10	

Do not run **dumped.exe** now, will be a crash ... It must fix IAT.

5. Finding and Fixing the Adress Import Table

And select Open ImpREC attached to active process and choose unpackme.exe. Change the value in the OEP window to the one we wrote down earlier (1000) then select IATAutosearch then click Get Imports.

Attach to an Active Process

e:\cracker\tutorials\hacnho\unpacking mew10\unpackme.exe (0000060C) Pick DLL

Imported Functions Found

- kernel32.dll FTunk:00002000 NbFunc:2 (decimal:2) valid:YES
- user32.dll FTunk:0000200C NbFunc:C (decimal:12) valid:YES

Show Invalid

Show Suspect

Auto Trace

Clear Imports

Log

Original IAT RVA found at: 00002024 in Section RVA: 00001000 Size:00004000
IAT read successfully.

Current imports:
2 (decimal:2) valid module(s) (added: +2 (decimal:+2))
E (decimal:14) imported function(s). (added: +E (decimal:+14))

Clear Log

IAT Infos needed

OEP 00001000 IAT AutoSearch

RVA 00001FFC Size 00000044

Load Tree Save Tree Get Imports

New Import Infos (IID+ASCII+LOADER)

RVA 00000000 Size 00000144

☒ Add new section

Options

About

Exit

Fix Dump

All Import Functions valid.

Now, click fix dump to fix the IAT **dumped.exe** file.

Use LordPE 1.4 by Y0da for rebuild our Dumped_.exe

[Rebuild Status]

Starting to rebuild dumped .exe...

Filesize: 5000h

Wipe Relocation...no Relocation present

Realigning...done

Current filesize: 3946h

File minimized to: 71%

Rebuild ImportTable...not needed

Validate PE image...done

New filesize: 3946h

File minimized to: 71%

Rebuilding finished.

OK

Now run unpacked files. Wow, not crash.

6. Conclusion

Special thanx to **R @ dier** for this template.

My Greetz to: Deux, infinite, NVH (c), softcracker_vn, luucorp, Aaron, hhphong, R @ dier, tlandn, Computer_Angel, Zombie, RCA, CTL, Moonbaby, Ferrari, Devilz, Neitsa, anh_surprise ... and you; -)!

To be continued ...

Written by [hacnho](#) (tutorial date: Sai Gon 20/5/2004)

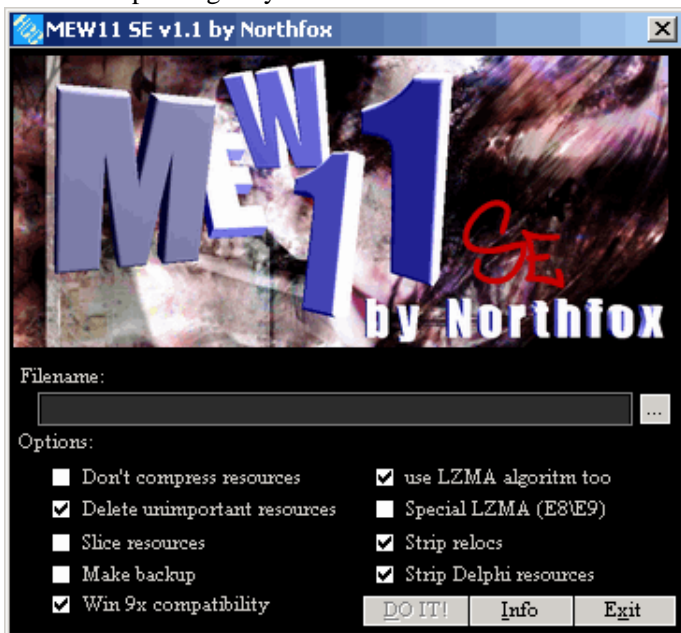
hacnho Tutorials # 15

Manual unpacking MEW 11 SE v1.1 - > Northfox [HCC]

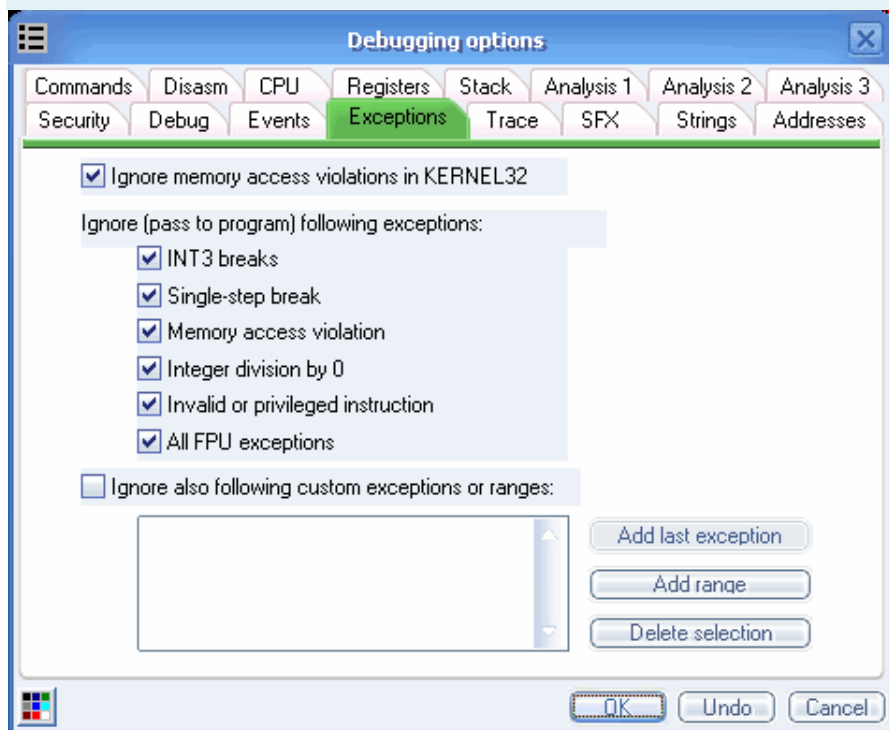
Information	Unpacking for Newbie's
Target	target1
Available	http://nhandan.info/hacnho/tuts/unpackme15_tuts.zip
Tools	1:10 OllyDbg plugin with Final OllyDump 2.21.108, OllyScript, Lord PE 1.4, 0.92 and PEiD EM Editor for 4:04 write scripts.
Protection	FSG 2.0 -> bart / XT
Level	Easy
Category	Manual unpacking

1.Introduction

Northfox [HCC] was released his packer. This is MEW 11 SE v1.1. I written this tut for help anyone want to learn a follow unpacking easy method.



Before unpacking, you can edit your options OllyDBG follow my method.

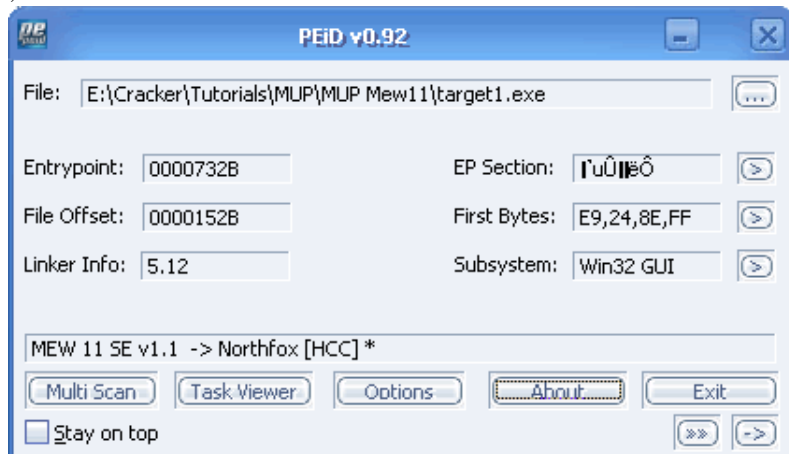


Since 25 June 2004, was not snaker PEiD0.92 update, so it can not detect this packer. I found was the sign of the packer. If you want, you can add those text string into the folders in the userdb.txt was installed PEiD.

```

;-----
[MEW 11 SE v1.1 -> Northfox [HCC]]
signature = E9? ? ? FF 0C? 0
ep_only = true
;-----

```



2.Find OEP

Step 1: Find OEP

Load-1 into target OllyDBG

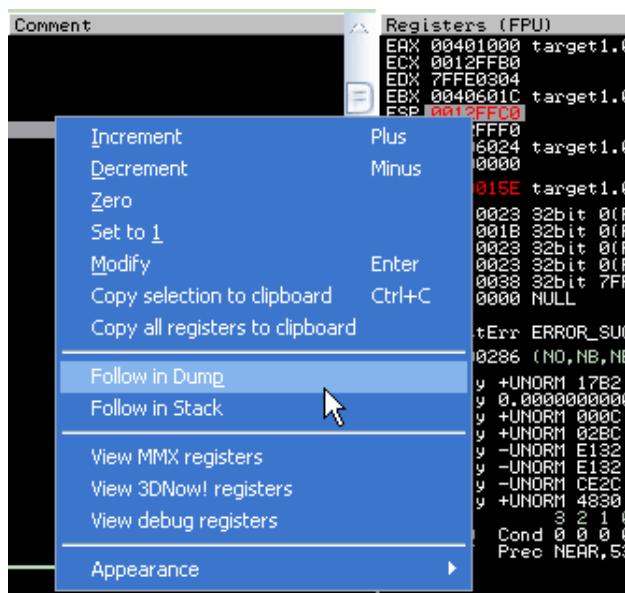
Address	Hex dump	Disassembly	Comment
0040732B	E9 248EFFFF	JMP target1.00400154	
00407330	0C 60	OR AL, 60	
00407332	0000	ADD BYTE PTR DS:[EAX], AL	
00407334	0000	ADD BYTE PTR DS:[EAX], AL	
00407336	0000	ADD BYTE PTR DS:[EAX], AL	
00407338	0000	ADD BYTE PTR DS:[EAX], AL	
0040733A	0000	ADD BYTE PTR DS:[EAX], AL	
0040733C	0273 00	ADD DH, BYTE PTR DS:[EBX]	
0040733F	000C60	ADD BYTE PTR DS:[EAX], CL	
00407342	0000	ADD BYTE PTR DS:[EAX], AL	
00407344	8978 40	MOV DWORD PTR DS:[EAX+40], EDI	
00407347	0000	ADD BYTE PTR DS:[EAX], AL	
00407349	0000	ADD BYTE PTR DS:[EAX], AL	

Press **F8** 6 times (you can see the ESP register in FPU is highlighting Windows):

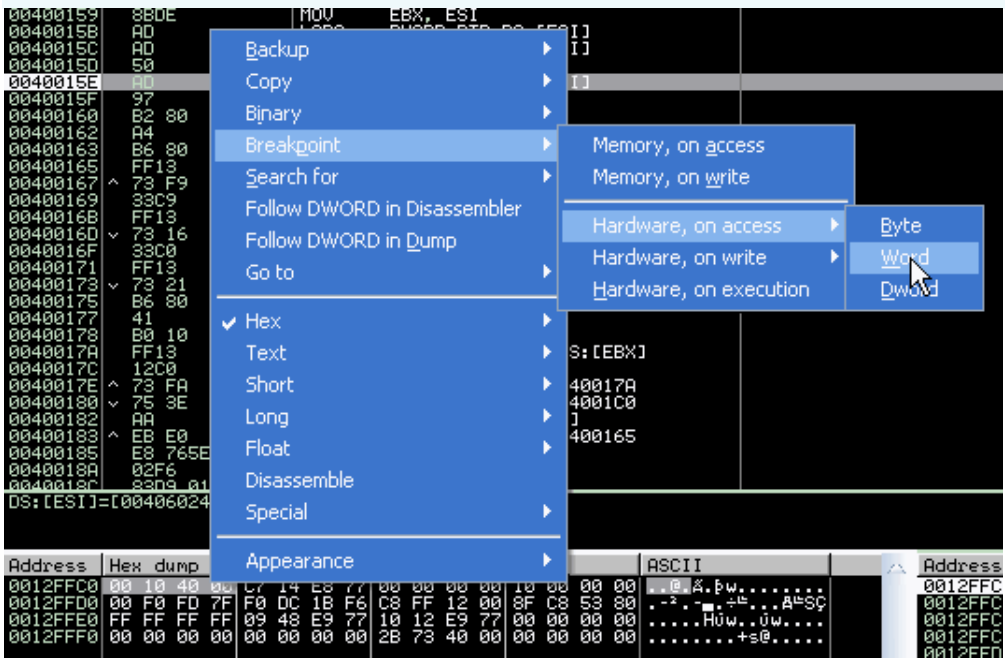
Address	Hex dump	Disassembly	Comment
00400154	BE 1C604000	MOV ESI, target1.0040601C	
00400159	8B0E	MOV EBX, ESI	
0040015B	AD	LODS DWORD PTR DS:[ESI]	
0040015C	AD	LODS DWORD PTR DS:[ESI]	
0040015D	50	PUSH EAX	
0040015E	AD	LODS DWORD PTR DS:[ESI]	
0040015F	97	XCHG EAX, EDI	
00400160	B2 80	MOV DL, 80	
00400162	A4	MOVS BYTE PTR ES:[EDI], BYTE PTR DS:	
00400163	B6 80	MOV DH, 80	
00400165	FF13	CALL NEAR DWORD PTR DS:[EBX]	
00400167	73 F9	JNB SHORT target1.00400162	
00400169	33C9	XOR ECX, ECX	
0040016B	FF13	CALL NEAR DWORD PTR DS:[EBX]	
0040016D	73 16	JNB SHORT target1.00400185	
0040016F	33C0	XOR EAX, EAX	
00400171	FF13	CALL NEAR DWORD PTR DS:[EBX]	

Registers (FPU)		
EAX	00401000	target1.00401000
ECX	0012FFB0	
EDX	7FFE0304	
EBX	0040601C	target1.0040601C
ESP	0012FFC0	
EBP	0012FFF0	
ESI	00406024	target1.00406024
EDI	00000000	
EIP	0040015E	target1.0040015E
C 0	ES 0023	32bit 0(FFFFFFFF)
P 1	CS 001B	32bit 0(FFFFFFFF)
0 0	SS 0023	32bit 0(FFFFFFFF)

At this line, you see in the Registers (FPU) table. The value of ESP is **0012FFC0**, then you right click on and choose the **ESP Follow in dump**



Then you go to the Hex dump window. Then right click on the value 0012FFC0 and select Breakpoint -> Hardware, on Access -> Word. Our breakpoint is now set.



Press **F9** to run. You will still be on OEP

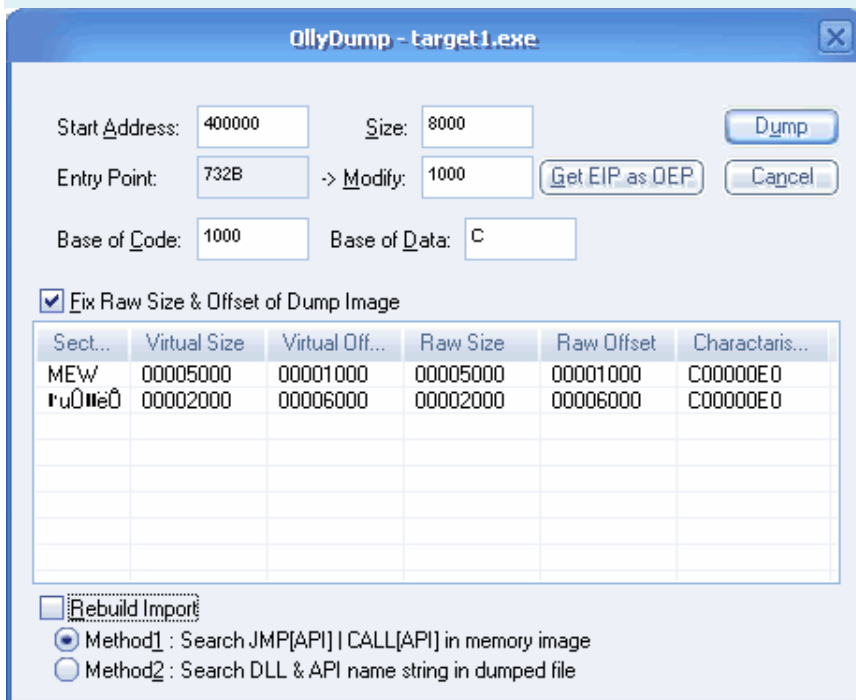
Address	Hex dump	Disassembly	Comment
00401000	6A 00	PUSH 0	
00401001	00		
00401002	E8 00	CALL target1.00401000	
00401003	00		
00401004	01		
00401005	00		
00401006	00		
00401007	A3 00	MOV DWORD PTR DS:[403000], EAX	
00401008	00		
00401009	30		
0040100A	40		
0040100B	00		

Press **Ctrl + A** for analyze.

Address	Hex dump	Disassembly	Comment
00401000	6A 00	PUSH 0	
00401002	E8 00010000	CALL target1.00401114	Module = NULL
00401007	A3 00304000	MOV DWORD PTR DS:[403000], EAX	GetModuleHandleA
0040100C	6A 00	PUSH 0	
0040100E	68 29104000	PUSH target1.00401029	
00401013	6A 00	PUSH 0	
00401015	6A 65	PUSH 65	
00401017	FF35 00304000	PUSH DWORD PTR DS:[403000]	
0040101D	E8 FE000000	CALL target1.00401120	
00401022	6A 00	PUSH 0	
00401024	E8 E5000000	CALL target1.0040110E	
00401029	55	PUSH EBP	

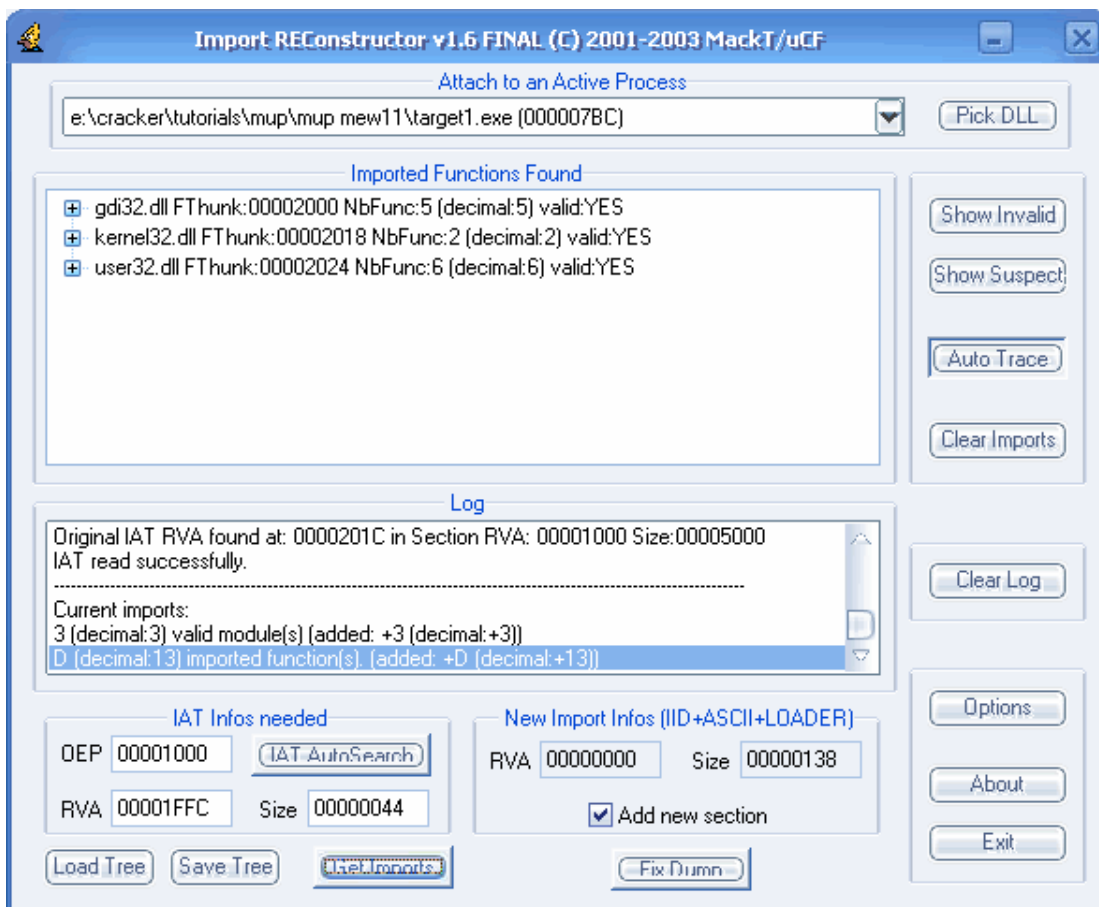
Step 2: dumping

- Go to the menu plugin, choose OllyDUMP:



Step 3: Finding and Fixing the Address Import Table

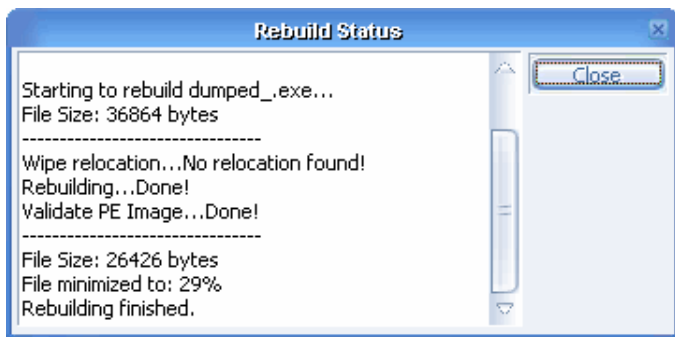
And select Open ImpREC attached to active process and choose unpackme.exe. Change the value in the OEP window to the one we wrote down earlier (1DFE) then select IATAutosearch then click Get Imports.



Now, click fix to fix IAT dump the file **dumped.exe**

Step 3: rebuild PE

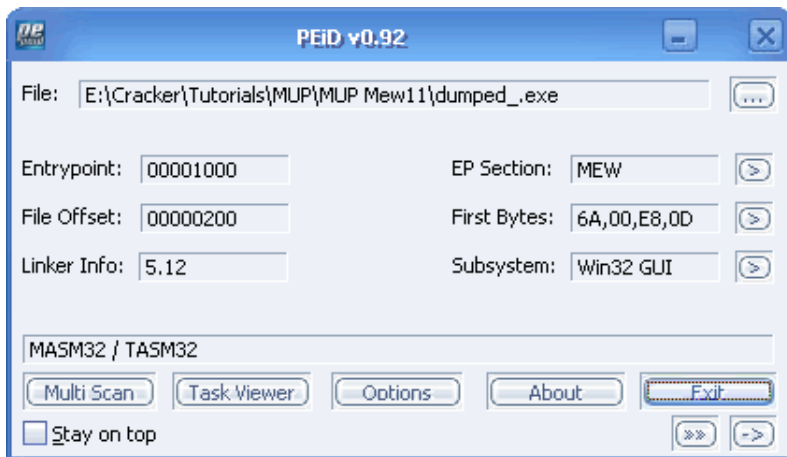
Use LordPE 1.4 by Y0da for rebuild our **dumped_.exe**



- Unpacked successful! Done ...

3. Testing Our Unpacked file

Now run the unpacked files. It's Okay! Using **PEiD 0.92** for detect: **MASM32 / TASM32**. Okie, MEW 11 SE v1.1 is now unpacked successful!



4. Create OllyScript

After find OEP in Olly, we create a need for auto OllyScript find OEP next time! Remember! We have found three step for OEP:

1. First: Find the special signal of MEW breakpoint set for 11!
 2. Second: Step over 6 times.
 3. Final: Set breakpoint, press F9 to run for jump to **OEP**
- Okay!**, Write in our step language OllyScript

```

Cut here -----
/*
////////////////////////////////////
// MEW 11 SE v1.1 -> Northfox [HCC] OEP finder
// Author: hacnho/VCT2k4
// Email: hacnho@hotmail.com
// Website: http://nhandan.info/hacnho
// OS: WinXP Pro, OllyDbg 1:10 Final, OllyScript v0.85
////////////////////////////////////
*/
STI // Step into (F7)

```

```
Sto // Step over (F8)
Sto
Sto
Sto
Sto
eob Break
findop eip, 50AD # # // Find the special signal
bphws ESP, "r" // Set a breakpoint on memory access
run // Run the program

Bread:
Police eip // Ctrl + A for Analyze
log eip // Logs to source OllyDbg log window.
CMT eip, "This is the OEP! Found by hacnho/VCT2k4" // Write a comment
Msg "Dumped and IAT fix now! Thanx for using my script ...!" // Show a message

ret // Exits script
Cut here -----
```

6. Conclusion

GrEeTs Fly Out: Deux, infinite, NVH (c), softcracker_vn, luucorp, Aaron, Canterwood, hhphong, R @ dier, tlandn, Computer_Angel, Zombie, RCA, CTL, Moonbaby, Nilrem, diablo2oo2, Ferrari, Devilz, anh_surprised .. . and you ;-)!
Thanx to authors of OllyDBG, ImpREC, LordPE, OllyScript, MEW 11,
To be continued ...
Written by [hacnho](#) (tutorial date: Saigon 23/08/2004)

FRIENDS SITE

[\[Exetools Forum\]](#) | [\[HVAOnline\]](#) | [\[Vncracking Group\]](#) | [\[REA Forum\]](#) | [\[hacnho's homepage\]](#) | [\[AR Team\]](#)
| [\[Vicki's Fan\]](#) | [\[Devilz Crack\]](#) |

...: Copyright © 2004 by hacnho <[==]> VCT-Vietnamese Cracking Team 2k4:: ..

::: [Manual Unpacking MoleBox v2.5.7 and Serial Fishing]:::

(_kienmanowar_)



I. Proem

Today some of the brothers in the REA's of Com and Trick working visit also made me itch career unbearable ... more he Com Trick and online every day to make a new status as the them Enhance to open. Ặc where he ngặt but they play the first data is not that, my career and flood earth is not known. Luckily, today I love Com rồi Rai, two brothers seated man step by step. With the help of his Com today I will write a small how **"Manual Unpacking a program packed with MoleBox (specifically ver 2.5.7)"** also performed well methods **"Serial Fishing"** . *a reminder of what I do not do the following purposes outside academic and research so I will not bear any responsibility if you do use it on purpose is not good!* Ok13! L3t's R0ck w1th m3 J

II. Target and Tools

Target:

Name: **WMV to AVI MPEG DVD WMV Converter**

Home site: <http://www.alloksoft.com/wmv.htm>

Description: WMV to AVI MPEG DVD WMV Converter is a powerful tool for Splendid and WMV to AVI, WMV to MPEG, WMV to DVD, WMV to VCD and WMV to SVCD video converter.

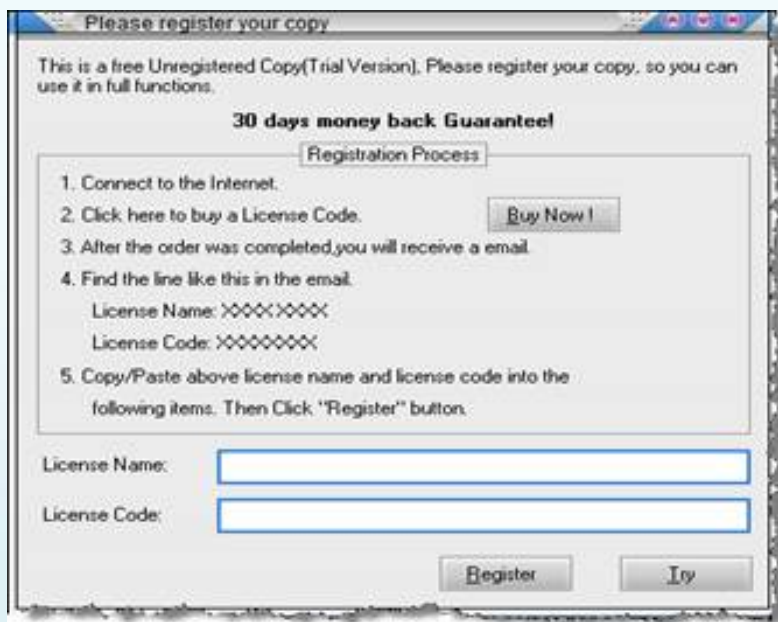
Tools:

Debugger: Ollydbg

PE Tools: PeiD, RDG Packer Detector, LordPE, ImpRec

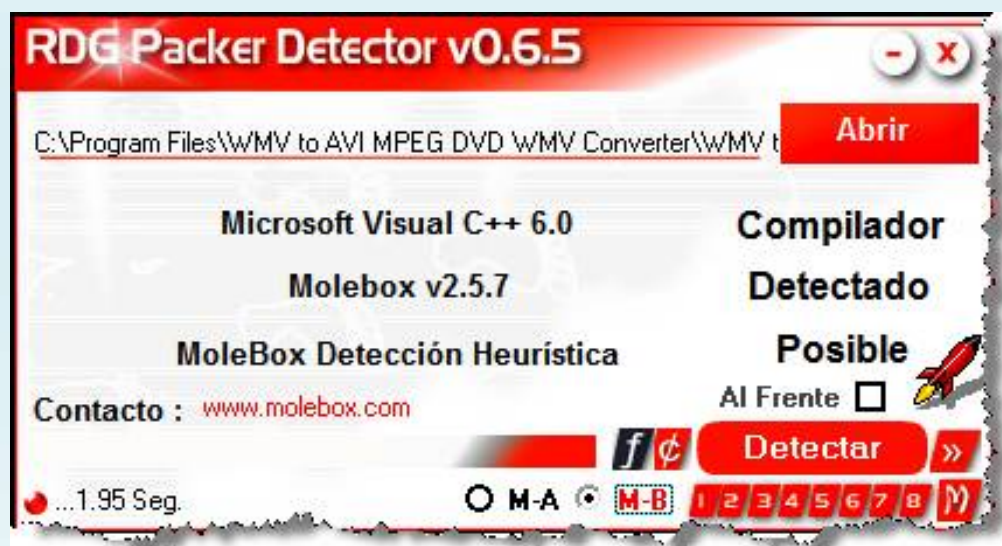
III. Manual Unpacking

I assume you down load the program and install the machines. Test the program you will see the following:



Ok so we know that this program will only give us 30 days to try and stop the use will be limited features. Now as usual we will check the use Packer / Protector time. I used **PeiD** and **RDG Packer Detector** to find information:





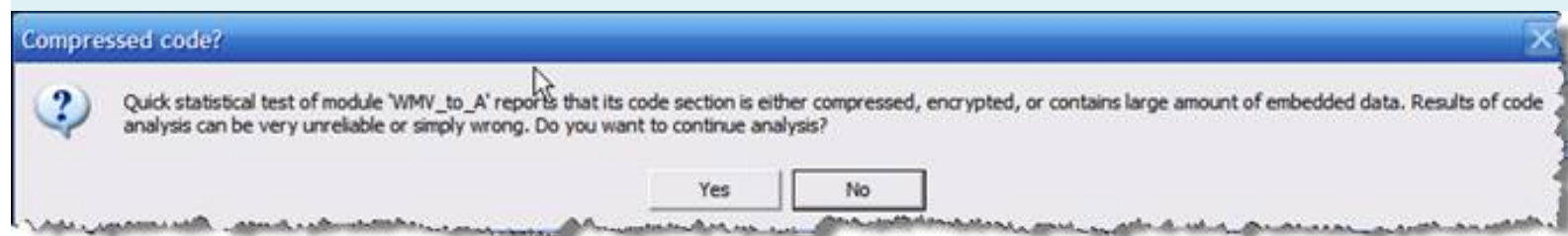
If we use the function in Deep Scan PeiD we also have the results that this program is using the code: **Microsoft Visual C++ 6.0**. Here I am not very much interested to CrypTo that this program to I just use that focuses on information that PeiD RDG and provide that: **MoleBox v2.5.7**. Information that I have been on the Packer this as follows:

***MoleBox is a runtime exe packer for Windows applications.** It bundles the executable together with the dll files and data into a single EXE file, without losing the ability to run the application.*

MoleBox compresses and encrypts all the application files. With MoleBox you can protect your application's data and media files from viewing and modifications, and your DLLs and ActiveX components from usage by third party programs. (More at: <http://www.molebox.com/features.shtml>).

Before I do not have the opportunity to try this they Packer, just read a few Tuts in which items of J Trick. Yesterday he was Com ru happy to do the same (as he also Com MUP and the child is finished), the dose should also try and make the final kaka today sit writing tut J .

OK so the information is enough, now we load the program to proceed and Olly MUP.Khi to load Olly if you encounter a message similar to the degree the "No" to ignore it and continue work:

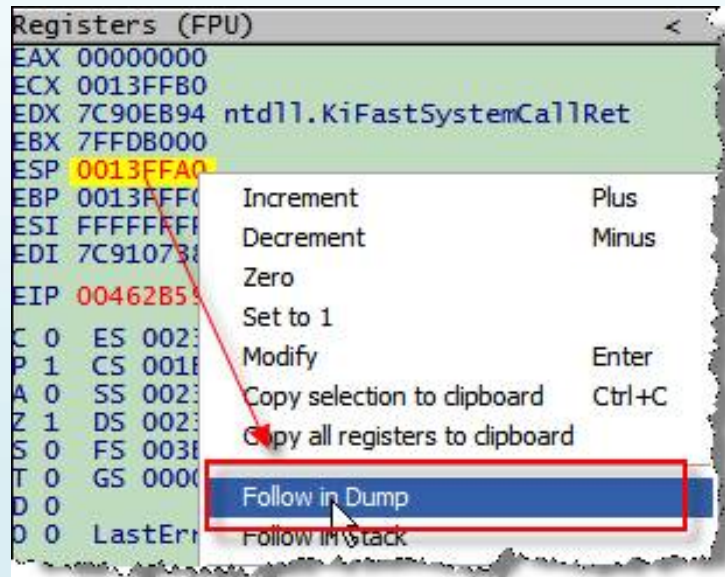


1st Search OEP and dump file:

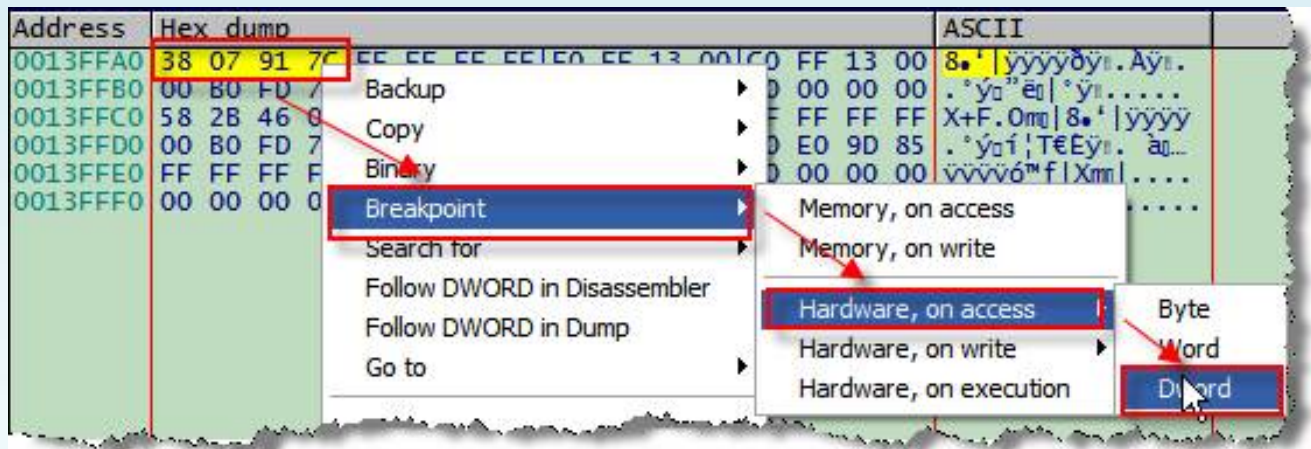
After you click "No" we stop here in Olly:

Address	Hex dump	Disassembly	Comment
00462B53	E8 00000000	CALL WMV_to_A.00462B58	
00462B58	60	PUSHAD	
00462B59	E8 4F000000	CALL WMV_to_A.00462BAD	
00462B5E	A3 C52E9F2D	MOV DWORD PTR DS:[2D9F2EC5], EAX	

Observations hairbreadth you'll see a command **PUSHAD**, so there are signs this month borrowing by way UPX. We press **F8** to trace through PUSHAD command, observation window and select **Registers** record **ESP**, click to select **Follow in dump**:



At window dump we choose DWORD, and set a BP on HW access similar image illustrated below:



Press **F9** to run the program, we will Break here in Olly:

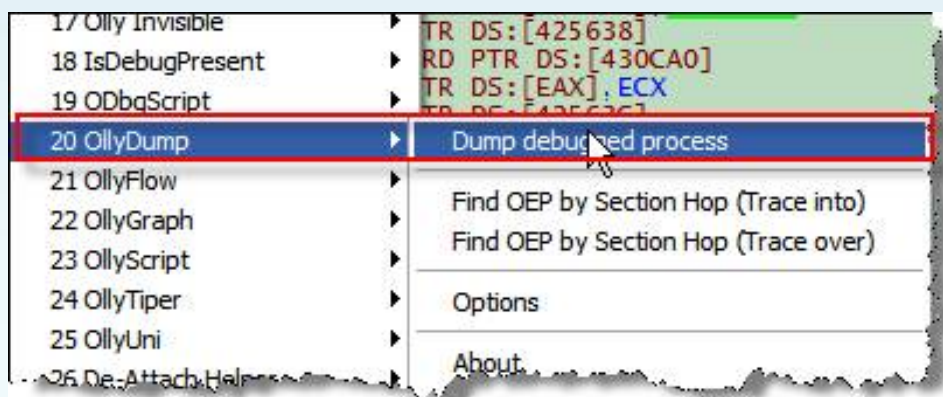
Address	Hex dump	Disassembly	Comment
00462730	61	POPAD	
00462731	58	POP EAX	WMV_to_A.00462B58
00462732	58	POP EAX	
00462733	FFD0	CALL EAX	
00462735	E8 A6C00000	CALL WMV_to_A.0046E7E0	
0046273A	CC	INT3	
0046273B	CC	INT3	

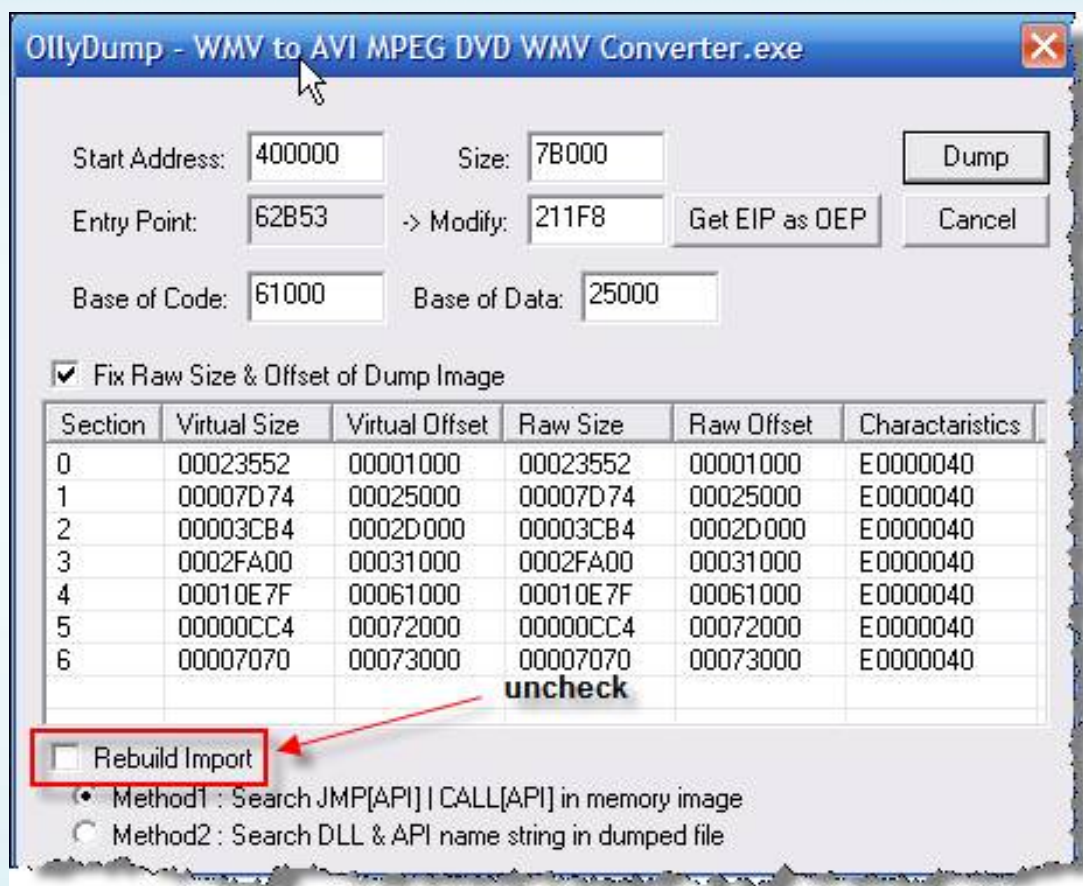
Hardware breakpoint 1 at WMV_to_A.00462731 - EIP points to next instruction

2 Press **F8** to trace through **POP EAX 2**, and press **F7** to Trace Call to order at **0x00462733**, we will go to the OEP of the program:

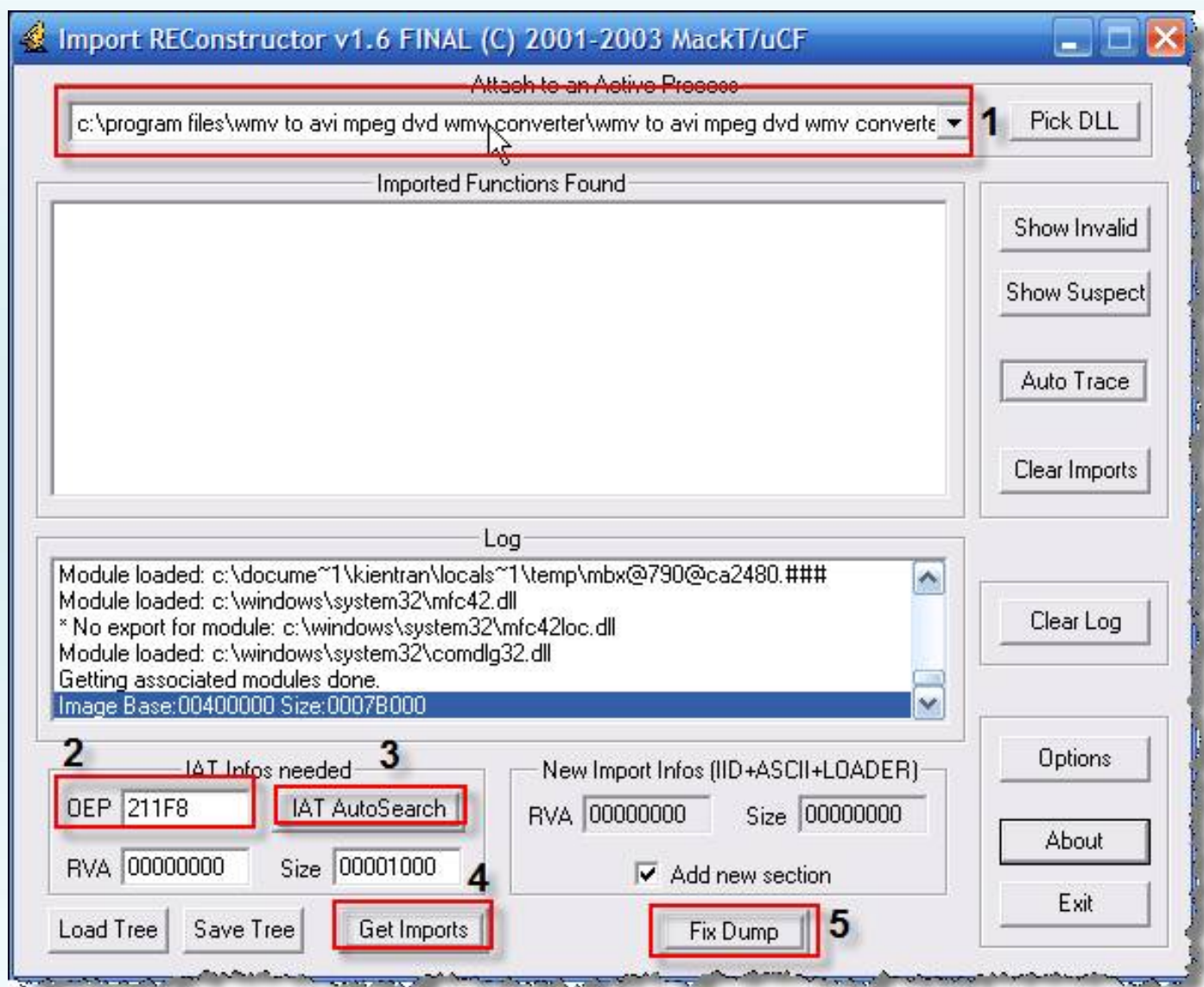
Address	Hex dump	Disassembly	Comment
004211F8	55	PUSH EBP	OEP
004211F9	88EC	MOV EBP, ESP	
004211FB	6A FF	PUSH -1	
004211FD	68 00854200	PUSH MMV_to_A.00428500	
00421202	68 84134200	PUSH MMV_to_A.00421384	
00421207	64 A1 00000000	MOV EAX, DWORD PTR FS:[0]	JMP to msvcrt._except_handler3
0042120D	50	PUSH EAX	
0042120E	64 8925 00000000	MOV DWORD PTR FS:[0], ESP	
00421215	83EC 68	SUB ESP, 68	

As usual when we work with the Packers, to the OEP and the dump and only J Fix IAT. Use Plugin OllyDump to dump and Save the file with the name of any options you put me here is dumped.exe.

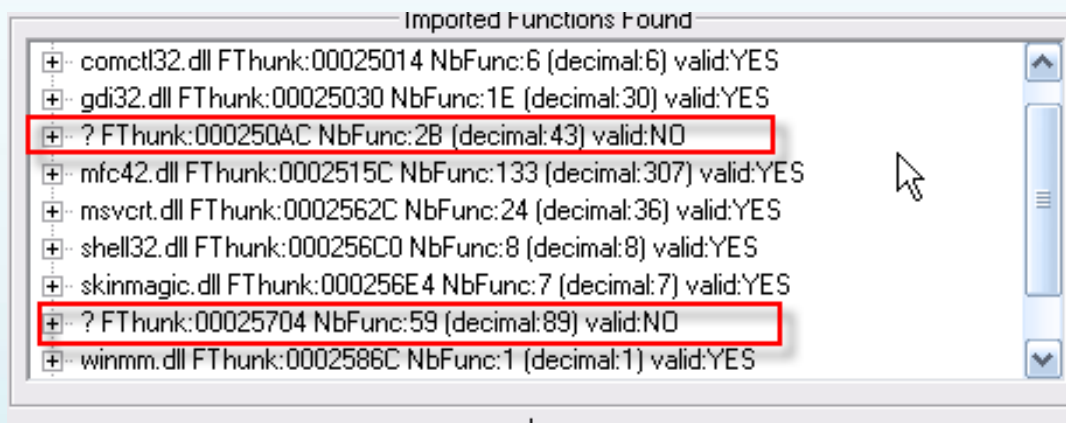




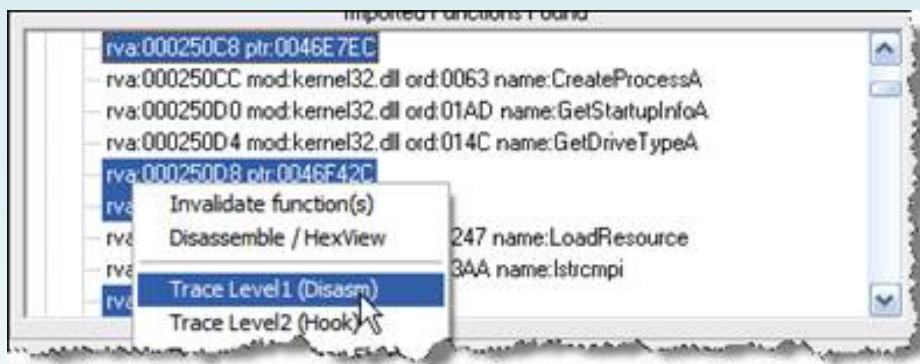
After the dump is complete, open and ImportRec conducted Fix IAT.



In the Get Imports may Invalid image as illustrated below:



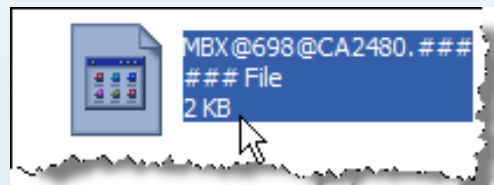
Now click on the button "Show Invalid", on a mouse to any function and as follows:



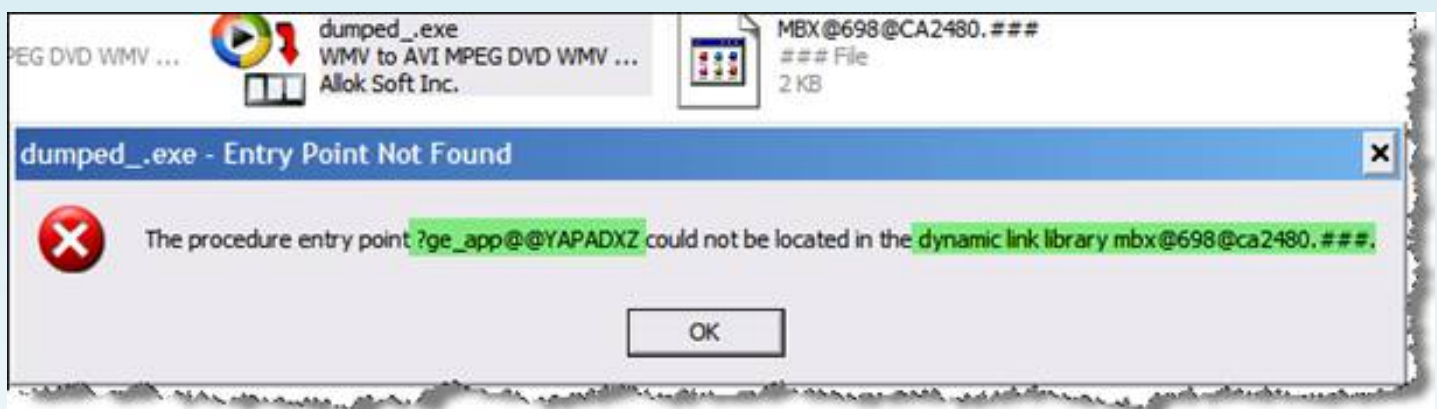
After implementation we see all have a valid function, click and select Fix dump file we have to dump at the Fix IAT. Tolerable J too simple, we try to run the file has fix dump see how:



Ặc Failed! With a reason it can not find the file qui quá What is **mbx @ 698** ... a search we have found this file in the Temp directory:



We copy the file in the Temp directory to install the program and try Run the file dumped_.exe see how.



Through this we draw the conclusion rằng MoleBox will be embedded in a dll file, but this dll file is encrypted to the name, when we run exe file it will extract the dll file and load it into the .s programs we will use the functions in this dll file. So this dll file name is what and how to get it? The second task of our J

2. Find dll, and dump Fix IAT, reloc. Fix dump and IAT of exe files.

Okie, so that we have a better overview, now you delete all the files have dump fix IAT and we will take a leo beginning to end J . Load the file in Olly, as the above steps until we stop at **EAX Call** function as above.

Address	Hex dump	Disassembly	Comment
00462733	FFD0	CALL EAX	WNV_to_A.004211F8
00462735	E8 A6600000	CALL MOV_to_A.0046E7E0	

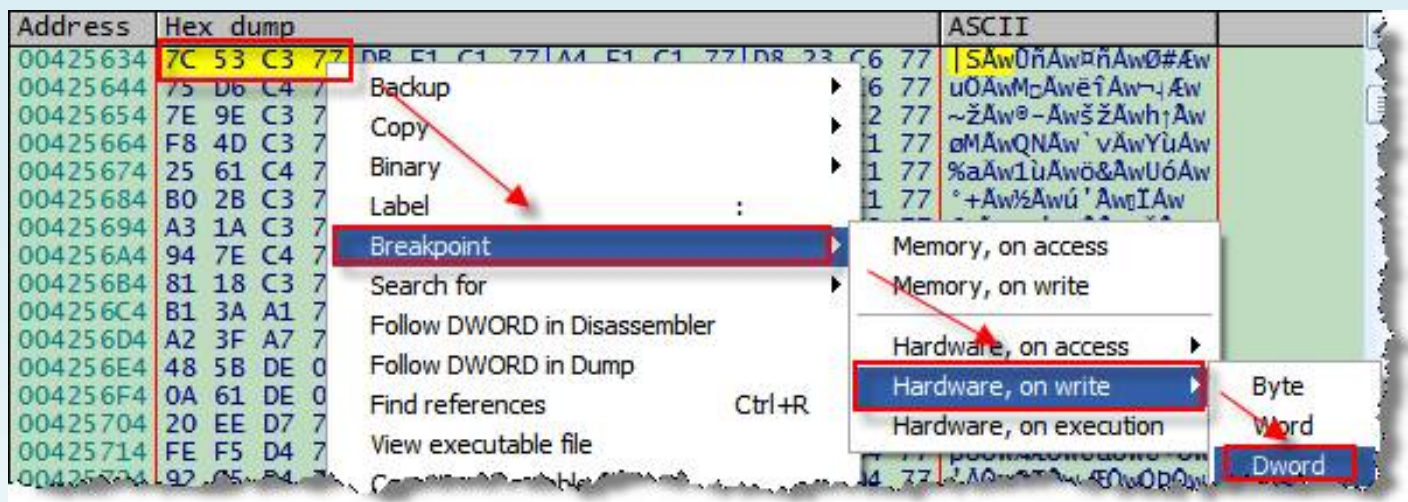
We will set a BP in order for this Call is ordered Call us will go to the OEP program. We see this month it destroyed IAT, which we read Invoice's place will be called to cancel the IAT this function (similar Armadillo) so that we must find command will jump to jump over the cancellation IAT (or often called with the name "magic jump"). Ok press Enter **Call EAX** in order, we will go to OEP of trinh.Tai here look down below 1 paragraph you'll see a command as follows:

00421220	895D FC	MOV	DWORD PTR SS:[EBP-4], EBX	
00421223	6A 02	PUSH	2	
00421225	FF15 34564200	CALL	DWORD PTR DS:[425634]	msvcrt.__set_app_type
00421228	59	POP	ECX	
0042122C	830D AC0C4300 FF	OR	DWORD PTR DS:[430CAC], FFFFFFFF	
00421233	830D 800C4300 FF	OR	DWORD PTR DS:[430CB0], FFFFFFFF	

To vet this morning in order as illustrated in the image above and select Follow in dump> Memory Address. In the window dump we will see many functions Api.

Address	Value	Comment
00425634	77C3537C	msvcrt.__set_app_type
00425638	77C1F1D8	msvcrt.__p_fmode
0042563C	77C1F1A4	msvcrt.__p_commode
00425640	77C623D8	OFFSET msvcrt._adjust_fdiv
00425644	77C4D675	msvcrt._setusermatherr
00425648	77C3084D	msvcrt._setmbcp
0042564C	77C1EEEE	msvcrt._getmainargs
00425650	77C617AC	OFFSET msvcrt._acmdln
00425654	77C39E7E	msvcrt._exit
00425658	77C32DAE	msvcrt._XcptFilter
0042565C	77C39E9A	msvcrt._exit
00425660	77C21868	msvcrt.type_info::~type_info
00425664	77C34DF8	msvcrt._onexit
00425668	77C34E51	msvcrt._dllonexit
0042566C	77C34E51	msvcrt._dllonexit

At **0x00425643** we will set a BP on HW write> Dword to see thẳng will write the address IAT to remember this region.



Then **Ctrl + F2** to Restart Olly again. Press **F9** first time we here at Break:

Address	Hex dump	Disassembly	Comment
00467B74	8B45 F8	MOV EAX, DWORD PTR SS:[EBP-8]	WMV_to_A.00425634
00467B77	40	INC EAX	
00467B78	40	INC EAX	
00467B79	8945 F8	MOV DWORD PTR SS:[EBP-8], EAX	
00467B7C	0FB745 D4	MOVZX EAX, WORD PTR SS:[EBP-2C]	

Dom down window dump found no significant (ie not appear correctly Api like Figure above). Press **F9** to continue, at the Break:

00467B92 5B POP EBX; 7FFDF000

Continue to press **F9** to see the API functions appear similar to the above and we stopped here in Olly:

Address	Hex dump	Disassembly	Comment
004696B1	EB 2C	JMP SHORT WMV_to_A.004696DF	
004696B3	8B55 F4	MOV EDX, DWORD PTR SS:[EBP-C]	
004696B6	8B02	MOV EAX, DWORD PTR DS:[EDX]	
004696B8	25 FFFF0000	AND EAX, 0FFFF	
004696BD	8945 D0	MOV DWORD PTR SS:[EBP-30], EAX	
004696C0	8B4D D0	MOV ECX, DWORD PTR SS:[EBP-30]	
004696C3	51	PUSH ECX	
004696C4	8B55 EC	MOV EDX, DWORD PTR SS:[EBP-14]	
004696C7	52	PUSH EDX	
004696C8	FF15 A8564700	CALL DWORD PTR DS:[475648]	kernel32.GetProcAddress

Address	Value	Comment
00425634	77C3537C	msvcrt.__set_app_type
00425638	0002C126	
0042563C	0002C116	
00425640	0002C106	

Ok so we are in the area and IAT. Looking down a bit, we'll see a command Call:

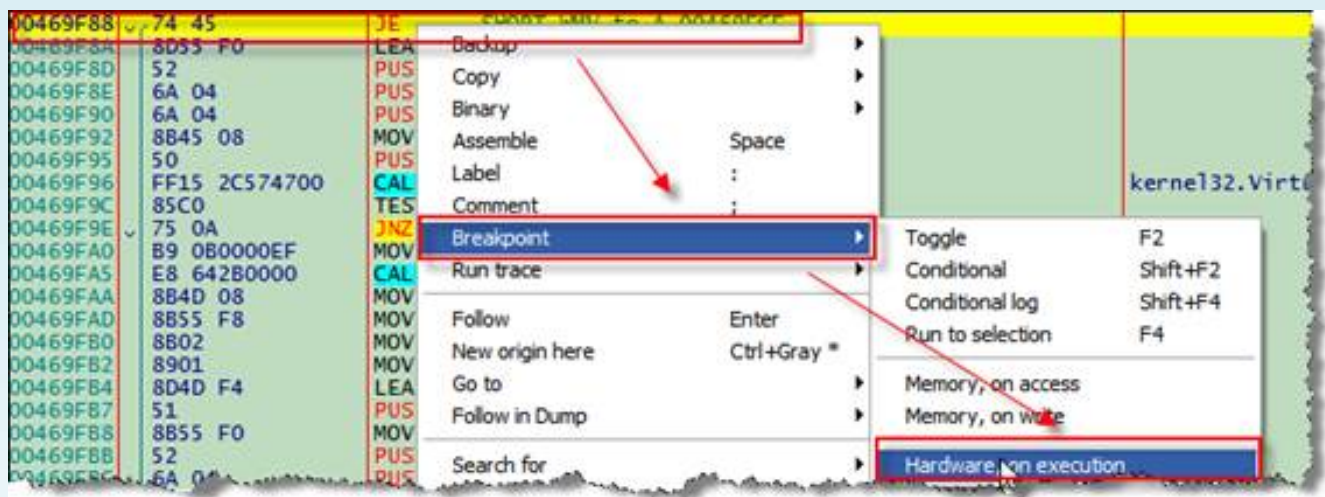
004696FA	52	PUSH EDX
004696FB	E8 50080000	CALL WMV_to_A.00469F50
00469700	83C4 0C	ADD ESP, 0C

Giang in a store that people here that contain the magic jump, it will have to consider cancellation IAT API or không.Nhan Enter to Call to order in this and look down a bit we will see a command JE-year jump between the two orders jumped JNZ, here is the death of huyet MoleBox the rumor's giang lake.

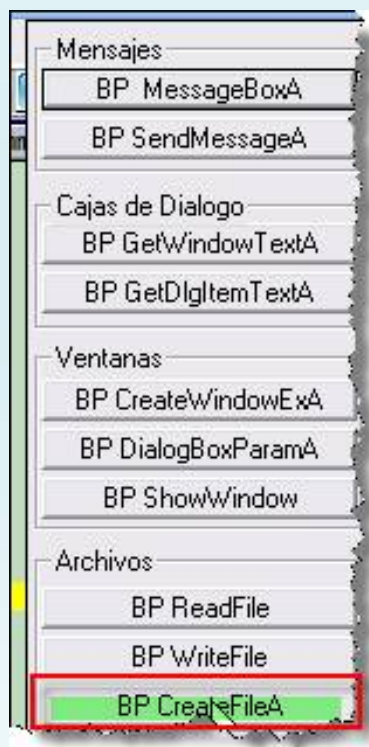
```

00469F64 75 0A JNZ SHORT WMV_to_A.00469F70
00469F66 B9 0A0000EF MOV ECX, 0A0000EF
00469F68 E8 9E2B0000 CALL WMV_to_A.0046CB0E
00469F70 8B45 08 MOV EAX, DWORD PTR SS:[EBP+8]
00469F73 8B08 MOV ECX, DWORD PTR DS:[EAX]
00469F75 51 PUSH ECX
00469F76 8B0D 34A04700 MOV ECX, DWORD PTR DS:[47A034]
00469F7C E8 B45C0000 CALL WMV_to_A.0046FC35
00469F81 8945 F8 MOV DWORD PTR SS:[EBP-8], EAX
00469F84 837D F8 00 CMP DWORD PTR SS:[EBP-8], 0
00469F88 74 45 JE SHORT WMV_to_A.00469FCF
00469F8A 8D55 F0 LEA EDI, DWORD PTR SS:[EBP-10]
00469F8D 52 PUSH EDI
00469F8E 6A 04 PUSH 4
00469F90 6A 04 PUSH 4
00469F92 8B45 08 MOV EAX, DWORD PTR SS:[EBP+8]
00469F95 50 PUSH EAX
00469F96 FF15 2C574700 CALL DWORD PTR DS:[47572C]
00469F9C 85C0 TEST EAX, EAX
00469F9E 75 0A JNZ SHORT WMV_to_A.00469FAA
  
```

So here we will set a BP on HW execute commands in this dance JE.



Summarized again at this time we know the OEP of the program and know the magic jump. Next we need to know and get the file that MoleBox created as above. Press **Ctrl + F2** to restart the Olly, an **CreateFileA BP**.



So we have 3 BP 1 BP in Call EAX, BP and 1 at CreateFileA execute on HW jump in orders. Now we press F9 to run the program, we will Break 1 times here:

Address	Value	Comment
0013FC60	00463C32	CALL to CreateFileA from WMV_to_A.00463C2C
0013FC64	00CA1F78	FileName = "C:\Program Files\WMV to AVI MPEG DVD WMV Converter\WMV to AVI MPEG DVD WMV Converter
0013FC68	80000000	Access = GENERIC_READ
0013FC6C	00000001	ShareMode = FILE_SHARE_READ
0013FC70	00000000	pSecurity = NULL
0013FC74	00000003	Mode = OPEN_EXISTING
0013FC78	00000000	Attributes = 0
0013FC7C	00000000	hTemplateFile = NULL

F9, the Break times 2:

Address	Value	Comment
0013F97C	0046841D	CALL to CreateFileA from WMV_to_A.00468417
0013F980	00CA20A8	FileName = "C:\PROGRAM FILES\WMV TO AVI MPEG DVD WMV CONVERTER\WMV TO AVI MPEG DVD WMV CONVERTER
0013F984	80000000	Access = GENERIC_READ
0013F988	00000001	ShareMode = FILE_SHARE_READ
0013F98C	00000000	pSecurity = NULL
0013F990	00000003	Mode = OPEN_EXISTING
0013F994	00000000	Attributes = 0
0013F998	00000000	hTemplateFile = NULL

F9, the Break 3 times but this is the magic jump at Break:

Address	Hex dump	Disassembly	Comment
00469F88	74 45	JE SHORT WMV_to_A.00469FCF	
00469F8A	8D55 F0	LEA EDX, DWORD PTR SS:[EBP-10]	
00469F8D	52	PUSH EDX	
00469F8E	6A 04	PUSH 4	

Press **Space Bar** and Jmp to change, and put BP on HW to execute here. Then click to **F9**, we pass through by the IAT canceled until we stop here:

Address	Value	Comment
0013FC48	0046B63C	CALL to CreateFileA from WMV_to_A.0046B636
0013FC4C	00D217E0	FileName = "C:\DOCUME~1\KIEN TRAN\LOCALS~1\TEMP\MBX@FA8@CA2480.###"
0013FC50	40000000	Access = GENERIC_WRITE
0013FC54	00000000	ShareMode = 0
0013FC58	00000000	pSecurity = NULL
0013FC5C	00000002	Mode = CREATE_ALWAYS
0013FC60	00000000	Attributes = 0
0013FC64	00000000	hTemplateFile = NULL

Wow, look, we see to the right place and that. How to know the name of the chain quái qui. Hihi very often simply because it will end up in the Stack of Stack window you look down hairbreadth is found its name:

Address	Value	Comment
0013FC70	7FFDF000	
0013FC74	0013FD44	
0013FC78	7C90EE18	ntdll.7C90EE18
0013FC7C	7C926AC8	ntdll.7C926AC8
0013FC80	FFFFFFFF	
0013FC84	7C926ABE	RETURN to ntdll.7C926ABE from ntdll.7C90EE02
0013FC88	7C9268AD	RETURN to ntdll.7C9268AD from ntdll.7C9268B7
0013FC8C	00000000	
0013FC90	0042BF8E	ASCII "aveData.dll"
0013FC94	7C91056D	RETURN to ntdll.7C91056D from ntdll.7C90EE02

So we will correct the name correctly, Follow in Address dump in the image above ('s name contains a banana loang ngoang) and Edit again we will be as follows:

Address	Value	Comment
0013FC48	0046B63C	CALL to CreateFileA from WMV_to_A.0046B636
0013FC4C	00D217E0	FileName = "C:\DOCUME~1\KIEN TRAN\LOCALS~1\TEMP\aveData.dll"
0013FC50	40000000	Access = GENERIC_WRITE
0013FC54	00000000	ShareMode = 0
0013FC58	00000000	pSecurity = NULL
0013FC5C	00000002	Mode = CREATE_ALWAYS
0013FC60	00000000	Attributes = 0
0013FC64	00000000	hTemplateFile = NULL

Now you leave BP CreateFileA on, press **Alt + O**, in the **Events** tab select **Break on New modules (dll)**. Back to Olly we press F9, we will module.Lai Break in new press **Alt + O** and select **Break on new modules**. Then press Alt + M to open the window memory, where you pull down the search area by Avedata as in the picture below:

00E60000	00001000	aveData		PE header	Imag	R	RWE
00E61000	0000B000	aveData	.text	code	Imag	R	RWE
00E6C000	00002000	aveData	.rdata	data, exports	Imag	R	RWE
00E6E000	00005000	aveData	.data		Imag	R	RWE
00E73000	00002000	aveData	.reloc		Imag	R	RWE
00E75000	00001000	aveData	__BOX__	SFX, relocat	Imag	R	RWE

Select section. Text and set a BP on access by pressing **F2**. Next press **F9** to run the program, will Olly ice here:

Address	Hex dump	Disassembly	Comment
0046608E	F3:A5	REP MOVSDWORD PTR ES:[EDI],DWORD PTR DS:[ESI]	
00466090	8BC8	MOV ECX,EAX	
00466092	83E1 03	AND ECX,3	
00466095	F3:A4	REP MOVSBYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	
00466097	B8 00000100	MOV EAX,10000	UNICODE "=:::~:\\"
0046609C	2B45 F8	SUB EAX,DWORD PTR SS:[EBP-8]	
0046609F	3945 10	CMP DWORD PTR SS:[EBP+10],EAX	
004660A2	73 08	JNB SHORT WMV_to_A.004660AC	

Pull down we found the command: 00466172 C2 0C00 RETN 0C

Set in BP and press **F9**, we will break there. Un BP and press F8 to trace out, we will come:

Address	Hex dump	Disassembly	Com.
004662E6	8945 DC	MOV DWORD PTR SS:[EBP-24],EAX	
004662E9	837D DC 00	CMP DWORD PTR SS:[EBP-24],0	
004662ED	74 08	JE SHORT WMV_to_A.004662F7	
004662EF	8B45 14	MOV EAX,DWORD PTR SS:[EBP+14]	
	8B4D D8	ECX,DWORD PTR SS:[EBP-20]	

The press **Alt + M** and clay at the BP **section**. AveData of **text**. Press **F9** to run, to break in here:

Address	Hex dump	Disassembly	Comment
0046A5D1	8B00	MOV EAX,DWORD PTR DS:[EAX]	sockspy.1000AEC0
0046A5D3	03C1	ADD EAX,ECX	
0046A5D5	8945 E8	MOV DWORD PTR SS:[EBP-18],EAX	
0046A5D8	8B45 E0	MOV EAX,DWORD PTR DS:[EBP-20]	

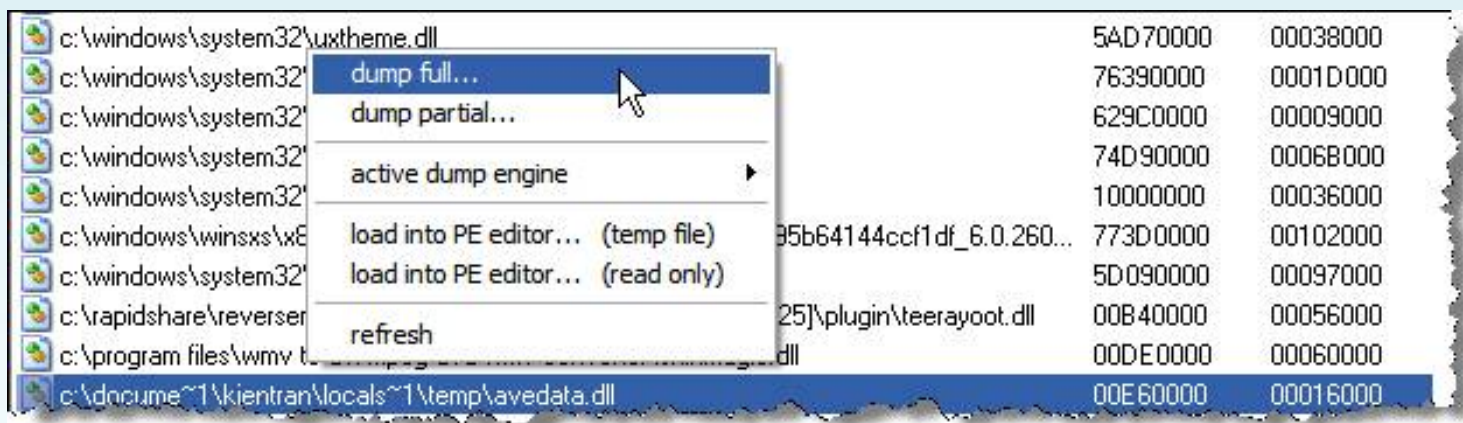
Looking to find command: 0046A5E8 C3 RETN. Set a BP here, press **F9** and Olly will break. Un-BP and press **F8** to trace out here we will:

Address	Hex dump	Disassembly
0046A741	83C4 10	ADD ESP,10
0046A744	8B45 DC	MOV EAX,DWORD PTR SS:[EBP-24]
0046A747	0345 CC	ADD EAX,DWORD PTR SS:[EBP-34]
0046A74A	8945 D4	MOV DWORD PTR SS:[EBP-2C],EAX
0046A74D	6A 5C	PUSH 5C

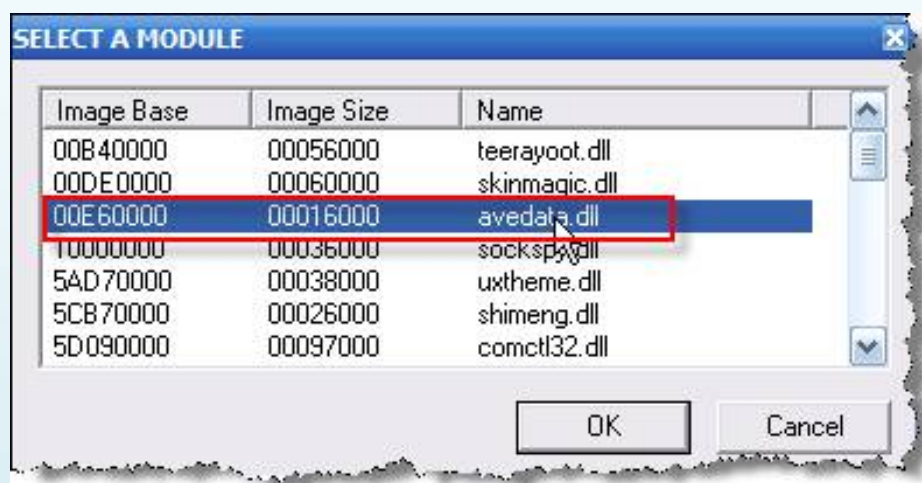
Once again set in the BP **section**. AveData **text** of press **F9**, and we will stop at the OEP aveData.dll.

Address	Hex dump	Disassembly	
00E643EC	55	PUSH EBP	OEP
00E643ED	8BEC	MOV EBP,ESP	
00E643EF	53	PUSH EBX	
00E643F0	8B5D 08	MOV EBX,DWORD PTR SS:[EBP+8]	
00E643F3	56	PUSH ESI	
00E643F4	8B75 0C	MOV ESI,DWORD PTR SS:[EBP+C]	
00E643F7	57	PUSH EDI	
00E643F8	8B7D 10	MOV EDI,DWORD PTR SS:[EBP+10]	
00E643FB	85F6	TEST ESI,ESI	

Then to the attractions here and now we will dump this dll file. Here I used to dump LordPE file. Open LordPE the process and select the modules and find aveData.dll, click to select full dump. Then save your name with a **aveData_dumped.dll**:



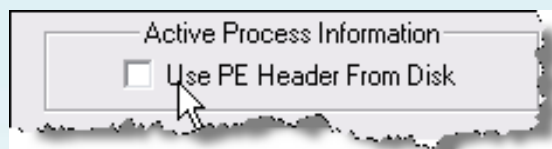
Now conducted Fix IAT for this dll file. ImportRec open up, select the Process then click Pick dll we are as follows:



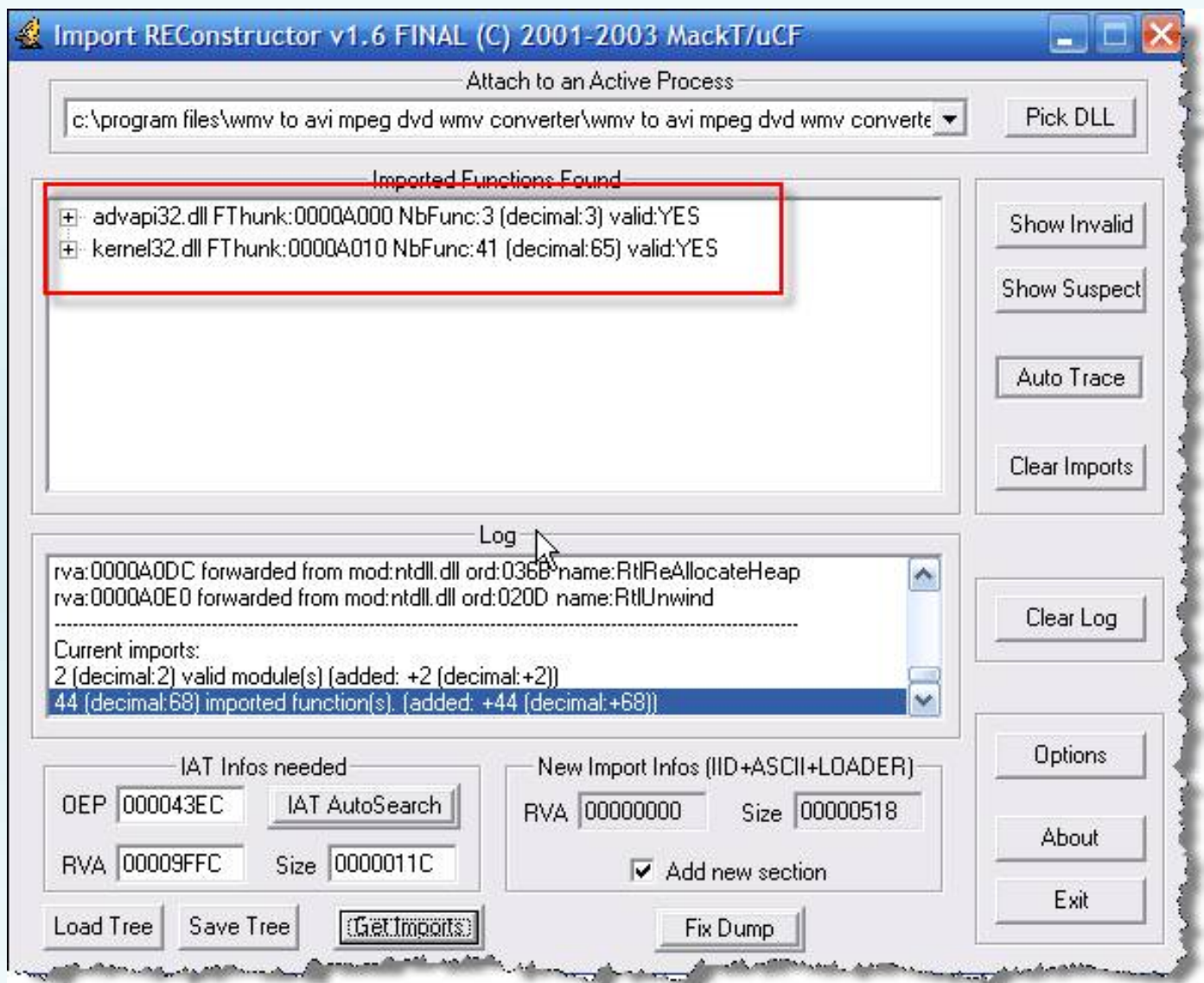
Click OK to select, we will see ImageBase and size is wrong:

```
>> Module selected: c:\docume~1\kientran\locals~1\temp\avedata.dll
Image Base:10000000 Size:00016000
```

Click on Options in ImportRec and select:



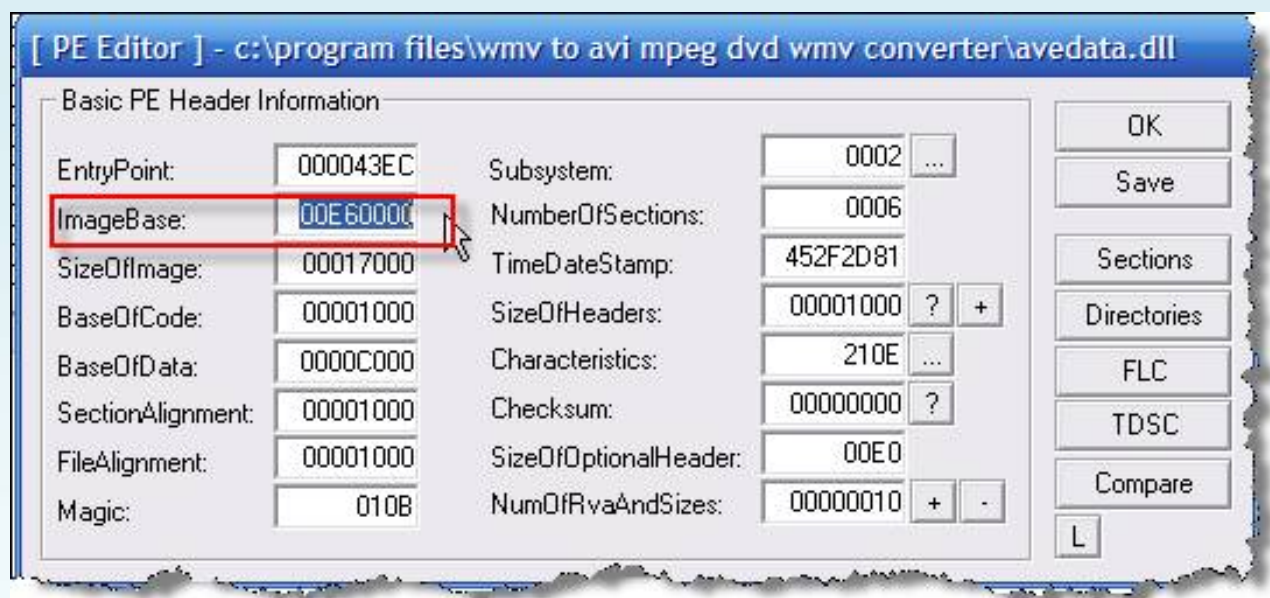
VBy OEP has filled hours we will fix ImageBase later. Hihi in previous pictures we see aveData start at 0x00E60000 (ImageBase), OEP that we find in the Olly 0x00E643EC so here OEP's file on the disk is = **0x00E643EC - 0x00E60000 = 0x43EC**. Fill in the OEP to ImportRec, click Search and IAT Get Auto Imports we have been as follows:



Conduct Fix dump dll files for which we have full dump with LordPE. ImportRec Close to rename the file aveData.dll. The next step we will conduct Fix ImageBase and Reloc of this dll file. Learn more **about**. **Reloc** section in PE lessons and tutorials written by Trickyboy. First is ImageBase, we will see one in the dump file ImageBase any, in Olly press Alt + M we have been as follows:

00E60000	00001000	aveData		PE header	Imag	R	RWE
00E61000	0000B000	aveData	.text	code	Imag	R	RWE
00E6C000	00002000	aveData	.rdata	data, exports	Imag	R	RWE
00E6E000	00005000	aveData	.data		Imag	R	RWE
00E73000	00002000	aveData	.reloc		Imag	R	RWE
00E75000	00001000	aveData	_BOX_	SFX relocat	Imag	R	RWE

Address PE Header is ImageBase, so we have a ImageBase **0x00E60000**. Open selected LordPE PE Editor and select aveData.dll then revised ImageBase:



Click Save to save them ImageBase. Now **to. Reloc**, click Sections we have the following information:

[Section Table]					
Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	0000A46E	00001000	0000A46E	E0000080
.rdata	0000C000	000014AC	0000C000	000014AC	E0000080
.data	0000E000	000049A4	0000E000	000049A4	E0000080
.reloc	00013000	000013E0	00013000	000013E0	E0000080
._BOX_	00015000	00001000	00015000	00001000	E0000020
.mact	00016000	00001000	00016000	00001000	E0000060

Back to Olly we see section. Reloc start at 0x00E73000. Get this value will be less ImageBase us Raw Offset: **0x00E73000 - 0x00E60000 = 0x00013000**. That is exactly Roffset not need to correct, is probably to fix the Rsize but here I plunge after it đấy. Back to LordPE click Directories Relocation and revised as below:

[Directory Table]					
Directory Information					
	RVA	Size			
ExportTable:	0000D310	0000019C	...	L	H
ImportTable:	00016000	00000028	...	L	H
Resource:	00000000	00000000	...	L	H
Exception:	00000000	00000000		L	H
Security:	00000000	00000000			H
Relocation:	00013000	000013E0	...	L	H
Debug:	00000000	00000000	...	L	H
Copyright:	00000000	00000000		L	H

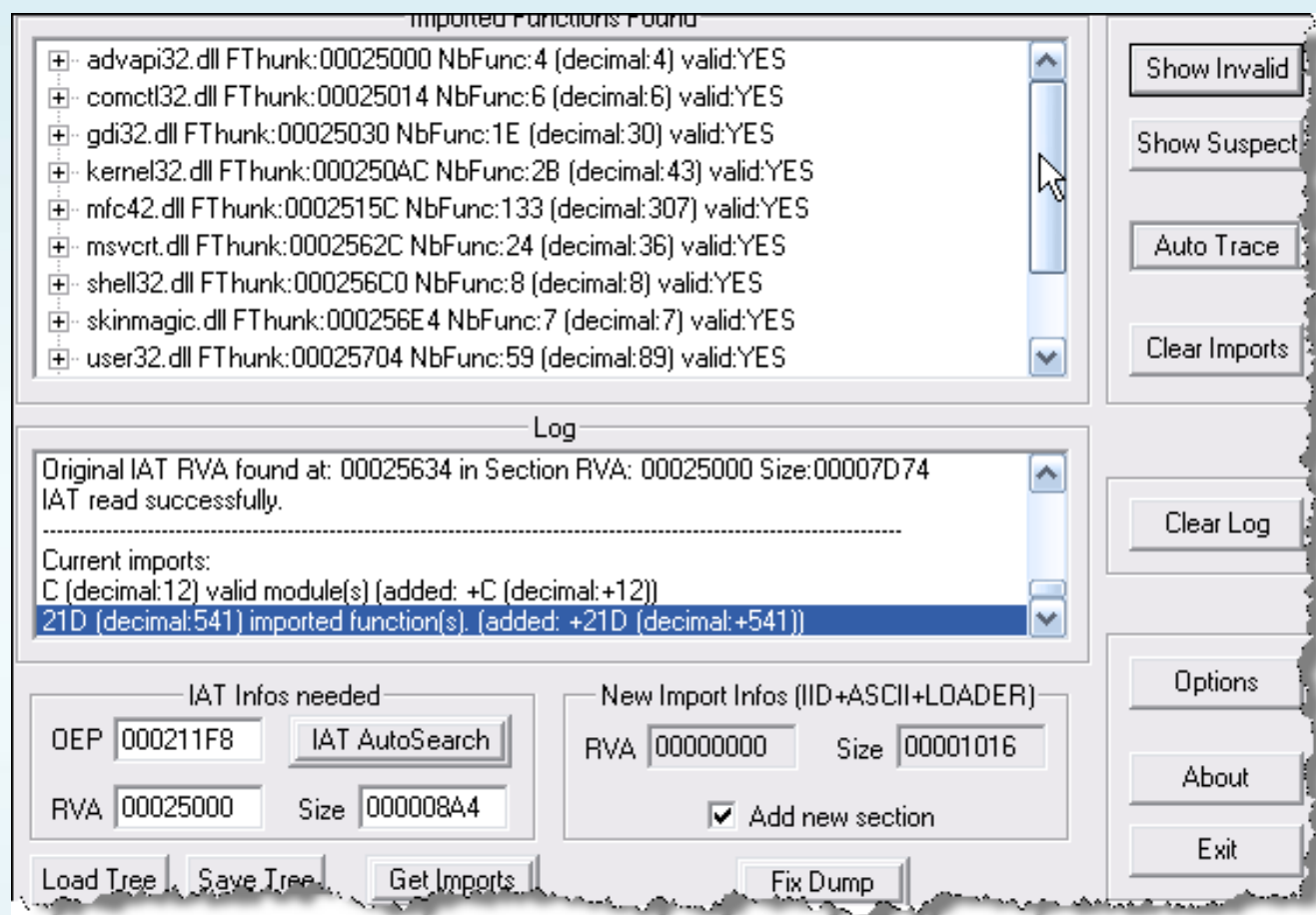
Click Save to save. Then it is finished and part or most interesting. Next is to dump the main file. Now on to review and remove all of the BP go only to the per-BP in order Call EAX only. Then press **Shift + F9** a play, we will stop me in this order:

Address	Hex dump	Disassembly	Comment
00462793	FFD0	CALL EAX	WMV_to_A.004211F8
00462735	E8 A6C00000	CALL WMV_to_A.0046E7ED	

Un-BP go, press **F7** to Trace Call in order we will stop OEP's main modules:

Address	Hex dump	Disassembly	Comment
004211F8	55	PUSH EBP	
004211F9	88EC	MOV EBP, ESP	
004211FB	6A FF	PUSH -1	
004211FD	68 00854200	PUSH WMV_to_A.00428500	
00421202	68 84134200	PUSH WMV_to_A.00421384	
00421207	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	JMP to msvcrt._except_handler3
00421209	50	PUSH EAX	
0042120E	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
00421215	83EC 68	SUB ESP, 68	

Conduct Fix IAT dump and only the owners do more chẵn ka ka J , a forgotten memory you choose to leave the Option has been selected in ImportRec time we fix dll file nhé. If done properly will be as follows:



No one is invalid, Fix dump and run the test concerns not any more tolerable J . Wow file dumped_.exe we've run too mượt keke.Neu your run it is please do from the beginning nhé!

IV. Serial Fishing

Next we'll take care of the crack of this program. Okie dumped_.exe load the file in Olly, press F9 to run the program and enter the username and the serial, click Register immediately receive the following:



Search String in Olly we see on the series here:

004154FE	PUSH	dumped_.0042E788	ASCII "%02d%02d%02d"
00415996	PUSH	dumped_.0042E82C	ASCII "Sorry"
0041599B	PUSH	dumped_.0042E804	ASCII "Invalid License Name or License Code"
004159EB	PUSH	dumped_.0042E7DC	ASCII "Register successful! License to:%s."
00415A03	PUSH	dumped_.0042E7D0	ASCII "Thank you"
00415A7C	PUSH	dumped_.0042E7C4	ASCII "data.ini"
00415ABF	PUSH	dumped_.0042E7B4	ASCII "License Name"
00415AC4	PUSH	dumped_.0042E7A8	ASCII "Register"

Follow this address we will be in here in Olly:

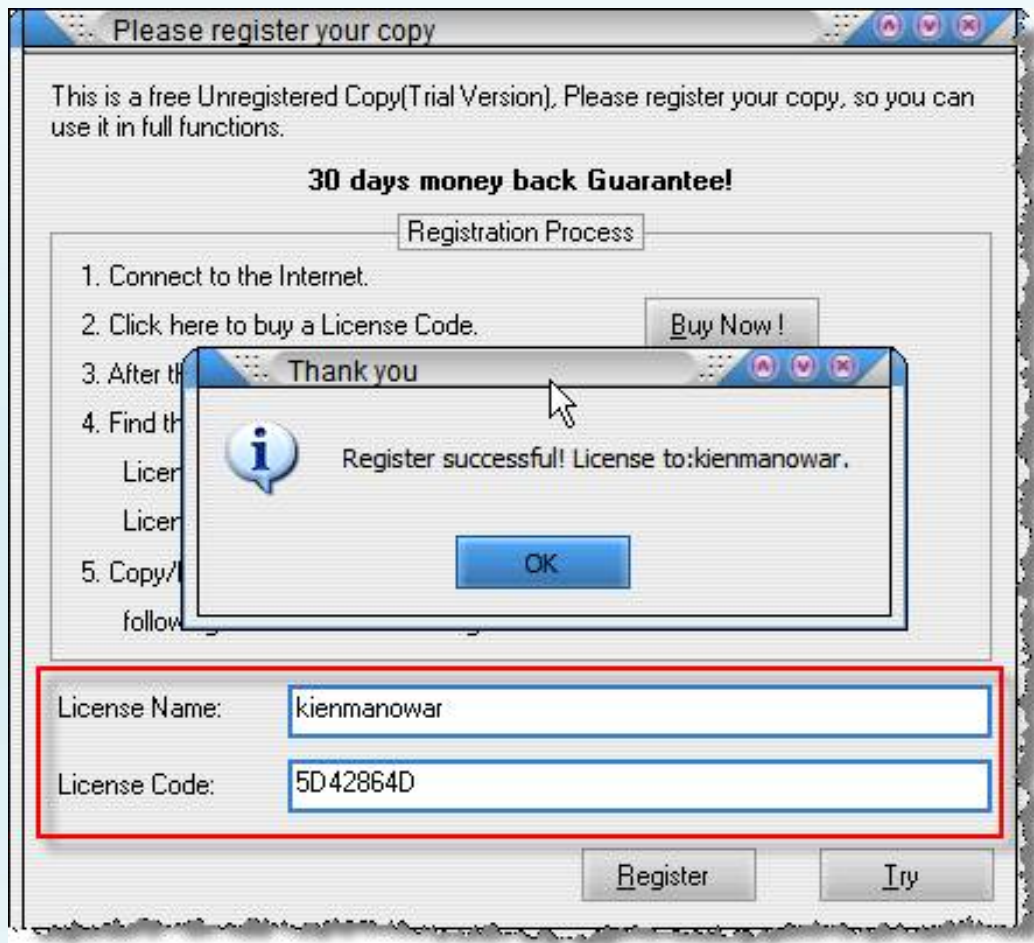
Address	Hex dump	Disassembly	Comment
00415984	. 8B0F	MOV ECX, DWORD PTR DS:[EDI]	
00415986	. 50	PUSH EAX	
00415987	. 51	PUSH ECX	
00415988	. E8 9B800000	CALL <JMP.&avedata.ge_check>	
0041598D	. 83C4 08	ADD ESP, 8	
00415990	. 85C0	TEST EAX, EAX	
00415992	. 75 2B	JNZ SHORT dumped_.0041598F	
00415994	. 6A 40	PUSH 40	
00415996	. 68 2CE84200	PUSH dumped_.0042E82C	ASCII "Sorry"
00415998	. 68 04E84200	PUSH dumped_.0042E804	ASCII "Invalid License Name or Li
004159A0	. 8BCD	MOV ECX, EBP	
004159A2	. E8 4B860000	CALL <JMP.&F42.#4224>	

Dich up some of you set a BP at **0x00415988** same as in Figure tren. Sau the press F9 to run, enter UserName Fake Fake Code and then click Register, we will break in the jaw in Olly. Press F7 to Trace in order to Call this, if you're interested in keygen then analyze kĩ the order in which my goal is just Fishing should not need my attention. After a trace I also need to be given to:

00E61E5E	6A 08	PUSH 8
00E61E60	52	PUSH EDX
00E61E61	50	PUSH EAX
00E61E62	E8 09240000	CALL avedata.00E64270
00E61E67	83C4 18	ADD ESP, 18
00E61E6A	85C0	TEST EAX, EAX
00E61E6C	5F	POP EDI
00E61E6D	5E	POP ESI
00E61E6E	5D	POP EBP
00E61E6F	5B	POP EBX
00E61E70	0F85 83000000	JNZ avedata.00E615E9

```
Registers (FPU)
EAX 00036FD8 ASCII "12345678"
ECX 0013AB5C
EDX 0013AD58 ASCII "5D42864D"
EBX 2395B0DB
ESP 0013A880
```

Wow with a user name: kienmanowar the right code is: **5D42864D**. Close Olly, run the program try to enter user name and corresponding code Bup tolerable Done! J



The program will immediately file a birth data.ini to save the information registered by the ta. Vay is finished, I ended permission message here!

Best Regards

_ [Kienmanowar] _



--+ +---==[**Greatz thanks to**]=---+ +--

My family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHOCrker, the_Lighthouse, Merc, Hoadongnoi, Nini ... all REA's members, TQN, HacNho, RongChauA, Deux, tlandn, light.phoenix, dqtlN, ARTEAM all my friend, and you.

Thanks to --+ +---==[]=---+ +--

iamidiot, WhyNotBar, trickyboy, dzungltvn, takada, hurt_heart, haule_nth, v. hytkl. v.. You have contributed greatly to the REA. Hope you will continue to promote J

I want to thank **Teddy Roggers** for his great site, Reversing.be folks (especially **haggar**), Arteam folks (**Shub-Nigurath**, **MaDMan_H3rCuL3s**) and all folks on crackmes.de, thank to all members of **unpack.cn** (especially **fly** and **linhanshi**). Great thanks to **lena151** (I like your tutorials). And finally, thanks to **Ricardo NARVAJA** and all members on **CRACKSLATINOS**.

>>>> If you have any suggestions, comments or corrections email me: **kienmanowar [at] reaonline.net**

KaGra Tutorials

Translated and written by: **kienmanowar**

Manual unpacking Morphine 1.4 - 2.7

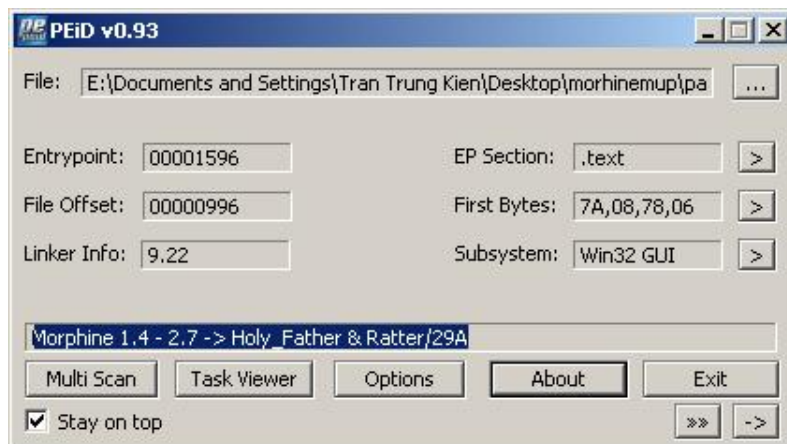
Information	Unpacking for Newbie's
Target	Target.exe
Available	http://www.reaonline.net
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, PEid 0.93, Lord PE 1.4, Plugin Command Line, ImpRec v1.6f.
Protection	Morphine 1.4 - 2.7
L Evel	Beginner
Category	Manual unpacking
Author	KaGra (February 06 2005)

1. Introduction

This article will guide you unpack **Morphine 1.4 - 2.7**.

PE 2.Detect and get info

PEid use to detect, we know the following:



Use **Lord PE 1.4** to search for more information:

access> DWORD, press **Ctrl + F9** (can you get a message like this: **"Bad or unknown format ... "forget it and just click OK final press F7 we will stop here:**

Address	Hex dump	Disassembly	Comment
004010E9	5E	POP ESI	USER32.77D42276
004010EA	5D	POP EBP	
004010EB	83C4 04	ADD ESP,4	
004010EE	5B	POP EBX	
004010EF	5A	POP EDX	
004010F0	83C4 08	ADD ESP,8	
004010F3	894C24 04	MOV DWORD PTR SS:[ESP+4],ECX	
004010F7	FFE0	JMP EAX	

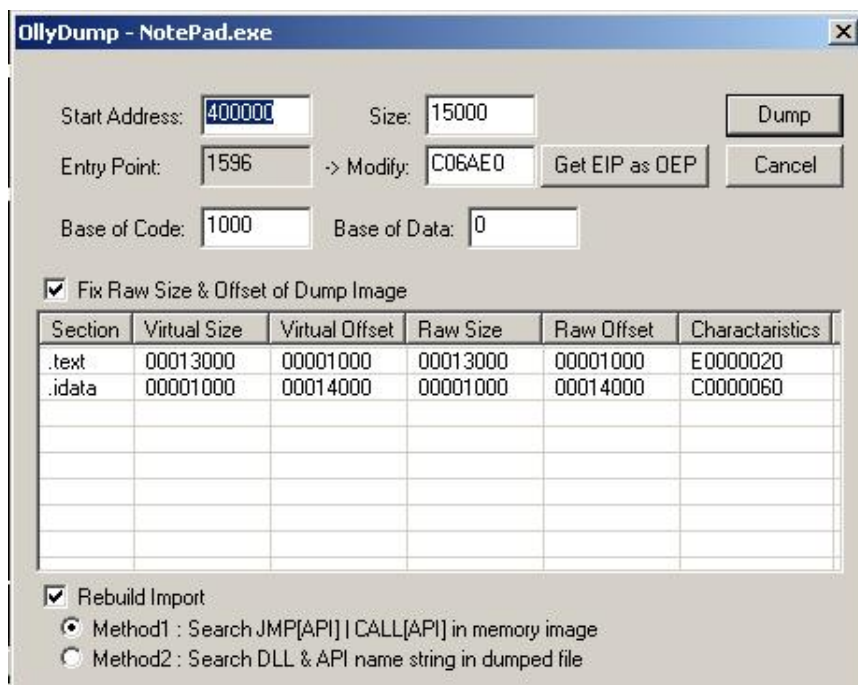
Oki, at a command **004010F7** jump to OEP. Now we delete Hardware breakpoint by the **Debug-> Breakpoints Hardware-> Delete** to delete. Then we perform lenh jump at **004010F7** and we will stay in **OEP**:

Address	Hex dump	Disassembly	Comment
01006AE0	6A 70	PUSH 70	
01006AE2	68 88180001	PUSH packed_1.01001888	
01006AE7	E8 BC010000	CALL packed_1.01006CA8	
01006AEC	33DB	XOR EBX,EBX	
01006AEE	53	PUSH EBX	
01006AEF	8B3D 4C110001	MOV EDI,DWORD PTR DS:[100114C]	kernel32.GetModuleHandleA
01006AF5	FFD7	CALL EDI	
01006AF7	66:8138 4D5A	CMP WORD PTR DS:[EAX],5A4D	

OEP so that we are **01006AE0**.

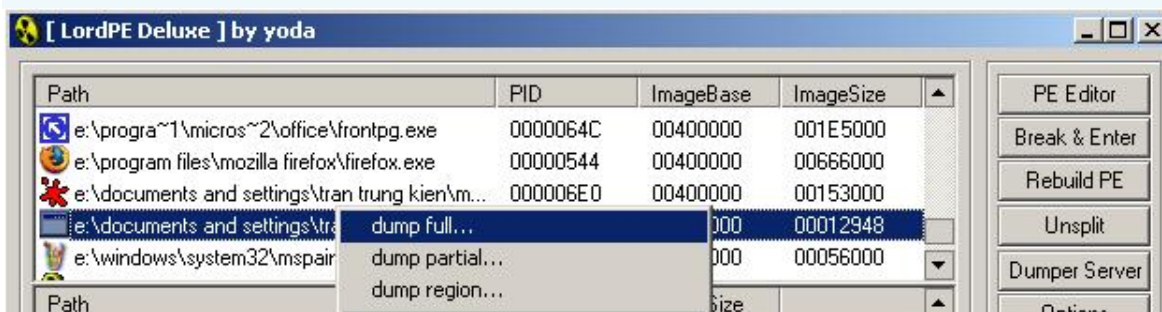
4. Unpacked dumping our files

At **01006AE0**, we try to dump in the Plugin with Olly. Click to select and **dump debugged process**. We will see the following:



Start address: 00400000 Size: 15000
Entry Point: 1596 Modify: C06AE0

Now we will change the value in the Modify the value of our OEP (01006AE0 = 01000000-6AE0) and value in the Start Address of 01,000,000. **Rebuilt** select **Import** and click Dump.Ac we can not dump the no.Vay we will try to dump it in LordPE see why. LordPE open up, select the program that we want to dump, **dump** your mouse to select **full**:



Oki, the dump file to complete dumped.exe we try to run it. The crash is now (it does not run and it is not because of the IAT (well, this packer does not do anything with IAT, like UPX)). Why is this happening again?

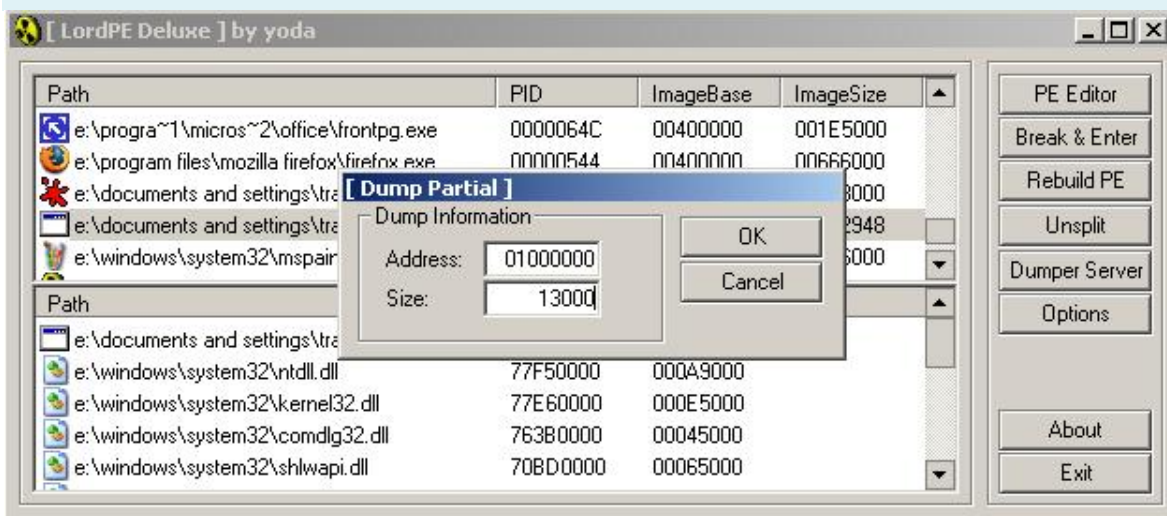
At Olly screen, we press **Alt + M** to open the window Memory map, we will see the following:

Address	Size	Owner	Section	Contains	Type	Access	Initial
00010000	00001000	00010000 (itself)			Priv 00021004	RW	RW
00020000	00001000	00020000 (itself)			Priv 00021004	RW	RW
0011F000	00001000	00030000			Priv 00021104	RW Guarded	RW
00120000	00010000	00030000			Priv 00021104	RW Guarded	RW
00130000	00001000	00030000 (itself)		stack of ma	Map 00041002	R	R
00140000	00010000	00140000 (itself)			Priv 00021004	RW	RW
00240000	00006000	00240000 (itself)			Priv 00021004	RW	RW
00250000	00001000	00250000 (itself)			Map 00041004	RW	RW
00260000	00016000	00260000 (itself)			Map 00041002	R	R
00280000	00034000	00280000 (itself)			Map 00041002	R	R
002C0000	00041000	002C0000 (itself)			Map 00041002	R	R
00310000	00006000	00310000 (itself)			Map 00041002	R	R
00320000	00004000	00320000 (itself)			Priv 00021004	RW	RW
00330000	00003000	00330000 (itself)			Map 00041002	R	R
00340000	00001000	00340000 (itself)			Priv 00021004	RW	RW
00350000	00001000	00350000 (itself)			Priv 00021004	RW	RW
00360000	00002000	00360000 (itself)			Map 00041002	R	R
00370000	00003000	00370000 (itself)			Priv 00021004	RW	RW
00380000	00002000	00380000 (itself)			Map 00041002	R	R
00400000	00015000	00400000 (itself)			Imag 01001002	R	RWE
00420000	00003000	00420000 (itself)			Map 00041020	R E	R E
004E0000	00002000	004E0000 (itself)			Map 00041020	R E	R E
004F0000	00103000	004F0000 (itself)			Map 00041002	R	R
00600000	00074000	00600000 (itself)			Map 00041020	R E	R E
01000000	00013000	NotePa_1 01000000 (itself)		PE header	Priv 00021040	RWE	RWE
70BD0000	00001000	SHLWAPI 70BD0000 (itself)		PE header	Imag 01001002	R	RWE
70BD1000	0005A000	SHLWAPI 70BD0000	.text	code, import-	Imag 01001002	R	RWE
70C2B000	00002000	SHLWAPI 70BD0000	.data	data	Imag 01001002	R	RWE

Well, actually that Packer was a change the contents of memory and Olly now think that only a single Section of exe file is run, starting at Adress is 01000000 and the size is 13,000 . Hold the window Olly this, we will open a window and Olly Load up exe files (not run), press **Alt + M** to open a window similar Memory map:

Address	Size	Owner	Section	Contains	Type	Access	Initial	Map
00010000	00001000	00010000 (itself)			Priv	RW	RW	
00020000	00001000	00020000 (itself)			Priv	RW	RW	
0011F000	00001000	00030000			Priv	RW Gua	RW	
00120000	00010000	00030000			Priv	RW Gua	RW	
00130000	00001000	00130000 (itself)		stack of main thread	Map	R	R	
00140000	00010000	00140000 (itself)			Priv	RW	RW	
00240000	00006000	00240000 (itself)			Priv	RW	RW	
00250000	00001000	00250000 (itself)			Map	RW	RW	
00260000	00016000	00260000 (itself)			Map	R	R	
00280000	00034000	00280000 (itself)			Map	R	R	
002C0000	00041000	002C0000 (itself)			Map	R	R	
00310000	00006000	00310000 (itself)			Map	R	R	
00400000	00001000	NotePad 00400000 (itself)		PE header	Imag	R	RWE	
00401000	00013000	NotePad 00400000	.text	code	Imag	R	RWE	
00414000	00001000	NotePad 00400000	.idata	imports	Imag	R	RWE	
77E60000	00001000	kernel32 77E60000 (itself)		PE header	Imag	R	RWE	
77E61000	00075000	kernel32 77E60000	.text	code, imports, exports	Imag	R	RWE	
77ED6000	00003000	kernel32 77E60000	.data	data	Imag	R	RWE	
77ED9000	00066000	kernel32 77E60000	.rsrc	resources	Imag	R	RWE	

Well, we'll see Header beginning with 00400000 Size is 1000, Code Section 13000 is the Size and Size imports section of the 1000 (total of 15,000). Oki, after observing all the above, the authors believe that the only section in the first memory image may include all of the Section (PE header, code and imports), although [it is the smaller size in 2000 (13,000 = 1000 for sure the PE header and 12,000 for two otherz)]. Therefore, we will dump file by using the function partial dump. LordPE Open up, get selected Process. Click right, select **partial dump** We change the value in 2 box is: Address: 01000000, Size: 13000. Similarly as follows:



5. Editing Section in the file Dumped

After we finished dump, remains the same window LordPE, use the open file **PE Editor dumped.exe**. Then select the Sections, we are as follows:

[Section Table]					
Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	00006D72	00000400	00006E00	60000020
.data	00008000	00001BA8	00007200	00000600	C0000040
.rsrc	0000A000	00008948	00007800	00008400	40000040

Change the value in Section text by right clicking on it and choose Edit section header image ... as illustrated below:

[Edit SectionHeader]

Section Header

Name: .text

VirtualAddress: 00001000

VirtualSize: 00012000

RawOffset: 00001000

RawSize: 00012000

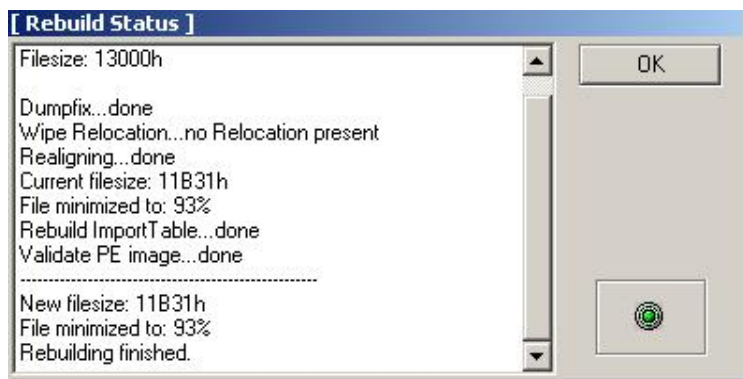
Flags: E0000020

Buttons: OK, Cancel

Click OK, then delete the rest of Section 2 only to retain the text section:

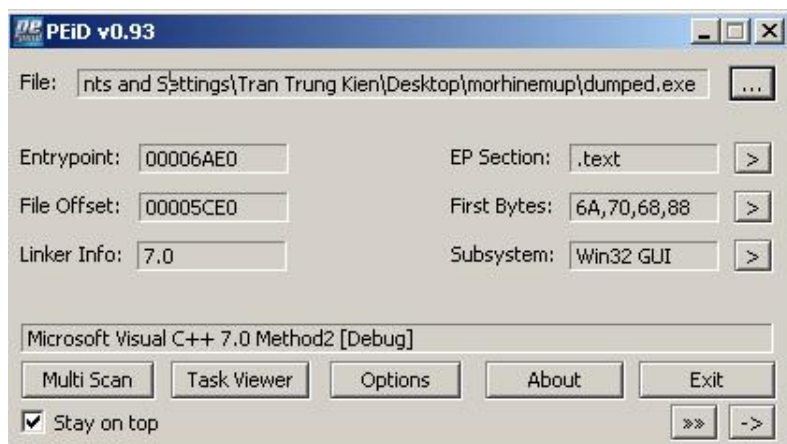
[Section Table]					
Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	00012000	00001000	00012000	E0000020

Finally, we used to LordPE Rebuilt the file **Dumped.exe**



6. Testing Our Unpacked file

Oki, a test file we Unpacked. Yup! It works.
Used to Detect PEid again we have been as follows:



So **Morphine 1.4 - 2.7** -> **Holy_Father & Ratter/29A** was successful unpack. Have fun:)

7. Conclusion

My Greetz to: tlandn (supported me this tut) and KaGra (author of this tut)
To thank my family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHQCkrker, the_Lighthouse, Hoadongnoi, Nini ... all REA's members, HacNho, RongChauA, Deux all my friend, and YOU!



Written by **kienmanowar** (tutorial date: HaNoi 14/03/2005)

...: Copyright © 2005 by kienmanowar <[=-=]> REA-cRaCkErTeAm (www.reaonline.net)::...

.: [MUP NTkrnl_Protector_0.1]:.

(_kienmanowar_)



I. Foreword

What day of the week is it, huh? Hic! During this week, I have to work hard and very busy K. Today is Sunday and I am free, so I decided to download UnpackMe (PE32bit): **UnPackMe_NTkrnl_Protector_0.1** from Teddy's site and try to manual unpacking it. My tutorial base on the idea of **bpx** from reversing.be. Ok13! L3t's R0ck w1th m3 J

II. Target and Tools

Target:

Name: **UnPackMe_NTkrnl_Protector_0.1.m.exe**

Home site: <http://www.tuts4you.com/>

Tools:

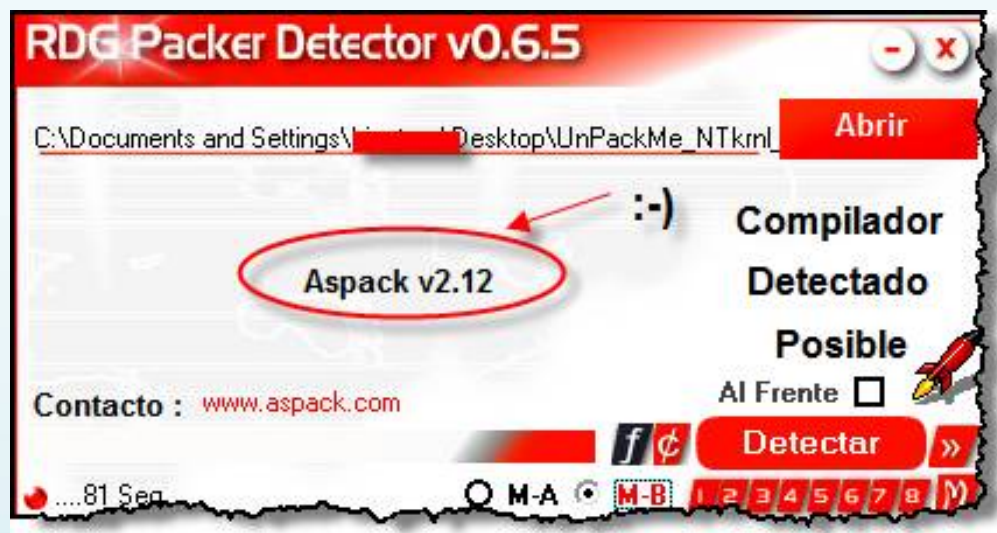
Debugger: Ollylce (16/2/07) with HideOD

PE Tools: RDG Packer Detector, PeiD, XPELister, LordPE, ImpRec (with **nt_krnlpsect_0_1.dll** plugin)

III. Manual Unpacking

_First, We try to use some famous dectector programs to find information about the target we'll work with:

_ ***RDG Packer Detector:***

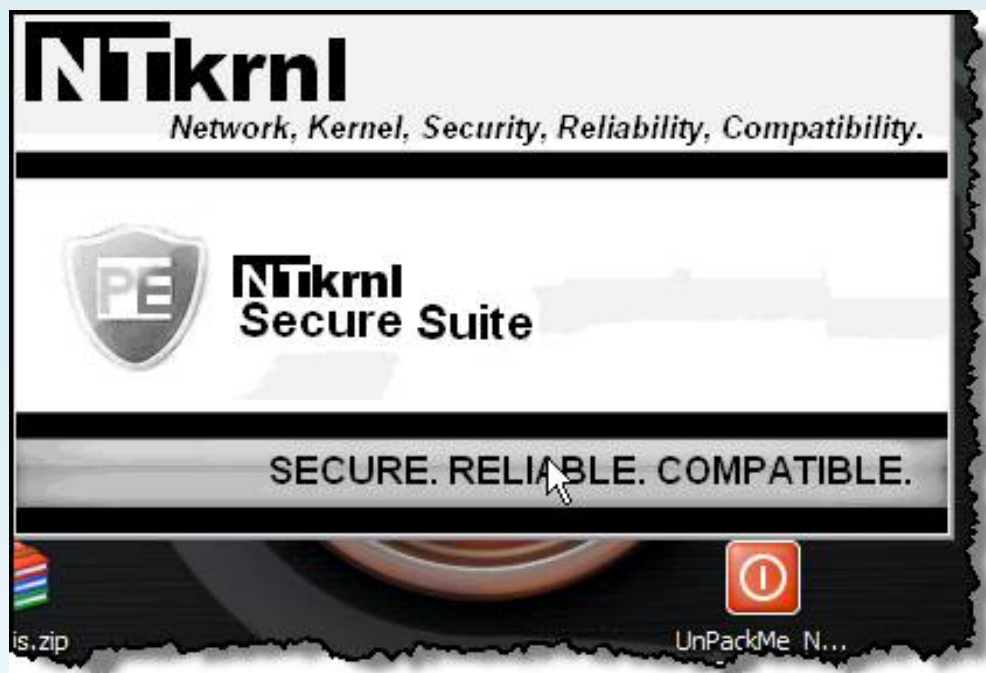


PeiD:

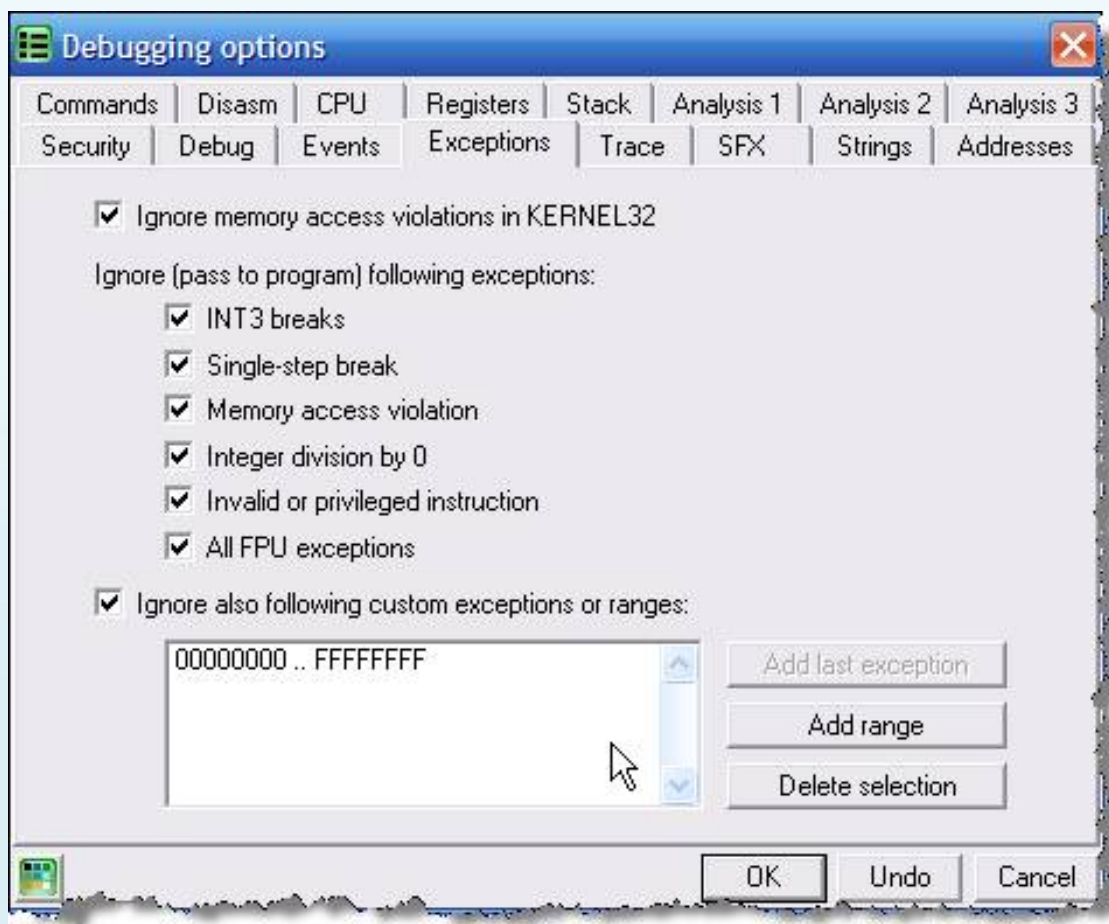


RDG detect this packer can not because of the signature database does not support it for this packer. My PeiD gives me exactly the result because it uses the signatures of J fly.

_Now We run me unpack this, it shows me the splash screen protector about this:



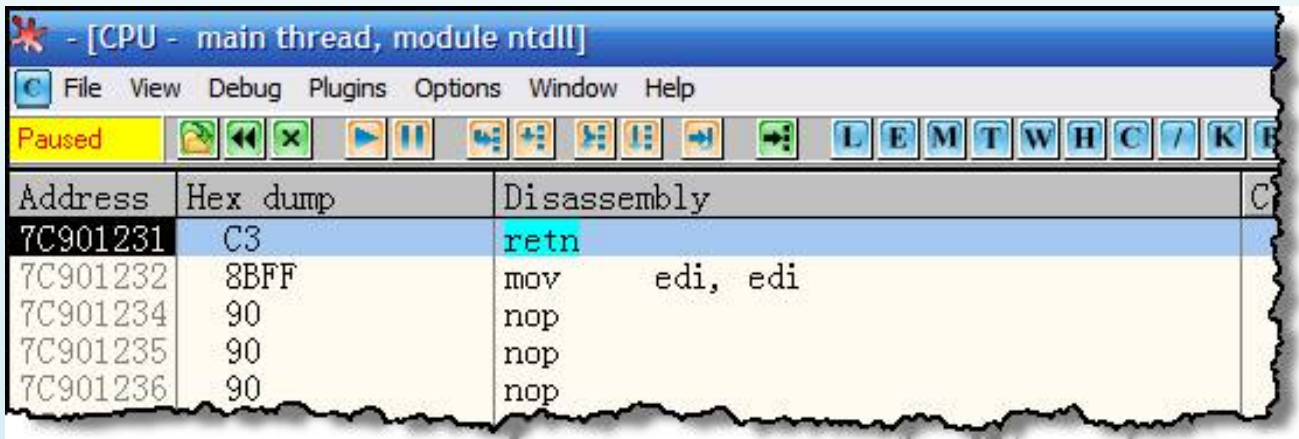
_Ok That's enough, now open Olly and configure it like that:



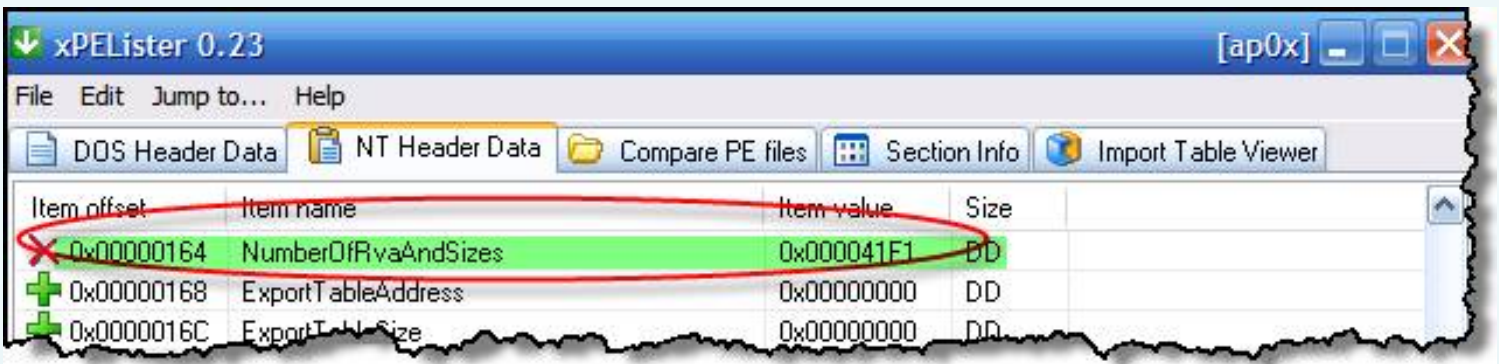
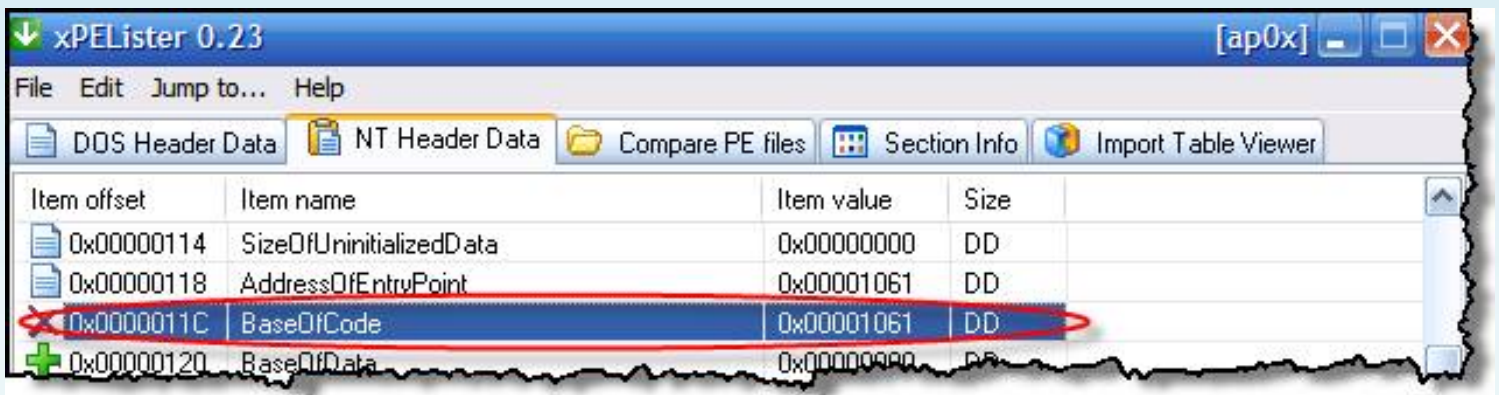
Olly _After configured like above picture, we will load into Olly target, when we get the loading errors L message like this:



_Press Ok, and we stop here in Olly. Huh, not in the main module of this target, we stopped at ntdll module:

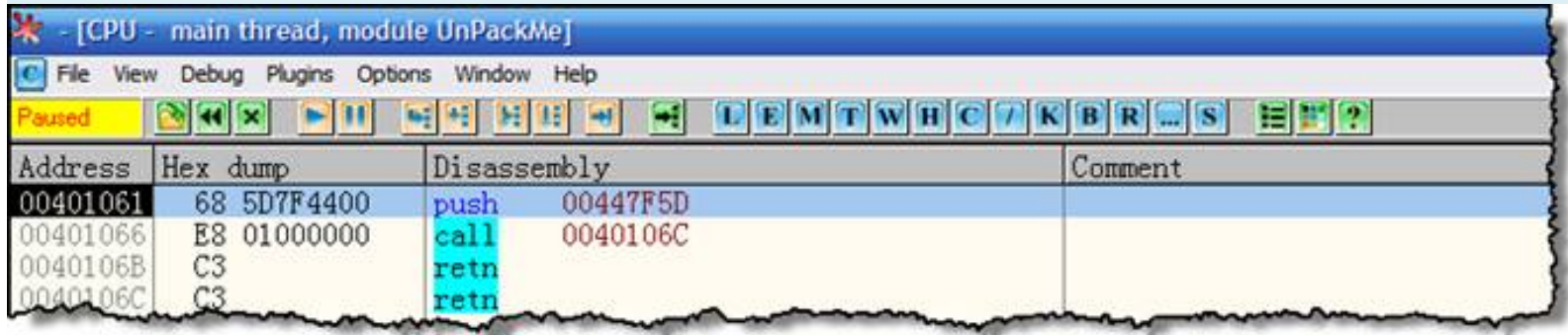


_Ok, Follow the above message, we can guess the target has some errors in PE format. Ok, let's open the **xPELister** tool to check:

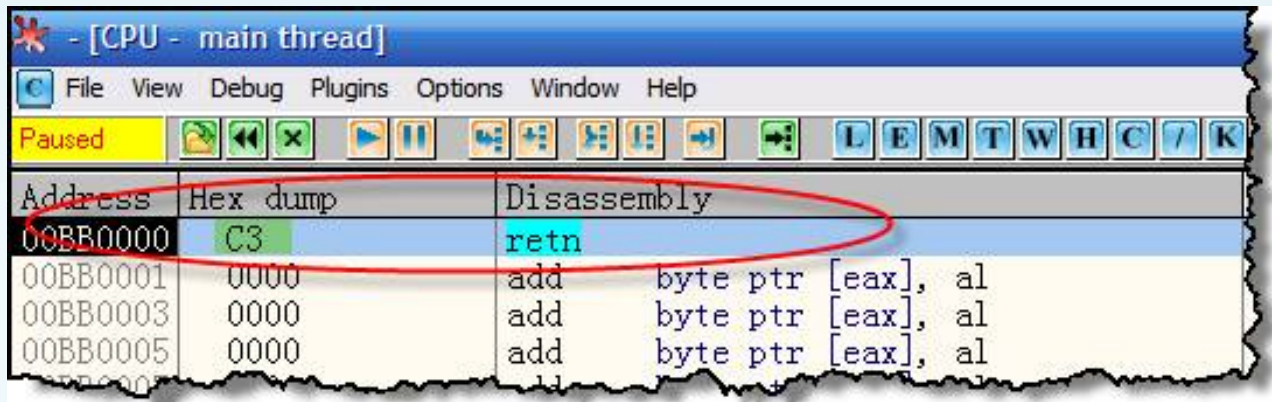


_Ok There are some wrong values, but we only care about values of two **BaseOfCode**

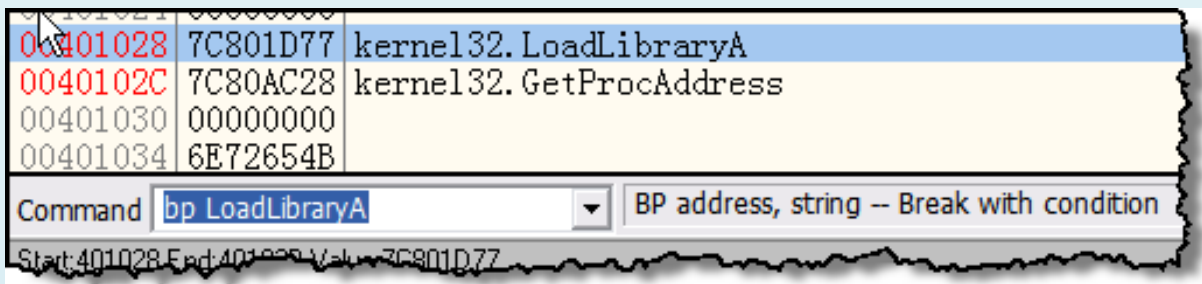
NumberOfRvaAndSizes and as you can see in the pictures above. To defeat the message when we load the target in Olly, we change the value of **BaseOfCode to: 0x1000** and **NumberOfRvaAndSizes to: 0x10**. Reload and save it in Olly, kaka we stop at the EP of this target:



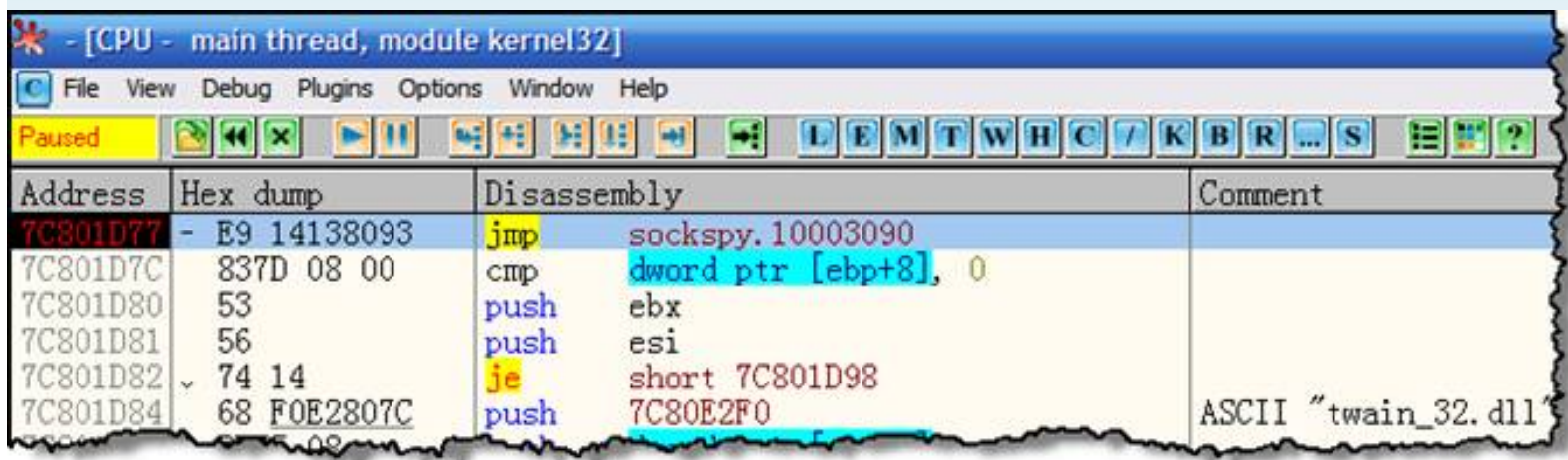
_Notice That the EP in here looks like ASPR entrypoint, compared RDG thought me unpack this is protected by ASPR J. Use HideOD plugin to hide our Ollydbg and press **F9** to run in our target environment debugged. Hic this target has an anti-debug trick so we stop here:



_To Bypass this anti-debug trick, we need to edit **from** the **C3**. Press **Ctrl + E** and edit, then set at BP **LoadLibraryA** API.

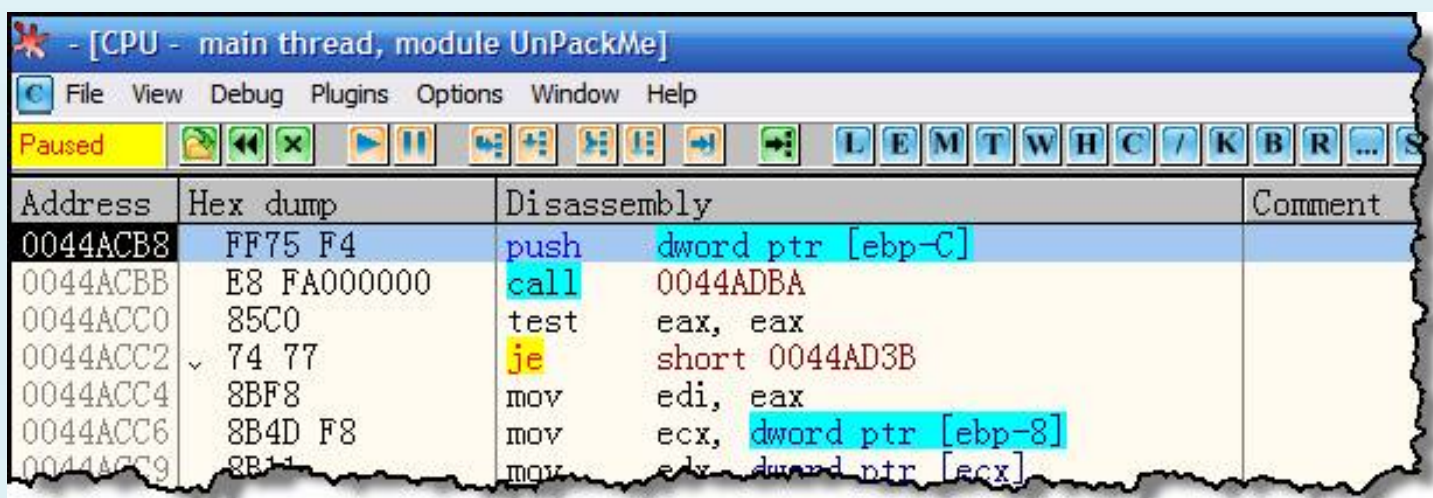


_After We set BP, press **F9** to run we'll stop at the BP that we we set. But in another module:



10000000	00001000	sockspy		PE header	Image	R	RWE
10001000	00009000	sockspy	.text	code	Image	R	RWE
1000A000	00001000	sockspy	.rdata	imports, exports	Image	R	RWE
1000B000	00029000	sockspy	.data	data	Image	R	RWE
10034000	00002000	sockspy	.reloc	relocations	Image	R	RWE

_That's Not a problem, remove and BP press **Alt + F9**. Olly will take me to here:



_We Are in the range that call to IAT (import Resolver), now we want to find the code that call to this code. Scroll down and find the **retn 4**th Set BP at this command and press **F9**, we stop here:

Address	Hex dump	Disassembly
0044AD4B	C783 84000000	mov dword ptr [ebx+84], 0
0044AD55	8B45 EC	mov eax, dword ptr [ebp-14]
0044AD58	8BE5	mov esp, ebp
0044AD5A	5D	pop ebp
0044AD5B	C2 0400	retn 4
0044AD5E	55	push ebp
0044AD5F	8BEC	mov ebp, esp
0044AD61	83C4 F0	add esp, -50

_Remove BP, and press **F8** to trace out the import Resolver. Olly will take our code to call that to the import Resolver. Next, scroll down and we'll see the **call eax** command. Ok, set the BP at this command, press **F9** to run we'll stop at Call eax. BP Remove and press **F7** to trace into this call. We are here:

* - [CPU - main thread]			
File View Debug Plugins Options Window Help			
Paused			
Address	Hex dump	Disassembly	Comment
00C5150C	6A 0C	push 0C	
00C5150E	68 C06BC400	push 0C46BC0	
00C51513	E8 742B0000	call 00C5408C	
00C51518	33C0	xor eax, eax	
00C5151A	40	inc eax	
00C5151B	8945 E4	mov dword ptr [ebp-1C], eax	
00C5151E	8B75 0C	mov esi, dword ptr [ebp+C]	
00C51521	33FF	xor edi, edi	
00C51524	3BFF	cmp edi, edi	

_Now, Press **Ctrl + B** to find (61 FF E0: popad (61), FF E0 (jmp eax)).

Enter binary string to search for

ASCII: aya

UNICODE: ?

HEX +00: 61 FF E0

☒ Entire block

☐ Case sensitive

<< >>

OK Cancel

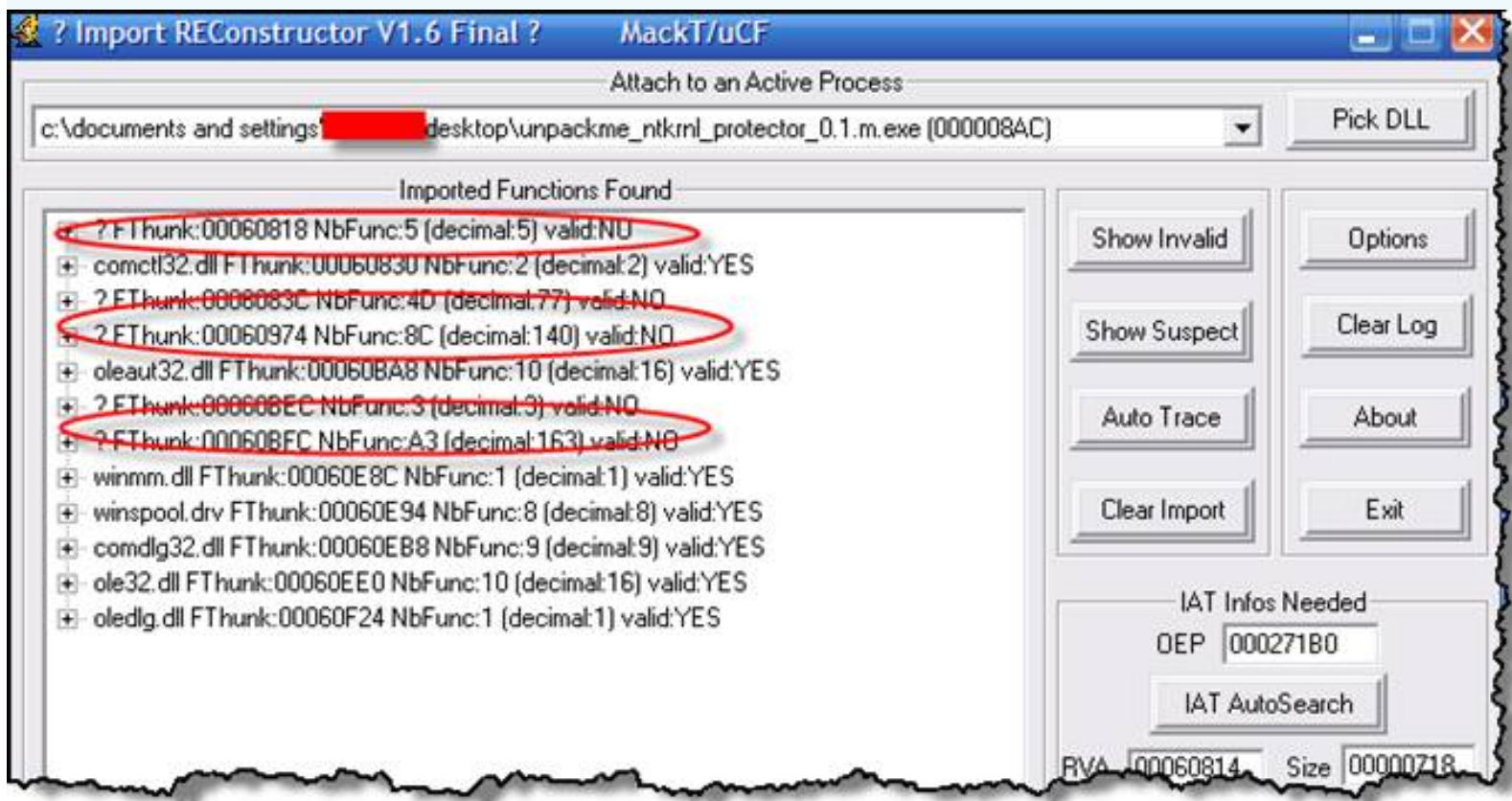
_Skip The first result, press **Ctrl + L** to Find Next and the second result is the final result that we need to find the OEP of this target. Set the BP at the **jmp eax** command.

Address	Hex dump	Disassembly	C
00C4AFB6	5F	pop edi	
00C4AFB7	5E	pop esi	
00C4AFB8	5B	pop ebx	
00C4AFB9	C9	leave	
00C4AFBA	83C4 3C	add esp, 3C	
00C4AFBD	A1 288FC500	mov eax, dword ptr [C58F28]	
00C4AFC2	894424 1C	mov dword ptr [esp+1C], eax	
00C4AFC6	61	popad	
00C4AFC7	FFE0	jmp eax	
00C4AFC9	5F	pop edi	
00C4AFCA	5E	pop esi	

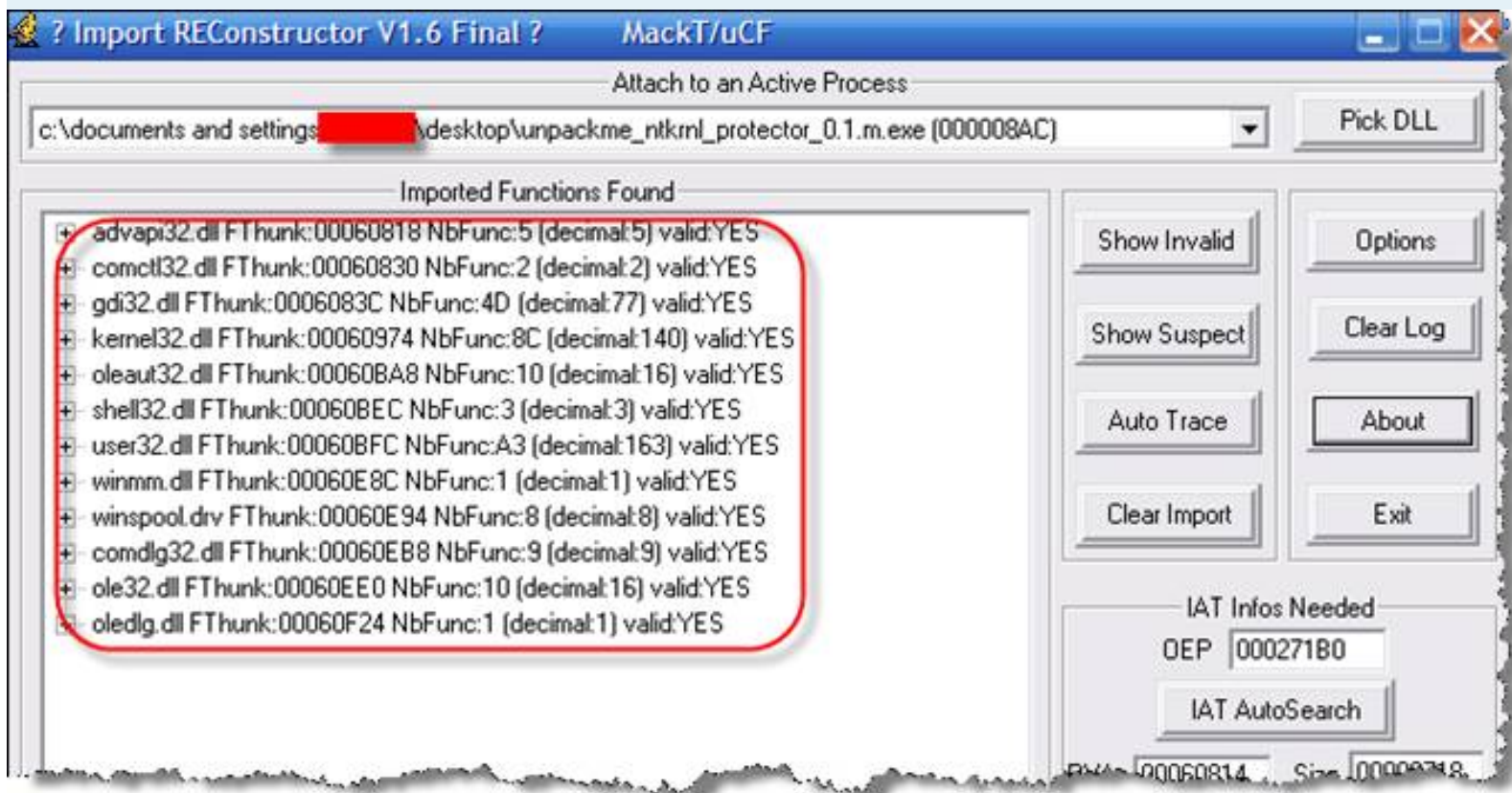
_Press F9 to run the splash screen appears and we stop at the BP. Now remove the BP and press F8, wow kaka we go to the OEP !!!!!!!!!!!

Address	Hex dump	Disassembly	C
004271B0	55	push ebp	
004271B1	8BEC	mov ebp, esp	
004271B3	6A FF	push -1	
004271B5	68 600E4500	push 00450E60	
004271BA	68 C8924200	push 004292C8	
004271BF	64:A1 00000000	mov eax, dword ptr fs:[0]	
004271C5	50	push eax	
004271C6	64:8925 00000000	mov dword ptr fs:[0], esp	
004271CD	83C4 A8	add esp, -58	
004271D0	53	push ebx	

_This Time to dump and fix IAT. Use Ollydump plugin to dump the target (uncheck **rebuild import** option) and save as any name you like (ex: dumped.exe). Now, open ImportRect to fix IAT, for the right process. Edit the OEP, then IAT Auto Search and Get Imports:



_huh, have some invalids. Not a problems, we will use the plug in: **nt_krnlp~~rot~~ect_0_1.dll** of **bpx** to resolve all invalids:



_Hehe, The plugin works very well! Let's fix dump now, and run our fixed file to test. J Kaka, it runs normally. The splash screen protector of the disappears!



_Repair Fixed the file by LordPE J

_The End. I hope my poor English with all of you can understand what I write. See you in another tutorials.

Best Regards

_ [Kienmanowar] _



--+ +---==[**Greatz thanks to**]=---+ +--

My family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHOCrker, the_Lighthouse, Merc, Hoadongnoi, Nini ... all REA's members, TQN, HacNho, RongChauA, Deux, tlandn, light.phoenix, dqtlN, dump all my friend, and you.

Thanks to --+ +---==[]=---+ +--

iamidiot, WhyNotBar, trickyboy, dzungltn, takada, hurt_heart, haule_nth, v. hytkl. v.. You have contributed greatly to the REA. Hope you will continue to promote J

I want to thank **Teddy Rogers** for his great site, Reversing.be folks (especially **haggar**), Arteam folks (**Shub-Nigurrath**, **MaDMAn_H3rCuL3s**) and all folks on crackmes.de, thank to all members of **unpack.cn** (especially **fly** and **linhanshi**). Great thanks to **lena151** (I like your tutorials). And finally, thanks to **Ricardo NARVAJA** and all members on **CRACKSLATINOS**.

>>>> If you have any suggestions, comments or corrections email me: **kienmanowar [at] reaonline.net**

Tutorials hacnho # 4

Manual unpacking PE Diminisher v0.1 template by **koncool** and **R @ dier**.

Information	Unpacking for Newbie's
Target	Unpackme.exe
Available	http://nhandan.info/hacnho/tuts/unpackme4_tuts.zip
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, Lord PE 1.4, PESniffer 3.2b.
Protection	PE Diminisher v0.1 - Crappy PE Packer by Teraphy
L Evel	Easy
Category	Manual unpacking

1. Introduction

Hi all, in this tut, I will unpack introduction to how easy packer of **Teraphy**. This is **Diminisher PE v0.1**. This is a very basic method for unpack a packer.

2. Getting Started

First step, you have to find some info from this PE software. Open Lord PE, PE Editor choose. And we have:

[PE Editor] - e:\cracker\unpackme.exe

Basic PE Header Information

EntryPoint:	00004000	Subsystem:	0002	...
ImageBase:	00400000	NumberOfSections:	0004	
SizeOfImage:	00005000	TimeDateStamp:	406A748A	
BaseOfCode:	00001000	SizeOfHeaders:	00000400	? +
BaseOfData:	00002000	Characteristics:	010F	...
SectionAlignment:	00001000	Checksum:	00008118	?
FileAlignment:	00000200	SizeOfOptionalHeader:	00E0	
Magic:	010B	NumOfRvaAndSizes:	00000010	+ -

OK
Save
Sections
Directories
FLC
TDSC
Compare
L

[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	00000026	00000400	00000020	E0000020
.rdata	00002000	00000092	00000600	00000051	C0000040
.data	00003000	000000A8	00000800	0000008B	C0000040
.teraphy	00004000	00001000	00000A00	0000041A	C0000040

EP: 4000, The value of flags this case is not needed, Image Base is always 400000, Import Table: 0000 and size is 00.

3. Finding the OEP

Load into unpackme.exe Olly. We still in line **00404000> 53 PUSH EBX**. Now, press F7 to trace to address **00404006**. At this line, you see in the **Registers (FPU)** table. The value of **ESP** is **0012FFAC**, then you right click on and choose **ESP Follow in the dump**.

The screenshot shows the OllyDbg interface. The assembly window displays the following instructions:

```

00404000 53      PUSH EBX
00404001 51      PUSH ECX
00404002 52      PUSH EDX
00404003 56      PUSH ESI
00404004 57      PUSH EDI
00404005 55      PUSH EBP
00404006 E8 00000000 CALL unpackme.0040400B
0040400B 5D      POP EBP
0040400C 8B05    MOV EDX,EBP
0040400E 81ED A2304000 SUB EBP,unpackme.004030A2
00404014 2B95 91334000 SUB EDX,DWORD PTR SS:[EBP+403391]
0040401A 81EA 0B000000 SUB EDX,0B
00404020 8995 9A334000 MOV DWORD PTR SS:[EBP+40339A],EDX
00404026 80BD 99334000 CMP BYTE PTR SS:[EBP+403399],0
0040402D 74 50    JE SHORT unpackme.0040407F
0040402F E8 02010000 CALL unpackme.00404136
00404034 8BFD    MOV EDI,EBP
00404036 8D9D 9A334000 LEA EBX,DWORD PTR SS:[EBP+40339A]
0040403C 8B1B    MOV EBX,DWORD PTR DS:[EBX]
0040403E 8D87 9E334000 LEA EAX,DWORD PTR DS:[EDI+40339E]
00404044 8B05    MOV EAX,DWORD PTR DS:[EAX]
00404046 03D8    ADD EBX,EAX
00404048 8D8F A2334000 LEA ECX,DWORD PTR DS:[EDI+4033A2]
0040404E 8B09    MOV ECX,DWORD PTR DS:[ECX]
00404050 66 8B85 8F3340 MOV AX,WORD PTR SS:[EBP+40338F]
00404057 8003 10    ADD BYTE PTR DS:[EBX],10
0040405A 3003    XOR BYTE PTR DS:[EBX],AL
0040405C 3023    XOR BYTE PTR DS:[EBX],AH
0040405E 8003 AA    ADD BYTE PTR DS:[EBX],0AA
00404061 66 C1C0 03    ROL AX,3
00404065 86F0    XCHG AL,AH

```

The registers window shows the following values:

```

Registers (FPU)
EAX 00000000
ECX 0012FFB0
EDX 7FFE0304
EBX 7FFDF000
ESP 0012FFAC

```

A context menu is open over the ESP register, with the following options:

- Increment
- Decrement
- Zero
- Set to 1
- Modify
- Copy to clipboard
- Follow in Dump
- Follow in Stack
- View MMX registers
- View 3DNow! registers
- View debug registers
- Make Label
- Appearance

The Hex dump window shows the following data:

```

Address Hex dump
0012FFAC F0 FF 12 00 20 B7 D5 77
0012FFB4 FF FF FF FF 04 03 FE 7F
0012FFB8 B0 FF 12 00 00 F0 FD 7F
0012FFC4 69 EB E7 77 20 B7 D5 77
0012FFC8 FF FF FF FF 00 F0 FD 7F
0012FFD4 54 0C 00 FF C8 FF 12 00

```

Then you go to the Hex dump window. Then right click on the value **0012FFC0** and select Breakpoint -> Hardware, on Access -> Word. Our breakpoint is now set.

Then Press **F9** to run unpackme, after that, continue press **F8 5 times** until you see as follows:

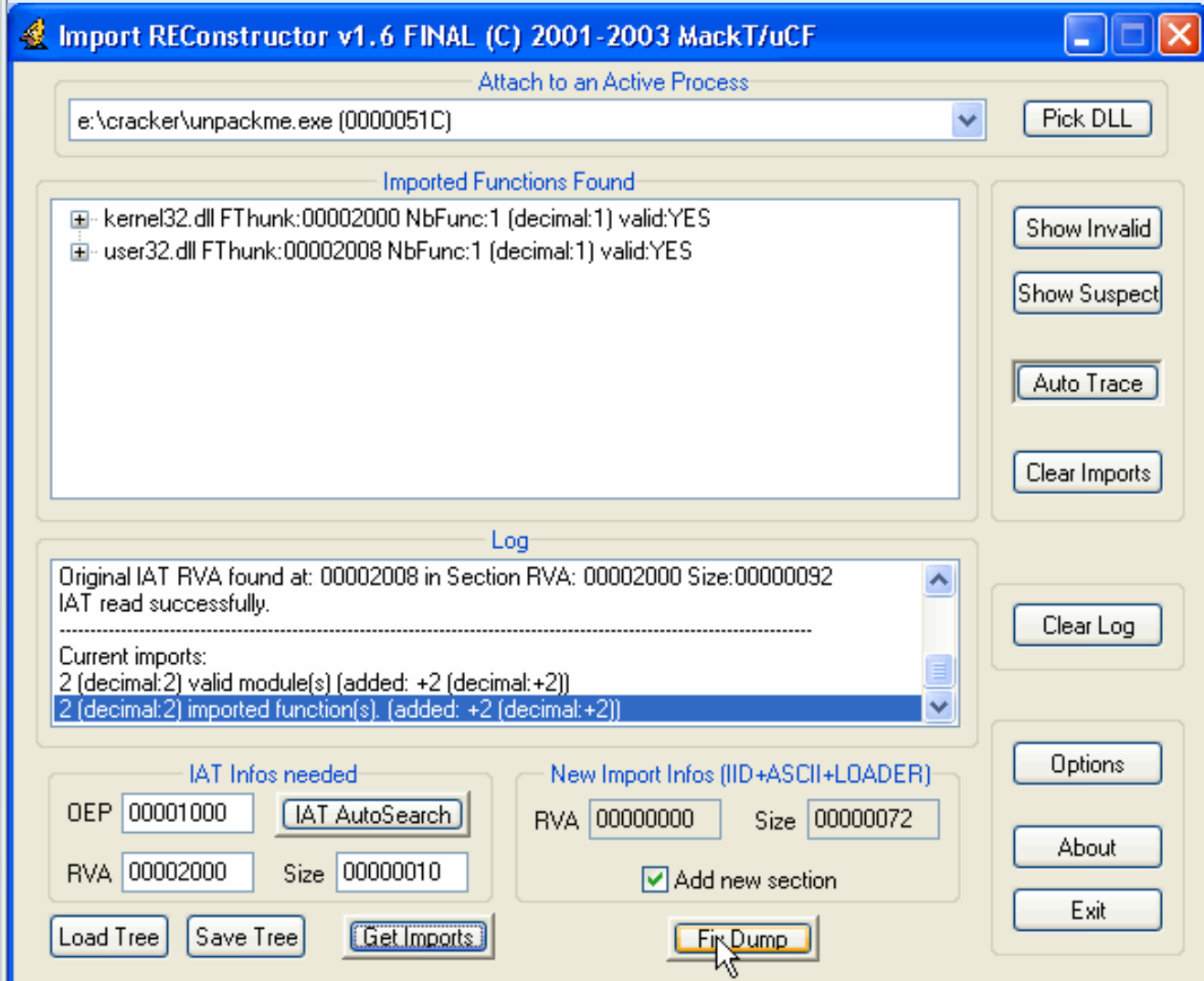
The screenshot shows the OllyDbg interface. The assembly window displays the following instructions:

```

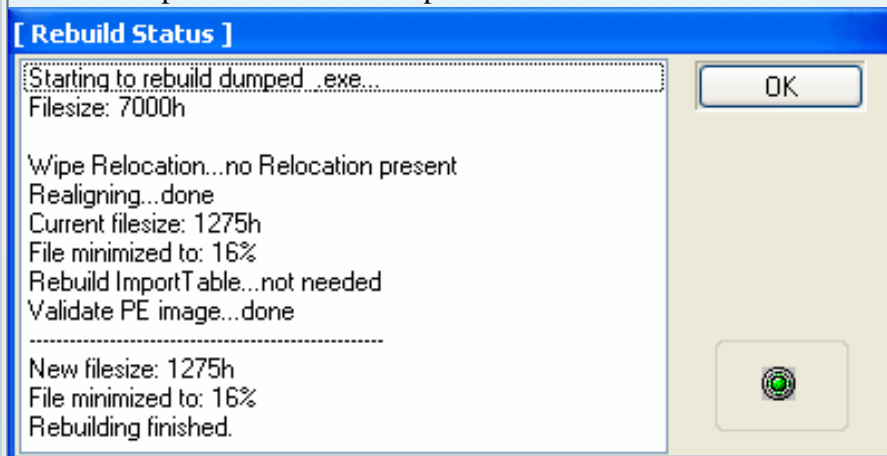
00404093 - FFE0    JMP EAX
00404095 8B95 9A334000 MOV EDX,DWORD PTR SS:[EBP+40339A]
0040409B 8BB5 F6334000 MOV ESI,DWORD PTR SS:[EBP+4033F6]
004040A1 33FF    XOR EDI,EDI
004040A3 03F2    ADD ESI,EDX
004040A5 03FA    ADD EDI,EDX
004040A7 8B46 0C    MOV EAX,DWORD PTR DS:[ESI+C]
004040AA 85C0    TEST EAX,EAX
004040AC 74 7F    JE SHORT unpackme.0040412D
004040AE 03C2    ADD EAX,EDX

```


And select Open ImpREC attached to active process and choose unpackme.exe. Change the value in the OEP window to the one we wrote down earlier (1000) then select IATAutosearch then click Get Imports.



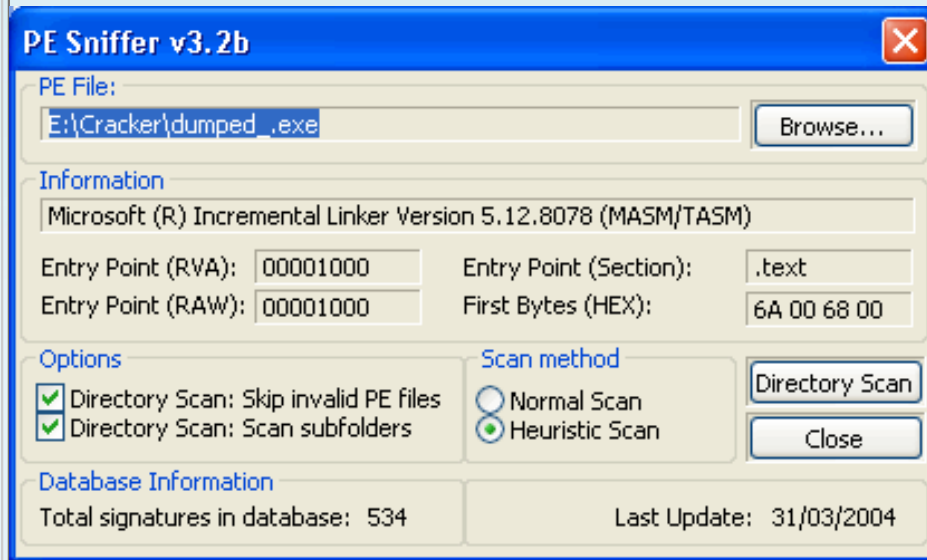
Import Functions all valid ... Now, click fix to fix IAT dump the file **dumped.exe**. And then LordPE open, choose rebuild PE optimize for size of unpackme ..



6. Testing Our Unpacked file

Now run unpacked files. Wow, not crash.

Using PESniffer 3.2b for detect: **MASM / TASM**. Okie, PE Diminisher v0.1 - Crappy PE Packer by **Teraphy** is now unpacked successful!



7. Conclusion

Special thanx to **koncool** et **R @ dier** for this template.

My Greetz to: Deux, RCA, Moonbaby, Computer_Angel, tlandn, [R @ dier](#), Zombie, Maip0301, tykhung, softcracker_vn, CTL, LeVuHoang ...

To be continued ...

Written by hacnho (tutorial date: Sai Gon 2/4/2004)

Tutorials hachno # 6

Manual unpacking PE Lock NT 2:04 template by koncool and R @ dier.

Information	Unpacking for Newbie's
Target	Unpackme.exe
Available	http://nhandan.info/hachno/tuts/unpackme6_tuts.zip
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, Lord PE 1.4, PESniffer 3.2b.
Protection	PELOCKnt v2.04 Copyright (C): Marquis: DE: soirée:
L Evel	Standard
Category	Manual unpacking

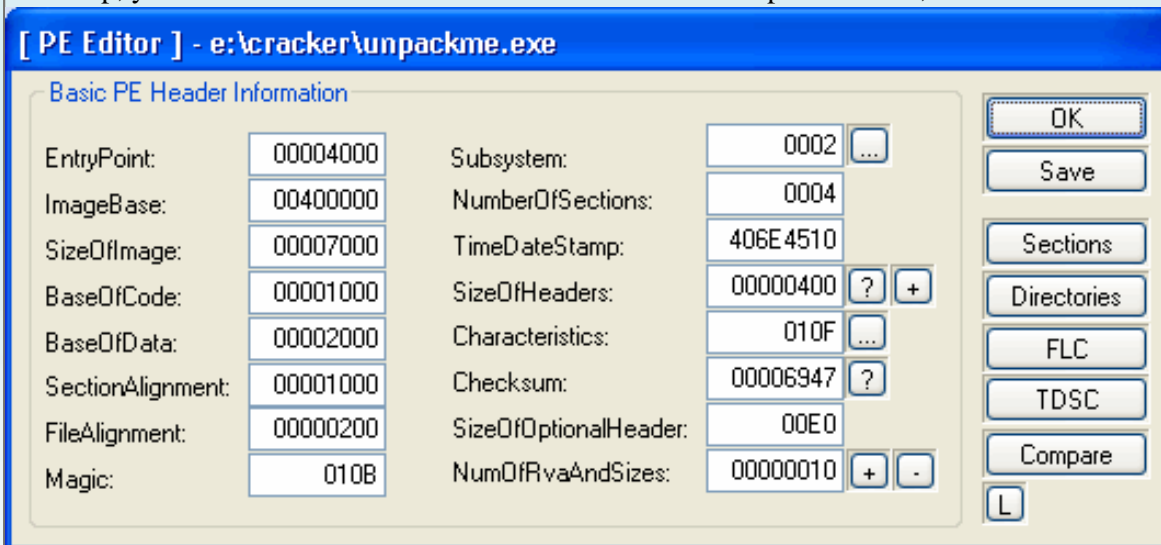
1. Introduction

Bonjour Mesdames et Mesieur ;-). Today, I will be the introduction method "how to unpack PELOCKnt v2.04 Copyright (C): Marquis: DE: soirée". I hope you have some time happiness with this tut!

Tuts only for the beginner. If you are a bro. Please do not mock my tuts ... You can contact me for help and improve it.

2. Getting Started

First step, you have to find some info from this PE software. Open Lord PE, PE Editor choose. And we have:



[Section Table]					
Name	VOffset	VSize	ROffset	RSize	Flags
PELOCKnt	00001000	00000026	00000400	00000200	E0000020
.rdata	00002000	00000092	00000600	00000200	C0000040
.data	00003000	000000A5	00000800	00000200	C0000040
PELOCKnt	00004000	00002A00	00000A00	00002A00	C0000060

EP: 4000, The value of flags this case is not needed, Image Base is always 400000, Import Table: 0000 and size is 00.

3. Finding the OEP

Load **unpackme.exe** into OllyDBG. And you still here:

00404000	EB 03	JMP SHORT unpackme.00404005	
00404002	CD 20	INT 20	
00404004	C7	???	Unknown command
00404005	1E	PUSH DS	
00404006	EB 03	JMP SHORT unpackme.0040400B	
00404008	CD 20	INT 20	
0040400A	EA 9CEB02EB 01	JMP FAR EB01:EB02EB9C	Far jump
00404011	01EB	ADD EBX,EBP	
00404013	60	PUSHAD	
00404014	EB 03	JMP SHORT unpackme.00404019	
00404016	CD 20	INT 20	
00404018	EB EB	JMP SHORT unpackme.00404005	
0040401A	01EB	ADD EBX,EBP	
0040401C	E8 03000000	CALL unpackme.00404024	
00404021	E9 EB045840	JMP 40984511	
00404026	50	PUSH EAX	
00404027	C3	RETN	

Then, you press **F7** until you see as follows:

00404013	60	PUSHAD	
00404014	EB 03	JMP SHORT unpackme.00404019	
00404016	CD 20	INT 20	
00404018	EB EB	JMP SHORT unpackme.00404005	
0040401A	01EB	ADD EBX,EBP	
0040401C	E8 03000000	CALL unpackme.00404024	
00404021	E9 EB045840	JMP 40984511	
00404026	50	PUSH EAX	
00404027	C3	RETN	
00404028	EB 03	JMP SHORT unpackme.0040402D	
0040402A	CD 20	INT 20	
0040402C	EB EB	JMP SHORT unpackme.00404019	
0040402E	03CD	ADD ECX,EBP	
00404030	2003	AND BYTE PTR DS:[EBX],AL	
00404032	FC	CLO	
00404033	EB 03	JMP SHORT unpackme.00404038	

Continued, you have to press **ALT + M** to open the **Memory of MAP** OllyDBG.

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00010000	00001000				Priv	RW	RW	
00020000	00001000				Priv	RW	RW	
00120000	00001000				Priv	RW	RW	
0012E000	00002000			stack of ma	Priv	RW	RW	
00130000	00001000				Map	R	R	
00140000	00004000				Priv	RW	RW	
00240000	00006000				Priv	RW	RW	
00250000	00001000				Map	RW	RW	
00260000	00016000				Map	R	R	
00280000	00034000				Map	R	R	
002C0000	00041000				Map	R	R	
00310000	00006000				Map	R	R	
00320000	00005000				Map	R E	R E	
003E0000	00002000				Map	R E	R E	
003F0000	00001000				Priv	RW	RW	
00400000	00001000	unpackme		PE header	Image	R	RWE	
00401000	00001000	unpackme	PELOCKnt	code	Image	R	RWE	
00402000	00001000	unpackme	.rdata	data				
00403000	00001000	unpackme	.data	data				
00404000	00003000	unpackme	PELOCKnt	SFX, imports				
00410000	00103000							
00520000	000BA000							
00820000	00001000							
77C70000	00001000	GDI32		PE header				
77C71000	000038000	GDI32	.text	code, import				
77CAC000	00001000	GDI32	.data	data				
77CA0000	00001000	GDI32	.rsrc	resources				
77CAE000	00002000	GDI32	.reloc	relocations				
77CC0000	00001000	RPCRT4		PE header				
77CC1000	000067000	RPCRT4	.text	code, import				
77D28000	00007000	RPCRT4	.orpc	code				
77D2F000	00001000	RPCRT4	.data	data				
77D30000	00001000	RPCRT4	.rsrc	resources				
77D31000	00004000	RPCRT4	.reloc	relocations				
77D40000	00001000	USER32		PE header				
77D41000	0005C000	USER32	.text	code, import				
77D90000	00002000	USER32	.data	data				
77D9F000	0002B000	USER32	.rsrc	resources				
77DCA000	00003000	USER32	.reloc	relocations				
77DD0000	00001000	ADVAPI32		PE header				
77DD1000	00065000	ADVAPI32	.text	code, import				
77E36000	00005000	ADVAPI32	.data	data				
77E3B000	0001B000	ADVAPI32	.rsrc	resources				
77F51000	00005000	ADVAPI32	.reloc	relocations				

Actualize
View in Disassembler Enter
Dump in CPU
Dump
Search Ctrl+B
Set break-on-access F2
Set memory breakpoint on access
Set memory breakpoint on write
Set access
Copy to clipboard
Sort by
Appearance

Et puis, you search the address line contain **401,000**. Right click on it and choose **Set breakpoint on memory access**. Now, press **Shift + F9**:

004040BC	AD	LODS DWORD PTR DS:[ESI]	
004040BD	EB 03	JMP SHORT unpackme.004040DC2	
004040BF	CD 20	INT 20	
004040C1	C786 E033C1EB	MOV DWORD PTR DS:[ESI+EBC133E0],EB01EB0	
004040CB	01EB	ADD EBX,EBP	
004040CD	F7C1 04000000	TEST ECX,4	
004040D3	EB 03	JMP SHORT unpackme.004040D8	
004040D5	CD 20	INT 20	
004040D7	EB EB	JMP SHORT unpackme.004040C4	
004040D9	01EB	ADD EBX,EBP	

Following, you press **Shift + F9** again:

004040FF	AB	STOS DWORD PTR ES:[EDI]	
00404E00	EB 03	JMP SHORT unpackme.00404E05	
00404E02	CD 20	INT 20	
00404E04	03E2	ADD ESP,EDX	
00404E06	02EB	ADD CH,BL	
00404E08	05 E912FFFF	ADD EAX,FFFF12E9	
00404E0D	FFEB	JMP FAR EBX	Illegal use of register
00404E0F	04 CD	ADD AL,0CD	
00404E11	20EB	AND BL,CH	

Waaaa, the job is very uninspired ... You have press **Shift + F9** until you see as follows (I count about **260 times** he he, what do you think about this ;;):

00401000	60	DB 60	CHAR 'j'
00401001	00	DB 00	
00401002	68	DB 68	CHAR 'h'
00401003	00	DB 00	
00401004	30	DB 30	CHAR '0'
00401005	40	DB 40	CHAR '@'
00401006	00	DB 00	
00401007	68	DB 68	CHAR 'h'
00401008	34	DB 34	CHAR '4'
00401009	30	DB 30	CHAR '0'
0040100A	40	DB 40	CHAR '@'
0040100B	00	DB 00	
0040100C	6A	DB 6A	CHAR 'j'
0040100D	00	DB 00	

Next, press **CTRL + A** analyze the code for:

00401000	. 6A 00	PUSH 0	<pre> Style = MB_OK!MB_APPLMODAL Title = ".....--<[VCT-Vietnamese Crackers TEAM!]> Text = "Try to Unpack me! Packed with PElockNT v2.4 hOwner = NULL MessageBoxA ExitCode = 0 ExitProcess kernel32.ExitProcess USER32.MessageBoxA </pre>
00401002	. 68 00004000	PUSH unpackme.00403000	
00401007	. 68 34304000	PUSH unpackme.00403034	
0040100C	. 6A 00	PUSH 0	
0040100E	. E8 00000000	CALL unpackme.00401020	
00401013	. 6A 00	PUSH 0	
00401015	. E8 00000000	CALL unpackme.0040101A	
0040101A	.- FF25 00204000	JMP DWORD PTR DS:[402000]	
00401020	\$. FF25 00204000	JMP DWORD PTR DS:[402000]	
00401026	00	DB 00	
00401027	00	DB 00	

Our OEP!

Congratulations! According OEP we found is **401,000**. And now we Calculate the real OEP of this unpackme by the formula:
Real OEP = OEP find in Olly-Image Base = 401000-400000 = **1000**.

4th dumping

At address **00401000**, we go to the menu Plugin -> OllyDump -> dump debugged process. And then, just press dump, save the file at **dumped.exe** unpacked.

OllyDump - unpackme.exe

Start Address: 400000 Size: 6400 **Dump**

Entry Point: 4000 -> Modify: 1000 **Get EIP as OEP** **Cancel**

Base of Code: 1000 Base of Data: 2000

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Off...	Raw Size	Raw Off...	Charact...
PELOCKnt&	00000026	00001000	00000026	00001000	E0000020
.rdata	00000092	00002000	00000092	00002000	C0000040
.data	000000A5	00003000	000000A5	00003000	C0000040
PELOCKnt	00002A00	00004000	00002A00	00004000	C0000060

☒ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

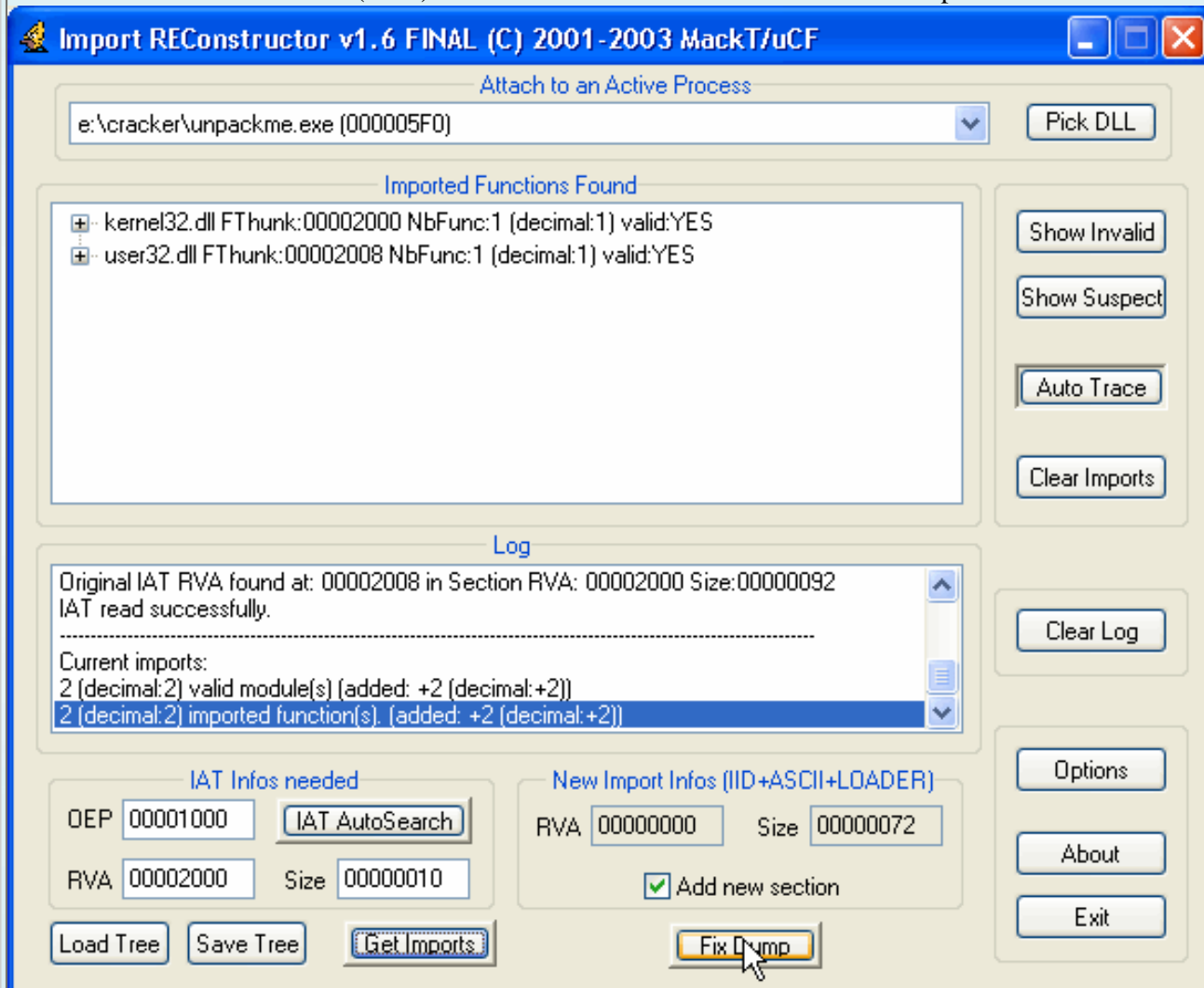
☐ Method2 : Search DLL & API name string in dumped file

Do not run **dumped.exe** now, will be a crash ... It must fix IAT.



5. Finding and Fixing the Adress Import Table

And select Open ImpREC attached to active process and choose unpackme.exe. Change the value in the OEP window to the one we wrote down earlier (1000) then select IATAutosearch then click Get Imports.

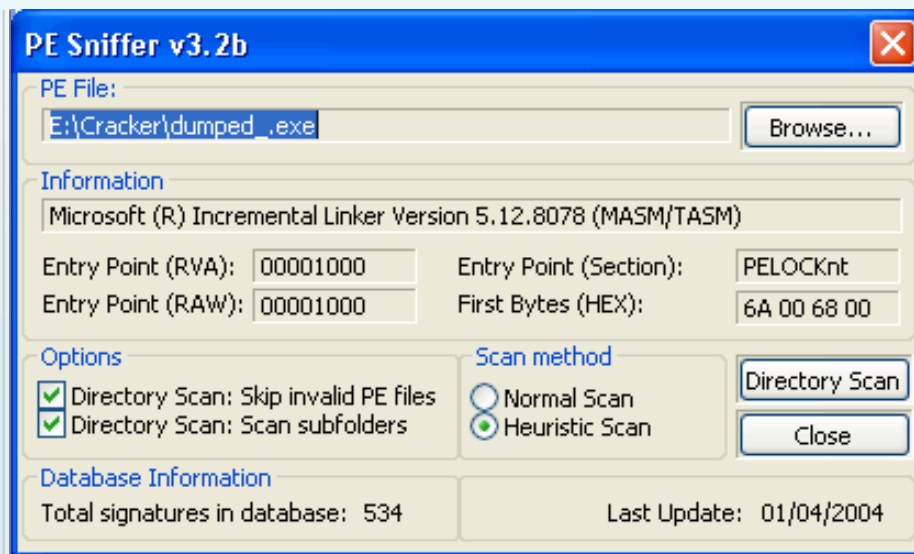


Import Functions all valid ... Now, click fix dump to fix the IAT **dumped.exe** file.

6. Testing Our Unpacked file

Now run unpacked files. Wow, not crash.

Using **PE 3.2b** for **Sniffer** detect: **MASM / TASM**. Okie, **PELOCKnt v2.04 by Marquis: DE: soirée** is now unpacked successful!



7. Conclusion

Special thanx to **koncool** et **R @ dier** for this template.

My Greetz to: Deux, NVH (c), luucorp, Maip0301, R @ dier, tlandn, CTL, JAL, LeVuHoang, 777, LeonHart, Bin ... and you ;-)!

To be continued ...

Written by [hacnho](#) (tutorial date: Sai Gon 5/4/2004)

hacnho Tutorials # 2

Manual unpacking PECompact 1.84 template by koncool and R @ dier.

Information	Unpacking for Newbie's
Target	Unikey 3:55 for XP (http://unikey.sourceforge.net/)
Available	http://nhandan.info/hacnho/tuts/unpackme2_tuts.zip
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, ImpRec 1.6 for XP, Lord PE 1.4, PESniffer 3.2b.
Protection	PECompact 1.84 by Jeremy Collake
Level	Easy
Category	Manual unpacking

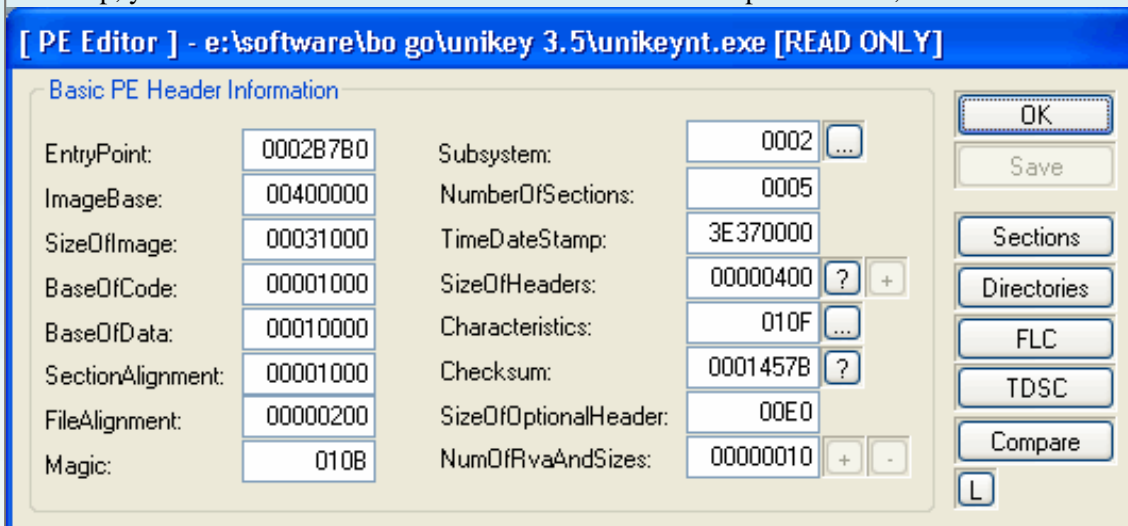
1. Introduction

Salut tout le monde!

This is my second tut for manual Unpacking. In this tut, I will introduction to manual unpacking PECompact 1.84. Because the tutorial "**Manual unpacking PeCompact 1.76**" of R @ dier can not apply for the newer version of PE Compact. So I found was a method for unpack the newer version of this packer and want to sharing with every body.

2. Getting Started

First step, you have to find some info from this PE software. Open Lord PE, PE Editor choose. And we have:



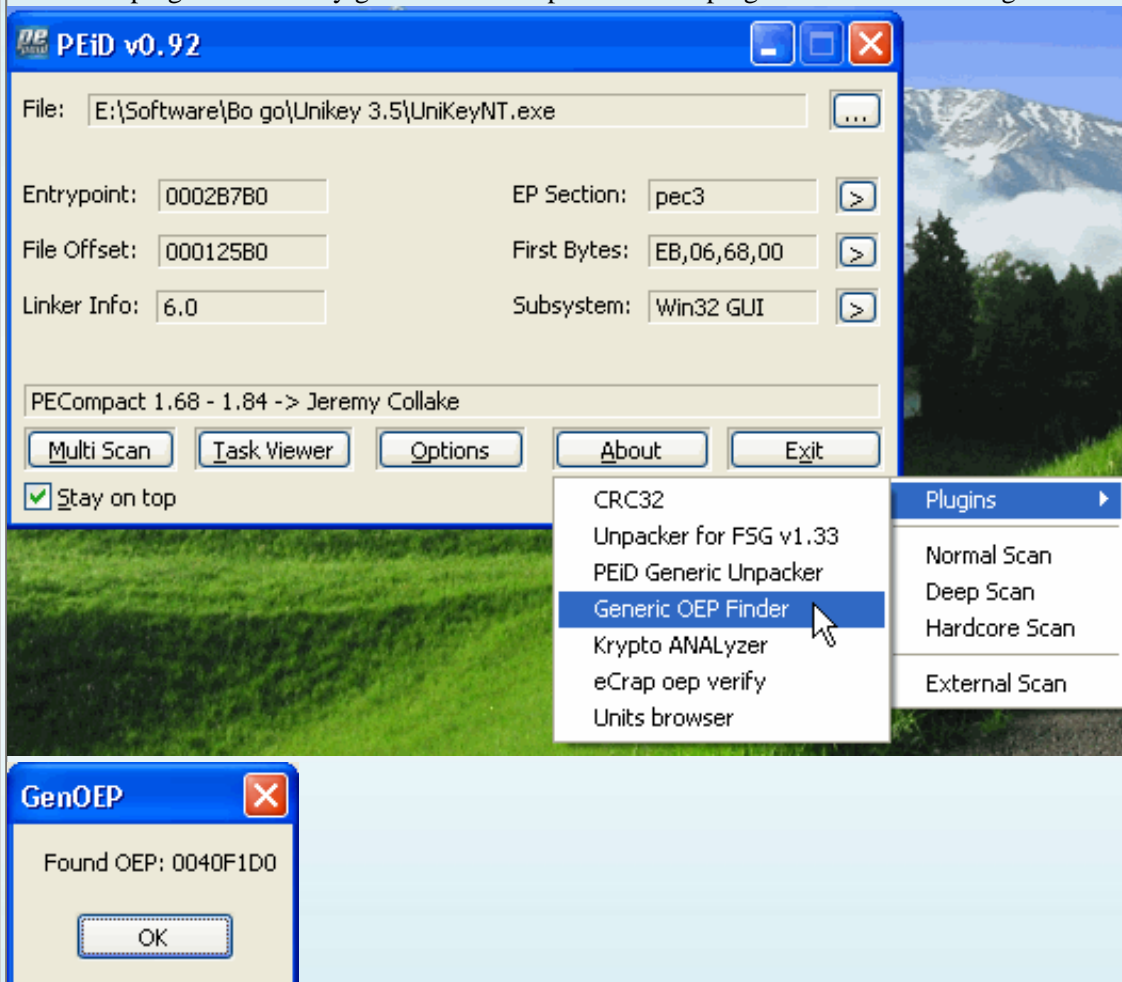
[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
pec1	00001000	0001C000	00000400	0000B000	E0000020
.rsrc	0001D000	00006000	0000B400	00004200	C0000040
pec2	00023000	00008000	0000F600	00002800	C2000040
pec3	0002B000	00005000	00011E00	00000E00	C0000040
.rsrc	00030000	00001000	00012C00	00000400	C0000040

EP: 2B7B0, The Flags is (E0000020; C0000040, C2000040, C0000040, C0000040), Image Base is always 400000, Import Table: 30000 is the size and AF.

3rd Finding the OEP

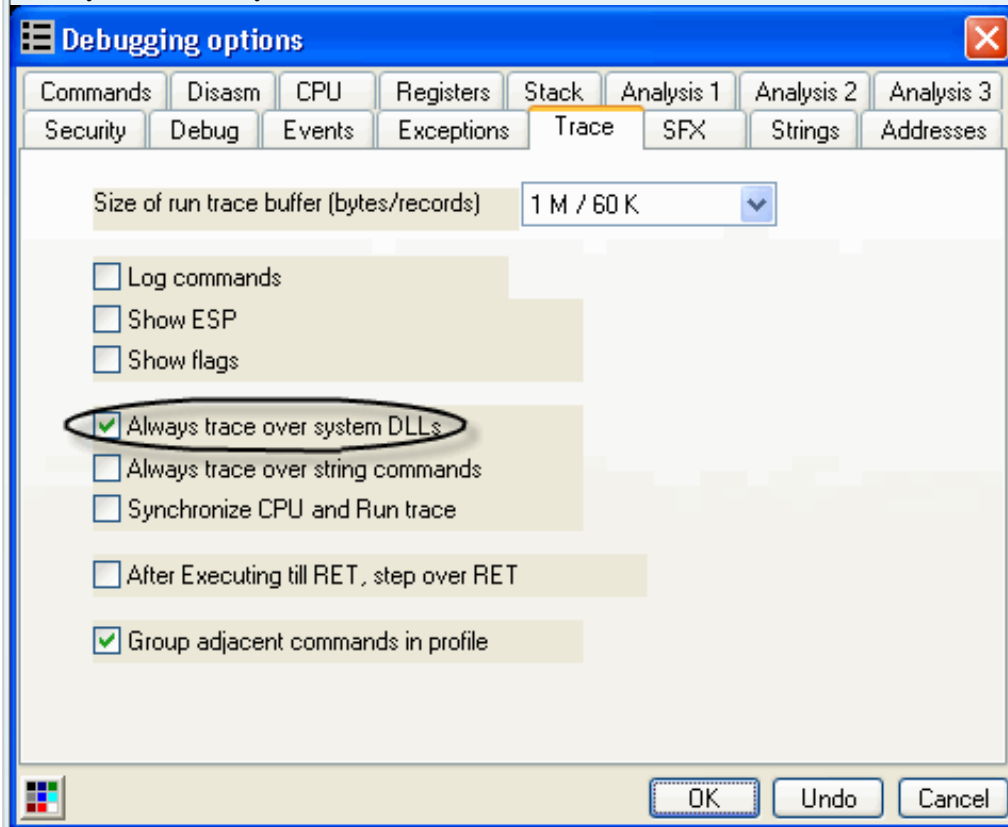
The PEiD plugin have a very good for tech unpack. This is plugin OEP. Now! Let's go to detect OEP of this soft.



Waaa ... The OEP is 0040F1D0. Okie, now we have to Calculate the real OEP by the Formule:
Real OEP = OEP find in PEiD-Image Base = 400000 = **40F1D0-F1D0**.

4th dumping

For dumping, we have to configure the options in Olly Trace. Press Alt + O and modify: Checked the CheckBox in the "Always trace over system DLLs"

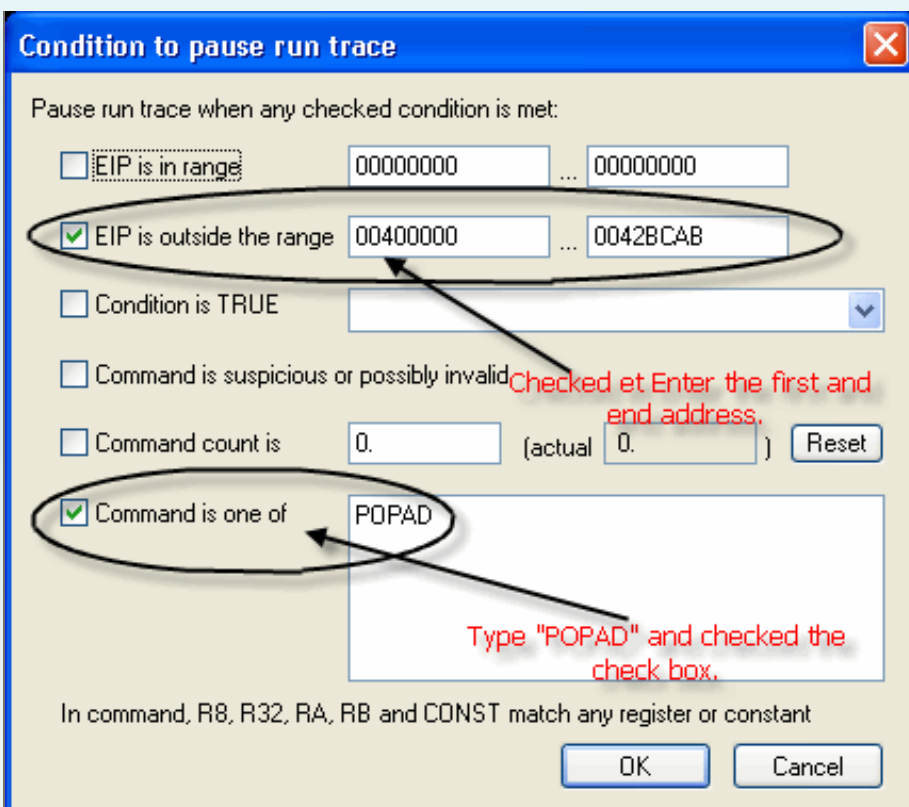


Now, this load unpackme in to Olly. And we still here:

0042B7A1	004465 6C	ADD BYTE PTR SS:[EBP+6C],AL	
0042B7A5	65:74 65	JE SHORT Un iKeyNT.0042B80D	Superfluous prefix
0042B7A8	46	INC ESI	
0042B7A9	696C65 41 0000	IMUL EBP,DWORD PTR SS:[EBP+41],EB000000	
0042B7B1	06	PUSH ES	
0042B7B2	68 00B00200	PUSH 2B000	
0042B7B7	C3	RETN	
0042B7B8	9C	PUSHFD	
0042B7B9	60	PUSHAD	
0042B7BA	E8 02000000	CALL Un iKeyNT.0042B7C1	
0042B7BF	33C0	XOR EAX,EAX	

Registers (FPU)	
EAX	00000000
ECX	0006FFB0
EDX	7FFE0304
EBX	7FFDF000
ESP	0006FFC4
EBP	0006FFF0
ESI	00000000
EDI	00000000
EIP	0042B7B0

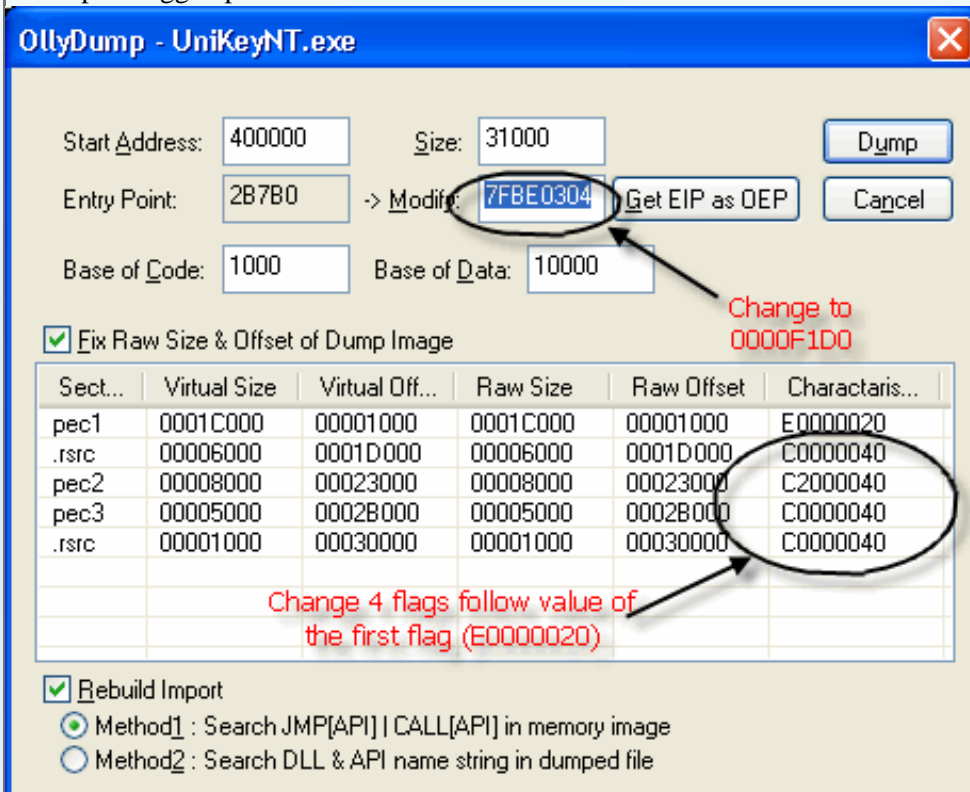
Next is a **CTRL + T** and bring up the trace window. All steps follow this pictures ...



After, you press **F11 + Ctrt** for Trace Into. Press Continue until you see the address 00430104:

004300FA	FFFF	???	Unknown command
004300FC	0000	ADD BYTE PTR DS:[EAX],AL	
004300FE	0000	ADD BYTE PTR DS:[EAX],AL	
00430100	C8 000000	ENTER 0,0	
00430104	55	PUSH EBP	
00430105	8B75 08	MOV ESI,DWORD PTR SS:[EBP+8]	
00430108	8B7D 0C	MOV EDI,DWORD PTR SS:[EBP+C]	
0043010B	FC	CLD	
0043010C	B2 80	MOV DL,80	

Okie, address line at 430,104. You press F9 to run Unikey. Unikey When running, we go to the menu Plugin -> OllyDump -> dump debugged process.

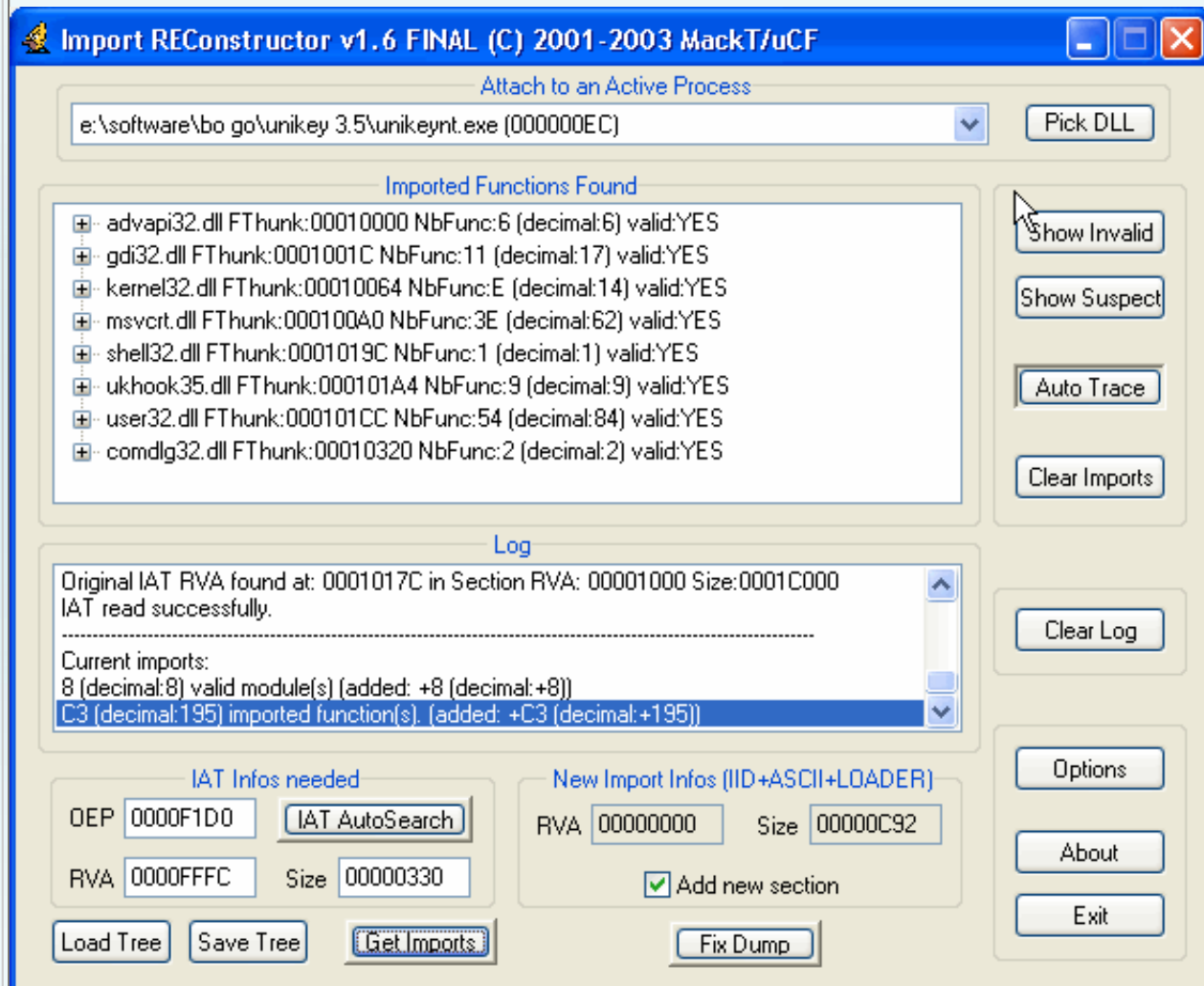


Change the Entry Point to modify **F1D0**. 4 flags and change (in Olly characteristics) follow the value of the first

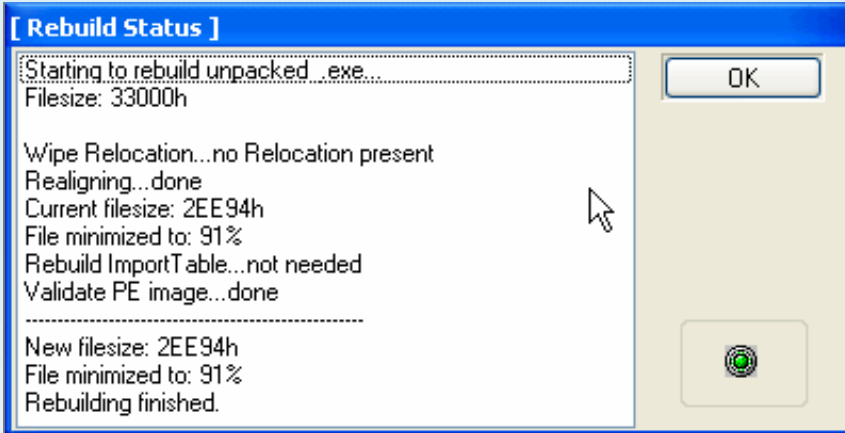
flag (**E0000020**). And then, just press dump, save the unpacked files. Now, do not shut down OllyDbg just yet, we need to get the import table and fix our exe.

5th Finding and Fixing the Adress Import Table

ImpREC open, select Print ImpRec attached to active process and choose our target program. Change the value in the OEP window to the one we wrote down earlier (**F1D0**) then select IAT Autosearch then click Get Imports.



Ohh, all imports is valid. We have to do now is fix our exe click on **Fix dump** and select our unpacked.exe and we are done :-> our dump will be saved as unpacked_.exe. And then, we have to rebuild this file unpacked. Open Lord PE. Choose rebuild PE. Browse to unpack the file, click OK.



6. Testing Our Unpacked file

Now run Unikey unpacked files. Wow, not crash.

Using PESniffer detect 3.2b for: **Microsoft Visual C + + 6.0**. Okie, PECompact 1.84 by **Jeremy Collake** is now unpacked successful!

7. Conclusion

Special thanx to **koncool** et **R @ dier** for this template.

My Greetz to: Deux, RCA, Moonbaby, Computer_Angel, tlandn, R @ dier, Zombie, Maip0301, tykhung, softcracker_vn, CTL, LeVuHoang ...

To be continued ...

Written by hacnho (tutorial date: Sai Gon 17/3/2004)

hacnho Tutorials # 10

Manual unpacking PECompact 2.0 Final -> Jeremy Collake

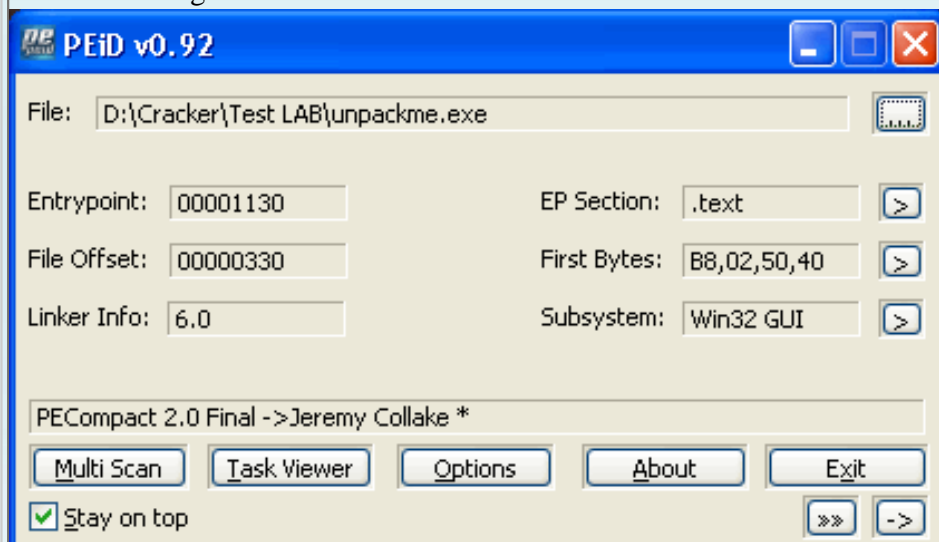
Information	Unpacking for Newbie's
Target	unpackme.exe
Available	http://nhandan.info/hacnho/tuts/unpackme10_tuts.zip
Tools	OllyDbg plugin 1.10c with OllyDump 2.21.108, ImportREC Final 1.6, 1.4 LordPE.
Protection	PECompact 2.0 Final -> Jeremy Collake
L Evel	Standard
Category	Manual unpacking

1. Introduction

Nowadays, the newest packer is PECompact 2.0x. This is a commercial packer packer, so very easy for unpack it. I will explain the ways for unpack this packer. I use PECompact 2.0 Final, but this method can support version 2.x.

2. Getting Started

Use PEiD and get some LordPE for PE Info.



[PE Editor] - d:\cracker\test lab\unpackme.exe

Basic PE Header Information

EntryPoint:	00001130	Subsystem:	0002	...	<input type="button" value="OK"/> <input type="button" value="Save"/> <input type="button" value="Sections"/> <input type="button" value="Directories"/> <input type="button" value="FLC"/> <input type="button" value="TDSC"/> <input type="button" value="Compare"/> <input type="button" value="L"/>
ImageBase:	00400000	NumberOfSections:	0002		
SizeOfImage:	00015000	TimeDateStamp:	409787CB		
BaseOfCode:	00001000	SizeOfHeaders:	00000200	? +	
BaseOfData:	00002000	Characteristics:	010F	...	
SectionAlignment:	00001000	Checksum:	000052CD	?	
FileAlignment:	00000200	SizeOfOptionalHeader:	00E0		
Magic:	010B	NumOfRvaAndSizes:	00000010	+ -	

[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	00003000	00000200	00000600	E0000060
.rsrc	00004000	000108CC	00000800	00001200	E0000020

EP: 1130, The value of flags this case is not needed, Image Base is always 400000, Import Table: 0000 and size is 00.

3. Finding the OEP

Load **unpackme.exe** into OllyDBG. And you still here:

00401130	\$ B8 02504000	MOV EAX,unpackme.00405002	
00401135	. 50	PUSH EAX	
00401136	. 64:FF35 0000	PUSH DWORD PTR FS:[0]	
0040113D	. 64:8925 0000	MOV DWORD PTR FS:[0],ESP	
00401144	. 33C0	XOR EAX,EAX	
00401146	. 8908	MOV DWORD PTR DS:[EAX],ECX	
00401148	. 50	PUSH EAX	
00401149	. 45	INC EBP	
0040114A	. 43	INC EBX	
0040114B	. 6F	OUTS DX,DWORD PTR ES:[EDI]	I/O command
0040114C	. 6D	INS DWORD PTR ES:[EDI],DX	I/O command
0040114D	70 61	J0 SHORT unpackme.004011B0	
0040114F	. 637432 00	ARPL WORD PTR DS:[EDX+ESI],SI	
00401153	. 6C	INS BYTE PTR ES:[EDI],DX	I/O command
00401154	. C2 2801	RETN 128	

Then, you press **F9** two times and you see as follows:

00401130	\$ B8 02504000	MOV EAX,unpackme.00405002	
00401135	. 50	PUSH EAX	
00401136	. 64:FF35 0000	PUSH DWORD PTR FS:[0]	
0040113D	. 64:8925 0000	MOV DWORD PTR FS:[0],ESP	
00401144	. 33C0	XOR EAX,EAX	
00401146	. 8908	MOV DWORD PTR DS:[EAX],ECX	
00401148	. 50	PUSH EAX	
00401149	. 45	INC EBP	
0040114A	. 43	INC EBX	
0040114B	. 6F	OUTS DX,DWORD PTR ES:[EDI]	I/O command
0040114C	. 6D	INS DWORD PTR ES:[EDI],DX	I/O command
0040114D	70 61	J0 SHORT unpackme.004011B0	
0040114F	. 637432 00	ARPL WORD PTR DS:[EDX+ESI],SI	
00401153	. 6C	INS BYTE PTR ES:[EDI],DX	I/O command
00401154	. C2 2801	RETN 128	

Continued, you have to press **ALT + M** to open the **Memory of MAP** OllyDBG.

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00010000	00001000				Priv	RW	RW	
00020000	00001000				Priv	RW	RW	
00020000	00001000				Priv	RW	RW	
00020000	00001000				Priv	RW	RW	
0002E000	00002000			stack of ma	Priv	RW	RW	
00030000	00001000				Map	R	R	
00040000	00003000				Priv	RW	RW	
00024000	00006000				Priv	RW	RW	
00025000	00001000				Map	RW	RW	
00026000	00016000				Map	R	R	
00028000	00034000				Map	R	R	
0002C000	00041000				Map	R	R	
00031000	00006000				Map	R	R	
00040000	00001000	unpackme	PE header		Image	R	RWE	
000401000	00003000	unpackme	.text	code	Text	R	RWE	
000404000	00011000	unpackme	.rsrc	import	Resource	R	R	
77E60000	00001000	kernel32		PE hea				
77E61000	00075000	kernel32	.text	code, i				
77ED6000	00003000	kernel32	.data	data				
77ED9000	00066000	kernel32	.rsrc	resour				
77F3F000	00006000	kernel32	.reloc	reloca				
77F50000	00001000	ntdll		PE hea				
77F51000	0006F000	ntdll	.text	code, s				
77FC0000	00005000	ntdll	.ECODE	code				
77FC5000	00005000	ntdll	.data	data				
77FCA000	0002C000	ntdll	.rsrc	resour				
77FF6000	00003000	ntdll	.reloc	reloca				
7F6F0000	00007000							
7FFB0000	00024000							
7FFDE000	00001000			data b				
7FFDF000	00001000							
7FFE0000	00001000							

Continued, press **Shift + F9**. And you still here:

00405017	C602 E9	MOV BYTE PTR DS:[EDX],0E9	
0040501A	83C2 05	ADD EDX,5	
0040501D	2BCA	SUB ECX,EDX	
0040501F	894A FC	MOV DWORD PTR DS:[EDX-4],ECX	
00405022	33C0	XOR EAX,EAX	
00405024	C3	RETN	
00405025	B8 AB3F40F0	MOV EAX,F0403FAB	
0040502A	64:8F05 000000	POP DWORD PTR FS:[0]	
00405031	83C4 04	ADD ESP,4	
00405034	55	PUSH EBP	

Then press **F9 SHITF + 3** times and you still here:

00405025	B8 AB3F40F0	MOV EAX,F0403FAB	
0040502A	64:8F05 000000	POP DWORD PTR FS:[0]	
00405031	83C4 04	ADD ESP,4	
00405034	55	PUSH EBP	
00405035	53	PUSH EBX	
00405036	51	PUSH ECX	
00405037	57	PUSH EDI	
00405038	56	PUSH ESI	
00405039	8D98 3B100010	LEA EBX,DWORD PTR DS:[EAX+1000103B]	
0040503F	8B53 18	MOV EDX,DWORD PTR DS:[EBX+18]	
00405042	8BE8	MOV EBP,EAX	

Next, press **Ctrl + F12**. And we see as follows:

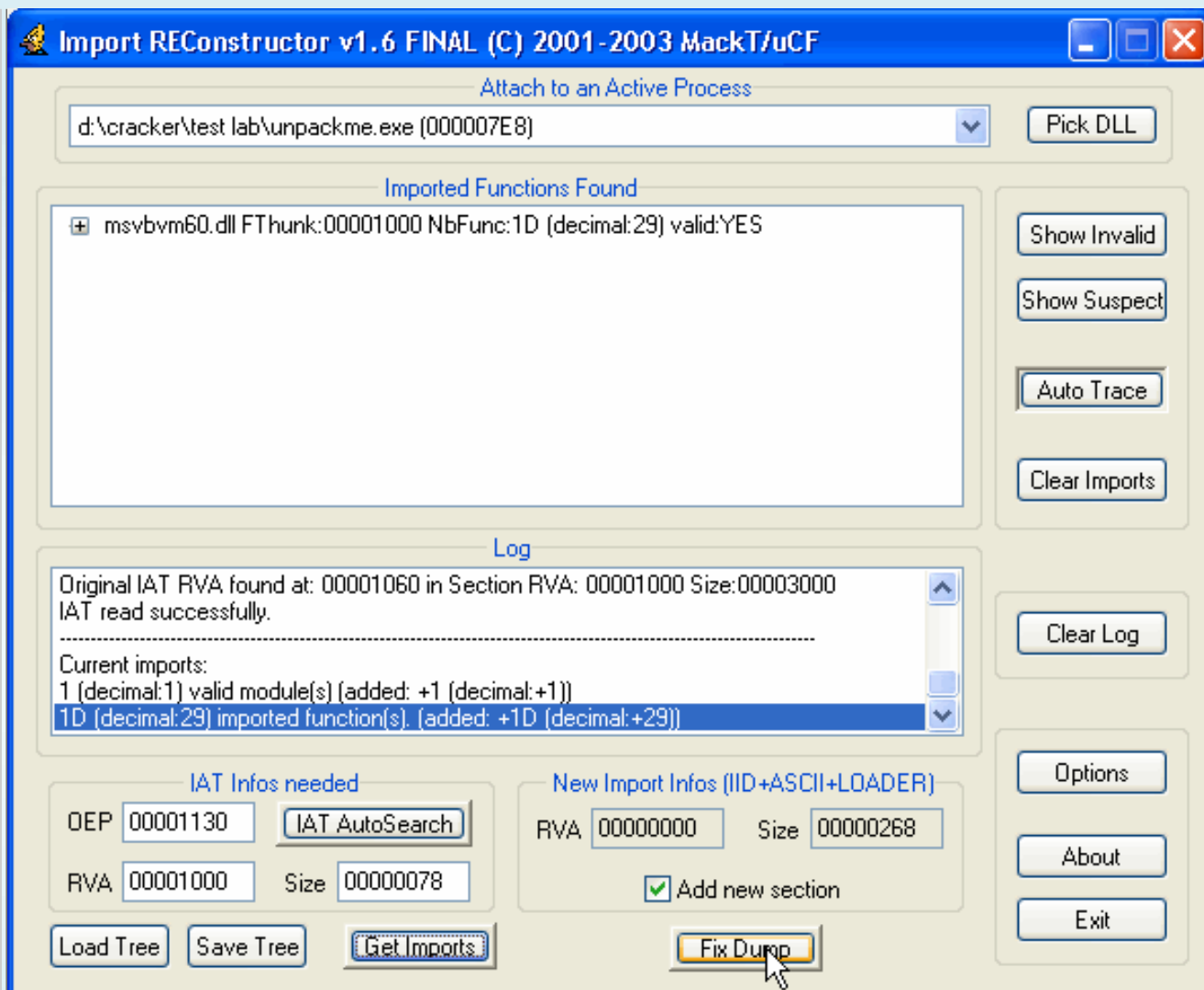
00320199	F3:04	REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:	
0032019B	8BF3	MOV ESI,EBX	
0032019D	8D8D B9120010	LEA ECX,DWORD PTR SS:[EBP+100012B9]	
003201A3	51	PUSH ECX	
003201A4	E8 45020000	CALL 003203EE	
003201A9	8B4E 2C	MOV ECX,DWORD PTR DS:[ESI+2C]	
003201AC	8B56 24	MOV EDX,DWORD PTR DS:[ESI+24]	
003201AF	0356 08	ADD EDX,DWORD PTR DS:[ESI+8]	
003201B2	6A 40	PUSH 40	
003201B4	68 00100000	PUSH 1000	
003201B9	51	PUSH ECX	
003201BA	6A 00	PUSH 0	

Final, press **Ctrl + F12** and we have:

00401130	68 24134000	PUSH unpackme.00401324	
00401132	E8 EFFFFFFF	CALL unpackme.00401128	JMP to MSVBUM60.ThunRTMain
0040113A	? 0000	ADD BYTE PTR DS:[EAX],AL	
0040113C	? 0000	ADD BYTE PTR DS:[EAX],AL	
0040113E	? 0000	ADD BYTE PTR DS:[EAX],AL	
00401140	? 3000	XOR BYTE PTR DS:[EAX],AL	
00401142	? 0000	ADD BYTE PTR DS:[EAX],AL	
00401144	? 40	INC EAX	
00401145	? 0000	ADD BYTE PTR DS:[EAX],AL	
00401147	? 0000	ADD BYTE PTR DS:[EAX],AL	
00401149	? 0000	ADD BYTE PTR DS:[EAX],AL	
0040114B	? 0008	ADD BYTE PTR DS:[EAX],CL	
0040114D	? 8C65 1B	MOV WORD PTR SS:[EBP+1B],FS	
00401150	? B3 06	MOV BL,6	
00401152	? A3 459F8119	MOV DWORD PTR DS:[19819F45],EAX	

Our OEP!

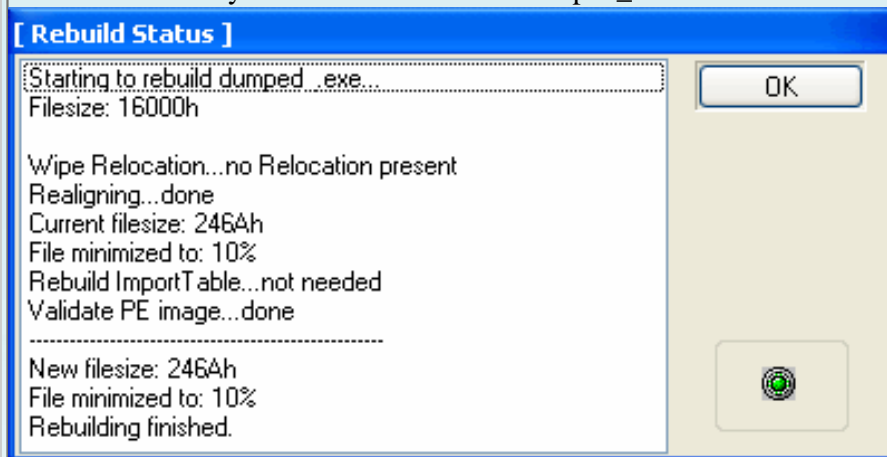
Congratulations! According OEP we found is **401130** And now we Calculate the real OEP of this unpackme by the formula:
Real OEP = OEP find in Olly-Image Base = 401130-400000 = **1130**.



All Import Functions valid.

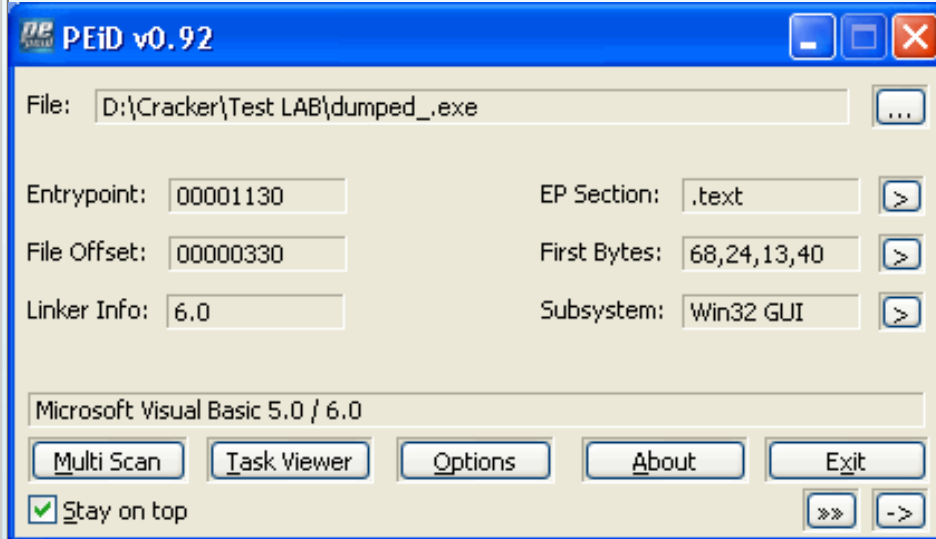
Now, click fix dump to fix the IAT **dumped.exe** file.

Use LordPE 1.4 by Y0da for rebuild our Dumped_.exe



6. Testing Our Unpacked file

Use PEiD for detect again:



Now run unpacked files. Wow, not crash.

7. Conclusion

Special thanx to **R @ dier** for this template.

My Greetz to: Deux, infinite, NVH (c), softcracker_vn, luucorp, Aaron, hhphong, R @ dier, tlandn, Computer_Angel, k3nny, Ferrari, Zombie, RCA, CTL, Moonbaby, Neitsa, JAL, LeVuHoang, 777, LeonHart , Bin ... and you ;-)!
To be continued ...

Written by [hacnho](#) (tutorial date: Sai Gon 4/5/2004)

Translated and written by: **kienmanowar**

Manual unpacking PECompact v2.38

Information	Unpacking for Newbie's
Target	Target.exe
Available	http://www.reaonline.net
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, PEid 0.93, Lord PE 1.4, Plugin Command Line, ImpRec v1.6f.
Protection	PECompact v2.38
L Evel	Beginner
Category	Manual unpacking

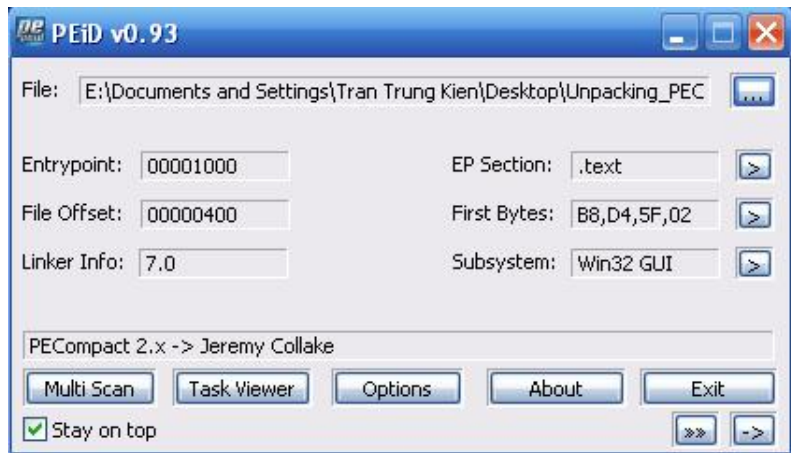
1. Introduction

This article aims to discuss how to unpack **PECompact 2:38** success through some basic steps. File for the practice has been enclosed with this article. Destination of the tutorial is targeted for Newbie when you get familiar with the Unpacking (same to me =)). Unpacking is a job quite difficult and daunting easy when we face the difficult Packer, so to get familiar with this job is no way that we must start from the example application the most, the first step in the firm to the knowledge of our strength

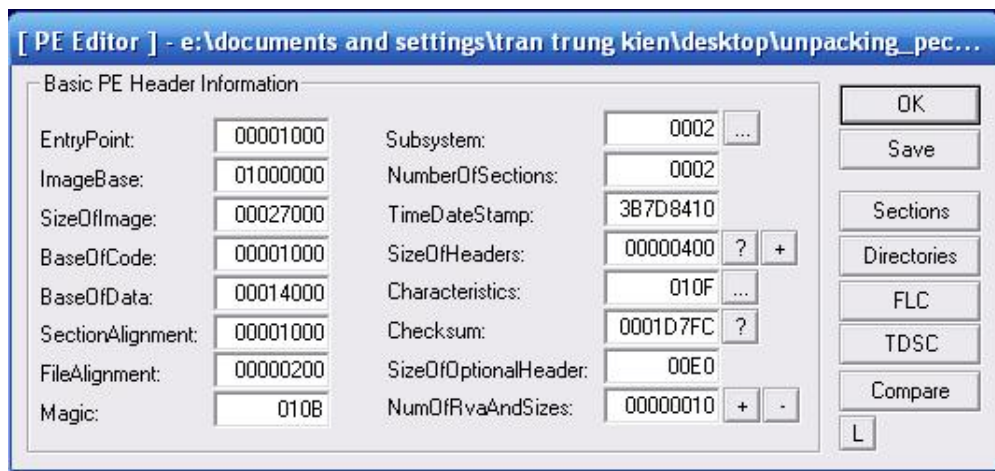
will gradually
up. And one can
at any time, so as
we will not be
any doubt before
a Packers **do**!).
Oki, let's do it!

PE 2.Detect and get info

Use **PEid of 0.93** to detect, we know the following:



Use **Lord PE 1.4** to search for more information:



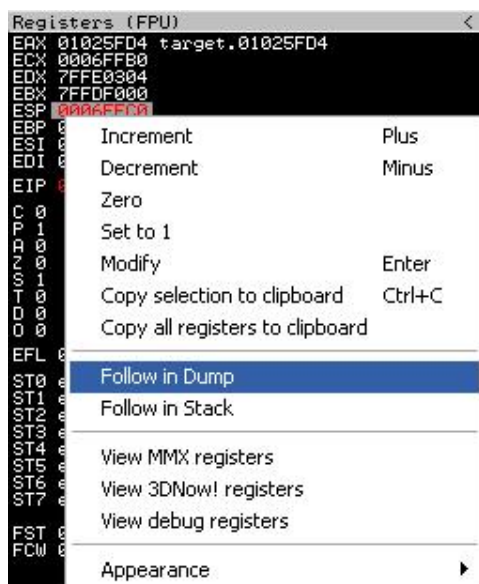
So we have been as follows: EntryPoint is: 1000, ImageBase is 1000000.

3.Finding the Original Entry Point

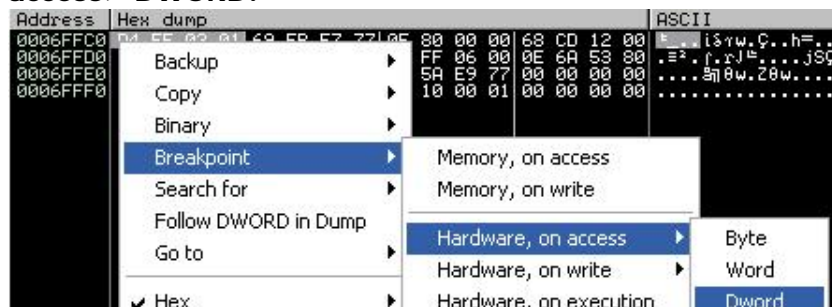
After the above information, we open and Olly Load target in Olly.Chon **No Ananlysis** and we will stop at **EntryPoint**:

Address	Hex dump	Disassembly	Comment
01001000	B8 045FD021	MOV EAX, target.01025FD4	
01001005	50	PUSH EAX	
01001006	64:FF35 000000	PUSH DWORD PTR FS:[0]	
0100100D	64:8925 000000	MOV DWORD PTR FS:[0], ESP	
01001014	33C0	XOR EAX, EAX	
01001016	8908	MOV DWORD PTR DS:[EAX], ECX	
01001018	50	PUSH EAX	
01001019	45	INC EBP	
0100101A	43	INC EBX	
0100101B	6F	OUTS DX, DWORD PTR ES:[EDI]	I/O command
0100101C	6D	INS DWORD PTR ES:[EDI], DX	I/O command
0100101D	70 61	JG SHORT target.01001080	
0100101F	637432 00	ARPL WORD PTR DS:[EDX+ESI], SI	
01001023	61	POPAD	
01001024	0B02	FILD DWORD PTR DS:[EDX]	
01001026	0E	PUSH CS	
01001027	6E	OUTS DX, BYTE PTR ES:[EDI]	I/O command
01001028	0067 00	ADD BYTE PTR DS:[EDI], AH	
0100102B	3300	XOR EAX, DWORD PTR DS:[EAX]	
0100102D	3B00	CMP EAX, DWORD PTR DS:[EAX]	
0100102F	305400 68	XOR BYTE PTR DS:[EAX+EAX+68], DL	
01001033	B9 2D92DB61	MOV ECX, 61DB922D	
01001038	64:0079 87	SAR BYTE PTR FS:[ECX-79], 1	
0100103C	E5 2C	IN EAX, 2C	I/O command

Oki, after Load in Olly finished, press F8 next 2 times. Transfer Register through window, right click on the bar and write **ESP Follow in dump**. We are as follows:



Highlight 4 bytes at **0006FFC0**, then right click and select **BreakPoint-> Hardware-on access> DWORD**:



Next, we press F9 1 times and Shift + F9 4 times. We will stop me in order **JMP EAX** jump:

Address	Hex dump	Disassembly	Comment
0102609F	FFEB	JMP NEAR EAX	target.01012475
010260A1	75 24	JNZ SHORT target.010260C7	
010260A3	0101	ADD DWORD PTR DS:[ECX], EAX	
010260A5	0000	ADD BYTE PTR DS:[EAX], AL	
010260A7	0000	ADD BYTE PTR DS:[EAX], AL	
010260A9	0000	ADD BYTE PTR DS:[EAX], AL	

Oki, at a command **0102609F** jump to OEP. Now we delete Hardware breakpoint by the **Debug-> Breakpoints Hardware-> Delete** to delete. Then we perform lenh jump at **0102609F** and we will stay in OEP. OEP What we have is **01,012,475**.

Address	Hex dump	Disassembly	Comment
01012475	6A 70	PUSH 70	
01012477	68 E0150001	PUSH target.010015E0	
0101247C	E8 47030000	JMP target.010127C8	
01012481	33DB	XOR EBX, EBX	
01012483	53	PUSH EBX	
01012484	8B3D 20100001	MOV EDI, DWORD PTR DS:[1001020]	kernel32.GetModuleHandleA
0101248A	FFD7	NEAR EDI	
0101248C	66:8138 4D5A	CMF WORD PTR DS:[EAX], 5A4D	

OEP calculation of the formula:

Real OEP = OEP find in Olly-Image Base = 01012475 - 01000000 = **00012475**.

Next press Ctrl + A to Analysis Code. We have been as follows:

Address	Hex dump	Disassembly	Comment
01012475	6A 70	PUSH 70	
01012477	68 E0150001	PUSH target.010015E0	
0101247C	E8 47030000	JMP target.010127C8	
01012481	33DB	XOR EBX, EBX	
01012483	53	PUSH EBX	
01012484	8B3D 20100001	MOV EDI, DWORD PTR DS:[1001020]	pModule => NULL
0101248A	FFD7	NEAR EDI	kernel32.GetModuleHandleA
0101248C	66:8138 4D5A	CMF WORD PTR DS:[EAX], 5A4D	GetModuleHandleA

4. Unpacked dumping our files

At **01012475**, we click and choose to **dump debugged process**. Uncheck **Rebuild Import**, click and save the dump under a name that **dumped.exe** example.

OllyDump - target.exe

Start Address: 1000000 Size: 27000 Dump

Entry Point: 1000 -> Modify: 12475 Get EIP as OEP Cancel

Base of Code: 1000 Base of Data: 14000

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
.text	0001E000	00001000	0001E000	00001000	E0000020
.rsrc	00008000	0001F000	00008000	0001F000	E0000020

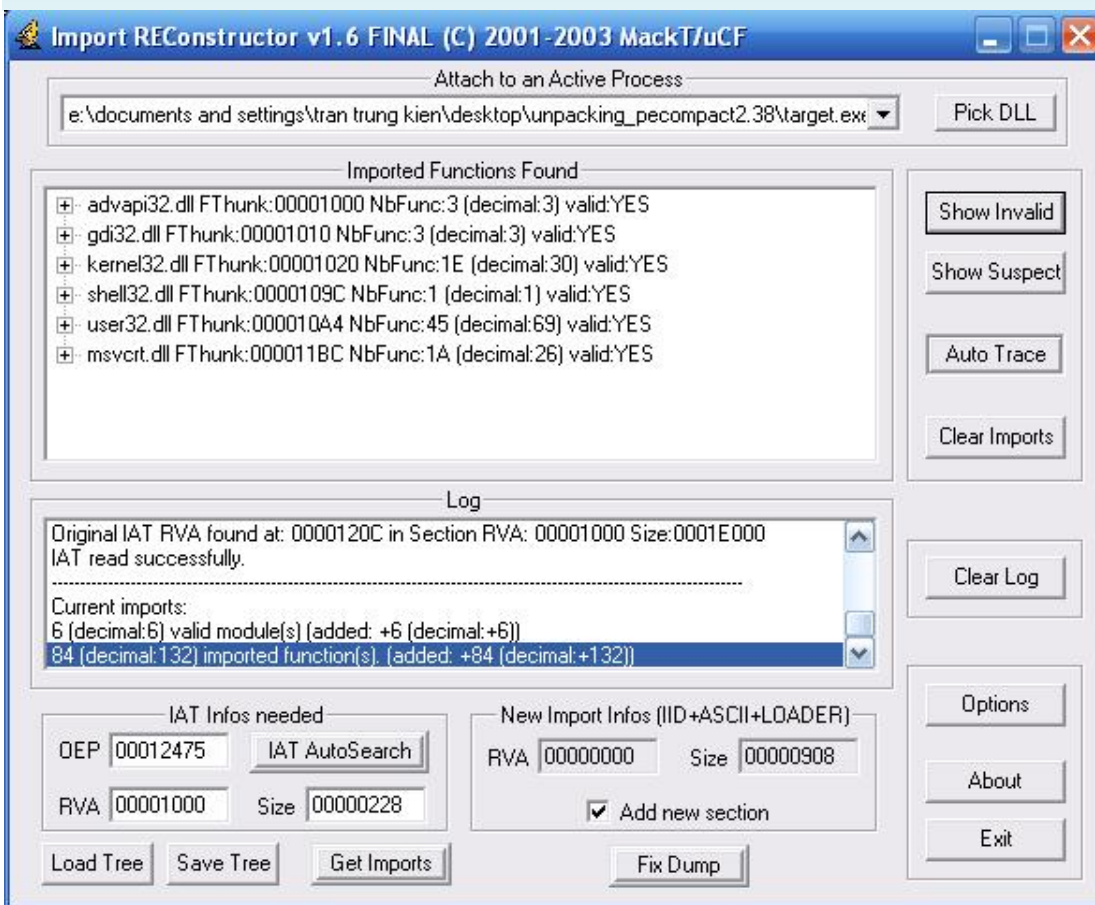
☐ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

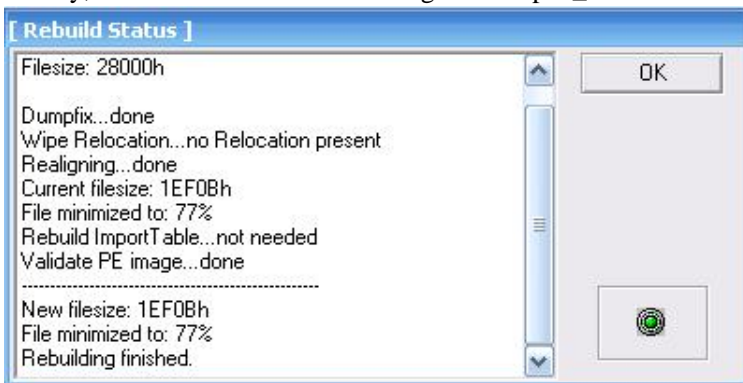
☐ Method2 : Search DLL & API name string in dumped file

5. Finding and Fixing the Address Import Table

Hold the window Olly, open **ImpRec** select list box in **target.exe Attach to an Active Process**. Enter OEP we have found the calculation of the above in, and click **Get Autosearch IAT Imports**. Then click **Show Invalid**. Keke too good not Invalid thanks at all. Finally click Fix dump file and select dumped.exe.



Finally, we used to LordPE Rebuilt again Dumped_.exe file.



6. Testing Our Unpacked file

Oki, a test file we Unpacked. Yup! It works.
Used to Detect PEid again we have been as follows:



So **PECompact v2.38** unpack was successful. Have fun:)

7. Conclusion

Greetz to: ARTEAM

To thank my family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHQCrkcr, the_Lighthouse, Hoadongnoi, Nini, tlandn, dqtlN ... all REA's members, HacNho, RongChauA, Deux all my friend, and YOU!



Written by **kienmanowar** (tutorial date: HaNoi 04/05/2005)

...: Copyright © 2005 by kienmanowar <[=-=]> REA-cRaCkErTeAm (www.reaonline.net):...

Manual Unpacking PEQuake v0.06

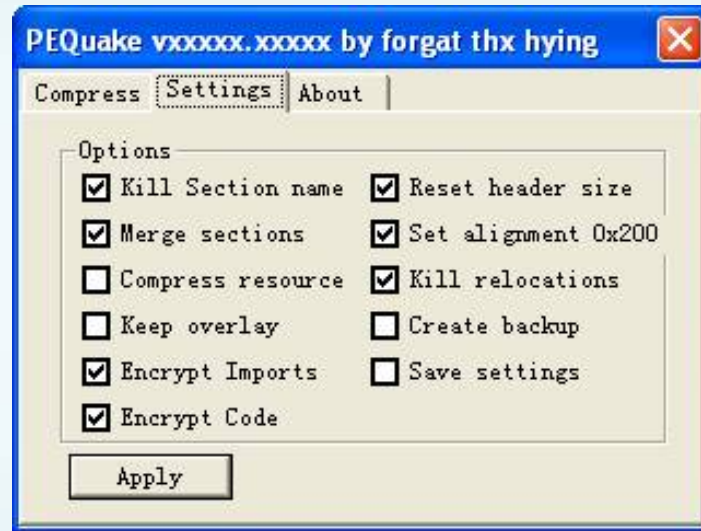
Target: PEQuake.exe (v0.06)

Tools: OllyDBG with OllyDump, CMDBar. (ImportREC, PETools if you want)

Writer: REA Trickyboy ()

I. Introduction

- Hi, sure some of you have heard the last name Packer PEQuake this, it was developed by forgat and quite small. The Option of the packer is also diverse, please review:



- The packer has Option ways of protecting it lies in the following 2 points:

+ Anti-Debugger

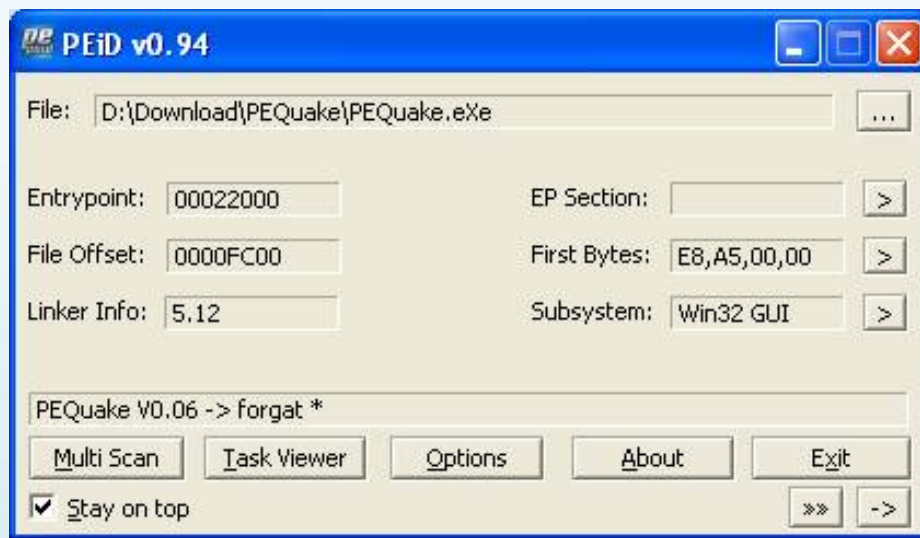
+ Encrypt IAT table.

- Tricky packer that through this touching thanks to a layer of very UnpackeMe Why Not Bar. Unpack hoài do that, but after walking on a net, also found the packer try this man. Also lượm a tut Chinese on the packer, but see tut and then too, tricky to do that should the Chinese tit blind. Finally have to unpack nghiên hours underground. Time would have to consult with you.

- We will always use the Packer PEQuake in this pack because it is by itself. J

II. Detecting

- It can be used to Detect PEiD:



- Or observe 1 billion:
- + Have LOADING gray when run:



- + Entry Point (EP) starts with a function CALL:

Address	Hex dump	Disassembly
00422000	E8 A5000000	CALL PEQuake.0042200A
00422005	2D 20020000	SUB EAX,220
0042200A	0000	ADD BYTE PTR DS:[EAX],AL
0042200C	0000	ADD BYTE PTR DS:[EAX],AL
0042200E	0000	ADD BYTE PTR DS:[EAX],AL
00422010	003D 2002002D	ADD BYTE PTR DS:[20000220],BH
00422016	2002	AND BYTE PTR DS:[EDX],AL
00422018	0000	ADD BYTE PTR DS:[EAX],AL
0042201A	0000	ADD BYTE PTR DS:[EAX],AL
0042201C	0000	ADD BYTE PTR DS:[EAX],AL
0042201E	0000	ADD BYTE PTR DS:[EAX],AL
00422020	0000	ADD BYTE PTR DS:[EAX],AL
00422022	0000	ADD BYTE PTR DS:[EAX],AL
00422024	0000	ADD BYTE PTR DS:[EAX],AL
00422026	0000	ADD BYTE PTR DS:[EAX],AL
00422028	0000	ADD BYTE PTR DS:[EAX],AL
0042202A	0000	ADD BYTE PTR DS:[EAX],AL
0042202C	0028	ADD BYTE PTR DS:[EAX],CH

- Thus it is enough to conclude it is PEQuake then. Of course, the aged PRO will detect through signature of the packer, is less tricky to do so temporarily. J

Anti-III.Bypass Debugger:

- Packer this anti-debugger by check to see programs Load from where, if explorer.exe from ko's Win is an independent in order to dance itself. Packer check Where When run, it will create an area code in memory and then check. I will remember to this region by set breakpoint (BP) in CreateThread function.
- So after Load to Olly, you enter the Command Bar "BP CreateThread," Enter.



- Press F9 to run, may have LOADING up and preparing to load. Break!

Address	Hex dump	Disassembly
7C81082F	8BFF	MOV EDI,EDI
7C810831	55	PUSH EBP
7C810832	8BEC	MOV EBP,ESP
7C810834	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
7C810837	FF75 18	PUSH DWORD PTR SS:[EBP+18]
7C81083A	FF75 14	PUSH DWORD PTR SS:[EBP+14]
7C81083D	FF75 10	PUSH DWORD PTR SS:[EBP+10]
7C810840	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]
7C810843	FF75 08	PUSH DWORD PTR SS:[EBP+08]
7C810846	6A FF	PUSH -1
7C810848	E8 D9FDFFFF	CALL kernel32.CreateRemoteThread
7C81084D	5D	POP EBP
7C81084E	C2 1800	RETN 18

- Remove BP this place. Ctrl-F9 to order RETN 18. And F8 to return, to us here:

Address	Hex dump	Disassembly
00344799	33F6	XOR ESI,ESI
0034479B	33FF	XOR EDI,EDI
0034479D	6A 01	PUSH 1
0034479F	FF95 B1634000	CALL DWORD PTR SS:[EBP+4063B1]
003447A5	0F31	RDTS
003447A7	83E0 1F	AND EAX,1F
003447AA	05 C0000000	ADD EAX,0C0
003447AF	8B08	MOV EBX,EAX
003447B1	C1E3 08	SHL EBX,8
003447B4	03D8	ADD EBX,EAX
003447B6	C1E3 08	SHL EBX,8
003447B9	03D8	ADD EBX,EAX
003447BB	C1E3 08	SHL EBX,8
003447BE	03D8	ADD EBX,EAX
003447C0	5D	POP EBP

- So check stage will take place in the Code are in the start address is 340000 & Size A000. Here we will analyze a bit to find out the first set in BP. If at this Run for more, we do not see the program up, to have full LOADING, while still reporting Olly Running:

Running

- From the failure to be either Olly Loop Where it is independent or 1 in order jumped by airplanes. 3 dance or order form is used JMP, JE, JNZ.

- JMP independent if the previous order to have a jump conditions to it. Hope is the ability to independently and in JE JNZ higher, so we easily than man. JE if independent, there is a **74 FE** OPCODE. JNZ is **75 FE**

- You drag a window onto the top. Search for Byte: 74 FE. Item not found.

- Search more of Byte: 75 FE. Found 4 orders jumped as: (Ctrl-L to the next order)

Address	Hex dump	Disassembly
00344896	- 75 FE	JNZ SHORT 00344896
00344898	60	PUSHAD
00344899	E8 00000000	CALL 0034489E
0034489E	5D	POP EBP
0034489F	81ED DC194000	SUB EBP,4019DC
003448A5	6A 00	PUSH 0
003448A7	6A 02	PUSH 2
003448A9	FF95 4A664000	CALL DWORD PTR SS:[EBP+40664A]
003448AF	93	XCHG EAX,EBX

Address	Hex dump	Disassembly
00345110	- 75 FE	JNZ SHORT 00345110
00345112	61	POPAD
00345113	BB 46520000	MOV EBX,5246
00345118	833C2B 00	CMP DWORD PTR DS:[EBX+EBP],0
0034511C	0F84 F2020000	JE 00345414
00345122	53	PUSH EBX
00345123	6A 04	PUSH 4
00345125	68 00100000	PUSH 1000
0034512A	FF342B	PUSH DWORD PTR DS:[EBX+EBP]
0034512D	6A 00	PUSH 0

Address	Hex dump	Disassembly
00345599	75 FE	JNZ SHORT 00345599
0034559B	B8 01000000	MOV EAX,1
003455A0	68 9F6F56B6	PUSH B6566F9F
003455A5	50	PUSH EAX
003455A6	E8 5D000000	CALL 00345608
003455AB	EB FF	JNZ SHORT 003455AC
003455AD	71 78	JND SHORT 00345627
003455AF	C2 5000	RETN 50
003455B2	EB D3	JNZ SHORT 003455B7
003455B4	5B	POP EBX

Address	Hex dump	Disassembly
00345EC6	75 FE	JNZ SHORT 00345EC6
00345EC8	8BD0 3A520000	MOV EDI,DWORD PTR SS:[EBP+523A]
00345ECE	03FD	ADD EDI,EBP
00345ED0	8DB5 6A460000	LEA ESI,DWORD PTR SS:[EBP+466A]
00345ED6	8B07	MOV EAX,DWORD PTR DS:[EDI]
00345ED8	68 9F6F56B6	PUSH B6566F9F
00345EDD	50	PUSH EAX
00345EDE	E8 5D000000	CALL 00345F40
00345EE3	EB FF	JNZ SHORT 00345EE4
00345EE5	71 78	JND SHORT 00345F5F

- Of course it is set in all 4 BP commands. Then F9 Run Break it in what was the Okie. It is and try this:

Address	Hex dump	Disassembly
003450FD	71 EB	JND SHORT 003450EA
003450FF	EB FA	JNZ SHORT 003450FB
00345101	EB 83	JNZ SHORT 00345086
00345103	EE AD	OUT DX,AL
00345104	0C AD	OR AL,0AD
00345106	25 5F5F5F5F	AND EAX,5F5F5F5F
0034510B	3D 4558504C	CMP EAX,4C505845
00345110	75 FE	JNZ SHORT 00345110
00345112	61	POPAD
00345113	BB 46520000	MOV EBX,5246
00345118	833C2B 00	CMP DWORD PTR DS:[EBX+EBP],0
0034511C	0F84 F2020000	JE 00345414
00345120	50	PUSH EAX

- Now that new orders for **Cmp EAX, 4C505845** quite special:

Reg	Value
EAX	4C4C4F00
ECX	kernel32.7C80C710
EDX	00344A30 ASCII "llydbg.exe"
EBX	00000054
ESP	00A6FF98
EBP	FFF42EC2
ESI	00344A32 ASCII "ydbg.exe"
EDI	00000BC0
EIP	00345110

- If the load from **Explorer.exe**, the obvious Byte here will match the order in Byte Cmp on. From that failure to find out how this command JNZ jump faster by any Search Command: **Cmp EAX, 4C505845** is enough.

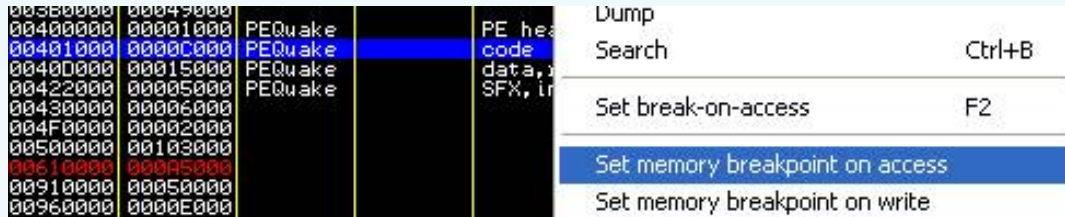
- So now it's too easy to bypass it. Here are Intergrity ko you should check that an NOP it. But it gently for Change flag Z = 1 to do it is to skip:

Address	Hex dump	Disassembly
003450FD	71 EB	JND SHORT 003450EA
003450FF	EB FA	JNZ SHORT 003450FB
00345101	EB 83	JNZ SHORT 00345086
00345103	EE AD	OUT DX,AL
00345104	0C AD	OR AL,0AD
00345106	25 5F5F5F5F	AND EAX,5F5F5F5F
0034510B	3D 4558504C	CMP EAX,4C505845
00345110	90	NOP
00345111	90	NOP
00345112	61	POPAD
00345113	BB 46520000	MOV EBX,5246
00345118	833C2B 00	CMP DWORD PTR DS:[EBX+EBP],0

- The bypass is part of our Anti-Debugger of PEQuake then đây. Now click Run will see the program run lickerish.
- But do not want to do from the beginning for you here, and one through the IV-Find OEP.

IV.Find OEP

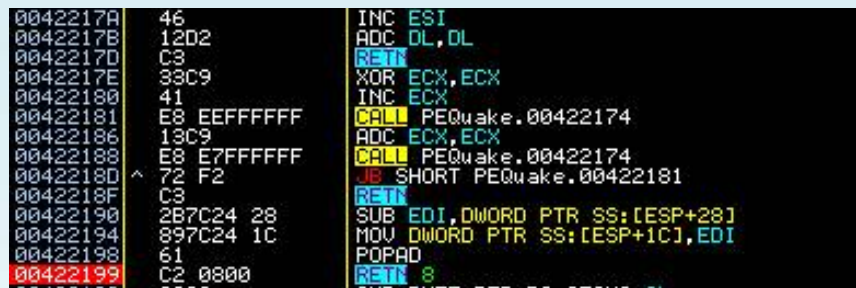
- Unpack the packer is, we always need a key 2 OEP β IAT table and they are usually the packer encoding. Tricky course as you know 2 things on what is.
- How to Find OEP is temporary, and tricky mò mam should not be optimal.
- Here we still follow the traditional waiting Packer unpack completely Code of the program, access to the Code section and Bread for the OEP. So we open Memory Map, Alt-M, "[Set breakpoint on memory access \(MBOA\)](#)" in the code section:



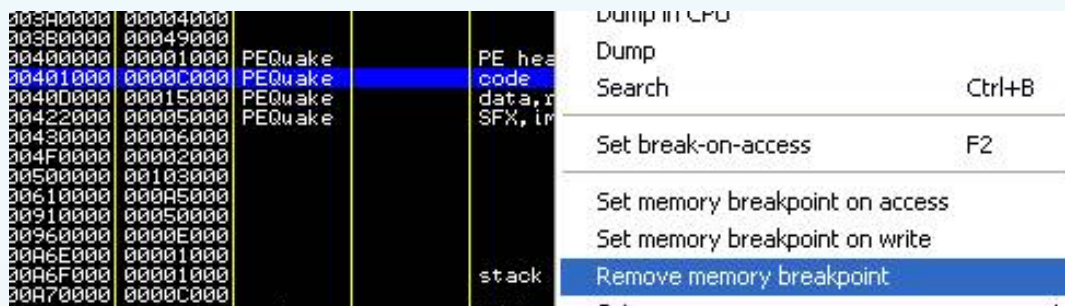
- F9 to run more, LOADING full bar, and Bread:



- At this time not to OEP, under an order JNB jumped up. If we continue F9 will trace each loop a long time. Need some trick to bypass tip. Hehe ... tricky but J
- Course for the last loop, BP set for just below it, then F9 is too soon (remember to remove the MBOA)
- But after several times mò mam, see also some tricky loop below, and get 1 BP placement help us go faster. Pull down a tí, over 2 orders RETN, and stopped in order RETN 3, BP set it up:



- Do not forget to delete MBOA other small, because now we're in the code section. If MBOA remain there, all the time RUN F9's like we were a Trace F8 order - are the region access to this Code.



- Now, F9, Bread:

Address	Hex dump	Disassembly
00422180	72 F2	JB SHORT PEQuake.00422181
0042218F	C3	RETN
00422190	2B7C24 28	SUB EDI,DWORD PTR SS:[ESP+28]
00422194	897C24 1C	MOV DWORD PTR SS:[ESP+1C],EDI
00422198	61	POPAD
00422199	C2 0800	RETN 8
0042219C	2800	SUB BYTE PTR DS:[EAX],AL
0042219F	0000	ADD BYTE PTR DS:[EAX],AL

- Set the MBOA to the code section for finding OEP still unknown when the program go back to. Remember to remove RETN 8 BP in the other. F9 any Break:

Address	Hex dump	Disassembly
003453F3	F3:A5	REP MOVS DWORD PTR ES:[EDI],DWORD PTR D
003453F5	59	POP ECX
003453F6	83E1 03	AND ECX,3
003453F9	F3:A4	REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:
003453FB	5E	POP ESI
003453FC	53	PUSH EBX
003453FD	68 00000000	PUSH 0000
00345402	6A 00	PUSH 0
00345404	54	PUSH ESI

- Look through the window Register:

Registers (FPU)	
EAX	0000C000
ECX	00003000
EDX	7C30E674 ntdll.KiFastSystemCa
EBX	00005246
ESP	0006FFD0
EBP	003444A8
ESI	00A70000
EDI	00401000 PEQuake.00401000
EIP	003453F3
C 0	ES 0023 32bit 0(FFFFFFFF)
P 1	CS 001B 32bit 0(FFFFFFFF)
A 0	SS 0023 32bit 0(FFFFFFFF)
Z 0	DS 0023 32bit 0(FFFFFFFF)
S 0	FS 003B 32bit 7FFDE000(FFF)
T 0	GS 0000 NULL
D 0	
O 0	LastErr ERROR_SUCCESS (00000000)
EFL	00010206 (NO,NB,NE,A,NS,PE,GE)
ST0	empty 0.0000000156560127370e-4
ST1	empty 0.2626643080556337280e-4
ST2	empty 0.0206914874782672400e-4
ST3	empty +UNORM 0002 01010000 000
ST4	empty -UNORM B7DB 00000047 FFF
ST5	empty -UNORM FD0C 006994E8 000
ST6	empty -UNORM 5D48 77070494 000

- ECX = 3000, if your patience will run F9 feedback in order to reduce the 3000 value gradually. He he .. Tricky offense F9 button of your use F8 trick tip is the last one ... ECX = 0 J

- Now back to F9, Bread:

Address	Hex dump	Disassembly	Comment
00345E7B	8907	MOV DWORD PTR DS:[EDI],EAX	kernel32.ReadFile
00345E7D	5A	POP EDX	
00345E7E	0FB642 FF	MOVZX EAX,BYTE PTR DS:[EDX-1]	
00345E82	03D0	ADD EDX,EAX	
00345E84	42	INC EDX	
00345E85	83C7 04	ADD EDI,4	
00345E88	59	POP ECX	
00345E89	E2 CA	LOOPD SHORT 00345E55	
00345E8B	E9 EFFCFFFF	JMP 00345B7F	
00345E90	64:A1 30000000	MOV EAX,DWORD PTR FS:[30]	
00345E96	85C0	TEST EAX,EAX	
00345E98	78 08	JS SHORT 00345EA2	
00345E9A	0FB648 02	MOVZX ECX,BYTE PTR DS:[EAX+2]	

- There is something special was shown. Remember prior to the OEP, the value of the table must be IAT Write to Memory truoc.O here we see a value eyehole. Find the memory location is also very important to rebuild Import, so look through the Register:

```
Registers (FPU)
EAX 7C80180E kernel32.ReadFile
ECX 7C919AEB ntdll.7C919AEB
EDX 7C97C0D8 ntdll.7C97C0D8
EBX 0000F2CE
ESP 00A6FFB0
EBP 003444AB
ESI 7C800000 kernel32.7C800000
EDI 0040B03C PEQuake.0040B03C
EIP 00345E7B
C 0 ES 0023 32bit 0(FFFFFFFF)
P 0 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL
D 0
```

- There is a value only under section code of the program, **EDI = 0040B03C** (401,000 <0040B03C <0040D000). Hopefully that is the address on the value of the IAT is to write. Follow us in dump in EDI:



- Look through the window dump:

Address	Hex dump	ASCII
0040B03C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B04C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B05C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B06C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B07C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B08C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B09C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B0AC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B0BC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B0CC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

- Still have not found anything, but please press F9 to any one, we still Break command in the dump and found in Memory:

Address	Hex dump	ASCII
0040B03C	0E 18 80 7C 00 00 00 00 00 00 00 00 00 00 00 00	#!@!.....
0040B04C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B05C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B06C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B07C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B08C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B09C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B0AC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B0BC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040B0CC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

- Yes is stored value IAT, code calls ReadFile function was to Write. If some F9 again, we'll see Byte of the following functions are to write:

Address	Hex dump	Disassembly
00345E7B	8907	MOV DWORD PTR DS:[EDI],EAX
00345E7D	5A	POP EDX
00345E7E	0FB642 FF	MOVZX EAX, BYTE PTR DS:[EDX-1]
00345E82	03D0	ADD EDX, EAX
00345E84	42	INC EDX
00345E85	83C7 04	ADD EDI, 4
00345E88	59	POP ECX
00345E89	^ E2 CA	LOOPE SHORT 00345E55
00345E8B	^ E9 EFFCFFFF	JMP 00345B7F
00345E90	64:A1 30000000	MOV EAX, DWORD PTR FS:[30]
00345E96	85C0	TEST ECX, ECX
00345E98	78 08	JS SHORT 00345EA2
00345E9A	0FB648 02	MOVZX ECX, BYTE PTR DS:[EAX+2]
00345E9E	E3 28	JECXZ SHORT 00345EC8
00345EA0	EB FE	JMP SHORT 00345EA0
00345EA2	E0	BUSH_F0X

00345E8B	^	E9 EFFFFF	JMP 00345B7F
00345E90		64:A1 30000000	MOV EAX,DWORD PTR FS:[30]
00345E96		85C0	TEST EAX,EAX
00345E98	✓	78 08	JG SHORT 00345EA2
00345E9A		0FB648 02	MOVZX ECX,BYTE PTR DS:[EAX+2]
00345E9E	✓	E3 28	JECXZ SHORT 00345EC8
00345EA0	-	EB FE	JNE SHORT 00345EA0
00345EA2	✓	50	PUSH FAX

[illegible]

Jump is NOT taken

Address	Hex	dump
0040B03C	0E 18 80 70	
0040B04C	81 9A 80 70	
0040B05C	BE 3E 82 70	
0040B06C	A2 CA 81 70	
0040B07C	00 00 00 00	
0040B08C	AE E2 D4 77	
0040B09C	A4 C2 D5 77	
0040B0AC	D4 54 D4 77	
0040B0BC	B1 84 D4 77	

Disassemble menu options:

- Long
- Float
- Disassemble
- Special
- Export table
- Appearance
- Signed decimal
- Unsigned decimal
- Hex
- Address
- Address with ASCII
- Address with UNICODE

file:///C:/RCE%20Unpacking%20eBook%20[Tra...Li]/Manual%20Unpacking%20PEQuake%20v0.htm (8 of 22) [1/9/2009 9:45:30 LithiumLi]

0040B03C	7C80180E	kernel32.ReadFile
0040B040	7C822BB7	kernel32.WritePrivateProfileStringA
0040B044	7C810F9F	kernel32.WriteFile
0040B048	7C809B14	kernel32.VirtualFree
0040B04C	7C809A81	kernel32.VirtualAlloc
0040B050	7C810DA6	kernel32.SetFilePointer
0040B054	7C90311B	ntdll.RtlZeroMemory
0040B058	7C809B77	kernel32.CloseHandle
0040B05C	7C823EBE	kernel32.GetPrivateProfileIntA
0040B060	7C830053	kernel32.CopyFileA
0040B064	7C80B529	kernel32.GetModuleHandleA
0040B068	7C810C8F	kernel32.GetFileSize
0040B06C	7C81CAA2	kernel32.ExitProcess
0040B070	7C801A24	kernel32.CreateFileA
0040B074	00000000	
0040B078	7CA0FE44	SHELL32.ShellExecuteA
0040B07C	00000000	
0040B080	77D4D4DE	user32.ShowWindow
0040B084	77D48C06	user32.SetTimer
0040B088	77D660D5	user32.SetDlgItemTextA
0040B08C	77D4F29F	user32.SendMessageA

- Pull up, IAT **40B000** Start at:

Address	Value	Comment
0040AFFC	00000000	
0040B000	500B1500	comctl32.InitCommonControls
0040B004	00000000	
0040B008	763B311E	comdlg32.GetOpenFileNameA
0040B00C	00000000	
0040B010	77F16DC0	GDI32.BitBlt

- Pull-down search IAT End:

Address	Value	Comment
0040B0AC	77D4C4D4	user32.EnableWindow
0040B0B0	77D588E1	user32.DialogBoxParamA
0040B0B4	77D65EA0	user32.CreateDialogParamA
0040B0B8	77D589A8	user32.CheckDlgButton
0040B0BC	77D4B4B1	user32.BeginPaint
0040B0C0	77D567A8	user32.LoadBitmapA
0040B0C4	00000000	
0040B0C8	00000000	
0040B0CC	00000000	
0040B0D0	00000000	
0040B0D4	00000000	
0040B0D8	00000000	

- What are IAT Length = **C0**

- Okie, and that information useful for the Import rebuild. So after IAT write to memory. Hope that it will access to OEP. We put the BP immediately jumped this command. Continue set MBOA in code section. F9 - Break:

Address	Hex dump	Disassembly
00346187	8958 FC	MOV DWORD PTR DS:[EAX-4],EBX
0034618A	83C7 08	ADD EDI,8
0034618D	^ E9 44FDFFFF	JMP 00345ED6
00346192	68 9F6F56B6	PUSH B6566F9F
00346197	50	PUSH EAX
00346198	E8 5D000000	CALL 003461FA
0034619D	^ EB FF	JMP SHORT 0034619E
0034619F	^ 71 78	JNC SHORT 00346219
003461A1	C2 5000	RETN 50
003461A4	^ EB D3	JMP SHORT 00346179

- Hic, OEP has not yet, but it do not break in the offset in the code of the program. This Code is the area that packer created to check the present. Now make sure it duties decrypt something. At first come, tricky đành blindly press F9 "to" look forward to a time when it is to OEP. And it has to really (do not you do the housework, so raw here, Section V to continue the fast)

Address	Hex dump	Disassembly	Comment
00406AFA	8D35 C5C14000	LEA ESI,DWORD PTR DS:[40C1C5]	<-- OEP
00406B00	8D3D E0C74000	LEA EDI,DWORD PTR DS:[40C7E0]	
00406B06	B9 10000000	MOV ECX,10	
00406B0B	F3:A4	REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:	
00406B0D	C705 C4C74000	MOV DWORD PTR DS:[40C7C4],-0C	
00406B17	C705 D4C74000	MOV DWORD PTR DS:[40C7D4],109	
00406B21	C605 DBC74000	MOV BYTE PTR DS:[40C7DB],1	
00406B28	6A 00	PUSH 0	
00406B2A	E8 45220000	CALL PEQuake.00408D74	
00406B2F	A3 34C54000	MOV DWORD PTR DS:[40C534],EAX	
00406B34	6A 00	PUSH 0	
00406B36	68 536B4000	PUSH PEQuake.00406B53	

- What is the complete Find OEP. One to the next.

V. Analyze decrypt IAT Jump & Find OEP Faster

- At the one we have is that the number "n" - number ko bit before, we must have a general way. What do you press F9 on the tricky subject is difficult to make the last item IV nhé. But one principle is to observe the hours Register:

Registers (FPU)	
EAX	00408D86 PEQuake.00408D86
ECX	00000000
EDX	00349AA3
EBX	FFF3FD0F
ESP	00A6FFB8
EBP	003444AB
ESI	00349AA7
EDI	00349AA7
EIP	00346187
C	1 ES 0023 32bit 0(FFFFFFFF)
P	0 CS 001B 32bit 0(FFFFFFFF)
A	1 SS 0023 32bit 0(FFFFFFFF)
Z	0 DS 0023 32bit 0(FFFFFFFF)
S	1 FS 003B 32bit 7FFDE000(FFF)
T	0 GS 0000 NULL
O	0
O	0 LastErr ERROR_CLASS_ALREADY_
EFL	00010293 (NO,B,NE,BE,S,PO,L,LF)

- Fortunately **EAX = 408D86** may be offset under the program code. We try to Goto it, Ctrl-G: 408D86:



- I'm here:

Address	Hex dump	Disassembly
00408D86	90	NOP
00408D87	E8 00000000	CALL PEQuake.00408D8C
00408D8C	90	NOP
00408D8D	E8 00000000	CALL PEQuake.00408D92
00408D92	90	NOP
00408D93	E8 00000000	CALL PEQuake.00408D98
00408D98	90	NOP
00408D99	E8 00000000	CALL PEQuake.00408D9E
00408D9E	90	NOP
00408D9F	E8 00000000	CALL PEQuake.00408DA4
00408DA4	90	NOP
00408DA5	E8 00000000	CALL PEQuake.00408DAA
00408DAA	90	NOP
00408DAB	E8 00000000	CALL PEQuake.00408DB0
00408DB0	90	NOP
00408DB1	E8 00000000	CALL PEQuake.00408DB6

- Looking back kỹ the CALL command is called to each other in order from top down. It lằng nhách to do? So here is the function CALL has been encrypted. Now a new F9 again, EAX is the address of one of the functions on CALL.

We again Goto 408D86 to address at this time:

Address	Hex dump	Disassembly
00408D74	90	NOP
00408D75	E8 00000000	CALL PEQuake.00408D7A
00408D7A	90	NOP
00408D7B	E8 00000000	CALL PEQuake.00408D80
00408D80	90	NOP
00408D81	E8 8FFDF3FF	CALL 00348B15
00408D86	90	NOP
00408D87	E8 00000000	CALL PEQuake.00408D8C
00408D8C	90	NOP
00408D8D	E8 00000000	CALL PEQuake.00408D92
00408D92	90	NOP

- There are 1 small change, in order CALL called on to do 408D86 which is 348B15. That is after F9, has been the change in bytes, and a call CALL other appear. It is called to address 34xxxx area remember that packer created, this should certainly related to decrypt. It decrypt what? We again opted to mò mam. Looking through the Register:

```

EAX 00408DAA PEQuake.00408DAA
ECX 00000000
EDX 00349AA3
EBX FFF3F068
ESP 00A6FFB8
EBP 003444AB
ESI 00348B15
EDI 0034308F
EIP 00346187

C 1 ES 0023 32bit 0(FFFFFFFF)
P 0 CS 001B 32bit 0(FFFFFFFF)
A 1 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 1 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL
O 0
O 0 LastErr ERROR_CLASS_ALREADY_
EFL 00010293 (NO,B,NE,BE,S,PO,L,L

```

- **ESI** are kept in the code offset by packer, **EDI** is still their, it demonstrated the new changes in the value and use, in our Follow dom and dump it into memory dump:

Address	Value	Comment
00349AAE	80408DAA	
00349AB3	00408040	PEQuake.00408040
00349AB7	80408DAA	
00349AB8	00408044	PEQuake.00408044
00349ABF	80408D9E	
00349AC3	00408048	PEQuake.00408048
00349AC7	80408D98	
00349AC8	0040804C	PEQuake.0040804C
00349ACF	80408D92	
00349AD3	00408050	PEQuake.00408050
00349AD7	80408D8C	
00349AD8	00408054	PEQuake.00408054
00349ADF	80408D5C	
00349AE3	00408058	PEQuake.00408058
00349AE7	80408D80	
00349AEB	0040805C	PEQuake.0040805C
00349AEF	80408D62	
00349AF3	00408060	PEQuake.00408060
00349AF7	80408D7A	

- Ah ... há you do not see any? Sure i remember ha, IAT we find now is: **40B000 - 40B0C0**. So the value here is the part where Address save IAT. Byte and the other, as 80408DAA - **80000000** = 408DAA is the address of 1 in the function CALL lãng nhach there.

- So now the thinking here is that the code in a call to the IAT.

- Normally, a program with the command to IAT:

JMP DWORD PTR DS: [xxxxxxx]

- This cost is 2 Byte 6 **FF 25**-byte opcode JMP by Byte and 4 - **xx xx xx xx** is a need to address the region to jump.

- Compare we see here packer has encryption JMP order that a command NOP 1 Byte and a command CALL 5 Byte. So our job is to counteract Patch 1 Byte NOP CALL + 5 bytes t h a t JMP 6 bytes. But it is in the

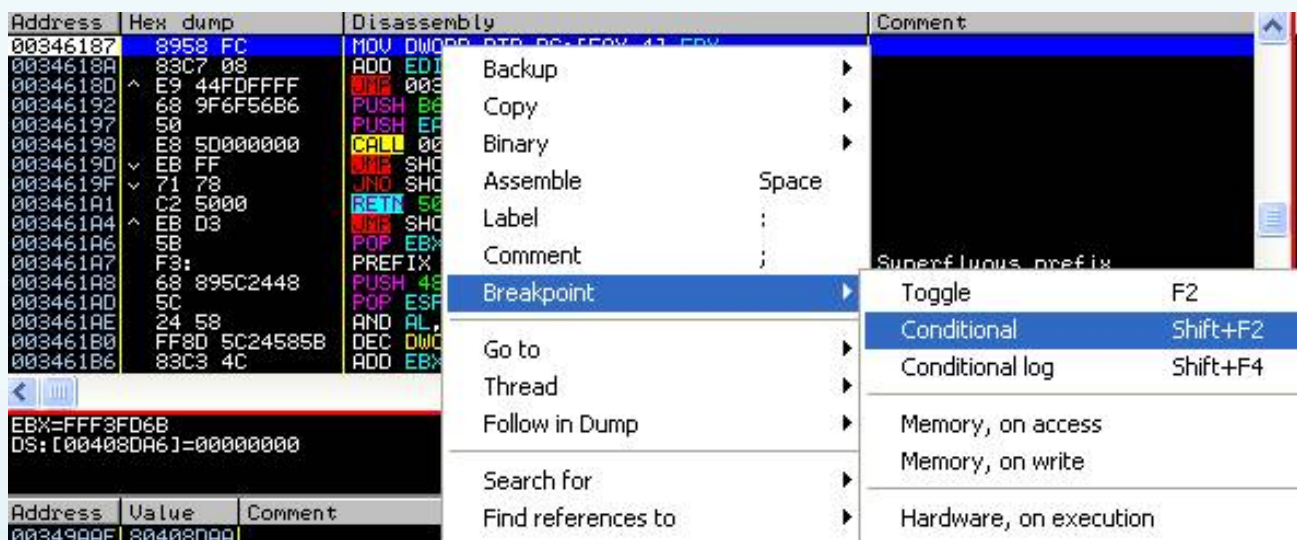
following categories, tricky time back to find OEP. The number "n" above, can be solved this place.

- Remember that every time F9 break in top position, a command CALL been changed to 348B15 CALL. Position the CALL command is in order from top down. Of each address found in the EAX, the CALL command will change the address **EAX-6**. In previous Byte 6. If you click "n-1" times, you will see the value of EAX held before OEP is to address the command CALL lãng nhach final + 6.

- So very simply, we put the MBOA, are now in order Break:

Address	Hex dump	Disassembly
00346187	8958 FC	MOV DWORD PTR DS:[EAX-4],EBX
0034618A	83C7 08	ADD EDI,8
0034618D	E9 44FDFFFF	JMP 00345ED6
00346192	68 9F6F56B6	PUSH B6566F9F
00346197	50	PUSH EAX
00346198	E8 5D000000	CALL 003461FA
0034619D	EB FF	JMP SHORT 0034619E
0034619F	71 78	JNO SHORT 00346219
003461A1	C2 5000	RET 50
003461A4	EB D3	JMP SHORT 00346179

- BP has set the conditions are satisfied, click right here (or Shift-F2):



- For the EAX == Address CALL lãng nhach final +6.? How do I find this last CALL? Of course to do after "n" F9 times, like the lãng nhach evil. There is a general search.

- From the CALL function, pulling down the bottom to find CALL:

Address	Hex dump	Disassembly
00408E46	90	NOP
00408E47	E8 00000000	CALL PEQuake.00408E4C
00408E4C	90	NOP
00408E4D	E8 00000000	CALL PEQuake.00408E52
00408E52	90	NOP
00408E53	E8 00000000	CALL PEQuake.00408E58
00408E58	90	NOP
00408E59	E8 00000000	CALL PEQuake.00408E5E
00408E5E	CC	INT3
00408E5F	CC	INT3
00408E60	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]
00408E64	83F8 02	CMP EAX,2
00408E67	7D 06	JGE SHORT PEQuake.00408E6F
00408E69	B8 64000000	MOV EAX,64

- The address is the need to find the address just below the CALL command. As said "each of the address found in the EAX, the CALL command will change the address EAX-6." You sure know ha. That address is the need to find **408E5E**.

- But how do find this optimization, because the table can Küç IAT scattered, including continuous together, such as encryption, and this will CALL years together, including continuous, CALL and final course i just have to pull down is found. UnpackMe is typical of why the CALL 2 years together.

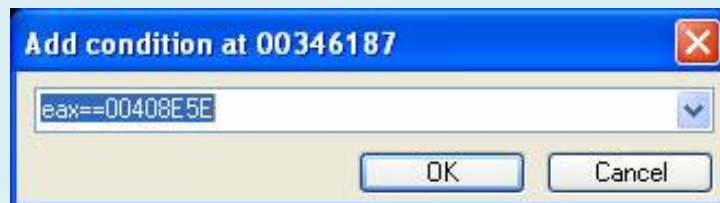
- There is a better way, back to the dump Memory:

Address	Value	Comment
00349AAF	80408DAA	
00349AB3	0040B040	PEQuake.0040B040
00349AB7	80408DA4	
00349AB8	0040B044	PEQuake.0040B044
00349ABF	80408D9E	
00349AC3	0040B048	PEQuake.0040B048
00349AC7	80408D98	
00349ACB	0040B04C	PEQuake.0040B04C
00349ACF	80408D92	
00349AD3	0040B050	PEQuake.0040B050
00349AD7	80408D8C	
00349ADB	0040B054	PEQuake.0040B054
00349ADF	80408D5C	
00349AE3	0040B058	PEQuake.0040B058
00349AE7	80408D80	
00349AEB	0040B05C	PEQuake.0040B05C
00349AEF	80408D62	
00349AF3	0040B060	PEQuake.0040B060
00349AF7	80408D7A	

- At 804xxxxx values, you have to find the greatest value here. This case we found:

80408E5E - $80000000 = 408E5E$

- Okie, hope you understand J. Back fill the box BP Conditional:



- OK, one of F9 (deleted MBOA) I still stop here:

Address	Hex dump	Disassembly
00346187	8958 FC	MOV DWORD PTR DS:[EAX-4],EBX
0034618A	83C7 08	ADD EDI,8
0034618D	E9 44FDFFFF	JMP 00345ED6
00346192	68 9F6F56B6	PUSH B6566F9F
00346197	50	PUSH EAX
00346198	E8 5D000000	CALL 003461FA
0034619D	EB FF	JMP SHORT 0034619E
0034619F	71 78	JNE SHORT 00346219
003461A1	C2 5000	RET 50
003461A4	EB D3	JMP SHORT 00346179

- But EAX = 00408E5E. What is past "n-1" times. Now delete this BP seats. MBOA set the code section, and F9 break at OEP darling:

Address	Hex dump	Disassembly	Comment
00406AFA	8035 C5C14000	LEA ESI,DWORD PTR DS:[40C1C5]	<-- OEP
00406B00	803D E0C74000	LEA EDI,DWORD PTR DS:[40C7E0]	
00406B06	B9 10000000	MOV ECX,10	
00406B08	F3:A4	REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:	
00406B0D	C705 C4C74000	MOV DWORD PTR DS:[40C7C4],-0C	
00406B17	C705 D4C74000	MOV DWORD PTR DS:[40C7D4],109	
00406B21	C605 DBC74000	MOV BYTE PTR DS:[40C7D8],1	
00406B28	6A 00	PUSH 0	
00406B2A	E8 45220000	CALL PEQuake.00408D74	
00406B2F	03 34C54000	MOV DWORD PTR DS:[40C534],EAX	

- This is how i find OEP optimal, but perhaps better F9 "n" times nữa you? Okie. How tired the rest hen. Then to the next.

VI. Jump to repair IAT

- When you unpack, if they have any memory area to keep Real IAT (IAT News ko encrypted), OEP, which means we can dump the Import and rebuild it. Đồng But back CALL lẫn nhạch any, Goto **408D86** :

Address	Hex dump	Disassembly
00408D86	90	NOP
00408D87	E8 89FDF3FF	CALL 00348B15
00408D8C	90	NOP
00408D8D	E8 83FDF3FF	CALL 00348B15
00408D92	90	NOP
00408D93	E8 7DFDF3FF	CALL 00348B15
00408D98	90	NOP
00408D99	E8 77FDF3FF	CALL 00348B15
00408D9E	90	NOP
00408D9F	E8 71FDF3FF	CALL 00348B15
00408DA4	90	NOP
00408DA5	E8 6BFDF3FF	CALL 00348B15
00408DAA	90	NOP
00408DAB	E8 65FDF3FF	CALL 00348B15
00408DB0	90	NOP
00408DB1	E8 5FFDF3FF	CALL 00348B15
00408DB6	90	NOP

- Yes the entire order before CALL **CALL** are registered **00348B15**, packer to decrypt the IAT program to call.
- But as the analysis in part V. You only need to change the patch NOP Byte 1 + 5 bytes Call à JMP 6 bytes.
- Now we return to dump Memory:

Address	Value	Comment
00349AAE	80408DAA	
00349AB3	0040B040	PEQuake.0040B040
00349AB7	0040B044	PEQuake.0040B044
00349AB8	0040B044	PEQuake.0040B044
00349ABF	80408D9E	
00349AC3	0040B048	PEQuake.0040B048
00349AC7	80408D98	
00349AC8	0040B04C	PEQuake.0040B04C
00349ACF	80408D92	
00349AD3	0040B050	PEQuake.0040B050
00349AD7	80408D8C	
00349AD8	0040B054	PEQuake.0040B054
00349ADF	80408D5C	
00349AE3	0040B058	PEQuake.0040B058
00349AE7	80408D80	
00349AEB	0040B05C	PEQuake.0040B05C
00349AEF	80408D62	
00349AF3	0040B060	PEQuake.0040B060
00349AF7	80408D7A	

- I see with each address has 2 parts, one is the Byte address CALL lãng nhach + **80000000**, which is the address of a memory region containing IAT.
- Get the 1 case highlighted above, you may think, we Patch NOP + CALL in the **JMP 408DAA [40B040]**. But please remember: "address of each found in EAX, the CALL command will change the address EAX-6"
- JMP also a mind that, with each command should JMP Patch, Patch will be at less than 6. If you do not believe, you just try to patch addresses do change, will be the surviving command NOP + CALL first and last legacy of CALL. So in the above example from 408DAA:

Address	Hex dump	Disassembly
00408DA4	90	NOP
00408DA5	E8 6BFDF3FF	CALL 00348B15
00408DAA	90	NOP
00408DAB	E8 65FDF3FF	CALL 00348B15
00408DB0	90	NOP
00408DB1	E8 5FFDF3FF	CALL 00348B15
00408DB6	90	NOP

- Move up 6 bytes, even in the NOP command:

Address	Hex dump	Disassembly
00408DA4	90	NOP
00408DA5	E8 6BFDF3FF	CALL 00348B15
00408DAA	90	NOP
00408DAB	E8 65FDF3FF	CALL 00348B15
00408DB0	90	NOP
00408DB1	E8 5FFDF3FF	CALL 00348B15
00408DB6	90	NOP

- Patch to JMP [40B040]:

Address	Hex dump	Disassembly	Comment
00408DA4	EF 25 40 80 40 00	JMP DWORD PTR DS:[40B040]	kernel32.WritePrivateProfileS
00408DAA	90	NOP	
00408DAB	E8 65 FDF3 FF	CALL 00348B15	
00408DB0	90	NOP	
00408DB1	E8 5F FDF3 FF	CALL 00348B15	
00408DB6	90	NOP	
00408DB7	E8 59 FDF3 FF	CALL 00348B15	
00408DBC	90	NOP	
00408DBD	E8 53 FDF3 FF	CALL 00348B15	
00408DC2	90	NOP	
00408DC3	E8 4D FDF3 FF	CALL 00348B15	

- To complete the patch, you pull up to find where to start in the memory dump:

Address	Value	Comment
00349A9F	00416574	PEQuake.00416574
00349AA7	80408086	
00349AAB	0040B03C	PEQuake.0040B03C
00349AB3	0040B040	PEQuake.0040B040
00349AB7	804080A4	
00349ABB	0040B044	PEQuake.0040B044
00349ABF	8040809E	
00349AC3	0040B048	PEQuake.0040B048
00349AC7	80408098	
00349ACB	0040B04C	PEQuake.0040B04C
00349ACF	80408092	
00349AD3	0040B050	PEQuake.0040B050
00349AD7	8040808C	
00349ADB	0040B054	PEQuake.0040B054
00349ADF	8040805C	
00349AE3	0040B058	PEQuake.0040B058
00349AE7	80408080	

- Down the end:

Address	Value	Comment
00349BC8	0040B01C	PEQuake.0040B01C
00349BCF	80408E3A	
00349BD3	0040B020	PEQuake.0040B020
00349BD7	80408E40	
00349BDB	0040B024	PEQuake.0040B024
00349BDF	80408E46	
00349BE3	0040B028	PEQuake.0040B028
00349BE7	80408E4C	
00349BEB	0040B02C	PEQuake.0040B02C
00349BEF	80408E52	
00349BF3	0040B030	PEQuake.0040B030
00349BF7	80408E58	
00349BFB	0040B034	PEQuake.0040B034
00349BFF	80408E5E	
00349C03	0040B078	PEQuake.0040B078
00349C0B	00000000	
00349C0F	00000000	
00349C13	00000000	

- So start from 349AA7 address, the address 408D86 - 6 = 408D80, you patch into JMP [40B30C]. You Patch formula by which to address 349BFF is completed.

Then dump-and rebuild IAT. (If OEP has to do is to use dump LordPE)

Jump to VII.Repair IAT quickly by Coding

- Hi hi, finished reading the above, you will have the village is now ... *"This patch may do all àh? Wá many."* Yes, the right hand each patch on the line as everyone oái. For example **PEQuake.exe** with this patch is only the number of small orders JMP. What's also UnpackMe to 84 lines. Hehe .. but first unpack it is very tricky patch hand. Only when writing this tricky new thinking otherwise.

- Of course the readers aged PRO khúc has to think of how now more quickly. Tricky think that speed to be improved only.

- I resolve problems with the patch itself through Olly help you (are content because the CPU).

- The kĩ see you again, we found that only works with 3 main value, 1 address in the dump Memory, and 2 parts Byte memory of it.

Address	Value	Comment
00349A9F	00416574	PEQuake.00416574
00349AA7	80408086	
00349AAB	0040B03C	PEQuake.0040B03C
00349AB3	0040B040	PEQuake.0040B040
00349AB7	804080A4	
00349ABB	0040B044	PEQuake.0040B044
00349ABF	8040809E	
00349AC3	0040B048	PEQuake.0040B048
00349AC7	80408098	
00349ACB	0040B04C	PEQuake.0040B04C
00349ACF	80408092	
00349AD3	0040B050	PEQuake.0040B050
00349AD7	8040808C	
00349ADB	0040B054	PEQuake.0040B054
00349ADF	8040805C	
00349AE3	0040B058	PEQuake.0040B058
00349AE7	80408080	

- From this address, from increased, to take part in Byte 1 to determine the need to address patch, and get to the byte in the 2 to address identified areas should remember to JMP. Patch until at last they stopped.
- So when the OEP, we find a region CODE available (in fact do not need to be the same):

Address	Hex dump	Disassembly
0040B400	0000	ADD BYTE PTR DS:[EAX],AL
0040B402	0000	ADD BYTE PTR DS:[EAX],AL
0040B404	0000	ADD BYTE PTR DS:[EAX],AL
0040B406	0000	ADD BYTE PTR DS:[EAX],AL
0040B408	0000	ADD BYTE PTR DS:[EAX],AL
0040B40A	0000	ADD BYTE PTR DS:[EAX],AL
0040B40C	0000	ADD BYTE PTR DS:[EAX],AL
0040B40E	0000	ADD BYTE PTR DS:[EAX],AL
0040B410	0000	ADD BYTE PTR DS:[EAX],AL
0040B412	0000	ADD BYTE PTR DS:[EAX],AL
0040B414	0000	ADD BYTE PTR DS:[EAX],AL
0040B416	0000	ADD BYTE PTR DS:[EAX],AL
0040B418	0000	ADD BYTE PTR DS:[EAX],AL
0040B41A	0000	ADD BYTE PTR DS:[EAX],AL
0040B41C	0000	ADD BYTE PTR DS:[EAX],AL
0040B41E	0000	ADD BYTE PTR DS:[EAX],AL

- At the time of the OEP, this place is empty, so tricky borrow temporary accommodation for creative ngenh relatives. Address **40B400**
- As the above analysis, we only work with 3 main value, 1 value is only a memory region, 2 value is left in the area remember that, so we need to use to record to 2 and more 1 to write back. Here tricky use EAX, EBX, EDX.
- Use which must preserve the way, pushing it stack up now:

Address	Hex dump	Disassembly
0040B400	50	PUSH EAX
0040B401	53	PUSH EBX
0040B402	55	PUSH EDI
0040B403	0000	ADD BYTE PTR DS:[EAX],AL
0040B405	0000	ADD BYTE PTR DS:[EAX],AL

- More secure, we cleaned 3 write this: There are 3 ways to make a clean record 1:
MOV EAX, 0 β cost of bytes 5
SUB EAX, EAX β take 2 bytes
XOR EAX, EAX β take 2 bytes.

-Of course it should take only 2 bytes, select tricky XOR:

Address	Hex dump	Disassembly
0040B400	50	PUSH EAX
0040B401	53	PUSH EBX
0040B402	52	PUSH EDX
0040B403	33C0	XOR EAX,EAX
0040B405	330B	XOR EBX,EBX
0040B407	33D2	XOR EDX,EDX
0040B409	0000	ADD BYTE PTR DS:[EAX],AL
0040B40B	0000	ADD BYTE PTR DS:[EAX],AL
0040B40D	0000	ADD BYTE PTR DS:[EAX],AL
0040B40F	0000	ADD BYTE PTR DS:[EAX],AL
0040B411	0000	ADD BYTE PTR DS:[EAX],AL
0040B413	0000	ADD BYTE PTR DS:[EAX],AL

- We start at Start in the memory: **349AA7**, it should MOV EAX:

Address	Hex dump	Disassembly
0040B400	50	PUSH EAX
0040B401	53	PUSH EBX
0040B402	52	PUSH EDX
0040B403	33C0	XOR EAX,EAX
0040B405	330B	XOR EBX,EBX
0040B407	33D2	XOR EDX,EDX
0040B409	B8 A79A3400	MOV EAX,349AA7
0040B40E	90	NOP
0040B40F	0000	ADD BYTE PTR DS:[EAX],AL
0040B411	0000	ADD BYTE PTR DS:[EAX],AL
0040B413	0000	ADD BYTE PTR DS:[EAX],AL
0040B415	0000	ADD BYTE PTR DS:[EAX],AL
0040B417	0000	ADD BYTE PTR DS:[EAX],AL

- We need the 1 byte in the address in EAX, so we MOV content in the memory address in EAX to EBX. (*)

Address	Hex dump	Disassembly
0040B400	50	PUSH EAX
0040B401	53	PUSH EBX
0040B402	52	PUSH EDX
0040B403	33C0	XOR EAX,EAX
0040B405	330B	XOR EBX,EBX
0040B407	33D2	XOR EDX,EDX
0040B409	B8 A79A3400	MOV EAX,349AA7
0040B40E	8B18	MOV EBX,DWORD PTR DS:[EAX]
0040B410	90	NOP
0040B411	0000	ADD BYTE PTR DS:[EAX],AL
0040B413	0000	ADD BYTE PTR DS:[EAX],AL
0040B415	0000	ADD BYTE PTR DS:[EAX],AL
0040B417	0000	ADD BYTE PTR DS:[EAX],AL
0040B419	0000	ADD BYTE PTR DS:[EAX],AL

- The byte to be less **80000000**, and then subtracting the **6** new address should correct patch. In ASM, we do make sure the area between value and 1 randomly. So to write temporary borrowing to keep EDX 80000000, then less:

Address	Hex dump	Disassembly
0040B400	50	PUSH EAX
0040B401	53	PUSH EBX
0040B402	52	PUSH EDX
0040B403	33C0	XOR EAX,EAX
0040B405	330B	XOR EBX,EBX
0040B407	33D2	XOR EDX,EDX
0040B409	B8 A79A3400	MOV EAX,349AA7
0040B40E	8B18	MOV EBX,DWORD PTR DS:[EAX]
0040B410	B9 00000030	MOV EDX,80000000
0040B415	2BD4	SUB EBX,EDX
0040B417	83FB 06	SUB EBX,6
0040B41A	0000	ADD BYTE PTR DS:[EAX],AL
0040B41C	0000	ADD BYTE PTR DS:[EAX],AL
0040B41E	0000	ADD BYTE PTR DS:[EAX],AL
0040B420	0000	ADD BYTE PTR DS:[EAX],AL
0040B422	0000	ADD BYTE PTR DS:[EAX],AL
0040B424	0000	ADD BYTE PTR DS:[EAX],AL

- Of course can get 80,000,000 - 6, to MOV EDX, and except for the neat, but I understand why they just write as the wall.

- So the command line **SUB EBX, 6** EBX has contained the correct address should patch. Now we start to patch the area with a memory address in EBX.

- Recall we need to patch NOP CALL + à JMP. Opcode is FF25, so if the dump from the area, the island is the byte is 25FF. EDX borrow temporarily keep this value:

0040B415	2B0A	SUB EBX,EDX
0040B417	83EB 06	SUB EBX,6
0040B41A	BA FF250000	MOV EDX,25FF
0040B41F	0000	ADD BYTE PTR DS:[EAX],AL
0040B421	0000	ADD BYTE PTR DS:[EAX],AL
0040B423	0000	ADD BYTE PTR DS:[EAX],AL

- Then they MOV region to address memory in EBX:

0040B410	BA 00000080	MOV EDX,80000000
0040B415	2B0A	SUB EBX,EDX
0040B417	83EB 06	SUB EBX,6
0040B41A	BA FF250000	MOV EDX,25FF
0040B41F	8913	MOV DWORD PTR DS:[EBX],EDX
0040B421	0000	ADD BYTE PTR DS:[EAX],AL
0040B423	0000	ADD BYTE PTR DS:[EAX],AL

- Region in memory before MOV EBX is OPCODE by NOP + CALL command. MOV after the command above, we have to rewrite Byte 2 FF 25. Now we add to the record Byte address to complete the IAT. The Byte is located in the region 2 in memory address in EAX.

To get the Byte this, we need to increase EAX 4 is completed. Due to the byte address the 1-byte address of part 2 of 4 bytes. Then we mov position it to the next byte of FF 25 in the EBX. News EBX must increase Byte 2:

0040B415	2B0A	SUB EBX,EDX
0040B417	83EB 06	SUB EBX,6
0040B41A	BA FF250000	MOV EDX,25FF
0040B41F	8913	MOV DWORD PTR DS:[EBX],EDX
0040B421	8B50 04	MOV EDX,DWORD PTR DS:[EAX+4]
0040B424	8953 02	MOV DWORD PTR DS:[EBX+2],EDX
0040B427	0000	ADD BYTE PTR DS:[EAX],AL
0040B429	0000	ADD BYTE PTR DS:[EAX],AL
0040B42B	0000	ADD BYTE PTR DS:[EAX],AL
0040B42D	0000	ADD BYTE PTR DS:[EAX],AL
0040B42F	0000	ADD BYTE PTR DS:[EAX],AL
0040B431	0000	ADD BYTE PTR DS:[EAX],AL
0040B433	0000	ADD BYTE PTR DS:[EAX],AL
0040B435	0000	ADD BYTE PTR DS:[EAX],AL

- Must use EDX as intermediaries for ASM ko work directly between the 2 regions in mind. So we have a complete command Patch jumped to 1 address.

- EAX does remain START address, you increase it to point to address the next address next month to 8 Byte **EAX+8**.

Address	Hex dump	Disassembly
0040B410	BA 00000080	MOV EDX,80000000
0040B415	2B0A	SUB EBX,EDX
0040B417	83EB 06	SUB EBX,6
0040B41A	BA FF250000	MOV EDX,25FF
0040B41F	8913	MOV DWORD PTR DS:[EBX],EDX
0040B421	8B50 04	MOV EDX,DWORD PTR DS:[EAX+4]
0040B424	8953 02	MOV DWORD PTR DS:[EBX+2],EDX
0040B427	83C0 08	ADD EAX,8
0040B42A	0000	ADD BYTE PTR DS:[EAX],AL
0040B42C	0000	ADD BYTE PTR DS:[EAX],AL
0040B42E	0000	ADD BYTE PTR DS:[EAX],AL
0040B430	0000	ADD BYTE PTR DS:[EAX],AL
0040B432	0000	ADD BYTE PTR DS:[EAX],AL

- After that period (*) are repeated. So it is a command to jump back up to create a loop. But loop should have standing. You see when EAX held last address, the patch is complete stop. But we increased the EAX+8 after Patch complete command jump 1. Therefore, for comparison when stopped, we must compare with last address + 8. If EAX is not with the values that make the process on. So we Patch to:

0040B407	83C0	ADD EAX,EDX
0040B409	B8 A79A3400	MOV EAX,349AA7
0040B40E	8B13	MOV EBX,DWORD PTR DS:[EAX]
0040B410	BA 00000080	MOV EDX,80000000
0040B415	2B0A	SUB EBX,EDX
0040B417	83EB 06	SUB EBX,6
0040B41A	BA FF250000	MOV EDX,25FF
0040B41F	8913	MOV DWORD PTR DS:[EBX],EDX
0040B421	8B50 04	MOV EDX,DWORD PTR DS:[EAX+4]
0040B424	8953 02	MOV DWORD PTR DS:[EBX+2],EDX
0040B427	83C0 08	ADD EAX,8
0040B42A	3D 079C3400	CMP EAX,349C07
0040B42F	75 00	JNZ SHORT PEQuake.0040B40E
0040B431	0000	ADD BYTE PTR DS:[EAX],AL
0040B433	0000	ADD BYTE PTR DS:[EAX],AL

- Address End is here:

```

00349BCB 0040B01C PEQuake.0040B01C
00349BCF 80408E3A
00349BD3 0040B020 PEQuake.0040B020
00349BD7 80408E40
00349BD8 0040B024 PEQuake.0040B024
00349BDF 80408E46
00349BE3 0040B028 PEQuake.0040B028
00349BE7 80408E4C
00349BEB 0040B02C PEQuake.0040B02C
00349BEF 80408E52
00349BF3 0040B030 PEQuake.0040B030
00349BF7 80408E58
00349BFB 0040B034 PEQuake.0040B034
00349BFF 80408E5E
00349C03 0040B078 PEQuake.0040B078
00349C07 00000000
00349C0B 00000000
00349C0F 00000000
00349C13 00000000

```

- Call, then returned value before creating this code for 3 record EAX, EBX, EDX order against evacuation from stack:

```

0040B41A 6A FF250000 MOV EAX,25FF
0040B41F 8913 MOV DWORD PTR DS:[EBX],EDX
0040B421 8B50 04 MOV EDX,DWORD PTR DS:[EAX+4]
0040B424 8953 02 MOV DWORD PTR DS:[EBX+2],EDX
0040B427 83C0 08 ADD EAX,8
0040B42A 3D 079C3400 CMP EAX,349C07
0040B42F ^ 75 DD JNZ SHORT PEQuake.0040B40E
0040B431 58 POP EDI
0040B432 5B POP EBX
0040B433 59 POP EAX
0040B434 0000 ADD BYTE PTR DS:[EAX],AL
0040B436 0000 ADD BYTE PTR DS:[EAX],AL
0040B438 0000 ADD BYTE PTR DS:[EAX],AL
0040B43A 0000 ADD BYTE PTR DS:[EAX],AL
0040B43C 0000 ADD BYTE PTR DS:[EAX],AL

```

- To the beautiful, tricky creating a command to jump when the OEP completed this patch, OEP = **406AFA**:

```

0040B42F ^ 75 DD JNZ SHORT PEQuake.0040B40E
0040B431 58 POP EDI
0040B432 5B POP EBX
0040B433 59 POP EAX
0040B434 ^ E9 C1B6FFFF JMP PEQuake.00406AFA
0040B439 0000 ADD BYTE PTR DS:[EAX],AL
0040B43B 0000 ADD BYTE PTR DS:[EAX],AL
0040B43D 0000 ADD BYTE PTR DS:[EAX],AL

```

- The need to do after creating the Patch Set New Origin is at the top of the patch:

Address	Hex dump	Disassembly	Comment
0040B3FE	0000	ADD BYTE PTR DS:[EAX],AL	
0040B400	50	PUSH EAX	
0040B401	53	PUSH EBX	Backup
0040B402	52	PUSH EDI	
0040B403	33C0	XOR EAX,EAX	Copy
0040B405	330B	XOR EBX,EBX	
0040B407	3302	XOR EDI,EDI	Binary
0040B409	B8 A79A3400	MOV EAX,349AA7	Undo selection Alt+BkSp
0040B40E	8B18	MOV EBX,DWORD PTR DS:[EAX]	
0040B410	BA 00000030	MOV EDI,30000000	Assemble Space
0040B415	2B0A	SUB EBX,EDI	
0040B417	83EB 06	SUB EBX,6	Label
0040B41A	BA FF250000	MOV EDI,25FF	Comment
0040B41F	8913	MOV DWORD PTR DS:[EBX],EDX	
0040B421	8B50 04	MOV EDX,DWORD PTR DS:[EAX+4]	
0040B424	8953 02	MOV DWORD PTR DS:[EBX+2],EDX	
0040B427	83C0 08	ADD EAX,8	Breakpoint
			Run trace
			New origin here Ctrl+Gray *

- Set at the end of BP, at about JMP OEP:

```

0040B432 5B POP EBX
0040B433 59 POP EAX
0040B434 ^ E9 C1B6FFFF JMP PEQuake.00406AFA
0040B439 0000 ADD BYTE PTR DS:[EAX],AL

```


- Now make sure that only a single BP here, please clear out even BP, MBOA previously. Then press F9. After Patch, break it in BP:

Address	Hex dump	Disassembly
0040B42F	75 00	JNZ SHORT PEQuake.0040B40E
0040B431	5A	POP EDX
0040B432	5B	POP EBX
0040B433	58	POP EAX
0040B434	E9 C1B6FFFF	JMP PEQuake.00406AFA
0040B439	0000	ADD BYTE PTR DS:[EAX],AL
0040B43B	0000	ADD BYTE PTR DS:[EAX],AL

- Now back to address CALL lăng nhách, see the results:

Address	Hex dump	Disassembly	Comment
00408D56	FF25 58B04000	JMP DWORD PTR DS:[40B058]	kernel32.CloseHandle
00408D5C	FF25 60B04000	JMP DWORD PTR DS:[40B060]	kernel32.CopyFileA
00408D62	FF25 70B04000	JMP DWORD PTR DS:[40B070]	kernel32.CreateFileA
00408D68	FF25 60B04000	JMP DWORD PTR DS:[40B06C]	kernel32.ExitProcess
00408D6E	FF25 68B04000	JMP DWORD PTR DS:[40B068]	kernel32.GetFileSize
00408D74	FF25 64B04000	JMP DWORD PTR DS:[40B064]	kernel32.GetModuleHandleA
00408D7A	FF25 50B04000	JMP DWORD PTR DS:[40B05C]	kernel32.GetPrivateProfileInt
00408D80	FF25 30B04000	JMP DWORD PTR DS:[40B03C]	kernel32.ReadFile
00408D86	FF25 54B04000	JMP DWORD PTR DS:[40B054]	ntdll.RtlZeroMemory
00408D8C	FF25 50B04000	JMP DWORD PTR DS:[40B050]	kernel32.SetFilePointer
00408D92	FF25 40B04000	JMP DWORD PTR DS:[40B04C]	kernel32.VirtualAlloc
00408D98	FF25 48B04000	JMP DWORD PTR DS:[40B048]	kernel32.VirtualFree
00408D9E	FF25 44B04000	JMP DWORD PTR DS:[40B044]	kernel32.WriteFile
00408DA4	FF25 40B04000	JMP DWORD PTR DS:[40B040]	kernel32.WritePrivateProfileString
00408DAA	FF25 60B04000	JMP DWORD PTR DS:[40B0BC]	uSer32.BeginPaint
00408DB0	FF25 88B04000	JMP DWORD PTR DS:[40B0B8]	uSer32.CheckDlgButton
00408DB6	FF25 64B04000	JMP DWORD PTR DS:[40B0B4]	uSer32.CreateDialogParamA

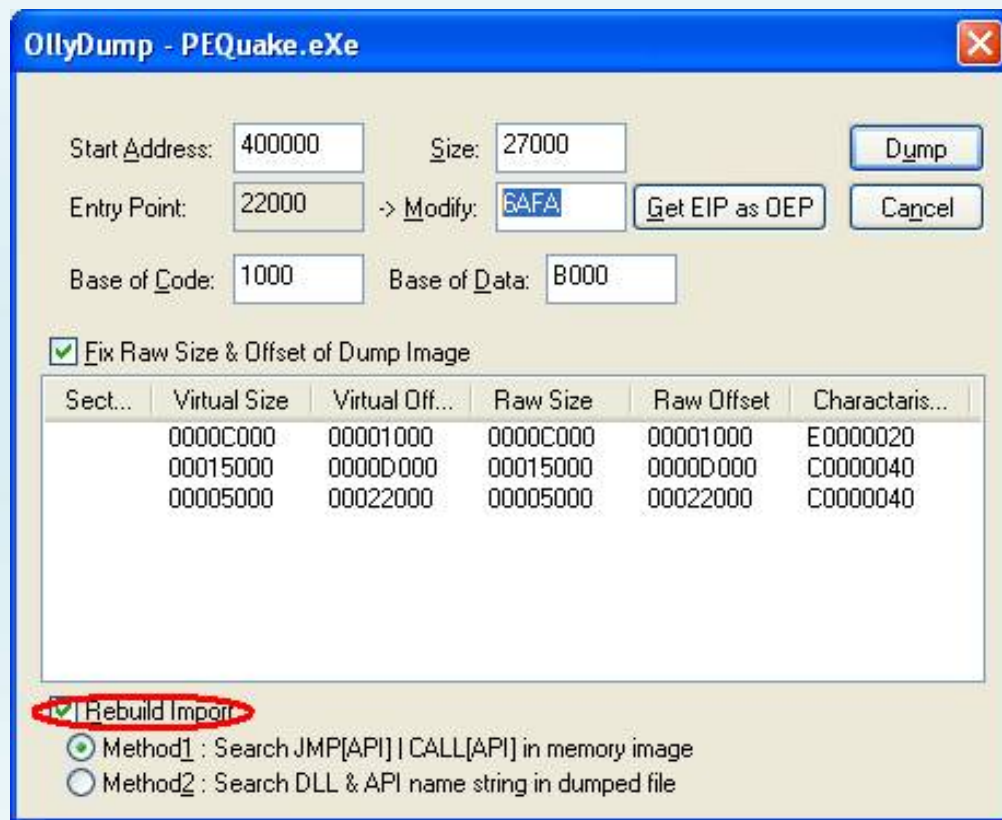
- The entire order was JMP Patch to correct its position, including survival, including excessive. So the code was effective and help us reduce mỗi hands Patch several times. Now i have to dump rubbish, you can Undo the Selection code patch is now on its original form (not to undo the table JMP à IAT)

Address	Hex dump	Disassembly
0040B400	0000	ADD BYTE PTR DS:[EAX],AL
0040B402	0000	ADD BYTE PTR DS:[EAX],AL
0040B404	0000	ADD BYTE PTR DS:[EAX],AL
0040B406	0000	ADD BYTE PTR DS:[EAX],AL
0040B408	0000	ADD BYTE PTR DS:[EAX],AL
0040B40A	0000	ADD BYTE PTR DS:[EAX],AL
0040B40C	0000	ADD BYTE PTR DS:[EAX],AL
0040B40E	0000	ADD BYTE PTR DS:[EAX],AL
0040B410	0000	ADD BYTE PTR DS:[EAX],AL
0040B412	0000	ADD BYTE PTR DS:[EAX],AL
0040B414	0000	ADD BYTE PTR DS:[EAX],AL
0040B416	0000	ADD BYTE PTR DS:[EAX],AL
0040B418	0000	ADD BYTE PTR DS:[EAX],AL
0040B41A	0000	ADD BYTE PTR DS:[EAX],AL
0040B41C	0000	ADD BYTE PTR DS:[EAX],AL
0040B41E	0000	ADD BYTE PTR DS:[EAX],AL

- Back OEP, New Origin here:

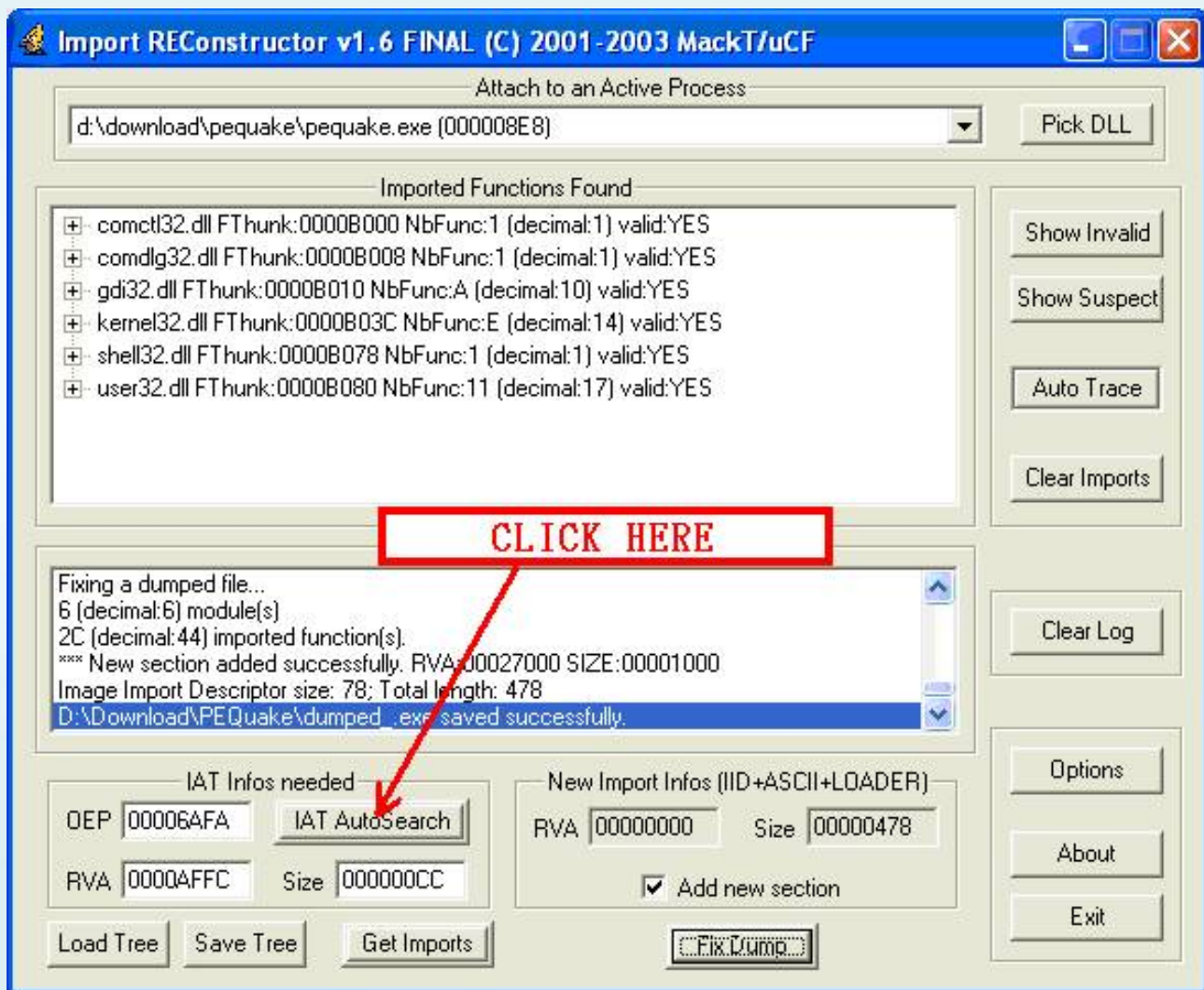
Address	Hex dump	Disassembly	Comment
00406AFA	8D35 C5C14000	LEA ESI,DWORD PTR DS:[40C1C5]	<-- OEP
00406B00	8D3D E0C74000	LEA EDI,DWORD PTR DS:[40C7E0]	
00406B06	B9 10000000	MOV ECX,10	
00406B0B	F3:A4	REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:	
00406B0D	C705 C4C74000	MOV DWORD PTR DS:[40C7C4],-0C	
00406B17	C705 D4C74000	MOV DWORD PTR DS:[40C7D4],109	
00406B21	C605 DBC74000	MOV BYTE PTR DS:[40C7DB],1	
00406B28	6A 00	PUSH 0	
00406B2A	E8 45220000	CALL PEQuake.00408D74	JMP to kernel32.GetModuleHar
00406B2F	A3 34C54000	MOV DWORD PTR DS:[40C534],EAX	
00406B34	6A 00	PUSH 0	
00406B36	68 536B4000	PUSH PEQuake.00406B53	
00406B3B	6A 00	PUSH 0	
00406B3D	68 00C04000	PUSH PEQuake.0040C000	
00406B42	FF35 34C54000	PUSH DWORD PTR DS:[40C534]	
00406B48	E8 6F220000	CALL PEQuake.00408DBC	
00406B4D	50	PUSH EAX	

- Dump and rebuild quickly with OllyDump:



- Remember to Check Import rebuild if you do not need to use ImportREC. The function of OllyDump to rebuild IAT has a long length of time will take a long time.

- So, if Length IAT long, you dump in OllyDump (uncheck rebuild Import) or PETools (ko lordPE used here), then use ImportREC fix IAT:



- Okie, the code is basic, but you want to write script or Code Patch tool to optional Memory Process.
- There is a bit tricky because of long lines trying to explain the line where the Code of suffering Newbie. As tricky when read by tut-aged foreign hate the patch is finished and then put it xó, wear readers learn that noob as tricky first must always understand. Hic hic ...

VIII.Ending

- Phu ... do you read bít tired but not tricky to write dactylogram always ne, this chat should be walking hand pain wá oi, in more rapid typing wá J . This is the first tricky self-analysis a small Packer and Repair Code JMP to IAT. As the level of reading and writing Code also poor, only know a few simple commands, if the seats do not expect the well-aged Pro only teach more. Now only rest tired .. Bye.

Thanks for all your help REA was tricky in the study.
Thanks to Oleh Yuschuk for your OllyDBG.
Thanks to forgat for your packer.
Thanks to Chinese cyclotron for your tut. L
And you for reading my document.

Tutorials hacnho # 5

Manual unpacking PE-shield v0.25 template by koncool and R @ dier.

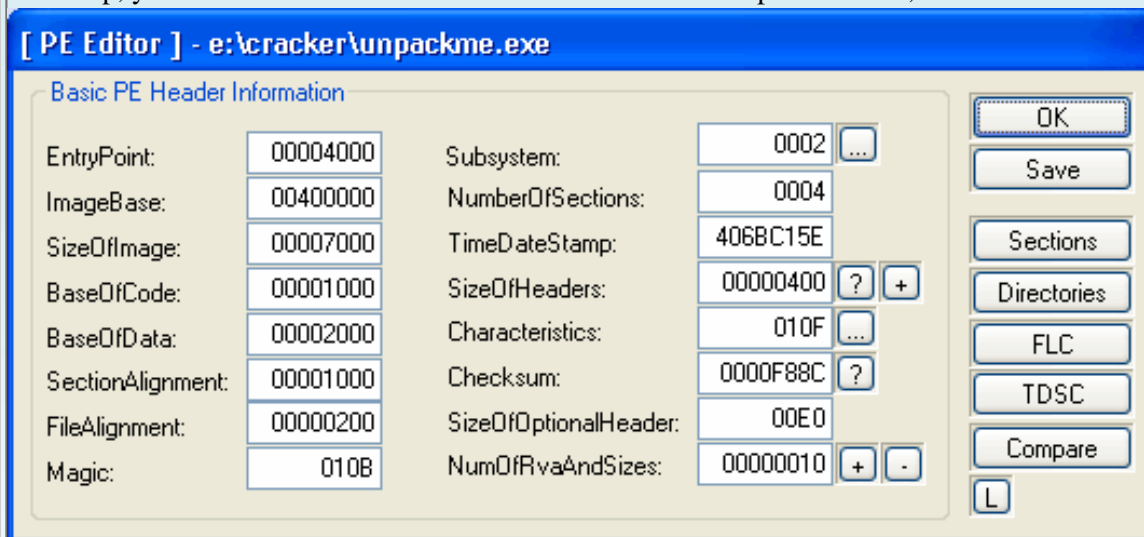
Information	Unpacking for Newbie's
Target	Unpackme.exe
Available	http://nhandan.info/hacnho/tuts/unpackme5_tuts.zip
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, Lord PE 1.4, PESniffer 3.2b.
Protection	PE-shield v0.25 by ANAKiN [DaVinci]
L Evel	Standard
Category	Manual unpacking

1. Introduction

Hi all, up to this time, I was written recv 4 tutorials and a lot of cheers from you. For this reason, I write the 5th continue tutorials. In this tut, I will be ways for the introduction unpacking PE-shield v0.25 by **ANAKiN [DaVinci]**. Tuts only for the beginner. If you are a bro. Please do not mock my tuts ... You can contact me for help and improve my tuts.

2. Getting Started

First step, you have to find some info from this PE software. Open Lord PE, PE Editor choose. And we have:



[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
PESHIELD	00001000	00001000	00000400	00000200	C0000040
PESHIELD	00002000	00001000	00000600	00000200	C0000040
PESHIELD	00003000	00001000	00000800	00000200	C0000040
ANAKIN2K	00004000	00003000	00000A00	00001600	C0000040

EP: 4000, The value of flags this case is not needed, Image Base is always 400000, Import Table: 0000 and size is 00.

3. Finding the OEP

Load **unpackme.exe** into OllyDBG. And you still here:

00404000	60	PUSHAD	
00404001	E8 2B000000	CALL unpackme.00404031	
00404006	00 0A000A00	OR EAX,000A000A	
0040400B	0A52 65	OR DL, BYTE PTR DS:[EDX+65]	
0040400E	67:6973 74 6572	IMUL ESI, DWORD PTR SS:[BP+DI+74], 646572	
00404016	20746F 3A	AND BYTE PTR DS:[EDI+EBP*2+3A], DH	
0040401A	204E 4F	AND BYTE PTR DS:[ESI+4F], CL	
0040401D	4E	DEC ESI	
0040401E	2D 434F4D4D	SUB EAX, 4D4D4F43	
00404023	45	INC EBP	
00404024	52	PUSH EDX	
00404025	43	INC EBX	
00404026	49	DEC ECX	
00404027	41	INC ECX	
00404028	4C	DEC ESP	
00404029	2121	AND DWORD PTR DS:[ECX], ESP	
0040402B	00 0A000A00	OR EAX, 000A000A	
00404030	0A5D 83	OR BL, BYTE PTR SS:[EBP-7D]	
00404033	ED	IN EAX, DX	I/O command
00404034	06	PUSH ES	
00404035	EB 02	JMP SHORT unpackme.00404039	
00404037	FA 0480B056 001	JMP FAR 0000:56B0B004	Far jump

Then, you must press **Shift + F9**. And you see as follows:

00404232	8DC0	LEA EAX, EAX	Illegal use of register
00404234	CD 20	INT 20	
00404236	64:8F03	POP DWORD PTR FS:[EBX]	
00404239	8BE0	MOV ESP, EAX	
0040423B	8D85 5C100000	LEA EAX, DWORD PTR SS:[EBP+105C]	
00404241	EB 02	JMP SHORT unpackme.00404245	
00404243	EA 0450EB01 EA	JMP FAR EB01:01EB5004	Far jump
0040424A	BF 040000EB	MOV EDI, EB000004	
0040424F	01C7	ADD EDI, EAX	
00404251	0BC0	OR EAX, EAX	
00404253	EB 01	JMP SHORT unpackme.00404256	
00404255	EA 0F842D05 001	JMP FAR 0000:052D080F	Far jump
0040425C	EB 03	JMP SHORT unpackme.00404261	
0040425E	44	INC ESP	
0040425F	50	PUSH EAX	
00404260	CA 8985	RETF 8589	Far return
00404263	2312	AND EDX, DWORD PTR DS:[EDX]	
00404265	0000	ADD BYTE PTR DS:[EAX], AL	
00404267	8B9D 97120000	MOV EBX, DWORD PTR SS:[EBP+1297]	
0040426D	EB 03	JMP SHORT unpackme.00404272	
0040426F	57	PUSH EDI	
00404270	CD 20	INT 20	
00404272	8DB5 580F0000	LEA ESI, DWORD PTR SS:[EBP+F58]	
00404278	EB 03	JMP SHORT unpackme.0040427D	
0040427A	CD 20	INT 20	

Continued, you have to press **ALT + M** to open the **Memory of MAP** OllyDBG.

Address	Size	Owner	Section	Contains	Type	Access	Init	Mapp
00010000	00001000	00010000			Priv 00021004	RW	RW	
00020000	00001000	00020000			Priv 00021004	RW	RW	
00120000	00001000	00030000			Priv 00021104	RW	RW	
0012E000	00002000	00030000		stack of main thread	Priv 00021104	RW	Guarded	RW
00130000	00001000	00130000			Map 00041002	R	R	
00140000	00003000	00140000			Priv 00021004	RW	RW	
00240000	00006000	00240000			Priv 00021004	RW	RW	
00250000	00001000	00250000			Map 00041004	RW	RW	
00260000	00016000	00260000			Map 00041002	R	R	
00280000	00034000	00280000			Map 00041002	R	R	
002C0000	00041000	002C0000			Map 00041002	R	R	
00310000	00006000	00310000			Map 00041002	R	R	
00400000	00001000	unpackme 00400000	PE header		Imag 01001002	R	RWE	
00401000	00001000	unpackme 00400000	code		Imag 01001002	R	RWE	
00402000	00001000	unpackme 00400000	PEShield	data			RWE	
00403000	00001000	unpackme 00400000	PEShield				RWE	
00404000	00003000	unpackme 00400000	ANAKIN2K	SFX, imports			RWE	
77E60000	00001000	kernel32 77E60000		PE header		Enter	RWE	
77E61000	00075000	kernel32 77E60000	.text	code, imports			RWE	
77ED6000	00003000	kernel32 77E60000	.data	data			RWE	
77ED9000	00006000	kernel32 77E60000	.rsrc	resources			RWE	
77F3F000	00006000	kernel32 77E60000	.reloc	relocations			RWE	
77F50000	00001000	ntdll 77F50000		PE header		Ctrl+B	RWE	
77F51000	00006000	ntdll 77F50000	.text	code, exports			RWE	
77FC0000	00005000	ntdll 77F50000	ECODE	code			RWE	
77FC5000	00005000	ntdll 77F50000	.data	data		F2	RWE	
77FCA000	0002C000	ntdll 77F50000	.rsrc	resources			RWE	
77FF6000	00003000	ntdll 77F50000	.reloc	relocations			RWE	
77F60000	00007000	77F60000					RWE	
77FB0000	00002400	77FB0000					RWE	
77FDE000	00001000	77FDE000		data block o			RWE	
77FDF000	00001000	77FDF000					RWE	
77FE0000	00001000	77FE0000					RWE	

Et puis, you search the address line contain **401,000**. Right click on it and choose **Set breakpoint on memory access**.

Now, press **Shift + F9**:

004044D7	8DC0	LEA EAX,EAX	Illegal use of register
004044D9	CD 20	INT 20	
004044DB	64:8F03	POP DWORD PTR FS:[EBX]	
004044DE	8BE0	MOV ESP,EAX	
004044E0	8B85 A7120000	MOV EAX,DWORD PTR SS:[EBP+12A7]	
004044E6	EB 03	JMP SHORT unpackme.004044EB	
004044E8	2BD2	SUB EDX,EDX	
004044EA	E9 03403CEB	JMP EB7C84F2	
004044EF	01E9	ADD ECX,EBP	
004044F1	8BC8	MOV ECX,EAX	
004044F3	EB 02	JMP SHORT unpackme.004044F7	
004044F5	EA 048B7008 EB	JMP FAR 01EB:08708B04	Far jump
004044FC	C8 33FFEB	ENTER 0FF33,0EB	
00404500	01C7	ADD EDI,EAX	
00404502	4F	DEC EDI	

Following, you press **Shift + F9** again:

0040499A	3107	XOR DWORD PTR DS:[EDI],EAX	
0040499C	EB 02	JMP SHORT unpackme.004049A0	
0040499E	0FFF	???	Unknown command
004049A0	310F	XOR DWORD PTR DS:[EDI],ECX	
004049A2	C1C0 03	ROL EAX,3	
004049A5	EB 03	JMP SHORT unpackme.004049AA	
004049A7	57	PUSH EDI	
004049A8	CD 20	INT 20	
004049AA	03C1	ADD EAX,ECX	
004049AC	83C7 04	ADD EDI,4	
004049AF	EB 01	JMP SHORT unpackme.004049B2	
004049B1	C8 49EB02	ENTER 0EB49,2	
004049B5	EA 0475E1EB 01	JMP FAR B801:EBE17504	Far jump
004049BC	EB A3	JMP SHORT unpackme.00404961	
004049BE	80BD 19120000	CMP BYTE PTR SS:[EBP+1219],1	
004049C5	75 0E	JNZ SHORT unpackme.004049D5	

Waaaa, the job is very uninspired ... You have press **Shift + F9** until you see as follows (I count **21 times**):

00401000	6A	DB 6A	CHAR 'j'
00401001	00	DB 00	
00401002	68	DB 68	CHAR 'h'
00401003	00	DB 00	
00401004	30	DB 30	CHAR '0'
00401005	40	DB 40	CHAR '@'
00401006	00	DB 00	
00401007	68	DB 68	CHAR 'h'
00401008	1E	DB 1E	
00401009	30	DB 30	CHAR '0'
0040100A	40	DB 40	CHAR '@'
0040100B	00	DB 00	
0040100C	6A	DB 6A	CHAR 'j'
0040100D	00	DB 00	
0040100E	E8	DB E8	
0040100F	00	DB 00	
00401010	00	DB 00	
00401011	00	DB 00	

Next, press **CTRL + A** analyze the code for:

```

00401000 . 6A 00      PUSH 0
00401002 . 68 00300000 PUSH unpackme.00403000
00401007 . 68 1E304000 PUSH unpackme.0040301E
0040100C . 6A 00      PUSH 0
0040100E . E8 00000000 CALL unpackme.00401020
00401013 . 6A 00      PUSH 0
00401015 . E8 00000000 CALL unpackme.0040101A
0040101A . FF25 00204000 JMP DWORD PTR DS:[402000]
00401020 . FF25 00204000 JMP DWORD PTR DS:[402000]
00401026 . 6A 00      PUSH 0
  
```

OUR OEP!

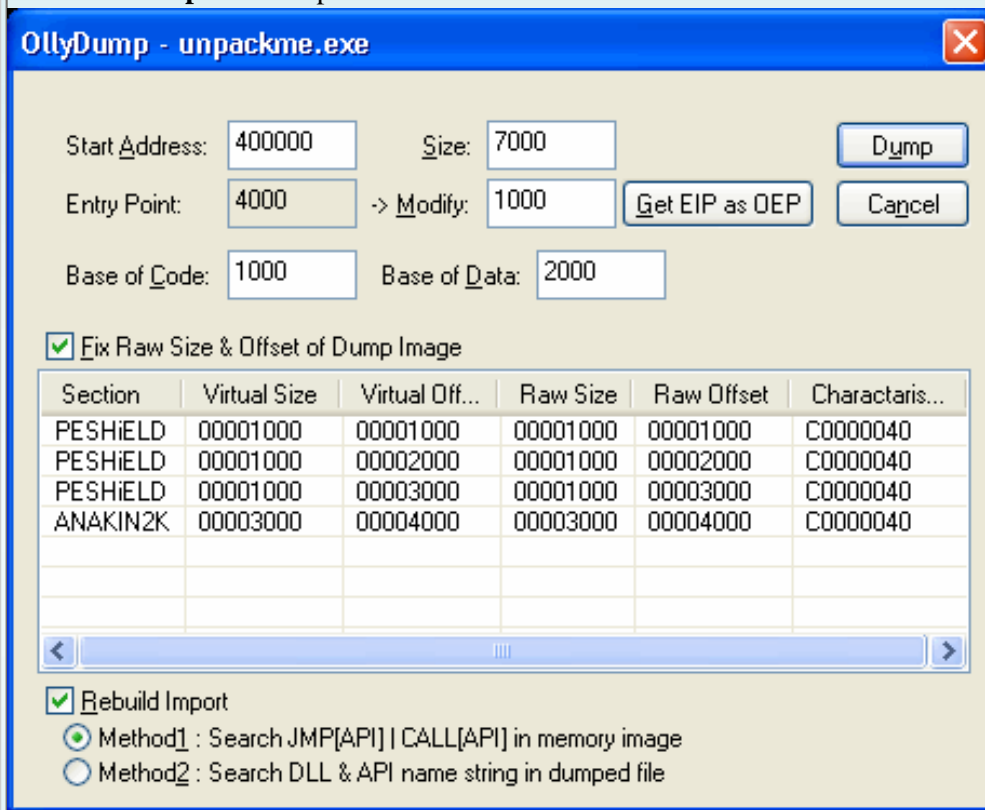
```

Style = MB_OK!MB_APPLMODAL
Title = "UCT-Vietnamese Crackers TEAM!"
Text = "Try to Unpack me! Packed with PE-SHIELD v0.25"
hOwner = NULL
MessageBoxA
ExitCode = 0
ExitProcess
kernel32.ExitProcess
USER32.MessageBoxA
  
```

Congratulations! According OEP we found is **401,000**. And now we Calculate the real OEP of this unpackme by the formula:
 Real OEP = OEP find in Olly-Image Base = 401000-400000 = **1000**.

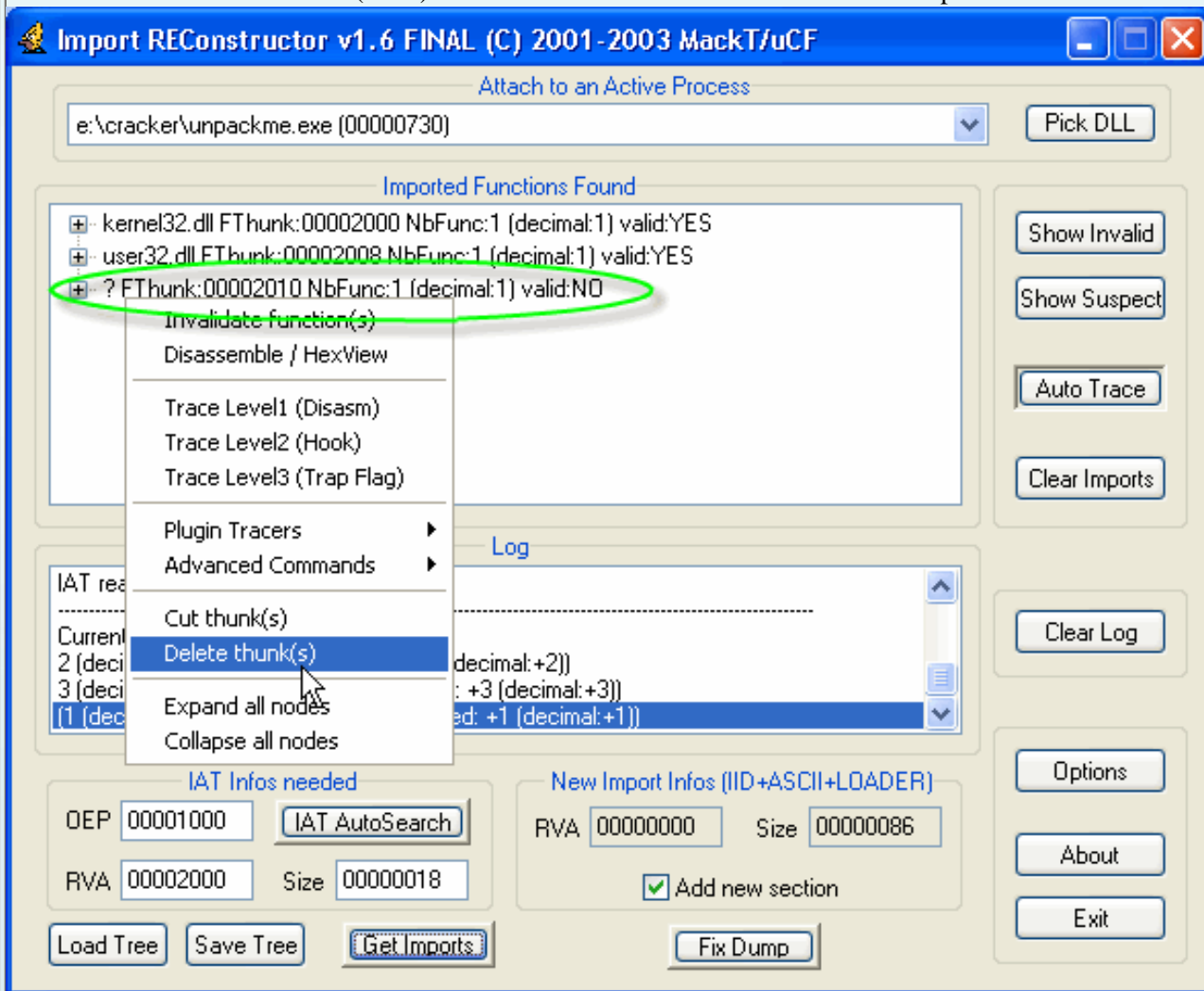
4th dumping

At address **00401000**, we go to the menu Plugin -> OllyDump -> dump debugged process. And then, just press dump, save the file at **dumped.exe** unpacked.



5. Finding and Fixing the Adress Import Table

And select Open ImpREC attached to active process and choose unpackme.exe. Change the value in the OEP window to the one we wrote down earlier (1000) then select IATAutosearch then click Get Imports.

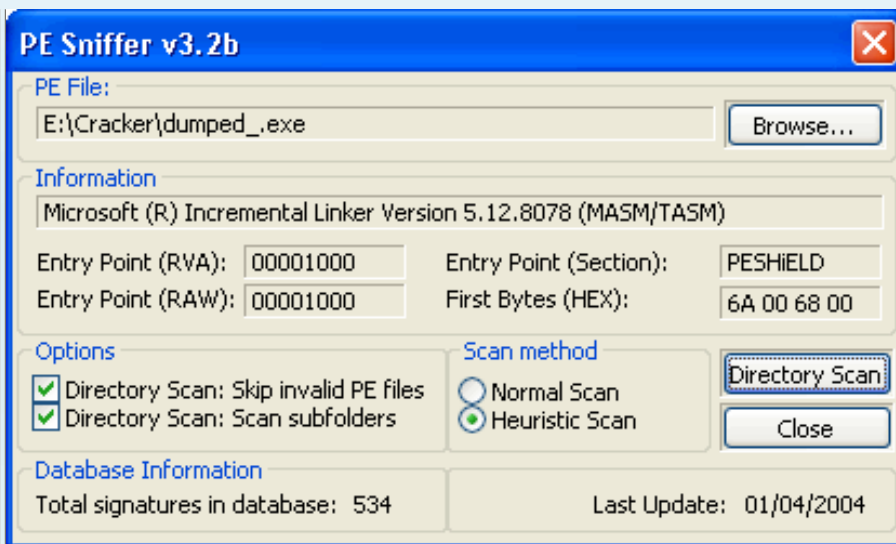


One Import Functions is invalid ... Before, right on this click import function and choose Delete thunks. Now, click fix to fix IAT dump the file **dumped.exe**. And then LordPE open, choose rebuild PE optimize for size of unpackme ..

6. Testing Our Unpacked file

Now run unpacked files. Wow, not crash.

Using **PE 3.2b** for **Sniffer** detect: **MASM / TASM**. Okie, PE-shield v0.25 by ANAKiN [DaVinci] is now unpacked successful!



7. Conclusion

Special thanx to **koncool** et **R @ dier** for this template.

My Greetz to: Deux, NVH (c), luucorp, Maip0301, R @ dier, tlandn, CTL, JAL, LeVuHoang, 777, LeonHart, Bin ... and you ;-)!

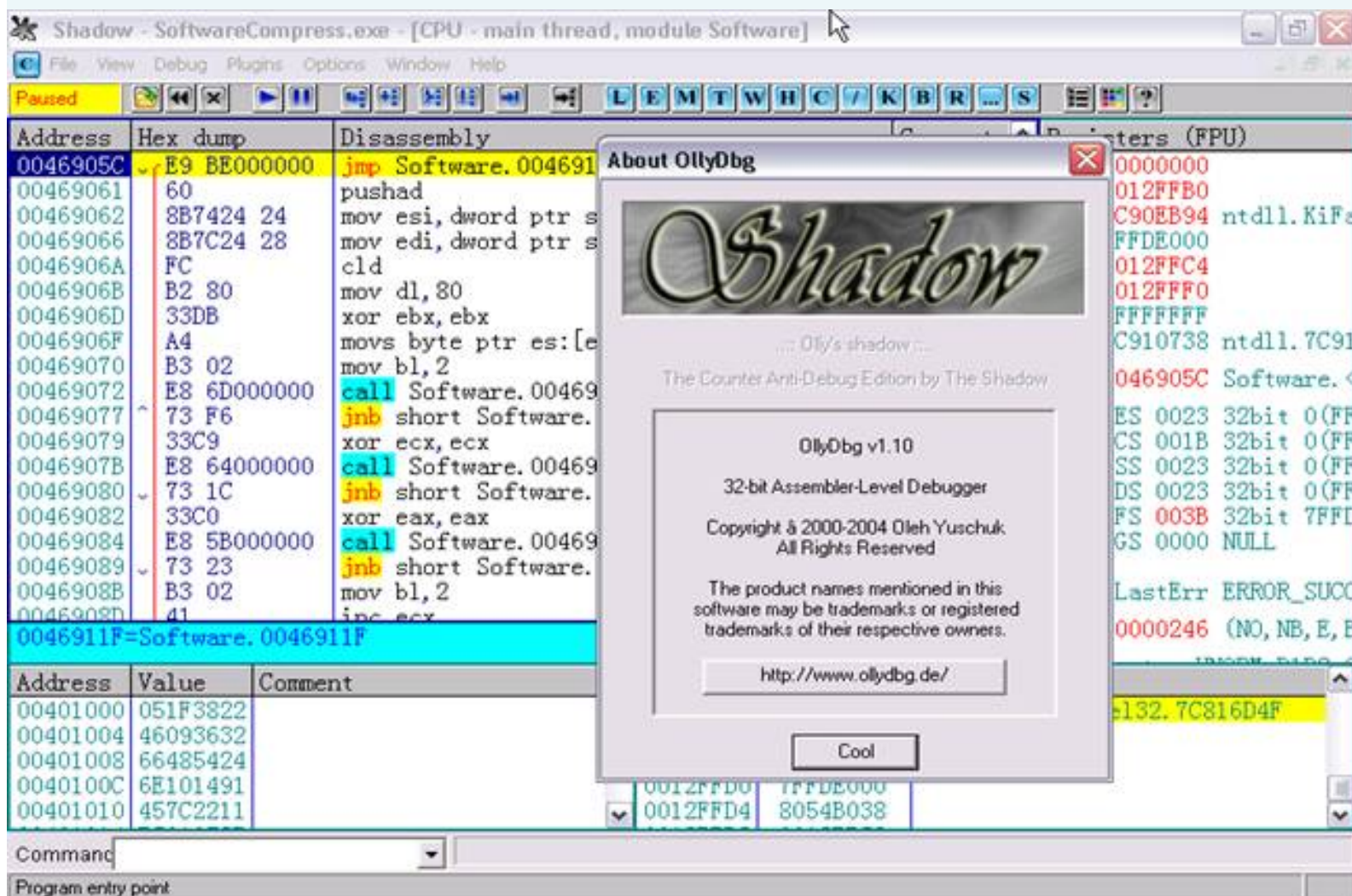
To be continued ...

Written by [hacnho](#) (tutorial date: Sai Gon 5/4/2004)

**MANUAL unpack
compress
Software 1.2**

Target: Software compress 1.2
Homepage : [Http://www.bgsopt.com/](http://www.bgsopt.com/)
Crack Tool: Shadow 1.Ollydbg
2. OllyDump plugin
3. Import REConstructor 1.6 Final
4. RDG Packer Detector v0.5.8
Author: Why Not Bar

Software compress is a soft 1.2 appears, its name has said to us and to its call. Brother we try unpack it to see stars. They see him through the Shadow Tool Ollydbg children download yesterday. Đep to them!



Ok, work only. First, the use of [RDG Packer Detector v0.5.8](#) scan it and our information is as follows:



Load target to Ollydbg Shadow:

0046905A	0000	add byte ptr ds:[eax], al	
0046905C	E9 BE000000	jmp Software.0046911F	
00469061	60	pushad	
00469062	8B7424 24	mov esi, dword ptr ss:[esp+24]	
00469066	8B7C24 28	mov edi, dword ptr ss:[esp+28]	
0046906A	FC	cld	
0046906B	B2 80	mov dl, 80	

Alt + M and select the image:

00360000	00001000	Software		PE header	PRIV	RW	RWE
00400000	00001000	Software		code	Image	R	RWE
00401000	00052000	Software		data	Image	R	RWE
00453000	00016000	Software					
00469000	00001000	Software					
0046A000	00001000	Software					
00470000	00005000						
00530000	00002000						
00540000	00103000						
00650000	0008C000						
77D40000	00001000	USER32					
77D41000	0005F000	USER32					
77DA0000	00002000	USER32					
77DA2000	0002B000	USER32					
77DC0000	00003000	USER32					
77F10000	00001000	GDI32					
77F11000	00041000	GDI32					
77F52000	00001000	GDI32					
77F53000	00001000	GDI32					
77F54000	00002000	GDI32					
7C800000	00001000	kernel32					
7C801000	00082000	kernel32					

Actualize		
View in Disassembler	Enter	
Dump in CPU		
Dump		
Search	Ctrl+B	
Set break-on-access	F2	
Set memory breakpoint on access		
Set memory breakpoint on write		
Set access		

Press F9 to 1 in one here:

00152A97	A4	movs byte ptr es:[edi], byte ptr ds:[esi]	
00152A98	B3 02	mov bl, 2	
00152A9A	E8 6D000000	call 00152B0C	
00152A9F	73 F6	jnb short 00152A97	
00152AA1	33C9	xor ecx, ecx	
00152AA3	E8 64000000	call 00152B0C	
00152AA8	73 1C	jnb short 00152AC6	

Scroll down the mouse until you see signs as follows:

00152C89	C2 0400	ret 4	
00152C8C	60	pushad	
00152C8D	FF7424 24	push dword ptr ss:[esp+24]	
00152C91	FF95 9D114100	call dword ptr ss:[ebp+41119D]	
00152C97	61	popad	
00152C98	C2 0400	ret 4	
00152C9B	0000	add byte ptr ds:[eax], al	
00152C9D	0000	add byte ptr ds:[eax], al	
00152C9F	0000	add byte ptr ds:[eax], al	

Scroll to Set 1 billion and BP (press F2) in 00152C71

Address	Disassembly	Comment	Module	Priv	Rtl	Rtl	Rtl
				Inag	R	Rtl	Rtl
00350000	00001000	Software	code				
00360000	00001000	Software	code				
00400000	00001000	Software	code				
00401000	00052000	Software	code				
00453000	00016000	Software	.rsrc				
00469000	00001000	Software	SoftComp				
0046A000	00001000	Software	.idata				
00470000	00005000	Software	code				
00530000	00002000	Software	code				
00540000	00103000	Software	code				
00650000	0008C000	Software	code				
77D40000	00001000	USER32	code				
77D41000	0005F000	USER32	code				
77DA0000	00002000	USER32	.data				
77DA2000	0002B000	USER32	.rsrc				
77DC0000	00003000	USER32	.reloc				
77F10000	00001000	GDI32	code				
77F11000	00041000	GDI32	code				
77F52000	00001000	GDI32	.data				
77F53000	00001000	GDI32	.rsrc				
77F54000	00002000	GDI32	.reloc				
7C800000	00001000	kernel32	code				
7C801000	00082000	kernel32	code				
7C883000	00005000	kernel32	.data				
7C888000	00066000	kernel32	.rsrc				
7C8EE000	00006000	kernel32	.reloc				

Actualize

View in Disassembler Enter

Dump in CPU

Dump

Search Ctrl+B

Set break-on-access F2

Set memory breakpoint on access

Set memory breakpoint on write

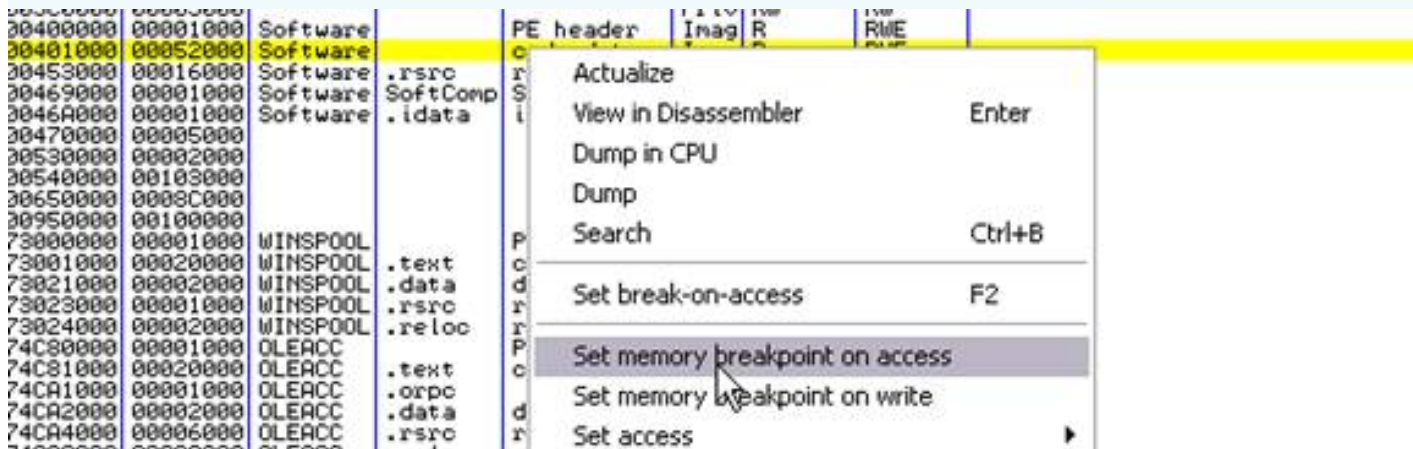
Remove memory breakpoint

Set access

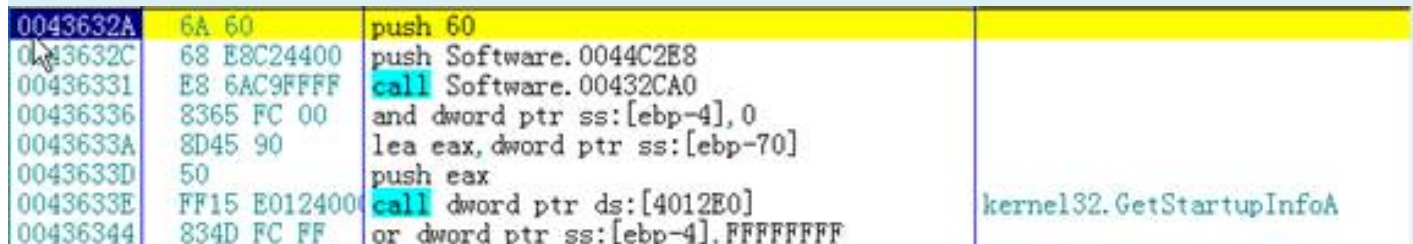
00152C6F	51	push ecx	
00152C70	57	push edi	
00152C71	- FFA5 9D114100	jmp dword ptr ss:[ebp+41119D]	kernel32.GlobalFree
00152C77	60	pushad	
00152C78	FF7424 24	push dword ptr ss:[esp+24]	
00152C7C	6A 40	push 40	
00152C7E	FF95 95114100	call dword ptr ss:[ebp+411195]	
00152C84	894424 1C	mov dword ptr ss:[esp+1C], eax	

0012FFA4	00000018	
0012FFA8	0043632A	Software.0043632A
0012FFAC	001529E0	
0012FFB0	001529E8	
0012FFB4	001A69E4	

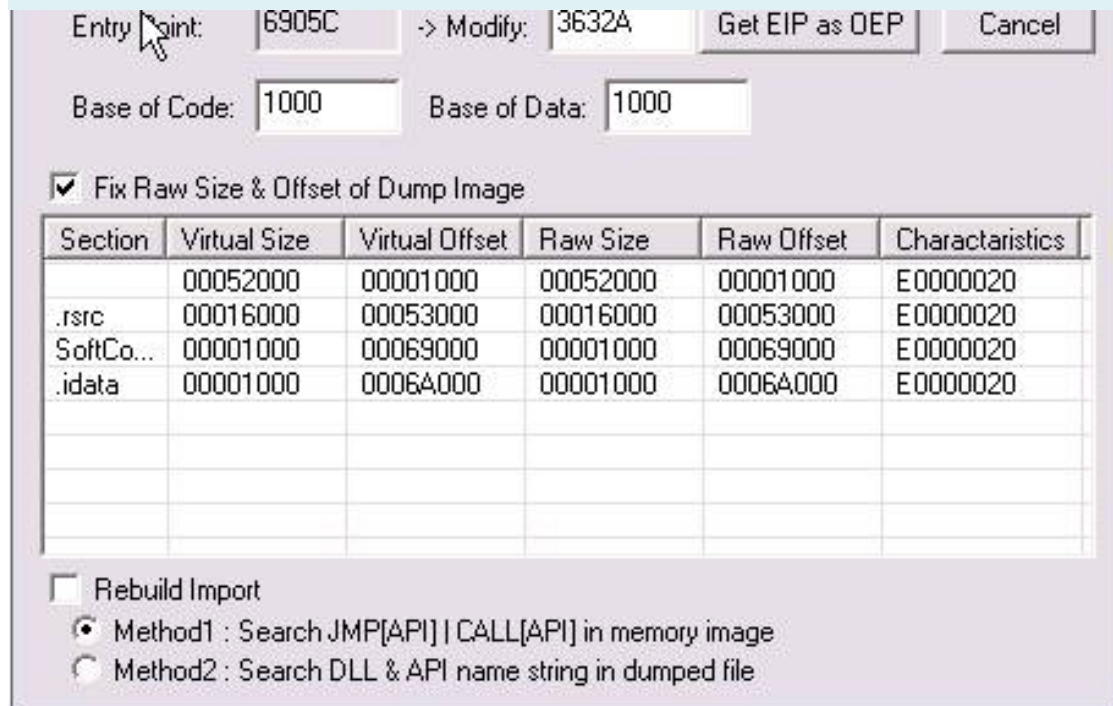
file:///C:/RCE%20Unpacking%20eBook%20[Tra...Unpacking%20Software%20Compress%201.2.htm (4 of 6) [1/9/2009 9:45:32 LithiumLi]



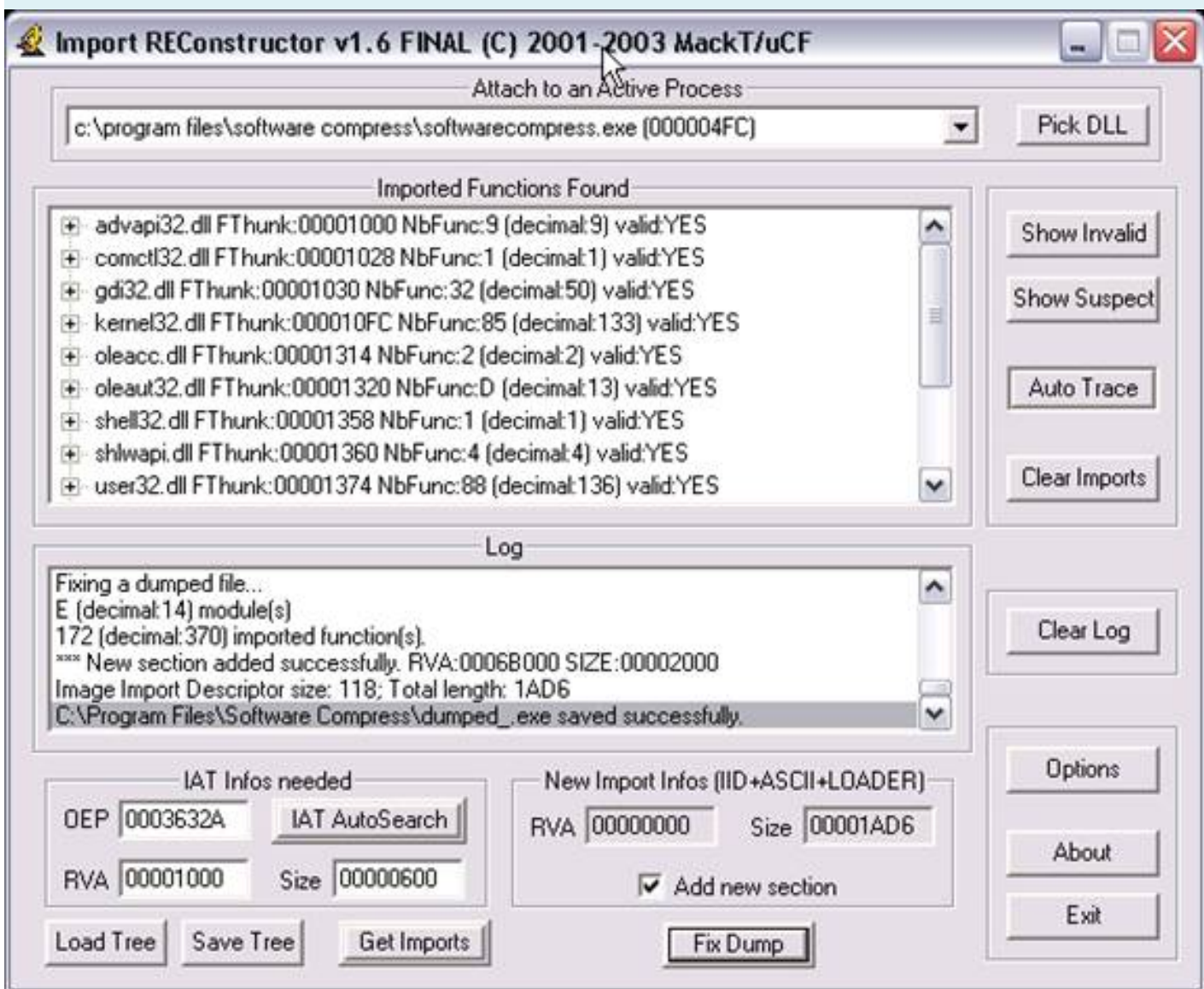
Press F9 one of you to be **OEP**



Ha ha, good too! Only dump, used here to dump OllyDump Plugin, select the image:



Open up to 1.6 ImportREC Fix dump



Test Run File "dumped_.exe." Ohh, File run run lickerish. Unpack Done!

*Written by
Why Not Bar*

KaGra Tutorials

Translated and written by: **kienmanowar**

Manual unpacking **SPLayer** **0:08**

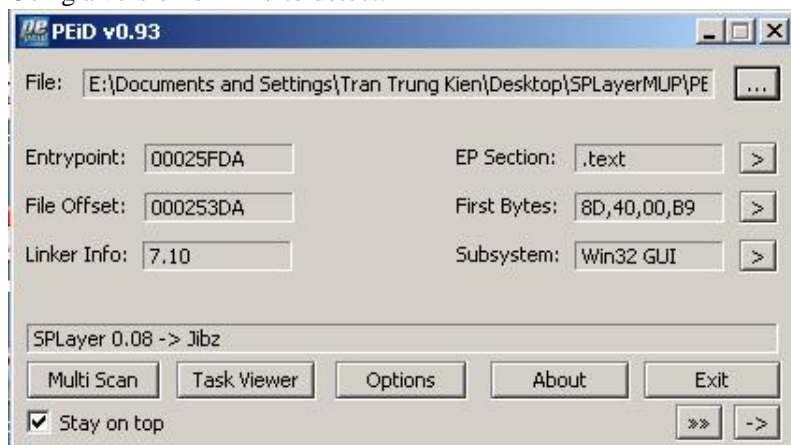
Information	Unpacking for Newbie's
Target	PEiD v0.93
Available	http:// www.reaonline.net
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, Lord PE 1.4, Plugin Command Line, ImpRec v1.6f.
Protection	SPLayer 0:08
L Evel	Beginner
Category	Manual unpacking
Author	KaGra (Thursday, 01 March 2005)

1. Introduction

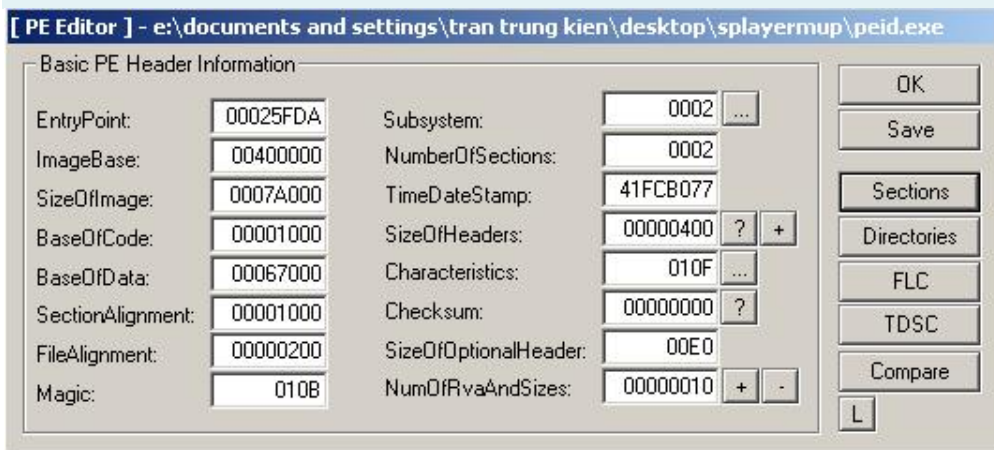
The author of this article after you download the program PEid v0.93 on, the goal is to find the authors want to know the API functions are used in the new version is not nay.Nhung Luckily, version v0. 93 are Protected by a Protector.Sau using a version of PEid to detect the author learned that Packer is: **SPLayer 0:08 -> Jibz**.

PE 2.Detect and get info

Using a version of PEid to detect:



Use Lord PE 1.4 to search for more information:



[Section Table]						
Name	VOffset	VSize	ROffset	RSize	Flags	
.text	00001000	00077000	00000400	00025000	E0000020	
.rsrc	00078000	00002000	00025400	00001800	E0000020	

So we have been as follows: EntryPoint is 25FDA, ImageBase always is: 400000.

3. Finding the Original Entry Point

After the above information, we open Olly target and Load (PEid.exe) in Olly. Chon **Yes to Ananlysis** and we will stop at **EntryPoint**:

Address	Hex dump	Disassembly
00425FDA	8D40 00	LEA EAX,DWORD PTR DS:[EAX]
00425FDD	B9 CC5F4200	MOV ECX,PEID.00425FCC
00425FE2	6A 0E	PUSH 0E
00425FE4	58	POP EAX
00425FE5	C0C01 04	ROR BYTE PTR DS:[ECX+EAX],4
00425FE9	48	DEC EAX
00425FEA	75 F9	JNZ SHORT PEID.00425FE5

Oki, after Load in Olly finished, we in the **Options** Menu \ **debugging Options** to customize the options in Exceptions as follows:

☒ Ignore memory access violations in KERNEL32

Press **Shift + F9** 1 times, Olly will stop here:

Address	Hex dump	Disassembly	Comment
00401016	89	DB 89	
00401017	08	DB 08	
00401018	50	DB 50	CHAR 'P'
00401019	45	DB 45	CHAR 'E'
0040101A	43	DB 43	CHAR 'C'
0040101B	32	DB 32	CHAR '2'
0040101C	00	DB 00	
0040101D	8E	DB 8E	
0040101E	E0	DB E0	

Now, in the window of Olly we click the **"M"** to open the window **Memory map**. We will set a **breakpoint on memory access** similar to Figure below:

Memory map							
Address	Size	Owner	Section	Contains	Type	Access	Initial
00010000	00001000				Priv	RW	RW
00020000	00001000				Priv	RW	RW
00120000	00001000				Priv	RW	Gua: RW
0012E000	00002000			stack of ma	Priv	RW	Gua: RW
00130000	00001000				Map	R	R
00140000	00003000				Priv	RW	RW
00240000	00006000				Priv	RW	RW
00250000	00001000				Map	RW	RW
00260000	00016000				Map	R	R
00280000	00034000				Map	R	R
002C0000	00041000				Map	R	R
00310000	00006000				Map	R	R
00400000	00001000	PEiD		PE header	Imag	R	RWE
00401000	00077000	PEiD	.text	code	Imag	R	RWE
00478000		Actualize			mag	R	RWE
77E60000		View in Disassembler		Enter	mag	R	RWE
77E61000					mag	R	RWE
77ED6000		Dump in CPU			mag	R	RWE
77ED9000		Dump			mag	R	RWE
77F3F000					mag	R	RWE
77F50000		Search		Ctrl+B	mag	R	RWE
77F51000					mag	R	RWE
77FC0000					mag	R	RWE
77FC5000		Set break-on-access		F2	mag	R	RWE
77FCA000					mag	R	RWE
77FF6000		Set memory breakpoint on access			mag	R	RWE
7F6F0000					ap	R E	R E
7FFB0000		Set memory breakpoint on write			ap	R	R
7FFDE000		Set access			priv	RWE	RWE
7FFDF000					priv	RWE	RWE
7FFE0000					priv	R	R
		Copy to clipboard					
		Sort by					
		Appearance					

Next press **Shift + F9** once, Olly will stop here:

Address	Hex dump	Disassembly
00479599	C602 E9	MOV BYTE PTR DS:[EDX],0E9
0047959C	83C2 05	ADD EDX,5
0047959F	2BCA	SUB ECX,EDX
004795A1	894A FC	MOV DWORD PTR DS:[EDX-4],ECX
004795A4	33C0	XOR EAX,EAX
004795A6	C3	RETN

Get **F7** to Trace and implement orders **RETN**, Olly will give us a window to the API's memory:

Address	Hex dump	Disassembly
77F833A0	64:8B25 000000	MOV ESP,DWORD PTR FS:[0]
77F833A7	64:8F05 000000	POP DWORD PTR FS:[0]
77F833AE	8BE5	MOV ESP,EBP
77F833B0	5D	POP EBP
77F833B1	C2 1400	RETN 14

Oki, if we continue to use F7 Trace (**36 times**) we will reach a code as follows:

Address	Hex dump	Disassembly
7FFE0300	8BD4	MOV EDX,ESP
7FFE0302	0F34	SYSENTER
7FFE0304	C3	RETN

At this code authors called (at Jamps kernel, Ring-3 debuggers can not proceed debugging). This means that if we press **F7** to continue the trace we can not continue to be Debug again. The program will run and ignore the position OEP that we need to find. So at **77F833A0**, we will open a window **Memory map** and set the same as BP, we have done in the tren.Sau press **Shift + F9**, we will stop here:

Address	Hex dump	Disassembly
00401016	E9	DB E9

Here we press **Ctrl + A** (for Analysis), we have been as follows:

Address	Hex dump	Disassembly
00401016	.- E9 8C850700	JMP PEID.004795A7

Press **F7** to Trace **JMP** order, here we will:

Address	Hex dump	Disassembly
004795A7	B8 6884BAFF	MOV EAX,FFB88468
004795AC	64:8F05 000000	POP DWORD PTR FS:[0]
004795B3	83C4 04	ADD ESP,4
004795B6	55	PUSH EBP
004795B7	53	PUSH EBX
004795B8	51	PUSH ECX
004795B9	57	PUSH EDI
004795BA	56	PUSH ESI
004795BB	52	PUSH EDX
004795BC	8D98 F8108D00	LEA EBX,DWORD PTR DS:[EAX+8D10F8]

Continue to Trace address **004795BC**. **PUSH** The order we see that the record of the program stored in Packer Stack. Then code enforcement and this will finally get them out of the Stack and take us to **OEP**. Therefore, when we Trace to address **004795BC**, to write to the **ESP** **Registers** window. Mouse must write at this bar and select **Follow in dump**. In the window **dump window** we will see the following:

Address	Hex dump	ASCII
0012FFAC	04 03 FE 7F 77 22 D4 77 FA FF FF FF 00 00 00 00
0012FFBC	00 F0 FD 7F F0 FF 12 00 69 EB E7 77 FA FF FF FF
0012FFCC	76 22 D4 77 00 F0 FD 7F F4 DC 94 F5 C8 FF 12 00
0012FFDC	0E 6A 53 80 FF FF FF FF 86 BB E9 77 18 5A E9 77
0012FFEC	00 00 00 00 00 00 00 00 00 00 00 00 DA 5F 42 00
0012FFFC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Oki, now we highlight 4 bytes at **0012FFAC (04 03 FE 7F)**, and mouse to select **Breakpoint-> Hardware-on access> Dword**. This will make for Olly to stop the record with the same **ESP** value with the value in (**0012FFAC**). This happens when the basic bar record is removed from Stack and prior to the **OEP**. This procedure is used in most of the Packer simple, but it does not mean that all Packer.

Address	Hex dump	ASCII
0012FFAC	04 03 FE 7F 77 22 D4 77 FA FF FF FF 00 00 00 00
0012FFBC	00 F0 FD 7F F0 FF 12 00 69 EB E7 77 FA FF FF FF
0012FFCC	76 22 D4 77 00 F0 FD 7F F4 DC 94 F5 C8 FF 12 00
0012FFDC	0E 6A 53 80 FF FF FF FF 86 BB E9 77 18 5A E9 77
0012FFEC	00 00 00 00 00 00 00 00 00 00 00 00 DA 5F 42 00
0012FFFC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Next we **Remove memory breakpoint**, press Shift + F9 1 times we'll stop here:

Address	Hex dump	Disassembly
0047964A	5E	POP ESI
0047964B	5F	POP EDI
0047964C	59	POP ECX
0047964D	5B	POP EBX
0047964E	5D	POP EBP
0047964F	FF0	JMP EAX

At **0047964A**, we trace through press **F7 JMP** commands us to address **00455F1E**, mouse right at this address and select Analysis> Analysis Code we will be as follows:

Address	Hex dump	Disassembly	Comment
00455F1E	. 6A 60	PUSH 60	
00455F20	. 68 08F54200	PUSH PEID.0042F508	
00455F25	. E8 B2180000	CALL PEID.004577DC	
00455F2A	. BF 94000000	MOV EDI,94	
00455F2F	. 8BC7	MOV EAX,EDI	
00455F31	. E8 1AFAFFFF	CALL PEID.00455950	
00455F36	. 8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00455F39	. 8BF4	MOV ESI,ESP	
00455F3B	. 893E	MOV DWORD PTR DS:[ESI],EDI	
00455F3D	. 56	PUSH ESI	
00455F3E	. FF15 9011400	CALL DWORD PTR DS:[401190]	VersionInformation GetVersionExA

OEP so that we are **00455F1E**. OEP calculation of the formula:

Real OEP = OEP find in Olly-Image Base = 00455F1E - 00400000 = **00055F1E**

4. Unpacked dumping our files

At **00455F1E**, right click and select **dump debugged process**. Uncheck **Rebuilt Import**, click and save the **dump** under a name that **dumped.exe** example.

OllyDump - PEiD.exe

Start Address: 400000 Size: 7A000 Dump

Entry Point: 25FDA -> Modify: 55F1E Get EIP as OEP Cancel

Base of Code: 1000 Base of Data: 67000

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
.text	00077000	00001000	00077000	00001000	E0000020
.rsrc	00002000	00078000	00002000	00078000	E0000020

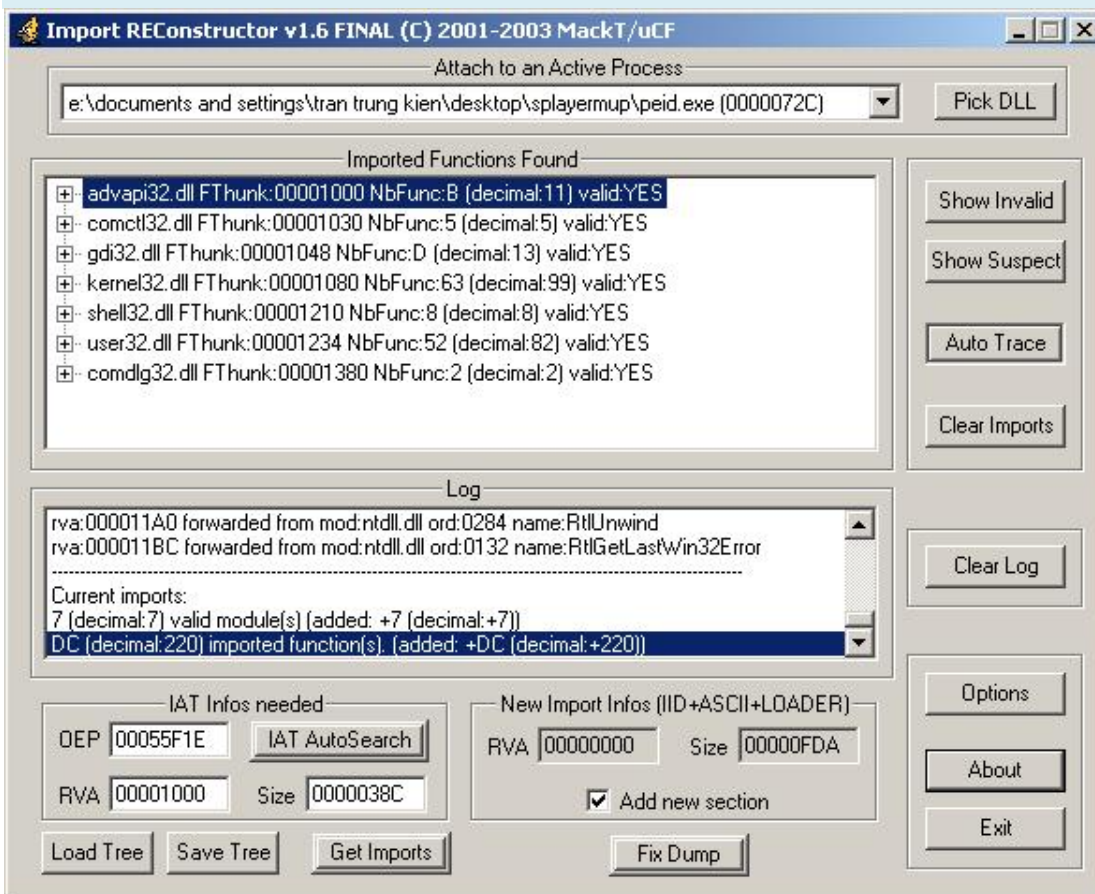
☐ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

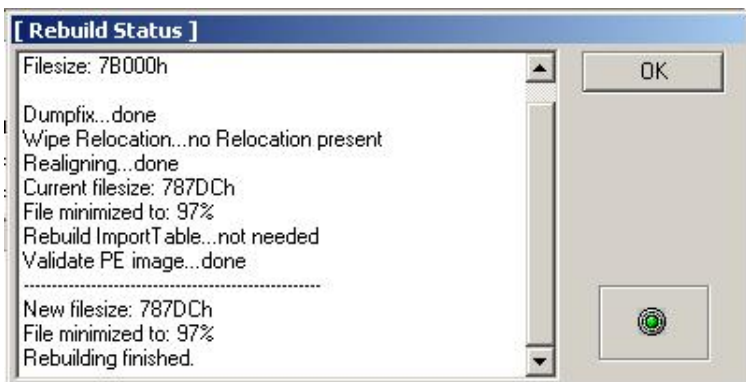
☐ Method2 : Search DLL & API name string in dumped file

5. Finding and Fixing the Address Import Table

Hold the window Olly, open **ImpRec** select list box in **Peid.exe Attach to an Active Process**. Enter OEP we have found the calculation of the above in, and click **Get Autosearch IAT Imports**. Then click **Show Invalid**. Keke too good not Invalid thanks at all. Finally click **Fix dump** file and select **dumped.exe**.

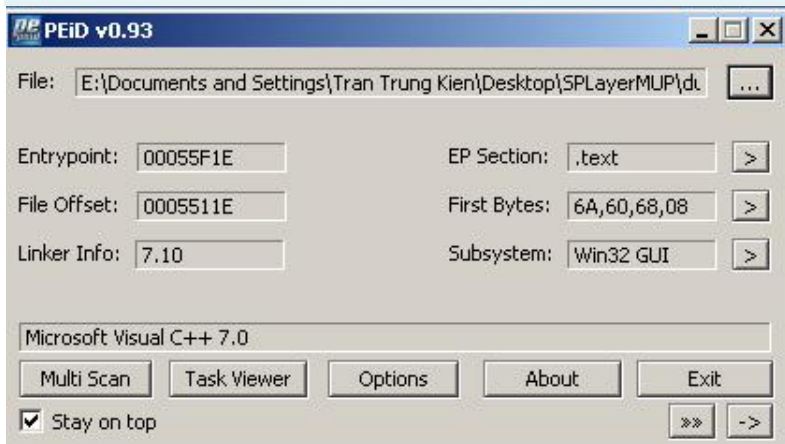


Finally, we used to LordPE Rebuilt again **Dumped_.exe** file.



6. Testing Our Unpacked file

Oki, a test file we Unpacked. Yup! It works.
Used to Detect PEid again we have been as follows:



So **SPlayer 0.88** unpack was successful. Have fun:)

7. Conclusion

My Greetz to: tlandn (supported me this tut) and KaGra (author of this tut)

To thank my family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHQCkrker, the_Lighthouse, Hoadongnoi, Nini ... all REA's members, HacNho, RongChauA, Deux all my friend, and YOU!



Written by **kienmanowar** (tutorial date: HaNoi 14/03/2005)

...: Copyright © 2005 by kienmanowar <[=-=]> REA-cRaCkErTeAm (www.reaonline.net) :...

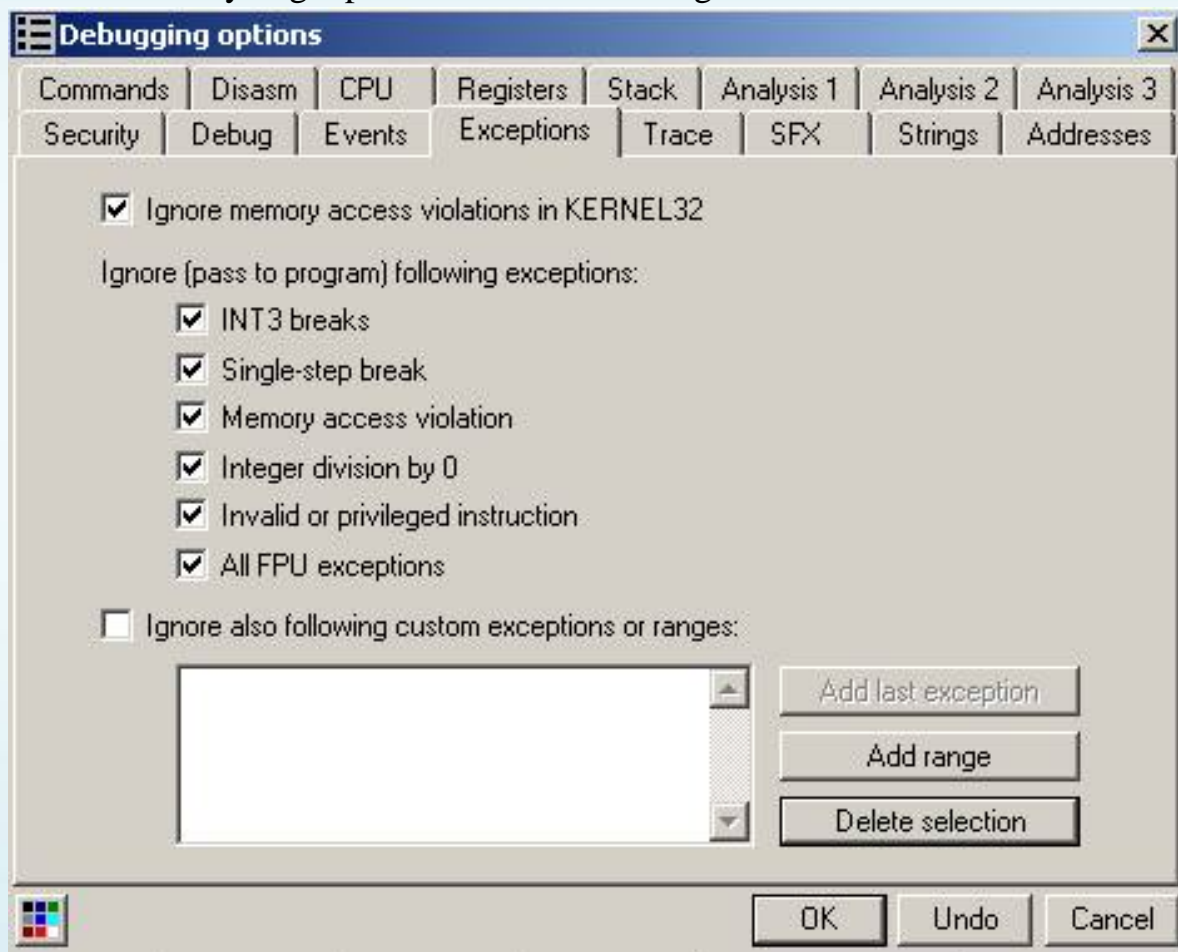
Tu hoc unpack SVKP 1:32 TUT 1 - ASM TARGET

Author: tlandn

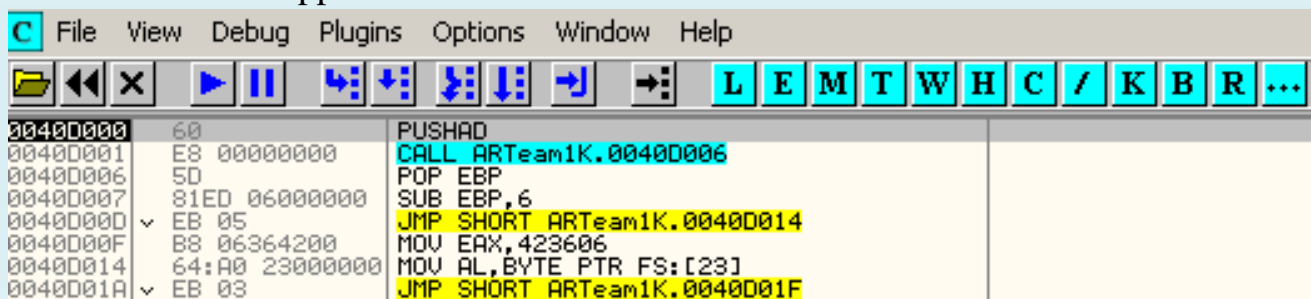
Welcome you. I have ideas, write a tut about SVKP hope is to share some of their knowledge. Hopefully you support. If you have ideas, please write something to share tut. Thanks. We will begin.

I. OEP Search:

First edit the OllyDbg Options like the following:



Load the file. What opportunities click OK. We will start here.



Press Alt-M to open the "Memory Map". Set "Breakpoint on access memory" in the same image:

00260000	00016000			Map	R	R	
00280000	00034000			Map	R	R	
002C0000	00041000			Map	R	R	
00310000	00006000			Map	R	R	
00320000	00004000			Map	R E	R E	
003E0000	00002000			Map	R E	R E	
003F0000	00001000			Priv	RW	RW	
00400000	00001000	ARTeam1K	PE header	Imag	R	RWE	
00401000	00001000	ARTeam1K	code				
00402000	00001000	ARTeam1K	data				
00403000	00001000	ARTeam1K					

00401000	00001000	ARTeam1K		code	Actualize	
00402000	00001000	ARTeam1K		data		
00403000	00001000	ARTeam1K			View in Disassembler	Enter
00404000	00009000	ARTeam1K		resources		
0040D000	0000F000	ARTeam1K	.svkp	SFX, impor	Dump in CPU	
00420000	00103000				Dump	
00530000	00078000				Search	Ctrl+B
00830000	00001000					
00840000	00004000					
00850000	00003000					
00870000	00002000					
01050000	00002000				Set break-on-access	F2
70BD0000	00001000	SHLWAPI		PE header		
70BD1000	0005A000	SHLWAPI	.text	code, impo		
70C2B000	00002000	SHLWAPI	.data	data	Set memory breakpoint on access	
70C2D000	00002000	SHLWAPI	.rsrc	resources		
70C2F000	00006000	SHLWAPI	.reloc	relocation	Set memory breakpoint on write	
71950000	00001000	comctl32		PE header		
71951000	00088000	comctl32	.text	code, impo	Set access	
719D9000	00001000	comctl32	.data	data		

Back "CPU Window." Press F9. The one stop here:

008F137F	6285 0E0B0000	BOUND EAX,QWORD PTR SS:[EBP+80E]	
008F1385	EB 02	JMP SHORT 008F1389	
008F1387	0FE88B D1EB02C1	PSUBSB MM1,QWORD PTR DS:[EBX+CD02EBD1]	
008F138E	208B C2EB02CD	AND BYTE PTR DS:[EBX+CD02EBC2],CL	
008F1394	208B 8A4F0800	AND BYTE PTR DS:[EBX+84F8A],CL	
008F139A	007C03 EB	ADD BYTE PTR DS:[EBX+EAX-15],BH	
008F139E	0369 74	ADD EBP,DWORD PTR DS:[ECX+74]	
008F13A1	FB	STI	

Press Shift-F9. The one stop here:

0091B6B1	8A06	MOV AL,BYTE PTR DS:[ESI]	
0091B6B3	46	INC ESI	
0091B6B4	47	INC EDI	
0091B6B5	8843 0F	MOV BYTE PTR DS:[EBX+F],AL	
0091B6B8	8A46 FF	MOV AL,BYTE PTR DS:[ESI-1]	
0091B6BB	55	PUSH EBP	
0091B6BC	E8 00000000	CALL 0091B6C1	
0091B6C1	5D	POP EBP	
0091B6C2	81ED 0D470000	SUB EBP,470D	
0091B6C8	8A8D 50030000	MOV CL,BYTE PTR SS:[EBP+350]	
0091B6CE	5D	POP EBP	
0091B6CF	32C1	XOR AL,CL	
0091B6D1	8847 FF	MOV BYTE PTR DS:[EDI-1],AL	
0091B6D4	8BC5	MOV EAX,EBP	
0091B6D6	4D	DEC EBP	
0091B6D7	85C0	TEST EAX,EAX	
0091B6D9	75 A4	JNZ SHORT 0091B67F	
0091B6DB	33C0	XOR EAX,EAX	
0091B6DD	5D	POP EBP	
0091B6DE	5F	POP EDI	
0091B6DF	5E	POP ESI	
0091B6E0	5B	POP EBX	
0091B6E1	C2 1400	RETN 14	

You note in 91B6E1 we have the "RETN 14" (blue line above). Set at the breakpoint by using the mouse click it and press F2.

0091B6B1	8A06	MOV AL,BYTE PTR DS:[ESI]	
0091B6B3	46	INC ESI	
0091B6B4	47	INC EDI	
0091B6B5	8843 0F	MOV BYTE PTR DS:[EBX+F],AL	
0091B6B8	8A46 FF	MOV AL,BYTE PTR DS:[ESI-1]	
0091B6BB	55	PUSH EBP	
0091B6BC	E8 00000000	CALL 0091B6C1	
0091B6C1	5D	POP EBP	
0091B6C2	81ED 0D470000	SUB EBP,470D	
0091B6C8	8A8D 50030000	MOV CL,BYTE PTR SS:[EBP+350]	
0091B6CE	5D	POP EBP	
0091B6CF	32C1	XOR AL,CL	
0091B6D1	8847 FF	MOV BYTE PTR DS:[EDI-1],AL	
0091B6D4	8BC5	MOV EAX,EBP	
0091B6D6	4D	DEC EBP	
0091B6D7	85C0	TEST EAX,EAX	
0091B6D9	75 A4	JNZ SHORT 0091B67F	
0091B6DB	33C0	XOR EAX,EAX	
0091B6DD	5D	POP EBP	
0091B6DE	5F	POP EDI	
0091B6DF	5E	POP ESI	
0091B6E0	5B	POP EBX	
0091B6E1	C2 1400	RETN 14	

Press Alt-M to the "Memory Map" select "Breakpoint on memory access"

00310000	00006000			Map	R	R
00320000	00004000			Map	R E	R E
003E0000	00002000			Map	R E	R E
003F0000	00001000			Priv	RW	RW
00400000	00001000	ARTeam1K	PE header	Image	R	RWE

003E0000	00002000				Map	R	E	R	E
003F0000	00001000				Priv	RW		RW	
00400000	00001000	ARTeam1K		PE header	Imag	R		RWE	
00401000	00001000	ARTeam1K		code					
00402000	00001000	ARTeam1K		data					
00403000	00001000	ARTeam1K							
00404000	00009000	ARTeam1K		resources					
0040D000	0000F000	ARTeam1K	.svkp	SFX, imports					
00420000	00103000								
00530000	00078000								
00830000	00001000								
00840000	00004000								
00850000	00003000								
00870000	00002000								
00880000	00045000								
008F0000	00043000								
00940000	00002000								
01050000	00002000								
70BD0000	00001000	SHLWAPI		PE header					
70BD1000	0005A000	SHLWAPI	.text	code, import					
70C2B000	00002000	SHLWAPI	.data	data					
70C2D000	00002000	SHLWAPI	.rsrc	resources					
70C2F000	00006000	SHLWAPI	.reloc	relocations					

Actualize
View in Disassembler Enter
Dump in CPU
Dump
Search Ctrl+B
Set break-on-access F2
Set memory breakpoint on access
Set memory breakpoint on write
Remove memory breakpoint

Back on "CPU Window." Press F9. We will break at 91B6E1 (points we've set the breakpoint above).

Uncheck breakpoint by pressing F2.

0091B6D9	^ 75 A4	JNZ SHORT 0091B67F	
0091B6DB	33C0	XOR EAX,EAX	
0091B6DD	5D	POP EBP	
0091B6DE	5F	POP EDI	
0091B6DF	5E	POP ESI	
0091B6E0	5B	POP EBX	
0091B6E1	C2 1400	RETN 14	

At the "Memory Map". Reset "on access memory Breakpoint" in the same image:

00260000	00016000				Map	R		R	
00280000	00034000				Map	R		R	
002C0000	00041000				Map	R		R	
00310000	00006000				Map	R		R	
00320000	00004000				Map	R	E	R	E
003E0000	00002000				Map	R	E	R	E
003F0000	00001000				Priv	RW		RW	
00400000	00001000	ARTeam1K		PE header	Imag	R		RWE	
00401000	00001000	ARTeam1K		code					
00402000	00001000	ARTeam1K		data					
00403000	00001000	ARTeam1K							
00404000	00009000	ARTeam1K		resources					
0040D000	0000F000	ARTeam1K	.svkp	SFX, import					
00420000	00103000								
00530000	00078000								
00830000	00001000								
00840000	00004000								
00850000	00003000								
00870000	00002000								
01050000	00002000								
70BD0000	00001000	SHLWAPI		PE header					
70BD1000	0005A000	SHLWAPI	.text	code, import					
70C2B000	00002000	SHLWAPI	.data	data					
70C2D000	00002000	SHLWAPI	.rsrc	resources					
70C2F000	00006000	SHLWAPI	.reloc	relocation					
71950000	00001000	comctl32		PE header					
71951000	00088000	comctl32	.text	code, import					
71909000	00001000	comctl32	.data	data					

Actualize
View in Disassembler Enter
Dump in CPU
Dump
Search Ctrl+B
Set break-on-access F2
Set memory breakpoint on access
Set memory breakpoint on write
Set access

Go to "CPU Window." Press F9. The program will break at 401,000:

00401000	6A 00	PUSH 0	
00401002	E8 19060000	CALL ARTeam1K.00401620	
00401007	A3 40304000	MOV DWORD PTR DS:[403040],EAX	
0040100C	6A 65	PUSH 65	
0040100E	FF35 40304000	PUSH DWORD PTR DS:[403040]	
00401014	E8 DD050000	CALL ARTeam1K.004015F6	
00401019	A3 44304000	MOV DWORD PTR DS:[403044],EAX	
0040101E	E8 45060000	CALL ARTeam1K.00401668	
00401023	68 FCFBE200	PUSH 0E2FBFC	
00401028	E8 1D060000	CALL ARTeam1K.0040164A	
0040102D	A3 C2304000	MOV DWORD PTR DS:[4030C2],EAX	

This is our OEP.

II. Dump program:

Using OllyDump to dump. Select the same image:



Start Address: Size:

Entry Point: -> Modify:

Base of Code: Base of Data:

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
	00001000	00001000	00001000	00001000	C0000040
	00001000	00002000	00001000	00002000	C0000040
	00001000	00003000	00001000	00003000	C0000040
	00009000	00004000	00009000	00004000	C0000040
.svkp	0000F000	0000D000	0000F000	0000D000	C0000040

☐ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Name the file "a.exe" J

III. Rebuild IAT:

Using Imprec load programs. Enter the parameters of the same image, click "IAT AutoSearch" then click "Get Imports"

Import REConstructor v1.6 FINAL (C) 2001-2003 MackT/uCF

Attach to an Active Process

Imported Functions Found

- + ? FTunk:00002000 NbFunc:1 (decimal:1) valid:NO
- + ? FTunk:00002008 NbFunc:5 (decimal:5) valid:NO
- + ? FTunk:00002020 NbFunc:8 (decimal:8) valid:NO
- + ? FTunk:00002044 NbFunc:E (decimal:14) valid:NO

Log

IAT read successfully.

Current imports:

0 (decimal:0) valid module(s)

1C (decimal:28) imported function(s). (added: +1C (decimal:+28))

1C (decimal:28) unresolved pointer(s)) (added: +1C (decimal:+28))

IAT Infos needed

OEP

New Import Infos (IID+ASCII+LOADER)

RVA Size

Import REConstructor v1.6 FINAL (C) 2001-2003 MackT/uCF

Attach to an Active Process

c:\a\arteam1kg.exe (00000390)

Pick DLL

Imported Functions Found

rva:00002000 ptr:009846D7

? FTThunk:00002008 NbFunc:5 (decimal:5) valid:NO

rva:00002008 ptr:00984677

rva:0000200C ptr:00984689

rva:00002010 ptr:0098469A

rva:00002014 ptr:009846AA

rva:00002018 ptr:009846BA

? FTThunk:00002020 NbFunc:5 (decimal:5) valid:NO

rva:00002020 ptr:009846C0

rva:00002024 ptr:009846D0

rva:00002028 ptr:009846E0

rva:00002032 ptr:009846F0

rva:00002036 ptr:00984700

rva:00002040 ptr:00984710

rva:00002044 ptr:00984720

rva:00002048 ptr:00984730

rva:00002052 ptr:00984740

rva:00002056 ptr:00984750

rva:00002060 ptr:00984760

rva:00002064 ptr:00984770

rva:00002068 ptr:00984780

rva:00002072 ptr:00984790

rva:00002076 ptr:009847A0

rva:00002080 ptr:009847B0

rva:00002084 ptr:009847C0

rva:00002088 ptr:009847D0

rva:00002092 ptr:009847E0

rva:00002096 ptr:009847F0

rva:000020A0 ptr:00984800

rva:000020A4 ptr:00984810

rva:000020A8 ptr:00984820

rva:000020AC ptr:00984830

rva:000020B0 ptr:00984840

rva:000020B4 ptr:00984850

rva:000020B8 ptr:00984860

rva:000020BC ptr:00984870

rva:000020C0 ptr:00984880

rva:000020C4 ptr:00984890

rva:000020C8 ptr:009848A0

rva:000020CC ptr:009848B0

rva:000020D0 ptr:009848C0

rva:000020D4 ptr:009848D0

rva:000020D8 ptr:009848E0

rva:000020DC ptr:009848F0

rva:000020E0 ptr:00984900

rva:000020E4 ptr:00984910

rva:000020E8 ptr:00984920

rva:000020EC ptr:00984930

rva:000020F0 ptr:00984940

rva:000020F4 ptr:00984950

rva:000020F8 ptr:00984960

rva:00002100 ptr:00984970

rva:00002104 ptr:00984980

rva:00002108 ptr:00984990

rva:00002112 ptr:009849A0

rva:00002116 ptr:009849B0

rva:00002120 ptr:009849C0

rva:00002124 ptr:009849D0

rva:00002128 ptr:009849E0

rva:00002132 ptr:009849F0

rva:00002136 ptr:00984A00

rva:00002140 ptr:00984A10

rva:00002144 ptr:00984A20

rva:00002148 ptr:00984A30

rva:00002152 ptr:00984A40

rva:00002156 ptr:00984A50

rva:00002160 ptr:00984A60

rva:00002164 ptr:00984A70

rva:00002168 ptr:00984A80

rva:00002172 ptr:00984A90

rva:00002176 ptr:00984AA0

rva:00002180 ptr:00984AB0

rva:00002184 ptr:00984AC0

rva:00002188 ptr:00984AD0

rva:00002192 ptr:00984AE0

rva:00002196 ptr:00984AF0

rva:000021A0 ptr:00984B00

rva:000021A4 ptr:00984B10

rva:000021A8 ptr:00984B20

rva:000021AC ptr:00984B30

rva:000021B0 ptr:00984B40

rva:000021B4 ptr:00984B50

rva:000021B8 ptr:00984B60

rva:000021BC ptr:00984B70

rva:000021C0 ptr:00984B80

rva:000021C4 ptr:00984B90

rva:000021C8 ptr:00984BA0

rva:000021CC ptr:00984BB0

rva:000021D0 ptr:00984BC0

rva:000021D4 ptr:00984BD0

rva:000021D8 ptr:00984BE0

rva:000021DC ptr:00984BF0

rva:000021E0 ptr:00984C00

rva:000021E4 ptr:00984C10

rva:000021E8 ptr:00984C20

rva:000021EC ptr:00984C30

rva:000021F0 ptr:00984C40

rva:000021F4 ptr:00984C50

rva:000021F8 ptr:00984C60

rva:00002200 ptr:00984C70

rva:00002204 ptr:00984C80

rva:00002208 ptr:00984C90

rva:00002212 ptr:00984CA0

rva:00002216 ptr:00984CB0

rva:00002220 ptr:00984CC0

rva:00002224 ptr:00984CD0

rva:00002228 ptr:00984CE0

rva:00002232 ptr:00984CF0

rva:00002236 ptr:00984D00

rva:00002240 ptr:00984D10

rva:00002244 ptr:00984D20

rva:00002248 ptr:00984D30

rva:00002252 ptr:00984D40

rva:00002256 ptr:00984D50

rva:00002260 ptr:00984D60

rva:00002264 ptr:00984D70

rva:00002268 ptr:00984D80

rva:00002272 ptr:00984D90

rva:00002276 ptr:00984DA0

rva:00002280 ptr:00984DB0

rva:00002284 ptr:00984DC0

rva:00002288 ptr:00984DD0

rva:00002292 ptr:00984DE0

rva:00002296 ptr:00984DF0

rva:000022A0 ptr:00984E00

rva:000022A4 ptr:00984E10

rva:000022A8 ptr:00984E20

rva:000022AC ptr:00984E30

rva:000022B0 ptr:00984E40

rva:000022B4 ptr:00984E50

rva:000022B8 ptr:00984E60

rva:000022BC ptr:00984E70

rva:000022C0 ptr:00984E80

rva:000022C4 ptr:00984E90

rva:000022C8 ptr:00984EA0

rva:000022CC ptr:00984EB0

rva:000022D0 ptr:00984EC0

rva:000022D4 ptr:00984ED0

rva:000022D8 ptr:00984EE0

rva:000022DC ptr:00984EF0

rva:000022E0 ptr:00984F00

rva:000022E4 ptr:00984F10

rva:000022E8 ptr:00984F20

rva:000022EC ptr:00984F30

rva:000022F0 ptr:00984F40

rva:000022F4 ptr:00984F50

rva:000022F8 ptr:0098

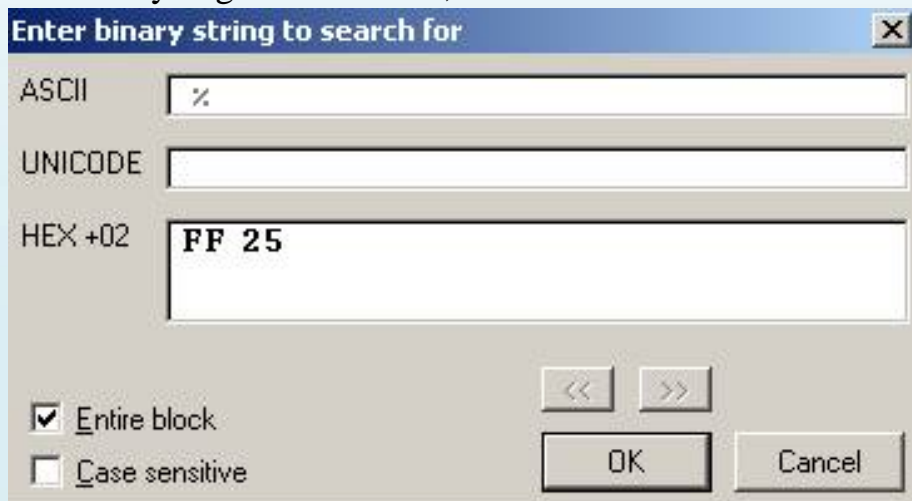
```

? FTunk:00002020 NbFunc:8 (decimal:8) valid:NO
  rva:00002020 ptr:009733DD
  rva:00002024 mod:kernel32.dll ord:01EA name:GlobalUnlock
  rva:00002028 mod:kernel32.dll ord:02B3 name:RtlZeroMemory
  rva:0000202C mod:kernel32.dll ord:01BF name:GetTickCount
  rva:00002030 mod:kernel32.dll ord:01D8 name:GlobalAlloc
  rva:00002034 ptr:00972070
  rva:00002038 mod:kernel32.dll ord:01E3 name:GlobalLock
  rva:0000203C mod:kernel32.dll ord:0398 name:lstrcpy

```

402034 (400000 + 2034).

Back to OllyDbg. We're at 401,000. Press Ctrl-B. Enter FF 25 under the same image:

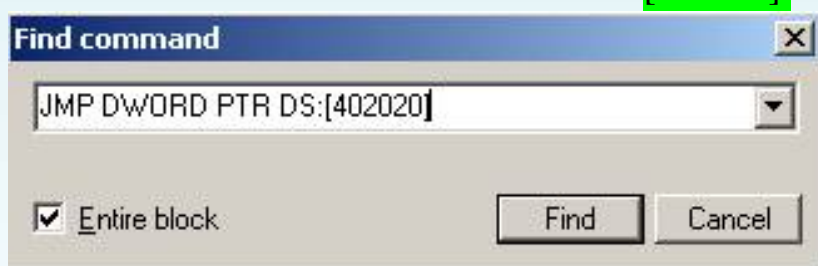


Click OK. We stop here:

004015C2	C2 0400	RETN 4	
004015C5	CC	INT3	
004015C6	- FF25 50204000	JMP DWORD PTR DS:[402050]	
004015CC	- FF25 78204000	JMP DWORD PTR DS:[402078]	
004015D2	- FF25 74204000	JMP DWORD PTR DS:[402074]	
004015D8	- FF25 70204000	JMP DWORD PTR DS:[402070]	

API is a form of JMP DWORD PTR DS: [XXXXXX].

Press Ctrl-F. Enter JMP DWORD PTR DS: [402020]. Click Find.



We stop here:

0040160E	- FF25 44204000	JMP DWORD PTR DS:[402044]	
00401614	- FF25 54204000	JMP DWORD PTR DS:[402054]	
0040161A	- FF25 34204000	JMP DWORD PTR DS:[402034]	
00401620	- FF25 20204000	JMP DWORD PTR DS:[402020]	
00401626	- FF25 2C204000	JMP DWORD PTR DS:[40202C]	
0040162C	- FF25 30204000	JMP DWORD PTR DS:[402030]	

Press Ctrl-R to see what the code calls this function. We are:

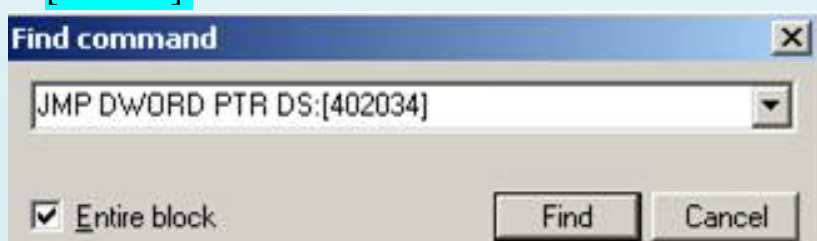
References in ARTeam1K: to 00401620		
Address	Disassembly	Comment
00401002	CALL ARTeam1K.00401620	
00401620	JMP DWORD PTR DS:[402020]	(Initial CPU selection)

Click Double click the first line (401,002).

We're at 401,000. Press F8. We in line 401002. Note the value recorded by EAX = 401000. Press F8 once to perform functions. We look at the value of EAX = 400000 -> This is a function

GetModuleHandleA.

For value API second (402,034) need to find we do the same. Press Ctrl-F. Enter JMP DWORD PTR DS: [402034]. Click Find.





We here:

```
0040160E - FF25 44204000 JMP DWORD PTR DS:[402044]
00401614 - FF25 54204000 JMP DWORD PTR DS:[402054]
0040161A - FF25 34204000 JMP DWORD PTR DS:[402034]
00401620 - FF25 20204000 JMP DWORD PTR DS:[402020]
00401626 - FF25 2C204000 JMP DWORD PTR DS:[40202C]
```

Just press Ctrl-R.

Address	Disassembly	Comment
00401058	CALL ARTeam1K.0040161A	
0040161A	JMP DWORD PTR DS:[402034]	(Initial CF

Kích Double click the first line (401,058).

Press F2 to set the breakpoint.

```
0040104B FF35 C2304000 PUSH DWORD PTR DS:[4030C2]
00401051 E8 FA050000 CALL ARTeam1K.00401650
00401056 6A 00 PUSH 0
00401058 E8 BD050000 CALL ARTeam1K.0040161A
0040105D 55 PUSH EBP
0040105E 8BEC MOV EBP,ESP
```

Press Alt-M to the "Memory Map" select "Breakpoint on memory access"

00310000	00006000			Map	R	E	R	
00320000	00004000			Map	R	E	R	
003E0000	00002000			Map	R	E	R	
003F0000	00001000			Priv	RW		RW	
00400000	00001000	ARTeam1K	PE header	Imag	R		RWE	
00401000	00001000	ARTeam1K	code					
00402000	00001000	ARTeam1K	data					
00403000	00001000	ARTeam1K						
00404000	00009000	ARTeam1K	resources					
0040D000	0000F000	ARTeam1K	.svkp	SFX, imports				
00420000	00103000							
00530000	00078000							
00830000	00001000							
00840000	00004000							
00850000	00003000							
00870000	00002000							
00880000	00045000							
008F0000	00043000							
00940000	00002000							
01050000	00002000							
70BD0000	00001000	SHLWAPI	PE header					
70BD1000	0005A000	SHLWAPI	code, import					
70C2B000	00002000	SHLWAPI	data					
70C2D000	00002000	SHLWAPI	resources					
70C2F000	00006000	SHLWAPI	relocations					

Back "CPU Window." Press F9 to run the program. When you activate the "Exit" to close the break in the OllyDbg 401,058 (for one set breakpoint).

In Imprec we know kernel32.dll under 402,034.

And "Memory Window" set "Breakpoint on access memory" on the section of text in the image as kernel32.dll:

77DD0000	00001000	ADVAPI32	PE header	Imag	R		RWE	
77DD1000	00065000	ADVAPI32	.text	code, import	Imag	R	RWE	
77E36000	00005000	ADVAPI32	.data	data	Imag	R	RWE	
77E3B000	0001B000	ADVAPI32	.rsrc	resources	Imag	R	RWE	
77E56000	00005000	ADVAPI32	.reloc	relocations	Imag	R	RWE	
77E60000	00001000	kernel32	PE header	Imag	R		RWE	
77E61000	00075000	kernel32	.text	code, import	Imag	R	RWE	
77ED6000	00003000	kernel32	.data	data				
77ED9000	00066000	kernel32	.rsrc	resources				
77F3F000	00006000	kernel32	.reloc	relocations				
77F50000	00001000	ntdll	PE header	Imag	R		RWE	
77F51000	0006F000	ntdll	.text	code, export				
77FC0000	00005000	ntdll	ECODE	code				
77FC5000	00005000	ntdll	.data	data				
77FCA000	0002C000	ntdll	.rsrc	resources				
77FF6000	00003000	ntdll	.reloc	relocations				
77F6F000	00007000							
77FB0000	00024000							
77FDE000	00001000		data block					
77FDF000	00001000							

Back "CPU Window." Press F9 3 (or 4) times. We will in kernel32.dll.

