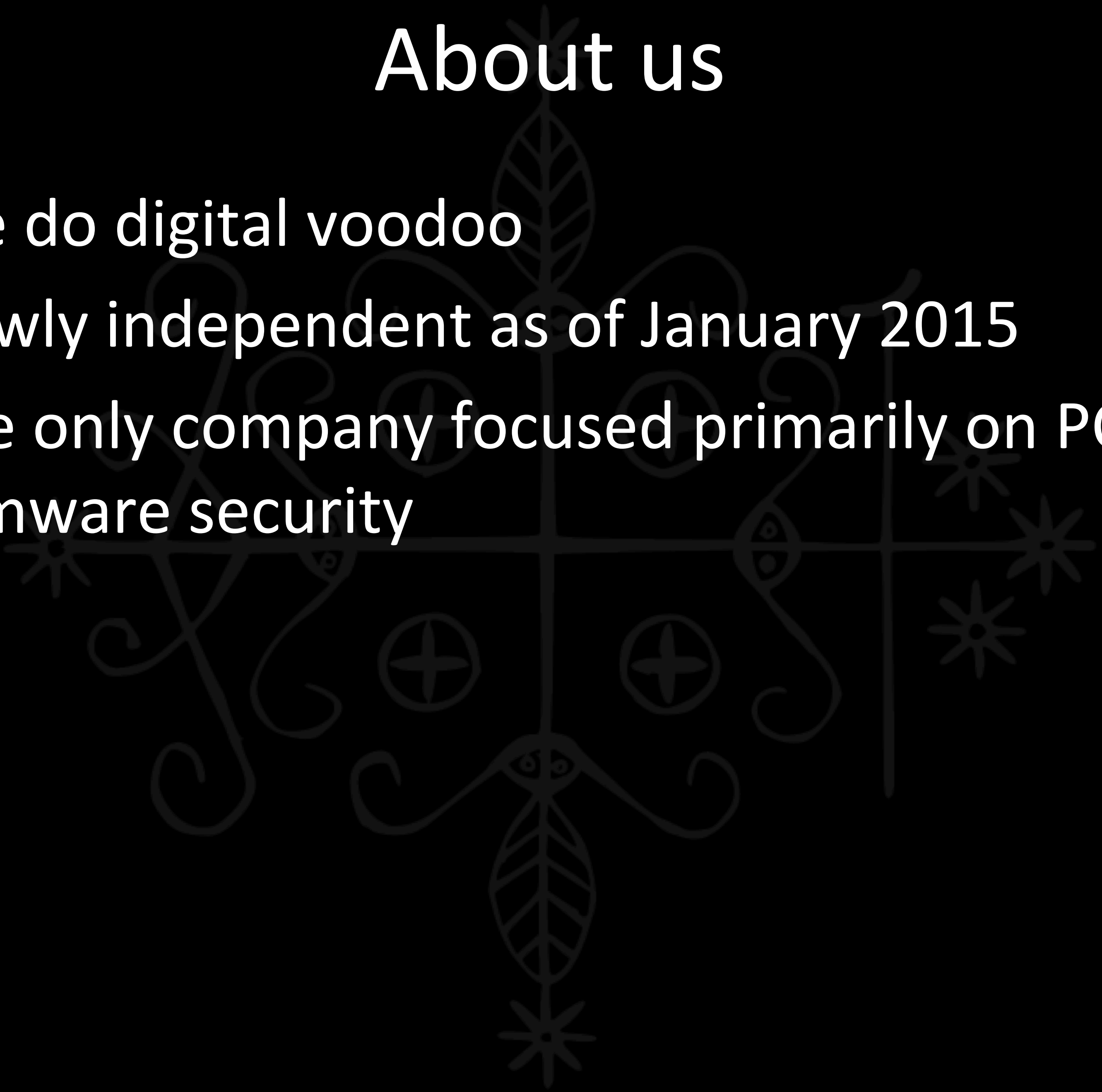
LEGBACORE

How Many Million BIOSes Would you Like to Infect?

Corey Kallenberg & Xeno Kovah

About us



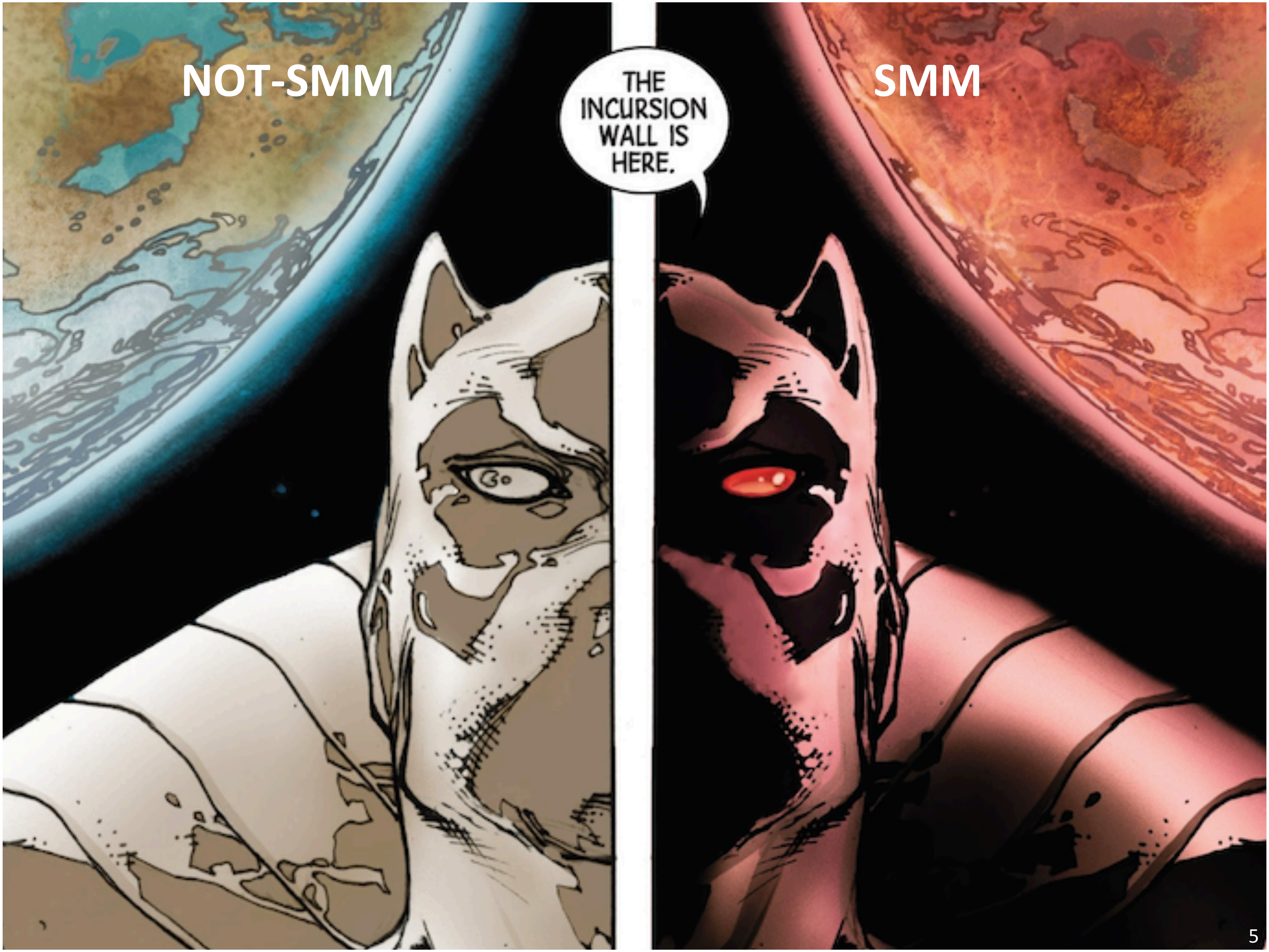
- We do digital voodoo
- Newly independent as of January 2015
- The only company focused primarily on PC firmware security

This talk has 2 main points

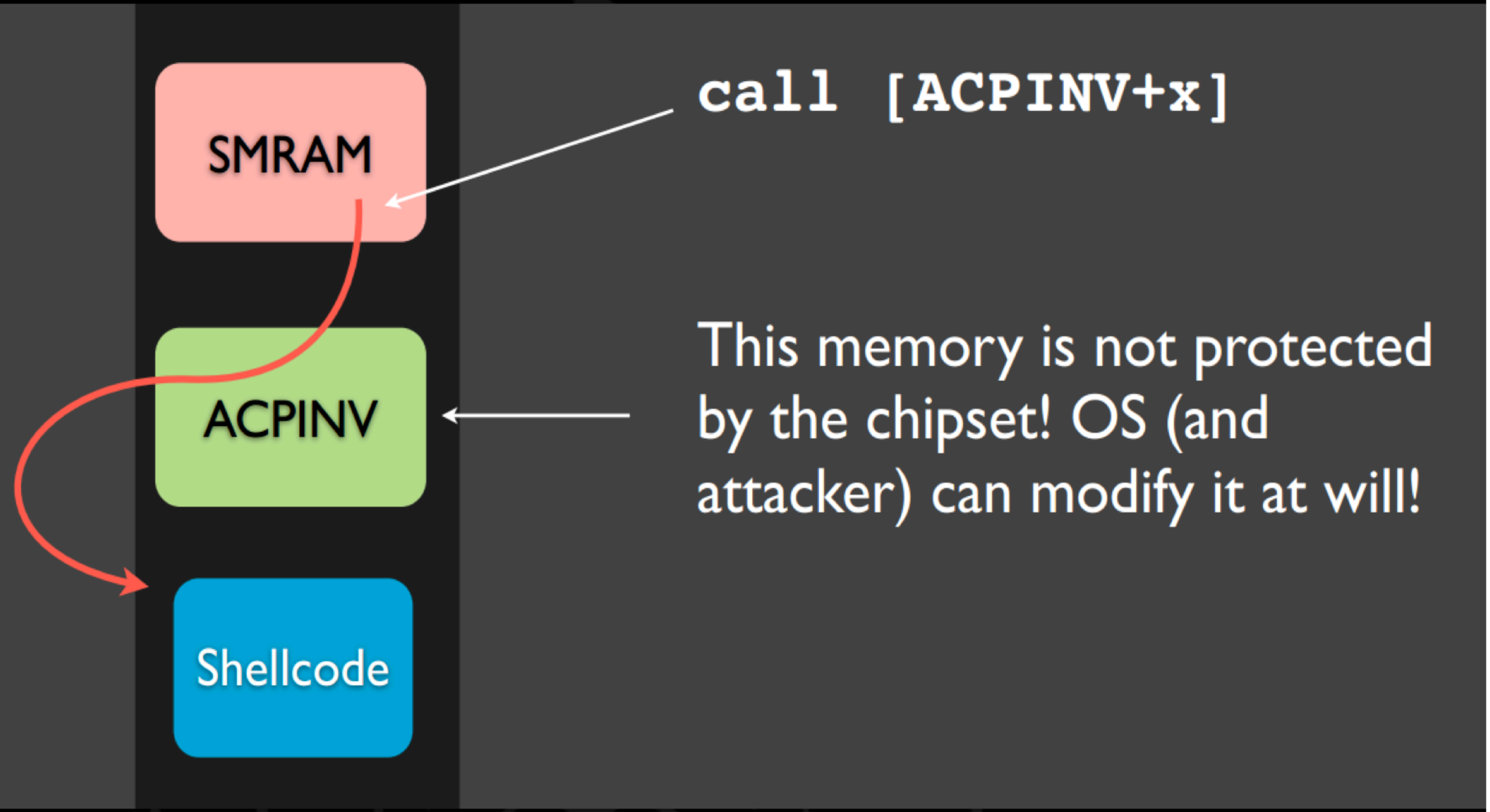
- Because almost no one applies BIOS patches, almost every BIOS in the wild is affected by *at least* one vulnerability, and can be infected
- The high amount of code reuse across UEFI BIOSes means that BIOS infection is automatable and reliable

What’s past is prologue

- Some (mostly-multi-vendor) BIOS vulnerabilities disclosed since 2012:
- CERT VU#127284[0], 912156[1](“Ruy Lopez”), 255726[1](“The Sicilian”), 758382[2] (“Setup bug”), 291102[4] (“Charizard”), 552286[5] (“King & Queen’s Gambit”), 533140[6] (“noname”), 766164[7] (“Speed Racer”), 976132[8] (“Venamis”), 577140[9](“Snorlax”)
- And a bunch from others that didn’t get VU#s


NOT-SMM
THE INCURSION WALL IS HERE.
SMM

Incursions (VU#631788)