77E77963	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]
77E77969	8B48 30	MOV ECX,DWORD PTR DS:[EAX+30]
77E7796C	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]
77E77970	83F8 F4	CMP EAX,-0C
77E77973	^ 0F84 4CB4FFFF	JE kernel32.77E72DC5
77E77979	83F8 F5	CMP EAX,-0B
77E7797C	^ 0F84 38B4FFFF	JE kernel32.77E72DBA
77E77982	83F8 F6	CMP EAX,-0A

Note the address we're standing is **77E77963**. Press Ctrl-N. Click Address column to rearrange the order to make the API. Search the **77E77963**

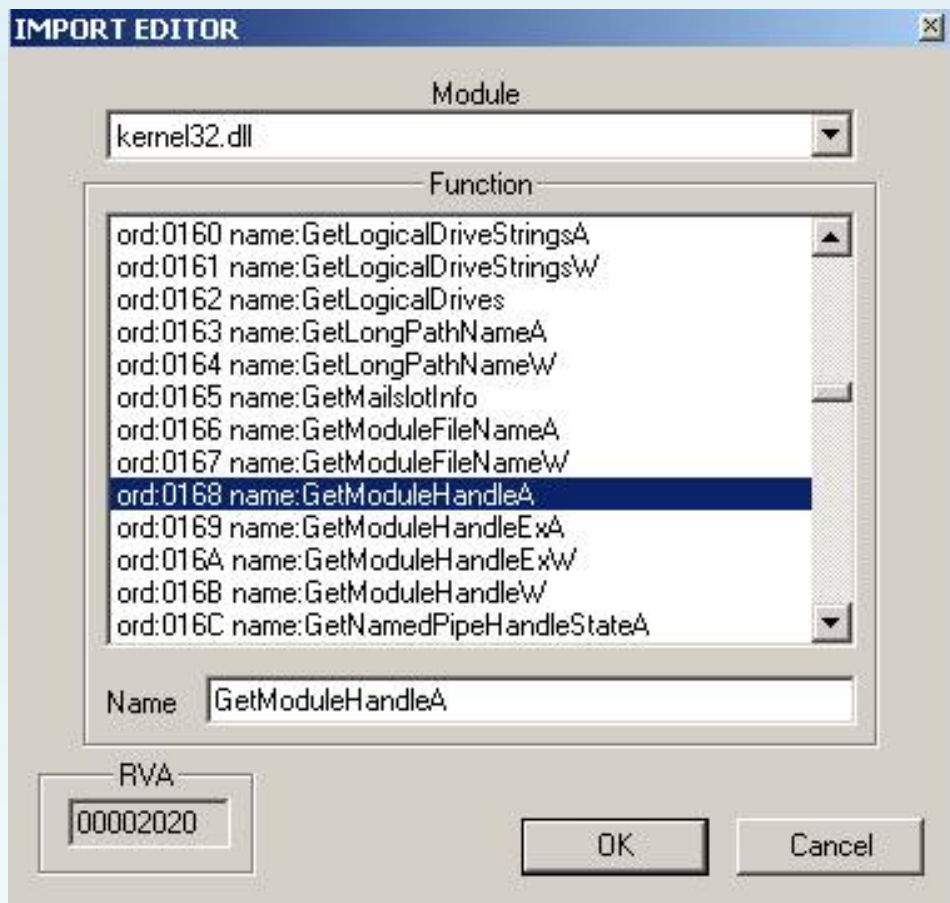
77E777EF	.text	Export	InterlockedIncrement
77E77800	.text	Export	WaitForSingleObjectEx
77E778C5	.text	Export	InterlockedDecrement
77E77963	.text	Export	CloseHandle
77E779B1	.text	Export	CreateFileW

That CloseHandle function.

So we have the results: **402,020 is GetModuleHandleA**

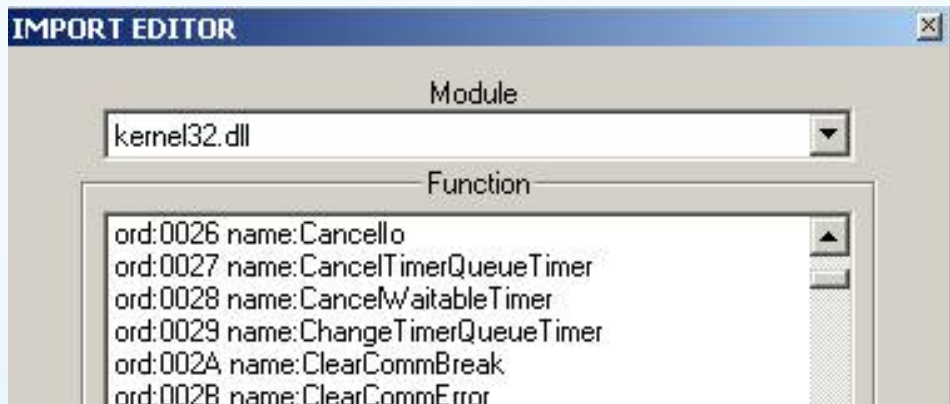
402,034 is CloseHandle

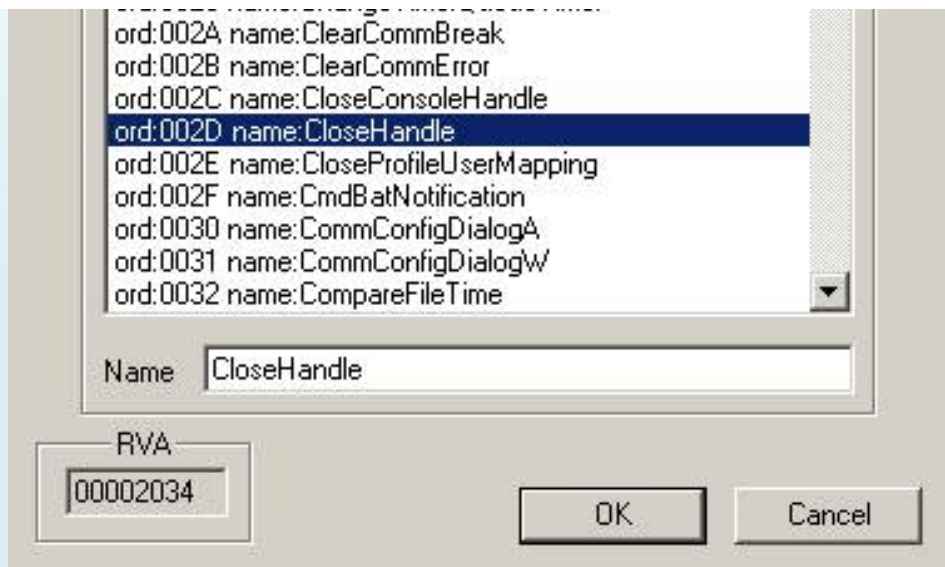
In Imprec revised 2 function properly:



402020:

402034:





Click "Fix dump." Select the file "a.exe". We are file "a_.exe".

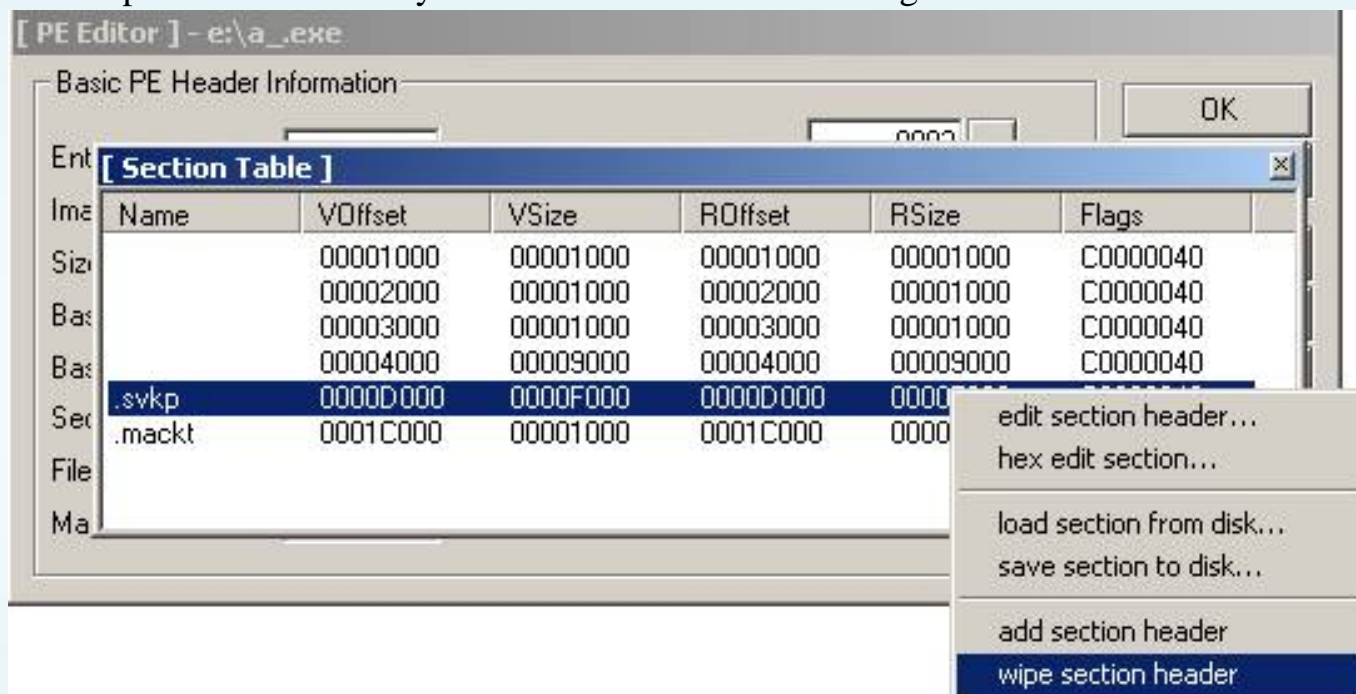
Test file "a_.exe". Running good.

IV. Rebuild EXE:

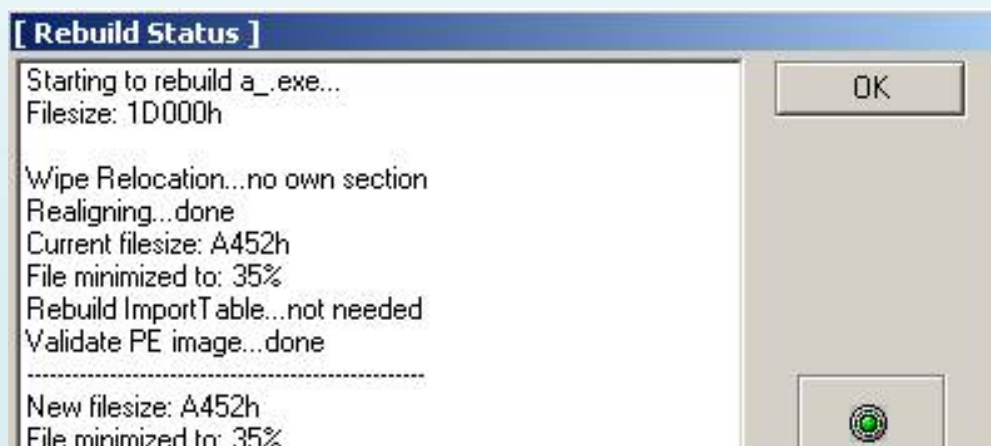
File program "a_.exe" we are still too large

(116 KB). We will reduce it.

LordPE open the file correctly. At the Section. Select the image as:



Then use LordPE rebuild the exe file.





File we have approximately 41 KB.

Them. Wish you success and happy J

Tlandn

Thanks: reaonline all members (many), Crusader, Ricardo, you ...

Tutorials hacnho # 7

Manual unpacking tElock 0.98b1 -> in!

The very important plugin "tELock1.dll" is by tlandn support.

Information	Unpacking for Newbie's
Target	unpackme.exe
Available	http://nhandan.info/hacnho/tuts/unpackme7_tuts.zip
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, Lord PE 1.4, PESniffer 3.2b, 1.6 Final ImportREC and Plugin for tELock1.dll ImpREC.
Protection	tElock version 0.98b1 -> in (to support v0.99)
L Evel	Standard
Category	Manual unpacking

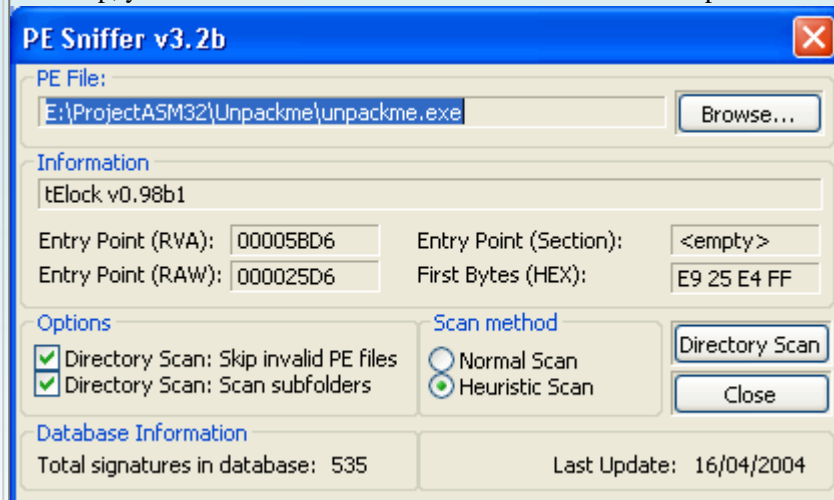
1. Introduction

I try to unpack tELock 0.98b1, but not success because I was check the options in Olly **rebuild Import** Plugin dump. So, Olly can not dump. But now, my good friends tlandn was a tut for maked unpack this packer in English language.

For this reason, I edited and designed for a complete tutorials for you! Thanx again for tlandn help me.

2. Getting Started

First step, you have to find some info from this PE software. Open Lord PE, PE Editor choose. And we have:



[PE Editor] - e:\projectasm32\unpackme\unpackme.exe

Basic PE Header Information

EntryPoint:	00005BD6	Subsystem:	0002	...
ImageBase:	00400000	NumberOfSections:	0004	
SizeOfImage:	00007000	TimeDateStamp:	407F5B47	
BaseOfCode:	00001000	SizeOfHeaders:	00000400	? +
BaseOfData:	00002000	Characteristics:	010F	...
SectionAlignment:	00001000	Checksum:	00005E0A	?
FileAlignment:	00000200	SizeOfOptionalHeader:	00E0	
Magic:	010B	NumOfRvaAndSizes:	00000010	+ -

OK Save Sections Directories FLC TDSC Compare L

[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	00001000	00000400	00000200	C0000040
.rdata	00002000	00001000	00000600	00000200	C0000040
.data	00003000	00001000	00000800	00000200	C0000040
	00004000	00003000	00000A00	00002200	C0000040

EP: 5BD6, flags The value of this case is not needed, Image Base is always 400000, Import Table: 5BE2 and size is 9C.

3. Finding the OEP

Load **unpackme.exe** into OllyDBG. And you still here:

00405BD6	E9 25E4FFFF	JMP unpackme.00404000
00405BD8	0000	ADD BYTE PTR DS:[EAX],AL
00405BD0	000CE1	ADD BYTE PTR DS:[ECX],CL
00405BE0	E8 931E5C00	CALL 009C7A78
00405BE5	0000	ADD BYTE PTR DS:[EAX],AL
00405BE7	0000	ADD BYTE PTR DS:[EAX],AL
00405BE9	0000	ADD BYTE PTR DS:[EAX],AL
00405BEB	0000	ADD BYTE PTR DS:[EAX],AL

Then, you press **F7** until you see as follows:

00404000	FC	CWD
00404001	60	PUSHAD
00404002	E8 02000000	CALL unpackme.00404009
00404007	E8 00E80000	CALL 0041280C
0040400C	0000	ADD BYTE PTR DS:[EAX],AL
0040400E	5E	POP ESI
0040400F	2BC9	SUB ECX,ECX
00404011	58	POP EAX
00404012	74 02	JE SHORT unpackme.00404016
00404014	CD 20	INT 20

And then, you press **Shift + F9 17 times** (after press Shift + F9 18 times, the unpackme run is now complete in memory).
Now you still here:

004057BE	F7F7	DIV EDI	
004057C0	EB E8	JMP SHORT unpackme.004057AA	
004057C2	85E4	TEST ESP,ESP	
004057C4	79 03	JNS SHORT unpackme.004057C9	
004057C6	0F9142 03	SETNO BYTE PTR DS:[EDI+3]	
004057CA	C6	???	Unknown command
004057CB	48	DEC EAX	
004057CC	33FF	XOR EDI,EDI	
004057CE	64:8F07	POP DWORD PTR FS:[EDI]	
004057D1	5F	POP EDI	
004057D2	EB 03	JMP SHORT unpackme.004057D7	
004057D4	FFEB	JMP FAR EBX	Illegal use of register
004057D6	FF83 E02E60E8	INC DWORD PTR DS:[EBX+E8602EE0]	
004057DC	06	PUSH ES	
004057DD	0000	ADD BYTE PTR DS:[EAX],AL	
004057DF	008B 642408EB	ADD BYTE PTR DS:[EBX+EB082464],CL	
004057E5	1A6467 FF	SBB AH,BYTE PTR DS:[EDI-1]	
004057E9	36:0000	ADD BYTE PTR SS:[EAX],AL	
004057EC	64:67:8926 000	MOV DWORD PTR FS:[0],ESP	
004057F2	9C	PUSHFD	
004057F3	810C24 0001000	OR DWORD PTR SS:[ESP],100	
004057FA	9D	POPF	
004057FB	F8	CLC	
004057FC	73 DC	JNB SHORT unpackme.004057DA	
004057FE	CD 20	INT 20	
00405800	64:67:8F06 000	POP DWORD PTR FS:[0]	
00405806	58	POP EAX	
00405807	61	POPAD	
00405808	EB 03	JMP SHORT unpackme.0040580D	
0040580A	FFEB	JMP FAR EBX	Illegal use of register
0040580C	FFFS	PUSH EBP	

Continued, you have to press **ALT + M** to open the **Memory of MAP** OllyDBG.

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00010000	00001000				Priv	RW	RW	
00020000	00001000				Priv	RW	RW	
00120000	00001000				Priv	RW	Gua: RW	
0012E000	00002000			stack of ma	Priv	RW	Gua: RW	
00130000	00001000				Map	R	R	
00140000	00004000				Priv	RW	RW	
00240000	00006000				Priv	RW	RW	
00250000	00001000				Map	RW	RW	
00260000	00016000				Map	R	R	
00280000	00034000				Map	R	R	
002C0000	00041000				Map	R	R	
00310000	00006000				Map	R	R	
00320000	00007000				Map	R E	R E	
003E0000	00002000				Map	R E	R E	
003F0000	00001000				Priv	RW	RW	
00400000	00001000	unpackme		PE header	Imag	RW	RWE	
00401000	00001000	unpackme	.text	code	Imag	R	R	
00402000	00001000	unpackme	.rdata					
00403000	00001000	unpackme	.data	data				
00404000	00003000	unpackme		SFX, import				
00410000	00103000							
00520000	00110000							
00820000	00001000							
00830000	00001000							
00840000	00001000							
77C70000	00001000	GDI32		PE header				
77C71000	0003B000	GDI32	.text	code, import				
77C8C000	00001000	GDI32	.data	data				
77CAD000	00001000	GDI32	.rsrc	resources				
77CAE000	00002000	GDI32	.reloc	relocations				
77CC0000	00001000	RPCRT4		PE header				
77CC1000	00067000	RPCRT4	.text	code, import				
77D28000	00007000	RPCRT4	.orpe	code				
77D2F000	00001000	RPCRT4	.data	data				
77D30000	00001000	RPCRT4	.rsrc	resources				
77D31000	00004000	RPCRT4	.reloc	relocations				
77D40000	00001000	user32		PE header				
77D41000	0005C000	user32	.text	code, import				
77D90000	00002000	user32	.data	data				
77D9F000	0002B000	user32	.rsrc	resources				
77DCA000	00003000	user32	.reloc	relocations				
77DD0000	00001000	ADVAPI32		PE header				
77DD1000	00065000	ADVAPI32	.text	code, import				
77E36000	00005000	ADVAPI32	.data	data	Imag	R	RWE	
77E3B000	0001B000	ADVAPI32	.rsrc	resources	Imag	R	RWE	
77E56000	00005000	ADVAPI32	.reloc	relocations	Imag	R	RWE	
77E60000	00001000	kernel32		PE header	Imag	R	RWE	
77E61000	00075000	kernel32	.text	code, import	Imag	R	RWE	
77ED6000	00003000	kernel32	.data	data	Imag	R	RWE	

Et puis, you search the address line contain **401,000**. Right click on it and choose **Set breakpoint on memory access**.

Now, press **Shift + F9**:

00401000	6A	DB 6A	CHAR 'j'
00401001	00	DB 00	
00401002	68	DB 68	CHAR 'h'
00401003	00	DB 00	
00401004	30	DB 30	CHAR '0'
00401005	40	DB 40	CHAR '@'
00401006	00	DB 00	
00401007	68	DB 68	CHAR 'h'
00401008	34	DB 34	CHAR '4'

Next, press **CTRL + A** analyze the code for:

```

00401000 . 6A 00 PUSH 0
00401002 . 68 00304000 PUSH unpackme.00403000
00401007 . 68 34304000 PUSH unpackme.00403034
0040100C . 6A 00 PUSH 0
0040100E . E8 00000000 CALL unpackme.00401020
00401013 . 6A 00 PUSH 0
00401015 . E8 00000000 CALL unpackme.0040101A
0040101A $- FF25 00204000 JMP DWORD PTR DS:[402000]
00401020 $- FF25 00204000 JMP DWORD PTR DS:[402000]
00401026 00 DB 00
00401028 00 DB 00

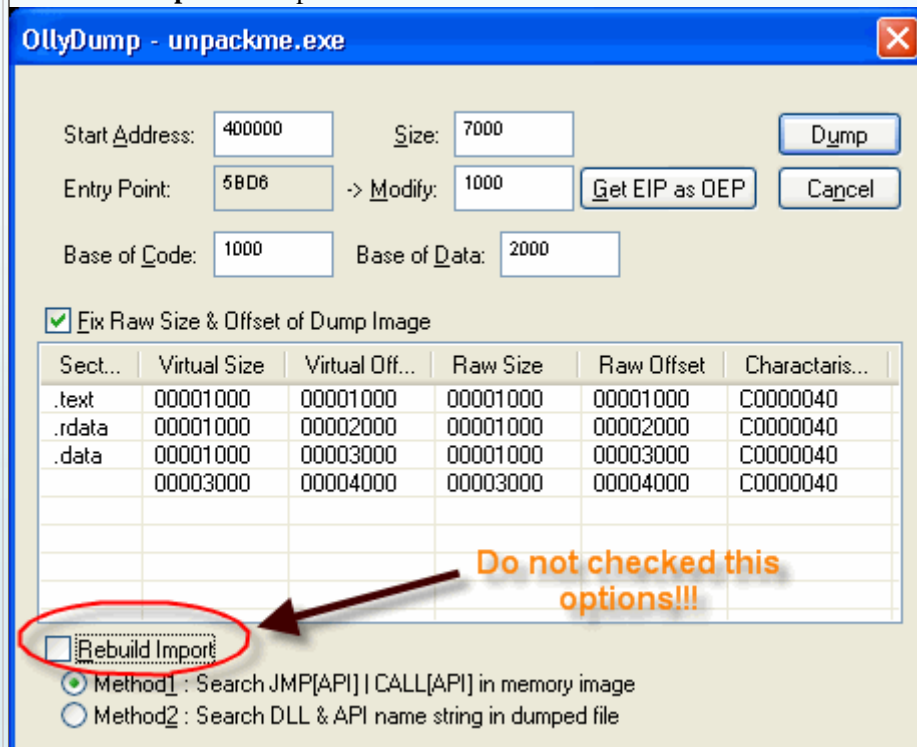
```

ASCII ".....--<[UCT-Vietnamese Crackers TEAM!]>=
ASCII "Try to Unpack me! Packed with tElock 0.98b1

Congratulations! According OEP we found is **401,000**. And now we Calculate the real OEP of this unpackme by the formula:
Real OEP = OEP find in Olly-Image Base = 401000-400000 = **1000**.

4th dumping

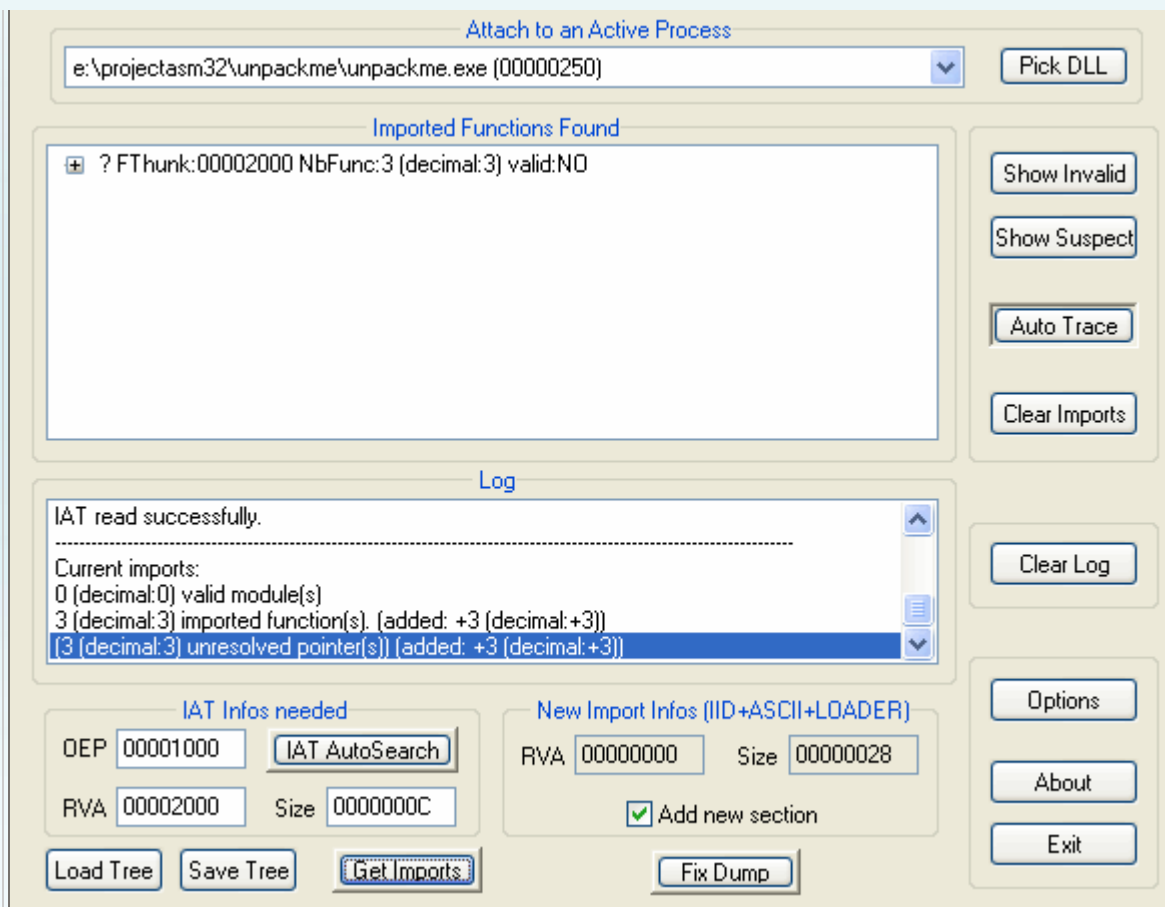
At address **00401000**, we go to the menu Plugin -> OllyDump -> dump debugged process. And then, just press dump, save the file at **dumped.exe** unpacked.



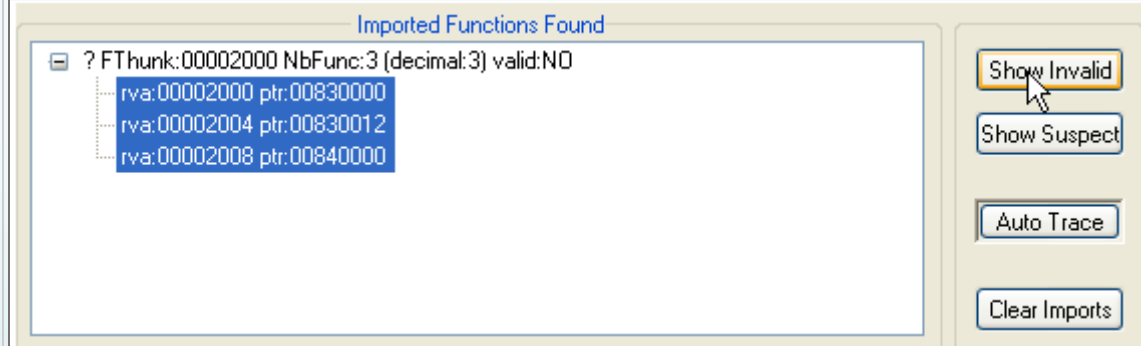
Do not run **dumped.exe** now, will be a crash ... It must fix IAT.

5. Finding and Fixing the Adress Import Table

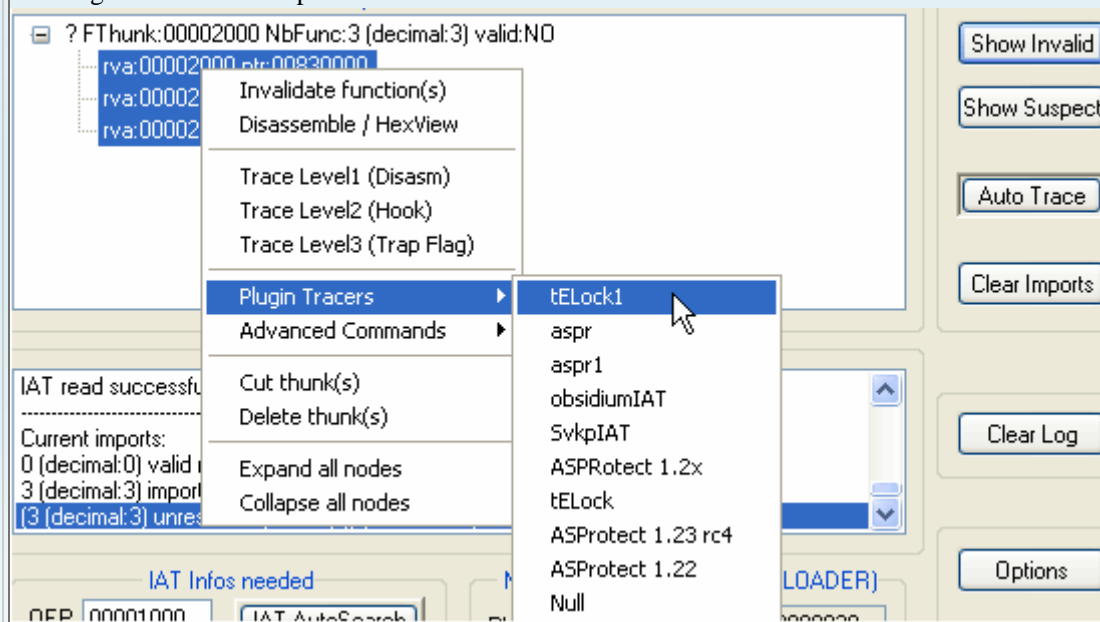
And select Open ImpREC attached to active process and choose unpackme.exe. Change the value in the OEP window to the one we wrote down earlier (1000) then select IATAutosearch then click Get Imports.



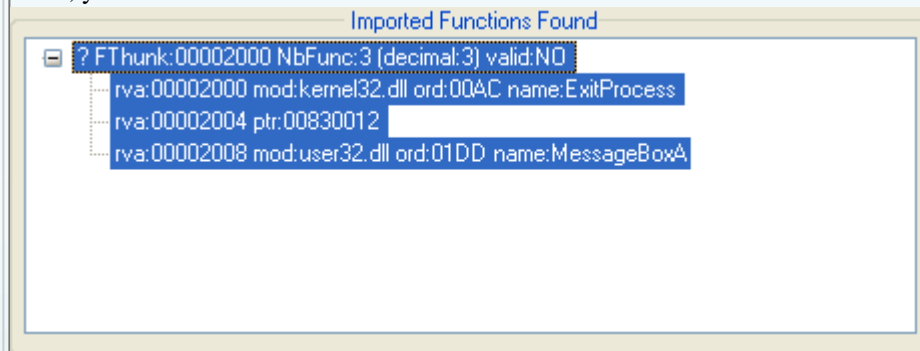
Humm, the unlucky import function is invalid. But no problem! Now, you must click on **Show Invalid** button.



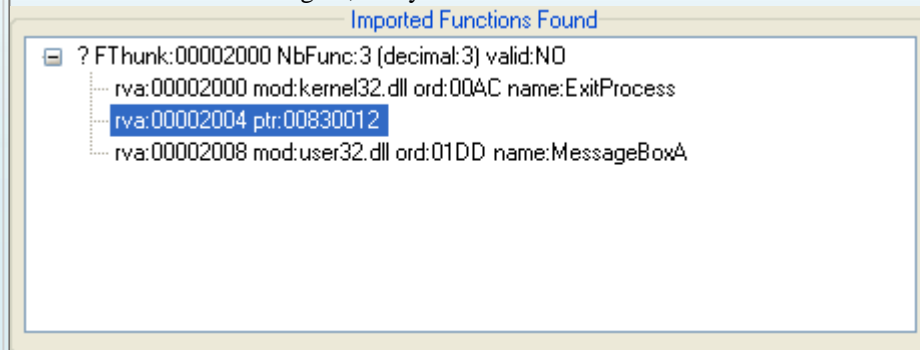
Then right click on the imports and choose:



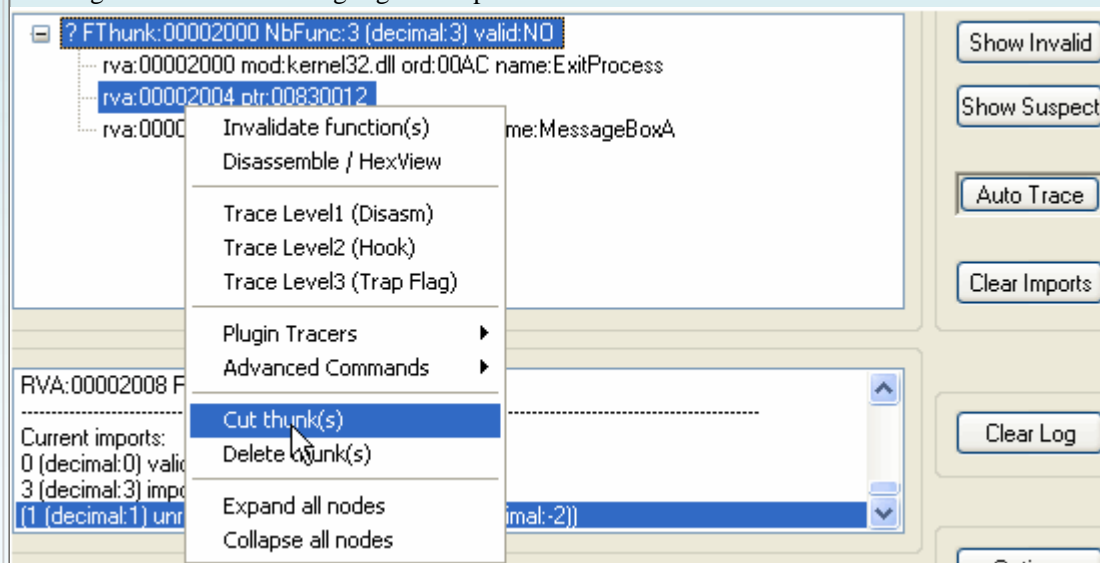
After, you see as follows:



Then click Show Invalid again, and you have:

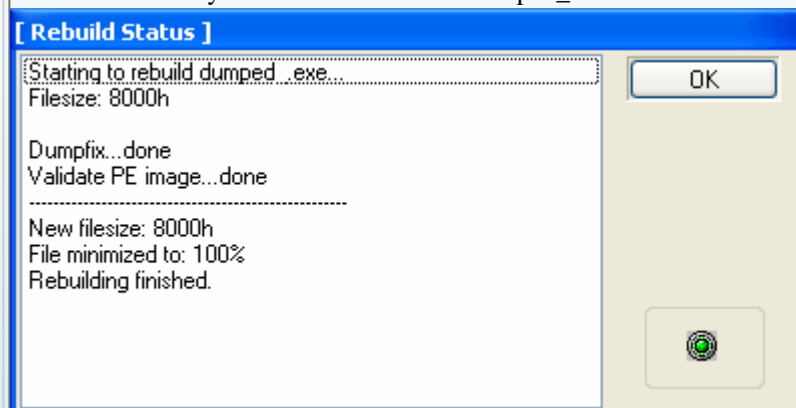


Then right-click one of the highlighted imports **Cut thunks** and choose:



Now, click fix dump to fix the IAT **dumped.exe** file.

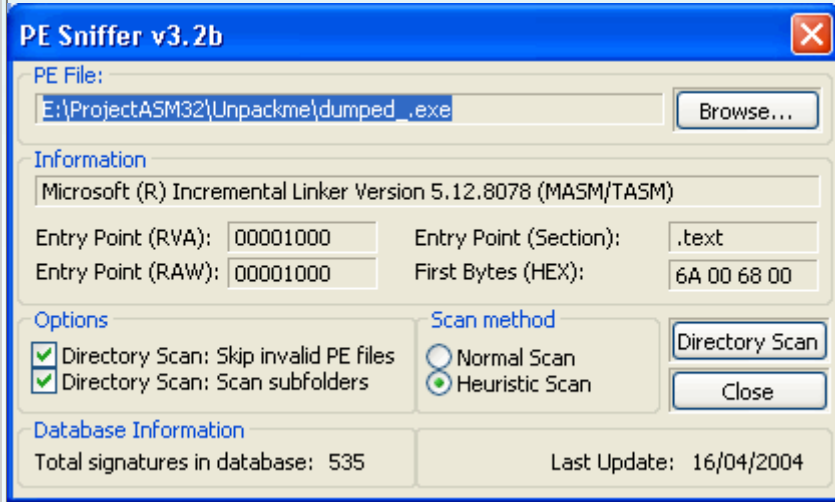
Use LordPE 1.4 by Y0da for rebuild our Dumped_.exe



6. Testing Our Unpacked file

Now run unpacked files. Wow, not crash.

Using **PE 3.2b** for **Sniffer** detect: **MASM / TASM**. Okie, tElock version 0.98b1 -> **in!** Is now unpacked successful!



7. Conclusion

Special thanx to **koncool** et **R @ dier** for this template.

My Greetz to: Deux, NVH (c), luucorp, Maip0301, R @ dier, tlandn, Computer_Angel, Zombie, RCA, CTL, JAL, LeVuHoang, 777, LeonHart, Bin ... and you ;-)!

To be continued ...

Written by [hacnho](#) (tutorial date: Sai Gon 10/4/2004)

Manual Unpacking Total Uninstall 3

EXECryptor 2.xx

I-Introduction:

Total Uninstall 3 in the pack but **ExeCryptor** methods to unpack Execryptor children have the opportunity to introduce the i can apply for this Target. Even **PeiD** and **RGD Packer Detector** mistakenly identified the **UPX**, very easy mistake and brought the wild oriented approach unpack, also cause that some parties SNDforum Pa vociferous Target with this. I evaluate authors Soft quite high because this is very smart when Protect soft as this. It would say in **Version 3.6x** ko **CRC Check** with the **new version 3.7** will have.

II-Tools & Target:

Tool and to use the Plugin:

- § **OLLYDBG_Execryptor 1:10**
- § **LordPE 1.4**
- § **ImportREC 1.6f**
- § **RDG Packer Detector v0.6.4**
- § **ID Protection 5.1f**
- § **CFF Explorer V**

Target: **Total Uninstall 3**

<http://www.martau.com/>

III-Bypass & AntiDebug Find OEP:

_Dung **RDG Packer Detector v0.6.4** scan target



ID PROTECTION _Dung v5.1f target scan



_Mo The CFF, the **Red Section** below may be signs of the pack with target **Execrypto R**.

CFF Explorer V - by Ntoskrnl - [Tu.exe]

Settings ?

Name	Virtual Size	Virtual Address	Raw Size	
Byte[8]	Dword	Dword	Dword	
CODE	00140000	00001000	0013F200	
DATA	00003000	00141000	00002200	
BSS	00002000	00144000	00000000	
.idata	00004000	00146000	00003C00	
dxmajpw1	00001000	0014A000	00001000	
.tls	00001000	0014B000	00000000	
.rdata	00001000	0014C000	00000200	
lrmarft9	00015000	0014D000	00015000	
.rsrc	0004E000	00162000	0004DA00	
w2b10ij9	0007F000	001B0000	0007EC03	
b2s.e91e	00001000	0022F000	00000200	

_ OK, Load OllyDBG_EXEcryptor on target and we stop here:

006149B3	55	PUSH EBP
006149B4	8BEC	MOV EBP, ESP
006149B6	53	PUSH EBX
006149B7	8BD9	MOV EBX, ECX
006149B9	871C24	XCHG DWORD PTR SS:[ESP], EBX
006149BC	- 0F80 D8E6F3FF	JO Tu.0055309A
006149C2	^ E9 6B60FAFF	JMP Tu.005BAA32
006149C7	67:64:8926 000	MOV DWORD PTR FS:[0], ESP
006149CD	8BFF	MOV EDI, EDI
006149CF	F1	INT1
006149D0	^ E9 7FDCFCFF	JMP Tu.005E2654
006149D5	C3	RETN
006149D6	81C1 412F1C49	ADD ECX, 491C2F41
006149DC	2BD9	SUB EBX, ECX
006149DE	59	POP ECX
006149DF	81E2 D3E9BA92	AND EDX, 92BAE9D3
006149E5	E8 ABEEFAFF	CALL Tu.005C3895
006149EA	1002	ADC BYTE PTR DS:[EDX], AL

_ Press **Alt + B 1 Breakpoint** you see, please delete it

Address	Module	Active	Disassembly
005618C3	Tu	One-shot	JMP Tu.005D41E5

Remove Del
Show in Disassembler Enter
Copy to clipboard
Appearance

_ Press **Alt + M** and press **F2** to set BP in Section 1. Code

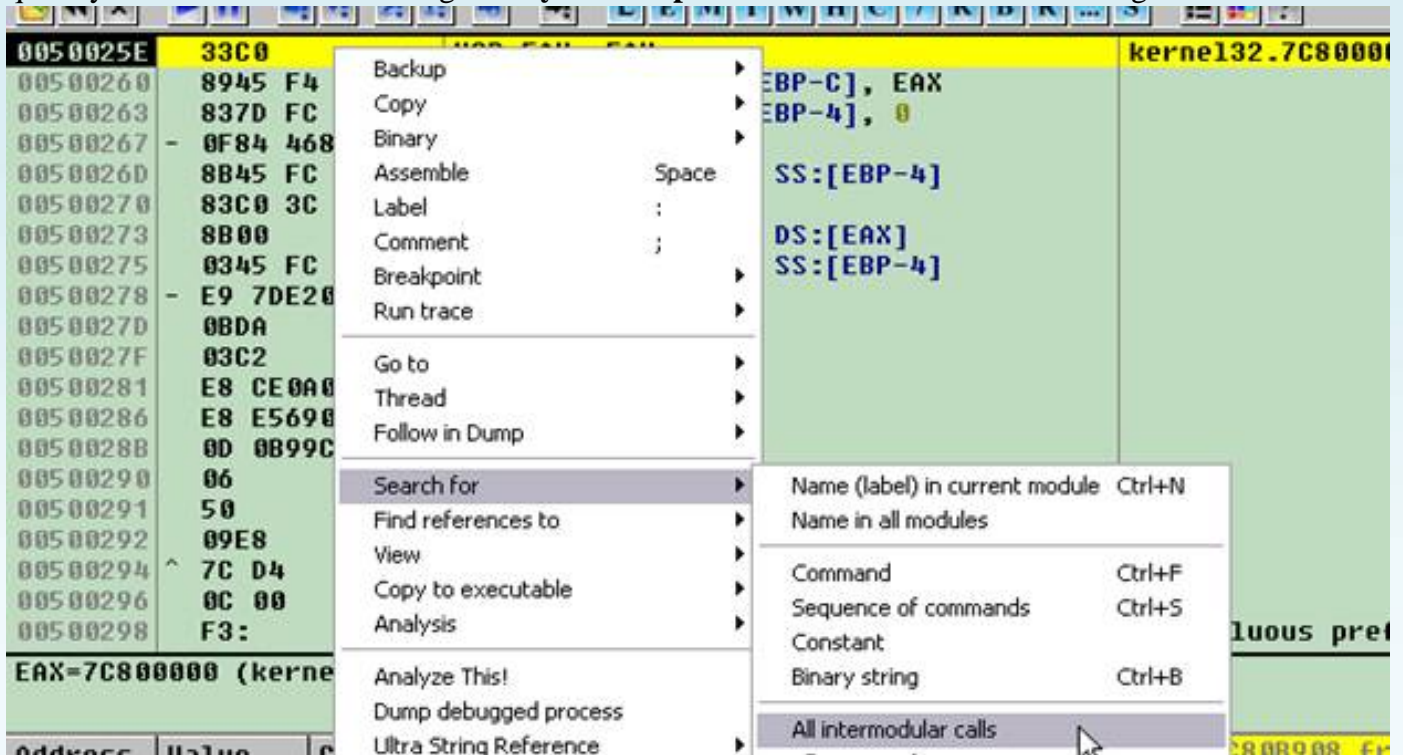
Address	Module	Type	Section	Priv	RW	RW
003F0000	00003000					
00400000	00001000	Tu	PE header	Imag	R	RWE
00401000	00140000	Tu	code	code	R	RWE
00541000	00003000	Tu	DATA	data		
00544000	00002000	Tu	BSS			
00546000	00004000	Tu	.idata	impor		
0054A000	00001000	Tu	dxmajpw1			
0054B000	00001000	Tu	.tls			
0054C000	00001000	Tu	.rdata			
0054D000	00015000	Tu	lrmarft9	SFX		
00562000	0004E000	Tu	.rsrc	resou		
005B0000	0007F000	Tu	w2b10ij9			
0062F000	00001000	Tu	b2s.e91e			
00630000	00004000					

Actualize
View in Disassembler Enter
Dump in CPU
Dump
Search Ctrl+B
Set break-on-access F2
Set memory breakpoint on access
Set memory breakpoint on write
Set access

_ Press **Shift + F9** you stop here

Address	Disassembly	Comment
0050025E	XOR EAX, EAX	
00500260	MOV DWORD PTR SS:[EBP-C], EAX	
00500263	CMPL DWORD PTR SS:[EBP-4], 0	
00500267	JE Tu.006085B3	
0050026D	MOV EAX, DWORD PTR SS:[EBP-4]	
00500270	ADD EAX, 3C	
00500273	MOV EAX, DWORD PTR DS:[EAX]	
00500275	ADD EAX, DWORD PTR SS:[EBP-4]	
00500278	JMP Tu.0058E4FA	
0050027D	OR EBX, EDX	
0050027F	ADD EAX, EDX	
00500281	CALL Tu.00550D54	
00500286	CALL Tu.00586C70	
0050028B	OR EAX, 00CF0000	

_ As we know Soft code using **Borland Delphi 6.0 - 7.0** Ham have always had **GetModuleHandleA** below. Based on this we quickly find the **OEP** of this Target. **OllyDBG** In press **F10** and select the same image



_ Type **GetModuleHandleA** and function 2

Address	Disassembly	Comment
00402A8C	CALL <JMP.&kernel32.GetModuleFileNameA>	kernel32.GetModuleFileNameA
00406466	CALL <JMP.&kernel32.GetModuleFileNameA>	kernel32.GetModuleFileNameA
004066A0	CALL <JMP.&kernel32.GetModuleFileNameA>	kernel32.GetModuleFileNameA
0040D7B9	CALL <JMP.&kernel32.GetModuleFileNameA>	kernel32.GetModuleFileNameA
0040D7D4	CALL <JMP.&kernel32.GetModuleFileNameA>	kernel32.GetModuleFileNameA
0040DF09	CALL <JMP.&kernel32.GetModuleFileNameA>	kernel32.GetModuleFileNameA
00417DC6	CALL <JMP.&kernel32.GetModuleFileNameA>	kernel32.GetModuleFileNameA
00417C27	CALL <JMP.&kernel32.GetModuleFileNameA>	kernel32.GetModuleFileNameA
00417D9D	CALL <JMP.&kernel32.GetModuleFileNameW>	kernel32.GetModuleFileNameW
004064E9	CALL <JMP.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA
0040711C	CALL <JMP.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA
0040EFE2	CALL <JMP.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA
00414BE6	CALL <JMP.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA
00415C8F	CALL <JMP.&kernel32.GetModuleHandleA>	kernel32.GetModuleHandleA

_ Click here to **double** our

00407109	E8 D2F8FFFF	CALL Tu.004069E0
0040710E	C3	RETN
0040710F	90	NOP
00407110	53	PUSH EBX
00407111	8BD8	MOV EBX, EAX
00407113	33C0	XOR EAX, EAX
00407115	A3 10475400	MOV DWORD PTR DS:[544710], EAX
0040711A	6A 00	PUSH 0
0040711C	E8 2BFFFFFF	CALL <JMP.&kernel32.GetModuleHandleA>
00407121	A3 18475400	MOV DWORD PTR DS:[544718], EAX
00407126	A1 18475400	MOV EAX, DWORD PTR DS:[544718]
0040712B	A3 B0105400	MOV DWORD PTR DS:[5410B0], EAX
00407130	33C0	XOR EAX, EAX
00407132	A3 B4105400	MOV DWORD PTR DS:[5410B4], EAX
00407137	33C0	XOR EAX, EAX
00407139	A3 B8105400	MOV DWORD PTR DS:[5410B8], EAX
0040713E	E8 C1FFFFFF	CALL Tu.00407104
00407143	BA 00405400	MOV EBX, Tu.00405400

_ We need to find a command to call call this code, press **Ctrl + F** to enter **call 407110** and click the **Find** it here

0053F9A7	00	DB 00
0053F9A8	A4F45300	DD Tu.0053F4A4
0053F9AC	- E9 121F0200	JMP Tu.<ModuleEntryPoint>
0053F9B1	. 0A0C73	OR CL, BYTE PTR DS:[EBX+ESI*2]
0053F9B4	> 6A 00	PUSH 0
0053F9B6	. 6A 00	PUSH 0
0053F9B8	. 49	DEC ECX
0053F9B9	. ^ 75 F9	JNZ SHORT Tu.0053F9B4
0053F9BB	. 51	PUSH ECX
0053F9BC	. 53	PUSH EBX
0053F9BD	. 56	PUSH ESI
0053F9BE	. 57	PUSH EDI
0053F9BF	. B8 CCF45300	MOV EAX, Tu.0053F4CC
0053F9C4	. E8 4777ECFF	CALL Tu.00407110
0053F9C9	. 33C0	XOR EAX, EAX
0053F9CB	. 55	PUSH EBP

Look up slightly _ is OEP ... Stolen Bytes but we need to Fix it, roll the mouse to **0053F9B4**, Press F2 Set 1 in which BP and press F9 will stop at **0053F9B4**

0053F9A5	00	DB 00	Registers (FPU) EAX 00000000 ECX 0000000C EDX 7C90EB94 ntdll EBX 7FFD5000 ESP 0012FFC0 EBP 0012FFC0 ESI FFFFFFFF EDI 7C910738 ntdll EIP 0053F9B4 Tu.00
0053F9A6	00	DB 00	
0053F9A7	00	DB 00	
0053F9A8	A4F45300	DD Tu.0053F4A4	
0053F9AC	- E9 121F0200	JMP Tu.<ModuleEntryPoint>	
0053F9B1	0A0C73	OR CL, BYTE PTR DS:[EBX+ESI*2]	
0053F9B4	> 6A 00	PUSH 0	
0053F9B6	. 6A 00	PUSH 0	
0053F9B8	. 49	DEC ECX	
0053F9B9	. ^ 75 F9	JNZ SHORT Tu.0053F9B4	
0053F9BB	. 51	PUSH ECX	C 1 ES 0023 32bit P 1 CS 001B 32bit A 0 SS 0023 32bit Z 0 DS 0023 32bit S 0 FS 0000 32bit
0053F9BC	. 53	PUSH EBX	
0053F9BD	. 56	PUSH ESI	
0053F9BE	. 57	PUSH EDI	
0053F9BF	. B8 CCF45300	MOV EAX, Tu.0053F4CC	
0053F9C4	. E8 4777ECFF	CALL Tu.00407110	

_Toi You look into the window and see **FPU** bar **ECX = 0x0C**. Note this is quite important for us to Fix **OEP**. We revised the language as follows:

Registers (FPU)

EAX	00000000
ECX	0000000C
EDX	7C90EB94 ntdll.KiFa
EBX	7FFD5000
ESP	0012FFC0
EBP	0012FFC0
ESI	FFFFFFFF
EDI	7C910738 ntdll.7C91
EIP	0053F9B4 Tu.0053F9B
C 1	ES 0023 32bit 0(FF
P 1	CS 001B 32bit 0(FF
A 0	SS 0023 32bit 0(FF
Z 0	DS 0023 32bit 0(FF
S 0	FS 003B 32bit 7FFD
T 0	GS 0000 NULL

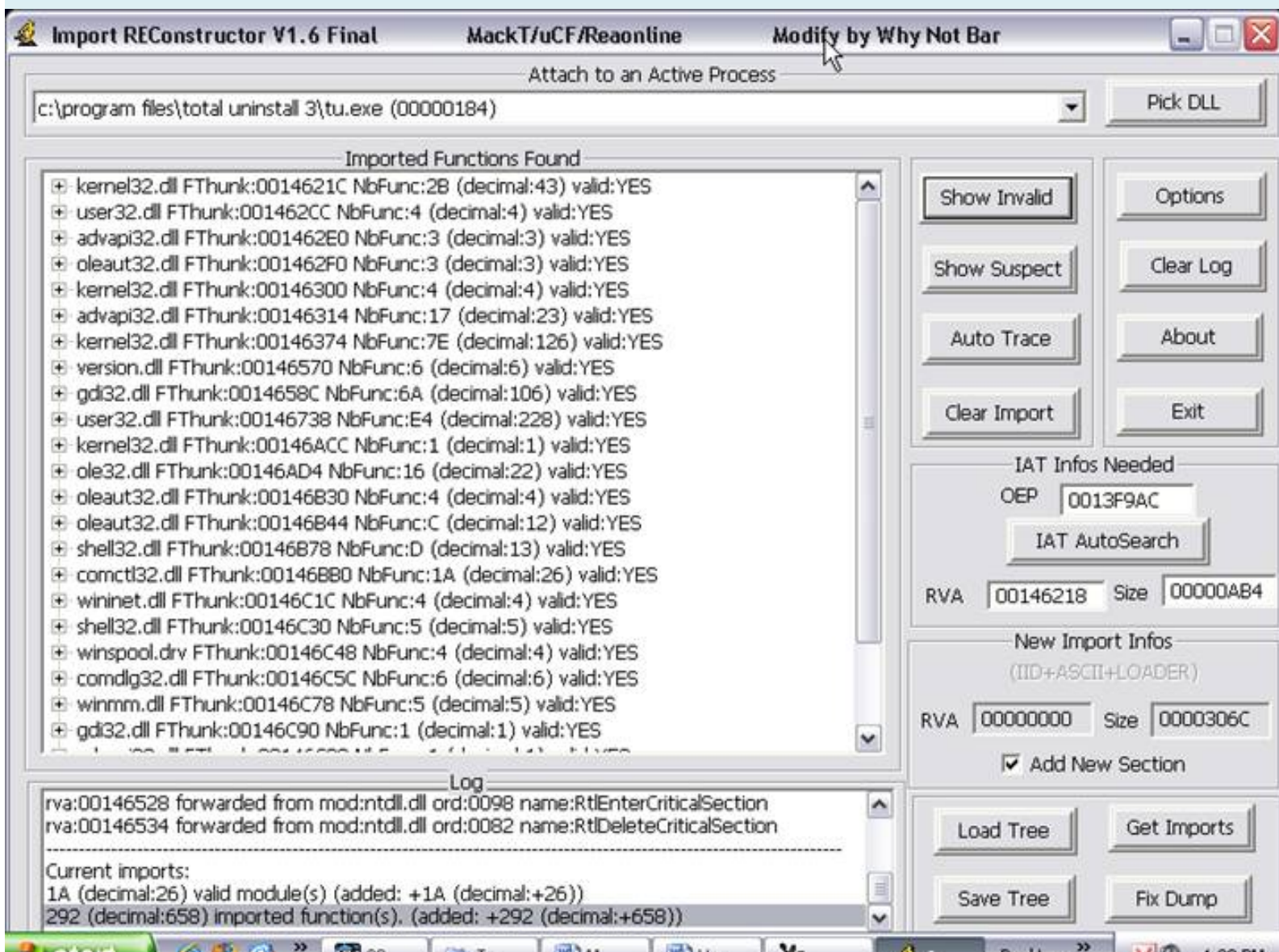
Thui _Gio dump file ...

Process List:

Process Name	PID	PPID	Working Set	Path
Explorer	0000068C	01000000	000FF000	C:\WINDOWS\Explorer.EXE
xmplay	00000394	00400000	0014D000	F:\Cool Data\multimedia\xmplay32\xmplay.e
WINWORD	0000062C	30000000	00BAA000	C:\Program Files\Microsoft Office\OFFICE11
UniKeyNT	000007E0	00400000	00041000	F:\Cool Data\Van Phong\uk408nt\UniKey\Un
HprSnap6	00000590	00400000	00419000	C:\Program Files\HyperSnap 6\HprSnap6.ex
OLLYDBG_Execryptor	00000708	00400000	00164000	D:\Crack tool\OlyDbg_Execryptor\OLLYDBG
Tu	00000184	00400000	00230000	C:\Program Files\Total Uninstall 3\Tu.exe
ImportRE	00000000	00400000	00089000	D:\Crack tool\ImpREC\ImportREConstructor
Task Expl	00000000	00400000	00076000	D:\Crack tool\Other\CFF Explorer\Task Expl

IV-rebuild Import & Fix CRC:

_ Open **ImportREC** up. Select **List** in Process **TU.exe**. **OEP** = Enter **0053F9AC - 00,400,000 (Imagebase) = 0013F9AC**, Click **IAT AutoSearch** à à **Get Imports Show Invalid**



_ Hú complete I have any function **Invalid** ... OK, Click **Fix Dumped Tu_Dumped.exe** File and select Run ... try.
 Hic ... hic ... Dek run Load **Tu_Dumped.exe** to **OllyDBG_Execryptor**, press **Shift + F9** to **EP**

0053F9AC	55	PUSH EBP
0053F9AD	8BEC	MOV EBP, ESP
0053F9AF	B9 0C000000	MOV ECX, 0C
0053F9B4	CC	INT3
0053F9B5	006A 00	ADD BYTE PTR DS:[EDX], CH
0053F9B8	49	DEC ECX
0053F9B9	75 F9	JNZ SHORT Tu_Dumpe.0053F9B4
0053F9BB	51	PUSH ECX
0053F9BC	53	PUSH EBX
0053F9BD	56	PUSH ESI
0053F9BE	57	PUSH EDI
0053F9BF	B8 CCF45300	MOV EAX, Tu_Dumpe.0053F4CC
0053F9C4	E8 4777ECFF	CALL Tu_Dumpe.00407110
0053F9C9	33C0	XOR EAX, EAX
0053F9CB	55	PUSH EBP
0053F9CD	68 0FFF5300	PUSH Tu_Dumpe.0053F50F

_ Tai 0053F9B4 revised as follows:

0053F9AC	55	PUSH EBP
0053F9AD	8BEC	MOV EBP, ESP
0053F9AF	B9 0C000000	MOV ECX, 0C
0053F9B4	6A 00	PUSH 0
0053F9B6	6A 00	PUSH 0
0053F9B8	49	DEC ECX
0053F9B9	75 F9	JNZ SHORT Tu_Dumpe.0053F9B4
0053F9BB	51	PUSH ECX

Ok, Save the file and try to run ... khua khua still running Dek Load File Again ... just to save OllyDBG_Execryptor, with this format is the best way is Trace between F7 and F8 since then to find from just Fix, but how This is quite long but absolutely accurate. Here they work quickly, press Ctrl + G to enter 00404628 and press F2 Set at BP

00404600	55	PUSH EBP
00404601	8BEC	MOV EBP, ESP
00404603	53	PUSH EBX
00404604	56	PUSH ESI
00404605	57	PUSH EDI
00404606	A1 3C465400	MOV EAX, DWORD PTR DS:[54463C]
00404608	85C0	TEST EAX, EAX
0040460D	74 4B	JE SHORT Tu_Dumpe.0040465A
0040460F	8B30	MOV ESI, DWORD PTR DS:[EAX]
00404611	33DB	XOR EBX, EBX
00404613	8B78 04	MOV EDI, DWORD PTR DS:[EAX+4]
00404616	33D2	XOR EDX, EDX
00404618	55	PUSH EBP
00404619	68 46464000	PUSH Tu_Dumpe.00404646
0040461E	64:FF32	PUSH DWORD PTR FS:[EDX]
00404621	64:8922	MOV DWORD PTR FS:[EDX], ESP
00404624	3BF3	CMP ESI, EBX
00404626	7E 14	JLE SHORT Tu_Dumpe.0040463C
00404628	8B04DF	MOV EAX, DWORD PTR DS:[EDI+EBX*8]
0040462B	43	INC EBX
0040462C	891D 40465400	MOV DWORD PTR DS:[544640], EBX
00404632	85C0	TEST EAX, EAX
00404634	74 02	JE SHORT Tu_Dumpe.00404638
00404636	FFD0	CALL NEAR EAX
00404638	3BF3	CMP ESI, EBX
0040463A	7F EC	JG SHORT Tu_Dumpe.00404628

Nhan For Shift + F9 until Soft Crash ... (press Shift + F9 a lot of times), times hit Shift + F9 before Crash soft signs are the following:

0040460F	8B30	MOV ESI, DWORD PTR DS:[EAX]	
00404611	33DB	XOR EBX, EBX	
00404613	8B78 04	MOV EDI, DWORD PTR DS:[EAX+4]	
00404616	33D2	XOR EDX, EDX	
00404618	55	PUSH EBP	
00404619	68 46464000	PUSH Tu_Dumpe.00404646	
0040461E	64:FF32	PUSH DWORD PTR FS:[EDX]	
00404621	64:8922	MOV DWORD PTR FS:[EDX], ESP	
00404624	3BF3	CMP ESI, EBX	
00404626	7E 14	JLE SHORT Tu_Dumpe.0040463C	
00404628	8B04DF	MOV EAX, DWORD PTR DS:[EDI+EBX*8]	Tu_Dumpe.004FF3
0040462B	43	INC EBX	
0040462C	891D 40465400	MOV DWORD PTR DS:[544640], EBX	
00404632	85C0	TEST EAX, EAX	
00404634	74 02	JE SHORT Tu_Dumpe.00404638	
00404636	FFD0	CALL NEAR EAX	
00404638	3BF3	CMP ESI, EBX	
0040463A	7F EC	JG SHORT Tu_Dumpe.00404628	
0040463C	33C0	XOR EAX, EAX	
0040463E	5A	POP EDX	
00404640	5B	POP EBP	

DS:[0053F87C]=004FF3F8 (Tu_Dumpe.004FF3F8) → Dia chi doan code Check CRC
EAX=00041CFC

Ctrl + F2, Ctrl + G to enter 004FF3F8

004FF3F5	C3	RETN	
004FF3F6	8BC0	MOV EAX, EAX	
004FF3F8	832D 28535400	SUB DWORD PTR DS:[545328], 1	
004FF3FF	73 1A	JNB SHORT Tu_Dumpe.004FF41B	
004FF401	E9 54A00D00	JMP Tu_Dumpe.005D945A	
004FF406	FA	CLI	
004FF407	62AD 57B87BB7	BOUND EBP, QWORD PTR SS:[EBP+B77BB857]	
004FF40D	0C 1B	OR AL, 1B	
004FF40F	838E 38402C2D	OR DWORD PTR DS:[ESI+2D2C4038], 79	
004FF416	71 A8	JNO SHORT Tu_Dumpe.004FF3C0	
004FF418	38AE 29C368F4	CMP BYTE PTR DS:[ESI+F468C329], CH	
004FF41F	4F	DEC EDI	

Patch _Va as follows:

004FF3F5	C3	RETN	
004FF3F6	8BC0	MOV EAX, EAX	
004FF3F8	C3	RETN	
004FF3F9	90	NOP	
004FF3FA	90	NOP	
004FF3FB	90	NOP	
004FF3FC	90	NOP	
004FF3FD	90	NOP	
004FF3FE	0173 1A	ADD DWORD PTR DS:[EBX+1A], ESI	
004FF401	E9 54A00D00	JMP Tu_Dumpe.005D945A	
004FF406	FA	CLI	
004FF407	62AD 57B87BB7	BOUND EBP, QWORD PTR SS:[EBP+B77BB857]	

_Save Run the file and try khua khua ... this is a good run ... unpack Done!

GrEeTs Fly Out: Computer_Angel, Zombie, Moonbaby, Hacnho, Benina, kienmanowar, Zoi, Deux, Merc, light Phoenix, Trickyboy, Takada, iamidiot, thienthandien, ... and you!

Nha Trang, the 1st 11, 2006

Manual unpack Execryptor 2.x

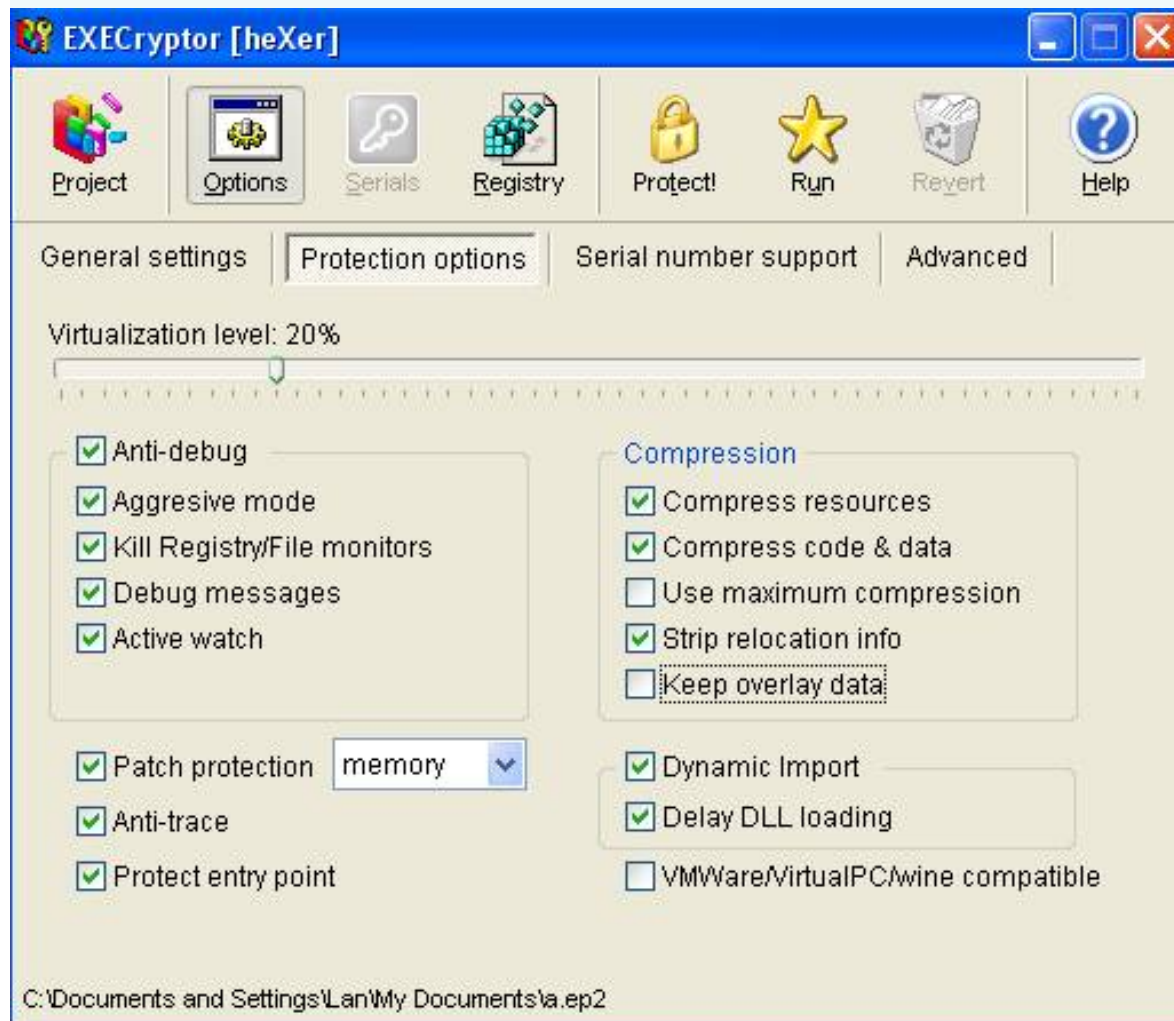
tlandn

Target: BitArray.exe (included)

Tools: Diablo OllyDbg or OllyDbg_Execryptor (download the tut about Execryptor by Why_Not_Bar)

Welcome you. Time is not correct to unpack. Both then read some tut Why_Not_Bar he's on Execryptor should see or learn more. In this tut I try to put all the methods (that I know) to unpack Execryptor.

We will make a program for the demo. I select the BitArray.exe. This is a program written in MFC. The aim is to give them tí (tut because now almost all of the programs written in VC + +). I used to pack Execryptor 2.2.6. I Options as follows:



OK. We start.

I) Determining Packer:

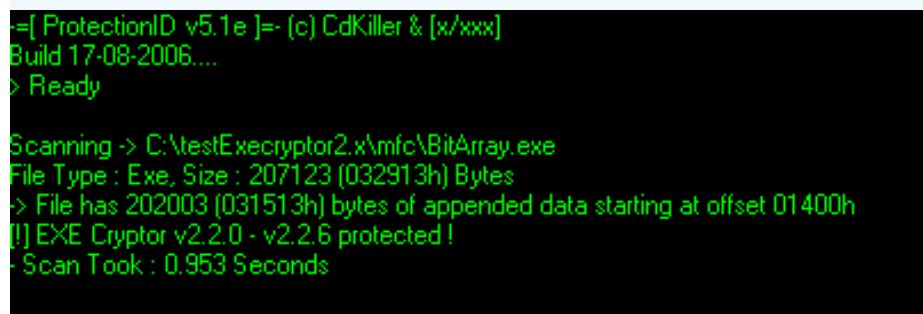
We can use PEID to check the program:



Or use RDP Packer Detector 0.6.4 beta (method B):



Or use the newer Protection 5.1e ID:



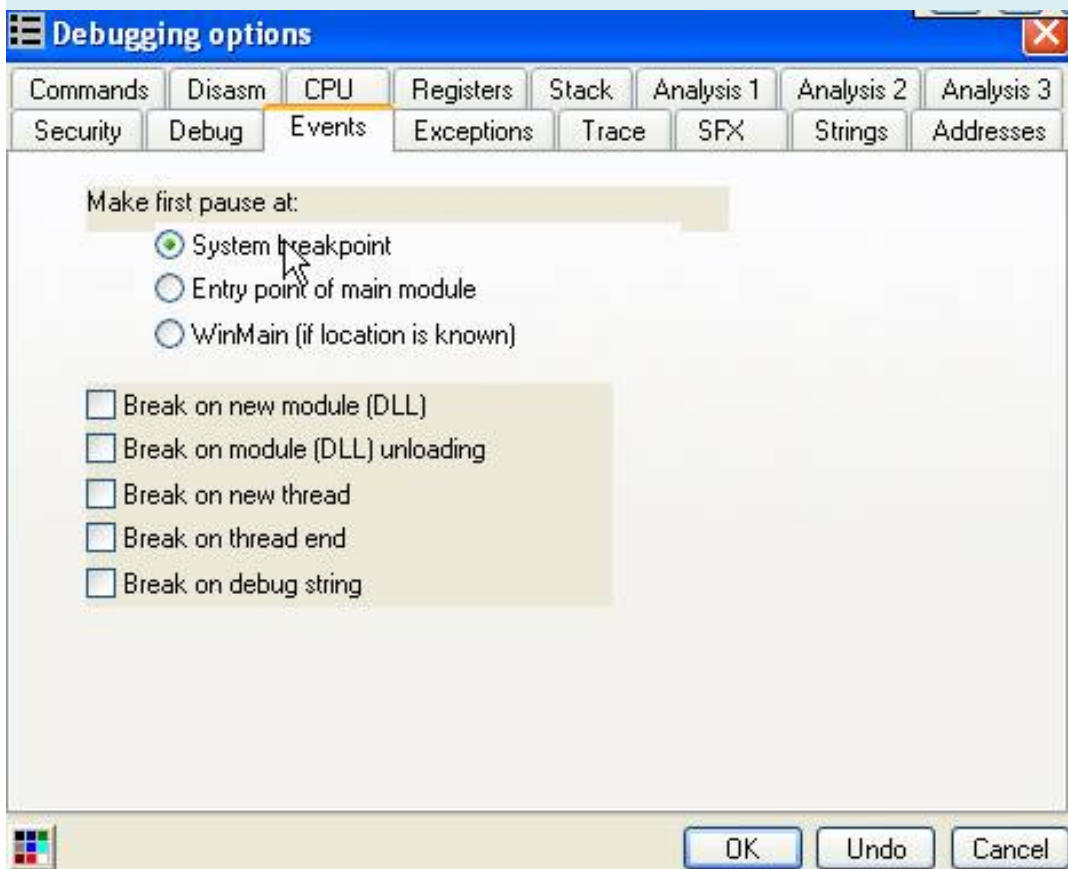
So just under 3 months if the program we are clearly using the pack Execryptor J 2.x

II) Find False OEP:

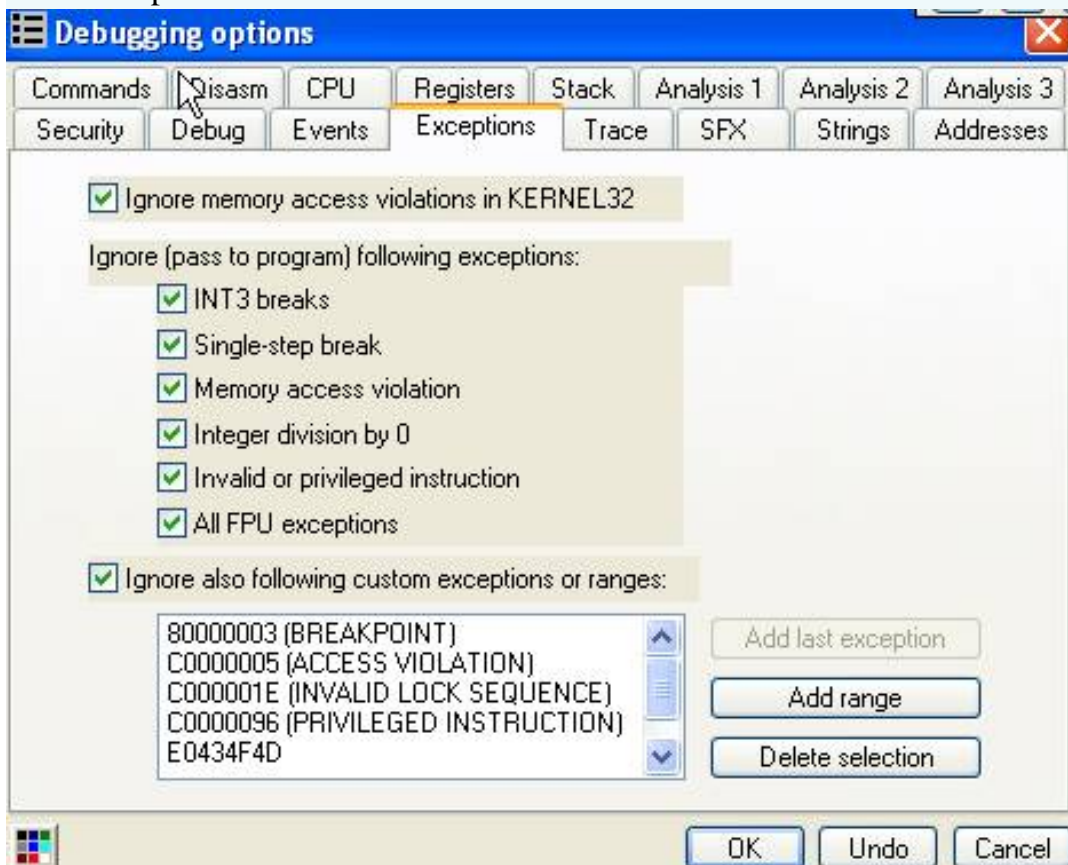
1) Method 1:

Running OllyDbg (I xài Diablo OllyDbg see this month because xài sướng and stability). Press Alt-O following the same tune:

The "Events":



The "Exceptions":



Done and then load the program to BitArray.exe. We stop here:

7C901231	C3	RETN
7C901232	8BFF	MOV EDI, EDI
7C901234	90	NOP
7C901235	90	NOP

Looking down the Status bar:

System startup breakpoint

Press Alt-B to the window "Breakpoint." Delete the breakpoint. Select the image as:

Address	Module	Active	Disassembly
004674EF	BitArray	One-shot	CALL BitArray.004673EB

Remove Del
Follow in Disassembler Enter
Copy to clipboard
Appearance

Remove and then finished press Alt-M to window "Memory Map". Set in Breakpoint section. Text. Image:

00320000	00006000			Map	00041002	R
00400000	00001000	BitArray	PE header	Imag	01001002	R
00401000	00002000	BitArray	.text			R
00403000	00001000	BitArray	p0qfq2uv	Actualize		
00404000	00001000	BitArray	.data	Dump in CPU		
00405000	00002000	BitArray	.rsrc	Dump		
00407000	00001000	BitArray	jjd5kura	Search		Ctrl+B
00408000	0002E000	BitArray	5d6hodfv			
00436000	00032000	BitArray	jqean7n7	code, imports		
77D40000	00001000	user32	PE header	Set break-on-access		F2
77D41000	0005F000	user32	.text	code, imports		
77DA0000	00002000	user32	.data	Set memory breakpoint on access		
77DA2000	0002B000	user32	.rsrc	resources		
				Set memory breakpoint on write		

Press Shift-F9. We stop here:

0045D8FE	F3:A4	REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	
0045D900	31DB	XOR EBX,EBX	
0045D902	5E	POP ESI	BitArray.0043617E
0045D903	EB 9D	JMP SHORT BitArray.0045D8A2	
0045D905	89F0	MOV EAX,ESI	BitArray.00436162
0045D907	5B	POP EBX	BitArray.0043617E
0045D908	5F	POP EDI	BitArray.0043617E
0045D909	5E	POP ESI	BitArray.0043617E
0045D90A	C3	RETN	

Press F2 set breakpoint at 0045D90A.

0045D8FE	F3:A4	REP MOVSB BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	
0045D900	31DB	XOR EBX,EBX	
0045D902	5E	POP ESI	
0045D903	EB 9D	JMP SHORT BitArray.0045D8A2	
0045D905	89F0	MOV EAX,ESI	
0045D907	5B	POP EBX	
0045D908	5F	POP EDI	
0045D909	5E	POP ESI	
0045D90A	C3	RETN	

Press Alt-M. Delete breakpoint on the section. Text. This type:

00370000	00001000				Priv 00021004	RW
00400000	00001000	BitArray		PE header	Imag 01001002	R
00401000	00002000	BitArray	.text			
00403000	00001000	BitArray	p0qfq2uv		Actualize	
00404000	00001000	BitArray	.data	data	Dump in CPU	
00405000	00002000	BitArray	.rsrc	resources	Dump	
00407000	00001000	BitArray	jjd5kura		Search	Ctrl+B
00408000	0002E000	BitArray	5d6hodfv			
00436000	00032000	BitArray	jqean7n7	code, imports	Set break-on-access	F2
00470000	00004000					
00530000	00002000				Set memory breakpoint on access	
00540000	00103000				Set memory breakpoint on write	
00650000	0009B000					
77D40000	00001000	user32		PE header	Remove memory breakpoint	
77D41000	0005F000	user32	.text	code, imports	Set access	
77D42000	00002000	user32	.data	data		

Press Shift-F9. We stopped at 0045D90A.

0045D907	5B	POP EBX
0045D908	5F	POP EDI
0045D909	5E	POP ESI
0045D90A	C3	RETN

Press F2 to remove breakpoint. Press Alt-M. Set breakpoint in the section. Text.

00400000	00001000	BitArray		PE header	Imag 01001002	R
00401000	00002000	BitArray	.text		Imag 01001002	R
00403000	00001000	BitArray	p0qfq2uv		Actualize	
00404000	00001000	BitArray	.data	data	Dump in CPU	
00405000	00002000	BitArray	.rsrc	resources	Dump	
00407000	00001000	BitArray	jjd5kura		Search	Ctrl+B
00408000	0002E000	BitArray	5d6hodfv			
00436000	00032000	BitArray	jqean7n7	code, imports	Set break-on-access	F2
00470000	00004000					
00530000	00002000				Set memory breakpoint on access	
00540000	00103000					

Press Shift-F9. We here:

0043613E	AC	LODS BYTE PTR DS:[ESI]
0043613F	D0E8	SHR AL,1
00436141	80F8 74	CMP AL,74
00436144	75 0E	JNZ SHORT BitArray.00436154
00436146	8B06	MOV EAX,DWORD PTR DS:[ESI]
00436148	0FC8	BSWAP EAX
0043614A	01C8	ADD EAX,ECX
0043614C	8906	MOV DWORD PTR DS:[ESI],EAX
0043614E	83C6 04	ADD ESI,4
00436151	83E9 04	SUB ECX,4
00436154	49	DEC ECX
00436155	7F E7	JG SHORT BitArray.0043613E
00436157	59	POP ECX
00436158	5E	POP ESI
00436159	C3	RETN

Continue set breakpoint (F2) at 00,436,159 (RETN). Press Alt-M remove breakpoint in section. Text. Then press Shift-F9. We will stop at 00,436,159.

00436155	7F E7	JG SHORT BitArray.0043613E
00436157	59	POP ECX
00436158	5E	POP ESI
00436159	C3	RETN

Press F2 remove breakpoint. Continue to press Alt-M breakpoint set in section. Text.

00400000	00001000	BitArray		PE header	Imag 01001002	R
00401000	00002000	BitArray	.text		Imag 01001002	R
00403000	00001000	BitArray	p0qfq2uv			
00404000	00001000	BitArray	.data	data		
00405000	00002000	BitArray	.rsrc	resources		
00407000	00001000	BitArray	jjd5kura			
00408000	0002E000	BitArray	5d6hodfv			
00436000	00032000	BitArray	jqean7n7	code, imports		
00470000	00004000					
00530000	00002000					
00540000	00103000					

Actualize

Dump in CPU

Dump

Search

Ctrl+B

Set break-on-access

F2

Set memory breakpoint on access

Press Shift-F9. We stopped at 00,402,971.

00402971	50	PUSH EAX
00402972	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
00402979	83EC 68	SUB ESP,68
0040297C	53	PUSH EBX
0040297D	56	PUSH ESI
0040297E	57	PUSH EDI

Here I would say 2 more hours to go to 00,402,971.

Method 2:

We use OEP Finder's deroko xyz. Choose the type:

oeppfinder vX.Y.Z by deroko/ARTeam

app C:\testExeCryptor2.x\mfc\BitArray.e

Sections

start : 0x00401000 - range : 0x00002000

☐ Custom range

user start range

Select tracing engine :

☐ deroko/ARTeam

☐ Trace all (using T flag)

☒ stealth (exeCryptor for example)

☒ Extra fast (not accurate sometimes)

☐ don't hook CreateThread

Current opcode/stolen bytes with deattach :

trace

deattach

sections

About

Exit

Click the "trace". We have notified:

continue?

eip in your range : 0x00402971

OK

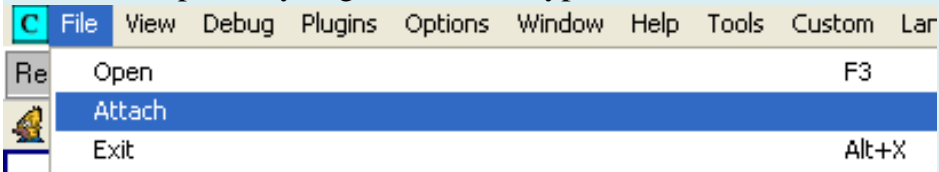
Cancel

Click "Cancel".



You remember the bytes "50 64" nhé.

Click OK. Open OllyDbg. Choose the type:



Select program BitArray. Click "Attach":

000006D0	alg	C:\WINDOWS\System32\alg.exe
00000F40	BitArray	C:\testExecryptor2.x\nfc\BitArray.exe
0000022C	csrss	\??C:\WINDOWS\system32\csrss.exe

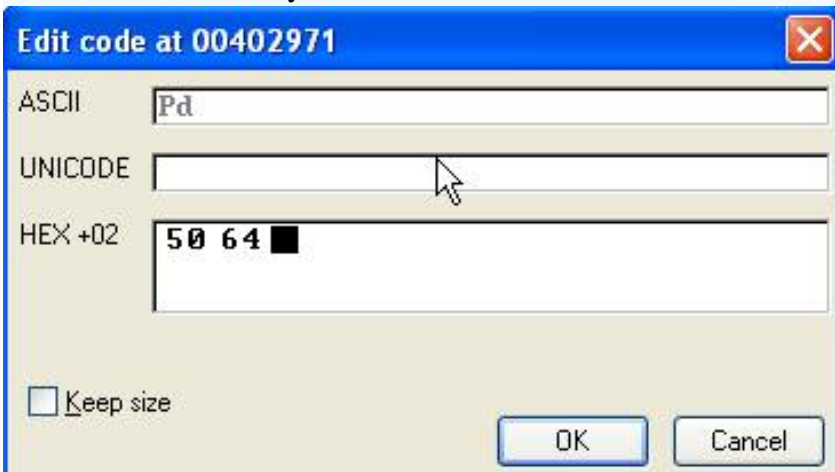
We here.

7C901231	C3	RETN
7C901232	8BFF	MOV EDI,EDI

Press F9 and F12. We stop here:

00402971	- EB FE	JMP SHORT 00402971
00402973	8925 00000000	MOV DWORD PTR DS:[0],ESP
00402979	83EC 68	SUB ESP,68

Press Ctrl-E. Enter bytes are "50 64".



Click OK.

00402971	50	PUSH EAX
00402972	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
00402979	83EC 68	SUB ESP,68
0040297C	53	PUSH EBX

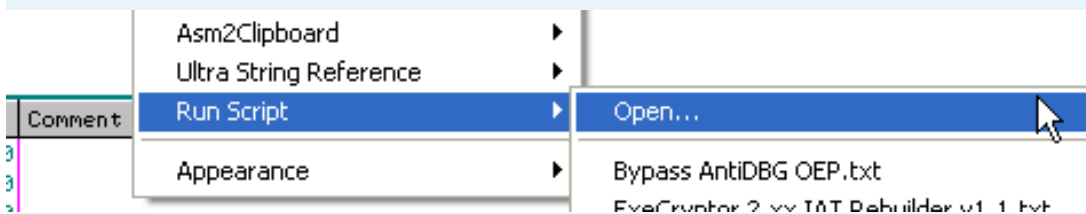
So how is finished 2nd. This fastest.

3 months:

We use the script "AntiDBG OEP.txt Bypass. Running OllyDbg. Open file BitArray. We stop here:

7C901231	C3	RETN
7C901232	8BFF	MOV EDI,EDI

We run the script. Click your mouse to select the image:



Select File "AntiDBG OEP.txt Bypass. An error message up:



We wear it shelves. Click OK. Then Shift-F9. With any error messages show up at the OK and then Shift-F9 until "Script Finished". We stop here:

0042C0F1	E8 168BFEFF	CALL BitArray.00414C0C
0042C0F6	2373 CF	AND ESI,DWORD PTR DS:[EBX-31]
0042C0F9	DC2C04	FSUBR QWORD PTR SS:[ESP+EAX]
0042C0FC	C00D E877B2FF FF	ROR BYTE PTR DS:[FFB277E8],0FF

Press Alt-M. Set breakpoint in the section. Text.

00400000	00001000	BitArray	PE header	Imag 01001002	R
00401000	00002000	BitArray	.text		
00403000	00001000	BitArray	hvp2amge		
00404000	00001000	BitArray	.data	data	
00405000	00002000	BitArray	.rsrc	resources	
00407000	00001000	BitArray	pu1kii0j		
00408000	0002E000	BitArray	i86hww0x		
00436000	00034000	BitArray	10libe25	code, imports	
00470000	00004000				
00530000	00002000				
00540000	00103000				

Press Shift-F9. We stopped at 00,402,971.

00402971	50	PUSH EAX
00402972	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
00402979	83EC 68	SUB ESP,68

So, with 3 ways. We achieved the same results as the program stopped in False OEP 00402971.

For some programs you try Method 1 is not the test 2 hours, 3 minutes. In the program I have encountered it at least a 3 on the way to successful J OK, so you get false OEP then we should do next? We continue.

III) Find and fix Stolen OEP Bytes:

The first is we must identify the program is written in what language? We will use and dump RDG Packer Detector to determine.

We are stopped at a false OEP 00402971. We will dump. Click your mouse to select the image:

00402971	50	PUSH EAX	Backup	▶
00402972	64:8925 00000000	MOV DWORD	Copy	▶
00402979	83EC 68	SUB ESP, 68	Binary	▶
0040297C	53	PUSH EBX	Assemble	Space
0040297D	56	PUSH ESI	Label	:
0040297E	57	PUSH EDI	Comment	;
0040297F	8965 E8	MOV DWORD	Breakpoint	▶
00402982	33DB	XOR EBX, E8	Run trace	▶
00402984	895D FC	MOV DWORD	Go to	▶
00402987	6A 02	PUSH 2	Follow in Dump	▶
00402989	FF15 DC314000	CALL DWORD	Search for	▶
0040298F	59	POP ECX	Find references to	▶
00402990	830D 18414000 FF	OR DWORD	View	▶
00402997	830D 1C414000 FF	OR DWORD	Copy to executable	▶
0040299E	FF15 D8314000	CALL DWORD	Analysis	▶
004029A4	8B0D 0C414000	MOV ECX, 0	Analyze This!	▶
004029AA	8908	MOV DWORD	Bookmark	▶
004029AC	FF15 D4314000	CALL DWORD	Code Ripper	▶
004029B2	8B0D 08414000	MOV ECX, 0	Dump debugged process	▶
004029B8	8908	MOV DWORD		
004029BA	A1 D0314000	MOV EAX, 0		
004029BF	8B00	MOV EAX, 0		
004029C1	A3 14414000	MOV DWORD		
004029C6	E8 16010000	CALL Bitf		
004029CB	391D 20404000	CMP DWORD		
004029D1	75 0C	JNZ SHOR		
004029D3	68 DE2A4000	PUSH Bitf		
004029D8	FF15 CC314000	CALL DWORD		

Then select the following:

OllyDump - BitArray.exe

Start Address: 400000 Size: 6A000 **Dump**

Entry Point: 69E8A -> Modify: 2971 **Get EIP as OEP** **Cancel**

Base of Code: 36000 Base of Data: 3000

☒ Fix Raw Size & Offset of Dump Image

Sect...	Virtual Size	Virtual Off...	Raw Size	Raw Offset	Charactaris...
.text	00002000	00001000	00002000	00001000	E0000020
hvp2a...	00001000	00003000	00001000	00003000	E0000060
.data	00001000	00004000	00001000	00004000	C0000040
.rsrc	00002000	00005000	00002000	00005000	C0000040
pu1kioj	00001000	00007000	00001000	00007000	C0000040
i86hw...	0002E000	00008000	0002E000	00008000	E0000020
10libe25	00034000	00036000	00034000	00036000	E0000060

☐ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

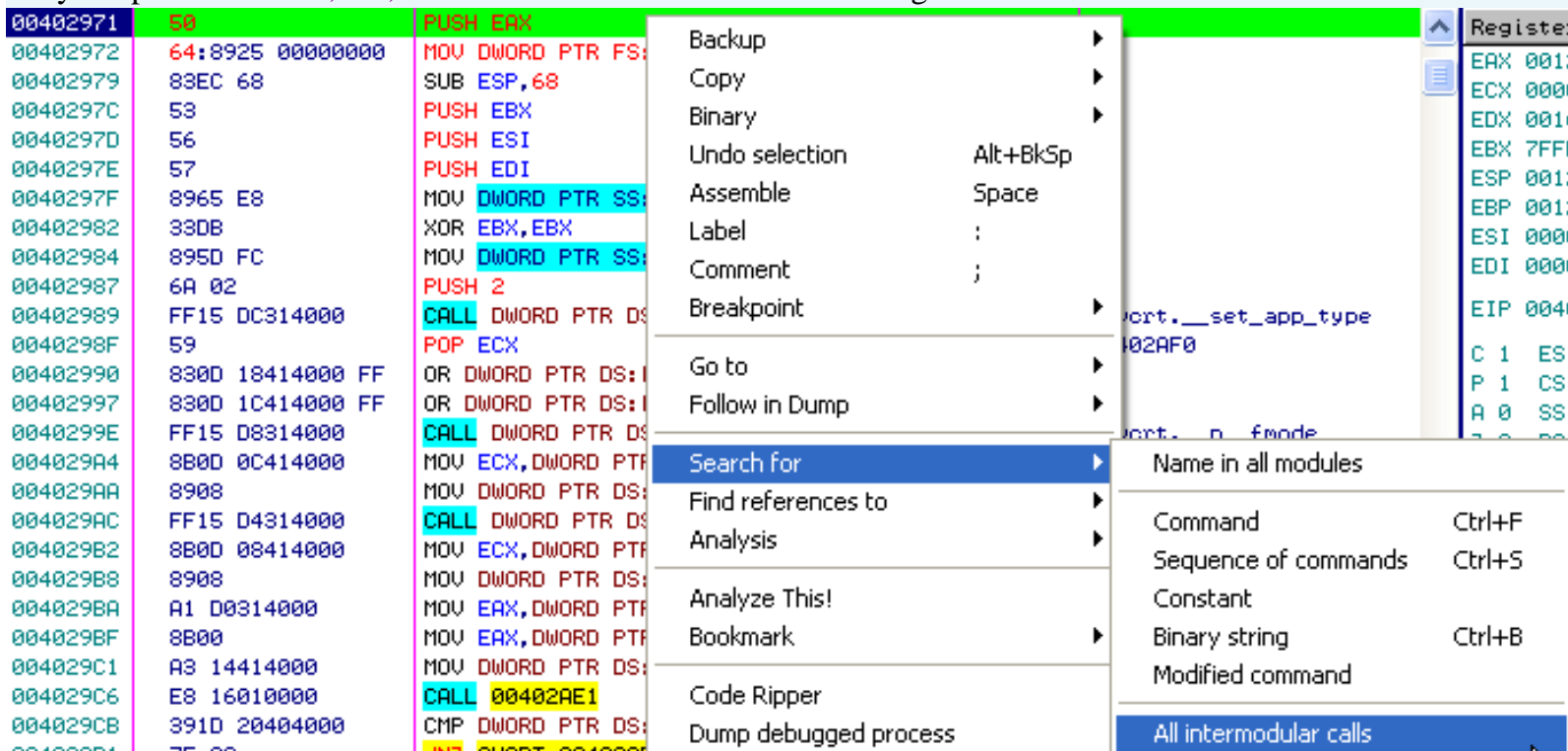
☐ Method2 : Search DLL & API name string in dumped file

Name the file a.exe. RDG then used to check the file a.exe

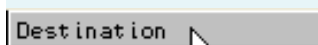


We see the program is written in VC++ 6. However, you should remember that the VC++ 6 with MFC and do not have OEP completely different. We must determine the exact program we MFC or not?

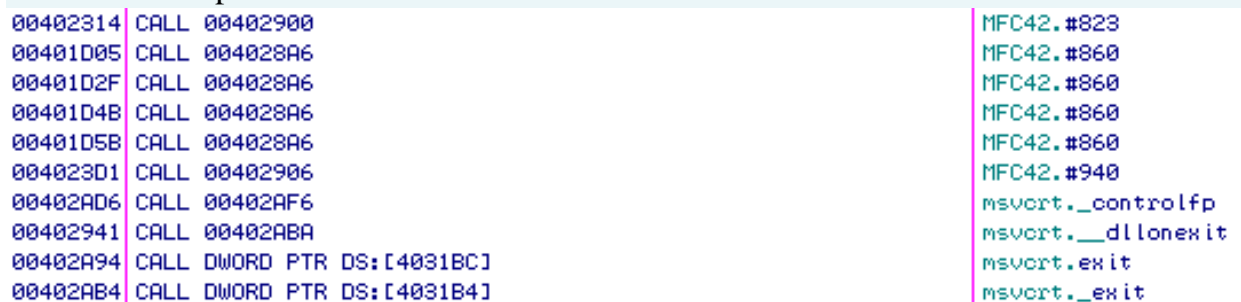
Very simple. We are 00402971. Click the mouse to select the image:



Click the title "Destination" to reorder the API. As in the picture:



Done and then pulled down a list. We found:



The program we have the function xài MFC42.XXXX

So clearly we are programs written in MFC.

The next is for a program written in MFC to consider signs in OEP how. I make the program "Keygenme # 1's 0x87k GAS". You can use any programs you do MFC found.

Open OllyDbg new (I call this a Olly2, OllyDbg beginning is our Olly1) loaded "Keygenme # 1's 0x87k GAS" to. Press F9. I have in OEP:

004088E4	55	PUSH EBP	
004088E5	8BEC	MOV EBP,ESP	
004088E7	6A FF	PUSH -1	
004088E9	68 F8A64000	PUSH +gAs+.0040A6F8	
004088EE	68 DE884000	PUSH <JMP.&MSVCRT.__except_handler3>	SE handler installation
004088F3	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
004088F9	50	PUSH EAX	
004088FA	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00408901	83EC 68	SUB ESP,68	
00408904	53	PUSH EBX	
00408905	56	PUSH ESI	
00408906	57	PUSH EDI	ntdll.7C910738
00408907	8965 E8	MOV [LOCAL.6],ESP	
0040890A	33DB	XOR EBX,EBX	
0040890C	895D FC	MOV [LOCAL.1],EBX	
0040890F	6A 02	PUSH 2	
00408911	FF15 38A24000	CALL DWORD PTR DS:[&MSVCRT.__set_app_type]	MSVCRT.__set_app_type
00408917	59	POP ECX	kernel32.7C816D4F
00408918	830D 58C54100 FF	OR DWORD PTR DS:[41C558],FFFFFFFF	
0040891F	830D 5CC54100 FF	OR DWORD PTR DS:[41C55C],FFFFFFFF	

We see the first function is called in the MFC is MSVCRT.__set_app_type".

In Olly1. We make this function, we found here (in the window Intermodular Found Calls):

00436479	CALL DWORD PTR DS:[4031F0]	msvcrt.realloc
00402B30	CALL DWORD PTR DS:[4031F4]	msvcrt._setmbcp
004029D8	CALL DWORD PTR DS:[4031CC]	msvcrt.__setusermatherr
00402989	CALL DWORD PTR DS:[4031DC]	msvcrt.__set_app_type
0044151E	CALL SHELL32.7CA01847	SHELL32.7CA01847
00402AA6	CALL 00402AC0	msvcrt._XoptFilter

Double-click the mouse on 00402989. We are here to:

0040295B	C3	RET	
0040295C	✓ E9 90970200	JMP 0042C0F1	
00402961	81E8 E241CEF0	SUB EAX,F0CE41E2	
00402967	✓ E9 94560000	JMP 00408000	
0040296C	0E	PUSH CS	
0040296D	CB	RET	Far return
0040296E	5C	POP ESP	00402AF0
0040296F	C3	RET	
00402970	✓ 74 50	JE SHORT 004029C2	
00402972	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00402979	83EC 68	SUB ESP,68	
0040297C	53	PUSH EBX	
0040297D	56	PUSH ESI	
0040297E	57	PUSH EDI	
0040297F	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00402982	33DB	XOR EBX,EBX	
00402984	895D FC	MOV DWORD PTR SS:[EBP-4],EBX	
00402987	6A 02	PUSH 2	
00402989	FF15 DC314000	CALL DWORD PTR DS:[4031DC]	msvcrt.__set_app_type
0040298F	59	POP ECX	00402AF0

You also remember false OEP Ours is 00402971, that is in this function on a tí. We will find real OEP. How to find what?

In Olly2. Distance from OEP to function "MSVCRT.__set_app_type" are:

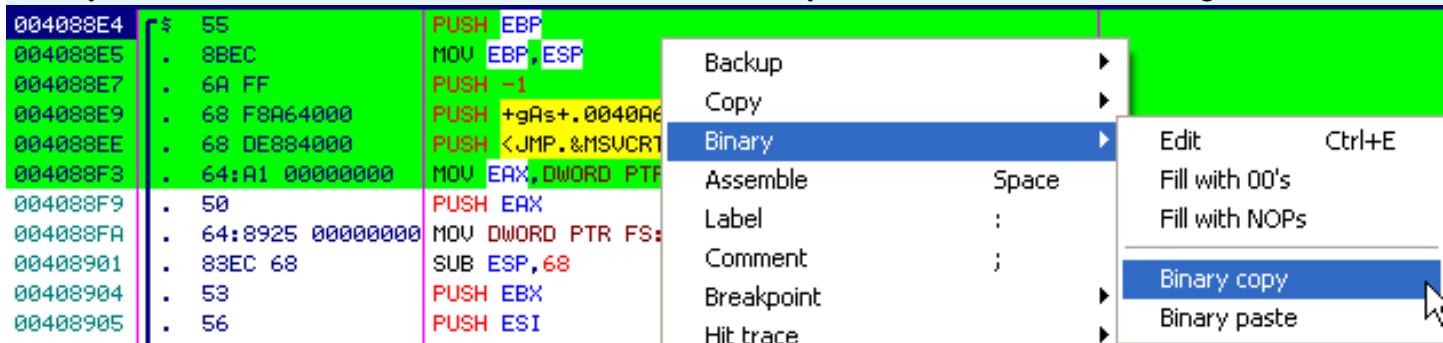
00408911 (set_app_type Ham) - 004088E4 (OEP) = 2D.

So in Olly1. Distance from OEP to function "MSVCRT.__set_app_type" must be 2D bytes. As against this, we are:

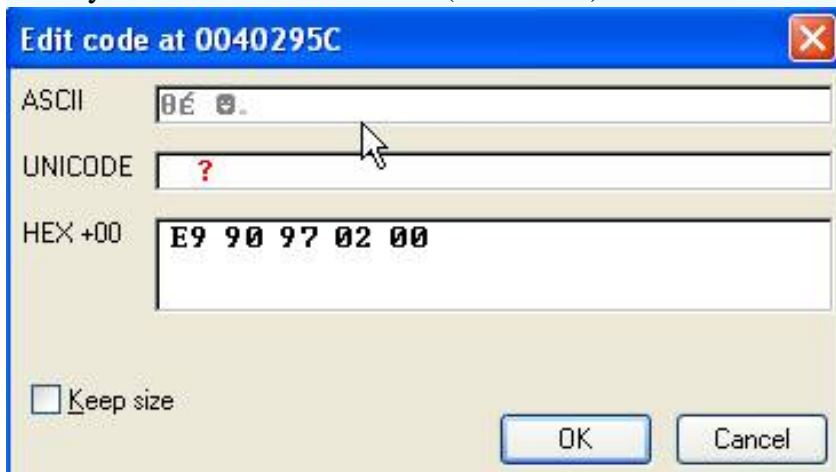
Real OEP = 00402989 (set_app_type Ham) - 2D = 0040295C

We have found real OEP. We need to fix Stolen bytes more. How fast is the copy of the bytes at OEP Olly2 to Olly1 then revised some value.

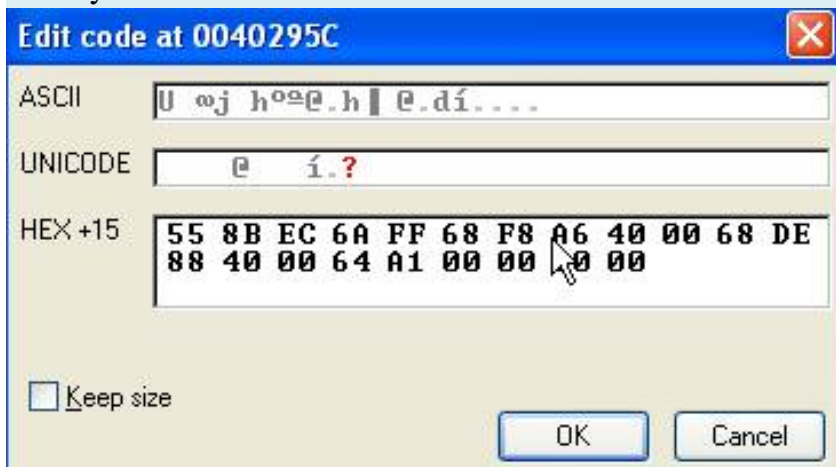
In Olly2 choose from OEP 004088E4 to 004088F3. Click your mouse to select the image:



In Olly1. Select line 0040295C (Real OEP). Press Ctrl-E.



Click your mouse to select Paste.



Click OK. We are as follows:

0040295C	55	PUSH EBP
0040295D	8BEC	MOV EBP,ESP
0040295F	6A FF	PUSH -1
00402961	68 F8A64000	PUSH 40A6F8
00402966	68 DE884000	PUSH 4088DE
0040296B	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
00402971	50	PUSH EAX
00402972	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
00402979	83EC 68	SUB ESP,68
0040297C	53	PUSH EBX
0040297D	56	PUSH ESI
0040297E	57	PUSH EDI

I think back down below:

PUSH EBP

MOV EBP, ESP

PUSH -1

PUSH 0040A6F8 (value should fix 1)

004088DE PUSH (value should fix 2)

MOV EAX, DWORD PTR FS: [0]

We need to fix 2 this value is more complete. In each of the MFC, 2 value is different. We must find the 2 values for this program by ourselves.

Very easy. You could look at the stack in Olly1.

0012FFB4	00402AF0	JMP to revert_except_handler3
0012FFB8	004037E8	
0012FFBC	FFFFFFFF	

The less you see? We have 2 value is 00402AF0 and 004037E8. So we will correct as follows (reverse order by the stack):

PUSH 0040A6F8 (value should fix 1)

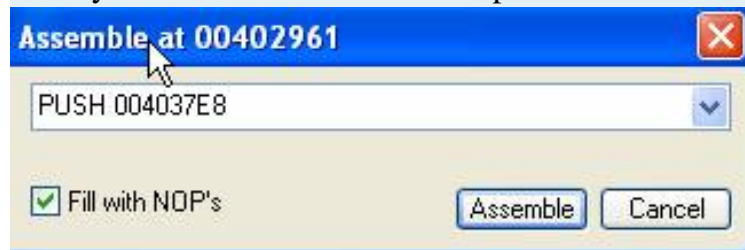
004088DE PUSH (value should fix 2)

Edit to:

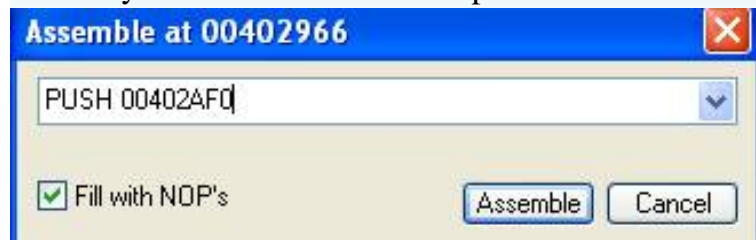
PUSH 004037E8 (value should fix 1)

PUSH 00402AF0 (value should fix 2)

In Olly1. Click on the 00402961. Space bar to hit. Edit the following:



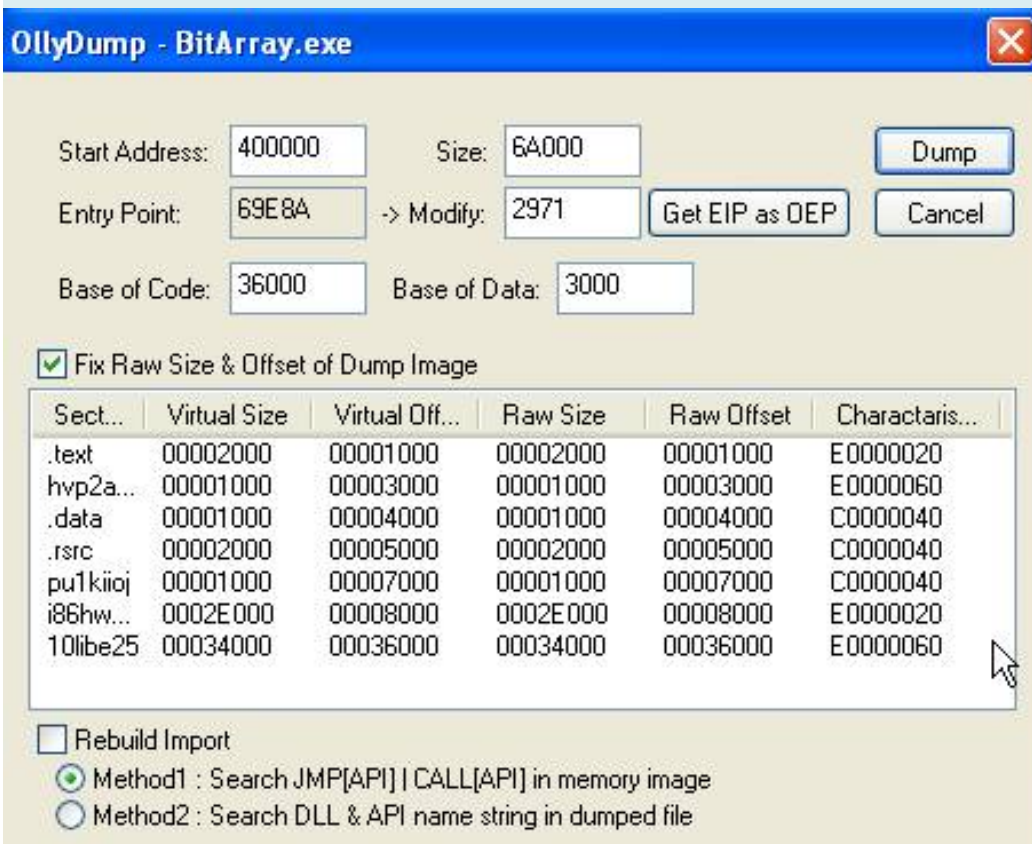
Similarly. Click line 00402966. Space bar to hit. Edit the following:



Final results we have:

0040295C	55	PUSH EBP
0040295D	8BEC	MOV EBP,ESP
0040295F	6A FF	PUSH -1
00402961	68 E8374000	PUSH 4037E8
00402966	68 F02A4000	PUSH 402AF0
0040296B	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
00402971	50	PUSH EAX
00402972	64:8925 00000000	MOV DWORD PTR FS:[0],ESP

Dump program. Using OllyDump.



Name the file is dump.exe. I am here you can Olly2 again.

IV) Fix IAT:

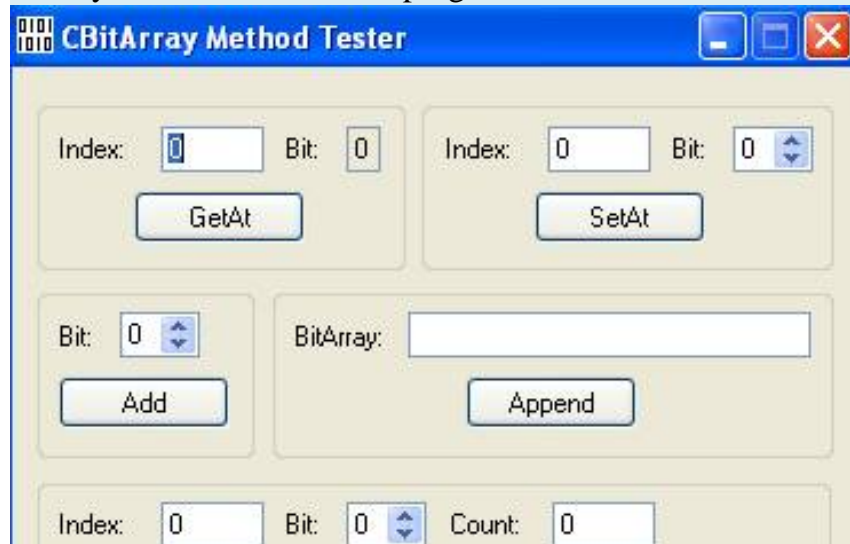
1) Method 1:

Using scripts "ExeCryptor 2.xx IAT Rebuilder v1.1.txt". The more you read the tut Why_Not_Bar to know how to use this script. You noted in a number of the script does not work (the program is run crash). If the crash when the fix was completed IAT not say anything but the crash between the IAT is the fix is lost húng! In the script we are good xài J

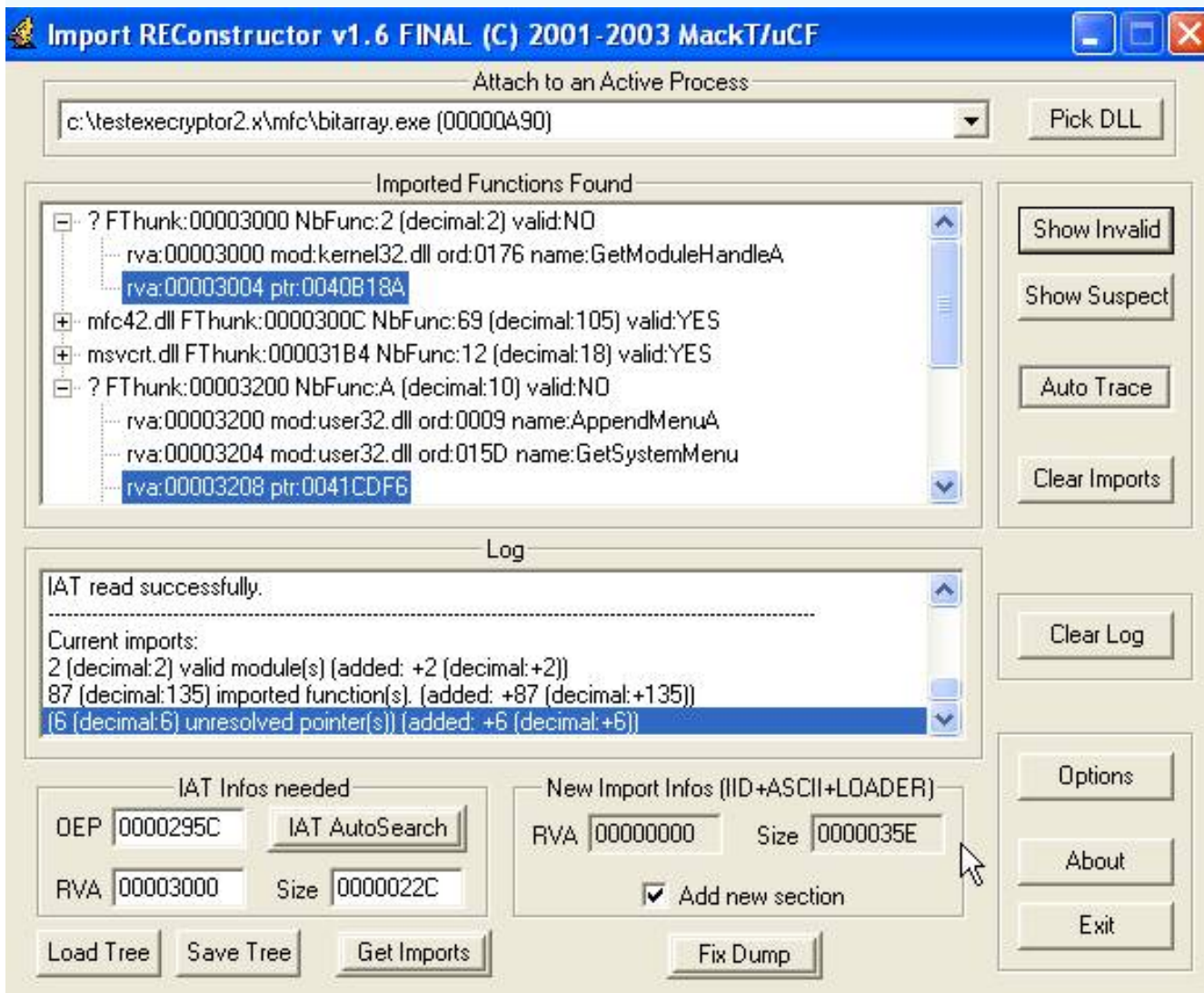
2) Method 2:

ExeCryptor have one or not we need to fix IAT full run new programs. IAT as long as we have a function GetModuleHandleA program stream. We will try to see.

In Olly1 click Shift-F9. The program we run:



Now run Imprec. Select BitArray process. Enter the number in, click the button IAT AutoSearch , OK , Get Import:



We see our IAT not complete. But what is not. Exemption can be a function GetModuleHandleA then. Click Fix dump. Dump.exe Select File. A notice of the J

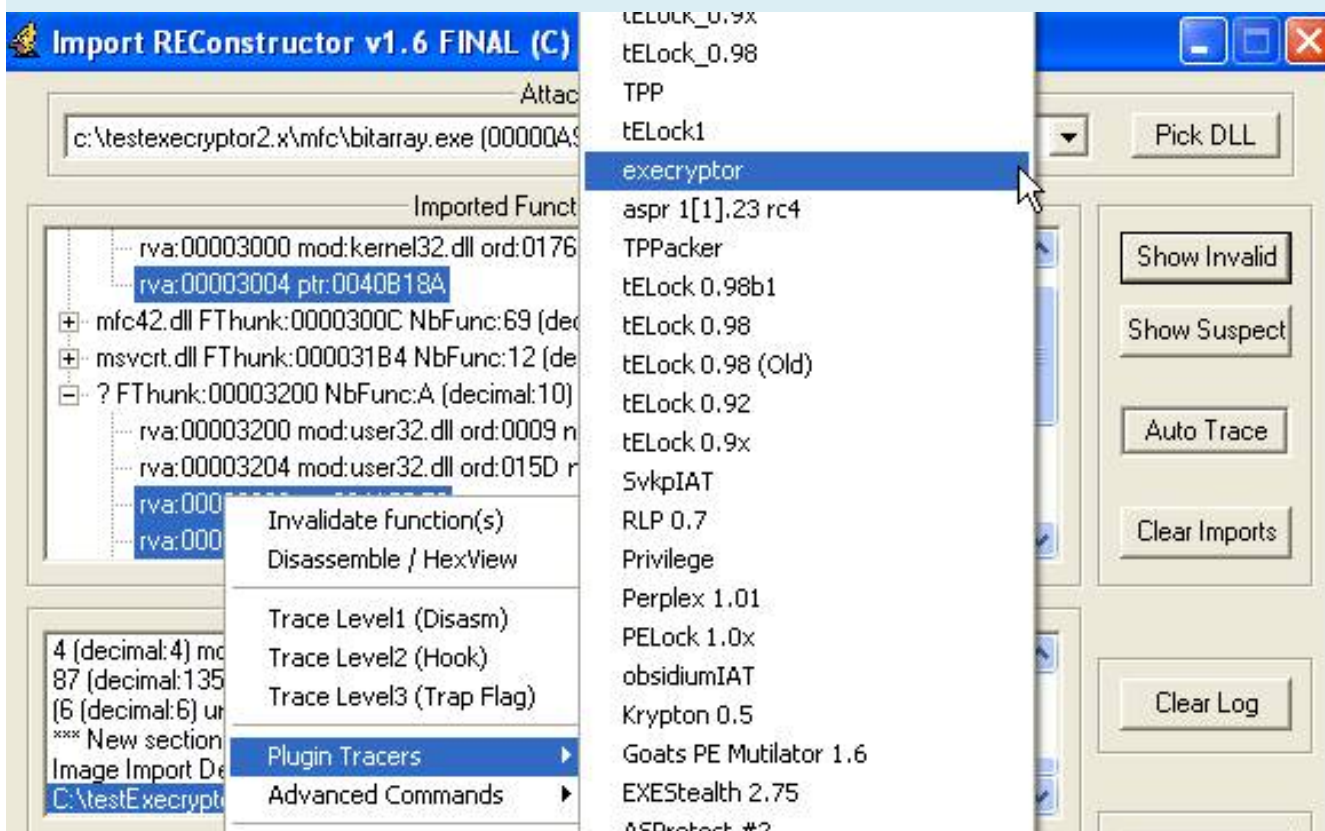


Do not. Click OK. We are dump_.exe file. Test this file. Good! Not surprised?

3) How 3:

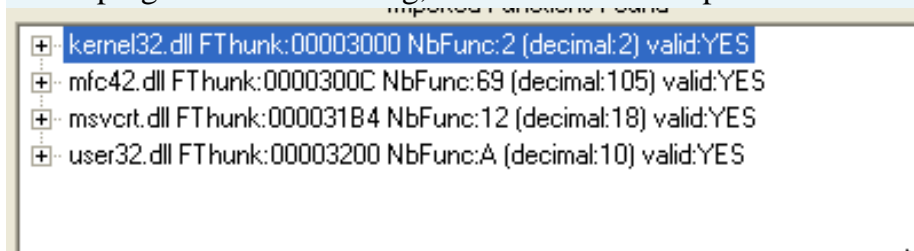
There is no need to fix IAT full run new programs. However, say what they say we still want a more complete IAT. Therefore we will use the plugin execryptor Imprec.

Hopefully you have not closed Imprec. Click "Show Invalid". Click your mouse to select the plugin execryptor as in the picture:



Wait a bit. Speaking before you run this plugin is quite long. You also do not close Olly1 nhé. I also take a while for new ways xài that this plugin;) catastrophic U.S.!

When plugin finished running, we will have a complete IAT.



The file you fix dump.exe normal. Done a test!

Them. 2h morning then. Make sure the company tomorrow to sleep too. Wishing you happy. Hopefully you will learn something new through this tut.

Greetingz: All reaonline.net members, ... and you.

tlandn

30-Aug-2006

Tutorials hacnho # 9

Manual unpacking UPX Protector 1.0x -> BlindAngel / TMG

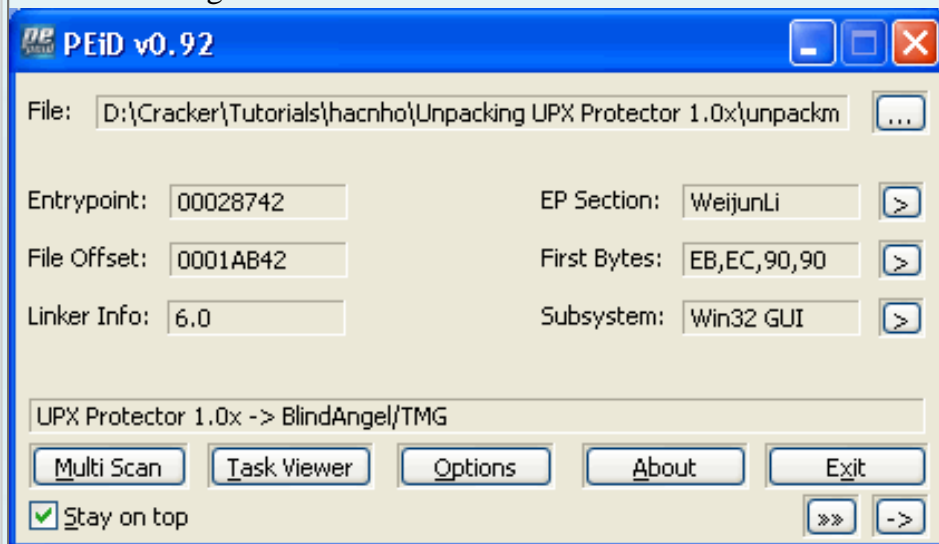
Information	Unpacking for Newbie's
Target	unpackme.exe
Available	http://nhandan.info/hacnho/tuts/unpackme9_tuts.zip
Tools	OllyDbg plugin 1.10c with OllyDump 2.21.108, ImportREC Final 1.6, 1.4 LordPE.
Protection	UPX Protector 1.0x -> BlindAngel / TMG
L Evel	Standard
Category	Manual unpacking

1. Introduction

Today, I will explain the ways for unpacking a packer of a member of TMG team. This is a packer enjoyable.

2. Getting Started

Use PEiD and get some LordPE for PE Info.



[PE Editor] - d:\cracker\tutorials\hacnh\unpacking upx protector 1.0x\unpack...

Basic PE Header Information

EntryPoint:	00028742	Subsystem:	0002	...
ImageBase:	00400000	NumberOfSections:	0003	
SizeOfImage:	0002A000	TimeDateStamp:	391FB26F	
BaseOfCode:	0000E000	SizeOfHeaders:	00000400	? +
BaseOfData:	00029000	Characteristics:	010F	...
SectionAlignment:	00001000	Checksum:	00000000	?
FileAlignment:	00000200	SizeOfOptionalHeader:	00E0	
Magic:	010B	NumOfRvaAndSizes:	00000010	+ -

OK

Save

Sections

Directories

FLC

TDSC

Compare

L

[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
.PCPEC	00001000	0000D000	00000400	00000000	E0000080
WeijunLi	0000E000	0001B000	00000400	0001A400	E0000040
.Stone	00029000	00001000	0001AE00	00000200	C0000040

EP: 28742, flags The value of this case is not needed, Image Base is always 400000, Import Table: 0000 and size is 00.

3. Finding the OEP

Load **unpackme.exe** into OllyDBG. And you still here:

00428742	EB EC	JMP SHORT unpackme.00428730
00428744	90	NOP
00428745	90	NOP
00428746	90	NOP
00428747	90	NOP
00428748	8A06	MOV AL,BYTE PTR DS:[ESI]
0042874A	46	INC ESI
0042874B	8B07	MOV BYTE PTR DS:[EDI],AL
0042874D	47	INC EDI
0042874E	01DB	ADD EBX,EBX

Then, you press **F7** one time and you see as follows:

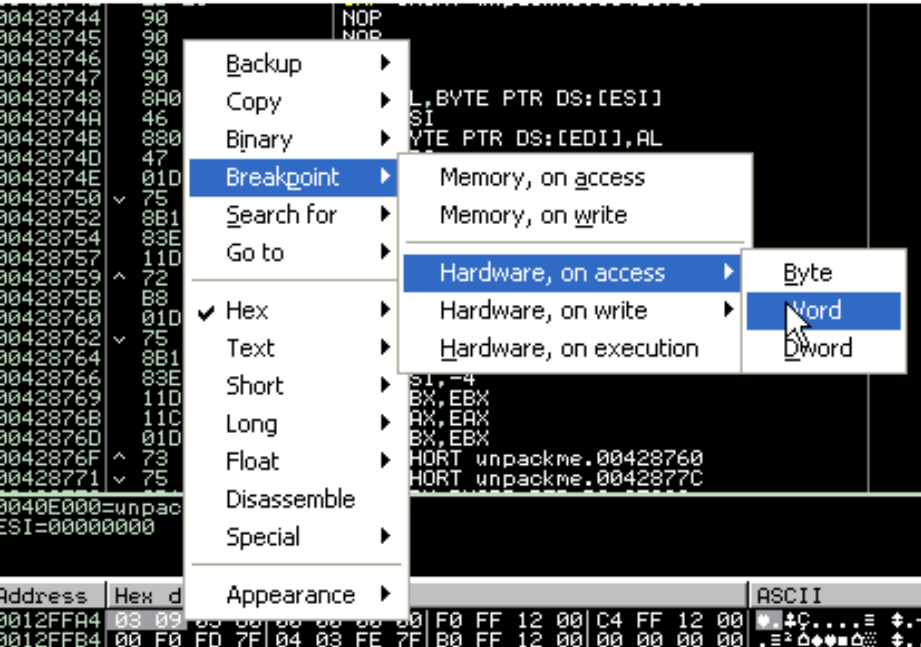
00428730	60	PUSHAD
00428731	BE 00E04000	MOV ESI,unpackme.0040E000
00428736	8DBE 0030FFFF	LEA EDI,DWORD PTR DS:[ESI+FFFF3000]
0042873C	57	PUSH EDI
0042873D	83CD FF	OR EBP,FFFFFFFF
00428740	EB 10	JMP SHORT unpackme.00428752
00428742	EB EC	JMP SHORT unpackme.00428730

Then, continue press **F7**

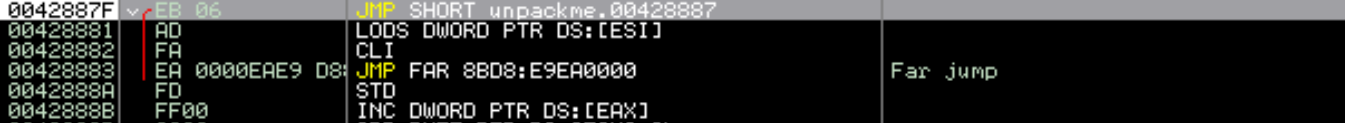
00428730	60	PUSHAD
00428731	BE 00E04000	MOV ESI,unpackme.0040E000
00428736	8DBE 0030FFFF	LEA EDI,DWORD PTR DS:[ESI+FFFF3000]
0042873C	57	PUSH EDI
0042873D	83CD FF	OR EBP,FFFFFFFF
00428740	EB 10	JMP SHORT unpackme.00428752
00428742	EB EC	JMP SHORT unpackme.00428730

At this line, you see in the **Registers (FPU)** table. The value of **ESP** is **0012FFA4**, then you right click on and choose **ESP Follow in the dump**.

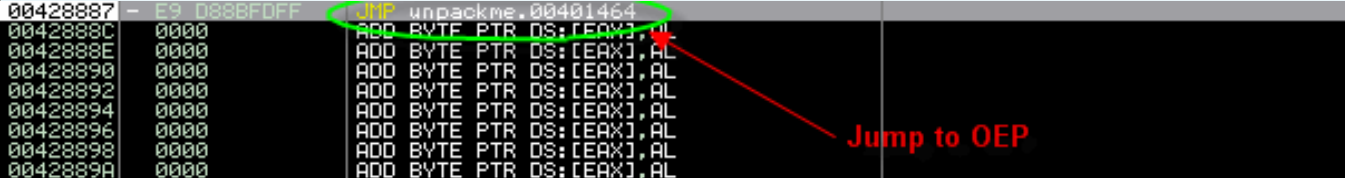
Then you go to the Hex dump window. Then right click on the value **0012FFA4** and select Breakpoint -> Hardware, on Access -> Word. Our breakpoint is now set.



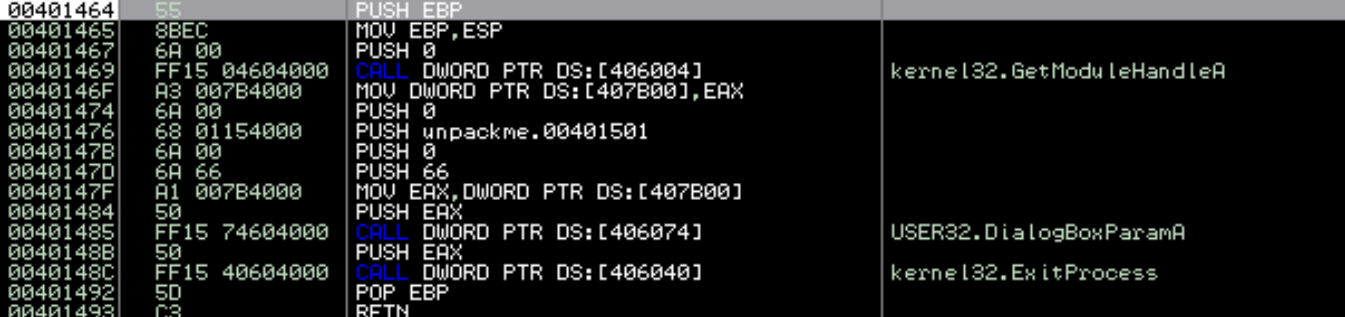
Continued, press **F9**. And you still here:



And then press **F8**:



Next, press **F7**. And we see as follows:



Congratulations! According to OEP we found is **401,464**. And now we Calculate the real OEP of this unpackme by the formula:

Real OEP = OEP find in Olly-Image Base = 401464-400000 = **1464**.

4th dumping

At address **0041464**, we go to the menu Plugin -> OllyDump -> dump debugged process. And then, just press dump, save the file at **dumped.exe** unpacked.

OllyDump - unpackme.exe

Start Address: 400000 Size: 2A000 Dump

Entry Point: 28742 -> Modify: 1464 Get EIP as OEP Cancel

Base of Code: E000 Base of Data: 29000

Our OEP!

☒ Fix Raw Size & Offset of Dump Image

Sect...	Virtual Size	Virtual Off...	Raw Size	Raw Offset	Characteris...
.PCPEC	0000D000	00001000	0000D000	00001000	E0000080
WeijunLi	0001B000	0000E000	0001B000	0000E000	E0000040
.Stone	00001000	00029000	00001000	00029000	C0000040

☐ Rebuild Imports **Don't check this options!**

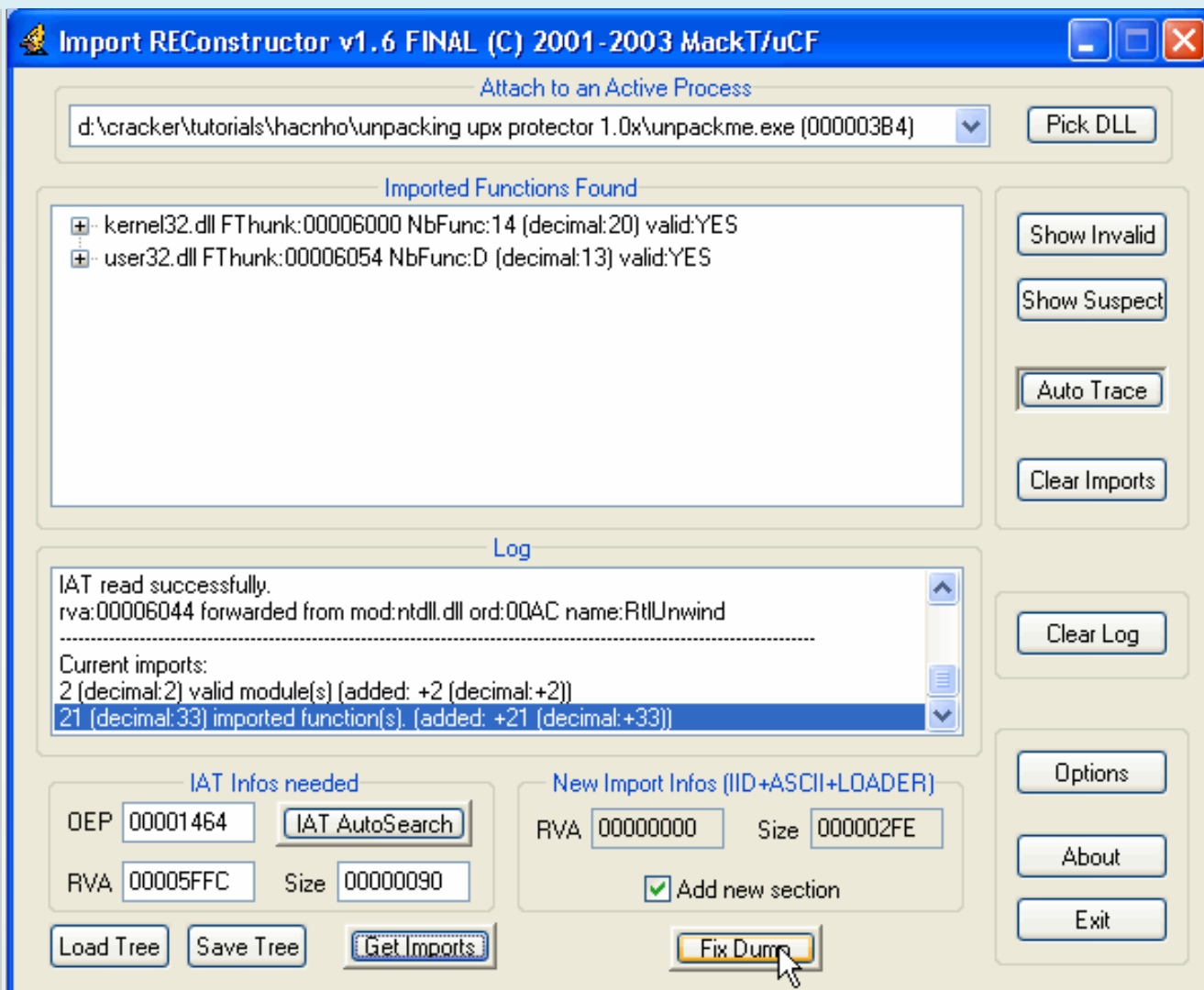
☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Do not run **dumped.exe** now, will be a crash ... It must fix IAT.

5. Finding and Fixing the Address Import Table

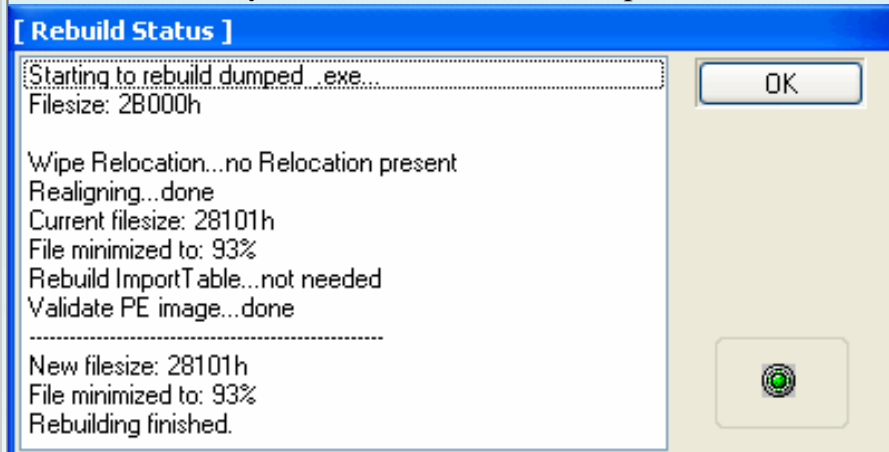
And select Open ImpREC attached to active process and choose unpackme.exe. Change the value in the OEP window to the one we wrote down earlier (1464) then select IATAutosearch then click Get Imports.



All Import Functions valid.

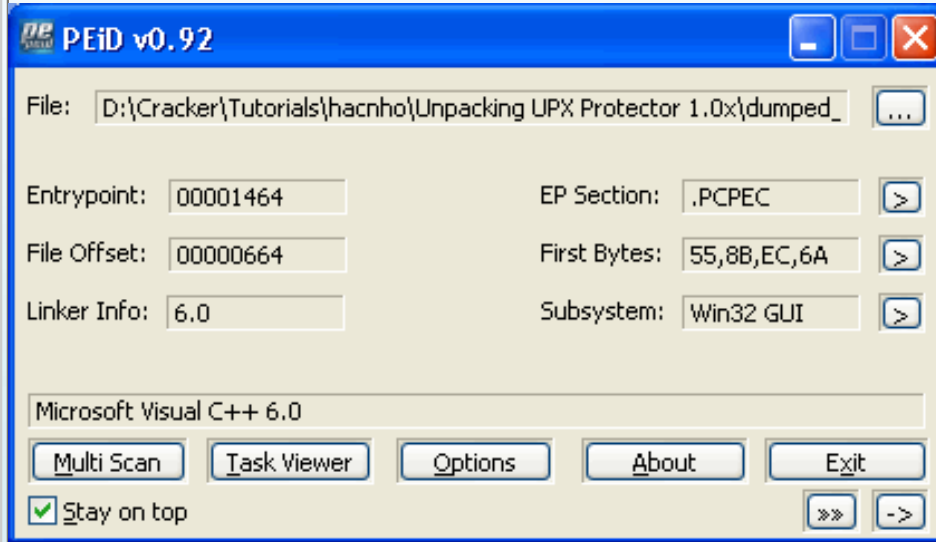
Now, click fix dump to fix the IAT **dumped.exe** file.

Use LordPE 1.4 by Y0da for rebuild our Dumped_.exe



6. Testing Our Unpacked file

Use PEiD for detect again:



Now run unpacked files. Wow, not crash.

7. Conclusion

Special thanx to **R @ dier** for this template.

My Greetz to: Deux, infinite, NVH (c), softcracker_vn, luucorp, Aaron, hhphong, R @ dier, tlandn, Computer_Angel, k3nny, Ferrari, Zombie, RCA, CTL, Moonbaby, Neitsa, JAL, LeVuHoang, 777, LeonHart , Bin ... and you ;-)!

To be continued ...

Written by [hacnho](#) (tutorial date: Sai Gon 3/5/2004)

hacnho Tutorials # 13

Manual unpacking Virogen crypt v0.75

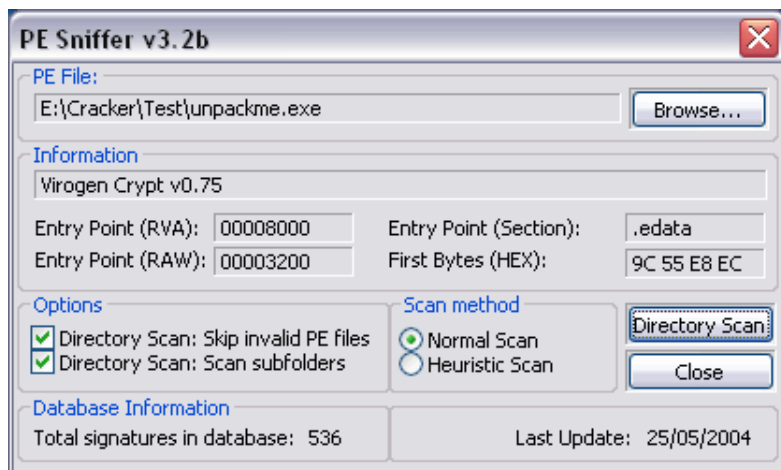
Information	Unpacking for Newbie's
Target	Unpackme.exe
Available	http://nhandan.info/hacnho/tuts/unpackme13_tuts.zip
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, Lord PE 1.4, PESniffer 3.2b.
Protection	Virogen crypt v0.75
Level	Easy
Category	Manual unpacking

1.Introduction

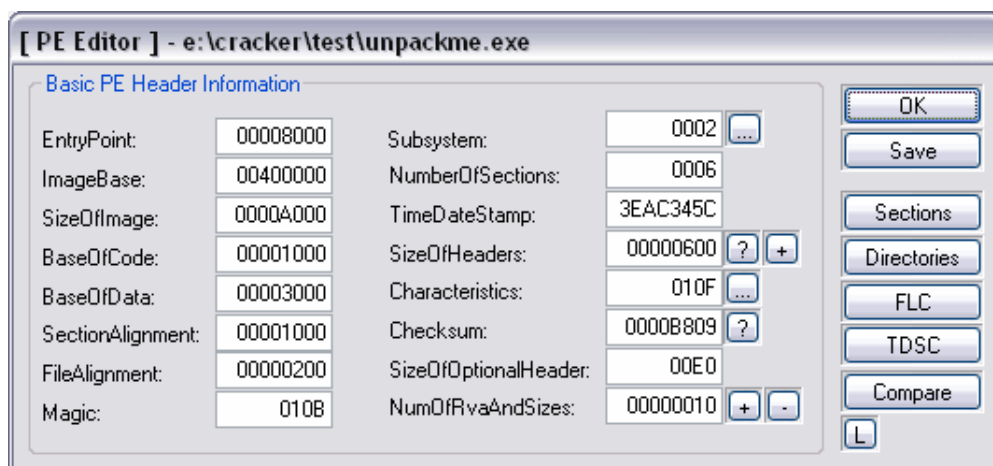
Team EAT crypter use this pack for their keygens. Today, we will try to unpack this crypter, so easy! I was replace the template's Radier by this template! What do you think about this!

PE 2.Detect and get info

First, use PE for Sniffer detect.



And get LordPE for PE Info.



[Section Table]						
Name	VOffset	VSize	ROffset	RSize	Flags	
.text	00001000	00002000	00000600	00000A00	C0000040	
.rdata	00003000	00001000	00001000	00000600	C0000040	
.data	00004000	00001000	00001600	00000200	C0000040	
.rsrc	00005000	00001000	00001800	00000400	C0000040	
.eatkey	00006000	00002000	00001C00	00001600	C0000040	
.edata	00008000	00001158	00003200	00000200	E0000060	

Then, we have the Entry Point is: 8000, Image Base is 40,000, is 00 and Importable Size is 00!

3.Finding the Original Entry Point

Now, after PE detect and get info, load this into unpackme Olly!

00408000	9C	PUSHFD	
00408001	55	PUSH EBP	
00408002	E8 EC000000	CALL unpackme.004080F3	
00408007	87D5	XCHG EBP,EDX	
00408009	5D	POP EBP	
0040800A	60	PUSHAD	
0040800B	87D5	XCHG EBP,EDX	
0040800D	80B0 15274000	CMPS BYTE PTR SS:[EBP+402715],1	
00408014	74 39	JE SHORT unpackme.0040804F	
00408016	C685 15274000	MOV BYTE PTR SS:[EBP+402715],1	
0040801D	E9 E4000000	JMP unpackme.00408106	
00408022	E9 BEED4BC1	JMP C18CDE5	
00408027	9B	WAIT	

Press **F7** to trace address 408,001.

00408000	9C	PUSHFD	
00408001	55	PUSH EBP	
00408002	E8 EC000000	CALL unpackme.004080F3	
00408007	87D5	XCHG EBP,EDX	
00408009	5D	POP EBP	
0040800A	60	PUSHAD	
0040800B	87D5	XCHG EBP,EDX	

At this line, you see in the Registers (FPU) table. The value of ESP is **0012FFC0**, then you right click on and choose the **ESP Follow in dump**

Registers (FPU)		
EAX	00000000	
ECX	0012FFB0	
EDX	7FFEF0304	
EBX	7FFDF000	
ESP	0012FFC0	
	0012FFC0	
	00000000	
	EE0508C1	
	00408001	unpackme
	ES 0023	32bit 00
	CS 001B	32bit 00
	SS 0023	32bit 00
	DS 0023	32bit 00
	FS 0038	32bit 7F
	GS 0000	NULL
LastErr ERROR_S		
00000286 (NO,NB,n		
	empty +UNORM 20A0	
	empty +UNORM 2400	
	empty +UNORM 17C0	
	empty 0 00000000	

Then you go to the Hex dump window. Then right click on the value **0012FFC0** and select Breakpoint -> Hardware, on Access -> Word. Our breakpoint is now set. Then Press **F9** to run unpackme, after that, continue press **F8** 5 times until you see as follows:

004063B0	75 08	JNZ SHORT unpackme.004063BA	
004063B2	B8 01000000	MOV EAX,1	
004063B7	C2 0C00	RETN 0C	
004063BA	68 FE1D4000	PUSH unpackme.00401DFE	
004063BF	C3	RETN	
004063C0	8B85 26040000	MOV EAX,DWORD PTR SS:[EBP+426]	
004063C6	8D8D 3B040000	LEA ECX,DWORD PTR SS:[EBP+43B]	
004063CC	51	PUSH ECX	
004063CD	50	PUSH EAX	
004063CE	FF0F 40050000	PUSH DWORD PTR SS:[EBP+510]	

Et puis, press **F7** 3 times, and you still here:

00401DFE	55	DB 55	CHAR 'U'
00401DFF	8B	DB 8B	
00401E00	EC	DB EC	
00401E01	6A	DB 6A	CHAR 'j'
00401E02	FF	DB FF	
00401E03	68	DB 68	CHAR 'h'
00401E04	78	DB 78	CHAR 'x'
00401E05	34	DB 34	CHAR '4'
00401E06	40	DB 40	CHAR 'e'
00401E07	00	DB 00	
00401E08	68	DB 68	CHAR 'h'
00401E09	84	DB 84	

Now! Press **Ctrl + A** for analyze:

00401DFE	55	PUSH EBP	
00401DFF	8BEC	MOV EBP,ESP	
00401E01	6A FF	PUSH -1	
00401E03	68 78344000	PUSH unpackme.00403478	
00401E08	68 841F4000	PUSH unpackme.00401F84	
00401E0D	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	JMP to MSUCRT._except_handler3
00401E13	50	PUSH EAX	
00401E14	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00401E1B	83EC 68	SUB ESP,68	
00401E1E	53	PUSH EBX	
00401E1F	56	PUSH ESI	

We are on OEP. The OEP is **401DFE**. Now we must Calculate the real OEP by the formula: Real OEP = OEP find in Olly-Image Base = 401DFE - 400,000 = **1DFE**

4th dumping our Unpacked file

At **401DFE** address, we go to the menu Plugin -> OllyDump -> dump debugged process. And then, just press dump, save the file unpacked at **dumped.exe**

OllyDump - unpackme.exe

Start Address: 400000 Size: 9158 **Dump**

Entry Point: 8000 -> Modify: 1DFE **Get EIP as OEP** **Cancel**

Base of Code: 1000 Base of Data: 3000

☒ **Fix Raw Size & Offset of Dump Image**

Sect...	Virtual Size	Virtual Off...	Raw Size	Raw Offset	Characteris...
.text	00002000	00001000	00002000	00001000	C0000040
.rdata	00001000	00003000	00001000	00003000	C0000040
.data	00001000	00004000	00001000	00004000	C0000040
.rsrc	00001000	00005000	00001000	00005000	C0000040
.eatkey	00002000	00006000	00002000	00006000	C0000040
.edata	00001158	00008000	00001158	00008000	E0000060

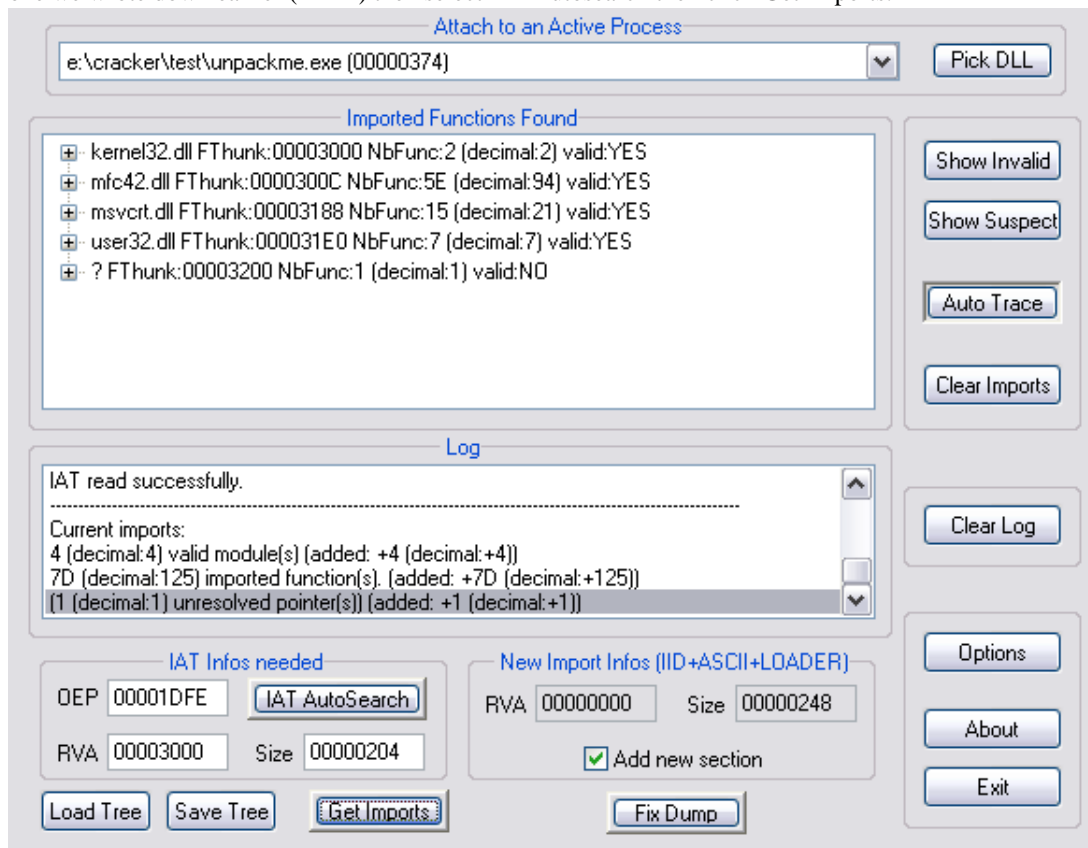
☐ **Rebuild Import**

☒ Method1 : Search JMP[API] | CALL[API] in memory image

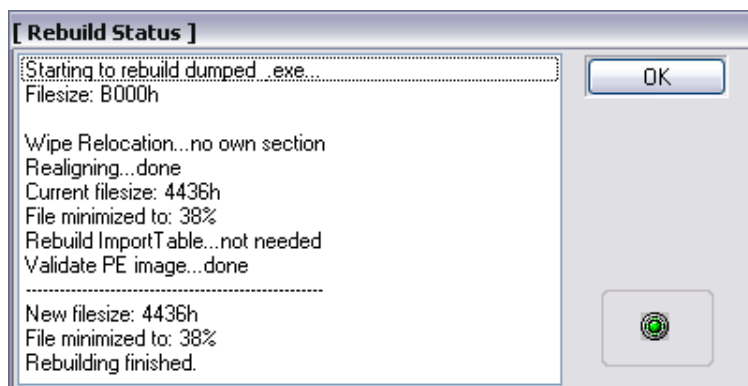
☐ Method2 : Search DLL & API name string in dumped file

5th Finding and Fixing the Address Import Table

And select Open ImpREC attached to active process and choose unpackme.exe. Change the value in the OEP window to the one we wrote down earlier (1DFE) then select IATAutosearch then click Get Imports.



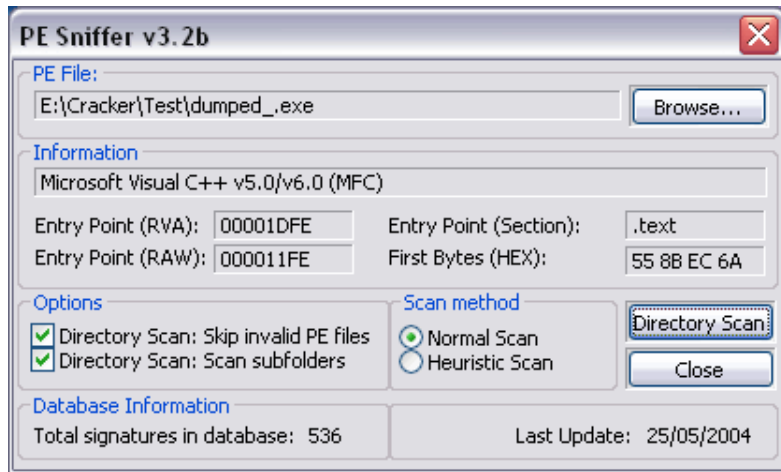
Humm, the unlucky import function is invalid. But no problem! Now, you must click on **Show Invalid** button. Then right click on the invalid imports and choose: Trace level1 (Disasm). Click Show Invalid again. Continue, right click invalid import, choose **Delete Thunk (s)**. Now, click fix dump to fix the IAT **dumped.exe** file. Use LordPE 1.4 by Y0da for rebuild our Dumped_.exe



6. Testing Our Unpacked file

Now run unpacked files. It's Okay!

Using **PE Sniffer** detect **3.2b** for: **Microsoft Visual C++ v5.0/v6.0 (MFC)**. Okie, Virogen crypt v0.75 is now unpacked successful!



7. Conclusion

My Greetz to: Deux, infinite, NVH (c), softcracker_vn, luucorp, Aaron, hhphong, R @ dier, tlandn, Computer_Angel, Zombie, RCA, CTL, Moonbaby, Ferrari, Devilz, Neitsa, anh_surprised ... and you; -)!

To be continued ...

Written by **hacnho** (tutorial date: Saigon 30/05/2004)

[\[Exetools Forum\]](#) | [\[HVAOnline\]](#) | [\[Vncracking Group\]](#) | [\[REA Forum\]](#) | [\[hacnho's homepage\]](#) | [\[AR Team\]](#) | [\[Vicki's Fan\]](#) | [\[VCT2k4\]](#) |

...: Copyright © 2004 by hacnho <[==]> VCT-Vietnamese Cracking Team 2k4:: ..

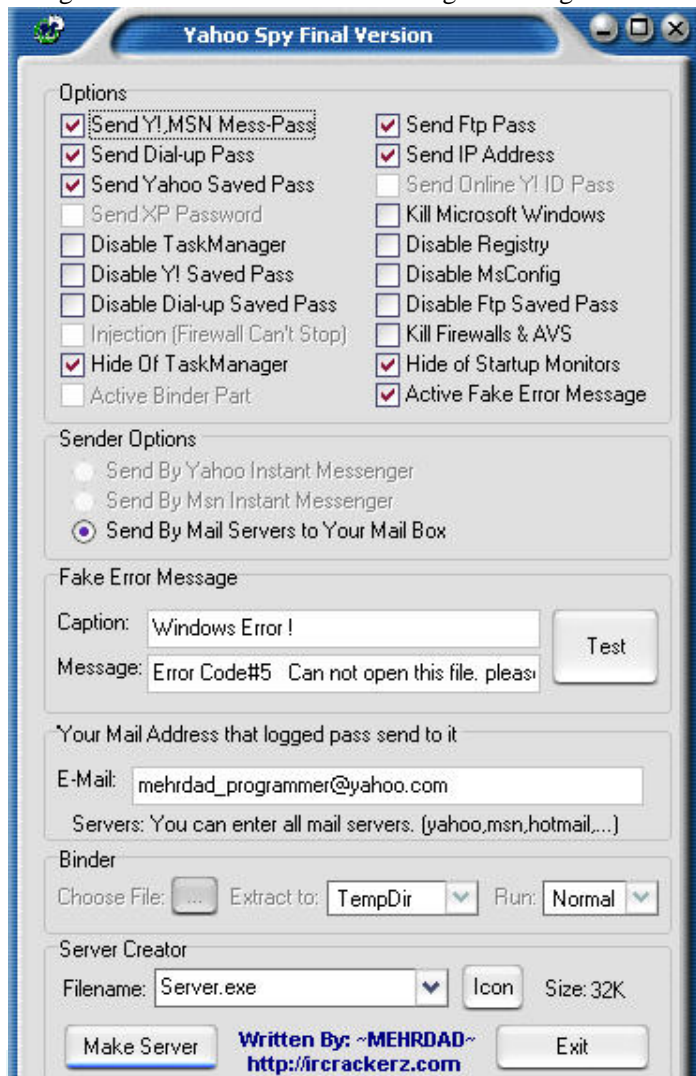
hacnho Tutorials # 16

Manual unpacking WWPack32 1.x

Information	Unpacking for Newbie's
Target	target
Available	http://nhandan.info/hacnho/tuts/unpackme16_tuts.zip
Tools	1:10 OllyDbg plugin with Final OllyDump 2.21.108, OllyScript, Lord PE 1.4, PEiD 0.92, EM Editor ProtectionID_v5.0_Final and write scripts for 4:04.
Protection	WWPack32 1.x
Level	Easy
Category	Manual unpacking

1.Introduction

The good tools for Yahoo Pass Hacking after Magic-PS 1.x is Yahoo Spy.





This is packed with tools WWPack32 1.x. PEiD and PESniffer PEScan or can not detect him. So, I am trying to detect with ProtectionID_v5.0_Final.



This packer is easy for unpack. Follow my tut

2.Find OEP

Step 1: Find OEP

Load-1 into target OllyDBG

Address	Hex dump	Disassembly	Comment
004AD5BC	EB E2	JMP SHORT target.004AD5A0	
004AD5BE	90	NOP	
004AD5BF	90	NOP	
004AD5C0	8A06	MOV AL, BYTE PTR DS:[ESI]	
004AD5C2	46	INC ESI	
004AD5C3	8B07	MOV BYTE PTR DS:[EDI], AL	
004AD5C5	47	INC EDI	
004AD5C6	010B	ADD EBX, EBX	
004AD5C8	75 07	JNZ SHORT target.004AD5D1	
004AD5CA	8B1E	MOV EBX, DWORD PTR DS:[ESI]	
004AD5CC	83EE FC	SUB ESI, -4	
004AD5CE	410B	ADD ECX, ECX	

Press **F8** 2 times (you can see the ESP register in FPU is highlighting Windows):

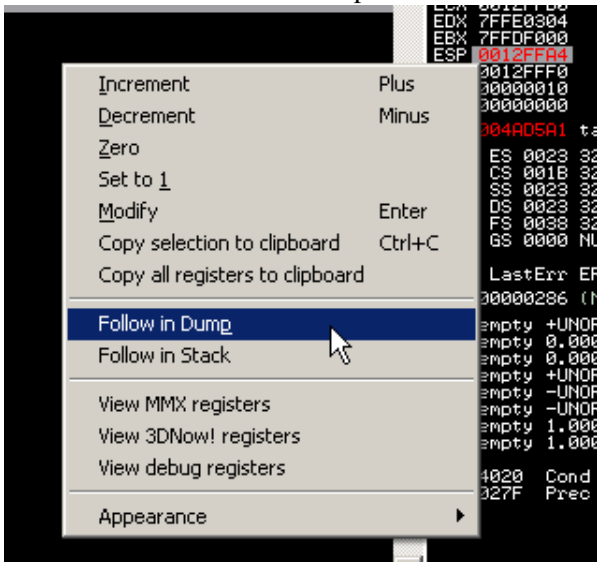
Address	Hex dump	Disassembly	Comment
004AD5A0	60	PUSHAD	
004AD5A1	BE 00404600	MOV ESI, target.00464000	
004AD5A6	8DBE 0000F9FF	LEA EDI, DWORD PTR DS:[ESI+FFF9D000]	
004AD5AC	C787 A0900700	MOV DWORD PTR DS:[EDI+790A0], 4C3D1	
004AD5B6	57	PUSH EDI	
004AD5B7	83CD FF	OR EBP, FFFFFFFF	
004AD5BA	EB 0E	JMP SHORT target.004AD5CA	
004AD5BC	EB E2	JMP SHORT target.004AD5A0	
004AD5BE	90	NOP	
004AD5BF	90	NOP	
004AD5C0	8A06	MOV AL, BYTE PTR DS:[ESI]	
004AD5C2	46	INC ESI	

```

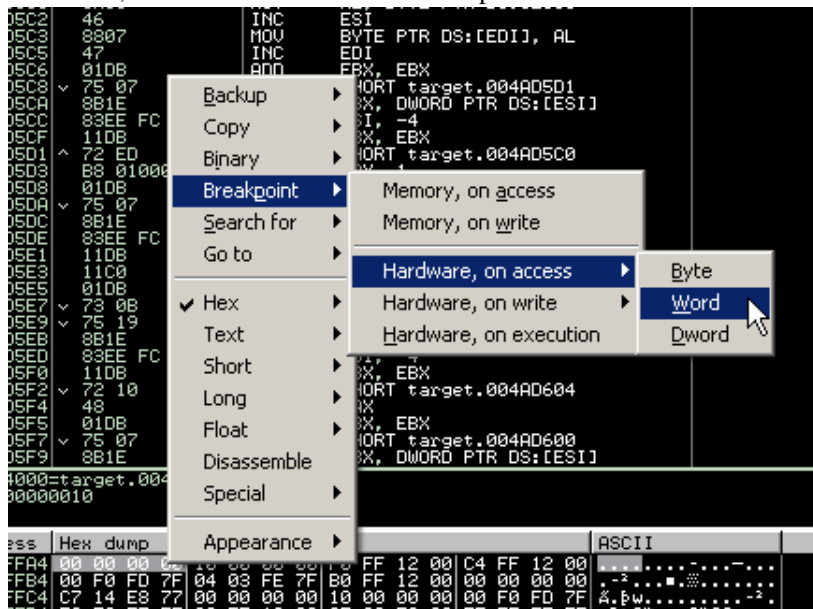
Registers (FPU)
EAX 00000000
ECX 0012FFB0
EDX 7FFE0304
EBX 7FFDF000
ESP 0012FFA4
EBP 0012FFF0
ESI 00560718
EDI 77F5164E ntdll.77F5
EIP 004AD5A1 target.004

```

At this line, you see in the Registers (FPU) table. The value of ESP is **0012FFA4**, then you right click on and choose the **ESP Follow in dump**



Then you go to the Hex dump window. Then right click on the value 0012FFA4 and select Breakpoint -> Hardware, on Access -> Word. Our breakpoint is now set.



Press **F9** to run. You will still be here:

Address	Hex dump	Disassembly	Comment
004AD70B	ES: 11	JMP SHORT target.004AD71E	
004AD70D	AD	LODS DWORD PTR DS:[ESI]	
004AD70E	FA	CLI	
004AD70F	EA 28D74A00 38	JMP FAR D738:004AD728	Far jump
004AD716	4A	DEC EDX	
004AD717	00A0 A0470000	ADD BYTE PTR DS:[EAX+47A0], AH	
004AD71D	EA E98DC1FC FF	JMP FAR 00FF:FCC18DE9	Far jump
004AD724	0000	ADD BYTE PTR DS:[EAX], AL	
004AD726	0000	ADD BYTE PTR DS:[EAX], AL	
004AD728	0000	ADD BYTE PTR DS:[EAX], AL	
004AD72A	0000	ADD BYTE PTR DS:[EAX], AL	

Press **F8** 2 times. This is OEP.

Address	Hex dump	Disassembly	Comment
004798B0	55	PUSH EBP	
004798B1	8BEC	MOV EBP, ESP	
004798B3	83C4 F0	ADD ESP, -10	
004798B6	B8 38964700	MOV EAX, target.00479638	
004798B8	E8 44C5F8FF	CALL target.00405E04	
004798C0	A1 D0B34700	MOV EAX, DWORD PTR DS:[47B3D0]	
004798C5	8B00	MOV EAX, DWORD PTR DS:[EAX]	
004798C7	E8 2405FEFF	CALL target.00459DF0	
004798CC	8B00 BCB14700	MOV ECX, DWORD PTR DS:[47B1BC]	target.0047CC88
004798D2	A1 D0B34700	MOV EAX, DWORD PTR DS:[47B3D0]	
004798D7	8B00	MOV EAX, DWORD PTR DS:[EAX]	
004798D9	8B15 B0804700	MOV EDI, DWORD PTR DS:[4780B0]	target.004780FC
004798DF	E8 2405FEFF	CALL target.00459E08	
004798E4	A1 D0B34700	MOV EAX, DWORD PTR DS:[47B3D0]	
004798E9	8B00	MOV EAX, DWORD PTR DS:[EAX]	
004798EB	E8 9805FEFF	CALL target.00459E88	
004798F0	E8 EBA5F8FF	CALL target.00403EE0	
004798F5	8D40 00	LEA EAX, DWORD PTR DS:[EAX]	

Step 2: dumping

- Go to the menu plugin, choose OllyDUMP:

OllyDump - target.exe

Start Address: 400000 Size: B1000 **Dump**

Entry Point: AD5BC -> Modify: 798B0 **Get EIP as OEP** **Cancel**

Base of Code: 64000 Base of Data: AE000

☒ **Fix Raw Size & Offset of Dump Image**

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
PREVI...	00063000	00001000	00063000	00001000	E0000080
.wwP...	0004A000	00064000	0004A000	00064000	E0000040
ANAKI...	00003000	000AE000	00003000	000AE000	C0000040

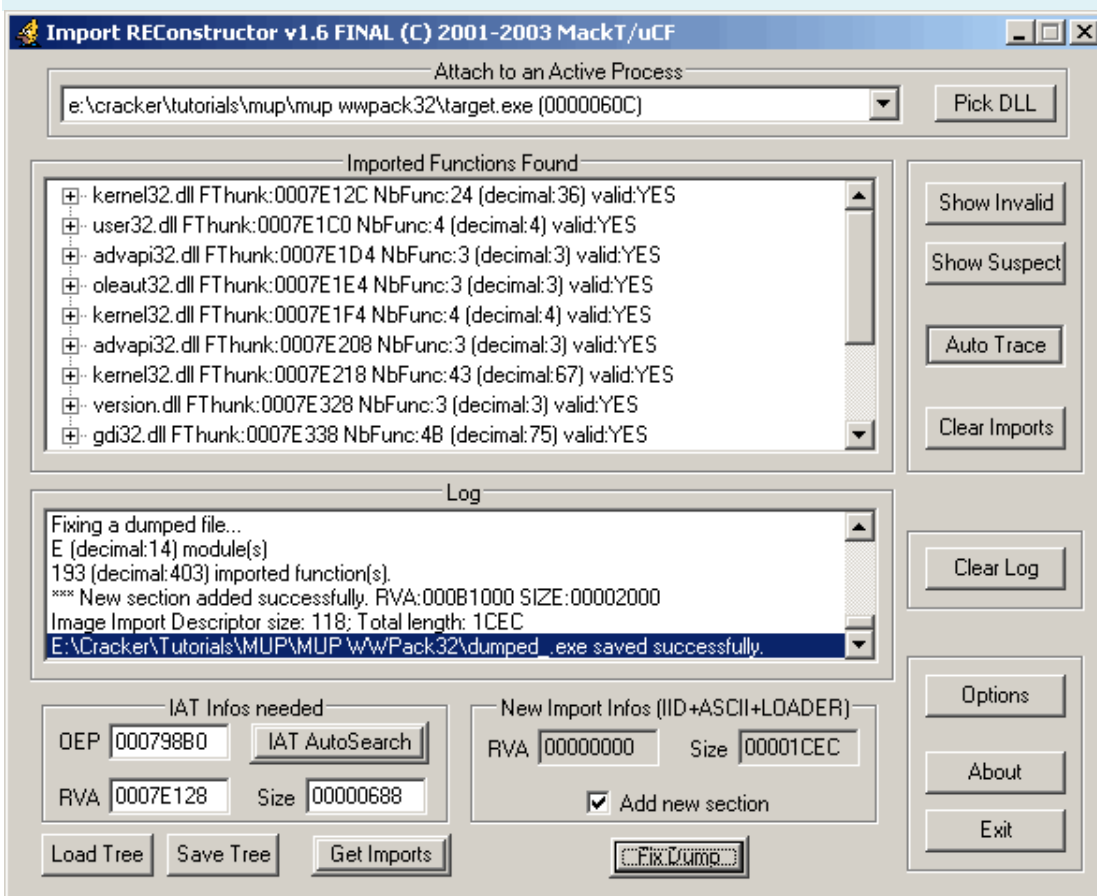
☐ **Rebuild Import**

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Step 3: Finding and Fixing the Adress Import Table

And select Open ImpREC attached to active process and choose unpackme.exe. Change the value in the OEP window to the one we wrote down earlier (798B0) then select IATAutosearch then click Get Imports.



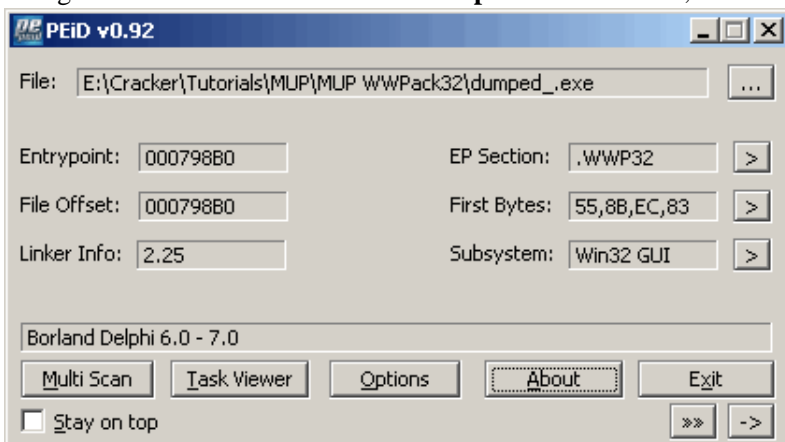
Now, click fix to fix IAT dump the file **dumped.exe**

- Unpacked successful! Done ...

3. Testing Our Unpacked file

Now run the unpacked files. It's Okay!

Using **0.92** for detect **PEiD: Borland Delphi 6.0 - 7.0**. Okie, now WWPack32 v1.xis unpacked successful!



4. Create OllyScript

After find OEP in Olly, we create a need for auto OllyScript find OEP next time! Remember! We have found three step for OEP:

1st First: Find the special signal of WWPack32 for breakpoint set!

2.Second: Step over 2 times.

3.Final: Set breakpoint, press F9 to run, press F8 2 times for jump to **OEP**

Okay!, Write in our step langue OllyScript

Cut here -----

/ *

////////////////////////////////////

// WWPack32 v1.x OEP finder

// Author: hacnho/VCT2k4

// Email: hacnho@hotmail.com

// Website: http://nhandan.info/hacnho

// OS: WinXP Pro, OllyDbg 1:10 Final, OllyScript v0.85

////////////////////////////////////

* /

STI // Step into (F7)

Sto // Step over (F8)

eob Break

findop eip, 60BE # # // Find the special signal

bphws ESP, "r" // Set a breakpoint on memory access

run // Run the program

Bread:

Sto

Sto

Police eip // Ctrl + A for Analyze

log eip // Logs to source OllyDbg log window.

CMT eip, "This is the OEP! Found by hacnho/VCT2k4" // Write a comment

Msg "Dumped and IAT fix now! Thanx for using my script ...!" // Show a message

ret // Exits script

Cut here -----

6. Conclusion

GrEeTs Fly Out: Deux, infinite, NVH (c), softcracker_vn, luucorp, Aaron, Canterwood, hhphong, R @ dier, tlandn, Computer_Angel, Zombie, RCA, CTL, Moonbaby, Nilrem, diablo2oo2, Ferrari, Devilz, anh_surprised .. . and you ;-)!

Thanx to authors of OllyDBG, ImpREC, LordPE, OllyScript, PEiD, ID Protection, WWPack32,

To be continued ...

Written by [hacnho](#) (tutorial date: Sai Gon 24/08/2004)

FRIENDS SITE

[\[Exetools Forum\]](#) | [\[HVAOnline\]](#) | [\[Vncracking Group\]](#) | [\[REA Forum\]](#) | [\[hacnho's homepage\]](#) | [\[Team AR\]](#)

| [\[diablo2oo2's\]](#) | [\[Devilz Crack\]](#) |

...: Copyright © 2004 by hacnho <[==]> VCT-Vietnamese Cracking Team 2k4:: ..

hacnho Tutorials # 3

Manual unpacking y0da's Crypter v1.2 template by koncool and R @ dier.

Information	Unpacking for Newbie's
Target	Unpackme.exe
Available	http://nhandan.info/hacnho/tuts/unpackme3_tuts.zip
Tools	OllyDbg plugin with 1:10 OllyDump 2.21.108, Lord PE 1.4, PESniffer 3.2b.
Protection	y0da's Crypter v1.2 by y0da
Level	Standard
Category	Manual unpacking

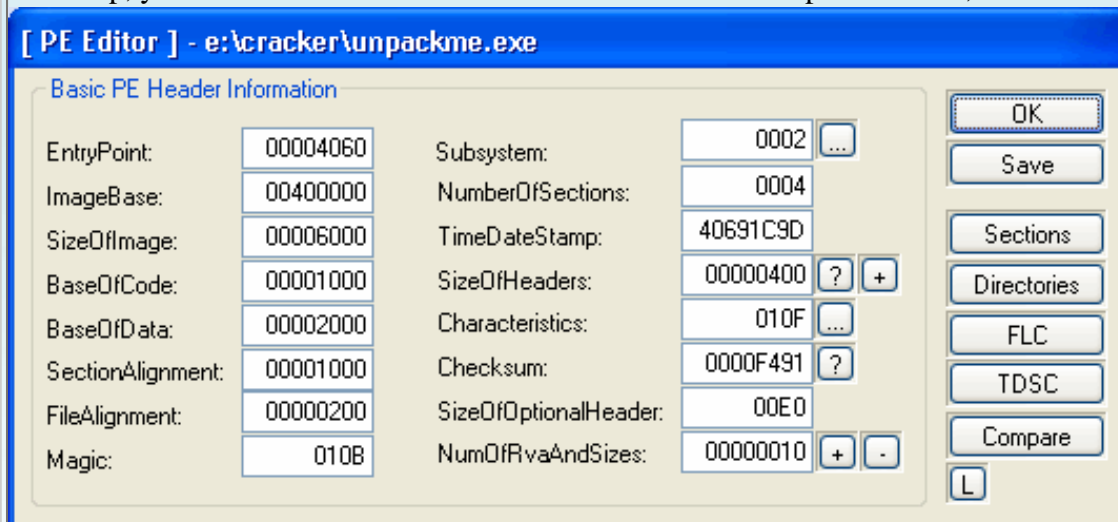
1. Introduction

Hi all,

In my 3rd tutorial. I will explain the manual method for unpacking a famous crypter. This is a crypter of y0da bro: **y0da's Crypter v1.2**. OK tested on Windows XP.

2. Getting Started

First step, you have to find some info from this PE software. Open Lord PE, PE Editor choose. And we have:



[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	00000026	00000400	00000200	E0000020
.rdata	00002000	00000092	00000600	00000200	C0000040
.data	00003000	00000096	00000800	00000200	C0000040
yC	00004000	00002000	00000A00	00000A46	E00000E0

EP: 4060, The Flags is (E0000020; C0000040; C0000040; E00000E0), Image Base is always 400000, Import Table: 4000 and is the size 3C.

3. Finding the OEP

I often use for PEiD detect OEP. But in this case, PEiD can not search the OEP. So, I have to search OEP "by hand". Okie, now use Olly for OEP detect. Load into unpackme.exe Olly. We still in line **00404060> 60 PUSHAD**. Now, press F7 to trace to address 404,061. At this line, you see in the **Registers (FPU)** table. The value of **ESP** is **0012FFA4**, then you right click on and choose **ESP Follow in the dump**.

The screenshot shows the OllyDbg interface. The assembly window displays code starting at address 00404060. The registers window on the right shows the ESP register with the value 0012FFA4, which is circled in green. A right-click context menu is open over the ESP register, with the option 'Follow in Dump' highlighted. The menu also includes options like 'Follow in Stack', 'View MMX registers', 'View 3DNow! registers', 'View debug registers', 'Make Label', and 'Appearance'.

Then you go to the Hex dump window. Then right click on the value **0012FFA4** and select Breakpoint -> Hardware, on Access -> Word. Our breakpoint is now set. Then Press F9 to run. Olly was break here:

The screenshot shows the assembly window at address 00404750. The code is as follows:

```

00404750 50      PUSH EAX
00404751 33C0    XOR EAX,EAX
00404752 64:FF30 PUSH DWORD PTR FS:[EAX]
00404753 64:8920 MOV DWORD PTR FS:[EAX],ESP
00404754 EB 01   JMP SHORT unpackme.00404769
00404755 8700    XCHG DWORD PTR DS:[EAX],EAX
00404756 0000    ADD BYTE PTR DS:[EAX],AL
00404757 0000    ADD BYTE PTR DS:[EAX],AL

```

Now, this is a very important step for finding the OEP. You must follow my step for Practice correct. OK, if

you understand, we continue ...

At Push EAX 40475D line, you press F7 to trace downward. And still line at **00404766 / EB 01 JMP**

SHORT unpackme.00404769. Here, you press F9 two times and then press Shift + F9 one time. And we have:

00401002	68	08 68	CHAR 'h'
00401003	00	08 00	
00401004	30	08 30	CHAR '0'
00401005	40	08 40	CHAR 'e'
00401006	00	08 00	
00401007	68	08 68	CHAR 'h'
00401008	1E	08 1E	

Et puis, you must press **Ctrl + A** for analyze. And you see:

00401000	6A 00	PUSH 0	
00401002	68 00304000	PUSH unpackme.00403000	ASCII "UCT-Vietnamese Crackers TEAM!"
00401007	68 00304000	PUSH unpackme.0040301E	ASCII "Try to Unpack me! Packed with y0da's Crypter v1.2"
0040100C	6A 00	PUSH 0	
0040100E	E8 0D000000	CALL unpackme.00401020	
00401013	6A 00	PUSH 0	
00401015	E8 00000000	CALL unpackme.0040101A	
0040101A	FF25 00204000	JMP DWORD PTR DS:[4020001]	
00401020	FF25 00204000	JMP DWORD PTR DS:[4020001]	OEP we found: 401000
00401026	00	DB 00	
00401027	00	DB 00	

Congratulations! According to OEP we found is 401,000. And now we Calculate the real OEP of this unpackme the Formule:

Real OEP = OEP find in Olly-Image Base = 401000-400000 = **1000**.

4. Dumping

At address **00401002**, we go to the menu Plugin -> OllyDump -> dump debugged process.

OllyDump - unpackme.exe

Start Address: 400000 Size: 6000 Dump

Entry Point: 4060 -> Modify: **1002** Get EIP as OEP Cancel

Base of Code: 1000 Base of Data: 2000

Change to 1000

☒ Fix Raw Size & Offset of Dump Image

Sect...	Virtual Size	Virtual Off...	Raw Size	Raw Offset	Characteris...
.text	00000026	00001000	00000026	00001000	E0000020
.rdata	00000092	00002000	00000092	00002000	C0000040
.data	00000096	00003000	00000096	00003000	C0000040
yC	00002000	00004000	00002000	00004000	E00000E0

Change to E0000060

☒ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Change modify the Entry Point to **1000**. And change the first flag (characteristics in Olly) to **E0000020**. And then, just press dump, save the file at **dumped.exe** unpacked.

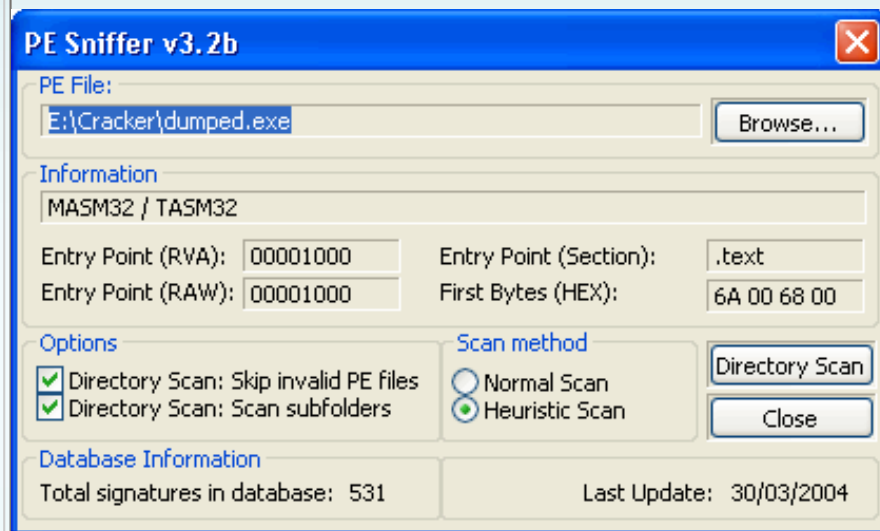
5. Finding and Fixing the Adress Import Table

Please remember: Do not fix IAT and rebuild it. Unpackme will be crash.

6. Testing Our Unpacked file

Now run unpacked files. Wow, not crash.

Using PESniffer 3.2b for detect: **MASM32 / TASM32**. Okie, y0da's Crypter v1.2 by **y0da** is now unpacked successful!



7. Conclusion

Special thanx to **koncool** et **R @ dier** for this template.

My Greetz to: Deux, RCA, Moonbaby, Computer_Angel, tlandn, [R @ dier](#), Zombie, Maipt0301, tykhung, softcracker_vn, CTL, LeVuHoang ...

To be continued ...

Written by hacnho (tutorial date: Sai Gon 30/3/2004)

UAManaIUnpick theZReippiraTool2.3

I-- Print d o t r u c t i o n:

Zip Repair Tool version before they have meat but with this new version level Protect its advanced than many. Version With this, the doctors can use a **script bypass Antidebug** can Fix IAT is able to run before nhucac tut they presented. But in this tut they want to present methods for MUP to OEP of Target.

II-- Tools T & g e r a t:

• T o o L and P l u g i n c a n d ú n g:

- O L L Y D B G _ E x c e r Y P t o r 1.10
- I k e s e r x p l o r
- I m p O R T R E C 1.6f
- R D G P A C K D e r e c t o r t o v 0.6.5
- C F F E x p l o r e r V

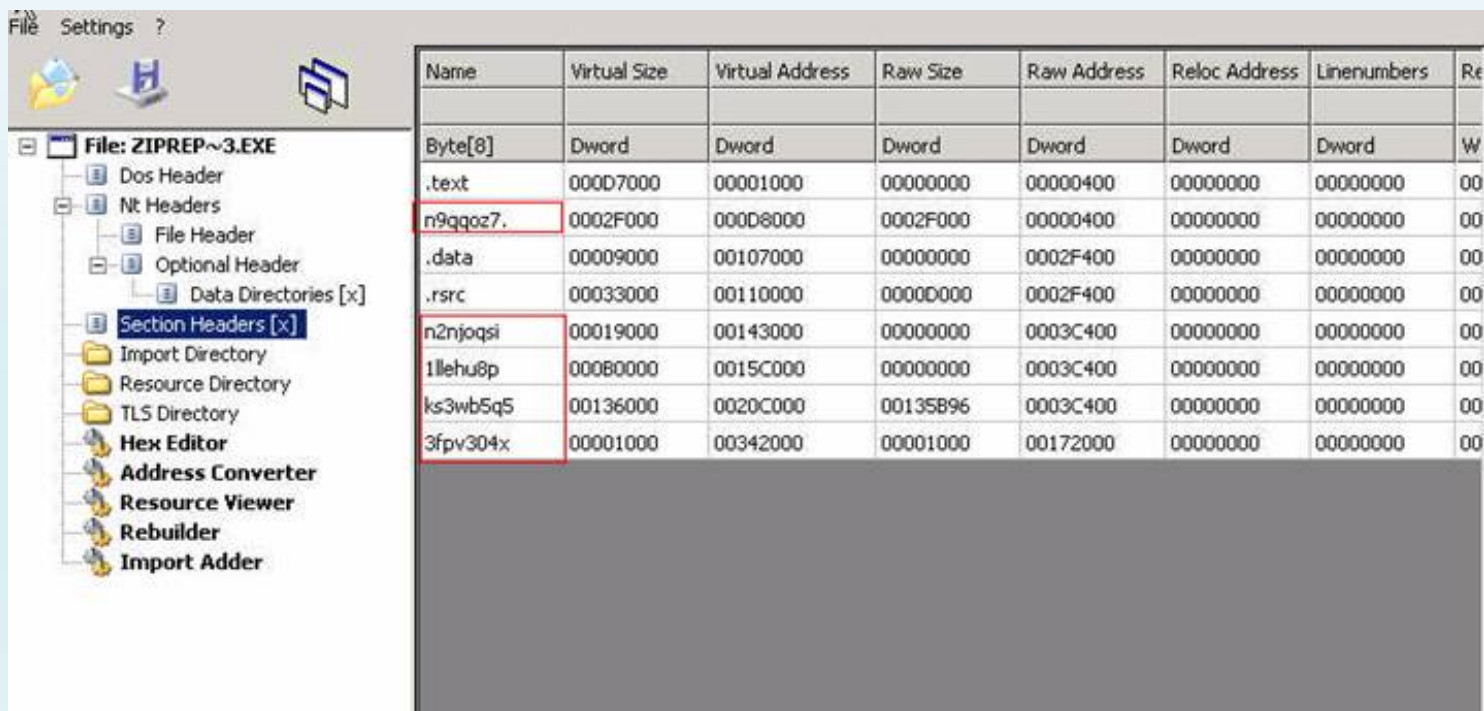
• O n e g e r t: **Zip R e p l a i r T o o 3.2**

III - B i t a l y s s P a n a t i D e b u g & F i n d O P E:

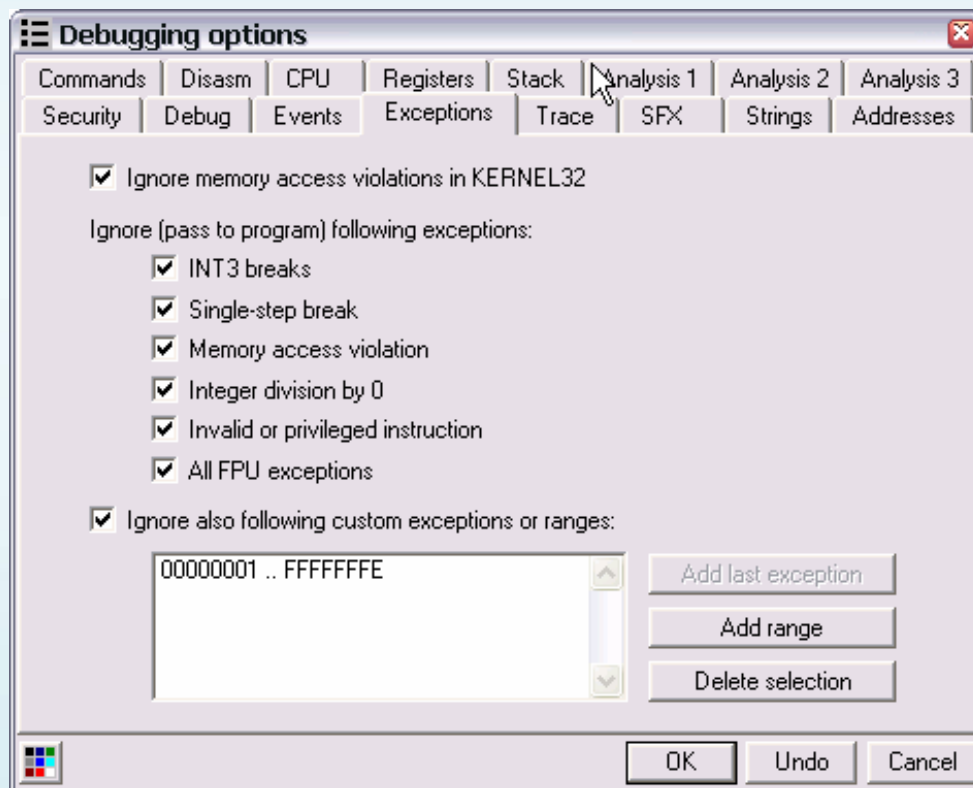
_ D ù n R D G P A C K g e r D e t e r c v 0.6.5 t o s c a n a g e r a t



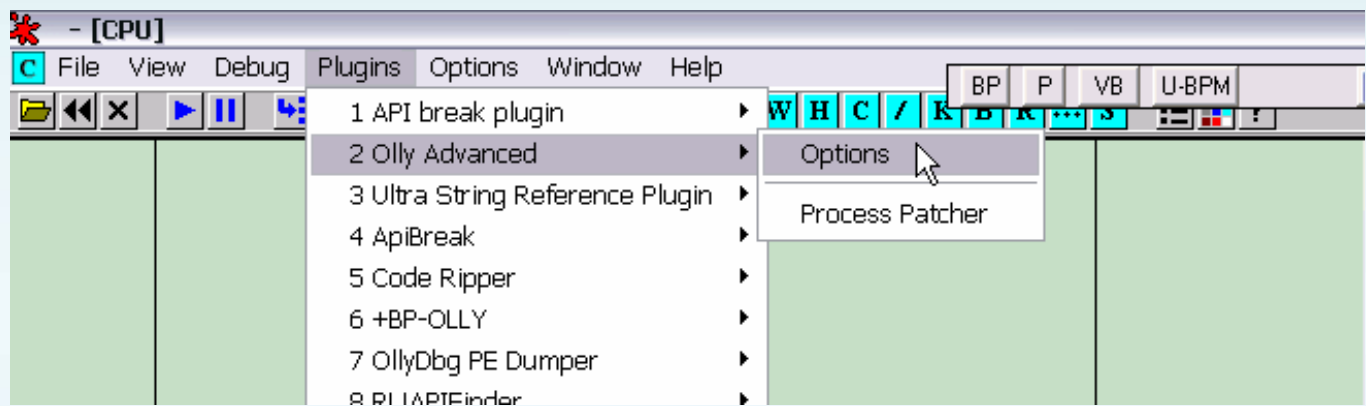
_ Open with CFF, The **Red Section** below may be a sign of the pack with target Execrypto R.



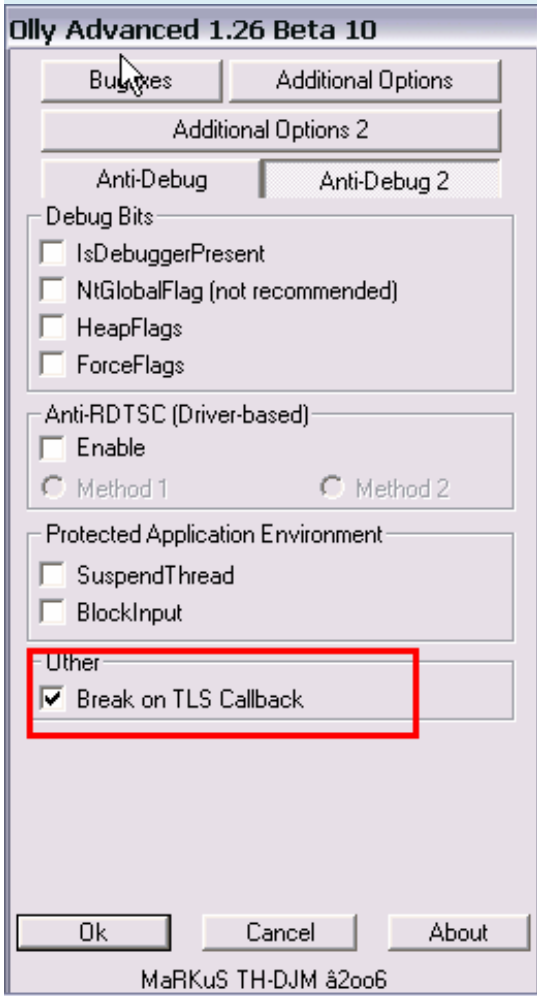
_OK OllyDBG_ExeCryptor open, press Alt + O and the following



_ Select Olly Advanced plugin and like



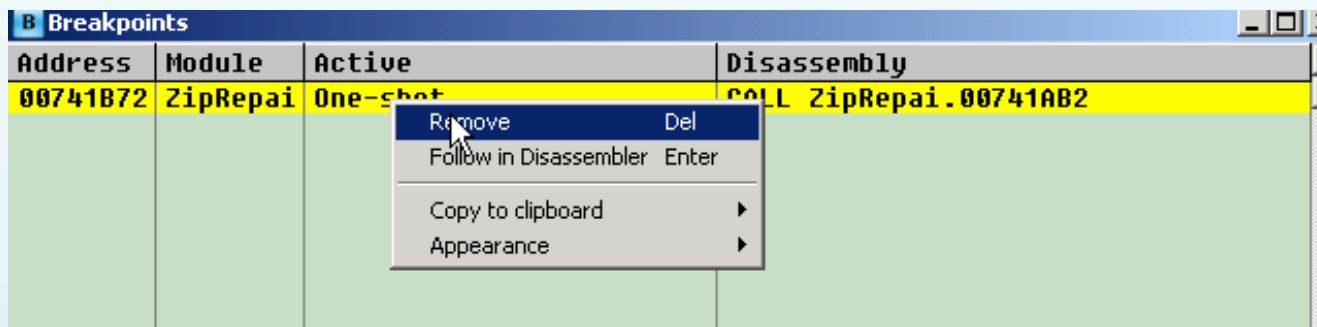
_Nhusau to choose:



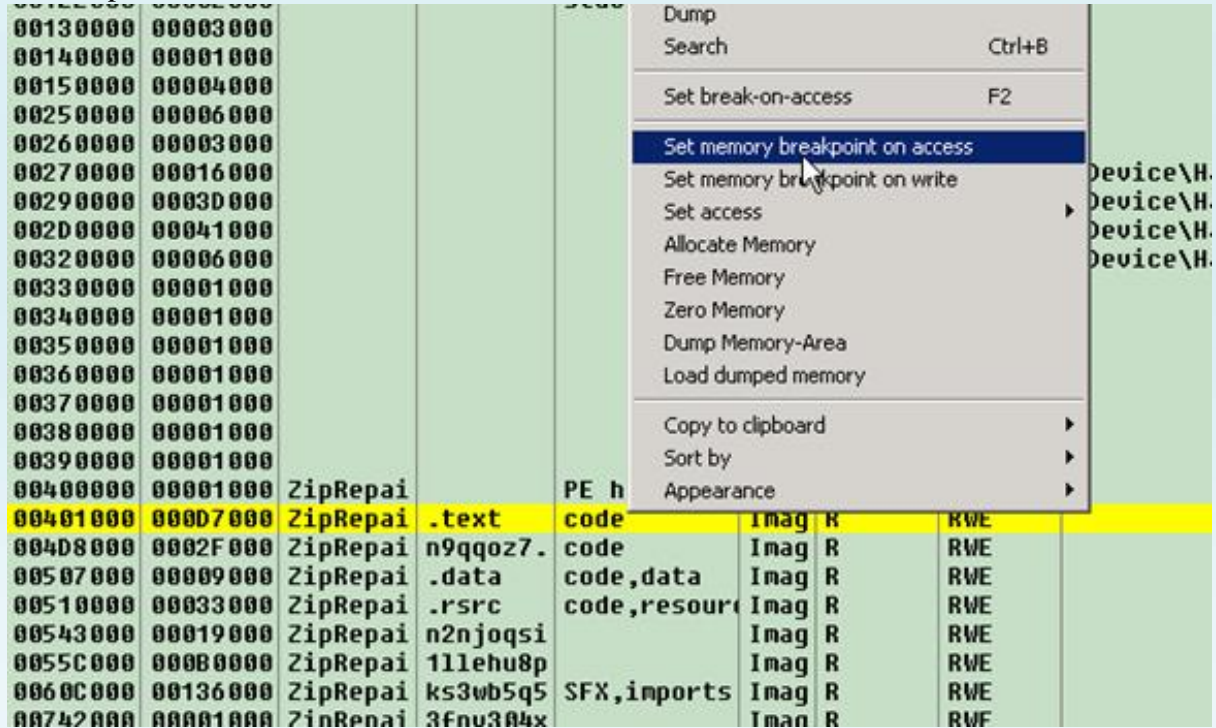
_ OK, Load OllyDBG_EXEcryptor on target and we stop here:

Address	Hex dump	Disassembly	Comment
00741B7E	E8 2FFFFFFF	CALL ZipRepai.00741AB2	
00741B83	05 CF3D0000	ADD EAX, 3DCF	
00741B88	FFE0	JMP NEAR EAX	
00741B8A	E8 04000000	CALL ZipRepai.00741B93	
00741B8F	FFFF	???	Unknown command
00741B91	FFFF	???	Unknown command
00741B93	5E	POP ESI	
00741B94	C3	RETN	
00741B95	00FF	ADD BH, BH	
00741B97	06	PUSH ES	
00741B98	A0 D2603F1F	MOV AL, BYTE PTR DS:[1F3F60D2]	
00741B9D	A3 C31EBAF2	MOV DWORD PTR DS:[F2BA1EC3], EAX	
00741BA2	9E	SAHF	
00741BA3	B2 ED	MOV DL, 0ED	
00741BA5	90	NOP	
00741BA6	D349 A9	ROR DWORD PTR DS:[ECX-57], CL	
00741BA9	1F	POP DS	

1st press **Alt + B 1 Breakpoint** you see, please delete it



2nd press **Alt + M** and press **F2** to Set 1 BP on access in **Section. Code**



3. Press **Shift + F9** you stop here

Address	Hex dump	Disassembly	Comment
0073AF80	F3:A4	REP MOVSB BYTE PTR ES:[EDI], BYTE PTR DS	
0073AF82	31DB	XOR EBX, EBX	
0073AF84	5E	POP ESI	
0073AF85	EB 9D	JMP SHORT ZipRepai.0073AF24	
0073AF87	89F0	MOV EAX, ESI	
0073AF89	5B	POP EBX	
0073AF8A	5F	POP EDI	
0073AF8B	5E	POP ESI	
0073AF8C	C3	RETN	
0073AF8D	8BC0	MOV EAX, EAX	
0073AF8F	0F89 8E150000	JNS ZipRepai.0073C523	
0073AF95	0F80 F4520000	JO ZipRepai.0074028F	
0073AF9B	5B	POP EBX	
0073AF9C	E9 67290000	JMP ZipRepai.0073D908	
0073AFA1	008B CC81C110	ADD BYTE PTR DS:[EBX+10C181CC], CL	
0073AFA7	0000	ADD BYTE PTR DS:[EAX], AL	
0073AFA9	008B 09C70113	ADD BYTE PTR DS:[EBX+1301C709], CL	

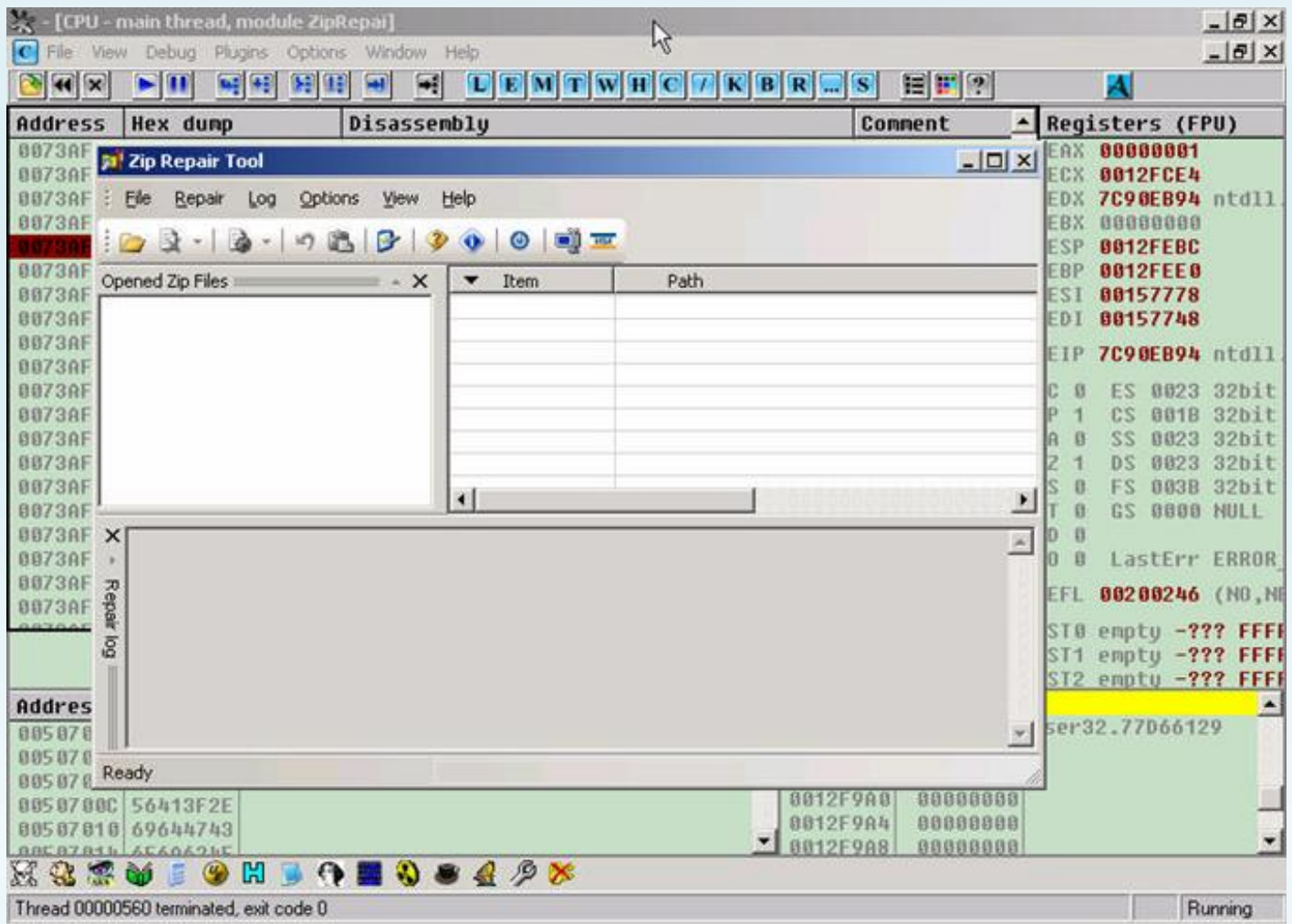
4th Set 1 BP at "0073AF87 89F0 MOV EAX, ESI"

Address	Hex dump	Disassembly	Comment
0073AF80	F3:A4	REP MOVSB BYTE PTR ES:[EDI], BYTE PTR DS	
0073AF82	31DB	XOR EBX, EBX	
0073AF84	5E	POP ESI	
0073AF85	EB 9D	JMP SHORT ZipRepair.0073AF24	
0073AF87	89F0	MOV EAX, ESI	
0073AF89	5B	POP EBX	
0073AF8A	5F	POP EDI	
0073AF8B	5E	POP ESI	
0073AF8C	C3	RET	
0073AF8D	8BC0	MOV EAX, EAX	
0073AF8F	0F89 8E150000	JNS ZipRepair.0073C523	

5. Press **Alt + M** to remove access on BP in **Section. Code**

00150000	00004000								
00250000	00006000								
00260000	00003000								
00270000	00016000								
00290000	0003D000								
002D0000	00041000								
00320000	00006000								
00330000	00001000								
00340000	00001000								
00350000	00001000								
00360000	00001000								
00370000	00001000								
00380000	00001000								
00390000	00001000								
00400000	00001000	ZipRepair							
00401000	00007000	ZipRepair	.text	code	Imag	R		RWE	
00408000	0002F000	ZipRepair	n9qqoz7.	code	Imag	R		RWE	
00507000	00009000	ZipRepair	.data	code,data	Imag	R		RWE	
00510000	00033000	ZipRepair	.rsrc	code,resource	Imag	R		RWE	
00543000	00019000	ZipRepair	n2njoqsi		Imag	R		RWE	
0055C000	000B0000	ZipRepair	11lehu8p		Imag	R		RWE	

_ Click hold **Shift + F9** until Soft run completely nhuday is our successful **bypass AntiDebug** gọi



_ Press **Ctrl + G** and type in **401000** and we come

Address	Hex dump	Disassembly	Comment
00401000	8B4424 04	MOV EAX, DWORD PTR SS:[ESP+4]	
00401004	3D 0E000780	CMP EAX, 8007000E	
00401009	75 05	JNZ SHORT ZipRepai.00401010	
0040100B	E8 084F0100	CALL ZipRepai.00415F18	
00401010	50	PUSH EAX	
00401011	E8 8C440100	CALL ZipRepai.004154A2	
00401016	CC	INT3	
00401017	CC	INT3	
00401018	CC	INT3	
00401019	CC	INT3	
0040101A	CC	INT3	
0040101B	CC	INT3	
0040101C	CC	INT3	
0040101D	CC	INT3	
0040101E	CC	INT3	
0040101F	CC	INT3	

_ Click the image below Search

Address	Hex dump	Disasm
00401000	8B4424 04	MOV EAX, [401004]
00401004	3D 0E000780	CMP EAX, 0E000780
00401009	75 05	JNZ 0040100B
0040100B	E8 084F0100	CALL 00401010
00401010	50	PUSH EBX
00401011	E8 8C440100	CALL 00401016
00401016	CC	INT3
00401017	CC	INT3
00401018	CC	INT3
00401019	CC	INT3
0040101A	CC	INT3
0040101B	CC	INT3
0040101C	CC	INT3
0040101D	CC	INT3
0040101E	CC	INT3
0040101F	CC	INT3
00401020	53	PUSH EBX
00401021	8B5C24 08	MOV EAX, [401024]
00401025	57	PUSH ESI
00401026	8B7C24 10	MOV EAX, [40102C]
00401027	57	PUSH ESI

Address	Value	Comment
00507000	004D8AF0	ASCII "bad a
00507004	004DF5E0	ZipRepai.004
00507008	00000000	
0050700C	56413F2E	

Comment	Registers
EAX 00000000	EAX 00000000
ECX 0012F000	ECX 0012F000
EDX 7C90EE00	EDX 7C90EE00
EBX 00000000	EBX 00000000
ESP 0012FE00	ESP 0012FE00
EBP 0012FE00	EBP 0012FE00
ESI 00157700	ESI 00157700
EDI 00157700	EDI 00157700
EIP 7C90EE00	EIP 7C90EE00
C 0 ES 00	C 0 ES 00
P 1 CS 00	P 1 CS 00
A 0 SS 00	A 0 SS 00

Search for	Find references to	View	Copy to executable	Analysis
Name (label) in current module Ctrl+N	Name in all modules	Command Ctrl+F	Sequence of commands Ctrl+S	Constant Ctrl+B
All intermodular calls	All commands	All sequences	All constants	All switches
All referenced text strings				

_Go **GetVersionExA** and line 3 (She also dek subliminal bit What exactly when this API functions such bit soft through the code with **Visual C++** to use the trial and exclude new exactly this function)

0043D335	CALL NEAR DWORD PTR DS:[4081C0]	kernel32.GetTickCount
004BEEE7	CALL NEAR DWORD PTR DS:[4081C0]	kernel32.GetTickCount
004BEF97	CALL NEAR DWORD PTR DS:[4081C0]	kernel32.GetTickCount
004C3408	CALL NEAR DWORD PTR DS:[4081C0]	kernel32.GetTickCount
0043B215	CALL NEAR DWORD PTR DS:[408200]	kernel32.GetTimeZoneInformation
0041A3E7	CALL NEAR DWORD PTR DS:[408540]	user32.GetTopWindow
0041AC41	CALL NEAR DWORD PTR DS:[408540]	user32.GetTopWindow
00411B6D	CALL NEAR DWORD PTR DS:[408300]	kernel32.GetVersion
0042A03D	CALL NEAR DWORD PTR DS:[408300]	kernel32.GetVersion
0042B997	CALL NEAR DWORD PTR DS:[408300]	kernel32.GetVersion
004175A1	CALL NEAR DWORD PTR DS:[4083D4]	kernel32.GetVersionExA
00425181	CALL NEAR DWORD PTR DS:[4083D4]	kernel32.GetVersionExA
00432CFA	CALL NEAR DWORD PTR DS:[4083D4]	kernel32.GetVersionExA
0047C147	CALL NEAR DWORD PTR DS:[4083D4]	kernel32.GetVersionExA
0047D1EC	CALL NEAR DWORD PTR DS:[4083D4]	kernel32.GetVersionExA
004813D7	CALL NEAR DWORD PTR DS:[4083D4]	kernel32.GetVersionExA
00481796	CALL NEAR DWORD PTR DS:[4083D4]	kernel32.GetVersionExA
004844BB	CALL NEAR DWORD PTR DS:[4083D4]	kernel32.GetVersionExA
004BD348	CALL NEAR DWORD PTR DS:[4083D4]	kernel32.GetVersionExA
004BD56E	CALL NEAR DWORD PTR DS:[4083D4]	kernel32.GetVersionExA
004BF54D	CALL NEAR DWORD PTR DS:[4083D4]	kernel32.GetVersionExA
004BFB37	CALL NEAR DWORD PTR DS:[4083D4]	kernel32.GetVersionExA

_ Double Click on, roll on top of this function and that the **OEP of Target**

Address	Hex dump	Disassembly	Comment
00432CAB	C3	RETN	
00432CAC	6A 60	PUSH 60	<== This is OEP
00432CAE	68 B8A44F00	PUSH ZipRepai.004FA4B8	
00432CB3	E8 F4040000	CALL ZipRepai.004331AC	
00432CB8	8365 FC 00	AND DWORD PTR SS:[EBP-4], 0	
00432CBC	8D45 90	LEA EAX, DWORD PTR SS:[EBP-70]	
00432CBF	50	PUSH EAX	
00432CC0	FF15 44824D00	CALL NEAR DWORD PTR DS:[4D8244]	ZipRepai.00606C1E
00432CC6	C745 FC FFFFFFFF	MOV DWORD PTR SS:[EBP-4], -2	
00432CCD	BF 94000000	MOV EDI, 94	
00432CD2	57	PUSH EDI	
00432CD3	6A 00	PUSH 0	
00432CD5	8B1D 48824D00	MOV EBX, DWORD PTR DS:[4D8248]	kernel32.GetProcessHeap
00432CD8	FFD3	CALL NEAR EBX	
00432CDD	50	PUSH EAX	
00432CDE	FF15 60824D00	CALL NEAR DWORD PTR DS:[4D8260]	ntdll.RtlAllocateHeap
00432CE4	8BF0	MOV ESI, EAX	
00432CE6	85F6	TEST ESI, ESI	
00432CE8	75 00	JNZ SHORT ZipRepai.00432CF7	
00432CEA	6A 12	PUSH 12	
00432CEC	E8 56FFFFFF	CALL ZipRepai.00432C47	
00432CF1	59	POP ECX	
00432CF2	E9 8A010000	JMP ZipRepai.00432E81	
00432CF7	893E	MOV DWORD PTR DS:[ESI], EDI	
00432CF9	56	PUSH ESI	
00432CFA	FF15 D4834D00	CALL NEAR DWORD PTR DS:[4D83D4]	kernel32.GetVersionExA
00432D00	56	PUSH ESI	

_ We easily determine the position start and end of the table IAT. We need to define the parameters of this script to be used for quick Fix IAT

Address	Value	Comment
004D8000	00596EB6	ZipRepai.00596EB6
004D8004	005BE050	ZipRepai.005BE050
004D8008	00564D9D	ZipRepai.00564D9D
004D800C	00549F21	ZipRepai.00549F21
004D8010	005797D9	ZipRepai.005797D9
004D8014	00567DDA	ZipRepai.00567DDA
004D8018	00577461	ZipRepai.00577461
004D801C	005629E5	ZipRepai.005629E5
004D8020	005771AF	ZipRepai.005771AF
004D8024	0058A693	ZipRepai.0058A693
004D8028	005B6FED	ZipRepai.005B6FED
004D802C	005ADF3D	ZipRepai.005ADF3D
004D8030	005E78E9	ZipRepai.005E78E9
004D8034	0059329A	ZipRepai.0059329A

IAT Start

Address	Value	Comment
004D87F0	77578712	ole32.OleCreateMenuDescriptor
004D87F4	775789AC	ole32.OleDestroyMenuDescriptor
004D87F8	77578404	ole32.OleTranslateAccelerator
004D87FC	7757825B	ole32.IsAccelerator
004D8800	775C7FFC	ole32.OleLockRunning
004D8804	7752CADE	ole32.OleGetClipboard
004D8808	774F4F91	ole32.RegisterDragDrop
004D880C	7750B899	ole32.CoLockObjectExternal
004D8810	774F5051	ole32.RevokeDragDrop
004D8814	774F974A	ole32.CreateStreamOnHGloab1
004D8818	00000000	IAT End
004D881C	00000000	
004D8820	004D6922	ZipRepai.004D6922
004D8824	004D692F	ZipRepai.004D692F

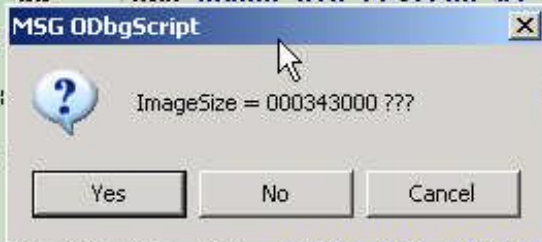
V I D E-Fix I A & T E R I build an MP or

_ Once you've collected enough information to press **Ctrl + F2** and make italy chang nhucac steps **1 -> 5**. Then press **Shift + F9** 2 times and press **Ctrl + G** complete address **OEP = 432CAC** and we come

Address	Hex dump	Disassembly	Comment
00432CA9	33C0	XOR EAX, EAX	
00432CAB	C3	RETN	
00432CAC	6A 60	PUSH 60	<=== This is OEP
00432CAE	68 B8A44F00	PUSH ZipRepai.004FA4B8	
00432CB3	E8 F4040000	CALL ZipRepai.004331AC	
00432CB8	8365 FC 00	AND DWORD PTR SS:[EBP-4], 0	
00432CBC	8D45 90	LEA EAX, DWORD PTR SS:[EBP-70]	
00432CBF	50	PUSH EAX	
00432CC0	FF15 44824D00	CALL NEAR DWORD PTR DS:[4D8244]	ZipRepai.00606C1E
00432CC6	C745 FC FFFFFFFF	MOV DWORD PTR SS:[EBP-4], -2	
00432CCD	BF 94000000	MOV EDI, 94	
00432CD2	57	PUSH EDI	
00432CD3	6A 00	PUSH 0	
00432CD5	8B1D 48824D00	MOV EBX, DWORD PTR DS:[4D8248]	ZipRepai.005BBAF0
00432CDB	FFD3	CALL NEAR EBX	
00432CDD	50	PUSH EAX	
00432CDE	FF15 60824D00	CALL NEAR DWORD PTR DS:[4D8260]	ZipRepai.005FBB30
00432CE4	8BF0	MOV ESI, EAX	

_ Set 1 BP in OEP and press Shift + F9 and again as we stop at the OEP. Remove the existing BP and started to run script automatically Fix IAT (*IAT sure to start and edit the script for End punctuality*)

Address	Hex dump	Disassembly	Comment
00432CA9	33C0	XOR EAX, EAX	
00432CAB	C3	RETN	
00432CAC	6A 60	PUSH 60	<=== This is OEP
00432CAE	68 B8A44F00	PUSH ZipRepai.004FA4B8	
00432CB3	E8 F4040000	CALL ZipRepai.004331AC	
00432CB8	8365 FC	AND EBX, EAX	
00432CBC	8D45 90	MOV EBX, [EBX+70]	
00432CBF	50	POP EAX	
00432CC0	FF15 44	JMP [EBX+08244]	ZipRepai.00606C1E
00432CC6	C745 FC	CMOVBE EAX, ECX	
00432CCD	BF 9400	MOV EBX, [EBX+2]	
00432CD2	57	JNB EBX	
00432CD3	6A 00	PUSH 0	
00432CD5	8B1D 48824D00	MOV EBX, DWORD PTR DS:[408248]	ZipRepai.005BBAF0
00432CDB	FFD3	CALL NEAR EBX	
00432CDD	50	POP EAX	
00432CDE	FF15 60824D00	CALL NEAR DWORD PTR DS:[408260]	ZipRepai.005FBB30
00432CE4	8BF0	MOV ESI, EAX	



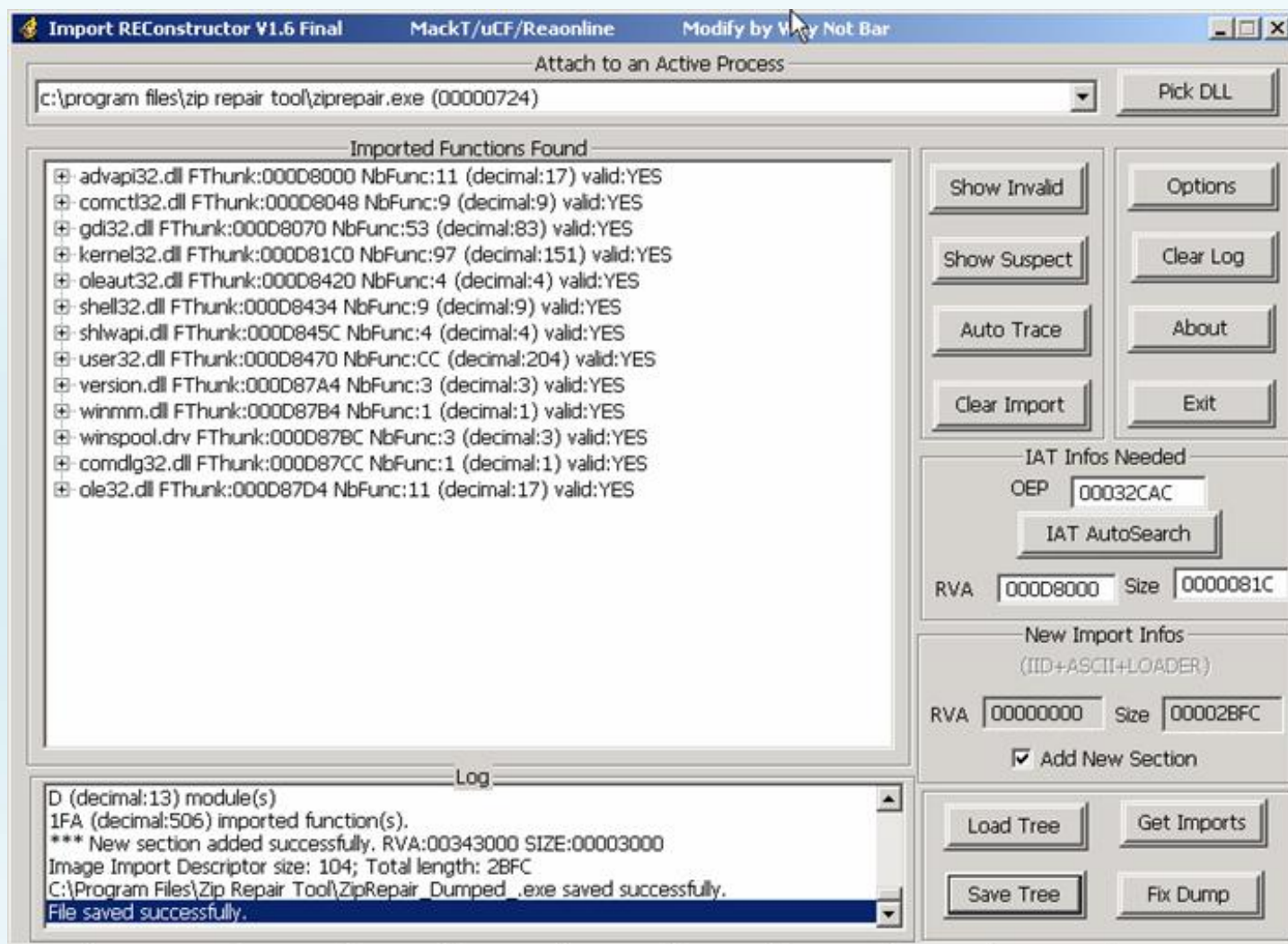
_ After waiting Script finished running you can complete the IAT

Address	Value	Comment
004D8000	77DDEBE7	advapi32.RegSetValueExA
004D8004	77DDEAF4	advapi32.RegCreateKeyExA
004D8008	77DD7883	advapi32.RegQueryValueExA
004D800C	77DD761B	advapi32.RegOpenKeyExA
004D8010	77DD07CC	advapi32.RegSetValueExW
004D8014	77DD7535	advapi32.RegCreateKeyExW
004D8018	77DFC1B5	advapi32.RegQueryInfoKeyA
004D801C	77DECF4A	advapi32.RegEnumValueA
004D8020	77DFC8C1	advapi32.RegEnumKeyExA
004D8024	77DD6FC8	advapi32.RegQueryValueExW
004D8028	77DD6A78	advapi32.RegOpenKeyExW
004D802C	77DFC41B	advapi32.RegOpenKeyA
004D8030	77DD6BF0	advapi32.RegCloseKey
004D8034	77DFCC10	advapi32.RegQueryValueA
004D8038	77DFCAC3	advapi32.RegEnumKeyA
004D803C	77DFC123	advapi32.RegDeleteKeyA
004D8040	77DDEDE5	advapi32.RegDeleteValueA
004D8044	00000000	
004D8048	5D0B15DD	COMCTL32.InitCommonControls
004D804C	5D0B02FC	COMCTL32._TrackMouseEvent
004D8050	5D0C1D6B	COMCTL32.ImageList_GetIcon
004D8054	5D0A129B	COMCTL32.ImageList_DrawEx
004D8058	5D09D440	COMCTL32.ImageList_ReplaceIcon
004D805C	5D098B5B	COMCTL32.ImageList_Create
004D8060	5D0B0B2E	COMCTL32.ImageList_GetIconSize
004D8064	5D09BD2E	COMCTL32.ImageList_Destroy
004D8068	5D09E1C2	COMCTL32.ImageList_GetImageCount
004D806C	00000000	
004D8070	77F16A3B	GDI32.DeleteObject

Full dump _ with

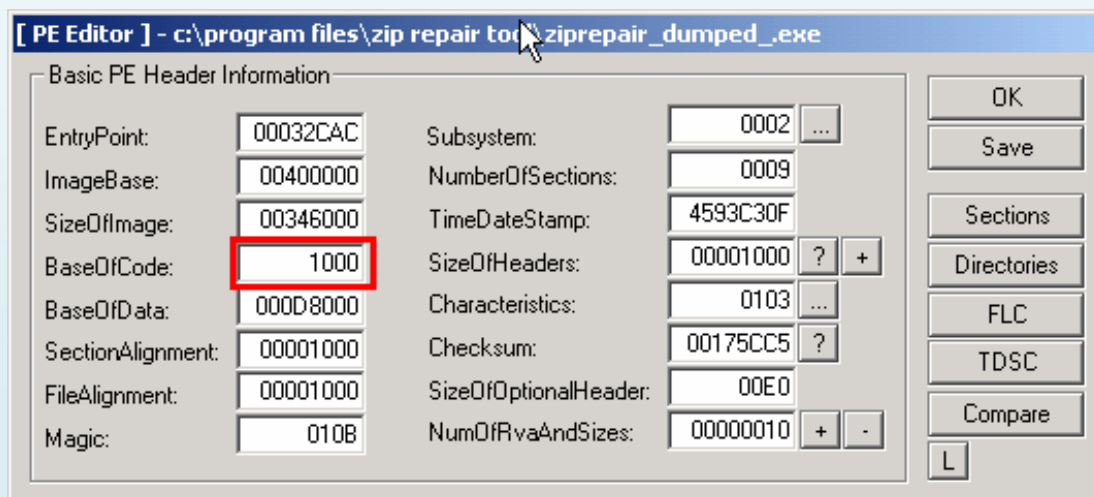
Process	Id	Address	Size	Path	Description
svchost	0000032C	01000000	00006000	C:\WINDOWS\system32\svchost.exe	Generic Host Process for Win32 S
svchost	0000036C	01000000	00006000	C:\WINDOWS\system32\svchost.exe	Generic Host Process for Win32 S
nvsvc32	000003E0	00400000	0001E000	C:\WINDOWS\system32\nvsvc32.exe	NVIDIA Driver Helper Service, Ver
alg	00000508	01000000	0000D000	C:\WINDOWS\System32\alg.exe	Application Layer Gateway Servic
Explorer	000006E4	01000000	000FF000	C:\WINDOWS\Explorer.EXE	Windows Explorer
WINWORD	00000178	30000000	00BAA000	C:\Program Files\Microsoft Office\OFFICE11\WIN...	Microsoft Office Word
UniKeyNT	000003CC	00400000	00041000	F:\Cool Data\Van Phong\uk40Bnt\UniKey\UniKeyNT...	
HprSnap6	00000564	00400000	00419000	C:\Program Files\HyperSnap 6\HprSnap6.exe	HyperSnap
Olllydbg_Exec...	000001D0	00400000	00164000	D:\Crack tool\OlllyDbg_Execryptor\Olllydbg_Execry...	OlllyDbg, 32-bit analysing debugg
NOTEPAD	000004A0	01000000	00014000	C:\WINDOWS\system32\NOTEPAD.EXE	Notepad
ZipRepair	00000724	Refresh		C:\Program Files\Zip Repair Tool\ZipRepair.exe	Zip Repair Tool MFC Application
NOTEPAD	000003DC	Dump PE		C:\WINDOWS\system32\NOTEPAD.EXE	Notepad
Task Explorer	000003FC	Dump Region		D:\Crack tool\Other\CFF Explorer\Task Explorer.exe	Task Explorer
		Priority			
Module	Address				Description
ZipRepair	00400000	Kill		n Files\Zip Repair Tool\ZipRepair.exe	Zip Repair Tool MFC Application
ntdll	7C900000	Open Folder		WS\system32\ntdll.dll	NT Layer DLL
kernel32	7C800000	File Properties		WS\system32\kernel32.dll	Windows NT BASE API Client DLL
user32	77D40000	Open with CFF Explorer		WS\system32\user32.dll	Windows XP USER API Client DLL
GDI32	77F10000			WS\system32\GDI32.dll	GDI Client DLL
advapi32	77DD0000			WS\system32\advapi32.dll	Advanced Windows 32 Base API
RPCRT4	77E70000		00091000	C:\WINDOWS\system32\RPCRT4.dll	Remote Procedure Call Runtime
COMCTL32	5D090000		00097000	C:\WINDOWS\system32\COMCTL32.dll	Common Controls Library
OLEAUT32	77120000		0008C000	C:\WINDOWS\system32\OLEAUT32.dll	
msvrrt	77C10000		00058000	C:\WINDOWS\system32\msvrrt.dll	Windows NT CRT DI I

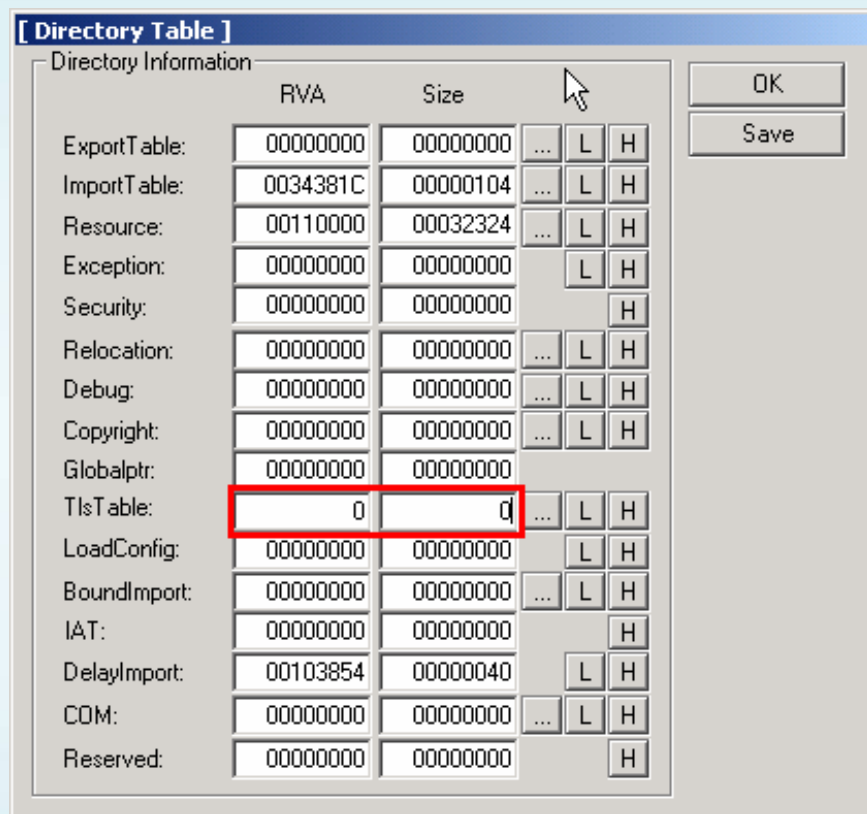
_ Open **ImportREC** up. Select **List** in **Process ZipRepair.exe**. **OEP** = Enter **00432CAC - 00,400,000 (Imagebase)**
= **00032CAC**, **IAT AutoSearch** Click -> **Get Imports** -> **Show Invalid**



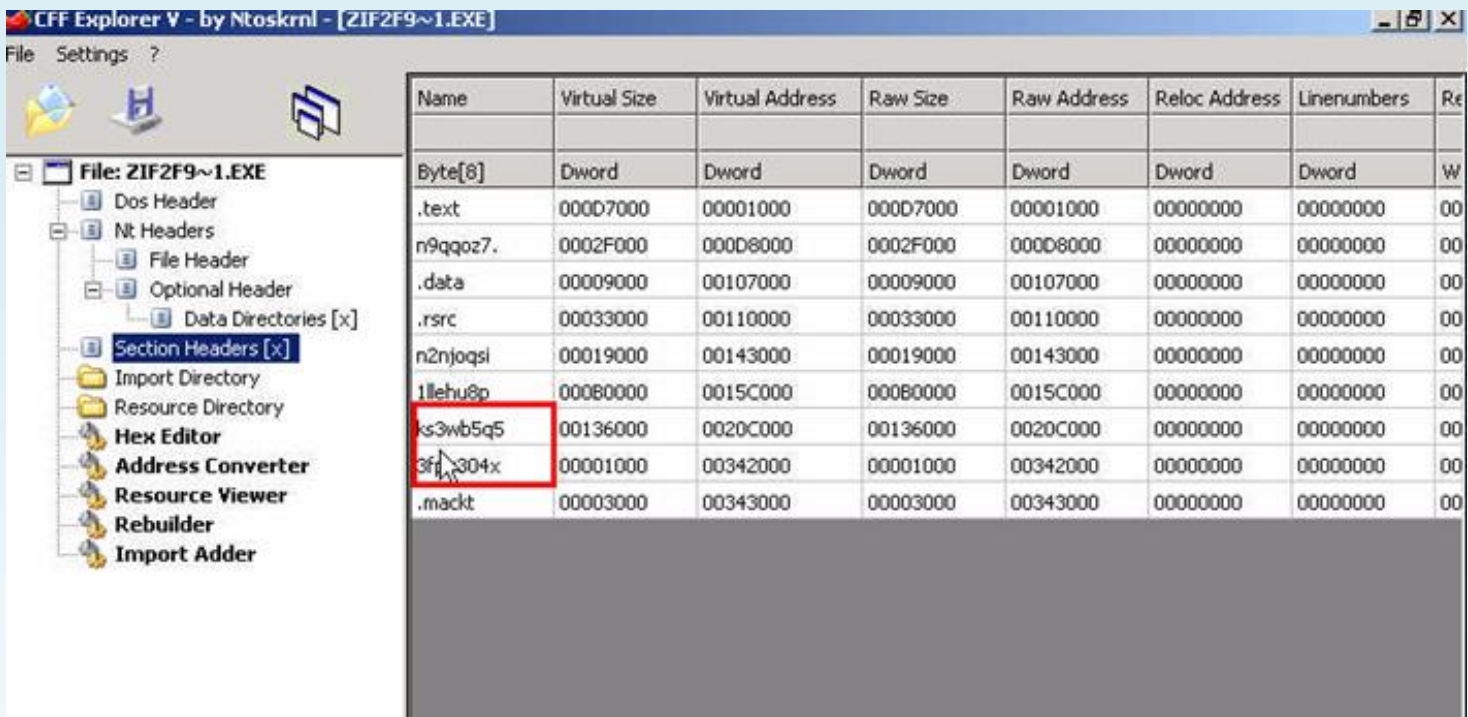
_ Hú complete I have any function **Invalid** ... OK, Click **Fix Dumped ZipRepair_Dumped .exe** File and select Run ... khua khua try .. **Unpack** ... **Done!** Respectable finish ... This is also what i know the contact of **Trickyboy**, also want to patch or Keygen's meeting of the **dump**. Newbie She is just what i should start Hahaha.

_ Also forgot to load File **ZipRepair_Dumped .exe** LordPE and Fix the nhusau





_ Save the run and try running nhuthuong, continue to load new File Save to **CFF**



Section 2 _Xoa with red circle and Save again.

G r l Ee TsF italy Ou the Co mpu t e r A _ of e l, e mbi Z o, M A B oo nb italy, H o acnh, Nina B e, e ki nman o

w ar, Z o i D e ux, M e r c, li ght to nix o e, i T r o c kyb italy, Takad a iamidi ot, of the e n t e n handi ... and italy

o u!

The N h a n a g, Day 2 7 th a n g2 20 0 7

Who titaly N Bar

CopyMemII Debugblocker Nanomites

SoftWare : **Movie Collector 4.4**
Packed : **Armadillo 4.xx - CopyMemII + + Debugblocker Nanomites**
Crack Tool **1.** **OllyDBG by hacnho.**
2. LordPE Deluxe 1.4-by yoda
3.Import REConstructor 1.6 Final
4. ArmInline 0.71
Author : **Why Not Bar**

Target With this he has Benina tut but then use Tool **ArmTools03f.exe**. In fact is the video or very intuitive to Newbie as they can quickly be practice (Thanks to **Benina**). But using gas Tool trouble when they use it for meat man allowed this month in Tool ArmInline 0.71. Please say just before they set out a method to unpack with soft **Nanomites** not be repaired tut about his Benina because it is too then edit! OK. Only start time?

On the treatment **CopyMemII + Debugblocker** According to the default Benina to know you do. If you do not know Uncle can find tut by Hacnho or your are to see more. Here they work fast because the mainly **Nanomites Fix**.

Target _Load to Olly

006F2403	55	PUSH EBP	
006F2404	8BEC	MOV EBP,ESP	
006F2406	6A FF	PUSH -1	
006F2408	68 20CB7100	PUSH MovieCol.0071CB20	
006F240D	68 10226F00	PUSH MovieCol.006F2210	SE handler installation
006F24E2	64:A1 00000001	MOV EAX,DWORD PTR FS:[0]	
006F24E8	50	PUSH EAX	
006F24E9	64:8925 00000001	MOV DWORD PTR FS:[0],ESP	
006F24F0	83EC 58	SUB ESP,58	
006F24F3	53	PUSH EBX	
006F24F4	56	PUSH ESI	
006F24F5	57	PUSH EDI	
006F24F6	8965 E8	MOV [LOCAL.6],ESP	
006F24F9	FF15 88717100	CALL NEAR DWORD PTR DS:[<&KERNEL32.GetVersion	kernel32.GetVersion
006F24FF	33D2	XOR EDX,EDX	
006F2501	8AD4	MOV DL,AH	
006F2503	8915 04F17100	MOV DWORD PTR DS:[71F104],EDX	

_Chay Scripts (with the tut) by [Benina](#) he wrote it or. Running Script finished one here

00401000	8105 A8CD1200	ADD DWORD PTR DS:[12CDA8],1000	
0040100A	8105 B4CD1200	ADD DWORD PTR DS:[12CDB4],1000	
00401014	8105 B8CD1200	ADD DWORD PTR DS:[12CDB8],1000	
0040101E	813D B8CD1200	CMP DWORD PTR DS:[12CDB8],MovieCol.0068	
00401028	0F85 28162E00	JNZ MovieCol.006E2656	
0040102E	68 68070000	PUSH 768	
00401033	E8 5993457C	CALL kernel32.DebugActiveProcessStop	
00401038	90	NOP	
00401039	0000	ADD BYTE PTR DS:[EAX],AL	
0040103B	0000	ADD BYTE PTR DS:[EAX],AL	
0040103D	0000	ADD BYTE PTR DS:[EAX],AL	
0040103F	0000	ADD BYTE PTR DS:[EAX],AL	
00401041	0000	ADD BYTE PTR DS:[EAX],AL	

_Mo Add more windows Attach PID, F9, F12, and the following patch

006890EC	55	PUSH EBP	
006890ED	8BEC	MOV EBP,ESP	
006890EF	83C4 DC	ADD ESP,-24	
006890F2	33C0	XOR EAX,EAX	
006890F4	8945 EC	MOV DWORD PTR SS:[EBP-14],EAX	
006890F7	B8 04896800	MOV EAX,MovieCol.00688904	
006890FC	E8 D7E9D7FF	CALL MovieCol.00407AD8	
00689101	33C0	XOR EAX,EAX	
00689103	55	PUSH EBP	
00689104	68 60926800	PUSH MovieCol.00689260	
00689109	64:FF30	PUSH DWORD PTR FS:[EAX]	
0068910C	64:8920	MOV DWORD PTR FS:[EAX],ESP	
0068910F	E8 1C36D0FF	CALL MovieCol.0045C730	

_Mo LordPE, Full dump, Next we Fix IAT. Close the window is working with the Child and Father Restart. Running Scripts "Armadillo Detach from Client." Running Script Done Attach one PID, F9, F12 to us here.

006F24D3	55	PUSH EBP	
006F24D4	8BEC	MOV EBP,ESP	
006F24D6	6A FF	PUSH -1	
006F24D8	68 20CB7100	PUSH MovieCol.0071CB20	
006F24D0	68 10226F00	PUSH MovieCol.006F2210	
006F24E2	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	SE handler installation
006F24E8	50	PUSH EAX	
006F24E9	64:8925 000000	MOV DWORD PTR FS:[0],ESP	
006F24F0	8BEC 58	SUB ESP,58	
006F24F3	53	PUSH EBX	
006F24F4	56	PUSH ESI	
006F24F5	57	PUSH EDI	
006F24F6	8965 F8	MOV DWORD PTR SS:[EBP-18],ESP	

_Chay Scripts "Armadillo Standard unpack" (with the tut)

006890EC	2270 65	AND DH,BYTE PTR DS:[EAX+65]	<- DEP
006890EF	40	INC EAX	
006890F0	B3 27	MOV BL,27	
006890F2	BA 03FEBE65	MOV EDX,65BEFE03	
006890F7	7B 73	JPD SHORT MovieCol.0068916C	
006890F9	72 E1	JB SHORT MovieCol.006890DC	
006890FB	C3	RETN	
006890FC	9F	LAHF	
006890FD	2C 60	SUB AL,60	
006890FF	14 88	ADC AL,88	
00689101	C8 49961F	ENTER 9649,1F	
00689105	00	WAIT	

_Ta Have **IAT Start:**

006961E0 ntdll.RtlDeleteCriticalSection 7C91188A

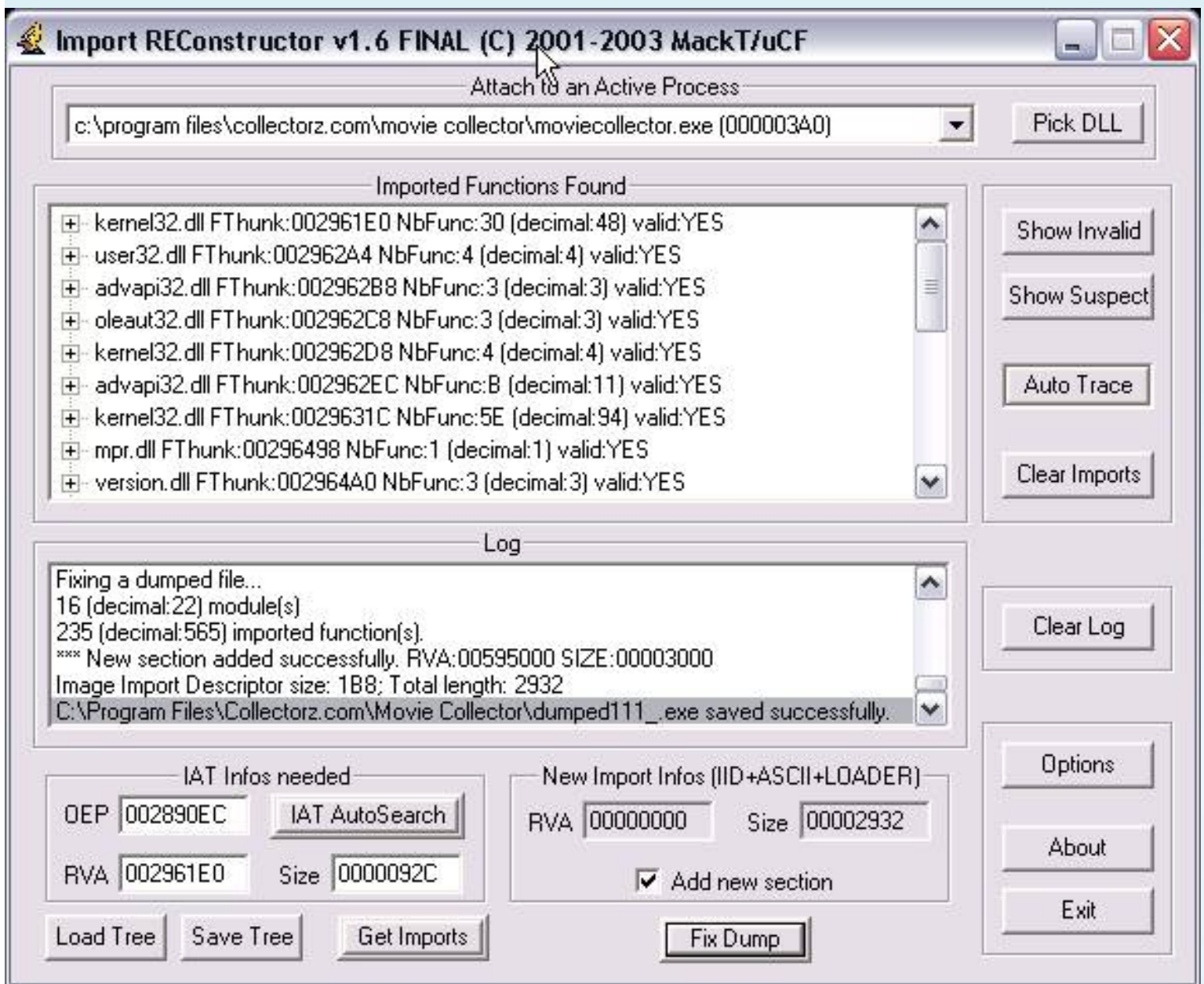
006961D8	00000000	
006961DC	00000000	
006961E0	7C91188A	ntdll.RtlDeleteCriticalSection
006961E4	7C9010ED	ntdll.RtlLeaveCriticalSection
006961E8	7C901005	ntdll.RtlEnterCriticalSection
006961EC	7C809FA1	kernel32.InitializeCriticalSection
006961F0	7C809B14	kernel32.VirtualFree

IAT and **end:** 00696B0C 7C8097F4 kernel32.MulDiv

00696AF0	00F359F2	
00696AF4	71A02E30	WSOCK32.setsockopt
00696AF8	71AB2BF4	WS2_32.inet_addr
00696AFC	71AB2BC0	WS2_32.ntohl
00696B00	00F3597E	
00696B04	76B44E5B	winmm.timeGetTime
00696B08	00F35A25	
00696B0C	7C8097F4	kernel32.MulDiv
00696B10	00F35A2A	
00696B14	6E72656B	

Len: 92C


_ Open ImportREC and enter parameters

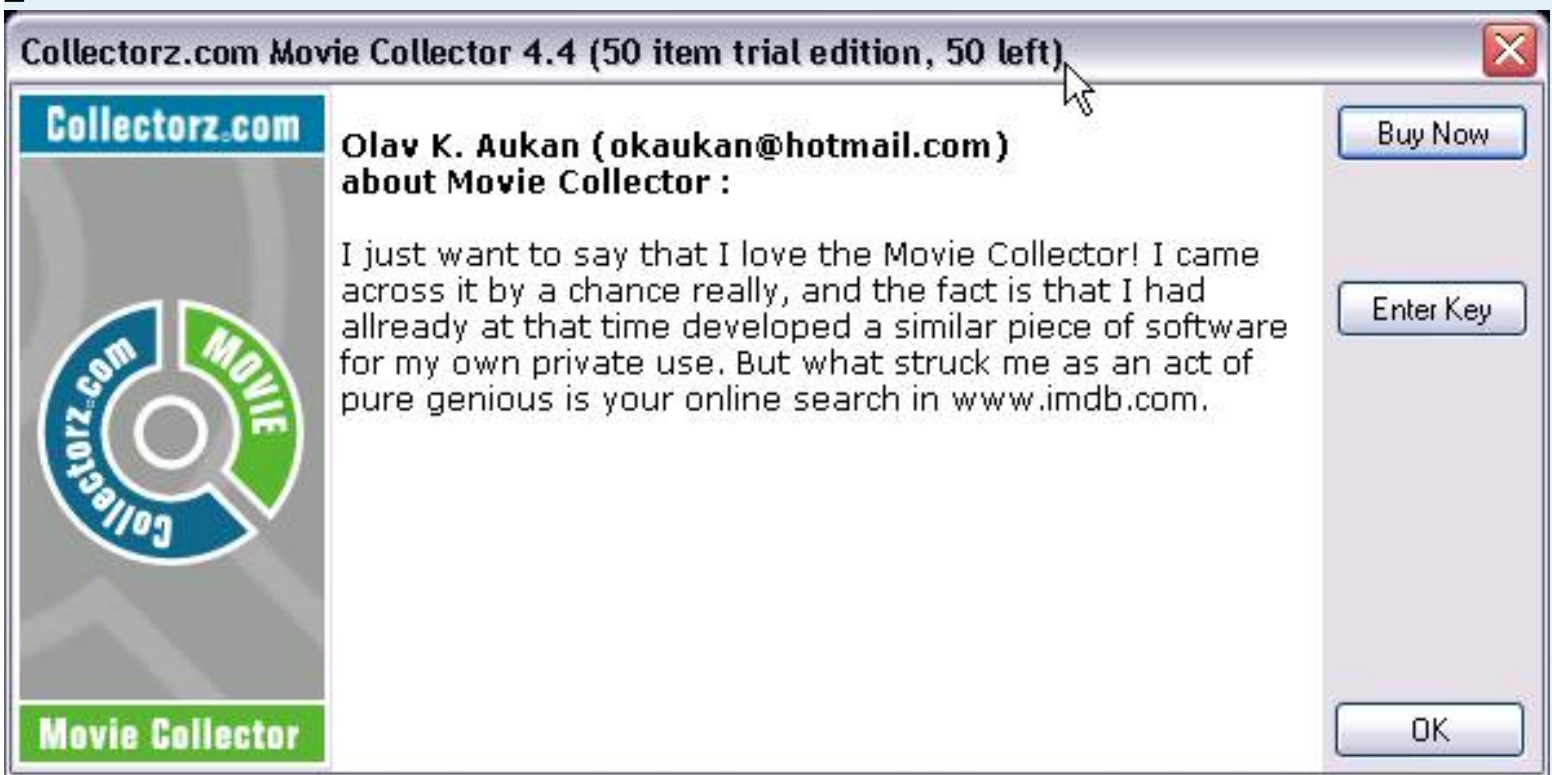


run try running lickerish. Done CopymemII and now to the main **Fix Nanomites**. Close all windows Olly. dumped.exe Load File to Olly.

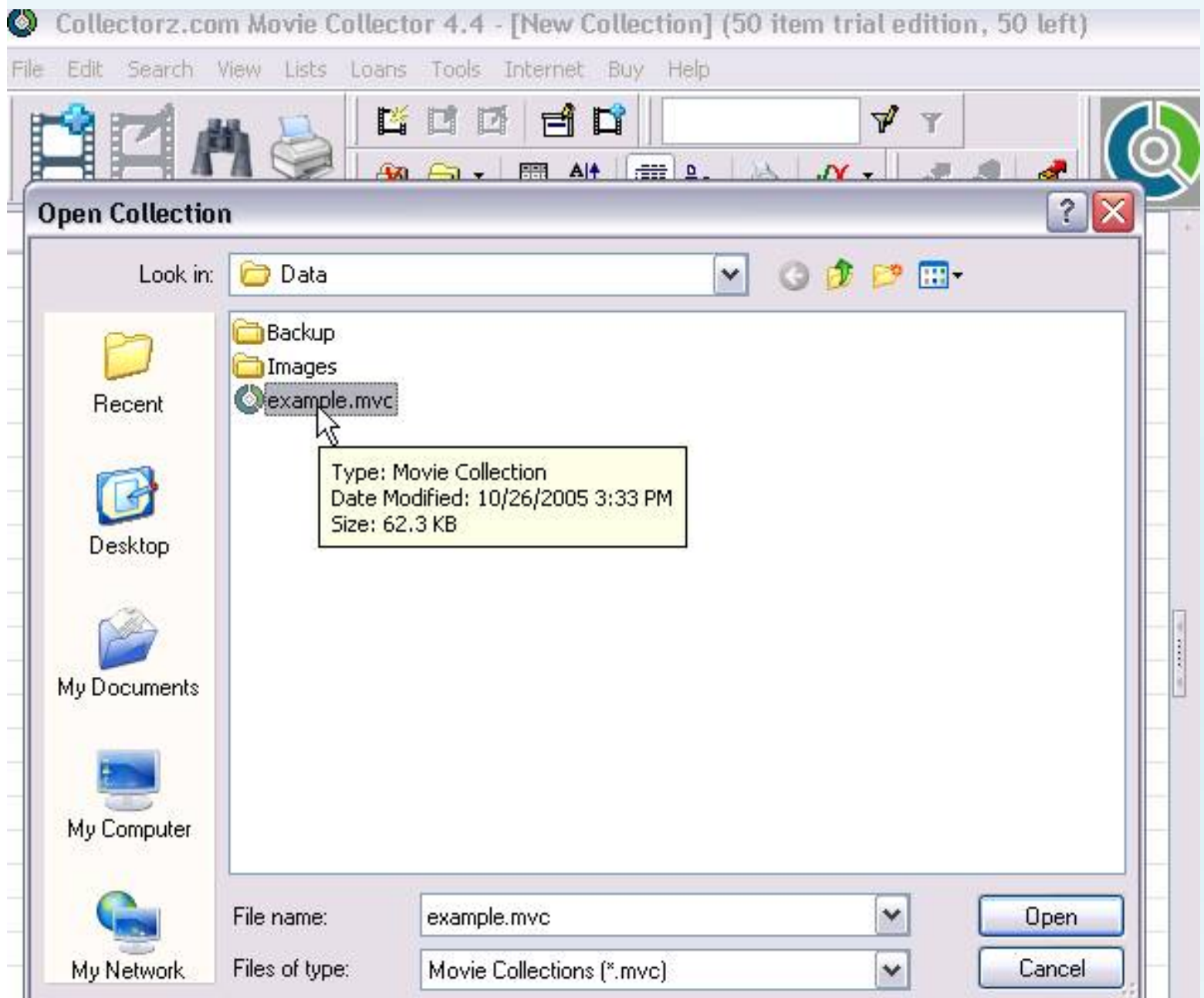
006890EC	55	PUSH EBP	
006890ED	8BEC	MOV EBP,ESP	
006890EF	83C4 DC	ADD ESP,-24	
006890F2	33C0	XOR EAX,EAX	
006890F4	8945 EC	MOV DWORD PTR SS:[EBP-14],EAX	
006890F7	B8 04896800	MOV EAX,dumped_.00688904	
006890FC	E8 07E9D7FF	CALL dumped_.00407AD8	
00689101	33C0	XOR EAX,EAX	
00689103	55	PUSH EBP	
00689104	68 60926800	PUSH dumped_.00689260	
00689109	64:FF30	PUSH DWORD PTR FS:[EAX]	
0068910C	64:8920	MOV DWORD PTR FS:[EAX],ESP	
0068910F	E8 1C36DDFF	CALL dumped_.0045C730	
00689114	3D 47000400	CMP EAX,40047	
00689119	7D 78	JGE SHORT dumped_.00689193	
0068911B	6A 04	PUSH 4	
0068911D	68 6C926800	PUSH dumped_.0068926C	
00689122	8D45 EC	LEA EAX,DWORD PTR SS:[EBP-14]	
00689125	50	PUSH EAX	

ASCII "Collectorz.com Movie Collect

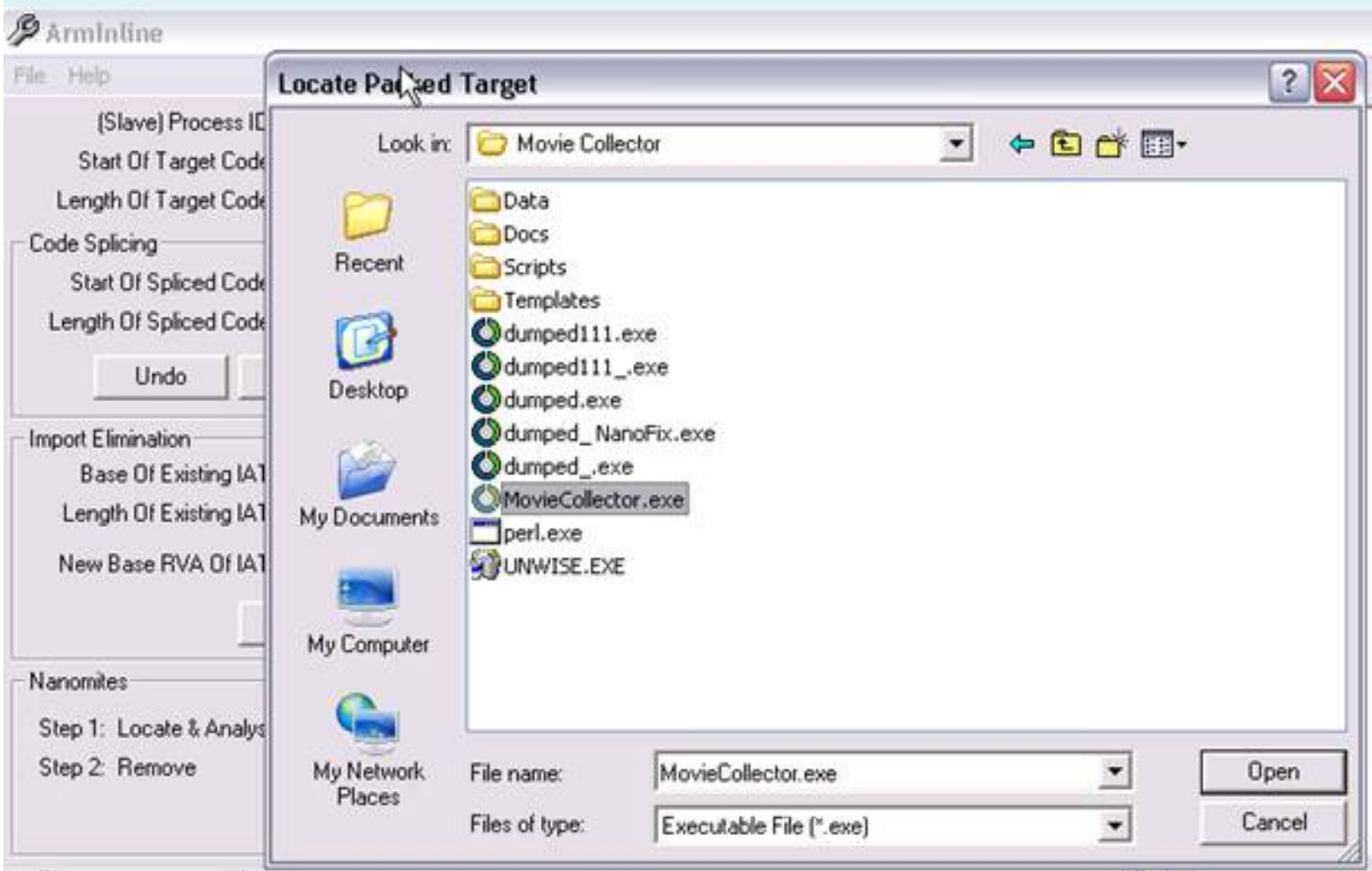
_Mo ArmInline 0.71 and enter the number and click the button  the NAG



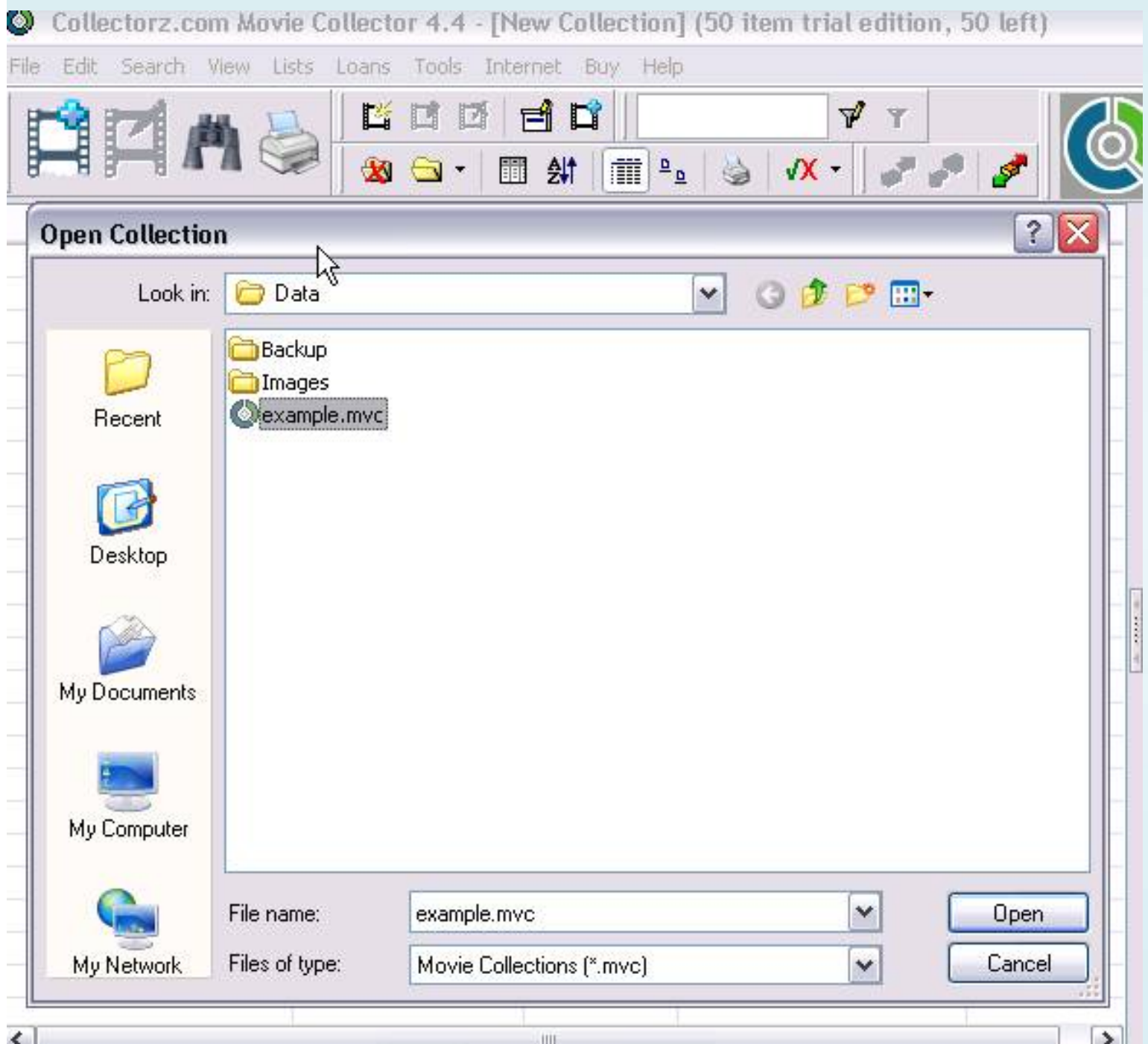
_Nhap OK, File -> Open and select the following:



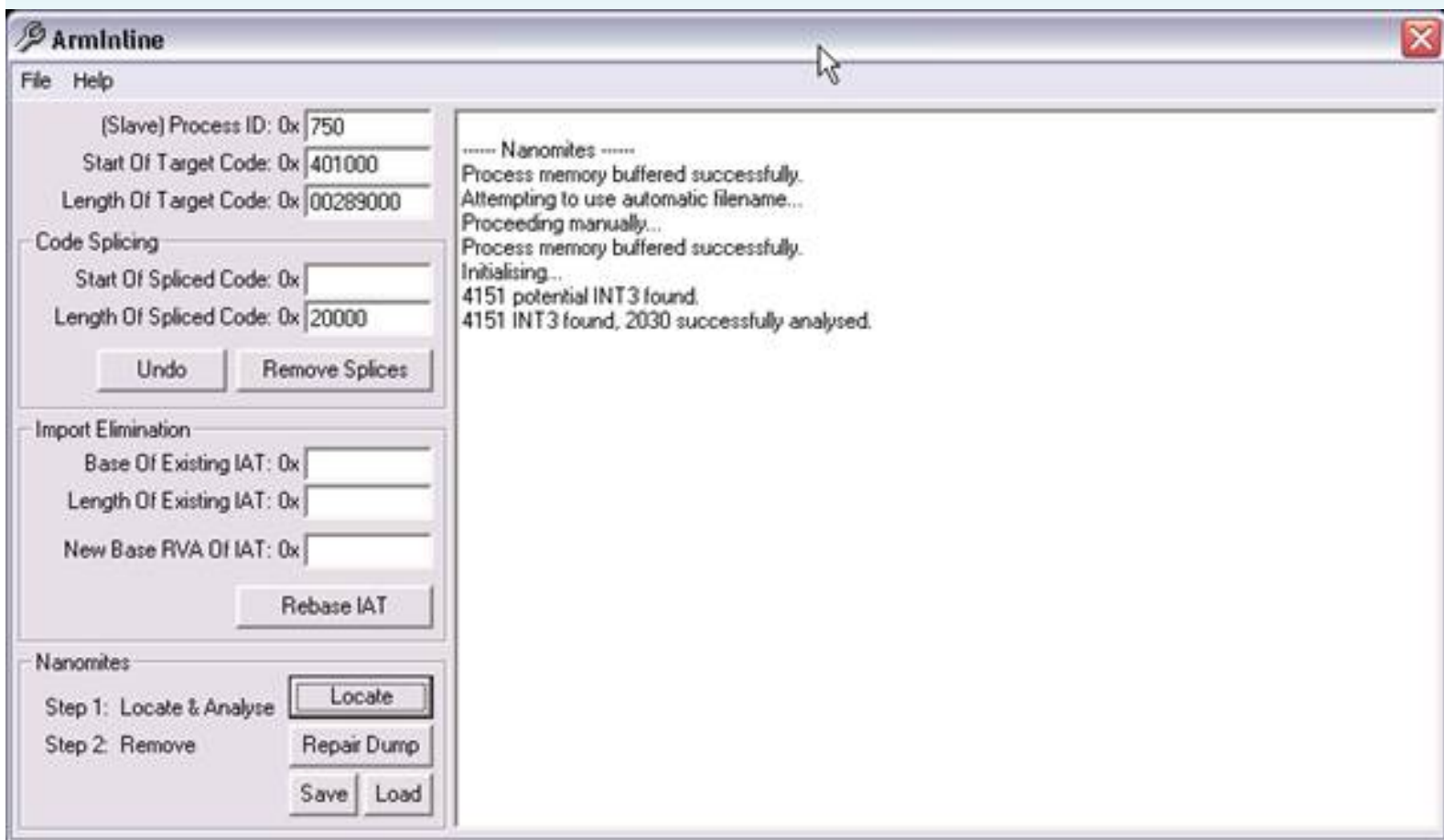
_Sau It, select MovieCollector.exe



1 _Lai does the Nag again, and OK Select File -> Open and select the following



_Doi Finished running you will see the following:



_Tiep By pressing buttons  and dumped_.exe

 ----- Removing Nanomites -----
Created C:\Program Files\Collectorz.com\Movie Collector\dumped_ NanoFix.exe
Adding new section header...
Appending new data...
Ammending PE header...
Done.

_Hehe That's it! Gently than when ArmTools03f.exe. As they use the tool is also provided unpack
Done is OK then, but you do it. Test, run good ==> unpack Done! !!!!!
CFF Explorer _Dung remove excess Section have new file storage 3.76Mb

Written by Why Not Bar

Armadillo 4.xx - Standard Protection

SoftWare : **My Screen Recorder Pro 2:22**

Copyright by: Copyright © 2005 DeskShare Incorporated. All rights reserved.

Download : [www.deskshare.com / download / msrp / msrp.exe](http://www.deskshare.com/download/msrp/msrp.exe)

Packed : ***Armadillo 4.xx - Standard Protection***

Language : Microsoft Visual C + + 7.0

Crack Too 1: *1.OllyDBG by hacnho.*

2. LordPE Deluxe 1.4-by yoda

3.Import REConstructor 1.6 Final

4th Diablo2oo2's Universal Patcher 2:10

Author : **Why Not Bar**

Features

Record your desktop screen activity to AVI, Windows Media Format (WMV) or Flash.

Generate and distribute self-contained and self-playable executables.

Create very small Flash and WMV recordings.

Make WMV recordings that are designed to be played from a streaming server.

Record the entire desktop, a desktop region or a specified window.

Create time lapse screen recordings.

Organize your screen recordings in easily accessible folder shortcuts.

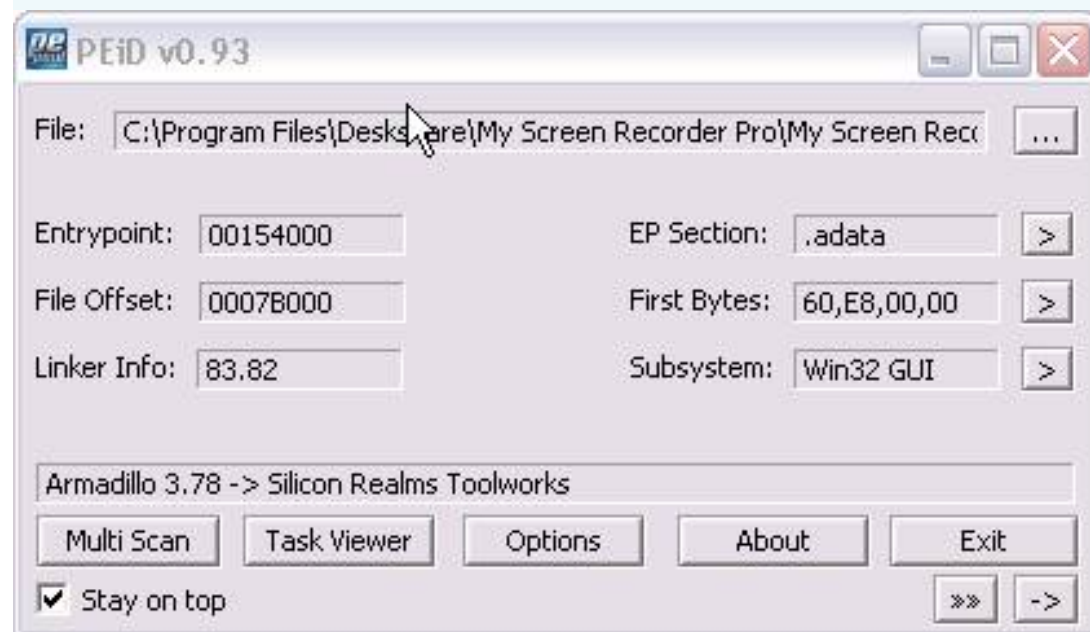
Fine-tune frame rate, audio quality, compressors and more.

Although the Hacnho is 1 of the series of professional thăng tut Arma from basic to advanced very detailed. But truly say that to read that the meaning of deep deep inside the series of tut Hacnho force you to have the 1 something to understand. If Newbie accept new chung must have the time, from the idea that children write a tut for Newbie. I say so because they also belong to the group Newbie perennial stop! As is Newbie should tut and writing tasks also extremely Newbie. I look forward to

the university in this area to bypass them.

I. Information:

_Dau First, we see Scan Soft What form this pack:



_Thay Pack with the *Armadillo*. At first, not read a series of positive or tut Hacnho, he met with Soft Pack *Armadillo* which is now run concern. Sợ ghê yet but now it seems much more confident and active search for it meat.

Soft _Run to find out some information to know which way to Crack and we are as follows:

At Nag reminder

There are the words unregistered

Click the Record 1 Nag reminder xài is only 30 days

The mission we are removing 2 Nag reminder, the words disappear unregistered

II - Cracking:

Soft In this we have 2 approaches to treat it as follows:

1. Do not unpack, we need to create 1 loader to patch it in Memory always. This is quite effective when we are not unpack.
2. Unpack and crack normal

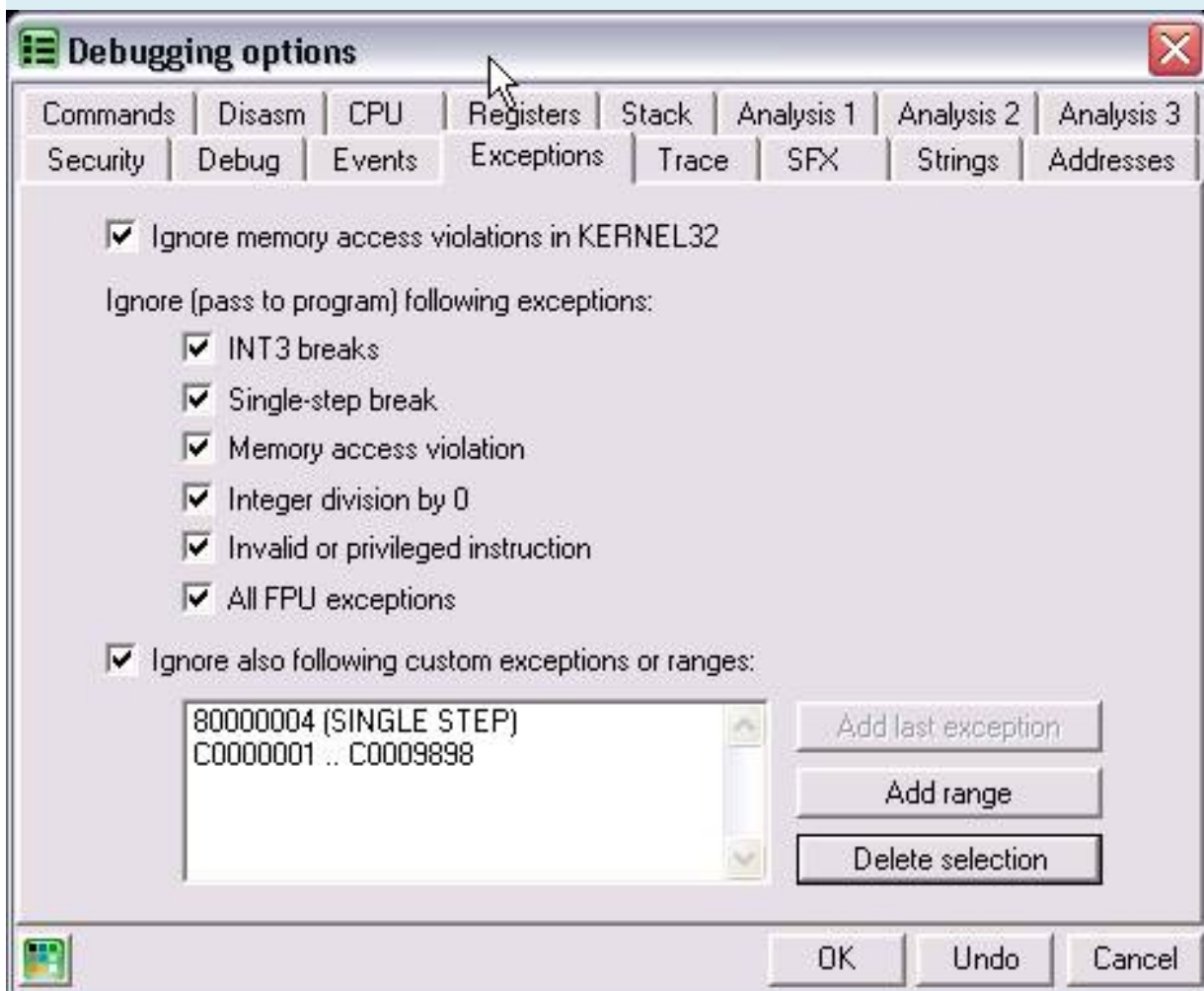
And they would look salt presented in 2 ways to Uncle new start looking more comprehensive.

1. Create Loader with DUP:

_ OK! gioLoad target is to olly

Address	Hex dump	Disassembly	Comment
00554000	60	PUSHAD	
00554001	E8 00000000	CALL My_Scree.00554006	
00554003	5D	POP EBP	
00554004	5D	PUSH EAX	
00554005	51	PUSH ECX	
00554006	0FCA	BSWAP EDX	
00554007	F7D2	NOT EDX	
00554008	9C	PUSHFD	
00554009	F7D2	NOT EDX	
0055400A	0FCA	BSWAP EDX	
0055400B	EB 0F	JMP SHORT My_Scree.00554023	
0055400C	B9 EB0FB8EB	MOV ECX,EBB80FEB	
0055400D	07	POP ES	Modification of segment register
0055400E	B9 EB0F90EB	MOV ECX,EB900FEB	
0055400F	08FD	OR CH,BH	
00554010	EB 0B	JMP SHORT My_Scree.0055402E	
00554011	F2:	PREFIX REPNE:	Superfluous prefix
00554012	EB F5	JMP SHORT My_Scree.0055401B	
00554013	EB F6	JMP SHORT My_Scree.0055401E	
00554014	F2:	PREFIX REPNE:	Superfluous prefix
00554015	EB 08	JMP SHORT My_Scree.00554033	
00554016	FD	STD	
00554017	EB E9	JMP SHORT My_Scree.00554017	
00554018	F3:	PREFIX REP:	Superfluous prefix
00554019	EB E4	JMP SHORT My_Scree.00554015	

_ You should be configured as follows:



_ Press Shift + F9, appear Nag reminder, you click Evaluate drilling jump.



_ Back to Olly press F12, Alt + M, select the address and click to select Show Call.

0012F3B0	77D493F5	Includes ntdll.KiFastSystemCallRet	USER32.77D493F3	0012F3E4
0012F3B4	77D6EA24	USER32.WaitMessage	USER32.77D6EA1F	0012F3E4
0012F3E8	77D5688A	USER32.77D6E895	USER32.77D56885	0012F3E4
0012F410	77D568CC	USER32.77D567D4	USER32.77D568C7	0012F40C
0012F430	77D5892D	USER32.DialogBoxIndirectParamAorW	USER32.77D58928	0012F42C
0012F45C	019BB0F1	USER32.DialogBoxParamA	019BB0EB	0012F458
0012F460	016B0000	hInst = 016B0000		
0012F464	00000065	pTemplate = 65		
0012F468	00000000	hOwner = NULL		
0012F46C	016B5CF0	DlgProc = DSRegMSR.016B5CF0		
0012F470	00000000	lParam = NULL		
0012F478	016B376F	Includes 019BB0F1	DSRegMSR.016B376D	0012F474
0012F498	016B3F3F	DSRegMSR.016B3740	DSRegMSR.016B3F3A	
0012F4A4	016B410C	DSRegMSR.016B3F20	DSRegMSR.016B4107	
0012F4C0	00405500	Includes DSRegMSR.016B410C	My_Scree.004054FD	
0012F564	0045134B	My_Scree.00405450	My_Scree.00451346	

Actualize

Hide arguments

Space

Follow address in stack

Show procedure

Enter

Show call

Execute to return

F4

Copy to clipboard

Appearance

_ I come:

Address	Hex dump	Disassembly	Comment
0045131F	8D4C24 1C	LEA ECX,DWORD PTR SS:[ESP+1C]	
00451329	C64424 4C 03	MOV BYTE PTR SS:[ESP+4C],3	
00451328	E8 631FFBFF	CALL My_Scree.00403290	
0045132D	8D4C24 0C	LEA ECX,DWORD PTR SS:[ESP+C]	
00451331	C64424 48 02	MOV BYTE PTR SS:[ESP+48],2	
00451336	E8 E506FBFF	CALL My_Scree.00401A20	
00451338	6A 04	PUSH 4	
0045133D	6A 00	PUSH 0	
0045133F	E8 BC3DFBFF	CALL My_Scree.00405100	
00451344	8BC8	MOV ECX,EAX	
00451346	E8 0541FBFF	CALL My_Scree.00405450	
00451348	85C0	TEST EAX,EAX	
0045134D	75 0B	JNZ SHORT My_Scree.0045135A	
0045134F	8935 74DF4F00	MOV DWORD PTR DS:[4FDF74],ESI	
00451355	A2 54AD4F00	MOV BYTE PTR DS:[4FAD54],AL	
0045135A	68 C8104D00	PUSH My_Scree.004D10C8	ASCII "hhctrl.ocx"
0045135F	FF15 44234C00	CALL NEAR DWORD PTR DS:[4C2344]	
00451365	68 480B0000	PUSH 0B48	
0045136A	A3 70DF4F00	MOV DWORD PTR DS:[4FDF70],EAX	
0045136F	E8 A39E0400	CALL My_Scree.0049B217	
00451374	83C4 04	ADD ESP,4	
00451377	894424 0C	MOV DWORD PTR SS:[ESP+C],EAX	
0045137B	85C0	TEST EAX,EAX	
0045137D	C64424 48 04	MOV BYTE PTR SS:[ESP+48],4	
00451382	74 0F	JE SHORT My_Scree.00451393	

_ You Set Hardware Breakpoint 1 at 00451346



_Ctrl + F2, Shift + F9, to the address we've set Breakpoint, press F7 to the function of this Call. The goal is to jump to address the Nag and patch it.

Address	Hex dump	Disassembly	Comment
00405450	81EC 0C000000	SUB ESP, 0C	
00405456	A1 A0C54F00	MOV EAX, DWORD PTR DS:[4FC5A0]	
0040545B	57	PUSH EDI	
0040545C	8BF9	MOV EDI, ECX	
0040545E	898424 8C000000	MOV DWORD PTR SS:[ESP+8C], EAX	
00405465	8B47 08	MOV EAX, DWORD PTR DS:[EDI+8]	
00405468	85C0	TEST EAX, EAX	
0040546A	0F84 03010000	JE My_Scree.00405573	
00405470	56	PUSH ESI	
00405471	6A 00	PUSH 0	
00405473	E8 A8D10700	CALL My_Scree.00482620	
00405478	50	PUSH EAX	
00405479	E8 73D10700	CALL My_Scree.004825F1	
0040547E	83C4 08	ADD ESP, 8	
00405481	33F6	XOR ESI, ESI	
00405483	E8 76D10700	CALL My_Scree.004825FE	
00405488	99	CDD	
00405489	B9 FF000000	MOV ECX, 0FF	
0040548E	F7F9	IDIV ECX	
00405490	46	INC ESI	
00405491	83FE 40	CMP ESI, 40	
00405494	8B5434 07	MOV BYTE PTR SS:[ESP+ESI+7], DL	
00405498	7C E9	JL SHORT My_Scree.00405483	
0040549A	C64434 08 00	MOV BYTE PTR SS:[ESP+ESI+8], 0	
0040549E	33C0	XOR EAX, EAX	

_Nhin The code Make sure you know you have to do it ha! Meaning the code as follows

00405468 85C0 TEST EAX, EAX <== Check have not registered

0040546A 0F84 03010000 JE My_Scree.00405573 <== down code inspection and springiness Nag if not registered.

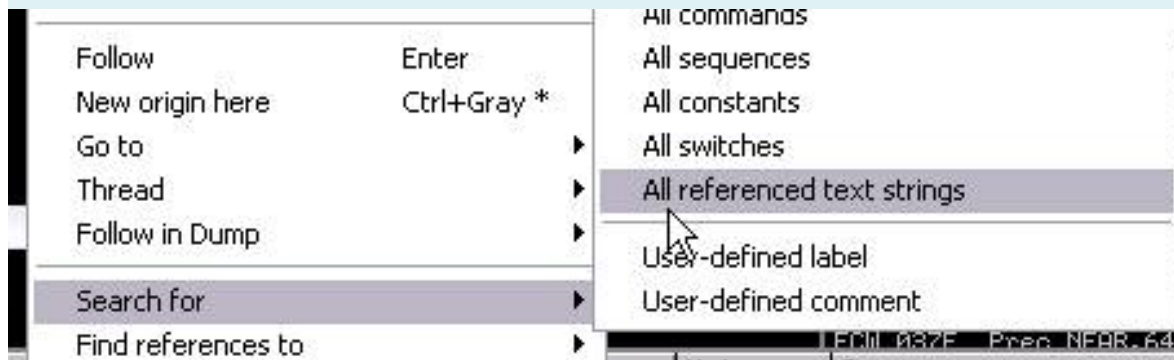
To jump from the Nag prompt and very simple one press Ctrl + E to:

0F84 to 0F85

Once the patch is as follows:

0040546A / 0F85 03010000 JNZ My_Scree.00405573

_Click Mouse to select the image below



_go chains are looking for *"unregistered"* and you find the series as follows:

004127E8 68 E06F4C00 PUSH My_Scree.004C6FE0; ASCII "My Screen Recorder Pro - unregistered"

00418E95 68 E06F4C00 PUSH My_Scree.004C6FE0; ASCII "My Screen Recorder Pro - unregistered"

00422F88 68 A4974C00 PUSH My_Scree.004C97A4; ASCII "unregistered - Trial Version"

0045151C 68 B8104D00 PUSH My_Scree.004D10B8; ASCII "- unregistered"

_Dia Only red line is the first we need. Make sure you ask more? Why do you know exactly the same. Nothing subliminal all, Try a few other Dek is only **0045151C** is OK. Double Click on one **0045151C** here

Address	Hex dump	Disassembly	Comment
004514EA	E8 D7910400	CALL My_Scree.0049A6C6	
004514EF	8B4E 1C	MOV ECX,DWORD PTR DS:[ESI+1C]	
004514F2	51	PUSH ECX	
004514F3	FF15 C8254C00	CALL NEAR DWORD PTR DS:[4C25C8]	USER32.UpdateWindow
004514F9	A1 74DF4F00	MOV EAX,DWORD PTR DS:[4FDF74]	
004514FE	85C0	TEST EAX,EAX	
00451500	75 30	JNZ SHORT My_Scree.0045153F	
00451502	8D4C24 0C	LEA ECX,DWORD PTR SS:[ESP+C]	
00451506	E8 4507FBFF	CALL My_Scree.00401C50	
00451508	8D5424 0C	LEA EDX,DWORD PTR SS:[ESP+C]	
0045150F	52	PUSH EDX	
00451510	8BCE	MOV ECX,ESI	
00451512	C64424 4C 07	MOV BYTE PTR SS:[ESP+4C],7	
00451517	E8 19790400	CALL My_Scree.00498E35	
0045151C	68 B8104D00	PUSH My_Scree.004D10B8	ASCII " - UNREGISTERED"
00451521	8D4C24 10	LEA ECX,DWORD PTR SS:[ESP+10]	
00451525	E8 2664FBFF	CALL My_Scree.00407950	
0045152A	8B4424 0C	MOV EAX,DWORD PTR SS:[ESP+C]	
0045152E	50	PUSH EAX	
0045152F	8BCE	MOV ECX,ESI	
00451531	E8 F3900400	CALL My_Scree.0049A629	
00451536	8D4C24 0C	LEA ECX,DWORD PTR SS:[ESP+C]	
0045153A	E8 E104FBFF	CALL My_Scree.00401A20	
0045153F	8B4424 10	MOV EAX,DWORD PTR SS:[ESP+10]	
00451543	83C0 F0	ADD EAX,-10	

_Cac Do not you see? On the orders it has 1 kìa dance! Code on the understanding you can simply

considered TEST you have not registered, if not to insert the text "- unregistered." You jump over it for not implementing code insert Text is OK.

00451500 / 75 3D JNZ SHORT My_Scree.0045153F

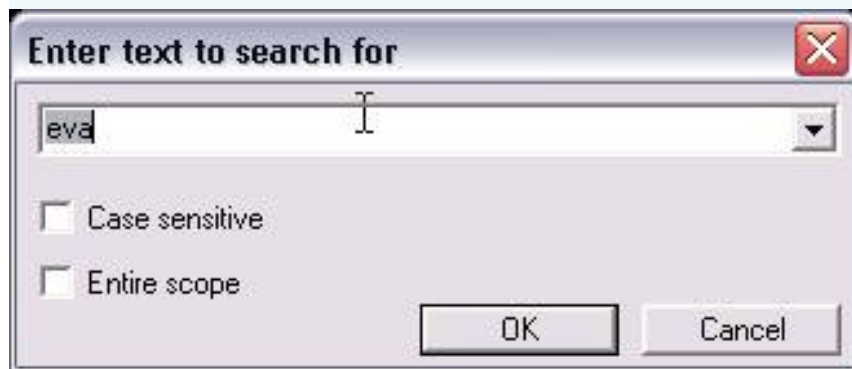
Patch's

00451500 / 74 3D JE SHORT My_Scree.0045153F

_Nhung Analysis at the beginning we have a 1 Nag reminder when more pressing Record



_Alt + R to back window and type Text Search



_Nhan OK to me here

Address	Hex dump	Disassembly	Comment
0041490C	C68424 1C820000	MOV BYTE PTR SS:[ESP+21C],8	
00414914	E8 77D7FEFF	JMP My_Scree.00402090	
00414919	808C24 88000000	LEA ECX,DWORD PTR SS:[ESP+88]	
00414920	C68424 18020000	MOV BYTE PTR SS:[ESP+218],3	
00414928	E8 03EC8000	JMP My_Scree.00423600	
00414920	C68424 EC010000	MOV BYTE PTR SS:[ESP+1EC],9	
00414935	E8 B68DFFFF	JMP My_Scree.004006F0	
0041493A	391D 740F4F00	CMPL DWORD PTR DS:[4F0F74],EBX	
00414940	75 0C	JNE SHORT My_Scree.0041494E	
00414942	53	PUSH EBX	
00414943	53	PUSH EBX	
00414944	68 F8714C00	CALL My_Scree.004C71F8	ASCII "This is an evaluation copy. All recorded videos will be
00414949	E8 1ACD8800	JMP My_Scree.004A1668	
0041494E	804C24 5C	LEA ECX,DWORD PTR SS:[ESP+5C]	
00414952	E8 3C740800	JMP My_Scree.00496D98	
00414957	83F8 02	CMPL EAX,2	
0041495A	75 4C	JNE SHORT My_Scree.004149A8	
0041495C	C705 04874F00	MOV DWORD PTR DS:[4F8704],1	
00414966	E8 35C2FFFF	JMP My_Scree.00410BA0	
00414968	804C24 5C	LEA ECX,DWORD PTR SS:[ESP+5C]	
0041496F	891D 3CDE4F00	MOV DWORD PTR DS:[4FDE3C],EBX	
00414975	891D E8DA4F00	MOV DWORD PTR DS:[4FDE81],EBX	
00414978	C68424 EC010000	MOV BYTE PTR SS:[ESP+1EC],3	
00414983	E8 18C4FFFF	JMP My_Scree.00410DA0	
00414988	804C24 1C	LEA ECX,DWORD PTR SS:[ESP+1C]	

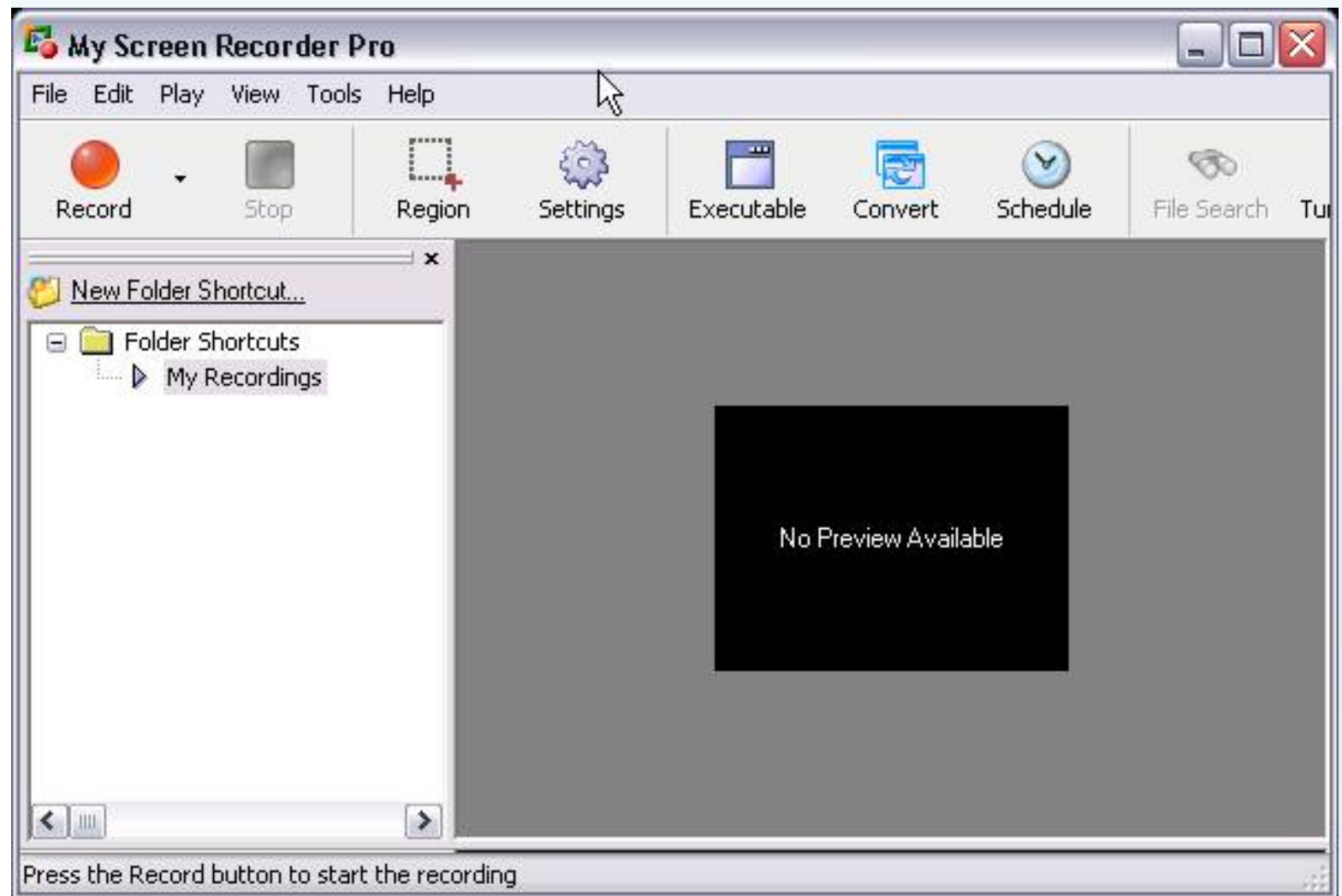
_He He! Above it has orders jumped reference! This code does the comparison you are not registered, if not jump NAG reminder. We need to do is to understand it as we have subscribed to from springiness NAG reminder. Say the long lines, but the implementation is as follows:

00414940 / 75 0C JNZ SHORT My_Scree.0041494E

Patch's

00414940 / 74 0C JE SHORT My_Scree.0041494E

_Toi You press F9 a new view. Everything has been resolved




_Bay Time we need to gather to address when we emerged to fill Patch DUP to create Loader as follows:

0040546A / 0F85 03010000 JNZ My_Scree.00405573

00451500 / 74 3D JE SHORT My_Scree.0045153F

00414940 / 74 0C JE SHORT My_Scree.0041494E

DUP _Chay 2.1 (with the tut) and press 



Nhap Button and click Save button  name and optional



_Chay Try Loader



_Gi Up any one! Calm review considered!. We note the report and reviewed the code does Nag first time we Patch

00405468 85C0 TEST EAX, EAX <== Check have not registered

0040546A 0F84 03010000 JE My_Scree.00405573 <== down code inspection and springiness Nag if not registered.

To jump from the Nag prompt and very simple one press Ctrl + E to:

0F84 to 0F85

Once the patch is as follows:

0040546A / 0F85 03010000 JNZ My_Scree.00405573

So that is when we Patch 0F84 to 0F85 it does not jump down to check the code and NAG but jumped to 1 NAG error message. So we just treat thẳng NAG this error message is more complete.

_ Load Soft and Olly to Shift + F9, one to address Set Breakpoint, press F7 to the function of this Call. 0F84 to 0F85 Patch, F8 Using Trace and it jumps to

```

00405570 81C4 8C000000 ADD ESP,8C
00405572 C2 0000 RETN 8
00405573 8B8C24 8C000000 MOV ECX,DWORD PTR SS:[ESP+0C]
0040557A B8 04000000 MOV EAX,4
0040557F 5F POP EDI
00405580 E8 998C0700 CALL My_Scree.0048121E
00405585 81C4 8C000000 ADD ESP,8C
00405588 C2 0000 RETN 8
0040558E CC INT3
0040558F CC INT3
00405590 8BC1 MOV EAX,ECX
00405592 33C9 XOR ECX,ECX
00405594 8BD8 MOV EDX,EAX
00405596 8BDB MOV DWORD PTR DS:[EDX],ECX

```

F8 _Dung trace over it to **RETN 8**

```

0045128E 8BC8 MOV ECX,EAX
00451290 E8 BB41F8FF CALL My_Scree.00405450
00451295 74F8 04 JNZ SHORT My_Scree.004512A5
00451298 0B JNZ SHORT My_Scree.004512A5
0045129A 6A 00 PUSH 0
0045129C 6A 00 PUSH 0
0045129E 68 E0104D00 PUSH My_Scree.004D10E0
004512A3 EB A5 JEB SHORT My_Scree.004512A9
004512A5 8D4C24 34 LEA ECX,DWORD PTR SS:[ESP+34]
004512A9 E8 E2FDFFFF CALL My_Scree.00451090
004512AE C64424 48 01 MOV BYTE PTR SS:[ESP+48],1
004512B3 E8 6CA00400 CALL My_Scree.0049B324
004512B8 8B10 MOV EDX,DWORD PTR DS:[EAX]
004512BA 8BC8 MOV ECX,EAX
004512BC FF52 0C CALL NEAR DWORD PTR DS:[EDX+C]
004512BF 83C8 10 ADD EBX,10

```

_ Is the Text that nay notice when we receive. Authors software quite wise when it set the NAG here. Make sure he understands this, he is also our patch 0F84 to 0F85 should set this to the NAG doa we damaged that it! But not in the stars Male He has asked her "quit fruits have thick nails nhện." You treat it only

00451298 / 75 0B JNZ SHORT dumped_.004512A5

Patch's

00451298 / 74 0B JE SHORT dumped_.004512A5

_ So we need to fill more 00451298 DUP as follows

3. Unpack and Crack:

_ OK! gioLoad target is to olly

Address	Hex dump	Disassembly	Comment
00554000	60	PUSHAD	
00554001	E8 00000000	CALL My_Scree.00554006	
00554003	5D	POP EBP	
00554004	50	PUSH EAX	
00554005	51	PUSH ECX	
00554006	0FCA	BSWAP EDX	
00554007	F7D2	NOT EDX	
00554008	9C	PUSHFD	
00554009	F7D2	NOT EDX	
0055400A	0FCA	BSWAP EDX	
0055400B	EB 0F	JNE SHORT My_Scree.00554023	
0055400C	B9 EB0FB8EB	MOV ECX,EBB80FEB	
0055400D	07	POP ES	Modification of segment register
0055400E	B9 EB0F90EB	MOV ECX,EB900FEB	
0055400F	08FD	OR CH,BH	
00554010	EB 0B	JNE SHORT My_Scree.0055402E	
00554011	F2:	PREFIX REPNE:	Superfluous prefix
00554012	EB F5	JNE SHORT My_Scree.0055401B	
00554013	EB F6	JNE SHORT My_Scree.0055401E	
00554014	F2:	PREFIX REPNE:	Superfluous prefix
00554015	EB 08	JNE SHORT My_Scree.00554033	
00554016	FD	STD	
00554017	EB E9	JNE SHORT My_Scree.00554017	
00554018	F3:	PREFIX REP:	Superfluous prefix
00554019	EB E4	JNE SHORT My_Scree.00554015	

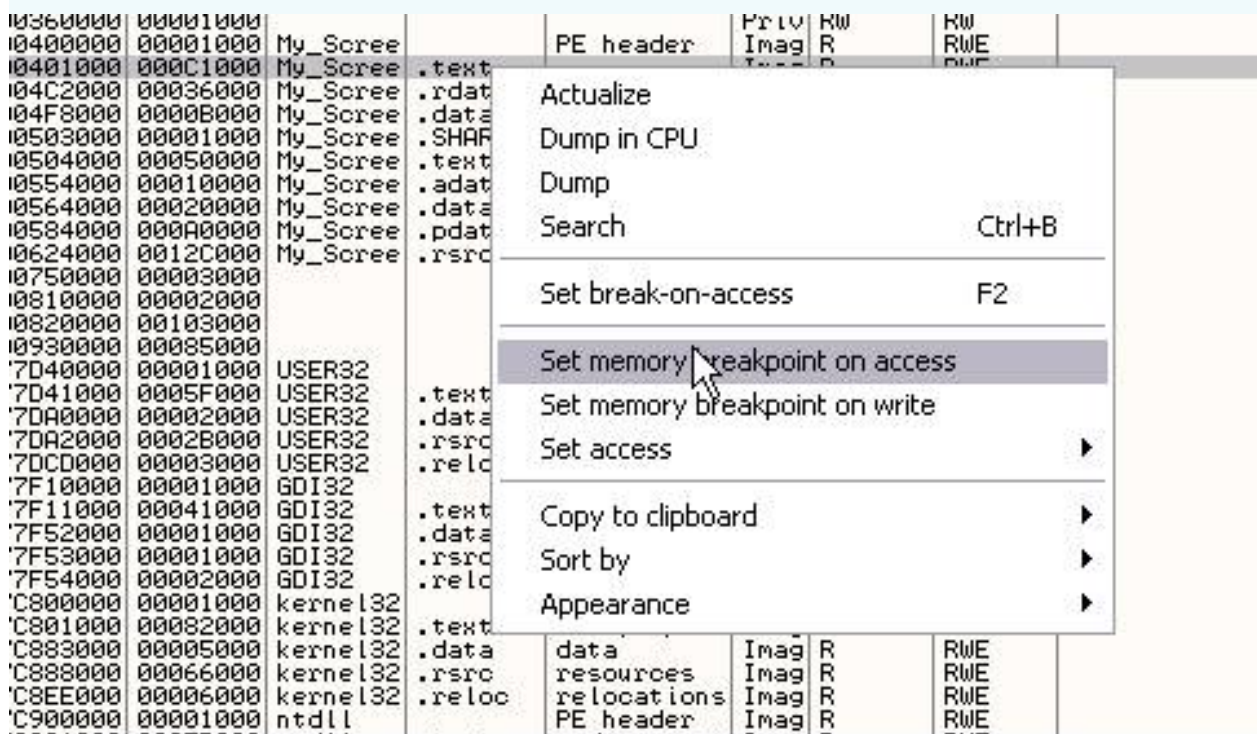
_Alt + F1, GetModuleHandleA He, Shifl 3 + F9, Alt + F9 to you here.

Address	Hex dump	Disassembly	Comment
003A5B47	8B00 6C503D00	MOV ECX,DWORD PTR DS:[3D506C]	
003A5B48	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003A5B49	A1 6C503D00	MOV EAX,DWORD PTR DS:[3D506C]	
003A5B4A	391C06	CMPL DWORD PTR DS:[ESI+EAX],EBX	
003A5B4B	75 16	JNE SHORT 003A5B70	
003A5B4C	8085 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
003A5B4D	50	PUSH EAX	
003A5B4E	FF15 B9622C00	CALL NEAR DWORD PTR DS:[2C62B8]	kernel32.LoadLibraryA
003A5B4F	8B00 6C503D00	MOV ECX,DWORD PTR DS:[3D506C]	
003A5B50	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003A5B51	A1 6C503D00	MOV EAX,DWORD PTR DS:[3D506C]	
003A5B52	391C06	CMPL DWORD PTR DS:[ESI+EAX],EBX	
003A5B53	0F84 2F010000	JNE 003A5C0D	
003A5B54	33C9	XOR ECX,ECX	
003A5B55	8B07	MOV EAX,DWORD PTR DS:[EDI]	
003A5B56	3918	CMPL DWORD PTR DS:[EAX],EBX	
003A5B57	74 06	JNE SHORT 003A5B8C	
003A5B58	41	INC ECX	
003A5B59	83C0 0C	ADD EAX,0C	
003A5B5A	EB F6	JNE SHORT 003A5B82	
003A5B5B	8B09	MOV EBX,ECX	
003A5B5C	C1E3 02	SHL EBX,2	

_Patch Magic Jump

Address	Hex dump	Disassembly	Comment
003A5B47	8B00 6C503D00	MOV ECX,DWORD PTR DS:[3D506C]	
003A5B48	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003A5B49	A1 6C503D00	MOV EAX,DWORD PTR DS:[3D506C]	
003A5B4A	391C06	CMPL DWORD PTR DS:[ESI+EAX],EBX	
003A5B4B	75 16	JNE SHORT 003A5B70	
003A5B4C	8085 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
003A5B4D	50	PUSH EAX	
003A5B4E	FF15 B9622C00	CALL NEAR DWORD PTR DS:[2C62B8]	kernel32.LoadLibraryA
003A5B4F	8B00 6C503D00	MOV ECX,DWORD PTR DS:[3D506C]	
003A5B50	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003A5B51	A1 6C503D00	MOV EAX,DWORD PTR DS:[3D506C]	
003A5B52	391C06	CMPL DWORD PTR DS:[ESI+EAX],EBX	
003A5B53	0F84 2F010000	JNE 003A5C0D	
003A5B54	33C9	XOR ECX,ECX	
003A5B55	8B07	MOV EAX,DWORD PTR DS:[EDI]	
003A5B56	3918	CMPL DWORD PTR DS:[EAX],EBX	
003A5B57	74 06	JNE SHORT 003A5B8C	
003A5B58	41	INC ECX	
003A5B59	83C0 0C	ADD EAX,0C	
003A5B5A	EB F6	JNE SHORT 003A5B82	
003A5B5B	8B09	MOV EBX,ECX	

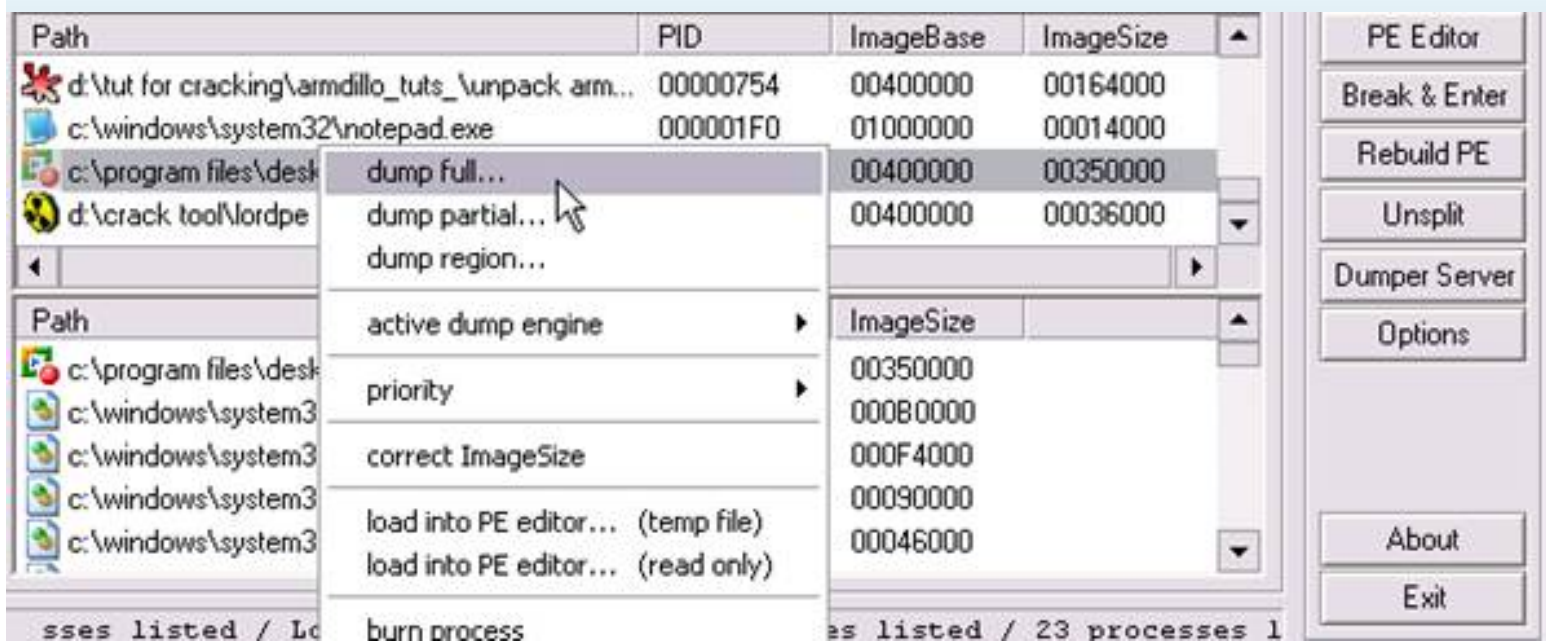
_Vi Copy must Patch Magic jump, just as we do so to IAT has been completed to the dump run Fix lickerish not crash. Delete Breakpoint GetModuleHandleA (HeGetModuleHandleA) and press Alt + M or click the "M" on the interface and Memory Set Breakpoint on access in Section. Text (like below).



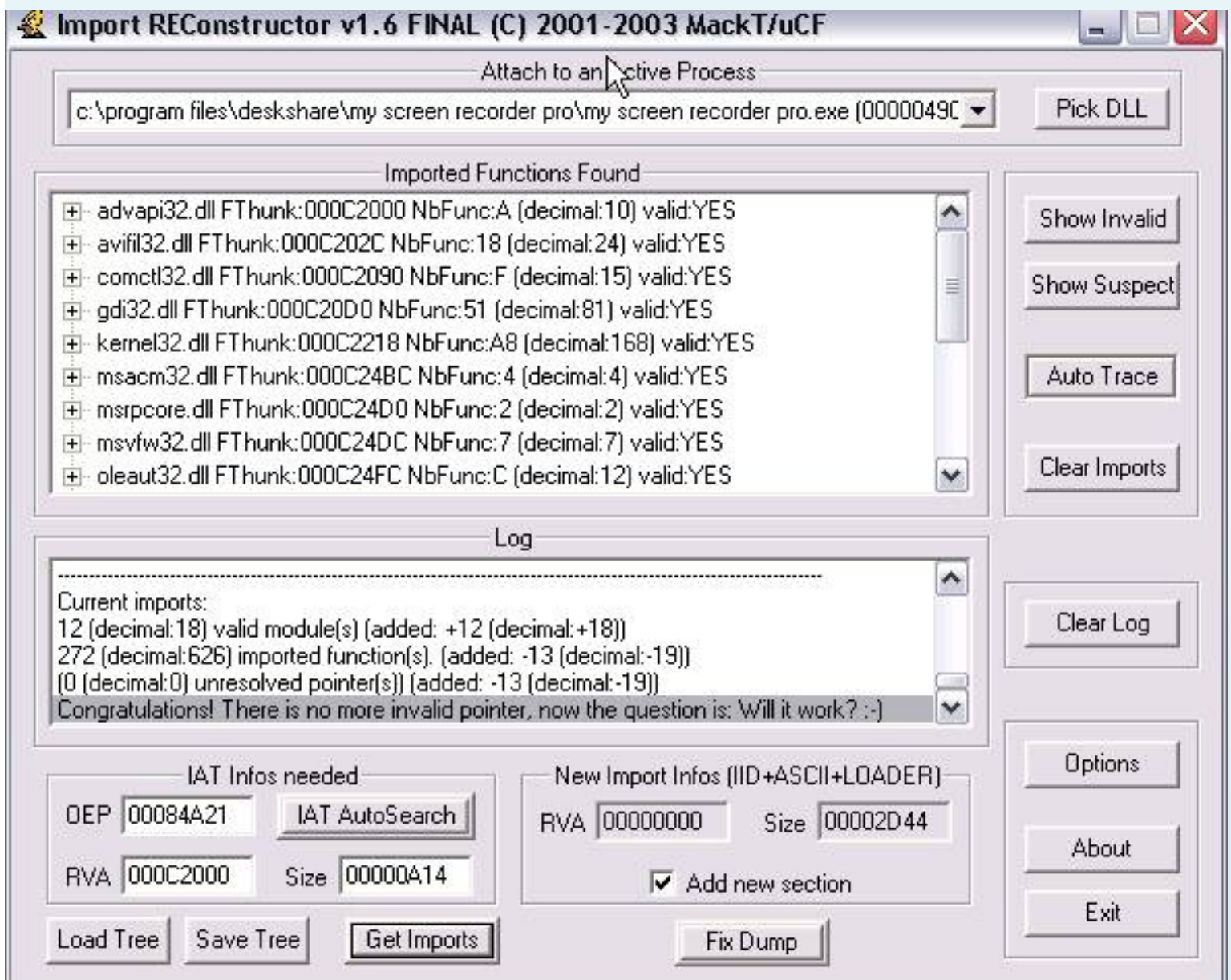
Press Shift + F9 to OEP:

Address	Hex dump	Disassembly	Comment
00484A21	6A 60	JMP SHORT My_Scree.00484A21	← OEP
00484A23	68 80F44D00	PUSH My_Scree.004DF480	
00484A28	E8 03070000	CALL My_Scree.00485130	
00484A2D	BF 94000000	MOV EDI, 94	
00484A32	8BC7	MOV EAX, EDI	
00484A34	E8 77C7FFFF	CALL My_Scree.004811B0	
00484A39	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
00484A3C	8BF4	MOV ESI, ESP	
00484A3E	893E	MOV DWORD PTR DS:[ESI], EDI	
00484A40	56	PUSH ESI	
00484A41	FF15 04234C00	CALL NEAR DWORD PTR DS:[4C2304]	kernel32.GetVersionExA
00484A47	8B4E 10	MOV ECX, DWORD PTR DS:[ESI+10]	
00484A4A	898D 84075000	MOV DWORD PTR DS:[500784], ECX	
00484A50	8B46 04	MOV EAX, DWORD PTR DS:[ESI+4]	
00484A53	A3 90075000	MOV DWORD PTR DS:[500790], EAX	
00484A58	8B56 08	MOV EDX, DWORD PTR DS:[ESI+8]	
00484A5B	8915 94075000	MOV DWORD PTR DS:[500794], EDX	
00484A61	8B76 0C	MOV ESI, DWORD PTR DS:[ESI+C]	
00484A64	81E6 FF7F0000	AND ESI, 7FFF	
00484A6A	8935 88075000	MOV DWORD PTR DS:[500788], ESI	
00484A70	83F9 02	CMPL ECX, 2	
00484A73	74 0C	JL SHORT My_Scree.00484A81	

Mo LordPE select Target Full dump:



_Mo ImportREC fill OEP: 84A21, click , Click , Click
Cut and Thunk

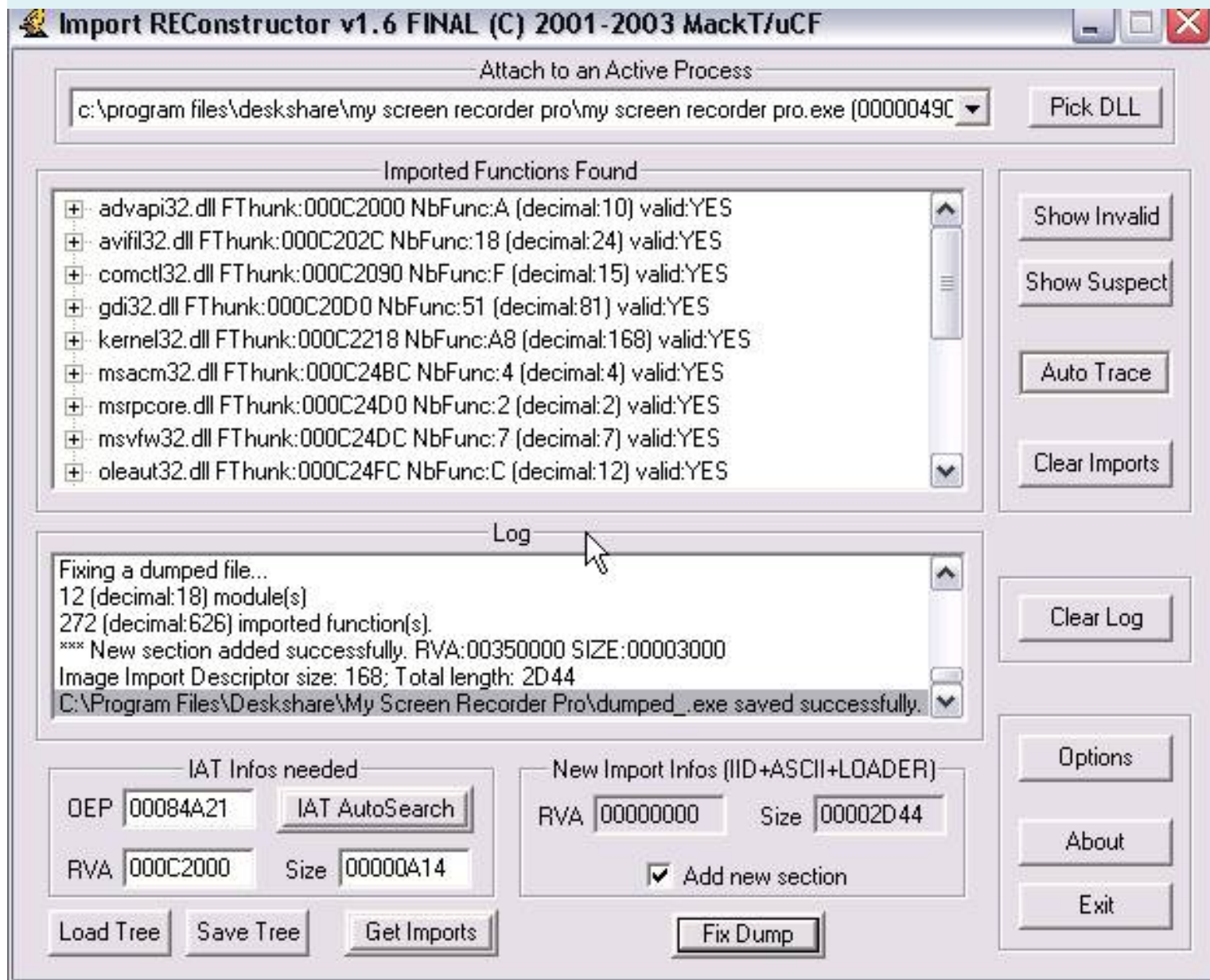


Load Tree

Save Tree

Get Imports

Fix Dump

Fix dump _nhan**Try dumped_.exe _Run file, do not run OK Crash ==> unpack Done****Crack _Viec the operation also nearly as we create as loader at first. Filedumped_.exe Load to Olly**

Address	Hex dump	Disassembly	Comment
00484A21	6A 60	JMP EB	
00484A23	68 80F44D00	PUSH dumped_.004DF480	
00484A28	E8 03070000	CALL dumped_.00485130	
00484A2D	BF 94000000	MOV EDI, 94	
00484A32	8BC7	MOV EBX, EDI	
00484A34	E8 77C7FFFF	CALL dumped_.004811B0	
00484A39	8965 E8	MOV DWORD PTR SS:[EBP-10], ESP	
00484A3C	8BF4	MOV ESI, ESP	
00484A3E	893E	MOV DWORD PTR DS:[ESI], EDI	
00484A40	56	PUSH ESI	
00484A41	FF15 04234C00	CALL NEAR DWORD PTR DS:[<kernel32.GetV kernel32.GetVersionExA	
00484A47	8B4E 10	MOV ECX, DWORD PTR DS:[ESI+10]	
00484A4A	990D 84075000	MOV DWORD PTR DS:[500784], ECX	
00484A50	8B46 04	MOV EBX, DWORD PTR DS:[ESI+4]	
00484A53	A3 90075000	MOV DWORD PTR DS:[500790], EBX	
00484A58	8B56 08	MOV EDX, DWORD PTR DS:[ESI+8]	
00484A5B	8915 94075000	MOV DWORD PTR DS:[500794], EDX	
00484A61	8B76 0C	MOV ESI, DWORD PTR DS:[ESI+C]	
00484A64	81E6 FF7F0000	AND ESI, 7FFF	
00484A6A	8935 88075000	MOV DWORD PTR DS:[500788], ESI	
00484A70	83F9 02	CMPL ECX, 2	
00484A73	74 0C	JNE SHORT dumped_.00484A81	

_ Press Shift + F9, appear Nag reminder, you click Evaluate drilling jump. Back to Olly press F12, Alt + M,

0012FC80	77D493F5	Includes ntdll.KiFastSystemCallRet	USER32.77D493F3	0012FCB4
0012FC84	77D6EA24	USER32.WaitMessage	USER32.77D6EA1F	0012FCB4
0012FCB8	77D5688A	USER32.77D6E895	USER32.77D56885	0012FCB4
0012FCE0	77D568CC	USER32.77D567D4	USER32.77D568C7	0012FCDC
0012FD00	77D5892D	USER32.DialogBoxIndirectParamAorW	USER32.77D58928	0012FCFC
0012FD2C	0109B0F1	USER32.DialogBoxParamA	0109B0EB	0012FD28
0012FD30	00C90000	hInst = 00C90000		
0012FD34	00000065	pTemplate = 65		
0012FD38	00000000	hOwner = NULL		
0012FD3C	00C95CF0	DlgProc = DSRegMSR.00C95CF0		
0012FD40	00000000	lParam = NULL		
0012FD48	00C9376F	Includes 0109B0F1	DSRegMSR.00C9376D	0012FD44
0012FD68	00C93F3F	DSRegMSR.00C93740	DSRegMSR.00C93F3A	
0012FD74	00C9410C	DSRegMSR.00C93F20	DSRegMSR.00C94107	
0012FD90	00405500	Includes DSRegMSR.00C9410C	dumped_.004054FD	
0012FE34	0045134B	dumped_.00405450	dumped_.00451346	

_ Selected address and click the mouse and double set Breakpoint like

0040544F	CC	INT3	
00405450	81EC 80000000	SUB EB	
00405456	A1 A0C54F00	MOV EA	
0040545B	57	PUSH E	
0040545C	8BF9	MOV ED	
0040545E	890424 8C000000	MOV DM	
00405465	8B47 08	MOV EA	
00405468	85C8	TEST E	
0040546A	0F84 03010000	JE dump	
00405470	56	PUSH E	
00405471	6A 08	PUSH 0	
00405473	E8 A8D10700	CALL d	
00405478	50	PUSH E	
00405479	E8 73D10700	CALL d	
0040547E	89C4 08	ADD ES	
00405481	33F6	XOR ES	
00405483	E8 76D10700	CALL d	
00405488	99	CDD	
00405489	B9 FF000000	MOV EC	
0040548E	F7F9	IDIV E	
00405490	46	INC ES	
00405491	83FE 40	CMPL ES	

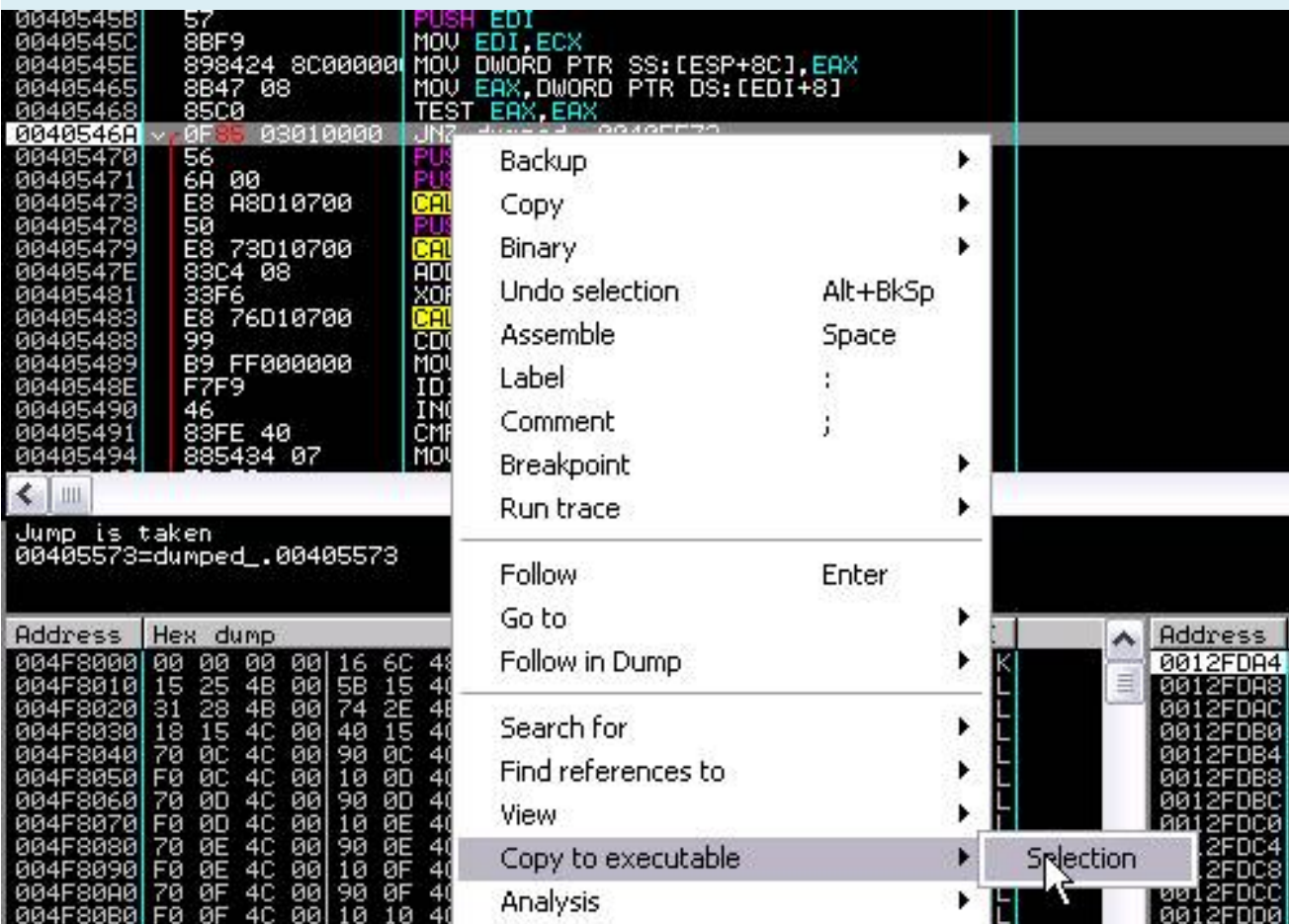
_ Ctrl + F2, Shift + F9, to the address we've set Breakpoint

Address	Hex dump	Disassembly	Comment
00405450	81EC 8C000000	SUB ESP,8C	
00405456	A1 A0C54F00	MOV EAX,DWORD PTR DS:[4FC5A0]	
00405458	57	PUSH EDI	
0040545C	8BF9	MOV EDI,ECX	
0040545E	890424 8C000000	MOV DWORD PTR SS:[ESP+8C],EAX	
00405465	8B47 08	MOV EAX,DWORD PTR DS:[EDI+8]	
00405468	85C0	TEST EAX,EAX	
0040546A	0F84 03010000	JE dumped_.00405573	
00405470	56	PUSH ESI	
00405471	6A 00	PUSH 0	
00405473	E8 A8D10700	JRNL dumped_.00482620	
00405478	50	PUSH EAX	
00405479	E8 73D10700	JRNL dumped_.004825F1	
0040547E	83C4 08	ADD ESP,8	
00405481	33F6	XOR ESI,ESI	
00405483	E8 76D10700	JRNL dumped_.004825FE	
00405488	99	CDQ	
00405489	B9 FF000000	MOV ECX,0FF	
0040548E	F7F9	IDIV ECK	
00405490	46	INC ESI	
00405491	83FE 40	CMF ESI,40	
00405494	8B5434 07	MOV BYTE PTR SS:[ESP+ESI+7],DL	

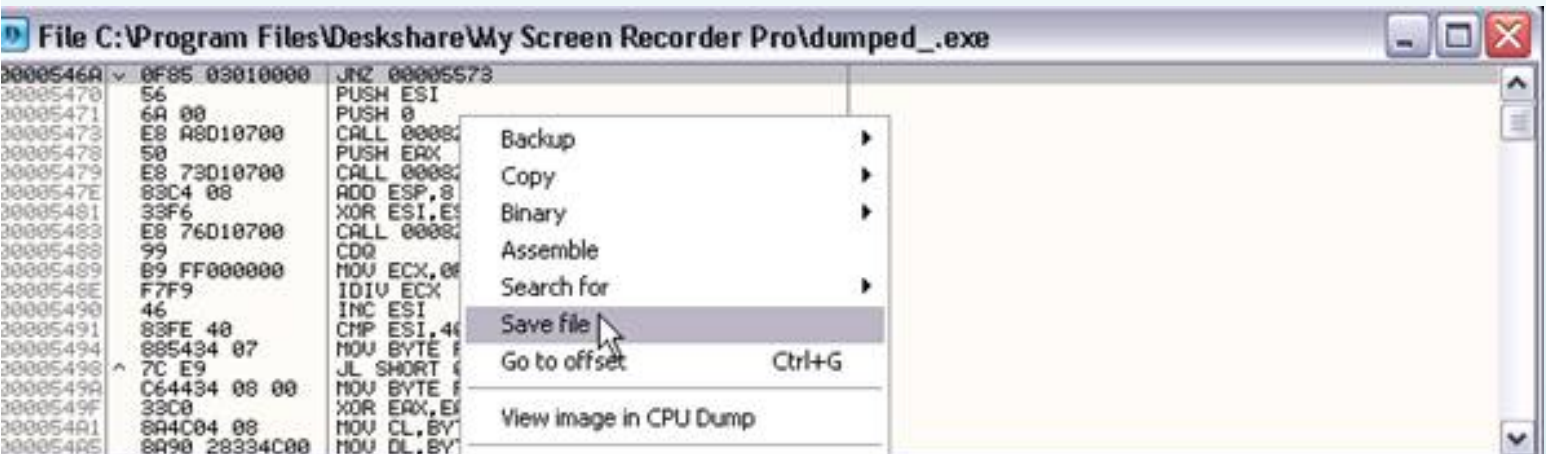
F8 to trace _Dung 0040546 and patch 0F84 to 0F85

Address	Hex dump	Disassembly	Comment
00405450	81EC 8C000000	SUB ESP,8C	
00405456	A1 A0C54F00	MOV EAX,DWORD PTR DS:[4FC5A0]	
00405458	57	PUSH EDI	
0040545C	8BF9	MOV EDI,ECX	
0040545E	890424 8C000000	MOV DWORD PTR SS:[ESP+8C],EAX	
00405465	8B47 08	MOV EAX,DWORD PTR DS:[EDI+8]	
00405468	85C0	TEST EAX,EAX	
0040546A	0F85 03010000	JNZ dumped_.00405573	
00405470	56	PUSH ESI	
00405471	6A 00	PUSH 0	
00405473	E8 A8D10700	JRNL dumped_.00482620	
00405478	50	PUSH EAX	
00405479	E8 73D10700	JRNL dumped_.004825F1	
0040547E	83C4 08	ADD ESP,8	
00405481	33F6	XOR ESI,ESI	
00405483	E8 76D10700	JRNL dumped_.004825FE	
00405488	99	CDQ	
00405489	B9 FF000000	MOV ECX,0FF	
0040548E	F7F9	IDIV ECK	
00405490	46	INC ESI	
00405491	83FE 40	CMF ESI,40	
00405494	8B5434 07	MOV BYTE PTR SS:[ESP+ESI+7],DL	

_nhap and mouse to do image



1 _Hien the window and click right and Save File



File Save _Chay new notice to appear



_Moi The same place as we analyze the Loader do so just Load File Save to Olly, Shift + F9, we set

up to address Patch and Breakpoint **0F84** to **0F85**, F8 Using Trace and it jumps to

```

0040556A 81C4 8C000000 ADD ESP,8C
00405570 C2 0000 RETN 8
00405573 8B8C24 8C000000 MOV ECX,DWORD PTR SS:[ESP+8C]
0040557A B8 04000000 MOV EAX,4
0040557F 5F POP EDI
00405580 E8 998C0700 CALL My_Scree.0048121E
00405585 81C4 8C000000 ADD ESP,8C
00405588 C2 0000 RETN 8
0040558E CC INT3
0040558F CC INT3
00405590 8BC1 MOV EAX,ECX
00405592 33C9 XOR ECX,ECX
00405594 8BD8 MOV EDX,EAX
00405596 8300 MOV DWORD PTR DS:[EDX],ECX

```

F8 _Dung trace over it to **RETN 8**

```

0045128E 8BC8 MOV ECX,EAX
00451290 E8 BB41F8FF CALL My_Scree.00405450
00451295 3BF8 04 CMP EAX,4
00451298 0B JNZ SHORT My_Scree.004512A5
0045129A 6A 00 PUSH 0
0045129C 6A 00 PUSH 0
0045129E 68 E0104D00 PUSH My_Scree.004D10E0
004512A3 EB A5 JEB SHORT My_Scree.004512A9
004512A5 8D4C24 34 LEA ECX,DWORD PTR SS:[ESP+34]
004512A9 E8 E2FDFFFF CALL My_Scree.00451090
004512AE C64424 48 01 MOV BYTE PTR SS:[ESP+48],1
004512B3 E8 6CA00400 CALL My_Scree.0049B324
004512B8 8B10 MOV EDX,DWORD PTR DS:[EAX]
004512BA 8BC8 MOV ECX,EAX
004512BC FF52 0C CALL NEAR DWORD PTR DS:[EDX+C]
004512BF 8300 MOV EBX,0

```

_Thit It to you

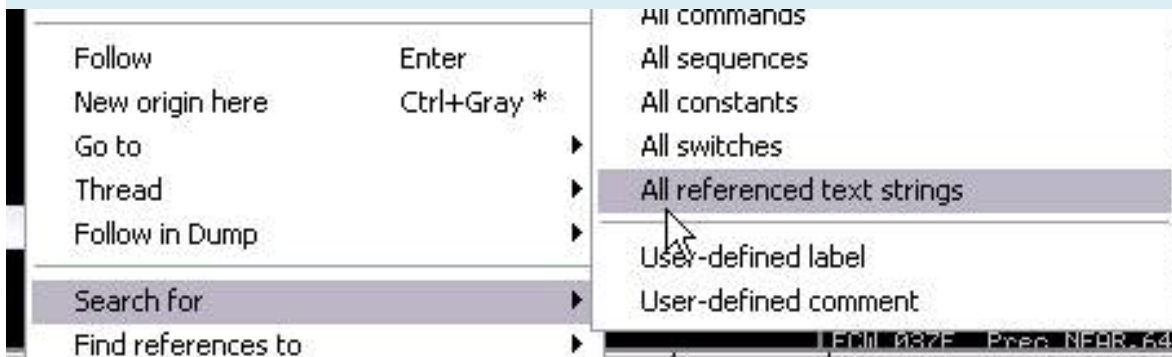
00451298 / 75 0B JNZ SHORT dumped_.004512A5

Patch's

00451298 / 74 0B JE SHORT dumped_.004512A5

_ Save the file and run the test, take the OK error message gòi

_ Now we treat more words on the "unregistered" Click to select the image below



_go chains are looking for **"unregistered"** and you find the series as follows:

004127E8 68 E06F4C00 PUSH My_Scree.004C6FE0; ASCII "My Screen Recorder Pro - unregistered"

00418E95 68 E06F4C00 PUSH My_Scree.004C6FE0; ASCII "My Screen Recorder Pro - unregistered"

00422F88 68 A4974C00 PUSH My_Scree.004C97A4; ASCII "unregistered - Trial Version"

0045151C 68 B8104D00 PUSH My_Scree.004D10B8; ASCII "- unregistered"

_ Double Click on 0045151C

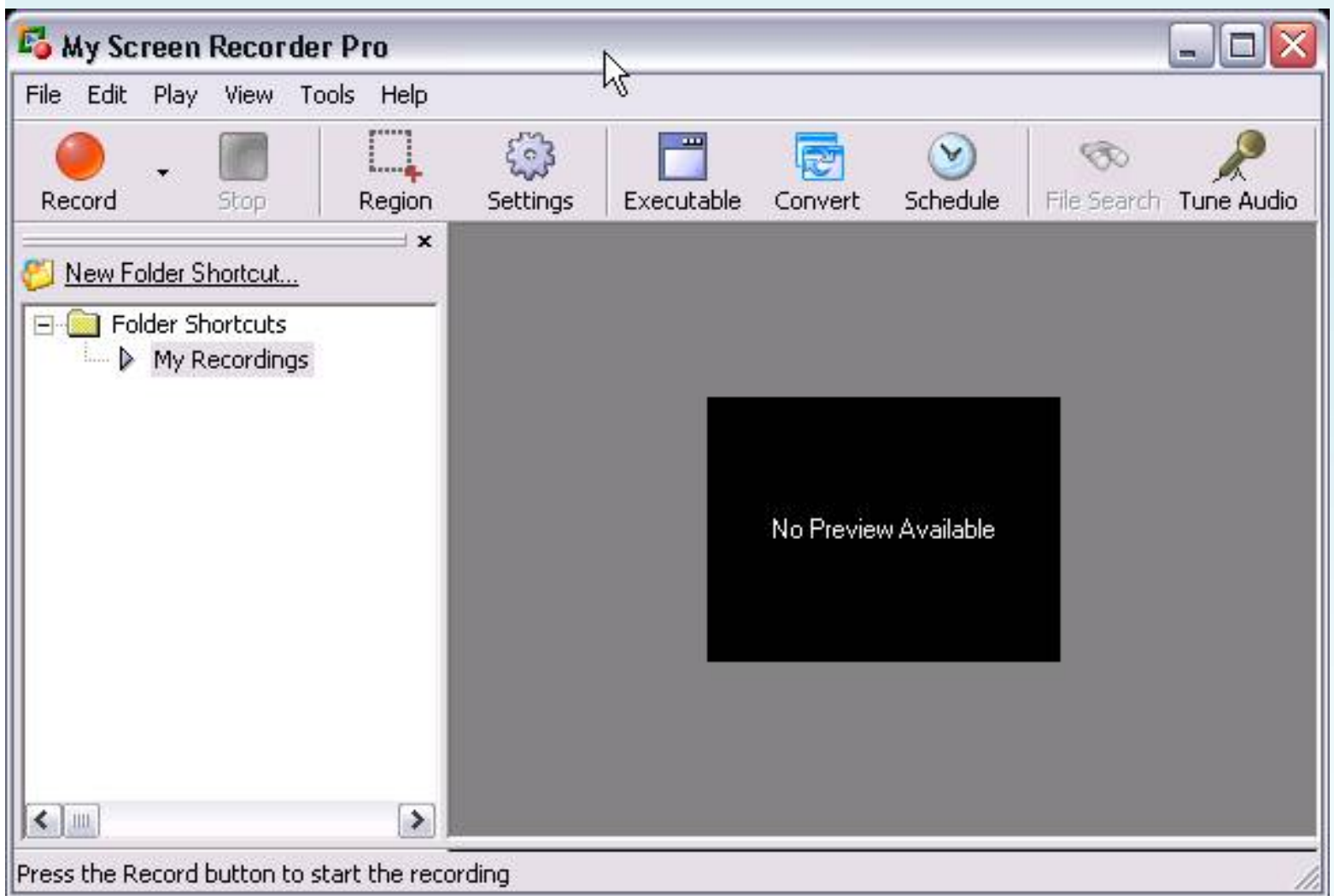
Address	Hex dump	Disassembly	Comment
004514EA	E8 D7910400	JMP dumped_.0049A6C6	
004514EF	8B4E 1C	MOV ECX, DWORD PTR DS:[ESI+1C]	
004514F2	51	PUSH ECX	
004514F3	FF15 C8254C00	JMP NEAR DWORD PTR DS:[<&user32.UpdateWindow]	USER32.UpdateWindow
004514F9	A1 74DF4F00	MOV EAX, DWORD PTR DS:[4FDF74]	
004514FE	85C0	TEST EAX, EAX	
00451500	75 3D	JNZ SHORT dumped_.0045153F	
00451502	8D4C24 0C	LEA ECX, DWORD PTR SS:[ESP+C]	
00451506	E8 4507FBFF	JMP dumped_.00401C50	
00451508	8D5424 0C	LEA EDX, DWORD PTR SS:[ESP+C]	
0045150F	52	PUSH EDX	
00451510	8BCE	MOV ECX, ESI	
00451512	C64424 4C 07	MOV BYTE PTR SS:[ESP+4C], 7	
00451517	E8 19790400	JMP dumped_.00498E35	
0045151C	68 B8104D00	PUSH dumped_.004D10B8	ASCII "- UNREGISTERED"
00451521	8D4C24 10	LEA ECX, DWORD PTR SS:[ESP+10]	
00451525	E8 2664FBFF	JMP dumped_.00407960	
0045152A	8B4424 0C	MOV EAX, DWORD PTR SS:[ESP+C]	
0045152E	50	PUSH EAX	
0045152F	8BCE	MOV ECX, ESI	
00451531	E8 F3900400	JMP dumped_.0049A629	
00451536	8D4C24 0C	LEA ECX, DWORD PTR SS:[ESP+C]	

00451500 / 75 3D JNZ SHORT My_Scree.0045153F

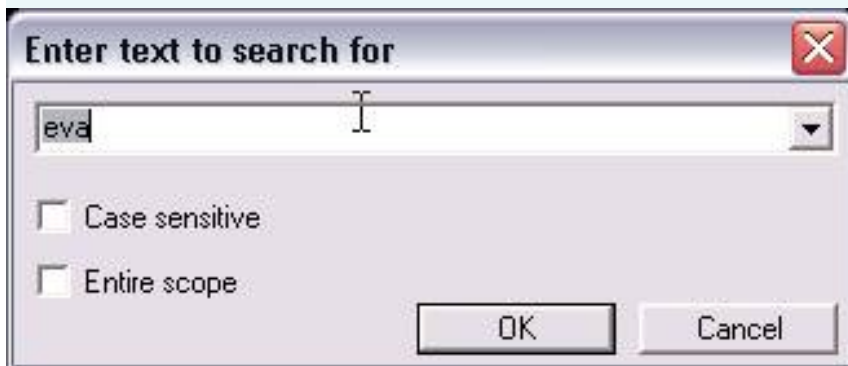
Patch's

00451500 / 74 3D JE SHORT My_Scree.0045153F

_ Save File and test to see how he he lost the word unregistered packages.



NAG 1 _Con the more pressing when the Record button on the window and type Text Search



_Nhan OK to me here

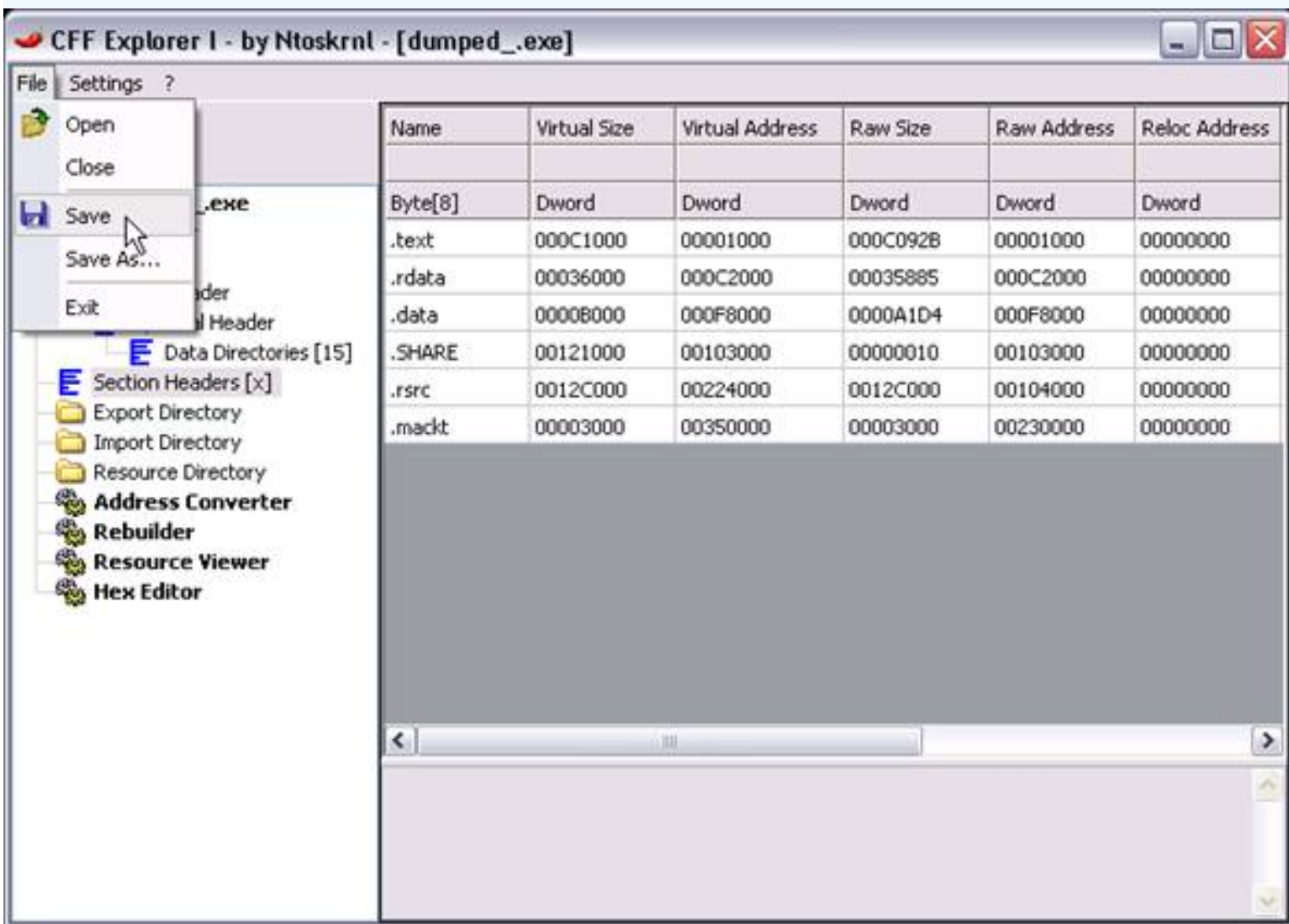
Address	Hex dump	Disassembly	Comment
0041490C	C68424 1C820000	MOV BYTE PTR SS:[ESP+21C],0	
00414914	E8 77D7FEFF	JMP My_Scree.00402090	
00414919	808C24 88800000	LEA ECX,DWORD PTR SS:[ESP+88]	
00414920	C68424 18020000	MOV BYTE PTR SS:[ESP+218],3	
00414928	E8 03EC0000	JMP My_Scree.00423600	
0041492D	C68424 EC010000	MOV BYTE PTR SS:[ESP+1EC],9	
00414935	E8 B680FFFF	JMP My_Scree.004006F0	
0041493A	391D 74DF4F00	CMPL DWORD PTR DS:[4FDF74],EBX	
00414940	75 0C	JNZ SHORT My_Scree.0041494E	
00414942	53	PUSH EBX	
00414943	53	PUSH EBX	
00414944	68 F8714C00	JMP My_Scree.004C71F8	ASCII "This is an evaluation copy. All recorded videos will be
00414949	E8 1ACD0000	JMP My_Scree.00401668	
0041494E	804C24 5C	LEA ECX,DWORD PTR SS:[ESP+5C]	
00414952	E8 3C740000	JMP My_Scree.00496D93	
00414957	83F8 02	CMPL EAX,2	
0041495A	75 4C	JNZ SHORT My_Scree.00414968	
0041495C	C705 04874F00	MOV DWORD PTR DS:[4F8704],1	
00414966	E8 35C2FFFF	JMP My_Scree.00410BA0	
0041496B	804C24 5C	LEA ECX,DWORD PTR SS:[ESP+5C]	
0041496F	891D 3CDE4F00	MOV DWORD PTR DS:[4FDE3C],EBX	
00414975	891D E8DA4F00	MOV DWORD PTR DS:[4FDAE8],EBX	
00414978	C68424 EC010000	MOV BYTE PTR SS:[ESP+1EC],0	
00414983	E8 18C4FFFF	JMP My_Scree.00410DA0	
00414988	804C24 1C	LEA ECX,DWORD PTR SS:[ESP+1C]	

00414940 / 75 0C JNZ SHORT My_Scree.0041494E

Patch's

00414940 / 74 0C JE SHORT My_Scree.0041494E

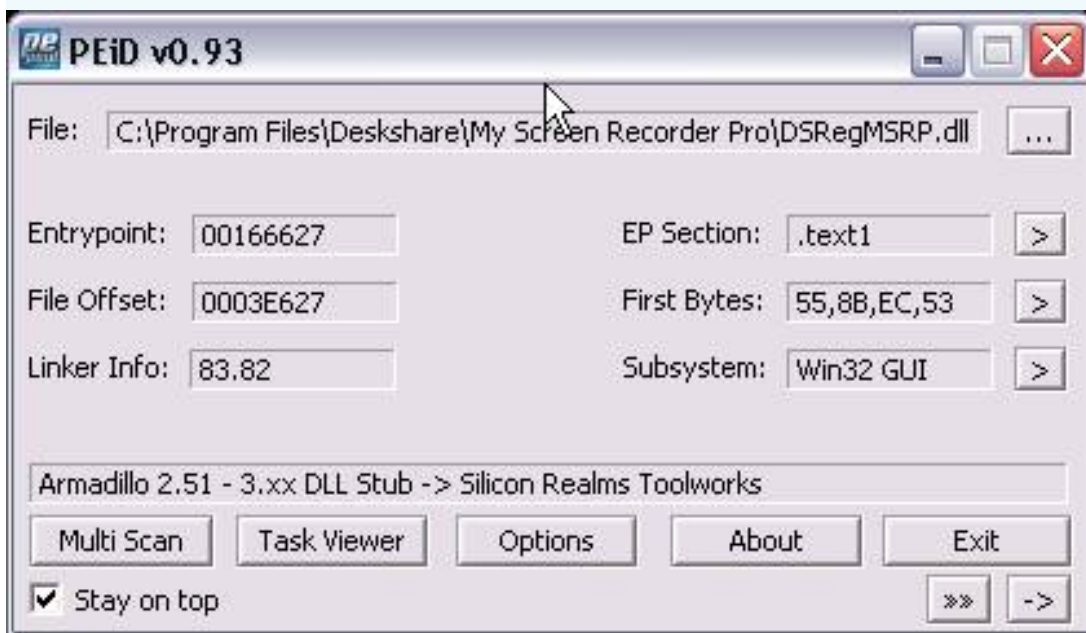
_Save File and Test, Oh everything has Done. Here you run to embrace but also to the smaller amount you use CFF Explorer to delete Section reduce the excess. Text1. ADATA. Data1. Pdata . This is the Section tàn of Arma.



Now as then finished it you. But in fact when you click About

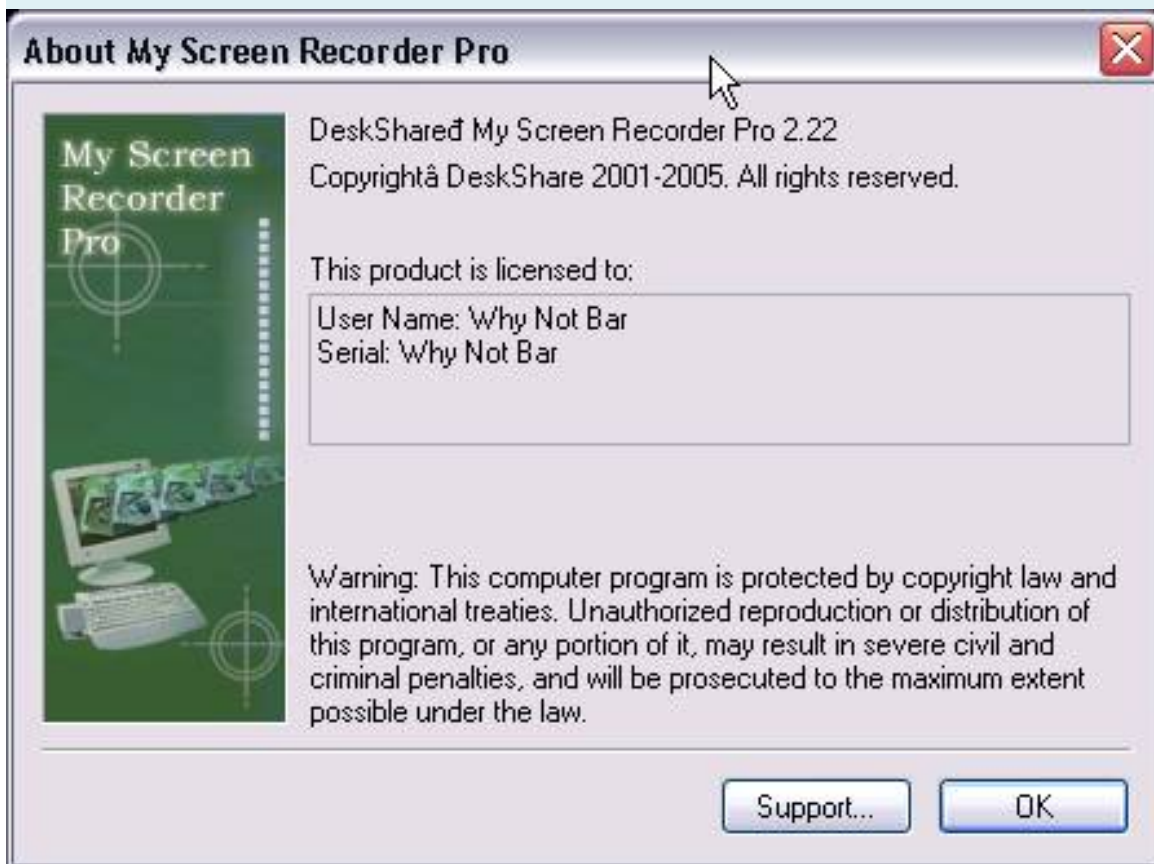


_ These are located in the File DSRegMSRP.dll and it is in Arma Pack



_ Muon Unpack it and see Uncle tut 9 of Hacnho. I do not presented here. As it can do for this tut Available on it but I have attached file unpack it in tut for your reference. The suggestions for when you unpack this dll files you should change its Image Base 20000000 is to run it. Subject 10000000

error feedback it only. If you are successful, the same as follows



_ Wish you success

Written by Why Not Bar (20-10-2005)

Obsidium 1.2.5.0 - unpacking

(I combined the practice of my own material with the author of this article to become more specific. So they can only be wrong, but the code is similar).

1. Intro:

The tools we need:

Windows XP, OllyDbg; ImpREC; LordPE; hex editor; PEID.

Target is attached.

Obsidium some of the technical anti-debug:

- CheckRemoteDebuggerPresent: this API detected debugger on XP machines.
- UnhandledExceptionFilter: do crash load is applied by the debugger.
- FindWindowA: it finds olly window class "OLLYDBG"
- IsDebuggerPresent: detect debugger.
- Threads: obsidium have one or two thread Ma Su used the technique on. (Because the procedure only checked once, and the procedure was performed 2 times).
- OEP stolen code: it will be some legs in OEP bytes.
- Cool import protection.

We only need to use plugin Hide Debugger is overcome. Here the authors present a brief bit about the technique:

Obsidium many "code nhảm" (junk code) with the command jmp short jump as uncomfortable as we explore it. I really hate junk code. Obsidium not GetProcAddress for the imports, which it uses a procedure to search for imports. It also checks the breakpoints (BP) in the code of the API, however we are still breakpoints place at the end of the API.

The procedure anti-debugger: CheckRemoteDebuggerPresent, FindWindowA, IsDebuggerPresent, can be overcome, by placing on BP and API changes EAX to 0.

But this function is used UnhandledExceptionFilter control exceptions that protect the community. The problem is if we use the debugger (Olly), the application will be that the debugger has taken control instead of the debugger system (Mr. Watson). We can make the system think that the application does not work in other debugger.

Look to UnhandledExceptionFilter and may be different for you:

`PUSH 248 <----- UnhandledExceptionFilter function starts here.`

`PUSH kernel32.7C8635E0`


```
CALL kernel32.7C8024CB
MOV EAX, DWORD PTR DS: [7C8836CC]
MOV DWORD PTR SS: [EBP-1C], EAX
MOV EBX, DWORD PTR SS: [EBP +8]
MOV DWORD PTR SS: [EBP-178], EBX
MOV DWORD PTR SS: [EBP-148], 4
XOR EDI, EDI
MOV DWORD PTR SS: [EBP-13C], EDI
MOV DWORD PTR SS: [EBP-16C], EDI
MOV EAX, DWORD PTR DS: [EBX]
TEST BYTE PTR DS: [EAX +4], 10
JE SHORT kernel32.7C862BD4
PUSH DWORD PTR DS: [EAX]
PUSH -1
CALL DWORD PTR DS: [<& ntdll.NtTerminatePr>; ntdll.ZwTerminateProcess
MOV EAX, DWORD PTR DS: [EBX]
MOV ESI, C0000005
Cmp DWORD PTR DS: [EAX], ESI
JNZ SHORT kernel32.7C862BF9
Cmp DWORD PTR DS: [EAX +14], 1
JNZ SHORT kernel32.7C862BF9
PUSH DWORD PTR DS: [EAX +18]
CALL kernel32.7C862874
Cmp EAX, -1
JNZ SHORT kernel32.7C862BF9
OR EAX, EAX
JMP kernel32.7C863458
```

```
MOV DWORD PTR SS: [EBP-124], EDI
```

```
PUSH EDI
```

```
PUSH 4
```

```
LEA EAX, DWORD PTR SS: [EBP-124]
```

```
PUSH EAX
```

```
PUSH 7
```

CALL kernel32.GetCurrentProcess <----- Here is the trick! This API returns FFFFFFFF. Change EAX to 0.

```
PUSH EAX
```

```
CALL DWORD PTR DS: [<& ntdll.NtQueryInform>; ntdll.ZwQueryInformationProcess
```

```
TEST EAX, EAX
```

```
...
```

```
...
```

With others, you also find the same points, is this:

```
PUSH7
```

```
CALL kernel32.GetCurrentProcess
```

Then edit the FFFFFFFFh EAX = EAX = 0.

2. OEP and stolen Code:

There is no general way to find the correct OEP (if there is a general way you can find OEP writing scripts.)

Instead I have the traces and even the test multiple times codes, exceptions, breakpoints. I found out where the command imports jumped be located, when the unpack crackme by the protection, I put BP on the most Import volume (not a problem because this small crackme).

Um, you should check all options in Olly exception, and then run crackme:

Then, look at the log window:

Log data

Message

Access indoctrination when reading [00000000]

Access indoctrination when reading [FFFFFFFF]

Module C: \ WINDOWS \ system32 \ advapi32.dll

Access indoctrination when reading [FFFFFFFF]

Breakpoint at kernel32.7C85994E

INT3 command at ntdll.DbgBreakPoint

Access indoctrination when executing [00000000]

Breakpoint at kernel32.7C862C10

Illegal instruction

Integer division by zero

Breakpoint at USER32.77D6F3DC

Access indoctrination when executing [00000000]

Breakpoint at kernel32.7C862C10

Integer division by zero

Access indoctrination when reading [00000000]

Access indoctrination when reading [8003F41E]

Integer division by zero

Access indoctrination when reading [00000000]

Access indoctrination when reading [FFFFFFFF]

Access indoctrination when reading [FFFFFFFF]

Access indoctrination when reading [8003F418]

Illegal instruction

Integer division by zero

Breakpoint at kernel32.7C812E10

INT3 command at 00396555

Integer division by zero

Integer division by zero

Integer division by zero

Integer division by zero

Integer division by zero

Integer division by zero

Illegal instruction

Integer division by zero <----- this one.

New thread with ID 00000304 created

Breakpoint at kernel32.7C85994E

Exception finally "integer division by zero." Olly restart again, uncheck option "integer division by zero," So when Olly run, we will end it on Exception. After some time we will here:

```
F7F0 DIV EAX <----- Exception.
```

```
55 PUSH EBP
```

```
8BEC MOV EBP, ESP
```

```
8B45 08 MOV EAX, DWORD PTR SS: [EBP +8]
```

```
6A 00 PUSH 0
```

```
68 CC971025 PUSH 251097CC
```

```
6A 00 PUSH 0
```

```
FF50 18 CALL DWORD PTR DS: [EAX +18]
```

```
5D POP EBP
```

```
C2 0800 RETN 8
```

```
55 PUSH EBP
```

```
8BEC MOV EBP, ESP
```

```
8B45 08          MOV EAX,DWORD PTR SS:[EBP+8]
```

```
8B00 MOV EAX, DWORD PTR DS: [EAX]
```

```
3D 1D0000C0 Cmp EAX, C000001D
```

```
74 0E JE SHORT UnPackMe.00415FD0
```

```
3D 940000C0 Cmp EAX, C0000094
```

```
75 41 JNZ SHORT UnPackMe.0041600A
```

```
B9 AB92AD01 MOV ECX, 1AD92AB
```

```
EB 05 JMP SHORT UnPackMe.00415FD5
```

```
B9 4E90AD01 MOV ECX, 1AD904E
```

```
E8 00000000 CALL UnPackMe.00415FDA
```

```
5A POP EDX

8B45 10 MOV EAX, DWORD PTR SS: [EBP +10]

81EA 893CBA00 SUB EDX, 0BA3C89

8D940A EFAB0CFF LEA EDX, DWORD PTR DS: [EDX + ECX FF0CABEF +]

8990 B8000000 MOV DWORD PTR DS: [EAX + b8], EDX

33C9 XOR ECX, ECX

8948 04 MOV DWORD PTR DS: [EAX +4], ECX

8948 08 MOV DWORD PTR DS: [EAX +8], ECX

8948 0C MOV DWORD PTR DS: [EAX + C], ECX

8948 10 MOV DWORD PTR DS: [EAX +10], ECX

C740 18 55003333 MOV DWORD PTR DS: [EAX +18], 33330055

33C0 XOR EAX, EAX

5D POP EBP

C3 RETN

33C0 XOR EAX, EAX

40 INC EAX

5D POP EBP

C3 RETN

33C0 XOR EAX, EAX

8D44C0 03 LEA EAX, DWORD PTR DS: [EAX + EAX * 8 +3]

010424 ADD DWORD PTR SS: [ESP], EAX

C3 RETN
```

At this code section is completely unpack. I put a memory breakpoints on the code section and we will run to be "FALSE OEP.

Consider a bit.

First, this section crackme code is 00404000, the second is always Obsidium jump to 'FALSE OEP "although it is not" legs "bytes, it will jump to the binary code in the distance.

We check to prevent placement, we will see SEH handler that will control the exceptions:

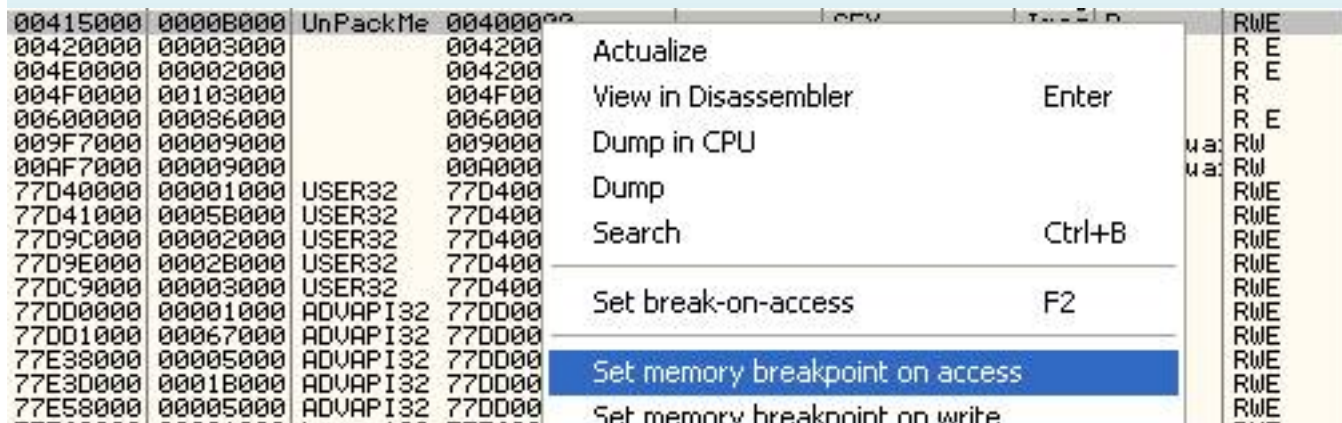
0012FF94 0012FFE0 Pointer to next SEH record

0012FF98 00415FB3 SE handler

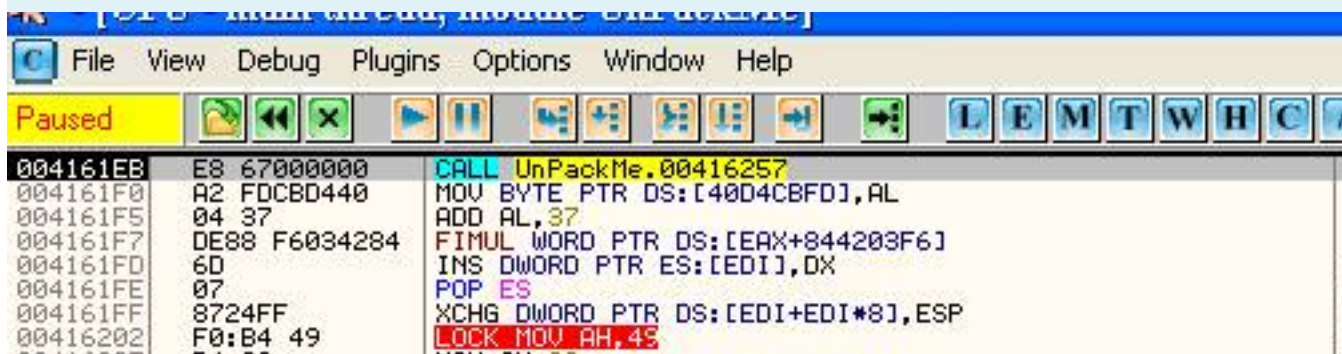
You set breakpoints in 0x00415FB3, Shift + F9, then F8 to RETN we ntdll.dll to:



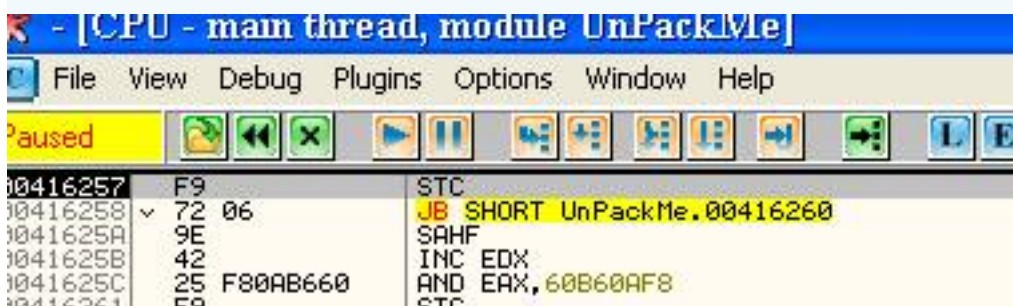
Then, we set breakpoints on the memory section of Obsidium:



In section 0041500. Then Shift + F9, we come:



F7 to address 004161EB:



Trace F8 repeatedly, you will see a loop in code:

004162B9	C1CF BF	ROR EDI,0BF
004162BC	83EA 04	SUB EDX,4
004162BF	0F85 04FFFFFF	JNZ UnPackMe.00416299
004162C5	EB 05	JMP SHORT UnPackMe.004162CC
004162C7	BD 7C3BF3D8	MOV EBP,D8F33B7C
004162C8	61	POPAD
004162CD	C3	RETN
004162CE	DB94D4 CD00D8A	FIST DWORD PTR SS:[ESP+EDX*8+01D800CD]

Trace out of the loop by F2 at 004162CD RETN, then Shift + F9 to RETN, F7, we arrive this:

004161EB	FF96 8C010000	CALL DWORD PTR DS:[ESI+18C]
004161F1	8B7E 04	MOV EDI,DWORD PTR DS:[ESI+4]
004161F4	FFB6 44010000	PUSH DWORD PTR DS:[ESI+144]
004161FA	FF96 80000000	CALL DWORD PTR DS:[ESI+80]
00416200	25 FF030000	AND EAX,3FF
00416205	8DBC07 00010000	LEA EDI,DWORD PTR DS:[EDI+EAX+100]
0041620C	56	PUSH ESI
0041620D	57	PUSH EDI
0041620E	8DB5 C83CBA00	LEA ESI,DWORD PTR SS:[EBP+BA3CC8]
00416214	B9 D2010000	MOV ECX,1D2
00416219	F3:A4	REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
0041621B	5F	POP EDI
0041621C	5E	POP ESI
0041621D	8D85 F63EBA00	LEA EAX,DWORD PTR SS:[EBP+BA3EF6]
00416223	68 80000000	PUSH 80
00416228	50	PUSH EAX
00416229	FF76 04	PUSH DWORD PTR DS:[ESI+4]
0041622C	FF76 20	PUSH DWORD PTR DS:[ESI+20]
0041622F	FF96 80000000	CALL DWORD PTR DS:[ESI+80]
00416235	6A 3A	PUSH 3A
00416237	57	PUSH EDI
00416238	50	PUSH EAX
00416239	FF76 04	PUSH DWORD PTR DS:[ESI+4]
0041623C	FF76 28	PUSH DWORD PTR DS:[ESI+28]
0041623F	FF96 80000000	CALL DWORD PTR DS:[ESI+80]
00416245	FFE7	JMP EDI
00416247	112A	CALL DWORD PTR DS:[ESI+18C]

Then you look down a bit, we'll see JMP EDI 00416245, 00416245 Please, you should à Hardware breakpoints set execution (F2 not be set at above), and remember to remove the trace breakpoints when completed, we to code the following:

0037831E	E8 00000000	CALL 00378323
00378323	5D	POP EBP
00378324	81ED CD3CBA00	SUB EBP,0BA3CCD
0037832A	64:67:8F06 0000	POP DWORD PTR FS:[0]
00378330	83C4 04	ADD ESP,4
00378333	C685 FD3CBA00	MOV BYTE PTR SS:[EBP+BA3CFD],0E9
0037833A	8B95 063DBA00	MOV EDX,DWORD PTR SS:[EBP+BA3D06]
00378340	0356 10	ADD EDX,DWORD PTR DS:[ESI+10]
00378343	8D8D 023DBA00	LEA ECX,DWORD PTR SS:[EBP+BA3D02]
00378349	2BD1	SUB EDX,ECX
0037834B	8995 FE3CBA00	MOV DWORD PTR SS:[EBP+BA3CFE],EDX
00378351	61	POPAD
00378352	9D	POPFD
00378353	E9 B7BC0800	JMP UnPackMe.0040400F
00378358	26:0000	ADD BYTE PTR ES:[EAX],AL

The doctors should F8 respectively are the new shape as above. We stopped in 00378353:

JMP 0040400F ----- à orders jumped to FALSE OEP

F7 um, we come:

- [CPU - main thread, module UnPackMe]

File View Debug Plugins Options Window Help

Paused

Address	Disassembly	Comment
0040400F	EB 01040000	CALL UnPackMe.00404415
00404014	68 00000000	PUSH 0
00404019	FF15 F4114000	CALL DWORD PTR DS:[4011F4]
0040401F	A3 07F04000	MOV DWORD PTR DS:[40F007],EAX
00404024	68	PUSHAD
00404025	8925 0BF04000	MOV DWORD PTR DS:[40F00B],ESP
0040402B	E9 30000000	JMP UnPackMe.00404060
00404030	8B25 0BF04000	MOV ESP,DWORD PTR DS:[40F00B]
00404036	61	POPAD
00404037	E8 19090000	CALL UnPackMe.00404955
0040403C	E8 6D040000	CALL UnPackMe.004044AE
00404041	89EC	MOV ESP,EBP
00404043	5D	POP EBP
00404044	FF35 D4F14000	PUSH DWORD PTR DS:[40F1D4]
0040404A	FF15 EC114000	CALL DWORD PTR DS:[4011EC]
00404050	9B	WAIT
00404051	DBE2	FCLEX
00404053	092D 00604000	FLOCW WORD PTR DS:[406000]
00404059	C3	RETN
0040405A	00	DB 00
0040405B	00	DB 00
0040405C	00	DB 00
0040405D	00	DB 00
0040405E	00	DB 00
0040405F	00	DB 00
00404060	> 68 00000000	PUSH 0
00404065	B8 7C604000	MOV EAX,UnPackMe.0040607C
0040406A	89C2	MOV EDX,EAX
0040406C	52	PUSH EDX
0040406D	B8 54604000	MOV EAX,UnPackMe.00406054
00404072	89C6	MOV ESI,EAX
00404074	56	PUSH ESI
00404075	E8 88F00000	CALL UnPackMe.00405005
0040407A	89C7	MOV EDI,EAX
0040407C	57	PUSH EDI
0040407D	B8 00000000	MOV EAX,00
00404082	58	PUSH EAX
00404083	E8 4D030000	CALL UnPackMe.004043D5
00404088	89C6	MOV ESI,EAX
0040408A	56	PUSH ESI
0040408B	E8 75F00000	CALL UnPackMe.00405005
00404090	89C7	MOV EDI,EAX
00404092	56	PUSH ESI
00404093	E8 C6050000	CALL UnPackMe.0040465E
00404098	57	PUSH EDI
00404099	B8 53604000	MOV EAX,UnPackMe.00406053

0012FFF0

ASCII "Coded by Teddy Rogers / SnD Team 2005

ASCII "If you unpack it write a tutorial...

[Arg1 = 00000000
UnPackMe.00405005

[Arg1 = 00000000
UnPackMe.004043D5

[Arg1 = 00000000
UnPackMe.00405005

[Arg1 = 00000000
UnPackMe.0040465E

If someone does not correct it so please use the plugin Analyze this! By OLLY.

OEP is at 00404000, the script is the junk code, they will complicate our blind. Now we conduct dump. (This is in crackme crackme still work fine).

3. Imports

You find the following binary string: FF 25, and then choose a command's dance, Enter, you will see the following:

003E0060	60	PUSHAD
003E0061	66:BE 22A2	MOV SI,0A222
003E0065	B7 67	MOV BH,67
003E0067	- E9 B553F9FF	JMP 00375421
003E006C	60	PUSHAD
003E006D	66:BE 23A2	MOV SI,0A223
003E0071	B7 67	MOV BH,67
003E0073	- E9 A953F9FF	JMP 00375421
003E0078	60	PUSHAD
003E0079	66:BE 20A2	MOV SI,0A220
003E007D	B7 67	MOV BH,67
003E007F	- E9 9D53F9FF	JMP 00375421
003E0084	60	PUSHAD
003E0085	66:BE 21A2	MOV SI,0A221
003E0089	B7 67	MOV BH,67
003E008B	- E9 9153F9FF	JMP 00375421
003E0090	60	PUSHAD
003E0091	66:BE 26A2	MOV SI,0A226
003E0095	B7 67	MOV BH,67
003E0097	- E9 8553F9FF	JMP 00375421
003E009C	60	PUSHAD
003E009D	66:BE 27A2	MOV SI,0A227
003E00A1	B7 67	MOV BH,67
003E00A3	- E9 7953F9FF	JMP 00375421
003E00A8	60	PUSHAD
003E00A9	66:BE 24A2	MOV SI,0A224
003E00AD	B7 67	MOV BH,67
003E00AF	- E9 6D53F9FF	JMP 00375421
003E00B4	60	PUSHAD
003E00B5	66:BE 25A2	MOV SI,0A225
003E00B9	B7 67	MOV BH,67
003E00BB	- E9 6153F9FF	JMP 00375421
003E00C0	60	PUSHAD
003E00C1	66:BE 2AA2	MOV SI,0A22A
003E00C5	B7 67	MOV BH,67
003E00C7	- E9 5553F9FF	JMP 00375421
003E00CC	60	PUSHAD
003E00CD	66:BE 2BA2	MOV SI,0A22B
003E00D1	B7 67	MOV BH,67
003E00D3	- E9 4953F9FF	JMP 00375421
003E00D8	60	PUSHAD
003E00D9	66:BE 28A2	MOV SI,0A228
003E00DD	B7 67	MOV BH,67
003E00DF	- E9 3D53F9FF	JMP 00375421
0A3FA9F4	6A	PUSHAD

The order will jump to skip the process of Obsidium.O that does not function Obsidium GetProcAddress that instead, the imports are stored as:

- Dll image base (handle),
- Some flag (FFFFFFFF is the good one),
- Import code (which is 2,4,1,80 or 40),
- First character of import,
- CRC32 hash.

Obsidium will get information that programs need to use dll, then it will use some technical check inside (this is necessary import or not, ...), then it will search in the export dll symbols The first will be matched with something in the table "obsidium import info", then it will search for the import is based on the CRC32 hash with a value in this table.

Now is an example of us:

003E00B4	60	PUSHAD
003E00B5	66: BE 25A2	MOV SI, 0A225
003E00B9	B7 67	MOV BH, 67
003E00BB	- E9 6153F9FF	JMP 00375421

We see the following analysis:

003E00B4 60 PUSHAD

003E00B5 66: BE 25A2 MOV SI, 0A225 <----- This is 2gia of which contains information

003E00B9 B7 67 MOV BH, 67 of <-----/ dll

003E00BB-E9 695389FF JMP 00395429

Then, you enter the address 003E00BB:

Paused			
00375421	0FB6D7	MOVZX EDX, BH	
00375424	0FB7C6	MOVZX EAX, SI	
00375427	E8 00000000	CALL 0037542C	
0037542C	5B	POP EBX	
0037542D	88EB	MOV EBP, EBX	
0037542F	8B5B 8C	MOV EBX, DWORD PTR DS:[EBX-74]	
00375432	81ED CBA0B000	SUB EBP, 0B8A00C8	
00375438	E8 00000000	CALL 0037543D	
0037543D	830424 0D	ADD DWORD PTR SS:[ESP], 0D	
00375441	8B8D BCA0B000	MOV ECX, DWORD PTR SS:[EBP+0B8A00C8]	
00375447	FFD1	CALL ECX	
00375449	C3	RETN	
0037544A	8B73 44	MOV ESI, DWORD PTR DS:[EBX+44]	
0037544D	3295 8CA1B000	XOR DL, BYTE PTR SS:[EBP+0B8A18C]	
00375453	66: 3385 8DA1B000	XOR AX, WORD PTR SS:[EBP+0B8A18D]	
0037545A	C1E2 03	SHL EDX, 3	
0037545D	8BCA	MOV ECX, EDX	
0037545F	D1E2	SHL EDX, 1	
00375461	03CA	ADD ECX, EDX	
00375463	8D7C0E 04	LEA EDI, DWORD PTR DS:[ESI+ECX+4]	
00375467	837F 04 00	CMPL DWORD PTR DS:[EDI+4], 0	
00375468	0F84 13020000	JE 00375684	
00375471	0377 14	ADD ESI, DWORD PTR DS:[EDI+14]	
00375474	8D34C6	LEA ESI, DWORD PTR DS:[ESI+EAX*8]	
00375477	0FB706	MOVZX EAX, WORD PTR DS:[ESI]	
0037547A	83F8 04	CMPL EAX, 4	
0037547D	0F84 97000000	JE 0037551A	
00375483	83F8 01	CMPL EAX, 1	
00375486	74 68	JE SHORT 003754F0	
00375488	3D 80000000	CMPL EAX, 80	
0037548D	0F84 B9020000	JE 0037574C	
00375493	83F8 40	CMPL EAX, 40	
00375496	0F84 AA000000	JE 00375546	
0037549C	0FB746 02	MOVZX EAX, WORD PTR DS:[ESI+2]	
003754A0	6A 01	PUSH 1	
003754A2	50	PUSH EAX	
003754A3	6A 00	PUSH 0	
003754A5	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
003754A8	FF37	PUSH DWORD PTR DS:[EDI]	
003754AA	FF53 50	CALL DWORD PTR DS:[EBX+50]	
003754AD	85C0	TEST EAX, EAX	
003754AF	75 51	JNZ SHORT 00375502	

View:

00375467 Cmp DS DWORD PTR [EAX + EDI + 4] check that, if the load is dll, the value is FFFFFFFF. I call this [1]

00375477 MOVZX EAX, WORD PTR DS: [ESI] <----- It will take from the table "import code" (2, 4, 1, 80, 40) [2]

003754AA CALL DWORD PTR DS: [EBX + 50] <----- If "import code" is 2, it

searches import it here.

TEST EAX, EAX <----- [3] EAX is API value. [3]

Most import is in "code 2" category.

I used to edit the script imports. First, we review panel Imports in the original file:

```
0040119C 003E0000
004011A0 003E000C
004011A4 003E0018
004011A8 003E0024
004011AC 003E0030
004011B0 003E003C
004011B4 003E0048
004011B8 003E0054
004011BC 00000000
004011C0 00000000
004011C4 00000000
004011C8 00000000
```

Now it is conducted manually fix Imports Heng.

Put "New Origin here" in the first 003E0000, then F7 to

003E0007 Jmp 00375421

Then F7 to 00375421, and then set break points in 3 position [1], [2], [3]:

00375421	0FB6D7	MOVZX EDX,BH	
00375424	0FB7C6	MOVZX EAX,SI	
00375427	E8 00000000	CALL 0037542C	
0037542C	5B	POP EBX	0012FFF0
0037542D	8BEB	MOV EBP,EBX	
0037542F	8B5B 8C	MOV EBX,DWORD PTR DS:[EBX-74]	
00375432	81ED CBA0B800	SUB EBP,0B8A0CB	
00375438	E8 00000000	CALL 0037543D	
0037543D	830424 0D	ADD DWORD PTR SS:[ESP],0D	
00375441	8B8D BCA0B800	MOV ECX,DWORD PTR SS:[EBP+B8A0BC]	
00375447	FFD1	CALL ECX	
00375449	C3	RETN	
0037544A	8B73 44	MOV ESI,DWORD PTR DS:[EBX+44]	
0037544D	3295 8CA1B800	XOR DL,BYTE PTR SS:[EBP+B8A18C]	
00375453	66:3385 8DA1B800	XOR AX,WORD PTR SS:[EBP+B8A18D]	
0037545A	C1E2 03	SHL EDX,3	
0037545D	8BCA	MOV ECX,EDX	
0037545F	D1E2	SHL EDX,1	
00375461	03CA	ADD ECX,EDX	
00375463	8D7C0E 04	LEA EDI,DWORD PTR DS:[ESI+ECX+4]	
00375467	837F 04 00	CMP DWORD PTR DS:[EDI+4],0	
0037546B	0F84 13020000	JE 00375684	
00375471	0377 14	ADD ESI,DWORD PTR DS:[EDI+14]	
00375474	8D34C6	LEA ESI,DWORD PTR DS:[ESI+EAX*8]	
00375477	0FB706	MOVZX EAX,WORD PTR DS:[ESI]	
0037547B	83F8 04	CMP EAX,4	
0037547D	0F84 97000000	JE 0037551A	
00375483	83F8 01	CMP EAX,1	
00375486	74 68	JE SHORT 003754F0	
00375488	3D 80000000	CMP EAX,80	
0037548D	0F84 B9020000	JE 0037574C	
00375493	83F8 40	CMP EAX,40	
00375496	0F84 AA000000	JE 00375546	
0037549C	0FB746 02	MOVZX EAX,WORD PTR DS:[ESI+2]	
003754A0	6A 01	PUSH 1	
003754A2	50	PUSH EAX	
003754A3	6A 00	PUSH 0	
003754A5	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
003754A8	FF37	PUSH DWORD PTR DS:[EDI]	
003754AA	FF53 50	CALL DWORD PTR DS:[EBX+50]	
003754AD	85C0	TEST EAX,EAX	
003754AF	75 51	JNZ SHORT 00375502	
003754B1	33C9	XOR ECX,ECX	

One Shift + F9, stop at 003754AD TEST EAX, EAX to see the API, then fill to the IAT, in turn as follows:

```

0040119C 6E ED D4 77 BB D7 D4 77 7C B5 D4 77 0B 05 D8 n. 77. W. .. w | ..
w. .. w

004011AC B2 02 D7 77 54 05 D5 77 9F F2 D6 77 54 00 B0 00 ...w. .. .. wT wT ...

004011BC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

004011CC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

004011DC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

004011EC 60 00 B0 00 37 97 80 7C 29 B5 80 7C 2D FF 80 7C `..|)..|-.| ... 7

004011FC 2F FE 80 7C D4 05 91 7C AE 30 82 7C 29 29 81 7C /..|...|® 0 .|)).|

0040120C 10 11 81 7C 3D 04 91 7C B9 8F 83 7C E4 00 B0 00 ...|=..|...|....

0040404A CALL DWORD PTR DS: [4011EC] DS: [004011EC] = 00B00060

00404EF2 CALL DWORD PTR DS: [40119C] USER32.CallNextHookEx

00404F89 CALL DWORD PTR DS: [4011F0] kernel32.GetCurrentThreadId

00404C66 CALL DWORD PTR DS: [4011A0] USER32.GetDesktopWindow

00404019 CALL DWORD PTR DS: [4011F4] kernel32.GetModuleHandleA

00404C7F CALL DWORD PTR DS: [4011A4] USER32.GetWindowRect

004057CE CALL DWORD PTR DS: [4011F8] kernel32.GlobalAlloc

00405614 CALL DWORD PTR DS: [4011FC] kernel32.GlobalFree

00404622 CALL DWORD PTR DS: [401204] kernel32.HeapCompact

00404453 CALL DWORD PTR DS: [401208] kernel32.HeapCreate

00404584 CALL DWORD PTR DS: [40120C] kernel32.HeapDestroy

00404326 CALL DWORD PTR DS: [401214] kernel32.lstrcatA

00404FC9 CALL DWORD PTR DS: [4011A8] USER32.MessageBoxA

004045EF CALL DWORD PTR DS: [401200] ntdll.RtlAllocateHeap

0040463E CALL DWORD PTR DS: [401200] ntdll.RtlAllocateHeap

004046DD CALL DWORD PTR DS: [401210] ntdll.RtlFreeHeap

00404FA4 CALL DWORD PTR DS: [4011AC] USER32.SetWindowsHookExA

```



```
00404C17 CALL DWORD PTR DS: [4011B0] USER32.SystemParametersInfoA
```

```
00404FDB CALL DWORD PTR DS: [4011B4] USER32.UnhookWindowsHookEx
```

Only a single type of a "code 40", I trace a call that other than the above and is ExitProcess. (you should do with my hands like).

Finish.

Wish you luck.

Thanks.

Toolcracking

Armadillo collect sand-stone

Picture Ripper 3: Armadillo 4.xx-Import + Nanomites Complete elimination tut!**I. Intro:**

Tout le monde _Salut, now in the Net, I found some software was packed with options and Import Code Splicing elimination. And we can not unpack because the import was make in memory, I call this IAT is: Virtual IAT. Armadillo make Memory in the IAT and software to redirect. So, we can not use this because when IAT fix dump our dumped file, imprec show a dialog:



_It's A blessing when Admirallo make a very very good tools is: ArmaInline (the newest version is 0.6). ArmaInline appear before, I was found just one method redirect the virtual IAT by **Ricardo Narvaja** in the tuts ([203-ARMADILLO CON DESTRUCCION DE TABLA parte 1](#) [A CHILD 208-ARMADILLO DESTRUCCION DE TABLA Y FINAL PARTE 6](#)) with the target HyperSnap-DX.v5.60! But think in this method only for pro and not for beginner!

_In Exetools Forum, I was published an article called: Code Splicing + IAT elimination. Sorry my blunder! And now, I complete a written tut for this options. I'll do my best to make this tut become very easy for every body! Ok, the end of Intro. Go go go!

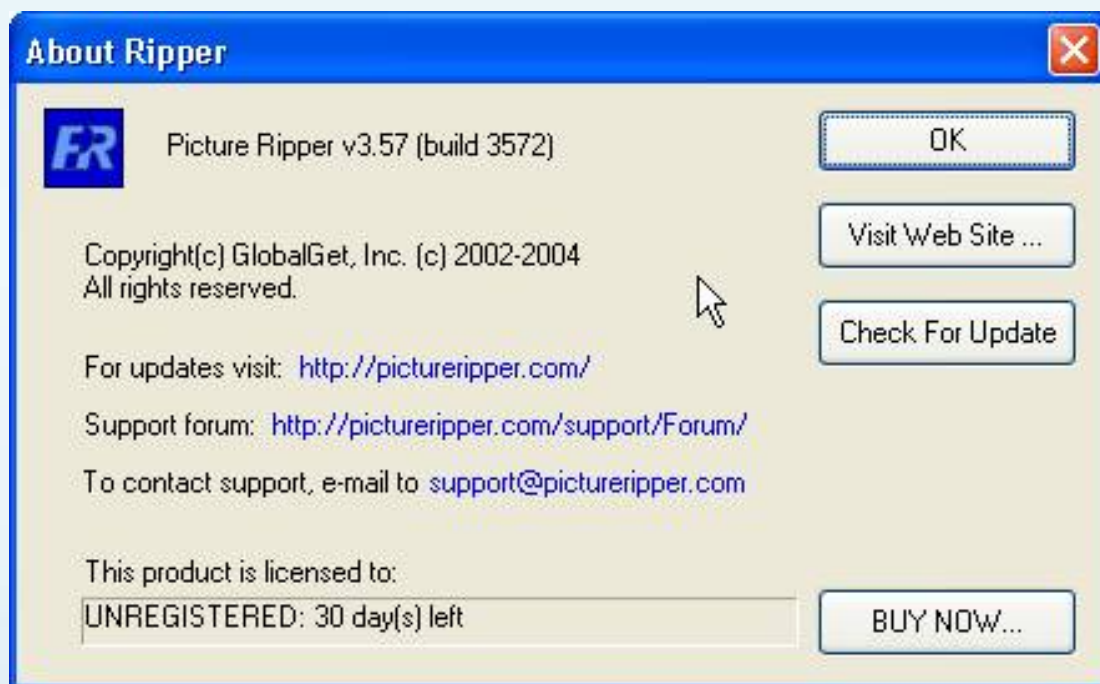
II. Tools

- 1.OllyDBG - The best config debugger for ArmMUP by hacnho.
- 2.LordPE 1.4 Deluxe
- 3.Import REConstructor 1.6 Final
- 4.ArmaInline 0.71

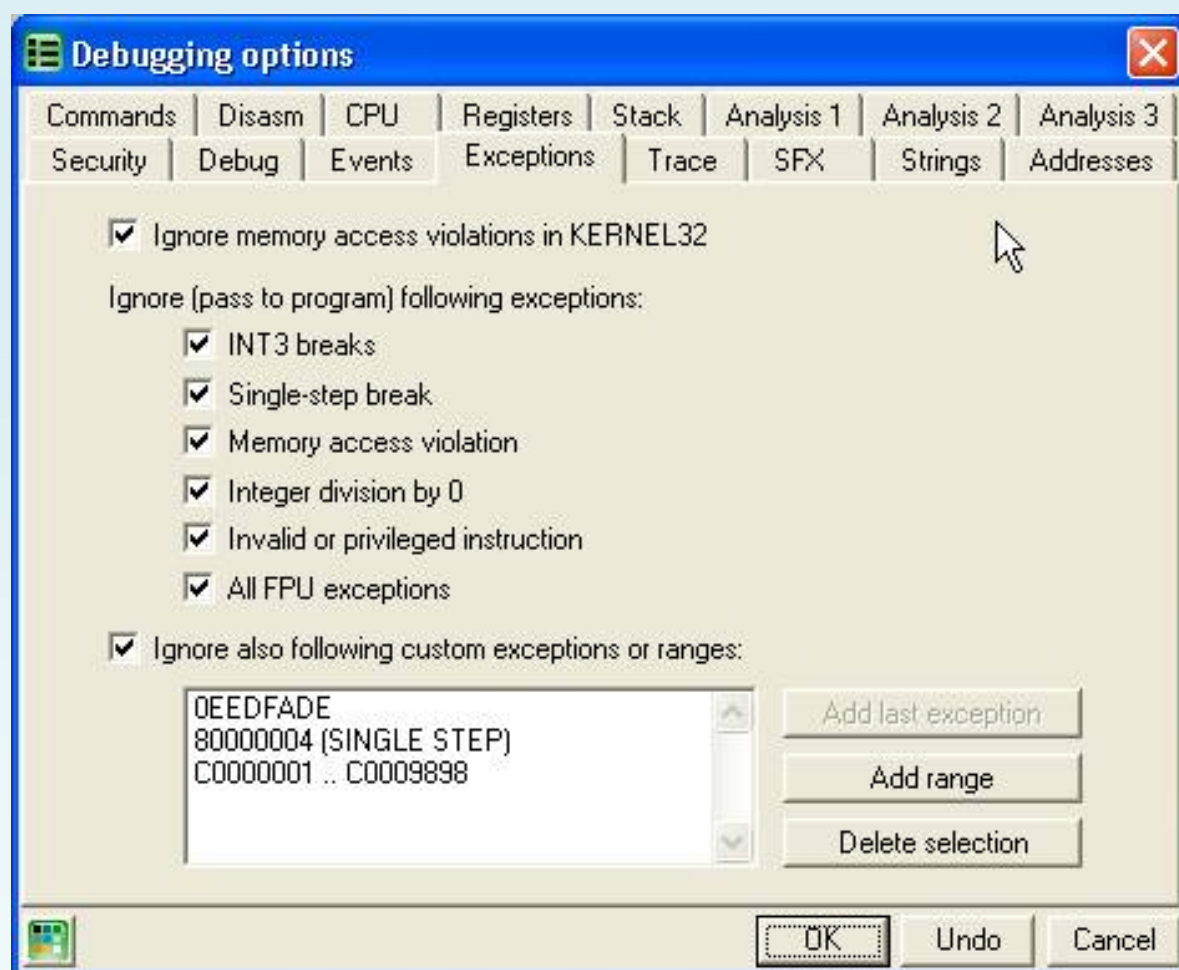
III. Unpacking

Target: Picture Ripper 3:57 build 3572

ARM Debug 4.xx-Blocker IAT elimination + + + Anti Breakpoint Nanomites



_Setting OllyDBG:



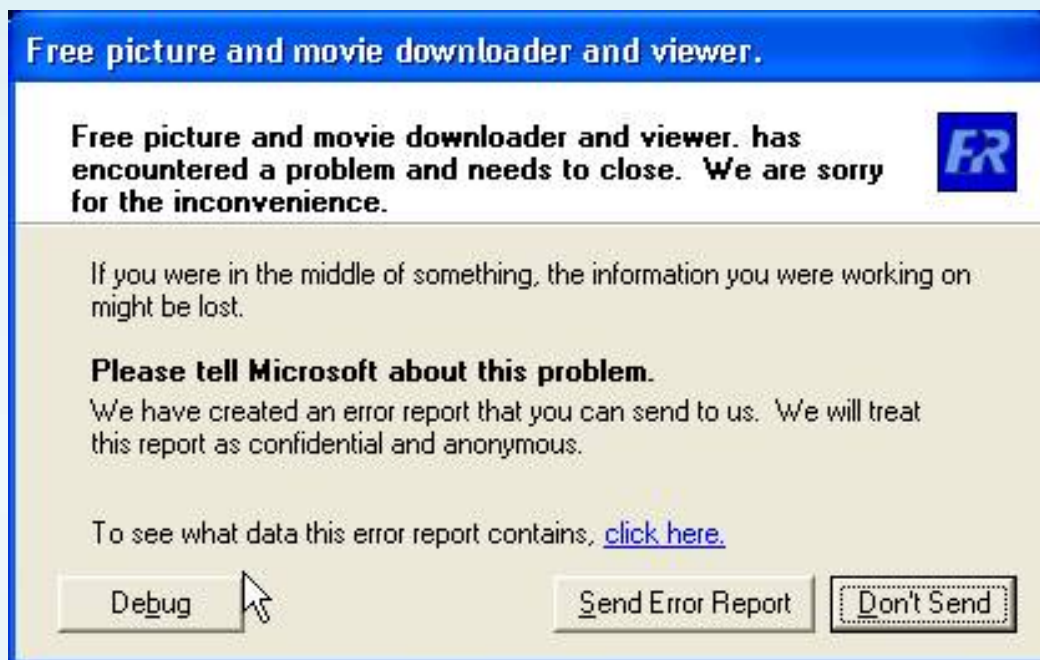
_Detect Target:

	c:\program files\messenger\msmsgs.exe	00000FC4	00400000
	c:\program files\pictureripper 3\pictureripper....	00000724	00400000
	c:\program files\pictureripper 3\pictureripper....	000000EC	00400000
	i:\cracker\utilities\lordpe deluxe - 1.4\lordpe....	00000F24	00400000

_Ok, Load into OllyDBG target:

Address	Hex dump	Disassembly	Comment
0059B000	60	ENTER	
0059B001	E8 00000000	CALL PictureR.0059B006	
0059B006	50	POP EBP	
0059B007	50	PUSH EAX	
0059B008	51	PUSH ECX	
0059B009	0FCA	BSWAP EDX	
0059B00B	F702	NOT EDX	
0059B00D	9C	PAUSE	
0059B00E	F702	NOT EDX	
0059B010	0FCA	BSWAP EDX	
0059B012	EB 0F	JMP SHORT PictureR.0059B023	
0059B014	89 EB0FB8EB	MOV ECX,EBB80FEB	
0059B019	07	POP ESI	Modification of segment regist

_bp WriteProcessMemory, Shift + F9:



_It's Crash. Oh, I think this target anti-bp! Ok. We must pass by the debug option by hand blocker. Ctrl + F2 restart target. Ctrl + G: WriteProcessMemory:



F2 _Press set a breakpoint at:

Address	Hex dump	Disassembly	Comment
77E61A94	55	PUSH EBP	
77E61A95	88EC	MOV EBP,ESP	
77E61A97	51	PUSH ECK	
77E61A98	51	PUSH ECK	
77E61A99	8845 0C	MOV EAX,DWORD PTR SS:[EBP+C]	
77E61A9C	53	PUSH EBX	
77E61A9D	885D 14	MOV EBX,DWORD PTR SS:[EBP+14]	
77E61AA0	56	PUSH ESI	
77E61AA1	8835 B012E677	MOV ESI,DWORD PTR DS:[&ntdll.NtProtectVirtualMemory]	ntdll.ZwProtectVirtualMemory
77E61AA7	57	PUSH EDI	
77E61AA8	887D 08	MOV EDI,DWORD PTR SS:[EBP+8]	
77E61AAB	8945 F8	MOV DWORD PTR SS:[EBP-8],EAX	

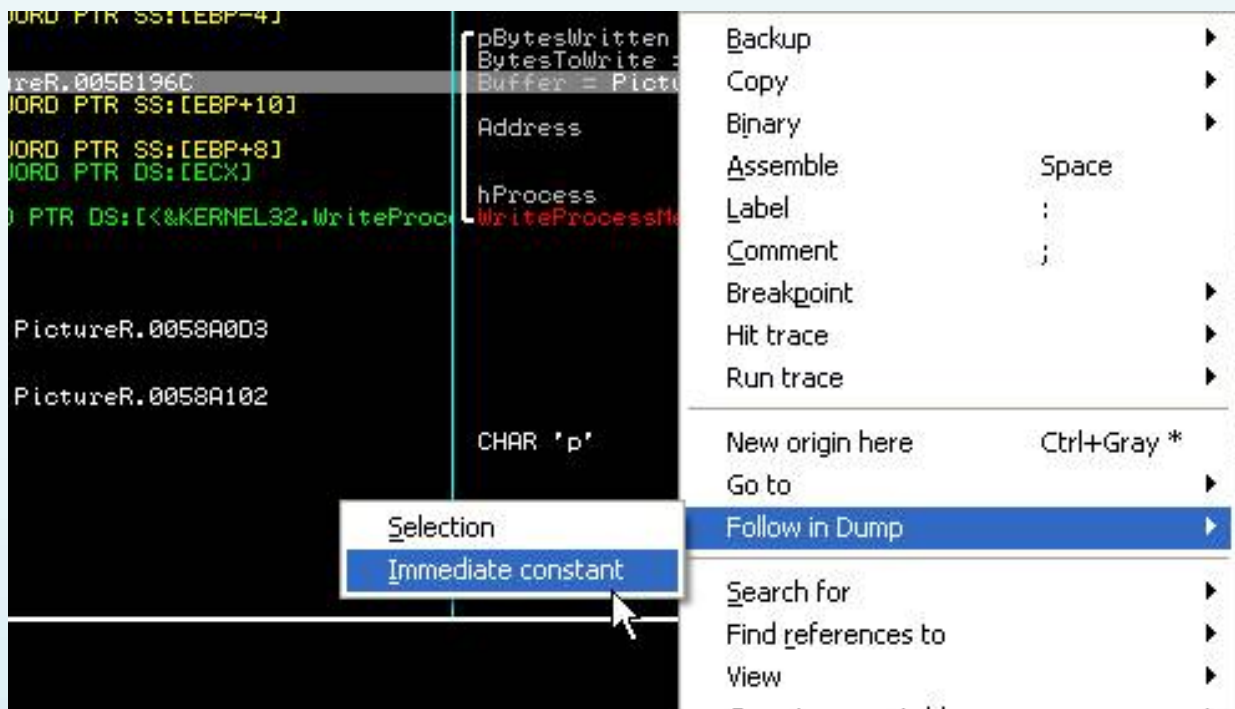
_Shift + F9, yeah. J OllyDBG break. Now, press Alt + F9:

Address	Hex dump	Disassembly	Comment
0058A0A3	<70 07	JL SHORT PictureR.0058A0AC	
0058A0A5	<7C 03	JL SHORT PictureR.0058A0AA	
0058A0A7	<EB 05	JMP SHORT PictureR.0058A0AE	
0058A0A9	E8 74FBEBF9	CALL FA449C22	
0058A0AE	<EB 5F	JMP SHORT PictureR.0058A10F	
0058A0B0	8055 FC	LEA EDX,DWORD PTR SS:[EBP-4]	
0058A0B3	52	PUSH EDX	
0058A0B4	6A 02	PUSH 2	
0058A0B6	68 6C195B00	PUSH PictureR.005B196C	
0058A0B8	8845 10	MOV EAX,DWORD PTR SS:[EBP+10]	
0058A0BE	50	PUSH EAX	
0058A0BF	884D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
0058A0C2	8B11	MOV EDX,DWORD PTR DS:[ECX]	
0058A0C4	52	PUSH EDX	
0058A0C5	FF15 FCB05A00	CALL DWORD PTR DS:[&kernel32.WriteProcessMemory]	kernel32.WriteProcessMemory

_Nothing Interest. Ok, Ctrl + A:

Address	Hex dump	Disassembly	Comment
0058A0A3	<70 07	JL SHORT PictureR.0058A0AC	
0058A0A5	<7C 03	JL SHORT PictureR.0058A0AA	
0058A0A7	>EB 05	JMP SHORT PictureR.0058A0AE	
0058A0A9	E8	DB E8	
0058A0AA	>74 FB	JE SHORT PictureR.0058A0A7	
0058A0AC	>EB F9	JMP SHORT PictureR.0058A0A7	
0058A0AE	>EB 5F	JMP SHORT PictureR.0058A10F	
0058A0B0	> 8055 FC	LEA EDX,DWORD PTR SS:[EBP-4]	
0058A0B3	. 52	PUSH EDX	
0058A0B4	. 6A 02	PUSH 2	
0058A0B6	. 68 6C195B00	PUSH PictureR.005B196C	
0058A0B8	. 8845 10	MOV EAX,DWORD PTR SS:[EBP+10]	
0058A0BE	. 50	PUSH EAX	
0058A0BF	. 884D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
0058A0C2	. 8B11	MOV EDX,DWORD PTR DS:[ECX]	
0058A0C4	. 52	PUSH EDX	
0058A0C5	. FF15 FCB05A00	CALL DWORD PTR DS:[&kernel32.WriteProcessMemory]	WriteProcessMemory
0058A0C8	. 50	PUSH EAX	

_Yeah! Ok, then right click at the buffer and follow in dump> Immediate Constant:



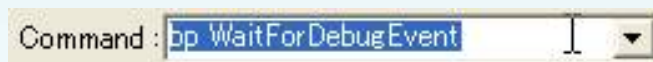
_Look Dump in Windows:

Address	Hex dump	ASCII
005B196C	60 E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00	'.'.....
005B197C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B198C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B199C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19AC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19BC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19CC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19DC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19EC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

_Next, Ctrt + E: 60E8 to change EBFE:

Address	Hex dump	ASCII
005B196C	EB FE 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B197C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B198C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B199C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19AC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19BC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19CC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19DC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B19EC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

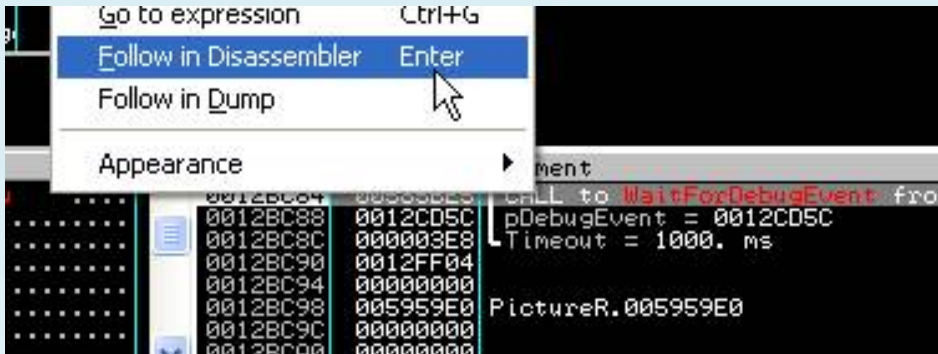
_Following, You set a breakpoint at API WaitForDebugEvent! Alt + F1: BP WaitForDebugEvent (Remember delete the breakpoint at Push ECX)! Attention, set breakpoint before WaitForDebugEvent, you must press F9 one time! If no, the child will be suspend:



_Shift + F9, look in the Stack Window:

Address	Value	Comment
0012BC84	00585BE5	CALL to WaitForDebugEvent from
0012BC88	0012CD5C	pDebugEvent = 0012CD5C
0012BC8C	000003E8	Timeout = 1000. ms
0012BC90	0012FF04	
0012BC94	00000000	
0012BC98	005959E0	PictureR.005959E0
0012BC9C	00000000	
0012BCA0	00000000	
0012BCA4	00000000	

Disassembler _Follow in simple or press Ctrl + F9, F8



_You Still here:

Address	Hex dump	Disassembly	Comment
00585BE5	. 85C0	TEST EAX, EAX	
00585BE7	✓ 0F84 23270000	JE PictureR.00588310	
00585BE0	. 8B95 FCFDFFFF	MOV EDX, DWORD PTR SS:[EBP-204]	
00585BF3	. 81E2 FF000000	AND EDX, 0FF	
00585BF9	. 85D2	TEST EDX, EDX	
00585BFB	✓ 74 12	JE SHORT PictureR.00585C0F	
00585BFD	. A1 701A5B00	MOV EAX, DWORD PTR DS:[5B1A70]	
00585C02	. 8378 20 00	CMPL DWORD PTR DS:[EAX+20], 0	
00585C06	✓ 74 07	JE SHORT PictureR.00585C0F	
00585C08	. C685 FCFDFFFF	MOV BYTE PTR SS:[EBP-204], 0	
00585C0F	> 68 70195B00	PUSH PictureR.005B1970	pCriticalSection = PictureR.0
00585C14	. FF15 A0B15A00	CALL DWORD PTR DS:[<&KERNEL32.EnterCrit	EnterCriticalSection
00585C1A	. 60	PUSHAD	
00585C1B	. 33C0	XOR EAX, EAX	

_Look Attach the Window:

00000704	WINWORD	CiceroUIWndFrame	C:\Program Files\Microsoft Office\OFFICE11\WINWORD.E
00000A00	msmsgs	ActiveMovie Window	C:\Program Files\Messenger\msmsgs.exe
00000A20	PictureR		C:\Program Files\PictureRipper 3\PictureRipper.exe
00000A50	PictureR		C:\Program Files\PictureRipper 3\PictureRipper.exe
00000AFC	MTDSEVER	Socket Notification Sink	C:\Program Files\mtd2002\MTDSEVER.EXE
00000B20	PictureR		C:\Program Files\PictureRipper 3\PictureRipper.exe

_Close Attach the Window, to patch:

Address	Hex dump	Disassembly	Comment
00585BE5	68 500A0000	PUSH 0A50	
00585BEA	E8 44969277	CALL kernel32.DebugActiveProcessStop	
00585BEF	90	NOP	
00585BF0	90	NOP	
00585BF1	90	NOP	
00585BF2	90	NOP	
00585BF3	. 81E2 FF000000	AND EDX, 0FF	

_Press F8, trace down:

Address	Hex dump	Disassembly	Comment	Registers (FPU)
00585BE5	58 50000000	PUSH 0058		EAX 00000001
00585BEA	58 44965277	CALL kernel32.DebugActiveProcessStop		ECX 00120774
00585BF7	58	NOP		EDX 7FFE0304
00585BF8	58	NOP		EBX 005959E0 Pl
00585BF1	58	NOP		ESP 00120C90
00585BF2	58	NOP		EBP 0012077C
00585BF3	58 81E2 FF000000	AND EDX,0FF		ESI 00002710

_Done! By Debug pass blocker successful! Continued, Fire Up a OllyDBG, Attach the child, F9, F12:

Address	Hex dump	Disassembly	Comment
0059B000	59 EB FE	JMP SHORT PictureR.<ModuleEntryPoint>	
0059B002	0000	ADD BYTE PTR DS:[EAX],AL	
0059B004	0000	ADD BYTE PTR DS:[EAX],AL	
0059B006	50	POP EBP	
0059B007	50	PUSH EAX	
0059B008	51	PUSH ECX	
0059B009	0FCA	BSWAP EDX	
0059B00A	5702	NOT EDI	

_Change EBFE to 60E8:

Address	Hex dump	Disassembly	Comment
0059B000	59 60	PUSHAD	
0059B001	59 E8 00000000	CALL PictureR.0059B006	
0059B006	50	POP EBP	
0059B007	50	PUSH EAX	
0059B008	51	PUSH ECX	
0059B009	0FCA	BSWAP EDX	
0059B00A	F702	NOT EDI	

_The Next step is magic patch jump and find OEP. Ok, now, press Alt + F1: He GetModuleHandleA, Shift + F9 first time:

Address	Value	Comment
00127B70	00ED19C0	CALL to GetModuleHandleA from
00127B74	00EE6364	pModule = "kernel32.dll"
00127B78	00EE7588	ASCII "VirtualAlloc"
00127B7C	00000001	
00127B80	00F40090	
00127B84	00000000	
00127B88	00000000	
00127B8C	00000000	

_ 2 nd:

Address	Value	Comment
00127B70	00ED19E9	CALL to GetModuleHandleA from
00127B74	00EE6364	pModule = "kernel32.dll"
00127B78	00EE757C	ASCII "VirtualFree"
00127B7C	00000001	
00127B80	00F40090	
00127B84	00000000	
00127B88	00000000	
00127B8C	00000000	
00127B90	00000000	

_3 Rd:

Address	Value	Comment
001278E0	00EB9BA7	CALL to GetModuleHandleA from
001278E4	00127A24	pModule = "kernel32.dll"
001278E8	00000000	
001278EC	CE300000	
001278F0	41F80012	
001278F4	00EE57C4	
001278F8	00000000	
001278FC	00000000	
00127900	00000000	

_Trace Down F8 with the Return and you here at:

Address	Hex dump	Disassembly	Comment
00EB9BA7	8B0D 74B7EE00	MOV ECX,DWORD PTR DS:[EBB774]	
00EB9BA0	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00EB9BB0	A1 74B7EE00	MOV EAX,DWORD PTR DS:[EBB774]	
00EB9BB5	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00EB9BB8	75 16	JNZ SHORT 00EB9BD0	
00EB9BBA	8D85 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
00EB9BC0	50	PUSH EAX	
00EB9BC1	FF15 DC00EE00	CALL DWORD PTR DS:[EE00DC]	kernel32.LoadLibraryA
00EB9BC7	8B0D 74B7EE00	MOV ECX,DWORD PTR DS:[EBB774]	
00EB9BCD	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00EB9BD0	A1 74B7EE00	MOV EAX,DWORD PTR DS:[EBB774]	
00EB9BD5	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00EB9BD8	75 16	JNZ SHORT 00EB9BD0	
00EB9BDE	33C9	XOR ECX,ECX	
00EB9BE0	8B07	MOV EAX,DWORD PTR DS:[EDI]	
00EB9BE2	3918	CMP DWORD PTR DS:[EAX],EBX	
00EB9BE4	74 06	JE SHORT 00EB9BEC	
00EB9BE6	41	INC ECX	
00EB9BE7	83C0 0C	ADD EAX,0C	
00EB9BEA	EB F6	JMP SHORT 00EB9BE2	

kernel32.LoadLibraryA

The magic jump!

_Change It to EB:

Address	Hex dump	Disassembly	Comment
00EB9BA7	8B0D 74B7EE00	MOV ECX,DWORD PTR DS:[EBB774]	
00EB9BA0	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00EB9BB0	A1 74B7EE00	MOV EAX,DWORD PTR DS:[EBB774]	
00EB9BB5	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00EB9BB8	75 16	JNZ SHORT 00EB9BD0	
00EB9BBA	8D85 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
00EB9BC0	50	PUSH EAX	
00EB9BC1	FF15 DC00EE00	CALL DWORD PTR DS:[EE00DC]	kernel32.LoadLibraryA
00EB9BC7	8B0D 74B7EE00	MOV ECX,DWORD PTR DS:[EBB774]	
00EB9BCD	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
00EB9BD0	A1 74B7EE00	MOV EAX,DWORD PTR DS:[EBB774]	
00EB9BD5	391C06	CMP DWORD PTR DS:[ESI+EAX],EBX	
00EB9BD8	E9 39010000	JMP 00EB9D16	
00EB9BD0	50	PUSH EAX	
00EB9BDE	33C9	XOR ECX,ECX	
00EB9BE0	8B07	MOV EAX,DWORD PTR DS:[EDI]	
00EB9BE2	3918	CMP DWORD PTR DS:[EAX],EBX	

_Now We find J OEP. Hd GetModuleHandleA, BP CreateThread, Shift + F9:

Address	Value	Comment
0012D744	00EBFEE0	CALL to CreateThread from 00B
0012D748	00000000	pSecurity = NULL
0012D74C	00000000	StackSize = 0
0012D750	00EC070C	ThreadFunction = 00EC070C
0012D754	00000000	pThreadParm = NULL
0012D758	00000000	CreationFlags = 0
0012D75C	0012D764	pThreadId = 0012D764
0012D760	005B1568	PictureR.005B1568

_Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
00EBFEE0	5E	POP ESI	PictureR.005B1568
00EBFEE1	C9	LEAVE	
00EBFEE2	C3	RETN	
00EBFEE3	55	PUSH EBP	
00EBFEE4	8BEC	MOV EBP,ESP	
00EBFEE6	81EC 28010000	SUB ESP,128	
00EBFEEC	56	PUSH ESI	
00EBFEED	57	PUSH EDI	
00EBFEEF	BE F063EE00	MOV ESI,0EE63F0	ASCII "MainClass"
00EBFEF3	8B0D 08FFFFFF	LEA EDI,DWORD PTR SS:[EBP-128]	

_Again, Ctrl + F9, F8, Scroll down the signal of OEP:

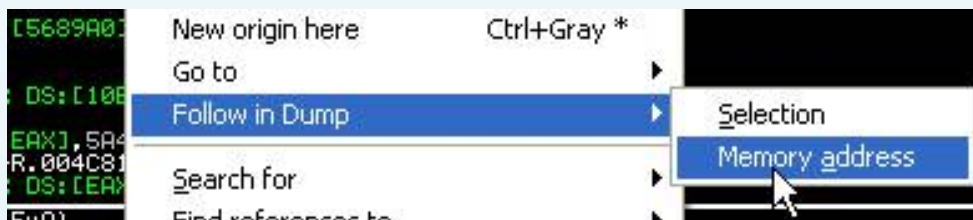

```

00EDA2A0 50          PUSH EAX
00EDA2A1 A1 2800EF00 MOV EAX,DWORD PTR DS:[EF0028]
00EDA2A6 8B48 70     MOV ECX,DWORD PTR DS:[EAX+70]
00EDA2A9 3348 4C     XOR ECX,DWORD PTR DS:[EAX+4C]
00EDA2AC 3348 40     XOR ECX,DWORD PTR DS:[EAX+40]
00EDA2AF 2BF9       SUB EDI,ECX
00EDA2B1 FFD7       CALL EDI
00EDA2B3 8BD8       MOV EBX,EAX
00EDA2B5 5F         POP EDI
00EDA2B6 8BC3       MOV EAX,EBX
00EDA2B8 5E         POP ESI
00EDA2B9 5B         POP EBX
00EDA2BA C3         RETN
00EDA2BB 837C24 08 01 CMP DWORD PTR SS:[ESP+8],1
    
```

_Press Call EDI at F2, F9, F7: OEP J !

Address	Hex dump	Disassembly	Comment
004C80A2	6A 60	PUSH 60	
004C80AC	68 88A35300	PUSH PictureR.0053A388	
004C80B1	E8 CE230000	CALL PictureR.004CA484	
004C80B6	BF 94000000	MOV EDI,94	
004C80BB	8BC7	MOV EAX,EDI	
004C80BD	E8 7EDCFFFF	CALL PictureR.004C5D40	
004C80C2	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
004C80C5	8BF4	MOV ESI,ESP	
004C80C7	893E	MOV DWORD PTR DS:[ESI],EDI	
004C80C9	56	PUSH ESI	
004C80CA	FF15 2C270E01	CALL DWORD PTR DS:[10E272C]	kernel32.GetVersionExA
004C80D0	8B4E 10	MOV ECX,DWORD PTR DS:[ESI+10]	
004C80D3	890D 98895600	MOV DWORD PTR DS:[568998],ECX	
004C80D9	8B46 04	MOV EAX,DWORD PTR DS:[ESI+4]	
004C80DC	A3 A4895600	MOV DWORD PTR DS:[5689A4],EAX	
004C80E1	8B56 08	MOV EDX,DWORD PTR DS:[ESI+8]	
004C80E4	8915 A8895600	MOV DWORD PTR DS:[5689A8],EDX	

_This Is the signal of Import elimination. We must redirect it to a real cave memory! But the first step is to find the IAT. At the API GetVetsionA, Follow in dump> Memory Address, and we have the IAT:



_IAT Start:

Address	Value	Comment
010E24B4	695B6F5C	
010E24B8	57725873	
010E24BC	77637968	SHELL32.77637968
010E24C0	424A5A52	
010E24C4	524E6956	
010E24C8	6D4B7746	
010E24CC	54417170	
010E24D0	7D7D504F	
010E24D4	45622355	

_IAT End:

Address	Value	Comment
010E2FE0	77C992B2	GDI32.ScaleWindowExtEx
010E2FE4	77E72F4B	kernel32.SetFileTime
010E2FE8	00EBADE9	
010E2FEC	00000000	
010E2FF0	01540002	
010E2FF4	000F1102	
010E2FF8	000B0188	
010E2FFC	000B0188	

_Total We have:

IAT Start: 010E24BC 77637968 SHELL32.77637968

IAT End: 010E2FE4 77E72F4B kernel32.SetFileTime

IAT Len: 010E2FE4-010E24BC = B28

OEP: C80AA

PID: 07F8

_Is Now, we must redirect the IAT to a Virtual Real IAT. ArmInline fire up and fill in:

PID: 0A50

Start Of Target Code: 401000: this is the address sections of text:

003F0000	00002000			Map	R	E	R	E
00400000	00001000	PictureR		Imag	R		RWE	
00401000	00101000	PictureR	.text	Imag	R		RWE	
00502000	0004F000	PictureR	.rdata	Imag	R		RWE	
00551000	00010000	PictureR	.data	Imag	R		RWE	

Length Of Target Code: 101000: Size of text sections:

(Slave) Process ID: 0x	A50
Start Of Target Code: 0x	401000
Length Of Target Code: 0x	101000

_In The table IAT elimination fill in:

Base Of Existing IAT: IAT Start: 010E24BC 77637968 SHELL32.77637968

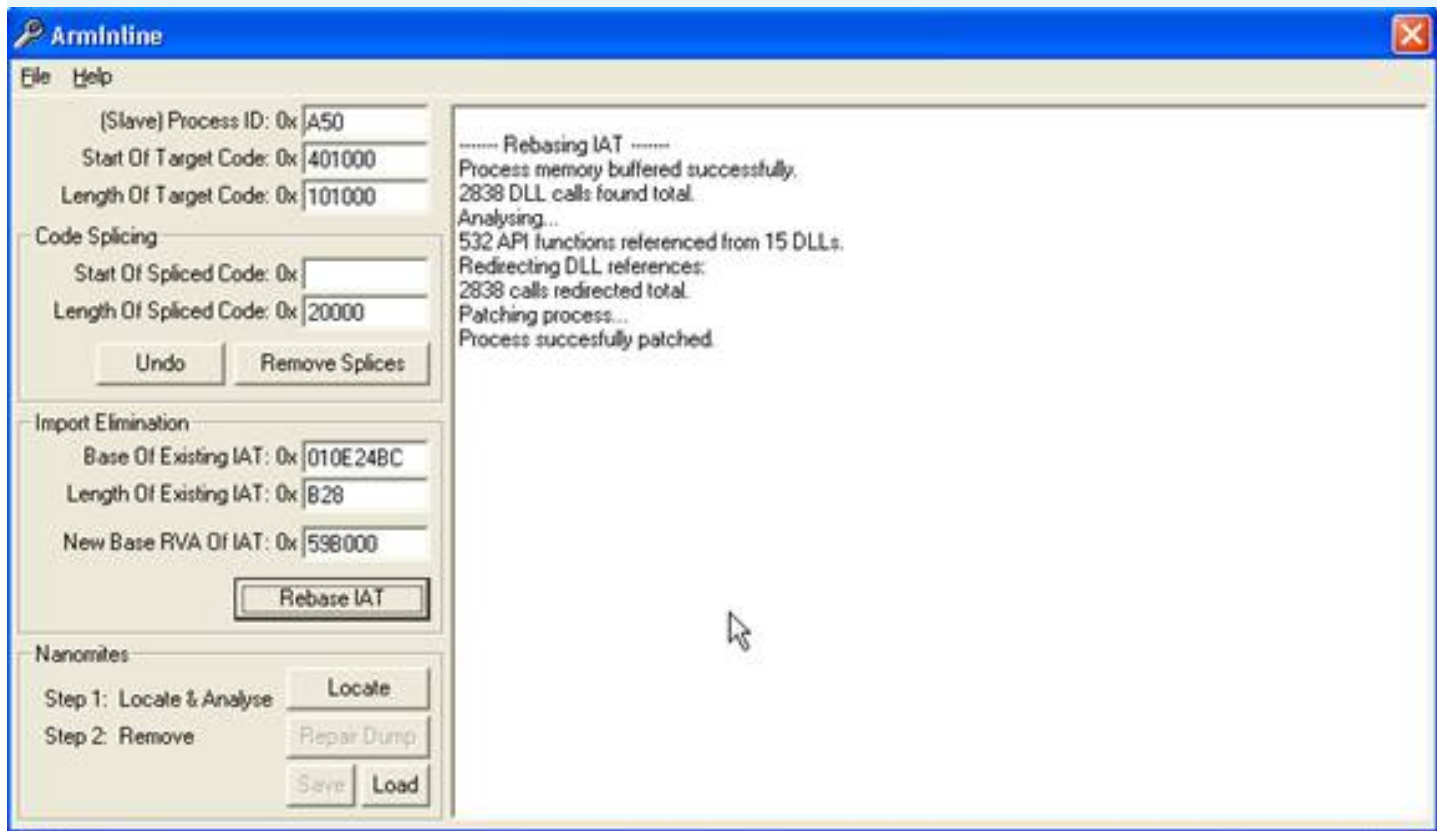
Length Of Existing IAT: IAT Len: 010E2FE4-010E24BC = B28

New Base RVA of IAT: We choose the section. ADATA to redirect the IAT: 59B000

0056B000	00030000	PictureR	.text1	code
0059B000	00010000	PictureR	.adata	code
005AB000	00020000	PictureR	.data1	data, impo

Import Elimination	
Base Of Existing IAT: 0x	010E24BC
Length Of Existing IAT: 0x	B28
New Base RVA Of IAT: 0x	59B000
Rebase IAT	

_Ok, Done, click Rebase IAT, and here we are:



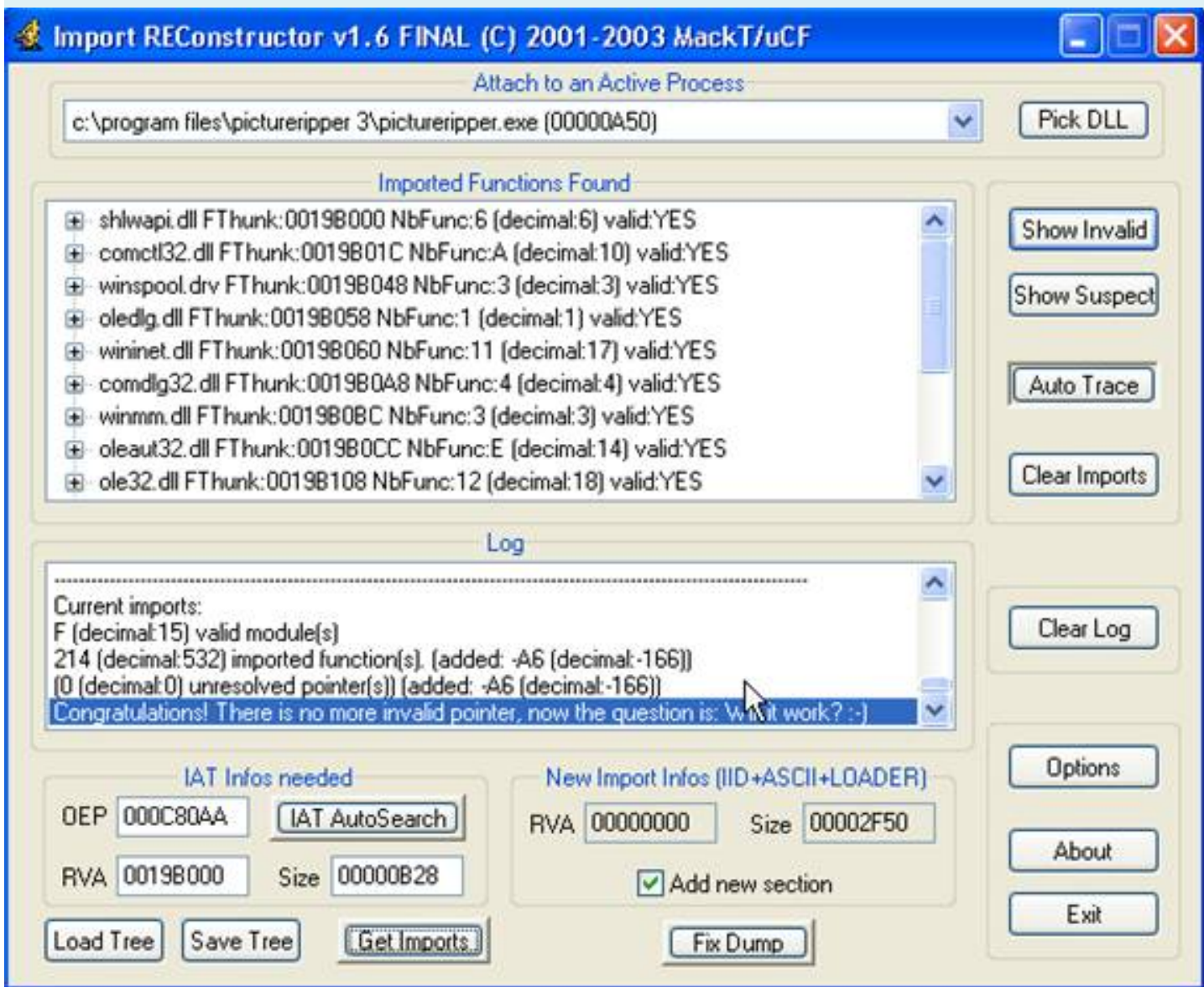
_Go Back to the CPU, and we see the IAT is successful redirect:

Address	Hex dump	Disassembly	Comment
004C80A8	6A 60	PUSH 60	
004C80AC	68 88A35300	PUSH PictureR.0053A388	
004C80B1	E8 CE230000	CALL PictureR.004CA484	
004C80B6	BF 94000000	MOV EDI,94	
004C80B8	8BC7	MOV EAX,EDI	
004C80BD	E8 7EDCFFFF	CALL PictureR.004C5D40	
004C80C2	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
004C80C5	8BF4	MOV ESI,ESP	
004C80C7	893E	MOV DWORD PTR DS:[ESI],EDI	
004C80C9	56	POP ESI	
004C80CA	FF15 E0B75900	CALL DWORD PTR DS:[59B7E0]	kernel32.GetVersionExA
004C80D0	894E 10	MOV ECX,DWORD PTR DS:[ESI+10]	
004C80D3	890D 98895600	MOV DWORD PTR DS:[568998],ECX	
004C80D9	8B46 04	MOV EAX,DWORD PTR DS:[ESI+4]	
004C80DC	A3 A4895600	MOV DWORD PTR DS:[5689A4],EAX	
004C80E1	8B56 08	MOV EDX,DWORD PTR DS:[ESI+8]	
004C80E4	8915 A8895600	MOV DWORD PTR DS:[5689A8],EDX	
004C80EA	8B76 0C	MOV ESI,DWORD PTR DS:[ESI+C]	
004C80ED	81E6 FF7F0000	AND ESI,7FFF	
004C80F3	8935 9C895600	MOV DWORD PTR DS:[56899C],ESI	
004C80F9	83F9 02	CMPL ECX,2	
004C80FC	74 0C	JE SHORT PictureR.004C810A	
004C80FE	81CE 00800000	OR ESI,8000	
004C8104	8935 9C895600	MOV DWORD PTR DS:[56899C],ESI	
004C810A	C1E0 08	SHL EAX,8	
004C810D	03C2	ADD EAX,EDX	
004C810F	A3 A0895600	MOV DWORD PTR DS:[5689A0],EAX	
004C8114	33F6	XOR ESI,ESI	
004C8116	56	PUSH ESI	
004C8117	8B3D A4B75900	MOV EDI,DWORD PTR DS:[59B7A4]	kernel32.GetModuleHandleA
004C811D	FFD7	CALL EDI	
004C811F	66 8130 1050	CMPL WORD PTR DS:[EAX],5040	
004C8124	75 1F	JNZ SHORT PictureR.004C8145	
004C8126	8B48 3C	MOV ECX,DWORD PTR DS:[EAX+3C]	

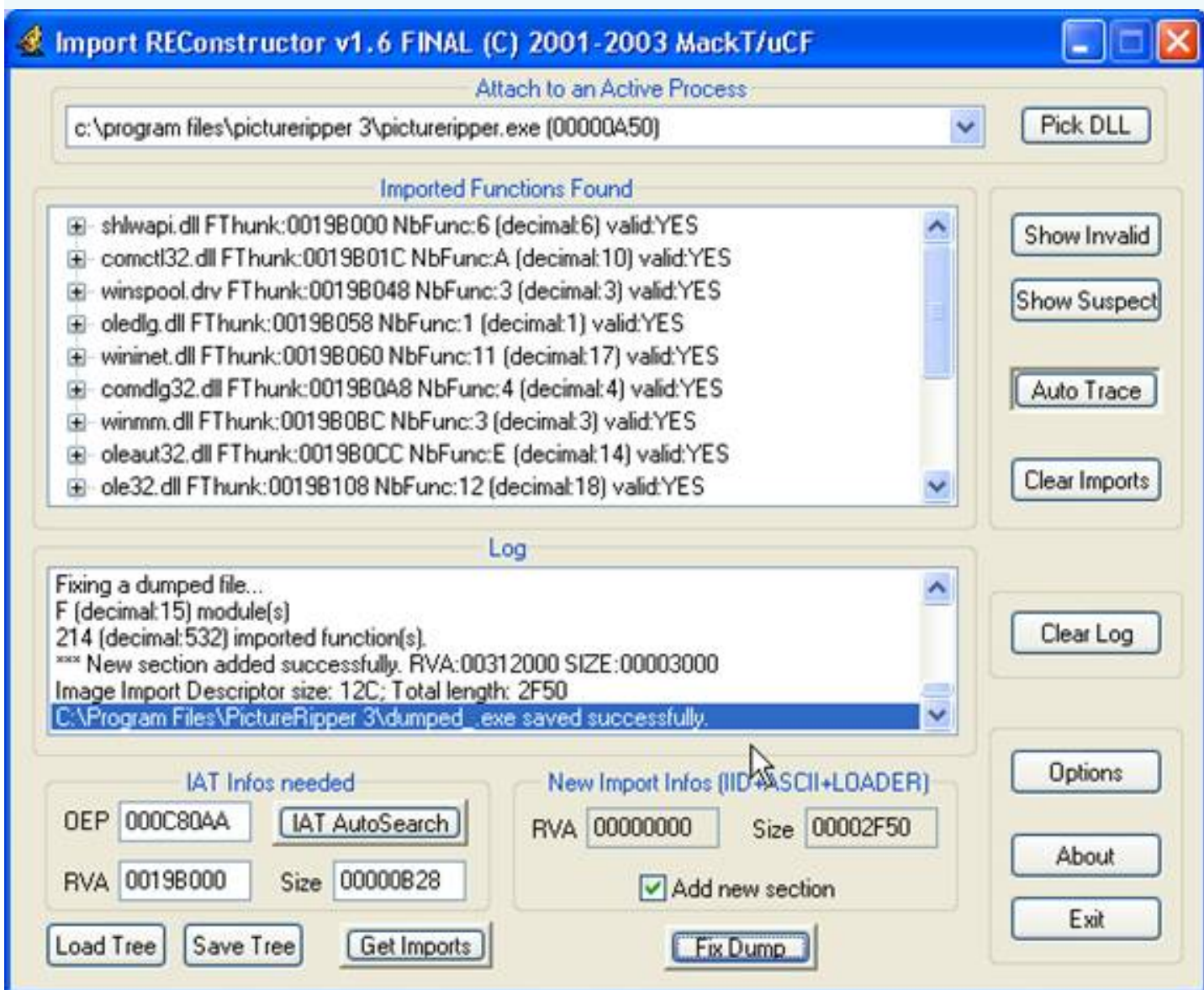
_Use LordPE and Full dump:

c:\program files\pictureripper 3\pictureripper....	00000A20	00400000	00312000
c:\program files\pictureripper 3\pictureripper....			00312000
i:\cracker\debug- disassembler\odbg...			00164000
c:\program files\emeditor\emeditor.exe			00069000
			00010000

_Fire Up Imprec, fill OEP, IAT Auto Search, GetImport, Show Invalid, Thanks Cuts:

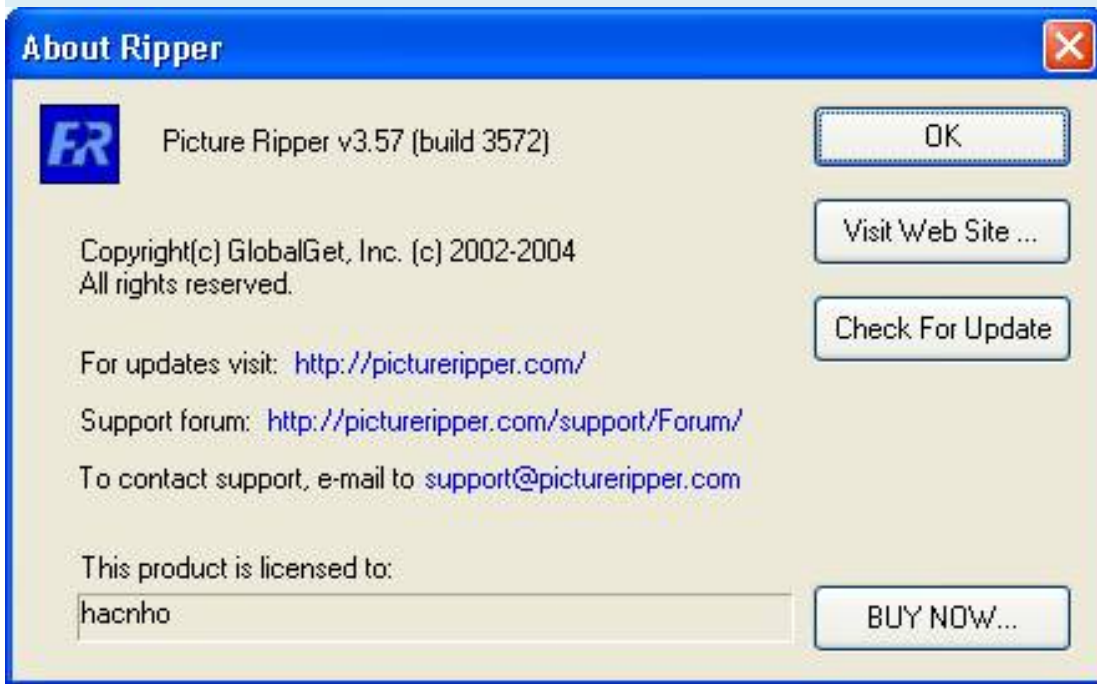


_Fix Dump:

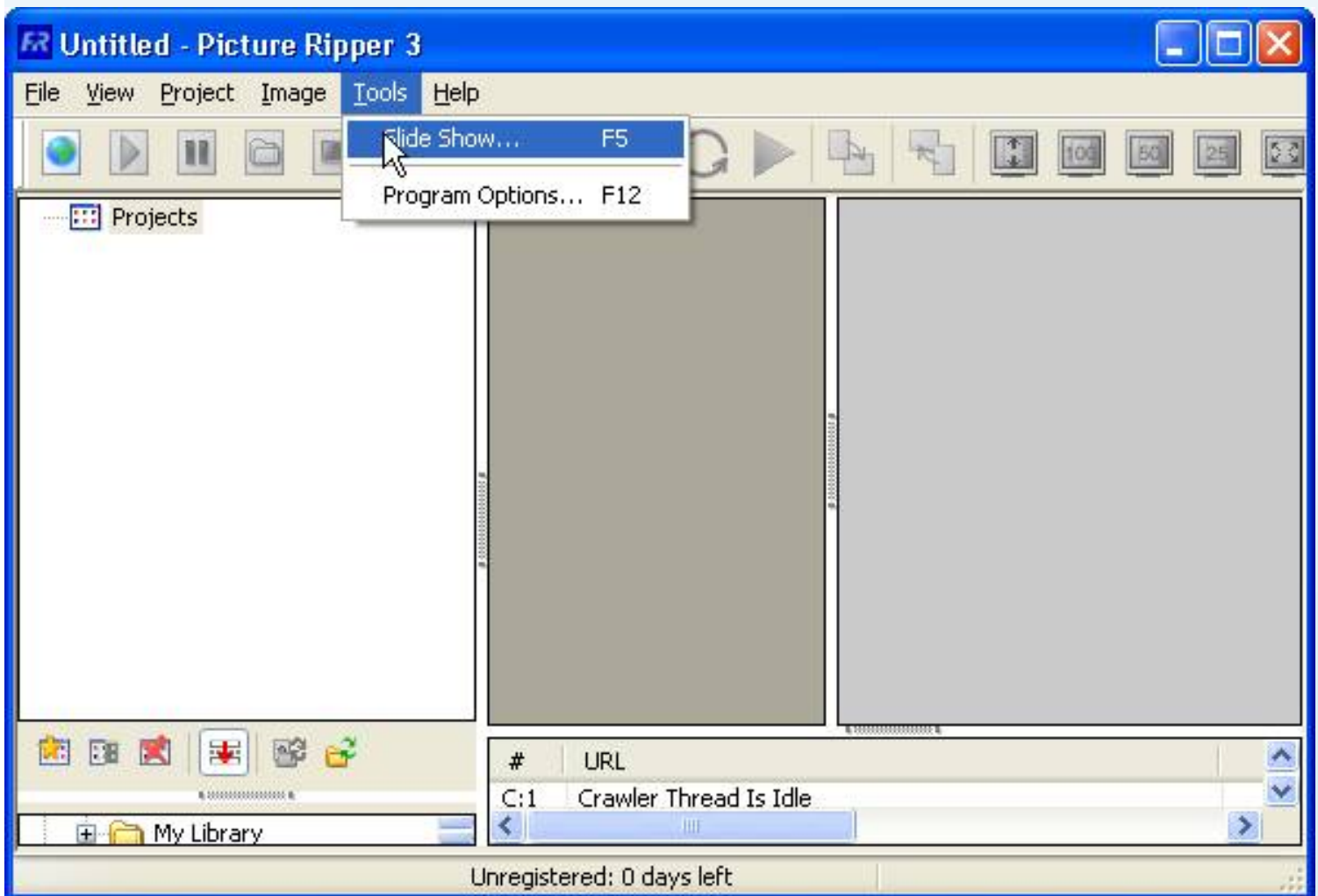


Run Dumped.exe:

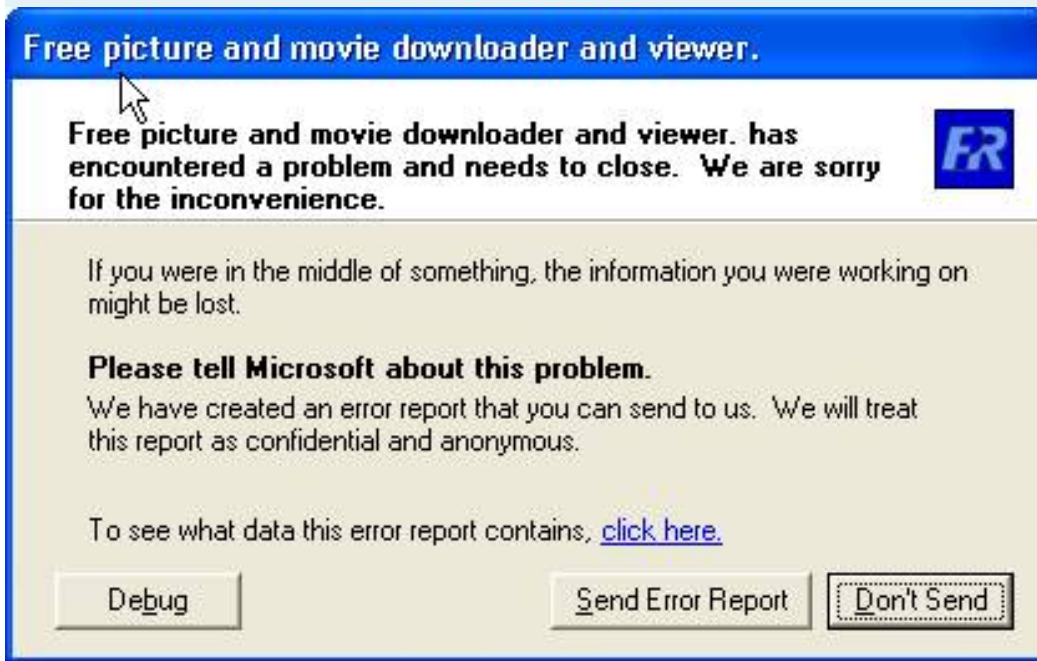




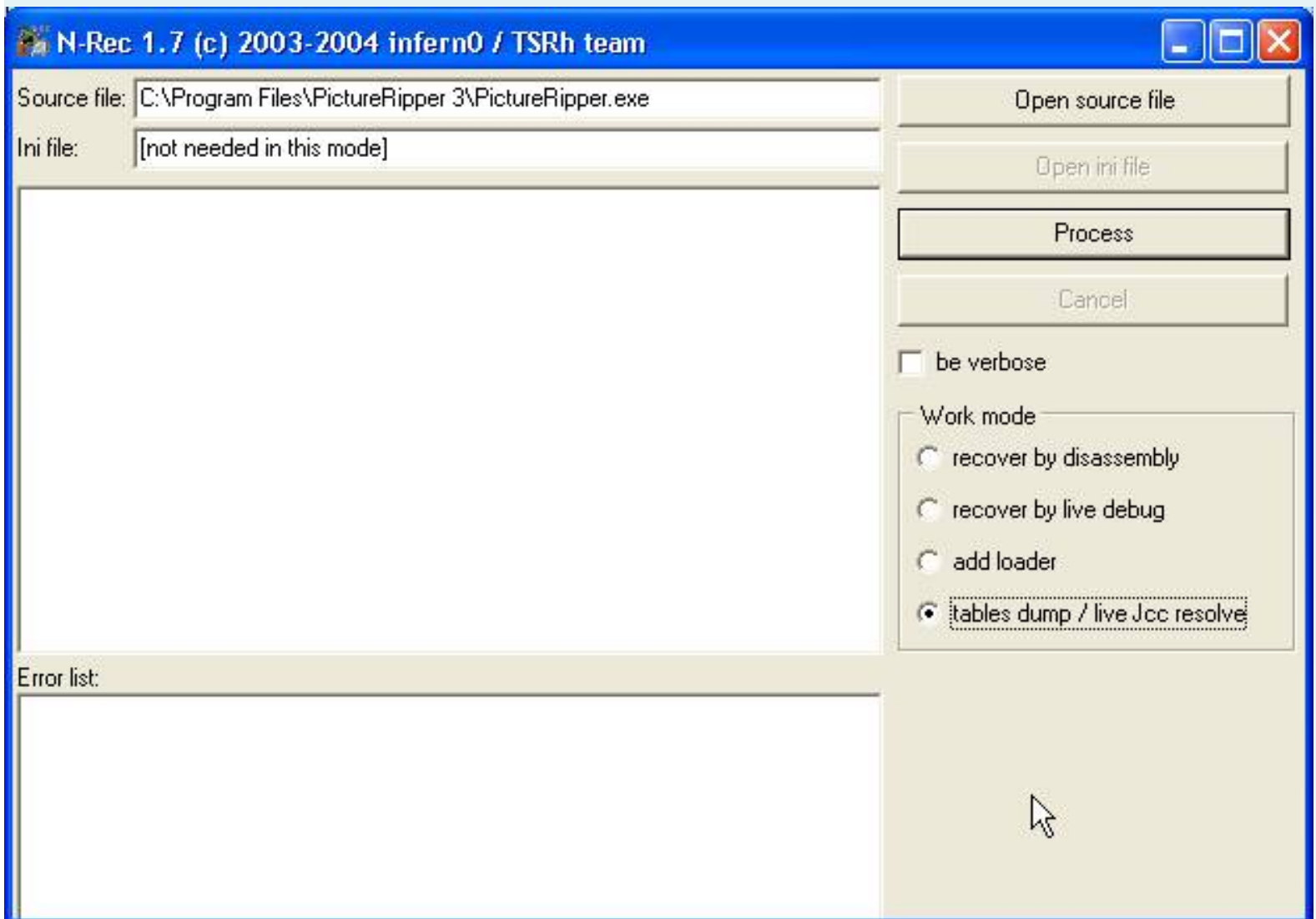
_So When we try to open the Options menu:



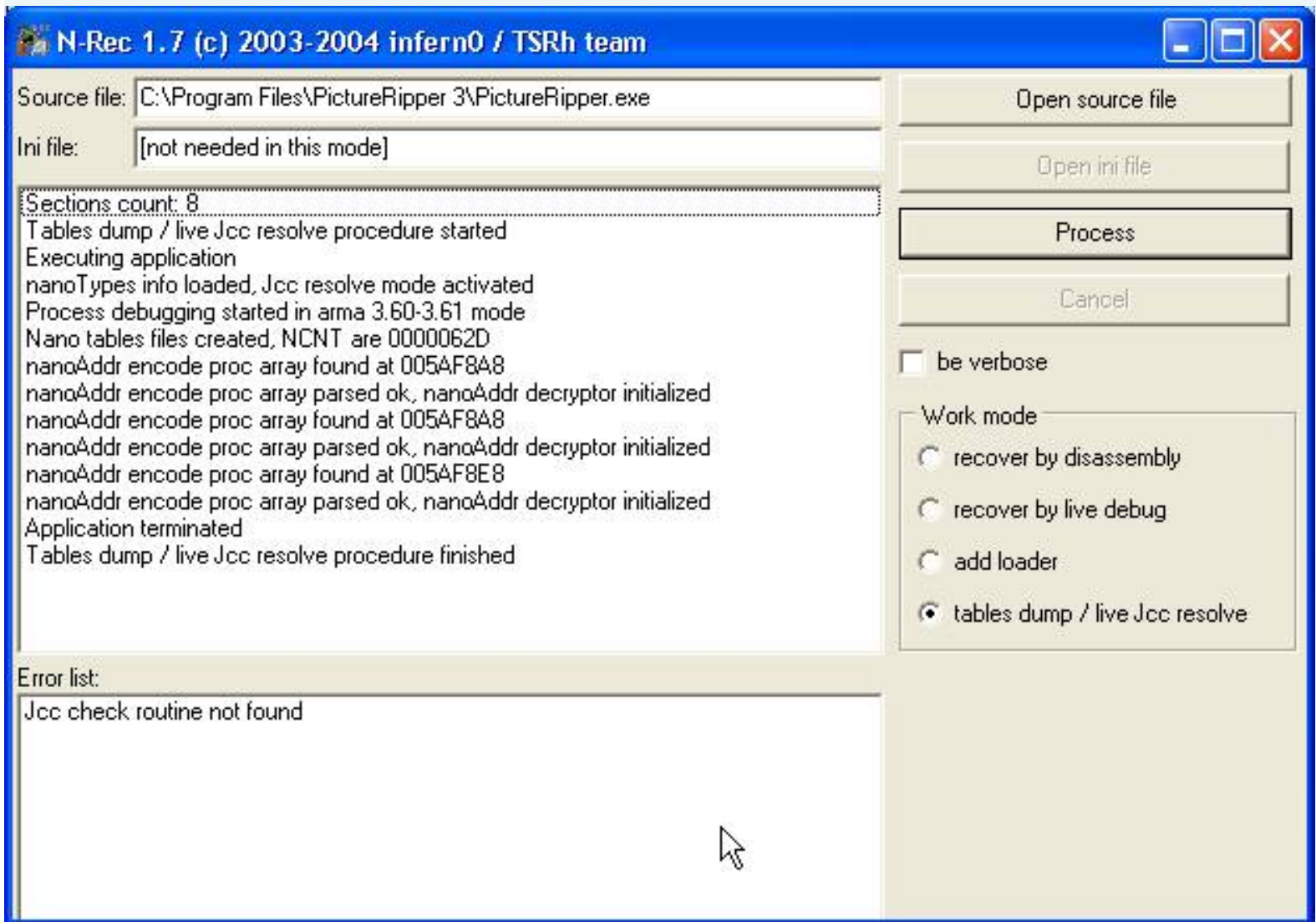
_We Crash:



_Ok, Le analyze this crash. Load N-REC to:



_And We found the CC:



We Use ArmInline 0.71 to fix Nanomites, ok, to load dumped.exe OllyDBG:

hacnho - dumped_.exe - [CPU - main thread, module dumped_]

File View Debug Plugins Options Window Help

Paused

Address	Hex dump	Disassembly	Registers (FPU)
004C80AA	6A 60	push 60	EAX 00000000
004C80AC	68 88A35300	push dumped_.0053A388	ECX 0012FFB0
004C80B1	E8 CE230000	call dumped_.004CA484	EDX 7C90EB94 ntdll.K
004C80B6	BF 94000000	mov edi,94	EBX 7FFDE000
004C80BB	8BC7	mov eax,edi	ESP 0012FFC4
004C80BD	E8 7EDCFFFF	call dumped_.004C5D40	EBP 0012FFFO
004C80C2	8965 E8	mov dword ptr ss:[ebp-18],esp	ESI FFFFFFFF
004C80C5	8BF4	mov esi,esp	EDI 7C910738 ntdll.7
004C80C7	893E	mov dword ptr ds:[esi],edi	EIP 004C80AA dumped_
004C80C9	56	push esi	C 0 ES 0023 32bit 0
004C80CA	FF15 E4B75900	call dword ptr ds:[<kernel32.GetVersio	P 1 CS 001B 32bit 0
004C80D0	8B4E 10	mov ecx,dword ptr ds:[esi+10]	A 0 SS 0023 32bit 0
004C80D3	890D 98895600	mov dword ptr ds:[568998],ecx	Z 1 DS 0023 32bit 0
004C80D9	8B46 04	mov eax,dword ptr ds:[esi+4]	S 0 FS 003B 32bit 7
004C80DC	A3 A4895600	mov dword ptr ds:[5689A4],eax	T 0 GS 0000 NULL
004C80E1	8B56 08	mov edx,dword ptr ds:[esi+8]	D 0 LastErr ERROR_S

Address	Hex dump	ASCII	Address	Value	Comment
005AB000	5F 3A C7 77 80 70 C7 77	_: 惹6p惹	0012FFC4	7C816D4F	RETURN to kernel32.7
005AB008	8E 4E C7 77 5C 85 C7 77	藏惹\吳w	0012FFC8	7C910738	ntdll.7C910738
005AB010	1A 5C C7 77 07 36 C7 77	□\惹□6惹	0012FFCC	FFFFFFFF	
005AB018	6E 3C C7 77 98 1B C7 77	n<惹?惹	0012FFD0	7FFDE000	
005AB020	E0 20 C7 77 6C 5B C7 77	?惹1[惹	0012FFD4	8054B038	
005AB028	00 00 00 00 9B A2 E7 77决壁	0012FFD8	0012FFC8	
005AB030	63 DE E6 77 86 AD E7 77	c咳w喘壁	0012FFDC	85665B40	

Command :

Program entry point

_View PID and text sections:

Select process to attach

Process	Name	Window	Path
00000330	svchost		C:\WINDOWS\system32\svchost.exe
00000358	svchost		C:\WINDOWS\System32\svchost.exe
0000038C	svchost		C:\WINDOWS\system32\svchost.exe
000003E0	svchost		C:\WINDOWS\system32\svchost.exe
00000444	spoolsv		C:\WINDOWS\system32\spoolsv.exe
0000048C	TurboLaunch	TurboLaunch	C:\Program Files\TurboLaunch\TurboLaunch.exe
00000540	Explorer	SysFader	C:\WINDOWS\Explorer.EXE
0000058C	UniKeyNT	UniKey 3.62	C:\Program Files\UniKey\UniKeyNT.exe
00000598	ctfmon	TF_FloatingLangBar_WndTitl	C:\WINDOWS\system32\ctfmon.exe
000005A4	sqlmangr	SQL Server Service Manager	C:\Program Files\Microsoft SQL Server\
000005F0	DkService		C:\Program Files\Executive Software\Di
00000630	MDM		C:\Program Files\Common Files\Microsof
000006AC	sqlservr		C:\PROGRAM1\MI6841\1\MSSQL\bin\sqlser
00000708	StarWind		C:\Program Files\Alcohol Soft\Alcohol
00000730	wdfmgr		C:\WINDOWS\system32\wdfmgr.exe
00000990	WINWORD	CiceroUIWndFrame	C:\Program Files\Microsoft Office\OFFI
000009F4	Snagit32	Snagit Capture Preview	C:\Program Files\TechSmith\Snagit 7\Sn
000009FC	TSCHelp	C:\Program Files\TechSmith	C:\Program Files\TechSmith\Snagit 7\TS
00000FF8	dumped_		C:\Program Files\PictureRipper 3\dump

Attach

Cancel

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00150000	00020000				Priv	RW	RW	
00250000	00006000				Priv	RW	RW	
00260000	00003000				Map	RW	RW	
00270000	00016000				Map	R	R	\Device\HarddiskVolume5\WIND
00290000	00030000				Map	R	R	\Device\HarddiskVolume5\WIND
002D0000	00041000				Map	R	R	\Device\HarddiskVolume5\WIND
00320000	00006000				Map	R	R	\Device\HarddiskVolume5\WIND
00330000	00041000				Map	R	R	
00380000	00001000				Priv	RWE	RWE	
00390000	00001000				Priv	RWE	RWE	
003A0000	00004000				Priv	RW	RW	
003B0000	00003000				Map	R	R	\Device\HarddiskVolume5\WIND
003C0000	00008000				Priv	RW	RW	
003D0000	00001000				Priv	RW	RW	
003E0000	00001000				Priv	RW	RW	
003F0000	00003000				Priv	RW	RW	
00400000	00001000	dumped_		PE header	Imag	R	RWE	
00401000	00101000	dumped_	.text		Imag	R	RWE	
00502000	0004F000	dumped_	.rdata		Imag	R	RWE	
00551000	0001A000	dumped_	.data		Imag	R	RWE	
0056B000	00030000	dumped_	.text1	code	Imag	R	RWE	
0059B000	00010000	dumped_	.adata	code	Imag	R	RWE	
005AB000	00020000	dumped_	.data1	data	Imag	R	RWE	
005CB000	000F0000	dumped_	.pdata		Imag	R	RWE	
006BB000	00057000	dumped_	.rsrc	resources	Imag	R	RWE	
00712000	00003000	dumped_	.mactt	imports	Imag	R	RWE	
00720000	00005000				Map	R E	R E	
007E0000	00002000				Map	R E	R E	
007F0000	00103000				Map	R	R	
00900000	000A4000				Map	R E	R E	
00C00000	00002000				Map	R	R	
00C10000	00002000				Map	R	R	
00C20000	00001000				Priv	RW	RW	
00CC0000	00002000				Map	R	R	
629C0000	00001000	LPK		PE header	Imag	R	RWE	
629C1000	00005000	LPK	.text	code, import	Imag	R	RWE	
629C6000	00001000	LPK	.data	data	Imag	R	RWE	
629C7000	00001000	LPK	.rsrc	resources	Imag	R	RWE	
629C8000	00001000	LPK	.reloc	relocations	Imag	R	RWE	
72000000	00001000	LPK		PE header	Imag	R	RWE	

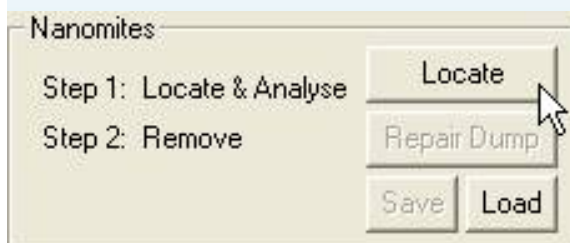
_Fill In ArmInline 0.71:

(Slave) Process ID: 0x FF8

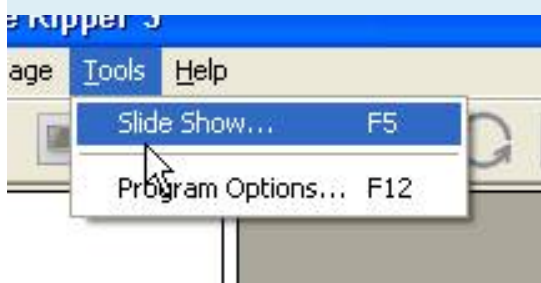
Start Of Target Code: 0x 401000

Length Of Target Code: 0x 101000

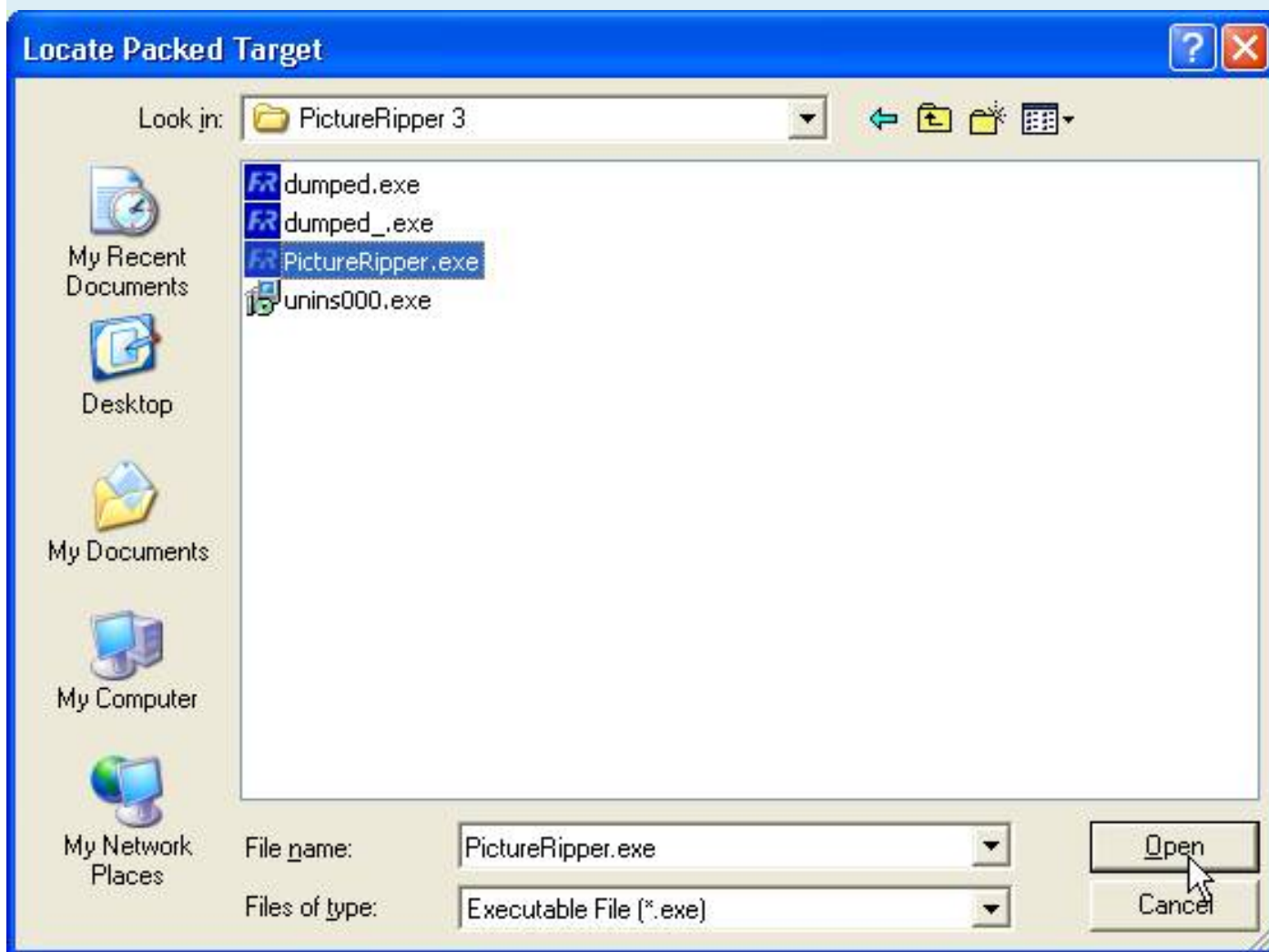
_Click Locate:



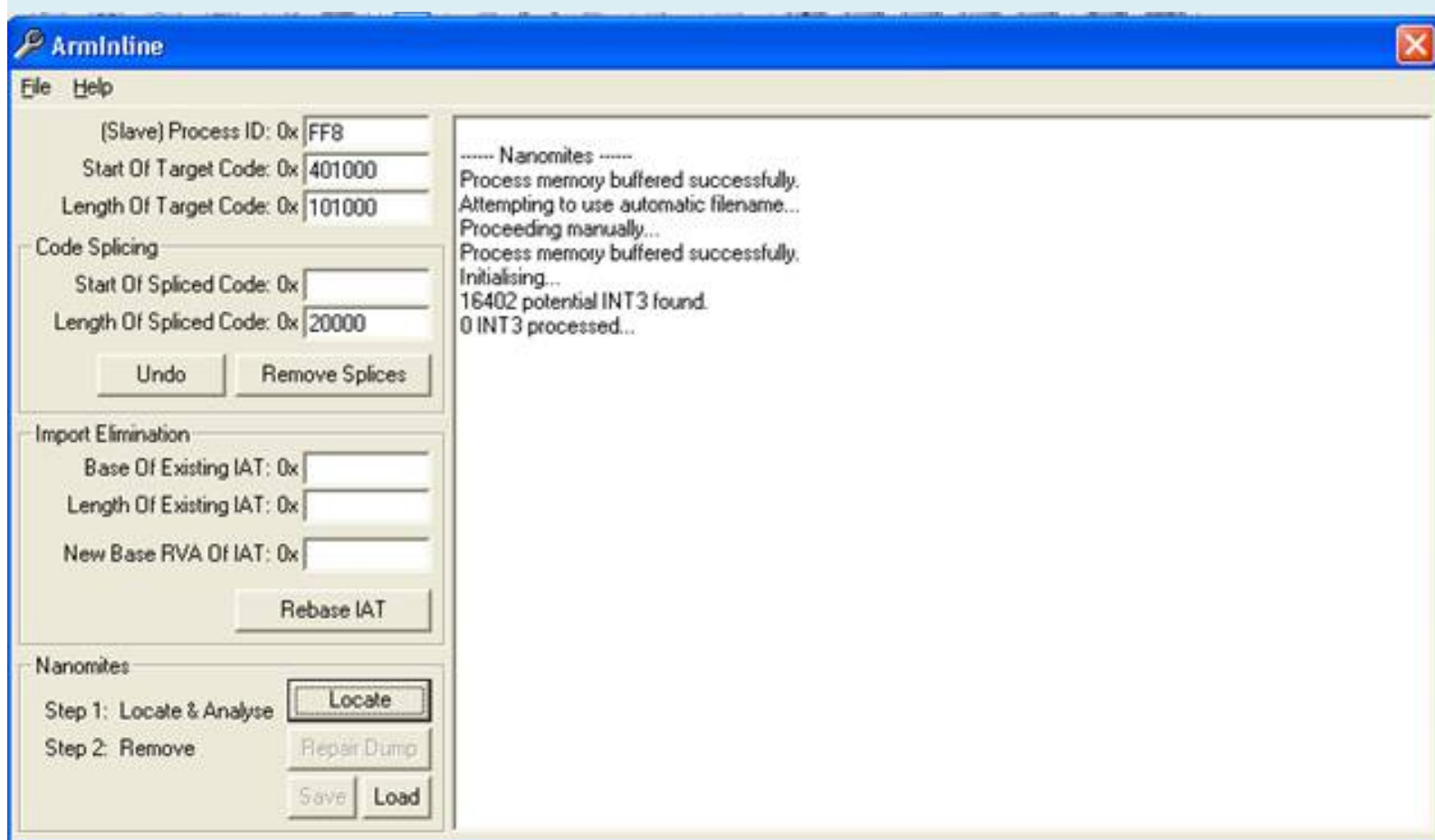
_Then Back to PictureRipper3:



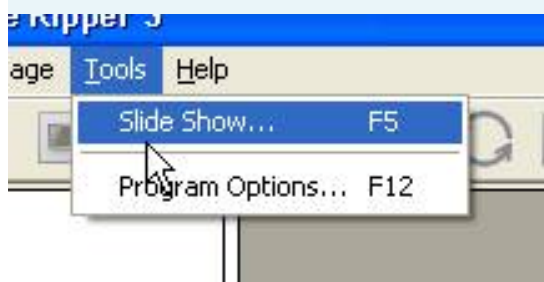
_Now In ArmInline: Load packet target:



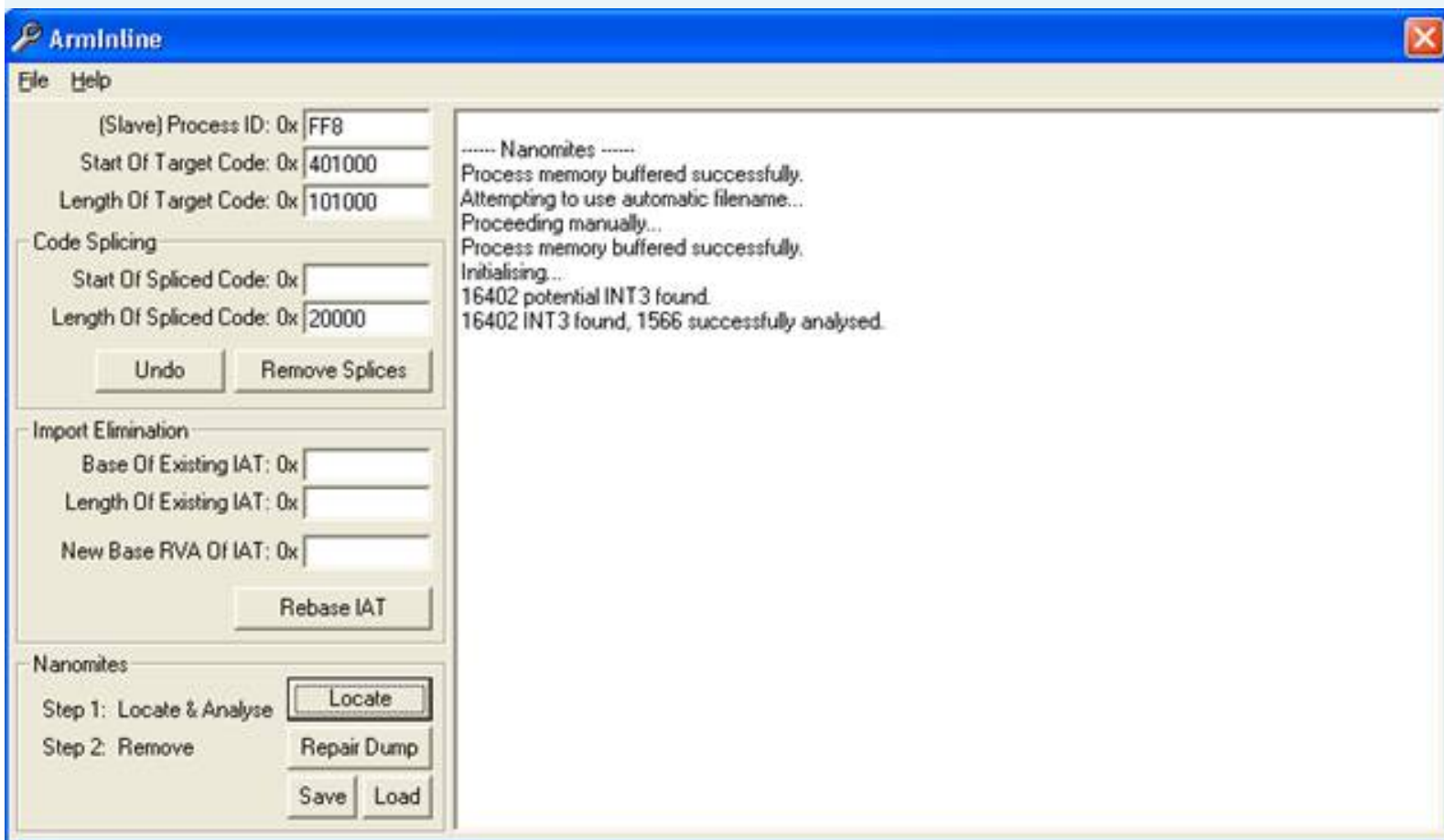
_We Choose the file PictureRipper.exe then:



_Back To PictureRipper:



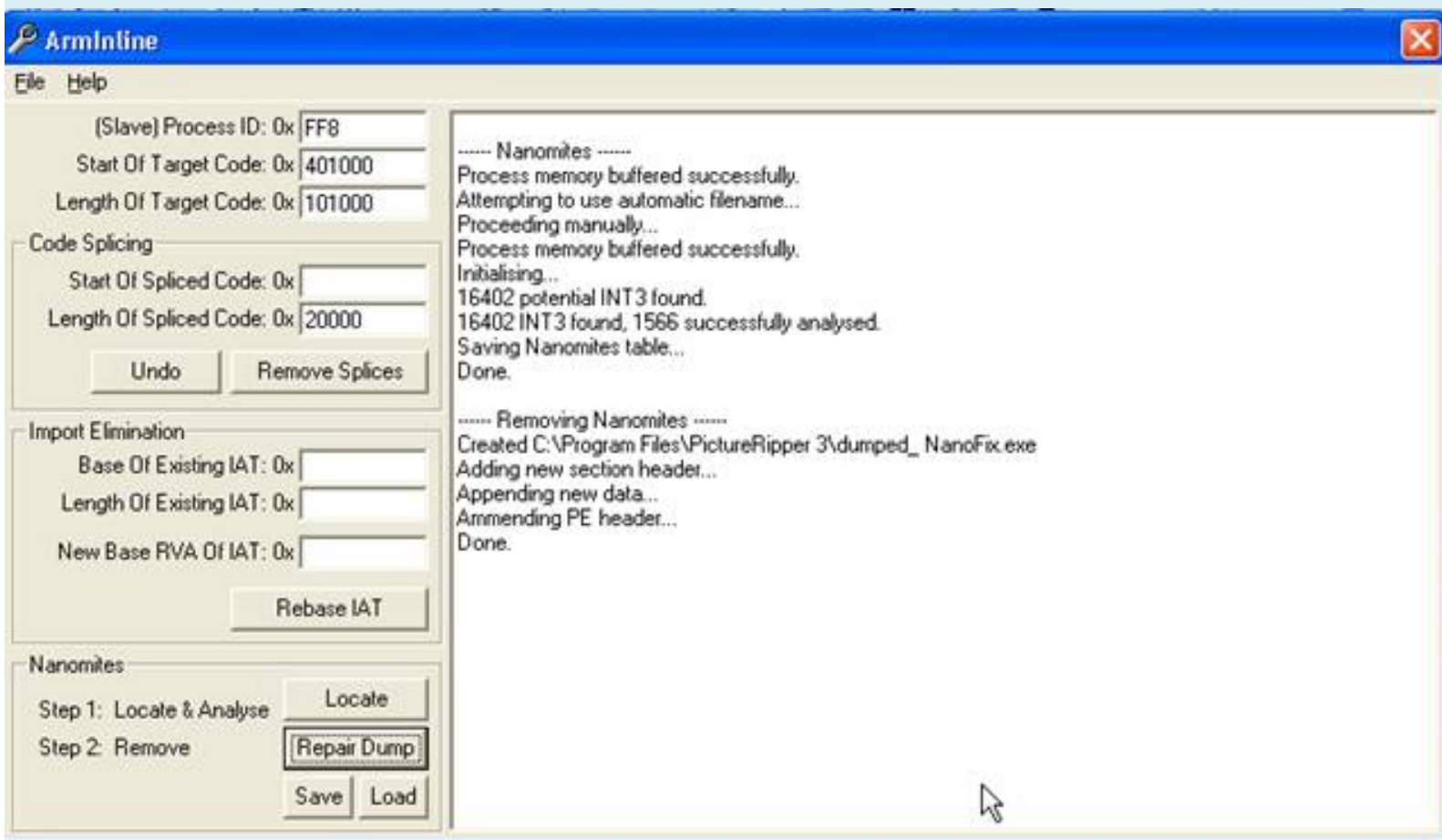
_Now:



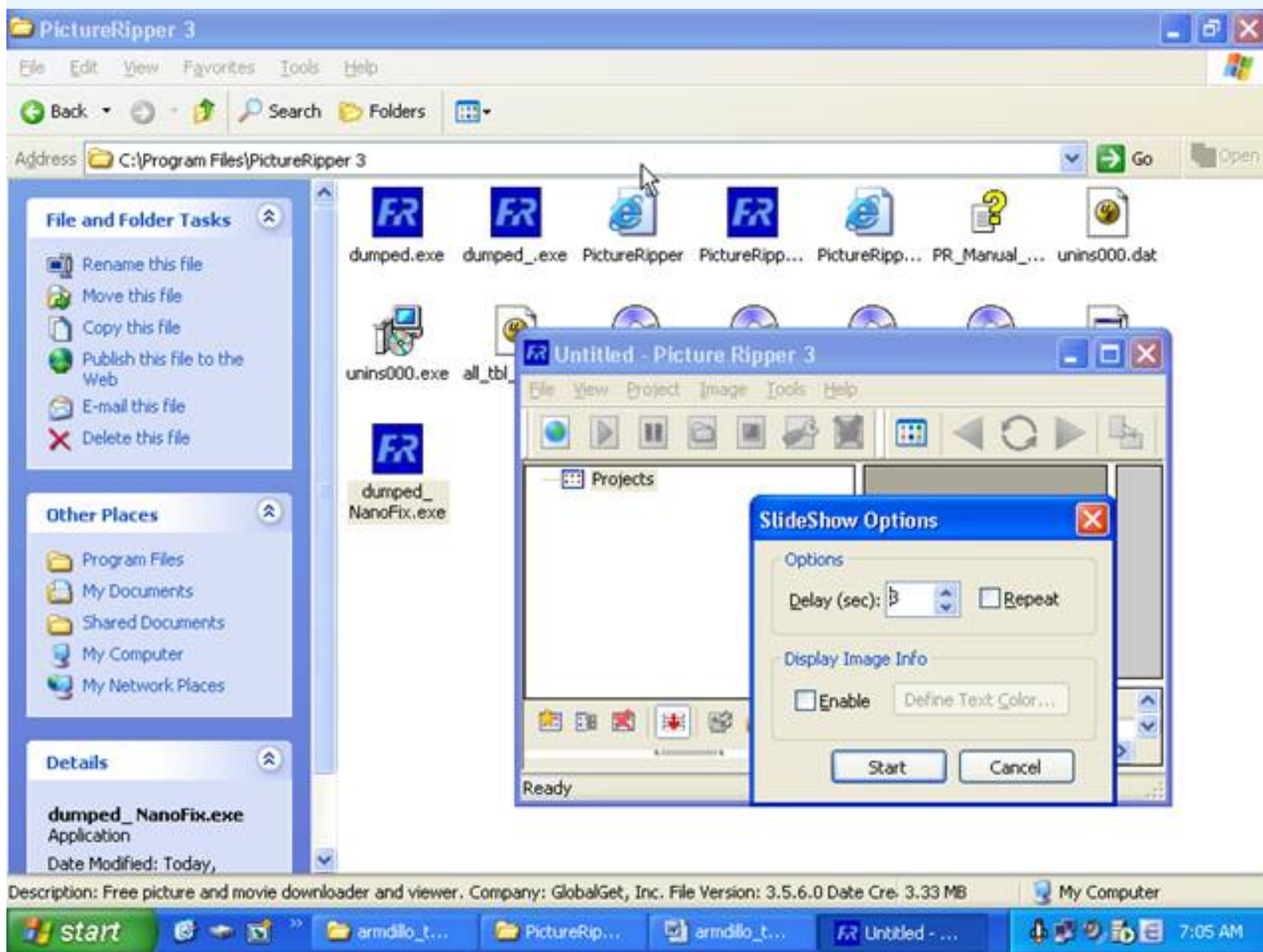
_Now You can save the NanoTable. Next. Click Repair dump:



_Ok:



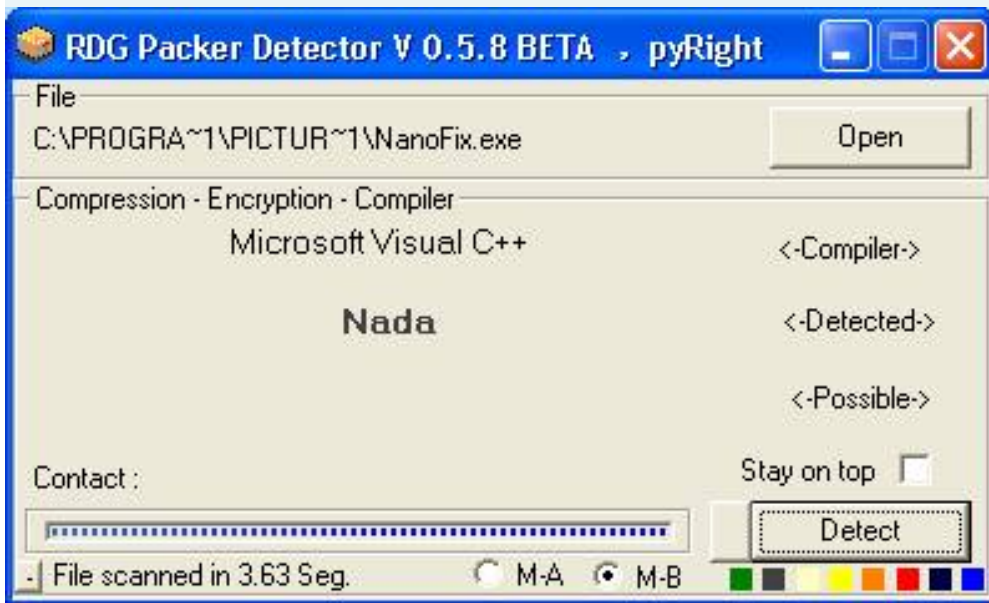
Run Dumped NanoFix.exe file, go to Options menu:



_Unpacked Successful!

Name	V. Offset	V. Size	R. Offset	R. Size	Flags
.data	00151000	00019268	00151000	00019268	C0000040
.text1	0016B000	00030000	0016B000	00030000	E0000020
.adata	0019B000	00010000	0019B000	00010000	E0000020
.data1	001AB000	00020000	001AB000	00020000	C0000040
.pdata	001CB000	000F0000	001CB000	000F0000	C0000040
.rsrc	002BB000	00057000	002BB000	00057000	40000040
.mact	00312000	00003000	00312000	00003000	E0000060
.nano	00315000	00041000	00315000	00040777	E00000E0

Close



IV. Conclusion

_For More tuts, please visit <http://tinicat.de/hacnho> or <http://hacnho.exetools.com>

_Bye!

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtn, hosiminh, Nilrem, fly, MaDMAn_H3rCul3s, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light, iamidiot, WhyNotBar, trickyboy, Merc ... and you!

Special Thanx Cracks Latinos.

Thanx OillyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 25/10/2005)

Sa they are a nui

One lost a child

Miraculous united Lu

Integration hunger Giang Ho

<u>slayer</u>	<u>Ikivo Animator</u>	<u>Snd</u>
<u>takada</u>	Tools you need:	<u>REA</u>
	ABEL (for loader), OllyDebugger	

Difficult: easy

Instruction

In this tut we will learn how to create applications for the loader packed with ProtectionPlus 4.x:).

This is 1 surrounded the other like Softwrap and 1 is interesting easily defeat (ko trouble as Softwrap). I always choose the easiest things to do hehe: P

Here, we will discuss how to create a loader, loader's task is to create a NOP 1 in memory function Call (Yes, namely 1 HAM CALL IS instance: P)

Read Ahead!

How to crack

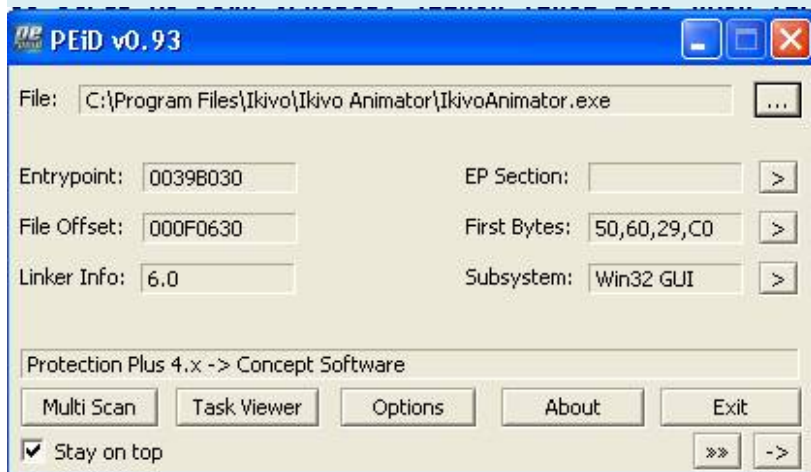
Target taking as Ex today is Ikivo Animator 1.1

You can download the following link

<http://www.ikivo.com:80/animator/downloads/IkivoAnimatorSetup.exe>

Ok here we go:)

First, scan the file is still Peid =



OK to load app Olly, and we will stop at EP

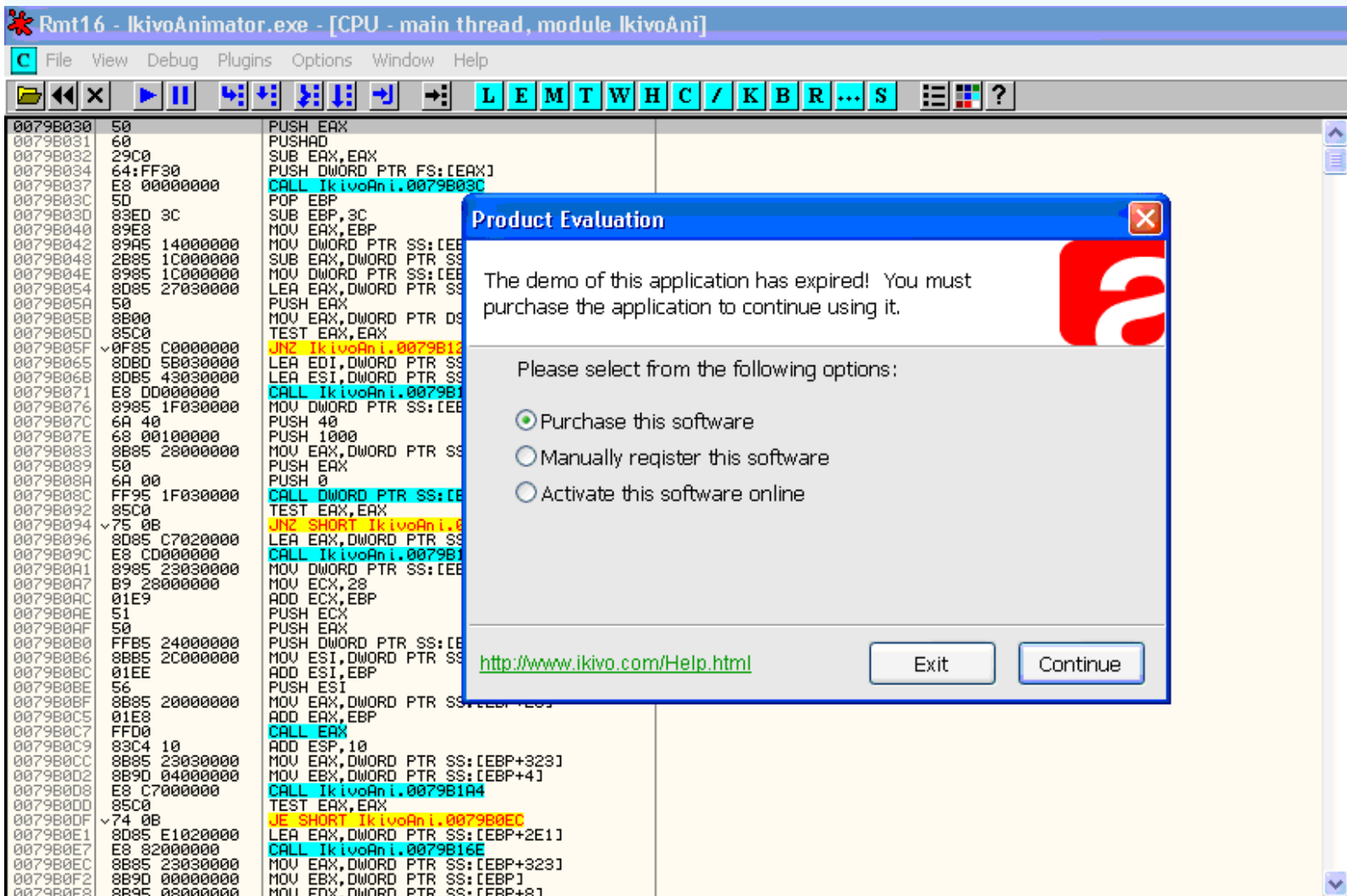
Rmt16 - IkivoAnimator.exe - [CPU - main thread, module IkivoAni]

File View Debug Plugins Options Window Help

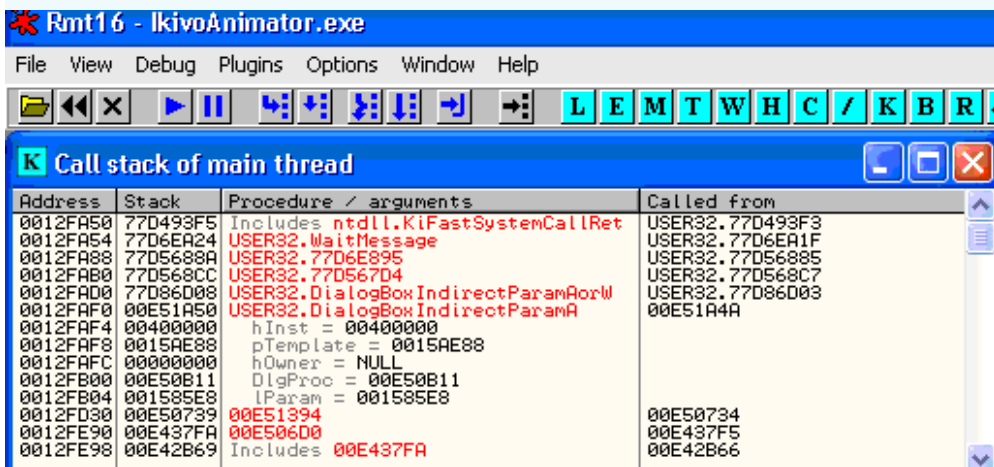
File View Debug Plugins Options Window Help

0079B030 50 PUSH EAX
 0079B031 60 PUSHAD
 0079B032 29C0 SUB EAX,EAX
 0079B034 64:FF30 PUSH DWORD PTR FS:[EAX]
 0079B037 E8 00000000 CALL IkivoAni.0079B03C
 0079B03C 5D POP EBP
 0079B03D 83ED 3C SUB EBP,3C
 0079B040 89E8 MOV EAX,EBP
 0079B042 89A5 14000000 MOV DWORD PTR SS:[EBP+14],ESP
 0079B048 2B85 1C000000 SUB EAX,DWORD PTR SS:[EBP+1C]
 0079B04E 8985 1C000000 MOV DWORD PTR SS:[EBP+1C],EAX
 0079B054 8D85 27030000 LEA EAX,DWORD PTR SS:[EBP+327]
 0079B05A 50 PUSH EAX
 0079B05B 8B00 MOV EAX,DWORD PTR DS:[EAX]
 0079B05D 85C0 TEST EAX,EAX
 0079B05F 75 08 JNZ IkivoAni.0079B125
 0079B065 8DBD 5B030000 LEA EDI,DWORD PTR SS:[EBP+35B]
 0079B06B 8DB5 43030000 LEA ESI,DWORD PTR SS:[EBP+343]
 0079B071 E8 D0000000 CALL IkivoAni.0079B153
 0079B076 8985 1F030000 MOV DWORD PTR SS:[EBP+31F],EAX
 0079B07C 6A 40 PUSH 40
 0079B07E 68 00100000 PUSH 1000
 0079B083 8B85 28000000 MOV EAX,DWORD PTR SS:[EBP+28]
 0079B089 50 PUSH EAX
 0079B08A 6A 00 PUSH 0
 0079B08C FF95 1F030000 CALL DWORD PTR SS:[EBP+31F]
 0079B092 85C0 TEST EAX,EAX
 0079B094 75 08 JNZ SHORT IkivoAni.0079B0A1
 0079B096 8DB5 C7020000 LEA EAX,DWORD PTR SS:[EBP+2C7]
 0079B09C E8 CD000000 CALL IkivoAni.0079B16E
 0079B0A1 8985 23030000 MOV DWORD PTR SS:[EBP+323],EAX
 0079B0A7 B9 28000000 MOV ECX,28
 0079B0AC 01E9 ADD ECX,EBP
 0079B0AE 51 PUSH ECX
 0079B0AF 50 PUSH EAX
 0079B0B0 FF85 24000000 PUSH DWORD PTR SS:[EBP+24]
 0079B0B6 8BB5 2C000000 MOV ESI,DWORD PTR SS:[EBP+2C]
 0079B0BC 01EE ADD ESI,EBP
 0079B0BE 56 PUSH ESI
 0079B0BF 8B85 20000000 MOV EAX,DWORD PTR SS:[EBP+20]
 0079B0C5 01E8 ADD EAX,EBP
 0079B0C7 FFD0 CALL EAX
 0079B0C9 83C4 10 ADD ESP,10
 0079B0CC 8B85 23030000 MOV EAX,DWORD PTR SS:[EBP+323]
 0079B0D2 8B9D 04000000 MOV EBX,DWORD PTR SS:[EBP+4]
 0079B0D8 E8 C7000000 CALL IkivoAni.0079B1A4
 0079B0DD 85C0 TEST EAX,EAX
 0079B0DF 75 08 JE SHORT IkivoAni.0079B0E0
 0079B0E1 8DB5 E1020000 LEA EAX,DWORD PTR SS:[EBP+2E1]
 0079B0E7 E8 82000000 CALL IkivoAni.0079B16E
 0079B0EC 8B85 23030000 MOV EAX,DWORD PTR SS:[EBP+323]
 0079B0F2 8B9D 00000000 MOV EBX,DWORD PTR SS:[EBP]
 0079B0F8 8B95 08000000 MOV EDI,DWORD PTR SS:[EBP+8]

Now HIT F9 to run app and you'll see a nag follows:
 (Do not Mail has expired, therefore I get installed THIS NAG)



Now press F12 to pause olly and then press ALT + K
we will see the following:)



Right click in E51394 then select 'Show Call '

0012FAF8	0015AE88	pTemplate = 0015AE88	
0012FAFC	00000000	hOwner = NULL	
0012FB00	00E50B11	DlgProc = 00E50B11	
0012FB04	001585E8	lParam = 001585E8	
0012FD30	00E50739	00E51394	00E50734
0012FE90	00E437FA	00E506D0	
0012FE98	00E42B69	Includes 00E437	

Actualize	
Hide arguments	Space
Follow address in stack	
Show procedure	Enter
Show call	
Execute to return	F4
Copy to clipboard	►
Appearance	►

Now scroll down a few lines 1, we will see like this: D

2 See screenshots below: P

```

00E50875 808D DCFEFFFF LEA ECX,DWORD PTR SS:[EBP-124]
00E5087B 57 PUSH EDI
00E5087C 56 PUSH ESI
00E5087D 51 PUSH ECX
00E5087E E8 1D73FFFF CALL 00E47BA0
00E50883 83C4 04 ADD ESP,4
00E50886 808405 DCFEFFFF LEA EAX,DWORD PTR SS:[EBP+EAX-124]
00E5088D 50 PUSH EAX
00E5088E E8 0A120000 CALL 00E51A9D
00E50893 83C4 0C ADD ESP,0C
00E50896 8085 DCFEFFFF LEA EAX,DWORD PTR SS:[EBP-124]
00E5089C 50 PUSH EAX
00E5089D 7E 05 JMP SHORT 00E508A4
00E5089F 68 30E4E600 PUSH 0E6E430
00E508A4 68 FB030000 PUSH 3FB
00E508A9 BF 01000000 MOV EDI,1
00E508AE FF75 08 PUSH DWORD PTR SS:[EBP+8]
00E508B1 FF15 8862E600 CALL DWORD PTR DS:[E66288]
00E508B7 7E 03010000 JMP 00E50A8F
00E508BC 8D9E 6E030000 LEA EBX,DWORD PTR DS:[ESI+36E]
00E508C2 53 PUSH EBX
00E508C3 E8 D872FFFF CALL 00E47BA0
00E508C8 83C4 04 ADD ESP,4
00E508CB 85C0 TEST EAX,EAX
00E508CD 7E 0F84 93000000 JE 00E50966
00E508D3 6A 00 PUSH 0

```

ASCII "The demo of this application has expired! You must purchas

USER32.SetDlgItemTextA

```

00E50941 51 00 PUSH 0
00E50943 51 PUSH ECX
00E50944 52 PUSH EDX
00E50945 E8 5672FFFF CALL 00E47BA0
00E5094A 83C4 04 ADD ESP,4
00E5094D 808C05 DCFEFFFF LEA ECX,DWORD PTR SS:[EBP+EAX-124]
00E50954 51 PUSH ECX
00E50955 E8 43110000 CALL 00E51A9D
00E5095A 83C4 0C ADD ESP,0C
00E5095D 808D DCFEFFFF LEA ECX,DWORD PTR SS:[EBP-124]
00E50963 51 PUSH ECX
00E50964 7E 05 JMP SHORT 00E5096B
00E50966 68 C0E3E600 PUSH 0E6E3C0
00E5096B 68 FB030000 PUSH 3FB
00E50970 FF75 08 PUSH DWORD PTR SS:[EBP+8]
00E50973 FF15 8862E600 CALL DWORD PTR DS:[E66288]
00E50979 837E 08 07 CMP DWORD PTR DS:[ESI+8],7
00E5097D 7E 0F8D 0C010000 JMP 00E50A8F
00E50983 83BE 94070000 CMP DWORD PTR DS:[ESI+794],0
00E5098A 7E 0F8C 90000000 JL 00E50A20
00E50990 6A 05 PUSH 5
00E50992 68 FC030000 PUSH 3FC
00E50997 FF75 08 PUSH DWORD PTR SS:[EBP+8]
00E5099A FF15 8462E600 CALL DWORD PTR DS:[E66284]
00E509A0 50 PUSH EAX
00E509A1 FF15 A462E600 CALL DWORD PTR DS:[E662A4]
00E509A7 68 BCF3F600 PUSH 0FAF3BC

```

ASCII "You are running an evaluation copy of this product or you h

USER32.SetDlgItemTextA

USER32.GetDlgItem

USER32.ShowWindow

hehe, just the words appear on the nag me:)

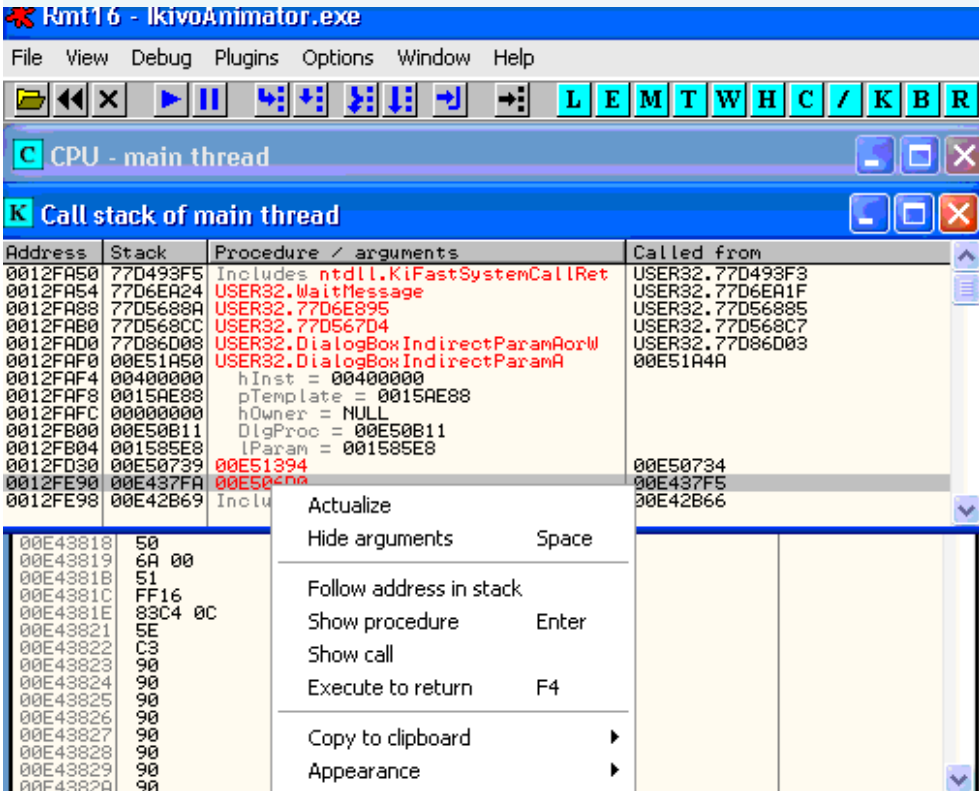
Now they roll up a few lines, for the start of this produce.

Rmt16 - IkivoAnimator.exe - [CPU - main thread]

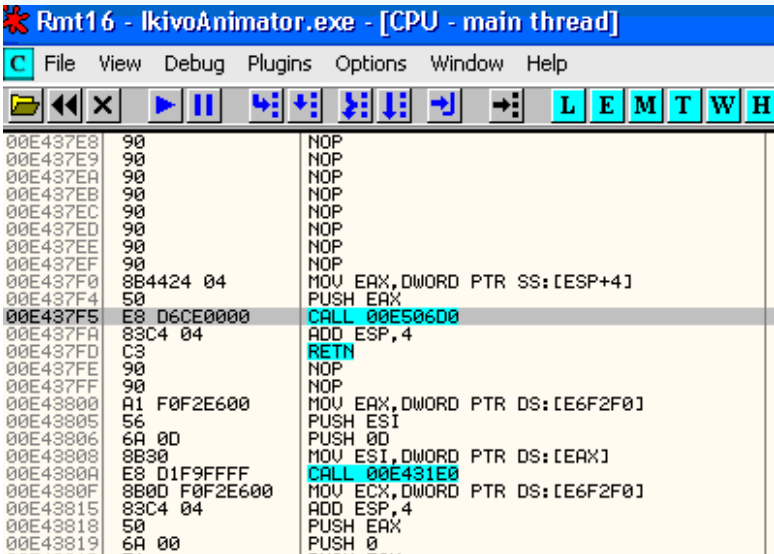
File View Debug Plugins Options Window Help

00E506CD	90	NOP	
00E506CE	90	NOP	
00E506CF	90	NOP	
00E506D0	55	PUSH EBP	
00E506D1	8BEC	MOV EBP,ESP	
00E506D3	81EC 38010000	SUB ESP,138	
00E506D9	56	PUSH ESI	
00E506DA	8D85 C8FEFFFF	LEA EAX,DWORD PTR SS:[EBP-138]	
00E506E0	57	PUSH EDI	
00E506E1	68 00010000	PUSH 100	
00E506E6	50	PUSH EAX	
00E506E7	6A 00	PUSH 0	
00E506E9	FF15 DC61E600	CALL DWORD PTR DS:[E661DC]	kernel32.GetModuleFileNameA
00E506EF	80BC05 C7FEFFFF	CMP BYTE PTR SS:[EBP+EAX-139],5C	
00E506F7	74 0F	JE SHORT 00E50708	
00E506F9	85C0	TEST EAX,EAX	
00E506FB	7E 0B	JLE SHORT 00E50708	
00E506FD	48	DEC EAX	
00E506FE	80BC05 C7FEFFFF	CMP BYTE PTR SS:[EBP+EAX-139],5C	
00E50706	75 F1	JNZ SHORT 00E506F9	
00E50708	33F6	XOR ESI,ESI	
00E5070A	8B7D 08	MOV EDI,DWORD PTR SS:[EBP+8]	
00E5070D	56	PUSH ESI	
00E5070E	8D8405 C8FEFFFF	LEA EAX,DWORD PTR SS:[EBP+EAX-138]	
00E50715	FF77 04	PUSH DWORD PTR DS:[EDI+4]	
00E50718	50	PUSH EAX	
00E50719	E8 7F130000	CALL 00E51A90	
00E5071E	83C4 0C	ADD ESP,0C	
00E50721	8D4D FC	LEA ECX,DWORD PTR SS:[EBP-4]	
00E50724	56	PUSH ESI	
00E50725	51	PUSH ECX	
00E50726	FF77 14	PUSH DWORD PTR DS:[EDI+14]	
00E50729	FF77 08	PUSH DWORD PTR DS:[EDI+8]	
00E5072C	8D8D C8FEFFFF	LEA ECX,DWORD PTR SS:[EBP-138]	
00E50732	51	PUSH ECX	
00E50733	56	PUSH ESI	
00E50734	E8 5B0C0000	CALL 00E51394	
00E50739	85C0	TEST EAX,EAX	
00E5073B	75 6F	JNZ SHORT 00E507AC	
00E5073D	3975 FC	CMP DWORD PTR SS:[EBP-4],ESI	
00E50740	74 66	JE SHORT 00E507A8	
00E50742	56	PUSH ESI	
00E50743	8D45 C8	LEA EAX,DWORD PTR SS:[EBP-38]	
00E50746	68 68E3E600	PUSH 0E6E368	ASCII "Error #"
00E50748	50	PUSH EAX	
00E5074C	E8 4C130000	CALL 00E51A90	
00E50751	83C4 0C	ADD ESP,0C	
00E50754	8D45 C8	LEA EAX,DWORD PTR SS:[EBP-38]	
00E50757	6A 0A	PUSH 0A	
00E50759	50	PUSH EAX	
00E5075A	E8 4174FFFF	CALL 00E47BA0	
00E5075E	83C4 04	ADD ESP,4	

00E506D0 55 PUSH EBP**This is the start produce:)****Now, we will find any known function Call this produce****We back window stack, and then follow function Call 2, you will find interesting**



We stop here

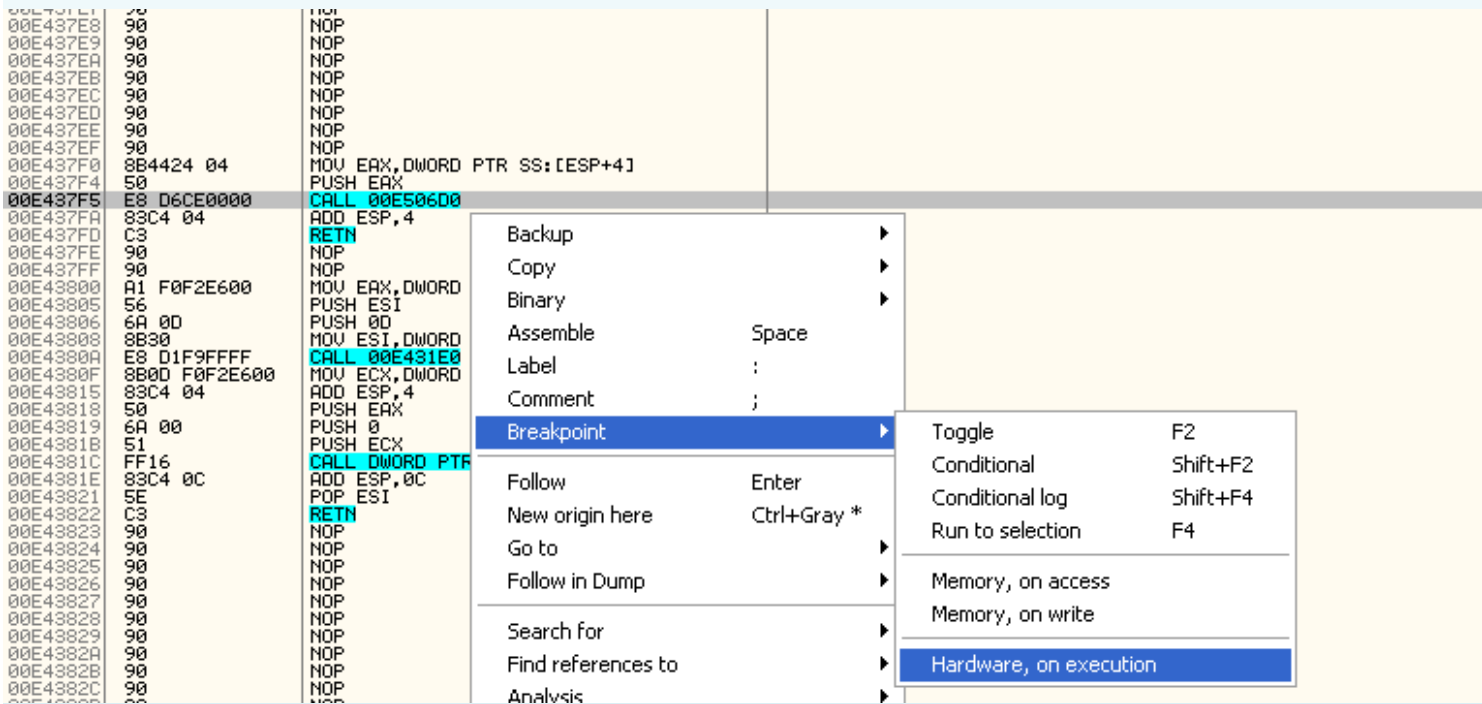


This is Ham Call we need, we produce it called see above:)
If desired, you can check by pressing Enter, and we will stop here

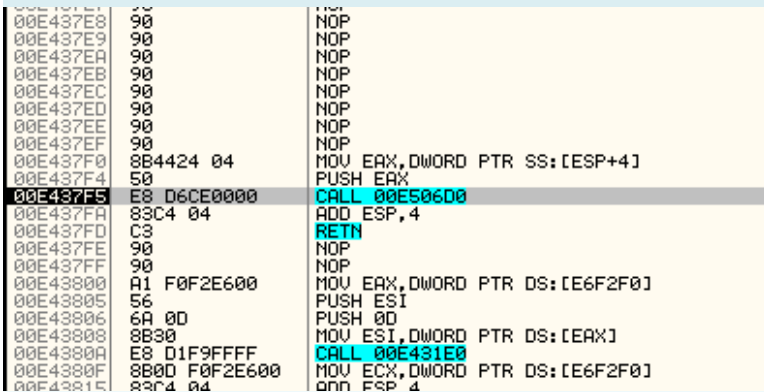
00E506D0 55 PUSH EBP

hehe:) OK, after finding a function Call are looking for, we only need to submit Call this function in memory we have completed the task:)

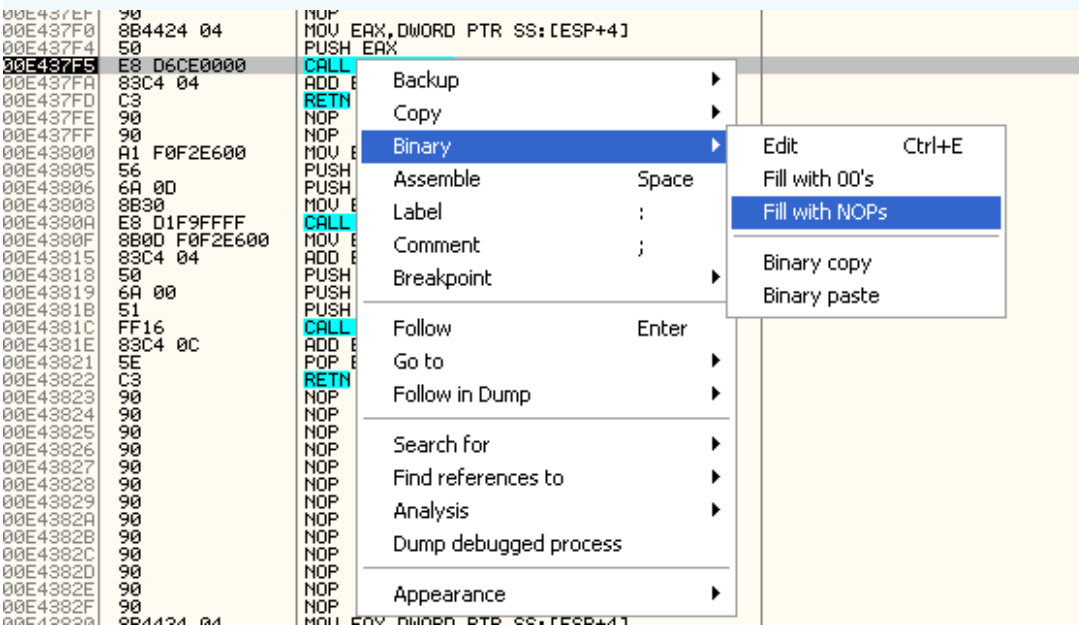
If you do not believe, you can check what I say
First Set 1 bp "Hardware On Execution" Call me at functions just find



Olly restart and run app (press F9), we will break at BP, we set



Olly in Right click and then select Binary ---> Fill with nops



00E437EE	90	NOP
00E437EF	90	NOP
00E437F0	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]
00E437F4	50	PUSH EAX
00E437F5	90	NOP
00E437F6	90	NOP
00E437F7	90	NOP
00E437F8	90	NOP
00E437F9	90	NOP
00E437FA	83C4 04	ADD ESP,4
00E437FD	C3	RETN
00E437FE	90	NOP
00E437FF	90	NOP
00E43800	A1 F0F2E600	MOV EAX,DWORD PTR DS:[E6F2F0]
00E43805	56	PUSH ESI
00E43806	6A 0D	PUSH 0D
00E43808	8B30	MOV ESI,DWORD PTR DS:[EAX]
00E4380A	E8 D1F9FFFF	CALL 00E431E0
00E4380C	90	NOP

Now go delete bp (the Debug menu -----> Hardware Breakpoints and then click Delete)

Rmt16 - lkivoAnimator.exe - [CPU - main th

File

View

Debug

Plugins

Options

Window

Help

Run

Pause

Restart

Close

Step into

Step over

Animate into

Animate over

Execute till return

Execute till user code

Open or clear run trace

Trace into

Trace over

Set condition

Close run trace

Hardware breakpoints

Inspect

Call DLL export

Arguments

Select import libraries

Select path for symbols

F9

F12

Ctrl+F2

Alt+F2

F7

F8

Ctrl+F7

Ctrl+F8

Ctrl+F9

Alt+F9

Ctrl+F11

Ctrl+F12

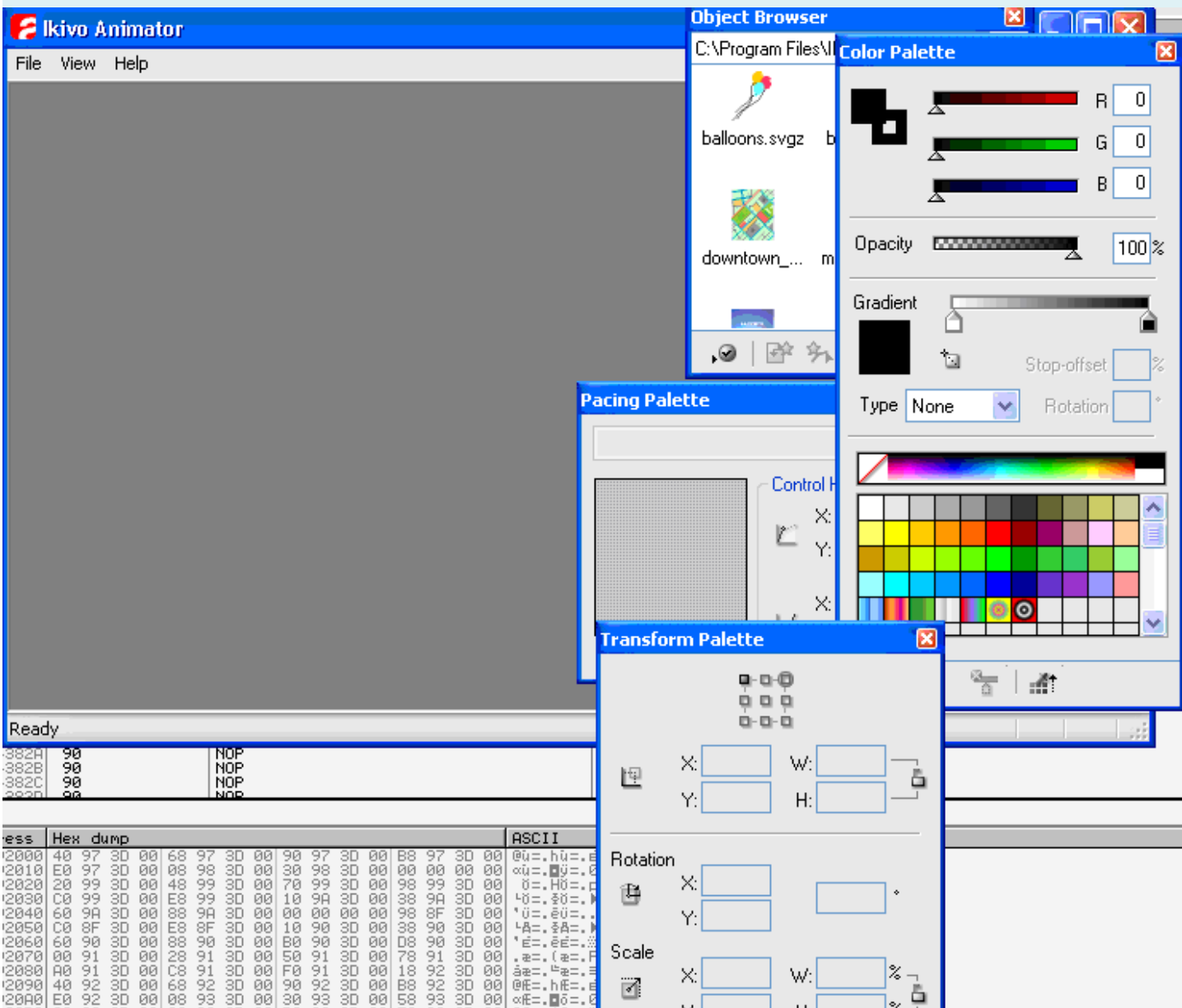
Ctrl+T

Hardware breakpoints

#	Base	Size	Stop on		
1	00E437F5		Execute	Follow 1	Delete 1
2				Follow 2	Delete 2
3				Follow 3	Delete 3
4				Follow 4	Delete 4

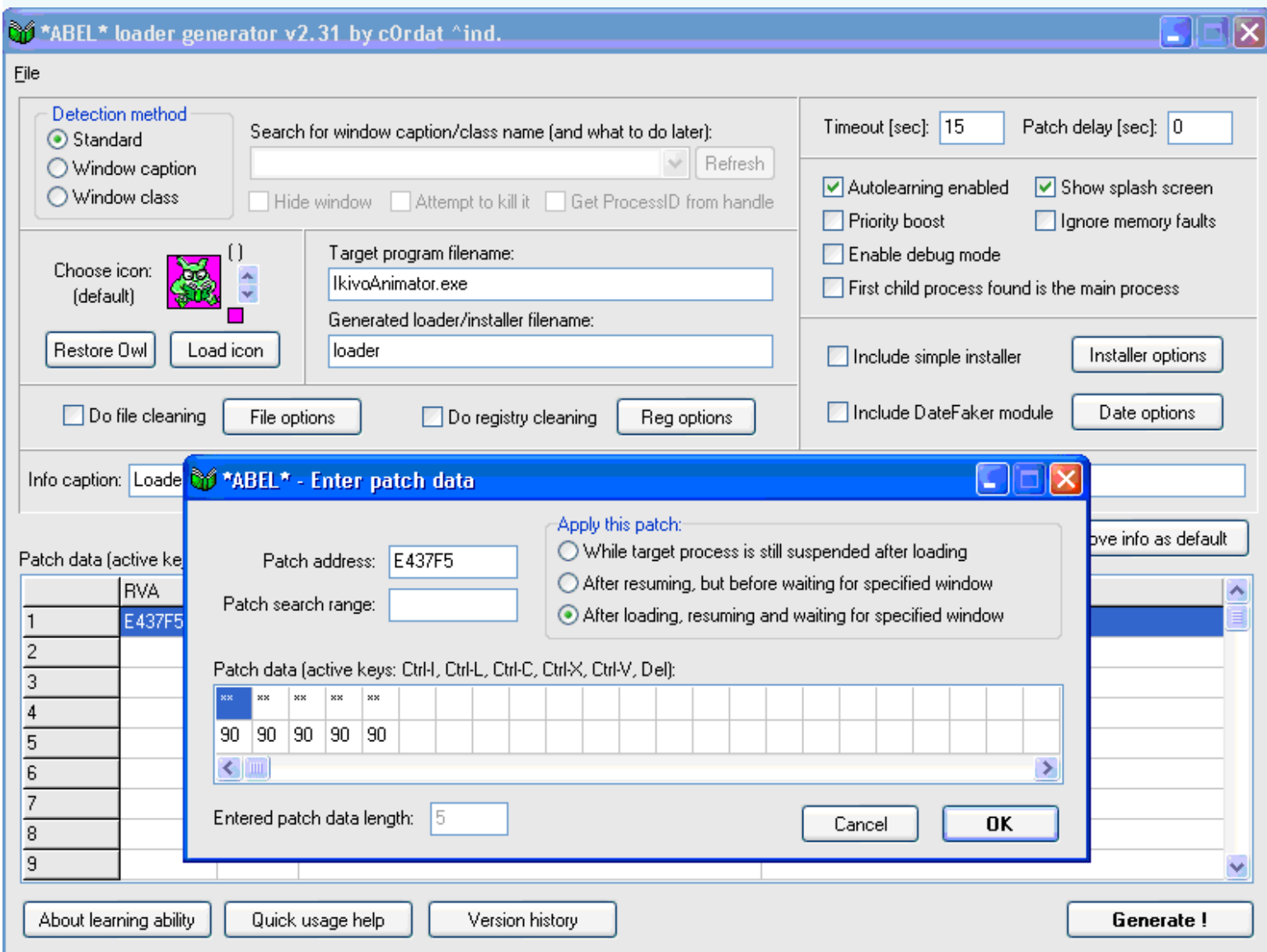
OK

Important or do in this step is If you press Shift-F9 run target that normal stars do, but if they do not run under difficult work from the beginning: D



Khuc khuc, on too:) target run very well and smooth:)
 That means We DEFEATE 4.xxx PROTECTION PLUS:): D

Now only the last step is to create loader (use ABEL), see pictures below for reference



Department surely, to my ** * * * * is because doing so may loader ignores the value of this address, it still NOP in the case have some minor changes:)

Done through!

Hopefully you already know how to defeat Protection Plus 4.x:))

If you want to try this with other target, then they can use this

Video Vault 3.0

<http://www.dvdxsoftware.com/>

Follow similar tut, and you crack it:)

Done!

Closing
words

Hope you enjoyed this guyz tut as much as I enjoyed doing it: P
Greetz to Madman_Hercules for his excellent tuts on unpacking and
inline patching Protection Plus 4.x:))
Greetz to all snd, Mp2k, ICU, TSRh, ARteam members

and to you too: P

Then Laterz
slayer / snd

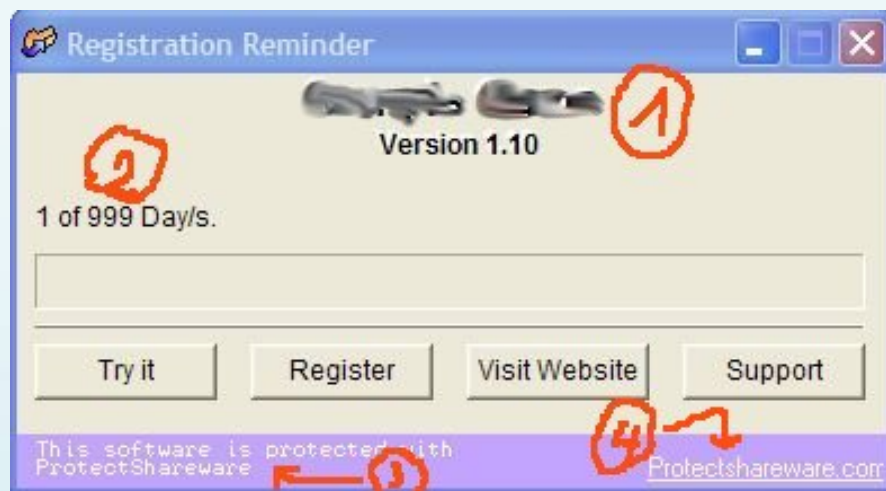
This article from Children's Lena the new post box English tut, or see the man well. Course is free for children with her, since many do not exactly

link via tut English:

<http://www.reaonline.net/forum/showthread.php?p=19818#post19818>

ProtectShareware

I accidentally found that this childish game (hehe, the only daughter I) see the type of protection it nè: NAG a notice accompanying information protect it (the form ni I did not know).



1. -> I deleted the name of the game to the appropriate law forum small.
2. 999 trial days? Hi, it is 1 of the demo course, the author must protect its rùi Code.
3. Name protector is "ProtectShareware" ... the name that I have not heard hê. Who or what is known about it? I've search the REA and other forums, but still do not see any information about it all. This tò mò should I need to visit the site of this protector only.
4. As seen on the site of the protector is very little information about it, is the most ads.

No helpfile, but generally found it can protect protect anything.
Is worthwhile!. I need it immediately to ro thui.;)

Protect
Shareware
features
include:
Advanced
Systems Trial
(which uses a
binary system)
Restrict
application
launch by
Date.
Hardware
Finger Print for
licensing the
software.
Debugger
Detection
System.
State of the art
encryption
Web based
registration
Ability to
register the
software
offline.
Select the
number of trial
runs.
Up to three
levels of key
generation can
be used.
Easy to use.

Quick and extremely effective.
Create trial version softwares and ebooks.
Create trialware for all other exe files.
Option to create trial version based on time, count and date.
Reset trials on new versions
Option to allow only 1 copy
Create exe backup and detect file modification.
Enable hardware fingerprint.
Option to maintain stolen code database.
Highly secured encryption
Supports Windows 98 and higher versions

5. This site is not for download again, as I refuse to do. ;)

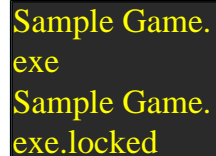
6. Ok. We will try Pack 1 a few programs in ProtectShareware to feel about it. Hum, through the ad and then: first, I try to pack 1 Delphi exe file -> crash it! 5 or 6 languages form, 2 form the assembler, or even do, but ... all crash. The Protector that this image is only used as a form of protection simply is stolen code (???) or not. I have tried all the 2, but i can understand the function quâi do this anymore. And after many files to try to conclude that it is the uong useless. That parents must read authors explain that he's on the site.

But the game I found a good running back! Okay, we go back to the game and see how it is.

7. Olly to load and run -> click the "Try it" and the game starts. But to see any Olly, game still runs. Also ha!

8. Attach the game running away ... but the name of it and then another! Only the top seed and more. "Locked"

9. -> Dom to the game. Rub with a special file here: (I've renamed "Sample Game.exe to photograph (*aged It fear that the law*))



Sample Game.
exe
Sample Game.
exe.locked

and several portus.lic File -> sure license file?

Ok. Sample affordable Game.exe a loader for Game.exe.locked Sample?

10. Uncheck the "locked" in the name of the file and DoubleClick -> It's run! Running game also delicious!

I recognized that far, I have a bit disappointed. However it is to have all?

I started playing, when you enter to try something and the game it crash! Hehe, looks like the above is not all.

Ok. Back Olly, load Sample Game.exe original (renamed before deleting the file locked by the other). Please say it is more written in VB.

11. If Sample Game.exe a loader, we try to set up a BP function API "loader" using commandbar plugin.



CreateProcessA
WriteProcessMemory
Resumethread

12. F9 and click "Try it".

---> I think i must do more to guess: a small window called "loader" sprung (hehe)

And after that break in CreateProcessA.

Address	Value	Comment
0012F264	004484A1	CALL to CreateProcessA from Sample_G.0044849C
0012F268	00000000	ModuleFileName = NULL
0012F26C	001F946C	CommandLine = "Sample Game.exe.locked"
0012F270	00000000	pProcessSecurity = NULL
0012F274	00000000	pThreadSecurity = NULL
0012F278	00000000	InheritHandles = FALSE
0012F27C	00000004	CreationFlags = CREATE_SUSPENDED
0012F280	00000000	pEnvironment = NULL
0012F284	001F221C	CurrentDir = "C:\Program Files\Sample Game\"
0012F288	0012F2E4	pStartupInfo = 0012F2E4
0012F28C	0045F028	pProcessInfo = Sample_G.0045F028
0012F290	0012F514	

The CreateProcess function creates a new process and its primary thread. The new process executes the specified executable file.

```

BOOL CreateProcess (
    LPCTSTR lpApplicationName, / / Pointer to name of executable module
    LPTSTR lpCommandLine, / / Pointer to command line string
    LPSECURITY_ATTRIBUTES lpProcessAttributes, / / pointer to process security Attributes
    LPSECURITY_ATTRIBUTES lpThreadAttributes, / / Pointer to thread security Attributes
    BOOL bInheritHandles, / / handle inheritance flag
    DWORD dwCreationFlags, / / creation flags
    LPVOID lpEnvironment, / / Pointer to new environment block
    LPCTSTR lpCurrentDirectory, / / pointer to current directory name
    LPSTARTUPINFO lpStartupInfo, / / Pointer to STARTUPINFO
    LPPROCESS_INFORMATION lpProcessInformation / / pointer to PROCESS_INFORMATION
);

```

I think i need to explain anything more nhé.

13. Click "run" again and again, all at the break WriteProcessMemory. Backtracing (*sure trace wa RETN so many times about the code*), we see orders jumped 1:

Address	Hex dump	Disassembly	Comment
0040EC44	\$ A1 F4FA4500	MOV EAX,DWORD PTR DS:[45FAF4]	
0040EC49	. 0BC0	OR EAX,EAX	
0040EC4B	. 74 02	JE SHORT Sample_G.0040EC4F	
0040EC4D	. FFE0	JMP EAX	kernel32.WriteProcessMemory
0040EC4F	> 68 2CEC4000	PUSH Sample_G.0040EC2C	
0040EC54	. B8 00A54000	MOV EAX,<JMP.&MSVBVM60.DllFunctionC	
0040EC59	. FFD0	CALL EAX	
0040EC5B	. FFE0	JMP EAX	

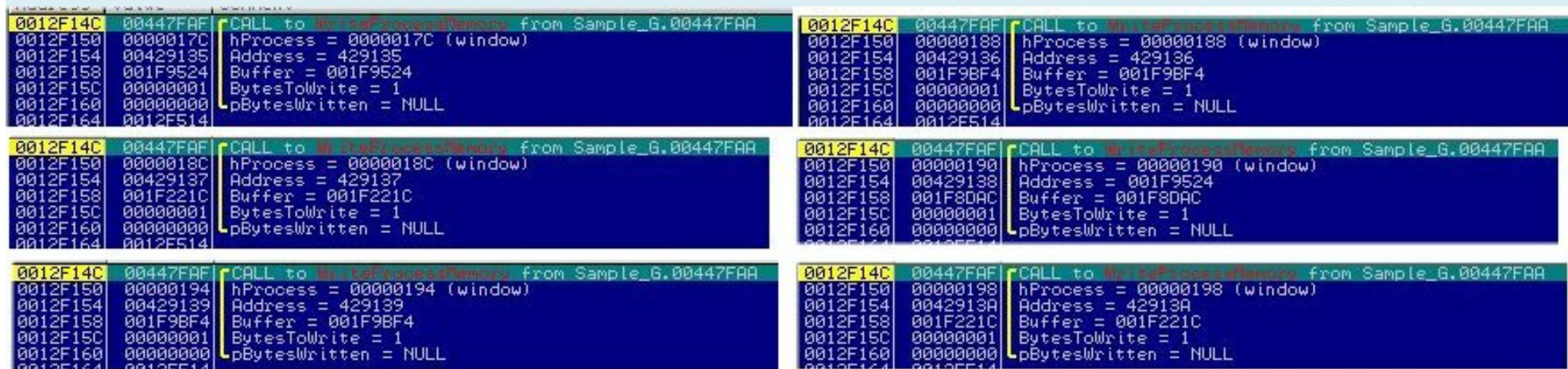
The WriteProcessMemory memory function writes a specified process. The entire area to be written to must be accessible, or the operation fails.

```

BOOL WriteProcessMemory (
    HANDLE hProcess, / / handle to process whose memory is written to
    LPVOID lpBaseAddress, / / address to start writing to
    LPVOID lpBuffer, / / Pointer to buffer to write data to
    DWORD nSize, / / number of bytes to write
    LPDWORD lpNumberOfBytesWritten / / actual number of bytes written
);

```

1 More few more pictures:



What is looking better then: WriteProcessMemory to write only 1 byte in each, at the beginning 429,136 Buffer area in turn contains 3 addresses, in each byte will be overwritten from this process through the other process. You can dom to dump window to see the bytes are, if you want to write.

14. Delete BP in WriteProcessMemory go (I also have a few more BP, for example OpenProcess break but there do have more information What should I remove all oi).
Press F9 and break in ResumeThread

Address	Value	Comment
0012F178	00447195	CALL to ResumeThread from Sample_G.00447190
0012F17C	00000188	hThread = 00000188 (window)
0012F180	0012F514	
0012F184	0012F5E4	

The ResumeThread function decrements a thread's suspend count. When the suspend count

```
is decremented to zero, the execution of the thread is resumed.
```

```
DWORD ResumeThread (
```

```
HANDLE hThread / / identifies thread to restart
);
```

Certainly do need to explain more -> press F9 and running game.

15. Think tí time. This can be understood as: loader "Sample Game.exe" will run "Sample Game.exe.locked" and suspend thread. Start patch at 429,136 until the completion of the game and resume thread run lickerish

16. Who can be more research, but the time we try in Olly. Load Sample Game.exe.locked and select -> Go To -> 429,136

Chaaa! all first byte 0, from 429,136 to 429,534. Be read, too many bytes to lose! Without some bytes here, game will crash. -> In the protection, protector has "cut" of bytes from the original file, then rename it to "locked", then create 1 loader to manage the registration and re-inject it into the byte file protection allows.

17. Now running game (outside Olly) Attach it to Olly Game.exe.locked Sample files. Goto to 429,136 -> all bytes have been loaded into the code and look better then đây.

18. To all black byte from 429,136 to 429,535 -> select binary copy. Then off to the end.

19. Rename. "Locked" to the original file name. Exe and then load it into Olly -> Go to to 429,136 -> Binary paste the bytes have to copy. Save the file there.

20. Delete the file loader. Running game. All good beginning. *Another one Bites the dust;*)

Additional comments:

After writing this article, I try to protect some files again, and good then, I have to protect *winasm studio executable* -> success! I also try to protect your bags *RadAsm.exe* -> lickerish! So I do not have luck with the other files? lol However, I still have not understood the stolen bytes option to do more. Select or do choose are the same àh.

If you want to try this is the protector please note that it will override the original file when creating a backup option if loader do is check.

Reviews's first site for this protector.

<<END OF TRANSLATION>>

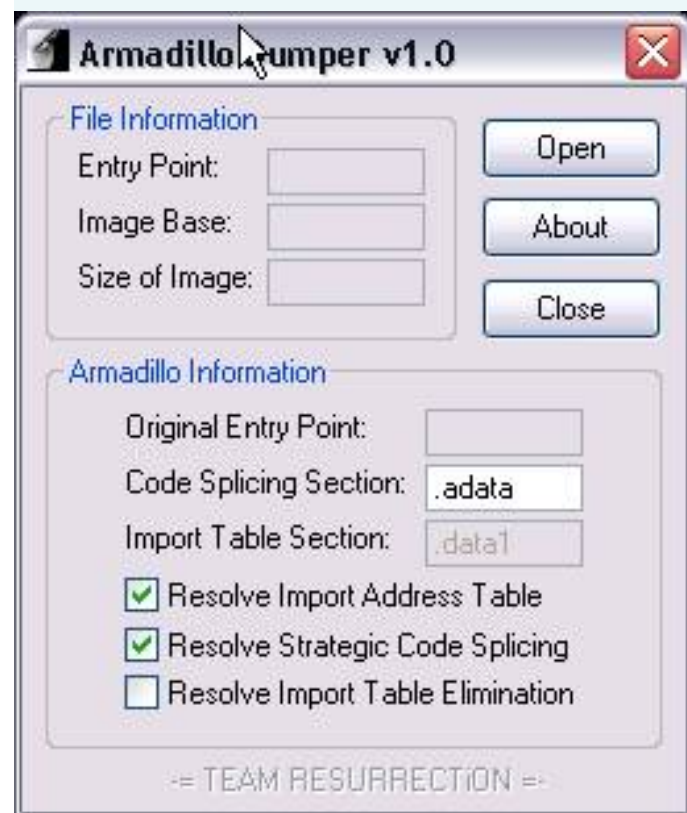
trickyboy

Big thanks to Lena151.
Best regard.

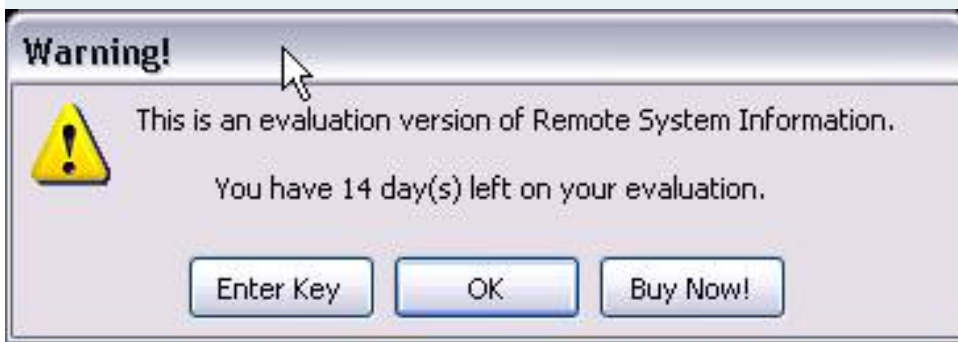
Debugblocker + + Code Splicing Nanomites

SoftWare : **Remote System Information 3.2**
Homepage : <http://www.digitallabs.net>
Packed : **Debugblocker + + Code Splicing Nanomites**
Crack Tool **1. ArmaDumper 1.0**
2. ArmInline 0.71
3. Import REConstructor 1.6
4. extended Task Manager
Author : **Why Not Bar**

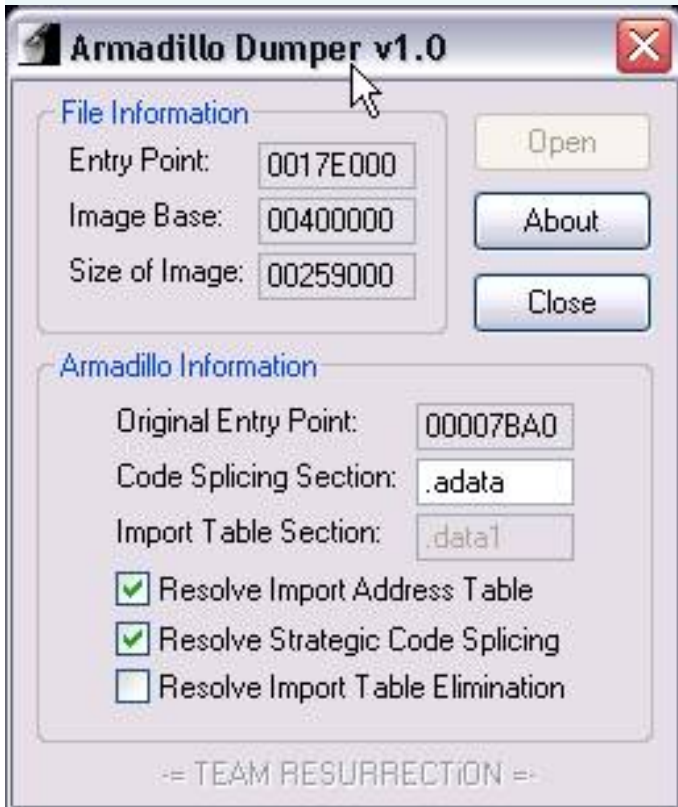
_Khi Unpack Pack 1 are soft in Arma we often target the Load Olly conducted since then in many ways to unpack it. But today, we try to unpack without touching Olly try to watch stars. Here they use Tool *ArmaDumper 1.0*. and select the following



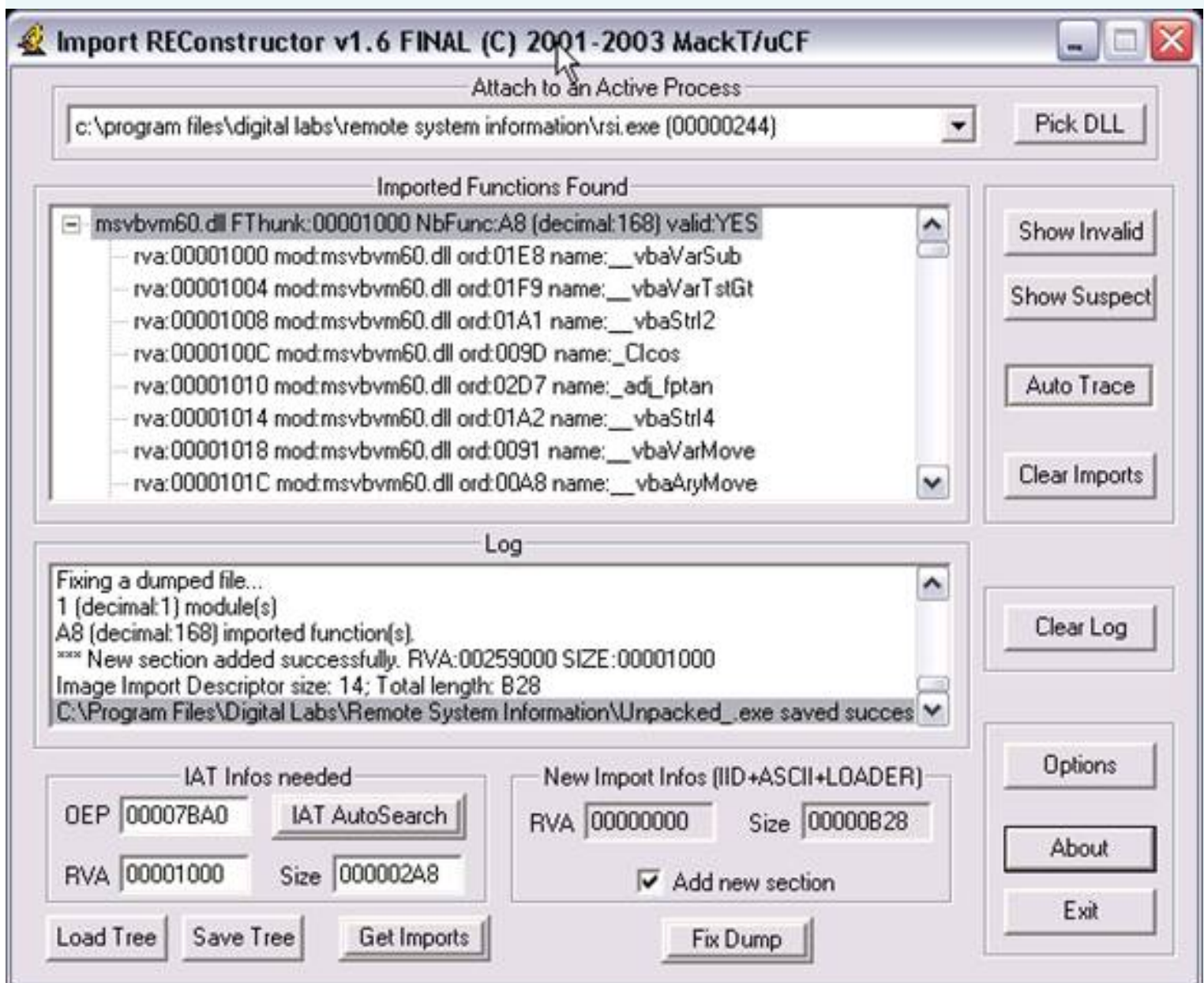
_Open Select File "rsi.exe" appear Nag



_Nhan OK, we have been as follows:



_M in ImportREC to conduct Fix dump



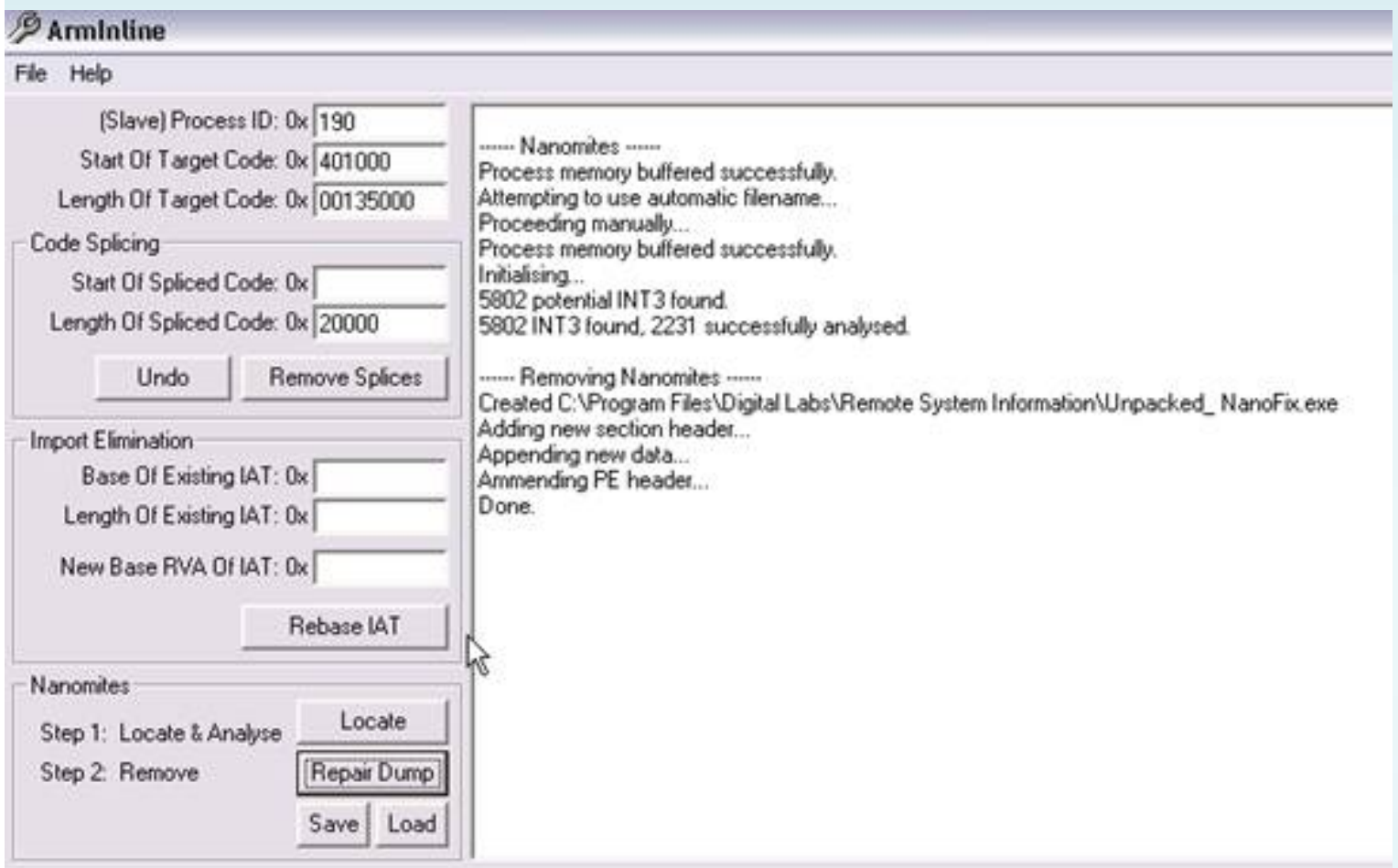
Run Try File "Unpacked.exe. Há a running call. But when you use the Open with crash



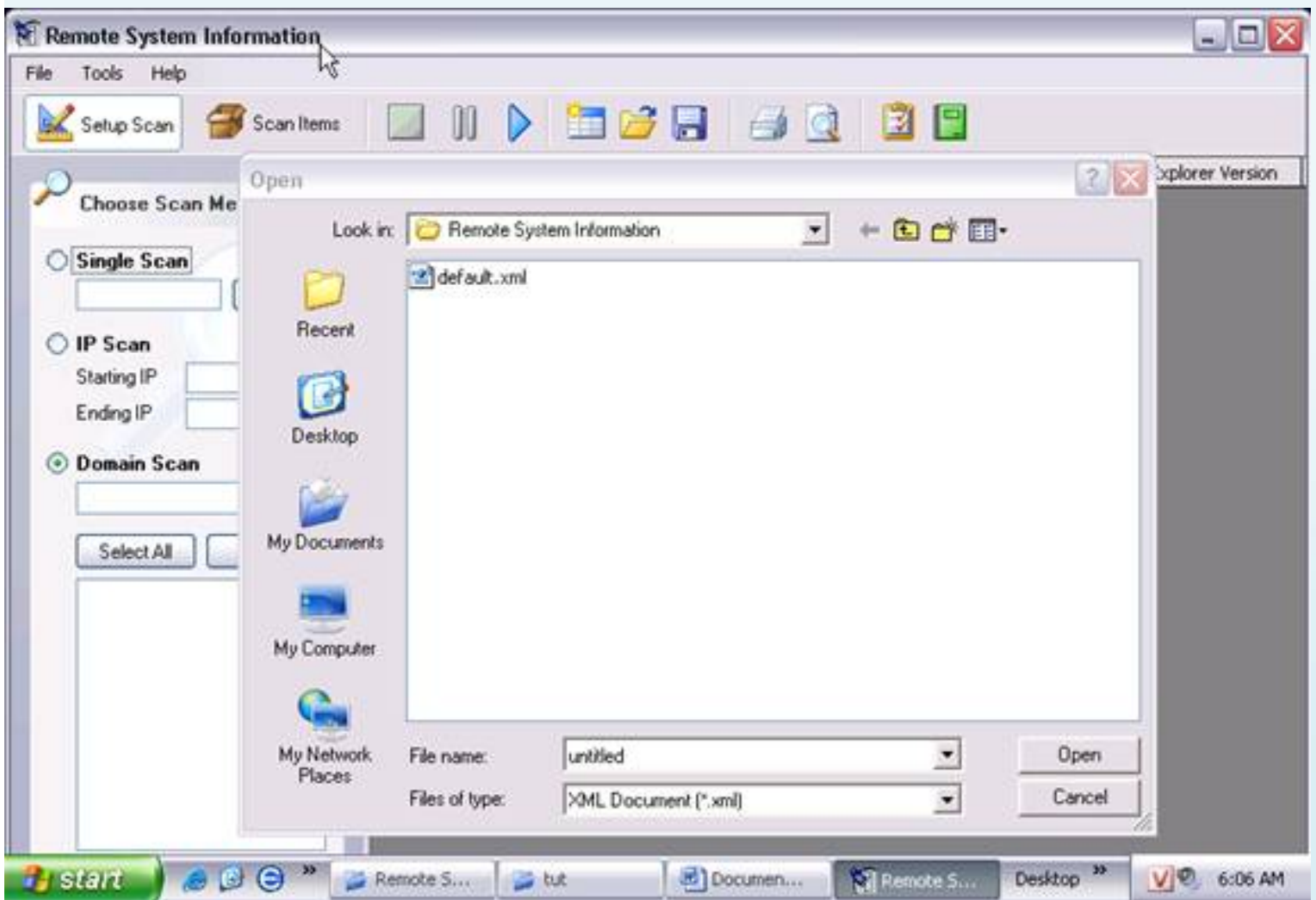
Do The phenomenon of Nano, we conducted Fix it is only OK. Run the file "Unpacked.exe" again. For it is, extended run Task Manager to see its PID

Image Name	PID	Description	User Name	CPU	Mem Usage	Handles
svchost.exe	844	Generic Host Process for Win32...	NT AUTHORITY\LOCAL SERVICE	0	272 K	198
System	4		NT AUTHORITY\SYSTEM	0	48 K	183
System Idle Pro...	0			96	16 K	0
UnKey.exe	1048		COMPUTER\Welcome	0	1,176 K	18
Unpacked_.exe	400		COMPUTER\Welcome	1	13,200 K	160
wdfmgr.exe	1236	Windows User Mode Driver Man...	NT AUTHORITY\LOCAL SERVICE	0	84 K	65
winlogon.exe	448	Windows NT Logon Application	NT AUTHORITY\SYSTEM	0	1,024 K	582
WINWORD.EXE	176	Microsoft Office Word	COMPUTER\Welcome	0	16,436 K	288
XTM.exe	1276	Warecase eXtended Task Mana...	COMPUTER\Welcome	2	2,980 K	58

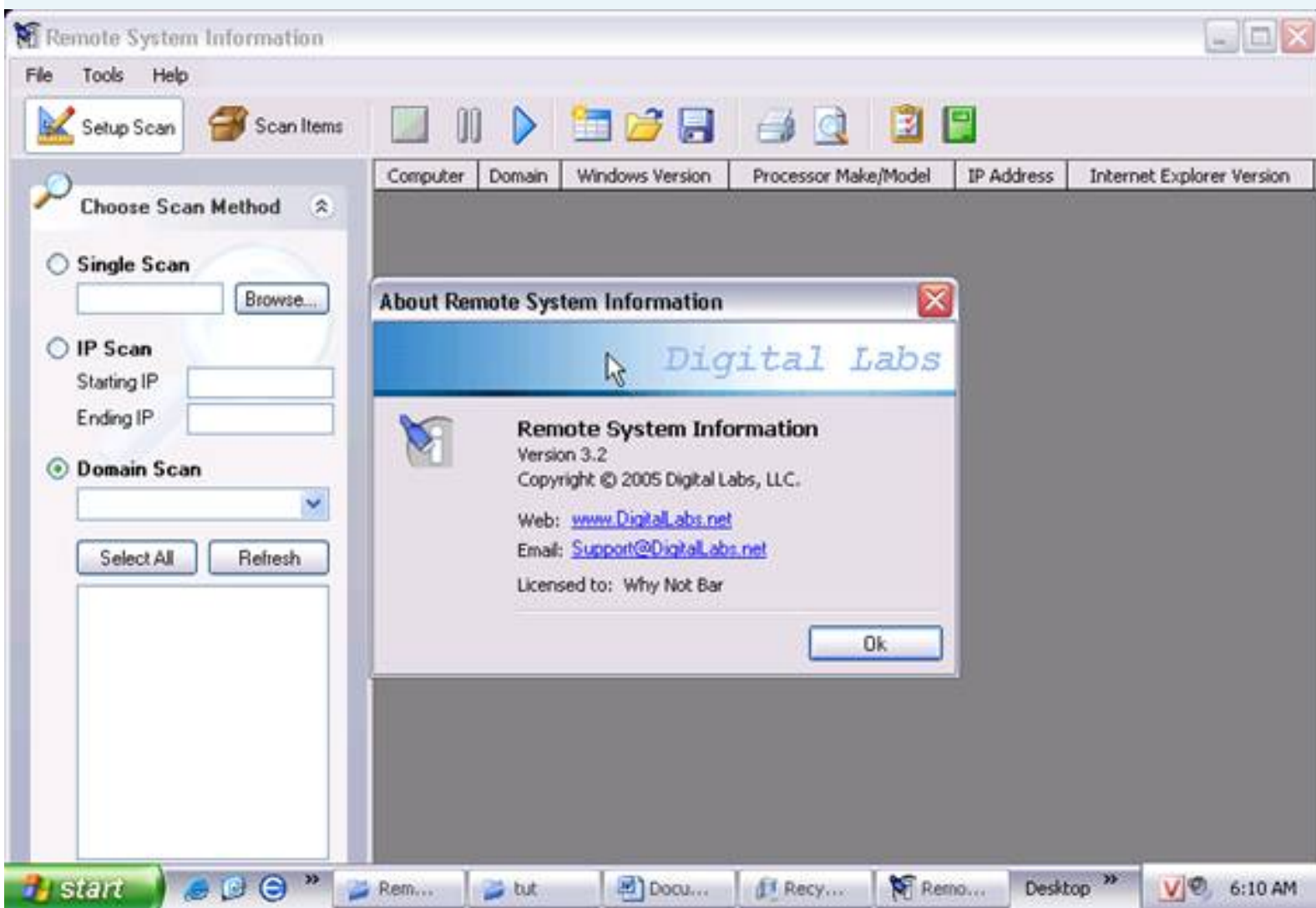
_O Machine is 400 HEX through it with the 190th Open ArmInline and to fill FIX Nano conducted as normal. If you do it right will result as follows:



Chay Try File "Unpacked NanoFix.exe" and use Open



_OK, Gõì Done! Ha ha Try to see the About information, it is always self Exit. So why? Do not worry carp File "ArmAccess.dll" folder on the installation is OK!



_Oh Yeah, unpack Done !!!!!!!!!!!!!!! Crack from the infected. Ha ha!

***Written by Why
Not Bar***

1:19 RLPack Research

Target: notepad_type_1.exe

Pack options:

Protect from generic unpacker

Import protection Redirection

Enforce memory protection

Olly notepad_type_1.exe to load, you see:

```
010238B9> 60 PUSHAD
```

```
010238BA E8 00000000 CALL notepad_.010238BF
```

```
010238BF 83C4 04 ADD ESP, 4
```

Khúc walking seems quite early this same UPX, one method to try to OEP by UPX.

And we have the results:

```
0100739D. 6A 70 PUSH 70
```

```
0100739F. 68 98180001 PUSH notepad_.01001898
```

```
010073A4. E8 BF010000 CALL notepad_.01007568
```

```
010073A9. 33DB XOR EBX, EBX
```

```
010073AB. 53 PUSH EBX
```

```
010073AC. 8B3D CC100001 MOV EDI, DWORD PTR DS: [10010CC]
```

```
010073B2. FFD7 CALL EDI
```

```
010073B4. 66:8138 4D5A Cmp WORD PTR DS: [EAX], 5A4D
```

```
010073B9. 75 1F JNZ SHORT notepad_.010073DA
```

```
010073BB. 8B48 3C MOV ECX, DWORD PTR DS: [EAX +3 C]
```

```
010073BE. 03C8 ADD ECX, EAX
```

```
010073C0. 8139 50450000 Cmp DWORD PTR DS: [ECX], 4550
```

This is the OEP, such OEP to find quite easily. Check to see intermodular calls APIs are not intact, we have the results:

```
01006D69 CALL DWORD PTR DS: [100102C] GDI32.AbortDoc
```

```
01007522 CALL DWORD PTR DS: [1001300] msvcrt._cexit
```

```
010030DA CALL DWORD PTR DS: [10012D0] comdlg32.ChooseFontW
```

```
01002003 CALL DWORD PTR DS: [10012E0] comdlg32.763B2BBF
```

```
010028D0 CALL DWORD PTR DS: [10012E0] comdlg32.763B2BBF
```

```
01002E09 CALL DWORD PTR DS: [10012E0] comdlg32.763B2BBF
```

```
010075E7 CALL notepad_.010075FC msvcrt._controlfp
```

```
010064A4 CALL DWORD PTR DS: [1001040] GDI32.CreateDCW
```

```
01006588 CALL DWORD PTR DS: [1001040] GDI32.CreateDCW
```

```
010030F7 CALL DWORD PTR DS: [1001064] GDI32.CreateFontIndirectW
```

Some API still raw state. But have 1 number of the API call being changed.

```
01001984 CALL DWORD PTR DS: [10010D8] DS: [010010D8] = 0094018F
```

```
010019EF CALL DWORD PTR DS: [1001224] DS: [01001224] = 009407E0
```



```

01001A19 CALL DWORD PTR DS: [1001220] DS: [01001220] = 009407CB
01001A64 CALL DWORD PTR DS: [100123C] DS: [0100123C] = 0094085E
01001B0F CALL DWORD PTR DS: [1001238] DS: [01001238] = 00940849
01001B1C CALL DWORD PTR DS: [1001234] DS: [01001234] = 00940834
01001B2B CALL DWORD PTR DS: [1001230] DS: [01001230] = 0094081F
01001B34 CALL DWORD PTR DS: [100122C] DS: [0100122C] = 0094080A
01001B5E CALL DWORD PTR DS: [100122C] DS: [0100122C] = 0094080A
01001B86 CALL DWORD PTR DS: [1001228] DS: [01001228] = 009407F5
01001C17 CALL DWORD PTR DS: [1001244] DS: [01001244] = 00940888
01001D52 CALL DWORD PTR DS: [1001244] DS: [01001244] = 00940888
01001D88 CALL DWORD PTR DS: [1001264] DS: [01001264] = 00940930
01001DA3 CALL DWORD PTR DS: [1001240] DS: [01001240] = 00940873

```

10010D8 area to remember and observe in the dump, you will have:

IAT begin = 01001000 77DD6FC8 EoYw ADVAPI32.RegQueryValueExW

IAT end = 01001344

Note to the value of the API changes were first

First IAT Redirect 0100108C = 00940000 .. ".

So we will set hardware breakpoint on memory write 0100108C in to see the RlPack create and write new value to how.

Set on write in hwbp 0100108C. Ctrl-F2 to restart and F9 to run. I will break in:

```
01023C36 F3: A4 REP MOVSB BYTE PTR ES: [EDI], BYTE PTR DS: [>
```

```
01023C38 5e POP ESI
```

```
01023C39 ^ 8E EB JMP SHORT notepad_.01023BC9
```

This is not where RlPack write new value that is just delete the bytes of 0. Okie, F7, F8 Then F9 to continue, we will stop at:

```
010269A4 8907 MOV DWORD PTR DS: [EDI], EAX <--
```

```
010269A6 C3 RETN
```

```
010269A7 60 PUSHAD
```

This is where RlPack write the address of the API. Trace the flow, we are to:

```
01023A85 / 0F84 DE1A0000 JE notepad_.01025569
```

```
01023A8B | E8 D42F0000 CALL notepad_.01026A64
```

```
01023A90 | E8 8F2E0000 CALL notepad_.01026924
```

```
01023A95 | 83C7 04 ADD EDI, 4 <- here we
```

```
01023A98 | 8385 E14B0000 0> ADD DWORD PTR SS: [EBP +4 BE1], 4
```

Here, we try BP No. 1 in order to call on the current location and run the program. 1 the number of experiments, we commented:

```
01023A6F 8B85 E14B0000 MOV EAX, DWORD PTR SS: [EBP +4 BE1]
```

```
01023A75 8B00 MOV EAX, DWORD PTR DS: [EAX]
```

```
01023A77 50 PUSH EAX
```

```

01023A78 FFB5 DD4B0000 PUSH DWORD PTR SS: [EBP +4 BDD]
01023A7E E8 D1370000 CALL notepad_.01027254 <- get the API address root
01023A83 85C0 TEST EAX, EAX
01023A85 0F84 DE1A0000 JE notepad_.01025569
01023A8B E8 D42F0000 CALL notepad_.01026A64 <- create a new address
API (when met some conditions)
01023A90 E8 8F2E0000 CALL notepad_.01026924 <- API address assigned to
the IAT

```

So the rest of the simple, one in 01023A8B NOP, delete all breakpoint and F9 to run. Then see table intermodular calls will result:

```

01002003 CALL DWORD PTR DS: [10012E0] comdlg32.763B2BBF
010028D0 CALL DWORD PTR DS: [10012E0] comdlg32.763B2BBF
01002E09 CALL DWORD PTR DS: [10012E0] comdlg32.763B2BBF
01002146 CALL DWORD PTR DS: [10010EC] kernel32.CompareStringW
01005896 CALL DWORD PTR DS: [10010EC] kernel32.CompareStringW
01005921 CALL DWORD PTR DS: [10010EC] kernel32.CompareStringW
010075E7 CALL notepad_.010075FC msvcrt._controlfp
010064A4 CALL DWORD PTR DS: [1001040] GDI32.CreateDCW
01006588 CALL DWORD PTR DS: [1001040] GDI32.CreateDCW
01006AD6 CALL DWORD PTR DS: [1001214] user32.CreateDialogParamW
0100522E CALL DWORD PTR DS: [10010B4] kernel32.CreateFileMappingW
01002653 CALL DWORD PTR DS: [1001104] kernel32.CreateFileW
01002DAA CALL DWORD PTR DS: [1001104] kernel32.CreateFileW
010033AA CALL DWORD PTR DS: [1001104] kernel32.CreateFileW
010043D0 CALL DWORD PTR DS: [1001104] kernel32.CreateFileW
01004A07 CALL DWORD PTR DS: [1001104] kernel32.CreateFileW
...

```

-> Full IAT. The rest of the dump and fix.

1:19 RlPack Anti Anti dump dump & Advance

Options:

Anticracking Protection

+ Anti dump protection

+ Anti Advance dump protection instructions 512

For simplicity, we only protect the options on the target, the goal is research on how to dump the Anti RlPack. If we dump, and run IAT fixed target, we found:

```

010074E4. 50 PUSH EAX; / pStartupinfo
010074E5. FF15 D0100001 CALL DWORD PTR DS: [<& kernel32.GetStartup> \
GetStartupInfoA
010074EB. F645 AC 01 TEST BYTE PTR SS: [EBP-54], 1

```

```

010074EF. 74 11 JE SHORT notepad_.01007502
010074F1. 0FB745 B0 MOVZX EAX, WORD PTR SS: [EBP-50]
010074F5. EB 0E JMP SHORT notepad_.01007505
010074F7> 803E 20 Cmp BYTE PTR DS: [ESI], 20
010074FA. ^ 76 D8 JBE SHORT notepad_.010074D4
010074FC. 46 INC ESI
010074FD. 8975 E0 MOV DWORD PTR SS: [EBP-20], ESI
01007500. ^ EB F5 JMP SHORT notepad_.010074F7
01007502> 6A 0A PUSH 0A
01007504. 58 POP EAX
01007505> 50 PUSH EAX
01007506. 56 PUSH ESI
01007507. 53 PUSH EBX
01007508. 53 PUSH EBX
01007509. FFD7 CALL EDI
0100750B. 50 PUSH EAX
0100750C. E8 25B4FFFF CALL notepad_.01002936 <- here

```

When running 0100750C to call and implement this program will be crash. Back to the target and not unpack 0100750C break in to see how RLPack work, we have to trace:

```

01002936 $ 8BFF MOV EDI, EDI
01002938. 55 PUSH EBP
01002939. 8BEC MOV EBP, ESP
0100293B. 83EC 20 SUB ESP, 20
0100293E. 56 PUSH ESI
0100293F. 57 PUSH EDI
01,002,940 .- E9 B0E890FF JMP 009111F5 <- 1st jmp

```

The first is quite similar to the function of file notepad not unpack. I remember calling area is the area from 910,000 remember AntiDump Code. We trace test JMP will take you where to go:

```

009111F5 83EC 04 SUB ESP, 4
009111F8 C70424 14110001 MOV DWORD PTR SS: [ESP], 1001114
009111FF 50 PUSH EAX
00911200 8B4424 04 MOV EAX, DWORD PTR SS: [ESP +4]
00911204 8B00 MOV EAX, DWORD PTR DS: [EAX]
00911206 894424 04 MOV DWORD PTR SS: [ESP +4], EAX
0091120A 58 POP EAX
0091120B 83C4 04 ADD ESP, 4
FC 0091120E FF5424 CALL DWORD PTR SS: [ESP-4]

```



```
00911212 - E9 2F176F00 JMP notepad_.01002946 <- return next code
```

We try to analyze the views of the code at 009111F5 to do:

```
009111F5 83EC 04 SUB ESP, 4
```

```
009111F8 C70424 14110001 MOV DWORD PTR SS: [ESP], 1001114 <- 01001114
```

IAT addr of GetCommandLineW

```
009111FF 50 PUSH EAX
```

```
00911200 8B4424 04 MOV EAX, DWORD PTR SS: [ESP +4] <- EAX = 01001114
```

```
00911204 8B00 MOV EAX, DWORD PTR DS: [EAX] <- EAX = [01001114] = addr
of GetCommandLineW
```

```
00911206 894424 04 MOV DWORD PTR SS: [ESP +4], EAX <- [ESP +4] = addr
of GetCommandLineW
```

```
0091120A 58 POP EAX
```

```
0091120B 83C4 04 ADD ESP, 4
```

```
FC 0091120E FF5424 CALL DWORD PTR SS: [ESP-4] <- Call GetCommandLineW
```

```
00911212 - E9 2F176F00 JMP notepad_.01002946 <- return next code
```

So, guess 1 JMP section from the area code to area code AntiDump equivalent to 1 instruction. And compared with the original code, we can be sure the code is equivalent to:

```
CALL DWORD PTR [1001114]
```

Similarly, we have 1 JMP other as follows:

```
00910022 68 791A0000 PUSH 1A79 <--
```

```
00910027 A1 0C009000 MOV EAX, DWORD PTR DS: [90000C] <- 00001991 =
```

```
0091002C 310424 XOR DWORD PTR SS: [ESP], EAX <- xor 00001991 = 1A79
```

```
0091002F 58 POP EAX
```

```
00910030 - E9 61196F00 JMP notepad_.01001996
```

So the command is equivalent to:

```
MOV EAX, 3E8
```

And specifically, we have:

```
00910035 50 PUSH EAX
```

```
00910036 8F05 D0A40001 POP DWORD PTR DS: [100A4D0]
```

```
0091003C - E9 5A196F00 JMP notepad_.0100199B
```

Equivalent to:

```
MOV DWORD PTR DS: [100A4D0], EAX
```

1 again:

```
009100B6 83EC 04 SUB ESP, 4
```

```
009100B9 50 PUSH EAX
```

```
009100BA A1 38980001 MOV EAX, DWORD PTR DS: [1009838]
```

```
009100BF 894424 04 MOV DWORD PTR SS: [ESP +4], EAX
```

```
009100C3 58 POP EAX
```

```
009100C4 - E9 26196F00 JMP notepad_.010019EF
```

Equivalent to:

```
PUSH DWORD PTR [1009838]
```

....

So, with 1 target we simply can completely fix = hands. However, the number of instructions indicated as much to fix the hand = requires your patience. Still have 1 option for you is writing the script 1, listed all types and fix automatically. This would cede to you and wish you success, lolz.

----- Stolen bytes case:

Options:

Anticracking Protection

+ Convert to OEP Virtual Machine

In this case, we only look at the "Convert OEP to Virtual Machine" the protection of other options do we check to focus more.

Implementation unpack UPX similar, we stop at:

```
0102468A 61 POPAD
0102468B 8980 A3440000 MOV DWORD PTR DS: [EAX +44 A3], EAX <- Stolen
OEP VM
01024691 8998 A7440000 MOV DWORD PTR DS: [EAX +44 A7], EBX
01024697 8988 AB440000 MOV DWORD PTR DS: [EAX +44 AB], ECX
0102469D 8990 AF440000 MOV DWORD PTR DS: [EAX +44 AF], EDX
010246A3 89B0 B3440000 MOV DWORD PTR DS: [EAX +44 B3], ESI
010246A9 89B8 B7440000 MOV DWORD PTR DS: [EAX +44 B7], EDI
010246AF 89A0 BF440000 MOV DWORD PTR DS: [EAX +44 BF], ESP
010246B5 89A0 C3440000 MOV DWORD PTR DS: [EAX +44 C3], ESP
010246BB 89A8 BB440000 MOV DWORD PTR DS: [EAX +44 BB], EBP
010246C1 8BE8 MOV EBP, EAX
010246C3 8BBD 9F440000 MOV EDI, DWORD PTR SS: [EBP +449 F]
010246C9 8BB5 9B440000 MOV ESI, DWORD PTR SS: [EBP +449 B]
010246CF E9 5A040000 JMP notepad_.01024B2E
```

And here, if trace xú 1, we will like to OEP old, and then we will see in OEP 0100739D is 1 of the order as NOP Moth above. So, is 0102468B predict the location of the VM Stolen OEP. Here, there are 2 ways to process, as you choose

OEP 1.Dump in VM, and to insert more VMSection.

From this position, we dump file, and fix IAT, is selected EP 0102468B = VM OEP. After the dump file and fix the dumped_1.exe. If Debug dumped_1.exe this file, we will find the cost mga record the EAX first is wrong, it will cause the Access bugs Error. Therefore, we fix the following:

```
01024686> b8 D9380201 MOV EAX, notepad_.010238D9
```

```
0102468B 8980 A3440000 MOV DWORD PTR DS: [EAX +44 A3], EAX
```

```
01024691 8998 A7440000 MOV DWORD PTR DS: [EAX +44 A7], EBX
```

Edit the file's new EP dumped_1.exe is 01024686.

And also continue to trace, we will see dumped_1.exe access to 1 duo.c other memory areas specified in the ESI and EDI = 960000 = 940000

```
010246C3 8BBD 9F440000 MOV EDI, DWORD PTR SS: [EBP +449 F]
```

```
010246C9 8BB5 9B440000 MOV ESI, DWORD PTR SS: [EBP +449 B]
```

Therefore, we must dump 2 in this area remember the original file, and import it into dumped_1.exe (remember to edit VAddress accurate).

After you add more -> file to run well.

2.Recover stolen bytes (for someone like "beauty").

The process of handling stolen bytes of RlPack beginning:

```
01024B2E 833E 00 Cmp DWORD PTR DS: [ESI], 0
```

```
01024B31 ^ 0F85 9DFBFFFF JNZ notepad_.010246D4
```

```
01024B37 8BB5 9F440000 MOV ESI, DWORD PTR SS: [EBP +449 F]
```

```
01024B3D EB 05 JMP SHORT notepad_.01024B44
```

```
01024B3F FF36 PUSH DWORD PTR DS: [ESI]
```

[ESI] will contain VM-Instruction

[EDI] stack memory

The value may be recorded in

```
0102468B 8980 A3440000 MOV DWORD PTR DS: [EAX +44 A3], EAX
```

```
01024691 8998 A7440000 MOV DWORD PTR DS: [EAX +44 A7], EBX
```

```
01024697 8988 AB440000 MOV DWORD PTR DS: [EAX +44 AB], ECX
```

```
0102469D 8990 AF440000 MOV DWORD PTR DS: [EAX +44 AF], EDX
```

```
010246A3 89B0 B3440000 MOV DWORD PTR DS: [EAX +44 B3], ESI
```

```
010246A9 89B8 B7440000 MOV DWORD PTR DS: [EAX +44 B7], EDI
```

```
010246AF 89A0 BF440000 MOV DWORD PTR DS: [EAX +44 BF], ESP
```

```
010246B5 89A0 C3440000 MOV DWORD PTR DS: [EAX +44 C3], ESP
```

```
010246BB 89A8 BB440000 MOV DWORD PTR DS: [EAX +44 BB], EBP
```

And it will be assigned again before returning to real OEP.

```
01024B75 8B85 A3440000 MOV EAX, DWORD PTR SS: [EBP +44 A3]
```

```
01024B7B 8B9D A7440000 MOV EBX, DWORD PTR SS: [EBP +44 A7]
```

```
01024B81 8B8D AB440000 MOV ECX, DWORD PTR SS: [EBP +44 AB]
```

```
01024B87 8B95 AF440000 MOV EDX, DWORD PTR SS: [EBP +44 AF]
```

```
01024B8D 8BB5 B3440000 MOV ESI, DWORD PTR SS: [EBP +44 B3]
```

```
01024B93 8BBD B7440000 MOV EDI, DWORD PTR SS: [EBP +44 B7]
```

```
01024B99 8BA5 BF440000 MOV ESP, SS DWORD PTR [EBP +44 BF]
```

```
01024B9F 8BAD BB440000 MOV EBP, DWORD PTR SS: [EBP +44 BB]
```

```
01024BA5 ^ E9 A3EFFFFFFF JMP notepad_.01023B4D
```


Corresponding to every instruction from the VM-1 Instruction will own the process called handler.

In this target, we have a review around 23 handler in turn is handler_00, handler_01 ..., handler_23

Example No. 1 handler:

Handler_00:

```
01024B2E 833E 00 Cmp DWORD PTR DS: [ESI], 0
01024B31 ^ 0F85 9DFBFFFF JNZ notepad_.010246D4
01024B37 8BB5 9F440000 MOV ESI, DWORD PTR SS: [EBP +449 F]
01024B3D EB 05 JMP SHORT notepad_.01024B44
```

Handler_01:

```
010246D4 833E 01 Cmp DWORD PTR DS: [ESI], 1
010246D7 75 1E JNZ SHORT notepad_.010246F7
010246D9 60 PUSHAD
010246DA 33C9 XOR ECX, ECX
010246DC 6A 08 PUSH 8
010246DE E8 620C0000 CALL notepad_.01025345
010246E3 50 PUSH EAX
010246E4 6A 07 PUSH 7
010246E6 E8 220A0000 CALL notepad_.0102510D
010246EB 897C24 1C MOV DWORD PTR SS: [ESP +1 C], EDI
010246EF 61 POPAD
010246F0 8BF8 MOV EDI, EAX
010246F2 E9 34040000 JMP notepad_.01024B2B
```

Handler_02:

```
010246F7 833E 02 Cmp DWORD PTR DS: [ESI], 2
010246FA 75 17 JNZ SHORT notepad_.01024713
010246FC 60 PUSHAD
010246FD 33C9 XOR ECX, ECX
010246FF FF76 04 PUSH DWORD PTR DS: [ESI +4]
01024702 E8 220E0000 CALL notepad_.01025529
01024707 897C24 1C MOV DWORD PTR SS: [ESP +1 C], EDI
0102470B 61 POPAD
0102470C 8BF8 MOV EDI, EAX
0102470E E9 18040000 JMP notepad_.01024B2B
```

Thus, to restore, then we must find hard to bear out this Handler 23 respectively as I do, and then perform the recovery process.

Here I will for example in the first instruction

```
0100739D> $ 6A 70 PUSH 70 <---
0100739F. 68 98180001 PUSH notepad_.01001898
```

010073A4. E8 BF010000 CALL notepad_.01007568

Set in 01024B2E BP is beginning process of all the handler, and F9, we will stop:

When the value of [ESI] is

DS: [00960000] = 00000002

So, is VM_Instruction 02 -> Handler_02, break in Handler_02 = 010246F7, we will:

010246F7 833E 02 Cmp DWORD PTR DS: [ESI], 2 <- PUSH OPCODE

010246FA 75 17 JNZ SHORT notepad_.01024713

010246FC 60 PUSHAD

010246FD 33C9 XOR ECX, ECX

010246FF FF76 04 PUSH DWORD PTR DS: [ESI +4] <- = 00000070 <- PUSH
VALUE

01024702 E8 220E0000 CALL notepad_.01025529 <- Save the record bar

01024707 897C24 1C MOV DWORD PTR SS: [ESP +1 C], EDI <- stack

0102470B 61 POPAD

0102470C 8BF8 MOV EDI, EAX <- stack current

0102470E E9 18040000 JMP notepad_.01024B2B <- continue through next VM-
Instruction

So I guess the VM-02 Instruction is PUSH

Unpacking SafeDisc 2.x - Trans by hacnho

Goal: The pure research, uses the discussion For a long time before .=.=' note

Goal: -> SD.EXE

(1) uses the tool, the examination, the discovery is

SsfeDisc 2.x

(2) S-ICE starts, supposes break point BPX GetVersion

After (3) returns to the master routine, found the entrance OEP

482302-PUSH EBP <----- OEP

482303-MOV EBP, ESP

482305 PUSH-FF

482307 PUSH-00466257

...

...

482302A-CALL [006F4320] ---> GetVersion

(4) DPLAYERX.DLL MAP32, SAFEDISC ==> decodes API

12BF4CB-61 75 0F -> Changes JMP Patch

12BF4F1 EB-917 -> Changes JMP Patch

(5) 0 = code corresponds 7D, 1 = 23 (examines MEMORY); U 120A270 (beginning)

(6) looks for the gap, reads in patch, but also? Shy; Import table

The attention, the code 0, wants the code 1 RUN 1 respectively,'s OK

Patch:

Loop: PUSH 9413FFBE

PUSH EBX

PUSH DWORD PTR 0; Code = 0

CALL [0013AF86]; decodes the API call

ADD ECX, EDX, ECX = search value

MOV EAX, [EBP +10]; True position API site

MOV EDX, 6F4000; IAT START

N0: Cmp DWORD PTR [EDX], ECX

JZ N1


```
INC EDX
Cmp EDX, 6F6000; IAT END
JZ N2
JMP N0
N1: MOV DWORD PTR [EDX], EAX
N2: INC EBX
Cmp EBX, 7D; Corresponding code ==> 7D
JNZ Loop
END: JMP END
```

(7) RUN LordPE => FULL dump ==> SD.EXE puts on file SD1.EXE

(8) RUN ImportREC inputs to revise OEP IAT, puts on file SD1.EXE

(9) TEST RUN SD1.EXE ... OK ~!!!! stopping work for the day



SafeDisc - Easy or Hard

Vol 1 - Bypass AntiDebug Find and OEP

I-Intro:

Anti-II Bypass Debugger & Find OEP:

I-Intro:

Rain and then sunshine again, human disease and the provinces, as well as protector of the meat and then upgrade again ... and again ... and then to the meat ...

Kì mạn this trick allowed them to introduce some pa Kon 1 small way to unpack Safedisc. Ah, say the right of Safedisc 1 story so long ago they would diffusion few lines.

From time immemorial, at which the game requires any disc into the CD was born file NoCD crack. And the time that the crack they guess the only tool we need the next W32DASM is enough meat but not to bother in the SoftIce. The parties should make the game to respond with new technologies protect high style. The tool pack the executable file of the game again, then write the content to unpack the file to secure the sector on the CD. So you want to play games we must have the original CD player is new, because the secure sector can not copy the style is generally.

Fortunately in the modern belgium hours, with some tool such as Alcohol 120%,

Deamon Tool, we can create and run the ISO file identical as the original CD (of course who created the ISO must have the CD 1 original). But do not be time to create the ISO also successful, so files crack NoCD will continue to be popular.

And most of them are aged Cracker Russia, including that balance has what you just bought this aged Lia lia disc would sit on the unpack, patch noCD, then share extensive network for pa Kon xài pagoda. Can say almost any game grid then will have "to" ...

Now, i sure that only the world in our group also when Cracker lot, but the party fold our hands just sitting ngo but also difficult to research that forms Protect CD, the reason is quite simple "hem with money would buy a disc" should be ... this is no reason 1 items instructions clearly to this protector of meat (Ah, if there is any particular research or writing underground underground wear the shelves of tools, the tools for embracing the private Đồng down discoloration nhé:)) ...)

However it is up to date, the last trick they also contemplate some tut 1 super wan, perhaps in the knowledge tut kĩ were too old compared to the first of several Russian-aged, but are still more than a warm formula to death. And now she would write again in what has been ngô in days: methods to unpack Safedisc - a protector of the easiest type protector (if compared with Securom or Starforce)

Version Safedisc kì to belong in this range 3.x - 4.x

Target: 2 games following is recommended: (1 of 2)

Raiden 3 +

+ Need For Speed Underground 2 (NFSU 2)

With NSFU 2, if the download is you can easily earn in the warez sites, not in the you can buy goods on the 2 CD set that practice.

The CD is the original resolution, how? You can on <http://gamecopyworld.com/> for Fixed Image files of this game. To mount with Deamon Tool to create 1 CD of the original (problem solving decrypt secure sector on CD)

With raiden 3 1 rare cases, to protect this Option All of Safedisc but do need to run the CD Original trick so choose to target it as illustrated in.

Unpack Safedisc will be divided into several items for each type protect, this will go to the first, the Bypass Anti Debugger by Safedisc and OEP.

Anti-II Bypass Debugger & Find OEP:

We only need 1 copy Olly super "cui" (the original is) to bypass AntiDebug. Now load the file "Raideen3.exe" to:

Address	Hex dump	Disassembly
007E609E	55	PUSH EBP
007E609F	8BEC	MOV EBP,ESP
007E60A1	60	PUSHAD
007E60A2	BB 9E607E00	MOV EBX,OFFSET Raideen3.<ModuleEntryPoint>
007E60A7	33C9	XOR ECX,ECX
007E60A9	8A00 3D607E00	MOV CL,BYTE PTR DS:[7E603D]
007E60AF	85C9	TEST ECX,ECX
007E60B1	74 0C	JE SHORT Raideen3.007E60BF
007E60B3	B8 13617E00	MOV EAX,Raideen3.007E6113
007E60B8	2BC3	SUB EAX,EBX
007E60BA	83E8 05	SUB EAX,5
007E60BD	EB 0E	JMP SHORT Raideen3.007E60CD

OEP by Safedisc is nothing difficult, since the EP you drag down 1 billion until found 1 LONG JUMP:

007E6144	FF30	PUSH DWORD PTR DS:[EAX]
007E6146	8B40 04	MOV EAX,DWORD PTR DS:[EAX+4]
007E6149	FFD0	CALL EAX
007E614B	58	POP EAX
007E614C	FF35 53617E00	PUSH DWORD PTR DS:[7E6153]
007E6152	C3	RETN
007E6153	72 16	JB SHORT Raideen3.007E616B
007E6155	61	POPAD
007E6156	1360 00	ADD ESP,DWORD PTR DS:[EAX+0]
007E6159	E9 124CD3FF	JMP Raideen3.0051AD70
007E615F	CC	INT3
007E6160	81EC E8020000	SUB ESP,2E8
007E6166	53	PUSH EBX
007E6167	55	PUSH EBP
007E6168	56	PUSH ESI

So OEP = 51AD70. Hehe, you just set up here BP (encourage use HWBP) is 1 time later, when the code was the unpack, JUMP command will take us to fly directly to OEP.

The issue now is to bypass the Anti Debugger. Safedisc first detect through

IsDebuggerPresent,

Address	Hex dump	Disassembly
7C8130B3	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]
7C8130B9	8B40 30	MOV EAX,DWORD PTR DS:[EAX+30]
7C8130BC	0FB640 02	MOVZX EAX,BYTE PTR DS:[EAX+2]
7C8130C0	C3	RETN
7C8130C1	90	NOP
7C8130C2	90	NOP

Leave to BP, for about RETN:

Address	Hex dump	Disassembly
6670C70C	8BF0	MOV ESI,EAX
6670C70E	66:85F6	TEST SI,SI
6670C711	74 13	JE SHORT 6670C726
6670C713	E8 624BFFFF	CALL 6670127A
6670C718	66:8BF0	MOV SI,AX
6670C71B	66:F7DE	NEG SI
6670C71E	1BF6	SBB ESI,ESI
6670C720	46	INC ESI
6670C721	66:85F6	TEST SI,SI
6670C724	75 13	JNZ SHORT 6670C739
6670C726	8B4424 08	MOV EAX,DWORD PTR SS:[ESP+8]
6670C72A	8B08	MOV ECX,DWORD PTR DS:[EAX]

If the RETN, EAX = 1 then that is our debugger has been detect, we just fix it is 0 we okie. Gio CTRL-F9 to return to:

Address	Hex dump	Disassembly
667047B4	56	PUSH ESI
667047B5	E8 367F0000	CALL 6670C6F0
667047BA	83C4 04	ADD ESP,4
667047BD	66:85C0	TEST AX,AX
667047C0	0F85 02010000	JNZ 667048C8
667047C6	F6C3 04	TEST BL,4
667047C9	76 12	JBE SHORT 667047DD
667047CB	56	PUSH ESI
667047CC	E8 1F7E0000	CALL 6670C5F0
667047D1	83C4 04	ADD ESP,4
667047D4	66:85C0	TEST AX,AX
667047D7	0F85 EB000000	JNZ 667048C8
667047DD	F6C3 08	TEST BL,8
667047E0	76 12	JBE SHORT 667047F4
667047E2	56	PUSH ESI
667047E3	E8 387D0000	CALL 6670C520
667047E8	83C4 04	ADD ESP,4
667047EB	66:85C0	TEST AX,AX
667047EE	0F85 D4000000	JNZ 667048C8
667047F4	F7C3 00080000	TEST EBX,800
667047FA	76 12	JBE SHORT 6670480E
667047FC	56	PUSH ESI
667047FD	E8 3E7C0000	CALL 6670C440
66704802	83C4 04	ADD ESP,4
66704805	66:85C0	TEST AX,AX
66704808	0F85 02010000	JNZ 667048C8

Rub, **IsDebuggerPresent** the outside, 1 Dong, there is another check to detect our debugger. If you view the items by haggar, you will know the name of some functions that used to detect Safedisc. Here i need a long line of fact dollop chả check this very important what we continue to Ctrl-F9 here:

Address	Hex dump	Disassembly
667013F9	83C4 08	ADD ESP,8
667013FC	66:8945 0C	MOV WORD PTR SS:[EBP+C],AX
66701400	58	POP EAX
66701401	66:8B45 0C	MOV AX,WORD PTR SS:[EBP+C]
66701405	66:F7D8	NEG AX
66701408	1BC0	SBB EAX,EAX
6670140A	66:25 0040	AND AX,4000
6670140E	03C6	ADD EAX,ESI
66701410	5E	POP ESI
66701411	C9	LEAVE
66701412	C3	RET
66701413	FF	CALL EBX

After Đồng check there, AX will be calculated at least 1. Continue Ctrl-F9:

Address	Hex dump	Disassembly
66703028	59	POP ECX
66703029	3D 00200000	CMP EAX,2000
6670302E	59	POP ECX
6670302F	74 23	JE SHORT 66703054
66703031	3D 00400000	CMP EAX,4000
66703036	74 D9	JE SHORT 66703011
66703038	33C9	XOR ECX,ECX
6670303A	3D 00000100	CMP EAX,10000
6670303F	0F94C1	SETC CL
66703042	49	DEC ECX
66703043	80E1 6A	AND CL,6A
66703046	81C1 FA000000	ADD ECX,0FA
6670304C	890D 94CC7E66	MOV DWORD PTR DS:[667ECC94],ECX
66703052	C9	LEAVE
66703053	C3	RET
66703054	C705 94CC7E66 FB000000	MOV DWORD PTR DS:[667ECC94],0FB
6670305E	C9	LEAVE
6670305F	C3	RET

Now if your EAX other 10,000 that is the debugger has been detected. This may be the last of the anonymous messages, but with Olly kĩ to hide the trick, also found itself in the EAX is 10,000:

```

Registers (FPU)
EAX 00010000 l
ECX 00000040
EDX 7C90EB94 r
EBX 0012FCB8
ESP 0012FB10
EBP 0012FB1C
ESI 66700000
EDI 00000000
EIP 66703028

```

So if your EAX = 2000 or 4000 then revised in the 10,000. Note that not only edit or Z flag patch for orders not skip jump nhé, if not with EAX 10,000 jump that was so, the error will occur.

The bypass is to be able to speak is too succinct, afedisc ung check how many but gom streamlined result of the check to EAX so we just need to fix such a presence through it.

Expand for you, Safedisc have 1 brother who is Safecast - used to protect soft (not games), and if it is safecast after bypass such as we have been completely out of the anti Safecast. However, with Safedisc is not necessarily, if you press Shift - F9 will now see that in BP LONG JUMP OEP not to break? Not have any notice of any detect, Olly not crash, as in the process evolve?

1 Make sure that something is not necessarily anti debugger, Okie, we restart everything again, bypass the anti as above, but this time not to run again but vẹo vẹo CTRL-F9 to return here:

Address	Hex dump	Disassembly
66703376	6A 32	PUSH 32
66703378	6A 6E	PUSH 6E
6670337A	E8 08F30100	CALL 66722687
6670337F	BE F4010000	MOV ESI,1F4
66703384	56	PUSH ESI
66703385	6A 64	PUSH 64
66703387	E8 FBF20100	CALL 66722687
6670338C	8D45 D4	LEA EAX,DWORD PTR SS:[EBP-2C]
6670338F	50	PUSH EAX
66703390	8D45 D8	LEA EAX,DWORD PTR SS:[EBP-28]
66703393	50	PUSH EAX
66703394	57	PUSH EDI
66703395	6A 03	PUSH 3
66703397	FF75 EC	PUSH DWORD PTR SS:[EBP-14]
6670339A	E8 C1FCFFFF	CALL 66703060
6670339F	83C4 44	ADD ESP,44
667033A2	56	PUSH ESI
667033A3	6A 6E	PUSH 6E

The process may evolve as 1 endless loop, so the only way we trace from CALL to find out what makes the stiff suspension. Find CALL this form as well as find the call CALL trial NAG, in turn Ctrl - F8 ... and we are outstretched here:

Address	Hex dump	Disassembly
667036F1	> F645 F0 02	TEST BYTE PTR SS:[EBP-10],2
667036F5	. 75 09	JNZ SHORT ~df394b.66703700
667036F7	. FF75 F8	PUSH DWORD PTR SS:[EBP-8]
667036FA	. E8 981B0700	CALL ~df394b.66775297
667036FF	. 59	POP ECX
66703700	. FF75 EC	PUSH DWORD PTR SS:[EBP-14]
66703703	. E8 F8480C00	CALL ~df394b.667C8000
66703708	. 8B43 14	MOV EAX,DWORD PTR DS:[EBX+14]
6670370B	. 59	POP ECX
6670370C	. 8B40 35	MOV EAX,DWORD PTR DS:[EAX+35]
6670370F	. A3 040E8066	MOV DWORD PTR DS:[66800ED4],EAX
66703714	. E8 E2CA0400	CALL ~df394b.667501FB
66703719	. FF73 10	PUSH DWORD PTR DS:[EBX+10]
6670371C	. 57	PUSH EDI
6670371D	. 6A 01	PUSH 1
6670371F	. 68 18CE7E66	PUSH ~df394b.667ECE18
66703724	. FF15 60B07C66	CALL DWORD PTR DS:[&KERNEL32.CreateEventA]
6670372A	. 9AC7	CMP FAX,FNT

Set HWBP here, then Restart, the Anti bypass, run and break in on CALL. F7 to trace to here:

Address	Hex dump	Disassembly
667C8000	55	PUSH EBP
667C8001	8BEC	MOV EBP,ESP
667C8003	81EC 10020000	SUB ESP,210
667C8009	53	PUSH EBX
667C800A	E8 B994F3FF	CALL ~df394b.667014C8
667C800F	85C0	TEST EAX,EAX
667C8011	> 0F84 EF000000	JE ~df394b.667C8106
667C8017	E8 C895F3FF	CALL ~df394b.667015E4
667C801C	66:3D 0100	CMP AX,1
667C8020	> 0F84 E0000000	JE ~df394b.667C8106
667C8026	FF15 58B07C66	CALL DWORD PTR DS:[&KERNEL32.GetCurrentProcessId]
667C802C	8D8D F4FEFFFF	LEA ECX,DWORD PTR SS:[EBP-10C]
667C8032	68 04010000	PUSH 104
667C8037	51	PUSH ECX

Remember to take this command Jump, continued from the trace:

Address	Hex dump	Disassembly
667C80BE	FFD3	CALL EBX
667C80C0	FF75 FC	PUSH DWORD PTR SS:[EBP-4]
667C80C3	FFD6	CALL ESI
667C80C5	E8 8684F4FF	CALL ~df394b.66710550
667C80CA	> FF75 FC	PUSH DWORD PTR SS:[EBP-4]
667C80CD	FF15 64B07C66	CALL DWORD PTR DS:[&KERNEL32.SetEvent]
667C80D3	85C0	TEST EAX,EAX
667C80D5	> 75 0C	JNZ SHORT ~df394b.667C80E3
667C80D7	FFD3	CALL EBX
667C80D9	FF75 FC	PUSH DWORD PTR SS:[EBP-4]
667C80DC	FFD6	CALL ESI
667C80DE	E8 6D84F4FF	CALL ~df394b.66710550
667C80E3	> 6A FF	PUSH -1
667C80E5	57	PUSH EDI
667C80E6	FF15 90B07C66	CALL DWORD PTR DS:[&KERNEL32.WaitForSingleObject]
667C80EC	FF75 FC	PUSH DWORD PTR SS:[EBP-4]
667C80EF	8945 F8	MOV DWORD PTR SS:[EBP-8],EAX
667C80F2	FFD6	CALL ESI
667C80F4	FF	PUSH FNT

Dom's on the see, with the number **-1** (or FFFFFFFF), is to PUSH, **WaitForSingleObject** function will make the current process hangs if debug is by Olly. For details on the wait time, **-1** is equivalent to Infinite Time. Process currently process for 1 second 2 (tentatively called the safedisc debugger) send it back to 1 signal to the contact between 2 process, here we are again Olly bulkhead to make the wait should not have to A, with the waiting period is Infinite should lead to repeats endlessly. But process 2 Where? Review process will see the list now:

OllyICE.exe	2296	allyDbg. 32-bit analysing deb..
Raiden3.exe	1908	
~e5.0001	996	Cleanup

So to bypass this, we have 2 hours. But remember we ordered Jump trace through the above,

if khúc jump in on, we will wait past the Infinite Time. With 2 minutes, we fix parameters PUSH PUSH -1 is 0 to shorten the waiting time down to 0 milisecond. Then trace and jump out through this:

Address	Hex	dump	Disassembly
667C80E6	FF15	90B07C66	CALL DWORD PTR DS:[<&KERNEL32.WaitForSingleObject>
667C80EC	FF75	FC	PUSH DWORD PTR SS:[EBP-4]
667C80EF	8945	F8	MOV DWORD PTR SS:[EBP-8],EAX
667C80F2	FFD6		CALL ESI
667C80F4	57		PUSH EDI
667C80F5	FFD6		CALL ESI
667C80F7	837D	F8 00	CMP DWORD PTR SS:[EBP-8],0
667C80FB	5F		POP EDI
667C80FC	5E		POP ESI
667C80FD	74	07	JF SHORT ~df394b.667C8106
667C80FF	FFD3		CALL EBX
667C8101	E8	4A84F4FF	CALL ~df394b.66710550
667C8106	5B		POP EBX
667C8107	C9		LEAVE
667C8108	C3		RET

Jump not here, they will fail. After the jump, we can secure Shift-F9 to OEP Jong JUMP:

007E6152	C3		RET
007E6153	72	16	JB SHORT Raiden3.007E616B
007E6155	61		POPAD
007E6156	1360	00	ADC ESP,DWORD PTR DS:[EAX+D]
007E6159	E9	124CD3FF	JMP Raide3.0051AD70
007E615E	CC		INT3
007E615F	CC		INT3
007E6160	81EC	E8020000	SUB ESP,2E8
007E6166	53		PUSH EBX
007E6167	55		PUSH EBP
007E6168	56		PUSH ESI
007E6169	57		PUSH EDI
007E616A	1F	00	IFD EAX,DWORD PTR SS:[ESP+60]

F7 to develop and occupies the threshold OEP:

Address	Hex	dump	Disassembly
0051AD70	6A	60	PUSH 60
0051AD72	68	00005400	PUSH Raide3.0054D000
0051AD77	E8	CC4B0000	CALL Raide3.0051F948
0051AD7C	BF	94000000	MOV EDI,94
0051AD81	8BC7		MOV EAX,EDI
0051AD83	E8	F8F1FFFF	CALL Raide3.00519F80
0051AD88	8965	E8	MOV DWORD PTR SS:[EBP-18],ESP
0051AD8B	8BF4		MOV ESI,ESP
0051AD8D	893E		MOV DWORD PTR DS:[ESI],EDI
0051AD8F	56		PUSH ESI
0051AD90	FF15	D4C05200	CALL DWORD PTR DS:[52C0D4]
0051AD96	8B4E	10	MOV ECX,DWORD PTR DS:[ESI+10]
0051AD99	890D	F4027E00	MOV DWORD PTR DS:[7E02F4],ECX
0051AD9F	8B46	04	MOV EAX,DWORD PTR DS:[ESI+4]
0051ADA2	A3	00037E00	MOV DWORD PTR DS:[7E0300],EAX

How is the complete purpose of this article, we temporarily boot thui. After this step, you have a lot to do with as Safedisc Restore IAT, Emulated Code fix, fix Wrong CALL IAT, fix LONG JUMP And especially khiến Nanomite fix.

See you in the following items.

Big thanks to:

All REA's members: Computer_Angel, Moonbaby, Zombie, hacnho, benina, kienmanowar, rongchaua, Deux, Merc, hoadongnoi, the_lighthouse, TQN, light.phoenix, hytkl, tlandn, hurt_heart, dzungltvn, Zoi, littleboy84, haule_nth, takada, Why not bar, iamidiot, Akira, dump, thienthandien, [kid], ...

Special thanks to:

fly, stephenteh, Gabri3l, MaDMAAn_H3rCuL3s, CondZero, Ricardo Narvaja + NCR, lena151,

haggar, ARTeam, snd, RES, CrackLatinos, all unpack.cn ... Authors who created tools and you.



Trick Xi News - 2007



SafeDisc - Easy or Hard

Vol 2 - Restore Full IAT

I-Analyze:

Coding II:

I-Analyze:

Welcome brother, we met again. Ki before, we stop at OEP:

Address	Hex dump	Disassembly
0051AD70	. 6A 60	PUSH 60 ← OEP
0051AD72	. 68 00005400	PUSH Raiden3.00540000
0051AD77	. E8 CC4B0000	CALL Raiden3.0051F948
0051AD7C	. BF 94000000	MOV EDI,94
0051AD81	. 8BC7	MOV EAX,EDI
0051AD83	. E8 F8F1FFFF	CALL Raiden3.00519F80
0051AD88	. 8965 E8	MOV DWORD PTR SS:[EBP-18],ESP
0051AD8B	. 8BF4	MOV ESI,ESP
0051AD8D	. 893E	MOV DWORD PTR DS:[ESI],EDI
0051AD8F	. 56	PUSH ESI
0051AD90	. FF15 D4C05200	CALL DWORD PTR DS:[52C0D4]
0051AD96	. 8B4E 10	MOV ECX,DWORD PTR DS:[ESI+10]
0051AD99	. 890D F4027E00	MOV DWORD PTR DS:[7E02F4],ECX
0051AD9F	. 8B46 04	MOV EAX,DWORD PTR DS:[ESI+4]
0051ADA2	. A3 00037E00	MOV DWORD PTR DS:[7E0300],EAX

Even as we see now 1 IAT CALL: **CALL [52C0D4]**. Follow dump in that it does
Memory Address:

Address	Value	Comment
0052C004	7C812AFE	kernel32.GetVersionExA
0052C008	7C80B994	kernel32.UnmapViewOfFile
0052C00C	7C812D76	kernel32.GetSystemInfo
0052C0E0	7C80AE4A	kernel32.IsProcessorFeaturePresent
0052C0E4	7C809B04	kernel32.VirtualFree
0052C0E8	7C809A71	kernel32.VirtualAlloc

Rui drag on the top of the table, dom are more import was "playing" oi:

Address	Value	Comment
0052C000	0142A56F	
0052C004	0142AA32	
0052C008	77DD7883	ADVAPI32.RegQueryValueExA
0052C00C	00000000	
0052C010	7228B126	DINPUT.DirectInputCreateA
0052C014	00000000	
0052C018	73F14C43	DSOUND.DirectSoundCreate
0052C01C	00000000	
0052C020	014278A7	
0052C024	01427D6A	
0052C028	0142822D	
0052C02C	014286F0	
0052C030	01428BB3	
0052C034	01429076	

Find References we value first

Address	Disassembly	Comment
00495EDB	CALL DWORD PTR DS:[52C000]	DS:[0052C000]=0142A56F

New Origin, always go into this with CALL F7, we decode to the IAT:

Address	Hex dump	Disassembly
0142A56F	68 9412EABF	PUSH BFEA1294
0142A574	9C	PUSHFD
0142A575	60	PUSHAD
0142A576	54	PUSH ESP
0142A577	68 AFA54201	PUSH 142A5AF
0142A57C	E8 399E3965	CALL "df394b.667C43BA"
0142A581	83C4 08	ADD ESP,8
0142A584	6A 00	PUSH 0
0142A586	58	POP EAX
0142A587	61	POPAD
0142A588	9D	POPFD
0142A589	C3	RETN

PUSH, POP chả mean anything, call the other is the content, go to:

Address	Hex dump	Disassembly
667C43BA	. 55	PUSH EBP
667C43BB	. 8BEC	MOV EBP,ESP
667C43BD	. 83EC 34	SUB ESP,34
667C43C0	. 53	PUSH EBX
667C43C1	. 56	PUSH ESI
667C43C2	. 57	PUSH EDI
667C43C3	. F0:FF05 DC007F66	LOCK INC DWORD PTR DS:[667F00DC]
667C43CA	✓ 74 0E	JE SHORT ~df394b.667C43DA
667C43CC	. 6A FF	PUSH -1
667C43CE	. FF35 E04B8466	PUSH DWORD PTR DS:[66844BE0]
667C43D4	. FF15 90B07C66	CALL DWORD PTR DS:[<&KERNEL32.WaitForSingleObject>]
667C43DA	✓ EB 0A	JNZ SHORT ~df394b.667C43E6
667C43DC	EB	DB EB
667C43DD	> 50	PUSH EAX
667C43DE	. 8BC5 ..	MOV EAX,EBP

F8 collude to here:

Address	Hex dump	Disassembly
667C44F1	✓ 74 39	JE SHORT ~df394b.667C452C
667C44F3	. 8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
667C44F6	. 83C0 24	ADD EAX,24
667C44F9	. 8945 D0	MOV DWORD PTR SS:[EBP-30],EAX
667C44FC	. 8B45 D0	MOV EAX,DWORD PTR SS:[EBP-30]
667C44FF	. 8B4D F4	MOV ECX,DWORD PTR SS:[EBP-C]
667C4502	. 8908	MOV DWORD PTR DS:[EAX],ECX
667C4504	. 8B45 D0	MOV EAX,DWORD PTR SS:[EBP-30]
667C4507	. 83C0 04	ADD EAX,4
667C450A	. 50	PUSH EAX
667C450B	. E8 CDB3FAFF	CALL ~df394b.6676F8DD
667C4510	. 59	POP ECX
667C4511	. F0:FF0D DC007F66	LOCK DEC DWORD PTR DS:[667F00DC]
667C4518	✓ 78 0C	JS SHORT ~df394b.667C4526
667C451A	. FF35 E04B8466	PUSH DWORD PTR DS:[66844BE0]
667C4520	. FF15 64B07C66	CALL DWORD PTR DS:[<&KERNEL32.SetEvent>]
667C4526	> 8B65 0C	MOV ESP,DWORD PTR SS:[EBP+C]
667C4529	. 61	POPAD
667C452A	. 9D	POPFD
667C452B	. C3	RET
667C452C	> 8B45 EC	MOV EAX,DWORD PTR SS:[EBP-14]
667C452F	. 8945 E0	MOV DWORD PTR SS:[EBP-20],EAX
667C4532	. 8D45 D8	LEA EAX,DWORD PTR SS:[EBP-28]
667C4535	. 50	PUSH EAX

English to children that it is about to jump that small. Also remember always code just below it, we jumped for it and trace back to here:

667C4689	. 8B45 CC	MOV EAX,DWORD PTR SS:[EBP-34]
667C468C	. 83C0 04	ADD EAX,4
667C468F	. 50	PUSH EAX
667C4690	. E8 48B2FAFF	CALL ~df394b.6676F8DD
667C4695	. 59	POP ECX
667C4696	. F0:FF0D DC007F66	LOCK DEC DWORD PTR DS:[667F00DC]
667C469D	✓ 78 0C	JS SHORT ~df394b.667C46AB
667C469F	. FF35 E04B8466	PUSH DWORD PTR DS:[66844BE0]
667C46A5	. FF15 64B07C66	CALL DWORD PTR DS:[<&KERNEL32.SetEvent>]
667C46AB	> 8B65 0C	MOV ESP,DWORD PTR SS:[EBP+C]
667C46AE	. 61	POPAD
667C46AF	. 9D	POPFD
667C46B0	. C3	RET
667C46B1	. F0:FF0D DC007F66	LOCK DEC DWORD PTR DS:[667F00DC]

Look wen wen with khúc not, na Ná same code that. When **RET** stop now, what do we see?


```

667C46A5 . FF15 64B07C66 CALL DWORD PTR DS:[<&KERNEL32.SetEvent>]
667C46A8 > 8B65 0C MOV ESP,DWORD PTR SS:[EBP+C]
667C46AE . 61 POPAD
667C46AF . 90 POPFD
667C46B0 . C3 RETN
667C46B1 . F0:FF00 DC007F66 LOCK DEC DWORD PTR DS:[667F00DC]
667C46B8 . 78 0C JS SHORT ~df394b.667C46C6

```

Address	Value	Comment
0012FFBC	77DD6BF0	ADVAPI32.RegCloseKey
0012FFC0	00495EE1	RETURN to Raiden3.00495EE1 frc
0012FFC4	7C816FF7	RETURN to kernel32.7C816FF7
0012FFC8	7C910738	ntdll.7C910738
0012FFCC	FFFFFFFF	
0012FFD0	7FFDB000	

Real value of import was located in the stack, you should return it to the equivalent command CALL 1 IAT me. From then also see the opening import sometimes greasy nhĩ serial? other other. Talking is always more in command Jump that when we stop now, it can not jump, going to return below it, but the value of its return to do the right value of real import. So only 2 code (the jump occurred at the) return of new value precisely. (if you do go back many times as a new draw this)

The value is also decode the same paragraph 1 above. So to restore the import table, I think will be 1 billion immediately.

II-Coding:

To start code, the core is the following:

- + In the trace decode until **RETN**, the value of EAX, EBX and ECX not change (3 can get this to work)
- + Central region decode **1420000** to **1450000**

003F0000	00001000	Raiden3		PE header	Map	RW	RW
00400000	00003000	Raiden3		code	Imag	R	RWE
00403000	00129000	Raiden3	.text	code	Imag	R	RWE
0052C000	00029000	Raiden3	.rdata	code	Imag	R	RWE
00555000	0028D000	Raiden3	.data	data	Imag	R	RWE
007E2000	00001000	Raiden3	.rsrc	resources	Imag	R	RWE
007E3000	00003000	Raiden3	stxt774		Imag	R	RWE
007E6000	00004000	Raiden3	stxt371	SFX, impo	Imag	R	RWE
007F0000	0000C000			Map	R	E	R
008B0000	00002000			Map	R	E	R
008C0000	00103000			Map	R		R
009D0000	00187000			Map	R	E	R
00CD0000	00008000			Priv	RW		RW
00DD0000	00001000			Priv	RW		RW
00E50000	0005A000			Priv	RW		RW
00ED1000	0000D000			Priv	RW		RW
00F50000	00001000			Map	RW		RW
00F60000	00001000			Map	RW		RW
00F70000	00003000			Priv	RWE		RWE
01370000	00051000			Map	R		R
013D0000	00001000			Map	RW		RW
013E0000	00010000			Map	RW		RW
01420000	0000B000			Priv	RWE		RW
01450000	00027000			Priv	RWE		RW
4F000000	00001000	D3D9		Imag	R		RWE
4F001000	0018A000	D3D9	.text	code, im	Imag	R	RWF

+ Start = IAT **52C000**

Address	Value	Comment
0052C000	0142A56F	
0052C004	0142AA32	
0052C008	77DD7883	ADVAPI32.RegQueryValueExA
0052C00C	00000000	

IAT + End = **52C238**

0052C228	76B44E5B	WINMM.timeGetTime
0052C22C	76B46154	WINMM.timeEndPeriod
0052C230	00000000	
0052C234	4FDFAED0	D3D9.Direct3DCreate9
0052C238	00000000	
0052C23C	00000000	

+ Where is the original value of **RETN**: (remember the code occurs when small jump)

667C469F	FF35 E04B8466	PUSH DWORD PTR DS:[66844BE0]
667C46A5	FF15 64B07C66	CALL DWORD PTR DS:[&KERNEL32.SetEvent>]
667C46AB	8B65 0C	MOV ESP,DWORD PTR SS:[EBP+C]
667C46AE	61	POPAD
667C46AF	9D	POPFD
667C46B0	C3	RETN
667C46B1	F0:FF0D DC007F66	LOCK DEC DWORD PTR DS:[667F00DC]

Okie, we have the following code to restore IAT (with the file. Asm in available).

- + Compile this code, then load the file has to compile Olly
- + Copy Binary Code
- + Search for (creating) a blank area in mind the target safedisc
- + Paste Binary Code through.

(code borrowed from 1 part *anonymous*, have to edit the match in progress)

Title imports

.386

. model small, stdcall

Option casemap: none

. code

_IB Equ 00400000h; ImageBase

_IATStart Equ 0012C000h; Start Offset of IAT

_IATSize Equ 00000238h; Size of IAT

_lowerrange equ 01420000h; Start Offset ` -vis the decode

_higherrange equ 01450000h; End Offset ` -vis the decode

start:

pushad

pushfd

mov eax, (_IB + _IATStart); IAT Start the EAX

_iloop:

mov ecx, dword ptr [eax]; chuyen value / (encode value) on ECX

Cmp ecx, _lowerrange; than offset sa'nh vo'i start ` -vis the decode

jl _next; ne'u be 'it `jump _next (i ca n` decode)

Cmp ecx, _higherrange; than offset sa'nh vo'i end of `vu decode

ja _next; ne'u lo'n it `jump _next (i ca n` decode)

_ireolve:

call ecx; Call decode at the value hie.n ta.i

; Go'c value is stored in EDX (POP EDX)

mov dword ptr [eax], edx; chuyen value go'c and `o in state import

_next:

add eax, 4; to import Ke 'tie'p

Cmp eax, (_IB + + _IATStart _IATSize); compared sa'nh elephant IAT End

je _iend, `If you test the` STOP restore IAT

jmp _iloop; do not contest the three `continue` restore IAT

_iend:

popfd

popad

nop; BP set here to Break

nop

end **start**

After Paste wa area empty, we do drill with the above. Back to this position:

667C469F	FF35 E04B8466	PUSH DWORD PTR DS:[66844BE0]
667C46A5	FF15 64B07C66	CALL DWORD PTR DS:[&KERNEL32.SetEvent]
667C46AB	8B65 0C	MOV ESP,DWORD PTR SS:[EBP+C]
667C46AE	61	POPAD
667C46AF	9D	POPFD
667C46B0	C3	RET
667C46B1	FF0D DC007E66	LOCK DEC DWORD PTR DS:[667F00DC]
667C46B2	70 0C	JNB SHORT 667C46C2

Edit to:

667C46A5	8B65 0C	MOV ESP,DWORD PTR SS:[EBP+C]
667C46AE	61	POPAD
667C46AF	9D	POPFD
667C46B0	5A	POP EDI
667C46B1	C3	RET
667C46B2	FF0D DC007E66	DEC DWORD PTR DS:[667F00DC]
667C46B3	70 0C	JNB SHORT 667C46C3

Set the Write (or Full Access) for the **section. Rdata** (restore the IAT), the code enforcement is often the write memory in the other.

Set the BP location in the Source.

Address	Hex dump	Disassembly
00F70000	60	PUSHAD
00F70001	9C	PUSHFD
00F70002	B8 00C05200	MOV EAX,52C000
00F70007	8B08	MOV ECX,DWORD PTR DS:[EAX]
00F70009	81F9 00004201	CMP ECX,1420000
00F7000F	7C 0C	JL SHORT 00F7001D
00F70011	81F9 00004501	CMP ECX,1450000
00F70017	77 04	JA SHORT 00F7001D
00F70019	FFD1	CALL ECX
00F7001B	8910	MOV DWORD PTR DS:[EAX],EDI
00F7001D	83C0 04	ADD EAX,4
00F70020	3D 38C25200	CMP EAX,52C238
00F70025	74 02	JE SHORT 00F70029
00F70027	EB 0E	JMP SHORT 00F70007
00F70029	9D	POPFD
00F7002A	61	POPAD
00F7002B	90	NOF
00F7002C	90	NOF
00F7002D	0000	0000

New Set Origin, F9 break and run, complete, IAT was full:

Address	Value	Comment
0052C000	77DFC41B	ADVAPI32.RegOpenKeyA
0052C004	77DD6BF0	ADVAPI32.RegCloseKey
0052C008	77DD7883	ADVAPI32.RegQueryValueExA
0052C00C	00000000	
0052C010	7228B126	DINPUT.DirectInputCreateA
0052C014	00000000	
0052C018	73F14C43	DSOUND.DirectSoundCreate
0052C01C	00000000	
0052C020	77F16E6F	GDI32.DeleteDC
0052C024	77F16C0A	GDI32.DeleteObject
0052C028	77F1E2E3	GDI32.CreateFontIndirectA
0052C02C	77F15EEB	GDI32.SetBkMode
0052C030	77F15B80	GDI32.SelectObject
0052C034	77F15D87	GDI32.SetTextColor
0052C038	77F15E39	GDI32.SetBkColor
0052C03C	77F19219	GDI32.CreateDIBSection
0052C040	77F15FF0	GDI32.CreateCompatibleDC
0052C044	00000000	
0052C048	7C801A24	kernel32.CreateFileA
0052C04C	7C80180E	kernel32.ReadFile
0052C050	7C810DA7	kernel32.WriteFile
0052C054	7C859F78	kernel32.OutputDebugStringA
0052C058	7C810BAE	kernel32.SetFilePointer
0052C05C	7C831E35	kernel32.DeleteFileA
0052C060	7C80FEA2	kernel32.GlobalUnlock
0052C064	7C80FC4F	kernel32.GlobalFree
0052C068	7C80FD4D	kernel32.GlobalAlloc
0052C06C	7C80FF39	kernel32.GlobalLock
0052C070	7C80E95F	kernel32.CreateMutexA
0052C074	7C910331	ntdll.RtlGetLastWin32Error
0052C078	7C833400	kernel32.SetEnvironmentVariableA

Later want to use the code for another target, we simply find the number needed to change the source is now expendable.

However Import full board is still on scoundrelly. The reason? See the following items.

Big thanks to:

All REA's members: Computer_Angel, Moonbaby, Zombie, hacnho, benina, kienmanowar, rongchaua, Deux, Merc, hoadongnoi, the_lighthouse, TQN, light.phoenix, hytkl, tlandn, hurt_heart, dzungltvn, Zoi, littleboy84, haule_nth, takada, Why not bar, iamidiot, Akira, dump, thienthandien, [kid], ...

Special thanks to:

fly, stephente, Gabri3l, MaDMAN_H3rCuL3s, CondZero, Ricardo Narvaja + NCR, lena151, haggar, ARTeam, snd, RES, CrackLatinos, all unpack.cn ... Authors who created tools and you.



Trick Xi News - 2007



SafeDisc - Easy or Hard

Vol 3 - Fix commands Stolen

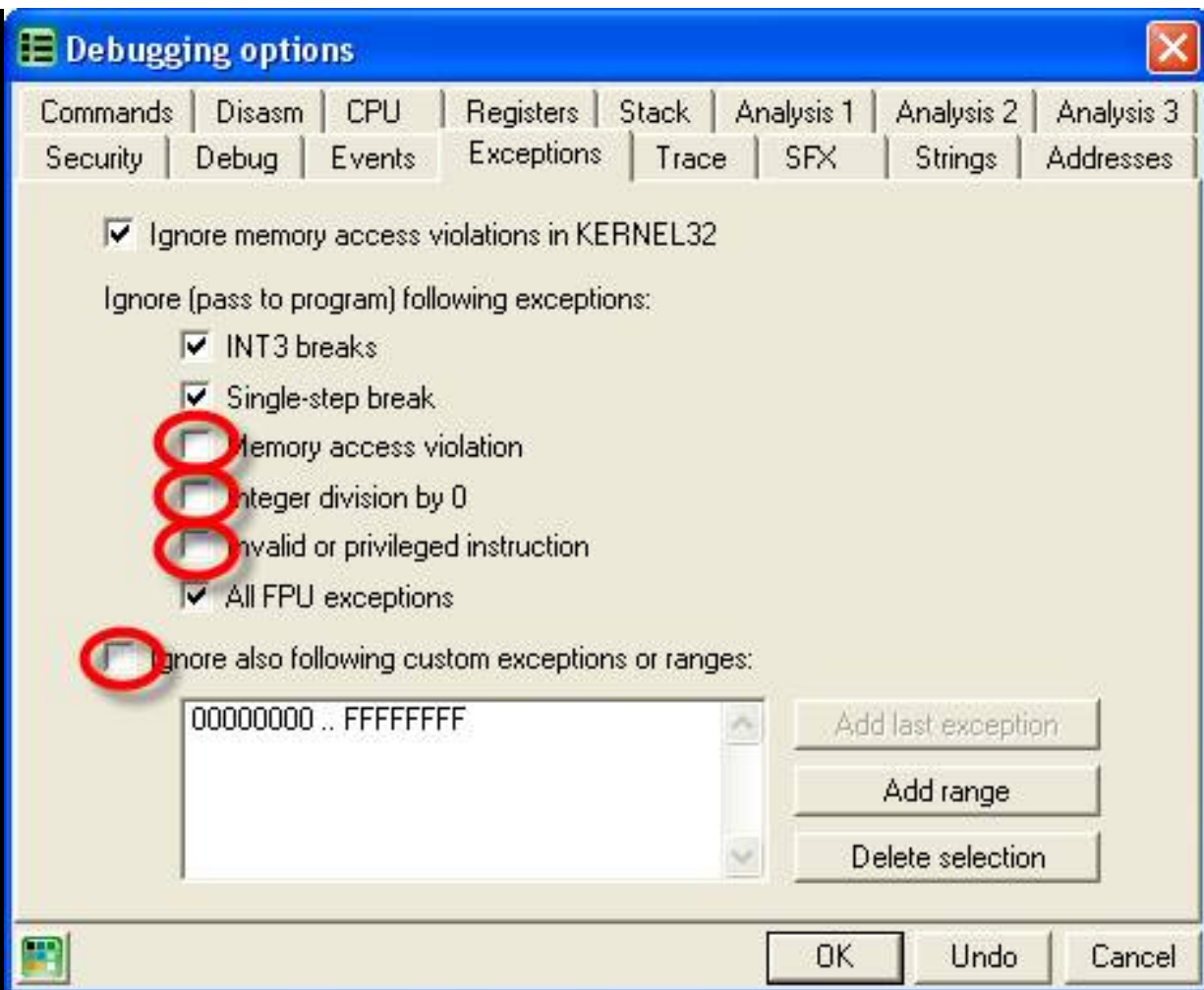
I-Analyze:

Coding II:

I-Analyze:

Phu, now have quite the mood to write more items on SafeDisc J . 2 In the first mentioned to Bypass Anti Debug, find and fix IAT OEP. File fix after the crash.

Now load file dump Olly to find out the cause. 1 way to find errors very quickly is to utilize Option Exception by Olly this many brothers on "professional" reverse crack to do that. Now we uncheck several categories:



Often the bug occurred after the dump file is the program to access a memory region 1 ko valid (indocrination access memory) in the case of 1 part of the code packer rôt have been falling again, or is divided error 0 (integer division by 0) or by the enforcement order ko valid (or invalid privileged instruction) in the code was stolen, not decrypt ...

But why uncheck custom Exception? As the favorite trick is that since **00000000 FFFFFFFF** so **that** it covers the bypass is the exception, to have to uncheck the box to the exception of the other means. When we touched ncheck, if 1 in this exception, Olly will break back immediately, stop in the position near the error occurred ko NAG error of the windows.

Hour 1 and F9, windows command line thui black, now find errors: (dom status bar down by Olly)

Access violation when executing [66844C14] - use Shift+F7/F8/F9 to pass exception to program

So is the program is access to the region in mind ko valid **[66844C14]**, we immediately follow stack to return to the most recent returns:



Here:

0051ADA7	8B56 08	MOV EDX,DWORD PTR DS:[ESI+8]
0051ADAA	8915 04037E00	MOV DWORD PTR DS:[7E0304],EDX
0051ADB0	8B76 0C	MOV ESI,DWORD PTR DS:[ESI+C]
0051ADB3	E8 D1B1EEFF	CALL Raiden3_.00405F89
0051ADB8	CC	INT3
0051ADB9	8935 F8027E00	MOV DWORD PTR DS:[7E02F8],ESI
0051ADBf	83F9 02	CMP ECX,2
0051ADC2	74 0C	JE SHORT Raiden3_.0051ADD0
0051ADC4	81CE 00800000	OR ESI,8000
0051ADCA	8935 F8027E00	MOV DWORD PTR DS:[7E02F8],ESI

Therefore CALL above the call to the arising problems. With restart, enter into this CALL:

00405F88	C3	RET
00405F89	51	PUSH ECX
00405F8A	50	PUSH EAX
00405F8B	E8 63F1FFFF	CALL Raiden3_.00405F83
00405F90	55	PUSH EBP
00405F91	8BEC	MOV EBP,ESP
00405F93	6A 07	PUSH 7
00405F95	E8 16FEFFFF	CALL Raiden3_.00405DB0
00405F9A	83C4 04	ADD ESP,4
00405F9D	68 0000003F	PUSH 3F000000
00405FA2	E8 098D0600	CALL Raiden3_.0046ECB0
00405FA7	83C4 04	ADD ESP,4
00405FAA	E8 618A0600	CALL Raiden3_.0046EA10

According to the week itself, we again enter into CALL almost immediately below first function best:

004050F3	B8 AB200000	MOV EAX,20AB
004050F8	59	POP ECX
004050F9	8D0408	LEA EAX,DWORD PTR DS:[EAX+ECX]
004050FC	8B00	MOV EAX,DWORD PTR DS:[EAX]
004050FE	FFE0	JMP EAX
00405100	55	PUSH EBP
00405101	8BEC	MOV EBP,ESP
00405103	83EC 10	SUB ESP,10
00405106	56	PUSH ESI
00405107	A1 E0416100	MOV EAX,DWORD PTR DS:[6141E0]
0040510C	0FBF88 AA010000	MOVSX ECX,WORD PTR DS:[EAX+1AA]

Look command **JMP EAX** 1. Q suspect, dom through the register at the time of birth defects:


```

Registers (FPU)
EAX 66844C14
ECX 00405F89 Raiden3
EDX 00000001
EBX 7FFDA000
ESP 0012FEA4
EBP 0012FFC0
ESI 00000A28
EDI 00000094
EIP 66844C14
C 0 ES 0023 32bit
P 1 CS 001B 32bit
A 0 SS 0023 32bit
Z 1 DS 0023 32bit
S 0 FS 003B 32bit
T 0 GS 0000 NULL
D 0

```

Morning as the date coincides with the value EAX region memory access ko valid. So we have been surviving this memory area file the pack.

Load the file pack, anti bypass, find OEP, i need to fix Full IAT, is now under OEP dom, CALL see that we follow back from the stack at this time:

Address	Hex dump	Disassembly
0051AD70	6A 60	PUSH 60
0051AD72	68 00005400	PUSH Raiden3.0054D000
0051AD77	E8 CC4B0000	CALL Raiden3.0051F948
0051AD7C	BF 94000000	MOV EDI, 94
0051AD81	8BC7	MOV EAX, EDI
0051AD83	E8 F8F1FFFF	CALL Raiden3.00519F80
0051AD88	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP
0051AD8B	8BF4	MOV ESI, ESP
0051AD8D	893E	MOV DWORD PTR DS:[ESI], EDI
0051AD8F	56	PUSH ESI
0051AD90	FF15 D4C05200	CALL DWORD PTR DS:[52C0D4]
0051AD96	8B4E 10	MOV ECX, DWORD PTR DS:[ESI+10]
0051AD99	890D F4027E00	MOV DWORD PTR DS:[7E02F4], ECX
0051AD9F	8B46 04	MOV EAX, DWORD PTR DS:[ESI+4]
0051ADA2	A3 00037E00	MOV DWORD PTR DS:[7E0300], EAX
0051ADA7	8B56 08	MOV EDX, DWORD PTR DS:[ESI+8]
0051ADAA	8915 04037E00	MOV DWORD PTR DS:[7E0304], EDX
0051ADB0	8B76 0C	MOV ESI, DWORD PTR DS:[ESI+C]
0051ADB3	E8 D1B1EEFF	CALL Raiden3.00405F89
0051ADB8	CC	INT3
0051ADBA	002F 50027E00	MOV DWORD PTR DS:[7E02F0], ESI

Now from OEP, we trace every CALL to this, remember to get the address is **51ADB3**, then go to trace:

Address	Hex dump	Disassembly
00405F89	51	PUSH ECX
00405F8A	50	PUSH EAX
00405F8B	E8 63F1FFFF	CALL Raiden3.004050F3
00405F90	55	PUSH EBP
00405F91	8BEC	MOV EBP, ESP
00405F93	6A 07	PUSH 7

Continue to trace CALL go right now as under:

Address	Hex dump	Disassembly
004050F3	B8 AB200000	MOV EAX,20AB
004050F8	59	POP ECX
004050F9	8D0408	LEA EAX,DWORD PTR DS:[EAX+ECX]
004050FC	8B00	MOV EAX,DWORD PTR DS:[EAX]
004050FE	- FFE0	JMP EAX
00405100	55	PUSH EBP
00405101	8BEC	MOV EBP,ESP

JMP EAX to trace where the dump file fails, then the F7:

Address	Hex dump	Disassembly
66844C14	58	POP EAX
66844C15	59	POP ECX
66844C16	68 00004000	PUSH 400000
66844C1B	54	PUSH ESP
66844C1C	9C	PUSHFD
66844C1D	60	PUSHAD
66844C1E	54	PUSH ESP
66844C1F	68 044C8466	PUSH "df394b.66844C04
66844C24	E8 77BEF2FF	CALL "df394b.66770AA0
66844C29	58	POP EAX
66844C2A	58	POP EAX
66844C2B	61	POPAD
66844C2C	9D	POPFD
66844C2D	5C	POP ESP
66844C2E	C3	RET
66844C2F	0000	ADD BYTE PTR DS:[EAX],0

Is the correct file to pack the area with this memory. It na Ná decrypt the IAT was presented in Vol 2. I CALL trace to the next day:

Address	Hex dump	Disassembly
66770AA0	55	PUSH EBP
66770AA1	8BEC	MOV EBP,ESP
66770AA3	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
66770AA6	50	PUSH EAX
66770AA7	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
66770AAA	E8 51FFFFFF	CALL "df394b.66770AA0
66770AAF	5D	POP EBP
66770AB0	C3	RET
66770AB1	CC	INT3

CALL is also the trace:

Address	Hex dump	Disassembly
66770A00	83EC 30	SUB ESP,30
66770A03	56	PUSH ESI
66770A04	8BF1	MOV ESI,ECX
66770A06	66:837E 7C 00	CMP WORD PTR DS:[ESI+7C],0
66770A0B	74 0A	JE SHORT "df394b.66770A17
66770A0D	8D46 60	LEA EAX,DWORD PTR DS:[ESI+60]
66770A10	50	PUSH EAX
66770A11	FF15 7CB07C66	CALL DWORD PTR DS:[<&KERNEL32.EnterCritical
66770A17	8D4C24 04	LEA ECX,DWORD PTR SS:[ESP+4]
66770A1B	E8 00F0FFFF	CALL "df394b.66770720
66770A20	8B4C24 38	MOV ECX,DWORD PTR SS:[ESP+38]
66770A24	51	PUSH ECX
66770A25	8D4C24 08	LEA ECX,DWORD PTR SS:[ESP+8]
66770A29	E8 82FEFFFF	CALL "df394b.667708B0
66770A2E	66:8B16	MOV DX,WORD PTR DS:[ESI]
66770A31	8D4C24 04	LEA ECX,DWORD PTR SS:[ESP+4]
66770A35	52	PUSH EDX

Now the comfort F8 quite a long, F7 need to do some other things more CALL, CALL to this:

Address	Hex dump	Disassembly
66770A4C	8D4C24 04	LEA ECX,DWORD PTR SS:[ESP+4]
66770A50	E8 2BF0FFFF	CALL "df394b.66770780"
66770A55	66:837E 7C 00	CMP WORD PTR DS:[ESI+7C],0
66770A5A	74 0A	JE SHORT "df394b.66770A66"
66770A5C	83C6 60	ADD ESI,60
66770A5F	56	PUSH ESI
66770A60	FF15 78B07C66	CALL DWORD PTR DS:[&KERNEL32.LeaveCriticalSection]
66770A66	8D4C24 04	LEA ECX,DWORD PTR SS:[ESP+4]
66770A6A	E8 11000000	CALL "df394b.66770A80"
66770A6F	66:B8 0100	MOV AX,1
66770A73	5E	POP ESI
66770A74	83C4 30	ADD ESP,30
66770A77	C2 0400	RETN 4
66770A7A	90	NOP

F7, to see it go right near the code:

Address	Hex dump	Disassembly
66770A66	8D4C24 04	LEA ECX,DWORD PTR SS:[ESP+4]
66770A6A	E8 11000000	CALL "df394b.66770A80"
66770A6F	66:B8 0100	MOV AX,1
66770A73	5E	POP ESI
66770A74	83C4 30	ADD ESP,30
66770A77	C2 0400	RETN 4
66770A7A	90	NOP
66770A7B	90	NOP
66770A7C	90	NOP
66770A7D	90	NOP
66770A7E	90	NOP
66770A7F	90	NOP
66770A80	55	PUSH EBP
66770A81	8BEC	MOV EBP,ESP
66770A83	83EC 08	SUB ESP,8
66770A86	53	PUSH EBX
66770A87	56	PUSH ESI
66770A88	57	PUSH EDI
66770A89	894D F8	MOV DWORD PTR SS:[EBP-8],ECX
66770A8C	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]
66770A8F	83C0 20	ADD EAX,20
66770A92	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX
66770A95	FF65 FC	JMP DWORD PTR SS:[EBP-4]
66770A98	5F	POP EDI
66770A99	5E	POP ESI
66770A9A	5B	POP EBX
66770A9B	8BE5	MOV ESP,EBP
66770A9D	5D	POP EBP
66770A9E	C3	RETN

Trace to JMP below, and F7 it fly:

Address	Hex dump	Disassembly
0012FE58	BC 80FE1200	MOV ESP,12FE80
0012FE5D	61	POPAD
0012FE5E	90	POPFD
0012FE5F	5C	POP ESP
0012FE60	C3	RETN
0012FE61	9C	PUSHFD
0012FE62	4E	DEC ESI

Why it's the same flight on the stack;)). Replace the shelves, there is **RETN** mùng then, any return to:

0051ADA7	8856 08	MOV EDX,DWORD PTR DS:[ESI+8]
0051ADAA	8915 04037E00	MOV DWORD PTR DS:[7E0304],EDX
0051ADB0	8B76 0C	MOV ESI,DWORD PTR DS:[ESI+C]
0051ADB3	81E6 FF7F0000	AND ESI,7FFF
0051ADB9	8935 F8027E00	MOV DWORD PTR DS:[7E02F8],ESI
0051ADBF	83F9 02	CMP ECX,2
0051ADC2	74 0C	JE SHORT Raiden3.0051ADD0
0051ADC4	81CE 00800000	OR ESI,8000
0051ADCA	8935 F8027E00	MOV DWORD PTR DS:[7E02F8],ESI
0051ADD0	C1E0 08	SHL EAX,8
0051ADD3	03C2	ADD EAX,EDX
0051ADD5	A3 FC027E00	MOV DWORD PTR DS:[7E02FC],EAX
0051ADDA	33F6	XOR ESI,ESI

You see it return to where do: **51ADB9**? Initially it to CALL in **51ADB3** position, but now questions CALL command was replaced by 1 several other commands. That is already the decrypt the original code to replace the stolen commands.

What more nhi? According to the logic of any copy paste the code to decrypt the files through the dump is ok. But have never stolen only 1 packer 1 position or do? If that is the OEP can, but only 1 code normally. So there are 2 deducing the following:

1. packer stolen many of the program, but can decrypt the code to implement in the original time of 1 day and then encrypt again (eg molebox, pespun, obsidium 3 trick in mind if i do:- P). In case this often CALL decrypt the call to the code have offset each other. (molebox is the individual fixed)
2. packer stolen many of the program but after i decrypt code will encrypt back. In case this is often the decrypt fixed at 1.

In Vol 2, we see the IAT's SafeDisc decrypt implemented in the same paragraph 1 should also conjecture that the stolen above will have 1 to the call CALL 1 position decrypt code.

But do remember CALL ko? **CALL 405F89**

We should analyze Ctrl-A to the dump file for the search command. Then search all questions on the command CALL:



The tolerable:

Address	Disassembly
0045DAFC	CALL Raiden3_.00405F89
004683B6	CALL Raiden3_.00405F89
0046AB3E	CALL Raiden3_.00405F89
0046CA54	CALL Raiden3_.00405F89
00495603	CALL Raiden3_.00405F89
004C3E6D	CALL Raiden3_.00405F89
00519F5B	CALL Raiden3_.00405F89
0051A725	CALL Raiden3_.00405F89
0051AAB8	CALL Raiden3_.00405F89
0051ADB3	CALL Raiden3_.00405F89

So many of the stolen. Now fix how? If you understand how to fix the part IAT you will easily understand how to fix this case stolen. As the number of CALL is quite small so we can implement in each hand CALL 1 to decrypt all the above, but must also do the general, if the amount of up to 100.1000 ... is the best we must write code or scripts. Here the trick select code, we step through the next item.

II-Coding:

Here, we see if CALL is implemented on the original code will be restored. So we must search code to all the questions on the CALL command and then to turn enforcement. With the script to do this script by simulating the operation with Olly, but today we will code I can call all over CALL J , 1 zi interesting art!

First, remember that the call to CALL in 1 place, so CALL Opcode of this are the same. : **E8 D1 B1 EE FF**

2, and to that kĩ CALL Opcode of this view have been match of 1 Opcode several commands to the other or do? If the same can we call this in CALL 1 blether à arising exception.

3, E8 is the command opcode CALL, 4 bytes behind ko describes it called offset to that is the offset between the number to call and order offset by CALL (in bytes **E8**). It would be useful to know where to call (based on opcode) we may be calculated as follows:

+ Get offset by CALL: **51ADB3**

+ 4 bytes added on (the reverse, that is number 1 int offset 2): **FFEEB1D1**

Result + = **405F84**

+ Community then **5** and the results have been (plus 5 for the length of the command CALL opcode is 5 bytes)

Result + = **405F89** (exactly what to see in Olly)

4, 1 CALL when called on, in the initial stack contains return value, but this value is wrong because the return after decrypt, the return could be offset other 1 (**51ADB8** # **51ADB9**). If the code to call a series of CALL such, we must know exactly where the pressure to return it immediately. Because each CALL return on each different place, while we want to return it to the inline code fixed by us.

Above, while the trace, we have found 1 **LONG JUMP** quite special:

66770A80	55	PUSH EBP
66770A81	8BEC	MOV EBP,ESP
66770A83	83EC 08	SUB ESP,8
66770A86	53	PUSH EBX
66770A87	56	PUSH ESI
66770A88	57	PUSH EDI
66770A89	894D F8	MOV DWORD PTR SS:[EBP-8],ECX
66770A8C	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]
66770A8F	83C0 20	ADD EAX,20
66770A92	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX
66770A95	FF65 FC	JMP DWORD PTR SS:[EBP-4]
66770A98	5F	POP EDI
66770A99	5E	POP ESI
66770A9A	5B	POP EBX
66770A9B	8BE5	MOV ESP,EBP
66770A9D	5D	POP EBP
66770A9E	C3	RET
66770A9F	CC	INT3

LONG JUMP When to this is the original code has been completely restored and the stack contains the return was accurate (if you need the test again nhé, break here and on **51ADB3** Goto preview). Therefore we will force it to return to the inline code at this Jump.

5, so the pressure is on the return of their inline code should probably be Stack overflow. To avoid this, need to save the stack pointer again. If observed kī during CALL to decrypt the value of **EDI** and **EBX** do change so we'll take it as a place to backup pointer. In case ko may also record any safety may be used to partition 1 clipboard (use the temporary memory area corresponding to each target to correct the value must be set and write access to the cove)

Finally CALL are located in this **section. Text** so you must scan from the beginning to the end of this section to implement all CALL can. OK, time is the Code: (code used by **anonymous**, and are optimized for editing more reasonable)

Title stolen_commands

.386

. model small, stdcall

Option casemap: none

. code

_IB Equ 00400000h; ImageBase

_text equ 00003000h; Section. ta.i text start 403000h

_textSize equ 00129000h; Size of section. text

_ICROffset Equ 00405F89h; offset by doan decrypt code CALL call to

start:

pushad

pushfd

_ICR:

mov **eax**, (+ _IB _text); Bat in a scan from a skin `section. text

_ICRLoop:

Cmp byte ptr [**eax**], 0E8h; check opcode of 1 lenh CALL


```
je _ICRCheck; if lenh 1 CALL `test jump _ICRCheck
```

_ICROk:

```
inc eax; neu ko phai opcode lenh CALL tang test eax wool 1
```

```
Cmp eax, (_IB + + _text _textSize); check have to position cuoi
```

```
; Crab section. Text or not?
```

```
je _ICREnd; if Location cuoi, jump _ICREnd
```

```
; Through the end of the `scan
```

```
jmp _ICRLoop; three `If not continue the contest is` m la.i
```

_ICRCheck:

```
mov ebx, dword ptr [eax + 01]; la'y and in 4-byte `ke 'pass opcode lenh CALL
```

```
add ebx, eax; there. he't by the ^`n` o and EBX
```

```
add ebx, 5; EBX tang le ^ 5 bytes
```

```
; ~ Will offset the duo.c said go.i to'i
```

```
Cmp ebx, _ICROffset; Check offset stored in EBX
```

```
; `Have three of the offset doa.n or decrypt ko?
```

```
jne _ICROk; three Ne'u ko `` transfer the test? n toi 'Location next
```

```
; Three Ne'u `` the contest:
```

```
mov edi, eax; luu la.i Location in EAX and EDI `o
```

```
mov ebx, esp; luu la.i stack pointer de tra'nh stack overflow
```

```
jmp eax; home? italy toi 'offset chu'a CALL 00405F89
```

```
; De ba't skin a go.i no '
```

_ICRJumpHere::; Doa.n code italy na `I'll let other LONG JUMP

```
; Jump `ve said na` y.
```

```
mov eax, edi; khoi phu.c la.i truo'c value of the EAX
```

```
mov esp, ebx; khoi phu.c la.i stack pointer
```

```
jmp _ICROk; return doa.n `ti` m ca'c CALL
```

_ICREnd:

```
popfd
```

```
popad
```

```
nop; dat bp ta.i here to Break!
```

```
nop
```

end **start**

OK, build the file with MASM. After the OEP, fix IAT ... just load the file code build complete, copy and paste code to 1 region in the file pack:

Address	Hex dump	Disassembly
007E3000	60	PUSHAD
007E3001	9C	PUSHFD
007E3002	B8 00304000	MOV EAX,Raiden3.00403000
007E3007	8038 E8	CMP BYTE PTR DS:[EAX],0E8
007E300A	74 0A	JE SHORT Raiden3.007E3016
007E300C	40	INC EAX
007E300D	3D 00C05200	CMP EAX,Raiden3.0052C000
007E3012	74 1E	JE SHORT Raiden3.007E3032
007E3014	EB F1	JMP SHORT Raiden3.007E3007
007E3016	8B58 01	MOV EBX,DWORD PTR DS:[EAX+1]
007E3019	0308	ADD EBX,EAX
007E301B	83C3 05	ADD EBX,5
007E301E	81FB 895F4000	CMP EBX,Raiden3.00405F89
007E3024	75 E6	JNZ SHORT Raiden3.007E300C
007E3026	8BF8	MOV EDI,EAX
007E3028	8BDC	MOV EBX,ESP
007E302A	FFEB	JMP EAX
007E302C	8BC7	MOV EAX,EDI
007E302E	8BE3	MOV ESP,EBX
007E3030	EB DA	JMP SHORT Raiden3.007E300C
007E3032	9D	POPFD
007E3033	61	POPAD
007E3034	90	NOP
007E3035	90	NOP

The ICRJumpHere above will offset 7E302C match. So we go back seats LONG JUMP:

66770A8F	83C0 20	ADD EAX,20
66770A92	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX
66770A95	FF65 FC	JMP DWORD PTR SS:[EBP-4]
66770A98	5F	POP EDI
66770A99	5E	POP ESI
66770A9A	5B	POP EBX

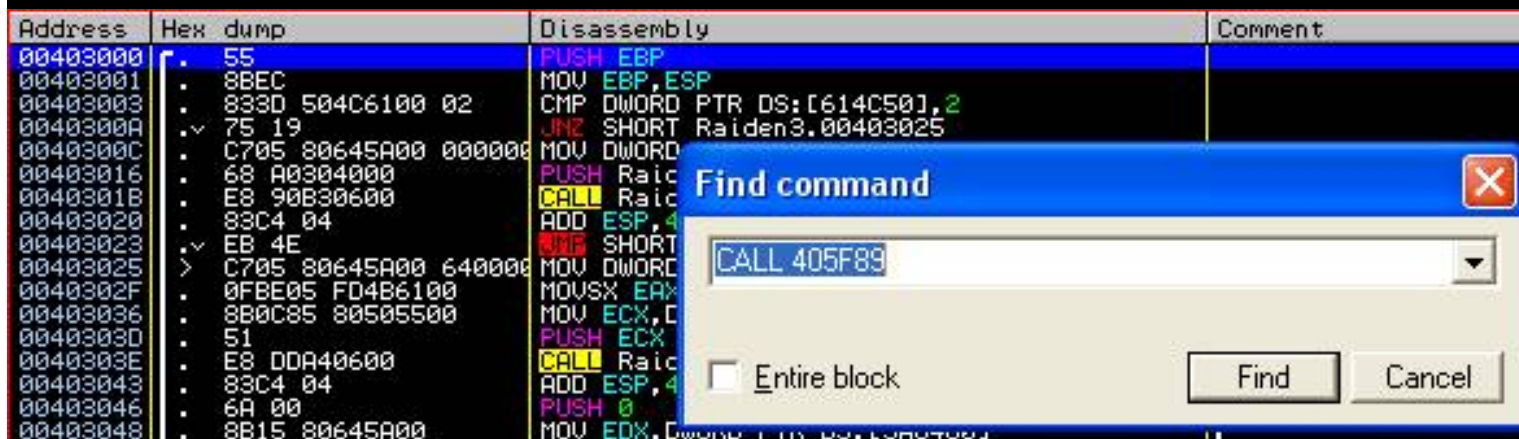
Edit to:

66770A8F	83C0 20	ADD EAX,20
66770A92	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX
66770A95	EB 9225079A	JMP Raiden3.007E302C
66770A9A	5B	POP EBX
66770A9B	8BE5	MOV ESP,EBP

Now New Origin Code at the top of the area just over the paste. BP set me in order to break the NOP.

Address	Hex dump	Disassembly
007E3000	60	PUSHAD
007E3001	9C	PUSHFD
007E3002	B8 00304000	MOV EAX,Raiden3.00403000
007E3007	8038 E8	CMP BYTE PTR DS:[EAX],0E8
007E300A	74 0A	JE SHORT Raiden3.007E3016
007E300C	40	INC EAX
007E300D	3D 00C05200	CMP EAX,Raiden3.0052C000
007E3012	74 1E	JE SHORT Raiden3.007E3032
007E3014	EB F1	JMP SHORT Raiden3.007E3007
007E3016	8B58 01	MOV EBX,DWORD PTR DS:[EAX+1]
007E3019	0308	ADD EBX,EAX
007E301B	83C3 05	ADD EBX,5
007E301E	81FB 895F4000	CMP EBX,Raiden3.00405F89
007E3024	75 E6	JNZ SHORT Raiden3.007E300C
007E3026	8BF8	MOV EDI,EAX
007E3028	8BDC	MOV EBX,ESP
007E302A	FFEB	JMP EAX
007E302C	8BC7	MOV EAX,EDI
007E302E	8BE3	MOV ESP,EBX
007E3030	EB DA	JMP SHORT Raiden3.007E300C
007E3032	9D	POPFD
007E3033	61	POPAD
007E3034	90	NOP
007E3035	90	NOP

F9 to run ... and bup, break. Done! To properly test the entire CALL 405F89 decrypt the code, we Goto to 403,000 - the start of the section. Text. Then find all the commands 405F89 CALL:



All have been decrypt:

Item not found

Necessary than before decrypt the code, can save some of offset CALL this. After decrypt, back to check the code has been restored or not.

So is the complete fix the stolen command. Code by **anonymous** 1 fact quite the banana trick but was optimal, the aged in our firm to do that kī;)). Finish this section, we have prepared are 1 step before going to fix the IAT CALL, to Full IAT in Vol 2 ko meaningless;)). I look back at the following:-P

Big thanks to:

All REA's members: Computer_Angel, Moonbaby, Zombie, hacnho, benina, kienmanowar, rongchaua, Deux, Merc, hoadongnoi, the_lighthouse, TQN, light.phoenix, hytkl, tlandn, hurt_heart, dzungltvn, Zoi, littleboy84, haule_nth, takada, Why not bar, iamidiot, Akira, dump, thienthandien, [kid], ...

Special thanks to:

fly, stephente, Gabri3l, MaDMAN_H3rCuL3s, CondZero, Ricardo Narvaja + NCR, lena151, hagggar, ARTeam, snd, RES, CrackLatinos, all unpack.cn ... Authors who created tools and you.



Trick Xi News - 2007

SoftWrap Loader 6.1.1

Written by: Gabri3l / ARTeam

<http://cracking.accessroot.com>

Loader to create a patch dll Softwrap

Target need to bup:

Professor Franklins' Instant Photo Effects 2 (*looks like version has not changed*)

http://www.swsoftware.com/photo_software_effects.aspx

The Tool to use:

PEID, Ollydbg 1:10, 1:14 dUP

Protection type:

Softwrap 6.1.1

Some other information:

The weighing more than 80 MB so I think some of you may not be it down ..

If you still want to follow, you should download XPSmoker 4.4: <http://www.xp-smoker.com/xpsmoker.html> (*the version on the homepage XPSmoker changed, the need to wait for a tricky up 4.5 up*)

It also used the version Softwrap to pack again. Dll is only to offset the load but the code is like italy chang.

I was trying to write this tut very easy to understand for those who do not download the target. Kì this lesson is to overcome the legal protection as well as how a Loader that may apply to other programs by the pack Softwrap this version.

Best view with Firefox at 1280x1024 resolution

Intro:

All tool to use
can be found at:
(*or some other
place:)))*)

[http://navig8.to/
diablo2oo2](http://navig8.to/diablo2oo2)

[http://home.t-
online.de/home/
Ollydbg/](http://home.t-online.de/home/Ollydbg/)

<http://Peid.has.it>

First, we need to
run a target

time. There are a BOX Softwrap notice is to be started before it can run programs. Click Next. When soft run completely close it again. Now the Softwrap was booted and we conducted "bup" it. Open target in PEiD, you pack it with Softwrap. Version not show clearly, but please click BUY in the NAG, we will see it is version 6.1.1.

Body:

How "bup" Softwrap:

1. Ok, Photon.exe to load Ollydbg (*Remember the first run was a time to boot the nag*)
2. There are the exception that this protection used to detect the debugger. So please check the following:
Ignore Memory Access Violations in Kernel32 INT 3 Single Step Breaks

Divison integer by Zero

And in addition to **custom exceptions:**

C000001D (ILLEGAL instruction)

3. Remember to check small exception? Click RUN go. Wait a bit, you see the NAG Softwrap. *(If time is a first run, a nag boot, I do not say things to nag them here.)*

4. Ok, back Olly and press Pause. Select View-> Call Stack. You will see:

Call stack of main thread
Address Stack Procedure /
arguments Called from Frame
0012F3F0 77D493F5 Includes
ntdll.KiFastSystemCallRet
USER32.77D493F3 0012F424
0012F3F4 77D6EA24
USER32.WaitMessage
USER32.77D6EA1F
0012F424
0012F428 77D5688A
USER32.77D6E895
USER32.77D56885 0012F424
0012F450 77D568CC
USER32.77D567D4
USER32.77D568C7
0012F44C
0012F470 77D5892D USER32.
DialogBoxIndirectParamAorW
USER32.77D58928 0012F46C
0012F49C 00EA355F
softwrap.00F35899
softwrap.00EA3559 0012F498
0012F4B4 00EA3876
softwrap.00EA353A

softwrap.00EA3871 * where *
BUNG NAG
0012F4D4
0012F4D8 00EA40C9
softwrap.00EA37EC
softwrap.00EA40C4
0012F4D4
0012FE24 00E919B8
softwrap.00EA3A69
softwrap.00E919B3 0012FE20
0012FE58 00508B3B Includes
softwrap.00E919B8
Photon.00508B39 0012FE54
0012FF9C 00506F25?
Photon.00508A87
Photon.00506F20

5. Remember that this is what should Stack on TOP will be called out first. Now we find any place called this nag.

6. You can Double-Click for each line. But I always do and then help you.

0012F4B4 00EA3876
softwrap.00EA353A <- This place is called a nag. Dom right through, we feel it is called 000EA3871 address.

7. Click mouse to select the Go-To-> Expression and enter the 000EA3871. Click OK

8. You come:

00EA3871 E8 C4FCFFFF
CALL softwrap.00EA353A

9. Right-click and select Analyze code. Pull up a bit and you'll see that a call is selected from a Select Case. I have more comments to better

understand the task Case. Ah, how I know that it tui nhi do? I used a bit of old but always exactly is allowed to try and false. I have tried each a month. Then restart and try the next case. You can also do so if you like. I just thought I should do is help you quickly.

00EA3843 48 DEC EAX; The start switch (cases 1 .. 4)

00EA3844 74 35 JE SHORT softwrap.00EA387B; Jump to nag boot 00EA3846 48 DEC EAX

00EA3847 74 28 JE SHORT softwrap.00EA3871; Jump to Nag Softwrap

00EA3849 48 DEC EAX

00EA384A 74 1B JE SHORT softwrap.00EA3867; Jump to box in the server error Softwrap

00EA384C 48 DEC EAX

00EA384D 0F85 A4010000 JNZ softwrap.00EA39F7;

Terminate program.

00EA3853 C705 48A6ED00

01> MOV DWORD PTR DS: [EDA648], 1; Case 4's switch

00EA3843 * This is where we need TO PROGRAM FOR THAT will run no longer NAG Trial *

00EA385D E8 35A1FFFF

CALL softwrap.00E9D997

00EA3862 E9 90010000 JMP softwrap.00EA39F7

00EA3867 E8 46FFFFFF

```
CALL softwrap.00EA37B2;  
Case 3 of switch  
00EA386C E9 86010000 JMP  
softwrap.00EA39F7  
00EA3871 E8 C4FCFFFF  
CALL softwrap.00EA353A;  
Case 2 of switch  
00EA3876 E9 7C010000 JMP  
softwrap.00EA39F7  
00EA387B 56 PUSH ESI;  
Case's first switch 00EA3843  
00EA387C 6A 08 PUSH 8
```

10. You always want to call the case 4. Code at this time can not be modified because it was running when unpack. So, let us restart the program and set a breakpoint immediately start the Switch.

11. A small trouble is we can not set a breakpoint normal. The reason: with integrity checks (*called a temporary check sovereignty*). When you set a debugger breakpoint the fact it was written prior to the directives that INT3 opcode (*see Help in Olly*). You should know that can stop the implementation of the command with a break INT3. But the override on any part of the code will cause the program to detect a debugger. So we will use memory breakpoint. However, as it implemented the program for longer but will not override the code section any time.

12. Ok, restart Photon.exe in

Olly. Now we want to set a breakpoint in 00EA3843 memory. Right-click, choose Go-To-> Expression and type 00EA3843. Humm, an error message because you do not find the address. Cause of Softwrap dll is not to load.

13. Open debugging and select Options tab (the tab) events. Check **On to Break New Module**.

14. Click the Run window, and the module is implemented will unwind. You can find the dll Softwrap are highlighted in red. Double Click it. You will go into this dll.

15. Right-click to select Go-To-> Expression and type 00EA3843. Click OK, you will come:

00EA3843 11DB ADC EBX,
EBX

16. However dll is not unpack. We are still here want to Break. Right-click and select Breakpoint-> Memory, On Access. This option will help Olly break all the time at which the access to memory areas set breakpoint.

17. Now and then run it. But back to Options and debugging UNcheck **Break New Module** on the skin. Then click Run.

18. You will stop here:

00F3499B 2B6E 14 SUB EBP,

DWORD PTR DS: [ESI +14]

19. Click Run again. Will break as many times in the dll to unpack, because memory regions are retrieved many times.

20. Continue to click Run until you break in 00EA3843, then ordered me to have EAX DEC.

21. Ok, we're in the Select Case. If you dom through EAX will see it is 2. Dom Select the case, code we saw in Case 2 is to jump by Nag Trial Softwrap. So the Case 2 command jumped from JE to JE softwrap.00EA3871

SHORT SHORT

softwrap.00EA3853 it to jump to case 4: MOV DWORD PTR DS: [EDA648], 1.

22. To change, select JE SHORT softwrap.00EA3871 and press SPACEBAR, edit the JE SHORT

softwrap.00EA3853. Done not do? Very good. Click RUN chà Ai! It runs without the nag again. (This is effective whether the time limit has expired trial)

Create Loader:

1. Now we know "bup" was replaced by JE to JE softwrap.00EA3871 SHORT SHORT softwrap.00EA3853 in dll's Softwrap.

2. However, we can not do this when the soft running integrity checks to detect any edit the

time. Therefore we need to make a loader wait until the inspection is completed and then the new patch code. dUP will help us (Diablo2oo2's Universal Patcher)

3. Open up and dUP Offset Patch card. [...] Click on the box to the original file. Select Photon.exe

4. Check on Virtual Address Mode (for packed PE files) because the file you want to patch the pack again. You can not patch it until it is fully load into memory.

5. Time to change the softwrap.00EA3871 JE JE SHORT SHORT softwrap.00EA3853. I was offset by 00EA3847. Change the code this as follows:

```
00EA3847 74 28 JE SHORT
softwrap.00EA3871
** BECOME **
```

```
00EA3847 74 0A JE SHORT
softwrap.00EA3853
```

6. 00EA3847 the same address and 74 bytes does not change. Only in 00EA3848 bytes from 28 to **0A**. *(Next byte is located at the next 00EA3847 should increase 00EA3848)*

7. So in the box Add Bytes 00EA3848 to type in the box and type in the Offset box Original Byte is 28. Patched the box Byte is 0A. ** Note that the Offset on you may be

different **

8. Now click ADD.

9. If you look at the bottom then we will see an option

"Wait for Memory Value Before patch (for loaders).

You can click the small question mark next to that party. They tell us that find a place in memory DWORD value that is written to AFTER integrity checks completed. So we need loader's waiting for DWORD value that is written on and then the new patch dll.

10. We need to find the place DWORD so.

11. Before continuing, we need a few minutes to analyze the problem. You know where the dll check if the program has been registered (*Select Case*), integrity checks will have to occur elsewhere before then. There is a small time after the completion of integrity checks to the check register. We should certainly find a place DWORD code has been brought into memory before the check register arises.

12. How to find a place that before the check register is implemented? Also, we did then đây! When you run the program and the first stop in NAG Trial, open the Call Stack window.

Accommodation is a list of

commands call occurred before
the NAG was created. So we
look back to Call Stack:

Call stack of main thread
Address Stack Procedure /
arguments Called from Frame
0012F3F0 77D493F5 Includes
ntdll.KiFastSystemCallRet
USER32.77D493F3 0012F424
0012F3F4 77D6EA24
USER32.WaitMessage
USER32.77D6EA1F
0012F424
0012F428 77D5688A
USER32.77D6E895
USER32.77D56885 0012F424
0012F450 77D568CC
USER32.77D567D4
USER32.77D568C7
0012F44C
0012F470 77D5892D USER32.
DialogBoxIndirectParamAorW
USER32.77D58928 0012F46C
0012F49C 00EA355F
softwrap.00F35899 but
softwrap.00EA3559 ** THIS
IS AFTER NAG HAVE
BEEN CHECK We NEED
NOT FIND DWORD HERE
**
0012F498
0012F4B4 00EA3876
softwrap.00EA353A
softwrap.00EA3871 ** This is
where BUNG NAG **
0012F4D4
So ** One of the lower jaw is
seats Price EFFECTION
DWORD CHEP TO BE **

0012F4D8 00EA40C9
softwrap.00EA37EC
softwrap.00EA40C4
0012F4D4
0012FE24 00E919B8
softwrap.00EA3A69
softwrap.00E919B3 0012FE20
0012FE58 00508B3B Includes
softwrap.00E919B8
Photon.00508B39 0012FE54
0012FF9C 00506F25?
Photon.00508A87
Photon.00506F20

13. Make sure you're thinking.
"We will find what you're
talking about a DWORD value
is copied into memory?" Let I
explained:

In assembly you have a
command to copy the program
memory (I will say more about
this in the next tut: a patch
program pack with UPX)
That command MOV. MOV
command will do what is the
name of it; It TRANSFER
value in the right address on
the left side. **For example:**
MOV EAX, ECX <-This is the
value of EAX to ECX.

14. When burned into
memory, we use MOV
attached type of data that we
need to move on. **For**
example: MOV BYTE PTR
DS: [401000], FF <FF-
TRANSFER BYTE on
memory at 401,000. From DS
before 401,000 Segment Data
means. Each program in the

memory to run Windows Data Segment have its own. Usually it is accompanied by the first address to machines that need to write information to 401,000 addresses in Data Segment of this program.

15. Ok, time, we know the program can write to a memory DWORD how.

Therefore type questions need to find a command MOV DWORD PTR DS: [xxxxxxx], XXX

16. Any Load Photon.exe to Olly. Remember to check the previous exception (without **Break on New Module**)

17. Click Run until the NAG unwind.

18. When does the NAG, Olly back and hit pause button, then select View-> Call Stack

19. You can find command MOV DWORD PTR DS in a call this. But if you do go to work and trace all the code that you see almost the whole mov in the value 0. And when the soft run to the 0, dUP will find the right location and patch them in line. (*Because the patch is found only when the selected DWORD*)

And then the program detects the patch and the loader failed. (*By Patch integrity checks before completion*)

So-I to help you find small -

20. Double-click on this line in

the Call Stack:

0012FE58 00508B3B Includes
softwrap.00E919B8
Photon.00508B39

*** There is need to correct for
this? In fact no one forced you
all, in case you find the
DWORD value to use is too
great. Many work for
reversing allowed to try and
false. And this is the first
choice but I like that in many
other authorized test .***

21. After the Double Click on
the line, you will go to right
here (*of gold*)

```
00E9192D> / $ 55
PUSH EBP
00E9192E | . 8BEC MOV
EBP, ESP
00E91930 | . 83EC 10
SUB ESP, 10
00E91933 | . 8B45 10
MOV EAX, DWORD PTR
SS: [EBP +10]
00E91936 | . FF05
887DED00 INC DWORD
PTR DS: [ED7D88]
00E9193C | . 833D
887DED00> Cmp DWORD
PTR DS: [ED7D88], 2
00E91943 | . 53 PUSH
EBX
00E91944 | . 56 PUSH
ESI
00E91945 | . 57 PUSH
EDI
00E91946 | . A3
```



```
20A7ED00 MOV DWORD
PTR DS: [EDA720],
EAX
00E9194B |. 7C 56 JL
SHORT
softwrap.00E919A3
00E9194D |. E8
CF670200 CALL
softwrap.00EB8121
00E91952 |. 8BF0 MOV
ESI, EAX
00E91954 |. C1E6 10
SHL ESI, 10
00E91957 |. E8
C5670200 CALL
softwrap.00EB8121
00E9195C |. 6A 10
PUSH 10
00E9195E |. 03F0 ADD
ESI, EAX
00E91960 |. 33DB XOR
EBX, EBX
00E91962 |. 8D45 F0
LEA EAX, DWORD PTR
SS: [EBP-10]
00E91965 |. 53 PUSH
EBX
00E91966 |. 50 PUSH
EAX
00E91967 |. 33FF XOR
EDI, EDI
00E91969 |. E8
025C0200 CALL
softwrap.00EB7570
00E9196E |. 8D45 F0
LEA EAX, DWORD PTR
SS: [EBP-10]
00E91971 |. 50 PUSH
```

```
EAX
00E91972 |. FF75 08
PUSH DWORD PTR SS:
[EBP + 8]
00E91975 |. 56 PUSH
ESI
00E91976 |. E8
62FDFFFF CALL
softwrap.00E916DD
00E9197B |. 83C4 18
ADD ESP, 18
00E9197E |. 83F8 FF
Cmp EAX, -1
00E91981 |. 74 1C JE
SHORT
softwrap.00E9199F
00E91983 |. 6A 01
PUSH 1
00E91985 |. 8D45 F0
LEA EAX, DWORD PTR
SS: [EBP-10]
00E91988 |. 50 PUSH
EAX
00E91989 |. FF75 0C
PUSH DWORD PTR SS:
[EBP + C]
00E9198C |. E8
0DFFFFFF CALL
softwrap.00E9189E
00E91991 |. 83C4 0C
ADD ESP, 0C
00E91994 |. 83F8 FF
Cmp EAX, -1
00E91997 |. 75 04
JNZ SHORT
softwrap.00E9199D
00E91999 |. 0BF8 OR
EDI, EAX
```

```
00E9199B | . EB 02
JMP SHORT
softwrap.00E9199F
00E9199D | > 8BFE MOV
EDI, ESI
00E9199F | > 8BC7 MOV
EAX, EDI
00E919A1 | . EB 67
JMP SHORT
softwrap.00E91A0A
00E919A3 | > 6A 01
PUSH 1
00E919A5 | . 68
4944EC00 PUSH
softwrap.00EC4449
00E919AA | . 33DB XOR
EBX, EBX
00E919AC | . 53 PUSH
EBX
00E919AD | . FF35
8C7DED00 PUSH DWORD
PTR DS: [ED7D8C];
softwrap.00E90000
00E919B3 | . E8
B1200100 CALL
softwrap.00EA3A69
*****
00E919B8 | . E8
64670200 CALL
softwrap.00EB8121
***** ***** YOU
HERE
*****
00E919BD | . 8BF0 MOV
ESI, EAX
00E919BF | . C1E6 10
SHL ESI, 10
00E919C2 | . E8
```

```
5A670200 CALL
softwrap.00EB8121
00E919C7 |. 6A 10
PUSH 10
00E919C9 |. 03F0 ADD
ESI, EAX
00E919CB |. E8
055C0200 CALL
softwrap.00EB75D5
00E919D0 |. 6A 10
PUSH 10
00E919D2 |. 8BF8 MOV
EDI, EAX
00E919D4 |. 53 PUSH
EBX
00E919D5 |. 57 PUSH
EDI
00E919D6 |. E8
955B0200 CALL
softwrap.00EB7570
00E919DB |. 57 PUSH
EDI
00E919DC |. FF75 08
PUSH DWORD PTR SS:
[EBP +8]
00E919DF |. 56 PUSH
ESI
00E919E0 |. E8
F8FCFFFF CALL
softwrap.00E916DD
00E919E5 |. 83C4 1C
ADD ESP, 1C
00E919E8 |. 83F8 FF
Cmp EAX, -1
00E919EB |. 74 14 JE
SHORT
softwrap.00E91A01
```



```
00E919ED | . 57 PUSH
EDI
00E919EE | . FF75 0C
PUSH DWORD PTR SS:
[EBP + C]
00E919F1 | . E8
16FEFFFF CALL
softwrap.00E9180C
00E919F6 | . 83CB FF
OR EBX, FFFFFFFF
00E919F9 | . 3BC3 Cmp
EAX, EBX
00E919FB | . 59 POP
ECX
00E919FC | . 59 POP
ECX
00E919FD | . 74 02 JE
SHORT
softwrap.00E91A01
00E919FF | . 8BDE MOV
EBX, ESI
00E91A01 | > 57 PUSH
EDI
00E91A02 | . E8
C95B0200 CALL
softwrap.00EB75D0
00E91A07 | . 59 POP
ECX
00E91A08 | . 8BC3 MOV
EAX, EBX
00E91A0A | > 5F POP
EDI
00E91A0B | . 5e POP
ESI
00E91A0C | . 5B POP
EBX
00E91A0D | . C9 LEAVE
```

00E91A0E \. C3 RETN

22. Right-Click and select the Olly Analysis-> Analyze Code. Then pull up to see the same code as above.

21. "So what do here?" you will wonder. "command only MOV DWORD PTR DS that I see here is copied to EAX."

This is true, but we should not use this month. Instead we will find it on the command in a tí.

22. Look at this line

00E91936:

00E91936 |. FF05 887DED00
INC DWORD PTR DS:
[ED7D88]

00E9193C |. 833D
887DED00> Cmp DWORD
PTR DS: [ED7D88], 2

23. Orders Increase INC. a value is to 1 times. Should INC DWORD PTR DS: [ED7D88] will plus 1 DWORD value to store in the area code

ED7D88. Then compare the value there with 2. You may think the value in ED7D88 beginning 00000000. Then to me by the command 00000001 inc. *(The click here see the current value of 1 should be a previously 0)*

24. Now back dUP, MemoryAddress in the box and type in 00ED7D88 MemoryValue for the 00000001

*** Copy many of 0? Because it*

*is a DWORD. I will say more in this tutorial # 5. Every pair of 00 'that you see as a BYTE, BYTE two to create a WORD and 4 BYTE create a DWORD. So should all now No. 8 ***

25. All is not any? Good.
Click "Create Loader." File name for Loader and click OK.
26. Now double-click on your new loader to do ... (hope that everything will suon) Oh .. it was delicious and healthy đấy!
No more NAG.

27. Change a computer and try to run the loader. Still healthy appetite! Great, you've made a loader to overcome the protection of Softwrap. Because the loader is made based on the platform is the function and memory dll softwrap so this can be applied to other programs are protected by this version dll.

Conclusion:

I use this program is the only way to present Patch Softwrap. But if you like it then xài please help buy bags.

Thanks to the whole
ARTeam:

[Nilrem] [JDog45]
[Shub - Nigurrath]
[MaDMAn_H3rCuL3s]
[Ferrari] [Kruger]
[Teerayoot] [R @ dier]
[ThunderPwr] [Eggi]

[EJ12N] [Stickman
373] [Bone Enterprise]
[KaGra]

Thanks to all the
people who take time
to write tutorials.

Thanks to all the
people who continue to
develop better tools.

Thanks to his excellent
Diablo2oo2 for patcher

Thanks to Exetools
Woodmann and for
being a great place of
learning.

Thanks also to the
Codebreakers Journal,
and the Anticrack
forum.

If you have questions,
comments edit what the
mail bags for:

Gabri3l2003 [at] yahoo.
com

Translated to English
by Trickyboy (REA
Team).

Special thanks to
Gabri3l
and all people who read
this tut

Stupid Execryptor - Fixing dump

I. Introduction:

II. Config:

III. Unpack:

IV. Fix dump:

1. Analyze:

2. Second dump:

3. Compare and Fix:

V. Find Place Of Real OEP:

VII. Ending:

I. Introduction:

Hi everyone, today man trick for permission to discuss issues of 1 quite intense in unpack Execryptor that dump file to fix it can run on any computer. It is interesting when playing with this packer, trick dot many things, not xài scripts available, or plugin Hide Olly is dump the whole, are generally not have strong faith to unpack always successful. This article only expect people to introduce a further aspect of that execryptor only. Any, to the small ...

II. Config Olly:

Perhaps no configuring is absolute, but even less effective for the target that we work this kì ([MultiTrans](#)), the trick will take a few ways for people to MUP successful.

First, invite pa k0n down here or the Shadow Olly Olly on Ice: (Shadow Olly has Gui English)

<http://www.tuts4you.com/blogs/download.php?list.4>

Then again here, down Olly Advanced plugin 1:25:

<http://www.tuts4you.com/blogs/download.php?view.75>

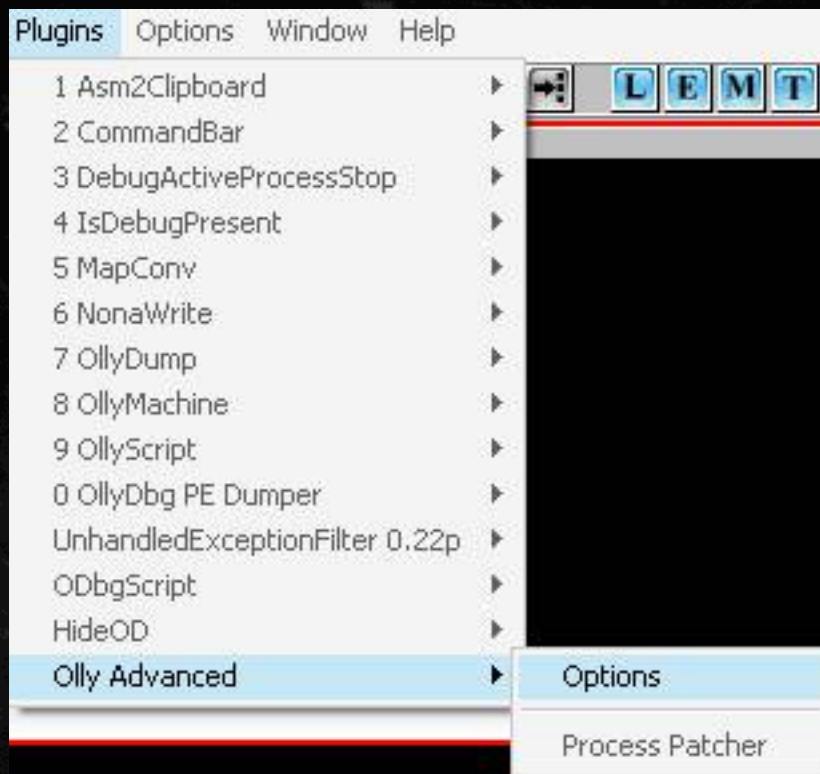
Back here down HideOD plugin:

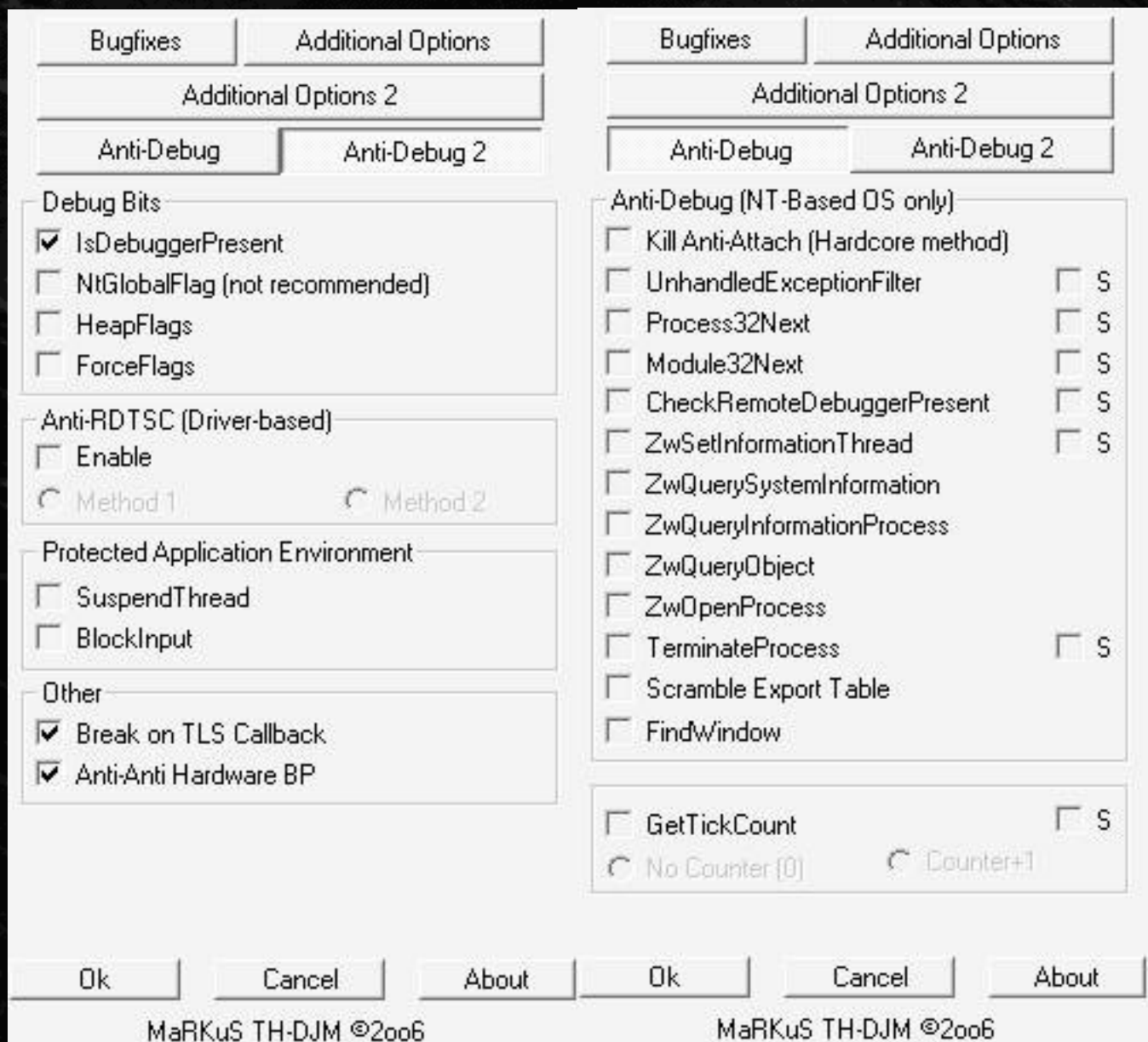
<http://www.tuts4you.com/blogs/download.php?view.58>

Continue down here zo ODbgScript (english and to his com)

<http://www.reaonline.net/forum/showpost.php?p=25199&postcount=27>

That is enough goods. Now we unpacked Olly the Shadow. Then unpacking to the other plugin, copy the folder PLUGIN Olly's Shadow. Call Olly Shadow run time. GUI's trick is that black Revising thui, if the white pa k0n do is copy. Olly Advanced plugin open up front, as image:

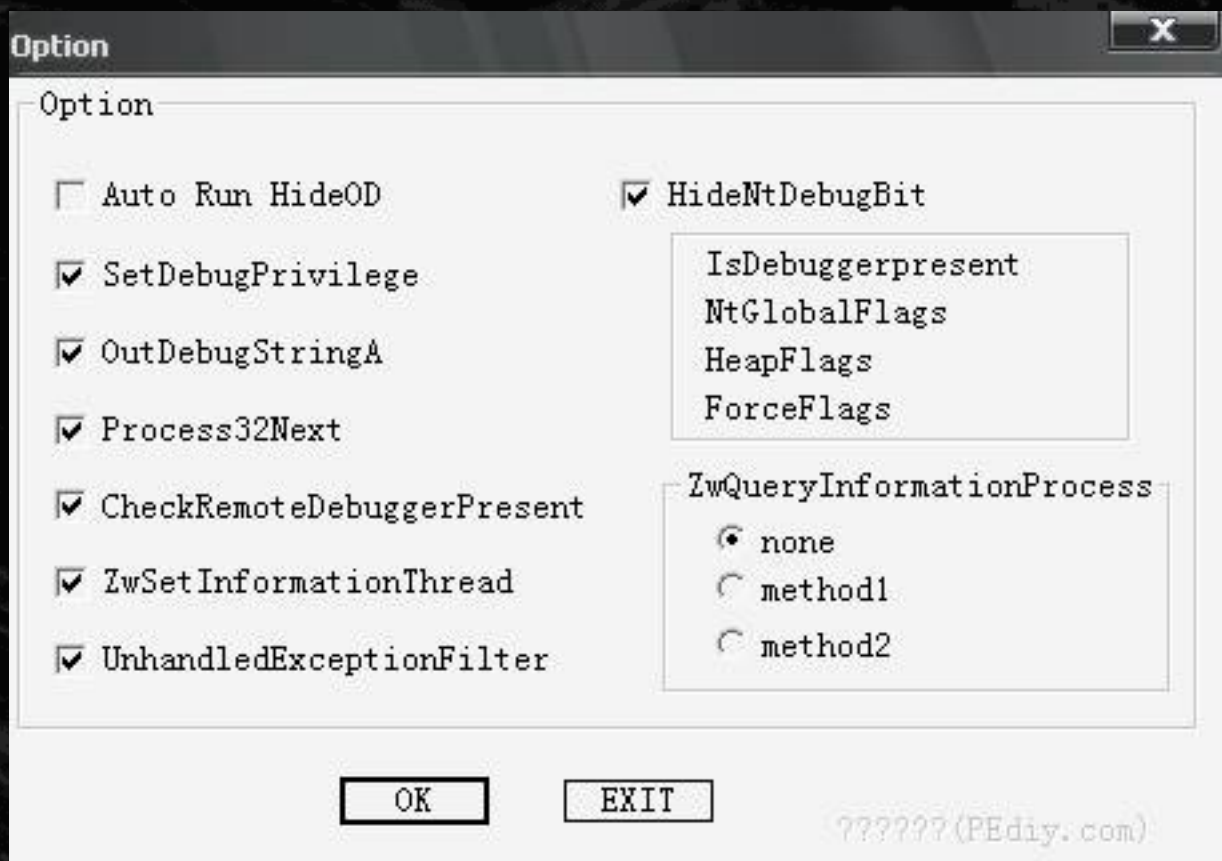




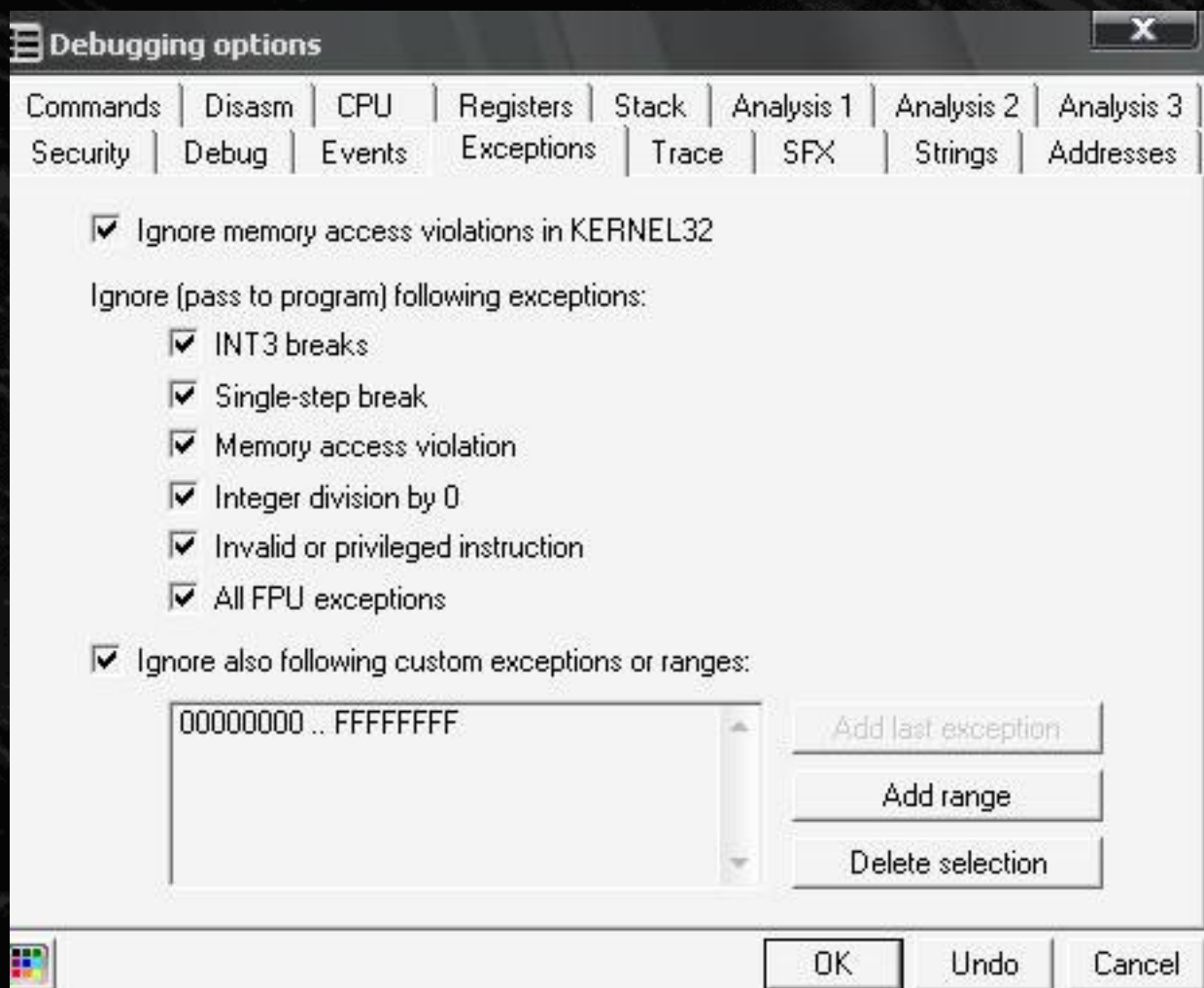
Reason entry Anti-Debug we do check that will detect Execryptor as a method Hide Olly that used to Hook.

Continue to check the plugin HideOD so easy to remember:





Some items do not need to check, but to remember ko pa kOn flipped the trick as check on the hen.
But this is in Olly Option:



Complete the configuration.

III.Unpack:

Now load MultiTrans zo (from scan anything because this topic has to say what it is packer). Olly Advanced will help you break in TLS (code be implemented prior to the EP):

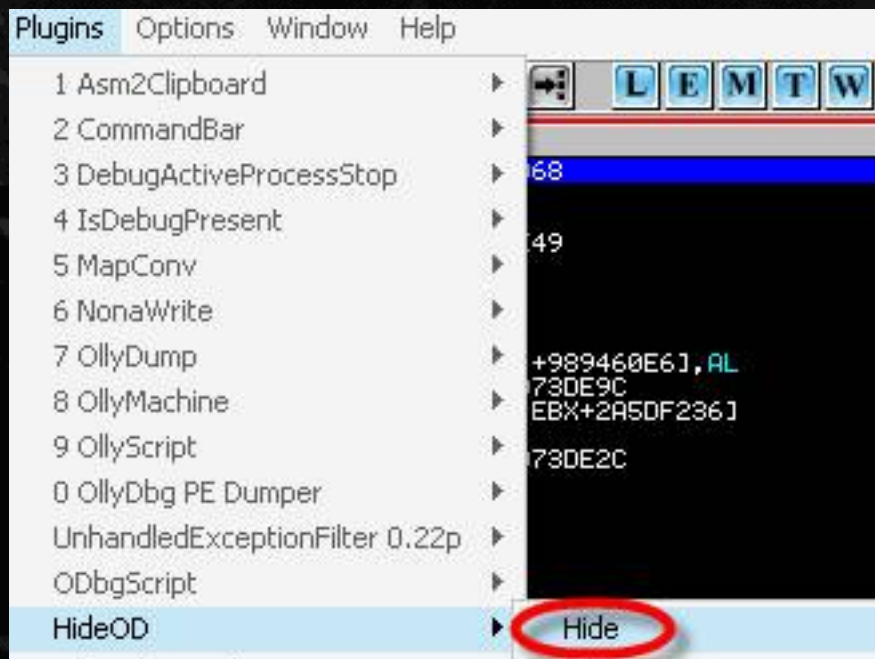
Address	Hex dump	Disassembly
0073DE34	E8 2FFFFFFF	CALL MultiTra.0073DD68
0073DE39	05 40280000	ADD EAX,2840
0073DE3E	^ FFE0	INC EAX
0073DE40	E8 04000000	CALL MultiTra.0073DE49
0073DE45	FFFF	???
0073DE47	CCCC	???

Alt-B, which BP Olly remove land in EP:

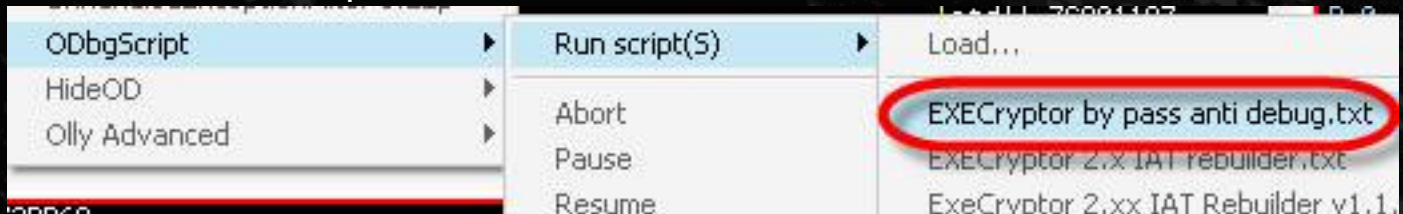
Address	Module	Active	Disassembly
0073DE28	MultiTra	One-shot	CALL MultiTra.0073DD68

Remove
Follow in D

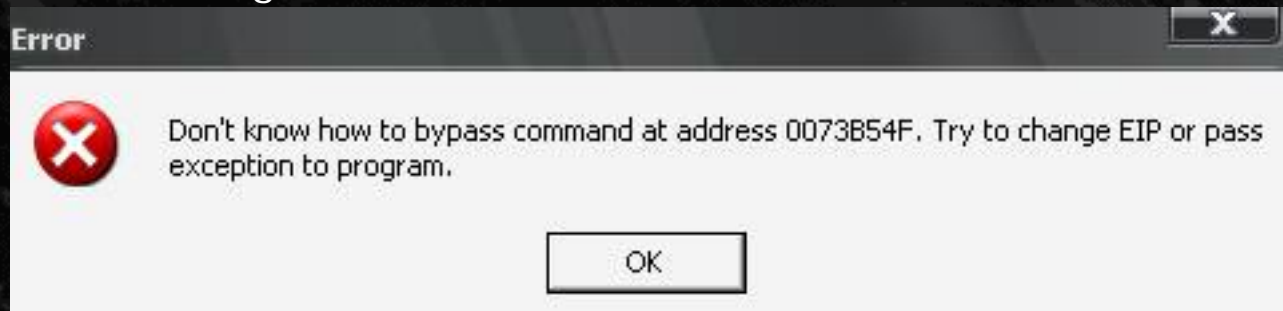
Start the HideOD:



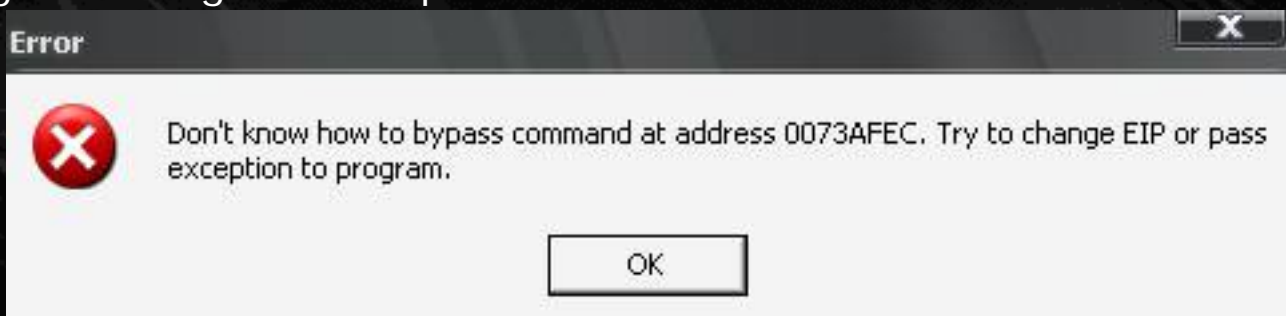
Then run script "Execryptor pass by anti debug.txt" attached this article (a similar script that uses Why in Zip Repair Tool 3.2), is running in memory plugin ODbgScript but must do OllyScript 0.92 nhé yore, because OllyScript do not run in this script:



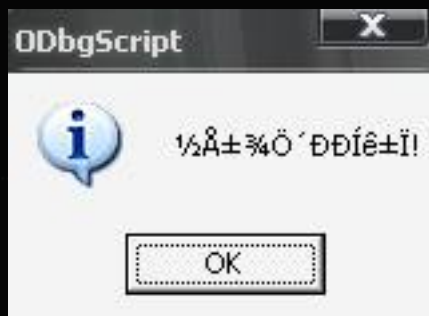
Meet 1 error message first:



Do not worry, OK, then we continue to press Shift-F9, it will run through training at 1 Dong and exception:



OK again and Shift-F9 scripts continue to work and inform Finished (in Chinese, the Com trans end nà ko):



One break in Encrypt OEP:

Address	Hex dump	Disassembly
005FA8E5	56	PUSH ESI
005FA8E6	891424	MOV DWORD PTR SS:[ESP],EDX
005FA8E9	68 2676829A	PUSH 9A827626
005FA8EE	5A	POP EDX
005FA8EF	81E2 4EC26984	AND EDX,8469C24E
005FA8F5	81C2 77255E80	ADD EDX,805E2577
005FA8FB	50 77030000	CALL MultiTask 004A7D77

Note: Script on the most effective on WinXP SP2, if you are running Terminate then Pause before ODbgScript Restart again.

Goto to 401,000 for IAT (how to find the familiar nhé oi), and:

- IAT Start - 566208
- IAT End - 566B6C

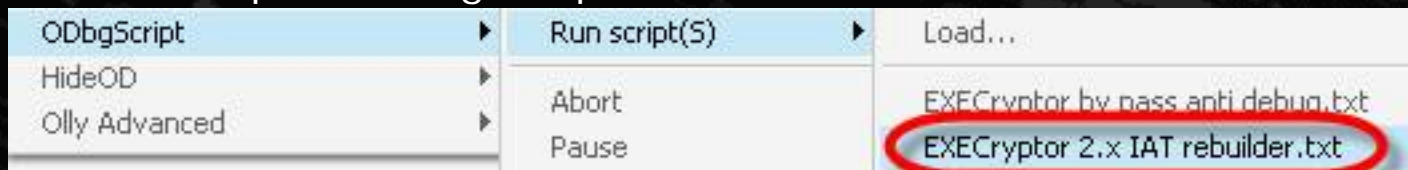
Load 2 parameters to script "EXECryptor 2.x IAT rebuilder.txt" attached in:

```

EXECryptor 2.x IAT rebuilder.tx
31  var c_gpa
32  var temp
33  var gmh
34  var ll
35  var ibase
36  var iend
37
38  //=====
39  // èiè?èà?è?èeó?ì IAT
40  //=====
41  mov iat_start,00566208
42  mov iat_end,00566B6C
43
44  //=====
45  // ???ó÷à?ì èi??eià?èt ? ?e??????
46  //=====

```

Save File Script, Running Script sit and wait:



This script will redirect IAT accurate enough if you sit patiently waiting for it.

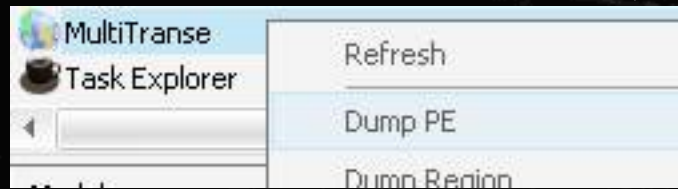
Results:

```

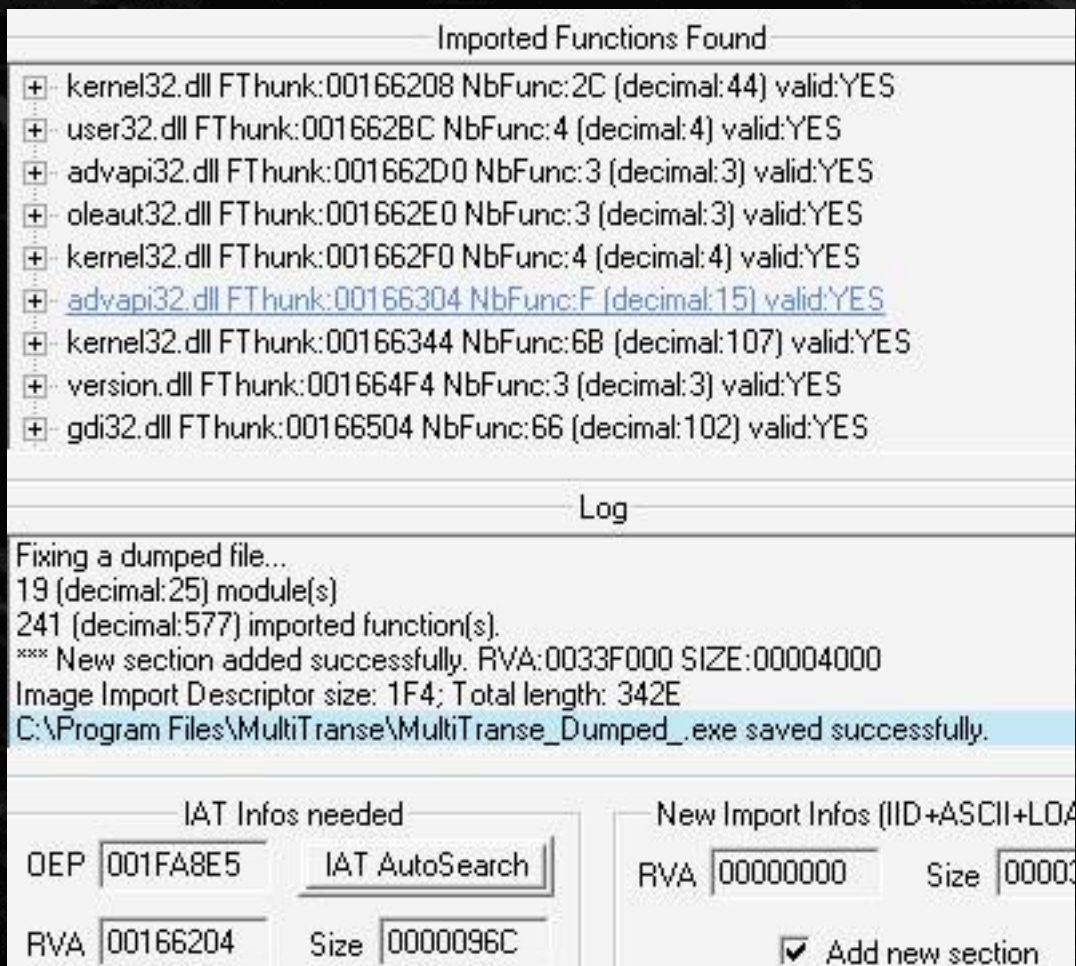
00566207 7C901000 ntdll.RtlDeleteCriticalSection
00566208 7C901001 ntdll.RtlLeaveCriticalSection
00566209 7C901002 ntdll.RtlEnterCriticalSection
00566210 7C809EF1 kernel32.InitializeCriticalSection
00566211 7C809AE4 kernel32.VirtualFree
00566212 7C809A51 kernel32.VirtualAlloc
00566213 7C80992F kernel32.LocalFree
00566214 7C809980 kernel32.LocalAlloc
00566215 7C809728 kernel32.GetCurrentThreadId
00566216 7C80977A kernel32.InterlockedDecrement
00566217 7C809766 kernel32.InterlockedIncrement
00566218 7C80B9D1 kernel32.VirtualQuery
00566219 7C80A0D4 kernel32.WideCharToMultiByte
00566220 7C809BF8 kernel32.MultiByteToWideChar
00566221 7C80B0B6 kernel32.lstrlenA
00566222 7C810111 kernel32.lstrcpyA
00566223 7C80104F kernel32.LoadLibraryExA
00566224 7C80A415 kernel32.GetThreadLocale
00566225 7C801EEE kernel32.GetStartupInfoA
00566226 7C80ADA0 kernel32.GetProcAddress
00566227 7C80B6A1 kernel32.GetModuleHandleA
00566228 7C80B4CF kernel32.GetModuleFileNameA
00566229 7C80D262 kernel32.GetLocaleInfoA
00566230 7C910331 ntdll.RtlGetLastWin32Error
00566231 7C812F10 kernel32.GetCommandLineA
00566232 7C80ABDE kernel32.FreeLibrary
00566233 7C8137D9 kernel32.FindFirstFileA
00566234 7C80EDD7 kernel32.FindClose
00566235 7C81CDDA kernel32.ExitProcess
00566236 7C80C058 kernel32.ExitThread
00566237 7C810637 kernel32.CreateThread
00566238 7C810D87 kernel32.WriteFile
00566239 7C862E62 kernel32.UnhandledExceptionFilter

```

Dump only (do not use because the LordPE detect):



Save the file. Fix IAT with ImportREC:

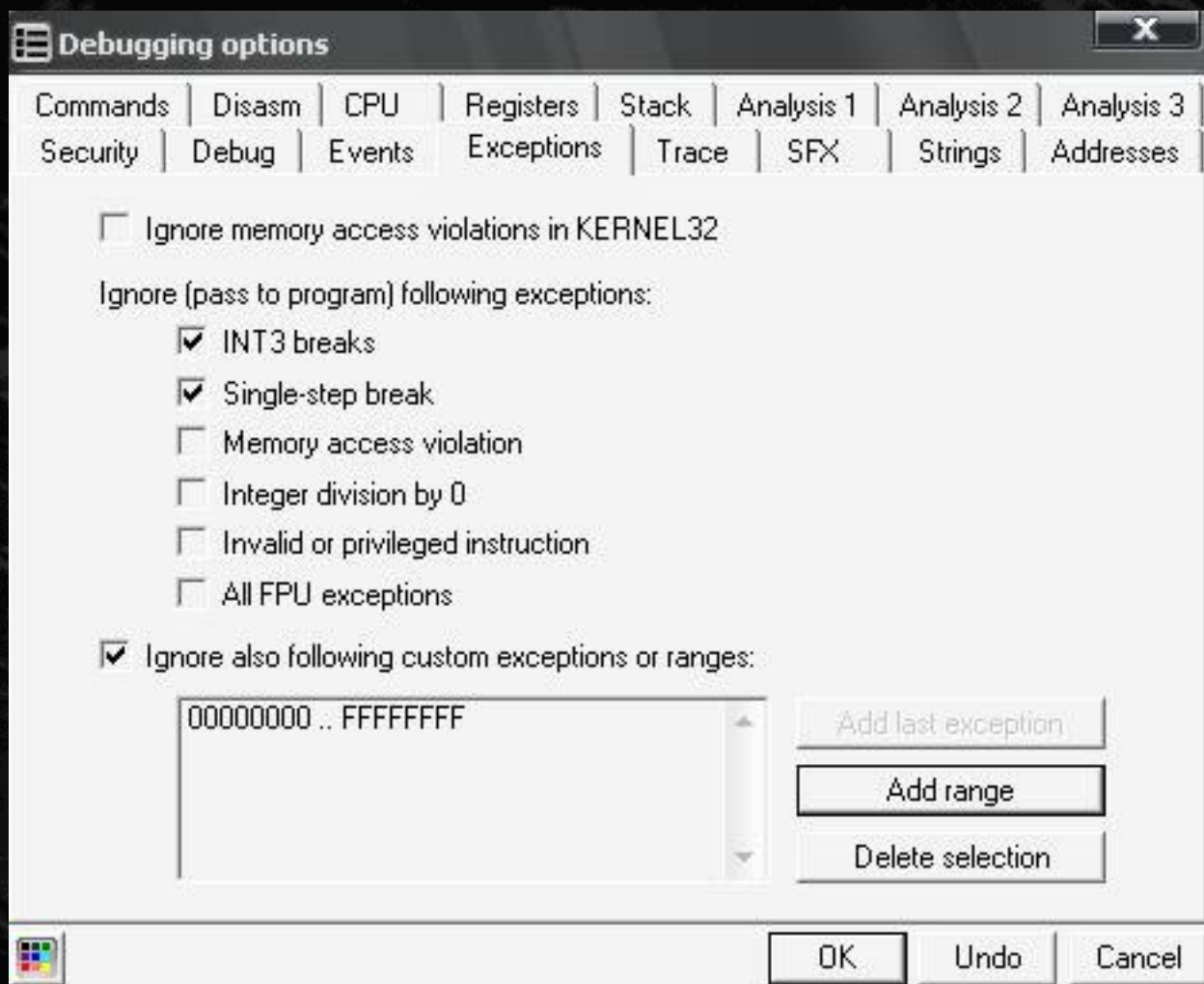


Remember to Save the Tree because we had to wait very long run scripts that. After this will xài to file đấý tree. Run the file dump, of course lickerish run. Finished as the dump. (This step unpack brief for why he and other children made a lot of items and Execryptor).

IV. Fix dump:

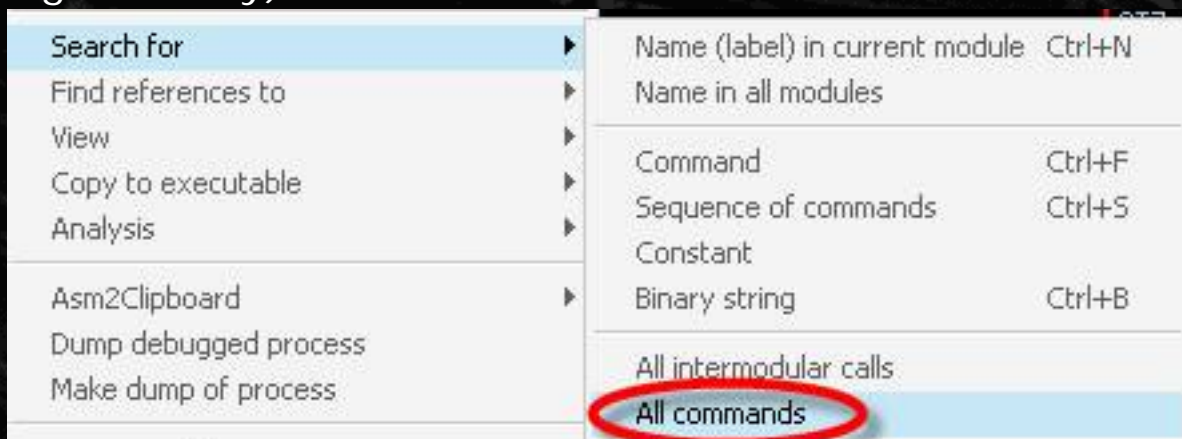
1. Analyze:

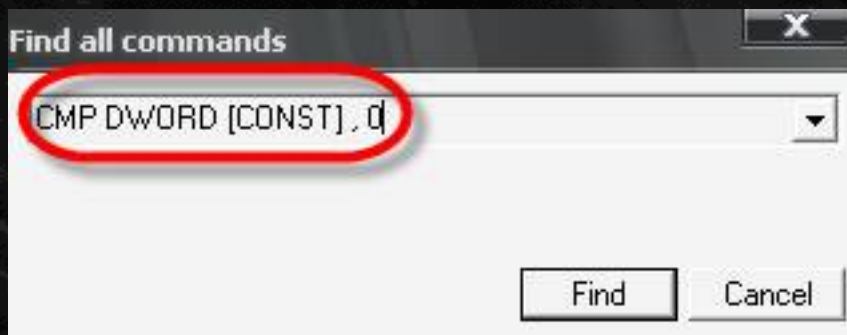
Now try to take over the dump file 1 machine. Trick to bring in Vmware Win SP1 and will be run when the crash. You load the file to fix dump Olly (should add 1 copy Olly Shadow) right on to consider why the crash. When you find errors in this form, you should edit this Option in Olly:



Exception No. 1 is a check for the exception do this error caused the crash and we need to check can be run in debug. Then you will break when touched exception "Memory Access Violation". By No. 1 procedure mò against Stack, Trace Log and the break thanks to Condition 1 ... quite long time you will find the cause of the error. Trick do not deepen the technical how, just get the results that research to say for more understandable, avoid confused messages.

So, why the error? Easy to explain more, we see these steps (When load at OEP Encrypt or to go to 1 in the section 5C9000, you must click and select the following on 2 May):





Asked on the order that is "Search me orders compare the value in 1 covered with 0. And the results on 2 May:

Address	Disassembly	Comment
005D69ED	CMP DWORD PTR DS:[5D9F4C],0	DS:[005D9F4C]=7C800000 (kernel32.7C800000)
005DEB04	CMP DWORD PTR DS:[57ED78],0	DS:[0057ED78]=7C801D77 (kernel32.LoadLibraryA)
005E09AB	CMP DWORD PTR DS:[5ECE4C],0	DS:[005ECE4C]=00000000
005E735E	CMP DWORD PTR DS:[5C91BC],0	DS:[005C91BC]=77D40000 (user32.77D40000)
005E7F1B	CMP DWORD PTR DS:[5D9F44],0	DS:[005D9F44]=7C80B6A1 (kernel32.GetModuleHandleA)
005E8EC4	CMP DWORD PTR DS:[605D30],0	DS:[00605D30]=00153A70
005F8CA6	CMP DWORD PTR DS:[619434],0	DS:[00619434]=00000000
005FA3E6	MOV DWORD PTR SS:[ESP],EDX	(Initial CPU selection)
005FDFB6	CMP DWORD PTR DS:[605D30],0	DS:[00605D30]=00153A70
00606698	CMP DWORD PTR DS:[5C91BC],0	DS:[005C91BC]=77D40000 (user32.77D40000)
006069B6	CMP DWORD PTR DS:[5D9F44],0	DS:[005D9F44]=7C80B6A1 (kernel32.GetModuleHandleA)
0060796A	CMP DWORD PTR DS:[5DF438],0	DS:[005DF438]=00152CB0
00612554	CMP DWORD PTR DS:[5ECE4C],0	DS:[005ECE4C]=00000000
00615680	CMP DWORD PTR DS:[619434],0	DS:[00619434]=00000000
00617AEB	CMP DWORD PTR DS:[5CF498],0	DS:[005CF498]=77DD0000 (advapi32.77DD0000)
00617CB9	CMP DWORD PTR DS:[5C91BC],0	DS:[005C91BC]=77D40000 (user32.77D40000)
0061B961	CMP DWORD PTR DS:[5CF498],0	DS:[005CF498]=77DD0000 (advapi32.77DD0000)

Máy A - WinXP SP2

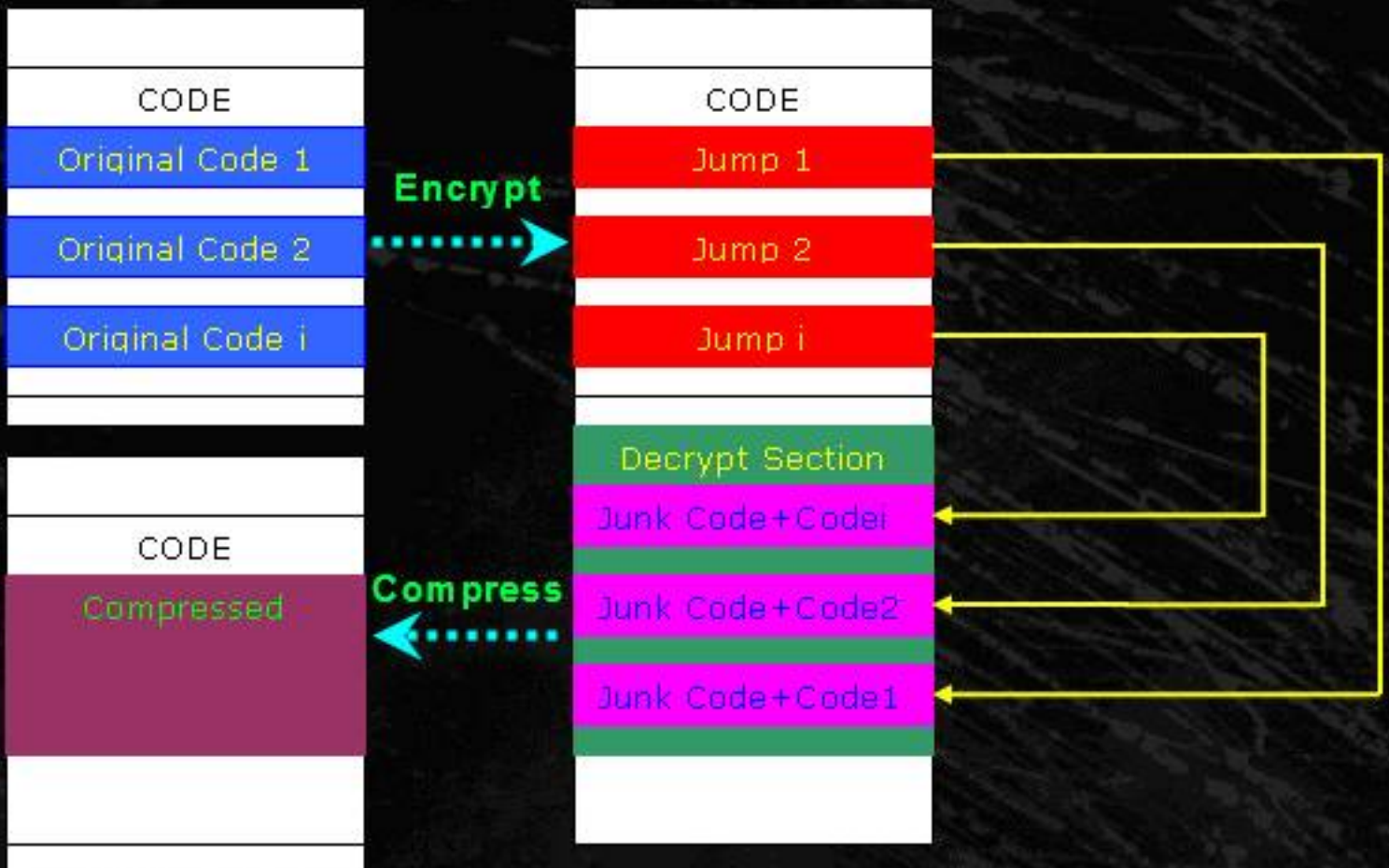
Address	Disassembly	Comment
005D69ED	CMP DWORD PTR DS:[5D9F4C],0	DS:[005D9F4C]=7C800000
005DEB04	CMP DWORD PTR DS:[57ED78],0	DS:[0057ED78]=7C801D77
005E09AB	CMP DWORD PTR DS:[5ECE4C],0	DS:[005ECE4C]=00000000
005E735E	CMP DWORD PTR DS:[5C91BC],0	DS:[005C91BC]=77D40000 (user32.77D40000)
005E7F1B	CMP DWORD PTR DS:[5D9F44],0	DS:[005D9F44]=7C80B6A1
005E8EC4	CMP DWORD PTR DS:[605D30],0	DS:[00605D30]=00153A70
005F8CA6	CMP DWORD PTR DS:[619434],0	DS:[00619434]=00000000
005FA3E5	PUSH ESI	(Initial CPU selection)
005FDFB6	CMP DWORD PTR DS:[605D30],0	DS:[00605D30]=00153A70
00606698	CMP DWORD PTR DS:[5C91BC],0	DS:[005C91BC]=77D40000 (user32.77D40000)
006069B6	CMP DWORD PTR DS:[5D9F44],0	DS:[005D9F44]=7C80B6A1
0060796A	CMP DWORD PTR DS:[5DF438],0	DS:[005DF438]=00152CB0
00612554	CMP DWORD PTR DS:[5ECE4C],0	DS:[005ECE4C]=00000000
00615680	CMP DWORD PTR DS:[619434],0	DS:[00619434]=00000000
00617AEB	CMP DWORD PTR DS:[5CF498],0	DS:[005CF498]=77DD0000 (ADVAPI32.77DD0000)
00617CB9	CMP DWORD PTR DS:[5C91BC],0	DS:[005C91BC]=77D40000 (user32.77D40000)
0061B961	CMP DWORD PTR DS:[5CF498],0	DS:[005CF498]=77DD0000 (ADVAPI32.77DD0000)

Máy B - WinXP SP1

So here are 2 points to note, first we found that by ImageBase module kernel32.dll (7C800000) on WinXP SP2 did not match ImageBase kernel32.dll on SP1 (Olly on SP1 should not be recognized). So the drag is just **LoadLibraryA** also always wrong. Article 2 is why the section 5C9000:

00400000	00001000	MultiTra		PE header
00401000	00156000	MultiTra	CODE	code
00557000	0000C000	MultiTra	DATA	data
00563000	00003000	MultiTra	BSS	
00566000	00004000	MultiTra	vbu.qmvp	
0056A000	00001000	MultiTra	ok3ytyit	
0056B000	00001000	MultiTra	.tls	
0056C000	00001000	MultiTra	.rdata	
0056D000	00016000	MultiTra	b0v9k07g	
00583000	00046000	MultiTra	.rsrc	resources
005C9000	00054000	MultiTra	9iw2t.87	SFX
0061D000	00121000	MultiTra	qsp5wxqs	
0073E000	00001000	MultiTra	k158u.cv	
0073F000	00004000	MultiTra	.mackt	imports

In this section, the file is 1 pack will remember empty area (00 bytes), but when you unpack that, to contain everything inside. So combining the above, plus the results that are the trick debug can draw the following: (see Figure)

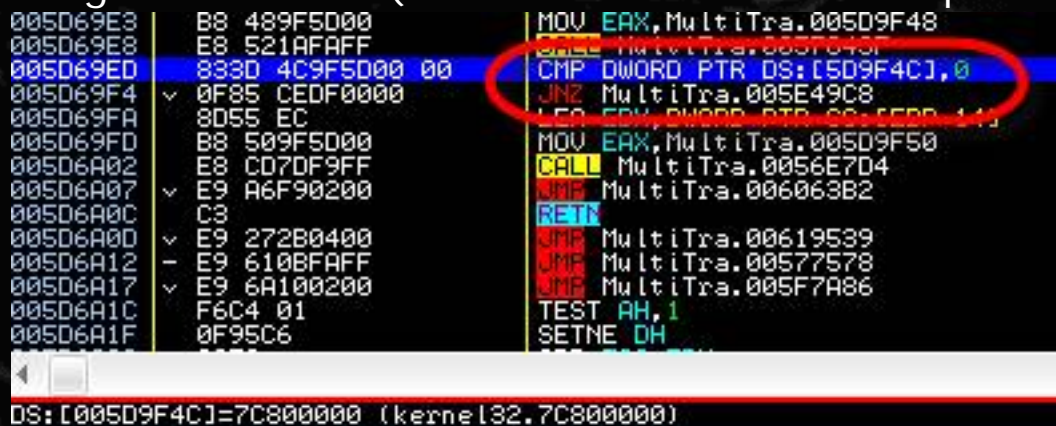


- Execryptor first encrypt a lot of orders in the soft, in each section will be replaced by encrypt command Jump 1 (with conditions or have ko)
- The jump will jump to 1 section for the implementation of the orders have been asked in the encrypt (temporarily called decrypt section), of course in more junk code
- Conduct encoding IAT
- Then new Execryptor conducted compress source code in the section (here is the code) and decrypt section (here is 5C9000) ...

When the dump file in Encrypt OEP, we just dump file is attached decrypt section but not yet completed the process unpack. To be completed decrypt this entire section, redirect the command bytes are replaced by the original order Jump in section code. Also in the unpack code in memory, has execryptor 1 in very interesting. As you see above, ImageBase kernel32.dll right in this, but wrong in another machine. Execryptor has stealthy ImageBase record of modules to the memory in decrypt section. So, I do it with this value?

When we use scripts to redirect the IAT, we have restored full value in the IAT, but CALL IAT to ensure the first restored all do right? Meanwhile, execryptor to encrypt a lot in the section code, and of course many of these will always stick CALL / JMP IAT. Imagebase Execryptor saved to decrypt the address of the API functions and implement the order CALL IAT this (assume the CALL [LoadLibraryA]) and for each module ImageBase the computer or a different address to the API functions to other and the dump file on the computer can not run on other machines.

The comparing it like to do? (You follow me to order Cmp 1 above):



```

005D69E3  B8 489F5D00  MOV EAX,MultiTra.005D9F48
005D69E8  E8 521AFAFF  CALL MultiTra.005D9F4C
005D69ED  833D 4C9F5D00 00 CMP DWORD PTR DS:[5D9F4C],0
005D69F4  0F85 CEDF0000 JNZ MultiTra.005E49C8
005D69FA  8D55 EC      LEA EBX,DWORD PTR DS:[5D9F4C]
005D69FD  B8 509F5D00  MOV EAX,MultiTra.005D9F50
005D6A02  E8 CD7DF9FF  CALL MultiTra.0056E7D4
005D6A07  0F85 06063B2 JMP MultiTra.006063B2
005D6A0C  C3          RETN
005D6A0D  0F85 19539 JMP MultiTra.00619539
005D6A12  0F85 77578 JMP MultiTra.00577578
005D6A17  0F85 F7A86 JMP MultiTra.005F7A86
005D6A1C  F6C4 01      TEST AH,1
005D6A1F  0F95C6      SETNE DH
  
```

DS:[005D9F4C]=7C800000 (kernel32.7C800000)

In addition to record ImageBase here, Execryptor always carefully before you get used to it. It will compare with 0 see here have NULL or do?

- If NULL prove it has not saved ImageBase Cmp order and will set up flags Z 1
- Order JNZ meet just below the flag Z = 1 will not run and jump down to go to the CALL command, the implementation process recover ImageBase (IM is by kernel32.dll) on the running
- If you have values stored in the flag Z = 0, JNZ jump, perform a decrypt the address 1 CALL IAT function is implemented and this function.

With the analysis, we see why the crash and then correct? When the address to which decrypt the wrong address compared to the API functions necessary to the course error. Kĩ But in comments to 2 pictures that we compare, we

do see only ImageBase modules that address both functions are saved again (LoadLibraryA). That is No. 1 CALL IAT function will be called directly based on the addresses stored, NULL if there are changes Z, Execryptor conducted to find where to save the value of the IM module, if the value of IM module also saved NULL the implementation of the IM to retrieve the module ... Then you know to fix the dump file is then đây. Execryptor will recover ImageBase of the modules if they see value in the region are NULL. One advantage of this and to fill all 00 positions will be cheated execryptor. But that is all? No! Things will only match when you eat the last items considered by haggar. But some things have not haggar to say in the next version of Execryptor (aged sure you) execryptor It is not only saved ImageBase but also save a lot Byte encryption. Byte code and can be born from ImageBase of modules, or by 1 method but it does ensure that the byte is not on the same machine. The byte also compare NULL or not? If NULL, the conduct of the byte code and, if not the NULL byte is used to decrypt the code should be conducted. (May be CALL IAT). And so will crash if the byte dump on this decrypt on another machine.

One section 5C9000 back for 1 case as an example:



Search for all orders by value from 1 to EAX cove. (EAX and EDX is 2 to record execryptor be used mainly when decrypt the encrypted bytes are stored):


```

005FC2FC MOV EAX,DWORD PTR DS:[68FFF71] [68FFF71E]=???
005FC395 MOV EAX,DWORD PTR DS:[5C91BC] [005C91BC]=77D40000
005FC417 MOV EAX,DWORD PTR DS:[5D55CC] DS:[005D55CC]=0B9D17C8
005FC506 MOV EAX,DWORD PTR DS:[25FF000] [25FF0000]=???
005FC8BE MOV EAX,DWORD PTR DS:[1700056] [1700056F]=???
005FCBFA MOV EAX,DWORD PTR DS:[50FFFDAC] [50FFFDAC]=???
005FCCF8 MOV EAX,DWORD PTR DS:[CAC3955] [CAC39558]=???
005FCD66 MOV EAX,DWORD PTR DS:[8FE991E] [8FE991EE]=???
005FCF37 MOV EAX,DWORD PTR DS:[E8C0097] [E8C0097D]=???
005FCFE5 MOV EAX,DWORD PTR DS:[5D3E701] DS:[005D3E70]=00000000
005FD1F6 MOV EAX,DWORD PTR DS:[D1FFF77] [D1FFF77A]=???
005FD227 MOV EAX,DWORD PTR DS:[83FFFFA] [83FFFFAC]=???
005FD27D MOV EAX,DWORD PTR DS:[59312C8] [59312C89]=???
005FD54C MOV EAX,DWORD PTR DS:[C681000] [C6810001]=???
005FD7C1 MOV EAX,DWORD PTR DS:[30] DS:[00000030]=???
005FD9E5 MOV EAX,DWORD PTR DS:[5B48000] [5B480000]=???
005FDA26 MOV EAX,DWORD PTR DS:[E900015] [E9000158]=???
005FDA44 MOV EAX,DWORD PTR DS:[3BBF000] [3BBF0000]=???
005FDC82 MOV EAX,DWORD PTR DS:[57D7F41] DS:[0057D7F4]=8F901253
005FDD09 MOV EAX,DWORD PTR DS:[A6E9013] [A6E90132]=???
005FDEE9 MOV EAX,DWORD PTR DS:[5ECE4C] [005ECE4C]=00000000
005FDF95 MOV EAX,DWORD PTR DS:[3489FFF] [3489FFFF]=???
005FE0E2 MOV EAX,DWORD PTR DS:[FF9FCFE] [FF9FCFE8]=???
005FE2B4 MOV EAX,DWORD PTR DS:[57D7E41] DS:[0057D7E4]=00000000
005FEB82 MOV EAX,DWORD PTR DS:[5ECE3C] DS:[005ECE3C]=00000000
005FEF25 MOV EAX,DWORD PTR DS:[AAA3C78] [AAA3C781]=???
005FEF2A MOV EAX,DWORD PTR DS:[243C870] [243C8702]=???
005FF10F MOV EAX,DWORD PTR DS:[C08158C] [C08158CE]=???
005FF293 MOV EAX,DWORD PTR DS:[8900015] [8900015A]=???
005FF37D MOV EAX,DWORD PTR DS:[56ECDC] DS:[0056ECDC]=00000000
005FF599 MOV EAX,DWORD PTR DS:[D381FFF] [D381FFF7]=???
005FF5EB MOV EAX,DWORD PTR DS:[E8FFF71] [E8FFF713]=???
005FF7D1 MOV EAX,DWORD PTR DS:[3CE3F4C] [3CE3F4C3]=???
005FFA65 MOV EAX,DWORD PTR DS:[F70000B] [F70000B4]=???
005FFA82 MOV EAX,DWORD PTR DS:[57ED781] [0057ED78]=7C801D77
005FFB55 MOV EAX,DWORD PTR DS:[7FF081F] [7FF081F0]=???

```

We see many special Byte be saved. Sometimes bytes is the address of the API function. Follow the line 1:

```

005FDC77 C3 RETN
005FDC78 - E9 1117F8FF JMP MultiTra.0057F38E
005FDC7D v E9 809E0100 JMP MultiTra.00617B02
005FDC82 8B05 F4D75700 MOV EAX,DWORD PTR DS:[57D7F41]
005FDC88 09C0 OR EAX,EAX
005FDC8A v 0F85 27E30100 JNZ MultiTra.0061BFB7
005FDC90 - E9 E752F8FF JMP MultiTra.00582F7C
005FDC95 v 0F86 F48E0100 JBE MultiTra.00616B8F
005FDC9B 81C7 62155000 ADD EDI,MultiTra.00501562
005FDCA1 873C24 XCHG DWORD PTR SS:[ESP],EDI
005FDCA4 v E9 422C0100 JMP MultiTra.006108EB
005FDCA9 83E1 07 AND ECX,7
005FDCAC B8 01000000 MOV EAX,1
005FDCB1 D3E0 SHL EAX,CL
005FDCB3 8B55 E4 MOV EDX,DWORD PTR SS:[EBP-1C]

```

DS:[0057D7F4]=8F901253
EAX=00000000

We see in the ImageBase execryptor bring compared with 0 and Z flag enabled using Cmp. But here? Execryptor also compare đấ Yes, but it does enable OR flags. And this order is also:

- Transfer value here to EAX
- Check ko NULL or set to flag Z
- ... Following a similar analysis of the modules ImageBase

Many explanations, many questions order form, a variety of junk code, but the idea of the author's execryptor still unclear. And it clicks at a execryptor fix that. Shallot NULL seats should we fill it to 00. On another machine dump file will run well.

The problem is: How do I know exactly remember all areas need to fill 00? Of course the first way is to debug the site, having any seats ko valid on another machine, then fill it to 00. And if players like debug hand you will lose not less than 1 day for the first time. (dom trick too should take 2 days) time is not long for the time we can break right where it should fill 00. As above, to identify any exception error is always happening, and any error is due to be encrypted Byte wrong. Since then mò opposed Stack, Trace Log in using Condition Breakpoint .. will find all the seats.

May the stars before doing this, the trick was thinking 1 method interesting for people to fix file dump 1 gently over how to debug. (but still quite long nhé) This method only requires patience but without knowledge subliminal at all. Invite people to stay 1 more time then to later.

2.Second dump:

Currently dump file on crash with SP2 on SP1, it is difficult to bear dump 1 files on SP1 see how it nhé (remember to install the soft on this but do not copy the original xi install wa small party). But the script "Bypass Antidebug" not very effective on SP1's trick, so why do we dump here? Rub rub, that is, after unpack in the memory we seize them Encrypt OEP:

Address	Hex dump	Disassembly
005FA8E5	56	PUSH ESI
005FA8E6	891424	MOV DWORD PTR SS:[ESP],EDX
005FA8E9	68 2676829A	PUSH 9A827626
005FA8EE	5A	POP EDX
005FA8EF	81E2 4EC26984	AND EDX,8469C24E
005FA8F5	81C2 77255E80	ADD EDX,805E2577
005FA8FD	EB 77030000	JMP MultiTask 006A7D77


So have Address = **5FA8E5**, 2 more memory Byte first **56 89**. Then we invite's out of this ABEL loader generator, turn to enter the parameters like below:

File

Detection method
☒ Standard
☐ Window caption
☐ Window class

Search for window caption/class name (and what to do later):
 Refresh

☐ Hide window ☐ Attempt to kill it ☐ Get ProcessID from handle

Choose icon: (default) 

Target program filename:

Generated loader/installer filename:

☐ Do file cleaning ☐ Do registry cleaning

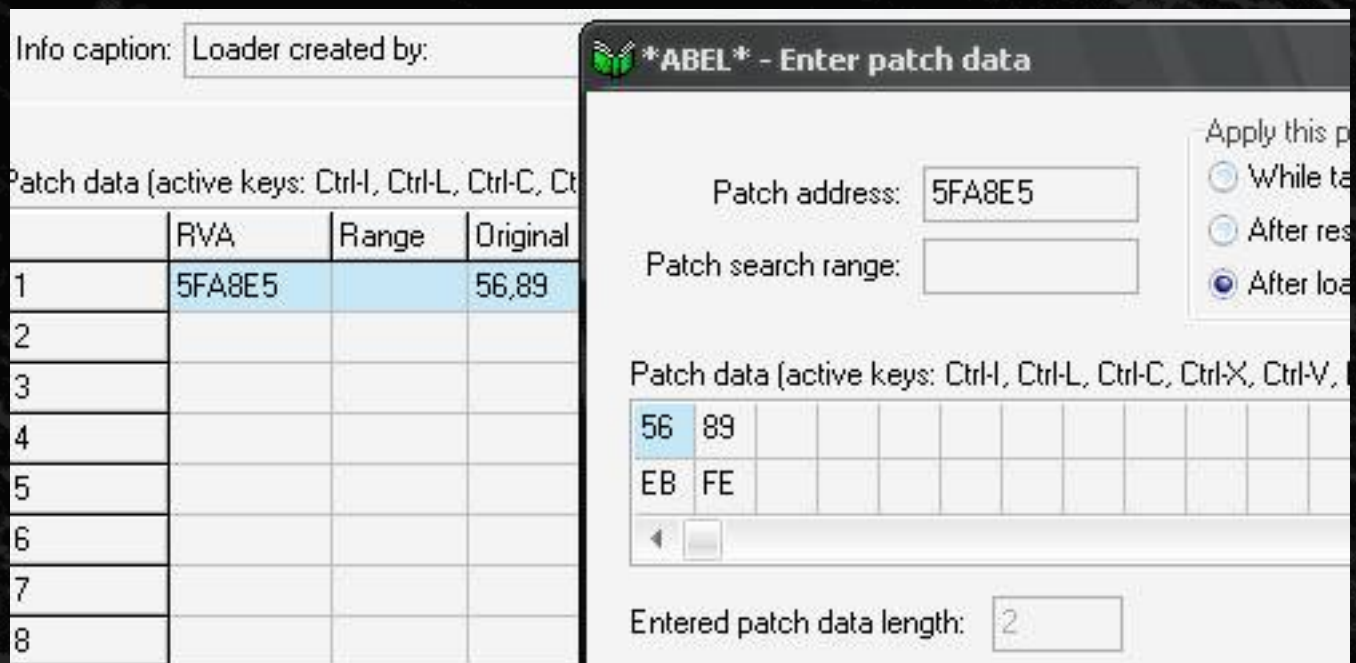
Info caption: Author name:

Timeout [sec]: Patch delay [sec]:

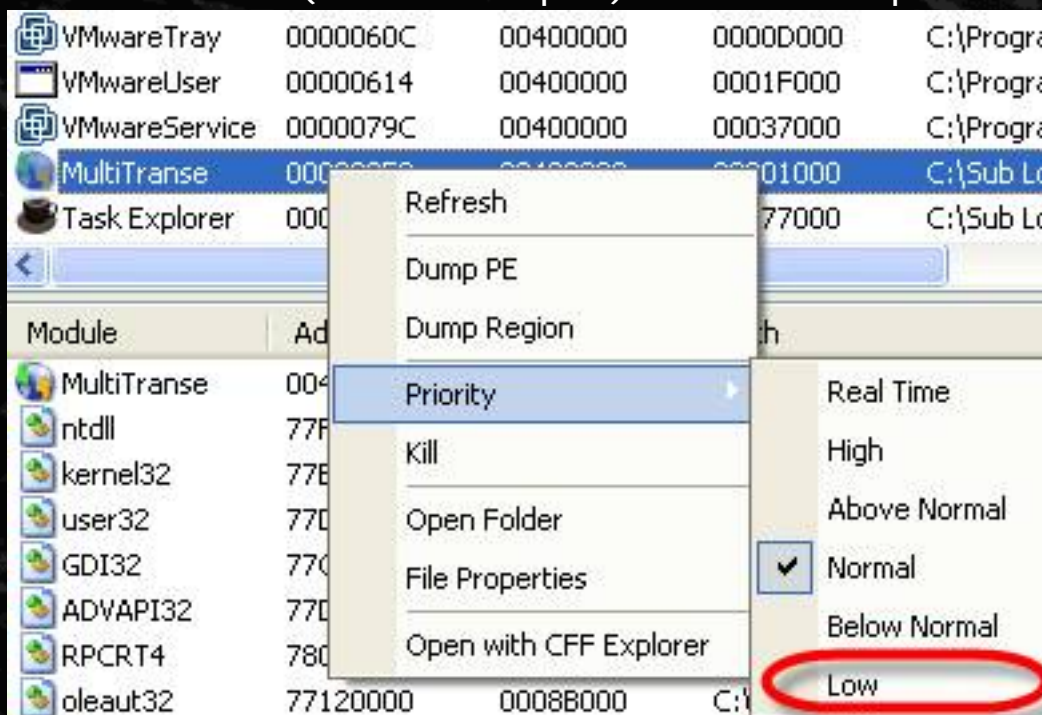
☒ Autolearning enabled ☐ Show splash screen
☒ Priority boost ☒ Ignore memory faults
☒ Enable debug mode
☐ First child process found is the main process

☐ Include simple installer
☐ Include DateFaker module

Author mail/www:

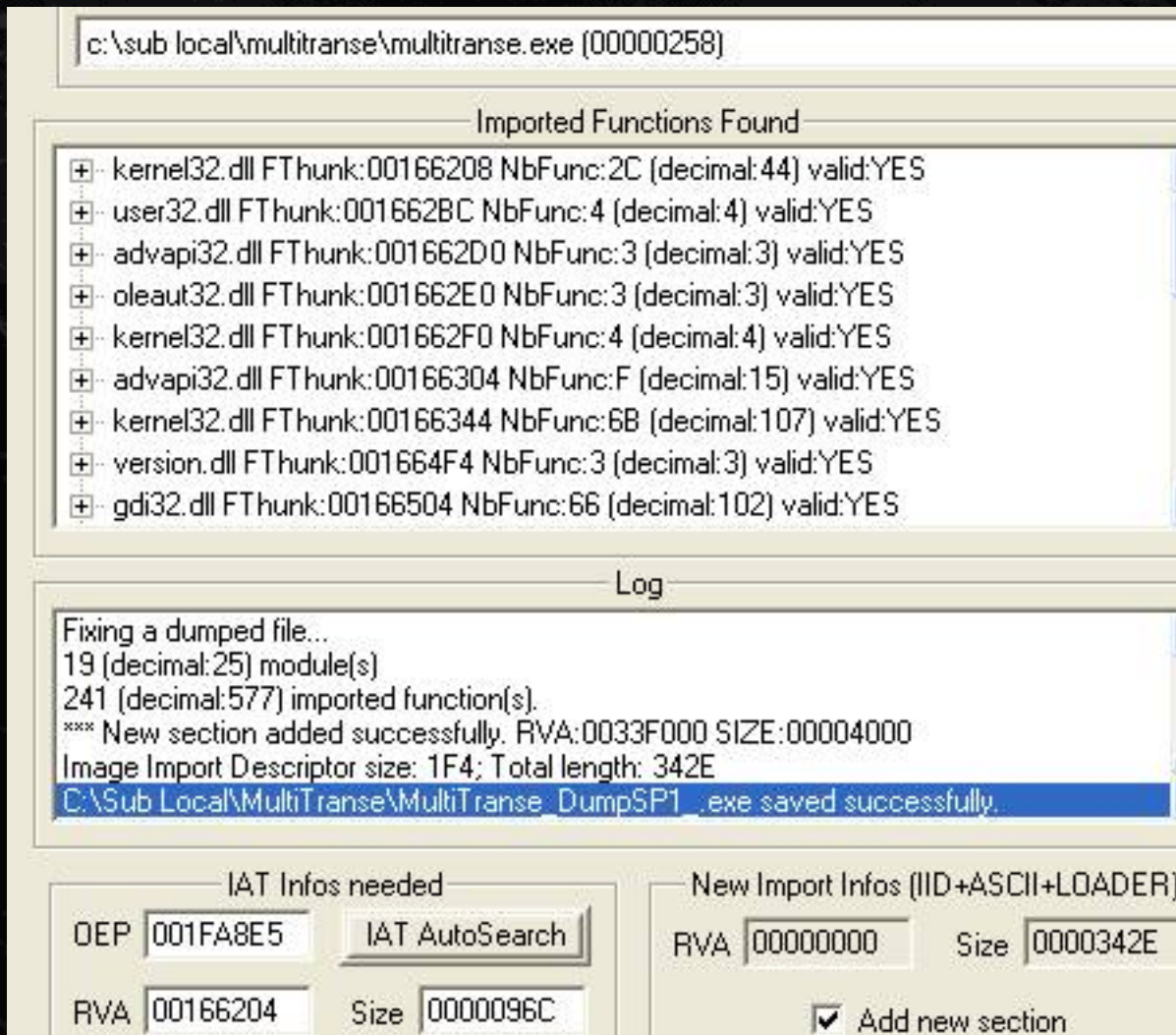


Then click to generate 1 Loader "all". One Tree file was made to save time by ImportREC fix IAT and loader through other SP1 machines. First run loader, it will load into memory MultiTranse.exe and enforcement, the file is just unpack that, it immediately to 5FA8E5 Goto and patch Byte 2 to **56 89 EB FE**. When soft to run the loop will be endless, we must quickly open 1 to process the view (or the dump is) will find the process running:

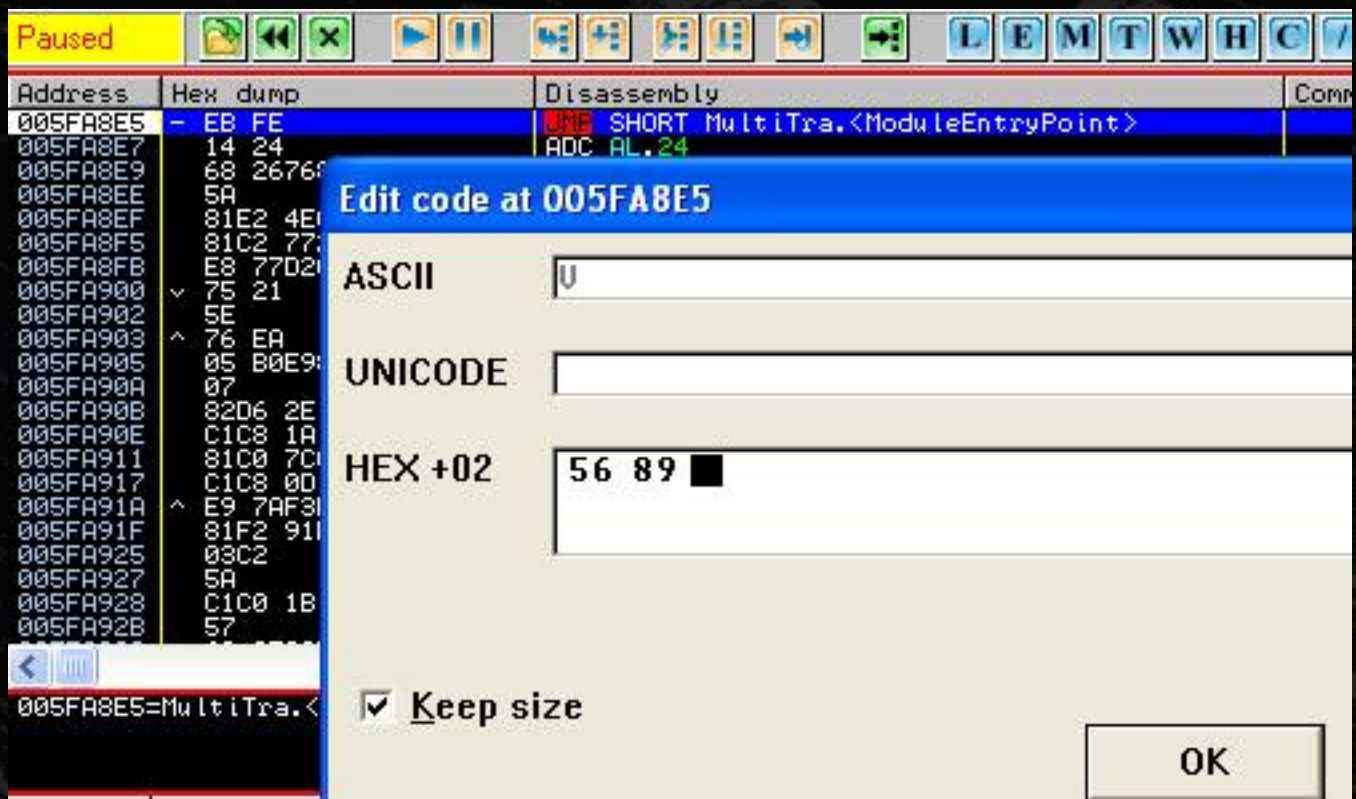


In small items trick, trick forget to do a task we must set a priority of this process of **Low** (or **Idle**) to avoid it consumes CPU is used too much, leading to the crash. Perhaps in this process does not need to do, but do be aware of the process, the other stars? If you have any study of Operating Systems will be more to the right priorities and the time slot between the process of how the trick... not too deep learning.

Now we are then dump. Then again ImportREC use, but remember that just load the file to save Tree has:



Shutting down the process Loop go. Load the file to fix, restore old Byte 2:

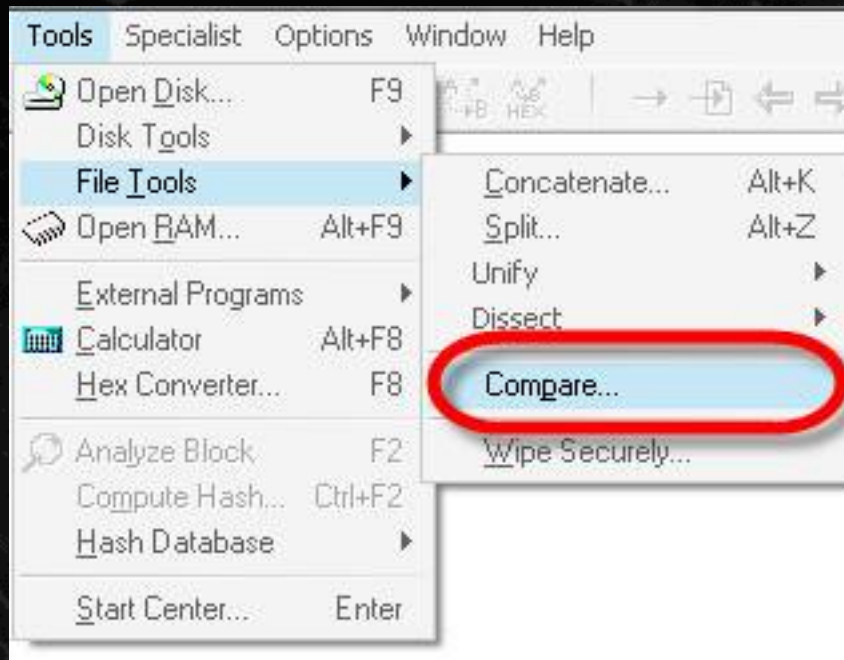


Save the file just fix. After that run, the file will run. Okie, hours But now we are doing what the quái? Dump file on the SP2 does not fix the dump 1 on SP1 children, life in the back more trouble then. Hehe, Calm down, we're close to the most interesting of this article đây. Invite over to the following:
D

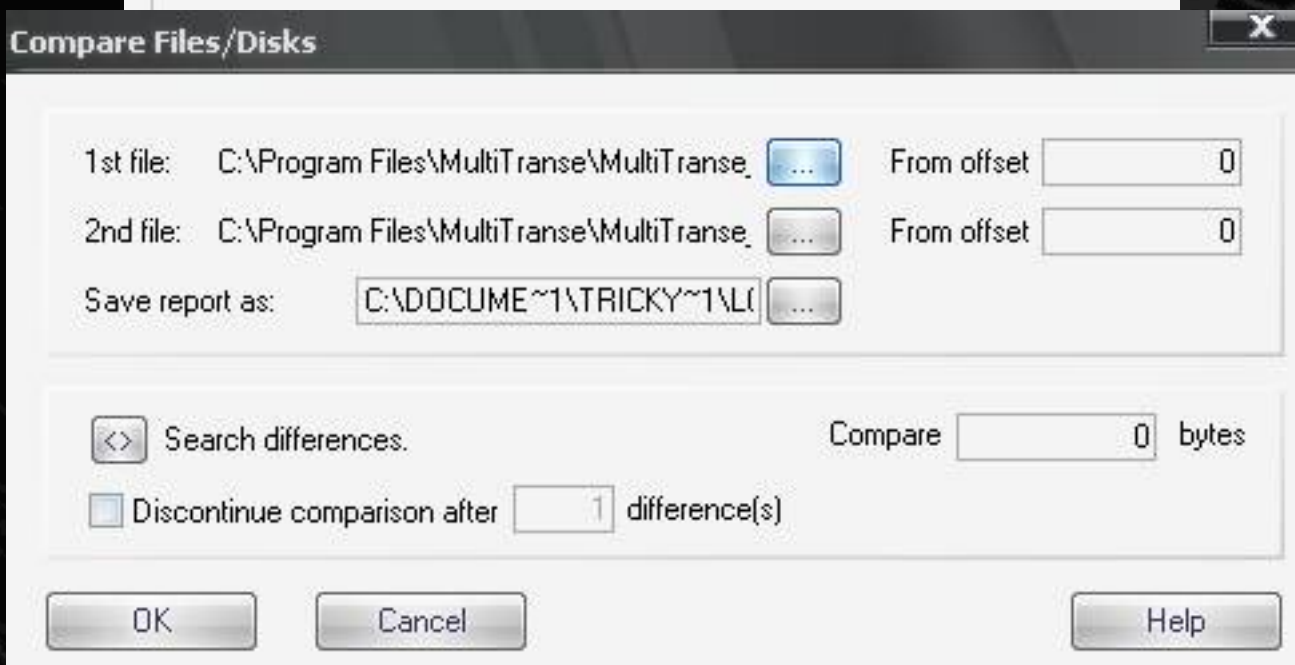
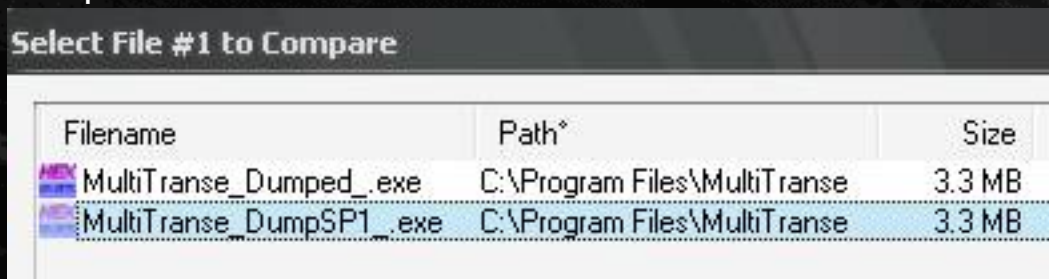
3.Compare and Fix:

Now think again. File dump on the other SP2 ko run on SP1, the SP1 ko running on SP2. But it tui all that different, the same size from the dump, the same PE Header, resource, and IAT á á, the Byte encryption and ImageBase on each machine is certainly different. And as the analysis, we only need to Fill in the Byte of which 00 are considered as the finish. So, where should debug patch to each place, we do not compare the position and have taken different bytes that the patch, hak hak hak. This trick is a trick ... U.S. not say much, we bring dump file on 1 May to 1 May remaining. Compare conduct is just. This trick made dump file from SP1 wa SP2 machine when compare, will revise the value in the dump file SP2, then bring wa SP1 test, if work is Okie.

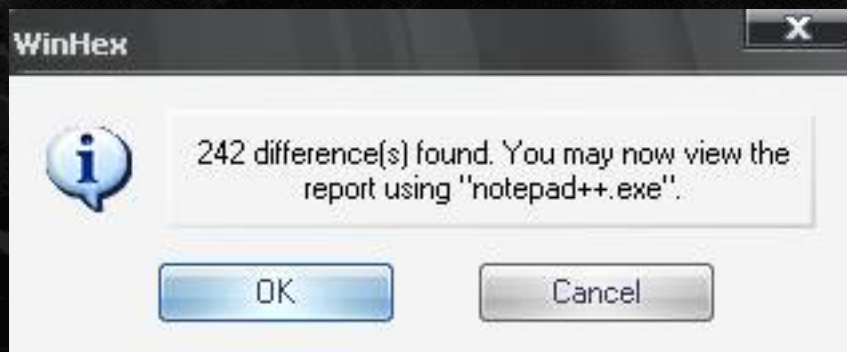
Compare also the art of Compare, not the message will be meaningless loss. WinHex trick used to compare because this is a very strong Hex Editor, handling files with high storage without fear Crash. Open Winhex to do, then:



2 files need Compare:



OK to compare:



Trick use notepad + + to show the report from winhex, because notepad by Win will be in the banana case report on the 300,000 Byte different. So here are 242 different Byte:

```

Report.txt
1  Search for differences
2
3  1. C:\Program Files\MultiTranse\MultiTranse_Dumped_.exe:
4  2. C:\Program Files\MultiTranse\MultiTranse_DumpSP1_.exe:
5  Offsets: hexadec.
6
7  16A0B4: A1 86
8  16A0B5: B6 AD
9  16A0B6: 80 E7
10 16A0B7: 7C 77
11 16A0B8: 77 61
12 16A0B9: 1D D9
13 16A0BA: 80 E7
14 16A0BB: 7C 77
15 16A0BC: A0 32
16 16A0BD: AD B3
17 16A0BE: 80 E7
18 16A0BF: 7C 77
19 16A0C0: DA FD
20 16A0C1: CD 98
  
```

But does not mean that we will patch 242 positions. We must patch by the following case:

Case 1: O A 3

Case 2: 4 a 7

Case 3: A B 8

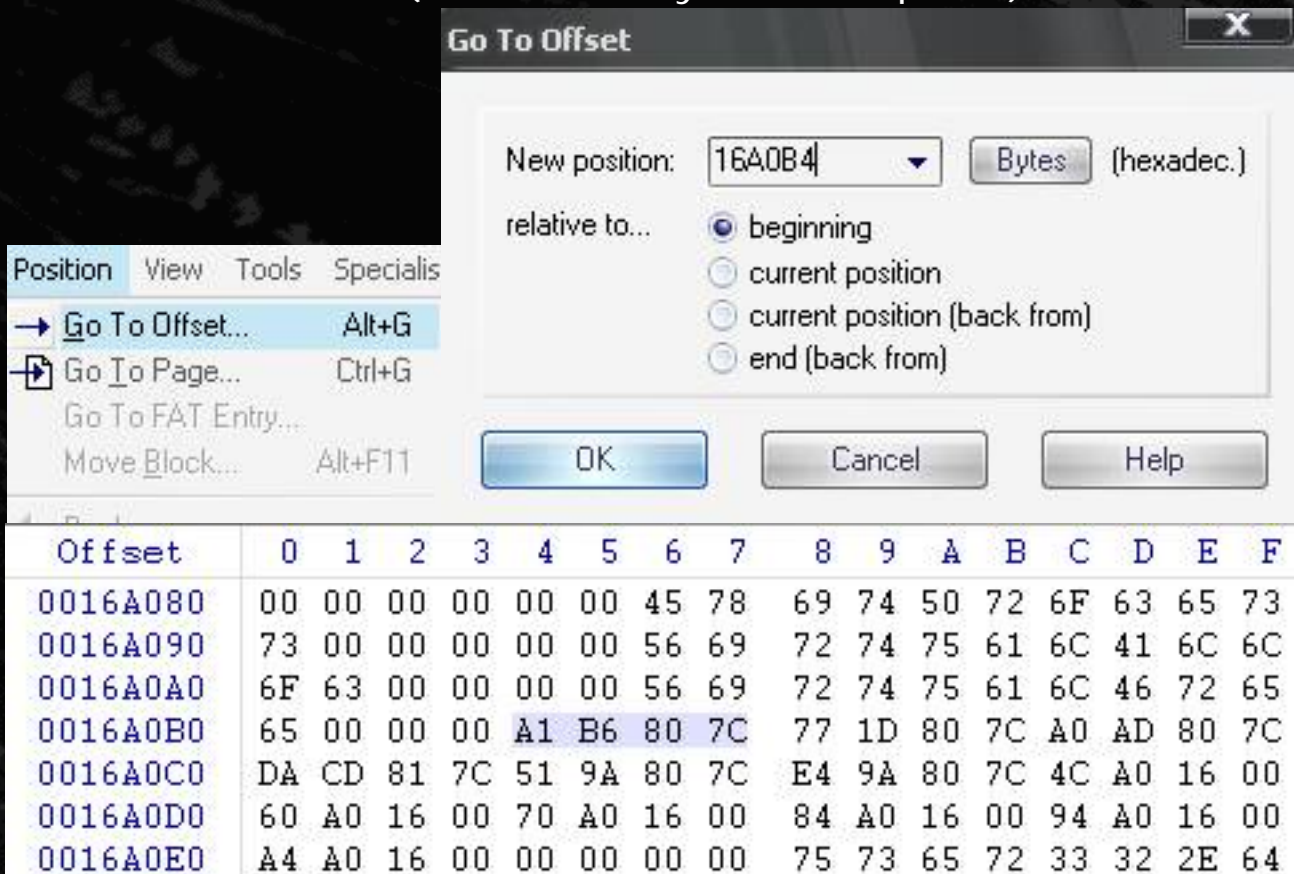
Case 4: C à F

That is we must always fill full 1 DWORD (4 bytes) by the address above.

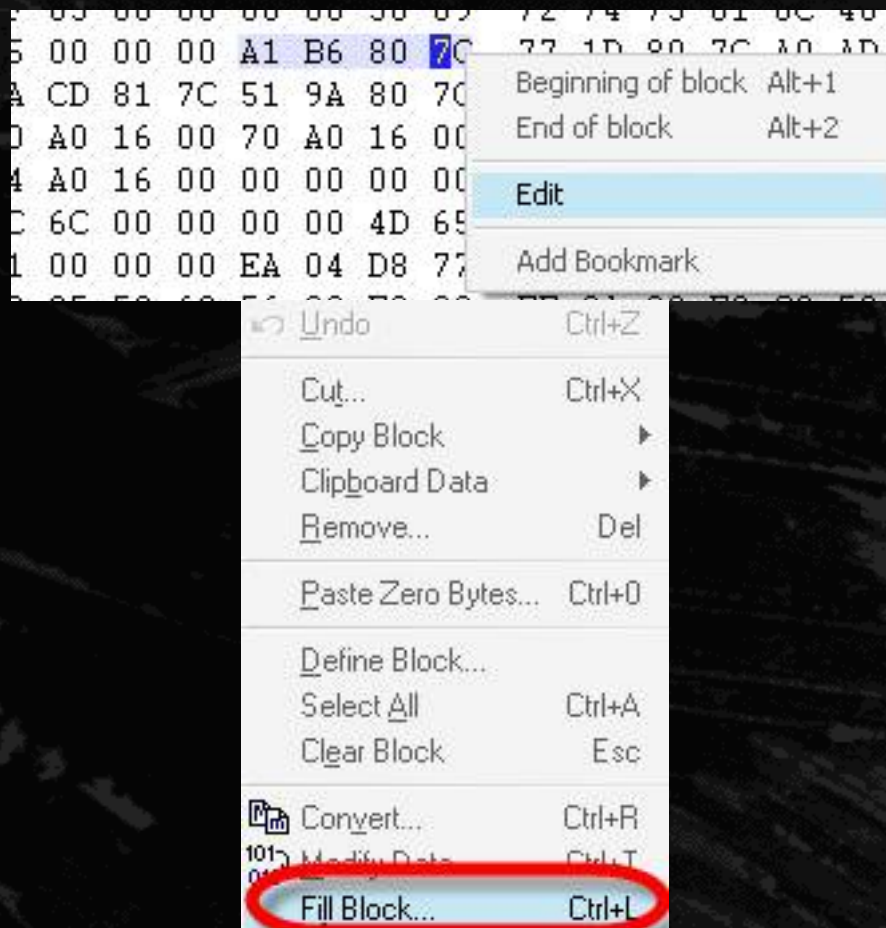
Retrieved from 1 Examples:

We see 16A0B4 is no difference of bytes, the number of rows in the address

of 4 case that is under 2, we will Fill 00 at 4 bytes in position 16A0B4, 16A0B5, 16A0B6, 16A0B7 (used to always Winhex patch):



Patch:



Fill Block

☒ Fill with hex values

00

☐ Fill with random bytes

Range: 0 to 255 (0..255)

Passes:

Pass #1

Add

Delete

< 0x00

< DoD

OK Cancel Help

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0016A080	00	00	00	00	00	00	45	78	69	74	50	72	6F	63	65	73
0016A090	73	00	00	00	00	00	56	69	72	74	75	61	6C	41	6C	6C
0016A0A0	6F	63	00	00	00	00	56	69	72	74	75	61	6C	46	72	65
0016A0B0	65	00	00	00	00	00	00	00	77	1D	80	7C	A0	AD	80	7C
0016A0C0	DA	CD	81	7C	51	9A	80	7C	E4	9A	80	7C	4C	A0	16	00
0016A0D0	60	A0	16	00	70	A0	16	00	84	A0	16	00	94	A0	16	00
0016A0E0	A4	A0	16	00	00	00	00	00	75	73	65	72	33	32	2E	64

With the remaining position, you feel the Offset Goto, to black, Ctrl-L and OK is completed.

Trick will be to some more examples:

At 16A0B8 (case 3), fill 00 in 16A0B8, 16A0B9, 16A0BA, 16A0BB:

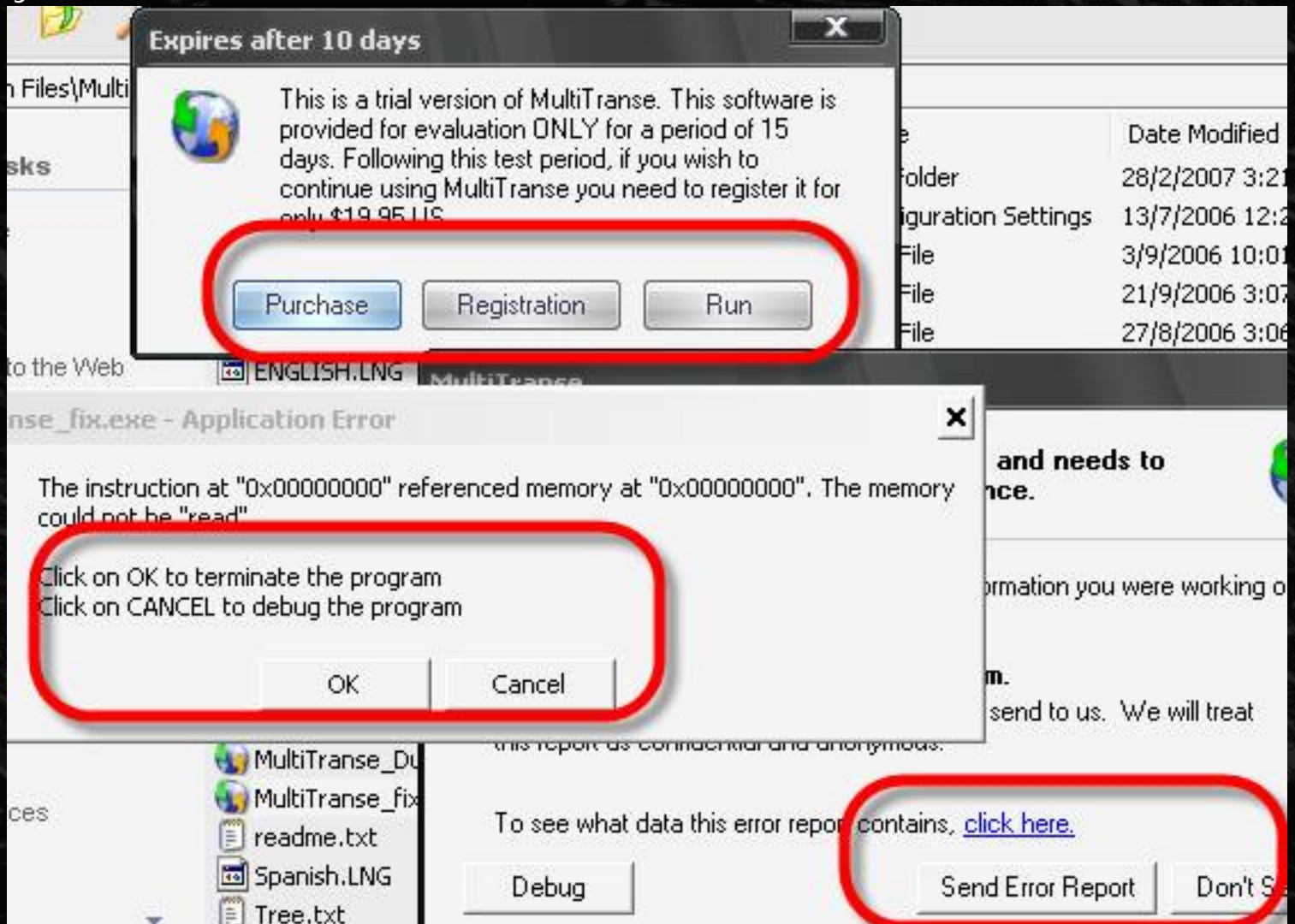
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0016A080	00	00	00	00	00	00	45	78	69	74	50	72	6F	63	65	73
0016A090	73	00	00	00	00	00	56	69	72	74	75	61	6C	41	6C	6C
0016A0A0	6F	63	00	00	00	00	56	69	72	74	75	61	6C	46	72	65
0016A0B0	65	00	00	00	00	00	00	00	00	00	00	00	A0	AD	80	7C
0016A0C0	DA	CD	81	7C	51	9A	80	7C	E4	9A	80	7C	4C	A0	16	00
0016A0D0	60	A0	16	00	70	A0	16	00	84	A0	16	00	94	A0	16	00

At 21941E (case 4) fill 00 at 21941C, 21941D, 21941E, 21941F:

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00219400	01	E9	5B	B6	FD	FF	E9	28	AE	F5	FF	E9	9A	8F	FB	FF
00219410	37	74	64	1B	63	2E	46	D8	B1	00	B4	ED	00	00	00	00
00219420	E8	3F	13	FF	FF	4D	A2	29	09	F6	05	10	56	E9	A3	86
00219430	FE	FF	22	46	00	00	00	00	E9	BA	77	FC	FF	E9	33	1C
00219440	FC	FF	8B	CC	81	C1	10	00	00	00	8B	09	C7	01	13	00

Every fill out is finished. Remember to fill the 4 on the case, to avoid cases

of mistaken Fill 00 on the E, F, 0.1 or 2,3,4,5. And always Fill each of 4 bytes, if the same address each other's case (as 16A0B4, 16A0B5) fill the first 1 is enough. You will wonder why the fill the 4 case so beautiful? Also it is in execryptor save bytes that the only beautiful too. According to the test of trick Zip Repair Tool 3.2 and how to fill the MultiTrans this good work. Subliminal than you can really code 1 tool and compare itself to fill 00 (based on 4 case made trick). Remember that if the Winhex used in this article, you must Goto and Fill out 76 times. Trick has saved text file 1, the address filter trick was to sit back and count the full 76 correct line. Fix finished Save nhé remember, do not have hi huc over 76 times that immediately close the contract. Now we take with the file via WinSP1 that any run on SP2 to see how this:



Soft Trial NAG still does demonstrate a good run but it comes with 2 errors ... hix hix, probably not fix this principle do wrong? Because the trick was done successfully on the Zip Repair Tool. Sure sure ... that considered, Zip Repair Tool no CRC check, but this month to know where it is, because when you fix Byte Such changes content when compared with their dump file, but this sure CRC byte is stored back in time to dump it will not show

NAG error if you run only dump file does not fix that.

Okie, items such as small trick has written, we Load file to fix this new Olly, Goto to **401,000** (section code):

Address	Hex dump	Disassembly
00401000	04 10	ADD AL,10
00401002	40	INC EAX
00401003	0003	ADD BYTE PTR DS:[EBX],AL
00401005	07	POP ES
00401006	42	INC EDX
00401007	6F	OUTS DX,DWORD PTR ES:[EDI]
00401008	6F	OUTS DX,DWORD PTR ES:[EDI]
00401009	6C	INS BYTE PTR ES:[EDI],DX
0040100A	2F 74	POP EDI

Ctrl-B Byte search form below:

Enter binary string to search for

ASCII

UNICODE

HEX +03

☒ Entire block

☐ Case sensitive

To place and is unique:

005556B0	C3	RETN
005556BE	8BC0	MOV EAX,EAX
005556C0	832D 18525600 01	SUB DWORD PTR DS:[565218],1
005556C7	73 1A	JNB SHORT MultiTra.005556E3
005556C9	E9 2B880200	JMP MultiTra.0057DEF9
005556CE	81C7 BFB870B9	ADD EDI,B970B8BF
005556D4	E9 C8F50800	JMP MultiTra.005E4CA1
005556D9	E9 8B060000	JMP MultiTra.00555D69
005556DE	1D B5F9D302	SBB EAX,2D3F9B5
005556E3	C3	RETN
005556F4	55	PUSH FRP

Patch of 1:

005556B0	C3	RETN
005556BE	8BC0	MOV EAX,EAX
005556C0	832D 18525600 01	SUB DWORD PTR DS:[565218],1
005556C7	73 1A	JNB SHORT MultiTra.005556E3
005556C9	E9 2B880200	JMP MultiTra.0057DEF9
005556CE	81C7 BFB870B9	ADD EDI,B970B8BF
005556D4	E9 C8F50800	JMP MultiTra.005E4CA1
005556D9	E9 8B060000	JMP MultiTra.00555D69
005556DE	1D B5F9D302	SBB EAX,2D3F9B5
005556E3	C3	RETN
005556F4	55	PUSH FRP

Save the file. File and run very mượt but not hú error.



Last night to try other machines (typically other SP1). File will make you happy: D

So the fix is a complete file to run on any computer. Too oải always pa kOn hen. However, though ...;)), the main issue is completed, trick adds about sub nè. We share the same over the next ...

V. Find Place Of Real OEP:

We return to Encrypt OEP, who have questions why the special trick name like or do?

Address	Hex dump	Disassembly
005FA8E5	56	PUSH ESI
005FA8E6	891424	MOV DWORD PTR SS:[ESP],EDX
005FA8E9	68 2676829A	PUSH 9A827626
005FA8EE	5A	POP EDX
005FA8EF	81E2 4EC26984	AND EDX,8469C24E
005FA8F5	81C2 77255E80	ADD EDX,805E2577
005FA8FB	50 77000000	CALL MultiTran_005FA87D77

1 OEP always true meaning to the section in the code (in this section is code). But the EP we found in the section decrypt - the results returned from the script bypass antidebug. Therefore it can not be OEP or we call in the other tut. This thinking has to encrypt it in real OEP. Now we will find its position to see why.

Soft with this code Delphi :



So GetModuleHandleA jaw jaw is always the first call. We set "BP GetModuleHandleA":

Command

Shift-F9 and break:

Address	Hex dump	Disassembly
7C80B6A1	8BFF	MOV EDI,EDI
7C80B6A3	55	PUSH EBP
7C80B6A4	8BEC	MOV EBP,ESP
7C80B6A6	837D 08 00	CMP DWORD PTR SS:[EBP+8],0
7C80B6AA	74 18	JE SHORT kernel32.7C80B6C4
7C80B6AC	FF75 08	PUSH DWORD PTR SS:[EBP+8]
7C80B6AF	E8 C0290000	CALL kernel32.7C80E074
7C80B6B4	85C0	TEST EAX,EAX
7C80B6B6	74 08	JE SHORT kernel32.7C80B6C0
7C80B6B8	FF70 04	PUSH DWORD PTR DS:[EAX+4]
7C80B6BB	E8 7D2D0000	CALL kernel32.GetModuleHandleW
7C80B6C0	5D	POP EBP
7C80B6C1	C2 0400	RET 4

For Return of

Address	Hex dump	Disassembly
7C80B729	50	PUSH EAX
7C80B72A	FF15 B412807C	CALL DWORD PTR DS:[<&ntdll.RtlImageNtHeader>]
7C80B730	85C0	TEST EAX,EAX
7C80B732	74 F84 41E9FFFF	JE kernel32.7C80A079
7C80B738	66:8378 5C 03	CMP WORD PTR DS:[EAX+5C],3
7C80B73D	74 F85 36E9FFFF	JNZ kernel32.7C80A079
7C80B743	33C0	XOR EAX,EAX
7C80B745	40	INC EAX
7C80B746	C3	RET
7C80B747	90	NOP

This section is still in kernel32, still less section of code, probably due to encrypt the above, we continue to Shift-F9 to break, and return:

00406D83	90	NOF
00406D84	53	PUSH EBX
00406D85	8BD8	MOV EBX,EAX
00406D87	33C0	XOR EAX,EAX
00406D89	A3 10375600	MOV DWORD PTR DS:[563710],EAX
00406D8E	6A 00	PUSH 0
00406D90	E8 2BFFFFFF	CALL <JMP.&kernel32.GetModuleHandleA>
00406D95	A3 18375600	MOV DWORD PTR DS:[563718],EAX
00406D9A	A1 18375600	MOV EAX,DWORD PTR DS:[563718]
00406D9F	A3 94705500	MOV DWORD PTR DS:[557094],EAX
00406DA4	33C0	XOR EAX,EAX
00406DA6	A3 98705500	MOV DWORD PTR DS:[557098],EAX
00406DAB	33C0	XOR EAX,EAX
00406DAD	A3 9C705500	MOV DWORD PTR DS:[55709C],EAX
00406DB2	E8 C1FFFFFF	CALL MultiTra.00406D78
00406DB7	BA 90705500	MOV EDI,MultiTra.00557090
00406DBC	8BC3	MOV EAX,EBX
00406DBE	E8 31D5FFFF	CALL MultiTra.004042F4
00406DC3	5B	POP EBX
00406DC4	C3	RET
00406DC5	8D40 00	LEA EAX,DWORD PTR DS:[EAX]

Correct form is the familiar functions of Delphi . But when Ctrl-A to Analyze, we see that the first function is not to find a call to:

00406D84	53	PUSH EBX
00406D85	8BD8	MOV EBX,EAX
00406D87	33C0	XOR EAX,EAX
00406D89	A3 10375600	MOV DWORD PTR DS:[563710],EAX
00406D8E	6A 00	PUSH 0
00406D90	E8 2BFFFFFF	CALL <JMP.&kernel32.GetModuleHandleA>
00406D95	A3 18375600	MOV DWORD PTR DS:[563718],EAX
00406D9A	A1 18375600	MOV EAX,DWORD PTR DS:[563718]
00406D9F	A3 94705500	MOV DWORD PTR DS:[557094],EAX
00406DA4	33C0	XOR EAX,EAX
00406DA6	A3 98705500	MOV DWORD PTR DS:[557098],EAX
00406DAB	33C0	XOR EAX,EAX
00406DAD	A3 9C705500	MOV DWORD PTR DS:[55709C],EAX
00406DB2	E8 C1FFFFFF	CALL MultiTra.00406D78
00406DB7	BA 90705500	MOV EDI,MultiTra.00557090
00406DBC	8BC3	MOV EAX,EBX
00406DBE	E8 31D5FFFF	CALL MultiTra.004042F4
00406DC3	5B	POP EBX
00406DC4	C3	RET
00406DC5	8D40 00	LEA EAX,DWORD PTR DS:[EAX]
00406DC8	55	PUSH EBP
00406DC9	8BEC	MOV EBP,ESP
00406DCB	33C0	XOR EAX,EAX
00406DCC	55	PUSH EBP

ESP=0055378C (MultiTra.0055378C) ← **Nothing**

Ham called to this always is very close to OEP. And do not see here prove it has been canceled. From this base also showed that the OEP is not only encrypt the stolen bytes more. May 1 star fast way to find placements for this case, that's Offset Encrypt OEP is **5FA8E5**. I pull up the section code, at 401,000, by search command:

Find command

JMP 5FA8E5

☒ Entire block

Find

Cancel

And then:

Address	Hex dump	Disassembly
0055505F	00	OR AX
00555060	E4565500	DD MultiTra.005556E4
00555064	E9 7C4B0A00	JMP MultiTra.<ModuleEntryPoint>
00555069	81C0 0FBC5F9F	ADD EAX,9F5FBC0F
0055506F	.	POP EDI
00555070	.	XCHG DWORD PTR SS:[ESP],EAX
00555073	C3	RET
00555074	E9 6CEB0100	JMP MultiTra.005748E5
00555079	C1CB 1F	ROR EBX,1F
0055507C	893C24	MOV DWORD PTR SS:[ESP],EDI
0055507F	5F	POP EDI
00555080	03DD	ADD EBX,EBP
00555082	E9 4EEFFFFFFF	JMP MultiTra.00554CD5
00555087	77	DB 77
00555088	87	DB 87
00555089	A7	DB A7
0055508A	07	DB 07
0055508B	E0	DB E0
0055508C	FA	DB FA
0055508D	97	DB 97
0055508E	18	DB 18
0055508F	BB 648920A1	MOV EBX,A1208964
00555094	A4	MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00555095	24 56	AND AL,56
00555097	00BA 00605500	ADD BYTE PTR DS:[EDX+5560D0],BH
0055509D	E8 DEE8EAF	CALL MultiTra.00404680
005550A2	E8 2134FEFF	CALL MultiTra.005391C8
005550A7	A1 18275600	MOV EAX,DWORD PTR DS:[562718]
005550AC	8B00	MOV EAX,DWORD PTR DS:[EAX]
005550AE	E8 2D38F1FF	CALL MultiTra.004695E0
005550B3	A1 18275600	MOV EAX,DWORD PTR DS:[562718]
005550B8	8B00	MOV EAX,DWORD PTR DS:[EAX]

Position circle that contains real OEP. It jump straight to the OEP Encrypt 1 Dong cancel stolen heavy below. So the franchise that would restore the original byte back to you nhé. Trick you only want to distinguish between Encrypt and Stolen Byte. Encrypt the code is still to ensure implementation of the only things you will not understand it do (to analyze the many new products are). But the Stolen Byte code will actually be done in 1 elsewhere, can not run in the code was stolen. Therefore Encrypt having difficulty reading the code, having Stolen the dump file will do is run peacetime J .

VI. Why?

During the message, the trick has many questions that you may also be questions and try to respond. This 2 more questions and answers to supplement the full on.

1. Tai stars execryptor code to encrypt and compress new source?

- If only compress the code can be uncompress, and then very vulnerable dissambler / debug through tools such as IDA, WDASM, Olly ... So to encrypt the code again, so files can be even unpack the also difficult to xài the tool through the crack in the soft.

2. Tai stars execryptor to save Byte encoding based ImageBase modules, during each run, execryptor unpack all the code, data is the excess or ko?

- It is not excessive if the thẳng as we just sit unpack / dump out the soft. The code of the summary is to make the dump file can not run on other

machines.

VII.Ending:

The battle took place is quite long you nhĩ? May have replied to finish. trick please remind me again ", the understanding is limited but knowledge is unlimited," in this article can not be sure from many mistakes, wrong on the technical, or thinking about how to use words, you expect please comment directly yourself. In the patch with WinHex, before the patent is 1 patcher do here, you should also patch first hand, the patch will train you for the patience. Also thanks to patience and passion, can trick should write this article đấy.

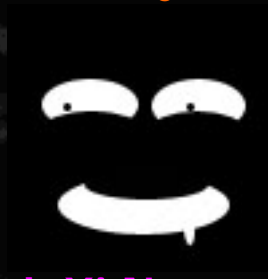
Also sorry you always have to read through the article Stupid Execryptor - Small trick, trick the items written in 1 mood any settlement should be used in a lot from khiếm house D. Who can complain to the disregard for, hope is fixing dump has not done one more frustration. Welcome all!

Big thanks to:

All REA's members: Computer_Angel, Moonbaby, Zombie, hacnho, benina, kienmanowar, rongchaua, Deux, Merc, hoadongnoi, the_lighthouse, TQN, light.phoenix, hytkl, tlandn, hurt_heart, dzungltn, Zoi, littleboy84, haule_nth, takada, Why not bar, iamidiot, Akira, dump, thienthandien, [kid], ...

Special thanks to:

fly, stephenteh, Gabri3l, MaDMAn_H3rCuL3s, CondZero, Ricardo Narvaja + NCR, lena151, haggar, ARTeam, snd, RES, CrackLatinos, all unpack. cn ... Authors who created tools and you.



Trick Xi News - 2007



Stupid Execryptor - Small trick

(laughs stories - funny story)

I-Introduction:

Identify II:

Meat-Man III:

Search the

I-Introduction:

Season to Valentine, the way people see dom we go deo Bong different players that play them, hix hix. The old well is not attractive to many emotions in the insurgent movement very a brother. The trick is they just opened down the mountain temple, she wrote a letter to his situation, but men det ới, the message written on a new hard work will, huhu, hours before hem with his situation zới of whether children who also belong to a " the country flavor heaven. " But after the application of capital lieng literature 5 points of the children, they also finished a man for her show, part of which he / she would like background music accompanied by the web. Mò also to open a 1 out of MIDI or rather, from the legs of ngenhac. info background music, blogs that it affected only play MP3 new pain, the children find the loay hoay-sop we convert from MIDI the MP3. Asked by children from radish MIDI the other file formats lem, the sop dom, tit sop, not a time trial and limit function. Damn damn.

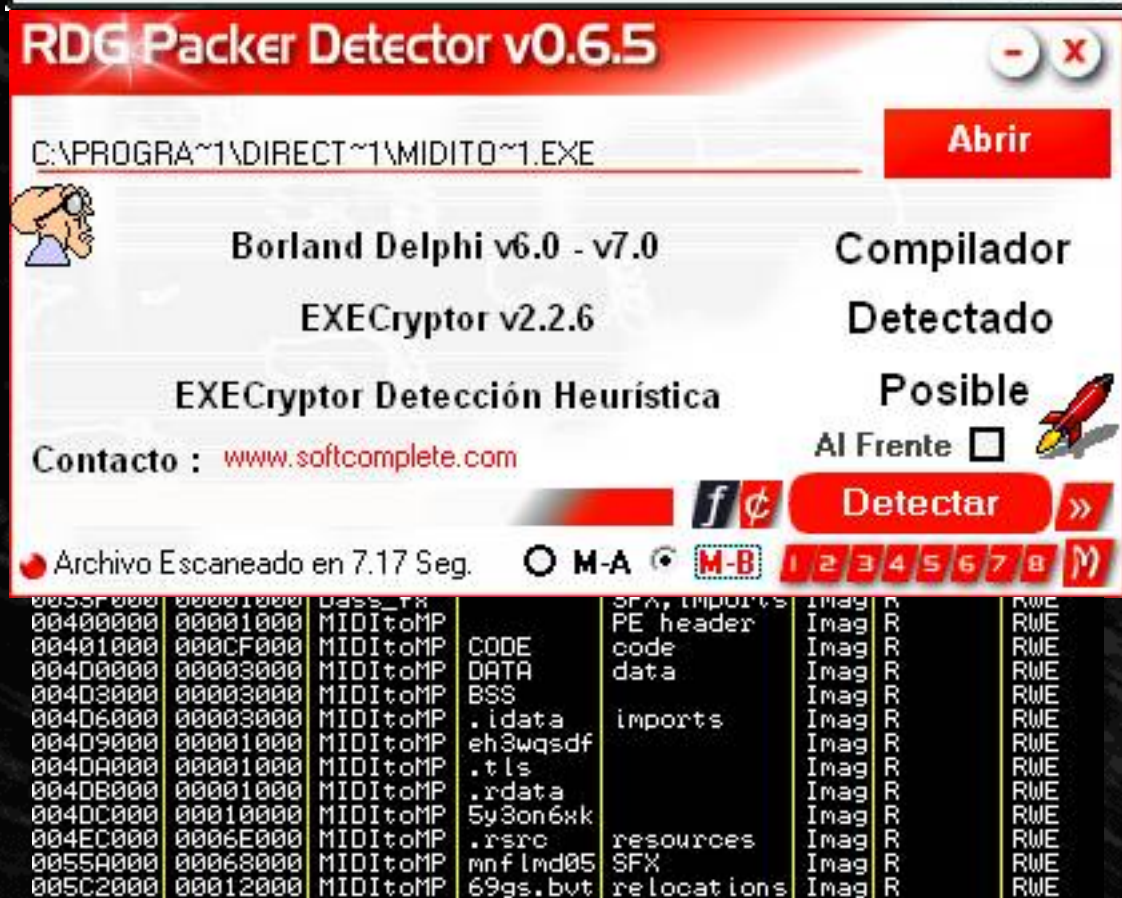
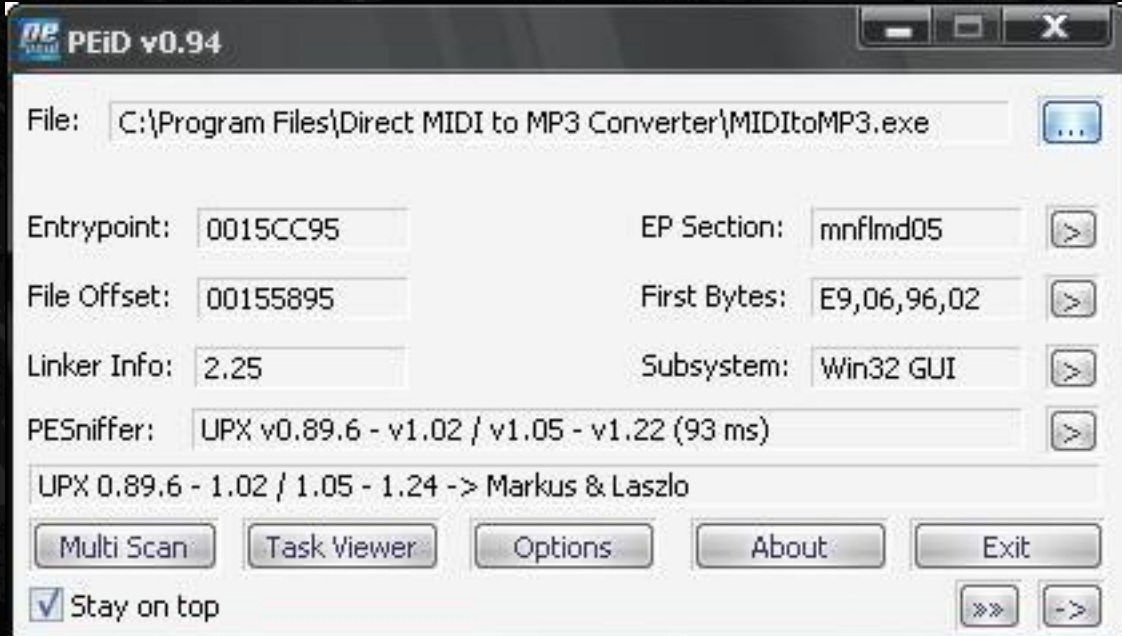
Then she is the guy "Direct MIDI to MP3 Converter. This guy or excellent, playing from MIDI convert the format mí lun. (Ho Ho, he stopped nghiũ we are brothers and experience, this child walking xí xon xì information lem lem, hem them know they should do more or GAY hi hi)

But he ới heaven, what does it feel NAG Trial hoài Ah, we make them mud. May stars func hem it can limit, they convert complete connection,

play blog lickerish oi. Now dom sop to the hands itch itch again. The mouse is the mò opened the "goods" of children. Hix hix. The game is the start:

II-Identify:

3 results this enough to bring children 1 overall look:



So, enough conclusion it is stated in Execryptor oi (PeiD as sources of UPX, the hash name section, and at least 1 results in true RDG 0.6.5)

III-man meat:

Call, the life it khôn the suffering, he hands itch to play right on the wild he will lose them under consideration. In the first hour that they have skin in the execryptor they chit now. Yes "to" throw them all he's Why. Why that time of the foreign children receive meals tum lum wife should hate firing home, wandering forever photos should not know where to find images tissue. May only know how incorporates time images back to life for children should also think that this little tí.

Yet, they load into Olly, it contracted pa it. Ah, I remember why he has the sound check System breakpoint. I just break it and will shrink to climb if they continue RUN RUN RUN. Recognizing that he execryptor right of the U.S. Olly.

But little of it is luck, brother-dom to the this:



Recognizing that information should not Detector 100&percent; but who have thec mec is how it is that Borland Delphi hem. Hi hi, so every load of children with Olly Breakpoint System, of course break here:

Address	Hex dump	Disassembly
7C901231	C3	RET
7C901232	8BFF	MOV EDI,EDI
7C901234	90	NOP
7C901235	90	NOP
7C901236	90	NOP
7C901237	90	NOP
7C901238	90	NOP
7C901239	CC	INT3
7C90123A	C3	RET
7C90123B	90	NOP

Now they fly and he nhé, to 401,000 vėjo section of code. Then she pulled down within walking 1:

0041B300	8BEC	MOV EBP,ESP	
0041B30F	81C4 ECFEFFFF	ADD ESP,-114	
0041B315	53	PUSH EBX	
0041B316	56	PUSH ESI	
0041B317	33C9	XOR ECX,ECX	
0041B319	8940 FC	MOV DWORD PTR SS:[EBP-4],ECX	
0041B31C	8BF2	MOV ESI,EDX	
0041B31E	8BD8	MOV EBX,EAX	
0041B320	33C0	XOR EAX,EAX	
0041B322	55	PUSH EBP	
0041B323	68 C2B34100	PUSH MIDItomP.0041B3C2	
0041B328	64:FF30	PUSH DWORD PTR FS:[EAX]	
0041B32B	64:8920	MOV DWORD PTR FS:[EAX],ESP	
0041B32E	85F6	TEST ESI,ESI	
0041B330	74 1D	JE SHORT MIDItomP.0041B34F	
0041B332	8D95 FCFEFFFF	LEA EDX,DWORD PTR SS:[EBP-104]	
0041B338	8B06	MOV EAX,DWORD PTR DS:[ESI]	
0041B33A	E8 A189FEFF	CALL MIDItomP.00403CE0	
0041B33F	8D95 FCFEFFFF	LEA EDX,DWORD PTR SS:[EBP-104]	
0041B345	8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]	
0041B348	E8 739AFEFF	CALL MIDItomP.00404DC0	
0041B34D	EB 0D	JMP SHORT MIDItomP.0041B35C	
0041B34F	8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]	
0041B352	BA D8B34100	MOV EDX,MIDItomP.0041B3D8	ASCII "nil"
0041B357	E8 9898FEFF	CALL MIDItomP.00404BF4	
0041B35C	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
0041B35F	8985 ECFEFFFF	MOV DWORD PTR SS:[EBP-114],EAX	
0041B365	C685 F0FEFFFF	MOV BYTE PTR SS:[EBP-110],0B	
0041B36C	8D95 FCFEFFFF	LEA EDX,DWORD PTR SS:[EBP-104]	
0041B372	8B03	MOV EAX,DWORD PTR DS:[EBX]	
0041B374	E8 6789FEFF	CALL MIDItomP.00403CE0	
0041B379	8D85 FCFEFFFF	LEA EAX,DWORD PTR SS:[EBP-104]	
0041B37F	8985 F4FEFFFF	MOV DWORD PTR SS:[EBP-10C],EAX	
0041B385	C685 F8FEFFFF	MOV BYTE PTR SS:[EBP-108],4	
0041B38C	8D85 ECFEFFFF	LEA EAX,DWORD PTR SS:[EBP-114]	
0041B392	50	PUSH EAX	
0041B393	6A 01	PUSH 1	
0041B395	8B0D F81C4D00	MOV ECX,DWORD PTR DS:[4D1CF8]	MIDItomP.0041678C
0041B39B	B2 01	MOV DL,1	

With the eyes wen phin xXx considered by them, they recognize that this month it has Cuc pack as the section CODE sop to protect all types executr. That is when you run, it's not unpack code in memory. So, also it only wants to Execryptor 2 things:

- + Based quite the Anti-Debug Execryptor's very strong for Olly
- + Borrowing the junk code kì very difficult to analyze by executr 1 to encrypt the original code.

To the temporary enough, they believe is right to recognize the RDG is sop using Borland Delphi. According Why, photos Khoai GetModuleHandleA the very function, from which the image to open up OEP. She is quite thick, they risk executr thought it right to cancel immediately threaten the IAT will? I find OEP Borland's way of goof. She tuốt populations ... I flipped, she tuốt luot thanh truoc down bottom section code:

Address	Hex dump	Disassembly
004CFFC2	0000	ADD BYTE PTR DS:[EAX],AL
004CFFC4	0000	ADD BYTE PTR DS:[EAX],AL
004CFFC6	0000	ADD BYTE PTR DS:[EAX],AL
004CFFC8	0000	ADD BYTE PTR DS:[EAX],AL
004CFFCA	0000	ADD BYTE PTR DS:[EAX],AL
004CFFCC	0000	ADD BYTE PTR DS:[EAX],AL
004CFFCE	0000	ADD BYTE PTR DS:[EAX],AL
004CFFD0	0000	ADD BYTE PTR DS:[EAX],AL
004CFFD2	0000	ADD BYTE PTR DS:[EAX],AL
004CFFD4	0000	ADD BYTE PTR DS:[EAX],AL
004CFFD6	0000	ADD BYTE PTR DS:[EAX],AL
004CFFD8	0000	ADD BYTE PTR DS:[EAX],AL
004CFFDA	0000	ADD BYTE PTR DS:[EAX],AL
004CFFDC	0000	ADD BYTE PTR DS:[EAX],AL
004CFFDE	0000	ADD BYTE PTR DS:[EAX],AL
004CFFE0	0000	ADD BYTE PTR DS:[EAX],AL
004CFFE2	0000	ADD BYTE PTR DS:[EAX],AL
004CFFE4	0000	ADD BYTE PTR DS:[EAX],AL
004CFFE6	0000	ADD BYTE PTR DS:[EAX],AL
004CFFE8	0000	ADD BYTE PTR DS:[EAX],AL
004CFFEA	0000	ADD BYTE PTR DS:[EAX],AL
004CFFEC	0000	ADD BYTE PTR DS:[EAX],AL
004CFFEE	0000	ADD BYTE PTR DS:[EAX],AL
004CFFF0	0000	ADD BYTE PTR DS:[EAX],AL
004CFFF2	0000	ADD BYTE PTR DS:[EAX],AL
004CFFF4	0000	ADD BYTE PTR DS:[EAX],AL
004CFFF6	0000	ADD BYTE PTR DS:[EAX],AL
004CFFF8	0000	ADD BYTE PTR DS:[EAX],AL
004CFFFA	0000	ADD BYTE PTR DS:[EAX],AL
004CFFFC	0000	ADD BYTE PTR DS:[EAX],AL
004CFFFE	0000	ADD BYTE PTR DS:[EAX],AL

Then they try to pull from to see the code:

Address	Hex dump	Disassembly
004CFA6F	0014F7	ADD BYTE PTR DS:[EDI+ESI*8],DL
004CFA72	4C	DEC ESP
004CFA73	0055 8B	ADD BYTE PTR SS:[EBP-75],DL
004CFA76	EC	IN AL,DX
004CFA77	83C4 F0	ADD ESP,-10
004CFA7A	53	PUSH EBX
004CFA7B	B8 3CF74C00	MOV EAX,MIDItomP.004CF73C
004CFA80	E8 A373F3FF	CALL MIDItomP.00406E28
004CFA85	8B1D 581E4D00	MOV EBX,DWORD PTR DS:[4D1E58]
004CFA88	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFA8D	E8 3E6DFBFF	CALL MIDItomP.004867D0
004CFA92	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFA94	BA 6CFB4C00	MOV EDI,MIDItomP.004CFB6C
004CFA99	E8 1A69FBFF	CALL MIDItomP.004863B8
004CFA9E	8B0D A01B4D00	MOV ECX,DWORD PTR DS:[4D1BA0]
004CFAA4	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFAA6	8B15 A0AB4C00	MOV EDI,DWORD PTR DS:[4CABA0]
004CFAAE	E8 376DFBFF	CALL MIDItomP.004867E8
004CFAB1	8B0D D41B4D00	MOV ECX,DWORD PTR DS:[4D1BD4]
004CFAB7	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFAB9	8B15 D8F04B00	MOV EDI,DWORD PTR DS:[4BF0D8]
004CFABF	E8 246DFBFF	CALL MIDItomP.004867E8
004CFAC4	8B0D 74204D00	MOV ECX,DWORD PTR DS:[4D2074]
004CFACA	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFACC	8B15 E40F4C00	MOV EDI,DWORD PTR DS:[4C0FE4]
004CFAD2	E8 116DFBFF	CALL MIDItomP.004867E8
004CFAD7	8B0D D01F4D00	MOV ECX,DWORD PTR DS:[4D1FD0]
004CFADD	8B03	MOV EAX,DWORD PTR DS:[EBX]

UI skin, so they see this right here is Delphi , As part of the EP sop in Borland Delphi or is in the last section CODE lem. Dom's hem brothers know what it thinks Stolen Byte hông hen? Cok a monkey at all, because you have not Analyze for Olly thui. What he hit Ctrl-A to help children hen:

Address	Hex dump	Disassembly
004CFA6D	00	DB 00
004CFA6E	00	DB 00
004CFA6F	00	DB 00
004CFA70	14F74C00	DD MIDItoMP.004CF714
004CFA74	55	PUSH EBP
004CFA75	8BEC	MOV EBP,ESP
004CFA77	83C4 F0	ADD ESP,-10
004CFA7A	53	PUSH EBX
004CFA7B	B8 3CF74C00	MOV EAX,MIDItoMP.004CF73C
004CFA80	E8 A373F3FF	CALL MIDItoMP.00406E28
004CFA85	8B1D 581E4D00	MOV EBX,DWORD PTR DS:[4D1E58]
004CFA8B	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFA8D	E8 3E6DFBFF	CALL MIDItoMP.004867D0
004CFA92	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFA94	BA 6CFB4C00	MOV EDI,MIDItoMP.004CFB6C
004CFA99	E8 1A69FBFF	CALL MIDItoMP.004863B8
004CFA9E	8B0D A01B4D00	MOV ECX,DWORD PTR DS:[4D1BA0]
004CFAA4	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFAA6	8B15 A0AB4C00	MOV EDI,DWORD PTR DS:[4CABA0]
004CFAAC	E8 376DFBFF	CALL MIDItoMP.004867E8
004CFAB1	8B0D D41B4D00	MOV ECX,DWORD PTR DS:[4D1BD4]
004CFAB7	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFAB9	8B15 D8F04B00	MOV EDI,DWORD PTR DS:[4BF0D8]
004CFABF	E8 246DFBFF	CALL MIDItoMP.004867E8
004CFAC4	8B0D 74204D00	MOV ECX,DWORD PTR DS:[4D2074]

Life begins with little light camera hen, OEP am now vet Asia. Now you mò IAT nhe. Mí foggy What type is, they are mò function probably due to them in section mu CODE:

Search for	Name (label) in current module	Ctrl+N
Find references to	Name in all modules	
View		
Copy to executable	Command	Ctrl+F
Analysis	Sequence of commands	Ctrl+S
	Constant	
Asm2Clipboard	Binary string	Ctrl+B
Bookmark		
	All intermodular calls	

1 Dong nè out:

Address	Disassembly	Destination
004CC8A4	CALL <JMP.&bass.BASS_Stop>	bass.BASS_Stop
004CC8B8	CALL <JMP.&bass.BASS_StreamFree>	bass.BASS_StreamFree
004CC8D0	CALL <JMP.&bass.BASS_Free>	bass.BASS_Free
004CCA4C	CALL <JMP.&shell32.DragQueryFileA>	shell32.DragQueryFileA
004CCA69	CALL <JMP.&shell32.DragQueryFileA>	shell32.DragQueryFileA
004CCA95	CALL <JMP.&shell32.DragFinish>	shell32.DragFinish
004CCC26	CALL <JMP.&shell32.ShellExecuteA>	shell32.ShellExecuteA
004CCC4E	CALL <JMP.&shell32.ShellExecuteA>	shell32.ShellExecuteA
004CCCFC	CALL <JMP.&shell32.ShellExecuteA>	shell32.ShellExecuteA
004CD3CA	CALL <JMP.&bass.BASS_Init>	bass.BASS_Init
004CD3F1	CALL <JMP.&bass.BASS_Start>	bass.BASS_Start
004CD3FD	CALL <JMP.&bass.BASS_PluginLoad>	bass.BASS_PluginLoad
004CD409	CALL <JMP.&bass.BASS_PluginLoad>	bass.BASS_PluginLoad
004CDC91	CALL MIDItoMP.00567AE8	MIDItoMP.00567AE8
004CDD66	CALL <JMP.&bass.BASS_StreamFree>	bass.BASS_StreamFree
004CDD6E	CALL <JMP.&bass.BASS_StreamFree>	bass.BASS_StreamFree
004CDD88	CALL <JMP.&bassmidi.BASS_MIDI_StreamCreateFile>	bassmidi.BASS_MIDI_StreamCreateFile
004CDD9D	CALL <JMP.&bass_fx.BASS_FX_TempoCreate>	bass_fx.BASS_FX_TempoCreate
004CDE01	CALL <JMP.&bass.BASS_ChannelSetFX>	bass.BASS_ChannelSetFX
004CDE2F	CALL <JMP.&bass_fx.BASS_FX_TempoSet>	bass_fx.BASS_FX_TempoSet
004CDE39	CALL <JMP.&bass.BASS_ChannelPlay>	bass.BASS_ChannelPlay
004CDE44	CALL <JMP.&bass.BASS_ChannelGetLength>	bass.BASS_ChannelGetLength
004CDE51	CALL <JMP.&bass.BASS_ChannelBytes2Seconds>	bass.BASS_ChannelBytes2Seconds

She also ranked second hem itself as what quá, double-click function of 1 is:


```

004CCC37 . 6A 00          PUSH 0
004CCC39 . A1 9C194D00    MOV EAX,DWORD PTR DS:[4D199C]
004CCC3E . E8 D983F3FF    CALL MIDItomp.0040501C
004CCC43 . 50             PUSH EAX
004CCC44 . 6A 00          PUSH 0
004CCC46 . 8BC3           MOV EAX,EBX
004CCC48 . E8 53FBF9FF    CALL MIDItomp.0046C7A0
004CCC4D . 50             PUSH EAX
004CCC4E . E8 6D57F6FF    CALL <JMP.&shell32.ShellExecuteA>
004CCC53 . 5B             POP EBX
004CCC54 . C3             RETN
004CCC55 . 8D40 00        LEA EAX,DWORD PTR DS:[EAX]
004CCC58 . 55             PUSH EBP

```

The men they call this CALL:

```

004323A0 . 832D 543A4D00 01 SUB DWORD PTR DS:[4D3A54],1
004323A7 . C3             RETN
004323A8 $- FF25 4C694D00 MOV EAX,EAX
004323AE . 8BC0           MOV EAX,EAX
004323B0 $- FF25 48694D00 MOV EAX,EAX
004323B6 . 8BC0           MOV EAX,EAX
004323B8 $- FF25 44694D00 MOV EAX,EAX
004323BE . 8BC0           MOV EAX,EAX
004323C0 $- FF25 40694D00 MOV EAX,EAX
004323C6 . 8BC0           MOV EAX,EAX
004323C8 . 55             PUSH EBP
004323C9 . 8BEC           MOV EBP,ESP
004323CA . 33F0           XOR EAX,EAX

```

Follow call them dump it in, 1 manipulate the very often, yes. I do all bromide hem ah, hix hix.:

Address	Value	Comment
004D6260	7C801D4F	kernel32.LoadLibraryExA
004D6264	7C80A415	kernel32.GetThreadLocale
004D6268	7C801EEE	kernel32.GetStartupInfoA
004D626C	7C80ADA0	kernel32.GetProcAddress
004D6270	7C80B6A1	kernel32.GetModuleHandleA
004D6274	7C80B4CF	kernel32.GetModuleFileNameA
004D6278	7C80D262	kernel32.GetLocaleInfoA
004D627C	7C910331	ntdll.RtlGetLastWin32Error
004D6280	7C812F1D	kernel32.GetCommandLineA
004D6284	7C80ABDE	kernel32.FreeLibrary
004D6288	7C8137D9	kernel32.FindFirstFileA
004D628C	7C80EDD7	kernel32.FindClose
004D6290	7C81CDDA	kernel32.ExitProcess
004D6294	7C810D87	kernel32.WriteFile
004D6298	7C862E62	kernel32.UnhandledExceptionFilter
004D629C	7C810B8E	kernel32.SetFilePointer
004D62A0	7C832044	kernel32.SetEndOfFile
004D62A4	7C937A40	ntdll.RtlUnwind
004D62A8	7C80180E	kernel32.ReadFile
004D62AC	7C812A09	kernel32.RaiseException
004D62B0	7C812F39	kernel32.GetStdHandle
004D62B4	7C810A77	kernel32.GetFileSize
004D62B8	7C810E51	kernel32.GetFileType
004D62BC	7C801A24	kernel32.CreateFileA
004D62C0	7C809B47	kernel32.CloseHandle
004D62C4	00000000	
004D62C8	77D611B3	user32.GetKeyboardType
004D62CC	77D50FE8	user32.LoadStringA
004D62D0	77D804EA	user32.MessageBoxA
004D62D4	77D50F90	user32.CharNextA
004D62D8	00000000	
004D62DC	77DD7883	advapi32.RegQueryValueExA
004D62E0	77DD761B	advapi32.RegOpenKeyExA
004D62E4	77DD761B	advapi32.RegOpenKeyExA

I pull up, pull down, balls out of it enough way, to find hem START - END do anything that monkey Asia, they want it koai dom beautiful hem phone. Furcate they add it to a conclusion: COC has canceled IAT (IAT or hide, or what the sugarcane for the camera to the coin, other other) The children also do chả dump read, edit them into EP OEP they found, save the file in the Editor is PE. They run the packages, it automatically close now, they debug it reported spills Stack camera ạ. Tran hem I say

stack the pieces to xài again. My goodness, they love to hem execryptor is Đóa, that it can pack hem of the CODE section.

So they must think the other, they remember when you unpack what it should immediately Break OEP new dump, fix the IAT. Make sure they have done, and done wrong books too new to lose the pain. Error arising difficult to control. Ui that ới Choi, children have to debug the OEP nôi, he said then, as he said execryptor by Olly that, subject to how it is to lose Olly. What he can not believe the PA tut unpack with Execryptor still there, some Western, he has not written all the cases protect the execryptor do tut dropped by more Đóa is to target some of the camera phone.

The children go to Vmware, have children in child nọc pork SoftIce in this, it is healthy or not they know hem but they heard SoftIce he is the father of the same with the other, he believes the Execryptor. Execryptor that he is the only so Olly and the SoftIce sure the statistics in this. I know the OEP so they just break in to SoftIce OEP is completed. But ... the complete break. Moa ới, hem that they dump in SoftIce, xài IceExt as \$ hit, it saved the file in 1000 Byte is also 1DA000 Size Image of this file it under hem. They only lose, sometimes NAG trial that does it in a garbage dump full of memory. So is she really tit ới. I think their clothing is true because what the hem bik, also dump the hem bik they also stand for something.

Uiiiiiii ... yes you, now they hum uiiiiiii bit more. At first they loe it out EP FE. 2 Byte khốn what should each of the Armadillo blankets ducks. Bik children than it is anything more but I see now that it debug it just jump right it. Do not know any statistics khùng out the orders of fiddling, but now they bik hem it is they have lost. What he has in mind is we opened immediately in OEP Olly without debug spent, but break at OEP in Olly the right is wá the imagination. However, we can he run the patch. The children return to OEP, they patch it:

Address	Hex	dump	Disassembly
004CFA6D	00		DB 00
004CFA6E	00		DB 00
004CFA6F	00		DB 00
004CFA70	14F74C00		DD MIDItoMP.004CF714
004CFA74	55		PUSH EBP
004CFA75	8BEC		MOV EBP,ESP
004CFA77	83C4 F0		ADD ESP,-10
004CFA7A	53		PUSH EBX
004CFA7B	B8 3C		
004CFA80	E8 A3		
004CFA85	8B1D		
004CFA88	8B03		
004CFA8D	E8 3E		
004CFA92	8B03		
004CFA94	BA 6C		
004CFA99	E8 1A		
004CFA9E	8B0D		
004CFAA4	8B03		
004CFAA6	8B15		
004CFAAC	E8 37		
004CFAB1	8B0D		
004CFAB7	8B03		
004CFAB9	8B15		
004CFABF	E8 24		
004CFAC4	8B0D		

Address	Hex	dump	Disassembly
004CFA6D	00		DB 00
004CFA6E	00		DB 00
004CFA6F	00		DB 00
004CFA70	14F74C00		DD MIDItoMP.004CF714
004CFA74	55		PUSH EBP
004CFA75	8BEC		MOV EBP,ESP
004CFA77	83C4 F0		ADD ESP,-10
004CFA7A	53		PUSH EBX
004CFA7B	B8 3C		
004CFA80	E8 A3		
004CFA85	8B1D		
004CFA88	8B03		
004CFA8D	E8 3E		
004CFA92	8B03		
004CFA94	BA 6C		
004CFA99	E8 1A		
004CFA9E	8B0D		
004CFAA4	8B03		
004CFAA6	8B15		
004CFAAC	E8 37		
004CFAB1	8B0D		
004CFAB7	8B03		
004CFAB9	8B15		
004CFABF	E8 24		
004CFAC4	8B0D		

Address	Hex	dump	Disassembly
004CFA6D	00		DB 00
004CFA6E	00		DB 00
004CFA6F	00		DB 00
004CFA70	14F74C00		DD MIDItoMP.004CF714
004CFA74	EB FE		JMP SHORT MIDItoMP.004CFA74
004CFA76	EC		IN AL,DX
004CFA77	83C4 F0		ADD ESP,-10
004CFA7A	53		PUSH EBX
004CFA7B	B8 3CF74C00		MOV EAX,MIDItoMP.004CF73C
004CFA80	E8 A373F3FF		CALL MIDItoMP.00406E28
004CFA85	8B1D		

Call to save the camera, we should save what debug the hem also fear what where. Cameras bik they have just done what the hem Đóa What? Ah they just think that when it sop run hem Olly have it running mượt, who hem detect all, both the dump. Other other, so she repeats it at the OEP run, this means it is the break in when you debug OEP. So Đóa ơi camera, you do not wan the form is based in New Olly kill some of it he crack? byte patch Winhex this is also the right man said. That phone, they demonstrate what he saw for home, children run time just patch file:

name.txt	150 KB
midihelp.chm	83 KB
MIDItoMP3	1 KB
MIDItoMP3.exe	1,842 KB
oggenc.exe	152 KB
Readme.txt	6 KB
unins000.dat	6 KB
unins000.exe	658 KB
MIDItoMP3_patch.exe	1,842 KB

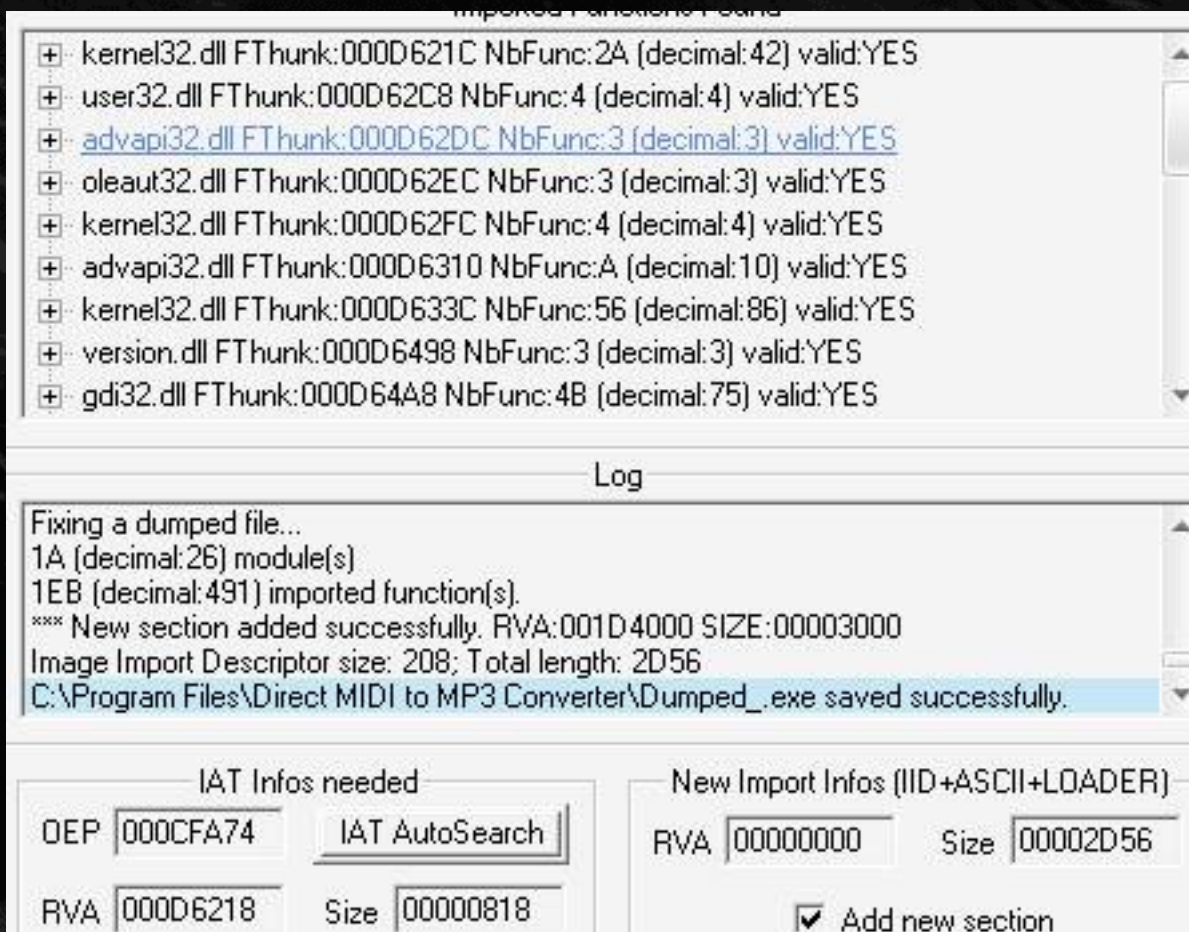
What he found more, and then run hok see what happens with all, but in memory still have it:

WINWORD.EXE	1812		Microsoft Office Word	Microsoft C
MIDItoMP3_patch.exe	2280	50.76	Direct MIDI to MP3 Converter	Piston Soft
Snagit32.exe	3780		Snagit 8	TechSmith
TscHelp.exe	2396		TechSmith HTML Help Helper	TechSmith

Hi hi, it is the last in OEP Đóa camera. Now they qualify dump books right then, what he means to use the dump for any use, not because xài LordPE detect it. But at the OEP Execryptor while it created some thread to detect tum lum should carefully home Camera:

c:\program files\snagit 8\tscHELP.exe	
c:\program files\direct midi to mp3 converter\miditomp3_patch.exe	Dump Full...
d:\conserve\programs\cracktools\petools 1.5 rc7\petools.exe	Dump Partial...