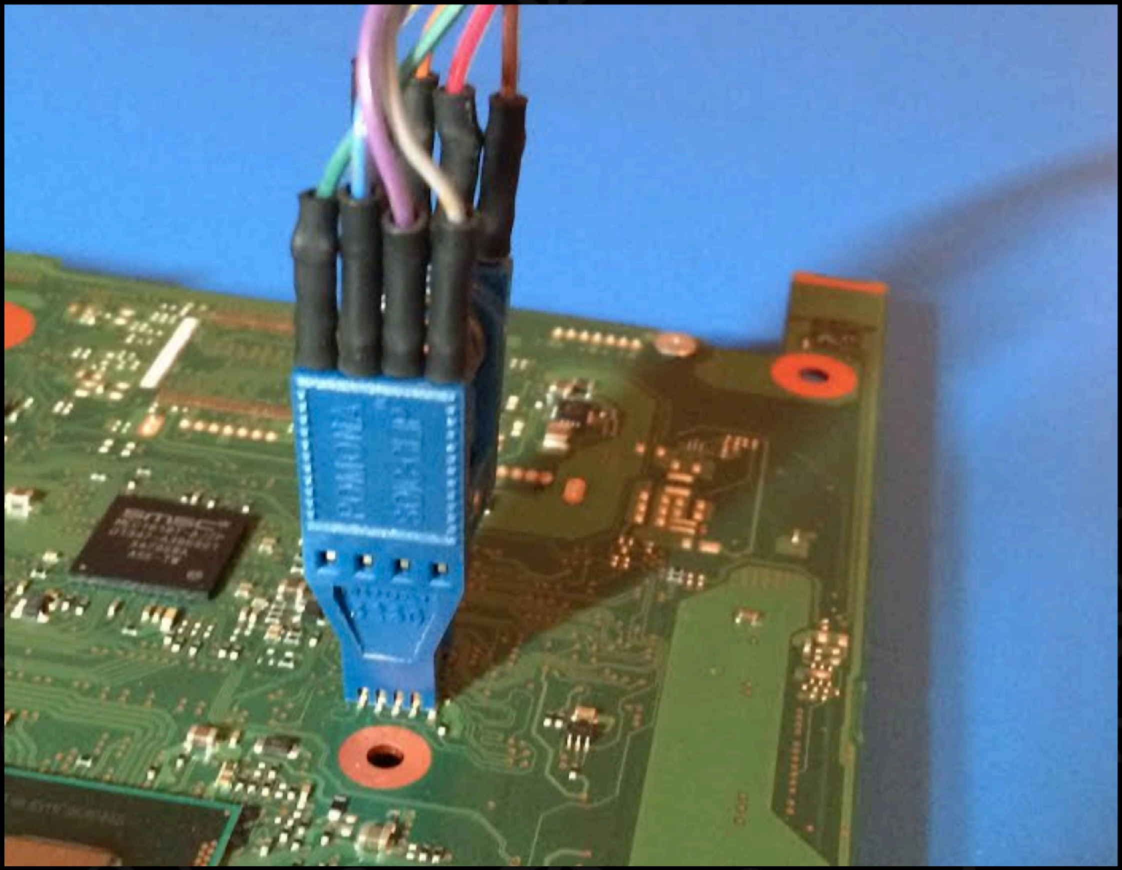
`call [ACPINV+x]`

This memory is not protected by the chipset! OS (and attacker) can modify it at will!

- In 2008 ITL disclosed an SMM vulnerability where on some Intel motherboards SMM code called through non-SMRAM function pointer table
  - Low hanging fruit SMM vulnerability!
- How prevalent are low hanging fruit SMM vulnerabilities today?

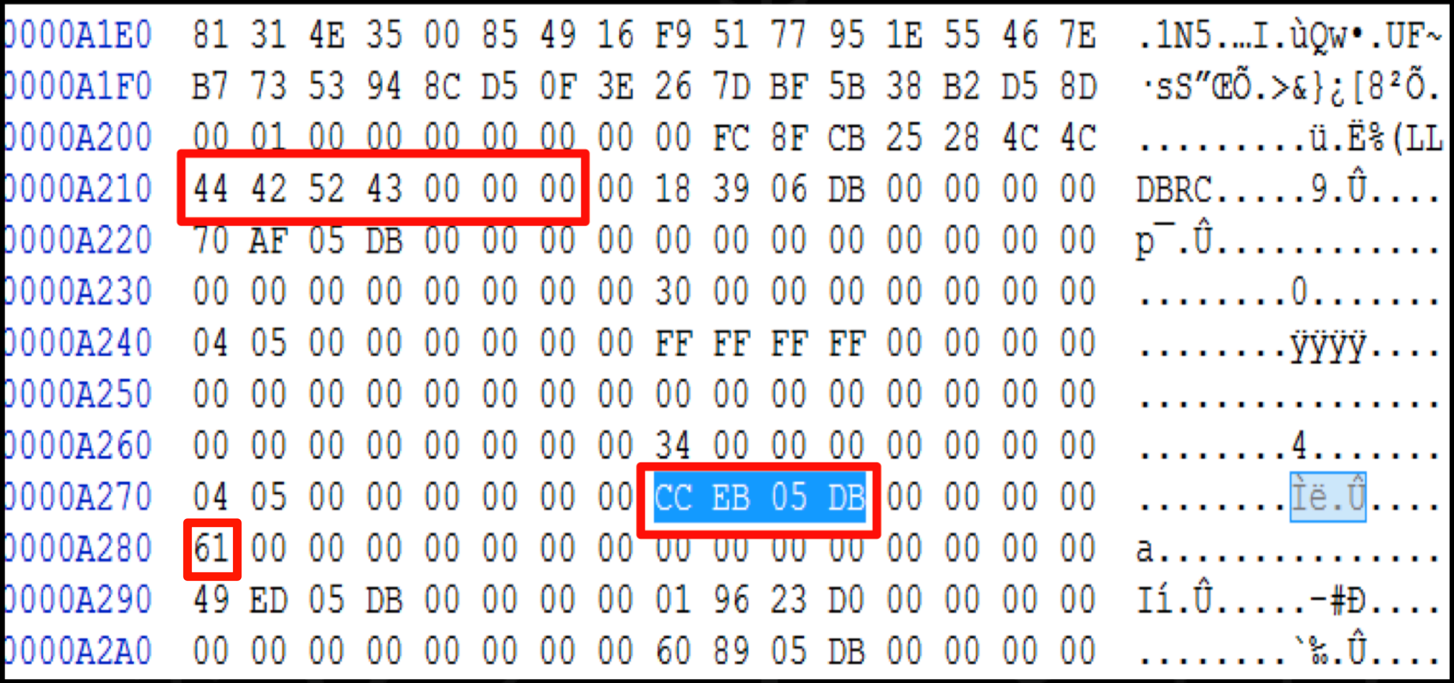
http://invisiblethingslab.com/resources/bh09usa/Attacking%20Intel%20BIOS.pdf

- But how do you hit what you cannot see?



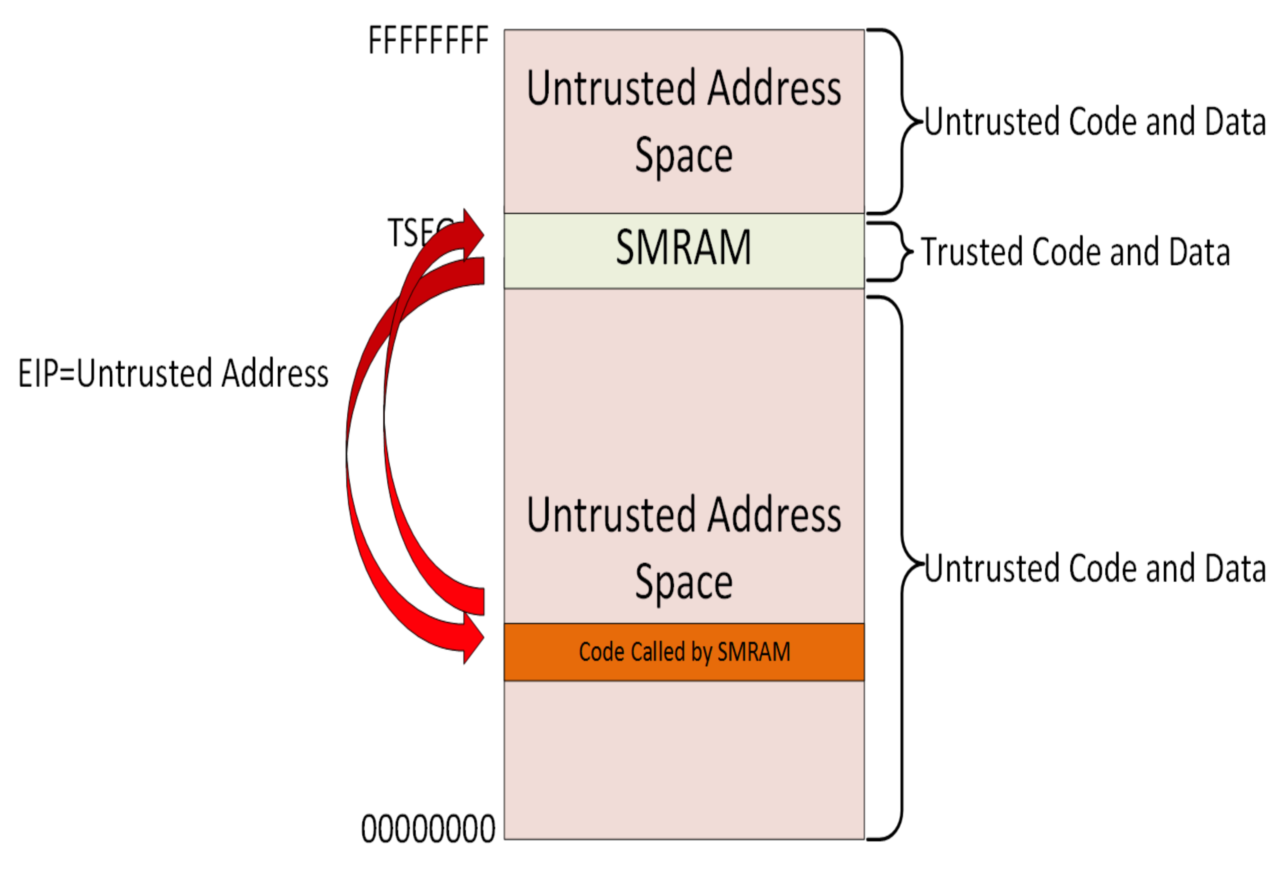
- Option 1: Reprogram firmware to disable SMRAM protection
  - Disable TSEG
  - Disable SMRRs

- Option 2: Use the power of the dark side



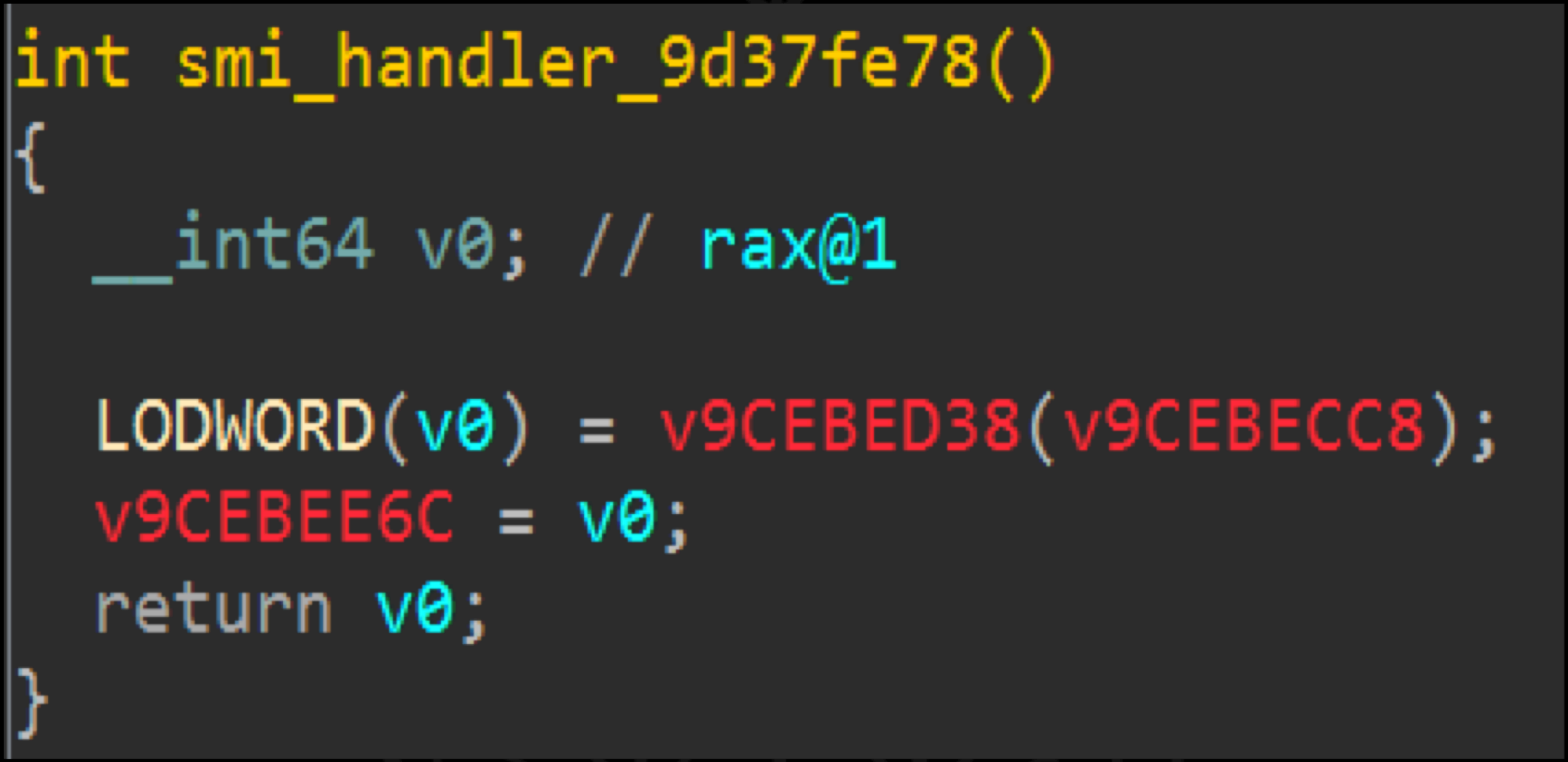
```
0000A1E0  81 31 4E 35 00 85 49 16 F9 51 77 95 1E 55 46 7E  .1N5.…I.ùQw•.UF~
0000A1F0  B7 73 53 94 8C D5 0F 3E 26 7D BF 5B 38 B2 D5 8D  ·sS”ŒÕ.>&}¿[8²Õ.
0000A200  00 01 00 00 00 00 00 00 00 FC 8F CB 25 28 4C 4C  .........ü.Ë%(LL
0000A210  44 42 52 43 00 00 00 00 18 39 06 DB 00 00 00 00  DBRC.....9.Û....
0000A220  70 AF 05 DB 00 00 00 00 00 00 00 00 00 00 00 00  p¯.Û............
0000A230  00 00 00 00 00 00 00 00 30 00 00 00 00 00 00 00  ........0.......
0000A240  04 05 00 00 00 00 00 00 FF FF FF FF 00 00 00 00  ........ÿÿÿÿ....
0000A250  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0000A260  00 00 00 00 00 00 00 00 34 00 00 00 00 00 00 00  ........4.......
0000A270  04 05 00 00 00 00 00 00 CC EB 05 DB 00 00 00 00  ........Ìë.Û....
0000A280  61 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  a...............
0000A290  49 ED 05 DB 00 00 00 00 01 96 23 D0 00 00 00 00  Ií.Û.....–#Ð....
0000A2A0  00 00 00 00 00 00 00 00 60 89 05 DB 00 00 00 00  ........`‰.Û....
```