IAT fix his call, he filled some of the OEP IAT Autostart - Get Import lickerish:



Fix finished, kill the process and repeats endlessly go there. Cameras drilling fix dump file to run for this month OEP also repeats, we load it on to Olly:

Address	Hex dump	Disassembly
004CFA74	EB FE	JMP SHORT Dumped_.<ModuleEntryPoint>
004CFA76	EC	IN AL,DX
004CFA77	83C4 F0	ADD ESP,-10
004CFA7A	53	PUSH EBX
004CFA7B	B8 3CF74C00	MOV EAX,Dumped_.004CF73C
004CFA80	E8 A373F3FF	CALL Dumped_.00406E28
004CFA85	8B1D 581E4D00	MOV EBX,DWORD PTR DS:[4D1E58]
004CFA88	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFA8D	E8 3E6DFBFF	CALL Dumped_.004867D0
004CFA92	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFA94	BA 6CFB4C00	MOV EDI,Dumped_.004CFB6C
004CFA99	E8 1A69FBFF	CALL Dumped_.004863B8
004CFA9E	8B0D A01B4D00	MOV ECX,DWORD PTR DS:[4D1BA0]
004CFA04	0000	MOV EAX,DWORD PTR DS:[EBX]

Minh OEP is correct is to EB FE 55 8B:

Address	Hex dump	Disassembly
004CFA74	55	PUSH EBP
004CFA75	8BEC	MOV EBP,ESP
004CFA77	83C4 F0	ADD ESP,-10
004CFA7A	53	PUSH EBX
004CFA7B	B8 3CF74C00	MOV EAX,Dumped_.004CF73C
004CFA80	E8 A373F3FF	CALL Dumped_.00406E28
004CFA85	8B1D 581E4D00	MOV EBX,DWORD PTR DS:[4D1E58]
004CFA88	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFA8D	E8 3E6DFBFF	CALL Dumped_.004867D0
004CFA92	8B03	MOV EAX,DWORD PTR DS:[EBX]
004CFA94	BA 6CFB4C00	MOV EDI,Dumped_.004CFB6C
004CFA99	E8 1A69FBFF	CALL Dumped_.004863B8

Save again. Call our brothers run when:

MIDItoMP3.exe	1,842 KB
oggenc.exe	152 KB
Readme.txt	6 KB
unins000.dat	6 KB
unins000.exe	658 KB
MIDItoMP3_patch.exe	1,842 KB
Dumped.exe	1,872 KB
Dumped_.exe	1,884 KB
Dumped_fix.exe	1,884 KB



Ah not unpack expired where home is, in the child before it expired thui.
Hi hi.May he used every Trial Reset is lickerish work. Services NAG the U.
S. they lose. But suppose we close NAG, or even to run in, we also Exit is
what happens:



Experience hem many children, but they again fossick in memory of corn cui children, why he has referred to CRC Check. I think this phone is it. I load the file to fix indefinite Olly. She pulled up the top truoc section code:

Address	Hex dump	Disassembly
00401000	04 10	ADD AL,10
00401002	40	INC EAX
00401003	0003	ADD BYTE PTR DS:[EBX],AL
00401005	07	POP ES
00401006	42	INC EDX
00401007	6F	OUTS DX,DWORD PTR ES:[EDI]
00401008	AF	OUTS AX,DWORD PTR ES:[EDI]

They call search form Byte later, this form automatically draws them all, his super-wan debug Why We, the children hok time:



How fortunate result is unique, if not only for the brothers when the search function in this form is Okie:


```

004A43C1  C3                RETN
004A43C2  8BC0             MOV EAX,EAX
004A43C4  832D 943C4D00 01 SUB DWORD PTR DS:[4D3C94],1
004A43CB  73 1A            JNB SHORT Dumped_f.004A43E7
004A43CD  E9 147B0E00      JMP Dumped_f.0058BEE6
004A43D2  68 C76D22A3      PUSH A3226DC7
004A43D7  5A              POP EDX
004A43D8  E9 D8D10100      JMP Dumped_f.004C15B5
004A43DD  04 2D            ADD AL,2D
004A43DF  02BD 27488D91    ADD BH,BYTE PTR SS:[EBP+918D4827]
004A43E5  863F             XCHG BYTE PTR DS:[EDI],BH
004A43E7  C3                RETN
004A43FA  EC              PUSH FRP

```

It CRC check this place. They not only play raw bạo RETN as he Why, they change only 1 Byte:

```

004A43C1  C3                RETN
004A43C2  8BC0             MOV EAX,EAX
004A43C4  832D 943C4D00 01 SUB DWORD PTR DS:[4D3C94],1
004A43CB  EB 1A            JNB SHORT Dumped_f.004A43E7
004A43CD  E9 147B0E00      JMP Dumped_f.0058BEE6
004A43D2  68 C76D22A3      PUSH A3226DC7
004A43D7  5A              POP EDX
004A43D8  E9 D8D10100      JMP Dumped_f.004C15B5
004A43DD  04 2D            ADD AL,2D
004A43DF  02BD 27488D91    ADD BH,BYTE PTR SS:[EBP+918D4827]
004A43E5  863F             XCHG BYTE PTR DS:[EDI],BH
004A43E7  C3                RETN

```

British neo color blind hem dom that subject. Call to save what he oiiiiiii ...

MIDItoMP3.exe	1,842 KB
oggenc.exe	152 KB
Readme.txt	6 KB
unins000.dat	6 KB
unins000.exe	658 KB
MIDItoMP3_patch.exe	1,842 KB
Dumped.exe	1,872 KB
Dumped_.exe	1,884 KB
Dumped_fix.exe	1,884 KB
Dumped_fix2.exe	1,884 KB

Running error that is more sure he is in the variables in oi hem but they must be at home. Khua khua

IV-story Search:

So after they finished laughing about the story that Execryptor oi. What does he kindly please make keygen or patch NAG also help children. If Patch NAG done he always patch the file but not the original patch file dump file as the original home is still the original code that. But keygen or patch that he go to some junk code that the hull suffered, innocent children .. amennnn ...

This child kì hem coa mentioned as a file server run on any machine, it must be more action as mapped overwrite Base Image of a Window module manually, edit the section of the Import Execryptor or disable

completely TLS . Ah, what does he have Khoai TSL clear as they aged CRACKLATINOS the parties do not sop to the dentist. As in the wá Run, the section of execryptor still check the byte by TSL in PE Header, which it has decided to remove some areas in mind or not. 1 that the area be sure to remove only a sop it tit Ngoi immediately.

They offer only the English, this child is not her responses they closed the temple, else they will burn the temple down the mountain. Hix hix. Happy Valentine a camera is not the same child, a happy New Year, a new year is plentiful health, of money as water streams (but hem to end the country), he wrote by inspiration, the heart that this should be called Cuc scream but is tut tit anything. hi hi ...

Big thanks to:

All REA's members: Computer_Angel, Moonbaby, Zombie, hacnho, benina, kienmanowar, rongchaua, Deux, Merc, hoadongnoi, the_lighthouse, TQN, light.phoenix, hytkl, tlandn, hurt_heart, dzungltn, Zoi, littleboy84, haule_nth, takada, Why not bar, iamidiot, Akira, dump, thienthandien, [kid], ...

Special thanks to:

fly, stephenteh, Gabri3l, MaDMAN_H3rCuL3s, CondZero, Ricardo Narvaja + NCR, lena151, haggar, ARTeam, snd, RES, CrackLatinos, all unpack. cn ... Authors who created tools and you.



Trick Xi News - 2007

Armadillo collect sand-stone

Sothink swf Decompiler <|> ARM 4.xx - Standard Protection + Cracking



_Load Target:

Address	Hex dump	Disassembly	Comment
005C4CB3	55	PUSH EBP	
005C4CB4	8BEC	MOV EBP,ESP	
005C4CB6	6A FF	PUSH -1	
005C4CB8	68 F8EE5E00	PUSH SWFDecom.005EEEF8	
005C4CB0	68 F0495C00	PUSH SWFDecom.005C49F0	SE handler installation
005C4CC2	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
005C4CC8	50	PUSH EAX	
005C4CC9	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
005C4CD0	83EC 58	SUB ESP,58	
005C4CD3	53	PUSH EBX	
005C4CD4	56	PUSH ESI	
005C4CD5	57	PUSH EDI	
005C4CD6	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
005C4CD9	FF15 88915E00	CALL DWORD PTR DS:[<&KERNEL32.GetVersion	kernel32.GetVersion
005C4CDF	33D2	XOR EDX,EDX	
005C4CE1	8AD4	MOV DL,AH	
005C4CE3	8915 84055F00	MOV DWORD PTR DS:[5F0584],EDX	
005C4CE9	8BC8	MOV ECX,EAX	
005C4CEB	81E1 FF000000	AND ECX,0FF	
005C4CF1	890D 80055F00	MOV DWORD PTR DS:[5F0580],ECX	
005C4CF7	C1E1 08	SHL ECX,8	
005C4CFA	03CA	ADD ECX,EDX	
005C4CFC	890D 7C055F00	MOV DWORD PTR DS:[5F057C],ECX	
005C4D02	C1E8 10	SHR EAX,10	
005C4D05	A3 78055F00	MOV DWORD PTR DS:[5F0578],EAX	
005C4D0A	33F6	XOR ESI,ESI	
005C4D0C	56	PUSH ESI	
005C4D0D	E8 78160000	CALL SWFDecom.005C638A	
005C4D12	59	POP ECX	
005C4D13	85C0	TEST EAX,EAX	
005C4D15	75 08	JNZ SHORT SWFDecom.005C4D1F	
005C4D17	6A 1C	PUSH 1C	
005C4D19	E8 B0000000	CALL SWFDecom.005C4DCE	
005C4D1E	59	POP ECX	

_Chay Script:

```
var GetModuleHandleA
```

```
var AddressOfMagicJump
var LenOfMagicJump
```

```
GPA "GetModuleHandleA", "kernel32.dll"
mov GetModuleHandleA, $ RESULT
```

```
bphws GetModuleHandleA, "x"
repeat:
esto
rtu
find eip, # 0F84 ?????????????????? 74 ?????????? EB? #
Cmp $ result, 0
je repeat
bphwc GetModuleHandleA
```

```
mov AddressOfMagicJump, $ RESULT
mov LenOfMagicJump, AddressOfMagicJump
add LenOfMagicJump, 2
mov LenOfMagicJump, [LenOfMagicJump]
inc LenOfMagicJump
mov [AddressOfMagicJump], 0E9
inc AddressOfMagicJump
mov [AddressOfMagicJump], LenOfMagicJump
CMT $ result, "<- MagicJump fixed"
msg "MagicJump fixed! Now, the next step is yours: OEP Finder!"
ret
```

_Sau When to run here:

Address	Hex dump	Disassembly	Comment
003C4C23	8B0D 948D3E00	MOV ECX,DWORD PTR DS:[3E8D94]	
003C4C29	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003C4C2C	A1 948D3E00	MOV EAX,DWORD PTR DS:[3E8D94]	
003C4C31	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
003C4C34	✓75 16	JNZ SHORT 003C4C4C	
003C4C36	8D85 DCFEFFFF	LEA EAX,DWORD PTR SS:[EBP-124]	
003C4C3C	50	PUSH EAX	
003C4C3D	FF15 90B03D00	CALL DWORD PTR DS:[3DB090]	kernel32.LoadLibraryA
003C4C43	8B0D 948D3E00	MOV ECX,DWORD PTR DS:[3E8D94]	
003C4C49	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003C4C4C	A1 948D3E00	MOV EAX,DWORD PTR DS:[3E8D94]	
003C4C51	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
003C4C54	✓E9 30010000	JMP 003C4D89	<- MagicJump fixed
003C4C59	0033	ADD BYTE PTR DS:[EBX],DH	
003C4C5B	C9	LEAVE	
003C4C5C	8B03	MOV EAX,DWORD PTR DS:[EBX]	
003C4C5E	393D	CMP DWORD PTR DS:[ESI],EDI	
003C4C60	✓74	JZ SHORT 003C4C62	
003C4C62	41	INC ECX	
003C4C63	83	INC ECX	
003C4C66	^EB	MOV EAX,DWORD PTR DS:[EBI]	
003C4C68	8B	MOV EAX,DWORD PTR DS:[EBI]	
003C4C6A	C1	MOV EAX,DWORD PTR DS:[EBI]	
003C4C6D	57	JNB SHORT 003C4C73	
003C4C6E	E8	CALL DWORD PTR DS:[EBI]	msvcrt.??2@YAPAXI0Z
003C4C73	8B	MOV EAX,DWORD PTR DS:[EBI]	
003C4C79	89	MOV EAX,DWORD PTR DS:[EBI]	
003C4C7C	57	JNB SHORT 003C4C82	
003C4C7D	E8	CALL DWORD PTR DS:[EBI]	msvcrt.??2@YAPAXI0Z
003C4C82	59	POP EAX	
003C4C83	59	POP EAX	
003C4C84	8B0D 908D3E00	MOV ECX,DWORD PTR DS:[3E8D90]	
003C4C8A	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
003C4C8D	8B03	MOV EAX,DWORD PTR DS:[EBX]	
003C4C8F	8B03	MOV EAX,DWORD PTR DS:[EBX]	

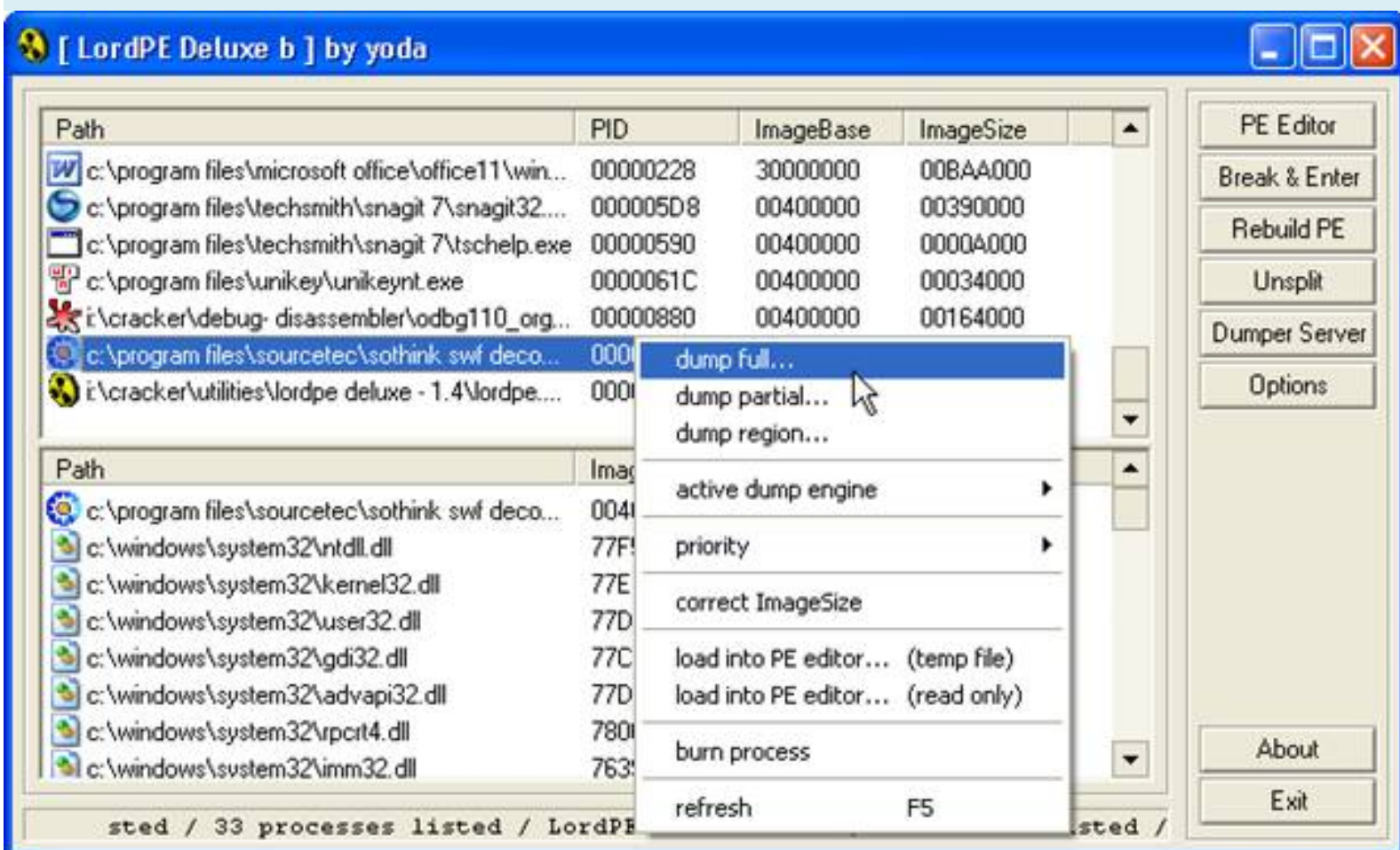
_Bp CreateThread, Ctrl + F9, F8, Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
003D7522	59	POP ECX	kernel32.77E7BE2B
003D7523	BF C0483E00	MOV EDI,3E48C0	
003D7528	8BCF	MOV ECX,EDI	
003D752A	E8 1A0DFEFF	CALL 003B8249	
003D752F	84C0	TEST AL,AL	
003D7531	75 09	JNZ SHORT 003D753C	
003D7533	6A 01	PUSH 1	
003D7535	8BCF	MOV ECX,EDI	
003D7537	E8 9058FEFF	CALL 003BCDCC	
003D753C	B9 D8063F00	MOV ECX,3F06D8	
003D7541	C705 70103E00 3	MOV DWORD PTR DS:[3E1070],3E1F30	
003D7548	E8 A3F2FFFF	CALL 003D67F3	
003D7550	6A 00	PUSH 0	
003D7552	E8 9CF2FFFF	CALL 003D67F3	
003D7557	A1 084F3E00	MOV EAX,DWORD PTR DS:[3E4F08]	
003D755C	59	POP ECX	
003D755D	8B16	MOV EDX,DWORD PTR DS:[ESI]	
003D755F	8B48 78	MOV ECX,DWORD PTR DS:[EAX+78]	
003D7562	3348 44	XOR ECX,DWORD PTR DS:[EAX+44]	
003D7565	3348 04	XOR ECX,DWORD PTR DS:[EAX+4]	
003D7568	030D 204F3E00	ADD ECX,DWORD PTR DS:[3E4F20]	SWFDecom.00400000
003D756E	85D2	TEST EDX,EDX	
003D7570	75 1B	JNZ SHORT 003D758D	
003D7572	8B90 88000000	MOV EDX,DWORD PTR DS:[EAX+88]	
003D7578	FF76 18	PUSH DWORD PTR DS:[ESI+18]	
003D757B	3350 78	XOR EDX,DWORD PTR DS:[EAX+78]	
003D757E	FF76 14	PUSH DWORD PTR DS:[ESI+14]	
003D7581	3350 4C	XOR EDX,DWORD PTR DS:[EAX+4C]	
003D7584	FF76 10	PUSH DWORD PTR DS:[ESI+10]	
003D7587	2BCA	SUB ECX,EDX	
003D7589	FFD1	CALL ECX	
003D758B	EB 20	JMP SHORT 003D75AD	
003D758D	83FA 01	CMP EDX,1	
003D7590	75 1D	JNZ SHORT 003D75AF	
003D7592	FF76 04	PUSH DWORD PTR DS:[ESI+4]	
003D7595	8B90 88000000	MOV EDX,DWORD PTR DS:[EAX+88]	
003D7598	3350 78	XOR EDX,DWORD PTR DS:[EAX+78]	
003D759E	FF76 08	PUSH DWORD PTR DS:[ESI+8]	
003D75A1	3350 4C	XOR EDX,DWORD PTR DS:[EAX+4C]	
003D75A4	6A 00	PUSH 0	
003D75A6	FF76 0C	PUSH DWORD PTR DS:[ESI+C]	
003D75A9	2BCA	SUB ECX,EDX	
003D75AB	FFD1	CALL ECX	
003D75AD	8B08	MOV EBX,EAX	

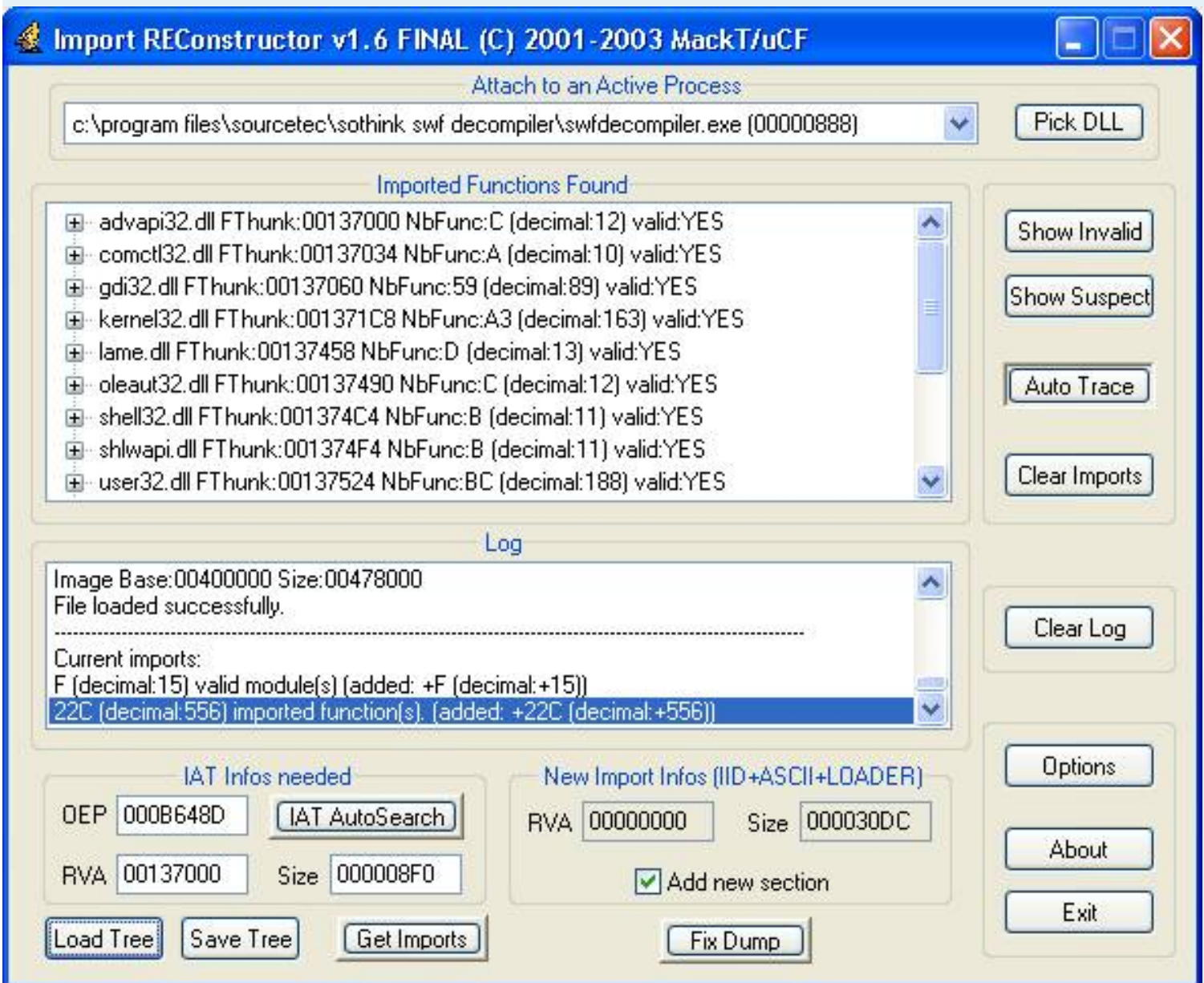
_F9, F7: OEP!

Address	Hex dump	Disassembly	Comment
004B648D	6A 60	PUSH 60	
004B648F	68 00495400	PUSH SWFDecom.00544900	
004B6494	E8 D7200000	CALL SWFDecom.004B8570	
004B6499	BF 94000000	MOV EDI,94	
004B649E	8BC7	MOV EAX,EDI	
004B64A0	E8 6B080000	CALL SWFDecom.004B6D10	
004B64A5	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
004B64A8	8BF4	MOV ESI,ESP	
004B64AA	893E	MOV DWORD PTR DS:[ESI],EDI	
004B64AC	56	PUSH ESI	
004B64AD	FF15 38745300	CALL DWORD PTR DS:[537438]	kernel32.GetVersionExA
004B64B3	8B4E 10	MOV ECX,DWORD PTR DS:[ESI+10]	
004B64B6	890D 645B5800	MOV DWORD PTR DS:[585B64],ECX	
004B64BC	8B46 04	MOV EAX,DWORD PTR DS:[ESI+4]	
004B64BF	A3 705B5800	MOV DWORD PTR DS:[585B70],EAX	
004B64C4	8B56 08	MOV EDX,DWORD PTR DS:[ESI+8]	
004B64C7	8915 745B5800	MOV DWORD PTR DS:[585B74],EDX	
004B64CD	8B76 0C	MOV ESI,DWORD PTR DS:[ESI+C]	
004B64D0	81E6 FF7F0000	AND ESI,7FFF	
004B64D6	8935 685B5800	MOV DWORD PTR DS:[585B68],ESI	
004B64DC	83F9 02	CMP ECX,2	
004B64DF	74 0C	JE SHORT SWFDecom.004B64E7	
004B64E1	81CE 00000000	OR ESI,0000	
004B64E7	8935 685B5800	MOV DWORD PTR DS:[585B68],ESI	
004B64ED	C1E0 08	SHL EAX,8	
004B64F0	03C2	ADD EAX,EDX	
004B64F2	A3 6C5B5800	MOV DWORD PTR DS:[585B6C],EAX	
004B64F7	33F6	XOR ESI,ESI	
004B64F9	56	PUSH ESI	
004B64FD	8B3D 54325300	MOV EDI,DWORD PTR DS:[533254]	kernel32.GetModuleHandleA

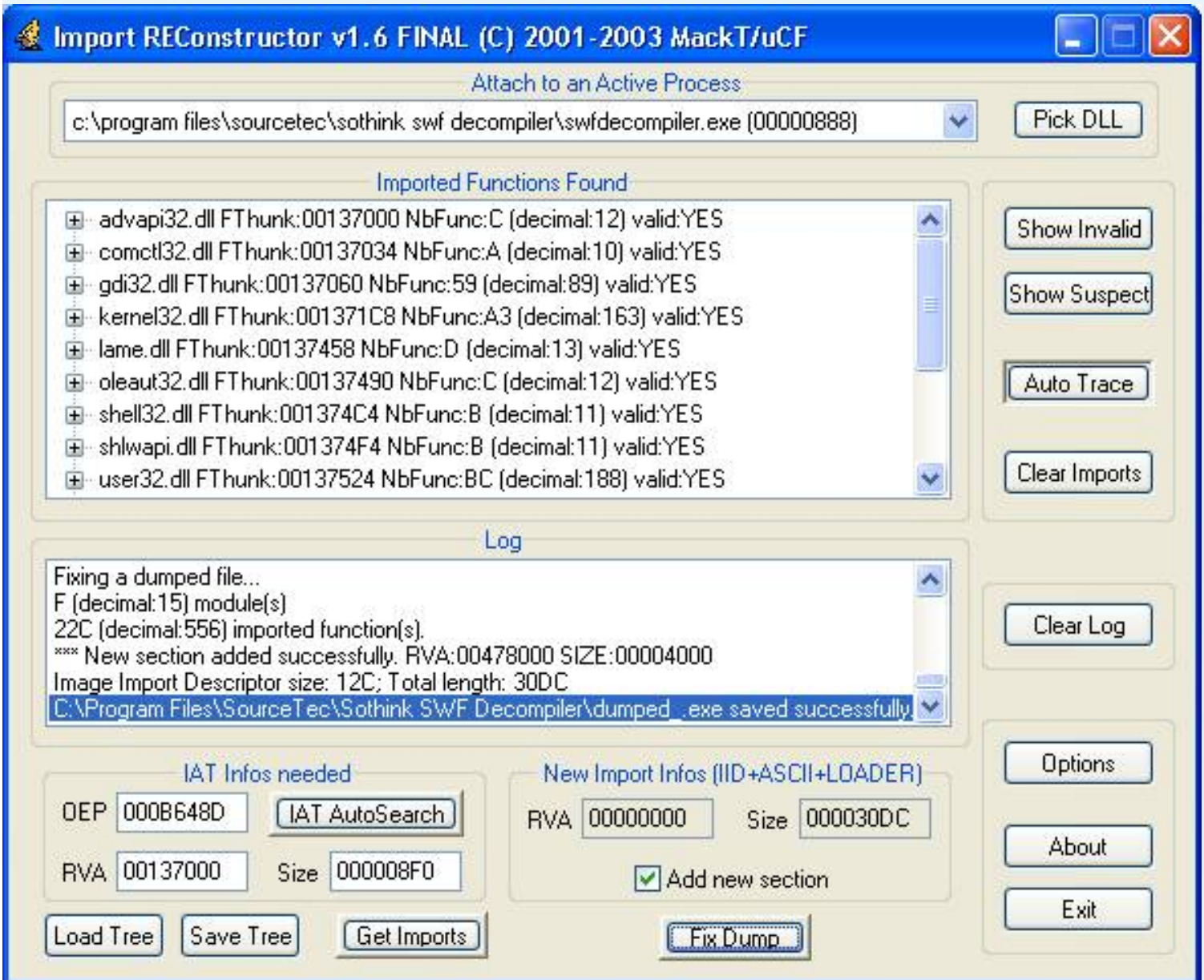
_LordPE DumpFull:



_ImpREC:



_FixDump:



_Optimize:

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address
Byte[8]	Dword	Dword	Dword	Dword	Dword
.text	00135A76	00001000	00135A76	00001000	00000000
.rdata	0003E69E	00137000	0003E69E	00137000	00000000
.data	000128B4	00176000	000128B4	00176000	00000000
text1	00050000	00189000	00050000	00189000	00000000
.adata	00010000	001D9000	00010000	001D9000	00000000
.data1	00020000	001E9000	00020000	001E9000	00000000
pdata	00210000	00209000	00210000	00209000	00000000
.rsrc	0005F000	00419000	0005F000	00419000	00000000
.mackt	00004000	00478000	00004000	00478000	00000000

Change Section Flags

Add Section (Header Only)

Add Section (Empty Space)

Add Section (File Data)

Delete Section (Header Only)

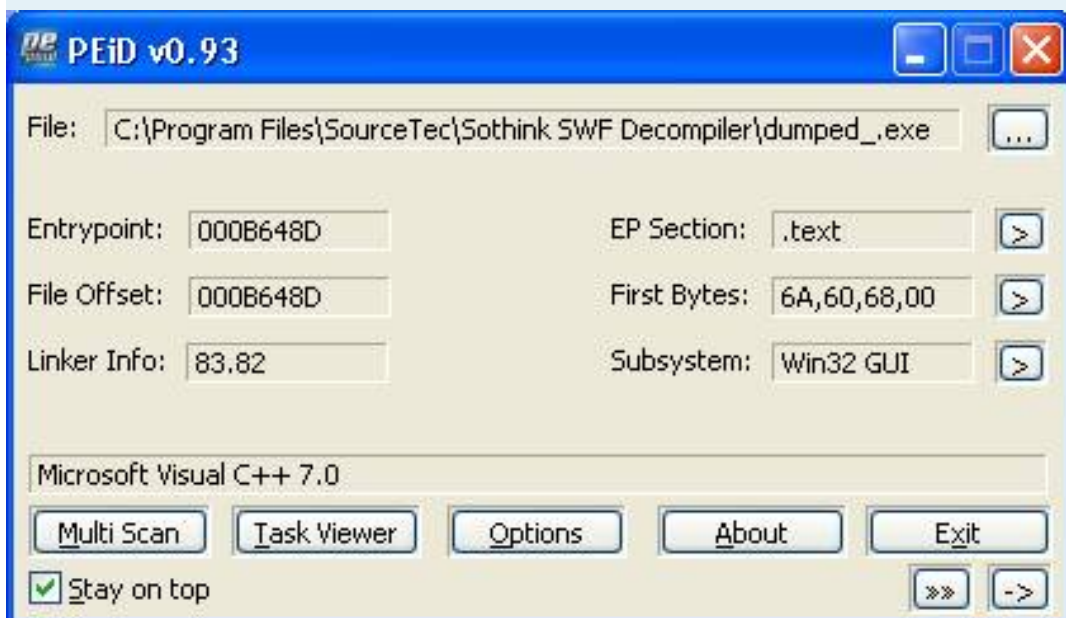
Delete Section (Header And Data)

Rebuild Image Size

Rebuild PE Header

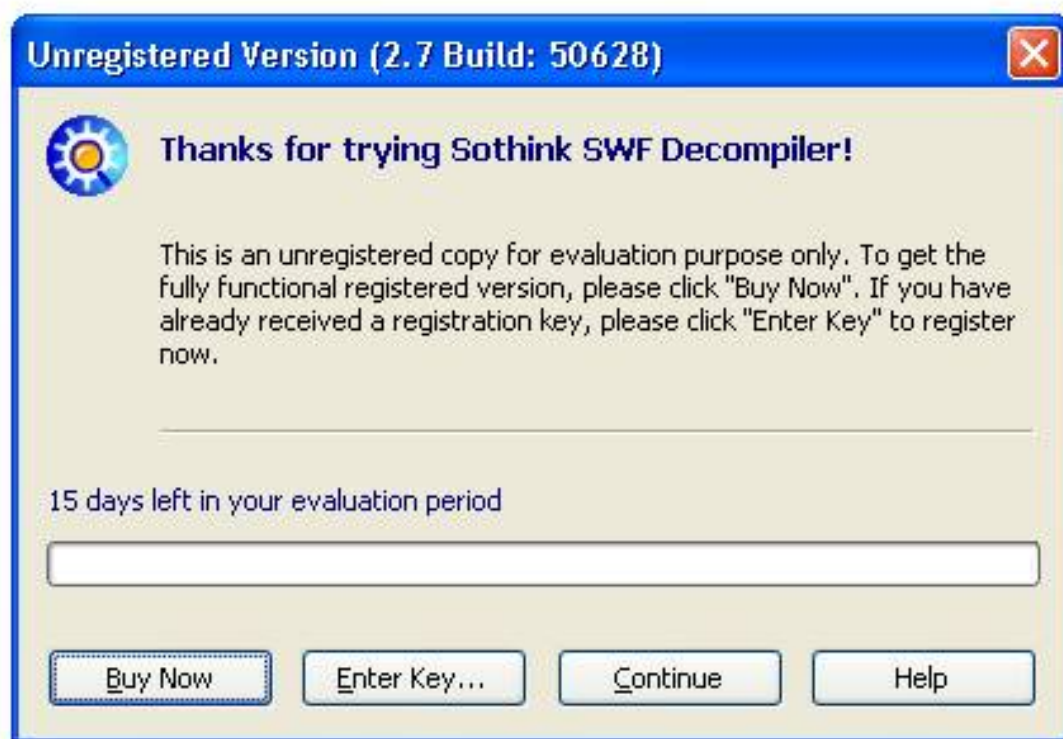
Dump Section

[_Detect:](#)

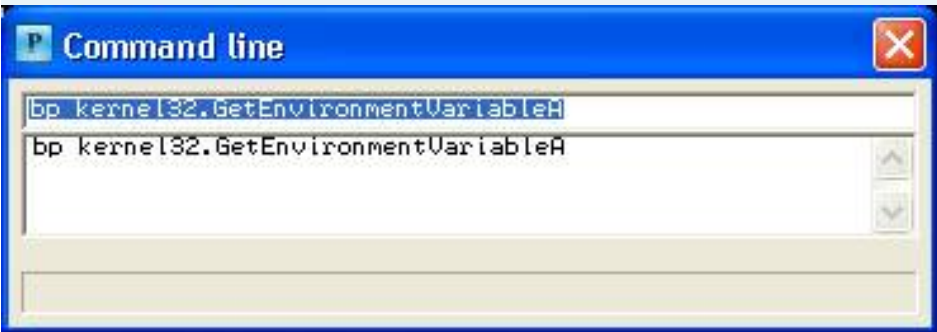


_Run Try:

- cracked.exe
- dumped.exe
- dumped_.exe
- Lame.dll
- LGPL.txt
- License.txt
- ReadMe.txt
- SWFDecompiler
- SWFDecompiler.chm
- SWFDecompiler.exe
- SWFDecompiler_.exe
- tree.txt
- unins000.dat
- unins000.exe



Cracking: Load up dumped.exe OllyDBG: Alt + F1 set breakpoint:



_F9:

Address	Hex dump	Disassembly	Comment
77E7D173	55	PUSH EBP	
77E7D174	8BEC	MOV EBP,ESP	
77E7D176	83EC 20	SUB ESP,20	
77E7D179	53	PUSH EBX	
77E7D17A	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
77E7D17D	8D45 E0	LEA EAX,DWORD PTR SS:[EBP-20]	
77E7D180	50	PUSH EAX	
77E7D181	FF15 8012E677	CALL DWORD PTR DS:[<&ntdll.RtlInitString	ntdll.RtlInitString
77E7D187	6A 01	PUSH 1	
77E7D189	8D45 E0	LEA EAX,DWORD PTR SS:[EBP-20]	
77E7D18C	50	PUSH EAX	
77E7D18D	8D45 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
77E7D190	50	PUSH EAX	
77E7D191	FF15 8010E677	CALL DWORD PTR DS:[<&ntdll.RtlAnsiString	ntdll.RtlAnsiStringToUnicodeString
77E7D197	33DB	XOR EBX,EBX	
77E7D199	3BC3	CMP EAX,EBX	
77E7D19B	0F8C FF180200	JL kernel32.77E9EAA0	
77E7D1A1	56	PUSH ESI	
77E7D1A2	8B75 10	MOV ESI,DWORD PTR SS:[EBP+10]	
77E7D1A5	B8 FD7F0000	MOV EAX,7FFD	
77E7D1AA	3BF0	CMP ESI,EAX	
77E7D1AC	57	PUSH EDI	
77E7D1AD	0F87 B1450000	JA kernel32.77E81764	
77E7D1B3	3BF3	CMP ESI,EBX	
77E7D1B5	8BC6	MOV EAX,ESI	
77E7D1B7	0F85 A7450000	JNZ kernel32.77E81764	
77E7D1BD	33C0	XOR EAX,EAX	
77E7D1BF	03C0	ADD EAX,EAX	
77E7D1C1	66:8945 FA	MOV WORD PTR SS:[EBP-6],AX	
77E7D1C5	64:01 12000000	MOV EAX,DWORD PTR SS:[EBP-4]	

Address	Value	Comment
0012FA40	004128AE	CALL to GetEnvironmentVariableA from dumped_004128AC
0012FA44	00555598	VarName = "KEYCREATED"
0012FA48	0012FA54	Buffer = 0012FA54
0012FA4C	00000104	BufSize = 104 (260.)
0012FA50	005872B8	dumped_.005872B8
0012FA54	0017DE58	ASCII "....."
0012FA58	7FFDE000	
0012FA5C	004E0045	dumped_.004E0045
0012FA60	00000055	
0012FA64	00150000	
0012FA68	7FFDE000	
0012FA6C	00020852	UNICODE "\\dumped_.exe"
0012FA70	7FFDE000	

_Kha Kha, see it as something that we createkey up J , Ctrl + F9

Address	Hex dump	Disassembly	Comment
77E7D255	C2 0C00	RETN 0C	
77E7D258	55	PUSH EBP	
77E7D259	8BEC	MOV EBP,ESP	
77E7D25B	56	PUSH ESI	
77E7D25C	8B35 B012E677	MOV ESI,DWORD PTR DS:[&ntdll.NtProtect	ntdll.ZwProtectVirtualMemory
77E7D262	57	PUSH EDI	
77E7D263	FF75 18	PUSH DWORD PTR SS:[EBP+18]	
77E7D266	8D45 10	LEA EAX,DWORD PTR SS:[EBP+10]	
77E7D269	FF75 14	PUSH DWORD PTR SS:[EBP+14]	
77E7D26C	50	PUSH EAX	
77E7D26D	8D45 0C	LEA EAX,DWORD PTR SS:[EBP+C]	
77E7D270	50	PUSH EAX	
77E7D271	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
77E7D274	FFD6	CALL ESI	
77E7D276	8BF8	MOV EDI,EAX	
77E7D278	85FF	TEST EDI,EDI	
77E7D27A	^0F8C A182FEFF	JL kernel32.77E65521	
77E7D280	33C0	XOR EAX,EAX	
77E7D282	40	INC EAX	
77E7D283	5F	POP EDI	
77E7D284	5E	POP ESI	
77E7D285	5D	POP EBP	
77E7D286	C2 1400	RETN 14	
77E7D289	FF15 B810E677	CALL DWORD PTR DS:[&ntdll.RtlAcquirePe	ntdll.RtlAcquirePebLock
77E7D28F	3935 2470ED77	CMP DWORD PTR DS:[77ED7024],ESI	
77E7D295	√75 4B	JNZ SHORT kernel32.77E7D2E2	
77E7D297	A1 3C70ED77	MOV EAX,DWORD PTR DS:[77ED703C]	
77E7D29C	53	PUSH EBX	
77E7D29D	57	PUSH EDI	
77E7D29F	50 4B	PUSH 4B	

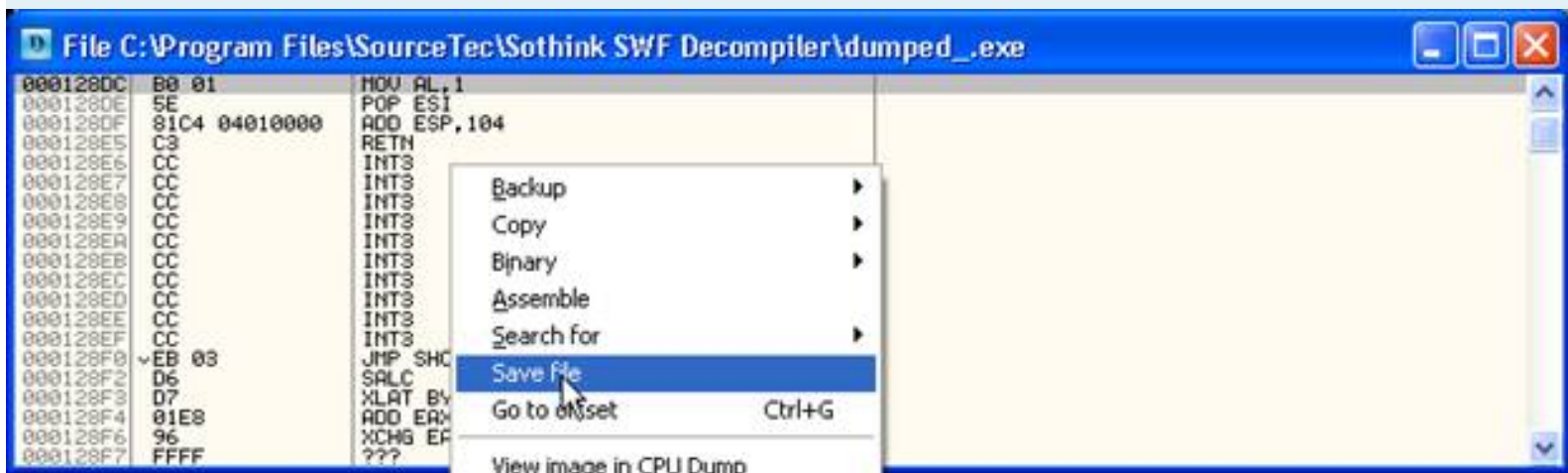
_F8:

Address	Hex dump	Disassembly	Comment
004128AE	85C0	TEST EAX,EAX	
004128B0	√76 2A	JBE SHORT dumped_.004128DC	
004128B2	68 04010000	PUSH 104	
004128B7	8D4C24 08	LEA ECX,DWORD PTR SS:[ESP+8]	
004128B8	51	PUSH ECX	
004128BC	68 90555500	PUSH dumped_.00555590	ASCII "KEYTYPE"
004128C1	FFD6	CALL ESI	
004128C3	85C0	TEST EAX,EAX	
004128C5	√76 15	JBE SHORT dumped_.004128DC	
004128C7	8D5424 04	LEA EDX,DWORD PTR SS:[ESP+4]	
004128CB	52	PUSH EDX	
004128CC	E8 8C1D0A00	CALL dumped_.004B465D	
004128D1	83C4 04	ADD ESP,4	
004128D4	5E	POP ESI	
004128D5	81C4 04010000	ADD ESP,104	
004128D8	C3	RETN	
004128DC	33C0	XOR EAX,EAX	
004128DE	5E	POP ESI	
004128DF	81C4 04010000	ADD ESP,104	
004128E5	C3	RETN	
004128E6	CC	INT3	
004128E7	CC	INT3	
004128E8	CC	INT3	
004128E9	CC	INT3	
004128EA	CC	INT3	
004128EB	CC	INT3	
004128EC	CC	INT3	
004128ED	CC	INT3	
004128EE	CC	INT3	
004128EF	CC	INT3	

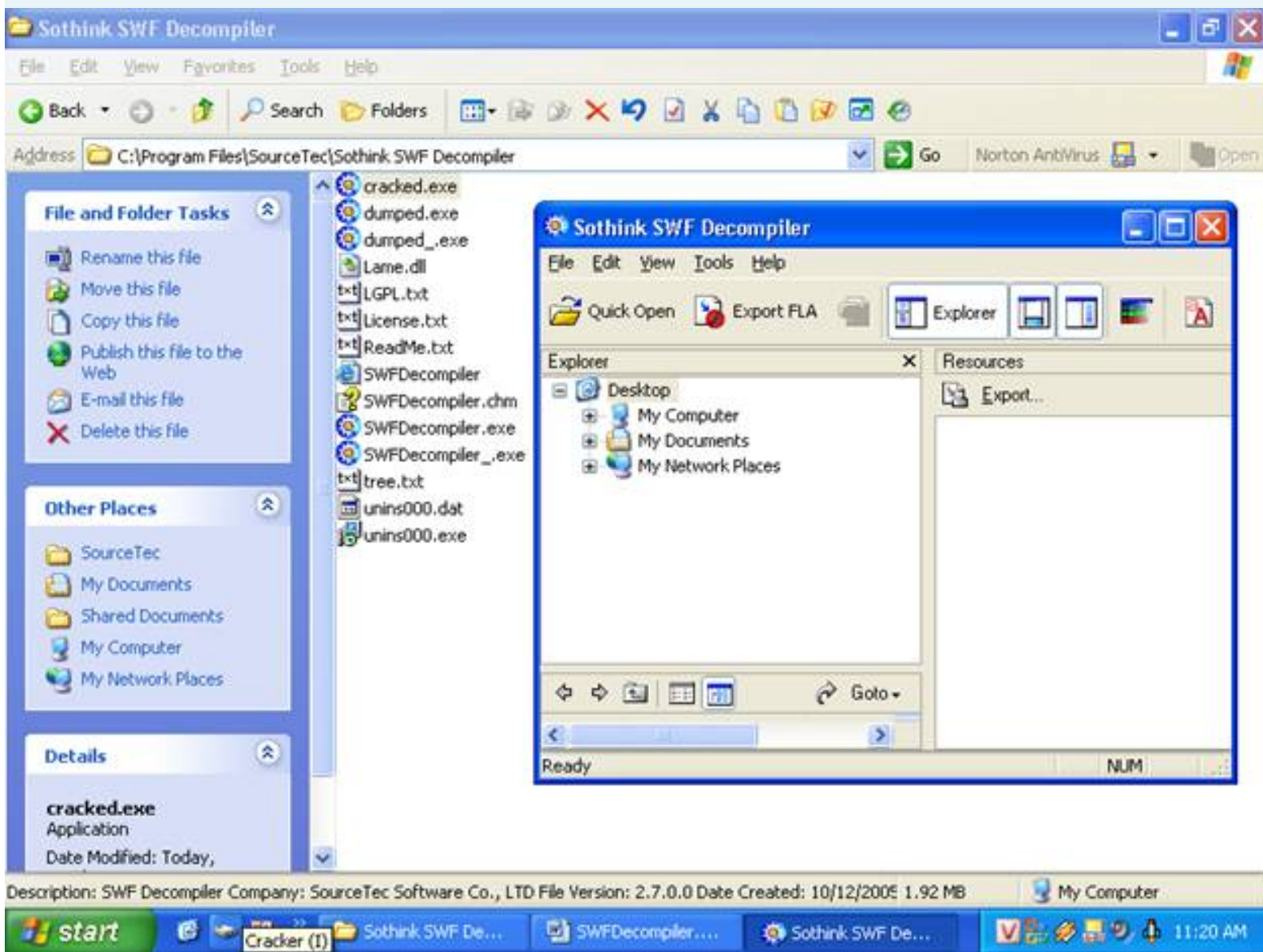
_Patch To:

Address	Hex dump	Disassembly	Comment
004128AE	85C0	TEST EAX,EAX	
004128B0	76 2A	JBE SHORT dumped_.004128DC	
004128B2	68 04010000	PUSH 104	
004128B7	8D4C24 08	LEA ECK,DWORD PTR SS:[ESP+8]	
004128BB	51	PUSH ECK	
004128BC	68 90555500	PUSH dumped_.00555590	ASCII "KEYTYPE"
004128C1	FFD6	CALL ESI	
004128C3	85C0	TEST EAX,EAX	
004128C5	76 15	JBE SHORT dumped_.004128DC	
004128C7	8D5424 04	LEA EDX,DWORD PTR SS:[ESP+4]	
004128CB	52	PUSH EDX	
004128CC	E8 8C1D0A00	CALL dumped_.004B465D	
004128D1	83C4 04	ADD ESP,4	
004128D4	5E	POP ESI	
004128D5	81C4 04010000	ADD ESP,104	
004128D8	C3	RETN	
004128DC	B0 01	MOV AL,1	
004128DE	5E	POP ESI	
004128DF	81C4 04010000	ADD ESP,104	
004128E5	C3	RETN	
004128E6	CC	INT3	
004128E7	CC	INT3	
004128E8	CC	INT3	
004128E9	CC	INT3	
004128EA	CC	INT3	
004128EB	CC	INT3	
004128EC	CC	INT3	
004128ED	CC	INT3	
004128EE	CC	INT3	
004128EF	CC	INT3	

_Save Again:



_Run Try cracked.exe:



_Unpacked.Cracked.by.hacnho?

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtlN, hosiminh, Nilrem, fly, MaDMAN_H3rCul3s, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light ... and you!

Special Thanx Cracks Latinos.

Merci FFF, RIF , N-Gen (closed), ICI-team me-pour aider des connaissances du Cracking Game!

Thanx OllyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 12/10/2005)

SoftWare : SWFText 1.2
Homepage : <http://www.antssoft.com/>
: + Standard ARMADiLLO protection only
Needed: 1. OllyDBG 1.1
2. LordPE Deluxe 1.4-by yoda
3. Import REConstructor 1.6 Final
4. CFF Explorer III
Author : Why Not Bar

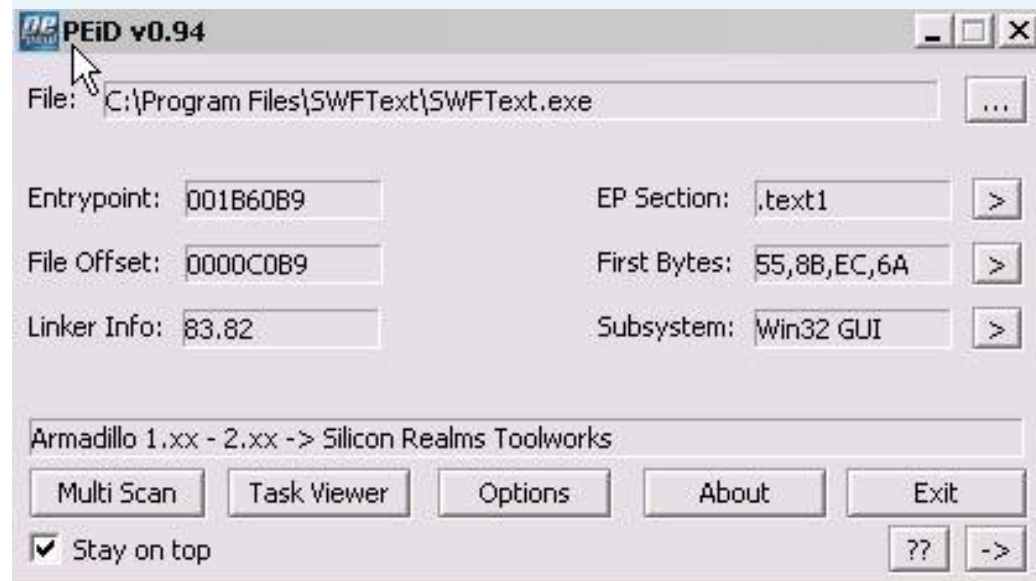
SWFText 1.2 is stated in **ARMADiLLO** but in no **Standard** IAT elimination + Codesplicing but to unpack it is not simple at all because the Magic Jump find quite some variation 1. With this tut you will present each particularized section to help you become familiar unpack Arma can understand and do.

Before you begin, to send a thank you to **iamidiot** guided Magic Fix Jump. Thanx Bro ..

_Tat The Tool use in TUT can Download at:

<http://tinicat.de/hacnho>

_Bao Hours so use **PeiD** scanning target is considered to pack in any form:



_ Next Target Test Run



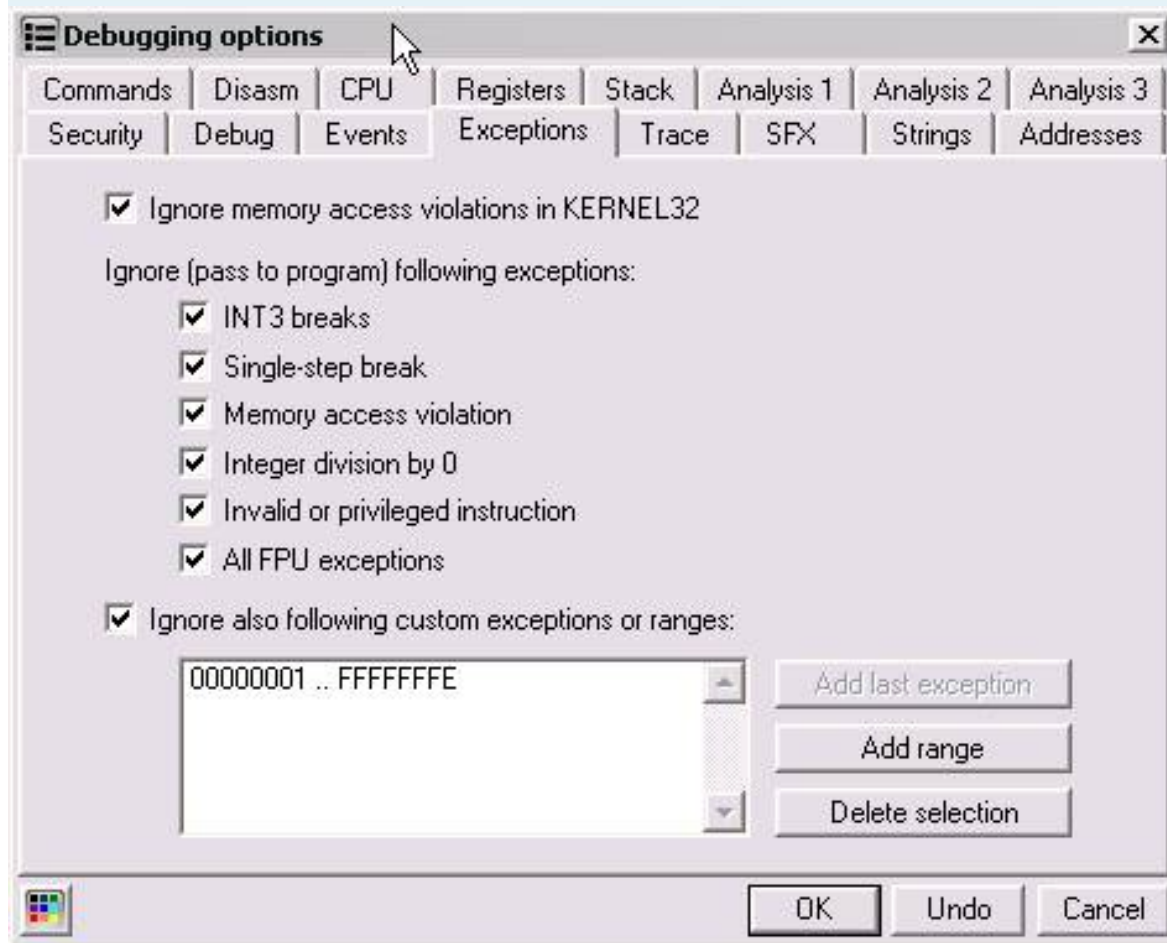
_ Then press **Ctrl + Alt + Delete** to start the taskbar Manager, and found only 1 process running so that is not **CopyMemII, Debugblocker Nanomites** and that only in **Standard**



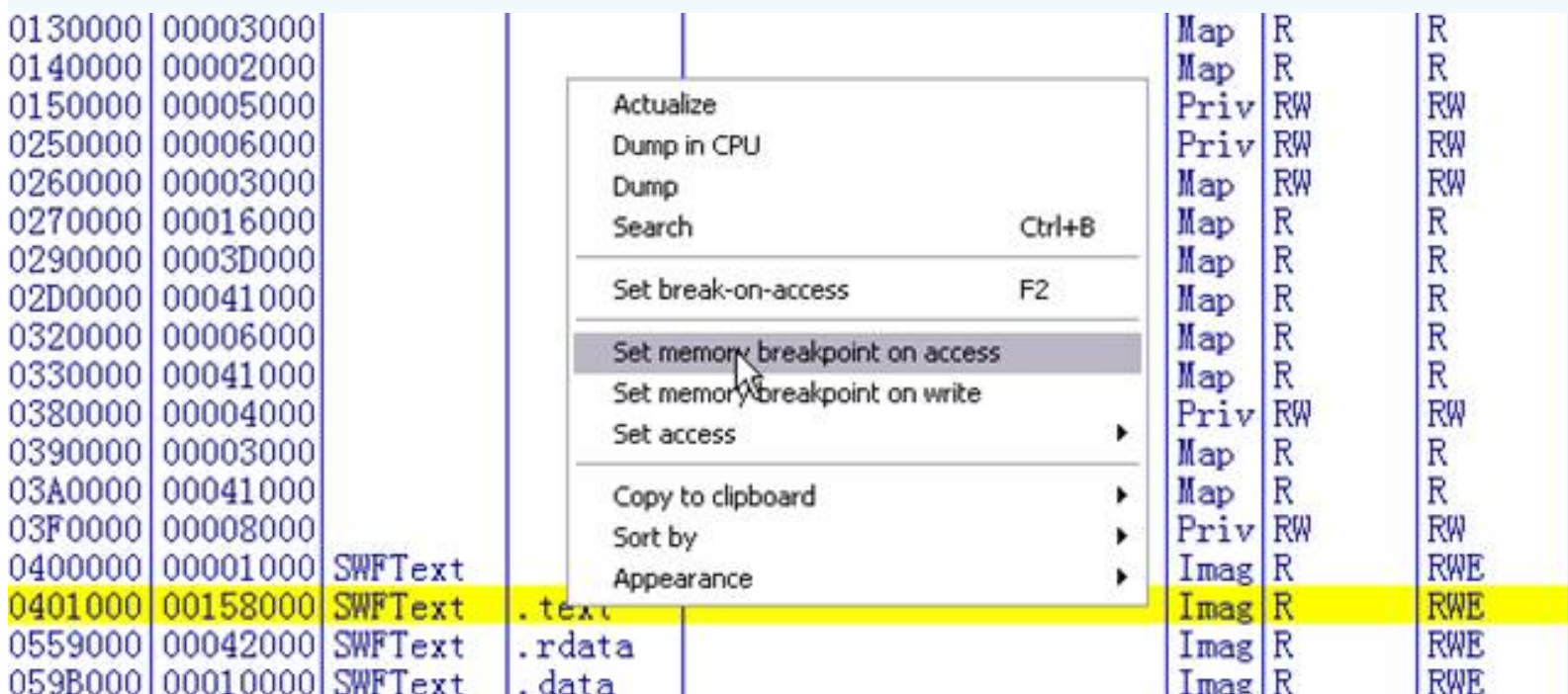
_ Load OllyDBG to target and we stopped at the **Entry Point** :

005B60B9	\$ 55	push ebp
005B60BA	. 8BEC	mov ebp, esp
005B60BC	. 6A FF	push -1
005B60BE	. 68 68E25C00	push SWFText.005CE268
005B60C3	. 68 005B5B00	push SWFText.005B5B00
005B60C8	. 64:A1 00000000	mov eax, dword ptr fs:[0]
005B60CE	. 50	push eax
005B60CF	. 64:8925 00000000	mov dword ptr fs:[0], esp
005B60D6	. 83EC 58	sub esp, 58

_ First, in Olly, we select the menu Options / debugging options, such as the image below:



_ Bay Time we actually find OEP of this program by **O pen W indow [Memory map] (Alt + M)**, click on the section. Text and set **break-on-access** on this section as the image:

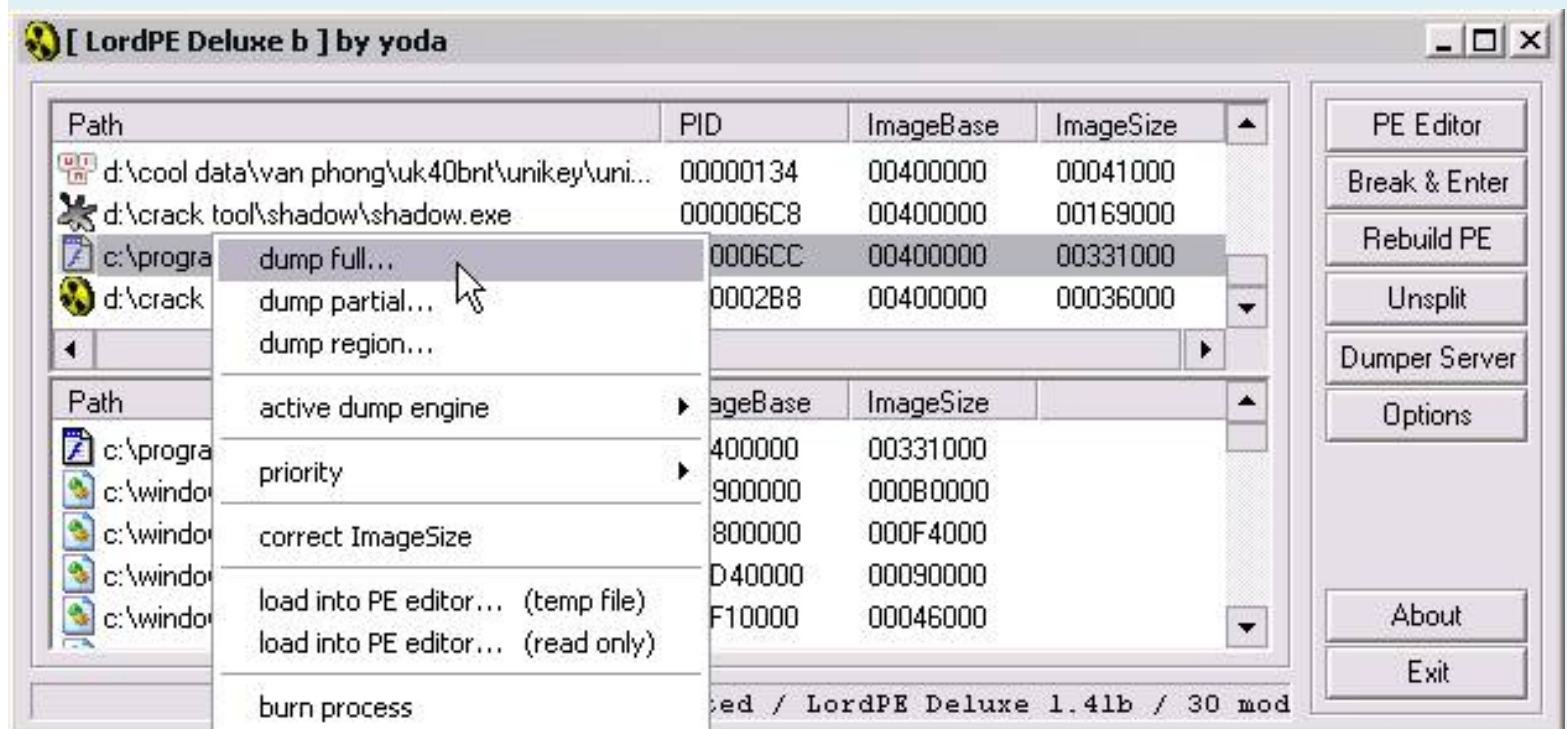


_ Press **Shift + F9**, and it will stop here and that is OEP's **original** program, you should remember the OEP = 00466EAA:

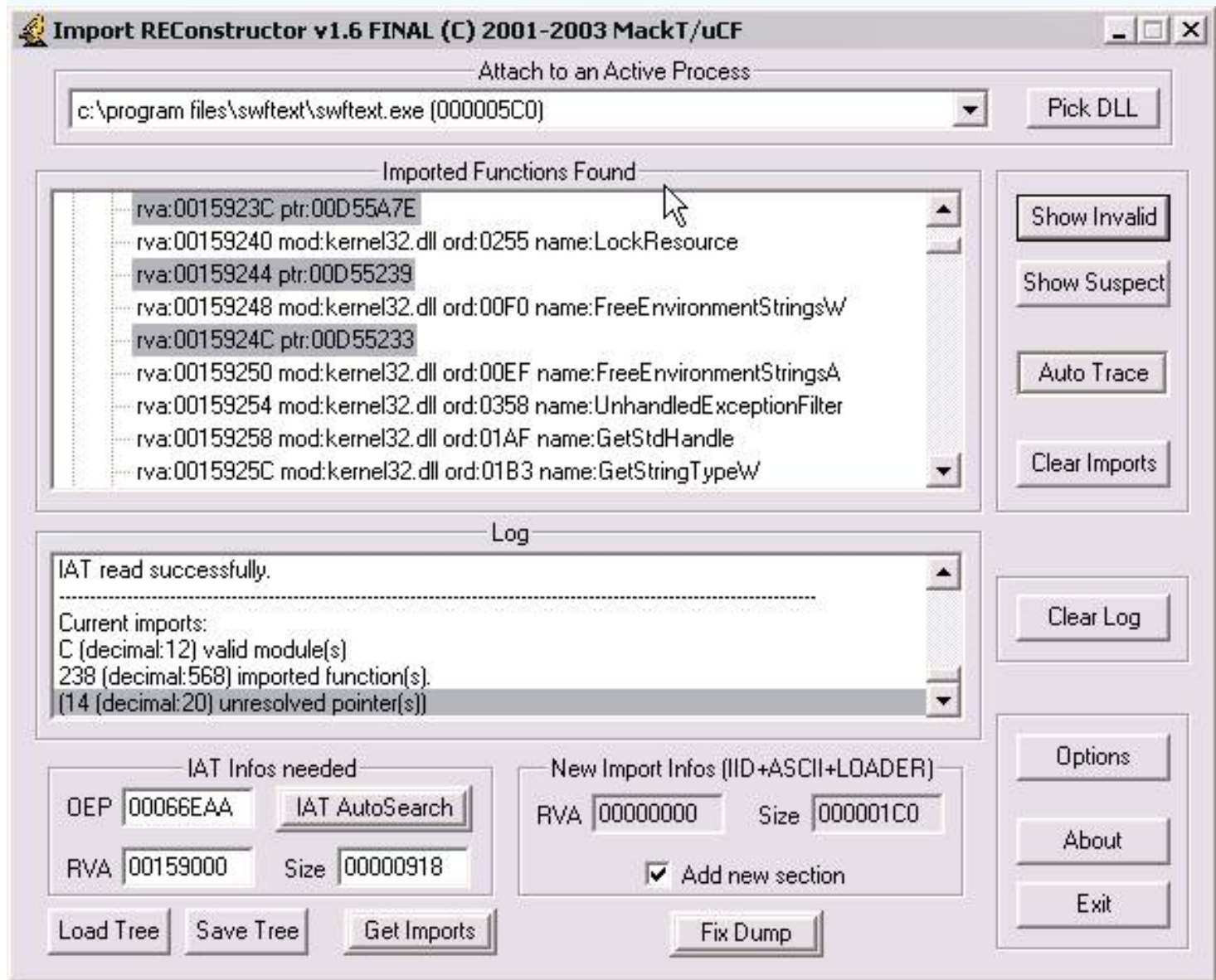
Address	Hex dump	Disassembly	Comment
00466EAA	6A 60	push 60	<== OEP
00466EAC	68 68B35600	push SWFText.0056B368	
00466EB1	E8 A62A0000	call SWFText.0046995C	
00466EB6	BF 94000000	mov edi, 94	
00466EBB	8BC7	mov eax, edi	
00466EBD	E8 3EFCFFFF	call SWFText.00466B00	
00466EC2	8965 E8	mov dword ptr ss:[ebp-18], esp	
00466EC5	8BF4	mov esi, esp	
00466EC7	893E	mov dword ptr ds:[esi], edi	
00466EC9	56	push esi	
00466ECA	FF15 6C945500	call dword ptr ds:[55946C]	kernel32.GetVersionExA
00466ED0	8B4E 10	mov ecx, dword ptr ds:[esi+10]	
00466ED3	890D FC725A00	mov dword ptr ds:[5A72FC], ecx	
00466ED9	8B46 04	mov eax, dword ptr ds:[esi+4]	
00466EDC	A3 08735A00	mov dword ptr ds:[5A7308], eax	
00466EE1	8B56 08	mov edx, dword ptr ds:[esi+8]	

_chac many ask why you **set** the **break-in on-access** section. **text** and press **Shift + F9** again to be the original **OEP**. Because the target Armadillo have run the task will unpacking and decryption of data into trouble chep section. **text**, then the enforcement of the directive section. **text** that has not **Section** code of original programs. So OEP will in this section. Therefore, we will set break-on-access on this section. Targets for the program access any data in this **section** will have a break and that the **OEP's** original program.

_Tiep As we proceed dump Full **SWFText.exe** process with LordPE files dumped.exe



_ Open **ImpRec 1.6**, Select process is **SWFText.exe**, fill **OEP = 66EAA (0x466EAA - 0x400000 = 66EAA)**, Press "IAT AutoSearch" Press button "Get Imports" and press button "Show Invalid" and we are as following



_Nhin To the **IAT** clearly we were going Arma hide some functions.'s Mission is we need to find Magic Jump and patch it to get the Full IAT. In Window CPU below 1 OEP we see little API function **kernel32. GetVersionExA**. C hung we will dump it in the window of Olly.Click to dump the mouse **00466ECA**, Select in **Folow dump / Memory address** as image.

The screenshot shows a debugger window with assembly code on the left and a context menu on the right. The assembly code is as follows:

Address	Disassembly	Comment
0466EB1	E8 A62A0000	call SWFTExt.0046995C
0466EB6	BF 94000000	mov edi, 94000000
0466EBB	8BC7	mov eax, edi
0466EBD	E8 3EFCFFFF	call SWFTExt.0046995C
0466EC2	8965 E8	mov dword [edi], eax
0466EC5	8BF4	mov esi, edi
0466EC7	893E	mov dword [edi], esi
0466EC9	56	push esi
0466ECA	FF15 6C945500	call dword [6C945500]
0466ED0	8B4E 10	mov ecx, [edi+10]
0466ED3	890D FC725A00	mov dword [FC725A00], ecx
0466ED9	8B46 04	mov eax, [edi+4]
0466EDC	A3 08735A00	mov dword [08735A00], eax
0466EE1	8B56 08	mov edi, [edi+8]

The context menu is open, showing the following options:

- Backup
- Copy
- Binary
- Assemble
- Label
- Comment
- Breakpoint
- Run trace
- Follow
- New origin here
- Go to
- Thread
- Follow in Dump
- Search for
- Find references to
- View
- Copy to executable

The menu is also showing the following options:

- Space
- :
- ;
- Enter
- Ctrl+Gray *
- Selection
- Memory address

The address 0466ECA is highlighted in yellow, and the instruction FF15 6C945500 is highlighted in blue. The address 0466EE1 is highlighted in red. The address 0466EE1 is highlighted in red.

In dump window, we set **1 breakpiont Hardware, write on / Dword** at address **0055946C** image as follows:

Address	Hex dump	Disassembly	Comment
00466EAA	6A 60	push 60	Backup EP
00466EAC	68 68B35600	push SWFText.	Copy
00466EB1	E8 A62A0000	call SWFText.	Binary
00466EB6	BF 94000000	mov edi, 94	Modify
00466EBB	8BC7	mov eax, edi	Label :
00466EBD	E8 3EFCFFFF	call SWFText.	Breakpoint Memory, on access
00466EC2	8965 E8	mov dword ptr	Search for Memory, on write
00466EC5	8BF4	mov esi, esp	Follow in Disassembler Remove memory breakpoint
00466EC7	893E	mov dword ptr	Follow in Dump
00466EC9	56	push esi	Find references Ctrl+R Hardware, on access
00466ECA	FF15 6C945500	call dword pt	Hardware, on write Byte
00466ED0	8B4E 10	mov ecx, dword	Hardware, on execution Word
00466ED3	890D FC725A00	mov dword ptr	Dword
00466ED9	8B46 04	mov eax, dword	
00466EDC	A3 08735A00	mov dword ptr	
00466EE1	8B56 08	mov edx, dword	
ds:[0055946C]=7C812851 (kernel32.GetVer			

Address	Value	Comment
00559464	7C80C6CF	kernel32.SetHandleCou
00559468	7C80BAF1	kernel32.SizeofResou
0055946C	7C812851	kernel32.GetVersionEs...
00559470	7C80A405	kernel32.GetThreadLocale
00559474	7C80D47E	kernel32.GetLocaleInfoA
00559478	7C800042	kernel32.GetACP

_Ok, Now press **Ctrl + F2** to Reload the target, press **Shift + F9** we *breakpoint* in which we **set breakpoint Hardware, write on / Dword** or hung as we will see the following:

Address	Hex dump	Disassembly	Comment
00D64553	F3:A5	rep movs dword ptr es:[edi], dword ptr ds:[esi]	
00D64555	FF2495 6846D60	jmp dword ptr ds:[edx*4+D64668]	
00D6455C	8BC7	mov eax, edi	
00D6455E	BA 03000000	mov edx, 3	
00D64563	83E9 04	sub ecx, 4	
00D64566	72 0C	jb short 00D64574	
00D64568	83E0 03	and eax, 3	
00D6456B	03C8	add ecx, eax	
00D6456D	FF2485 8045D60	jmp dword ptr ds:[eax*4+D64580]	
00D64574	FF248D 7846D60	jmp dword ptr ds:[ecx*4+D64678]	
00D6457B	90	nop	
00D6457C	FF248D FC45D60	jmp dword ptr ds:[ecx*4+D645FC]	
00D64583	90	nop	

_ Press **Shift + F9** and more to us here:

Address	Hex dump	Disassembly	Comment
00D62285	8B85 74FCFFFF	mov eax, dword ptr ss:[ebp-38C]	SWFText.0055946C
00D6228B	83C0 04	add eax, 4	
00D6228E	8B85 74FCFFFF	mov dword ptr ss:[ebp-38C], eax	
00D62294	E9 78FEFFFF	jmp 00D62111	
00D62299	0FB685 80FCFFF	movzx eax, byte ptr ss:[ebp-380]	
00D622A0	85C0	test eax, eax	
00D622A2	74 7F	je short 00D62323	
00D622A4	6A 00	push 0	

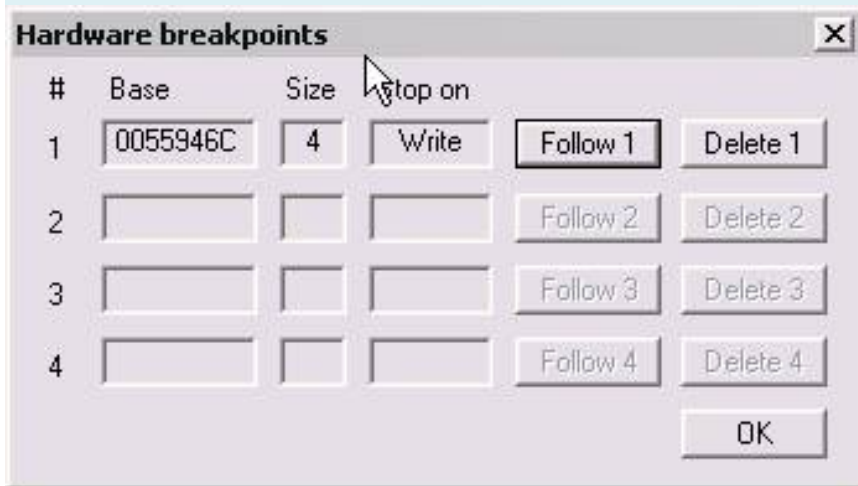
_ben vet under bright yellow 1 order **jmp 00D62111**. Ctrl + G and enter **00D62111** and we jump to here:

Address	Hex dump	Disassembly	Comment
00D620FA	C1E0 02	shl eax, 2	
00D620FD	50	push eax	
00D620FE	8B85 64FDFFFF	mov eax, dword ptr ss:[ebp-29C]	
00D62104	0385 7CFCFFFF	add eax, dword ptr ss:[ebp-384]	
00D6210A	50	push eax	
00D6210B	FF15 08B1D600	call dword ptr ds:[D6B108]	kernel32.VirtualProtect
00D62111	6A 01	push 1	
00D62113	58	pop eax	
00D62114	85C0	test eax, eax	
00D62116	0F84 7D010000	je 00D62299	
00D6211C	83A5 6CFCFFFF 00	and dword ptr ss:[ebp-394], 0	
00D62123	8B85 9CFEFFFF	mov eax, dword ptr ss:[ebp-164]	
00D62129	0FBEE0	movsx eax, byte ptr ds:[eax]	

_Cuon Mouse to the bottom and 1 stop bit at **00D6218F**

00D62188	0FB785 68FCFFFF	movzx eax,word ptr ss:[ebp-398]	
00D62188	50	push eax	
00D62189	FFB5 88FCFFFF	push dword ptr ss:[ebp-378]	
00D6218F	E8 D229FFFF	call 00D54B66	
00D62194	8985 6CFCFFFF	mov dword ptr ss:[ebp-394],eax	
00D6219A	83BD 6CFCFFFF 00	cmp dword ptr ss:[ebp-394],0	
00D621A1	75 58	jnz short 00D621FB	
00D621A3	FF15 C4B0D600	call dword ptr ds:[D6B0C4]	ntdll.RtlGetLastWin32Error
00D621A9	83F8 32	cmp eax,32	
00D621AC	75 0A	jnz short 00D621B8	

Enter to _Nhan and the **Call** command, but before you delete **breakpoint Hardware, write on / Dword** first time that we have set. Select Debug menu / Hardware breakpoints:



The button "Delete 1" to remove BP has set, and click OK. After one to enter into this:

Address	Hex dump	Disassembly	Comment
00D54B66	55	push ebp	
00D54B67	8BEC	mov ebp,esp	
00D54B69	53	push ebx	
00D54B6A	56	push esi	
00D54B6B	57	push edi	
00D54B6C	33FF	xor edi,edi	
00D54B6E	33DB	xor ebx,ebx	
00D54B70	66:F745 0E FFFF	test word ptr ss:[ebp+E],0FFFF	
00D54B76	75 03	jnz short 00D54B7B	
00D54B78	8B5D 0C	mov ebx,dword ptr ss:[ebp+C]	
00D54B7B	57	push edi	
00D54B7C	FF15 A4B0D600	call dword ptr ds:[D6B0A4]	kernel32.GetModuleHandleA
00D54B82	8B4D 08	mov ecx,dword ptr ss:[ebp+8]	
00D54B85	3BC8	cmp ecx,eax	
00D54B87	75 07	jnz short 00D54B90	

1 _cuon mouse down slightly yellow line is the Magic Jump

00D54B82	8B4D 08	mov ecx, dword ptr ss:[ebp+8]	
00D54B85	3BC8	cmp ecx, eax	
00D54B87	75 07	jnz short 00D54B90	
00D54B89	B8 18D3D600	mov eax, 0D6D318	
00D54B8E	EB 30	jmp short 00D54BC0	
00D54B90	393D D8D7D600	cmp dword ptr ds:[D6D7D8], edi	
00D54B96	B8 D8D7D600	mov eax, 0D6D7D8	
00D54B9B	74 0C	je short 00D54BA9	
00D54B9D	3B48 08	cmp ecx, dword ptr ds:[eax+8]	
00D54BA0	74 1B	je short 00D54BBD	
00D54BA2	83C0 0C	add eax, 0C	
00D54BA5	3938	cmp dword ptr ds:[eax], edi	
00D54BA7	75 F4	jnz short 00D54B9D	
00D54BA9	FF75 0C	push dword ptr ss:[ebp+C]	
00D54BAC	FF75 08	push dword ptr ss:[ebp+8]	
00D54BAF	E8 41000000	call 00D54BF5	
00D54BB4	59	pop ecx	
00D54BB5	59	pop ecx	
00D54BB6	5F	pop edi	
00D54BB7	5E	pop esi	
00D54BB8	5B	pop ebx	
00D54BB9	5D	pop ebp	
00D54BBA	C2 0800	retn 8	
00D54BBD	8B40 04	mov ecx, dword ptr ds:[eax+4]	

Set one hour _Bay 1 breakpoint Hardware, on Execution

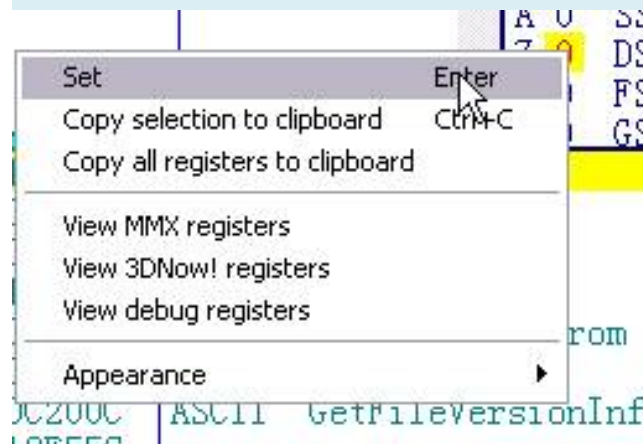
00D54B90	393D D8D7D600	cmp dword ptr ds:[D6D7D8], edi	
00D54B96	B8 D8D7D600	mov eax, 0D6D7D8	
00D54B9B	74 0C	je short 00D54BA9	
00D54B9D	3B48 08	cmp ecx, dword ptr ds:[eax+8]	
00D54BA0	74 1B	je short 00D54BBD	
00D54BA2	83C0 0C	add eax, 0C	
00D54BA5	3938	cmp dword ptr ds:[eax], edi	
00D54BA7	75 F4	jnz short 00D54B9D	
00D54BA9	FF75 0C	push dword ptr ss:[ebp+C]	
00D54BAC	FF75 08	push dword ptr ss:[ebp+8]	
00D54BAF	E8 41000000	call 00D54BF5	
00D54BB4	59	pop ecx	
00D54BB5	59	pop ecx	
00D54BB6	5F	pop edi	
00D54BB7	5E	pop esi	
00D54BB8	5B	pop ebx	
00D54BB9	5D	pop ebp	
00D54BBA	C2 0800	retn 8	
00D54BBD	8B40 04	mov ecx, dword ptr ds:[eax+4]	

Backup									
Copy									
Binary									
Assemble	Space								
Label	:								
Comment	;								
Breakpoint	<table> <tr> <td>Toggle</td><td>F2</td></tr> <tr> <td>Conditional</td><td>Shift+F2</td></tr> <tr> <td>Conditional log</td><td>Shift+F4</td></tr> <tr> <td>Run to selection</td><td>F4</td></tr> </table>	Toggle	F2	Conditional	Shift+F2	Conditional log	Shift+F4	Run to selection	F4
Toggle	F2								
Conditional	Shift+F2								
Conditional log	Shift+F4								
Run to selection	F4								
Follow	Enter								
New origin here	Ctrl+Gray *								
Go to									
Follow in Dump									
Search for									
Find references to									
Analysis									
Dump debugged process									

_Bay Time to press Ctrl + F2, Shift + F9 and we just stopped at Breakpoint Set. I used to come to Srcipt Fix IAT, the script has Code as follows:

IATscript.osc***dbh******oe LABEL******eob Babel******run******LABEL:******esto******jmp LABEL******Babel :******Cmp eip, 00 D54B9B******jne fin******mov! zF, 1******run******jmp Babel******Fin:******ret***

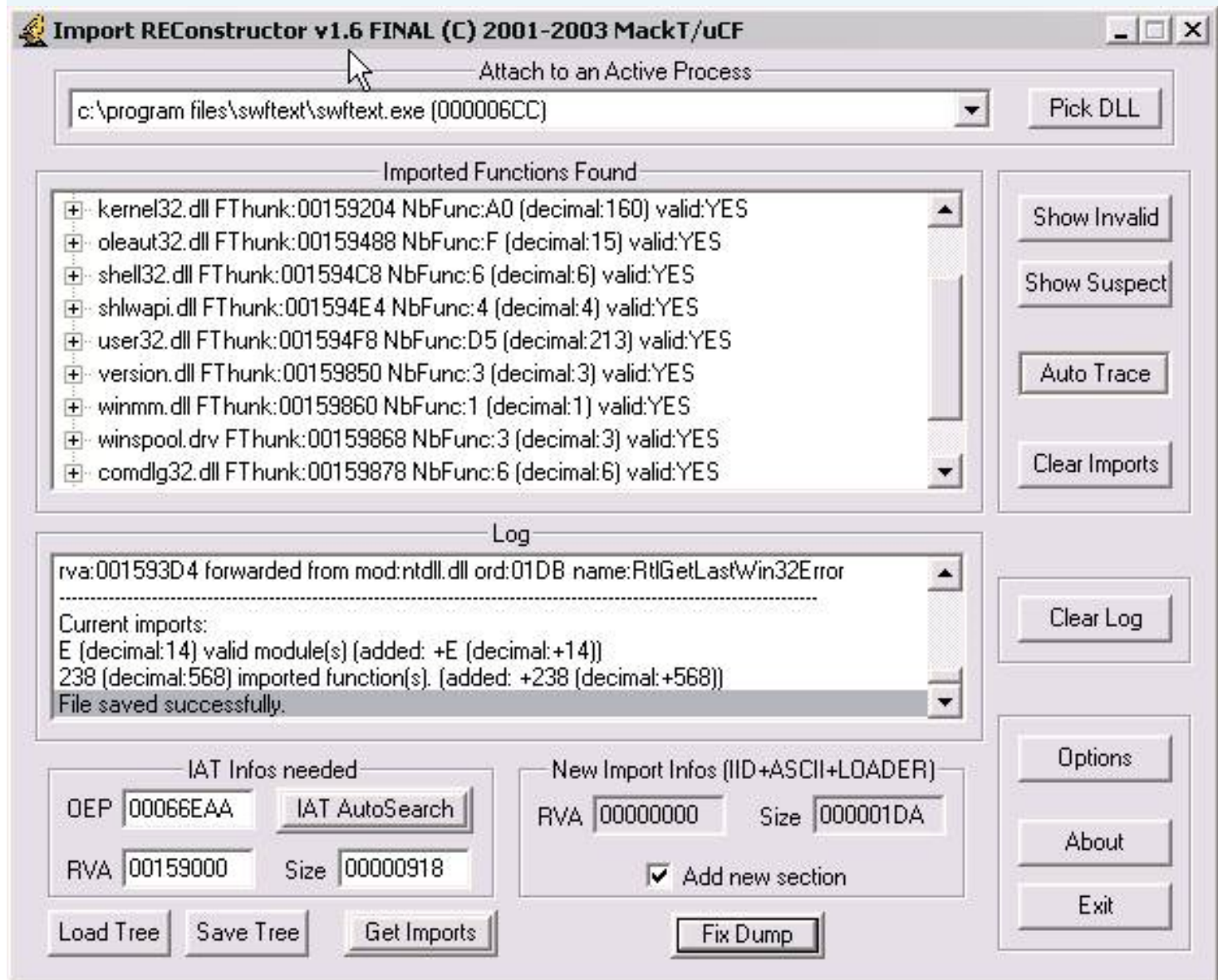
_Dia Only 00 D54B9B Scripts in the address Magic Jump we just stopped in Olly. Address this change by each of you should pay attention focus when running scripts. Some of the action before the Run Scripts you must set the flag Z = 1 if it is not with you must make it out to lose function when run Scripts



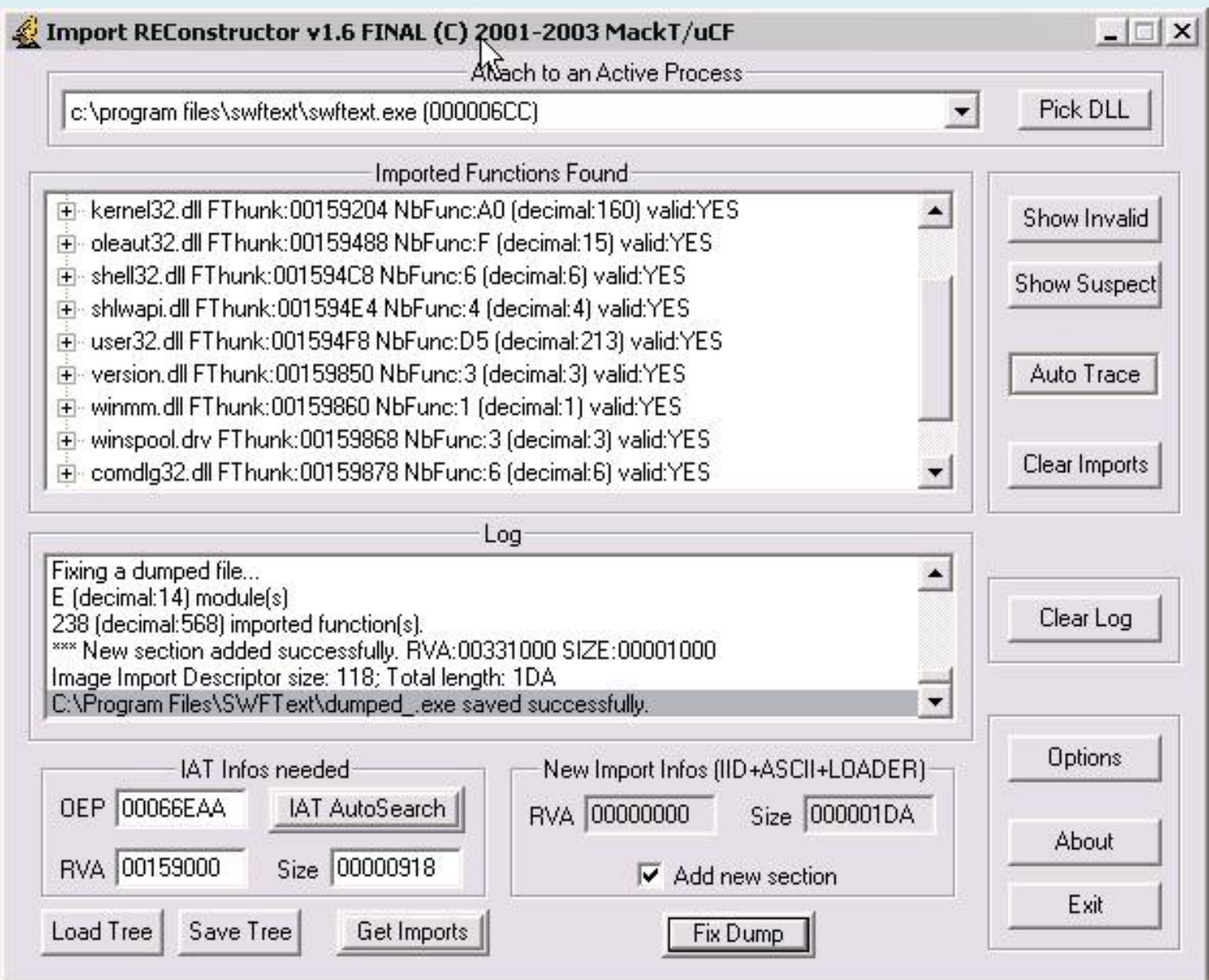
_Tiep We jump to the OEP 00466EAA = (Ctrl + G to enter 00466EAA) and set **1 breakpoint Hardware, OEP on Execution** in doing so to complete the IAT Fix Script automatically jump to OEP always.

_ Once done 2 requirements on our Run Scripts "IATscript.osc", the script is finished running

_tiep by **1.6 ImpRec Open**, Select proccess is **SWFText.exe**, fill **OEP 66EAA** = $((0x466EAA - 0x400000 = 66EAA)$, Press "IAT AutoSearch" Press button "Get Imports" and press button "Show Invalid" and it was like follows:



_he he, without any health **Invalid** re, Press button "Fix dump" file to fix IAT dumped.exe



Buoc Final Test Run **file dumped.exe**. Oh yeah, File run well, always take the NAG limited time as we accidentally Crack It has always gòi Suong ... ha !!!!!!!!!!!

==> **Unpack Done !!!!!**

_ Far as the process is completed unpack. But if you want to reduce the small space to unpack the files we conducted delete Section tàn Arma balance is also xót. **Pdata. Data1. Text1**. Of course the delete This section does not affect the program. Here we should use CFF Explorer for quick

CFF Explorer III - by Ntoskrnl - [dumped_.exe]

File Settings ?

File: dumped_.exe

- Dos Header
- Nt Headers
- File Header
- Optional Header
- Data Directories [15]
- Section Headers [x]
- Import Directory
- Resource Directory
- Resource Viewer
- Address Converter

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers
000002C8	000002D0	000002D4	000002D8	000002DC	000002E0	000002E4
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword
.text	00157D34	00001000	00157D34	00001000	00000000	00000000
.rdata	0004161C	00159000	0004161C	00159000	00000000	00000000
.data	0000F308	0019B000	0000F308	0019B000	00000000	00000000
.text1	00020000	001AB000	00020000	001AB000	00000000	00000000
.data1	00020000	001CB000	00020000	001CB000	00000000	00000000
.pdata	00130000	001EB000	00130000	001EB000	00000000	00000000
.rsrc	00016000	0031B000	00016000	0031B000	00000000	00000000
.mactt	00001000	00331000	00001000	00331000	00000000	00000000

_Sau Then Save the file capacity is 1.75 Mb

Dung PeiD scanning the File Dumped.exe considered stars

PEiD v0.94

File: C:\Program Files\SWFText\dumped_.exe

Entrypoint: 00066EAA EP Section: .text

File Offset: 00066EAA First Bytes: 6A,60,68,68

Linker Info: 83.82 Subsystem: Win32 GUI

Microsoft Visual C++ 7.0 [Debug]

Multi Scan Task Viewer Options About Exit

☒ Stay on top

Tut hope this helps you more. You find yourself writing so long but when you understand about how to insure your target, you just 2 minutes is OK ... wish you success.

Why Not Bar
24/1/2006

Tag & Rename 3.2 RC 3 - Inline Patching ASProtect 2.2 SKE -> Alexey Solodovnikov

Original Tutorial by Chinese fly - Convert to Vietnamese by hacnho

See the best in Web Layout mode 800 * 600



下载页面: <http://www.skycn.com/soft/2893.html>
软件大小: 2577 KB
软件语言: 英文
软件类别: 国外软件 / 共享版 / 文件更名
应用平台: Win9x/NT/2000/XP
加入时间: 2005-11-11 10:09:47
下载次数: 10274
开发商: <http://www.softpointer.com>
软件介绍: 使用 MP3/VQF 的 Tag Info 批量为文件改名。

【作者声明】: 只是感兴趣, 没有其他目的。失误之处敬请诸位大侠赐教

【调试环境】: WinXP、OllyDBG、PEID、LordPE、IDA

Download: <http://www.skycn.com/soft/2893.html>

Size: 2577 KB

Language: English

Categories: The overseas software / shares the version / document to change the name

Compatible: Win9x/NT/2000/XP

Lower paint J: 2005-11 - 1110:09:47

Luợt download: 10,274

Development: <http://www.softpointer.com>

The use: Uses the MP3/VQF Tag Info batch to change name as the document.

[Author Thanh Minh]: How tut feel for inspiration, not for purpose. Please ignore and hope for your forgiveness for my mistake!

[Tools]: WinXP, OllyDBG, PEiD, LordPE, IDA!

【脱壳过程】：

AsPrtect V2.X脱壳越来越麻烦，某些时候Inline Patch会省点事。国外不少人用此方法来对付AsPrtect，JohnWho写过教程，近来看到temerata也做了不少Inline Patch破解AsPrtect加壳程序。关于为何Patch可以参看以前的《Patch注册ASProtect V1.X壳保护程序的方法》。下面的记录算是再学习一次Inline Patch，没有新的成果。代码也较多，不适合新手练习。

感谢前人的研究成果。感谢shoooo的帮忙分析自校验。
下面以Tag&Rename.V3.2.RC3[3.2.56.119]为例来演示。

[Progress unpack]:

Asprotect 2.x packer is a very difficult game, unpack it very troubles, so the patch inline object of our life. Many people have written tut inline patch of form ASProtect this! Recently, JohnWho describe the curriculum (wrote a tut) to form inline patch by patch directly to ASProtect 1.x. Call na conduct as "Patch to register ASProtect V1.X Procedure Shell Protection Method" Today we will learn how to inline patch on the new form of ASProtect is 2.x. The process can inject code more complex, so not for you to know.

Thank in person the money of research results this (long lines this month, including J). Shoooo thank you helped me a lot in the process of analyzing this software!

一、获取壳代码重定位基址

前面部分的几个解压代码留在Patch部分分析，下面先简单跟踪一下流程。
设置Ollydbg忽略所有的异常选项，用IsDebug插件去掉Ollydbg的调试器标志。

CODE: [Copy to clipboard]

```
00401000      68 01A08700      push 87A001
//进入Ollydbg后暂停在这
00401005      E8 01000000      call 0040100B
0040100A      C3               retn
```

I: Load up OllyDBG target, remember to select the option HIDE Plugin IsDebuggerPresent.

```
00401000> 68 01A08700 PUSH TagRenam.0087A001 // First Olly temporarily stop here
00401005 E8 01000000 CALL TagRenam.0040100B
0040100A C3 RETN
```

HE VirtualAlloc Shift+F9，中断2次后取消断点，Alt+F9返回

CODE: [Copy to clipboard]

```
0087A517      FF95 F0030000      call dword ptr ss:[ebp+3F0] ; kernel32.VirtualAlloc
0087A51D      8985 31040000      mov dword ptr ss:[ebp+431],eax
//[ebp+431]=[0087A869]=01280000 壳代码重定位基址 ★
0087A523      8985 D0010000      mov dword ptr ss:[ebp+1D0],eax
//[ebp+1D0]=[0087A608]=01280000
```

HE VirtualAlloc, Shift + F9 2 times, HD VirtualAlloc, then Alt + F9, to here:

```
0087A517 FF95 F0030000 call dword ptr ss: [ebp + 3 F0]; kernel32.VirtualAlloc
```

```
0087A51D 8985 31040000 mov dword ptr ss:[ebp+431], eax
/ / [ebp+431] = [0087A869] = 01280000
0087A523 8985 D0010000 ss mov dword ptr [ebp+1 D0], eax
/ / [ebp+1 D0] = [0087A608] = 01280000
```

二、壳代码完全解压

HE GetModuleHandleA Shift+F9，中断2次后取消断点，Alt+F9返回

CODE: [Copy to clipboard]

```
012B14A6        FF95 EC314400        call dword ptr ss:[ebp+4431EC]
012B14AC        85C0                test eax,eax
012B14AE        75 07                jnz short 012B14B7
```

II: Now ASProtect was completely unwind on memory:

HE GetModuleHandleA Shift + F9 2 times, delete breakpoint, Alt + F9:

```
012B14A6 FF95 EC314400 call dword ptr ss:[ebp+4431 EC]
012B14AC 85C0 test eax, eax
012B14AE 75 07 jnz short 012B14B7
```

Ctrl+F向下找popad，看到AsPrtect解压壳代码的那个循环

CODE: [Copy to clipboard]

```
012B15C1        61                popad
//记住这里 ★
012B15C2        75 08                jnz short 012B15CC
012B15C4        B8 01000000        mov eax,1
012B15C9        C2 0C00                retn 0C
012B15CC        68 DC892A01        push 12A89DC
012B15D1        C3                retn
```

Ctrl + F: POPAD, to here:

012815C1	61	popad	//Remembers here *
012815C2	75 08	jnz short 012815CC	
012815C4	B8 01000000	mov eax,1	
012815C9	C2 0C00	retn 0C	
012815CC	68 00000000	push 0	
012815D1	C3	retn	

三、自校验：CreateFileMapping

自从AsPrtect V1.X被直接Patch后作者在V2.X里面增强了自校验。
要想办法找到可以搞定自校验的时机，AsPrtect开始准备自校验的同时也给我们提供了解决它的机会。
Ctrl+G:CreateFileMappingA 在函数尾下断，中断后取消断点，Alt+F9返回

CODE: [Copy to clipboard]

```

01298630      6A 00      push 0
01298632      6A 00      push 0
01298634      6A 00      push 0
01298636      6A 02      push 2
//需要修改push 8 ★
01298638      6A 00      push 0
0129863A      53          push ebx
0129863B      A1 E4972A01  mov eax,dword ptr ds:[12A97E4]
01298640      8B40 1C      mov eax,dword ptr ds:[eax+1C]
01298643      FFD0          call eax ; kernel32.CreateFileMappingA
01298645      A3 14B42A01  mov dword ptr ds:[12AB414],eax
0129864A      53          push ebx
0129864B      A1 E4972A01  mov eax,dword ptr ds:[12A97E4]
01298650      8B40 18      mov eax,dword ptr ds:[eax+18]
01298653      FFD0          call eax
01298655      833D 14B42A01 00  cmp dword ptr ds:[12AB414],0
0129865C      0F84 66040000  je 01298AC8
01298662      6A 00      push 0
01298664      6A 00      push 0
01298666      6A 00      push 0
01298668      6A 04      push 4
//需要修改push 1 ★
0129866A      A1 14B42A01  mov eax,dword ptr ds:[12AB414]
0129866F      50          push eax
01298670      A1 E4972A01  mov eax,dword ptr ds:[12A97E4]
01298675      8B40 08      mov eax,dword ptr ds:[eax+8]
01298678      FFD0          call eax ; kernel32.MapViewOfFileEx
0129867A      8BD8          mov ebx,eax
//EAX值是映像文件的开始地址 ★
0129867C      50          push eax
0129867D      E8 4A010000  call 012987CC

```

III: Ctrl + G: CreateFileMappingA, F2, Shift + F9 2 times, delete breakpoint, Alt + F9:

```

01298630 6A 00 push 0
01298632 6A 00 push 0
01298634 6A 00 push 0
01298636 6A 02 push 2
/ / Should be revised to push 8 ★
01298638 6A 00 push 0
0129863A 53 push ebx
0129863B A1 E4972A01 mov eax, dword ptr ds: [12A97E4]
01298640 8B40 1C mov eax, dword ptr ds: [eax +1 C]
01298643 FFD0 call eax; kernel32.CreateFileMappingA
01298645 A3 14B42A01 mov dword ptr ds: [12AB414], eax
0129864A 53 push ebx
0129864B A1 E4972A01 mov eax, dword ptr ds: [12A97E4]
01298650 8B40 18 mov eax, dword ptr ds: [eax +18]
01298653 FFD0 call eax
01298655 833D 14B42A01 00 Cmp dword ptr ds: [12AB414], 0
0129865C 0F84 66040000 je 01298AC8
01298662 6A 00 push 0
01298664 6A 00 push 0
01298666 6A 00 push 0
01298668 6A 04 push 4
/ / Should be revised to push ★ 1

```



```

0129866A A1 14B42A01 mov eax, dword ptr ds: [12AB414]
0129866F 50 push eax
01298670 A1 E4972A01 mov eax, dword ptr ds: [12A97E4]
01298675 8B40 08 mov eax, dword ptr ds: [eax +8]
01298678 FFD0 call eax; kernel32.MapViewOfFileEx
0129867A 8BD8 mov ebx, eax
/ / Value of EAX is mapped by EP 401000 ★
0129867C 50 push eax
0129867D E8 4A010000 call 012987CC

```

为何不仅仅修改MapViewOfFileEx的参数？看看MSDN

CODE: [Copy to clipboard]

```

HANDLE CreateFileMapping(
    HANDLE hFile,                // handle to file
    LPSECURITY_ATTRIBUTES lpAttributes, // security
    DWORD flProtect,             // protection
    DWORD dwMaximumSizeHigh,     // high-order DWORD of size
    DWORD dwMaximumSizeLow,      // low-order DWORD of size
    LPCTSTR lpName               // object name
);

flProtect :
2=PAGE_READONLY
4=PAGE_READWRITE
8=PAGE_WRITECOPY //需要修改成这个参数

```

Win9X平台上若不修改CreateFileMapping，则MapViewOfFileEx的FILE_MAP_COPY会失败

CODE: [Copy to clipboard]

```

LPVOID MapViewOfFileEx(
    HANDLE hFileMappingObject, // handle to file-mapping object
    DWORD dwDesiredAccess,     // access mode
    DWORD dwFileOffsetHigh,    // high-order DWORD of offset
    DWORD dwFileOffsetLow,     // low-order DWORD of offset
    SIZE_T dwNumberOfBytesToMap, // number of bytes to map
    LPVOID lpBaseAddress       // starting address
);

dwDesiredAccess :
1=FILE_MAP_COPY
4=FILE_MAP_READ

```

Ham CreateFileMapping in MSDN:

```

HANDLE CreateFileMapping (
HANDLE hFile, / / handle to file to map
LPSECURITY_ATTRIBUTES lpAttributes, / / optional security Attributes
DWORD flProtect, / / protection for mapping object
DWORD dwMaximumSizeHigh, / / high-order 32 bits of object size
DWORD dwMaximumSizeLow, / / low-order 32 bits of object size
LPCTSTR lpName / / name of the file-mapping object
);

```

flProtect

Specifies the desired protection for the file view, when the file is mapped. This parameter can be one of the following values:

ValueDescription

PAGE_READONLY Gives read-only access to the committed region of pages. An attempt to write to or execute the committed region results in an access indoctination. The file specified by the hFile parameter must have been created with GENERIC_READ access.

PAGE_READWRITE Gives read-write access to the committed region of pages. The file specified by hFile must have been created with GENERIC_READ and GENERIC_WRITE access.

Gives PAGE_WRITECOPY copy on write access to the committed region of pages. The files specified by the hFile parameter must have been created with GENERIC_READ and GENERIC_WRITE access.

Ham MapViewOfFileEx in MSDN:

```

LPVOID MapViewOfFileEx (
HANDLE hFileMappingObject, / / file-mapping object to map into address
space
DWORD dwDesiredAccess, / / access mode
DWORD dwFileOffsetHigh, / / high-order 32 bits of file offset
DWORD dwFileOffsetLow, / / low-order 32 bits of file offset
DWORD dwNumberOfBytesToMap, / / number of bytes to map
LPVOID lpBaseAddress / / suggested starting address for mapped view
);

```

Parameters

hFileMappingObject

Identifies an open handle to a file-mapping object. The CreateFileMapping OpenFileMapping functions and return this handle.

dwDesiredAccess

Specifies the type of access to the file-mapping object and, therefore, the page protection of the pages mapped by the file. This parameter can be one of the following values:

ValueMeaning

FILE_MAP_WRITE Read-and-write access. The hFileMappingObject parameter must have been created with PAGE_READWRITE protection. A read-write view of the file is mapped.

FILE_MAP_READ read-only access. The hFileMappingObject parameter must have been created with PAGE_READWRITE or PAGE_READONLY protection. A read-only

view of the file is mapped.
Same as FILE_MAP_ALL_ACCESS FILE_MAP_WRITE.
FILE_MAP_COPY Copy on write access. If you create the map with PAGE_WRITECOPY the view with FILE_MAP_COPY, you will receive a view to the file. If you write to it, the pages are automatically swappable and the modifications you make will not go to the original data file.Windows 95: You must pass PAGE_WRITECOPY to CreateFileMapping; otherwise, an error will be returned.If you share the mapping between multiple DuplicateHandle processes using one or OpenFileMapping process and writes to a view, the modification is propagated to the other process. The original file does not change.Windows NT: There is no restriction as to how the parameter hFileMappingObject must be created. Copy on write is valid for any type of view. If you share the mapping between multiple processes using DuplicateHandle or OpenFileMapping and one process writes to a view, the modification is not propagated to the other process. The original file does not change.

Copy on write access. If you create the map with PAGE_WRITECOPY the view with FILE_MAP_COPY, you will receive a view to the file. If you write to it, the pages are swappable automatically and make modifications will you go to the not original data file.

Windows 95/98/ME: You must pass PAGE_WRITECOPY to CreateFileMapping; otherwise, an error will be returned.

四、获得ASProtect注册名Pre-Dip处理的地址

现在设置Ollydbg忽略除了“内存访问异常、同时忽略以下制定异常”之外的所有其它异常选项。
Shift+F9 运行N次，直至堆栈中第2次看见硬盘指纹：

CODE: [Copy to clipboard]

```
0013FF1C    0013FF28    指针到下一个 SEH 记录
0013FF20    0129C356    SE 句柄
0013FF24    00000000
0013FF28    0013FF80    指针到下一个 SEH 记录
0013FF2C    0129D173    SE 句柄
0013FF30    0013FF78
0013FF34    01280000
0013FF38    01240000
0013FF3C    0129B470
0013FF40    00000000
0013FF44    012D5468    ASCII "wWDDDAAgLTg="
0013FF48    012D5434    ASCII "AFBC4FD7-E635"
```

IV: Search Hardware Finger Print of ASProtect by pressing Shift + F9 a few times until you reach here:

```
0013FF1C 0013FF28 point to next SEH Record
0013FF20 0129C356 SE 句柄 (SE cú Comments: P t h a t i f I d o n o t d o i s S e n t e n s e H a n d l e J )
0013FF24 00000000
0013FF28 0013FF80 SEH Record point to the next
SE 0013FF2C 0129D173 句柄
0013FF30 0013FF78
0013FF34 01280000
0013FF38 01240000
0013FF3C 0129B470
0013FF40 00000000
0013FF44 012D5468 ASCII "wWDDDAAgLTg"
```


0013FF48 012D5434 ASCII "AFBC4FD7-E635"

Alt+M打开内存查看窗口，在00401000段“设置内存访问断点”，Shift+F9，断下

CODE: [Copy to clipboard]

```
004B7B44      55                push ebp
//中断在这里
004B7B45      8BEC             mov ebp,esp
004B7B47      A1 B0037100      mov eax,dword ptr ds:[7103B0]
004B7B4C      A3 AC037100      mov dword ptr ds:[7103AC],eax
004B7B51      8B45 08          mov eax,dword ptr ss:[ebp+8]
004B7B54      A3 B0037100      mov dword ptr ds:[7103B0],eax
004B7B59      5D              pop ebp
004B7B5A      C2 0400         retn 4
```

Press Alt + M, located on Access Memory Breakpoint, F9 + Shiftf, stop here:

```
004B7B44 55 push ebp
/ / break here
004B7B45 8BEC mov ebp, esp
004B7B47 A1 B0037100 mov eax, dword ptr ds: [7103B0]
004B7B4C A3 AC037100 ds mov dword ptr [7103AC], eax
004B7B51 8B45 08 mov eax, dword ptr ss: [ebp +8]
004B7B54 A3 B0037100 mov dword ptr ds: [7103B0], eax
004B7B59 5D pop ebp
004B7B5A C2 0400 retn 4
```

取消内存断点，向上看：

CODE: [Copy to clipboard]

```
004B7B18      55                push ebp
//下断，Shift+F9直至中断在这里，取消断点
004B7B19      8BEC             mov ebp,esp
004B7B1B      8B45 08          mov eax,dword ptr ss:[ebp+8]
//[ebp+8]=[0012FF24]=01283A29 注册名保存地址
004B7B1E      85C0             test eax,eax
004B7B20      74 0C            je short 004B7B2E
004B7B22      8038 00          cmp byte ptr ds:[eax],0
004B7B25      74 07            je short 004B7B2E
004B7B27      C605 B4037100 01 mov byte ptr ds:[7103B4],1
//注意：[7103B4]应该置1 ★
004B7B2E      8B15 A8037100      mov edx,dword ptr ds:[7103A8]
004B7B34      8915 A4037100      mov dword ptr ds:[7103A4],edx
004B7B3A      A3 A8037100      mov dword ptr ds:[7103A8],eax
//注册名保存地址放入[7103A8] ★
004B7B3F      5D              pop ebp
004B7B40      C2 0400         retn 4
//返回0129CABF
```

CODE: [Copy to clipboard]

```

0129CAB9      50          push eax
//注册名保存地址      ★
0129CABA      8B47 04      mov eax,dword ptr ds:[edi+4]
0129CABD      FFD0          call eax ; 004B7B18

```

Clear Memory breakpoint, roll up a few lines:

```

004B7B18 55 push ebp
/ / Set a breakpoint on acces here, Shift + F9, ice, remove breakpoint
004B7B19 8BEC mov ebp, esp
004B7B1B 8B45 08 mov eax, dword ptr ss: [ebp +8]
/ / [ebp +8] = [0012FF24] = 01283A29 stored registration information
004B7B1E 85C0 test eax, eax
004B7B20 74 0C je short 004B7B2E
Cmp 004B7B22 8038 00 byte ptr ds: [eax], 0
004B7B25 74 07 je short 004B7B2E
004B7B27 C605 B4037100 01 mov byte ptr ds: [7103B4], 1
/ / Note: [7103B4] equal to 1 (see the stack) ★
004B7B2E 8B15 A8037100 mov edx, dword ptr ds: [7103A8]
004B7B34 8915 A4037100 mov dword ptr ds: [7103A4], edx
004B7B3A A3 A8037100 mov dword ptr ds: [7103A8], eax
/ / Value of the registration information is recorded in the last variable EAX: [7103A8] ★
004B7B3F 5D pop ebp
004B7B40 C2 0400 retn 4
/ / Return 0129CABF

0129CAB9 50 push eax
/ / Save registration information ★
0129CABA 8B47 04 mov eax, dword ptr ds: [edi +4]
0129CABD FFD0 call eax; 004B7B18

```

五、程序中需要修改的地方

但是这里我们直接修改此Pre-Dip后程序还是显示未注册，呵呵
Tag&Rename和AlfaClock、Bee Icons等不同，还需要修改程序

1、判断是否是注册版

CODE: [Copy to clipboard]

```

0070CFC5      E8 FE4FDBFF      call 004C1FC8
//进入修改
0070CFCA      84C0              test al,al
0070CFCC      0F85 0E010000     jnz 0070D0E0
0070CFD2      33C9              xor ecx,ecx
0070CFD4      B2 01             mov dl,1
0070CFD6      A1 C45E6A00       mov eax,dword ptr ds:[6A5EC4]
0070CFDB      E8 B485D8FF       call 00495594
0070CFE0      8B15 D88F7100     mov edx,dword ptr ds:[718FD8]
0070CFE6      8902              mov dword ptr ds:[edx],eax
0070CFE8      33C0              xor eax,eax
0070CFEA      55                push ebp
0070CFEB      68 B9D07000       push 70D0B9
0070CFF0      64:FF30           push dword ptr fs:[eax]
0070CFF3      64:8920           mov dword ptr fs:[eax],esp
0070CFF6      A1 D88F7100       mov eax,dword ptr ds:[718FD8]
0070CFFB      8B00              mov eax,dword ptr ds:[eax]
0070CFFD      E8 9ED8D8FF       call 0049A8A0
0070D002      A1 D88F7100       mov eax,dword ptr ds:[718FD8]
0070D007      8B00              mov eax,dword ptr ds:[eax]
0070D009      8B10              mov edx,dword ptr ds:[eax]
0070D00B      FF92 88000000     call dword ptr ds:[edx+88]
0070D011      8B0D B0867100     mov ecx,dword ptr ds:[7186B0]
0070D017      A1 B48B7100       mov eax,dword ptr ds:[718BB4]
0070D01C      8B00              mov eax,dword ptr ds:[eax]
0070D01E      8B15 6C876E00     mov edx,dword ptr ds:[6E876C]
0070D024      E8 CF1AD9FF       call 0049EAF8
0070D029      A1 D88F7100       mov eax,dword ptr ds:[718FD8]
0070D02E      8B00              mov eax,dword ptr ds:[eax]
0070D030      8B10              mov edx,dword ptr ds:[eax]
0070D032      FF92 88000000     call dword ptr ds:[edx+88]
0070D038      68 DC050000       push 5DC
0070D03D      E8 3A4DD0FF       call 00411D7C
0070D042      A1 B0867100       mov eax,dword ptr ds:[7186B0]
0070D047      8B00              mov eax,dword ptr ds:[eax]
0070D049      BA 70D27000       mov edx,70D270 ; ASCII "Tag&Rename 3.2 rc 3 UNREGISTERED"
0070D04E      E8 418AD6FF       call 00475A94

```

CODE: [Copy to clipboard]

```

004C20B1      C645 FB 00        mov byte ptr ss:[ebp-5],0
//Patch 点:  C645 FB 01        mov byte ptr ss:[ebp-5],1 ★
004C20B5      EB 2E             jmp short 004C20E5

```

V: The order restored improving local destination: P, conduct Argentina is starting to inline patch to the program was where to find the correct patch. We have found the Unreg such as the patch and then, but a soft pack with the way 2.x ASProtect patch will vary, for example billion compared with Tag Rename BeeIcon or AlfaClock!

1. We judgments code is related to the check Registered or unregistered!

```

0070CFC5 E8 FE4FDBFF call 004C1FC8
/ / Call this function to
0070CFCA 84C0 test al, al
0070CFCC 0F85 0E010000 jnz 0070D0E0
0070CFD2 33C9 xor ecx, ecx
0070CFD4 B2 01 mov dl, 1
0070CFD6 A1 C45E6A00 mov eax, dword ptr ds: [6A5EC4]
0070CFDB E8 B485D8FF call 00495594
0070CFE0 8B15 D88F7100 mov edx, dword ptr ds: [718FD8]
0070CFE6 8902 mov dword ptr ds: [edx], eax
0070CFE8 33C0 xor eax, eax
0070CFEA 55 push ebp
0070CFEB 68 B9D07000 push 70D0B9
0070CFF0 64: FF30 push dword ptr fs: [eax]
0070CFF3 64:8920 mov dword ptr fs: [eax], esp
0070CFF6 A1 D88F7100 mov eax, dword ptr ds: [718FD8]
0070CFFB 8B00 mov eax, dword ptr ds: [eax]
0070CFFD E8 9ED8D8FF call 0049A8A0
0070D002 A1 D88F7100 mov eax, dword ptr ds: [718FD8]
0070D007 8B00 mov eax, dword ptr ds: [eax]
0070D009 8B10 mov edx, dword ptr ds: [eax]
0070D00B FF92 88000000 call dword ptr ds: [edx +88]
0070D011 8B0D B0867100 mov ecx, dword ptr ds: [7186B0]
0070D017 A1 B48B7100 mov eax, dword ptr ds: [718BB4]
0070D01C 8B00 mov eax, dword ptr ds: [eax]
0070D01E 8B15 6C876E00 mov edx, dword ptr ds: [6E876C]
0070D024 E8 CF1AD9FF call 0049EAF8
0070D029 A1 D88F7100 mov eax, dword ptr ds: [718FD8]
0070D02E 8B00 mov eax, dword ptr ds: [eax]
0070D030 8B10 mov edx, dword ptr ds: [eax]
0070D032 FF92 88000000 call dword ptr ds: [edx +88]
0070D038 68 DC050000 push 5DC
0070D03D E8 3A4DD0FF call 00411D7C
0070D042 A1 B0867100 mov eax, dword ptr ds: [7186B0]
0070D047 8B00 mov eax, dword ptr ds: [eax]
THREE 0070D049 mov edx 70D27000, 70D270; ASCII "Tag & Rename 3.2 rc 3 unregistered"
0070D04E E8 418AD6FF call 00475A94

```

Jump to this, change the flag patch check:

```

004C20B1 C645 FB 00 mov byte ptr ss: [ebp-5], 0
/ / Patch's: C645 FB 01 mov byte ptr ss: [ebp-5], 1 ★
004C20B5 EB 2e jmp short 004C20E5

```

2. 显示注册名

CODE: [Copy to clipboard]

```
00684611    E8 8230D8FF    call 00407698
00684616    8D45 E4        lea eax,dword ptr ss:[ebp-1C]
00684619    8B55 F8        mov edx,dword ptr ss:[ebp-8]
//[ebp-8]指向注册名，最后一起修改
0068461C    E8 770DD8FF    call 00405398
```

不要问我是如何找的，大家都会调试，只不过某些人在这方面用的时间多点而已。

2. Display the list (here means Registration Name):

```
00684611 E8 8230D8FF call 00407698
00684616 8D45 E4 Lea eax, dword ptr ss: [ebp-1C]
00684619 8B55 F8 mov edx, dword ptr ss: [ebp-8]
/ / [ebp-8] is a pointer to the value name Registration
0068461C E8 770DD8FF call 00405398
```

Do not ask me why the stars of the J back in, you can debug, only that I've made too many soft and should nhòm one that is right, it is a habit, you must create your own habits that (damn, and Medical nhe relatives, this month it per confess, confess on maize, dek bít where that open)!

六、RegOpenKeyExA

Ctrl+F在“整个段块”搜索命令：push 20019

CODE: [Copy to clipboard]

```
0128A289    68 19000200    push 20019
//找到这里
0128A28E    6A 00          push 0
0128A290    8BC7          mov eax,edi
0128A292    E8 8D97FFFF    call 01283A24
0128A297    50            push eax
0128A298    56            push esi
0128A299    E8 36B4FFFF    call 012856D4 ; jmp to advapi32.RegOpenKeyExA
//Call RegOpenKeyExA
0128A29E    85C0          test eax,eax
0128A2A0    0F85 84000000  jnz 0128A32A
```

点右键，搜索全部命令：call 012856D4

地址 反汇编 注释

CODE: [Copy to clipboard]

```
0128A299    call 012856D4    jmp to advapi32.RegOpenKeyExA
0128A376    call 012856D4    jmp to advapi32.RegOpenKeyExA
0128A3E2    call 012856D4    jmp to advapi32.RegOpenKeyExA
0128A495    call 012856D4    jmp to advapi32.RegOpenKeyExA
0128A57D    call 012856D4    jmp to advapi32.RegOpenKeyExA
0129E1C3    call 012856D4    jmp to advapi32.RegOpenKeyExA
0129E260    call 012856D4    jmp to advapi32.RegOpenKeyExA
```

```

0129E1C3  call 012856D4      jmp to advapi32.RegOpenKeyExA
0129E260  call 012856D4      jmp to advapi32.RegOpenKeyExA

```

一般修改后面3个CALL后的跳转就行了，记住这几个地方
其实在本例中不需要修改这里，但是为了解说也Patched了

CODE: [Copy to clipboard]

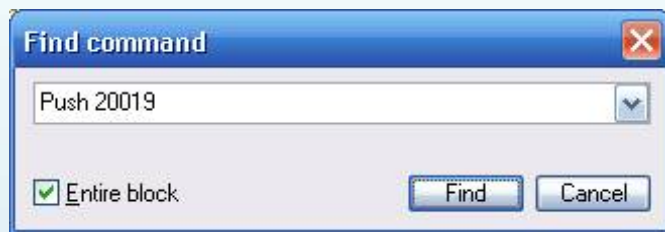
```

0128A57D  E8 52B1FFFF  call 012856D4 ; jmp to advapi32.RegOpenKeyExA
0128A582  85C0        test eax,eax
0128A584  75 30       jnz short 0128A5B6
0129E1C3  E8 0C75FEFF  call 012856D4 ; jmp to advapi32.RegOpenKeyExA
0129E1C8  85C0        test eax,eax
0129E1CA  75 42       jnz short 0129E20E
0129E260  E8 6F74FEFF  call 012856D4 ; jmp to advapi32.RegOpenKeyExA
0129E265  85C0        test eax,eax
0129E267  0F85 88000000 jnz 0129E2F5

```

VI: RegOpenKeyA

Ctrl + F:



To here:

```

0128A289 68 19000200 push 20019
// You will be here!
0128A28E 6A 00 push 0
0128A290 8BC7 mov eax, edi
0128A292 E8 8D97FFFF call 01283A24
0128A297 50 push eax
0128A298 56 push esi
0128A299 E8 36B4FFFF call 012856D4; jmp to advapi32.RegOpenKeyExA
// Call RegOpenKeyExA
0128A29E 85C0 test eax, eax
0128A2A0 0F85 84000000 jnz 0128A32A

```

Call 012856D4 in line, you enter here to jump to:

```

0128A299 call 012856D4 jmp to advapi32.RegOpenKeyExA
0128A376 call 012856D4 jmp to advapi32.RegOpenKeyExA
0128A3E2 call 012856D4 jmp to advapi32.RegOpenKeyExA
0128A495 call 012856D4 jmp to advapi32.RegOpenKeyExA
0128A57D call 012856D4 jmp to advapi32.RegOpenKeyExA
0129E1C3 call 012856D4 jmp to advapi32.RegOpenKeyExA
0129E260 call 012856D4 jmp to advapi32.RegOpenKeyExA

```

Remember the position of this line. Ctrl + R to see what the call of this function:

```

0128A57D E8 52B1FFFF call 012856D4; jmp to advapi32.RegOpenKeyExA

```



```

0128A582 85C0 test eax, eax
0128A584 75 30 jnz short 0128A5B6
0129E1C3 E8 0C75FEFF call 012856D4; jmp to advapi32.RegOpenKeyExA
0129E1C8 85C0 test eax, eax
0129E1CA 75 42 jnz short 0129E20E
0129E260 E8 6F74FEFF call 012856D4; jmp to advapi32.RegOpenKeyExA
0129E265 85C0 test eax, eax
0129E267 0F85 88000000 jnz 0129E2F5

```

七、一起 Patch 吧

大体流程清楚了，下面开始Patch吧。复制TagRename.exe改名为Patch.exe

看看Patch.exe的PE信息，SizeOfImage=004A2000，区段信息如下：

CODE: [Copy to clipboard]

No	Name	VSize	VOffset	RSize	ROffset	Charact
01		0030E000	00001000	000F4A00	00000400	E0000040
02		0000B000	0030F000	00004000	000F4E00	E0000040
03		00002000	0031A000	00000000	000F8E00	E0000040
04		00004000	0031C000	00003800	000F8E00	E0000040
05		00001000	00320000	00000200	000FC600	E0000040
06		00001000	00321000	00000000	000FC800	E0000040
07		00001000	00322000	00000200	000FC800	E0000040
08		00029000	00323000	00000000	000FCA00	E0000040
09	.rsrc	0012E000	0034C000	0009BA00	000FCA00	E0000040
0A	.0000	00027000	0047A000	00026400	00198400	E0000040
0B	.adata	00001000	004A1000	00000000	001BE800	E0000040

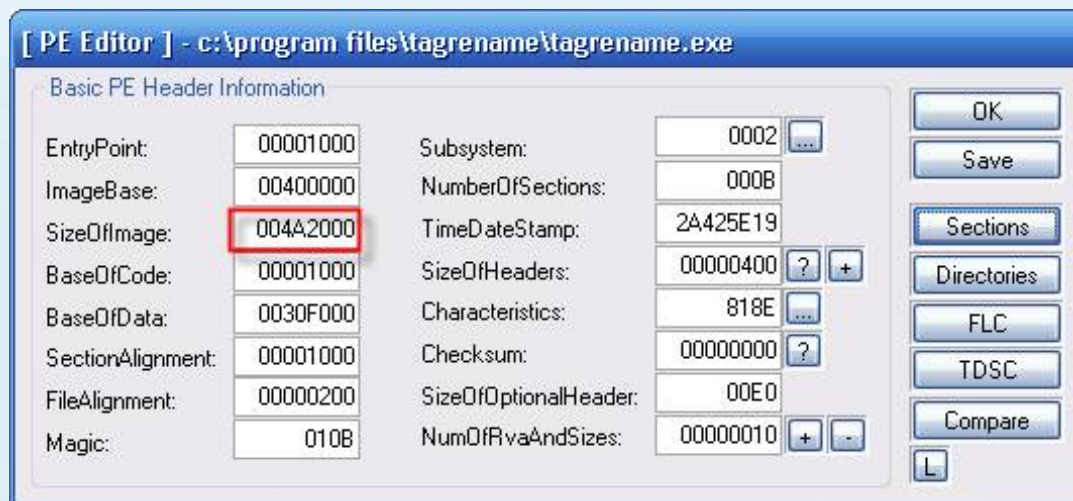
用LordPE把最后一个区段VSize和RSize都改为1500，保存后LordPE自动把SizeOfImage改为004A2500

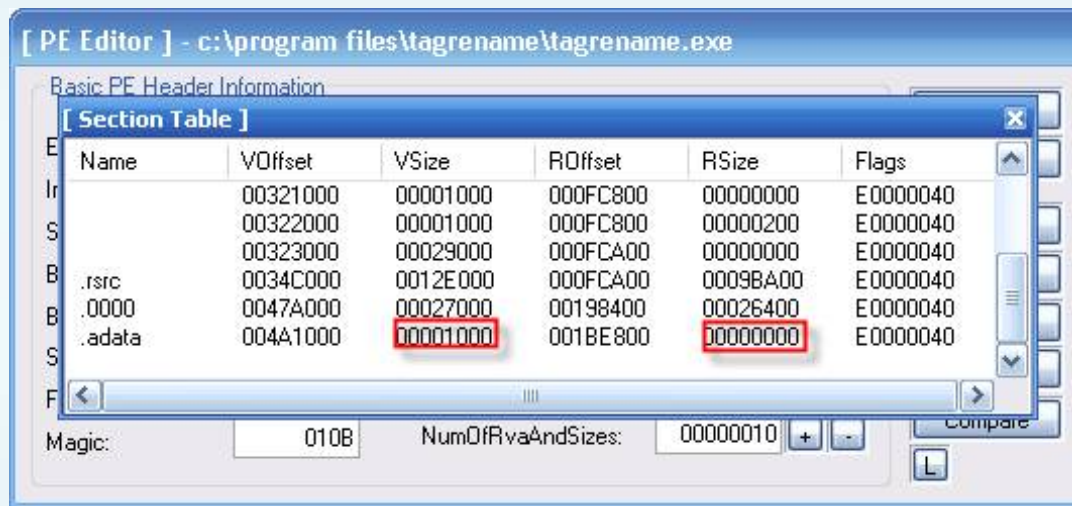
用WinHex在Patch.exe末尾添加1500H长度的00数据。当然，如果代码不多的话可以在程序中找到空地。这样添加1500H长度后我们就获得了后面500H空白可以用，而不必管是否会被其他覆盖了。008A2000（偏移0X1BF800）处我们就可以写代码了。

新开OllyDBG载入Patch.exe，我们要先Patch几个循环解码，再Patch上面六步分析的地方。

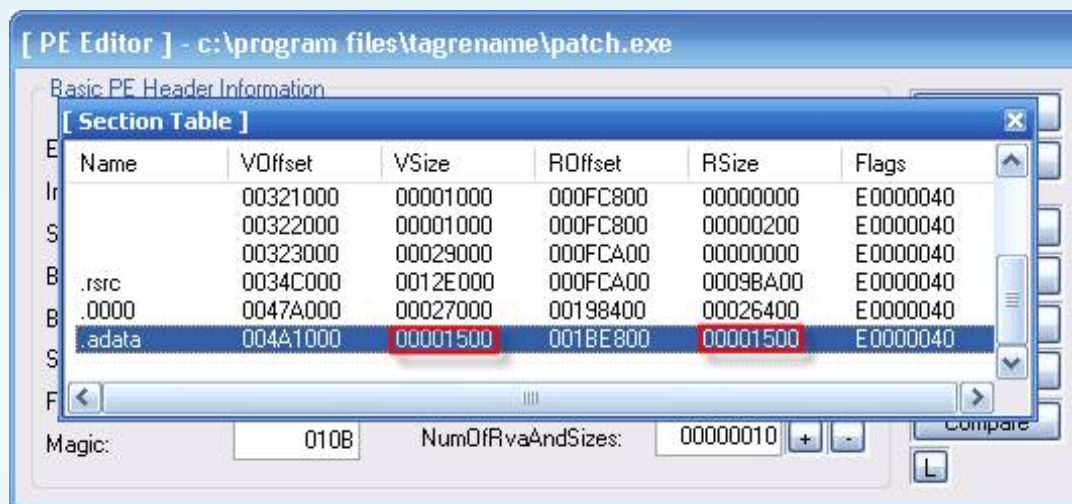
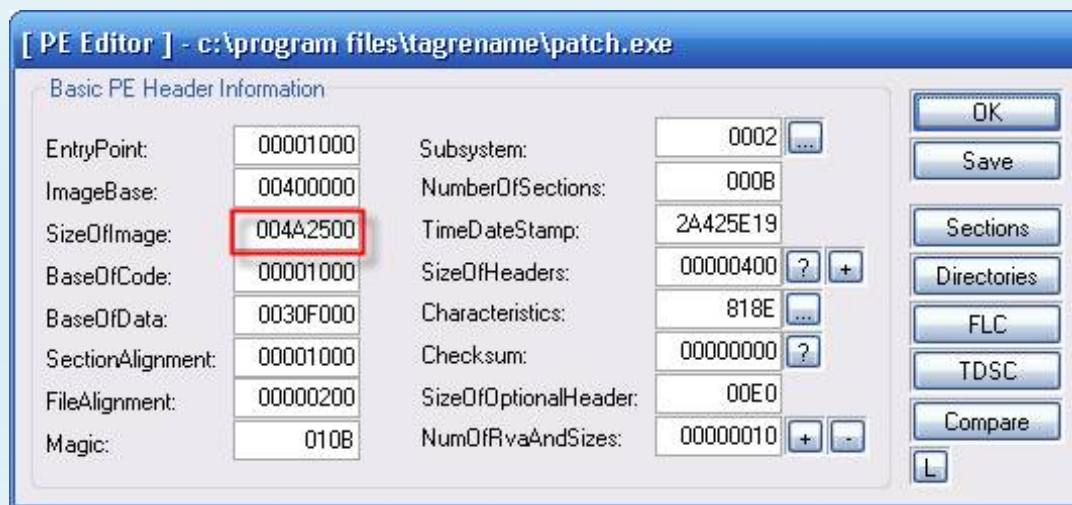
VII: Start patch:

Copy the file TagRename.exe and renamed Patch.exe.'s Info View PE file Patch.exe





Using LordPE change into:



The expansion of the sections and imagebase help us to have a cave to inject code. Ok, now open a new OllyDBG, Patch.exe to load, we need to patch some loop check code of ASProtect then inject it into the following steps:

1、第1个点

CODE: [Copy to clipboard]

```

00401000      68 01A08700      push 87A001
//进入Olllydbg后暂停在这
00401005      E8 01000000      call 0040100B
0040100A      C3                retn

```

F7单步走，看ASProtect解压壳代码

CODE: [Copy to clipboard]

```

0087A0FD      BE A93C493D      mov esi,3D493CA9
0087A102      81F6 543D493D      xor esi,3D493D54
0087A108      B8 EB8B342E      mov eax,2E348BEB
0087A10D      8B0A            mov ecx,dword ptr ds:[edx]
0087A10F      0F8B 08000000      jpo 0087A11D
0087A115      68 924FB86F      push 6FB84F92
0087A11A      8BFB            mov edi,ebx
0087A11C      5F              pop edi
0087A11D      81C1 21A29555      add ecx,5595A221
0087A123      80DB 8D          sbb bl,8D
0087A126      81C1 4646A24E      add ecx,4EA24646
0087A12C      51              push ecx
0087A12D      68 AF332832      push 322833AF
0087A132      B0 5B            mov al,5B
0087A134      5F              pop edi
0087A135      58              pop eax
0087A136      81C1 0770A50C      add ecx,0CA57007
0087A13C      0F81 00000000      jno 0087A142
0087A142      890A            mov dword ptr ds:[edx],ecx
0087A144      81EA 31F64154      sub edx,5441F631
0087A14A      BB A208717E      mov ebx,7E7108A2
0087A14F      81C2 2DF64154      add edx,5441F62D
0087A155      8BC7            mov eax,edi
0087A157      83EE 01          sub esi,1
0087A15A      0F85 ADFFFFFF      jnz 0087A10D
//解码循环 第1个点 接口 ★
//Patch ①、 E9 E17E0200      jmp 008A2040

```

Patch代码：

CODE: [Copy to clipboard]

```

008A2040      0F85 C780FDFF      jnz 0087A10D
//0087A15A代码挪这里执行
008A2046      C705 5AA18700 0F85>mov dword ptr ds:[87A15A],FFAD850F
008A2050      66:C705 5EA18700 F>mov word ptr ds:[87A15E],0FFFF
//还原0087A15A处修改的代码
008A2059      C705 D8A18700 8C7E>mov dword ptr ds:[87A1D8],27E8C
//Patch第2个点
008A2063      E9 F880FDFF      jmp 0087A160
//继续流程

```

1. Step 1:


```

00401000 68 01A08700 push 87A001
/ / You are here after load up OllyDBG
00401005 E8 01000000 call 0040100B
0040100A C3 retn

```

Press F7 is some way to surf the ASProtect unpacking code completely in memory:

```

0087A0FD BE A93C493D mov esi, 3D493CA9
0087A102 81F6 543D493D xor esi, 3D493D54
0087A108 b8 EB8B342E mov eax, 2E348BEB
0087A10D 8B0A mov ecx, dword ptr ds: [edx]
0087A10F 0F8B 08000000 jpo 0087A11D
0087A115 68 924FB86F push 6FB84F92
0087A11A 8BFB mov edi, ebx
0087A11C 5F pop edi
0087A11D 81C1 21A29555 add ecx, 5595A221
0087A123 80DB sbb BL 8D, 8D
0087A126 81C1 4646A24E add ecx, 4EA24646
0087A12C 51 push ecx
0087A12D 68 AF332832 push 322833AF
0087A132 mov al B0 5B, 5B
0087A134 5F pop edi
0087A135 58 pop eax
0087A136 81C1 0770A50C add ecx, 0CA57007
0087A13C 0F81 00000000 jno 0087A142
0087A142 890A mov dword ptr ds: [edx], ecx
0087A144 81EA 31F64154 sub edx, 5441F631
0087A14A BB A208717E mov ebx, 7E7108A2
0087A14F 81C2 2DF64154 add edx, 5441F62D
0087A155 8BC7 mov eax, edi
0087A157 83EE 01 sub esi, 1

```

```
0087A15A 0F85 ADFFFFFF jnz 0087A10D
```

/ / Loop decompress code, we need to patch this command jump:

```
0087A15A E9 E17E0200 jmp 008A2040
```

008A2040 at one patch as follows:

Cave 1:

```

008A2040 0F85 C780FDFF jnz 0087A10D
/ / I think the jump from the command loop at 0087A15A
008A2046 C705 5AA18700 0F85> mov dword ptr ds: [87A15A], FFAD850F
008A2050 66: C705 5EA18700 F> mov word ptr ds: [87A15E], 0FFFF
008A2059 C705 D8A18700 8C7E> mov dword ptr ds: [87A1D8], 27E8C
/ / Patch through step 2
008A2063 E9 F880FDFF jmp 0087A160
/ / Next to the store (ie after inject code this, do we jump to a command 0087A160,
this command after jumping at 008715A!)

```

2、第2个点

CODE: [Copy to clipboard]

```

0087A192    FF30          push dword ptr ds:[eax]
0087A194    B9 3390F94E   mov ecx,4EF99033
0087A199    5F           pop edi
0087A19A    0FB7CB       movzx ecx,bx
0087A19D    81F7 1D90B243 xor edi,43B2901D
0087A1A3    8BD7         mov edx,edi
0087A1A5    81EF 92625E0E sub edi,0E5E6292
0087A1AB    81D1 52F1012A adc ecx,2A01F152
0087A1B1    81C7 63A4242F add edi,2F24A463
0087A1B7    8AEF         mov ch,bh
0087A1B9    8938         mov dword ptr ds:[eax],edi
0087A1BB    B3 B8        mov bl,0B8
0087A1BD    83E8 04      sub eax,4
0087A1C0    68 4D3C7652  push 52763C4D
0087A1C5    51           push ecx
0087A1C6    80CA 05      or dl,5
0087A1C9    59           pop ecx
0087A1CA    5A           pop edx
0087A1CB    4E           dec esi
0087A1CC    0F85 18000000 jnz 0087A1EA
0087A1D2    66:81F1 80F3  xor cx,0F380
0087A1D7    E9 3A000000  jmp 0087A216
//解码循环 第2个点
//Patch ②、 E9 8C7E0200  jmp 008A2068

```

Patch代码：

CODE: [Copy to clipboard]

```

008A2068    C705 D8A18700 3A00>mov dword ptr ds:[87A1D8],3A
//还原0087A1D8处修改的代码
008A2072    C705 CFA28700 E9B6>mov dword ptr ds:[87A2CF],27DB6E9
//Patch第3个点
008A207C    66:C705 D3A28700 0>mov word ptr ds:[87A2D3],0FF00
008A2085    E9 4D81FDFF   jmp 0087A1D7

```

2. Step 2:

```

0087A192 FF30 push dword ptr ds: [eax]
0087A194 B9 3390F94E mov ecx, 4EF99033
0087A199 5F pop edi
0087A19A 0FB7CB movzx ecx, BX
0087A19D 81F7 1D90B243 xor edi, 43B2901D
0087A1A3 8BD7 mov edx, edi
0087A1A5 81EF 92625E0E sub edi, 0E5E6292
0087A1AB 81D1 52F1012A ADC ecx, 2A01F152
0087A1B1 81C7 63A4242F add edi, 2F24A463
0087A1B7 8AEF the mov, BH
0087A1B9 8938 mov dword ptr ds: [eax], edi
0087A1BB B3 b8 mov BL, 0B8
0087A1BD 83E8 04 sub eax, 4
0087A1C0 68 4D3C7652 push 52763C4D

```

```

0087A1C5 51 push ecx
0087A1C6 80CA 05 or dl, 5
0087A1C9 59 pop ecx
0087A1CA 5A pop edx
0087A1CB 4E dec esi
0087A1CC 0F85 18000000 jnz 0087A1EA
0087A1D2 66:81 F1 80F3 xor cx, 0F380
0087A1D7 E9 3A000000 jmp 0087A216
/ / Need to patch Jump the position that we need to inject code, here is the cave 2:
0087A1D7 E9 8C7E0200 jmp 008A2068

```

Jump to 008A2068 (cbu like the same):

Cave 2:

```

008A2068 C705 D8A18700 3A00> mov dword ptr ds: [87A1D8], 3A
008A2072 C705 CFA28700 E9B6> mov dword ptr ds: [87A2CF], 27DB6E9
008A207C 66: C705 D3A28700 0> mov word ptr ds: [87A2D3], 0FF00
008A2085 E9 4D81FDFF jmp 0087A1D7

```

3、第3个点

CODE: [Copy to clipboard]

```

0087A2C4      83EA 04          sub edx,4
0087A2C7      BB E633514B     mov ebx,4B5133E6
0087A2CC      83EE 01          sub esi,1
0087A2CF      0F85 87FFFFFF    jnz 0087A25C
//解码循环 第3个点
//Patch ☹、 E9 B67D0200      jmp 008A208A

```

Patch代码：

CODE: [Copy to clipboard]

```

008A208A      0F85 CC81FDFF    jnz 0087A25C
008A2090      C705 CFA28700 0F85>mov dword ptr ds:[87A2CF],FF87850F
008A209A      66:C705 D3A28700 F>mov word ptr ds:[87A2D3],0FFFF
//还原0087A2CF处修改的代码
008A20A3      C705 7FA38700 E937>mov dword ptr ds:[87A37F],27D37E9
008A20AD      66:C705 83A38700 0>mov word ptr ds:[87A383],0FF00
//Patch第4个点
008A20B6      E9 1A82FDFF      jmp 0087A2D5

```

3. Step 3:

```

0087A2C4 83EA 04 sub edx, 4
0087A2C7 BB E633514B mov ebx, 4B5133E6
0087A2CC 83EE 01 sub esi, 1
0087A2CF 0F85 87FFFFFF jnz 0087A25C
/ / Patch's
/ / 0087A2CF E9 B67D0200 jmp 008A208A

```

Cave 3:

```

008A208A 0F85 CC81FDFF jnz 0087A25C
008A2090 C705 CFA28700 0F85> mov dword ptr ds: [87A2CF], FF87850F

```



```

008A209A 66: C705 D3A28700 F> mov word ptr ds: [87A2D3], 0FFFF
008A20A3 C705 7FA38700 E937> mov dword ptr ds: [87A37F], 27D37E9
008A20AD 66: C705 83A38700 0> mov word ptr ds: [87A383], 0FF00
008A20B6 E9 1A82FDFF jmp 0087A2D5

```

4、第4个点

CODE: [Copy to clipboard]

```

0087A34D 81EF F17FF02D sub edi,2DF07FF1
0087A353 81F3 B6D34803 xor ebx,348D3B6
0087A359 66:BF 62DB mov di,0DB62
0087A35D 81C3 B7C8470B add ebx,0B47C8B7
0087A363 8918 mov dword ptr ds:[eax],ebx
0087A365 81D7 61475018 adc edi,18504761
0087A36B 81E8 47B71B45 sub eax,451BB747
0087A371 B1 80 mov cl,80
0087A373 81C0 43B71B45 add eax,451BB743
0087A379 BA 0C25D675 mov edx,75D6250C
0087A37E 4E dec esi
0087A37F 0F85 B4FFFFFF jnz 0087A339
//解码循环 第4个点
//Patch ④、 E9 377D0200 jmp 008A20BB

```

Patch代码：

CODE: [Copy to clipboard]

```

008A20BB 0F85 7882FDFF jnz 0087A339
//0087A37F代码挪这里执行
008A20C1 C705 7FA38700 0F85>mov dword ptr ds:[87A37F],FFB4850F
008A20CB 66:C705 83A38700 F>mov word ptr ds:[87A383],0FFFF
//还原0087A37F处修改的代码
008A20D4 C705 27A48700 E9C0>mov dword ptr ds:[87A427],27CC0E9
008A20DE 66:C705 2BA48700 0>mov word ptr ds:[87A42B],0FF00
//Patch第5个点
008A20E7 E9 9982FDFF jmp 0087A385

```

4. Step 4:

```

0087A34D 81EF F17FF02D sub edi, 2DF07FF1
0087A353 81F3 B6D34803 xor ebx, 348D3B6
0087A359 66: BF 62DB mov di, 0DB62
0087A35D 81C3 B7C8470B add ebx, 0B47C8B7
0087A363 8918 mov dword ptr ds: [eax], ebx
0087A365 81D7 ADC edi 61475018, 18504761
0087A36B 81E8 47B71B45 sub eax, 451BB747
0087A371 B1 80 mov cl, 80
0087A373 81C0 43B71B45 add eax, 451BB743
0087A379 THREE 0C25D675 mov edx, 75D6250C
0087A37E 4E dec esi
0087A37F 0F85 B4FFFFFF jnz 0087A339
/ / Skip to Cave 4
0087A37F E9 377D0200 jmp 008A20BB

```

Cave 4:

```

008A20BB 0F85 7882FDFF jnz 0087A339
008A20C1 C705 7FA38700 0F85> mov dword ptr ds: [87A37F], FFB4850F
008A20CB 66: C705 83A38700 F> mov word ptr ds: [87A383], 0FFFF
008A20D4 C705 27A48700 E9C0> mov dword ptr ds: [87A427], 27CC0E9
008A20DE 66: C705 2BA48700 0> mov word ptr ds: [87A42B], 0FF00
008A20E7 E9 9982FDFF jmp 0087A385

```

5. 第5个点

CODE: [Copy to clipboard]

```

0087A3F4 59 pop ecx
0087A3F5 5A pop edx
0087A3F6 81E8 9A2CB57B sub eax, 7BB52C9A
0087A3FC 50 push eax
0087A3FD 8F041F pop dword ptr ds:[edi+ebx]
0087A400 66:BE D174 mov si, 74D1
0087A404 66:BE 09ED mov si, 0ED09
0087A408 81EB 2FB9E712 sub ebx, 12E7B92F
0087A40E B9 7DD3FF76 mov ecx, 76FFD37D
0087A413 81C3 2BB9E712 add ebx, 12E7B92B
0087A419 0F8B 02000000 jpo 0087A421
0087A41F B2 72 mov dl, 72
0087A421 81FB D8FAFFFF cmp ebx, -528
0087A427 0F85 91FFFFFF jnz 0087A3BE
//解码循环 第5个点
//Patch ⑤、 E9 C07C0200 jmp 008A20EC

```

Patch代码:

CODE: [Copy to clipboard]

```

008A20EC 0F85 CC82FDFF jnz 0087A3BE
//0087A427代码挪这里执行
008A20F2 C705 27A48700 0F85>mov dword ptr ds:[87A427],FF91850F
008A20FC 66:C705 2BA48700 F>mov word ptr ds:[87A42B],0FFFF
//还原0087A427处修改的代码
008A2105 C705 97A58700 E981>mov dword ptr ds:[87A597],27B81E9
008A210F 66:C705 9BA58700 0>mov word ptr ds:[87A59B],0
//Patch第6个点
008A2118 E9 1083FDFF jmp 0087A42D

```

5. Step 5:

```

0087A3F4 59 pop ecx
0087A3F5 5A pop edx
0087A3F6 81E8 9A2CB57B sub eax, 7BB52C9A
0087A3FC 50 push eax
0087A3FD 8F041F pop dword ptr ds: [edi + ebx]
0087A400 66: BE D174 mov si, 74D1
0087A404 66: BE 09ED mov si, 0ED09
0087A408 81EB 2FB9E712 sub ebx, 12E7B92F
0087A40E B9 7DD3FF76 mov ecx, 76FFD37D
0087A413 81C3 2BB9E712 add ebx, 12E7B92B
0087A419 0F8B 02000000 jpo 0087A421
0087A41F B2 72 mov dl, 72
0087A421 81FB D8FAFFFF Cmp ebx, -528

```

```

0087A427 0F85 91FFFFFF jnz 0087A3BE
/ / Skip to Cave 5
0087A427 E9 C07C0200 jmp 008A20EC

```

Cave 5:

```

008A20EC 0F85 CC82FDFD jnz 0087A3BE
008A20F2 C705 27A48700 0F85> mov dword ptr ds:[87A427], FF91850F
008A20FC 66: C705 2BA48700 F> mov word ptr ds:[87A42B], 0FFFF
008A2105 C705 97A58700 E981> mov dword ptr ds:[87A597], 27B81E9
008A210F 66: C705 9BA58700 0> mov word ptr ds:[87A59B], 0
008A2118 E9 1083FDFD jmp 0087A42D

```

6、第6个点 获取壳代码重定位基址

CODE: [Copy to clipboard]

```

0087A517 FF95 F0030000 call dword ptr ss:[ebp+3F0] ; kernel32.VirtualAlloc
0087A51D 8985 31040000 mov dword ptr ss:[ebp+431],eax
//[ebp+431]=[0087A869]=01280000 壳代码基址
0087A523 8985 D0010000 mov dword ptr ss:[ebp+1D0],eax
//[ebp+1D0]=[0087A608]=01280000
0087A529 64:67:A1 0000 mov eax,dword ptr fs:[0]
0087A52E 8985 2D040000 mov dword ptr ss:[ebp+42D],eax
0087A534 8B55 5B mov edx,dword ptr ss:[ebp+5B]
0087A537 8B85 D0010000 mov eax,dword ptr ss:[ebp+1D0]
0087A53D 8902 mov dword ptr ds:[edx],eax
0087A53F 8B85 08040000 mov eax,dword ptr ss:[ebp+408]
0087A545 8942 04 mov dword ptr ds:[edx+4],eax
0087A548 8D85 9F030000 lea eax,dword ptr ss:[ebp+39F]
0087A54E 8B40 55 mov eax,dword ptr ds:[eax+55]
0087A551 8942 08 mov dword ptr ds:[edx+8],eax
0087A554 8B85 EC030000 mov eax,dword ptr ss:[ebp+3EC]
0087A55A 8942 10 mov dword ptr ds:[edx+10],eax
0087A55D 8B85 E8030000 mov eax,dword ptr ss:[ebp+3E8]
0087A563 8942 14 mov dword ptr ds:[edx+14],eax
0087A566 8B95 CC010000 mov edx,dword ptr ss:[ebp+1CC]
0087A56C BB F8010000 mov ebx,1F8
0087A571 8B7C1A 0C mov edi,dword ptr ds:[edx+ebx+C]
0087A575 0BFF or edi,edi
0087A577 74 1E je short 0087A597
0087A579 8B4C1A 10 mov ecx,dword ptr ds:[edx+ebx+10]
0087A57D 0BC9 or ecx,ecx
0087A57F 74 11 je short 0087A592

```



```

0087A581    03BD D0010000    add edi,dword ptr ss:[ebp+1D0]
0087A587    8B741A 14        mov esi,dword ptr ds:[edx+ebx+14]
0087A58B    03F2            add esi,edx
0087A58D    C1F9 02         sar ecx,2
0087A590    F3:A5          rep movs dword ptr es:[edi],dword ptr ds:[esi]
0087A592    83C3 28         add ebx,28
0087A595    EB DA          jmp short 0087A571
0087A597    8B85 CC010000   mov eax,dword ptr ss:[ebp+1CC]
//解码循环 第6个点
//Patch ⑥、 E9 817B0200    jmp 008A211D

```

Patch代码：

CODE: [Copy to clipboard]

```

008A211D    C705 97A58700 8B85>mov dword ptr ds:[87A597],1CC858B
008A2127    C605 9BA58700 00    mov byte ptr ds:[87A59B],0
//还原0087A597处修改的代码
008A212E    8B85 31040000    mov eax,dword ptr ss:[ebp+431]
//取得壳代码基址
008A2134    C780 ED100300 684C>mov dword ptr ds:[eax+310ED],8A214C68
008A213E    66:C780 F1100300 0>mov word ptr ds:[eax+310F1],0C300
//Patch第7个点 012B10ED
008A2147    E9 4B84FDFF      jmp 0087A577

```

6. Step 6: You remember the code contains functions VirtualAlloc not, we'll patch this code:

```

0087A517 FF95 F0030000 call dword ptr ss:[ebp+3 F0]; kernel32.VirtualAlloc
0087A51D 8985 31040000 mov dword ptr ss:[ebp+431], eax
/ / [ebp+431] = [0087A869] = 01280000
0087A523 8985 D0010000 ss mov dword ptr [ebp+1 D0], eax
/ / [ebp+1 D0] = [0087A608] = 01280000
0087A529 64:67: A1 0000 mov eax, dword ptr fs:[0]
0087A52E 8985 2D040000 mov dword ptr ss:[ebp+42 D], eax
0087A534 8B55 5B mov edx, dword ptr ss:[ebp+5 B]
0087A537 8B85 D0010000 mov eax, dword ptr ss:[ebp+1 D0]
0087A53D 8902 mov dword ptr ds:[edx], eax
0087A53F 8B85 08040000 mov eax, dword ptr ss:[ebp+408]
0087A545 8942 04 mov dword ptr ds:[edx+4], eax
0087A548 8D85 9F030000 lea eax, dword ptr ss:[ebp+39 F]
0087A54E 8B40 55 mov eax, dword ptr ds:[eax+55]
0087A551 8942 08 mov dword ptr ds:[edx+8], eax
0087A554 8B85 EC030000 mov eax, dword ptr ss:[ebp+3 EC]
0087A55A 8942 10 mov dword ptr ds:[edx+10], eax
0087A55D 8B85 E8030000 mov eax, dword ptr ss:[ebp+3 E8]
0087A563 8942 14 mov dword ptr ds:[edx+14], eax
0087A566 8B95 CC010000 mov edx, dword ptr ss:[ebp+1 CC]
0087A56C BB F8010000 mov ebx, 1F8
0087A571 8B7C1A 0C mov edi, dword ptr ds:[ebx+edx+C]
0087A575 0BFF or edi, edi
0087A577 74 1E je short 0087A597
0087A579 8B4C1A 10 mov ecx, dword ptr ds:[ebx+edx+10]
0087A57D 0BC9 or ecx, ecx
0087A57F 74 11 je short 0087A592
0087A581 03BD D0010000 add edi, dword ptr ss:[ebp+1 D0]
0087A587 8B741A 14 mov esi, dword ptr ds:[ebx+edx+14]
0087A58B 03F2 add esi, edx

```

```

0087A58D C1F9 02 SAR ecx, 2
0087A590 F3: A5 rep movs dword ptr es: [edi], dword ptr ds: [esi]
0087A592 83C3 28 add ebx, 28
EB DA 0087A595 jmp short 0087A571
0087A597 8B85 CC010000 mov eax, dword ptr ss: [ebp +1 CC]
/ / Skip to Cave 6:
0087A597 E9 817B0200 jmp 008A211D

```

Cave 6:

```

008A211D C705 97A58700 8B85> mov dword ptr ds: [87A597], 1CC858B
008A2127 C605 9BA58700 00 mov byte ptr ds: [87A59B], 0
008A212E 8B85 31040000 mov eax, dword ptr ss: [ebp +431]
008A2134 C780 ED100300 684C> mov dword ptr ds: [eax +310 ED], 8A214C68
008A213E 66: C780 F1100300 0> mov word ptr ds: [eax +310 F1], 0C300
008A2147 E9 4B84FDFF jmp 0087A597

```

7、第7个点

CODE: [Copy to clipboard]

```

012B10C4 FF95 79294400 call dword ptr ss:[ebp+442979]
012B10CA 8985 75294400 mov dword ptr ss:[ebp+442975],eax
012B10D0 8D9D 452A4400 lea ebx,dword ptr ss:[ebp+442A45]
012B10D6 50 push eax
012B10D7 53 push ebx
012B10D8 E8 74050000 call 012B1651
012B10DD 8BC8 mov ecx,eax
012B10DF 8DBD 452A4400 lea edi,dword ptr ss:[ebp+442A45]
012B10E5 8BB5 75294400 mov esi,dword ptr ss:[ebp+442975]
012B10EB F3:A4 rep movs byte ptr es:[edi],byte ptr ds:[esi]
012B10ED 8B85 75294400 mov eax,dword ptr ss:[ebp+442975]
//第7个点
//Patch ⑦、
012B10ED 68 4C218A00 push 8A214C
012B10F2 C3 retn

```

Patch代码：

CODE: [Copy to clipboard]

```

008A214C A1 69A88700 mov eax,dword ptr ds:[87A869]
//取得壳代码基址
008A2151 C780 ED100300 8B85>mov dword ptr ds:[eax+310ED],2975858B
008A215B 66:C780 F1100300 4>mov word ptr ds:[eax+310F1],44
//还原012B10ED处修改的代码
008A2164 C780 C2150300 687E>mov dword ptr ds:[eax+315C2],8A217E68
008A216E 66:C780 C6150300 0>mov word ptr ds:[eax+315C6],0C300
//Patch第8个点 012B15C2
008A2177 05 ED100300 add eax,310ED
008A217C FFEO jmp eax
//返回012B10ED处继续执行

```

7. Step 7:

```
012B10C4 FF95 79294400 call dword ptr ss: [ebp +442979]
```

```

012B10CA 8985 75294400 mov dword ptr ss: [ebp +442975], eax
012B10D0 8D9D 452A4400 Lea ebx, dword ptr ss: [ebp +442 A45]
012B10D6 50 push eax
012B10D7 53 push ebx
012B10D8 E8 74050000 call 012B1651
012B10DD 8BC8 mov ecx, eax
012B10DF 8DBD 452A4400 Lea edi, dword ptr ss: [ebp +442 A45]
012B10E5 8BB5 75294400 mov esi, dword ptr ss: [ebp +442975]
012B10EB F3: A4 rep movs byte ptr es: [edi], byte ptr ds: [esi]
012B10ED 8B85 75294400 mov eax, dword ptr ss: [ebp +442975]
/ / Patch's
012B10ED 68 4C218A00 push 8A214C
012B10F2 C3 retn

```

Cave 7:

```

008A214C A1 69A88700 mov eax, dword ptr ds: [87A869]
/ / Get the value of the code base address ASProtect
008A2151 C780 ED100300 8B85> mov dword ptr ds: [eax +310 ED], 2975858B
008A215B 66: C780 F1100300 4> mov word ptr ds: [eax +310 F1], 44
008A2164 C780 C2150300 687E> mov dword ptr ds: [eax +315 C2], 8A217E68
008A216E 66: C780 C6150300 0> mov word ptr ds: [eax +315 C6], 0C300
008A2177 05 ED100300 add eax, 310ED
008A217C FFE0 jmp eax

```

8、第8个点 CreateFileMappingA处理

CODE: [Copy to clipboard]

```

012B15C1      61                popad
012B15C2      75 08            jnz short 012B15CC
//第8个点
//Patch ⑧、
012B15C2      68 7E218A00      push 8A217E
012B15C7      C3              retn

```

Patch代码：

CODE: [Copy to clipboard]

```

008A217E      A1 69A88700      mov eax,dword ptr ds:[87A869]
//取得壳代码基址
008A2183      C780 C2150300 7508>mov dword ptr ds:[eax+315C2],1B80875
008A218D      66:C780 C6150300 0>mov word ptr ds:[eax+315C6],0
//还原012B15C2处修改的代码
008A2196      C680 37860100 08 mov byte ptr ds:[eax+18637],8
//修改01298636处的push 2为push 8 ★ CreateFileMappingA 参数
008A219D      C680 69860100 01 mov byte ptr ds:[eax+18669],1
//修改01298668处的push 4为push 1 ★ MapViewOfFileEx 参数
008A21A4      C780 7A860100 68BE>mov dword ptr ds:[eax+1867A],8A21BE68
008A21AE      66:C780 7E860100 0>mov word ptr ds:[eax+1867E],0C300
//Patch第9个点 0129867A
008A21B7      05 C2150300      add eax,315C2
008A21BC      FFE0            jmp eax
//返回012B15C2继续执行

```

8. Step 8:

You also remember our search for order in the POPAD not, belgium hours we will patch it in code:

```
012B15C1 61 popad
012B15C2 75 08 jnz short 012B15CC
/ / Patch's
012B15C2 68 7E218A00 push 8A217E
012B15C7 C3 retn
```

Cave 8:

```
008A217E A1 69A88700 mov eax, dword ptr ds: [87A869]
/ / Get the value of the code base address ASProtect
008A2183 C780 C2150300 7508> mov dword ptr ds: [eax +315 C2], 1B80875
008A218D 66: C780 C6150300 0> mov word ptr ds: [eax +315 C6], 0
008A2196 C680 37860100 08 mov byte ptr ds: [eax +18637], 8
/ / Patch Push Push 2 to 8
008A219D C680 69860100 01 mov byte ptr ds: [eax +18669], 1
/ / Patch Push Push 4 to 1
008A21A4 C780 7A860100 68BE> mov dword ptr ds: [eax +1867 A], 8A21BE68
008A21AE 66: C780 7E860100 0> mov word ptr ds: [eax +1867 e], 0C300
008A21B7 05 C2150300 add eax, 315C2
008A21BC FFE0 jmp eax
```

9、第9个点 自校验处理

CODE: [Copy to clipboard]

```
01298630      6A 00      push 0
01298632      6A 00      push 0
01298634      6A 00      push 0
01298636      6A 02      push 2
01298638      6A 00      push 0
0129863A      53          push ebx
0129863B      A1 E4972A01  mov eax,dword ptr ds:[12A97E4]
01298640      8B40 1C     mov eax,dword ptr ds:[eax+1C]
01298643      FFD0       call eax ; kernel32.CreateFileMappingA
01298645      A3 14B42A01  mov dword ptr ds:[12AB414],eax
0129864A      53          push ebx
0129864B      A1 E4972A01  mov eax,dword ptr ds:[12A97E4]
01298650      8B40 18     mov eax,dword ptr ds:[eax+18]
01298653      FFD0       call eax
01298655      833D 14B42A01 00  cmp dword ptr ds:[12AB414],0
0129865C      0F84 66040000  je 01298AC8
01298662      6A 00      push 0
01298664      6A 00      push 0
01298666      6A 00      push 0
01298668      6A 04      push 4
0129866A      A1 14B42A01  mov eax,dword ptr ds:[12AB414]
0129866F      50          push eax
01298670      A1 E4972A01  mov eax,dword ptr ds:[12A97E4]
01298675      8B40 08     mov eax,dword ptr ds:[eax+8]
01298678      FFD0       call eax ; kernel32.MapViewOfFileEx
0129867A      8BD8       mov ebx,eax
//第9个点
//Patch ㉔、
0129867A      68 BE218A00  push 8A21BE
0129867F      C3          retn
```

Patch代码：

CODE: [Copy to clipboard]

```

008A21BE      8B1D 69A88700      mov ebx,dword ptr ds:[87A869]
//取得壳代码基址
008A21C4      C683 37860100 02   mov byte ptr ds:[ebx+18637],2
//还原01298636处修改的代码
008A21CB      C683 69860100 04   mov byte ptr ds:[ebx+18669],4
//还原01298668处修改的代码
008A21D2      C783 7A860100 8BD8>mov dword ptr ds:[ebx+1867A],E850D88B
008A21DC      66:C783 7E860100 4>mov word ptr ds:[ebx+1867E],14A
//还原0129867A处修改的代码
008A21E5      C780 50010000 0020>mov dword ptr ds:[eax+150],4A2000
//EAX值是映像文件的开始地址，还原映像文件的SizeOfImage
008A21EF      66:C780 90030000 0>mov word ptr ds:[eax+390],1000
//还原映像文件最后一个区段的VSize为1000
008A21F8      66:C780 98030000 0>mov word ptr ds:[eax+398],0
//还原映像文件最后一个区段的RSize为0000
008A2201      C780 5A851900 0F85>mov dword ptr ds:[eax+19855A],FFAD850F
008A220B      66:C780 5E851900 F>mov word ptr ds:[eax+19855E],0FFFF
//还原映像文件中第一个接口修改的代码
008A2214      C783 B9CA0100 682F>mov dword ptr ds:[ebx+1CAB9],8A222F68
008A221E      66:C783 BDCA0100 0>mov word ptr ds:[ebx+1CABD],0C300
//Patch第10个点 0129CAB9
008A2227      81C3 7A860100      add ebx,1867A
008A222D      FFE3              jmp ebx
//返回0129867A继续执行

```

9. Step 9:

Now we patch the two functions in the function CreateFileMappingA MapViewOfFileEx:

```

01298630 6A 00 push 0
01298632 6A 00 push 0
01298634 6A 00 push 0
01298636 6A 02 push 2
01298638 6A 00 push 0
0129863A 53 push ebx
0129863B A1 E4972A01 mov eax, dword ptr ds: [12A97E4]
01298640 8B40 1C mov eax, dword ptr ds: [eax +1 C]
01298643 FFD0 call eax; kernel32.CreateFileMappingA
01298645 A3 14B42A01 mov dword ptr DS: [12AB414], eax
0129864A 53 push ebx
0129864B A1 E4972A01 mov eax, dword ptr ds: [12A97E4]
01298650 8B40 18 mov eax, dword ptr ds: [eax +18]
01298653 FFD0 call eax
01298655 833D 14B42A01 00 Cmp dword ptr ds: [12AB414], 0
0129865C 0F84 66040000 je 01298AC8
01298662 6A 00 push 0
01298664 6A 00 push 0
01298666 6A 00 push 0
01298668 6A 04 push 4
0129866A A1 14B42A01 mov eax, dword ptr ds: [12AB414]
0129866F 50 push eax
01298670 A1 E4972A01 mov eax, dword ptr ds: [12A97E4]
01298675 8B40 08 mov eax, dword ptr ds: [eax +8]
01298678 FFD0 call eax; kernel32.MapViewOfFileEx
0129867A 8BD8 mov ebx, eax

```

```

/ / Patch's
0129867A 68 BE218A00 push 8A21BE
0129867F C3 retn

```

Cave 9:

```

008A21BE 8B1D 69A88700 mov ebx, dword ptr ds: [87A869]
008A21C4 C683 37860100 02 mov byte ptr ds: [ebx +18637], 2
008A21CB C683 69860100 04 mov byte ptr ds: [ebx +18669], 4
008A21D2 C783 7A860100 8BD8> mov dword ptr ds: [ebx +1867 A], E850D88B
008A21DC 66: C783 7E860100 4> mov word ptr ds: [ebx +1867 e], 14A
008A21E5 C780 50010000 0020> mov dword ptr ds: [eax +150], 4A2000
/ / Return value EAX original SizeOfImage (deceive ASProtect)
008A21EF 66: C780 90,030,000 0> mov word ptr ds: [eax +390], 1000
/ / Return Vsize = 1000
008A21F8 66: C780 98,030,000 0> mov word ptr ds: [eax +398], 0
/ / Return Rsize = 1000
008A2201 C780 5A851900 0F85> mov dword ptr ds: [eax +19855 A], FFAD850F
008A220B 66: C780 5E851900 F> mov word ptr ds: [eax +19855 E], 0FFFF
008A2214 C783 B9CA0100 682F> mov dword ptr ds: [ebx +1 CAB9], 8A222F68
008A221E 66: C783 BDCA0100 0> mov word ptr ds: [ebx CABD +1], 0C300
008A2227 81C3 7A860100 add ebx, 1867A
008A222D FFE3 jmp ebx

```

10、第10个点 注册名Pre-Dip处理

CODE: [Copy to clipboard]

```

0129CAB9      50                push eax
//第10个点
0129CABA      8B47 04          mov eax,dword ptr ds:[edi+4]
0129CABD      FF00          call eax
//注册名Pre-Dip处理

//Patch 10、
0129CAB9      68 2F228A00      push 8A222F
0129CABE      C3                retn

```


Patch代码：

CODE: [Copy to clipboard]

```

008A222F      A1 69A88700      mov eax,dword ptr ds:[87A869]
008A2234      C780 B9CA0100 508B>mov dword ptr ds:[eax+1CAB9],4478B50
008A223E      66:C780 BDCA0100 F>mov word ptr ds:[eax+1CABD],0D0FF
//还原0129CAB9处修改的代码
008A2247      C680 84A50000 EB   mov byte ptr ds:[eax+A584],0EB
008A224E      C680 CAE10100 EB   mov byte ptr ds:[eax+1E1CA],0EB
008A2255      C780 67E20100 E989>mov dword ptr ds:[eax+1E267],89E9
008A225F      C605 B4204C00 01   mov byte ptr ds:[4C20B4],1
//修改程序中判断是否注册的标志位
008A2266      C705 1D466800 60DC>mov dword ptr ds:[68461D],21DC60
//Patch第11个点 0068461C
008A2270      68 04208A00      push 8A2004 ; ASCII "fly [2005.12.01]"
//Push 注册名地址,预先在8A2004处写下注册名
008A2275      A1 69A88700      mov eax,dword ptr ds:[87A869]
008A227A      05 BACA0100      add eax,1CABA
008A227F      FFE0             jmp eax
//返回0129CABA继续执行

```

10. Step 10:

This step is a step we put UserName to cave in to the table about, deceive with ASProtect that is valid UserName J!

```

0129CAB9 50 push eax
0129CABA 8B47 04 mov eax, dword ptr ds: [edi +4]
0129CABD FFD0 call eax
/ / Pre-Registration processing Dip

/ / Patch's
0129CAB9 68 2F228A00 push 8A222F
0129CABE C3 retn

```

Cave 10:

```

008A222F A1 69A88700 mov eax, dword ptr ds: [87A869]
008A2234 C780 B9CA0100 508B> mov dword ptr ds: [eax +1 CAB9], 4478B50
008A223E 66: C780 BDCA0100 F> mov word ptr ds: [eax +1 CABD], 0D0FF
008A2247 C680 EB 84A50000 mov byte ptr ds: [eax + A584], 0EB
008A224E C680 EB CAE10100 mov byte ptr ds: [eax +1 E1CA], 0EB
008A2255 C780 67E20100 E989> mov dword ptr ds: [eax +1 E267], 89E9
008A225F C605 B4204C00 01 mov byte ptr ds: [4C20B4], 1
008A2266 C705 1D466800 60DC> mov dword ptr ds: [68461D], 21DC60
008A2270 68 04208A00 push 8A2004; ASCII "fly [2005.12.01]"
/ / Push the string UserName
008A2275 A1 69A88700 mov eax, dword ptr ds: [87A869]
008A227A 05 BACA0100 add eax, 1CABA
008A227F FFE0 jmp eax

```

11. 第11个点 注册名显示

CODE: [Copy to clipboard]

```

00684616      8D45 E4          lea eax,dword ptr ss:[ebp-1C]
00684619      8B55 F8          mov edx,dword ptr ss:[ebp-8]
0068461C      E8 770DD8FF      call 00405398
//第11个点
//Patch 11.
0068461C      E8 60DC2100      call 008A2281

```

Patch代码:

CODE: [Copy to clipboard]

```

008A2281      BA 04208A00      mov edx,8A2004
//Patch 注册名地址,注意8A2004-4处应该是字符串长度
008A2286      E9 0D31B6FF      jmp 00405398
//继续流程

```

11. Step 11:

```

00684616 8D45 E4 Lea eax, dword ptr ss: [ebp-1C]
00684619 8B55 F8 mov edx, dword ptr ss: [ebp-8]
0068461C E8 770DD8FF call 00405398
/ / Patch's
0068461C E8 60DC2100 call 008A2281

```

Cave 11:

```

008A2281 THREE 04208A00 mov edx, 8A2004
/ / Push to record EDX value contains UserName
008A2286 E9 0D31B6FF jmp 00405398

```

VIII: The end!

In summary the entire code that we need to Inject Patch.exe file is:

```

008A2040 0F85 C780FDFD jnz 0087A10D
008A2046 C705 5AA18700 0F85ADFF mov dword ptr ds: [87A15A], FFAD850F
008A2050 66: C705 5EA18700 FFFF mov word ptr ds: [87A15E], 0FFFF
008A2059 C705 D8A18700 8C7E0200 mov dword ptr ds: [87A1D8], 27E8C
008A2063 E9 F880FDFD jmp 0087A160
008A2068 C705 D8A18700 3A000000 mov dword ptr ds: [87A1D8], 3A
008A2072 C705 CFA28700 E9B67D02 mov dword ptr ds: [87A2CF], 27DB6E9
008A207C 66: C705 D3A28700 00FF mov word ptr ds: [87A2D3], 0FF00
008A2085 E9 4D81FDFD jmp 0087A1D7
008A208A 0F85 CC81FDFD jnz 0087A25C
008A2090 C705 CFA28700 0F8587FF mov dword ptr ds: [87A2CF], FF87850F
008A209A 66: C705 D3A28700 FFFF mov word ptr ds: [87A2D3], 0FFFF
008A20A3 C705 7FA38700 E9377D02 mov dword ptr ds: [87A37F], 27D37E9
008A20AD 66: C705 83A38700 00FF mov word ptr ds: [87A383], 0FF00
008A20B6 E9 1A82FDFD jmp 0087A2D5
008A20BB 0F85 7882FDFD jnz 0087A339
008A20C1 C705 7FA38700 0F85B4FF mov dword ptr ds: [87A37F], FFB4850F

```

```

008A20CB 66: FFFF C705 83A38700 mov word ptr ds: [87A383], 0FFFF
008A20D4 C705 27A48700 E9C07C02 mov dword ptr ds: [87A427], 27CC0E9
008A20DE 66: C705 2BA48700 00FF mov word ptr ds: [87A42B], 0FF00
008A20E7 E9 9982FDFF jmp 0087A385
008A20EC 0F85 CC82FDFF jnz 0087A3BE
008A20F2 C705 27A48700 0F8591FF mov dword ptr ds: [87A427], FF91850F
008A20FC 66: C705 2BA48700 FFFF mov word ptr ds: [87A42B], 0FFFF
008A2105 C705 97A58700 E9817B02 mov dword ptr ds: [87A597], 27B81E9
008A210F 66: C705 9BA58700 0000 mov word ptr ds: [87A59B], 0
008A2118 E9 1083FDFF jmp 0087A42D
008A211D C705 97A58700 8B85CC01 mov dword ptr ds: [87A597], 1CC858B
008A2127 C605 9BA58700 00 mov byte ptr ds: [87A59B], 0
008A212E      8B85 31040000      mov eax,dword ptr ss:[ebp+431]
008A2134      C780 ED100300 684C218A      mov dword ptr ds:[eax+310ED],8A214C6>
008A213E      66:C780 F1100300 00C3      mov word ptr ds:[eax+310F1],0C300
008A2147      E9 4B84FDFF      jmp 0087A597
008A214C      A1 69A88700      mov eax,dword ptr ds:[87A869]
008A2151      C780 ED100300 8B857529      mov dword ptr ds:[eax+310ED],2975858>
008A215B      66:C780 F1100300 4400      mov word ptr ds:[eax+310F1],44
008A2164      C780 C2150300 687E218A      mov dword ptr ds:[eax+315C2],8A217E6>
008A216E      66:C780 C6150300 00C3      mov word ptr ds:[eax+315C6],0C300
008A2177      05 ED100300      add eax,310ED
008A217C      FFE0      jmp eax
008A217E      A1 69A88700      mov eax,dword ptr ds:[87A869]
008A2183      C780 C2150300 7508B801      mov dword ptr ds:[eax+315C2],1B80875
008A218D      66:C780 C6150300 0000      mov word ptr ds:[eax+315C6],0
008A2196      C680 37860100 08      mov byte ptr ds:[eax+18637],8
008A219D      C680 69860100 01      mov byte ptr ds:[eax+18669],1
008A21A4      C780 7A860100 68BE218A      mov dword ptr ds:[eax+1867A],8A21BE6>
008A21AE      66:C780 7E860100 00C3      mov word ptr ds:[eax+1867E],0C300
008A21B7      05 C2150300      add eax,315C2
008A21BC      FFE0      jmp eax
008A21BE      8B1D 69A88700      mov ebx,dword ptr ds:[87A869]
008A21C4      C683 37860100 02      mov byte ptr ds:[ebx+18637],2
008A21CB      C683 69860100 04      mov byte ptr ds:[ebx+18669],4
008A21D2      C783 7A860100 8BD850E8      mov dword ptr ds:[ebx+1867A],E850D88>
008A21DC      66:C783 7E860100 4A01      mov word ptr ds:[ebx+1867E],14A
008A21E5      C780 50010000 00204A00      mov dword ptr ds:[eax+150],4A2000
008A21EF      66:C780 90030000 0010      mov word ptr ds:[eax+390],1000
008A21F8      66:C780 98030000 0000      mov word ptr ds:[eax+398],0
008A2201      C780 5A851900 0F85ADFF      mov dword ptr ds:[eax+19855A],FFAD85>
008A220B      66:C780 5E851900 FFFF      mov word ptr ds:[eax+19855E],0FFFF
008A2214      C783 B9CA0100 682F228A      mov dword ptr ds:[ebx+1CAB9],8A222F6>
008A221E      66:C783 BDCA0100 00C3      mov word ptr ds:[ebx+1CABD],0C300
008A2227      81C3 7A860100      add ebx,1867A
008A222D      FFE3      jmp ebx
008A222F      A1 69A88700      mov eax,dword ptr ds:[87A869]
008A2234      C780 B9CA0100 508B4704      mov dword ptr ds:[eax+1CAB9],4478B50
008A223E      66:C780 BDCA0100 FFD0      mov word ptr ds:[eax+1CABD],0D0FF
008A2247      C680 84A50000 EB      mov byte ptr ds:[eax+A584],0EB
008A224E      C680 CAE10100 EB      mov byte ptr ds:[eax+1E1CA],0EB
008A2255      C780 67E20100 E9890000      mov dword ptr ds:[eax+1E267],89E9
008A225F      C605 B4204C00 01      mov byte ptr ds:[4C20B4],1
008A2266      C705 1D466800 60DC2100      mov dword ptr ds:[68461D],21DC60
008A2270      68 04208A00      push 8A2004
008A2275      A1 69A88700      mov eax,dword ptr ds:[87A869]
008A227A      05 BACA0100      add eax,1CABA
008A227F      FFE0      jmp eax
008A2281      BA 04208A00      mov edx,8A2004
008A2286      E9 0D31B6FF      jmp 00405398

```


CODE: [Copy to clipboard]

```

10 00 00 00 66 6C 79 20 5B 32 30 30 35 2E 31 32 2E 30 31 5D 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0F 85 C7 80 FD FF C7 05 5A A1 87 00 0F 85 AD FF 66 C7 05 5E A1 87 00 FF FF C7 05 D8 A1 87 00 8C
7E 02 00 E9 F8 80 FD FF C7 05 D8 A1 87 00 3A 00 00 00 C7 05 CF A2 87 00 E9 B6 7D 02 66 C7 05 D3
A2 87 00 00 FF E9 4D 81 FD FF 0F 85 CC 81 FD FF C7 05 CF A2 87 00 0F 85 87 FF 66 C7 05 D3 A2 87
00 FF FF C7 05 7F A3 87 00 E9 37 7D 02 66 C7 05 83 A3 87 00 00 FF E9 1A 82 FD FF 0F 85 78 82 FD
FF C7 05 7F A3 87 00 0F 85 B4 FF 66 C7 05 83 A3 87 00 FF FF C7 05 27 A4 87 00 E9 C0 7C 02 66 C7
05 2B A4 87 00 00 FF E9 99 82 FD FF 0F 85 CC 82 FD FF C7 05 27 A4 87 00 0F 85 91 FF 66 C7 05 2B
A4 87 00 FF FF C7 05 97 A5 87 00 E9 81 7B 02 66 C7 05 9B A5 87 00 00 00 E9 10 83 FD FF C7 05 97
A5 87 00 8B 85 CC 01 C6 05 9B A5 87 00 00 8B 85 31 04 00 00 C7 80 ED 10 03 00 68 4C 21 8A 66 C7
80 F1 10 03 00 00 C3 E9 4B 84 FD FF A1 69 A8 87 00 C7 80 ED 10 03 00 8B 85 75 29 66 C7 80 F1 10
03 00 44 00 C7 80 C2 15 03 00 68 7E 21 8A 66 C7 80 C6 15 03 00 00 C3 05 ED 10 03 00 FF E0 A1 69
A8 87 00 C7 80 C2 15 03 00 75 08 B8 01 66 C7 80 C6 15 03 00 00 00 C6 80 37 86 01 00 08 C6 80 69
86 01 00 01 C7 80 7A 86 01 00 68 BE 21 8A 66 C7 80 7E 86 01 00 00 C3 05 C2 15 03 00 FF E0 8B 1D
69 A8 87 00 C6 83 37 86 01 00 02 C6 83 69 86 01 00 04 C7 83 7A 86 01 00 8B D8 50 E8 66 C7 83 7E
86 01 00 4A 01 C7 80 50 01 00 00 00 20 4A 00 66 C7 80 90 03 00 00 00 10 66 C7 80 98 03 00 00 00
00 C7 80 5A 85 19 00 0F 85 AD FF 66 C7 80 5E 85 19 00 FF FF C7 83 B9 CA 01 00 68 2F 22 8A 66 C7
83 BD CA 01 00 00 C3 81 C3 7A 86 01 00 FF E3 A1 69 A8 87 00 C7 80 B9 CA 01 00 50 8B 47 04 66 C7
80 BD CA 01 00 FF D0 C6 80 84 A5 00 00 EB C6 80 CA E1 01 00 EB C7 80 67 E2 01 00 E9 89 00 00 C6
05 B4 20 4C 00 01 C7 05 1D 46 68 00 60 DC 21 00 68 04 20 8A 00 A1 69 A8 87 00 05 BA CA 01 00 FF
E0 BA 04 20 8A 00 E9 0D 31 B6 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Run thử Patch.exe:



_So easy, huh!

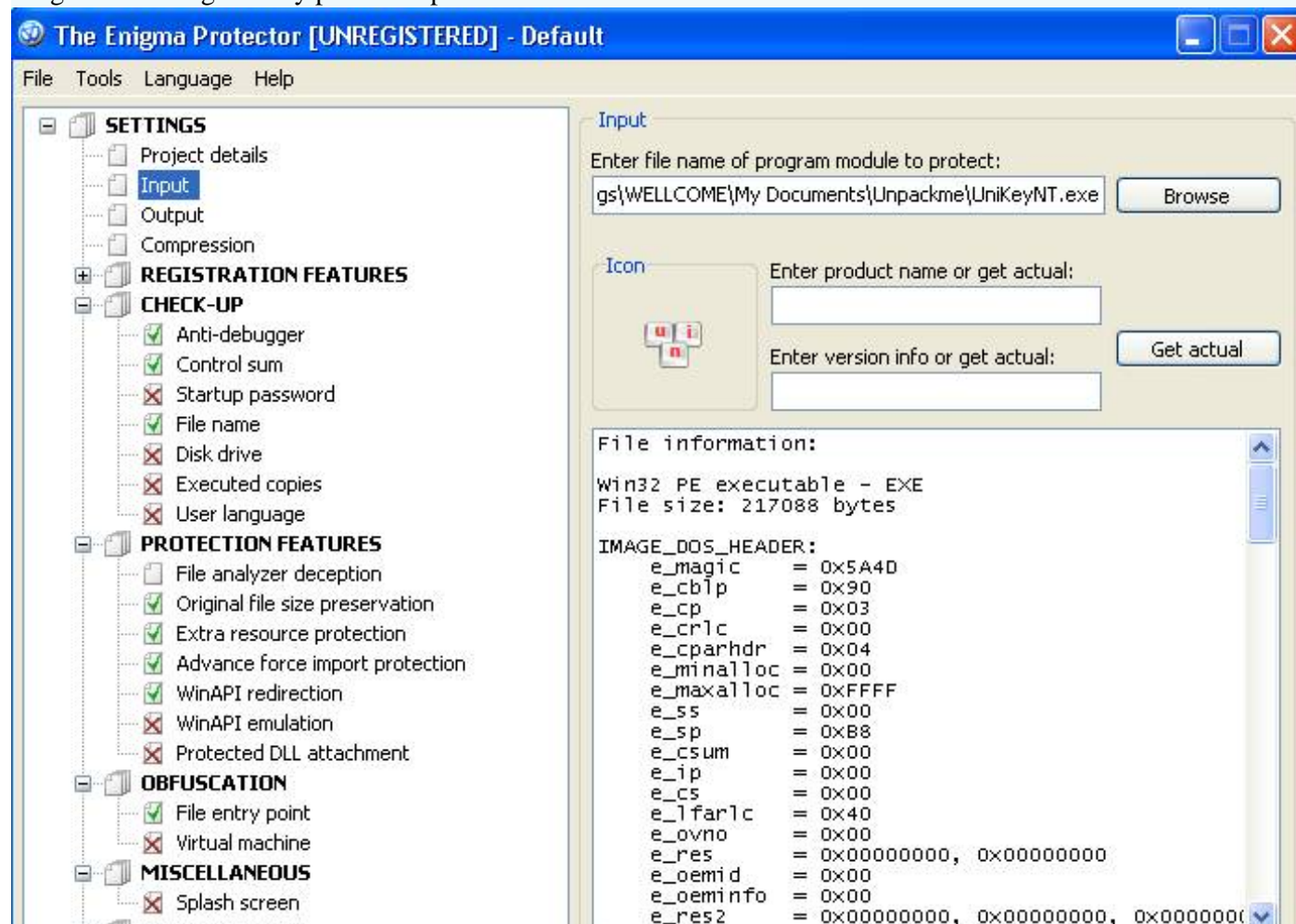
Welcome brothers gambling, then they have no long has tuts. But today because chán the sad life and the end of the week, so they try to take some time. Write a mini-tut 1 for reading the game. In fact this is not necessarily called tut, I just write as you understand them all, they should not go into the analysis. She analyzes some options of Enigma Protector lost 1 evening, so time is very urgent, they should also not have time to check kī. To understand this for good, require you to have:

- 1) Have knowledge of unpack
- 2) Once the packer unpack relatively such as armadillo
- 3) More and more : D

The Protector Egnima 1:33 - Mini-tut

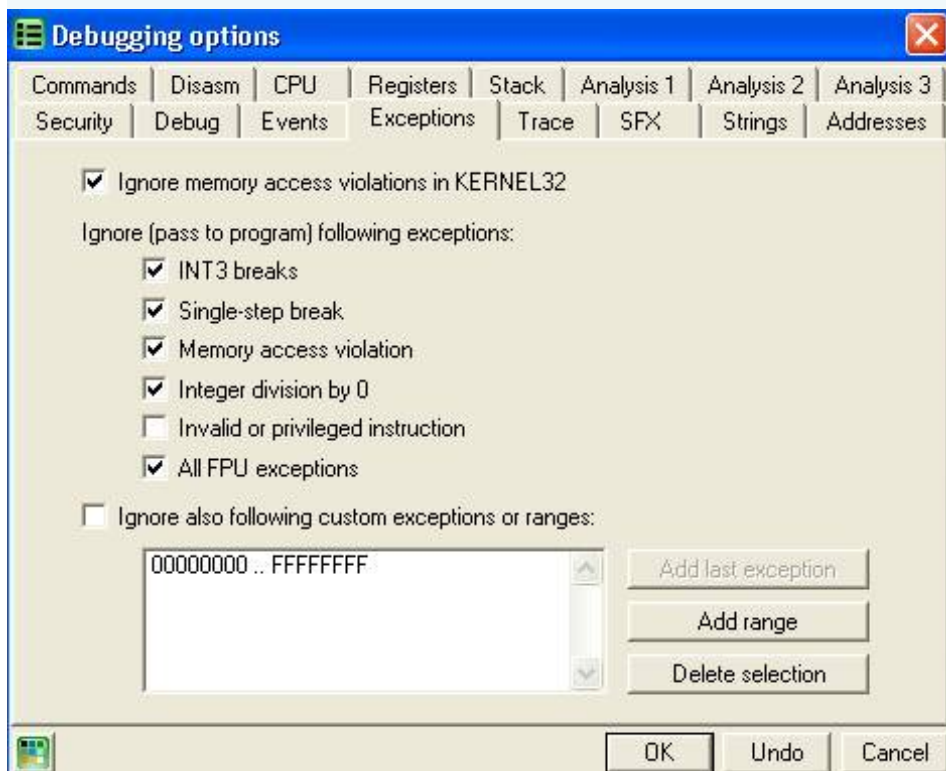
Tools: Olly, LordPE, ImportREC ... as well as new mod: D

Target the message is they pack the option as follows:





Olly on target load (you have to go hide Olly small: D). Press Shift-F9 to nag springiness, then to config options as follows:



Config like Olly will break when having exception Illegal instruction. Do not understand the children's Olly kī not hide or mechanism by antidebug Enigma Protector, but when they target springiness nag try set memory BP is the crash. Configuring such a stop after the exception, we can freely set the memory bp without fear crash were: D.

After clicking nag config Olly to stop at the exception, we will here

```

00A3D77A 0F0B ud2
00A3D77C E8 8B4AFFFF call 00A3220C
00A3D781 33C0 xor eax, eax
00A3D783 5A pop edx
00A3D784 59 pop ecx
00A3D785 59 pop ecx

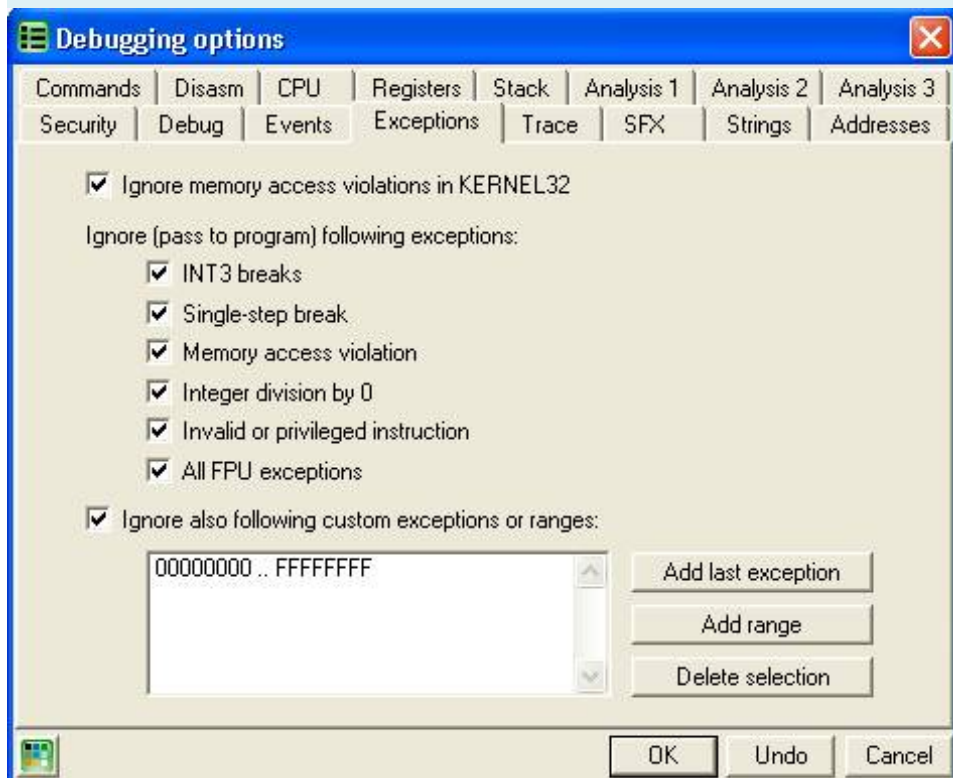
```



```
00A3D786 64:8910 mov dword ptr fs: [eax], edx
00A3D789 EB 0A jmp short 00A3D795
00A3D78B ^ E9 E85FFEFF jmp 00A23778
00A3D790 E8 4B63FEFF call 00A23AE0
```

Press Shift-F9 and count the number of clicks before the target run, do not understand each restart, every time different. There are time-shift F9 12 times, have at 13, 18, 19 Here they will press Shift-F9 10 times: D. After shift-F9 10 times the

Olly the config as follows:



Now we will set the memory bp section text (note that if some doctors have open Unikey then go off the house)

```

00A228DB F3: A5 rep movs dword ptr es: [edi], dword p>
00A228DD 89C1 mov ecx, eax
00A228DF 83E1 03 and ecx, 3
00A228E2 F3: A4 rep movs byte ptr es: [edi], byte ptr>
00A228E4 5F pop edi
00A228E5 5e pop esi
00A228E6 C3 retn
FC Lea 00A228E7 8D7431 esi, dword ptr [ecx + esi-4]
FC Lea 00A228EB 8D7C39 edi, dword ptr [ecx + edi-4]
00A228EF C1F9 02 SAR ecx, 2
00A228F2 78 11 js short 00A22905
FD 00A228F4 std
00A228F5 F3: A5 rep movs dword ptr es: [edi], dword p>
00A228F7 89C1 mov ecx, eax
00A228F9 83E1 03 and ecx, 3

```

```

00A228FC 83C6 03 add esi, 3
00A228FF 83C7 03 add edi, 3
00A22902 F3: A4 rep movs byte ptr es: [edi], byte ptr>
00A22904 FC cld
00A22905 5F pop edi
00A22906 5e pop esi
00A22907 C3 retn

```

Remove memory BP, BP set at **00A22907 C3 retn** and Shift-F9
Trace to here:

```

00AAAB92 A1 7014AD00 mov eax, dword ptr [AD1470]
00AAAB97 E8 507BF7FF call 00A226EC
00AAAB9C C643 0D 01 mov byte ptr [ebx + D], 1
00AAABA0 A1 5C60AB00 mov eax, dword ptr [AB605C]
00AAABA5 50 push eax
00AAABA6 A1 C862AB00 mov eax, dword ptr [AB62C8]
00AAABAB 50 push eax
00AAABAC E8 3FB5F7FF call 00A260F0
00AAABB1 8B0D E060AB00 mov ecx, dword ptr [AB60E0]
00AAABB7 8B15 5C60AB00 mov edx, dword ptr [AB605C]
00AAABBD 8B12 mov edx, dword ptr [edx]
00AAABBF A1 C862AB00 mov eax, dword ptr [AB62C8]
00AAABC4 E8 0B89F8FF call 00A334D4
... ..
00AAB0CD E8 DE5BF9FF call 00A40CB0
00AAB0D2 A1 EC62AB00 mov eax, dword ptr [AB62EC]
00AAB0D7 80B8 B0260000 0> Cmp byte ptr [eax +26 B0], 0
00AAB0DE 74 07 je short 00AAB0E7
00AAB0E0 E8 0F59F9FF call 00A409F4
00AAB0E5 EB 05 jmp short 00AAB0EC
00AAB0E7 E8 CC59F9FF call 00A40AB8
00AAB0EC C643 24 01 mov byte ptr [ebx +24], 1
00AAB0F0 E8 17A1F9FF call 00A4520C
00AAB0F5 C643 25 01 mov byte ptr [ebx +25], 1
00AAB0F9 E8 3246F9FF call 00A3F730
00AAB0FE C643 26 01 mov byte ptr [ebx +26], 1
00AAB102 E8 4990F9FF call 00A44150
00AAB107 C643 27 01 mov byte ptr [ebx +27], 1
00AAB10B E8 E86FF8FF call 00A320F8
00AAB110 C643 28 01 mov byte ptr [ebx +28], 1
00AAB114 E8 5B6FF8FF call 00A32074

```



```

00AAB119 C643 29 01 mov byte ptr [ebx +29], 1
00AAB11D E8 8AA0F9FF call 00A451AC
00AAB122 C643 2A 01 mov byte ptr [ebx +2 A], 1
00AAB126 E8 19A4F9FF call 00A45544
00AAB12B E8 846FF9FF call 00A420B4
00AAB130 33C0 xor eax, eax
... ..

```

Scroll down to see signs of this:

00AAB146	004B B1	add	byte ptr [ebx-4F], cl	
00AAB149	AA	stos	byte ptr es:[edi]	
00AAB14A	0089 C38D95E4	add	byte ptr [ecx+E4958DC3], cl	
00AAB150	FE	???		Unknown command
00AAB151	FFFF	???		Unknown command
00AAB153	8B03	mov	eax, dword ptr [ebx]	
00AAB155	E8 CE80F7FF	call	00A23228	
00AAB15A	8D95 E4FEFFFF	lea	edx, dword ptr [ebp-11C]	
00AAB160	8D45 E4	lea	eax, dword ptr [ebp-1C]	
00AAB163	E8 AC91F7FF	call	00A24314	
00AAB168	FF75 E4	push	dword ptr [ebp-1C]	
00AAB16B	68 98B3AA00	push	0AAB398	
00AAB170	FF73 04	push	dword ptr [ebx+4]	
00AAB173	8D45 E8	lea	eax, dword ptr [ebp-18]	
00AAB176	BA 03000000	mov	edx, 3	
00AAB17B	E8 B092F7FF	call	00A24430	
00AAB180	8B45 E8	mov	eax, dword ptr [ebp-18]	
00AAB183	E8 348FF9FF	call	00A440BC	
00AAB188	E8 5389F7FF	call	00A23AE0	
00AAB18D	E8 3E25F9FF	call	00A3D6D0	
00AAB192	E8 4D2DF9FF	call	00A3DEE4	
00AAB197	50	push	eax	
00AAB198	89C1	mov	ecx, eax	
00AAB19A	B8 D8B1AA00	mov	eax, 0AAB1D8	
00AAB19F	E8 142DF9FF	call	00A3DEB8	
00AAB1A4	010424	add	dword ptr [esp], eax	
00AAB1A7	C3	retn		
00AAB1A8	C8 548F16	enter	8F54, 16	
00AAB1AC	F5	cmc		
00AAB1AD	C8 B57254	enter	72B5, 54	
00AAB1B1	1A3A	sbb	bh, byte ptr [edx]	

Now we put in hardware bp **00AAB1A7 C3 retn**

Then Shift-F9, dom through stack, we see this, and remember it

0006FE50	00AAB1D8	
0006FE54	0006FFE0	Pointer to next SEH record
0006FE58	00AAB34F	SE handler
0006FE5C	0006FF88	
0006FE60	00AD2154	

Now the memory map

00400000	00001000	UnikeyNT		PE header	Imag	R	RWE
00401000	00019000	UnikeyNT					
0041A000	00007000	UnikeyNT					
00421000	00032000	UnikeyNT				Enter	
00453000	00010000	UnikeyNT					
00463000	0000E000	UnikeyNT	.rsrc				
00471000	00062000	UnikeyNT	.data			Ctrl+B	
004E0000	00103000					Ctrl+L	
005F0000	000F7000						
008F0000	00023000					F2	
00920000	00010000						
00A20000	000D4000						
00B00000	00004000						
00B20000	00020000						

- Actualize
- View in Disassembler Enter
- Dump in CPU
- Dump
- Search Ctrl+B
- Search next Ctrl+L
- Set break-on-access F2
- Set memory breakpoint on access
- Set memory breakpoint on write
- Set access ►

Press Shift-F9 to 1, we stop here

```

004114E0 68 34154100 push 00411534
004114E5 64: A1 00000000 mov eax, dword ptr fs: [0]
004114EB 50 push eax
004114EC 8B4424 10 mov eax, dword ptr [esp +10]
004114F0 896C24 10 mov dword ptr [esp +10], ebp
004114F4 8D6C24 10 lea ebp, dword ptr [esp +10]
004114F8 2BE0 sub esp, eax
004114FA 53 push ebx
004114FB 56 push esi
004114FC 57 push edi
004114FD 8B45 F8 mov eax, dword ptr [ebp-8]
00411500 8965 E8 mov dword ptr [ebp-18], esp
00411503 50 push eax
8B45 FC 00411504 mov eax, dword ptr [ebp-4]
C745 FC 00411507 FFFFFFFF> mov dword ptr [ebp-4], -1
0041150E 8945 F8 mov dword ptr [ebp-8], eax
00411511 8D45 F0 lea eax, dword ptr [ebp-10]
00411514 64: A3 00000000 mov dword ptr fs: [0], eax
0041151A C3 retn

```

Perhaps you also know, this is not the OEP, OEP because of VC GetVersion often function at the top. BP then remove memory trace to return from the code above, we come:

```
00B4C004 6A 60 push 60
00B4C006 68 60DA4100 push 41DA60
00B4C00B E8 D0548CFF call UnikeyNT.004114E0
00B4C010 BF 94000000 mov edi, 94
00B4C015 8BC7 mov eax, edi
00B4C017 E8 741E8CFF call UnikeyNT.0040DE90
00B4C01C 89A5 E8FFFFFF mov dword ptr [ebp-18], esp
00B4C022 8BF4 mov esi, esp
00B4C024 893E mov dword ptr [esi], edi
00B4C026 56 push esi
00B4C027 FF15 D0A3B300 call dword ptr [B3A3D0]
00B4C02D 8B4E 10 mov ecx, dword ptr [esi +10]
00B4C030 890D 3C0A4500 mov dword ptr [450A3C], ecx
00B4C036 8B46 04 mov eax, dword ptr [esi +4]
00B4C039 8905 480A4500 mov dword ptr [450A48], eax
00B4C03F 8B56 08 mov edx, dword ptr [esi +8]
00B4C042 8915 4C0A4500 mov dword ptr [450A4C], edx
00B4C048 8B76 0C mov esi, dword ptr [esi + C]
00B4C04B 81E6 FF7F0000 and esi, 7FFF
00B4C051 8935 400A4500 mov dword ptr [450A40], esi
Cmp 00B4C057 81F9 02000000 ecx, 2
00B4C05D 0F84 0C000000 je 00B4C06F
```

.....

We realize that OEP code in this area belong to small Egnima Protector created, and **00B4C027 FF15 D0A3B300 call dword ptr [B3A3D0]** is a function GetVersion .. This is the OEP stolen. OK, find out how a previous. Now Ctrl-F2 to restart target.

I. WinAPI redirection

Do the same for this

Remove memory BP, BP set at **00A22907 C3 retn** then Shift-F9

Trace to here:

```
00AAAB92 A1 7014AD00 mov eax, dword ptr [AD1470]
00AAAB97 E8 507BF7FF call 00A226EC
00AAAB9C C643 0D 01 mov byte ptr [ebx + D], 1
00AAABA0 A1 5C60AB00 mov eax, dword ptr [AB605C]
00AAABA5 50 push eax
```



```

00AAABA6 A1 C862AB00 mov eax, dword ptr [AB62C8]
00AAABAB 50 push eax
00AAABAC E8 3FB5F7FF call 00A260F0
00AAABB1 8B0D E060AB00 mov ecx, dword ptr [AB60E0]
00AAABB7 8B15 5C60AB00 mov edx, dword ptr [AB605C]
00AAABBD 8B12 mov edx, dword ptr [edx]
00AAABBF A1 C862AB00 mov eax, dword ptr [AB62C8]
00AAABC4 E8 0B89F8FF call 00A334D4

```

In the analysis Enigma Protector, they found 4 API not hide. They are:

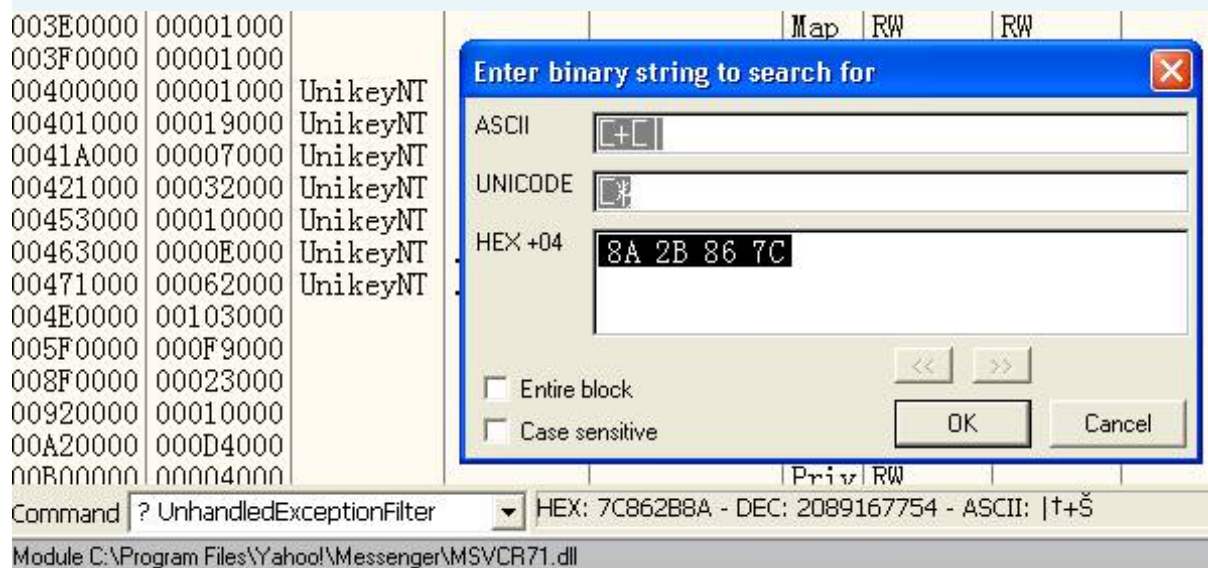
kernel32.UnhandledExceptionFilter

kernel32.DebugBreak

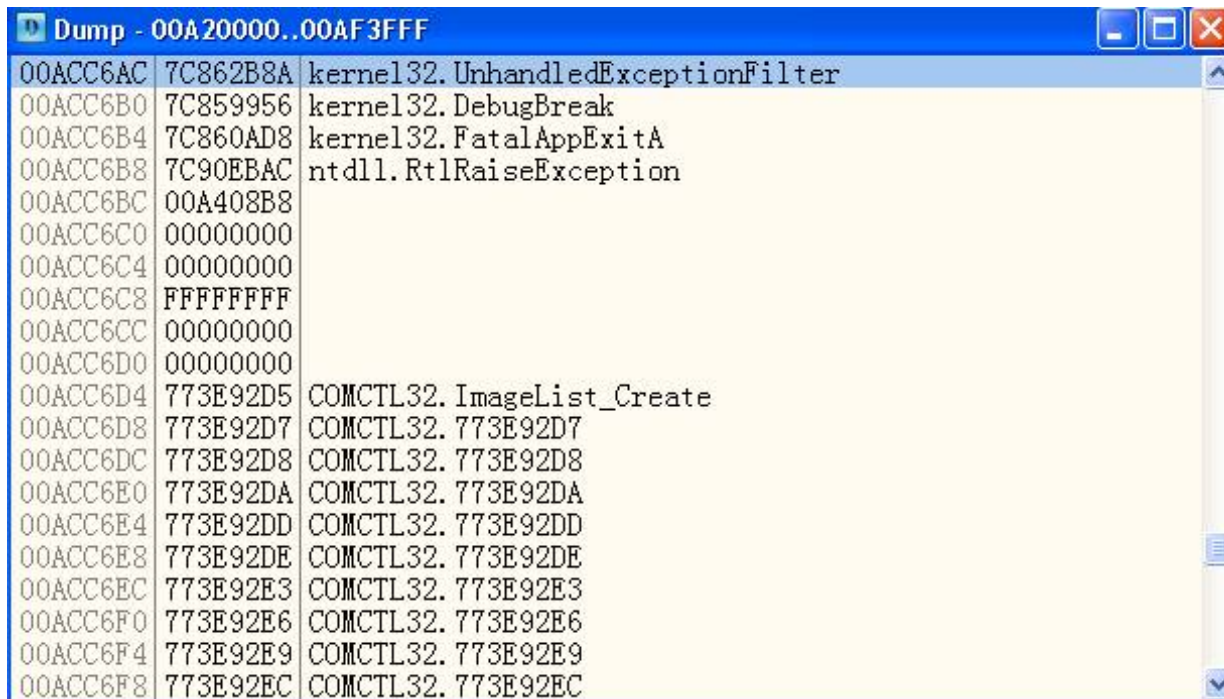
kernel32.FatalAppExitA

ntdll.RtlRaiseException

OK, we will map to memory for this function



Select a view of Long => address to see clearly for



Press Ctrl-R and then select the first line in 00A20000 References 00AF3FFF to 00ACC6AC ... 00ACC6AC, 0 item

Address = 00A40432

Disassembly = Cmp eax, dword ptr [ACC6AC]

Comments = kernel32.UnhandledExceptionFilter

We came here

```

00A40430 3D2 xor edx, edx
00A40432 3B05 ACC6AC00 Cmp eax, dword ptr [ACC6AC]; kernel32.UnhandledExceptionFilter
00A40438 74 18 je short 00A40452
00A4043A 3B05 B0C6AC00 Cmp eax, dword ptr [ACC6B0]; kernel32.DebugBreak
00A40440 74 10 je short 00A40452
00A40442 3B05 B8C6AC00 Cmp eax, dword ptr [ACC6B8]; ntdll.RtlRaiseException
00A40448 74 08 je short 00A40452
00A4044A 3B05 B4C6AC00 Cmp eax, dword ptr [ACC6B4]; kernel32.FatalAppExitA
00A40450 75 02 jnz short 00A40454
00A40452 B2 01 mov dl, 1
00A40454 8BC2 mov eax, edx
00A40456 C3 retn

```

Here, it is easily found if `eax = 1`, the API will not hide. So we just patch 1 billion that it is finished. **00A40450**

Patch / **75 02 jnz short 00A40454**

to nop

Done press Shift-F9, remember we are still in hardware bp **00AAB1A7 C3 ret**

Dom through the window stack considered

0006FE50	4FC69842	
0006FE54	0006FFE0	Pointer to next SEH record
0006FE58	00AAB34F	SE handler
0006FE5C	0006FF88	
0006FE60	00AD2154	

Nè so what? I do have to address as the return nhi;). No problem, the doctors have in mind when they first said that some doctors note that this does not address, in a machine they 00AAB1D8. Now we will modify to help

Olly return to the right place: D

0006FE50	00AAB1D8	
0006FE54	0006FFE0	Pointer to next SEH record
0006FE58	00AAB34F	SE handler
0006FE5C	0006FF88	
0006FE60	00AD2154	

Done set memory bp section on text -> Shift-F9, we will stop at the old place: D

```
004114E0 68 34154100 push 00411534
004114E5 64: A1 00000000 mov eax, dword ptr fs: [0]
004114EB 50 push eax
004114EC 8B4424 10 mov eax, dword ptr [esp +10]
004114F0 896C24 10 mov dword ptr [esp +10], ebp
004114F4 8D6C24 10 lea ebp, dword ptr [esp +10]
... ..
```

I. Stolen OEP

Remove memory BP -> trace through the code that we stop here

```
00B5B314 6A 60 push 60
00B5B316 68 60DA4100 push 41DA60
00B5B31B E8 C0618BFF call UnikeyNT.004114E0
00B5B320 BF 94000000 mov edi, 94
00B5B325 8BC7 mov eax, edi
00B5B327 E8 642B8BFF call UnikeyNT.0040DE90
00B5B32C 89A5 E8FFFFFF mov dword ptr [ebp-18], esp
00B5B332 8BF4 mov esi, esp
00B5B334 893E mov dword ptr [esi], edi
```



```

00B5B336 56 push esi
00B5B337 FF15 0032B300 call dword ptr [B33200]; kernel32.GetVersionExA
00B5B33D 8B4E 10 mov ecx, dword ptr [esi +10]
00B5B340 890D 3C0A4500 mov dword ptr [450A3C], ecx

```

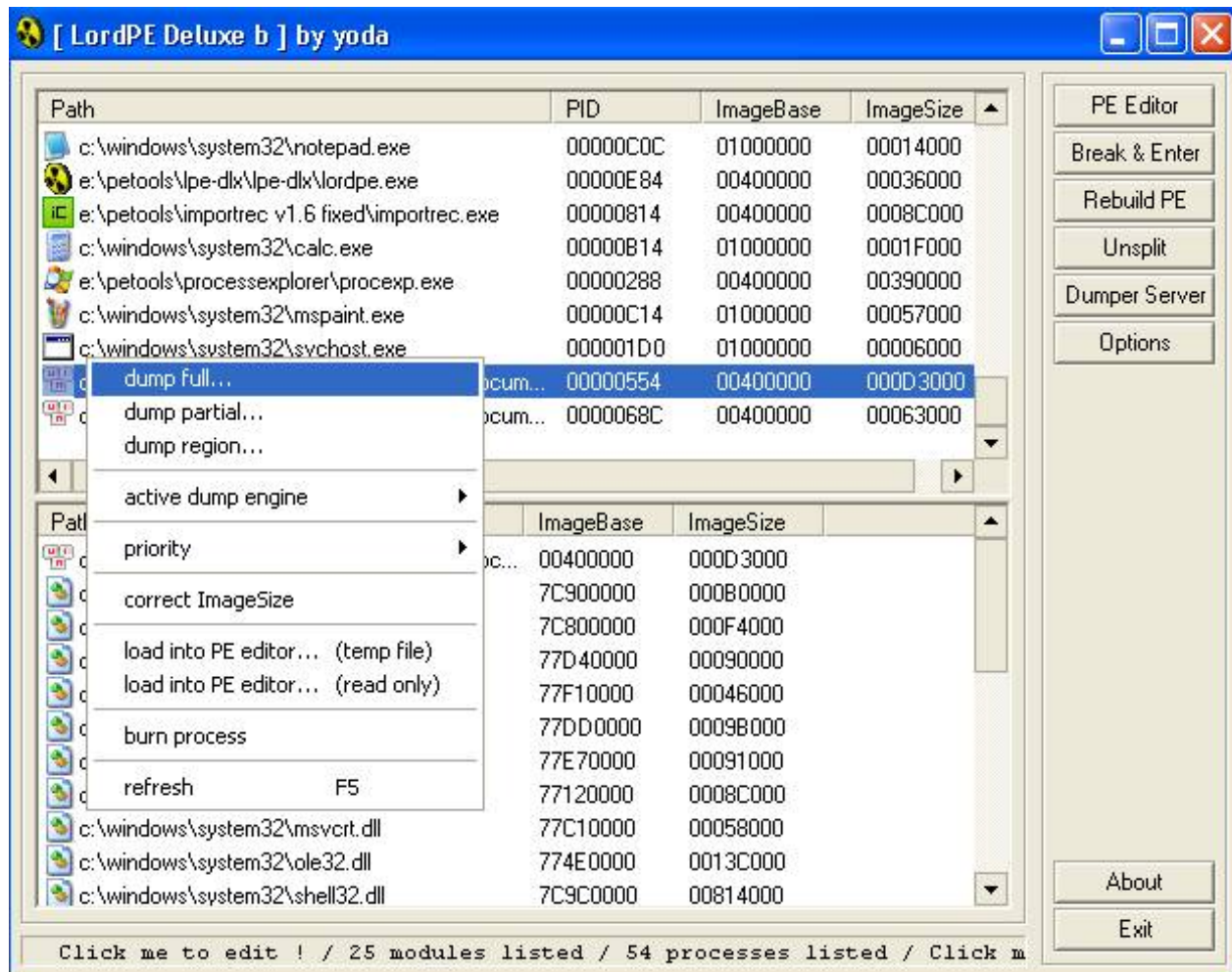
... ..

Now we see that then GetVersion function. We also do not need to address concerns that contains the original OEP, Goto text section (401,000), scroll down to find the code cave. OK, I will start from address **0041971F 0000 add byte ptr [eax], al**

Copy the entire code at the OEP to 0041971F. Note few medical doctors that if they use the Copy Binary Call will function very wrong address, this is what doctors can fix your hands.
Copy the selected finished **0041971F 6A 60 push 60** and then "New origin here"

II.Fix dump

Now conducted only full dump, where they used LordPE



Then Fix IAT. If we use the IAT AutoSearch will notice the following:



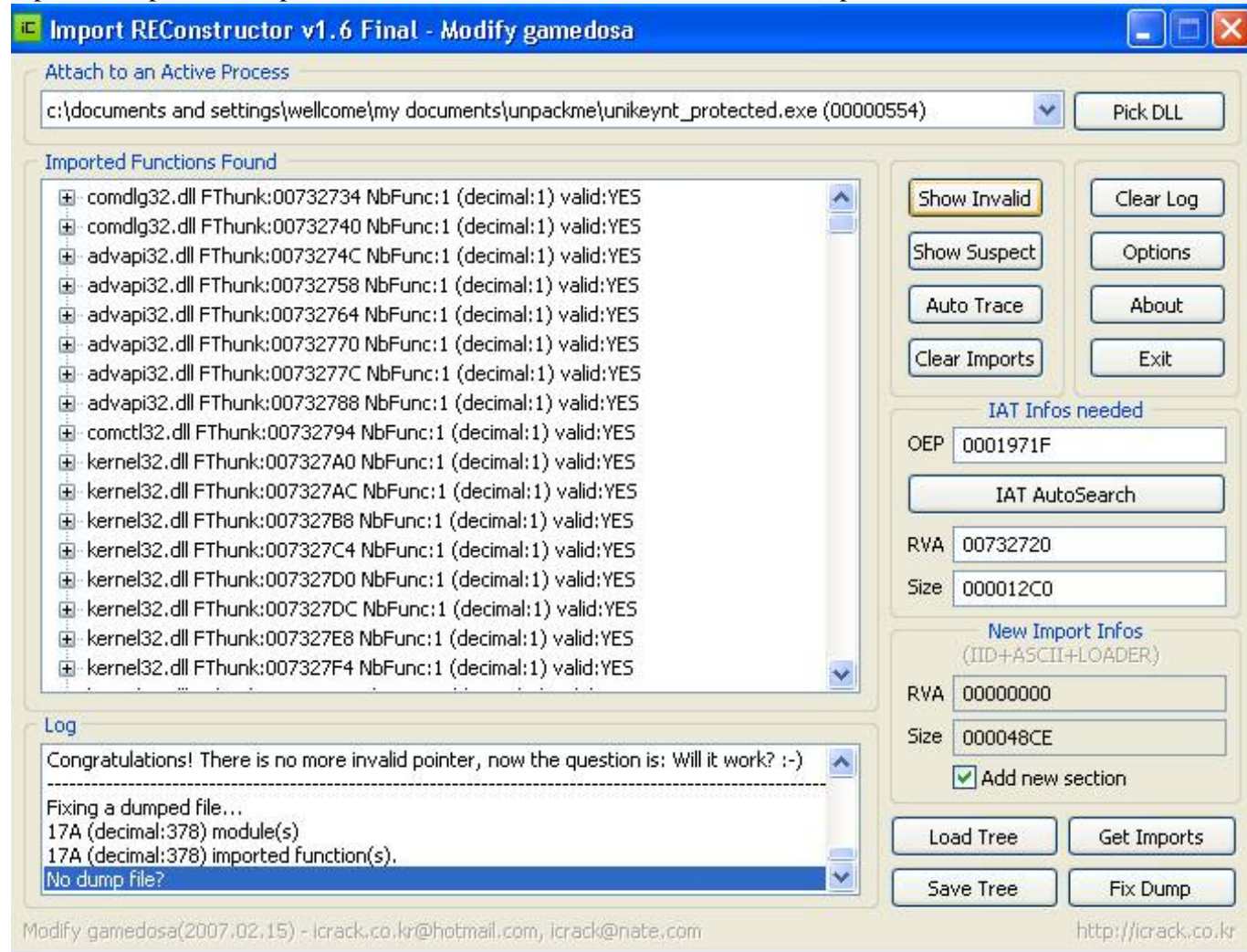
It is probably of course, because the IAT is in our memory by Enigma Protector created. You have to go find IAT Start and end only

00B32720 = 00000000> IAT Start

00B339E0 00000000 => IAT End

What's more waiting, they bring that information to Imporect only: D

Input is complete GetImports => Show Invaild => Cut Thunks => Fix dump



Hehe, so then it is finished

Of course by the time urgency, they should not have been explained kī. And the Enigma Protector also add option WinAPI emulation + Virtual Machine ..

If using WinAPI emulation combined with WinAPI redirection is or will be. They can appoint the following as examples: when the WinAPI emulation function ExitProcess will have 1 code in the memory by Enigma Protector created, and 1 in the code kernel32.dll. But it Virtual Machine request map file something, they do not understand this should not zu research: D. If you have time, then maybe they will write a 1 to 2 items on this child, with more detailed explanations than + features a higher level of Enigma Protector: D

Takada (14/10/2007)

**Themida-EMI will then go through
EMI dth a 1.9.3.0 - Unpackmele and el6
Auto Trial # 6 @ 2007 Italy hectares of the O (hectares of the hot o @ i m a l . C o
m)**

(Write the gift of friendship REA - all copy must be agreed by the author)

[Art cletitl e i]: Themida 1.9.3.0-Unpackmele and el6

[A u t h o r]: **hacnh o**

[A u t r h o h o m e a G E p]: <http://hvaonline.net>

[T a r g e t]: 4V N Mod z A u d i t i o N L o a d e r - A u g u s t 2 0 0 7

[S i z e]: 1.29 MB (1,359.872 bytes TES)

[D o w n l o a d G E a d]: 4V N . o r g

[P a c k e r]: Themida in c e n L i w i l l V 1.9.3.0C.B-> Or e a n
sTec h n o l o g i e s

[C o m p o i l a t i o n a l s o u r c e s]: M i c r o s o f t V i s o f A L C + + 8 . 0

[T o o l s]: The0DBG + hideToolz **fly** by, LordPE, ImportREC, peid0.94

[O S]: W i n X P _ S P 2

[G r e e T Z]: R E A , E x e c u t o o l s , P e d i t a l y , U n p a c k C N , S p r o j e c t
dARTe Me a m m b e r z



_Chao You, we met in this tut Friday. Perhaps you have gradually familiar with Themida and no longer is and the fear of it anymore: D! After training 5 success over the past 5 tuts, maybe we'll make down the mountain thẳng any meat to give it a focused xom. He he, he does make time in the flag? Perhaps only the Vietnamese

people use Vietnamese goods, for S-ROBOT_Ultra_1008 or 4VN-2007.10.24.0 AuWindow many ;-). Islamic truocc pm when tickets hì huc at this present to the boil: P, the time to open it re im after a new need to understand it or Audition sro client and an Internet connection to run a new, really crazy T_T, the bags have dial up, how is heaven, only six đành target see what they have, AutoTT Themida + or VB 6 + Themida, ôi Dao few things to it since, is that VC ++, á à must and today has a pa 4VNLoader sent the bags for something, a month to 8 and also to run, thanks to help see, try to check, the VC ++8.0, browse browse, take the meat small children ...

_Truoc The start guide unpack this, I particularly note: I could not guide you crack any software in Vietnam, the purpose of me unpack the software originated from Vietnam is to learn only. And when run on the PC is stopped immediately, and not contrary to learn how to encode of this software. For software 4VNLoader by 4VN team, I should not play Audition does not know it is a version pub or private, are free or charge. So my target temporarily compressed with Winrar with password. When he or Com Zombie REA to confirm, as if it is private or charge it to force me to remove it and only as a tut learning vegetarian only. The feeling you!

_Trong Tut this we will use the Virtual Machine: VMS Virtual APT-wrapper (I think the aged 4VN to the level 7, 8 and up, but if this is to ensure some of his father this tightrope on 10 for it beautiful;)), may not function EP Virtualization.



_Khong The more tired. But also unpack tuốt à!. To cope with this type protect, unpack the previous model has been bankrupt, just as redirect code-injection. And as you well know, thằng any intervention in most bạo raw files, Dear servers khua Tau! Drink this month, I borrowed đành Buddha glass flowers, bring Tau deal 4VN only: P! So this is interesting? You will see clearly. Speaking in advance if you never know what is Inject Code tired the day: D! OK to do the children:

_ L o rg adta et l v aoOl italy:

Address	Hex dump	Disassembly	Comment
008B2014	B8 00000000	MOV EAX, 0	
008B2019	60	PUSHAD	
008B201A	0BC0	OR EAX, EAX	
008B201C	74 68	JE SHORT 4UNLoade.008B2086	
008B201E	E8 00000000	CALL 4UNLoade.008B2023	
008B2023	58	POP EAX	
008B2024	05 53000000	ADD EAX, 53	
008B2029	8038 E9	CMP BYTE PTR DS:[EAX], 0E9	kernel32.7C816D4F
008B202C	75 13	JNZ SHORT 4UNLoade.008B2041	
008B202E	61	POPAD	
008B202F	EB 45	JMP SHORT 4UNLoade.008B2076	
008B2031	DB2D 37208B00	FLO TBYTE PTR DS:[8B2037]	

Alt + M, set a breakpoint on write:

003A0000	00002000			View in Disassembler	Enter
00400000	00001000	4UNLoade	PE heade	Dump in CPU	
00401000	003BB000	4UNLoade	code, dat	Dump	
007BC000	000F5000	4UNLoade	.rsrc	Search	Ctrl+B
008B1000	00001000	4UNLoade	.idata	Set break-on-access	F2
008B2000	00144000	4UNLoade	JunBaJo	Set memory breakpoint on access	
00A00000	00007000			Set memory breakpoint on write	
00AC0000	00002000			Set access	
00AD0000	00103000				
00BE0000	0010E000				
629C0000	00001000	LPK	PE heade		
629C1000	00005000	LPK	code, imp		
629C6000	00001000	LPK	.data		
629C7000	00001000	LPK	.rsrc		
629C8000	00001000	LPK	resource		
74D90000	00001000	USP10	relocati		
			PE heade		

_Sh T i f + F 93times:

Address	Hex dump	Disassembly	Comme
009CB152	F3:A4	REP MOVSB BYTE PTR ES:[EDI], BYTE PTR DS	
009CB154	C685 192B9206	MOV BYTE PTR SS:[EBP+6922B19], 56	
009CB158	68 396D1FD4	PUSH D41F6D39	
009CB160	FFB5 49049206	PUSH DWORD PTR SS:[EBP+6920449]	
009CB166	8085 38229706	LEA EAX, DWORD PTR SS:[EBP+6972238]	
009CB16C	FFD0	CALL NEAR EAX	
009CB16E	68 00800000	PUSH 8000	
009CB173	6A 00	PUSH 0	
009CB175	F3	REP MOVSB	

_F7, F8:

Address	Hex dump	Disassembly	Comment
009CB152	F3:A4	REP MOVSB BYTE PTR ES:[EDI], BYTE PTR DS	
009CB154	C685 192B9206	MOV BYTE PTR SS:[EBP+6922B19], 56	
009CB158	68 396D1FD4	PUSH D41F6D39	
009CB160	FFB5 49049206	PUSH DWORD PTR SS:[EBP+6920449]	
009CB166	8085 38229706	LEA EAX, DWORD PTR SS:[EBP+6972238]	
009CB16C	FFD0	CALL NEAR EAX	
009CB16E	68 00800000	PUSH 8000	
009CB173	6A 00	PUSH 0	
009CB175	F3	REP MOVSB	

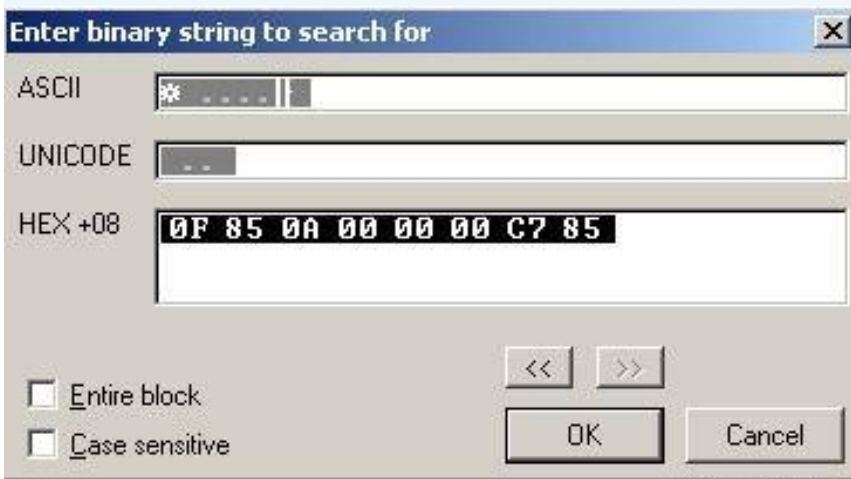
_Sh T i f + F9:

Address	Hex dump	Disassembly	Comment
009D94E3	8908	MOV DWORD PTR DS:[EAX], ECX	
009D94E5	AD	LODS DWORD PTR DS:[ESI]	
009D94E6	C746 FC 000000	MOV DWORD PTR DS:[ESI-4], 0	
009D94ED	89B5 AD0C9206	MOV DWORD PTR SS:[EBP+6920CA0], ESI	4UNLoade.009CBA33
009D94F3	83F8 FF	CMP EAX, -1	
009D94F6	0F85 20000000	JNZ 4UNLoade.009D951C	
009D94FC	813E DDDDDDD0	CMP DWORD PTR DS:[ESI], DDDDDDD0	
009D9502	0F85 14000000	JNZ 4UNLoade.009D951C	
009D9508	C706 00000000	MOV DWORD PTR DS:[ESI], 0	

_Gio Press F8 trace down.

Address	Hex dump	Disassembly	Comment
009D8C02	C785 39169206	MOV DWORD PTR SS:[EBP+6921639], 0	
009D8C0C	C785 51249206	MOV DWORD PTR SS:[EBP+6922451], 0	
009D8C16	83BD 2FBE9B06	CMP DWORD PTR SS:[EBP+69BBE2F], 0	
009D8C1D	0F84 08000000	JE 4UNLoade.009D8C2B	
009D8C23	8D9D 8D469A06	LEA EBX, DWORD PTR SS:[EBP+69A468D]	
009D8C29	FFD3	CALL NEAR EBX	4UNLoade.009C11A0
009D8C2B	FF85 29129206	INC DWORD PTR SS:[EBP+6921229]	
009D8C31	83BD 29129206	CMP DWORD PTR SS:[EBP+6921229], 64	
009D8C38	0F82 62000000	JB 4UNLoade.009D8CA0	
009D8C3E	C785 29129206	MOV DWORD PTR SS:[EBP+6921229], 1	

_Go Ctrl + B: 0F850A000000C785

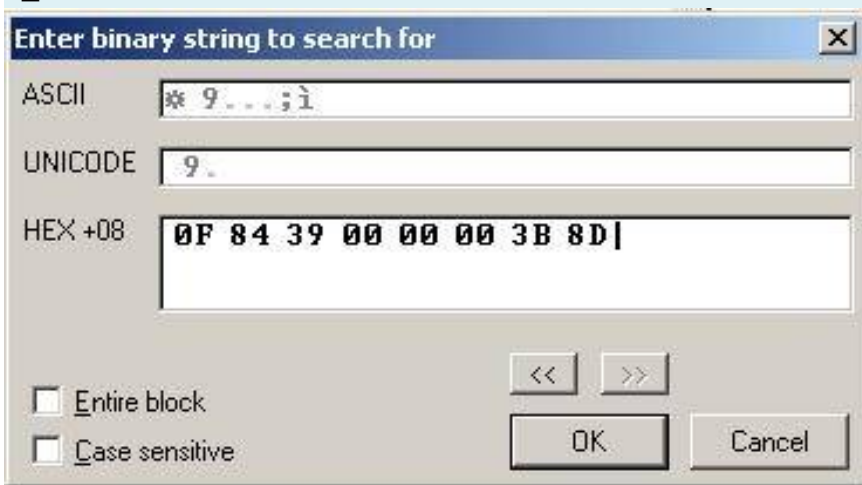


No To bring you here:

009D8C7C	3985 B51A9206	CMP DWORD PTR SS:[EBP+6921AB5], EAX	
009D8C82	0F84 17000000	JE 4UNLoade.009D8C9F	
009D8C88	83BD 552D9206	CMP DWORD PTR SS:[EBP+6922D55], 0	
009D8C8F	0F85 0A000000	JNZ 4UNLoade.009D8C9F	
009D8C95	C785 65309206	MOV DWORD PTR SS:[EBP+6923065], 1	
009D8C9F	61	POPAD	
009D8CA0	B9 50F49F3C	MOV ECX, 3C9FF450	
009D8CA5	BA 220B2C74	MOV EDX, 742C0B22	

Nho Address 009 D 8 C 8 F / 0F8 5 0 A 0 0 0 0 Jn Z 4V NL oad 009 E. D 9 C 8 F

_ GO C t r l + B: 0F843 9 0 0 0 0 0 0 3 B 8 D



Toi Here:

009D8DCE	0F84 39000000	JE 4UNLoade.009D8E00	
009D8DD4	3B8D A0D9206	CMP ECX, DWORD PTR SS:[EBP+6920DAD], kernel32.7C800000	
009D8DDA	0F84 2D000000	JE 4UNLoade.009D8E00	
009D8DE0	3B8D CD1F9206	CMP ECX, DWORD PTR SS:[EBP+6921FCD], USER32.77D40000	
009D8DE6	0F84 21000000	JE 4UNLoade.009D8E00	
009D8DEC	3B8D F9219206	CMP ECX, DWORD PTR SS:[EBP+69221F9], ADVAPI32.77DD0000	
009D8DF2	0F84 15000000	JE 4UNLoade.009D8E00	
009D8DF8	8D9D 4AD39B06	LEA EBX, DWORD PTR SS:[EBP+69BD34A], 4UNLoade.009C11A0	
009D8DFE	FFD3	CALL NEAR EBX	
009D8E00	8BF8	MOV EDI, EAX	
009D8E02	8985 A9309206	MOV DWORD PTR SS:[EBP+69230A9], EAX	
009D8E08	E9 B4060000	JMP 4UNLoade.009D94C1	
009D8E0D	8D9D 4AD39B06	LEA EBX, DWORD PTR SS:[EBP+69BD34A], 4UNLoade.009C11A0	
009D8E13	FFD3	CALL NEAR EBX	
009D8E15	82BD 251F9206	CMP DWORD PTR SS:[EBP+6921F2E], 0	

Nho Address 009 D 8 DCE 1 in F8 4 3 9 0 0 0 0 O O J E 4 V N L o a d e. 009 D 8 EOD

Nho Finish what more hi, labor Dear, if you have read the series I wrote about tut Armadillo will know next to you is do Ctrl + F2. Why, because when you press F8 target trace through the API it will be canceled hash, no nothing. When you restart again erasing the memory OllyDB UDD home, not the breakpoint in the function library is a very tired. Now repeat the above steps to place F7, F8 and Shift + F9 2 times, finished press Alt + M remove the breakpoint. Now press Ctrl + G: 009D8C8F:



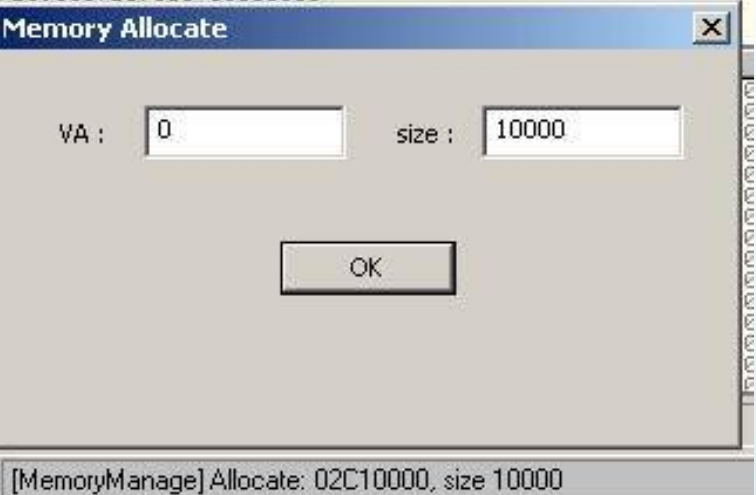
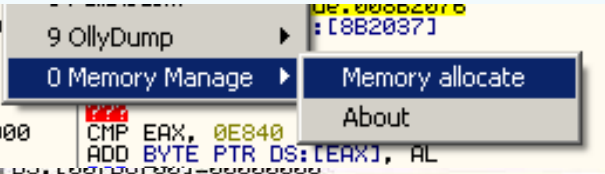
Tai That the patch:



Tuong Autonomy in OO9 D C D E 8 to patch

Address	Hex dump	Disassembly	Comment
009D8DCE	EB 28	JMP SHORT 4UNLoad.009D8DF8	
009D8DD0	3900	CMP DWORD PTR DS:[EAX], EAX	4UNLoad.007B6700
009D8DD2	0000	ADD BYTE PTR DS:[EAX], AL	
009D8DD4	3B8D AD09206	CMP ECX, DWORD PTR SS:[EBP+6920AD0]	kernel32.7C800000
009D8DDA	0F84 2D000000	JE 4UNLoad.009D8E00	
009D8DE0	3B8D CD1F9206	CMP ECX, DWORD PTR SS:[EBP+6921FCD]	USER32.77D40000
009D8DE6	0F84 21000000	JE 4UNLoad.009D8E00	
009D8DEC	3B8D E9210000	CMP ECX, DWORD PTR SS:[EBP+6921FCD]	USER32.77D40000

Ok Complete patch continue, we need to create a code cave to write code in order to do this you should use plugin MemoryManage.dll. Create and simply click OK all, look to see time we have a space to inject into the code.



Ctrl + G to fly 2C10000 see stars

Address	Hex dump	Disassembly	Com
02C10000	0000	ADD BYTE PTR DS:[EAX], AL	
02C10002	0000	ADD BYTE PTR DS:[EAX], AL	
02C10004	0000	ADD BYTE PTR DS:[EAX], AL	
02C10006	0000	ADD BYTE PTR DS:[EAX], AL	
02C10008	0000	ADD BYTE PTR DS:[EAX], AL	
02C1000A	0000	ADD BYTE PTR DS:[EAX], AL	
02C1000C	0000	ADD BYTE PTR DS:[EAX], AL	

Trong Slippy, the hours we paste this binary code is invisible.

Themida - then will go through

A 3000 0 0 0 008 9 08 A DC 746FC000 0 00 00E 9000 0 0 0 0050 A1 0 0 000 0 00 89 078 0 7FF FE 8750 86

6 C 747F E FF15 EB 0666 C 74 7FE FF2558 E 900 0 00 00 050 A 1 0 0 0 000 0 08 94 701 8 07F F F E 8750
8

66 C 747 FF FF15 EB 0666 C 74 7F FFF2 5 58 0 F85 0 00 00 000 E 9 0 0 0 000 0 083 C 704 E 9 0 0 0 000 00

Copy This to the clipboard, paste into Olly binary format.

Address	Hex dump	Disassembly	Comment
02C10000	0000	ADD BYTE PTR DS:[EAX], AL	
02C10001		[EAX], AL	
02C10002		[EAX], AL	
02C10003		[EAX], AL	
02C10004		[EAX], AL	
02C10005		[EAX], AL	
02C10006		[EAX], AL	
02C10007		[EAX], AL	
02C10008		[EAX], AL	
02C10009		[EAX], AL	
02C1000A		[EAX], AL	
02C1000B		[EAX], AL	
02C1000C		[EAX], AL	
02C1000D		[EAX], AL	
02C1000E		[EAX], AL	
02C1000F		[EAX], AL	
02C10010		[EAX], AL	
02C10011		[EAX], AL	
02C10012		[EAX], AL	
02C10013		[EAX], AL	
02C10014		[EAX], AL	
02C10015		[EAX], AL	
02C10016		[EAX], AL	
02C10017		[EAX], AL	
02C10018		[EAX], AL	
02C10019		[EAX], AL	
02C1001A		[EAX], AL	
02C1001B		[EAX], AL	
02C1001C		[EAX], AL	
02C1001D		[EAX], AL	
02C1001E		[EAX], AL	
02C1001F		[EAX], AL	
02C10020		[EAX], AL	
02C10021		[EAX], AL	
02C10022		[EAX], AL	
02C10023		[EAX], AL	
02C10024		[EAX], AL	
02C10025		[EAX], AL	
02C10026		[EAX], AL	
02C10027		[EAX], AL	
02C10028		[EAX], AL	
02C10029		[EAX], AL	
02C1002A		[EAX], AL	
02C1002B		[EAX], AL	
02C1002C		[EAX], AL	
02C1002D		[EAX], AL	
02C1002E		[EAX], AL	
02C1002F		[EAX], AL	
02C10030		[EAX], AL	
02C10031		[EAX], AL	
02C10032		[EAX], AL	
02C10033		[EAX], AL	
02C10034		[EAX], AL	
02C10035		[EAX], AL	
02C10036		[EAX], AL	
02C10037		[EAX], AL	
02C10038		[EAX], AL	
02C10039		[EAX], AL	
02C1003A		[EAX], AL	
02C1003B		[EAX], AL	
02C1003C		[EAX], AL	
02C1003D		[EAX], AL	
02C1003E		[EAX], AL	
02C1003F		[EAX], AL	
02C10040		[EAX], AL	
02C10041		[EAX], AL	
02C10042		[EAX], AL	
02C10043		[EAX], AL	
02C10044		[EAX], AL	
02C10045		[EAX], AL	
02C10046		[EAX], AL	
02C10047		[EAX], AL	
02C10048		[EAX], AL	
02C10049		[EAX], AL	
02C1004A		[EAX], AL	
02C1004B		[EAX], AL	
02C1004C		[EAX], AL	
02C1004D		[EAX], AL	
02C1004E		[EAX], AL	
02C1004F		[EAX], AL	
02C10050		[EAX], AL	
02C10051		[EAX], AL	
02C10052		[EAX], AL	
02C10053		[EAX], AL	
02C10054		[EAX], AL	
02C10055		[EAX], AL	
02C10056		[EAX], AL	
02C10057		[EAX], AL	
02C10058		[EAX], AL	
02C10059		[EAX], AL	
02C1005A		[EAX], AL	
02C1005B		[EAX], AL	
02C1005C		[EAX], AL	
02C1005D		[EAX], AL	
02C1005E		[EAX], AL	
02C1005F		[EAX], AL	
02C10060		[EAX], AL	
02C10061		[EAX], AL	
02C10062		[EAX], AL	
02C10063		[EAX], AL	
02C10064		[EAX], AL	
02C10065		[EAX], AL	
02C10066		[EAX], AL	
02C10067		[EAX], AL	
02C10068		[EAX], AL	
02C10069		[EAX], AL	
02C1006A		[EAX], AL	
02C1006B		[EAX], AL	
02C1006C		[EAX], AL	
02C1006D		[EAX], AL	
02C1006E		[EAX], AL	
02C1006F		[EAX], AL	
02C10070		[EAX], AL	
02C10071		[EAX], AL	
02C10072		[EAX], AL	
02C10073		[EAX], AL	
02C10074		[EAX], AL	
02C10075		[EAX], AL	
02C10076		[EAX], AL	
02C10077		[EAX], AL	
02C10078		[EAX], AL	
02C10079		[EAX], AL	
02C1007A		[EAX], AL	
02C1007B		[EAX], AL	
02C1007C		[EAX], AL	
02C1007D		[EAX], AL	
02C1007E		[EAX], AL	
02C1007F		[EAX], AL	
02C10080		[EAX], AL	
02C10081		[EAX], AL	
02C10082		[EAX], AL	
02C10083		[EAX], AL	
02C10084		[EAX], AL	
02C10085		[EAX], AL	
02C10086		[EAX], AL	
02C10087		[EAX], AL	
02C10088		[EAX], AL	
02C10089		[EAX], AL	
02C1008A		[EAX], AL	
02C1008B		[EAX], AL	
02C1008C		[EAX], AL	
02C1008D		[EAX], AL	
02C1008E		[EAX], AL	
02C1008F		[EAX], AL	
02C10090		[EAX], AL	
02C10091		[EAX], AL	
02C10092		[EAX], AL	
02C10093		[EAX], AL	
02C10094		[EAX], AL	
02C10095		[EAX], AL	
02C10096		[EAX], AL	
02C10097		[EAX], AL	
02C10098		[EAX], AL	
02C10099		[EAX], AL	
02C1009A		[EAX], AL	
02C1009B		[EAX], AL	
02C1009C		[EAX], AL	
02C1009D		[EAX], AL	
02C1009E		[EAX], AL	
02C1009F		[EAX], AL	
02C100A0		[EAX], AL	
02C100A1		[EAX], AL	
02C100A2		[EAX], AL	
02C100A3		[EAX], AL	
02C100A4		[EAX], AL	
02C100A5		[EAX], AL	
02C100A6		[EAX], AL	
02C100A7		[EAX], AL	
02C100A8		[EAX], AL	
02C100A9		[EAX], AL	
02C100AA		[EAX], AL	
02C100AB		[EAX], AL	
02C100AC		[EAX], AL	
02C100AD		[EAX], AL	
02C100AE		[EAX], AL	
02C100AF		[EAX], AL	
02C100B0		[EAX], AL	
02C100B1		[EAX], AL	
02C100B2		[EAX], AL	
02C100B3		[EAX], AL	
02C100B4		[EAX], AL	
02C100B5		[EAX], AL	
02C100B6		[EAX], AL	
02C100B7		[EAX], AL	
02C100B8		[EAX], AL	
02C100B9		[EAX], AL	
02C100BA		[EAX], AL	
02C100BB		[EAX], AL	
02C100BC		[EAX], AL	
02C100BD		[EAX], AL	
02C100BE		[EAX], AL	
02C100BF		[EAX], AL	
02C100C0		[EAX], AL	
02C100C1		[EAX], AL	
02C100C2		[EAX], AL	
02C100C3		[EAX], AL	
02C100C4		[EAX], AL	
02C100C5		[EAX], AL	
02C100C6		[EAX], AL	
02C100C7		[EAX], AL	
02C100C8		[EAX], AL	
02C100C9		[EAX], AL	
02C100CA		[EAX], AL	
02C100CB		[EAX], AL	
02C100CC		[EAX], AL	
02C100CD		[EAX], AL	
02C100CE		[EAX], AL	
02C100CF		[EAX], AL	
02C100D0		[EAX], AL	
02C100D1		[EAX], AL	
02C100D2		[EAX], AL	
02C100D3		[EAX], AL	
02C100D4		[EAX], AL	
02C100D5		[EAX], AL	
02C100D6		[EAX], AL	
02C100D7		[EAX], AL	
02C100D8		[EAX], AL	
02C100D9		[EAX], AL	
02C100DA		[EAX], AL	
02C100DB		[EAX], AL	
02C100DC		[EAX], AL	
02C100DD		[EAX], AL	
02C100DE		[EAX], AL	
02C100DF		[EAX], AL	
02C100E0		[EAX], AL	
02C100E1		[EAX], AL	
02C100E2		[EAX], AL	
02C100E3		[EAX], AL	
02C100E4		[EAX], AL	
02C100E5		[EAX], AL	
02C100E6		[EAX], AL	
02C100E7		[EAX], AL	
02C100E8		[EAX], AL	
02C100E9		[EAX], AL	
02C100EA		[EAX], AL	
02C100EB		[EAX], AL	
02C100EC		[EAX], AL	
02C100ED		[EAX], AL	
02C100EE		[EAX], AL	
02C100EF		[EAX], AL	
02C100F0		[EAX], AL	
02C100F1		[EAX], AL	
02C100F2		[EAX], AL	
02C100F3		[EAX], AL	
02C100F4		[EAX], AL	
02C100F5		[EAX], AL	
02C100F6		[EAX], AL	
02C100F7		[EAX], AL	
02C100F8		[EAX], AL	
02C100F9		[EAX], AL	
02C100FA		[EAX], AL	
02C100FB		[EAX], AL	
02C100FC		[EAX], AL	
02C100FD		[EAX], AL	
02C100FE		[EAX], AL	
02C100FF		[EAX], AL	

Sau The paste:

Address	Hex dump	Disassembly	Comment
02C10000	A3 00000000	MOV DWORD PTR DS:[0], EAX	4UNLoade.007B6700
02C10005	8908	MOV DWORD PTR DS:[EAX], ECX	
02C10007	AD	LODS DWORD PTR DS:[ESI]	
02C10008	C746 FC 00000000	MOV DWORD PTR DS:[ESI-4], 0	
02C1000F	E9 00000000	JMP 02C10014	
02C10014	50	PUSH EAX	4UNLoade.007B6700
02C10015	A1 00000000	MOV EAX, DWORD PTR DS:[0]	
02C1001A	8907	MOV DWORD PTR DS:[EDI], EAX	4UNLoade.007B6700
02C1001C	807F FF E8	CMP BYTE PTR DS:[EDI-1], 0E8	
02C10020	75 08	JNZ SHORT 02C1002A	
02C10022	66:C747 FE FF15	MOV WORD PTR DS:[EDI-2], 15FF	
02C10028	EB 06	JMP SHORT 02C10030	
02C1002A	66:C747 FE FF25	MOV WORD PTR DS:[EDI-2], 25FF	
02C10030	58	POP EAX	4UNLoade.007B6700
02C10031	E9 00000000	JMP 02C10036	
02C10036	50	PUSH EAX	4UNLoade.007B6700
02C10037	A1 00000000	MOV EAX, DWORD PTR DS:[0]	
02C1003C	8947 01	MOV DWORD PTR DS:[EDI+1], EAX	4UNLoade.007B6700
02C1003F	807F FF E8	CMP BYTE PTR DS:[EDI-1], 0E8	
02C10043	75 08	JNZ SHORT 02C1004D	
02C10045	66:C747 FF FF15	MOV WORD PTR DS:[EDI-1], 15FF	
02C1004B	EB 06	JMP SHORT 02C10053	
02C1004D	66:C747 FF FF25	MOV WORD PTR DS:[EDI-1], 25FF	
02C10053	58	POP EAX	4UNLoade.007B6700
02C10054	0F85 00000000	JNZ 02C1005A	
02C1005A	E9 00000000	JMP 02C1005F	
02C1005F	83C7 04	ADD EDI, 4	
02C10062	E9 00000000	JMP 02C10067	
02C10067	0000	ADD BYTE PTR DS:[EAX], AL	

Doan Code is original, we will edit as follows:

Address	Hex dump	Disassembly	Comment
02C10000	A3 0001C102	MOV DWORD PTR DS:[2C10100], EAX	4UNLoade.007B6700
02C10005	8908	MOV DWORD PTR DS:[EAX], ECX	
02C10007	AD	LODS DWORD PTR DS:[ESI]	
02C10008	C746 FC 00000000	MOV DWORD PTR DS:[ESI-4], 0	
02C1000F	- E9 D994DCFD	JMP 4UNLoade.009D94ED	
02C10014	50	PUSH EAX	4UNLoade.007B6700
02C10015	A1 0001C102	MOV EAX, DWORD PTR DS:[2C10100]	
02C1001A	8907	MOV DWORD PTR DS:[EDI], EAX	4UNLoade.007B6700
02C1001C	807F FF E8	CMP BYTE PTR DS:[EDI-1], 0E8	
02C10020	75 08	JNZ SHORT 02C1002A	
02C10022	66:C747 FE FF15	MOV WORD PTR DS:[EDI-2], 15FF	
02C10028	EB 06	JMP SHORT 02C10030	
02C1002A	66:C747 FE FF25	MOV WORD PTR DS:[EDI-2], 25FF	
02C10030	58	POP EAX	4UNLoade.007B6700
02C10031	- E9 8D95DCFD	JMP 4UNLoade.009D95C3	
02C10036	50	PUSH EAX	4UNLoade.007B6700
02C10037	A1 0001C102	MOV EAX, DWORD PTR DS:[2C10100]	
02C1003C	8947 01	MOV DWORD PTR DS:[EDI+1], EAX	4UNLoade.007B6700
02C1003F	807F FF E8	CMP BYTE PTR DS:[EDI-1], 0E8	
02C10043	75 08	JNZ SHORT 02C1004D	
02C10045	66:C747 FF FF15	MOV WORD PTR DS:[EDI-1], 15FF	
02C1004B	EB 06	JMP SHORT 02C10053	
02C1004D	66:C747 FF FF25	MOV WORD PTR DS:[EDI-1], 25FF	
02C10053	58	POP EAX	4UNLoade.007B6700
02C10054	- 0F85 6995DCFD	JNZ 4UNLoade.009D95C3	
02C1005A	- E9 4C95DCFD	JMP 4UNLoade.009D95AB	
02C1005F	83C7 04	ADD EDI, 4	
02C10062	- E9 8694DCFD	JMP 4UNLoade.009D94ED	
02C10067	0000	ADD BYTE PTR DS:[EAX], AL	
02C10069	0000	ADD BYTE PTR DS:[EAX], AL	

A30001C1028908ADC746FC00000000 E 9D994DCFD50A10001C1028907807FFF E 87508 6
6C747FEFF 1 5 E BO6 66C747FEFF2558 E 98D95DCFD50A1 O 001C10289 4 701807FFF E
8750866C747FFFF15 E BO666C747FFFF2558OF 856995DCFD E 94C95DC F D83C704 E 9 8 694DCFD
_Bay Time to jump 009 D 94 E 3 patch:

Address	Hex dump	Disassembly
009D94E3	- E9 186B2302	JMP 02C10000
009D94E8	FC	CLD
009D94E9	0000	ADD BYTE PTR DS:[EAX], AL
009D94EB	0000	ADD BYTE PTR DS:[EAX], AL
009D94ED	89B5 AD0C9206	MOV DWORD PTR SS:[EBP+6920CAD], ESI
009D94F3	83F8 FF	CMP EAX, -1
009D94F6	0F85 20000000	JNZ 4UNLoade.009D951C
009D94FC	813E 00000000	CMP DWORD PTR DS:[ESI], 00000000

_Toi Here:

009D9597	8B	MOV EAX, EBX	
009D9598	58	POP EAX	4UNLoade.007B6700
009D9599	AA	STOS BYTE PTR ES:[EDI]	
009D959A	- E9 24000000	JMP 4UNLoade.009D95C3	
009D959F	58	POP EAX	4UNLoade.007B6700
009D95A0	AA	STOS BYTE PTR ES:[EDI]	
009D95A1	807F FF E9	CMP BYTE PTR DS:[EDI-1], 0E9	
009D95A5	0F85 18000000	JNZ 4UNLoade.009D95C3	
009D95AB	83BD 2FBE9B06	CMP DWORD PTR SS:[EBP+69BBE2F], 0	
009D95B3	0F84 00000000	JE 4UNLoade.009D95C0	

And the patch:

009D9598	58	POP EAX	
009D9599	AA	STOS BYTE PTR ES:[EDI]	
009D959A	- E9 657A8EFF	JMP 02C1004	
009D959F	58	POP EAX	
009D95A0	AA	STOS BYTE PTR ES:[EDI]	
009D95A1	807F FF E9	CMP BYTE PTR DS:[EDI-1], 0E9	
009D95A5	0F85 18000000	JNZ 4UNLoade.009D95C3	

_Tiep Proceed to:

009D95A1	807F FF E9	CMP BYTE PTR DS:[EDI-1], 0E9	
009D95A5	0F85 18000000	JNZ 4UNLoade.009D95C3	
009D95AB	83BD 2FBE9B06	CMP DWORD PTR SS:[EBP+69BBE2F], 0	
009D95B2	0F84 00000000	JE 4UNLoade.009D95C0	
009D95B3	8B4F 044F9006	MOV EAX, DWORD PTR SS:[EBP+6904F0d1]	

Patch:

009D95A0	AA	STOS BYTE PTR ES:[EDI]	
009D95A1	807F FF E9	CMP BYTE PTR DS:[EDI-1], 0E9	
009D95A5	- E9 8C6A2302	JMP 02C10036	
009D95AA	0083 BD2FBE9B	ADD BYTE PTR DS:[EBX+9BBE2FBD], AL	
009D95B0	06	PUSH ES	
009D95B1	80AF	ADD BYTE PTR DS:[EDI], AL	

_Toi:

009D95BE	FFD3	CALL NEAR EBX	
009D95C0	8847 04	MOV BYTE PTR DS:[EDI+4], AL	
009D95C3	8B85 A9309206	MOV EAX, DWORD PTR SS:[EBP+69230	
009D95C9	2B07	SAR EAX, 01	

Submit it:

009D95BE	FFD3	CALL NEAR EBX
009D95C0	90	NOP
009D95C1	90	NOP
009D95C2	90	NOP
009D95C3	8B85 A9309206	MOV EAX, DWORD PTR SS:[EBP+692306]
009D95C9	2BC7	SUB EAX, EDI
009D95CB	83E8 04	SUB EAX, 4

Toi:

009D95CE	AB	STOS DWORD PTR ES:[EDI]
009D95CF	AD	LODS DWORD PTR DS:[ESI]
009D95D0	C746 FC 000000	MOV DWORD PTR DS:[ESI-4], 0
009D95D7	E9 11FFFFFF	JMP 4UNLoad.009D94ED

Submit to:

009D95C9	2BC7	SUB EAX, EDI
009D95CB	83E8 04	SUB EAX, 4
009D95CE	90	NOP
009D95CF	AD	LODS DWORD PTR DS:[ESI]
009D95D0	C746 FC 000000	MOV DWORD PTR DS:[ESI-4], 0
009D95D7	E9 11FFFFFF	JMP 4UNLoad.009D94ED
009D95DC	89B5 AD0C9206	MOV DWORD PTR SS:[EBP+6920CAD], 0

Nhuc Than the first, normally only one patch so to avoid the scattered and cancel IAT. CopymemII and elimination of Armadillo IAT, please see the old tut.

Writing the code is finished, press Ctrl + hour B C7010000000083C104

Enter binary string to search for

ASCII: []

UNICODE: []

HEX +09: C7 01 00 00 00 00 83 C1 04

☐ Entire block

☐ Case sensitive

<< >>

OK Cancel

Toi Here:

Address	Hex dump	Disassembly	Comment
009D95FD	C701 00000000	MOV DWORD PTR DS:[ECX], 0	
009D9603	83C1 04	ADD ECX, 4	
009D9606	898D C9219206	MOV DWORD PTR SS:[EBP+69221C9], ECX	
009D960C	E9 10F5FFFF	JMP 4UNLoad.009D8B21	
009D9611	E9 A4060000	JMP 4UNLoad.009D9CBA	
009D9616	60	PUSHAD	
009D9617	8B8D C9219206	MOV ECX, DWORD PTR SS:[EBP+69221C9]	
009D961D	8B09	MOV ECX, DWORD PTR DS:[ECX]	

Dat A breakpoint:

Command: [he 009D9611]

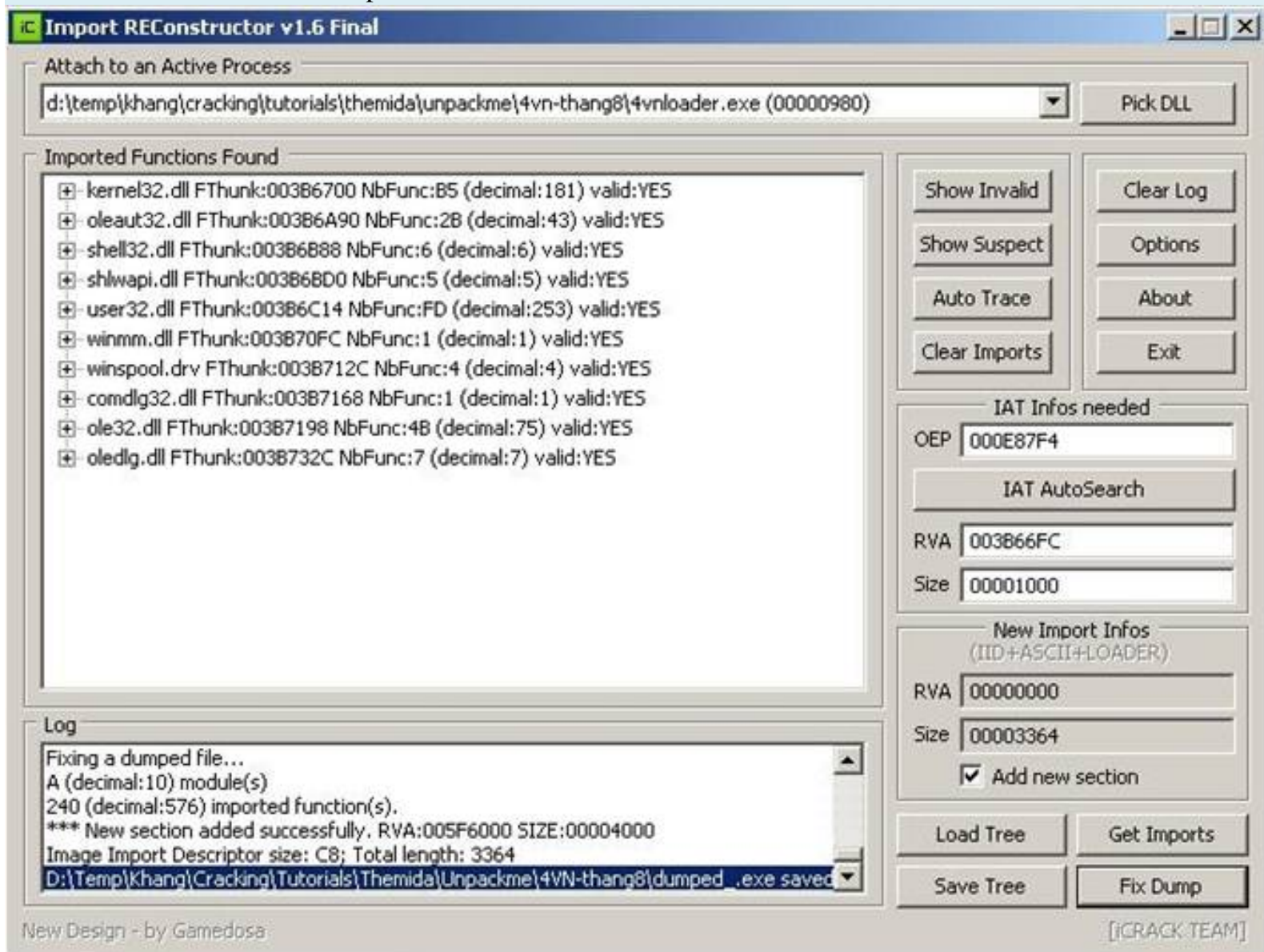
Shift + F9:

Address	Hex dump	Disassembly	Comment
009D9611	E9 A4060000	JMP 4UNLoad.009D9CBA	
009D9616	60	PUSHAD	
009D9617	8B8D C9219206	MOV ECX, DWORD PTR SS:[EBP+69221C9]	
009D961D	8B09	MOV ECX, DWORD PTR DS:[ECX]	
009D961F	898D 2BBE9B06	MOV DWORD PTR SS:[EBP+698BE2B], ECX	
009D9625	8138 4E54444C	CMP DWORD PTR DS:[EAX], 4C44544E	
009D962B	0F85 1C000000	JNZ 4UNLoad.009D964D	
009D9631	66:8178 04 4C2	CMP WORD PTR DS:[EAX+4], 2E4C	

Sau Remove the breakpoint with HD 009D9611. Press Alt + M, press F2 at the code section. F9.

Address	Hex dump	Disassembly	Comment
004E87F4	E9 E73A1500	JMP 4UNLoade.0063C2E0	OEP
004E87F9	E9 52B40C00	JMP 4UNLoade.005B3C50	
004E87FE	E9 3D9F0700	JMP 4UNLoade.00562740	
004E8803	E9 88C51B00	JMP 4UNLoade.006A4D90	
004E8808	E9 23C61400	JMP 4UNLoade.00634E30	
004E880D	E9 BE530D00	JMP 4UNLoade.005B0B00	
004E8812	E9 C9E60600	JMP 4UNLoade.00556EE0	
004E8817	E9 A4B50500	JMP 4UNLoade.00543DC0	
004E881C	E9 EFC60400	JMP 4UNLoade.00534F10	
004E8821	E9 3ABA1C00	JMP 4UNLoade.006B4260	
004E8826	E9 05EF0E00	JMP 4UNLoade.005D7730	
004E882B	E9 60F40900	JMP 4UNLoade.00587C90	
004E8830	E9 8BFC0500	JMP 4UNLoade.005484C0	
004E8835	E9 76BA0500	JMP 4UNLoade.005442B0	
004E883A	E9 39B71B00	JMP 4UNLoade.006A3F78	JMP to ole32.CoRegisterClassObject
004E883F	E9 7C331700	JMP 4UNLoade.0065BB00	
004E8844	E9 77DD0F00	JMP 4UNLoade.005E65C0	
004E8849	E9 42120800	JMP 4UNLoade.00569A90	
004E884E	E9 9DA40300	JMP 4UNLoade.00522CF0	
004E8853	E9 48BA1C00	JMP 4UNLoade.006B42A0	
004E8858	E9 D3C01900	JMP 4UNLoade.00684930	
004E885D	E9 DEB40E00	JMP 4UNLoade.005D3D40	
004E8862	E9 496C0800	JMP 4UNLoade.0056F4B0	
004E8867	E9 54D60900	JMP 4UNLoade.00585EC0	
004E886C	E9 FDAB1B00	JMP 4UNLoade.006A346E	JMP to USER32.MessageBoxW
004E8871	E9 3A821900	JMP 4UNLoade.00680AB0	
004E8876	E9 A5540400	JMP 4UNLoade.0052DD20	
004E887B	E9 D0090700	JMP 4UNLoade.00559250	
004E8880	E9 87B01B00	JMP 4UNLoade.006A390C	JMP to USER32.IsRectEmpty
004E8885	E9 46C10E00	JMP 4UNLoade.005D49D0	
004E888A	E9 015B0C00	JMP 4UNLoade.005F0C40	

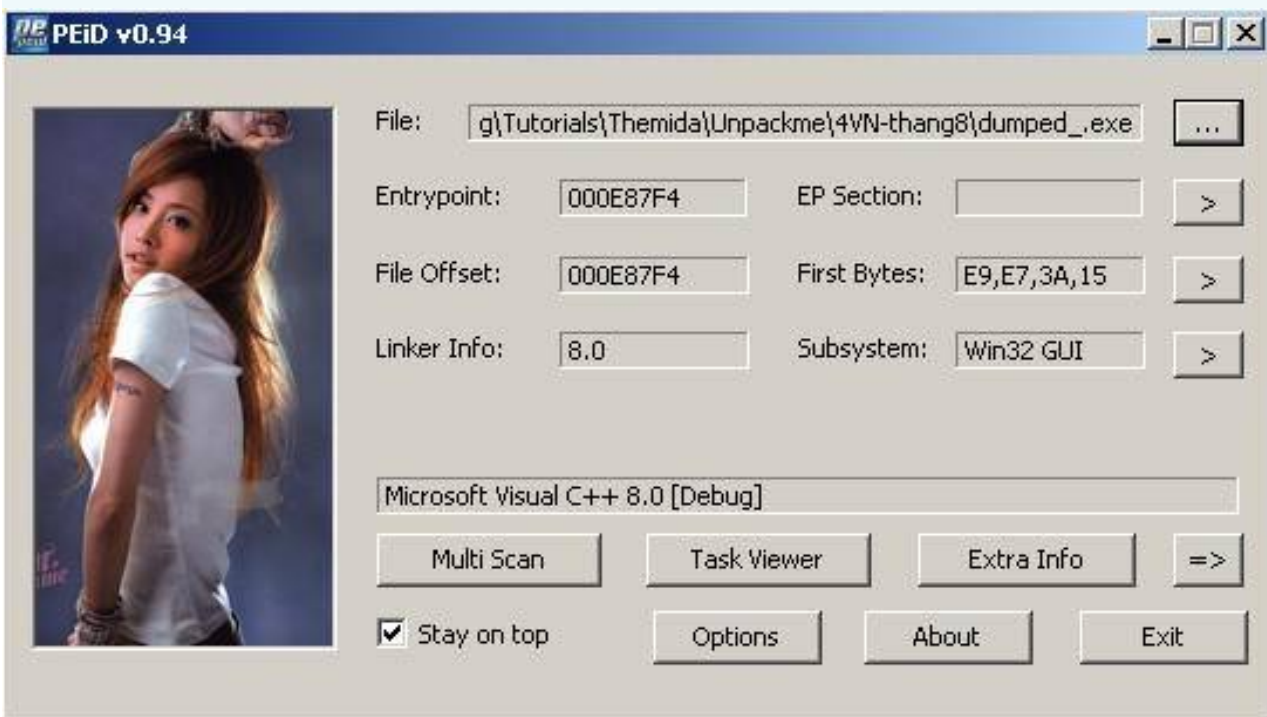
Full _IAT. Now conducted dump, and fix TAT:



Run Try:



PeiD:



A n s e i t a l y s u g g t h e O N S , c o m m e n t s o r c O r r e c t i o n L e a s p e s P v i a M m e : A C _ g r a p e s _ h _ @ _ h _ o _ m _ o _ t m a i l . c

26 - 1 0 - 2 0 0 7

Themida - then will go through

Themida 1.9.1.0 - Unpackme level 1

Tutorial # 1 @ 2007 by hacnho (hacnho@hotmail.com)

(Write the gift of friendship REA - all copy must be agreed by the author)

[Article title]: Themida 1.9.1.0 - Unpackme level 1
[Author]: **hacnho**
[Author home page]: <http://hvaonline.net>
[Target]: A Tool HVA
[Size]: 252 KBS
[Download Page]: hvaonline.net
[Packer]: Themida | WinLicense V 1.9.1.0 -> Oreans Technologies
[Compilation language]: Microsoft Visual C + + 7.0
[Tools]: The0DBG + hideToolz **fly** by, LordPE, ImportREC, peid0.94
[OS]: WinXP_SP2
[Greetz]: REA, Exetools, PEDIY, UnpackCN, snd and ARTeam Memberz

_Chao Relatives, this time I return is to add to your passion MUP a packer, perhaps

it is not new, but unpack the bags, it was popular then, but in Vietnam, many

people unpack successful but do not share technical unpack this, if you ask them to

get the answer is complex lem, Khe very much, need a high level, lol. Mk, dek

_Ok, So I wrote this tut series based on any principles. Simply stop, know where to write to it. To simplify, I only based on version 1.9.1.0 to leak out as a premise to their own pack and unpack all his options by Themida. Can not properly with other software, but basically it is a method similar. Gradually, if this is finished, maybe we will discuss to the meat soft as 4VN or TLG such ;-). If the brothers can see

their contribution and then write a post on here, thanks to the British themselves

before, I was a noob in this issue.

_Công Particular need is what hi. Like normal. We will unpack the debugger

OllyDBG, LordPE, Imprec tools and some other support, but more important is the

end of the first and a computer would ram for the 512. Prior to unpack the bags

want to say is the father of Oreans-themida:), with some pa sure who should be

very warm ong as the region is located in the forum and the anti debug method is

discovered, referred to this new I remember he has failed to com, to which I have

aged by rapid ACC Zom share, when I upload the file to the folder Rapid, I see a

file OllyICe_mod_by_CA, down about 10 I know I do it, dispose ago, a today he

said a VIP party exetools send bags to unpack execryptor asked hem know, I

always ù, Ho Ho, of which I know about this packer dead line, which is about to

load Olly Crash, really mad, the pre OllyICE Com his mod to that, I try to see how

load,;), it runs the vo vo the new business, aged com him him :-). Then load the

finish I do, bít dead line, he he is the bags to dispose exetools that it always unpack

itself, but the evil is the mod Olly that pass by the new themida 1.8.xx pain, so it is

aged VIP Horses also herald the moss Straighten it oreans fix that bug. Chắc his

anger Com lem bags, bags of U.S. hem know. Then I sold up a little out for the first

time that it themida hot start. He he, through this thread I apologize for him Com:

((I hem bit bit sources that children do they put it! Back issues of the type of anti-

debug themida very diverse, they may we have the tools plugin + addon to pass by.

I suggest you straighten HideOD 1.7 (1.81 I do not use it to hide thẳng OllyIce

process as futile to hook driver oreans) + + OllyICE HideToolz 2.1, or if it does

HanOlly do not use HideToolz. unpackme tested on 1.9.5.0 and 1.9.4.0 run well.

Edit Exception as follows:

☒ Ignore memory access violations in KERNEL32

Ignore (pass to program) following exceptions:

- ☒ INT3 breaks
- ☒ Single-step break
- ☒ Memory access violation
- ☒ Integer division by 0
- ☒ Invalid or privileged instruction
- ☒ All FPU exceptions

☒ Ignore also following custom exceptions or ranges:

80000004 (SINGLE STEP)
C0000005 (ACCESS VIOLATION)
C000001D (ILLEGAL INSTRUCTION)
C000008F (FLOAT INEXACT RESULT)
C0000096 (PRIVILEGED INSTRUCTION)
50000000

_Trong Tuts we will start with the easiest, tentatively called Unpackme-1 level. In

the following tuts we will do with the type more difficult to protect.

_Bay Hours Themida 1.9.1.0 open up, creating a new project and protect the

following:

Protection Options for Unpackme.exe

Information Macros

Macros VM: 0
CodeReplace Macros: 0
ENCRYPT Macros: 0
CLEAR Macros: 0

XBundler files

No files to bundle

Protection Options

Anti-Debugger: enabled
Anti-Dumpers: Disabled
API-wrapping Level: 0
Virtual Machine: enabled
Entry Point Ofuscation: Disabled
Memory Guard: Disabled
Anti-Monitor File: Disabled
Anti-Monitor Registry: Disabled
Resource Encryption: Disabled
VMWare compatible: Disabled
Delphi / BCB form protection: disabled

Advanced Protection Options

Encrypt Application: Disabled

. NET assemblies: Disabled

Dll plugin: Disabled

Active Context: Disabled

Section Last Name: hacnho

Compression

Application compression: Disabled

Resources compression: Disabled

SecureEngine compression: Disabled

Virtual Machine Settings

Number of Virtual wrapped APIs: 0

Entry Point Virtualization: 0 instructions

Virtual Machine Processor: Mutable CISC processor

Number of CPUs: 1

Opcode Type: Static opcodes

Dynamic Opcode: Disabled

_Bay Hours to load target OllyDBG:

Address	Hex dump	Disassembly	Comment
00444014	✓ E9 30350000	JMP Unpackme.00447549	
00444019	0000	ADD BYTE PTR DS:[EAX], AL	
0044401B	0000	ADD BYTE PTR DS:[EAX], AL	
0044401D	0000	ADD BYTE PTR DS:[EAX], AL	
0044401F	0000	ADD BYTE PTR DS:[EAX], AL	
00444021	0000	ADD BYTE PTR DS:[EAX], AL	

_Nhan Ctrl + G: ZwFreeVirtualMemory



_Toi And press F2 to RETN command:

Address	Hex dump	Disassembly	Comm
7C90DA48	B8 53000000	MOV EAX, 53	
7C90DA4D	BA 0003FE7F	MOV EDX, 7FFE0300	
7C90DA52	FF12	CALL NEAR DWORD PTR DS:[EDX]	
7C90DA54	C2 1000	RETN 10	
7C90DA57	90	NOP	
7C90DA58	90	NOP	
7C90DA59	90	NOP	

Nhan _Bay hours Shift + F9 and see the window of the Register:


```

Registers (FPU)
EAX 00000000
ECX 0012FFB0
EDX 7C90EB94 ntdll.KiFastSystemCallRet
EBX 7FFD7000
ESP 0012FFC4
EBP 0012FFF0
ESI FFFFFFFF
EDI 7C910738 ntdll.7C910738
EIP 00444014 Unpackme.<ModuleEntryPoint>
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_NO_IMPERSONATION_TOKEN (0000051D)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty 1.5738549221224447450e-3281
ST1 empty 0.00000000000000000040e-4933
ST2 empty -7.1911010359405521500e+757
ST3 empty -UNORM C020 00000017 E35BA6A8
ST4 empty 5.6552512923172035960e-4925
ST5 empty -2.2633026551341701380e+1032
ST6 empty -UNORM CBC0 00010101 E35BA760
ST7 empty 2.1065780380201079220e-3281
      3 2 1 0      E S P U O Z D I
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1

```

_Nhan Shift + F9 to when it is half red: P! (I count 16 times)

```

Registers (FPU)
EAX 00000000
ECX 0012FF2C
EDX 7C90EB94 ntdll.KiFastSystemCallRet
EBX 0240002C
ESP 0012FF30
EBP 0012FF4C
ESI 7C90DA48 ntdll.ZwFreeVirtualMemory
EDI 00418BFA Unpackme.00418BFA
EIP 7C90DA54 ntdll.7C90DA54

C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty -UNORM D0A8 01050104 00000000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 1.000000000000000000000000
ST7 empty 1.000000000000000000000000

      3 2 1 0      E S P U O Z D I
FST 4020 Cond 1 0 0 0 Err 0 0 1 0 0 0 0 0 (EQ)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1

```

_Gio Press F2 to remove breakpoint, press Alt + M to Memory window, set a

Breakpoint in the section on Access code:

00400000	00001000	Unpackme		PE header	Image R	RWE	
00401000	00038000	Unpackme		code	Image R	RWE	
00439000	0000A000	Unpackme	.rsrc	data, res			
00443000	00001000	Unpackme	.idata	imports			
00444000	00082000	Unpackme	hacnho	SFX			
004D0000	00103000						
005E0000	00005000						
005F0000	000E1000						
008F0000	00003000						
00900000	00008000						
00910000	00001000						
00920000	00001000						
00930000	00004000						
00940000	000F1000						
00A40000	0008D000						
00AD0000	00097000						
00B70000	00001000						
00B80000	00001000						

Actualize

View in Disassembler

Dump in CPU

Dump

Search

Set break-on-access

Set memory breakpoint on access

Set memory breakpoint on write

Enter

Ctrl+B

F2

_Nhan The F9, to OEP:

Address	Hex dump	Disassembly	Comment
00400DE1	. 6A 60	PUSH 60	OEP :D!
00400DE3	. 68 38B64200	PUSH Unpackme.0042B638	
00400DE8	. E8 47130000	CALL Unpackme.0040F134	
00400DED	. BF 94000000	MOV EDI, 94	
00400DF2	. 8BC7	MOV EAX, EDI	
00400DF4	. E8 07FCFFFF	CALL Unpackme.0040DAD0	
00400DF9	. 8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
00400DFC	. 8BF4	MOV ESI, ESP	
00400DFE	. 893E	MOV DWORD PTR DS:[ESI], EDI	
0040DE00	. 56	PUSH ESI	
0040DE01	. 90	NOP	
0040DE02	. E8 4A4A407C	CALL kernel32.GetVersionExA	[pVersionInformation = NULL GetVersionExA
0040DE07	. 8B4E 10	MOV ECX, DWORD PTR DS:[ESI+10]	
0040DE0A	. 890D 4C704300	MOV DWORD PTR DS:[43704C], ECX	
0040DE10	. 8B46 04	MOV EAX, DWORD PTR DS:[ESI+4]	
0040DE13	. A3 58704300	MOV DWORD PTR DS:[437058], EAX	
0040DE18	. 8B56 08	MOV EAX, DWORD PTR DS:[ESI+8]	
0040DE1B	. 8B56 08	MOV EAX, DWORD PTR DS:[ESI+8]	

ntdll.KiFastSystemCallRet

_Tien The dump with LordPE:



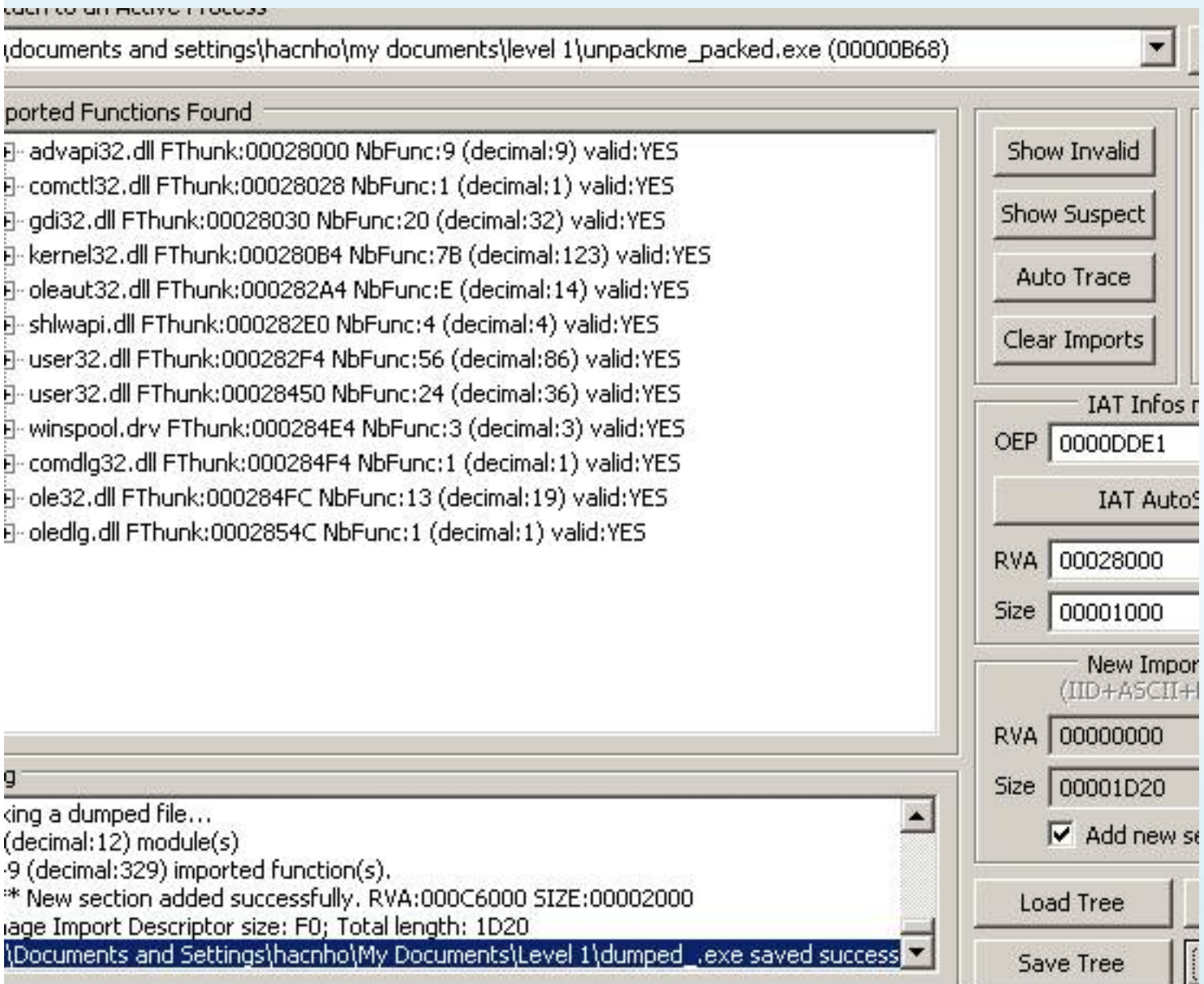
_Tim IAT table, in the CPU, IAT see this is not the mix, have easily IAT Full:

56	XOR ESI, ESI	
56	PUSH ESI	
8B3D 3C82420	MOV EDI, DWORD PTR DS:[42823C]	pModule = NULL
FFD7	CALL NEAR EDI	kernel32.GetModuleHandleA
66:8138 4D5A	CMP WORD PTR DS:[EAX], 5A4D	GetModuleHandleA

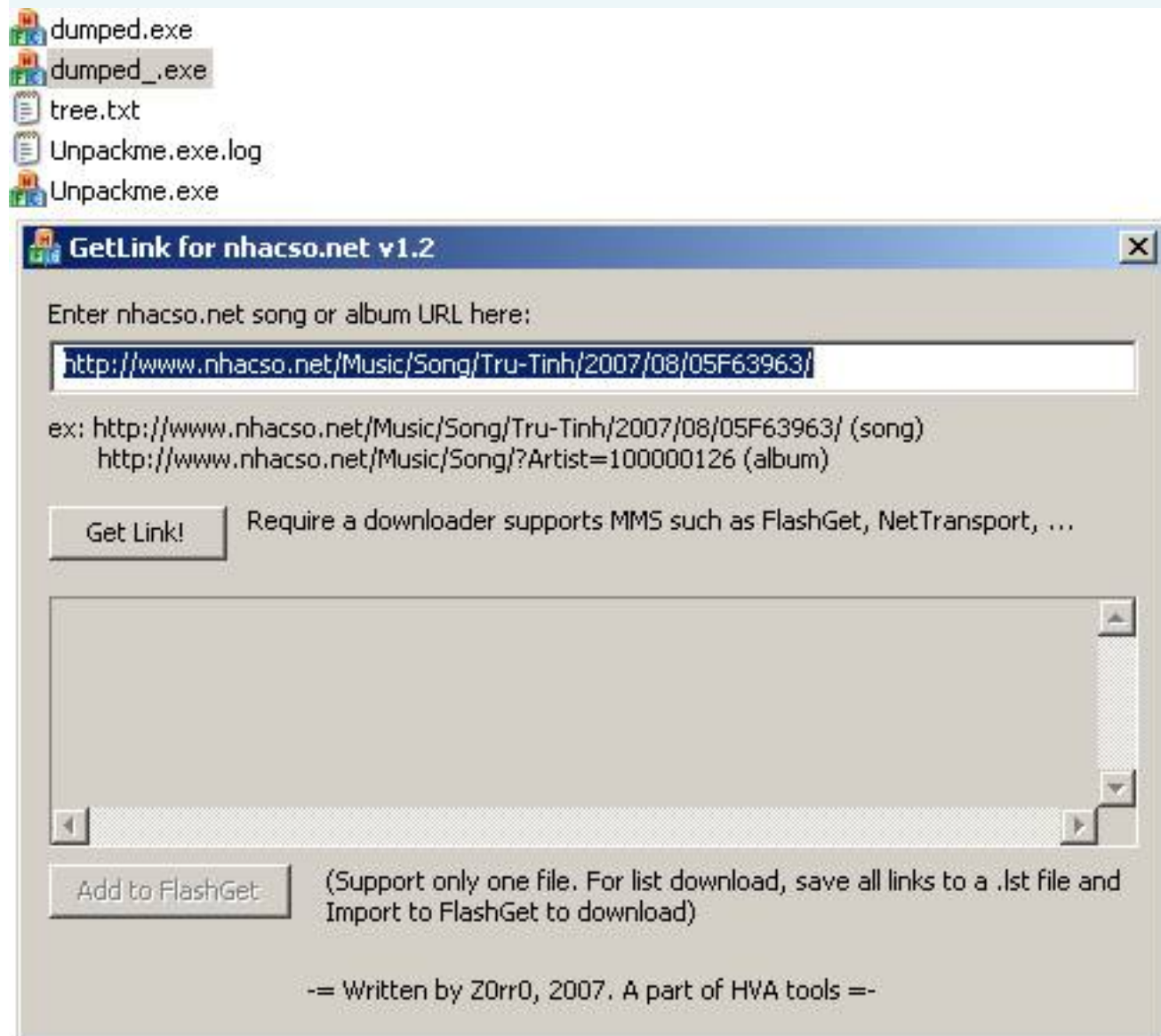
Print _Follow dump> Memory Address:

00428000	77DD7883	ADVAPI32.RegQueryValueExA
00428004	77DD761B	ADVAPI32.RegOpenKeyExA
00428008	77DFC123	ADVAPI32.RegDeleteKeyA
0042800C	77DFCAC3	ADVAPI32.RegEnumKeyA
00428010	77DFC41B	ADVAPI32.RegOpenKeyA
00428014	77DFCC10	ADVAPI32.RegQueryValueA
00428018	77DDEAF4	ADVAPI32.RegCreateKeyExA
0042801C	77DDEBE7	ADVAPI32.RegSetValueExA
00428020	77DD6BF0	ADVAPI32.RegCloseKey
00428024	00000000	
00428028	5D0B15DD	COMCTL32.InitCommonControls
0042802C	00000000	
00428030	77F184D4	GDI32.GetBkColor
00428034	77F18528	GDI32.GetTextColor
00428038	77F185C5	GDI32.CreateFontIndirect

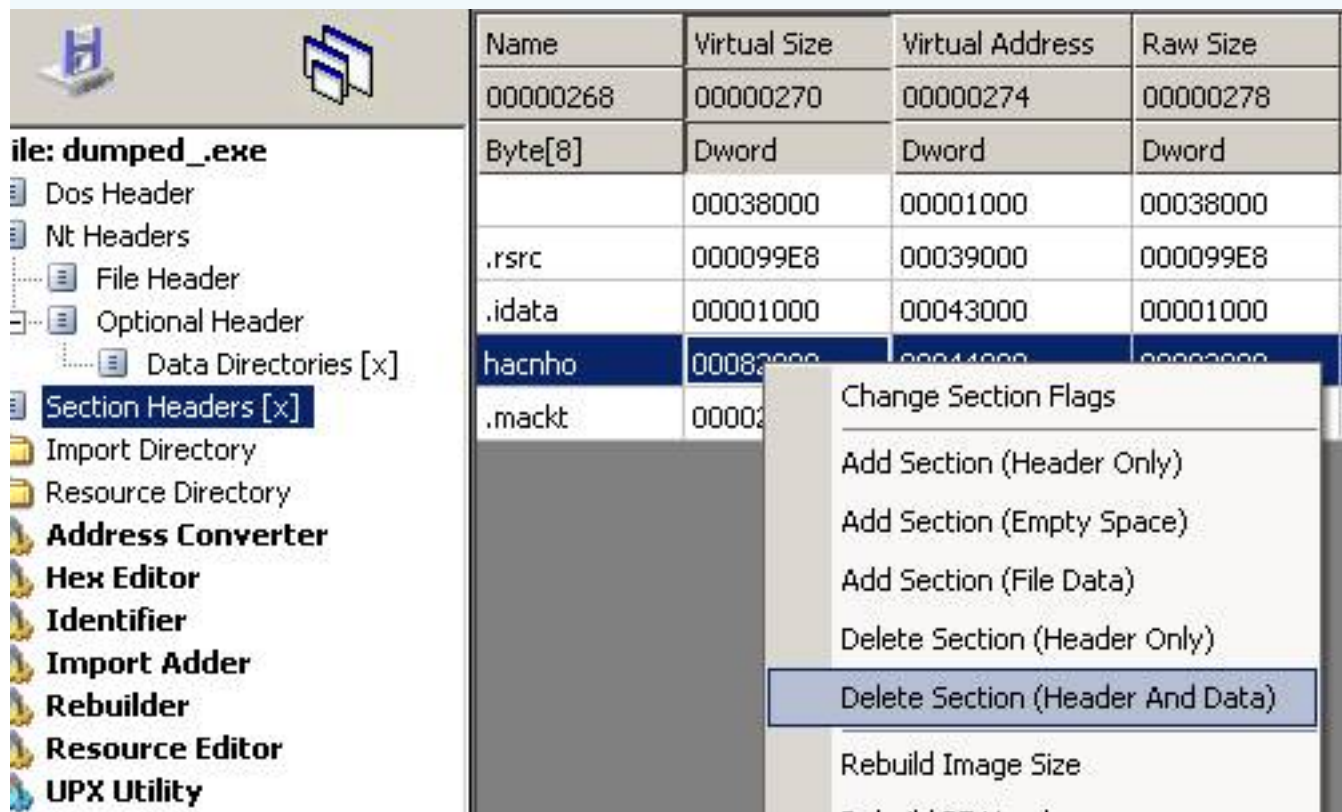
_Mo Imprec up:



_Gio Oral test:



_De File lightweight, can delete the section themida created:



_Unpack Done!

Any suggestions, comments or corrections please PM me via:

22-10-2007

Themida - then will go through

Themida 1.9.1.0 - Unpackme level 2

Tutorial # 2 @ 2007 by hacnho (hacnho@hotmail.com)

(Write the gift of friendship REA - all copy must be agreed by the author)

[Article title]: Themida 1.9.1.0 - Unpackme level 2
[Author]: **hacnho**
[Author home page]: <http://hvaonline.net>
[Target]: A Tool HVA
[Size]: 252 KBS
[Download Page]: hvaonline.net
[Packer]: Themida | WinLicense V 1.9.1.0 -> Oreans Technologies
[Compilation language]: Microsoft Visual C + + 7.0
[Tools]: The0DBG + hideToolz **fly** by, LordPE, ImportREC, peid0.94
[OS]: WinXP_SP2
[Greetz]: REA, Exetools, PEDIY, UnpackCN, snd and ARTeam Memberz

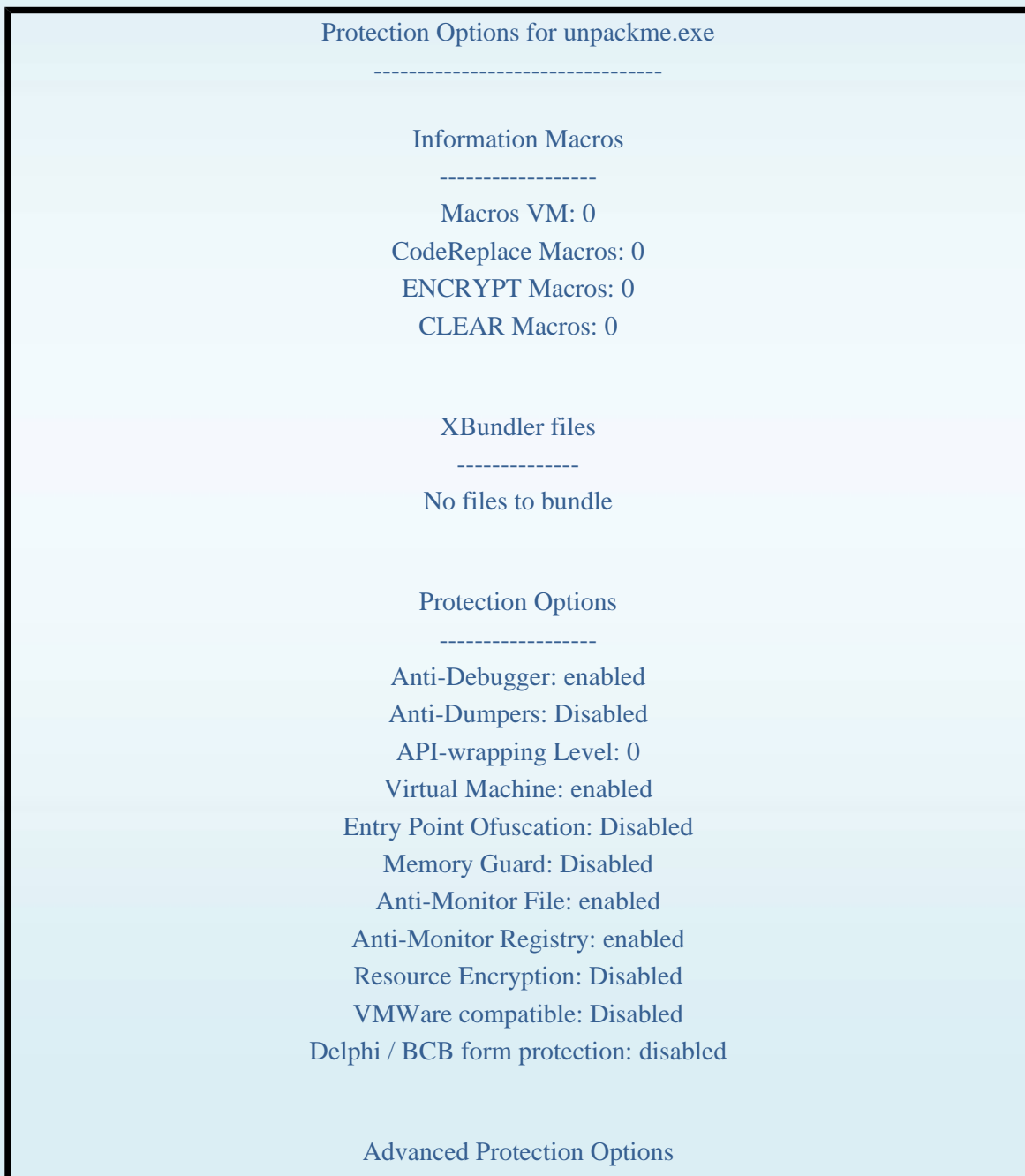
_Chao Of the property, we met each other again at 2 tut, tut also easy et, only

protect the increase of Themida 1.9.1.0 up. Ok let's go ... Also, remember that you

edit option plugin Phantom of the following home:



__Mo Themida 1.9.1.0 to create a new project and protect the following:



Encrypt Application: Disabled
. NET assemblies: Disabled
Dll plugin: Disabled
Active Context: Disabled
Section Last Name: hacnho

Compression

Application compression: enabled
Resources compression: enabled
SecureEngine compression: enabled

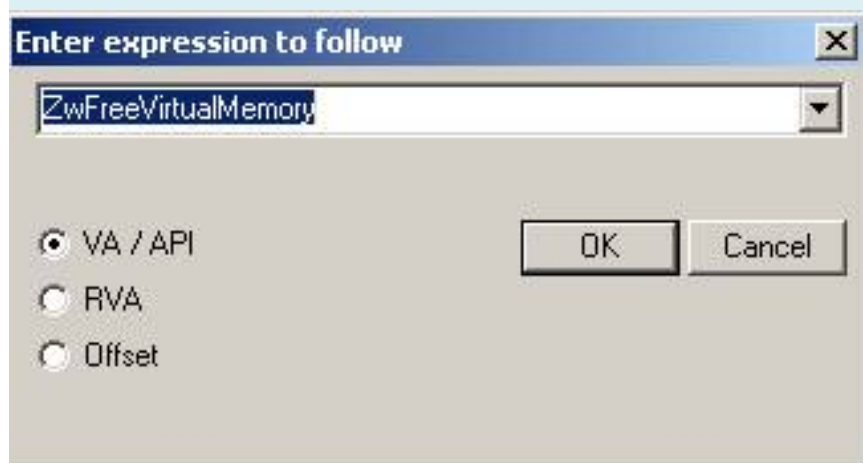
Virtual Machine Settings

Number of Virtual wrapped APIs: 0
Entry Point Virtualization: 0 instructions
Virtual Machine Processor: Mutable CISC processor
Number of CPUs: 1
Opcode Type: Static opcodes
Dynamic Opcode: Disabled

_Bay Hours to load target OllyDBG:

Address	Hex dump	Disassembly	Comment
00444014	B8 00000000	MOV EAX, 0	
00444019	60	PUSHAD	
0044401A	0BC0	OR EAX, EAX	
0044401C	74 68	JE SHORT Unpackm_.00444086	
0044401E	E8 00000000	CALL Unpackm_.00444023	
00444023	58	POP EAX	kernel32.7C816D4F
00444024	05 53000000	ADD EAX, 53	
00444029	8038 E9	CMP BYTE PTR DS:[EAX], 0E9	
0044402C	75 13	JNZ SHORT Unpackm_.00444041	
0044402E	61	POPAD	
0044402F	EB 45	JMP SHORT Unpackm_.00444076	
00444031	0B30 27404400	FIN TRUTE PTR DS.E4440271	

_Nhan Ctrl + G: ZwFreeVirtualMemory



_Toi And press F2 to RETN command:

Address	Hex dump	Disassembly	Comm
7C90DA48	B8 53000000	MOV EAX, 53	
7C90DA4D	BA 0003FE7F	MOV EDX, 7FFE0300	
7C90DA52	FF12	CALL NEAR DWORD PTR DS:[EDX]	
7C90DA54	C2 1000	RETN 10	
7C90DA57	90	NOP	
7C90DA58	90	NOP	
7C90DA59	90	NOP	

Nhan_Bay hours Shift + F9 and see the window of the Register:

Registers (FPU)		<	<	<
EAX	00000000			
ECX	0012FFB0			
EDX	7C90EB94	ntdll.KiFastSystemCallRet		
EBX	7FFD7000			
ESP	0012FFC4			
EBP	0012FFF0			
ESI	FFFFFFFF			
EDI	7C910738	ntdll.7C910738		
EIP	00444014	Unpackme.<ModuleEntryPoint>		
C 0	ES 0023	32bit	0(FFFFFFFF)	
P 1	CS 001B	32bit	0(FFFFFFFF)	
A 0	SS 0023	32bit	0(FFFFFFFF)	
Z 1	DS 0023	32bit	0(FFFFFFFF)	
S 0	FS 003B	32bit	7FFDF000(FFF)	
T 0	GS 0000	NULL		
D 0				
O 0	LastErr	ERROR_NO_IMPERSONATION_TOKEN (0000051D)		
EFL	00000246	(NO, NB, E, BE, NS, PE, GE, LE)		
ST0 empty	1.5738549221224447450e-3281			
ST1 empty	0.000000000000000040e-4933			
ST2 empty	-7.1911010359405521500e+757			
ST3 empty	-UNORM C020 00000017 E35BA6A8			
ST4 empty	5.6552512923172035960e-4925			
ST5 empty	-2.2633026551341701380e+1032			
ST6 empty	-UNORM CBC0 00010101 E35BA760			
ST7 empty	2.1065780380201079220e-3281			
	3 2 1 0	E S P U O Z D I		
FST 0000	Cond 0 0 0 0	Err 0 0 0 0 0 0 0 0	(GT)	
FCW 027F	Prec NEAR, 53	Mask	1 1 1 1 1 1	

_Vua Press F9 just look down the window stack, the more it this payment MS: P!

0012FF70	005089F8	Unpackm_.005089F8
0012FF74	005089A1	ASCII "ntice.sys"
0012FF78	00000000	

0012FF70	00508A02	Unpackm_.00508A02
0012FF74	00508996	ASCII "iceext.sys"

0012FF70	0050E4DD	Unpackm_.0050E4DD
0012FF74	0050E414	ASCII "Filem"
0012FF78	0000000E	

0012FF70	00510A78	Unpackm_.00510A78
0012FF74	00510A01	ASCII "REGMON"
0012FF78	0000000C	

0012FF70	00510A82	Unpackm_.00510A82
0012FF74	00510A08	ASCII "regsys"
0012FF78	0000000C	

0012FF70	00510A8C	Unpackm_.00510A8C
0012FF74	00510A0F	ASCII "sysregm"

0012FF70	00510A96	Unpackm_.00510A96
0012FF74	00510A17	ASCII "PROCMON"
0012FF78	00000007	

_Nhan Shift + F9 to when it is red again: P! (I count 20 times)

```

Registers (FPU)
EAX 00000000
ECX 0012FF2C
EDX 7C90EB94 ntdll.KiFastSystemCallRet
EBX 0240002C
ESP 0012FF30
EBP 0012FF4C
ESI 7C90DA48 ntdll.ZwFreeVirtualMemory
EDI 00418BFA Unpackme.00418BFA
EIP 7C90DA54 ntdll.7C90DA54

C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty -UNORM D0A8 01050104 00000000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 1.000000000000000000000000
ST7 empty 1.000000000000000000000000

      3 2 1 0      E S P U O Z D I
FST 4020 Cond 1 0 0 0 Err 0 0 1 0 0 0 0 0 (EQ)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1

```

_Gio Press F2 to remove breakpoint, press Alt + M to Memory window, set a

Breakpoint in the section on Access code:

00400000	00001000	Unpackme		PE header	Image R	RWE	
00401000	00038000	Unpackme		code	Image R	RWE	
00439000	0000A000	Unpackme	.rsrc	data, res			
00443000	00001000	Unpackme	.idata	imports			
00444000	00082000	Unpackme	hacnho	SFX			
004D0000	00103000						
005E0000	00005000						
005F0000	000E1000						
008F0000	00003000						
00900000	00008000						
00910000	00001000						
00920000	00001000						
00930000	00004000						
00940000	000F1000						
00A40000	0008D000						
00AD0000	00097000						
00B70000	00001000						
00B80000	00001000						

Actualize	
View in Disassembler	Enter
Dump in CPU	
Dump	
Search	Ctrl+B
Set break-on-access	F2
Set memory breakpoint on access	
Set memory breakpoint on write	

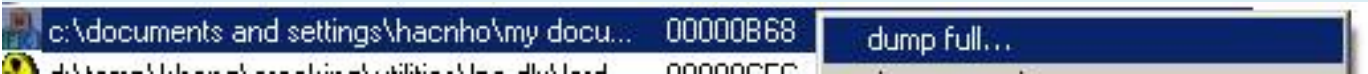
_Nhan The F9, to OEP:

Address	Hex dump	Disassembly	Comment
00400DE1	. 6A 60	PUSH 60	
00400DE3	. 68 38B64200	PUSH Unpackm_.0042B638	
00400DE8	. E8 47130000	CALL Unpackm_.0040F134	
00400DED	. BF 94000000	MOV EDI, 94	
0040DDF2	. 8BC7	MOV EAX, EDI	
0040DDF4	. E8 D7FCFFFF	CALL Unpackm_.0040DAD0	
0040DDF9	. 8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
0040DDFC	. 8BF4	MOV ESI, ESP	
0040DDFE	. 893E	MOV DWORD PTR DS:[ESI], EDI	
0040DE00	. 56	PUSH ESI	
0040DE01	. 90	NOP	
0040DE02	. E8 4A4A407C	CALL kernel32.GetVersionExA	
0040DE07	. 8B4E 10	MOV ECX, DWORD PTR DS:[ESI+10]	
0040DE0A	. 890D 4C704300	MOV DWORD PTR DS:[43704C], ECX	
0040DE10	. 8B46 04	MOV EAX, DWORD PTR DS:[ESI+4]	
0040DE13	. A3 58704300	MOV DWORD PTR DS:[437058], EAX	
0040DE18	. 8B56 08	MOV EDX, DWORD PTR DS:[ESI+8]	
0040DE1B	. 8915 5C704300	MOV DWORD PTR DS:[43705C], EDX	
0040DE21	. 8B7C 0C	MOV ESI, DWORD PTR DS:[ESI+0C]	

[pVersionInformation = FA60D71E
GetVersionExA

Unpackm_.004ECC5C

_Tien The dump with LordPE:



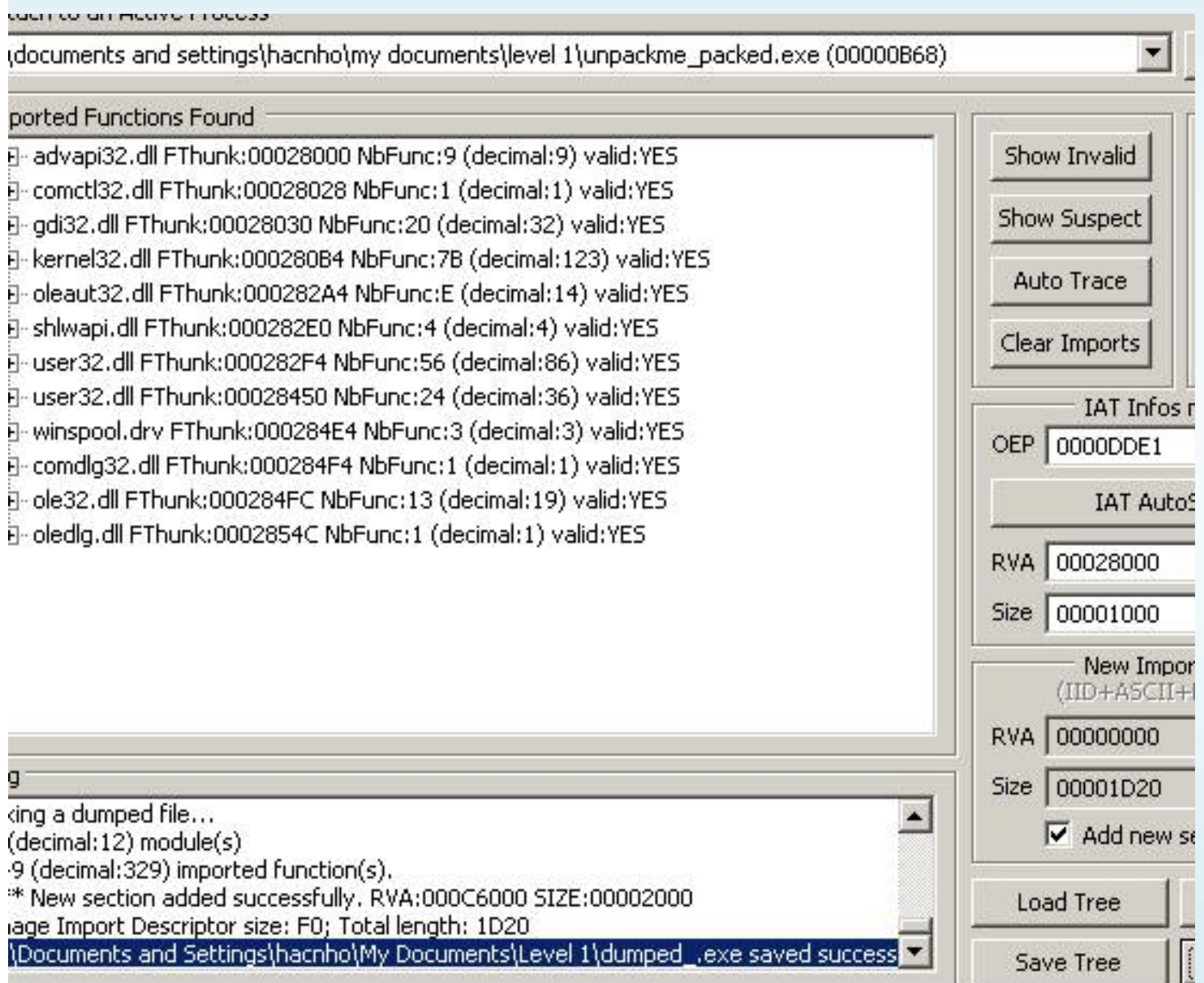
_Tim IAT table, in the CPU, IAT see this is not the mix, have easily IAT Full:

33fb	XOR ESI, ESI	[pModule = NULL kernel32.GetModuleHandleA GetModuleHandleA
56	PUSH ESI	
8b3d 3c82420	MOV EDI, DWORD PTR DS:[42823C]	
FFD7	CALL NEAR EDI	
66:8138 4d5a	CMP WORD PTR DS:[EAX], 5A4D	

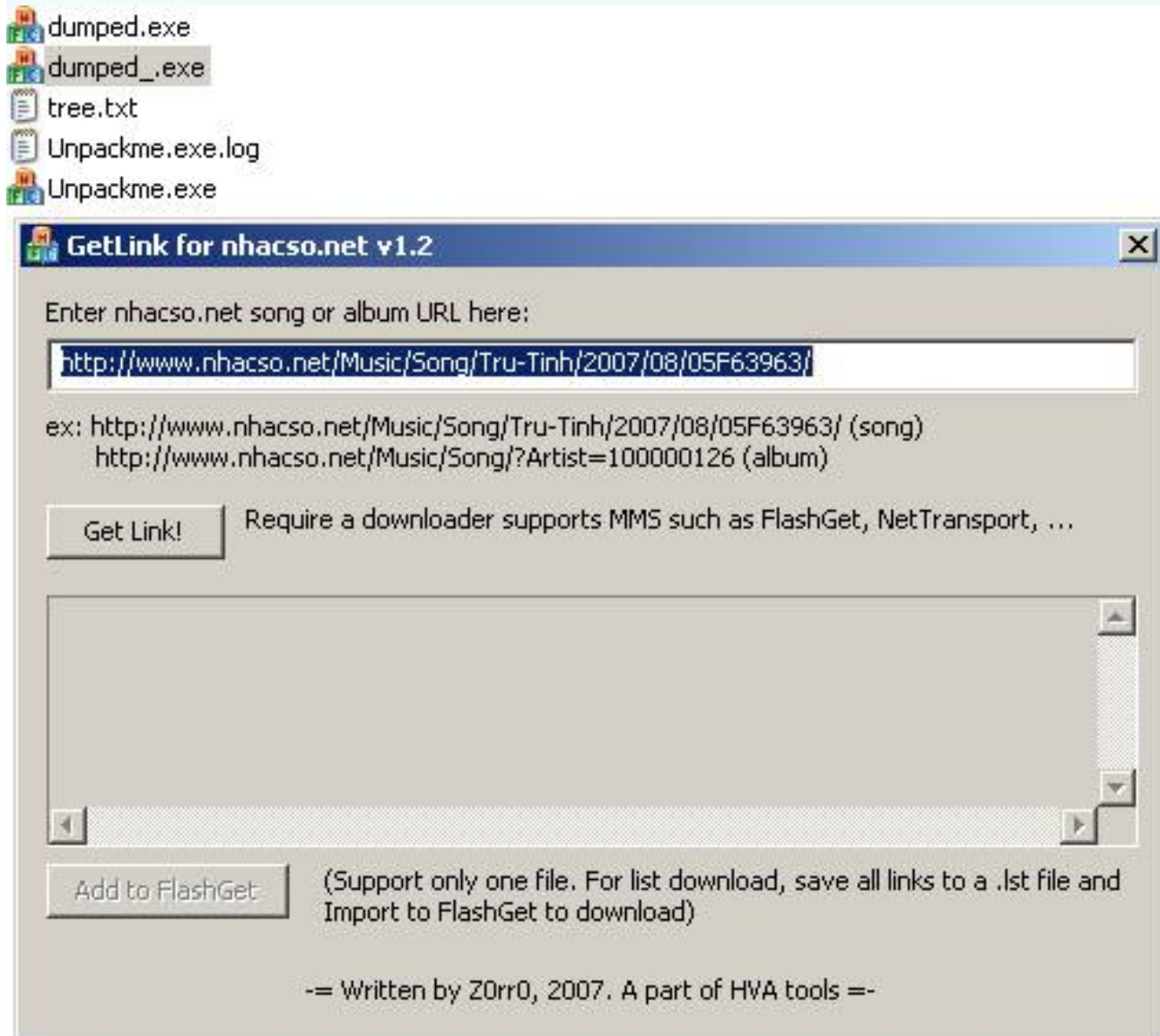
Print _Follow dump> Memory Address:

00428000	77DD7883	ADVAPI32.RegQueryValueExA
00428004	77DD761B	ADVAPI32.RegOpenKeyExA
00428008	77DFC123	ADVAPI32.RegDeleteKeyA
0042800C	77DFCAC3	ADVAPI32.RegEnumKeyA
00428010	77DFC41B	ADVAPI32.RegOpenKeyA
00428014	77DFCC10	ADVAPI32.RegQueryValueA
00428018	77DDEAF4	ADVAPI32.RegCreateKeyExA
0042801C	77DDEBE7	ADVAPI32.RegSetValueExA
00428020	77DD6BF0	ADVAPI32.RegCloseKey
00428024	00000000	
00428028	5D0B15DD	COMCTL32.InitCommonControls
0042802C	00000000	
00428030	77F184D4	GDI32.GetBkColor
00428034	77F18528	GDI32.GetTextColor
00428038	77F183CF	GDI32.CreateDeviceContextFromHandle

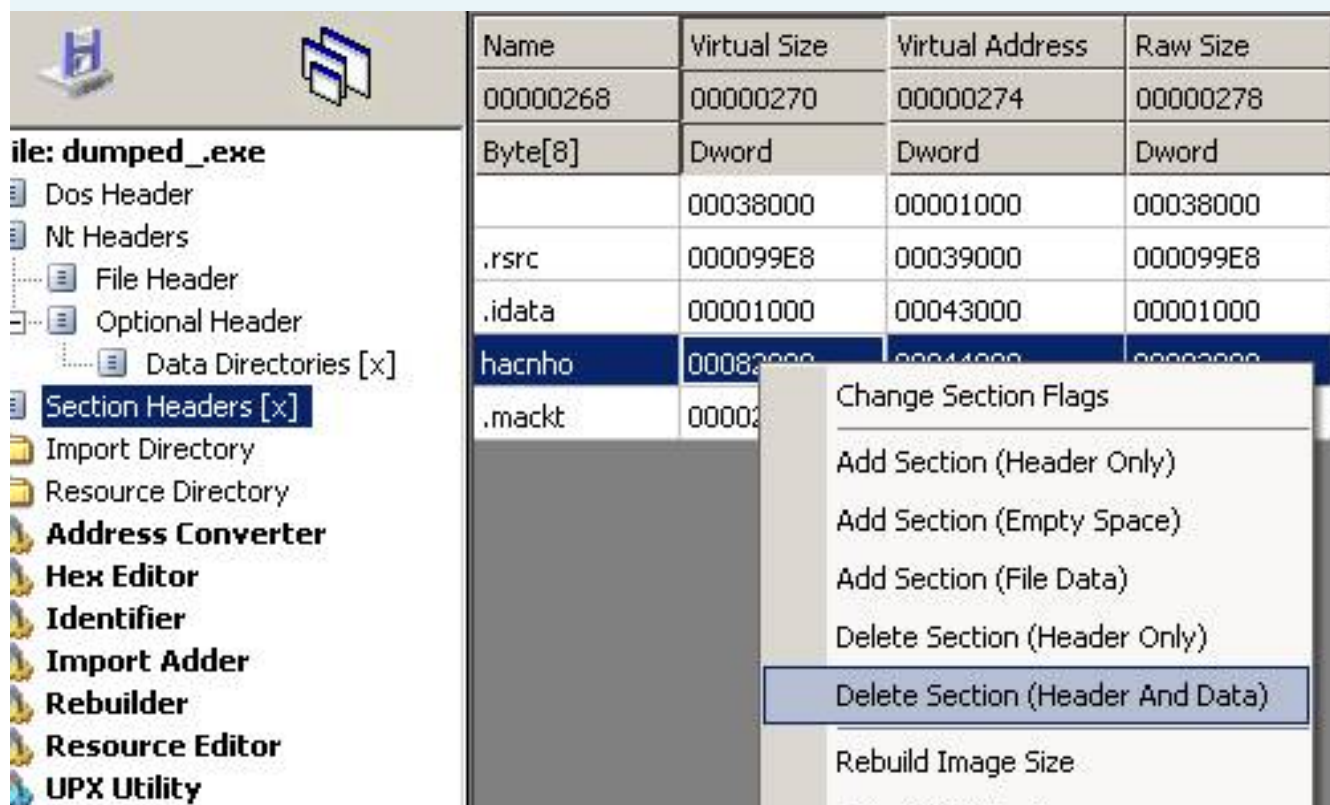
_Mo Imprec up:



_Gio Oral test:



_De File lightweight, can delete the section themida created:



_Unpack Done!

Any suggestions, comments or corrections please PM me via:

hacnho@hotmail.com

22-10-2007

Themida - then will go through

Themida 1.9.1.0 - Unpackme level 3

Tutorial # 3 @ 2007 by hacnho (hacnho@hotmail.com)

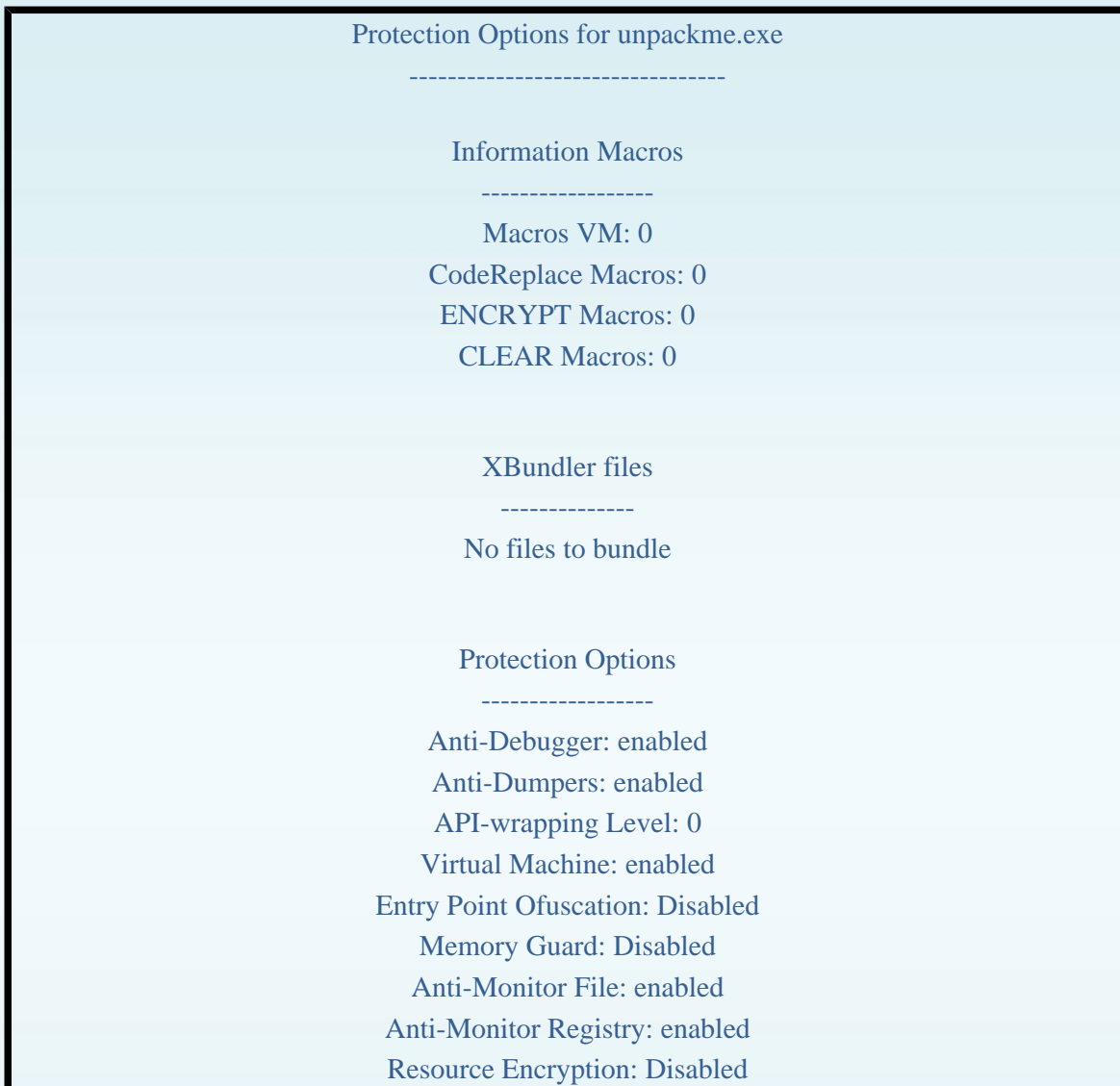
(Write the gift of friendship REA - all copy must be agreed by the author)

[Article title]: Themida 1.9.1.0 - Unpackme level 3
[Author]: **hacnho**
[Author home page]: <http://hvaonline.net>
[Target]: A Tool HVA
[Size]: 252 KBS
[Download Page]: hvaonline.net
[Packer]: Themida | WinLicense V 1.9.1.0 -> Oreans Technologies
[Compilation language]: Microsoft Visual C + + 7.0
[Tools]: The0DBG + hideToolz **fly** by, LordPE, ImportREC, peid0.94
[OS]: WinXP_SP2
[Greetz]: REA, Exetools, PEDIY, UnpackCN, snd and ARTeam Memberz

3 _Trong tut this we will add an increased level are selected Anti dump, and how to

solve it ...

__Mo Themida 1.9.1.0 to create a new project and protect the following:



VMWare compatible: Disabled
Delphi / BCB form protection: disabled

Advanced Protection Options

Encrypt Application: Disabled
. NET assemblies: Disabled
 Dll plugin: Disabled
 Active Context: Disabled
 Section Last Name: hacnho

Compression

Application compression: enabled
Resources compression: enabled
SecureEngine compression: enabled

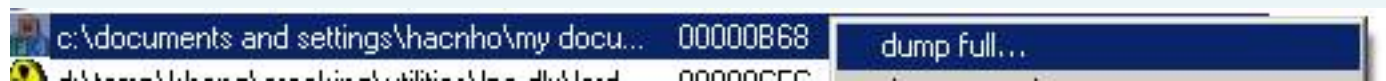
Virtual Machine Settings

Number of Virtual wrapped APIs: 0
Entry Point Virtualization: 0 instructions
Virtual Machine Processor: Mutable CISC processor
 Number of CPUs: 1
 Opcode Type: Static opcodes
 Dynamic Opcode: Disabled

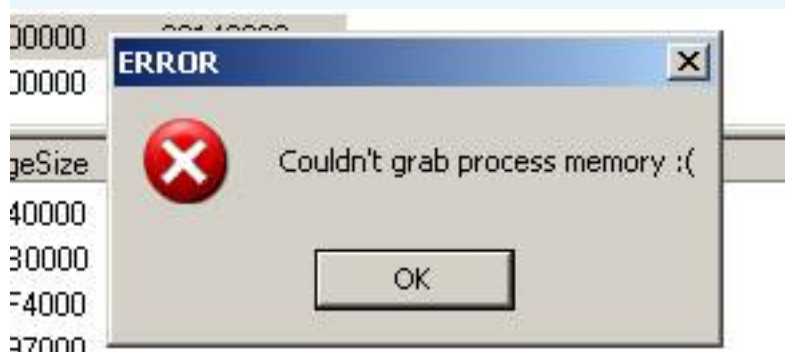
_Bay Hours to load target OllyDBG, conducting the search tut2 such as OEP. After

the OEP.

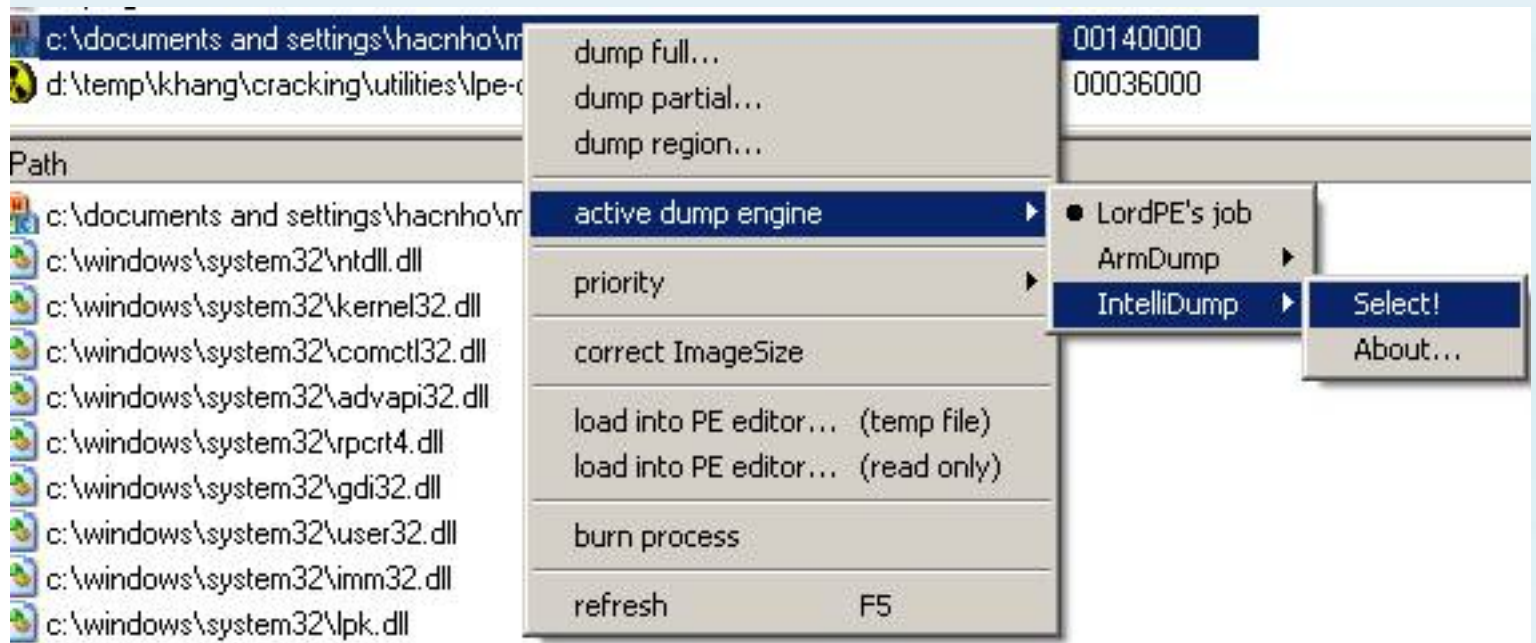
_Tien The dump with LordPE:



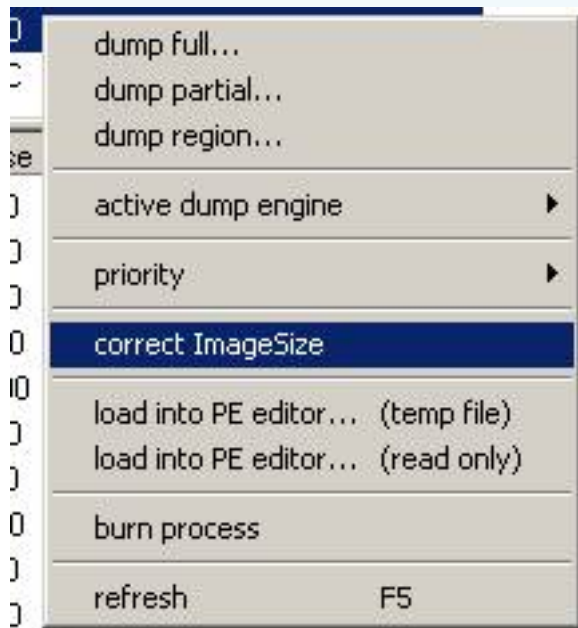
_Va Receive:



_Ok Now, click in the process need to dump:



_Sau Then right click to choose:



_Tien The dump and fix IAT normal.

Any suggestions, comments or corrections please PM me via:

hacnho@hotmail.com

22-10-2007

Themida - then will go through


Themida 1.9.1.0 - Unpackme level 4

Tutorial # 4 @ 2007 by hacnho (hacnho@hotmail.com)

(Write the gift of friendship REA - all copy must be agreed by the author)


[Article title]: Themida 1.9.1.0 - Unpackme level 4
[Author]: **hacnho**
[Author home page]: <http://hvaonline.net>
[Target]: A Tool HVA
[Size]: 252 KBS
[Download Page]: hvaonline.net
[Packer]: Themida | WinLicense V 1.9.1.0 -> Oreans Technologies
[Compilation language]: Microsoft Visual C + + 7.0
[Tools]: The0DBG + hideToolz **fly** by, LordPE, ImportREC, peid0.94
[OS]: WinXP_SP2
[Greetz]: REA, Exetools, PEDIY, UnpackCN, snd and ARTeam Memberz


_Trong Tut 4 nay we will process which Oreans called API-wrapping. I will at the API level 1-wrapping.


Protection Options

Advanced Anti-Debugger

☒ Enable Protection

Advanced API-Wrapping

Level 1


Compression

☒ Application
☒ Resources
☒ SecureEngine

Anti Dumpers

☒ Enable Protection


Anti-Patching


File Patching

Monitor Blockers

☒ Files Monitors
☒ Registry Monitors

Entry Point Obfuscation


☐ Enable Protection


Metamorph Security

☒ Enable Protection


Delphi/BCB Form Protection

☐ Enable Protection

Resources Encryption

☒ Enable Encryption

Memory Guard

☒ Enable Protection

VMWare/Virtual PC

☐ Compatible

When Debugger Found

Display Message

Mo Themida 1.9.1.0 to create a new project and protect the following:

Protection Options for unpackme.exe

Information Macros

Macros VM: 0

CodeReplace Macros: 0

ENCRYPT Macros: 0

CLEAR Macros: 0

XBundler files

No files to bundle

Protection Options

Anti-Debugger: enabled

Anti-Dumpers: enabled
API-wrapping Level: 1
Virtual Machine: enabled
Entry Point Ofuscation: Disabled
Memory Guard: enabled
Anti-Monitor File: enabled
Anti-Monitor Registry: enabled
Resource Encryption: enabled
VMWare compatible: Disabled
Delphi / BCB form protection: disabled

Advanced Protection Options

Encrypt Application: Disabled
. NET assemblies: Disabled
Dll plugin: Disabled
Active Context: Disabled
Section Last Name: hacnho

Compression

Application compression: enabled
Resources compression: enabled
SecureEngine compression: enabled

Virtual Machine Settings

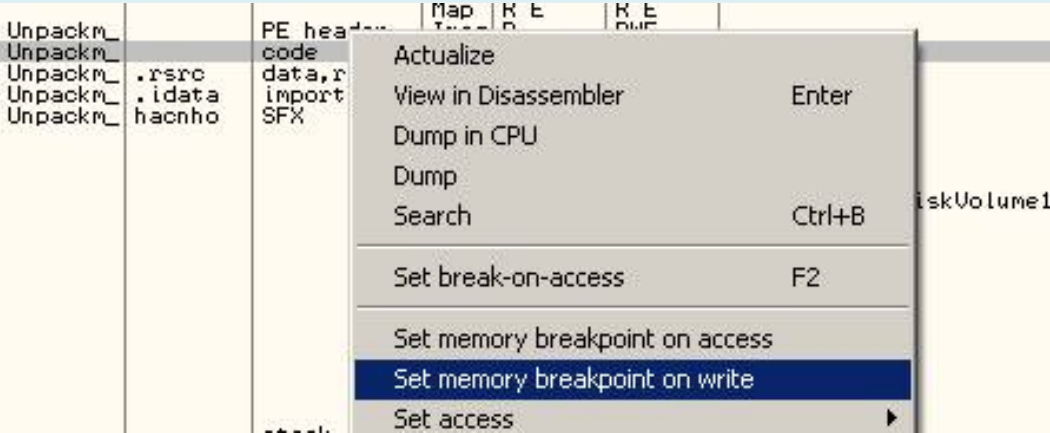
Number of Virtual wrapped APIs: 0
Entry Point Virtualization: 0 instructions
Virtual Machine Processor: Mutable CISC processor
Number of CPUs: 1
Opcode Type: Static opcodes
Dynamic Opcode: Disabled

_Bay Hours to load target OllyDBG:

Address	Hex dump	Disassembly	Commen
00444014	B8 00000000	MOV EAX, 0	
00444019	60	PUSHAD	
0044401A	0BC0	OR EAX, EAX	
0044401C	74 68	JE SHORT Unpackm_.00444086	
0044401E	E8 00000000	CALL Unpackm_.00444023	
00444023	58	POP EAX	
00444024	05 53000000	ADD EAX, 53	
00444029	8038 E9	CMP BYTE PTR DS:[EAX], 0E9	

_Khong Tuts like 3 times before this API-wrapping, to fix Magic Jump is the first step we

press Alt + M to put on a Memory Write Breakpoint in code section:



_Nhan Shift + F9:

Address	Hex dump	Disassembly	Comment
0051FF00	F3:A4	REP MOVS BYTE PTR ES:[EDI], BYTE PTR DS	
0051FF02	C685 8D02EB05	MOV BYTE PTR SS:[EBP+5EB028D], 56	
0051FF09	68 396D1FD4	PUSH D41F6D39	
0051FF0E	FFB5 1D0FEB05	PUSH DWORD PTR SS:[EBP+5EB0F1D]	
0051FF14	8D85 2672EE05	LEA EAX, DWORD PTR SS:[EBP+5EE7226]	
0051FF1A	FFD0	CALL NEAR EAX	

_Nhan F7, F8 and then press Shift + F9:

Address	Hex dump	Disassembly	Comment
00527642	8908	MOV DWORD PTR DS:[EAX], ECX	
00527644	AD	LODS DWORD PTR DS:[ESI]	
00527645	C746 FC 000000	MOV DWORD PTR DS:[ESI-4], 0	
0052764C	89B5 F517EB05	MOV DWORD PTR SS:[EBP+5EB17F5], ESI	Unpackm_.0052059A
00527652	83F8 FF	CMP EAX, -1	
00527655	0F85 20000000	JNZ Unpackm_.0052767B	
0052765B	813E DDDDDDDD	CMP DWORD PTR DS:[ESI], DDDDDDDD	
00527661	0F85 14000000	JNZ Unpackm_.0052767B	

_Tiep To Shift + F9:

Address	Hex dump	Disassembly	Comment
005276FF	AA	STOS BYTE PTR ES:[EDI]	
00527700	807F FF E9	CMP BYTE PTR DS:[EDI-1], 0E9	
00527704	0F85 18000000	JNZ Unpackm_.00527722	
0052770A	83BD DB7CF205	CMP DWORD PTR SS:[EBP+5F27CDB], 0	
00527711	0F84 08000000	JE Unpackm_.0052771F	
00527717	8D9D B5B1F105	LEA EBX, DWORD PTR SS:[EBP+5F1B1B5]	
0052771D	FFD3	CALL NEAR EBX	Unpackm_.0051962E

_Tiep To Shift + F9:

0527720	AB	STOS DWORD PTR ES:[EDI]	
052772E	AD	LDS DWORD PTR DS:[ESI]	
052772F	C746 FC 000000	MOV DWORD PTR DS:[ESI-4], 0	
0527736	E9 11FFFFFF	JMP Unpackm_.0052764C	

_Sau Then pressing F8 trace down:

0052772D	AB	STOS DWORD PTR ES:[EDI]	
0052772E	AD	LODS DWORD PTR DS:[ESI]	
0052772F	C746 FC 000000	MOV DWORD PTR DS:[ESI-4], 0	
00527736	E9 11FFFFFF	JMP Unpackm_.0052764C	
0052773B	89B5 F517EB05	MOV DWORD PTR SS:[EBP+5EB17F51], ESI	Unpackm_.005205A2
00527741	52	PUSH EDX	ntdll.KiFastSystemCallRet
00527742	68 00800000	PUSH 8000	

_Toi Here:

Address	Hex dump	Disassembly	Comment
0052764C	89B5 F517EB05	MOV DWORD PTR SS:[EBP+5EB17F51], ESI	Unpackm_.005205A2
00527652	83F8 FF	CMP EAX, -1	
00527655	0F85 20000000	JNZ Unpackm_.0052767B	
0052765B	813E 00000000	CMP DWORD PTR DS:[ESI], 00000000	
00527661	0F85 14000000	JNZ Unpackm_.0052767B	

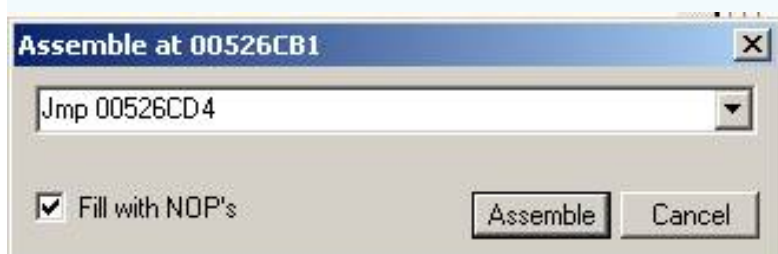
_Nhan F8 for more to come:

Address	Hex dump	Disassembly	Comment
0052764C	89B5 F517EB05	MOV DWORD PTR SS:[EBP+5EB17F51], ESI	Unpackm_.005205AE
00527652	83F8 FF	CMP EAX, -1	
00527655	0F85 20000000	JNZ Unpackm_.0052767B	
0052765B	813E 00000000	CMP DWORD PTR DS:[ESI], 00000000	
00527661	0F85 14000000	JNZ Unpackm_.0052767B	
00527667	C706 00000000	MOV DWORD PTR DS:[ESI], 0	

_Chúc You happy to Magic Jump;). Một quá: P!

Address	Hex dump	Disassembly	Comment
00526C27	C785 7931EB05	MOV DWORD PTR SS:[EBP+5EB3179], 0	
00526C31	C785 9D05EB05	MOV DWORD PTR SS:[EBP+5EB059D], 0	
00526C3B	83BD 0B7CF205	CMP DWORD PTR SS:[EBP+5F27CDB], 0	
00526C42	0F84 08000000	JE Unpackm_.00526C50	
00526C48	8D9D B4A7F105	LEA EBX, DWORD PTR SS:[EBP+5F1A7B4]	
00526C4E	FFD3	CALL NEAR EBX	Unpackm_.0051962E
00526C50	FF85 1D07EB05	INC DWORD PTR SS:[EBP+5EB071D]	
00526C56	83BD 1D07EB05	CMP DWORD PTR SS:[EBP+5EB071D], 64	
00526C5D	0F82 72000000	JB Unpackm_.00526CD5	
00526C63	C785 1D07EB05	MOV DWORD PTR SS:[EBP+5EB071D], 1	
00526C6D	60	PUSHAD	
00526C6E	8D85 6965F205	LEA EAX, DWORD PTR SS:[EBP+5F26569]	
00526C74	FFD0	CALL NEAR EAX	
00526C76	8DB5 AB7DF205	LEA ESI, DWORD PTR SS:[EBP+5F27DAB]	
00526C7C	8DBD 589BF205	LEA EDI, DWORD PTR SS:[EBP+5F29B58]	
00526C82	2BFE	SUB EDI, ESI	Unpackm_.005205AE
00526C84	8BD7	MOV EDX, EDI	Unpackm_.004117C4
00526C86	8BBD 5932EB05	MOV EDI, DWORD PTR SS:[EBP+5EB3259]	
00526C8C	B9 FFFFFFFF	MOV ECX, -1	
00526C91	33C0	XOR EAX, EAX	
00526C93	8A06	MOV AL, BYTE PTR DS:[ESI]	
00526C95	32C1	XOR AL, CL	
00526C97	46	INC ESI	Unpackm_.005205AE
00526C98	8B0487	MOV EAX, DWORD PTR DS:[EDI+EAX*4]	
00526C9B	C1E9 08	SHR ECX, 8	
00526C9E	33C8	XOR ECX, EAX	
00526CA0	4A	DEC EDX	ntdll.KiFastSystemCallRet
00526CA1	0F85 EAFFFFFF	JNZ Unpackm_.00526C91	
00526CA7	8BC1	MOV EAX, ECX	
00526CA9	F7D0	NOT EAX	
00526CAB	3985 D91AEB05	CMP DWORD PTR SS:[EBP+5EB1AD9], EAX	
00526CB1	0F84 1D000000	JE Unpackm_.00526CD4	Magic Jump
00526CB7	B8 00000000	MOV EAX, 0	
00526CBC	0BC0	OR EAX, EAX	

_Patch To:



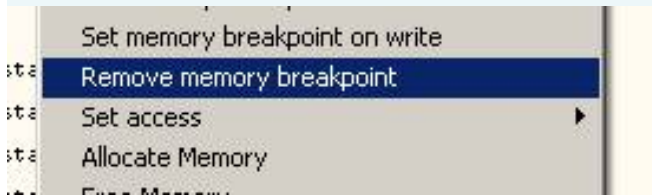
00526CB0	EB 21	JMP SHORT Unpackm_.00526CD4	Magic Jump
00526CB1	90	NOP	
00526CB3	90	NOP	
00526CB4	90	NOP	
00526CB5	90	NOP	
00526CB6	90	NOP	
00526CB7	B8 00000000	MOV EAX, 0	
00526CB8	0BC0	OR EAX, EAX	

_Cuon Down some orders jumped 4 meeting. Submit it always nhé: D!

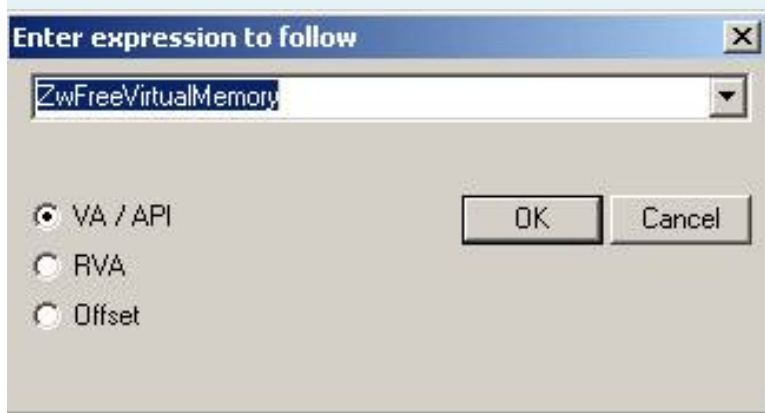
00526DE3	03C1	ADD EAX, ECX	Unpackm_.005205AE
00526DE5	5E	POP ESI	
00526DE6	83BD 1D02EB05	CMP DWORD PTR SS:[EBP+5EB021D], 1	kernel32.7C800000
00526DE8	0F84 39000000	JE Unpackm_.00526E2C	
00526DF3	3B8D 790EEB05	CMP ECX, DWORD PTR SS:[EBP+5EB0E79]	USER32.77D40000
00526DF9	0F84 2D000000	JE Unpackm_.00526E2C	
00526DFF	3B8D 6911EB05	CMP ECX, DWORD PTR SS:[EBP+5EB1169]	ADVAPI32.77DD0000
00526E05	0F84 21000000	JE Unpackm_.00526E2C	
00526E08	3B8D C52FEB05	CMP ECX, DWORD PTR SS:[EBP+5EB2FC5]	Unpackm_.0051962E
00526E11	0F84 15000000	JE Unpackm_.00526E2C	
00526E17	8D9D 2C93F205	LEA EBX, DWORD PTR SS:[EBP+5F2932C]	Unpackm_.0051962E
00526E1D	FFD3	CALL NEAR EBX	
00526E1F	8BF8	MOV EDI, EAX	
00526E21	8985 091AEF05	MOV DWORD PTR SS:[EBP+5EB1A09], EAX	

00526DE6	83BD 1D02EB05	CMP DWORD PTR SS:[EBP+5EB021D], 1	kernel32.7C800000
00526DE8	90	NOP	
00526DEE	90	NOP	USER32.77D40000
00526DEF	90	NOP	
00526DF0	90	NOP	ADVAPI32.77DD0000
00526DF1	90	NOP	
00526DF2	90	NOP	Unpackm_.0051962E
00526DF3	3B8D 790EEB05	CMP ECX, DWORD PTR SS:[EBP+5EB0E79]	
00526DF9	90	NOP	kernel32.7C800000
00526DFA	90	NOP	
00526DFB	90	NOP	USER32.77D40000
00526DFC	90	NOP	
00526DFD	90	NOP	ADVAPI32.77DD0000
00526DFE	90	NOP	
00526DFF	3B8D 6911EB05	CMP ECX, DWORD PTR SS:[EBP+5EB1169]	Unpackm_.0051962E
00526E05	90	NOP	
00526E06	90	NOP	kernel32.7C800000
00526E07	90	NOP	
00526E08	90	NOP	USER32.77D40000
00526E09	90	NOP	
00526E0A	90	NOP	ADVAPI32.77DD0000
00526E0B	90	NOP	
00526E08	3B8D C52FEB05	CMP ECX, DWORD PTR SS:[EBP+5EB2FC5]	Unpackm_.0051962E
00526E11	90	NOP	
00526E12	90	NOP	kernel32.7C800000
00526E13	90	NOP	
00526E14	90	NOP	USER32.77D40000
00526E15	90	NOP	
00526E16	90	NOP	ADVAPI32.77DD0000
00526E17	8D9D 2C93F205	LEA EBX, DWORD PTR SS:[EBP+5F2932C]	
00526E1D	FFD3	CALL NEAR EBX	Unpackm_.0051962E
00526E1F	8BF8	MOV EDI, EAX	
00526E21	8985 091AEF05	MOV DWORD PTR SS:[EBP+5EB1A09], EAX	

_Nhan Alt + M, remove breakpoint:



__Nhan Ctrl + G: ZwFreeVirtualMemory



__Toi And press F2 to RETN command:

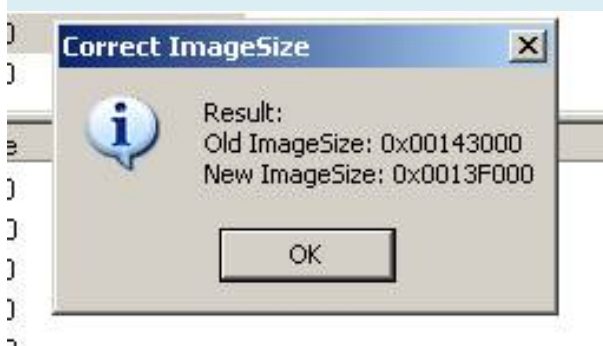
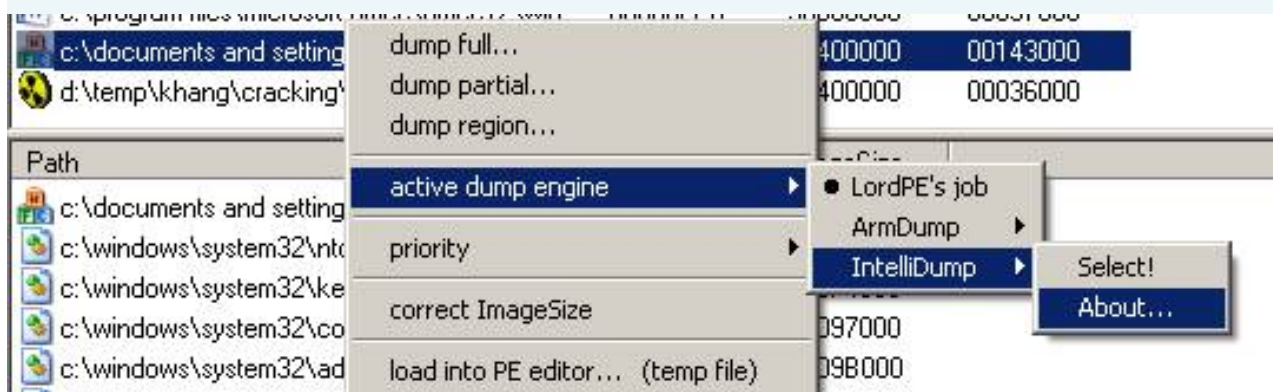
Address	Hex	Disassembly
7C90DA48	B8 53000000	MOV EAX, 53
7C90DA4D	BA 0003FE7F	MOV EDX, 7FFE0300
7C90DA52	FF12	CALL NEAR DWORD PTR DS:[EDX]
7C90DA54	C2 1000	RETN 10
7C90DA57	90	NOP
7C90DA58	90	NOP
7C90DA59	90	NOP
7C90DA5A	90	NOP

_Nhan F9 14 times, then put breakpoint, press Alt + M, press F2 at the code section. Press F9.

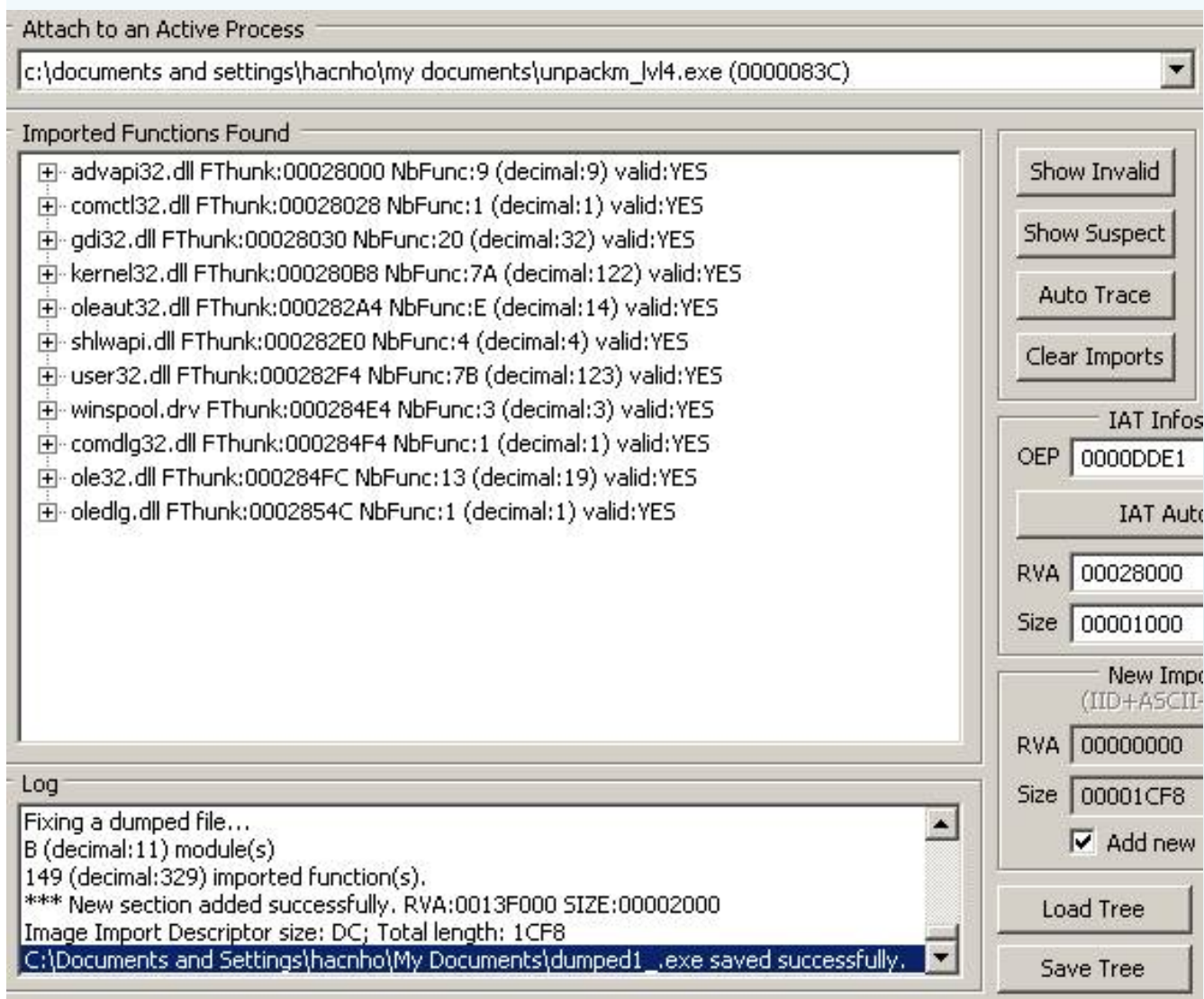
See what's nhé: D!

Address	Hex dump	Disassembly	Comment
00400DE1	. 6A 60	PUSH 60	
00400DE3	. 68 38B64200	PUSH Unpackm_.0042B638	
00400DE8	. E8 47130000	CALL Unpackm_.0040F134	
00400DED	. BF 94000000	MOV EDI, 94	
00400DF2	. 8BC7	MOV EAX, EDI	
00400DF4	. E8 D7FCFFFF	CALL Unpackm_.0040DAD0	
00400DF9	. 8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
00400DFC	. 8BF4	MOV ESI, ESP	
00400DFE	. 893E	MOV DWORD PTR DS:[ESI], EDI	
00400E00	. 56	PUSH ESI	
00400E01	. 90	NOP	
00400E02	. E8 4A4A407C	CALL kernel32.GetVersionExA	pVersionInformation = Unpackm_.004F08C6
00400E07	. 8B4E 10	MOV ECX, DWORD PTR DS:[ESI+10]	GetVersionExA
00400E0A	. 890D 4C704300	MOV DWORD PTR DS:[43704C], ECX	ntdll.KiFastSystemCallRet
00400E10	. 8B46 04	MOV EAX, DWORD PTR DS:[ESI+4]	
00400E13	. A3 58704300	MOV DWORD PTR DS:[437058], EAX	
00400E18	. 8B56 08	MOV EDX, DWORD PTR DS:[ESI+8]	
00400E1B	. 8915 5C704300	MOV DWORD PTR DS:[43705C], EDX	Unpackm_.004E94BA
00400E21	. 8B76 0C	MOV ESI, DWORD PTR DS:[ESI+C]	
00400E24	. 81E6 FF7F0000	AND ESI, 7FFF	
00400E2A	. 8935 50704300	MOV DWORD PTR DS:[437050], ESI	Unpackm_.004F08C6
00400E30	. 83F9 02	CMP ECX, 2	
00400E33	√ 74 0C	JE SHORT Unpackm_.0040DE41	
00400E35	. 81CE 00800000	OR ESI, 8000	
00400E38	. 8935 50704300	MOV DWORD PTR DS:[437050], ESI	Unpackm_.004F08C6
00400E41	> C1E0 08	SHL EAX, 8	
00400E44	. 03C2	ADD EAX, EDX	Unpackm_.004E94BA
00400E46	. A3 54704300	MOV DWORD PTR DS:[437054], EAX	
00400E4B	. 33F6	XOR ESI, ESI	Unpackm_.004F08C6
00400E4D	. 56	PUSH ESI	pModule = "rhj_4".uyrp9=0.0%unl3-*\$+_\$.a%0"
00400E4E	. 8B3D 3C824200	MOV EDI, DWORD PTR DS:[42823C]	kernel32.GetModuleHandleA
00400E54	. FFD7	CALL NEAR EDI	GetModuleHandleA
00400E56	. 66:8138 4D5A	CMP WORD PTR DS:[EAX], 5A4D	
00400E5B	√ 75 1F	JNZ SHORT Unpackm_.0040DE7C	
00400E5D	. 8B4E 10	MOV ECX, DWORD PTR DS:[ESI+10]	

_Mo LordPE:



_Imprec:



_Test Test, work fine!

Any suggestions, comments or corrections please PM me via:

hacnho@hotmail.com

22-10-2007

Themida - then will go through

Themida 1.9.1.0 - Unpackme level 5

Tutorial # 5 @ 2007 by hacnho (hacnho@hotmail.com)

(Write the gift of friendship REA - all copy must be agreed by the author)

[Article title]: Themida 1.9.1.0 - Unpackme level 5

[Author]: **hacnho**

[Author home page]: <http://hvaonline.net>

[Target]: Tuts4You Unpackme: UnPackMe_Themida 1.9.1.0.c, d

[Size]: 1:53 MB (1,613,312 bytes)

[Download Page]: Tuts4You.Com

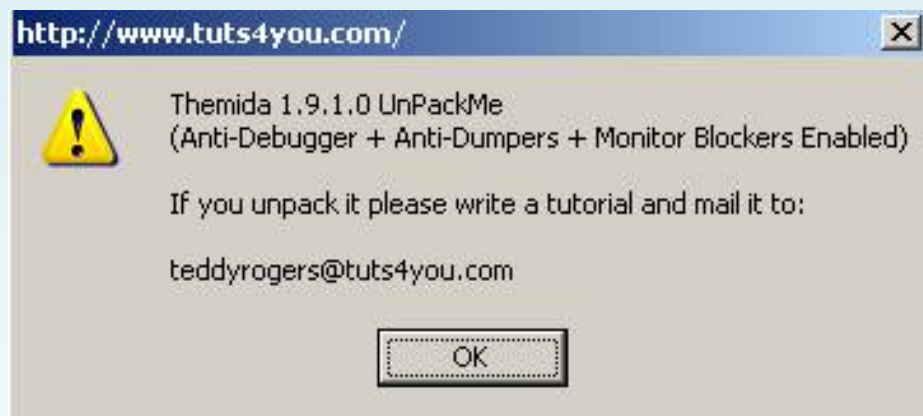
[Packer]: Themida | WinLicense V 1.9.1.0 -> Oreans Technologies

[Compilation language]: Microsoft Visual C + + 6.0

[Tools]: The0DBG + hideToolz **fly** by, LordPE, ImportREC, peid0.94

[OS]: WinXP_SP2

[Greetz]: REA, Exetools, PEDIY, UnpackCN, snd and ARTeam Memberz





_Moi Wedding go on, slightly against, say that the wife actually look delicious thẳng

you too ;-), hix dom tui to see his body too! He said he xin xin nhảm her little children

do not chùi. Now we will study what it is, does the tree leaves garden for 4 tuts and also

just enough to drop. Now we will use this data to the first: D! As you know Tuts4you

are very or unpack me, and it is the place to try unpacker of his. So the usually we will

download 2 unpackme in unpackme Themida 1.9.1.0 of it. To illustrate this tut for

portfolio UnPackMe_Themida 1.9.1.0.c, D. For information about the file's log protect

themida you can contact Ted at snd forum.

I. UnPackMe_Themida 1.9.1.0.c

_Cau Say any hủ: Load OllyDBG to target.

Address	Hex dump	Disassembly	Comment
0046A014	B8 00000000	MOV EAX, 0	
0046A019	60	PUSHAD	
0046A01A	0BC0	OR EAX, EAX	
0046A01C	74 68	JE SHORT UnPackMe.0046A086	
0046A01E	E8 00000000	CALL UnPackMe.0046A023	
0046A023	58	POP EAX	kernel32.7C816D4F
0046A024	05 53000000	ADD EAX, 53	
0046A029	8038 E9	CMP BYTE PTR DS:[EAX], 0E9	
0046A02C	75 13	JNZ SHORT UnPackMe.0046A041	
0046A02E	61	POPAD	
0046A02F	EB 45	JMP SHORT UnPackMe.0046A076	
0046A031	DB2D 37A04600	FLD TBYTE PTR DS:[46A037]	

_De OEP to you as similar tut 4. To record from the tuts 4, man allowed me to do the

steps to Fake OEP. Now you press Alt + M, set a break point in the section on write

code:

003F0000	00002000	UnPackMe		PE header	Dump	
00400000	00001000	UnPackMe		code	Search	Ctrl+B
00401000	000062000	UnPackMe		.rsrc		
00463000	00006000	UnPackMe		data, reso		
00469000	00001000	UnPackMe		.idata		
0046A000	002B8000	UnPackMe		imports		
00730000	00103000		Tuts4You	SFX	Set break-on-access	F2
00840000	00004000				Set memory breakpoint on access	
00850000	000EB000				Set memory breakpoint on write	
00B50000	00003000				Set access	
00B60000	00008000					
00B70000	00001000					
00B80000	00001000					
00B90000	00001000					

Shit _Nhan + F9:

Address	Hex dump	Disassembly	Comment
006F5E00	F3:A4	REP MOVSB BYTE PTR ES:[EDI], BYTE PTR DS	
006F5E02	C685 C1081B07	MOV BYTE PTR SS:[EBP+71B08C1], 56	
006F5E09	68 396D1FD4	PUSH D41F6D39	
006F5E0E	FFB5 35301B07	PUSH DWORD PTR SS:[EBP+71B3035]	
006F5E14	8D85 81012807	LEA EAX, DWORD PTR SS:[EBP+7280181]	
006F5E1A	FFD0	CALL NEAR EAX	

_Nhan F8, and then press Shift + F9 4 times:

Address	Hex dump	Disassembly	Comment
00709617	AB	STOS DWORD PTR ES:[EDI]	
00709618	F5	CMC	
00709619	AD	LODS DWORD PTR DS:[ESI]	
0070961A	F9	STC	
0070961B	C746 FC 8F62D2	MOV DWORD PTR DS:[ESI-4], 3CD2628F	
00709622	50	PUSH EAX	
00709623	B8 8F62D23C	MOV EAX, 3CD2628F	

_Sau The press F8 to here:

Address	Hex dump	Disassembly	Comment
00706393	C785 7D181B07	MOV DWORD PTR SS:[EBP+71B187D], 0	
0070639D	60	PUSHAD	
0070639E	60	PUSHAD	
0070639F	81DB E0CB6B6A	SBB EBX, 6A6BCBE0	
007063A5	B3 E3	MOV BL, 0E3	
007063A7	61	POPAD	
007063A8	61	POPAD	
007063A9	E9 0E000000	JMP UnPackMe.007063BC	
007063AE	103CFB	CALL 007063CF	

_Nhan Ctrl + B type in 3985

Enter binary string to search for

ASCII

UNICODE

HEX +02

☐ Entire block

☐ Case sensitive

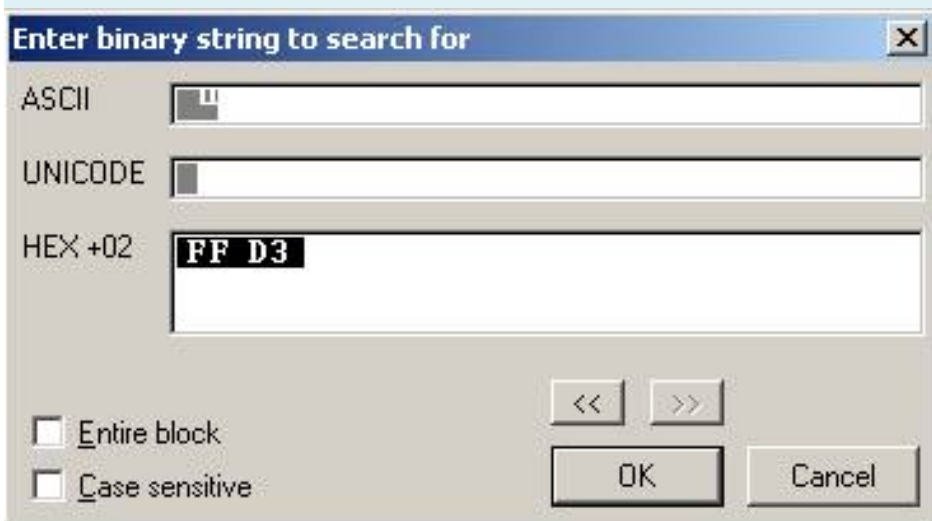
<< >>

OK Cancel

Magic _Toi JUMP, conducted patch it:

Address	Hex dump	Disassembly	Comment
00706679	3985 29121B07	CMP DWORD PTR SS:[EBP+71B1229], EAX	
0070667F	E9 B6000000	JMP UnPackMe.0070673A	Magic Jump
00706684	90	NOP	
00706685	F9	STC	
00706686	0F80 19000000	JO UnPackMe.007066A5	
0070668C	E9 14000000	JMP UnPackMe.007066A5	
00706691	9F	LAHF	

_Ctrl + B: FF D3



_Toi Here:

Address	Hex dump	Disassembly	Comment
00706E0A	83BD 052E1B07	CMP DWORD PTR SS:[EBP+71B2ED5], 1	
00706E11	0F84 00000000	JE UnPackMe.00706EE7	
00706E17	60	PUSHAD	
00706E18	80D9 DD	SBB CL, 0DD	
00706E1B	BA 12F27A5A	MOV EDX, 5A7AF212	
00706E20	61	POPAD	
00706E21	3B8D E1321B07	CMP ECX, DWORD PTR SS:[EBP+71B32E1]	kernel32.7C800000
00706E27	0F84 BA000000	JE UnPackMe.00706EE7	
00706E2D	0F88 0C000000	JS UnPackMe.00706E3F	
00706E33	E9 07000000	JMP UnPackMe.00706E3F	
00706E38	2F	DAS	
00706E39	22BA 94CFC69B	AND BH, BYTE PTR DS:[EDX+9BC6CF94]	
00706E3F	3B8D F10E1B07	CMP ECX, DWORD PTR SS:[EBP+71B0EF1]	USER32.77D40000
00706E45	0F84 9C000000	JE UnPackMe.00706EE7	
00706E4B	0F81 09000000	JNO UnPackMe.00706E5A	
00706E51	60	PUSHAD	
00706E52	0F81 00000000	JNO UnPackMe.00706E58	
00706E58	F9	STC	
00706E59	61	POPAD	
00706E5A	3B8D 75281B07	CMP ECX, DWORD PTR SS:[EBP+71B2875]	ADVAPI32.77DD0000
00706E60	0F84 81000000	JE UnPackMe.00706EE7	
00706E66	F9	STC	
00706E67	8D9D 1CA72F07	LEA EBX, DWORD PTR SS:[EBP+72FA71C]	
00706E6D	60	PUSHAD	
00706E6E	8AEA	MOV CH, DL	
00706E70	66:8BC8	MOV CX, AX	
00706E73	61	POPAD	
00706E74	FFD3	CALL NEAR EBX	UnPackMe.006E7937
00706E76	60	PUSHAD	
00706E77	B8 7EBDA53D	MOV EAX, 3DA5BD7E	
00706E7C	60	PUSHAD	

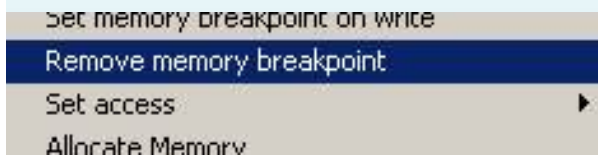
4 _Nop orders jumped JE, if you see only 3 orders jumped immediately enter the

command jump:

Address	Hex dump	Disassembly	Comment
00706DFD	5C	POP ESP	
00706DFE	E9 07000000	JMP UnPackMe.00706E0A	
00706E03	D1F5	SAL EBP, 1	
00706E05	87CD	XCHG EBP, ECX	winmm.PlaySoundA
00706E07	7E 0D	JLE SHORT UnPackMe.00706E16	
00706E09	25 83BDD52E	AND EAX, 2ED5BD83	
00706E0E	1B07	SBB EAX, DWORD PTR DS:[EDI]	
00706E10	010F	ADD DWORD PTR DS:[EDI], ECX	winmm.PlaySoundA
00706E12	84D0	TEST AL, DL	

Nop _Sau when completed to avoid the destroyed IAT, conducted to find OEP. Now

press Alt + M, remove memory breakpoint:



_Ctrl + G: ZwFreeVirtualMemory, press F2 in order retn, press F9 to the register

window is red again;~)

```

Registers (FPU)      <  <
EAX 00000000
ECX 0012FF2C
EDX 7C90EB94 ntdll.KiFastSystemCallRet
EBX 027B0030
ESP 0012FF30
EBP 0012FF4C
ESI 7C90DA48 ntdll.ZwFreeVirtualMemory
EDI 0041C029 UnPackMe.0041C029
EIP 7C90DA54 ntdll.7C90DA54

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_INVALID_PARAMETER (00000057)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty 4.4776609930943070490e-4932
ST1 empty 0.00000000000000000040e-4933
ST2 empty -6.0443058488430905180e-4175
ST3 empty +UNORM 2020 00000017 E29CEBA8
ST4 empty 5.6552512923172035960e-4925
ST5 empty -1.9091779628264741320e-3900
ST6 empty 1.00000000000000000000
ST7 empty 1.00000000000000000000

          3 2 1 0      E S P U O Z D I
FST 4020  Cond 1 0 0 0  Err 0 0 1 0 0 0 0 0 (EQ)
FCW 027F  Prec NEAR,53  Mask  1 1 1 1 1 1

```

_Bay Hours if Unpackme has no copy of the VMS, but how to OEP, khua khua, but

said the gamer VLTK is "to eat by foreign (VinaGau) do". Thằng also Themida, not

information systems, ok , the time you press Alt + M to set a breakpoint on access to or

F2 section code and press F9 to see:

Address	Hex dump	Disassembly	Comment
00CD005F	FF32	PUSH DWORD PTR DS:[EDX]	kernel32.GetVersion
00CD0061	^ E9 94B8F9FF	JMP 00C6C5FA	
00CD0066	BA 0C000000	MOV EDX, 0C	
00CD0068	01F2	ADD EDX, ESI	UnPackMe.006FDFFB
00CD006D	FF32	PUSH DWORD PTR DS:[EDX]	kernel32.GetVersion
00CD006F	^ E9 1DBBF2FF	JMP 00BFC891	
00CD0074	B0 B6	MOV AL, 0B6	
00CD0076	88C3	MOV BL, AL	
00CD0078	58	POP EAX	UnPackMe.00460ADC
00CD0079	80E9 D3	SUB CL, 0D3	
00CD007C	00D9	ADD CL, BL	

What is the _Cai, signs of GetVersion functions, OEP to nearly oi. Now continue to

press F9 again:

Address	Hex dump	Disassembly	Comment
0042710C	33D2	XOR EDX, EDX	
0042710E	8AD4	MOV DL, AH	
004271E0	8915 34E64500	MOV DWORD PTR DS:[45E634], EDX	
004271E6	8BC8	MOV ECX, EAX	
004271E8	81E1 FF000000	AND ECX, 0FF	
004271EE	8900 30E64500	MOV DWORD PTR DS:[45E630], ECX	
004271F4	C1E1 08	SHL ECX, 8	
004271F7	03CA	ADD ECX, EDX	
004271F9	8900 2CE64500	MOV DWORD PTR DS:[45E62C], ECX	
004271FF	C1E8 10	SHR EAX, 10	
00427202	A3 28E64500	MOV DWORD PTR DS:[45E628], EAX	

_Cuon Up OEP will see:

004271A7	80CC 10	OR AH, 10	
004271AA	C3	RETN	
004271AB	90	NOP	
004271AC	90	NOP	
004271AD	90	NOP	
004271AE	90	NOP	
004271AF	90	NOP	
004271B0	115A C6	ADC DWORD PTR DS:[EDX-3A], EBX	OEP
004271B3	B8 615FF9BF	MOV EAX, BFF95F61	
004271B8	6BD0 D7	IMUL EDX, EAX, -29	
004271BB	^ E1 CE	LOOPDE SHORT UnPackMe.0042718B	
004271BD	3A7A 65	CMP BH, BYTE PTR DS:[EDX+65]	
004271C0	AB	STOS DWORD PTR ES:[EDI]	
004271C1	0FA1	POP FS	Modification of segment register
004271C3	6D	INS DWORD PTR ES:[EDI], DX	I/O command
004271C4	43	INC EBX	
004271C5	1F	POP DS	Modification of segment register
004271C6	^ 73 A7	JNB SHORT UnPackMe.0042716F	
004271C8	BF 4933B0D3	MOV EDI, D3B03349	

_Nhung Do it through: P! It has been destroyed, time to restore it, restore the stars, but

remember how to hem Armadillo, only doing so, make any thằng code with VC6 + +, I

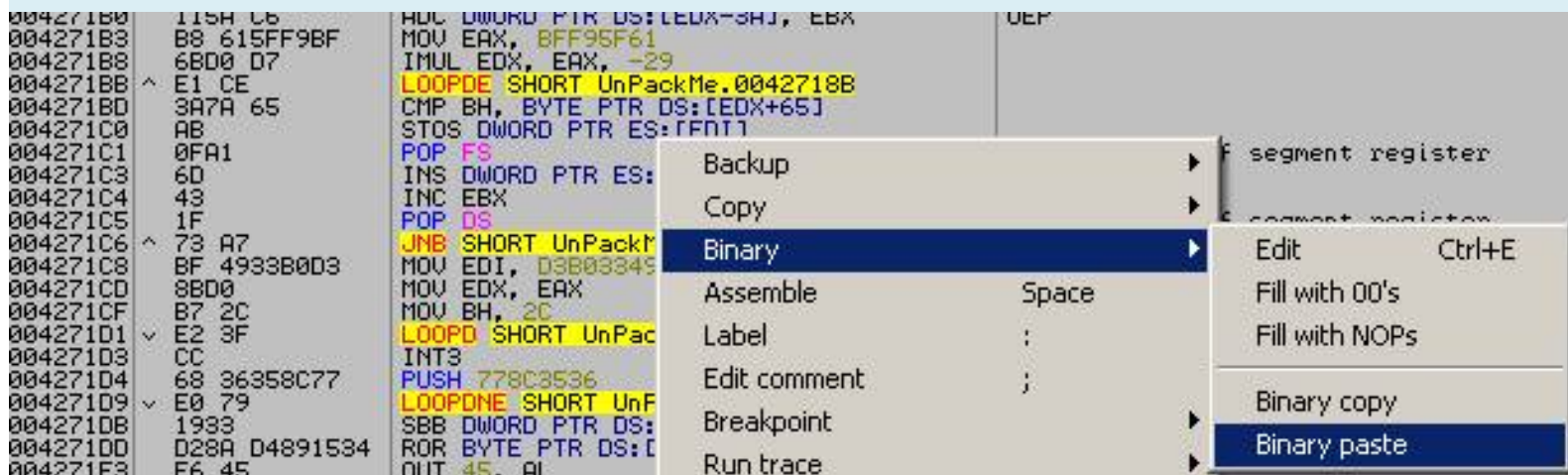
recommend using the APIAddress:

Address	Hex dump	Disassembly	Comment
00402670	55	PUSH EBP	
00402671	8BEC	MOV EBP, ESP	
00402673	6A FF	PUSH -1	
00402675	68 A8364000	PUSH APIAddre.004036A8	
0040267A	68 F6274000	PUSH <JMP.&MSVCRT._except_handler3>	SE handler installation
0040267F	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
00402685	50	PUSH EAX	
00402686	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
0040268D	83EC 68	SUB ESP, 68	
00402690	53	PUSH EBX	
00402691	56	PUSH ESI	
00402692	57	PUSH EDI	
00402693	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	ntdll.7C910738
00402696	33DB	XOR EBX, EBX	

_Copy From 402,670 to 402,694:

Address	Hex dump	Disassembly	Comment
00402670	55	PUSH EBP	
00402671	8BEC	MOV EBP, ESP	
00402673	6A FF	PUSH -1	
00402675	68 A8364000	PUSH APIAddre.004036A8	
0040267A	68 F6274000	PUSH <JMP.&MSVCRT._except_handler3>	SE handler installation
0040267F	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
00402685	50	PUSH EAX	
00402686	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
0040268D	83EC 68	SUB ESP, 68	
00402690	53	PUSH EBX	
00402691	56	PUSH ESI	
00402692	57	PUSH EDI	
00402693	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	ntdll.7C910738
00402696	33DB	XOR EBX, EBX	

_Paste Invisible:



_Ra This here:

Address	Hex dump	Disassembly	Comment
004271B0	55	PUSH EBP	OEP
004271B1	8BEC	MOV EBP, ESP	
004271B3	6A FF	PUSH -1	
004271B5	68 A8364000	PUSH UnPackMe.004036A8	
004271B8	68 F6274000	PUSH UnPackMe.004027F6	
004271BF	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	ASCII "f,"
004271C5	50	PUSH EAX	
004271C6	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
004271CD	83EC 68	SUB ESP, 68	
004271D0	53	PUSH EBX	
004271D1	56	PUSH ESI	
004271D2	57	PUSH EDI	
004271D3	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
004271D6	35 8C77E079	XOR EAX, 79E0778C	
004271DB	1933	SBB DWORD PTR DS:[EBX], ESI	
004271DD	D28A D4891534	ROR BYTE PTR DS:[EDI], CL	

_Con Memory function GetVersion to this? Now the patch 4271D6 GetVersion to Call:

Address	Hex dump	Disassembly	Comment
004271B0	55	PUSH EBP	OEP
004271B1	8BEC	MOV EBP, ESP	
004271B3	6A FF	PUSH -1	
004271B5	68 A8364000	PUSH UnPackMe.004036A8	
004271B8	68 F6274000	PUSH UnPackMe.004027F6	
004271BF	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
004271C5	50	PUSH EAX	
004271C6	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
004271CD	83EC 68	SUB ESP, 68	
004271D0	53	PUSH EBX	
004271D1	56	PUSH ESI	
004271D2	57	PUSH EDI	
004271D3	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
004271D6	E8 D0A23E7C	CALL kernel32.GetVersion	
004271DB	90	NOP	
004271DC	33D2	XOR EDX, EDX	
004271DE	8AD4	MOV DL, AH	

_Dump And FixIAT not be, not where we need to edit 2 push function. Now you look

down the stack of books:

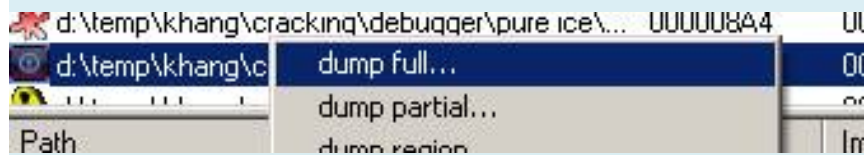
Address	Value	Comment
0012FF48	65B78399	
0012FF4C	005C6EB8	RETURN to UnPackMe.005C6EB8 from UnPackMe.005C6EC7
0012FF50	00CE0000	
0012FF54	006FE16E	UnPackMe.006FE16E
0012FF58	F940F052	
0012FF5C	0012FF70	
0012FF60	0012FF9C	
0012FF64	0000A042	
0012FF68	7C90EB94	ntdll.KiFastSystemCallRet
0012FF6C	006FA40A	UnPackMe.006FA40A
0012FF70	FFF03D64	
0012FF74	65B78399	
0012FF78	00699F5A	UnPackMe.00699F5A
0012FF7C	0012FF20	
0012FF80	0012FF9C	
0012FF84	0012FFE0	Pointer to next SEH record
0012FF88	004292C8	SE handler
0012FF8C	00450E60	UnPackMe.00450E60
0012FF90	FFFFFFFF	
0012FF94	00699F5A	UnPackMe.00699F5A

_Dien Window to stop the CPU:

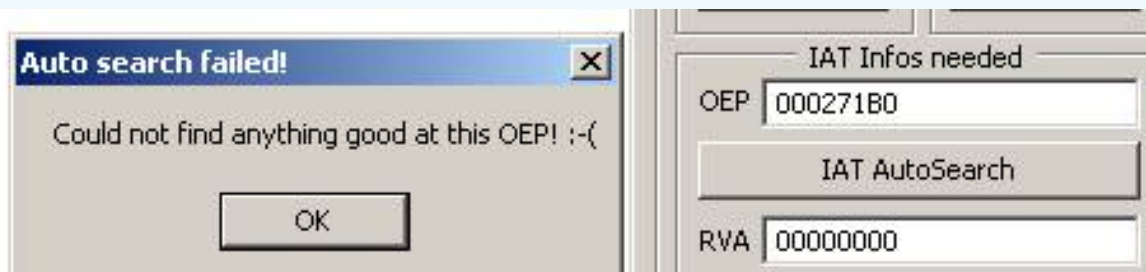
Address	Hex dump	Disassembly	Comment
004271B0	55	PUSH EBP	OEP
004271B1	8BEC	MOV EBP, ESP	
004271B3	6A FF	PUSH -1	
004271B5	68 600E4500	PUSH UnPackMe.00450E60	
004271B9	68 C8924200	PUSH UnPackMe.004292C8	
004271BF	64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
004271C5	50	PUSH EAX	
004271C6	64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
004271CD	83EC 68	SUB ESP, 68	
004271D0	53	PUSH EBX	
004271D1	56	PUSH ESI	
004271D2	57	PUSH EDI	
004271D3	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
004271D6	E8 D0A23E7C	CALL kernel32.GetVersion	
004271D8	90	NOP	
004271DC	33D2	XOR EDX, EDX	

_Qua Standard time Ctrl + * to set new origin at the OEP. Open LordPE full dump, it is

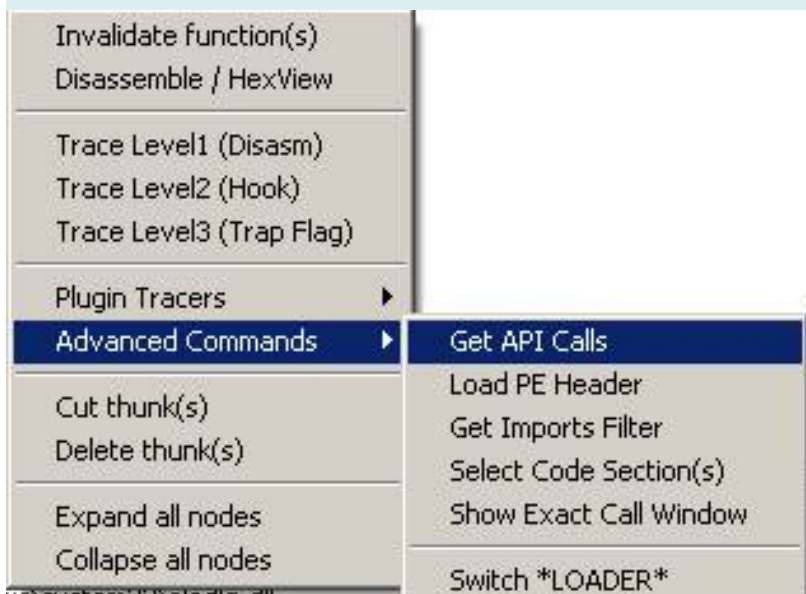
this anti dumper should do tuts nhe 2: D!



_Mo Imprec, fill OEP and receive: D!



_Khong Stars, must click in the Function of Imported Imprec selected:



_Nhan OK and as follows:

Imported Functions Found

```

+ ? FThunk:000589D8 NbFunc:1 (decimal:1) valid:NO
+ ? FThunk:00058A60 NbFunc:1 (decimal:1) valid:NO
+ ? FThunk:00058DE0 NbFunc:1 (decimal:1) valid:NO
+ ? FThunk:00059114 NbFunc:1 (decimal:1) valid:NO
+ ? FThunk:00059134 NbFunc:1 (decimal:1) valid:NO
+ ? FThunk:00059144 NbFunc:1 (decimal:1) valid:NO
+ ? FThunk:00059154 NbFunc:1 (decimal:1) valid:NO
+ ? FThunk:000591D0 NbFunc:1 (decimal:1) valid:NO
+ ? FThunk:00059548 NbFunc:4 (decimal:4) valid:NO
+ ? FThunk:00059568 NbFunc:2 (decimal:2) valid:NO
+ ? FThunk:0005B588 NbFunc:1 (decimal:1) valid:NO
+ ? FThunk:0005B5B8 NbFunc:2 (decimal:2) valid:NO
+ ? FThunk:0005B788 NbFunc:2 (decimal:2) valid:NO
+ ? FThunk:0005BB30 NbFunc:1 (decimal:1) valid:NO
+ ? FThunk:0005C418 NbFunc:1 (decimal:1) valid:NO
+ advapi32.dll FThunk:00060818 NbFunc:1 (decimal:1) valid:YES
+ advapi32.dll FThunk:00060820 NbFunc:1 (decimal:1) valid:YES

```

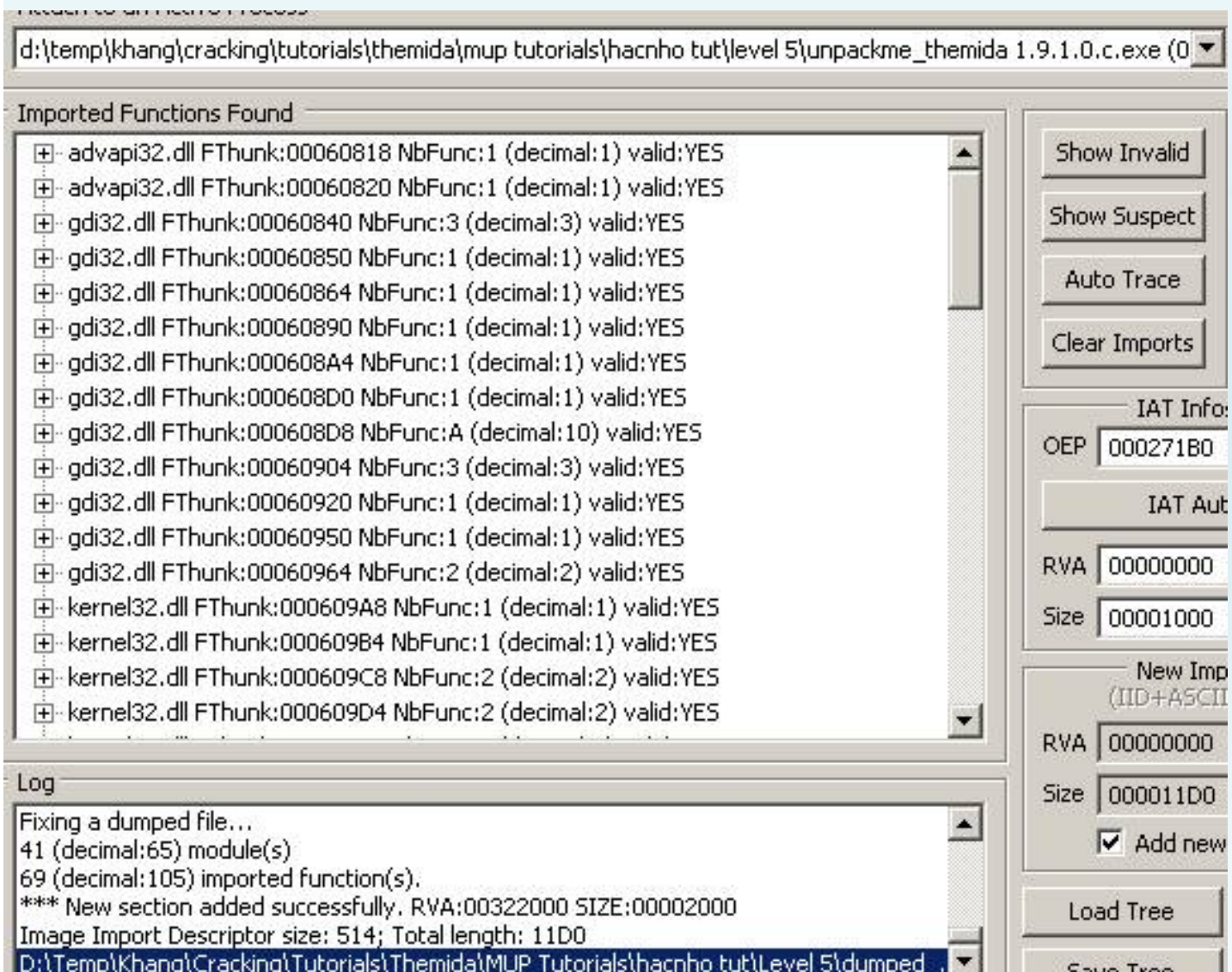
Log

Image Base:00400000 Size:00322000

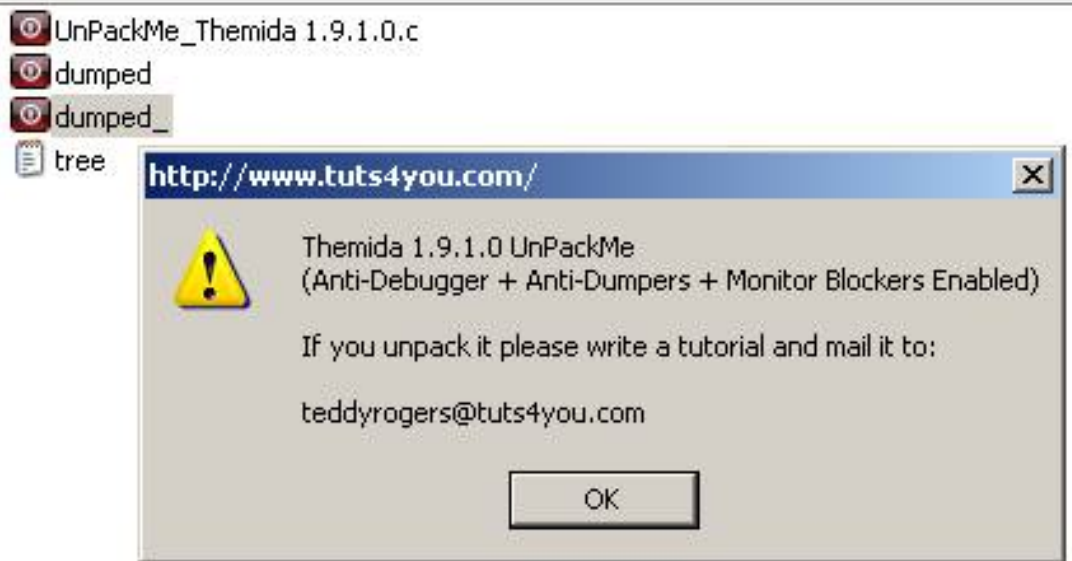
Current imports:

41 (decimal:65) valid module(s) (added: +41 (decimal:+65))
 7F (decimal:127) imported function(s). (added: +7F (decimal:+127))
 (16 (decimal:22) unresolved pointer(s)) (added: +16 (decimal:+22))

[_Show Invalid, Cut thunk:](#)



_Tien Fix the dump, and a test:



II.UnPackMe_Themida 1.9.1.0.d

In fact this month as italy chang **UnPackMe_Themida 1.9.1.0.c** but instead nop 4 JE

command, you only need to submit 3 commands:

Address	Hex dump	Disassembly
0069A065	66:8BCB	MOV CX, BX
0069A068	61	POPAD
0069A069	83BD 59179409	CMP DWORD PTR SS:[EBP+9941759], 1
0069A070	90	NOP
0069A071	90	NOP
0069A072	90	NOP
0069A073	90	NOP
0069A074	90	NOP
0069A075	90	NOP
0069A076	60	PUSHAD
0069A077	60	PUSHAD
0069A078	F5	CMC
0069A079	0FBFD2	MOVSX EDX, DX
0069A07C	61	POPAD
0069A07D	E8 07000000	CALL UnPackMe.0069A089
0069A082	AA	STOS BYTE PTR ES:[EDI]
0069A083	AB	STOS DWORD PTR ES:[EDI]
0069A084	7A 8C	JPE SHORT UnPackMe.0069A012
0069A086	B2 11	MOV DL, 11
0069A088	32F9	XOR BH, CL
0069A08A	5F	POP EDI
0069A08B	61	POPAD
0069A08C	3B8D 85229409	CMP ECX, DWORD PTR SS:[EBP+9942285]
0069A092	90	NOP
0069A093	90	NOP
0069A094	90	NOP
0069A095	90	NOP
0069A096	90	NOP
0069A097	90	NOP
0069A098	60	PUSHAD
0069A099	66:BA 52A2	MOV DX, 0A252
0069A09D	61	POPAD
0069A09E	3B8D 8D1F9409	CMP ECX, DWORD PTR SS:[EBP+9941F8D]
0069A0A4	90	NOP
0069A0A5	90	NOP
0069A0A6	90	NOP
0069A0A7	90	NOP
0069A0A8	90	NOP
0069A0A9	90	NOP
0069A0AA	✓ E9 05000000	JMP UnPackMe.0069A0B4
0069A0AF	41	INC ECX
0069A0B0	43	INC EBX
0069A0B1	93	XCHG EAX, EBX
0069A0B2	B4 AA	MOV AH, 0AA
0069A0B4	✓ E9 06000000	JMP UnPackMe.0069A0BF
0069A0B9	96	XCHG EAX, ESI

A child _Xong again: P! See the tut 6.

PS: cover tut tut from the video of what. Thank you!

-- Perhaps all that they do in the body in the gift bags to name hem aged and

aged Zom, watch aged sad sad del nick bags. Dear Please themida bags, with a

Newbie, also aged dragon with his South Com tề list tomb Dung - North Kieu

Phong, a specialist of aged. Themida NET, a specialized therapy aged 32 bit

Themida. I have not cry is the father was then on, I do dare to write a gift. And to

learn much more aged RCA. Hix hix. But aged Zom, some meals I guide you to

the meat thẳng 4VNMod Audition, not sure I have the whole network with them

aged not more, but say what writing gift. lol!

Any suggestions, comments or corrections please PM me via: hacnho@hotmail.com

25-10-2007

Themida - then will go through

Winlicense 1.9.xx - Unpackme level 6 Expansion 1

Tutorial # 6 exp 1 @ 2007 by hacnho (hacnho@hotmail.com)

(Write the gift of friendship REA - all copy must be agreed by the author)

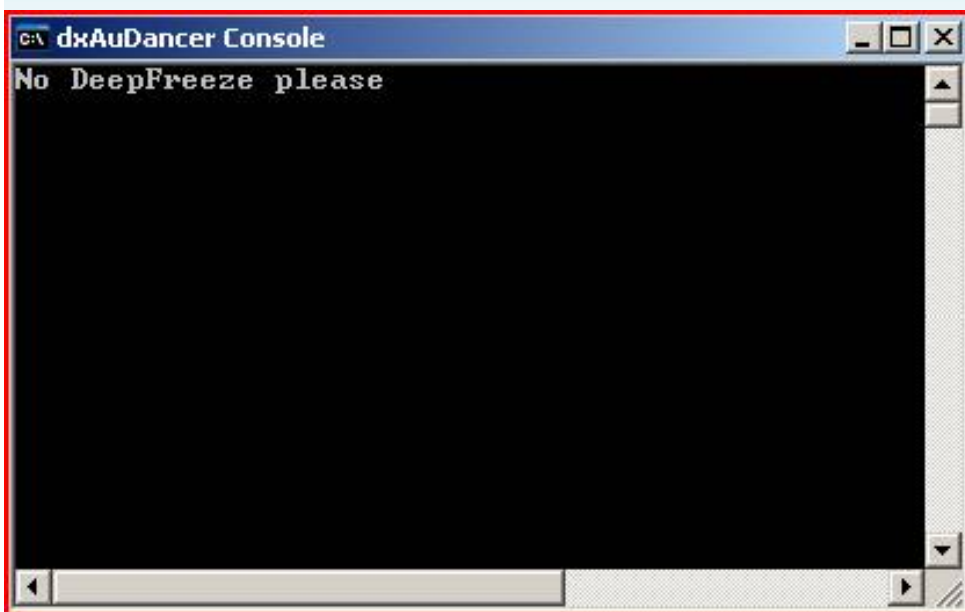
[Article title]: Winlicense 1.9.xx - Unpackme level 6 exp 1
[Author]: **hacnho**
[Author home page]: <http://hvaonline.net>
[Target]: Unknow target: P!
[Size]: 1.88 MB (1,978,368 bytes)
[Download Page]: Google
[Packer]: Themida | V WinLicense 1.9.xx CB-> Oreans Technologies
[Compilation language]: Borland Delphi 7
[Tools]: The0DBG + hideToolz **fly** by, LordPE, ImportREC, peid0.94
[OS]: WinXP_SP2
[Greetz]: REA, Exetools, PEDIY, UnpackCN, snd and ARTeam Memberz

_Hi Her children, we continue the expansion of the tutorial # 6. In this section we will resolve the Delphi + Winlicense. Thank you trojanvietnam target was send this to me. Actually, when I received this target also chả very excited, because a busy story, with it have not written for the console and as such have the icon, slightly doubt be put to, he he. Today it's cold will be appropriate to sit meat it: P. Ok, here we go unpack.

_Dau First check with PeiD:



_Chay Try:



_Hoi Difficult to understand, we have set new DeepFreeze not do it for me, sure the players do all the things, that only the army, just to unpack, tí charge, running in Vmware for sure eat. Now OllyDBG sure that you have to pass by Themida / Winlicense 1.9.xx We load in and Olly in here:

Address	Hex dump	Disassembly	Comment
00416014	B3 00000000	MOV EAX, 0	
00416019	60	PUSHAD	
0041601A	0BC0	OR EAX, EAX	
0041601C	74 68	JE SHORT unpackme.00416086	
0041601E	E8 00000000	CALL unpackme.00416023	
00416023	58	POP EAX	kernel32.7C816D4F
00416024	05 53000000	ADD EAX, 53	
00416029	8038 E9	CMP BYTE PTR DS:[EAX], 0E9	
0041602C	75 13	JNZ SHORT unpackme.00416041	
0041602F	61	POPAD	

_Lam How to determine what do with this pack, simply stop, view:



_Bay Time you press Alt + M, set a breakpoint on Hardware write, the press Shift + F9 will come:

Address	Hex dump	Disassembly	Comment
00735179	8918	MOV DWORD PTR DS:[EAX], EBX	
0073517B	8BC9	MOV ECX, ECX	
0073517D	EB 0C	JMP SHORT unpackme.0073518B	
0073517F	0000	ADD BYTE PTR DS:[EAX], AL	
00735181	0000	ADD BYTE PTR DS:[EAX], AL	
00735183	0100	ADD DWORD PTR DS:[EAX], EAX	unpackme.0040C6F8
00735185	0000	ADD BYTE PTR DS:[EAX], AL	
00735187	00	ADD DWORD PTR DS:[EAX], EAX	

_Kha Is not? No problem, continue Shift + F9:

Address	Hex dump	Disassembly	Comment
0074ABB1	8903	MOV DWORD PTR DS:[EBX], EAX	
0074ABB1	51	PUSH ECX	
0074ABB2	52	PUSH EDX	unpackme.00401000
0074ABB3	8D85 74420110	LEA EAX, DWORD PTR SS:[EBP+10014274]	
0074ABB9	FFD0	CALL NEAR EAX	
0074ABBB	8BF8	MOV EDI, EDX	unpackme.00401000
0074ABBD	8BF1	MOV ESI, ECX	
0074ABBE	8BD1	MOV EDI, ECX	

_Bay Hour press F7, F8, and then press Shift + F9 2 times:

Address	Hex dump	Disassembly	Comment
0075ABF9	8908	MOV DWORD PTR DS:[ESI], ECX	
0075ABFB	AD	LODS DWORD PTR DS:[ESI]	
0075ABFC	C746 FC 000000	MOV DWORD PTR DS:[ESI-4], 0	
0075AC03	89B5 39160110	MOV DWORD PTR SS:[EBP+10011639], ESI	unpackme.0074EC82
0075AC09	83F8 FF	CMP EAX, -1	
0075AC0C	7505 20000000	JNZ unpackme.0075AC32	
0075AC12	813E 00000000	CMP DWORD PTR DS:[ESI], 00000000	
0075AC18	7505 14000000	JNZ unpackme.0075AC32	
0075AC1E	C746 00000000	MOV DWORD PTR DS:[ESI], 0	

_Tiep To press F8 to trace here:

Address	Hex dump	Disassembly	Comment
0075A306	C785 05310110	MOV DWORD PTR SS:[EBP+10013105], 0	
0075A310	C785 E1280110	MOV DWORD PTR SS:[EBP+100128E1], 0	
0075A31A	83BD 2B8C1710	CMP DWORD PTR SS:[EBP+10178C2B], 0	
0075A321	7505 08000000	JE unpackme.0075A32F	
0075A327	8B00 F2030510	MOV EAX, DWORD PTR SS:[EBP+100503F5]	

_Gio If you have to work more hours. Take the way of tut then also, or use one's home SubZero. You look down slightly:

0075A37C	8BC1	MOV EAX, ECX	
0075A37E	F7D0	NOT EAX	
0075A380	3985 01090110	CMP DWORD PTR SS:[EBP+10010901], EAX	unpackme.00410104
0075A386	7505 17000000	JE unpackme.0075A3A3	unpackme.00410104
0075A38C	83BD 41130110	CMP DWORD PTR SS:[EBP+10011341], 0	
0075A393	7505 0A000000	JNE unpackme.0075A3A3	Patch tha'nh JMP SHORT 0075A3A3
0075A399	C785 31090110	MOV DWORD PTR SS:[EBP+10010931], 1	
0075A3A3	61	POPAD	
0075A3A4	B9 9C024335	MOV ECX, 3543029C	
0075A3A9	BA 37BA5D24	MOV EDI, 245DBA37	
0075A3AE	AD	LODS DWORD PTR DS:[ESI]	
0075A3AF	89B5 39160110	MOV DWORD PTR SS:[EBP+10011639], ESI	unpackme.0074EC82
0075A3B5	C746 FC 000000	MOV DWORD PTR DS:[ESI-4], 0	

_Chung Will become pach 0075A393 JMP 0075A3A3 (opcode: EB0E0A).

Now press Ctrl + B: type in:

0F 84 ? ? ? ? 3B 8D ? ? ? ? 0F 84 ? ? ? ? 3B 8D ? ? ? ? 0F 84 ? ? ? ? 3B 8D ? ? ? ?

Or FF D3:

Enter binary string to search for

ASCII

* ? ? ? ? ; \ ? ? ? ? * ? ? ? ? ; \ ? ? ? ?

UNICODE

?? ?? ?? ??

HEX +24

0F 84 ? ? ? ? ? ? ? ? 3B 8D ? ? ? ? ? ? ? ?
0F 84 ? ? ? ? ? ? ? ? 3B 8D ? ? ? ? ? ? ? ?

☐ Entire block
 ☐ Case sensitive

<<

>>

OK

Cancel

_Muc Purpose for us to make order 4 JE check Image API Base. Will come:

Address	Hex dump	Disassembly	Comment
0075A4E4	0F84 39000000	JE unpackme.0075A523	
0075A4EA	3B8D A10C0110	CMP ECX, DWORD PTR SS:[EBP+10010CA1]	kernel32.7C800000
0075A4F0	0F84 20000000	JE unpackme.0075A523	
0075A4F6	3B8D 55090110	CMP ECX, DWORD PTR SS:[EBP+10010955]	USER32.77D40000
0075A4FC	0F84 21000000	JE unpackme.0075A523	
0075A502	3B8D 69250110	CMP ECX, DWORD PTR SS:[EBP+10012569]	ADVAPI32.77D00000
0075A508	0F84 15000000	JE unpackme.0075A523	

_Nhin Down:

Information
SS:[005F1875]=7C800000 (kernel32.7C800000)
ECX=7C9010ED (ntdll.RtlLeaveCriticalSection)

_Chung Will patch to address 00754E4 JMP SHORT 0075A50E:

Address	Hex dump	Disassembly	Comment
0075A4E4	0F84 39000000	JE unpackme.0075A523	Patch tha'nh JMP SHORT 0075A50E
0075A4EA	3B8D A10C0110	CMP ECX, DWORD PTR SS:[EBP+10010CA1]	kernel32.7C800000
0075A4F0	0F84 20000000	JE unpackme.0075A523	
0075A4F6	3B8D 55090110	CMP ECX, DWORD PTR SS:[EBP+10010955]	USER32.77D40000
0075A4FC	0F84 21000000	JE unpackme.0075A523	
0075A502	3B8D 69250110	CMP ECX, DWORD PTR SS:[EBP+10012569]	ADVAPI32.77D00000

_Den Ago we also made similar tut 6, Alloc a cave code to write code to:

```
A3 00 01 93 02 89 08 AD C7 46 FC 00 00 00 00 E9 EF AB E2 FD 50 A1 00 01 93 02 89
07 80 7F FF E875 08 66 C7 47 FE FF 15 EB 06 66 C7 47 FE FF 25 58 E9 A3 AC E2 FD
50 A1 00 01 93 02 89 47 01 80 7F FF E8 75 08 66 C7 47 FF FF 15 EB 06 66 C7 47 FF FF
25 58 0F 85 7F AC E2 FD E9 62 AC E2 FD 83 C7 04 E9 9C AB E2 FD 90
```

Address	Hex dump	Disassembly	Comment
02930000	A3 00019302	MOV DWORD PTR DS:[2930100], EAX	unpackme.00410104
02930005	8908	MOV DWORD PTR DS:[EAX], ECX	
02930007	AD	LODS DWORD PTR DS:[ESI]	
02930008	C746 FC 000000	MOV DWORD PTR DS:[ESI-4], 0	
0293000F	E9 EFABE2FD	JMP unpackme.0075AC03	
02930014	50	PUSH EAX	unpackme.00410104
02930015	A1 00019302	MOV EAX, DWORD PTR DS:[2930100]	
0293001A	8907	MOV DWORD PTR DS:[EDI], EAX	unpackme.00410104
0293001C	807F FF E8	CMP BYTE PTR DS:[EDI-1], 0E8	
02930020	75 08	JNZ SHORT 0293002A	
02930022	66:C747 FE FF15	MOV WORD PTR DS:[EDI-2], 15FF	
02930028	EB 06	JMP SHORT 02930030	
0293002A	66:C747 FE FF25	MOV WORD PTR DS:[EDI-2], 25FF	
02930030	58	POP EAX	unpackme.00410104
02930031	E9 A3ACE2FD	JMP unpackme.0075ACD9	
02930036	50	PUSH EAX	unpackme.00410104
02930037	A1 00019302	MOV EAX, DWORD PTR DS:[2930100]	
0293003C	8947 01	MOV DWORD PTR DS:[EDI+1], EAX	unpackme.00410104
0293003F	807F FF E8	CMP BYTE PTR DS:[EDI-1], 0E8	
02930043	75 08	JNZ SHORT 0293004D	
02930045	66:C747 FF FF15	MOV WORD PTR DS:[EDI-1], 15FF	
0293004B	EB 06	JMP SHORT 02930053	
0293004D	66:C747 FF FF25	MOV WORD PTR DS:[EDI-1], 25FF	
02930053	58	POP EAX	unpackme.00410104
02930054	0F85 7FACE2FD	JNZ unpackme.0075ACD9	
0293005A	E9 62ACE2FD	JMP unpackme.0075ACC1	
0293005F	83C7 04	ADD EDI, 4	
02930062	E9 9CABE2FD	JMP unpackme.0075AC03	
02930067	90	NOP	
02930068	0000	ADD BYTE PTR DS:[EAX], AL	
02930069	0000	ADD BYTE PTR DS:[EAX], AL	

_Neu Like you can do the type of SubZero also. I will train and then the familiar, time adjusting to the jump command:

a.

Address	Hex dump	Disassembly	Comment
0075ABF9	8908	MOV DWORD PTR DS:[EDI], ECX	Patch tha'nh JMP 2930000
0075ABFB	AD	LODS DWORD PTR DS:[ESI]	
0075ABFC	C746 FC 000000	MOV DWORD PTR DS:[ESI-4], 0	

b.

0075ACAF	AA	STOS BYTE PTR ES:[EDI]	
0075ACB0	E9 24000000	JMP unpackme.0075ACD9	Patch tha'nh JMP 2930014
0075ACB5	58	POP EAX	unpackme.00410104
0075ACB6	AA	STOS BYTE PTR ES:[EDI]	
0075ACB7	807F FF E9	CMP BYTE PTR DS:[EDI-1], 0F9	

c.

0075ACB7	807F FF E9	CMP BYTE PTR DS:[EDI-1], 0E9	
0075ACB8	0F85 18000000	JNZ unpackme.0075ACD9	Patch tha'nh JMP 2930036
0075ACC1	03BD 2B8C1710	CMP DWORD PTR SS:[EBP+10178C2B], 0	
0075ACC8	0F84 08000000	JE unpackme.0075ACD6	
0075ACCE	809D 9A8F0E10	LEA EBX, DWORD PTR SS:[EBP+100E8F9A]	

d.

0075AC9D	E9 43000000	JMP unpackme.0075ACE5	
0075ACA2	83F8 50	CMPEAX, 50	Hop
0075ACA5	0F82 0A000000	JB unpackme.0075ACB5	
0075ACAA	A0 90	MOV AL, 90	

e.

0075ACD4	FFD3	CALL NEAR EBX	unpackme.006C932
0075ACD6	8347 04	CMPEAX, 4	NOP 3 bytes
0075ACD9	8B95 B9230110	MOV ECX, DWORD PTR SS:[EBP+100123B91]	

f.

0075ACE1	83E8 04	SUB EAX, 4	
0075ACE4	AD	STOS DWORD PTR ES:[EDI]	NOP 1 byte
0075ACE5	AD	LODS DWORD PTR DS:[ESI]	

_Tim OEP: Ctrl + B type E9 A4 06 00 00

Enter binary string to search for

ASCII

UNICODE

HEX +05

☒ Entire block

☐ Case sensitive

_Toi Here:

Address	Hex dump	Disassembly	Co
0075AD27	E9 A4060000	JMP unpackme.0075B3D0	
0075AD2C	60	PUSHAD	
0075AD2D	8B8D 612D0110	MOV ECX, DWORD PTR SS:[EBP+10012D61]	
0075AD33	8B09	MOV ECX, DWORD PTR DS:[ECX]	

Hardware _Dat a breakpoint on Exccution:

Address	Hex dump	Disassembly	Comment
0075AD27	E9 A4060000	JMP	
0075AD2C	60	Backup	
0075AD2D	8B80 612D0110	MOV	11
0075AD33	8B09	Copy	
0075AD35	8980 278C1710	MOV	CX
0075AD3B	8138 4E54444C	Binary	
0075AD41	0F85 1C000000	JMP	
0075AD47	66:8178 04 4C2	Assemble Space	
0075AD4D	0F85 10000000	JMP	
0075AD53	8BF0	Label	:
0075AD55	83C6 06	MOV	
0075AD58	8B85 552A0110	MOV	
0075AD5E	E9 06010000	JMP	
0075AD63	8BF0	Breakpoint	Toggle F2
0075AD65	8BD0	Run trace	Conditional Shift+F2
0075AD67	33C9		Conditional log Shift+F4
0075AD69	AC		
0075AD6A	3C 00	Follow Enter	
0075AD6C	0F84 5D000000	JMP	
0075AD72	3C 2D	Go to	Memory, on access
0075AD74	0F84 EFFFFFFF	JMP	
0075AD7A	3C 2E	Thread	Memory, on write
0075AD7C	0F84 3D000000	JMP	
0075AD82	3C 30	Follow in Dump	
0075AD84	0F82 00010000	JMP	Hardware, on execution

_Nhan F9, remove breakpoint, Alt + M and set the breakpoint code section:

00400000	00001000	unpackme	PE header	Inag	R	RWE
00401000	000013000	unpackme	code	base	B	base
00414000	00001000	unpackme	.rs			
00415000	00001000	unpackme	.ic			
00416000	00417000	unpackme	dxs			
00830000	00007000					
008F0000	00002000					
00900000	00103000					
00A10000	00132000					
00D10000	000F1000					
00E10000	0008D000					
00EA0000	00097000					
00F40000	00001000					
010BB000	00001000					
010BC000	00004000					
011BB000	00001000					
011BC000	00004000					
012BB000	00001000					
013BC000	00004000					

_Nhan F9:

Address	Hex dump	Disassembly	Comment
005F9EDE	FF32	PUSH DWORD PTR DS:[EAX]	unpackme.0040C034
005F9EE0	E9 4CEBFFFF	JMP unpackme.005F8A31	
005F9EE5	81EA 8805245D	SUB EDX, 5D240588	
005F9EEB	E9 F93B0000	JMP unpackme.005FDAE9	
005F9EF0	59	POP ECX	
005F9EF1	68 B1270000	PUSH 27B1	
005F9EF2	00103000	MOV DWORD PTR DS:[EAX], EDI	

OEP _Chua know or not, we find String Reference see why:

Follow in Dump	▶	
Search for	▶	Name (label) in current module Ctrl+N
Find references to	▶	Name in all modules
View	▶	
Copy to executable	▶	Command Ctrl+F
Analysis	▶	Sequence of commands Ctrl+S
	▶	Constant
Detach Process	▶	Binary string Ctrl+B
	▶	Next Ctrl+L
Process Patcher		
Bookmark	▶	All intermodular calls
Dump debugged process		All commands
		All sequences
Run Script	▶	All constants
Script Functions...	▶	All switches
Appearance	▶	All referenced text strings

Code With Delphi, but signs of Dephi are:

Address	Hex dump	Disassembly	Comment
0045EC7C	55	PUSH EBP	
0045EC7D	8BEC	MOV EBP, ESP	
0045EC7F	83C4 F0	ADD ESP, -10	
0045EC82	B8 6CEA4500	MOV EAX, AutoShut.0045EA6C	
0045EC87	E8 A86FAFF	CALL AutoShut.00405C34	
0045EC8C	A1 48004600	MOV EAX, DWORD PTR DS:[460048]	
0045EC91	8B00	MOV EAX, DWORD PTR DS:[EAX]	
0045EC93	E8 A050FFFF	CALL AutoShut.00454A38	
0045EC98	8B00 30014600	MOV ECX, DWORD PTR DS:[460130]	AutoShut.00461BE8
0045EC9E	A1 48004600	MOV EAX, DWORD PTR DS:[460048]	
0045ECA3	8B00	MOV EAX, DWORD PTR DS:[EAX]	
0045ECA5	8B15 58D94500	MOV EDX, DWORD PTR DS:[45D958]	AutoShut.0045D9A4
0045ECAB	E8 A050FFFF	CALL AutoShut.00454A50	
0045ECB0	8B00 68014600	MOV ECX, DWORD PTR DS:[460168]	AutoShut.00461BF0
0045ECB6	A1 48004600	MOV EAX, DWORD PTR DS:[460048]	
0045ECB8	8B00	MOV EAX, DWORD PTR DS:[EAX]	
0045ECBD	8B15 D4E74500	MOV EDX, DWORD PTR DS:[45E7D4]	AutoShut.0045E820
0045ECC3	E8 8850FFFF	CALL AutoShut.00454A50	
0045ECC8	8B00 94FE4500	MOV ECX, DWORD PTR DS:[45FE94]	AutoShut.00461BE0
0045ECCD	A1 48004600	MOV EAX, DWORD PTR DS:[460048]	
0045ECD3	8B00	MOV EAX, DWORD PTR DS:[EAX]	
0045ECD5	8B15 D4D54500	MOV EDX, DWORD PTR DS:[45D5D4]	AutoShut.0045D620
0045ECD8	E8 7050FFFF	CALL AutoShut.00454A50	
0045ECE0	A1 48004600	MOV EAX, DWORD PTR DS:[460048]	
0045ECE5	8B00	MOV EAX, DWORD PTR DS:[EAX]	
0045ECE7	E8 E450FFFF	CALL AutoShut.00454A00	
0045ECEC	E8 6B50FAFF	CALL AutoShut.00403D5C	
0045ECF1	8D40 00	LEA EAX, DWORD PTR DS:[EAX]	
0045ECF4	0000	ADD BYTE PTR DS:[EAX], AL	

Do So we press F9 2 times more:

Address	Hex dump	Disassembly	Comment
00404418	53	PUSH EBX	
00404419	8BD8	MOV EBX, EAX	unpackme.0040A2EC
0040441B	33C0	XOR EAX, EAX	unpackme.0040A2EC
0040441D	A3 F8C64000	MOV DWORD PTR DS:[40C6F8], EAX	unpackme.0040A2EC
00404422	6A 00	PUSH 0	
00404424	E8 2BFFFFFF	CALL unpackme.00404354	JMP to kernel32.GetModuleHandleA
00404429	A3 00C74000	MOV DWORD PTR DS:[40C700], EAX	unpackme.0040A2EC
0040442E	A1 00C74000	MOV EAX, DWORD PTR DS:[40C700]	
00404433	A3 90B04000	MOV DWORD PTR DS:[40B090], EAX	unpackme.0040A2EC
00404438	33C0	XOR EAX, EAX	unpackme.0040A2EC
0040443A	A3 94B04000	MOV DWORD PTR DS:[40B094], EAX	unpackme.0040A2EC

_He He refers GetModuleHandleA, upcoming OEP then, press F8 trace hours from:

Address	Hex dump	Disassembly	Comment
005F89EA	60	PUSHADD	
005F89EB	9C	PUSHAD	
005F89EC	FC	CLD	
005F89ED	E8 00000000	CALL unpackme.005F89F2	
005F89F2	5F	POP EDI	
005F89F3	81EF 1E7E0110	SUB EDI, 10017E1E	
005F89F9	8BC7	MOV EAX, EDI	

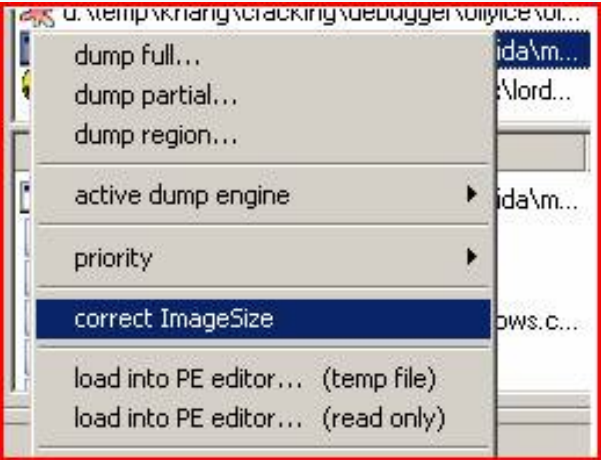
_Nho For this to you, a dump VMS we will go through it here: P! Now press F9 2 times nhé:

Address	Hex dump	Disassembly	Comment
004091D4	55	PUSH EBP	
004091D5	8BEC	MOV EBP, ESP	
004091D7	81C4 98FBFFFF	ADD ESP, -468	
004091D9	53	PUSH EBX	
004091DE	33D2	XOR EDX, EDX	
004091E0	8995 98FBFFFF	MOV DWORD PTR SS:[EBP-468], EDX	
004091E6	8995 04FCFFFF	MOV DWORD PTR SS:[EBP-3FC], EDX	
004091EC	8995 0CFCFFFF	MOV DWORD PTR SS:[EBP-3F4], EDX	
004091F2	8995 08FCFFFF	MOV DWORD PTR SS:[EBP-3F8], EDX	
004091F8	8995 14FCFFFF	MOV DWORD PTR SS:[EBP-3EC], EDX	
004091FE	8995 10FCFFFF	MOV DWORD PTR SS:[EBP-3F0], EDX	
00409204	8995 18FCFFFF	MOV DWORD PTR SS:[EBP-3E8], EDX	
0040920A	8945 F4	MOV DWORD PTR SS:[EBP-C], EAX	unpackme.00400000
0040920D	33C0	XOR EAX, EAX	unpackme.00400000
0040920F	55	PUSH EBP	
00409210	68 B39F4000	PUSH unpackme.00409FB3	
00409215	64:FF30	PUSH DWORD PTR FS:[EAX]	
00409218	64:8920	MOV DWORD PTR FS:[EAX], ESP	
0040921B	A1 30B24000	MOV EAX, DWORD PTR DS:[40B230]	
00409220	C600 01	MOV BYTE PTR DS:[EAX], 1	
00409223	A1 C0B04000	MOV EAX, DWORD PTR DS:[40B0C0]	
00409228	50	PUSH EAX	unpackme.00400000
00409229	E8 3EB3FFFF	CALL unpackme.0040456C	JMP to kernel32.SetConsoleTitleA
0040922E	6A 64	PUSH 64	
00409230	E8 4FB3FFFF	CALL unpackme.00404584	JMP to kernel32.Sleep
00409235	E9 9F5E3F00	JMP unpackme.007FF0D9	
0040923A	04 30	ADD AL, 30	
0040923C	68 30F4F4FF	PUSH 55F4F430	

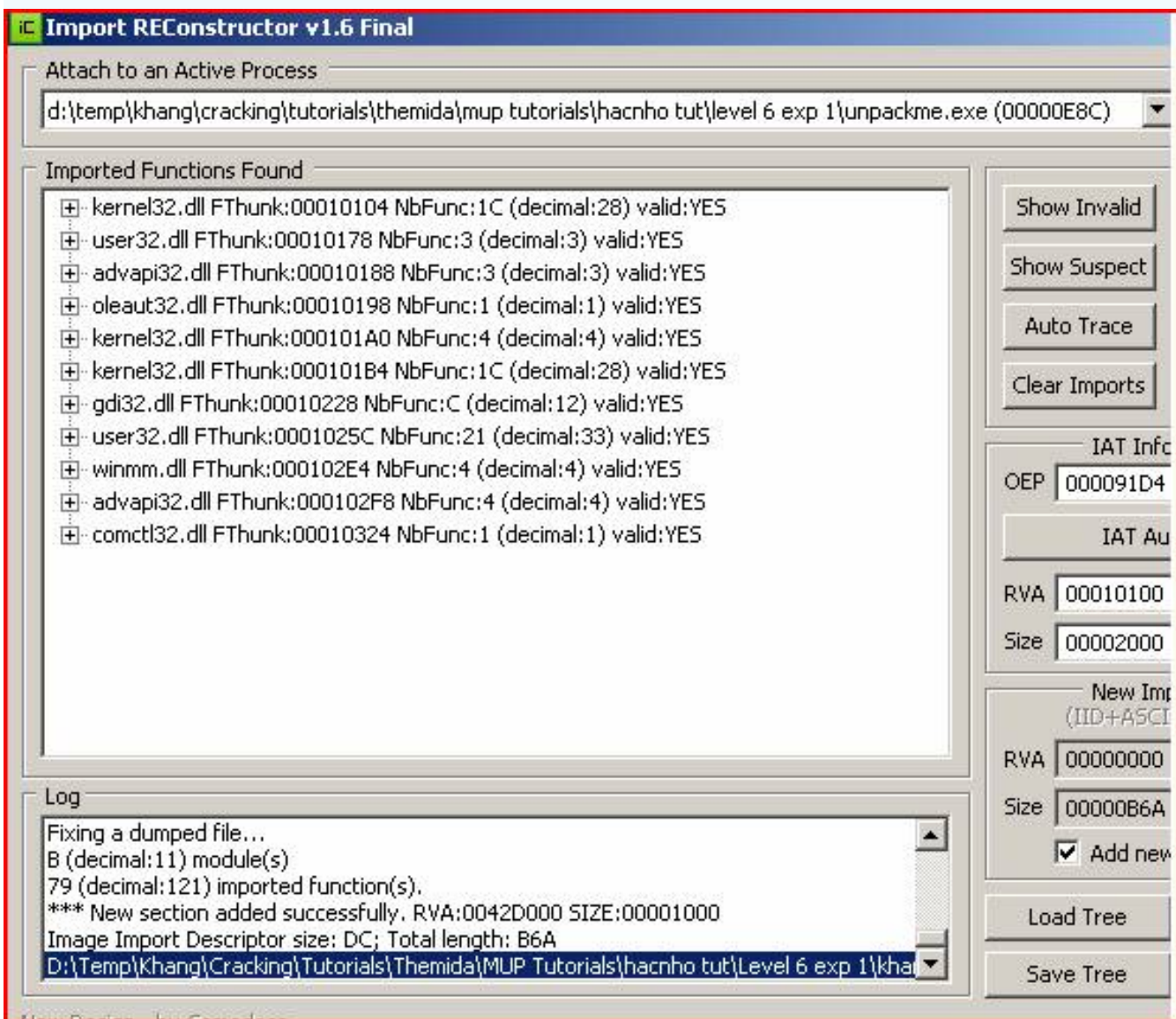
_Ctrl + A:

Address	Hex dump	Disassembly	Comment
004091D4	. 55	PUSH EBP	OEP :0*
004091D5	. 8BEC	MOV EBP, ESP	
004091D7	. 81C4 98FBFFFF	ADD ESP, -468	
004091D0	. 53	PUSH EBX	
004091DE	. 33D2	XOR EDX, EDX	
004091E0	. 8995 98FBFFFF	MOV DWORD PTR SS:[EBP-468], EDX	
004091E6	. 8995 04FCFFFF	MOV DWORD PTR SS:[EBP-3FC], EDX	
004091EC	. 8995 0CFCFFFF	MOV DWORD PTR SS:[EBP-3F4], EDX	
004091F2	. 8995 08FCFFFF	MOV DWORD PTR SS:[EBP-3F8], EDX	
004091F8	. 8995 14FCFFFF	MOV DWORD PTR SS:[EBP-3EC], EDX	
004091FE	. 8995 10FCFFFF	MOV DWORD PTR SS:[EBP-3F0], EDX	
00409204	. 8995 18FCFFFF	MOV DWORD PTR SS:[EBP-3E8], EDX	
0040920A	. 8945 F4	MOV DWORD PTR SS:[EBP-C], EAX	unpackme.00400000
0040920D	. 33C0	XOR EAX, EAX	unpackme.00400000
0040920F	. 55	PUSH EBP	
00409210	. 68 B39F4000	PUSH unpackme.00409FB3	
00409215	. 64:FF30	PUSH DWORD PTR FS:[EAX]	
00409218	. 64:8920	MOV DWORD PTR FS:[EAX], ESP	
0040921B	. A1 30B24000	MOV EAX, DWORD PTR DS:[40B230]	
00409220	. C600 01	MOV BYTE PTR DS:[EAX], 1	
00409223	. A1 C0B04000	MOV EAX, DWORD PTR DS:[40B0C0]	
00409228	. 50	PUSH EAX	
00409229	. E8 3EB3FFFF	CALL unpackme.0040456C	[Title = "MZP"
0040922E	. 6A 64	PUSH 64	SetConsoleTitleA
00409230	. E8 4FB3FFFF	CALL unpackme.00404584	Timeout = 100. ms
00409235	. E9 9F5E3F00	JMP unpackme.007FF0D9	Sleep
0040923A	. 04	DB 04	
0040923B	. 30 68 3A 54	ASCII "0h:T"	
0040923F	. 54	DB 54	CHAR 'T'
00409240	. 5F	DB 5F	CHAR '^'

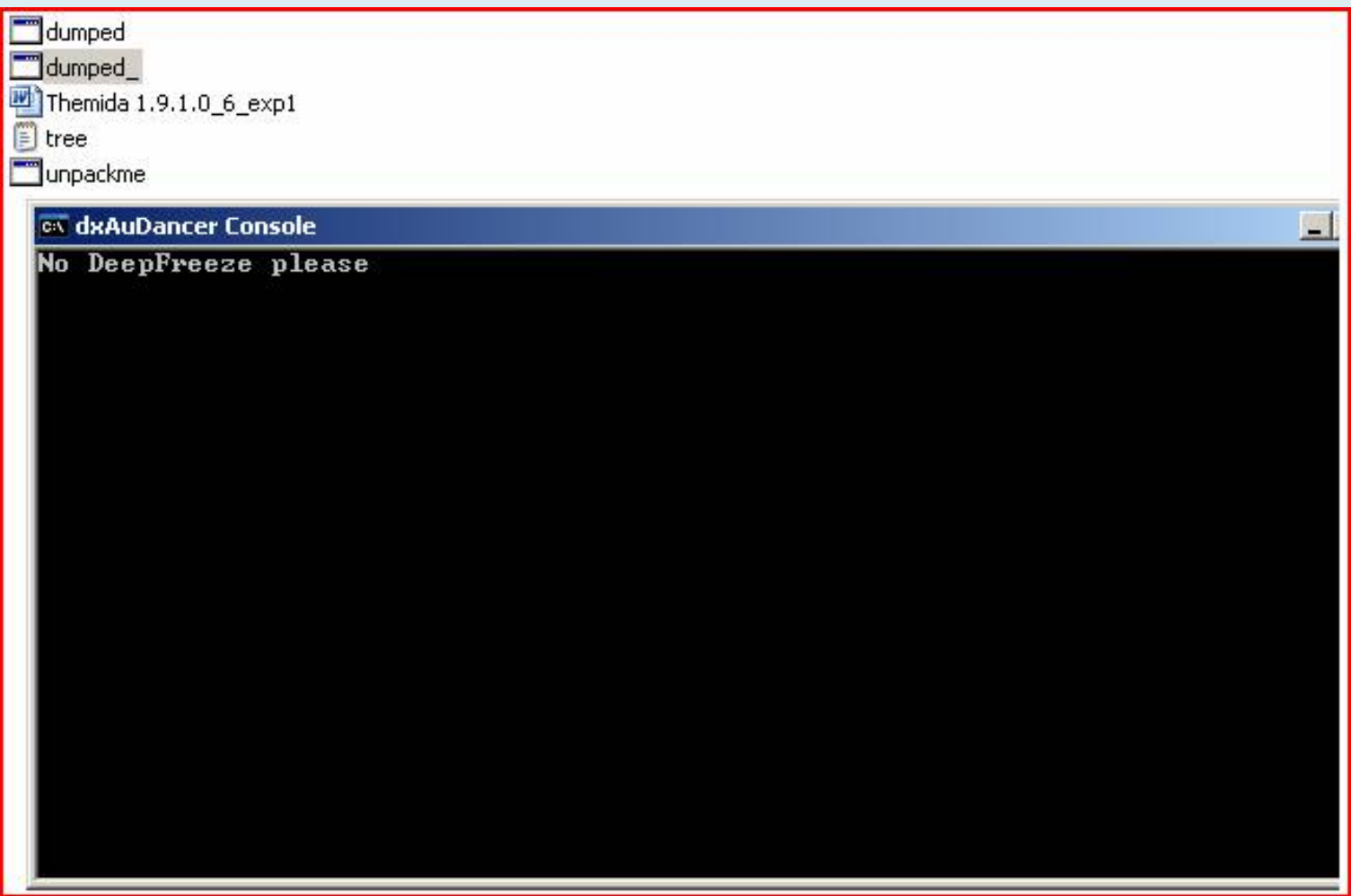
_Dump With LordPE, remember correct ImageSize home:



_Mo ImpREC and enter parameters, and fix dump:



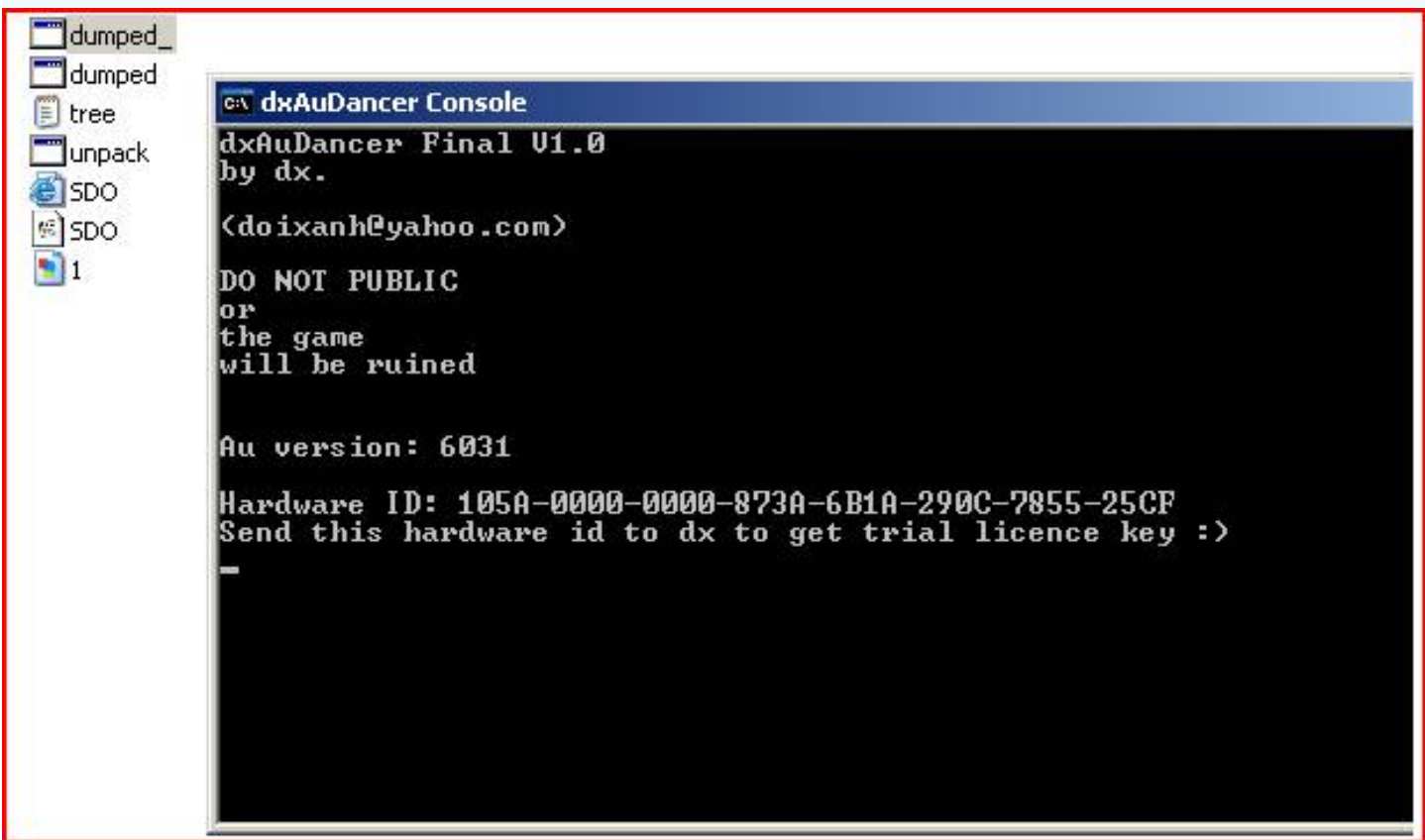
[_Run Try:](#)



_Unpacked Done. Now load dumped into Olly and little: D

00409FC8	ASCII	"No DeepFreeze pl"
00409FD8	ASCII	"ease",0
00409FE0	ASCII	"DeepFrz",0
00409FF0	ASCII	"\drivers\deepfrz"
0040A000	ASCII	".sys",0

_Oai, Do check the driver. Sys DeepFreeze by her, sure to cancel the license: P! Disable DF Now, try running it see why:



_Oh What is this term, difficult hui too =))!

_Tra's A question you do not dump work on another machine, which is due to dump anti VM, in a different tuts will guide me. I'm the dump when the VM code, stack and contexts for ver 1.9.x to the environment debugger it re im running out it crash, then find out it changes the EBP implementation VM. Hix hix. Currently, only fix is only deroko, ông pm, he Privacy: Tao blade, with the VMS is try to create the first J ! Hopefully AoRE fix the J!

_Con Some ask, why del section Themida causing the crash, only simple, soft because of this code do not make that key block for the License Manager Winlic's work, it should del crash is probably obvious: P!

_Tutorial Without target. Only for research only J!

_Công Tool for anyone not already have:

<http://rapidshare.com/files/65258179/HanOlly1.1.rar>

<http://rapidshare.com/files/65255791/OllyICE.rar>

<http://rapidshare.com/files/65252636/PeID.rar> (**pretty pretty =))**)

http://rapidshare.com/files/65253703/ImportREC_v1.6_New.rar

http://rapidshare.com/files/65697523/wark_v1.3.zip

Themida - then will go through

Themida 1.9.xx - 6 Unpackme level 2 Expansion

Tutorial # 6 exp 2 @ 2007 by hacnho (hacnho@hotmail.com)

(Write the gift of friendship REA - all copy must be agreed by the author)

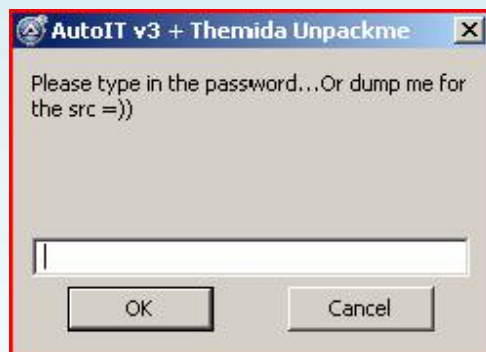
```
[Article title]: Themida 1.9.xx - Unpackme level 6 exp 2
[Author]: hacnho
[Author home page]: http://hvaonline.net
[Target]: AutoIT
[Size]: 591 KB (605.216 bytes)
[Download Page]: Google
[Packer]: Themida | V WinLicense 1.9.xx CB-> Oreans Technologies
[Compilation language]: AutoIT
[Tools]: The0DBG + hideToolz fly by, LordPE, ImportREC, peid0.94
[OS]: WinXP_SP2
[Greetz]: REA, Exetools, PEDIY, UnpackCN, snd and ARTeam Memberz
```

_Chung We meet again, and exposed to the target-related Themida. This time we will do unpackme and unpack it for sure, he he, dangerous pelt: K! In the extended # 2 this we will discuss 3 issues: Dealing with the target code AutoIT v3.2.1.0 and protect with a version number which Themida / Winlic. Created loader themida defeat. Writing scripts to avoid repeating the work nham chán.

I. AUTOIT + THEMIDA: NOT feasible?

_Hom It reads on his blog Computer_Angel see him say Themida + AutoIT easy xôi, tò mò you also know hêm xoi is easy to do because there is not time to Themida is that any child is different again. This week a series of free writing tut Themida, check to themida a target that one can know what is this: P (đặc any theory, any theory đặc =))). It is quite simple, but if it is ver 3.2.8.1 nhé dream, if not to check Allow Decompile the scope tired. In tut I do not discuss this further to AutoIT decompile, only to be a dump when it's src protect it with Themida and compile with ver 3.2.1.0. In CHM version of this series tut, will update more dump src for the next version. Temporarily stop here. Accompanied tut I have attached a Unpackme based 3.2.8.1 compile and pack with themida, if you have handled the AutoIT v3.2.8.1 can apply here: D!

_De Do tut, you need to write a code as a small crackme / unpackme:



_Sau The pack with Themida Custom Build. Pack with the option as follows:

Protection Options for Unpackme_3210.exe

----- Information Macros

Macros VM: 0

CodeReplace Macros: 0

ENCRYPT Macros: 0

CLEAR Macros: 0

XBundler files

No files to bundle
Protection Options

Anti-Debugger: enabled

Anti-Dumpers: enabled

API-wrapping: enabled

Virtual Machine: enabled

Entry Point Ofuscation: enabled

Memory Guard: enabled

Anti-Monitor File: enabled

Anti-Monitor Registry: enabled

Resource Encryption: enabled

VMWare compatible: enabled

Delphi / BCB form protection: enabled

Advanced Protection Options

Encrypt Application: enabled

. NET assemblies: Disabled

Dll plugin: Disabled

Active Context: enabled

Section Last Name: Themida
Compression

Application compression: enabled

Resources compression: enabled

SecureEngine compression: enabled

Virtual Machine Settings

Number of Virtual wrapped APIs: 0

Entry Point Virtualization: 0 instructions

Virtual Machine Processor: Mutable CISC processor

Number of CPUs: 1

Opcode Type: Static opcodes

Dynamic Opcode: Disabled

_Truoc Unpack when I say over previous theory, with autoit original, pack it with UPX, simply find the string> AUTOIT SCRIPT <3.2.1.0 or in>>> AUTOIT SCRIPT <<<in 3.8.2.1, an Call now breakpoint commands, and trace down memory dump ECX. But with Themida the subject, you have a way to OEP first, but whether to have the full OEP IAT where the search string: P. So with themida we determine the following steps: By pass the type of Themida protect, prevent VMS, restore the OEP if so, get Full IAT, to stop it, from the need to dump and IAT fix for tired, that is src get it.

Now here we go:

00475014	B8 00000000	MOV EAX, 0
00475019	60	PUSHAD
0047501A	0BC0	OR EAX, EAX
0047501C	74 68	JE SHORT Unpackme.00475086
0047501E	E8 00000000	CALL Unpackme.00475023
00475023	58	POP EAX
00475024	05 53000000	ADD EAX, 53
00475029	8038 E9	CMP BYTE PTR DS:[EAX], 0E9

_De Avoid rewriting steps unpack, please review tut 6 exp 1. I only make place is important to patch:

a.

005480E7	0F85 0A000000	JNZ Unpackme.005480F7
005480ED	C785 D0031B07 0	MOV DWORD PTR SS:[EBP+71B03DD], 1
005480E7	EB 0E	JMP SHORT Unpackme.005480F7
005480F9	AD00	OR DL, BYTE PTR DS:[EDX+1]

b.

00548226	0F84 39000000	JE Unpackme.00548265
0054822C	3B8D 091A1B07	CMP ECX, DWORD PTR SS:[EBP+71B1AD9]
00548232	0F84 2D000000	JE Unpackme.00548265
00548238	3B8D A5231B07	CMP ECX, DWORD PTR SS:[EBP+71B23A5]
0054823E	0F84 21000000	JE Unpackme.00548265
00548244	3B8D A5261B07	CMP ECX, DWORD PTR SS:[EBP+71B26A5]
0054824A	0F84 15000000	JE Unpackme.00548265
00548250	809D 9B3C2207	LEA EBX, DWORD PTR SS:[EBP+7223C9B]
00548226	EB 28	JMP SHORT Unpackme.00548250
00548228	3900	CMP DWORD PTR DS:[EAX], EAX
0054822A	0000	ADD BYTE PTR DS:[EAX], AL
0054822C	3B8D 091A1B07	CMP ECX, DWORD PTR SS:[EBP+71B1AD9]
00548232	0F84 2D000000	JE Unpackme.00548265
00548238	3B8D A5231B07	CMP ECX, DWORD PTR SS:[EBP+71B23A5]
0054823E	0F84 21000000	JE Unpackme.00548265
00548244	3B8D A5261B07	CMP ECX, DWORD PTR SS:[EBP+71B26A5]
0054824A	0F84 15000000	JE Unpackme.00548265
00548250	809D 9B3C2207	LEA EBX, DWORD PTR SS:[EBP+7223C9B]

c.

0054893B	8908	MOV DWORD PTR DS:[EAX],ECX
0054893D	AD	LODS DWORD PTR DS:[ESI]
0054893E	C746 FC 00000000	MOV DWORD PTR DS:[ESI-4],0
0054893B	-E9 C0761102	JMP 02660000
00548940	FC	CLD
00548941	0000	ADD BYTE PTR DS:[EDI],0

d.

005489F1	AA	STOS BYTE PTR ES:[EDI]
005489F2	✓E9 24000000	JMP Unpackme.00548A1B
005489F7	58	POP EAX
005489F8	00	STOS BYTE PTR ES:[EDI]
005489F1	AA	STOS BYTE PTR ES:[EDI]
005489F2	-E9 1D761102	JMP 02660014
005489F7	58	POP EAX
005489F8	00	STOS BYTE PTR ES:[EDI]

e.

005489F3	807F FF E9	CMP BYTE PTR DS:[EDI-1],0E9
005489FD	✓0F85 18000000	JNZ Unpackme.00548A1B
005489A3	83B0 80272207	CMP DWORD PTR SS:[EBP+7227280],0
005489F3	807F FF E9	CMP BYTE PTR DS:[EDI-1],0E9
005489FD	-E9 34761102	JMP 02660036
00548A02	0083 BD802722	ADD BYTE PTR DS:[EBX+222780BD],AL
00548A03	07	PNP ES

f.

00548A2B	C746 FC 00000000	MOV DWORD PTR DS:[ESI-4],0
00548A2F	^E9 11FFFFFF	JMP Unpackme.00548945
00548A34	89B5 492F1B07	MOV DWORD PTR SS:[EBP+71B2F49],ESI
00548A2B	C746 FC 00000000	MOV DWORD PTR DS:[ESI-4],0
00548A2F	-E9 2B761102	JMP 0266005F
00548A34	89B5 492F1B07	MOV DWORD PTR SS:[EBP+71B2F49],ESI

g.

00548A23	03E0 04	SUB EAX,4
00548A26	AB	STOS DWORD PTR ES:[EDI]
00548A27	AD	LODS DWORD PTR DS:[ESI]
00548A23	83E8 04	SUB EAX,4
00548A26	90	NOP
00548A27	AD	LODS DWORD PTR DS:[ESI]

h.

00548A16	FFD3	CALL EBX
00548A18	8947 04	MOV BYTE PTR DS:[EDI+4],AL
00548A1A	83B5 052F1B07	MOV DWORD PTR SS:[EBP+71B2FD5],ESI
00548A18	90	NOP
00548A19	90	NOP
00548A1A	90	NOP
00548A1B	8B85 D52F1B07	MOV EAX,DWORD PTR SS:[EBP+71B2FD5]

And this is inject code:

02660000	A3 00016602	MOV DWORD PTR DS:[2660100],EAX
02660005	8908	MOV DWORD PTR DS:[EAX],ECX
02660007	AD	LODS DWORD PTR DS:[ESI]
02660008	C746 FC 00000000	MOV DWORD PTR DS:[ESI-4],0
0266000F	-E9 3189EEFD	JMP Unpackme.00548945
02660014	50	PUSH EAX
02660015	A1 00016602	MOV EAX,DWORD PTR DS:[2660100]
0266001A	8907	MOV DWORD PTR DS:[EDI],EAX
0266001C	807F FF E8	CMP BYTE PTR DS:[EDI-1],0E8
02660020	✓75 08	JNZ SHORT 0266002A
02660022	66:C747 FE FF15	MOV WORD PTR DS:[EDI-2],15FF
02660028	✓EB 06	JMP SHORT 02660030
0266002A	66:C747 FE FF25	MOV WORD PTR DS:[EDI-2],25FF
02660030	58	POP EAX
02660031	-E9 E589EEFD	JMP Unpackme.00548A1B
02660036	50	PUSH EAX
02660037	A1 00016602	MOV EAX,DWORD PTR DS:[2660100]
0266003C	8947 01	MOV DWORD PTR DS:[EDI+1],EAX
0266003F	807F FF E8	CMP BYTE PTR DS:[EDI-1],0E8
02660043	✓75 08	JNZ SHORT 0266004D
02660045	66:C747 FF FF15	MOV WORD PTR DS:[EDI-1],15FF
0266004B	✓EB 06	JMP SHORT 02660053
0266004D	66:C747 FF FF25	MOV WORD PTR DS:[EDI-1],25FF
02660053	58	POP EAX
02660054	-0F85 C189EEFD	JNZ Unpackme.00548A1B
0266005A	-E9 A489EEFD	JMP Unpackme.00548A03
0266005F	83C7 04	ADD EDI,4
02660062	-E9 DE88EEFD	JMP Unpackme.00548945
02660067	90	NOP
02660068	0000	ADD BYTE PTR DS:[EAX],AL
0266006A	0000	ADD BYTE PTR DS:[EAX],AL

A3 00 01 66 02 89 08 AD C7 46 FC 00 00 00 00 E9 31 89 EE FD 50 A1 00 01 66 02 89 07 80 FF 7F
E8 75 08 66 C7 47 FE FF 15 EB 06 66 C7 47 FE FF 25 58 E9 E5 89 EE FD 50 A1 00 01 66 02 89 47
01 80 7F FF E8 75 08 66 C7 47 FF FF 15 EB 06 66 C7 47 FF FF 25 58 0F 85 C1 89 EE FD E9 A4 89

EE FD 83 C7 04 E9 DE 88 EE FD 90

Ctrl + B:

Enter binary string to search for

ASCII:

UNICODE:

HEX +05:

☒ Entire block

☐ Case sensitive

<< >>

OK Cancel

Toi Here:

00548A64	^E9 10F5FFFF	JMP Unpackme.00547F79
00548A69	^E9 A4060000	JMP Unpackme.00549112
00548A6E	60	PUSHAD
00548A6F	8B80 000E1B07	MOV ECX, DWORD PTR SS:[EBP+71B0E00]
00548A75	8B09	MOV ECX, DWORD PTR DS:[ECX]

Dat Execution on a breakpoint, press F9, remove breakpoint, Alt + M placed on access breakpoint, press F9:

004422E4	\$ 6A 60	PUSH 60
004422E6	68 300A4500	PUSH Unpackme.00450A30
004422EB	E8 70050000	CALL Unpackme.00442860
004422F0	BF 94000000	MOV EDI, 94
004422F5	8BC7	MOV EAX, EDI
004422F7	E8 14680000	CALL Unpackme.00448B10
004422FC	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP
004422FF	8BF4	MOV ESI, ESP
00442301	893E	MOV DWORD PTR DS:[ESI], EDI
00442303	56	PUSH ESI
00442304	FF15 84024500	CALL DWORD PTR DS:[<&KERNEL32.GetVersion
0044230A	8B4E 10	MOV ECX, DWORD PTR DS:[ESI+10]
0044230D	890D 08C44500	MOV DWORD PTR DS:[45C408], ECX
00442313	8B46 04	MOV EAX, DWORD PTR DS:[ESI+4]

pVersionInformation
GetVersionExA

Ok. OEP has to. Now Right-click search String reference: Search string:

Address	Disassembly	Text string
0040EA5E	PUSH Unpackme.004522B4	ASCII "#include-once"
0040EA59	PUSH Unpackme.004522A8	ASCII "#include"
0040EA5B	PUSH Unpackme.00452290	ASCII "Cannot parse #include"
0040EB0D	PUSH Unpackme.00452318	ASCII "AutoIt"
0040EB2A	PUSH Unpackme.00452280	ASCII "#comments-start"
0040EB3E	PUSH Unpackme.0045227C	ASCII "#cs"
0040EB88	PUSH Unpackme.00452280	ASCII "#comments-start"
0040EB9C	PUSH Unpackme.0045227C	ASCII "#cs"
0040EBB0	PUSH Unpackme.00452240	ASCII "#comments-end"
0040EBC4	PUSH Unpackme.0045223C	ASCII "#ce"
0040EC0F	PUSH Unpackme.00452250	ASCII "Unterminated group of comment in file :!%s"
0040ECE7	PUSH Unpackme.0045222C	ASCII ">AUTOIT SCRIPT<"
0040EE37	PUSH Unpackme.004522C4	ASCII "#notrayicon"
0040EE55	PUSH Unpackme.00452280	ASCII "#comments-start"
0040EE69	PUSH Unpackme.0045227C	ASCII "#cs"
0040EE8C	PUSH Unpackme.00452280	ASCII "#comments-start"
0040EEA0	PUSH Unpackme.0045227C	ASCII "#cs"
0040EEB4	PUSH Unpackme.00452240	ASCII "#comments-end"
0040FFC8	PUSH Unpackme.0045223C	ASCII "#ne"

Nhan You enter here will be to:

0040ECE3	8D45 0C	LEA EAX, DWORD PTR SS:[EBP+C]
0040ECE6	50	PUSH EAX
0040ECE7	68 2C224500	PUSH Unpackme.0045222C
0040ECEC	8D45 E8	LEA EAX, DWORD PTR SS:[EBP-18]
0040ECF3	E8 31030300	CALL Unpackme.0043F025
0040ECF4	85C0	TEST EAX, EAX
0040ECF6	74 22	JE SHORT Unpackme.0040ED1A
0040ECF8	395D E8	CMP DWORD PTR SS:[EBP-18], EBX
0040ECFB	74 09	JE SHORT Unpackme.0040ED06
0040ECFD	FF75 E8	PUSH DWORD PTR SS:[EBP-18]
0040ECFF	50	PUSH EAX

Arg2
Arg1 = 0045222C ASCII ">AUTOIT SCRIPT<"
Set breakpoint tai day!

Nhan F2, then press F9 2 times, meeting house, it does not crash OK, so on, if you crash is not yet complete, as the nhé.

Now press F8 one time:

0040ECE3	8D45 0C	LEA EAX, DWORD PTR SS:[EBP+C]
0040ECE6	50	PUSH EAX
0040ECE7	68 2C224500	PUSH Unpackme.0045222C
0040ECEC	8D45 E8	LEA EAX, DWORD PTR SS:[EBP-18]
0040ECF3	E8 31030300	CALL Unpackme.0043F025
0040ECF4	85C0	TEST EAX, EAX
0040ECF6	74 22	JE SHORT Unpackme.0040ED1A
0040ECF8	395D E8	CMP DWORD PTR SS:[EBP-18], EBX
0040ECFB	74 09	JE SHORT Unpackme.0040ED06
0040ECFD	FF75 E8	PUSH DWORD PTR SS:[EBP-18]
0040ECFF	50	PUSH EAX

Arg2
Arg1 = 0045222C ASCII ">AUTOIT SCRIPT<"
Set breakpoint tai day!

Nhin Register through the window:


```
Registers (FPU)
EAX 00000000
ECX 003DAE98 ASCII "; <AUT2EXE VERSION: 3.2.0.1>";
EDX 00000000
EBX 00000000
ESP 0012E7AC
EBP 0012F7D0
ESI 00466FC0 Unpackme.00466FC0
EDI 0012F7B8
EIP 0040ECF4 Unpackme.0040ECF4
```

Follow _Gio dump in and look down dump small window:

```
003DAD98 .....
003DADD8 .....
003DAE18 .....
003DAE58 .....
003DAE98 ; <AUT2EXE VERSION: 3.2.0.1>....; -----
003DAED8 .....; <AUT2EXE INCLU
003DAF18 DE-START: C:\Documents and Settings\hacnho\My Documents\Unpackme
003DAF58 .au3>...; -----
003DAF98 .....#Region ;**** Directives created by Aut
003DAFD8 oIt3Wrapper_GUI *****#AutoIt3Wrapper_Version=Beta..#AutoIt3Wrap
003DB018 per_UseUpX=n..#EndRegion ;**** Directives created by AutoIt3Wrap
003DB058 per_GUI *****; <AUT2EXE VERSION: v3.2.8.1>....; -----
003DB098 .....;
003DB0D8 <AUT2EXE INCLUDE-START: C:\Documents and Settings\hacnho\My Doc
003DB118 uments\Unpackme\unpackme.au3>...; -----
003DB158 .....; AutoIt Ver
003DB198 sion: 3.2.8.1...; Language: English...; Platform: Win9
003DB1D8 x/NT/XP...; Author: hacnho (hacnho@hotmail.com)...; Scr
003DB218 ipt Function: Demo an input box...;.....; Loop around until the
003DB258 user gives a valid "autoit" answer..$bLoop = 1..While $bLoop =
003DB298 1.. $text = InputBox("AutoIT v3 + Themida Unpackme", "Please
003DB2D8 type in the password...Or dump me for the src =)").. If @err
003DB318 or = 1 Then.. MsgBox(4096, "Goodbye", "A counsel for you
003DB358 : Foget Themida :D!").. Exit.. Else.. ; They clicke
003DB398 d OK, but did they type the right thing?.. If $text <> "h
003DB3D8 acnho is so good :P!" Then.. Else.. $bLoop
003DB418 = 0 ; Exit the loop - ExitLoop would have been an alternativ
003DB458 e too :).. EndIf.. EndIf..WEnd....; Print the success
003DB498 message..MsgBox(4096,"Done", "Good job, boy!")....; Finished!..
003DB4D8 ...;
003DB518 .....; <AUT2EXE INCLUDE-END: C:\Documents and Setti
003DB558 ngs\hacnho\My Documents\Unpackme\unpackme.au3>...; -----
003DB598 .....;
003DB5D8 .....;
003DB618 .....; <AUT2EXE INCLUDE-END: C:\Documents and Set
003DB658 tings\hacnho\My Documents\Unpackme.au3>...; -----
003DB698 .....
```

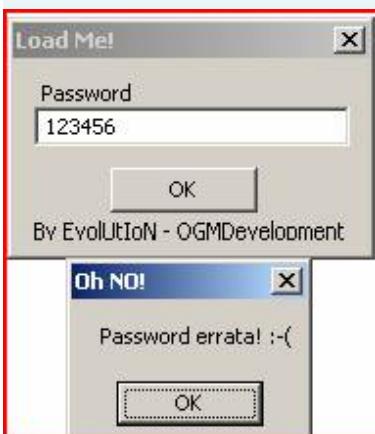
_Don But the hi: P!

II. CREATE LOADER THEMIDA TREATMENT

_Khi You no way to unpack themida it is fully running, there are 3 ways to target treatment, the first is to create a standard loader to patch the memory, then the program has the springiness compressed memory, the second is inline patch, and this requires more elaborately, we must first pass by the inspection function as CRC checksum, MD5 εξαμπλε, then write a code in a position that is vacant and in a section that; ;), is the task that lies in the area, the program has run a variety nhào bup line. And 3 is to forget it, find something to do with J ! In this part 2 we will write a loader to handle the target packed with 1.8.xx themida mission is set to be one to OEP, trace need to find meat (which is another way to stack his pp Moon), written loader, use the function to read the writing process bác Bill to handle it. We will turn over 2 target, in an ASM (thx Evolution, my camarade) and one written in VB (thx DucCu: P). _Neu You have made through the loader not to discuss what, if not read the invitation of the tut

Translated by Arteam bract: **Cracking with Loader v1.1.**

1.Target 1:



_Khong Discuss how to find OEP again, now is the work of you. Suppose now you are in the original OEP:

00401030	68 44304000	PUSH LoadMe_n.00403044	pOldProtect = LoadMe_n.00403044
00401035	6A 40	PUSH 40	NewProtect = PAGE_EXECUTE_READWRITE
00401037	68 00100000	PUSH 1000	Size = 1000 (4096.)
0040103C	68 00104000	PUSH LoadMe_n.00401000	Address = LoadMe_n.00401000
00401041	E8 E0000000	CALL <JMP.&kernel32.VirtualProtect>	VirtualProtect
00401046	51	PUSH ECX	
00401047	50	PUSH EAX	
00401048	B8 00304000	MOV EAX, LoadMe_n.00403000	ASCII "OGM"
0040104D	8B00	MOV EAX, DWORD PTR DS:[EAX]	
0040104F	B9 00154000	MOV ECX, LoadMe_n.00401500	
00401054	8901	MOV DWORD PTR DS:[ECX], EAX	
00401056	58	POP EAX	
00401057	59	POP ECX	
00401058	6A 00	PUSH 0	pModule = NULL
0040105A	E8 C1000000	CALL <JMP.&kernel32.GetModuleHandleA>	GetModuleHandleA
0040105F	A3 40304000	MOV DWORD PTR DS:[403040], EAX	
00401064	6A 00	PUSH 0	lParam = NULL
00401066	68 80104000	PUSH LoadMe_n.00401080	DlgProc = LoadMe_n.00401080
0040106B	6A 00	PUSH 0	hOwner = NULL
0040106D	6A 65	PUSH 65	pTemplate = 65
0040106F	FF35 40304000	PUSH DWORD PTR DS:[403040]	hInst = NULL
00401075	E8 88000000	CALL <JMP.&user32.DialogBoxParamA>	DialogBoxParamA
0040107A	50	PUSH EAX	ExitCode
0040107B	E8 9A000000	CALL <JMP.&kernel32.ExitProcess>	ExitProcess
00401080	55	PUSH EBP	
00401081	8BEC	MOV EBP, ESP	
00401083	817D 0C 110100	CMP DWORD PTR SS:[EBP+C], 111	
0040108A	75 60	JNZ SHORT LoadMe_n.004010EC	
0040108C	817D 10 EC0300	CMP DWORD PTR SS:[EBP+10], 3EC	
00401093	75 67	JNZ SHORT LoadMe_n.004010FC	
00401095	6A 08	PUSH 8	Count = 8
00401097	68 4C304000	PUSH LoadMe_n.0040304C	Buffer = LoadMe_n.0040304C
0040109C	68 E9030000	PUSH 3E9	ControlID = 3E9 (1001.)
004010A1	FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd
004010A4	E8 65000000	CALL <JMP.&user32.GetDlgItemTextA>	GetDlgItemTextA
004010A9	C705 48304000	MOV DWORD PTR DS:[403048], LoadMe_n.00403048	
004010B3	68 48304000	PUSH LoadMe_n.00403048	
004010B8	E8 43FFFFFF	CALL LoadMe_n.00401000	
004010BD	83F8 FF	CMPL EAX, -1	
004010C0	75 15	JNZ SHORT LoadMe_n.004010D7	
004010C2	6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
004010C4	68 1B304000	PUSH LoadMe_n.0040301B	Title = "OK!"
004010C9	68 04304000	PUSH LoadMe_n.00403004	Text = "Password corretta! :-)"
004010CE	6A 00	PUSH 0	hOwner = NULL
004010D0	E8 3F000000	CALL <JMP.&user32.MessageBoxA>	MessageBoxA
004010D5	EB 13	JMP SHORT LoadMe_n.004010EA	
004010D7	6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
004010D9	68 34304000	PUSH LoadMe_n.00403034	Title = "Oh NO!"
004010DE	68 1F304000	PUSH LoadMe_n.0040301F	Text = "Password errata! :-("
004010F3	6A 00	PUSH 0	hOwner = NULL

_Nhin To see us string OGM is correct, but we assume hem bit (because the actual process of creating a number of key soft very complicated. And we need to patch this site:

00401000	. 55	PUSH EBP	
00401001	. 8BEC	MOV EBP,ESP	
00401003	. 57	PUSH EDI	
00401004	. 56	PUSH ESI	
00401005	. 51	PUSH ECX	
00401006	. BF 00304000	MOV EDI,LoadMe_n.00403000	ASCII "OGM"
00401008	. 8B75 08	MOV ESI,DWORD PTR SS:[EBP+8]	
0040100E	. 8B36	MOV ESI,DWORD PTR DS:[ESI]	
00401010	. 33C9	XOR ECX,ECX	
00401012	. 33C0	XOR EAX,EAX	
00401014	. 75 09	JMP SHORT LoadMe_n.0040101F	
00401016	. 8A07	MOV AL,BYTE PTR DS:[EDI]	
00401018	. 3B06	CMP BYTE PTR DS:[ESI],AL	
0040101A	. 75 10	JNZ SHORT LoadMe_n.0040102C	Patch here
0040101C	. 47	INC EDI	
0040101D	. 46	INC ESI	
0040101E	. 41	INC ECX	
0040101F	. 83F9 04	CMP ECX,4	
00401022	. 75 F2	JNZ SHORT LoadMe_n.00401016	
00401024	. 00	MOV ECX,-1	

Registers (MMX)	
EAX	0000004F
ECX	00000000
EDX	00150608
EBX	00000000
ESP	0012FBB0
EBP	0012FBB0
ESI	0040304C ASCII "123456"
EDI	00403000 ASCII "OGM"
EIP	00401018 LoadMe_n.00401018
C 1	ES 0023 32bit 0(FFFFFFFF)
P 1	CS 001B 32bit 0(FFFFFFFF)

_Dia Just a patch to patch 40101A and submitted. Now we are written in ASM nhé:

486

.model flat, stdcall

casemap option: none

```
include \masm32 \include \windows.inc
```

```
include \masm32 \include \user32.inc
```

```
include \masm32 \include \kernel32.inc
```

```
include \masm32 \include \advapi32.inc
```

```
includelib \masm32 \lib \user32.lib
```


includelib \masm32 \lib \kernel32.lib

includelib \masm32 \lib \advapi32.lib

.data

db program "LoadMe.exe", 0

wordset db 90h, 90h

wordtopatch db 75H, 10h

addresstopatch DWORD 40101Ah

.data?

cProcStart

cProcInfos

STARTUPINFO <>

PROCESS_INFORMATION <>

wordread db 10h dup (?)

byteswritten DWORD?

.code

start:

invoke CreateProcess, NULL, addr program, NULL, NULL, NULL, CREATE_SUSPENDED, NULL, NULL, addr cProcStart, addr

cProcInfos

;

xor eax, eax

. EAX while! = 1

invoke ResumeThread, cProcInfos.hThread

invoke Sleep, 5

invoke SuspendThread, cProcInfos.hThread

invoke ReadProcessMemory, cProcInfos.hProcess, addresstopatch, addr wordread, 2h, addr byteswritten

xor ecx, ecx

xor edx, edx

xor eax, eax

mov ecx, OFFSET wordread

mov cl, byte ptr [ECX]

mov edx, OFFSET wordtopatch

mov dl, byte ptr [EDX]

.if cl! = dl

xor eax, eax

jmp OFFSET resume

.endif

mov ecx, OFFSET wordread

inc ecx

mov cl, byte ptr [ECX]

mov edx, OFFSET wordtopatch

inc edx

mov dl, byte ptr [EDX]

. if cl! = dl

xor eax, eax

jmp OFFSET resume

. endif

mov eax, 1

Resume:

.endw

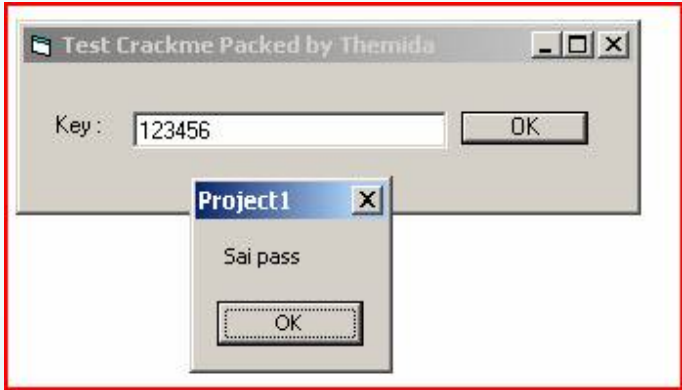
invoke WriteProcessMemory, cProcInfos.hProcess, addresstopatch, addr wordset, 2h, addr byteswritten

invoke ResumeThread, cProcInfos.hThread

RET

END start

2. Target 2



_OEP's Target:

0040117C	.-FF25 4C104000	JMP	DWORD PTR DS:[<&msvbvm60.EVENT_SINK	msvbvm60.EVENT_SINK_QueryInterface
00401182	.-FF25 38104000	JMP	DWORD PTR DS:[<&msvbvm60.EVENT_SINK	msvbvm60.EVENT_SINK_AddRef
00401188	.-FF25 44104000	JMP	DWORD PTR DS:[<&msvbvm60.EVENT_SINK	msvbvm60.EVENT_SINK_Release
0040118E	.-FF25 78104000	JMP	DWORD PTR DS:[<&msvbvm60.ThunRTMain	msvbvm60.ThunRTMain
00401194	\$ 68 54134000	PUSH	dumped_.00401354	
00401199	. E8 F0FFFFFF	CALL	<JMP.&msvbvm60.ThunRTMain>	
0040119E	. 0000	ADD	BYTE PTR DS:[EAX],AL	
004011A0	. 0000	ADD	BYTE PTR DS:[EAX],AL	
004011A2	. 0000	ADD	BYTE PTR DS:[EAX],AL	
004011A4	. 3000	XOR	BYTE PTR DS:[EAX],AL	
004011A6	. 0000	ADD	BYTE PTR DS:[EAX],AL	
004011A8	. 40	INC	EAX	
004011A9	. 0000	ADD	BYTE PTR DS:[EAX],AL	
004011AB	. 0000	ADD	BYTE PTR DS:[EAX],AL	
004011AD	. 0000	ADD	BYTE PTR DS:[EAX],AL	
004011AF	. 0017	ADD	BYTE PTR DS:[EDI],DL	
004011B1	. AA	STOS	BYTE PTR ES:[EDI]	
004011B2	. 312F	XOR	DWORD PTR DS:[EDI],EBP	

_Gio Search any site considered necessary patch:

00401DEB	. 8945 B8	MOV DWORD PTR SS:[EBP-48],EAX	
00401DC2	. 74 43	JE SHORT dumped_.00401E07	NOP
00401DC4	. 8D55 88	LEA EDX,DWORD PTR SS:[EBP-78]	
00401DC7	. 8D4D C8	LEA ECX,DWORD PTR SS:[EBP-38]	
00401DCA	. C745 90 1C184	MOV DWORD PTR SS:[EBP-70],dumped_.00401E07	UNICODE "Dung pass"
00401DD1	. C745 88 08000	MOV DWORD PTR SS:[EBP-78],8	
00401DD8	. FF15 7C104000	CALL DWORD PTR DS:[<&msvbm60.__vbaVarDup	msvbm60.__vbaVarDup
00401DDE	. 8D55 98	LEA EDX,DWORD PTR SS:[EBP-68]	
00401DE1	. 8D45 A8	LEA ECX,DWORD PTR SS:[EBP-58]	
00401DE4	. 52	PUSH EDX	
00401DE5	. 8D4D B8	LEA ECX,DWORD PTR SS:[EBP-48]	
00401DE8	. 50	PUSH EAX	
00401DE9	. 51	PUSH ECX	
00401DEA	. 8D55 C8	LEA EDX,DWORD PTR SS:[EBP-38]	
00401DED	. 57	PUSH EDI	
00401DEE	. 52	PUSH EDX	
00401DEF	. FF15 24104000	CALL DWORD PTR DS:[<&msvbm60.rtcMsgBox	msvbm60.rtcMsgBox
00401DF5	. 8D45 98	LEA ECX,DWORD PTR SS:[EBP-68]	
00401DF8	. 8D4D A8	LEA ECX,DWORD PTR SS:[EBP-58]	
00401DFB	. 50	PUSH EAX	
00401DFC	. 8D55 B8	LEA EDX,DWORD PTR SS:[EBP-48]	
00401DFF	. 51	PUSH ECX	
00401E00	. 8D45 C8	LEA ECX,DWORD PTR SS:[EBP-38]	
00401E03	. 52	PUSH EDX	
00401E04	. 50	PUSH EAX	
00401E05	. EB 41	JMP SHORT dumped_.00401E48	
00401E07	. 8D55 88	LEA EDX,DWORD PTR SS:[EBP-78]	
00401E0A	. 8D4D C8	LEA ECX,DWORD PTR SS:[EBP-38]	
00401E0D	. C745 90 34184	MOV DWORD PTR SS:[EBP-70],dumped_.00401E07	UNICODE "Sai pass"
00401E14	. C745 88 08000	MOV DWORD PTR SS:[EBP-78],8	
00401E1B	. FF15 7C104000	CALL DWORD PTR DS:[<&msvbm60.__vbaVarDup	msvbm60.__vbaVarDup
00401E21	. 8D4D 98	LEA ECX,DWORD PTR SS:[EBP-68]	

_Cai For your practice. If it does not bit dUP it neat: P!

III. VIẾT SCRIPT

_Neu You unpack a dozen near themida target protected by the work place really nham feet, and we should have thought of writing a script to change it as we all work like magic fix jump, OEP not find? Now we will write a simple script only, include:

1. There are tasks set breakpoint, press the key combination such as Shift + F9, F7, F8 ...
2. Instead of typing Ctrl + B to enter the code insensitivity, we use function findmem or find eip. Use asm mov to patch or replace the command Ctrl + E.
3. Alloc use alternative for creating a region memory space.
4. Use dump command to dump the process from using LordPE ...

_De Illustrator for writing a script I used unpackme is AutoIT unpackme above. Note in the analysis I will use the format code Author (pseudocode) for analysis, the conclusion we will write to a complete nhé.

_Buoc 1: We open the Notepad or any editor, I use the Debug Olly Script Editor 1.2. Then use OllyDBG open on target, we will here:

00475014	B8 00000000	MOV EAX,0
00475019	60	PUSHAD
0047501A	0BC0	OR EAX,EAX
0047501C	74 68	JE SHORT Unpackme.00475086
0047501E	E8 00000000	CALL Unpackme.00475023
00475023	58	POP EAX
00475024	05 53000000	ADD EAX,53
00475029	8032 F9	CMPL RVT, PTR DS:[EAX] 0F9

_Nhu Of time before we press Alt + M to set a breakpoint on write. The command which is equivalent in

the language OllyScript:

/*

BPWM addr, size: Set breakpoint on memory write. Size is the size of memory in bytes.

Example: bpwm 401,000, FF

* /

_Chung We set a breakpoint in the code section with addr, size is 401,000, 6F000. So we will have the

first order: 401,000.6 bpwm F000. In a written script so if we will use the variables, this variable to assign

the target to change the script will take. To get two of this, we use GMI:

/ *

Gets information about a module to which the specified address belongs.

"info" can be:

MODULEBASE, MODULESIZE, CODEBASE, ESIZE COD, MEMBASE, MEMSIZE, ENTRY, NSECT, database,

RELOCTABLE, RELOCSIZE RESBASE, RESSIZE, IDATABASE, DATATABLE, EDATATABLE, and strings EDATASIZE

NAME, PATH, VERSION

(If you want other info in the future versions plz tell me).

Sets the reserved \$ RESULT variable (0 if data not found).

Example:

GMI eip, CODEBASE // After this \$ Result is the address to the codebase of the modules to which belongs eip

* /

_Chu That three letters this: CODEBASE, MODULEBASE, CODESIZE, is three variables we need to

get information. Retrieved finished we will assign to a temporary variable \$ RESULT (Return value for

some functions like FIND etc.). OK if you follow this to J , we will have the following script:

/*

Var codebase // declare variables to use, this value is a section of code addr

Var codesize // Size of the code section

Var modulebase // PE Header

gmi eip, MODULEBASE // Get information PE Header

mov modulebase, \$ RESULT // Takes PE Header to change (at this time is 40,000)

gmi eip, CODEBASE // Get information addr Section code

```
mov codebase, $ RESULT // Insert (this time is 40,100)
```

```
gmi eip, CODESIZE // Get the information section of the code size
```

```
mov codesize, $ RESULT // Insert (this time is 6F000)
```


bpwm codebase, codesize // Set a breakpoint in the section on write code

* /

_Viet Long lines like, but if we do it in Olly only 1 second is complete, OK, if at OllyDBG after you

set breakpoint is hit Shift + F9. In here you can order ESTO (Executes Shift-F9 in OllyDbg) . Now we

add the command:

/ *

ESTO

* /

_Khi Press Shift + F9 we will come:

00541381	F3:A4	REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:
00541383	C685 412B1B07 5	MOV BYTE PTR SS:[EBP+71B2B41],56
0054138A	68 396D1FD4	PUSH 041F6D89
0054138C	FFD5 FF001007	PUSH 04000000 PTR DS:[EBP+31B2B41]

_De Amenities track we put a label (Label) is a REP:

/ *

REP:

* /

Olly _Trong come if we press Shift + F9 2 times more in the script will add 2 ESTO:

/ *

ESTO

ESTO

* /

_Toi Here we will press F7 and F8, but if it does not stop in the form

/ *

REP MOVS BYTE PTR ES: [EDI], BYTE PTR DS: [ESI]

* /

_Khi That we will be entangled the endless loop, so we need to create a loop to avoid entangled loop.

Listen to the headaches. What started, simply stop, we will use OllyScript to create a loop to check the

cursor by Olly is in line with REP when we press Shift + F9 3 times not, if that is not repeated to the

point that the time. To create a loop in the language we use OllyScript structure monitoring a function

Cmp (compare, type True / False). The resolution as follows:

/ *

find eip, # F3A4 ????# // check if there are standing in line REP (opcode F3A4)

Cmp \$ result, 0 // If you do not see this opcode the implementation of orders

REP je // Jump to label REP to perform the command ESTO

* /

_Neu That we will continue to implement the command press F7, F8

/ *

STI // step into

STO // Step over

* /

_Tiep By, we will press Shift + F9 should add one more command ESTO nhé:

/ ESTO * * /

_Neu In OllyDBG hours this code by hand to you here:

00548938	8908	MOV DWORD PTR DS:[EAX],ECX	WS2_32. __WSAFDIsSet
0054893D	AD	LODS DWORD PTR DS:[ESI]	
0054893E	C746 FC 00000000	MOV DWORD PTR DS:[ESI-4],0	
00548945	89B5 492F1B07	MOV DWORD PTR SS:[EBP+71B2F49],ESI	

_Khuc This is quite important to pass by IAT elimination so we will put to a loop to make sure that

we're standing here (opcode 8908AD). Write as follows:

/ *

Move:

find eip, # 8908AD? # / / check is in line with opcode 8908AD

Cmp \$ result, 0 // If you do not see this opcode the implementation of orders

je move // Jump to label move to implement the order if the search result opcode = 0, if not

jmp MAINTASK // Jump to label MAINTASK to work: P!

* /

_Luu That is the label we need from the report, but the variable is declared. Now, if done by hand, we

will find the command have jumped opcode: 0F850A000000C785 and patch it into JMP. The script is

written as follows:

/*

MAINTASK:

mov add, eip // get the current position of the cursor on the window CPU

findmem 0F850A000000C785 # # // Search opcode

mov addr1, \$ RESULT // Take the address to a temporary variable

mov [addr1], 0A0EEB // Patch 3 bytes in temporary changes in value 0A0EEB

*/

_Tuong Self we find in the command chain JE 4 JE order to cancel IAT: P:

0054821F	83BD 6D1F1B07 0	CMP DWORD PTR SS:[EBP+71B1F6D],1
00548226	0F84 39000000	JE Unpackme.00548265
0054822C	3B8D D91A1B07	CMP ECX,DWORD PTR SS:[EBP+71B1AD9]
00548232	0F84 2D000000	JE Unpackme.00548265
00548238	3B8D A5231B07	CMP ECX,DWORD PTR SS:[EBP+71B23A5]
0054823E	0F84 21000000	JE Unpackme.00548265
00548244	3B8D A5261B07	CMP ECX,DWORD PTR SS:[EBP+71B26A5]
0054824A	0F84 15000000	JE Unpackme.00548265
00548250	8D9D 9B3C2207	LEA EBX,DWORD PTR SS:[EBP+7223C9B]
00548257	FFD0	CALL EBX

/ *

findmem 0F84390000003B8D # #

mov addr2, \$ RESULT

mov [addr2], 3928EB

* /

_Chung We need to get the address of the command to addr1 chien: D, so get the address of it:

/ *

mov tmpbp, addr1

* /

_Gio If used to Manage Memory alloc a cave, the code we use:

/ *

alloc 1000 // Allocate new memory page, you can read / write and execute.

* /

_Gio Read address area code, to a variable, so the set is soft, we use the command:

```
/*
```

```
mov mem, $ RESULT // read the address area code (example: 2660000)
```

```
log mem // write the log
```

```
*/
```

_Gio We will write a code of how to train, and edit it, using
add two more variables are tmp and memtmp.

```
/*
```

```
mov tmp, mem // transfer the address area code changes just to alloc tmp
```

```
mov [tmp],
```

```
# A3000000008908ADC746FC00000000E90000000050A1000000008907807FFFE8750866C747FEFF15EB0666  
C747FEFF2558E90000000050A100000000894701807FFFE8750866C747FFFF15EB0666C747FFFF25580F8500  
000000E90000000083C704E900000000 # // Write the code in case of Chinatown area code to create =)).
```

```
mov memtmp, tmp // Take value area code just to write variables memtmp
```

```
*/
```

_Luc In this window OllyDBG at 2660000 will like this:

02660000	A3 00000000	MOV DWORD PTR DS:[0],EAX
02660005	8908	MOV DWORD PTR DS:[EAX],ECX
02660007	AD	LODS DWORD PTR DS:[ESI]
02660008	C746 FC 00000000	MOV DWORD PTR DS:[ESI-4],0
0266000F	✓E9 00000000	JMP 02660014
02660014	50	PUSH EAX
02660015	A1 00000000	MOV EAX,DWORD PTR DS:[0]
0266001A	8907	MOV DWORD PTR DS:[EDI],EAX
0266001C	807F FF E8	CMP BYTE PTR DS:[EDI-1],0E8
02660020	✓75 08	JNZ SHORT 0266002A
02660022	66:C747 FE FF15	MOV WORD PTR DS:[EDI-2],15FF
02660028	✓EB 06	JMP SHORT 02660030
0266002A	66:C747 FE FF25	MOV WORD PTR DS:[EDI-2],25FF
02660030	58	POP EAX
02660031	✓E9 00000000	JMP 02660036
02660036	50	PUSH EAX
02660037	A1 00000000	MOV EAX,DWORD PTR DS:[0]
0266003C	8947 01	MOV DWORD PTR DS:[EDI+1],EAX
0266003F	807F FF E8	CMP BYTE PTR DS:[EDI-1],0E8
02660043	✓75 08	JNZ SHORT 0266004D
02660045	66:C747 FF FF15	MOV WORD PTR DS:[EDI-1],15FF
02660048	✓EB 06	JMP SHORT 02660053
0266004D	66:C747 FF FF25	MOV WORD PTR DS:[EDI-1],25FF
02660053	58	POP EAX
02660054	✓0F85 00000000	JNZ 0266005A
0266005A	✓E9 00000000	JMP 0266005F
0266005F	83C7 04	ADD EDI,4
02660062	✓E9 00000000	JMP 02660067
02660067	0000	ADD BYTE PTR DS:[EAX],AL

_Gio We need to patch the value of the variable memtmp from 2660000 to 2660100 (note the value to address other nhé):

```

/ *

add memtmp, 100 // added value memtmp 100

add tmp, 1 // patch in the first byte News at 2660001

* /

```

_Khi It will OllyDBG out this:



02660000	A3 00016602	MOV DWORD PTR DS:[2660100],EAX
0266000E	8B40	MOV EAX, [EAX+00000040]

_Tiep Community we continue to value at +15 tmp is 100 (ie address 2660000 +15 = 2660015):

```

/ *

mov [tmp], memtmp // Bringing value to the variable tmp 2660100

add tmp, 15 // patch to address 2660000 +15 value

* /

```


02660014	50	FUSH EAX
02660015	A1 00016602	MOV EAX,DWORD PTR DS:[2660100]
02660016	0000	MOV EAX,DWORD PTR DS:[2660100]

_Tuong Self we continue to do:

/ *

mov [tmp], memtmp

add tmp, 22

* /

02660037	A1 00016602	MOV EAX,DWORD PTR DS:[2660100]
02660038	0017 01	MOV EAX,DWORD PTR DS:[2660100]

_Sau That we get the address to mem tmp

/ *

mov tmp, mem // Bring to address tmp 2660000

* /

_Ban Also remember remember tmpbp variables in the air, then its value is 548E07 not change (because nay xài first time that has changed). OK time to get it:

/ *

Find tmpbp, 8908AD # #

mov tmpbp, \$ RESULT // Insert value of \$ RESULT (54893B is now) to turn tmpbp

* /

0054893B	8908	MOV DWORD PTR DS:[EAX],ECX
0054893D	AD	LODS DWORD PTR DS:[ESI]
0054893F	834C EC 00000000	MOVL DWORD PTR DS:[ECI],41 0

_Toi The patch orders jmp ago, the first patch to order 54893B jmp 2660000:

/ *

EVAL

Evaluates a string expression that contains variables.

The variables that are declared in the current script can be enclosed in curly braces () to be Inserted.

Sets the reserved \$ RESULT variable

Example:

```
var x
```

```
mov x, 1000
```

```
eval "The value of x is (x)" // After this $ Result is the value of x is 1000 "
```

* /

```
/*
```

```
mov addr1, tmpbp // Bring to address 54893B variable addr1
```

```
add addr1, 0A // eval command a blank area to enter into the expression, we should leave 10, enough to command jmp
```

```
eval "jmp (tmp)" // Get string jmp (2660000)
```

```
asm tmpbp, $ RESULT // same as you type the space bar, and so the field of
```

```
*/
```



_Tuong Our own patch in turn:

```
/*
```

```
Find tmpbp, E92400000058 # #
```

```
mov tmpbp, $ RESULT
```

```
add tmp, 14
```

```
eval "jmp (tmp)"
```

```
asm tmpbp, $ RESULT
```

```
*/
```

005489F1	AA	STOS BYTE PTR ES:[EDI]
005489F2	E9 10761102	JMP 02660014
005489F7	58	POP EAX
005489FC	nn	STOS BYTE PTR ES:[EDI]

_Khuc The same, you see the small, just get the value, and then patch it. Chang Y as on, if I have from the analysis, sure some dozen pages long. L !

Line	Command	Result	EIP	Values <---
74	find tmpbp, #0F851800000083BD#	5489FD		5489F2
75	mov tmpbp, \$RESULT			5489FD
76	mov addr3, tmpbp			5489FD
77	add addr3, 06			5489FD
78	add tmp, 22			2660014
79	eval "jmp {tmp}"	"jmp 2660036"		2660036
80	asm tmpbp, \$RESULT	5		"jmp 2660036" 5489FD
81	find tmpbp, #884704#	548A18		5489FD
82	mov tmpbp, \$RESULT			548A18
83	mov addr2, tmpbp			548A18
84	add addr2, 03			548A18
85	mov [tmpbp], #909090#			548A18
86	find tmpbp, #ABAD#	548A26		548A18
87	mov tmpbp, \$RESULT			548A26
88	mov [tmpbp], #90#			548A26
89	add tmpbp, 9			548A26
90	add tmp, 29			2660036
91	eval "jmp {tmp}"	"jmp 266005F"		266005F
92	asm tmpbp, \$RESULT	5		"jmp 266005F" 548A2F
93	mov mentmp, mem			2660000
94	add mentmp, 0F			2660000
95	eval "jmp {addr1}"	"jmp 548945"		548945
96	asm mentmp, \$RESULT	5		"jmp 548945" 266000F
97	add mentmp, 22			266000F
98	eval "jmp {addr2}"	"jmp 548A1B"		548A1B
99	asm mentmp, \$RESULT	5		"jmp 548A1B" 2660031
100	add mentmp, 23			2660031
101	eval "jne {addr2}"	"jne 548A1B"		548A1B
102	asm mentmp, \$RESULT	6		"jne 548A1B" 2660054
103	add mentmp, 06			2660054
104	eval "jmp {addr3}"	"jmp 548A03"		548A03
105	asm mentmp, \$RESULT	5		"jmp 548A03" 266005A
106	add mentmp, 08			266005A
107	eval "jmp {addr1}"	"jmp 548945"		548945
108	asm mentmp, \$RESULT	5		"jmp 548945" 2660062

_Khi Patch has finished and all, as usual, we typed in OllyDBG Ctrl + G: E9A4060000, here we use findeip:

/ *


```
find eip, E9A4060000 # # // Find the address contains opcode
```

```
mov tmpbp, $ RESULT // switch to temporary variables
```

```
bphws tmpbp, "x" // Set a breakpoint on execution
```

```
esto // Shift + F9
```

```
* /
```

_Sau That we need to set a breakpoint in the section on access code and press Shift + F9:

```
/ *
```

```
bphwc tmpbp // Remove Hardware Breakoint
```

```
mov tmp, codebase // add read the section of code
```

```
add tmp, codesize // transfer size section of code to change tmp
```

OEP:

```
bprm codebase, codesize // Set breakpoint
```

```
esto // Shift + F9
```

```
bpmc // remove breakpoint
```

```
Cmp eip, tmp
```

```
ja OEP
```

```
eval "dumped.exe" // Filename dump
```

```
dpe $ result, eip // dump file
```

```
msg "script finished, check the oep ~ place by yourself. If OEP OK, fire up and fix IAT ImpREC"
```

```
ret
```

```
* /
```

_Va Script is complete:

```
/* Edit this script from the script of odoko for 1.9.xx ver */
```

```
var modulebase
```

```
var codebase
```

var codesize

var tmpbp

var apibase

var mem

var tmp

BPHWCALL

gmi eip, MODULEBASE

mov modulebase, \$ RESULT

gmi eip, CODEBASE

mov codebase, \$ RESULT

gmi eip, CODESIZE

mov codesize, \$ RESULT

bpwm codebase, codesize

ESTO

REP:

ESTO

ESTO

find eip, # F3A4 ????

Cmp \$ result, 0

je REP

STI

STO

ESTO

Move:

find eip, # 8908AD? #

Cmp \$ result, 0

je move

jmp MAINTASK

MAINTASK:

bpmc

mov add, eip

findmem 0F850A000000C785 # #

mov add1, \$ RESULT

mov [add1], 0A0EEB

findmem 0F84390000003B8D # #

mov add2, \$ RESULT

mov [add2], 3928EB

mov tmpbp, add1

alloc 1000

mov mem, \$ RESULT

log mem

mov tmp, mem

mov

[tmp], # A3000000008908ADC746FC00000000E90000000050A1000000008907807FFFE8750866C74

7FEFF15EB0666C747FEFF2558E90000000050A100000000894701807FFFE8750866C747FFFF15E

B0666C747FFFF25580F8500000000E90000000083C704E900000000 #

mov memtmp, tmp

add memtmp, 100

add tmp, 1

mov [tmp], memtmp

add tmp, 15

mov [tmp], memtmp

add tmp, 22

mov [tmp], memtmp

mov tmp, mem

Find tmpbp, 8908AD # #

mov tmpbp, \$ RESULT

mov addr1, tmpbp

add addr1, 0A

eval "jmp (tmp)"

asm tmpbp, \$ RESULT

Find tmpbp, E92400000058 # #

mov tmpbp, \$ RESULT

add tmp, 14

```
eval "jmp (tmp)"
```

```
asm tmpbp, $ RESULT
```

```
Find tmpbp, 0F851800000083BD # #
```



```
mov tmpbp, $ RESULT
```

```
mov addr3, tmpbp
```

```
add addr3, 06
```

add tmp, 22

eval "jmp (tmp)"

asm tmpbp, \$ RESULT

Find tmpbp, # 884704 #

mov tmpbp, \$ RESULT

mov addr2, tmpbp

add addr2, 03

mov [tmpbp], # 909090 #

Find tmpbp, ABAD # #


```
mov tmpbp, $ RESULT
```

```
mov [tmpbp], # 90 #
```

```
add tmpbp, 9
```

add tmp, 29

eval "jmp (tmp)"

asm tmpbp, \$ RESULT

mov memtmp, mem

add memtmp, 0F

eval "jmp (addr1)"

asm memtmp, \$ RESULT

add memtmp, 22

eval "jmp (addr2)"

asm memtmp, \$ RESULT

add memtmp, 23

eval "jne) (addr2"

asm memtmp, \$ RESULT

add memtmp, 06

eval "jmp addr3 ()"

asm memtmp, \$ RESULT

add memtmp, 08

eval "jmp (addr1)"

asm memtmp, \$ RESULT

find eip, E9A4060000 # #

mov tmpbp, \$ RESULT

add tmpbp, 14

bphws tmpbp, "x"

esto

bphwc tmpbp

mov tmp, codebase

add tmp, codesize

oep:

bprm codebase, codesize

esto

bpmc

Cmp eip, tmp

ja oep


```
eval "dumped.exe"
```

```
dpe $ result, eip
```

```
msg "script finished, check the oep ~ place by yourself. If OEP OK, fire up and fix IAT ImpREC"
```

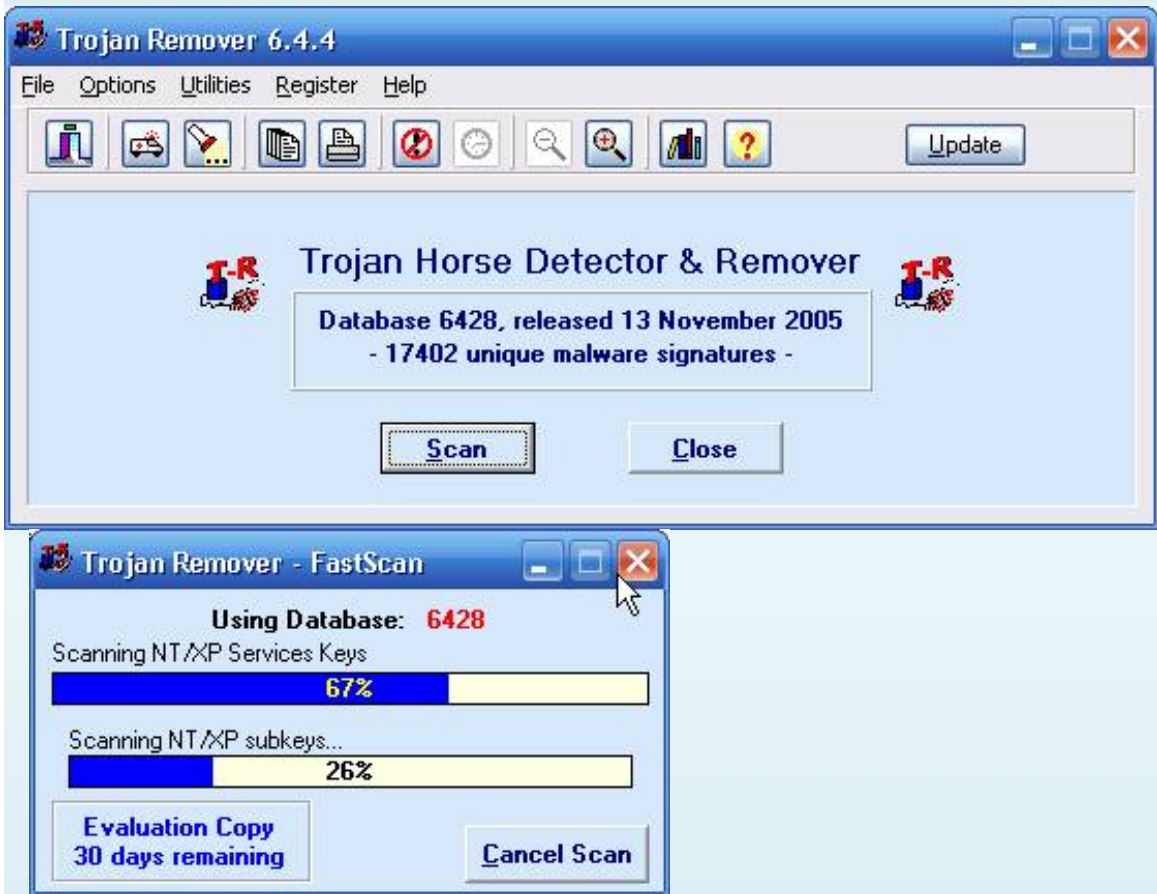
ret

_Co Can say the script is extremely simple and basic for anyone new to familiarize with Themida, as going into Themida, you need to do more work than as self-created script fix stolen bytes,
Ôi referred to but tired. Stop that from going: P.

【作者声明】： 只是感兴趣，没有其他目的。失误之处敬请诸位大侠赐教

Trojan-remover DebugBlocker + Nanomites

Distribution: unregistered evaluation copy
Version: 6.4.4 - 30th October 2005
Copyright (c) 1999-2005 Simply Super Software - All Rights Reserved



_Xin Greeting, we met again in this tut to discuss a target by Protect Armadillo 4.x can rename the file when run, more unpack when running the process will create a process to prevent other with a notice as suspend making machines we can not fix Nanomites! Ok, let's goooooo!

FixDBG - {RmvTrjan*exe} - [UPC= main thread, module RmvTrjan]

File View Debug Plugins Options Window Help Custom CrackerTools Languages

Paused

Address	Hex dump	Disassembly	Comment	Registers
004A1D63	55	push ebp		EAX 00000000
004A1D64	8BEC	mov ebp,esp		ECX 0012FF00
004A1D66	6A FF	push -1		EDX 7C90E1
004A1D68	68 F8BE4C00	push RmvTrjan.004CBEF8		EBX 7FFDC000
004A1D6D	68 A01A4A00	push RmvTrjan.004A1AA0	SE handler installation	ESP 0012FF00
004A1D72	64:A1 00000000	mov eax,dword ptr fs:[0]		EBP 0012FF00
004A1D78	50	push eax		ESI 00000000
004A1D79	64:8925 00000000	mov dword ptr fs:[0],esp		EDI 0012D000
004A1D80	83EC 58	sub esp,58		EIP 004A1D80
004A1D83	53	push ebx		C 0 ES 00
004A1D84	56	push esi		P 1 CS 00
004A1D85	57	push edi		A 0 SS 00
004A1D86	8965 E8	mov dword ptr ss:[ebp-18],esp		Z 1 DS 00
004A1D89	FF15 88614C00	call dword ptr ds:[<4KERNEL32.GetVersion	kernel32.GetVersion	S 0 FS 00
004A1D8F	33D2	xor edx,edx	ntdll.KiFastSystemCall	T 0 GS 00
004A1D91	8AD4	mov dl,ah	ntdll.KiFastSystemCall	D 0
004A1D93	8915 7CD54C00	mov dword ptr ds:[4CD57C],edx		O 0 Last
004A1D99	8BC8	mov ecx,eax		EFL 00000000
004A1D9B	81E1 FF000000	and ecx,0FF		ST0 empty

ebp=0012FFFF0

Address	Value	Comment	Address	Value	Comment
004C6000	77F16E98	GDI32.DeleteDC	0012FFC4	7C816D4F	RETURN to
004C6004	77F1ED09	GDI32.RealizePalette	0012FFC8	0012D184	
004C6008	77F1834F	GDI32.SelectPalette	0012FFCC	00000020	
004C600C	77F1B251	GDI32.CreateDCA	0012FFD0	7FFDC000	

Command: bp WriteProcessMemory

Program entry point

By DebugBlocker pass:

[UPC= main thread, module RmvTrjan]

File View Debug Plugins Options Window Help Custom CrackerTools Languages

Paused

Address	Hex dump	Disassembly	Comment	Registers (FPU)
00491F5F	50	push eax		EAX 00000001
00491F60	E8 2C843C7C	call kernel32.DebugActiveProcessSt		ECX 0012BCA8
00491F65	90	nop		EDX 7C90EB94 n
00491F66	? 008B 85FCFDFF	add byte ptr ds:[ebx+FFFD8C85],cl		EBX 7FFD8000
00491F6C	? FF25 FF000000	jmp dword ptr ds:[FF]		ESP 0012BCC8
00491F72	. 85C0	test eax,eax		EBP 0012D7B0
00491F74	. 74 13	je short RmvTrjan.00491F89		ESI 00002710
00491F76	. 8B0D 1CD34C00	mov ecx,dword ptr ds:[4CD31C]		EDI 0012C31C
00491F7C	. 8379 20 00	cmp dword ptr ds:[ecx+20],0		EIP 00491F66 R
00491F80	. 74 07	je short RmvTrjan.00491F89		C 0 ES 0023 3
00491F82	. C685 FCFDFFFF	mov byte ptr ss:[ebp-204],0		P 0 CS 001B 3
00491F89	> 68 10D24C00	push RmvTrjan.004CD210	pCriticalSection = Rmv	A 0 SS 0023 3
00491F8E	. FF15 A4614C00	call dword ptr ds:[<4KERNEL32.Ent	EnterCriticalSection	Z 0 DS 0023 3
00491F94	. 60	pushad		S 0 FS 003B 3
00491F95	. 33C0	xor eax,eax		T 0 GS 0000 N
00491F97	. 75 02	jnz short RmvTrjan.00491F9E		D 0
00491F99	. EB 15	jmp short RmvTrjan.00491FB0		O 0 LastErr E
00491F9B	> EB 33	jmp short RmvTrjan.00491FD0		EFL 00000202 (
00491F9D	C0	db C0		ST0 empty -UNO

eax=00000001

Address	Value	Comment	Address	Value	Comment
004C6000	77F16E98	GDI32.DeleteDC	0012BCC8	0012FF2C	
004C6004	77F1ED09	GDI32.RealizePalette	0012BCCC	00000000	
004C6008	77F1834F	GDI32.SelectPalette	0012BCD0	7FFD8000	
004C600C	77F1B251	GDI32.CreateDCA	0012BCD4	00000000	

Command:

_Attach, F9, F12:

[UPC= main thread, module RmvTrjan]

File View Debug Plugins Options Window Help Custom CrackerTools Languages

Paused

Address	Hex dump	Disassembly	Comment	Registers (FPU)
004A1D63	55	push ebp		EAX 00000000
004A1D64	8BEC	mov ebp,esp		ECX 0012FFB0
004A1D66	6A FF	push -1		EDX 7C90EB94 n
004A1D68	68 F8BE4C00	push RmvTrjan.004CBEF8		EBX 7FFDF000
004A1D6D	68 A01A4A00	push RmvTrjan.004A1AA0	SE handler installatio	ESP 0012FFC4
004A1D72	64:A1 00000000	mov eax,dword ptr fs:[0]		EBP 0012FFF0
004A1D78	50	push eax		ESI 7C9155C9 n
004A1D79	64:8925 00000000	mov dword ptr fs:[0],esp		EDI 0012B6E8
004A1D80	83EC 58	sub esp,58		EIP 004A1D63 R
004A1D83	53	push ebx		C 0 ES 0023 3
004A1D84	56	push esi	ntdll.7C9155C9	P 1 CS 001B 3
004A1D85	57	push edi		A 0 SS 0023 3
004A1D86	8965 E8	mov dword ptr ss:[ebp-18],esp		Z 1 DS 0023 3
004A1D89	FF15 88614C00	call dword ptr ds:[<4KERNEL32.GetV	kernel32.GetVersion	S 0 FS 003B 3
004A1D8F	33D2	xor edx,edx	ntdll.KiFastSystemCall	T 0 GS 0000 N
004A1D91	8AD4	mov dl,ah		D 0
004A1D93	8915 7CD54C00	mov dword ptr ds:[4CD57C],edx	ntdll.KiFastSystemCall	O 0 LastErr E
004A1D99	8BC8	mov ecx,eax		EFL 00000246 (
004A1D9B	81E1 FF000000	and ecx,0FF		ST0 empty 2.46

ebp=0012FFF0

Address	Value	Comment	Address	Value	Comment
004C6000	77F16E98	GDI32.DeleteDC	0012FFC4	7C816D4F	RETURN to ke:
004C6004	77F1ED09	GDI32.RealizePalette	0012FFC8	0012B6E8	
004C6008	77F1834F	GDI32.SelectPalette	0012FFCC	7C9155C9	RETURN to nt:
004C600C	77F1B251	GDI32.CreateDCA	0012FFD0	7FFDF000	

Command:

Thread 00000FE8 terminated, exit code 0

Fix MagicJump:

- [UPC= main thread]

File View Debug Plugins Options Window Help Custom CrackerTools Languages

Paused

Address Hex dump Disassembly Comment Registers (FPU)

00AF4FD8	8B0D AC0DB200	mov ecx,dword ptr ds:[B20DAC]		EAX 7C800000 k
00AF4FDE	89040E	mov dword ptr ds:[esi+ecx],eax	kernel32.7C800000	ECX 7C80E82B k
00AF4FE1	A1 AC0DB200	mov eax,dword ptr ds:[B20DAC]		EDX 7C97C0D8 n
00AF4FE6	391C06	cmp dword ptr ds:[esi+eax],ebx		EBX 00000000
00AF4FE9	75 16	jnz short 00AF5001		ESP 00127800
00AF4FEB	8D85 B4FEFFFF	lea eax,dword ptr ss:[ebp-14C]		EBP 00127A94
00AF4FF1	50	push eax	kernel32.7C800000	ESI 00000000
00AF4FF2	FF15 B432B100	call dword ptr ds:[B132B4]	kernel32.LoadLibraryA	EDI 00B1809C
00AF4FF8	8B0D AC0DB200	mov ecx,dword ptr ds:[B20DAC]		EIP 00AF4FD8
00AF4FFE	89040E	mov dword ptr ds:[esi+ecx],eax	kernel32.7C800000	C 0 ES 0023 3
00AF5001	A1 AC0DB200	mov eax,dword ptr ds:[B20DAC]		P 1 CS 001B 3
00AF5006	391C06	cmp dword ptr ds:[esi+eax],ebx		A 0 SS 0023 3
00AF5009	E9 30010000	jmp 00AF513E	<- MagicJump fixed	Z 0 DS 0023 3
00AF500E	0033	add byte ptr ds:[ebx],dh		S 0 FS 003B 3
00AF5010	C9	leave		T 0 GS 0000 N
00AF5011	8B07	mov eax,dword ptr ds:[edi]		D 0
00AF5013	3918	cmp dword ptr ds:[eax],ebx		O 0 LastErr E
00AF5015	74 06	je short 00AF501D		EFL 00200206 (
00AF5017	41	inc ecx	kernel32.7C80E82B	ST0 empty 2.46

ds:[00B20DAC]=00C23FF0
ecx=7C80E82B (kernel32.7C80E82B)

Address	Value	Comment	Address	Value	Comment
004C6000	77F16E98	GDI32.DeleteDC	00127800	0012CC64	
004C6004	77F1ED09	GDI32.RealizePalette	00127804	D8C96153	
004C6008	77F1834F	GDI32.SelectPalette	00127808	00000000	
004C600C	77F1B251	GDI32.CreateDCA	0012780C	00B1809C	

Command:

_Breakpoint CreateThread, Ctrl + F9, F8 2 times:

- [UPC= main thread]

File View Debug Plugins Options Window Help Custom CrackerTools Languages

Paused

Address	Hex dump	Disassembly	Comment	Registers (FPU)
00B0D88B	2BCA	sub ecx,edx	ntdll.KiFastSystemCallP	EAX 000006B8
00B0D88D	FFD1	call ecx	kernel32.7C8107FD	ECX 7C8107FD k
00B0D88F	EB 1A	jmp short 00B0D8AB		EDX 7C90EB94 n
00B0D891	83FF 01	cmp edi,1		EBX FFFFFFFF
00B0D894	75 17	jnz short 00B0D8AD		ESP 0012D7A4
00B0D896	FF76 04	push dword ptr ds:[esi+4]		EBP 0012DF14
00B0D899	FF76 08	push dword ptr ds:[esi+8]		ESI 004CC808 R
00B0D89C	6A 00	push 0		EDI 0012FF2C
00B0D89E	52	push edx	ntdll.KiFastSystemCallP	EIP 00B0D828
00B0D89F	8B50 30	mov edx,dword ptr ds:[eax+30]		C 0 ES 0023 3
00B0D8A2	3350 08	xor edx,dword ptr ds:[eax+8]		P 1 CS 001B 3
00B0D8A5	3310	xor edx,dword ptr ds:[eax]		A 0 SS 0023 3
00B0D8A7	2BCA	sub ecx,edx	ntdll.KiFastSystemCallP	Z 1 DS 0023 3
00B0D8A9	FFD1	call ecx	kernel32.7C8107FD	S 0 FS 003B 3
00B0D8AB	8BD8	mov ebx,eax		T 0 GS 0000 N
00B0D8AD	5F	pop edi		D 0
00B0D8AE	8BC3	mov eax,ebx		O 0 LastErr E
00B0D8B0	5E	pop esi	RmvTrjan.004CC808	EFL 00200246 (
00B0D8B1	5B	pop ebx		ST0 empty 2.46

ecx=7C8107FD (kernel32.7C8107FD)

Address	Value	Comment	Address	Value	Comment
004C6000	77F16E98	GDI32.DeleteDC	0012D7A4	00000000	
004C6004	77F1ED09	GDI32.RealizePalette	0012D7A8	0012FF2C	
004C6008	77F1834F	GDI32.SelectPalette	0012D7AC	00000000	
004C600C	77F1B251	GDI32.CreateDCA	0012D7B0	7FFDF000	

Command: bp CreateThread BP address, string -- Break with condition

_F2, F9: J OEP

[UPC= main thread, module RmvTrjan]

File View Debug Plugins Options Window Help Custom CrackerTools Languages

Paused

Address	Hex dump	Disassembly	Comment	Registers (FPU)
0045692C	55	push ebp	OEP	EAX 004C6370 R
0045692D	8BEC	mov ebp,esp		ECX 0045692C R
0045692F	83C4 F0	add esp,-10		EDX 4AB94DEC
00456932	B8 4C674500	mov eax,RmvTrjan.0045674C		EBX FFFFFFFF
00456937	E8 60FAFAFF	call RmvTrjan.0040639C		ESP 0012D794
0045693C	A1 58804500	mov eax,dword ptr ds:[458058]		EBP 0012DF14
00456941	8B00	mov eax,dword ptr ds:[eax]		ESI 004CC808 R
00456943	E8 6CBOFFFF	call RmvTrjan.004519B4		EDI 00000001
00456948	A1 58804500	mov eax,dword ptr ds:[458058]		EIP 0045692C R
0045694D	8B00	mov eax,dword ptr ds:[eax]		C 0 ES 0023 3
0045694F	33D2	xor edx,edx		P 0 CS 001B 3
00456951	E8 6EACFFFF	call RmvTrjan.004515C4		A 1 SS 0023 3
00456956	8B0D 947E4500	mov ecx,dword ptr ds:[457E94]	RmvTrjan.00459BFC	Z 0 DS 0023 3
0045695C	A1 58804500	mov eax,dword ptr ds:[458058]		S 0 FS 003B 3
00456961	8B00	mov eax,dword ptr ds:[eax]		T 0 GS 0000 N
00456963	8B15 642F4500	mov edx,dword ptr ds:[452F64]	RmvTrjan.00452FB0	D 0
00456969	E8 5EB0FFFF	call RmvTrjan.004519CC		O 0 LastErr E
0045696E	A1 58804500	mov eax,dword ptr ds:[458058]		EFL 00200212 (
00456973	8B00	mov eax,dword ptr ds:[eax]		ST0 empty 2.46

ebp=0012DF14

Address	Value	Comment	Address	Value	Comment
004C6000	77F16E98	GDI32.DeleteDC	0012D794	00B0D8AB	RETURN to 001
004C6004	77F1ED09	GDI32.RealizePalette	0012D798	00400000	ASCII "M2P"
004C6008	77F1834F	GDI32.SelectPalette	0012D79C	00000000	
004C600C	77F1B251	GDI32.CreateDCA	0012D7A0	00141F0E	

Command: bp CreateThread BP address, string -- Break with condition

_LordPE:

[LordPE Deluxe] by yoda

Path	PID	ImageBase	ImageSize
c:\program files\turbolaunch\turbolaunch.exe	00000A28	00400000	001A0000
j:\cracker\unpacker\armadillo\armadetach 1...	00000B7C	00400000	00008000
c:\program files\trojan remover\rmvtrjan.exe	00000E40	00400000	0014C000
c:\program files\trojan remover\rmvtrjan.exe	0		

Path: c:\program files\trojan remover\rmvtrjan.exe

Path: c:\windows\system32\ntdll.dll

Path: c:\windows\system32\kernel32.dll

Path: c:\windows\system32\user32.dll

Path: c:\windows\system32\gdi32.dll

dump full...

dump partial...

dump region...

active dump engine

priority

correct ImageSize

load into PE editor... (temp file)

load into PE editor... (read only)

PE Editor

Break & Enter

Rebuild PE

Unsplit

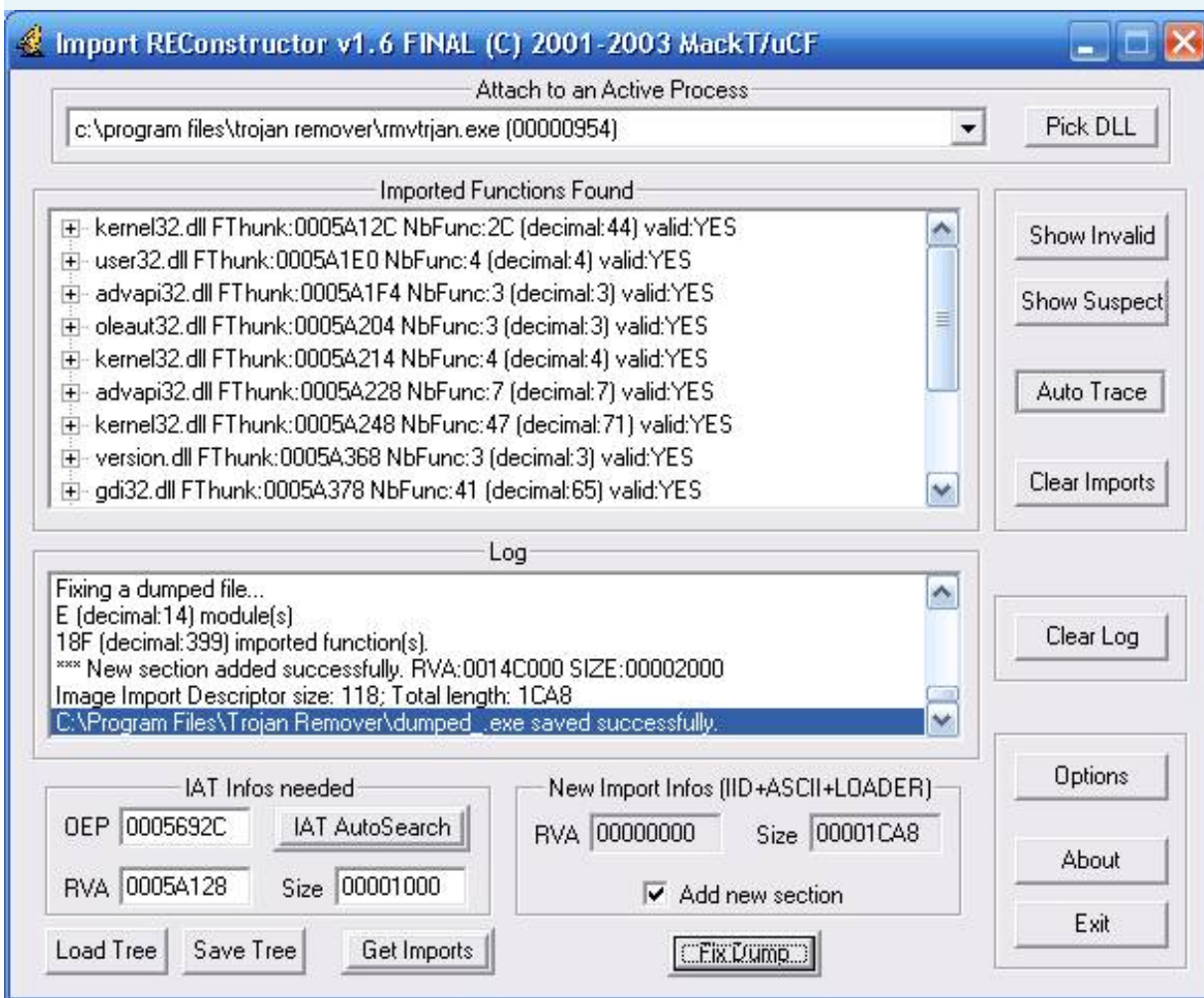
Dumper Server

Options

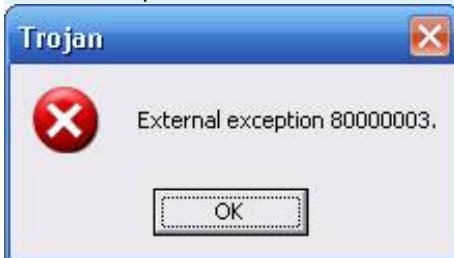
About

Exit

_ImpREC:



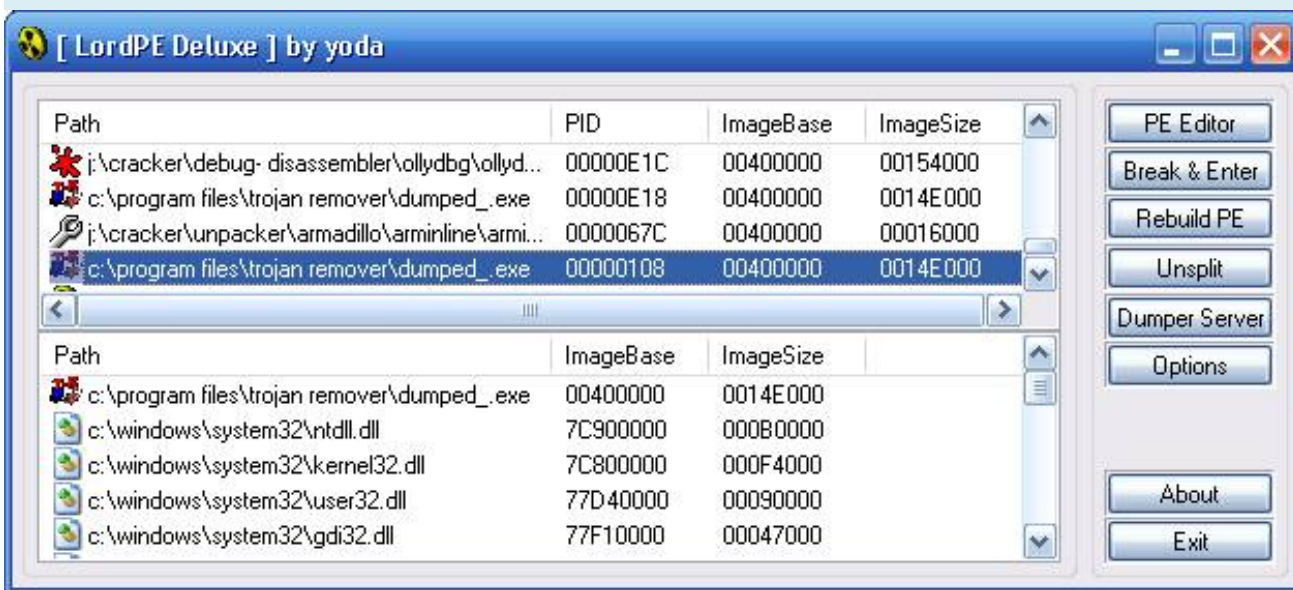
Run Dumped.exe:



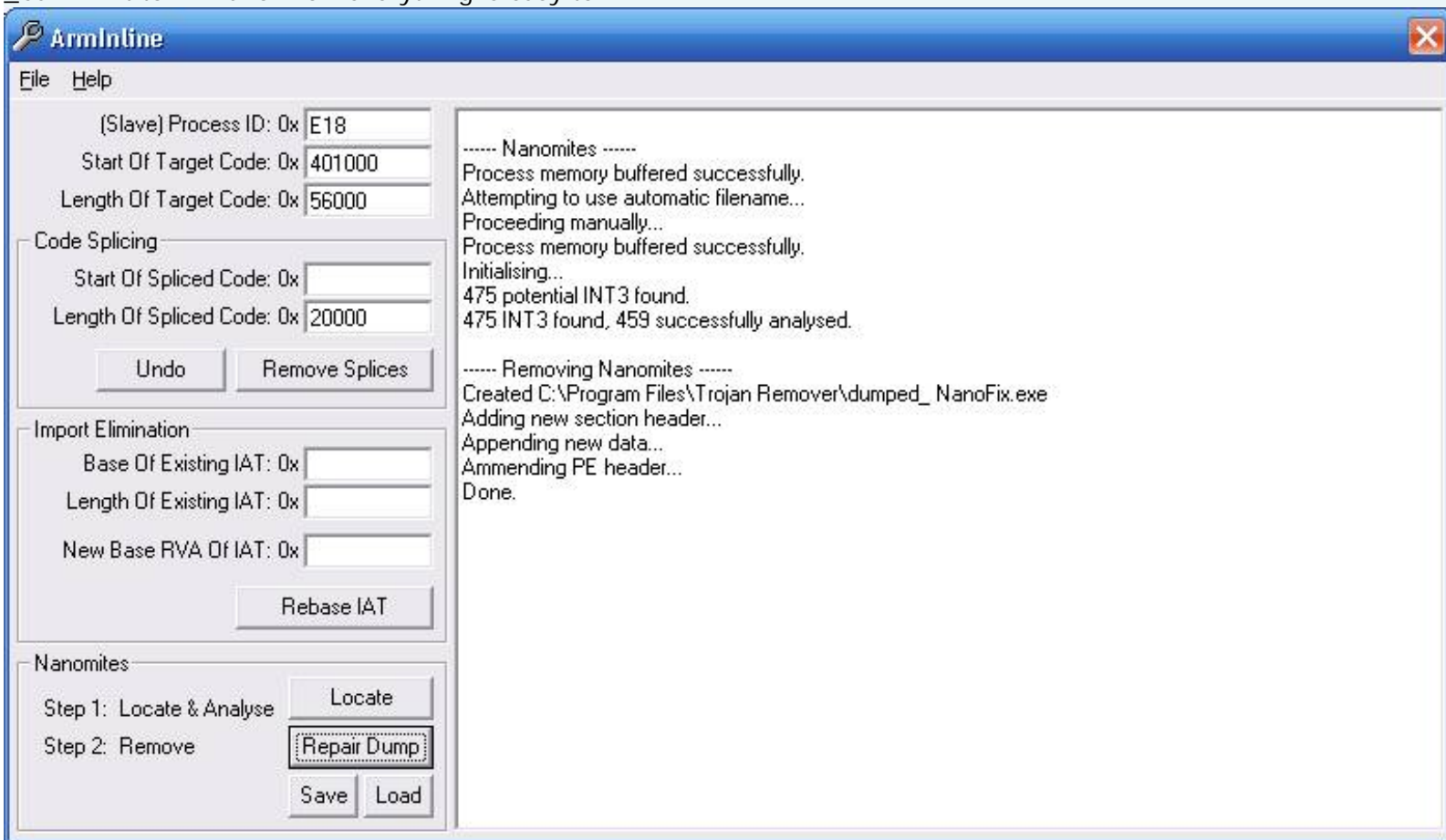
_Co Nanomites. Using ArmInline to fix the computer to suspend notice:



File Dumped.exe create a process debug format blocker to prevent it:



Can Kill it to fix Nano. Now everything is easy to:



Run:



_Bye!

Hacnho (<http://hacnho.exetools.com>)

Unpack manual PEDiminisher 0.1 -> Teraphy

Tool: - 0.93 PEID: detect packer.

- OllyDebugger (OllyDump plugin): Look for OEP, dump.

- ImpRec: FixIAT

I-Identify packer:



II-unpack:

1) Find OEP

Load program to Ollydebugger.

00405000	53	PUSH EBX	
00405001	51	PUSH ECX	
00405002	52	PUSH EDX	
00405003	56	PUSH ESI	
00405004	57	PUSH EDI	
00405005	55	PUSH EBP	
00405006	E8 00000000	CALL testfile.0040500B	
0040500B	5D	POP EBP	
0040500C	8BD5	MOV EDX,EBP	
0040500E	81ED A2304000	SUB EBP,testfile.004030A2	
00405014	2B95 91334000	SUB EDX,DWORD PTR SS:[EBP+403391]	
0040501A	81EA 0B000000	SUB EDX,0B	
00405020	8995 9A334000	MOV DWORD PTR SS:[EBP+40339A],EDX	
00405026	80BD 99334000	CMPI BYTE PTR SS:[EBP+403399],0	
0040502D	74 50	JE SHORT testfile.0040507F	
0040502F	E8 02010000	CALL testfile.00405136	
00405034	8BFD	MOV EDI,EBP	
00405036	8D9D 9A334000	LEA EBX,DWORD PTR SS:[EBP+40339A]	
0040503C	8B1B	MOV EBX,DWORD PTR DS:[EBX]	
0040503E	8D87 9E334000	LEA EAX,DWORD PTR DS:[EDI+40339E]	
00405044	8B00	MOV EAX,DWORD PTR DS:[EAX]	
00405046	03D8	ADD EBX,EAX	
00405048	8D8F A2334000	LEA ECX,DWORD PTR DS:[EDI+4033A2]	
0040504E	8B09	MOV ECX,DWORD PTR DS:[ECX]	
00405050	66:8B85 8F334000	MOV AX,WORD PTR SS:[EBP+40338F]	
00405057	8003 10	ADD BYTE PTR DS:[EBX],10	
00405059	3003	XOR BYTE PTR DS:[EBX],AL	
0040505C	3023	XOR BYTE PTR DS:[EBX],AH	
0040505E	8003 AA	ADD BYTE PTR DS:[EBX],0AA	
00405061	66:C1C0 03	ROL AX,3	
00405065	86E0	XCHG AL,AH	
00405067	43	INC EBX	
00405068	E2 ED	LOOPD SHORT testfile.00405057	
0040506A	E8 FF000000	CALL testfile.0040516E	
0040506F	83C7 08	ADD EDI,8	

Unpack routine

As seen above, will have 1 loop as the work is unpacking code into memory, after unpacking will complete

implementation.

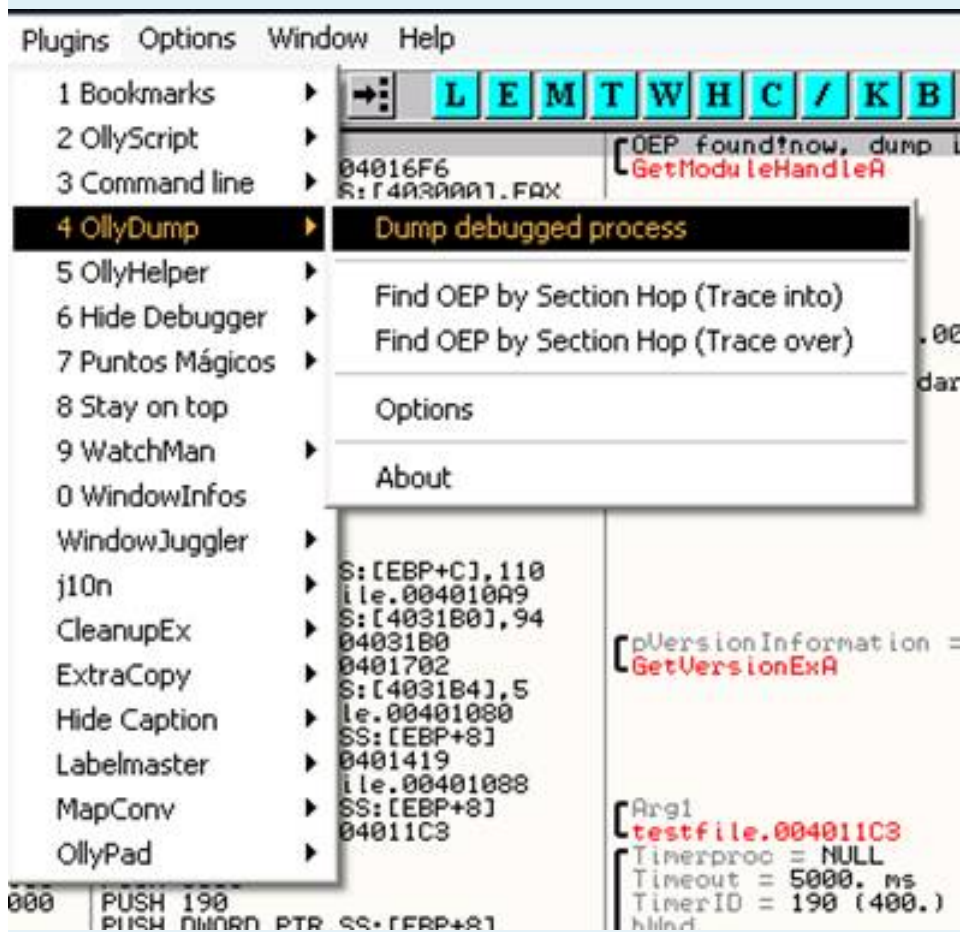
So let's trace through the loop and see this view:

00405044	8B00	MOV EAX,DWORD PTR DS:[EAX]	
00405046	0308	ADD EBX,EAX	
00405048	8D8F A2334000	LEA ECX,DWORD PTR DS:[EDI+4033A2]	
0040504E	8B09	MOV ECX,DWORD PTR DS:[ECX]	
00405050	66:8B85 8F334000	MOV AX,WORD PTR SS:[EBP+40338F]	
00405057	8003 10	ADD BYTE PTR DS:[EBX],10	
0040505A	3003	XOR BYTE PTR DS:[EBX],AL	
0040505C	3023	XOR BYTE PTR DS:[EBX],AH	
0040505E	8003 AA	ADD BYTE PTR DS:[EBX],0AA	
00405061	66:C1C0 03	ROL AX,3	
00405065	86E0	XCHG AL,AH	
00405067	43	INC EBX	
00405068	^E2 ED	LOOPD SHORT testfile.00405057	
0040506A	E8 FF000000	CALL testfile.0040516E	
0040506F	83C7 08	ADD EDI,8	
00405072	FE8D 99334000	DEC BYTE PTR SS:[EBP+403399]	
00405078	^75 BC	JNZ SHORT testfile.00405036	
0040507A	E8 16000000	CALL testfile.00405095	
0040507F	8B85 95334000	MOV EAX,DWORD PTR SS:[EBP+403395]	
00405085	8B9D 9A334000	MOV EBX,DWORD PTR SS:[EBP+40339A]	
00405088	03C3	ADD EAX,EBX	
0040508D	5D	POP EBP	
0040508E	5F	POP EDI	
0040508F	5E	POP ESI	
00405090	5A	POP EDX	
00405091	59	POP ECX	
00405092	58	POP EBX	
00405093	FFE0	JMP EAX	jump to OEP
00405095	8B95 9A334000	MOV EDX,DWORD PTR SS:[EBP+40339A]	
00405098	8B85 F6334000	MOV ESI,DWORD PTR SS:[EBP+4033F6]	
004050A1	33FF	XOR EDI,EDI	
004050A3	03F2	ADD ESI,EDX	
004050A5	03FA	ADD EDI,EDX	
004050A7	8B46 0C	MOV EAX,DWORD PTR DS:[ESI+C]	
004050AA	85C0	TEST EAX,EAX	
004050AC	^74 7F	JE SHORT testfile.0040512D	

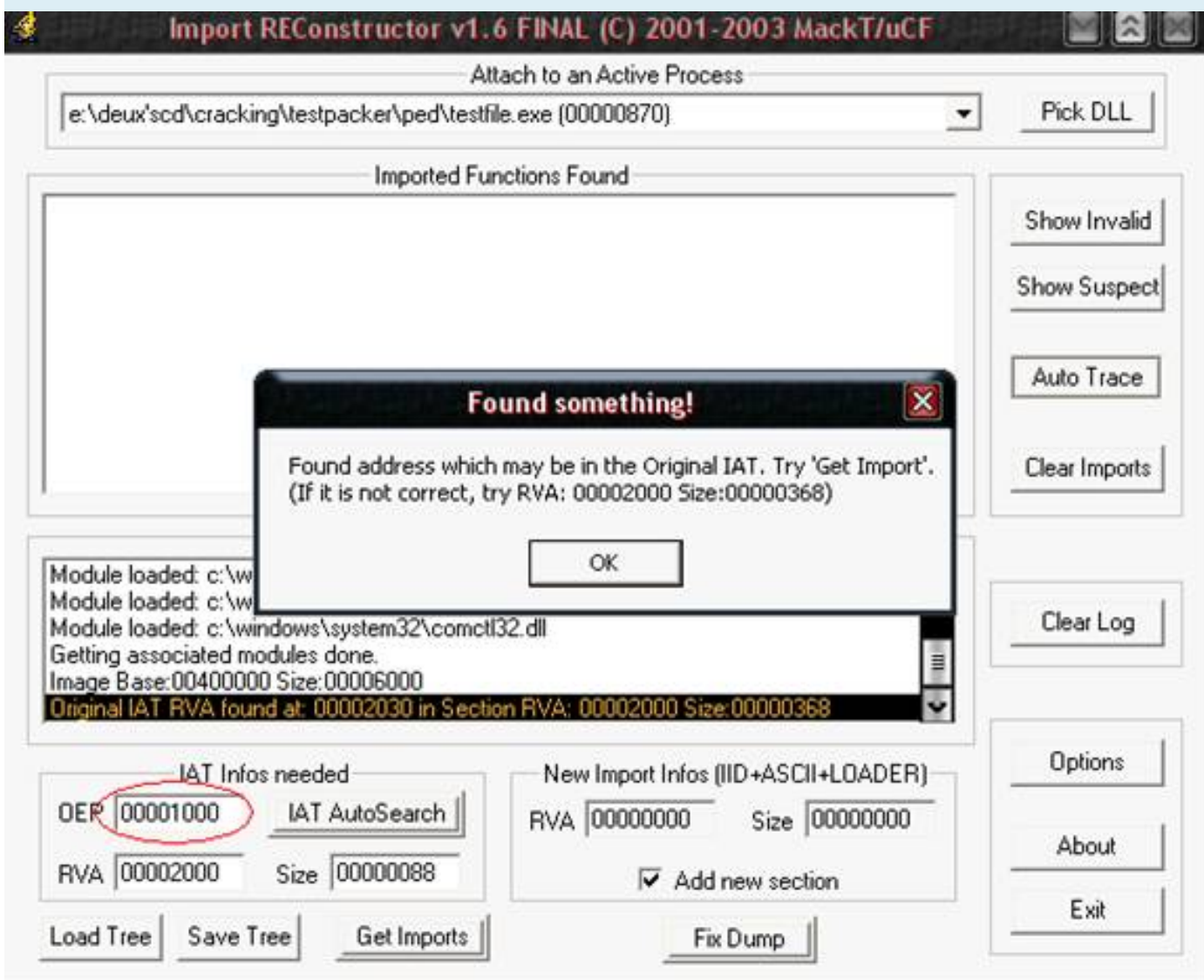
In 00405093, F8 to trace to the OEP.

00401000	. 6A 00	PUSH 0	[OEP found!now, dump it and fix IAT!]
00401002	. E8 EF060000	CALL testfile.004016F6	GetModuleHandleA
00401007	. A3 00304000	MOV DWORD PTR DS:[403000],EAX	
0040100C	. C705 44324000	MOV DWORD PTR DS:[403244],8	
00401016	. C705 48324000	MOV DWORD PTR DS:[403248],100	
00401020	. 68 44324000	PUSH testfile.00403244	
00401025	. E8 02070000	CALL testfile.0040172C	
0040102A	. 6A 00	PUSH 0	
0040102C	. 68 4A104000	PUSH testfile.0040104A	
00401031	. 6A 00	PUSH 0	
00401033	. 68 60304000	PUSH testfile.00403060	
00401038	. FF35 00304000	PUSH DWORD PTR DS:[403000]	
0040103E	. E8 41060000	CALL testfile.00401684	
00401043	. 6A 00	PUSH 0	
00401045	. E8 A6060000	CALL testfile.004016F0	
0040104A	. 55	PUSH EBP	
0040104B	. 8BEC	MOV EBP,ESP	
0040104D	. 83C4 C0	ADD ESP,-40	
00401050	. 817D 0C 100100	CMP DWORD PTR SS:[EBP+C],110	
00401057	^75 50	JNZ SHORT testfile.004010A9	
00401059	. C705 B0314000	MOV DWORD PTR DS:[4031B0],94	
00401063	. 68 B0314000	PUSH testfile.004031B0	
00401068	. E8 95060000	CALL testfile.00401702	
0040106D	. 833D B4314000	CMP DWORD PTR DS:[4031B4],5	
00401074	^72 0A	JB SHORT testfile.00401080	
00401076	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	
00401079	. E8 9B030000	CALL testfile.00401419	

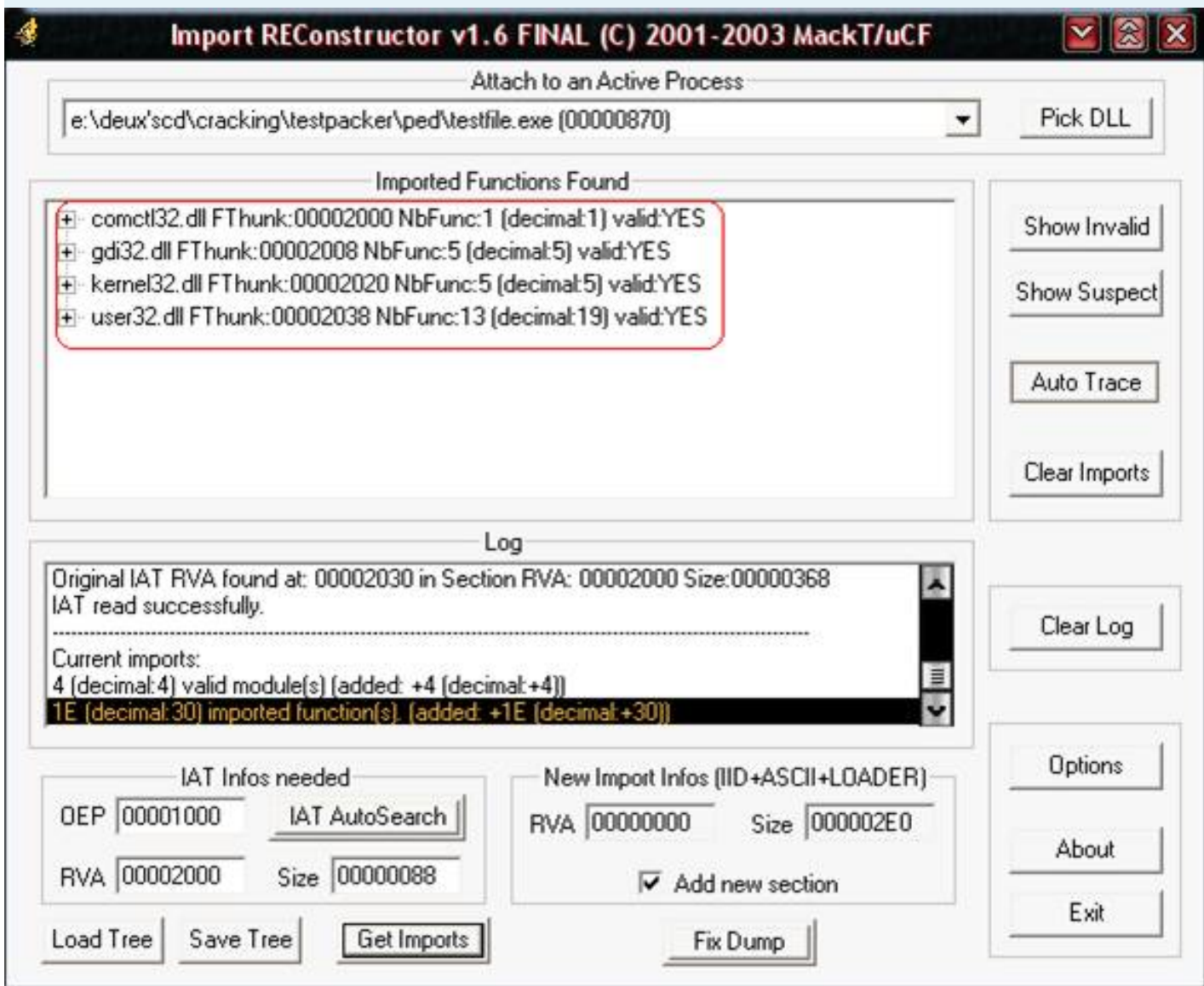
2) dump and IAT Fix:



After the dump and save the file, open ImpRec up, select the exe file is run:



Enter OEP found t h a t IAT à Auto Search Get Imports:



Now fix to fix IAT dump file for via dump is completed.



Conclusion:

Authors of the packer used Push Pop chant and write instead of using PushPad and Popad, but in general is simple. Can see similar Aspack.Sau unpack the code to use memory eax jump to jump to OEP

Tutorial by Deux.

Contact me: deux@surfy.net

Reverse Engineering Association: www.reaonline.net

EOF -----

-Ni hao ma (What is washed), today Tàn per referral to a Communication tut that Mo đắc quite interested. Chà là be heard to chiêu *CopyMEM II + Debugblocker* our unpack the Armadillo, but as first loss failed to have a chance to do. Tự stop gã **Takada** sent a soft *XP Tools* is it, should be six per tut 4 of the tools that **hacnho** training. Preparation has also not make more of the series wa **benina** film about **GetRight 5**. Finally also completed. But the time ni *CopyMEM II + Debugblocker* may perimeter of the wá that gòi per ko should focus on but it simply is thăng Crack XP Tools. This is also a soft per xuc relatively perfect from the first hour (missing code keygen only).

Try-wa of tools, including 23 soft Tool by it, like love to see Mo, which is the most appropriate function Delete information extant of a soft is not uninstalled completely. Wa tut this is mainly to review the vulnerabilities of the security program, then you try to text code for soft solidified home. Which, together make up wo keng leng leng keng ...

Unpack and Crack Full XP Tools version 4:58

<http://www.xptools.net/>

Protection: Armadillo with *CopyMEM II + Debugblocker*

Tools: OllyDBG Lord PE ImportREC CFF Explorer PUPE 2002 and OllyScript with Magic
Jump Finder (for Armadillo Detach from Client

I / unpack:

I/Defeat CopyMEM II

XP Tools



The Swiss army knife for your Windows

XP Tools

Version 4.58

For Windows 95/98/ME/NT/2000/XP/2003

Copyright(C) 2004 XPTools.net



You have 3 days left to try XP Tools.

[HomePage](#)[E-Mail to Us](#)[Register...](#)[Order](#)

The Options need for Olly tut just like then. Load soft to do:

```

Khd83 - xptools.exe - [CPU - main thread, module xptools]
File View Debug Plugins Options Window Help
[Icons] [L] [E] [M] [T] [W] [H] [C] [/] [K] [B] [R] [...] [S]
00EC0689 55 PUSH EBP
00EC068A 8BEC MOV EBP,ESP
00EC068C 6A FF PUSH -1
00EC068E 68 E0D7ED00 PUSH xptools.00EDD7E0
00EC0693 68 7000EC00 PUSH xptools.00EC0070
00EC0698 64:A1 000000 MOV EAX,DWORD PTR FS:[0]
00EC069E 50 PUSH EAX
00EC069F 64:8925 0000 MOV DWORD PTR FS:[0],ESP
00EC06A6 83EC 58 SUB ESP,58
00EC06A9 53 PUSH EBX
00EC06AA 56 PUSH ESI
00EC06AB 57 PUSH EDI
00EC06AC 8965 E8 MOV DWORD PTR SS:[EBP-18],ESP
00EC06AF FF15 4C81ED00 CALL DWORD PTR DS:[<&KERNEL32.GetVersion kernel32.GetVersion
00EC06B5 33D2 XOR EDX,EDX
00EC06B7 8A04 MOV DL,AH
00EC06B9 8915 90E1ED00 MOV DWORD PTR DS:[EDE190],EDX
00EC06BF 8BC8 MOV ECX,EAX
00EC06C1 81E1 FF000000 AND ECX,0FF
00EC06C7 8900 8CE1ED00 MOV DWORD PTR DS:[EDE18C],ECX
00EC06CD C1E1 08 SHL ECX,8
00EC06D0 03CA ADD ECX,EDX
00EC06D2 8900 88E1ED00 MOV DWORD PTR DS:[EDE188],ECX
00EC06D8 C1E8 10 SHR EAX,10
00EC06DB A3 84E1ED00 MOV DWORD PTR DS:[EDE184],EAX
00EC06E0 33F6 XOR ESI,ESI
00EC06E2 56 PUSH ESI
00EC06E3 E8 72160000 CALL xptools.00EC1D5A
00EC06E8 59 POP ECX
00EC06E9 5F POP EDI

```

-Type:

Command : bp WaitForDebugEvent

-F9, break:

```

77EAF13C  55          PUSH EBP
77EAF13D  8BEC        MOV EBP,ESP
77EAF13F  83EC 68     SUB ESP,68
77EAF142  56          PUSH ESI
77EAF143  FF75 0C     PUSH DWORD PTR SS:[EBP+C]
77EAF146  8D45 F8     LEA EAX,DWORD PTR SS:[EBP-8]
77EAF149  50          PUSH EAX

```

-NGO down, remember that ni:

```

0012DAEC  00EB0B44    CALL to WaitForDebugEvent
0012DAF0  0012EB9C    pDebugEvent = 0012EB9C
0012DAF4  000003E8    Timeout = 1000. ms

```

-Ctrl-F2 (Restart), to type:

Command : bp WriteProcessMemory

F9-1 play:

```

0012D7EC  00EB4E6E    CALL to WriteProcessMemory
0012D7F0  0000004C    hProcess = 0000004C
0012D7F4  00EC0689    Address = EC0689
0012D7F8  0012DADC    Buffer = 0012DADC
0012D7FC  00000002    BytesToWrite = 2
0012D800  0012DAE0    pBytesWritten = 0012DAE
0012D804  0012F054    UNICODE "e132.dll"

```

The F9-2:

```

0012D7EC  00EB4EBF    CALL to WriteProcessMemory
0012D7F0  0000004C    hProcess = 0000004C
0012D7F4  00EC0689    Address = EC0689
0012D7F8  00EDE000    Buffer = xptools.00EDE0
0012D7FC  00000002    BytesToWrite = 2
0012D800  0012DAE0    pBytesWritten = 0012DAE
0012D804  0012F054    UNICODE "e132.dll"

```

The F9-3:

```

0012D98C  00EB467F    CALL to WriteProcessMemory
0012D990  0000004C    hProcess = 0000004C
0012D994  0063D000    Address = 63D000
0012D998  016FCC00    Buffer = 016FCC00
0012D99C  00001000    BytesToWrite = 1000 (409
0012D9A0  0012D9A8    pBytesWritten = 0012D9A8

```

Signs 1000-Bytes target market is CopyMEM II. What the remaining seats Address and Buffer memory per home help.

-Ctrl-F9 to order the RETN.

```

77E61B24  C2 1400     RETN 14
77E61B27  BE 050000C0 MOV ESI,C0000005
77E61B2C  74 16       JMP SHORT kernel32.77E61B44
77E61B2E  33C0       XOR EAX,EAX
77E61B30  74 EE       JMP SHORT kernel32.77E61B20
77E61B32  8D4D 14     LEA ECX,DWORD PTR SS:[EBP+14]
77E61B35  51         PUSH ECX

```

Look under Xeo, drag it down a little time, to this:

```

0012DAA4 016FDC00
0012DAA8 00001000
0012DAAC 016FDC00
0012DAB0 0012DAE4
0012DAB4 00EB338C RETURN to xptools.00EB338C from xptools.00EB36D3
0012DAB8 0000023C
0012DABC 016FB940
0012DAC0 00000000
0012DAC4 00000002
0012DAC8 000023C2
0012DACC 0012F5A4
0012DAD0 00000000
0012DAD4 00EDC670 xptools.00EDC670
0012DAD8 00EDC670 xptools.00EDC670
0012DAE0 00000001

```

-Be sure the address is returned from the back window wa CPU, Ctrl-G to address it:

Enter expression to follow

00EB36D3

OK Cancel

```

00EB36D3 $ 55 PUSH EBP
00EB36D4 . 8BEC MOV EBP,ESP
00EB36D6 . 81EC 00010000 SUB ESP,100
00EB36D8 . 53 PUSH EBX
00EB36DA . 56 PUSH ESI
00EB36DC . 57 PUSH EDI
00EB36DE . 8B45 00 MOV EAX,DWORD PTR SS:[EBP+0]

```

Ctrl-R-here:

Address	Disassembly
00EB3387	CALL xptools.00EB36D3
00EB3642	CALL xptools.00EB36D3
00EB36D3	PUSH EBP

Enter to-Call function 2:

```

00EB3642 E8 8C000000 CALL xptools.00EB36D3
00EB3647 . 83C4 0C ADD ESP,0C
00EB3649 . 9C PUSHFD
00EB364B . 60 PUSHAD
00EB364C .v EB 2B JMP SHORT xptools.00EB3679

```

Click-it must submit:

```

00EB3642 90 NOP
00EB3643 90 NOP
00EB3644 90 NOP
00EB3645 90 NOP
00EB3646 90 NOP
00EB3647 . 83C4 0C ADD ESP,0C
00EB3649 . 9C PUSHFD
00EB364B . 60 PUSHAD
00EB364C .v EB 2B JMP SHORT xptools.00EB3679

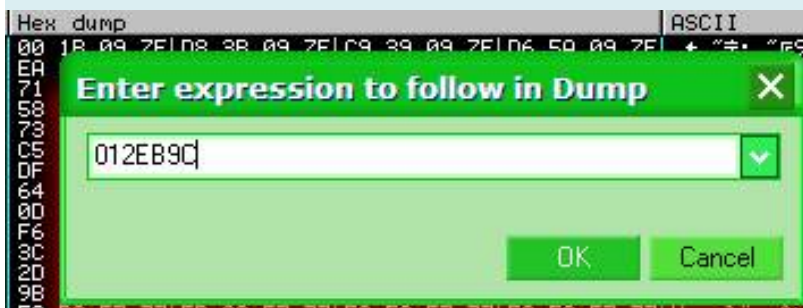
```

To-2 months ago Patch create unlimited loop. (Patch to EBFE)

+ The first is a patch at EP Buffer month with Father Process:

[EP Buffer = (OEP child process - Address3) Buffer3 +] (formula and called temporary)

** With *OEP child process*: the small pDebugEvent the top ko, dump in the window, Ctrl-G will see to it OEP:



Address	Value	Comment
0012EB9C	00000001	
0012EBA0	00000C00	
0012EBA4	00000840	
0012EBA8	80000001	
0012EBAC	00000000	
0012EBB0	00000000	
0012EBB4	0063D918	xptools.0063D918
0012EBB8	00000002	
0012EBBC	00000000	
0012EBC0	0063D918	xptools.0063D918
0012EBC4	0063D918	xptools.0063D918
0012EBC8	00000000	

**** Address3 and Buffer3 have been stopped at the BP *WriteProcessMemory* 3 times.**

0012D98C	00EB467F	CALL to WriteProcessMemo
0012D990	0000004C	hProcess = 0000004C
0012D994	0063D000	Address = 63D000
0012D998	016FCC00	Buffer = 016FCC00
0012D99C	00001000	BytesToWrite = 1000 (409
0012D9A0	0012D9A8	BytesWritten = 0012D9A8

So EP-Buffer here = $(63D918 - 63D000) 16FCC00 + = 16FD518$ (the number of each machine Buffer3 each other, even each installation is each other, only child OEP process of installation time is similar)

-Ctrl-G to that (in the window dump) and the **558B** Patch **EBFE**. Or PUPE also.

+ How 2: we are always in every patch of OEP child process, such as how to find the OEP (OEP = **63D918**), but need more of it PID, choose Attach:

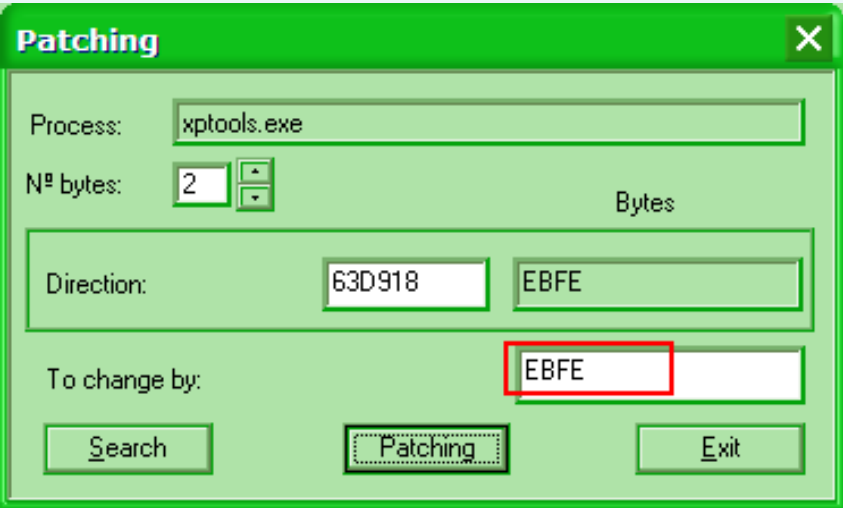
00000228	winlogon	NetDDE Agent	\\??\C:\WINDOWS\system32\
00000B30	xptools		E:\XP Tools\xptools.exe
00000C00	xptools		E:\XP Tools\xptools.exe

Open-choose 2, because as with the 1 after Defeat DebugBlocker, word will get a NAG error, although still to be done but do not like Mo NAG it.

-Collect enough information, open up PUPE 2002 (2 hours, the force is used only ni, ko is Ctrl-G):



Often-child process PID is located on the PUPE. Select Patch it:



Close-call PUPE, Olly any way back, remove bp:

Command : bc WriteProcessMemory|

-Next:

Command : bp WaitForDebugEvent|

F9-play:

77EAF13C	55	PUSH EBP
77EAF13D	8BEC	MOV EBP,ESP
77EAF13F	83EC 68	SUB ESP,68
77EAF142	56	PUSH ESI
77EAF143	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
77EAF146	8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]
77EAF149	50	PUSH EAX
77EAF14B	58	POP EAX

Down-NGO, and Click to *Follow Dissambler*:

0012DAEC	00EB0B44	[CALL to WaitForDebugEvent
0012DAF0	0012EB9C	pDebugEvent = 0012EB9C
0012DAF4	000003E8	Timeout = 1000. ms

Follow in Disassembler	Enter	[CALL to WaitForDebugEvent
Follow in Dump		pDebugEvent = 0012EB9C
		Timeout = 1000. ms

One-to:

00EB0B44	85C0	TEST EAX,EAX
00EB0B46	75 0F84 B1230000	JNE xptools.00EB2EFD
00EB0B4C	60	PUSHAD
00EB0B4D	33C0	XOR EAX,EAX
00EB0B4F	75 02	JNZ SHORT xptools.00EB0B53

Fill out tui-ni to submit it:

00EB0B31	61	POPAD
00EB0B32	68 E8030000	PUSH 3E8
00EB0B37	8B95 E4F5FFFF	MOV EDX,DWORD PTR SS:[EBP-A10]
00EB0B3D	52	PUSH EDX
00EB0B3E	FF15 A080ED00	CALL DWORD PTR DS:[K&KERNEL32.WaitForDebugEvent]

-Yes:


```

00EB0B31 90 NOP
00EB0B32 90 NOP
00EB0B33 90 NOP
00EB0B34 90 NOP
00EB0B35 90 NOP
00EB0B36 90 NOP
00EB0B37 90 NOP
00EB0B38 90 NOP
00EB0B39 90 NOP
00EB0B3A 90 NOP
00EB0B3B 90 NOP
00EB0B3C 90 NOP
00EB0B3D 90 NOP
00EB0B3E 90 NOP
00EB0B3F 90 NOP
00EB0B40 90 NOP
00EB0B41 90 NOP
00EB0B42 90 NOP
00EB0B43 90 NOP
00EB0B44 85C0 TEST EAX,EAX
00EB0B46 0F84 B1230000 JE xptools.00EB2EFD
00EB0B4C 60 PUSHAD

```

- New orders origin immediately TEXT EAX, EAX:

```

00EB0B44 85C0 TEST EAX,EAX
00EB0B46 0F84 B1230000 JE xptools.00EB2EFD
00EB0B4C 60 PUSHAD

```

Patch-time Mr. JE below:

```

00EB0B44 85C0 TEST EAX,EAX
00EB0B46 E9 B50455FF JMP xptools.00401000
00EB0B48 90 NOP
00EB0B4C 60 PUSHAD

```

At the command-JMP has Patch, Enter to go to 401,000:

```

00401000 0000 ADD BYTE PTR DS:[EAX],AL
00401002 0000 ADD BYTE PTR DS:[EAX],AL
00401004 0000 ADD BYTE PTR DS:[EAX],AL
00401006 0000 ADD BYTE PTR DS:[EAX],AL
00401008 0000 ADD BYTE PTR DS:[EAX],AL
0040100A 0000 ADD BYTE PTR DS:[EAX],AL
0040100C 0000 ADD BYTE PTR DS:[EAX],AL
0040100E 0000 ADD BYTE PTR DS:[EAX],AL
00401010 0000 ADD BYTE PTR DS:[EAX],AL
00401012 0000 ADD BYTE PTR DS:[EAX],AL
00401014 0000 ADD BYTE PTR DS:[EAX],AL
00401016 0000 ADD BYTE PTR DS:[EAX],AL
00401018 0000 ADD BYTE PTR DS:[EAX],AL
0040101A 0000 ADD BYTE PTR DS:[EAX],AL
0040101C 0000 ADD BYTE PTR DS:[EAX],AL
0040101E 0000 ADD BYTE PTR DS:[EAX],AL
00401020 0000 ADD BYTE PTR DS:[EAX],AL

```

Ngo-time window to dump:

Address	Value	Comment
0012EB9C	00000001	
0012EBA0	00000C00	
0012EBA4	00000840	
0012EBA8	80000001	
0012EBAC	00000000	
0012EBB0	00000000	
0012EBB4	0063D918	xptools.0063D918
0012EBB8	00000002	
0012EBBC	00000000	
0012EBC0	0063D918	xptools.0063D918
0012EBC4	0063D918	xptools.0063D918
0012EBC8	00000000	
0012EBCC	E1AB2D01	

Patch-class order with the corresponding Address:

```

00401000 8105 B4EB1200 ADD DWORD PTR DS:[12EBB4],1000
0040100A 8105 C0EB1200 ADD DWORD PTR DS:[12EBC0],1000
00401014 8105 C4EB1200 ADD DWORD PTR DS:[12EBC4],1000
0040101E 0000 ADD BYTE PTR DS:[EAX],AL
00401020 0000 ADD BYTE PTR DS:[EAX],AL

```

-Alt-M, more information:

00400000	00001000	xptools		PE header	Image
00401000	0023E000	xptools	CODE		Image
0063F000	0000E000	xptools	DATA		Image
00641000	00833000	xptools	BSS		Image
00E80000	00004000	xptools	.idata		Image
00E84000	00001000	xptools	.tls		Image
00E85000	00001000	xptools	.xdata		Image

Back-patch, the text can be for any area with **Cmp** in 3 areas are small, because they are starting from a value of **400,000** (the value that we will modify below), then they are public **1000** from the same:

00401000	8105 B4EB1200	ADD DWORD PTR DS:[12EBB4],1000
0040100A	8105 C0EB1200	ADD DWORD PTR DS:[12EBC0],1000
00401014	8105 C4EB1200	ADD DWORD PTR DS:[12EBC4],1000
0040101E	813D B4EB1200	CMP DWORD PTR DS:[12EBB4],xptools.0063F000
00401028	0000	ADD BYTE PTR DS:[EAX],AL
0040102D	0000	ADD BYTE PTR DS:[EAX],AL

Back-seat **TEST EAX, EAX:**

00EB0B44	85C0	TEST EAX, EAX
00EB0B46	E9 B50455FF	JMP xptools.00401000
00EB0B48	90	NOP
00EB0B4C	60	PUSHAD
00EB0B4D	33C0	WOP EAX, EAX

Remember Address in-command filed, back to **401,000**, Patch (NOP remember a few more things for the good):

00401000	8105 B4EB1200	ADD DWORD PTR DS:[12EBB4],1000
0040100A	8105 C0EB1200	ADD DWORD PTR DS:[12EBC0],1000
00401014	8105 C4EB1200	ADD DWORD PTR DS:[12EBC4],1000
0040101E	813D B4EB1200	CMP DWORD PTR DS:[12EBB4],xptools.0063F000
00401028	0F85 1DFBA000	JNZ xptools.00EB0B48
0040102E	90	NOP
0040102F	90	NOP
00401030	0000	ADD BYTE PTR DS:[EAX],AL

Time-to do the end, the window is still a dump ni must do:

Address	Value	Comment
0012EB9C	00000001	
0012EBA0	00000C00	
0012EBA4	00000840	
0012EBA8	00000001	
0012EBAC	00000000	
0012EBB0	00000000	
0012EBB4	0063D918	xptools.0063D918
0012EBB8	00000002	
0012EBBC	00000000	
0012EBC0	0063D918	xptools.0063D918
0012EBC4	0063D918	xptools.0063D918
0012EBC8	00000000	

Click-to Modify each month and **400,000** of them (MZIP):

00401054	0000	ADD BYTE	Binary
00401056	0000	ADD BYTE	Modify
00401058	0000	ADD BYTE	Breakpoint
0040105A	0000	ADD BYTE	Search for
0040105C	0000	ADD BYTE	Follow in Dump
0040105E	0000	ADD BYTE	Go to
00401060	0000	ADD BYTE	Hex
00401062	0000	ADD BYTE	Text
00401064	0000	ADD BYTE	Short
00401066	0000	ADD BYTE	Long
00401068	0000	ADD BYTE	Float

Modify data at 0012...

Hexadecimal: 400000

Signed: 4194304

Unsigned: 4194304

OK Cancel

-Done:

Address	Value	Comment
0012EB9C	00000001	
0012EBA0	00000C00	
0012EBA4	00000840	
0012EBA8	80000001	
0012EBAC	00000000	
0012EBB0	00000000	
0012EBB4	00400000	ASCII "MZP"
0012EBB8	00000002	
0012EBBC	00000000	
0012EBC0	00400000	ASCII "MZP"
0012EBC4	00400000	ASCII "MZP"
0012EBC8	00000000	
0012EBCC	F1AB2001	

Now a **set-BP** immediately **NOP** commands:

00401000	8105 B4EB1200	ADD DWORD PTR DS:[12EBB4],1000
0040100A	8105 C0EB1200	ADD DWORD PTR DS:[12EBC0],1000
00401014	8105 C4EB1200	ADD DWORD PTR DS:[12EBC4],1000
0040101E	813D B4EB1200	CMP DWORD PTR DS:[12EBB4],xptools.0063F000
00401028	0F85 10FBA000	JNZ xptools.00EB0B4B
0040102E	90	NOP
0040102F	90	NOP
00401030	0000	ADD BYTE PTR DS:[EAX],AL
00401032	0000	ADD BYTE PTR DS:[EAX],AL

-If you want to check the Trace, the lower BP may place orders immediately **JNZ** to see the value is added to the questions in order to increase the like, then find speed for me to order this subject **NOP** to a large degree that we bring to compare me in order **Cmp**.

-Now press F9, a long, break it in order **NOP**:

00401000	8105 B4EB1200	ADD DWORD PTR DS:[12EBB4],1000
0040100A	8105 C0EB1200	ADD DWORD PTR DS:[12EBC0],1000
00401014	8105 C4EB1200	ADD DWORD PTR DS:[12EBC4],1000
0040101E	813D B4EB1200	CMP DWORD PTR DS:[12EBB4],xptools.0063F000
00401028	0F85 10FBA000	JNZ xptools.00EB0B4B
0040102E	90	NOP
0040102F	90	NOP
00401030	0000	ADD BYTE PTR DS:[EAX],AL
00401032	0000	ADD BYTE PTR DS:[EAX],AL

The hour-always thoroughly **DebugBlocker** ship, remember PID process but the child, any patch:

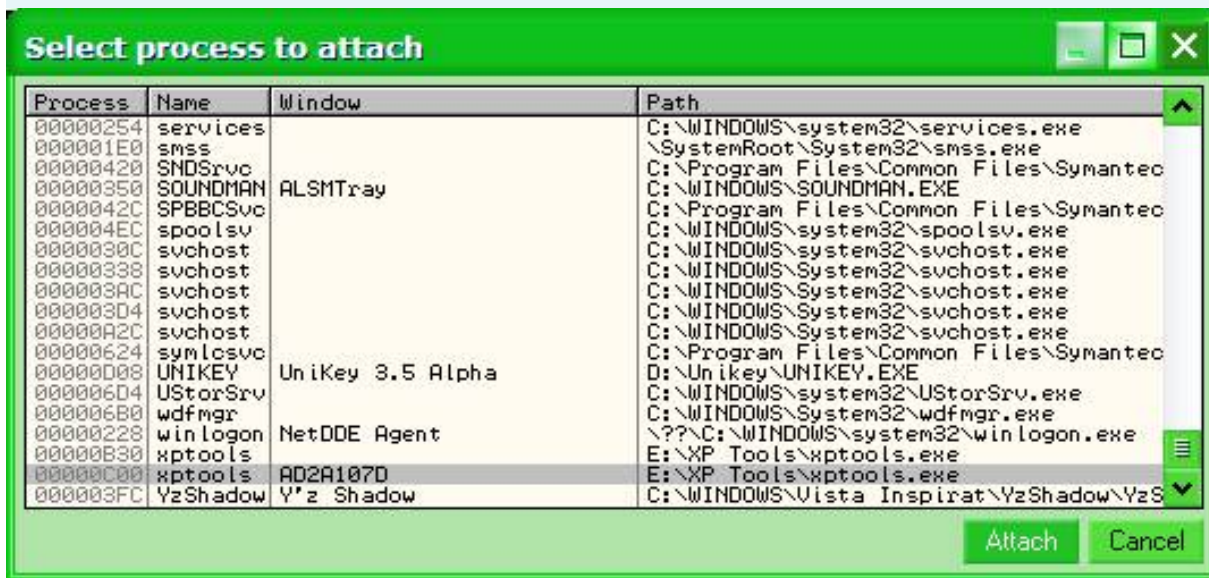
00401014	8105 C4EB1200	ADD DWORD PTR DS:[12EBC4],1000
0040101E	813D B4EB1200	CMP DWORD PTR DS:[12EBB4],xptools.0063F000
00401028	0F85 10FBA000	JNZ xptools.00EB0B4B
0040102E	68 000C0000	PUSH 0C00
00401033	E8 FBE1AA77	CALL kernel32.DebugActiveProcessStop
00401038	90	NOP
00401039	90	NOP
0040103A	0000	ADD BYTE PTR DS:[EAX],AL

Patch-Trace to complete the order wa any CALL:

00401014	8105 C4EB1200	ADD DWORD PTR DS:[12EBC4],1000
0040101E	813D B4EB1200	CMP DWORD PTR DS:[12EBB4],xptools.0063F000
00401028	0F85 10FBA000	JNZ xptools.00EB0B4B
0040102E	68 000C0000	PUSH 0C00
00401033	E8 FBE1AA77	CALL kernel32.DebugActiveProcessStop
00401038	90	NOP
00401039	90	NOP
0040103A	0000	ADD BYTE PTR DS:[EAX],AL

Registers (FPU)	
EAX	00000001
ECX	00120A8C
EDX	7FFE0304
EBX	0012F5A4
ESP	00120A08
EBP	0012F5B4
ESI	00002309
EDI	00000009

Open-more Olly other Attach child process to do:



```

77F767CE  C3      RETN
77F767CF  CC      INT3
77F767D0  C3      RETN
77F767D1  8B4424 04  MOV EAX,DWORD PTR SS:[ESP+4]
77F767D5  CC      INT3
77F767D6  C2 0400  RETN 4
77F767D9  64:A1 18000000 MOV EAX,DWORD PTR FS:[18]
77F767DF  C3      RETN
77F767E0  F7      PUSH EBX

```

-F9, F12 and Alt-C to return to the CPU:

```

0063D918 - EB FE      JMP SHORT xpctools.0063D918
0063D91A  EC      IN AL,DX
0063D91B  81C4 D4FEFFFF ADD ESP,-12C
0063D921  53      PUSH EBX
0063D922  56      PUSH ESI
0063D923  57      PUSH EDI
0063D924  33C0    XOR EAX,EAX
0063D926  8985 D4FEFFFF MOV DWORD PTR SS:[EBP-12C],EAX

```

Click-to dump --- Follow it:

```

0063D918 - EB FE      JMP SHORT xpctools.0063D918
0063D91A  EC      IN AL,DX
0063D91B  81C4 D4FEFFFF ADD ESP,-12C
0063D921  53      PUSH EBX
0063D922  56      PUSH ESI
0063D923  57      PUSH EDI
0063D924  33C0    XOR EAX,EAX
0063D926  8985 D4FEFFFF MOV DWORD PTR SS:[EBP-12C],EAX
0063D92C  8945 D8  MOV EAX,EDX
0063D92F  8945 DC  MOV EAX,EBX
0063D932  8945 E0  MOV EAX,ESI
0063D935  8945 E4  MOV EAX,EDI
0063D938  8945 E8  MOV EAX,EBP
0063D93B  8945 EC  MOV EAX,ESP
0063D93E  8945 F0  MOV EAX,EBP
0063D941  B8 68D06300 MOV EAX,68D06300
0063D946  E8 D8A6DCFF CALL EBX
0063D948  33C0    XOR EAX,EAX
0063D94D  55      PUSH EBX
0063D94E  68 CFDF6300 MOV EAX,CFDF6300
0063D953  64:FF30  PUSH EAX
0063D956  64:8920  MOV EAX,8920
0063D959  33D2    XOR EDI,EDI
0063D95B  55      PUSH EBX
0063D95C  68 9FDF6300 MOV EAX,9FDF6300

```

Backup
Copy
Binary
Assemble Space
Label :
Comment ;
Breakpoint
Run trace
Follow Enter
Go to
Thread
Follow in Dump Selection

```

Address Hex dump
0063D918 EB FE EC 81 C4 D4 FE FF
0063D928 D4 FE FF FF 89 45 D8 89
0063D938 89 45 E8 89 45 EC 89 45

```

Patch-old **EBFE** as -> **558B**:

Address	Hex	dump
0063D918	55 8B EC 81 C4 D4 FE FF	
0063D928	D4 FE FF FF 89 45 D8 89	
0063D938	89 45 E8 89 45 EC 89 45	

In-CPU:

0063D918	55	PUSH EBP
0063D919	8B EC	MOV EBP, ESP
0063D91B	81 C4 D4 FE FF FF	ADD ESP, -12C
0063D921	53	PUSH EBX
0063D922	56	PUSH ESI
0063D923	57	PUSH EDI
0063D924	33 C0	XOR EAX, EAX
0063D926	89 85 D4 FE FF FF	MOV DWORD PTR SS:[EBP-12C], EAX
0063D92C	89 45 D8	MOV DWORD PTR SS:[EBP-28], EAX

And-dump is the Lord of PE chiêu dump Full Ta kiếp:

[LordPE Deluxe] by yoda			
Path	PID	ImageBase	ImageSize
e:\xp tools\xptools.exe	00000B30	00400000	00EB1000
e:\xp tools\xptools.exe	00000C00	00400000	00EB1000
d:\conserve\programs\download\crack_tool...	00000110	00400000	00153000
d:\conserve\programs\download\crack_tool...	00000B34	00400000	00036000

dump full...
dump partial...
dump region...

Full-dump it. These Close Olly time. In Olly's child process, from OEP, dom down to see the command CALL:

0063D918	55	PUSH EBP
0063D919	8B EC	MOV EBP, ESP
0063D91B	81 C4 D4 FE FF FF	ADD ESP, -12C
0063D921	53	PUSH EBX
0063D922	56	PUSH ESI
0063D923	57	PUSH EDI
0063D924	33 C0	XOR EAX, EAX
0063D926	89 85 D4 FE FF FF	MOV DWORD PTR SS:[EBP-12C], EAX
0063D92C	89 45 D8	MOV DWORD PTR SS:[EBP-28], EAX
0063D92F	89 45 DC	MOV DWORD PTR SS:[EBP-24], EAX
0063D932	89 45 E0	MOV DWORD PTR SS:[EBP-20], EAX
0063D935	89 45 E4	MOV DWORD PTR SS:[EBP-1C], EAX
0063D938	89 45 E8	MOV DWORD PTR SS:[EBP-18], EAX
0063D93B	89 45 EC	MOV DWORD PTR SS:[EBP-14], EAX
0063D93E	89 45 F0	MOV DWORD PTR SS:[EBP-10], EAX
0063D941	B8 68 D0 06 30 00	MOV EAX, xptools.0063D068
0063D946	E8 D8 A6 DC FF	CALL xptools.00408038
0063D94B	33 C0	XOR EAX, EAX

-No Trace to it, you just press Enter to:

00408038	50	PUSH EAX
00408039	6A 00	PUSH 0
0040803B	E8 F8 FE FF FF	CALL xptools.00407F38
00408040	BA 08 F1 63 00	MOV EDX, xptools.0063F108
00408045	52	PUSH EDX
00408046	89 45 D8	MOV DWORD PTR SS:[EBP-28], EAX

Lai-a CALL command below:

00408038	50	PUSH EAX
00408039	6A 00	PUSH 0
0040803B	E8 F8 FE FF FF	CALL xptools.00407F38
00408040	BA 08 F1 63 00	MOV EDX, xptools.0063F108
00408045	52	PUSH EDX

-Enter into it:

00407F38	- FF25 4403E800	JMP DWORD PTR DS:[E80344]	
00407F3E	8BC0	MOV EAX,EAX	
00407F40	- FF25 4003E800	JMP DWORD PTR DS:[E80340]	kernel32.LocalAlloc
00407F46	8BC0	MOV EAX,EAX	
00407F48	- FF25 3C03E800	JMP DWORD PTR DS:[E8033C]	kernel32.TlsGetValue
00407F4E	8BC0	MOV EAX,EAX	
00407F50	- FF25 3803E800	JMP DWORD PTR DS:[E80338]	kernel32.TlsSetValue
00407F56	8BC0	MOV EAX,EAX	

-Here we go to find small IAT, Click right, Follow dump:

Follow in Dump	Selection
Search for	Memory address

Being-in window dump:

Address	Value	Comment
00E80344	0181910F	
00E80348	77E7ADA9	kernel32.GetModuleFileNameA
00E8034C	01816363	
00E80350	77DE63B1	ADVAPI32.RegSetValueExA
00E80354	77DD2410	ADVAPI32.RegQueryValueExA
00E80358	77DE6CF4	ADVAPI32.RegQueryInfoKeyA
00E8035C	77DD229A	ADVAPI32.RegOpenKeyExA
00E80360	77DE696D	ADVAPI32.RegFlushKey
00E80364	77DE6A68	ADVAPI32.RegEnumValueA
00E80368	77DE6F8B	ADVAPI32.RegEnumKeyExA
00E8036C	77DE65D4	ADVAPI32.RegDeleteValueA
00E80370	77DE68E2	ADVAPI32.RegDeleteKeyA
00E80374	018196C8	
00E80378	01819308	
00E8037C	77DD2FCD	ADVAPI32.OpenProcessToken
00E80380	77DF582F	ADVAPI32.LookupPrivilegeValueA
00E80384	77DD4E99	ADVAPI32.GetUserNameA
00E80388	77DD31FD	ADVAPI32.AdjustTokenPrivileges

Search Start-IAT, pulled up:

00E80208	77E7A7DF	kernel32.GetCurrentThreadId
00E8020C	77F525CA	ntdll.RtlDeleteCriticalSection
00E80210	77F75690	ntdll.RtlLeaveCriticalSection
00E80214	77F755DE	ntdll.RtlEnterCriticalSection
00E80218	77E7A745	kernel32.InitializeCriticalSection

-In fact this IAT, we get the name of a memory function is then, can then use *APIAddress* for it, but here only the per always remember the IAT Start Address is: **00E80208**

-Why do not open up to fix ImportREC IAT always, because the files are done Fix the Crash dump her Kon oi.Do the value of this table IAT damaged many seats (text can be checked by clicking to select HEX to 16, so some value is then compared with the table that we IAT *Defeat DebugBlocker* a normal)

Magic Jump 2/Find and Fix IAT:

Close all-time, open a different light in Olly, soft load again. Then run *Scripts Armadillo Detach from Client* (*DebugBlocker* to kill it that):

7 OllyDump	
8 OllyScript	Run script...
9 ApiBreak	Abort

<input checked="" type="checkbox"/> Armadillo Detach from Client + Unpack (1000 bytes method).txt
<input checked="" type="checkbox"/> Armadillo Detach from Client.txt

Wait-it makes complete:



-The script for this I was working out better than *Armadillo DETECH Plugin* for Olly. With the script only after having failed form **Anti-Breakpoint**, which is the form you need to move BP down a bit when set bp *WriteProcessMemory*. So we should not rely on script that make it understand how to use that in time to save time.

So-ready Attach the child process and then, a quick process PID's child with Alt-L:

```

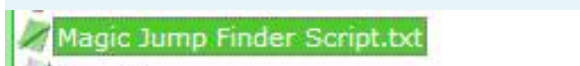
lpBuffer] = 0000FEEB
77EAF13C Breakpoint at kernel32.WaitForDebugEvent
77EAF13C Breakpoint at kernel32.WaitForDebugEvent
77EAF13C Breakpoint at kernel32.WaitForDebugEvent
pDebugEvent = 0012EB9C
child_ProcID = 0000006C
77E7D342 New thread with ID 00000208 created

```

Open-more Olly another child Attach process, Follow dump and change into **EBFE 558B**, such as call ha:

00EC0689	55	PUSH EBP	
00EC068A	8BEC	MOV EBP,ESP	
00EC068C	6A FF	PUSH -1	
00EC068E	68 E0D7ED00	PUSH xptools.00EDD7E0	
00EC0693	68 7000EC00	PUSH xptools.00EC0070	SE handler installat
00EC0698	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00EC069E	50	PUSH EAX	
00EC069F	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00EC06A6	83EC 58	SUB ESP,58	
00EC06A9	53	PUSH EBX	
00EC06AA	56	PUSH ESI	
00EC06AB	57	PUSH EDI	
00EC06AC	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00EC06AF	FF15 4C81ED00	CALL DWORD PTR DS:[<&kernel32.GetVersion	kernel32.GetVersion

-Now you will find Magic Jump and Fix it. Maybe *"he GetModuleHandleA"*, then press F9 xiu tired to sign "kernel32.dll", Magic Fix and Jump. Fortunately Scripts is now once again demonstrated the effectiveness of them. Using *Magic Jump Finder Scripts*:



-Patiently waiting for some:



Stop-packages:

01815407	8B0D B80B8401	MOV ECX,DWORD PTR DS:[1840BB8]	
0181540D	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
01815410	A1 B80B8401	MOV EAX,DWORD PTR DS:[1840BB8]	
01815415	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
01815418	75 16	JNZ SHORT 01815430	
0181541A	8D85 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
01815420	50	PUSH EAX	
01815421	FF15 BC708301	CALL DWORD PTR DS:[18370BC]	kernel32.LoadLibraryA
01815427	8B0D B80B8401	MOV ECX,DWORD PTR DS:[1840BB8]	
0181542D	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
01815430	A1 B80B8401	MOV EAX,DWORD PTR DS:[1840BB8]	
01815435	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
01815438	0F84 AD000000	JE 018154EB	Magic Jump
0181543E	33C9	XOR ECX,ECX	
01815440	8B03	MOV EAX,DWORD PTR DS:[EBX]	
01815442	3938	CMP DWORD PTR DS:[EAX],EDI	
01815444	74 06	JE SHORT 0181544C	

Fix to-do:

01815407	8B0D B80B8401	MOV ECX,DWORD PTR DS:[1840BB8]	
0181540D	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
01815410	A1 B80B8401	MOV EAX,DWORD PTR DS:[1840BB8]	
01815415	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
01815418	75 16	JNZ SHORT 01815430	
0181541A	8D85 B4FEFFFF	LEA EAX,DWORD PTR SS:[EBP-14C]	
01815420	50	PUSH EAX	
01815421	FF15 BC708301	CALL DWORD PTR DS:[18370BC]	kernel32.LoadLibraryA
01815427	8B0D B80B8401	MOV ECX,DWORD PTR DS:[1840BB8]	
0181542D	89040E	MOV DWORD PTR DS:[ESI+ECX],EAX	
01815430	A1 B80B8401	MOV EAX,DWORD PTR DS:[1840BB8]	
01815435	393C06	CMP DWORD PTR DS:[ESI+EAX],EDI	
01815438	E3 AE000000	JMP 018154EB	
0181543D	90	HOP	
0181543E	33C9	XOR ECX,ECX	

Remember to-Hardware BreakPoint BP is also considered the home to remove, then press Alt-M, set up a BP:

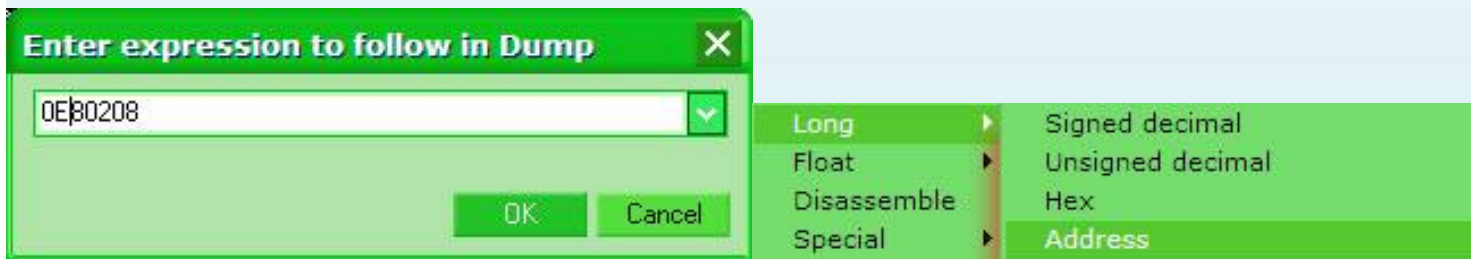
00320000	00013000			Map	R	E	R	E
003E0000	00002000			Map	R	E	R	E
003F0000	00008000			Priv	RW		RW	
00400000	00001000	xptools	PE header	Imag	R		RWE	
00401000	0023E000	xptools	CODE	Imag	R		RWE	
0063F000	0000E000	xptools	DATA	Imag	R		RWE	
0064D000	00833000	xptools	BSS	Imag	R		RWE	
00E80000	00004000	xptools	.idata	Imag	R		RWE	
00E84000	00001000	xptools	.tls	Imag	R		RWE	
00E85000	00001000	xptools	.rdata	Imag	R		RWE	
00E86000	00022000	xptools	.reloc	Imag	R		RWE	
00EA8000	00020000	xptools	.text	Imag	R		RWE	
00EC8000	00010000	xptools	.adata	Imag	R		RWE	
00ED0000	00010000	xptools	.data	Imag	R		RWE	
00EE8000	00010000	xptools	.reloc1	Imag	R		RWE	
00EF8000	001A0000	xptools	.pdata	Imag	R		RWE	
01098000	00219000	xptools	.rsrc	Imag	R		RWE	

-Shift-F9 vuuuuuuu:

0063D918	58	POP EAX	0182ECAB
0063D919	2B10	SUB EDX,DWORD PTR DS:[EAX]	
0063D91B	A9 C97402D7	TEST EAX,D70274C9	
0063D920	F2:	PREFIX REPNE:	Superfluous prefix
0063D921	F3:AA	REP STOS BYTE PTR ES:[EDI]	
0063D923	7F 3E	JG SHORT xptools.0063D963	
0063D925	60	PUSHAD	
0063D926	75 AD	JNZ SHORT xptools.0063D8D5	
0063D928	D95E 03	FSTP DWORD PTR DS:[ESI+3]	
0063D92B	D7	XLAT BYTE PTR DS:[EBX+AL]	
0063D92C	84E5	TEST CH,AH	

The OEP-gõì but this time the code here has been encrypted, which is why we dump the Lord with Full PE when to OEP ko encrypted before (this time thàng still *CopyMEM II*). But IAT is set by this month ko damaged, because if the corrupt soft ko can be perimeter Run a).

-Remember the address of IAT Start we find the time, dump in the window, Ctrl-G to it:



-Choose the viewpoint:

Address	Value	Comment
00E801FC	00000000	
00E80200	00000000	
00E80204	00000000	
00E80208	77E7A7DF	kernel32.GetCurrentThreadId
00E8020C	77F525CA	ntdll.RtlDeleteCriticalSection
00E80210	77F75690	ntdll.RtlLeaveCriticalSection
00E80214	77F755DE	ntdll.RtlEnterCriticalSection
00E80218	77F7A745	kernel32.InitializeCriticalSection

IAT Start

-Now go find a new IAT End, pulling down:

Address	Value	Comment
00E80C30	77E739A1	kernel32.MulDiv
00E80C34	01816330	
00E80C38	0192D388	SHELL32.SHEmptyRecycleBinA
00E80C3C	0192D0B1	SHELL32.SHQueryRecycleBinA
00E80C40	018162A5	
00E80C44	6E726568	
00E80C48	32336C65	
00E80C4C	6C6C642E	
00E80C50	00000000	

Len = A44

Okie-many number WA, not me nê profile information, select a little down and die Cuc month that time.

But wait-what more do it without resistance, to put in ImportREC, select child process, and in line 3-D:

IAT Infos needed

OEP: 0023D918 IAT AutoSearch

RVA: 00A80208 Size: 00000A44

Get-Import, Show Invalid, which has thẳng Cut Thunks that month, default default default, the people fell down many turtles, asked: "who want to have under ko?" Gât first dollop:

Imported Functions Found	
kernel32.dll	FTThunk:00A80208 NbFunc:32 (decimal:50) valid:YES
user32.dll	FTThunk:00A802D4 NbFunc:4 (decimal:4) valid:YES
advapi32.dll	FTThunk:00A802E8 NbFunc:3 (decimal:3) valid:YES
oleaut32.dll	FTThunk:00A802F8 NbFunc:F (decimal:15) valid:YES
kernel32.dll	FTThunk:00A80338 NbFunc:5 (decimal:5) valid:YES
advapi32.dll	FTThunk:00A80350 NbFunc:F (decimal:15) valid:YES
kernel32.dll	FTThunk:00A80390 NbFunc:84 (decimal:132) valid:YES
mpr.dll	FTThunk:00A805A4 NbFunc:3 (decimal:3) valid:YES
version.dll	FTThunk:00A805B4 NbFunc:3 (decimal:3) valid:YES

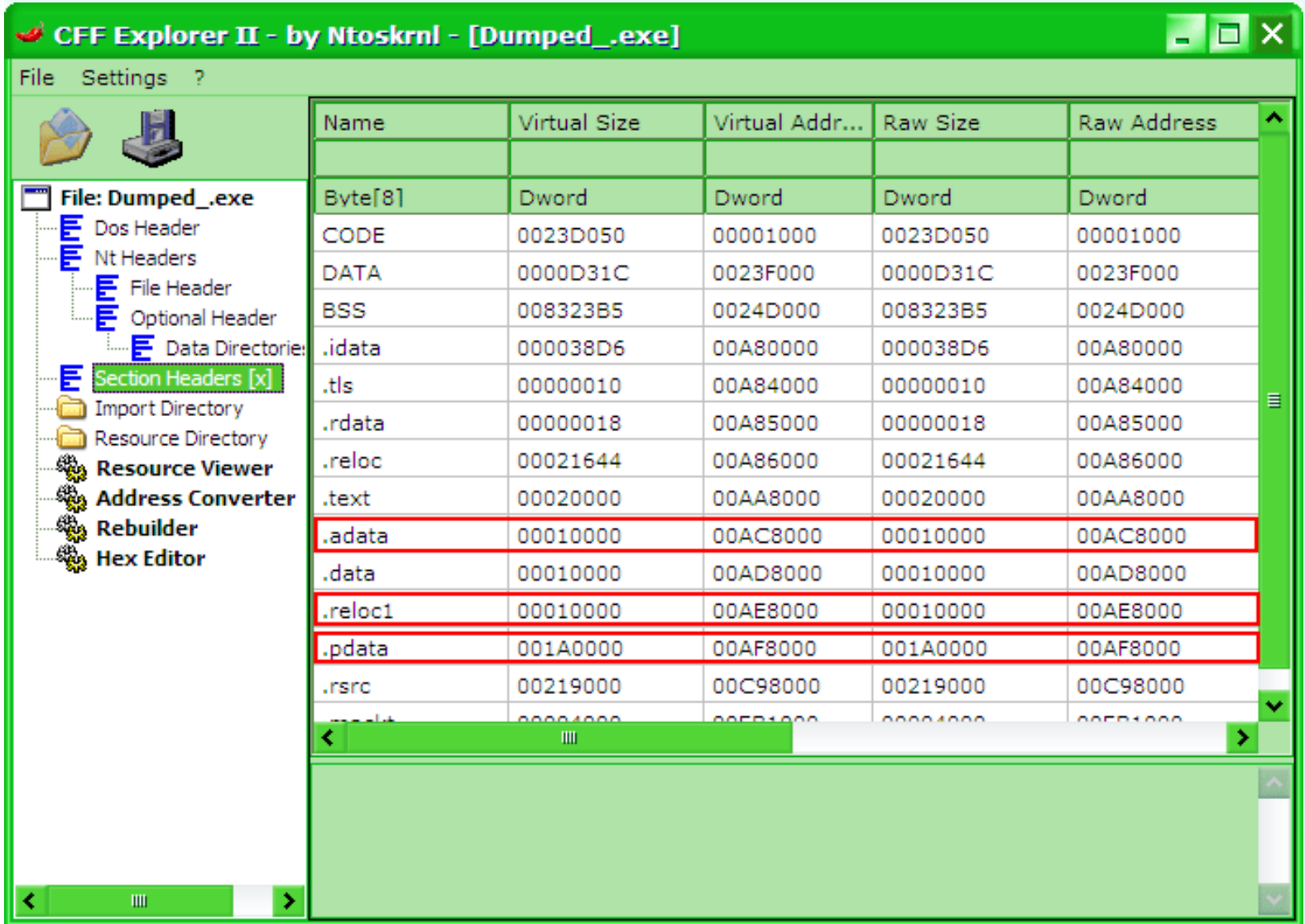
According to well-kill, dump Fix:





3/Rebuild:

Often Fix-it should dump complete rebuild the file to go a little light. Open CFF Explorer II, to do that:



-Delete Section dùm to do:

Change Section Flags

Add Section (Header Only)

Add Section (Empty Space)

Add Section (File Data)

Delete Section (Header Only)

Delete Section (Header And Data)

-Yes, but remember to add a few ni:

CODE	0023E000	00001000	0023D050	00001000
DATA	0000E000	0023F000	0000D31C	0023F000
BSS	00833000	0024D000	008323B5	0024D000
.idata	00004000	00A80000	000038D6	00A80000
.tls	00001000	00A84000	00000010	00A84000
.rdata	00001000	00A85000	00000018	00A85000
.reloc	00022000	00A86000	00021644	00A86000
.text	00030000	00AA8000	00020000	00AA8000
.data	001C0000	00AD8000	00010000	00AC8000
.rsrc	00219000	00C98000	00219000	00AD8000
.mact	00004000	00EB1000	00004000	00CF1000

-Base is still adjusting of Code, the Data to Analyze little later is correct:

CFF Explorer II - by Ntoskrnl - [Dumped_.exe]

File Settings ?

File: Dumped_.exe

- Dos Header
- Nt Headers
- File Header
- Optional Header**
- Data Directory:
- Section Headers [x]
- Import Directory
- Resource Directory
- Resource Viewer
- Address Converter

Member	Offset	Size	Value	Mean
Magic	00000118	Word	010B	PE32
MajorLinkerVersion	0000011A	Byte	53	
MinorLinkerVersion	0000011B	Byte	52	
SizeOfCode	0000011C	Dword	00023000	
SizeOfInitializedData	00000120	Dword	001DD000	
SizeOfUninitializedData	00000124	Dword	00000000	
AddressOfEntryPoint	00000128	Dword	0023D918	
BaseOfCode	0000012C	Dword	00AA8000	
BaseOfData	00000130	Dword	00A85000	
ImageBase	00000134	Dword	00400000	

Edit-on position with flash flash that the number per marked:

BaseOfCode	0000012C	Dword	00001000
BaseOfData	00000130	Dword	0023F000
ImageBase	00000134	Dword	00400000

The per-Save with the name:

Run-hand or do the time, run the **rebuilt.exe** koi, he he:

file:///C:/RCE%20Unpacking%20eBook%20[Tr...20Full%20XP%20Tools%20version%204.58.htm (18 of 42) [1/9/2009 9:46:22 LithiumLi]

still has the limit on the stars, the stars ..? ? Ko stars, watch the 3 hour:



-Phu, so we unpack DONE! gòi home. Hope that people understand more about *CopyMEM II*, especially **Takada**, so that the parents let go, do them a promotion to the higher level, parents see the filing ngò also very high, including the code to take oach oach. ..

-To ensure that many here feel the warm summer formula for self-stop stop lăng per xet. Ko where, what Tàn per the text that you get know, a we will go to the Crack 2 is more soft this. Open go buy meet interview. What you sit to relax a little time.

*Now this late-WA, also bít ko ko Interviews again, but .. chaaaaa from offish seen in torrential lố, gas business focused around the newsstand, including to air in her home that sap from rolling Interviews 're radiate, mừng chairs. Hold book stories, but not yet open but saw dao gas, gas for flood ngào end, shrine of, hix hix, kì this sure-to book stories always rách WA, put out to open the can do, Ummmmm Mo hair blowing purpose is to, first brain chao island, chairs ... Rammmm per seat automatically stop pop-up after a dao by Lam Vu pin to Ve Thập Nuong pin that as per the per ... aaa again at the **Cuu wifeless About***

II / Cracking Full:

Difficulties-frosted new from **Cuu About wifeless** with you. Hì hi.

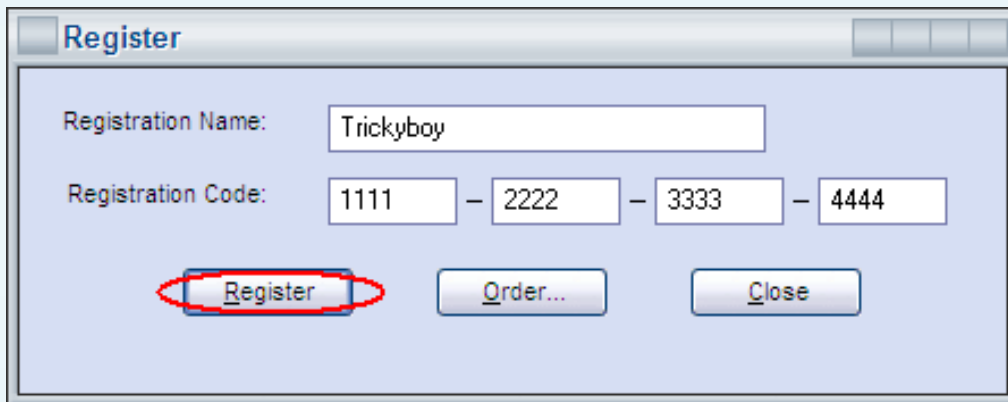
Um-speaking part on the next bit, after unpack, we do not doubt still be soft Crack. It is still counting the time as usual. Thus, from very unpack Ta Ro with the media that the grapes for Tàn reopened falling within the times of a noop. wai Do what now? Ko unpack probably finish it and then sit ngo count all that time?

In fact-here are 2 months, or a patch for it thought we register call, or find a real number Serial (2 hours by easily

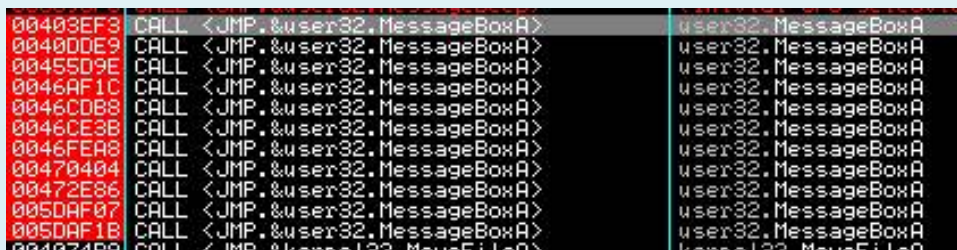
call home for the elderly to buy soft Serial number). But by going tuu so many long wá per bank of the call, only dành Patch:

Soft 1/Patch to use without register:

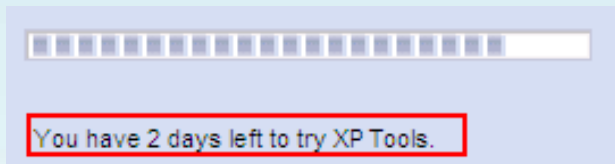
-Whether or Patch what should go away from them as signs of the table registered by Soft. Soft Run, Run during which it stops at **POP**, the degree to Shift-F9 to Run (perimeter have to press the long-long) and select the name and register with Fake Serial:



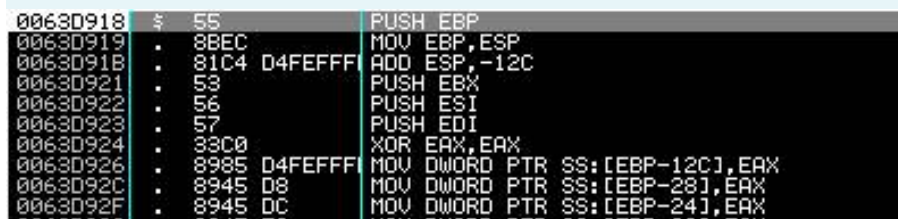
Register-Star press that do see it spray the Irak nhi, it is now WA, but each death, a new deep cold. So do not use methods against time with Stack of the Moon is then. And with the type ko NAG thềm the way it is set up 2 bp **GetDlgItemText** children and **GetWindowText**. But unfortunately the press Register also Cuc thềm break. then we try to set up BP all functions **MessageBoxA** see:



Multiple-WA, if the number of functions that are more easy ways ni also very congested. Okie, and hope you can start at the Bread for the **MessageBoxA** function. Tàn per But although that tàn Van Cuong, open i do like the open just like what the soft available:



Oooooo-remember ... a few words in the chain only to the memory you have. "Load File **rebuilt.exe** to go:

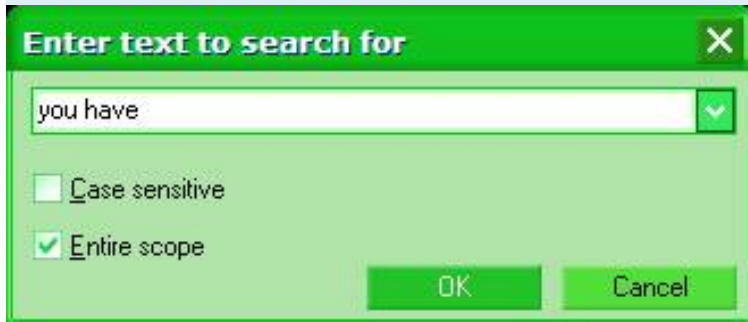


At per-rebuild has emphasized that to fix **Base of the Code** and **Data** gòi to do, like Olly hours Text Search String

exactly remember the previous rebuild under Search nha.Con who always want to unpack the files are still raw Xi is but remember to use *the Ultra String Plugin* by Olly to Search. Thôi, Search's take:



Cuc lúc-Type:



-Phu. They have to chain it. Kĩ looking back to see, what's on. Ái ai, ni a new right is the truth, they support:

0057015A	MOV EDX,rebuilde.005702DC	ASCII "c_frmAbout_licensed"
0057015F	MOV EAX,rebuilde.005702F8	ASCII "This product is licensed to: "
00570176	PUSH rebuilde.00570320	ASCII " " " " " "
0057017E	MOV EDX,rebuilde.0057032C	ASCII "c_frmAbout_code"
00570183	MOV EAX,rebuilde.00570344	ASCII "Registration Code: "
005701FE	MOV EDX,rebuilde.00570360	ASCII "c_frmAbout_msgOutOfDate"
00570203	MOV EAX,rebuilde.00570380	ASCII "Your XPTools has expired. Please purchase it online."
00570270	MOV EDX,rebuilde.005703C0	ASCII "c_frmAbout_prgLeft"
00570275	MOV EAX,rebuilde.005703DC	ASCII "You have %d days left to try XPTools."
00570280	ASCII "c_frmAbout_licen"	

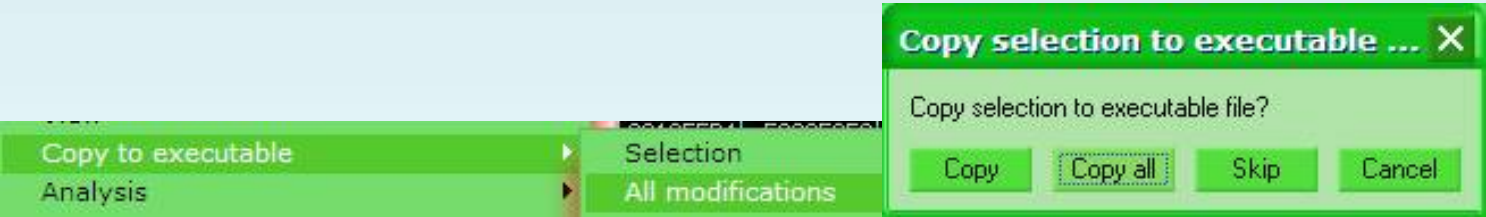
-Enter to have the "licensed", look to see what is on:

00570130	CALL rebuilde.005703C0	
00570142	TEST AL,AL	
00570144	JE rebuilde.005701D4	
0057014A	MOV DL,1	
0057014C	MOV EAX,DWORD PTR DS:[EBX+228]	
00570152	CALL rebuilde.00436354	
00570157	LEA ECX,DWORD PTR SS:[EBP-8]	
0057015A	MOV EDX,rebuilde.005702DC	ASCII "c_frmAbout_licensed"
0057015F	MOV EAX,rebuilde.005702F8	ASCII "This product is licensed"
00570164	CALL rebuilde.00520B38	
00570169	PUSH DWORD PTR SS:[EBP-8]	
0057016C	MOV EAX,DWORD PTR DS:[64BADC]	
00570171	MOV EAX,DWORD PTR DS:[EAX]	
00570173	PUSH DWORD PTR DS:[EAX+4]	
00570176	PUSH rebuilde.00570320	ASCII " " " " " "
0057017B	LEA ECX,DWORD PTR SS:[EBP-C]	
0057017E	MOV EDX,rebuilde.0057032C	ASCII "c_frmAbout_code"
00570183	MOV EAX,rebuilde.00570344	ASCII "Registration Code: "

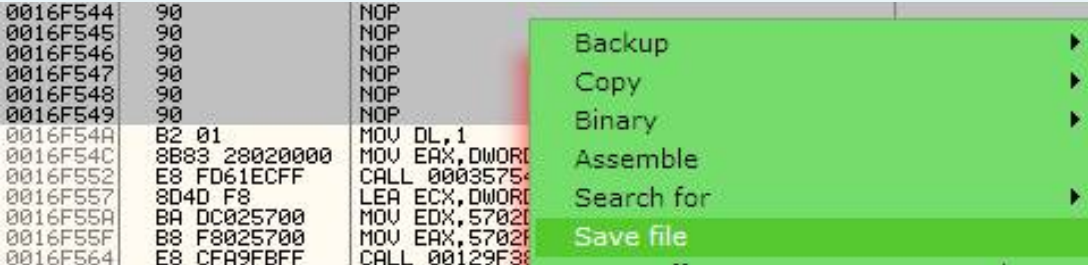
Há-A, where it can check whether we have registered or not, if it has not jump down to the number of days left, if it is to command wa JE and the "licensed to" the name of we Registration Code and more. So, here we do not want to skip it, submit it to:

00570130	CALL rebuilde.005703C0	
00570142	TEST AL,AL	
00570144	NOP	
00570145	NOP	
00570146	NOP	
00570147	NOP	
00570148	NOP	
00570149	NOP	
0057014A	MOV DL,1	
0057014C	MOV EAX,DWORD PTR DS:[EBX+228]	
00570152	CALL rebuilde.00436354	
00570157	LEA ECX,DWORD PTR SS:[EBP-8]	
0057015A	MOV EDX,rebuilde.005702DC	ASCII "c_frmAbout_licensed"
0057015F	MOV EAX,rebuilde.005702F8	ASCII "This product is licensed"
00570164	CALL rebuilde.00520B38	

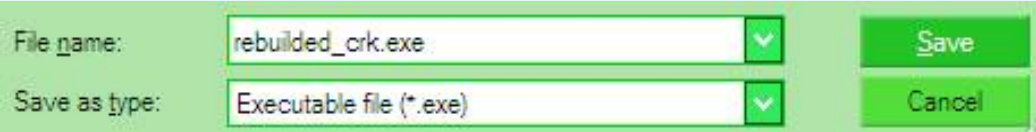
NOP-sent it back to Save a few, Click to:



Lai-Click to continue:



Here per-add a new name, means the hard drive increased 12 MB, actually Damn, per hard drive is only a few hundred megabytes àh.



-Running the new file to do:

XP Tools



The Swiss army knife for your Windows


XP Tools


Version 4.58


For Windows 95/98/ME/NT/2000/XP/2003

Copyright(C) 2004 XPTools.net

This product is licensed to:
Registration Code:

 [Home Page](#)

 [E-Mail to Us](#)

 [Register...](#)

 [Order](#)

Hi-hi, but no one's name and **Registration Code** but even stars still comfortable than anything String count back obnoxious. However, when considered to do the count back, it meant that we have not yet registered nhé, time use one of the **Cuu wifeless About** to go to the future to any of this **is most Khac** (This Dịch Great Mind and Body Hồng bí first)



Set-up 365 white children:



-Run the file from soft **rebuilted_crk.exe**. Then select:



-Dunggggggg A break of the first:

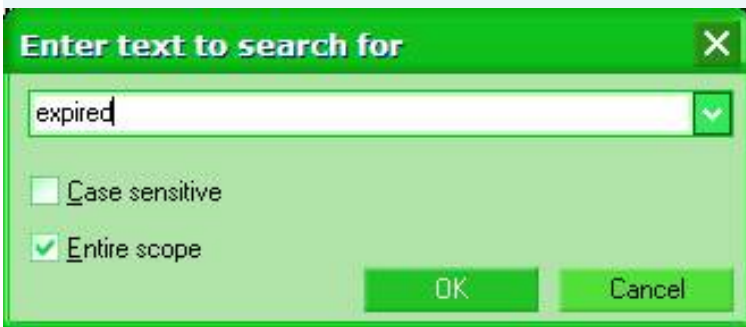


Mo-ko be, try to choose a soft view of Tool, Dunggggggggg:



- "It has not, the idea màỵ màỵ delicious hả Wiuewi well too in the day". Chac sure, do not say Open. Thien ngoai natural property. Are within the same province, again Olly tuốt shell (with **rebuilted_crk file. exe**).

-Okie, NAG and has the task easier as, to some aged Search:



-Tới đây :

```

0056FE03 MOV EDX,rebuild.0056FF68 ASCII "c_frmAbout_msgOutOfDate"
0056FE03 MOV EAX,rebuild.0056FF88 ASCII "Your XPTools has expired. Please purchase it online."
0056FEC4 MOV EDX,rebuild.0056FFCC ASCII "c_PswError"
0056FEC9 MOV EAX,rebuild.0056FFE0 ASCII "Incorrect Password."
0056FE68 ASCII "c_frmAbout_msgOu"

```

-Nhưng Ctrl-L đi tiếp mấy bước coi có nhiều thằng :

```

0057015A MOV EDX,rebuild.0057020C ASCII "c_frmAbout_licensed"
0057015F MOV EAX,rebuild.005702F8 ASCII "This product is licensed to: "
00570176 PUSH rebuild.00570320 ASCII "j"
0057017E MOV EDX,rebuild.0057032C ASCII "c_frmAbout_code"
00570183 MOV EAX,rebuild.00570344 ASCII "Registration Code: "
005701FF MOV EDX,rebuild.00570360 ASCII "c_frmAbout_msgOutOfDate"
00570203 MOV EAX,rebuild.00570380 ASCII "Your XPTools has expired. Please purchase it online."
00570270 MOV EDX,rebuild.005703C0 ASCII "c_frmAbout_prgLeft"
00570275 MOV EAX,rebuild.005703DC ASCII "You have %d days left to try XPTools."
005702DC ASCII "c_frmAbout_licen"

```

-Tiếp:

```

0057398C DD rebuild.00573982 ASCII "ModuleManager"
00573983 ASCII "TModuleManager"
00573CB5 MOV EDX,rebuild.00573FA0 ASCII "c_frmAbout_msgOutOfDate"
00573CC3 MOV EAX,rebuild.00573FC0 ASCII "Your XPTools has expired. Please purchase it online."
00573CF0 PUSH rebuild.00573FFC ASCII "http://www.xptools.net/purchase.htm"
00573CF5 PUSH rebuild.00574020 ASCII "OPEN"
00573D8F MOV ECX,rebuild.00574030 ASCII "ModuleManager: ModuleParent not assigned."
00573EC1 MOV EDX,rebuild.00574064 ASCII "hint_"

```

-DONE !!! , có 3 mạng , nhớ lấy Address của mỗi thằng ,goto tới thằng đầu tiên nào :

```

0056FDEE > 840B TEST BL,BL
0056FDF0 < 75 30 JNZ SHORT rebuild.0056FE22
0056FDF2 . 66:A1 5CFF56 MOV AX,WORD PTR DS:[56FF5C]
0056FDF8 . 50 PUSH EAX
0056FDF9 . 6A 00 PUSH 0
0056FDFB . 804D FC LEA ECX,DWORD PTR SS:[EBP-4]
0056FDFE . BA 68FF5600 MOV EDX,rebuild.0056FF68 ASCII "c_frmAbout_msgOutOfDate"
0056FE03 . B8 88FF5600 MOV EAX,rebuild.0056FF88 ASCII "Your XPTools has expired"
0056FE08 . E8 2BADBFBF CALL rebuild.0052AB38
0056FE0D . 8B55 FC MOV EDI,DWORD PTR SS:[EBP-4]

```

-Ở trên nó kiểm tra quá gì ko bít nhưng bít rằng nếu thỏa thì hiện NAG hết hạn ,ko thỏa thì ko hiện ,vì thế ta cần nó luôn nhảy ở đây ,patch **JNZ** thành **JMP** :

```

0056FDEE > 840B TEST BL,BL
0056FDF0 < EB 30 JMP SHORT rebuild.0056FE22
0056FDF2 . 66:A1 5CFF56 MOV AX,WORD PTR DS:[56FF5C]
0056FDF8 . 50 PUSH EAX
0056FDF9 . 6A 00 PUSH 0
0056FDFB . 804D FC LEA ECX,DWORD PTR SS:[EBP-4]
0056FDFE . BA 68FF5600 MOV EDX,rebuild.0056FF68 ASCII "c_frmAbout_msgOutOfDate"
0056FE03 . B8 88FF5600 MOV EAX,rebuild.0056FF88 ASCII "Your XPTools has expired"
0056FE08 . E8 2BADBFBF CALL rebuild.0052AB38
0056FE0D . 8B55 FC MOV EDI,DWORD PTR SS:[EBP-4]

```

-Goto tới em thứ 2 với address **0570203** :

005701D8	8378 14 00	CMP DWORD PTR DS:[EAX+14],0	
005701DF	7F 3F	JG SHORT rebuild.00570220	
005701E1	33D2	XOR EDX,EDX	
005701E3	8B83 40020000	MOV EAX,DWORD PTR DS:[EBX+240]	
005701E9	E8 6661ECFF	CALL rebuild.00436354	
005701EE	33D2	XOR EDX,EDX	
005701F0	8B83 3C020000	MOV EAX,DWORD PTR DS:[EBX+23C]	
005701F6	E8 5961ECFF	CALL rebuild.00436354	
005701FB	8D40 F0	LEA ECX,DWORD PTR SS:[EBP-10]	
005701FE	BA 60035700	MOV EDX,rebuild.00570360	ASCII "c_frmAbout_msgOutOfDate"
00570203	B8 80035700	MOV EAX,rebuild.00570380	ASCII "Your XPTools has expired"
00570208	E8 2BA9FBFF	CALL rebuild.0052AB38	

-Lại một lệnh so sánh ,nếu lớn hơn thì nhảy luôn ,còn bé hơn hay bằng thì hiện NAG hết hạn ,vậy mình lại muốn nó nhảy ,patch **JG** thành **JMP** :

005701D8	8378 14 00	CMP DWORD PTR DS:[EAX+14],0	
005701DF	EB 3F	JMP SHORT rebuild.00570220	
005701E1	33D2	XOR EDX,EDX	
005701E3	8B83 40020000	MOV EAX,DWORD PTR DS:[EBX+240]	
005701E9	E8 6661ECFF	CALL rebuild.00436354	
005701EE	33D2	XOR EDX,EDX	
005701F0	8B83 3C020000	MOV EAX,DWORD PTR DS:[EBX+23C]	
005701F6	E8 5961ECFF	CALL rebuild.00436354	
005701FB	8D40 F0	LEA ECX,DWORD PTR SS:[EBP-10]	
005701FE	BA 60035700	MOV EDX,rebuild.00570360	ASCII "c_frmAbout_msgOutOfDate"
00570203	B8 80035700	MOV EAX,rebuild.00570380	ASCII "Your XPTools has expired"
00570208	E8 2BA9FBFF	CALL rebuild.0052AB38	
0057020D	8B55 F0	MOV EDX,DWORD PTR SS:[EBP-10]	

-Goto tới cái thứ 3 ,address **0573CC3** :

00573CAC	807D F3 00	CMP BYTE PTR SS:[EBP-0],0	
00573CB0	75 66	JNZ SHORT rebuild.00573D18	
00573CB2	66:A1 943F57	MOV AX,WORD PTR DS:[573F94]	
00573CB8	50	PUSH EAX	
00573CB9	6A 00	PUSH 0	
00573CBB	8D40 E4	LEA ECX,DWORD PTR SS:[EBP-1C]	
00573CBE	BA A03F5700	MOV EDX,rebuild.00573FA0	ASCII "c_frmAbout_msgOutOfDate"
00573CC3	B8 C03F5700	MOV EAX,rebuild.00573FC0	ASCII "Your XPTools has expired"
00573CC8	E8 6B6EFBFF	CALL rebuild.0052AB38	
00573CCD	8B55 E4	MOV EDX,DWORD PTR SS:[EBP-1C]	
00573CD0	A1 48BF6400	MOV EAX,DWORD PTR DS:[64BF48]	

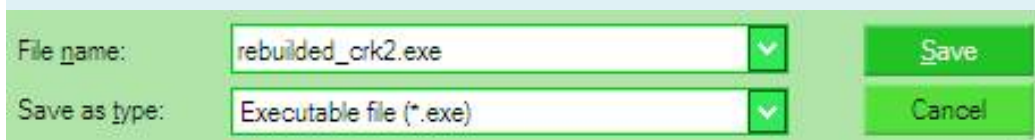
-Suy luận như trên ,patch **JNZ** thành **JMP** :

00573CAC	807D F3 00	CMP BYTE PTR SS:[EBP-0],0	
00573CB0	EB 66	JMP SHORT rebuild.00573D18	
00573CB2	66:A1 943F57	MOV AX,WORD PTR DS:[573F94]	
00573CB8	50	PUSH EAX	
00573CB9	6A 00	PUSH 0	
00573CBB	8D40 E4	LEA ECX,DWORD PTR SS:[EBP-1C]	
00573CBE	BA A03F5700	MOV EDX,rebuild.00573FA0	ASCII "c_frmAbout_msgOutOfDate"
00573CC3	B8 C03F5700	MOV EAX,rebuild.00573FC0	ASCII "Your XPTools has expired"
00573CC8	E8 6B6EFBFF	CALL rebuild.0052AB38	
00573CCD	8B55 E4	MOV EDX,DWORD PTR SS:[EBP-1C]	
00573CD0	A1 48BF6400	MOV EAX,DWORD PTR DS:[64BF48]	

-Patch cho đã thì nhớ SAVE lại dùm mỗ :

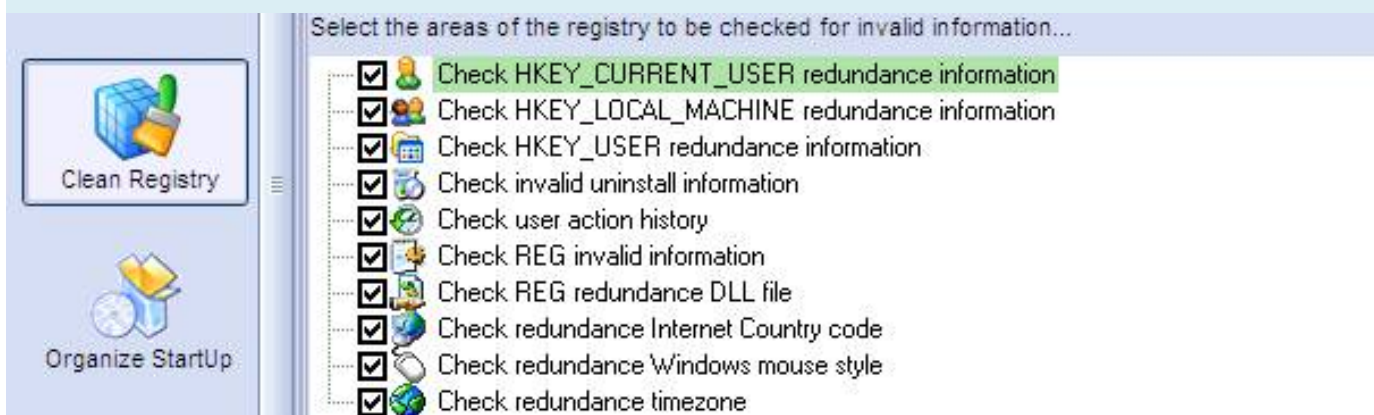


-Thêm một cái tên nào :



-Lại tăng thêm 12MB ,hix ...chạy **rebuilt_crk2.exe** .Nhấn Help -About ko thấy gì hết ,mừng tập 1.Thử thêm một vài Tool của nó:

	Software	Uninstall Command	Size
Manage Windows	Adobe Photoshop 7.0	C:\WINDOWS\ISUNINST.EXE -f"C:\Program F...	142.46 MB
Manage Process	Advanced WMA Workshop version...	"C:\Program Files\LitexMedia\Advanced WMA ...	2.14 MB
Protect IE	Advanced Uninstaller PRO 2005 - v...	"C:\Program Files\Advanced Uninstaller PRO 2...	20.18 MB
XP Uninstaller	AVI/MPEG/ASF/WMV Splitter 3.25	"C:\Program Files\AVI MPEG ASF WMV Splitter...	1.30 MB
	AVI/MPEG/RM/WMV Joiner 4.81	"C:\Program Files\AVI MPEG RM WMV Joiner\...	5.64 MB
	Big Money Deluxe 1.11	C:\Program Files\PopCap Games\Big Money De...	12.87 MB
	BookWorm Deluxe 1.0	C:\Program Files\PopCap Games\BookWorm D...	12.07 MB
	CD Speed	C:\PROGRA~1\CDSPEE~1\UNWISE.EXE C:\...	0.60 MB
	CloneCD	"C:\Program Files\SlySoft\CloneCD\ccd-uninst...	4.96 MB
	Cucusoft MPEG/AVI to DVD/VCD/...	"C:\Program Files\Avi-Dvd-Pro\unins000.exe"	6.70 MB
	DiaryOne 5.5	"C:\Program Files\DiaryOne\unins000.exe"	unkown
	Dynomite 2.01	C:\Program Files\PopCap Games\Dynomite\Un...	5.01 MB
	Enable S3 for USB Device	C:\WINDOWS\IsUninst.exe -f"C:\Program Files...	0.02 MB
	File Recover 5.0	"C:\Program Files\File Recover\unins000.exe"	unkown
	Font Unicode	C:\WINDOWS\unvise32.exe C:\WINDOWS\F...	unkown
	GoldWave v5.10	"C:\Program Files\GoldWave\uninstall.exe" "Gold...	7.27 MB
	Heroes of Might and Magic® IV	C:\WINDOWS\IsUninst.exe -f"C:\Program Files...	726.19 MB
	IconPackager	C:\PROGRA~1\STARDOCK\OBJECT~1\ICON...	306.70 MB
	Corel Graphics Suite 11	C:\PROGRA~1\COMMON~1\INSTAL~1\Drive	190.13 MB



-Mừng tập 2 rồi ha .Coi như tới đây đã xong quá trình [Patch to use soft without register](#) .

-Nhưng sự đòi hỏi khi ko đơn giản ,giả như một cao thủ ngộ ra thức nào đó thì lại muốn ngộ tiếp tăng cao hơn .Mỗi nhìn kĩ lại :

This product is licensed to: **NULL**
Registration Code: **NULL**

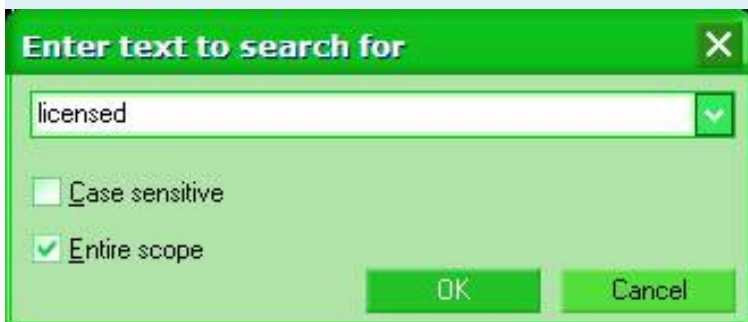
-Nãy giờ 100 mấy chục bức hình rồi mà cũng thế ,sao đòi lắm công gai ,còn lòng mỗ thì lại đầy ham muốn .Mỗ muốn trong soft có tên mình cơ ,có thêm Serial càng tốt .Tới đây ,mỗ cố nhét tên mình vào ,nhưng hậu quả là khi run soft ,nó bỏ qua các name mặc định thêm vào của các Tool ,chỉ để tên mỗ ,nên coi như thất bại :



Okie ,giờ mỡ rắng luyện lên tăng cao hơn nhá :

2/Find Real Serial (or Registration Code) :

-Thật ra ra ko bít là có tìm được Real Serial hay ko ,nhưng với cái kiểu tìm String dễ như trên thì ta cố tiếp xem sao .Mở file được Update mới nhất lên , **rebuilted_crk2.exe** .Dò với String :



```

0057015A MOV EDX,rebuilted.005702DC ASCII "c_frmAbout_licensed"
0057015F MOV EAX,rebuilted.005702F8 ASCII "This product is licensed to: "
00570176 PUSH rebuilted.00570320 ASCII " "
0057017E MOV EDX,rebuilted.0057032C ASCII "c_frmAbout_code"
00570183 MOV EAX,rebuilted.00570344 ASCII "Registration Code: "
005701FE MOV EDX,rebuilted.00570360 ASCII "c_frmAbout_msgOutOfDate"
00570203 MOV EAX,rebuilted.00570380 ASCII "Your XPTools has expired. Please purchase it online."
00570270 MOV EDX,rebuilted.005703C0 ASCII "c_frmAbout_prgLeft"
00570275 MOV EAX,rebuilted.005703DC ASCII "You have %d days left to try XPTools."
00570283 MOV EAX,rebuilted.005703E0 ASCII " "

```

-Enter nó nào :

```

00570142 . 84C0 TEST AL,AL
00570144 . 90 NOP
00570145 . 90 NOP
00570146 . 90 NOP
00570147 . 90 NOP
00570148 . 90 NOP
00570149 . 90 NOP
0057014A . B2 01 MOV DL,1
0057014C . 8B83 28020000 MOV EAX,DWORD PTR DS:[EBX+228]
00570152 . E8 FD61ECFF CALL rebuilde.00436354
00570157 . 8D4D F8 LEA ECX,DWORD PTR SS:[EBP-8]
0057015A . BA DC025700 MOV EDX,rebuilde.005702DC
0057015F . B8 F8025700 MOV EAX,rebuilde.005702F8
00570164 . E8 CE99EBFF CALL rebuilde.00520B38
ASCII "c_frmAbout_licensed"
ASCII "This product is licensed to: "

```

-Giờ ta kéo lên đầu hàm này xem trước khi nhảy thì nó được gọi từ đâu :

```

00570113 . 00 DB 00
00570114 . C3 RETN
00570115 . 8D40 00 LEA EAX,DWORD PTR DS:[EAX]
00570118 $ 55 PUSH EBP
00570119 . 8BEC MOV EBP,ESP
0057011B . 33C9 XOR ECX,ECX
0057011D . 51 PUSH ECX
0057011E . 51 PUSH ECX
0057011F . 51 PUSH ECX
00570120 . 51 PUSH ECX
00570121 . 51 PUSH ECX
00570122 . 51 PUSH ECX
00570123 . 51 PUSH ECX
00570124 . 51 PUSH ECX
00570125 . 53 PUSH EBX
00570126 . 8BD8 MOV EBX,EAX
00570128 . 33C0 XOR EAX,EAX
0057012A . 55 PUSH EBP
0057012B . 68 C7025700 PUSH rebuilde.005702C7
00570130 . 64:FF30 PUSH DWORD PTR FS:[EAX]
00570133 . 64:8920 MOV DWORD PTR FS:[EAX],ESP
00570136 . A1 DCBA6400 MOV EAX,DWORD PTR DS:[64BADC]
00570138 . 8B00 MOV EAX,DWORD PTR DS:[EAX]
0057013D . E8 8672FDFF CALL rebuilde.005473C8
00570142 . 84C0 TEST AL,AL
00570144 . 90 NOP

```

-Ở đây có 3 lời gọi:

```
Local calls from 0056FFF8, 00570611, 0057763D
```

-Chư vị có thể goto tới từng thằng Call này ,BP nó ,rồi run soft ,Register với Fake Serial xem nó Break tại đâu .Ở đây mỗi thử trước dùm các vị nên ko cần làm như trên ,ta goto tới em này luôn:

```

Local calls from 0056FFF8, 00570611, 0057763D
Copy pane to clipboard
Go to CALL from 0056FFF8
Go to CALL from 00570611
Go to CALL from 0057763D

```

Address	Value	Comment
00ED8000	7E091B00	6DI32.S
00ED8004	7E093B08	6DI32.E
00ED8008	7E0939C9	6DI32.D
00ED800C	7E095AD6	6DI32.C

-Tôi :

```

00570611 . E8 02FBFFFF CALL rebuilde.00570118
00570616 . 33C0 XOR EAX,EAX
00570618 . 5A POP EDX
00570619 . 59 POP ECX
0057061A . 58 POP ECX

```

-Xem trước khi gọi thằng nì thì nó lại làm gì ở trên ,kéo lên nào , chaaaaaaaaa:

005705AF	. 84C0	TEST AL,AL	
005705B1	74 4B	JE SHORT rebuilde.005705FE	
005705B3	66:A1 4C0657	MOV AX,WORD PTR DS:[57064C]	
005705B9	50	PUSH EAX	
005705BA	6A 00	PUSH 0	
005705BC	804D E8	LEA ECX,DWORD PTR SS:[EBP-18]	
005705BF	BA 58065700	MOV EDX,rebuilde.00570658	ASCII "C_FrmAbout_lbThanks"
005705C4	B8 74065700	MOV EAX,rebuilde.00570674	ASCII "We sincerely appreciate your s
005705C9	E8 6AA5FBFF	CALL rebuilde.0052AB38	
005705CE	8B55 E8	MOV EDX,DWORD PTR SS:[EBP-18]	
005705D1	B1 02	MOV CL,2	
005705D3	8B86 0C02000	MOV EAX,DWORD PTR DS:[ESI+20C]	
005705D9	E8 4687FDFF	CALL rebuilde.00548D24	
005705DE	6A 00	PUSH 0	

-Chà ,đang kéo lên tự nhiên thấy nàng ấy àh .Đề ý lệnh **JE** đằng trước nó .Còn cái chuỗi bít nó là gì ko nào :

```
ASCII "C_FrmAbout_lbThanks"
ASCII "We sincerely appreciate your support of XPTools. Now you become a legal user of XPTools."
```

-Vậy là nếu bằng thì nhảy luôn ,còn ko thì hiện NAG chúc mừng đăng kí thành công .Patch thẳng **JE** lại hả ,cho mỗi lần nhấn Register đều được chúc mừng sao ?Ày ,nên nhớ ta đang luyện lên tầng cao hơn ,ko làm kiểu Sub_Pro đó nữa .Thường trước khi nhảy ,thì nó phải gọi hàm nào đó để kiểm tra các điều kiện ,và thường thì lệnh **CALL** gần **nhất** chính là lời gọi quan trọng sau cùng .Bít đâu nó gọi tới hàm tính Real Serial thì sao?Đặt BP lên nó :

005705A7	. 84C0	TEST AL,AL	
005705AA	E8 656FFDFF	CALL rebuilde.00547514	
005705AF	. 84C0	TEST AL,AL	
005705B1	74 4B	JE SHORT rebuilde.005705FE	
005705B3	66:A1 4C0657	MOV AX,WORD PTR DS:[57064C]	
005705B9	50	PUSH EAX	
005705BA	6A 00	PUSH 0	
005705BC	804D E8	LEA ECX,DWORD PTR SS:[EBP-18]	
005705BF	BA 58065700	MOV EDX,rebuilde.00570658	ASCII "C_FrmAbout_lbThanks"
005705C4	B8 74065700	MOV EAX,rebuilde.00570674	ASCII "We sincerely appreciate your s
005705C9	E8 6AA5FBFF	CALL rebuilde.0052AB38	

-F9 run soft ,chiều thức như cũ :

Register

Registration Name:

Registration Code: - - -

-Nhấn Register xem ,chờ à ..giờ cha nội mới chịu break hả ,sao làm khổ đời con thế :

005705AA	. E8 656FFDFF	CALL rebuilde.00547514	
005705AF	. 84C0	TEST AL,AL	
005705B1	74 4B	JE SHORT rebuilde.005705FE	
005705B3	66:A1 4C0657	MOV AX,WORD PTR DS:[57064C]	
005705B9	50	PUSH EAX	
005705BA	6A 00	PUSH 0	
005705BC	804D E8	LEA ECX,DWORD PTR SS:[EBP-18]	
005705BF	BA 58065700	MOV EDX,rebuilde.00570658	ASCII "C_FrmAbout_lbThanks"
005705C4	B8 74065700	MOV EAX,rebuilde.00570674	ASCII "We sincerely appreciate your s

-Còn đợi gì nữa ,Trace F7 vào lời gọi CALL này:

00547514	\$ 55	PUSH EBP
00547515	. 8BEC	MOV EBP,ESP
00547517	. 83C4 F0	ADD ESP,-10
0054751A	. 53	PUSH EBX
0054751B	. 894D F8	MOV DWORD PTR SS:[EBP-8],ECX
0054751E	. 8955 FC	MOV DWORD PTR SS:[EBP-4],EDX
00547521	. 8B08	MOV EBX,EBX

-Ở đây mỗi cũng Trace muốn không luôn ,nên cho mấy vị bit trước luôn ,Trace F8 tới lệnh **CALL thứ 3** :

```

00547514 $ 55 PUSH EBP
00547515 . 8BEC MOV EBP,ESP
00547517 . 83C4 F0 ADD ESP,-10
0054751A . 53 PUSH EBX
0054751B . 894D F8 MOV DWORD PTR SS:[EBP-8],ECX
0054751E . 8955 FC MOV DWORD PTR SS:[EBP-4],EDX
00547521 . 8B08 MOV EBX,EAX
00547523 . 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
00547526 . E8 45CFEBFF CALL rebuilde.00404470
0054752B . 8B45 F8 MOV EAX,DWORD PTR SS:[EBP-8]
0054752E . E8 30CFEBFF CALL rebuilde.00404470
00547533 . 33C0 XOR EAX,EAX
00547535 . 55 PUSH EBP
00547536 . 68 FF755400 PUSH rebuilde.005475FF
0054753B . 64:FF30 PUSH DWORD PTR FS:[EAX]
0054753E . 64:8920 MOV DWORD PTR FS:[EAX],ESP
00547541 . 8B4D F8 MOV ECX,DWORD PTR SS:[EBP-8]
00547544 . 8B55 FC MOV EDX,DWORD PTR SS:[EBP-4]
00547547 . 8BC3 MOV EAX,EBX
00547549 . E8 D6030000 CALL rebuilde.00547924
0054754E . 8B45 F7 MOV BYTE PTR SS:[EBP-9],AL
00547551 . 807D F7 00 CMP BYTE PTR SS:[EBP-9],0

```

-Tại sao lại là lệnh CALL này ,để ý trong quá trình Trace ,thấy trước lệnh CALL có 2 lệnh MOV vào ECX ,EDX từ 2 vùng nhớ là **[EBP-8]** , **[EBP-4]** .Mà 2 vùng nhớ này nhiều khả năng là Name và Fake Serial của chúng ta .Quả đúng thế :

```

Registers (FPU)
EAX 03694C90 ASCII "xqt"
ECX 03700F8C ASCII "1111-2222-3333-4444"
EDX 0370246C ASCII "Trickyboy"
EBX 03694C90 ASCII "xqt"
ESP 0012FAC4
EBP 0012FAE4
ESI 036E2C88
EDI 0012FD04
EIP 00547549 rebuilde.00547549

```

-Hừm đưa đã vào 2 thanh ghi thì phải làm gì với tụi nó chứ.Rồi Trace F7 vào nào :

```

00547924 $ 55 PUSH EBP
00547925 . 8BEC MOV EBP,ESP
00547927 . 51 PUSH ECX
00547928 . B9 0F000000 MOV ECX,0F
0054792D > 6A 00 PUSH 0
0054792F . 6A 00 PUSH 0
00547931 . 49 DEC ECX
00547932 . 75 F9 JNZ SHORT rebuilde.0054792D
00547934 . 51 PUSH ECX
00547935 . 874D FC XCHG DWORD PTR SS:[EBP-4],ECX

```

-Trace F8 xuống cho đã đi nào ,cho tới khi thấy cái chuỗi **"XPTools"**

```

00547A38 > 6A 10 PUSH 10
00547A3A . 8D45 D0 LEA EAX,DWORD PTR SS:[EBP-30]
00547A3D . 50 PUSH EAX
00547A3E . 8D45 C8 LEA EAX,DWORD PTR SS:[EBP-38]
00547A41 . B9 807D5400 MOV ECX,rebuilde.00547D80
00547A46 . 8B55 FC MOV EDX,DWORD PTR SS:[EBP-4]
00547A48 . E8 80030000 CALL rebuilde.00404300

```

ASCII "XPTools"

-Trace F8 wa nó tới đây xem:

```

00547A38 > 6A 10 PUSH 10
00547A3A . 8D45 D0 LEA EAX,DWORD PTR SS:[EBP-30]
00547A3D . 50 PUSH EAX
00547A3E . 8D45 C8 LEA EAX,DWORD PTR SS:[EBP-38]
00547A41 . B9 807D5400 MOV ECX,rebuilde.00547D80
00547A46 . 8B55 FC MOV EDX,DWORD PTR SS:[EBP-4]
00547A49 . E8 80030000 CALL rebuilde.00404300
00547A4E . 8B45 C8 MOV EAX,DWORD PTR SS:[EBP-38]
00547A51 . 8D55 CC LEA EDI,DWORD PTR SS:[EBP-34]

```

ASCII "XPTools"

```
Registers (FPU)
EAX 0012FA84
ECX 00547D80 ASCII "XPTools"
EDX 0370246C ASCII "Trickyboy"
EBX FFFFFFFF
ESP 0012FA10
EBP 0012FABC
ESI 036E2C88
EDI 0012FD04
EIP 00547A49 rebuilde.00547A49
```

-Theo một số soft thông thường ,coi mồi nó sắp nối Name và chuỗi mặc định này lại lằm à .Trace tiếp :

00547A38	> 6A 10	PUSH 10	
00547A3A	. 8045 D0	LEA EAX,DWORD PTR SS:[EBP-30]	
00547A3D	. 50	PUSH EAX	
00547A3E	. 8045 C8	LEA EAX,DWORD PTR SS:[EBP-38]	
00547A41	. B9 807D5400	MOV ECX,rebuilde.00547D80	ASCII "XPTools"
00547A46	. 8B55 FC	MOV EDX,DWORD PTR SS:[EBP-4]	
00547A49	. E8 BAC8EBFF	CALL rebuilde.00404308	
00547A4E	. 8B45 C8	MOV EAX,DWORD PTR SS:[EBP-38]	
00547A51	. 8D55 CC	LEA EDX,DWORD PTR SS:[EBP-34]	
00547A54	. E8 4726ECFF	CALL rebuilde.0040A0A0	
00547A59	. 8B55 CC	MOV EDX,DWORD PTR SS:[EBP-34]	

-Quả đúng ...KINH :

```
Registers (FPU)
EAX 036E950C ASCII "TrickyboyXPTools"
ECX 00000000
EDX 036E9515 ASCII "XPTools"
EBX FFFFFFFF
ESP 0012FA10
EBP 0012FABC
ESI 036E2C88
EDI 0012FD04
EIP 00547A51 rebuilde.00547A51
```

-Vậy nó mã hóa tiếp như thế nào ,đi tiếp mấy vị ới :

00547A38	> 6A 10	PUSH 10	
00547A3A	. 8045 D0	LEA EAX,DWORD PTR SS:[EBP-30]	
00547A3D	. 50	PUSH EAX	
00547A3E	. 8045 C8	LEA EAX,DWORD PTR SS:[EBP-38]	
00547A41	. B9 807D5400	MOV ECX,rebuilde.00547D80	ASCII "XPTools"
00547A46	. 8B55 FC	MOV EDX,DWORD PTR SS:[EBP-4]	
00547A49	. E8 BAC8EBFF	CALL rebuilde.00404308	
00547A4E	. 8B45 C8	MOV EAX,DWORD PTR SS:[EBP-38]	
00547A51	. 8D55 CC	LEA EDX,DWORD PTR SS:[EBP-34]	
00547A54	. E8 4726ECFF	CALL rebuilde.0040A0A0	
00547A59	. 8B55 CC	MOV EDX,DWORD PTR SS:[EBP-34]	
00547A5C	. 33C9	XOR ECX,ECX	
00547A5E	. A1 F8795300	MOV EAX,DWORD PTR DS:[5379F8]	
00547A63	. E8 9811FFFF	CALL rebuilde.00538C00	

-Chà tới đó thì nó viết hoa lên đòi ,ra là rứa :

```
Registers (FPU)
EAX 0012FA53
ECX 036E9524
EDX 036E952C ASCII "TRICKYBOYXPTOOLS"
EBX FFFFFFFF
ESP 0012FA10
EBP 0012FABC
ESI 036E2C88
EDI 0012FD04
EIP 00547A5C rebuilde.00547A5C
```

-Lại lết từng bước nặng nhọc ,kiên nhẫn lên nào ,sắp đạt được cái gì đó rồi :

00547A3E	. 8045 C8	LEA EAX,DWORD PTR SS:[EBP-38]	
00547A41	. B9 807D5400	MOV ECX,rebuild.00547D80	ASCII "XPTools"
00547A46	. 8B55 FC	MOV EDX,DWORD PTR SS:[EBP-4]	
00547A49	. E8 BAC8EBFF	CALL rebuild.00404308	
00547A4E	. 8B45 C8	MOV EAX,DWORD PTR SS:[EBP-38]	
00547A51	. 8D55 CC	LEA EDX,DWORD PTR SS:[EBP-34]	
00547A54	. E8 4726ECFF	CALL rebuild.0040A0A0	
00547A59	. 8B55 CC	MOV EDX,DWORD PTR SS:[EBP-34]	
00547A5C	. 33C9	XOR ECX,ECX	
00547A5E	. A1 F8795300	MOV EAX,DWORD PTR DS:[5379F8]	
00547A63	. E8 9811FFFF	CALL rebuild.00538C00	
00547A68	. 8B45 D0	MOV EAX,DWORD PTR SS:[EBP-30]	
00547A6B	. 8D55 F0	LEA EDX,DWORD PTR SS:[EBP-10]	
00547A6E	. E8 85FDFFFF	CALL rebuild.005477F8	
00547A73	. 8045 C4	LEA EAX,DWORD PTR SS:[EBP-3C]	
00547A76	. 50	PUSH EAX	

-Chà ,đúng là có chuyện lạ :

Registers (FPU)		
EAX	036C8440	ASCII "3CFE6353E19FF7ABF9AD5055C423A2ED"
ECX	036E9544	
EDX	00640438	rebuild.00640438
EBX	FFFFFFFF	
ESP	0012FA18	
EBP	0012FABC	
ESI	036E2C88	
EDI	0012FD04	
EIP	00547A6B	rebuild.00547A6B

-Thế là từ chuỗi viết hoa trên ,nó cho ra một lô số đề dài thông lòng ,đoán một xiu ,thôi thì Trace tiếp :

00547A30	. 50	PUSH EAX	
00547A3E	. 8045 C8	LEA EAX,DWORD PTR SS:[EBP-38]	
00547A41	. B9 807D5400	MOV ECX,rebuild.00547D80	ASCII "XPTools"
00547A46	. 8B55 FC	MOV EDX,DWORD PTR SS:[EBP-4]	
00547A49	. E8 BAC8EBFF	CALL rebuild.00404308	
00547A4E	. 8B45 C8	MOV EAX,DWORD PTR SS:[EBP-38]	
00547A51	. 8D55 CC	LEA EDX,DWORD PTR SS:[EBP-34]	
00547A54	. E8 4726ECFF	CALL rebuild.0040A0A0	
00547A59	. 8B55 CC	MOV EDX,DWORD PTR SS:[EBP-34]	
00547A5C	. 33C9	XOR ECX,ECX	
00547A5E	. A1 F8795300	MOV EAX,DWORD PTR DS:[5379F8]	
00547A63	. E8 9811FFFF	CALL rebuild.00538C00	
00547A68	. 8B45 D0	MOV EAX,DWORD PTR SS:[EBP-30]	
00547A6B	. 8D55 F0	LEA EDX,DWORD PTR SS:[EBP-10]	
00547A6E	. E8 85FDFFFF	CALL rebuild.005477F8	
00547A73	. 8045 C4	LEA EAX,DWORD PTR SS:[EBP-3C]	
00547A76	. 50	PUSH EAX	

-Trace một phát tới đó rồi ngưng chur vị ơ ,phải coi nó làm quái gì với chuỗi đó chứ ,đây có thể là lệnh **CALL** gọi hàm mã hóa lô số đề này ,Trace F7 vào nó :

005477F8	. 55	PUSH EBP	
005477F9	. 8BEC	MOV EBP,ESP	
005477FB	. 83C4 E8	ADD ESP,-18	
005477FE	. 53	PUSH EBX	
005477FF	. 56	PUSH ESI	
00547800	. 57	PUSH EDI	

-Rồi Trace F8 như ngựa tới đây ,nhanh lên mấy nị ,chậm chân cúp điện là toi công :

005478A6	. 52	PUSH EDX	
005478A7	. 50	PUSH EAX	Arg2
005478A8	. 8D55 F4	LEA EDX,DWORD PTR SS:[EBP-C]	Arg1
005478AB	. B8 01000000	MOV EAX,1	
005478B0	. E8 7F2DECFF	CALL rebuild.0040A634	rebuild.0040A634
005478B5	. 8B45 F4	MOV EAX,DWORD PTR SS:[EBP-C]	
005478B8	. FF30	PUSH DWORD PTR DS:[EAX]	
005478BA	. 8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]	
005478BD	. E8 CAC8EBFF	CALL rebuild.0040448C	
005478C2	. 5A	POP EDX	
005478C3	. 8B5438 FF	MOV BYTE PTR DS:[EAX+EDI-1],DL	
005478C7	. 47	INC EDI	
005478C8	. 4E	DEC ESI	
005478C9	. 0F85 2BFFFFFF	JNZ rebuild.005478A0	

-Trace thêm một cái ,thấy ổng giảm kí tự đầu tiên của lô số đề đó đi 1 giá trị :

```

Registers (FPU)
EAX 036E8A54 ASCII "2CFE6353E19FF7ABF9AD5055C423A2ED"
ECX 00000000
EDX 036F0032 ASCII "mss.exe"
EBX FFFFFFF3
ESP 0012F9D4
EBP 0012FA10
ESI 00000020
EDI 00000001
EIP 005478C7 rebuilde.005478C7

```

-Giờ Trace tiếp thì gặp lệnh **JNZ** nhảy lùi về ,vậy set một bp ngay lệnh **JNZ** này (chắc ăn thì set thêm một bp ngay dưới lệnh **JNZ** vì ko biết chừng nào thoát ra vòng lặp này nữa) :

```

005478B8 . 8D45 FC LEA EAX,DWORD PTR SS:[EBP-4]
005478BD . E8 CACBEBFF CALL rebuilde.0040448C
005478C2 . 5A POP EDX
005478C3 . 8B5438 FF MOV BYTE PTR DS:[EAX+EDI-1],DL
005478C7 > 47 INC EDI
005478C8 . 4E DEC ESI
005478C9 . ^ 0F85 7BFFFFFF JNZ rebuilde.0054784A
005478CE > 8B45 F8 MOV EAX,DWORD PTR SS:[EBP-8]

```

```

005478B8 . FF30 PUSH DWORD PTR DS:[EAX]
005478BA . 8D45 FC LEA EAX,DWORD PTR SS:[EBP-4]
005478BD . E8 CACBEBFF CALL rebuilde.0040448C
005478C2 . 5A POP EDX
005478C3 . 8B5438 FF MOV BYTE PTR DS:[EAX+EDI-1],DL
005478C7 > 47 INC EDI
005478C8 . 4E DEC ESI
005478C9 . ^ 0F85 7BFFFFFF JNZ rebuilde.0054784A
005478CE > 8B45 F8 MOV EAX,DWORD PTR SS:[EBP-8]

```

-Giờ nhấn F9 ,nó break tại bp ngay **JNZ** ,lúc này thì *kí tự kế tiếp bị giảm xuống 1 giá trị* ,và cứ thế nhấn F9 , *từng kí tự* của lô số đề giảm ào ào đi 1 giá trị (giá xăng giảm thế thì khỏe hi?) .Cho tới kí tự cuối cùng ,từ từ nào .F9 một cái ...àooooo :

```

Registers (FPU)
EAX 036B1168 ASCII "E5E5B15DC6A00D129BECB7E4A48F4454"
ECX 00000000
EDX 036B0045
EBX FFFFFFF46
ESP 0012F9D4
EBP 0012FA10
ESI 0000001F
EDI 00000002

```

-Bây giờ cũng break nhưng lại cho ra một lô số đề mới ,cũng lại nhấn F9 đi vù vù ,lại giảm như điên ,cho tới khi ra thêm *lô số đề thứ 3* :

```

Registers (FPU)
EAX 036F2338 ASCII "FED904A61A695501D7860AADB2FESDCC"
ECX 00000000
EDX 036F2338 ASCII "FED904A61A695501D7860AADB2FESDCC"
EBX FFFFFFF30
ESP 0012F9D4

```

-Thêm một lần đề luôn F9 ,giảm zùn zụt (hi vọng là lần cuối) ,đến kí tự cuối cùng ,chưa set bp nào ở dưới thì set dùm mỗ ,ở đây mô set lên lệnh **CALL** ,rồi cho wa kí tự cuối luôn nào ,break :

```

005478C8 . 4E DEC ESI
005478C9 . ^ 0F85 7BFFFFFF JNZ rebuilde.0054784A
005478CF > 8B45 F8 MOV EAX,DWORD PTR SS:[EBP-8]
005478D2 . 8B55 FC MOV EDX,DWORD PTR SS:[EBP-4]
005478D5 . E8 B6C7EBFF CALL rebuilde.00404090
005478DA . 33C0 XOR EAX,EAX

```

-À đây là lô cuối cùng sau khi hội đồng vé số dùng tích phân suy rộng ,nội suy La-Răng để tính ra :

```

Registers (FPU)
EAX 0012FAAC
ECX 00000000
EDX 036F2338 ASCII "FDC8F395095844F0C675F99CA1ED4CBB"
EBX FFFFFFF43
ESP 0012F9D4

```


-Cứ từ từ mà Trace F8 đi nào .cho tới khi nó **RETN** 2 lần ,đá ta văng ra ngoài .Rồi ta lại Trace F8 tiếp ,cho tới :

```
00547C67 . E8 5DC8EBFF CALL rebuild.004044C4
00547C67 . FF75 8C    PUSH DWORD PTR SS:[EBP-74]
00547C6A . 68 907D5400 PUSH rebuild.00547D90
00547C6F . 8D45 88    LEA EAX,DWORD PTR SS:[EBP-78]
00547C72 . 50        PUSH EAX
00547C73 . B9 04000000 MOV ECX,4
00547C78 . BA 05000000 MOV EDX,5
00547C7D . 8B45 F0    MOV EAX,DWORD PTR SS:[EBP-10]
00547C80 . E8 3FC8EBFF CALL rebuild.004044C4
00547C85 . FF75 88    PUSH DWORD PTR SS:[EBP-78]
00547C88 . 68 907D5400 PUSH rebuild.00547D90
00547C8D . 8D45 84    LEA EAX,DWORD PTR SS:[EBP-7C]
00547C90 . 50        PUSH EAX
00547C91 . B9 04000000 MOV ECX,4
00547C96 . BA 09000000 MOV EDX,9
00547C9B . 8B45 F0    MOV EAX,DWORD PTR SS:[EBP-10]
00547C9E . E8 21C8EBFF CALL rebuild.004044C4
00547CA3 . FF75 84    PUSH DWORD PTR SS:[EBP-7C]
00547CA6 . 68 907D5400 PUSH rebuild.00547D90
00547CAB . 8D45 80    LEA EAX,DWORD PTR SS:[EBP-80]
00547CAF . 50        PUSH EAX
00547CB4 . B9 04000000 MOV ECX,4
00547CB9 . BA 0D000000 MOV EDX,0D
00547CBF . 8B45 F0    MOV EAX,DWORD PTR SS:[EBP-10]
00547CC0 . E8 03C8EBFF CALL rebuild.004044C4
00547CC1 . FF75 80    PUSH DWORD PTR SS:[EBP-80]
00547CC4 . 8D45 EC    LEA EAX,DWORD PTR SS:[EBP-14]
00547CC7 . BA 07000000 MOV EDX,7
00547CCC . E8 ABC6EBFF CALL rebuild.0040437C
00547CD1 . 8B45 F8    MOV EAX,DWORD PTR SS:[EBP-8]
00547CD4 . 8B55 EC    MOV EDX,DWORD PTR SS:[EBP-14]
00547CD7 . E8 F0C6EBFF CALL rebuild.004043CC
00547CE1 . 75 24     JNZ SHORT rebuild.00547D02
00547CE2 . C645 F7 01 MOV BYTE PTR SS:[EBP-9],1
00547CE2 . 6A 00     PUSH 0
00547CE4 . BA B07D5400 MOV EDX,rebuild.00547DB0
00547CE9 . B9 9C7D5400 MOV ECX,rebuild.00547D9C
00547CEE . B8 01000000 MOV EAX,80000001
00547CF3 . E8 80E5FFFF CALL rebuild.00546278
Stack SS:[0012FA48]=03702EEC, (ASCII "FDC8")
```

-Một nhóm gồm 4 số đầu tiên trong cái lô đề cuối cùng trên .Nhớ lấy ,Trace tiếp :

```
00547C85 . FF75 88    PUSH DWORD PTR SS:[EBP-78]
00547C88 . 68 907D5400 PUSH rebuild.00547D90
00547C8D . 8D45 84    LEA EAX,DWORD PTR SS:[EBP-7C]
00547C90 . 50        PUSH EAX
00547C91 . B9 04000000 MOV ECX,4
00547C96 . BA 09000000 MOV EDX,9
00547C9B . 8B45 F0    MOV EAX,DWORD PTR SS:[EBP-10]
00547C9E . E8 21C8EBFF CALL rebuild.004044C4
00547CA3 . FF75 84    PUSH DWORD PTR SS:[EBP-7C]
00547CA6 . 68 907D5400 PUSH rebuild.00547D90
00547CAB . 8D45 80    LEA EAX,DWORD PTR SS:[EBP-80]
00547CAF . 50        PUSH EAX
00547CB4 . B9 04000000 MOV ECX,4
00547CB9 . BA 0D000000 MOV EDX,0D
00547CBF . 8B45 F0    MOV EAX,DWORD PTR SS:[EBP-10]
00547CC0 . E8 03C8EBFF CALL rebuild.004044C4
00547CC1 . FF75 80    PUSH DWORD PTR SS:[EBP-80]
00547CC4 . 8D45 EC    LEA EAX,DWORD PTR SS:[EBP-14]
00547CC7 . BA 07000000 MOV EDX,7
00547CCC . E8 ABC6EBFF CALL rebuild.0040437C
00547CD1 . 8B45 F8    MOV EAX,DWORD PTR SS:[EBP-8]
00547CD4 . 8B55 EC    MOV EDX,DWORD PTR SS:[EBP-14]
00547CD7 . E8 F0C6EBFF CALL rebuild.004043CC
00547CE1 . 75 24     JNZ SHORT rebuild.00547D02
00547CE2 . C645 F7 01 MOV BYTE PTR SS:[EBP-9],1
00547CE2 . 6A 00     PUSH 0
00547CE4 . BA B07D5400 MOV EDX,rebuild.00547DB0
00547CE9 . B9 9C7D5400 MOV ECX,rebuild.00547D9C
00547CEE . B8 01000000 MOV EAX,80000001
00547CF3 . E8 80E5FFFF CALL rebuild.00546278
Stack SS:[0012FA44]=037093F4, (ASCII "F395")
```

- 4 số kế tiếp trong lô trên ,tiếp tiếp:

```
00547C9E . E8 21C8EBFF CALL rebuild.004044C4
00547CA3 . FF75 84    PUSH DWORD PTR SS:[EBP-7C]
00547CA6 . 68 907D5400 PUSH rebuild.00547D90
00547CAB . 8D45 80    LEA EAX,DWORD PTR SS:[EBP-80]
00547CAE . 50        PUSH EAX
00547CAF . B9 04000000 MOV ECX,4
00547CB4 . BA 0D000000 MOV EDX,0D
00547CBF . 8B45 F0    MOV EAX,DWORD PTR SS:[EBP-10]
00547CC0 . E8 03C8EBFF CALL rebuild.004044C4
00547CC1 . FF75 80    PUSH DWORD PTR SS:[EBP-80]
00547CC4 . 8D45 EC    LEA EAX,DWORD PTR SS:[EBP-14]
00547CC7 . BA 07000000 MOV EDX,7
00547CCC . E8 ABC6EBFF CALL rebuild.0040437C
00547CD1 . 8B45 F8    MOV EAX,DWORD PTR SS:[EBP-8]
00547CD4 . 8B55 EC    MOV EDX,DWORD PTR SS:[EBP-14]
00547CD7 . E8 F0C6EBFF CALL rebuild.004043CC
00547CE1 . 75 24     JNZ SHORT rebuild.00547D02
00547CE2 . C645 F7 01 MOV BYTE PTR SS:[EBP-9],1
00547CE2 . 6A 00     PUSH 0
00547CE4 . BA B07D5400 MOV EDX,rebuild.00547DB0
00547CE9 . B9 9C7D5400 MOV ECX,rebuild.00547D9C
00547CEE . B8 01000000 MOV EAX,80000001
00547CF3 . E8 80E5FFFF CALL rebuild.00546278
Stack SS:[0012FA40]=03709408, (ASCII "0958")
```

- 4 số tiếp theo trong lô trên (cho tới giờ chưa hề có thêm biến đổi nào ,chỉ là cắt các số đầu tiên ra) .Lết tiếp :

```
00547C9E . E8 21C8EBFF CALL rebuild.004044C4
00547CA3 . FF75 84    PUSH DWORD PTR SS:[EBP-7C]
00547CA6 . 68 907D5400 PUSH rebuild.00547D90
00547CAB . 8D45 80    LEA EAX,DWORD PTR SS:[EBP-80]
00547CAE . 50        PUSH EAX
00547CAF . B9 04000000 MOV ECX,4
00547CB4 . BA 0D000000 MOV EDX,0D
00547CBF . 8B45 F0    MOV EAX,DWORD PTR SS:[EBP-10]
00547CC0 . E8 03C8EBFF CALL rebuild.004044C4
00547CC1 . FF75 80    PUSH DWORD PTR SS:[EBP-80]
00547CC4 . 8D45 EC    LEA EAX,DWORD PTR SS:[EBP-14]
00547CC7 . BA 07000000 MOV EDX,7
00547CCC . E8 ABC6EBFF CALL rebuild.0040437C
00547CD1 . 8B45 F8    MOV EAX,DWORD PTR SS:[EBP-8]
00547CD4 . 8B55 EC    MOV EDX,DWORD PTR SS:[EBP-14]
00547CD7 . E8 F0C6EBFF CALL rebuild.004043CC
00547CE1 . 75 24     JNZ SHORT rebuild.00547D02
00547CE2 . C645 F7 01 MOV BYTE PTR SS:[EBP-9],1
00547CE2 . 6A 00     PUSH 0
00547CE4 . BA B07D5400 MOV EDX,rebuild.00547DB0
00547CE9 . B9 9C7D5400 MOV ECX,rebuild.00547D9C
00547CEE . B8 01000000 MOV EAX,80000001
00547CF3 . E8 80E5FFFF CALL rebuild.00546278
Stack SS:[0012FA3C]=03709198, (ASCII "44F0")
```

-Lại 4 số kế tiếp ,nhưng đủ rồi ,lúc còn lại bỏ đi ,ko xô số tụi nó đâu .Trace F8 tới đây:

```
00547CD7 . E8 F0C6EBFF CALL rebuild.004043CC
00547CDD . 75 24     JNZ SHORT rebuild.00547D02
00547CDE . C645 F7 01 MOV BYTE PTR SS:[EBP-9],1
00547CE2 . 6A 00     PUSH 0
00547CE4 . BA B07D5400 MOV EDX,rebuild.00547DB0
00547CE9 . B9 9C7D5400 MOV ECX,rebuild.00547D9C
00547CEE . B8 01000000 MOV EAX,80000001
00547CF3 . E8 80E5FFFF CALL rebuild.00546278
Stack SS:[0012FA3C]=03709198, (ASCII "44F0")
ASCII "Software\XP"
ASCII "AutoUpdate"
```

-Thế thế ,đời là thế ,chú mày so sánh 2 bộ như thế thì huynh bí đích thị đây là **Real Serial** rồi .

```
Registers (FPU)
EAX 03709680 ASCII "1111-2222-3333-4444"
ECX 31313131
EDX 037091AC ASCII "FDC8-F395-0958-44F0"
EBX FFFFFFFF
ESP 0012FA18
EBP 0012FABC
ESI 036CA2E0
EDI 0012FD04
EIP 00547CD7 rebuilde.00547CD7
```

-Vậy **Real Serial** chính là **16 số đầu tiên** của **lô số để sau cùng** .Cố Trace thêm nữa thì nó ra lại chỗ này :

005705A7	. 8B4D FC	MOV ECX,DWORD PTR SS:[EBP-4]	
005705AA	. E8 656FFDFF	CALL rebuilde.00547514	
005705AF	. 84C0	TEST AL,AL	
005705B1	74 4B	JE SHORT rebuilde.005705FE	
005705B3	66:A1 4C0657	MOV AX,WORD PTR DS:[57064C]	
005705B9	50	PUSH EAX	
005705BA	6A 00	PUSH 0	
005705BC	804D E8	LEA ECX,DWORD PTR SS:[EBP-18]	
005705BF	BA 58065700	MOV EDX,rebuilde.00570658	ASCII "C_FrmAbout_
005705C4	B8 74065700	MOV EAX,rebuilde.00570674	ASCII "We sincerely
005705C9	E8 6AA5FBFF	CALL rebuilde.0052AB38	
005705CE	8B55 E8	MOV EDX,DWORD PTR SS:[EBP-18]	

-Vậy kiểm tra đã đời ,cu cậu chuẩn bị nhảy đây mà .Thôi thôi ,ghi lại **Real Serial** rồi .Close hết Olly một cách tự tin đi nào .Đã có Real Serial thì cần chi 3 cái file Dump ,Rebuild gì nữa ,chạy luôn file gốc **xptools.exe** đi nào (file *chưa unpack*) .Bùm bùm ... huynh bít ,huynh chưa đăng kí ,cứ la hét đi ,nào Register :


Register

Registration Name:

Registration Code: - - -

-Úuuuuuuuu ...àaaaa

Information


We sincerely appreciate your support of XPTools. Now you become a legal user of XP Tools.

-Thằng cu ngáp luôn ...Một đút hơi

XP Tools

The Swiss army knife for your Windows



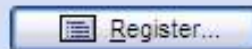
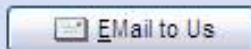
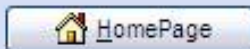
XP Tools

Version 4.58

For Windows 95/98/ME/NT/2000/XP/2003

Copyright(C) 2004 XPTools.net

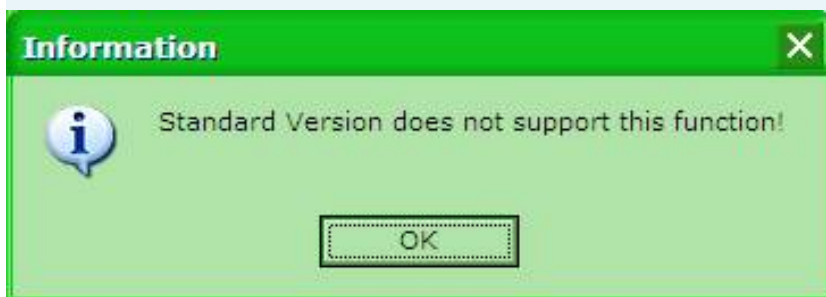
This product is licensed to: Trickyboy
Registration Code: FDC8-F395-0958-44F0



-Mà khoan ,giờ ra ngoài cửa sổ Windows Explore cái ,soft này có một chức năng là **Lock with ExeLock** ,giúp bảo vệ riêng một file Execute nào đó .Chọn đại một em ,Click phải , **XPTools --> Lock with ExeLock** :

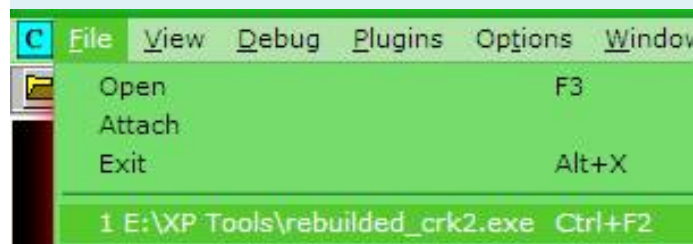


-Banggggg ui da ..ai quỳnh mỡ :



-Ừa ừa ,là sao rứa ,mỡ bỏ ngân lượng ra mua hàng hoàng mà sao chỉ có **Xì-tan-đa** là sao? Lên lại homepage thì đúng là chỉ cho down về một file này thôi .như vậy soft này chia ra làm 2 loại đăng kí ,và có thể Serial nạp vào chỉ là Serial cho bản **Xì-tan-đa** .Ặcmỡ muốn Rồ ,cuộc đời thật nhục nhã khi ko Rồ .

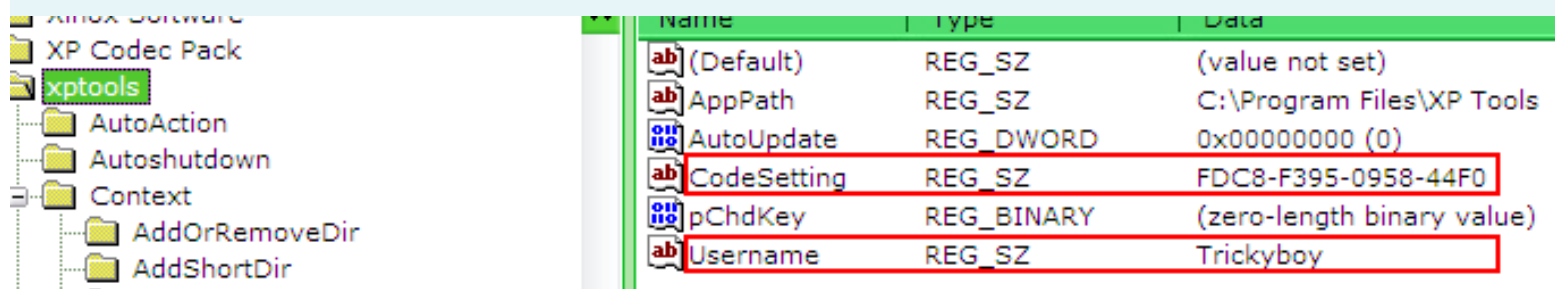
-Load lại file **rebuilted_crk2.exe** :



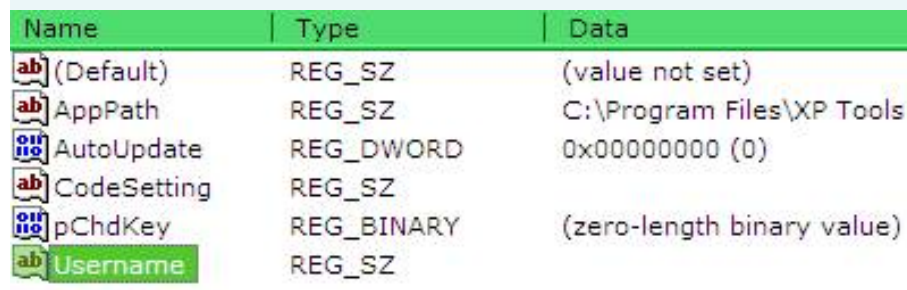
-Quên nữa ,đăng kí hợp lệ rồi thì có thể thông tin đang lưu trong Registry ,vào tìm khóa sau :



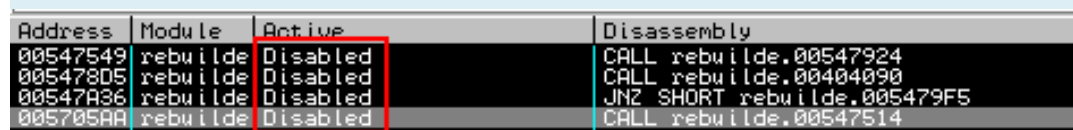
-Kéo xuống tới Software thì cho bung :



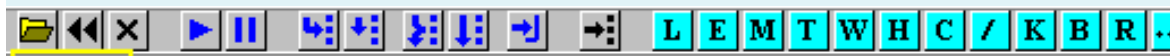
-Delete đi nào :



-Đóng ông đó đi ,ghét thấy cái mặt của mày lắm .Qua lại cung Olly ,Alt-B cái:



-Còn nhiều BP quá thì **Disable** nó đi (Click phải , **Disable All**) .Nhưng vẫn nhảy tới hàm CALL từ address 05705AA .Set lại bp cho nó :



005705A9	E8 656FFDFF	CALL rebuild.00547514	
005705B1	84C0	TEST AL,AL	
005705B3	74 4B	JE SHORT rebuild.005705FE	
005705B9	66:A1 4C0657	MOV AX,WORD PTR DS:[57064C]	
005705BA	50	PUSH EAX	
005705BC	6A 00	PUSH 0	
005705BF	804D E8	LEA ECX,DWORD PTR SS:[EBP-18]	
005705C4	BA 58065700	MOV EDX,rebuild.00570658	ASCII "C_FrmAbout_
005705C9	B8 74065700	MOV EAX,rebuild.00570674	ASCII "We sincerel
005705CB	E8 6A05E8FF	CALL rebuild.00520B38	

-Cũng Run soft ,nhập Name và Fake Serial như cũ .Register ,break .

-Trace F7 vào lệnh CALL trên ,rồi cũng Trace F8 tới lệnh CALL thứ 3 (ko thì bp nó rồi F9 sẽ break tại đó) :

00547514	55	PUSH EBP	
00547515	8BEC	MOV EBP,ESP	
00547517	83C4 F0	ADD ESP,-10	
0054751A	53	PUSH EBX	
0054751B	894D F8	MOV DWORD PTR SS:[EBP-8],ECX	
0054751E	8955 FC	MOV DWORD PTR SS:[EBP-4],EDX	
00547521	8B08	MOV EBX,EAX	
00547523	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00547526	E8 45CFEBFF	CALL rebuild.00404470	
0054752B	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
0054752E	E8 30CFEBFF	CALL rebuild.00404470	
00547533	33C0	XOR EAX,EAX	
00547535	55	PUSH EBP	
00547536	68 FF755400	PUSH rebuild.005475FF	
0054753B	64:FF30	PUSH DWORD PTR FS:[EAX]	
0054753E	64:8920	MOV DWORD PTR FS:[EAX],ESP	
00547541	8B4D F8	MOV ECX,DWORD PTR SS:[EBP-8]	
00547544	8B55 FC	MOV EDX,DWORD PTR SS:[EBP-4]	
00547547	8BC3	MOV EAX,EBX	
00547549	E8 06030000	CALL rebuild.00547924	
0054754E	8B45 F7	MOV BYTE PTR SS:[EBP-9],AL	
00547551	807D F7 00	CMP BYTE PTR SS:[EBP-9],0	
00547555	0F84 89000000	JE rebuild.005475E4	

-F7 vào ,khoan trace tiếp ,giờ kéo xuống tìm chuỗi "XPTools" :

00547A3A	8D45 D0	LEA EAX,DWORD PTR SS:[EBP-30]	
00547A3D	50	PUSH EAX	
00547A3E	8D45 C8	LEA EAX,DWORD PTR SS:[EBP-38]	
00547A41	B9 807D5400	MOV ECX,rebuild.00547D80	ASCII "XPTools"
00547A46	8B55 FC	MOV EDX,DWORD PTR SS:[EBP-4]	
00547A49	E8 BAC8EBFF	CALL rebuild.00404308	

-Vậy mã hóa ở đây chỉ dành cho **Xi-tan-da** ,Damn it ... Bỏ qua đi ,kéo xuống cho tới thẳng Pro :

00547B00	C645 F7 01	MOV BYTE PTR SS:[EBP-9],1	
00547B04	6A 01	PUSH 1	
00547B06	B9 9C7D5400	MOV ECX,rebuild.00547D9C	ASCII "AutoUpdate"
00547B0B	BA B07D5400	MOV EDX,rebuild.00547DB0	ASCII "Software\XPT
00547B10	B8 01000000	MOV EAX,80000001	
00547B15	E8 5EE7FFFF	CALL rebuild.00546278	
00547B1A	33C0	XOR EAX,EAX	
00547B1C	5A	POP EDX	
00547B1D	59	POP ECX	
00547B1E	59	POP ECX	
00547B1F	64:8910	MOV DWORD PTR FS:[EAX],EDX	
00547B22	E9 FB010000	JMP rebuild.00547D22	
00547B27	6A 10	PUSH 10	
00547B29	8D45 B4	LEA EAX,DWORD PTR SS:[EBP-4C]	
00547B2C	50	PUSH EAX	
00547B2D	8D45 AC	LEA EAX,DWORD PTR SS:[EBP-54]	
00547B30	B9 CC7D5400	MOV ECX,rebuild.00547DCC	ASCII "xptoolspro"
00547B35	8B55 FC	MOV EDX,DWORD PTR SS:[EBP-4]	

-Đúng là mày rồi ,chỉ mày mới là Pro ,set bp tại chuỗi này :

00547B27	6A 10	PUSH 10	
00547B29	8D45 B4	LEA EAX,DWORD PTR SS:[EBP-4C]	
00547B2C	50	PUSH EAX	
00547B2D	8D45 AC	LEA EAX,DWORD PTR SS:[EBP-54]	
00547B30	B9 CC7D5400	MOV ECX,rebuild.00547DCC	ASCII "xptoolspro"
00547B35	8B55 FC	MOV EDX,DWORD PTR SS:[EBP-4]	

-F9 cho nó break tại đó .Việc làm tiếp theo nếu để mỗ nói nữa thì chắc các hạ đó bỏ nghề học món này được rồi đó .
Trace :

```
Registers (FPU)
EAX 00537A44 rebuilde.00537A44
ECX 00000000
EDX 036BAE64 ASCII "TRICKYBOYXPTOOLSPRO"
EBX FFFFFFFF
ESP 0012FA10
EBP 0012F0B0
```

-Trace

-Lô số đề thứ 1 đã ra lò :

```
Registers (FPU)
EAX 036BAF08 ASCII "F5E5B15DC6A00D129BECB7E4A48F4454"
ECX 036BAE7C
EDX 0012FAAC
EBX FFFFFFFF
```

-Trace :

00547B30	. B9 CC7D5400	MOV ECX,rebuilde.00547DCC	ASCII "xptoolspro"
00547B35	. 8B55 FC	MOV EDX,DWORD PTR SS:[EBP-4]	
00547B38	. E8 CBC7EBFF	CALL rebuilde.00404308	
00547B3D	. 8B45 AC	MOV EAX,DWORD PTR SS:[EBP-54]	
00547B40	. 8D55 B0	LEA EDX,DWORD PTR SS:[EBP-50]	
00547B43	. E8 5825ECFF	CALL rebuilde.0040A0A0	
00547B48	. 8B55 B0	MOV EDX,DWORD PTR SS:[EBP-50]	
00547B4B	. 33C9	XOR ECX,ECX	
00547B4D	. A1 F8795300	MOV EAX,DWORD PTR DS:[5379F8]	
00547B52	. E8 A910FFFF	CALL rebuilde.00538C00	
00547B57	. 8B45 B4	MOV EAX,DWORD PTR SS:[EBP-4C]	
00547B5A	. 8D55 F0	LEA EDX,DWORD PTR SS:[EBP-10]	
00547B5D	. E8 96FCFFFF	CALL rebuilde.005477F8	

-Trace F7 vào nó Lại trace :

005478A6	. 52	PUSH EDX	Arg2 Arg1
005478A7	. 50	PUSH EAX	
005478A8	. 8D55 F4	LEA EDX,DWORD PTR SS:[EBP-C]	
005478AB	. B8 01000000	MOV EAX,1	
005478B0	. E8 7F2DECFF	CALL rebuilde.0040A634	rebuilde.0040A634
005478B5	. 8B45 F4	MOV EAX,DWORD PTR SS:[EBP-C]	
005478B8	. FF30	PUSH DWORD PTR DS:[EAX]	
005478BA	. 8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]	
005478BD	. E8 CACBEBFF	CALL rebuilde.0040448C	
005478C2	. 5A	POP EDX	
005478C3	. 8B5438 FF	MOV BYTE PTR DS:[EAX+EDI-1],DL	
005478C7	. 47	INC EDI	
005478C8	. 4E	DEC ESI	
005478C9	. 7F85 7BFFFF	JNZ rebuilde.0054784A	
005478CC	. 8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	

-Giảm cho đã đi em trai ,rồi lô số 2 ,và 3 , rồi Trace cho tới đây :

00547BE5	. 8B55 EC	MOV EDX,DWORD PTR SS:[EBP-14]	
00547BE8	. E8 0FC7EBFF	CALL rebuilde.0040430C	
00547BED	. 75 27	JNZ SHORT rebuilde.00547C16	
00547BEF	. C645 F7 01	MOV BYTE PTR SS:[EBP-9],1	
00547BF3	. 6A 01	PUSH 1	
00547BF5	. BA B07D5400	MOV EDX,rebuilde.00547DB0	ASCII "Software\XP
00547BFA	. B9 9C7D5400	MOV ECX,rebuilde.00547D9C	ASCII "AutoUpdate"
00547BFF	. B8 01000000	MOV EAX,80000001	

-Hào hào đích thị là mày :

```
Registers (FPU)
EAX 037072AC ASCII "1111-2222-3333-4444"
ECX 00000000
EDX 036BAEAC ASCII "E4D4-A04C-B59F-FC01"
EBX FFFFFFFF
ESP 0012FA18
EBP 0012FABC
ESI 036BCC30
EDI 0012FD04

Serial for Pro
```

-Chỉ muội mới xứng với huynh ,giống như chỉ Đại Tà Vương mới xứng với Dịch Phong. Giờ có thể vớt tụi Dump , Rebuild đi được rồi đó (Recycle Bin là nơi thích hợp nhất) :

XP Tools



The Swiss army knife for your Windows


XP Tools


Version 4.58

For Windows 95/98/ME/NT/2000/XP/2003

Copyright(C) 2004 XPTools.net

This product is licensed to: Trickyboy
Registration Code: E4D4-A04C-B59F-FC01


 [HomePage](#)

 [E-Mail to Us](#)

 [Register...](#)

 [Order](#)

-Thử lại chức năng **Lock with ExeLock** cho chắc ăn :



Password Protection

Password

Please specify a password that will be used to protect the program. Password are case sensitive.

Password:

Verification:

IMPORTANT: make suer you remember the password you just entered. Upon successful protection the original executable file will be saved with .ExeLock file type. You might want to store this file separately emergency purposes.

OK Cancel

-DONE !!! nó cho ta nhập pass để bảo vệ file Execute rồi .Hi hi ,hự ...ự ... ko hiểu sao mỗi lần thổ huyết ,chắc do vận kình nhiều quá ,khí tụ đan điền ,khí tu đan điền ...dưỡng sức thôi ...

-Thế đây ,giang hồ quả lăm chuyện kì ảo ,chém bằng kiếm hình (**Patch to use Soft without register**) thì nó ko chết mà chỉ khẩu phục chứ tâm chưa phục .Okie tâm gấp tâm ,ta dùng kiếm tâm để chém nó thì với **Xì-tan-đa** ,nó ngáp ngáp chứ chưa chết hẳn .Thêm một nhát kiếm ý **Pro** ,ngắt hẳn ,quắc cần câu .Khè khè ...

-Rồi ,mọi bước Unpack ,Patch và Find Serial đã quá rõ ràng ,cứ như "Tôi là Maria ".Code keygen xin nhường lại cho chư vị đồng đạo .Tại trình độ mờ tối tới thời điểm hoàn thành bí kíp này là 3 tháng tròn (mở chính thức nhập ma hồi 26/07/2005 ,lúc mờ nghỉ hè) .Mong rằng vài tháng nữa biết code keygen ,mở mới vào tiết đầu về ASM thôi àh . À **Takada** ,số Serial huynh cho đệ hôm trước là của **Xì-tan-đa** ,khi viết Tut ,vô tình huynh ngộ thêm thức **Serial for Pro** này ,nên nếu đệ có Code keygen thì code thăng Pro nha .

-Trăm bức hình cũng ko bằng tấm **Armadillo 4.4** và **PEiD 0.94** của cụ **nhỏ** ,cụ kinh thật ,khoe hàng nào là sốt món đó .Xong Tut này ,chắc mờ chuyên nghề sang chụp hình wá ,mở thấy mờ có khiếu chụp hình lắm .Vì quay phim Video thì mờ ko có hứng và dễ làm người xem nôn nóng mà ko bít khúc nào khúc nào cả .Mà nhắc hình thì lại ghét cha **MaDMan** ,chả chụp hình có mười mấy tấm ,mà Tut nào cũng nặng tới mấy MB ,cũng phải chả chơi hình toàn dạng .BMP .Thằng cha đó Crack giỏi mà sao chụp hình ngu KINH ...hehehe

-Thôi ,dừng kiểm tại đây ,quay lại với nội suy La-Răng ,Niu-ton và Súp bô chín .

- **Chân thành cảm ơn** các anh chị Admin ,Mod và tất cả các thành viên của REA.

- **Đặc biệt** là anh **hacnho** (trong bài em gởi là c) .

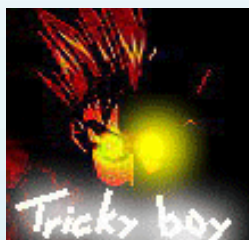
- **Thanx to :**

- **MaDMan_H3rCuL3s (I hate this guy)**

- **Ricardo Narvaja (I like this guy)**

- **And YOU ...**

Written by Trickyboy (tự Tàn Kiếm)



(26/10/2005)

(Beta) unpack Armadillo - Standard protection only

Author: Benina

Target: **FlashFavorite v1. 4.5**

Website: <http://www.pipissoft.com/>

Protection ..: + Standard ARMADiLLO protection only

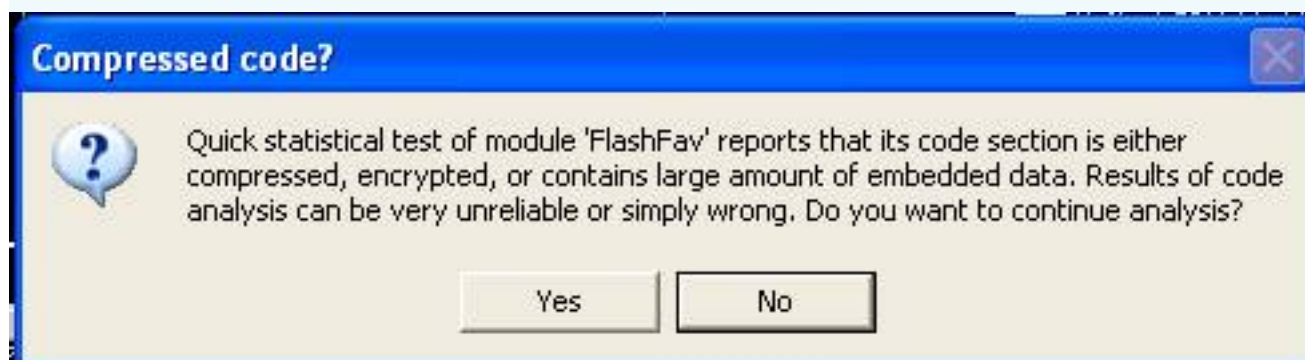
Difficulty: For Newbie

Tools Needed:

- 1.) Olly Debug v1.10 or better
- 2.) LordPE Deluxe
- 3.) Import Reconstructor v1.6 Final

Step 1: Find OEP

Load file FashFavorite.exe in Olly, appear a message:



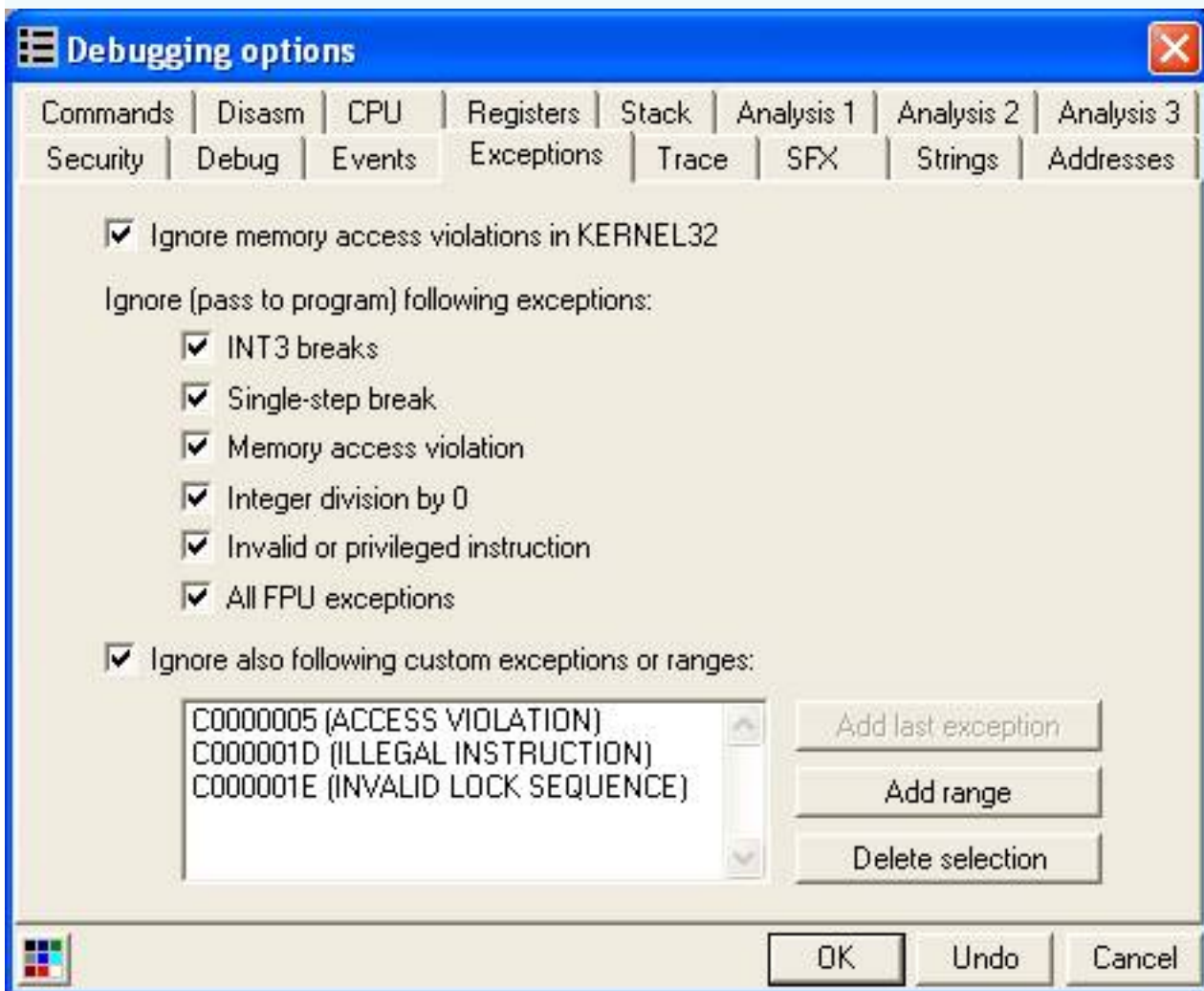
Human button "NO".

Armadillo's Entry Point will:

```
0044D000> 60 PUSHAD
0044D001 E8 00000000 CALL FlashFav.0044D006
0044D006 5D POP EBP
0044D007 50 PUSH EAX
0044D008 51 PUSH ECX
0044D009 0FCA BSWAP EDX
F7D2 NOT 0044D00B EDX
0044D00D PUSHFD 9C
F7D2 NOT 0044D00E EDX
0044D010 0FCA BSWAP EDX
0044D012 EB 0F JMP SHORT FlashFav.0044D023
0044D014 B9 EB0FB8EB MOV ECX, EBB80FEB
```

```
0044D019 07 POP ES; modification of segment register
0044D01A B9 EB0F90EB MOV ECX, EB900FEB
OR 08FD CH 0044D01F, BH
0044D021 EB 0B JMP SHORT FlashFav.0044D02E
0044D023 F2: prefix REPNE;; Superfluous prefix
0044D024 ^ EB F5 JMP SHORT FlashFav.0044D01B
0044D026 ^ EB F6 JMP SHORT FlashFav.0044D01E
0044D028 F2: prefix REPNE;; Superfluous prefix
0044D029 EB 08 JMP SHORT FlashFav.0044D033
0044D02B FD STD
0044D02C ^ E9 EB JMP SHORT FlashFav.0044D017
0044D02E F3: prefix REP;; Superfluous prefix
E4 ^ 0044D02F EB JMP SHORT FlashFav.0044D015
0044D031 FC CLD
0044D032 - E9 9D0FC98B JMP 8C0DDFD4
```

First, in Olly, we select the menu Options / debugging options, such as the image below:



And select Options HideDebugger of Plugins:



Armadillo will unpacking and decryption of data into trouble chep section. Text, then the enforcement of the directive section. Text. This section of the code not the original. So OEP will in this section. Therefore, we will set break-on-access on this section. The purpose of the program to access any data in this section will have a break

In Olly, open window [Memory map] (Alt + M), click on the section. Text and click to set F2 BP as image:

Address	Size	Owner	Section	Contains	Type	Access	Initial	
00010000	00001000				Priv	RW	RW	
00020000	00001000				Priv	RW	RW	
0012C000	00001000				Priv	RW	Gua: RW	
0012D000	00003000			stack of ma	Priv	RW	Gua: RW	
00130000	00001000				Map	R	R	
00140000	00002000				Map	R	R	
00150000	00004000				Priv	RW	RW	
00250000	00006000				Priv	RW	RW	
00260000	00001000				Map	RW	RW	
00270000	00016000				Map	R	R	
00290000	00034000				Map	R	R	
002D0000	00041000				Map	R	R	
00320000	00006000				Map	R	R	
00330000	00001000				Priv	RWE	RWE	
00340000	00001000				Priv	RWE	RWE	
00350000	00008000				Priv	RW	RW	
00360000	00001000				Priv	RW	RW	
00370000	00001000				Priv	RW	RW	
00400000	00001000	FlashFav		PE header	Imag	R	RWE	
00401000	00016000	FlashFav	.text		Imag	R	RWE	
00417000	00005000	FlashFav	.rdata		Imag	R	RWE	
0041C000	00001000	FlashFav	.data		Imag	R	RWE	
0041D000	00030000	FlashFav	.text1	code	Imag	R	RWE	
0044D000	00010000	FlashFav	.adata	code	Imag	R	RWE	
0045D000	00020000	FlashFav	.data1	data, import	Imag	R	RWE	
0047D000	00060000	FlashFav	.pdata		Imag	R	RWE	
004DD000	00101000	FlashFav	.rsrc	resources	Imag	R	RWE	

now, press Shift + F9, and it will stop here

004154A2 55 PUSH EBP

004154A3 8BEC MOV EBP, ESP

004154A5 6A FF PUSH -1

004154A7 68 38934100 PUSH FlashFav.00419338

004154AC 68 06564100 PUSH FlashFav.00415606; JMP to msvcrt.__except_handler3

004154B1 64: A1 00000000 MOV EAX, DWORD PTR FS: [0]

004154B7 50 PUSH EAX

004154B8 64:8925 00000000> MOV DWORD PTR FS: [0], ESP

004154BF 83EC 68 SUB ESP, 68

004154C2 53 PUSH EBX

004154C3 56 PUSH ESI

004154C4 57 PUSH EDI

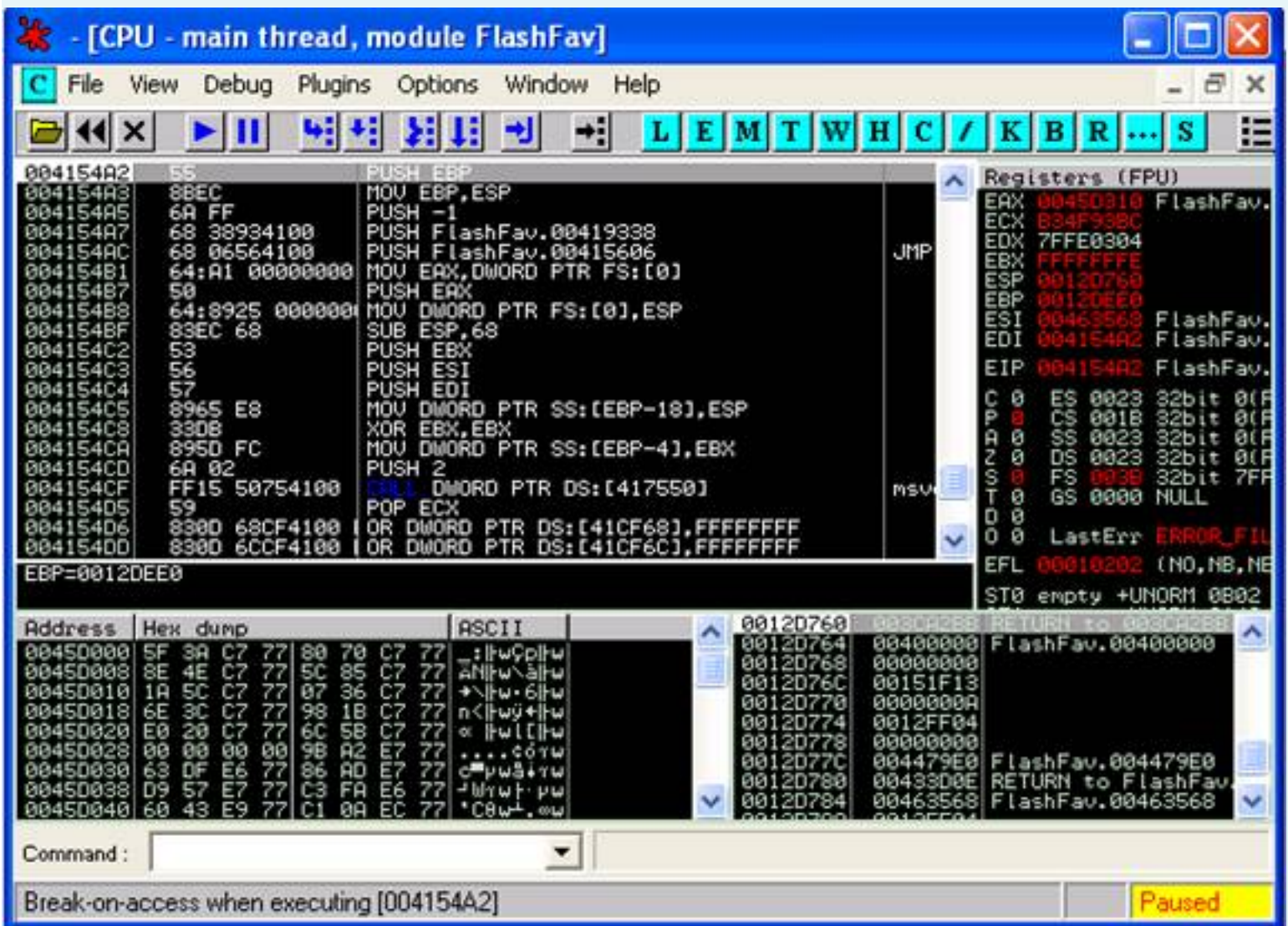
004154C5 8965 E8 MOV DWORD PTR SS: [EBP-18], ESP

004154C8 33DB XOR EBX, EBX

004154CA 895D FC MOV DWORD PTR SS: [EBP-4], EBX

004154CD 6A 02 PUSH 2

004154CF FF15 50754100 CALL DWORD PTR DS: [417550]; msvcrt.__set_app_type



In the state we find them "Break-on-access when executing [004154A2]", as well section. Text information as follows:

Address = 00401000

Size = 00016000 (90112.)

We see in the address 004154A2 section. Text. So

Address [004154A2] is the program's OEP

Step 2: Find an address any of the desire of the API program has original unpacking the section. Text, then set bp at this address:

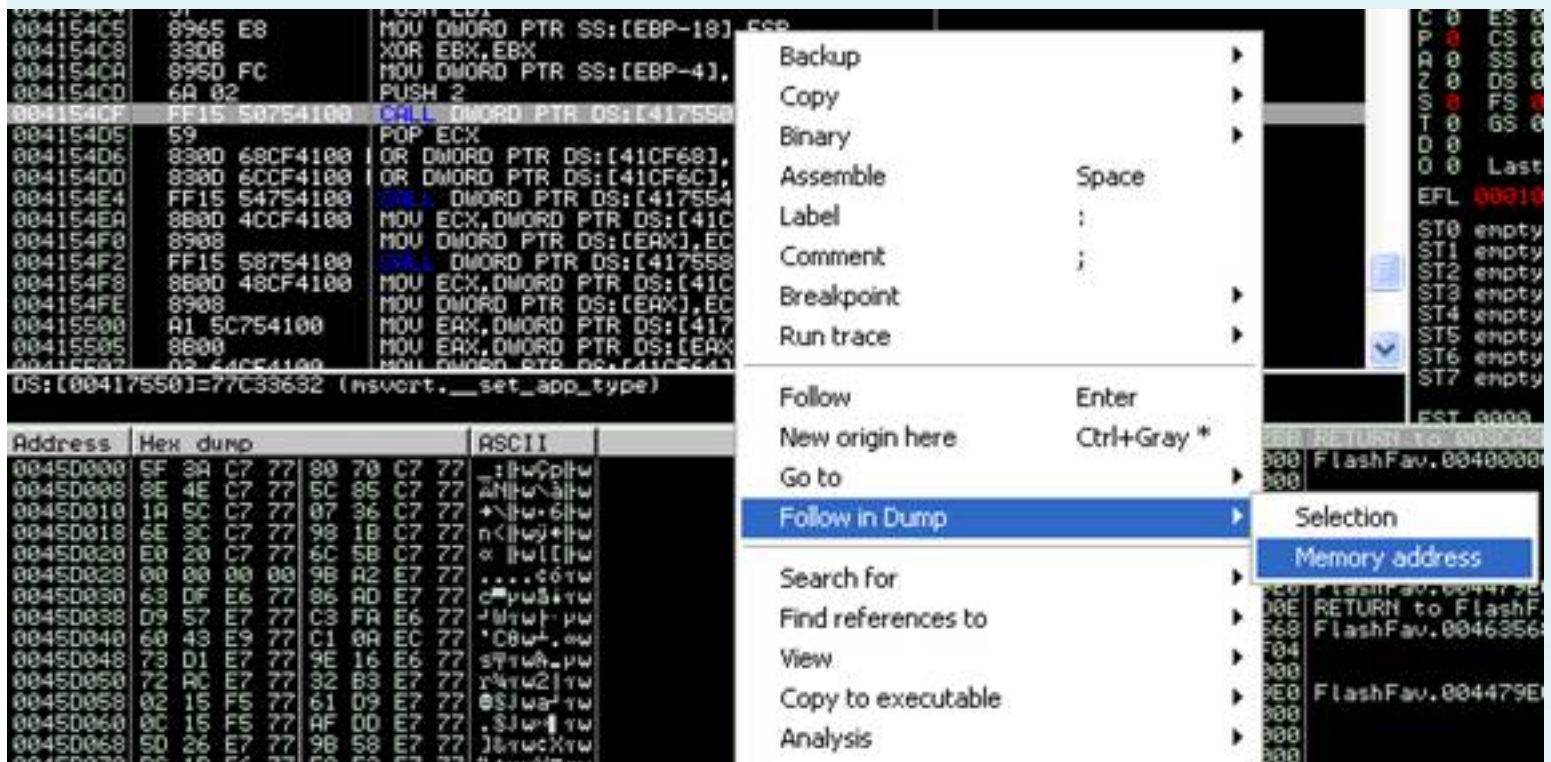
Armadillo blended IAT your original program. So, we will find the address 1 of 1 ham API any IAT table in the program set to original bp Hardware, on write at that address. The purpose of us is: when the Armadillo write bytes to the IAT (which is a specific message to the address we set bp), and Olly will stop for us to find code that Olly blended IAT table.

We see in the window of CPU Olly at 004154CF addr as follows:

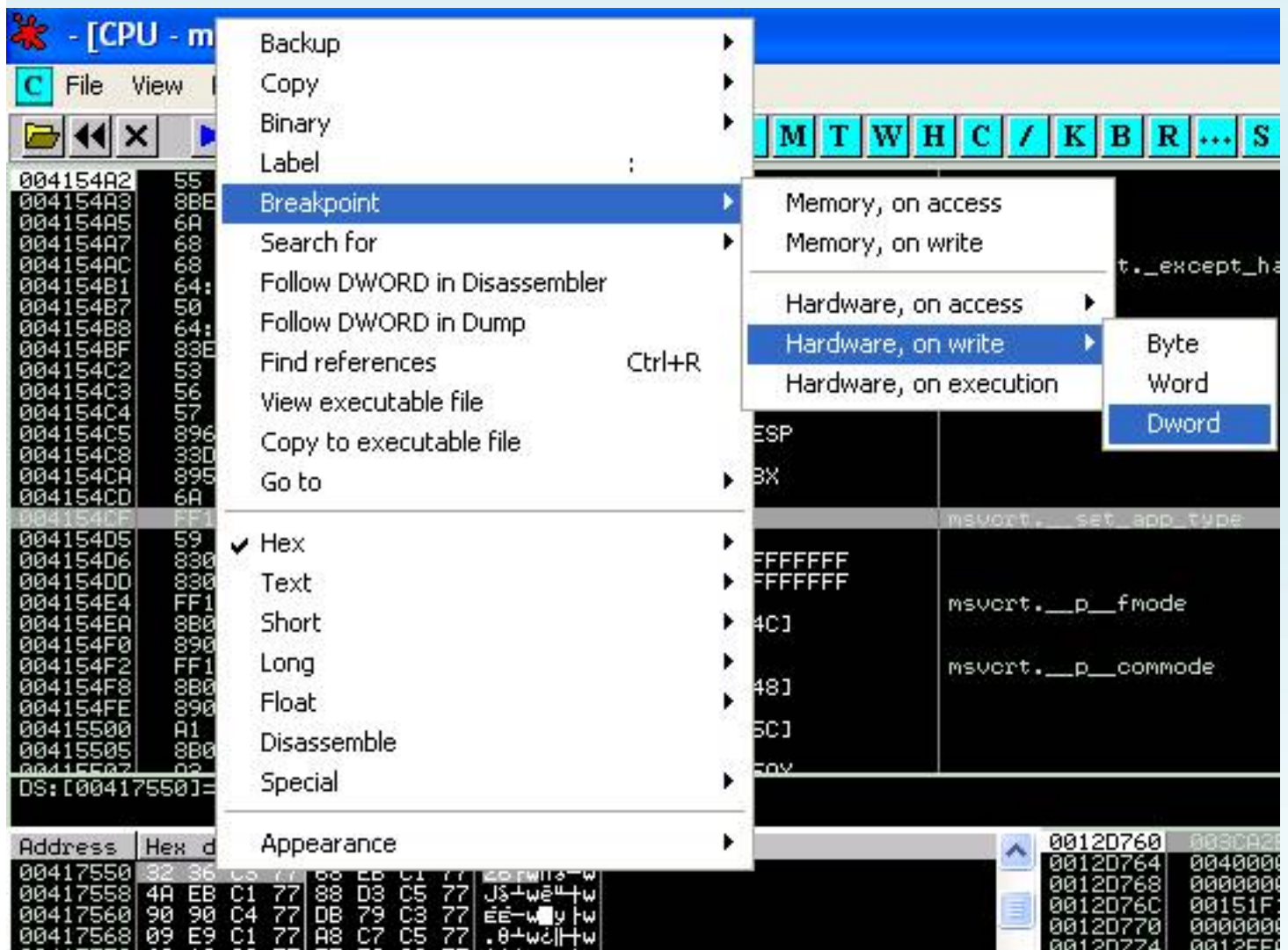
004154CF FF15 50754100 CALL DWORD PTR DS:[417550]; msvcrt.__set_app_type

So address [417,550] is 1 addr IAT table in the program's original.

Now we will dump it in the dump window of the mouse must Olly. Click 004154CF, Select in Follow dump / Memory address as image.



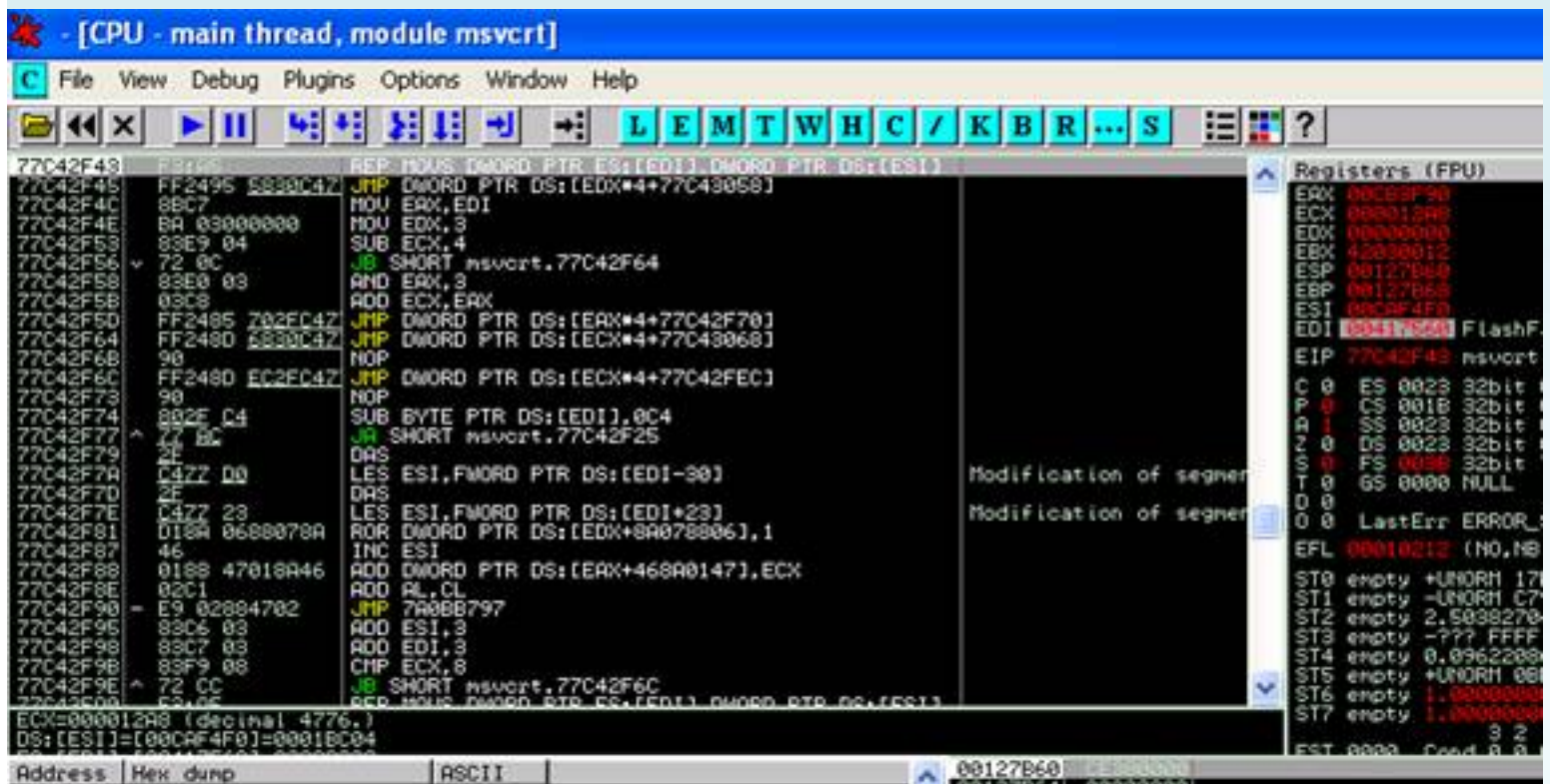
In dump window of Olly, we choose to click 4 bytes at 00417550, and set breakpoint Hardware, write on / Dword at address 00417550 image as follows:



Now we close them again Olly

Step 3: Find code blended IAT of Armadillo.

Load file FashFavorite.exe in Olly again. RUN with Shift + F9. Stop at breakpoint that we set Hardware set breakpoint, write on / Dword. We'll see the picture as follows:



77C42F43 F3: A5 REP MOVSD WORD PTR ES: [EDI], DWORD PTR DS: [ESI]

77C42F45 FF2495 5830C477 JMP DWORD PTR DS: [EDX * 4 + 77 C43058]

77C42F4C 8BC7 MOV EAX, EDI

77C42F4E BA 03000000 MOV EDX, 3

77C42F53 83E9 04 SUB ECX, 4

77C42F56 72 0C JB SHORT msvcrt.77C42F64

77C42F58 83E0 03 AND EAX, 3

77C42F5B 03C8 ADD ECX, EAX

77C42F5D FF2485 702FC477 JMP DWORD PTR DS: [EAX * 4 + 77 C42F70]

77C42F64 FF248D 6830C477 JMP DWORD PTR DS: [ECX * 4 + 77 C43068]

77C42F6B 90 NOP

77C42F6C FF248D EC2FC477 JMP DWORD PTR DS: [ECX * 4 + 77 C42FEC]

77C42F73 90 NOP

77C42F74 802F C4 SUB BYTE PTR DS: [EDI], 0C4

77C42F77 77 JA SHORT msvcrt.77C42F25

In dump window, we Goto 00417550 to address that we set breakpoint in step 2.
Click on the window dump, Ctrl + G, go to the address "00417550" as the image below:



Press OK.

In window dump we see:

```
00417550 46 BC 01 00 38 BC 01 00 F ¼ ¼ .8.
00417558 28 BC 01 00 18 BC 01 00 (¼.¼.
00417560 00 00 00 00 00 00 00 00 .....
00417568 00 00 00 00 00 00 00 00 .....
```

The value at 00417550 ko must address is 1 of ham API (its value = 0001BC46, business addresses, such as ham API as follows 77xxxxxx), measured by our press F9 to continue to run. No stop at just Market follows:

```
003C70B3 8B85 04C8FFFF MOV EAX, DWORD PTR SS: [EBP-37FC]; FlashFav.00417550
003C70B9 83C0 04 ADD EAX, 4
003C70BC 8985 04C8FFFF MOV DWORD PTR SS: [EBP-37FC], EAX
003C70C2 ^ E9 CEFCEFFF JMP 003C6D95
003C70C7 FF15 9C023D00 CALL DWORD PTR DS: [3D029C]; kernel32.GetTickCount
```

Look at the window dump us see:

```
00417550 32 36 C3 77 38 BC 01 00 26Aw8 ¼.
00417558 28 BC 01 00 18 BC 01 00 (¼.¼.
00417560 04 BC 01 00 F8 BB 01 00
```

¼. Ø.

Value [77C33632] at address 00417550 is the address of 1 ham API. So, this is the area code area code that Armadillo after blended into chep IAT of original programs, this is the area code we

need to find:

```

003C70B3 8B85 04C8FFFF MOV EAX, DWORD PTR SS: [EBP-37FC]; FlashFav.00417550
003C70B9 83C0 04 ADD EAX, 4
003C70BC 8985 04C8FFFF MOV DWORD PTR SS: [EBP-37FC], EAX
003C70C2 ^ E9 CEFCEFFF JMP 003C6D95
003C70C7 FF15 9C023D00 CALL DWORD PTR DS: [3D029C]; kernel32.GetTickCount

```

two lines marked yellow to identify the code now

At the direction 003C70B3, Armadillo write values of the IAT ham API. Therefore, area code "upset, blended" IAT is in the direction of the CPU window nay. Trong Olly, up by a little, we will see the area code as below:

```

003C6F30 FF15 5C033D00 CALL DWORD PTR DS: [3D035C]; msvcrt._stricmp
003C6F36 59 POP ECX
003C6F37 59 POP ECX
003C6F38 85C0 TEST EAX, EAX
003C6F3A 75 11 JNZ SHORT 003C6F4D
003C6F3C 8B85 4CC2FFFF MOV EAX, DWORD PTR SS: [EBP-3DB4]
003C6F42 8B40 08 MOV EAX, DWORD PTR DS: [EAX + 8]
003C6F45 8985 58C2FFFF MOV DWORD PTR SS: [EBP-3DA8], EAX
003C6F4B EB 02 JMP SHORT 003C6F4F
003C6F4D ^ 9D EB JMP SHORT 003C6EEC
003C6F4F 8B85 98C4FFFF MOV EAX, DWORD PTR SS: [EBP-3B68]
003C6F55 40 INC EAX
003C6F56 8985 98C4FFFF MOV DWORD PTR SS: [EBP-3B68], EAX
003C6F5C 83BD 58C2FFFF 0> Cmp DWORD PTR SS: [EBP-3DA8], 0
003C6F63 75 42 JNZ SHORT 003C6FA7
003C6F65 0FB785 5CC2FFFF MOVZX EAX, WORD PTR SS: [EBP-3DA4]
003C6F6C 85C0 TEST EAX, EAX
003C6F6E 74 0F JE SHORT 003C6F7F
003C6F70 0FB785 5CC2FFFF MOVZX EAX, WORD PTR SS: [EBP-3DA4]
003C6F77 8985 5CADFFFF MOV DWORD PTR SS: [EBP + FFFFAD5C], EAX
003C6F7D EB 0C JMP SHORT 003C6F8B
003C6F7F 8B85 54C2FFFF MOV EAX, DWORD PTR SS: [EBP-3DAC]
003C6F85 8985 5CADFFFF MOV DWORD PTR SS: [EBP + FFFFAD5C], EAX
003C6F8B 6A 01 PUSH 1

```

```
003C6F8D FFB5 5CADFFFF PUSH DWORD PTR SS: [EBP + FFFFAD5C]
003C6F93 FFB5 90C4FFFF PUSH DWORD PTR SS: [EBP-3B70]
003C6F99 E8 6E31FEFF CALL 003AA10C
003C6F9E 83C4 0C ADD ESP, 0C
003C6FA1 8985 58C2FFFF MOV DWORD PTR SS: [EBP-3DA8], EAX
003C6FA7 83BD 58C2FFFF 0> Cmp DWORD PTR SS: [EBP-3DA8], 0
003C6FAE 75 42 JNZ SHORT 003C6FF2
003C6FB0 0FB785 5CC2FFFF MOVZX EAX, WORD PTR SS: [EBP-3DA4]
003C6FB7 85C0 TEST EAX, EAX
003C6FB9 74 0F JE SHORT 003C6FCA
003C6FBB 0FB785 5CC2FFFF MOVZX EAX, WORD PTR SS: [EBP-3DA4]
003C6FC2 8985 58ADFFFF MOV DWORD PTR SS: [EBP + FFFFAD58], EAX
003C6FC8 EB 0C JMP SHORT 003C6FD6
003C6FCA 8B85 54C2FFFF MOV EAX, DWORD PTR SS: [EBP-3DAC]
003C6FD0 8985 58ADFFFF MOV DWORD PTR SS: [EBP + FFFFAD58], EAX
003C6FD6 6A 00 PUSH 0
003C6FD8 FFB5 58ADFFFF PUSH DWORD PTR SS: [EBP + FFFFAD58]
003C6FDE FFB5 90C4FFFF PUSH DWORD PTR SS: [EBP-3B70]
003C6FE4 E8 2331FEFF CALL 003AA10C
003C6FE9 83C4 0C ADD ESP, 0C
003C6FEC 8985 58C2FFFF MOV DWORD PTR SS: [EBP-3DA8], EAX
003C6FF2 83BD 58C2FFFF 0> Cmp DWORD PTR SS: [EBP-3DA8], 0
003C6FF9 0F85 98000000 JNZ 003C7097
003C6FFF 0FB785 5CC2FFFF MOVZX EAX, WORD PTR SS: [EBP-3DA4]
003C7006 85C0 TEST EAX, EAX
003C7008 74 54 JE SHORT 003C705E
003C700A FF15 E4003D00 CALL DWORD PTR DS: [3D00E4]; ntdll.RtlGetLastWin32Error
```

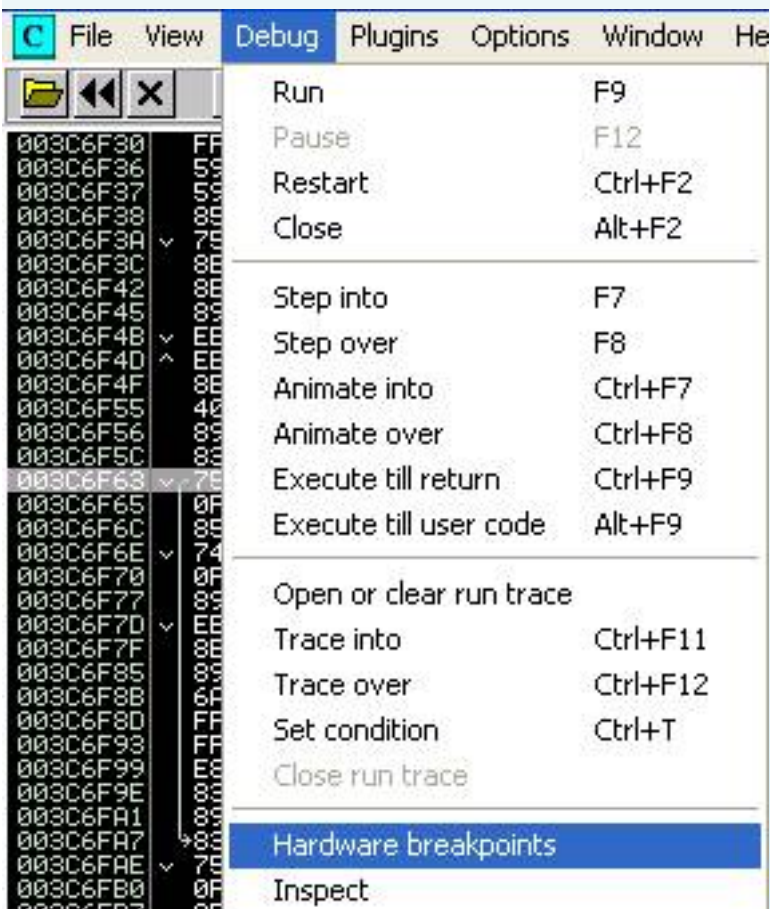



But the line I marked red, to identify the area code now

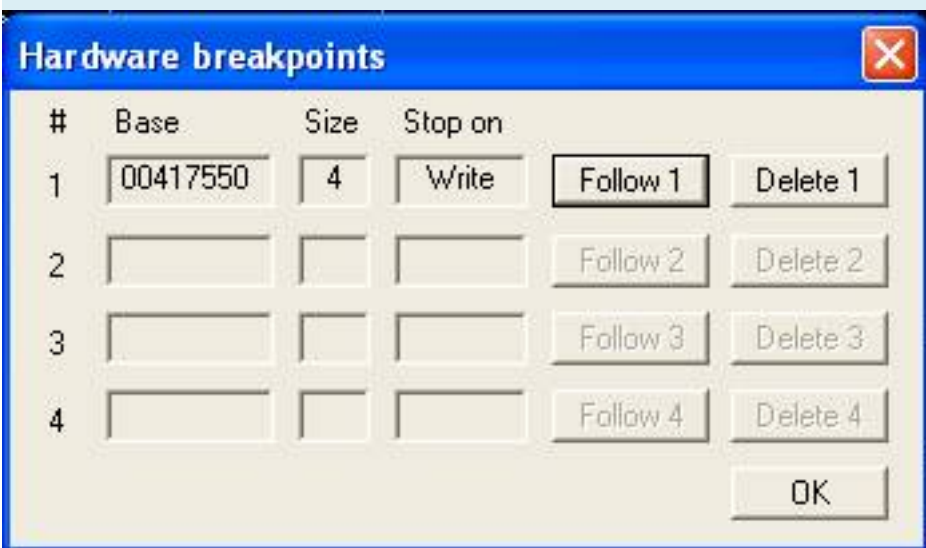
At 003C6F63 direction (of which 1 tut Spain called a pair of orders in 1 Magical jump Jump) direction is decided curl API ham or raw chep xi address your desire to IAT. If here, the Zero flag = 1, and ko "blended" chep addr of raw ham API to IAT, and if FZ = 0 will "upset"

Before going ahead, let bp clear that we set in step 2

Select the menu Debug / Hardware breakpoints:



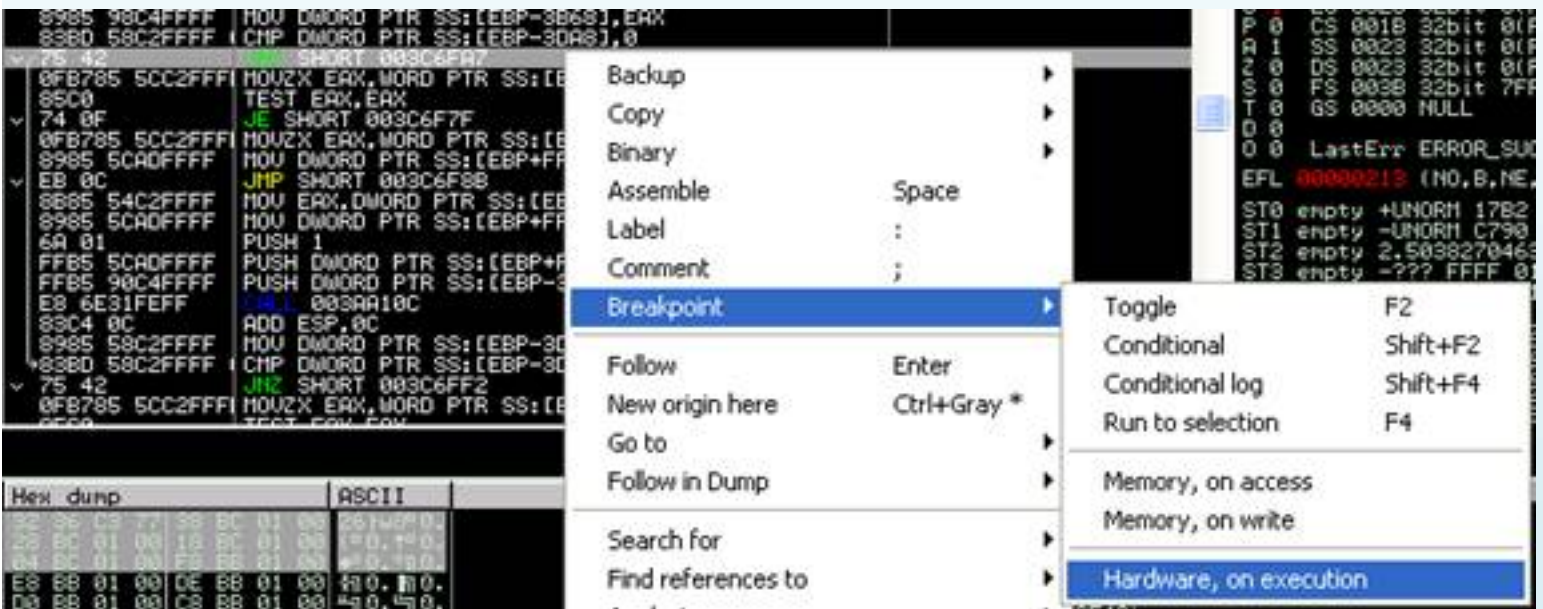
Appear:



Human button "Delete 1" to remove BP has set, and click OK

Let's set breakpoint **Hardware, execute on** 003C6F63 address at:

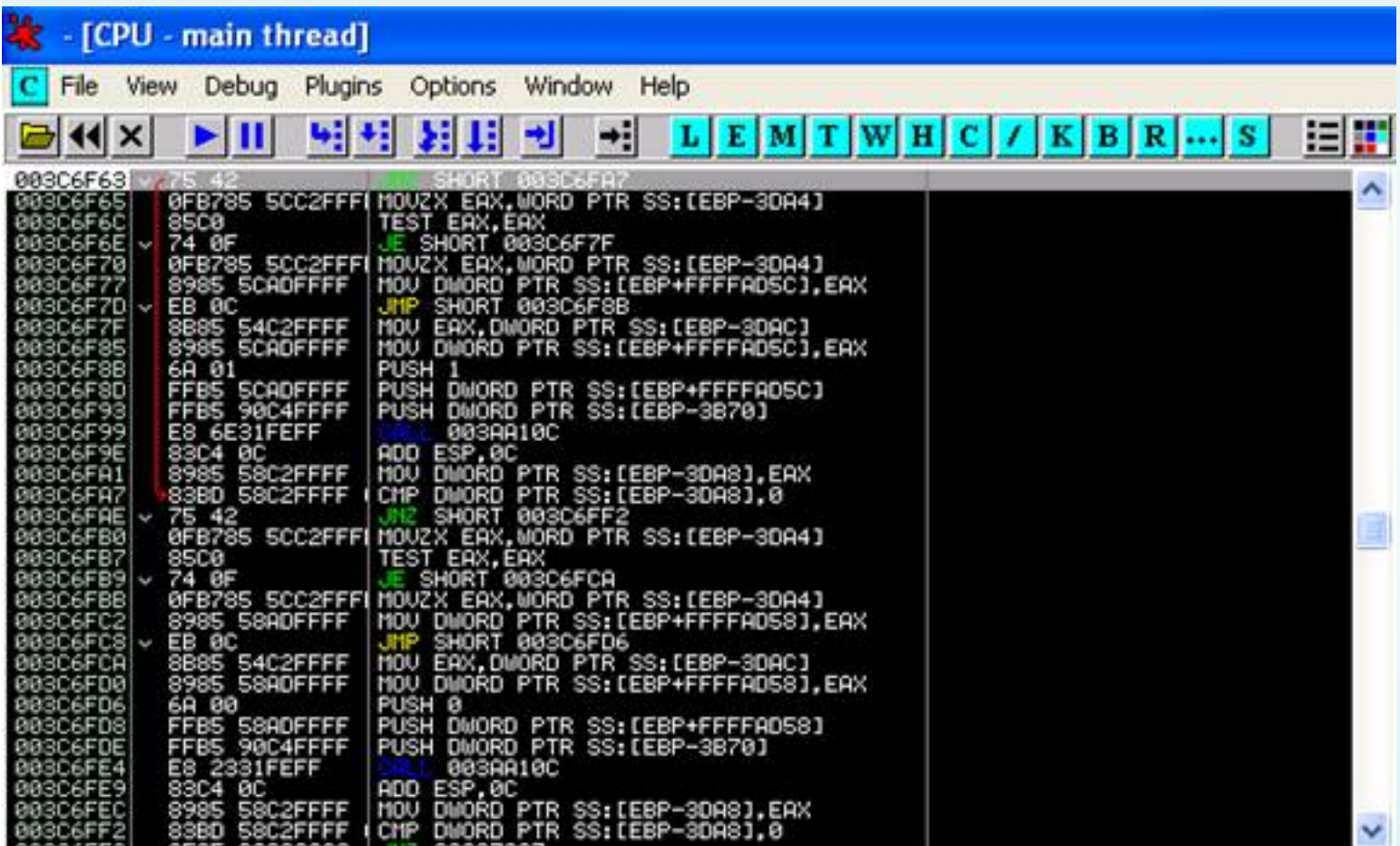
Click the mouse on the right direction and make 003C6F63 image as follows:



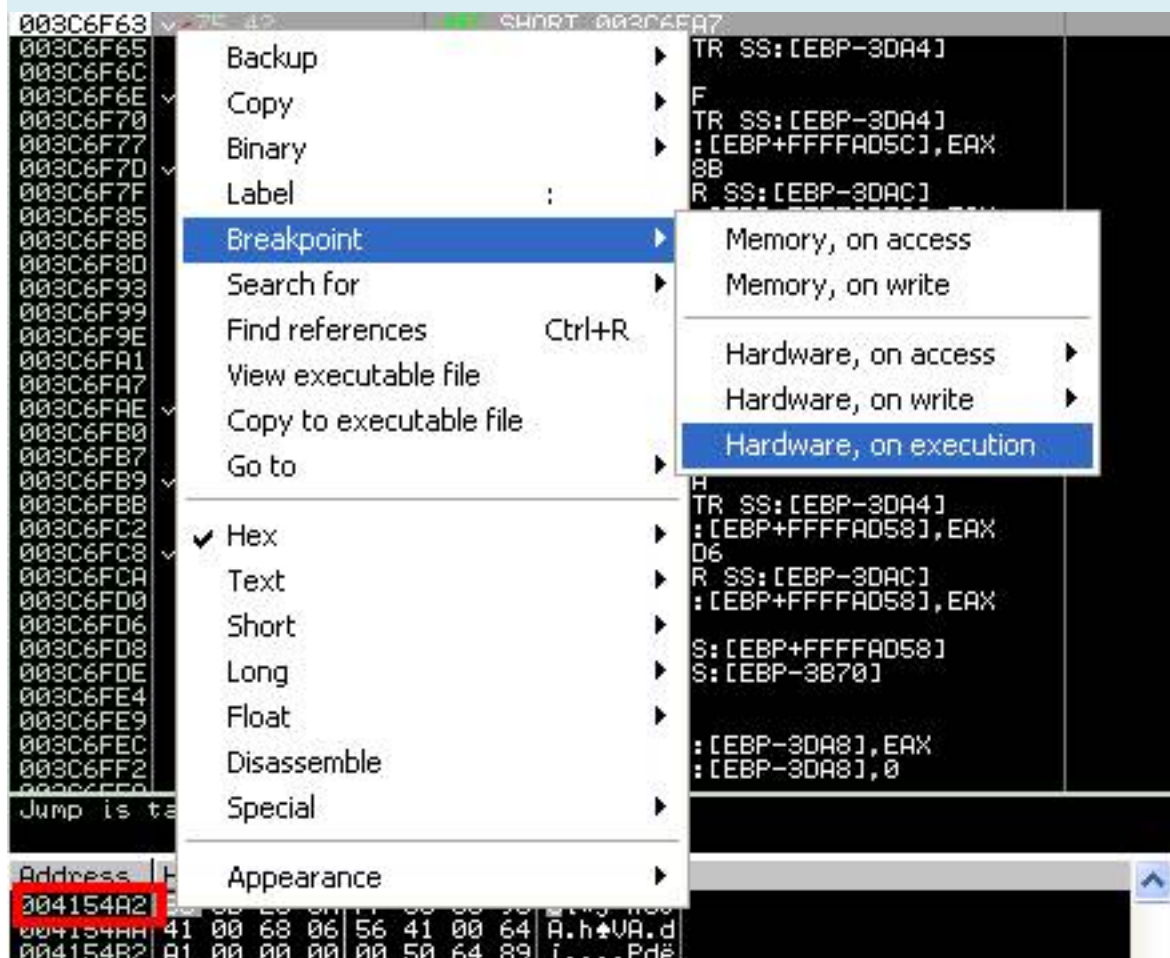
Now let Olly close again.

Step 4: Run 1 scripts corrective IAT:

Load file `FashFavorite.exe` in Olly again. Run by Shift + F9. We will stop at breakpoint BP has set Hardware, execute on the Step 3.

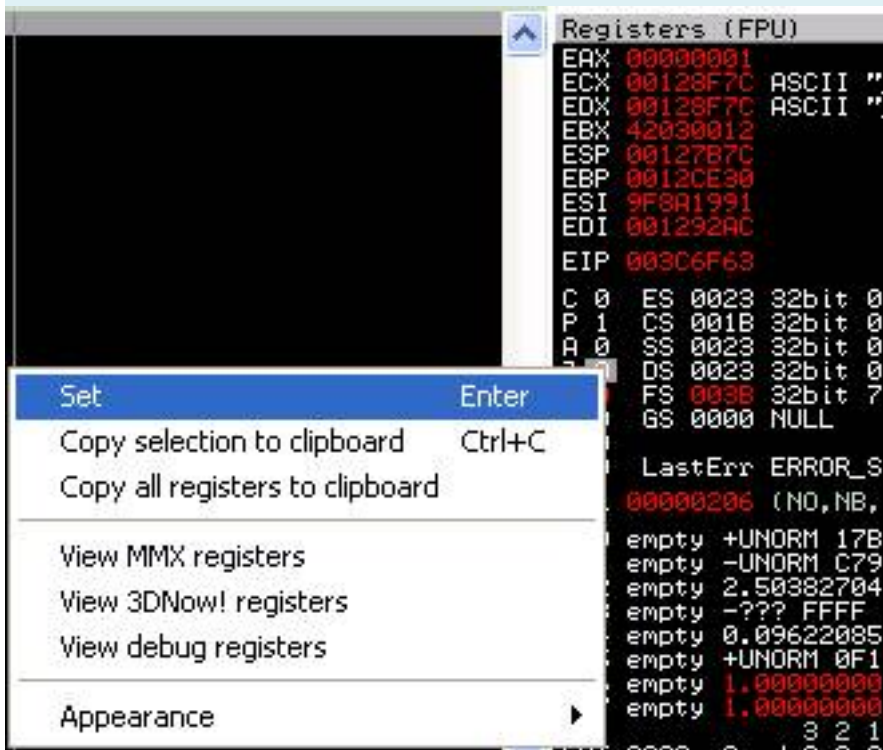


Before you run ollyscript, let us set breakpoint Hardware, execute on at OEP (004154A2) that we find in step 1. The goal is the complete script Olly run will stop here. In dump window, we Goto to addr 004154A2. Select 4 bytes, and must click the mouse on the selected bytes. BP set image as follows:



Note: Before running ollyscript

ZF = 0 If we let them set it equal to 1 (set zF = 1). Because if i ollsript run loss ham 1. We do the following, click on the Zero flag, right click, select "Set" as the image



Ollyscript used to repair the file IATscript.osc IAT is as follows:

```
IATscript.osc
```

```
dbh
```

```
eoe LABEL
```

```
eob Babel
```

```
run
```

```
LABEL:
```

```
esto
```

```
jmp LABEL
```

```
Babel :
```

```
Cmp eip, 003C6F63
```

```
jne fin
```

```
mov! zF, 1
```

```
run
```

```
jmp Babel
```

```
Fin:
```

```
ret
```

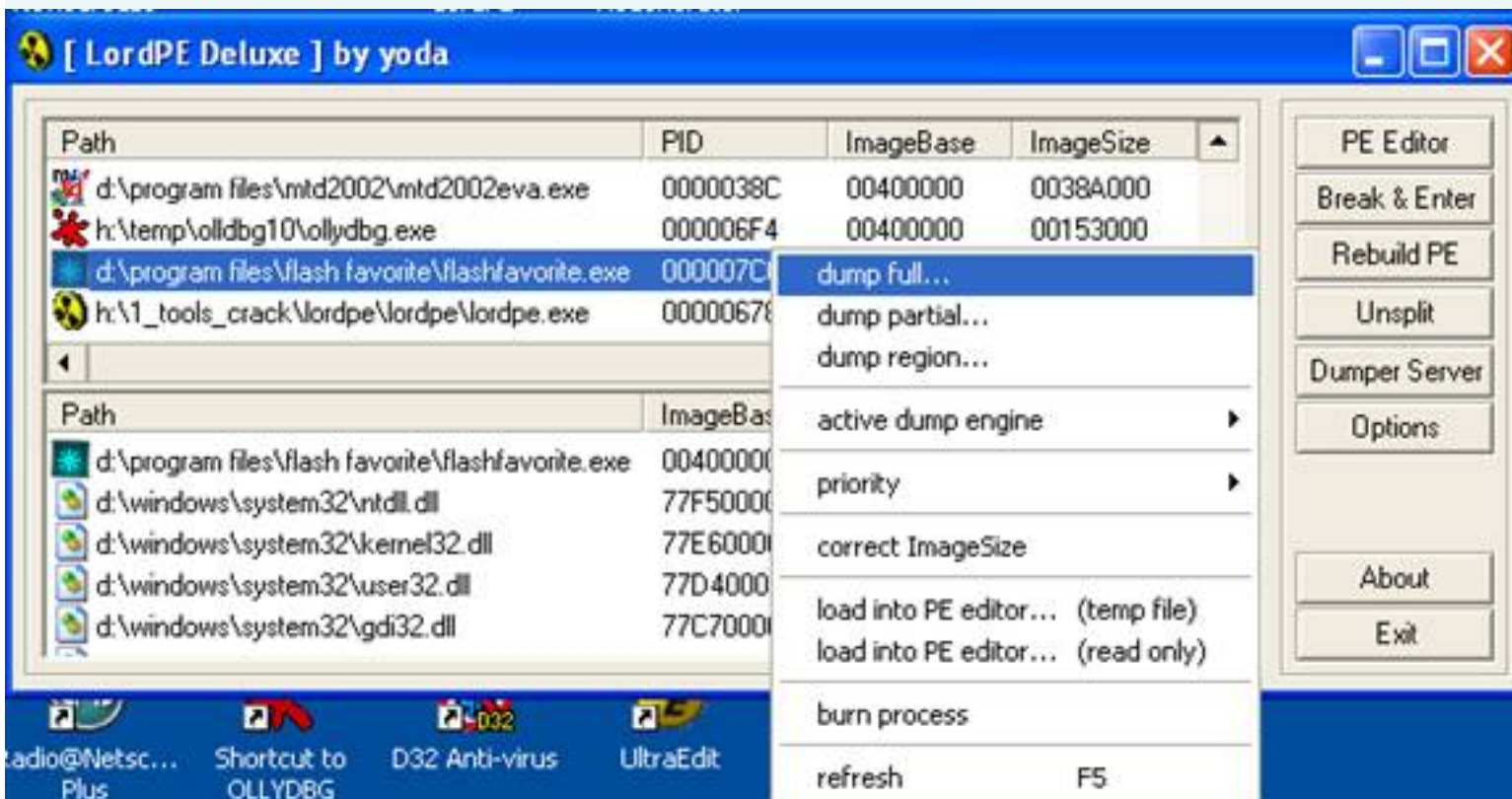
Address **003C6F63** addr in the script is that we find in step 3 (addr this as your computer)

Now, we IATscript.osc running and then it will stop at OEP we have set bp:

Address	Disassembly	Comment
004154A2	PUSH EBP	
004154A3	MOV EBP,ESP	
004154A5	PUSH -1	
004154A7	PUSH FlashFav.00419338	
004154AC	PUSH FlashFav.00415606	
004154B1	MOV EAX,DMWORD PTR FS:[0]	JMP to msvort.__except_handler3
004154B7	PUSH EAX	
004154B8	MOV DMWORD PTR FS:[0],ESP	
004154BF	SUB ESP,68	
004154C2	PUSH EBX	
004154C3	PUSH ESI	
004154C4	PUSH EDI	
004154C5	MOV DMWORD PTR SS:[EBP-18],ESP	
004154C8	XOR EBX,EBX	
004154CA	MOV DMWORD PTR SS:[EBP-4],EBX	
004154CD	PUSH 2	
004154CF	CALL DMWORD PTR DS:[417550]	msvort.__set_app_type
004154D5	POP ECX	
004154D6	OR DMWORD PTR DS:[41CF68],FFFFFFFF	
004154D0	OR DMWORD PTR DS:[41CF6C],FFFFFFFF	
004154E4	CALL DMWORD PTR DS:[417554]	msvort.__p__fmode
004154EA	MOV ECX,DMWORD PTR DS:[41CF4C]	
004154F0	MOV DMWORD PTR DS:[EAX],ECX	
004154F2	CALL DMWORD PTR DS:[417558]	msvort.__p__connode
004154F8	MOV ECX,DMWORD PTR DS:[41CF48]	
004154FE	MOV DMWORD PTR DS:[EAX],ECX	
00415500	MOV EAX,DMWORD PTR DS:[41755C]	
00415505	MOV EAX,DMWORD PTR DS:[EAX]	
00415507	MOV DMWORD PTR DS:[41CF64],EAX	
0041550C	CALL FlashFav.00415633	
00415511	CMP DMWORD PTR DS:[41CDC0],EBX	
00415517	JNZ SHORT FlashFav.00415525	

Step 5: dump:

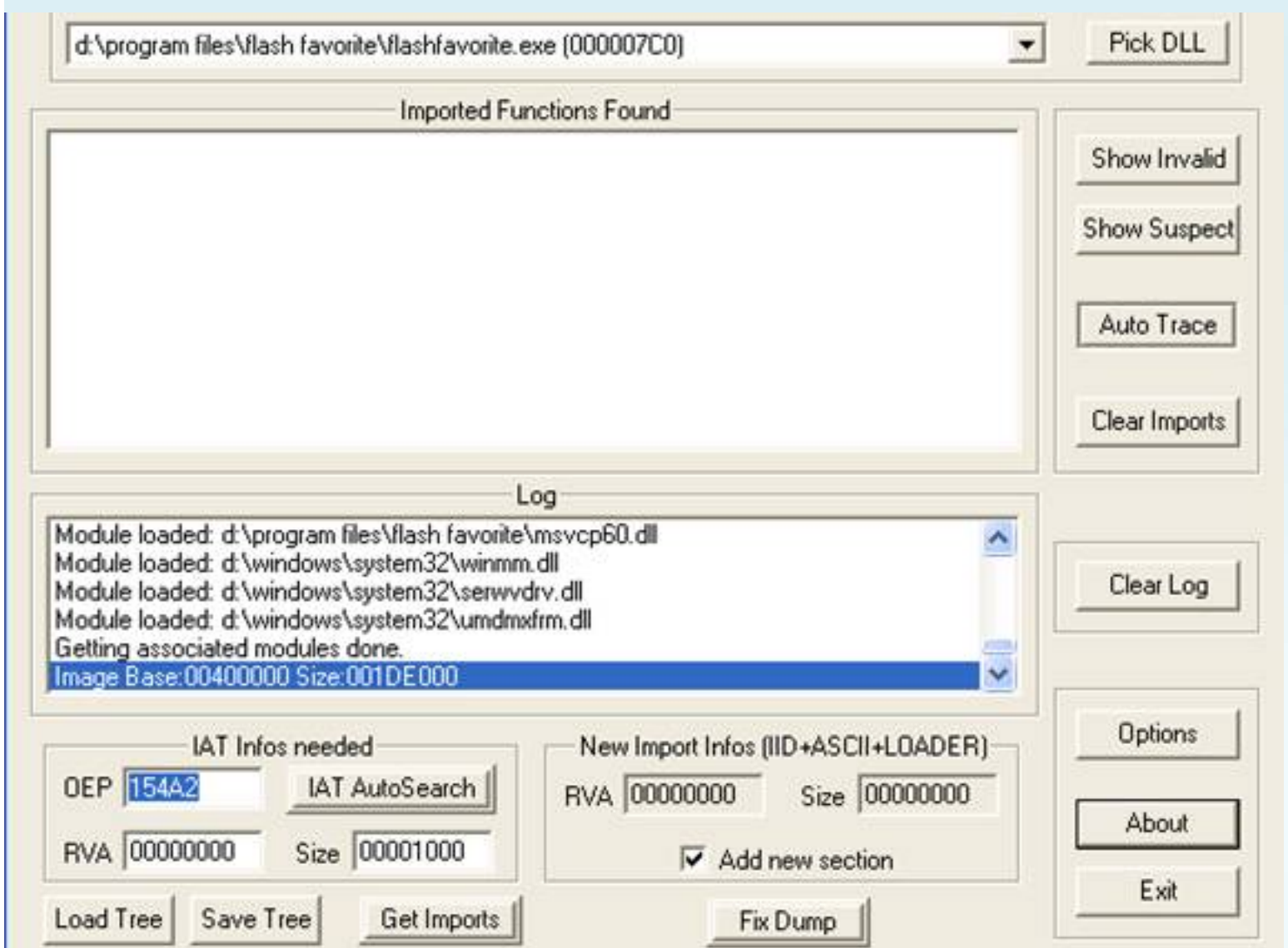
We will dump the first full fashfavorite.exe process with LordPE files dumped.exe



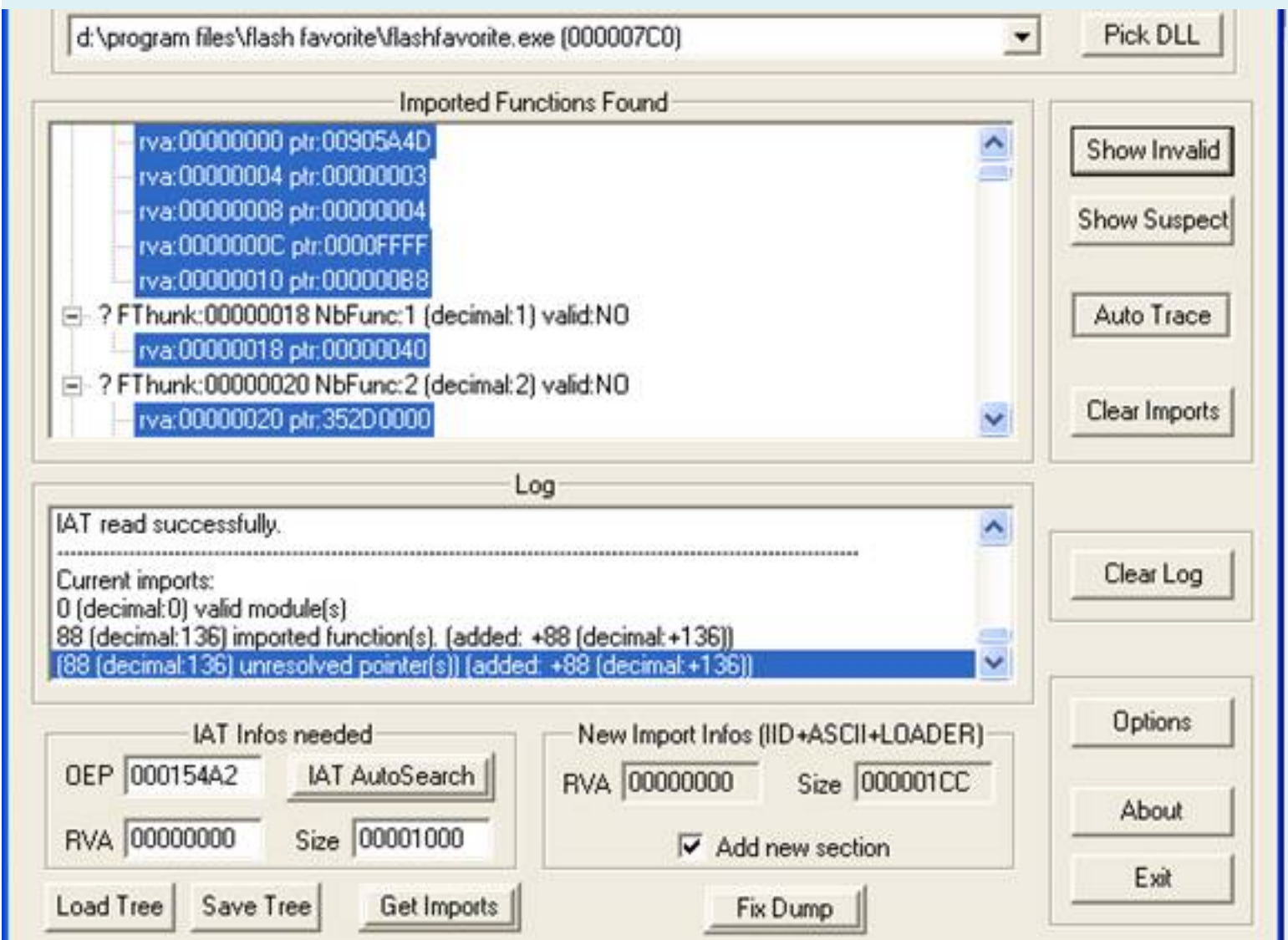
Step 6: Fix IAT:

ImpRec Open, Select process is flashfavarite.exe.

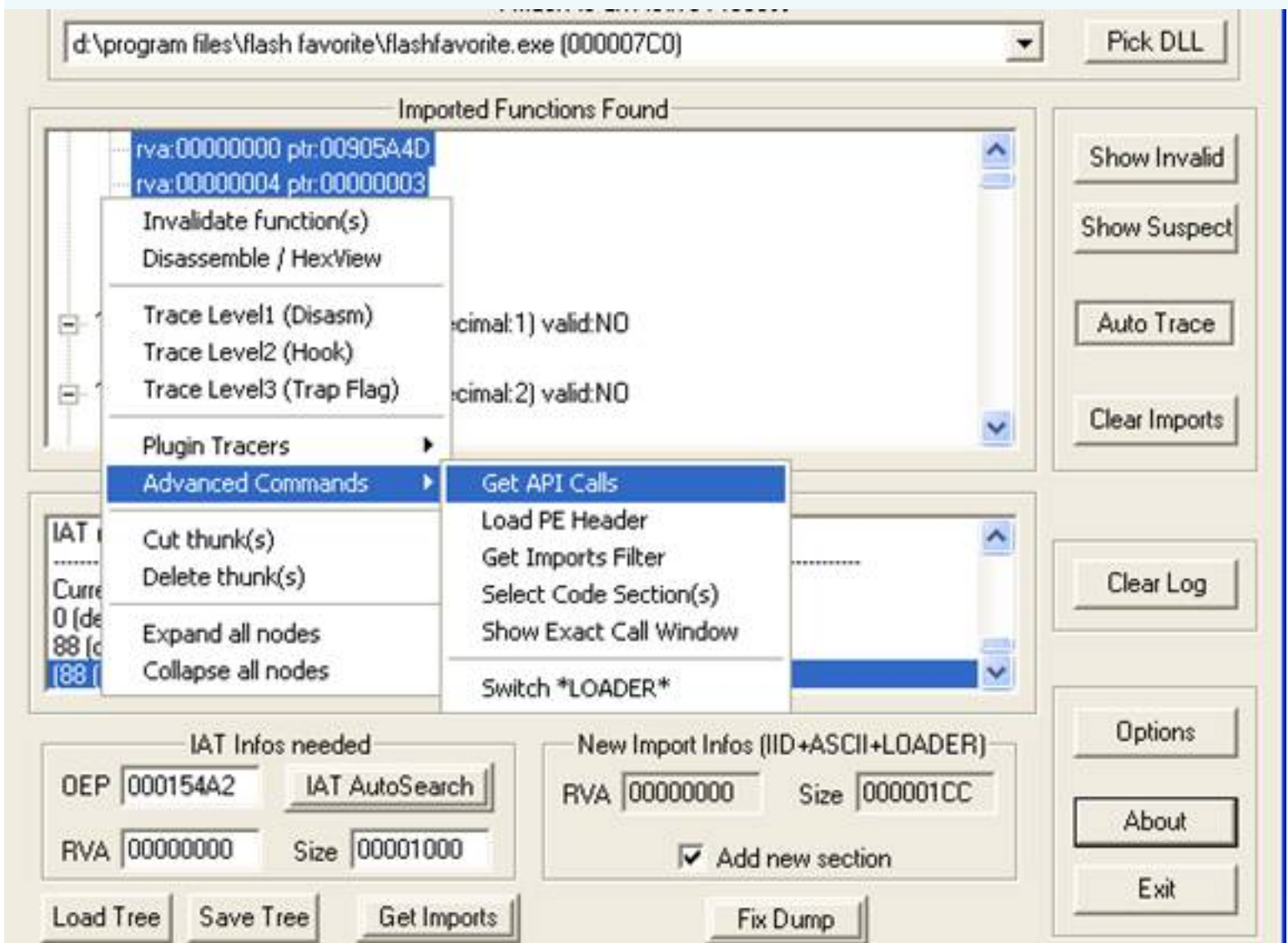
Set OEP = 154A2 as image



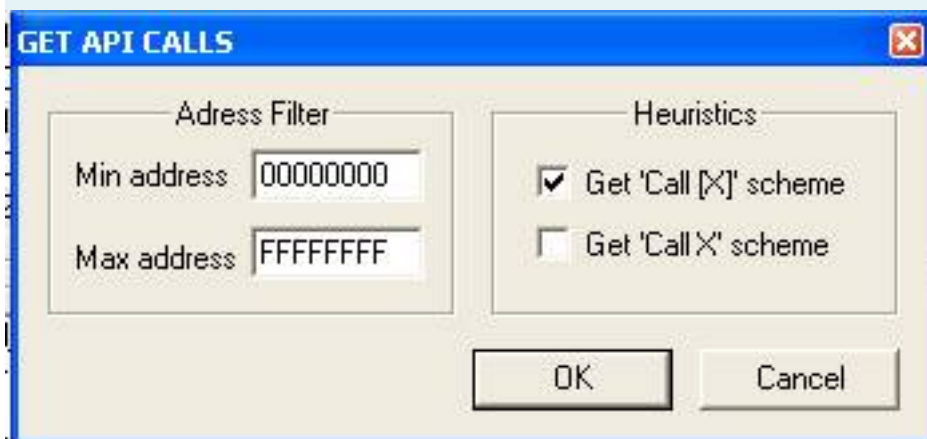
Press button "Get Imports" and then press button "Show Invalid" We'll have images as follows (Note the IAT AutoSearch ko)



Right click on the Fthunk do not recognize (such as green image above), select Advanced commands / Get API Calls

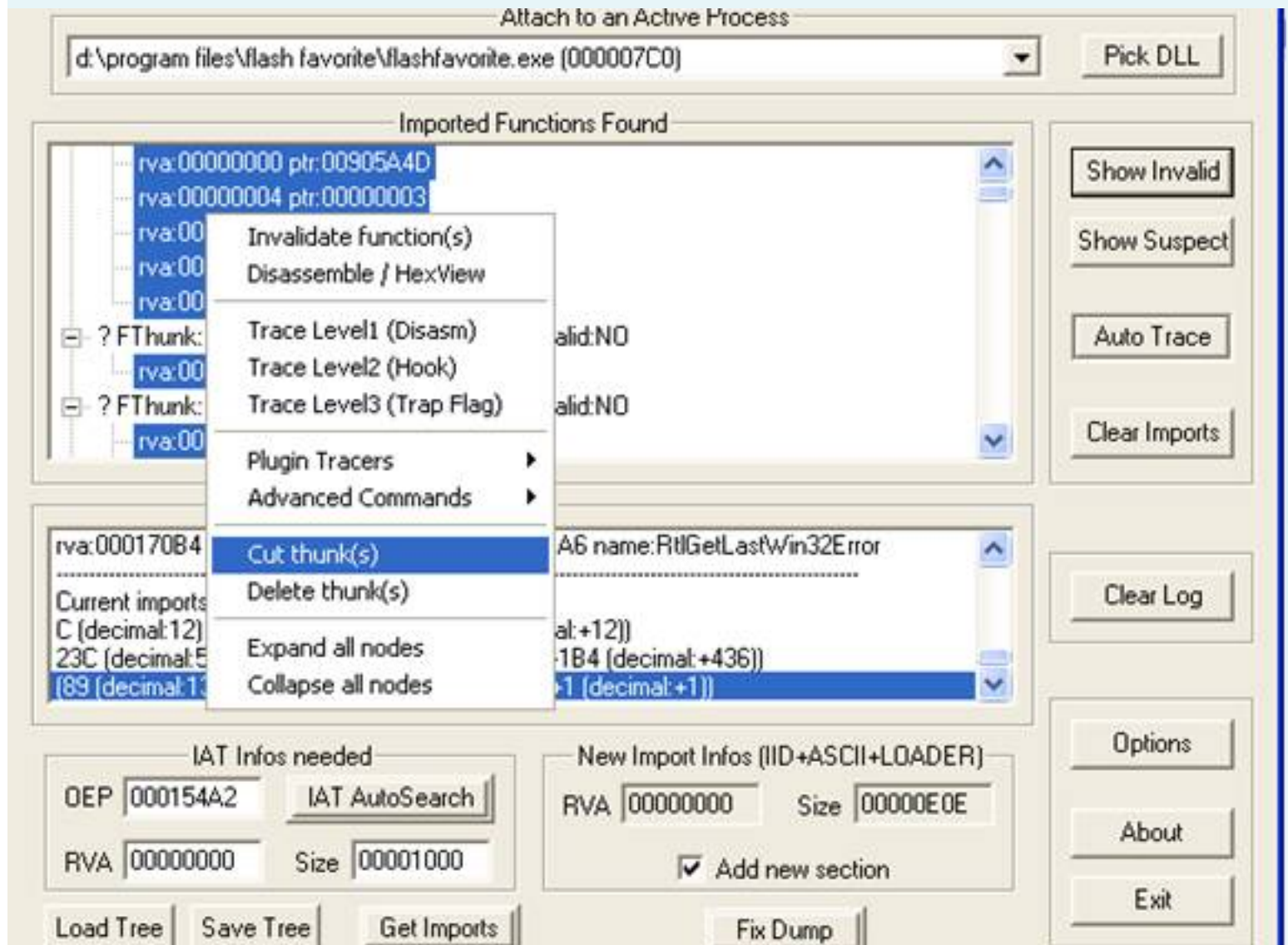


Appear:

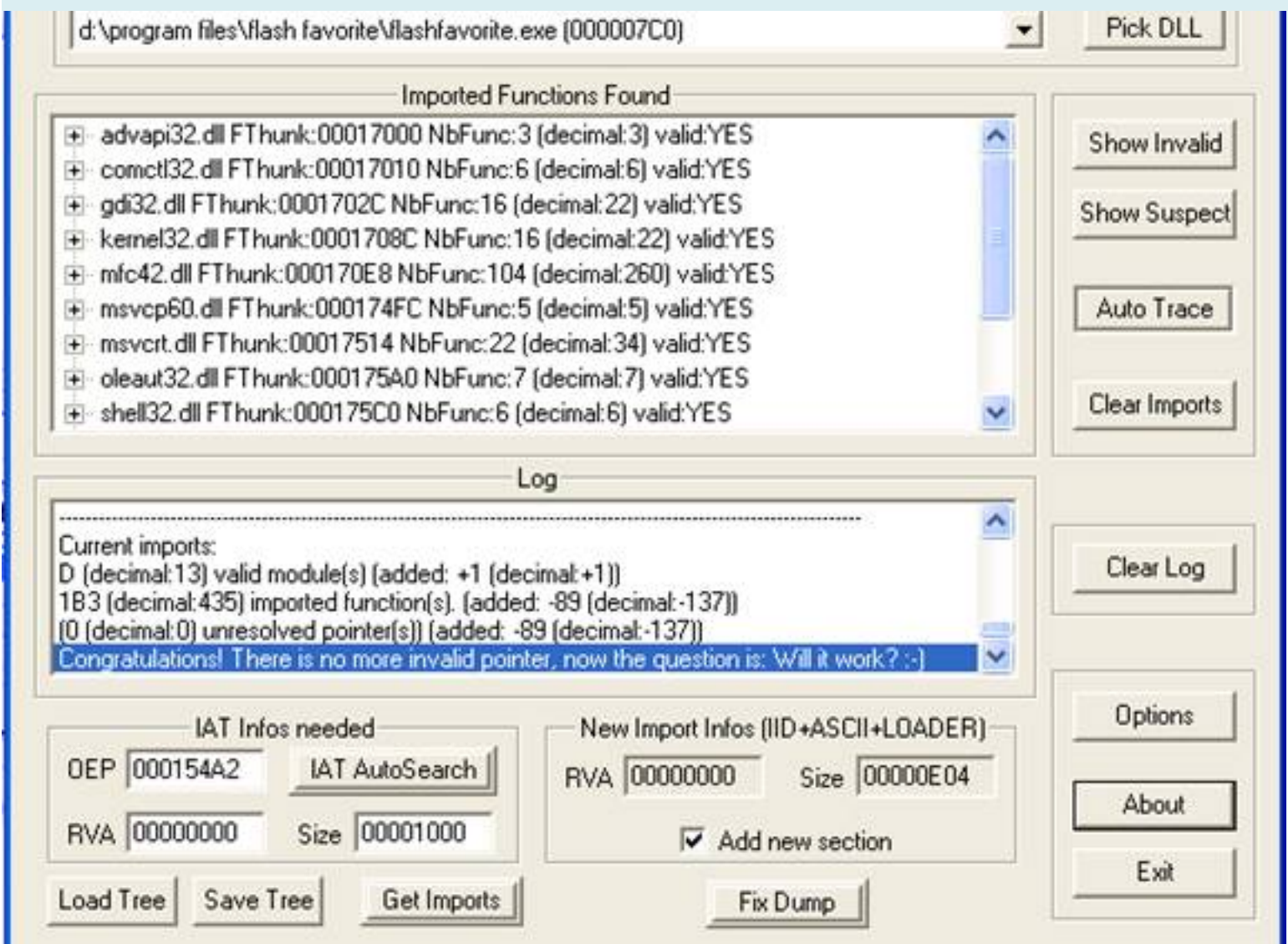


Press "OK"

After, press button "Show Invalid" again, and "Cut thunks"



We'll see the following:



Press button "Fix dump" file to fix IAT dumped.exe.

Run the file dumped_.exe ko be scrash.

Benina 15/5/2005

UnPaCkInG - ASPack 1.06b / 1.061b

Homepage: <http://crackmes.de>

Coder: FireWorx (FireWorx - Crackme 14 - Level 1)

Cracked File: Crackme14.exe (Borland Delphi 4.0 - 5.0)

Packed: ASPack 1.06b / 1.061b -> Alexey Solodovnikov

Crack Tool: OllyDbg 1.10b, PEiD 0.92, ImportREC 1.6F

Unpack Type: Manual

- Use PEiD we know is the program is using **ASPack PACK 1.06b / 1.061b -> Alexey Solodovnikov**

- Load up the program with Olly. Select No (not Analysis). I will come:

CODE

```
0041F6E0> 90  
Submit <== We  
here  
0041F6E1 75 00  
JNZ SHORT  
Crackme1.0041F6E3  
0041F6E3 - E9  
18190300 JMP  
Crackme1.00451000
```

- The F8 until we go to:

CODE

```
00451000 60  
PUSHAD <== We  
here  
00451001 E8  
00000000 CALL  
Crackme1.00451006
```

- Press F8 one command line to the next. In the book **Registers (FPU)** we have to lick your mouse in the value **ESP** and **Follow Indump**. **Hex dump** window will switch to new window.

Select the **first 4 bytes** of this window and lick your mouse to select **BreakPoint ==> Hardware on access ==> Dword**. Then press **F9** to us:

CODE

```
0045104F - FFE0
JMP EAX;
Crackme1.004419F8
<== We here
00451051 80BD
A2AE4300 0> Cmp
BYTE PTR SS:
[EBP +43 AEA2], 0
00451058 74 1D
JE SHORT
Crackme1.00451077
```

- The next F8 again we will go to OEP (right now the value of EAX is OEP)

CODE

```
004419F8 55 PUSH
EBP <== We here
004419F9 8BEC
MOV EBP, ESP
004419FB 83C4 F4
ADD ESP, 0C -
004419FE b8
F0184400 MOV
EAX,
Crackme1.004418F0
00441A03 E8
9C41FCFF CALL
Crackme1.00405BA4
```

- Note: After coming here is not using the mouse or do anything on the screen move up and down. Because the command line will be lost and we must do from the beginning.

- Then select the next **Plugins ==> Olly dump ==> dump debugged process**. After you select

will be completed at the Entry Point in the box -> Modify is the value of the OEP. Click OK and save as you like (the extension is. Exe).

- Keep raw Olly xuot in the process. Use Import REConstructor v1.6F load © 2001-2003 program. Fill in the OEP OEP value is above. Select IAT Auto Search, then select Get Imports and finally dump Fix (file with the fix is a new file is created above). In this program does not have any balance at all thunk.

- I will be a new file. File this run entirely possible. To improve our conduct over the cleaning and reduce the file size of the file after Fix dump (to file as small as possible. However, if you do not like you can ignore). To implement this process we use Delux LordPE v1.4. Load up the program, select rebuild PE. Select File Fix we dump on, lick the Open and we have a complete new file to run CRACK.

- Check the file with PEiD, we know the program is written in Borland Delphi 4.0 - 5.0.

Production: DriveScrubber 2.0a

Copyright by: Eugene Roshal

Homepage: www.iolo.com

Packed: ASPack 2.1 -> Alexey Solodovnikov

Coder: Borland Delphi 4.0 - 5.0

Tools: OllyDbg 1.09d LordPE + + 1.6 ImportREC

After you load up with Olly DriveScrubber, ignored (click Yes / Ok) Dialog in the notice ...
I will start here

CODE

```
004B3001> 60  
PUSHAD -> Olly  
are here Paused  
004B3002 E8  
72050000 CALL  
InstallD.004B3579  
004B3007 EB 33  
JMP SHORT  
InstallD.004B303C
```

Press F8 to run to 004B3002 .. Now look through the window Registers value **ESP = 0012FFC4**.
Right Click on the ESP -> Follow in dump.

Continue in the window after selecting Hex Follow in the dump will jump to

CODE

```
0012FFC4  
C7 14 E8  
77 78 E0  
12 00 F0  
88 FA 77  
00 F0 FD  
7F to  
ewxa. Duw.  
ŏy  
0012FFD4  
F0 4C 27  
EF C8 FF
```



```

12 00 04
45 53 80
FF FF FF
FF
öL'iEy.

```

```
ES € YYYY
```

Select the first 4 bytes (C7 14 E8 77), Right Click -> BreakPoint -> Hardware, on access -> Dword.

Now the press F9 to run, Olly will pause again. Then we will in ...

CODE

```

004B34F4 / 75 08
JNZ SHORT
InstallD.004B34FE
-> You are here
004B34F6 | b8
01000000 MOV
EAX, 1
004B34FB | C2
0C00 RETN 0C
004B34FE \ 68
70EB4700 PUSH
InstallD.0047EB70
004B3503 C3 RETN

```

Press F8 to trace to 004B3503, click Next again to 1 Return on

CODE

```

0047EB6E
47 DB 47;
Char 'G'
0047EB6F
00 DB 00
0047EB70
55 DB 55;
Char 'U' -
> Pause
here
0047EB71
8B DB 8B
DB
0047EB72
EC EC

```

Now do not Olly **Analyse** should not see the code ... press Ctrl + A analysing Wait will have done the following code

CODE

```

0047EB6B 00 DB 00
0047EB6C
80E84700 DD
InstallD.0047E880
0047EB70. 55 PUSH
EBP -> OEP
0047EB71. 8BEC
MOV EBP, ESP
0047EB73. 83C4 F4
ADD ESP, 0C -
0047EB76. 53 PUSH
EBX
0047EB77. B8
A8E84700 MOV
EAX,

```

InstallD.0047E8A8

Right here .. Using LordPE, select the Process and Full dump ... I will be dumped.exe file.
Continue using ImportREC, select Process fill **OEP is 0047EB70 - 400,000 = 7EB70** -> Click
IAT Auto Search.

RVA = 00082164 -> 00082200

Size = 0000074C -> 00000800

* Increase Size & reduce RVA aims to expand the home region Imports imports missing.

Ok .. Select ... Get Imports will ImportREC Import function of APIs for us ...

Because this information to select the balance should be the invalid -> Show Invalid -> Trace
Level -> still invalid -> Show Invalid Cut Trunks and go ... Fix dump.

And we have complete files unpack ... Check PeiD DriveScrubber will receive the code with the
Borland Delphi 4.0 - 5.0.

Practice unpack ASPack victim with Opera 7:23

Using PeiD we are following information: ASPack 2:12 -> Alexey Solodovnikov

Okai .. Use OllyDbg load up Opera. Press CTRL-B, type 61 and press CTRL-L to find the following code:

CODE

```
007303AF. 61
```

```
POPAD
```

```
007303B0. 75
```

Original Vietnamese text:

Bây giờ thì nhấn F9 để run chương trình, Olly sẽ pause ngay tại 007303AF nơi ta vừa set bpx.

[+ Suggest a better translation](#)

```
EAX, 1
```

```
007303B7. C2
```

```
0C00 RETN 0C
```

```
007303BA> 68
```

```
31B16600 PUSH
```

```
opera.0066B131
```

```
007303BF. C3
```

```
RETN
```

Set bpx 007303AF. Now the press F9 to run the program, Olly will pause at 007303AF where we've set bpx.

Now press the F8 to find OEP Trace ... Every time a press F8 you will find the following code:

CODE

```
0066B131 00 DB
```

```
00
```

```
0066B132. 8BEC
```

```
MOV EBP, ESP
```

```
0066B134. 6A FF
```

```
PUSH -1
```

```
0066B136. 68
```

```
C0716C00 PUSH
```

```
opera.006C71C0
```

```
0066B13B. 68
```

```
D8B96600 PUSH
```

```
opera.0066B9D8;
```

```
SE handler
```

```
installation
```

```
0066B140. 64:
```



```

A1 00000000>
MOV EAX, DWORD
PTR FS: [0]
0066B146. 50
PUSH EAX

```

**.. Here Zombie mun a little note ... the first Zombie made it to the trace here is just a Byte 0 (DB 00), then press Ctrl-A to analysy. When the Code will appear more clearly. However hì hì After pressing CTRL-A in time to trace 0066B131 you'll see here is the code PUSH EBP ... And the next time it is a DB 00: (thui shelves .. it .. Every who addr 0066B131 is called ...

Now the dump is full. Olly resources for the state Paused 0066B131, use LordPE load up Opera.exe process. RightClick select Full dump. It will save files in the folder dumped.exe Opera.exe.

Now it continues to fix the IAT. ImportREConstructor a tool to use for this ... As on. Select Process Opera.exe.

OEP is now: **0066B131-26B131 = 400000** ...

Enter OEP. Select IAT auto search ... It will enter the following information: **RVA = 0027A000 Size = 000007EC.**

But little thui ... Select Imports Get it imports all the APIs ... Done properly fix dump, select File dumped.exe This full dump -> Fixed dumped_.exe the file -> File is correct and Asia ...

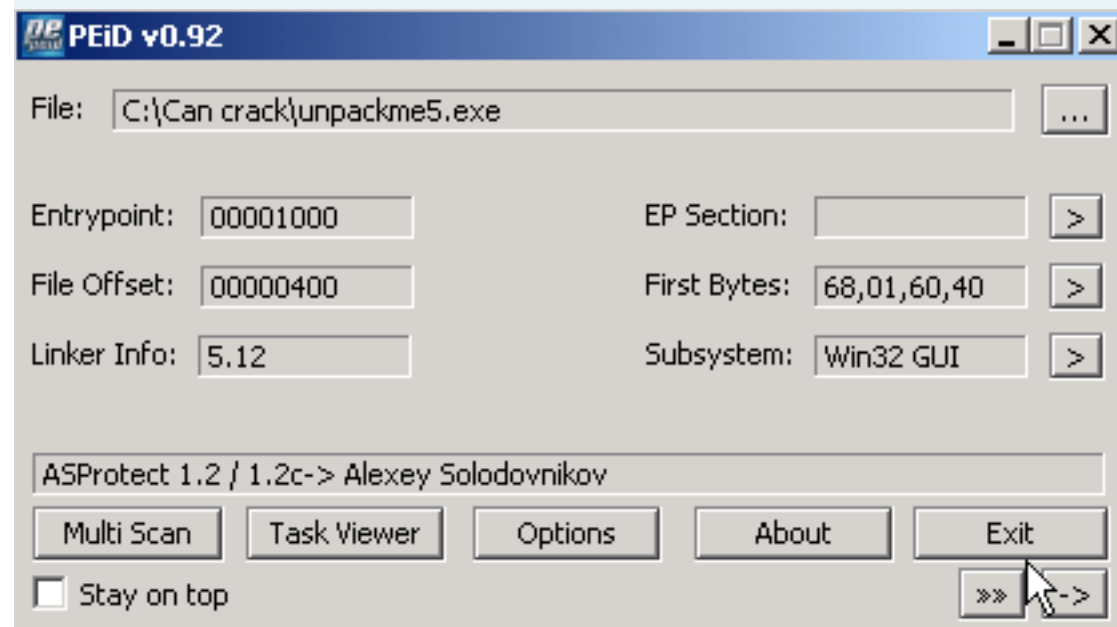
Learning unpack Asprotect 1.2/1.2c

Target: unpackme # 5

Packer: Asprotect 1.2/1.2c

Download: <http://nhandan.info/hacnho>

First used to see PEID pack it with anything.



We see it in the pack Asprotect 1.2/1.2c. Use OllyDbg open file. Press Shift-F9 have been 21 times, the program runs. Load the program in OllyDbg (Ctrl-F2). Press Shift-F9 continuous 20 times. We at 88169A.

0088169A	FF02	INC DWORD PTR DS:[EDX]
0088169C	EB E8	JMP SHORT 00881686
0088169E	8BC1	MOV EAX,ECX
008816A0	2BF6	SUB ESI,ESI
008816A2	64:8F06	POP DWORD PTR FS:[ESI]
008816A5	5E	POP ESI
008816A6	E8 00000000	CALL 008816AB
008816AB	33C2	XOR EAX,EDX
008816AD	8B3C24	MOV EDI,DWORD PTR SS:[ESP]
008816B0	58	POP EAX

Press Alt-M to open Memory map. Set Breakpoint in section 2 (code).

00310000	00006000				Map	R	R	
00320000	00004000				Map	R E	R E	
003E0000	00002000				Map	R E	R E	
003F0000	00001000				Priv	RW	RW	
00400000	00001000	unpackme		PE header	Imag	R	RWE	
00401000	00001000	unpackme		code				
00402000	00001000	unpackme						
00403000	00001000	unpackme						
00404000	00002000	unpackme	.rsrc	resources				
00406000	0000C000	unpackme	.data	data, import				
00412000	00001000	unpackme	.data					
00420000	00103000							
00530000	0008A000							
00830000	00001000							
00860000	00019000							
00880000	00004000							
77C70000	00001000	GDI32		PE header				
77C71000	0003B000	GDI32	.text	code, import				
77CAC000	00001000	GDI32	.data	data				
77CAD000	00001000	GDI32	.rsrc	resources				
77CAE000	00002000	GDI32	.reloc	relocation				
77CC0000	00001000	RPCRT4		PE header				
77CC1000	00067000	RPCRT4	.text	code, import				
77D28000	00007000	RPCRT4	.orpc	code				
77D2F000	00001000	RPCRT4	.data	data				
77D30000	00001000	RPCRT4	.rsrc	resources				
77D31000	00004000	RPCRT4	.reloc	relocation				
77D40000	00001000	user32		PE header				
77D41000	0005C000	user32	.text	code, import				
77D9D000	00002000	user32	.data	data				
77D9F000	0002B000	user32	.rsrc	resources				
77DCA000	00003000	user32	.reloc	relocation				

Actualize

View in Disassembler Enter

Dump in CPU

Dump

Search Ctrl+B

Set break-on-access F2

Set memory breakpoint on access

Set memory breakpoint on write

Set access ▶

Copy to clipboard ▶

Sort by ▶

Appearance ▶

Close Memory map. Press Shift-F9. We OEP at 401,000. Using OllyDump.

Entry Point: 1000 -> Modify: 1000 Get EIP as OEP Cancel

Base of Code: 1000 Base of Data: 2000

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
	00001000	00001000	00001000	00001000	C0000040
	00001000	00002000	00001000	00002000	C0000040
	00001000	00003000	00001000	00003000	C0000040
.rsrc	00002000	00004000	00002000	00004000	C0000040
.data	0000C000	00006000	0000C000	00006000	C0000040
.data	00001000	00012000	00001000	00012000	C0000040

☒ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Them. Happy happy.

tländn

<<<<->>>> XP Utilities Lite <<<<->>>>

Content PEiD we know is the program pack with **ASProtect 1:23 RC4 Registered -> Alexey Solodovnikov**

Load up the program with Olly. Select No (not Analysis). I will come:

QUOTE

```
00401000> 68
01B05200 PUSH
Autils.0052B001
<=== We here
00401005 E8
01000000 CALL
Autils.0040100B
```

As programs Anti-Debug so we must disable it by using your Olly Plugin (IsDebugPresent) choose Hide. Or, press **Ctrl-G** in the Hex dump window, go to **7FFDF002**, Ok people, at this location to see the value 01 to 00.

This made the per-call program with Olly.Nhan Shift-F9 28 times (29 times will run the program):

QUOTE

```
009F3D03
3100 XOR
DWORD PTR
DS:
[EAX],
EAX <====
We came
here
```

Or when one of us to italy the window stack, when you see the line:

0012FF5C 00A6A530 ASCII "lgJ4vADQtuw"

Contest appear chu italy, see the line:

0012FF80 00400000 ASCII "MZIP"

appear stopped contest (as to when this is the time to 28)

People continue next Shift-Alt-F7.Nhan M to open a window onto Memory map. Select the engine Owner: Contains and Autils: code. Lick mouse to select the Set breakpoint on memory access. Then the next F9 we will return to main screen right here:

QUOTE

```
0040619C
- FF25
80724C00
JMP DWORD
PTR DS:
[4C7280]
<=== We
here
004061A2
8BC0 MOV
EAX, EAX
004061A4
- FF25
7C724C00
JMP DWORD
PTR DS:
[4C727C]
004061AA
8BC0 MOV
EAX, EAX
004061AC
- FF25
78724C00
JMP DWORD
PTR DS:
[4C7278]
004061B2
8BC0 MOV
EAX, EAX
004061B4
- FF25
74724C00
JMP DWORD
PTR DS:
[4C7274]
004061BA
8BC0 MOV
EAX, EAX
```

Content LordPE delux 1.4 Adutils.exe load the file, select the full dump, I can file dumped.
exe

Human F8 to jump ahead to the next order. One of F8 to jump to order RETN and through to the
next order. Then we will go to here:

QUOTE

```
00406271 A3
68564C00 MOV
DWORD PTR DS:
[4C5668], EAX
<=== We here
00406276 A1
68564C00 MOV
EAX, DWORD PTR
DS: [4C5668]
0040627B A3
A8204C00 MOV
DWORD PTR DS:
[4C20A8], EAX
00406280 33C0
XOR EAX, EAX
00406282 A3
AC204C00 MOV
DWORD PTR DS:
[4C20AC], EAX
00406287 33C0
XOR EAX, EAX
00406289 A3
B0204C00 MOV
DWORD PTR DS:
[4C20B0], EAX
0040628E E8
C1FFFFFF CALL
Adutils.00406254
A4204C00 BA
00406293 MOV
EDX,
Adutils.004C20A4
00406298 8BC3
```

```
MOV EAX, EBX
<=====
Remember value
EAX
0040629A E8
BDDAFFFF CALL
Adutils.00403D5C
0040629F 5B POP
EBX
004062A0 C3
RETN <=== The
next F8 when
you come here.
```

And here we find the value of EAX = 004C1250 After the F8 RETN order for us to here

QUOTE

```
004C1781 0000
ADD BYTE PTR
DS: [EAX], AL
004C1783 E8
D84AF4FF CALL
Adutils.00406260
<= ===== OEP
C1783
(temporary)
004C1788 A1
40414C00 MOV
EAX, DWORD PTR
DS: [4C4140]
```

We find the value is unavailable OEP **C1783**

Stay in main screen. Press Ctrl-B, and go to the FF 25, then press Ctrl-L continue to search to đoạn:

QUOTE

```

00401254
- FF25
34724C00
JMP DWORD
PTR DS:
[4C7234]
0040125A
8BC0 MOV
EAX, EAX
0040125C
- FF25
30724C00
JMP DWORD
PTR DS:
[4C7230]
00401262
8BC0 MOV
EAX, EAX
00401264
- FF25
2C724C00
JMP DWORD
PTR DS:
[4C722C]

```

So we predicted IAT of programs

$RVA = 4C7234 - 400,000 = C7234$ (here we choose is C7000)

Size = 900

Use Import REConstructor v1.6F © 2001-2003 Adutils.exe load file. Prefill values above the **IAT infos needed**. Then the Get Imports, the next Show Invalid, then the **Trace level 1 (disasm)**. After the trace is complete, the next **Show Invalid**, then lick the mouse and choose to continue **Plugin Tracers** ==> **ASProtect 1:23 RC4**. Trace Once completed, the next **Show Invalid**, then lick the mouse and choose to continue **Plugin Tracers** ==> **ASProtect 1:22**. Finally the Show Invalid time more and more people Cut thunks. Then dump Fix (file with the fix is a new file is created above). Here we have a file dumped_.exe
Load file dumped_.exe up and we will go to here:

QUOTE


```
004C1778
0000 ADD
BYTE PTR
DS:
[EAX], AL
004C177A
0000 ADD
BYTE PTR
DS:
[EAX], AL
004C177C
0000 ADD
BYTE PTR
DS:
[EAX], AL
004C177E
0000 ADD
BYTE PTR
DS:
[EAX], AL
004C1780
0000 ADD
BYTE PTR
DS:
[EAX], AL
004C1782
00E8 ADD
AL, for I
am here
===
```

Notice is now here we have a case of 11 bytes. Complete Stolen Bytes. Once completed, we are now:

QUOTE

```
004C1778 55
PUSH EBP <===
real OEP
004C1779 8BEC
MOV EBP, ESP
004C177B b8
50124C00 MOV
EAX,
dumped_.004C1250
<== Instead
value EAX with
our values at
the heart
004C1780 83C4
F4 ADD ESP, 0C -
```

Once completed we will get real value OEP. Here:

OEP = 4C1778 - 400,000 = C1778

Must click and select: Copy to executable ==> All modification. Select Copy All next. At the new screen you must click the mouse and select Save File next. Save the file again with new names.

Here we have dumped_1.exe file. Content LordPE deluxe 1.4 load dumped_1.exe file, select PE Editor Dumped_1.exe selected file. At the new window, in the Entry Point to see the new value is C1778. Human Save and then OK.

Then select rebuild PE to initiate the process and remove RAC dimple Filesize programs then run honaf all good, not CRASH

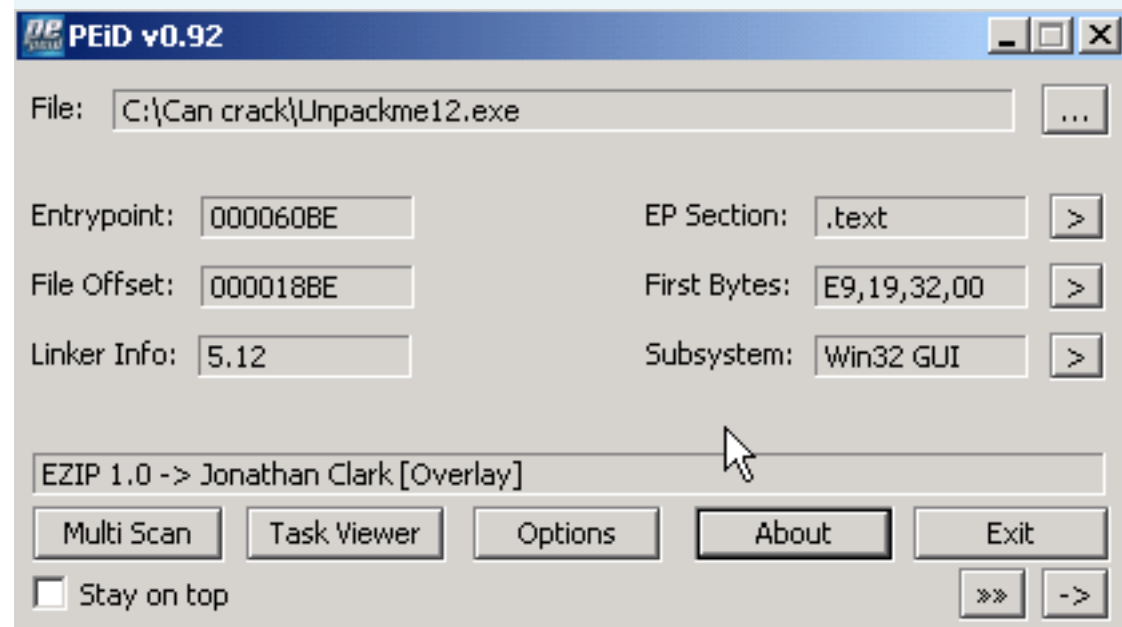
Learning unpack Ezip 1.0

Target: unpackme # 12

Packer: Ezip 1.0

Download: <http://nhandan.info/hacnho>

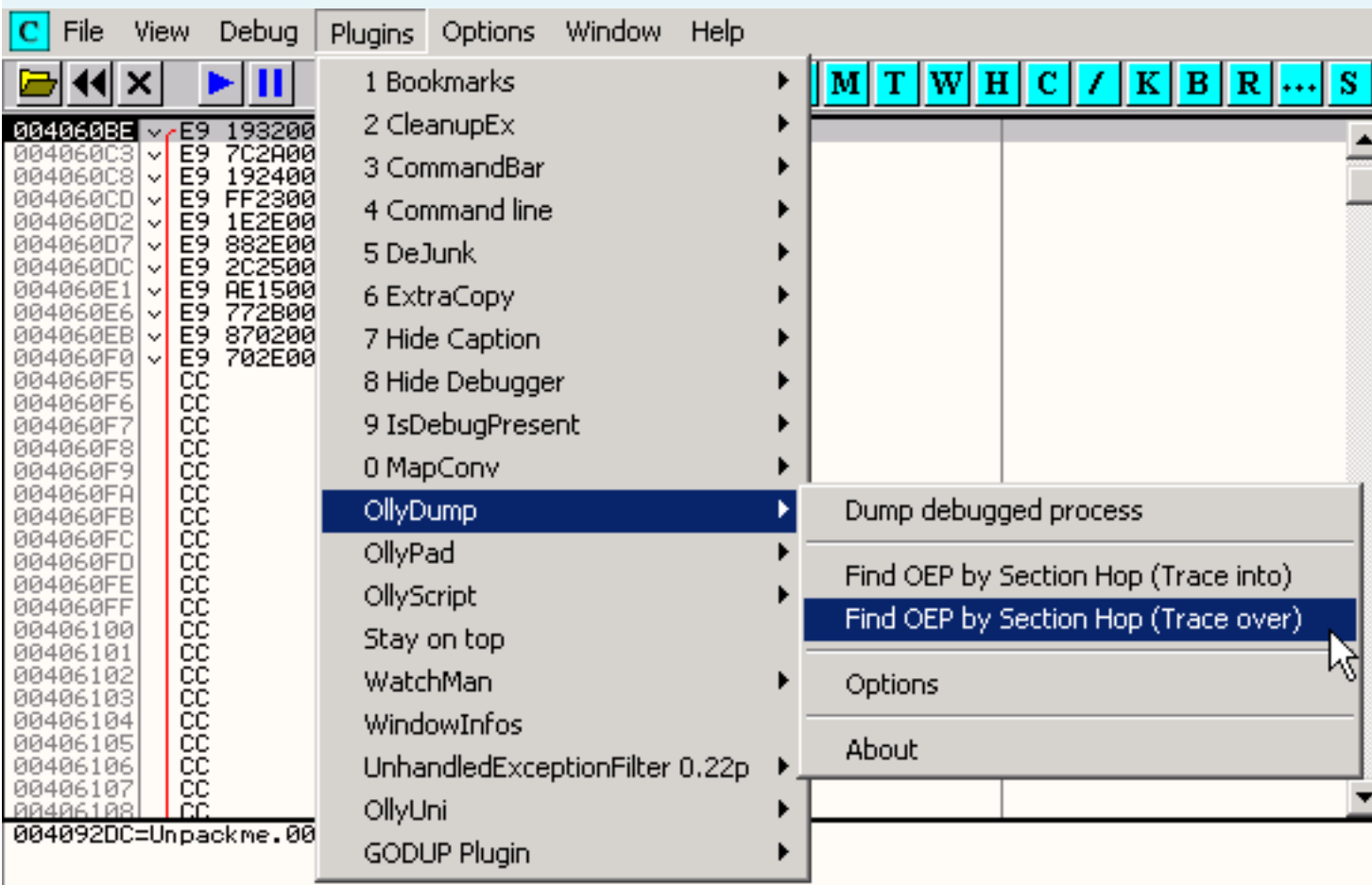
Top PEID handy it is to see the pack with ge.



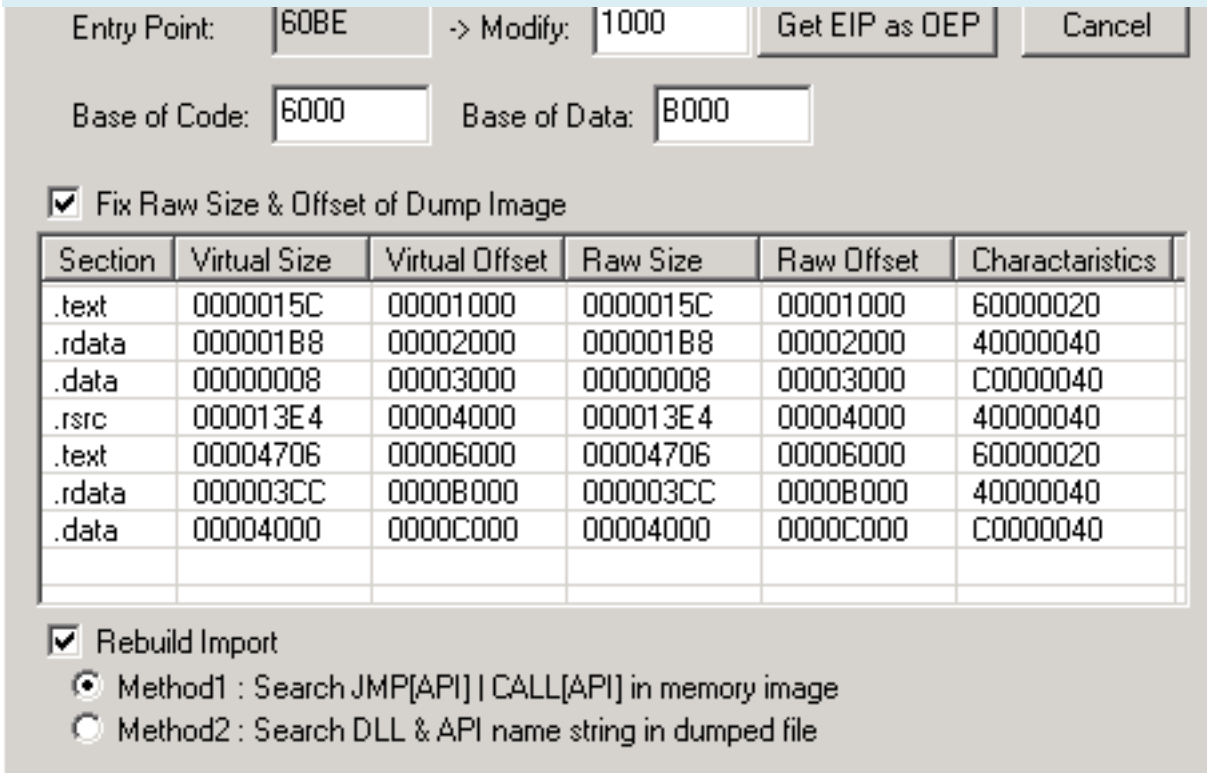
We see it in the pack Ezip 1.0.

Method 1:

Dung OllyDbg open the file. Dung OllyDump -> Find OEP by Section Hop (Trace Over)

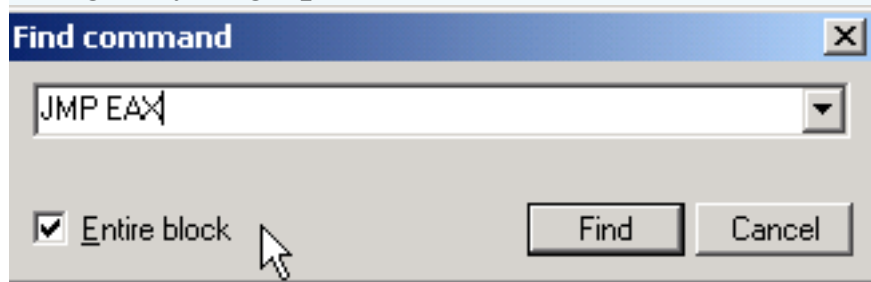


We will stay at 401,000. This main lr OEP's trenh. Dung OllyDump to dump file.



Method 2:

Dung OllyDbg open the file. Press Ctrl-F. Enter vxo JMP EAX. Click Find.



We at 409,688. Set Breakpoint (F2). Press F9 to run. Chapter trenh will break at 409,688. Press F8. Chapter trenh OEP will at 401,000. Dung OllyDump as 1 month.

So lr completed. Happy happy.
tlandn

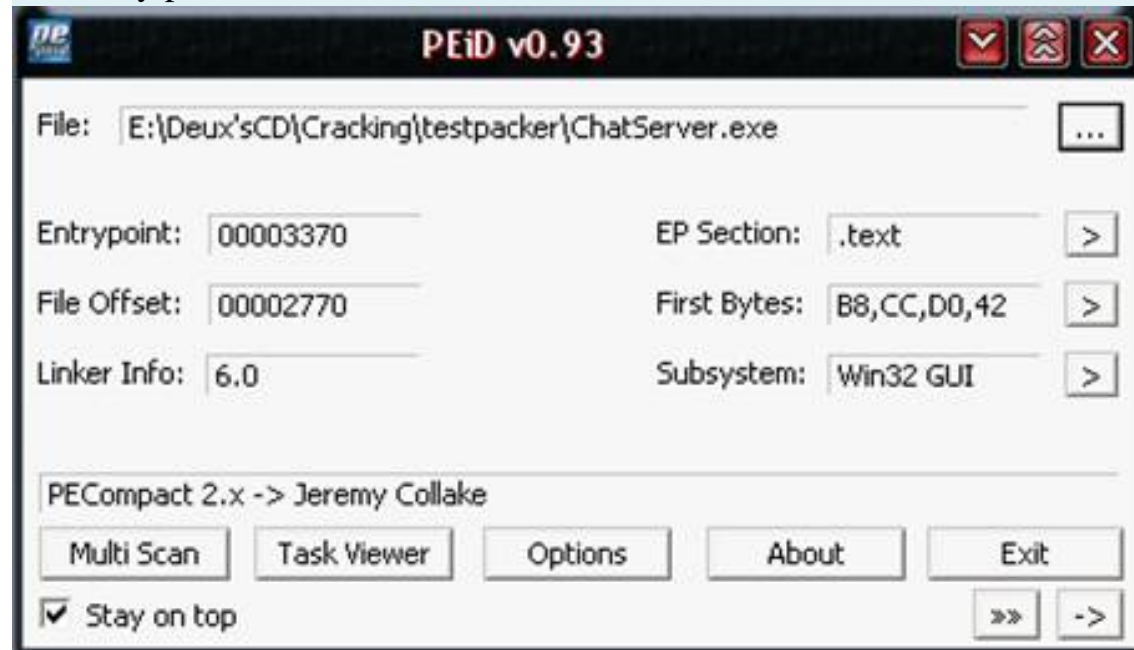
Unpack manual PECompact version 2.55 (released 2005-04-28)

Tool: - OllyDebugger (OllyDump plugin): Look for OEP, dump, fix IAT.

- 0.93 PEID: detect packer.

Test file: use compile the file from VC + + 6.0.

I-Identify packer:

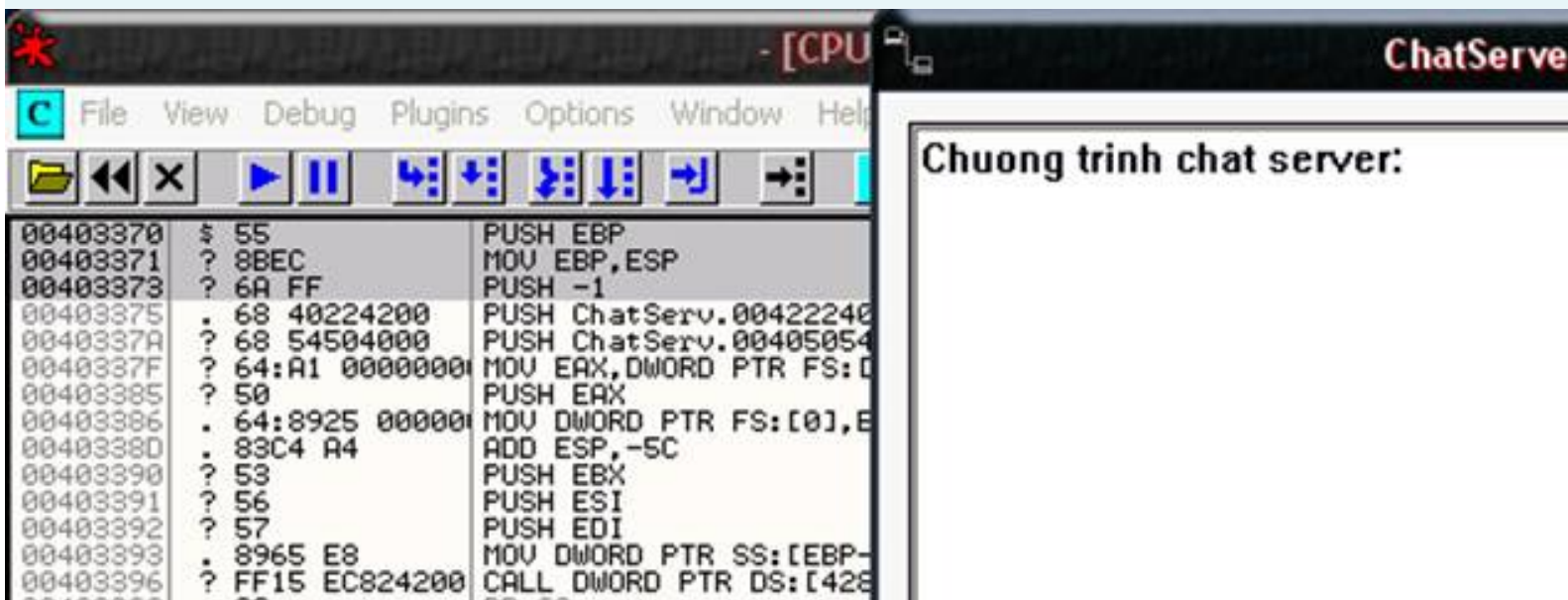


II-unpack:

Load program to Ollydebugger.

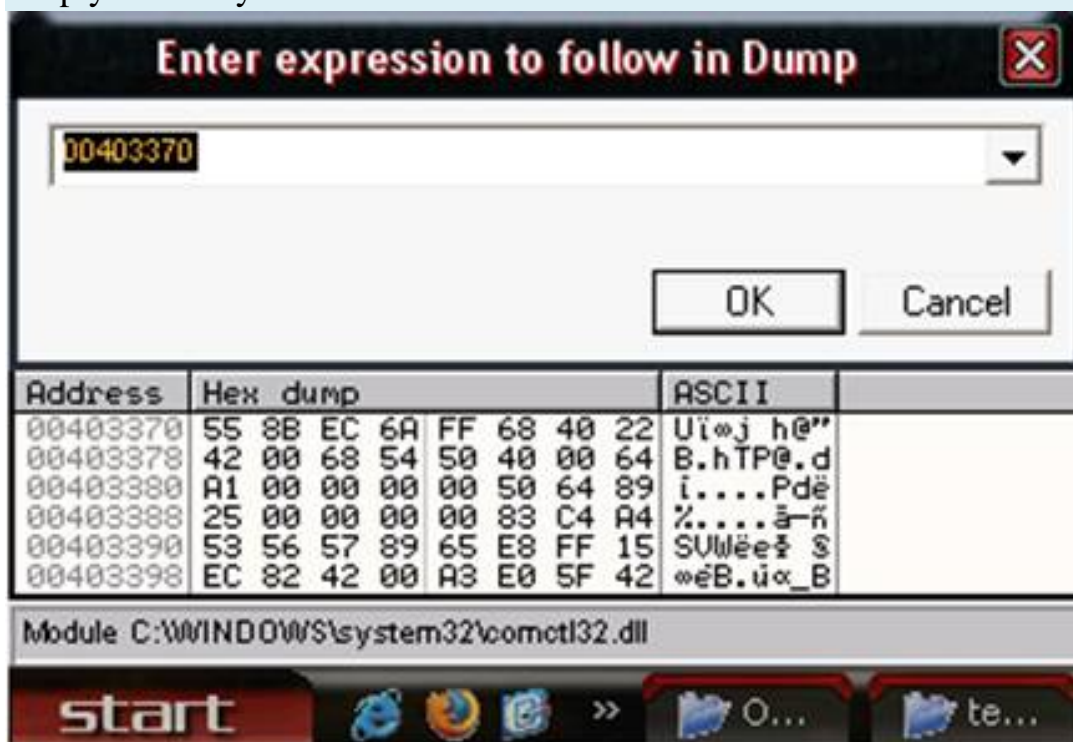
00403370	\$ B8 CCD04200	MOV EAX, ChatServ.0042D0CC
00403375	. 50	PUSH EAX
00403376	. 64:FF35 000000	PUSH DWORD PTR FS:[0]
0040337D	. 64:8925 000000	MOV DWORD PTR FS:[0],ESP
00403384	. 33C0	XOR EAX,EAX
00403386	. 8908	MOV DWORD PTR DS:[EAX],ECX
00403388	. 50	PUSH EAX
00403389	. 45	INC EBP
0040338A	. 43	INC EBX
0040338B	. 6F	OUTS DX,DWORD PTR ES:[EDI]
0040338C	. 6D	INS DWORD PTR ES:[EDI],DX
0040338D	. 70 61	JO SHORT ChatServ.004033F0
0040338F	. 637432 00	ARPL WORD PTR DS:[EDX+ESI],SI
00403393	. 15 F4C2A7A0	ADC EAX,A0A7C2F4
00403398	. 0F	DB 0F
00403399	. 88	DB 88
0040339A	. 7B	DB 7B
0040339B	. 5B	DB 5B
0040339C	. EB	DB EB
0040339D	. 0B	DB 0B

Some things to note, if F9 to run the program will run normally, but the trace will fail exceptions, and will be the endless loop, this is perhaps a way anti debug but I do not spare carefully to see this, it should always F9 to see how it runs.

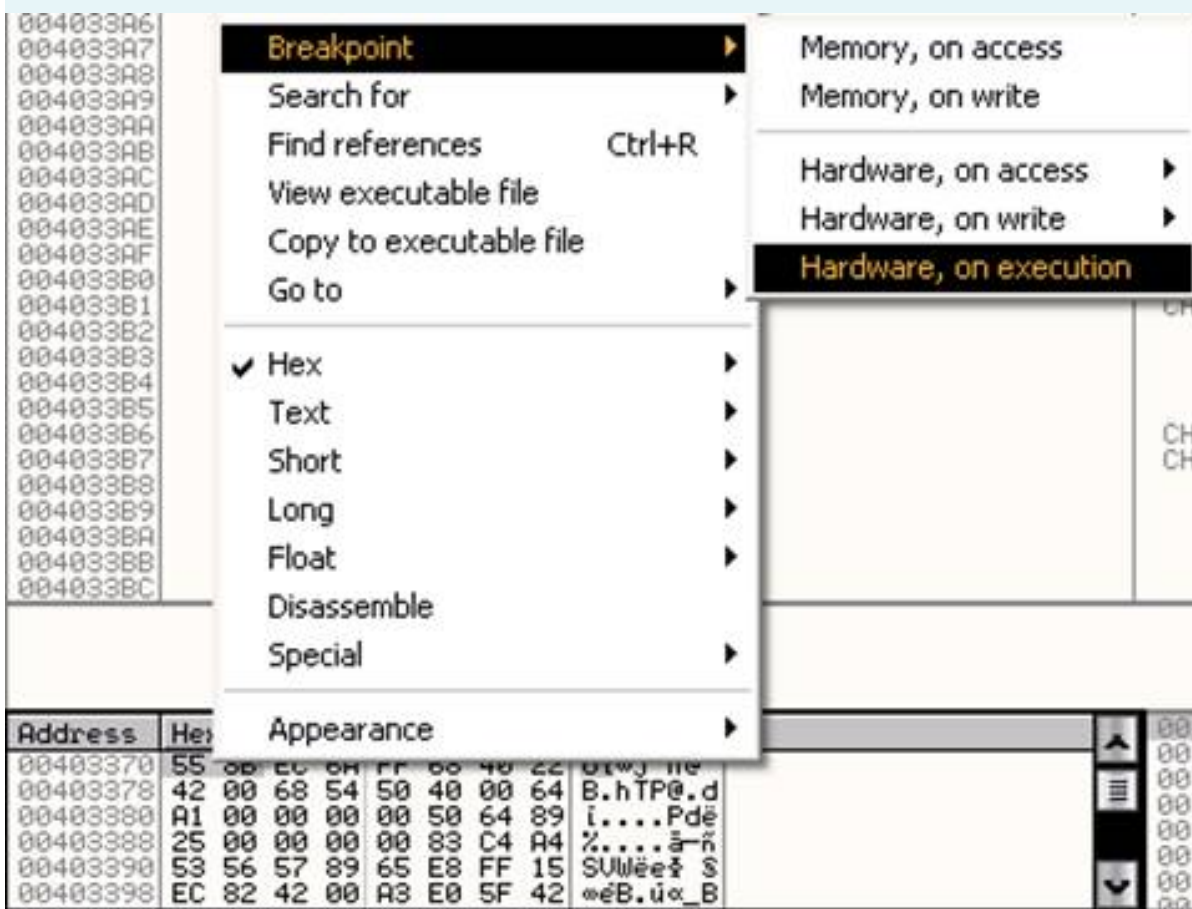


No need to look far from where, is 00403370 OEP of the program, the problem is how to stop at places where this can not trace the command line program to set break point.

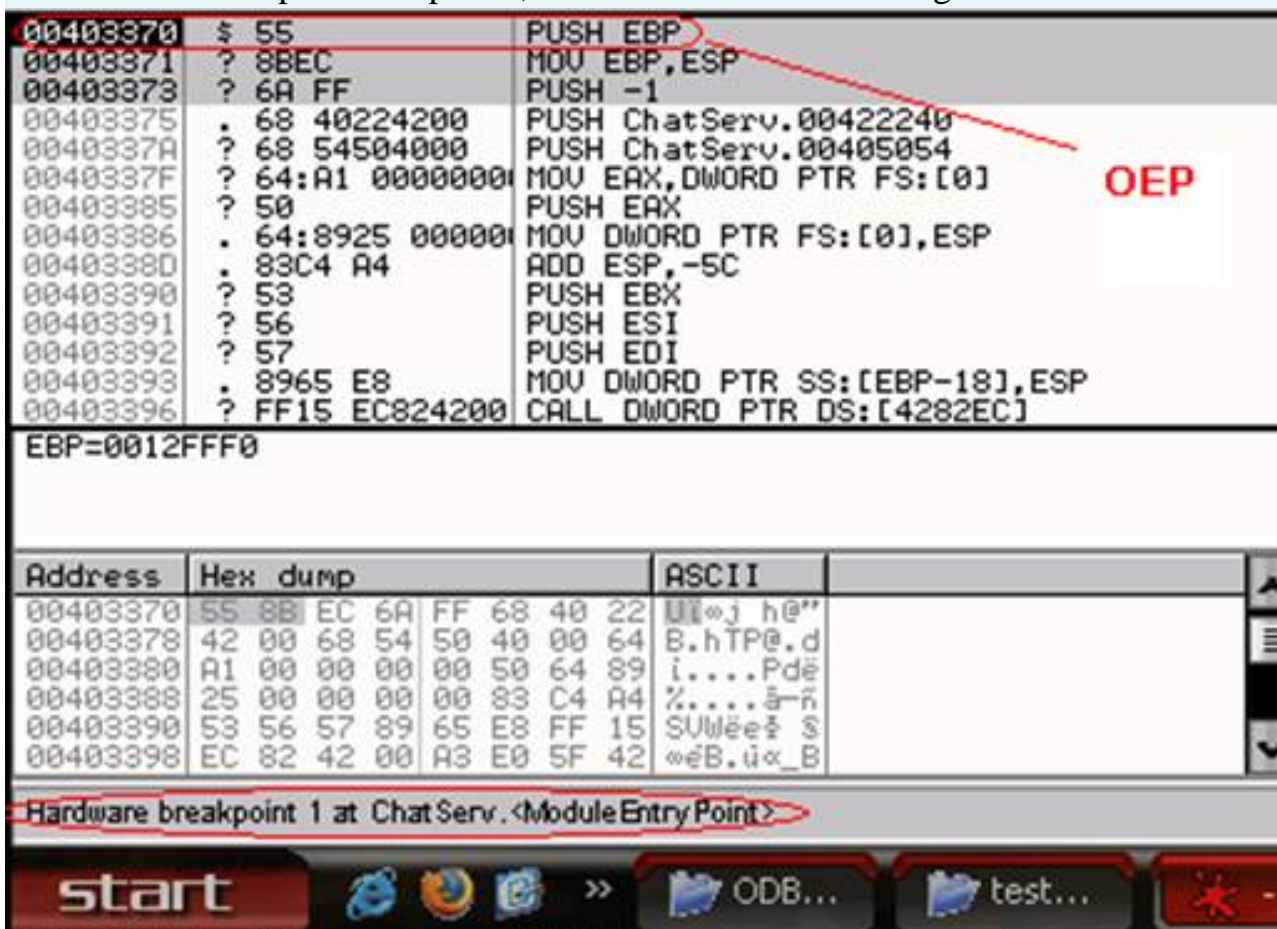
Because with manual unpack often say a machine is to find pushpad, set breakpoint, or exception last enable memory to set breakpoint, but with this packer, few things are superfluous. However, so there simply are many more:



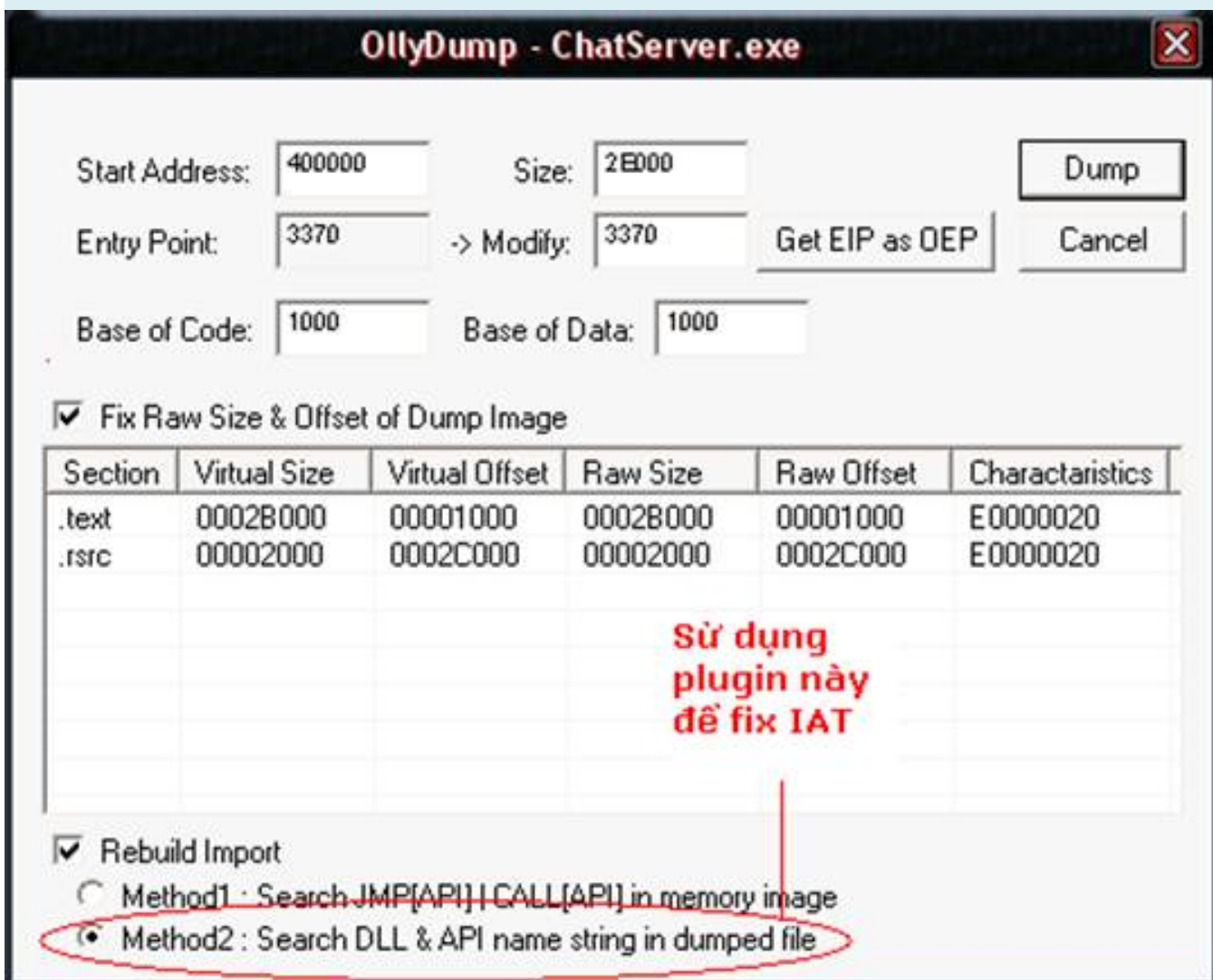
Ctrl + G-> OEP to enter to that position. Set break point:



Sauk set the breakpoint completed, F9 to run and break waiting in OEP

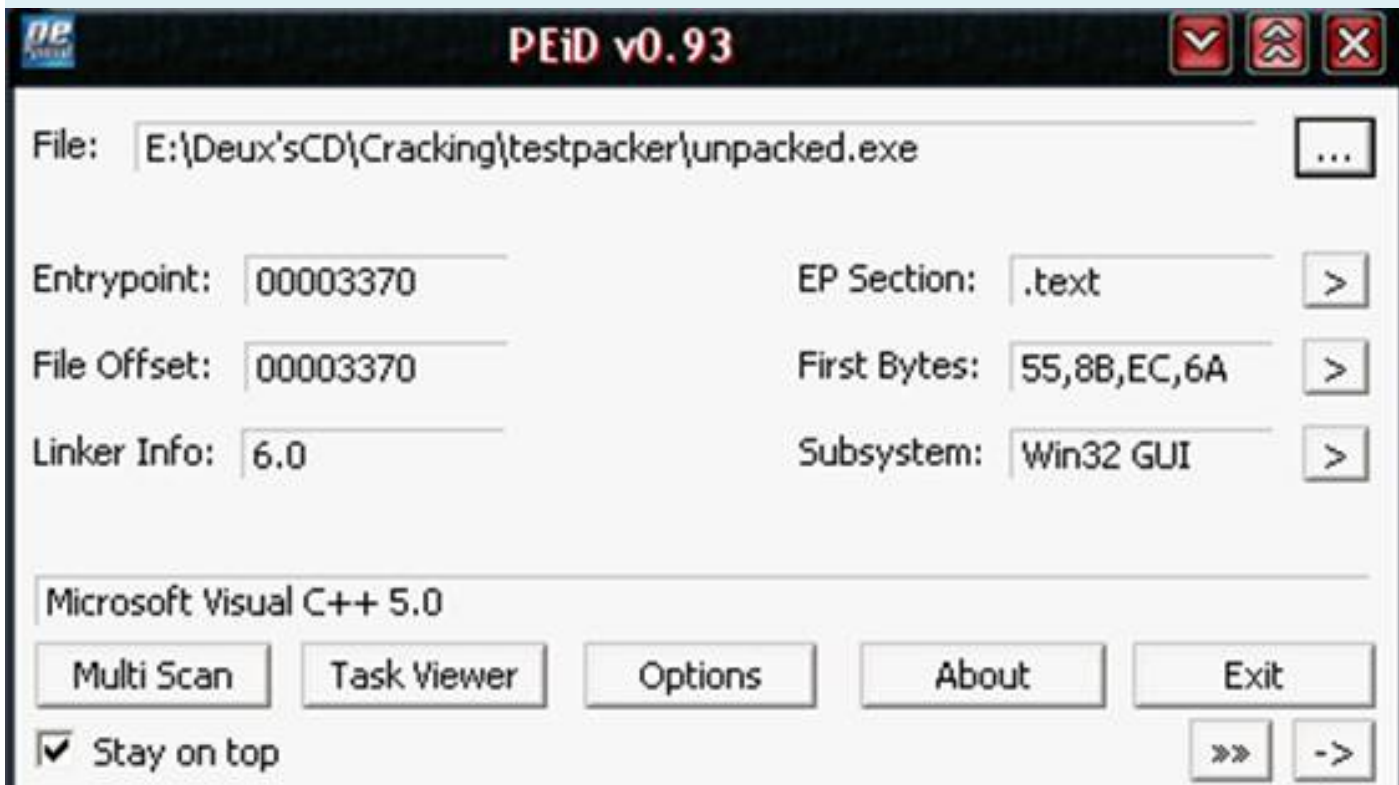


Now use OllyDump, there are some things to note:



After you save, like dimple exe files you can use LordPE.

Results:



Some special OEP also with initial EP, set break point in that run and is ok.

Should not be used to fix IAT ImpRect ..

Tutorial by Deux.

Contact me: deux@surfy.net

Reverse Engineering Association: www.reaonline.net

EOF -----

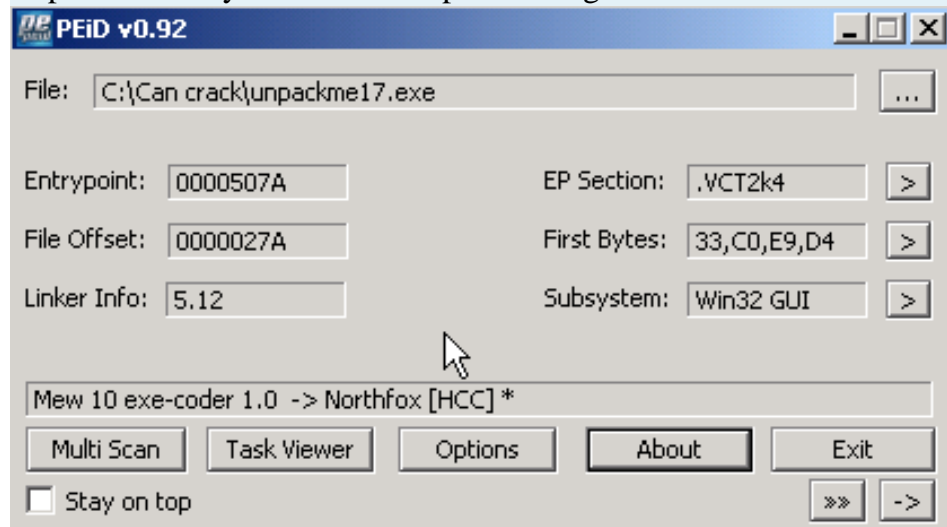
Learning unpack Mew 10 exe-coder 1.0

Target: unpackme # 17

Packer: Mew 10 exe-coder 1.0

Download: <http://nhandan.info/hacnho>

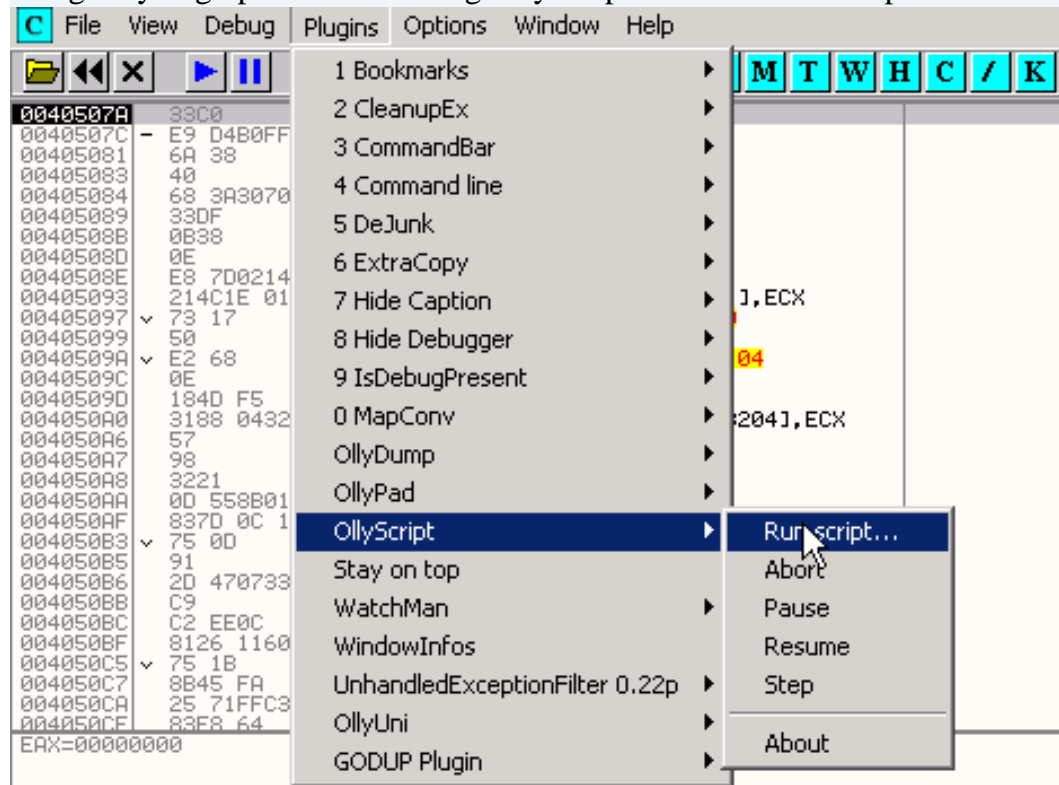
Top PEID handy it is to see the pack with ge.



We see it in the pack Mew 10 exe-coder 1.0.

Method 1:

Dung OllyDbg open the file. Dung OllyScript. I have KCM in zip file.



Select File Chapter mew10_1_0.txt trenh OEP will stop at 401,000. Dung OllyDump to dump and Imprec to fix IAT.

00401000	6A 40	PUSH 40	OEP Reached !
00401002	68 00304000	PUSH unpackme.00403000	ASCII "--=[UCT-Viet Cracking Team
00401007	68 33304000	PUSH unpackme.00403033	ASCII " --+=[SoftWare iFORMAT
0040100C	6A 00	PUSH 0	
0040100E	E8 7D020000	CALL unpackme.00401290	JMP to user32.MessageBoxA
00401013	6A 00	PUSH 0	
00401015	E8 4C020000	CALL unpackme.00401266	JMP to kernel32.GetModuleHandleA
0040101A	6A 01	PUSH 1	
0040101C	6A 00	PUSH 0	
0040101E	6A 00	PUSH 0	
00401020	50	PUSH EAX	
00401021	E8 68000000	CALL unpackme.0040108E	
00401026	6A 40	PUSH 40	
00401028	68 F5314000	PUSH unpackme.004031F5	ASCII "Final paroles!"
0040102D	68 04324000	PUSH unpackme.00403204	ASCII "More cracks, please contact
00401032	6A 00	PUSH 0	
00401034	E8 57020000	CALL unpackme.00401290	JMP to user32.MessageBoxA
00401039	50	PUSH EAX	
0040103A	E8 21020000	CALL unpackme.00401260	JMP to kernel32.ExitProcess
0040103F	55	POP EBP	

Entry Point: 507A

-> Modify: 1000

Get EIP as OEP

Cancel

Base of Code: 0

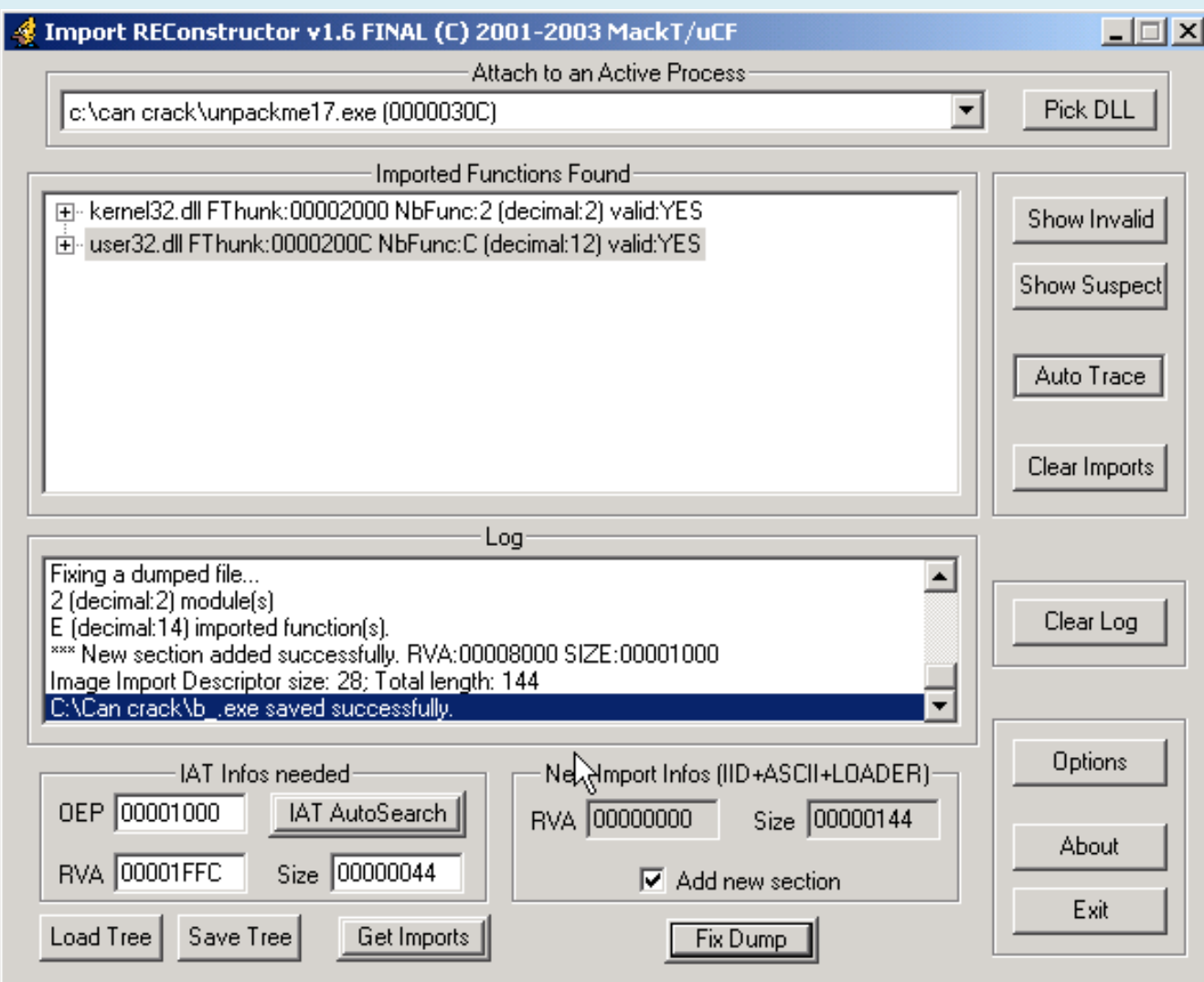
Base of Data: C

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
.hacnho	00004000	00001000	00004000	00001000	E00000C0
.VCT2k4	00001200	00005000	00001200	00005000	E0000020

☒ Rebuild Import

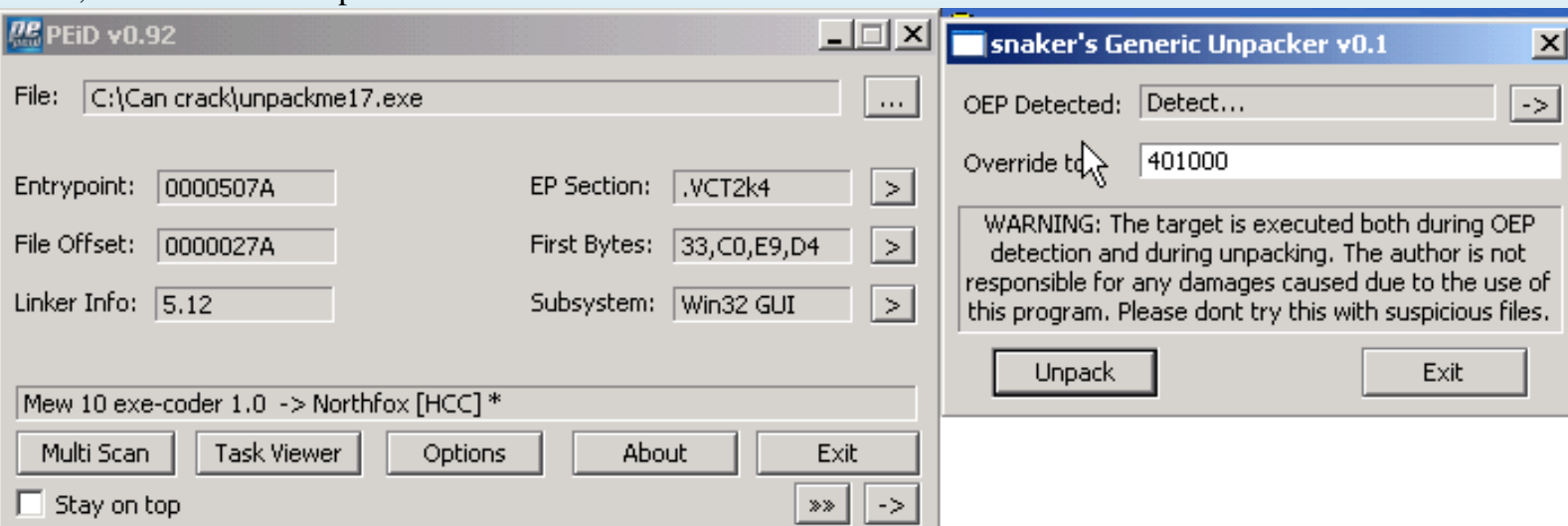
- ☒ Method1 : Search JMP[API] | CALL[API] in memory image
☐ Method2 : Search DLL & API name string in dumped file



Click Fix dump file has chosen to dump OllyDump.

Method 2:

After that the OEP's trenh lr 401,000. You can use PEID Generic Unpacker to unpack. Enter vro textbox to Override lr 401,000. Then click unpack



How fast nry with conditions that must lr OEP.

3 months:

This is a lr I think to OEP tem. Dung OllyDbg open the file. Press Alt-M to open Memory map. Set breakpoint in section 2 as henh

004000AC 00 00 00 00 00 00 00 00 00 00 00 00
004000BC 00 00 00 00 00 00 00 00 00 00 00 00
004000CC 00 00 00 00 00 00 00 00 00 00 00 00
004000DC 00 00 00 00 00 00 00 00 00 00 00 00
004000EC 00 00 00 00 00 00 00 00 00 00 00 00

Command

Break-on-access when writing to [00401000]

tlandn

Tam ngắ: <http://kah.serveftp.net/upload/UnpackMes/Neolite2.exe>

Tool: Ollydbg Ollydump + 2.2x

Author: Computer_Angel

Load program to Olly, we have:

QUOTE

```
00440172> $ / E9
A6000000 JMP
Neolite2.0044021D
00440177 |
F21C4400 DD
Neolite2.00441CF2
0044017B. |
78004400 DD <&
kernel32.
LoadLibraryA>
0044017F. |
7C004400 DD <&
kernel32.
GetProcAddress>
00,440,183th |
00000000 DD
00000000
00,440,187th |
F23A0000 DD
00003AF2
0044018B |
34024400 DD
Neolite2.00440234
0044018F. | 4E 65
6F 4C 6> ASCII
"NeoLite
Executab"
0044019F. | 6C 65
20 46 6> ASCII
```

```
"le File
compress"
004401AF. | 6F 72
0D 0A 4> ASCII
"or Copyright (c"
004401BF. | 29 20
31 39 3> ASCII
") 1998.1999
NeoW"
004401CF. | 6F 72
78 20 4> ASCII
"orx portion
Inc."
004401DF. | 73 20
43 6F 7> ASCII
"s Copyright ©"
004401EF. | 31 39
39 37 2> ASCII
"Lee Ha 1997-
1999"
004401FF. | 73 69
75 6B 0> ASCII
"siuk All Rights"
0044020F. | 20 52
65 73 6> ASCII
"reserved.", 0
0044021C. | 00 DB
00
0044021D> \
8B4424 04 MOV
EAX, DWORD PTR
SS: [ESP +4]
00440221st 2305
83014400 AND
```



```
EAX, DWORD PTR
DS: [440183]
00440227. E8
ED040000 CALL
Neolite2.00440719
```

F8 order to trace the JMP, we desire to trace CALL Neolite2.00440719, look down below 1 xiu, we find:

QUOTE

```
0044022C.
FE05
1C024400
INC BYTE
PTR DS:
[44021C]
00440232.
FFE0 JMP
EAX
00440234.
803D
1C024400>
Cmp BYTE
PTR DS:
[44021C],
0
```

Set Breakpoint at JMP EAX, and press F9, CT will break point at JMP EAX.
Now, the F8 to trace the JMP, we will go to OEP.

QUOTE

```
004251D0 55 PUSH
EBP
004251D1 8BEC
MOV EBP, ESP
004251D3 83C4 F4
ADD ESP, 0C -
004251D6 E8
75DFFDFF CALL
Neolite2.00403150
004251DB E8
90F2FDFF CALL
Neolite2.00404470
004251E0 E8
6B1DFEFF CALL
Neolite2.00406F50
004251E5 E8
6286FEFF CALL
Neolite2.0040D84C
```

So OEP is 004251D0. Here, the plugins Ollydump, dump at this location
One file has been unpack.

Another method:

Go Pack program with Neolite 2.0 -> Neoworx Inc. [Overlay] I have a method to unpack again.
It is slightly more complicated than the way of Computer_Angel performed.
IPMonitor - Version 4.8 (Build 804) @ <http://www.ipmonitor.tsarfin.com>

- Load up the program with Olly. Select No (not Analysis). I will come:

QUOTE

```

0043C2BA> $ / E9
A6000000 JMP
IPMonito.0043C365
0043C2BF |
3ADE4300 DD
IPMonito.0043DE3A
0043C2C3. |
28C14300 DD <&
KERNEL32.
LoadLibraryA>
0043C2C7. |
24C14300 DD <&
KERNEL32.
GetProcAddress>

```

- Press F7 until now:

QUOTE

```

0043C861
$ 53 PUSH
EBX
0043C862.
51 PUSH
ECX
0043C863.
52 PUSH
EDX
0043C864.
56 PUSH
ESI
0043C865.
57 PUSH
EDI

```

- In the book Registers (FPU) we have to lick your mouse in the value ESP and Follow Indump. Hex dump window will switch to new window. Select the first 4 bytes of this window and lick

your mouse to select BreakPoint ==> Hardware, on access ==> Dword. Then press F9 to us:

QUOTE

```
0043C374. FE05
64C34300 INC
BYTE PTR DS:
[43C364] <== We
here
0043C37A. FFE0
JMP EAX <== JMP
to OEP (We can
add the will of
EAX register at
(FPU)
0043C37C. 803D
64C34300> Cmp
BYTE PTR DS:
[43C364], 0
0043C383. 75 13
JNZ SHORT
IPMonito.0043C398
```

- The last order F7 JMP EAX OEP is to:

QUOTE

```
0041AB11 6A 74
PUSH 74; <==
This is OEP @
dump!
0041AB13 68
98414200 PUSH
IPMonito.00424198
0041AB18 E8
F3010000 CALL
IPMonito.0041AD10
```

- Then select the next Plugins ==> Olly dump ==> dump debugged process. After you select

will be completed at the Entry Point in the box -> Modify is the value of the OEP (in this program is 1AB11). We save this value. Lick OK and save as you like (the extension is. Exe).

- Hold the Olly during the process. Use Import REConstructor v1.6F load © 2001-2003 program. Fill in the OEP OEP value is above 1AB11. Select IAT Auto Search, then select Get Imports and finally dump Fix (file with the fix is a new file is created above). In this program does not have any balance at all thunk.

- I will be new fle. File this run entirely possible. To improve our conduct over the cleaning and reduce the file size of the file after Fix dump (to file as small as possible. However, if you do not like you can ignore). To implement this process we use LordPE Delux v1.4. Load up the program, select rebuild PE. Select File Fix we dump on, lick the Open and we have a complete new file to run CRACK.

- Check the file with PEiD, we know the program is written in Micorsoft Visual C + + 7.0 Method2.

UnPaCkInG - PE Pack 1.0

Homepage: <http://crackmes.de>

Coder: gabrus666 (gabrus666 - vbcrackme1 - Level 1)

Cracked File: crackme.exe (Microsoft Visual Basic 5.0 / 6.0)

Packed: PE Pack 1.0 -> ANAKiN

Crack Tool: OllyDbg 1.10b, PEiD 0.92, ImportREC 1.6F

Unpack Type: Manual

Content PEiD we know is the program is in **PE Pack Pack 1.0 -> ANAKiN**

- Load up the program with Olly. Select No (not Analysis). I will come:

CODE

```
00401A41> $ / 74
00 JE SHORT
crackme.00401A43;
<== We here
00401A43> - \ E9
B8350000 JMP
crackme.00405000
```

- The F8 until we go to:

CODE

```
00405000 60
PUSHAD; <==
We here
00405001 E8
00000000 CALL
crackme.00405006
```

- Press F8 one command line to the next. In the book **Registers (FPU)** we have to lick your mouse in the value **ESP** and **Follow Indump. Hex dump** window will switch to new window. Select **the first 4 bytes** of this window and lick your mouse to select **BreakPoint ==> Hardware on access ==> Dword**. Then press **F9** to us:

CODE

```
00405270 - FFE0
JMP EAX;
crackme.004011A8
00405272 8D85
CE050000 LEA
EAX, DWORD
PTR SS: [EBP +5
CE]
00405278 50
PUSH EAX
```

- The next **F8** again we will go to OEP (right now the value of EAX is OEP)

CODE

```
004011A8 68 DB
68; char 'h'
004011A9
F4134000 DD
crackme.004013F4
004011AD E8 DB
E8
004011AE F0 DB
F0
004011AF DB FF
FF
```

- Note: This code is so because Olly is not ANALYZE. But not affect the process unpack all.

- Then select the next **Plugins ==> Olly dump ==> dump debugged process**. After you select will be completed at the Entry Point in the box -> Modify is the value of the OEP. Click OK and save as you like (the extension is. Exe).

- Hold the Olly during the process. Use Import REConstructor v1.6F load © 2001-2003 program. Fill in the OEP OEP value is above. Select IAT Auto Search, then select Get Imports and finally dump Fix (file with the fix is a new file is created above). In this program does not have any balance at all thunk.

- I will be a new file. File this run entirely possible. To improve our conduct over the cleaning

and reduce the file size of the file after Fix dump (to file as small as possible. However, if you do not like you can ignore).

To implement this process we use LordPE Delux v1.4. Load up the program, select rebuild PE. Select File Fix we dump on, lick the Open and we have a complete new file to run CRACK.

- Check the file with PEiD, we know the program is written in Microsoft Visual Basic 5.0 / 6.0.

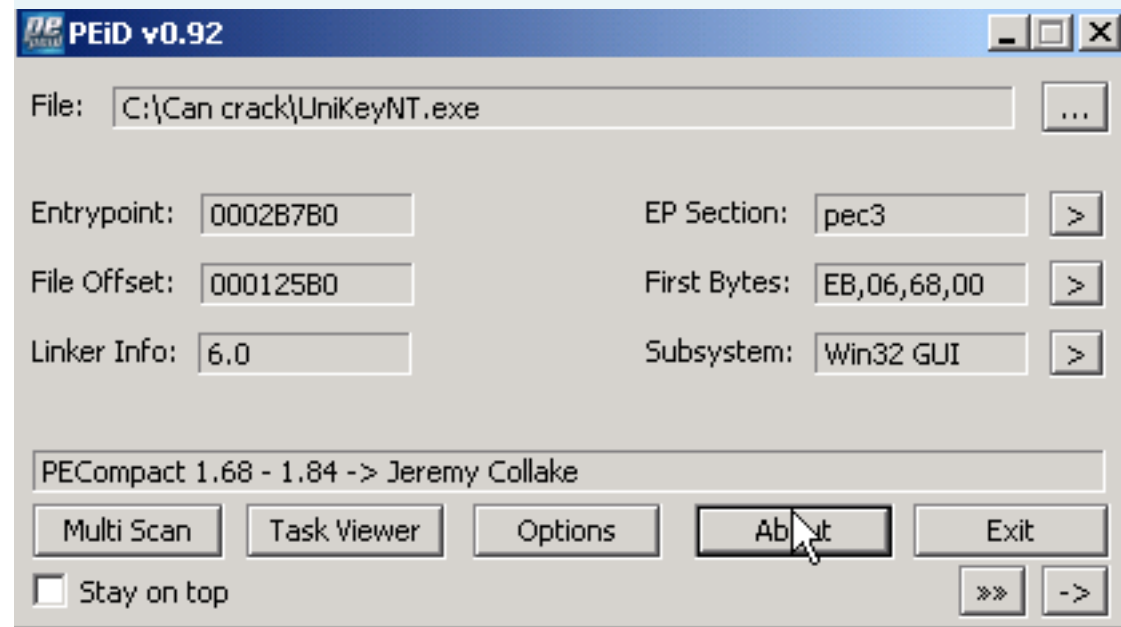
Learning unpack PECompact 1.68 - 1.84

Target: unpackme # 22

Packer: PECompact 1.68 - 1.84

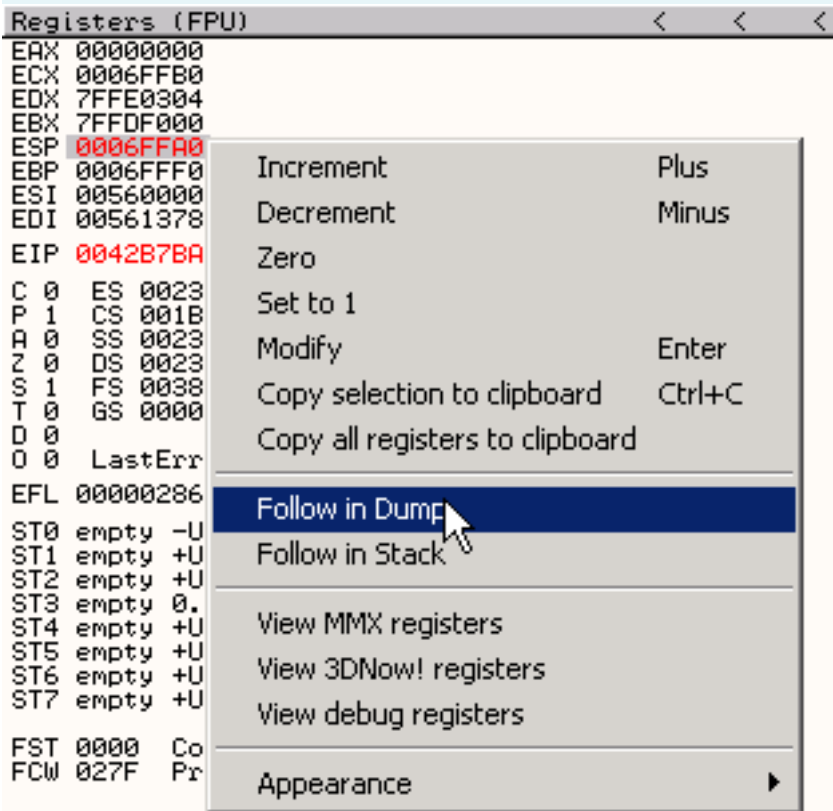
Download: <http://nhandan.info/hacnho>

First used to see PEID pack it with anything.

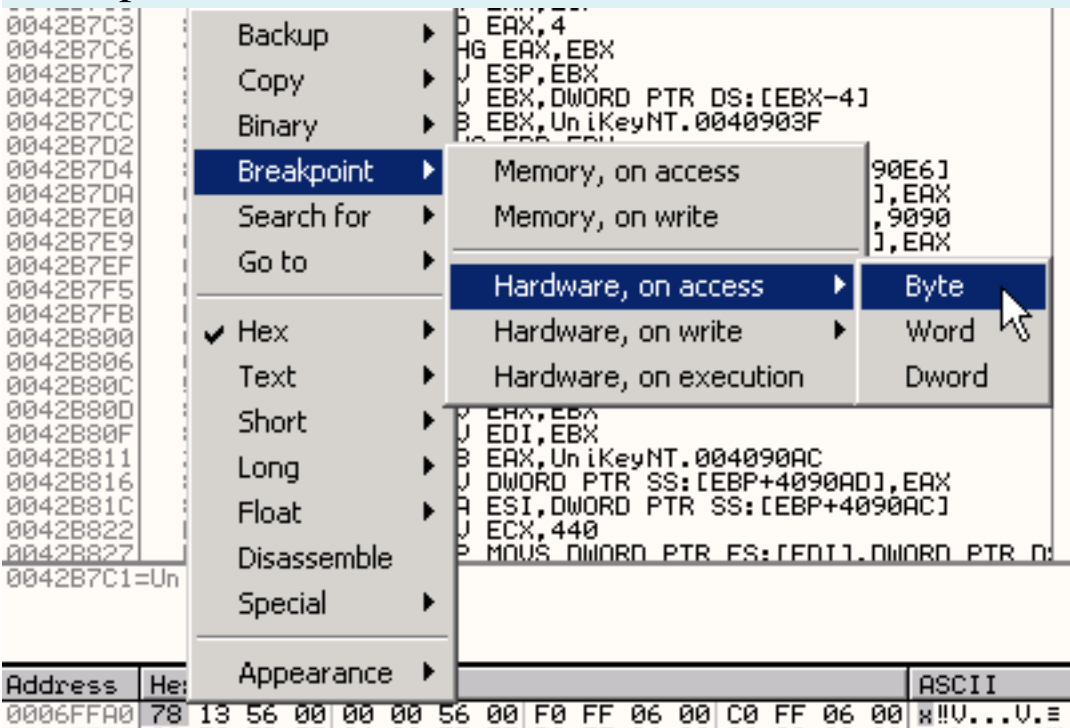


We see it in the pack PECompact 1.68 - 1.84

Using OllyDbg open the file. Search PUSHAD. We see it in 42B7B9. F8 to trace it through to 42B7BA. Thanh noted ESP change. Click your mouse button to select Follow in dump.



In the window below the left corner we are 6FFA0. Click to select the first byte as in the picture:



Press F9. The break in 42D54F. Press F8 to trace 42B002. Also located at the breakpoint as the ESP. Press F9. The break in 42B366. F8 3 times. We at OEP 40F1D0. Using OllyDump and Imprec to fix IAT.

Entry Point: 2B7B0 -> Modify: F1D0

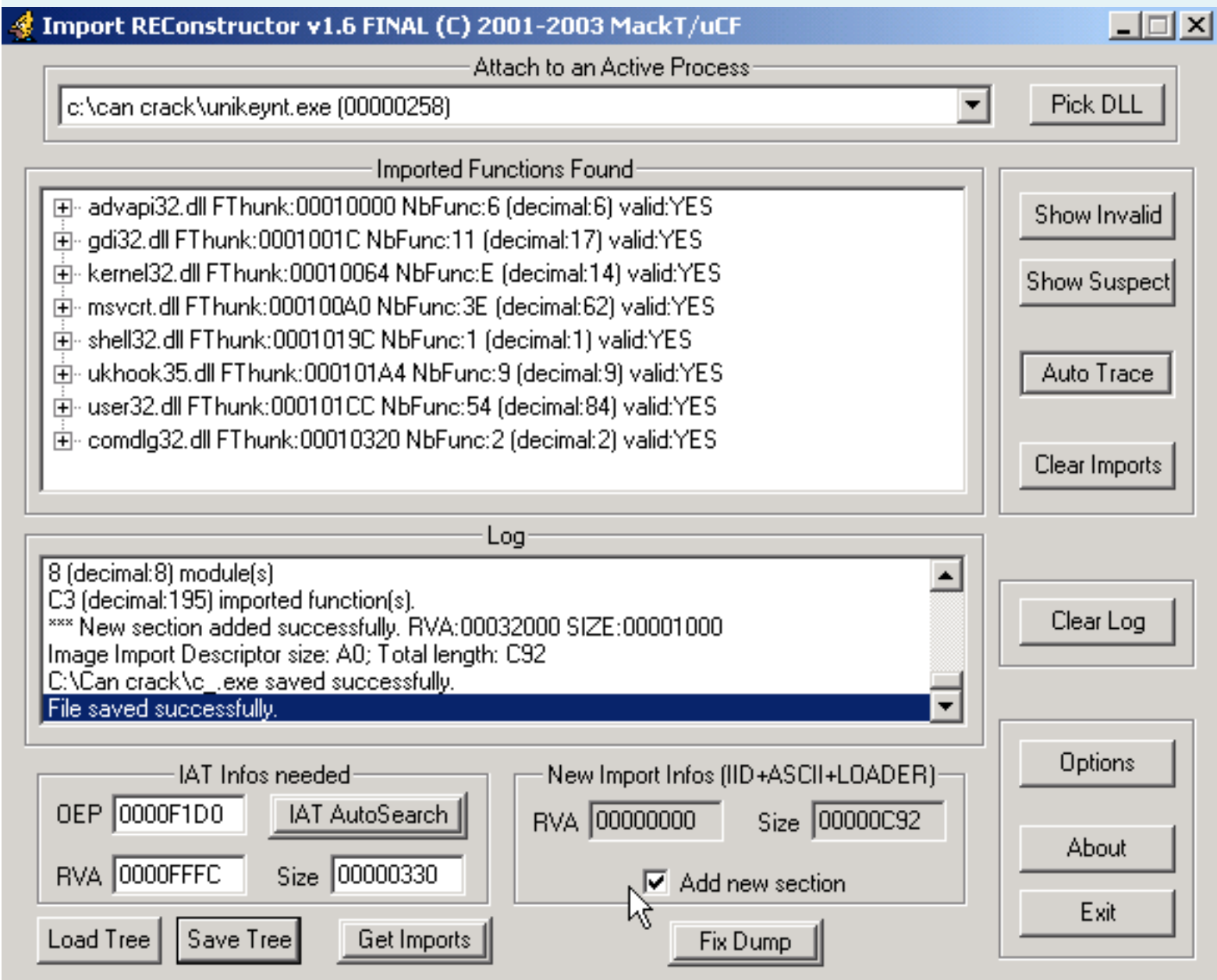
Base of Code: 1000 Base of Data: 10000

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
pec1	0001C000	00001000	0001C000	00001000	E0000020
.rsrc	00006000	0001D000	00006000	0001D000	C0000040
pec2	00008000	00023000	00008000	00023000	C2000040
pec3	00005000	0002B000	00005000	0002B000	C0000040
.rsrc	00001000	00030000	00001000	00030000	C0000040

☒ Rebuild Import

- ☒ Method1 : Search JMP[API] | CALL[API] in memory image
☐ Method2 : Search DLL & API name string in dumped file



Them. Happy happy.
tlandn

Homepage: <http://www.magictweak.com>

Production: Efreesky Software (Magic Utilities 2003 2.31)

Copyright by: Copyright © 2000-2003 Efreesky Software. All Rights Reserved.

Cracked File: mgutil.exe (Microsoft Visual C + + 6.0)

Type: Name / Serial

Packed: PECompact 1.68 - 1.84 -> Jeremy Collake

Crack Tool: OllyDbg 1.09d, PEiD 0.92, ImportREC, LordPE Delux v1.4

Unpack Type: Manual

- Load up the program with Olly. Select **No (not Analysis)**. I will come:

CODE

```
0056B000> / EB
06 JMP SHORT
mgutil.0056B008;
<== We here
0056B002 | 68
1C180500 PUSH
5181C
0056B007 | C3
RETN
0056B008 \ 9C
PUSHFD
0056B009 60
PUSHAD; <==
Use F8 to trace
here
```

- After coming here, visible window **Registers (FPU)** we have to lick your mouse in the value **ESP** and **Follow Indump. Hex dump** window will switch to new window. Select the first **4 bytes** of this window and lick your mouse to select **BreakPoint ==> Hardware, on access ==> Dword**. Then press **F9** to us:

CODE

```
0056C550 50
PUSH EAX; <==
We here
0056C551 68
1C184500 PUSH
mgutil.0045181C
0056C556 C2
0400 RETN 4
```

- File **F8** order to trace **RETN 4** and trace through this order, we go to:

CODE

```
0045181C 55
PUSH EBP;
<== We here:
OEP
0045181D
8BEC MOV
EBP, ESP
0045181F 6A
FF PUSH -1
```

- Then select the next **Plugins ==> Olly dump ==> dump debugged process**. After you select will be completed at the **Entry Point** in the box -> **Modify** is the value of the **OEP** (in this program is 5181C). We save this value. Lick OK and save as you like (the extension is. Exe).

- Keep raw Olly xuot in the process. Use **Import REConstructor v1.6F** load © **2001-2003** program. Fill in the OEP OEP value is above 5181C. Select **IAT Auto Search**, then select **Get Imports** and finally **dump Fix** (file with the fix is a new file is created above). In this program does not have any balance at all thunk.

- I will be new fle. File this run entirely possible. To improve our conduct over the cleaning and reduce the file size of the file after Fix dump (to file as small as possible. However, if you do not like you can ignore).

- To implement this process we use **LordPE Delux v1.4**. Load up the program, select **rebuild PE**. Select File Fix we dump on, lick the Open and we have a complete new file to run CRACK.

- Check the file with PEiD, we know the program is written in Microsoft Visual C + + 6.0.

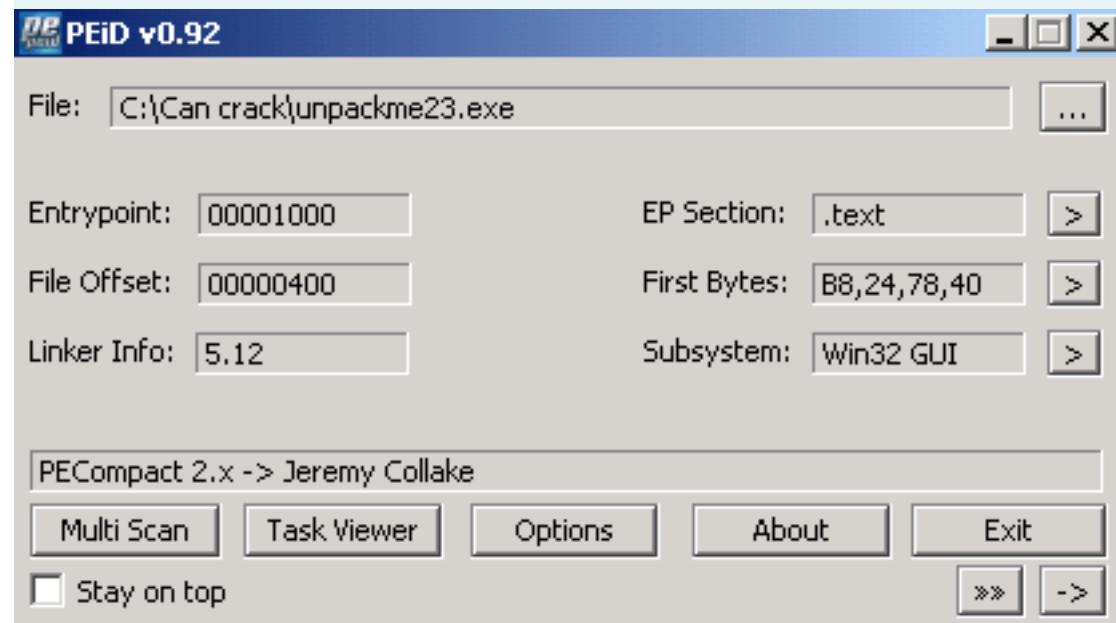
Learning unpack PECompact 2.x

Target: unpackme # 23

Packer: PECompact 2.x

Download: <http://nhandan.info/hacnho>

First used to see PEiD pack it with anything.



We see it in the pack PECompact 2.x

Using OllyDbg open the file. Press Shift-F9 program break in 401,016. Press Alt-M to open Memory map. Set breakpoint in section 2 (code).

00130000	00001000				Map	R	R	
00140000	00003000				Priv	Rw	Rw	
00240000	00006000				Priv	Rw	Rw	
00250000	00001000				Map	Rw	Rw	
00260000	00016000				Map	R	R	
00280000	00034000				Map	R	R	
002C0000	00041000				Map	R	R	
00310000	00006000				Map	R	R	
00400000	00001000	unpackme		PE header	Image	R	RwE	
00401000	00005000	unpackme	.text	code				
00406000	00002000	unpackme	.rsrc	imports, res				
77E60000	00001000	kernel32		PE header				
77E61000	00075000	kernel32	.text	code, import				Actualize
77ED6000	00003000	kernel32	.data	data				View in Disassembler Enter
77ED9000	00066000	kernel32	.rsrc	resources				Dump in CPU
77F3F000	00006000	kernel32	.reloc	relocations				Dump
77F50000	00001000	ntdll		PE header				Search Ctrl+B
77F51000	0006F000	ntdll	.text	code, export				
77FC0000	00005000	ntdll	ECODE	code				
77FC5000	00005000	ntdll	.data	data				
77FCA000	0002C000	ntdll	.rsrc	resources				
77FF6000	00003000	ntdll	.reloc	relocations				Set break-on-access F2
7F6F0000	00007000							
7FFB0000	00024000							
7FFDE000	00001000			data block				Set memory breakpoint on access
7FFDF000	00001000							Set memory breakpoint on write
7FFE0000	00001000							Set access ▶
								Copy to clipboard ▶
								Sort by ▶
								Appearance ▶

Close Memory map. Press Shift-F9. The break in 407,839.

00407839	C602 E9	MOV BYTE PTR DS:[EDX],0E9	
0040783C	83C2 05	ADD EDX,5	
0040783F	2BCA	SUB ECX,EDX	
00407841	894A FC	MOV DWORD PTR DS:[EDX-4],ECX	
00407844	33C0	XOR EAX,EAX	
00407846	C3	RETN	
00407847	B8 CE6640F0	MOV EAX,F04066CE	

Press F8 6 times. The program will be in here 77xxxxxx

77F833A0	64:8B25 00000000	MOV ESP,DWORD PTR FS:[0]	
77F833A7	64:8F05 00000000	POP DWORD PTR FS:[0]	
77F833AE	8BE5	MOV ESP,EBP	
77F833B0	5D	POP EBP	
77F833B1	C2 1400	RETN 14	
77F833B4	8B4C24 04	MOV ECX,DWORD PTR SS:[ESP+4]	
77F833B8	F741 04 06000000	TEST DWORD PTR DS:[ECX+4],6	
77F833BF	B8 01000000	MOV EAX,1	

Press Alt-F9. We at 401,016. This is not our OEP.

00401000	B8 24784000	MOV EAX,unpackme.00407824	
00401005	50	PUSH EAX	
00401006	64:FF35 00000000	PUSH DWORD PTR FS:[0]	
0040100D	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00401014	33C0	XOR EAX,EAX	
00401016	- E9 2C680000	JMP unpackme.00407847	
0040101B	6F	OUTS DX,DWORD PTR ES:[EDI]	I/O command
0040101C	6D	INS DWORD PTR ES:[EDI],DX	I/O command
0040101D	70 61	J0 SHORT unpackme.00401080	
0040101F	637432 00	ARPL WORD PTR DS:[EDX+ESI],SI	

Press Alt-M. Remove breakpoint.

0012E000	00002000		stack of ma	Priv	Rw	Gua	Rw
00130000	00001000			Map	R		R
00140000	00003000			Priv	Rw		Rw
00240000	00006000			Priv	Rw		Rw
00250000	00001000			Map	Rw		Rw
00260000	00016000			Map	R		R
00280000	00034000			Map	R		R
002C0000	00041000			Map	R		R
00310000	00006000			Map	R		R
00400000	00001000	unpackme	PE header	Imag	R		RwE
00401000	00005000	unpackme	code	Imag	R		RwE
00406000	00002000	unpackme	imports, re				
77E60000	00001000	kernel32	PE header				
77E61000	00075000	kernel32	code, impor				
77ED6000	00003000	kernel32	data				
77ED9000	00066000	kernel32	resources				
77F3F000	00006000	kernel32	relocation				
77F50000	00001000	ntdll	PE header				
77F51000	0006F000	ntdll	code, expor				
77FC0000	00005000	ntdll	code				
77FC5000	00005000	ntdll	data				
77FCA000	0002C000	ntdll	resources				
77FF6000	00003000	ntdll	relocation				
7F6F0000	00007000						
7FFB0000	00024000						
7FFDE000	00001000		data block				
7FFDF000	00001000						
7FFE0000	00001000						

Actualize	
View in Disassembler	Enter
Dump in CPU	
Dump	
Search	Ctrl+B
Set break-on-access	F2
Set memory breakpoint on access	
Set memory breakpoint on write	
Remove memory breakpoint	
Set access	

Close Memory map. Continue with F8 to trace 4078EF (JMP EAX).

004078DA	5A	POP EDX	
004078DB	03CA	ADD ECX,EDX	
004078DD	68 00800000	PUSH 8000	
004078E2	6A 00	PUSH 0	
004078E4	57	PUSH EDI	
004078E5	FF11	CALL DWORD PTR DS:[ECX]	
004078E7	8BC6	MOV EAX,ESI	
004078E9	5A	POP EDX	
004078EA	5E	POP ESI	
004078EB	5F	POP EDI	
004078EC	59	POP ECX	
004078ED	5B	POP EBX	
004078EE	5D	POP EBP	
004078EF	- FFE0	JMP EAX	unpac
004078F1	0010	ADD BYTE PTR DS:[EAX],DL	

F8. We OEP at 401,000. Using Ollydump.

Entry Point: -> Modify:

Base of Code: Base of Data:

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
.text	00005000	00001000	00005000	00001000	E0000020
.rsrc	00002000	00006000	00002000	00006000	E0000020

☒ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Them. Happy happy.
tlandn

Information
Unpacking for
Newbies
Target: 1.76
PeCompact
unpack-me
Tools:
OllyDbg 1:09
Protection:
packed /
Decompressed
on the fly
Beginner
Level
Category
Unpacking

Download: <http://kah.serveftp.net/upload/UnpackMes/Unpackme PeCompact 1.76. Exe>

PECompact is the so nen or file for running Windows. The program will initiate nen file implementers or modules then make unpacking (unpack) on memory at the time of initiating real thi.PECompact nen code, data, Import Directory, resources and other selected portions of Windows portable executables . When running, the file will be running on mem rebuilt without delays a nao.Okai .. Now initiate UnpackMe unpack files ... File this is code with C + + and the pack with 1.76 PeCompact Ops before .. you need to go a set number of options for Olly: Option Menu -> debugging option -> Trace (tab) ->> Always check trace over system DLLs Click OK.

I. Find OEP:

Create Trace Condition (CTRL + T), check option and enter POPAD Command.

This code sometimes converse with PUSHAD you see in the code where the program.

Initiate Trace (CTRL + F11) for a little trace several times and you'll see POPAD, POPFD ->>

Here was nearly OEP then ...

CODE


```
0040654E 61 POPAD
0040654F POPFD 9D
00406550 50 PUSH
EAX
00406551 68
60154000 PUSH
Unpackme.00401560
00406556 C2 0400
RETN 4
```

->> Trace through to RETN to OEP

CODE

```
00,401,560th 55
PUSH EBP <- OEP
00401561st 8BEC
MOV EBP, ESP
00,401,563th 6A FF
PUSH -1
00401565th 68
C8234000 PUSH
Unpackme.004023C8
0040156A. 68
E6164000 PUSH
Unpackme.004016E6;
JMP to MSVCRT.
_except_handler3;
SE handler
installation
```

As complete the heart OEP.

II. Dump and complete unpack:

Plugins -> OllyDump -> dump Debugged process. (The easiest way: D)

Check rebuild Imports

Use more LordPE's PE rebuild to complete jobs and improve the size.