- We did a little RE work to determine which SMM code we could invoke from the OS by writing to port 0xB2
- In this case, function 0xDB05EDCC within SMM can be reached by writing 0x61 to port 0xB2
- Almost every UEFI system we surveyed used this format to record reachable SMM code

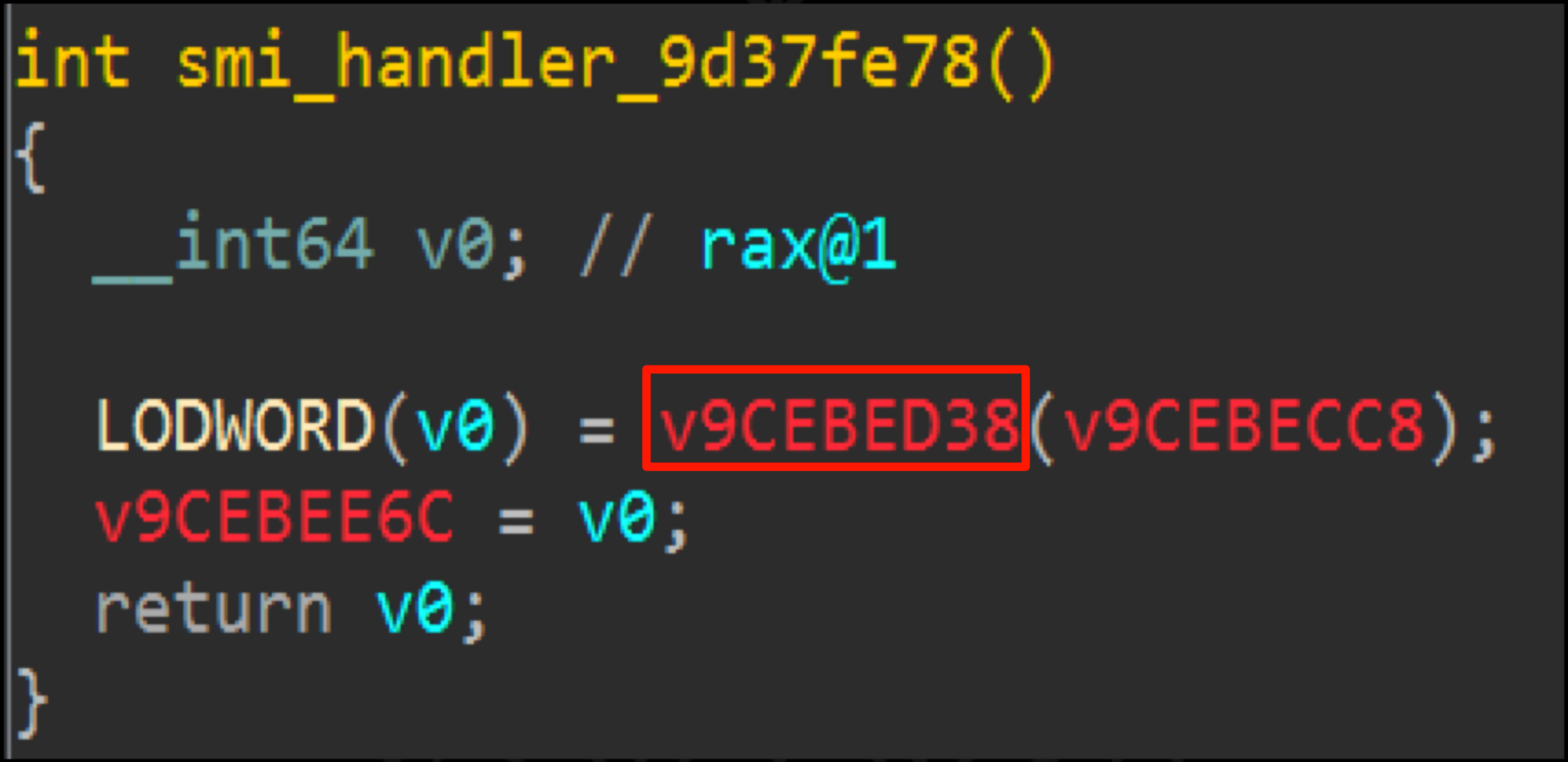


- We found a lot of these vulnerabilities
- They were so easy to find, we could write a ~300 line IDAPython script that found so many I stopped counting and (some) vendors stopped emailing me back

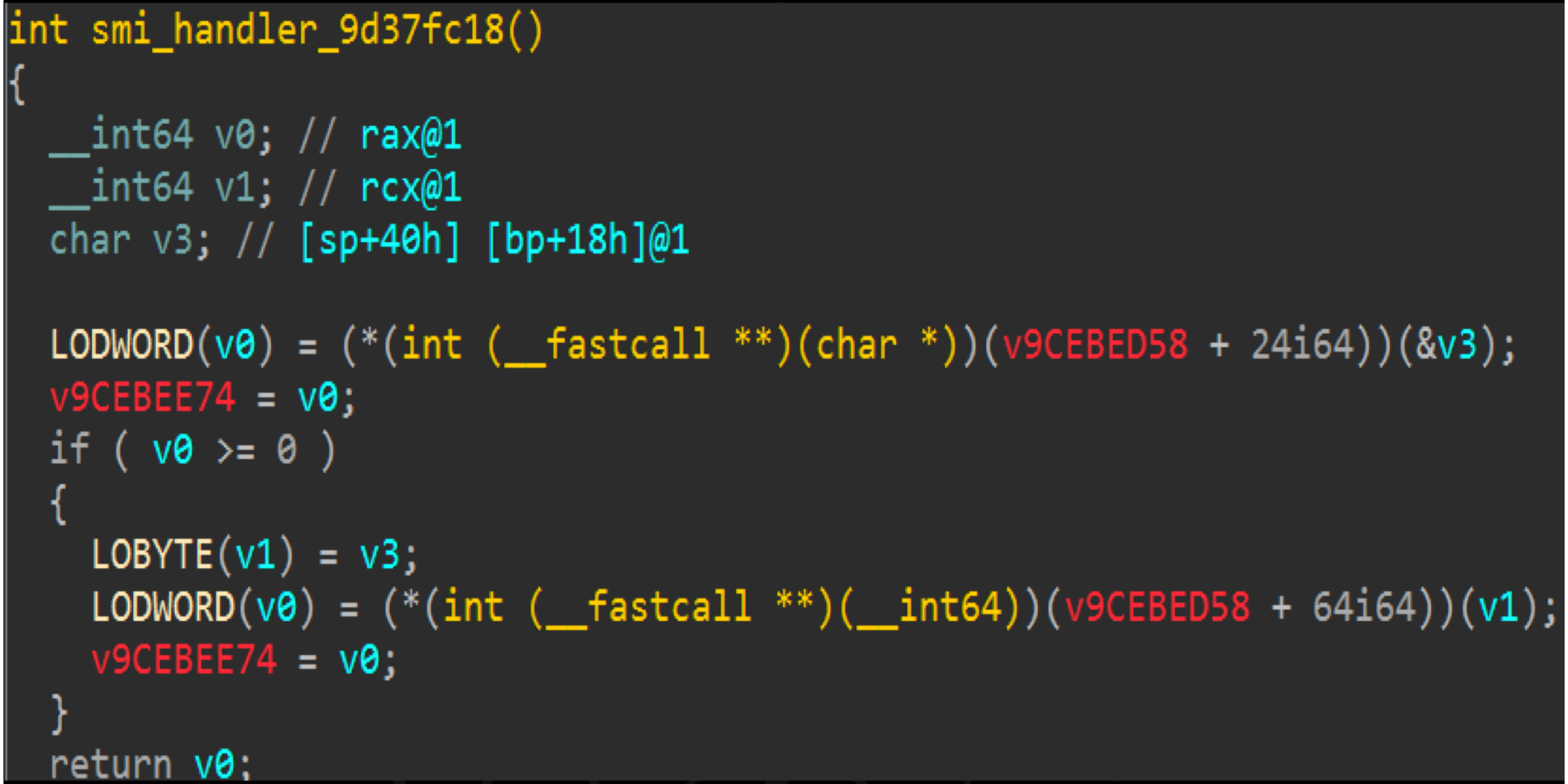
- You’re the next contestant on… Is it vulnerable???
- Hint: Hexrays detects the external memory accesses and colors them red.
  - When you see red, bad!



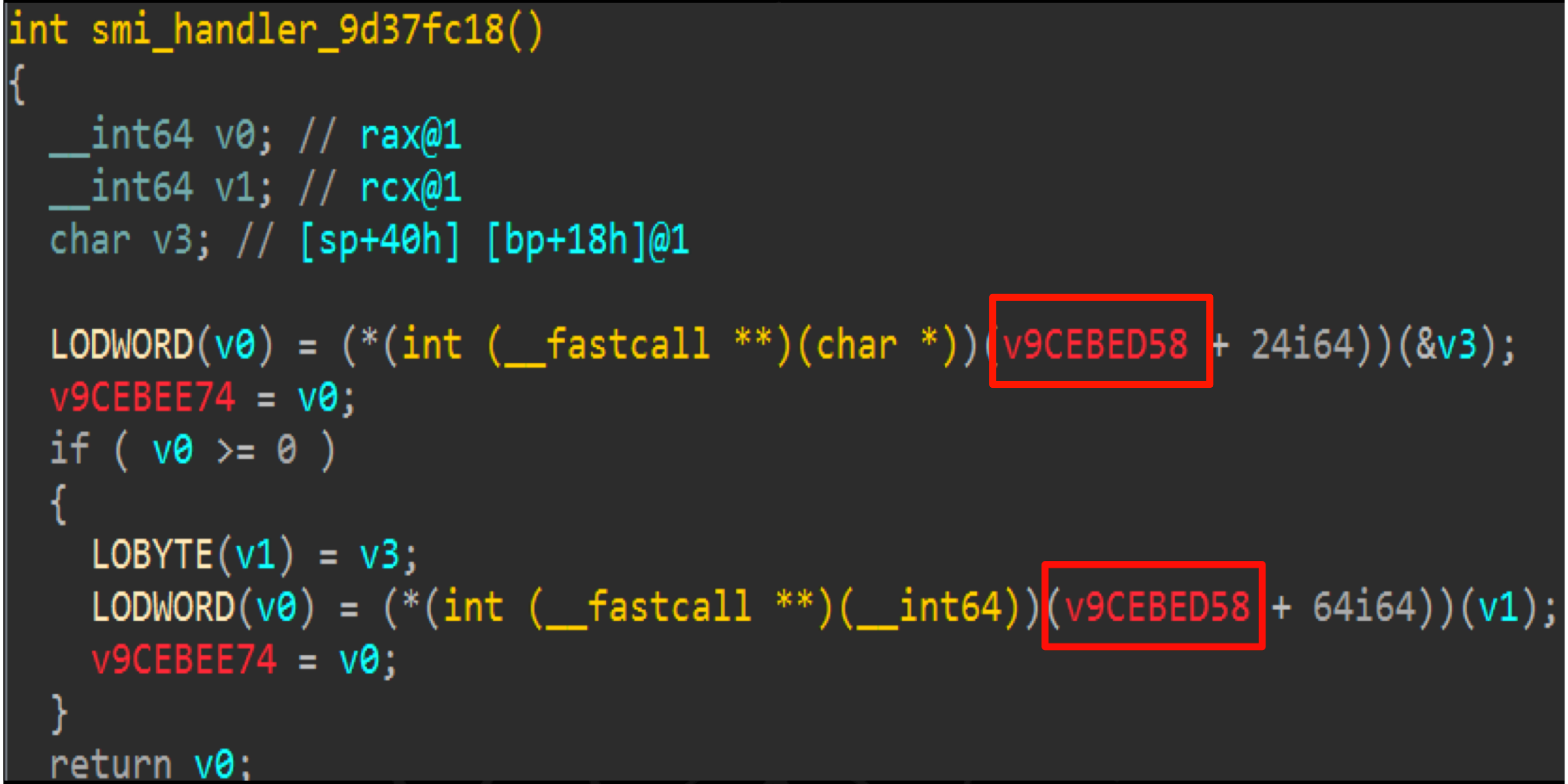
```
int smi_handler_9d37fe78()
{
  __int64 v0; // rax@1

  LODWORD(v0) = v9CEBED38(v9CEBECC8);
  v9CEBEE6C = v0;
  return v0;
}
```