```
+-----+
| Solution: UnpackMe # 1_by_KLiZMA |
| Author: kienmanowar |
| Protection: Unknown packer (like Upx) |
| Language: Borland Delphi |
| Date: 05/13/06 |
| Great thanx for iamidiot to give me your hint |
+-----+
```

Tools: Ollydbg, PEid v0.94, RDG Packer Detector v0.6.3 Beta, ImpRec v1.6

[Manual Unpacking]

Try to detect this Unpackme with PEid and RDG, I get some information:

PEiD:

+ Normal Scan: Nothing found *

Scan + Hardcore: UPolyX v0.5 *

RDG:

+ Normal Scan: UG2002 Cruncher v0.3b3

+ Advanced Scan: UPX v0.8x (UPX Scrambler Heuristico)

With these information, I do not know what exactly packer in which this UnpackMe used. So I use PEid plugin (Generic OEP Finder) to find OEP of UnpackMe, it gives me: 00463C80. Okie:) May be this is the right OEP!

Close and open PeiD Ollydbg to load this in UnpackMe. A messagebox appears, choose it. I have:

```
0047E000> 60 pushad <== Stop here (EP)
0047E001 E8 00000000 call unpackme.0047E006
0047E006 5D pop ebp
0047E007 81ED 48124000 sub ebp, unpackme.00401248
0047E00D 60 pushad
0047E00E E8 2B030000 call unpackme.0047E33E
0047E013 61 popad
0047E014 8A7D 60 mov bh, byte ptr ss: [ebp +60]
0047E017 6262 DB bound esp, qword ptr ds: [edx-25]
0047E01A 6A 62 push 62
6262 EF 0047E01C bound esp, qword ptr ds: [edx-11]
0047E01F D7 xlat byte ptr ds: [ebx + al]
0047E020 EA 7022628A 496> jmp far 6049:8 A622270
0047E027 6262 9D bound esp, qword ptr ds: [edx-63]
0047E02A F7?; Unknown command
22 Lea 0047E02B 8D70 esi, dword ptr ds: [eax +22]
0047E02E 62E9 bound ebp, ecx; Illegal use of register
0047E030 THREE F2F29DF7 mov edx, F79DF2F2
```

Oh! I see Pushad signature, like UPX. Press Alt + M to open Memory map window.

Memory map

Address Size (Owner Section Contains Type Access Initial Mapped as

```
.....
00370000 00003000 (0 Map R R
00400000 00001000 (unpackme 0 PE header IMAG R RWE
00401000 0004B000 (unpackme 0. KLiZMA IMAG R RWE
0044C000 00031000 (unpackme 0. KLiZMA code IMAG R RWE
0047D000 00001000 (unpackme 0. Rsrc data, imports IMAG R RWE
0047E000 00001000 (unpackme 0. KLiZMA SFX R RWE IMAG
00480000 00004000 (0 Map RE RE
00540000 00002000 (0 Map RE RE
00550000 00103000 (0 Map R R
00660000 0006A000 (0 Map RE RE
.....
```

In this window, select section:

```
00401000 0004B000 (unpackme 0. KLiZMA IMAG R RWE
```

And Right click and set a Memory Breakpoint on Access.And then Press F9 to Run, Olly breaks here:

```
0047CCA3 8807 mov byte ptr ds: [edi], al <== Stop here after Press F9 (1st)
```

```
0047CCA5 47 inc edi
```

```
0047CCA6 01DB add ebx, ebx
```

```
0047CCA8 75 07 jnz short unpackme.0047CCB1
```

```
0047CCAA 8B1E mov ebx, dword ptr ds: [esi]
```

```
0047CCAC 83EE FC sub esi, -4
```

```
ADC 0047CCAF 11DB ebx, ebx
```

```
72 ED 0047CCB1 ^ jb short unpackme.0047CCA0
```

Come back Memory Map Window and clear Memory BP.And then back to the CPU window, scroll down to find the signature:):

```
0047CDEB 83C3 04 add ebx, 4
```

```
EB 0047CDEE ^ E1 jmp short unpackme.0047CDD1
```

```
0047CDF0 FF96 near 84CE0700 call dword ptr ds: [esi +7 CE84]
```

```
0047CDF6 61 popad <=== Aha Popad
```

```
0047CDF7 ^ E9 846EFEFF jmp unpackme.00463C80 <=== Jump to OEP (OEP Like found in Peid)
```

As you see, this signature like UPX. And now, BP set at 0047CDF7 ^ E9 846EFEFF

jmp unpackme.00463C80

Press F9 to Run, Break at this BP, and remove this Press F8, kaka we stop at OEP of UnpackMe:

```
00463C80 55 push ebp <=== Right OEP
```

```
00463C81 8BEC mov ebp, esp
```

```
00463C83 83C4 F0 add esp, -10
```

```
00463C86 b8 903A4600 mov eax, unpackme.00463A90
```

```
00463C8B E8 A41FFAFF call unpackme.00405C34
```

```
00463C90 A1 F8584600 mov eax, dword ptr ds: [4658F8]
```

```
00463C95 8B00 mov eax, dword ptr ds: [eax]
```

```
00463C97 E8 14B2FEFF call unpackme.0044EEB0
```

Now, with dump Ollydump Plugin, not check Import Rebuilt. Press dump and save as: dumped.exe. Fire up ImportRec, select Process, write OEP, IAT Press Auto Search, Get Imports and finally Fix Dump.We have dumped_.exe. Test it: kaka it runs before I double-click:) lol detect and again by PEid: Borland Delphi 6.0 - 7.0.

[Cracking]

MUP UnpackMe After this, come to part 2 to change "unregistered" to "registered". To do this task, dumped_.exe Load into Ollydbg. Press F9 to run it, we'll see a beautiful girl with "unregistered" string below. Back to Ollydbg, Press Alt + M to open Windows Memory map, here we select:

Memory map, item 0

Address = 00010000

Size = 00001000 (4096.)

Owner = 00010000 (itself)

Section =

Type = 00021004 Priv

Access = RW

Initial access = RW

Right-click and select Search (or Ctrl + B), then type: unregistered in Ascii textbox and Press OK to Search this string.After that Olly break at:

Memory map, item 25

Address = 0047D000

Size = 00001000 (4096.)

Owner = 00400000 dumped_

Section =. rsrc

= Contains data, resources

Type = 01001002 IMAG

Manual unpack UPX 0.89.6 - 1.02 / 1.05 - 1.24

I - Find OEP:

- Use PEiD we know is the Pack with the UPX 0.89.6 - 1.02 / 1.05 - 1.24 (Delphi) stub -> Markus & Laszlo.
- Load up the program with Olly. Select No (not Analysis). I will come:

CODE

```
004A86C0> 60
PUSHAD <=== We
here.
004A86C1 BE
00E04600 MOV
ESI,
regedito.0046E000
004A86C6 8DBE
0030F9FF LEA
EDI, DWORD PTR
DS: [ESI
FFF93000 +]
```

- Pull the screen down the encounter until the command line:

CODE

```
004A8847 61 POPAD
004A8848 ^ E9
0322FDFF JMP
regedito.0047AA50
<=== Set
BreakPoint here
004A884D 0000
ADD BYTE PTR DS:
[EAX], AL
004A884F 0068 88
ADD BYTE PTR DS:
[EAX-78], CH
```

- After setting BreakPoint we press F9, the program will stop here. F7 to our staff:

CODE

```
0047AA50 55 PUSH EBP <== We here
0047AA51 8BEC MOV EBP, ESP
0047AA53 83C4 F4 ADD ESP, 0C -
```

- Right here we choose Plugins ==> OllyDump ==> dump debugged process. Write value at -> Modify. Human OK and save your program in any name what you like. With the extension is. Exe.

II - Find and RVA RVA size:

- You will find two ways: (1) IAT Auto Search, and (2) Manual
- You should find using IAT Auto Search for more quickly.
- However, in some cases IAT Auto Search can not find out RVA RVA and size so we must be heart by Manual.

II.1 - Content IAT Auto Search:

- Stay in the program, the program ImportREC v1.6 load the file, change the value OEP we just find the above.
- The IAT Auto Search
- Trade should we change the last two of RVA to RVA Size 00 and we should increase slightly
- Then Get the Imports, the next Show Invalid, then the Trace level 1 (disasm).
- Finally Fix the dump. Select the file that you get saved above.
- Fix then dump (file with the fix is a new file is created above).
- I will be a new file. If nothing special examination FILE has been successfully unpack.

II.2 - Manual:

- Stay in main screen. Press Ctrl-B screen in the CPU, and go to the FF 25, then press Ctrl-L continue to find code to the like this:

CODE

```
00497284 -  
FF25  
34214B00  
JMP DWORD  
PTR DS:  
[4B2134];  
advapi32.  
GetUserNameA
```

- Press Ctrl-L to find out the code. Similarly, in the heart, note the largest value max and min in most small DS: [4B2134].
- So we have:
OEP = value we find the above
RVA = min - 400,000
Size = max - min ==> usually increases slightly
- Use Import REConstructor v1.6F © 2001-2003 load file.
- Complete the value calculation above the IAT infos needed. Then Get the Imports, the next Show Invalid, then the Trace level 1 (disasm).
- After the trace is complete, the next Show Invalid, then the next Show Invalid, and the Cut thunks.
- Fix then dump (file with the fix is a new file is created above).
- I will be a new file. If nothing special examination FILE has been successfully unpack.

III - How clean and reduce the file size after unpack:

- To improve our conduct over the cleaning and reduce the file size of the file after Fix dump (to file as small as possible. However, if you do not like you can ignore).
- To implement this process we use LordPE Delux v1.4. Load up the program, select rebuild PE. Select File Fix we dump on, lick the Open and we have a complete new file to run CRACK.
- Check the file with PEiD, we know the language of the program.

IV - Note:

- The instructions on here to correct most cases encountered in the SOFTWARE or CRACKME.
- Some special cases we need to initiate a slightly different take slightly. However the basic steps do not have the difference.

- Address in the instructions may vary, but you should note to order me. The question this order can not be different.

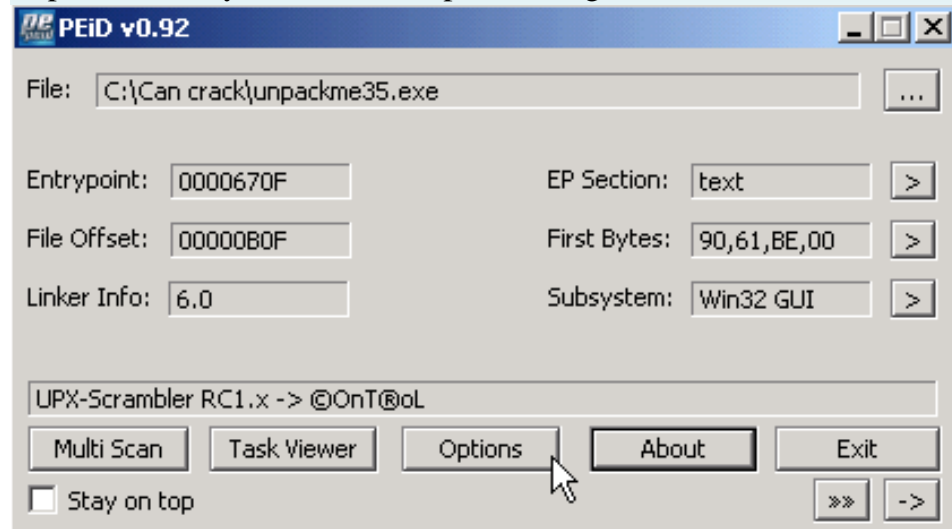
Learning unpack UPX Scramble RC 1st x

Target: unpackme # 35

Packer: RC 1.x UPX Scramble

Download: <http://nhandan.info/hacnho>

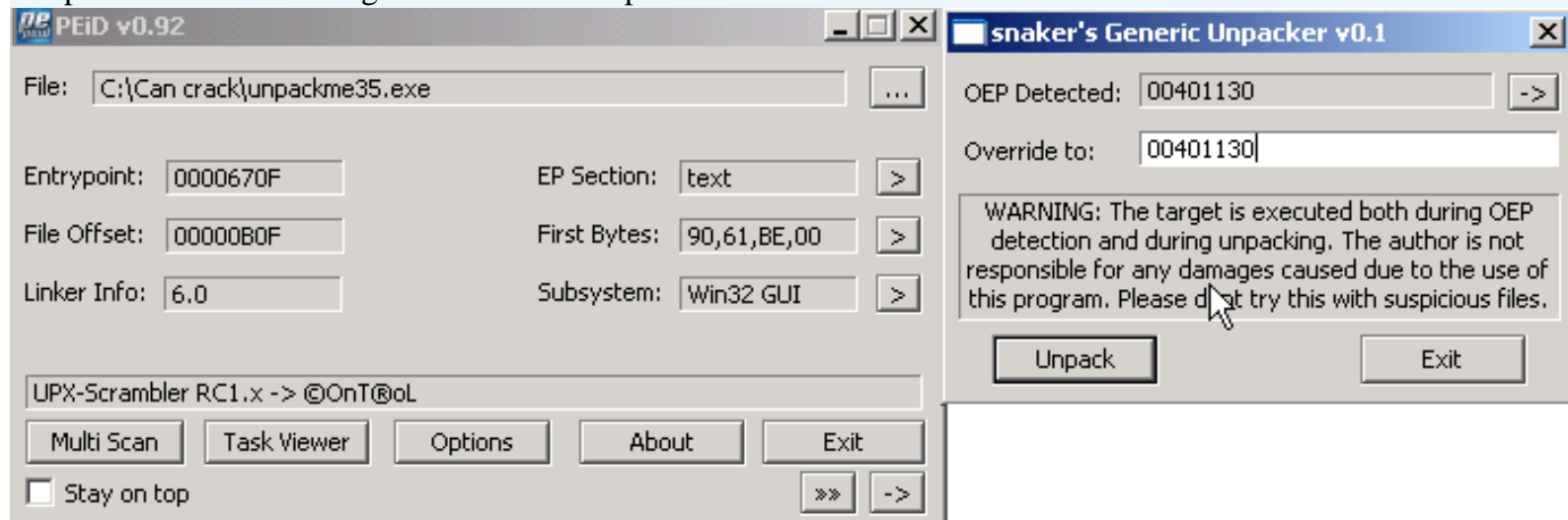
Top PEID handy it is to see the pack with ge.



We feel it is pack with UPX Scramble RC 1.x.

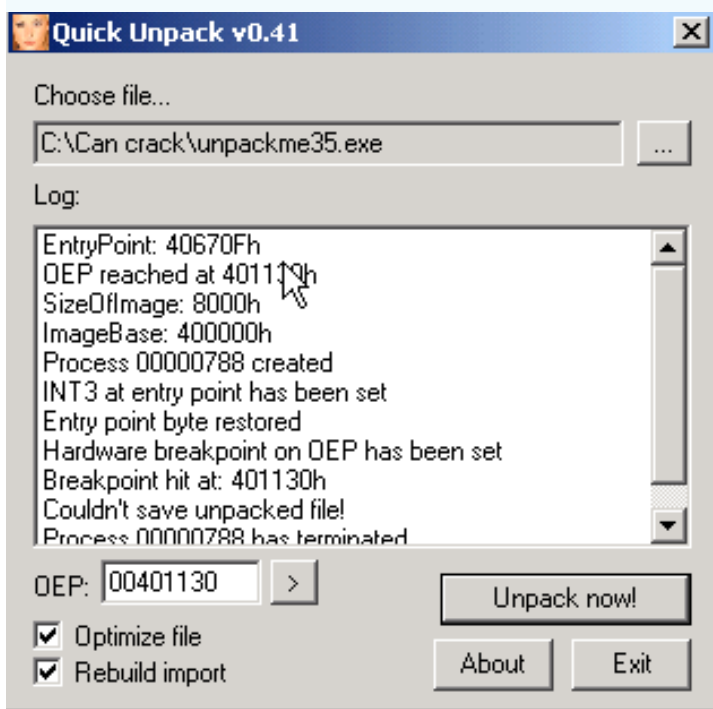
Method 1:

Simple and effective. Dung PEID Generic Unpacker.



Method 2:

Dung Quick unpack 0:41.

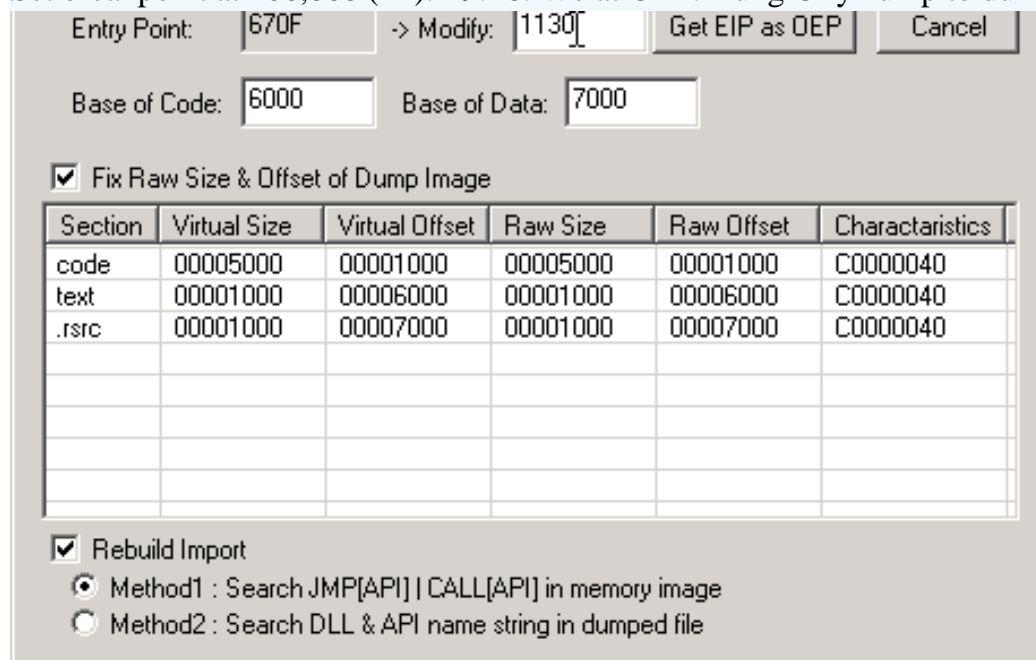


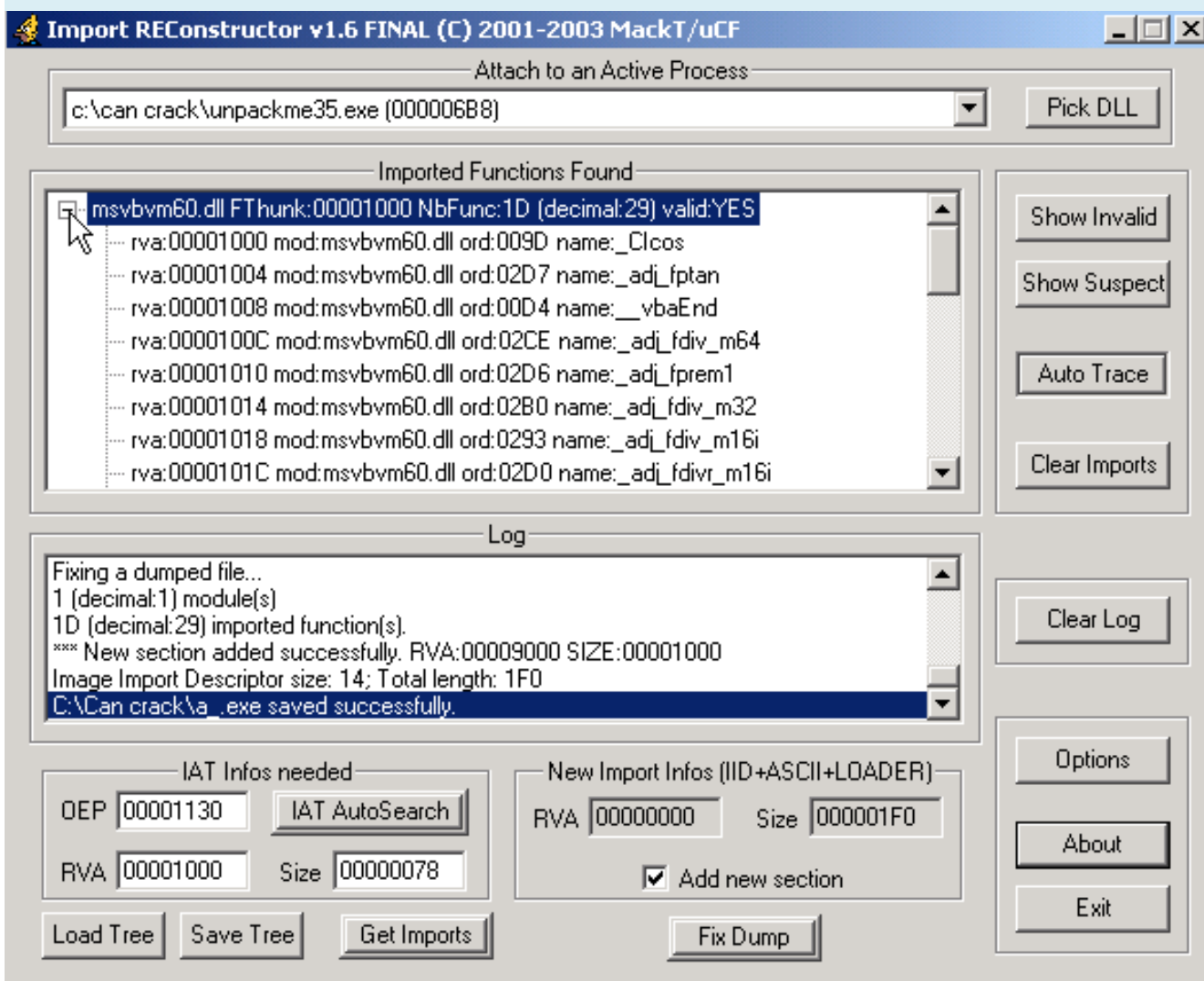
3 months:

Dung OllyDbg open the file. Pull down most file (Featured by a series of byte 0).

00406850	FF96 28690000	CALL DWORD PTR DS:[ESI+6928]
00406856	09C0	OR EAX,EAX
00406858	74 07	JE SHORT unpackme.00406861
0040685A	8903	MOV DWORD PTR DS:[EBX],EAX
0040685C	83C3 04	ADD EBX,4
0040685F	EB D8	JMP SHORT unpackme.00406839
00406861	FF96 2C690000	CALL DWORD PTR DS:[ESI+692C]
00406867	60	PUSHAD
00406868	E9 C3A8FFFF	JMP unpackme.00401130
0040686D	0000	ADD BYTE PTR DS:[EAX],AL
0040686F	0000	ADD BYTE PTR DS:[EAX],AL
00406871	0000	ADD BYTE PTR DS:[EAX],AL
00406873	0000	ADD BYTE PTR DS:[EAX],AL
00406875	0000	ADD BYTE PTR DS:[EAX],AL
00406877	0000	ADD BYTE PTR DS:[EAX],AL

Set breakpoint at 406,868 (F2). F9.F8. We at OEP. Dung OllyDump to dump file and Imprec to fix IAT.





So lr completed. Happy happy.

tländn

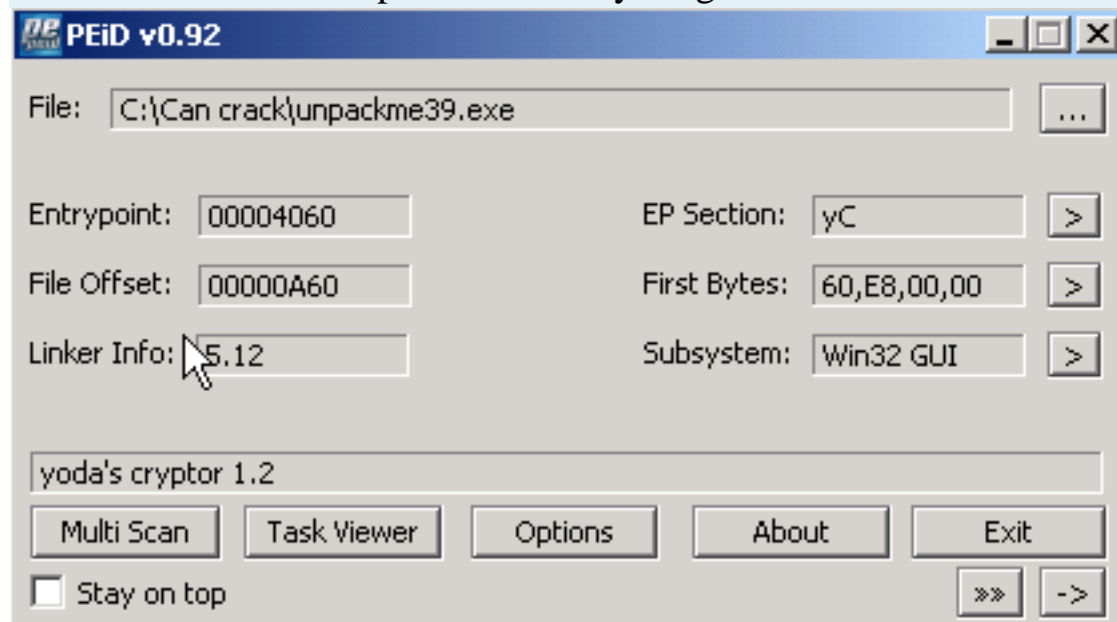
Learning unpack Yoda Cryptor 1.2

Target: unpackme # 39

Packer: Yoda Cryptor 1.2

Download: <http://nhandan.info/hacnho>

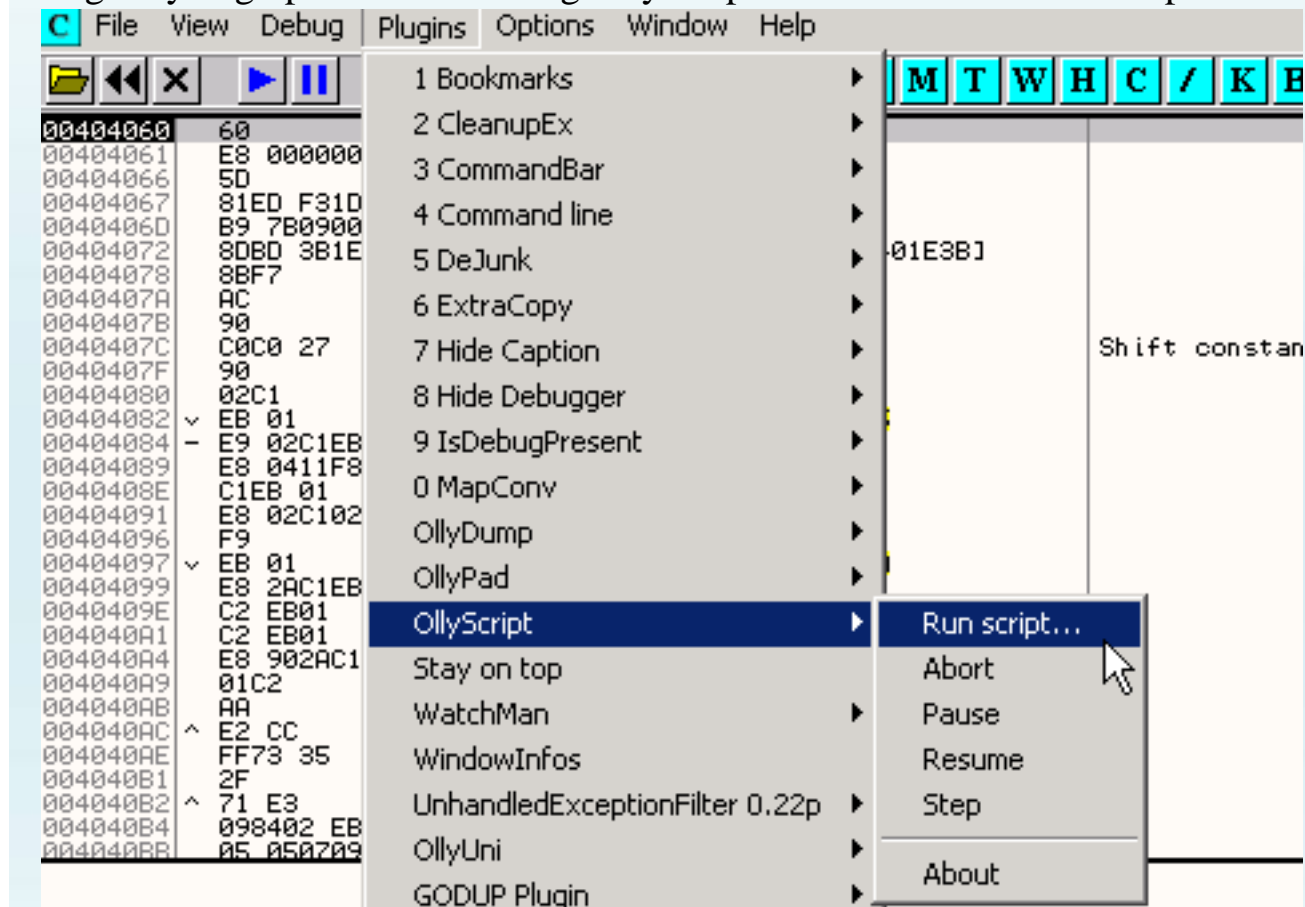
First used to see PEID pack it with anything.



We see it in the pack Yoda Cryptor 1.2.

Method 1:

Using OllyDbg open the file. Using OllyScript. I have enclosed in the zip file



Select File y0da_crypter_1.2.txt

The program will stop at OEP 401,000. Using OllyDump to file dump

Entry Point: 4060 -> Modify: 1000 Get EIP as OEP Cancel

Base of Code: 1000 Base of Data: 2000

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
.text	00000026	00001000	00000026	00001000	E0000020
.rdata	00000092	00002000	00000092	00002000	C0000040
.data	00000096	00003000	00000096	00003000	C0000040
yC	00002000	00004000	00002000	00004000	E00000E0

☒ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Method 2:

Using OllyDbg open the file. Search PUSHAD. May too, we are standing there. Press F8. Thanh noted ESP will change. Click on the button to select Follow ESP Print dump

Registers (FPU)

EAX	00000000
ECX	0012FFB0
EDX	7FFE0304
EBX	7FFDF000
ESP	0012FFA4
EBP	0012FFF0
ESI	00560000
EDI	00561378
EIP	00404061
C 0	ES 0023
P 1	CS 001B
A 0	SS 0023
Z 0	DS 0023
S 1	FS 0038
T 0	GS 0000
D 0	
O 0	LastErr
EFL	00000286
ST0	empty +U
ST1	empty +U
ST2	empty +U
ST3	empty 0.
ST4	empty +U
ST5	empty +U
ST6	empty +U
ST7	empty +U
FST	0000 Co
FCW	027F Pr

Increment Plus

Decrement Minus

Zero

Set to 1

Modify Enter

Copy selection to clipboard Ctrl+C

Copy all registers to clipboard

Follow in Dump

Follow in Stack

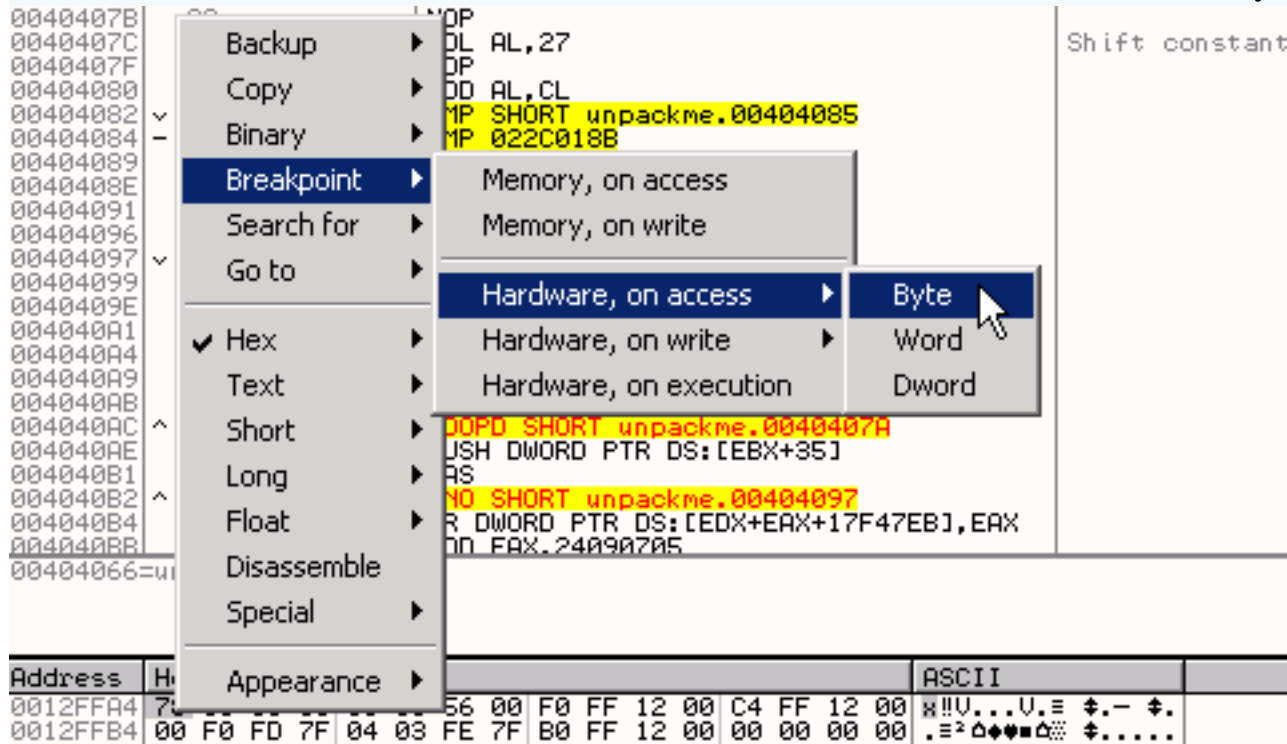
View MMX registers

View 3DNow! registers

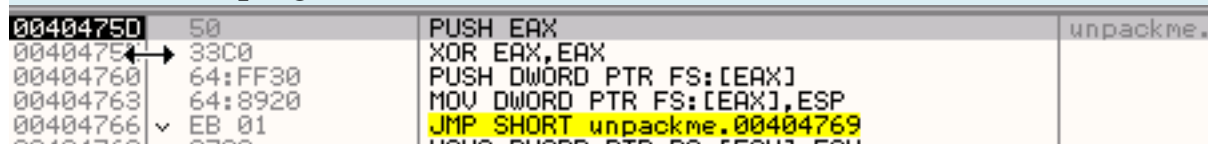
View debug registers

Appearance

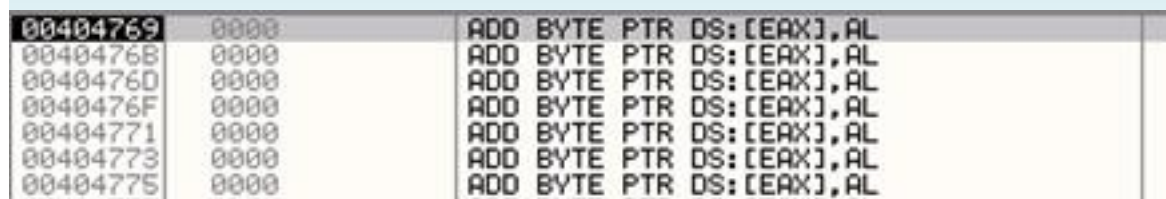
In the window below the left corner we are 12FFA4. Click to select the first byte as in the picture



Press F9. The program will break at 40475D.



Press F8 6 times.



Press Shift-F9. We will be at 401,002. But this is not OEP. OEP located on a line that is at 401,000. We use OllyDump.

Entry Point: -> Modify:

Base of Code: Base of Data:

☒ Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
.text	00000026	00001000	00000026	00001000	E0000020
.rdata	00000092	00002000	00000092	00002000	C0000040
.data	00000096	00003000	00000096	00003000	C0000040
yC	00002000	00004000	00002000	00004000	E00000E0

☒ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Them. Happy happy.

tländn

*** Lam nhảm bomb:

-From the day that the homepage of ARTeam, I have a lot of down tut English, but much time do not follow (in English that dot).

-For the Newbie dot English bags a little more today, I would post a few lines based on 2 tut by gã [MaDMAn_H3rCuL3s](#) (gã I hate this). This is not translate the level I is not sufficient to all understand, the only, the tut-aged English I find this place has not the same as when you make enough, do not know where to stagger a few points when I stopped doing that any shortfalls are a few lines so lao.Do with that image in this article I have to capture to the Newbie bags to track more.

Then, throw out long bomb, the time to issue and that (for 2 to buy Bom).

[Section 1: Unpacking Armadillo v3.x anti-dump](#) (first tut)

Target	GPS Express v3.2
Available	http://intechhosting.com/~ access/ARTeam/tools/GPSE3.exe
Tools	OllyDbg 1:10, ImpRec, LordPE, Olly Plugin: Hide Debugger , CMDBar

Note-you 2 plugin contributed by Olly quite important in this series tut.

-As for the current pack is often the (compressed) to combat some parents Cracker as we, and this is a style pack is quite popular, used Armadillo.

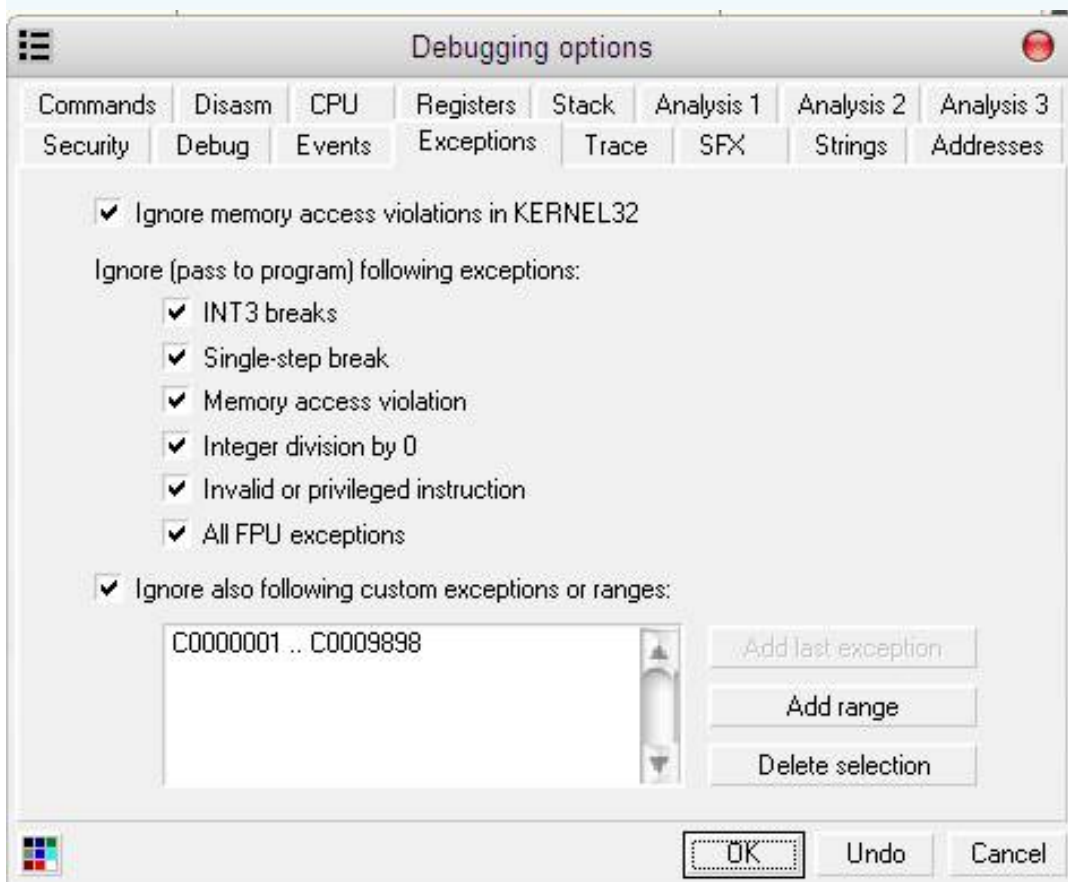
-Our mission is to find our real OEP program, dump it, then fix IAT.

-As the author said, is the message for the Newbie simple steps, short understandable. The explained further, please wait for the reference of his tut [hacnho](#) on small Armadillo.

I. Search and dumping OEP:

First-2 is a copy of Olly plugin to plugin directory of Olly nhé.

-Then, we have to edit like this in Olly (usually when unpack Armadillo). In the Options-debugging options Exception-selected card, then check the following:



-We check on the process to debug soon will be, at least fails.

-If you have the plugin Cmdbar.dll after Olly, will see this below:

Command :

Time-load thẳng GSPEXpress.exe vào. Sau đó click vào CmdBar nhé:

Command :

-Enter one, it will set a break-Point elsewhere. (Want to know where you click "B" in Olly, double-click the line that will take us to where the break-Point).

Time-Shift-click the F9 to run it a bit, break it in just put BP (BP should remove go here). Press Ctrl-F9 continue to RENT in the order.

77E7BE53	55	PUSH EBP
77E7BE54	8BEC	MOV EBP,ESP
77E7BE56	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
77E7BE59	FF75 18	PUSH DWORD PTR SS:[EBP+18]
77E7BE5C	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77E7BE5F	FF75 10	PUSH DWORD PTR SS:[EBP+10]
77E7BE62	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]
77E7BE65	FF75 08	PUSH DWORD PTR SS:[EBP+08]
77E7BE68	6A FF	PUSH -1
77E7BE6A	E8 30FEFFFF	CALL kernel32.CreateRemoteThread
77E7BE6F	5D	POP EBP
77E7BE70	C2 1800	RETN 18

F7-one to come:

00C2F8B2	5E	POP ESI
00C2F8B3	C9	LEAVE
00C2F8B4	C3	RETN

Lai-Ctrl-F9 to here:

00C2F8B2	5E	POP ESI
00C2F8B3	C9	LEAVE
00C2F8B4	C3	RETN

F7-the will to here:

00C49343	A1 D8E6C500	MOV EAX,DWORD PTR DS:[C5E6D8]	
00C49348	59	POP ECX	
00C49349	8B88 80000000	MOV ECX,DWORD PTR DS:[EAX+80]	
00C4934F	3348 3C	XOR ECX,DWORD PTR DS:[EAX+3C]	
00C49352	3348 30	XOR ECX,DWORD PTR DS:[EAX+30]	
00C49355	F6C1 40	TEST CL,40	
00C49358	75 08	JNZ SHORT 00C49362	
00C4935A	6A 01	PUSH 1	
00C4935C	E8 D5D7FDFF	CALL 00C26B36	
00C49361	59	POP ECX	
00C49362	6A 00	PUSH 0	
00C49364	C705 B852C500 D	MOV DWORD PTR DS:[C552B8],0C55FD4	ASCII "RC"
00C4936E	E8 F402FEFF	CALL 00C29667	
00C49373	59	POP ECX	
00C49374	E8 FCF3FEFF	CALL 00C38775	
00C49379	8BF8	MOV EDI,EAX	
00C4937B	A1 D8E6C500	MOV EAX,DWORD PTR DS:[C5E6D8]	
00C49380	8B48 70	MOV ECX,DWORD PTR DS:[EAX+70]	
00C49383	3348 48	XOR ECX,DWORD PTR DS:[EAX+48]	

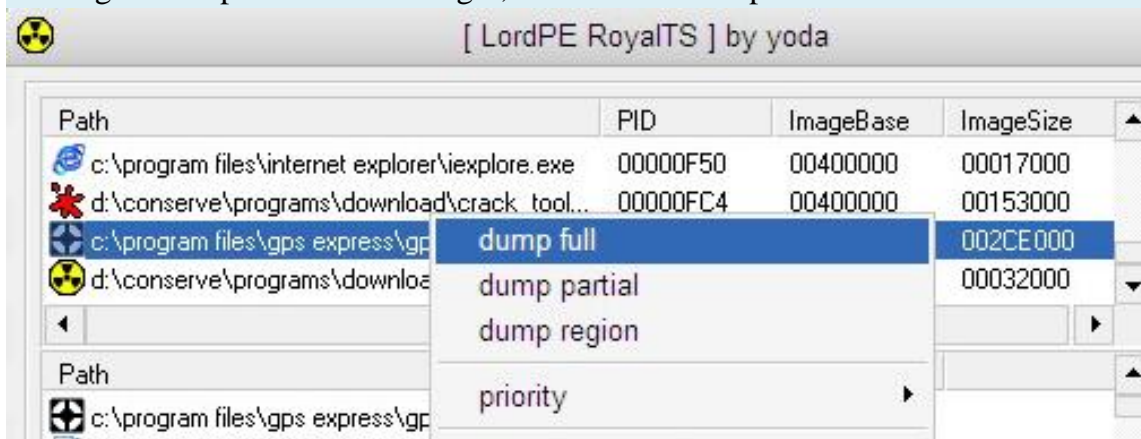
Now pull-down search command RETN me, in this order on me a bit with a CALL EDI:

00C493D3	A1 D8E6C500	MOV EAX,DWORD PTR DS:[C5E6D8]	
00C493D8	8B48 74	MOV ECX,DWORD PTR DS:[EAX+74]	
00C493DB	3348 3C	XOR ECX,DWORD PTR DS:[EAX+3C]	
00C493DE	3348 10	XOR ECX,DWORD PTR DS:[EAX+10]	
00C493E1	2BF9	SUB EDI,ECX	
00C493E3	FFD7	CALL EDI	
00C493E5	8BD8	MOV EBX,EAX	
00C493E7	5F	POP EDI	
00C493E8	8BC3	MOV EAX,EBX	
00C493EA	5E	POP ESI	
00C493EB	5B	POP EBX	
00C493EC	C3	RETN	

-You set a Break Point-like (F2) and then press Shift-F9 to break it in the (BP to delete). Now the press F7 small, we will come:

00401000	A1 FCB44F00	MOV EAX,DWORD PTR DS:[4FB4FC]	
00401005	C1E0 02	SHL EAX,2	
00401008	A3 00B54F00	MOV DWORD PTR DS:[4FB500],EAX	
0040100D	57	PUSH EDI	
0040100E	51	PUSH ECX	
0040100F	33C0	XOR EAX,EAX	
00401011	BF 58635200	MOV EDI,GPSExpre.00526358	
00401016	B9 44AB5200	MOV ECX,GPSExpre.0052AB44	
00401018	3BCF	CMP ECX,EDI	
0040101D	76 05	JBE SHORT GPSExpre.00401024	
0040101F	2BCF	SUB ECX,EDI	
00401021	5C	POP ECX	

Lao-authors say this is the OEP su.Chung we remember the 401,000 Address and now it's one small dump it. (To the original program and Olly OEP by it, not anything close) Open father Lord-up PE , select the file and then running GPSExpress.exe Click right, choose "full dump"



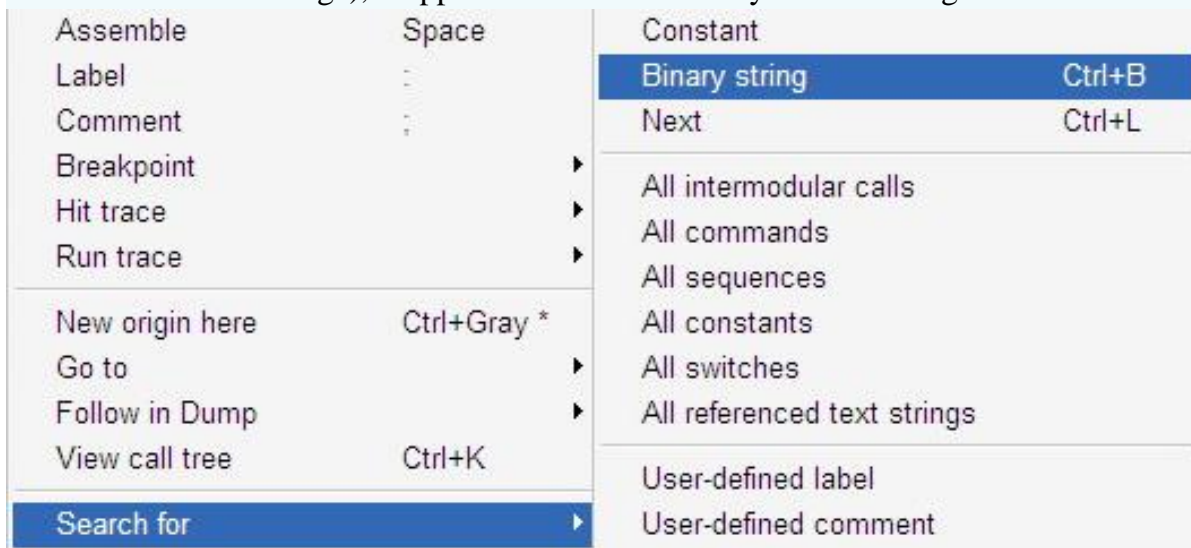
-Save will default to the name dumped.exe (if you do not change the name). Done is a code.

Removal II.Tieu anti-dump and revised IAT:

-As the name of the section 1, this program is pack it in and Armadillo have excellent anti-dump chiêu whether we

should find the OEP, it is the dump file dumped.exe Vàn hong.Gio will fix IAT dump file to run perfect.

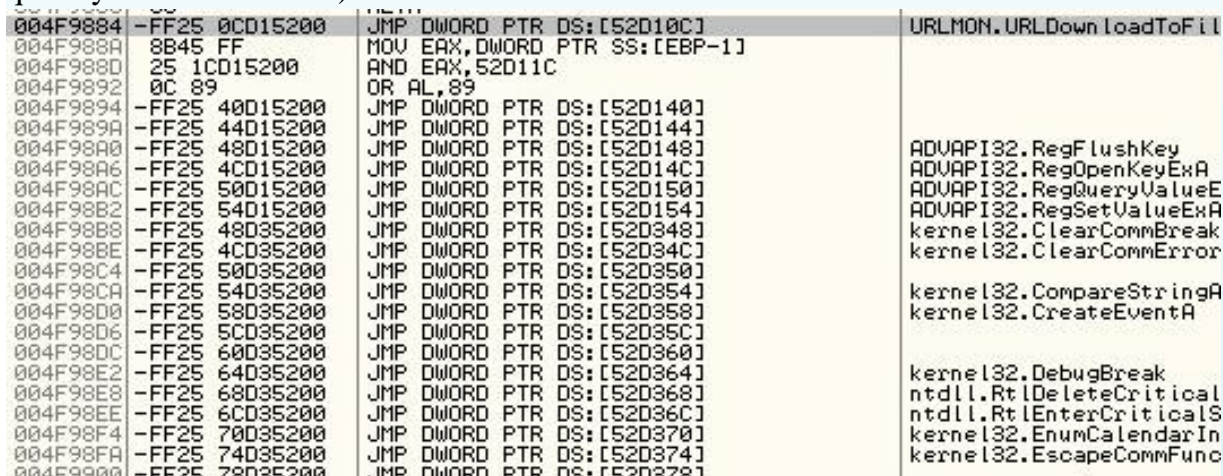
-Now they have told you the resources to Olly, one hour back, OEP are not really true? Press Ctrl-B (or right click and then select the image), it appears the box for Binary Search String:



-Type in the image:



Then-OK.no Search start at first we just tien.Nhung Ctrl-B from its Search more, until the stop here (click wá quickly lose that bear àh):



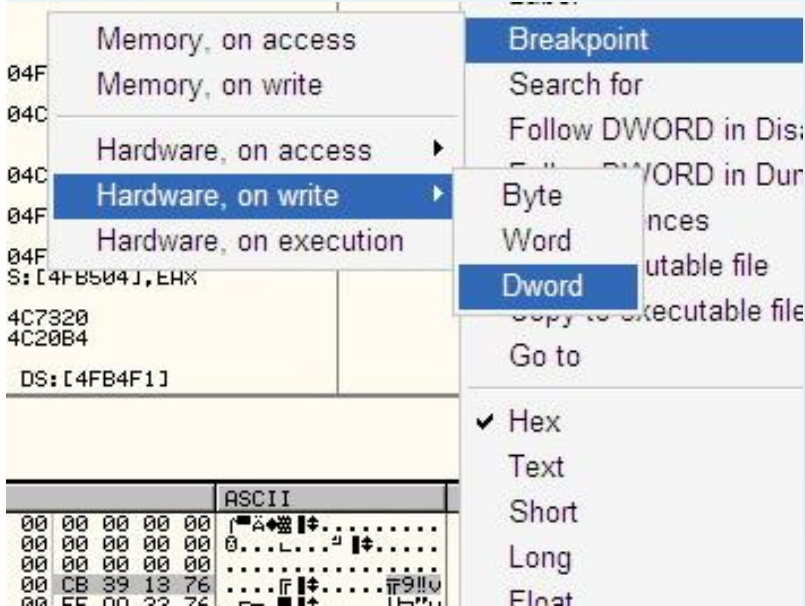
Then one-click to select this line at the [dump Follow - Memory Address](#):



Look-down window below, pull up a bit and then the bowl as Byte type:

Address	Hex dump	ASCII
0052D000	F4 DF 8E 04 B2 DE 12 00 00 00 00 00 00 00 00
0052D001	01 00 00 00 1C 00 00 00 BC DE 12 00 00 00 00
0052D002	00 00 00 00 00 00 00 00 00 00 00 00 00 00
0052D003	00 00 00 00 C9 DE 12 00 00 00 00 00 CB 39 13 76
0052D004	16 A9 C2 00 DF DE 12 00 00 00 00 00 55 AA 22 76
0052D005	0C A9 C2 00 F5 DE 12 00 03 DF 12 00 15 DF 12 00
0052D006	23 DF 12 00 33 DF 12 00 47 DF 12 00 00 00 00 00
0052D007	8A DC C2 00 4A E0 C2 00 6D 69 DE 77 9A 22 DD 77
0052D008	10 34 00 77 B1 63 0E 77 CB 00 C3 00 CB 0E 13 00

The Byte-by you may be different, but I aged and authors in the same Address 0052D100 it is.
-To complete the right-click, select the following **Breakpoint - Hardware, write on - Dword**:



-And now Restart the program there. (Ctrl-F2). Press Shift-F9, waiting for it to stop at first Breakpoint Hardware:

77C42F43	F3:85	REP MOVSD PTR ES:[EDI],DWORD PTR DS:[EDX*4+77C43058]
77C42F45	FF2495 5830C477	JMP DWORD PTR DS:[EDX*4+77C43058]
77C42F4C	8BC7	MOV EAX,EDI
77C42F4E	BA 03000000	MOV EDI,3
77C42F53	83E9 04	SUB ECX,4
77C42F56	72 0C	JB SHORT msvcrt.77C42F64
77C42F58	83E0 03	AND EAX,3
77C42F5D	83C0	AND ECX,ECX

00C461D5	8B85 10C8FFFF	MOV EAX,DWORD PTR SS:[EBP-37F0]	GPSExpre.0052D10C
00C461D8	83C0 04	ADD EAX,4	
00C461DE	8985 10C8FFFF	MOV DWORD PTR SS:[EBP-37F0],EAX	
00C461E4	^E9 CEFCEFFF	JMP 00C45EB7	
00C461E9	FF15 90F2C400	CALL DWORD PTR DS:[C4F290]	kernel32.GetTickCount
00C461EF	2B85 A0C4FFFF	SUB EAX,DWORD PTR SS:[EBP-3B60]	
00C461F5	8B8D A4C4FFFF	MOV ECX,DWORD PTR SS:[EBP-3B5C]	
00C461FB	6BC9 32	IMUL ECX,ECX,32	
00C461FE	81C1 D0070000	ADD ECX,7D0	
00C46204	3BC1	CMP EAX,ECX	
00C46206	^76 07	JBE SHORT 00C4620F	
00C46208	C685 34C8FFFF 0	MOV BYTE PTR SS:[EBP-37CC],1	
00C4620F	83BD E4C6FFFF 0	CMP DWORD PTR SS:[EBP-391C],0	

Lao-authors say this is the place playing Anti-Dump.Ta look further down the father saw a JBE.

00C461D5	8B85 10C8FFFF	MOV EAX,DWORD PTR SS:[EBP-37F0]	GPSExpre.0052D10C
00C461D8	83C0 04	ADD EAX,4	
00C461DE	8985 10C8FFFF	MOV DWORD PTR SS:[EBP-37F0],EAX	
00C461E4	^E9 CEFCEFFF	JMP 00C45EB7	
00C461E9	FF15 90F2C400	CALL DWORD PTR DS:[C4F290]	kernel32.GetTickCount
00C461EF	2B85 A0C4FFFF	SUB EAX,DWORD PTR SS:[EBP-3B60]	
00C461F5	8B8D A4C4FFFF	MOV ECX,DWORD PTR SS:[EBP-3B5C]	
00C461FB	6BC9 32	IMUL ECX,ECX,32	
00C461FE	81C1 D0070000	ADD ECX,7D0	
00C46204	3BC1	CMP EAX,ECX	
00C46206	^76 07	JBE SHORT 00C4620F	
00C46208	C685 34C8FFFF 0	MOV BYTE PTR SS:[EBP-37CC],1	
00C4620F	83BD E4C6FFFF 0	CMP DWORD PTR SS:[EBP-391C],0	

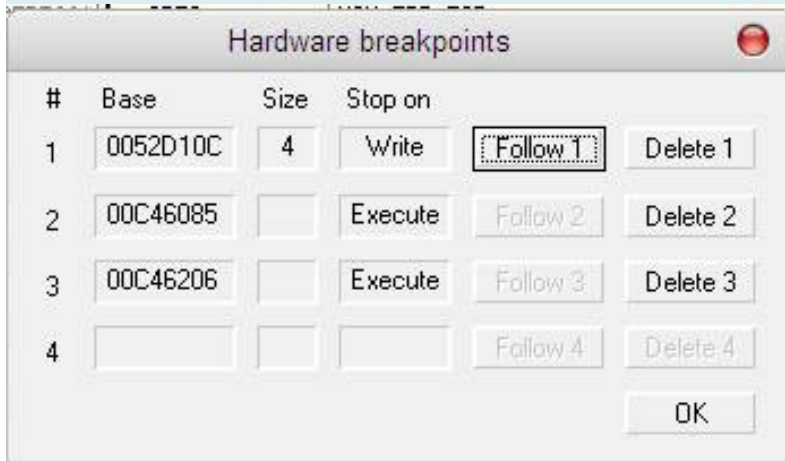
-And this father is always jumping, we also set a similar BP on here:

00C46206	^76 07	JBE SHORT 00C4620F		
00C46208	C685 34C8FFFF 0	MOV BYTE PTR SS:[EBP-37CC],1		Backup
00C4620F	83BD E4C6FFFF 0	CMP DWORD PTR SS:[EBP-391C],0		Copy
00C46216	^0F85 8A000000	JNZ 00C462A6		Binary
00C4621C	0FB685 90C4FFFF	MOVZX EAX,BYTE PTR SS:[EBP-3B70]		Assemble
00C46223	85C0	TEST EAX,EAX		Label
00C46225	^74 7F	JE SHORT 00C462A6		Comment
00C46227	6A 00	PUSH 0		Breakpoint
00C46229	8B85 94C4FFFF	MOV EAX,DWORD PTR SS:[EBP-3B6C]		
00C4622F	C1E0 02	SHL EAX,2		
00C46232	50	PUSH EAX		
00C46233	8B85 0CC7FFFF	MOV EAX,DWORD PTR SS:[EBP-3B74]		
00C46239	0385 8CC4FFFF	ADD EAX,DWORD PTR SS:[EBP-3B74]		
00C4623F	50	PUSH EAX		
00C46240	E8 7B1B0000	CALL 00C462A6		
00C46245	83C4 0C	ADD EAX,4		
00C46248	8B85 94C4FFFF	MOV EAX,DWORD PTR SS:[EBP-3B6C]		
00C4624E	C1E0 02	SHL EAX,2		
00C46251	50	PUSH EAX		
00C46252	FFB5 6CC8FFFF	PUSH 0		
00C46258	8B85 0CC7FFFF	MOV EAX,DWORD PTR SS:[EBP-3B74]		
00C4625E	0385 8CC4FFFF	ADD EAX,DWORD PTR SS:[EBP-3B74]		
00C46264	50	PUSH EAX		
00C46265	E8 187F0000	CALL 00C462A6		
00C4626A	83C4 0C	ADD EAX,4		
00C4626D	6A 01	PUSH 1		
00C4626F	8B85 94C4FFFF	MOV EAX,DWORD PTR SS:[EBP-3B6C]		
00C46275	C1E0 02	SHL EAX,2		
00C46278	50	PUSH EAX		
00C46279	8B85 0CC7FFFF	MOV EAX,DWORD PTR SS:[EBP-3B74]		
00C4627F	0385 8CC4FFFF	ADD EAX,DWORD PTR SS:[EBP-3B74]		
00C46285	50	PUSH EAX		

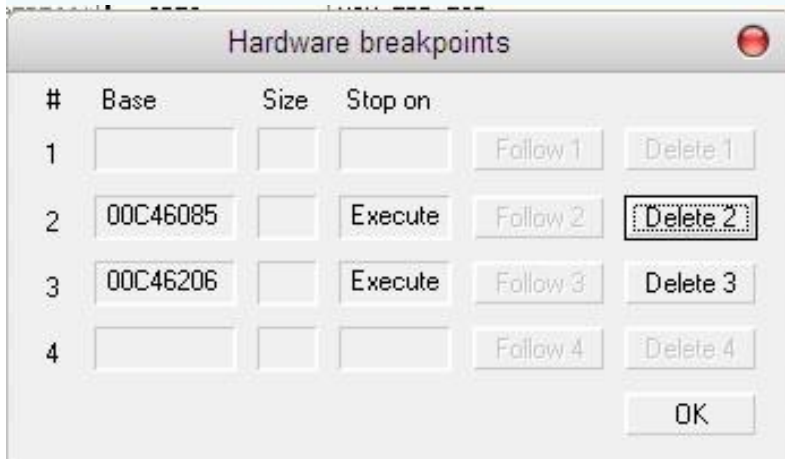
Restart-timer again. (Ctrl-F2). Before we must do to remove a HW BP excess Debug then go da.Chon Hardware breakpoints:



-Then see this:



-Then, we delete father Write attend, for it is:

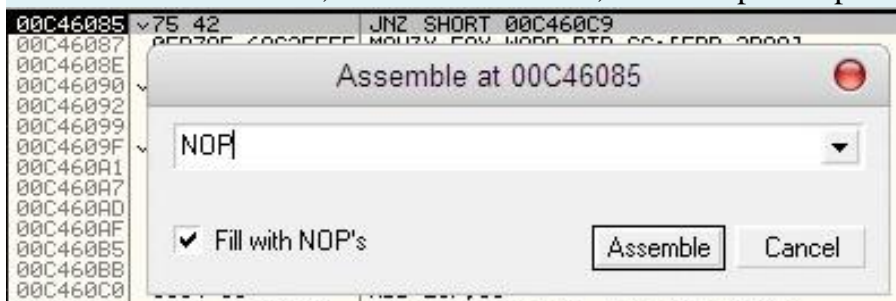


Then we OK.Gio-Shift-F9 again, and then it stops running at first HW BP:

00C46085	75 42	JNZ SHORT 00C460C9
00C46087	0FB785 68C2FFFF	MOVZX EAX,WORD PTR SS:[EBP-3D98]
00C4608E	85C0	TEST EAX,EAX
00C46090	74 0F	JE SHORT 00C460A1
00C46092	0FB785 68C2FFFF	MOVZX EAX,WORD PTR SS:[EBP-3D98]

Hardware breakpoint 2 at 00C46085

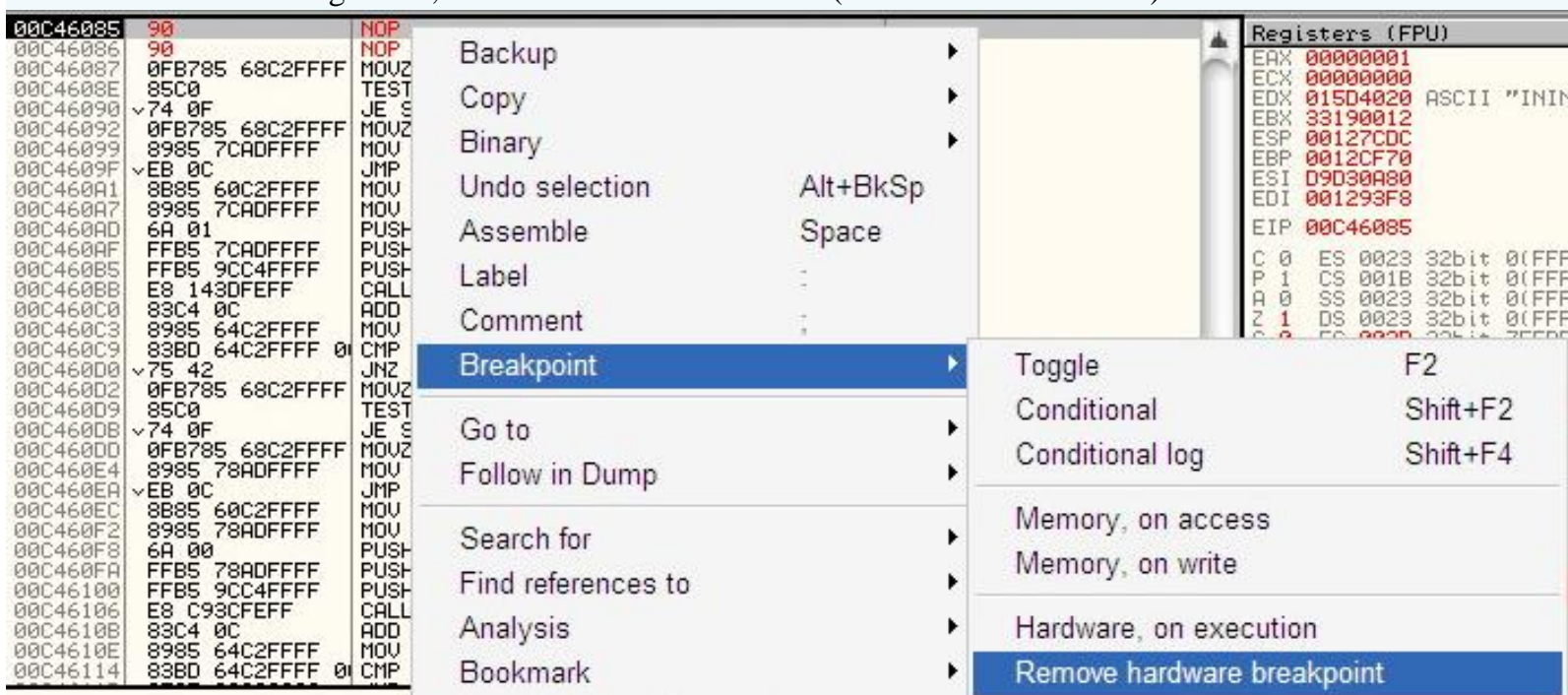
Time-out for it from us, we NOP for it then, and then press Space to like:



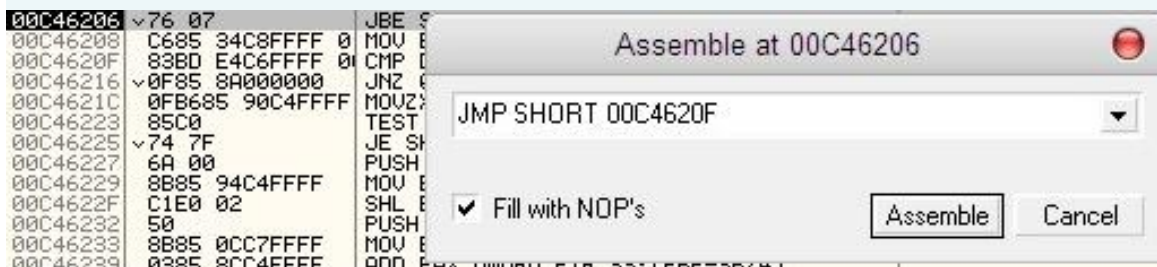
-Okie.Ta be quỳ nay.He the summer, jumped out small children.

00C46085	90	NOP
00C46086	90	NOP
00C46087	0FB785 68C2FFFF	MOVZX EAX,WORD PTR SS:[EBP-3D98]

Then one-click em.Ta right at it, and remove the HW BP this (the value of the benefit):



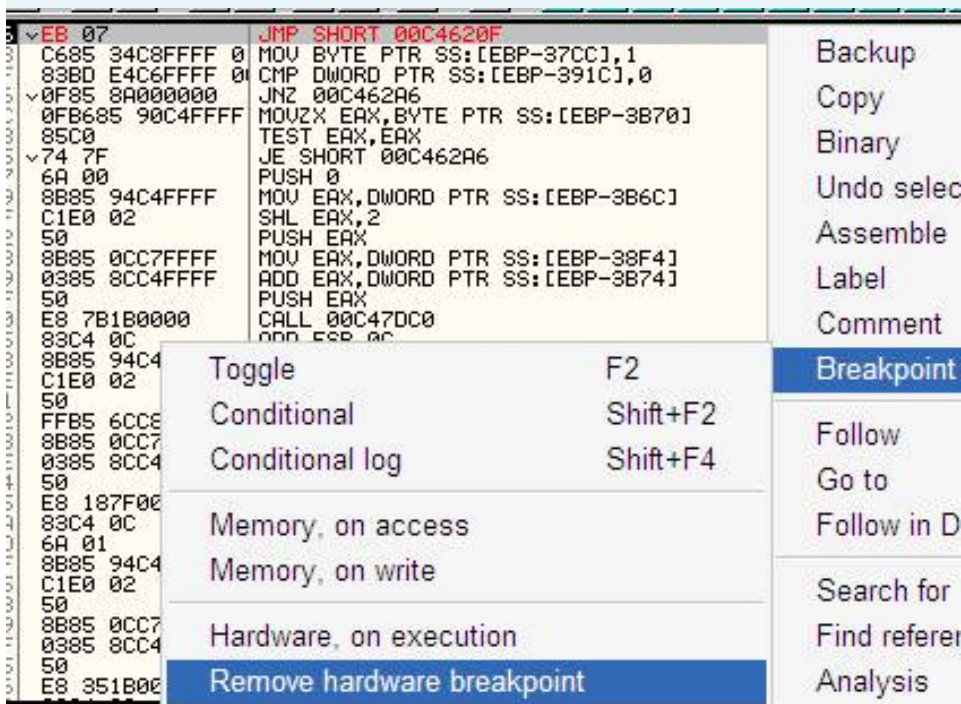
One-Shift-F9 to 2 children, and also edit it by (it always jump directly to the Cmp line under it 2):



-Ready are:

00C46206	EB 07	JMP SHORT 00C4620F
00C46208	C685 34C8FFFF 0	MOV BYTE PTR SS:[EBP-37CC],1
00C4620F	83BD E4C6FFFF 0	CMP DWORD PTR SS:[EBP-391C],0
00C46216	0F85 8A000000	JNZ 00C462A6

Time-HW BP also removed seats in the new edit:

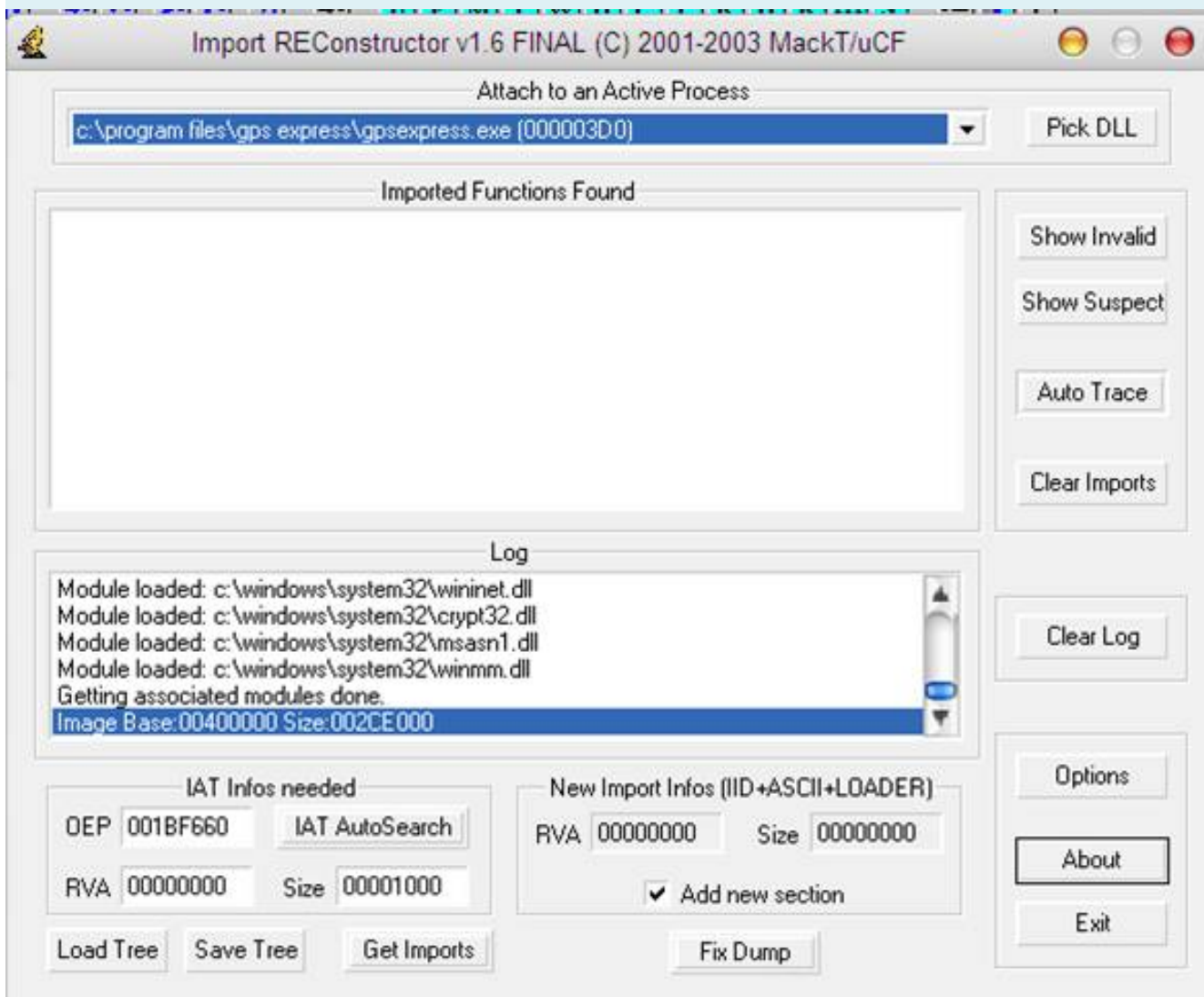


-And now to do is press F9 to run the program .. he he suspended, and then finish it up:

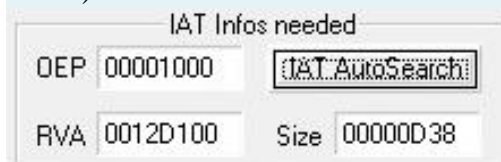
00C467C0	07	POP ES	Modification of segment
00C467C1	07	POP ES	Modification of segment
00C467C2	13EC	ADC EBP,ESP	
00C467C4	5F	POP EDI	
00C467C5	D190 CC820126	RCL DWORD PTR DS:[EAX+260182CC],1	

Debugged program was unable to process exception

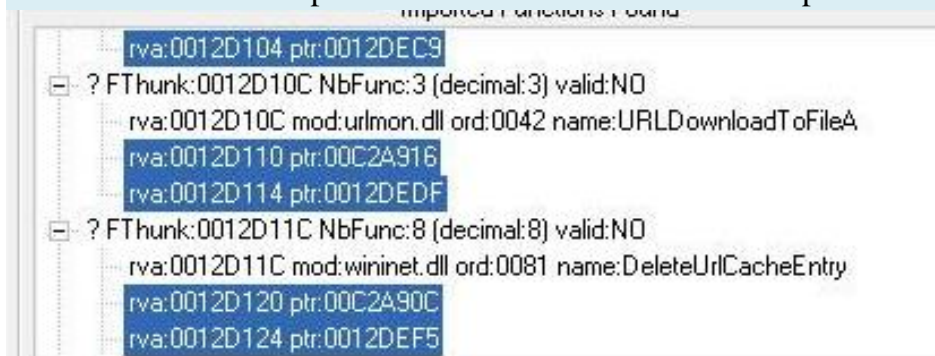
-To the principle of small, arm machines that I like the game. Now open them up ImportREC This, he he (to complete the happy, smiling for sướng):



-Select [thằng gpsexpress.com](http://gpsexpress.com) you need to treat ly. Cac OEP also remember the original is not how many, heaven oi I have to be sure that, as 401,000. I take exception to the 400,000 hectares in 1000 (down the wrong grade 1 school home). We enter 1000 on children in OEP This [ImportREC](#). Then click [IAT AutoSearch](#), is this:

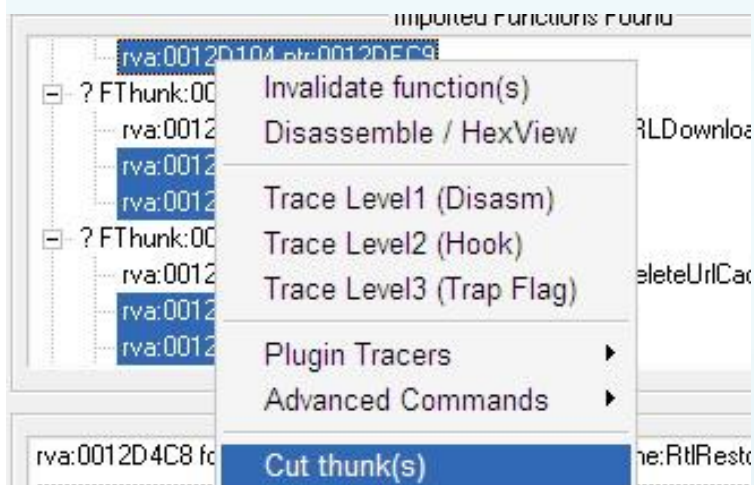


Now one-click Get Imports and then click Show Invalid report to see your child's treatment:

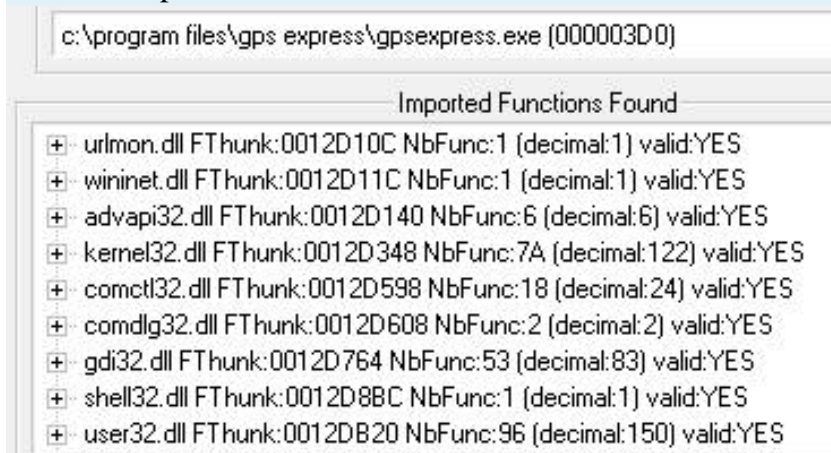


-A ha, they mec we are some months [mắm](#) this complaint nè, time we have to click on them, select Cut Thunk (s)

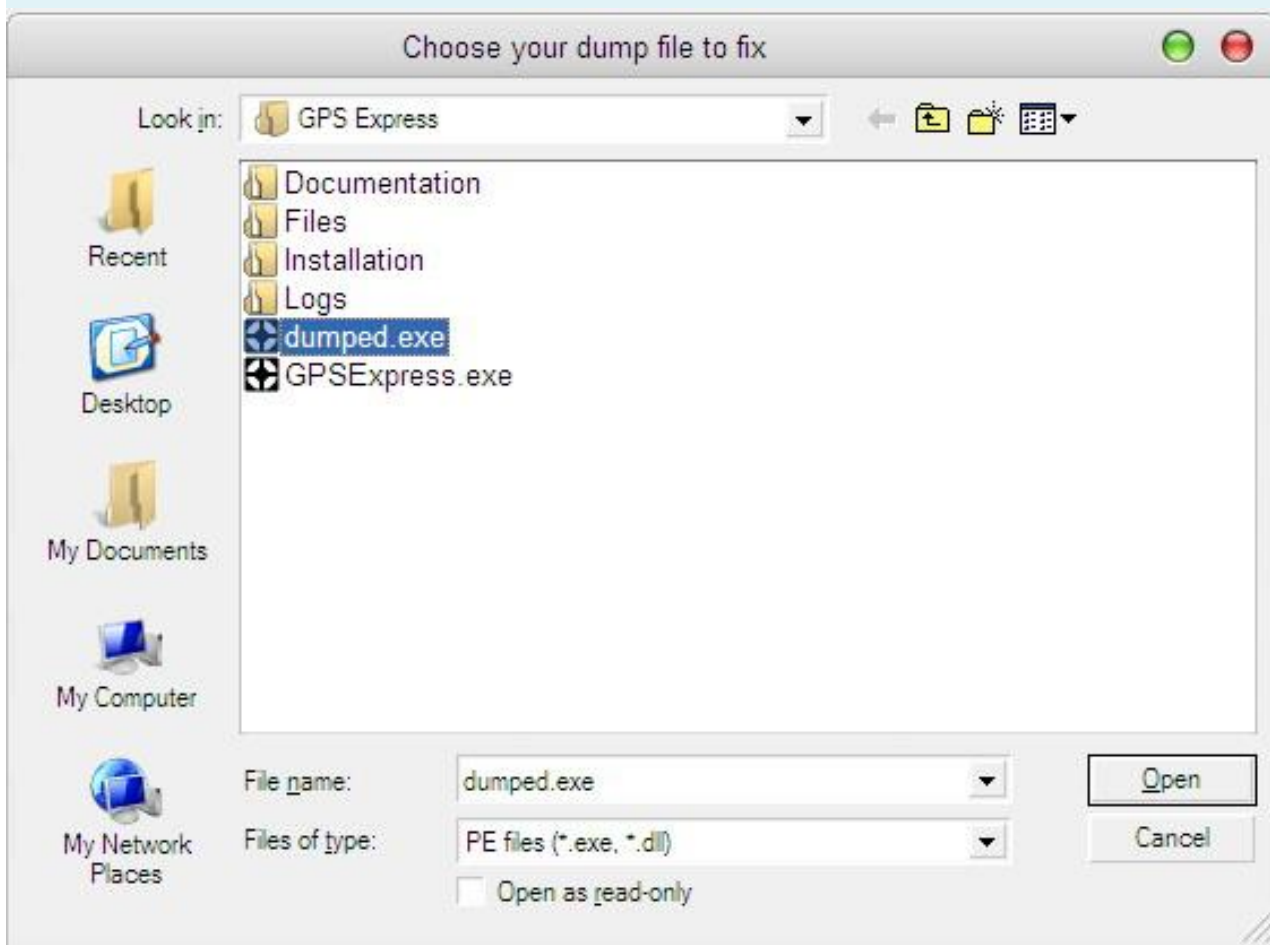
for both the children ("sever" bøn them home).



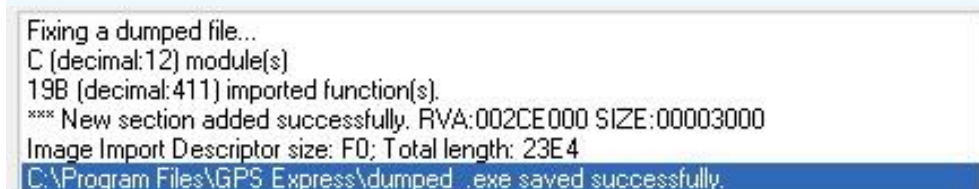
Cut-to complete the children also do all that ... YES other other:



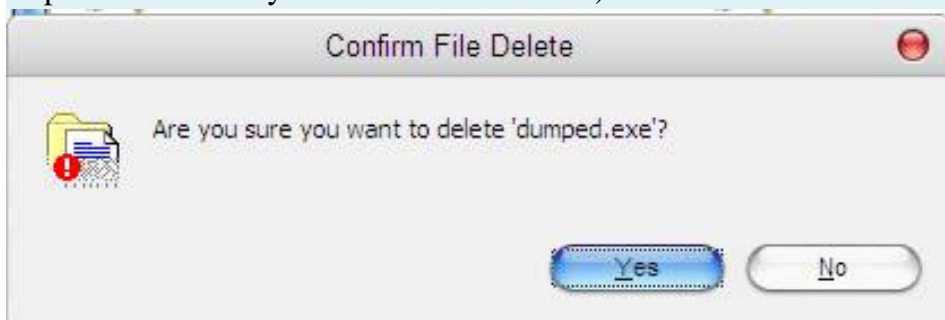
Dear-winning finish up, we ask them to [dump Fix](#), we choose initial thằng [dumped.exe](#) are Orphaned kìa of offenders:



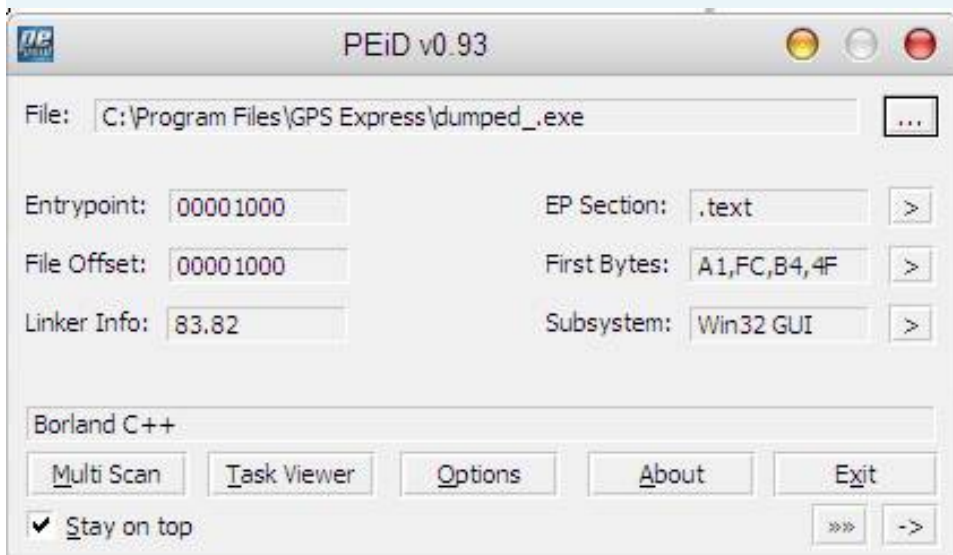
Okie-one, so that they will report suong nha.Ta temporary special children được rồi ([ImportREC](#) they were abusing value).



- (Thằng Orphaned [dumped.exe](#) This is also the.Em cows eventually fall for it, delete it always sướng mat.A this step is not necessary to make fun thoi.hi HI):



Other time-other is running the file that they [dumped_.exe](#) This Save for one, lickerish roi.Load it PEID report found:



-It's mission as completed, you want to do it any more Crack Crack, I just bit unpack only vague, but it is always Crack [Source long, durable dot, hard](#) training.

I self-summary some content next time, having this type of soft, you have the following steps:

- Check all items in part by Olly Exception to the Debug suon will be.
 - Set a BP in the "BP CreateThread in Cmdbar.
- Since the BP-jump to OEP are looking for. Note CALL command a previous order that RETN.
OEP dump-in and fix IAT similar manner on.

*** Knowledge in tut probably quite old, but also help us deal with some of the Soft pack in the old Armadillo.

But outside-soft form anti-dump also soft before touching associated anti-dump we must meet no less a gã coffee chosen as anti-Breakpoint, he disabled the majority of BP's we, as local Pets Olly suspend our nua.Vay of what he here? Please welcome to see [part 2](#) small (not in a very complete Capture image)

The media-thank all of you in the REA has helped them get more knowledge and experience in Cracking. **While writing tut confusing** if errors have been expected forgiveness and guidance them.Hix ...

*** Bom Quang Volume 2:

- **KHO** ... human life is always part of any living being ... ardent, the explosion as something that is italy as having no good story! The number is calculated Trickyboy I wrote part 2 tut tut 1 day after the software that the "sanctity trời it" this "death machine"> ... <!

- "Death Machine" here he said this is the natural expiration using the procedure described in the only appropriate when the software is xài "Free Trial." The first should advise relatives xuc he want to do this by if they are in any free, then set it small, the complete set xuc but do not remove it .. xó (Do not know how the changes when it expires đành should say that, at the offender's bags Source wá it big ... hix hix ..)

In tut-2 also would raise a number of deficiencies in the Newbie 1. Dieu bags valued as a tut is the match between words and pictures anh.Tung steps must be clear, not too brief as readers from embarrassing ... to disappointment.

(*** *Update tut 1*)

As the part-1, I say the aged images that illustrate the author than any other time I do a few lines of command and distasteful this has answers roi.Do program expires at home so I use đành out services, but for the bags to the Windows XP Service Pack 2.Tinh flags as the new detection is the difference compared with aged ta.Ban first thought when the other operating system will be different and Offset (address command line) but the hours I know more about the other is a line lenh.Vay speak with her children when the test is on [Windows XP SP2, 1.8GHz Celeron](#), see quite the same-aged, but I do successful in [Windows SP 1, Pentium 4, 2.4GHz](#) and from now, if anyone has any tut, I will always if information systems are used to the Newbie bags not be embarrassing. Okie?

(The difference relatives tut see more English and then compared with bags of home tut)

-In addition, differences in the operating system seem quite important, proof is how I do on Windows XP SP1 failed on SP2, cause the owners of the first sleeping bag, 2 bags are not out by all authors (as do the aged that are kia SP2), the objective is probably right that I sit in is crazy ^ _ ^.

-I have to find SP2 and have many functions to load the library when running programs (perhaps due to security is strong) and some of which affect the Debug is the soft end Pack.Nen Finally, I should let that be to find a SP1 not expire with this soft ... hix hix (the trial is probably always on Win 2000 but the date is 1 only 24 pieces of gold that!)

-As introduced in section 1, in this part 2 we will see a high of coffee chosen as his **anti-Breakpoint**. He just has this way is the most powerful or dissemble a few breakpoint (BP) which is set in Olly (or debug a program is). Khốn where BP is the important new chit chu.hix ...

-In this tut, aged authors infamous "Hec Cu Lec" (not want to consider a name more) Newbie ensure we have a basic knowledge of **OutputDebugStringA** and **IsDebuggerPresent** (say in the Newbie juiceless sources added). But Why not, I ensure that I read the tut as you will fulfill tun dropped (although Hồng understand anything, he he).

-And before I start from 2 to explain which "is the" Newbie đám we must understand, explain

briefly only small.

**** OutputDebugStringA:** conduct na say, this is a function is responsible for calling the error message when we attempt any Debug programs (such as Into Trace, Trace Over ...). And it has come apart, the Olly (or the Debug any) to close the door. Khỏ that, so we must thoroughly it deceive, it does not appear again. (how is little more bit). Note always is when the xuc he's been soft Statistics of the Armadillo (3.x - 4.x), you should thoroughly immediately as soon as this function has soft Load in Olly.O's first soft bags do not do this because I do not see a computer error (on the SP2 see it now)

**** IsDebuggerPresent:** this is due as the name of it, "is" means test (the same as in the programming), "Debug" Debug is the "Present" means the current (same as the current which in English). In summary the "check immediately at the time there are thằng Debug programs or not? Yes, the springiness of a NAG chùi and App Close (close the program). Other other, but I say the chả bít What nha.Chung them where the trick is also how he said this is mòm to their work is the duoc.Tren SP1 occasionally it certainly is, but more often SP2. As work on SP1, having little trouble with it so I did not know how thoroughly no.Viec which means that "if tut follow us on SP2" == "failed not know" (you expect the high hand help).

-Add a new concept (with bags) that aged authors to consider using a **script** with **OllyScript** plugin. Talking about the past is always a **script** for her con.Cac **script** is actually the code is defined available (usually a text file format. "Txt"), for as fast a process that we have to do is step by step New duoc.Giong long as you find an exact OEP is too many steps in the Olly sometimes use **Generic OEP Finder** plugin or a **script** used to find the OEP faster.

-Here, Hec Cu Lec say that the **script** does not fit the soft pack with Armadillo type nay.Va I also try to use a **script** to the Armadillo thằng but is in a Target khác.Khi bring this through the soft thua . That is limited between the **Script, Plugin** compared with the "craft" is not always used duoc.Do the most important is you must have strong knowledge, experience wires, with 2 things that do not fear much dependent. (I lack of first 2 ^ _ ^)

-Oai, the bags of Bom then, only every third child of main dishes Lầu today on, make sure not colic not pay (Bup.., Accordingly lost mine .. hix hix ...)

Section 2: Unpacking Armadillo v4.x (anti-BP) (2 tut by Trickyboy)

Target	Easy DVD to VCD burner v2.0.43
Available	http://intechhosting.com/ access ~ / ARTeam / tools / EasyDVDtoVCD.exe

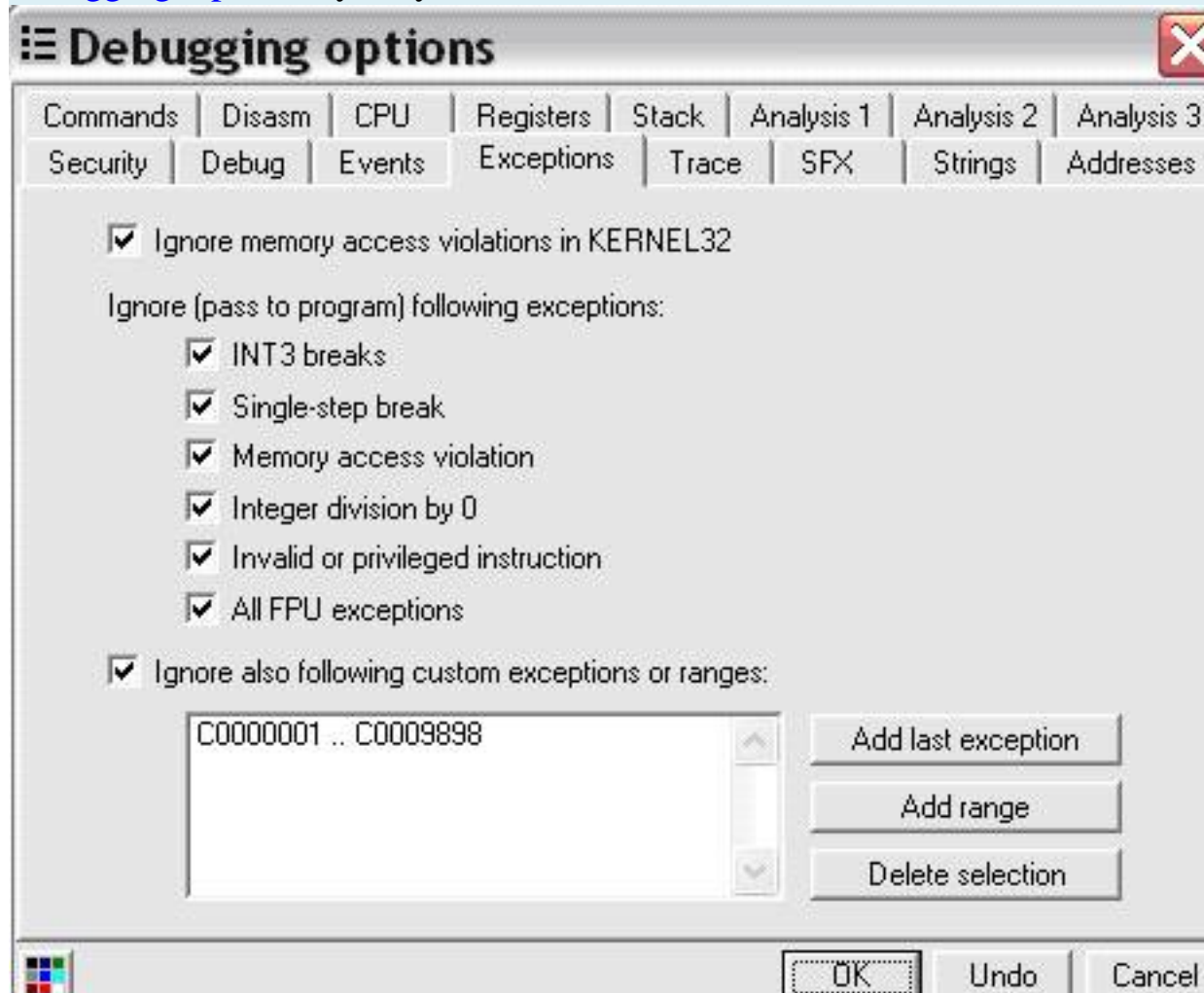
Tools	OllyDbg 1: 10 , ImpRec , LordPE , Armaccess.dll , Olly Plugin: Hide
	Debugger, CmdBar

(Test system is [Windows XP SP1](#), [Pentium 4](#))

-Oai, toys kì also quite the same as the one hectare, he added a library thui [Armaccess.dll](#) (explained below). Forget it, the 2 tut were aged Hec Cu Lec as **Level: Intermediate** (suffering, the new bit on this life, very intelligent man àh).

IVuot through anti-BP (**also** called the anti-anti Breakpoint, ha ha, Khoai said invasions ghê)

-The scenario is perfect Armadillo's children, we must also check all the cards in the [Exception debugging Options](#) by Olly.



-Then, move on already available, she Target Load dễ thương to [EasyDVDtoVCD.exe](#). (But he says they hate her attempt to make him lose a few days).

-If you have any message they choose **NO** ghen her children:

Compressed code?



Quick statistical test of module 'EasyDVDt' reports that its code section is either compressed, encrypted, or contains large amount of embedded data. Results of code analysis can be very unreliable or simply wrong. Do you want to continue analysis?

Life-ngộ that, it is that we welcome any NO Ah, he he "refuses to be much more than that." Now we're here in the house:

004EC000	60	PUSHAD
004EC001	E8 00000000	CALL EasyDVDt.004EC006
004EC006	50	POP EBP
004EC007	50	PUSH EAX
004EC008	51	PUSH ECX
004EC009	0FCA	BSWAP EDX
004EC00B	F7D2	NOT EDX
004EC00D	9C	PUSHFD
004EC00E	F7D2	NOT EDX
004EC010	0FCA	BSWAP EDX
004EC012	EB 0F	JMP SHORT EasyDVDt.004EC023
004EC014	B9 F8F8F8F8	MOV ECX, F8F8F8F8

-Now is looking down below the CmdBar small, I typed a few things to do:

Command :

-Be sure not to type the incorrect capitalization nhé, then enter it to set a BP do. Muon where bit Where do the way in small tut part 1 (writing to create the perfect way to contact you).

-But must also address àh (Huế less capital, he he). The reason is that when set in BP, if we try to Shift-F9 khôn nổi it will not stop with BP that a pause in which to prepare **Terminate** (the Treo). Hix hix, to the local hard Olly undoubtedly households , one hour to press a button nhất **B** Olly nha.No out in this way:

Address	Module	Active	Disassembly
77E61A94	kernel32	Always	PUSH EBP

One-double-click on this line will be to set points at BP:

77E61A94	55	PUSH EBP
77E61A95	8BEC	MOV EBP,ESP
77E61A97	51	PUSH ECX
77E61A98	51	PUSH ECX
77E61A99	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
77E61A9C	53	PUSH EBX
77E61A9D	8B5D 14	MOV EBX,DWORD PTR SS:[EBP+14]
77E61AA0	56	PUSH ESI
77E61AA1	8B35 B012E677	MOV ESI,DWORD PTR DS:[<&ntdll.NtProtect ntdll.ZwProtectVirtua

Ra-BP is set when the first function, the program will be detected immediately because of anti-

BP way of it, it should assign hours to go a little, small not accept the "xà you", we remove this BP seats, down which under a few lines (not a small line 2, have at it with his children on thăng, suffering ...), time set in BP is also where, blindly press the F2 (considered the shooting, UI skin, press F9 flipped):

77E61A94	55	PUSH EBP
77E61A95	8BEC	MOV EBP,ESP
77E61A97	51	PUSH ECX
77E61A98	51	PUSH ECX
77E61A99	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
77E61A9C	53	PUSH EBX
77E61A9D	8B5D 14	MOV EBX,DWORD PTR SS:[EBP+14]
77E61AA0	56	PUSH ESI
77E61AA1	8B35 B012E677	MOV ESI,DWORD PTR DS:[<&ntdll.NtProtect
77E61AA7	57	PUSH EDI
77E61AA8	8B7D 08	MOV EDI,DWORD PTR SS:[EBP+8]
77E61AAB	8945 F8	MOV DWORD PTR SS:[EBP-8],EAX
77E61AAE	8D45 14	LEA EAX,DWORD PTR SS:[EBP+14]
77E61AB1	50	PUSH EAX
77E61AB2	6A 04	PUSH 4
77E61AB4	8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]
77E61AB7	50	PUSH EAX
77E61AB8	8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]
77E61ABB	50	PUSH EAX
77E61ABC	57	PUSH EDI

-Now is an press Shift-F9 then. Break a (hate to leave), it stops at the King set BP:

77E61A94	55	PUSH EBP
77E61A95	8BEC	MOV EBP,ESP
77E61A97	51	PUSH ECX
77E61A98	51	PUSH ECX
77E61A99	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
77E61A9C	53	PUSH EBX
77E61A9D	8B5D 14	MOV EBX,DWORD PTR SS:[EBP+14]
77E61AA0	56	PUSH ESI
77E61AA1	8B35 B012E677	MOV ESI,DWORD PTR DS:[<&ntdll.NtProtect
77E61AA7	57	PUSH EDI
77E61AA8	8B7D 08	MOV EDI,DWORD PTR SS:[EBP+8]
77E61AAB	8945 F8	MOV DWORD PTR SS:[EBP-8],EAX
77E61AAE	8D45 14	LEA EAX,DWORD PTR SS:[EBP+14]
77E61AB1	50	PUSH EAX
77E61AB2	6A 04	PUSH 4
77E61AB4	8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]
77E61AB7	50	PUSH EAX
77E61AB8	8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]
77E61ABB	50	PUSH EAX
77E61ABC	57	PUSH EDI

-Then, remove BP place this (rule is clean mark after the cause). Then press Alt-F9 small, ragged here:

004CC356	70 07	JO SHORT EasyDVDt.004CC35F	
004CC358	7C 03	JL SHORT EasyDVDt.004CC35D	
004CC35A	EB 05	JMP SHORT EasyDVDt.004CC361	
004CC35C	E8 74FBEBF9	CALL FA38BED5	
004CC361	EB 5F	JMP SHORT EasyDVDt.004CC3C2	
004CC363	8D55 FC	LEA EDX,DWORD PTR SS:[EBP-4]	
004CC366	52	PUSH EDX	
004CC367	6A 02	PUSH 2	
004CC369	68 0C325000	PUSH EasyDVDt.0050320C	
004CC36E	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]	
004CC371	50	PUSH EAX	1
004CC372	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
004CC375	8B11	MOV EDX,DWORD PTR DS:[ECX]	
004CC377	52	PUSH EDX	
004CC378	FF15 10C14F00	CALL DWORD PTR DS:[<&KERNEL32.WriteProc	kernel32.WriteProcess
004CC37E	50	PUSH EAX	
004CC37F	F7D0	NOT EBX	

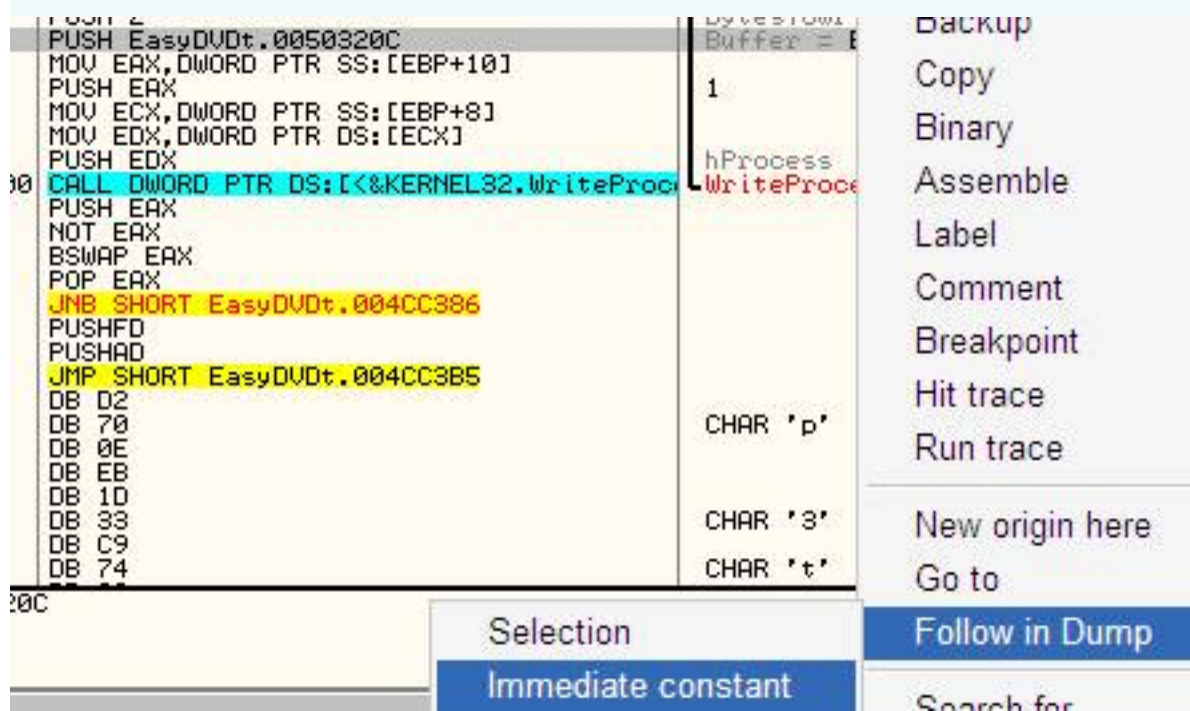
-There will be other bags, shelves, but it does wear out anything, we just Ctrl-A to identify it for sure the food, if it is the same as this is considered as the beginning of time, you have done true:

004CC356	70 07	JO SHORT EasyDVDt.004CC35F	
004CC358	7C 03	JL SHORT EasyDVDt.004CC35D	
004CC35A	EB 05	JMP SHORT EasyDVDt.004CC361	
004CC35C	E8	DB E8	
004CC35D	74 FB	JE SHORT EasyDVDt.004CC35A	
004CC35F	EB F9	JMP SHORT EasyDVDt.004CC35A	
004CC361	EB 5F	JMP SHORT EasyDVDt.004CC3C2	
004CC363	8D55 FC	LEA EDX,DWORD PTR SS:[EBP-4]	
004CC366	52	PUSH EDX	pBytesWritten
004CC367	6A 02	PUSH 2	BytesToWrite = 2
004CC369	68 0C325000	PUSH EasyDVDt.0050320C	Buffer = EasyDVDt.0050
004CC36E	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]	
004CC371	50	PUSH EAX	1
004CC372	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]	
004CC375	8B11	MOV EDX,DWORD PTR DS:[ECX]	
004CC377	52	PUSH EDX	hProcess
004CC378	FF15 10C14F00	CALL DWORD PTR DS:[<&KERNEL32.WriteProc	WriteProcessMemory
004CC37E	50	PUSH EAX	

-Instead of this?

004CC369	68 0C325000	PUSH EasyDVDt.0050320C	Buffer = EasyDVDt.0050
----------	-------------	------------------------	------------------------

At the command-JO has a mellow, but the same here, our rules are not looking up nha.Gio we choose the Buffer must then click this, select the image:



-Then look down the small window, saw this nè:

Address	Hex dump	ASCII
0050320C	60 E8 00 00 00 00 00 00 00 00 00 00 00 00	'z.....
0050321C	00 00 00 00 00 00 00 00 00 00 00 00 00 00

Now one-two tô 2 bytes and 60 E8 SpaceBar press to fix it:

ASCII

UNICODE

HEX +00

☒ Keep size

-City:

ASCII	δ
UNICODE	
HEX +02	EB FE

☒ Keep size

OK Cancel

Okie-di.Roi, say a little more, I do try 2 soft, see the rules modified to EB FE like the fact that (under the school health) .. prices are different is 60 E8, each soft each other in the number of Buffer nay.Over!

Now back-type box CmdBar as follows:

Command : `bp WaitForDebugEvent`

Enter cai.no re-set BP roi.Gio press F9 again to stop this at BP:

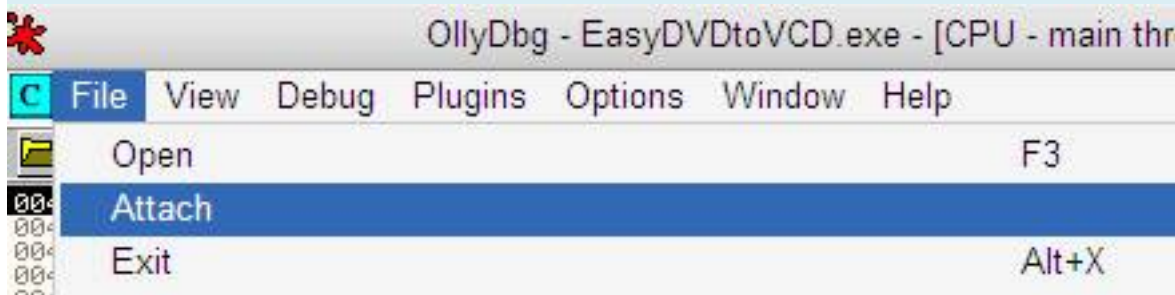
77EAF130	55	PUSH EBP
77EAF13D	8BEC	MOV EBP,ESP
77EAF13F	83EC 68	SUB ESP,68
77EAF142	56	PUSH ESI
77EAF143	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
77EAF146	8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]
77EAF149	50	PUSH EAX
77EAF14A	E8 5BB1FCFF	CALL kernel32.77E7A2AA
77EAF14F	8BF0	MOV ESI,EAX

BP-I delete di.Roi here Alt-F9 a phat.Toi here:

004C7F0F	. 85C0	TEST EAX,EAX	
004C7F11	.v0F84 AC260000	JE EasyDVDt.004CA5C3	
004C7F17	. 8B85 FCFDFFFF	MOV EAX,DWORD PTR SS:[EBP-204]	
004C7F1D	. 25 FF000000	AND EAX,0FF	
004C7F22	. 85C0	TEST EAX,EAX	
004C7F24	.v74 13	JE SHORT EasyDVDt.004C7F39	
004C7F26	. 8B0D 1C335000	MOV ECX,DWORD PTR DS:[50331C]	
004C7F2C	. 8379 20 00	CMP DWORD PTR DS:[ECX+20],0	
004C7F30	.v74 07	JE SHORT EasyDVDt.004C7F39	
004C7F32	. C685 FCFDFFFF	MOV BYTE PTR SS:[EBP-204],0	
004C7F39	> 68 10325000	PUSH EasyDVDt.00503210	
004C7F3E	. FF15 A4C14F00	CALL DWORD PTR DS:[I&KERNEL32.EnterCrit	pCriticalSection = E EnterCriticalSection

-Be sure to be asked to order **TEST EAX, EAX** small.

-Now we select File-Attach:

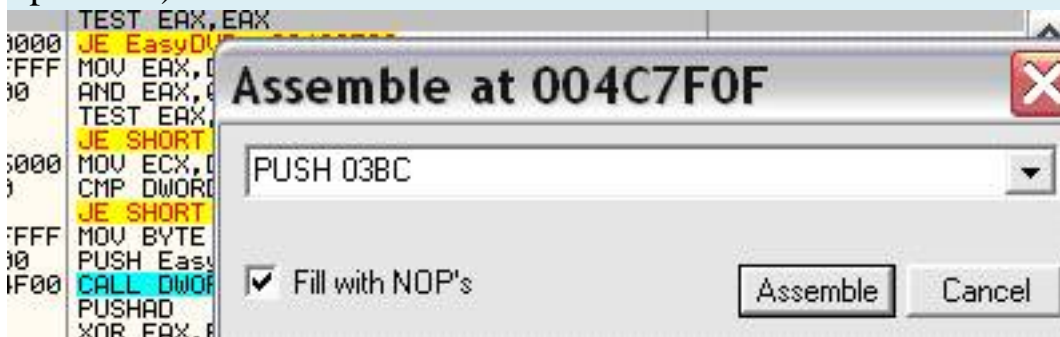


-Click the Name to the name ranked as follows:

Process	Name	Window	Path
00000194	20053515		C:\DOCUME~1\CTYPHA~1\LOCALS~1\Temp\200
000001B4	ccApp	ccApp	C:\Program Files\Common Files\Symantec
0000049C	ccEvtMgr		C:\Program Files\Common Files\Symantec
0000044C	ccSetMgr		C:\Program Files\Common Files\Symantec
000001E0	csrss		??\C:\WINDOWS\system32\csrss.exe
000005F0	DefWatch		C:\Program Files\Symantec AntiVirus\De
0000018C	EasyDVDt		C:\Program Files\EasyDVDtoVCD\EasyDVDt
000003BC	EasyDVDt		C:\Program Files\EasyDVDtoVCD\EasyDVDt
00000480	Explorer	Start Menu	C:\WINDOWS\Explorer.EXE

-Now have interesting day. Tu course, there are 2 files EasyDVDtoVCD.exe the load in memory, while we only have one child. So one month will not have the protection of anti-BP. Thang colored red is thằng we are working, so thằng rest is meat thằng need it. But if one more time again and then Olly Attach this month, the Cuc duoc. Ta have solved a number of stories have. Ba-con remember what month Offset 2 home, by this time I was 03BC (each work is that each other). And forget it, if in the process, and her children click Name to sort it but he was not the red again located on the much less will be failure in the steps ke. Do that, if it come up on , as more damaged, it must first work from home.

-Then Cancel Attach the window go. Start editing the command TEST EAX, EAX like this (press SpaceBar):



Okie-one, and then to fix the next:



-Finally, be as follows:

```

68 BC030000 PUSH 3BC
E8 1A739E77 CALL kernel32.DebugActiveProcessStop
90          NOP
90          NOP
90          NOP
90          NOP
. 25 FF000000 AND EAX,0FF

```

One-time press F8 a few things to it via the command line 2, when it is to:

```

68 BC030000 PUSH 3BC
E8 1A739E77 CALL kernel32.DebugActiveProcessStop
90          NOP
90          NOP
90          NOP
90          NOP
. 25 FF000000 AND EAX,0FF
. 85C0      TEST EAX,EAX

```

Wa-one doubt the value to write EAX, it is 1 nhé (aged Cu Hec Lec must say it is 1 new warrant for next steps - such as formula> ..., including using **script**):

```

Registers (FPU)
EAX 00000001

```

-Then, we can xuc his children now and then, we opened a second Olly 2 other (for Olly first for fun). Choose **File-Attach**. Then ranked Name to the name, choose File Offset is available as **03BC** above (or select the file below):

000001E0	csrss		C:\WINDOWS\system32\csrss.exe
000005F0	DefWatch		C:\Program Files\Symantec AntiVirus\De
0000018C	EasyDVDt		C:\Program Files\EasyDVDtoVCD\EasyDVDt
000003BC	EasyDVDt		C:\Program Files\EasyDVDtoVCD\EasyDVDt
00000480	Explorer	Start Menu	C:\WINDOWS\Explorer.EXE
00000230	lsass		C:\WINDOWS\system32\lsass.exe
0000069C	SUTCFNT	UHookAPTServicellindow	C:\Program Files\MT0300\N\SUTCFNT.FXF

One-Attach it nha.Roi Olly it OK for fun, we stop here:

77F767CE	C3	RETN
77F767CF	CC	INT3
77F767D0	C3	RETN
77F767D1	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]
77F767D5	CC	INT3
77F767D6	C2 0400	RETN 4
77F767D9	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]
77F767DF	C3	RETN
77F767E0	57	PUSH EDI
77F767E1	8B7C24 0C	MOV EDI,DWORD PTR SS:[ESP+C]
77F767E5	8B5424 08	MOV EDX,DWORD PTR SS:[ESP+8]
77F767E9	C702 00000000	MOV DWORD PTR DS:[EDX],0
77F767EF	897A 04	MOV DWORD PTR DS:[EDX+4],EDI
77F767F2	0BFF	OR EDI,EDI
77F767F4	74 11	JE SHORT ntdll.77F76807

OEP II.Tim real and dump it:

-Since Attach after, you have to actually kī gradually, if not, you may have to close out the 2 Olly to do from the beginning that, first and then told, that have suffered àh wrong.

-Now press F9 to run it (although not found anything, notice the word Running right). Then, press F12 to stop it at:

004EC000	-EB FE	JMP SHORT EasyDVDt.<ModuleEntryPoint>
004EC002	0000	ADD BYTE PTR DS:[EAX],AL
004EC004	0000	ADD BYTE PTR DS:[EAX],AL
004EC006	5D	POP EBP
004EC007	50	PUSH EAX
004EC008	51	PUSH ECX

Click-to this line, choose:

	JMP SHORT EasyDVDt.<ModuleEntryPoint>	
	ADD BYT	Backup
	ADD BYT	
	POP EBP	Copy
	PUSH EA	
	PUSH EC	Binary
	BSWAP E	
	NOT EDX	Assemble
	PUSHFD	Space
	NOT EDX	
	BSWAP E	Label
	JMP SHO	
EB	MOV ECX	Comment
EB	POP ES	
	MOV ECX	Breakpoint
	OR CH,B	
	JMP SHO	Run trace
	PREFIX	
	JMP SHO	
	JMP SHO	
	PREFIX	
	JMP SHO	Follow
	STD	Enter
	JMP SHO	
	PREFIX	
	JMP SHO	Follow in Dump
	CIN	Selection

Looking down the small window shows:

Address	Hex dump
004EC000	EB FE 00 00 00 00 5D 50

One bowl-two bytes EB FE revised to 60 E8 as the old house. (Repeat with the other soft may be different this)

Address	Hex	dump
004EC000	60 E8	00 00 00 00 5D 5D

-Ready on the find:

004EC000	60	PUSHAD
004EC001	E8 00000000	CALL EasyDVDt.004EC006
004EC002	EB	POP EBP

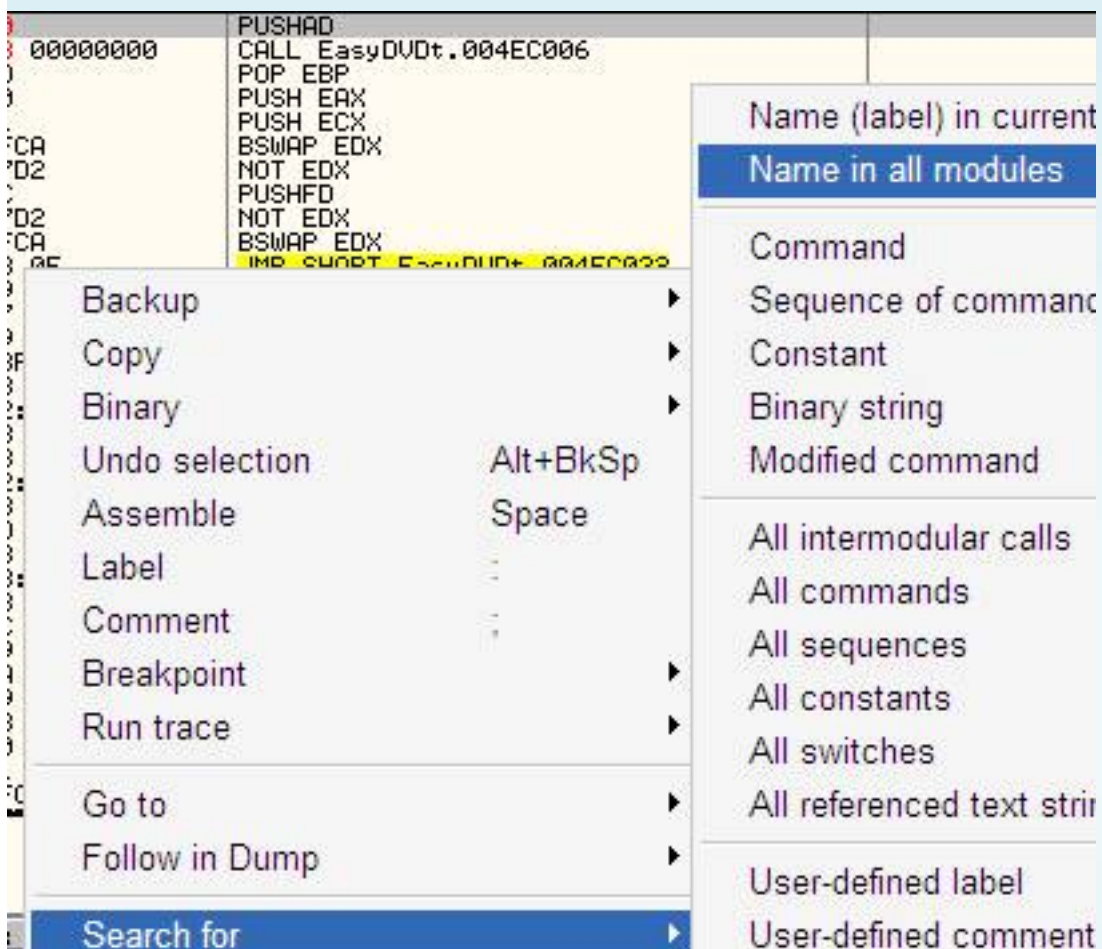
-Now is the time we made quite the same as part 1, this time no anti-BP and then again (please attach this month is no anti-dump as I said the part 1)

I-type to CmdBar following:

Command :

If time-Shift-F9 can to BP has been set but it will make more errors encountered day. Ma said then, if the wrong It is a step which must be done from the beginning of it, because thằng 2 process which we are doing is through xỏ from the 1st, it only exists at this time only, she must ensure that every child walking, do not quay anything ah.

-Now her children are remembered and 2 **OutputDebugStringA** em **IsDebuggerPresent** not? Usually when they start xoi Armadillo time, we must also treat children **OutputDebugStringA** this before. But do we do in the process, joker 1 ho households do not see anything I should not say, and important objects that we need to play the thằng 2 hú co. Co this is in on it ne. O I said then, if we just try to Debug it could always be close Olly is the love, the reason he is at 2 this Ministry. Now they treat each small
Right-click your mouse, select **Search for** ---- **Name in all modules** (like):



-It will appear a window correct? If the name intricate click to sort NAME lai.Roi pull down (or press repeatedly the word "O" and "U") for this child:

77E4B001	kernel32	.text	Export	OpenWaitableTimerA
77E92372	kernel32	.text	Export	OpenWaitableTimerW
77D6867B	USER32	.text	Export	OpenWindowStationA
77D6AD4E	USER32	.text	Export	OpenWindowStationW
77E949B7	kernel32	.text	Export	OutputDebugStringA
77DD1510	ADVAPI32	.text	Import	KERNEL32.OutputDebugStringW
77EAF0F0	kernel32	.text	Export	OutputDebugStringW
004FC26C	EasyDVDt	.data1	Import	USER32.PackDDFParam

Double-click on it-what, to this:

77E949B7	68 2C020000	PUSH 22C
77E949BC	68 8853E977	PUSH kernel32.77E95388
77E949C1	E8 1259FEFF	CALL kernel32.77E7A2D8
77E949C6	8365 FC 00	AND DWORD PTR SS:[EBP-4],0
77E949CA	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
77E949CD	8BC1	MOV EAX,ECX
77E949CF	8D70 01	LEA ESI,DWORD PTR DS:[EAX+1]
77E949D2	8A10	MOV DL,BYTE PTR DS:[EAX]
77E949D4	40	INC EAX
77E949D5	84D2	TEST DL,DL
77E949D7	75 F9	JNZ SHORT kernel32.77E949D2
77E949D9	2BC6	SUB EAX,ESI
77E949DB	40	INC EAX
77E949DC	8945 E0	MOV DWORD PTR SS:[EBP-20],EAX
77E949DF	894D E4	MOV DWORD PTR SS:[EBP-1C],ECX
77E949E2	8D45 E0	LEA EAX,DWORD PTR SS:[EBP-20]
77E949E5	50	PUSH EAX
77E949E6	6A 02	PUSH 2
77E949E8	6A 00	PUSH 0
77E949EA	68 06000140	PUSH 40010006
77E949EF	E8 43EEFDFF	CALL kernel32.RaiseException
77E949F4	834D FC FF	OR DWORD PTR SS:[EBP-4],FFFFFFFF
77E949F8	E8 A259FEFF	CALL kernel32.77E7A39F
77E949FD	C2 0400	RETN 4

Ham-usually begins as a push-value is finished and is **RETN 4**, or **RETN** (the SP2). So this refers to the Cuc do all the right places **PUSH 22C**, is a SpaceBar, edit **RETN** to **4** are:

RETN 4

☒ Fill with NOP's

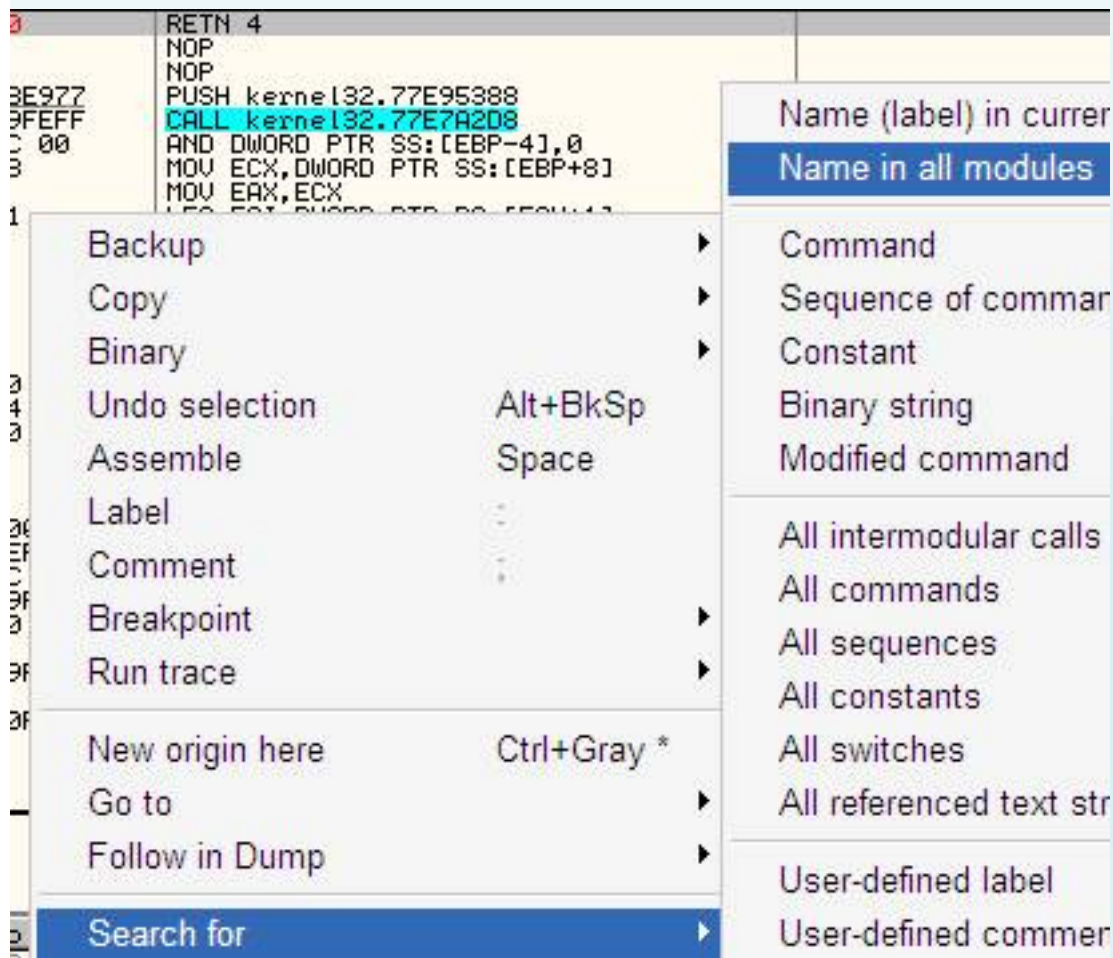
Assemble

Cancel

-Done are:

C2 0400	RETN 4
90	NOP
90	NOP
68 8853E977	PUSH kernel32.77E95388
E8 1259FEFF	CALL kernel32.77E7A2D8

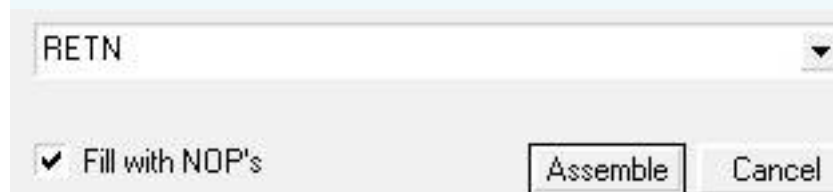
Now em-up to the other, must also click, ---- **Name Search for in all** modules:



-I find them **IsDebuggerPresent** right? Press "I" and "s" for fast and then drag it down a little is found àh (do not ask why they do not find this first order for the ABC is but I see them there should be more important xuc first):

77E65400	kernel32	.text	Export	IsDBCSLeadByteEx
7E091164	GDI32	.text	Import	KERNEL32.IsDBCSLeadByteEx
77E72740	kernel32	.text	Export	IsDebuggerPresent
77D60D16	USER32	.text	Export	IsDialogMessageA
77D4B130	USER32	.text	Export	IsDialogMessageW

How xu-chang italy children there, double click, and then head for the last variable function function:



-Is this:

77E72740	C3	RETN
77E72741	90	NOP
77E72742	90	NOP
77E72743	90	NOP
77E72744	90	NOP
77E72745	90	NOP
77E72746	8B40 30	MOV EAX,DWORD PTR DS:[EAX+30]
77E72749	0FB640 02	MOVZX EAX,BYTE PTR DS:[EAX+2]
77E7274D	C3	RETN

So-2 is completed em. That many out how you can submit the question of command, only contain the command RETN is, or at the top of function, to jump to the end function using JMP also.

Now click the button - **C** back to top khúc small. Press Shift-F9 to see what happens (quite long)? Also, it stopped at BP and, accordingly idea ... but then suspended machine. (Note that many here as you will see a message that the program failed What then, OK, then it does a window with a line loi. Ban click to select it to [resume](#) the program will stop at this BP):



Your program is suspended and can't run. Please resume main thread

OK

77FDE000	ERROR_MOD_NOT_FOU	Suspended	32 +	Actualize
				Resume

-We are stopped here right?:

77E7BE53	55	PUSH EBP
77E7BE54	8BEC	MOV EBP,ESP
77E7BE56	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
77E7BE59	FF75 18	PUSH DWORD PTR SS:[EBP+18]
77E7BE5C	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77E7BE5F	FF75 10	PUSH DWORD PTR SS:[EBP+10]
77E7BE62	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]
77E7BE65	FF75 08	PUSH DWORD PTR SS:[EBP+08]
77E7BE68	6A FF	PUSH -1
77E7BE6A	E8 30FEFFFF	CALL kernel32.CreateRemoteThread
77E7BE6F	5D	POP EBP

The day-tut same as part 1 and then to Italy under do. Nho taskbar to see the benefit to the present moment is not small, at this time, the computer I see it:



-That is it then, but is being blocked by BP should not run tiep. Roi we Disable the BP for it to di. Nhan button **B**. Friar BP is set right? Click to it, select Disable (temporarily disabled, a little more xài):

Always	PUSH EBP	Remove	Del
		Disable	Space

-Click **C** BP back seat in treating ly.No disabled and that

77E7BE53	55	PUSH EBP
77E7BE54	8BEC	MOV EBP,ESP
77E7BE56	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
77E7BE59	FF75 18	PUSH DWORD PTR SS:[EBP+18]
77E7BE5C	FF75 14	PUSH DWORD PTR SS:[EBP+14]

Yes it is the right Ctrl-F9 as the part 1 to the more deeply in order to find the first order CALL RETN I say that the part that 1.Tu Trace to find OEP.Nhung the program before running with springiness to a table, click OK or cry registered so we must for the table appear to da.Nhan F9, springiness small table:

Order Easy DVD to VCD Burner Now!

Get Rid of the 10-Percent Conversion Limit!

\$34.95



*Copy Your DVD to CD-R/RW
and Watch the Movie on TV!*

OK

Enter Key

Order Now!

Order Now! Unconditional 30-Day Money-Back Guarantee.

-Before you click OK, back Olly 2 was (now is the time to do that Olly 2 home). Stay in place at this time ha.Ta BP set to the BP di.Bam F2 it up:

77E7BE53	55	PUSH EBP
77E7BE54	8BEC	MOV EBP,ESP
77E7BE56	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
77E7BE59	FF75 18	PUSH DWORD PTR SS:[EBP+18]
77E7BE5C	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77E7BE5F	FF75 10	PUSH DWORD PTR SS:[EBP+10]
77E7BE62	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]
77E7BE65	FF75 08	PUSH DWORD PTR SS:[EBP+08]
77E7BE68	6A FF	PUSH -1
77E7BE6A	E8 30FEFFFF	CALL kernel32.CreateRemoteThread
77E7BE6F	5D	POP EBP
77E7BE70	C2 1800	RETN 18

Back-click OK the program, ah ... it stopped at BP and this home:

77E7BE53	55	PUSH EBP
77E7BE54	8BEC	MOV EBP,ESP
77E7BE56	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
77E7BE59	FF75 18	PUSH DWORD PTR SS:[EBP+18]
77E7BE5C	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77E7BE5F	FF75 10	PUSH DWORD PTR SS:[EBP+10]
77E7BE62	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
77E7BE65	FF75 08	PUSH DWORD PTR SS:[EBP+8]
77E7BE68	6A FF	PUSH -1
77E7BE6A	E8 30FEFFFF	CALL kernel32.CreateRemoteThread
77F7BF6F	5D	POP EBP

Delete BP-go here, then press Ctrl-F9 to order it to me RETN 18 below.

77E7BE53	55	PUSH EBP
77E7BE54	8BEC	MOV EBP,ESP
77E7BE56	FF75 1C	PUSH DWORD PTR SS:[EBP+1C]
77E7BE59	FF75 18	PUSH DWORD PTR SS:[EBP+18]
77E7BE5C	FF75 14	PUSH DWORD PTR SS:[EBP+14]
77E7BE5F	FF75 10	PUSH DWORD PTR SS:[EBP+10]
77E7BE62	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
77E7BE65	FF75 08	PUSH DWORD PTR SS:[EBP+8]
77E7BE68	6A FF	PUSH -1
77E7BE6A	E8 30FEFFFF	CALL kernel32.CreateRemoteThread
77E7BE6F	5D	POP EBP
77E7BE70	C2 1800	RETN 18

Then-F7 to it here:

00AFB3DE	5F	POP EDI
00AFB3DF	5E	POP ESI
00AFB3E0	C9	LEAVE
00AFB3E1	C3	RETN

Lai-Ctrl-F9 to RETN more.

00AFB3DE	5F	POP EDI
00AFB3DF	5E	POP ESI
00AFB3E0	C9	LEAVE
00AFB3E1	C3	RETN

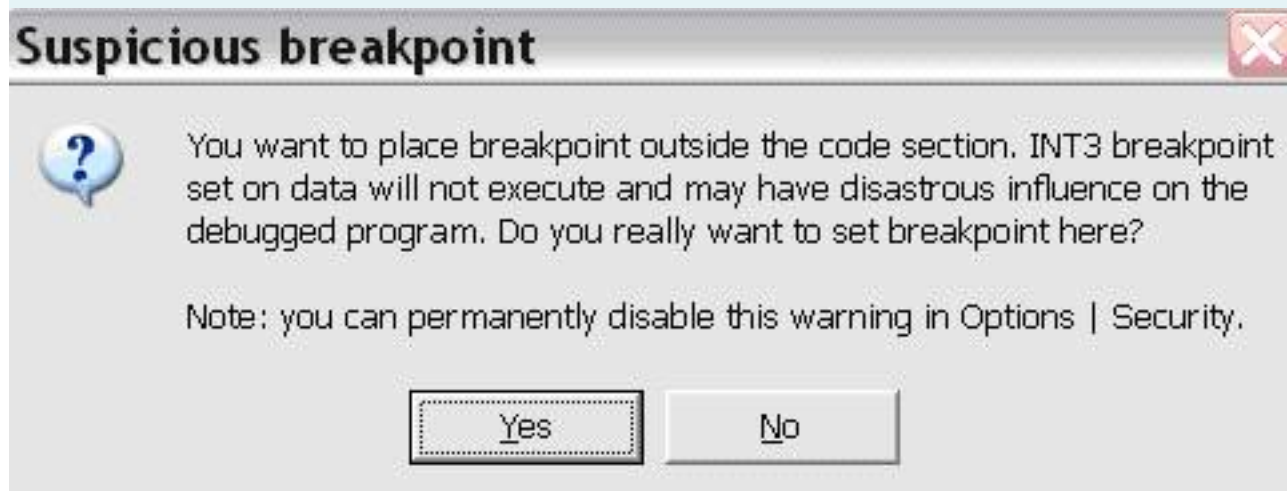
Then the-F7 again. Stop here:

00B0D825	59	POP ECX	kernel32.77E7BE2
00B0D826	BF D8C8B100	MOV EDI,0B1C8D8	
00B0D82B	8BCF	MOV ECX,EDI	
00B0D82D	E8 84A8FDFF	CALL 00AE82B6	
00B0D832	84C0	TEST AL,AL	
00B0D834	75 09	JNZ SHORT 00B0D83F	
00B0D836	6A 01	PUSH 1	
00B0D838	8BCF	MOV ECX,EDI	
00B0D83A	E8 54F8FDFF	CALL 00AED093	
00B0D83F	B9 40BBB100	MOV ECX,0B1BB40	
00B0D844	C705 7090B100 D	MOV DWORD PTR DS:[B19070],0B19DD8	ASCII "RC"
00B0D84E	E8 C174FEFF	CALL 00AF4D14	
00B0D853	6A 00	PUSH 0	
00B0D855	E8 BA74FEFF	CALL 00AF4D14	
00B0D85A	A1 20CFB100	MOV EAX,DWORD PTR DS:[B1CF20]	
00B0D85F	59	POP ECX	
00B0D860	8B15 38CFB100	MOV EDX,DWORD PTR DS:[B1CF38]	EasyDVDt.0040000

Hi-time ... to find answers in the order RETN then that. Drag down to:

00B0D8A9	2BCA	SUB ECX,EDX
00B0D8AB	FFD1	CALL ECX
00B0D8AD	8BD8	MOV EBX,EAX
00B0D8AF	5F	POP EDI
00B0D8B0	8BC3	MOV EAX,EBX
00B0D8B2	5E	POP ESI
00B0D8B3	5B	POP EBX
00B0D8B4	C3	RETN

- Found and then looked up a few lines, he'll see a CALL something above, (is **CALL ECX**). I set him at a BP **CALL EC X** this, it asked what the press YES:



00B0D34E	FFD1	CALL ECX
00B0D3AD	8BD8	MOV EBX,EAX
00B0D3AF	5F	POP EDI
00B0D3B0	8BC3	MOV EAX,EBX
00B0D3B2	5E	POP ESI
00B0D3B3	5B	POP EBX
00B0D3B4	C3	RETN

Click-Shift-F9 to stop it at this BP.

00B0D34E	FFD1	CALL ECX
00B0D3AD	8BD8	MOV EBX,EAX
00B0D3AF	5F	POP EDI
00B0D3B0	8BC3	MOV EAX,EBX
00B0D3B2	5E	POP ESI
00B0D3B3	5B	POP EBX
00B0D3B4	C3	RETN

Delete BP-go, then an F7 phat.Oaiiii OEP to actually program and then, tired ghê ...:

00401C5C	-FF25 FC104000	JMP DWORD PTR DS:[4010FC]
00401C62	-FF25 90114000	JMP DWORD PTR DS:[401190]
00401C68	68 88644000	PUSH EasyDVDt.00406488
00401C6D	E8 F0FFFFFF	CALL EasyDVDt.00401C62
00401C72	0000	ADD BYTE PTR DS:[EAX],AL
00401C74	50	PUSH EAX
00401C75	0000	ADD BYTE PTR DS:[EAX],AL
00401C77	0030	ADD BYTE PTR DS:[EAX],DH
00401C79	0000	ADD BYTE PTR DS:[EAX],AL
00401C7B	0048 00	ADD BYTE PTR DS:[EAX],CL
00401C7E	0000	ADD BYTE PTR DS:[EAX],AL
00401C80	0000	ADD BYTE PTR DS:[EAX],AL

-I called the OEP, but also aged Hec Cu Lec say the VB is something, say that quái .. Cuc explain anything. We just simply take small issues, but we are Newbie ... he he.Gio dump it then that she was a child, opening his "sovereign" to do PE, and then select the process is necessary xuc thẳng process with this small (remember the others that you will):

Then right-click no.chon **full dump** is finished ... (and Olly still leave the house)

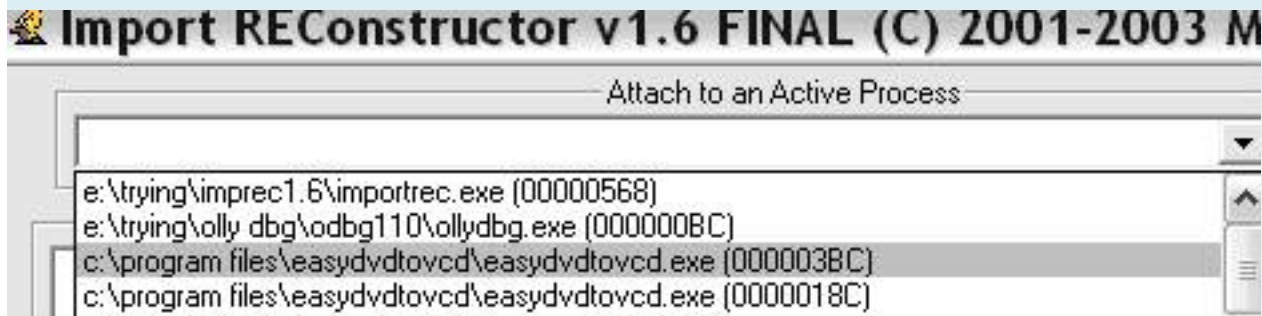
III.Sua IAT and the program (and rebuild Fix IAT):

-Now we connect with them on past treatment [ImportREC](#), we miss them the stone tut part 1 hour should apologize a child.

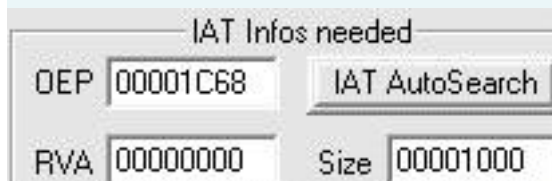
--- Why yesterday he xu such treatment? (ImportREC) ---

--- I must know him, it is because the job ... --- (he he nhái-Lam Truong Asia)

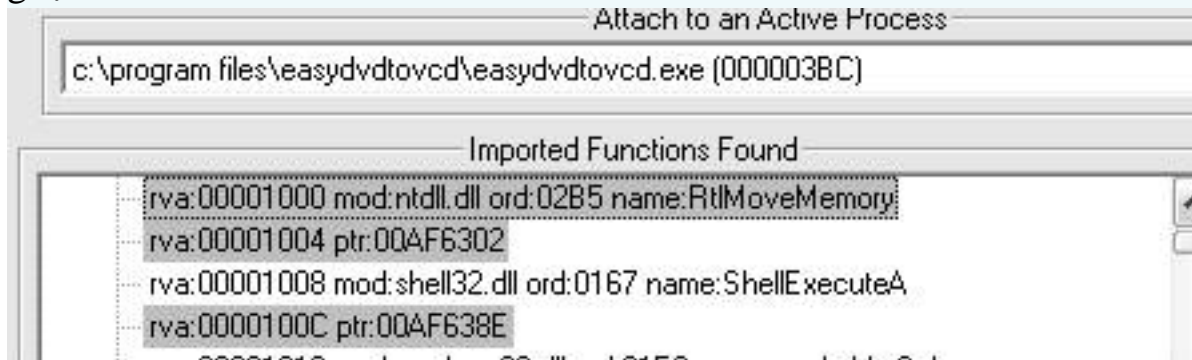
-So they are working toward and that process should choose xuc he is 2 to address small **03BC**:



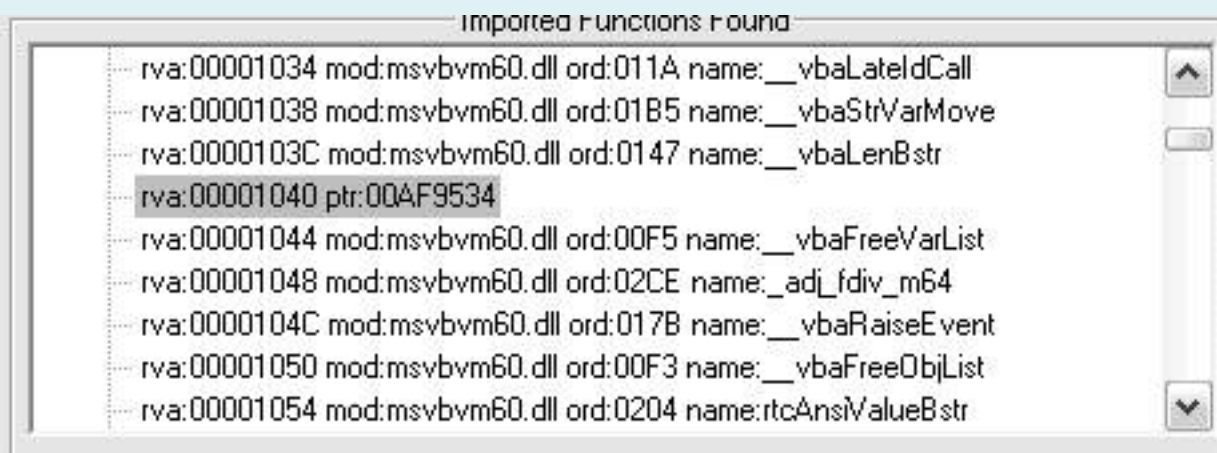
-Place OEP also fill the [formula](#), except as the 400,000 tut part 1. What is OEP 401C68 -400000 = 1C68:



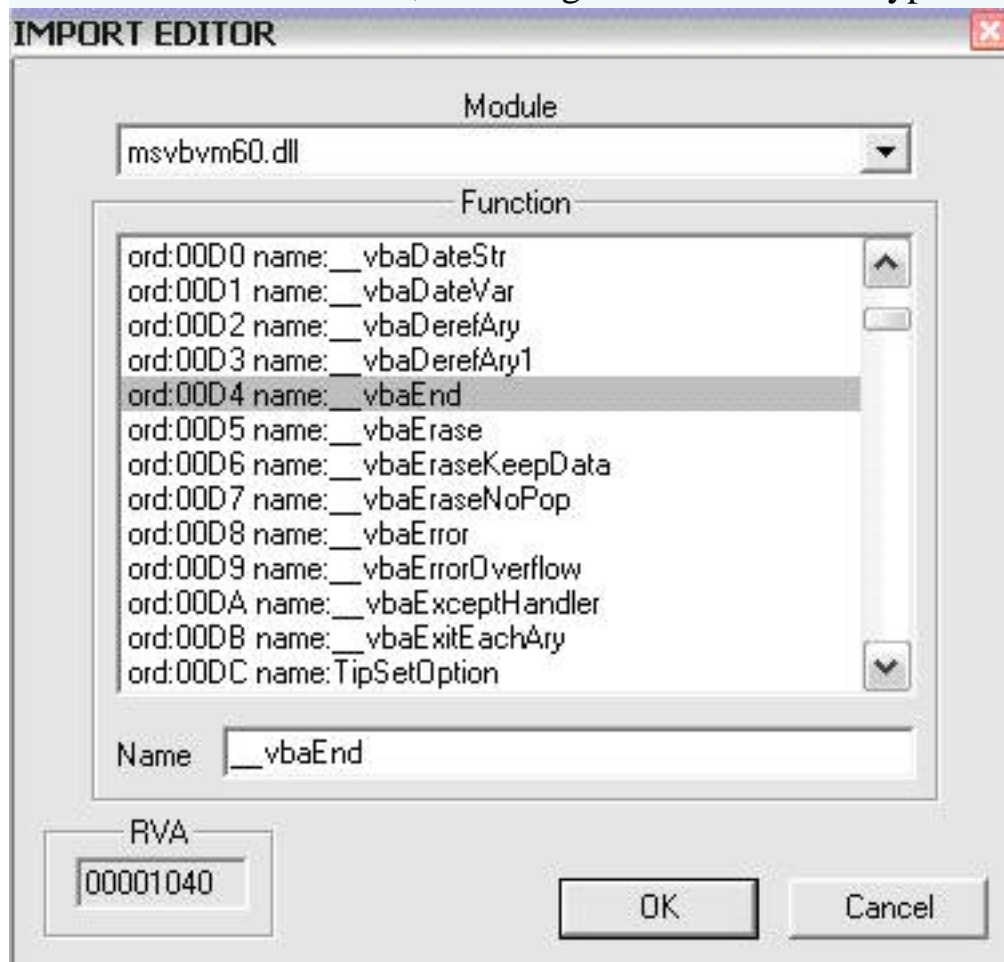
Then [click-IAT AutoSearch](#) ... Then click [Show Invalid](#) children to listen This report considered ghẹo who have more children?:



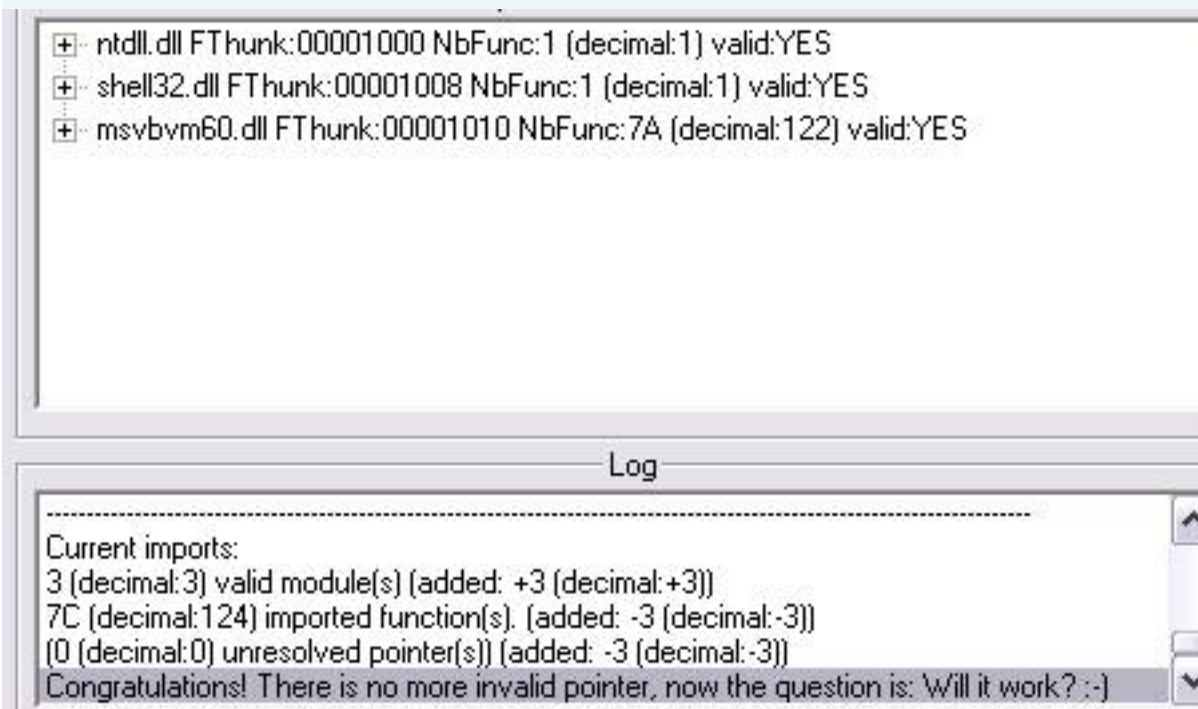
-A lot .. but the gang targeted thẳng to "headless". ... You pull down the search of this (pay attention to address it, and I aged Cu Lec this same place):



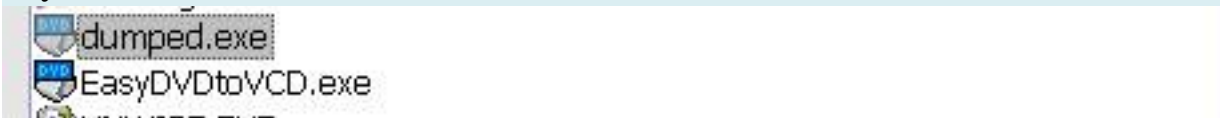
Double-click on him-this, then drag down to select or type in the image are:



Ok-a cai.Xong of life's ... ha ha .. but there dăm garbage ha, if they are available when you click bold **Show Invalid** Click the time we must then select **Cut .. Thunk (s)** Ác WA. . clump again be prescind Em ... This is one of the problems again ..:



-Fix the time ... and then to **dump** Quách ... Choose brother Orphaned **dumped.exe** we created the nay (end of Part II) ...

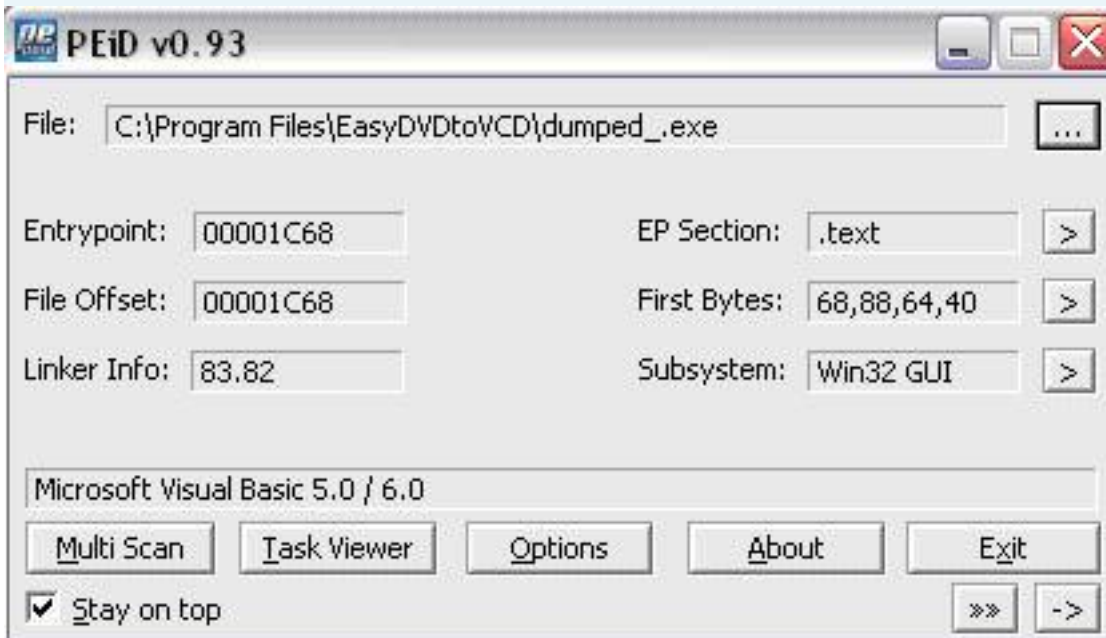


- OK ... save is successful, then it is ..

-Style bags Trickyboy the "old horse wen road ... they kick ImportREC .. (he is only banned bit hold the hearing children thét) ... The rest of the year Orphaned the fever for the garbage, should the new households use ... households:



Load-dumped_.exe to file **PEID** we see OKIE, code with **Microsoft Visual Basic 5.0/6.0**:



But do not jump-happy, but remember that the library [Armaccess.dll](#) that I have mentioned in the Tool does not need? Now you just RUN [dumped_.exe](#) file this view, will receive this message:



-So it must be more demanding file library under new [Armaccess.dll](#) run, so we just copy the file to the folder with them [thằng dumped_.exe](#) is completed.



-The media always say, some programs (including statements by Armadillo) after complete unpack the files have any claim under this new run, you are also considered as the heart smaller. And this file I do not know who out of it, I only Search only that the NET. (format as a Plugin)

-Now it has a problem problems ... File dump and fix this .. we have accidentally Crack it always zói (RUN always without registration springiness NAG) While the principle is still keeping unpack the Register Key functions of the people hix regret that knowledge shallow .. I just know this only to say ... (FFF group also see how that is the file's Crack FFF [Armaccess.dll](#) not need, but more in some places chẳng crack the pack to finish in 2:12 ASPack, and add more in the name to register it, while with this do not know who registered more ... hix hix)

Besides-ra.Tut also a mistake of this soft is also a file is [VCDBurner.exe](#) also pack in a style that Armadillo that if you want to use 100% soft then we have to unpack this thằng nhái more (father Cu Lec not, certainly not by the related topic is the Anti-BP)). A is a temporary period of gác always ... I continue to learn he was ... he ... households available information highway, he is using this code [Borland](#) Delphi, employers recognized this innovative soft U.S., the same program code, but with 2 stars loại.Tai hả know? considered the thought, I have in hand a complete and file Crack pack again with 2:12 ASPack FFF's in your hands again [Quick](#) has [unpack v0.7](#) -> Over!

So-Sincerely relatives gambling! Newbie that many say is too afraid to U.S., expecting her child, if not more what is viewed as entertainment tut ha.

-Thanks again to all of you in the [REA](#). Tut they have quite a bit thanks to the efforts of you ... hix hix ...

I. Introduction:

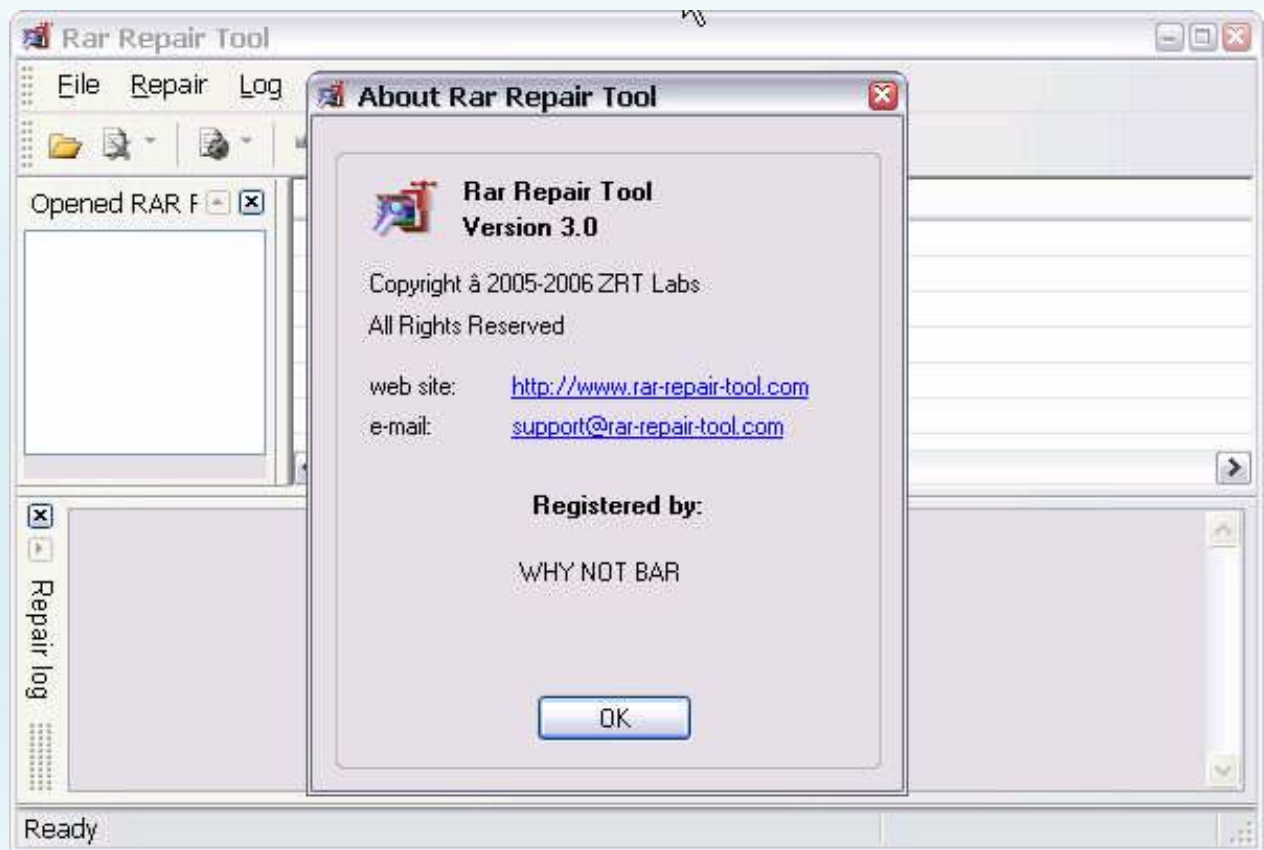
RAR Repair Tool 3.0 is a Soft or rather it is designed to fix damaged RAR and SFX archives. You try to limit File Repair under 5Mb Full also want to take the almost 50 USD to buy the quyen. Nhung an hour to bring the members Reaonline which offers 50 USD to purchase the copyright *'Nhuc Chi English Hung*. If Bro have the same opinion with our children is the same meat Target this, this is by Pack *EXE Cryptor v2.2.x*. According to the evaluation of individual children Packer this more difficult to swallow and *Asprotect Armadillo*.

II. Tools:

- Tool v AC A P l c u d g in need úng:

- *LY D L O B G _ E x c e r i t a l y p t o r 1 s t 1 0*
- *O E P f e d i n R V X . M e d i c i n e . Z b y d e r o o k*
- *L o r d P D e e e x l u*
- *B O D i g S c r 1 p t . 4 8*
- *I m p o r t R E C 1 . 6 F*
- *R G D P a c k e r e t D e c t o r a n d 0 . 6 . 4*
- *F C x F E p l o e r r V*

- S c r i p t : *"E x e c t r y p o r 2 . X I A T b u i r l e d e r"* by *K a G a n d t h e 1 s t 1*.
- T a r g e t : *R A R R E P T o o l a i r V 3 r d 0 C y r p o i g h t © 2 0 0 5 - 2 0 0 6 Z R T A B L s*.
- H o m e p a g e : [h t t p : / / w w w . R a r - e r a p i r - t o o m o m e n t u m o m /](http://www.Rar-erapir-too-momentum.com/)



III. Unpacking:

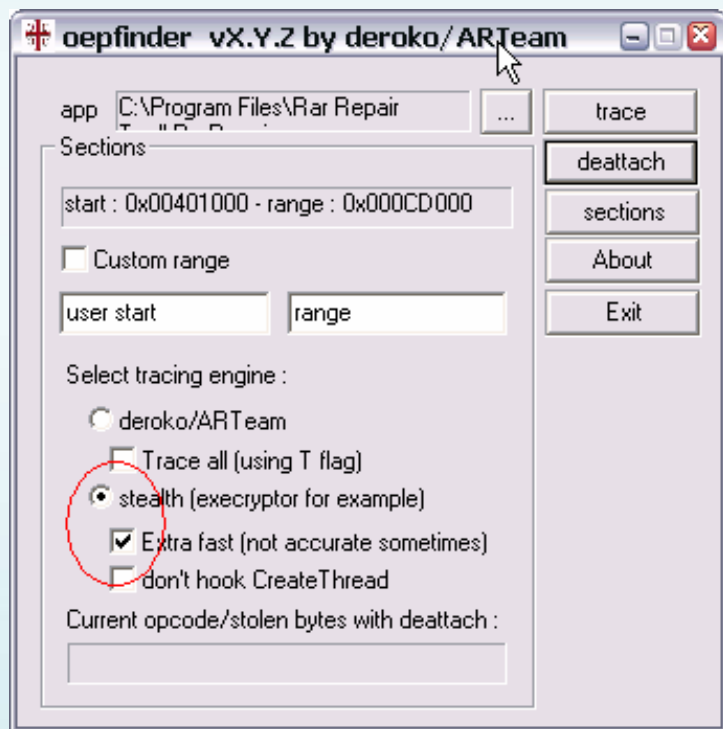
A job using familiar **RDG Packer Detector v0.6.4** Detect Target



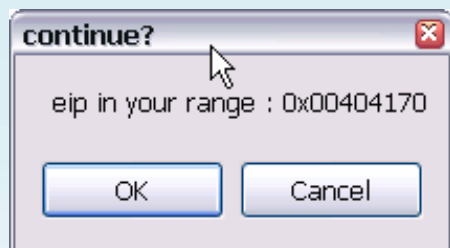
Packer see this as "every contract" gọi Anti debug ... It is in this khuông to identify you when you are on target Load OllyDBG you do not do the Run is But also on "Martial qit thickness varies with the Thai ... "and has appeared Patch for OllyDBG to bypass the AntiDBG this, it takes for people to have Patch and config it to create the OllyDBG For Download **EXECryptor** you from loss of use of search. When has the OllyDBG For **EXECryptor** and Tools necessary we

start Unpacking

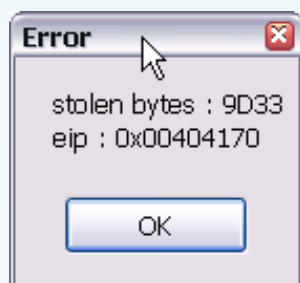
1st Run **OEPFinder vXYZ** deroko by selecting "RarRepair.exe"



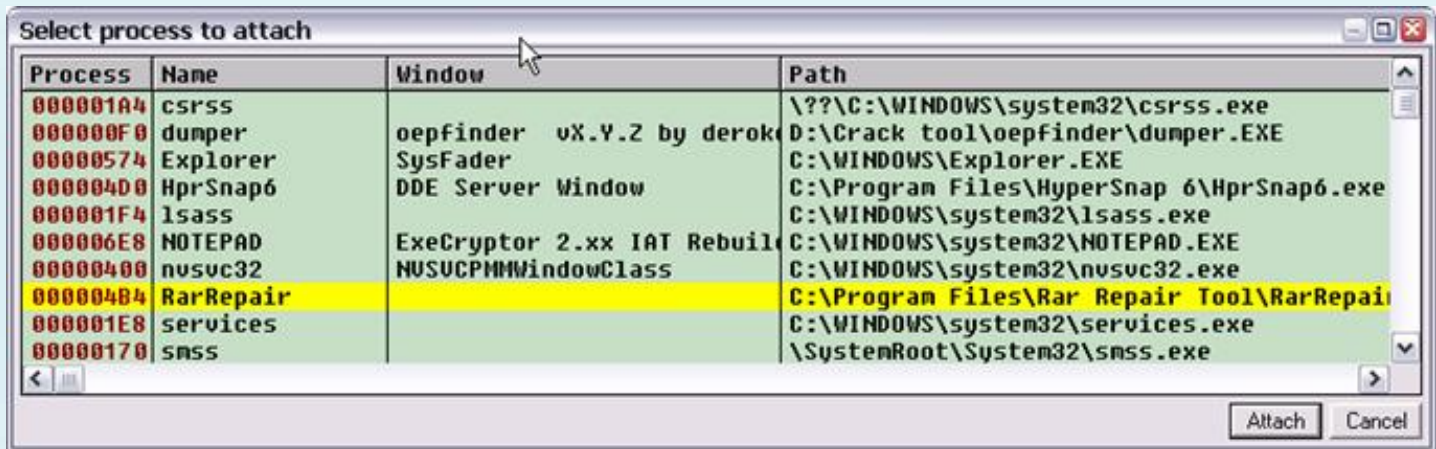
2nd Press **Trace** will appear this NAG



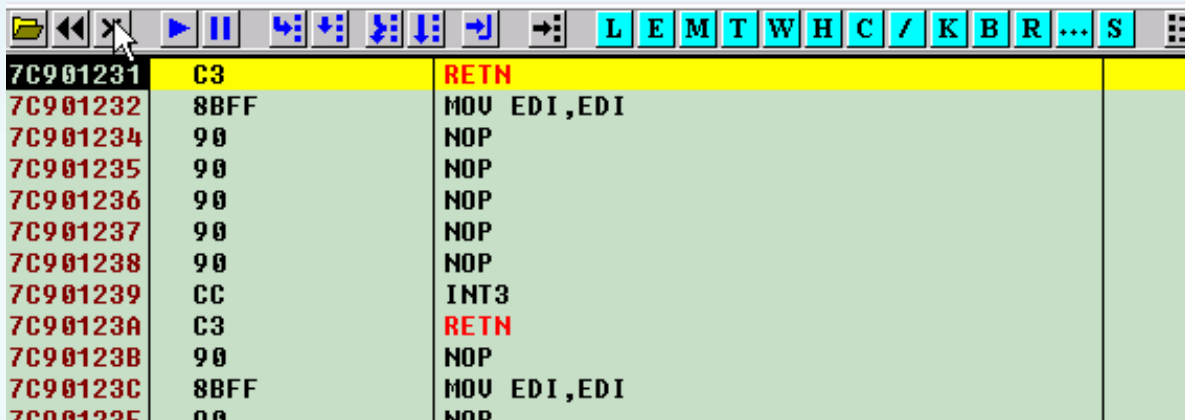
3. Click **Cancel**



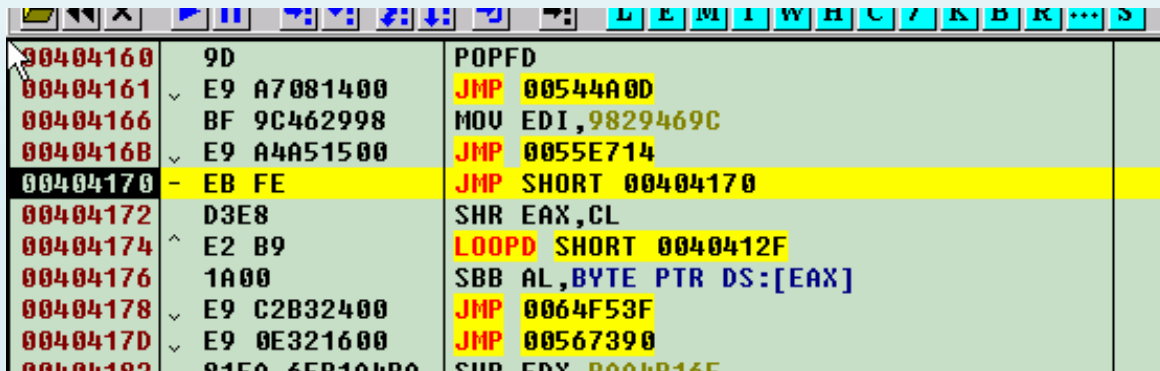
4. Remember the number, click **OK** and boot **OllyDBG_EXEcryptor** select the correct file and click **Attack**



5. You will stop here



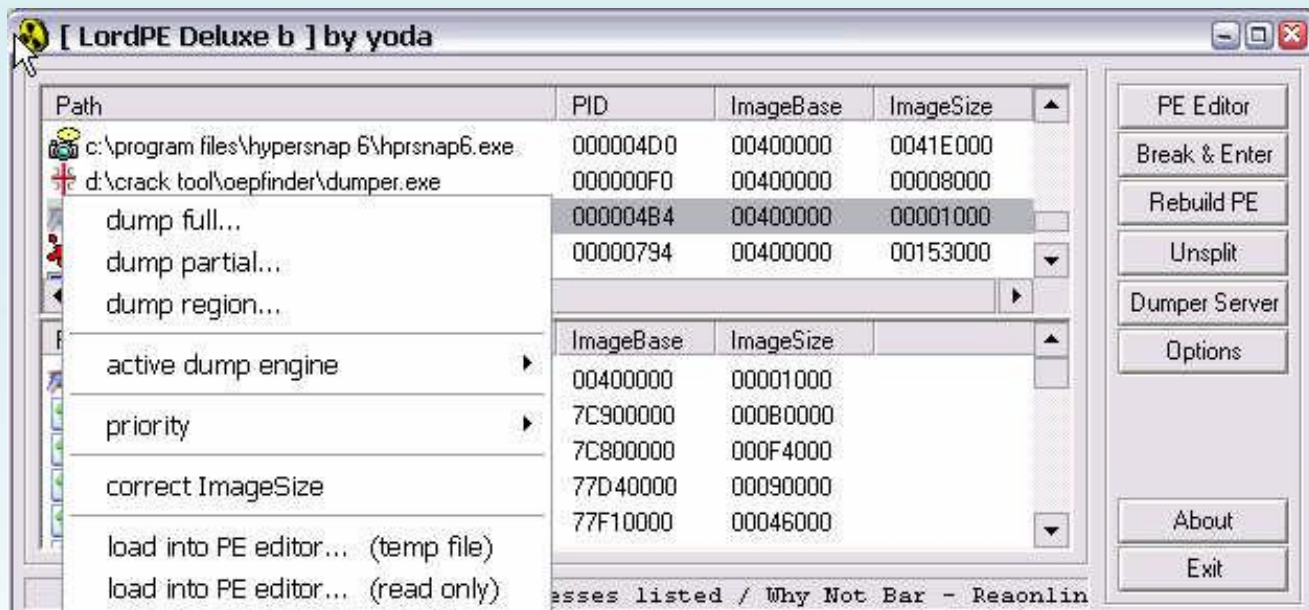
6th press **F9**, **F12** will stop here



7. Press **Ctrl + E** and the Patch

00404160	9D	POPFD
00404161	E9 A7081400	JMP 00544A0D
00404166	BF 9C462998	MOV EDI,9829469C
0040416B	E9 A4A51500	JMP 0055E714
00404170	9D	POPFD
00404171	33D3	XOR EDX,EBX
00404173	E8 E2B91A00	CALL 005AFB5A
00404178	E9 C2B32400	JMP 0064F53F
0040417D	E9 0E321600	JMP 00567390
00404182	81EA 6FB1A4BA	SUB EDX,BAA4B16F
00404188	81C2 9640034F	ADD EDX,4F034096
0040418E	E9 92321900	JMP 00597425
00404193	9D	POPFD

8. Press Shift + F9 Soft Run will complete ... This is can we say success is 20&percent;. Hehe
 thui dump file using **LordPE**. Note should do is close **OlllyDBG_EXEcryptor** working when they
 have not consider



9th after Full dump, we need to determine the position start and end of the IAT.

In **OlllyDBG_EXEcryptor** press **Ctrl + G** and enter 401000 and scroll down to 1 little mouse will see

0040104E	C706 A8E84C00	MOV DWORD PTR DS:[ESI],4CE8A8	
00401054	E8 D74B0B00	CALL 004B5C30	
00401059	E8 D24B0B00	CALL 004B5C30	
0040105E	8B40 0C	MOV EAX,DWORD PTR DS:[EAX+C]	
00401061	68 80000000	PUSH 80	
00401066	50	PUSH EAX	
00401067	FF15 ACE44C00	CALL NEAR DWORD PTR DS:[4CE4AC]	user32.LoadIconA
0040106D	8B4C24 08	MOV ECX,DWORD PTR SS:[ESP+8]	
00401071	8946 5C	MOV DWORD PTR DS:[ESI+5C],EAX	

10. And it is easy to identify the **IAT Start**

004CDFF4	00000000	
004CDFF8	00000000	
004CDFFC	00000000	
004CE000	77DD6BF0	ADVAPI32.RegCloseKey
004CE004	005C4B41	ASCII "hp=\"
004CE008	005EE7D4	
004CE00C	0055F06E	
004CE010	0052399F	
004CE014	77DFC123	ADVAPI32.RegDeleteKeyA
004CE018	0055FFA9	
004CE01C	00634E6F	
004CE020	00524032	
004CE024	0063A745	
004CE028	0058A64A	
004CE02C	00000000	
004CE030	773E55D0	COMCTL32.ImageList_GetIconSize

IAT start

11. Scroll down we determine IAT End

004CE7FC	77578712	ole32.OleCreateMenuDescriptor
004CE800	00000000	
004CE804	10001BC0	unrar.RARProcessFile
004CE808	10001480	unrar.RARReadHeader
004CE80C	10001C00	unrar.RARSetCallback
004CE810	10001380	unrar.RARCloseArchive
004CE814	10001C60	unrar.RARSetKeepBrokenSwitch
004CE818	10001C40	unrar.RARSetPassword
004CE81C	10001070	unrar.RAROpenArchiveEx
004CE820	10001C80	unrar.RARSetAllYesSwitch
004CE824	00000000	
004CE828	004D0658	

IAT End

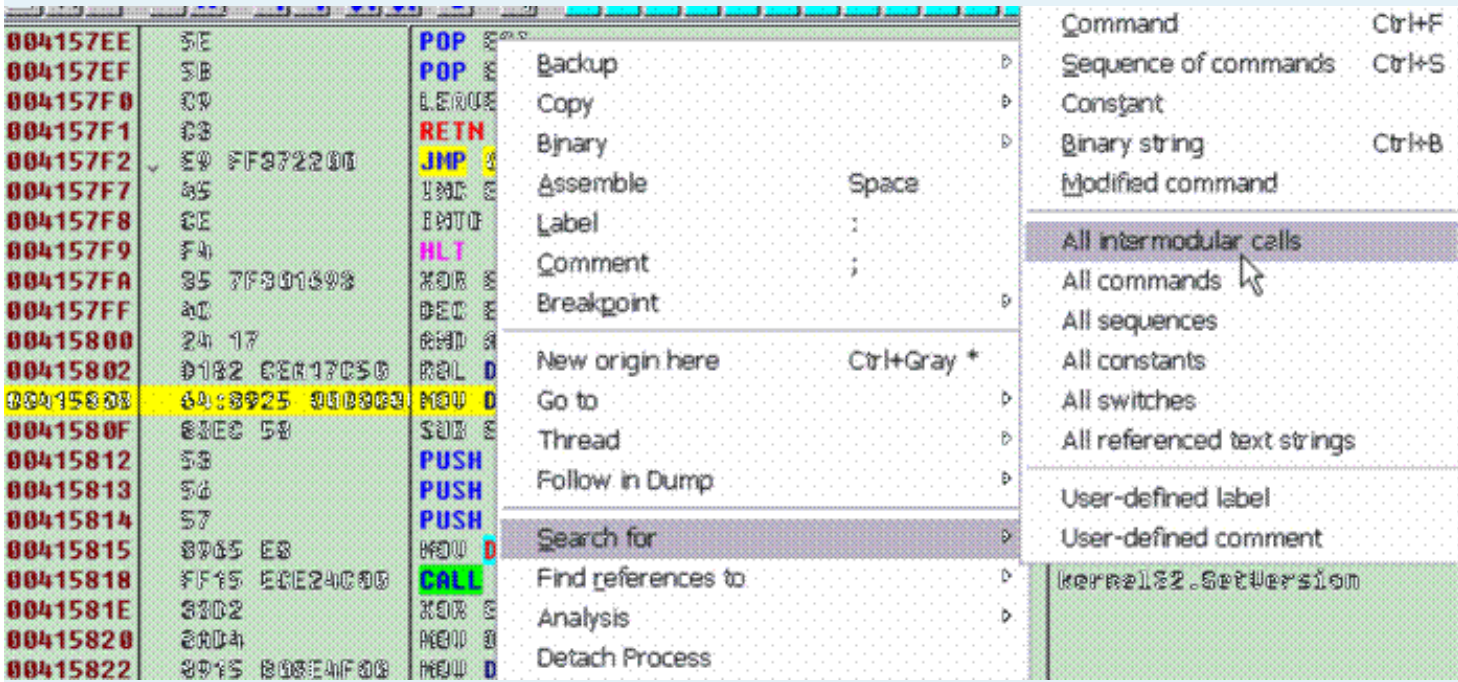
12. Currently, we have 1 File Dumped by **LordPE** and determine the **IAT**, to determine the position start and end of the IAT is very important because the number will be used when using *Script*. The remember that you should address when may we stopped in **OllyDBG_EXEcryptor** not **OEP's** Soft so work is we must find the original **OEP**. To find the original OEP many hours looking for the basic but at least we must know **Soft** Code is what language? But currently we do not know exactly be? For we are using **RDG Packer Detector v0.6.4** Scan File **Dumped.exe** and will receive the following information:



13. So we know the Code with Soft *Visual C++ 6.0*. Now, we borrow or 1 Soft Crackme that you have the Code in *Visual C++ 6.0* Load **OlllyDBG** to see the Code of **EntryPoint**. This is necessary because in addition to help us find the OEP in Target with our pack, but also useful for Fix Stolen Bytes in steps. Here they borrow **UltraEdit-32**. Load **UltraEdit-32** to 1 OlllyDBG to observe and we'll see.

00712600	55	PUSH EBP	
00712604	8BEC	MOV EBP,ESP	
00712606	6A FF	PUSH -1	
00712608	68 08627600	PUSH 00712608.00708268	
0071260D	68 70717100	PUSH 0071260D.00717170	SE handler installation
00712612	6A 01 010000	MOV EAX,DWORD PTR FS:[0]	
00712618	50	PUSH EAX	
00712619	6A 0925 0000	MOV DWORD PTR FS:[0],ESP	
00712620	83EC 58	SUB ESP,58	
00712623	53	PUSH EBX	
00712624	56	PUSH ESI	
00712625	57	PUSH EDI	
00712626	8B65 00	MOV [LOCAL.6],ESP	
00712629	FF15 00F57000	CALL NEAR DWORD PTR DS:[<&KERNEL32.GetVersion	kernel32.GetVersion
0071262F	8BD2	MOV EDX,EDX	
00712634	8BDC	MOV EAX,EDX	

Soft duoc code bang
Visual C++ 6.0



14. You easily notice Soft code in *Visual C + + 6.0* under OEP function is always the API "**GetVersion**" based on this we find to OEP for our target. Move over **OllyDBG_EXEcrypto** working with **RarRepair.exe** "press **F10** and select the image

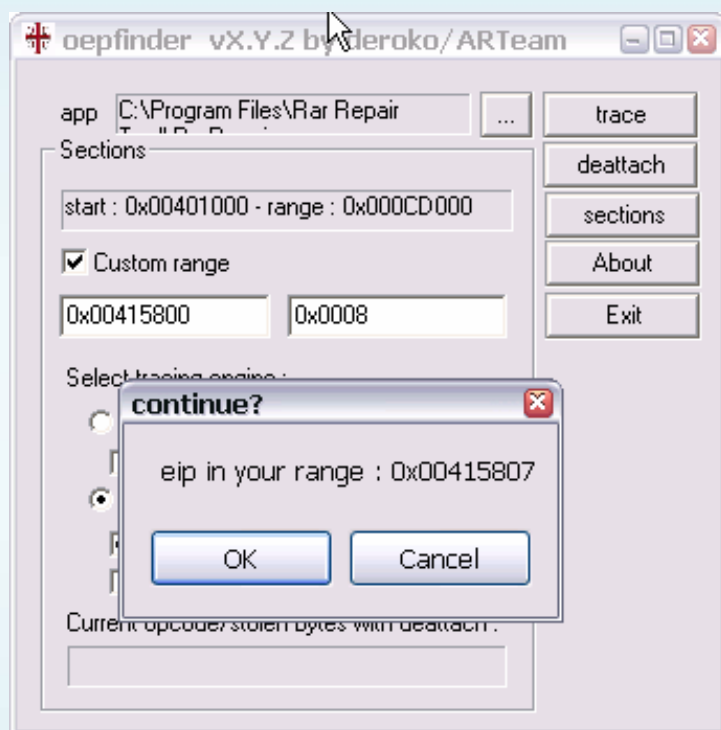
15th Search function "**GetVersion**" and we have nhusau:

00410C60	CALL NEAR DWORD PTR DS:[4CE284]	kernel32.GetTimeZoneInformation
00410E5F	CALL NEAR DWORD PTR DS:[4CE284]	kernel32.GetTimeZoneInformation
00410E5F	CALL NEAR DWORD PTR DS:[4CE528]	user32.GetTopWindow
00410E5F	CALL NEAR DWORD PTR DS:[4CE528]	user32.GetTopWindow
00410E5F	CALL NEAR DWORD PTR DS:[4CE2EC]	kernel32.GetVersion
00415818	CALL NEAR DWORD PTR DS:[4CE2EC]	kernel32.GetVersion
0041782E	CALL NEAR DWORD PTR DS:[4CE2EC]	kernel32.GetVersion
004190F2	CALL NEAR DWORD PTR DS:[4CE2EC]	kernel32.GetVersion
004190F2	CALL NEAR DWORD PTR DS:[4CE2EC]	kernel32.GetVersion
004190F2	CALL NEAR DWORD PTR DS:[4CE2EC]	kernel32.GetVersion
004190F2	CALL NEAR DWORD PTR DS:[4CE2EC]	kernel32.GetVersion
004190F2	CALL NEAR DWORD PTR DS:[4CE1E8]	kernel32.GetVersionEx0
004190F2	CALL NEAR DWORD PTR DS:[4CE1E8]	kernel32.GetVersionEx0

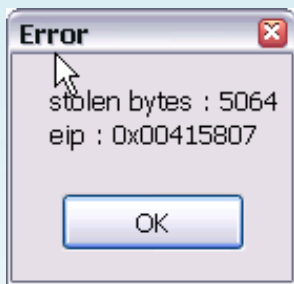
04157EF	58	POP EAX	
04157F0	07	LEAVE	
04157F1	03	RETN	
04157F2	E9 FF372200	JMP 00680FF0	
04157F7	45	INC EBP	
04157F8	0E	INT0	
04157F9	F4	HLT	Privileged command
04157FA	85 7F8D1600	MOV EAX,9316807F	
04157FF	4C	DEC ESP	
0415800	24 17	MOV AL,17	
0415802	D182 0EA17C50	ROL DWORD PTR DS:[EDX+507CA1CE],1	
0415808	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
041580F	83EC 58	SUB ESP,58	
0415812	58	PUSH EAX	
0415813	56	PUSH ESI	
0415814	57	PUSH EDI	
0415815	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0415818	FF15 ECE24C00	CALL NEAR DWORD PTR DS:[4CE2EC]	kernel32.GetVersion
041581E	3302	XOR EDX,EDX	
0415820	80B4	MOV BL,00	
0415822	8915 B88E4F00	MOV DWORD PTR DS:[4F8EB0],EDX	
0415828	8BC8	MOV ECX,EAX	
041582A	81E1 FF000000	AND ECX,0FF	
0415830	8900 8F8E4F00	MOV DWORD PTR DS:[4F8EAC],ECX	

16. Double Click to function "GetVersion" 2 is here to

17. Dám code yellow to contain the **OEP** has been *Stolen bytes*. We'll Fix it. Again using **OEPfinder** the following options (*remember OllyDBG_EXEcrypto close and OEPfinder ago*)



18. Click **Cancel**



19. Remember this information and click **OK**, **OllyDBG_EXEcrypto** open, select the PID, click **Attach**, **F9**, **F12** to us here:

00415807	- EB FE	JMP SHORT 00415807
00415809	8925 00000000	MOV DWORD PTR DS:[0],ESP
0041580F	83EC 58	SUB ESP,58
00415812	53	PUSH EBX
00415813	56	PUSH ESI
00415814	57	PUSH EDI
00415815	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP
00415818	FF15 ECE24C00	CALL NEAR DWORD PTR DS:[4CE2EC]
0041581E	33D2	XOR EDX,EDX
00415820	8AD4	MOV DL,AH
00415822	8915 B08E4F00	MOV DWORD PTR DS:[4F8EB0],EDX
00415828	8BC8	MOV ECX,EAX
0041582A	81E1 FF000000	AND ECX,0FF
00415830	890D AC8E4F00	MOV DWORD PTR DS:[4F8EAC],ECX
00415836	C1E1 08	SHL ECX,8
00415839	03CA	ADD ECX,EDX
0041583B	890D A88E4F00	MOV DWORD PTR DS:[4F8EA8],ECX
00415841	C1E8 10	SHR EAX,10

20th press **Ctrl + E** and the nhusau Patch:

00415807	50	PUSH EAX
00415808	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
0041580F	83EC 58	SUB ESP,58
00415812	53	PUSH EBX
00415813	56	PUSH ESI
00415814	57	PUSH EDI
00415815	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP
00415818	FF15 ECE24C00	CALL NEAR DWORD PTR DS:[4CE2EC]
0041581E	33D2	XOR EDX,EDX
00415820	8AD4	MOV DL,AH
00415822	8915 B08E4F00	MOV DWORD PTR DS:[4F8EB0],EDX
00415828	8BC8	MOV ECX,EAX
0041582A	81E1 FF000000	AND ECX,0FF
00415830	890D AC8E4F00	MOV DWORD PTR DS:[4F8EAC],ECX
00415836	C1E1 08	SHL ECX,8
00415839	03CA	ADD ECX,EDX

21. Now it was time to Fix IAT, you can manually Fix Scirpt but will use more quickly. But before the Run Scirpt we need to know when some 1 run for thoroughly to

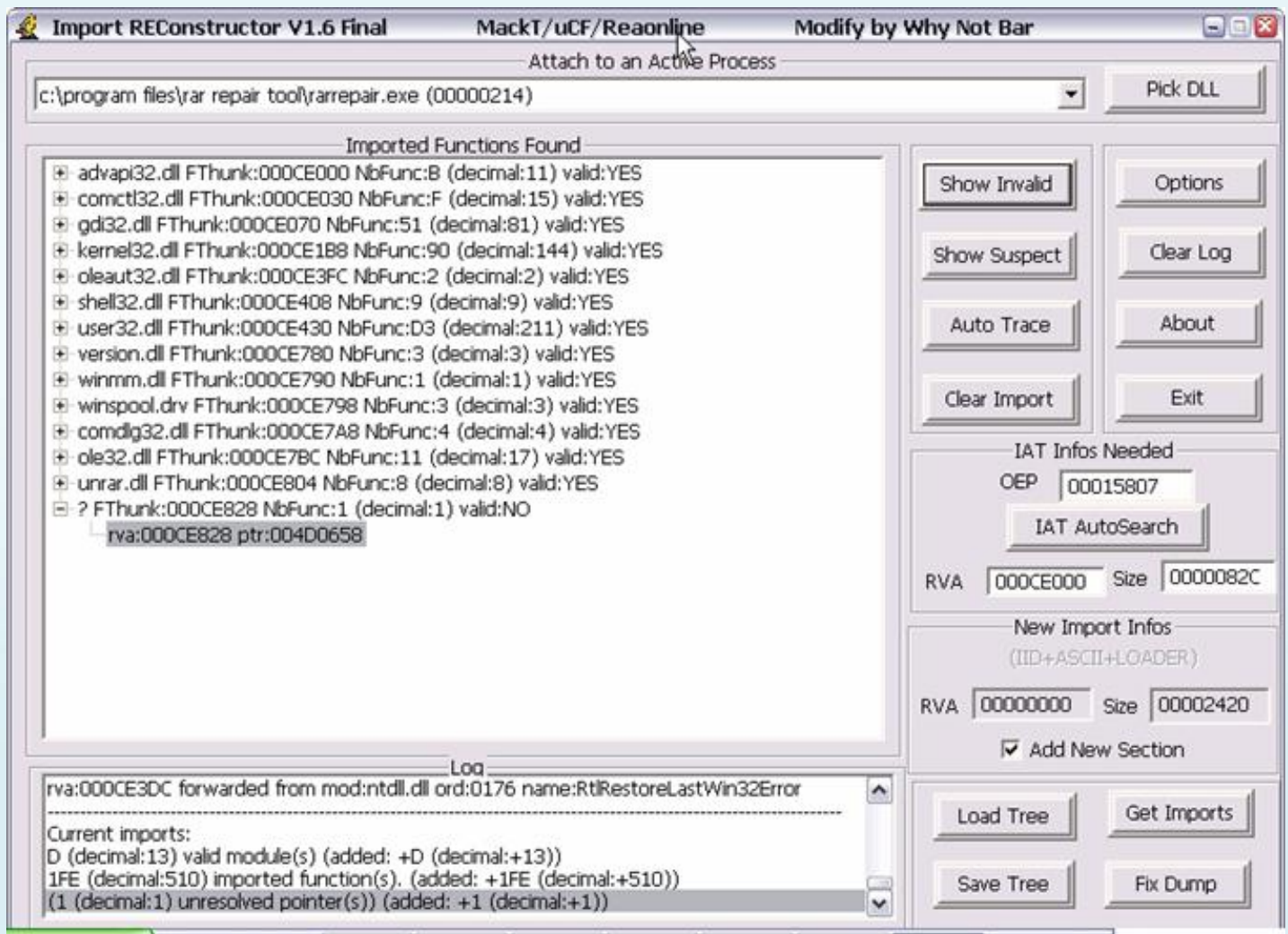
Imov AT start, 004CE000 // AT start

one because of a Ted, 004CE824 // In d A TE

22. Thugian Sitting waiting Scirpt little run we will have completed the IAT is Full

004CDFFC	00000000	
004CE000	77DD6BF0	ADVAPI32.RegCloseKey
004CE004	77DD6AF4	ADVAPI32.RegCreateKeyExA
004CE008	77DD7883	ADVAPI32.RegQueryValueExA
004CE00C	77DD761B	ADVAPI32.RegOpenKeyExA
004CE010	77DDEDE5	ADVAPI32.RegDeleteValueA
004CE014	77DFC123	ADVAPI32.RegDeleteKeyA
004CE018	77DDEBE7	ADVAPI32.RegSetValueExA
004CE01C	77DECF4A	ADVAPI32.RegEnumValueA
004CE020	77DFC8C1	ADVAPI32.RegEnumKeyExA
004CE024	77DFC1B5	ADVAPI32.RegQueryInfoKeyA
004CE028	77DFCAC3	ADVAPI32.RegEnumKeyA
004CE02C	00000000	
004CE030	773E55D0	COMCTL32.ImageList_GetIconSize
004CE034	773E92D5	COMCTL32.ImageList_Create
004CE038	773E5084	COMCTL32.ImageList_Destroy
004CE03C	773E5537	COMCTL32.ImageList_GetIcon
004CE040	773D582A	COMCTL32._TrackMouseEvent
004CE044	773E53CD	COMCTL32.ImageList_Draw
004CE048	773D405C	COMCTL32.InitCommonControls
004CE04C	773E518D	COMCTL32.ImageList_ReplaceIcon
004CE050	773E50C0	COMCTL32.ImageList_GetImageCount
004CE054	773E5335	COMCTL32.ImageList_DrawEx
004CE058	773E5660	COMCTL32.ImageList_GetImageInfo
004CE05C	773DCFDA	COMCTL32.PropertySheetA
004CE060	773D7B54	COMCTL32.DestroyPropertySheetPage
004CE064	773D7EED	COMCTL32.CreatePropertySheetPageA
004CE068	773E52EE	COMCTL32.ImageList_AddMasked

23. To ensure that we eat more **Dumped Full** and open again **ImportREC** enter parameters



24. 1 Invalid function should be **thunk Cut (s)**, then **dump Fix** nhubinh often, if you try to Run the File **Dumped_.exe** will run ko
 very simple because we have not Fix Stolen Bytes. Now we start Recovery **OEP**. Load File **Dumped_.EXE** to OllyDBG

004157F1	03	RETN	
004157F2	90	NOP	
004157F3	90	NOP	
004157F4	90	NOP	
004157F5	90	NOP	
004157F6	90	NOP	
004157F7	90	NOP	
004157F8	90	NOP	
004157F9	90	NOP	
004157FA	90	NOP	
004157FB	90	NOP	
004157FC	90	NOP	
004157FD	90	NOP	
004157FE	90	NOP	
004157FF	90	NOP	
00415800	90	NOP	
00415801	90	NOP	
00415802	90	NOP	
00415803	90	NOP	
00415804	90	NOP	
00415805	90	NOP	
00415806	90	NOP	
00415807	50	PUSH EAX	
00415808	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
0041580F	83EC 58	SUB ESP,58	
00415812	58	PUSH EBX	
00415813	56	PUSH ESI	

25. Đám code from 00415807 to 00415F72 queen code we need to Fix Stolen Bytes for valid. Back to the 13 you remember EP Soft standard code when using *Visual C++ 6.0*

007125FF	5E	POP ESI	
00712600	5B	POP EBX	
00712601	EB	LEAVE	
00712602	EB	RETN	
00712603	55	PUSH ESP	
00712604	8BEC	MOV EBX,ESP	
00712606	6A FF	PUSH -1	
00712608	68 00000000	PUSH 0	
00712609	68 70717100	PUSH 00717170	SE handler insta
00712612	64:01 00000000	MOV EAX,DWORD PTR FS:[0]	
00712618	50	PUSH EAX	
00712619	64:0025 00000000	MOV DWORD PTR FS:[0],ESP	
00712620	8BEC 58	SUB ESP,58	
00712623	53	PUSH EBX	
00712624	56	PUSH ESI	
00712625	57	PUSH EDI	
00712626	8B65 EB	MOV [LOCAL.0],ESP	
00712629	FF15 00000000	CALL NEAR DWORD PTR DS:[<&KERNEL32.GetVersion>]	kernel32.GetVersion
0071262F	8B62	MOV EBX,EDX	
00712631	8B62	MOV EBX,EDX	

2 em này chúng ta cần tìm

26. At issue is how we also are looking for exactly 2 address them as marked on the sides to fill in for valid. This ko too difficult and takes only 1 thui little ... you need to do from step 1 to step 8 for Soft Run completely and observe the Stack you will see

0012FFA4	FFFFFFFF	
0012FFA8	0012FF4C	
0012FFAC	0055FD9A	
0012FFB0	0012FFE0	Pointer to next SEH record
0012FFB4	00419BF0	SE handler
0012FFB8	004D34A8	
0012FFBC	00000000	
0012FFC0	0012FFF0	
0012FFC4	7C816D4F	RETURN to kernel32.7C816D4F
0012FFC8	00000000	
0012FFCC	0000D347	
0012FFD0	7FFDF000	
0012FFD4	8054B038	
0012FFD8	0012FFC8	
0012FFDC	81967CA0	
0012FFE0	FFFFFFFF	End of SEH chain
0012FFE4	7C8399F3	SE handler
0012FFE8	7C816D58	kernel32.7C816D58
0012FFEC	00000000	
0012FFF0	00000000	
0012FFF4	00000000	
0012FFF8	007FFB91	<ModuleEntryPoint>
0012FFFC	00000000	

27. Now we conduct nhusau Patch:

004157EE	5C	POP ESI
004157EF	5B	POP EBX
004157F0	C0	LEAVE
004157F1	C8	RETN
004157F2	E9 FF372200	JMP Dropped_00438FF6
004157F7	45	INC ESP
004157F8	DE	INT3
004157F9	F4	HLT
004157FA	85 7F801698	XOR EAX,9316807F
004157FF	4C	DEC ESP
00415800	24 17	AND AL,17
00415802	D182 0EA17C50	ROL DWORD PTR DS:[EDX+507CA1CE],1
00415808	64:8925 00000000	MOV DWORD PTR FS:[0],ESP
0041580F	88EC 58	SUB ESP,58
00415812	58	PUSH EBX
00415813	56	PUSH ESI
00415814	57	PUSH EDI
00415815	8965 EB	MOV DWORD PTR SS:[EBP-18],ESP
00415818	FF15 ECE24C00	CALL NEAR DWORD PTR DS:[<&kernel32.GetVersion>]
0041581E	8B02	MOV EBX,EBX
00415820	8AD4	MOV DL,AD
00415822	8B15 00000000	MOV DWORD PTR DS:[4F8EB0],EBX

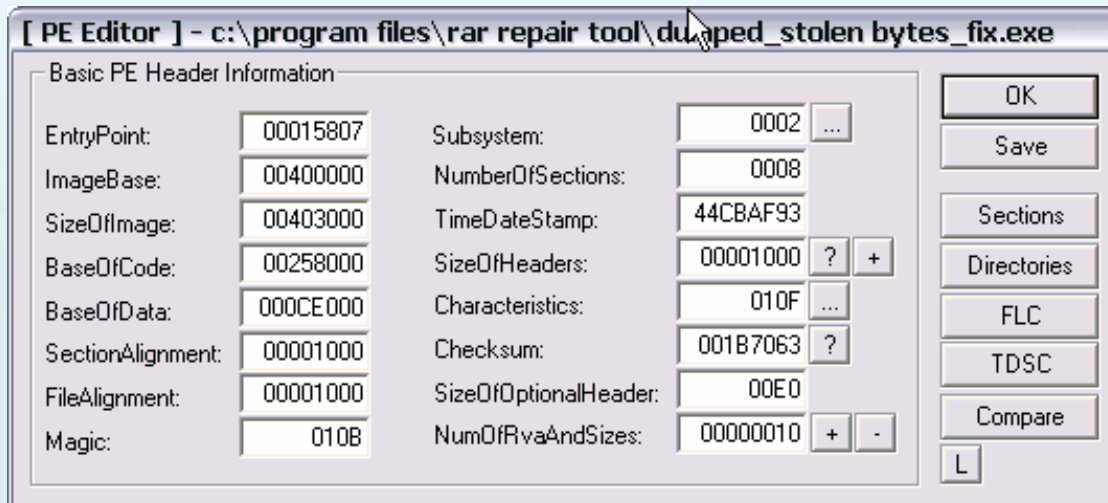
Truoc khi Fix Stolen bytes

004157EF	58	POP EBX	
004157F0	07	LEAVE	
004157F1	03	RETN	
004157F2	55	PUSH EBP	
004157F3	8BEC	MOV EBP,ESP	
004157F5	6A FF	PUSH -1	
004157F7	68 A8344D00	PUSH Dumped_004D34A8	
004157FC	68 F09B4100	PUSH Dumped_00419BF0	
00415801	64A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00415807	50	PUSH EAX	
00415808	6408725 00000000	MOV DWORD PTR FS:[0],ESP	
0041580F	83EC 58	SUB ESP,58	
00415812	58	PUSH EBX	
00415813	56	PUSH ESI	
00415814	57	PUSH EDI	
00415815	8B45 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00415818	FF15 ECE24C00	CALL NEAR DWORD PTR DS:[<&kerne132.GetVersion]	kerne132-GetVersio
0041581E	8B45 E8	MOV DWORD PTR SS:[EBP-18],ESP	

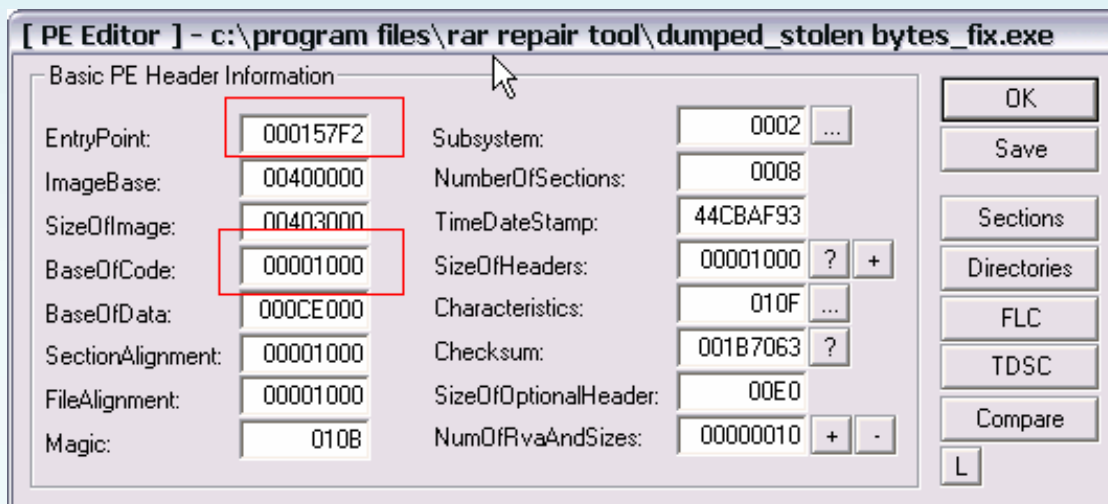
Sau khi Fix
Stolen Bytes

2 gia tri nay
trong Stack

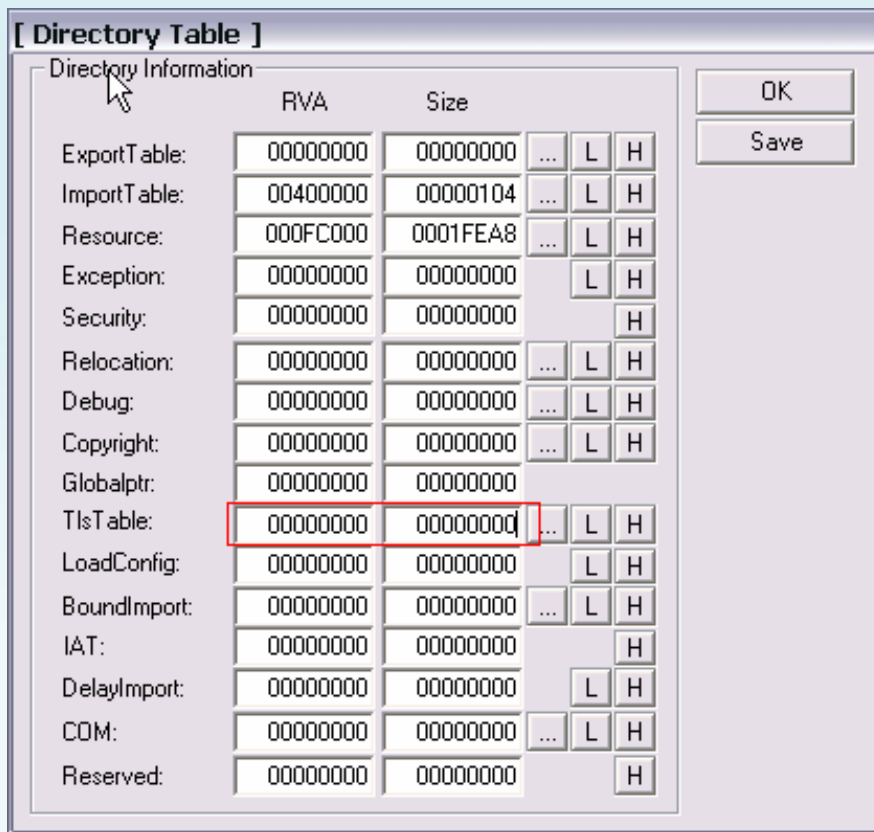
28th Hour Save the file with the name **Dumped_Stolen Bytes_Fix.exe**. huhu ... Target this sector yet it ... man tè of teaching that subject dek first run, including information systems they considered trial Bro Run like crazy ... đâm a fracture of the keyboard ... but always coming back here I think some clear target this quite similar some children pretty fresh, man tí for the district, including the brother lucid enough to find the man for hours with it ... to be taught how to watch ... Load **Dumped_Stolen** to **Bytes_Fix.exe** PE Editor



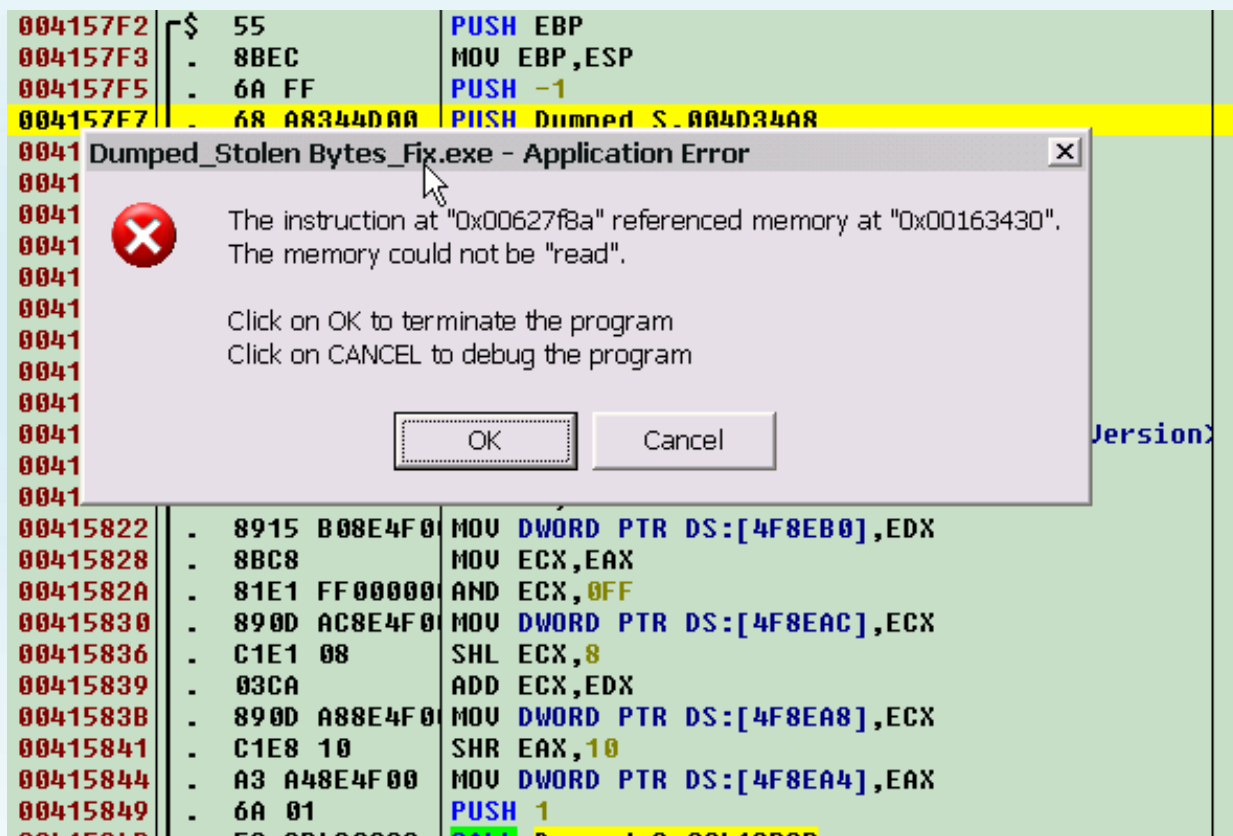
29. We need to edit and **EntryPoint BaseOfCode** properly



and



30th Yeah! Load **Dumped_Stolen Bytes_Fix.exe** to **OllyDBG** and press **Shift + F9**, appear this message



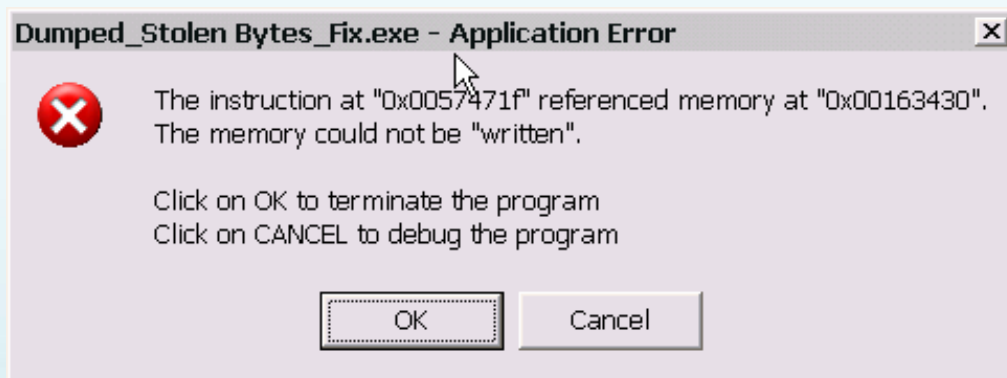
31. In OllyDBG press **Ctrl + G** to enter and come **00627F8A**

00627F8A	8B12	MOV EDX,DWORD PTR DS:[EDX]
00627F8C	2BD0	SUB EDX,EAX
00627F8E	68 2E7E3F3E	PUSH 3E3F7E2E
00627F93	59	POP ECX
00627F94	^ E9 BFBAF5FF	JMP Dumped_S.00583A58
00627F99	52	PUSH EDX
00627F9B	58 01550000	POP Dumped_S.00620540

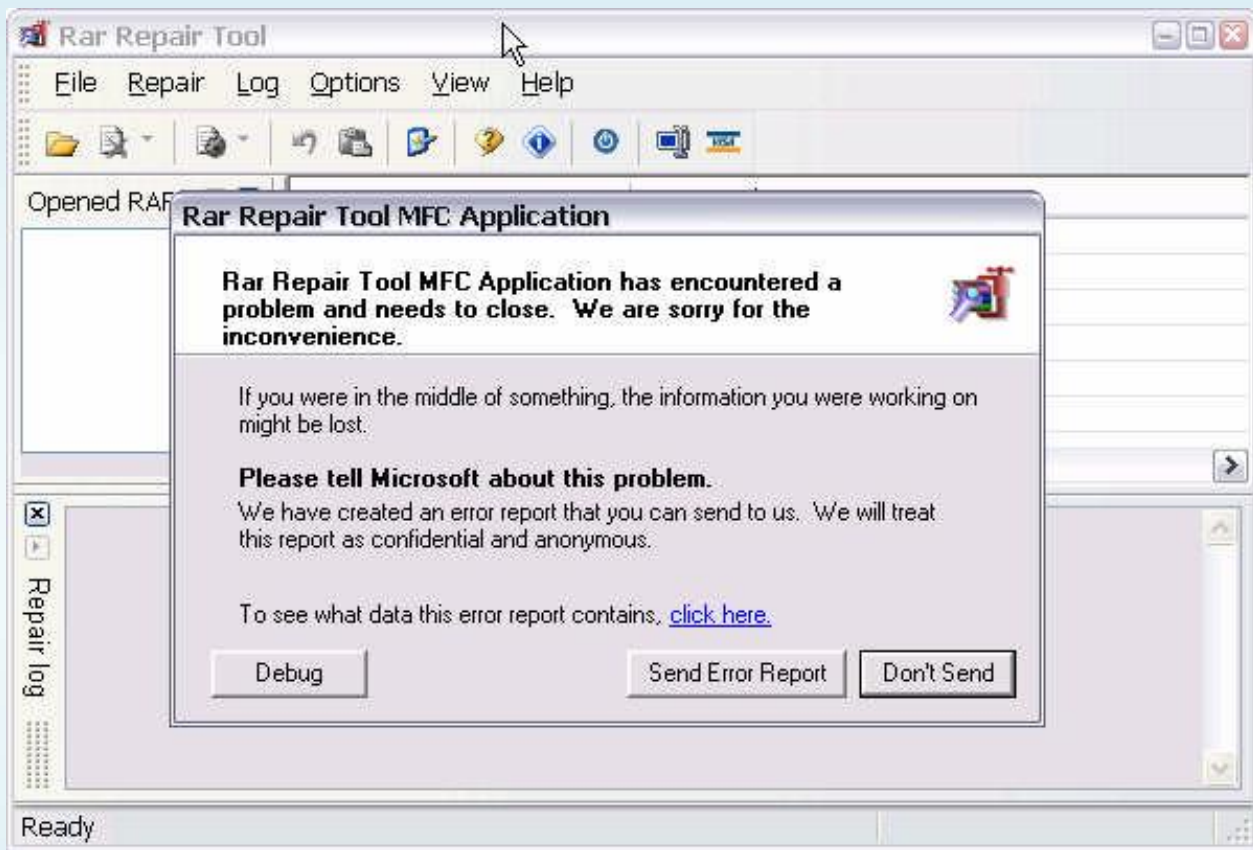
32. Very simple **NOP** it

00627F8A	90	NOP
00627F8B	90	NOP
00627F8C	2BD0	SUB EDX,EAX
00627F8E	68 2E7E3F3E	PUSH 3E3F7E2E
00627F93	59	POP ECX
00627F94	^ E9 BFBAF5FF	JMP Dumped_S.00583A58
00627F99	52	PUSH EDX
00627F9B	58 01550000	POP Dumped_S.00620540

33. Save again, back to **OllyDBG** Load, press **Shift + F9** and will appear 2 NAG error message similar and you just Submitting nhutren



34th Hour meeting minutes to run ... File brothers try their new seo Fix see Hichic lickerish run but only 3 seconds thui then appear Nag error



35.

0041586E	> 88F6	XOR ESI,ESI	
00415870	8975 FC	MOV [LOCAL.1],ESI	
00415873	E8 E28F0000	CALL Dumped_S.00419850	
00415878	FF15 56E24C01	CALL NEAR DWORD PTR DS:[<&kernel32.GetCo	[GetCommandLineA
0041587E	83 50044F00	MOV DWORD PTR DS:[4FB458],EAX	
00415883	E8 A68E0000	CALL Dumped_S.00419728	
00415888	83 940E4F00	MOV DWORD PTR DS:[4F8E94],EAX	
0041588D	E8 498C0000	CALL Dumped_S.00419400	
00415892	E8 8B8B0000	CALL Dumped_S.00419422	
00415897	E8 A3120000	CALL Dumped_S.00416B8F	
0041589C	8975 D0	MOV [LOCAL.12],ESI	
0041589F	8045 00	LEA EAX,[LOCAL.23]	
004158A2	50	PUSH EAX	
004158A3	FF15 56E24C01	CALL NEAR DWORD PTR DS:[<&kernel32.GetS	[pStartupinfo GetStartupInfoA
004158A9	E8 1C8B0000	CALL Dumped_S.004198C0	
004158AE	8045 0C	MOV [LOCAL.25],EAX	
004158B1	F645 D0 01	TEST BYTE PTR SS:[EBP-30],1	
004158B5	74 06	JE SHORT Dumped_S.004158BD	

... hichic pain oi brother too, must do our brothers action that is wrong screen **Check** the **CRC**. Brother sector do it? I also lôt piece of cloth to cover themselves blindfold. Load the file to the new Fix **OlllyDBG**, mouse scroll down a bit you'll see

36. Nhuhinh Set 1 Breakpoint, press **F9**, **F7**

00416B3F	\$	A1 18244F00	MOV EAX,DWORD PTR DS:[4F2418]
00416B44	.	85C0	TEST EAX,EAX
00416B46	~	74 02	JE SHORT Dumped_S.00416B4A
00416B48	.	FFD0	CALL NEAR EAX
00416B4A	>	68 08134F00	PUSH Dumped_S.004F1308
00416B4F	.	68 F0124F00	PUSH Dumped_S.004F12F0
00416B54	.	E8 EC000000	CALL Dumped_S.00416C45
00416B59	.	68 EC124F00	PUSH Dumped_S.004F12EC
00416B5E	.	68 00104F00	PUSH Dumped_S.004F1000
00416B63	.	E8 DD000000	CALL Dumped_S.00416C45
00416B68	.	83C4 10	ADD ESP,10
00416B6B	.	C3	RETN
00416B6C	\$	6A 00	PUSH 0

37. Press **Enter** to function **Call** spotting bright yellow

00416C45	\$	56	PUSH ESI
00416C46	.	8B7424 08	MOV ESI,DWORD PTR SS:[ESP+8]
00416C4A	>	3B7424 0C	CMP ESI,DWORD PTR SS:[ESP+C]
00416C4E	~	73 0D	JNB SHORT Dumped_S.00416C5D
00416C50	.	8B06	MOV EAX,DWORD PTR DS:[ESI]
00416C52	.	85C0	TEST EAX,EAX
00416C54	~	74 02	JE SHORT Dumped_S.00416C58
00416C56	.	FFD0	CALL NEAR EAX
00416C58	>	83C6 04	ADD ESI,4
00416C5B	^	EB ED	JMP SHORT Dumped_S.00416C4A
00416C5D	>	5E	POP ESI
00416C5E	.	C3	RETN

38. Set 1 BreakPoint nhuhinh in order Call, Press **Shift + F9 25 times** and hit **F7**

00406250	. 55	PUSH EBP
00406251	. 8BEC	MOV EBP,ESP
00406253	. E8 08000000	CALL Dumped_S.00406260
00406258	. 5D	POP EBP
00406259	. C3	RETN
0040625A	. CC	INT3
0040625B	. CC	INT3
0040625C	. CC	INT3
0040625D	. CC	INT3
0040625E	. CC	INT3
0040625F	. CC	INT3
00406260	\$ 55	PUSH EBP
00406261	. 8BEC	MOV EBP,ESP
00406263	. E8 886A1500	CALL Dumped_S.0055CCF0
00406268	. E8 D7941900	CALL Dumped_S.0059F744
0040626D	. 5D	POP EBP
0040626E	. C3	RETN
0040626F	. CC	INT3

Lenh Call
goi ham
Check CRC

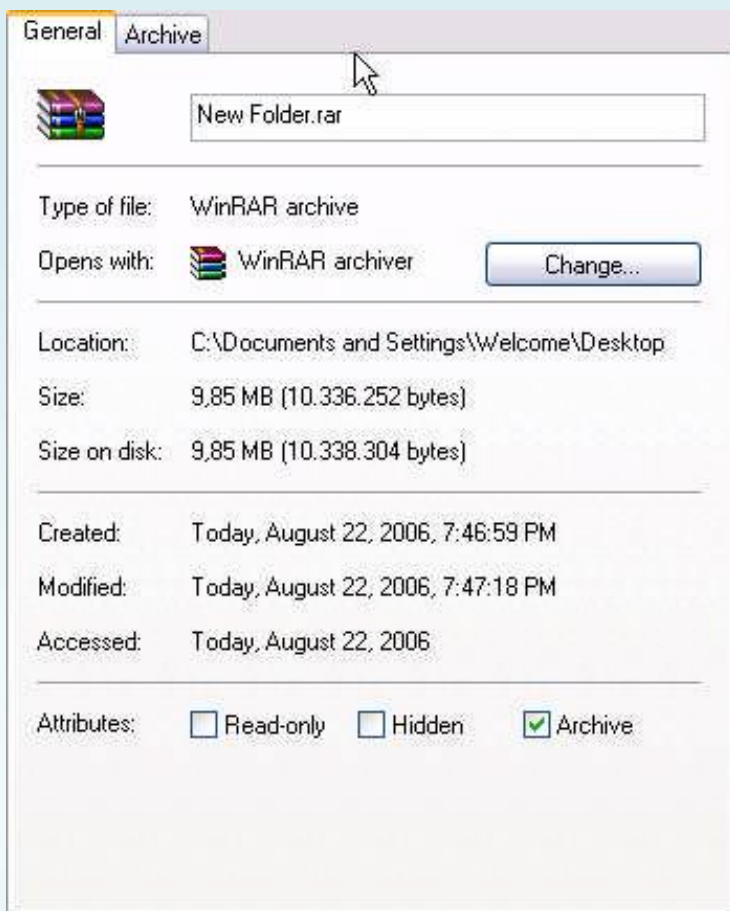
39. Call the NOP command

0040624E	. 00	RETN
0040624F	. CC	INT3
00406250	. 55	PUSH EBP
00406251	. 8BEC	MOV EBP,ESP
00406253	. E8 08000000	CALL Dumped_S.00406260
00406258	. 5D	POP EBP
00406259	. C3	RETN
0040625A	. CC	INT3
0040625B	. CC	INT3
0040625C	. CC	INT3
0040625D	. CC	INT3
0040625E	. CC	INT3
0040625F	. CC	INT3
00406260	\$ 55	PUSH EBP
00406261	. 8BEC	MOV EBP,ESP
00406263	. E8 886A1500	CALL Dumped_S.0055CCF0
00406268	. 90	NOP
00406269	. 90	NOP
0040626A	. 90	NOP
0040626B	. 90	NOP
0040626C	. 90	NOP
0040626D	. 5D	POP EBP
0040626E	. C3	RETN
0040626F	. CC	INT3

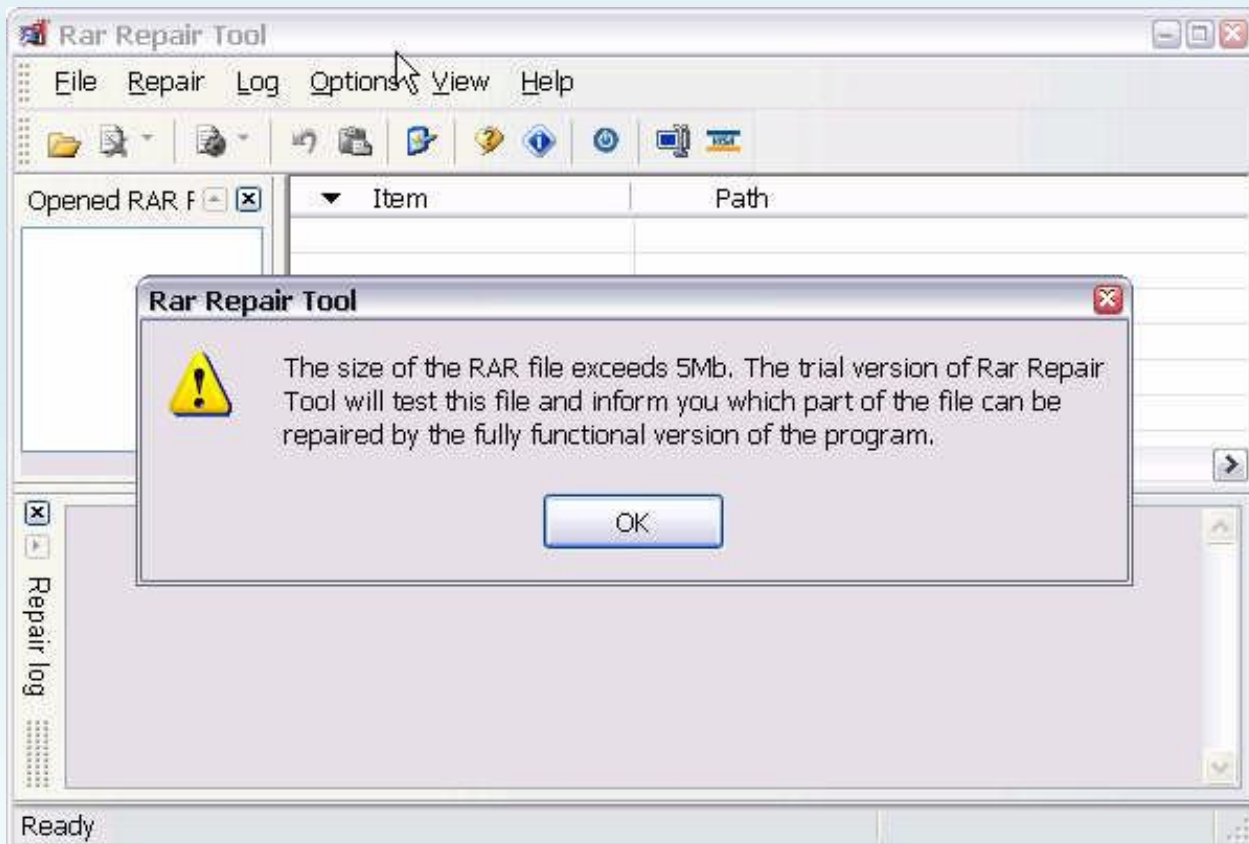
40th Save File Run Try Again Hehehehe. Lickerish run, they end it kneel ...
Unpacked successful!

IV. C c k i n g:

_Preparing 1 *. RAR files have storage> 5Mb



Unpacked _Run File and select File this



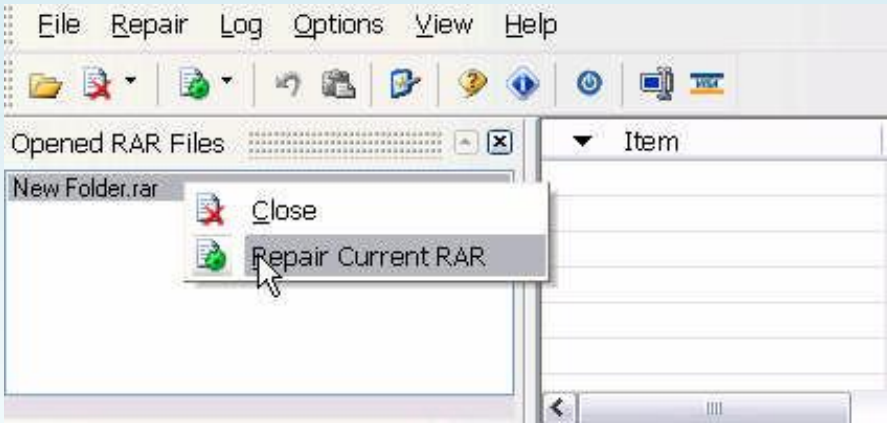
file:///C:/RCE%20Unpacking%20eBook%20[Tr...Cracking%20RAR%20Repair%20Tool%203.0.htm (25 of 31) [1/9/2009 9:46:51 LithiumLi]

00597B63	F3:	PREFIX REP:	superfluous prefix
00597B64	8B09	MOV ECX,DWORD PTR DS:[ECX]	
00597B66	53	PUSH EBX	
00597B67	8BD9	MOV EBX,ECX	
00597B69	871C24	XCHG DWORD PTR SS:[ESP],EBX	
00597B6C	FF15 94E24C00	CALL NEAR DWORD PTR DS:[<&kernel32.GetF	kernel32.GetFileSize
00597B72	E8 3ED2FBFF	CALL Dumped_S.00554DB5	
00597B77	5A	POP EDX	
00597B78	B5 4D	MOV CH,4D	
00597B7A	60	PUSHAD	
00597B7B	60	PUSHAD	
00597B7C	05 00110E00	ADD EAX,000E1100	

_ Very gently NOP Ham Call GetFileSzie

00597B64	8B09	MOV ECX,DWORD PTR DS:[ECX]	
00597B66	53	PUSH EBX	
00597B67	8BD9	MOV EBX,ECX	
00597B69	871C24	XCHG DWORD PTR SS:[ESP],EBX	
00597B6C	90	NOP	
00597B6D	90	NOP	
00597B6E	90	NOP	
00597B6F	90	NOP	
00597B70	90	NOP	
00597B71	90	NOP	
00597B72	E8 3ED2FBFF	CALL Dumped_S.00554DB5	
00597B77	5A	POP EDX	
00597B78	B5 4D	MOV CH,4D	

_ **Save** the file and run the test results. Load File> 5Mb to do springiness **NAG** parentheses is the way it nãi subject dek Save the results after conducting the Fix File Rar fail. Hichic ... quite Láu the Star ko load file just to patch **OllyDBG**, Press **Shift + F9** to run completely, select the file, type **Alt + F1** to **Bp GetFileSize** and nhuhinh



_ Select seats for Save need results and we will stop in Olly

Address	Value	Comment
7C810C8F	8BFF	MOV EDI,EDI
7C810C91	55	PUSH EBP
7C810C92	8BEC	MOV EBP,ESP
7C810C94	51	PUSH ECX
7C810C95	51	PUSH ECX
7C810C96	8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]
7C810C99	50	PUSH EAX
7C810C9A	FF75 08	PUSH DWORD PTR SS:[EBP+8]
7C810C9D	E8 7FFFFFFF	CALL kernel32.GetFileSizeEx
7C810CA2	85C0	TEST EAX,EAX
7C810CA4	0F84 EA8FFFFF	JE kernel32.7C809C94
7C810CAA	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
7C810CAD	85C0	TEST EAX,EAX
7C810CAF	0F85 D58FFFFF	JNZ kernel32.7C809C8A

Address	Value	Comment
004F1000	00000000	
004F1004	004B5675	Dumped
004F1008	004B56B5	Dumped
004F100C	004B5798	Dumped
004F1010	004B5A09	Dumped

Address	Value	Comment
0012F6FC	006285C0	CALL to GetFileSize from Dumped_S.006285BA
0012F700	000000B8	hFile = 000000B8 (window)
0012F704	0012F988	pFileSizeHigh = 0012F988
0012F708	00000000	
0012F70C	00D77560	
0012F710	00D7924C	

_ Press Alt + F9

Address	Value	Comment
006285AF	E5 E9	IN EAX,0E9
006285B1	3E:B4 00	MOV AH,0
006285B4	008B F9873C24	ADD BYTE PTR DS:[EBX+243C87F9],
006285BA	FF15 94E24C00	CALL NEAR DWORD PTR DS:[<&kerne
006285C0	E8 A1B9FDFF	CALL Dumped_S.00603F66
006285C5	F0:25 15ECE03E	LOCK AND EAX,3EE0EC15
006285CB	23D2	AND EDX,EDX
006285CD	E9 903AF8FF	JMP Dumped_S.005AC062
006285D2	870C24	XCHG DWORD PTR SS:[ESP],ECX

_ Observation you will see storage *. Rar file is contained in the acceptance on EAX

Command
HEX: 9DB7FC - DEC: 10336252 - ASCII: □-0

Opens with:

WinRAR archiver

Location:
C:\Documents and Settings\Welcome\

Size:
9.85 MB (10.336.252 bytes)

_ Where you can do NOP raw bạo nhutren that need lithe little children it is a new subject

006285B7	3E B4 00	MOV AH,0	superfluous prefix
006285B4	008B F9873C24	ADD BYTE PTR DS:[EBX+243C87F9],	
006285BA	90	NOP	
006285BB	58	POP EAX	
006285BC	58	POP EAX	
006285BD	90	NOP	
006285BE	90	NOP	
006285BF	90	NOP	
006285C0	E8 A1B9FDFF	CALL Dropped_S.00603F66	
006285C5	F0:25 15ECE03E	LOCK AND EAX,3EE0EC15	LOCK prefix is not allowed
006285CB	23D2	AND EDX,EDX	
006285CD	E9 903AF8FF	JMP Dropped_S.005AC062	
006285D2	870C24	XCHG DWORD PTR SS:[ESP],ECX	
006285D5	59	POP ECX	
006285D6	50	PUSH EAX	

_ Save the file Test try hahahaha ... **Full Function** ... giờ only hours each eye is thirstly thui



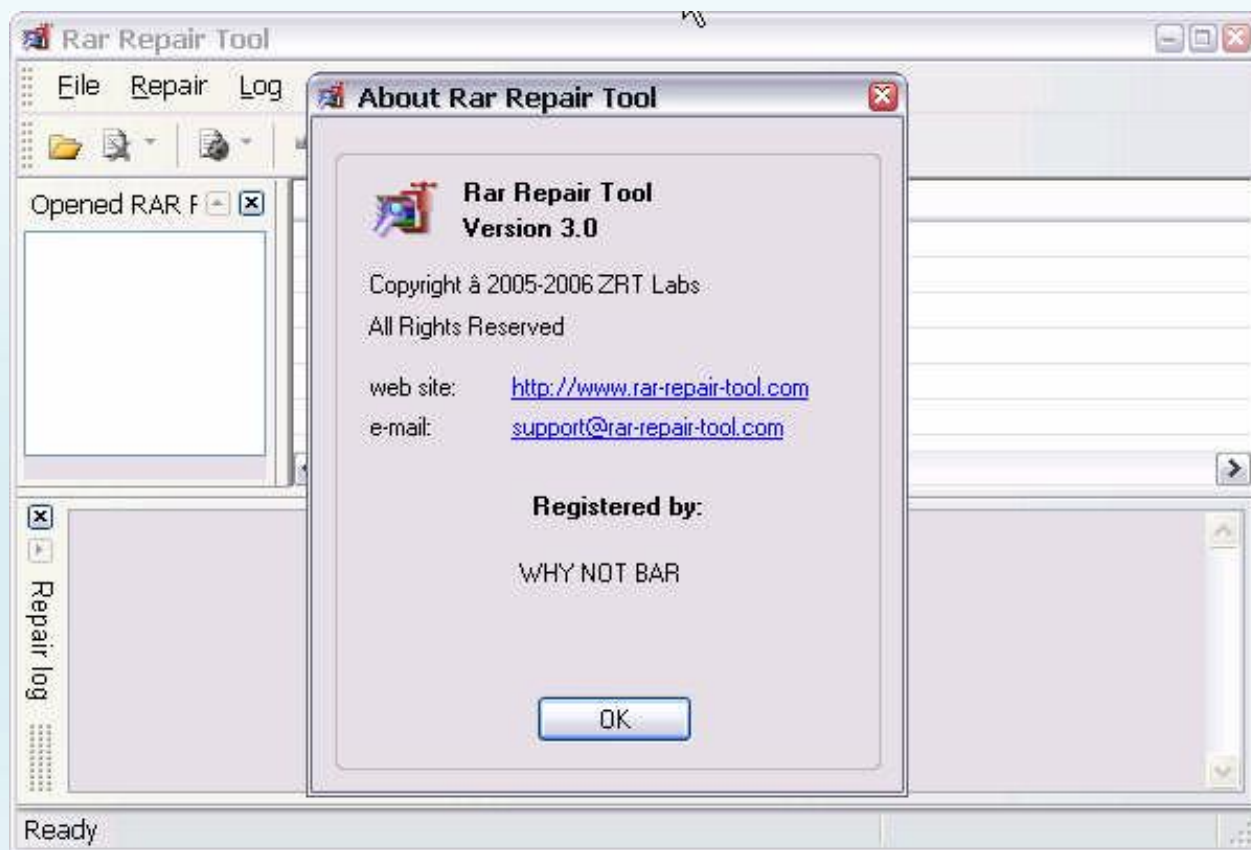
_ To treat thui load on "**UltraEdit-32**" Search string *Unregis* you will come

```

)00f12c0h: 0f 00 4c 00 57 00 4c 00 88 00 4c 00 2f 01 4c 00 ; ..L.W.L.L.L.L.
)00f12d0h: 5c 01 4c 00 89 01 4c 00 b6 01 4c 00 57 02 4c 00 ; \.L%.L.q.L.W.L.
)00f12e0h: 19 03 4c 00 7b 03 4c 00 71 05 4c 00 00 00 00 00 ; ..L.{.L.q.L....
)00f12f0h: 00 00 00 00 25 4c 41 00 6f 75 41 00 ae db 41 00 ; ....%LA.ouA.@UA.
)00f1300h: 28 7f 49 00 86 8c 41 00 00 00 00 00 00 00 00 00 ; (OI.+EA.....
)00f1310h: 56 dc 41 00 00 00 00 00 00 00 00 00 97 8c 41 00 ; VUA.....-EA.
)00f1320h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
)00f1330h: 55 4e 52 45 47 49 53 54 45 52 45 44 20 43 4f 50 ; UNREGISTERED COP
)00f1340h: 59 20 2d 20 50 4c 45 41 53 45 20 52 45 47 49 53 ; Y - PLEASE REGIS
)00f1350h: 54 45 52 00 53 65 72 69 61 6c 4e 75 6d 62 65 72 ; TER.SerialNumber
)00f1360h: 00 00 00 00 52 65 67 69 73 74 72 61 74 69 6f 6e ; ....Registration
)00f1370h: 4e 61 6d 65 00 00 00 00 2e 25 64 00 20 00 00 00 ; Name.....%d. ...
)00f1380h: 25 64 2e 25 64 00 00 00 6d 61 69 6c 74 6f 3a 00 ; %d.%d...mailto:.
)00f1390h: 43 34 00 00 43 33 00 00 43 32 00 00 43 6f 6c 75 ; C4..C3..C2..Colu
)00f13a0h: 6d 6e 73 57 69 64 74 68 00 00 00 00 43 31 00 00 ; mnsWidth....C1..
)00f13b0h: c8 13 4f 00 b8 13 4f 00 72 72 74 75 70 74 6d 70 ; È.O.,.O.rrtuptmp
)00f13c0h: 2e 64 61 74 00 00 00 00 72 72 74 70 74 6d 70 00 ; .dat....rrtptmp.
)00f13d0h: 52 54 41 54 40 40 00 00 72 2e 2e 00 21 00 00 00 ; STATC...+

```

_ To insert name them any treatment that they do have the ideas But they put the name of the children



_If any aged Chum can Code Kegen Newbie ... She should be able to do is obscure the Pro level, they should abandon the thui other Bro. Bibi .. appointment in tut other time they called the man to tí This reduction xo chết

Gre e Ts italy Ou F lt: C om p ut _ An e r l e g, e Zombi, M oo n b by a, c H a grape, B e n i n a k e

i n a m n o w a r, o i Z, D x u e, M e r c, i l e g H T p h o w h e r e x, c k y b o T r i i t a l y, T a k a d a i a

m i d i o t, a n d e n t h a n d i n e, a n d y o u ...!

Ordenar all summer

Nha Trang, on 23 8 2006

Why Not Bar

Unpacking ActiveMark level 2 entry point

I. Introduction:

There are a lot of game when down by Pack **ActiveMark**. Protect the one developed for this game and have caused many difficulties for us when the game meat that the presence of it. At first she has 1 tut Manual unpack ActiveMark 5.xx and to officially ActiveMark dust they write more about 1 tut unpack ActiveMark but as **level 2 entry point**. This type is quite prevalent and quite popular request should she wrote. In tut they presented to unpack ActiveMARK **level 2 entry point** to use tuManual Tool available. How how it will read more clearly and then

II. Tools:

- Tool and the AC A P l a c i n g a n d ú n g :
 - *Oll italy B D G 1. 1 0*
 - *Lord P E 1. 4*
 - *H x W e o r k s h o p 4. 2*
 - *I m p o r t R E C 1. 6*
 - *P I D E 0. 9 4 v a p l i n t h e G E O F P r e d i n*
 - *A M D u m p e r f o r e v C T I A M A R K*
 - *A c t i v e M a r k . V e r s i o n*
- T a r g e t : **T h e D a V i n c i d e C o**
H o m e p a g e : [http://w w w . b i g h f i s a g e m . s . C o m /](http://www.bighfisagem.s.Com/)

III. Unpacking:

1st T i m e O P V A D u m u l l P F :

C á c h 1: Code of the al

_ D u n g **Eid P o c i t a t e r g a c o p c k h a k o i t a l y** and use lugin **P O F E P d e r i n**



_ Although **PEiD** ko identify the target is by Protect **ActiveMark**. To assert that you can use **ActiveMark**. Version

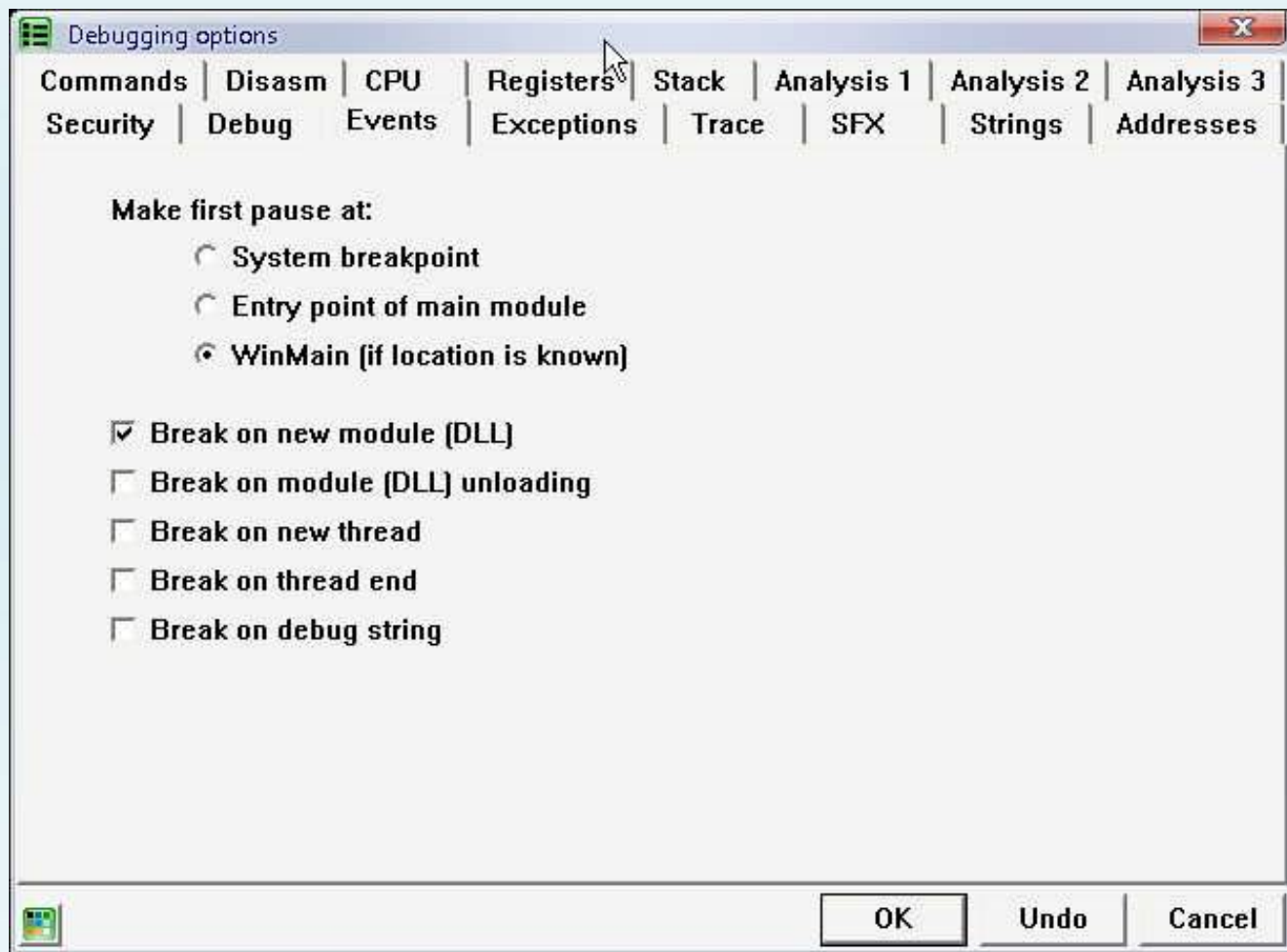


_ Nhuvay we know exactly which version is used and ActiveMARK target this **OEP = 005EE7B3**. now we load

directly to Target **OllyDBG** and we stop here

Address	Hex dump	Disassembly	Comment
00738000	8925 04107500	MOV DWORD PTR DS:[751004],ESP	
00738006	68 2E807300	PUSH 73802E	
0073800B	7F 03	JG SHORT 00738010	00738010
0073800D	90	NOP	
0073800E	8D3F	LEA EDI,DWORD PTR DS:[EDI]	
00738010	7E 04	JLE SHORT 00738016	00738016
00738012	EB 02	JMP SHORT 00738016	00738016
00738014	FA	CLI	
00738015	B6 64	MOV DH,64	
00738017	FF35 00000000	PUSH DWORD PTR DS:[0]	
0073801D	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00738024	EB 03	JMP SHORT 00738029	00738029
00738026	D089 A3E96504	ROR BYTE PTR DS:[ECX+465E9A3],	
0073802C	0000	ADD BYTE PTR DS:[EAX],AL	
0073802E	FF	PUSH EBP	

_ Press **Alt + O** and select as follows:



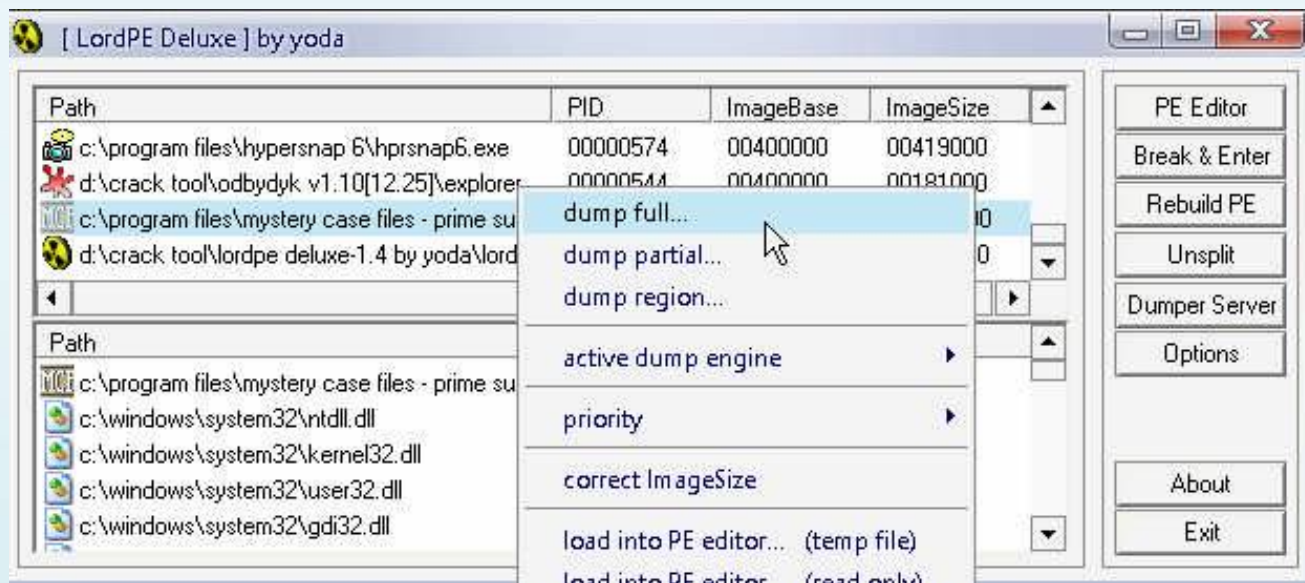
_ Press **Alt + F1** and **005EE7B3** He Set 1



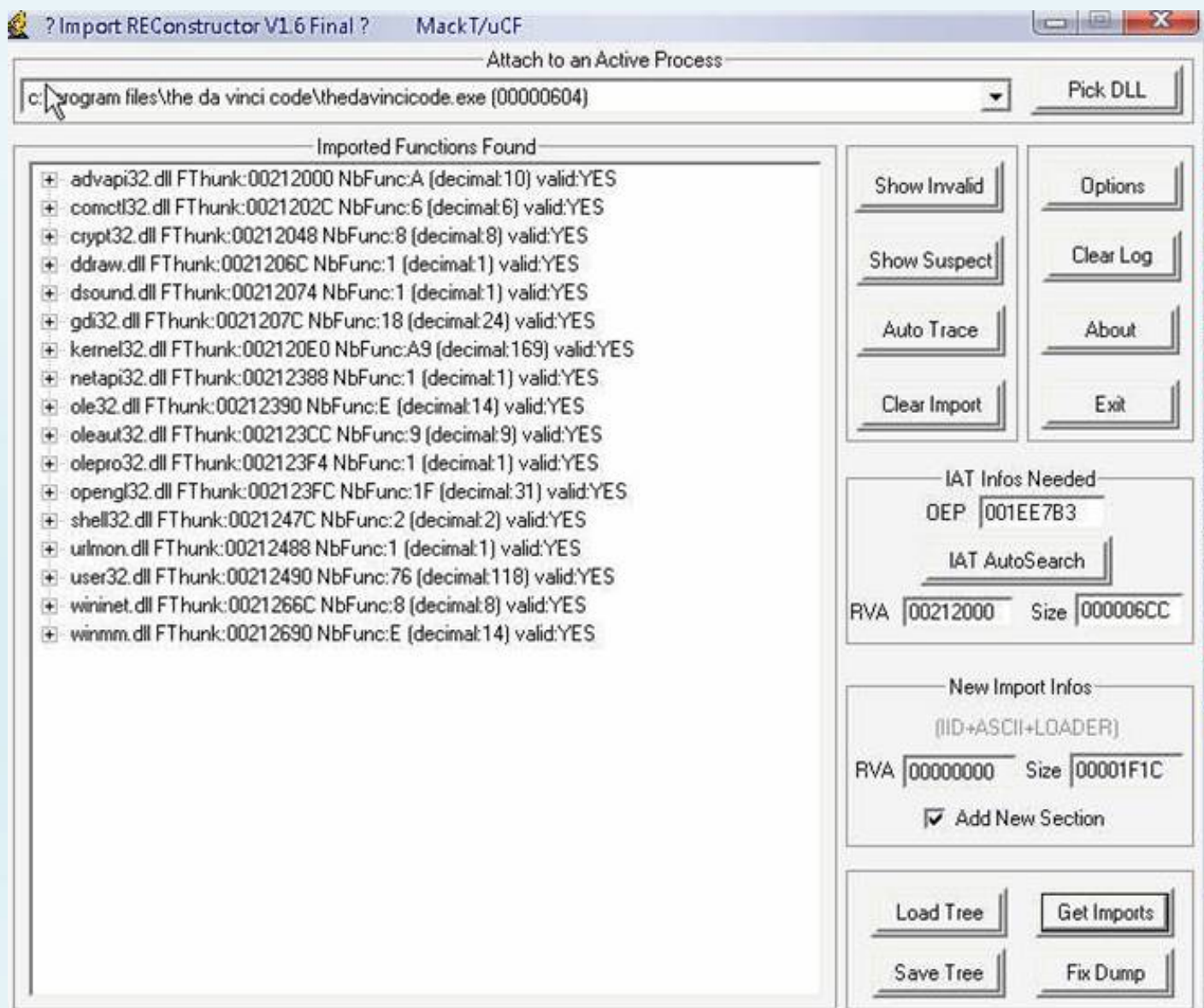
_ Continue to press **F9** to stop at the **Set Breakpoint** and is **OEP** (At this target, we hit 12 times F9)

Address	Hex dump	Disassembly	Comment
005EE7B3	55	PUSH EBP	<==OEP
005EE7B4	8BEC	MOV EBP,ESP	
005EE7B6	6A FF	PUSH -1	
005EE7B8	68 08F75000	PUSH 50F708	
005EE7BD	68 904B5F00	PUSH 5F4B90	
005EE7C2	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
005EE7C8	50	PUSH EAX	
005EE7C9	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
005EE7D0	83EC 58	SUB ESP,58	
005EE7D3	53	PUSH EBX	
005EE7D4	56	PUSH ESI	
005EE7D5	57	PUSH EDI	
005EE7D6	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
005EE7D9	FF15 24226100	CALL NEAR DWORD PTR DS:[612224]	kernel32.GetVersion
005EE7DF	33D2	XOR EDX,EDX	
005EE7E1	8AD4	MOV DL,AH	
005EE7E3	8915 58D46000	MOV DWORD PTR DS:[60D458],EDX	
005EE7E9	8BC8	MOV ECX,EAX	
005EE7EB	81E1 FF000000	AND ECX,0FF	

_Dung **LordPE** to dump Full



_ Open **ImportREC** select and fill **PID OEP**



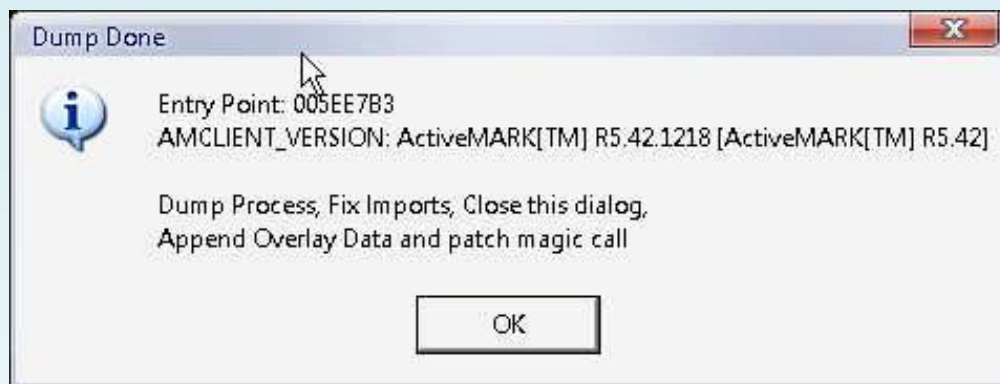
_ Click **Fix dump** select File **dumped.exe**. **Dumped_.exe** Run Test File. hahahaha ... they run chit line We need to add 1 more step encrypted data is copied from the original file to file, the file again **Dumped_.exe** new run

C á c h 2: d ù n o o g l t a d m u m p e r f o r r A c t i v e M A R K

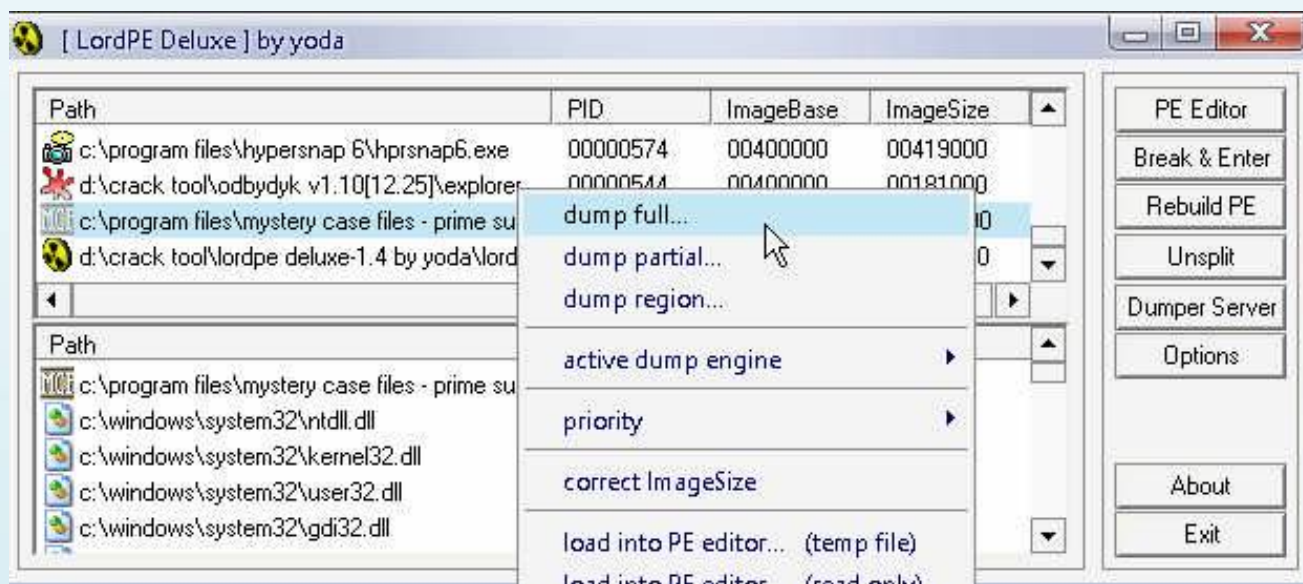
_ M i n A d m m u p e r :



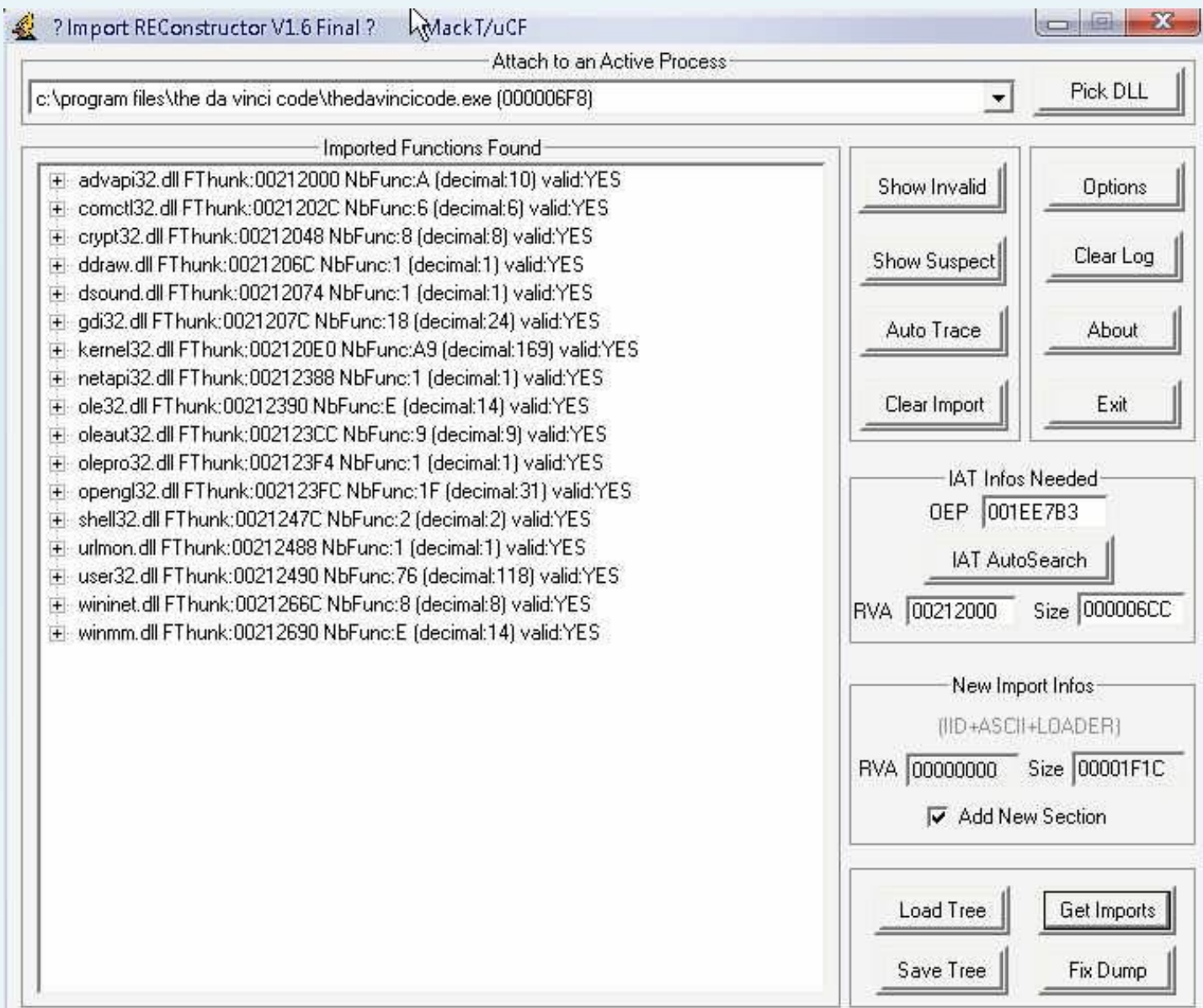
_ Select File "**TheDaVinciCode.exe**" wait you will see



___ Using **LordPE** to dump Full



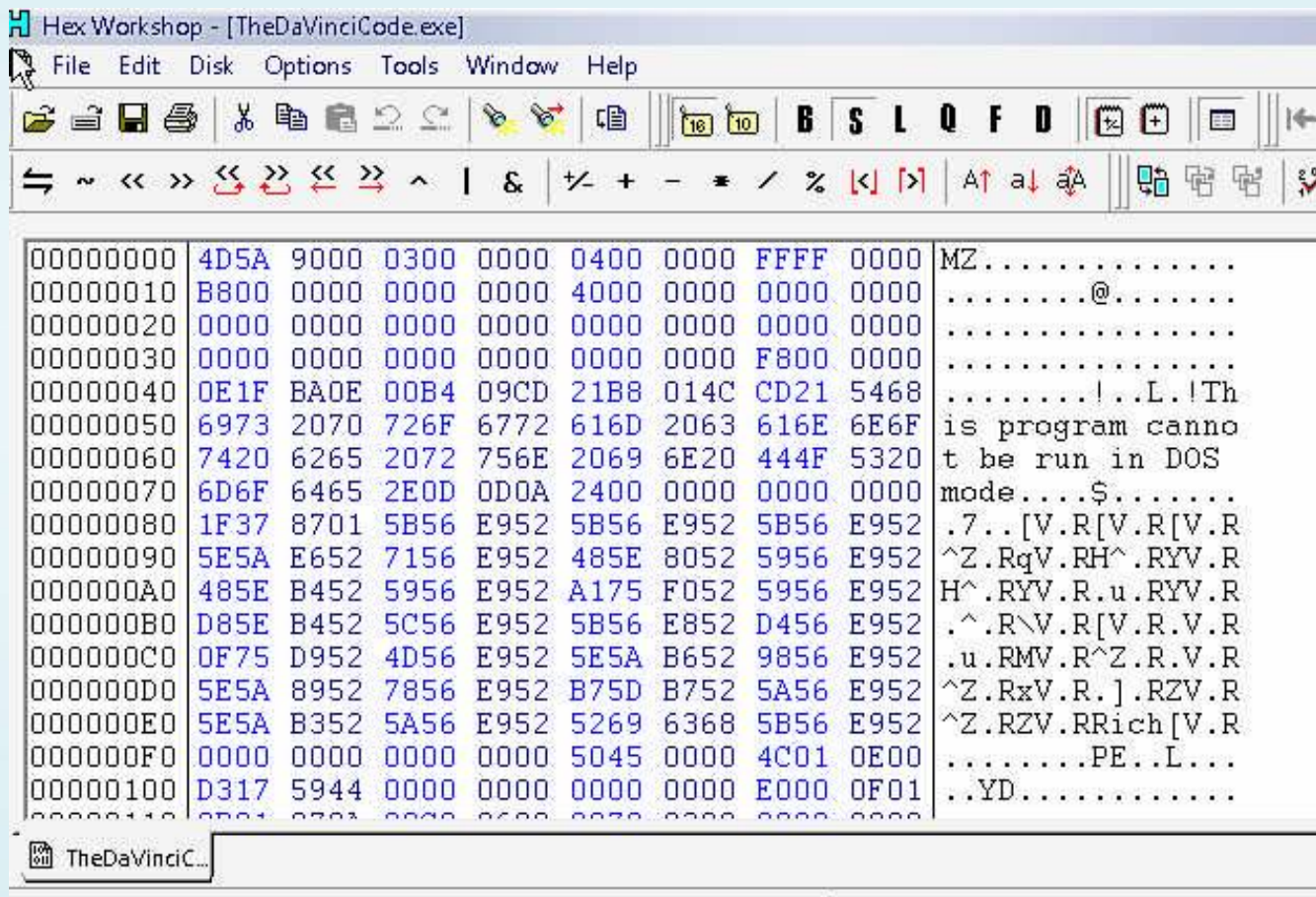
_ Open **ImportREC** select and fill **PID OEP**



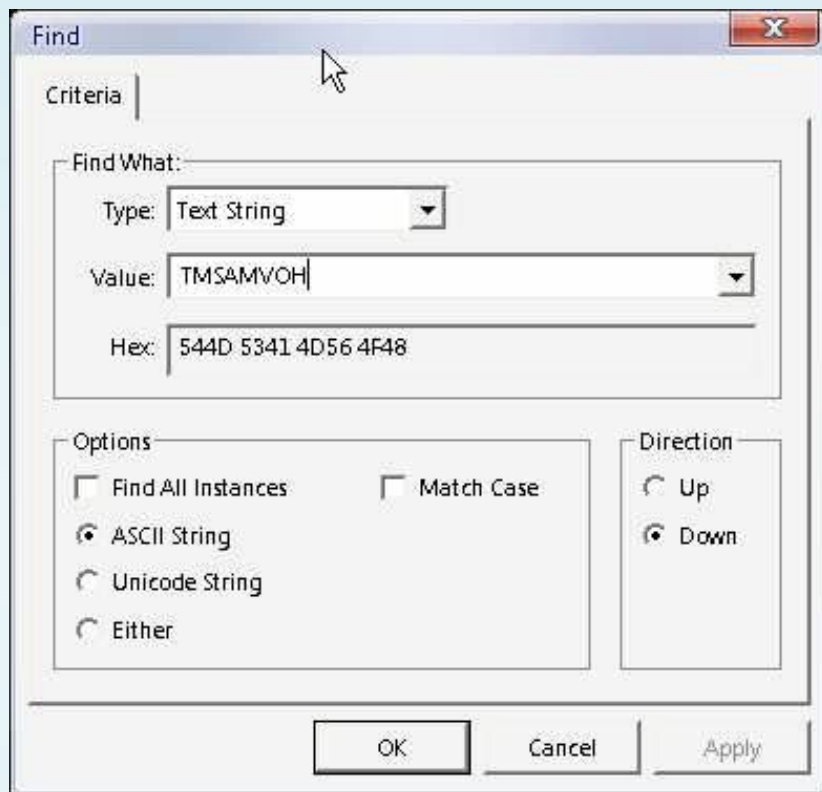
_Hehe .. result similar way 1

2nd Fixing Dumped .Exe

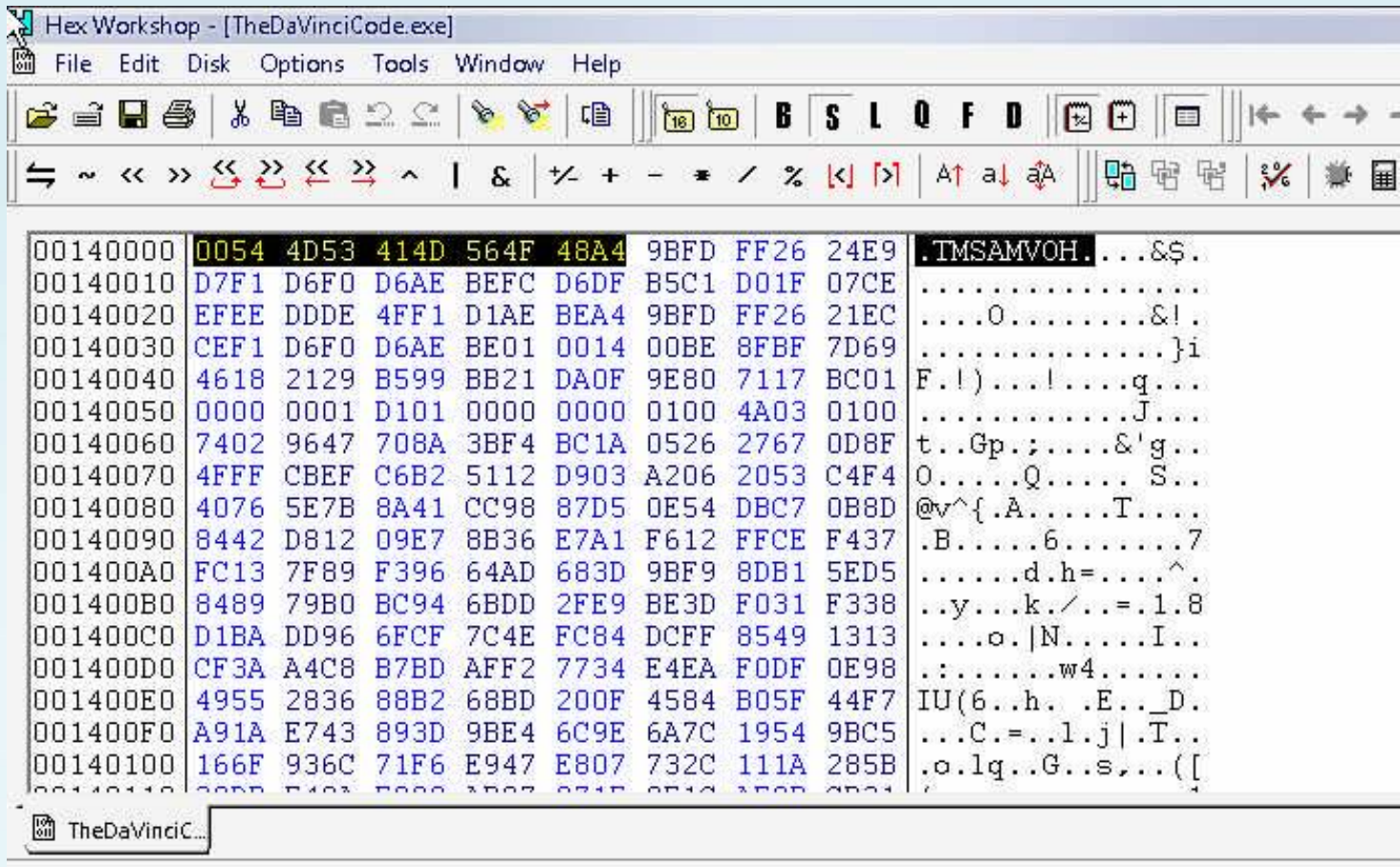
_Bay Time we conducted Fix File **Dumped.exe**. Running *Hex Workshop 4.2* and select File **"TheDaVinciCode.exe"** and we are as follows



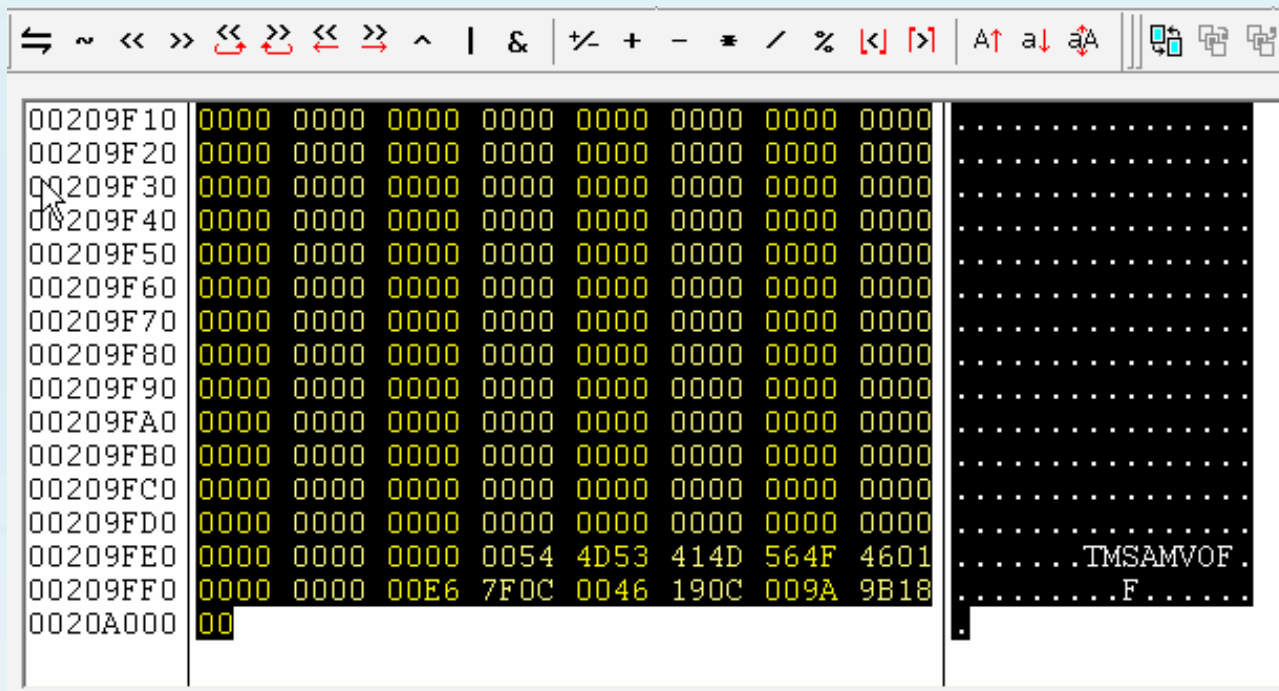
_ Press **Ctrl + F** and type "**TMSAMVOH**"



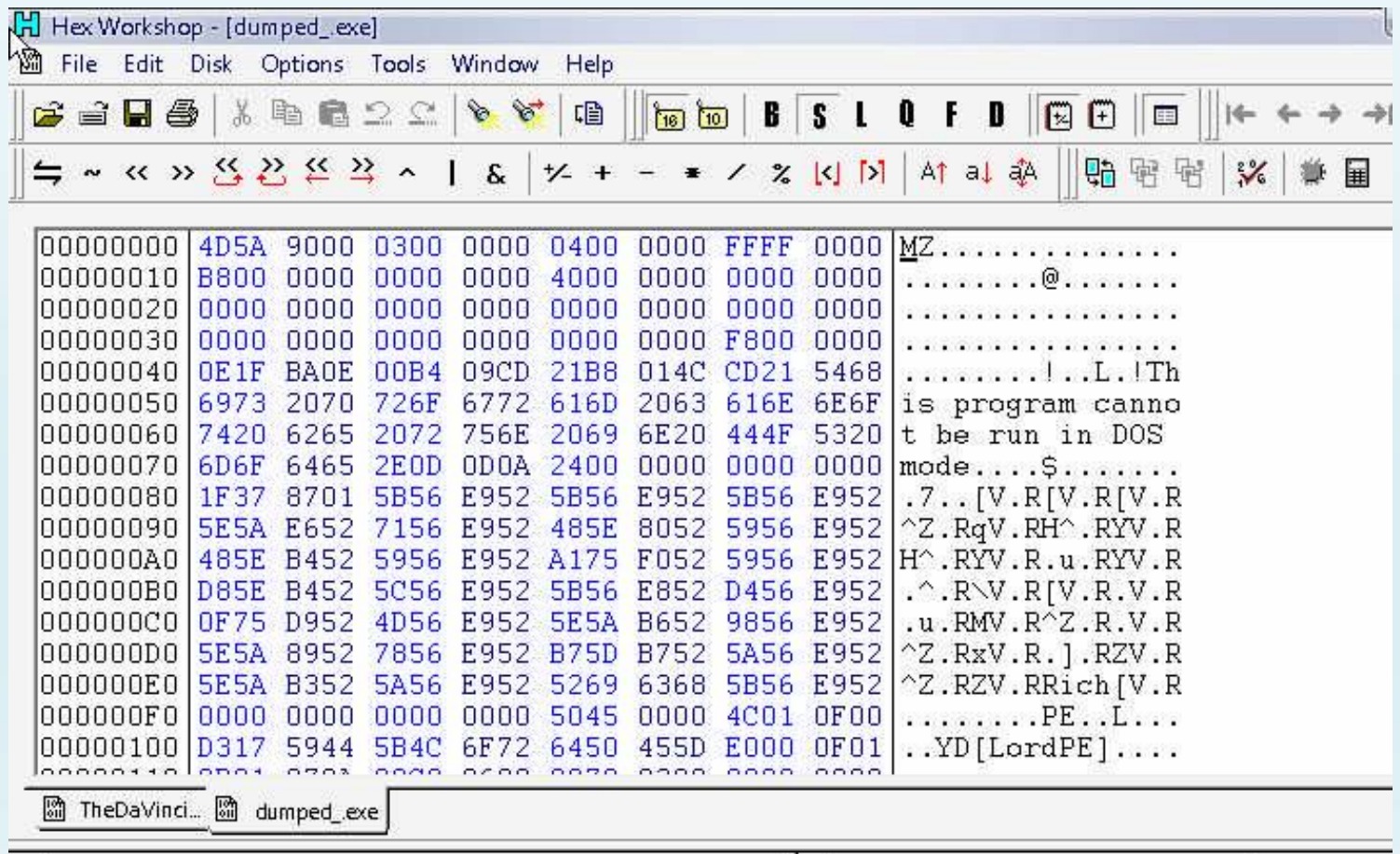
_ Click **OK** to you here



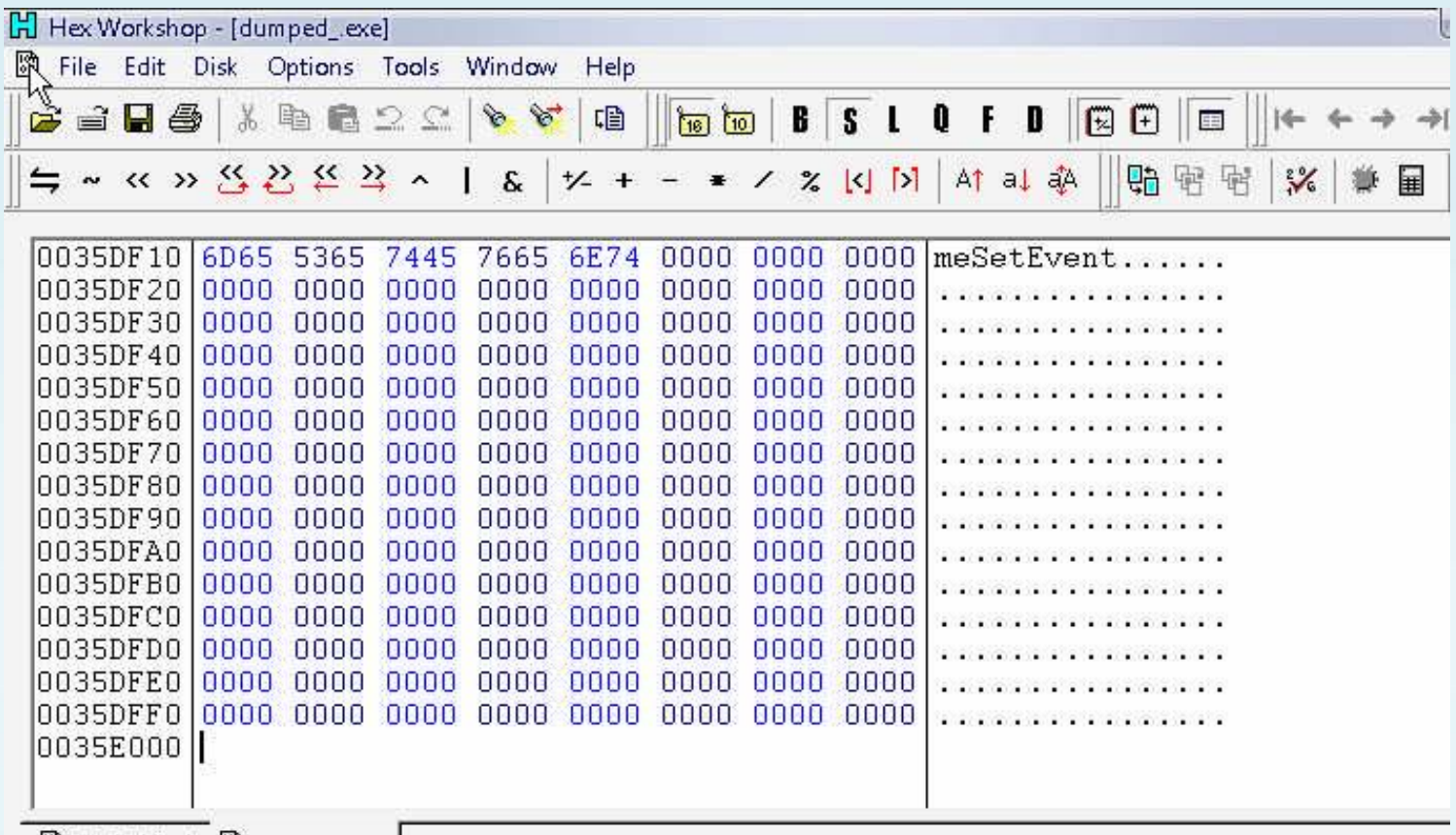
_ Follow from **00140000** to address final



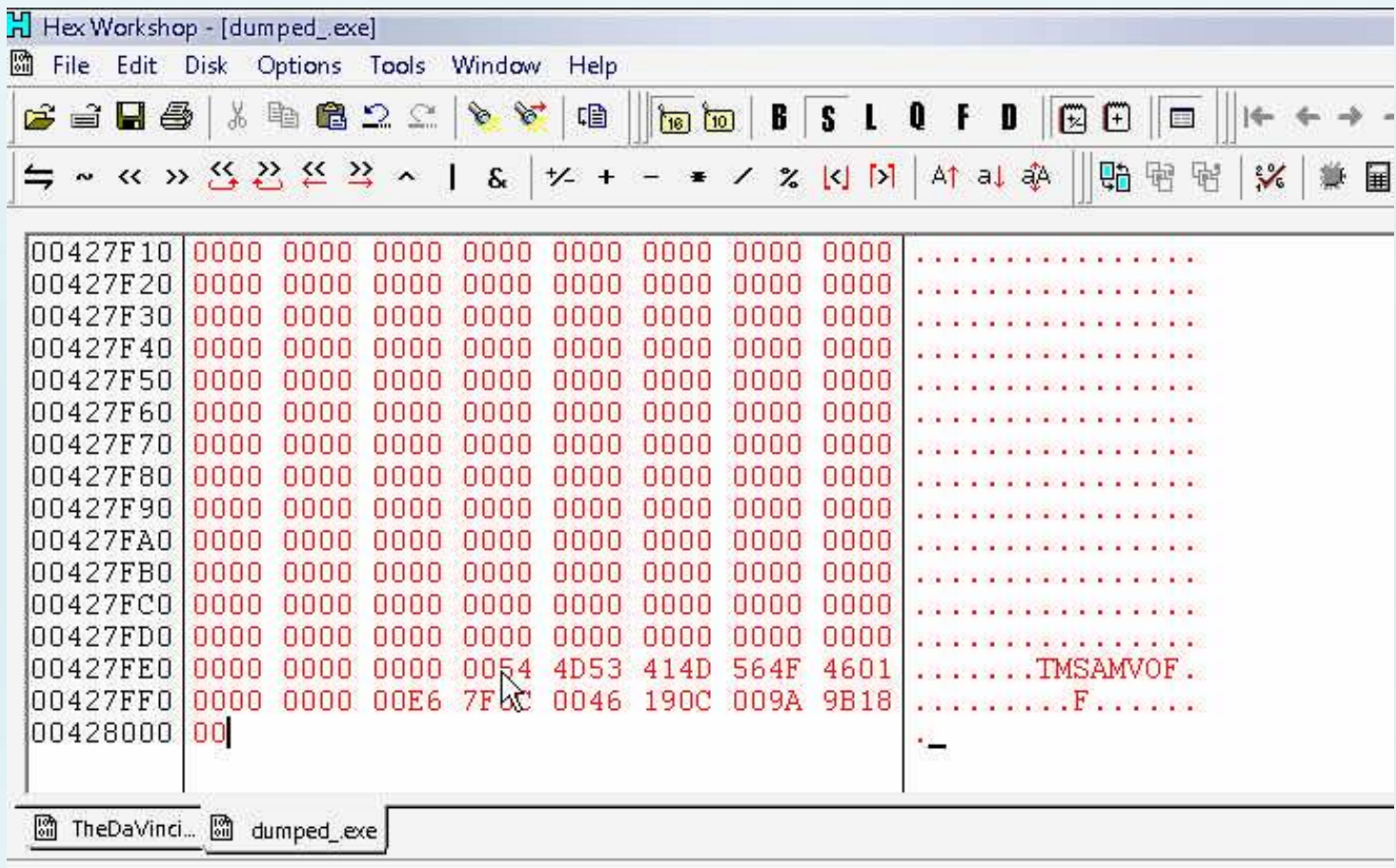
_ Press **Ctrl + C** to copy this code and use *Hex Workshop 4.2* to open files **Dumped_.exe**



_ Scroll down to the last address



_ Press **Ctrl + V** to Paste the code we **Copy** ago when this



_ **Save** the time and name to **Unpacked.exe**. File test **Unpacked.exe** considered stars ... haha ... Numbness both running File lickerish **Unpacked successful**



IV. Cracking:

Thua win to finish we continue with the task to **Bypass** Cracking the **NAG** and **Remove Time limit** or say otherwise is to get the Full. Speaking as a Load File Unpacked.exe to Olly

Address	Hex dump	Disassembly	Comment
005EE7B3	55	PUSH EBP	
005EE7B4	8BEC	MOV EBP,ESP	
005EE7B6	6A FF	PUSH -1	
005EE7B8	68 08F75000	PUSH 50F708	
005EE7BD	68 904B5F00	PUSH 5F4B90	
005EE7C2	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
005EE7C8	50	PUSH EAX	
005EE7C9	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
005EE7D0	83EC 58	SUB ESP,58	
005EE7D3	53	PUSH EBX	
005EE7D4	56	PUSH ESI	
005EE7D5	57	PUSH EDI	
005EE7D6	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
005EE7D9	FF15 24226100	CALL NEAR DWORD PTR DS:[612224]	kernel32.GetVersion
005EE7DF	33D2	XOR EDX,EDX	
005EE7E1	8AD4	MOV DL,AH	
005EE7E3	8915 58D46000	MOV DWORD PTR DS:[60D458],EDX	
005EE7E9	8BC8	MOV ECX,EAX	
005EE7EB	81E1 FF000000	AND ECX,0FF	
005EE7F1	890D 54D46000	MOV DWORD PTR DS:[60D454],ECX	
005EE7F7	C1E1 08	SHL ECX,8	
005EE7FA	03CA	ADD ECX,EDX	
005EE7FC	890D 58D46000	MOV DWORD PTR DS:[60D458],ECX	

_ Click to select the image and

The screenshot shows the OllyDbg interface. The main window displays a memory dump with columns for Address, Hex, and Disassembly. The address range is from 005EE7B3 to 005EE805. The disassembly column shows various instructions like PUSH, MOV, CALL, XOR, AND, SHL, ADD, MOV, SHR, and MOV. The 'Copy' menu is open, showing options like Binary, Assemble, Label, Comment, Breakpoint, Hit trace, Run trace, Go to, Follow in Dump, View call tree, Search for, Find references to, View, Copy to executable, Analysis, Detach Process, Process Patcher, Analyze This!, Asm2Clipboard, Bookmark, and Code Ripper. The 'Search for' option is highlighted, and a search dialog is open, showing the search criteria 'Name (label) in current module' and 'Name in all modules'. The search results list various commands and sequences, including 'SE handler installation'.

Address	Hex	Disassembly
005EE7B3	55	PUSH
005EE7B4	8BEC	MOV
005EE7B6	6A FF	PUSH
005EE7B8	68 08F75000	PUSH
005EE7BD	68 904B5F00	PUSH
005EE7C2	64:A1 00000000	MOV
005EE7C8	50	PUSH
005EE7C9	64:8925 00000000	MOV
005EE7D0	83EC 58	SUB
005EE7D3	53	PUSH
005EE7D4	56	PUSH
005EE7D5	57	PUSH
005EE7D6	8965 E8	MOV
005EE7D9	FF15 24226100	CALL
005EE7DF	33D2	XOR
005EE7E1	8AD4	MOV
005EE7E3	8915 58D46000	MOV
005EE7E9	8BC8	MOV
005EE7EB	81E1 FF000000	AND
005EE7F1	890D 54D46000	MOV
005EE7F7	C1E1 08	SHL
005EE7FA	03CA	ADD
005EE7FC	890D 50D46000	MOV
005EE802	C1E8 10	SHR
005EE805	A3 4CD46000	MOV

_Next we work under nhuhinh

Address	Disassembly	Text string
00401507	PUSH 46DD90	ASCII "rot"
0040156E	PUSH 46DD94	ASCII "body"
004015DD	PUSH 46DD9C	ASCII "idle"
00401657	PUSH 46DDA0	ASCII "idle"
00401718	PUSH 46DDA4	ASCII "idle"
00401734	PUSH 46DDA8	ASCII "idle"
00401750	PUSH 46DDAC	ASCII "idle"
00401773	PUSH 46DDB0	ASCII "idle"
00401792	PUSH 46DDB4	ASCII "idle"
004017A9	PUSH 46DDB8	ASCII "idle"
004017C6	PUSH 46DDBC	ASCII "idle"
004017E3	PUSH 46DDC0	ASCII "idle"
004017FE	PUSH 46DDC4	ASCII "idle"
00401815	PUSH 46DDC8	ASCII "sound"
00401837	PUSH 46DE00	ASCII "FLASH_LIGHT_ITEM_ROLLOVER"
00401846	PUSH 46DE1C	ASCII "STORY"
004019A8	PUSH 46DE44	ASCII "idle"
004019CB	PUSH 46DE4C	ASCII "active"
004019EE	PUSH 46DE54	ASCII "active"
00401D00	PUSH 46DE5C	ASCII "idle"

_Click OK and Ctrl + L, we will come

0056DEC6	PUSH 4A8598	ASCII "OPEN_KILL"
0056E062	ASCII "p~",0	
0056E1C3	MOV EDI,4A4F88	ASCII "GoOnline"
0056E215	PUSH 4A869C	ASCII "GetOfflineResponse"
0056E528	PUSH 4A8564	ASCII "out of memory"
0056E618	PUSH 4A867C	ASCII "LoadStatePool"
0056E929	DD unpacked.0056E92F	ASCII "4jW"
0056E92E	ASCII "%4jW",0	
0056EC22	ASCII "1(",0	
0056EE3E	ASCII "#\J<W",0	
0056F113	DD unpacked.0056F119	ASCII "mKW"
0056F117	ASCII ">PmKW",0	
0056F4D9	ASCII "hyqW",0	
0056F5BF	MOV EDI,4A8718	ASCII ".lcn"
0056F9EF	PUSH 4A86EC	ASCII "SetKey"
0057046C	PUSH 4A85D4	ASCII "TIMEOUT_KILL"
0057065E	ASCII "XZ"^W",0	
00570763	ASCII "-y",0	
00570A5C	ASCII "5cW",0	
00570D36	ASCII "5=W",0	
00570E39	ASCII "Kt"	
005713A8	PUSH 4A4F04	ASCII "rb"
0057145B	ASCII "-{",0	
005714E2	DD unpacked.0057A9B4	ASCII "PhX%W"
00571931	MOV ESI,4A85E4	ASCII "/stubinfo/shuffling"
00571AF9	ASCII "Nt"	

Dup _ Click on the yellow one to come

Address	Hex dump	Disassembly	Comment
0056F9E9	. 59	POP ECX	
0056F9EA	. C9	LEAVE	
0056F9EB	> EB 02	JMP SHORT 0056F9EF	0056F9EF
0056F9ED	FF	DB FF	
0056F9EE	24	DB 24	CHAR '\$'
0056F9EF	> 68 EC864A00	PUSH 4A86EC	ASCII "SetKey"
0056F9F4	. FF35 FDF95600	PUSH DWORD PTR DS:[56F9FD]	unpacked.0056E809
0056F9FA	. C3	RET	
0056F9FB	58	DB 58	CHAR 'X'
0056F9FC	5A	DB 5A	CHAR 'Z'
0056F9FD	. 09E85600	DD unpacked.0056E809	
0056FA01	. 83EC 20	SUB ESP,20	
0056FA04	> EB 02	JMP SHORT 0056FA08	0056FA08
0056FA06	FF	DB FF	
0056FA07	15	DB 15	
0056FA08	> 8D71 20	LEA ESI,DWORD PTR DS:[ECX+20]	
0056FA0B	. FF35 13FA5600	PUSH DWORD PTR DS:[56FA13]	unpacked.0057783D
0056FA11	. C3	RET	
0056FA12	E8	DB E8	
0056FA13	. 3D785700	DD unpacked.0057783D	
0056FA17	D0	DB D0	
0056FA18	. E8 4AEA0000	CALL 0057E467	0057E467

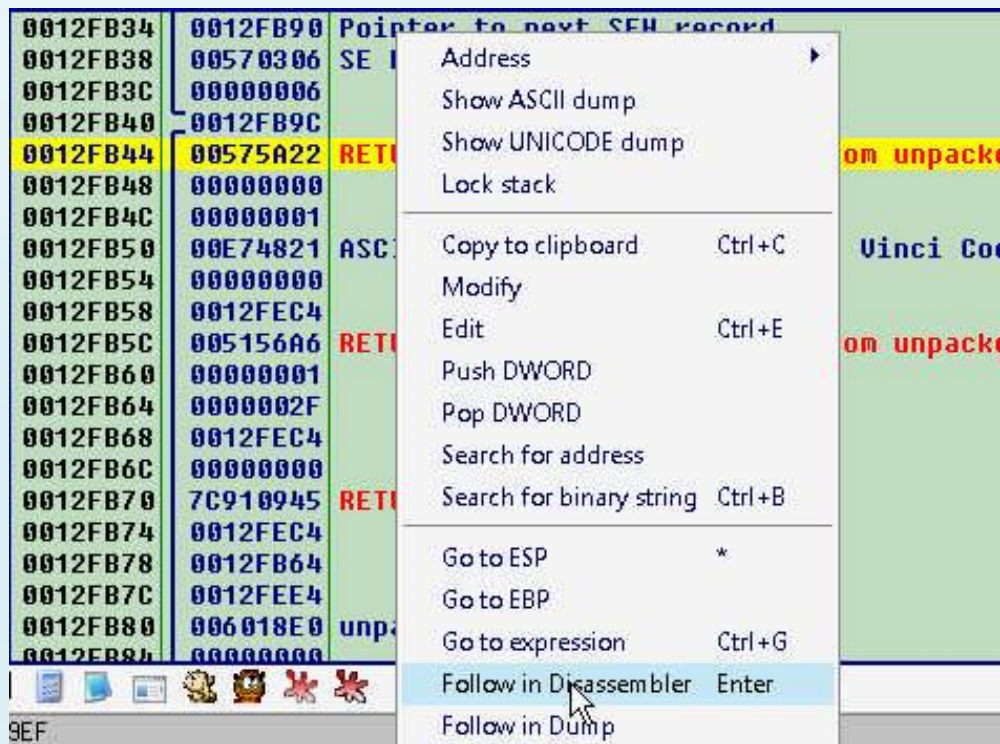
_ Press **F2** to Set Breakpoint 1 at 56F96F and press **F9** stop at Breakpoint Set. In the window you will see Stack as follows:

0012F9F4	0059C9F5	unpacked.0059C9F5
0012F9F8	0060AFA0	unpacked.0060AFA0
0012F9FC	00000000	
0012FA00	00000000	
0012FA04	00000000	
0012FA08	00000000	
0012FA0C	00000001	
0012FA10	00000000	
0012FA14	0012FE80	
0012FA18	00000001	
0012FA1C	0060AFA0	unpacked.0060AFA0
0012FA20	0012FE80	
0012FA24	FFFFFFFF	
0012FA28	0060AFA0	unpacked.0060AFA0

_ Scroll down until you see signs as follows

0012FB1C	00000000	
0012FB20	00E77900	
0012FB24	00000000	
0012FB28	00000000	
0012FB2C	00000000	
0012FB30	0057B9D5	RETURN to unpacked.0057B9D5 from unpacked.005EE794
0012FB34	0012FB90	Pointer to next SEH record
0012FB38	00570306	SE handler
0012FB3C	00000006	
0012FB40	0012FB9C	
0012FB44	00575A22	RETURN to unpacked.00575A22 from unpacked.00570484
0012FB48	00000000	
0012FB4C	00000001	
0012FB50	00E74821	ASCII "C:\Program Files\The Da Vinci Code\unpacked.exe"
0012FB54	00000000	
0012FB58	0012FEC4	
0012FB5C	005156A6	RETURN to unpacked.005156A6 from unpacked.00514F53
0012FB60	00000001	
0012FB64	0000002F	
0012FB68	0012FEC4	

_ Hehe ... this game coming gòi targets ... click to select the image



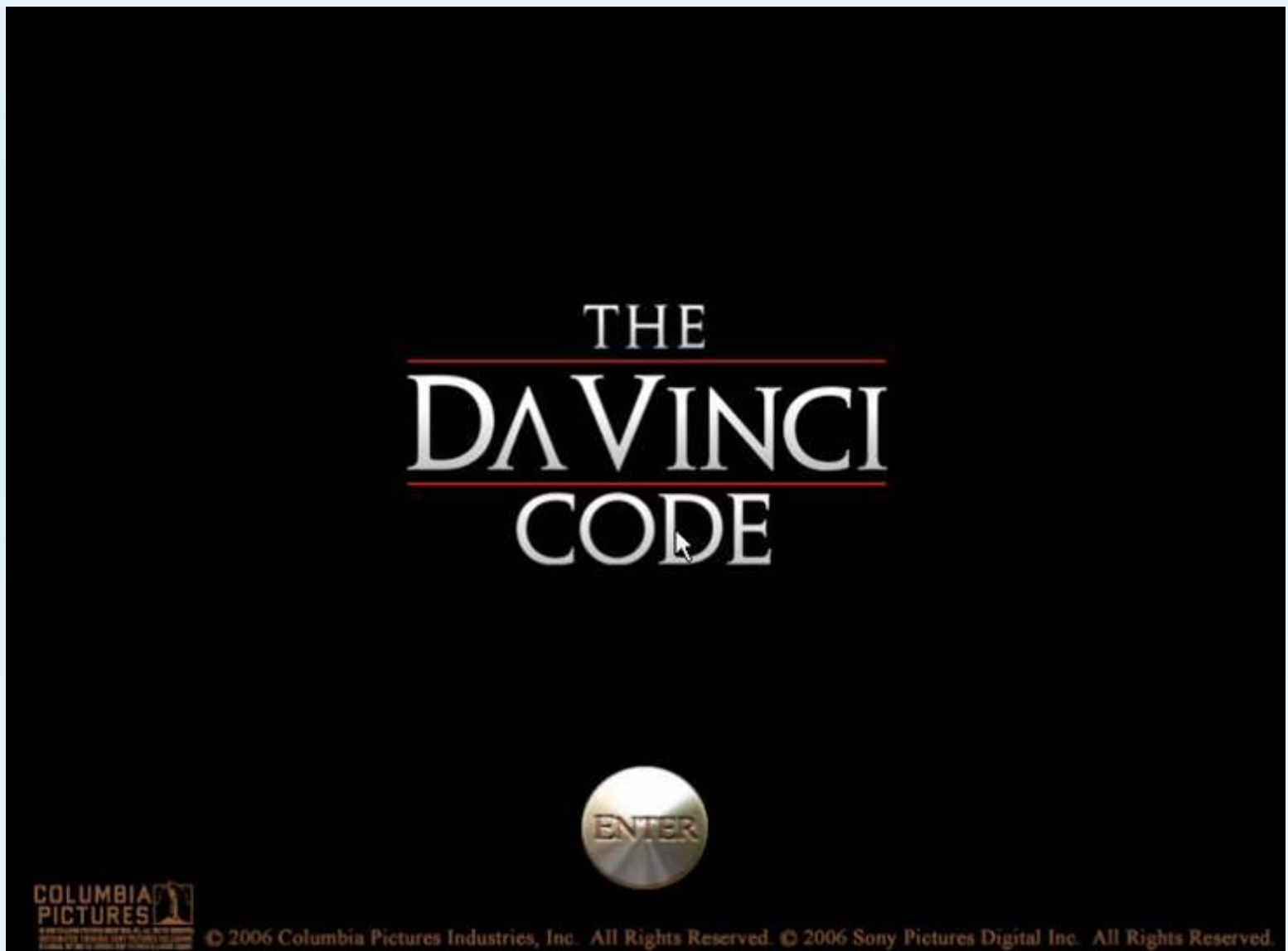
_va to us here

Address	Hex dump	Disassembly	Comment
00575A1A	. C3	RET	
00575A1B	. 8BCE	MOV ECX,ESI	
00575A1D	. E8 62AAFFFF	CALL 00570484	00570484
00575A22	EB 03	JMP SHORT 00575A27	00575A27
00575A24	EB	DB EB	
00575A25	F8	DB F8	
00575A26	B4	DB B4	
00575A27	E9 2CC8FFFF	JMP 00572258	00572258
00575A2C	EB 02	JMP SHORT 00575A30	00575A30
00575A2E	80	DB 80	
00575A2F	. FA	CLI	
00575A30	E9 11C3FFFF	JMP 00571D46	00571D46
00575A35	8D	DB 8D	
00575A36	EB 03	JMP SHORT 00575A3B	00575A3B
00575A38	50	DB 50	CHAR 'P'
00575A39	46	DB 46	CHAR 'F'
00575A3A	7E	DB 7E	CHAR '~'

_ You see the command **JMP** order **Call 1** ... haha ... and that is **Magic Call** ... Call you submit this order is also the time **ActiveMARK** officially dust.

Address	Hex dump	Disassembly	Comment
00575A1A	. C3	RET	
00575A1B	. 8BCE	MOV ECX,ESI	
00575A1D	90	NOP	
00575A1E	90	NOP	
00575A1F	90	NOP	
00575A20	90	NOP	
00575A21	90	NOP	
00575A22	.. EB 03	JMP SHORT 00575A27	00575A27
00575A24	EB	DB EB	
00575A25	F8	DB F8	
00575A26	B4	DB B4	
00575A27	>^ E9 2CC8FFFF	JMP 00572258	00572258
00575A2C	.. EB 02	JMP SHORT 00575A30	00575A30
00575A2E	80	DB 80	

_ Save the set and 1 for the name of the **gas-Unpacked Cracked.exe**. ... Test considered the more sướng again
NAG no longer run into ... game. ... Crack Done



_ Almost all of the game is using Protect ActiveMARK have to resolve as on you from any work that meat ... There is also Amloader For ActiveMARK Tool can help you resolve the many that they must stand out we stop and you can refer to



Gre e Ts F O italy u l t: *C om p ut _ An e r l e g, e Zombi, M oo n b by a, c H a grape, B e n i n a k i n e m a n o a w r, o i Z, D x u e, M e rc, i l e g HT pho where x, c k ybo Tri italy, T a k a d a i a midiot, the light o e ni x, t h e i nth a n d e n a n dyou ...!*

N ha Tr a n g, italy 2 à The 4th Asia ng5 year 200 6

Why N ot Bar

I. Introduction:

Hi Newbie the brother of the first in TUT we learn 2 Tools help us to quickly unpack the Soft Pack with **Armadillo**. In this tut we continue to make 2 more familiar New Tool is **dilloDIE 1.4** and **ArmStripperv0.1beta2**

dilloDIE 1.4: the evaluation of the child Good Tools this operation because it is quite stable, unpack the lot Target

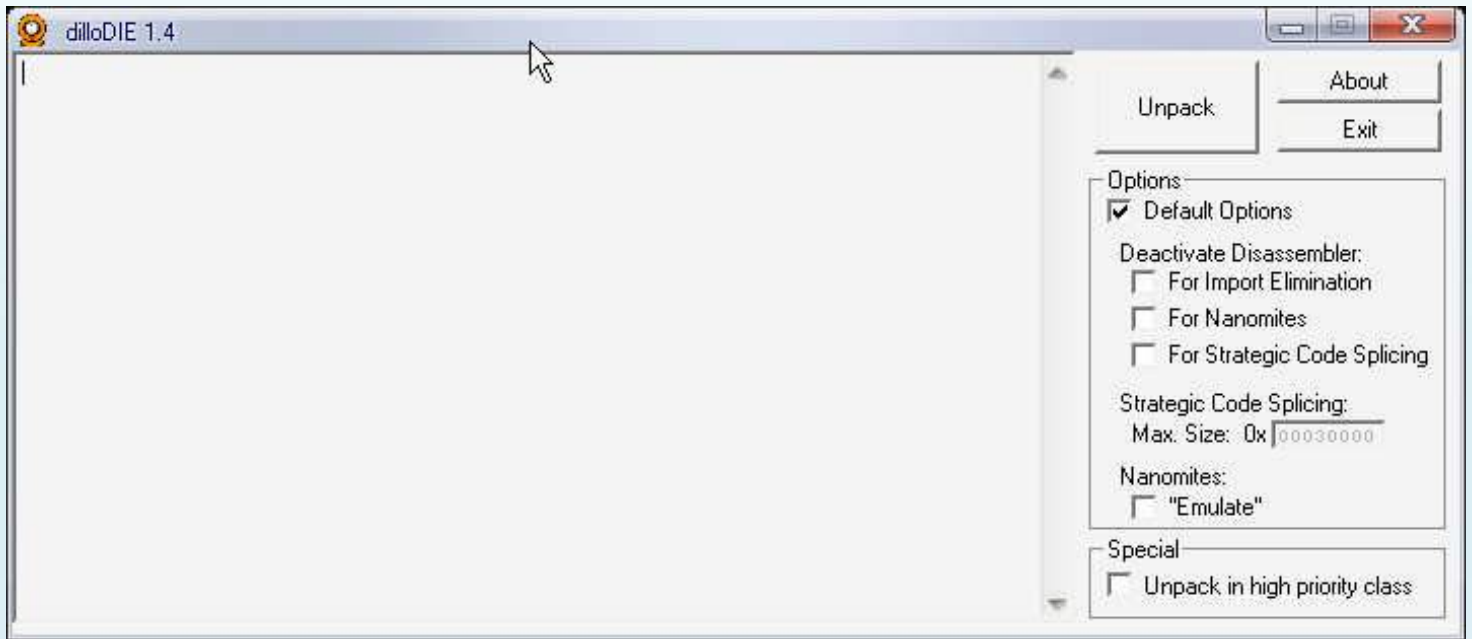
supported features training:

Standard Features

ugblocker Compiler IMEI

Names

Import Elimination



Armadillostripper Belgium T-H_c@k: Tools is in the process of development should be at the time not unpack

version 0.1 to be a 2

- Added in fr DP protect options

- Added de splicing illker (er v italy unst a BL e)

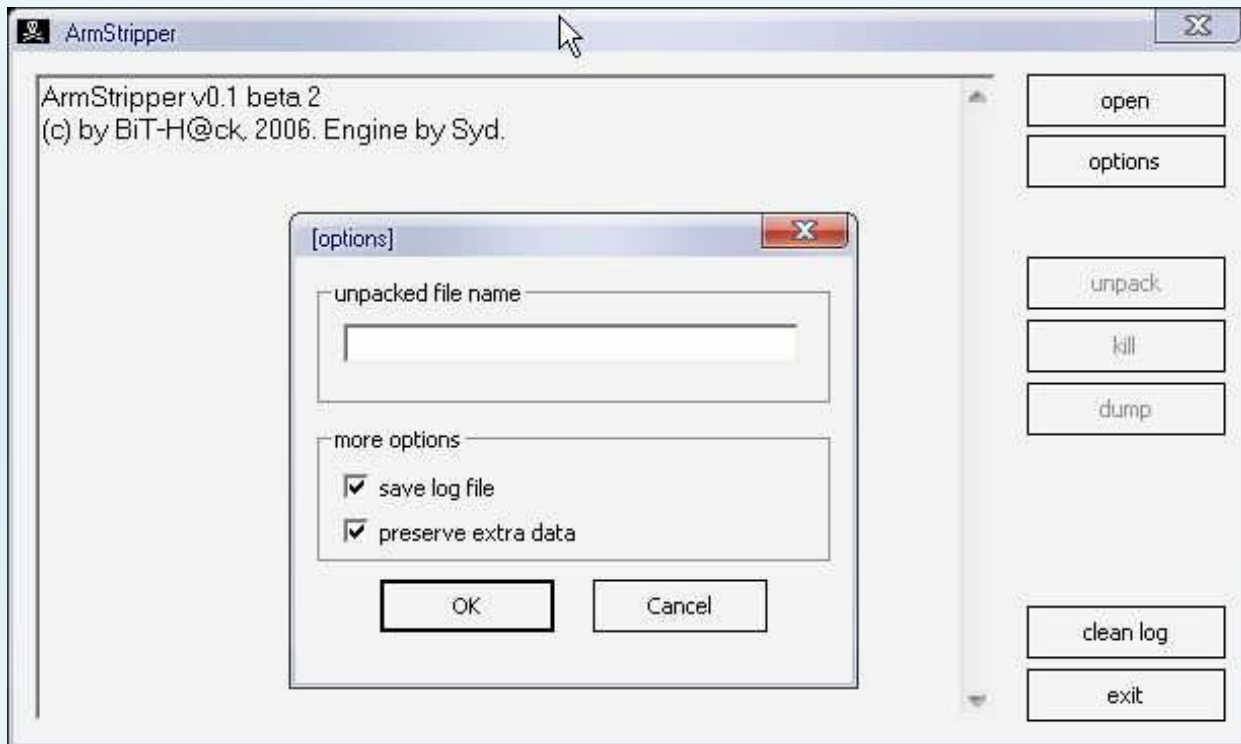
- Added de bugblockers support

- Fix many bugs

* Requirements:

-stripper works only under and k2 XP, also you must have inistradmator's rights;

- The ere will be now in 9 X versions;



II. Tools:

• Toolc and gùn

- *ARM dillo a F i n d P r e d o t e c t V 1. 2*
- *ARM dillostrip a p e r B y t h e T - H @ _ c k a b o u t r s i o n 0. 1 e 2 i s*
- *D i l l o I D E 1. 4*
- *P I D E 0. 9 4*
- *C F F E x p l o r r e I V*
- *O l l i t a l y D G B 1. 1*

• Tar of GE:

Ea s i t a l y C D - D A E x o f t h e C t o t h a t o f P r e s s i o n a l v 9 . 1 . 1 u i l D 3

H o m e p a g e : [HTtp://ww.w.Kp_o_i_o_s_o_f_t._C_o_m/](http://www.Kpoisoftware.com/)



UnPackM_Areadm_eill o4.40



III. Un packing:

T U T # 1: S using enough of gDill I D E O 1.4

_ Use **PEiD 0.94** scans have been considered Soft pack and pack or do any kind?

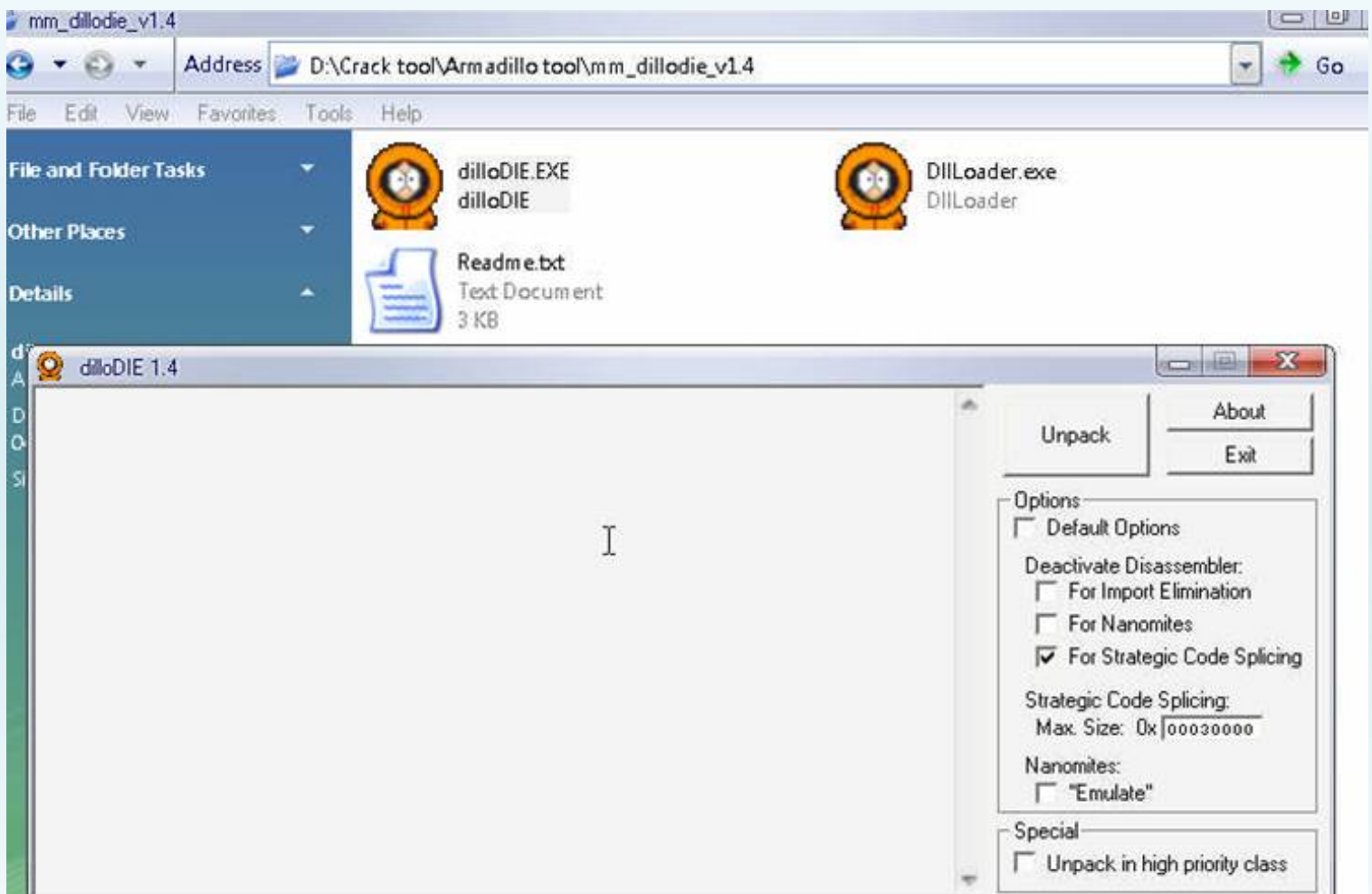


_ To know exactly in the Pack File **Protection Options** do we use **Armadillo FindProtected v1.2**



_ This is Hehe this has *Debug-Blocker*, *Enable Strategic Code Splicing* *Standard Protection*.

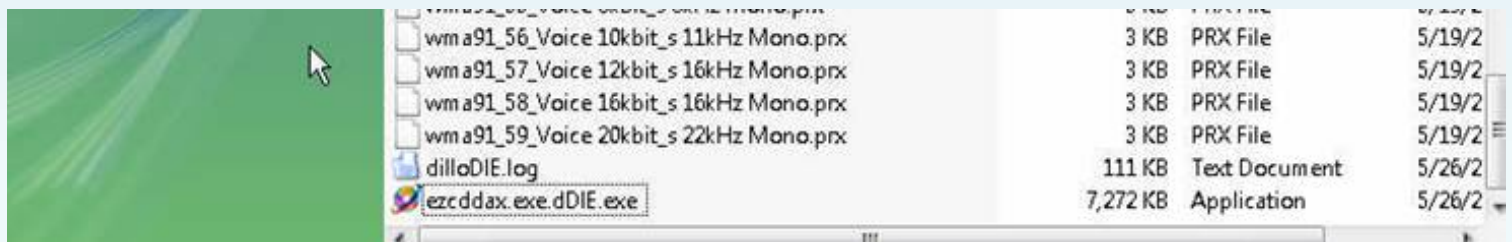
_ Time is running **DilloDIE 1.4** and below nhuhinh



_ The file and select "**ezcddax.exe**" and now they sit waiting **DilloDIE** run òu self treatment ...



_ Hahaha ... This is done When finished ... will create 2 files



_ File dilloDIE.log is recorded over the handling of it is "ezcdax.exe.dDIE.exe" File is already unpack. Run test considered stars



_ _Unpack Done ... Devay but also if you want to reduce the amount unpack the files using **CFF Explorer** to delete some of the **Section** is the Arma

CFF Explorer IV - by Ntoskrnl - [EZCDDA~1.EXE]

File Settings ?

File: EZCDDA~1.EXE

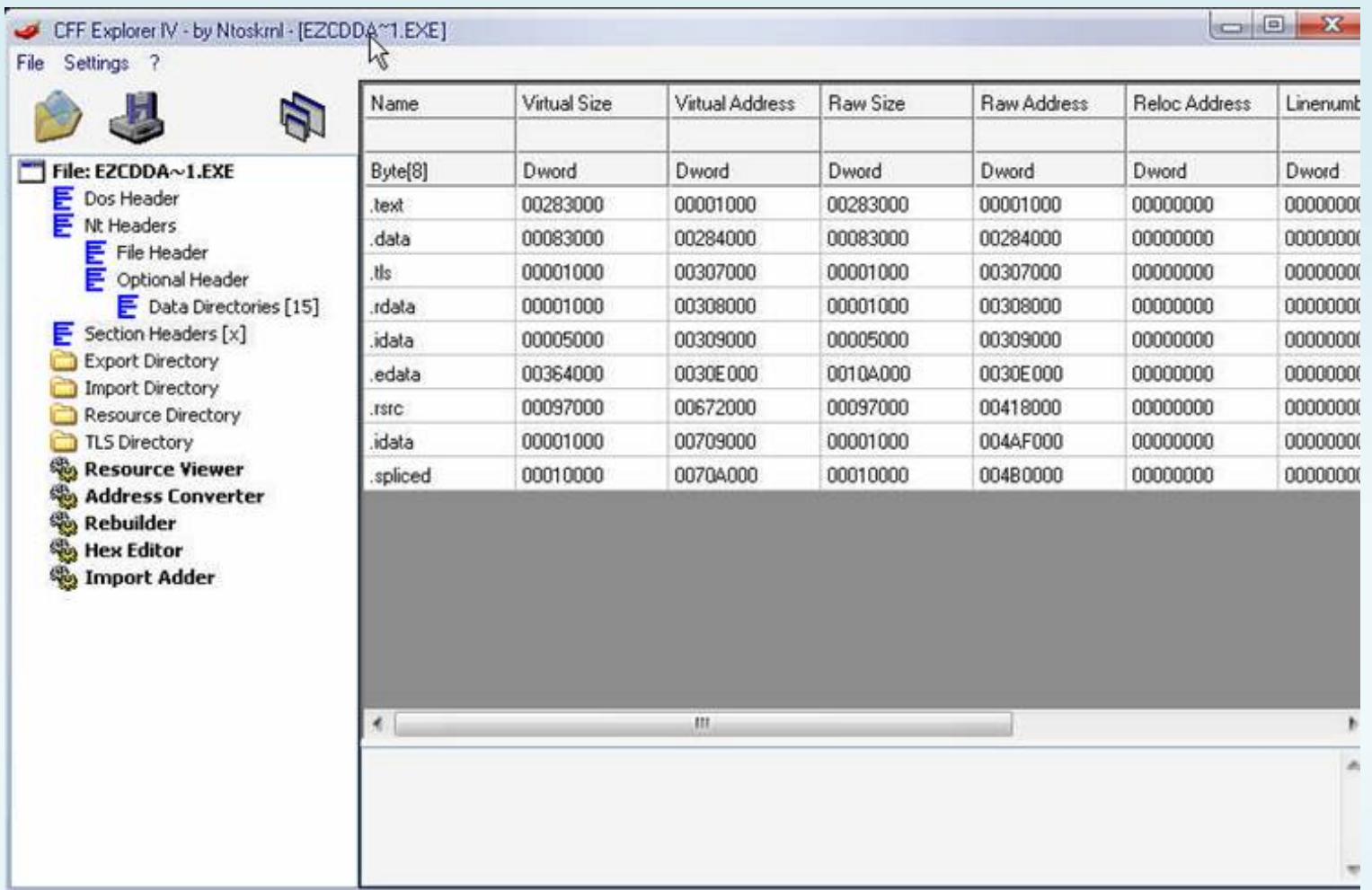
- Dos Header
- Nt Headers
- File Header
- Optional Header
- Data Directories [15]
- Section Headers [x]
- Export Directory
- Import Directory
- Resource Directory
- TLS Directory
- Resource Viewer
- Address Converter
- Rebuilder
- Hex Editor
- Import Adder

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Lin
000004D8	000004E0	000004E4	000004E8	000004EC	000004F0	000004F4	000004F8	000
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Wd
.text	00283000	00001000	00283000	00001000	00000000	00000000	0000	000
.data	00083000	00284000	00083000	00284000	00000000	00000000	0000	000
.tls	00001000	00307000	00001000	00307000	00000000	00000000	0000	000
.rdata	00001000	00308000	00001000	00308000	00000000	00000000	0000	000
.idata	00005000	00309000	00005000	00309000	00000000	00000000	0000	000
.edata	0010A000	0030E000	0010A000	0030E000	00000000	00000000	0000	000
.reloc	0002A000	00418000	0002A000	00418000	00000000	00000000	0000	000
.text1	0005d200	00442000	00050000	00442000	00000000	00000000	0000	000
.adata	00010000	00492000	00010000	00492000	00000000	00000000	0000	000
.data1	00020000	004A2000	00020000	004A2000	00000000	00000000	0000	000
.pdata	001B0000	004C2000	001B0000	004C2000	00000000	00000000	0000	000
.rsrc	00097000	00672000	00097000	00672000	00000000	00000000	0000	000
.idata	00001000	00709000	00001000	00709000	00000000	00000000	0000	000
.spliced	00010000	0070A000	00010000	0070A000	00000000	00000000	0000	000

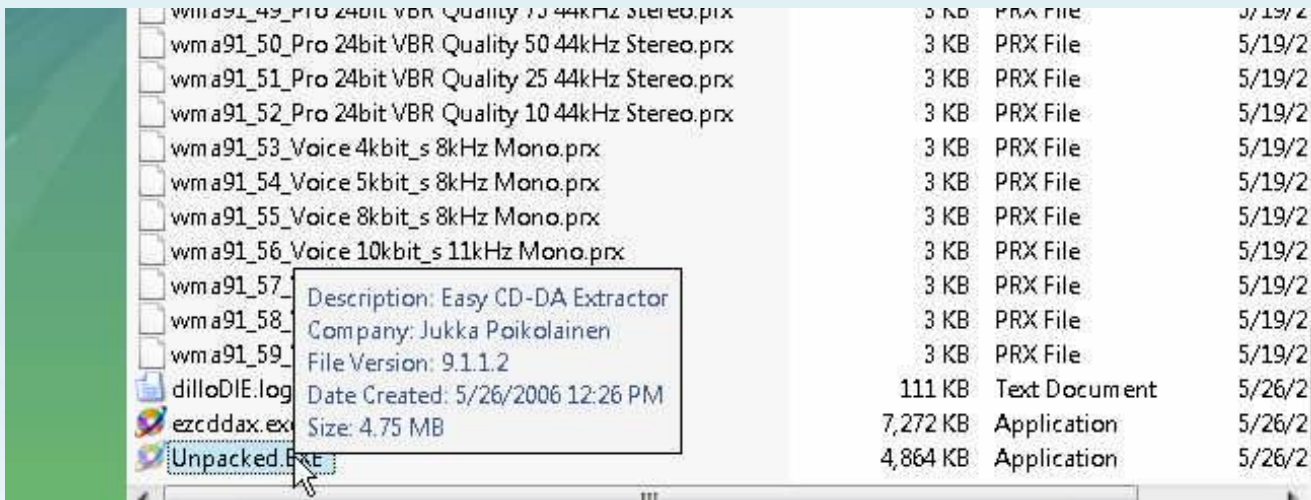
This section contains:

Import Directory: 00709000

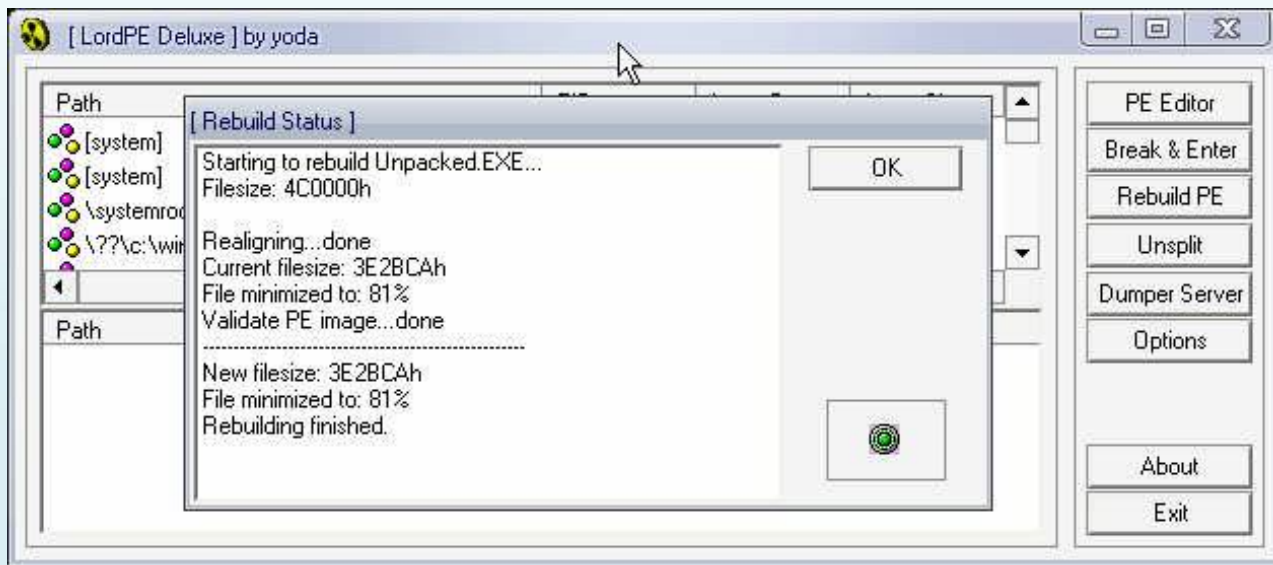
_S a k u hiXoa



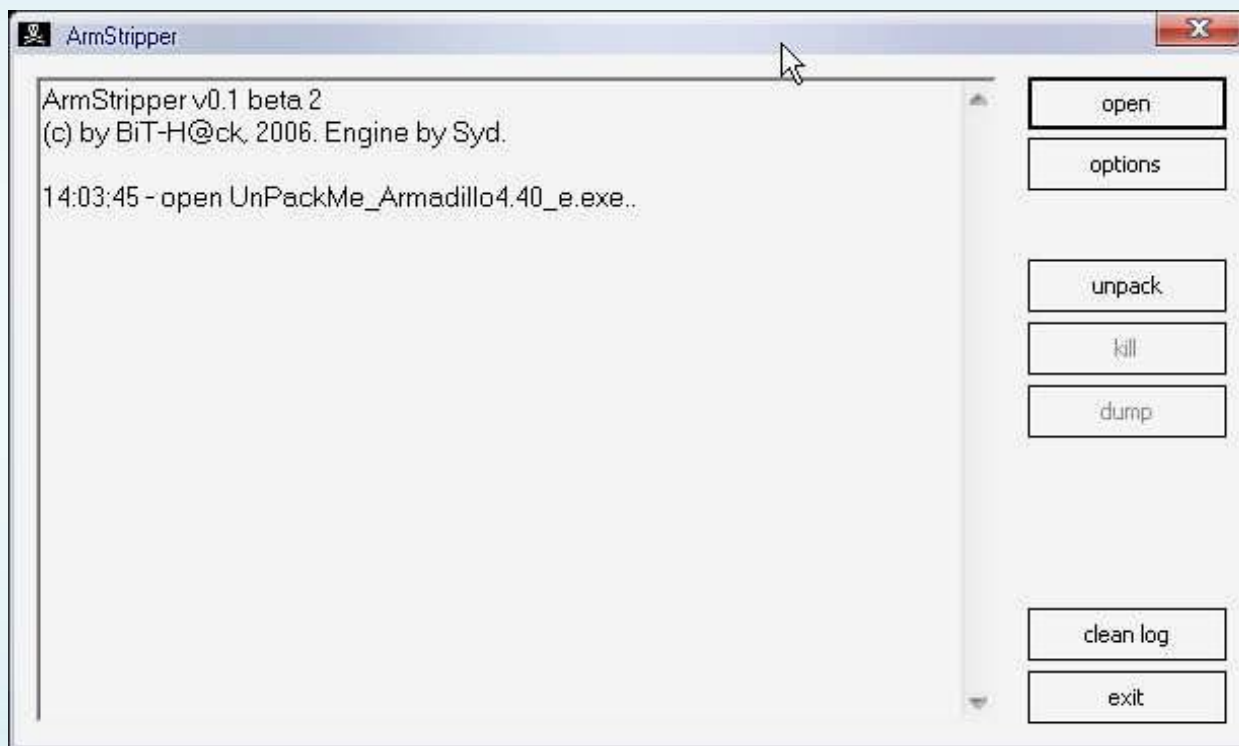
_ Save with a different name, such as **Unpacked.exe**



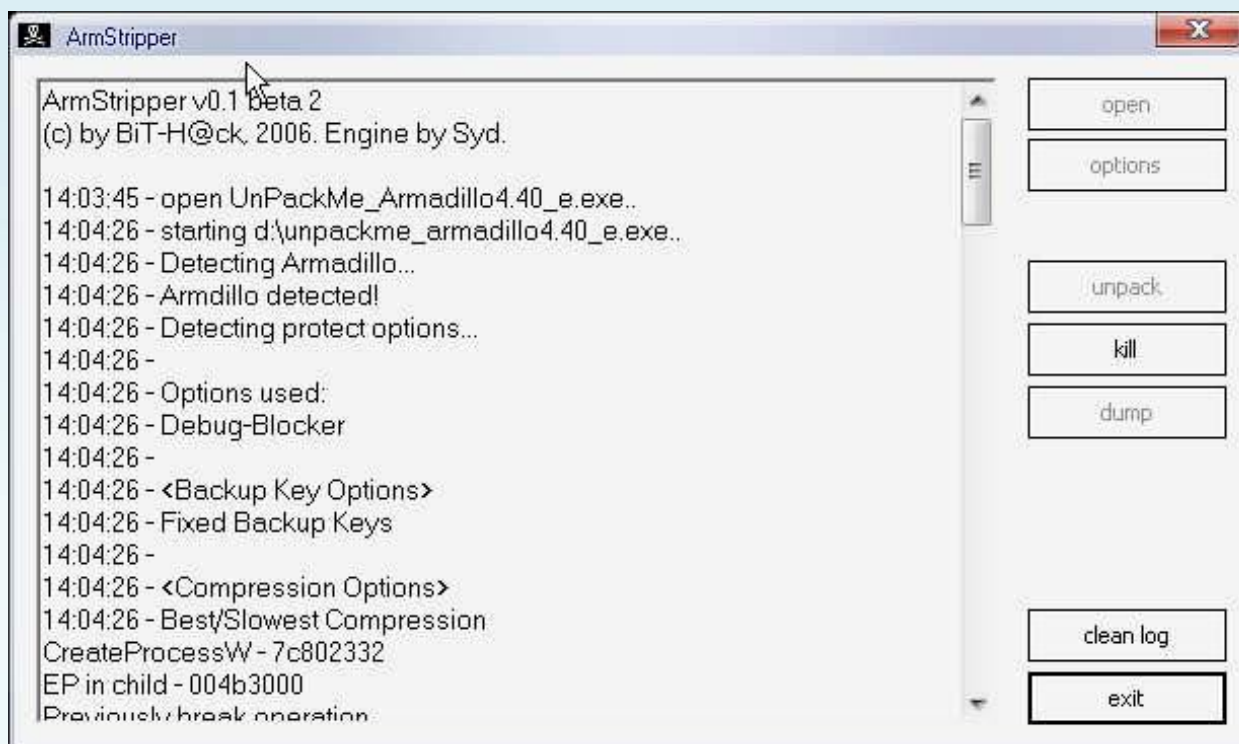
_ Test still tasty healthy, if still not fill you can give it a little more to **rebuild the PE LordPE**



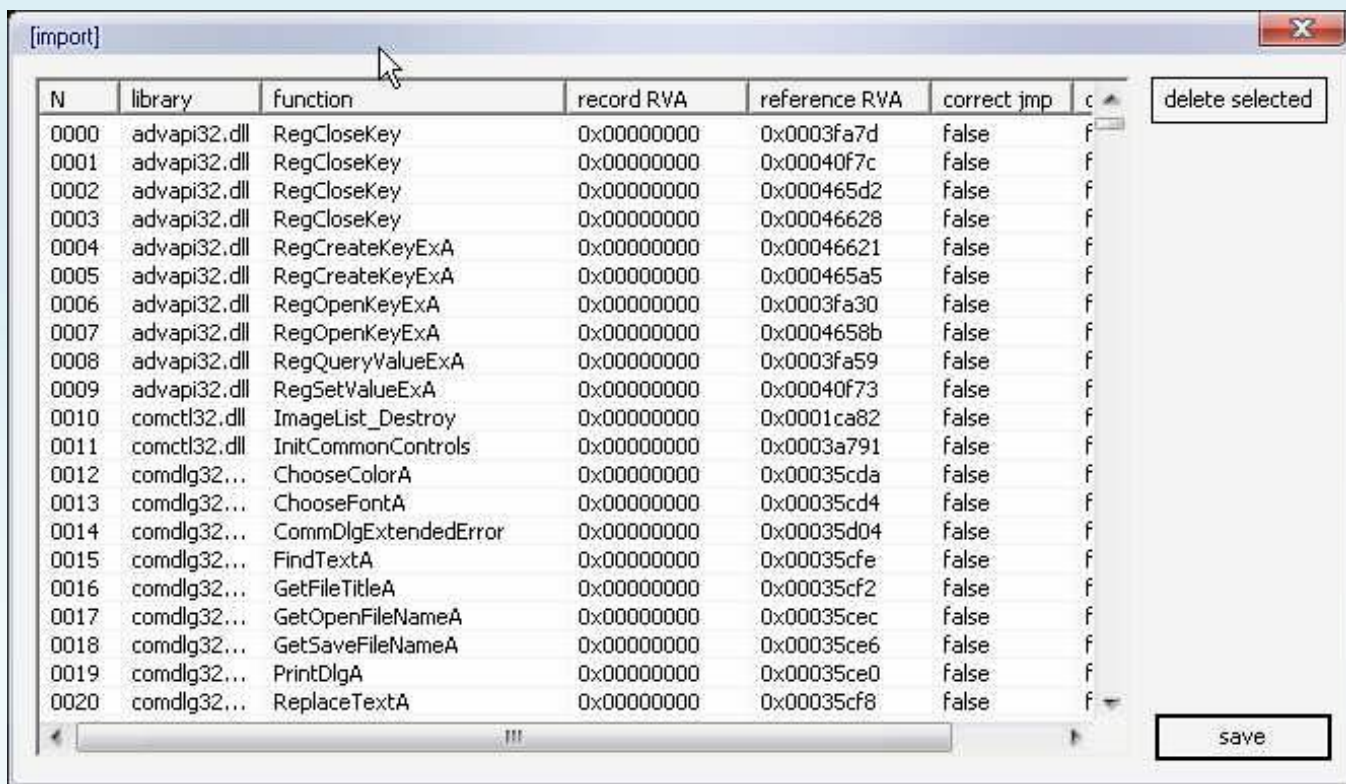
wma91_54_Voice 5kbit_s 8kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_55_Voice 8kbit_s 8kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_56_Voice 10kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_57_Voice 12kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_58_Voice 16kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_59_Voice 20kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_60_Voice 24kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_61_Voice 32kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_62_Voice 48kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_63_Voice 64kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_64_Voice 96kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_65_Voice 128kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_66_Voice 192kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_67_Voice 256kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_68_Voice 384kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_69_Voice 512kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_70_Voice 768kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_71_Voice 1024kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_72_Voice 1536kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_73_Voice 2048kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_74_Voice 3072kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_75_Voice 4096kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_76_Voice 6144kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_77_Voice 8192kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_78_Voice 12288kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_79_Voice 16384kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_80_Voice 24576kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_81_Voice 32768kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_82_Voice 49152kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_83_Voice 65536kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_84_Voice 98304kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_85_Voice 131072kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_86_Voice 196608kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_87_Voice 262144kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_88_Voice 393216kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_89_Voice 524288kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_90_Voice 786432kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_91_Voice 1048576kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_92_Voice 1572832kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_93_Voice 2097088kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_94_Voice 3145632kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_95_Voice 4194176kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_96_Voice 6291264kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_97_Voice 8388352kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_98_Voice 12582528kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_99_Voice 16776696kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_100_Voice 25165032kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_101_Voice 33553368kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_102_Voice 50330048kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_103_Voice 67106728kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_104_Voice 100660096kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_105_Voice 134213472kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_106_Voice 201320256kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_107_Voice 268427040kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_108_Voice 402640608kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_109_Voice 536854176kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_110_Voice 805281280kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_111_Voice 1073708384kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_112_Voice 1610562560kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_113_Voice 2147416704kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_114_Voice 3221125120kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_115_Voice 4294833536kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_116_Voice 6442250240kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_117_Voice 8589666944kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_118_Voice 12884500480kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_119_Voice 17179334016kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_120_Voice 25769000960kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_121_Voice 34358667904kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_122_Voice 51542502400kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_123_Voice 68726336896kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_124_Voice 102689664000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_125_Voice 136652992000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_126_Voice 205039360000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_127_Voice 273425728000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_128_Voice 409639372800kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_129_Voice 545853017600kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_130_Voice 818779526400kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_131_Voice 1091706035200kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_132_Voice 1637559040000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_133_Voice 2183412044800kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_134_Voice 3275116800000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_135_Voice 4366821568000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_136_Voice 6550232320000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_137_Voice 8733643072000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_138_Voice 13101464320000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_139_Voice 17469285568000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_140_Voice 26203928320000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_141_Voice 34938571072000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_142_Voice 52307840000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_143_Voice 69677110080000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_144_Voice 104515660800000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_145_Voice 139354211200000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_146_Voice 208031321600000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_147_Voice 276708432000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_148_Voice 415062656000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_149_Voice 553416880000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_150_Voice 830125312000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_151_Voice 1106833744000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_152_Voice 1660250368000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_153_Voice 2213666992000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_154_Voice 3320500224000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_155_Voice 4427333456000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_156_Voice 6640999680000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_157_Voice 8854665920000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_158_Voice 13281998720000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_159_Voice 17709331520000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_160_Voice 26563998720000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_161_Voice 35418665920000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_162_Voice 53127998720000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_163_Voice 70837331520000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_164_Voice 106255998720000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_165_Voice 141674659200000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_166_Voice 212511998720000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_167_Voice 283349315200000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_168_Voice 425023699200000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_169_Voice 566698073600000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_170_Voice 850047107200000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_171_Voice 1133396140800000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_172_Voice 1700094272000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_173_Voice 2266792403200000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_174_Voice 3400188800000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_175_Voice 4533585200000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_176_Voice 6800377600000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_177_Voice 9067170000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_178_Voice 13600704000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_179_Voice 18134238000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_180_Voice 27201830400000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_181_Voice 35869422800000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_182_Voice 53703776000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_183_Voice 71538128000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_184_Voice 107307136000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_185_Voice 143076144000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_186_Voice 214614272000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_187_Voice 286152400000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_188_Voice 429228544000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_189_Voice 572304688000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_190_Voice 858462080000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_191_Voice 1144619488000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_192_Voice 1716929280000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_193_Voice 2289239040000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_194_Voice 3433858560000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_195_Voice 4578478080000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_196_Voice 6867711360000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_197_Voice 9156944640000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_198_Voice 13735418880000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_199_Voice 18313893120000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_200_Voice 27467786240000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_201_Voice 36621679360000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_202_Voice 54926028800000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_203_Voice 73230378240000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_204_Voice 109846144000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_205_Voice 146461910400000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_206_Voice 219692883200000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_207_Voice 292923856000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_208_Voice 439385792000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_209_Voice 585847728000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_210_Voice 878771136000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_211_Voice 1171694544000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_212_Voice 1757541888000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_213_Voice 2343389232000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_214_Voice 3516778464000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_215_Voice 4690167696000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_216_Voice 6989602880000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_217_Voice 9289038080000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_218_Voice 13933670400000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_219_Voice 18578302880000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_220_Voice 27872655360000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_221_Voice 37167007840000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_222_Voice 55000960000000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_223_Voice 73834912000000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_224_Voice 110750720000000000000000000kbit_s 11kHz Mono.prx	3 KB	PRX File	5/19/2
wma91_225_Voice 147666534400000000000000000kbit_s 11kHz Mono.prx	3 KB		



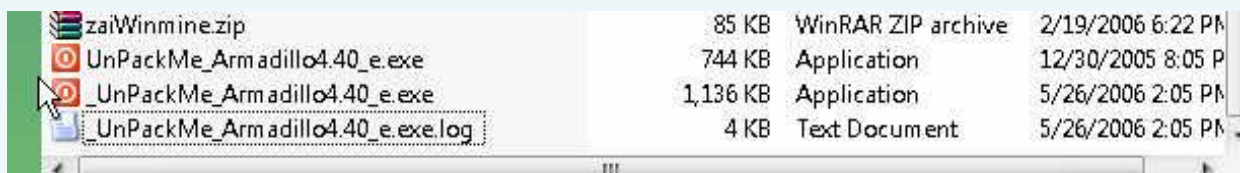
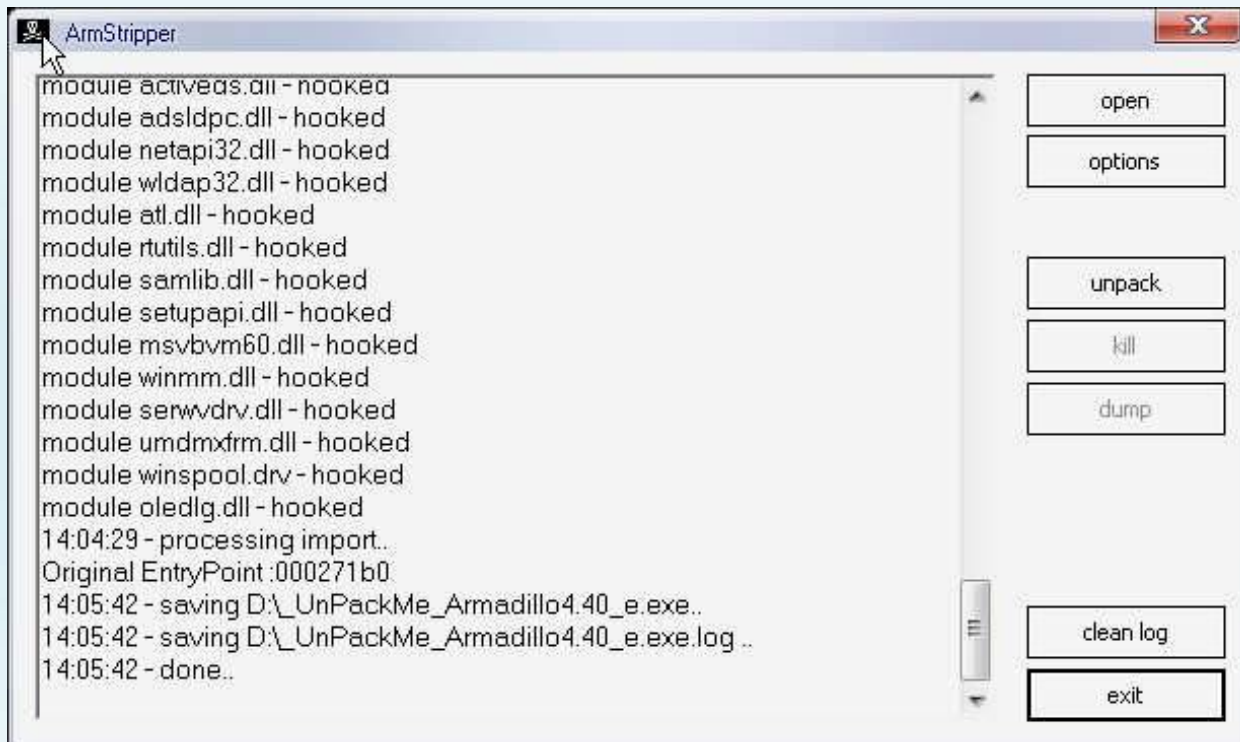
_ Click **unpack**



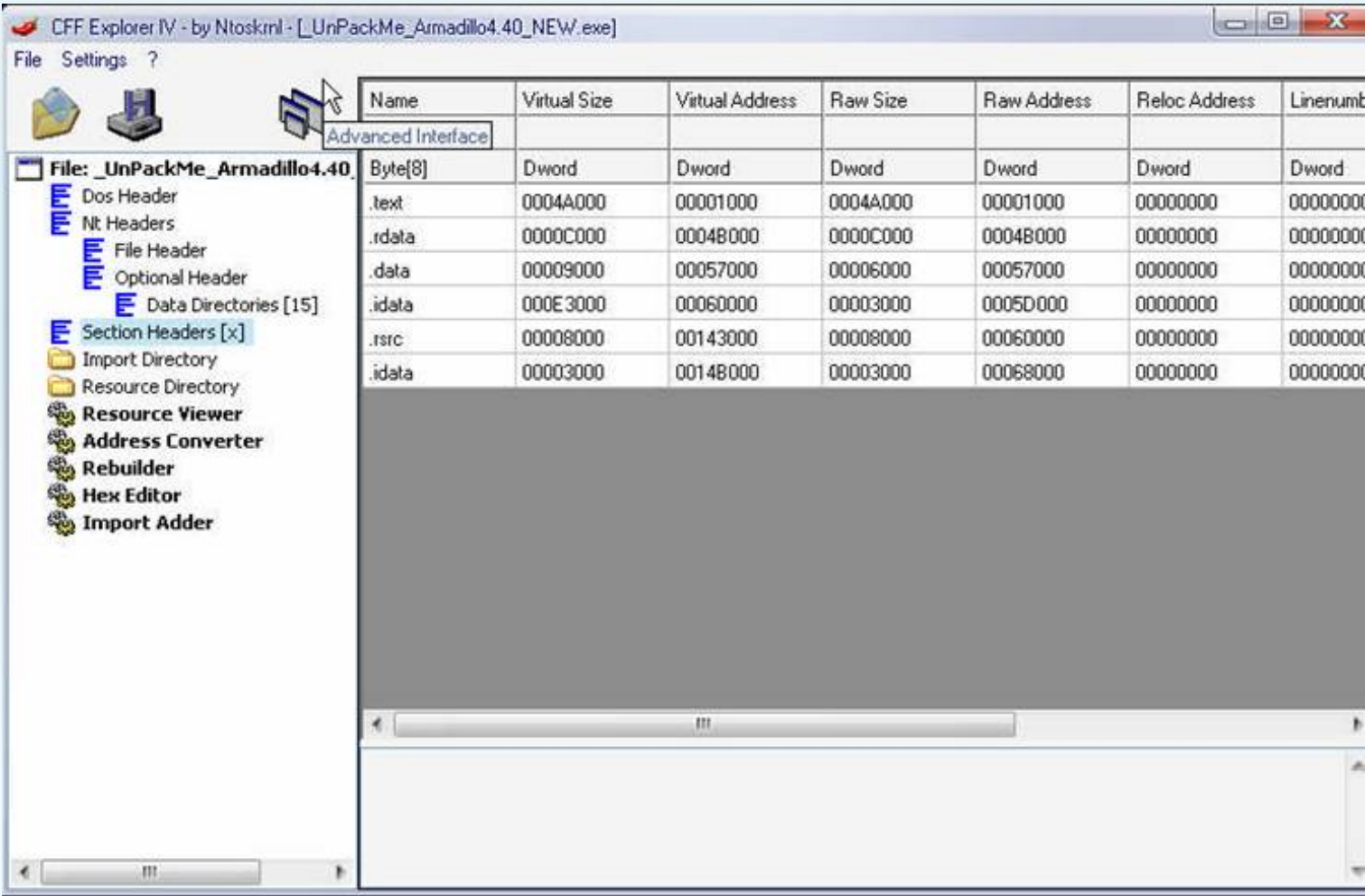
_va the **Import** window



_ Click **Save** is completed



- _ Test File and File _UnPackMe_Armadillo4.40_e.exe good run Unpack Done ..!!!
- _ Delete the Section legacy



zaiWinmine.zip	85 KB	WinRAR ZIP archive	2/19/2006 6:22 PM
UnPackMe_Armadillo4.40_e.exe	744 KB	Application	12/30/2005 8:05 PM
UnPackMe_Armadillo4.40_e.exe	744 KB	Application	5/26/2006 2:05 PM
UnPackMe_Armadillo4.40_e.exe	744 KB	Application	5/26/2006 2:05 PM
UnPackMe_Armadillo4.40_e.exe	744 KB	Application	5/26/2006 2:05 PM
UnPackMe_Armadillo4.40_NEW.exe	428 KB	Application	5/26/2006 2:09 PM

_Bye You ... appointment in the tut ...

GREELFTs italy Ou t: C om p ut _A n e r l e g, e Zombi, M oo n b y a, c H a grape, B e n i n a k i n e m a n o a w
r, o i Z, D x u e, M e rc, the i c k y b o italy, T a k a d a i a m i d i o t, H T l i g h p o e w h e r e x, and e n t h a n d i n e, a n d
you ...!

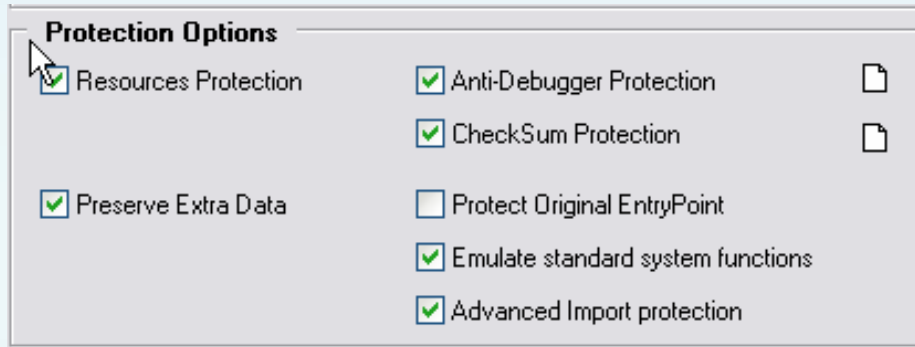
N A T h a n g, the à italy 2 6 t h a n n g 5 c l i c k 2 0 0 6

W h y N o t B a r

UnipACK the A S Prote CT2.3 S KEbuild06th 2 6

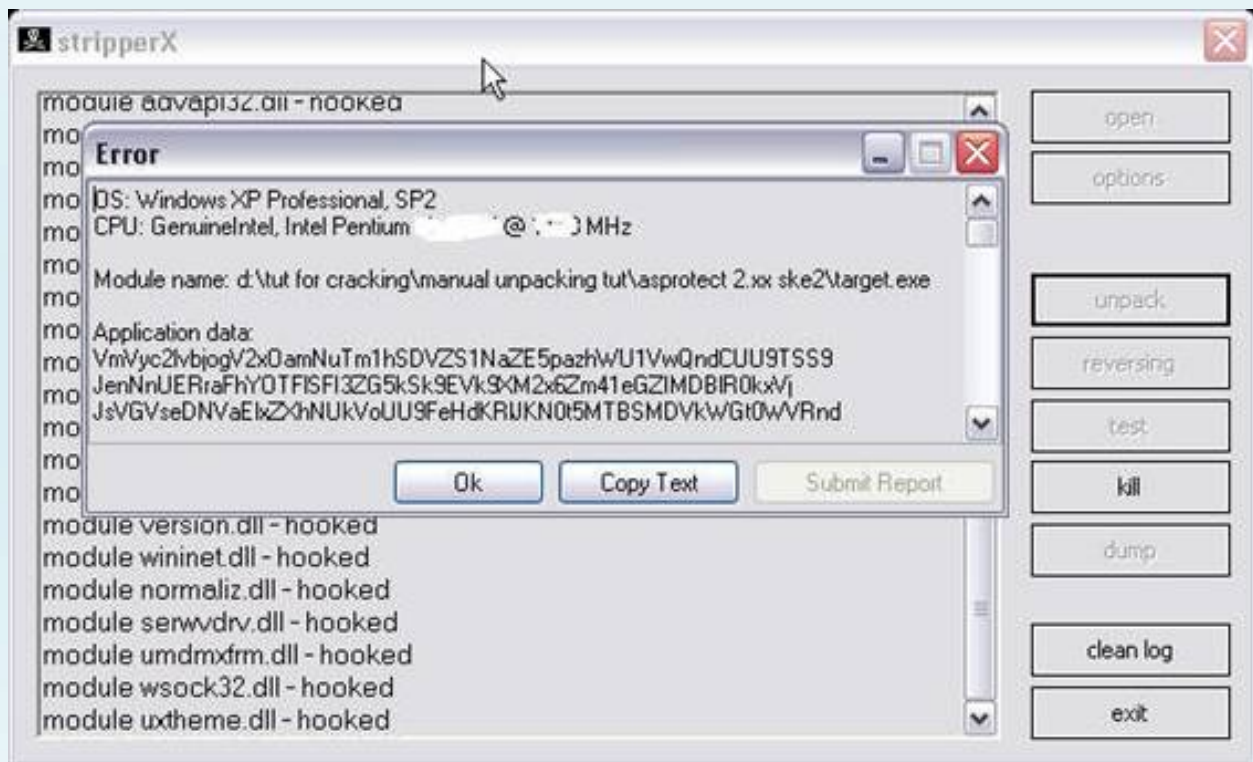
I-- Print d o t r u c t i o n :

Today, his children will learn how to unpack ASProtect 2.3 SKE Protection Options for the following:



And hope will continue to form Protect Original EntryPoint

_ With version 2.3 SKE ASProtect the tool Stripper 2:13 unpack not be



II - T o o l s & T a g e r t :

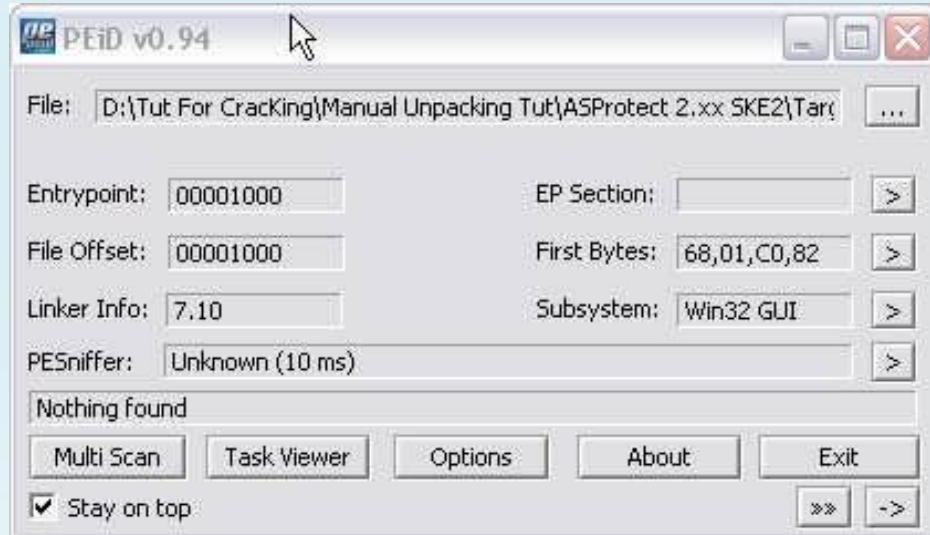
- T o o L and P l u g i n c a n d ú n g :

- **O LLY DBG 1:10**
- **Pe ID 0.94 & V e r A 0:15 plugin**
- **I mp O R T R E C 1.6f**
- **C F F E xpl or e r V**

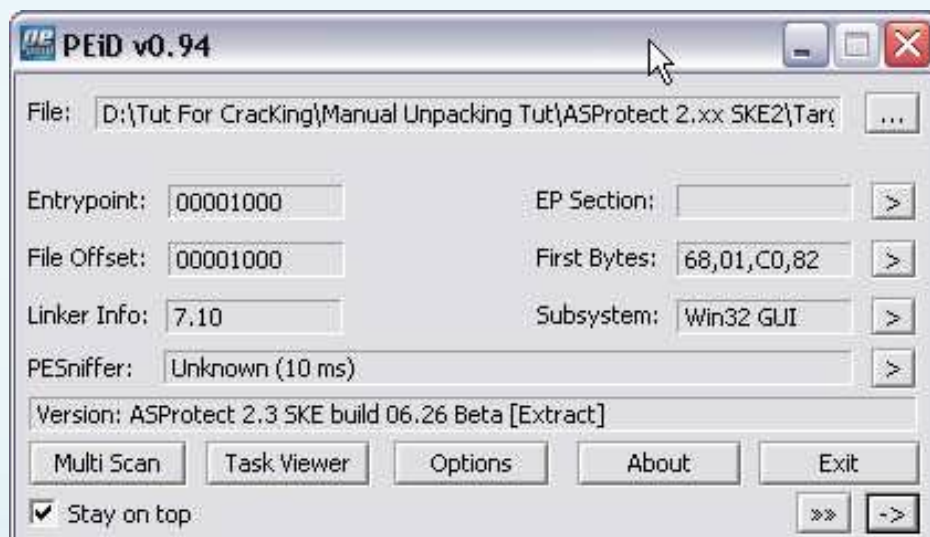
- **One g e r t: *K è m t h e t o u t***

III - U n p a c k i n g

_ D ù n I D P E g 0.94 s intervention in a g e r t



_ Enough V e n g r a p 0:15 lugin



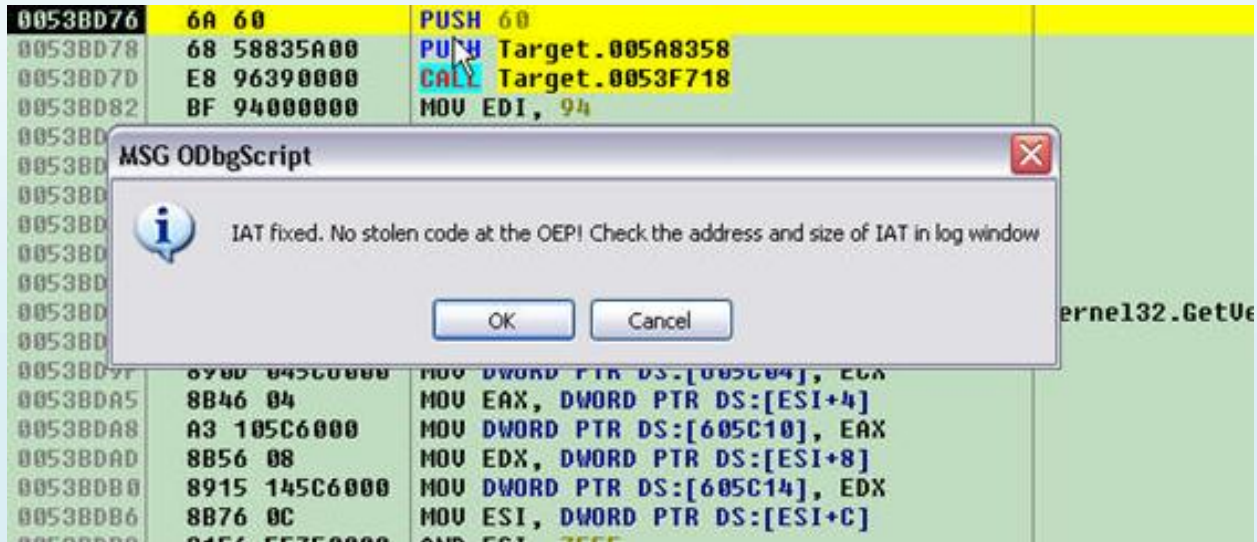
_ Lo T a d g e r t to OllyDBG

00401000	68 01C08200	PUSH Target.0082C001	
00401005	68 0B104000	PUSH Target.0040100B	
0040100A	C3	RETN	
0040100B	C3	RETN	
0040100C	2C CB	SUB AL, 0CB	
0040100E	F2:	PREFIX REPNE:	Superfluous pr
0040100F	815F 59 7E857F	SBB DWORD PTR DS:[EDI+59], CF7F857E	
00401016	D9AE 3DD8919D	FLDCW WORD PTR DS:[ESI+9D91DB3D]	
0040101C	95	XCHG EAX, EBP	
0040101D	37	AAA	
0040101E	9B	WAIT	
0040101F	9E	SAHF	
00401020	8760 BB	XCHG DWORD PTR DS:[EAX-45], ESP	
00401023	BC 29E8166A	MOV ESP, 6A16E829	
00401028	7C 23	JL SHORT Target.0040104D	
0040102A	E5 6F	IN EAX, 6F	I/O command
0040102C	93	XCHG EAX, EBX	
0040102D	3D FC1EE5E9	CMP EAX, E9E51EFC	

_O Ago they used Sscript "Aspr2.XX_IATfixer_v2.2s.osc" to Fix IAT and to OEP. You can if you prefer manual, reference tut "Aspro_SKE_211_Esyst_28ingl_29" by doctors ToolCracking translated

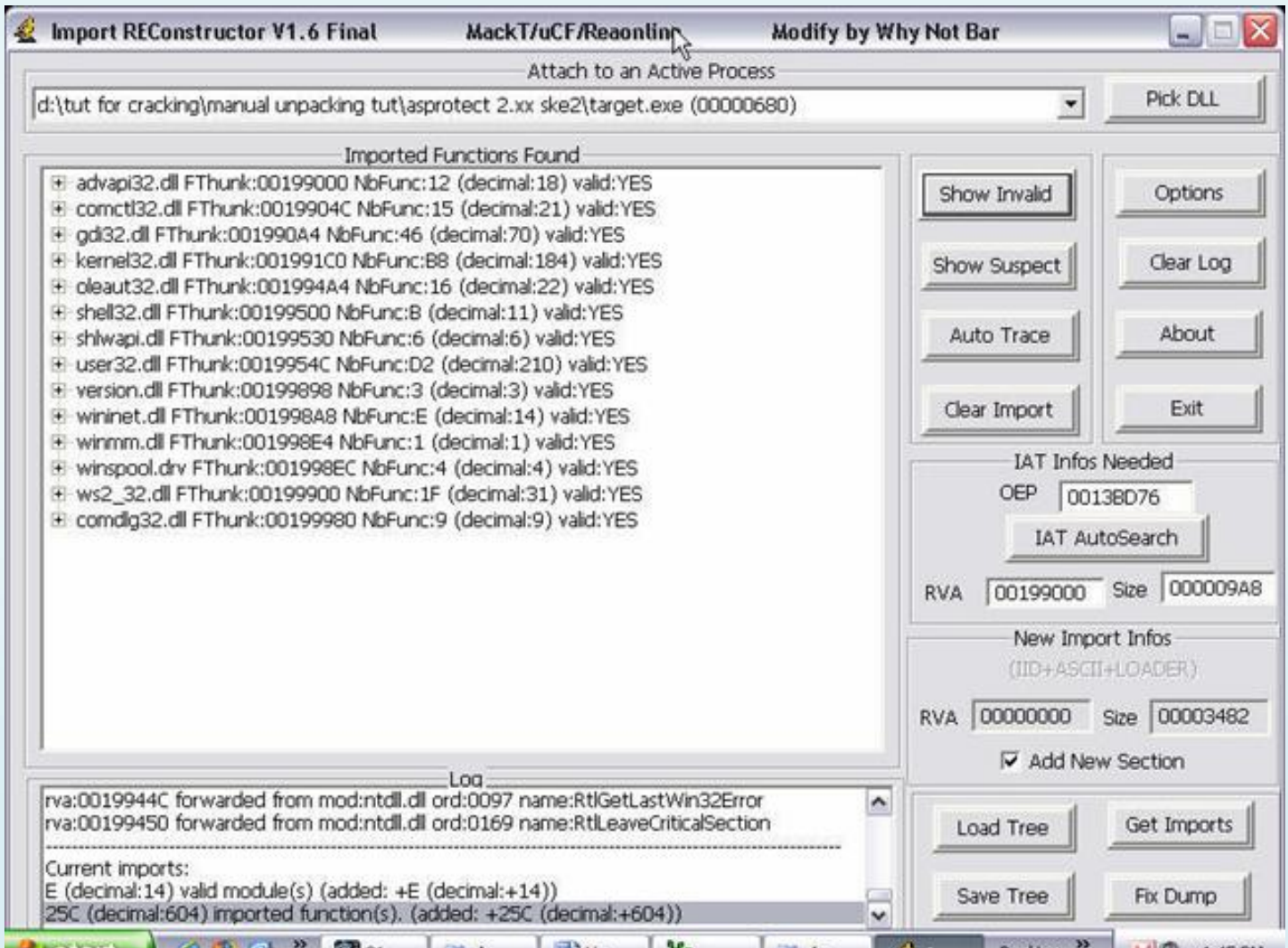
00401020	00401023	00401028	0040102A	0040102C	0040102D	0082C001=1	Address	U	00599000	00599004	00599008	0059900C	00599010	00599014	00599018	0059901C	00599020	00599024	00599028	0059902C	00599030	00599034	00599038	0059903C	00599040	00599044	00599048	0059904C	00599050	00599054	00599058	0059905C	00599060	00599064	00599068	0059906C	00599070	00599074	00599078	0059907C	00599080	00599084	00599088	0059908C	00599090	00599094	00599098	0059909C	005990A0	005990A4	005990A8	005990AC	005990B0	005990B4	005990B8	005990BC	005990C0	005990C4	005990C8	005990CC	005990D0	005990D4	005990D8	005990DC	005990E0	005990E4	005990E8	005990EC	005990F0	005990F4	005990F8	005990FC	00599100	00599104	00599108	0059910C	00599110	00599114	00599118	0059911C	00599120	00599124	00599128	0059912C	00599130	00599134	00599138	0059913C	00599140	00599144	00599148	0059914C	00599150	00599154	00599158	0059915C	00599160	00599164	00599168	0059916C	00599170	00599174	00599178	0059917C	00599180	00599184	00599188	0059918C	00599190	00599194	00599198	0059919C	005991A0	005991A4	005991A8	005991AC	005991B0	005991B4	005991B8	005991BC	005991C0	005991C4	005991C8	005991CC	005991D0	005991D4	005991D8	005991DC	005991E0	005991E4	005991E8	005991EC	005991F0	005991F4	005991F8	005991FC	00599200	00599204	00599208	0059920C	00599210	00599214	00599218	0059921C	00599220	00599224	00599228	0059922C	00599230	00599234	00599238	0059923C	00599240	00599244	00599248	0059924C	00599250	00599254	00599258	0059925C	00599260	00599264	00599268	0059926C	00599270	00599274	00599278	0059927C	00599280	00599284	00599288	0059928C	00599290	00599294	00599298	0059929C	005992A0	005992A4	005992A8	005992AC	005992B0	005992B4	005992B8	005992BC	005992C0	005992C4	005992C8	005992CC	005992D0	005992D4	005992D8	005992DC	005992E0	005992E4	005992E8	005992EC	005992F0	005992F4	005992F8	005992FC	00599300	00599304	00599308	0059930C	00599310	00599314	00599318	0059931C	00599320	00599324	00599328	0059932C	00599330	00599334	00599338	0059933C	00599340	00599344	00599348	0059934C	00599350	00599354	00599358	0059935C	00599360	00599364	00599368	0059936C	00599370	00599374	00599378	0059937C	00599380	00599384	00599388	0059938C	00599390	00599394	00599398	0059939C	005993A0	005993A4	005993A8	005993AC	005993B0	005993B4	005993B8	005993BC	005993C0	005993C4	005993C8	005993CC	005993D0	005993D4	005993D8	005993DC	005993E0	005993E4	005993E8	005993EC	005993F0	005993F4	005993F8	005993FC	00599400	00599404	00599408	0059940C	00599410	00599414	00599418	0059941C	00599420	00599424	00599428	0059942C	00599430	00599434	00599438	0059943C	00599440	00599444	00599448	0059944C	00599450	00599454	00599458	0059945C	00599460	00599464	00599468	0059946C	00599470	00599474	00599478	0059947C	00599480	00599484	00599488	0059948C	00599490	00599494	00599498	0059949C	005994A0	005994A4	005994A8	005994AC	005994B0	005994B4	005994B8	005994BC	005994C0	005994C4	005994C8	005994CC	005994D0	005994D4	005994D8	005994DC	005994E0	005994E4	005994E8	005994EC	005994F0	005994F4	005994F8	005994FC	00599500	00599504	00599508	0059950C	00599510	00599514	00599518	0059951C	00599520	00599524	00599528	0059952C	00599530	00599534	00599538	0059953C	00599540	00599544	00599548	0059954C	00599550	00599554	00599558	0059955C	00599560	00599564	00599568	0059956C	00599570	00599574	00599578	0059957C	00599580	00599584	00599588	0059958C	00599590	00599594	00599598	0059959C	005995A0	005995A4	005995A8	005995AC	005995B0	005995B4	005995B8	005995BC	005995C0	005995C4	005995C8	005995CC	005995D0	005995D4	005995D8	005995DC	005995E0	005995E4	005995E8	005995EC	005995F0	005995F4	005995F8	005995FC	00599600	00599604	00599608	0059960C	00599610	00599614	00599618	0059961C	00599620	00599624	00599628	0059962C	00599630	00599634	00599638	0059963C	00599640	00599644	00599648	0059964C	00599650	00599654	00599658	0059965C	00599660	00599664	00599668	0059966C	00599670	00599674	00599678	0059967C	00599680	00599684	00599688	0059968C	00599690	00599694	00599698	0059969C	005996A0	005996A4	005996A8	005996AC	005996B0	005996B4	005996B8	005996BC	005996C0	005996C4	005996C8	005996CC	005996D0	005996D4	005996D8	005996DC	005996E0	005996E4	005996E8	005996EC	005996F0	005996F4	005996F8	005996FC	00599700	00599704	00599708	0059970C	00599710	00599714	00599718	0059971C	00599720	00599724	00599728	0059972C	00599730	00599734	00599738	0059973C	00599740	00599744	00599748	0059974C	00599750	00599754	00599758	0059975C	00599760	00599764	00599768	0059976C	00599770	00599774	00599778	0059977C	00599780	00599784	00599788	0059978C	00599790	00599794	00599798	0059979C	005997A0	005997A4	005997A8	005997AC	005997B0	005997B4	005997B8	005997BC	005997C0	005997C4	005997C8	005997CC	005997D0	005997D4	005997D8	005997DC	005997E0	005997E4	005997E8	005997EC	005997F0	005997F4	005997F8	005997FC	00599800	00599804	00599808	0059980C	00599810	00599814	00599818	0059981C	00599820	00599824	00599828	0059982C	00599830	00599834	00599838	0059983C	00599840	00599844	00599848	0059984C	00599850	00599854	00599858	0059985C	00599860	00599864	00599868	0059986C	00599870	00599874	00599878	0059987C	00599880	00599884	00599888	0059988C	00599890	00599894	00599898	0059989C	005998A0	005998A4	005998A8	005998AC	005998B0	005998B4	005998B8	005998BC	005998C0	005998C4	005998C8	005998CC	005998D0	005998D4	005998D8	005998DC	005998E0	005998E4	005998E8	005998EC	005998F0	005998F4	005998F8	005998FC	00599900	00599904	00599908	0059990C	00599910	00599914	00599918	0059991C	00599920	00599924	00599928	0059992C	00599930	00599934	00599938	0059993C	00599940	00599944	00599948	0059994C	00599950	00599954	00599958	0059995C	00599960	00599964	00599968	0059996C	00599970	00599974	00599978	0059997C	00599980	00599984	00599988	0059998C	00599990	00599994	00599998	0059999C	00600000	00600004	00600008	0060000C	00600010	00600014	00600018	0060001C	00600020	00600024	00600028	0060002C	00600030	00600034	00600038	0060003C	00600040	00600044	00600048	0060004C	00600050	00600054	00600058	0060005C	00600060	00600064	00600068	0060006C	00600070	00600074	00600078	0060007C	00600080	00600084	00600088	0060008C	00600090	00600094	00600098	0060009C	006000A0	006000A4	006000A8	006000AC	006000B0	006000B4	006000B8	006000BC	006000C0	006000C4	006000C8	006000CC	006000D0	006000D4	006000D8	006000DC	006000E0	006000E4	006000E8	006000EC	006000F0	006000F4	006000F8	006000FC	00600100	00600104	00600108	0060010C	00600110	00600114	00600118	0060011C	00600120	00600124	00600128	0060012C	00600130	00600134	00600138	0060013C	00600140	00600144	00600148	0060014C	00600150	00600154	00600158	0060015C	00600160	00600164	00600168	0060016C	00600170	00600174	00600178	0060017C	00600180	00600184	00600188	0060018C	00600190	00600194	00600198	0060019C	006001A0	006001A4	006001A8	006001AC	006001B0	006001B4	006001B8	006001BC	006001C0	006001C4	006001C8	006001CC	006001D0	006001D4	006001D8	006001DC	006001E0	006001E4	006001E8	006001EC	006001F0	006001F4	006001F8	006001FC	00600200	00600204	00600208	0060020C	00600210	00600214	00600218	0060021C	00600220	00600224	00600228	0060022C	00600230	00600234	00600238	0060023C	00600240	00600244	00600248	0060024C	00600250	00600254	00600258	0060025C	00600260	00600264	00600268	0060026C	00600270	00600274	00600278	0060027C	00600280	00600284	00600288	0060028C	00600290	00600294	00600298	0060029C	006002A0	006002A4	006002A8	006002AC	006002B0	006002B4	006002B8	006002BC	006002C0	006002C4	006002C8	006002CC	006002D0	006002D4	006002D8	006002DC	006002E0	006002E4	006002E8	006002EC	006002F0	006002F4	006002F8	006002FC	00600300	00600304	00600308	0060030C	00600310	00600314	00600318	0060031C	00600320	00600324	00600328	0060032C	00600330	00600334	00600338	0060033C	00600340	00600344	00600348	0060034C	00600350	00600354	00600358	0060035C	00600360	00600364	00600368	0060036C	00600370	00600374	00600378	0060037C	00600380	00600384	00600388	0060038C	00600390	00600394	00600398	0060039C	006003A0	006003A4	006003A8	006003AC	006003B0	006003B4	006003B8	006003BC	006003C0	006003C4	006003C8	006003CC	006003D0	006003D4	006003D8	006003DC	006003E0	006003E4	006003E8	006003EC	006003F0	006003F4	006003F8	006003FC	00600400	00600404	00600408	0060040C	00600410	00600414	00600418	0060041C	00600420	00600424	00600428	0060042C	00600430	00600434	00600438	0060043C	00600440	00600444	00600448	0060044C	00600450	00600454	00600458	0060045C	00600460	00600464	00600468	0060046C	00600470	00600474	00600478	0060047C	00600480	00600484	00600488	0060048C	00600490	00600494	00600498	0060049C	006004A0	006004A4	006004A8	006004AC	006004B0	006004B4	006004B8	006004BC	006004C0	006004C4	006004C8	006004CC	006004D0	006004D4	006004D8	006004DC	006004E0	006004E4	006004E8	006004EC	006004F0	006004F4	006004F8	006004FC	00600500	00600504	00600508	0060050C	00600510	00600514	00600518	0060051C	00600520	00600524	00600528	0060052C	00600530	00600534	0060053
----------	----------	----------	----------	----------	----------	------------	---------	---	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	---------

_Nhan es Y



OK _Nhan is finished **Fix IAT** and the **script** by **OEP**. In addition, this script also **Tay** from the dump file with the name "**un_Target.exe**" This takes out the **LordPE** used to dump

_ Open **ImportREC** up. Select **List** in **Process target.exe**. **OEP** = Enter **0053BD76 - 00,400,000 (Imagebase) = 0013BD76, IAT AutoSearch** Click -> **Get Imports** -> **Show Invalid**





_ Hú complete I have any function *Invalid* ... OK, Click **Fix Dumped un_Target.exe** File and select Run Test
(Note: do not close the window OllyDBG again because we still rely on it to Fix Code of steps. I call a temporary window is OllyDBG 1)



_ N han **C** lick h e r e



_ We easily found the address **02310000** is a memory address areas outside of Target and **SectionTable** tasks we

need a Fix for reasonable. Load "**un_Target.exe**" to **OllyDBG 2**

0053BD76	6A 60	PUSH 60	<===OEP
0053BD78	68 58835A00	PUSH un_Targe.005A8358	
0053BD7D	E8 96390000	CALL un_Targe.0053F718	
0053BD82	BF 94000000	MOV EDI, 94	
0053BD87	8BC7	MOV EAX, EDI	
0053BD89	E8 42360000	CALL un_Targe.0053F3D0	
0053BD8E	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
0053BD91	8BF4	MOV ESI, ESP	
0053BD93	893E	MOV DWORD PTR DS:[ESI], EDI	
0053BD95	56	PUSH ESI	
0053BD96	FF15 DCA28F00	CALL NEAR DWORD PTR DS:[<&kernel32.GetUkernel32.GetU	
0053BD9C	8B4E 10	MOV ECX, DWORD PTR DS:[ESI+10]	
0053BD9F	890D 045C6000	MOV DWORD PTR DS:[605C04], ECX	
0053BDA5	8B46 04	MOV EAX, DWORD PTR DS:[ESI+4]	
0053BDA8	A3 105C6000	MOV DWORD PTR DS:[605C10], EAX	

_ N han of the C + F, G o J 02310000 MP and a far

0053F260	83EA 01	SUB EDX, 1	
0053F263	75 F6	JNZ SHORT un_Targe.0053F25B	
0053F265	8B4424 08	MOV EAX, DWORD PTR SS:[ESP+8]	
0053F269	5F	POP EDI	
0053F26A	C3	RETN	
0053F26B	8B4424 04	MOV EAX, DWORD PTR SS:[ESP+4]	
0053F26F	C3	RETN	
0053F270	E9 8B0DD001	JMP 02310000	
0053F275	92	XCHG EAX, EDX	
0053F276	77 29	JA SHORT un_Targe.0053F2A1	
0053F278	1F	POP DS	
0053F279	72 72	JB SHORT un_Targe.0053F2ED	
0053F27B	6232	BOUND ESI, QWORD PTR DS:[EDX]	
0053F27D	76 4F	JBE SHORT un_Targe.0053F2CE	
0053F27F	55	PUSH EBP	
0053F280	BF 96E5005D	MOV EDI, 5D00E596	
0053F285	50	PUSH EAX	
0053F286	58	POP EAX	
0053F287	5D	POP EBP	

_ Press **Ctrl + L** also considered **JMP** order any more than the same teachings and in this case do? Only 1 Jump single command, appears to work quietly here. Back to the **OllyDBG 1**, press **Ctrl + G**, type **02310000**, and we come

02310000	68 FFFFFFFF	PUSH -1	
02310005	50	PUSH EAX	
02310006	3E:EB 02	JMP SHORT 0231000B	Superfluous prefix
02310009	CD 20	INT 20	
0231000B	33C0	XOR EAX, EAX	
0231000D	64:8B00	MOV EAX, DWORD PTR FS:[EAX]	
02310010	50	PUSH EAX	
02310011	13C1	ADC EAX, ECX	
02310013	EB 01	JMP SHORT 02310016	
02310015	9A B84EE445 00	CALL FAR 0300:45E44EB8	
0231001C	44	INC ESP	
0231001D	24 18	AND AL, 18	
0231001F	F3:	PREFIX REP:	Superfluous prefix
02310020	EB 02	JMP SHORT 02310024	
02310022	CD 20	INT 20	
02310024	8D4424 0C	LEA EAX, DWORD PTR SS:[ESP+C]	
02310028	EB 01	JMP SHORT 0231002B	
0231002A	F3:	PREFIX REP:	Superfluous prefix
0231002B	FF30	PUSH DWORD PTR DS:[EAX]	
0231002D	F2:	PREFIX REPNE:	Superfluous prefix
0231002E	EB 01	JMP SHORT 02310031	
02310030	F0:83E8 33	LOCK SUB EAX, 33	LOCK prefix is not
02310034	B8 1EBD4500	MOV EAX, 45BD1E	
02310039	58	POP EAX	

_ C h u n g a t c h has 2ca F XC than the case of this

• **Method 1:** In the window **OllyDBG 1** copy from the end **0231000 RETN** code. Through the window **OllyDBG 2** find 1 area code blank paste code to just copy it, then the command Fix Jump jump to the new address for the right is OK

_ Copy code from **0,231,000 -> RETN**

0231002A	F3:	PREFIX REP:	Superfluous prefix
0231002B	FF30	PUSH DWORD PTR DS:[EAX]	
0231002D	F2:	PREFIX	Superfluous prefix
0231002E	EB 01	JMP SH	
02310030	F0:83E8 33	LOCK SU	
02310034	B8 1EBD4500	MOV EAX	
02310039	58	POP EAX	
0231003A	64:8925 000000	MOV DWO	
02310041	55	PUSH EB	
02310042	8F4424 0C	POP DWO	
02310046	8D6C35 DD	LEA EBP	
0231004A	8D6C24 0C	LEA EBP	
0231004E	50	PUSH EA	
0231004F	C3	RETN	
02310050	0000	ADD BYT	
02310052	0000	ADD BYT	
02310054	0000	ADD BYT	
02310056	0000	ADD BYT	
02310058	0000	ADD BYT	
0231005A	0000	ADD BYT	
0231005C	0000	ADD BYT	
0231005E	0000	ADD BYT	

_ Through the window **OllyDBG 2** find 1 area code and here they found **006C9000**

006C9000	0000	ADD BYTE PTR DS:[EAX], AL	
006C9002	0000	ADD BYTE PTR DS:[EAX], AL	
006C9004	0000	ADD BYTE PTR DS:[EAX], AL	
006C9006	0000	ADD BYTE PTR DS:[EAX], AL	
006C9008	0000	ADD BYTE PTR DS:[EAX], AL	
006C900A	0000	ADD BYTE PTR DS:[EAX], AL	
006C900C	0000	ADD BYTE PTR DS:[EAX], AL	
006C900E	0000	ADD BYTE PTR DS:[EAX], AL	
006C9010	0000	ADD BYTE PTR DS:[EAX], AL	
006C9012	0000	ADD BYTE PTR DS:[EAX], AL	
006C9014	0000	ADD BYTE PTR DS:[EAX], AL	
006C9016	0000	ADD BYTE PTR DS:[EAX], AL	

_ Danvao paragraph c o d e C o p just italy B E N OllyDBG1

006C9013	EB 01	JMP SHORT un_Targe.006C9016	
006C9015	9A B84EE445 00	CALL FAR 0300:45E44EB8	
006C901C	44	INC ESP	
006C901D	24 18	AND AL, 18	
006C901F	F3:	PREFIX REP:	Superfluous pr
006C9020	EB 02	JMP SHORT un_Targe.006C9024	
006C9022	CD 20	INT 20	
006C9024	8D4424 0C	LEA EAX, DWORD PTR SS:[ESP+C]	
006C9028	EB 01	JMP SHORT un_Targe.006C902B	
006C902A	F3:	PREFIX REP:	Superfluous pr
006C902B	FF30	PUSH DWORD PTR DS:[EAX]	
006C902D	F2:	PREFIX REPNE:	Superfluous pr
006C902E	EB 01	JMP SHORT un_Targe.006C9031	
006C9030	F0:83E8 33	LOCK SUB EAX, 33	LOCK prefix is
006C9034	B8 1EBD4500	MOV EAX, un_Targe.0045BD1E	
006C9039	58	POP EAX	
006C903A	64:8925 000000	MOV DWORD PTR FS:[0], ESP	
006C9041	55	PUSH EBP	
006C9042	8F4424 0C	POP DWORD PTR SS:[ESP+C]	
006C9046	8D6C35 DD	LEA EBP, DWORD PTR SS:[EBP+ESI-23]	
006C904A	8D6C24 0C	LEA EBP, DWORD PTR SS:[ESP+C]	
006C904E	50	PUSH EAX	
006C904F	C3	RETN	
006C9050	0000	ADD BYTE PTR DS:[EAX], AL	
006C9052	0000	ADD BYTE PTR DS:[EAX], AL	

_ Save and 0053F270 to Fix the Jump command to correct

0053F263	75 F0	JNZ SHORT un_Targe.0053F25B
0053F265	8B4424 08	MOV EAX, DWORD PTR SS:[ESP+8]
0053F269	5F	POP EDI
0053F26A	C3	RETN
0053F26B	8B4424 04	MOV EAX, DWORD PTR SS:[ESP+4]
0053F26F	C3	RETN
0053F270	- E9 8B9D1800	JMP un_Targe.006C9000
0053F275	92	XCHG EAX, EDX
0053F276	✓ 77 29	JA SHORT un_Targe.0053F2A1
0053F278	1F	POP DS
0053F279	✓ 72 72	JB SHORT un_Targe.0053F2ED
0053F27B	6232	BOUND ESI, QWORD PTR DS:[EDX]
0053F27D	✓ 76 4F	JBE SHORT un_Targe.0053F2CE
0053F27F	55	PUSH EBP
0053F280	BF 96E5005D	MOV EDI, 5D00E596
0053F285	50	PUSH EAX
0053F286	58	POP EAX
0053F287	2B60 5E	SUB ESP, DWORD PTR DS:[EAX+5E]
0053F28A	53	PUSH EBX

_ **Save again** ... Run ... khua test considered khua hen run gòi

- **Method 2:** This may seem more professional. You notice in the code from **02310000** -> **RETN** many spam code with 1 hairbreadth of **ASM**, you can MOD the code for other neat and patch directly in the order that **Jump 02310000** i need it to 1 new area code on nhucach

_Trong Window **OllyDBG 2**, press **Ctrl + A** in **0053F270** and we see

0053F26F	L. C3	RETN	
0053F270	\$- E9 8B0DD001	JMP 02310000	
0053F275	92	DB 92	
0053F276	77	DB 77	CHAR 'w'
0053F277	29	DB 29	CHAR ')'
0053F278	1F	DB 1F	
0053F279	72	DB 72	CHAR 'r'
0053F27A	72	DB 72	CHAR 'r'
0053F27B	62	DB 62	CHAR 'b'
0053F27C	32	DB 32	CHAR '2'
0053F27D	76	DB 76	CHAR 'v'
0053F27E	4F	DB 4F	CHAR 'O'
0053F27F	55	DB 55	CHAR 'U'
0053F280	BF	DB BF	
0053F281	96	DB 96	
0053F282	E5	DB E5	
0053F283	00	DB 00	
0053F284	5D	DB 5D	CHAR 'J'
0053F285	50	DB 50	CHAR 'P'
0053F286	58	DB 58	CHAR 'X'
0053F287	2B	DB 2B	CHAR '+'
0053F288	60	DB 60	CHAR ``
0053F289	5E	DB 5E	CHAR '^'
0053F28A	53	DB 53	CHAR 'S'
0053F28B	38	DB 38	CHAR '8'
0053F28C	3D	DB 3D	CHAR '='
0053F28D	7F	DB 7F	
0053F28E	AA	DB AA	
0053F28F	\$ 55	PUSH EBP	
0053F290	8BEC	MOV EBP, ESP	

_Bay Hours of your MOD to the code from 0,231,000 -> RETN to insert the code from 0053F270 -> 0053F28E on but just to ensure the right structure

CodeGoCCOODDI

02310000	68 FFFFFFFF	PUSH -1
02310005	50	PUSH EAX
02310006	3E:EB 02	JMP SHORT 0231000B
02310009	CD 20	INT 20
0231000B	33C0	XOR EAX, EAX
0231000D	64:8B00	MOV EAX, DWORD PTR FS:[EAX]
02310010	50	PUSH EAX
02310011	13C1	ADC EAX, ECX
02310013	EB 01	JMP SHORT 02310016
02310015	9A B84EE445 00	CALL FAR 0300:45E44EB8
0231001C	44	INC ESP
0231001D	24 18	AND AL, 18
0231001F	F3:	PREFIX REP:
02310020	EB 02	JMP SHORT 02310024
02310022	CD 20	INT 20
02310024	8D4424 0C	LEA EAX, DWORD PTR SS:[ESP+C]
02310028	EB 01	JMP SHORT 0231002B
0231002A	F3:	PREFIX REP:
0231002B	FF30	PUSH DWORD PTR DS:[EAX]
0231002D	F2:	PREFIX REPNE:
0231002E	EB 01	JMP SHORT 02310031
02310030	F0:83E8 33	LOCK SUB EAX, 33
02310034	B8 1EBD4500	MOV EAX, 45BD1E
02310039	58	POP EAX
0231003A	64:8925 000000	MOV DWORD PTR FS:[0], ESP
02310041	55	PUSH EBP
02310042	8F4424 0C	POP DWORD PTR SS:[ESP+C]
02310046	8D6C35 0D	LEA EBP, DWORD PTR SS:[EBP+ESI-23]
0231004A	8D6C24 0C	LEA EBP, DWORD PTR SS:[ESP+C]

0053F26B	> 8B4424 04	MOV EAX, DWORD PTR SS:[ESP+4]
0053F26F	L. C3	RETN
0053F270	6A FF	PUSH -1
0053F272	50	PUSH EAX
0053F273	64:A1 000000	MOV EAX, DWORD PTR FS:[0]
0053F279	50	PUSH EAX
0053F27A	8B4424 0C	MOV EAX, DWORD PTR SS:[ESP+C]
0053F27E	64:8925 0000	MOV DWORD PTR FS:[0], ESP
0053F285	896C24 0C	MOV DWORD PTR SS:[ESP+C], EBP
0053F289	8D6C24 0C	LEA EBP, DWORD PTR SS:[ESP+C]
0053F28D	50	PUSH EAX
0053F28E	C3	RETN
0053F28F	\$ 55	PUSH EBP
0053F290	. 8BEC	MOV EBP, ESP
0053F292	. 83EC 20	SUB ESP, 20
0053F295	. 8B45 08	MOV EAX, [ARG.1]
0053F298	. 56	PUSH ESI
0053F299	. 57	PUSH EDI
0053F29A	. 6A 08	PUSH 8
0053F29C	. 59	POP ECX
0053F29D	. BE E4845A00	MOV ESI, un_Targe.005A84E4
0053F2A2	. 8D7D E0	LEA EDI, [LOCAL.8]
0053F2A5	. F3:A5	REP MOVS DWORD PTR ES:[EDI], EAX
0053F2A7	. 8945 F8	MOV [LOCAL.2], EAX
0053F2AA	. 8B45 0C	MOV EAX, [ARG.2]
0053F2AD	. 8945 FC	MOV [LOCAL.1], EAX
0053F2B0	. 8D45 F4	LEA EAX, [LOCAL.3]
0053F2B3	. 5A	PUSH EAX

02310042	8F4424 0C	POP DWORD PTR SS:[ESP+0]	0053F2A0	. 8945 FC	MOV [LOCAL.1], EAX
02310046	8D6C35 DD	LEA EBP, DWORD PTR SS:[EBP+ESI-23]	0053F2B0	. 8D45 F4	LEA EAX, [LOCAL.3]
0231004A	8D6C24 0C	LEA EBP, DWORD PTR SS:[ESP+C]	0053F2B3	. 50	PUSH EAX
0231004E	50	PUSH EAX	0053F2B4	. FF75 F0	PUSH [LOCAL.4]
0231004F	C3	RETN	0053F2B7	. FF75 F4	PUSH [LOCAL.7]

_save Run and try to run nhuNgua considered **unpack Done!**

_ Òn D G C x FF that will reduce a c t i o n admits

CFF Explorer V - by Ntoskrnl - [UN_TAR~3.EXE]

File Settings ?

File: UN_TAR~3.EXE

Dos Header

Nt Headers

File Header

Optional Header

Data Directories [x]

Section Headers [x]

Import Directory

Resource Directory

Relocation Directory

Hex Editor

Address Converter

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Re
000002B0	000002B8	000002BC	000002C0	000002C4	000002C8	000002CC	0C
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	W
	00198000	00001000	00198000	00001000	00000000	00000000	00
	00061000	00199000	00061000	00199000	00000000	00000000	00
	0011D000	001FA000	0011D000	001FA000	00000000	00000000	00
.rsrr	000E8000	00317000	000E8000	00317000	00000000	00000000	00
	0002D000	003FF000	0002D000	003FF000	00000000	00000000	00
.3800hk	000CD000	0042C000	000CD000	0042C000	00000000	00000000	00
.idata	00001000	004F9000	00001000	004F9000	00000000	00000000	00
.mackit	00004000	004FA000	00004000	004FA000	00000000	00000000	00

Save _ again

G r l Ee TsF italy Ou the Co mpu t e r A _ of e l, e mbi Z o, M A B oo nb italy, H o acnh, Nina B e, e ki nman o w

ar, Z o i D e ux, M e r c, l o e ight to nix, T r o b icky italy, Takad a iamidi ot, of the e n t e n handi ... and italy o u!

The N h a n a g, Day 1 0 be a n g 1 1 20 0 6

WhotitalyNBar

I. Introduction:

*Armadillo ASProtect and certainly do not need to introduce sure you also know it is what the public and it causes difficulties for our nhuthe when the meat of Soft that the presence of it. Armadillo **Hacnho** he was exposed to the bone with a series of impressive tut unpack Armadillo make "**Silicon Realms Toolworks**" Update regularly to ... hehe ... but on the ASProtect now also about to go to the dust that 4Rum large **Cracking Unpacking** and the world launch of a series of tut, Tool and Scripts to guide and 2.xx ASProtect unpack quickly. orientate the movement and help to many new people join REA can use the script to unpack Soft They should write this **TUT** for reference.*

II. Tools:

- Tool v AC A P l c u d g in need úng:
 - *O ll italy D G B 1. 1 0*
 - *UM D Oll italy ppl u in g*
 - *B O D i g Scr 1 pt. 4 7*
 - *I mport R E C 1. 6*
 - *P I D E 0. 9 4*
- Sc r i pt:
 - *"As the rot e ct 2. XXSKE I A T F i x e R V 1st 0 2" by X V ol.*
 - *"As the rot e ct 2nd x xSKE O P e d e f r in" B VolX italy*
- Target: In this TUT get 2 children 2 to various forms you can easily grasp.
 - 1st *h e n a n c e M o v e i 2. 1*
h xx p: // mov a v i. c om / enh a n c e mov e i /
 - 2. *4 usics M M P 3 B it a t e r C h a n r G E 1st 5*
h xx p: // w _w w. _usics_m_4. _co_m/_mp_3 - MP_3 - _children_and_very_e_e_r._h_tm

Please say any more Bro want MUP ASProtect 2.xx tut should read "**UN P A C K I N G A S P R E C T O T V 2. 1 S K W I T H E A D and N C E D I M P T O R P O R T E C T I O N**" by **M a D _ A n M R C H 3 of 3 L s [R A T E A m]**.

III. Unpacking:

UT T # 1: **U n p a c of the ce M a n o v e i 2. 1**

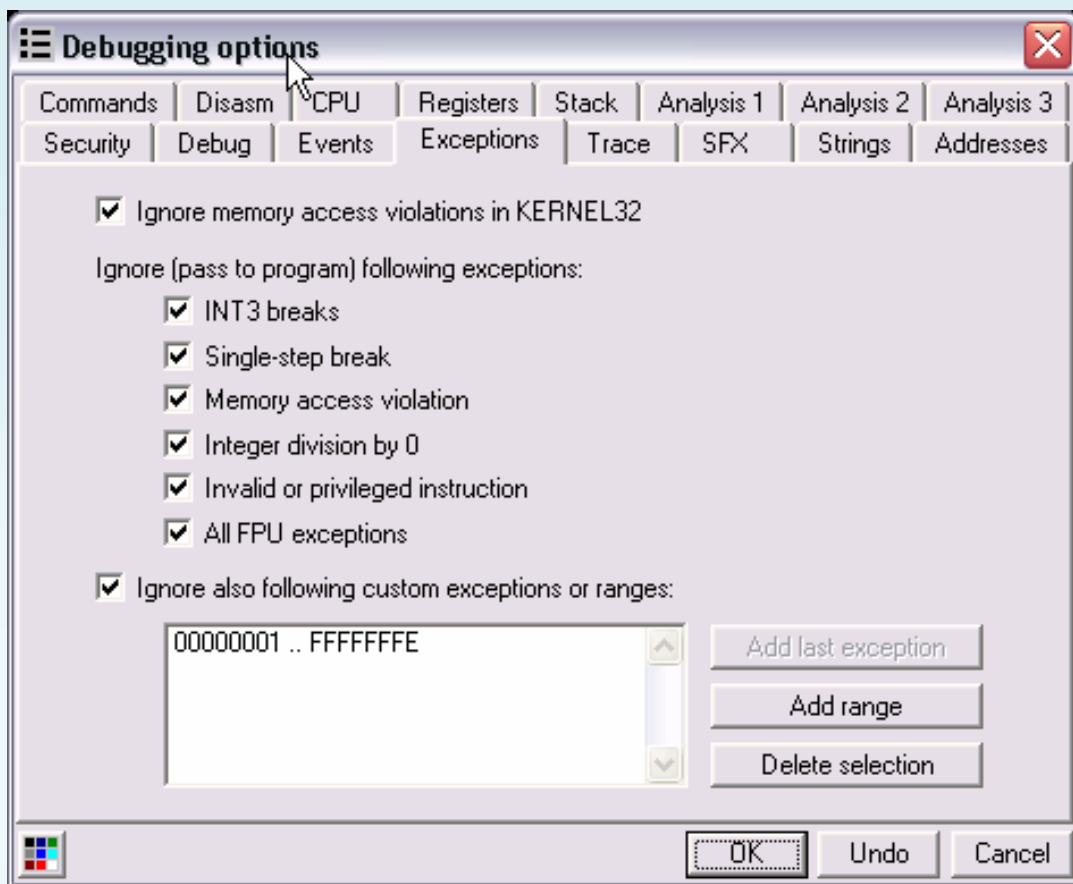
_First they treat statistics **EnhanceMovie 2.1** A operation is familiar to Detect **PEiD** considered it or pack ko?



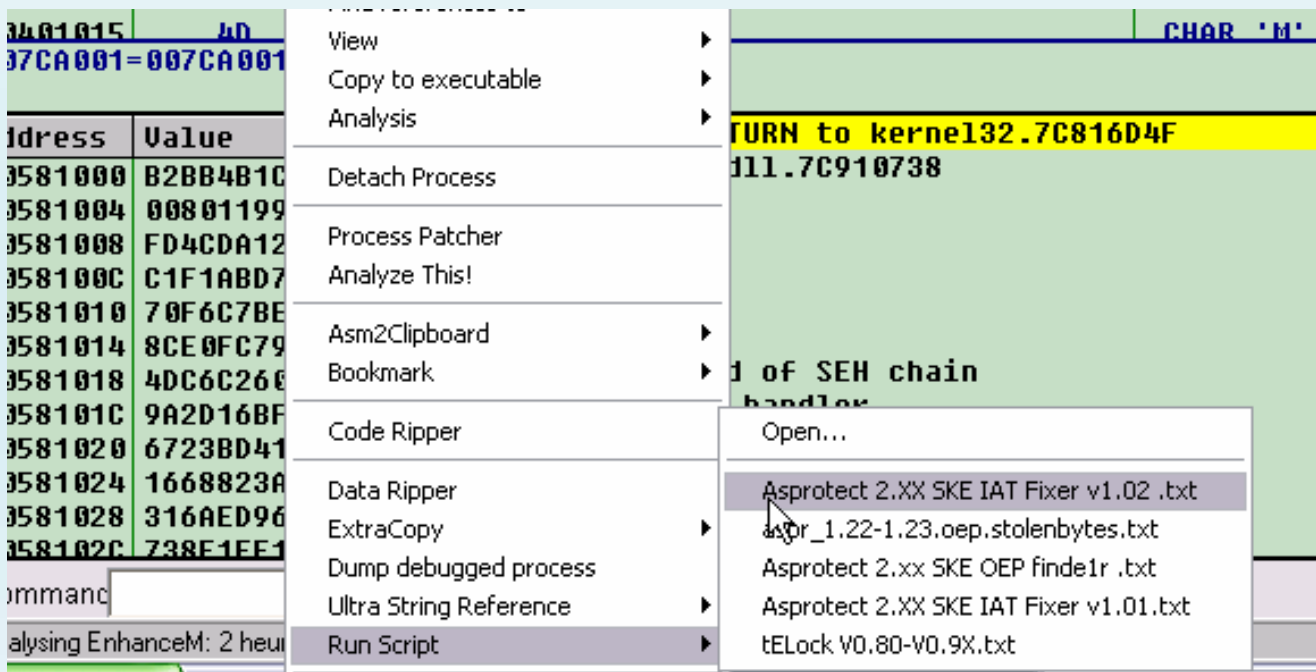
Hic hic _see "ASProtect 2.1x SKE -> Alexey Solodovnikov" quai filed. breathe a very deep and Load target to **OllyDBG** and we stop here:

Paused				L E M T W H C P K B R ... S			
Address	Hex	dump	Disassembly		Comment		
00401000	\$	68 01A07C00	PUSH 7CA001				
00401005	.	E8 01000000	CALL 0040100B		0040100B		
0040100A	.	C3	RET				
0040100B	\$	C3	RET				
0040100C		DD	DB DD				
0040100D		CD	DB CD				
0040100E		2D	DB 2D		CHAR '-'		
0040100F		47	DB 47		CHAR 'G'		
00401010		9F	DB 9F				
00401011		E6	DB E6				

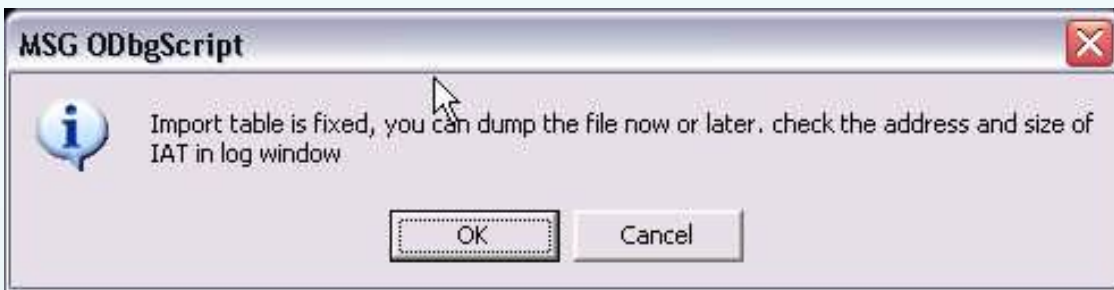
_Han **N T A L O V** + achon as a u s:



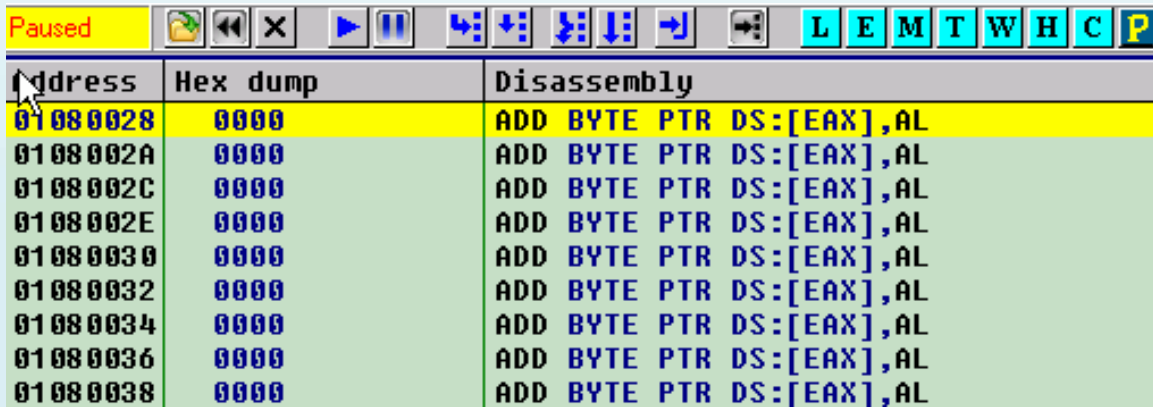
_ Now we run a script "*Asprotect 2.XX SKE IAT Fixer v1.02*" to automatically Hook APIs that have marked ASPR



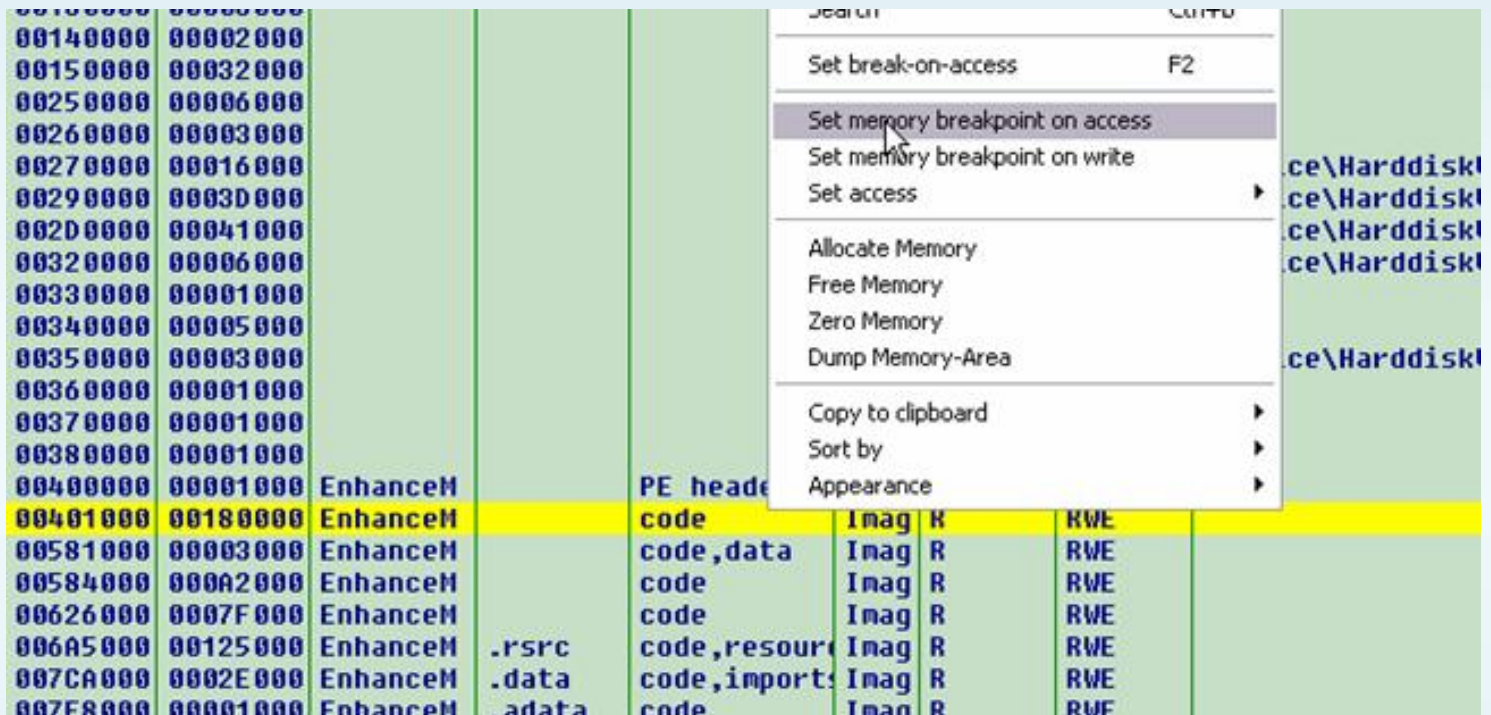
_ Haha, Script running very well and the following notice:



_ In lly we stop here:



_Nhien Of us now is to find the **OEP** to dump files. Press **Alt + M** and 1 *Memory Breakpoint* Set on the *Access Code Section*



_N han **F9** to our **O P E**:

Address	Hex dump	Disassembly	Comment
0054DE28	59	POP ECX	
0054DE29	C3	RET	
0054DE2A	6A 60	PUSH 60	<==OEP
0054DE2C	68 50196100	PUSH 611950	
0054DE31	E8 CE2A0000	CALL 00550904	00550904
0054DE36	BF 94000000	MOV EDI,94	
0054DE3B	8BC7	MOV EAX,EDI	
0054DE3D	E8 0EE7FFFF	CALL 0054C550	0054C550
0054DE42	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0054DE45	8BF4	MOV ESI,ESP	
0054DE47	893E	MOV DWORD PTR DS:[ESI],EDI	
0054DE49	56	PUSH ESI	
0054DE4A	FF15 40425800	CALL NEAR DWORD PTR DS:[584240]	kernel32.GetVersionExA
0054DE50	8B4F 10	MOV ECX,DWORD PTR DS:[ESI+10]	

_ And what further delay "dump" thui, where they used to dump **OllyDUMP plugin**, remember to choose the same as the image below

OllyDump - EnhanceMovie.exe

Start Address: 400000 Size: 3F9000 **Dump**

Entry Point: 1000 -> Modify: 14DE2A Get EIP as OEP **Cancel**

Base of Code: 1000 Base of Data: 184000

☒ Fix Raw Size & Offset of Dump Image

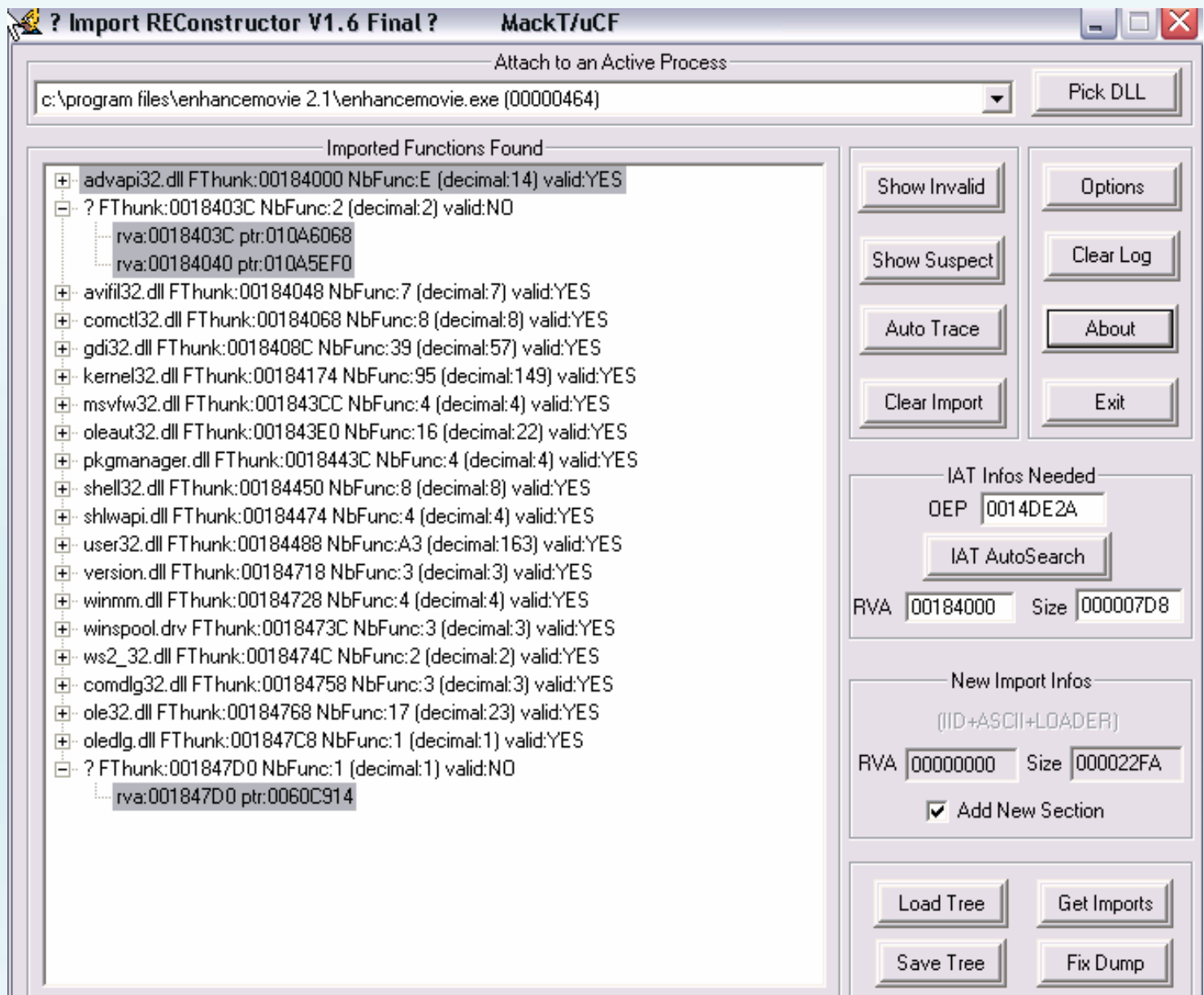
Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
	00180000	00001000	00180000	00001000	E0000040
	00003000	00181000	00003000	00181000	E0000040
	000A2000	00184000	000A2000	00184000	E0000040
	0007F000	00226000	0007F000	00226000	E0000040
.rsrc	00125000	002A5000	00125000	002A5000	E0000040
.data	0002E000	003CA000	0002E000	003CA000	E0000040
.adata	00001000	003F8000	00001000	003F8000	E0000040

☐ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

_ Open **ImportREC** and fill the **OEP**, **Get Imports** -> **Show Valid**



_ According to the experience they have been the Soft Code with *Microsoft Visual C++* and the pack with *ASProtect 2.xx SKE* when Run Scripts to Fix IAT, the lack *FindResourceA* 2 function and *GetProcAddress*. I do not know exactly but i Soft most of the meat he has over this phenomenon. (The Uncle MUP the situation has chan italy so they try gòi)

_ So we need to change the nhusau (Double Click the need to change)

Thun k F: 00184 0 3 C N bFu the c: 00000 0 0 2

0	0018403C	?	0000	010A6068
0	00184040	?	0000	010EA5F0

See the

Thun k F: 00184 0 3 C N bFu the c: 00000 0 0 2
1 0018403 C k e l e RN 3 2. D ll 00 E0 R F i e nd so u rc e A
1 0018404 0 k e l e RN 32. D ll 019 8 Ge t Ad P ro c d e r ss

Select **Show Valid-> Cut ThunK (s)**, click to select File **Fix dump** dump

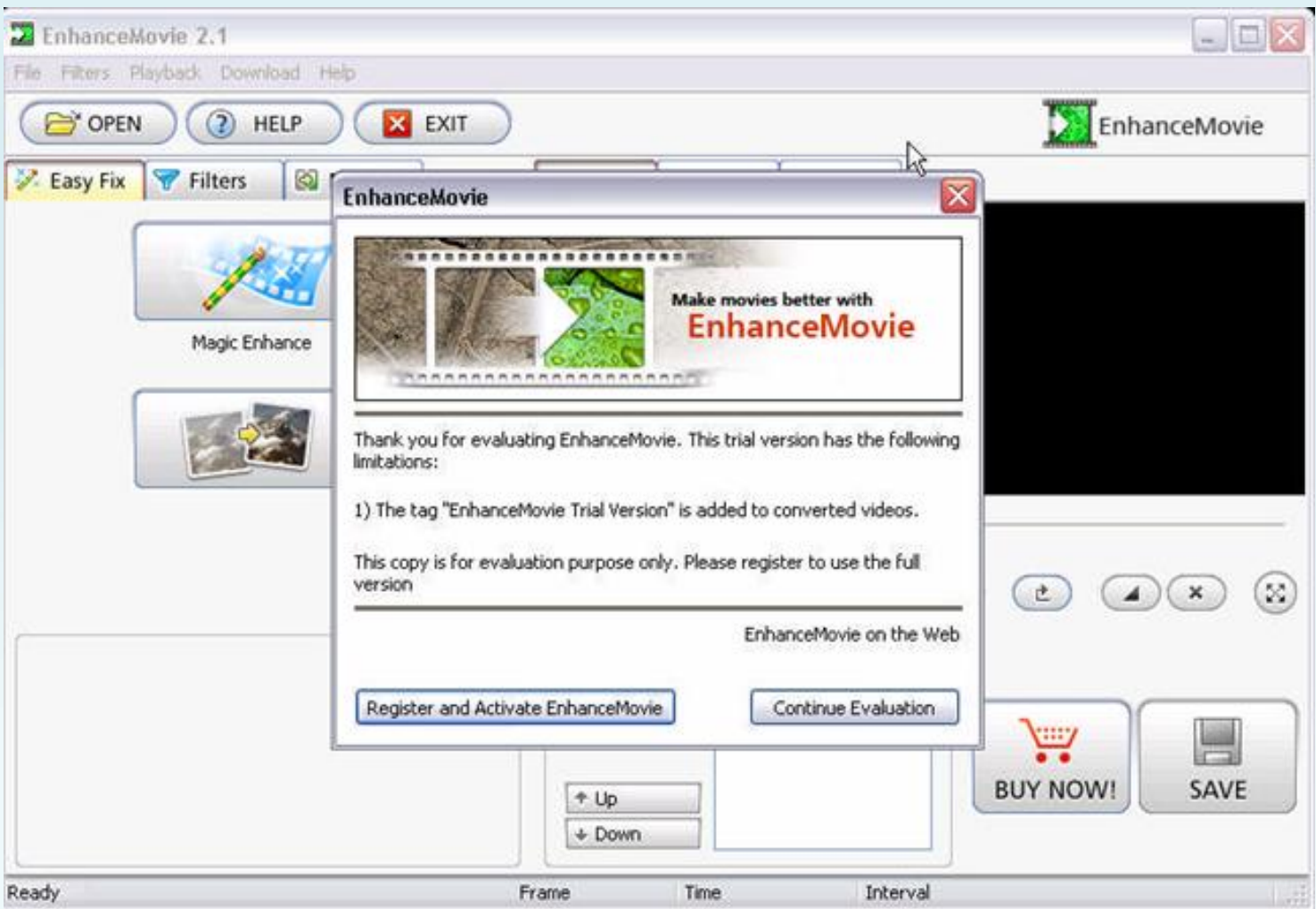
Imported Functions Found