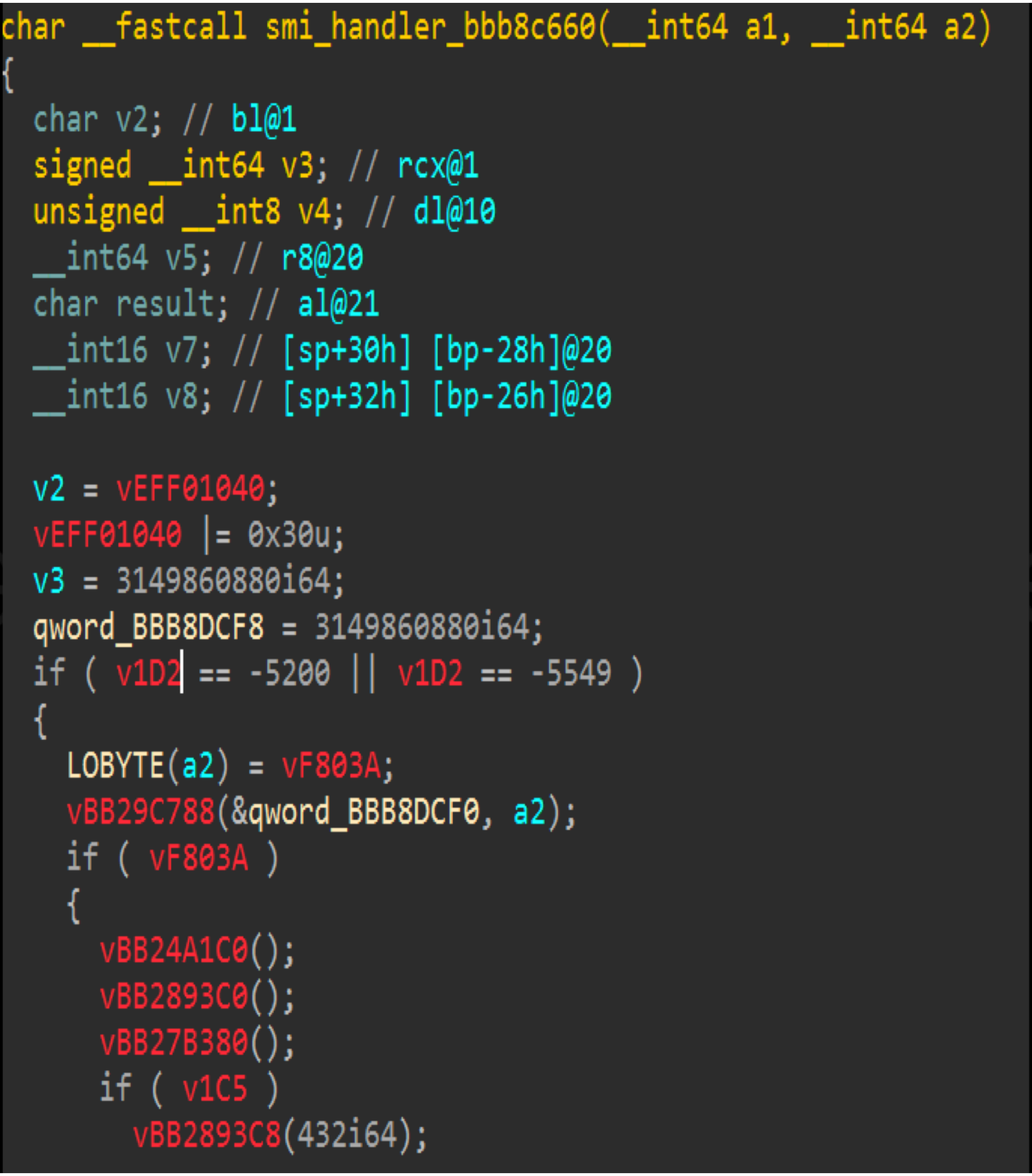
```
int smi_handler_9d37fe78()
{
  __int64 v0; // rax@1

  LODWORD(v0) = v9CEBED38(v9CEBECC8);
  v9CEBEE6C = v0;
  return v0;
}
```



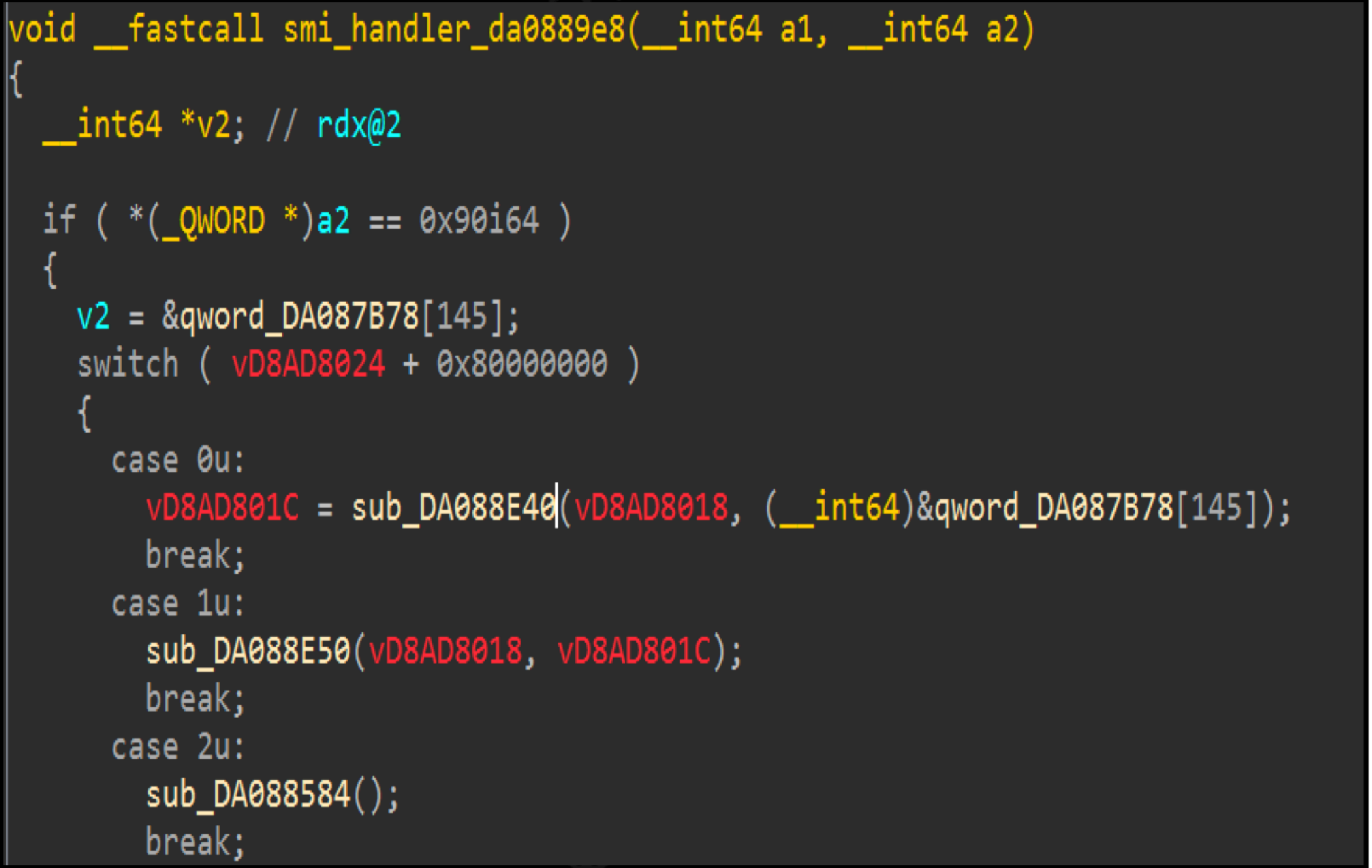
```
int smi_handler_9d37fc18()
{
  __int64 v0; // rax@1
  __int64 v1; // rcx@1
  char v3; // [sp+40h] [bp+18h]@1

  LODWORD(v0) = (*(int (__fastcall **)(char *))(v9CEBED58 + 24i64))(&v3);
  v9CEBEE74 = v0;
  if ( v0 >= 0 )
  {
    LOBYTE(v1) = v3;
    LODWORD(v0) = (*(int (__fastcall **)(__int64))(v9CEBED58 + 64i64))(v1);
    v9CEBEE74 = v0;
  }
  return v0;
```



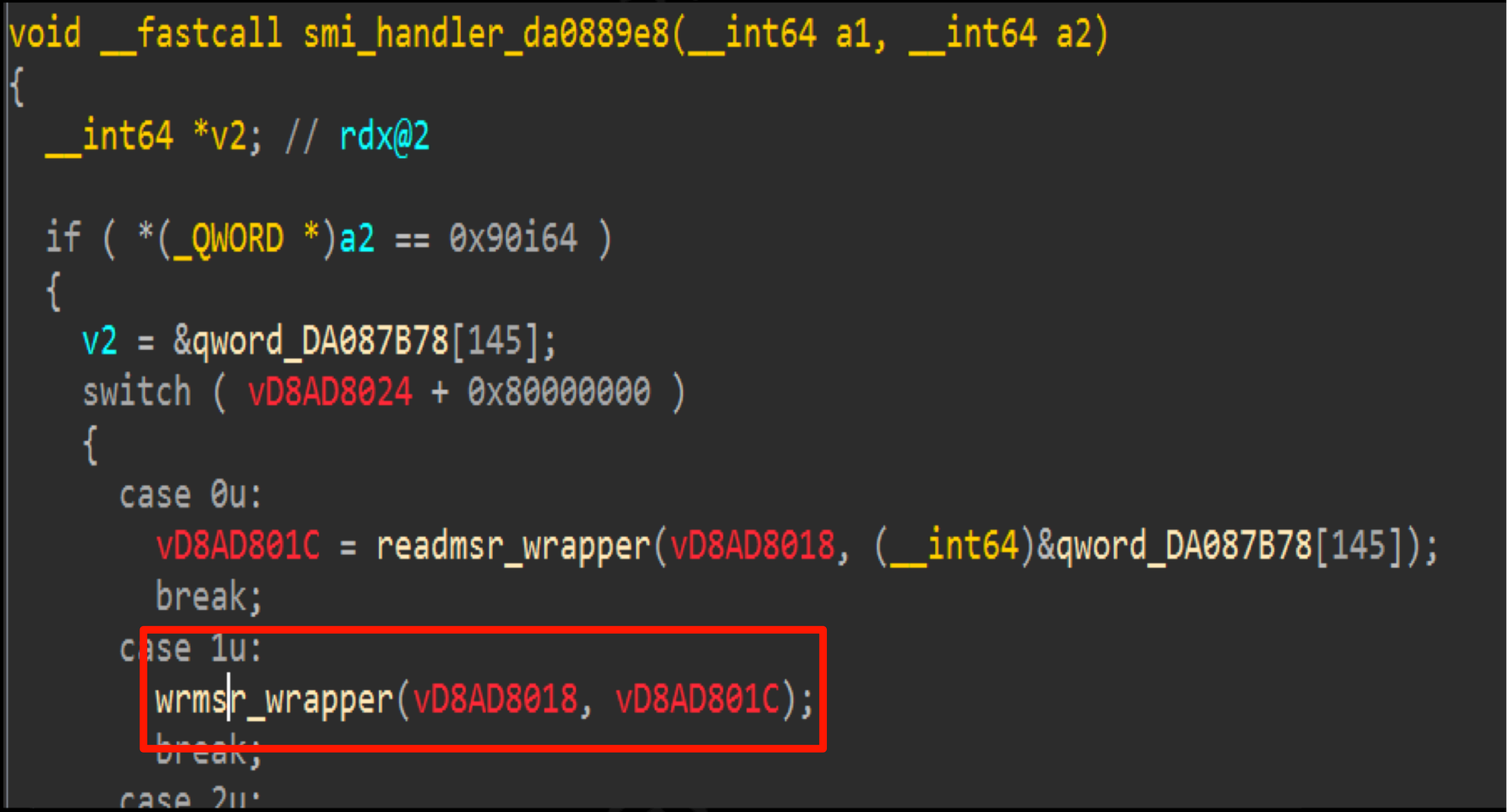
```
int smi_handler_9d37fc18()
{
  __int64 v0; // rax@1
  __int64 v1; // rcx@1
  char v3; // [sp+40h] [bp+18h]@1

  LODWORD(v0) = (*(int (__fastcall **)(char *))(v9CEBED58 + 24i64))(&v3);
  v9CEBEE74 = v0;
  if ( v0 >= 0 )
  {
    LOBYTE(v1) = v3;
    LODWORD(v0) = (*(int (__fastcall **)(__int64))(v9CEBED58 + 64i64))(v1);
    v9CEBEE74 = v0;
  }
  return v0;
```