```

+ advapi32.dll FT hunk:00184000 NbFunc:E (decimal:14) valid:YES
- kernel32.dll FT hunk:0018403C NbFunc:2 (decimal:2) valid:YES
  rva:0018403C mod:kernel32.dll ord:00E0 name:FindResourceA
  rva:00184040 mod:kernel32.dll ord:0198 name:GetProcAddress
+ avifil32.dll FT hunk:00184048 NbFunc:7 (decimal:7) valid:YES
+ comctl32.dll FT hunk:00184068 NbFunc:8 (decimal:8) valid:YES
+ gdi32.dll FT hunk:0018408C NbFunc:39 (decimal:57) valid:YES
+ kernel32.dll FT hunk:00184174 NbFunc:95 (decimal:149) valid:YES
+ msvfw32.dll FT hunk:001843CC NbFunc:4 (decimal:4) valid:YES
+ oleaut32.dll FT hunk:001843E0 NbFunc:16 (decimal:22) valid:YES
+ pkgmanager.dll FT hunk:0018443C NbFunc:4 (decimal:4) valid:YES
+ shell32.dll FT hunk:00184450 NbFunc:8 (decimal:8) valid:YES
+ shlwapi.dll FT hunk:00184474 NbFunc:4 (decimal:4) valid:YES
+ user32.dll FT hunk:00184488 NbFunc:A3 (decimal:163) valid:YES
+ version.dll FT hunk:00184718 NbFunc:3 (decimal:3) valid:YES
+ winmm.dll FT hunk:00184728 NbFunc:4 (decimal:4) valid:YES
+ winspool.drv FT hunk:0018473C NbFunc:3 (decimal:3) valid:YES
+ ws2_32.dll FT hunk:0018474C NbFunc:2 (decimal:2) valid:YES
+ comdlg32.dll FT hunk:00184758 NbFunc:3 (decimal:3) valid:YES
+ ole32.dll FT hunk:00184768 NbFunc:17 (decimal:23) valid:YES
+ oledlg.dll FT hunk:001847C8 NbFunc:1 (decimal:1) valid:YES

```

Test File Dumped.exe, hahaha run lickerish



_D Ung **P I D E** q u é F i l t D e e d p u m _ . E x e



_ **Unpacked successful** Hopefully you can to the soft meat left on the form Homepage protect identical.

UT T # 2: U n p a c k 4 m i c s U.S. M P 3 B i t a t e C h a n g e r 1.5

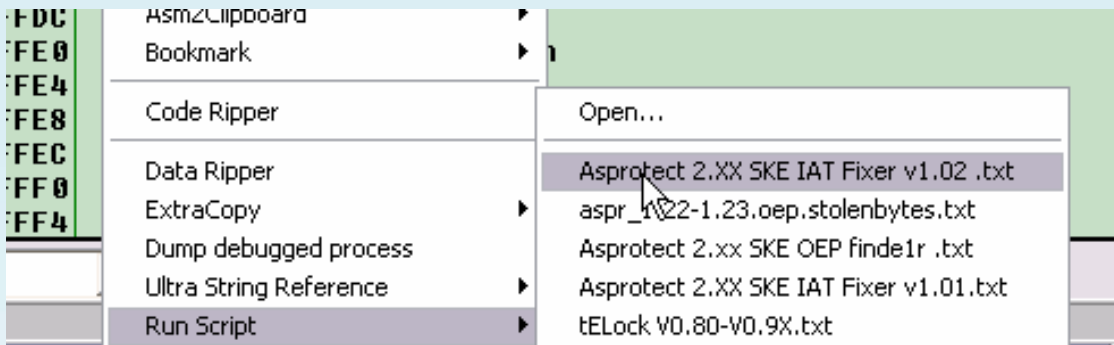
_ More of **us em 4 M i c s M P 3 B e l g i u m t r a t e C h a n g e r 1.5.**



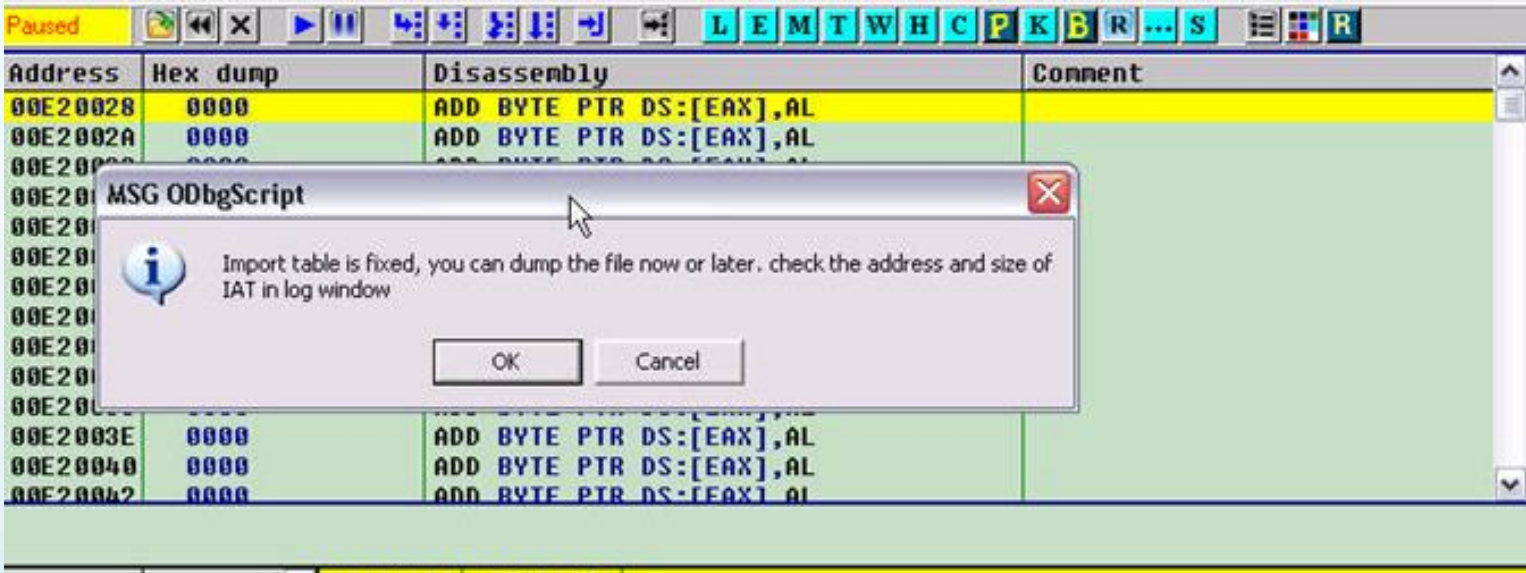
Lo_a d T a r g e TV O I I A B D G italy and stop here:

Address	Hex dump	Disassembly	Comment
00401000	\$ 68 01307A00	PUSH 7A3001	
00401005	. E8 01000000	CALL 0040100B	0040100B
0040100A	. C3	RET	
0040100B	. C3	RET	
0040100C	. 0B8E 1C3257CF	OR ECX,DWORD PTR DS:[ESI+CF57321C]	
00401012	> 36:54	PUSH ESP	
00401014	? CA 0971	RETF 7109	
00401017	. C8 258BDB	ENTER 8B25,0DB	
0040101B	? A0 3B8FDEA8	MOV AL,BYTE PTR DS:[A8DE8F3B]	
00401020	. 691E 50DC4077	IMUL EBX,DWORD PTR DS:[ESI],7740DC50	rpModule
00401026	? EA C917FAEF 8	JMP FAR F28A:EFFA17C9	

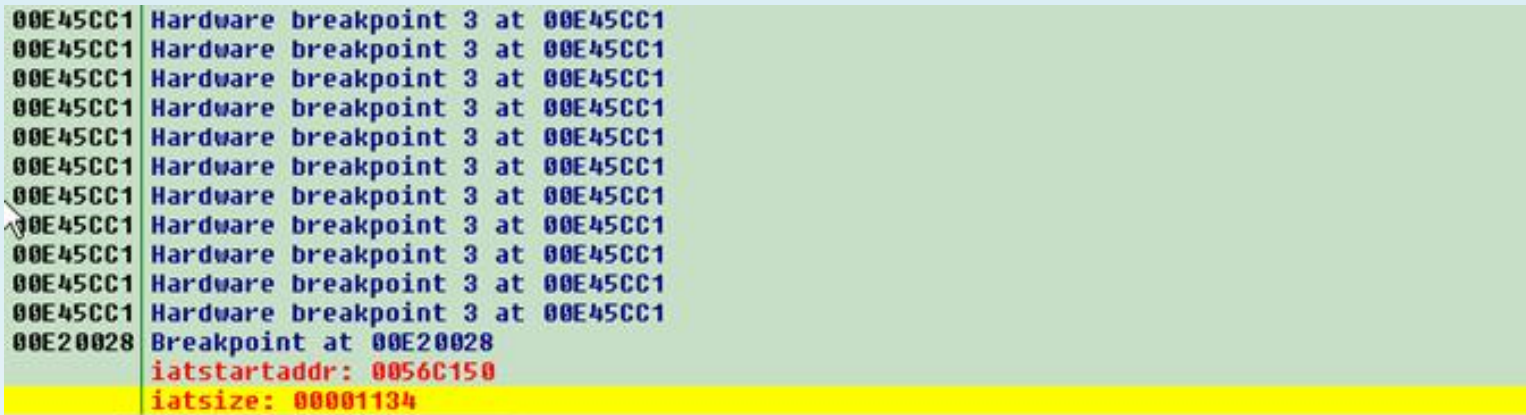
_ Just as the nrun Sc ê r r i p t "A s e t o p r c t 2nd XX KE A S T F i x e RV 1st 0 2"



_ Script run and finished the NAG report easy trade



_ Han N T A L + L and C using a number of G L O



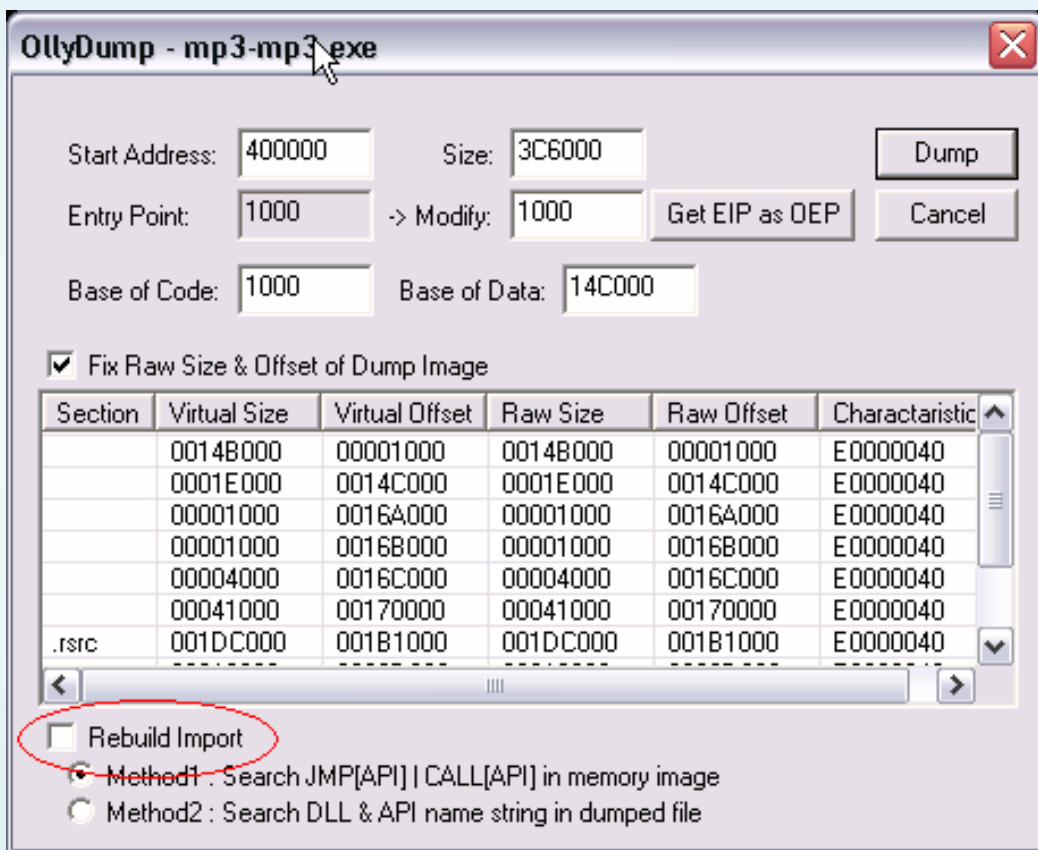
_ The case of a breakdown of the A T s t r a t: 0056 C 15 0

IAT S i z e: 0000113 4

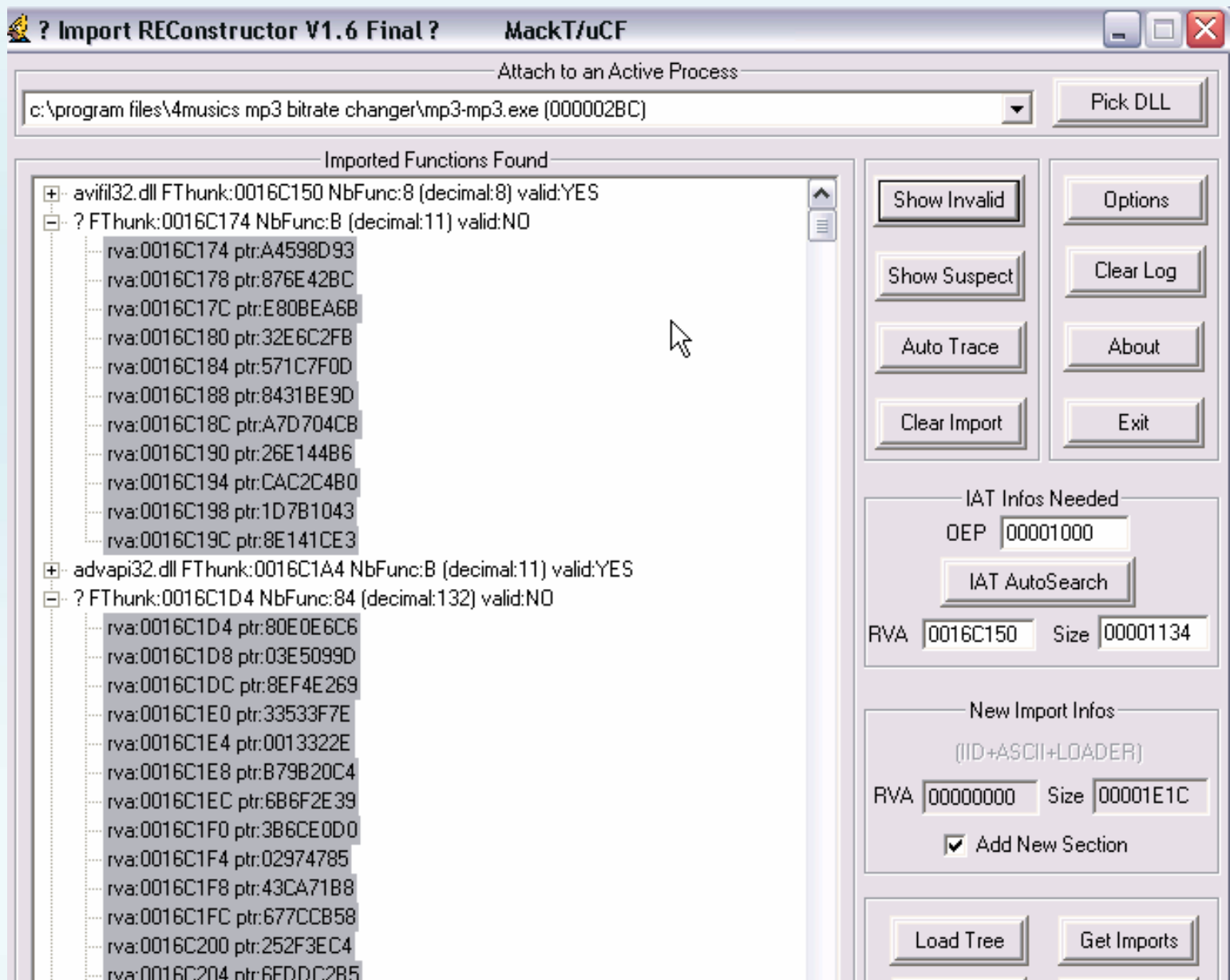
Yg _Ba at the hun g a t t I m to O P E. Han N A L T + M and S T E M E m o r y 1 B r e a k p o i n t o n A c c e s s o f E N S E C T I C o n o d e

Address	Hex dump	Disassembly	Comment
00401000	\$ EB 10	JMP SHORT 00401012	<==OEP
00401002	. 66:623A	BOUND DI,DWORD PTR DS:[EDX]	
00401005	. 43	INC EBX	
00401006	. 2B2B	SUB EBP,DWORD PTR DS:[EBX]	
00401008	. 48	DEC EAX	
00401009	. 4F	DEC EDI	
0040100A	. 4F	DEC EDI	
0040100B	. 4B	DEC EBX	
0040100C	. 90	NOP	
0040100D	.- E9 64C75400	JMP 0094D776	
00401012	> A1 57C75400	MOV EAX,DWORD PTR DS:[54C757]	
00401017	. C1E0 02	SHL EAX,2	
0040101A	. A3 5BC75400	MOV DWORD PTR DS:[54C75B],EAX	
0040101F	. 52	PUSH EDX	

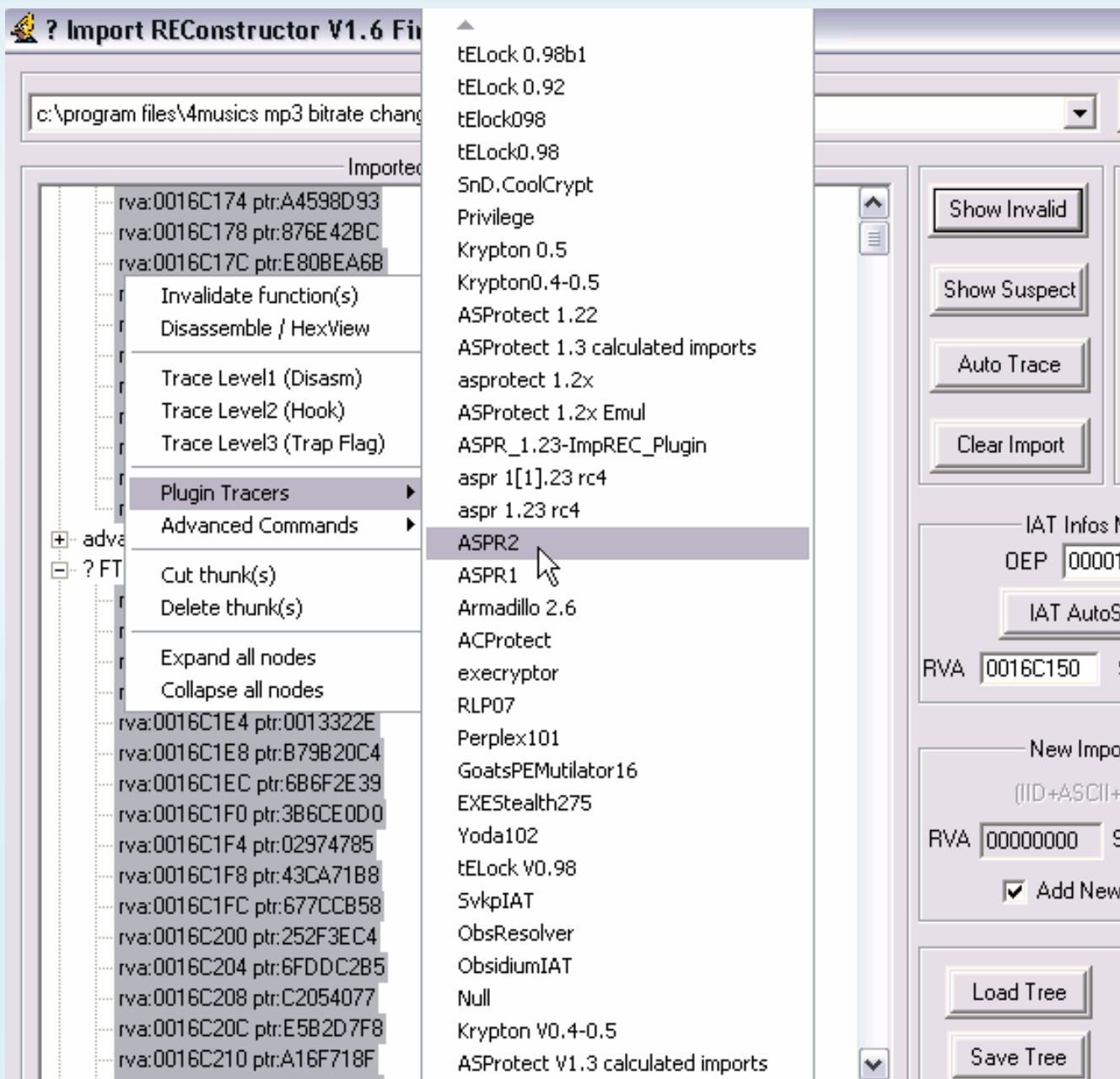
file:///C:/RCE%20Unpacking%20eBook%20[Tra...i]/Unpacking%20ASProtect%202.XX%20SKE.htm (13 of 21) [1/9/2009 9:46:58 LithiumLi]



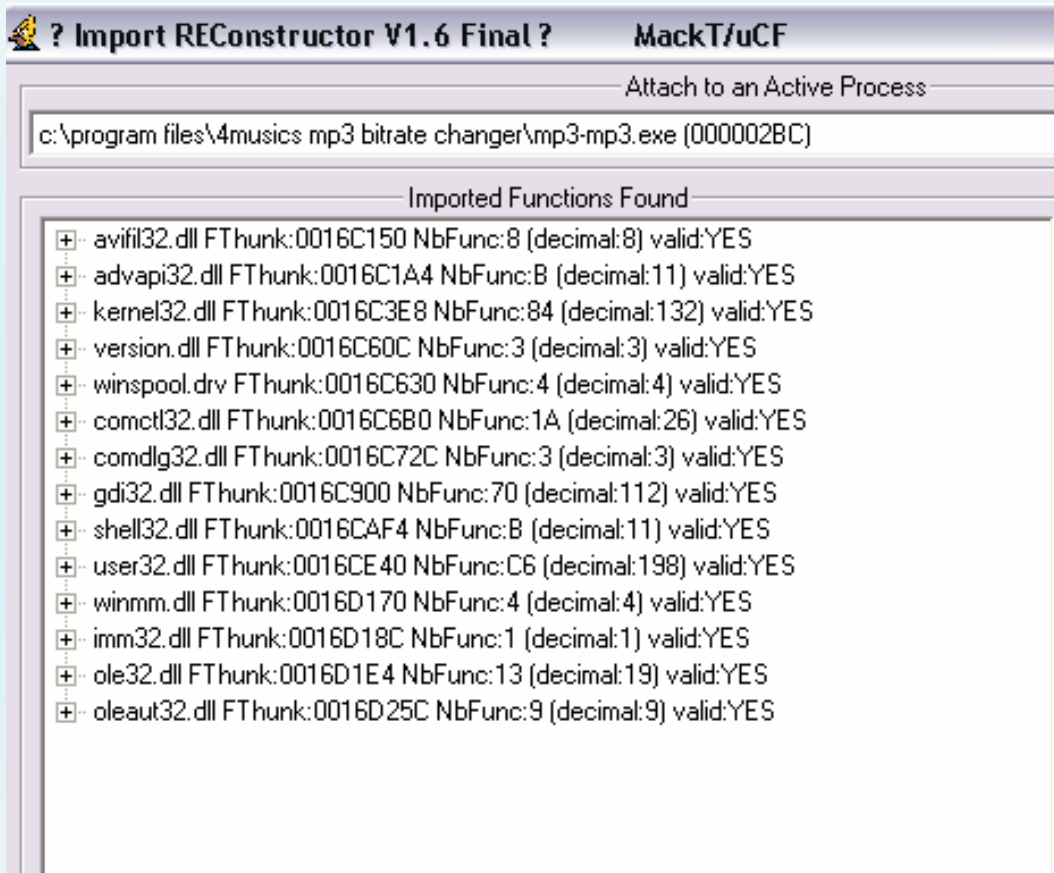
_ Open **ImportREC** and fill **OEP = 1000**, **RVA = 0016C150**, **size = 1134** hit **Get Imports** -> **Show Valid**



_ Click to select and **Plugin Tracers -> ASPR2**



_ Wait for plugin finished running, select **Show Valid-> Cut Thunk (s)**, click to select File Fix dump dump.



_ Or using the File Dumped_.exe



_ Hichic, what teaching heaven ... calm thinking slices 1 *Check* sure this is *file size*. If true nhudu guess we Bypass it Path. Ok, Load File Dumped_.exe and **Oilly** to stop here

Address	Hex dump	Disassembly	Comment
00401000	\$ EB 10	JMP SHORT 00401012	00401012
00401002	. 66:623A	BOUND DI,DWORD PTR DS:[EDX]	
00401005	. 43	INC EBX	
00401006	. 2B2B	SUB EBP,DWORD PTR DS:[EBX]	
00401008	. 48	DEC EAX	
00401009	. 4F	DEC EDI	
0040100A	. 4F	DEC EDI	
0040100B	. 4B	DEC EBX	
0040100C	. 90	NOP	
0040100D	. E9 64C75400	JMP 0094D776	

_N han **F 9, F 12, T A L K** + and C than that of h h h ì n

Address	Stack	Procedure	Called from	Frame
0012F6A0	77D493F5	Includes ntdll.KiFastSystemCallRet	USER32.77D493F3	0012F6D4
0012F6A4	77D6EA24	USER32.WaitMessage	USER32.77D6EA1F	0012F6D4
0012F6D8	77D5688A	USER32.77D6E895	USER32.77D56885	0012F6D4
0012F700	77D6B7C5	USER32.77D567D4	USER32.77D6B7C0	0012F6FC
0012F9C0	77D6B12B	USER32.SoftModalMessageBox	USER32.77D6B126	0012F9BC
0012FB10	77D95FDF	USER32.77D6AFB6	USER32.77D95FDA	0012FB0C
0012FB68	77D96084	USER32.MessageBoxTimeoutW	USER32.77D9607F	0012FB64
0012FB9C	77D80598	? USER32.MessageBoxTimeoutA	USER32.77D80593	0012FB98
0012FBBC	77D80550	? USER32.MessageBoxExA	USER32.77D80548	0012FB88
0012FBD8	004044D7	? <JMP.&user32.MessageBoxA>	dumped_.004044D2	0012FBD4
0012FBEC	00406CFA	? dumped_.004044B0	dumped_.00406CF5	
0012FE80	00406E1B	? dumped_.004044D8	00406E16	
0012FF8C	0054657B	Includes dumped_.00406E1B	00546578	0012FF88

Actualize

Show arguments

Thread

Follow address in stack

Show procedure

Show call

Execute to return

Copy to clipboard

Appearance

Space

Enter

F4

_ And here is to:

Address	Hex dump	Disassembly	Comment
00406CD3	. BA 03000000	MOV EDX,3	
00406CD8	8B85 B0FDFFFF	MOV EAX,DWORD PTR SS:[EBP-250]	
00406CDE	8B08	MOV ECX,DWORD PTR DS:[EAX]	
00406CE0	. FF51 FC	CALL NEAR DWORD PTR DS:[ECX-4]	
00406CE3	. 66:C785 9CFDFFFF 9806	MOV WORD PTR SS:[EBP-264],698	
00406CEC	> 80BD 8BFDFFFF 00	CMP BYTE PTR SS:[EBP-275],0	
00406CF3	~ 75 05	JNZ SHORT 00406CFA	00406CFA
00406CF5	. E8 B6D7FFFF	CALL 004044B0	004044B0
00406CFA	> 8A85 8BFDFFFF	MOV AL,BYTE PTR SS:[EBP-275]	
00406D00	. 8B95 8CFDFFFF	MOV EDX,DWORD PTR SS:[EBP-274]	
00406D06	. 64:8915 00000000	MOV DWORD PTR FS:[0],EDX	
00406D0D	. 8BE5	MOV ESP,EBP	
00406D0F	. 8BB0 74FDFFFF	MOV EDI,DWORD PTR SS:[EBP-28C]	
00406D15	8B85 78FDEEEE	MOV ESI,DWORD PTR SS:[EBP-288]	

_ Set 1 in Bp **406CEC**, press **Ctrl + F2, F9** we stopped at BP Set

00406CE0	. FF51 FC	CALL NEAR DWORD PTR DS:[ECX-4]	
00406CE3	. 66:C785 9CFDFFFF 9806	MOV WORD PTR SS:[EBP-264],698	
00406CEC	> 80BD 8BFDFFFF 00	CMP BYTE PTR SS:[EBP-275],0	
00406CF3	~ 75 05	JNZ SHORT 00406CFA	00406CFA
00406CF5	. E8 B6D7FFFF	CALL 004044B0	004044B0
00406CFA	> 8A85 8BFDFFFF	MOV AL,BYTE PTR SS:[EBP-275]	
00406D00	. 8B95 8CFDFFFF	MOV EDX,DWORD PTR SS:[EBP-274]	

P_ a separated

00406CF3 75 05 JNZSH OR 00406TCFA

See the

00406CF3 EB 05 JMPSHORTTC00406FA

_ Click 00406D22 press F8 to F7 we come

Address	Hex dump	Disassembly	Comment
00406E16	. E8 BDD6FFFF	CALL 004044D8	004044D8
00406E1B	. 84C0	TEST AL,AL	
00406E1D	~ 75 14	JNZ SHORT 00406E33	00406E33
00406E1F	. 33C0	XOR EAX,EAX	
00406E21	. 8B95 0CFFFFFF	MOV EDX,DWORD PTR SS:[EBP-F4]	
00406E27	. 64:8915 00000000	MOV DWORD PTR FS:[0],EDX	
00406E2E	~ E9 680C0000	JMP 00407A9B	00407A9B
00406E33	> 66:C785 1CFFFFFF 0800	MOV WORD PTR SS:[EBP-E4],8	
00406E3C	. C605 3FCC5400 00	MOV BYTE PTR DS:[54CC3F],0	
00406E43	. 66:C785 1CFFFFFF 1400	MOV WORD PTR SS:[EBP-E4],14	
00406E4C	. 68 D0A45400	PUSH 54A4D0	Entry address
00406E54	. 6A 00	PUSH 0	

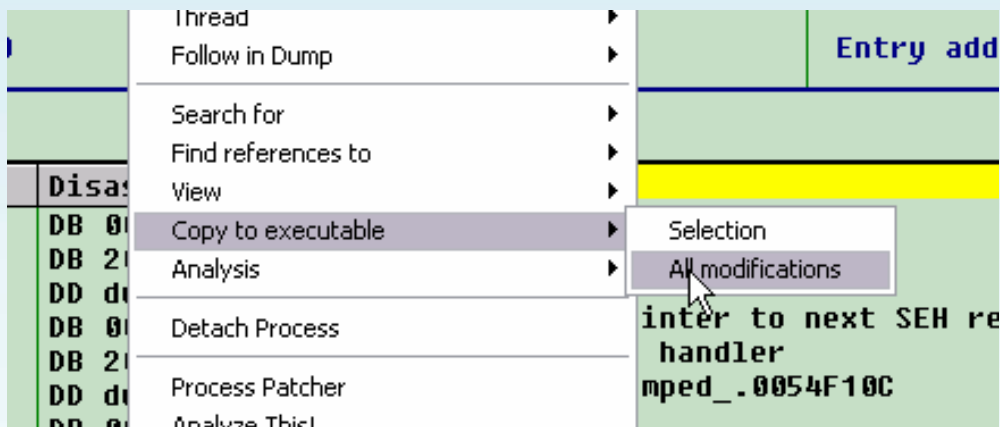
P_ a separated

00,406E1D 75 14 JNTZSH OR 00406E33

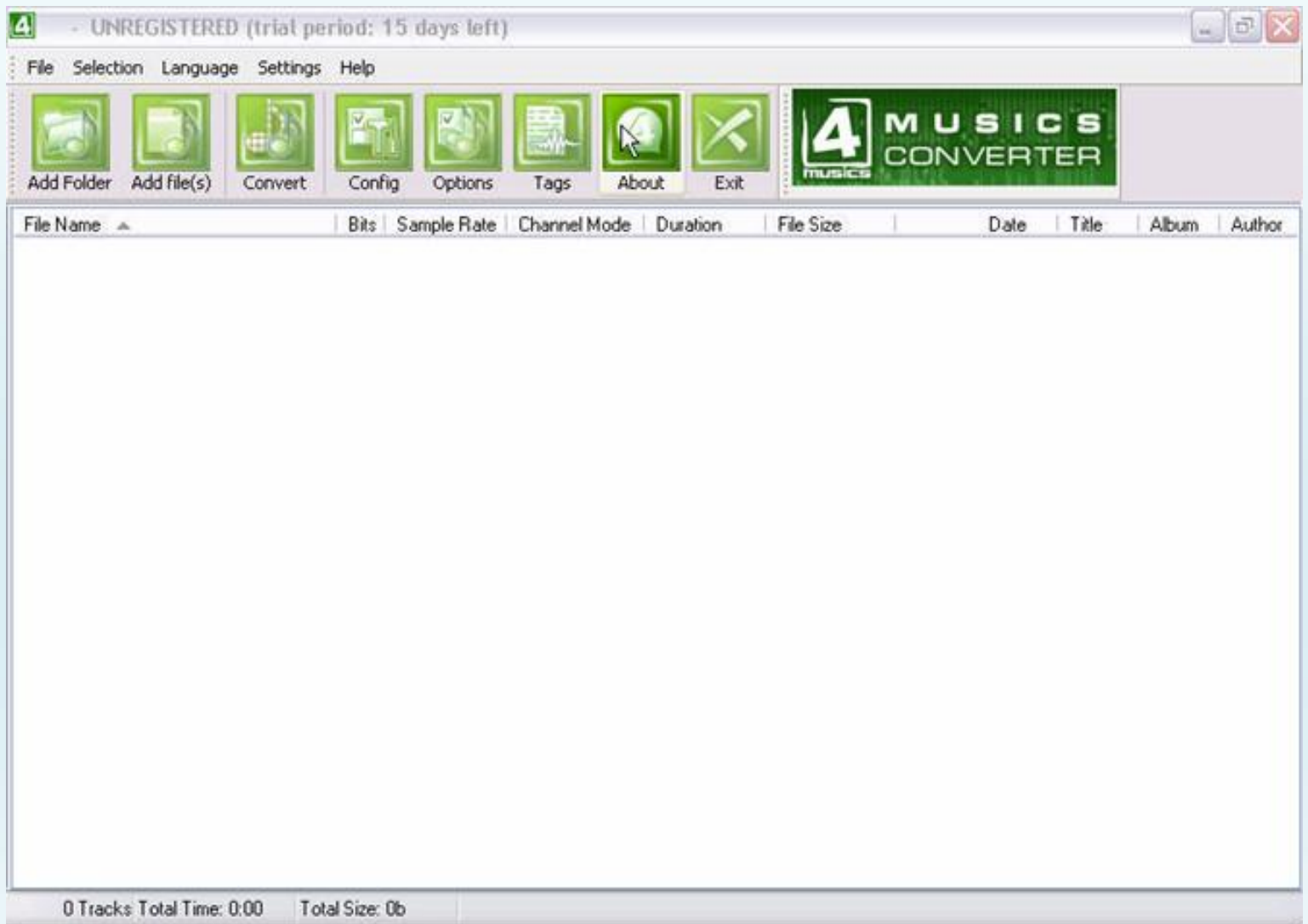
See the

00,406E1D EB 14 JMPSHORT00406E33

_ S a u save it again



_ Temporary File name for this new Dumped_1.exe is, and try running



_ As we have already put the NAG error message. **PEiD** Using the File Scan Dumped_1.exe



_ H a h a h a h a n U p a c k e d c s u c e l l s s f u ...

G r e e T s i t a l y O u F l t: *C o m p u t _ A n e r l e g , e Z o m b i , M o o n b y a , c H a g r a p e , B e n i n a k e i n a m n o w a r , o i Z , D x u e , M e r c , i l e g H T p h o w h e r e x , c k y b o T r i i t a l y , T a k a d a i a m i d i o t , t h e l i g h t o e n i x , t h e i n t h a n d n e , a n d V o l X ... y o u !*

N h a T r a n g , t h e à i t a l y 2 8 s e e t h e 4 - y e a r 2 0 0 6

Why N o t B a r



I. I n t r u c o d t i o n:

After completing TUT **"Unpacking & Cracking RAR Repair Tool 3.0"**

I think the society is ... But i really doubt, many brothers and external Water Passion with Packer has IM and email to them a lot but accomplishing them are the following: *"Your method is or if OEPfinder vX.YZ by deroko ko work is how to implement nhuTUT"*. Question or are asked and also accidentally hit in the children's nhot ... hehe ... because they also pain bít comet. Thui jokes, Method which applies only 2.2x which EXEcryptor only, but with the 2.3x EXEcryptor need another 1 ... How do nhuthe the subject will be more difficult to see clearly

II. T o o l s:

•Tool and Plugin to use:

- LYDL O B G _ E x c e r i t a l y p t o r 1 s t 1 0
- O T P R E C O N i D v T h e 5 t h 1 E
- s k E a T x p l o r e r I
- B O D i g S c r 1 p t. 4 8
- I m p o r t R E C 1. 6 F
- R G D P a c k e r e t D e c t o r a n d 0. 6. 4
- F C x F E p l o e r r V

- Script: *"Exectryp or 2. X I A T bui r l e d e r"* by K a G and the 1st 1.
- Target: **GoldenFTP Server Prov2.80**
- Home page: <http://www.Goldenftpserver.Com>



III. Unpack the i c k:

_A job using familiar **RDG Packer Detector v0.6.4** Detect Target

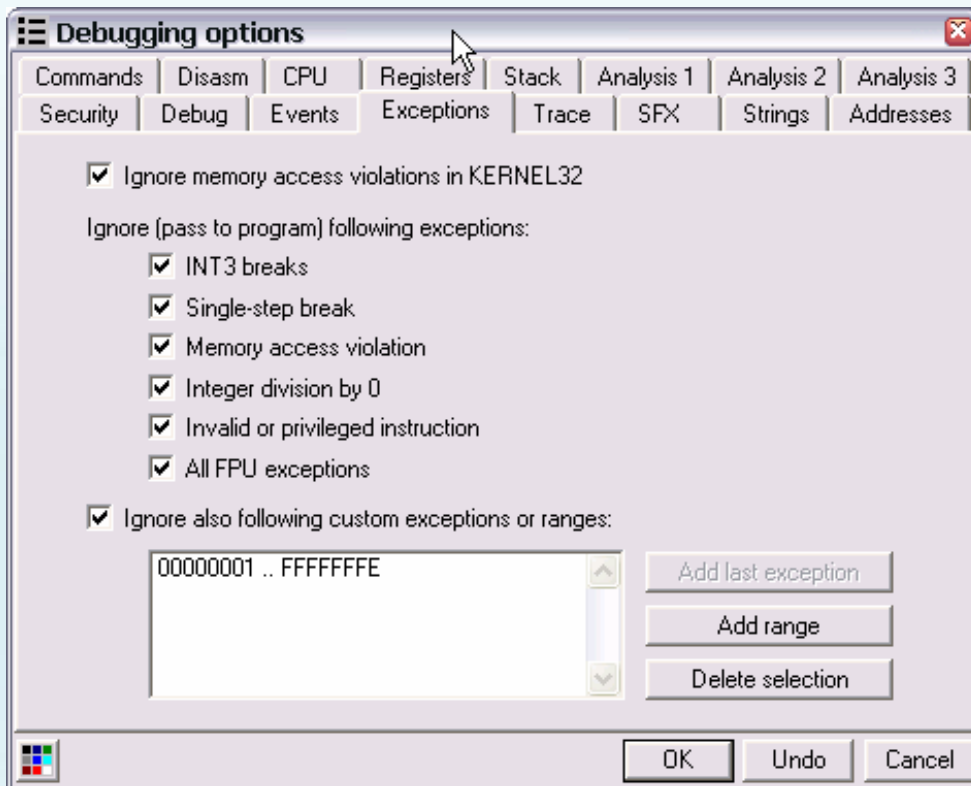
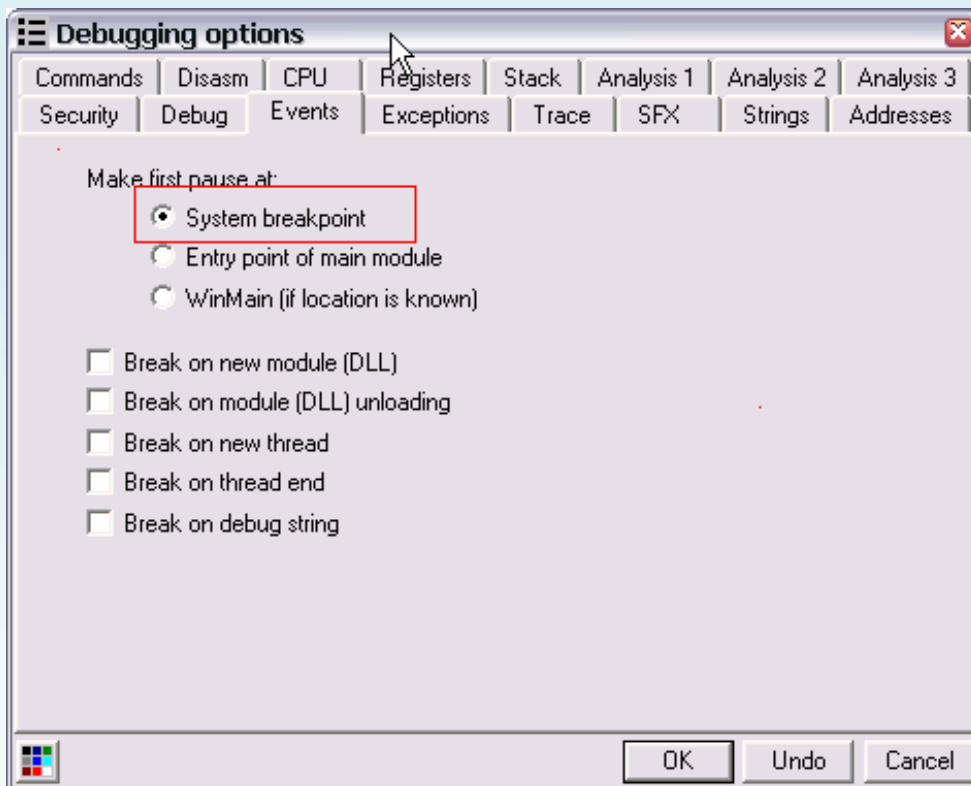


_ If *OEPfinder* vX.YZ used by the Target *deroko* do this is ... the very form it is the Pack with **EXEcryptor 2.3x**. Make sure there are many bottles, he asked why they know well ... hehe ... Nothing subliminal they use **ID PROTECTION v5.1e** and the following results:



_ This result is also the answer why do *OEPfinder* work. And the children know the **Deroko** Update tools will do this again So we must thui play by hand. Open **OillyDBG_EXEcryptor**, press **Alt + O** and configuration as

s a u:



_Target Load to **OllyDBG_EXEcryptor** and you'll stop here:

7C901231	C3	RETN	Registers (FPU)
7C901232	8BFF	MOV EDI,EDI	EAX 00251EA4
7C901234	90	NOP	ECX 00000007
7C901235	90	NOP	EDX 00000080
7C901236	90	NOP	EBX 7FFD4000
7C901237	90	NOP	ESP 0012FB20
7C901238	90	NOP	EBP 0012FC94
7C901239	CC	INT3	ESI 00251F18
7C90123A	C3	RETN	EDI 00251EA4

_ Next to his brother SET tua **Breakpoint** Trick of the new Soft
Run entirely on Memory ko that was Crash. As they say should use lười
Scripts **"Bypass AntiDBG"**

d a t a:

*RH and I nst a nce and
a r e s e c o d G V a g e r
VMS
v e r a r a P V P V o
e e a v e r y m p*

c o d e:

*g p a "W i t h a r t u I F r e e", "R N k e l e 3 2. d l I"
b p w h s \$ U R E S L T, "x"
r u n
b p c h w \$ U R E S L T
r t u
m i g p i e, M O D B U L E A S E m o v I
n s t h a n c e, E \$ R S T U L T m o v e
m p, \$ R E S U L T
d d e t a p m, 3 C
m o v e m p t [t e m p]
A d d e m p t, h I n s t a n c e
A d d t e m p 2 8*

m o v e m p t [t e m p] a d d e m

p t, h I n s t a t e n c e b c m p

mov e p t e mp

mi e g e m p i, M O M E B A R Y S E

mov co d e s e g, \$ R E S U L T

find \$ E S R U L T, E # 2 C C # 9D

mov [\$ R E S U L T], # 2 in the 90 #

gpa "u m e n W i s w o n d", "u s e r 32. dll"

mov [\$ R E S U L T], # 8 B C 09 C 85 C 09 D 20800 # 057856341 2C

pag "C r e t e a t h r e a d", "k e l e R N 32. dll"

find \$ E S R U L T, # FF7518 #

mov [\$ R E S U L T], A # 6 # 0490

pag "Z W C e r a t e a d e G L N", "to the dll. d ll"

BP \$ R E S U L T

loop 1:

run

m p e c i p, \$ R E S U L T

jne loo p 1

bc \$ R E S U L T

BP e loop p 2:

run

m p e c i p p e j ne

loop2 bc e p

mov e t MP, co d e s e g t e

mp sub, 1

gm e t e m each, and M E M O R Y B A S E

mov VM s e g, \$ R E S U L T

gm e t e m each, and M E M O R I Y S Z E

b p e r m v m s g, \$ R E S U L T

run

b P mc

mov e p o e x a STI

b e p o p r m, 1

loop 3:

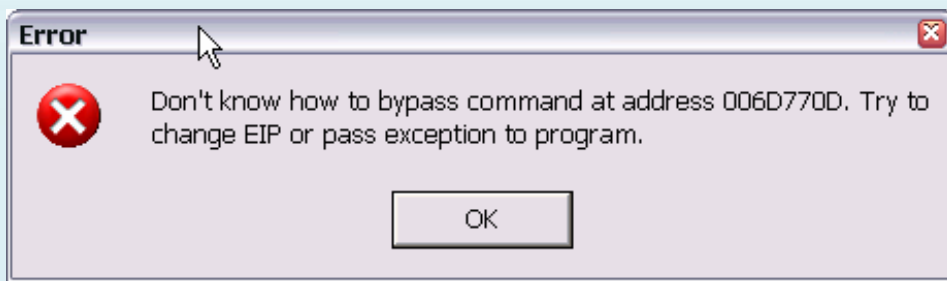
run

c m p p e i, p j e o ne

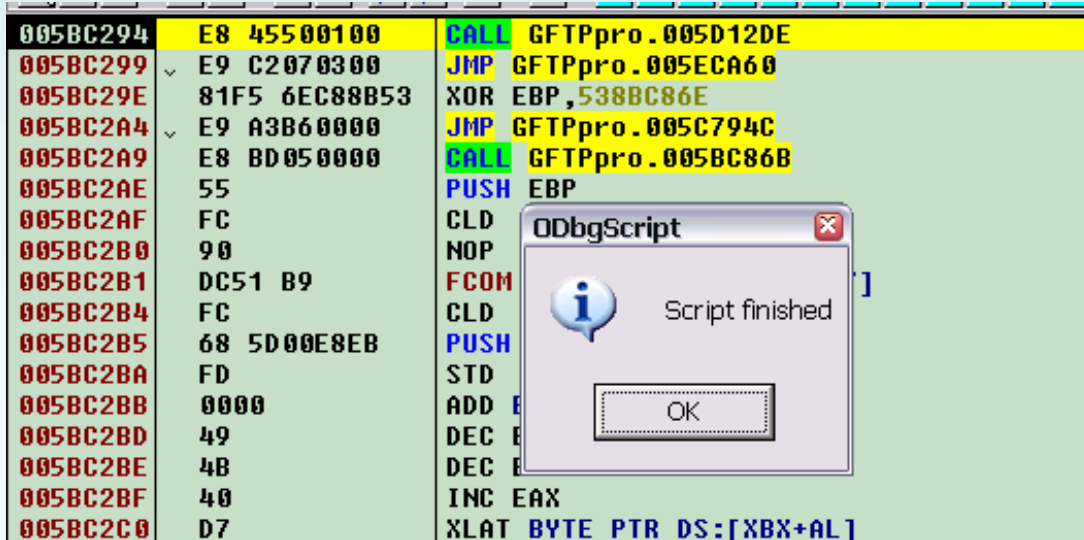
loop3

b p e r t mc

Scripts _Run some time to appear this message



_ Click **OK**, **Shift F9** until coming here:



_ Press **Shift + F9** Soft will **run** completely

The screenshot shows a debugger window with assembly code on the left, registers on the right, and a GoldenFTP Server trial window in the center.

Assembly Code:

Address	Value	Instruction
00401000	EB 10	JMP SHORT GFTPpro.00401012
00401002	66:623A	BOUND DI,DWORD PTR DS:[EDX]
00401005	43	INC EBX
00401006	2B2B	
00401008	48	
00401009	4F	
0040100A	4F	
0040100B	4B	
0040100C	90	
0040100D	E9 C875	
00401012	E9 7DB2	
00401017	8905 68	
0040101D	8D05 2C	
00401023	C600 C3	
00401026	E9 01B8	
0040102B	5A	
0040102C	E9 20DF	
0059C82C	GFTPpro.0	

Registers (FPU):

Register	Value
EAX	00000000
ECX	00000001
EDX	F602F628
EBX	00000000
ESP	0012E544
EBP	0012E630
ESI	00252E98
EDI	0012E604
EIP	7C90E96C
C 0	ES 0023 3:
P 1	CS 001B 3:
A 0	SS 0023 3:
Z 1	DS 0023 3:
S 0	FS 003B 3:
T 0	GS 0000 N:
D 0	
O 0	LastErr E:

GoldenFTP Server Professional Trial Window:

GoldenFTPServer.com - Golden FTP Server professional

golden ftp server

PROFESSIONAL

Trial Version 2.80

Why wait any longer?
[Order Now!](#)

Your 21-days Golden FTP Server pro trial period has 18 days remaining.

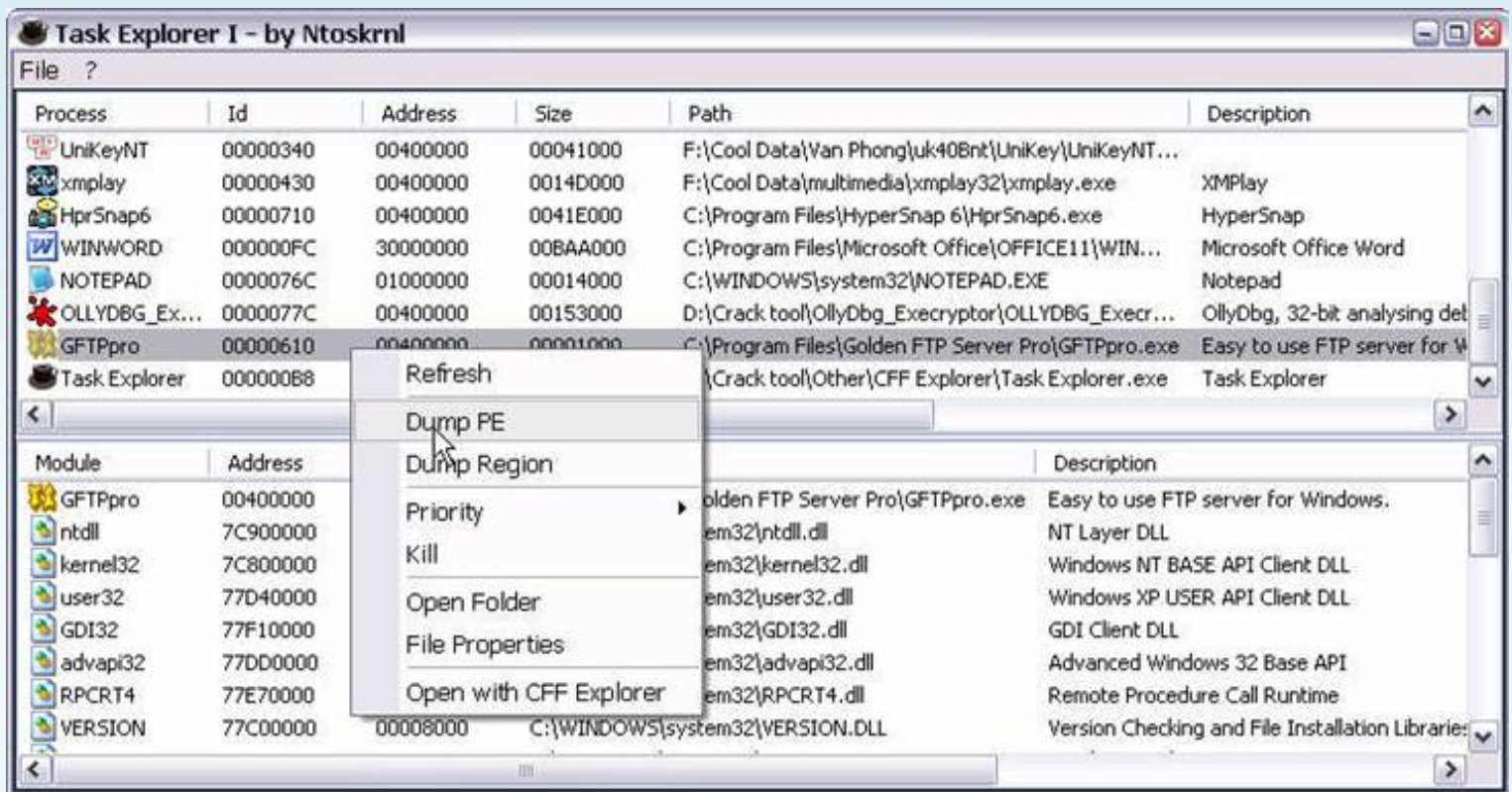
+ Instant Delivery Download immediately after placing your order

+ Safe and Secure Your online order is private, encrypted and secure

[Order Now!](#) [Order Later](#)

GFTPpro.004A6F58

_NhuTut Just before you dump file and use **RDG Packer Detector v0.6.4** scan to determine the Soft Code with what? This helps us find **OEP** for this target.



_ Now need to find Soft is code with **Borland C + + 1999**. She would like the us 1 soft computer that is also a **Winrar 3.6**



Lo_a n d Wi rar and O O D B Il italy G

00401000	EB 10	JMP SHORT WinRAR.00401012	
00401002	66:623A	BOUND DI,DWORD PTR DS:[EDX]	
00401005	43	INC EBX	
00401006	2B2B	SUB EBP,DWORD PTR DS:[EBX]	
00401008	48	DEC EAX	
00401009	4F	DEC EDI	
0040100A	4F	DEC EDI	
0040100B	4B	DEC EBX	
0040100C	90	NOP	
0040100D	E9 B4014A00	JMP 008A11C6	
00401012	A1 A7014A00	MOV EAX,DWORD PTR DS:[4A01A7]	
00401017	C1E0 02	SHL EAX,2	
0040101A	A3 AB014A00	MOV DWORD PTR DS:[4A01AB],EAX	
0040101F	52	PUSH EDX	
00401020	6A 00	PUSH 0	
00401022	E8 B3D90900	CALL <JMP.&KERNEL32.GetModuleHandleA>	
00401027	8BD0	MOV EDX,EAX	

1_or rather it is soft to the Code by **Borland C + + 1999** are
Entrypoint is 1000. This may think of the OEP is **GFTPpro**
 1000. This is soft Nhuday ko **Stolen bytes** are tired ... that Time to
 find location
 beginning and end of the **IAT**

Address	Value	Comment
004EB100	00000000	
004EB104	00000000	
004EB108	77DD778E	advapi32.InitializeSecurityDescriptor → IAT start
004EB10C	77DD6BF0	advapi32.RegCloseKey
004EB110	005A8B73	GFTPpro.005A8B73
004EB114	005C93B2	GFTPpro.005C93B2
004EB118	005E79D0	GFTPpro.005E79D0
004EB11C	005AB0BC	GFTPpro.005AB0BC
004EB120	005AD745	GFTPpro.005AD745
004EB124	005E9D1D	GFTPpro.005E9D1D
004EB128	77E0D2FD	advapi32.SetFileSecurityA
004EB12C	77DD778E	advapi32.InitializeSecurityDescriptor

_ Scroll down to the bottom we have **IAT End**

Address	Value	Comment
004EBE00	00000000	
004EBE04	77124B59	OLEAUT32.SysAllocStringLen
004EBE08	77124B50	OLEAUT32.SysFreeString
004EBE0C	7714C99D	OLEAUT32.SysReAllocStringLen
004EBE10	77124C3B	OLEAUT32.SysStringLen
004EBE14	771265C4	OLEAUT32.VariantChangeTypeEx
004EBE18	771248C0	OLEAUT32.VariantClear
004EBE1C	7714D348	OLEAUT32.VariantCopyInd
004EBE20	00000000	
004EBE24	00000000	
004EBE28	00000000	
004EBE2C	00000000	

IAT End

_ Press **Ctrl + F2** and Running Scripts "**AntiDBG Bypass**", press **Shift + F9** to when stop here

005BC294	E8 45500100	CALL GFTPpro.005D12DE
005BC299	E9 C2070300	JMP GFTPpro.005ECA60
005BC29E	81F5 6EC88B53	XOR EBP,538BC86E
005BC2A4	E9 A3B60000	JMP GFTPpro.005C794C
005BC2A9	E8 BD050000	CALL GFTPpro.005BC86B
005BC2AE	55	PUSH EBP
005BC2AF	FC	CLD
005BC2B0	90	NOP
005BC2B1	DC51 B9	FCOM
005BC2B4	FC	CLD
005BC2B5	68 5D00E8EB	PUSH
005BC2BA	FD	STD
005BC2BB	0000	ADD E
005BC2BD	49	DEC E
005BC2BE	4B	DEC E
005BC2BF	40	INC EAX
005BC2C0	D7	XLAT BYTE PTR DS:[EBX+AL]



_ Yeah, press **Ctrl + G** to enter and 401,000 Select like

00401000	EB 10	JMP SHORT GFTPpro.004
00401002	66:623A	BOUND DI,DWORD PTR DS
00401005	43	INC EBX
00401006	2B2B	SUB EBP,DWORD PTR DS:
00401008	48	DEC EAX
00401009	4F	DEC EDI
0040100A	4F	DEC EDI
0040100B	4B	DEC EBX
0040100C	90	NOP
0040100D	E9 C8754C00	JMP 008C85DA
00401012	E9 7DB21B00	JMP GFTPpro.005BC294
00401017	8905 68B34E00	MOV DWORD PTR DS:[4EB
0040101D	8D05 2CC85900	LEA EAX,DWORD PTR DS:

Binary	
Assemble	Space
Label	:
Comment	;
Breakpoint	
Run trace	
Follow	Enter
New origin here	Ctrl+Gray *
Go to	
Follow in Dump	

_ Time running Scripts "**Execryptor 2.x IAT rebuilder** to Fix IAT but remember

Edit the nhusau

```

mov temp5,esp
mov IATstart,004EB108
mov IATend,004EBE20

```

IAT Start

IAT End

_Ok, time and Run Scripts waiting finished running we'll get the **Full IAT**

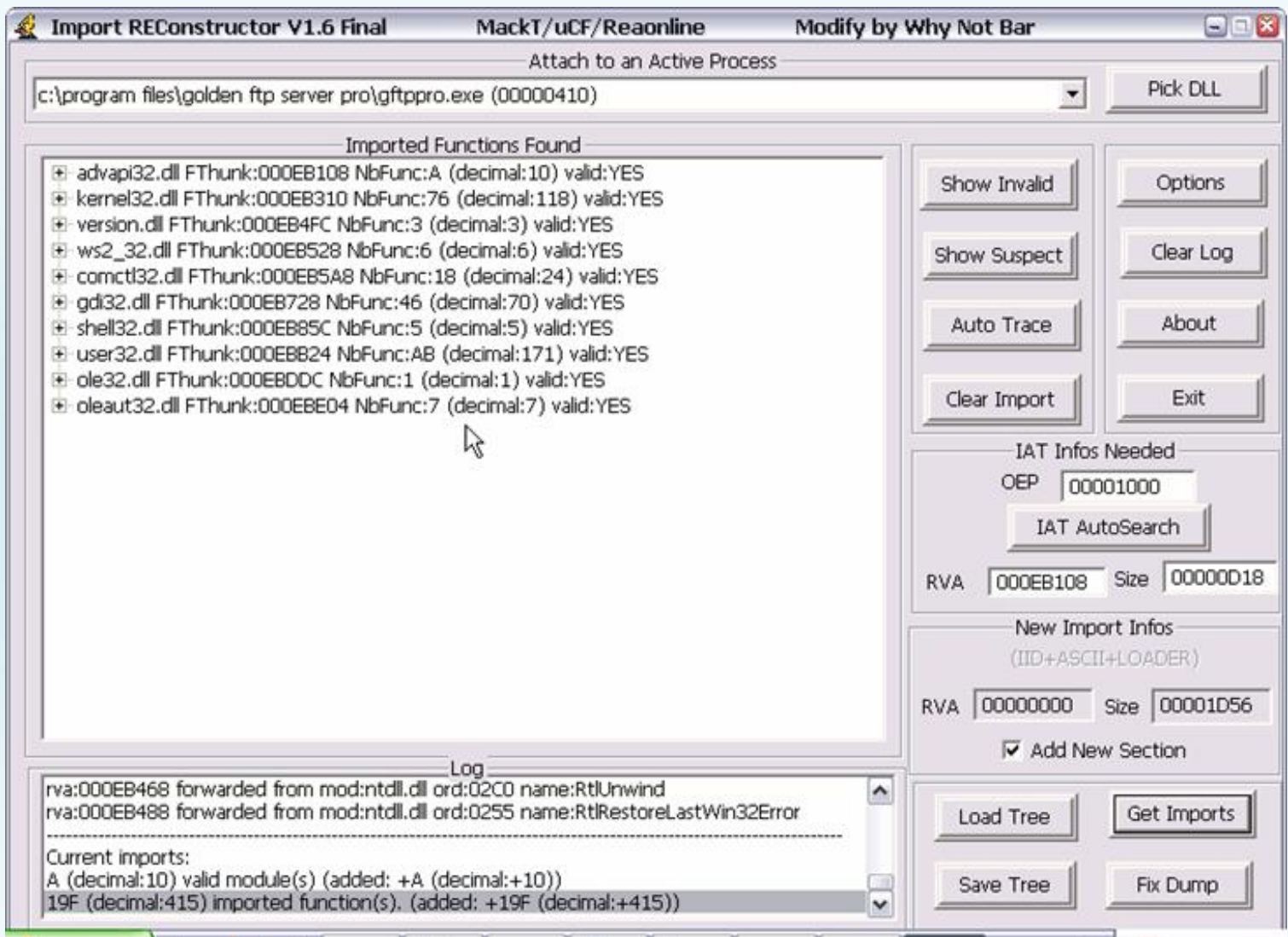
77D4E7B8	B8 B1110000	MOV EAX,11B1	
77D4E7BD	BA 0003FE7F	MOV EDX,7FFE0300	
77D4E7C2	FF12	CALL NEAR DWORD PTR DS:[EDX]	
77D4E7C4	C2 0800		
77D4E7C7	80FA FF		
77D4E7CA	0F84 BF6C0100		
77D4E7D0	E9 769DFFFF		
77D4E7D5	66:2106		
77D4E7D8	E9 710A0000		
77D4E7DD	90		
77D4E7DE	90		
77D4E7DF	90		
77D4E7E0	90	NOP	
77D4E7E1	90	NOP	

ODbgScript

Script finished

OK

Dumped _Full time more and more open ImportREC enter parameters



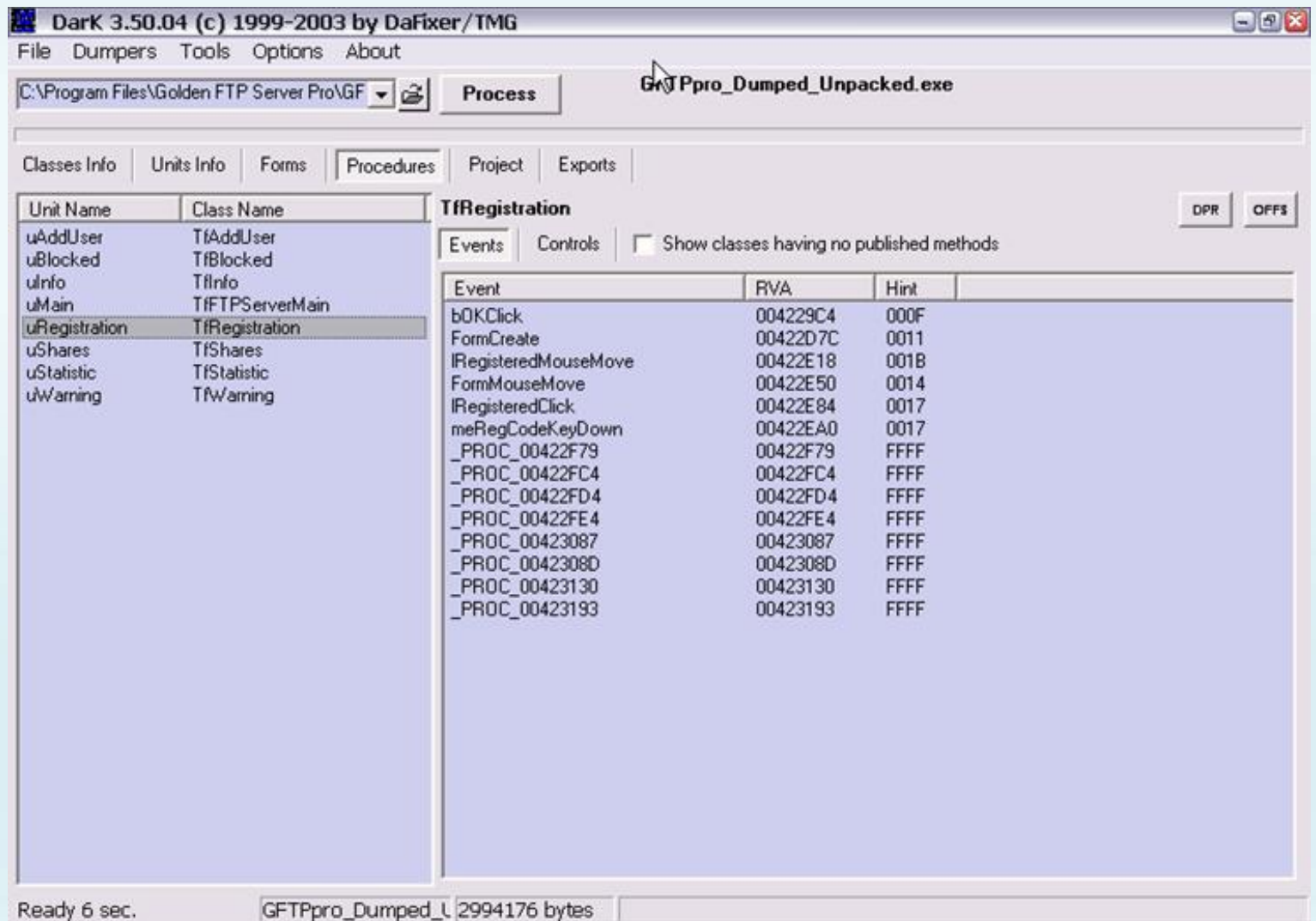
Ko has any function should **Fix Invalid dump** nhubinh often, try Run GFTPpro_Dumped_.exe khua khua *Unpacked* nhungua run **su cce ssfu I I!**

I V. Cr ac k i n g:

_The Crack found this line too afraid I have to do is to Cracker đành franchise for a few ... Bro.

004031C5	MOV EDX,GFTPpro_.004C7F47	.log
00403275	MOV ESI,GFTPpro_.004C7F55	Default
00403295	PUSH GFTPpro_.004C7F5D	unregistered
004032A6	PUSH GFTPpro_.004C7F6A	unregistered
00403652	PUSH GFTPpro_.004C7F94	:)
00403657	PUSH GFTPpro_.004C7F77	Hi, cracker! :) How are you?
00403D56	MOV EDX,GFTPpro_.004C8006	.cfg
00403E1A	MOV ESI,GFTPpro_.004C88D9	Golden FTP Server Pro
00403EF0	MOV ESI,GFTPpro_.004C88F1	Golden FTP Server
00403F40	PUSH GFTPpro_.004C8903	FTPServer
00403F61	PUSH GFTPpro_.004C890D	TfFTPServerMain

DeDe is _Dung lot of What's the treatment from the ... "I do not listen, they do see, they do what bit brothers are Crack ... "



_Byebye In brothers in other tut

GRE T e s t Fly Ou: *Co m put _ An e r l e g, e Zo m belgium, each on a by b, c H a n ho, nin B e a k i a m e n no W A R, Z o i, D e u x, M r c e, i l e g HT pho where x, c k ybo Tri italy, T a k a d a i a midiot, and e nth a n d i n e, a nd you ...!*

Nha Tr a n g, italy 2 à The 5th Asia ng8 year 200 6

Whot italyN Bar

Unpacking the Flash Recovery 2.35



I -- Introduction:

Flash Recovery is a 2:35 ExeCryptor by 2.3x extremely difficult to unpack, but it is a very Soft brother or review of its features:

"DiskInter the als Flash recovery is a flash memory file recovery tool that every digital camera owner

should have and handy. Essentially, the program is a marijuana" first aid kit the "ford of the digital photograph

RSA and a domestic rescue team when it's needed. Unfortunately, although the program is free

a den of allydeletedorlost uetohardw a re d (e committed in italy rmemor) malfuncti n.Som o e e s heart,

flashm moryg e e e tsr - formatted.Theg o n od ewsisthatinallofthe s ecasesthe imagesarenotlo s tD

skInternalsFlashRec i o n verycanu deletea estorepho r n d t h ograp sinamatterof secondsorminu s. e t o w

Thisish it w orks. ouconnect Y italy ourc me a aorflashmemorysticktoPC.C o r o mputerrec gnizes it a s

anexternaldisk . T he rogramt p h t enstar ssc a nningme m ory, owi n s h u geverypict rethatc a nbe

recovered. All of them are done with the same EFR evaluation version of DiskInternals Flash Recovery. To

save these images, you need to register the program. DiskInternals Flash Recovery is not just simple,

the it's a simple. There is no need to set any option of the entire process done automatically

with a built-in Photo Recovery wizard. The program records the original images from hard drives,

external drives, or a memory card devices such as SmartMedia, CompactFlash, Memory Stick,

MicroDrive, xD Picture Card, FlashCard, PC Card, Multimedia Card, Digital Card, Secure,

and many others. "



!! -- Theools T&g e r a t:

• **T oo L and P lugin c a n d n g s:**

- *LY D L O B G _ E x c e r i t a l y p t o r 1 s t 1 0*
- *OTPRE C O NiDv The 5th 1 E*
- *skE a T x p l o r e r I*
- *B O D i g S c r 1 p t. 4 8*
- *I m p o r t R E C 1. 6 F*
- *FCx FE p l o e r r V*

• **One g e r t:** *F i s a s h R e c o v e r y 2 n d 3 5*

HT_t_p: / / _ w _ w w . d i i n t e r s k n a l s . c o m /

II I - B i t a l y s s P a n a t i g u D e b , D u m p i l e F & F i n d O P E:

1st D ù n O T E G P R I C T I O N D v5. 1 f s c a n o f a g e r t



2nd O K, L o a d i n a g e r l t v a o O l y D B G _ E X E c r i t a l y p t o r a n d w e s t o p h e r e:

Address	Hex	Disassembly	Comments
0072685E	E8 EBF0FFFF	CALL FR.0072674E	
00726863	05 36060000	ADD EAX,636	
00726868	FFE0	JMP NEAR EAX	
0072686A	E8 04000000	CALL FR.00726873	
0072686F	FFFF	???	Unknown command
00726871	FFFF	???	Unknown command
00726873	5E	POP ESI	
00726874	C3	RETN	
00726875	0017	ADD BYTE PTR DS:[EDI],DL	
00726877	05 0B5A197A	ADD EAX,7A195ADB	
0072687C	6A 20	PUSH 20	
0072687E	4E	DEC ESI	
0072687F	8057 A6 E1	ADC BYTE PTR DS:[EDI-5A],0E1	
00726883	3C 7A	CMP AL,7A	
00726885	E0 1D	LOOPDNE SHORT FR.007268A4	
00726887	D372 54	SAL DWORD PTR DS:[EDX+54],CL	

3rd press **Alt + B** 1 Breakpoint you see, please delete it

Address	Module	Active	Disassembly
00726852	FR	One-shot	CALL FR.0072674E
<div>Remove Del</div> <div>Follow in Disassembler Enter</div>			

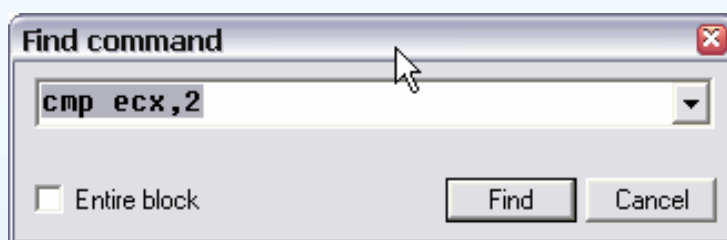
4th press **Alt + M** and press **F2** to set BP in Section 1. Code

003F0000	00008000				Priv	RW	RW
00400000	00001000	FR		PE header	Imag	R	RWE
00401000	0011F000	FR	CODE		Imag	R	RWE
00520000	00004000	FR	DATA	data	Imag	R	RWE
00524000	00007000	FR	BSS		Imag	R	RWE
00528000	00003000	FR	ap8h8390		Imag	R	RWE
0052E000	00001000	FR	tiikvzks		Imag	R	RWE
0052F000	00001000	FR	.tls		Imag	R	RWE
00530000	00001000	FR	rdata		Imag	R	RWE

5. Press **Shift + F9** you stop here

00723609	E8 1F010000	CALL FR.0072372D
0072360E	0245 EF	ADD AL, BYTE PTR SS:[EBP-11]
00723611	AA	STOS BYTE PTR ES:[EDI]
00723612	EB E9	JMP SHORT FR.007235FD
00723614	E8 FC000000	CALL FR.00723715
00723619	0F82 97000000	JB FR.007236B6
0072361F	E8 F1000000	CALL FR.00723715
00723624	73 5B	JNB SHORT FR.00723681
00723626	B9 04000000	MOV ECX, 4
0072362B	E8 FD000000	CALL FR.0072372D
00723630	48	DEC EAX
00723631	74 DE	JE SHORT FR.00723611
00723633	0F89 C7000000	JNS FR.00723700
00723639	E8 D7000000	CALL FR.00723715
0072363E	73 1B	JNB SHORT FR.0072365B
00723640	55	FINISH FRP

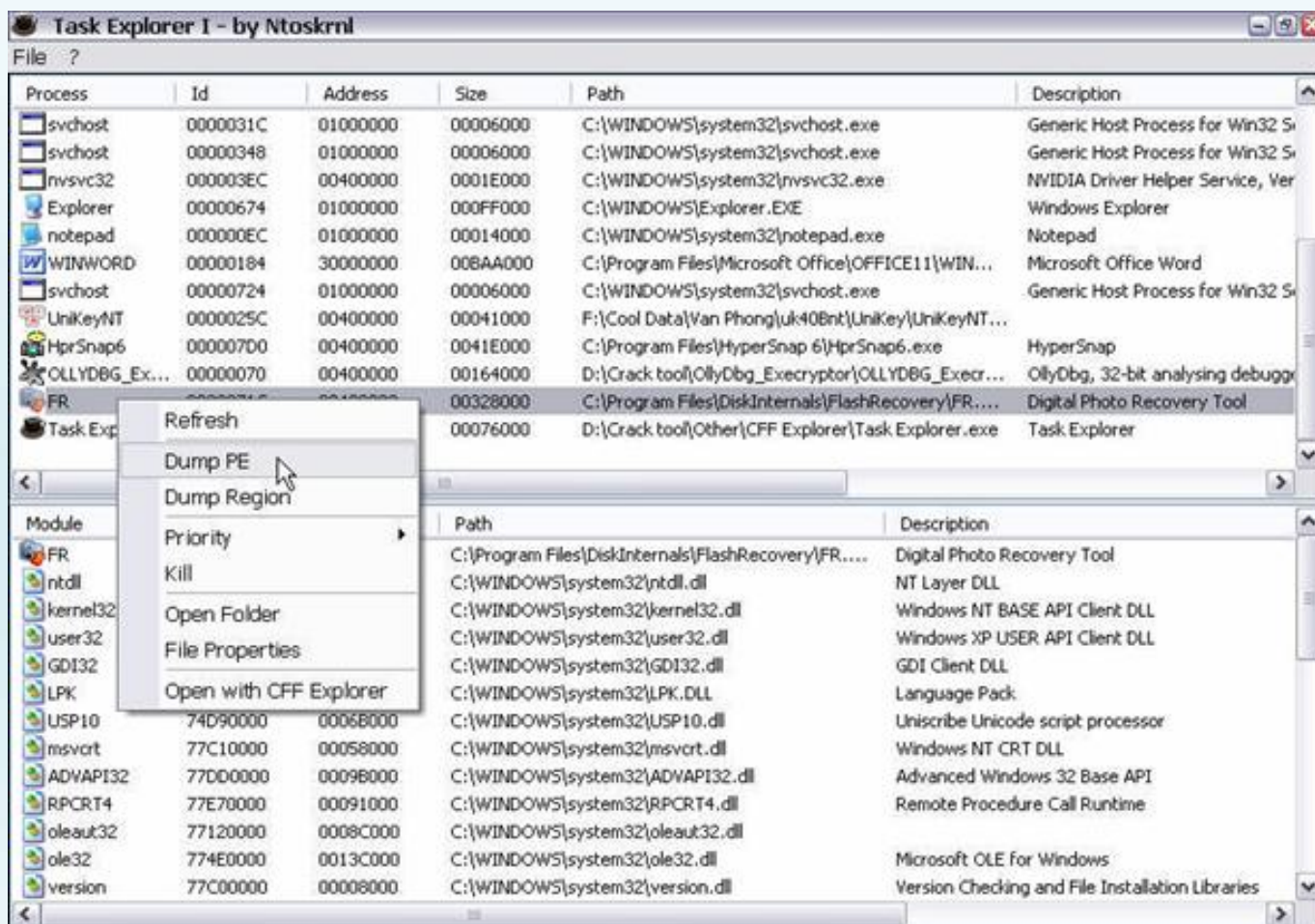
6. Press **Ctrl + F** to enter **Cmp ECX, 2**



7. Click **Find**, press **F2** to set it at **BP**

0072369B	09C0	OR EAX, EAX
0072369D	74 04	JE SHORT FR.007236A3
0072369F	89C3	MOV EBX, EAX
007236A1	EB 5E	JMP SHORT FR.00723701
007236A3	83F9 02	CMP ECX, 2
007236A6	74 66	JE SHORT FR.0072370E
007236A8	41	INC ECX
007236A9	E8 7F000000	CALL FR.0072372D
007236AE	8945 F4	MOV DWORD PTR SS:[EBP-C], EAX
007236B1	E9 47FFFFFF	JMP FR.007235FD
007236B6	E8 7E000000	CALL FR.00723739

8. Press **Shift + F9 17 times** Soft Run will completely. Soft Run When we have identified and **IAT IAT Start End** to conduct **Fix IAT** and i forgot to **dump the Full** Soft considered to be in code to what it can to find **OEP**. First, dump the file has



9. Use **RDG Packer Detector v0.6.4** Scan File **FR_Dumped.exe** and will receive the following information:



10th tolerable, Soft know what the code in our search goi **IAT IAT Start** and **End** is very easy to find nhusau:

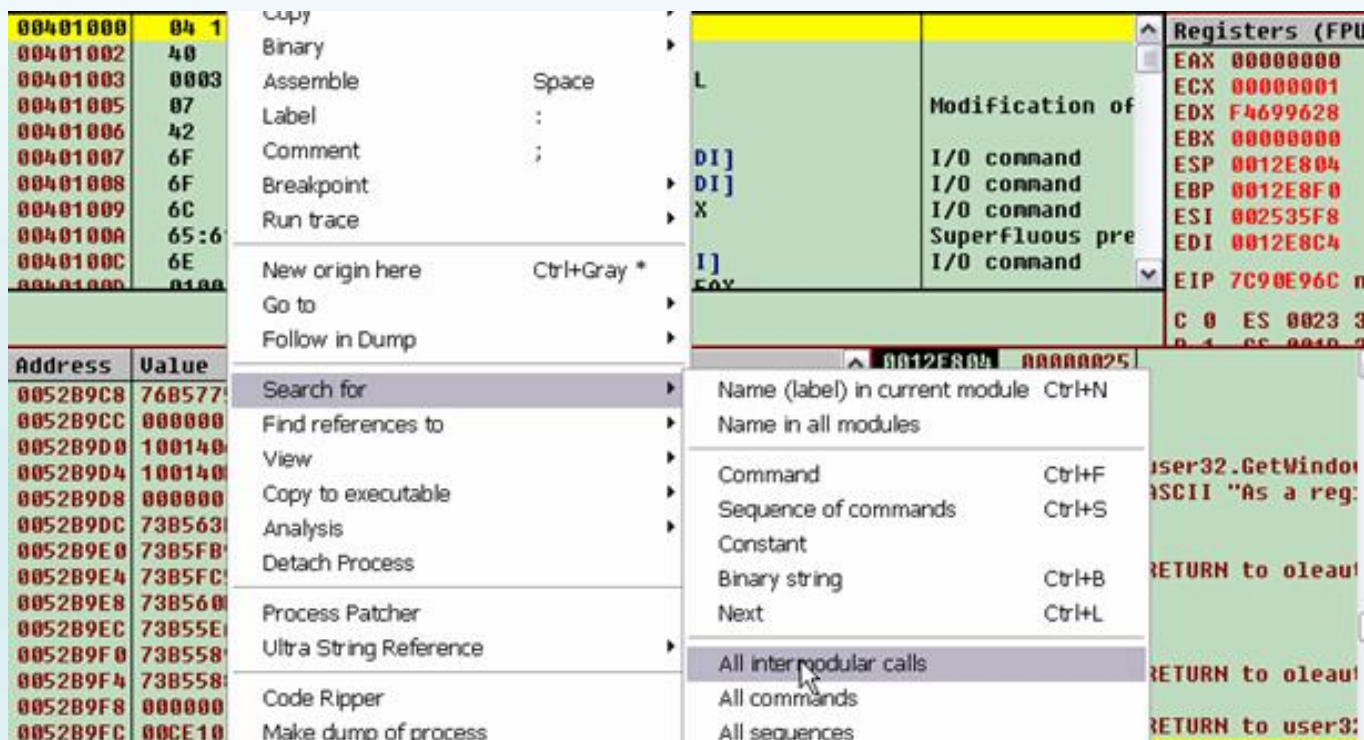
Address	Value	Comment
0052B1C0	00000000	
0052B1C4	00000000	
0052B1C8	00000000	
0052B1CC	0060A9AD	FR.0060A9AD
0052B1D0	0058DC80	FR.0058DC80
0052B1D4	005A9C09	FR.005A9C09
0052B1D8	7C809FA1	kernel32.InitializeCriticalSection
0052B1DC	7C809B14	kernel32.VirtualFree
0052B1E0	7C809A81	kernel32.VirtualAlloc
0052B1E4	005D9581	FR.005D9581
0052B1E8	7C8099BD	kernel32.LocalAlloc
0052B1EC	7C8114AB	kernel32.GetVersion
0052B1F0	7C809737	kernel32.GetCurrentThreadId
0052B1F4	7C809794	kernel32.InterlockedDecrement
0052B1F8	7C80977B	kernel32.InterlockedIncrement
0052B1FC	7C80B859	kernel32.VirtualQuery
0052B200	0058DC80	FR.0058DC80

IAT Start

Address	Value	Comment
0052B9C8	76B5775B	winmm.mmiInstallIOProcA
0052B9CC	00000000	
0052B9D0	10014060	MIG_29.Init
0052B9D4	100140B0	MIG_29.Decode
0052B9D8	00000000	
0052B9DC	73B563B9	AVIFIL32.AVIStreamOpenFromFileA
0052B9E0	73B5FB95	AVIFIL32.AVIStreamGetFrame
0052B9E4	73B5FC55	AVIFIL32.AVIStreamGetFrameOpen
0052B9E8	73B560E7	AVIFIL32.AVIStreamLength
0052B9EC	73B55EA9	AVIFIL32.AVIStreamInfoA
0052B9F0	73B5589F	AVIFIL32.AVIFileExit
0052B9F4	73B55881	AVIFIL32.AVIFileInit
0052B9F8	00000000	
0052B9FC	00CE1000	WiaD11.WiaDeviceName
0052BA00	00000000	
0052BA04	00000000	
0052BA08	00000000	

IAT End

11. As we know Soft code using **Borland Delphi 6.0 - 7.0** Ham have always had **GetModuleHandleA** below. Based on this we quickly find the **OEP** of this Target. In **OlllyDBG** press **Ctrl + G**, enter **401000** and choose the same image



12. GetModuleHandleA and Type 2 function

00405C56	CALL FR.00401340	kernel32.GetModuleFileNameA
00405E90	CALL FR.00401340	kernel32.GetModuleFileNameA
0040C949	CALL FR.00406DB8	kernel32.GetModuleFileNameA
0040C964	CALL FR.00406DB8	kernel32.GetModuleFileNameA
0040D01D	CALL FR.00406DB8	kernel32.GetModuleFileNameA
0047FECF	CALL FR.00406DB8	kernel32.GetModuleFileNameA
00405CD9	CALL FR.00401348	kernel32.GetModuleHandleA
004069D8	CALL FR.00406908	kernel32.GetModuleHandleA
0040E2BA	CALL FR.00406DC0	kernel32.GetModuleHandleA
0040F201	CALL FR.00406DC0	kernel32.GetModuleHandleA
0042EFDA	CALL FR.00406DC0	kernel32.GetModuleHandleA

13. Tatoi Double Click here

004069C5	E8 06F8FFFF	CALL FR.004061D0	
004069CA	C3	RETN	
004069CB	90	NOP	
004069CC	53	PUSH EBX	
004069CD	8BD8	MOV EBX,EAX	
004069CF	33C0	XOR EAX,EAX	
004069D1	A3 CC005200	MOV DWORD PTR DS:[5200CC],EAX	
004069D6	6A 00	PUSH 0	
004069D8	E8 2BFFFFFF	CALL FR.00406908	JMP to kernel32.GetModuleHandleA
004069DD	A3 64465200	MOV DWORD PTR DS:[524664],EAX	
004069E2	A1 64465200	MOV EAX,DWORD PTR DS:[524664]	
004069E7	A3 D8005200	MOV DWORD PTR DS:[5200D8],EAX	
004069EC	33C0	XOR EAX,EAX	
004069EE	A3 DC005200	MOV DWORD PTR DS:[5200DC],EAX	
004069F3	33C0	XOR EAX,EAX	

14. We need to find a command to call call this code, press **Ctrl + F** to enter **004069CC** and click **OK** to us here

0051F038	55	PUSH EBP	<=== OEP
0051F039	8BEC	MOV EBP,ESP	
0051F03B	83C4 F0	ADD ESP,-10	
0051F03E	B8 38EB5100	MOV EAX,FR.0051EB38	
0051F043	E8 8479EEFF	CALL FR.004069CC	
0051F048	E8 833CF6FF	CALL FR.00482CD0	
0051F04D	84C0	TEST AL,AL	
0051F04F	75 30	JNZ SHORT FR.0051F081	
0051F051	A1 08345200	MOV EAX,DWORD PTR DS:[523408]	
0051F056	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0051F058	E8 4323F6FF	CALL FR.004813A0	
0051F05D	8B00 B4325200	MOV ECX,DWORD PTR DS:[5232B4]	FR.0052A2D8
0051F063	A1 08345200	MOV EAX,DWORD PTR DS:[523408]	
0051F068	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0051F06A	8B15 047F5100	MOV EDI,DWORD PTR DS:[517F04]	FR.00517F50

15. Looking up little that is ... very OEP may have ko **Stolen Bytes**. Press **Ctrl + F2** to prepare the next step

VID E-Fix IA & T E R I build mp ort:

_ Repeat the steps from 3 -> 8 in part III, but remember to press **Shift + F9 16 times** for only **17 times** soft **click Run** will. Ok, press **Ctrl + G** to enter **0051F038 (OEP)** and press **Shift + F9** we stopped at the **OEP**. To this, we use the script "**ExeCryptor 2.xx IAT Rebuilder v1.1**" to the Quick Fix IAT, but before you run memory Revising 2 address **IAT IAT Start** and **End** Scripts to run for exactly

```

File Edit Format View Help
mov temp5,esp
mov IATstart,0052B1CC
mov IATend,0052BA00

again:
mov esp,temp5

mov temp2,[IATstart]

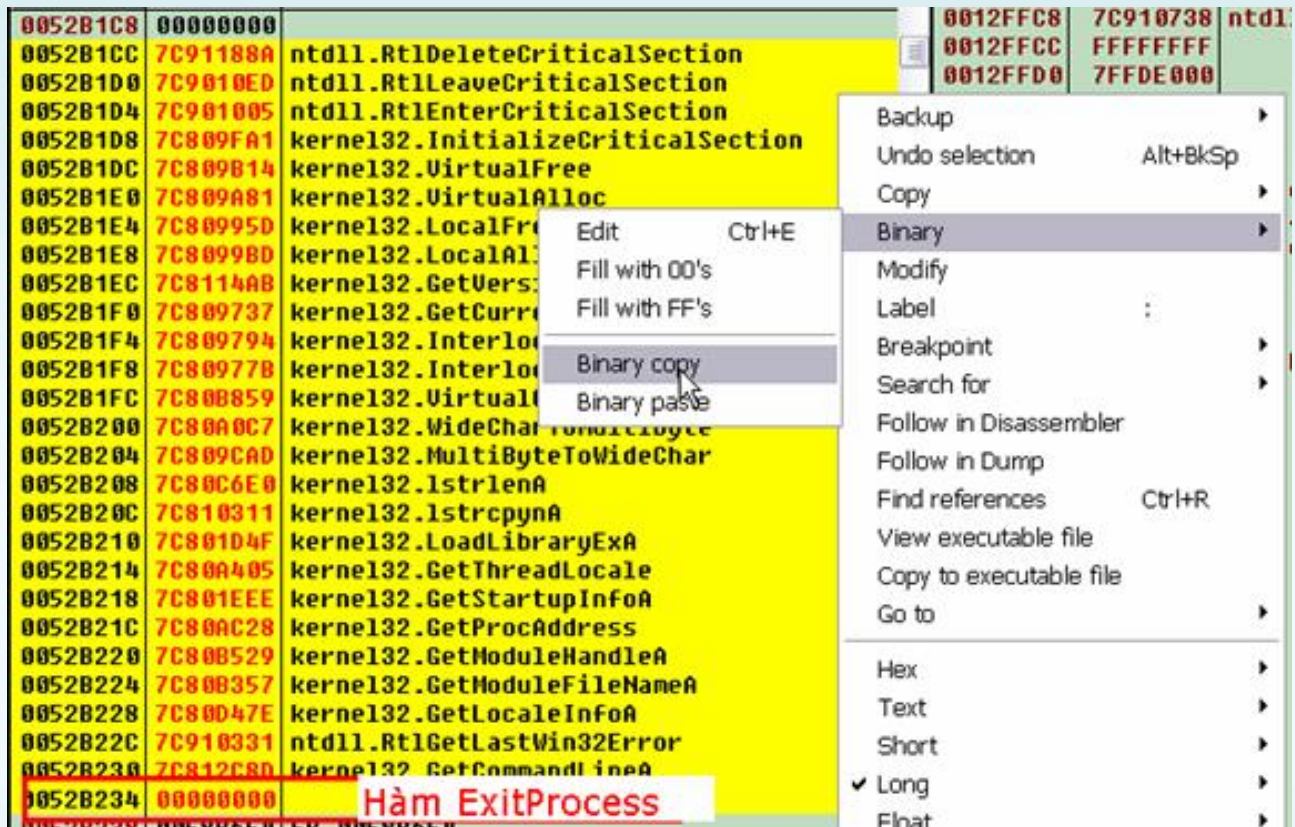
cmp temp2,00000000
je here //in case of zeros,somewhere is a bug...
cmp temp2,50000000
ja here //in case that the IAT has a valid pointer :

```

_ **1:48 ODbgScript** used to run a **script**, running time 1 it follows:



_ Ko stars, click OK, the window we **dump** Copy that the IAT has Scripts and fix **Fill with 00's** for the function "**ExitProcess**" Fix Script because we do not properly function after Fix



_ Ctrl + F2 and Repeat the steps from 3 -> 8 in part III, but remember to press Shift + F9 16 times for only 17 times soft click Run will. Ok, press Ctrl + G to enter 0051F038 (OEP) and press Shift + F9 we stopped at the OEP. In the window dump press Ctrl + G to enter 0052B1CC (IAT Start) and paste the IAT at our new Copy and Run Scripts. But fix it 2 more functions are being Crash ...

0052B1FC	7C80B859	kernel32.VirtualQuery
0052B200	7C80A0C7	kernel32.WideCharToMultiByte
0052B204	7C809CAD	kernel32.MultiByteToWideChar
0052B208	7C80C6E0	kernel32.lstrlenA
0052B20C	7C810311	kernel32.lstrcpynA
0052B210	7C801D4F	kernel32.LoadLibraryExA
0052B214	7C80A405	kernel32.GetThreadLocale
0052B218	7C801EEE	kernel32.GetStartupInfoA
0052B21C	7C80AC28	kernel32.GetProcAddress
0052B220	7C80B529	kernel32.GetModuleHandleA
0052B224	7C80B357	kernel32.GetModuleFileNameA
0052B228	7C80D47E	kernel32.GetLocaleInfoA
0052B22C	7C910331	ntdll.RtlGetLastWin32Error
0052B230	7C812C8D	kernel32.GetCommandLineA
0052B234	00000000	
0052B238	7C813559	kernel32.FindFirstFileA
0052B23C	7C80EFD7	kernel32.FindCloseA
0052B240	00539C0B	FR.00539C0B
0052B244	005E816A	FR.005E816A
0052B248	005CC056	FR.005CC056
0052B24C	005D9F14	FR.005D9F14
0052B250	00585B60	FR.00585B60

Crash tại hàm này

_Ta Conduct the IAT to copy this, repeat the above steps before running the script as thican nhusau to do is crash **Fill it with 00's in 52B540**

Backup
Copy
Binary
Modify
Label
Breakpoint
Search for
Follow in Dump
Find references
View executable file
Copy to executable file
Go to

Fill with 00's
Fill with FF's
Binary copy

0052B21C	7C80AC28	kernel32.GetProcAddress
0052B220	7C80B529	kernel32.GetModuleHandleA
0052B224	7C80B357	kernel32.GetModuleFileNameA
0052B228	7C80D47E	kernel32.GetLocaleInfoA
0052B22C	7C910331	ntdll.RtlGetLastWin32Error
0052B230	7C812C8D	kernel32.GetCommandLineA
0052B234	00000000	
0052B238	7C813559	kernel32.FindFirstFileA
0052B23C	7C80EFD7	kernel32.FindCloseA
0052B240	00000000	
0052B244	005E816A	FR.005E816A
0052B248	005CC056	FR.005CC056

_Ta Should ignore this function to Fix little later, the script runs at 1 long crash was similar in nhutren 2 address this

0052B2D4
0052B820

_cach do the same **with** nhutren **Fill 00's** with 2 function and wait script we finished running 4 conducted fix addresses that we **Fill with 00's**

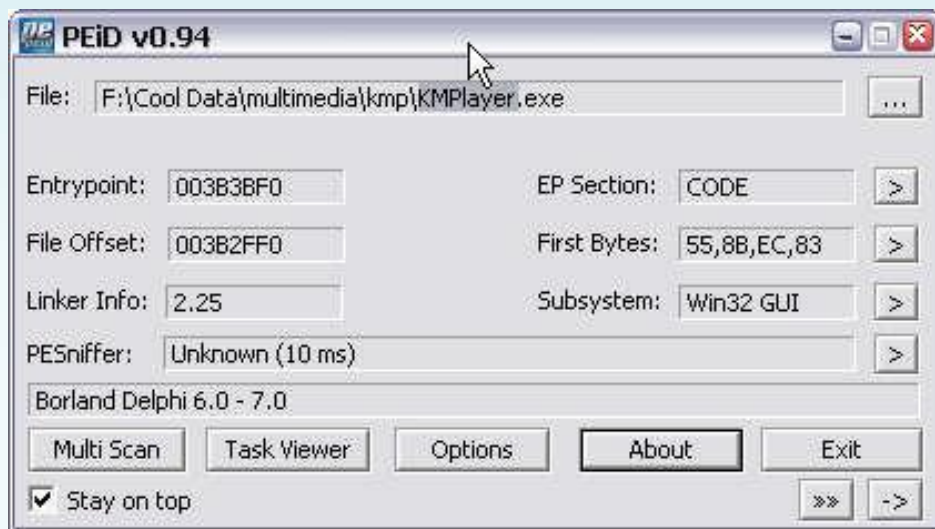
0052B234 0 0 00 0 000
0052B240 0 0 00 0 000
0052B2D4 00 0 0 0 000
0052B820 0 0 00 0 000

_Here is how Fix IAT has nhuTrace by hand, but they use Plugin 1 quite well how effective is

1 soft loans to code in **Borland Delphi 6.0 - 7.0** review of the functions to Fix the scripts

that run when pó hand I choose this because many soft Code Try using **Delphi 6.0 - 7.0**

are usually table IAT sem sem together. Here they choose **KMPlayer**



_ Load **KMPlayer** to **OillyDBG** and easily identified with its IAT and conducting comparative

Address	Value	Comment
007E3290	7C810311	kernel32.dll
007E3294	7C801D4F	kernel32.LoadLibraryExA
007E3298	7C80A405	kernel32.GetThreadLocale
007E329C	7C801EEE	kernel32.GetStartupInfoA
007E32A0	7C80AC28	kernel32.GetProcAddress
007E32A4	7C80B529	kernel32.GetModuleHandleA
007E32A8	7C80B357	kernel32.GetModuleFileNameA
007E32AC	7C80D47E	kernel32.GetLocaleInfoA
007E32B0	7C910331	ntdll.RtlGetLastWin32Error
007E32B4	7C8397A1	kernel32.GetCurrentDirectoryA
007E32B8	7C812C8D	kernel32.GetCommandLineA
007E32BC	7C80AA66	kernel32.FreeLibrary
007E32C0	7C813559	kernel32.FindFirstFileA
007E32C4	7C80FED7	kernel32.FindClose
007E32C8	7C826219	kernel32.CreateDirectoryA
007E32CC	7C81CAA2	kernel32.ExitProcess
007E32D0	7C80CCA9	kernel32.ExitThread
007E32D4	7C81082F	kernel32.CreateThread

Address	Value	Comment
0052B1F8	7C80977B	kernel32.VirtualQuery
0052B1FC	7C808859	kernel32.VirtualQuery
0052B200	7C80A0C7	kernel32.WideCharToMultiByte
0052B204	7C809CAD	kernel32.MultiByteToWideChar
0052B208	7C80C6E0	kernel32.lstrlenA
0052B20C	7C810311	kernel32.lstrcpyA
0052B210	7C801D4F	kernel32.LoadLibraryExA
0052B214	7C80A405	kernel32.GetThreadLocale
0052B218	7C801EEE	kernel32.GetStartupInfoA
0052B21C	7C80AC28	kernel32.GetProcAddress
0052B220	7C80B529	kernel32.GetModuleHandleA
0052B224	7C80B357	kernel32.GetModuleFileNameA
0052B228	7C80D47E	kernel32.GetLocaleInfoA
0052B22C	7C910331	ntdll.RtlGetLastWin32Error
0052B230	7C812C8D	kernel32.GetCommandLineA
0052B234	00000000	
0052B238	7C813559	kernel32.FindFirstFileA
0052B23C	7C80EED7	kernel32.FindClose
0052B240	00000000	
0052B244	7C80CC49	kernel32.ExitThread
0052B248	7C81082F	kernel32.CreateThread
0052B24C	7C810F9F	kernel32.WriteFile
0052B250	7C862B8A	kernel32.UnhandledExceptionFilter
0052B254	7C810D46	kernel32.SetFilePointer

_ Enough So they fix this 2

0052B234 0 0 00 0
000

0052B234 C 80 A 7 A k e n e r L3 2nd F r e e L a
66 r r i b i t a l y

0052B240 0 0 00 0
000

0052B240 7 C 8 2 6 k e n e r l32. C r e a t e D i
219 r e c y a T o r

_ Continue to address 0052B2D4

Address	Value	Comment
KMPlayer (Delphi 6.0 - 7.0)		
007E3318	7714C99D	oleaut32.SysFreeString
007E331C	77124B59	oleaut32.SysAllocStringLen
007E3320	00000000	
007E3324	7C809BF5	kernel32.TlsSetValue
007E3328	7C809750	kernel32.TlsGetValue
007E332C	7C8099BD	kernel32.LocalAlloc
007E3330	7C80B529	kernel32.GetModuleHandleA
007E3334	00000000	
007E3338	77DDD7CC	advapi32.RegSetValueExW
007E333C	77DDEBE7	advapi32.RegSetValueExA
007E3340	77E35FC2	advapi32.RegSetValueW
007E3344	77DE6F49	advapi32.RegSetValueA
007E3348	77DD6FC8	advapi32.RegQueryValueExW
007E334C	77DD7883	advapi32.RegQueryValueExA
007E3350	77DD08E2	advapi32.RegQueryValueW
007E3354	77DFCC10	advapi32.RegQueryValueA
007E3358	77DFCCEF	advapi32.RegQueryInfoKeyW
007E335C	77DFC1B5	advapi32.RegQueryInfoKeyA

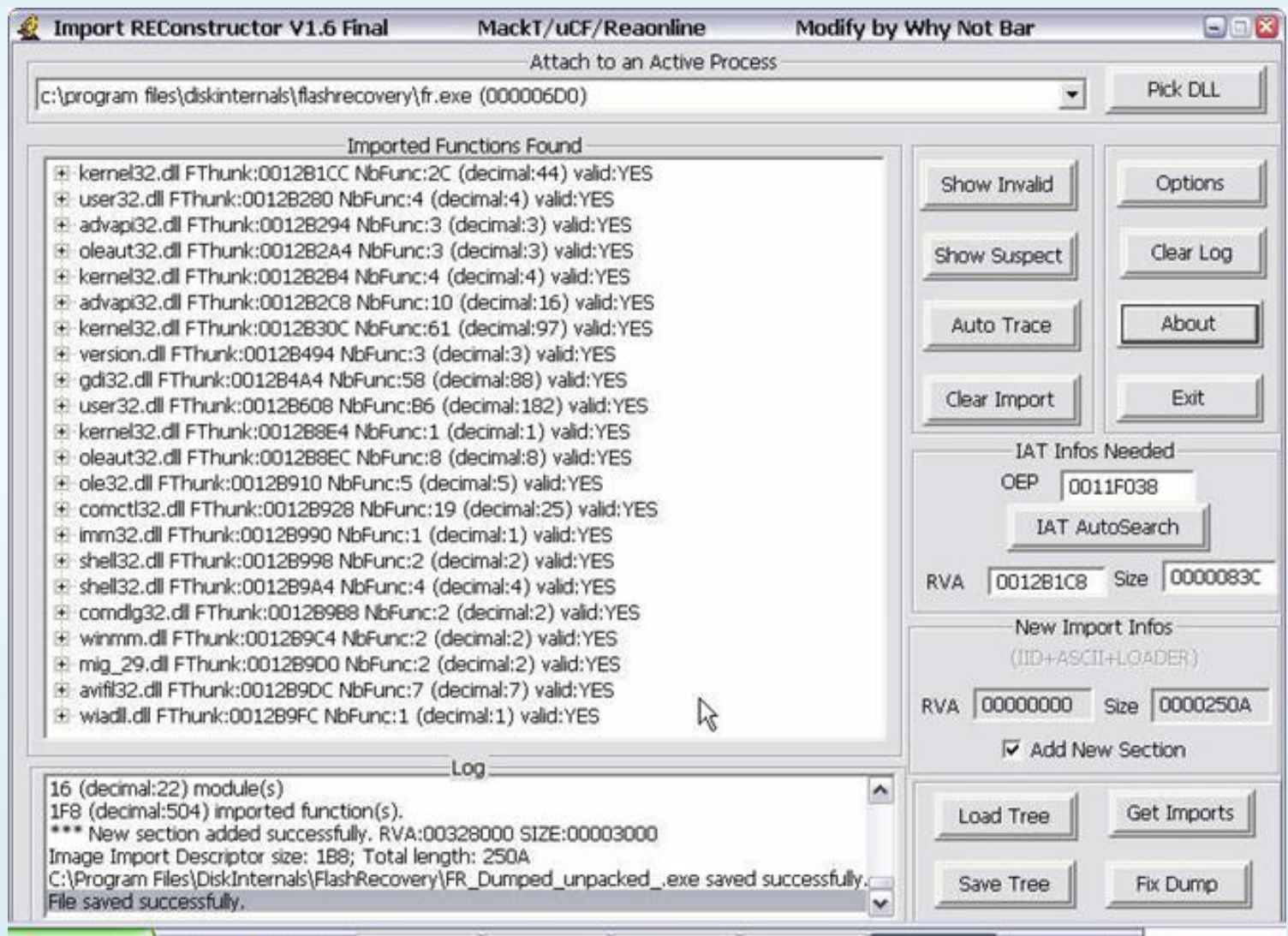
Address	Value	Comment
FlashRecovery (ExeCryptor)		
0052B29C	77DD6BF0	ADVAPI
0052B2A0	00000000	
0052B2A4	77124850	oleaut32.SysFreeString
0052B2A8	7714C99D	oleaut32.SysReAllocStringLen
0052B2AC	77124B59	oleaut32.SysAllocStringLen
0052B2B0	00000000	
0052B2B4	7C809BF5	kernel32.TlsSetValue
0052B2B8	7C809750	kernel32.TlsGetValue
0052B2BC	7C8099BD	kernel32.LocalAlloc
0052B2C0	7C80B529	kernel32.GetModuleHandleA
0052B2C4	00000000	
0052B2C8	77DD07CC	ADVAPI32.RegSetValueExW
0052B2CC	77DDEBE7	ADVAPI32.RegSetValueExA
0052B2D0	77DD6FC8	ADVAPI32.RegQueryValueExW
0052B2D4	00000000	
0052B2D8	77DD6A78	ADVAPI32.RegOpenKeyExW
0052B2DC	77DD761B	ADVAPI32.RegOpenKeyExA
0052B2E0	77DEB908	ADVAPI32.RegFlushKey
0052B2E4	77DD7535	ADVAPI32.RegCreateKeyExW
0052B2E8	77DD6AF4	ADVAPI32.RegCreateKeyExA
0052B2EC	77DD6BF0	ADVAPI32.RegCloseKey

_ With 0052B820

Address	Value	Comment
KMPlayer (Delphi 6.0 - 7.0)		
007E3A70	77D4B556	user32.GetClientRect
007E3A74	77D4D470	user32.GetClassNameW
007E3A78	77D4E032	user32.GetClassNameA
007E3A7C	77D520D6	user32.GetClassInfoW
007E3A80	77D64D4A	user32.GetClassInfoA
007E3A84	77D494FF	user32.GetCapture
007E3A88	77D4DF1E	user32.GetActiveWindow
007E3A8C	77D4E5FE	user32.FrameRect
007E3A90	77D6F7D0	user32.FindWindowExA
007E3A94	77D6F3C6	user32.FindWindowA
007E3A98	77D4D3C5	user32.FillRect
007E3A9C	77D89E6D	user32.ExitWindowsEx
007E3AA0	77D4BDD1	user32.EqualRect
007E3AA4	77D4D935	user32.EnumWindows
007E3AA8	77D4FACD	user32.EnumThreadWindows
007E3AAC	77D65B91	user32.EnumDisplaySettingsA
007E3AB0	77D6DA71	user32.EnumClipboardFormats
007E3AB4	77D4E5BA	user32.EnumChildWindows

Address	Value	Comment
FlashRecovery (ExeCryptor)		
0052B7F0	77D4D7BB	user32
0052B7F4	77D4F21D	user32.GetDCEx
0052B7F8	77D48697	user32.GetDC
0052B7FC	77D4C566	user32.GetCursorPos
0052B800	77D4CECD	user32.GetCursor
0052B804	77D6FCB2	user32.GetClipboardData
0052B808	77D48556	user32.GetClientRect
0052B80C	77D4E032	user32.GetClassNameA
0052B810	77D64D4A	user32.GetClassInfoA
0052B814	77D494FF	user32.GetCapture
0052B818	77D4DF1E	user32.GetActiveWindow
0052B81C	77D4E5FE	user32.FrameRect
0052B820	00000000	
0052B824	77D4D3C5	user32.FillRect
0052B828	77D4BDD1	user32.EqualRect
0052B82C	77D4D935	user32.EnumWindows
0052B830	77D4FACD	user32.EnumThreadWindows
0052B834	77D4E5BA	user32.EnumChildWindows
0052B838	77D4B4C5	user32.EndPaint
0052B83C	77D4C4D4	user32.EnableWindow

_ Fix After we have completed the **full IAT**, **dump Full** and open only to **ImportREC**.
 Select **List** in **Process FR.exe**. Enter **OEP = 0051F038 - 00,400,000 (Imagebase) = 0011F038**, **IAT AutoSearch** Click -> **Get Imports** -> **Show Invalid**



_Haha I have any function **Invalid** ... OK, click File Dumped.exe FR_Dumped.exe select Run and try File **FR_Dumped_.exe**



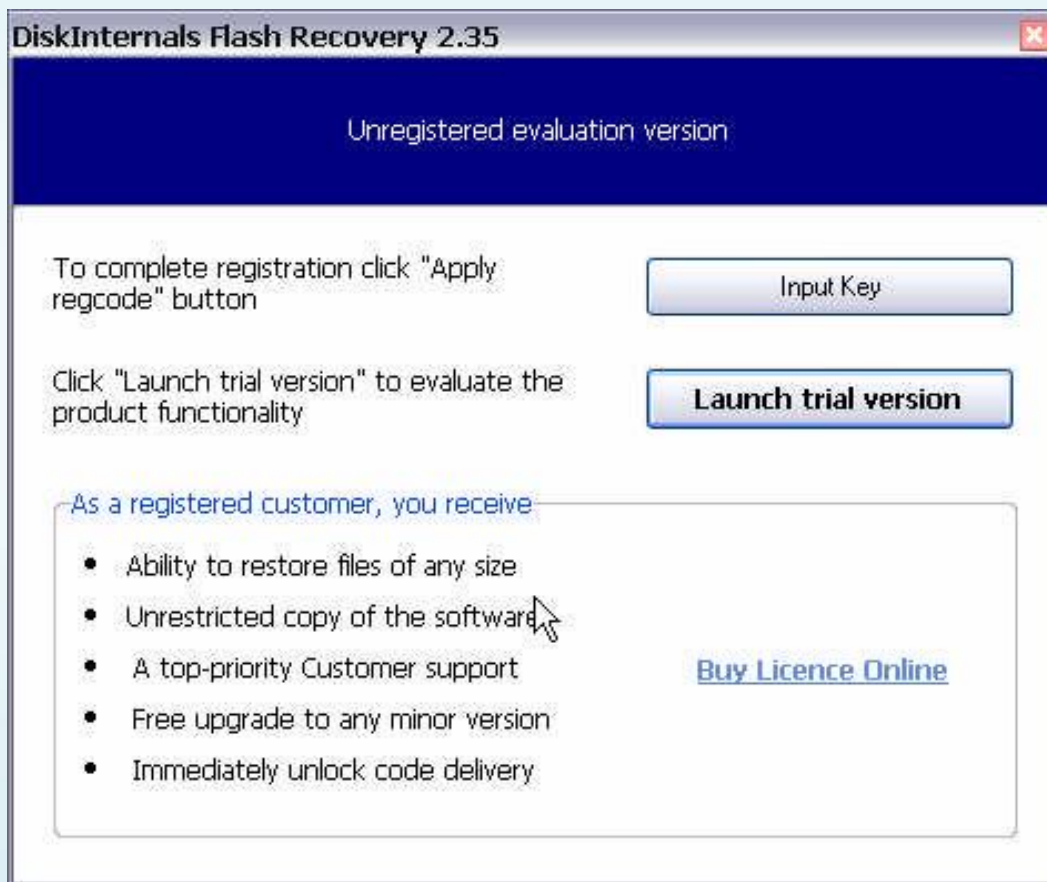
_Click **Debug** or Load **FR_Dumped_.exe** to OillyDBG we stop here

0051F038	CC	INT3	
0051F039	8BEC	MOV EBP,ESP	
0051F03B	83C4 F0	ADD ESP,-10	
0051F03E	B8 38EB5100	MOV EAX,FR_Dumpe.0051EB38	
0051F043	E8 8479EEFF	CALL FR_Dumpe.004069CC	
0051F048	E8 833CF6FF	CALL FR_Dumpe.00482CD0	
0051F04D	84C0	TEST AL,AL	
0051F04F	75 30	JNZ SHORT FR_Dumpe.0051F081	
0051F051	A1 08345200	MOV EAX,DWORD PTR DS:[523408]	
0051F056	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0051F058	E8 4323F6FF	CALL FR_Dumpe.004813A0	
0051F05D	8B0D B4325200	MOV ECX,DWORD PTR DS:[5232B4]	FR_Dumpe.0052
0051F063	A1 08345200	MOV EAX,DWORD PTR DS:[523408]	
0051F068	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0051F06A	8B15 047E5100	MOV EDX,DWORD PTR DS:[517E04]	FR_Dumpe.005-
0051F070	E8 4323F6FF	CALL FR_Dumpe.004813B8	
0051F075	A1 08345200	MOV EAX,DWORD PTR DS:[523408]	
0051F07A	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0051F07C	E8 B723F6FF	CALL FR_Dumpe.00481438	
0051F081	E8 1E5E5E5E	CALL FR_Dumpe.00481500	

_OEP we are teaching our SEO ... Press Ctrl + E and 55 to fix CC:

0051F038	55	PUSH EBP	
0051F039	8BEC	MOV EBP,ESP	
0051F03B	83C4 F0	ADD ESP,-10	
0051F03E	B8 38EB5100	MOV EAX,FR_Dumpe.0051EB38	
0051F043	E8 8479EEFF	CALL FR_Dumpe.004069CC	
0051F048	E8 833CF6FF	CALL FR_Dumpe.00482CD0	
0051F04D	84C0	TEST AL,AL	
0051F04F	75 30	JNZ SHORT FR_Dumpe.0051F081	
0051F051	A1 08345200	MOV EAX,DWORD PTR DS:[523408]	
0051F056	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0051F058	E8 4323F6FF	CALL FR_Dumpe.004813A0	
0051F05D	8B0D B4325200	MOV ECX,DWORD PTR DS:[5232B4]	FR_Dumpe.0052
0051F063	A1 08345200	MOV EAX,DWORD PTR DS:[523408]	
0051F068	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0051F06A	8B15 047E5100	MOV EDX,DWORD PTR DS:[517E04]	FR_Dumpe.005-
0051F070	E8 4323F6FF	CALL FR_Dumpe.004813B8	
0051F075	A1 08345200	MOV EAX,DWORD PTR DS:[523408]	
0051F07A	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0051F07C	E8 B723F6FF	CALL FR_Dumpe.00481438	
0051F081	E8 1E5E5E5E	CALL FR_Dumpe.00481500	

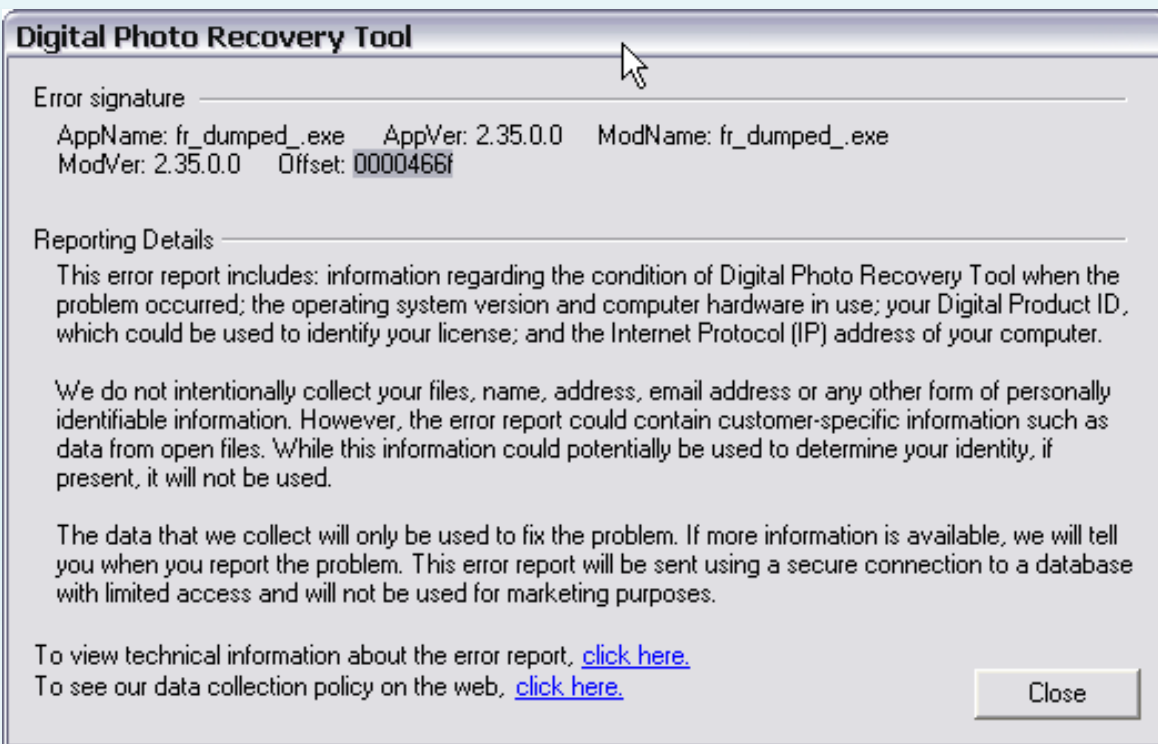
_ Save the Test Run ... considered. ... Haha running nhuNgua



_ But the press **Cancel**



_ Click click **Debug** to stop us at the point of causing **Crash** or click **Click Here** to address causes crash



_ Nhuday in **OlllyDBG** we will stop here

0040465B	8B06	mov eax, dword ptr ds:[esi]
0040465D	50	push eax
0040465E	E8 9DCCFFFF	call <jmp.&kernel32.CreateDirectoryA>
00404663	8B03	mov eax, dword ptr ds:[ebx]
00404665	56	push esi
00404666	8BF0	mov esi, eax
00404668	8BFB	mov edi, ebx
0040466A	B9 0B000000	mov ecx, 0B
0040466F	F3:A5	rep movs dword ptr es:[edi], dword ptr ds:[esi]
00404671	5E	pop esi
00404672	^ E9 76FFFFFF	jmp FR_Dumpe.004045ED
00404677	5D	pop ebp
00404678	5F	pop edi
00404679	5E	pop esi

_ **Submit** address this and **save** again.

0040465D	8B00	mov eax, dword ptr ds:[esi]
0040465D	50	push eax
0040465E	E8 9DCCFFFF	call <jmp.&kernel32.CreateDirectoryA>
00404663	8B03	mov eax, dword ptr ds:[ebx]
00404665	56	push esi
00404666	8BF0	mov esi, eax
00404668	8BFB	mov edi, ebx
0040466A	B9 0B000000	mov ecx, 0B
0040466F	90	nop
00404670	90	nop
00404671	5E	pop esi
00404672	^ E9 76FFFFFF	jmp FR_Dumpe.004045ED
00404677	5D	pop ebx

_ When you click **Cancel** ko will also be more Crash **unpack ... Done!**

G r I Ee TsF italy Ou the Compu t e r A _ of e l, e mbi Z o, M A B oo nb italy, H o

acnh, Nina B e, e ki nman o w ar, Z o i D e ux, M e r c, l o e ight to nix, T r o b

icky italy, Takad a iamidi ot, of the e n t e n handi ... and italy o u!

Nha T r an g, 2 6 On the G9 a n 20 0 6

**Whot italyN
Bar**

I. I n t r o d u c t i o n:

Today, we unpack SLVc0deProtector 1.1. This form pack he tland the English thie u. kienmanowar iamidiot and have had to unpack the da crackme.scp.exe in detail, but Target has Stolen code clearly ko ko even all of Stolen Code Fix also still running as horses. I find the soft pack as this but i find it dành used as the main SLVc0deProtector 1.1 Target for this tut. Tut considered this as a lie from the store's own children and those who like to unpack.

I. T o o l s:

• *T oo L and P is the evil in the g a n d n g s:*

• *O D D B italy YK 1st 1 0*

• *R L! W e a l e s 0. 5p l u g i n* (RL! Weasle is OllyDBG plugin that can help you reslove invalid

ImpRec imports. It feautres unique specialized tracers and several different levels for generic tracer

protections that copy data from dlls and store space allocated to them. This is done by skipping

obfuscation and hashing correct instructions that are then compared to original. dll ones)

• *Lord P E 1. 4*

• *I m p o r t R E C 1. 6*

• *P I D E 0. 9 4*

Tar of GE: S L V c e d 0 P r o t e c t o r 1. 1 (k è o m t h e t u t)

III. I m O T E to PvaFixS l e C o n d e:

_ Dung **Eid P 0. 9 4** é a result of a g e r t:



Lo _ a dt a rg e t o **O** and **Y K DbyD**



_ Click **OK** to continue, and stop here

Address	Hex dump	Disassembly	Comment
00401000	E8 00000000	CALL 00401005	00401005
00401005	58	POP EAX	
00401006	C600 EB	MOV BYTE PTR DS:[EAX],0EB	
00401009	C640 01 08	MOV BYTE PTR DS:[EAX+1],8	
0040100D	FFE0	JMP NEAR EAX	
0040100F	- E9 4CD00000	JMP 0040E060	0040E060
00401014	2C E6	SUB AL,0E6	
00401016	F1	INT1	
00401017	60	PUSHAD	
00401018	D9A6 B1206068	FLDENV (28-BYTE) PTR DS:[ESI+68]	
0040101E	^ 71 E1	JNO SHORT 00401001	00401001
00401020	8338	ADD EB8,DWORD PTR DS:[EAX]	

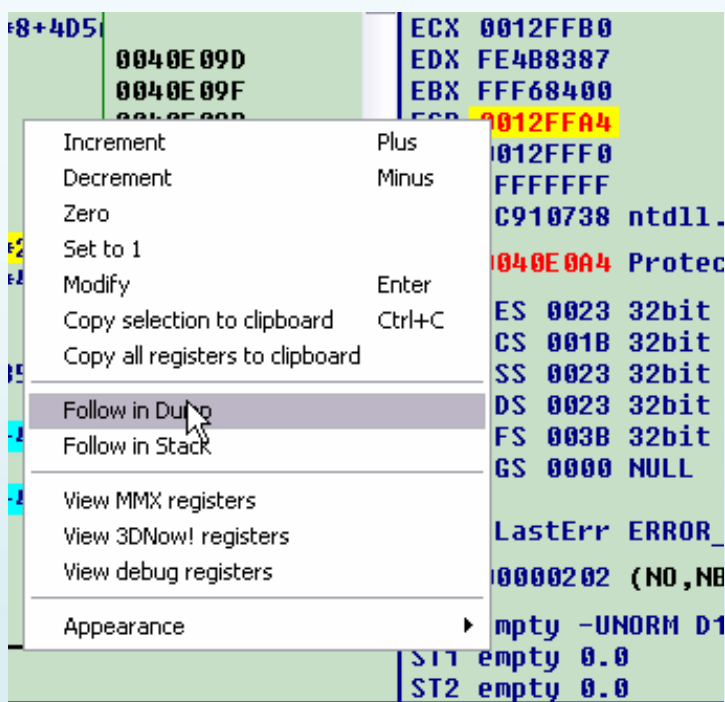
_ Order "Jump 0040E060" we are here to:

Address	Hex dump	Disassembly	Comment
0040E060	05 1D939270	ADD EAX,7092931D	
0040E065	F6D3	NOT BL	
0040E067	8D0445 1564453D	LEA EAX,DWORD PTR DS:[EAX*2+3D45]	
0040E06E	~ EB 02	JMP SHORT 0040E072	0040E072
0040E070	~ EB 02	JMP SHORT 0040E074	0040E074
0040E072	^ EB FC	JMP SHORT 0040E070	0040E070
0040E074	81EA 0D68457E	SUB EDX,7E45680D	
0040E07A	2BCE	SUB ECX,ESI	
0040E07C	43	INC EBX	
0040E07D	69C0 B8AFE73D	IMUL EAX,EAX,3DE7AFB8	
0040E083	C1E3 E2	SHL EBX,0E2	
0040E086	8D0485 F4D27756	LEA EAX,DWORD PTR DS:[EAX*4+5675]	
0040E08D	05 E8F2B856	ADD EAX,56B8F2E8	
0040E092	8D04C5 5CE75A4D	LEA EAX,DWORD PTR DS:[EAX*8+4D5A]	
0040E099	~ EB 02	JMP SHORT 0040E09D	0040E09D
0040E09B	~ EB 02	JMP SHORT 0040E09F	0040E09F
0040E09D	^ EB FC	JMP SHORT 0040E09B	0040E09B
0040E09F	49	DEC ECX	
0040E0A0	F71424	NOT DWORD PTR SS:[ESP]	
0040E0A3	60	PUSHAD	
0040E0A4	8D0445 2C20D23E	LEA EAX,DWORD PTR DS:[EAX*2+3ED3]	

_ _Nhin Down slightly under 1 shows **PUSHAD** order. Set **Breakpoint** 1 at the press **F9**, and we stop at just Set BreakPoint

0040E070	EB 02	JMP SHORT 0040E074	0040E074
0040E072	EB FC	JMP SHORT 0040E070	0040E070
0040E074	81EA 0D68457E	SUB EDX,7E45680D	
0040E07A	2BCE	SUB ECX,ESI	
0040E07C	43	INC EBX	
0040E07D	69C0 B8AFE73D	IMUL EAX,EAX,3DE7AFB8	
0040E083	C1E3 E2	SHL EBX,0E2	
0040E086	8D0485 F4D27756	LEA EAX,DWORD PTR DS:[EAX*4+567	
0040E08D	05 E8F2B856	ADD EAX,56B8F2E8	
0040E092	8D04C5 5CE75A4D	LEA EAX,DWORD PTR DS:[EAX*8+4D5	
0040E099	EB 02	JMP SHORT 0040E09D	0040E09D
0040E09B	EB 02	JMP SHORT 0040E09F	0040E09F
0040E09D	EB FC	JMP SHORT 0040E09B	0040E09B
0040E09F	49	DEC ECX	
0040E0A0	F71424	NOT DWORD PTR SS:[ESP]	
0040E0A3	60	PUSHAD	
0040E0A4	8D0445 2C20D23E	LEA EAX,DWORD PTR DS:[EAX*2+3ED	
0040E0AB	8D0485 F8278B08	LEA EAX,DWORD PTR DS:[EAX*4+88B	
0040E0B2	42	INC EDX	
0040E0B3	E8 01000000	CALL 0040E0B9	0040E0B9
0040E0B8	A0 5DEB0169	MOV AL,BYTE PTR DS:[6901EB5D]	

_ Press **F8** and Choose **ESP** -> **Follow in dump**



_ In the window **dump**

Address	Value	Comment
0012FFA4	7C910738	ntdll.7C910738
0012FFA8	FFFFFFFF	
0012FFAC	0012FFF0	
0012FFB0	0012FFC4	
0012FFB4	FFF68400	
0012FFB8	FE4B8387	
0012FFBC	0012FFB0	
0012FFC0	8469F53C	
0012FFC4	837E92B0	
0012FFC8	7C910738	ntdll.7C910738
0012FFCC	FFFFFFFF	

_ Next we need to set 1 bp on hardware access in dword

40E08D	05 E8F2B856		
40E092	8D04C5 5CE75A4D		
40E099	EB 02		
40E09B	EB 02		
40E09D	EB FC		
40E09F	49		
40E0A0	F71424		
40E0A3	60		
40E0A4	8D0445 2C20D23E		
40E0AB	8D0485 F8278B08		
40E0B2	42		
40E0B3	E8 01000000		
40E0B8	A0 5DEB0169		
40E0BD	81ED 5F1A4000		
40E0C3	8D85 921A4000		
dress=47A60AA4			
X=8469F53C			
dress	Value	Comment	
12FFA4	7C910738	ntdll.7C910738	
12FFA8	FFFFFFFF		

Modify

Breakpoint

Search for

Follow in Disassembler

Follow in Dump

Go to

Hex

Text

Short

Long

Float

Disassemble

Special

Export table

Appearance

Memory, on access

Memory, on write

Hardware, on access

Hardware, on write

Hardware, on execution

Byte

Word

Dword

R DS:[EAX*2+3ED]

R DS:[EAX*4+88B]

DS:[6901EB5D]

R SS:[ERP+401A9]

0040E0B9

_ Press Shift + F9 we stop here

Address	Hex dump	Disassembly	Comment
0040E56B	83C4 08	ADD ESP,8	
0040E56E	F71424	NOT DWORD PTR SS:[ESP]	
0040E571	68 73474000	PUSH 404773	
0040E576	90	NOP	
0040E577	90	NOP	
0040E578	90	NOP	
0040E579	90	NOP	
0040E57A	90	NOP	
0040E57B	90	NOP	
0040E57C	90	NOP	
0040E57D	90	NOP	

_ Press F8 PUSH 2 to 404,773, Scroll down the code you see PUSH 40478E. haha 40478E address the OEP. Press Ctrl + G to enter 40478E to us here

Address	Hex dump	Disassembly	Comment
0040478E	90	NOP	Stolen OEP!
0040478F	90	NOP	
00404790	90	NOP	
00404791	90	NOP	
00404792	90	NOP	
00404793	E8 3A0C0000	CALL 004053D2	004053D2
00404798	68 C4784000	PUSH 4078C4	ASCII "FuCk..."
0040479D	6A 01	PUSH 1	
0040479F	6A 00	PUSH 0	
004047A1	E8 BA0B0000	CALL 00405360	00405360
004047A6	E8 DF0B0000	CALL 0040538A	0040538A
004047AB	35 B7000000	XOR EAX,0B7	
004047B0	75 02	JNZ SHORT 004047B4	004047B4
004047B2	EB 5E	JMP SHORT 00404812	00404812
004047B4	6A 00	PUSH 0	

_ Hichic! 4 command Submit and it is **Stolen Code**. We need to Fix it Fix ... but ... as seo first time to place this child is also deaf ... but a ray chot am when they look up to see the window **dump "0012FFC0 7B960AC3"**

Address	Value	Comment
0012FFA4	7C910738	ntdll.7C910738
0012FFA8	FFFFFFFF	
0012FFAC	0012FFF0	
0012FFB0	0012FFC4	
0012FFB4	FFF68400	
0012FFB8	FE4B8387	
0012FFBC	0012FFB0	
0012FFC0	7B960AC3	
0012FFC4	837E92B0	
0012FFC8	7C910738	ntdll.7C910738
0012FFCC	FFFFFFFF	

_ Ngo FPU up window address bar is **0012FFC0 ESP**

Registers (FPU)			
EAX	FFF68400		
ECX	0012FFC4		
EDX	0012FFF0		
EBX	FFFFFFFF		
ESP	0012FFC0		
EBP	0040E0EA	Protect.0040E0EA	
ESI	0040E17A	Protect.0040E17A	
EDI	0012FFE0		
EIP	0040E571	Protect.0040E571	
C 0	ES 0023	32bit 0(FFFFFFFF)	
P 1	CS 001B	32bit 0(FFFFFFFF)	
A 1	SS 0023	32bit 0(FFFFFFFF)	
Z 0	DS 0023	32bit 0(FFFFFFFF)	
S 0	FS 003B	32bit 7FFDF000(FFF)	
T 0	GS 0000	NULL	
D 0			

_ This is already clear, to **40478E** and revised as follows:

Address	Hex dump	Disassembly	Comment	Registers (FPU)
0040478E	90	NOP	Stolen OEP!	EAX FFF68400
0040478F	90	NOP		ECX 0012FFC4
00404790	90	NOP		EDX 0012FFF0
00404791	90	NOP		EBX FFFFFFFF
00404792	90	NOP		ESP 0012FFC0
00404793	E8 3A0C0000	CALL 004053D2	004053D2	EBP 0040E0EA Prote
00404794			ASCII "FuCk... I	ESI 0040E17A Prote
00404795				EDI 0012FFE0
00404796				EIP 0040E571 Prote
00404797			00405360	C 0 ES 0023 32bi
00404798			0040538A	P 1 CS 001B 32bi
00404799			004047B4	A 1 SS 0023 32bi
0040479A			00404812	Z 0 DS 0023 32bi
0040479B				S 0 FS 003B 32bi
0040479C				T 0 GS 0000 NULL

Assemble at 0040478E

Push 7B960AC3

☒ ?? NOP ??

Assemble ??

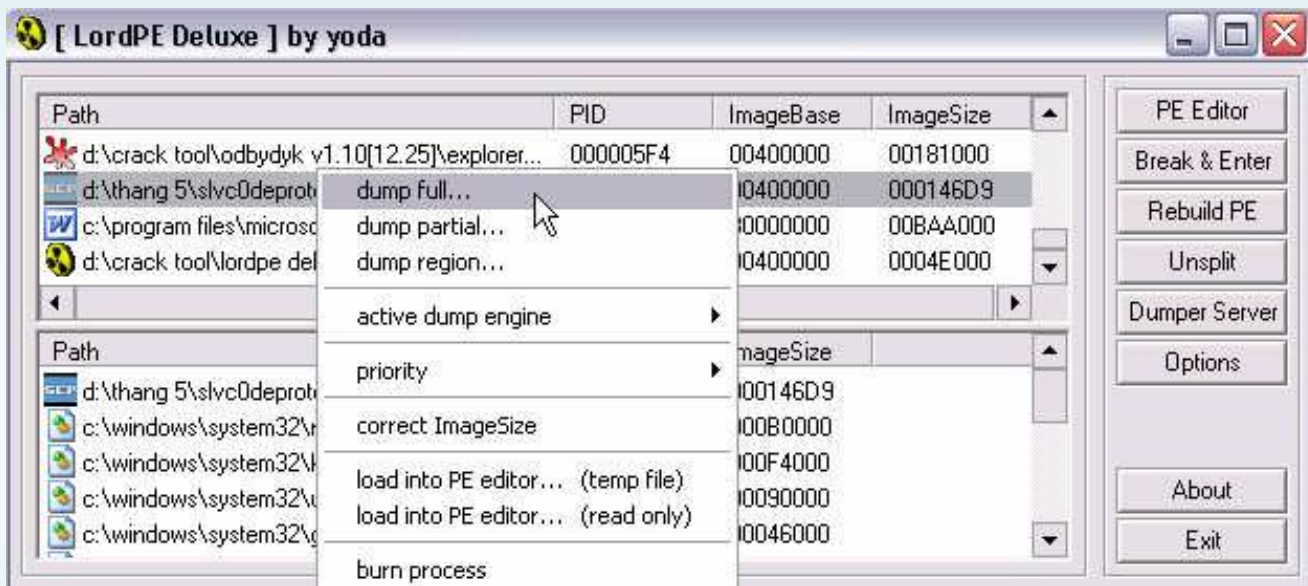
Address	Value	Comment
0012FFA4	7C910738	ntdll.7C910738
0012FFA8	FFFFFFFF	
0012FFAC	0012FFF0	
0012FFB0	0012FFC4	
0012FFB4	FFF68400	
0012FFB8	FE4B8387	
0012FFBC	0012FFB0	
0012FFC0	7B960AC3	
0012FFC4	837E92B0	
0012FFC8	7C910738	ntdll.7C910738
0012FFCC	FFFFFFFF	

OllyDbg v1.10

_S a u k the F i x

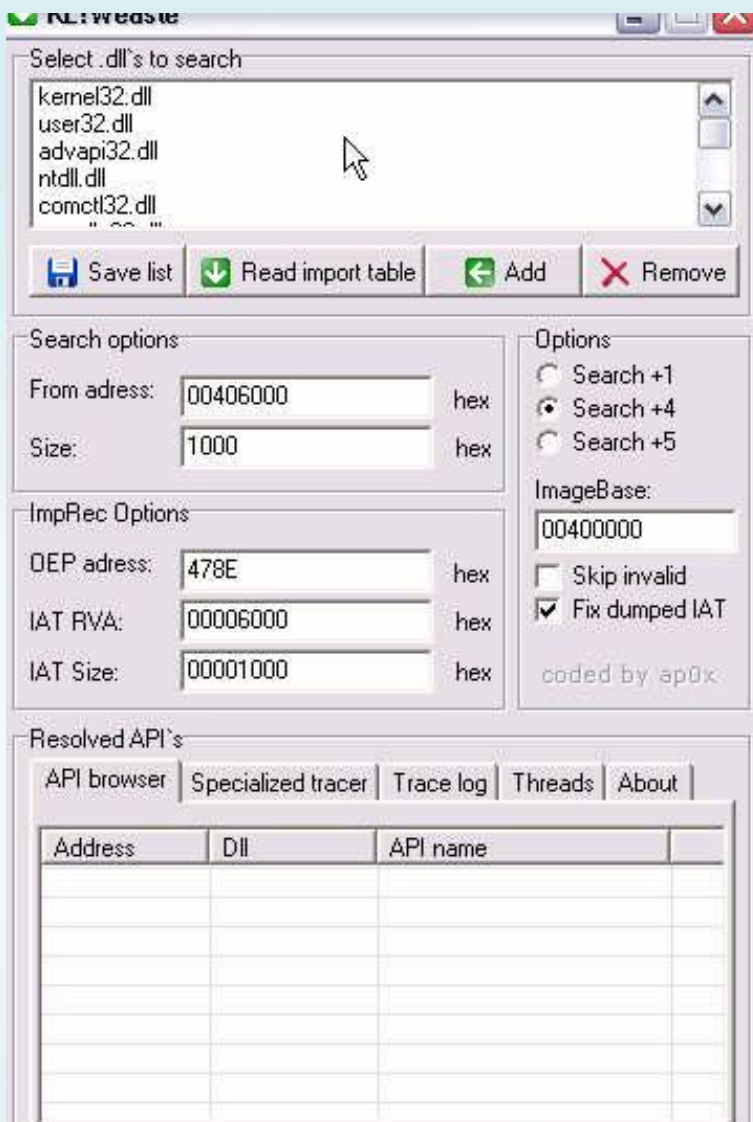
Address	Hex dump	Disassembly	Comment
00404788	E8 E50B0000	CALL 00405372	00405372
0040478D	C3	RET	
0040478E	68 0A967B	PUSH 7B960AC3	Stolen OEP!
00404793	E8 3A0C0000	CALL 004053D2	004053D2
00404798	68 C4784000	PUSH 4078C4	ASCII "FuCk... I hate EU
0040479D	6A 01	PUSH 1	
0040479F	6A 00	PUSH 0	
004047A1	E8 BA0B0000	CALL 00405360	00405360
004047A6	E8 DF0B0000	CALL 0040538A	0040538A
004047AB	35 B7000000	XOR EAX, 0B7	
004047B0	75 02	JNZ SHORT 004047B4	004047B4
004047B2	EB 5E	JMP SHORT 00404812	00404812
004047B4	6A 00	PUSH 0	
004047B6	E8 E10B0000	CALL 0040539C	0040539C
004047BB	A3 08704000	MOV DWORD PTR DS:[407008], EAX	

_Khic haha ... just waiting ... what more full dump with only LordPE ..



IV. Etna l e v API I:

_ Now to the very important resolve the API, you can manually or use RL! Weasle. I choose RL! Weasle for fast and accurate ... (I luòi lords). Plugin to choose RL! Weasle and fill the parameters as follows:

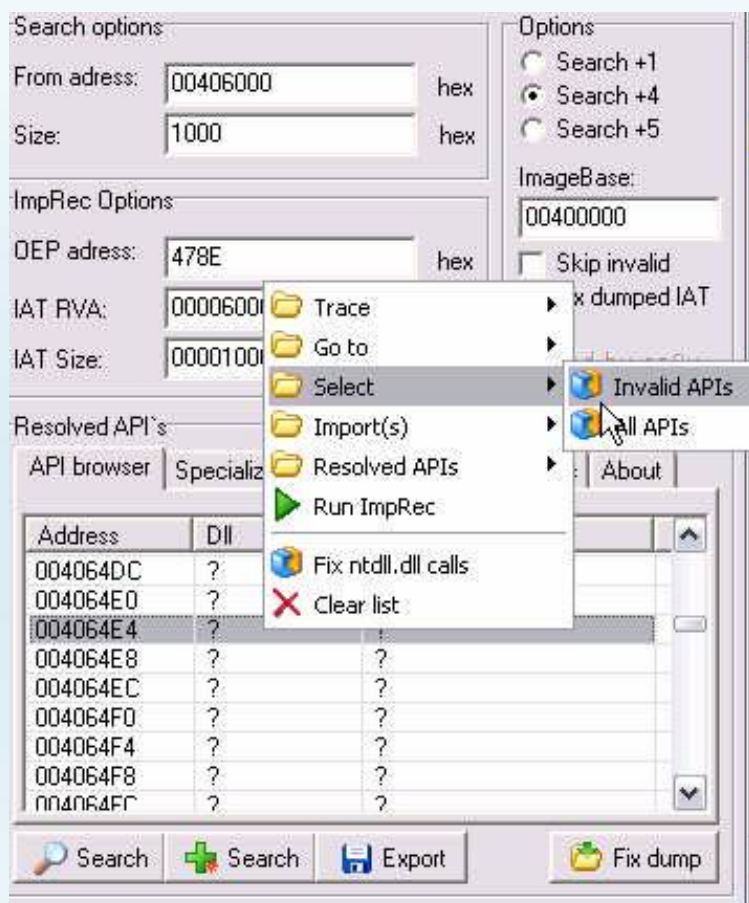




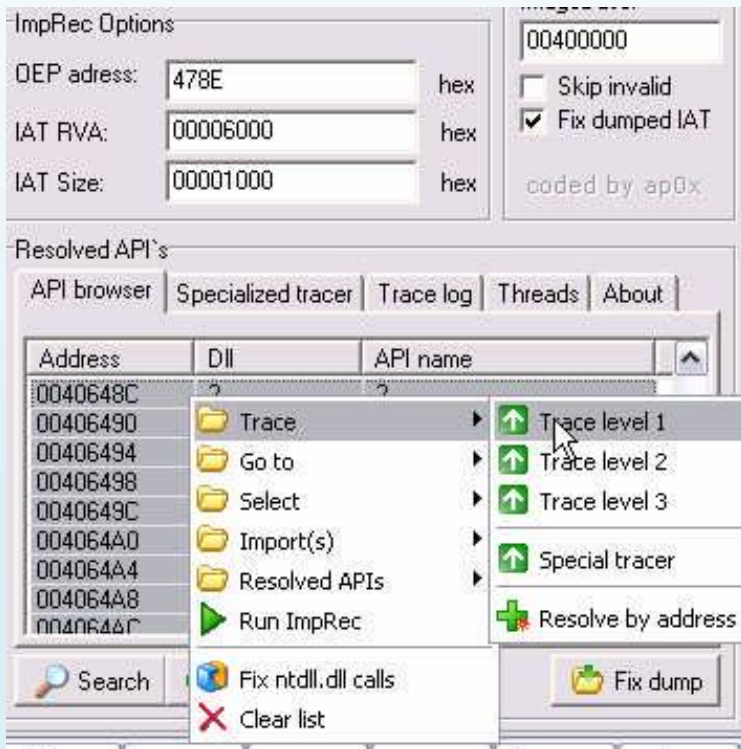
_ Make sure you ask why they filled **406,000**. Very simply because it is only the first Section of the table contains IAT. To determine where you pull up screen press Ctrl + B and enter the FF 25. But 47E8 is OEP

004052CF	68 FFFFFFFF	PUSH 0FFFFFFF	
004052D4	68 13040000	PUSH 413	
004052D9	FF35 0E7C4000	PUSH DWORD PTR DS:[407C0E]	
004052DF	E8 7E010000	CALL 00405462	00405462
004052E4	C9	LEAVE	
004052E5	C2 0C00	RET 0C	
004052E8	- FF25 C8604000	JMP NEAR DWORD PTR DS:[4060C8]	
004052EE	- FF25 C4604000	JMP NEAR DWORD PTR DS:[4060C4]	
004052F4	- FF25 C0604000	JMP NEAR DWORD PTR DS:[4060C0]	
004052FA	- FF25 10604000	JMP NEAR DWORD PTR DS:[406010]	
00405300	- FF25 14604000	JMP NEAR DWORD PTR DS:[406014]	
00405306	- FF25 18604000	JMP NEAR DWORD PTR DS:[406018]	
0040530C	- FF25 1C604000	JMP NEAR DWORD PTR DS:[40601C]	
00405312	- FF25 20604000	JMP NEAR DWORD PTR DS:[406020]	
00405318	- FF25 24604000	JMP NEAR DWORD PTR DS:[406024]	

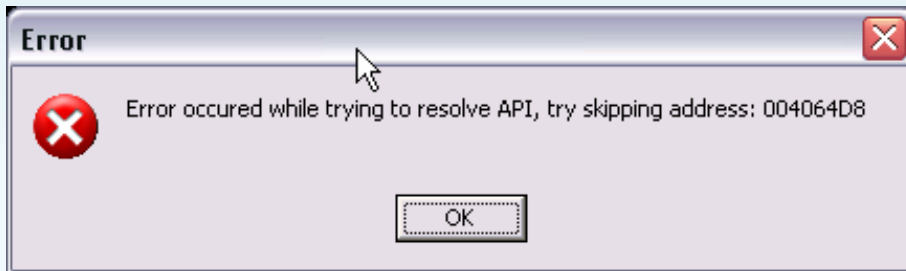
_ Done elsewhere click the **Search** button and select the image:



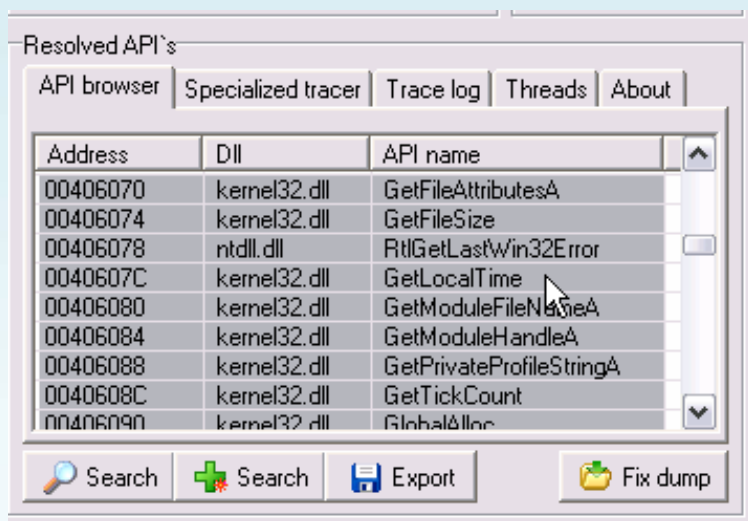
_va continue to choose Select-> **Invalid APIs-> Trace level 1**



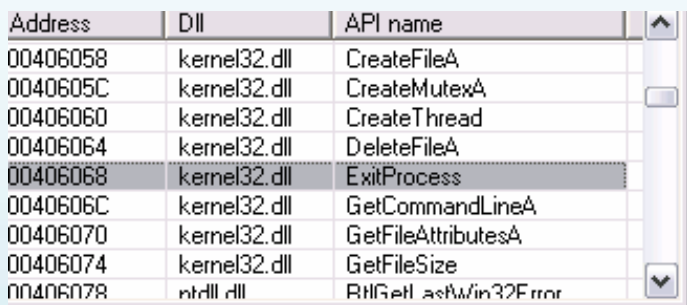
_ It appears



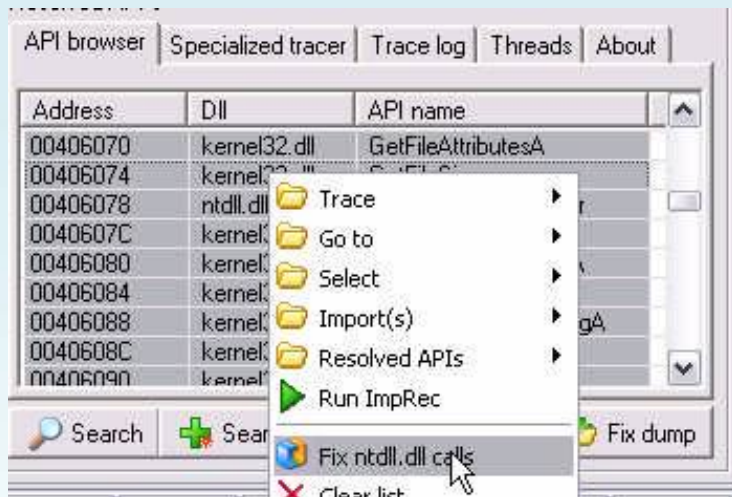
_ You just click OK to when all is the only way. And we have nhusau



_ **Select-> Invalid APIs-> Trace level 2.** haha we Trace added this function

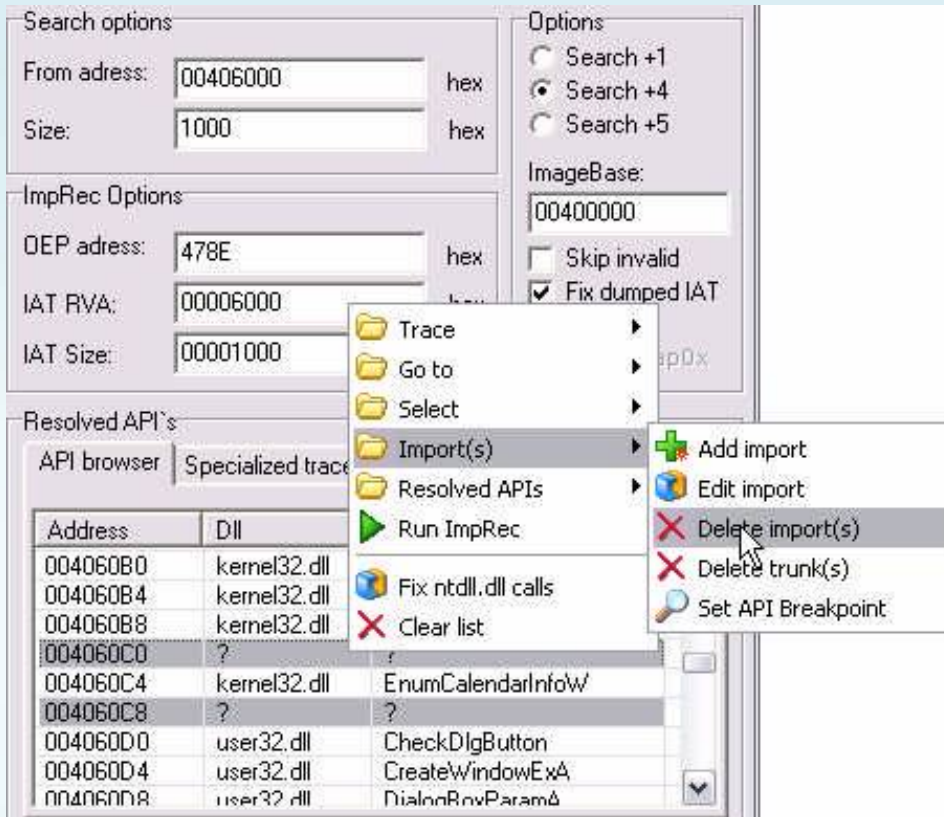


_ Next select **Fix NT d ll C a lls**





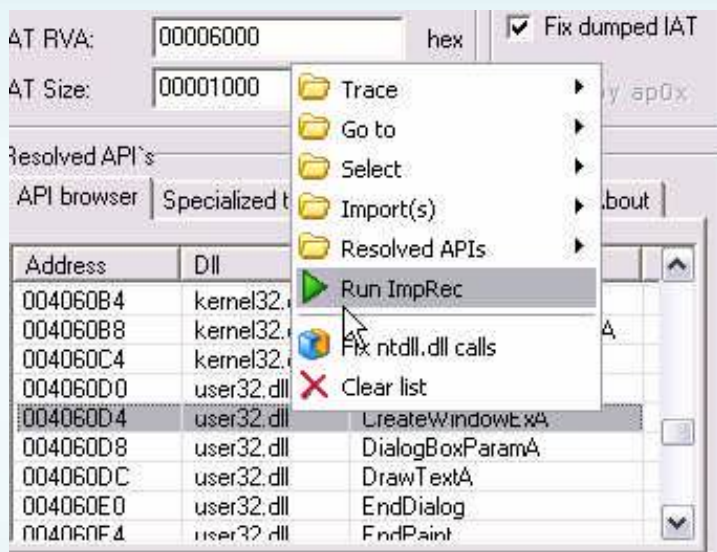
_ **S e l e c t -> I n d** and the **A** and **P** is to the end



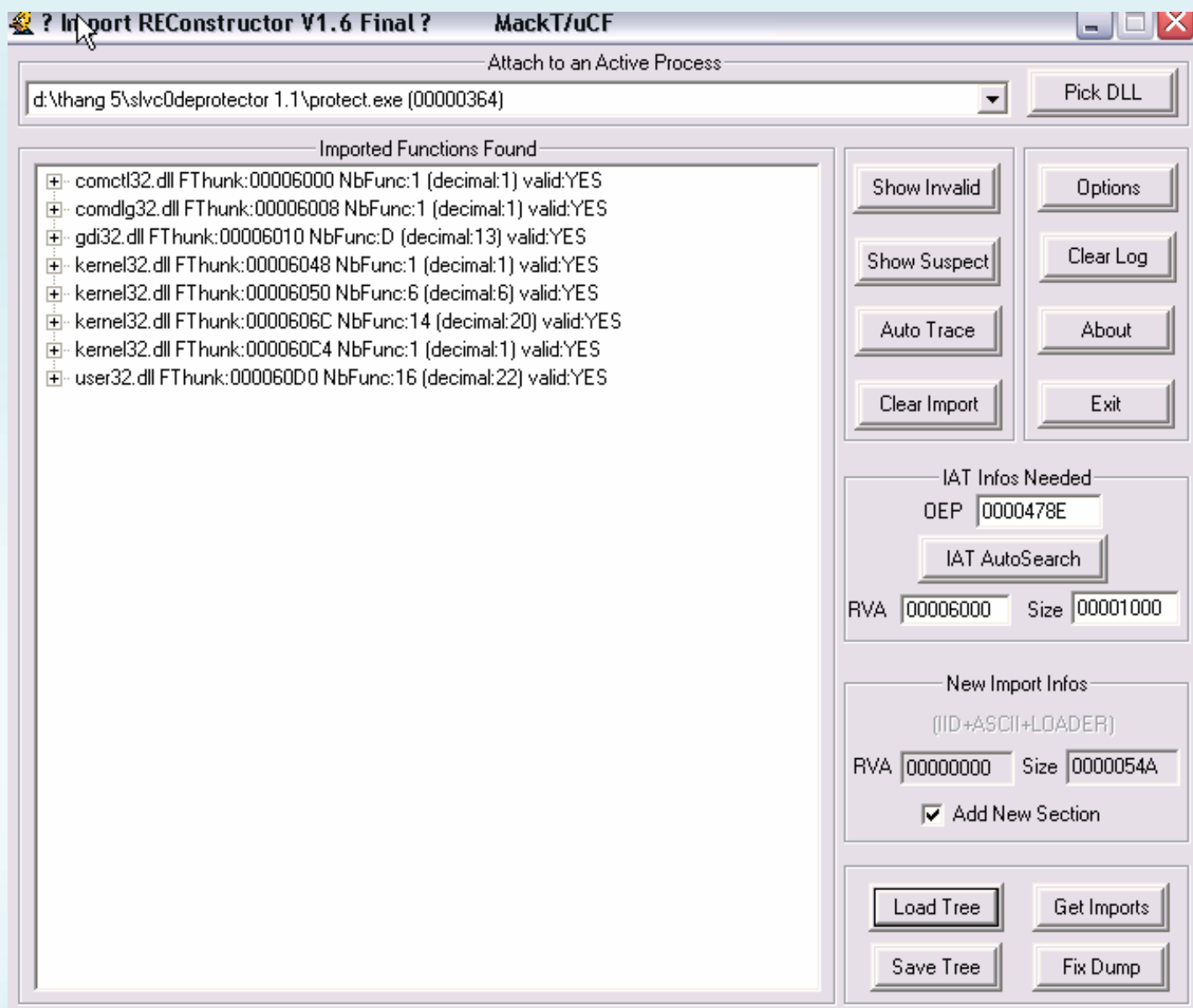
_ The **Export** button han



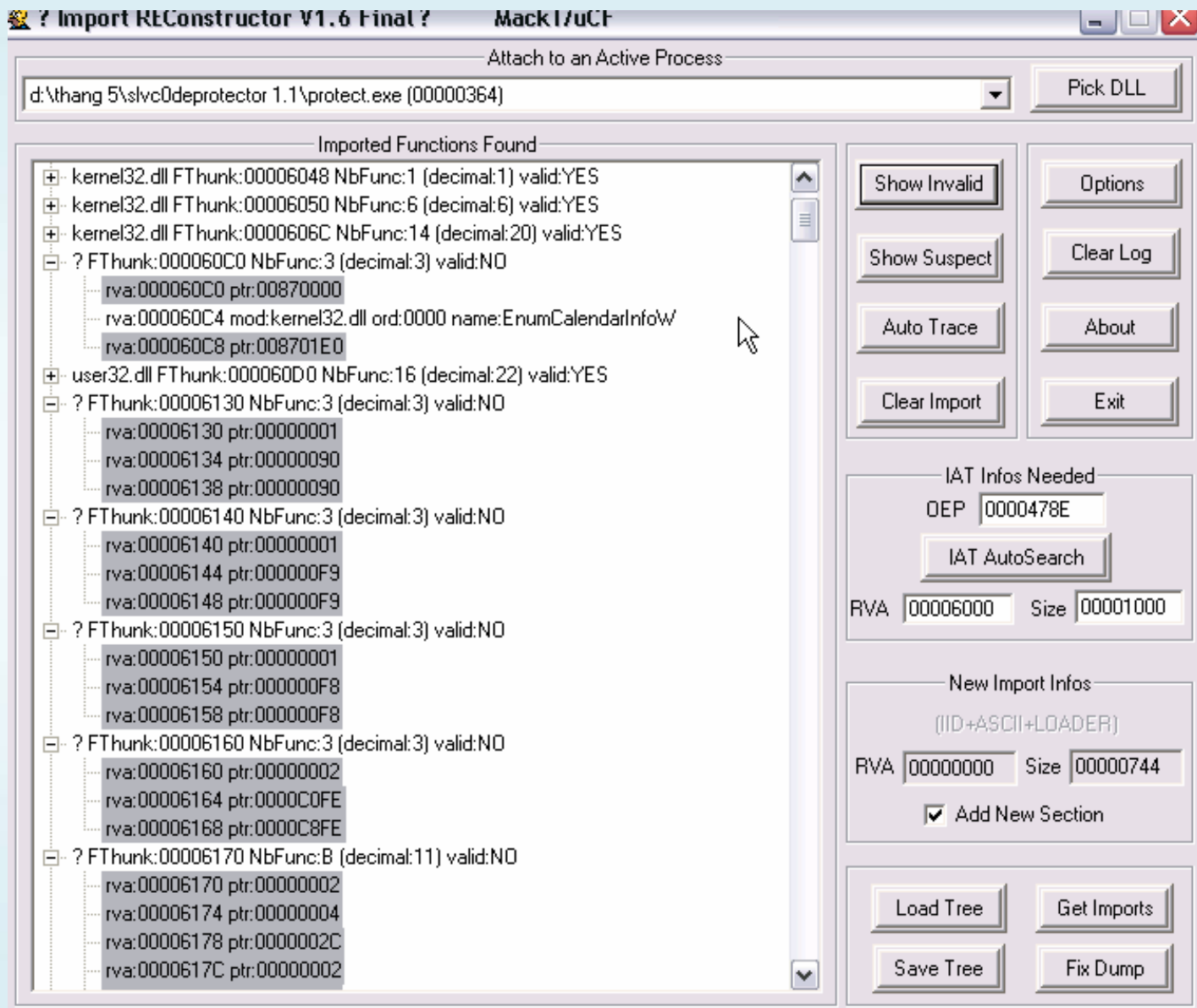
_ Now **I** m running **R p e c**



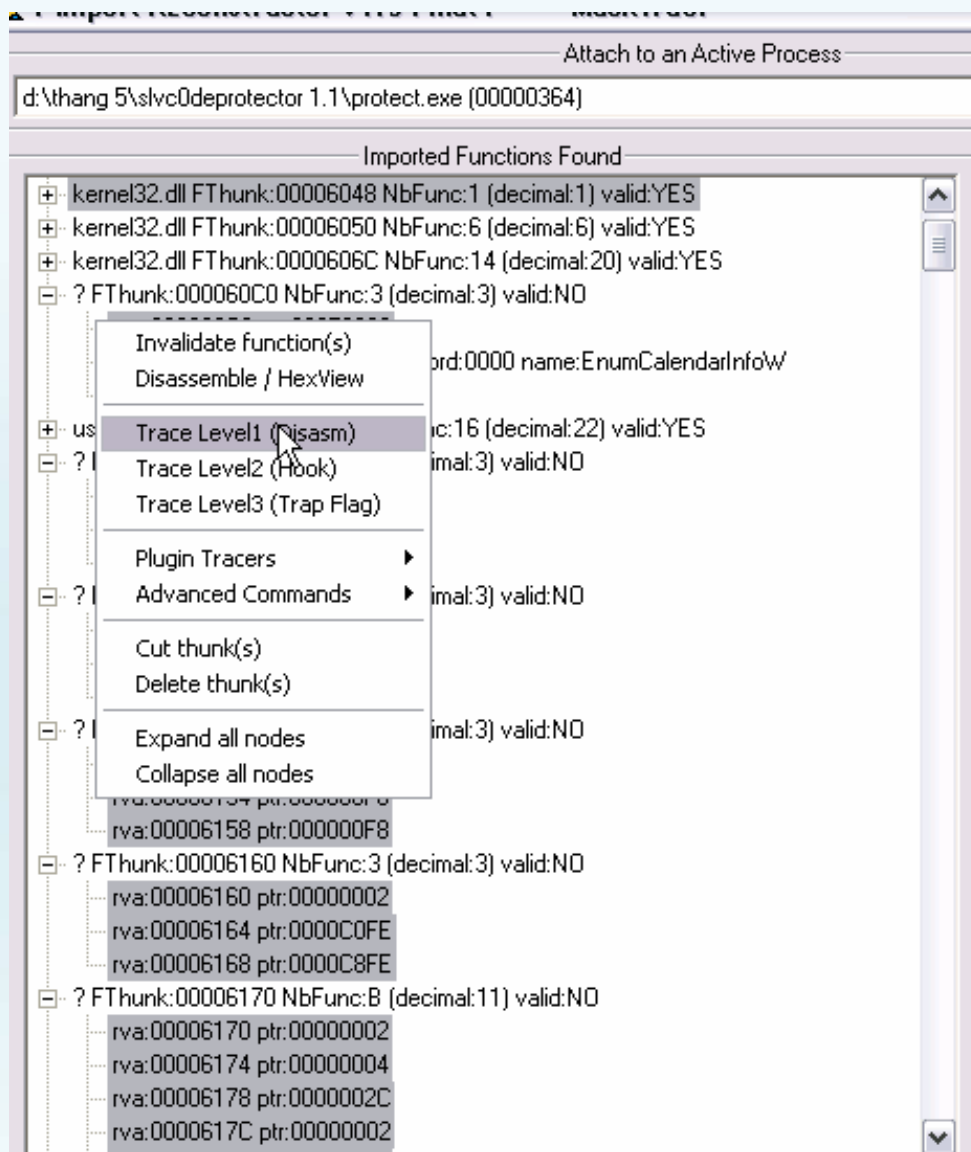
_ More than C P ID and Lo ad T e r e that we've very Ex p o



Fix _Dung hit with the **dump** we need Fix 1 little more ... if i will be eligible Crash tè ... **Get Imports** Click -> **Show Invalid**



_ And click to select the image

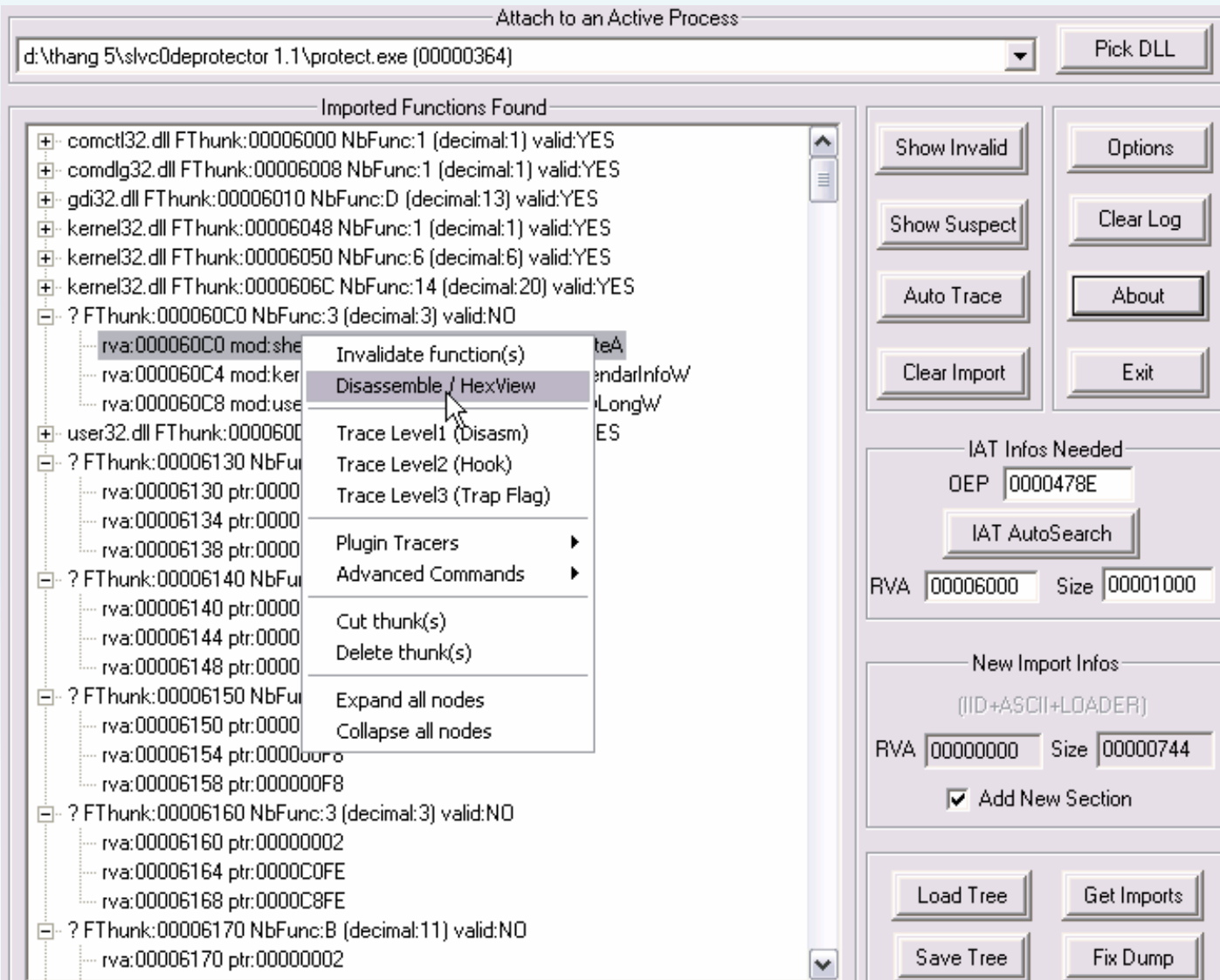


_ C than **S h o w I n v a l i d** and you see the following

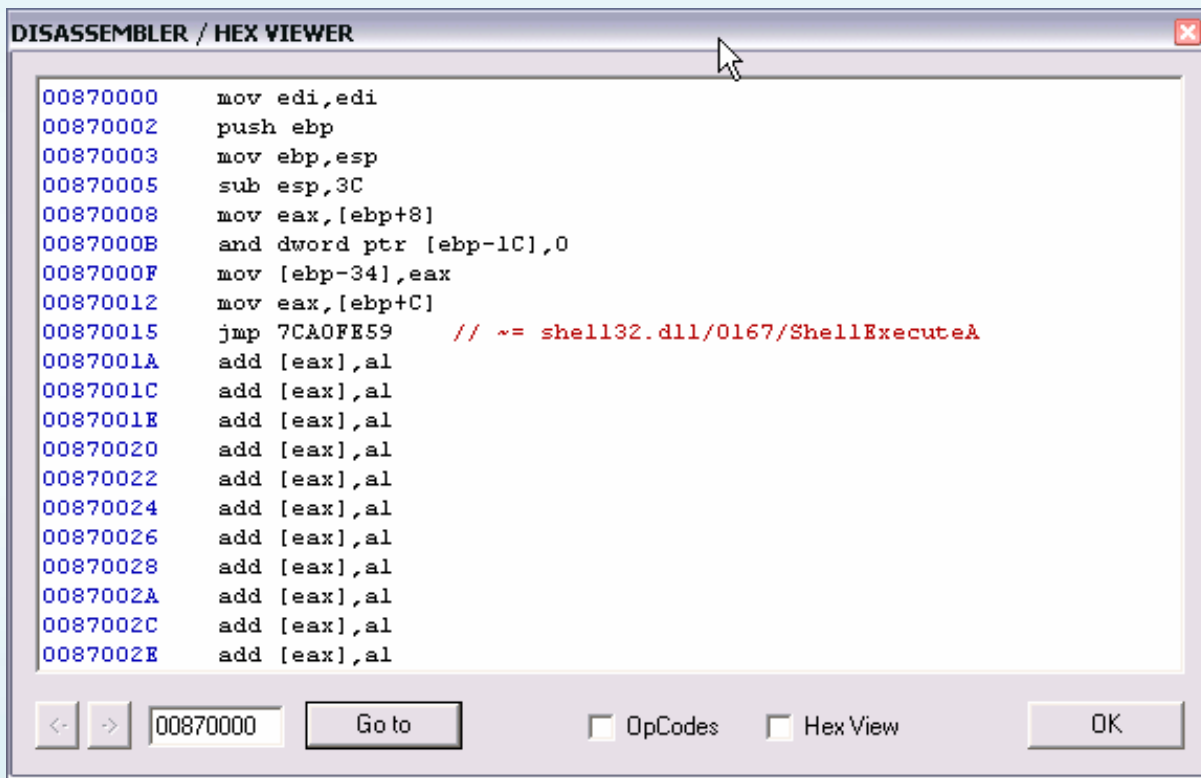
Imported Functions Found

+ comctl32.dll FTThunk:00006000 NbFunc:1 (decimal:1) valid:YES
 + comdlg32.dll FTThunk:00006008 NbFunc:1 (decimal:1) valid:YES
 + gdi32.dll FTThunk:00006010 NbFunc:D (decimal:13) valid:YES
 + kernel32.dll FTThunk:00006048 NbFunc:1 (decimal:1) valid:YES
 + kernel32.dll FTThunk:00006050 NbFunc:6 (decimal:6) valid:YES
 + kernel32.dll FTThunk:0000606C NbFunc:14 (decimal:20) valid:YES
 - ? FTThunk:000060C0 NbFunc:3 (decimal:3) valid:NO
 rva:000060C0 mod:shell32.dll ord:0167 name:ShellExecuteA
 rva:000060C4 mod:kernel32.dll ord:0000 name:EnumCalendarInfoW
 rva:000060C8 mod:user32.dll ord:0170 name:GetWindowLongW
 + user32.dll FTThunk:000060D0 NbFunc:16 (decimal:22) valid:YES
 - ? FTThunk:00006130 NbFunc:3 (decimal:3) valid:NO
 rva:00006130 ptr:00000001
 rva:00006134 ptr:00000090
 rva:00006138 ptr:00000090
 - ? FTThunk:00006140 NbFunc:3 (decimal:3) valid:NO
 rva:00006140 ptr:00000001
 rva:00006144 ptr:000000F9
 rva:00006148 ptr:000000F9
 - ? FTThunk:00006150 NbFunc:3 (decimal:3) valid:NO
 rva:00006150 ptr:00000001
 rva:00006154 ptr:000000F8
 rva:00006158 ptr:000000F8
 - ? FTThunk:00006160 NbFunc:3 (decimal:3) valid:NO
 rva:00006160 ptr:00000002
 rva:00006164 ptr:0000C0FE
 rva:00006168 ptr:0000C8FE
 - ? FTThunk:00006170 NbFunc:B (decimal:11) valid:NO
 rva:00006170 ptr:00000002

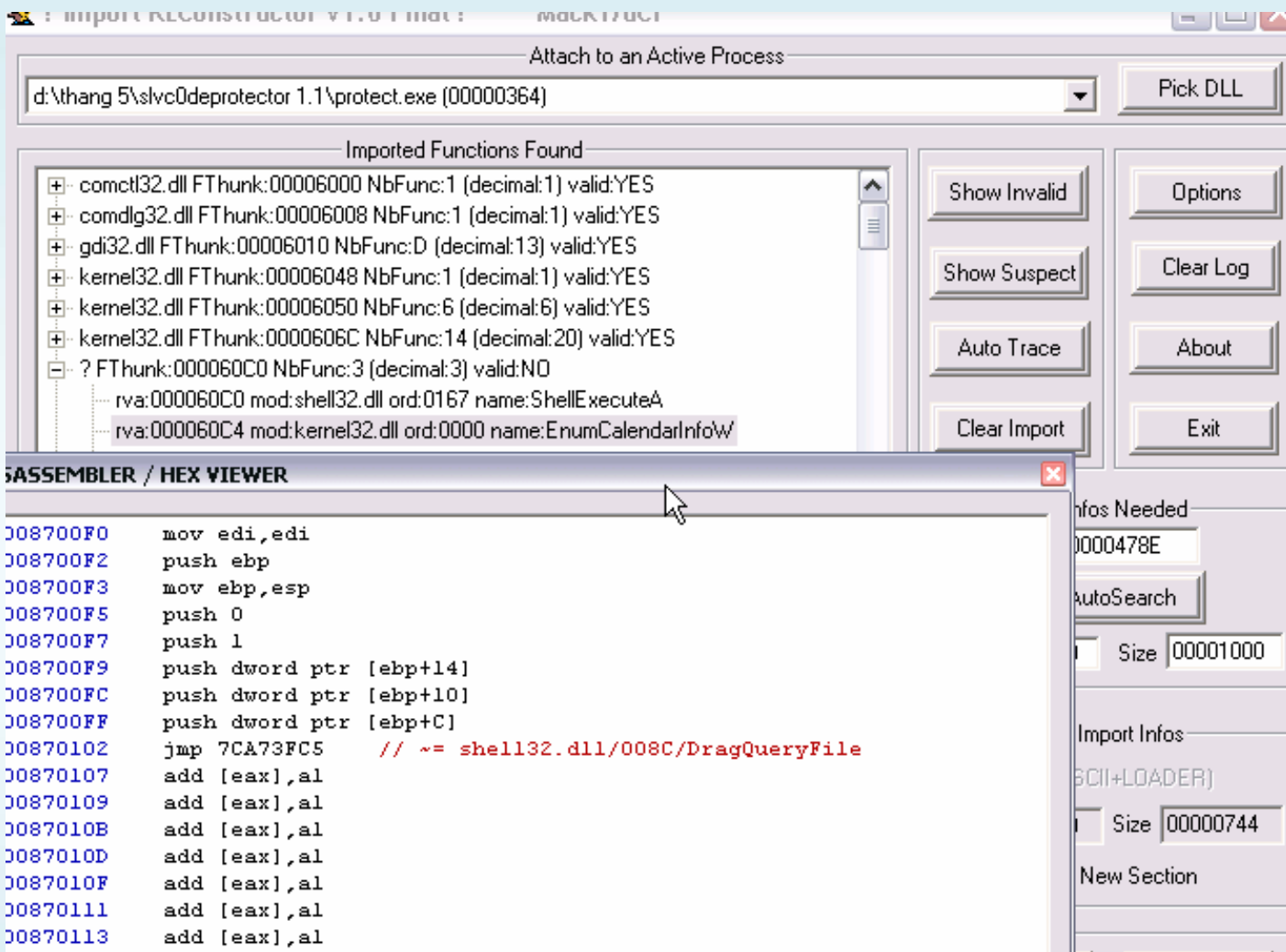
_ Hours seo ... **Cut thanks** all hả Āc default line to dust ... we need to Fix hours each child an individual child. We start choosing them first



_ Appears window **DISASSEMBLER 1 / HEX viewer**



_hehe ... This is the function of this we do need to correct. Choose 2 more children



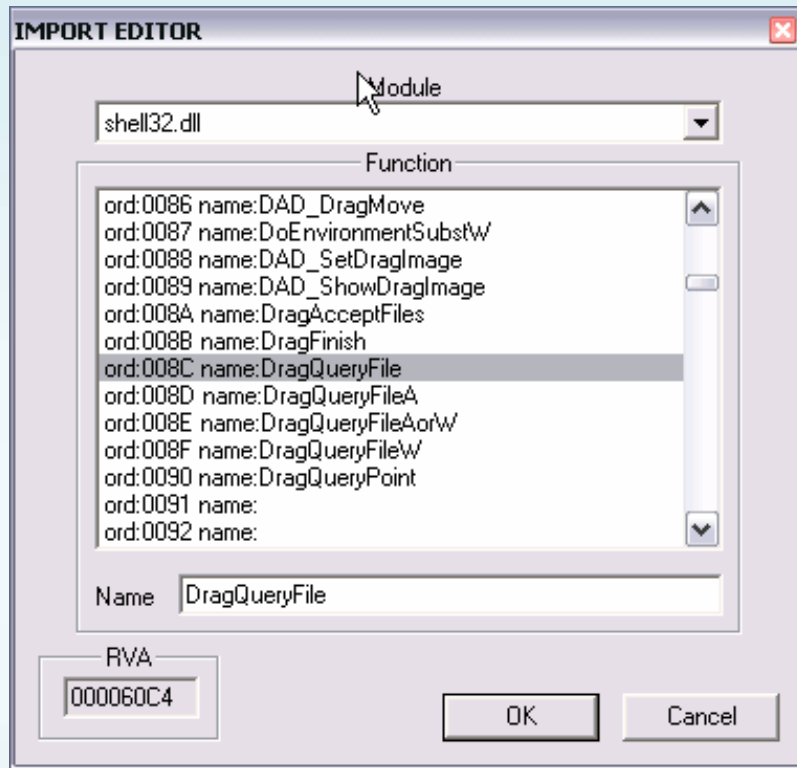
```

00870111    add [eax],al
00870113    add [eax],al
00870115    add [eax],al
00870117    add [eax],al
00870119    add [eax],al

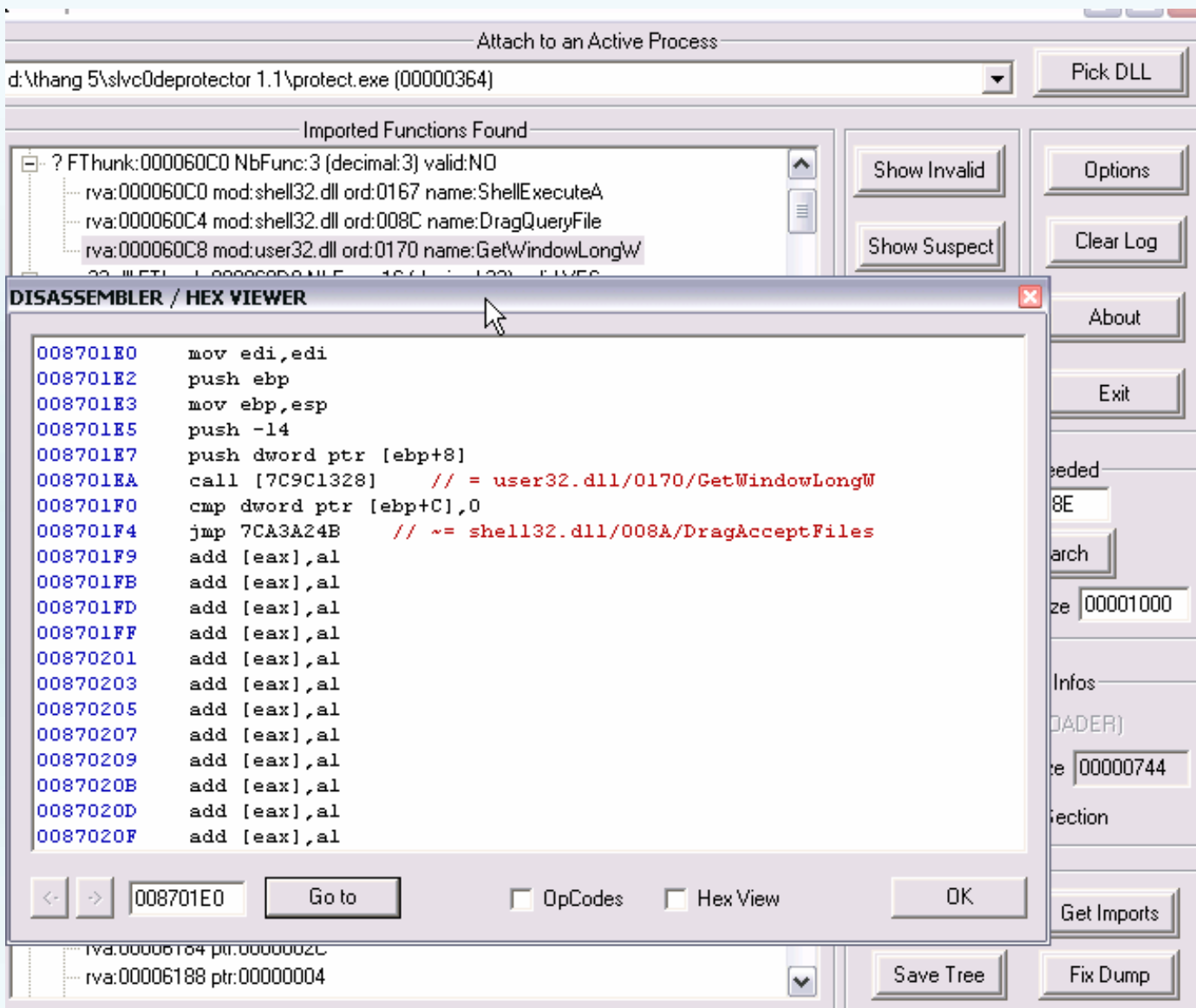
```

Get Imports

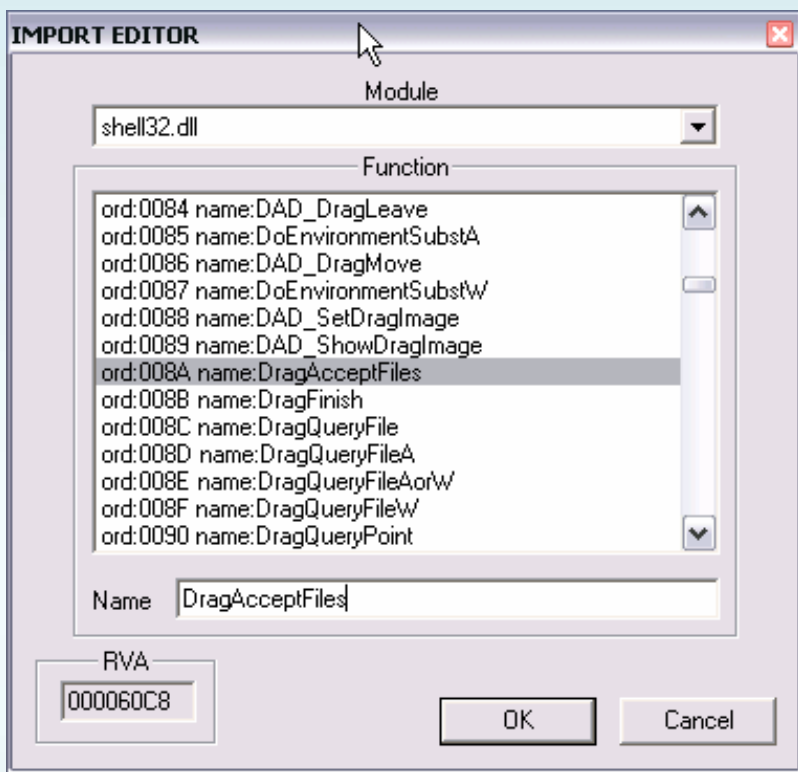
_ This function silkworms bay we revise nhusau:



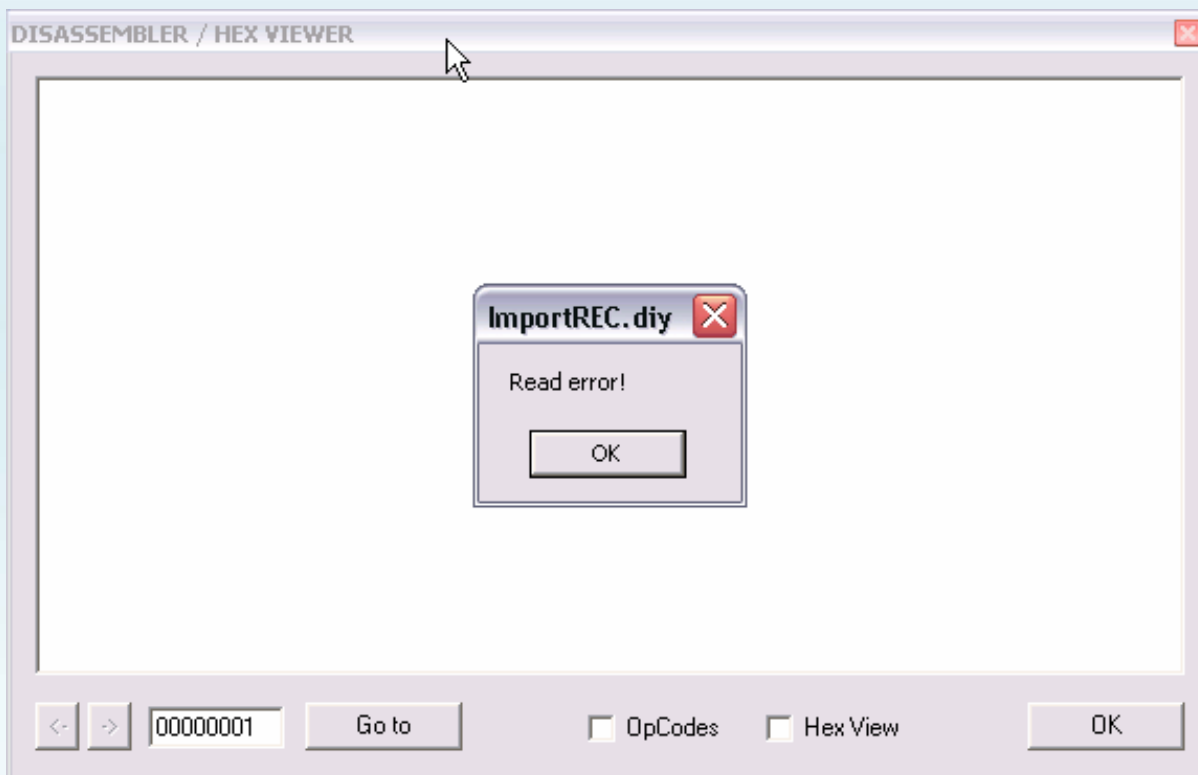
_Tiep 3 children



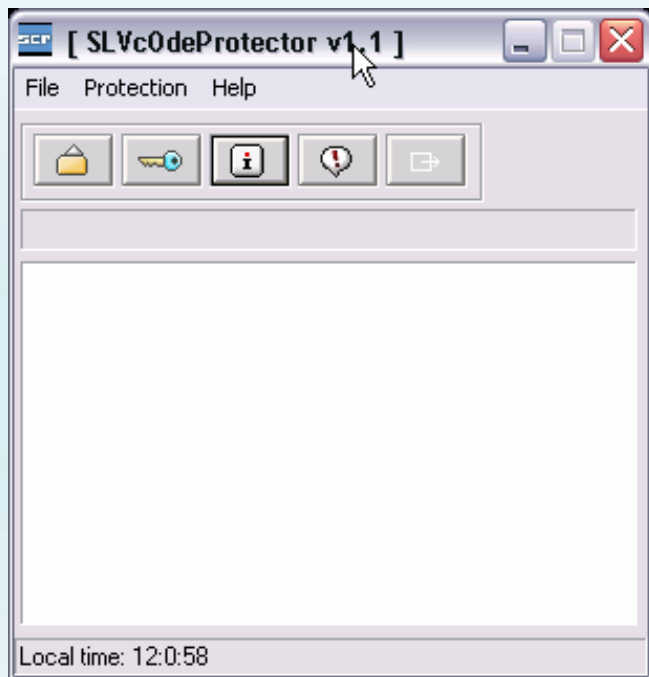
_ta to be revised nhus au:



_ N han **S or the WI and a lid** and the more that the h The h:



_ **Cut Thunks** time is up, select **Fix dump**. Run Test File **Dumped_.exe**



_H a h a h a h a n **U p a c k e d c s u c e l l s s f u ...**

G r e e T s i t a l y O u F l t : *C o m p t u r e g _ A n e l , e Z o m b i , M o o n b y a , c H a g r a p e , B e n i n a k i n e m a n o a w r , o i Z , D x u e , M e r c , i l e g H T p h o w h e r e x , c k y b o T r i i t a l y , T a k a d a i a m i d i o t , t h e l i g h t o e n i x , t h e i n t h a n d i n e , a n d q u a l i t y o f ... a n d y o u !*

N h a T r a n g , i t a l y 1 à T h e 8 t h A s i a n g 5 y e a r 2 0 0 6

W h y N o t B a r

UNPACKING SLVc0deProtector 1:11 tut 1

tlandn

Target: Protect.exe (included)

Tools: Diablo OllyDbg (This in place or not to be used to fix OllyAdvanced NumOfRVA)

In this tut I try to give a general method to unpack (by hand, including fix IAT) of the target packed with SLV c0deProtector 1.11.

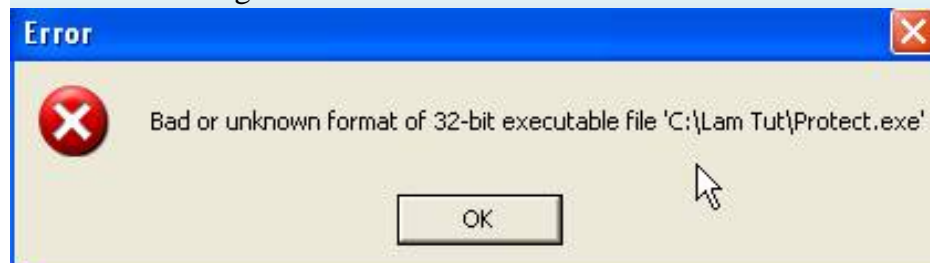
I) Find OEP:

As usual we PEiD used to check the pack with what?



Clearly the program is pack with SLVc0deProtector 1.1

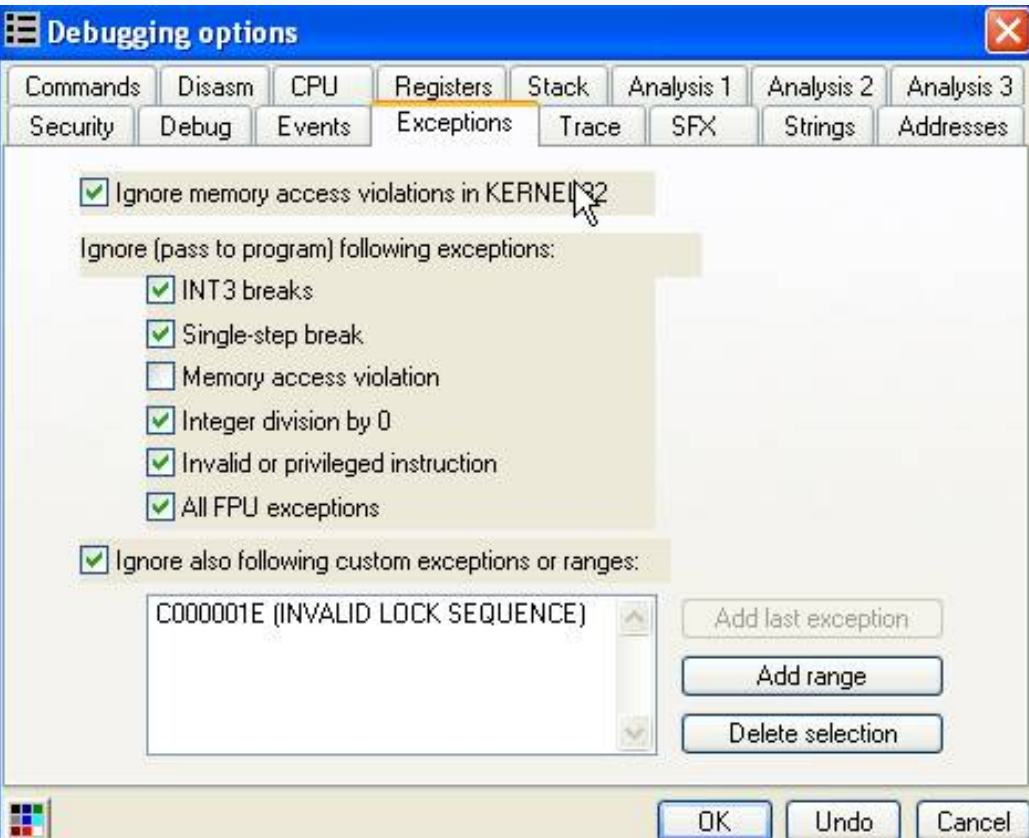
We start. Load program to OllyDbg. If you use a non OllyDbg Diablo xài OllyAdvanced or not the plugin you will get an error following:



Therefore we will xài OllyDbg version of Diablo or OllyAdvanced plugin. Here I xài OllyDbg version of Diablo for Health. Load the program. We here.

00401000	E8 00000000	call	00401005
00401005	58	pop	eax
00401006	C600 EB	mov	byte ptr [eax], 0EB
00401009	C640 01 08	mov	byte ptr [eax+1], 8

Revising Option OllyDbg by following the same:



Press Shift-F9 5 times the program running. We restart OllyDbg (press Ctrl-F2). Press Shift-F9 4 times. We here:

0040E62A	0000	add	[eax], al
0040E62C	83C4 04	add	esp, 4
0040E62F	52	push	edx
0040E630	C3	retn	
0040E631	6955 8B EC608B	imul	edx, [ebp-75], 458B60EC

Press Alt-M to Memory Map. Select the section and the second set as in the picture:

[illegible]

Press Shift-F9. We 0040478E in OEP.

0040478E	90	nop	
0040478F	90	nop	
00404790	90	nop	
00404791	90	nop	
00404792	90	nop	
00404793	E8 3A0C0000	call 004053D2	
00404798	68 C4784000	push 004078C4	ASCII "FuCh
0040479D	6A 01	push 1	
0040479F	6A 00	push 0	

II) Search for "Stolen Bytes":

You note that here we have "stolen bytes". In this program is 5 bytes corresponding 5 orders [submitted](#).

Here I try to find a general to be able to find "stolen bytes" in the target.

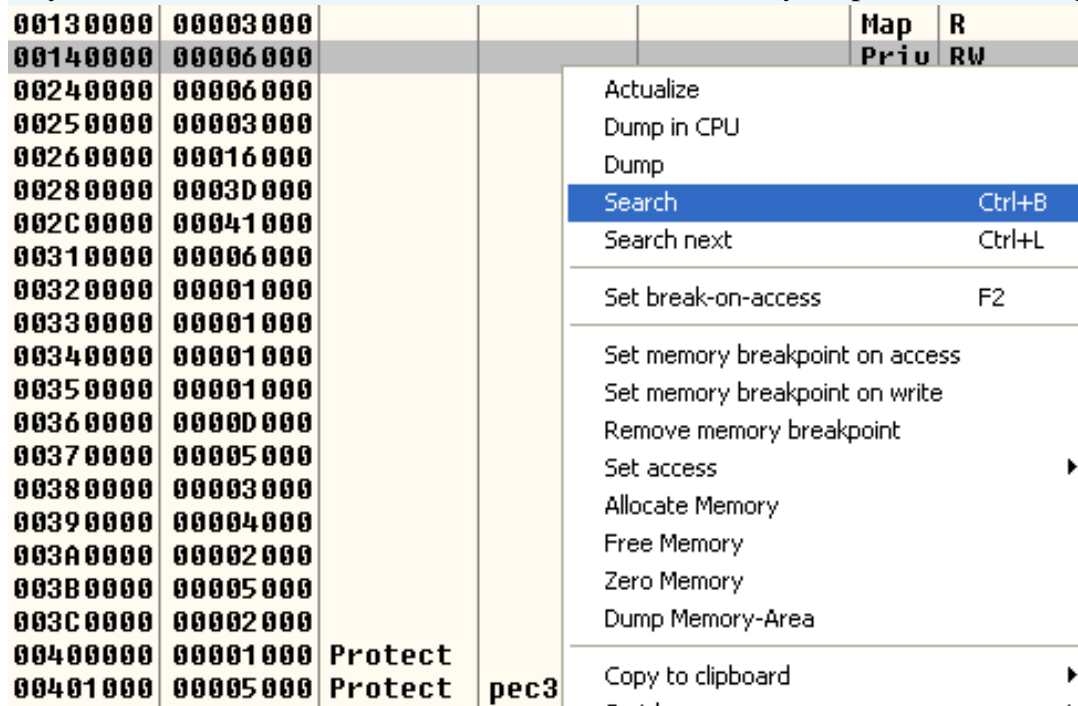
According guess I think the packer will implement "stolen bytes" where new and then jump to our OEP. We should stop in **0040478E** guess I think there will be a command

PUSH 0040478E (This line is opcodes 68 8E 47 40 00)

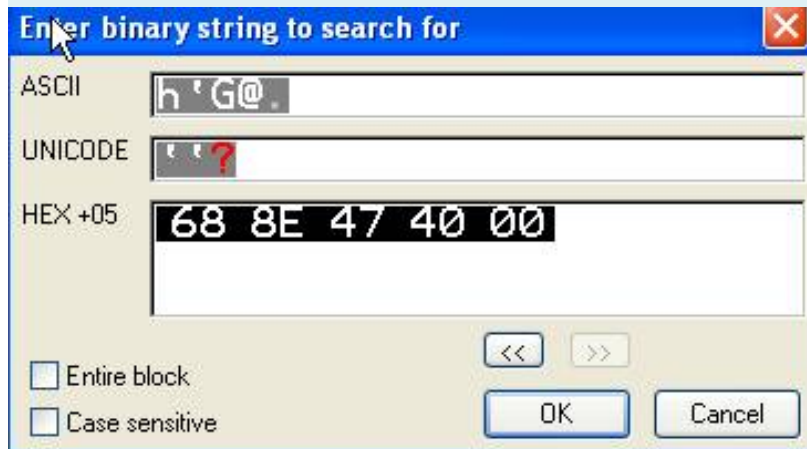
RET

Or can the **JMP 0040478E**

I try on. We are at OEP 0040478E. Press Alt-M to Memory Map. Select the image as:



Enter **68 8E 47 40 00**.



Click OK. We are.



As a result **0040E5EE**. We try to jump to see which stars. In the window CPU press Ctrl-G. Enter **0040E5EE**.

0040478E	90	nop
0040478F	90	nop
00404790	90	nop
00404791	90	nop
00404792	90	nop
00404793	E8 3A0C0000	call
00404798	68 C4784000	push

Enter expression to follow

Click OK. We here.

0040E5EA	90	nop
0040E5EB	90	nop
0040E5EC	90	nop
0040E5ED	90	nop
0040E5EE	68 8E474000	push 0040478E
0040E5F3	C3	retn

Just like we predicted. Now at the top is a little "stolen bytes". Pull up:

0040E571	68 73474000	push 00404773
0040E576	90	nop
0040E577	90	nop

We find the line:

PUSH 00404773

This line corresponds 5 bytes (68 73 47 40 00). This is "stolen bytes" our.

Now press the "*" to return to OEP **0040478E**. Press Spacebar to enter the command line:

0040478E	90	nop
0040478F	90	nop
00404790	90	nop
00404791	90	nop
00404792	90	nop
00404793	E8 3A0C0000	call
00404798	68 C4784000	push
0040479D	6A 01	push

Assemble at 0040478E

☒ Fill with NOP's

We are.

0040478E	68 73474000	push 00404773
00404793	E8 3A0C0000	call 004053D2
00404798	68 C4784000	push 004078C4
0040479D	6A 01	push 1
0040479F	6A 00	push 0

We will dump with OllyDump. Choose the type:

File View Debug **Plugins** Options Window Help Tools Custom Languages

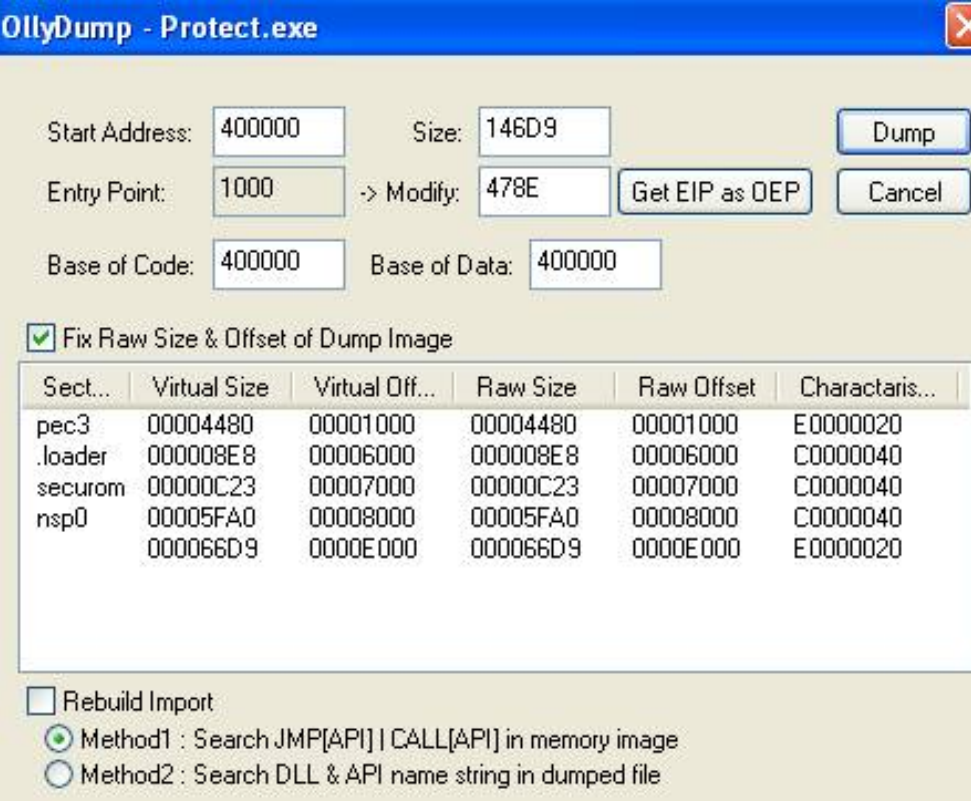
Paused

1 AnalyzeThis
2 ApiBreak
3 RL!APIFinder
4 Asm2Clipboard
5 Bookmarks
6 Borland Map File Importer
7 CleanUp
8 CommandBar
9 Code Ripper
0 Data Ripper
FindCrypt
GODUP Plugin
Labelmaster
OdbgScript
OllyDump
OllyScript

L E M T W H C
04773
053D2
078C4
05360
0538A
, 0B7
rt 004047B4
rt 00404812
05300
Dump debugged process

Address	Hex	du
0040478E	68 7	
00404793	E8 3	
00404798	68 C	
0040479D	6A 0	
0040479F	6A 0	
004047A1	E8 B	
004047A6	E8 D	
004047AB	35 B	
004047B0	75 0	
004047B2	EB 5	
004047B4	6A 0	
004047B6	E8 E	
004047BB	A3 0	

Choose the type:



OllyDump - Protect.exe

Start Address: 400000 Size: 146D9 **Dump**

Entry Point: 1000 -> Modify: 478E **Get EIP as OEP** **Cancel**

Base of Code: 400000 Base of Data: 400000

☒ Fix Raw Size & Offset of Dump Image

Sect...	Virtual Size	Virtual Off...	Raw Size	Raw Offset	Charactaris...
pec3	00004480	00001000	00004480	00001000	E0000020
.loader	000008E8	00006000	000008E8	00006000	C0000040
securom	00000C23	00007000	00000C23	00007000	C0000040
nsp0	00005FA0	00008000	00005FA0	00008000	C0000040
	000066D9	0000E000	000066D9	0000E000	E0000020

☐ Rebuild Import

☒ Method1 : Search JMP[API] | CALL[API] in memory image

☐ Method2 : Search DLL & API name string in dumped file

Click the "dump". File name is a.exe

III) Fix IAT:

Now to fix the period of IAT. I will find "magic point" and patch it.

At OEP. Press Ctrl-B. Fill **FF 25**.



Enter binary string to search for

ASCII: %

UNICODE: ¢

HEX +02: FF 25

☐ Entire block

☐ Case sensitive

OK **Cancel**

Click OK. We here:

004052DF	E8 7E010000	call	00405462
004052E4	C9	leave	
004052E5	C2 0C00	retn	0C
004052E8	- FF25 C8604000	jmp	[4060C8]
004052EE	- FF25 C4604000	jmp	[4060C4]
004052F4	- FF25 C0604000	jmp	[4060C0]
004052FA	- FF25 10604000	jmp	[406010]
00405300	- FF25 14604000	jmp	[406014]

Click your mouse to select the image.

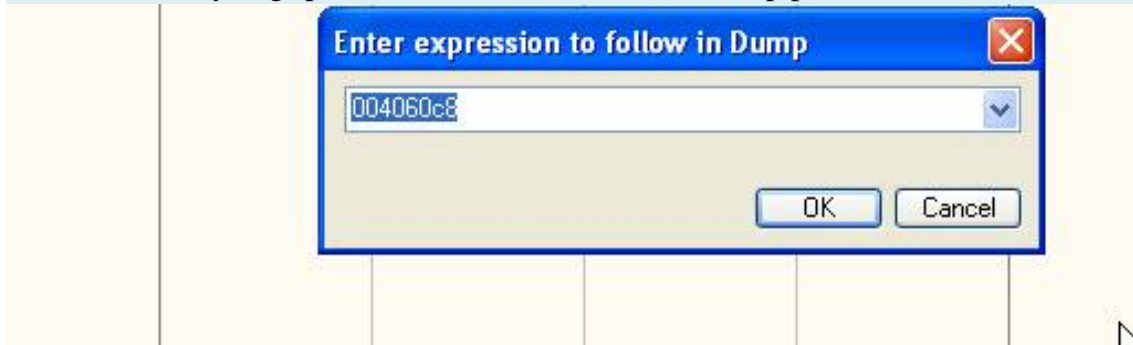
004052E4	C9	leave		New origin here	Ctrl+Gray *
004052E5	C2 0C00	retn	0C	Go to	
004052E8	- FF25 C8604000	jmp	[4060C8]	Follow in Dump	Selection
004052EE	- FF25 C4604000	jmp	[4060C4]		Memory address
004052F4	- FF25 C0604000	jmp	[4060C0]		

In the window dump.

004060C8	E0 01 39 00	00 00 00 00	30 2A 39 00	20 2B 39 00
004060D8	10 2C 39 00	00 2D 39 00	F0 2D 39 00	E0 2E 39 00
004060E8	D0 2F 39 00	C0 30 39 00	B0 31 39 00	A0 32 39 00
004060F8	90 33 39 00	80 34 39 00	70 35 39 00	60 36 39 00

The value false IAT form 0039XXXX (translated back from XX XX 39 00).

Now restart OllyDbg (press Ctrl-F2). In the window dump press Ctrl-G to enter 004060C8.



Click OK. We here.

004060C8	C7 00 8B 1B	2B 3B 4B DC	F6 83 FC AC	B8 43 BC 6C
004060D8	4A 04 7C 2C	DD C4 3C ED	E9 84 FD AD	05 44 BD 6D

Select 4 bytes and mouse click to select the image.

004060C8	C7 00 8B 1B	Breakpoint	Memory, on access
004060D8	4A 04 7C 2C	Search for	Memory, on write

Now press Shift-F9. Stop here by Exception.

Address	Hex dump	Disassembly
0040E20B	AD	lods dword ptr [esi]
0040E20C	A0 8B642414	mov al, [1424648B]
0040E211	64:8F05 000000	pop dword ptr fs:[0]

Press Shift-F9 again. Stop here

Address	Hex dump	Disassembly
0040F40C	AA	stos byte ptr es:[edi]
0040F40D	^ E2 B9	loopd short 0040F3C8
0040F40F	↓ EB 01	jmp short 0040F412
0040F411	69C3 696183C6	imul eax, ebx, C6836169

This is not important. Continue pressing F9 4 times more. We stop here:

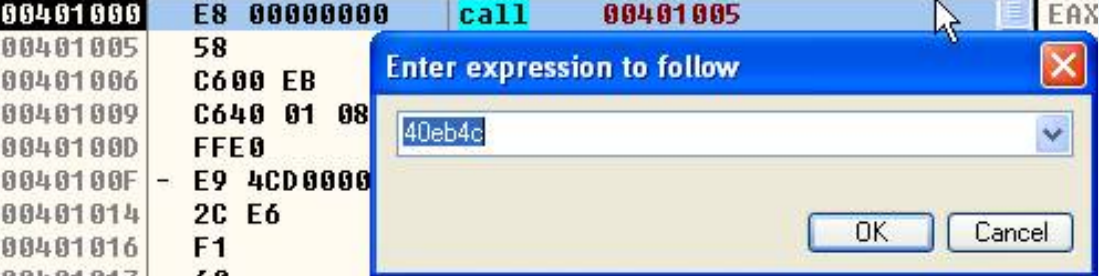
0040EB46	8D7F 01	lea edi, [edi+1]	
0040EB49	8B77 04	mov esi, [edi+4]	
0040EB4C	56	push esi	
0040EB4D	8F02	pop dword ptr [edx]	003901E0
0040EB4F	83F0 01	xor eax, 1	
0040EB52	↓ EB 01	jmp short 0040EB55	

This new place is really important. Look below the window.

Stack [0012FB80]=003901E0 (003901E0)
ds:[004060C8]=0000642C

Packer is trying to record the value 003901E0 to address 004060C8. Looking through the window to write.

Press Ctrl-F2 reboot. Press Ctrl-G fill 0040EB4C.



Address	Hex dump	Disassembly
0040EB4C	8F	???
0040EB4D	A6	cmps byte ptr [esi], byte ptr es:[edi]
0040EB4E	75 35	jnz short 0040EB85
0040EB4F	75 35	jnz short 0040EB85

Click your mouse to select the image.

Address	Hex dump	Copy		Comme
0040EB4C	8F	Binary		Unkno
0040EB4D	A6	Assemble	Space	
0040EB4E	75 35	Label	:	
0040EB50	3960 DB	Comment	;	
0040EB53	D10A	Breakpoint		
0040EB55	A4	Run trace		
0040EB56	90			
0040EB57	42	New origin here	Ctrl+Gray *	
0040EB58	44	Go to		
0040EB59	2169 50	Follow in Dump		
		Search for		
		Find references to		
		View		
		Copy to executable		

Now press Shift-F9 to break in when 0040EB4C (2).

Address	Hex dump	Disassembly
0040EB4C	56	push esi
0040EB4D	8F 02	pop dword ptr [edx]
0040EB4F	83 F0 01	xor eax, 1

Edit the ESI PUSH PUSH EAX. Press Spacebar to enter the picture:

Address	Hex dump	Disassembly	Comment	Register
0040EB4C	56	push esi		
0040EB4D	8F 02	pop dword ptr [edx]		
0040EB4F	83 F0 01	xor eax, 1		
0040EB52	EB 01	jmp short 0040EB54		
0040EB54	40	inc eax		
0040EB55	83 F0 01	xor eax, 1		
0040EB58	60	pushad		

Assemble at 0040EB4C

push eax

☒ Fill with NOP's

Assemble

Cancel

Click assemble.

Address	Hex dump	Disassembly	Comment
0040EB4C	50	push eax	shell3
0040EB4D	8F 02	pop dword ptr [edx]	
0040EB4F	83 F0 01	xor eax, 1	
0040EB52	EB 01	jmp short 0040EB55	

After editing is completed, we put in the Hardware Breakpoint. Choose the type:

Address	Hex dump	Copy	Binary	Undo selection	Alt+BkSp	Assemble	Space	Label	:	Comment	;	Breakpoint	Toggle	F2	Conditional	Shift+	Conditional log	Shift+	Memory, on access	Memory, on write	Hardware, on execution	Remove hardware breakpoint
0040EB4C	50																					
0040EB4D	8F 02																					
0040EB4F	83 F0 01																					
0040EB52	EB 01																					
0040EB54	40																					
0040EB55	83 F0 01																					
0040EB58	60																					
0040EB59	8BD8																					
0040EB5B	8BFE																					
0040EB5D	8985 752040																					
eax=7CA73FB3 (shell32)																						

Press Ctrl-G to enter OEP (0040478E).

Address	Hex dump	Disassembly	Comment
0040EB4C	50	push eax	
0040EB4D	8F 02	pop dword ptr [edx]	
0040EB4F	83 F0 01	xor eax, 1	
0040EB52	EB 01	jmp short 0040EB54	
0040EB54	40	inc eax	
0040EB55	83 F0 01	xor eax, 1	
0040EB58	60	pushad	

Enter expression to follow

40478e

OK

Cancel

Press F2 breakpoint in the set.

0040478E	90	nop	
0040478F	90	nop	
00404790	90	nop	
00404791	90	nop	
00404792	90	nop	
00404793	E8 3A0C0000	call 004053D2	

Uncheck Hardware Breakpoint.

Press Shift-F9 until the break in the OEP (4 times).

0040478E	90	nop	
0040478F	90	nop	
00404790	90	nop	
00404791	90	nop	
00404792	90	nop	
00404793	E8 3A0C0000	call	004053D2
00404798	68 C4784000	push	004078C4
0040479D	60 01	push	1

jmp to kernel32.SetUnhai
ASCII "FuCk... I hate EI

Now Imprec used to fix IAT. Open Imprec. Enter OEP: 478E. Click "IAT AutoSearch". Then "Get Imports".

Attach to an Active Process
 c:\lam tut\protect.exe (00000B98)

Imported Functions Found

- + comctl32.dll FTThunk:00006000 NbFunc:1 (decimal:1) valid:YES
- + comdlg32.dll FTThunk:00006008 NbFunc:1 (decimal:1) valid:YES
- + gdi32.dll FTThunk:00006010 NbFunc:1 (decimal:13) valid:YES
- + imagehlp.dll FTThunk:00006048 NbFunc:1 (decimal:1) valid:YES
- + kernel32.dll FTThunk:00006050 NbFunc:1B (decimal:27) valid:YES
- + shell32.dll FTThunk:000060C0 NbFunc:3 (decimal:3) valid:YES
- + user32.dll FTThunk:000060D0 NbFunc:16 (decimal:22) valid:YES

Log

IAT read successfully.
rva:00006078 forwarded from mod:ntdll.dll ord:015C name:RtlGetLastWin32Error

Current imports:
7 (decimal:7) valid module(s) (added: +7 (decimal:+7))
44 (decimal:68) imported function(s). (added: +44 (decimal:+68))

IAT Infos needed
 OEP 0000478E IAT AutoSearch
 RVA 00006000 Size 0000012C

New Import Infos (IID+ASCII+LOADER)
 RVA 00000000 Size 00000684
☒ Add new section

Load Tree Save Tree Get Imports Fix Dump

Click "Fix dump" file selected a.exe. We are a_.exe file. Test. Good!

IV) SLVc0deProtector Killer v1.1:

This is a program for SLVc0deProtector unpacker v1.1 written by Super Cracker. You can use this program to unpack our target (you note for a target program is not unpack).

Them. Happy happy. Hopefully you will learn something new through this tut.

Greetingz: All Members Reaonline.net and you.

tlandn

20-May-2006

UNPACKING SLVc0deProtector 1:11 tut 2

tlandn

Target: crackme.scp.exe (included)

Tools: Diablo OllyDbg, weasle 0.5, the script (included)

Sitting sad do not know what to write tut for your reference. At 1 in the tut I explained very well how to unpack by hand (including fix IAT). However, in human life is often like something so fast in this tut I will talk about how to use scripts and unpack plugin 0.5 weasle

I) Find Stolen OEP and Bytes:

As usual we PEId used to check the pack with what?

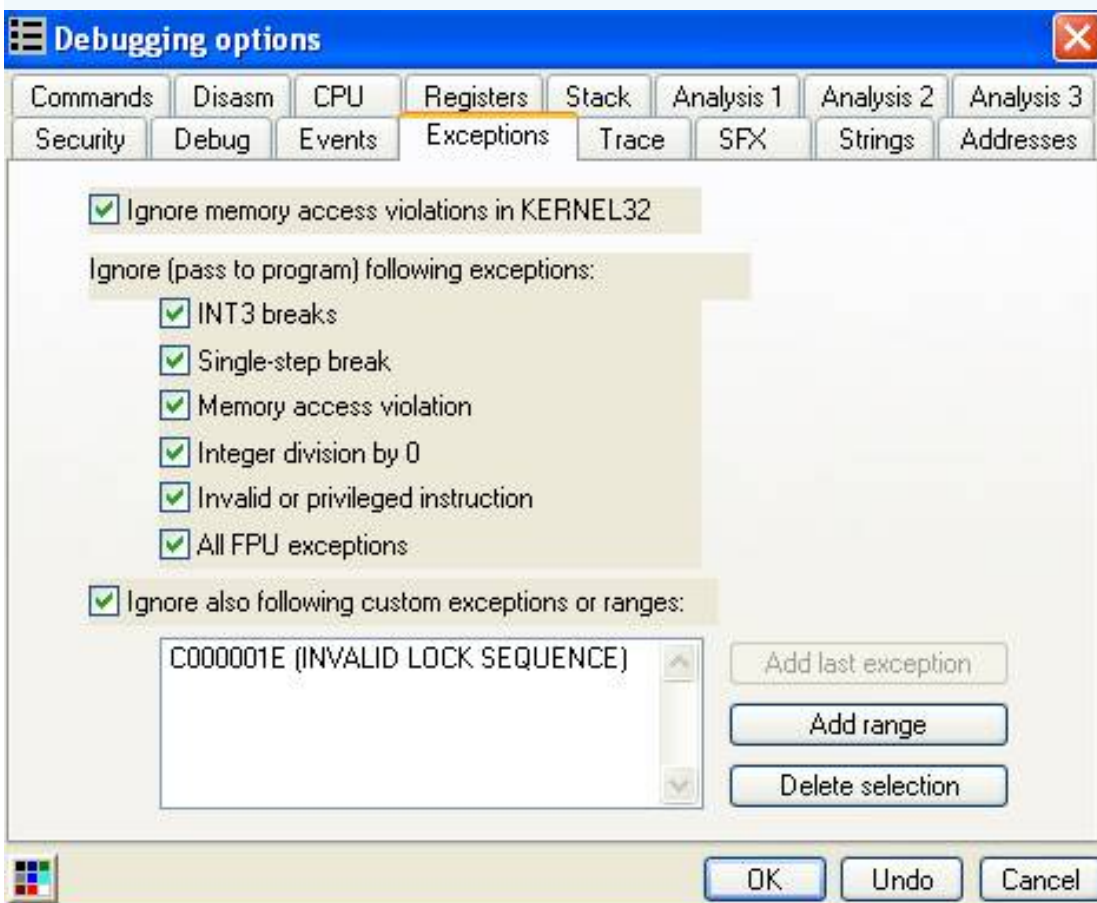


Clearly the program is pack with SLVc0deProtector 1.1

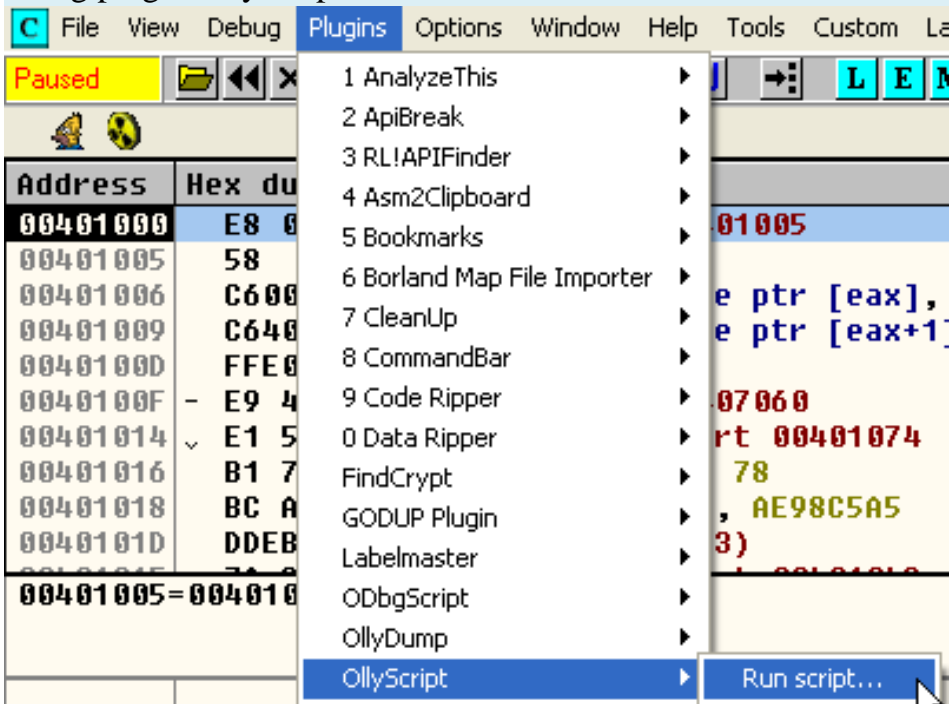
We start. Load program to OllyDbg.

Address	Hex dump	Disassembly
00401000	E8 00000000	call 00401005
00401005	58	pop eax
00401006	C600 EB	mov byte ptr [eax], 0EB
00401009	C640 01 08	mov byte ptr [eax+1], 8

Adjust the number of Option OllyDbg similar to following:



Using plugin OllyScript.



Select the script file "SLVc0deProtector 1.1.txt" (accompanied by tut). It runs a stop to it here:

Address	Hex dump	Disassembly	Comment
00407123	59	pop ecx	
00407124	E2 D0	loadd short 004070F6	
00407126	8D85 4C1B		
0040712C	50		
0040712D	64:FF35 0		
00407134	64:8925 0		
0040713B	E8 250000		
00407140	95		
00407141	E8 0D0000		
00407146	59		

Stack [0012FF9C]=00
ecx=FFFFFFFF

Error



Don't know how to bypass command at address 0040723C. Try to change EIP or pass exception to program.

OK



Click OK and then Shift-F9. We stopped at the OEP 0040758E.

Address	Hex dump	Disassembly	Comment
0040758E	55	push ebp	Stolen OEP!
0040758F	8BEC	mov ebp, esp	
00407591	6A FF	push -1	
00407593	68 F8404000	push 004040F8	
00407598	68 F41D4000	push 00401DF4	
0040759D	64:A1 00000000	mov eax, fs:[0]	
004075A3	50	push eax	
004075A4	64:8925 000000	mov fs:[0], esp	
004075AB	83EC 58	sub esp, 58	

Dump in OllyDump.

File View Debug Plugins Options Window Help Tools Custom Languages

Paused

1 AnalyzeThis
2 ApiBreak
3 RLIAPIFinder
4 Asm2Clipboard
5 Bookmarks
6 Borland Map File Importer
7 CleanUp
8 CommandBar
9 Code Ripper
0 Data Ripper
FindCrypt
GODUP Plugin
Labelmaster
ODbgScript
OllyDump
OllyScript
OllyVBHelper
OllyDbg PE Dumper

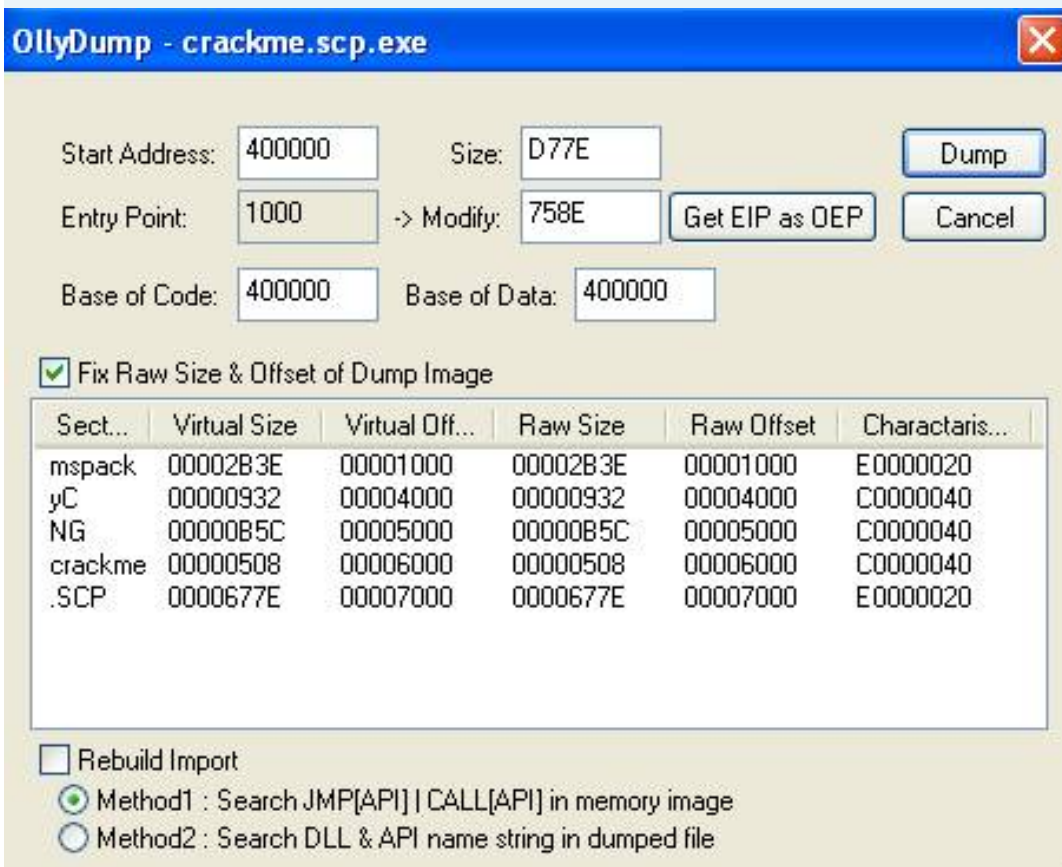
→ L E M T W H C P

→ , esp
→ 040F8
→ 01DF4
→ , fs:[0]
→ [0], esp
→ , 58

→ Dump debugged process
→ Find OEP by Section Hop (Trace into)
→ Find OEP by Section Hop (Trace over)

Address Hex du
0040758E 55
0040758F 8BEC
00407591 6A F
00407593 68 F
00407598 68 F
0040759D 64:A
004075A3 50
004075A4 64:8
004075AB 83EC
004075AE 53
ebp=004070EA (c

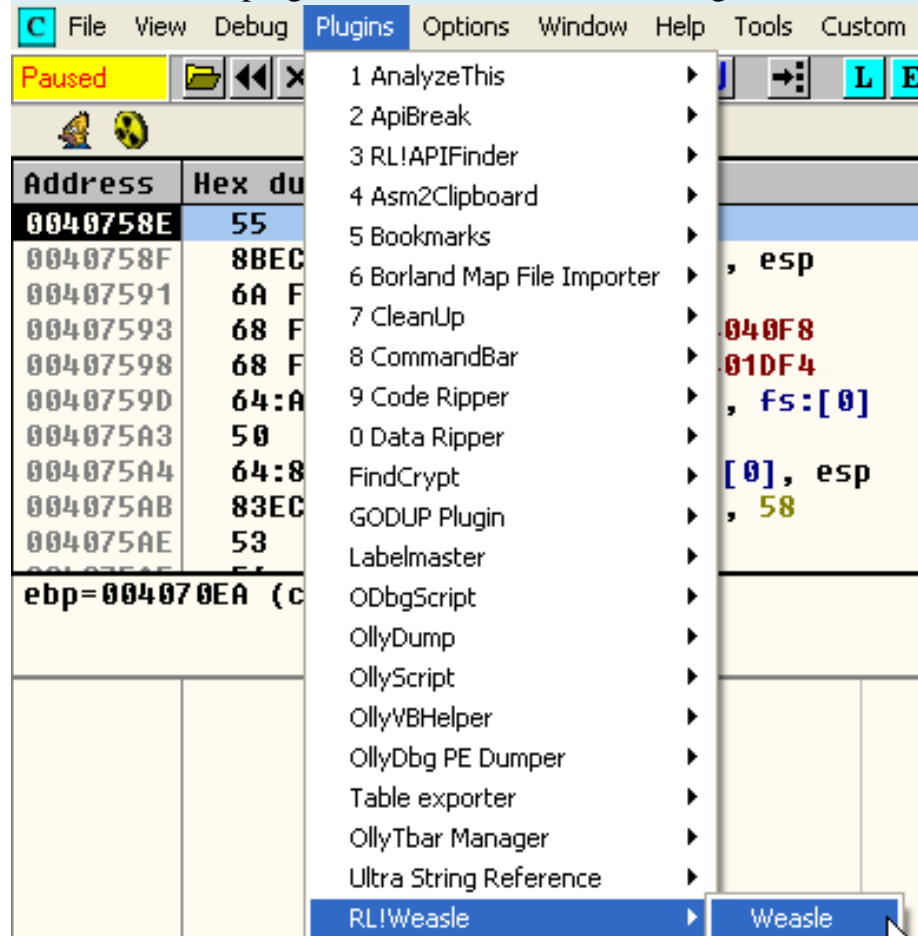
Select the image.



Name the file a.exe

II) Fix IAT:

We will use 0.5 plugin weasle to fix IAT. Running weasle.



We will make a few details to fill in weasle Imprec. Booting Imprec. Select the correct process (crackme.scp.exe). Enter OEP: 758E (0040758E - 00,400,000). Click "IAT AutoSearch".

IAT Infos needed

OEP

RVA Size

Search the **RVA: 00004000** and **Size: E4**

Enter the number on the "Search Options" in weasle like in the picture:

Search options

From address: hex

Size: hex

Note the fill the "From address:" we must add to **RVA 00400000** (get in Imprec)

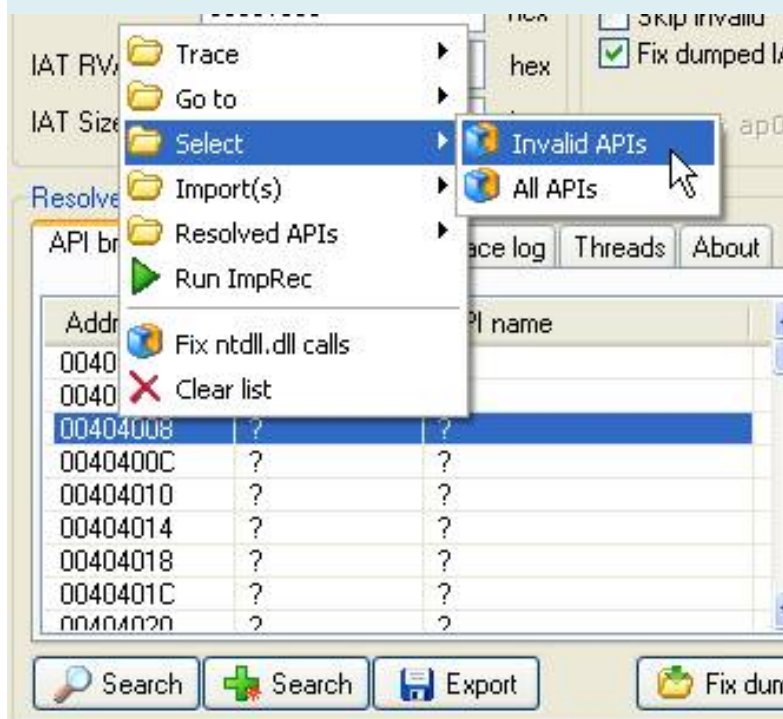
Click "Search."

Resolved API's

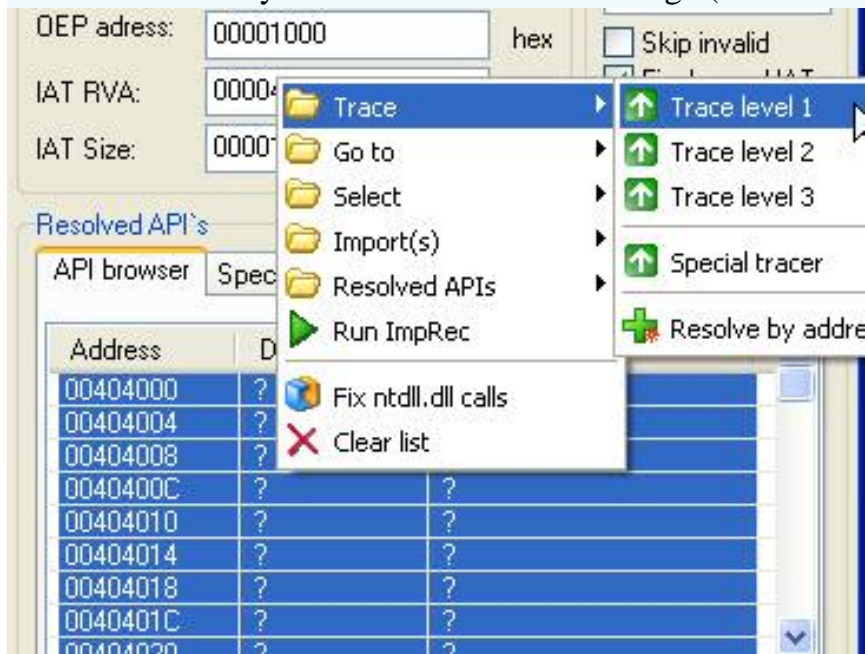
API browser Specialized tracer Trace log Threads About

Address	Dll	API name
00404000	?	?
00404004	?	?
00404008	?	?
0040400C	?	?
00404010	?	?
00404014	?	?
00404018	?	?
0040401C	?	?
00404020	?	?

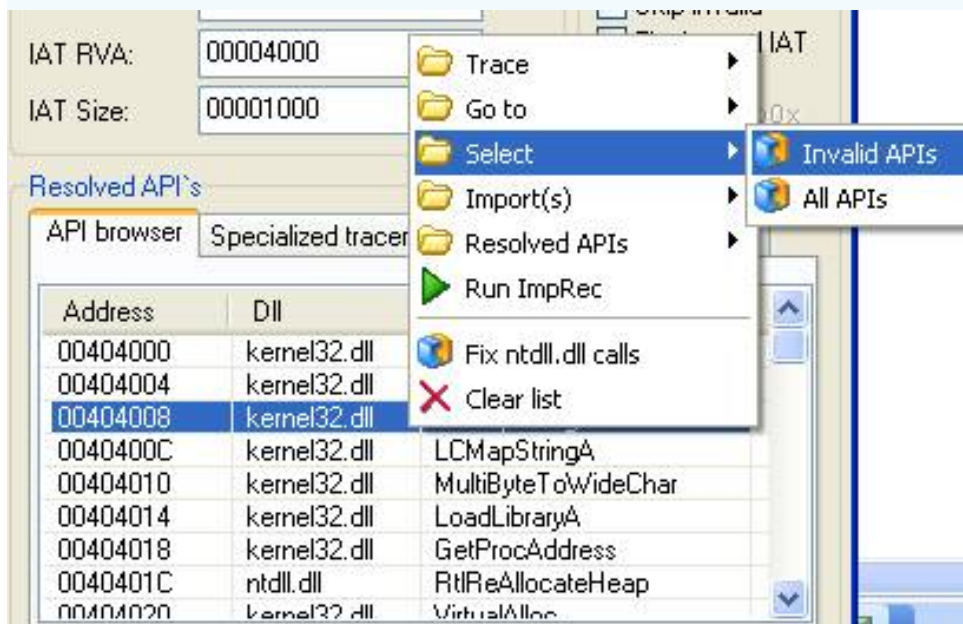
Click your mouse to select the image:



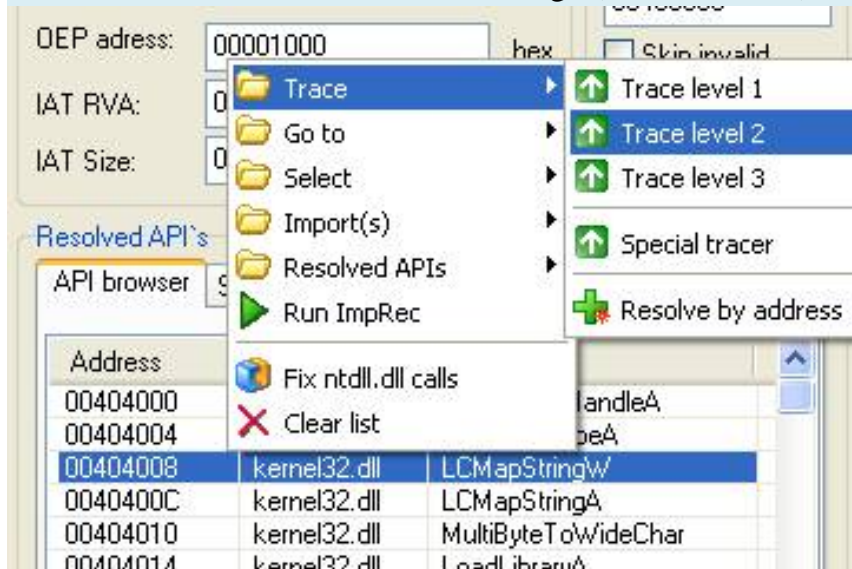
Continue to click your mouse to select the image (Trace Level 1).



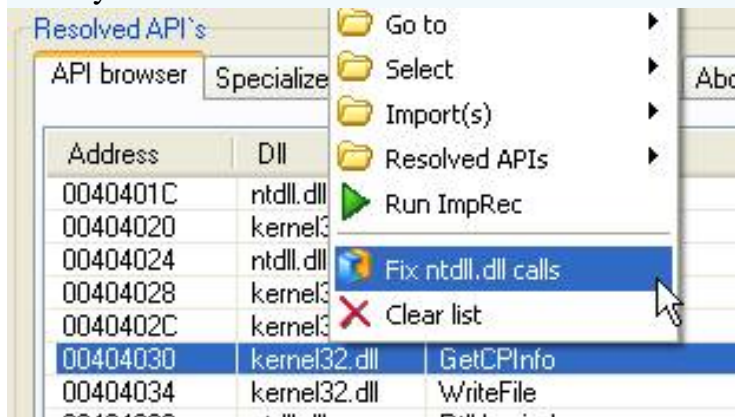
Try again with Trace Level 2. Click your mouse to select the image:



Then click the mouse to select the image (Trace Level 2).



Click your mouse to select the "Fix ntdll.dll calls."



Before export IAT this Imprec we must enter the parameters to the "Options Imprec"

ImpRec Options

OEP address: hex

IAT RVA: hex

IAT Size: hex

Click the Export button. Save as IAT.txt

Close weasle again. Using Imprec IAT.txt to load the file (click "Load Tree")

Imported Functions Found

- + kernel32.dll FTThunk:00004000 NbFunc:24 (decimal:36) valid:YES
- + user32.dll FTThunk:00004094 NbFunc:13 (decimal:19) valid:YES

Log

C:\Lam Tut\A_.exe saved successfully.
File loaded successfully.

Current imports:

2 (decimal:2) valid module(s) (added: +2 (decimal:+2))
37 (decimal:55) imported function(s). (added: +37 (decimal:+55))

IAT Infos needed

OEP

RVA Size

New Import Infos (IID+ASCII+LOADER)

RVA Size

☒ Add new section

Click "Fix dump" file selected a.exe. We are a_.exe file. Test. Good!

IV) SLVc0deProtector Killer v1.1:

For target this program does not work.

Them. Happy happy. Hopefully you will learn something new through this tut.

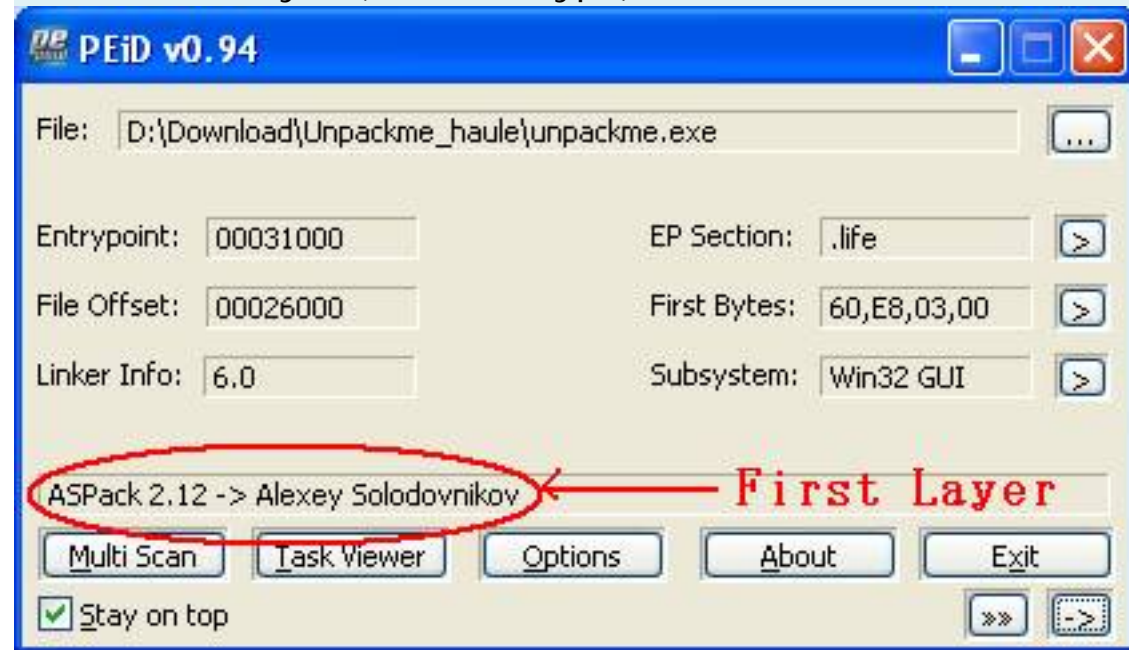
Greetingz: All Members Reaonline.net and you.

tlandn

20-May-2006

Unpacking Unpackme (ASPack MSLRH +) Author: REA trickyboy ()

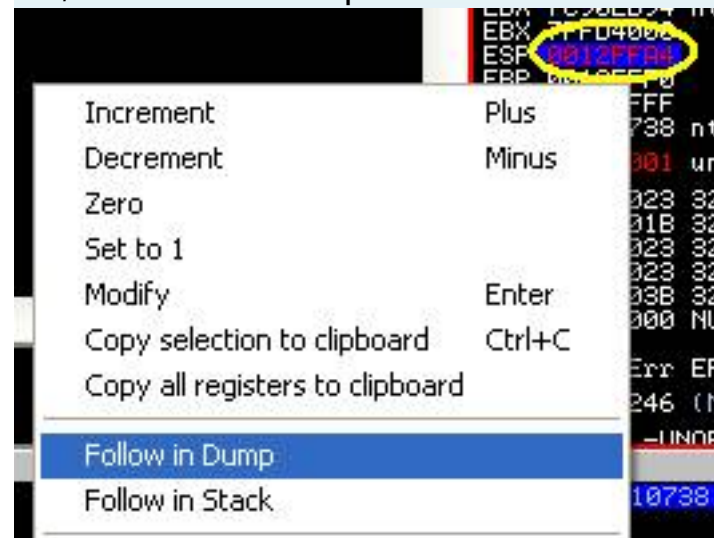
Tools: OllyDBG, Plugin HideOD, PETools, LordPE ... and more: D
ASPack first layer (or of the type):



Load Olly:

Address	Hex dump	Disassembly
00431000	60	PUSHAD
00431001	E8 03000000	CALL unpackme.00431009
00431006	E9 EB045D45	JMP 45A014F6
0043100B	55	PUSH EBP
0043100C	C3	RET
0043100D	E8 01000000	CALL unpackme.00431013

F8, and then dump to Follow in record ESP:



Fill up to 4 bytes, click to select:

Address	Hex dump	Backup	Copy	Binary	Breakpoint	Search for	Follow DWORD in Disassembler	Follow DWORD in Dump	Go to	ASCII	Memory, on access	Memory, on write	Hardware, on access	Hardware, on write	Hardware, on execution	Address	Value
0012FFA4	38 07 91 7C									00 8·a!						0012FFA4	7C91
0012FFB4	00 40 FD 7F									00 .@²ô\$e!						0012FFA8	FFFF
0012FFC4	4F 6D 81 7C									7F Omü!8·a!						0012FFAC	0012
0012FFD4	FA 22 55 80									FF "UC5 + 5·ii						0012FFB0	0012
0012FFE4	F3 99 83 7C															2FFB4	7FFD
0012FFF4	00 00 00 00															2FFB8	7C90
																2FFBC	0012
																2FFC0	0000
																2FFC4	7C91

F9 one, the EP's **MSLRH**, preparing to dump Top Layer:

Address	Hex dump	Disassembly	Comment
00431022	60	PUSHAD	← EP of MSLRH
00431023	EB 05	JMP SHORT unpackme.0043102A	
00431025	E8 EB044000	CALL 00431515	
0043102A	EB FA	JMP SHORT unpackme.00431026	
0043102C	E8 0A000000	CALL unpackme.0043103B	
00431031	E8 EB0C0000	CALL unpackme.00431021	
00431036	E8 F6FFFFFF	CALL unpackme.00431031	
0043103B	E8 F2FFFFFF	CALL unpackme.00431032	
00431040	83C4 08	ADD ESP, 8	
00431043	74 04	JE SHORT unpackme.00431049	
00431045	75 02	JNE SHORT unpackme.00431049	
00431047	EB 02	JMP SHORT unpackme.0043104B	
00431049	EB 01	JMP SHORT unpackme.0043104C	
0043104B	81E8 0A000000	SUB EAX, 0A	

Đặc trưng của [MSLRH]

Search by IAT **GetModuleHandleA** search function (usually target is available):

Command: HEX: 7C80B529 -

Byte reverse search function on the table will see IAT here:

Address	Value	Comment
00402000	7C81EE79	kernel32.lstrcmpA
00402004	7C80C6E0	kernel32.lstrlenA
00402008	7C80B529	kernel32.GetModuleHandleA
0040200C	7C801EEE	kernel32.GetStartupInfoA
00402010	00000000	
00402014	73DE224E	MFC42.#5731
00402018	73E69FED	MFC42.#3922
0040201C	73DE1CF0	MFC42.#1089
00402020	73DE97A0	MFC42.#2512
00402024	73DE1273	MFC42.#5199
00402028	73DE0CC9	MFC42.#2396
0040202C	73DD1300	MFC42.#3346
00402030	73DD5126	MFC42.#5300
00402034	73E2F314	MFC42.#5302
00402038	73E68F4B	MFC42.#2725
0040203C	73DD1265	MFC42.#4079
00402040	73DD2CEA	MFC42.#4698
00402044	73DD121E	MFC42.#5307
00402048	73DD1300	MFC42.#5300

IAT Start

Address	Value	Comment
004021BC	77C30840	msvcrt._setmbcp
004021C0	00000000	
004021C4	77D4E2AE	USER32.SendMessageA
004021C8	77D4E7B8	USER32.GetSystemMenu
004021CC	77D5716C	USER32.AppendMenuA
004021D0	77D4C48A	USER32.IsIconic
004021D4	77D601EF	USER32.DrawIcon
004021D8	77D4C4D4	USER32.EnableWindow
004021DC	77D521AE	USER32.LoadIconA
004021E0	77D4B556	USER32.GetClientRect
004021E4	77D48F75	USER32.GetSystemMetrics
004021E8	00000000	
004021EC	00000000	
004021F0	00401000	unpackme.00401000
004021F4	004021F8	unpackme.004021F8
004021F8	00000111	
004021FC	00000000	
00402200	0000E146	

IAT End

IAT and then have the dump:

OllyDump - unpackme.exe

Start Address: 400000 Size: 34000 **Dump**

Entry Point: 31000 -> Modify: 31022 **Get EIP as OEP** **Cancel**

Base of Code: 12000 Base of Data: 2000

☒ **Fix Raw Size & Offset of Dump Image**

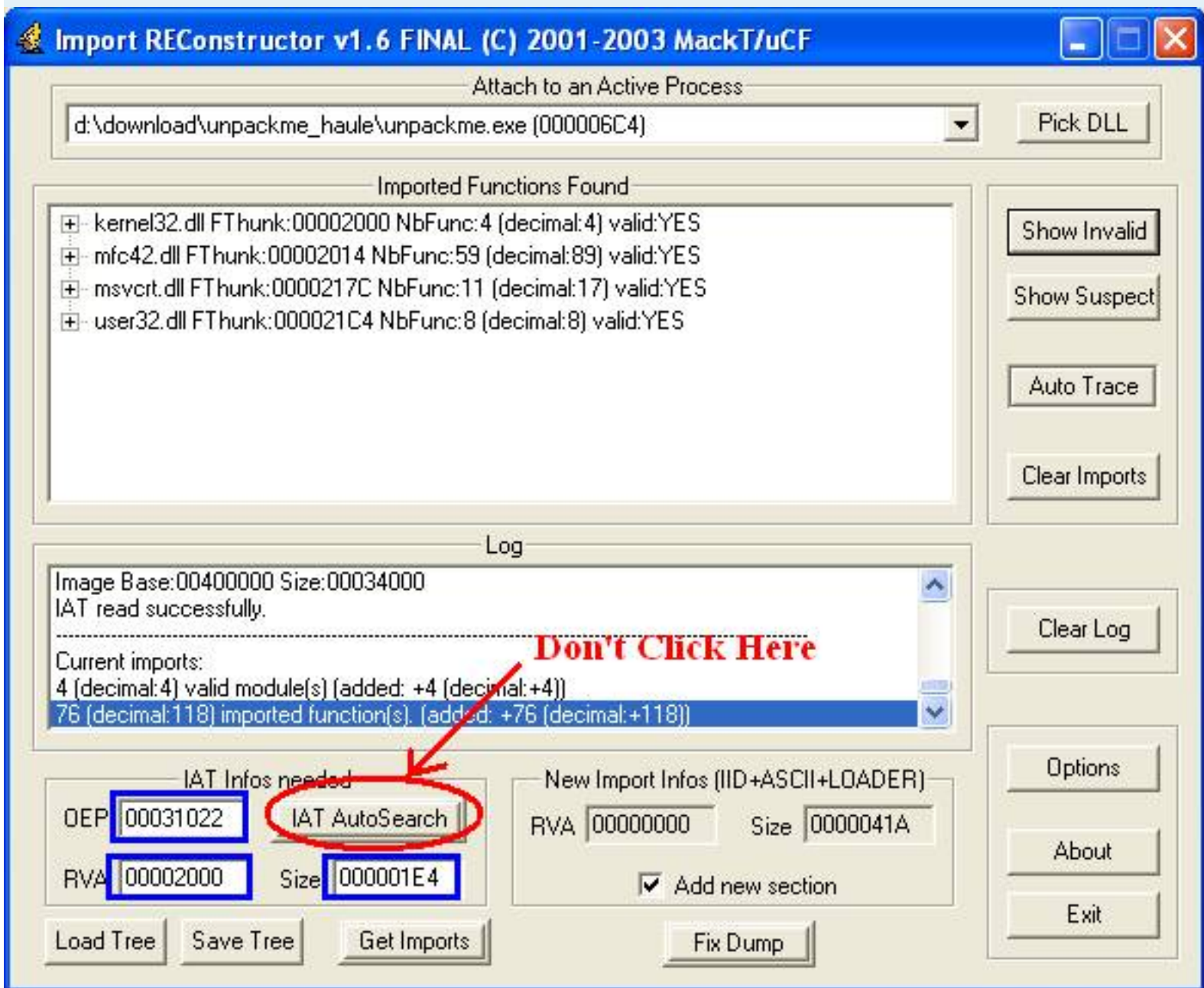
Sect...	Virtual Size	Virtual Off...	Raw Size	Raw Offset	Charactaris..
.edata	00000DCC	00001000	00000DCC	00001000	E0000020
.edata	00000A16	00002000	00000A16	00002000	40000040
.edata	00000188	00003000	00000188	00003000	C0000040
.edata	0000A4EC	00004000	0000A4EC	00004000	C0000040
.edata	00003000	0000F000	00003000	0000F000	C0000040
.ie^t	00012000	00012000	00012000	00012000	E0000020
.text	00001000	00024000	00001000	00024000	C0000040

☐ **Rebuild Import**

☒ **Method1** : Search JMP[API] | CALL[API] in memory image

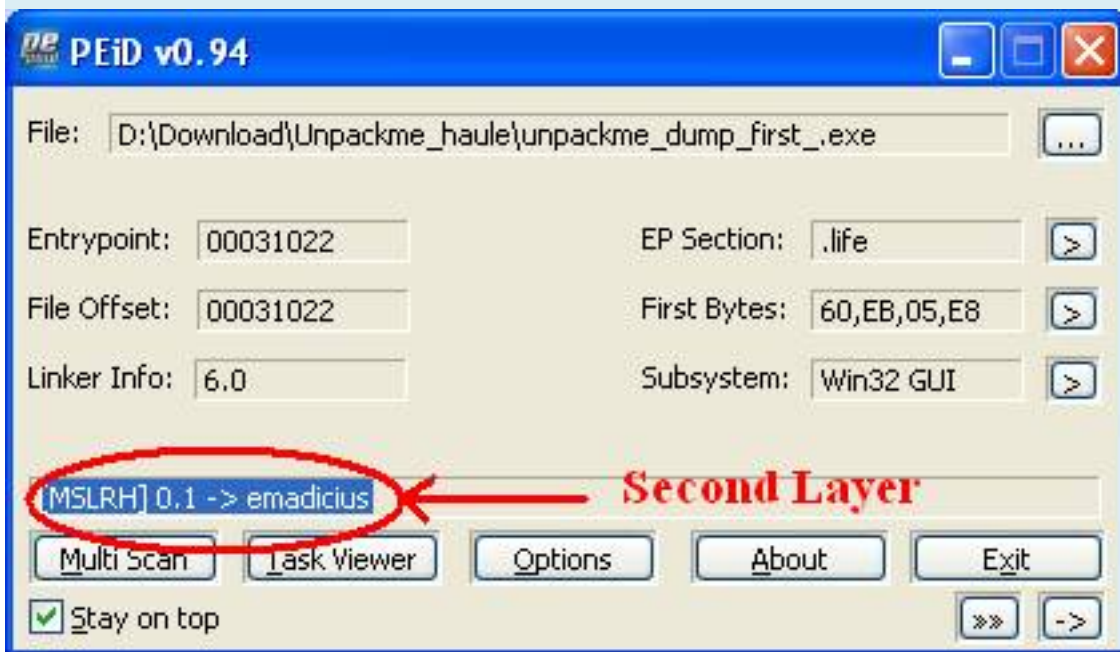
☐ **Method2** : Search DLL & API name string in dumped file

No IAT click **Auto Search** which to enter the number:



Fix dump. Save the file **unpackme_dump_first_.exe**

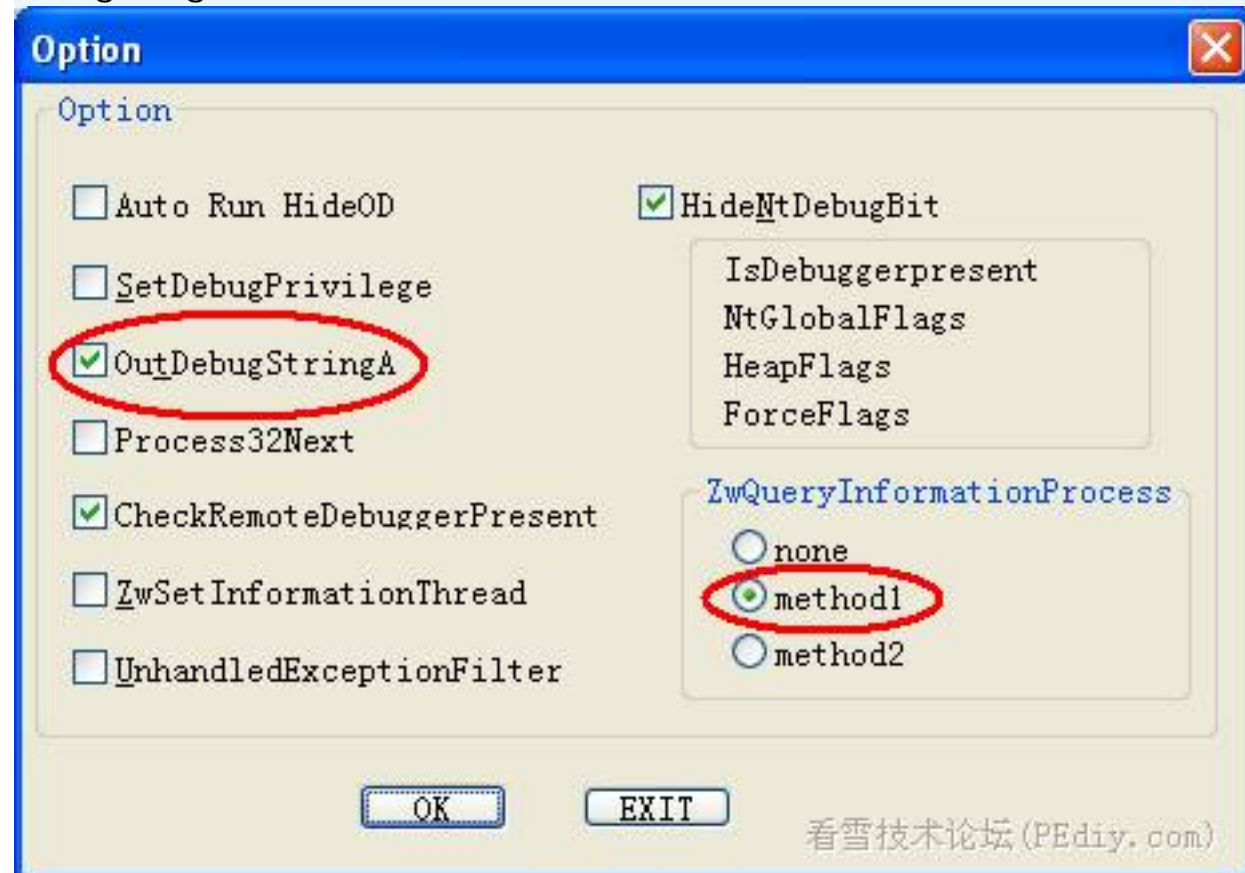
Clearly layer 2 is **MSLRH**:



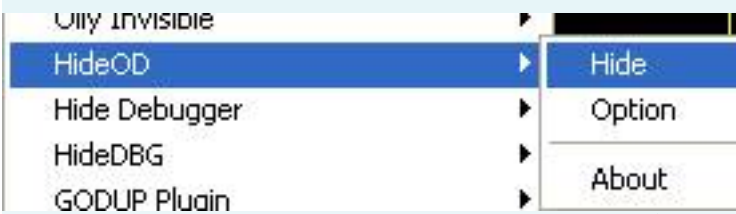
Load dump file just to fix (or do you do now to **remove** the layer 1 **BPOA** in the other **4 bytes** memory and save the file with ImportREC Tree)

This month is mainly used **OutputDebugStringA**, **IsDebuggerPresent** **ZwQueryInformationProcess** to Detect and the Debugger.

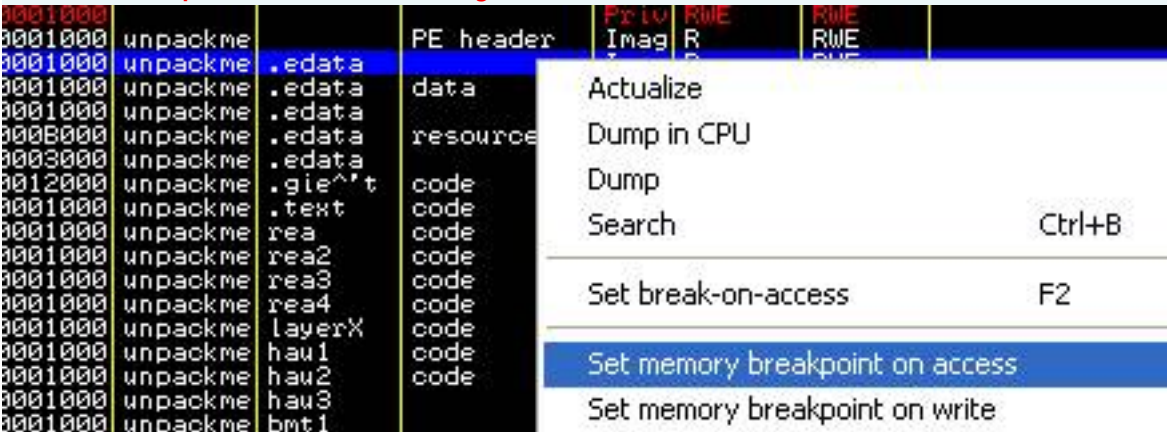
Using Plugin **HideOD**:



Enable it:



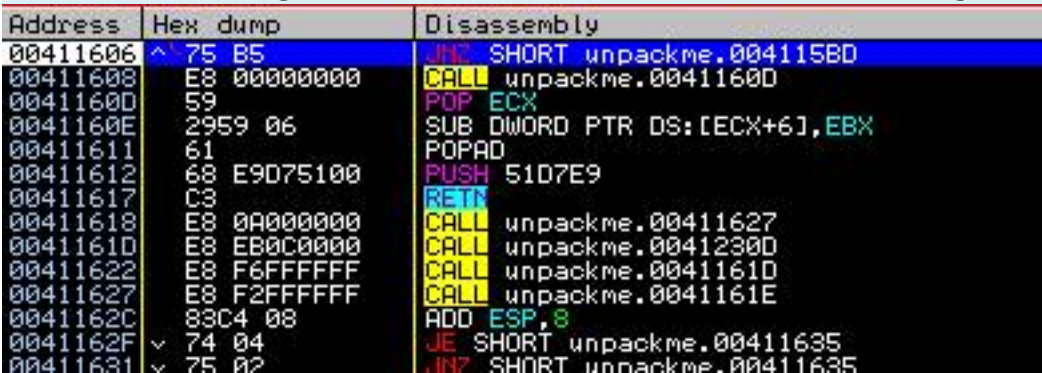
Set breakpoint on memory access (MBOA) in the code section (a guess: D):



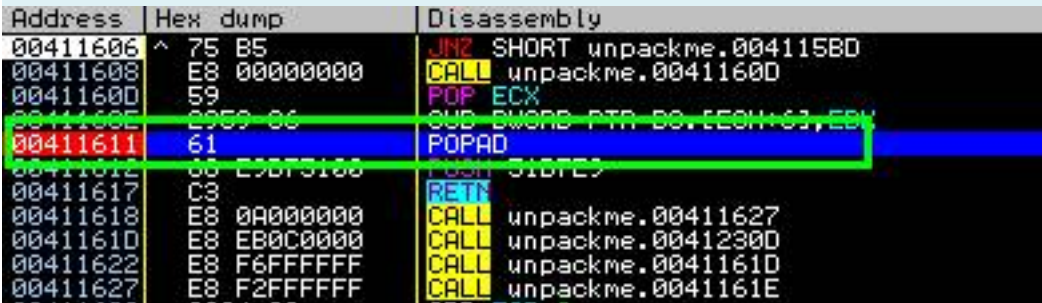
The F9, break:



Trace F7 through JMP command until far, 1 JNZ single command:



For that function under POPAD, set it at BP:



Delete the MBOA. F9 exit loop and Bread:

Address	Hex dump	Disassembly	Comment
00411606	75 B5	JNZ SHORT unpackme.004115BD	
00411608	E8 00000000	CALL unpackme.0041160D	
0041160D	59	POP ECX	
0041160E	2959 06	SUB DWORD PTR DS:[ECX+6],EBX	
00411611	61	POPAD	
00411612	68 701B4000	PUSH unpackme.00401B70	
00411617	C3	RETN	
00411618	E8 0A000000	CALL unpackme.00411627	
0041161D	E8 EB0C0000	CALL unpackme.0041230D	
00411622	E8 F6FFFFFF	CALL unpackme.0041161D	
00411627	E8 F2FFFFFF	CALL unpackme.0041161E	
0041162C	83C4 08	ADD ESP,8	
0041162F	74 04	JE SHORT unpackme.00411635	
00411631	75 02	JNZ SHORT unpackme.00411635	
00411632	EB 02	JMP SHORT unpackme.00411637	

**Don't have
Stolen Bytes**

**Jump to OEP
=401B70**

After POPAD do have **Stolen Bytes**, about 1 PUSH xxxxxx command and RETN equivalent JMP xxxxxx order, here is the **OEP = 401B70**.

Trace it to:

Address	Hex dump	Disassembly	Comment
00401B70	55	PUSH EBP	
00401B71	8BEC	MOV EBP,ESP	
00401B73	6A FF	PUSH -1	
00401B75	68 08254000	PUSH unpackme.00402508	
00401B7A	68 F61C4000	PUSH <JMP.&msvort._except_handler3>	
00401B7F	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	

<-- Real OEP (Final Layer)

IAT italy chang at first, i need to find even more should be used ImportREC ko ko also stars.

Address	Value	Comment
00402000	7C81EE79	kernel32.lstrcmpA
00402004	7C80C6E0	kernel32.lstrlenA
00402008	7C80B529	kernel32.GetModuleHandleA
0040200C	7C801EEE	kernel32.GetStartupInfoA
00402010	00000000	
00402014	73DE224E	mfc42.#5731
00402018	73E69FED	mfc42.#3922
0040201C	73DE1CF0	mfc42.#1089
00402020	73DE97A0	mfc42.#2512
00402024	73DE1273	mfc42.#5199
00402028	73DE0CC9	mfc42.#2396
0040202C	73DD1300	mfc42.#3346
00402030	73DD5126	mfc42.#5300
00402034	73E2F314	mfc42.#5302
00402038	73E68F4B	mfc42.#2725
0040203C	73DD1265	mfc42.#4079
00402040	73DD2CF0	mfc42.#4688

**Like as
First Layer**

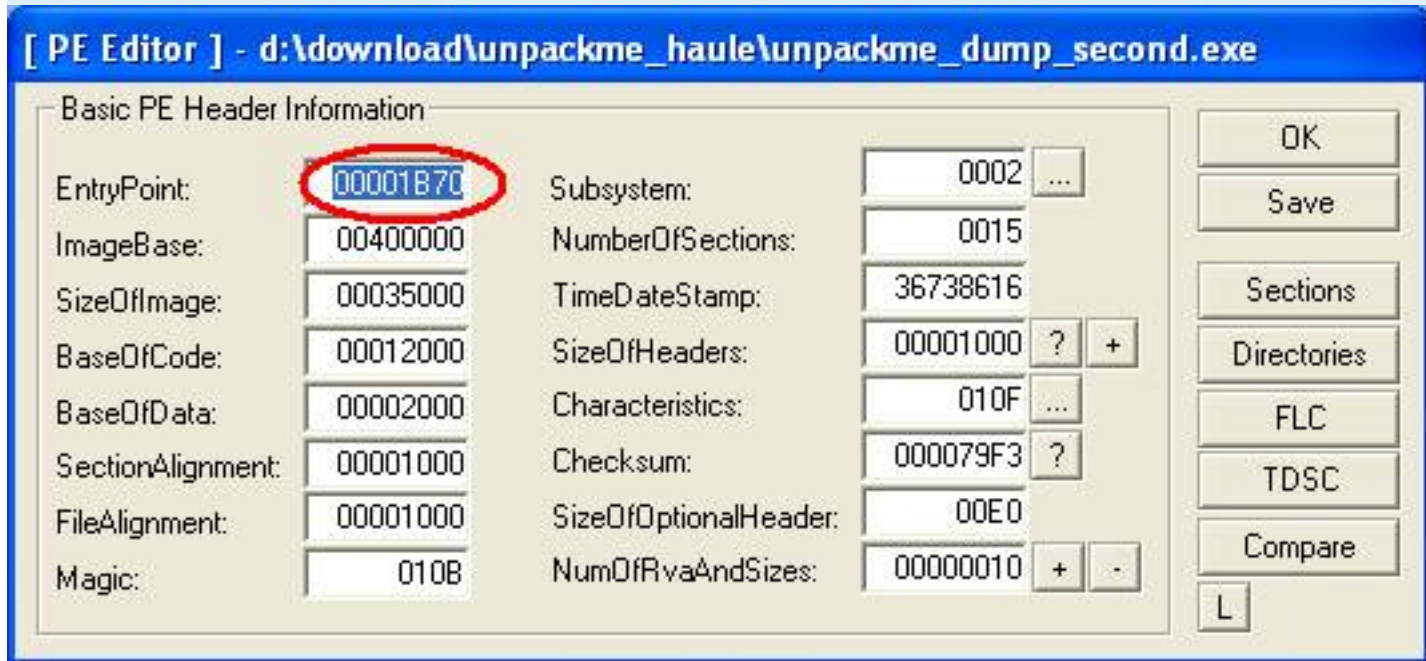
Full dump it in PETools (by LordPE and OllyDump been Detect):



Now you wonder, do not fix hả IAT Choi? OEP has not loaded into the dump? Hong Where do you àh.

1, IAT as old as the dump, it fly by đi.

2, because when you dump that Packer was fully unpack the source code, just fix EP by OEP there is more complete. How to: open **LordPE -> PE Editor**, the fix:



Save. Running lickerish.

Collapse file, use CFF Explorer, delete all the section, until only:

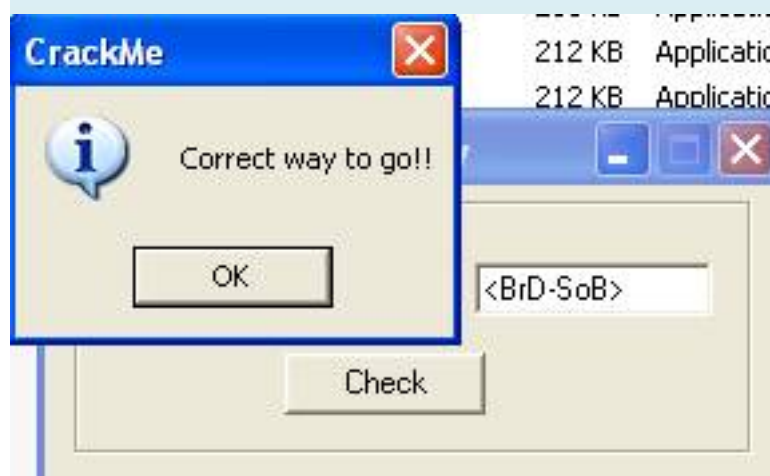
Byte[8]	Dword	Dword	Dword	Dword
.edata	00001000	00001000	00000DCC	00001000
.edata	00001000	00002000	00000A16	00002000
.edata	00001000	00003000	00000188	00003000
.edata	00020000	00004000	0000A4EC	00004000
.text	00010000	00024000	00001000	0000F000
.mact	00001000	00034000	00001000	00010000

Always remember revised **Code Of Base** properly, to support the Crack:

AddressOfEntryPoint	00000108	Dword	00001B70
BaseOfCode	0000010C	Dword	00001000
BaseOfData	00000110	Dword	00002000
ImageBase	00000114	Dword	00400000

Then use LordPE rebuild.

Cracking: a serial **<BrD-SoB>** (is it as entertaining, is not tricky home)



DONE!
Enjoy It!

UNPACKING wrapper USED BY GAMEHOUSE.COM

tlandn

Target: Sudoku (GameHouse)

Tools: OllyDbg

Then I have just read the tut on unpacking ColdIce wrapper used by GameHouse. Or to see and modify slightly for your reference J

D) Unpacking:

We start. Load program to OllyDbg. We here.

00502000	68 80205000	push	00502080	
00502005	FF15 74265000	call	[<&sudoku.mainDLLInit>]	sudoku
0050200B	0000	add	[eax], al	
0050200D	0000	add	[eax], al	

Press F9. A Nag appear.



Now in OllyDbg press Alt-M. Memory window appear. Select section. Text must click the mouse to select the image:

00350000	00003000				
00360000	00001000				
00370000	00001000				
00380000	00005000				
00390000	00002000				
003A0000	00002000				
003B0000	00002000				
003C0000	0000C000				
003D0000	00008000				
003E0000	00005000				
003F0000	00002000				
00400000	00001000	GHSudoku		PE he	
00401000	000A9000	GHSudoku	.text	code	
004AA000	00018000	GHSudoku	.rdata	code	

Set memory breakpoint on access

Set memory breakpoint on write

Set access

Allocate Memory

Free Memory

Zero Memory

Dump Memory-Area

Copy to clipboard

Sort by

Appearance

Now in the window NAG. Click the "Try Now."

Enter License Code

Buy Now

Trial time left: 120

Try Now

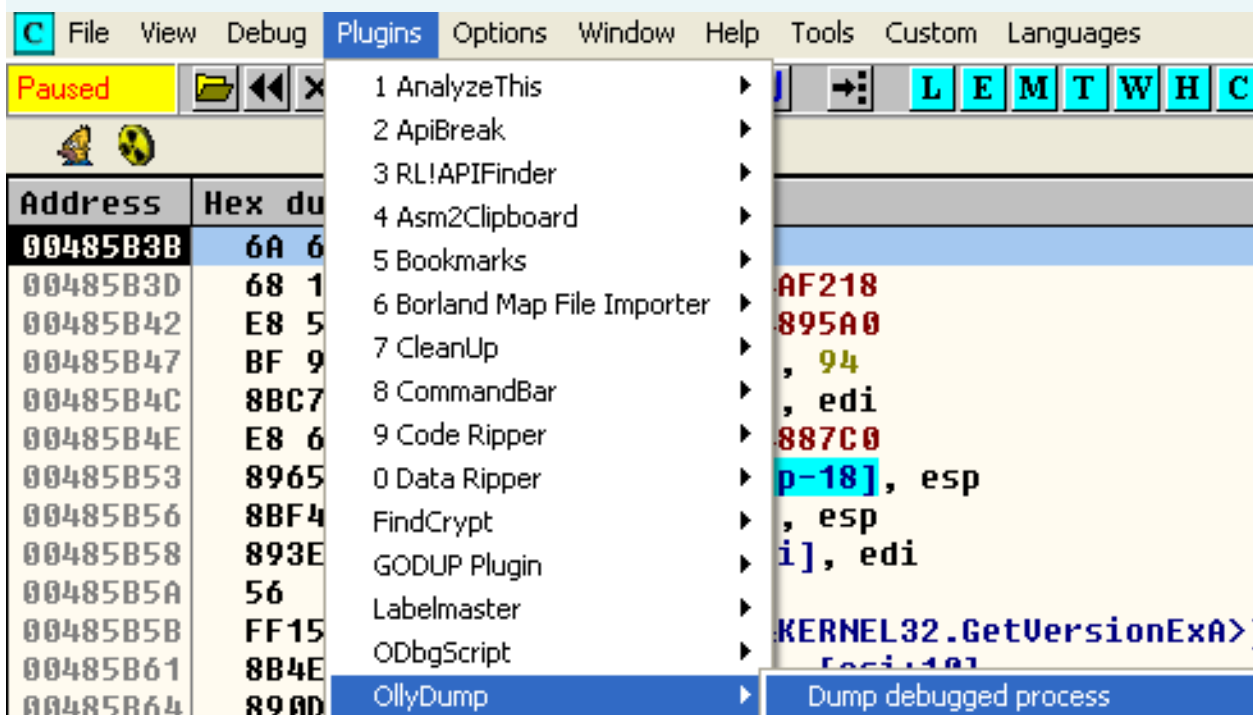
OllyDbg will break in 00485B3B

Address	Hex dump	Disassembly	Comment
00485B3B	6A 60	push 60	<- OEP
00485B3D	68 18F24A00	push 004AF218	
00485B42	E8 593A0000	call 004895A0	
00485B47	BF 94000000	mov edi, 94	
00485B4C	8BC7	mov eax, edi	

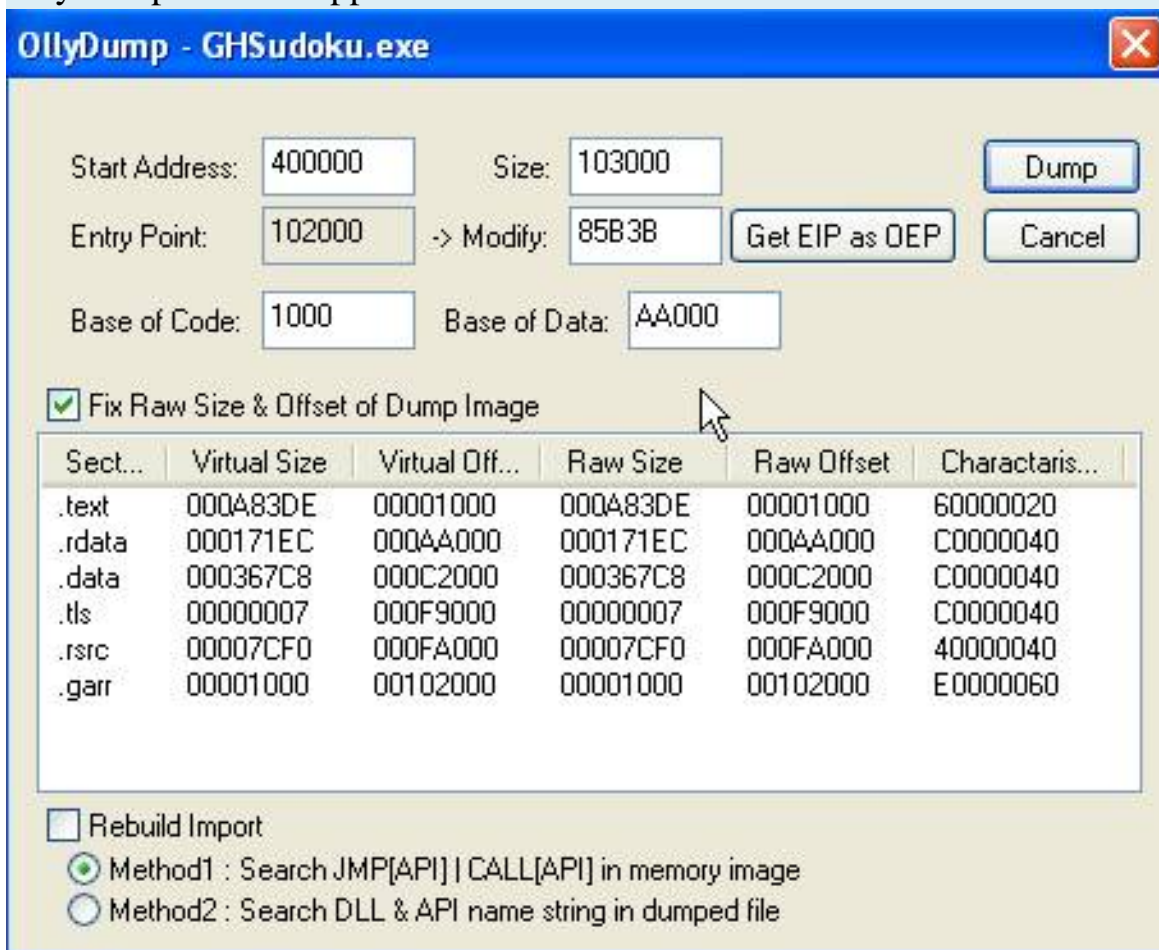
This is the OEP of the program (after unwrap). Looking through the window to see our record contains EDI 00485B3B (OEP of us, you remember it to write scripts in the following).

Registers (FPU)		
EAX	0050200B	GHSudoku
ECX	7C90FB71	ntdll
EDX	00000002	
EBX	7FFDF000	
ESP	0012FFC0	
EBP	0012FFB8	
ESI	FFFFFFFF	
EDI	00485B3B	GHSudoku

Now we will dump and fix IAT Imprec use. Choose the type:



OllyDump window appear.

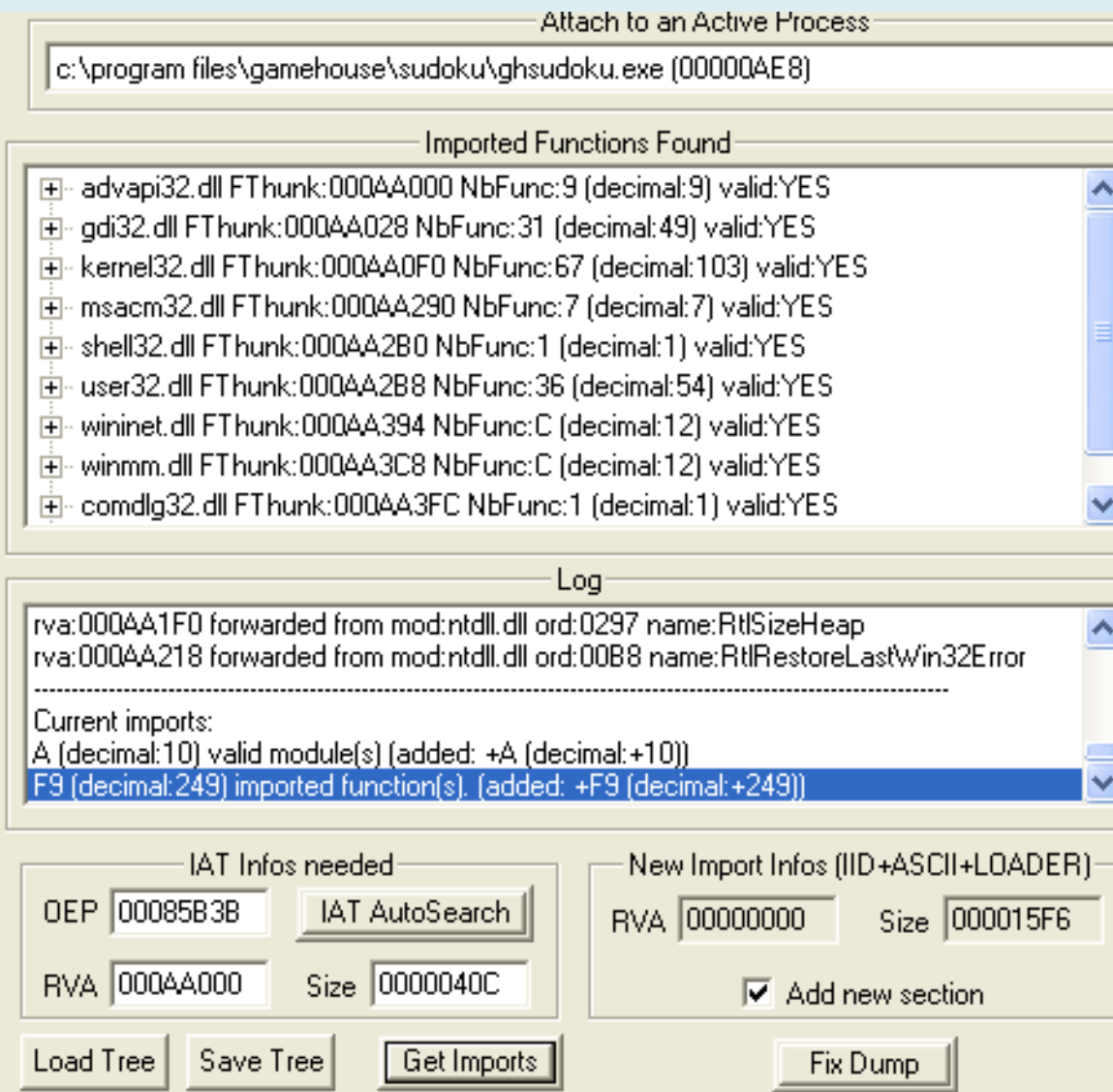


Select as the image. Click "dump". Name the file a.exe.

Now run Imprec.

Enter OEP: 85B3B (00485B3B - 00400000)

Click "IAT AutoSearch." Then, "Get Imports".



Click "Fix dump" file selected a.exe. We are a_.exe file. Test this file. Good!.

II) Script Writing:

You also remember when we break in the OEP EDI contains the address OEP. Therefore we deduce wrapper program will jump to OEP our **JMP** using **EDI** (May packer often jump to OEP using JMP thanh_ghi).

We will try to deduce our Correct. Using OllyDbg open programs.

00502000	68 80205000	push	00502080	
00502005	FF15 74265000	call	[<&sudoku.mainDLLInit>]	51

Press F7 2. We here:

180079F0	55	push	ebp	
180079F1	8BEC	mov	ebp, esp	
180079F3	81EC 7C020000	sub	esp, 27C	
180079F9	57	push	edi	
180079FA	C745 F0 00000000	mov	dword ptr [ebp-10], 0	
180079A1	C745 9C 00000000	mov	dword ptr [ebp-64], 0	
180079A8	EB 09	jmp	short 180079A13	

Now we find command JMP EDI. Click your mouse to select the image:

180079F0	55	Copy	
180079F1	8BEC	Binary	
180079F3	81EC 7C020000	Assemble	Space
180079F9	57	Label	:
180079FA	C745 F0 000000	Comment	;
18007A01	C745 9C 000000	Breakpoint	
18007A08	EB 09	Run trace	
18007A0A	8B45 9C		
18007A0D	83C0 01		
18007A10	8945 9C	Go to	
18007A13	837D 9C 03	Follow in Dump	
18007A17	7D 10	Search for	Name (label) in current module Ctrl+N
18007A19	8B4D 9C	Find references to	Name in all modules
18007A1C	C7048D F4D300	View	
18007A27	EB E1	Conv to executable	Command Ctrl+F

Enter:

Find command

☐ Entire block

Find Cancel

Click "Find". We are:

18007DAF	8B7D 08	mov	edi, [ebp+8]
18007DB2	8BE5	mov	esp, ebp
18007DB4	58	pop	eax
18007DB5	58	pop	eax
18007DB6	FFE7	jmp	edi
18007DB8	6A 00	push	0
18007DBA	FF15 C8900018	call	[<&KERNEL32.ExitProcess>]
18007DC0	5F	pop	edi

Only 1 match. Press F2 to set a breakpoint in 18007DB6. Press F9 to run the program. After pressing the button "Try Now" in the NAG screen. We break in 18007DB6

18007DB4	58	pop	eax
18007DB5	58	pop	eax
18007DB6	- FFE7	jmp	edi
18007DB8	6A 00	push	0
18007DBA	FF15 C8900018	call	[<&KERNEL32.ExitProcess>]
18007DC0	5F	pop	edi

Press F8. We at J OEP

Address	Hex dump	Disassembly
00485B3B	6A 60	push 60
00485B3D	68 18F24A00	push 004AF218
00485B42	E8 593A0000	call 004895A0

With all the information we can write a generic script for all the games in GameHouse.

msgyn "You must checked all the exceptions to proceed, continue?"

```
STI / / F7
```

```
STI / / F7
```

```
find eip, # # FFE7 / / Search for JMP EDI (corresponding FFE7)
```

```
msg "Press' Try Now 'button appears when the Dialog"
```

```
go $ RESULT / / Run the program to place JMP EDI we found
```

```
Sto / / F8
```

```
CMT eip, "<- OEP"
```

```
msg "Bam! You're at the OEP, now dump'n'fix ImpRec with the IAT.
```

```
Locu/06"
```

```
ret
```

You can download gamehouse.com on the targer on trial.

Them. Happy happy. Hopefully you will learn something new through this tut.

Greetingz: All Reaonline.net Members, ColdIce, and you Locu/06.

tlandn

20-May-2006

TUT # 5: snd Unpackme - Armadillo 3.70a-Debug-Blocker



_Hiiii Although aged grape was 7 tut to the children of Armadillo, but I am old and not enough health to run a race with grapes and aged for you: (. Now I try to new tut second, what they do first but then the child is the tit Ngoi. Although the OEP, dump full, Fix IAT, what is done only for each Run unpack the files have not done it all:). The pile, if aged grapes that stand beside me now I sure he aged throttle too:).

_Sau Try the test with Target's aged in small given tut 1 and 2, I draw a number of articles as follows: stage find OEP seems most easily, just buckle and dump Fix IAT is hard. Hix I use the LordPE I still xài, full dump, but the fix IAT Cuc. When the full dump and fix the IAT is run it does not run. When I xài the aged in small, AC dump and fix IAT lickerish, run away alcohol. Only the children **PE Tools v1.5.600.2005** at the time was not. And the media always say in my home computer using OS XP Pro not have 1 or 2 Pack ráo anything, so when Olly's aged a little but when BP **ArmaDetach.dll** plug on the Access Memory, press Shif + F9 is considered as having such as break wind, not in OEP Break:). So I đành ngam ngui suffer as they put into this xó.

_Dang Read the tut 2 on the Target, see aged remember that you write. Reliance on skilled their Super Newbie, I do a circuit from the beginning to end. Everything Okie, but when run, they ặc not run:) but also errors in the load User32.dll children.

_Thoi Card and then say, I will write a tut for this target. Uncle do not try it can be reference or higher is done manually and then, seeing how I do it to ignore.

_Load To target and Olly Olly configured to handle the process as parents aged grapes said:

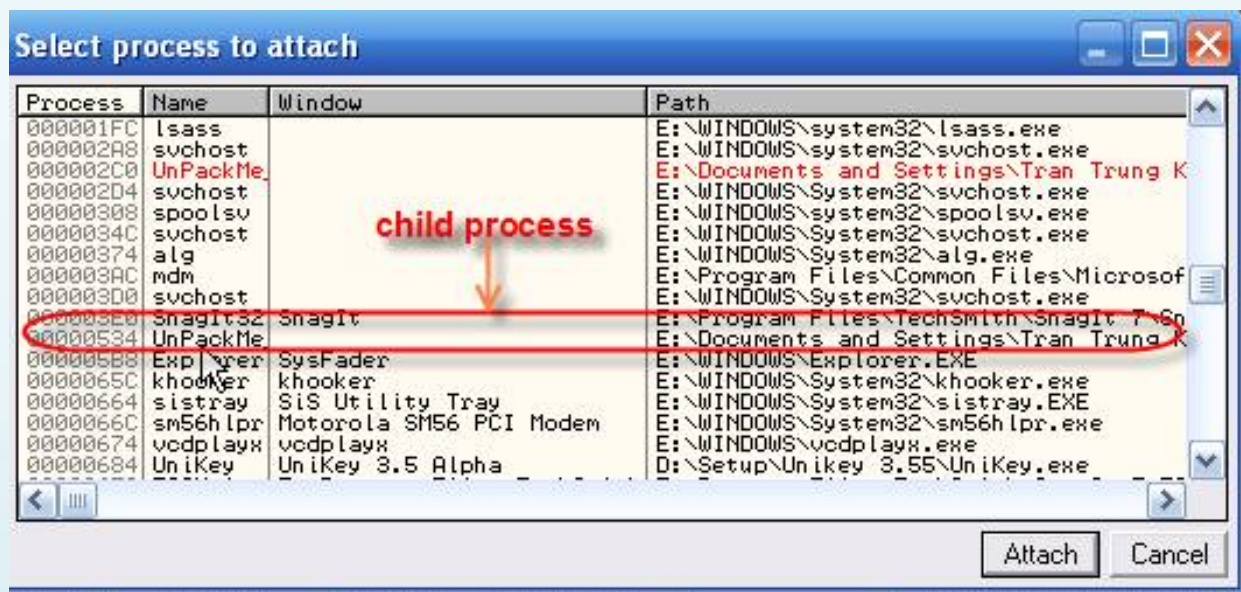
Address	Hex dump	Disassembly	Comment
0043AB19	55	MOV EBP, ESP	
0043AB1A	8BEC	MOV EBP, ESP	
0043AB1C	6A FF	PUSH -1	
0043AB1E	68 386A4500	PUSH UnPackMe.00456A38	
0043AB23	68 00A54300	PUSH UnPackMe.0043A500	
0043AB28	64:A1 000000	MOV EAX, DWORD PTR FS:[0]	SE handler installation
0043AB2E	50	PUSH EAX	
0043AB2F	64:8925 0000	MOV DWORD PTR FS:[0], ESP	
0043AB36	83EC 58	SUB ESP, 58	
0043AB39	53	PUSH EBX	
0043AB3A	56	PUSH ESI	
0043AB3B	57	PUSH EDI	
0043AB3C	8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
0043AB3F	FF15 4C114500	CALL DWORD PTR DS:[4C114500]	kernel32.GetVersion

_De Flush public, I always used to Detach Script for it quickly. But doctors want to do the Manual comfortable no matter what. After running script, I have been as follows:

Address	Hex dump	Disassembly	Comment
0042ADD7	50	PUSH EAX	
0042ADD8	E8 218DA877	CALL kernel32.DebugActiveProcessStop	
0042ADD0	90	NOP	
0042ADDE	? 0060 33	ADD BYTE PTR DS:[EAX+33], AH	
0042ADE1	? C075 02 EB	SAL BYTE PTR SS:[EBP+2], 0EB	Shift constant out of range
0042ADE5	? 15 EB3C075	ADC EAX, 75C033EB	

Registers (FPU)
 EAX 00000001
 ECX 0012DA38
 EDX 7FFE0304
 EBX 7FFDF000
 ESP 0012DAB4
 EBP 0012F5A0
 ESI 00002710
 EDI 0012F03C UNICODE "kernel32.dll"

_Kha Kha, so the joker is considered as completed. Process View of the month so that the children know which street Attach. I have been as follows:



_Okie, Hours Olly a more open and more children to Attach thẳng. Configuring the aged grape as Olly said in the tut. Then press F9, F12, and we Patch bytes are as follows:

Address	Hex dump	Disassembly	Comment
0043AB19	55	PUSH EBP	
0043AB1A	8BEC	MOV EBP,ESP	
0043AB1C	6A FF	PUSH -1	
0043AB1E	68 386A4500	PUSH UnPackMe.00456A38	
0043AB23	68 00A54300	PUSH UnPackMe.0043A500	SE handler installation
0043AB28	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
0043AB2E	50	PUSH EAX	
0043AB2F	64:8925 000000	MOV DWORD PTR FS:[0],ESP	
0043AB36	83EC 58	SUB ESP,58	
0043AB39	53	PUSH EBX	
0043AB3A	56	PUSH ESI	
0043AB3B	57	PUSH EDI	
0043AB3C	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0043AB3F	FF15 4C114500	CALL DWORD PTR DS:[4C114500]	kernel32.GetVersion
0043AB45	3302	XOR EAX,EAX	

_Tiep By pressing Alt + M to open the Memory windows, set a BP in the section on access text:

00404000	00002000	UnPackMe	.text
00406000	00001000	UnPackMe	.data
00407000	00009000	UnPackMe	.bss
00410000	00001000	UnPackMe	IMPORT
00411000	00003000	UnPackMe	.text
00441000	00010000	UnPackMe	.adata
00451000	00020000	UnPackMe	.data1
00471000	00030000	UnPackMe	.pdata
004A1000	00002000	UnPackMe	.rsrc
004B0000	00103000		
005C0000	00090000		
008C0000	00001000		
008E0000	00001000		
008F0000	00001000		
00900000	00001000		
77C70000	00001000	GD132	

_Nhan Shift + F9, you Break in OEP. Hix Olly here that my xài thẳng Plugin ArmaDetach.dll break is regarded as always:).

Address	Hex dump	Disassembly	Comment
00404000	9B	WAIT	
00404001	DBE3	FINIT	
00404003	9B	WAIT	
00404004	DBE2	FCLEX	
00404006	D92D 00604000	FLDCW WORD PTR DS:[406000]	
0040400C	55	PUSH EBP	
0040400D	89E5	MOV EBP,ESP	
0040400F	E8 01040000	CALL UnPackMe.00404415	
00404014	68 00000000	PUSH 0	
00404019	FF15 F4114000	CALL DWORD PTR DS:[4011F4]	
0040401F	A3 07F04000	MOV DWORD PTR DS:[40F007],EAX	
00404024	60	PUSHAD	
00404025	8925 0BF04000	MOV DWORD PTR DS:[40F00B],ESP	
0040402B	E9 30000000	JMP UnPackMe.00404060	
00404030	8B25 0BF04000	MOV ESP,DWORD PTR DS:[40F00B]	
00404036	61	POPAD	
00404037	E8 19090000	CALL UnPackMe.00404955	
0040403C	E8 6D040000	CALL UnPackMe.004044AE	
00404041	89EC	MOV ESP,EBP	
00404043	5D	POP EBP	
00404044	FF35 04F14000	PUSH DWORD PTR DS:[40F104]	
0040404A	FF15 EC114000	CALL DWORD PTR DS:[4011EC]	
00404050	9B	WAIT	
00404051	DBE2	FCLEX	
00404053	D92D 00604000	FLDCW WORD PTR DS:[406000]	
00404059	C3	RET	
0040405A	0000	ADD BYTE PTR DS:[EAX],AL	
0040405C	0000	ADD BYTE PTR DS:[EAX],AL	
0040405E	0000	ADD BYTE PTR DS:[EAX],AL	
00404060	68 00000000	PUSH 0	
00404065	B8 72604000	MOV EAX,UnPackMe.00406072	ASCII "Coded by Teddy Roge
0040406A	89C2	MOV EDX,EAX	
0040406C	52	PUSH EDX	
0040406D	B8 4A604000	MOV EAX,UnPackMe.0040604A	ASCII "If you unpack it wri

_Hii If you follow the tut aged grapes for Magic Jump also. But I do then dump and run fix ok but they do not chay.Ly is in place Fix IAT, because when ImportREC to implement IAT Auto Search, it will address omission of the IAT. So I made the classic way as follows: at the OEP, press Ctrl + B and enter the **FF 15**, why is **FF 15** that is not **FF 25**. Please Dear with you that I have but then Search no. You will find the first code as follows:

00404019 FF15 F4114000 CALL DWORD PTR DS: [4011F4]

_Nhan Ctrl + L to continue to search all the code similar to that. Fortunately in this target it is less, but more sure I am always filled. During the search, note value of the largest and smallest **max min** in **DS**: **[4011F4]**

_Sau A search I have been the final results as follows:

00404EF2 FF15 9C114000 CALL DWORD PTR DS: [40119C]; USER32.CallNextHookEx

00404019 FF15 F4114000 CALL DWORD PTR DS: [4011F4]

0040404A FF15 EC114000 CALL DWORD PTR DS: [4011EC]

00404326 FF15 14124000 CALL DWORD PTR DS: [401214]; kernel32.lstrcatA

00404453 FF15 08124000 CALL DWORD PTR DS: [401208]; kernel32.HeapCreate

00404584 FF15 0C124000 CALL DWORD PTR DS: [40120C]; kernel32.HeapDestroy

004045EF FF15 00124000 CALL DWORD PTR DS: [401200]; ntdll.RtlAllocateHeap

00404622 FF15 04124000 CALL DWORD PTR DS: [401204]; kernel32.HeapCompact

004046DD FF15 10124000 CALL DWORD PTR DS: [401210]; ntdll.RtlFreeHeap

00404C17 FF15 B0114000 CALL DWORD PTR DS: [4011B0]; USER32.SystemParametersInfoA

00404C66 FF15 A0114000 CALL DWORD PTR DS: [4011A0]; USER32.GetDesktopWindow

```

00404C7F FF15 A4114000 CALL DWORD PTR DS: [4011A4]; USER32.GetWindowRect
00404EF2 FF15 9C114000 CALL DWORD PTR DS: [40119C]; USER32.CallNextHookEx
00404F89 FF15 F0114000 CALL DWORD PTR DS: [4011F0]; kernel32.GetCurrentThreadId
00404FA4 FF15 AC114000 CALL DWORD PTR DS: [4011AC]; USER32.SetWindowsHookExA
00404FC9 FF15 A8114000 CALL DWORD PTR DS: [4011A8]
00404FDB FF15 B4114000 CALL DWORD PTR DS: [4011B4]; USER32.UnhookWindowsHookEx
00405614 FF15 FC114000 CALL DWORD PTR DS: [4011FC]; kernel32.GlobalFree
004057CE FF15 F8114000 CALL DWORD PTR DS: [4011F8]

```

_Keke If you do find a way Magic Jump I do not know your computer is still on but when I finished Fix IAT, they run as they error during load User32. And in the process fix IAT I have not seen any in the User32. Hix but if I do say in how the right is the real User32. I notice not so little any wrong:).

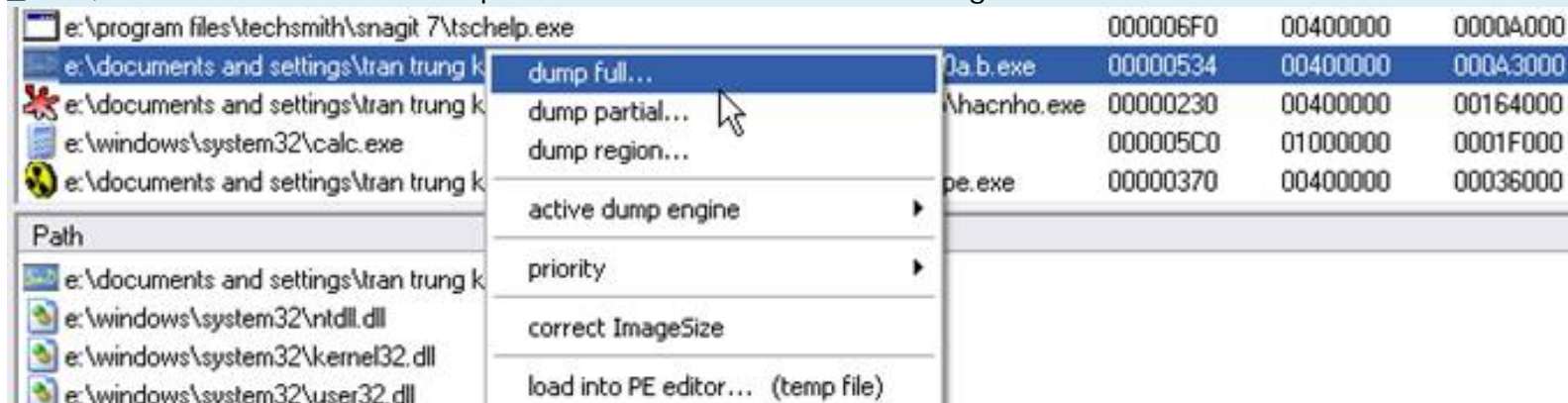
_Sau When the results as above. I will be as follows:

OEP = value that we found in the = 404000 - 400000 = 4000

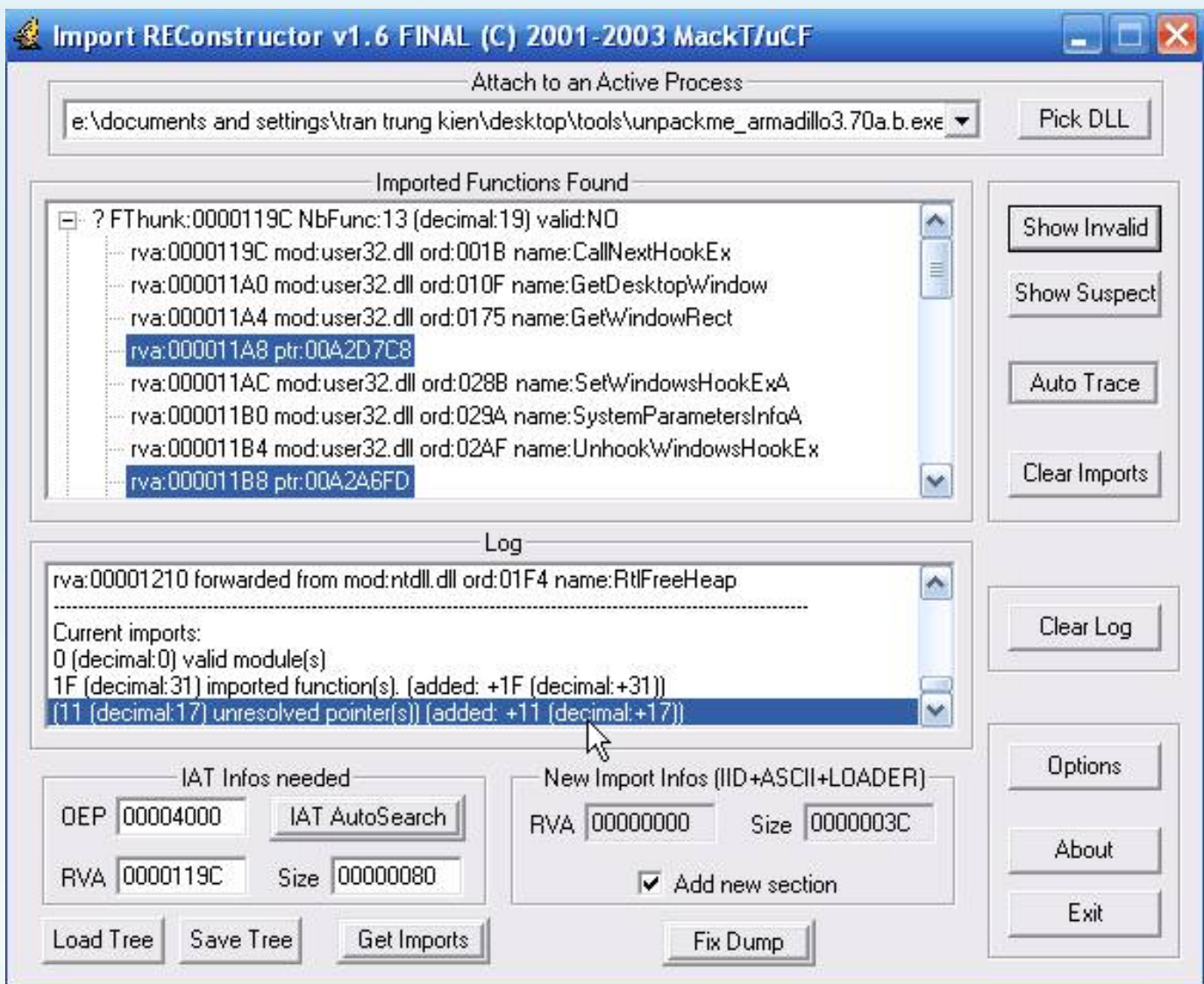
RVA = min - **40119C** = 400,000 - 400,000 = **119C**

Size = max - min = **401214 - 40119C = 78** (rounded to 80)

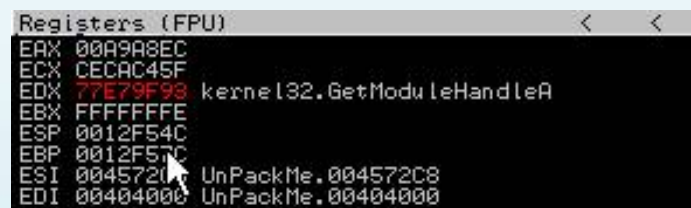
_Okie, Next LordPE used to dump full. Remember to choose the right Process nhé.



_Tiep As we open up ImportREC. Selecting the right process and fill in the information exactly as what is found above. Do not click Search IAT Auto Imports Get the press. I will be as follows:



_ According to what we have found above. We will just go back to Fix the do.Dau address is: 00404019 FF15 F4114000 CALL DWORD PTR DS: [4011F4]. At this location, click to select the New Origin here. Sau press F7 to this function, according to press F7 to trace into this function. Results will be as follows:

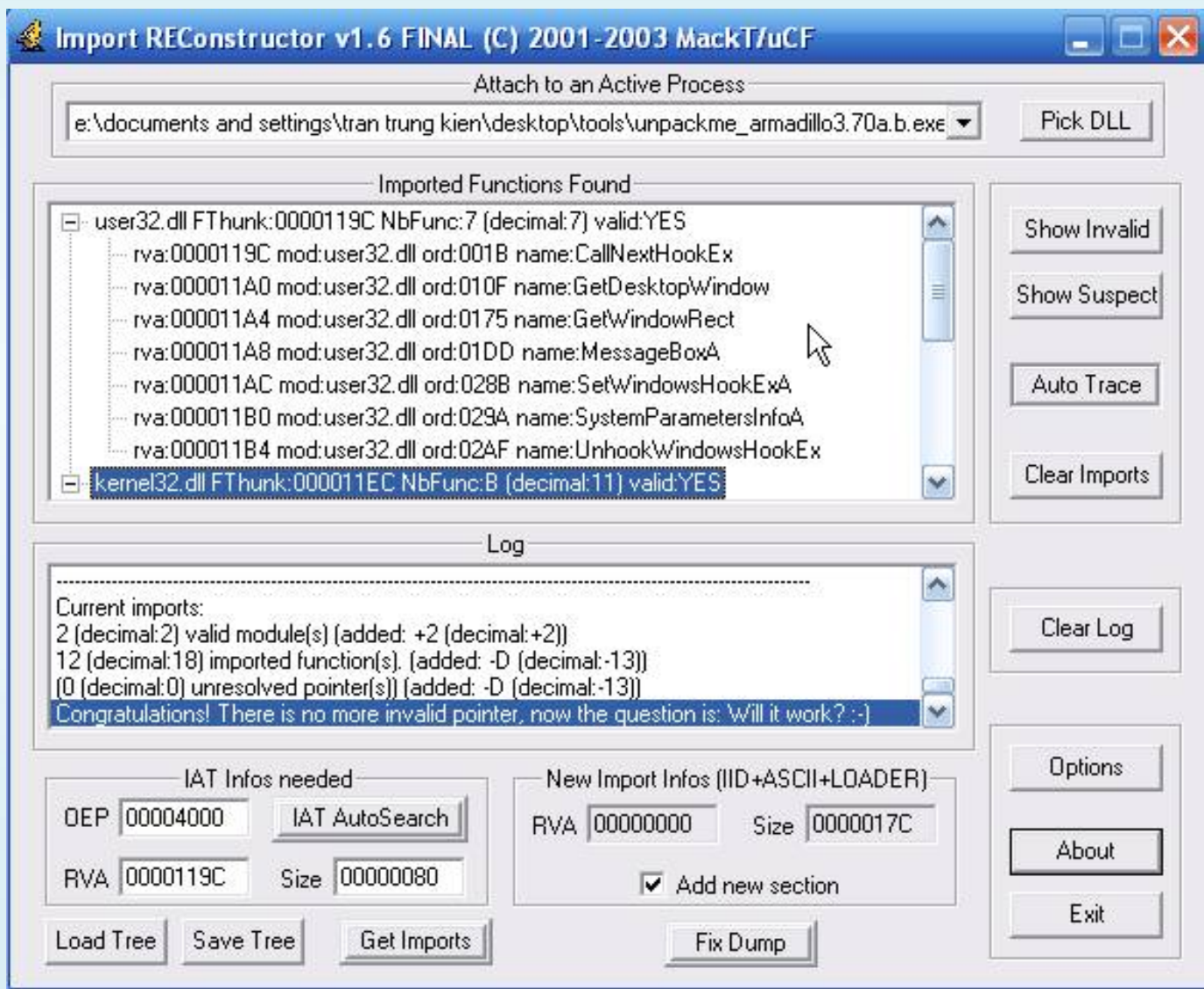


_Vay In ImportRec will Fix is **GetModuleHandleA**. Next is 0040404A FF15 EC114000 CALL DWORD PTR DS: [4011EC], we have a **ExitProcess**. Continue will be as follows:

00404FC9 FF15 A8114000 CALL DWORD PTR DS: [4011A8]; **MessageBoxA**

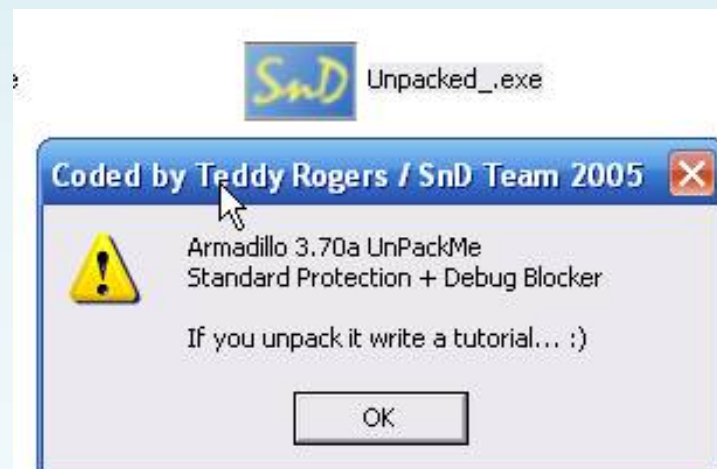
004057CE FF15 F8114000 CALL DWORD PTR DS: [4011F8]; **GlobalAlloc**

Invalid _Con what we Think Cut off, not to release them hiiii.



_Buoc Final step and is meeting Fix dump. Hiii not know when to fix them will finish what is known as nhi. Chi Fix the report is found ImportREC Success. Success does not know is not true:).

_Hix Hix, they try to run one, prince falter. Ola enough, they run without a roar yet, but can. I think he does not run always run!



_Okie So it is Done nhé. I look back:)



- Finished - [September 27, 2005](#)

--+ +---==[**Greatz thanks to**]=---+ +--

- Thank to my family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHQCrker, the_Lighthouse, Hoadongnoi, Nini, L ight.phoenix, Merc ... all REA's members, HacNho, RongChauA, Deux, tlandn, dqtlN, CracksLatinos, ARTEAM all my friend, and you.

Special thanks to --+ +---==[]=---+ +--

coruso_trac, patmsvn, tm_tr v. .. v.. VSEC in all brothers.

>>>> If you have any suggestions, comments or corrections email me: [kienbigmummy \[at\] gmail.com](mailto:kienbigmummy@gmail.com)

REVERSE ENGINEERING ASSOCIATION

<http://www.reaonline.net>



+ + **CopyMemII** **Debugblocker** **IAT** **elimination** **+ Code** **Splicing +** **Nanomites**

Target : **UnpackMe (4.30 + Nanomites Armadillo)**

Crack Tool **1.** **OllyDBG by hacnho.**

2. LordPE Deluxe 1.4-by yoda

3.Import REConstructor 1.6 Final

4. ArmInline 0.71

Author : **Why Not Bar**

Party Web site by Hacnho 4 UnpackMe I clean meat packages. But I have to thắng UnpackME *Nanomites*. We unpack it home!

Target _Load to Olly

00447733	55	PUSH EBP	
00447734	8BEC	MOV EBP,ESP	
00447736	6A FF	PUSH -1	
00447738	68 88114700	PUSH UnpackMe.00471188	
0044773D	68 70744400	PUSH UnpackMe.00447470	SE handler installation
00447742	64:A1 000000	MOV EAX,DWORD PTR FS:[0]	
00447748	50	PUSH EAX	
00447749	64:8925 0000	MOV DWORD PTR FS:[0],ESP	
00447750	83EC 58	SUB ESP,58	
00447753	53	PUSH EBX	
00447754	56	PUSH ESI	
00447755	57	PUSH EDI	
00447756	8965 E8	MOV [LOCAL.6],ESP	
00447759	FF15 88014600	CALL NEAR DWORD PTR DS:[<&KERNEL32.GetU	kernel32.GetVersion
0044775F	33D2	XOR EDX,EDX	
00447761	8AD4	MOV DL,AH	
00447763	8915 14284700	MOV DWORD PTR DS:[4728141],EDX	

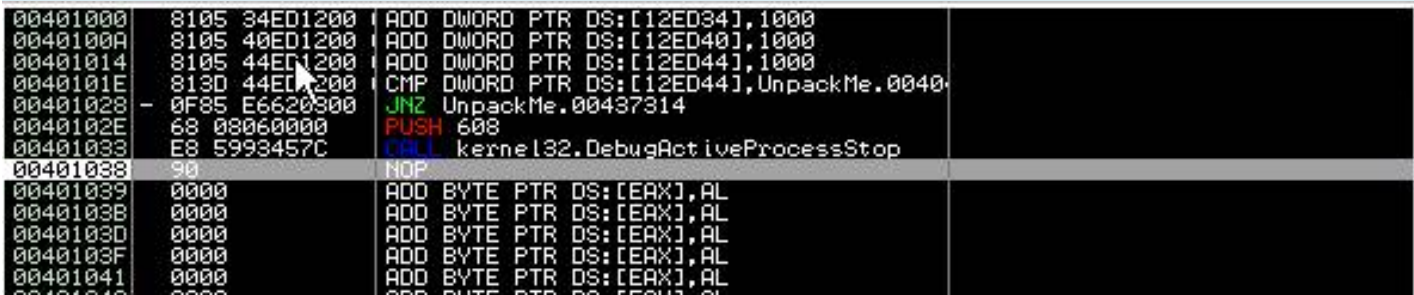
_Su To 0.92 OllyScript run script "DetachFarther_MethodRicardo_hipu_benina." There notification



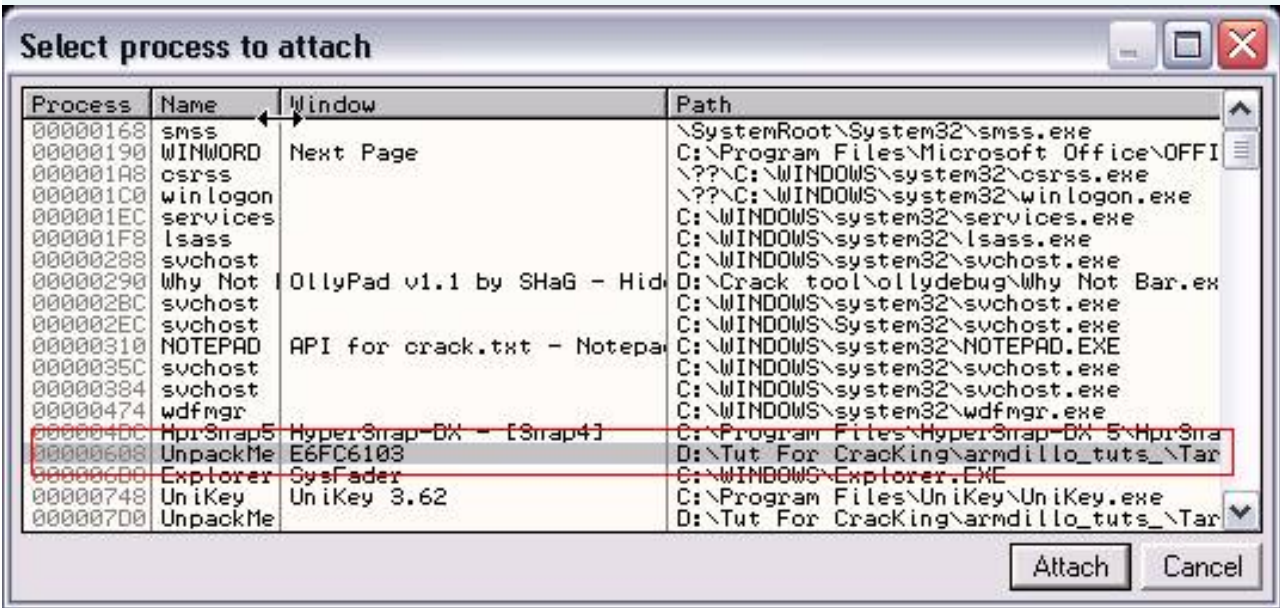
_Nhan OK to continue



558B _Ghi remember and click OK. Continue you click OK 3 times longer to complete the run script is here to



1 _Mo window and Olly Attach PID child



Attach _Nhan, F9, F12 and edit the 558B

00403468	55	PUSH EBP	
00403469	85EC	MOV EBP,ESP	
0040346B	83C4 F0	ADD ESP,-10	
0040346E	B8 38344000	MOV EAX,UnpackMe.00403438	
00403473	E8 94FEFFFF	CALL UnpackMe.0040330C	
00403478	33C0	XOR EAX,EAX	
0040347A	55	PUSH EBP	
0040347B	68 27354000	PUSH UnpackMe.00403527	
00403480	64:FF30	PUSH DWORD PTR FS:[EAX]	
00403483	64:8920	MOV DWORD PTR FS:[EAX],ESP	
00403486	E8 70FFFFF	CALL UnpackMe.00403408	JMP to COMCTL32.InitCommonContro
0040348B	CC	INT3	
0040348C	59	POP ECX	
0040348D	AD	LODS DWORD PTR DS:[ESI]	
0040348E	F2:66:	PREFIX REPNE:	Superfluous prefix
00403490	50	PUSH EAX	
00403491	B8 0A000000	MOV EAX,0A	

_Bay Time we find the address IAT. How to find IAT as follows: you point your mouse to command the first Call

00403473 E8 94FEFFFF CALL UnpackMe.0040330C

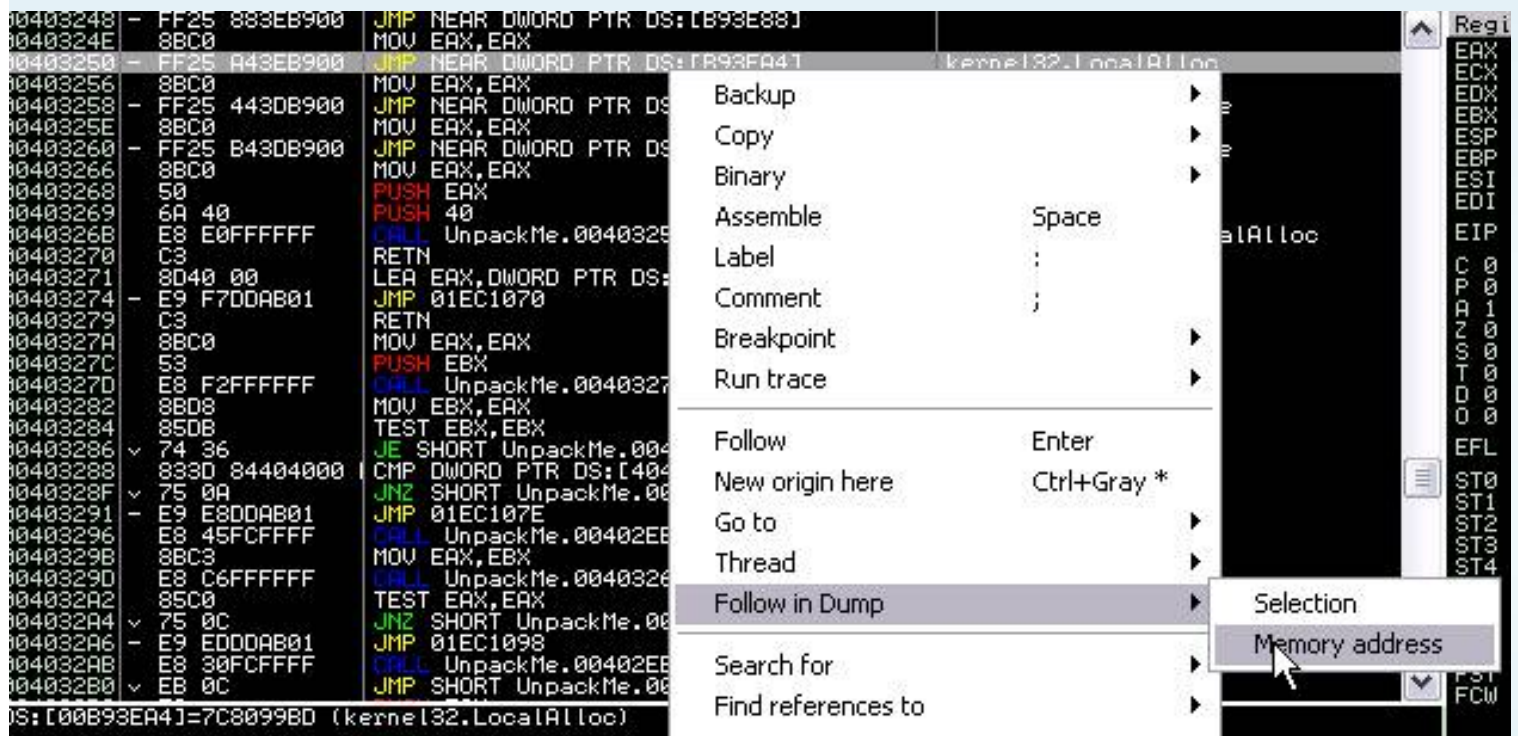
And press Enter you here to

0040330C	53	PUSH EBX	
0040330D	8BD8	MOV EBX,EAX	
0040330F	33C0	XOR EAX,EAX	
00403311	A3 84404000	MOV DWORD PTR DS:[404084],EAX	
00403316	6A 00	PUSH 0	
00403318	E8 2BFFFFFF	CALL UnpackMe.00403248	
0040331D	A3 50564000	MOV DWORD PTR DS:[405650],EAX	
00403322	A1 50564000	MOV EAX,DWORD PTR DS:[405650]	
00403327	A3 8C404000	MOV DWORD PTR DS:[40408C],EAX	
0040332C	33C0	XOR EAX,EAX	
0040332E	A3 90404000	MOV DWORD PTR DS:[404090],EAX	
00403333	33C0	XOR EAX,EAX	
00403335	A3 84404000	MOV DWORD PTR DS:[404084],EAX	

_Thay Below 1 Call command again we also point to the mouse and press Enter to you here:

00403248	- FF25 883EB900	JMP NEAR DWORD PTR DS:[B93E88]	
0040324E	8BC0	MOV EAX,EAX	
00403250	- FF25 A43EB900	JMP NEAR DWORD PTR DS:[B93EA4]	kernel32.LocalAlloc
00403256	8BC0	MOV EAX,EAX	
00403258	- FF25 443DB900	JMP NEAR DWORD PTR DS:[B93D44]	kernel32.TlsGetValue
0040325E	8BC0	MOV EAX,EAX	
00403260	- FF25 B43DB900	JMP NEAR DWORD PTR DS:[B93D84]	kernel32.TlsSetValue
00403266	8BC0	MOV EAX,EAX	
00403268	50	PUSH EAX	
00403269	6A 40	PUSH 40	
0040326B	E8 E0FFFFFF	CALL UnpackMe.00403250	JMP to kernel32.LocalAlloc
00403270	C3	RETN	
00403271	8D40 00	LEA EAX,DWORD PTR DS:[EAX]	
00403274	- E9 F70DAB01	JMP 01EC1070	
00403279	C3	RETN	

_O You should remember "kernel32.LocalAlloc" to little more we find IAT complete easily. Remember completed as selected pictures



Scroll back up the mouse to find the IAT start: **00B93C44**

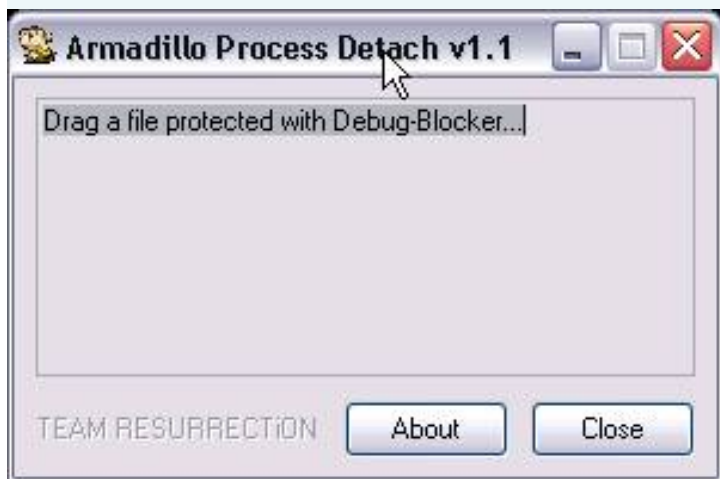
Address	Value	Comment
00B93C3C	FE5421FB	
00B93C40	00000000	
00B93C44	00770487	
00B93C48	ADD7252F	
00B93C4C	EA241941	
00B93C50	008D0055	
00B93C54	010C01BE	
00B93C58	7C809A81	kernel32.VirtualAlloc
00B93C5C	00A26FD9	
00B93C60	00A26FA0	
00B93C64	00A2701A	

Scroll down the search IAT end: **00B93EF0**

Address	Value	Comment
00B93ED0	00A2701C	
00B93ED4	7C80A405	kernel32.GetThreadLocale
00B93ED8	00A2701C	
00B93EDC	00A26F00	
00B93EE0	00A27011	
00B93EE4	00A26F90	
00B93EE8	00A26EA2	
00B93EEC	00A27031	
00B93EF0	00A26FA4	
00B93EF4	00000000	
00B93EF8	00550004	

Yes length: **2AC**

Ok Them. IAT now find complete paste is complete. Hold the window Olly 2. **ArmaDetach 1.1** Open



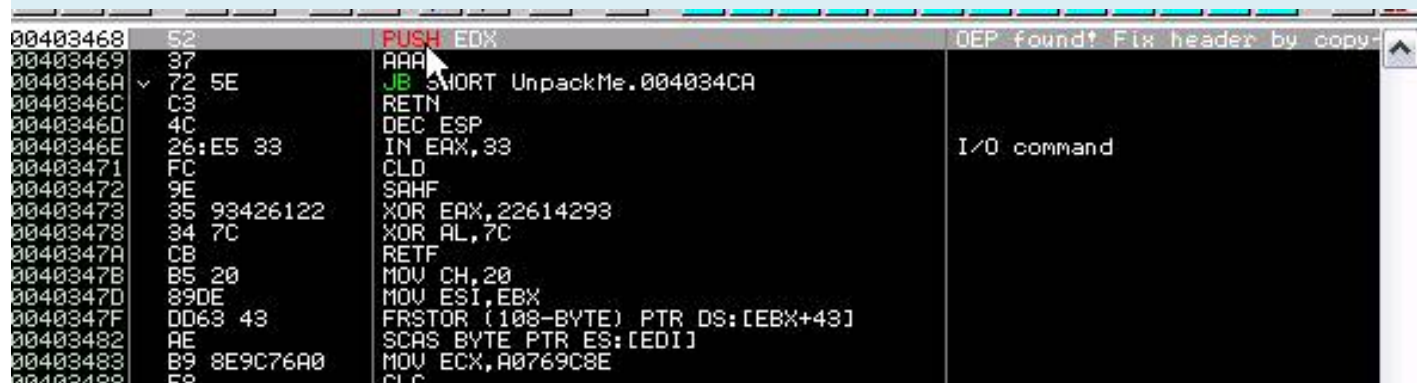
_Loi UnpackMe released into the window to see the program as follows:



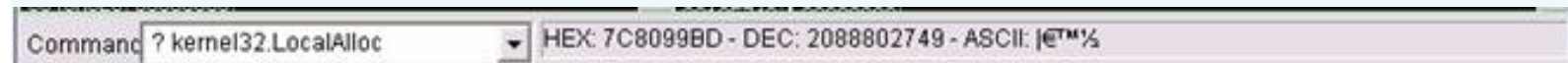
1 _Mo Olly again and [Child process ID](#) and Attach. F9, F12, to patch 558B

00447733	55	PUSH EBP	
00447734	8BEC	MOV EBP,ESP	
00447736	6A FF	PUSH -1	
00447738	68 88114700	PUSH UnpackMe.00471188	
0044773D	68 70744400	PUSH UnpackMe.00447470	SE handler installation
00447742	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
00447748	50	PUSH EAX	
00447749	64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
00447750	83EC 58	SUB ESP,58	
00447753	53	PUSH EBX	
00447754	56	PUSH ESI	
00447755	57	PUSH EDI	
00447756	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
00447759	FF15 88A14600	CALL NEAR DWORD PTR DS:[<&KERNEL32.GetU	kernel32.GetVersion
0044775F	33D2	XOR EDX,EDX	
00447761	8AD4	MOV DL,AH	
00447763	8915 14284700	MOV DWORD PTR DS:[472814],EDX	

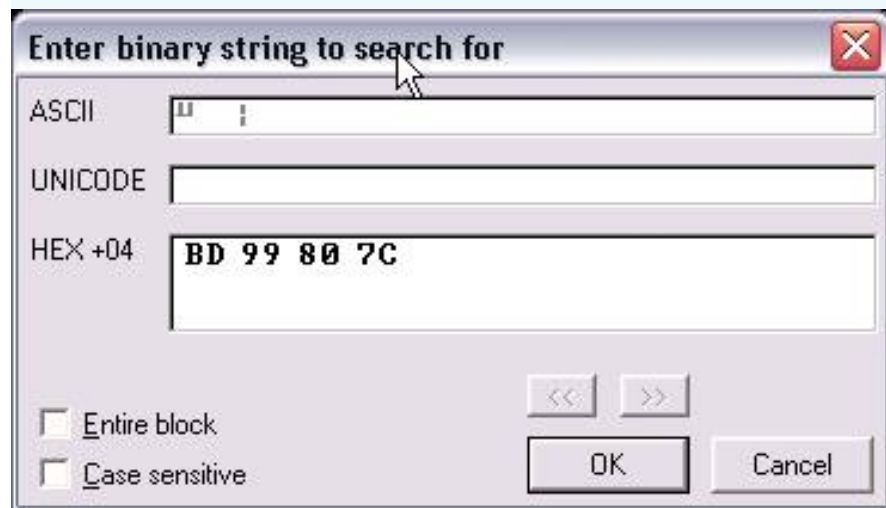
_ Use OllyScript 0.92 Script run "[Armadillo 4.30a - standard script](#) (with the tut). The script is finished running to you here.



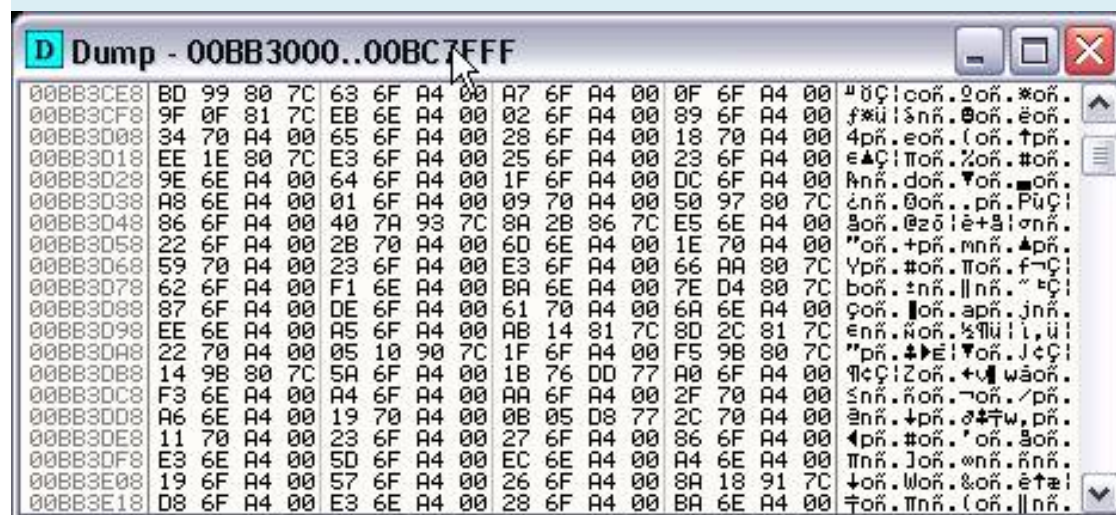
_ You remember "**kernel32.LocalAlloc**" at this time to consider that you do not remember? Now time to use it gòi! Ok, type the following:



_Alt + M, Ctrl + B, reverse the number again as follows:



_nhan Ok we are as follows:



_Alt + C, in the window dump press Ctrl + G and enter: [00bb3CE8](#)

Address	Value	Comment
00BB3CE0	00A46FD2	
00BB3CE4	00A46F0F	
00BB3CE8	7C809980	kernel32.LocalAlloc
00BB3CEC	00A46F63	
00BB3CF0	00A46FA7	
00BB3CF4	00A46F0F	
00BB3CF8	7C810F9F	kernel32.WriteFile
00BB3CFC	00A46EEB	
00BB3D00	00A46F02	
00BB3D04	00A46F89	
00BB3D08	00A47034	

_cuon upturn find IAT start: [00BB3C44](#)

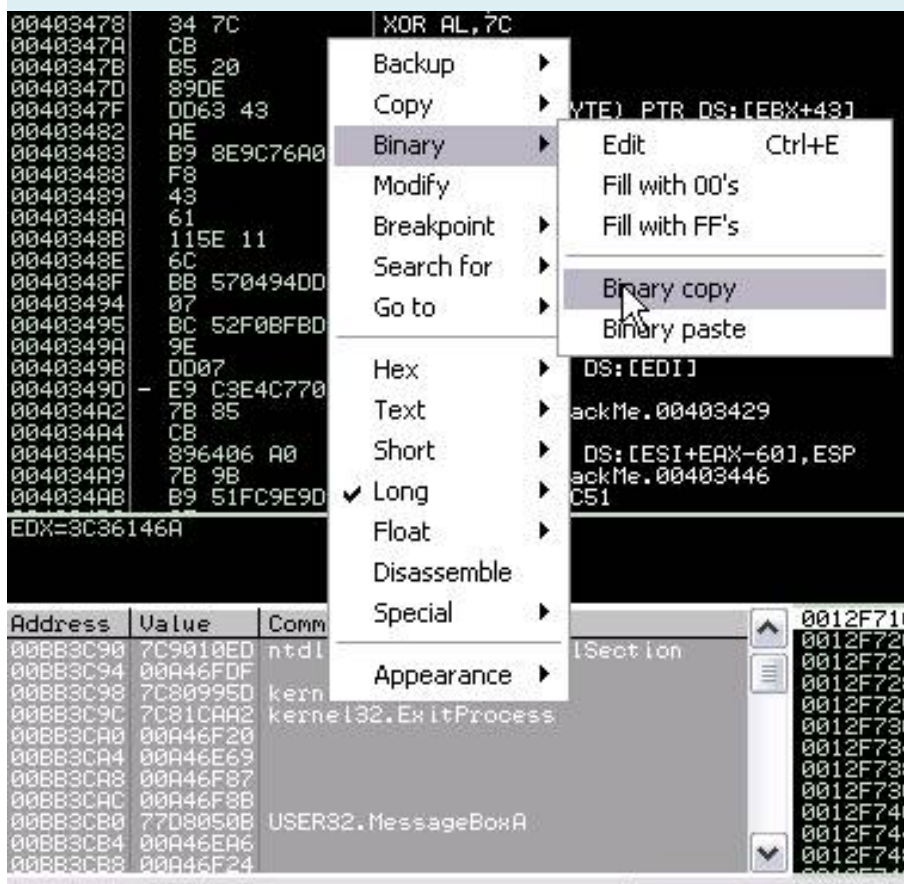
Address	Value	Comment
00BB3C38	015410AC	
00BB3C3C	FEEC21F	
00BB3C40	00000000	
00BB3C44	A0770487	
00BB3C48	ADD7252F	
00BB3C4C	EA241941	
00BB3C50	013F0055	
00BB3C54	010C011F	
00BB3C58	7C809A81	kernel32.VirtualAlloc
00BB3C5C	00A46FD9	
00BB3C60	00A46FA0	

_Cuon To find IAT end: [00BB3EF0](#)

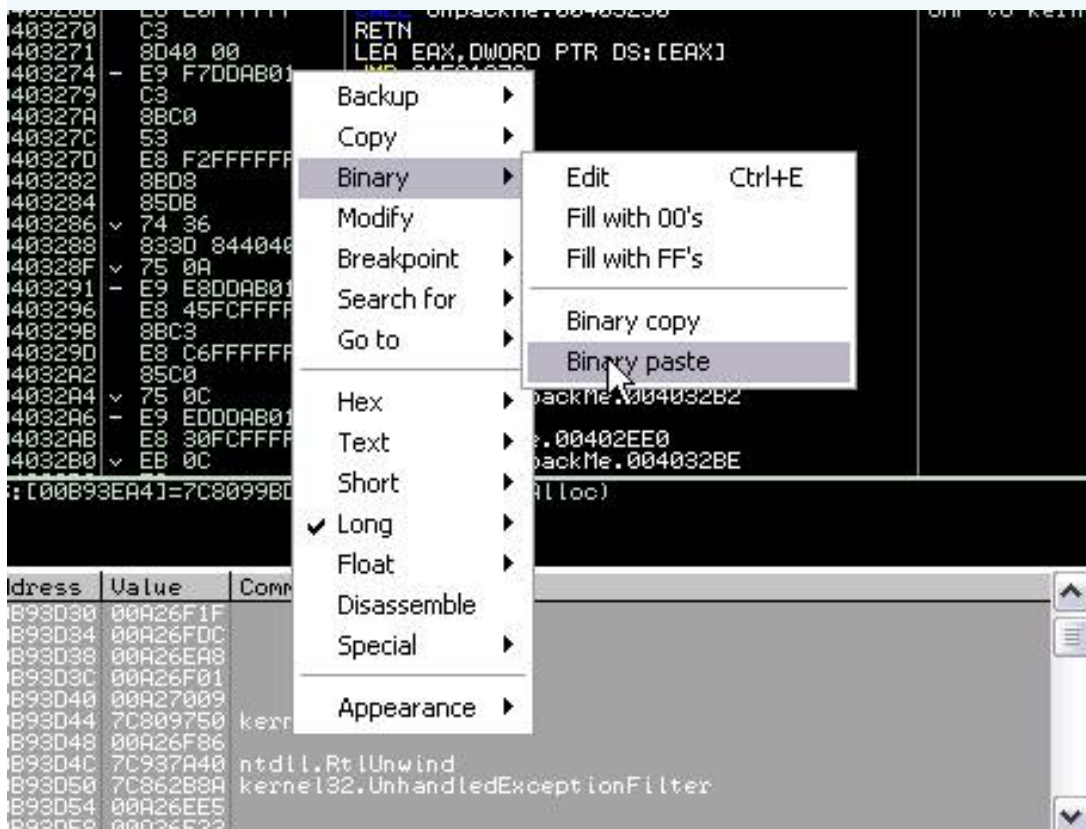
Address	Value	Comment
00BB3EC0	00A46FE3	
00BB3ED0	00A4701C	
00BB3ED4	7C80A405	kernel32.GetThreadLocale
00BB3ED8	00A4701C	
00BB3EDC	00A46F00	
00BB3EE0	00A47011	
00BB3EE4	00A46F90	
00BB3EE8	00A46EA2	
00BB3EEC	00A47031	
00BB3EF0	00A46FA4	
00BB3EF4	00000000	

_co length: [2AC](#)

_Danh Mark from the start to IAT IAT End and select as follows:

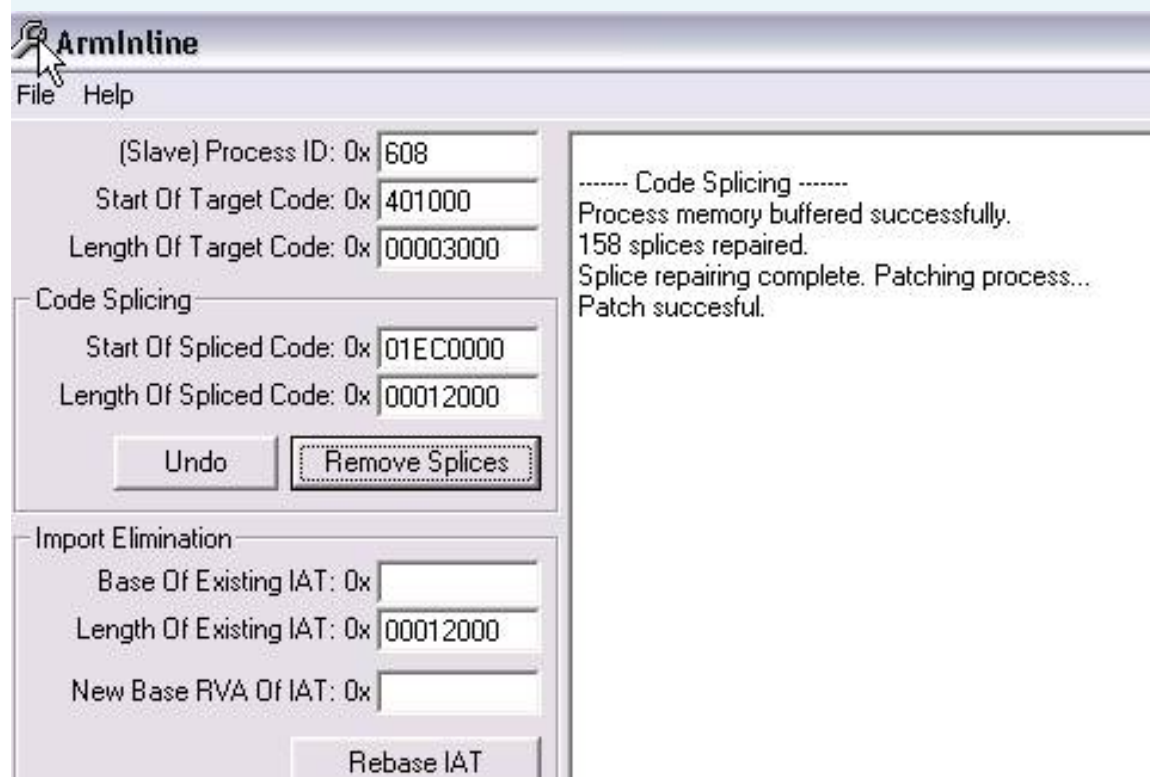


_Ok! now you work we now paste the IAT is not complete only.

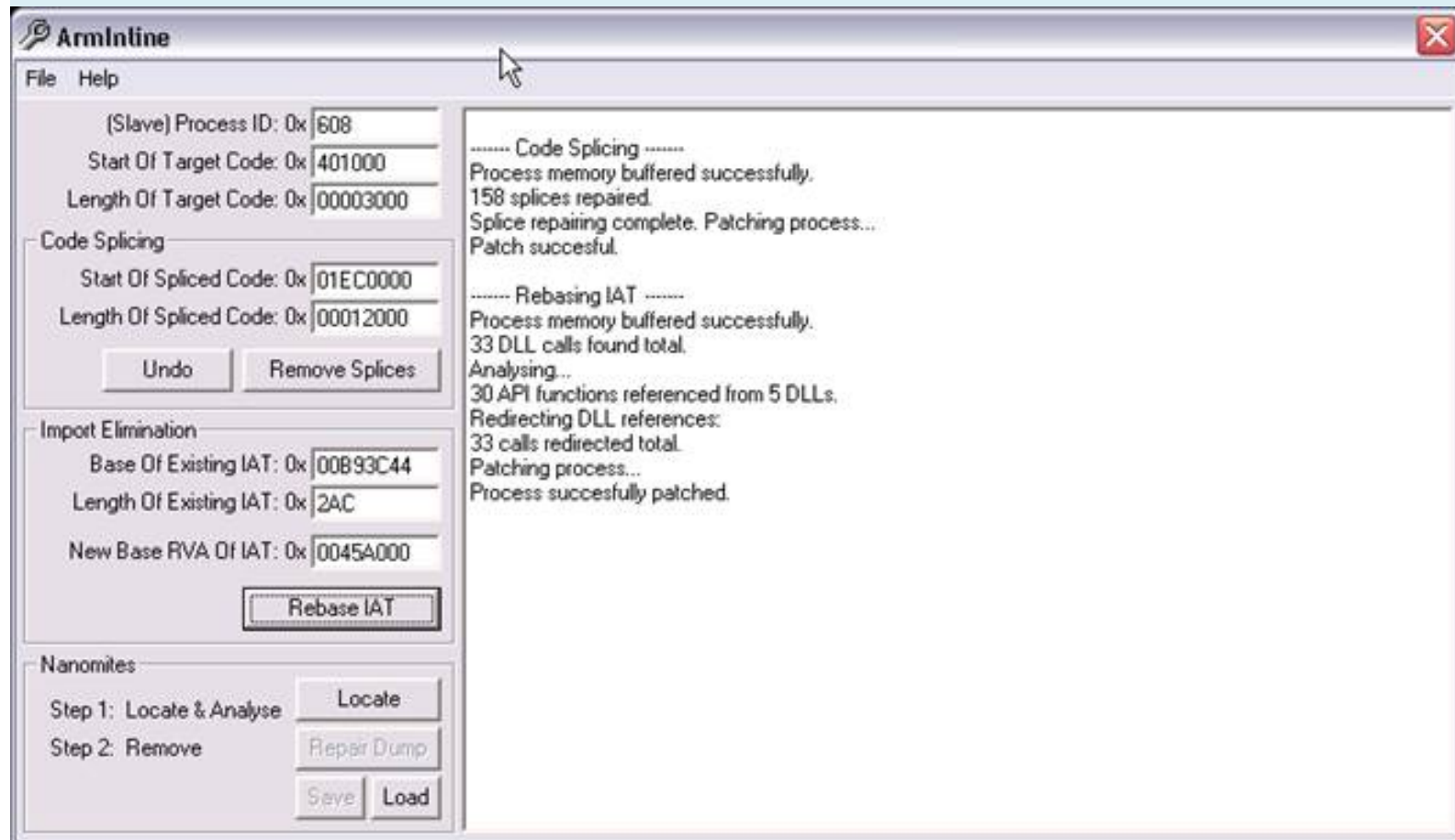


_Nhu This is our Fix the IAT. ArmaDetach now closed and olly there again. Next we Fix **IAT elimination +**

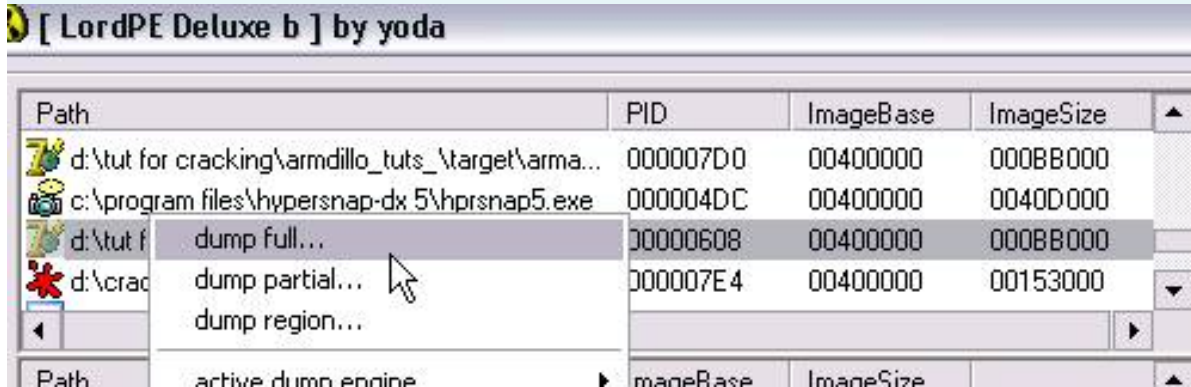
Code Splicing. ArmaInline M in to fill up as follows:



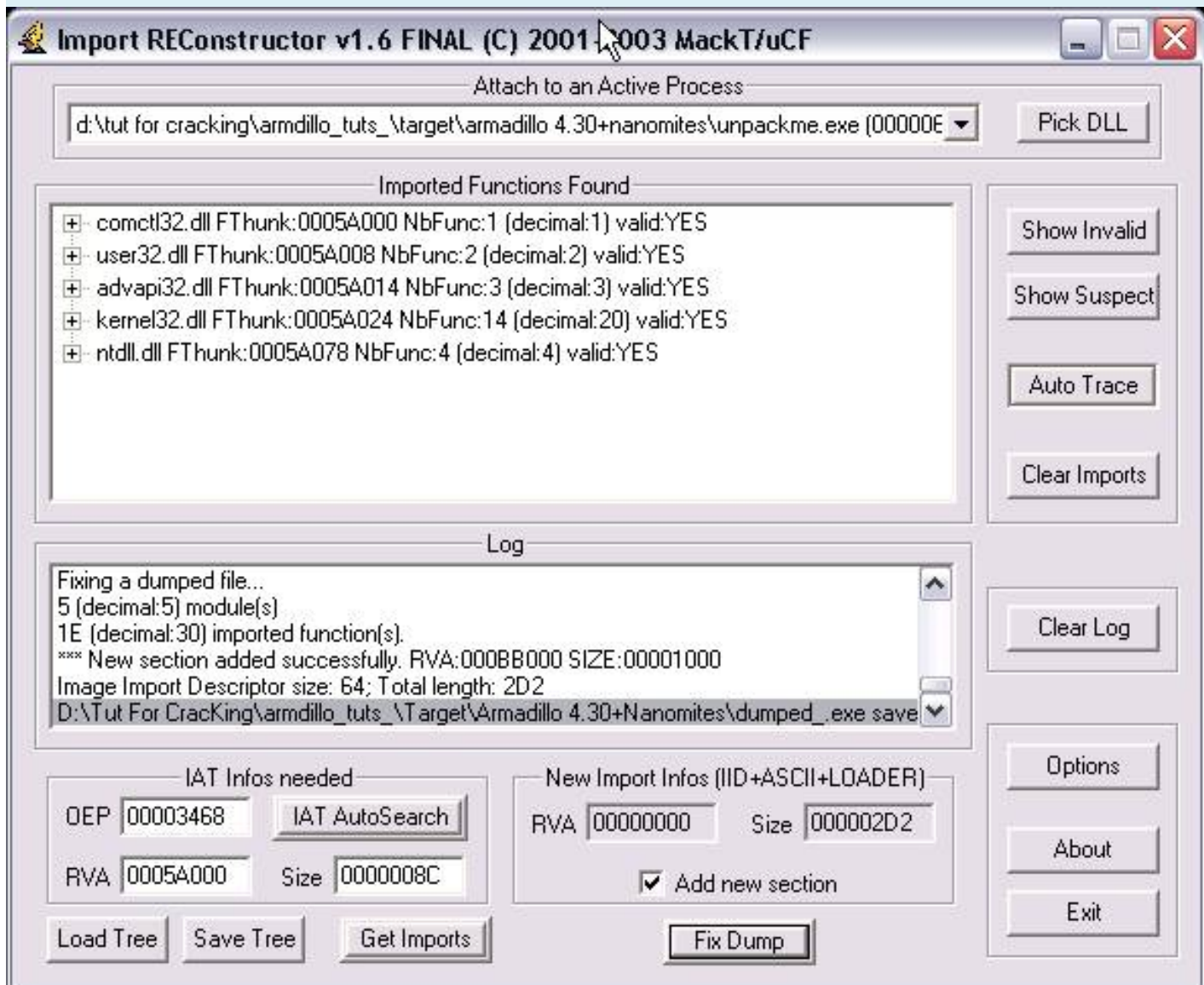
Xong Thằng **Code Splicing** g io to **IAT elimination**



_Tot Call, dump Full stop! Use LordPE



_ Open ImportREC fill and dump Fix



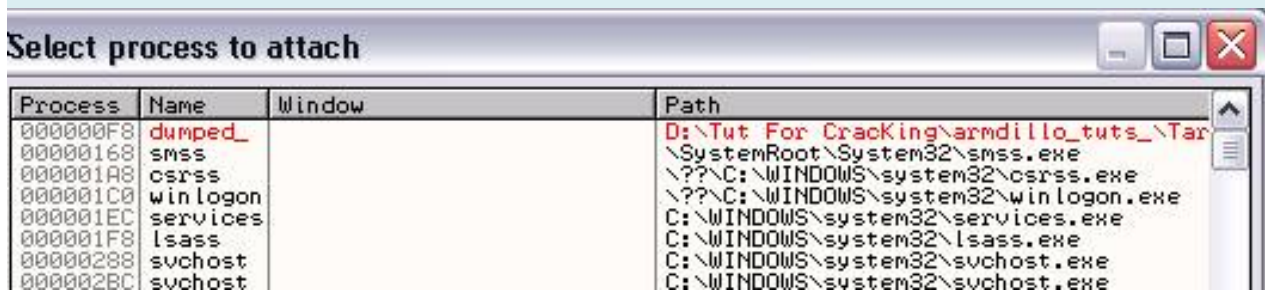
_ Run the file to try "dumped_.exe"



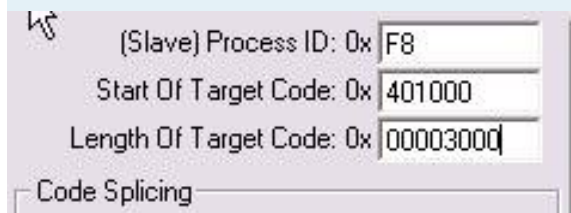
_ Do not Send button Press

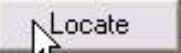


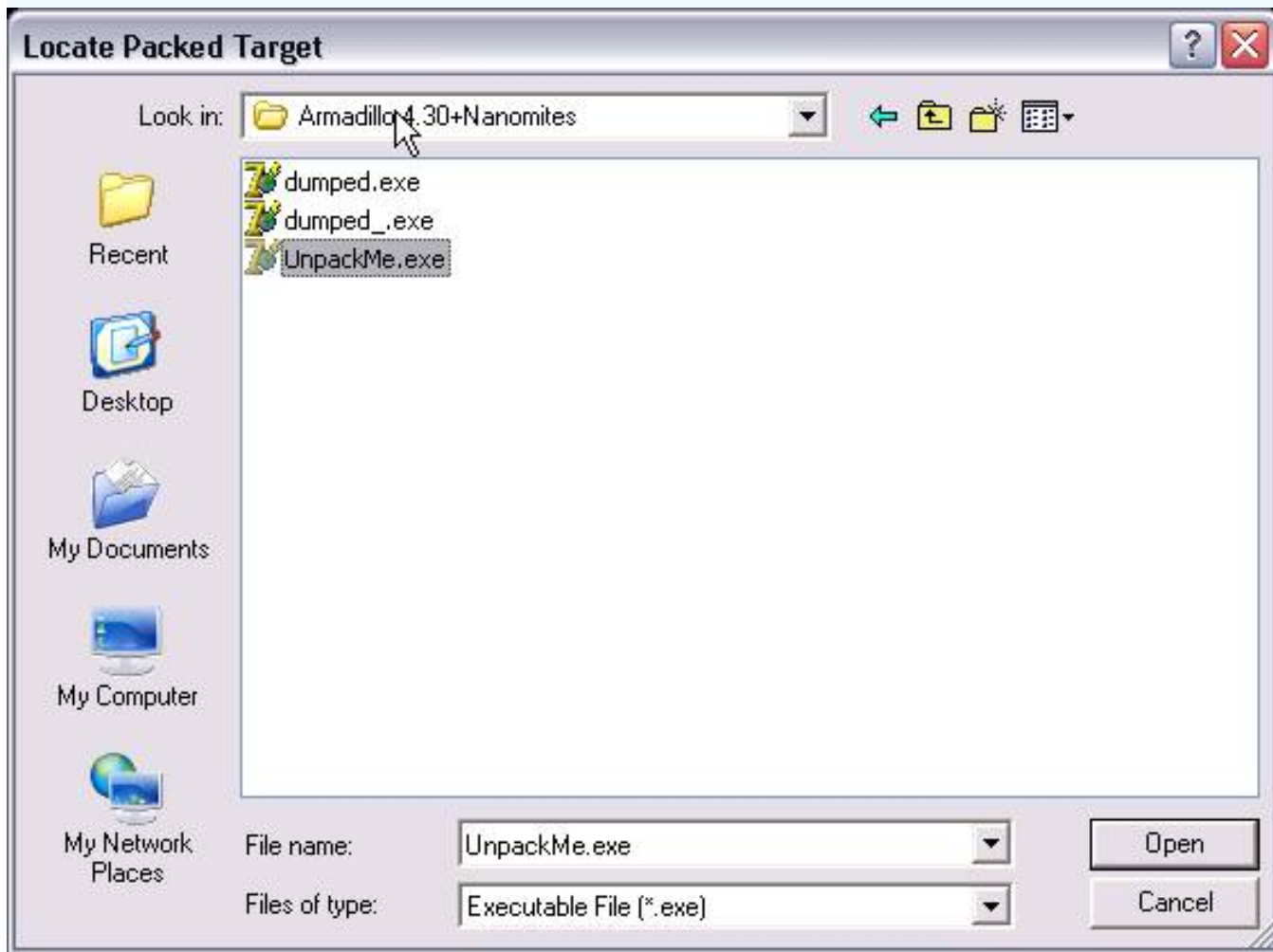
_ He he! Meet cases this is not our governor is the wrong signal by the Ministry of Nano appears that that! If these children are sure daughter named Nanomite s was very difficult to unpack it! He he. Right now, Close to 2 of the book Olly always. Load File "dumped_.exe" to Olly and PID



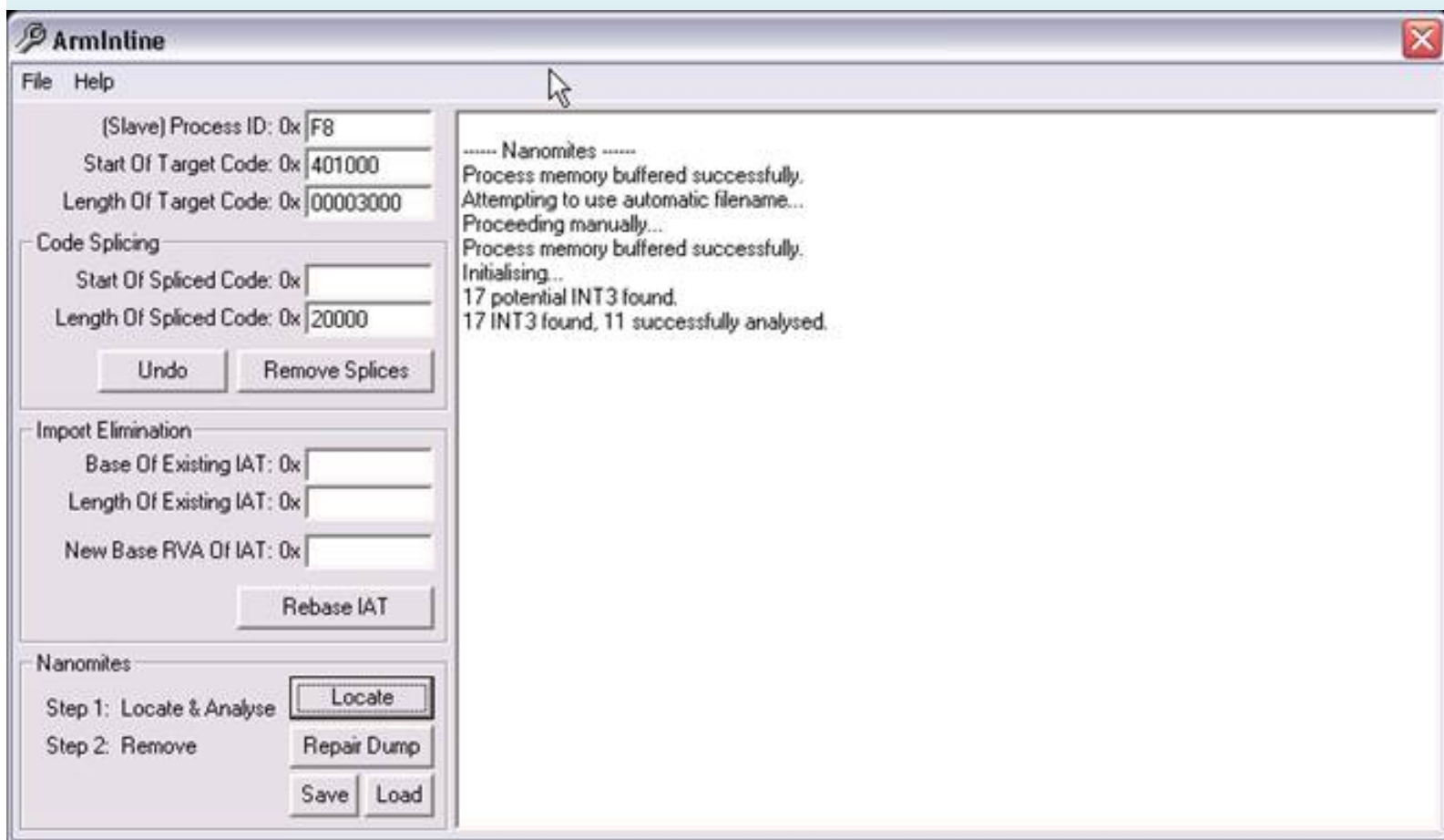
_ Dien To ArmInline



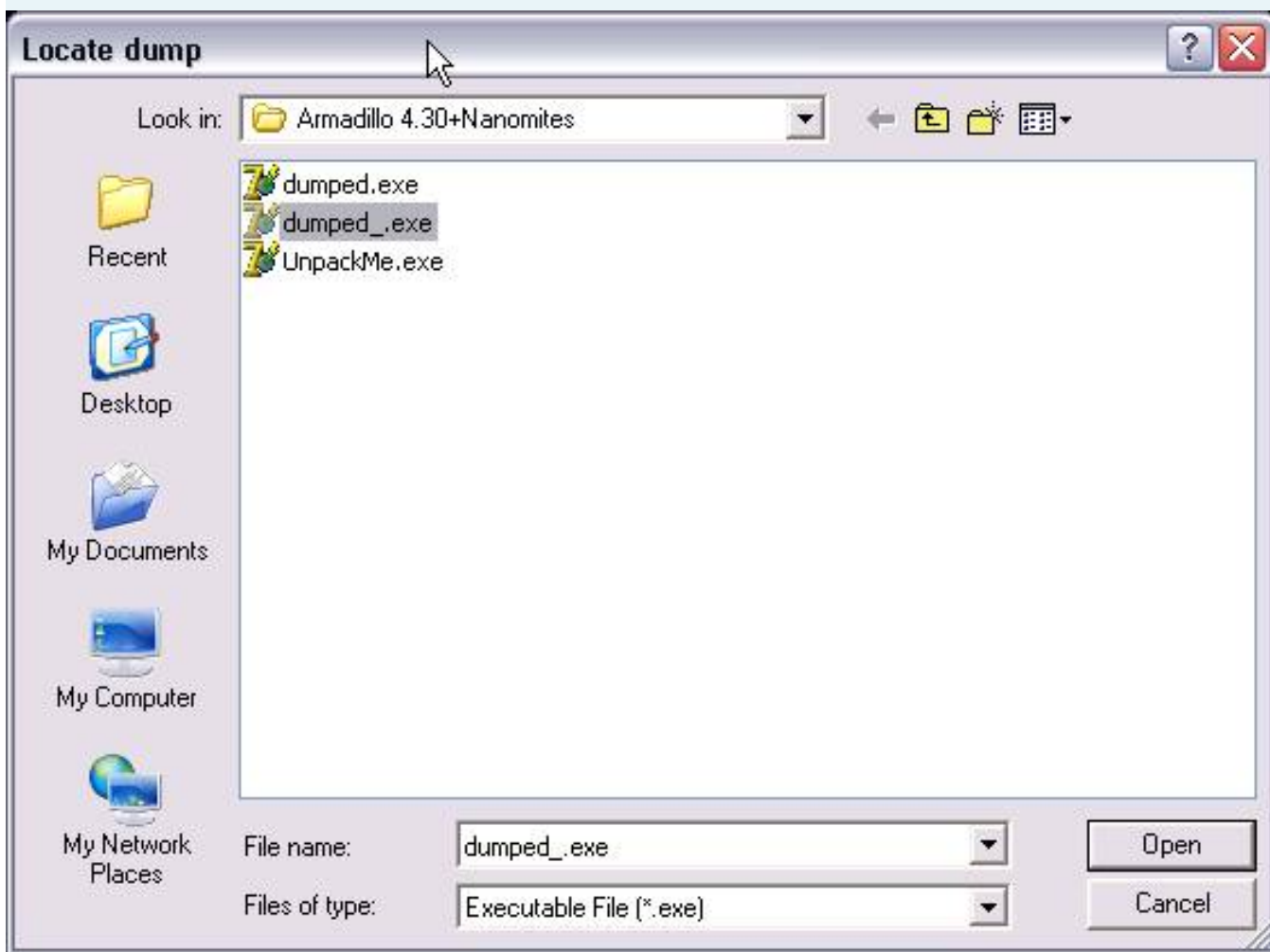
Click  and select File UnpackMe



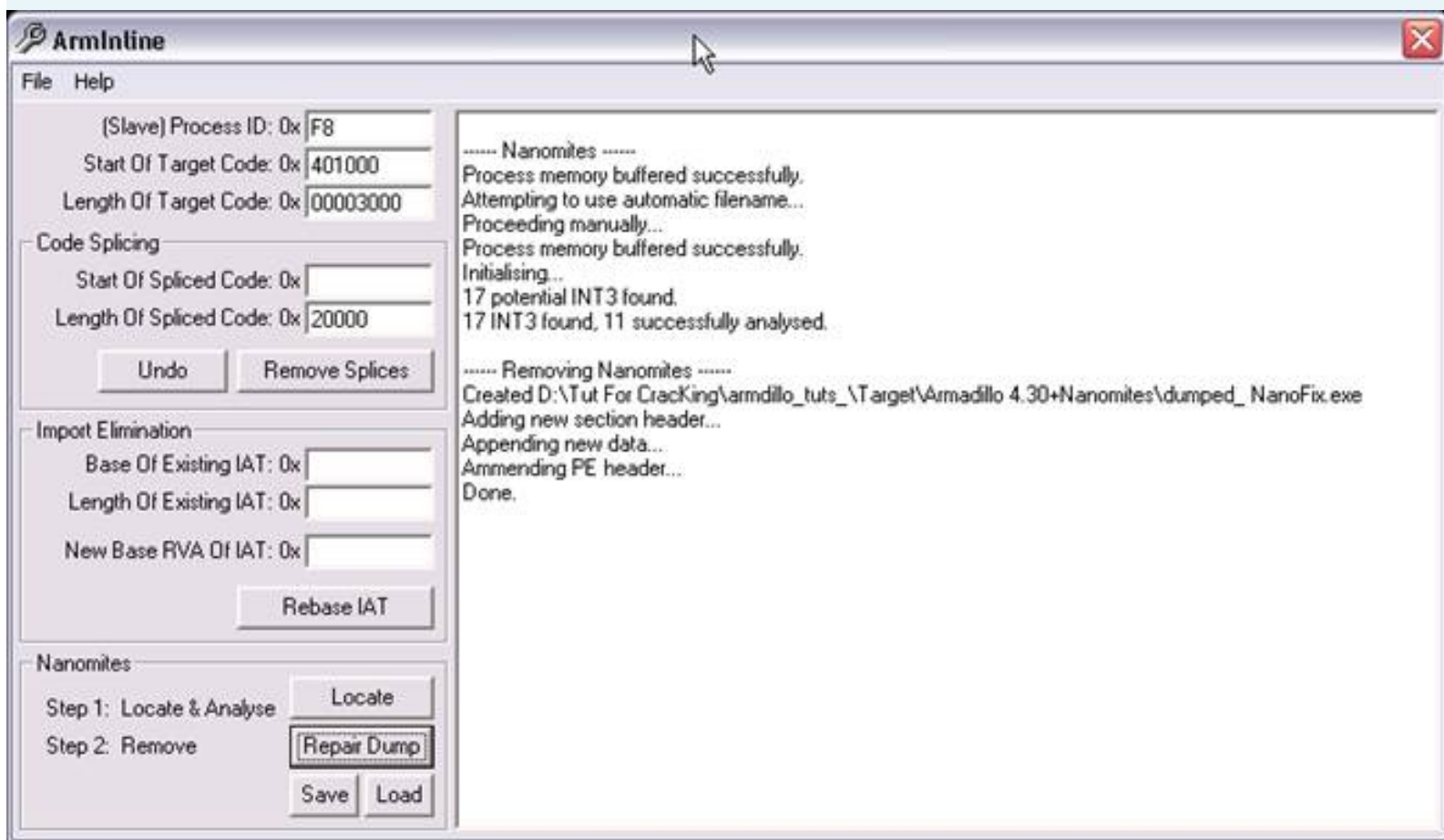
ta found as follows:



_Nhan More  select "dumped_.exe"



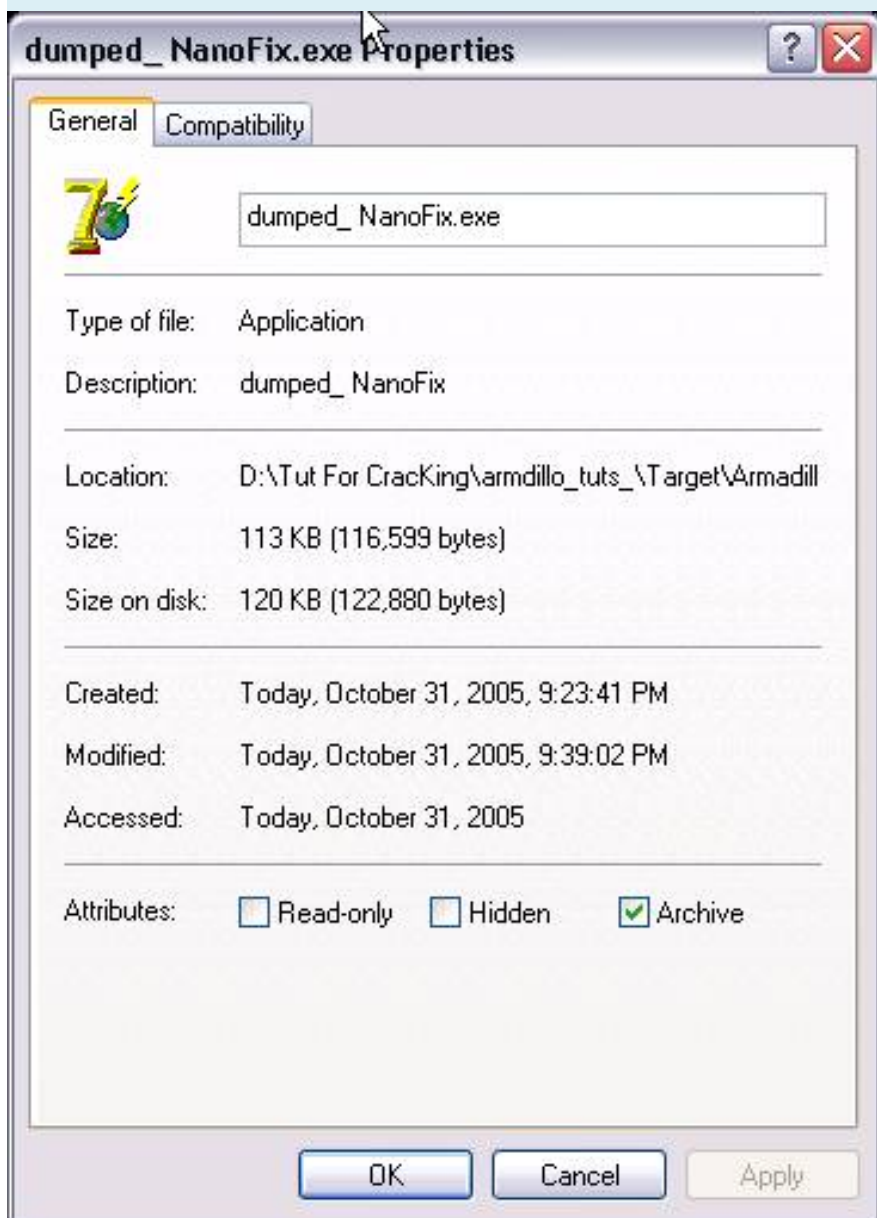
_ He he! Done gòi of Uncle oi.



Chay "Dumped NanoFix.exe" see the stars



CFF _Dung Delete Section admit they have one new File Size 113 Kb



_Unpack Done !!!!!!!!! Bye

Written by Why Not Bar

Unwrapping_Reflexive_Arcade_EvilInvasion

tlandn

Target: Evil Invasion

Download: <http://www.reflexive.com/> (Use the search function to find the game)

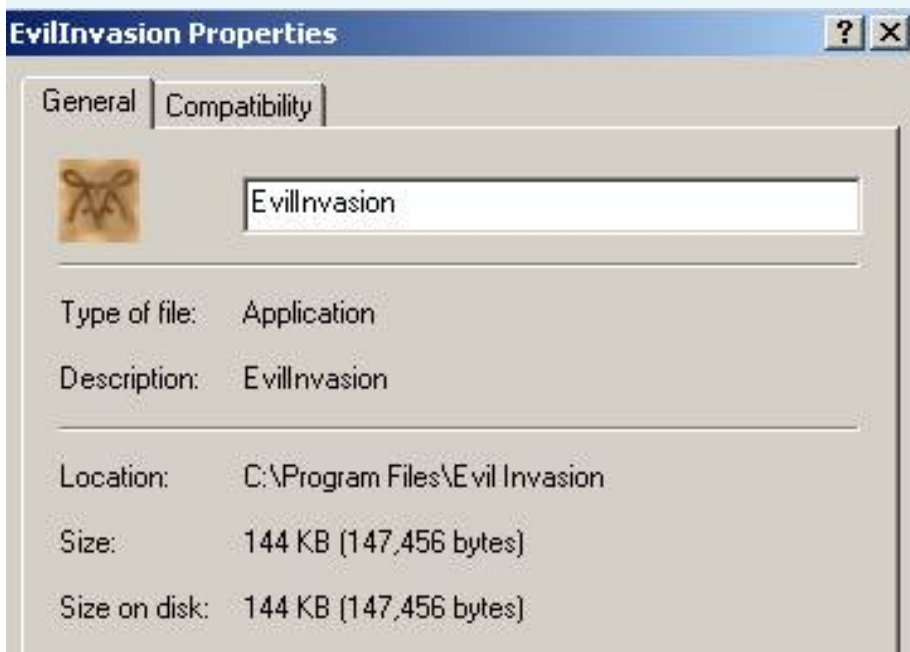
Toolz: OllyDbg J

Then I have just read by tut HighEnergy about unwrap the game's site <http://www.reflexive.com/>. Look or should refer to write this tut for you.

Ok. We start. Install program. View the folder settings to see what's:



You see the size of the file EvilInvasion.exe.

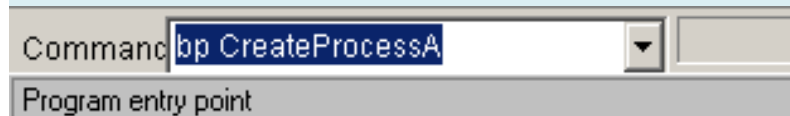


Only 144 KB. We find this size too small for 1 game.

OK. Open OllyDbg load file EvilInvasion.exe.

00411075	55	PUSH EBP	
00411076	8BEC	MOV EBP,ESP	
00411078	6A FF	PUSH -1	
0041107A	68 E8A84100	PUSH EvilInva.0041A8E8	
0041107F	68 28584100	PUSH EvilInva.00415828	
00411084	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	SE handler installation
0041108A	50	PUSH EAX	
0041108B	64:8925 0000	MOV DWORD PTR FS:[0],ESP	
00411092	83EC 58	SUB ESP,58	

Set CreateProcessA BP:



Press F9. Info on the register:



Click "Play Game". We break in OllyDbg at:

7C802367	8BFF	MOV EDI,EDI	EvilInva.00421D08
7C802369	55	PUSH EBP	
7C80236A	8BEC	MOV EBP,ESP	
7C80236C	6A 00	PUSH 0	
7C80236E	FF75 2C	PUSH DWORD PTR SS:[EBP+2C]	
7C802371	FF75 28	PUSH DWORD PTR SS:[EBP+28]	

Look Stack window:


```

0012F054 00401F8F CALL to CreateProcessA from EvilInva.00401F89
0012F058 00421D2C ModuleFileName = "EvilInvasion.RWG"
0012F05C 0012F08C CommandLine = ""EvilInvasion.RWG" ""
0012F060 00000000 pProcessSecurity = NULL
0012F064 00000000 pThreadSecurity = NULL
0012F068 00000000 InheritHandles = FALSE
0012F06C 00000004 CreationFlags = CREATE_SUSPENDED
0012F070 00000000 pEnvironment = NULL
0012F074 00421E3C CurrentDir = "C:\Program Files\Evil Invasion"
0012F078 0012FC88 pStartupInfo = 0012FC88
0012F07C 00421D18 pProcessInfo = EvilInva.00421D18
0012F080 00421D2C ASCII "EvilInvasion.RWG"
0012F084 00421D08 EvilInva.00421D08

```

You note their seats. The program will load the file into EvilInvasion.RWG. Press Alt-F9.