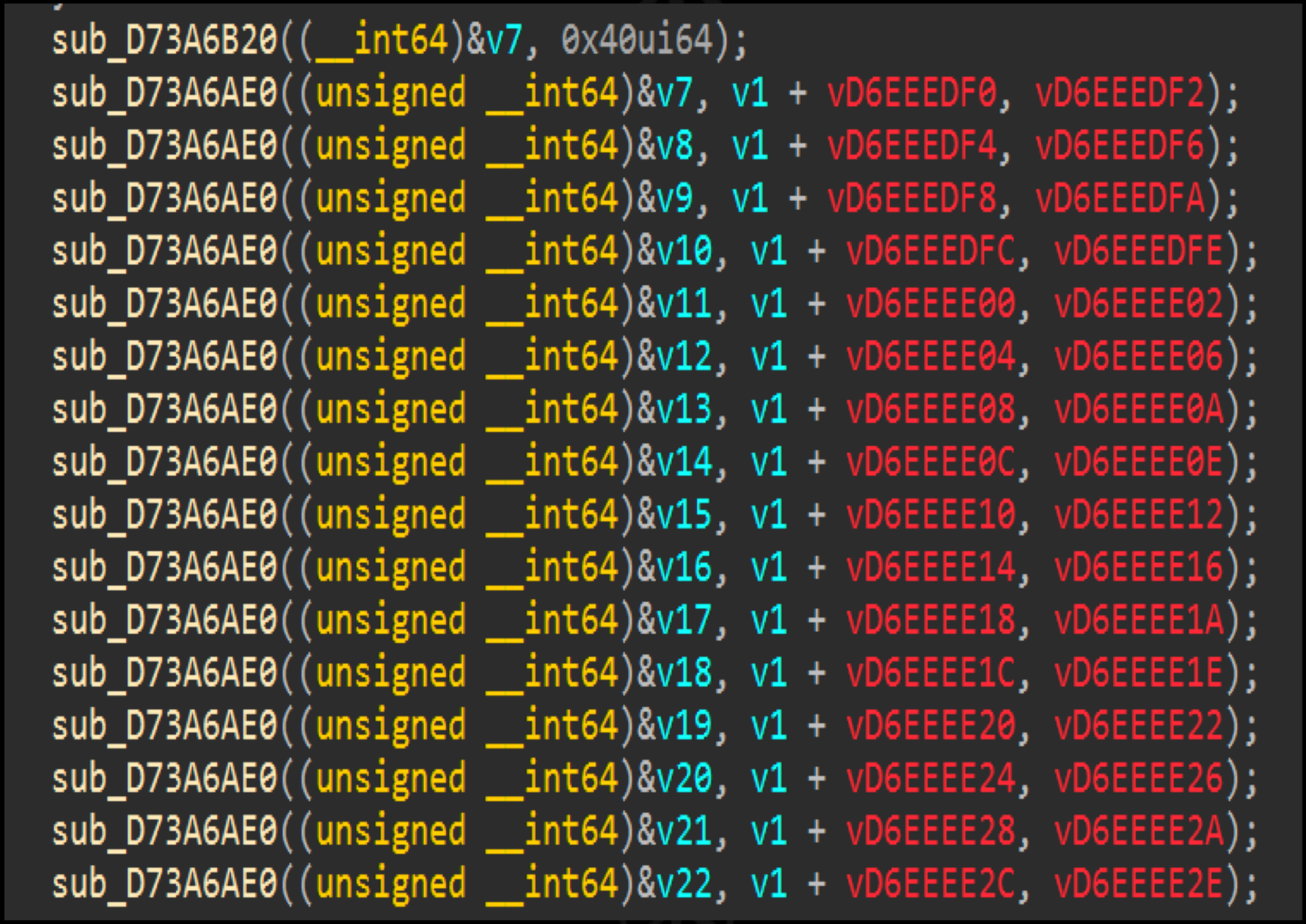
```
char __fastcall smi_handler_bbb8c660(__int64 a1, __int64 a2)
{
  char v2; // bl@1
  signed __int64 v3; // rcx@1
  unsigned __int8 v4; // dl@10
  __int64 v5; // r8@20
  char result; // al@21
  __int16 v7; // [sp+30h] [bp-28h]@20
  __int16 v8; // [sp+32h] [bp-26h]@20

  v2 = vEFF01040;
  vEFF01040 |= 0x30u;
  v3 = 3149860880i64;
  qword_BBB8DCF8 = 3149860880i64;
  if ( v1D2 == -5200 || v1D2 == -5549 )
  {
    LOBYTE(a2) = vF803A;
    vBB29C788(&qword_BBB8DCF0, a2);
    if ( vF803A )
    {
      vBB24A1C0();
      vBB2893C0();
      vBB27B380();
      if ( v1C5 )
        vBB2893C8(432i64);
```



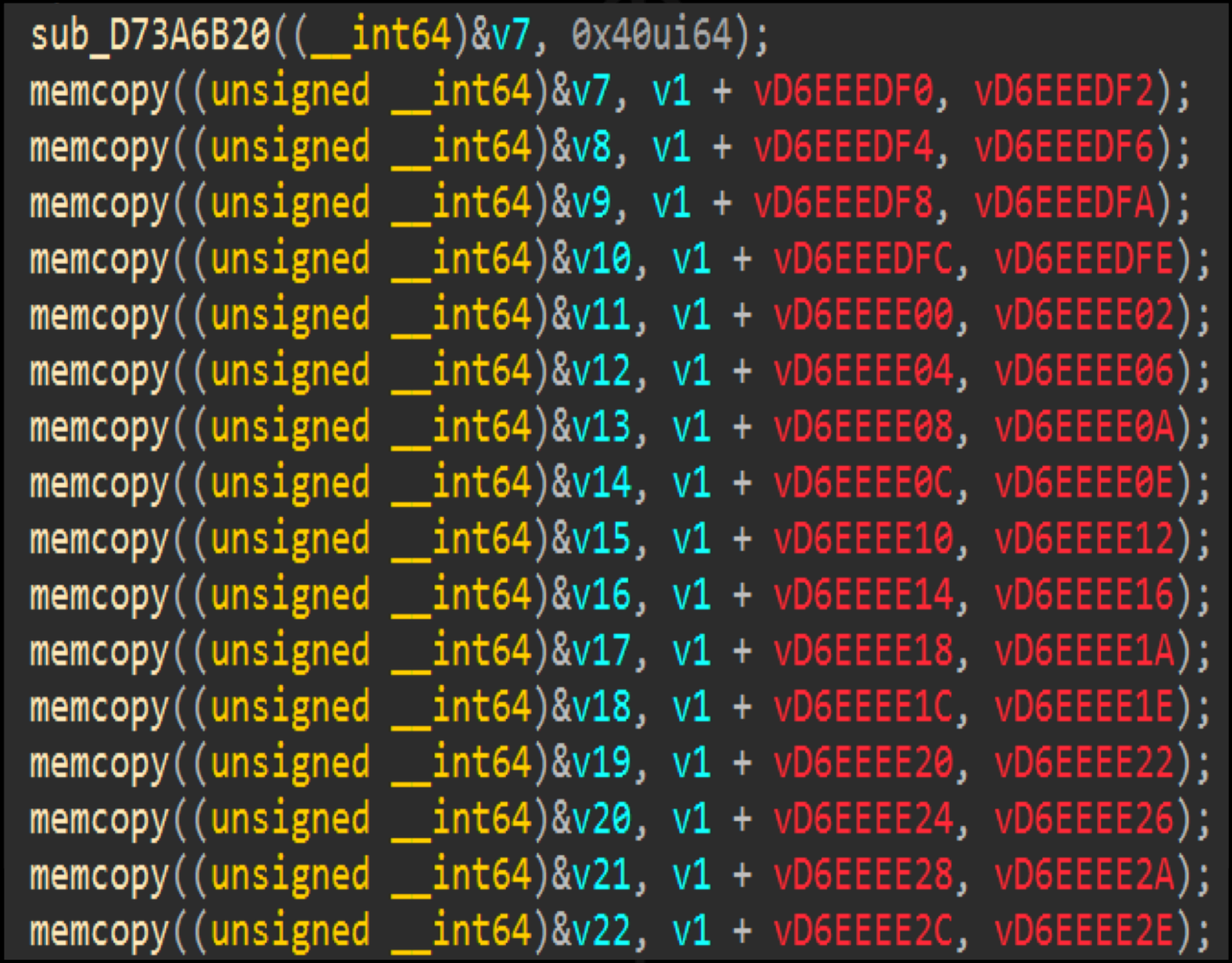
```
void __fastcall smi_handler_da0889e8(__int64 a1, __int64 a2)
{
  __int64 *v2; // rdx@2

  if ( *(_QWORD *)a2 == 0x90i64 )
  {
    v2 = &qword_DA087B78[145];
    switch ( vD8AD8024 + 0x80000000 )
    {
      case 0u:
        vD8AD801C = sub_DA088E40(vD8AD8018, (__int64)&qword_DA087B78[145]);
        break;
      case 1u:
        sub_DA088E50(vD8AD8018, vD8AD801C);
        break;
      case 2u:
        sub_DA088584();
        break;
```



```
void __fastcall smi_handler_da0889e8(__int64 a1, __int64 a2)
{
  __int64 *v2; // rdx@2

  if ( *(_QWORD *)a2 == 0x90i64 )
  {
    v2 = &qword_DA087B78[145];
    switch ( vD8AD8024 + 0x80000000 )
    {
      case 0u:
        vD8AD801C = readmsr_wrapper(vD8AD8018, (__int64)&qword_DA087B78[145]);
        break;
      case 1u:
        wrmsr_wrapper(vD8AD8018, vD8AD801C);
        break;
      case 2u:
```

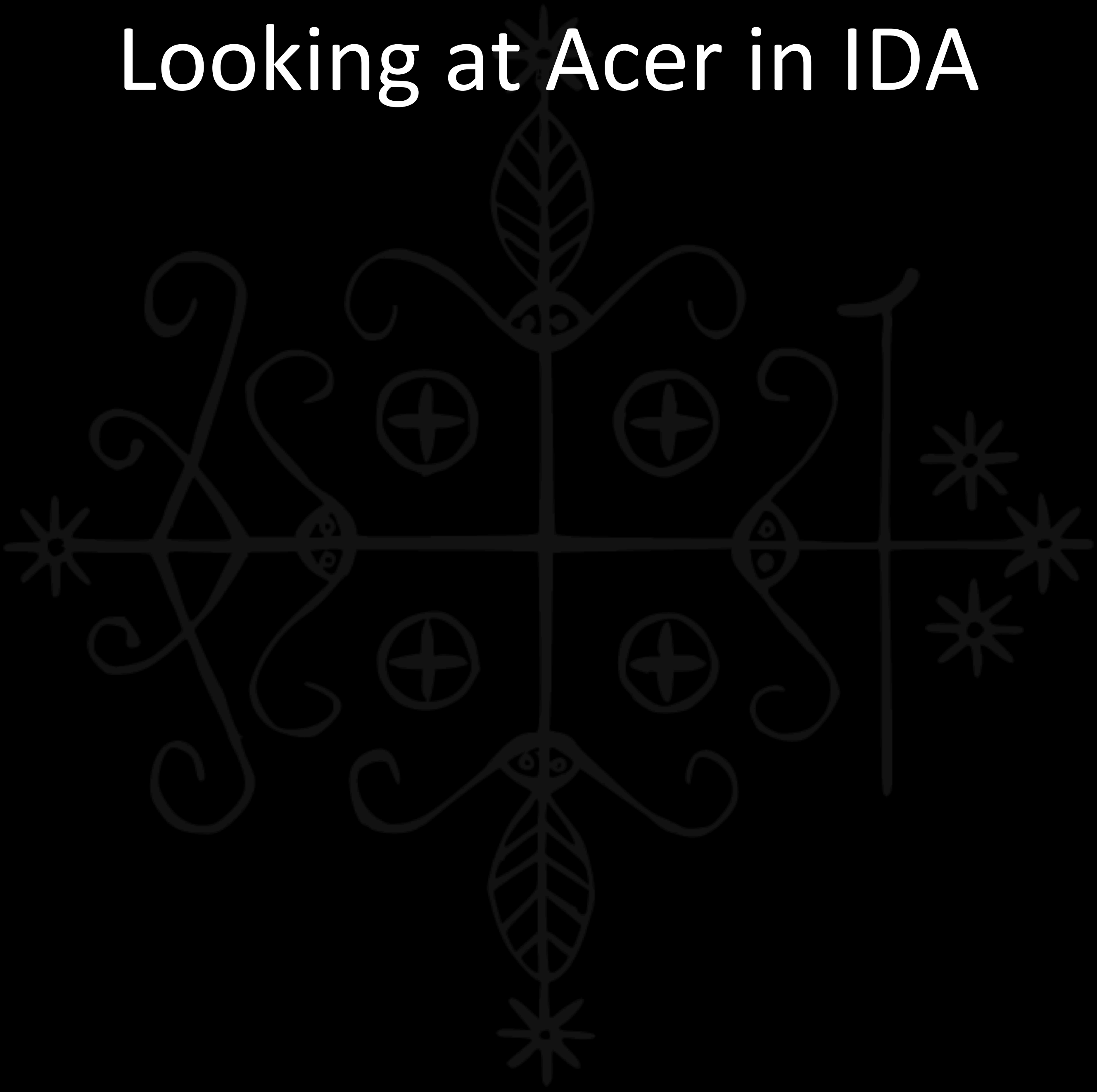


```
sub_D73A6B20((__int64)&v7, 0x40ui64);
sub_D73A6AE0((unsigned __int64)&v7, v1 + vD6EEEDF0, vD6EEEDF2);
sub_D73A6AE0((unsigned __int64)&v8, v1 + vD6EEEDF4, vD6EEEDF6);
sub_D73A6AE0((unsigned __int64)&v9, v1 + vD6EEEDF8, vD6EEEDFA);
sub_D73A6AE0((unsigned __int64)&v10, v1 + vD6EEEDFC, vD6EEEDFE);
sub_D73A6AE0((unsigned __int64)&v11, v1 + vD6EEEE00, vD6EEEE02);
sub_D73A6AE0((unsigned __int64)&v12, v1 + vD6EEEE04, vD6EEEE06);
sub_D73A6AE0((unsigned __int64)&v13, v1 + vD6EEEE08, vD6EEEE0A);
sub_D73A6AE0((unsigned __int64)&v14, v1 + vD6EEEE0C, vD6EEEE0E);
sub_D73A6AE0((unsigned __int64)&v15, v1 + vD6EEEE10, vD6EEEE12);
sub_D73A6AE0((unsigned __int64)&v16, v1 + vD6EEEE14, vD6EEEE16);
sub_D73A6AE0((unsigned __int64)&v17, v1 + vD6EEEE18, vD6EEEE1A);
sub_D73A6AE0((unsigned __int64)&v18, v1 + vD6EEEE1C, vD6EEEE1E);
sub_D73A6AE0((unsigned __int64)&v19, v1 + vD6EEEE20, vD6EEEE22);
sub_D73A6AE0((unsigned __int64)&v20, v1 + vD6EEEE24, vD6EEEE26);
sub_D73A6AE0((unsigned __int64)&v21, v1 + vD6EEEE28, vD6EEEE2A);
sub_D73A6AE0((unsigned __int64)&v22, v1 + vD6EEEE2C, vD6EEEE2E);
```



```
sub_D73A6B20((__int64)&v7, 0x40ui64);
memcopy((unsigned __int64)&v7, v1 + vD6EEEDF0, vD6EEEDF2);
memcopy((unsigned __int64)&v8, v1 + vD6EEEDF4, vD6EEEDF6);
memcopy((unsigned __int64)&v9, v1 + vD6EEEDF8, vD6EEEDFA);
memcopy((unsigned __int64)&v10, v1 + vD6EEEDFC, vD6EEEDFE);
memcopy((unsigned __int64)&v11, v1 + vD6EEEE00, vD6EEEE02);
memcopy((unsigned __int64)&v12, v1 + vD6EEEE04, vD6EEEE06);
memcopy((unsigned __int64)&v13, v1 + vD6EEEE08, vD6EEEE0A);
memcopy((unsigned __int64)&v14, v1 + vD6EEEE0C, vD6EEEE0E);
memcopy((unsigned __int64)&v15, v1 + vD6EEEE10, vD6EEEE12);
memcopy((unsigned __int64)&v16, v1 + vD6EEEE14, vD6EEEE16);
memcopy((unsigned __int64)&v17, v1 + vD6EEEE18, vD6EEEE1A);
memcopy((unsigned __int64)&v18, v1 + vD6EEEE1C, vD6EEEE1E);
memcopy((unsigned __int64)&v19, v1 + vD6EEEE20, vD6EEEE22);
memcopy((unsigned __int64)&v20, v1 + vD6EEEE24, vD6EEEE26);
memcopy((unsigned __int64)&v21, v1 + vD6EEEE28, vD6EEEE2A);
memcopy((unsigned __int64)&v22, v1 + vD6EEEE2C, vD6EEEE2E);
```

Looking at Acer in IDA



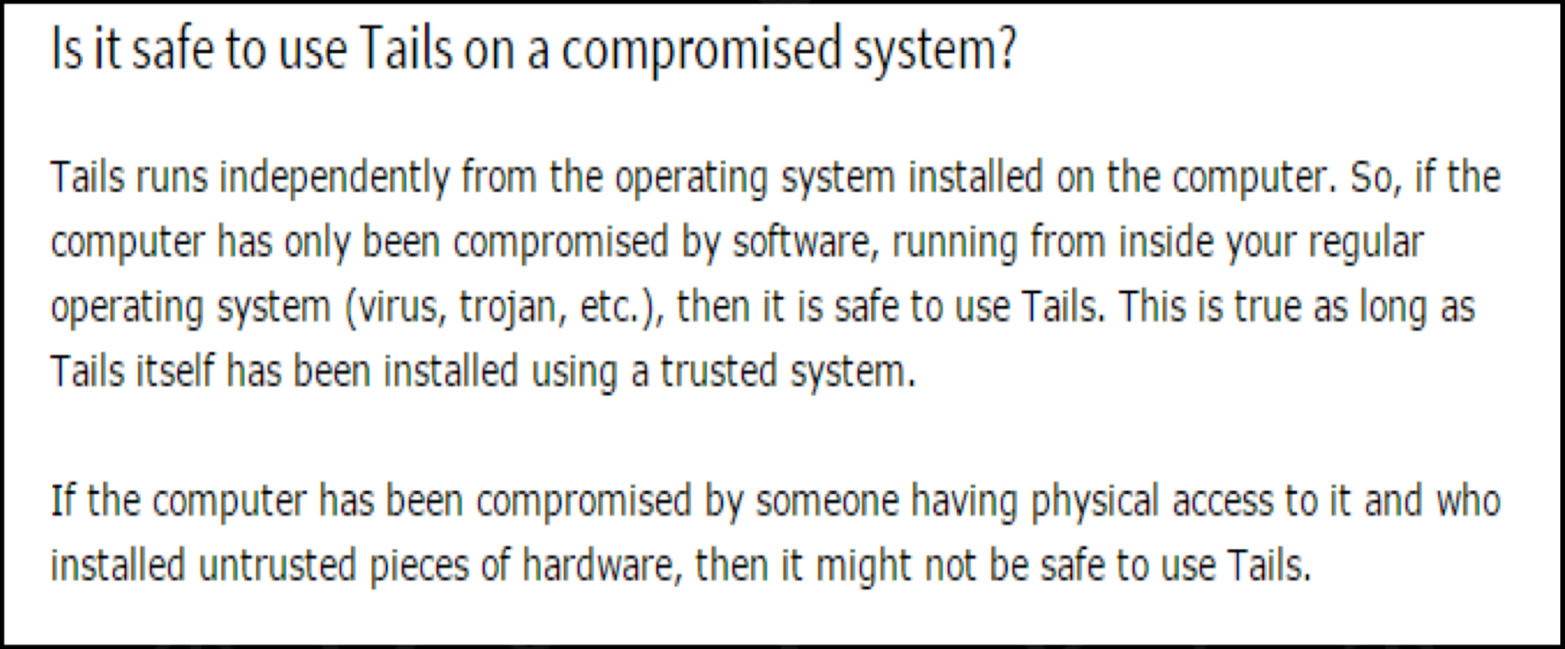
Vendor Response

- Many vendors didn’t reply to our emails and/or claimed they weren’t vulnerable
  - They are vulnerable
- Dell responded and is pushing patches for all of our disclosures
- Lenovo also responded and is releasing patches

What’s possible once you’ve broken into BIOS/SMM?

LightEater

Hello my friends.
Welcome to my home
in the Deep Dark

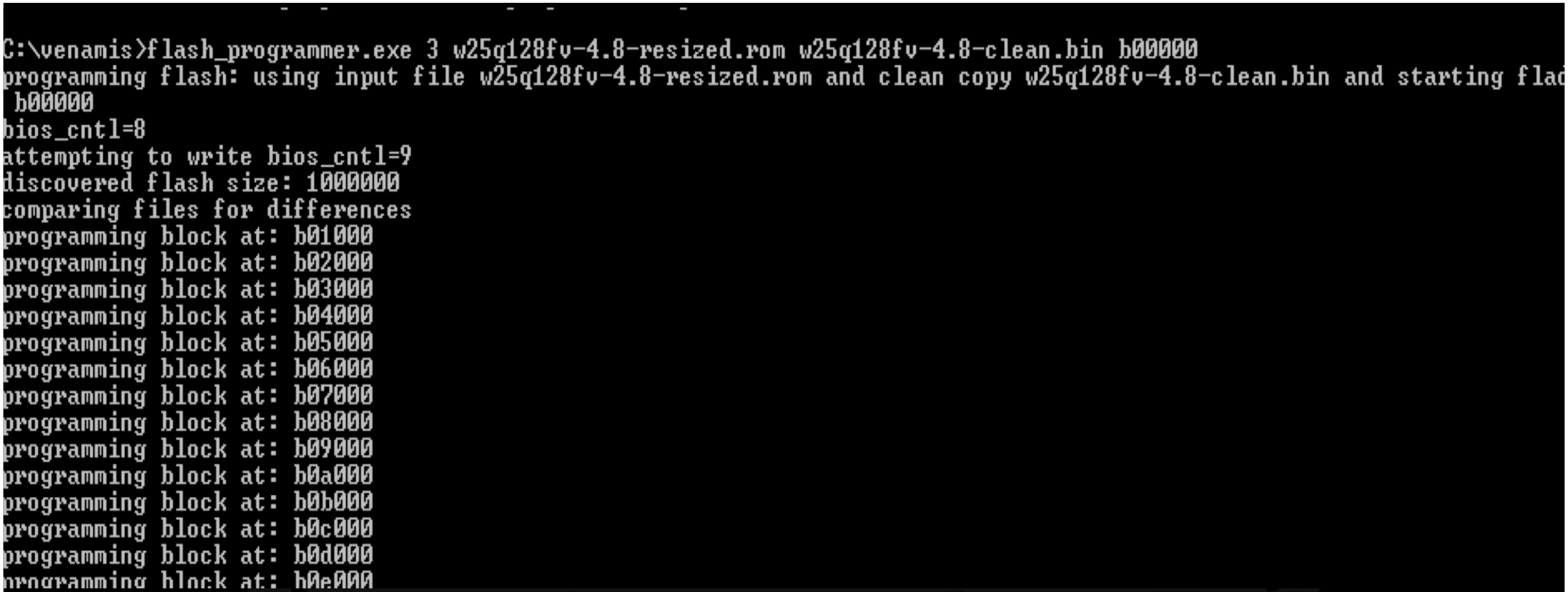


Is it safe to use Tails on a compromised system?

Tails runs independently from the operating system installed on the computer. So, if the computer has only been compromised by software, running from inside your regular operating system (virus, trojan, etc.), then it is safe to use Tails. This is true as long as Tails itself has been installed using a trusted system.

If the computer has been compromised by someone having physical access to it and who installed untrusted pieces of hardware, then it might not be safe to use Tails.