```

00401F7E . 8D85 B8F3FFF LEA EAX,[LOCAL.786]
00401F84 . 53 PUSH EBX
00401F85 . 50 PUSH EAX
00401F86 . FF75 08 PUSH [ARG.1]
00401F89 . FF15 64A1410 CALL DWORD PTR DS:[<&KERNEL32.CreatePro
00401F8F . FF77 18 PUSH DWORD PTR DS:[EDI+18]
00401F92 . 53 PUSH EBX
00401F93 . 68 FF0F1F00 PUSH 1F0FFF
00401F98 . FF15 70A1410 CALL DWORD PTR DS:[<&KERNEL32.OpenProce
00401F9E . 3BC3 CMP EAX,EBX
00401FA0 . 8945 FC MOV [LOCAL.1],EAX
00401FA3 . 0F84 B000000 JE EvilInva.00402059

```

pProcessSecurity
 CommandLine
 ModuleFileName
 CreateProcessA
 ProcessId = E14
 Inheritable
 Access = PROCESS_ALL_A
 OpenProcess

Pull down 1 billion, you see:

```

00401FC2 . 85C0 TEST EAX,EAX
00401FC4 . 74 37 JE SHORT EvilInva.00401FFD
00401FC6 . FFB7 2801000 PUSH DWORD PTR DS:[EDI+128]
00401FCC . 8D8D B8FBFFF LEA ECX,[LOCAL.274]
00401FD2 . E8 84370000 CALL EvilInva.0040575B
00401FD7 . 391E CMP DWORD PTR DS:[ESI],EBX
00401FD9 . 895D 08 MOV [ARG.1],EBX
00401FDC . 76 2C JBE SHORT EvilInva.0040200A
00401FDE . 8D8D B8FBFFF LEA ECX,[LOCAL.274]
00401FE4 . E8 89F1FFFF CALL EvilInva.00401172
00401FE9 . 8B4E 08 MOV ECX,DWORD PTR DS:[ESI+8]
00401FEC . 8B55 08 MOV EDX,[ARG.1]
00401FEF . 03CA ADD ECX,EDX
00401FF1 . 2801 SUB BYTE PTR DS:[ECX],AL
00401FF3 . 42 INC EDX
00401FF4 . 3B16 CMP EDX,DWORD PTR DS:[ESI]
00401FF6 . 8955 08 MOV [ARG.1],EDX
00401FF9 . 72 E3 JB SHORT EvilInva.00401FDE
00401FFB . EB 0D JMP SHORT EvilInva.0040200A
00401FFD . 68 74C44100 PUSH EvilInva.0041C474
00402002 . 53 PUSH EBX
00402003 . E8 22380000 CALL EvilInva.0040582A
00402008 . 59 POP ECX
00402009 . 59 POP ECX
0040200A . 8D45 0C LEA EAX,[ARG.2]
0040200D . 895D 0C MOV [ARG.2],EBX
00402010 . 50 PUSH EAX
00402011 . FF36 PUSH DWORD PTR DS:[ESI]
00402013 . FF76 08 PUSH DWORD PTR DS:[ESI+8]
00402016 . FF76 04 PUSH DWORD PTR DS:[ESI+4]
00402019 . FF75 FC PUSH [LOCAL.1]
0040201C . FF15 6CA1410 CALL DWORD PTR DS:[<&KERNEL32.WriteProc

```

ASCII "Could not Read encrypted EXE."

pBytesWritten
 BytesToWrite
 Buffer
 Address
 hProcess
 WriteProcessMemory

OK. Talking na conduct is this: file EvilInvasion.exe role as a loader, it displays notifications to register and decrypt files EvilInvasion.RWG. Done through it will write the byte code has a good memory and run areas. EvilInvasion.RWG file as an exe files have been encrypted.

So how to resolve how? Very simply, we will wait encrypted when it is finished copying the area code has been on a file that, rename the file extensions. Exe is finished.

Press Ctrl-F2 to restart Olly. Set WriteProcessMemory BP:

Command

Program entry point

Press F9. Notification window appear. Click "Play Game". We will stop here:

7C80220F	8BFF	MOV EDI,EDI	EvilInva.00421D08
7C802211	55	PUSH EBP	
7C802212	8BEC	MOV EBP,ESP	
7C802214	51	PUSH ECX	
7C802215	51	PUSH ECX	
7C802216	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]	
7C802219	53	PUSH EBX	
7C80221A	8B5D 14	MOV EBX,DWORD PTR SS:[EBP+14]	
7C80221D	56	PUSH ESI	
7C80221E	8B35 B812807C	MOV ESI,DWORD PTR DS:[<&ntdll.NtProtectVirtualMemory	ntdll.ZwProtectVirtualMemory

Stack window:

0012F068	00402022	CALL to WriteProcessMemory from Evil
0012F06C	000000B4	hProcess = 000000B4 (window)
0012F070	00438B44	Address = 438B44
0012F074	00F50048	Buffer = 00F50048
0012F078	000016F7	BytesToWrite = 16F7 (5879.)
0012F07C	0012FCE0	BytesWritten = 0012FCE0
0012F080	00421D2C	ASCII "EvilInvasion.RWG"
0012F084	00421D08	EvilInva.00421D08
0012F088	00000000	

The attention you place on their image. Region byte is encrypted as 00F50048 (your computer may be different) and the length is 16F7.

OK. We will copy this area bytes. In the window of Olly dump. Press Ctrl-G. 00F50048 Enter:

Address	Hex dump	ASCII
0041C000	00 00 00 00	
0041C010	93 29 40 00	
0041C020	12 0F 41 00	
0041C030	00 00 00 00	
0041C040	00 00 00 00	
0041C050	09 18 40 AF	
0041C060	DB 32 F1 60	
0041C070	1E C9 D6 A4	
0041C080	ED ED B5 18	
0041C090	9D 64 D5 71	
0041C0A0	58 9E 32 23	
0041C0B0	77 9E C1 B7	
0041C0C0	92 99 8C 9D	
0041C0D0	32 07 22 38	

Click OK. We here:

00F50048	6A 74 68 F0 2A 44 00 E8 34 03 00 00 33 DB 89 5D	jth=*D.3
00F50058	E0 53 8B 3D 7C E0 43 00 FF D7 66 81 38 4D 5A 75	αSi=!αC.

We will select areas bytes beginning 00F50048 extended to 0F5173F (00F50048 + 16F7 = 0F5173F).

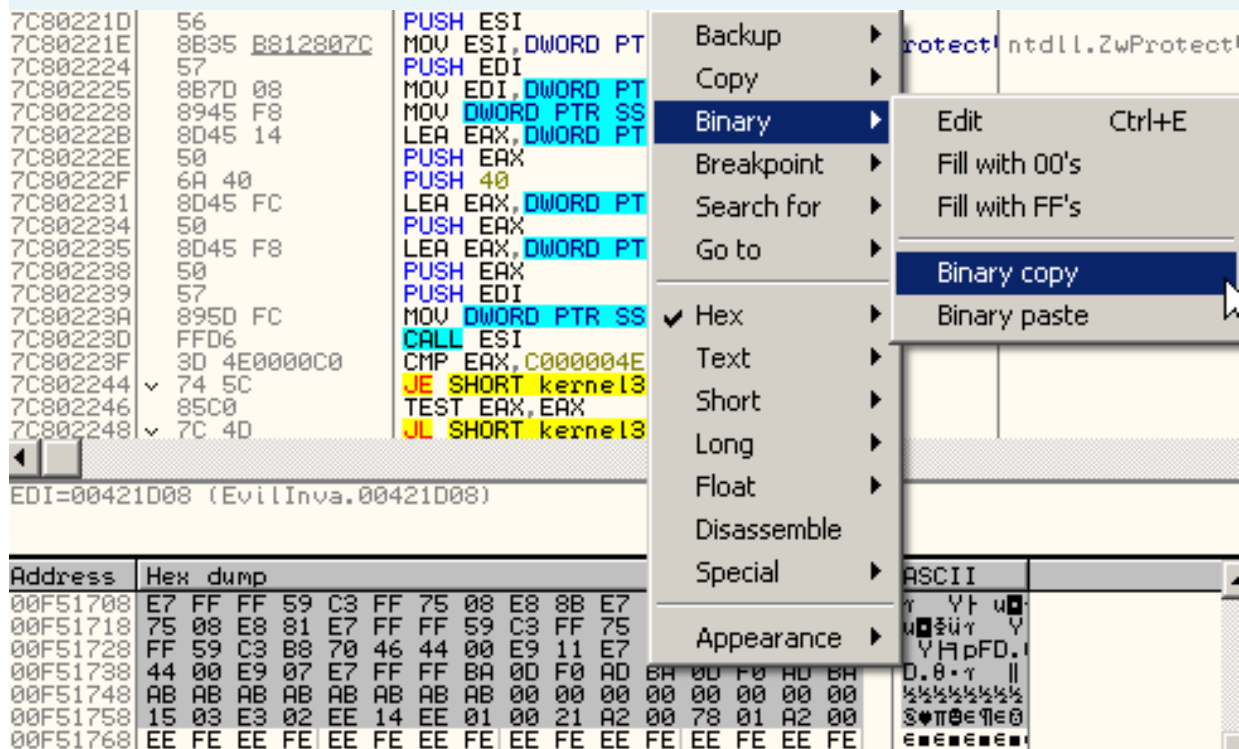
OK. The beginning:

00F50048	6A 74 68 F0 2A 44 00 E8 34 03 00 00 33 DB 89 5D	jth=*D.3
00F50058	E0 53 8B 3D 7C E0 43 00 FF D7 66 81 38 4D 5A 75	αSi=!αC.
00F50068	1F 8B 48 3C 03 C8 81 39 50 45 00 00 75 12 0F B7	∇(H<•u9
00F50078	41 18 3D 0B 01 00 00 74 1F 3D 0B 02 00 00 74 05	A↑=00..t
00F50088	89 5D E4 EB 27 83 B9 84 00 00 00 0E 76 F2 33 C0	ē123'ā ā
00F50098	39 99 F8 00 00 00 EB 0E 83 79 74 0E 76 E2 33 C0	90°...\$ā
00F500A8	39 99 E8 00 00 00 0F 95 C0 89 45 E4 89 5D FC 6A	90\$...#0
00F500B8	02 FF 15 58 E2 43 00 59 83 0D E8 A7 44 00 FF 83	0 \$XTC.Y
00F500C8	0D EC A7 44 00 FF FF 15 5C E2 43 00 8B 0D AC A7	.α0D. 3
00F500D8	44 00 89 08 FF 15 60 E2 43 00 8B 0D A8 A7 44 00	D.ā 3'P

End (note: enough money nor a copy):

00F51708	E7 FF FF 59 C3 FF 75 08 E8 8B E7 FF FF 59 C3 FF	γ Yf u
00F51718	75 08 E8 81 E7 FF FF 59 C3 FF 75 08 E8 77 E7 FF	u3ūγ Y
00F51728	FF 59 C3 B8 70 46 44 00 E9 11 E7 FF FF B8 C0 46	YH pFD.
00F51738	44 00 E9 07 E7 FF FF BA 0D F0 AD BA 0D F0 AD BA	D.0·γ
00F51748	AB AB AB AB AB AB AB 00 00 00 00 00 00 00 00	%&%&%&%&%&
00F51758	15 03 E3 02 EE 14 EE 01 00 21 A2 00 78 01 A2 00	3•π0€9€0
00F51768	EE FE EE FE EE FE EE FE EE FE EE FE EE FE	€€€€€€€€

Click your mouse to select the image as:



Open an Olly 2. Load file EvilInvasion.RWG:

0043BB44	\$ EC	IN AL,DX	I/O command
0043BB45	. 25 11A735A2	AND EAX,A235A711	
0043BB4A	. AF	SCAS DWORD PTR ES:[EDI]	
0043BB4B	. 4A	DEC EDX	
0043BB4C	. 842B	TEST BYTE PTR DS:[EBX],CH	
0043BB4E	. F8	CLC	
0043BB4F	. D29B 1E54657	RCR BYTE PTR DS:[EBX+7465541E],CL	

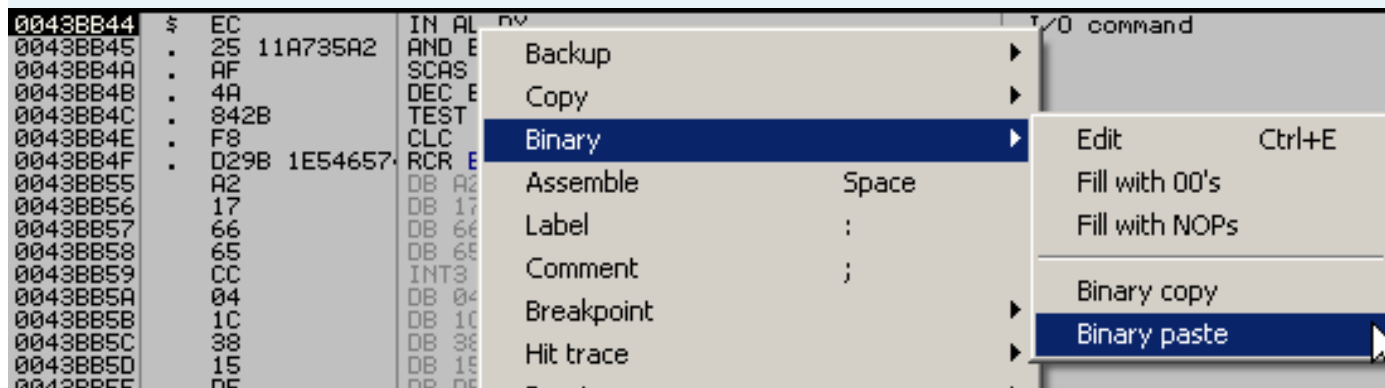
We will choose the region since the end 0043BB44 file. Start:

0043BB44	EC	IN AL,DX	I/O command
0043BB45	25 11A735A2	AND EAX,A235A711	
0043BB4A	AF	SCAS DWORD PTR ES:[EDI]	
0043BB4B	4A	DEC EDX	
0043BB4C	842B	TEST BYTE PTR DS:[EBX],CH	
0043BB4E	F8	CLC	
0043BB4F	D29B 1E54657	RCR BYTE PTR DS:[EBX+7465541E],CL	

Finish:

0043DFE2	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFE4	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFE6	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFE8	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFEA	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFEC	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFEE	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFF0	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFF2	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFF4	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFF6	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFF8	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFFA	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFFC	0000	ADD BYTE PTR DS:[EAX],AL	
0043DFFE	0000	ADD BYTE PTR DS:[EAX],AL	

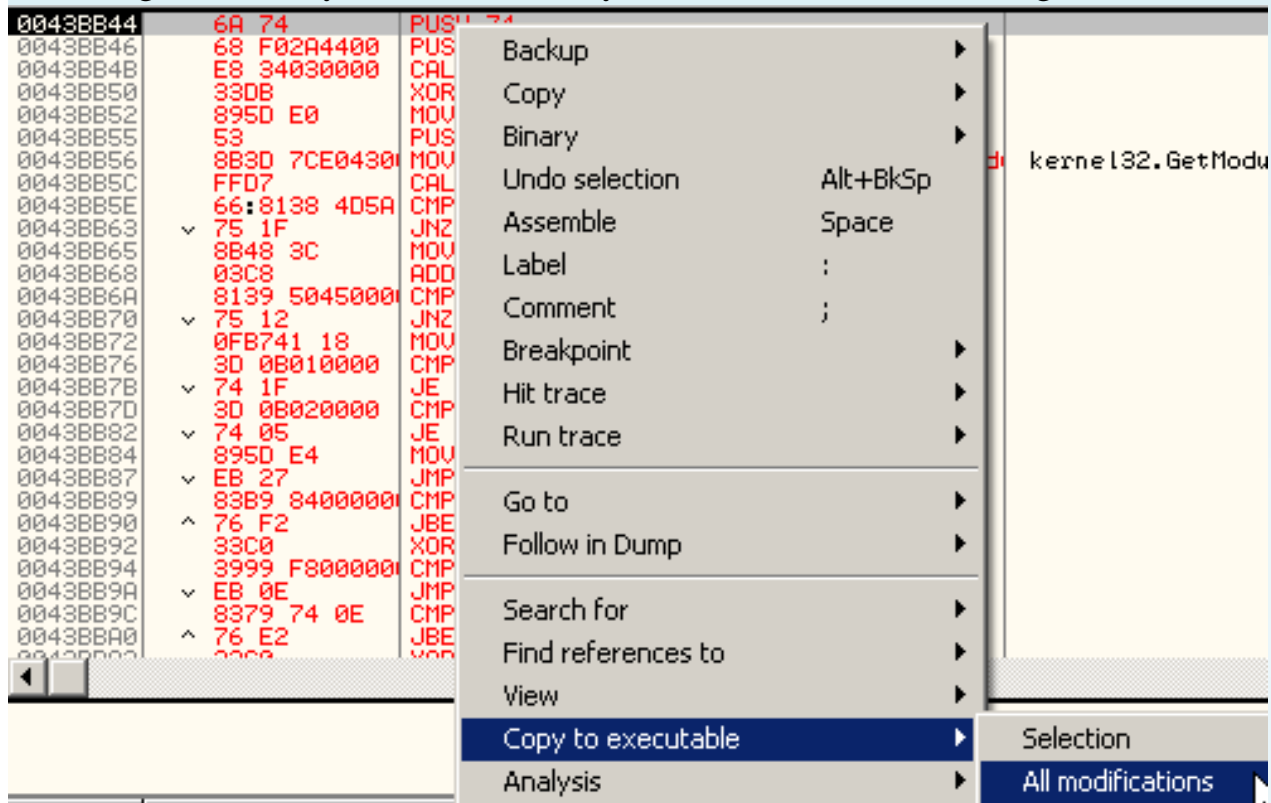
Click your mouse to select the image:



We are:

0043BB44	6A 74	PUSH 74	
0043BB46	68 F02A4400	PUSH EvilInva.00442AF0	
0043BB48	E8 34030000	CALL EvilInva.0043BE84	
0043BB50	33DB	XOR EBX,EBX	
0043BB52	895D E0	MOV DWORD PTR SS:[EBP-20],EBX	
0043BB55	53	PUSH EBX	
0043BB56	8B3D 7CE0430	MOV EDI,DWORD PTR DS:[<&KERNEL32.GetModu	kernel32.GetModuleHandleA
0043BB5C	FFD7	CALL EDI	
0043BB5E	66:8138 4D5A	CMP WORD PTR DS:[EAX],5A4D	
0043BB63	75 1F	JNZ SHORT EvilInva.0043BB84	

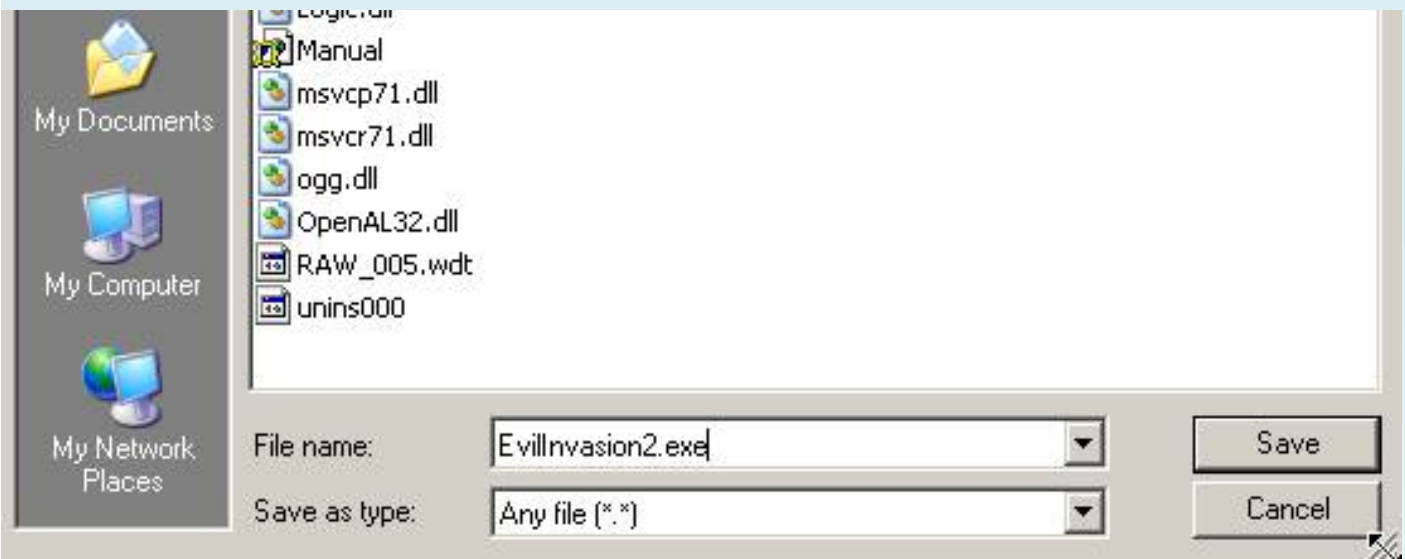
Breathing more easily and J. Click your mouse to select the image:



Report on the button click "Copy All". Click your mouse to select the image:



Name the file is EvilInvasion2.exe:



Test file EvilInvasion2.exe. Run the J



Have fun!
tlandn

Greetz: All VCT memberz, HighEnergy, and you ... J

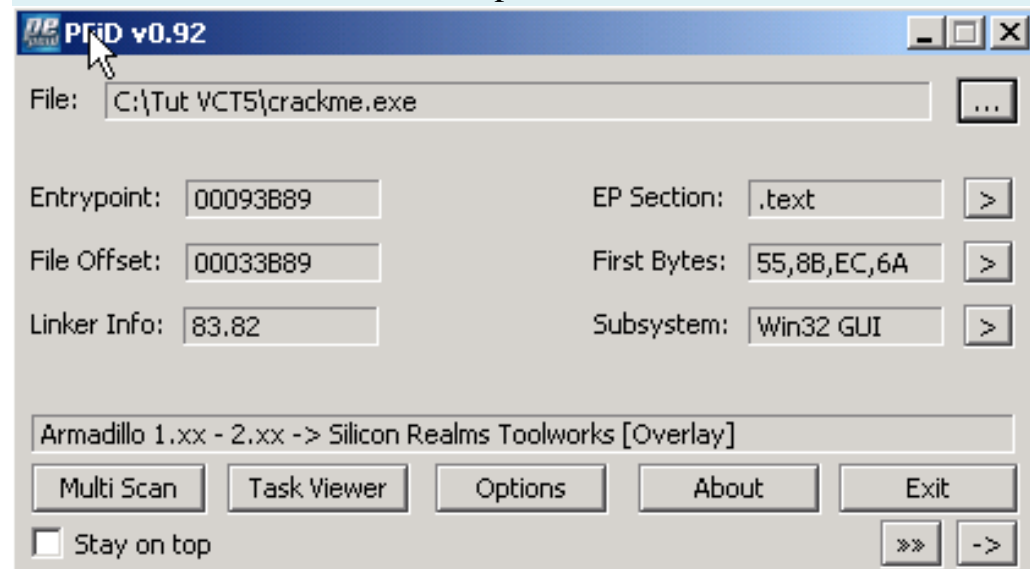
ARMADILLO unpack 3.70a

Target: VCT crackme # 5

Download: <http://tothesky.us>

Tut maker: tlandn

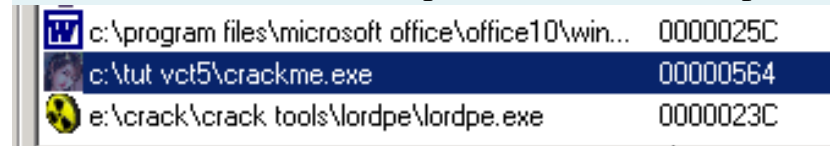
PEID first used to determine the pack with what?



0. Identify strategies to cope with the Armadillo:

As in many Armadillo option allows choosing the type of protection many different so we must consider the file crackme.exe our protection like?

Run the file crackme.exe. Open LordPE we see a process crackme.exe

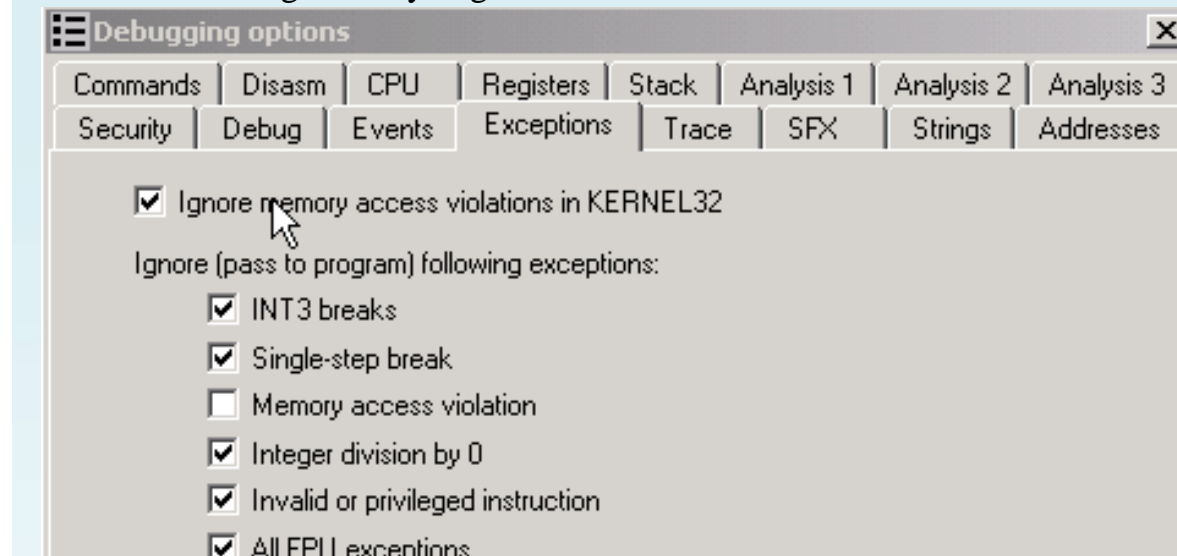


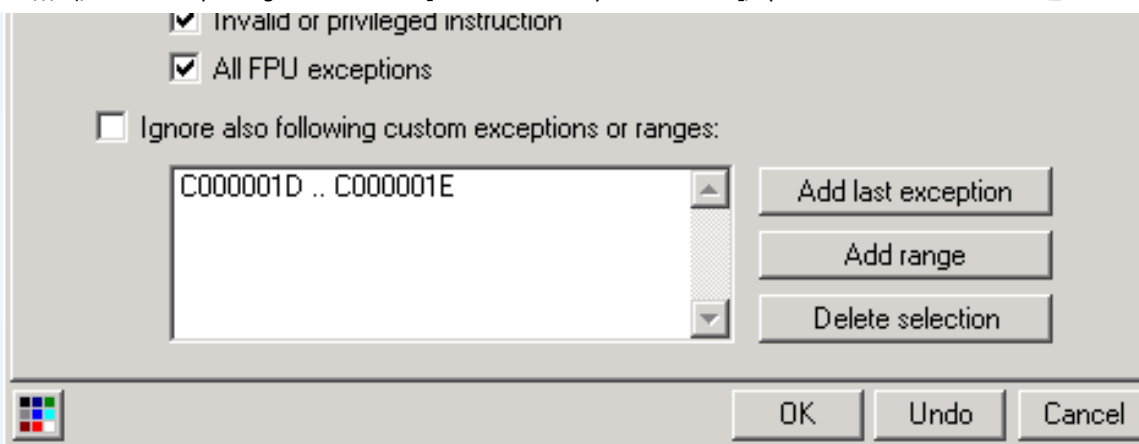
This signal is praiseworthy. Because it lets us know the program is not protected or CopyMemII Nanomites (this is very touching, the more tired).

We see the pack in the Armadillo. PEID But we know only the general like it. The problem is we want to know exactly which version Armadillo hacnho use.

I find the exact version of the Armadillo:

First refined setting for OllyDbg as follows:





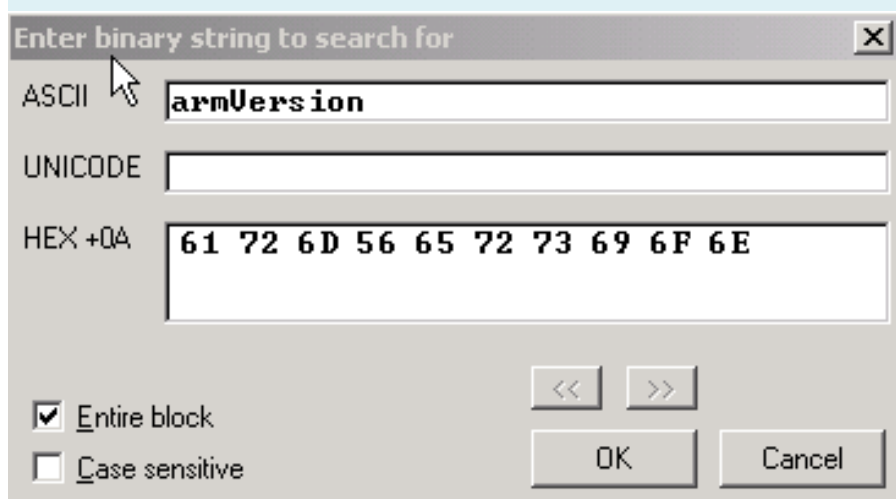
Then open crackme.exe. Press F9 (Run). The program will stop here.

00AC71E6	8900	MOV DWORD PTR DS:[EAX],EAX	
00AC71E8	90	NOP	
00AC71E9	E9 57010000	JMP 00AC7345	

Error follows:

Access violation when writing to [00000000] - use Shift+F7/F8/F9 to pass exception to program

Press Ctrl-B. Enter as follows:



Click OK. The program will stop. Click your mouse button to select Print Follow dump -> Selection.

00ACEF48	64:72 56	JB SHORT 00ACEFA1	Superfluous prefix	EIP 00AC71E6
00ACEF48	65:72 73	JB SHORT 00ACEFC1	Superfluous prefix	
00ACEF4E	696F 6E 3E0A00	IMUL EBP,DWORD PTR DS:[EDI+6E],20000A3E		
00ACEF55	2020	AND BYTE PTR DS:[EAX],AH		
00ACEF57	3C 61	CMP AL,61		
00ACEF59	72 6D	JB SHORT 00ACEFC8		
00ACEF5B	56	PUSH ESI		
00ACEF5C	65:72 73	JB SHORT 00ACEFD2		
00ACEF5F	696F 6E 207873	IMUL EBP,DWORD PTR DS:[EDI+6E]		
00ACEF66	3A7479 70	CMP DH,BYTE PTR DS:[ECX+EDI*2]		
00ACEF6A	65:3D 22787364	CMP EAX,64737822		
00ACEF70	3A73 74	CMP DH,BYTE PTR DS:[EBX+74]		
00ACEF73	72 69	JB SHORT 00ACEFDE		
00ACEF75	6E	OUTS DX,BYTE PTR ES:[EDI]		
00ACEF76	67:223E 2573	AND BH,BYTE PTR DS:[7325]		
00ACEF7B	3C 2F	CMP AL,2F		
00ACEF7D	61	POPAD		
00ACEF7E	72 6D	JB SHORT 00ACEFED		

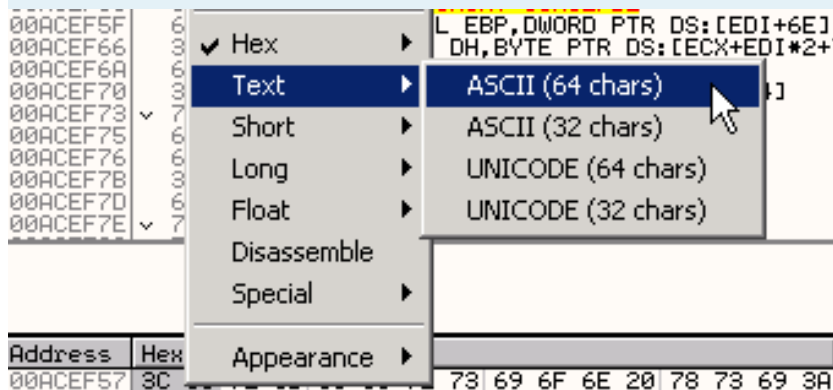
Address	Hex	dump
004AA000	B0 1B C7 77	E2 29 C7 77 89 28 C7 77 90 58 C7 77
004AA010	34 68 C7 77	3F 41 C7 77 0B 0C C7 77 0B 6D C7 77

Backup
Copy
Binary
Assemble
Label
Comment
Breakpoint
New origin here
Go to
Follow in Dump

Space
:
;
Ctrl+Gray *

Selection

In dump Windows to change the type Text viewpoint.



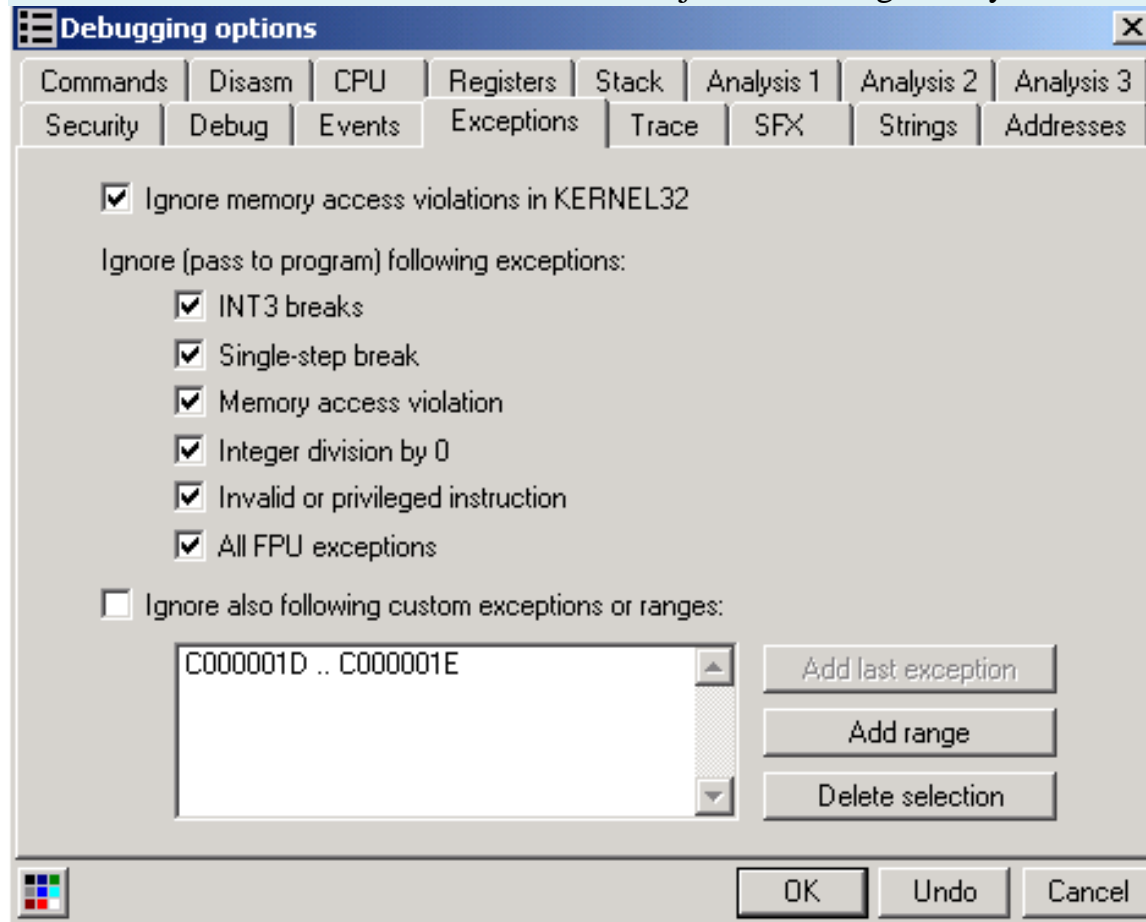
We found as follows:

Address	ASCII dump
00ACEF57	<armVersion xsi:type="xsd:string">%s</armVersion>....3.70a...
00ACEF97	<enhancedHardwareID xsi:type="xsd:string">%u</enhancedHardwareID>

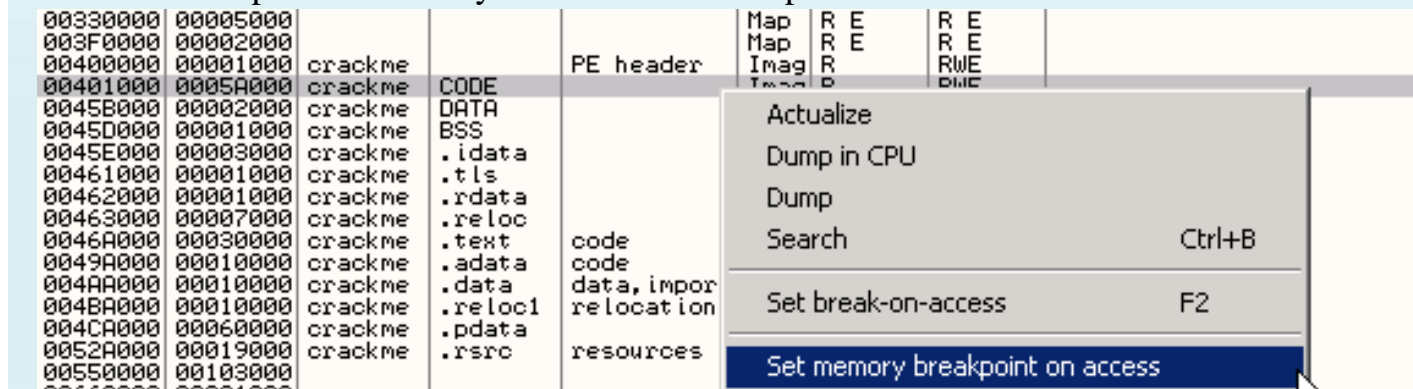
Note seats are yellow J We see that version 3.70a J

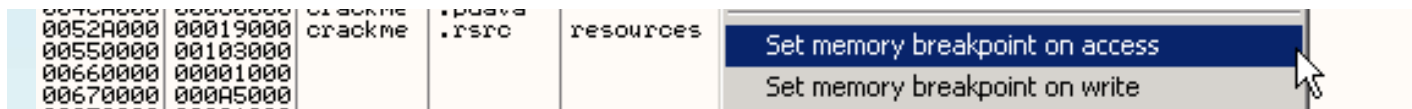
II. OEP Search:

Press Ctrl-F2 to load the file crackme.exe. Adjust the setting in Olly as follows:



Press Alt-M to open the Memory Window. Set breakpoint in the section on access code.





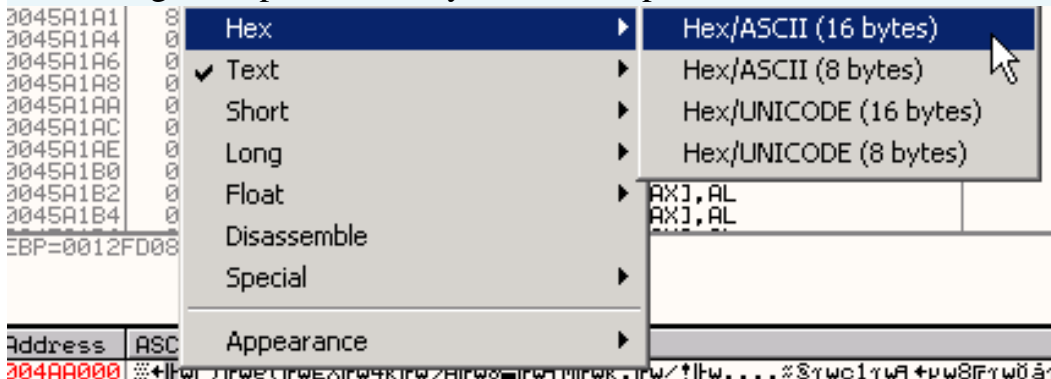
Close Window Memory. Press F9. The program will stop at 0045A15C.

0045A15C	55	PUSH EBP	
0045A15D	8BEC	MOV EBP,ESP	
0045A15F	83C4 F0	ADD ESP,-10	
0045A162	B8 3C9F4500	MOV EAX,crackme.00459F3C	
0045A167	E8 70B0FAFF	CALL crackme.00405EDC	
0045A16C	A1 80C54500	MOV EAX,DWORD PTR DS:[45C580]	
0045A171	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0045A173	E8 B891FFFF	CALL crackme.00453330	
0045A178	8B00 68C64500	MOV ECX,DWORD PTR DS:[45C668]	crackme.0045DC30
0045A17E	A1 80C54500	MOV EAX,DWORD PTR DS:[45C580]	

So: OEP = 0045A15C.

III. RVA Search Start and End RVA:

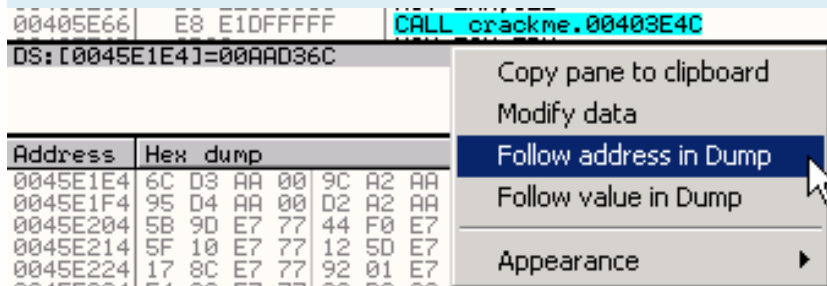
In exchange dump Window style Hex viewpoint.



At OEP trace with F7 to this place

00405E17	90	NOP	
00405E18	- FF25 E4E14500	JMP DWORD PTR DS:[45E1E4]	
00405E1E	8BC0	MOV EAX,EAX	
00405E20	- FF25 E0E14500	JMP DWORD PTR DS:[45E1E0]	kernel32.LocalI
00405E26	8BC0	MOV EAX,EAX	
00405E28	- FF25 DCE14500	JMP DWORD PTR DS:[45E1DC]	kernel32.TlsGet

We see 45E1E4. In the window a little click to select Address Follow in dump.



In Window dump we found as follows:

Address	Hex dump
0045E1E4	6C D3 AA 00 9C A2 AA 00 D7 23 DD 77 EA 22 DD 77
0045E1F4	95 D4 AA 00 D2 A2 AA 00 67 31 E7 77 A6 BC AA 00
0045E204	5B 9D E7 77 44 F0 E7 77 0A 98 E7 77 E6 1B E6 77
0045E214	5F 10 E7 77 12 5D E7 77 0A 98 E7 77 0A 98 E7 77
0045E224	17 8C E7 77 92 01 E7 77 0A 98 E7 77 0A 98 E7 77
0045E234	5F 10 E7 77 12 5D E7 77 0A 98 E7 77 0A 98 E7 77

Around which we have seen a lot 77. We will find the starting point and end of the RVA.

Roll up the window to see the signs of the 77 stops. We are.

0045E100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0045E110	00 00 00 00 00 00 00 00 75 32 F5 77 00 E3 F7 77u2Jw.T%w
0045E120	1F E2 F7 77 08 99 E7 77 34 9E E7 77 0A 98 E7 770rw4Rrw.0rw
0045E130	45 9A E7 77 81 98 E7 77 0A DE AA 00 C4 7C E7 77	E0rw00rw.0rw.0rw
0045E140	C5 78 E7 77 EF 77 E7 77 44 F0 E7 77 24 99 E7 77	+rw0rw0rw0rw0rw0rw
0045E150	CE 7C E7 77 72 46 E7 77 EF 3B E7 77 74 A7 AA 00	0rw0rw0rw0rw0rw0rw

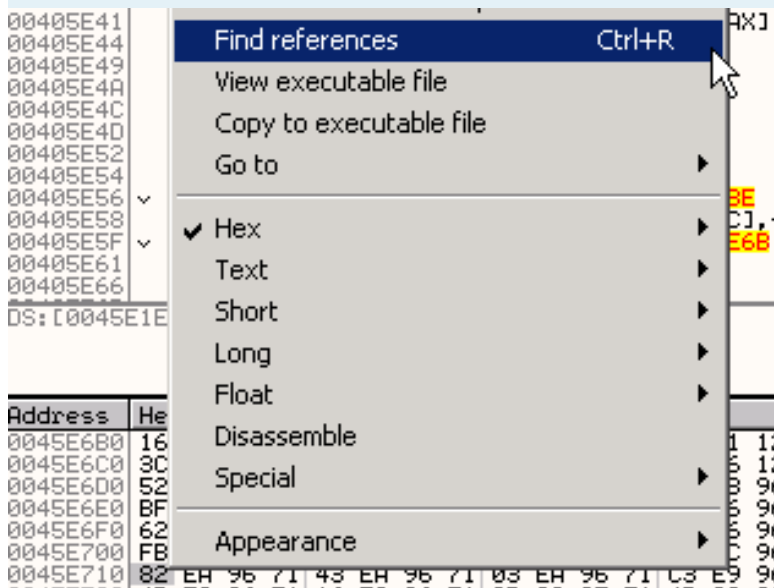
The attention you give me a bowl of gold starting point RVA address is 45E118.

Start RVA = 45E118

Now points to finish RVA. Scroll down the window dump also see signs of the 77 stops. We are.

Address	Hex dump	ASCII
0045E6B0	16 2E 12 77 92 2F 13 77 52 2F 13 77 5B 31 12 77	..+wE/!!wR/!!w[!+w
0045E6C0	3C 2B 12 77 82 34 12 77 1D 15 12 77 51 16 12 77	<+wE4+w#3+wQ..+w
0045E6D0	52 A3 AA 00 2C EE 96 71 EB ED 96 71 5D E8 96 71	Rû.,eûq\$ûq]ûûq
0045E6E0	BF 1F 97 71 86 E6 96 71 A2 E7 96 71 39 E6 96 71	γΨûqâpûqôγûq9pûq
0045E6F0	62 E7 96 71 84 E7 96 71 3C E7 96 71 B2 E6 96 71	brûqâγûq<γûqûûq
0045E700	FB E6 96 71 22 ED 96 71 81 EB 96 71 12 EC 96 71	γpûq"ûûqûûûq+ûûq
0045E710	82 EA 96 71 43 EA 96 71 03 EA 96 71 C3 E9 96 71	éûûqCûûqûûûq ûûq
0045E720	4B E9 96 71 16 E9 96 71 CB 28 97 71 4D A3 AA 00	Kûûq..ûûqûûq(ûûqMûû.
0045E730	6B 65 72 6E 65 6C 33 32 2E 64 6C 6C 00 00 00 00	kernel32.dll....

Now there is a problem we appear to see the number 71. For No. 71 which belonged RVA not we do the following: Choose an address that contains the number 71. Here I choose 45E710. Click the button to select Find References



A window appear

References in crackme:CODE to 0045E710..0045E710		
Address	Disassembly	Comment
00423CC0	JMP DWORD PTR DS:[45E710]	COMCTL32.ImageList_GetBkColor

We see here a function of ImageList_GetBkColor COMCTL32.DLL. So this is good value -> Select always 71.

Address	Hex dump	ASCII
0045E6D0	52 A3 AA 00 2C EE 96 71 EB ED 96 71 5D E8 96 71	Rû.,eûq\$ûq]ûûq
0045E6E0	BF 1F 97 71 86 E6 96 71 A2 E7 96 71 39 E6 96 71	γΨûqâpûqôγûq9pûq
0045E6F0	62 E7 96 71 84 E7 96 71 3C E7 96 71 B2 E6 96 71	brûqâγûq<γûqûûq
0045E700	FB E6 96 71 22 ED 96 71 81 EB 96 71 12 EC 96 71	γpûq"ûûqûûûq+ûûq
0045E710	82 EA 96 71 43 EA 96 71 03 EA 96 71 C3 E9 96 71	éûûqCûûqûûûq ûûq
0045E720	4B E9 96 71 16 E9 96 71 CB 28 97 71 4D A3 AA 00	Kûûq..ûûqûûq(ûûqMûû.
0045E730	6B 65 72 6E 65 6C 33 32 2E 64 6C 6C 00 00 00 00	kernel32.dll....

Note of gold was ended by RVA.

End = 45E72C RVA

Length = RVA End - Start RVA = 45E72C - 45E118 = 614 (hex)

IV. Find Magic Jump:

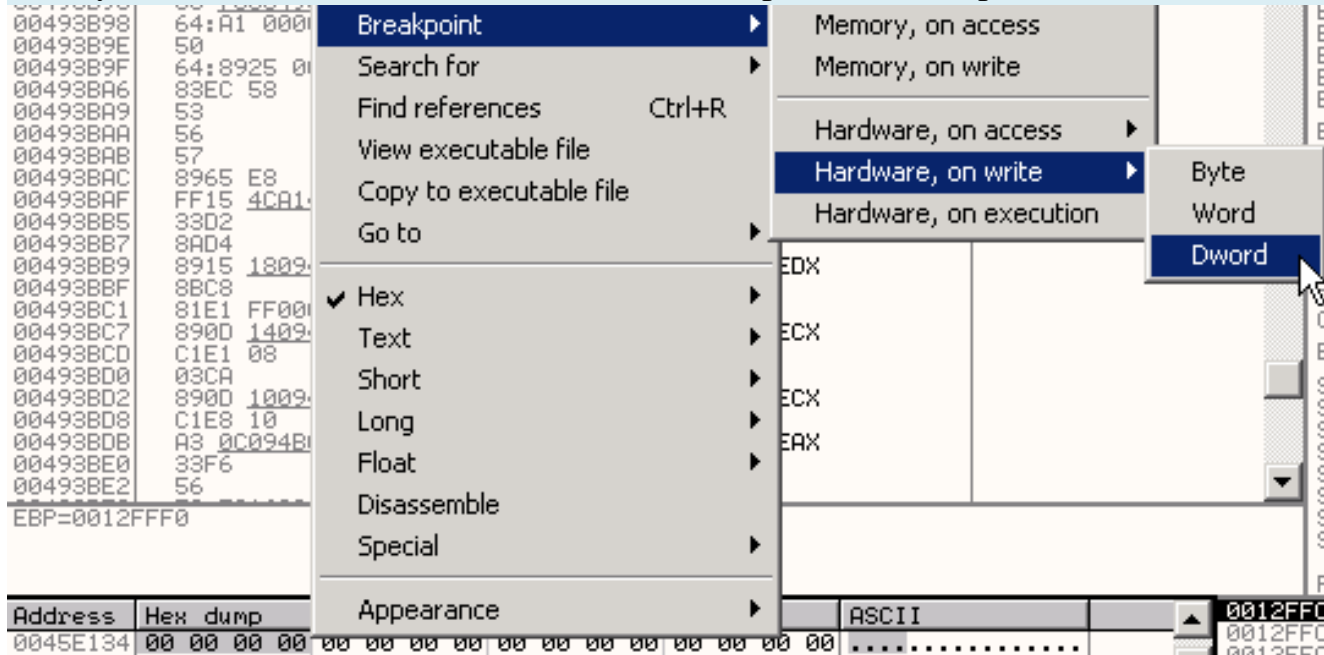
So Magic Jump ear is what? Speaking conduct na Magic Jump is set to decide that Armadillo will damage our IAT table or not. We will fix it so that it does not damage our table IAT.

What steps will be like? First dump in Windows press Ctrl-G. Enter the address begins by RVA: 45E118. From here one doubt them down for a billion bad IAT (which does not contain seats 77). We see 45E138 (to gold) is bad RVA. Note the value in 45E138 is **00AADE0A**

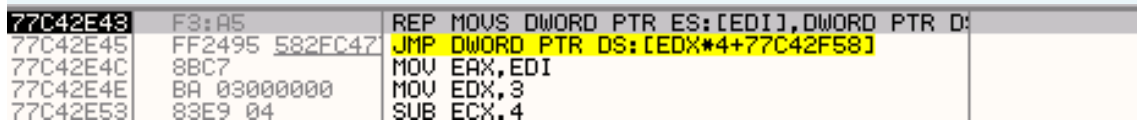
0045E118	75 32 F5 77 00 E3 F7 77 1F E2 F7 77 08 99 E7 77	u2Jw..T#wΨΓ#wûûq
0045E128	34 9E E7 77 0A 98 E7 77 45 9A E7 77 81 98 E7 77	4Rγw.γγwEûγwûûγγw
0045E138	0A DE AA 00 C4 7C E7 77 C5 78 E7 77 EF 77 E7 77	.R.-!γw+γγwûûγγw
0045E148	44 F0 E7 77 24 99 E7 77 CE 7C E7 77 72 46 E7 77	D=γw\$ûγwûûγγwγwFγw

We will get a first 45E138 address is **45E134** (to green).

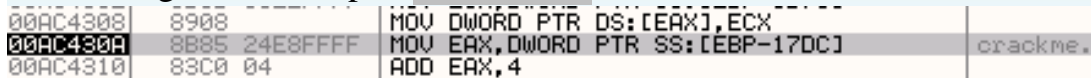
Press Ctrl-F2 to open the file crackme.exe. In dump Windows press Ctrl-G. Enter **45E134**. Select 4 bytes. Click your mouse button to set the Hardware Breakpoint as in the picture.



Press F9. Program stops here.

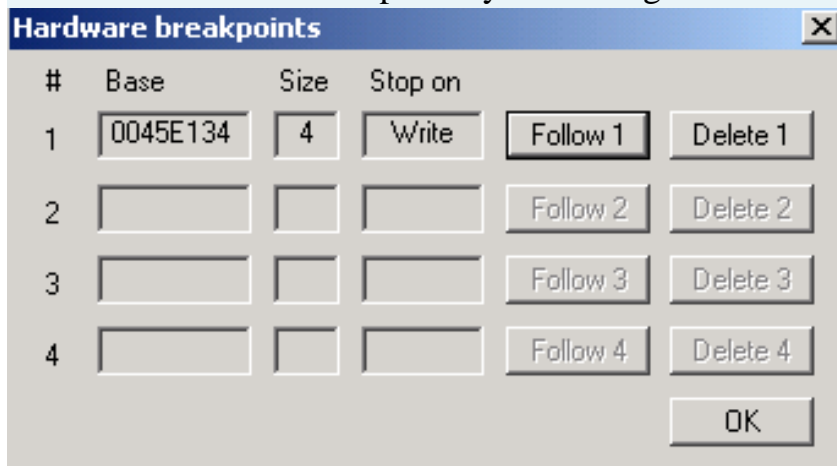


Press F9 again. The stop in 00AC430A.



In a line we see the value of ECX is to IAT.

Remove Hardware Breakpoint by the Debug menu -> Hardware Breakpoints



Click Delete to delete 1 Hardware Breakpoints. Click OK.

We will trace to find the Magic Jump. Trace the F8. You note that JE, JNZ. You should write a paper with JE, JNZ that it does not jump or dance.

00AC4059 JE 00AC431E not jump

00AC4080 JNZ SHORT 00AC40C6 Jumping

00AC40D4 JNZ 00AC4164 Jumping

00AC408E JE SHORT 00AC4201 not jump

00AC41B7 JE SHORT 00AC4201 not jump

00AC41EC JNZ SHORT 00AC41FF Jumping

Here we will fall into the loop. Set Breakpoint at 00AC4201 and press F9 to exit the loop. We will in 00AC4201.

00AC41EA	85C0	TEST EAX,EAX	
00AC41EC	75 11	JNZ SHORT 00AC41FF	
00AC41EE	8B85 7CE2FFFF	MOV EAX,DWORD PTR SS:[EBP-1D84]	
00AC41F4	8B40 08	MOV EAX,DWORD PTR DS:[EAX+8]	
00AC41F7	8985 88E2FFFF	MOV DWORD PTR SS:[EBP-1D78],EAX	
00AC41FD	EB 02	JMP SHORT 00AC4201	
00AC41FF	EB 9D	JMP SHORT 00AC419E	
00AC4201	83BD 88E2FFFF	CMP DWORD PTR SS:[EBP-1D78],0	
00AC4208	75 3F	JNZ SHORT 00AC4249	

Continue to trace F8 and more notes.

00AC4208 JNZ SHORT 00AC4249 NHAY

00AC4250 JNZ 00AC42EE NHAY

Trace 00AC430A to stop it. Start from here to trace the F8 (no need to write paper) and compare the orders jumped to the record we have seen in the changes do not. Here I record for easy tracking.

00AC4059 JE 00AC431E not jump

00AC4080 JNZ SHORT 00AC40C6 Jumping

00AC40D4 JNZ 00AC4164 Jumping

00AC408E JE SHORT 00AC4201 not jump

00AC41B7 JE SHORT 00AC4201 not jump

00AC41EC JNZ SHORT 00AC41FF Jumping

Also set breakpoint at 00AC4201 and press F9 to exit the loop. To trace

00AC4208 JNZ SHORT 00AC4249 unwinking >>>>> Note: This on the other.

So is Magic Jump 00AC4208. We will correct

00AC4208 JNZ SHORT 00AC4249

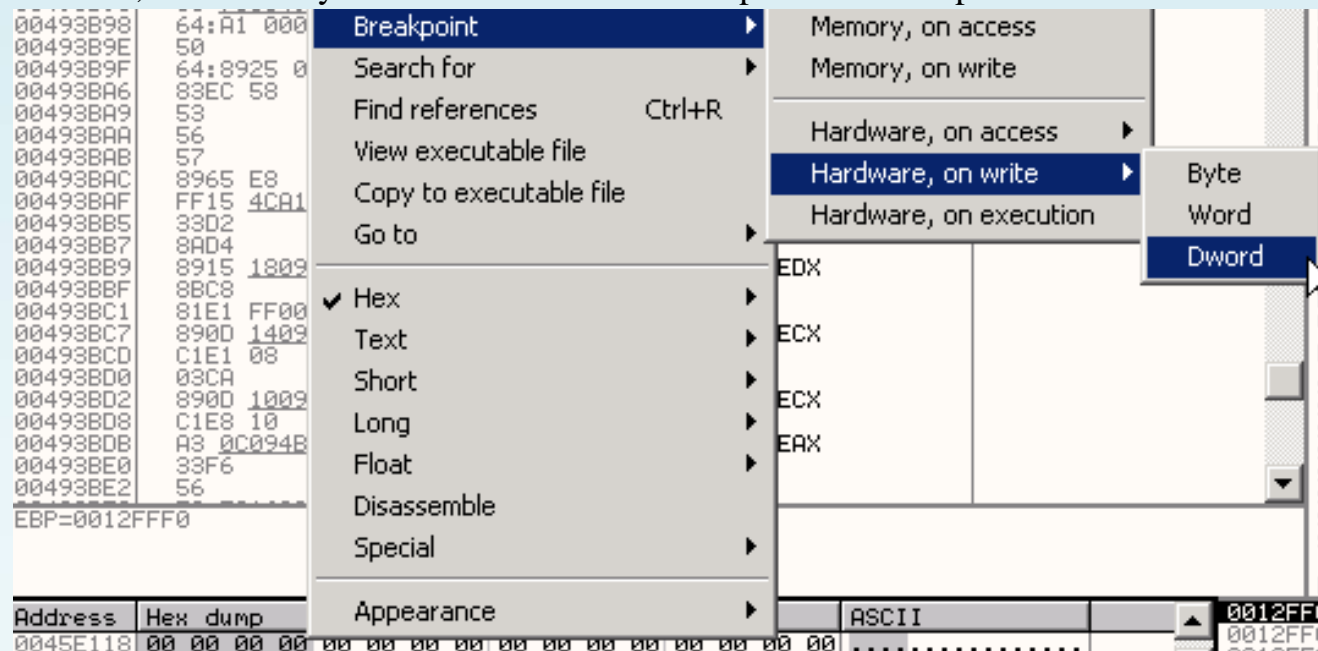
to

00AC4208 NOP

V. Fix Magic Jump, dump and IAT Fix:

Press Ctrl-F2 to reboot crackme.exe. In dump Windows press Ctrl-G. Enter the address start RVA:

0045E118, marked 4 bytes and set Hardware Breakpoint as in the picture



Press F9. Program stops here.

77C42E43	F3:A5	REP MOVSDWORD PTR ES:[EDI],DWORD PTR D:
77C42E45	FF2495 582EC47	JMP DWORD PTR DS:[EDX*4+77C42F58]
77C42E4C	8BC7	MOV EAX,EDI

F9 again stop here.

00AC430A	8B85 24E8FFFF	MOV EAX,DWORD PTR SS:[EBP-17DC]	crackme.
00AC4310	83C0 04	ADD EAX,4	
00AC4313	8985 24E8FFFF	MOV DWORD PTR SS:[EBP-17DC],EAX	
00AC4319	^ E9 36FDFFFF	JMP 00AC4054	

To this we will edit Magic Jump. Press Ctrl-G. Enter the address Magic Jump.

Enter expression to follow

00AC4208

OK Cancel

Click OK. Edit 00AC4208 to NOP submitted. Note 2 bytes to do.

00AC41FD	^ EB 02	JMP SHORT 00AC4201	
00AC41FF	EB 9D	JMP SHORT 00AC419E	
00AC4201	83BD 88E2FFFF	CMP DWORD PTR SS:[EBP-1D78],0	
00AC4208	90	NOP	
00AC4209	90	NOP	
00AC420A	0FB785 8CE2FFF	MOVZX EAX,WORD PTR SS:[EBP-1D74]	

Press F9 and the crash? ;)

00AC3728	1890 6D7725CB	SBB BYTE PTR DS:[EAX+CB25776D],DL	
00AC372E	^ 75 9F	JNZ SHORT 00AC36CF	
00AC3730	FD	STD	
00AC3731	68 74C26BF3	PUSH F36BC274	
00AC3736	8EB8 0E387016	MOV SEG?,WORD PTR DS:[EAX+1670380E]	Undefin

We expect the program to run normally with IAT is intact. But the crash was. So how to fix IAT and dump? Speaking as the song of Musicians Tuan Khanh is: Small?

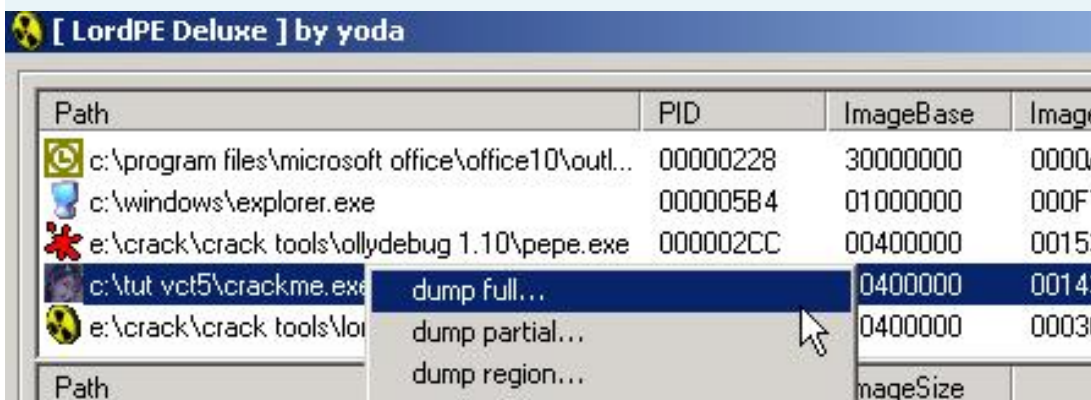
You pay attention, although the crash was the IAT we still intact.

You open up a OllyDbg other (called the Olly2 and we are open before Olly1) and open the file in Olly2 crackme.exe. Set breakpoint on memory access in section CODE to OEP (if not understand the review above). Press F9. In Olly2 we are in OEP 0045A15C. Now we will copy the original IAT in Olly1 bring to Olly2.

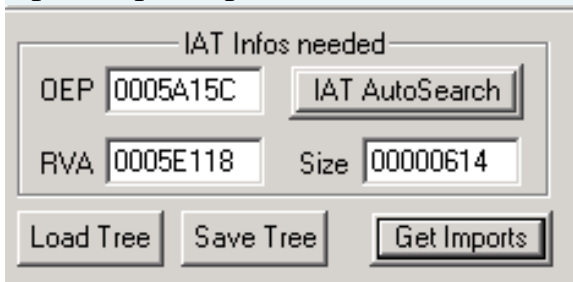
In Olly1 mark from the original starting point for RVA points to end RVA (from 0045E118 to 0045E72C)

0045E118	75 32 F5 77 00 E3 F7 77 1F E2 F7 77 08 99 E7 77	u2Jw.π#w▼Γ#w■0γw
0045E128	34 9E E7 77 0A 98 E7 77 45 9A E7 77 81 98 E7 77	4Rγw.γγwEγwūγγw
0045E138	86 C4 E7 77 C4 7C E7 77 C5 78 E7 77 EF 77 E7 77	8-γw-!γw+γγwnγγw
0045E148	44 F0 E7 77 24 99 E7 77 CE 7C E7 77 72 46 E7 77	D=γw\$0γwγf!γwγFγw
0045E158	EF 38 E7 77 B8 05 E8 77 21 7F E7 77 7A 17 E6 77	n:γwγ*§w!0γwz§γw
0045E168	FD A5 E7 77 93 9F E7 77 99 A0 E7 77 3C 51 E7 77	*Nγw0fγw0āγw<Qγw
0045E178	38 C9 E7 77 18 06 E8 77 9E 5D E7 77 AA 8E E7 77	8fγw↑*§wR]γw-āγw
0045E6B8	52 2F 13 77 5B 31 12 77 3C 2B 12 77 82 34 12 77	R/!!w[1#w<+§we4#w
0045E6C8	1D 15 12 77 51 16 12 77 52 A3 AA 00 2C EE 96 71	#§#wQ_#wRū_.,eūq
0045E6D8	EB ED 96 71 5D E8 96 71 BF 1F 97 71 86 E6 96 71	§φūq]§ūqγūqāpūq
0045E6E8	A2 E7 96 71 39 E6 96 71 62 E7 96 71 84 E7 96 71	ōγūq9pūqbrūqāγūq
0045E6F8	3C E7 96 71 B2 E6 96 71 FB E6 96 71 22 ED 96 71	<γūq§pūqγpūq"φūq
0045E708	81 EB 96 71 12 EC 96 71 82 EA 96 71 43 EA 96 71	ū§ūq#φūqēūqCūūq
0045E718	03 EA 96 71 C3 E9 96 71 4B E9 96 71 16 E9 96 71	*ūūq†θūqKθūq_θūq
0045E728	CB 28 97 71 4D A3 AA 00 6B 65 72 6E 65 6C 33 32	π(ūqHū_·kernel32

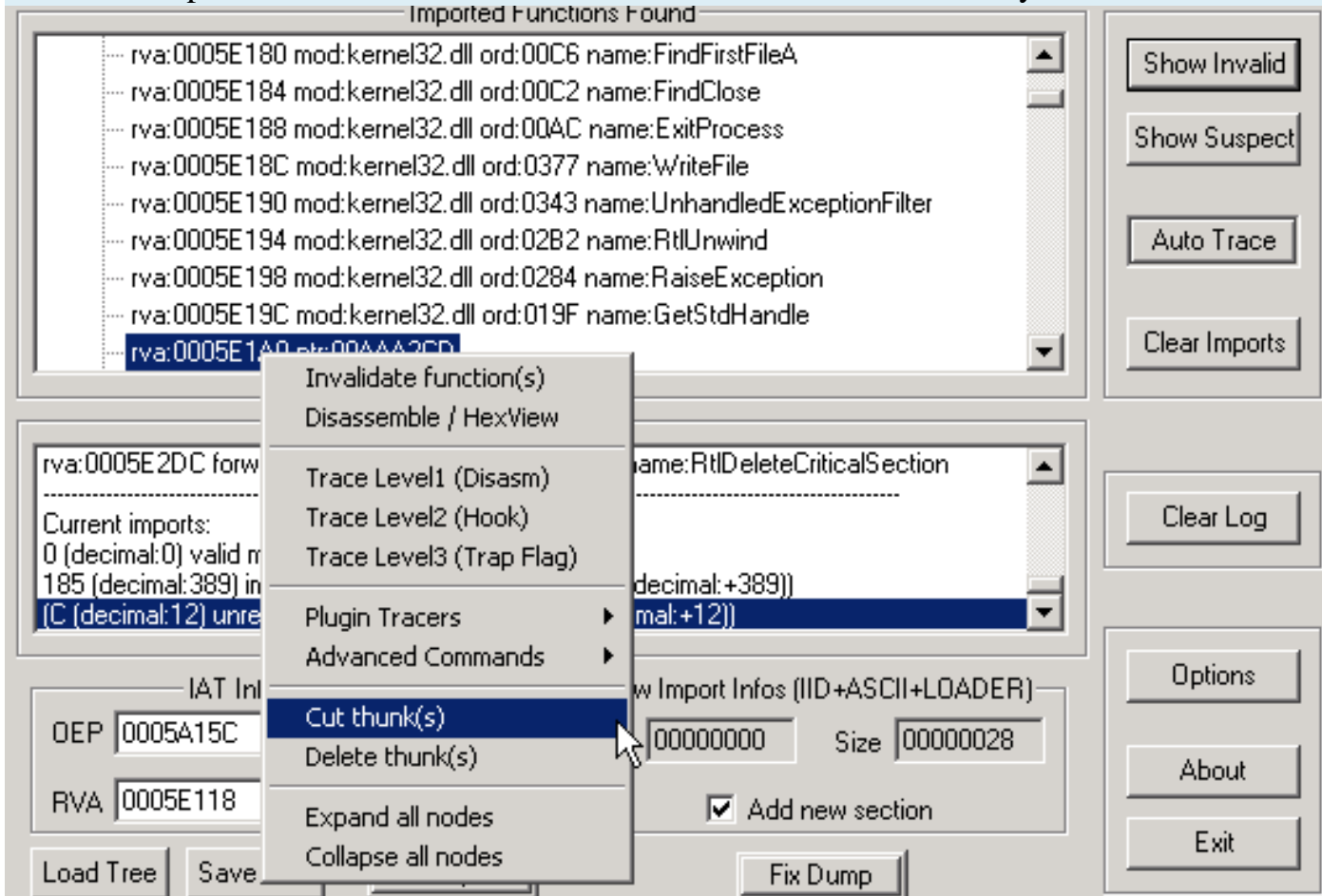
Click your mouse to select Binary -> Binary copy



Open Imprec up. Select crackme.exe. Fill in the parameters of the following:

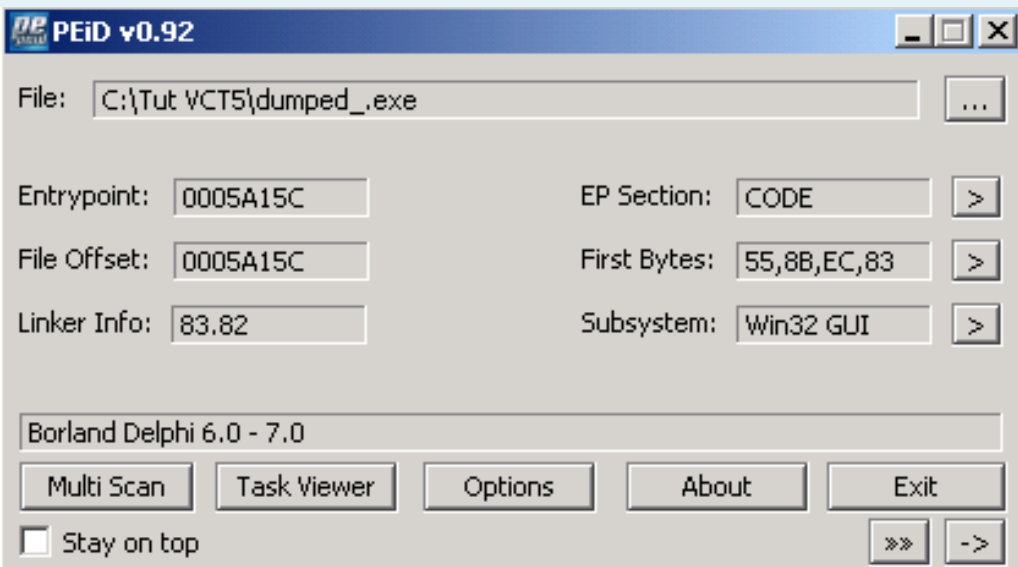


Click Get Imports. Invalid still some Thunks. Click Show Invalid. Click your mouse to select Cut Thunks.



IAT Now we have full. Click Fix selected dump file dump.exe. Dump_.exe file will be created. Test file dump_.exe J Good!

Using PEID review.



Bonus) Check Enable Button:

When we run and press crackme Check not find work. In rule.txt attached crackme we find the task to Check Enable button.

We will use ExeScope to perform tasks seems like this complex.

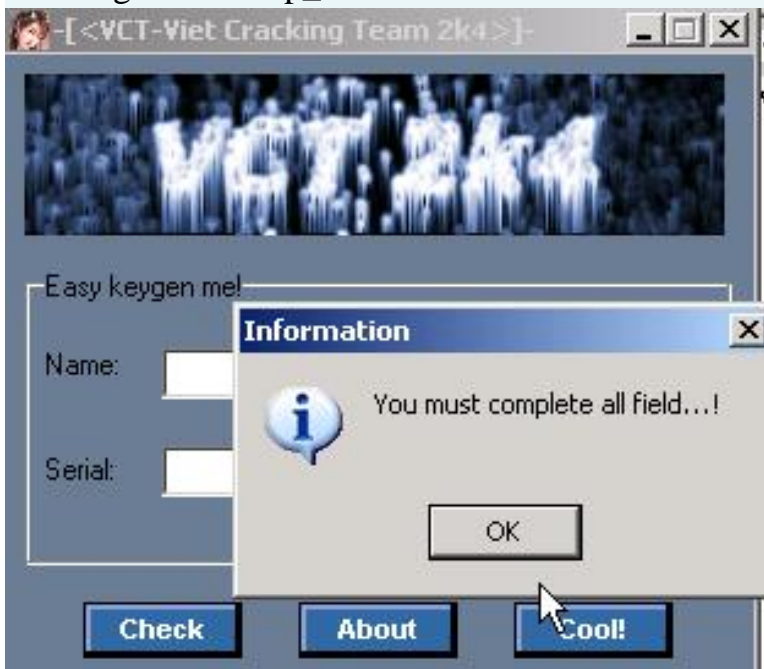
Using ExeScope open file dump_.exe. In the Resource -> RCDATA -> TForm1 we found as follows:

<ul style="list-style-type: none"> + Header + Import - Resource <ul style="list-style-type: none"> + Bitmap + Dialog + String - RCDATA <ul style="list-style-type: none"> DVCLAL PACKAGEINFO TFORM1 + Cursor + Icon + XPManifest 	<pre> ClientHeight = 254 ClientWidth = 306 Color = 9469549 Font.Charset = DEFAULT_CHARSET Font.Color = clWindowText Font.Height = -11 Font.Name = 'MS Sans Serif' Font.Style = [] OldCreateOrder = False PixelsPerInch = 96 TextHeight = 13 object AOLButton1: TAOLButton Left = 32 Top = 224 Width = 65 Height = 23 Caption = 'Check' Color = 10840372 RaiseColor = 14131850 Enabled = False OnClick = AOLButton1Click ShowHint = True ParentShowHint = False Font.Charset = DEFAULT_CHARSET Font.Color = clWhite Font.Height = -11 Font.Name = 'Arial' Font.Style = [fsBold] end </pre>
--	---

You note of gold. We will be revised to True. Save the file and close ExeScope.


```
object AOLButton1: TAOLOButton
  Left = 32
  Top = 224
  Width = 65
  Height = 23
  Caption = 'Check'
  Color = 10840372
  RaiseColor = 14131850
  Enabled = True
  OnClick = AOLButton1Click
  ShowHint = True
  ParentShowHint = False
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWhite
  Font.Height = -11
  Font.Name = 'Arial'
  Font.Style = [fsBold]
end
```

Running back dump_.exe. Click Check.



Them. Happy happy.
tlandn

<u>slayer</u>	<u>Various</u> <u>Asprotect</u> <u>Loader</u> <u>Tricks</u>	<u>Snd</u>
<u>takada</u> REA	Tools you need:	<u>REA</u>
	OllyDbg, ABEL, PELG, DUP	

Difficult: easy

Instruction

Hi everyone =)

Today we will learn tricks of the tool to create different loader, which can be used to create loader against asprotect. Target I use will be introduced below, (after understand you can also use other target to practice):)

LETS GO! = D

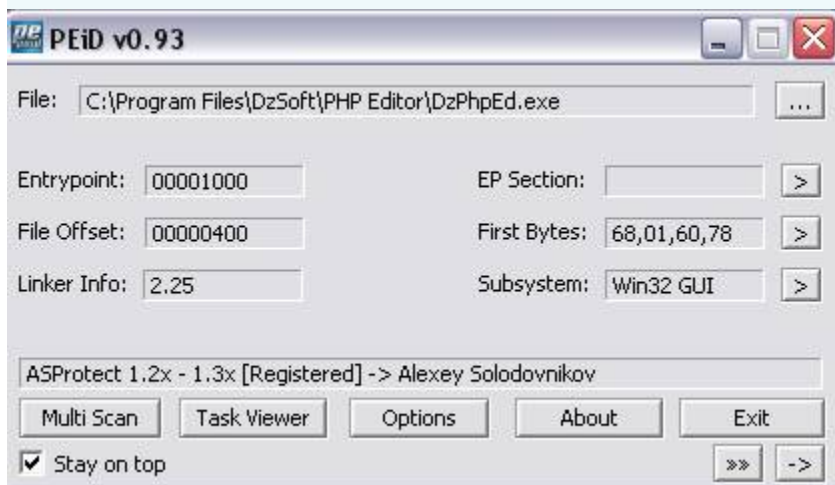
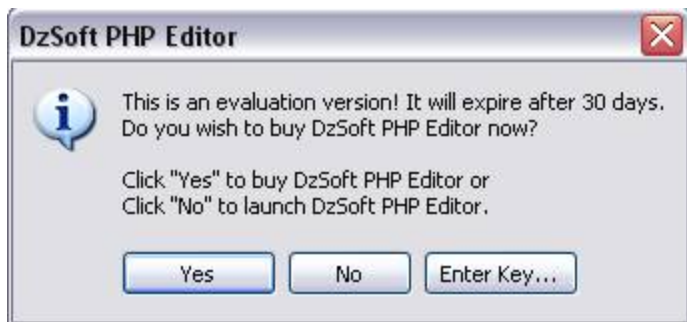
How to crack

Asprotect have CRC check, so the normal 1 loader can do is work :-(but we can use 1 features other loader to help us see them as nhuung features gi. Trick first, I will you only use the option "Window Caption" options in ABEL (a loader generator with many features are excellent). Target will be selected as DzSoft PHP Editor = 3.6)

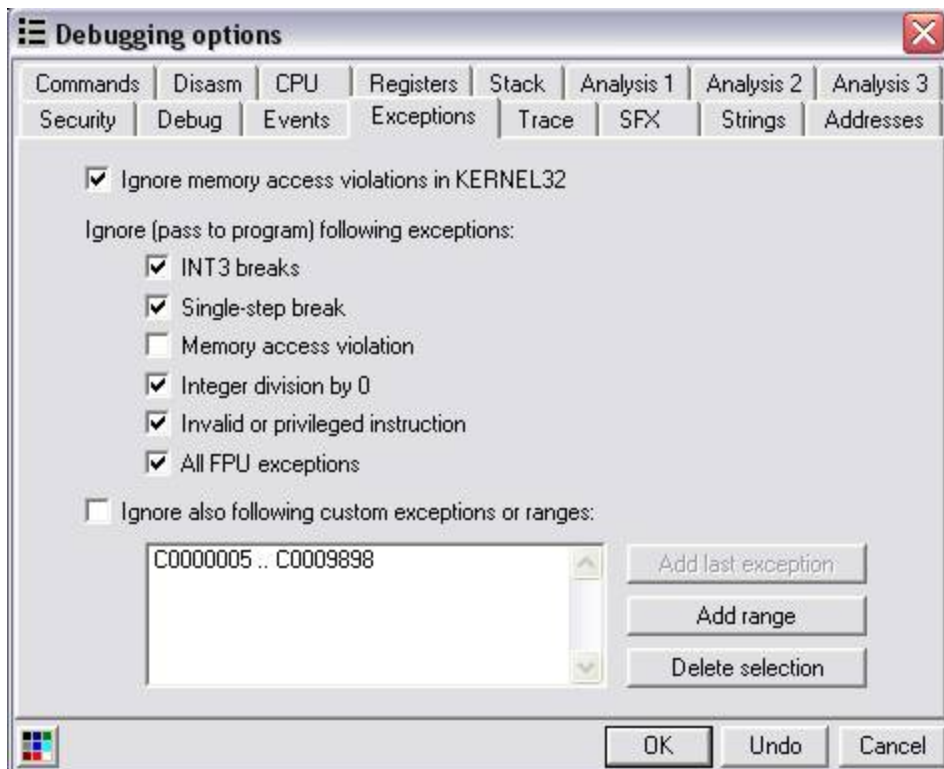
Ok, many people asked me that how to set hardware breakpoints 1 working in asprotect appz. This is the answer. To crack any aspr appz trial, any limitations, we must also break for Olly there, so we need to code in the program and then set by BP, the BP will work. Do it like?.

Simple. We will seek exception to the final, looked up to see 1 RETN command set in which BP then open memory map and set membp 1 (F2) in the code section, then press F9, Olly will break at OEP =). Sometimes i will break the OEP but I sure it will break the code in the program =)

Now they look to target our



30 day trial and more promotional aspr =). Load target to Olly.
Set exceptions like image below



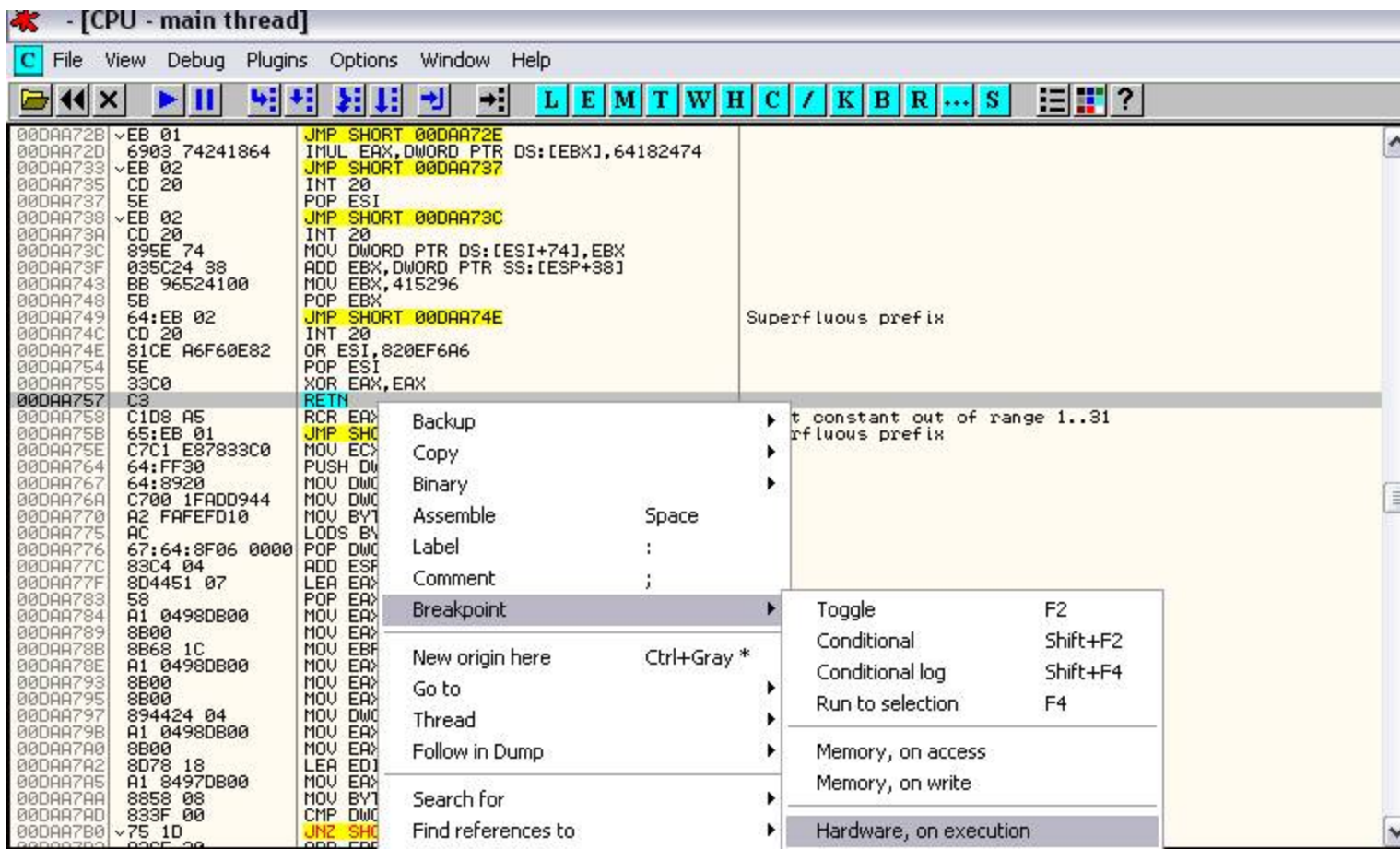
Address	Hex	Mnemonic	Comment
00401000	68 01607800	PUSH DzPhpEd.00786001	
00401005	E8 01000000	CALL DzPhpEd.0040100B	
0040100A	C3	RETN	
0040100B	C3	RETN	
0040100C	86A2 3E56D215	XCHG BYTE PTR DS:[EDX+15D2563E],AH	
00401012	75 74	JNZ SHORT DzPhpEd.00401088	
00401014	082D 06E7B73F	OR BYTE PTR DS:[3FB7E7D6],CH	
0040101A	23147A	AND EDX,DWORD PTR DS:[EDX+EDI*2]	
0040101D	DAD4	FCMOVB ST,ST(4)	
0040101F	69C1 7A2EE504	IMUL EAX,ECX,4E52E7A	
00401025	E6 68	OUT 68,AL	I/O command
00401027	8F	??	Unknown command
00401028	5F	POP EDI	
00401029	76 11	JBE SHORT DzPhpEd.0040103C	
0040102B	6C	INS BYTE PTR ES:[EDI],DX	I/O command
0040102C	A3 E384825C	MOV DWORD PTR DS:[5C8284E3],EAX	
00401031	A4	MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	
00401032	5D	POP EBP	
00401033	32A5 2B058583	XOR AH,BYTE PTR SS:[EBP+8385052B]	
00401039	F6BF 0D66839B	IDIV BYTE PTR DS:[EDI+9B83660D]	
0040103F	25 2FD979C6	AND EAX,C679D92F	
00401044	299456 5D59CFD6	SUB DWORD PTR DS:[ESI+EDX*2+D6CF595D],EAX	
0040104B	CA 09DC	RETF 0DC09	Far return
0040104E	B1 44	MOV CL,44	
00401050	D032	SAL BYTE PTR DS:[EDX],1	
00401052	F763 E5	MUL DWORD PTR DS:[EBX-1B]	
00401055	-66:7D 9E	JGE SHORT 00000FF6	
00401058	FB	STI	
00401059	8E3B	MOV SEG?,WORD PTR DS:[EBX]	Undefined segment register
0040105B	FD	STD	
0040105C	64:8013 31	ADC BYTE PTR FS:[EBX],31	
00401060	94	XCHG EAX,ESP	
00401061	ED	IN EAX,DX	I/O command
00401062	52	PUSH EDX	
00401063	56	PUSH ESI	
00401064	A4	MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]	
00401065	2D E2676303	SUB EAX,36367E2	
0040106A	94	XCHG EAX,ESP	
0040106B	F9	STC	
0040106C	57	PUSH EDI	
0040106D	C58A 93B90DE0	LDS ECX,FWORD PTR DS:[EDX+E00DB993]	Modification of segment register
00401073	64:63B1 254DF261	ARPL WORD PTR FS:[ECX+65F24D25],SI	
0040107A	6296 B6D163AA	BOUND EDX,QWORD PTR DS:[ESI+AA63D1B6]	

We are at EP. Now, press Shift-F9 continued to run the program. Back to the code window of Olly, Olly you will see constantly in the final exception;). Just so

```

[CPU - main thread]
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
00DAA76A C700 1FADD944 MOV DWORD PTR DS:[EAX],44D9AD1F
00DAA770 A2 FAFED10 MOV BYTE PTR DS:[10FDFEFA],AL
00DAA775 AC LODS BYTE PTR DS:[ESI]
00DAA776 67:64:8F06 0000 POP DWORD PTR FS:[0]
00DAA77C 83C4 04 ADD ESP,4
00DAA77F 8D4451 07 LEA EAX,DWORD PTR DS:[ECX+EDX*2+7]
00DAA783 58 POP EAX
00DAA784 A1 0498DB00 MOV EAX,DWORD PTR DS:[DB9804]
00DAA789 8B00 MOV EAX,DWORD PTR DS:[EAX]
00DAA78B 8B68 1C MOV EBP,DWORD PTR DS:[EAX+1C]
00DAA78E A1 0498DB00 MOV EAX,DWORD PTR DS:[DB9804]
00DAA793 8B00 MOV EAX,DWORD PTR DS:[EAX]
00DAA795 8B00 MOV EAX,DWORD PTR DS:[EAX]
00DAA797 894424 04 MOV DWORD PTR SS:[ESP+4],EAX
00DAA79B A1 0498DB00 MOV EAX,DWORD PTR DS:[DB9804]
00DAA7A0 8B00 MOV EAX,DWORD PTR DS:[EAX]
00DAA7A2 8D78 18 LEA EDI,DWORD PTR DS:[EAX+18]
00DAA7A5 A1 8497DB00 MOV EAX,DWORD PTR DS:[DB9784]
00DAA7AA 8B58 08 MOV BYTE PTR DS:[EAX+8],BL
00DAA7AD 833F 00 CMP DWORD PTR DS:[EDI],0
00DAA7B0 75 10 JNZ SHORT 00DAA7CF
00DAA7B2 83C5 20 ADD EBP,20
00DAA7B5 A1 7096DB00 MOV EAX,DWORD PTR DS:[DB9670]
00DAA7BA 8078 09 00 CMP BYTE PTR DS:[EAX+9],0
00DAA7BE 75 0F JNZ SHORT 00DAA7CF
00DAA7C0 B8 1F000000 MOV EAX,1F
00DAA7C5 E8 0680FEFF CALL 000927D0
00DAA7CA C1E0 02 SHL EAX,2
00DAA7CD 2BE8 SUB EBP,EAX
00DAA7CF E8 10CEFFFF CALL 00DA75E4
00DAA7D4 8BD8 MOV EBX,EAX
00DAA7D6 833D 04B4DB00 0 CMP DWORD PTR DS:[DBB404],0
00DAA7DD 74 15 JE SHORT 00DAA7F4
00DAA7DF 6A 04 PUSH 4
00DAA7E1 B9 04B4DB00 MOV ECX,0DBB404
00DAA7E6 8D4424 04 LEA EAX,DWORD PTR SS:[ESP+4]
00DAA7EA BA 04000000 MOV EDX,4
00DAA7EF E8 2C70FFFF CALL 00DA1820
00DAA7F4 833D 34B4DB00 0 CMP DWORD PTR DS:[DBB434],0
00DAA7FB 74 15 JE SHORT 00DAA812
00DAA7FD 6A 0C PUSH 0C
00DAA7FF B9 34B4DB00 MOV ECX,0DBB434
00DAA804 8D4424 04 LEA EAX,DWORD PTR SS:[ESP+4]
  
```

Scroll up a little and you will see 1 order RETN



Set a hardware bp on execution like image. Olly restart the program. Now you can check all the exceptions in Olly, we not need them. We stopped in the EP's packer
Press Shift-F9 to run program and we will break the hardware breakpoint was set;)

[CPU - main thread]

File View Debug Plugins Options Window Help

L E M T W H C / K B R ... S

00DAA757	C3	RETN	
00DAA758	C108 A5	RCR EAX,0A5	Shift constant out of range 1..31
00DAA75B	65:EB 01	JMP SHORT 00DAA75F	Superfluous prefix
00DAA75E	C7C1 E87833C0	MOV ECX,C03378E8	
00DAA764	64:FF30	PUSH DWORD PTR FS:[EAX]	
00DAA767	64:8920	MOV DWORD PTR FS:[EAX],ESP	
00DAA76A	C700 1FADD944	MOV DWORD PTR DS:[EAX],44D9AD1F	
00DAA770	A2 FAFED10	MOV BYTE PTR DS:[10FDFEFA],AL	
00DAA775	AC	LODS BYTE PTR DS:[ESI]	
00DAA776	67:64:8F06 0000	POP DWORD PTR FS:[0]	
00DAA77C	83C4 04	ADD ESP,4	
00DAA77F	8D4451 07	LEA EAX,DWORD PTR DS:[ECX+EDX*2+7]	
00DAA783	58	POP EAX	
00DAA784	A1 0498DB00	MOV EAX,DWORD PTR DS:[DB9804]	
00DAA789	8B00	MOV EAX,DWORD PTR DS:[EAX]	
00DAA78B	8B68 1C	MOV EBP,DWORD PTR DS:[EAX+1C]	
00DAA78E	A1 0498DB00	MOV EAX,DWORD PTR DS:[DB9804]	
00DAA793	8B00	MOV EAX,DWORD PTR DS:[EAX]	
00DAA795	8B00	MOV EAX,DWORD PTR DS:[EAX]	
00DAA797	894424 04	MOV DWORD PTR SS:[ESP+4],EAX	
00DAA79B	A1 0498DB00	MOV EAX,DWORD PTR DS:[DB9804]	
00DAA7A0	8B00	MOV EAX,DWORD PTR DS:[EAX]	
00DAA7A2	8D78 18	LEA EDI,DWORD PTR DS:[EAX+18]	
00DAA7A5	A1 8497DB00	MOV EAX,DWORD PTR DS:[DB9784]	
00DAA7AA	8B58 08	MOV BYTE PTR DS:[EAX+8],BL	
00DAA7AD	833F 00	CMP DWORD PTR DS:[EDI],0	
00DAA7B0	75 10	JNZ SHORT 00DAA7CF	
00DAA7B2	83C5 20	ADD EBP,20	
00DAA7B5	A1 7096DB00	MOV EAX,DWORD PTR DS:[DB9670]	
00DAA7BA	8078 09 00	CMP BYTE PTR DS:[EAX+9],0	
00DAA7BE	75 0F	JNZ SHORT 00DAA7CF	
00DAA7C0	B8 1F000000	MOV EAX,1F	
00DAA7C5	E8 0680FEFF	CALL 00D927D0	
00DAA7CA	C1E0 02	SHL EAX,2	
00DAA7CD	2BE8	SUB EBP,EAX	
00DAA7CF	E8 10CEFFFF	CALL 00DA75E4	
00DAA7D4	8BD8	MOV EBX,EAX	
00DAA7D6	833D 04B4DB00 0	CMP DWORD PTR DS:[DBB404],0	
00DAA7DD	74 15	JE SHORT 00DAA7F4	
00DAA7DF	6A 04	PUSH 4	
00DAA7E1	B9 04B4DB00	MOV ECX,0DBB404	
00DAA7E6	8D4424 04	LEA EAX,DWORD PTR SS:[ESP+4]	
00DAA7EA	BA 04000000	MOV EDX,4	
00DAA7F5	EB 0670FFFF	CALL 00D927D0	

Return to 7C9037BF (ntdll.7C9037BF)


Address	Hex dump	ASCII
00642000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00@i@.
00642010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00@i@.
00642020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00@i@.
00642030	00 8D 40 00 00 8D 40 00 01 8D 40 00 00 00 00 00	...@i@.i@.
00642040	00 00 00 00 00 88 22 40 00 48 24 40 00 C8 27 40 00	...@.H\$@.r@.
00642050	00 C8 CC C8 C9 D7 CF C8 CD CE DB D8 DA D9 CA DC	..@...@...@...
00642060	DD DE DF E0 E1 E3 00 E4 E5 8D 40 00 03 00 00 00	...@...@...@...
00642070	00 00 00 00 01 00 00 00 02 00 00 00 03 00 00 00	...@...@...@...
00642080	00 00 00 00 00 00 00 00 00 00 00 00 45 72 72 6FError
00642090	72 00 8B C0 52 75 6E 74 69 6D 65 20 65 72 72 6F	r.iRuntime error
006420A0	72 20 20 20 20 20 61 74 20 30 30 30 30 30 30 30	r.at 0000000
006420B0	30 00 8B C0 30 31 32 33 34 35 36 37 38 39 41 42	0.i40123456789AB
006420C0	43 44 45 46 00 00 00 00 00 00 00 00 00 00 00 00	CDEF.....
006420D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
006420E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
006420F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00642100	32 00 8B C0 1F 00 1C 00 1F 00 1E 00 1F 00 1E 00	2.iL.L.V.V.V.V.V.
00642110	1F 00 1F 00 1E 00 1F 00 1E 00 1F 00 1F 00 1D 00	V.V.V.V.V.V.V.V.V.
00642120	1F 00 1E 00 1F 00 1E 00 1F 00 1F 00 1F 00 1F 00	V.V.V.V.V.V.V.V.V.
00642130	1F 00 1F 00 00 00 00 00 00 00 00 00 00 00 00 00	V.V.V.V.V.V.V.V.V.

Command



Hardware breakpoint 1 at 00DAA757

Open Memory map and set mem 1 bp in the section on access code

like following

 - [Memory map]

M File View Debug Plugins Options Window Help

 L E M T W H C / K B R ... S 

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00010000	00001000				Priv	RW	RW	
00020000	00001000				Priv	RW	RW	
0012B000	00001000				Priv	RW	Gua	RW
0012C000	00004000			stack of ma	Priv	RW	Gua	RW
00130000	00003000				Map	R	R	
00140000	00002000				Map	R	R	
00150000	0000C000				Priv	RW	RW	
00250000	00006000				Priv	RW	RW	
00260000	00003000				Map	RW	RW	
00270000	00016000				Map	R	R	\Device\HarddiskVolume1\WINDOWS\system32\unicode.nls
00290000	0003D000				Map	R	R	\Device\HarddiskVolume1\WINDOWS\system32\locale.nls
002D0000	00041000				Map	R	R	\Device\HarddiskVolume1\WINDOWS\system32\sortkey.nls
00320000	00006000				Map	R	R	\Device\HarddiskVolume1\WINDOWS\system32\sorttbls.nls
00330000	00001000				Priv	RWE	RWE	
00340000	00001000				Priv	RWE	RWE	
00350000	00001000				Priv	RW	RW	
00360000	00001000				Priv	RW	RW	
00370000	00005000				Priv	RW	RW	
00380000	00003000				Map	R	R	\Device\HarddiskVolume1\WINDOWS\system32\ctype.nls
00390000	00003000				Priv	RW	RW	
003A0000	00002000				Map	R	R	
003B0000	00002000				Map	R	R	
003C0000	00004000				Priv	RW	RW	
003D0000	00002000				Map	R	R	
003E0000	00001000				Priv	RWE	RWE	
003F0000	00001000				Priv	RWE	RWE	
00400000	00001000	DzPhpEd		PE header	Imag	R	RWE	
00401000	00241000	DzPhpEd		code	Imag	R	RWE	
00642000	00007000	DzPhpEd		data	Imag	R	RWE	
00649000	0000B000	DzPhpEd			Imag	R	RWE	
00654000	00004000	DzPhpEd			Imag	R	RWE	
00658000	00001000	DzPhpEd		exports	Imag	R	RWE	
00659000	00001000	DzPhpEd			Imag	R	RWE	
0065A000	00001000	DzPhpEd			Imag	R	RWE	
0065B000	00027000	DzPhpEd			Imag	R	RWE	
00682000	00104000	DzPhpEd	.rsrc	resources	Imag	R	RWE	
00786000	0005E000	DzPhpEd	.data	imports,rel	Imag	R	RWE	
007E4000	00001000	DzPhpEd	.adata		Imag	R	RWE	

Press F9 to continue to run soft, we will stop the code below

[CPU - main thread, module DzPhpEd]

File View Debug Plugins Options Window Help

LEMTW H C / K B R ... S

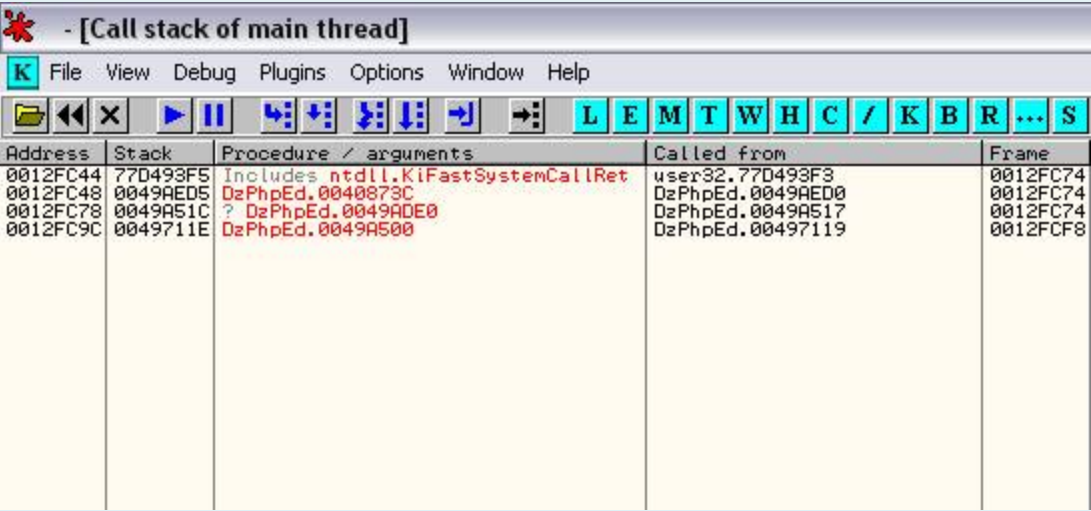
00407436	74 2D	JE SHORT DzPhpEd.00407465	
00407438	E8 87A0FFFF	CALL DzPhpEd.004014C4	JMP to kernel32.GetVersion
0040743D	25 FF000000	AND EAX,0FF	
00407442	66:83F8 04	CMP AX,4	
00407446	76 0C	JBE SHORT DzPhpEd.00407454	
00407448	C705 C0956400 0	MOV DWORD PTR DS:[6495C0],3	
00407452	EB 20	JMP SHORT DzPhpEd.00407474	
00407454	E8 DB9FFFFF	CALL DzPhpEd.00401434	
00407459	E8 86FEFFFF	CALL DzPhpEd.004072E4	
0040745E	A3 C0956400	MOV DWORD PTR DS:[6495C0],EAX	
00407463	EB 0F	JMP SHORT DzPhpEd.00407474	
00407465	E8 CA9FFFFF	CALL DzPhpEd.00401434	
0040746A	E8 75FEFFFF	CALL DzPhpEd.004072E4	
0040746F	A3 C0956400	MOV DWORD PTR DS:[6495C0],EAX	
00407474	E8 43A0FFFF	CALL DzPhpEd.004014B0	
00407479	A3 34906400	MOV DWORD PTR DS:[649034],EAX	
0040747E	C3	RETN	
0040747F	90	NOP	
00407480	E8 7B8B8E00	CALL 00FF0000	
00407485	98	CWDE	
00407486	8BC0	MOV EAX,EAX	
00407488	-FF25 80436500	JMP DWORD PTR DS:[654380]	kernel32.LocalAlloc
0040748E	8BC0	MOV EAX,EAX	
00407490	-FF25 7C436500	JMP DWORD PTR DS:[65437C]	kernel32.TlsGetValue
00407496	8BC0	MOV EAX,EAX	
00407498	-FF25 78436500	JMP DWORD PTR DS:[654378]	kernel32.TlsSetValue
0040749E	8BC0	MOV EAX,EAX	
004074A0	50	PUSH EAX	
004074A1	6A 40	PUSH 40	
004074A3	E8 E0FFFFFF	CALL DzPhpEd.00407488	JMP to kernel32.LocalAlloc
004074A8	C3	RETN	
004074A9	8D40 00	LEA EAX,DWORD PTR DS:[EAX]	
004074AC	B8 FA000000	MOV EAX,0FA	
004074B1	C3	RETN	
004074B2	8BC0	MOV EAX,EAX	
004074B4	53	PUSH EBX	
004074B5	E8 F2FFFFFF	CALL DzPhpEd.004074AC	
004074BA	8BD8	MOV EBX,EAX	
004074BC	85DB	TEST EBX,EBX	
004074BE	74 36	JE SHORT DzPhpEd.004074F6	
004074C0	833D C4206400 F	CMP DWORD PTR DS:[6420C4],-1	
004074C7	75 0A	JNZ SHORT DzPhpEd.004074D3	
004074C9	B8 E2000000	MOV EAX,0E2	
004074CF	E8 149FFFFF	CALL DzPhpEd.00407488	

Address	Hex dump	ASCII
00642000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 000i@.
00642010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 002!!0.iLi@.
00642020	00 00 00 00 32 13 8B C0 02 00 8B C0 00 8D 40 00 00i@.0i@.
00642030	00 8D 40 00 00 8D 40 00 01 8D 40 00 00 00 00 00 00i@.0i@.
00642040	00 00 00 00 88 22 40 00 48 24 40 00 C8 27 40 00 007"0.H\$0.0@.
00642050	00 C8 CC C8 C9 D7 CF C8 CD CE DB D8 DA D9 CA DC00000000
00642060	DD DE DF E0 E1 E3 00 E4 E5 8D 40 00 03 00 00 00 0000000000
00642070	00 00 00 00 01 00 00 00 02 00 00 00 03 00 00 0000000000
00642080	00 00 00 00 00 00 00 00 00 00 00 00 45 72 72 6F00000000
00642090	72 00 8B C0 52 75 6E 74 69 6D 65 20 65 72 72 6F00000000
006420A0	72 20 20 20 20 20 61 74 20 30 30 30 30 30 30 3000000000
006420B0	30 00 8B C0 30 31 32 33 34 35 36 37 38 39 41 4200000000
006420C0	43 44 45 46 00 00 00 00 00 00 00 00 00 00 00 0000000000
006420D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000
006420E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000
006420F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000
00642100	32 00 8B C0 1F 00 1C 00 1F 00 1E 00 1F 00 1E 0000000000
00642110	1F 00 1F 00 1E 00 1F 00 1E 00 1F 00 1F 00 1D 0000000000
00642120	1F 00 1E 00 1F 00 1E 00 1F 00 1E 00 1F 00 1F 0000000000
00642130	1F 00 1F 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

Command: Break-on-access when executing [00407480]

Usually we are at the OEP khong.OK this time, do not have worry anything, we're in the code of the program and our target was completely unpack in memory:) from the time, the BP is set to work.Buoc next trial is to find the screen and then find bytes to patch loader is in memory In this program, I will nag you how to find simple:)
Back Olly. Restart, press Shift-F9, open the Memory map -> set bp on access code in the section, press Shift + F9 to press Shift + F9 to run a trial program and screen appear

Now, press F12 to pause in Olly again, press ALT + K to open the window Call stack



Enter Call functions 3 and we will be here.

- [CPU - main thread, module DzPhpEd]

File View Debug Plugins Options Window Help

Assembly List:

```

00497119 E8 E2330000 CALL DzPhpEd.0049A500
0049711E A1 F49B6400 MOV EAX,DWORD PTR DS:[649BF4]
00497123 80B8 9C000000 0 CMP BYTE PTR DS:[EAX+9C],0
0049712A 74 0F JE SHORT DzPhpEd.0049713B
0049712C 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
0049712F C780 4C020000 0 MOV DWORD PTR DS:[EAX+24C],2
00497139 EB 14 JMP SHORT DzPhpEd.0049714F
0049713B 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
0049713E 83B8 4C020000 0 CMP DWORD PTR DS:[EAX+24C],0
00497145 74 08 JE SHORT DzPhpEd.0049714F
00497147 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
0049714A E8 10F0FFFF CALL DzPhpEd.00496E6C
0049714F 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
00497152 8B80 4C020000 MOV EAX,DWORD PTR DS:[EAX+24C]
00497158 85C0 TEST EAX,EAX
0049715A 74 B8 JE SHORT DzPhpEd.00497114
0049715C 8945 F8 MOV DWORD PTR SS:[EBP-8],EAX
0049715F 6A 00 PUSH 0
00497161 6A 00 PUSH 0
00497163 68 01B00000 PUSH 0B001
00497168 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
0049716B E8 A48AFEFF CALL DzPhpEd.0047FC14
00497170 50 PUSH EAX
00497171 E8 4E14F7FF CALL DzPhpEd.004085C4
00497176 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
00497179 E8 968AFEFF CALL DzPhpEd.0047FC14
0049717E 8BD8 MOV EBX,EAX
00497180 E8 CF10F7FF CALL DzPhpEd.00408254
00497185 3BD8 CMP EBX,EAX
00497187 74 05 JE SHORT DzPhpEd.0049718E
00497189 33C0 XOR EAX,EAX
0049718B 8945 E4 MOV DWORD PTR SS:[EBP-1C],EAX
0049718E 33C0 XOR EAX,EAX
00497190 5A POP EDX
00497191 59 POP ECX
00497192 59 POP ECX
00497193 64:8910 MOV DWORD PTR FS:[EAX],EDX
00497196 68 AB714900 PUSH DzPhpEd.004971AB
0049719B 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
0049719E E8 61F0FFFF CALL DzPhpEd.00496F04
004971A3 C3 RETN
004971A4 ^E9 93D5F6FF JMP DzPhpEd.0040473C
004971A9 ^EB F0 JMP SHORT DzPhpEd.0049719B
004971AE 33C0 XOR EAX,EAX

```

Disassembly:

```

CALL DzPhpEd.0049A500
MOV EAX,DWORD PTR DS:[649BF4]
CMP BYTE PTR DS:[EAX+9C],0
JE SHORT DzPhpEd.0049713B
MOV EAX,DWORD PTR SS:[EBP-4]
MOV DWORD PTR DS:[EAX+24C],2
JMP SHORT DzPhpEd.0049714F
MOV EAX,DWORD PTR SS:[EBP-4]
CMP DWORD PTR DS:[EAX+24C],0
JE SHORT DzPhpEd.0049714F
MOV EAX,DWORD PTR SS:[EBP-4]
CALL DzPhpEd.00496E6C
MOV EAX,DWORD PTR SS:[EBP-4]
MOV EAX,DWORD PTR DS:[EAX+24C]
TEST EAX,EAX
JE SHORT DzPhpEd.00497114
MOV DWORD PTR SS:[EBP-8],EAX
PUSH 0
PUSH 0
PUSH 0B001
MOV EAX,DWORD PTR SS:[EBP-4]
CALL DzPhpEd.0047FC14
PUSH EAX
CALL DzPhpEd.004085C4
MOV EAX,DWORD PTR SS:[EBP-4]
CALL DzPhpEd.0047FC14
MOV EBX,EAX
CALL DzPhpEd.00408254
CMP EBX,EAX
JE SHORT DzPhpEd.0049718E
XOR EAX,EAX
MOV DWORD PTR SS:[EBP-1C],EAX
XOR EAX,EAX
POP EDX
POP ECX
POP ECX
MOV DWORD PTR FS:[EAX],EDX
PUSH DzPhpEd.004971AB
MOV EAX,DWORD PTR SS:[EBP-4]
CALL DzPhpEd.00496F04
RETN
JMP DzPhpEd.0040473C
JMP SHORT DzPhpEd.0049719B
XOR EAX,EAX

```

Comments:

```

JMP to user32.SendMessageA
JMP to user32.GetActiveWindow

```

This is nag routine we need to find the end of this routine
scroll down!

00497261	E8 12240000	CALL DzPhpEd.00497260
00497266	C3	RETN
00497267	^E9 D0D4F6FF	JMP DzPhpEd.0040473C
0049726C	^EB EE	JMP SHORT DzPhpEd.0049725C
0049726E	33C0	XOR EAX,EAX
00497270	5A	POP EDX
00497271	59	POP ECX
00497272	59	POP ECX
00497273	64:8910	MOV DWORD PTR FS:[EAX],EDX
00497276	68 8B724900	PUSH DzPhpEd.0049728B
0049727B	8D45 E0	LEA EAX,DWORD PTR SS:[EBP-20]
0049727E	E8 10DC66FF	CALL DzPhpEd.00404EA0
00497283	C3	RETN
00497284	^E9 B3D4F6FF	JMP DzPhpEd.0040473C
00497289	^EB F0	JMP SHORT DzPhpEd.0049727B
0049728B	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]
0049728E	5E	POP ESI
0049728F	5B	POP EBX
00497290	8BE5	MOV ESP,EBP
00497292	5D	POP EBP
00497293	C3	RETN
00497294	55	PUSH EBP
00497295	8BEC	MOV EBP,ESP
00497297	51	PUSH ECX
00497298	53	PUSH EBX
00497299	56	PUSH ESI
0049729A	57	PUSH EDI
0049729B	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX
0049729E	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]
004972A1	80B8 A6010000 0	CMP BYTE PTR DS:[EAX+1A6],0
004972A8	^74 52	JE SHORT DzPhpEd.004972FC
004972AA	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]
004972AD	E8 2E55FEFF	CALL DzPhpEd.0047C7E0
004972B2	8BF0	MOV ESI,EAX
004972B4	4E	DEC ESI
004972B5	85F6	TEST ESI,ESI

1 at a BP 00497293 C3 RETN

Press Shift-F9, then select "NO" in the nag screen

00497272	59	POP ECX
00497273	64:8910	MOV DWORD PTR FS:[EAX],EDX
00497276	68 8B724900	PUSH DzPhpEd.0049728B
0049727B	8D45 E0	LEA EAX,DWORD PTR SS:[EBP-20]
0049727E	E8 10DC66FF	CALL DzPhpEd.00404EA0
00497283	C3	RETN
00497284	^E9 B3D4F6FF	JMP DzPhpEd.0040473C
00497289	^EB F0	JMP SHORT DzPhpEd.0049727B
0049728B	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]
0049728E	5E	POP ESI
0049728F	5B	POP EBX
00497290	8BE5	MOV ESP,EBP
00497292	5D	POP EBP
00497293	C3	RETN
00497294	55	PUSH EBP
00497295	8BEC	MOV EBP,ESP
00497297	51	PUSH ECX
00497298	53	PUSH EBX
00497299	56	PUSH ESI
0049729A	57	PUSH EDI
0049729B	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX
0049729E	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]
004972A1	80B8 A6010000 0	CMP BYTE PTR DS:[EAX+1A6],0
004972A8	^74 52	JE SHORT DzPhpEd.004972FC
004972AA	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]
004972AD	E8 2E55FEFF	CALL DzPhpEd.0047C7E0
004972B2	8BF0	MOV ESI,EAX
004972B4	4E	DEC ESI
004972B5	85F6	TEST ESI,ESI
004972B7	^7C 43	JL SHORT DzPhpEd.004972FC
004972B8	4E	DEC ESI

RETN command will take us to nag exit routine, so pressing F8

We use here

00626C4B	E8 D01CE5FF	CALL DzPhpEd.00478920
00626C50	8B45 F0	MOV EAX,DWORD PTR SS:[EBP-10]
00626C53	8B40 48	MOV EAX,DWORD PTR DS:[EAX+48]
00626C56	8B55 F4	MOV EDX,DWORD PTR SS:[EBP-C]
00626C59	2B42 48	SUB EAX,DWORD PTR DS:[EDX+48]
00626C5C	8B55 F8	MOV EDX,DWORD PTR SS:[EBP-8]
00626C5F	8B52 40	MOV EDX,DWORD PTR DS:[EDX+40]
00626C62	2B00	SUB EDX,EAX
00626C64	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]
00626C67	E8 681CE5FF	CALL DzPhpEd.00478804
00626C6C	B2 04	MOV DL,4
00626C6E	8B45 EC	MOV EAX,DWORD PTR SS:[EBP-14]
00626C71	E8 62DDE6FF	CALL DzPhpEd.00494908
00626C76	8B45 EC	MOV EAX,DWORD PTR SS:[EBP-14]
00626C79	8B10	MOV EDX,DWORD PTR DS:[EAX]
00626C7B	FF92 EC000000	CALL DWORD PTR DS:[EDX+EC]
00626C81	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX
00626C84	33C0	XOR EAX,EAX
00626C86	5A	POP EDX
00626C87	59	POP ECX
00626C88	59	POP ECX
00626C89	64:8910	MOV DWORD PTR FS:[EAX],EDX
00626C8C	68 A16C6200	PUSH DzPhpEd.00626CA1
00626C91	8B45 EC	MOV EAX,DWORD PTR SS:[EBP-14]
00626C94	E8 0FD3DDFF	CALL DzPhpEd.00403FA8
00626C99	C3	RETN
00626C9A	^E9 9DDAD0FF	JMP DzPhpEd.0040473C
00626C9F	^EB F0	JMP SHORT DzPhpEd.00626C91
00626CA1	8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]
00626CA4	5E	POP ESI
00626CA5	5B	POP EBX
00626CA6	8BE5	MOV ESP,EBP
00626CA8	5D	POP EBP
00626CA9	C3	RETN
00626CAA	0000	ADD BYTE PTR DS:[EAX],AL
00626CAC	2300	AND EAX,DWORD PTR DS:[EAX]
00626CAE	0000	ADD BYTE PTR DS:[EAX],AL
00626CB0	FFFF	

Ham Call you see, is a function called nag screen;). We are not in the right place, so trace back with F8, the carrots we also stop here

OllyDbg - [CPU - main thread, module DzPhpEd]

File View Debug Plugins Options Window Help

Assembly code (hex, disassembly, comment):

```

00632E3E 8B90 C0070000 MOV EDX,DWORD PTR DS:[EAX+7C0]
00632E44 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
00632E47 8B80 48080000 MOV EAX,DWORD PTR DS:[EAX+848]
00632E4D 8B08 MOV ECX,DWORD PTR DS:[EAX]
00632E4F FF51 68 CALL DWORD PTR DS:[ECX+68]
00632E52 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]
00632E55 8B80 48080000 MOV EAX,DWORD PTR DS:[EAX+848]
00632E5B E8 90940000 CALL DzPhpEd.0063C2F0
00632E60 8D45 B0 LEA EAX,DWORD PTR SS:[EBP-50]
00632E63 8B15 F4856400 MOV EDX,DWORD PTR DS:[6485F4]
00632E69 8B12 MOV EDX,DWORD PTR DS:[EDX]
00632E6B E8 2822D0FF CALL DzPhpEd.00405098
00632E70 8B45 B0 MOV EAX,DWORD PTR SS:[EBP-50]
00632E73 8D55 B4 LEA EDX,DWORD PTR SS:[EBP-4C]
00632E76 E8 756FD0FF CALL DzPhpEd.00409DF0
00632E7B 837D B4 00 CMP DWORD PTR SS:[EBP-4C],0
00632E7F 0F85 6A010000 JNZ DzPhpEd.00632FEF
00632E85 A1 2C846400 MOV EAX,DWORD PTR DS:[64842C]
00632E8A 8338 01 CMP DWORD PTR DS:[EAX],1
00632E8D 7D 79 JGE SHORT DzPhpEd.00632F08
00632E8F 8D45 AC LEA EAX,DWORD PTR SS:[EBP-54]
00632E92 50 PUSH EAX
00632E93 8D55 A4 LEA EDX,DWORD PTR SS:[EBP-5C]
00632E96 A1 DC816400 MOV EAX,DWORD PTR DS:[6481DC]
00632E9B E8 EC43D0FF CALL DzPhpEd.0040728C
00632EA0 8B55 A4 MOV EDX,DWORD PTR SS:[EBP-5C]
00632EA3 8D40 A8 LEA ECX,DWORD PTR SS:[EBP-58]
00632EA6 B8 2C336300 MOV EAX,DzPhpEd.0063332C
00632EAB E8 9C68E8FF CALL DzPhpEd.004B974C
00632EB0 8B45 A8 MOV EAX,DWORD PTR SS:[EBP-58]
00632EB3 8B15 C87A6400 MOV EDX,DWORD PTR DS:[647AC8]
00632EB9 8B12 MOV EDX,DWORD PTR DS:[EDX]
00632EBB 8955 9C MOV DWORD PTR SS:[EBP-64],EDX
00632EBE C645 A0 00 MOV BYTE PTR SS:[EBP-60],0
00632EC2 8D55 9C LEA EDX,DWORD PTR SS:[EBP-64]
00632EC5 33C9 XOR ECX,ECX
00632EC7 E8 188ADDFF CALL DzPhpEd.0040B8E4
00632ECB 8B45 AC MOV EAX,DWORD PTR SS:[EBP-54]
00632ECF 33D2 XOR EDX,EDX
00632ED1 E8 A63BFFFF CALL DzPhpEd.00626A7C
00632ED6 83F8 06 CMP EAX,6
00632ED9 75 0F JNZ SHORT DzPhpEd.00632EEA
00632EDB E8 247DE8FF CALL DzPhpEd.004BAC04
00632EDF E8 12F8FFFF CALL DzPhpEd.004BAC04

```

Comment: ASCII "const_TrialExpiries"

Status bar: Jump is taken
00632EEA=DzPhpEd.00632EEA

hehe, very clear, do not need to explain anything more. I do not go into here

If we change the 00632E7F JNZ to JMP, will always jump over the trial that is already screen bypass trial;). If you want to check, then set 1 bp hardware in order jnz and then follow the steps on (the memmap -> Code Section BP and stuff: P) you will break in order jnz then you or learn;)

Now we will move to 1 part interesting

Created loader:)

For this program we will use the feature "window caption" Ok, open ABEL. Now, please write to address JNZ order and we can close Olly.

Select "Window Caption" as following

***ABEL* loader generator v2.31 by c0rdat ^ind.**




File

Detection method

☐ Standard
☒ Window caption
☐ Window class

Search for window caption/class name (and what to do later):
 Refresh

☐ Hide window ☐ Attempt to kill it ☐ Get ProcessID from handle

Choose icon: (default)   
Restore Owl Load icon

Target program filename:

Generated loader/installer filename:

☐ Do file cleaning File options ☐ Do registry cleaning Reg options

Timeout [sec]: 15 Patch delay [sec]: 0

☒ Autolearning enabled ☒ Show splash screen
☐ Priority boost ☐ Ignore memory faults
☐ Enable debug mode
☐ First child process found is the main process

☐ Include simple installer Installer options
☐ Include DateFaker module Date options

Info caption: Loader created by: Author name: Author mail/www:

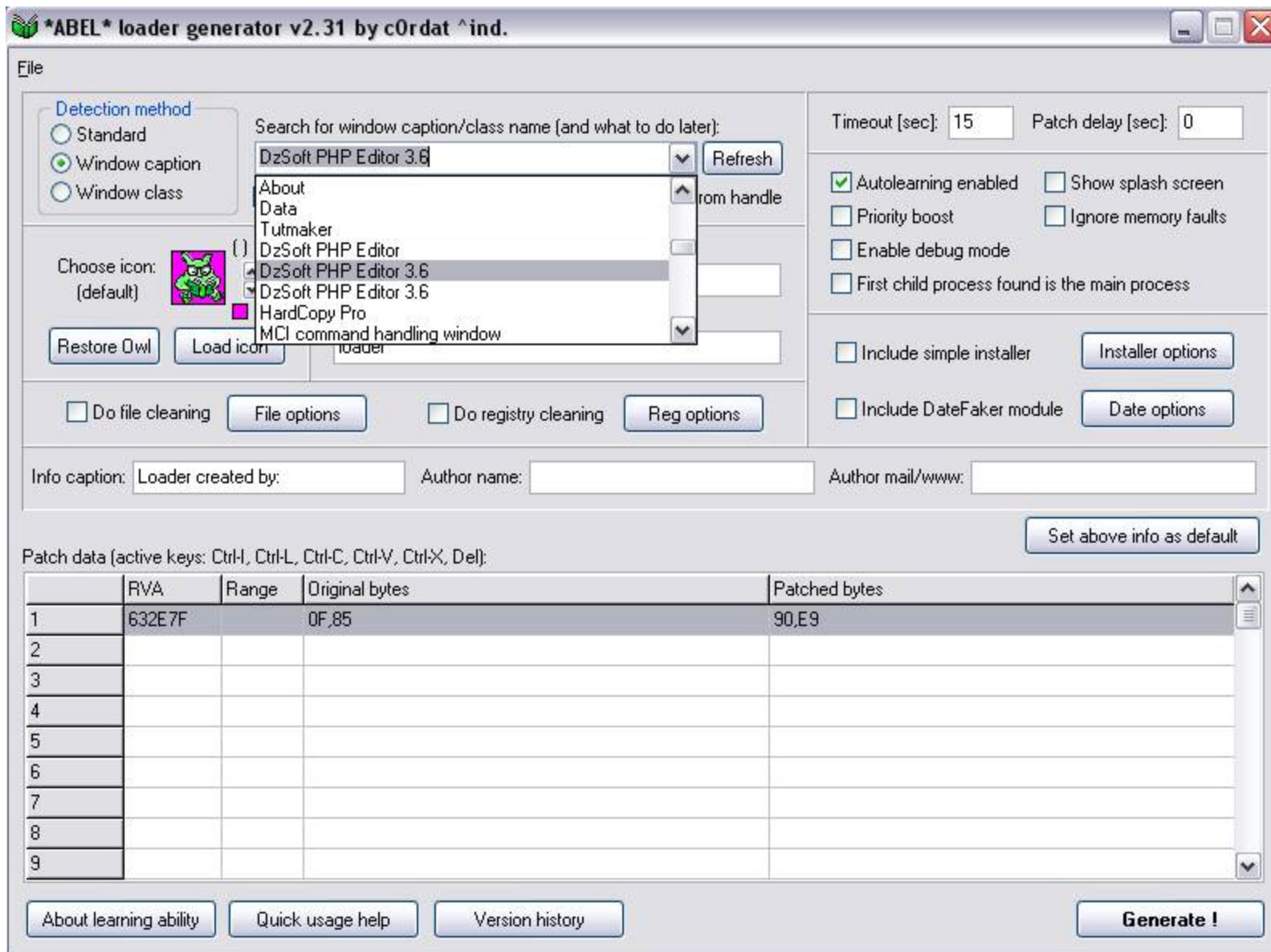
Set above info as default

Patch data (active keys: Ctrl-I, Ctrl-L, Ctrl-C, Ctrl-V, Ctrl-X, Del):

	RVA	Range	Original bytes	Patched bytes
1				
2				
3				
4				
5				
6				
7				
8				
9				

About learning ability Quick usage help Version history **Generate !**

Now program run normally, then click "refresh" in the ABEL immediately when you see the nag screen. Click on the arrow next to



Due to the CRC check asprotect loader should be able to do patch in memory is normal because we use the "window caption, it will patch program right time we need => now need to enter the patch as the image =>

***ABEL* loader generator v2.31 by c0rdat ^ind.**


File

Detection method

☐ Standard
☒ Window caption
☐ Window class

Search for window caption/class name (and what to do later):

☐ Hide window ☐ Attempt to kill it ☐ Get ProcessID from handle

Choose icon:  (default)

Target program filename:

Generated loader/installer filename:

☐ Do file cleaning ☐ Do registry cleaning

Timeout [sec]: Patch delay [sec]:

☒ Autolearning enabled ☐ Show splash screen
☐ Priority boost ☐ Ignore memory faults
☐ Enable debug mode
☐ First child process found is the main process

☐ Include simple installer
☐ Include DateFaker module

Info caption: Author name: Author mail/www:

Patch data (active keys: Ctrl-H, Ctrl-L, Ctrl-C, Ctrl-V, Ctrl-X, Del):

	RVA	Range	Original bytes	Patched bytes
1	632E7F		0F,85	90,E9
2				
3				
4				
5				
6				
7				
8				
9				

Now, run loader just created, and see the results => It (should) work = D

Some notes: - Sometimes when you install, the program folder name matches the program's window caption. When it is, we should reduce the Explorer Windoww di.Boi because records are also ABEL caption = Windows

Explorer). Then click the button "refresh" the ABEL only the caption of the window we need to patch program

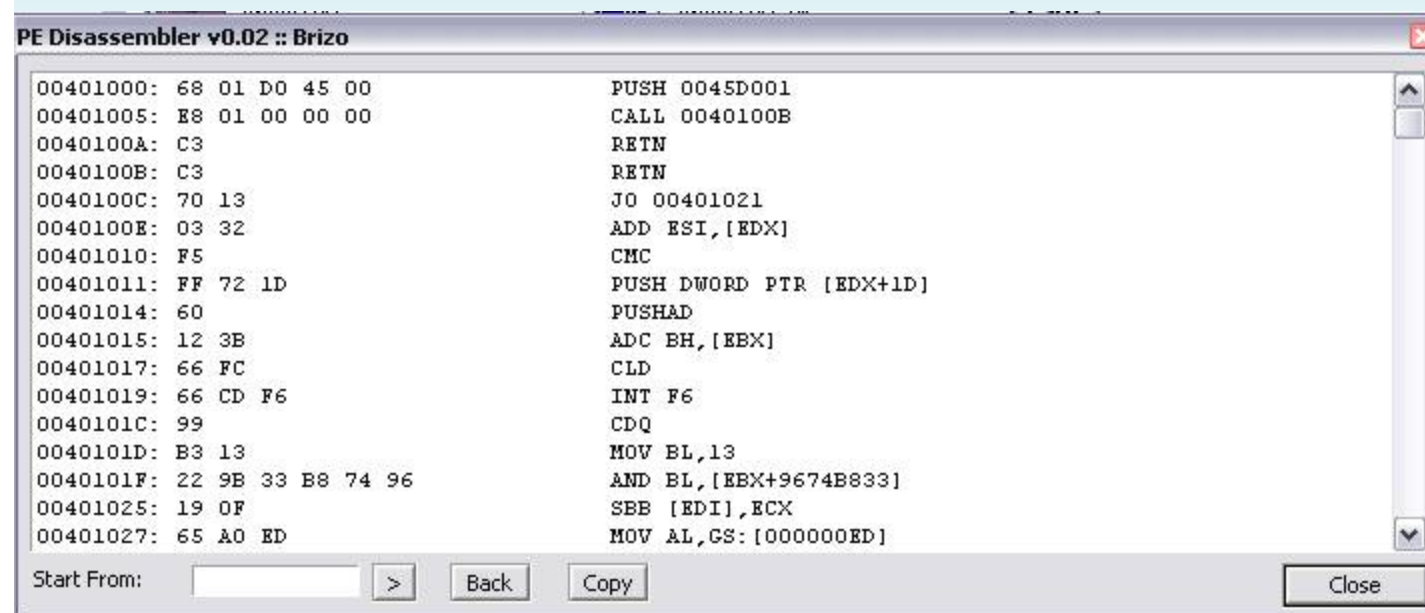
but also note that you can use other features but do not necessarily use the "Window caption":))

OK, now we will see a different trick =>

"The window CLASS TRICK"

Target selected are DVD Region + CSS Free =)

First analyze target



30 day trial =)

I do not repeat the action on working to create breakpoints because I sure you know how = D Ok, so do take the time we go crack 30 day trial

After the trace me to the conclusion

Address	Hex	Mnemonic	Comment
00403669	8BF8	MOV EDI,EAX	
0040366B	81C7 B8020000	ADD EDI,2B8	
00403671	C70424 F4F3EE00	MOV DWORD PTR SS:[ESP],0EEF3F4	
00403678	FF15 EC804100	CALL DWORD PTR DS:[4180EC]	GDI32.CreateSolidBrush
0040367E	50	PUSH EAX	
0040367F	8BCF	MOV ECX,EDI	
00403681	E8 A2130100	CALL DVDRegio.00414A28	JMP to mfc42.#1641
00403686	53	PUSH EBX	
00403687	8BCE	MOV ECX,ESI	
00403689	E8 B6130000	CALL DVDRegio.00404A44	
0040368E	E8 F7ECFFFF	CALL DVDRegio.0040238A	
00403693	8BC8	MOV ECX,EAX	
00403695	E8 78FBFFFF	CALL DVDRegio.00403215	
0040369A	85C0	TEST EAX,EAX	
0040369C	75 20	JNZ SHORT DVDRegio.004036BE	
0040369E	8BCE	MOV ECX,ESI	
004036A0	C786 B4020000 0	MOV DWORD PTR DS:[ESI+2B4],1	
004036AA	E8 3A0A0000	CALL DVDRegio.004040E9	
004036AF	391D E0054200	CMP DWORD PTR DS:[4205E0],EBX	
004036B5	74 07	JE SHORT DVDRegio.004036BE	
004036B7	33F6	XOR ESI,ESI	
004036B9	E9 D3000000	JMP DVDRegio.00403791	
004036BE	53	PUSH EBX	
004036BF	8BCE	MOV ECX,ESI	
004036C1	E8 F3120000	CALL DVDRegio.004049B9	
004036C6	E8 BFECFFFF	CALL DVDRegio.0040238A	
004036CB	8BC8	MOV ECX,EAX	
004036CD	E8 90F8FFFF	CALL DVDRegio.00402F62	
004036D2	8945 E4	MOV DWORD PTR SS:[EBP-1C],EAX	
004036D5	E8 B0ECFFFF	CALL DVDRegio.0040238A	
004036DA	8BC8	MOV ECX,EAX	
004036DC	E8 16F4FFFF	CALL DVDRegio.00402AF7	
004036E1	8BCE	MOV ECX,ESI	
004036E3	E8 5A120000	CALL DVDRegio.00404942	
004036E8	8BCE	MOV ECX,ESI	
004036EA	8BF8	MOV EDI,EAX	
004036EC	E8 EE1E0000	CALL DVDRegio.004055DF	
004036F1	3BF8	CMP EDI,EBX	
004036F3	75 04	JNZ SHORT DVDRegio.004036F9	
004036F5	3BC3	CMP EAX,EBX	
004036F7	74 25	JE SHORT DVDRegio.0040371E	
004036F9	3BC3	CMP EAX,EBX	
004036FB	6A FF	PUSH -1	

004036AA E8 3A0A0000 CALL DVDRegio.004040E9, this function is called nag => and in JNZ 0040369C will jump over it =>

Now let's see, how to create perfect loader 1 =>

ABEL open up, but this time we do not use "Window captions" = (because you will receive error error 45 = (

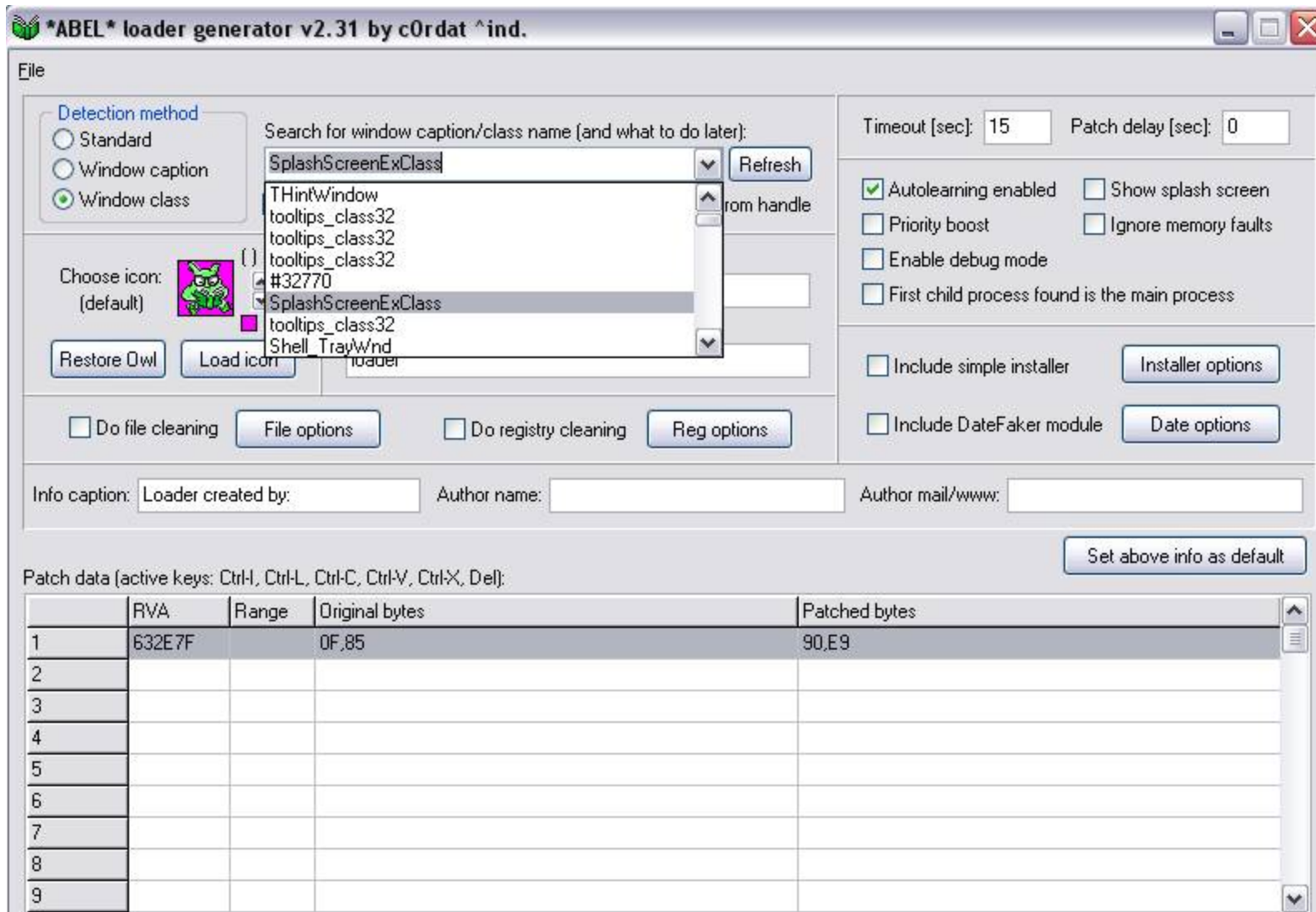
Well, in the run when this target will appear SplashScreen = 1)



DVD Region+CSS Free 5.9.6.8 - Trial version

hehe we can use 1 features or other quite well =)

Now in Class ABEL select Window and then click the arrow buttons scroll down a little and you will find interesting =)



hehehe, select window class and then enter the values need to patch as following

***ABEL* loader generator v2.31 by c0rdat ^ind.**

File

Detection method

☐ Standard
☐ Window caption
☒ Window class


Search for window caption/class name (and what to do later):
 Refresh

☐ Hide window ☐ Attempt to kill it ☐ Get ProcessID from handle

Timeout [sec]: Patch delay [sec]:

☒ Autolearning enabled ☐ Show splash screen
☐ Priority boost ☐ Ignore memory faults
☐ Enable debug mode
☐ First child process found is the main process

☐ Include simple installer
☐ Include DateFaker module

Choose icon:  (default)

Target program filename:

Generated loader/installer filename:

☐ Do file cleaning ☐ Do registry cleaning

Info caption: Author name: Author mail/www:

Patch data (active keys: Ctrl-I, Ctrl-L, Ctrl-C, Ctrl-V, Ctrl-X, Del):

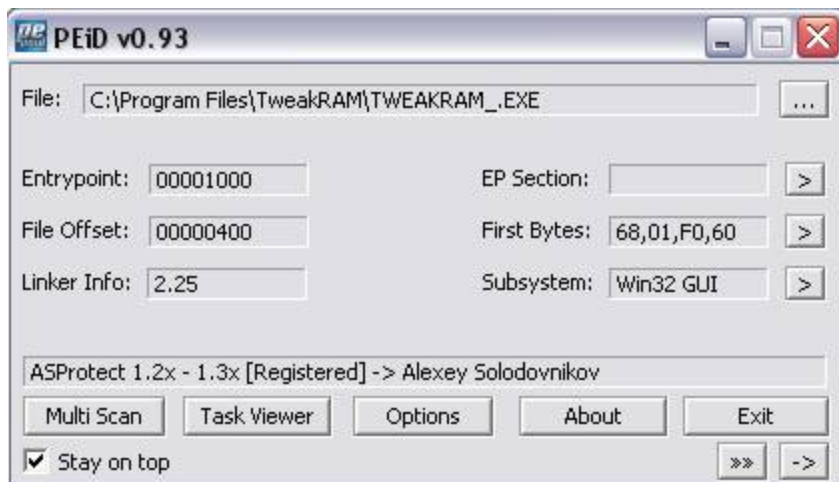
	RVA	Range	Original bytes	Patched bytes
1	40369C	75,20		EB,20
2				
3				
4				
5				
6				
7				
8				
9				

Run loader and see =>

The trick to 1 more

"The PELG Trick" = P

My target is to use TwaekRAM = 5.3)



30 day trial + aspr

Search function calls Call nag

[CPU - main thread, module TWEAKRAM]

File View Debug Plugins Options Window Help

LEA EAX, DWORD PTR SS:[EBP-A] 8045 F6
 PUSH EAX 50
 LEA EAX, DWORD PTR SS:[EBP-4] 8045 FC
 PUSH EAX 50
 CALL TWEAKRAM.00495F7C E8 F43EF3FF
 CMP BYTE PTR SS:[EBP-9], 0 807D F7 00
 JE SHORT TWEAKRAM.005620AA 74 1C
 CMP BYTE PTR SS:[EBP-A], 0 807D F6 00
 JE SHORT TWEAKRAM.005620AA 74 16
 CMP BYTE PTR SS:[EBP-5], 0 807D FB 00
 JNZ SHORT TWEAKRAM.005620AA 75 10
 CMP BYTE PTR SS:[EBP-7], 0 807D F9 00
 JNZ SHORT TWEAKRAM.005620AA 75 0A
 CMP BYTE PTR SS:[EBP-8], 0 807D F8 00
 JE TWEAKRAM.0056214D 0F84 A3000000
 MOV EAX, DWORD PTR DS:[568490] A1 90845600
 CMP DWORD PTR DS:[EAX], 0 8338 00
 JNZ TWEAKRAM.0056214D 0F85 95000000
 XOR EAX, EAX 33C0
 PUSH EBP 55
 PUSH TWEAKRAM.00562143 68 43215600
 PUSH DWORD PTR FS:[EAX] 64:FF30
 MOV DWORD PTR FS:[EAX], ESP 64:8920
 MOV ECX, DWORD PTR DS:[568378] 8B0D 78835600
 MOV ECX, DWORD PTR DS:[ECX] 8B09
 MOV DL, 1 B2 01
 MOV EAX, DWORD PTR DS:[5507BC] A1 BC075500
 CALL TWEAKRAM.0048B024 E8 4A8FF2FF
 MOV EDX, DWORD PTR DS:[568490] 8B15 90845600
 MOV DWORD PTR DS:[EDX], EAX 8902
 MOV EAX, DWORD PTR DS:[568490] A1 90845600
 MOV EAX, DWORD PTR DS:[EAX] 8B00
 MOV EDX, DWORD PTR DS:[EAX] 8B10
 CALL DWORD PTR DS:[EDX+EC] FF92 EC000000
 DEC EAX 48
 JNZ SHORT TWEAKRAM.0056210B 75 17
 MOV EAX, DWORD PTR DS:[568490] A1 90845600
 MOV EAX, DWORD PTR DS:[EAX] 8B00
 CALL TWEAKRAM.004030C4 E8 C41CEAFF
 MOV EAX, DWORD PTR DS:[568490] A1 90845600
 XOR EDX, EDX 33D2
 MOV DWORD PTR DS:[EAX], EDX 8910
 JMP SHORT TWEAKRAM.00562139 EB 2E
 MOV EAX, DWORD PTR DS:[568490] 8B15 90845600

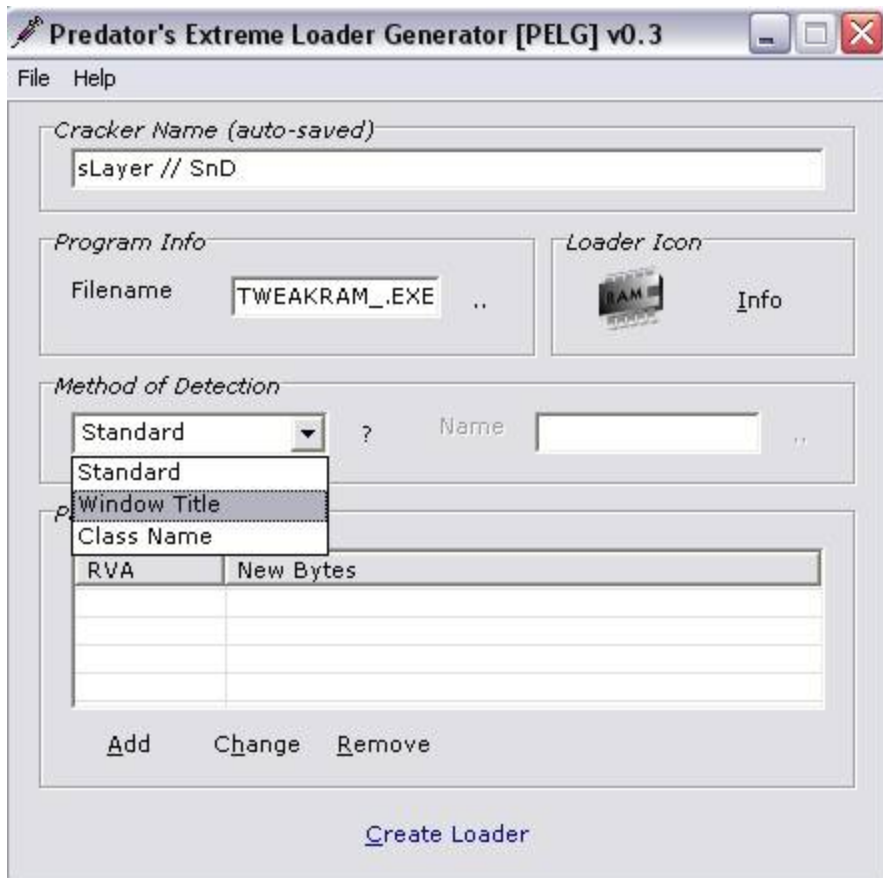
EAX=00000001

005620EB FF92 EC000000 CALL DWORD PTR DS: [EDX + EC]

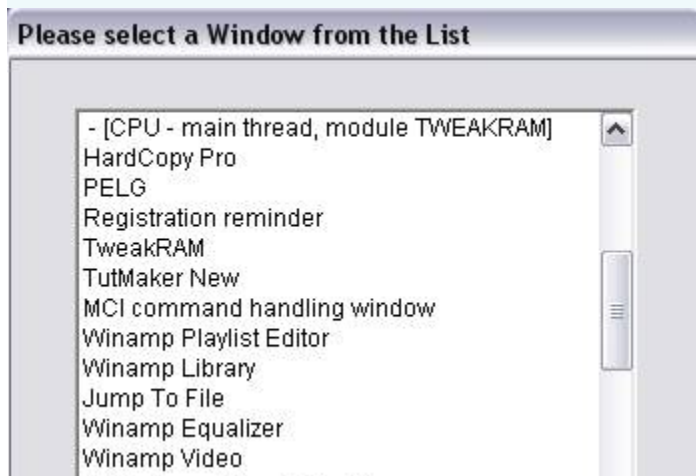
Yeah, it is =)

See how to create 1 pelg loader nice =)

Open up PELG

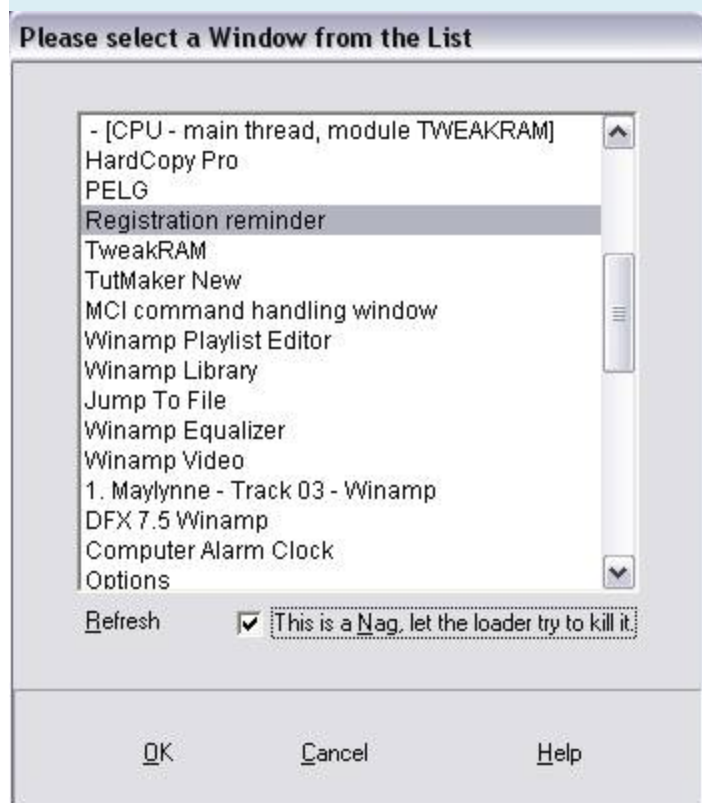


Select the window title and then click the button bowse near the "name" It will show all the window title
Scroll down to 1 little window title to the registration of our



You can see "Registration reminder"? Yep, that's nag title =)

Select the check and then on "This is a nag, let the loader try to kill it"



Now, click OK to create a loader and then run loader test, you will see the loader kills the nag screen and load the program do not have nag. Nhung complete, loader still having little Iraq roi. Chinh clock to 2 years and then run loader test. At this time, do not run program again = (

Back to where we have found in Olly nag

```

0056207F 8D45 FC      LEA EAX,DWORD PTR SS:[EBP-4]
00562082 50           PUSH EAX
00562083 E8 F4EF3FF  CALL TWEAKRAM.00495F7C
00562088 807D F7 00   CMP BYTE PTR SS:[EBP-9],0
0056208C 74 1C        JE SHORT TWEAKRAM.005620AA
0056208E 807D F6 00   CMP BYTE PTR SS:[EBP-A],0
00562092 74 16        JE SHORT TWEAKRAM.005620AA
00562094 807D FB 00   CMP BYTE PTR SS:[EBP-5],0
00562098 75 10        JNZ SHORT TWEAKRAM.005620AA
0056209A 807D F9 00   CMP BYTE PTR SS:[EBP-7],0
0056209E 75 0A        JNZ SHORT TWEAKRAM.005620AA
005620A0 807D F8 00   CMP BYTE PTR SS:[EBP-8],0
005620A4 74 84        JE TWEAKRAM.0056214D
005620AA A1 90845600  MOV EAX,DWORD PTR DS:[568490]
005620AF 8338 00      CMP DWORD PTR DS:[EAX],0
005620B2 75 85        JNZ TWEAKRAM.0056214D
005620B8 33C0        XOR EAX,EAX
005620BA 55           PUSH EBP
005620BB 68 43215600  PUSH TWEAKRAM.00562143
005620C0 64:FF30     PUSH DWORD PTR FS:[EAX]
005620C3 64:8920     MOV DWORD PTR FS:[EAX],ESP
005620C6 8B0D 78835600 MOV ECX,DWORD PTR DS:[568378]
005620CC 8B09        MOV ECX,DWORD PTR DS:[ECX]
005620CE B2 01        MOV DL,1
005620D0 A1 BC075500  MOV EAX,DWORD PTR DS:[5507BC]
005620D5 E8 4A8FF2FF  CALL TWEAKRAM.0048B024
005620DA 8B15 90845600 MOV EDX,DWORD PTR DS:[568490]
005620E0 8902        MOV DWORD PTR DS:[EDX],EAX
005620E2 A1 90845600  MOV EAX,DWORD PTR DS:[568490]
005620E7 8B00        MOV EAX,DWORD PTR DS:[EAX]
005620E9 8B10        MOV EDX,DWORD PTR DS:[EAX]
005620EB FF92 EC000000 CALL DWORD PTR DS:[EDX+EC]
005620F1 48          DEC EAX
005620F2 75 17        JNZ SHORT TWEAKRAM.0056210B
005620F4 A1 90845600  MOV EAX,DWORD PTR DS:[568490]
005620F9 8B00        MOV EAX,DWORD PTR DS:[EAX]
005620FB E8 C41CEAFF  CALL TWEAKRAM.00403DC4
00562100 A1 90845600  MOV EAX,DWORD PTR DS:[568490]
00562105 33D2        XOR EDX,EDX
00562107 8910        MOV DWORD PTR DS:[EAX],EDX
00562109 EB 2E        JMP SHORT TWEAKRAM.00562139
0056210B A1 90845600  MOV EAX,DWORD PTR DS:[568490]
00562110 8B00        MOV EAX,DWORD PTR DS:[EAX]
00562112 E8 0A1CEAFF  CALL TWEAKRAM.00403DC4

```

Look here kī

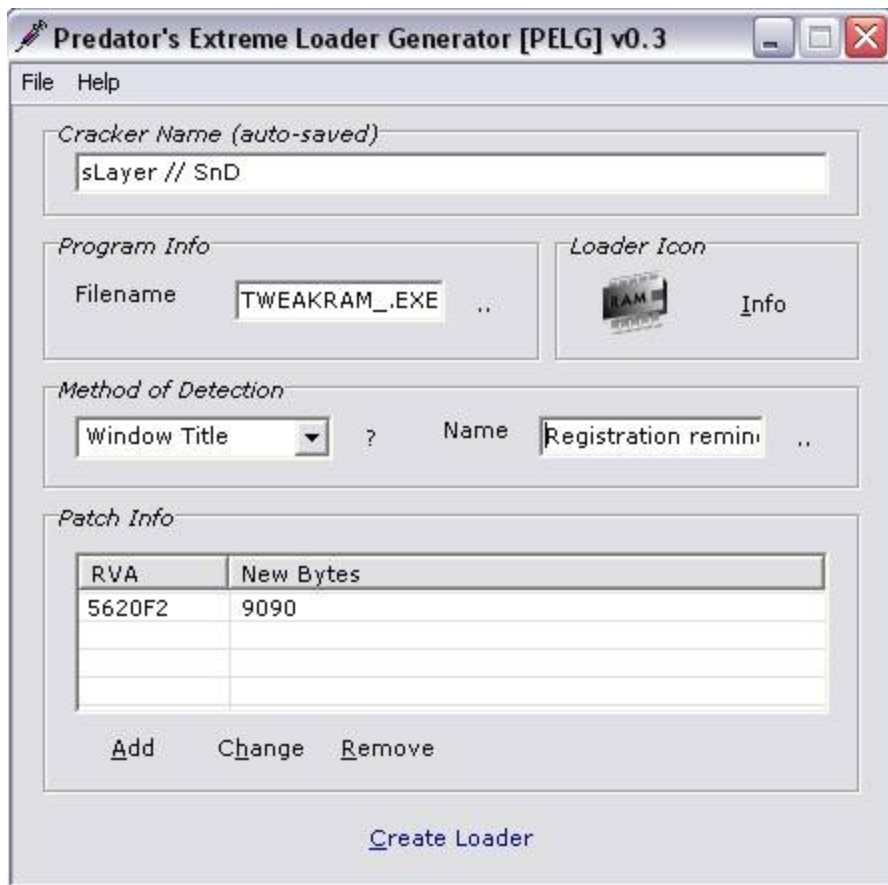
005620EB FF92 EC000000 CALL DWORD PTRDS: [EDX +]----> EC Nag

005620F1 48 DEC EAX

005620F2 75 17 JNZ SHORTTWEAKRAM.0056210B--> If overdue trial

Then jump to close program, if i continue to load the program =)

In PELG (select window title) to enter data patch
as following



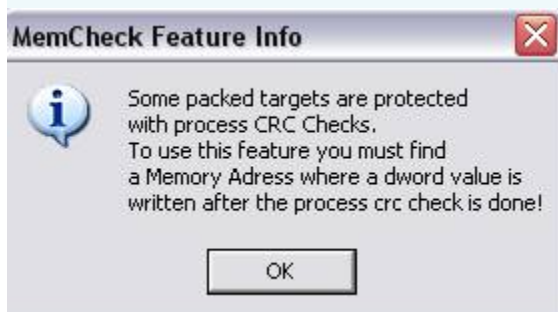
Create loader then run view, the smooth as i have what happened =)

The DUP Trick ** ** = P

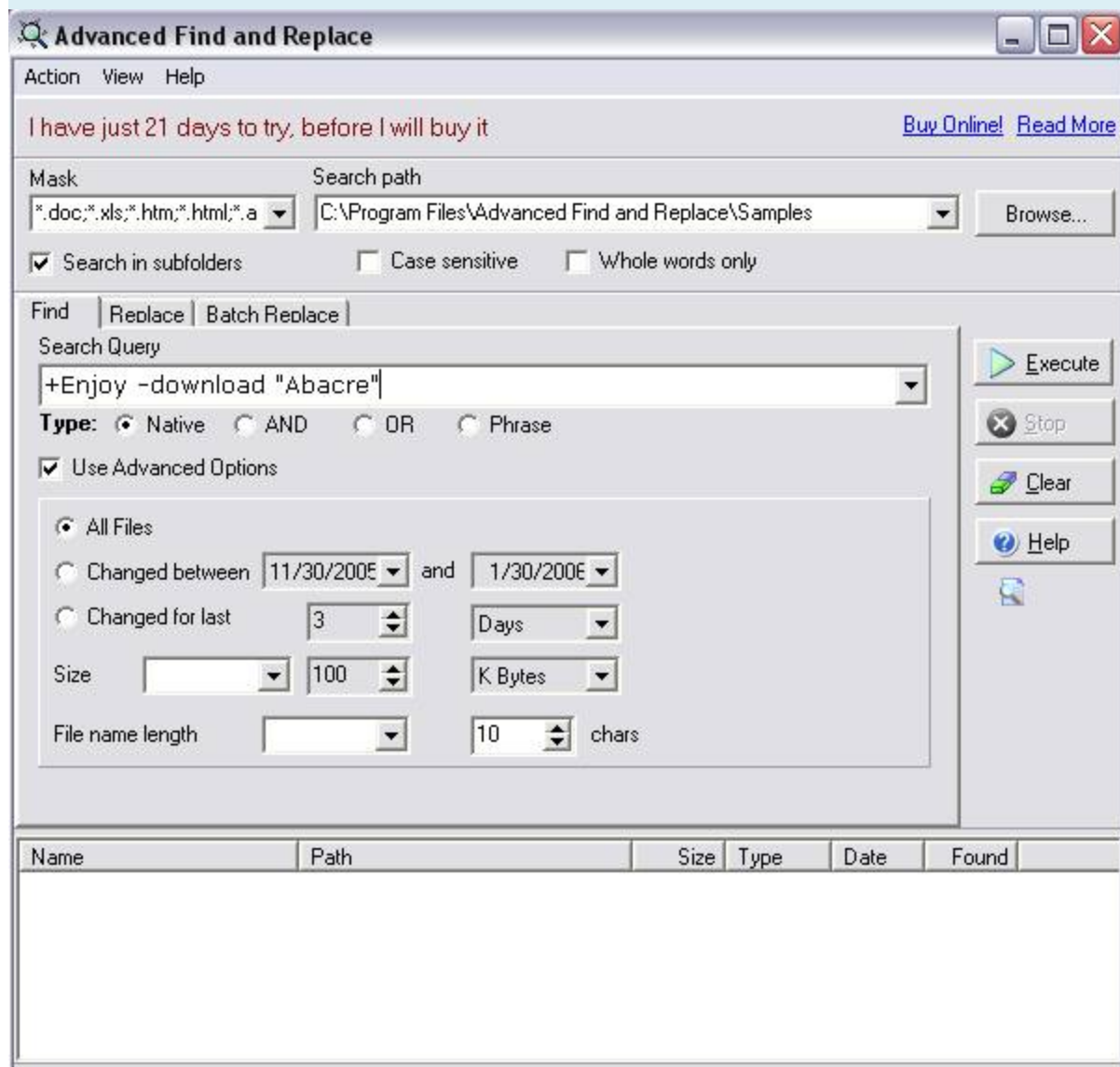
Trick for DUP (1 tools are quite common)

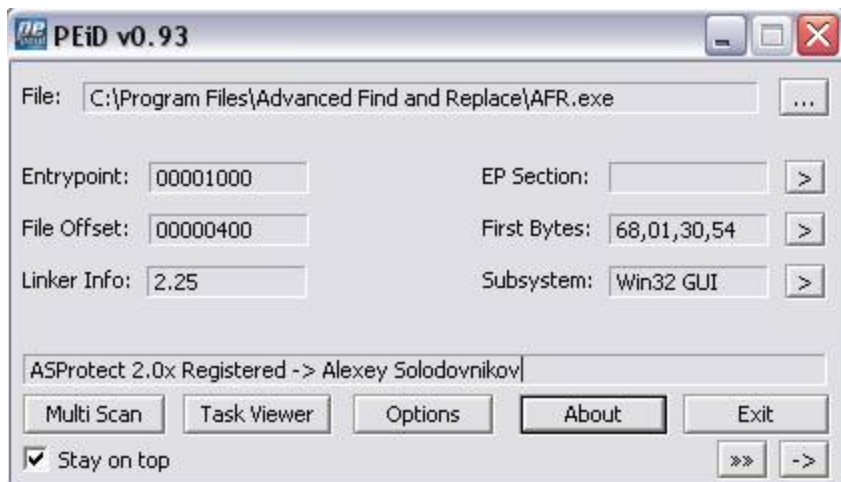
we will use 1 or features quite a memcheck "

see what it is. 1 short description of this feature as follows:



Preview we can find the address and value how =)
Target for this trick is Advanced Find and Replace 2.3





21 day trial are more aspr

First, I will show you the bytes need to patch program run as reg
(you are responsible for finding that the bytes in detail =))

- [CPU - main thread, module AFR]

File View Debug Plugins Options Window Help

004F0822 8B08 MOV ECX,DWORD PTR DS:[EAX]
 004F0824 FF51 68 CALL DWORD PTR DS:[ECX+68]
 004F0827 8B83 98030000 MOV EAX,DWORD PTR DS:[EBX+398]
 004F082D 8B10 MOV EDX,DWORD PTR DS:[EAX]
 004F082F FF52 50 CALL DWORD PTR DS:[EDX+50]
 004F0832 8BD0 MOV EDX,EAX
 004F0834 8B83 38040000 MOV EAX,DWORD PTR DS:[EBX+438]
 004F083A E8 1D6AF8FF CALL AFR.0047725C
 004F083F 8BCB MOV ECX,EBX
 004F0841 B2 01 MOV DL,1
 004F0843 A1 383B4E00 MOV EAX,DWORD PTR DS:[4E3B38]
 004F0848 E8 77ECF8FF CALL AFR.0047F4C4
 004F084D 8B15 84DD5000 MOV EDX,DWORD PTR DS:[50DD84]
 004F0853 8902 MOV DWORD PTR DS:[EDX],EAX
 004F0855 80BB F5050000 00 Cmp BYTE PTR DS:[EBX+5F5],0
 004F085C 0F84 8C000000 JE AFR.004F08EE
 004F0862 33D2 XOR EDX,EDX
 004F0864 8B83 F0030000 MOV EAX,DWORD PTR DS:[EBX+3F0]
 004F086A E8 5DF2F6FF CALL AFR.0045FACC
 004F086F 33D2 XOR EDX,EDX
 004F0871 8B83 CC030000 MOV EAX,DWORD PTR DS:[EBX+3CC]
 004F0877 E8 C86BF8FF CALL AFR.00477444
 004F087C 33D2 XOR EDX,EDX
 004F087E 8B83 50040000 MOV EAX,DWORD PTR DS:[EBX+450]
 004F0884 E8 BB6BF8FF CALL AFR.00477444
 004F0889 33D2 XOR EDX,EDX
 004F088B 8B83 D0030000 MOV EAX,DWORD PTR DS:[EBX+3D0]
 004F0891 E8 AE6BF8FF CALL AFR.00477444
 004F0896 33D2 XOR EDX,EDX
 004F0898 8B83 C8030000 MOV EAX,DWORD PTR DS:[EBX+3C8]
 004F089E E8 A16BF8FF CALL AFR.00477444
 004F08A3 A1 84DD5000 MOV EAX,DWORD PTR DS:[50DD84]
 004F08A8 8B00 MOV EAX,DWORD PTR DS:[EAX]
 004F08AA 8B80 48030000 MOV EAX,DWORD PTR DS:[EAX+348]
 004F08B0 33D2 XOR EDX,EDX
 004F08B2 E8 15F2F6FF CALL AFR.0045FACC
 004F08B7 8D55 F4 LEA EDX,DWORD PTR SS:[EBP-C]
 004F08BA A1 50DC5000 MOV EAX,DWORD PTR DS:[50DC50]
 004F08BF E8 78A6F1FF CALL AFR.0040AF3C
 004F08C4 8B4D F4 MOV ECX,DWORD PTR SS:[EBP-C]
 004F08C7 8D45 F8 LEA EAX,DWORD PTR SS:[EBP-8]
 004F08CA BA 9C0A4F00 MOV EDI,AFR.004F0A9C
 004F08CF E8 8849F1FF CALL AFR.0040525C
 004F08D4 8B55 F4 MOV ECX,DWORD PTR SS:[EBP-C]

AFR.0050FDB8

ASCII "Registered to: JJJ"

DS: [00072550]=00

004F0855 80BB F5050000 00 Cmp BYTE PTR DS: [EBX +5 F5], 0

Change the following

004F0855 80BB F5050000 01 Cmp BYTE PTR DS: [EBX +5 F5], 1

program has been reg =>

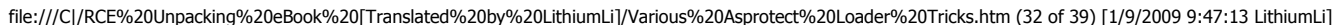
Now to the new attractions =>

We must find the address and the value of memcheck

Hum, you need time to think why I just set 1 working

breakpoint (how I've said in the first tut)

hihi, because it will be useful at this time =>.



003B0000	00001000			Priv	RWE	RWE
003C0000	00002000			Map	R	R
003D0000	00001000			Priv	RWE	RWE
003E0000	00001000			Priv	RWE	RWE
003F0000	00001000			Priv	RWE	RWE
00400000	00001000	AFR	PE header	Imag	R	RWE
00410000	0010B000	AFR	code	Imag	R	RWE
0050C000	00003000	AFR	data	Imag	R	RWE
0050F000	00001000	AFR		Imag	R	RWE
00510000	00003000	AFR		Imag	R	RWE
00513000	00001000	AFR		Imag	R	RWE
00514000	00001000	AFR		Imag	R	RWE
00515000	00001000	AFR		Imag	R	RWE
00516000	00010000	AFR		Imag	R	RWE
00526000	0001D000	AFR	.rsrc	resources	Imag	RWE
00543000	00022000	AFR	.data	imports,rel	Imag	RWE
00565000	00001000	AFR	.adata		Imag	RWE

- [CPU - main thread, module AFR]

File View Debug Plugins Options Window Help

L E M T W H C / K B R ... S


Address	Disassembly	Comment
00407412	C3	RETN
00407413	90	NOP
00407414	E8 E78B0C00	CALL 00CD0000
00407419	D9	Unknown command
0040741A	8BC0	MOV EAX,EAX
0040741C	-FF25 90025100	JMP DWORD PTR DS:[510290]
00407422	8BC0	MOV EAX,EAX
00407424	-FF25 8C025100	JMP DWORD PTR DS:[51028C]
0040742A	8BC0	MOV EAX,EAX
0040742C	-FF25 88025100	JMP DWORD PTR DS:[510288]
00407432	8BC0	MOV EAX,EAX
00407434	50	PUSH EAX
00407435	6A 40	PUSH 40
00407437	E8 E0FFFFFF	CALL AFR.0040741C
0040743C	C3	RETN
0040743D	8D40 00	LEA EAX,DWORD PTR DS:[EAX]
00407440	B8 34000000	MOV EAX,34
00407445	C3	RETN
00407446	8BC0	MOV EAX,EAX
00407448	53	PUSH EBX
00407449	E8 F2FFFFFF	CALL AFR.00407440
0040744E	8BD8	MOV EBX,EAX
00407450	85DB	TEST EBX,EBX
00407452	<74 36	JE SHORT AFR.0040748A
00407454	833D ECC05000 FF	CMP DWORD PTR DS:[50C0EC],-1
00407458	<75 0A	JNZ SHORT AFR.00407467
0040745D	B8 E2000000	MOV EAX,0E2
00407462	E8 DDAFFFFF	CALL AFR.00404F44
00407467	8BC3	MOV EAX,EBX
00407469	E8 C6FFFFFF	CALL AFR.00407434
0040746E	85C0	TEST EAX,EAX
00407470	<75 0C	JNZ SHORT AFR.0040747E
00407472	B8 E2000000	MOV EAX,0E2
00407477	E8 C8DAFFFF	CALL AFR.00404F44
0040747C	<EB 0C	JMP SHORT AFR.0040748A
0040747E	50	PUSH EAX
0040747F	A1 ECC05000	MOV EAX,DWORD PTR DS:[50C0EC]
00407484	50	PUSH EAX
00407485	E8 A2FFFFFF	CALL AFR.0040742C
0040748A	5B	POP EBX
0040748B	C3	RETN
0040748C	BA0D 64F65000	MOV CL,BYTE PTR DS:[50F664]
00407492	A1 ECC05000	MOV EAX,DWORD PTR DS:[50C0EC]
00407493	84C0	TEST CL,CL

kernel32.LocalAlloc
 kernel32.TlsGetValue
 kernel32.TlsSetValue
 JMP to kernel32.LocalAlloc
 JMP to kernel32.TlsSetValue



We are in the code of the program =). We have not to OEP

but i do, still OK.

Now, open the memory map and then set a breakpoint on memory write " section in the image data such as =)

 - [Memory map]

M File View Debug Plugins Options Window Help

 L E M T W H C / K B R ... S 

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00010000	00001000				Priv	RW	RW	
00020000	00001000				Priv	RW	RW	
0012B000	00001000				Priv	RW	Guar	
0012C000	00004000			stack of ma	Priv	RW	Guar	
00130000	00003000				Map	R	R	
00140000	0000E000				Priv	RW	RW	
00240000	00006000				Priv	RW	RW	
00250000	00003000				Map	RW	RW	
00260000	00016000				Map	R	R	\Device\HarddiskVolume1\WINDOWS\
00280000	0003D000				Map	R	R	\Device\HarddiskVolume1\WINDOWS\
002C0000	00041000				Map	R	R	\Device\HarddiskVolume1\WINDOWS\
00310000	00006000				Map	R	R	\Device\HarddiskVolume1\WINDOWS\
00320000	00001000				Priv	RWE	RWE	
00330000	00001000				Priv	RWE	RWE	
00340000	00001000				Priv	RW	RW	
00350000	00001000				Priv	RW	RW	
00360000	00005000				Priv	RW	RW	
00370000	00003000				Map	R	R	\Device\HarddiskVolume1\WINDOWS\
00380000	00004000				Priv	RW	RW	
00390000	00003000				Priv	RW	RW	
003A0000	00002000				Map	R	R	
003B0000	00001000				Priv	RWE	RWE	
003C0000	00002000				Map	R	R	
003D0000	00001000				Priv	RWE	RWE	
003E0000	00001000				Priv	RWE	RWE	
003F0000	00001000				Priv	RWE	RWE	
00400000	00001000	AFR		PE header	Imag	R	RWE	
00401000	0010B000	AFR		code	Imag	R	RWE	
0050C000	00003000	AFR		data				
0050F000	00001000	AFR						
00510000	00003000	AFR						
00513000	00001000	AFR						
00514000	00001000	AFR						
00515000	00001000	AFR						
00516000	00010000	AFR						
00526000	0001D000	AFR	.rsrc	reso				
00543000	00022000	AFR	.data	impo				
00565000	00001000	AFR	.adata					
00570000	00007000							
00630000	00002000							
00640000	00103000							
00750000	000F6000							
00A50000	00001000							
00A60000	00001000							
00A70000	00001000							
00A80000	00001000							
00A90000	00032000							
00AD0000	00004000							
00AD8000	0000C000							
00AE8000	00004000							
00BD0000	00051000							
00C30000	00001000							
00C40000	00001000							
00C50000	00001000							
00C60000	00001000				Priv	RWE	RWE	
00C70000	00001000				Priv	RWE	RWE	
00C80000	00001000				Priv	RWE	RWE	
00C90000	00001000				Priv	RWE	RWE	
00CA0000	00001000				Priv	RWE	RWE	
00CB0000	00003000				Priv	RWE	RWE	
00CC0000	00003000				Priv	RWE	RWE	
00CD0000	00001000				Priv	RWE	RWE	
00CF0000	00001000				Priv	RWE	RWE	

Actualize

Dump in CPU

Dump

Search Ctrl+B

Set break-on-access F2

Set memory breakpoint on access

Set memory breakpoint on write

Set access ▶

Copy to clipboard ▶

Sort by ▶

Appearance ▶

```

00CF0000 00001000
00D00000 00001000
00D10000 00001000
00D20000 00001000
00D30000 00001000

```

```

Priv RWE RWE
Priv RWE RWE
Priv RWE RWE
Priv RWE RWE

```

Press F9 and you will be here ..

```

- [CPU - main thread]
File View Debug Plugins Options Window Help
[Icons] [LEMTWHC/KBR...S] [Icons]

00CB0560 A3 F8C05000 MOV DWORD PTR DS:[50C0F8],EAX AFR.00400000
00CB0572 33C0 XOR EAX,EAX
00CB0574 A3 FCC05000 MOV DWORD PTR DS:[50C0FC],EAX
00CB0579 33C0 XOR EAX,EAX
00CB057B A3 00C15000 MOV DWORD PTR DS:[50C100],EAX
00CB0580 68 AC07CB00 PUSH 0CB07AC
00CB0585 E8 76FA0900 CALL 00D50000
00CB058A B9 62C64000 MOV ECX,40C662
00CB058F 334C24 28 XOR ECX,DWORD PTR SS:[ESP+28]
00CB0593 C1D1 FB RCL ECX,0FB Shift constant out of range 1..31
00CB0596 C1D1 D7 RCL ECX,0D7 Shift constant out of range 1..31
00CB0599 8D8C22 0B000000 LEA ECX,DWORD PTR DS:[EDX+B]
00CB05A0 2BCA SUB ECX,EDX
00CB05A2 F3:A5 REP MOVS DWORD PTR ES:[EDI],DWORD PTR D
00CB05A4 vE9 A4040000 JMP 00CB0A40
00CB05A9 68 DB06CB00 PUSH 0CB06DB
00CB05AE E8 4DFA0900 CALL 00D50000
00CB05B3 vEB 01 JMP SHORT 00CB05B6
00CB05B5 9A 83E873EB 01F3 CALL FAR F301:E873E883 Far call
00CB05BC 334424 28 XOR EAX,DWORD PTR SS:[ESP+28]
00CB05C0 FF35 B4DF5000 PUSH DWORD PTR DS:[50DFB4]
00CB05C6 81D8 46291000 SBB EAX,00102946
00CB05CC 58 POP EAX
00CB05CD vE9 24050000 JMP 00CB0AF6
00CB05D2 B8 2E6D4700 MOV EAX,476D2E
00CB05D7 B8 66434500 MOV EAX,454366
00CB05DC 8B07 MOV EAX,DWORD PTR DS:[EDI]
00CB05DE 89C6 MOV ESI,EAX
00CB05E0 33C0 XOR EAX,EAX
00CB05E2 8907 MOV DWORD PTR DS:[EDI],EAX
00CB05E4 FFD6 CALL ESI
00CB05E6 E8 15FA0900 CALL 00D50000
00CB05EB 26:EB 02 JMP SHORT 00CB05F0 Superfluous prefix
00CB05EE CD 20 INT 20
00CB05F0 8BE5 MOV ESP,EBP
00CB05F2 ^E9 3FFDFFFF JMP 00CB0336
00CB05F7 5F POP EDI
00CB05F8 5E POP ESI
00CB05F9 5B POP EBX
00CB05FA C3 RETN
00CB05FB 31C0 XOR EAX,EAX
00CB05FD 8705 00C05000 XCHG DWORD PTR DS:[50C000],EAX
00CB0603 F7D8 NEG EAX
00CB0605 10C0 CDB EBX,EBX

EAX=00400000 (AFR.00400000), ASCII "M2P"
DS:[0050C0F8]=00000000

```

hehe try dom considered = D.

00400000 written at 0050C0F8 this is done

after aspr completed CRC check =) so we will use this value for DUP =)

Look down below pictures to see how to create loader =)

file:///C:/RCE%20Unpacking%20eBook%20[Translated%20by%20LithiumLi]/Various%20Asprotect%20Loader%20Tricks.htm (36 of 39) [1/9/2009 9:47:13 LithiumLi]

About 1 second but I forgot to mention at the top
is a program to crack 1 asprotect trial
I will go through with basic 1 items

file:///C:/RCE%20Unpacking%20eBook%20[Translated%20by%20LithiumLi]/Various%20Asprotect%20Loader%20Tricks.htm (37 of 39) [1/9/2009 9:47:13 LithiumLi]

It uses asprotected trial, you crack how it addresses below may be different in your computer to DVD Catalyst.
 exe Load Olly then press Shift + F9. If there are exceptions, the press Shift-F9 until you notice " sorry your 5 day
 trial has expired "
 Now press F12 and then press ALT + K to call call stack window. It will be like after

```
Call stack of main thread
Address Stack Procedure / arguments Called from Frame
0012F934 77D493F5 Includes ntdll.KiFastSystemCallRet USER32.77D493F3 0012F968
0012F938 77D6EA24 USER32.WaitMessage USER32.77D6EA1F 0012F968
0012F96C 77D5688A USER32.77D6E895 USER32.77D56885 0012F968
0012F994 77D6B7C5 USER32.77D567D4 USER32.77D6B7C0 0012F990
0012FC54 77D6B12B USER32.SoftModalMessageBox USER32.77D6B126 0012FC50
0012FDA4 77D95FDF USER32.77D6AFB6 USER32.77D95FDA 0012FDA0
0012FDFC 77D96084 USER32.MessageBoxTimeoutW USER32.77D9607F 0012FDF8
0012FE30 77D80598? USER32.MessageBoxTimeoutA USER32.77D80593 0012FE2C
0012FE50 77D80550? USER32.MessageBoxExA USER32.77D8054B 0012FE4C
0012FE54 00000000 hOwner = null
0012FE58 00C2D231 Text = "Sorry, but your 5-day tria
0012FE5C 00C2D287 title = "Trial Version"
0012FE60 00000010 Style = MB_OK | MB_ICONHAND | MB_APPLM
0012FE64 00000000 LanguageID = 0 (LANG_NEUTRAL)
0012FE6C 00BF53DB? 00BE587C 00BF53D6 0012FE68
0012FE70 00000000 hOwner = null
0012FE74 00C2D231 Text = "Sorry, but your 5-day tria
0012FE78 00C2D287 title = "Trial Version"
0012FE7C 00000010 Style = MB_OK | MB_ICONHAND | MB_APPLM
0012FE88 00BF54C2? 00BF53C800BF54BD -----> Follow Call to function in Olly 0012FE84
0012FF24 00BFBE85 00BF5454 00BFBE80 0012FF78
0012FE88 00BF54C2? 00BF53C8 00BF54BD
```

right click on the line and then select follow in disassembler. You will be here

```
00BF5487 0F85 8C000000 JNZ 00BF5519
00BF548D 807C24 65 00 Cmp BYTE PTR SS: [ESP +65], 0
00BF5492 75 07 JNZ SHORT 00BF549B
00BF5494 807C24 64 00 Cmp BYTE PTR SS: [ESP +64], 0
00BF5499 74 65 JE SHORT 00BF5500
00BF549B 807C24 65 00 Cmp BYTE PTR SS: [ESP +65], 0
00BF54A0 75 10 JNZ SHORT 00BF54B2
00BF54A2 66:837 C24 40 00 Cmp WORD PTR SS: [ESP +40], 0
00BF54A8 75 1A JNZ SHORT 00BF54C4
```

```
00BF54AA 66:837 C24 3C 00 Cmp WORD PTR SS: [ESP +3 C], 0
00BF54B0 74 12 JE SHORT 00BF54C4
00BF54B2 8B5424 6D MOV EDX, DWORD PTR SS: [ESP +6 D]
00BF54B6 8B8424 85000000 MOV EAX, DWORD PTR SS: [ESP +85]
00BF54BD E8 06FFFFFF CALL 00BF53C8
00BF54C2 EB 55 JMP SHORT 00BF5519
00BF54C4 66:837 C24 46 00 Cmp WORD PTR SS: [ESP +46], 0
00BF54CA 75 1A JNZ SHORT 00BF54E6
```

Now set at 1 bp hardware BF5494 then restart and run the program in Olly
it will break in the hardware you set bp

Please change 00BF5494 807C24 64 00 Cmp BYTE PTR SS: [ESP +64], 0

to

00BF5494 807C24 64 01 Cmp BYTE PTR SS: [ESP +64], 1

then press F9, hehe, run a very good program. Aspr trial was defeate
loader to enter the address and bytes to the DUP or enter ABEL
the CRC check bytes tut because of me or because aspr CRC
check = locator)

Well thats it from my side.
Cya all and take care =)

Closing
words

My main emphasis here was not on any patching asprotect appbut to show the different methods =)

Greetz fly out to all snd, ICU, Mp2K, ARTEAM, TSRh Team Members
and special greets to Cordat, d2k2, Predator for their outstanding tools
=)

Adios
slayer / / snd

Takada

Armadillo collect sand-stone

Warecase (TM) extended Task Manager (TM) <|> ARM 4.xx - Standard Protection + Code Splicing + IAT elimination



Product Name	License Type	Product ID	Price, USD	Online Store
eXtended Task Manager for Windows 2000/XP/2003	Home	10786-1	\$55.00	Order Now!
	Enterprise	10786-6	\$85.00	Order Now!

C laos this one can unpack, but I do tuts to introduce new tools: D.

1. Tools:

_OillyDBG Vs Armadillo MOD by **hacnho** (Version 0.2 Fixed OutputDebugStringW).
 _LordPE 1.4 Deluxe **kanxue** by MOD
 _ImpREC 1.6 Final by **MaRKuS_TH MOD-DJM**

2. Unpack

_Load Target:

```

var GetModuleHandleA
var AddressOfMagicJump
var LenOfMagicJump

GPA "GetModuleHandleA", "kernel32.dll"
mov GetModuleHandleA, $ RESULT

bphws GetModuleHandleA, "x"
repeat:
esto
rtu
find eip, # 0F84 ???????????????????? 74 ?????????? EB? #
Cmp $ result, 0
je repeat
bphwc GetModuleHandleA

mov AddressOfMagicJump, $ RESULT
mov LenOfMagicJump, AddressOfMagicJump
add LenOfMagicJump, 2
mov LenOfMagicJump, [LenOfMagicJump]
inc LenOfMagicJump
mov [AddressOfMagicJump], 0E9
inc AddressOfMagicJump
mov [AddressOfMagicJump], LenOfMagicJump

```


CMT \$ result, "<- MagicJump fixed"

msg "MagicJump fixed! Now, the next step is yours: OEP Finder!"

ret

_Sau When to run here:

Address	Hex dump	Disassembly	Comment
003C4CE4	8B0D 949D3E00	mov ecx, dword ptr ds:[3E9D94]	kernel32.77E60000
003C4CEA			
003C4CED			
003C4CF2			
003C4CF5			
003C4CF7			
003C4CFD			kernel32.77E60000
003C4CFE			kernel32.LoadLibraryA
003C4D04			
003C4D0A			kernel32.77E60000
003C4D0D	A1 949D3E00	mov eax, dword ptr ds:[3E9D94]	
003C4D12	393C06	cmp dword ptr ds:[esi+eax], edi	
003C4D15	E9 30010000	jmp 003C4E4A	<- MagicJump fixed
003C4D1A	0033	add byte ptr ds:[ebx], dh	
003C4D1C	C9	leave	
003C4D1D	8B03	mov eax, dword ptr ds:[ebx]	
003C4D1F	3938	cmp dword ptr ds:[eax], edi	
003C4D21	74 06	je short 003C4D29	



_BP CreateThread, F9, Ctrl + F9, F8, Ctrl + F9, F8:

Address	Hex dump	Disassembly	Comment
003D7630	FF76 10	push dword ptr ds:[esi+10]	
003D7633	2BCA	sub ecx, edx	
003D7635	FFD1	call near ecx	kernel32.77E7BE2B
003D7637	EB 1D	jmp short 003D7656	
003D7639	83FA 01	cmp edx, 1	
003D763C	75 1A	jnz short 003D7658	
003D763E	FF76 04	push dword ptr ds:[esi+4]	
003D7641	8B50 7C	mov edx, dword ptr ds:[eax+7C]	
003D7644	3350 74	xor edx, dword ptr ds:[eax+74]	
003D7647	FF76 08	push dword ptr ds:[esi+8]	
003D764A	3350 04	xor edx, dword ptr ds:[eax+4]	
003D764D	6A 00	push 0	
003D764F	FF76 0C	push dword ptr ds:[esi+C]	XTM.00400000
003D7652	2BCA	sub ecx, edx	
003D7654	FFD1	call near ecx	kernel32.77E7BE2B
003D7656	8BD8	mov ebx, eax	
003D7658	5F	pop edi	
003D7659	8BC3	mov eax, ebx	

_F9, F7: OEP:

Address	Hex dump	Disassembly	Comment
0050C720	55	push ebp	
0050C721	8BEC	mov ebp, esp	
0050C723	6A FF	push -1	
0050C725	68 082A5300	push XTM.00532A08	
0050C72A	68 8AC85000	push XTM.0050C88A	jmp to msvcrt._except.
0050C72F	64:A1 00000000	mov eax, dword ptr fs:[0]	
0050C735	50	push eax	XTM.005C3370
0050C736	64:8925 00000000	mov dword ptr fs:[0], esp	
0050C73D	83EC 68	sub esp, 68	
0050C740	53	push ebx	
0050C741	56	push esi	XTM.005C9E08
0050C742	57	push edi	
0050C743	8965 E8	mov dword ptr ss:[ebp-18], esp	
0050C746	33DB	xor ebx, ebx	
0050C748	895D FC	mov dword ptr ss:[ebp-4], ebx	
0050C74B	6A 02	push 2	
0050C74D	FF15 3413EA00	call dword ptr ds:[EA1334]	msvcrt.__set_app_type
0050C753	59	pop ecx	003D7656

IAT Elimination:

Address	Hex dump	Disassembly	Comment
0043CBFF	90	nop	
0043CBF0	C705 74865500	mov dword ptr ds:[558674], XTM.0051F	
0043CBFA	C3	retn	
0043CBFB	90	nop	
0043CBFC	90	nop	
0043CBFD	90	nop	
0043CBFE	90	nop	
0043CBFF	90	nop	
0043CC00	- FF25 C01DEA00	jmp dword ptr ds:[EA1DC0]	XTMExtUI.CHookDialog:
0043CC06	- FF25 981CEA00	jmp dword ptr ds:[EA1C98]	XTMExtUI.CColorListVi.
0043CC0C	- FF25 8C1CEA00	jmp dword ptr ds:[EA1C8C]	XTMExtUI.CColorListVi.
0043CC12	- FF25 881CEA00	jmp dword ptr ds:[EA1C88]	XTMExtUI.CColorListVi.
0043CC18	- FF25 801CEA00	jmp dword ptr ds:[EA1C80]	XTMExtUI.CColorListVi.
0043CC1E	- FF25 7C1CEA00	jmp dword ptr ds:[EA1C7C]	XTMExtUI.CColorListVi.
0043CC24	- FF25 B41BEA00	jmp dword ptr ds:[EA1BB4]	XTMExtUI.CHookFrameWn.
0043CC2A	CC	int3	
0043CC2B	CC	int3	
0043CC2C	CC	int3	

IAT Start:

Address	Value	Comment
00EA12D4	00080102	
00EA12D8	7712174B	OLEAUT32.SysFreeString
00EA12DC	77123D8F	OLEAUT32.SysStringLen
00EA12E0	73DD6784	MFC42.#324_CDialog::CDialog
00EA12E4	73E6067B	MFC42.#952_??_afxSOCK@@@3UAFX_SOCKET_CAL
00EA12E8	77121650	OLEAUT32.SysAllocString
00EA12EC	003C5DED	

IAT End:

Address	Value	Comment
00EA2DF0	00060009	
00EA2DF4	00080102	
00EA2DF8	77C10000	msvcrt.77C10000
00EA2DFC	771B0000	OLE32.771B0000
00EA2E00	8000101B	
00EA2E04	80000ED5	
00EA2E08	800012F6	

Splicing Code:

Memory map					
Address	Size	Owner	Section	Contains	Type
015C6000	00001000	XTMEngn	.rsrc	resources	Image
015C7000	00006000	XTMEngn	.reloc	relocations	Image
015D0000	00020000				Private
015F0000	00001000	XTMGen		PE header	Image
015F1000	00006000	XTMGen	.text	code	Image
015F7000	00004000	XTMGen	.rdata	imports, exports	Image
015FB000	00002000	XTMGen	.data	data	Image

Dump - 015D0000..015EFFFF					
015D0000	55	push	ebp		
015D0001	8BEC	mov	ebp, esp		
015D0003	51	push	ecx		
015D0004	53	push	ebx		
015D0005	EB 00	jmp	short 015D0007		
015D0007	66:87CB	xchg	bx, cx		
015D000A	F7D7	not	edi		
015D000C	F7D7	not	edi		
015D000E	66:87CB	xchg	bx, cx		
015D0011	5B	pop	ebx		
015D0012	53	push	ebx		
015D0013	56	push	esi		
015D0014	57	push	edi		
015D0015	894D FC	mov	dword ptr ss:[ebp-4], ecx		
015D0018	- E9 381BE3FE	jmp	XTM.00401B55		
015D001D	66:87F2	xchg	dx, si		
015D0020	70 02	ja	short 015D0024		
015D0022	70 27	ja	short 015D004B		
015D0024	66:97	xchg	ax, di		
015D0026	F7D3	not	ebx		

Sections:

Memory map					
Address	Size	Owner	Section	Contains	Type
00400000	00001000	XTM			Image
00401000	0011D000	XTM	.text		Image
0051E000	0003A000	XTM	.rdata		Image
00558000	0000B000	XTM	.data		Image
00563000	00050000	XTM	.text1	code	Image
005B3000	00010000	XTM	.adata		Image
005C3000	00020000	XTM	.data1	data,imports	Image

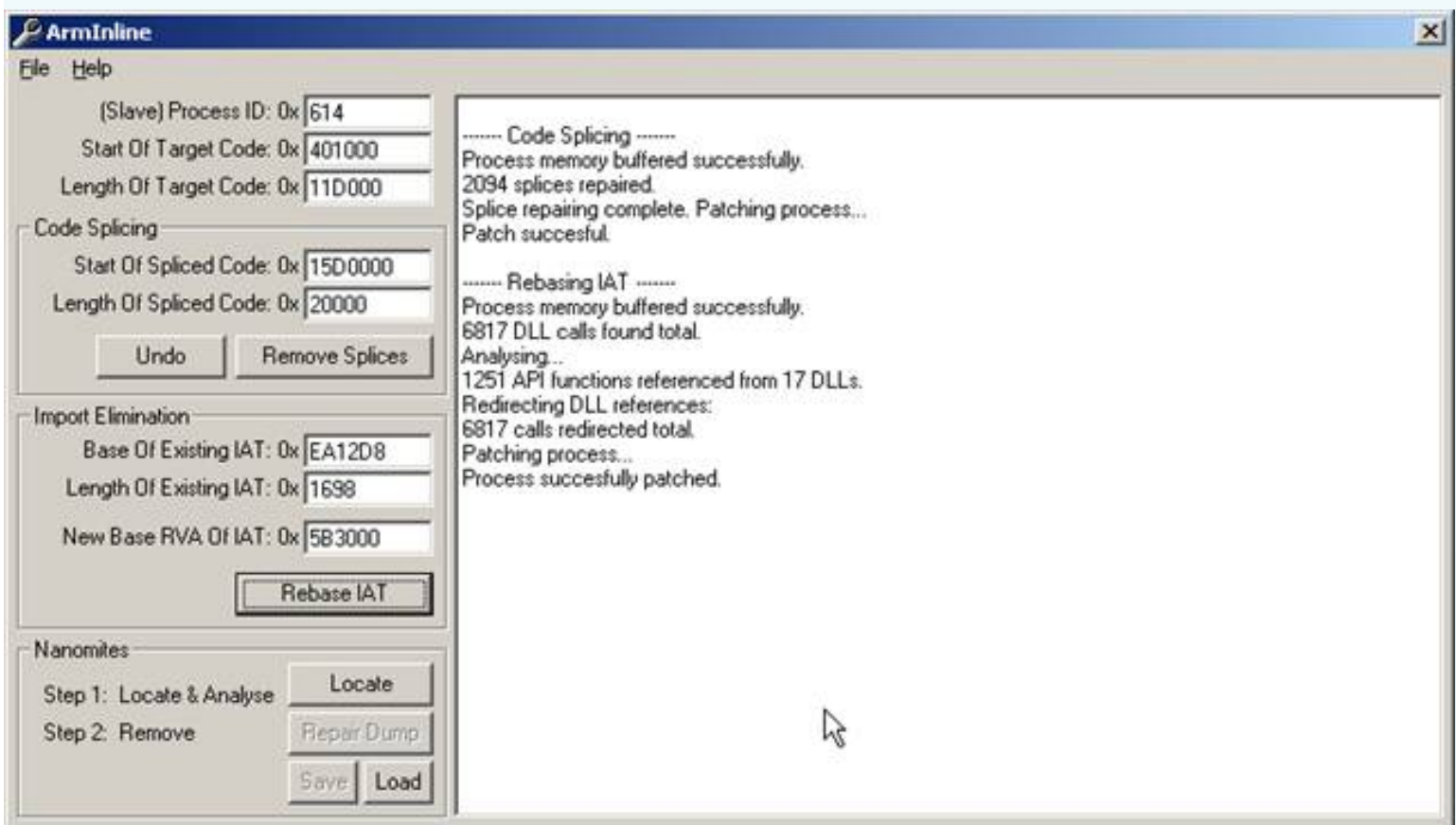
_PID

Select process to attach			
Process	Name	Window	Path
00000200	csrss		\\??\C:\WINDOWS\system32\csrss.exe
00000218	winlogon		\\??\C:\WINDOWS\system32\winlogon.exe
00000244	services	NetDDE Agent	C:\WINDOWS\system32\services.exe
00000250	lsass		C:\WINDOWS\system32\lsass.exe
000002F8	svchost		C:\WINDOWS\system32\svchost.exe
00000324	svchost		C:\WINDOWS\System32\svchost.exe
00000358	svchost		C:\WINDOWS\System32\svchost.exe
000003D0	Snagit32	Snagit Capture Preview	C:\Program Files\TechSmith\Snagit 7\Snagit32.exe
000003D8	spoolsv		C:\WINDOWS\system32\spoolsv.exe
00000420	svchost		C:\WINDOWS\System32\svchost.exe
00000440	DkService		C:\Program Files\Executive Software\Diskeeper\Diskeeper.exe
00000480	StarWind		C:\Program Files\Alcohol Soft\Alcohol 120\StarWind.exe
0000049C	wdfmgr		C:\WINDOWS\System32\wdfmgr.exe
00000614	XTM		C:\Program Files\Warecase\XTM 1.90\XTM.exe
00000648	TSCHelp	C:\Program Files\TechSmith	C:\Program Files\TechSmith\Snagit 7\TSCHelp.exe
000006D8	Explorer	SysFader	C:\WINDOWS\Explorer.EXE
00000710	UniKeyNT	UniKey 3.62	C:\Program Files\UniKey\UniKeyNT.exe
00000718	ctfmon	TF_FloatingLangBar_WndTitl	C:\WINDOWS\System32\ctfmon.exe
00000734	TurboLau	TurboLaunch	C:\Program Files\TurboLaunch\TurboLaunch.exe

Attach

Cancel

0.71 _ArmInline:



[_OillyDBG:](#)

OllyDbg - XTM.exe - [CPU - main thread, module XTM]

File View Debug Plugins Options Window Help

Paused

Address Hex dump Disassembly Comment Registers (FPU)

Address	Hex dump	Disassembly	Comment	Registers (FPU)
0050C720	55	push ebp		EAX 005C3370 XTH.005C3370
0050C721	8BEC	mov ebp, esp		ECX 0050C720 XTH.0050C720
0050C723	6A FF	push -1		EDX 676AAD1E
0050C725	68 082A5300	push XTH.00532A08		EBX FFFFFFFF
0050C72A	68 8AC85000	push XTH.0050C88A	jap to msvcrt._except.	ESP 0012D794
0050C72F	64:A1 00000000	mov eax, dword ptr fs:[0]		EBP 0012DF14
0050C735	50	push eax	XTH.005C3370	ESI 005C9E08 XTH.005C9E08
0050C736	64:8925 00000000	mov dword ptr fs:[0], esp		EDI 003E58C0
0050C73D	83EC 68	sub esp, 68		EIP 0050C720 XTH.0050C720
0050C740	53	push ebx		C 0 ES 0023 32bit 0(F)
0050C741	56	push esi	XTH.005C9E08	P 0 CS 001B 32bit 0(F)
0050C742	57	push edi		A 0 SS 0023 32bit 0(F)
0050C743	8965 E8	mov dword ptr ss:[ebp-18], esp		Z 0 DS 0023 32bit 0(F)
0050C746	33DB	xor ebx, ebx		S 0 FS 003B 32bit 7(F)
0050C748	895D FC	mov dword ptr ss:[ebp-4], ebx		T 0 GS 0000 NULL
0050C74B	6A 02	push 2		D 0
0050C74D	FF15 FC405B00	call dword ptr ds:[5B40FC]	msvcrt._set_app_type	0 0 LastErr ERROR_FILE_NOT_FOUND
0050C753	59	pop ecx	003D7656	EFL 00000202 (NO,NB,NE)

ds:[005B40FC]=77C33632 (msvcrt._set_app_type)

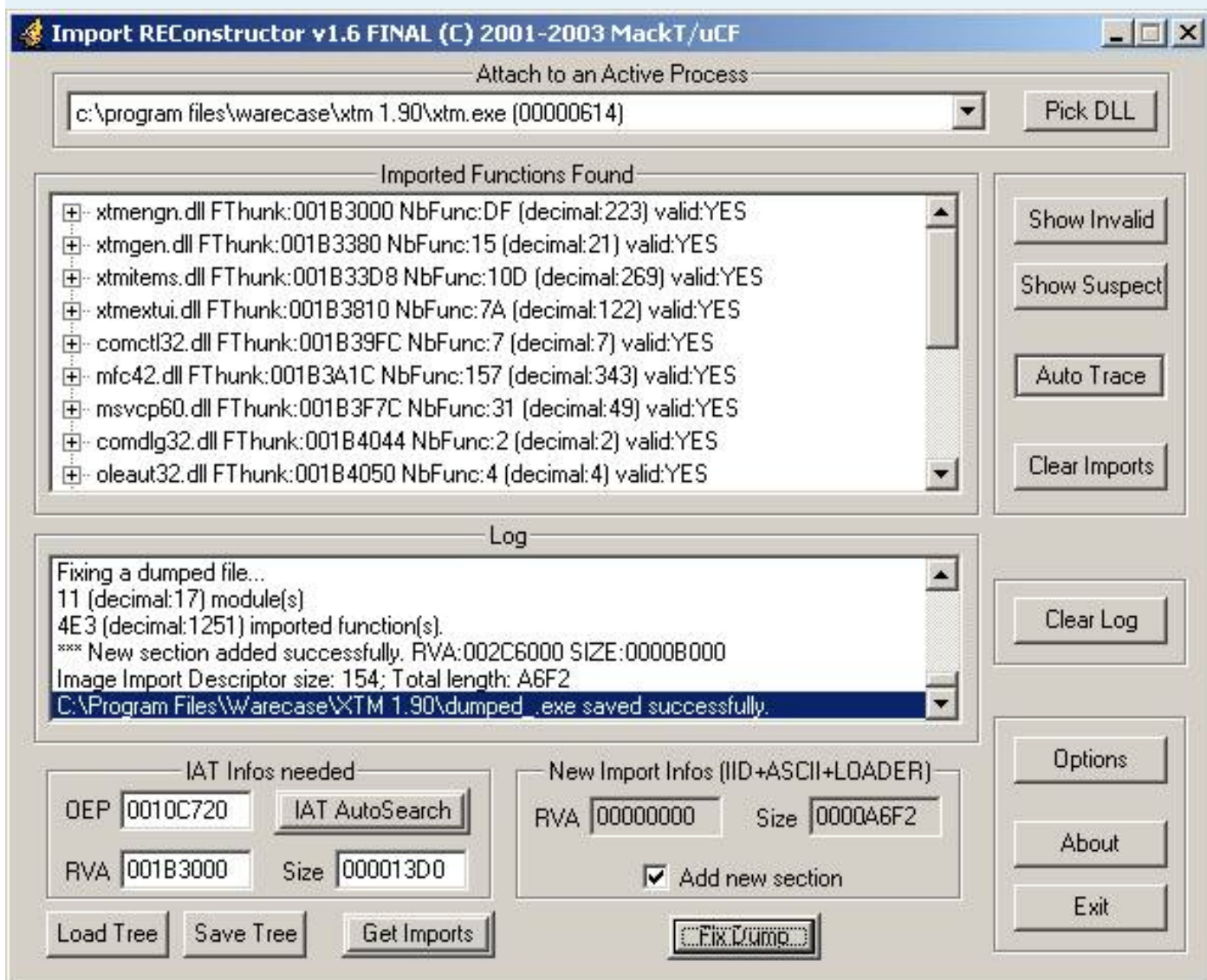
Address	Value	Comment	Address	Value	Comment
00EA2D70	00060009		0012D794	003D7656	RETURN to 003D7656
00EA2D74	00080102		0012D798	00400000	XTH.00400000
00EA2D78	77C10000	msvcrt.77C10000	0012D79C	00000000	
00EA2D7C	771B0000	OLE32.771B0000	0012D7A0	00151F0C	
00EA2E00	8000101B		0012D7A4	0000000A	
00EA2E04	80000ED5		0012D7A8	0012FF2C	
00EA2E08	800012F6		0012D7AC	00000000	

Command: bp CreateThread BP address, string -- Break with condition

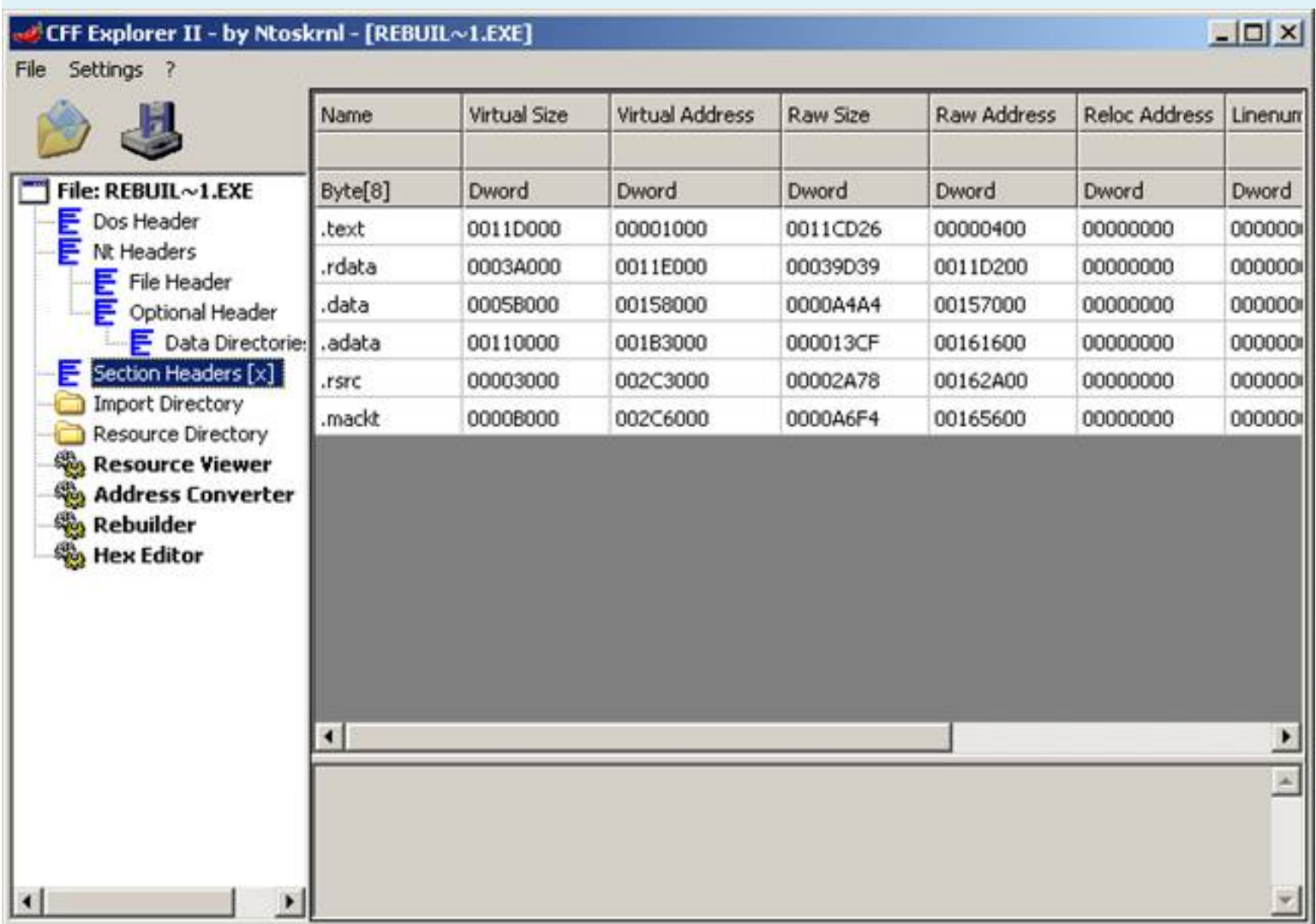
_LordPE:

Path	PID	ImageBase	ImageSize
c:\program files\alcohol soft\alcohol 120\star...	00000480	00400000	0003A000
c:\windows\system32\wdfmgr.exe	0000049C	01000000	0000C000
c:\windows\explorer.exe	000006D8	01000000	000F8000
c:\program files\unikey\unikeynt.exe	00000710	00400000	00034000
c:\windows\system32\ctfmon.exe	00000718	00400000	00006000
c:\program files\microsoft office\office11\win...	00000140	30000000	008AA000
c:\program files\techsmith\snagit 7\snagit32...	000003D0	00400000	00390000
c:\program files\techsmith\snagit 7\tschelp.e...	00000648	00400000	0000A000
c:\program files\winamp\winamp.exe	000007DC	00400000	00121000
i:\cracker\debug-disassembler\ollydbg\hacn...	000000F8	00400000	00169000
c:\program files\...		00400000	002C6000
i:\cracker\unpac...		00400000	00016000
c:\program files\...		00400000	001A0000
i:\cracker\utilities		00400000	00035E28
Path		ImageSize	
c:\program files\w...		002C6000	
c:\windows\sys...		000A7000	
c:\windows\sys...		000E6000	
c:\windows\sys...		0008C000	
c:\windows\sys...		00040000	
c:\windows\sys...		0008D000	
c:\windows\sys...		00086000	
c:\windows\sys...		0001C000	
c:\windows\system32\lpk.dll	629C0000	00008000	
c:\windows\system32\usp10.dll	72FA0000	0005A000	
c:\windows\winsxs\x86_microsoft.windows.c...	71950000	000E4000	
c:\windows\system32\msvcrt.dll	77C10000	00053000	
c:\windows\system32\shlwapi.dll	70A70000	00064000	
c:\windows\system32\oleaut32.dll	77120000	00088000	

_ ImpREC:



[_Rebuild:](#)



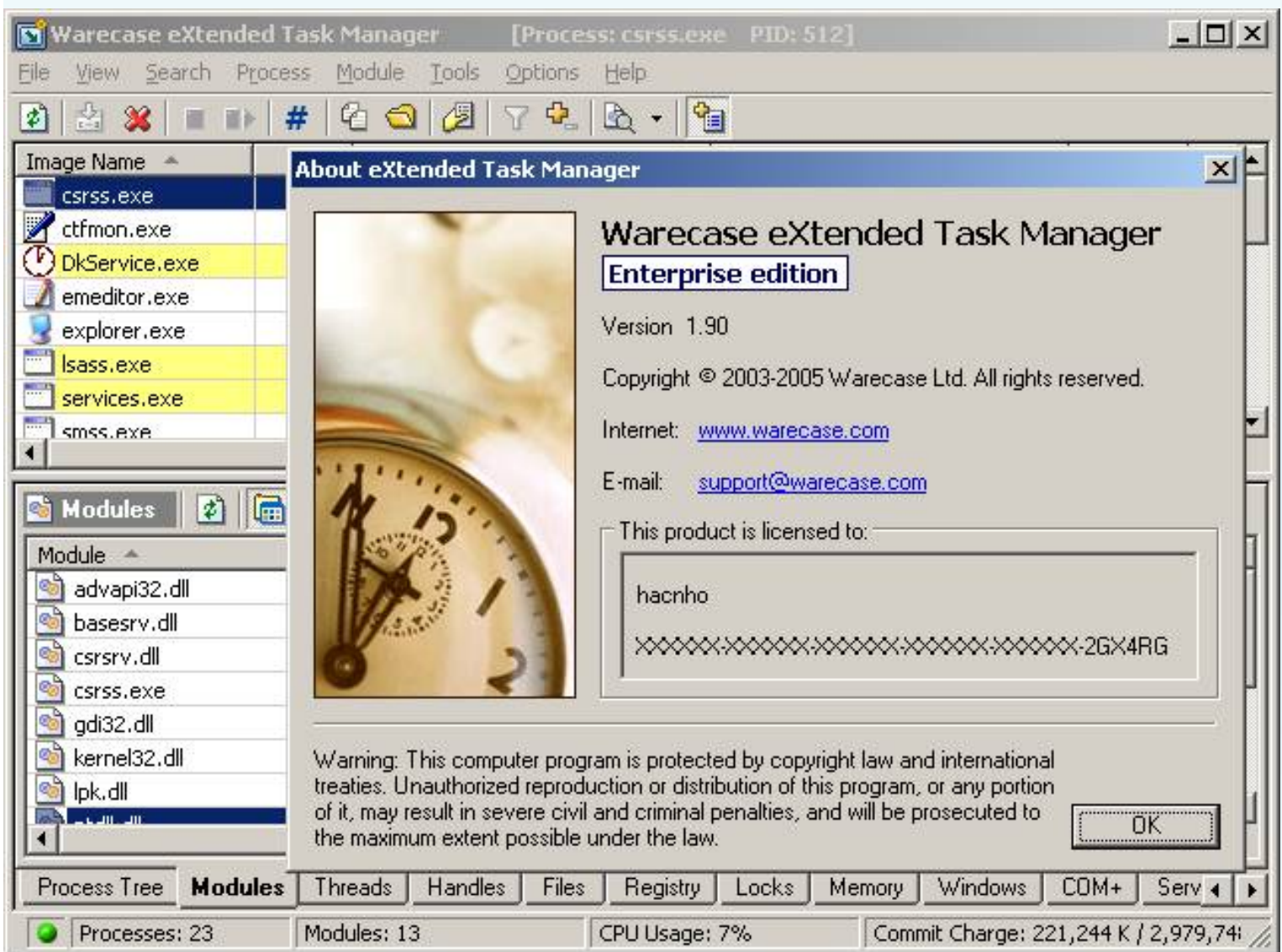
_Run Try:



Warecase
eXtended Task Manager
 Version 1.90
 Enterprise edition

Copyright © 2003-2005 Warecase Ltd. All rights reserved.
 Warecase™ is either a trademark or registered trademark of
 Warecase Company.

www.warecase.com



_Unpacked.Cracked.by.hacnho

附言:如果您在我的网站喜欢有些讲解, 与我联系! 我为您将翻译!

嘿儒: hacnho in Chinese.

GrEeTs Fly Out: Deux, infinite, Computer_Angel, Zombie, NVH (c), softcracker_vn, luucorp, Aaron, JMI, Canterwood, hhphong, R @ dier, tlandn, RCA, CTL, Moonbaby, kienmanowar, benina, TQN, the_lighthouse, Nini , hoadongnoi, dqtlm, hosiminh, Nilrem, fly, MaDMAn_H3rCul3s, Teerayoot, Ferrari, Kruger, Kelvin, Devilz, NXL, Phoenix light, iamidiot, WhyNotBar, trickyboy ... and you!

Special Thanx Cracks Latinos.

Thanx OillyDBG of the authors.

To be continued ...

Written by [hacnho](#) (tutorial date: VietNam 15/10/2005)

Sa they are a nui

One lost a child

Miraculous united Lu

Integration hunger Giang Ho

Yoda's Protector v1.02 - manually unpacking tutorial

(_kienmanowar_)

I. Introduction:

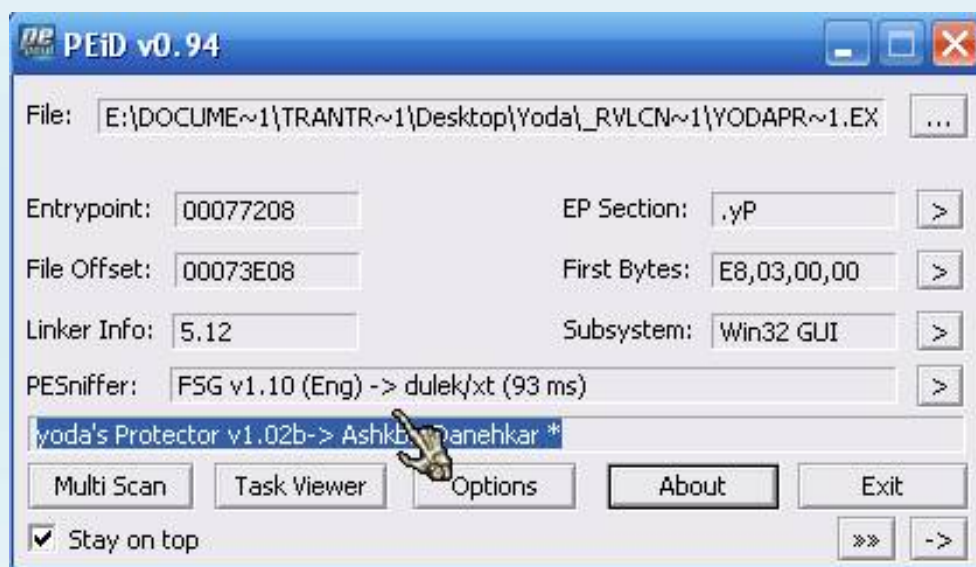
Hi all! To continue with a series of messages about how **MUP Yoda's Protector**, today I will be with you with a different version is **v1.02**. In the first part and I **Whynotbar** provided for how the number 1 on the latest versions of Yoda's. Version of this article, though quite old, but by the interests of my collection like I should write this tut (**based on the tut of the TBN + NCR**), and more important, though it is old but when many hiii not be touched sure we know how MUP J .

II. Tools & Target:

- _ *OllyDbg_final [Fixed by hacnho]*
- _ *PeiD v0.94*
- _ *RDG Packer Detector*
- _ *LordPE*
- _ *ImportRec v1.6*
- _ *Target is protected by Yoda's Protector v1.02*

III. MUPing:

_Dung **PeiD** and **RDG Packer Detector** to get information on target:

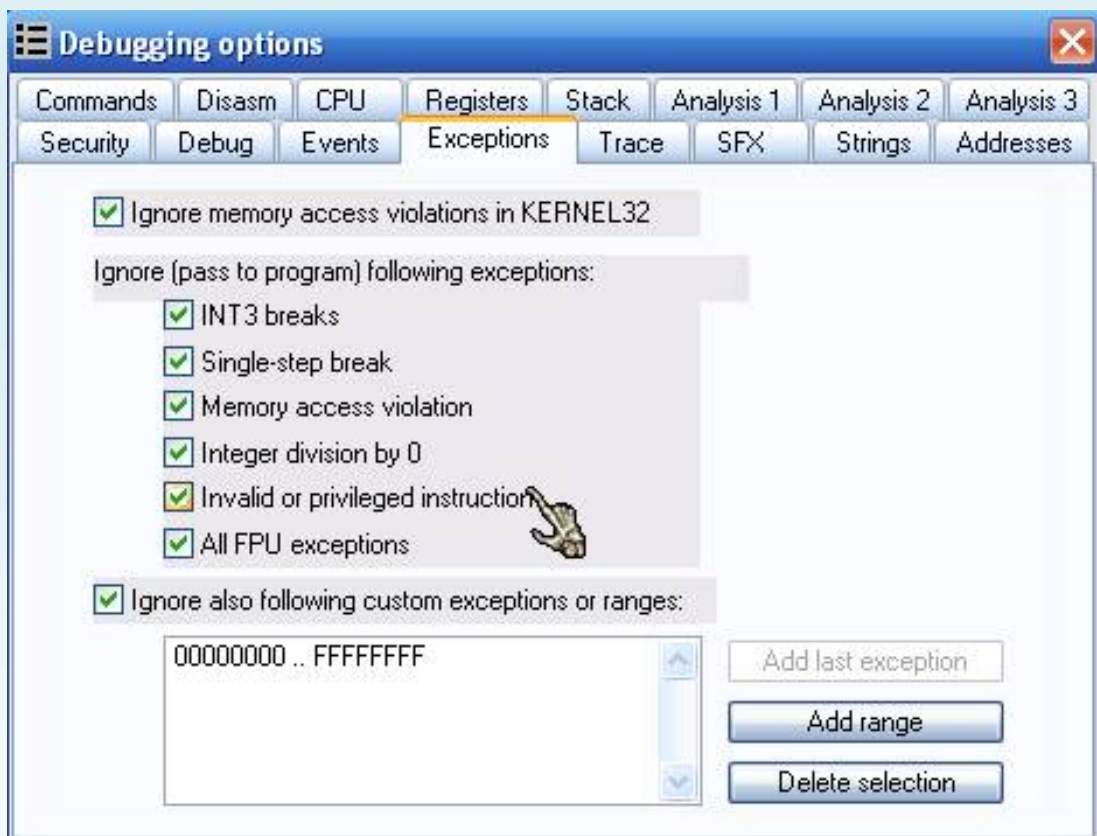




Oilly _Mo and load up on target in Olly. We stopped here in Olly:

Address	Hex dump	Disassembly	Comment
00477208	E8 03000000	CALL YodaProt.00477210	
0047720D	EB 01	JMP SHORT YodaProt.00477210	
0047720F	E8 BB550000	CALL 0047C7CF	
00477214	00E8	ADD AL, CH	
00477216	0300	ADD EAX, DWORD PTR DS:[EAX]	
00477218	0000	ADD BYTE PTR DS:[EAX], AL	
0047721A	EB 01	JMP SHORT YodaProt.0047721D	
0047721C	E9 E88F0000	JMP 00480209	
00477221	00E8	ADD AL, CH	
00477223	0300	ADD EAX, DWORD PTR DS:[EAX]	
00477225	0000	ADD BYTE PTR DS:[EAX], AL	
00477227	EB 01	JMP SHORT YodaProt.0047722A	
00477229	E9 E8820000	JMP 0047F516	
0047722B	00E8	ADD AL, CH	
00477230	0300	ADD EAX, DWORD PTR DS:[EAX]	
00477232	0000	ADD BYTE PTR DS:[EAX], AL	
00477234	EB 01	JMP SHORT YodaProt.00477237	
00477236	E8 E8B80000	CALL 00482B23	
0047723B	00E8	ADD AL, CH	
0047723D	0300	ADD EAX, DWORD PTR DS:[EAX]	
0047723F	0000	ADD BYTE PTR DS:[EAX], AL	
00477241	EB 01	JMP SHORT YodaProt.00477244	
00477243	E8 E8AB0000	CALL 00481E30	
00477248	00E8	ADD AL, CH	
0047724A	0300	ADD EAX, DWORD PTR DS:[EAX]	
0047724C	0000	ADD BYTE PTR DS:[EAX], AL	

_Do The **EP** (Entry Point) by Packer. The goal of us now is how to find the OEP, as well as the dump file and Rebuilt again with IAT. First we will find the way to the OEP, to do this is to configure the **Exceptions** in Olly as follows:



_Ok Now press **F9** to implement the program, immediately, we will use exception to the first:



_Khong Have anything to worry about both, you just click OK to accept the notice then press **Shift + F9** to bypass no.Chung we will stop here in Olly:

Address	Hex dump	Disassembly	Comment
004773FE	5B	POP EBX	
004773FF	5F	POP EDI	YodaProt.0047975E
00477400	79 DC	JNS SHORT YodaProt.004773DE	
00477402	86E0	XCHG AL, AH	
00477404	9A B45A3290 8F27	CALL FAR 278F:90325AB4	
0047740B	A4	MOVS BYTE PTR ES:[EDI], BYTE PTR DS:[ESI]	
0047740C	E5 FC	IN EAX, OFC	
0047740E	12D3	ADC DL, BL	
00477410	3008	XOR BYTE PTR DS:[EAX], CL	
00477412	8901	MOV DWORD PTR DS:[ECX], EAX	
00477414	9A 9E40AE40 9BB4	CALL FAR B49B:40AE409E	
0047741B	9D	POPPD	
0047741C	7C 6E	JL SHORT YodaProt.0047748C	
0047741E	A2 A2838834	MOV BYTE PTR DS:[348883A2], AL	
00477423	94	XCHG EAX, ESP	
00477424	396432 EF	CMP DWORD PTR DS:[EDX+ESI-11], ESP	
00477428	1A20	SBB AH, BYTE PTR DS:[EAX]	
0047742A	B9 977B519C	MOV ECX, 9C517B97	
0047742F	B1 56	MOV CL, 56	
00477431	65:63F8	ARPL AX, DI	
00477434	EA A3976B32 FDB3	JMP FAR B3FD:326B97A3	
0047743B	BE AD268E90	MOV ESI, 908E26AD	
00477440	14 EF	ADC AL, 0EF	

Debugged program was unable to process exception

_Hic Status Bar if the information on it as we went to dust and then, as any one of the other is considered to terminate the line, so we must find another way to OEP only. Uncheck all **Exceptions** that we have set in, then press **F9** 1 times we here at Break:

Address	Hex dump	Disassembly	Comment
004772BB	C3	RETN	
004772BC	E8 03000000	CALL YodaProt.004772C4	
004772C1	EB 01	JMP SHORT YodaProt.004772C4	
004772C3	E9 33DBB984	JMP 85014DFB	
004772C8	4E	DEC ESI	
004772C9	42	INC EDX	
004772CA	0081 E98D2A42	ADD BYTE PTR DS:[ECX+422A8DE9], AL	
004772D0	008B D581C28D	ADD BYTE PTR DS:[EBX+8DC281D5], CL	
004772D6	2A42 00	SUB AL, BYTE PTR DS:[EDX]	
004772D9	8D3A	LEA EDI, DWORD PTR DS:[EDX]	
004772DB	8BF7	MOV ESI, EDI	
004772DD	33C0	XOR EAX, EAX	
004772DF	E8 03000000	CALL YodaProt.004772E7	
004772E4	EB 01	JMP SHORT YodaProt.004772E7	
004772E6	E9 E8170000	JMP YodaProt.00478AD3	
004772EB	0090 9090E9C1	ADD BYTE PTR DS:[EAX+C1E99090], DH	
004772F1	1D 000033C0	SBB EAX, C0330000	
004772F6	64:FF30	PUSH DWORD PTR FS:[EAX]	
004772F9	64:8920	MOV DWORD PTR FS:[EAX], ESP	
004772FC	43	INC EBX	
004772FD	CC	INT3	
004772FE	C3	RETN	

_Chung We bypass it by pressing **Shift + F7** and press **F9** to run. Once again we are back at Break position on:

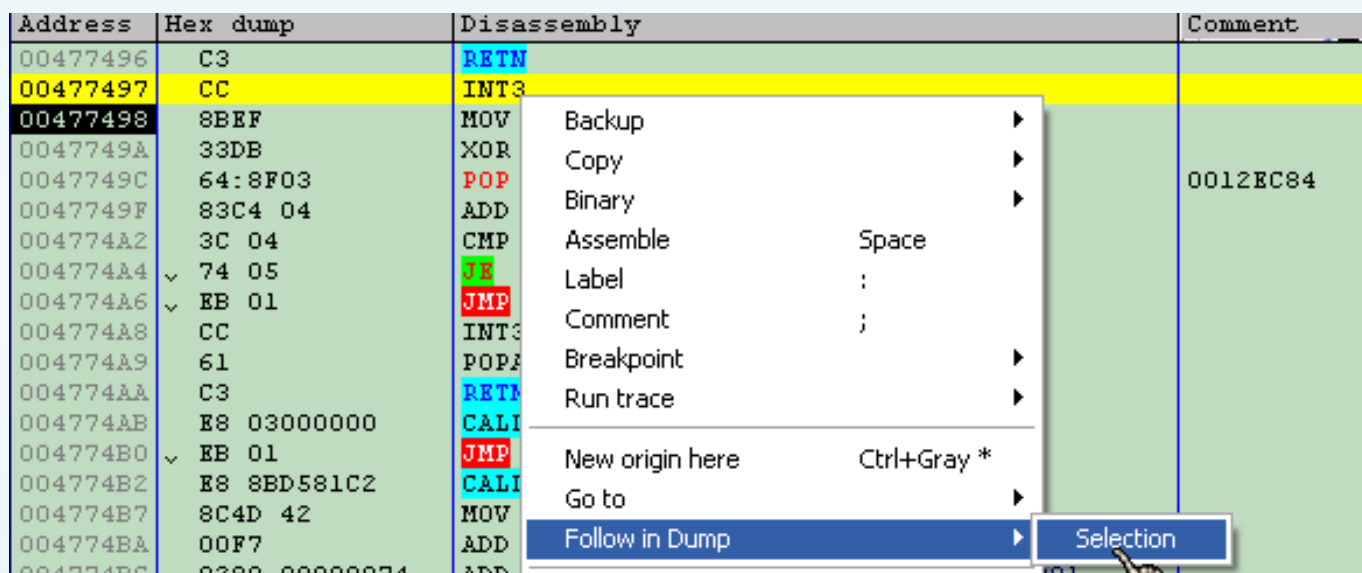
Address	Hex dump	Disassembly	Comment
004772BB	C3	RETN	
004772BC	E8 03000000	CALL YodaProt.004772C4	
004772C1	EB 01	JMP SHORT YodaProt.004772C4	
004772C3	E9 33DBB984	JMP 85014DFB	
004772C8	4E	DEC ESI	
004772C9	42	INC EDX	ntdll.77F833B4
004772CA	0081 E98D2A42	ADD BYTE PTR DS:[ECX+422A8DE9], AL	
004772D0	008B D581C28D	ADD BYTE PTR DS:[EBX+8DC281D5], CL	
004772D6	2A42 00	SUB AL, BYTE PTR DS:[EDX]	
004772D9	8D3A	LEA EDI, DWORD PTR DS:[EDX]	
004772DB	8BF7	MOV ESI, EDI	
004772DD	33C0	XOR EAX, EAX	
004772DF	E8 03000000	CALL YodaProt.004772E7	
004772E4	EB 01	JMP SHORT YodaProt.004772E7	
004772E6	E9 E8170000	JMP YodaProt.00478AD3	
004772EB	0090 9090E9C1	ADD BYTE PTR DS:[EAX+C1E99090], DL	
004772F1	1D 000033C0	SBB EAX, C0330000	
004772F6	64:FF30	PUSH DWORD PTR FS:[EAX]	
004772F9	64:8920	MOV DWORD PTR FS:[EAX], ESP	
004772FC	43	INC EBX	
004772FD	CC	INT3	
004772FE	C3	RETN	
004772FF	90	NOP	

_Tiep To add a few more times until you come in Olly (my 4 times press **Shift + F7, F9**):

Address	Hex dump	Disassembly	Comment
0047747B	8BFD	MOV EDI, EBP	
0047747D	8BD5	MOV EDX, EBP	
0047747F	81C2 2A4A4200	ADD EDX, YodaProt.00424A2A	
00477485	8D02	LEA EAX, DWORD PTR DS:[EDX]	
00477487	33DB	XOR EBX, EBX	
00477489	50	PUSH EAX	YodaProt.00470004
0047748A	64:FF33	PUSH DWORD PTR FS:[EBX]	
0047748D	64:8923	MOV DWORD PTR FS:[EBX], ESP	
00477490	66:B8 0400	MOV AX, 4	
00477494	EB 01	JMP SHORT YodaProt.00477497	
00477496	C3	RETN	
00477497	CC	INT3	
00477498	8BEF	MOV EBP, EDI	
0047749A	33DB	XOR EBX, EBX	
0047749C	64:8F03	POP DWORD PTR FS:[EBX]	0012EC84
0047749F	83C4 04	ADD ESP, 4	
004774A2	3C 04	CMP AL, 4	
004774A4	74 05	JE SHORT YodaProt.004774AB	
004774A6	EB 01	JMP SHORT YodaProt.004774A9	
004774A8	CC	INT3	
004774A9	61	POPAD	
004774AA	C3	RETN	

_Tai Here for our comments on the position that we have Break **INT3** command, this command is responsible for the exception in that we are using that tai. De hairbreadth below you will see a command comparison with AL 4 and ordered a conditional jump **JE**. Orders jumped conditions for implementation of this exception, it will perform in the AL 4 and jump when the command is implemented, the means is the process by Debugger we were Detect. However, if the way that we make the jump this order can not be made, that is our command at **JMP JE** orders below the means of the Debugger we have escaped the detection.

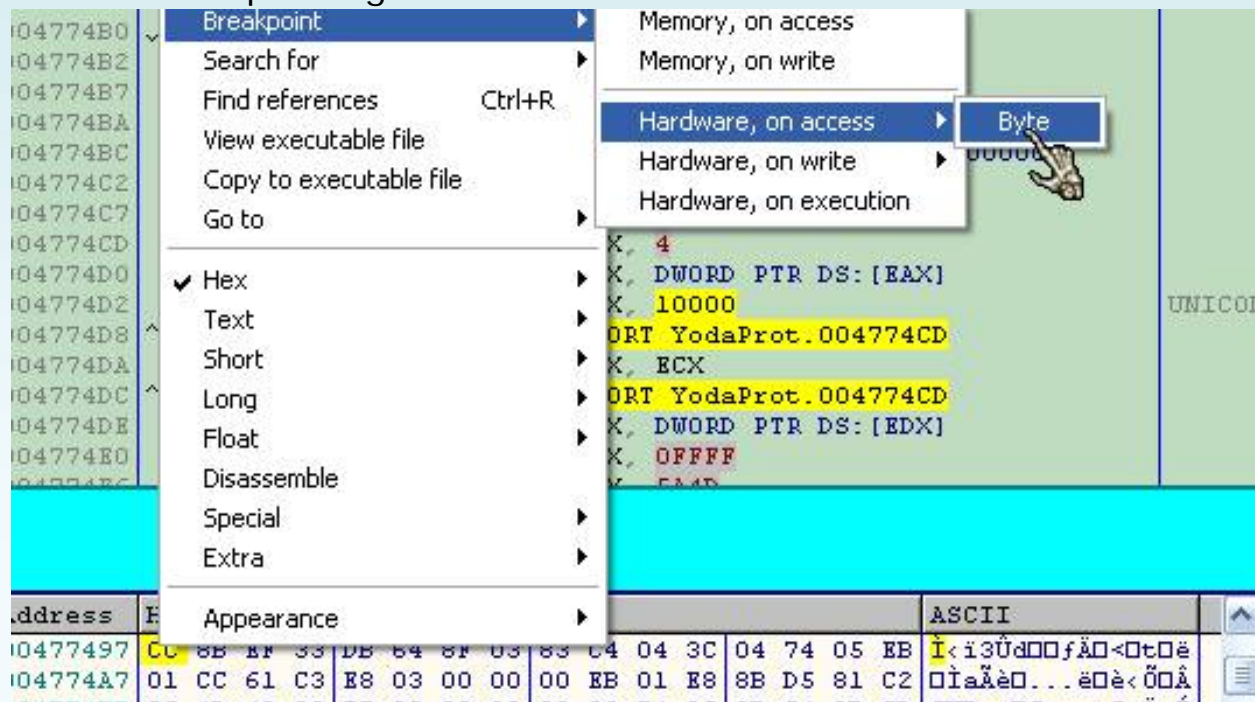
_Ok We will bypass and avoid the exception to this is right at **hien.Chuot 00477497 CC INT3**, select **Follow in dump -> Selection**:



_Ta Information is as follows in the dump:

Address	Hex dump	ASCII
00477497	CC 8B EF 33 DB 64 8F 03 83 C4 04 3C 04 74 05 EB	I<i3Üd□□fÄ□<□t□ë
004774A7	01 CC 61 C3 E8 03 00 00 00 EB 01 E8 8B D5 81 C2	□ïaÄë□...ë□è<Ö□Ä
004774B7	8C 4D 42 00 F7 02 80 00 00 00 74 35 8B C4 8B CD	□MB..÷□ë...t5<Ä<Í
004774C7	81 C1 F3 2B 42 00 83 C0 04 8B 10 81 FA 00 00 01	□Äó+B..fÄ□<□□ú..□
004774D7	00 72 F3 3B D1 73 EF 8B 1A 81 E3 FF FF 00 00 81	.ró;Ñsi<□□äÿÿ..□
004774E7	FB 4D 5A 00 00 75 DF 8B CD 81 C1 A8 4A 42 00 89	ûM2..uß<Í□Ä"JB.%
004774F7	11 8B D5 81 C2 A8 4A 42 00 8B 02 03 40 3C 05 80	□<Ö□Ä"JB.<□□@<□ë

_Ta'll See **0xCC** corresponding to **INT3**. Mouse set to 1 and BP as follows:



_Bay Hour press **Ctrl + F2** to Restart the program. Then press **F9** to implement and bypass the **Exceptions** until we Break BP in which we have set:

Address	Hex dump	Disassembly	Comment
00477304	EB 01	JMP SHORT YodaProt.00477307	
00477306	E9 FEC8EB01	JMP 02333C09	
0047730B	E9 F9C0C084	JMP 85083409	
00477310	FEC8	DEC AL	
00477312	EB 01	JMP SHORT YodaProt.00477315	
00477314	E8 EB01C234	CALL 35097504	
00477319	EC	IN AL, DX	
0047731A	EB 01	JMP SHORT YodaProt.0047731D	
0047731C	E8 F8EB01C2	CALL C2495F19	
00477321	EB 01	JMP SHORT YodaProt.00477324	
00477323	E8 F8EB01E8	CALL E8495F20	
00477328	EB 01	JMP SHORT YodaProt.0047732B	
0047732A	E8 02C1902A	CALL 2AD83431	
0047732F	C102 C1	ROL DWORD PTR DS:[EDX], 0C1	
00477332	EB 01	JMP SHORT YodaProt.00477335	
00477334	E8 3495EB01	CALL 0233086D	
00477339	E8 EB01E904	CALL 05307529	
0047733E	D6	SALC	
0047733F	EB 01	JMP SHORT YodaProt.00477342	
00477341	E8 EB01C202	CALL 03097531	

Hardware breakpoint 1 at YodaProt.00477304 - EIP points to next instruction

_Trong Dump window jump to the position that we have set BP, we will see that **0xCC** is not written to that location:

Address	Hex dump	ASCII
00477497	10 DA FC E0 BB E3 7A 51 B8 6C 01 63 0E E4 DD D7	00000000000000000000000000000000
004774A7	1F EC 92 D7 CA DB CB AE CE D6 9E A8 B6 F4 11 45	00000000000000000000000000000000
004774B7	04 FF 6E 2C F8 4C C1 A8 C8 A8 0C 7B C0 E1 BD 74	00000000000000000000000000000000
004774C7	1F B4 BF 56 E7 23 3B A2 83 DC 43 BE 61 21 C1 31	00000000000000000000000000000000
004774D7	C2 C5 BD 55 9E D4 7C D9 EE B8 BA 20 7F 2F CF 37	00000000000000000000000000000000
004774E7	3E F1 ED 2E 4E F6 79 53 57 B6 9A A4 62 42 C9 B3	00000000000000000000000000000000
004774F7	1E D2 53 B2 E5 AF 6E 59 C8 D2 68 D5 4C 6B D5 B0	00000000000000000000000000000000
00477507	C8 A8 46 60 C8 D2 68 D5 1E A2 2B B6 26 46 BB 22	00000000000000000000000000000000

_Tiep To press **F9** to Run again, we break here:

Address	Hex dump	Disassembly	Comment
00477361	01E8	ADD EAX, EBP	
00477363	90	NOP	
00477364	AA	STOS BYTE PTR ES:[EDI]	
00477365	E2 9C	LOOPE SHORT YodaProt.00477303	
00477367	E8 03000000	CALL YodaProt.0047736F	
0047736C	EB 01	JMP SHORT YodaProt.0047736F	
0047736E	E9 33DBB92E	JMP 2F014EA6	
00477373	2942 00	SUB DWORD PTR DS:[EDX], EAX	
00477376	81E9 86274200	SUB ECX, YodaProt.00422786	
0047737C	8BD5	MOV EDX, EBP	
0047737E	81C2 86274200	ADD EDX, YodaProt.00422786	
00477384	8D3A	LEA EDI, DWORD PTR DS:[EDX]	
00477386	8BF7	MOV ESI, EDI	
00477388	33C0	XOR EAX, EAX	YodaProt.00477498

_Aha Now is **0xCC** was recorded in the position we set BP:

Address	Hex dump	ASCII
00477497	CC DA FC E0 BB E3 7A 51 B8 6C 01 63 0E E4 DD D7	ÛÜà>ãzQ,10c0ãÿ×
004774A7	1F EC 92 D7 CA DB CB AE CE D6 9E A8 B6 F4 11 45	0i'×ÊÜÊ@iÖz`qð0E
004774B7	04 FF 6E 2C F8 4C C1 A8 C8 A8 0C 7B C0 E1 BD 74	0ÿn,øLÄ"È".{Äá%t
004774C7	1F B4 BF 56 E7 23 3B A2 83 DC 43 BE 61 21 C1 31	0'¿Vç#;çfÜC%a!Ä1
004774D7	C2 C5 BD 55 9E D4 7C D9 EE B8 BA 20 7F 2F CF 37	ÄÄ%UzÖ Üi,° 0/I7
004774E7	3E F1 ED 2E 4E F6 75 D3 57 B6 9A A4 62 42 C9 B3	>ñi.NöuÓWq[sxbBÉ³
004774F7	1E D2 53 B2 E5 AF 6E 4D C8 D2 68 D5 4C 6B D5 B0	00S²ä~nMÈ0hÖLkÖ°
00477507	C8 A8 4C 60 C8 D3 68 FB 1E A3 3B D6 3C 4C BB 33	È"È"ÈÖh,10c0ãÿ×

_Okie We change **0xCC** to **0x90** (NOP), in press **Ctri 0xCC + B** and modified to **90**:

Address	Hex dump	ASCII
00477497	90 DA FC E0 BB E3 7A 51 B8 6C 01 63 0E E4 DD D7	ÛÜà>ãzQ,10c0ãÿ×
004774A7	1F EC 92 D7 CA DB CB AE CE D6 9E A8 B6 F4 11 45	0i'×ÊÜÊ@iÖz`qð0E
004774B7	04 FF 6E 2C F8 4C C1 A8 C8 A8 0C 7B C0 E1 BD 74	0ÿn,øLÄ"È".{Äá%t
004774C7	1F B4 BF 56 E7 23 3B A2 83 DC 43 BE 61 21 C1 31	0'¿Vç#;çfÜC%a!Ä1
004774D7	C2 C5 BD 55 9E D4 7C D9 EE B8 BA 20 7F 2F CF 37	ÄÄ%UzÖ Üi,° 0/I7
004774E7	3E F1 ED 2E 4E F6 75 D3 57 B6 9A A4 62 42 C9 B3	>ñi.NöuÓWq[sxbBÉ³
004774F7	1E D2 53 B2 E5 AF 6E 4D C8 D2 68 D5 4C 6B D5 B0	00S²ä~nMÈ0hÖLkÖ°
00477507	C8 A8 4C 60 C8 D3 68 FB 1E A3 3B D6 3C 4C BB 33	È"È"ÈÖh,10c0ãÿ×

_Vay Is completed, the next we do? As has been said to you on when we used to detect PEiD information, we know the version of Yoda's used to protect the target, but if you use the Plugin PEid for the OEP immediately be PEiD Terminate contact. Mechanism of it as tut by + **NCR** as follows:

*This made me remember the old ACProtect in which it was detected when debugger by means of a disguised and quite dirty trick. Becomes what one is to verify that PID (Process ID) that sent the program, in this case crackme, was one of the **explorer.exe** that is the same OS and not another application. If the PIDs does not agree then chau, it has to debugger or loader to playing with the application, but we followed as if nothing;). In order to make that trick there are several ways, but I believe that it is commonest to use the function **CreateToolhelp32Snapshot Process32Firts Or Process32Next** along with. These three functions which do are to work together to cross all the processes loaded in memory and to detect different things, for example caption of a window, the class of the window, or in this case the PID.*

_Tom By my understanding is that is Target PID will check to see it from where the load, if it is load it from Explorer.exe run a normal. But if through Loader Debugger or it will Terminate line.

_Do The next job we as follows: Delete BP we've set, a set in the API function:

CreateToolhelp32Snapshot



Command: he CreateToolhelp32Snapshot HE address -- HW break on execution

Hardware breakpoint 1 at YodaProt.00477365 - EIP points to next instruction

F9 Run _Nhan to target, we will Break in API function:

Address	Hex dump	Disassembly	Comment
77EBB1E7	55	PUSH EBP	
77EBB1E8	8BEC	MOV EBP, ESP	
77EBB1EA	83EC 0C	SUB ESP, 0C	
77EBB1ED	56	PUSH ESI	kernel32.77E60000
77EBB1EE	8B75 0C	MOV ESI, DWORD PTR SS:[EBP+C]	
77EBB1F1	85F6	TEST ESI, ESI	kernel32.77E60000
77EBB1F3	75 07	JNZ SHORT kernel32.77EBB1FC	
77EBB1F5	E8 5C54FCFF	CALL kernel32.GetCurrentProcessId	
77EBB1FA	8BF0	MOV ESI, EAX	

Address	Hex dump	ASCII	Address	Value	Comment
00477497	90 8B EF 33 DB 64 8F 03 83 C4 04 3C 04 74 05 EB	0<13U&00;AD<0t0e	0012EBEC	00477902	CALL to CreateToolhelp32Snapshot from YodaPro
004774A7	01 CC 61 C3 E8 03 00 00 00 EB 01 E8 8B D5 81 C2	01a&0...e0&00A	0012EBF0	00000002	Flags = TH32CS_SNAPPROCESS
004774B7	8C 4D 42 00 F7 02 80 00 00 00 74 35 8B C4 8B CD	8MB.+0E...t5<A<I	0012EBF4	00000000	ProcessID = 0
004774C7	81 C1 F3 2B 42 00 83 C0 04 8B 10 81 FA 00 00 01	0A6+B.f&0:00&...0	0012EBF8	004778C1	RETURN to YodaProt.004778C1 from YodaProt.0047
004774D7	00 72 F3 3B D1 73 EF 8B 1A 81 E3 FF FF 00 00 81	.r6;Nsi<00&9y..0	0012EBFC	0047781B	RETURN to YodaProt.0047781B from YodaProt.0047
004774E7	FB 4D 5A 00 00 75 DF 8B CD 81 C1 A8 4A 42 00 89	0MZ..u&0A<JB.z	0012EC00	00477813	RETURN to YodaProt.00477813 from YodaProt.0047
004774F7	11 8B D5 81 C2 A8 4A 42 00 8B 02 03 40 3C 05 80	0<00A<JB.<00&0&0	0012EC04	004777F2	RETURN to YodaProt.004777F2 from YodaProt.0047
00477507	00 00 00 8B 08 03 0A 83 C1 10 8B 01 03 02 8B 18	...<00.f&0:000<0	0012EC08	004777D1	RETURN to YodaProt.004777D1 from YodaProt.0047
00477517	8B D5 81 C2 CA 28 42 00 89 1A 83 C0 04 8B 18 8B	00A&B(B.t0;f&0:0<	0012EC0C	004777B0	RETURN to YodaProt.004777B0 from YodaProt.0047
00477527	D5 81 C2 CE 28 42 00 89 1A 8B D5 81 C2 86 27 42	00A&f(B.t0:00A<+B	0012EC10	0047778F	RETURN to YodaProt.0047778F from YodaProt.0047
00477537	00 8D 02 50 8B D5 81 C2 CA 28 42 00 FF 12 E8 03	.00P<00A&f(B.y0&0	0012EC14	0047776E	RETURN to YodaProt.0047776E from YodaProt.0047

_Nhan **Ctrl + F9** (execute till Return) and press **F7** we stop here in Olly:

Address	Hex dump	Disassembly	Comment
00477902	8BF0	MOV ESI, EAX	
00477904	8BC5	MOV EAX, EBP	
00477906	05 B44E4200	ADD EAX, YodaProt.00424EB4	
0047790B	50	PUSH EAX	
0047790C	56	PUSH ESI	kernel32.77E60000
0047790D	8BD5	MOV EDX, EBP	
0047790F	81C2 0A294200	ADD EDX, YodaProt.0042290A	
00477915	FF12	CALL DWORD PTR DS:[EDX]	
00477917	85C0	TEST EAX, EAX	
00477919	0F84 AF000000	JE YodaProt.004779CE	
0047791F	8BD5	MOV EDX, EBP	
00477921	81C2 B44E4200	ADD EDX, YodaProt.00424EB4	
00477927	8D0A	LEA ECX, DWORD PTR DS:[EDX]	
00477929	51	PUSH ECX	
0047792A	56	PUSH ESI	kernel32.77E60000
0047792B	8BD5	MOV EDX, EBP	
0047792D	81C2 0E294200	ADD EDX, YodaProt.0042290E	

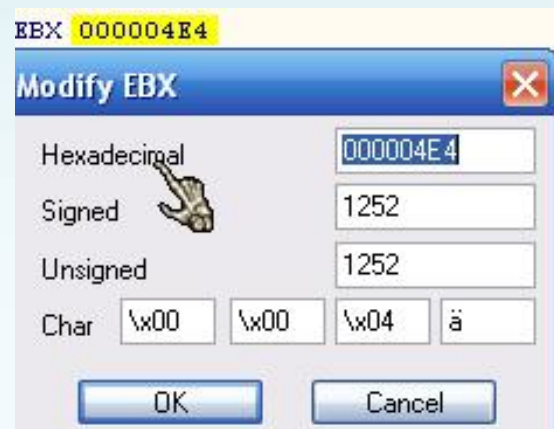
_Chung We are in code enforcement of trinh.Tai here we press **F8** to trace until after having jumped order:

Address	Hex dump	Disassembly	Comment
0047799B	81C2 B44E4200	ADD EDX, YodaProt.00424EB4	
004779A1	8B42 18	MOV EAX, DWORD PTR DS:[EDX+18]	
004779A4	8BD5	MOV EDX, EBP	
004779A6	81C2 AC4E4200	ADD EDX, YodaProt.00424EAC	
004779AC	8902	MOV DWORD PTR DS:[EDX], EAX	
004779AE	5A	POP EDX	YodaProt.004778C1
004779AF	5E	POP ESI	YodaProt.004778C1
004779B0	8BD5	MOV EDX, EBP	
004779B2	81C2 B44E4200	ADD EDX, YodaProt.00424EB4	
004779B8	8D0A	LEA ECX, DWORD PTR DS:[EDX]	
004779BA	51	PUSH ECX	YodaProt.004797BB
004779BB	56	PUSH ESI	
004779BC	8BD5	MOV EDX, EBP	
004779BE	81C2 0E294200	ADD EDX, YodaProt.0042290E	
004779C4	FF12	CALL DWORD PTR DS:[EDX]	
004779C6	85C0	TEST EAX, EAX	
004779C8	0F85 76FFFFFF	JNZ YodaProt.00477944	
004779CE	8BD5	MOV EDX, EBP	
004779D0	81C2 AC4E4200	ADD EDX, YodaProt.00424EAC	

_Day A loop for all Detect Process are implemented. To bypass this loop is set by BP in command under the command then jumped on the press **F9** to Run will Break here:

Address	Hex dump	Disassembly	Comment
004779CE	8BD5	MOV EDX, EBP	
004779D0	81C2 AC4E4200	ADD EDX, YodaProt.00424EAC	
004779D6	8B02	MOV EAX, DWORD PTR DS:[EDX]	
004779D8	3BC3	CMP EAX, EBX	
004779DA	74 30	JE SHORT YodaProt.00477A0C	
004779DC	8BD5	MOV EDX, EBP	
004779DE	81C2 8C4D4200	ADD EDX, YodaProt.00424D8C	
004779E4	F702 80000000	TEST DWORD PTR DS:[EDX], 80	
004779EA	75 20	JNZ SHORT YodaProt.00477A0C	
004779EC	50	PUSH EAX	
004779ED	6A 01	PUSH 1	
004779EF	68 FFOF1F00	PUSH 1FOFFF	
004779F4	8BD5	MOV EDX, EBP	

_Nhin Down to see a comparison between the command EAX and EBX, this is a very important order, it will compare PID.Nhan F8 to trace down to see we will have to record in 2 EAX and EBX:



_Dung **LordPE** to see a list of our process and observe the PIDs:

Path	PID	ImageBase	ImageSize
e:\windows\system32\svchost.exe	00000354	01000000	00006000
e:\windows\system32\spoolsv.exe	000003AC	01000000	0000F000
e:\windows\system32\svchost.exe	0000042C	01000000	00006000
e:\program files\alcohol soft\alcohol 120\star...	00000478	00400000	0003A000
e:\windows\system32\svchost.exe	00000484	01000000	00006000
e:\windows\explorer.exe	000004E4	01000000	000F7000
d:\setup\unikey\unikeynt.exe	000003C8	00400000	00034000
e:\windows\integrator.exe	00000400	00400000	00029000
e:\program files\getdiz\getdiz.exe	000003E8	00400000	0009D000
e:\docume~1\trantr~1\desktop\foxitr~1.exe	000006B0	00400000	0029E000
e:\program files\microsoft office\office11\win...	000007E4	30000000	008AA000
e:\program files\techsmith\snagit 7\snagit32...	000006A8	00400000	00390000
e:\program files\techsmith\snagit 7\tschelp.exe	00000504	00400000	0000A000
e:\program files\lmt300\qviewmtd.exe	00000518	00400000	00102000
e:\program files\lmt300\qshlfmt.exe	0000039C	00400000	00052000
e:\documents and settings\tran trung kien\m...	000001A4	00400000	00154000
e:\program files\mozilla firefox\firefox.exe	000006DC	00400000	006EE000
e:\documents and settings\tran trung kien\de...	00000530	00400000	0007C000
e:\documents and settings\tran trung kien\m...	000000C0	00400000	00036000

_Nhu So we see that it will compare with the PID's Olly Explorer.exe. Tuc target is to check that it is load from **Explorer.exe** Correct? Therefore we will modify the content by recording it to EAX same EBX:

Registers (FPU)	
EAX	000004E4
ECX	004797BF YodaProt.004797BF
EDX	00479786 YodaProt.00479786
EBX	000004E4
ESP	0012EBF8
EBP	000548DA
ESI	00000000
EDI	00479786 ASCII "XPLORER.EXE"

_Nhan **F9** to run, we Break the API in which we set BP, the implementation process of the bypass and on to the final press **F9** we will stop here in Olly:

Address	Hex dump	Disassembly	Comment
004792F1	0000	ADD BYTE PTR DS:[EAX], AL	
004792F3	0000	ADD BYTE PTR DS:[EAX], AL	
004792F5	0000	ADD BYTE PTR DS:[EAX], AL	
004792F7	0000	ADD BYTE PTR DS:[EAX], AL	
004792F9	0000	ADD BYTE PTR DS:[EAX], AL	
004792FB	0000	ADD BYTE PTR DS:[EAX], AL	
004792FD	0000	ADD BYTE PTR DS:[EAX], AL	
004792FF	0000	ADD BYTE PTR DS:[EAX], AL	
00479301	0000	ADD BYTE PTR DS:[EAX], AL	
00479303	0000	ADD BYTE PTR DS:[EAX], AL	
00479305	0000	ADD BYTE PTR DS:[EAX], AL	
00479307	0000	ADD BYTE PTR DS:[EAX], AL	
00479309	0000	ADD BYTE PTR DS:[EAX], AL	
0047930B	0000	ADD BYTE PTR DS:[EAX], AL	
0047930D	0000	ADD BYTE PTR DS:[EAX], AL	
0047930F	0000	ADD BYTE PTR DS:[EAX], AL	
00479311	0000	ADD BYTE PTR DS:[EAX], AL	
00479313	0000	ADD BYTE PTR DS:[EAX], AL	
00479315	0000	ADD BYTE PTR DS:[EAX], AL	
00479317	0000	ADD BYTE PTR DS:[EAX], AL	
00479319	0000	ADD BYTE PTR DS:[EAX], AL	
0047931B	0000	ADD BYTE PTR DS:[EAX], AL	
0047931D	0000	ADD BYTE PTR DS:[EAX], AL	

Access violation when writing to [00000000] - use Shift+F7/F8/F9 to pass exception to program

_Tai Ago we press **Alt + M** to open the Memory window, then set at BP's target code section:

003F00	000020	YodaProt		PE header	Map	R E	R E	
004000	000010	YodaProt		code	Ima	RW	RWE	
004010	000010	YodaProt		code, data	Ima	RW	RWE	
004020	000010	YodaProt		code				
004030	000010	YodaProt		code, resou				
004040	000730	YodaProt	.rsrc	code, resou				
004770	000050	YodaProt	.yP	code, impo				
004800	001030							
005900	000010							
005A00	000AA0							
008A00	000010							
5F0000	000010							
773400	000010	comctl32		PE header				
773410	000660	comctl32	.text	code, impo				
773A70	000010	comctl32	.data	code, data				

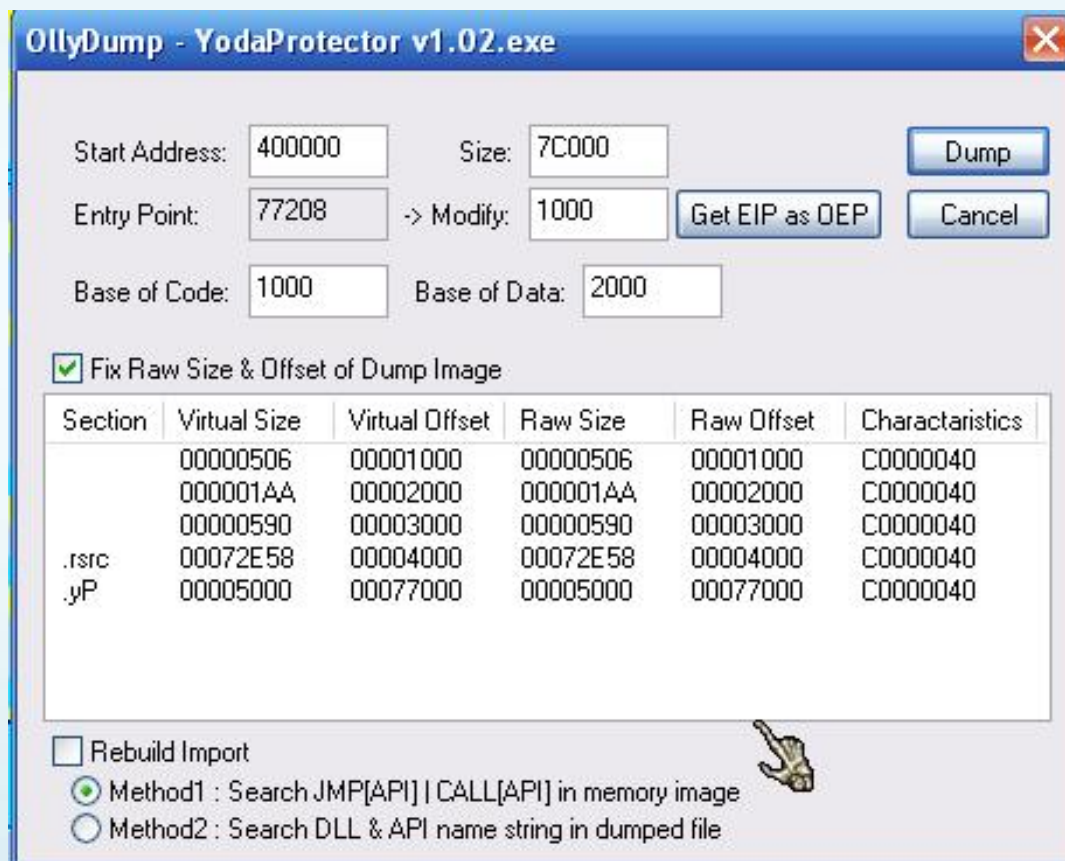
- Actualize
- View in Disassembler Enter
- Dump in CPU
- Dump
- Search Ctrl+B
- Set break-on-access F2
- Set memory breakpoint on access
- Set memory breakpoint on write

_Tiep By pressing **Shift + F7, F9**, we will stop at the OEP of the program:

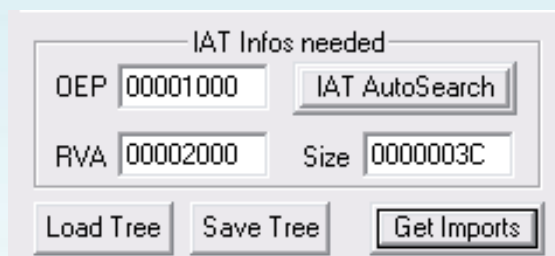
Address	Hex dump	Disassembly	Comment
00401000	6A 00	PUSH 0	
00401002	E8 BD040000	CALL YodaProt.004014C4	
00401007	A3 78324000	MOV DWORD PTR DS:[403278], EAX	
0040100C	E8 EF040000	CALL YodaProt.00401500	
00401011	6A 00	PUSH 0	
00401013	68 2E104000	PUSH YodaProt.0040102E	
00401018	6A 00	PUSH 0	
0040101A	6A 65	PUSH 65	
0040101C	FF35 78324000	PUSH DWORD PTR DS:[403278]	
00401022	E8 A9040000	CALL YodaProt.004014D0	
00401027	6A 00	PUSH 0	
00401029	E8 90040000	CALL YodaProt.004014BE	
0040102E	55	PUSH EBP	YodaProt.00477
0040102F	8BEC	MOV EBP, ESP	
00401031	8B45 0C	MOV EAX, DWORD PTR SS:[EBP+C]	
00401034	3D 10010000	CMP EAX, 110	
00401039	0F85 96000000	JNZ YodaProt.004010D5	
0040103F	68 20030000	PUSH 320	

Memory breakpoint when executing [00401000]

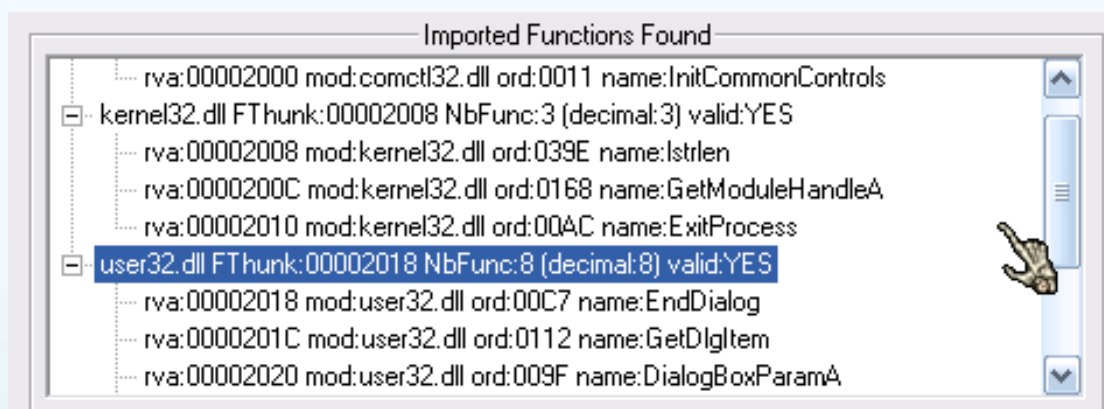
_Wow! OEP là dump only. Conduct dump file by using Olly Plugin and save less than 1 then another, for example: **dumped.exe**.



_Mo **ImportRec** up to conduct Fix IAT. Complete information on IAT as follows:



Get _Nhan Imports, we are all Invalids. Do not worry, do the tuts first mouse to select **Trace Level1 (DisAsm)**:



_Cuoi Fix the **dump** file and test that we take the time to unpack, phew then run it decently J:



_Bai Written here is to end. See you again in other tuts! J

—
Best Regards

— [Kienmanowar] _

---+---==[Greatz thanks to]=---+---

My family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHOCrker, the_Lighthouse, Merc, Hoadongnoi, Nini ... all REA's members, TQN, HacNho, RongChauA, Deux, tlandn, light.phoenix, dqtlN, ARTEAM all my friend, and you.

Thanks to ---+---==[]=---+---

iamidiot, WhyNotBar, trickyboy, dzungltvn, takada, hurt_heart, haule_nth, v. hytkl. v.. You have contributed greatly to the REA. Hope you will continue to promote J

>>> If you have any suggestions, comments or corrections email me: **kienbigmummy [at] gmail.
com**

Yoda's Protector v1.03.2 beta3 - manually unpacking tutorial (_kienmanowar_)

I. Introduction:

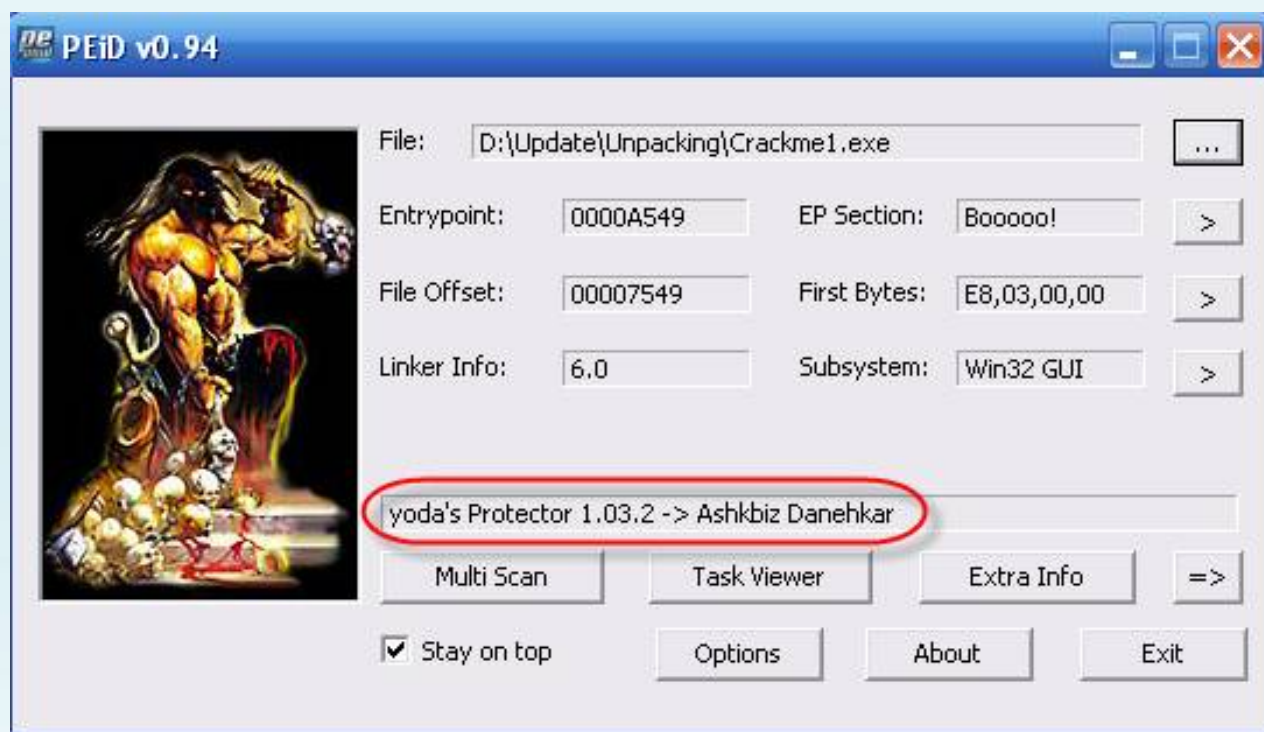
This is the **hangar** by tut reversing.be, I consult and write for the purpose of the collection and write tru.Bai this focus on the latest version of **Yoda's Protector**, the later versions of Yoda has a lot of the anti-debug, which is next to the API as **BlockInput** (The BlockInput function blocks keyboard and mouse input events from reaching applications) etc.. etc., to our disappointment in the MUPing.

II. Tools & Target:

- _ **ODbyDYK v1.10**
- _ **PeiD v0.94**
- _ **LordPE**
- _ **ImportRec v1.6**
- _ **dup2 v2.13 BETA 25**
- _ **YP PEiD signatures.txt**
- _ **Crackme1.exe**

III. MUPing:

_Dung **PeiD** to get information on target:



_Run Target, then open up Olly. In Olly, select functions **File -> Attach** to Attach to the target Olly:

Select process to attach

Process	Name	Window	Path
000002BC	Crackme1	Knight's Crackme#1	D:\Update\Unpacking\Crackme1.exe
00000200	csrss		\\?\E:\WINDOWS\system32\csrss.exe
00000484	Explorer	SysFader	E:\WINDOWS\Explorer.EXE

Address	Hex dump	Disassembly	Comment
77F7F571	C3	RETN	
77F7F572	8BFF	MOV EDI,EDI	
77F7F574	CC	INT3	
77F7F575	C3	RETN	
77F7F576	8BFF	MOV EDI,EDI	
77F7F578	8B4424 04	MOV EAX,DWORD PTR SS:[ESP+4]	
77F7F57C	CC	INT3	
77F7F57D	C2 0400	RETN 4	
77F7F580	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]	
77F7F586	C3	RETN	
77F7F587	57	PUSH EDI	

Attach _Sau when we stopped at the system as BP's on, here we press **Alt + E** to open the window **Executables**. Select a target and Double Click, we will in turn return to the screen by Olly CPU:

Base	Size	Entry	Name (system)	File version	Path
00400000	00002000	0040A549	Crackme1		D:\Update\Unpacking\Crackme1.exe
0FFD0000	00022000	0FFDF7E0	rsaenh (system)	5.1.2518.0 (main.010714-2114)	E:\WINDOWS\System32\rsaenh.dll
5AD70000	00034000	5AD71583	UxTheme (system)	6.00.2600.0000 (xpclient.010817-114)	E:\WINDOWS\system32\UxTheme.dll
77C10000	00053000	77C1E94F	msvcrt (system)	7.0.2600.0 (xpclient.010817-114)	E:\WINDOWS\system32\msvcrt.dll
77C70000	00040000		GDI32 (system)	5.1.2600.0 (xpclient.010817-114)	E:\WINDOWS\system32\GDI32.dll
77D40000	0008D000	77D4514B	User32 (system)	5.1.2600.0 (xpclient.010817-114)	E:\WINDOWS\system32\User32.dll
77DD0000	0008B000	77DD1CFB	ADVAPI32 (system)	5.1.2600.0 (XPClient.010817-114)	E:\WINDOWS\system32\ADVAPI32.dll
77E60000	000E5000	77E7A241	kernel32 (system)	5.1.2600.0 (xpclient.010817-114)	E:\WINDOWS\system32\kernel32.dll
77F50000	000A9000		ntdll (system)	5.1.2600.0 (xpclient.010817-114)	E:\WINDOWS\System32\ntdll.dll
78000000	0006E000	780072BA	RPCRT4 (system)	5.1.2600.109 (xpclient_qfe.021108)	E:\WINDOWS\system32\RPCRT4.dll

Address	Hex dump	Disassembly	Comment
00401000	33C0	XOR EAX,EAX	
00401002	394424 08	CMP DWORD PTR SS:[ESP+8],EAX	
00401006	7E 14	JLE SHORT Crackme1.0040101C	
00401008	8B4C24 04	MOV ECX,DWORD PTR SS:[ESP+4]	
0040100C	8B0C81	MOV ECX,DWORD PTR DS:[ECX+EAX*4]	
0040100F	3B4C24 0C	CMP ECX,DWORD PTR SS:[ESP+C]	
00401013	74 0A	JE SHORT Crackme1.0040101F	
00401015	40	INC EAX	
00401016	3B4424 08	CMP EAX,DWORD PTR SS:[ESP+8]	
0040101A	7C EC	JL SHORT Crackme1.00401008	
0040101C	32C0	XOR AL,AL	
0040101E	C3	RETN	
0040101F	B0 01	MOV AL,1	
00401021	C3	RETN	

_Oki We are in the target code. At press **F9** to run a target normally. The next step we will find it to OEP. Press **Alt + M** to open the window **memory**, we set a BP in the following code section of the target:

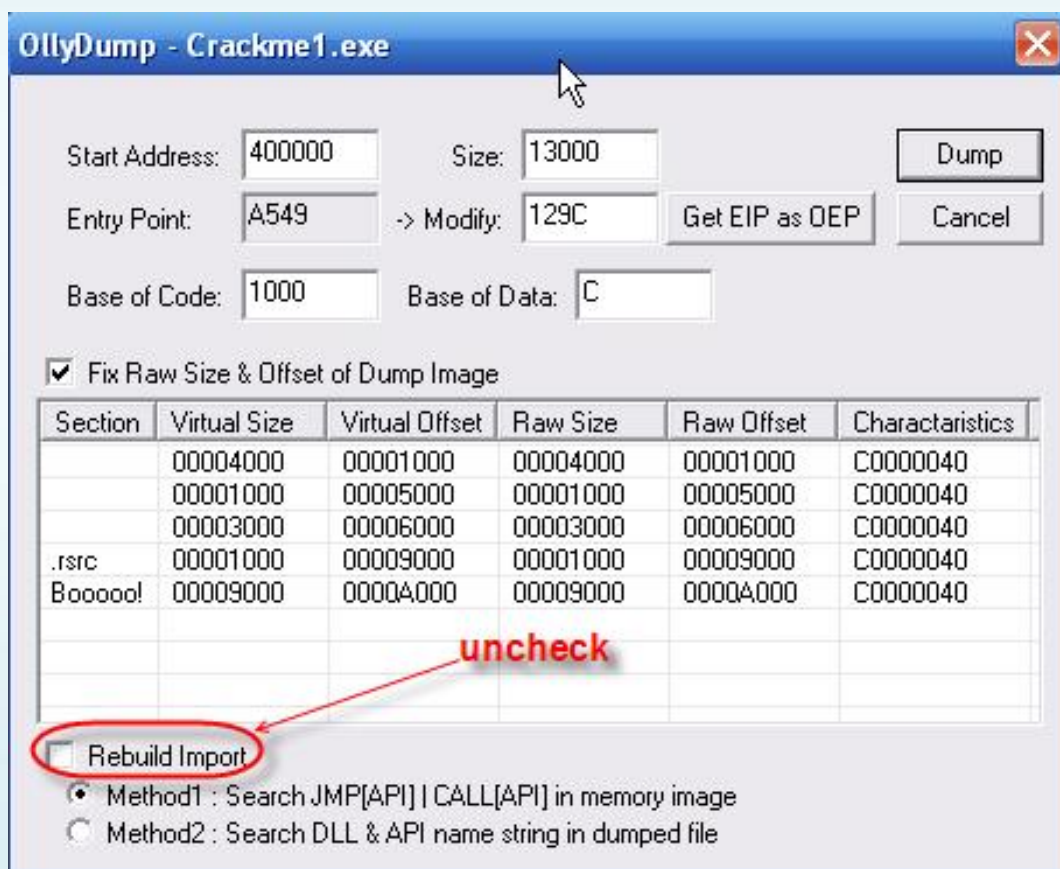
003F0000	00001000				Priv	RW	RW	
00400000	00001000	Crackme1		PE header	Imag	RW	RWE	
00401000	00004000	Crackme1		code, data				
00405000	0000E000				<div> Actualize View in Disassembler Dump in CPU Dump Search Set break-on-access Set memory breakpoint on access Set memory breakpoint on write </div>			
00420000	00103000							
00530000	0007A000							
00830000	00001000							
00840000	00002000							
00850000	00003000							
00860000	00005000							
00870000	00050000							
008C0000	00001000							
0094F000	00001000							

HarddiskVo1

_Sau When placed on the BP, one click is run target, immediately Olly will break here:

Address	Hex dump	Disassembly	Comment
0040129C	. 55	PUSH EBP	
0040129D	. 8BEC	MOV EBP,ESP	
0040129F	. 8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]	
004012A2	. 83E8 10	SUB EAX,10	Switch (cases 10..111)
004012A5	~ 0F84 A0000000	JE Crackme1.0040134B	
004012A8	. 2D 00010000	SUB EAX,100	
004012B0	~ 74 50	JE SHORT Crackme1.00401302	
004012B2	. 48	DEC EAX	
004012B3	~ 0F85 9D000000	JNZ Crackme1.00401356	
004012B9	. 0FB745 10	MOVZX EAX,WORD PTR SS:[EBP+10]	Case 111 (WM COMMAND) of
004012BD	. 2D EB030000	SUB EAX,3EB	Switch (cases 3EB..3ED)
004012C2	~ 74 30	JE SHORT Crackme1.004012F4	
004012C4	. 48	DEC EAX	
004012C5	~ 74 10	JE SHORT Crackme1.004012D7	
004012C7	. 48	DEC EAX	
004012C8	~ 75 33	JNZ SHORT Crackme1.004012FD	
004012CA	. 6A 00	PUSH 0	[Arq2 = 00000000; Case 3E
004012CC	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	
004012CF	. FF15 9C504000	CALL DWORD PTR DS:[40509C]	Arq1 0014B336
004012D5	~ EB 26	JMP SHORT Crackme1.004012FD	
004012D7	> 6A 00	PUSH 0	Case 3EC of switch 00401
004012D9	. 68 5C134000	PUSH Crackme1.0040135C	
004012DE	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	

_Tai Here we will dump the target using **OllyDump Plugin** by Olly. Save with a name any, do not close the Olly:



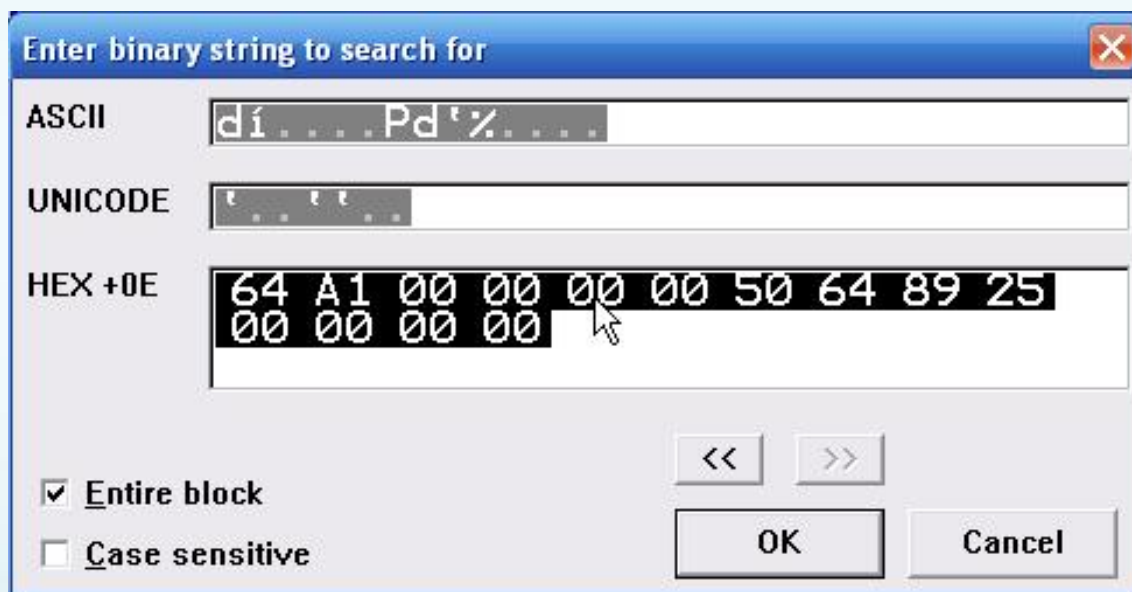
_Dung PeiD to scan the dump file, we have information as follows:



_Day Information is very important for the next step of us. Using **PeiD** to find a file that is compiled using **MS Visual C++ 6.0**. For example, I found 1 files follows: **# CoSH Crackme 2.exe**. Open a window Olly and other files to load. We have been as follows:

Address	Hex dump	Disassembly	Comment
004018A0	55	PUSH EBP	
004018A1	8BEC	MOV EBP,ESP	
004018A3	6A FF	PUSH -1	
004018A5	68 E8244000	PUSH CoSH Cra.004024E8	
004018AA	68 261A4000	PUSH <JMP.&MSUCRT. except handler3>	SE handler installation
004018AF	64:A1 000000	MOV EAX,DWORD PTR FS:[0]	
004018B5	50	PUSH EAX	
004018B6	64:8925 0000	MOV DWORD PTR FS:[0],ESP	
004018BD	83EC 68	SUB ESP,68	
004018C0	53	PUSH EBX	
004018C1	56	PUSH ESI	
004018C2	57	PUSH EDI	
004018C3	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
004018C6	33DB	XOR EBX,EBX	
004018C8	895D FC	MOV DWORD PTR SS:[EBP-4],EBX	
004018CB	6A 02	PUSH 2	
004018CD	FF15 C0214000	CALL DWORD PTR DS:[<&MSUCRT. set ap	MSUCRT. set app type
004018D3	59	POP ECX	

OEP _Ta found packed in a file that we find similar to what we have been in the picture, so we need to find information on opcodes in this file has the features dump. Su **Binary copy** of Olly to copy information from **0x004018AF** to address **0x004018B6**, then return to the screen by Olly target, roll your mouse over to the address **0x00401000**. Click to select **Search> Binary String (Ctrl + B)** and paste the following:



_Sau Then click OK, we'll stop here in Olly:

Address	Hex dump	Disassembly	Comment
004014B4	. 55	PUSH EBP	
004014B5	. 8BEC	MOV EBP, ESP	
004014B7	. 6A FF	PUSH -1	
004014B9	. 68 C0504000	PUSH 4050C0	
004014BE	. 68 9C224000	PUSH 40229C	SE handler installa
004014C3	. 64:A1 00000000	MOV EAX, DWORD PTR FS:[0]	
004014C9	. 50	PUSH EAX	
004014CA	. 64:8925 00000000	MOV DWORD PTR FS:[0], ESP	
004014D1	. 83EC 58	SUB ESP, 58	
004014D4	. 53	PUSH EBX	
004014D5	. 56	PUSH ESI	
004014D6	. 57	PUSH EDI	
004014D7	. 8965 E8	MOV DWORD PTR SS:[EBP-18], ESP	
004014DA	. FF15 30504000	CALL DWORD PTR DS:[405030]	
004014E0	. 33D2	XOR EDX, EDX	
004014E2	. 8AD4	MOV DL, AH	
004014E4	. 8915 80854000	MOV DWORD PTR DS:[408580], EDX	
004014EA	. 8BC8	MOV ECX, EAX	
004014EC	. 81E1 FF000000	AND ECX, 0FF	
004014F2	. 890D 7C854000	MOV DWORD PTR DS:[40857C], ECX	

_Trong Case if you find another similar position we need to consider with care. Next, select the **004014DA |. FF15 30504000 CALL DWORD PTR DS: [405030]**, press **Enter** to follow orders CALL see this as anything, we have the following information:

Address	Hex dump	Disassembly	Comment
0014B2B4	- E9 CD11D377	JMP kernel32.GetVersion	
0014B2B9	- E9 F7A9D277	JMP kernel32.ExitProcess	
0014B2BE	- E9 D462E077	JMP ntdll.RtlFreeHeap	
0014B2C3	- E9 3064E077	JMP ntdll.RtlAllocateHeap	
0014B2C8	- E9 E763D177	JMP kernel32.TerminateProcess	
0014B2CD	- E9 BEE9D277	JMP kernel32.GetCurrentProcess	
0014B2D2	- E9 ADE7D677	JMP kernel32.UnhandledExceptionFilter	
0014B2D7	- E9 BDEDD277	JMP kernel32.GetModuleFileNameA	
0014B2DC	- E9 D012D577	JMP kernel32.FreeEnvironmentStringsA	
0014B2E1	- E9 FB16D377	JMP kernel32.FreeEnvironmentStringsA	
0014B2E6	- E9 39E6D277	JMP kernel32.WideCharToMultiByte	
0014B2EB	- E9 12C4D177	JMP kernel32.GetEnvironmentStringsA	
0014B2F0	- E9 ECC8D277	JMP kernel32.GetEnvironmentStringsW	
0014B2F5	- E9 3716D377	JMP kernel32.SetHandleCount	
0014B2FA	- E9 3EE9D277	JMP kernel32.GetStdHandle	
0014B2FF	- E9 02D1D277	JMP kernel32.GetFileType	
0014B304	- E9 55F9D277	JMP kernel32.GetEnvironmentVariableA	
0014B309	- E9 4013D377	JMP kernel32.GetVersionExA	

_Chung We notice that it points to Redirected API, in the position as we see the pictures that **GetVersion** API, the API is often used in the first version of the VC + +. From that we conclude we are in need to find OEP:

004014B4 /. 55 PUSH EBP <== Our OEP

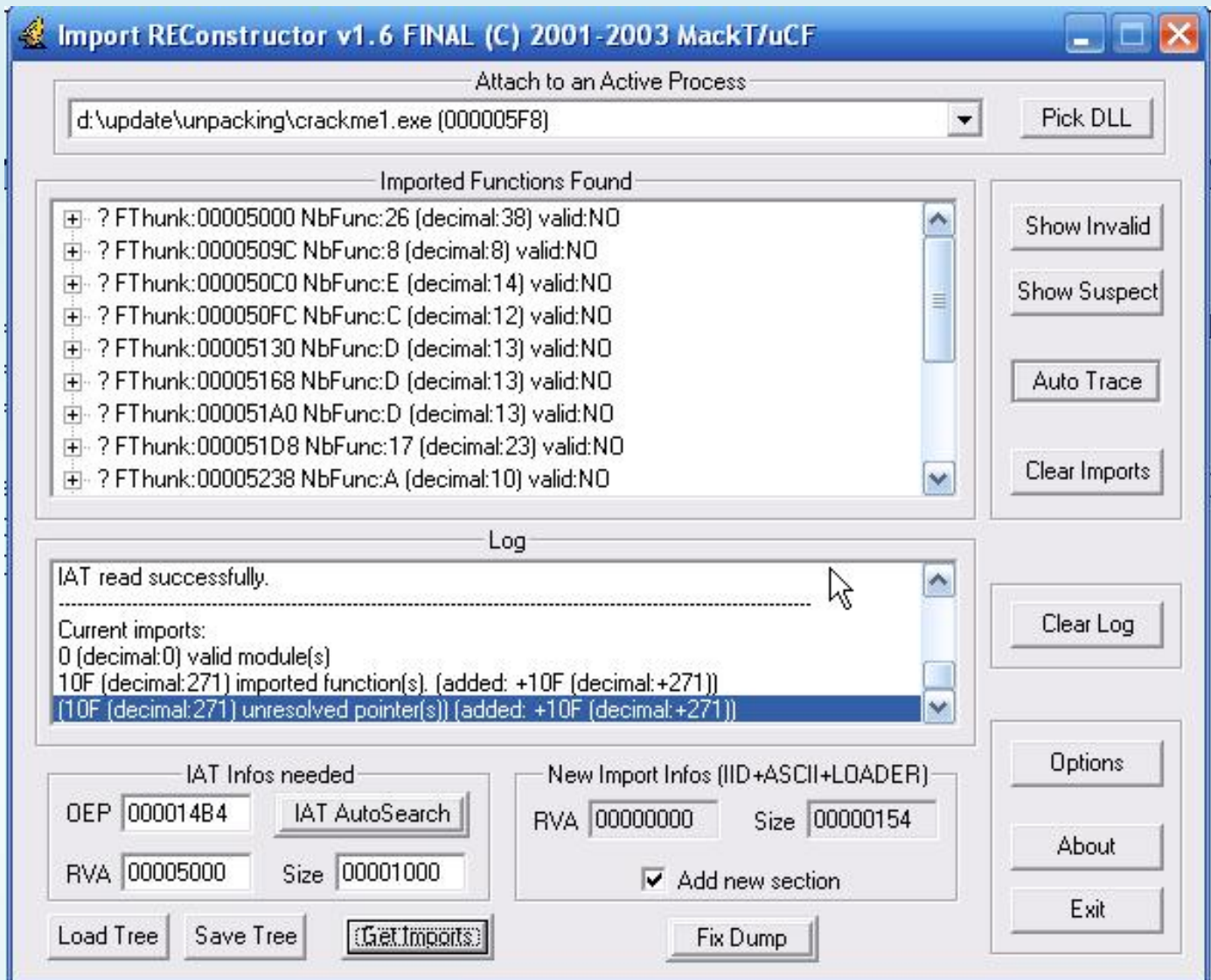
004014B5 |. 8BEC MOV EBP, ESP

004014B7 |. 6A FF PUSH -1

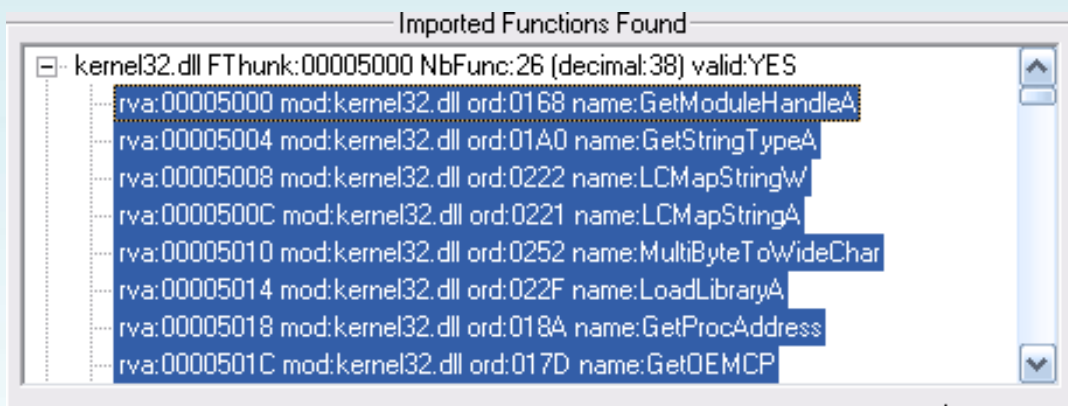
_Tai Will have 2 hours to fix IAT, the first **ImportRec** open up, select the correct process is our target and complete information to OEP as follows: **000014B4. IAT** Click **Auto Search** we receive notice as follows:



Ok _Nhan Size and enter the ImportRec have suggestions for which ta.Sau **Get Imports**, AC we are all Invalid:



_Khong Stars, we will use the features of ImportRec for the API. Click on **Show Invalid**, right on the mouse and select Rva **Trace Level1 (Diasm)** we have been as follows:

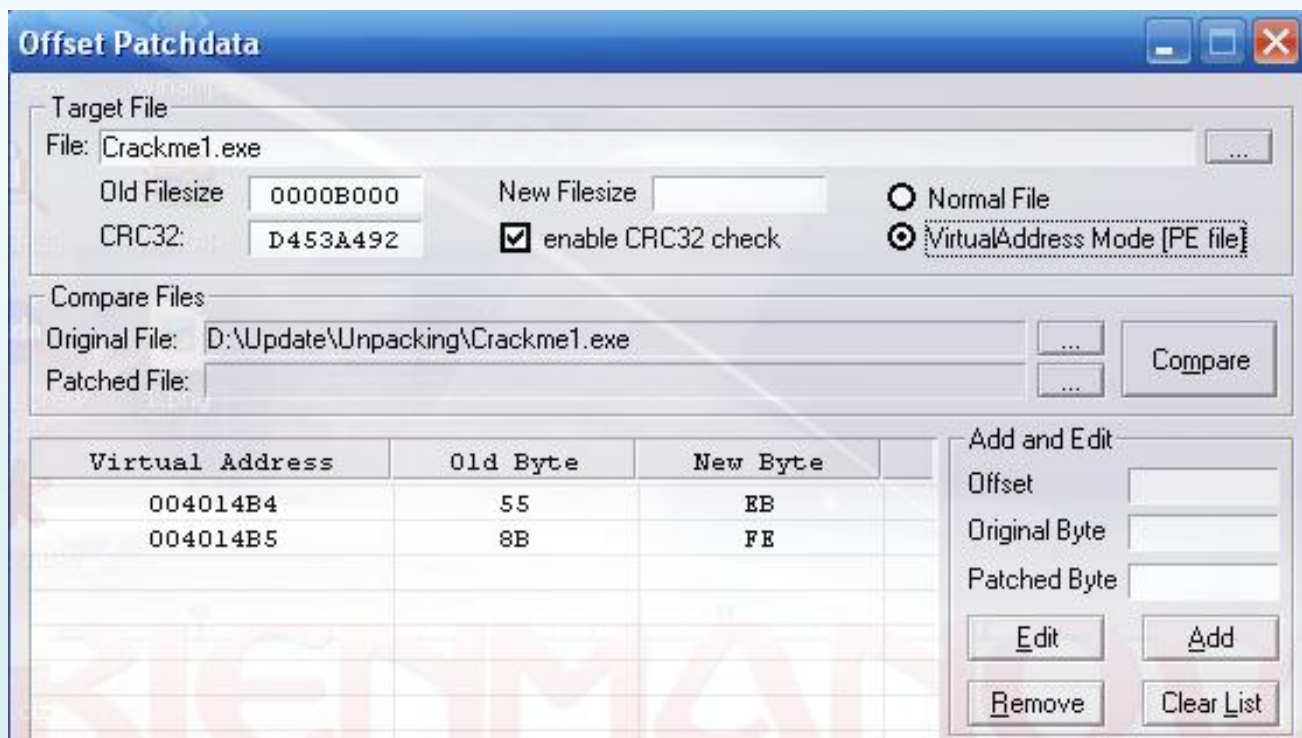


Show hit by **_Tiep Invalid Cut** and **thunks** all, finally selecting **Fix dump** file and have them dump ta. Thoat Import Rec try and test the file has Fix IAT. Keke Run it then kìa:

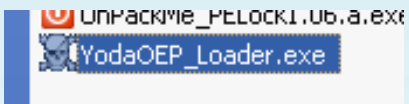


_Den Is complete, we are then MUPing it. However, do not understand why aged **Hangar** do in this manner is not, the file of Laos after being run IAT Fix crash. Self-aged and do not understand the real cause lies where, aged said that should not run to Target and then dump it. Must stop at the OEP, then a new dump and fix imports. Therefore aged this idea as follows: to create a loader implementation of the OEP to find and then replaced by **EBFE**. This is an endless loop, this means that the file on the memory will unpack a whole, but will not be thi. Sau that we can Attach target and make the rest of the story. Tolerable seems complex includes J .

_De Create loader we will use **Dup2** by **diablo2oo2**:



_Save Loader in this folder contains our Target.



Loader _Double click create, AC automatically stop seeing khiếp always slow machines, up 100&percent; CPU, hii simply because we are a respectable infinite loop. It is a good sign because loader we create has the right job that it needs to do. Now we open up and Olly Attach to target Olly. **Alt + E Executables** open window, select the target and double click on it, we return to the main screen by Olly:

Address	Hex dump	Disassembly	Comment
00401000	\$ 33C0	XOR EAX,EAX	
00401002	. 394424 08	CMP DWORD PTR SS:[ESP+8],EAX	
00401006	~ 7E 14	JLE SHORT Crackme1.0040101C	
00401008	> 8B4C24 04	MOV ECX,DWORD PTR SS:[ESP+4]	
0040100C	. 8B0C81	MOV ECX,DWORD PTR DS:[ECX+EAX*4]	
0040100F	. 3B4C24 0C	CMP ECX,DWORD PTR SS:[ESP+C]	
00401013	~ 74 0A	JE SHORT Crackme1.0040101F	
00401015	. 40	INC EAX	
00401016	. 3B4424 08	CMP EAX,DWORD PTR SS:[ESP+8]	
0040101A	^ 7C EC	JL SHORT Crackme1.00401008	
0040101C	> 32C0	XOR AL,AL	
0040101E	. C3	RETN	
0040101F	> B0 01	MOV AL,1	
00401021	. C3	RETN	

_Sau The open window **Memory (Alt + M)**, BP set on the access code section.Tiep the press **F9** to run, we will OEP at Break:

Address	Hex dump	Disassembly	Comment
004014B4	> EB FE	JMP SHORT Crackme1.004014B4	<== Our OEP
004014B6	. EC	IN AL,DX	SE handler installa
004014B7	. 6A FF	PUSH -1	
004014B9	. 68 C0504000	PUSH 4050C0	
004014BE	. 68 9C224000	PUSH 40229C	
004014C3	. 64:A1 00000000	MOV EAX,DWORD PTR FS:[0]	
004014C9	. 50	PUSH EAX	
004014CA	. 64:8925 00000000	MOV DWORD PTR FS:[0],ESP	
004014D1	. 83EC 58	SUB ESP,58	
004014D4	. 53	PUSH EBX	
004014D5	. 56	PUSH ESI	
004014D6	. 57	PUSH EDI	

Kha _Kha you have not seen anything, we're in OEP (infinite loop), the next one is replacing **EBFE on 558B**. At press **Ctrl + E** and edit:

Edit code at 004014B4

ASCII

U

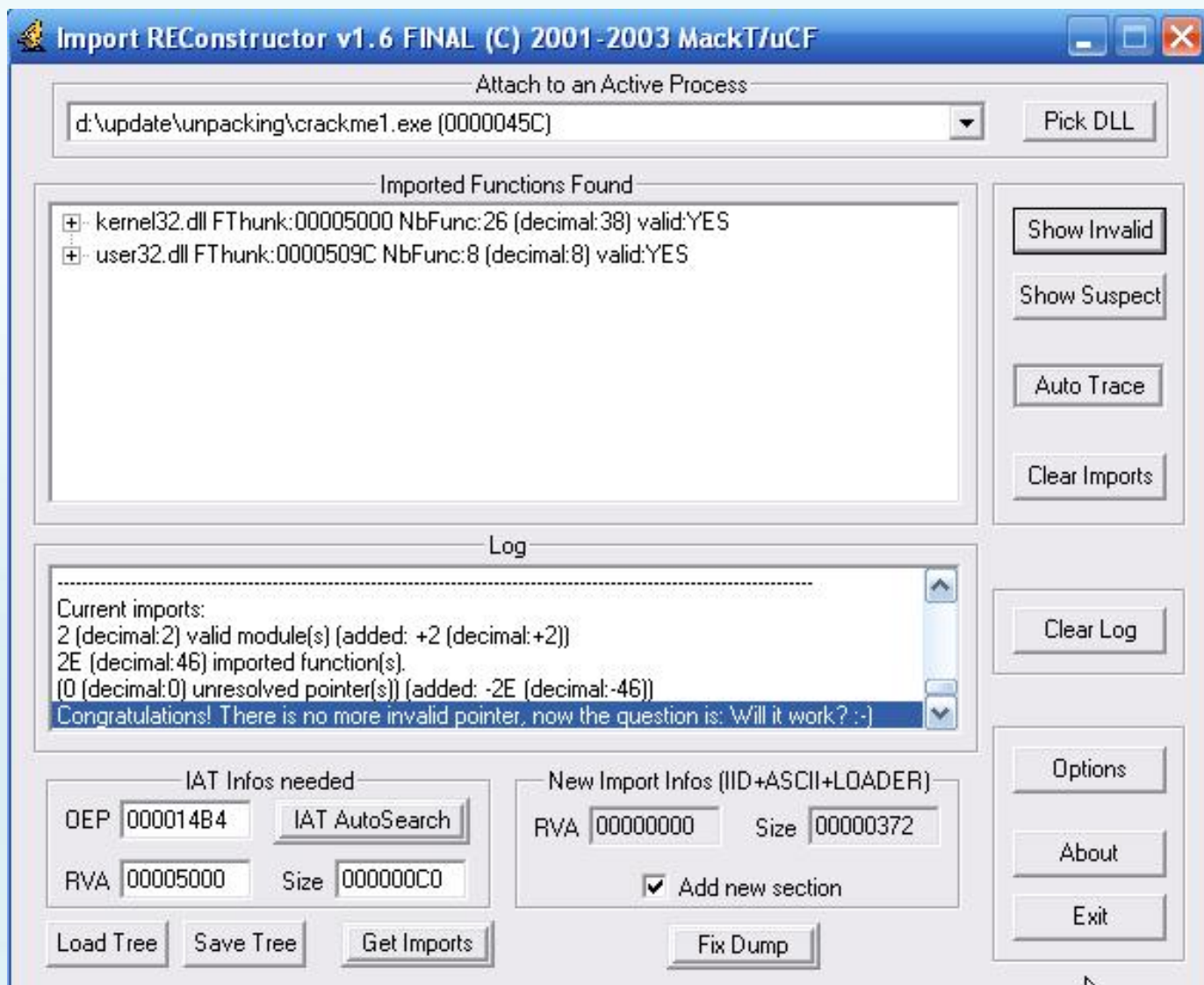
UNICODE

HEX +02

55 8B

Address	Hex dump	Disassembly	Comment
004014B4	55	PUSH EBP	<== Our OEP
004014B5	8BEC	MOV EBP,ESP	SE handler installa
004014B7	. 6A FF	PUSH -1	
004014B9	. 68 C0504000	PUSH 4050C0	
004014BE	. 68 9C224000	PUSH 40229C	

_Phu, Not writing long tut tired out phết J. Now the end of last MUPing stories. Dump in the target as the Olly said, then open ImportRec fill OEP and dump Fix:



_Run Try to see what what kaka run the service. So **yoda's protector 1.03.2** officially dust and J:



Best Regards

_ [Kienmanowar] _

---+---=[**Greatz thanks to**]=---+---

My family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHOCrker, the_Lighthouse, Merc, Hoadongnoi, Nini ... all REA's members, TQN, HacNho, RongChauA, Deux, tlandn, light.phoenix, dqtIn, ARTEAM All my friend, and you.

Thanks to --+ +---==[]=---+ +--

iamidiot, WhyNotBar, trickyboy, dzungltvn, takada, hurt_heart, haule_nth, v. hytkl. v.. You have contributed greatly to the REA. Hope you will continue to promote J

>>> If you have any suggestions, comments or corrections email me: **kienbigmummy [at] gmail.com**

Yoda's Protector v1.03.2 - manually unpacking tutorial (_kienmanowar_)

I. Introduction:

I wrote some tutorials about MUP protector for this forum (www.reaonline.net), but in English. Today, I try to MUP unpackme from an www.tuts4you.com (**UnPackMe_Yoda'sProtector1 .03.2. E.exe**) and decide to write another tutorial about this protector. I hope with this tutorial, it can give you some idea to work with this protector. Have fun and enjoy it! J

II. Tools & Target:

_ ODbyDYK v1.10
_ PeiD v0.94
_ ImportRec v1.6
_ UnPackMe_Yoda'sProtector1 .03.2. E.exe

III. MUPing:

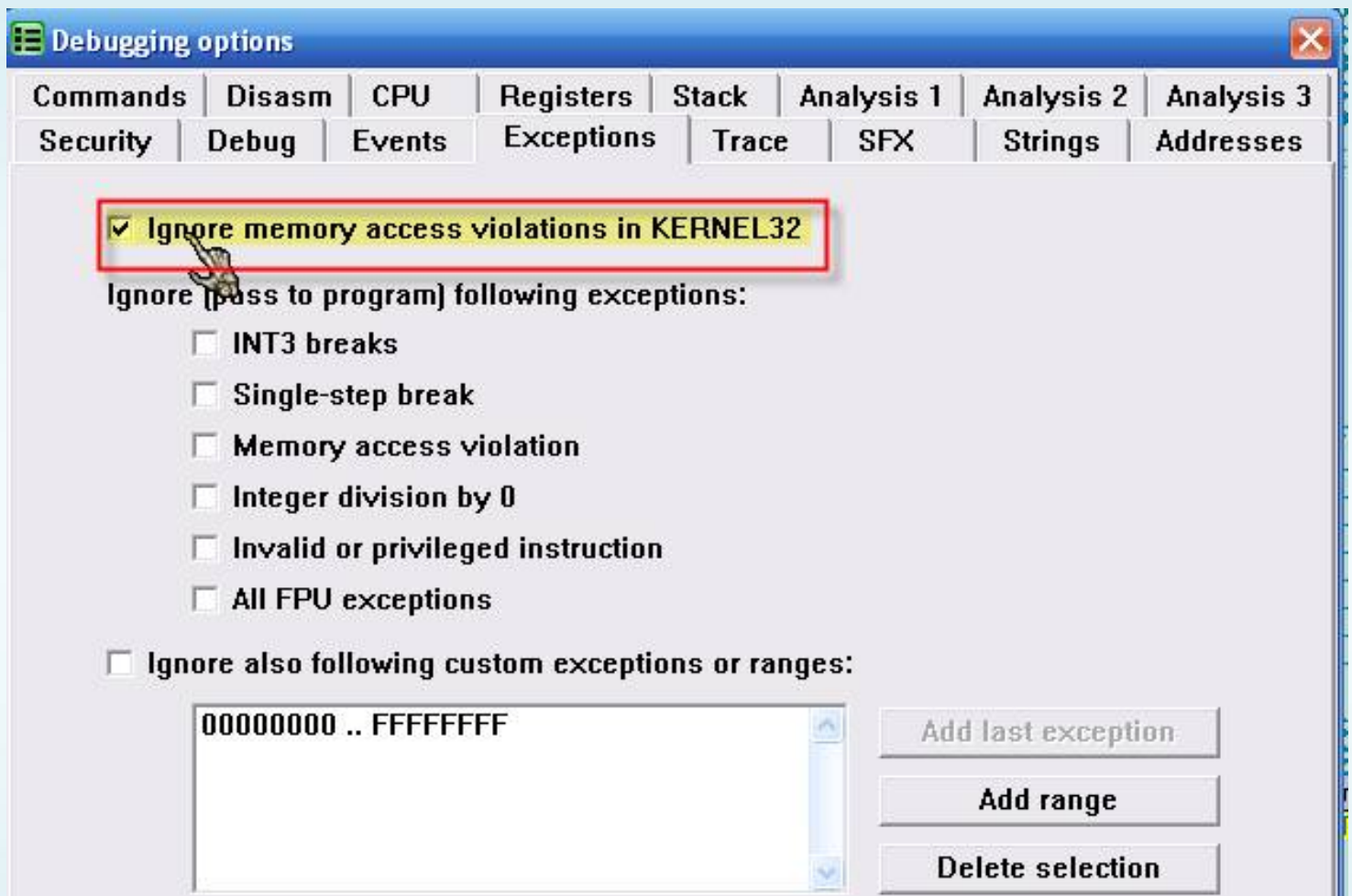
_Use **PEiD v.0.94** to detect and get some information:



_After That, **Olllydbg** open and load it into target. If a message box appears "... **Do you want to continue analysis**", click **No** to continue:

Address	Hex dump	Disassembly	Comment
00411549	E8 03000000	CALL UnPackMe.00411551	
00411551	EB 01	JMP SHORT UnPackMe.00411551	
00411553	E8 BB550000	CALL UnPackMe.00416B10	
00411555	00E8	ADD AL,CH	
00411557	0300	ADD EAX,DWORD PTR DS:[EAX]	
00411559	0000	ADD BYTE PTR DS:[EAX],AL	
0041155B	EB 01	JMP SHORT UnPackMe.0041155E	
0041155D	E8 E88F0000	CALL 0041A54A	
00411562	00E8	ADD AL,CH	
00411564	0300	ADD EAX,DWORD PTR DS:[EAX]	
00411566	0000	ADD BYTE PTR DS:[EAX],AL	
00411568	EB 01	JMP SHORT UnPackMe.0041156B	
0041156A	E9 E8820000	JMP UnPackMe.00419857	
0041156F	00E8	ADD AL,CH	
00411571	0300	ADD EAX,DWORD PTR DS:[EAX]	
00411573	0000	ADD BYTE PTR DS:[EAX],AL	
00411575	EB 01	JMP SHORT UnPackMe.00411578	
00411577	C2 E8B8	RETN 0B8E8	
0041157A	0000	ADD BYTE PTR DS:[EAX],AL	
0041157C	00E8	ADD AL,CH	
0041157E	0300	ADD EAX,DWORD PTR DS:[EAX]	
00411580	0000	ADD BYTE PTR DS:[EAX],AL	

We stop at **_Okie Entry point** of this target. Now we have to configure **Exceptions** in Ollydbg target to work with. Uncheck **Ignore** all exceptions except **memory access violations** in **KERNEL32**.



_Then Press **OK** to accept this configuration. Print Ollydbg, press **Shift + F9** and count how many times until our target runs completely. On my machine, I press 8 times (may be differ from your machine)

_ **Ctrl + F2** to restart our target and press **Shift + F9** (n-1) times. Olly will pause at last exception as the following picture below:

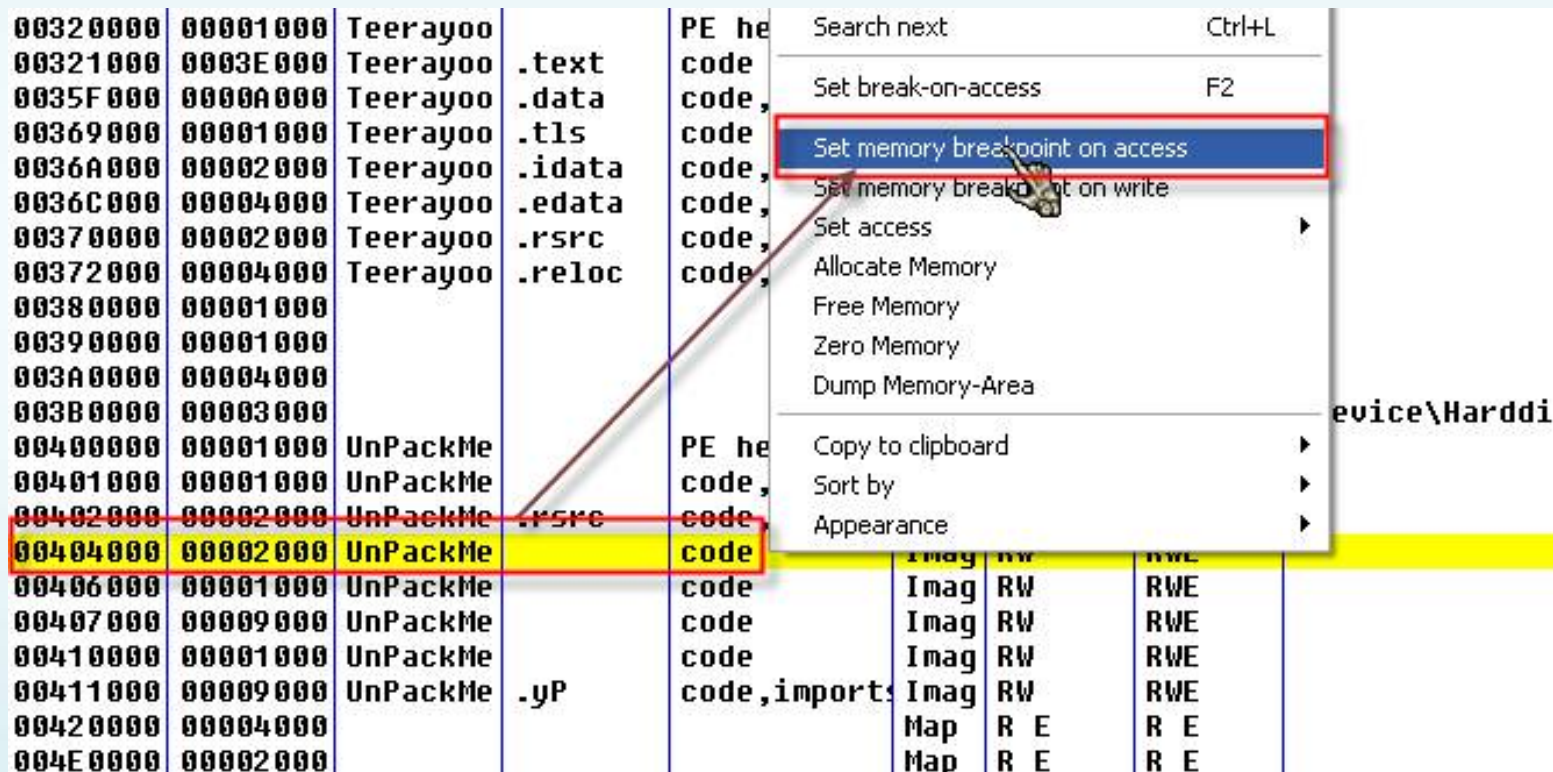
Address	Hex dump	Disassembly	Comment
004141BD	0000	ADD BYTE PTR DS:[EAX],AL	
004141BF	0000	ADD BYTE PTR DS:[EAX],AL	
004141C1	0000	ADD BYTE PTR DS:[EAX],AL	
004141C3	0000	ADD BYTE PTR DS:[EAX],AL	
004141C5	0000	ADD BYTE PTR DS:[EAX],AL	
004141C7	0000	ADD BYTE PTR DS:[EAX],AL	
004141C9	0000	ADD BYTE PTR DS:[EAX],AL	
004141CB	0000	ADD BYTE PTR DS:[EAX],AL	
004141CD	0000	ADD BYTE PTR DS:[EAX],AL	
004141CF	0000	ADD BYTE PTR DS:[EAX],AL	
004141D1	0000	ADD BYTE PTR DS:[EAX],AL	
004141D3	0000	ADD BYTE PTR DS:[EAX],AL	
004141D5	0000	ADD BYTE PTR DS:[EAX],AL	
004141D7	0000	ADD BYTE PTR DS:[EAX],AL	
004141D9	0000	ADD BYTE PTR DS:[EAX],AL	
004141DB	0000	ADD BYTE PTR DS:[EAX],AL	
004141DD	0000	ADD BYTE PTR DS:[EAX],AL	
004141DF	0000	ADD BYTE PTR DS:[EAX],AL	
004141E1	0000	ADD BYTE PTR DS:[EAX],AL	
004141E3	0000	ADD BYTE PTR DS:[EAX],AL	
004141E5	0000	ADD BYTE PTR DS:[EAX],AL	

Access violation when writing to [00000000] - use Shift+F7/F8/F9 to pass exception to program

_At This position, we have two methods to find the **OEP** of this protector. I will show you now.

1st Method:

_Press **Alt + M** to open the **Memory Map Window**, and set a breakpoint on memory access code section at the target. Like the following picture below:



_Okie, Finish him now! J . Press **Shift + F9** times and only one kaka **OEP** we land at the target:

Address	Hex dump	Disassembly	Comment
00404000	9B	WAIT	<== Our OEP
00404001	DBE3	FINIT	
00404003	9B	WAIT	
00404004	DBE2	FCLEX	
00404006	D92D 00604000	FLDCW WORD PTR DS:[406000]	
0040400C	55	PUSH EBP	
0040400D	89E5	MOV EBP,ESP	
0040400F	E8 01040000	CALL UnPackMe.00404415	
00404014	68 00000000	PUSH 0	
00404019	FF15 F4114000	CALL DWORD PTR DS:[4011F4]	kernel32.GetModuleHandleA
0040401F	A3 07F04000	MOV DWORD PTR DS:[40F007],EAX	
00404024	60	PUSHAD	
00404025	8925 0BF04000	MOV DWORD PTR DS:[40F00B],ESP	
0040402B	E9 30000000	JMP UnPackMe.00404060	
00404030	8B25 0BF04000	MOV ESP,DWORD PTR DS:[40F00B]	
00404036	61	POPAD	
00404037	E8 19090000	CALL UnPackMe.00404955	
0040403C	E8 6D040000	CALL UnPackMe.004044AE	
00404041	89EC	MOV ESP,EBP	
00404043	5D	POP EBP	
00404044	FF35 D4F14000	PUSH DWORD PTR DS:[40F1D4]	
0040404A	FF15 EC114000	CALL DWORD PTR DS:[4011EC]	kernel32.ExitProcess
00404050	9B	WAIT	
00404051	DBE2	FCLEX	
00404053	D92D 00604000	FLDCW WORD PTR DS:[406000]	
00404059	C3	RETN	

2nd method:

_At The last exception which we pause as you see in the picture above. Check the **Stack Window** and you will yoda's address of exception handler:

Address	Value	Comment
0012F808	0012EC84	Pointer to next SEH record
0012EB0C	00413F50	SE handler
0012EB10	0041263E	UnPackMe.0041263E
0012EB14	0041262B	UnPackMe.0041262B
0012EB18	0041261E	UnPackMe.0041261E
0012EB1C	00412438	UnPackMe.00412438
0012EB20	004121C2	UnPackMe.004121C2
0012EB24	00412132	UnPackMe.00412132
0012EB28	00411F54	UnPackMe.00411F54
0012EB2C	00411E9F	UnPackMe.00411E9F
0012EB30	00411E82	UnPackMe.00411E82
0012EB34	00411E69	UnPackMe.00411E69
0012EB38	00411E50	UnPackMe.00411E50
0012EB3C	00411E37	UnPackMe.00411E37
0012EB40	00411E1E	UnPackMe.00411E1E
0012EB44	00411E05	UnPackMe.00411E05

Click on _Right address of SE Handler and select: **Follow in Disassembler**



_We Are here in Ollydbg:

Address	Hex dump	Disassembly	Comment
00413F50	55	PUSH EBP	
00413F51	8BEC	MOV EBP,ESP	
00413F53	57	PUSH EDI	
00413F54	36:8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]	
00413F58	3E:8BB8 C40000	MOV EDI,DWORD PTR DS:[EAX+C4]	
00413F5F	3E:FF37	PUSH DWORD PTR DS:[EDI]	
00413F62	33FF	XOR EDI,EDI	
00413F64	64:8F07	POP DWORD PTR FS:[EDI]	
00413F67	3E:8380 C40000	ADD DWORD PTR DS:[EAX+C4],8	
00413F6F	3E:8BB8 A40000	MOV EDI,DWORD PTR DS:[EAX+A4]	
00413F76	C1C7 07	ROL EDI,7	
00413F79	3E:89B8 B80000	MOV DWORD PTR DS:[EAX+B8],EDI	
00413F80	B8 00000000	MOV EAX,0	
00413F85	5F	POP EDI	
00413F86	C9	LEAVE	
00413F87	C3	RETN	

_Press **F2** to set BP at **0x00413F50**, then Press **Shift + F9** and you'll break there:

Address	Hex dump	Disassembly	Comment
00413F50	55	PUSH EBP	
00413F51	8BEC	MOV EBP,ESP	
00413F53	57	PUSH EDI	
00413F54	36:8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]	
00413F58	3E:8BB8 C40000	MOV EDI,DWORD PTR DS:[EAX+C4]	
00413F5F	3E:FF37	PUSH DWORD PTR DS:[EDI]	
00413F62	33FF	XOR EDI,EDI	
00413F64	64:8F07	POP DWORD PTR FS:[EDI]	
00413F67	3E:8380 C40000	ADD DWORD PTR DS:[EAX+C4],8	
00413F6F	3E:8BB8 A40000	MOV EDI,DWORD PTR DS:[EAX+A4]	
00413F76	C1C7 07	ROL EDI,7	
00413F79	3E:89B8 B80000	MOV DWORD PTR DS:[EAX+B8],EDI	
00413F80	B8 00000000	MOV EAX,0	
00413F85	5F	POP EDI	
00413F86	C9	LEAVE	
00413F87	C3	RET	

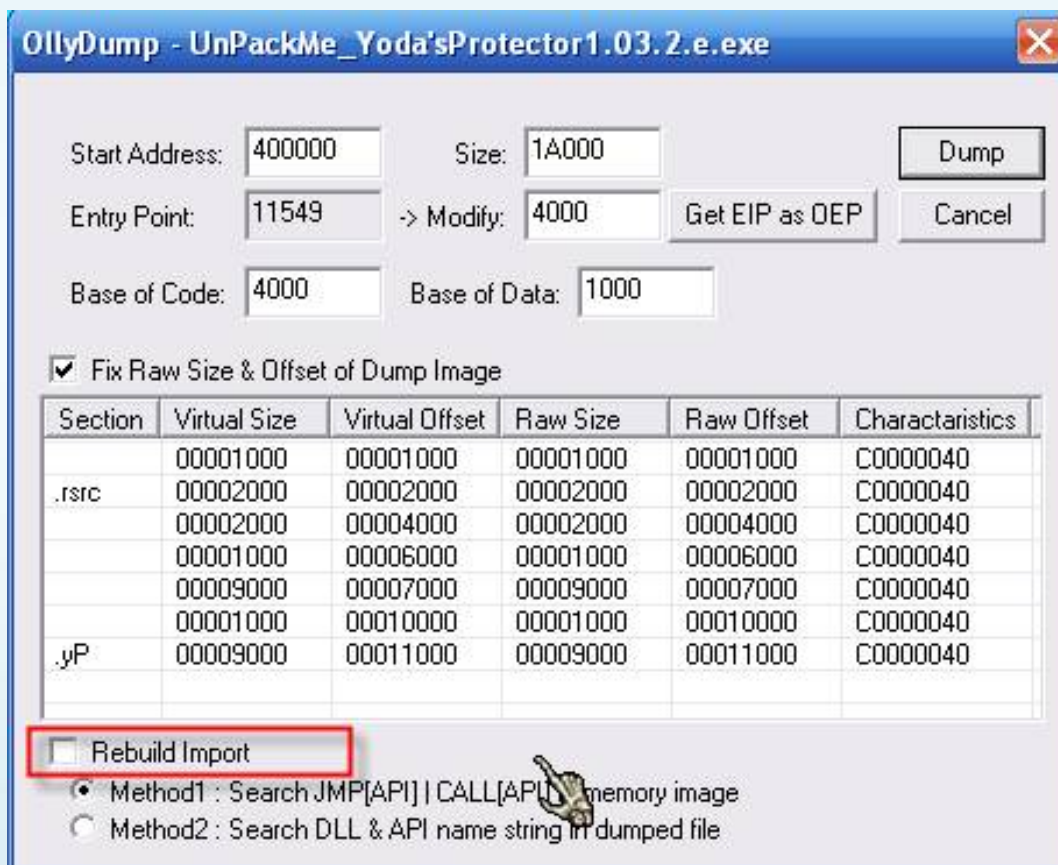
_Remove Break point and Press **F8** to trace down the line as marked in the picture above.

Address	Hex dump	Disassembly	Comment
00413F50	55	PUSH EBP	
00413F51	8BEC	MOV EBP,ESP	
00413F53	57	PUSH EDI	
00413F54	36:8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]	
00413F58	3E:8BB8 C40000	MOV EDI,DWORD PTR DS:[EAX+C4]	
00413F5F	3E:FF37	PUSH DWORD PTR DS:[EDI]	
00413F62	33FF	XOR EDI,EDI	
00413F64	64:8F07	POP DWORD PTR FS:[EDI]	
00413F67	3E:8380 C40000	ADD DWORD PTR DS:[EAX+C4],8	
00413F6F	3E:8BB8 A40000	MOV EDI,DWORD PTR DS:[EAX+A4]	
00413F76	C1C7 07	ROL EDI,7	
00413F79	3E:89B8 B80000	MOV DWORD PTR DS:[EAX+B8],EDI	UnPackMe.00404000
00413F80	B8 00000000	MOV EAX,0	
00413F85	5F	POP EDI	
00413F86	C9	LEAVE	
00413F87	C3	RET	

_EDI Hold **0x00404000** value which is **OEP** address. Simply go to **0x00404000**, BP place there and press **Shif + F9**. You will break on **OEP**:

Address	Hex dump	Disassembly	Comment
00404000	9B	WAIT	<= Our OEP
00404001	DBE3	INIT	
00404003	9B	WAIT	
00404004	DBE2	FCLEX	
00404006	D92D 00604000	FLDCW WORD PTR DS:[406000]	
0040400C	55	PUSH EBP	
0040400D	89E5	MOV EBP,ESP	
0040400F	E8 01040000	CALL UnPackMe.00404415	
00404014	68 00000000	PUSH 0	
00404019	FF15 F4114000	CALL DWORD PTR DS:[4011F4]	kernel32.GetModuleHandle
0040401F	A3 07F04000	MOV DWORD PTR DS:[40F007],EAX	
00404024	60	PUSHAD	
00404025	8925 0BF04000	MOV DWORD PTR DS:[40F00B],ESP	
0040402B	E9 30000000	JMP UnPackMe.00404060	
00404030	8B25 0BF04000	MOV ESP,DWORD PTR DS:[40F00B]	
00404036	61	POPAD	
00404037	E8 19090000	CALL UnPackMe.00404955	
0040403C	E8 6D040000	CALL UnPackMe.004044AE	
00404041	89EC	MOV ESP,EBP	
00404043	5D	POP EBP	
00404044	FF35 D4F14000	PUSH DWORD PTR DS:[40F1D4]	
0040404A	FF15 EC114000	CALL DWORD PTR DS:[4011EC]	kernel32.ExitProcess

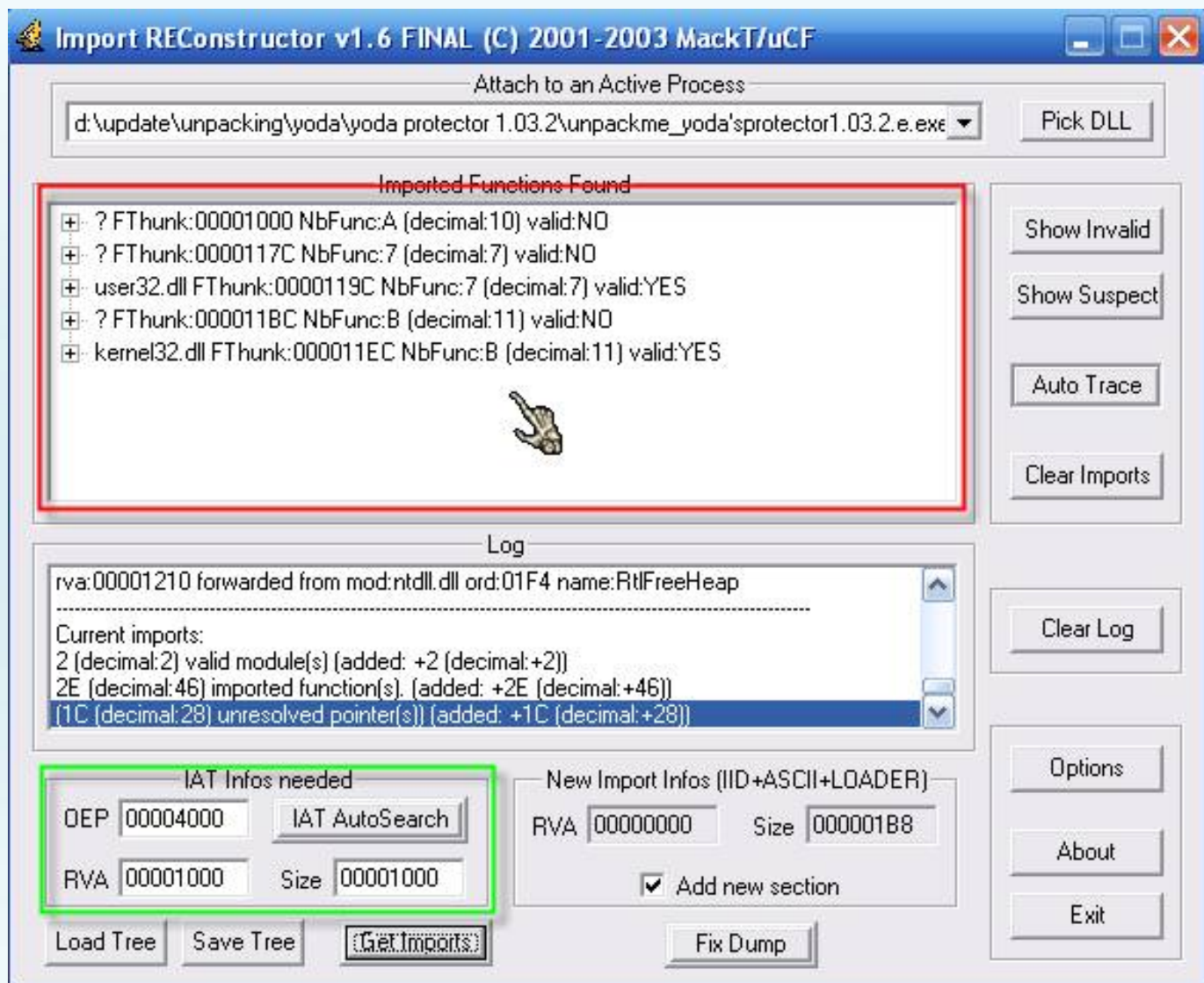
OEP _After stop at the target, we will dump and Fix IAT. Use **Ollydump** Plugin to dump and save as with any name you want.



_Open **ImportRec**, load Process OEP and modify information in the IAT. Then press **IAT AutoSearch**, we get a messagebox:



Size and **_Modify RVA** same values as information which suggested ImportRec for us. **Get Press Imports** and we receive:



Show press **_Next Invalid Cut** and **thunks** all invalid thunks, press **Fix** Finally **dump** dumped the file and select to fix. Quit **ImportRec** and test our fixed file. Wow! Runs it

immediately:



_And That was all. It was not hard, is not it:)

Best Regards

_ [Kienmanowar] _

--+ +---==[**Greatz thanks to**]==---+ +---

My family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHQCrkcr, the_Lighthouse, Merc, Hoadongnoi, Nini ... all REA's members, TQN, HacNho, RongChauA, Deux, tlandn, light.phoenix, dqtlN, ARTEAM all my friend, and you.

Thanks to --+ +---==[]==---+ +---

iamidiot, WhyNotBar, trickyboy, dzungltn, takada, hurt_heart, haule_nth, v. hytkl. v..

I want to thank **Teddy Rogers** for his great site, **Reversing.be** folks, **Arteam** folks and all folks on **crackmes.de**, thank to all members of unpack.cn (especially **fly** and **linhanshi**).

Sorry guys for grammar lot of mistakes, I tried my best but who has strenght for checking text after Writting. Because English is not my mother language J . See you!

>>>> If you have any suggestions, comments or corrections email me: **kienbigmummy [at] gmail.com**

Yoda's Protector v1.03.3 - manually unpacking tutorial

(_kienmanowar_)

I. Introduction:

In previous articles I have to guide you on MUP **version 1.03.2**, adding that **Whynotbar** also mentioned the way to work on this version. Posts follow me here and you will also work on other version is: **v1.03.3**. As I know it probably is the final version of **Yoda's Protector** and the author of it also has decided not relax and develop more ... hii .. but planning to start a Newer solutions J. This new version of Yoda's Protector have mechanisms Terminating Olly and can make the OS we were close with hic hic.

Yoda Protector on Yoda's Cryptor, by the time the new technical to be applied to how to protect programs Protect by Yoda. The technical old was applied is **erasing PE header, CRC checking (code and file), check IsDebuggerPresent, API Redirecting and destroying import technical information**, to learn about the technical and this you can learn more about **ExeStealth v2.74a** (a protected Rip whole way of Yoda). The new technical later be applied as follows:

C of the Asian Anti - olly's where as follows, to use the combination of the Asian c h à m API to obtain the PID of all of Asia is the process to run the E N that of italy not for us. Then where will I m the search process has be Started (in the case of italy à l à Ollydbg) and to terminate that. It compares the PID of the process with its PID . If the PID is not the same (. Exe was started in Olly) it will terminate the process.

One way but it makes use we feel very distasteful that it is using the API **BlockInput** before it using the other test. API function will block all imported equipment as standard keyboard, mouse etc.. Etc.. so we are block from your system. Then it will implement measures to check and good conduct code. If during this time they stop at an Exception or it is found Olly, the system we will wait response from us (we are waiting for the impact) but we can not achieve any any impact except the single Restart the computer default default L If everything is pass by a good, it will again use the API function on again to unblock the device for us ... This can say is how fairly or Protect!

II. Tools & Target:

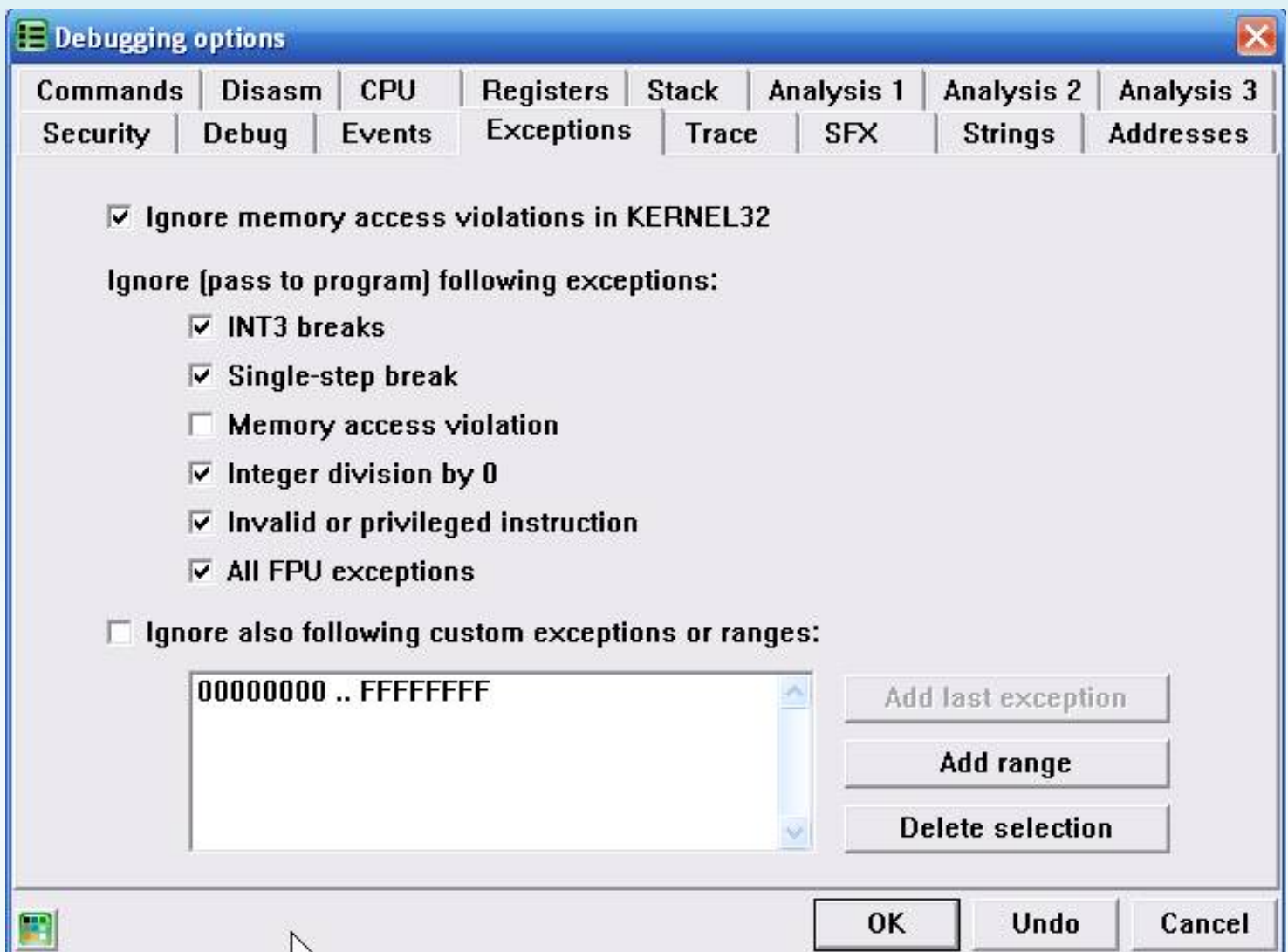
- _ **ODbyDYK v1.10**
- _ **PeiD v0.94**
- _ **LordPE**
- _ **ImportRec v1.6**
- _ **Crackme05.exe**

III. MUPing:

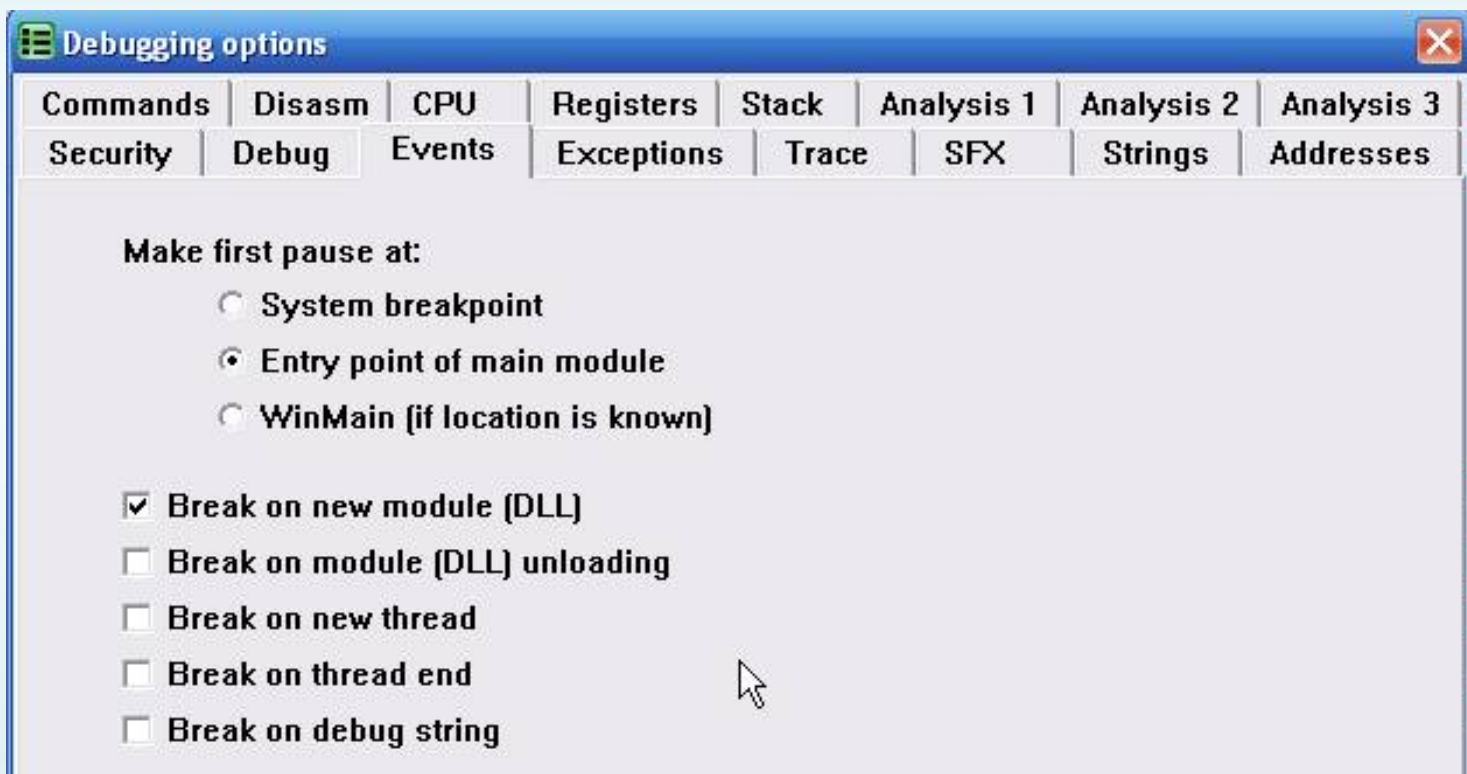
_Dung **PeiD** to get information on target:



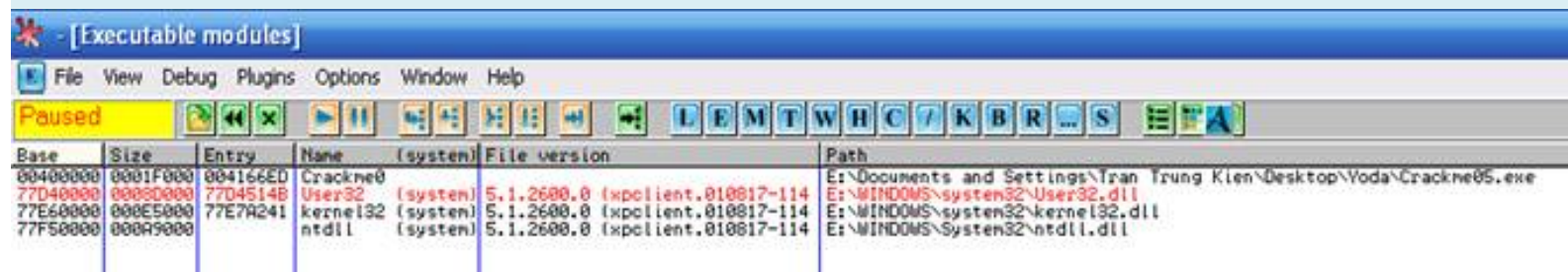
Olly _Mo and load up on target in Olly. In Olly you configure the Exceptions as follows:



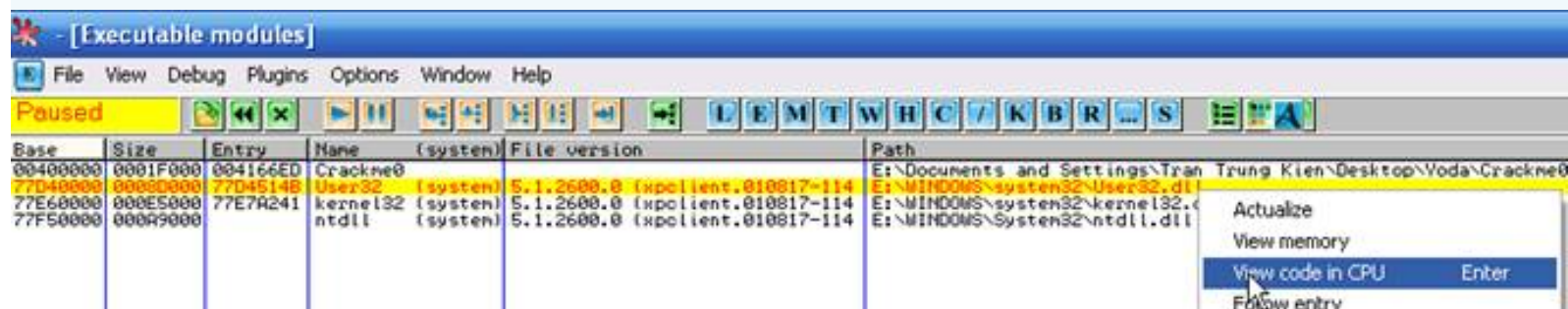
_Tiep **Events** tab in our configuration as follows:



_Cau Pictures by such we do not want to block the enforcement functions in **user32.dll** **BlockInput**. So when does it target Run Break when the load on **user32.dll**. After such a configuration, press **F9** to run target and observations in the window when **Executables** to load the **user32.dll** up:



User32.dll _Sau when the load up, re-configured to select Olly function **Break on new modules (dll)**. Next we'll patch it to function **BlockInput** not have the opportunity to lock the devices they ta.Tai screen **Executables**, right on the mouse and select **user32.dll** as follows:



_Sau Screen at the CPU press **Ctrl + G** to conduct search function API **BlockInput**:

Address	Hex dump	Disassembly	Comment
77D41000	3E:18F6	SBB DH,DH	
77D41003	77 20	JA SHORT User32.77D41025	
77D41005	12F5	ADC DH,CH	
77D41007	77 08		
77D41009	50		
77D4100A	FC		
77D4100B	77 F8		
77D4100D	16		
77D4100E	F5		
77D4100F	77 AA		
77D41011	1E		
77D41012	F677 24		
77D41015	FE		
77D41016	F5		
77D41017	77 05		
77D41019	B3 F7		
77D4101B	77 C6		
77D4101D	01 F7776F6F		

Enter expression to follow

BlockInput

☒ VA / API
☐ RVA
☐ Offset

OK Cancel

Unknown command

_Nhan Ok, we'll stay in here in Olly:

Address	Hex dump	Disassembly	Comment
77D99369	B8 3A110000	MOV EAX, 113A	
77D9936E	BA 0003FE7F	MOV EDX, 7FFE0300	
77D99373	FFD2	CALL EDX	
77D99375	C2 1000	RETN 10	

_Tien The patch by **NOP** 3 commands except command Retn 10, and set 1 in order BP Retn 10 as follows:

Address	Hex dump	Disassembly	Comment
77D99369	90	NOP	
77D9936A	90	NOP	
77D9936B	90	NOP	
77D9936C	90	NOP	
77D9936D	90	NOP	
77D9936E	90	NOP	
77D9936F	90	NOP	
77D99370	90	NOP	
77D99371	90	NOP	
77D99372	90	NOP	
77D99373	90	NOP	
77D99374	90	NOP	
77D99375	C2 1000	RETN 10	

_Coi As we kill the API and has been avoided from locked handcuffs leg hehe, hic next steps we must fight the anti Olly. How to bypass also are not very difficult, because Yoda use **CreateToolhelp32Snapshot** API function to get all the processes, and functions to get PID **GetCurrentProcessId** by itself. Then Yoda will check process that opened it coincides with PID it does not, if not the same as it will now Terminate the process. To bypass this process is done as follows: open **LordPE** and find information about PID by Olly:

e:\program files\microsoft office\office11\win...	0000068C	30000000	00BA4000
e:\documents and settings\tran trung kien\m...	000006FC	00400000	00181000
e:\documents and settings\tran trung kien\de...	000004F0	00400000	0001F000

_Sau It back Olly and press **Ctrl + G**, enter the name refers to [GetCurrentProcessId](#) to find it:



OK we will _Nhan in here in Olly:

Address	Hex dump	Disassembly	Comment
77E80656	64:A1 18000000	MOV EAX,DWORD PTR FS:[18]	
77E8065C	8B40 20	MOV EAX,DWORD PTR DS:[EAX+20]	
77E8065F	C3	RETN	

_Ham API will return PID Protected by file, but we will patch it so that it returns the value of the PID Olly. One patch as follows:

Address	Hex dump	Disassembly	Comment
77E80656	B8 FC060000	MOV EAX,6FC	
77E8065B	90	NOP	
77E8065C	90	NOP	
77E8065D	90	NOP	
77E8065E	90	NOP	
77E8065F	C3	RETN	

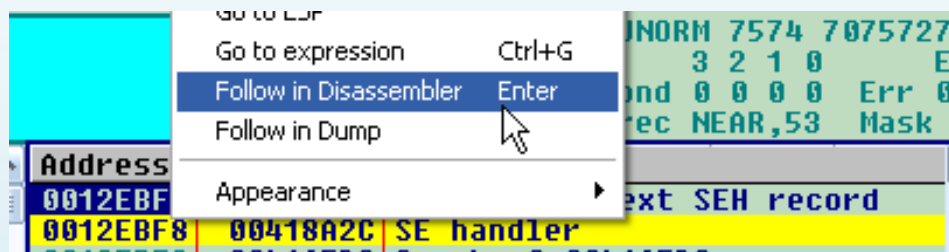
_Oki Is so complete, we now press **F9** to Run, Olly Break will happen again because Exception is a good sign for us:

Address	Hex dump	Disassembly	Comment
00418C99	0000	ADD BYTE PTR DS:[EAX],AL	
00418C9B	0000	ADD BYTE PTR DS:[EAX],AL	
00418C9D	0000	ADD BYTE PTR DS:[EAX],AL	
00418C9F	0000	ADD BYTE PTR DS:[EAX],AL	
00418CA1	0000	ADD BYTE PTR DS:[EAX],AL	
00418CA3	0000	ADD BYTE PTR DS:[EAX],AL	
00418CA5	0000	ADD BYTE PTR DS:[EAX],AL	
00418CA7	0000	ADD BYTE PTR DS:[EAX],AL	
00418CA9	0000	ADD BYTE PTR DS:[EAX],AL	
00418CAB	0000	ADD BYTE PTR DS:[EAX],AL	
00418CAD	0000	ADD BYTE PTR DS:[EAX],AL	
00418CAF	0000	ADD BYTE PTR DS:[EAX],AL	
00418CB1	0000	ADD BYTE PTR DS:[EAX],AL	
00418CB3	0000	ADD BYTE PTR DS:[EAX],AL	
00418CB5	0000	ADD BYTE PTR DS:[EAX],AL	
00418CB7	0000	ADD BYTE PTR DS:[EAX],AL	
00418CB9	0000	ADD BYTE PTR DS:[EAX],AL	
00418CBB	0000	ADD BYTE PTR DS:[EAX],AL	
00418CBD	0000	ADD BYTE PTR DS:[EAX],AL	
00418CBF	0000	ADD BYTE PTR DS:[EAX],AL	
00418CC1	0000	ADD BYTE PTR DS:[EAX],AL	
00418CC3	0000	ADD BYTE PTR DS:[EAX],AL	

_Quan In the **Stack Window** window, we see the following:

Address	Value	Comment
0012EBF4	0012EC84	Pointer to next SEH record
0012EBF8	00418A2C	SE handler
0012EBFC	00416FD2	Crackme0.00416FD2
0012EC00	00416FBF	Crackme0.00416FBF
0012EC04	00416FAC	Crackme0.00416FAC
0012EC08	00416F9F	Crackme0.00416F9F
0012EC0C	00416EF6	Crackme0.00416EF6
0012EC10	00416DD0	Crackme0.00416DD0
0012EC14	00416B99	Crackme0.00416B99
0012EC18	00416B21	Crackme0.00416B21
0012EC1C	00416986	Crackme0.00416986
0012EC20	00416947	Crackme0.00416947
0012EC24	00416934	Crackme0.00416934
0012EC28	00416927	Crackme0.00416927
0012EC2C	00416000	Crackme0.00416000

_Chuot Must address by yoda's exception handler, select **Follow in Disassembler** (Enter):



_Ta Be here in Olly:

Address	Hex dump	Disassembly	Comment
00418A2C	55	PUSH EBP	
00418A2D	8BEC	MOV EBP,ESP	
00418A2F	57	PUSH EDI	
00418A30	36:8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]	
00418A34	3E:8BB8 C4000000	MOV EDI,DWORD PTR DS:[EAX+C4]	
00418A3B	3E:FF37	PUSH DWORD PTR DS:[EDI]	
00418A3E	33FF	XOR EDI,EDI	
00418A40	64:8F07	POP DWORD PTR FS:[EDI]	
00418A43	3E:8380 C4000000	ADD DWORD PTR DS:[EAX+C4],8	
00418A4B	3E:8BB8 A4000000	MOV EDI,DWORD PTR DS:[EAX+A4]	
00418A52	C1C7 07	ROL EDI,7	
00418A55	3E:89B8 B8000000	MOV DWORD PTR DS:[EAX+B8],EDI	
00418A5C	B8 00000000	MOV EAX,0	
00418A61	5F	POP EDI	
00418A62	C9	LEAVE	
00418A63	C3	RETN	

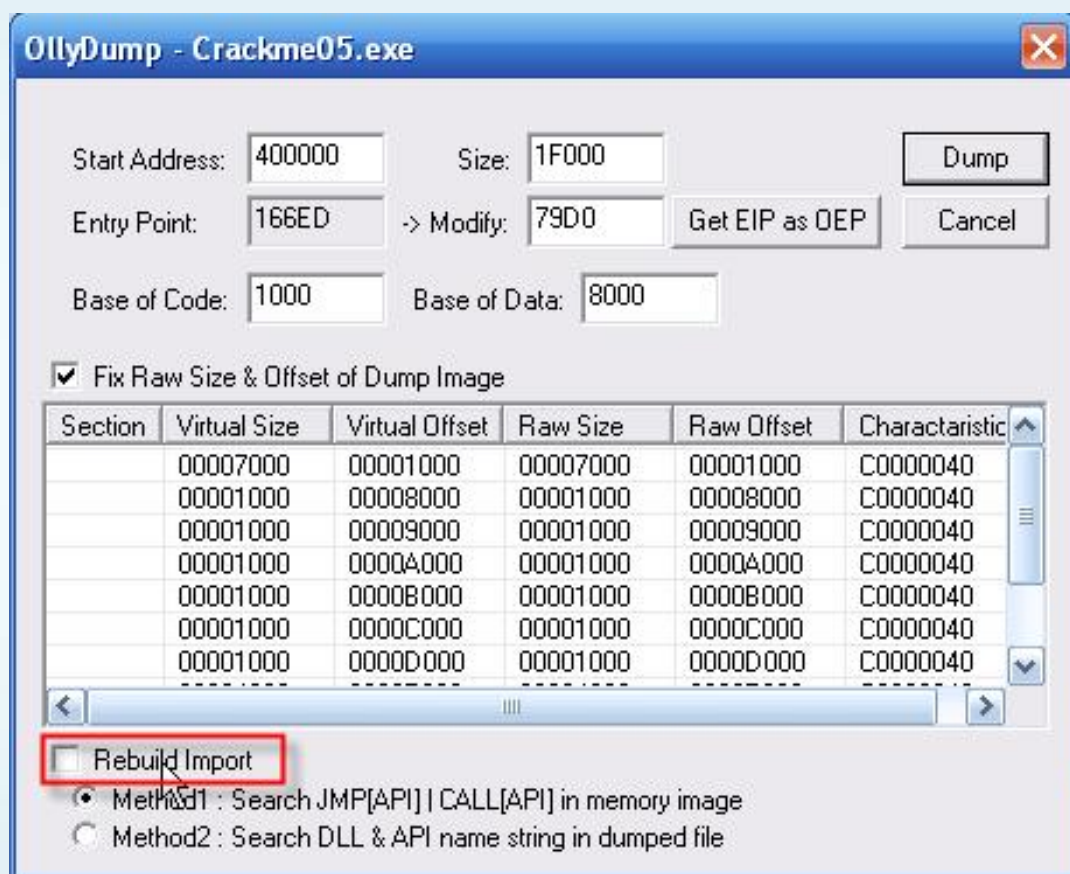
_Tai This press **F2** to set BP, then press **Shift + F9** to bypass Exception, we will address Break **00418A2C**. Un BP we've set out, then press **F8** to trace to address **00418A55**. Observations Registers the window we are seeing record **EDI** are stored address our OEP:

Address	Hex dump	Disassembly	Comment	Registers (FPU)
00418A2C	55	PUSH EBP		EAX 0012E928
00418A2D	8BEC	MOV EBP,ESP		ECX 00418A2C Crackne0.00418A2C
00418A2F	57	PUSH EDI		EDX 77F833B4 ntdll.77F833B4
00418A30	36:8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]		EBX 00000000
00418A34	3E:8BB8 C4000000	MOV EDI,DWORD PTR DS:[EAX+C4]		ESP 0012E820
00418A3B	3E:FF37	PUSH DWORD PTR DS:[EDI]		EBP 0012E824
00418A3E	33FF	XOR EDI,EDI		ESI 00000000
00418A40	64:8F07	POP DWORD PTR FS:[EDI]		EDI 004079D0 Crackne0.004079D0
00418A43	3E:8380 C4000000	ADD DWORD PTR DS:[EAX+C4],8		EIP 00418A55 Crackne0.00418A55
00418A4B	3E:8BB8 A4000000	MOV EDI,DWORD PTR DS:[EAX+A4]		C 0 ES 0023 32bit 0(FFFFFFFF)
00418A52	C1C7 07	ROL EDI,7		P 1 CS 0018 32bit 0(FFFFFFFF)
00418A55	3E:89B8 B8000000	MOV DWORD PTR DS:[EAX+B8],EDI	Crackne0.004079D0	

_Chuot Must write in to choose **Follow in EDI Disassembler**, then set at 1 BP OEP of our last press **Shift + F9** we OEP at Break:

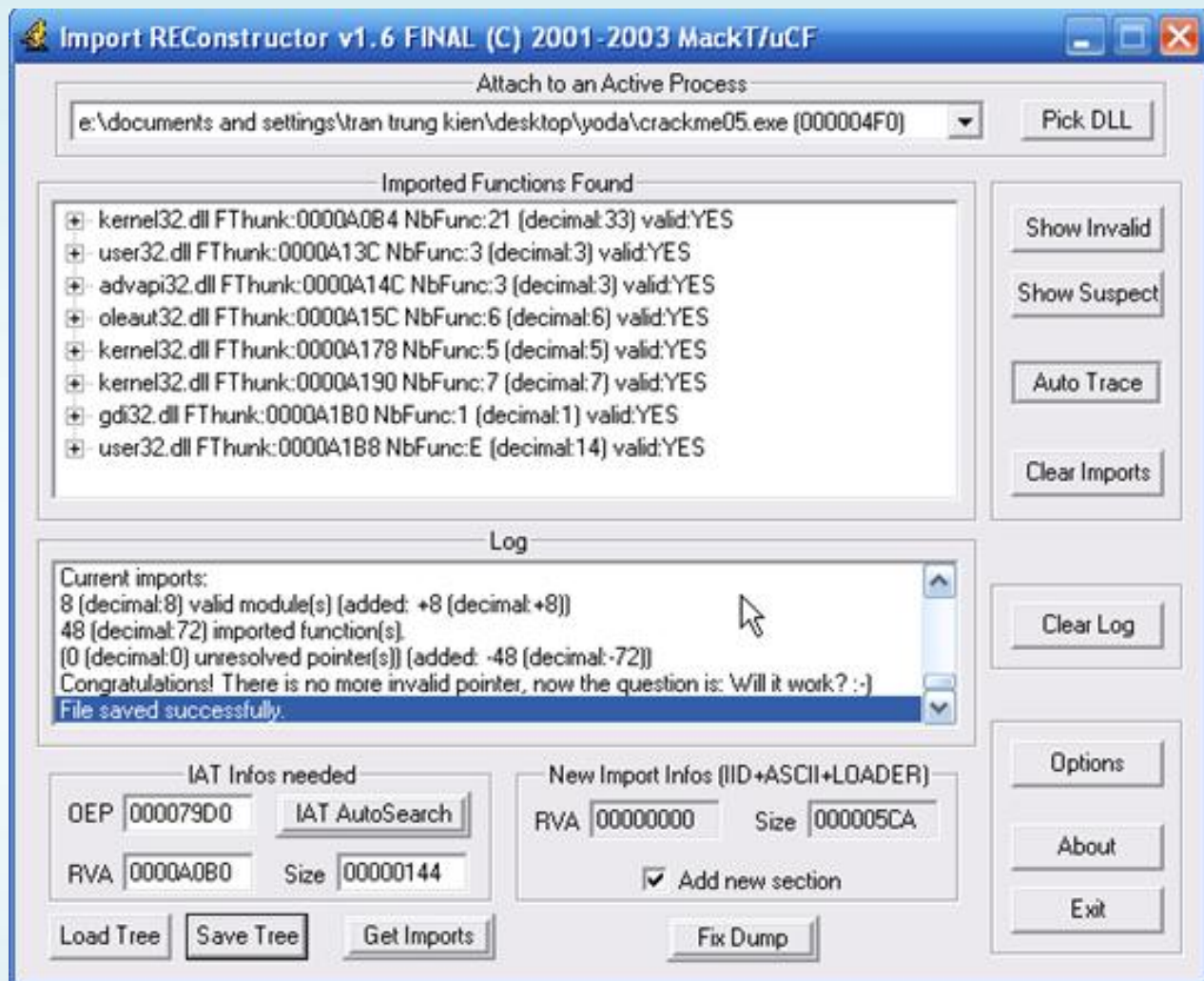
Address	Hex dump	Disassembly	Comment
004079D0	55	PUSH EBP	Crackme0.00417058
004079D1	8BEC	MOV EBP,ESP	
004079D3	83C4 F4	ADD ESP,-0C	
004079D6	53	PUSH EBX	
004079D7	56	PUSH ESI	
004079D8	57	PUSH EDI	
004079D9	B8 98794000	MOV EAX,Crackme0.00407998	
004079DE	E8 D1C8FFFF	CALL Crackme0.004045B4	
004079E3	BE CC954000	MOV ESI,Crackme0.004095CC	
004079E8	BF E4954000	MOV EDI,Crackme0.004095E4	
004079ED	BB A0954000	MOV EBX,Crackme0.004095A0	
004079F2	33C0	XOR EAX,EAX	
004079F4	55	PUSH EBP	
004079F5	68 847C4000	PUSH Crackme0.00407C84	
004079FA	64:FF30	PUSH DWORD PTR FS:[EAX]	
004079FD	64:8920	MOV DWORD PTR FS:[EAX],ESP	
00407A00	A1 80824000	MOV EAX,DWORD PTR DS:[408280]	
00407A05	8B00	MOV EAX,DWORD PTR DS:[EAX]	
00407A07	A3 C8954000	MOV DWORD PTR DS:[4095C8],EAX	
00407A0C	C703 C0000000	MOV DWORD PTR DS:[EBX],0C0	
00407A12	C743 04 90784000	MOV DWORD PTR DS:[EBX+4],Crackme0.00	
00407A19	A1 C8954000	MOV EAX,DWORD PTR DS:[4095C8]	
00407A1E	8943 10	MOV DWORD PTR DS:[EBX+10],EAX	
00407A21	C743 1C 10000000	MOV DWORD PTR DS:[EBX+1C],10	
00407A28	B8 947C4000	MOV EAX,Crackme0.00407C94	ASCII "GLAVNI PROZOR"
00407A2D	8943 24	MOV DWORD PTR DS:[EBX+24],EAX	
00407A30	68 007F0000	PUSH 7F00	

_Bay Hour final stage is a dump file and Fix IAT. You can use **LordPE** to dump file, but the target was to Protect by Yoda, the dump with LordPE or lose the icon. So with this case I will use the Plugin Ollydump to dump file:

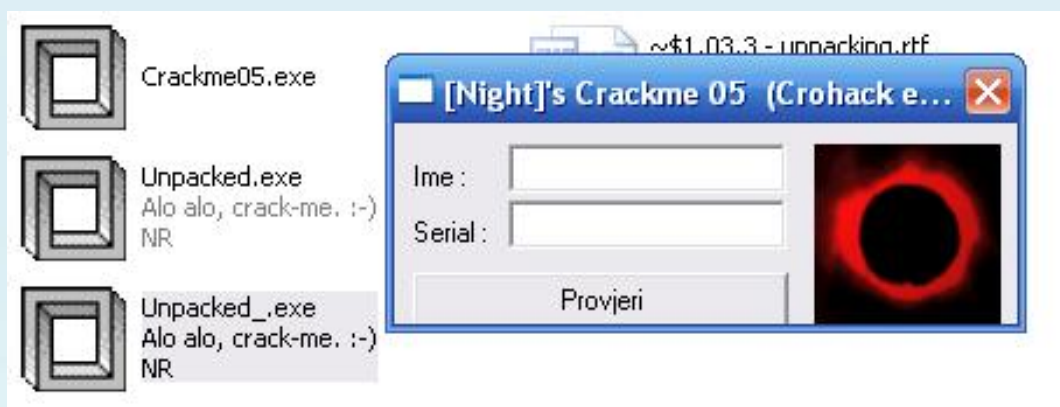


_Tiep By **ImportRec** open up and load the correct process is our target. Fill in the OEP,

click **IAT AutoSearch** and finally click **Get Imports**. Khuc we receive an Invalid Thunk but do not, you click Show Invalid then right click on Invalid thunk and **Trace Level1 (Disasm)**. We are as follows:



Fix dump _Hehe stop, then try the test file was unpacked. Wow! Run then kìa tolerable:



_Hen Meet you again in the following message! J

Best Regards

[Kienmanowar]

--+ +---==[**Greatz thanks to**]=---+ +--

My family, Computer_Angel, Moonbaby, Zombie_Deathman, Littleboy, Benina, QHOCrker, the_Lighthouse, Merc, Hoadongnoi, Nini ... all REA's members, TQN, HacNho, RongChauA, Deux, tlandn, light.phoenix, dqtlN, ARTEAM all my friend, and you.

Thanks to --+ +---==[]=---+ +--

iamidiot, WhyNotBar, trickyboy, dzungltvn, takada, hurt_heart, haule_nth, v. hytkl. v.. You have contributed greatly to the REA. Hope you will continue to promote J

>>> If you have any suggestions, comments or corrections email me: **kienbigmummy [at] gmail.com**