- Tails says that because it runs independent of the operating system, if you have previously been compromised by software means (not physical access), you should be safe…

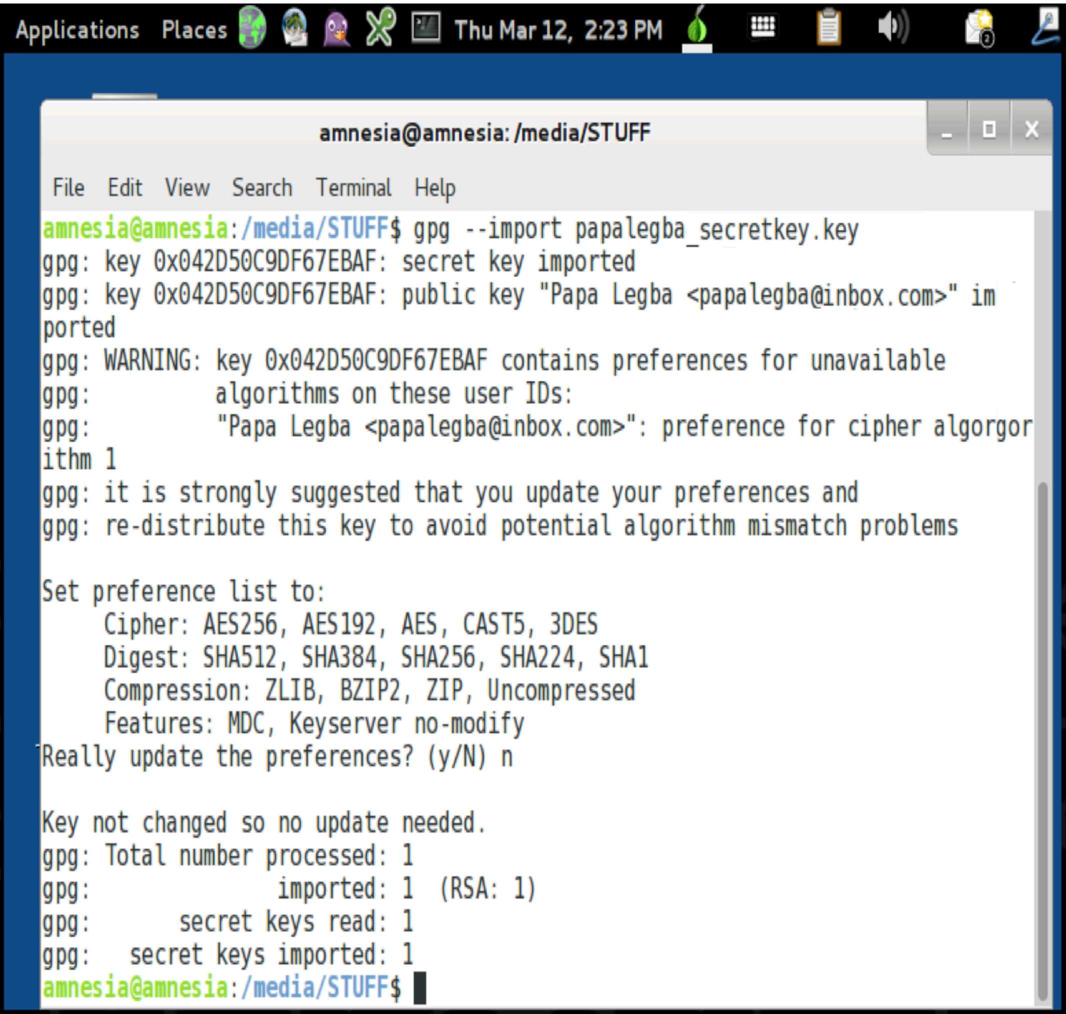


```
C:\venamis>flash_programmer.exe 3 w25q128fv-4.8-resized.rom w25q128fv-4.8-clean.bin b00000
programming flash: using input file w25q128fv-4.8-resized.rom and clean copy w25q128fv-4.8-clean.bin and starting fla
 b00000
bios_cntl=8
attempting to write bios_cntl=9
discovered flash size: 1000000
comparing files for differences
programming block at: b01000
programming block at: b02000
programming block at: b03000
programming block at: b04000
programming block at: b05000
programming block at: b06000
programming block at: b07000
programming block at: b08000
programming block at: b09000
programming block at: b0a000
programming block at: b0b000
programming block at: b0c000
programming block at: b0d000
programming block at: b0e000
```

- Exploit delivered remotely on target Windows 10 system.
  - No physical access is necessary
  - All you need is a remote cmd.exe with admin access
- Exploit bypasses BIOS flash protection and reprograms the portion of the flash associated with System Management Mode

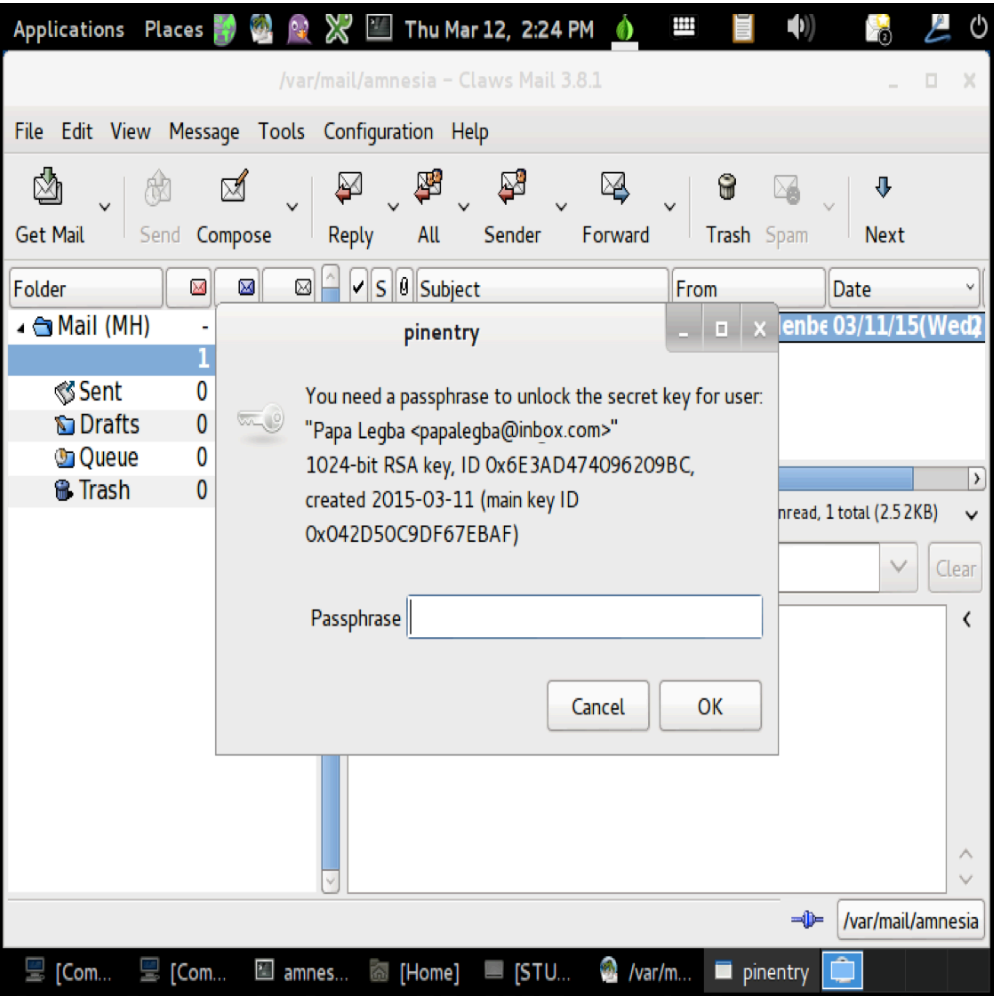
```
I hunger...
```

- Malware that was delivered remotely to the main OS (Windows 10) waits in the background and runs in System Management Mode
- It waits for your secrets to be revealed



```
amnesia@amnesia:/media/STUFF$ gpg --import papalegba_secretkey.key
gpg: key 0x042D50C9DF67EBAF: secret key imported
gpg: key 0x042D50C9DF67EBAF: public key "Papa Legba <papalegba@inbox.com>" im
ported
gpg: WARNING: key 0x042D50C9DF67EBAF contains preferences for unavailable
gpg:          algorithms on these user IDs:
gpg:          "Papa Legba <papalegba@inbox.com>": preference for cipher algorgor
ithm 1
gpg: it is strongly suggested that you update your preferences and
gpg: re-distribute this key to avoid potential algorithm mismatch problems

Set preference list to:
     Cipher: AES256, AES192, AES, CAST5, 3DES
     Digest: SHA512, SHA384, SHA256, SHA224, SHA1
     Compression: ZLIB, BZIP2, ZIP, Uncompressed
     Features: MDC, Keyserver no-modify
Really update the preferences? (y/N) n

Key not changed so no update needed.
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
gpg:       secret keys read: 1
gpg:   secret keys imported: 1
amnesia@amnesia:/media/STUFF$ 
```

- If you are practicing OPSEC, perhaps you have a private email and private key that you only access from the “secure” Tails so to avoid having confidential communications compromised


Applications Places Thu Mar 12, 2:24 PM
/var/mail/amnesia - Claws Mail 3.8.1
File Edit View Message Tools Configuration Help
Get Mail Send Compose Reply All Sender Forward Trash Spam Next
Folder
Subject
From
Date
Mail (MH)
Sent 0
Drafts 0
Queue 0
Trash 0
enbε 03/11/15(Wed)
nread, 1 total (2.52KB)
Clear
/var/mail/amnesia
[Com... [Com... amnes... [Home] [STU... /var/m... pinentry

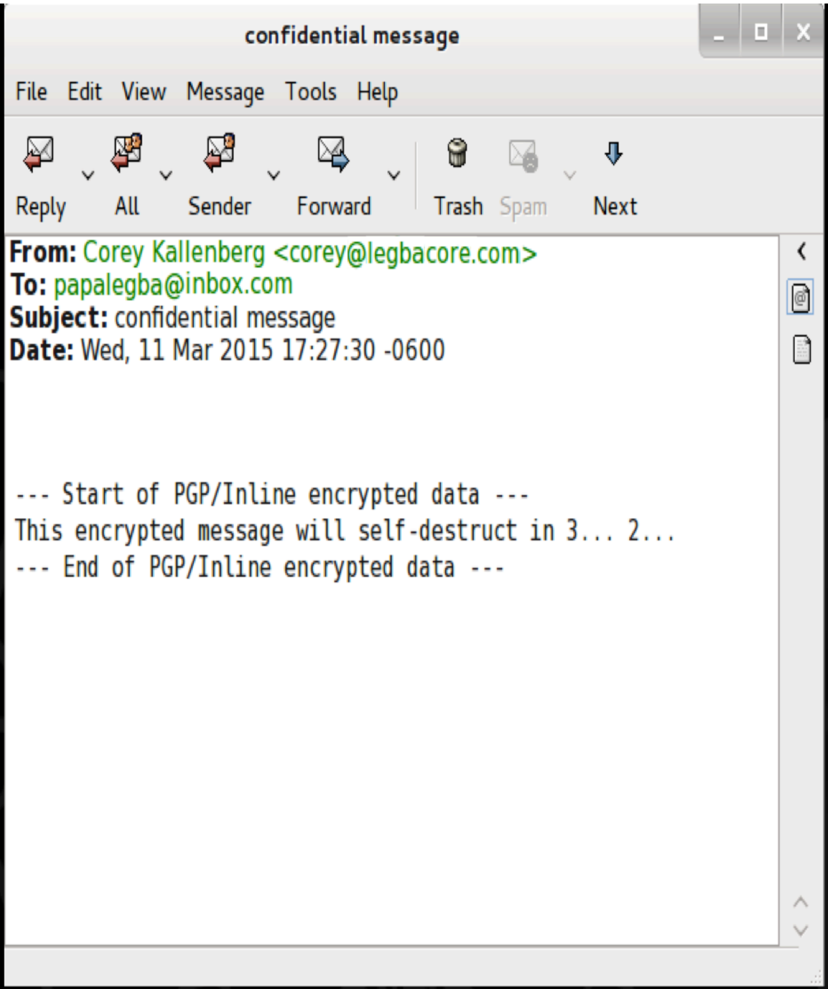
pinentry

You need a passphrase to unlock the secret key for user:
"Papa Legba <papalegba@inbox.com>"
1024-bit RSA key, ID 0x6E3AD474096209BC,
created 2015-03-11 (main key ID
0x042D50C9DF67EBAF)

Passphrase

Cancel OK

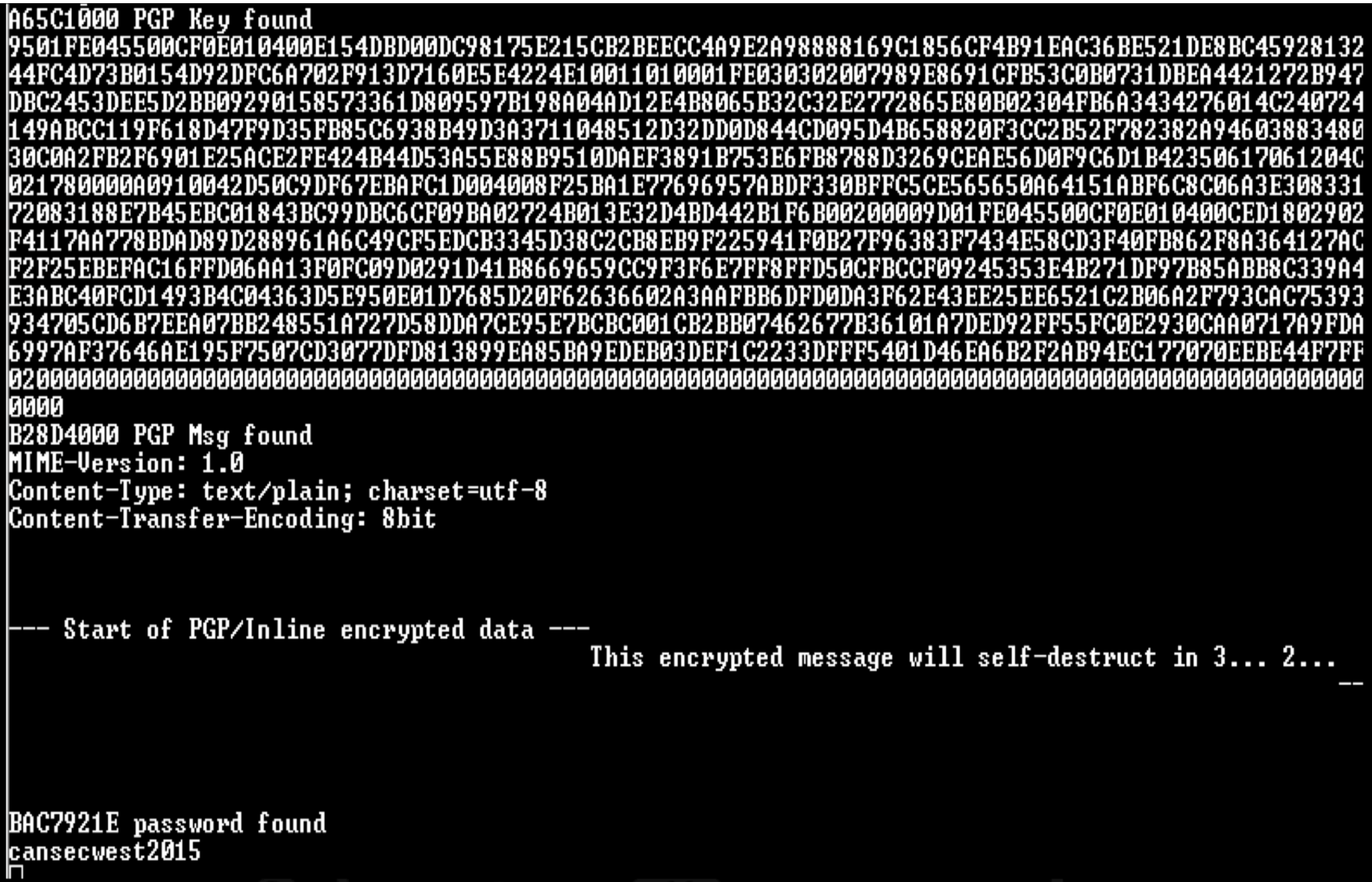
- Using this style of OPSEC, the password for your key should never be entered on your normal operating system (Win10 in this case).
- Since we are in Tails, we are okay though…



From: Corey Kallenberg <corey@legbacore.com>
To: papalegba@inbox.com
Subject: confidential message
Date: Wed, 11 Mar 2015 17:27:30 -0600

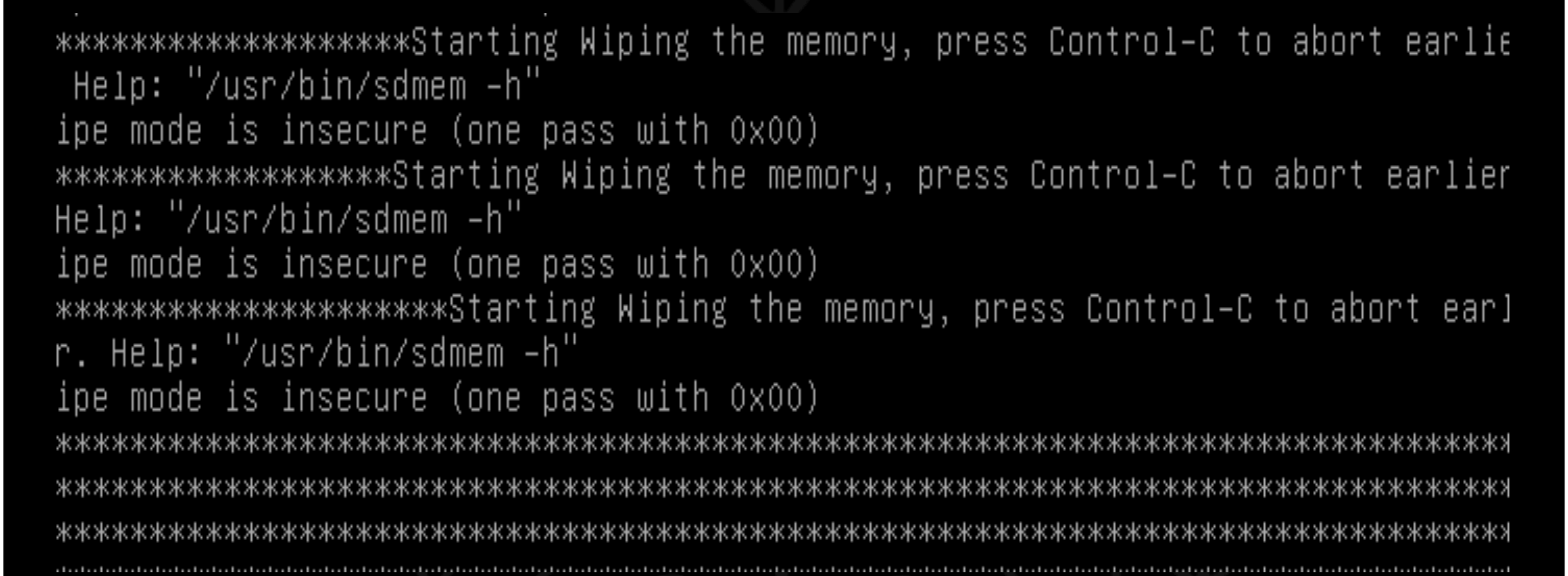
```
--- Start of PGP/Inline encrypted data ---
This encrypted message will self-destruct in 3... 2...
--- End of PGP/Inline encrypted data ---
```

- Hence all of your confidential communications should remain shielded from any malware that was delivered to your Win10 installation
- Using our malware, this isn't the case…



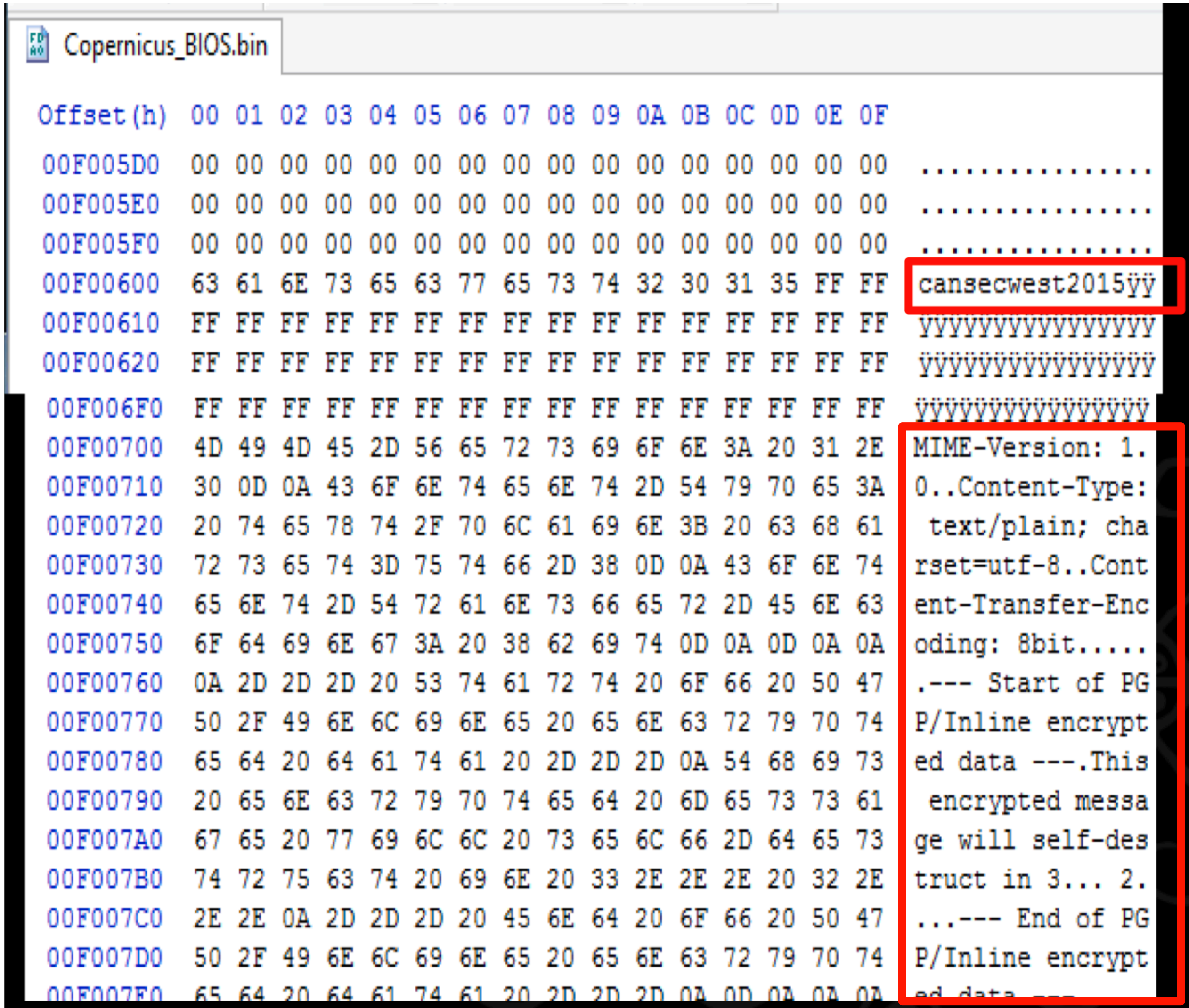
```
A65C1000 PGP Key found
9501FE045500CF0E010400E154DBD00DC98175E215CB2BEECC4A9E2A98888169C1856CF4B91EAC36BE521DE8BC45928132
44FC4D73B0154D92DFC6A702F913D7160E5E4224E10011010001FE030302007989E8691CFB53C0B0731DBEA4421272B947
DBC2453DEE5D2BB092901585733361D809597B198A04AD12E4B8065B32C32E2772865E80B02304FB6A3434276014C240724
149ABCC119F618D47F9D35FB85C6938B49D3A3711048512D32DD0D844CD095D4B658820F3CC2B52F782382A94603883480
30C0A2FB2F6901E25ACE2FE424B44D53A55E88B9510DAEF3891B753E6FB8788D3269CEAE56D0F9C6D1B423506170612040C
021780000A0910042D50C9DF67EBAFC1D004008F25BA1E77696957ABDF330BFFC5CE565650A64151ABF6C8C06A3E308331
72083188E7B45EBC01843BC99DBC6CF09BA02724B013E32D4BD442B1F6B00200009D01FE045500CF0E010400CED1802902
F4117AA778BDAD89D288961A6C49CF5EDCB3345D38C2CB8EB9F225941F0B27F96383F7434E58CD3F40FB862F8A364127AC
F2F25EBEFAC16FFD06AA13F0FC09D0291D41B8669659CC9F3F6E7FF8FFD50CFBCCF09245353E4B271DF97B85ABB8C339A4
E3ABC40FCD1493B4C04363D5E950E01D7685D20F62636602A3AAFBB6DFD0DA3F62E43EE25EE6521C2B06A2F793CAC75393
934705CD6B7EEA07BB248551A727D58DDA7CE95E7BCBC001CB2BB07462677B36101A7DED92FF55FC0E2930CAA0717A9FDA
6997AF37646AE195F7507CD3077DFD813899EA85BA9EDEB03DEF1C2233DFFF5401D46EA6B2F2AB94EC177070EEBE44F7FF
02000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000
B28D4000 PGP Msg found
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit



--- Start of PGP/Inline encrypted data ---
                                          This encrypted message will self-destruct in 3... 2...
                                                                                                --




BAC7921E password found
cansecwest2015
```

- Runs independent of any operating system you put on the platform
- Has access to all of memory
- Can steal all of your secrets no matter what “secure operating system” you are using



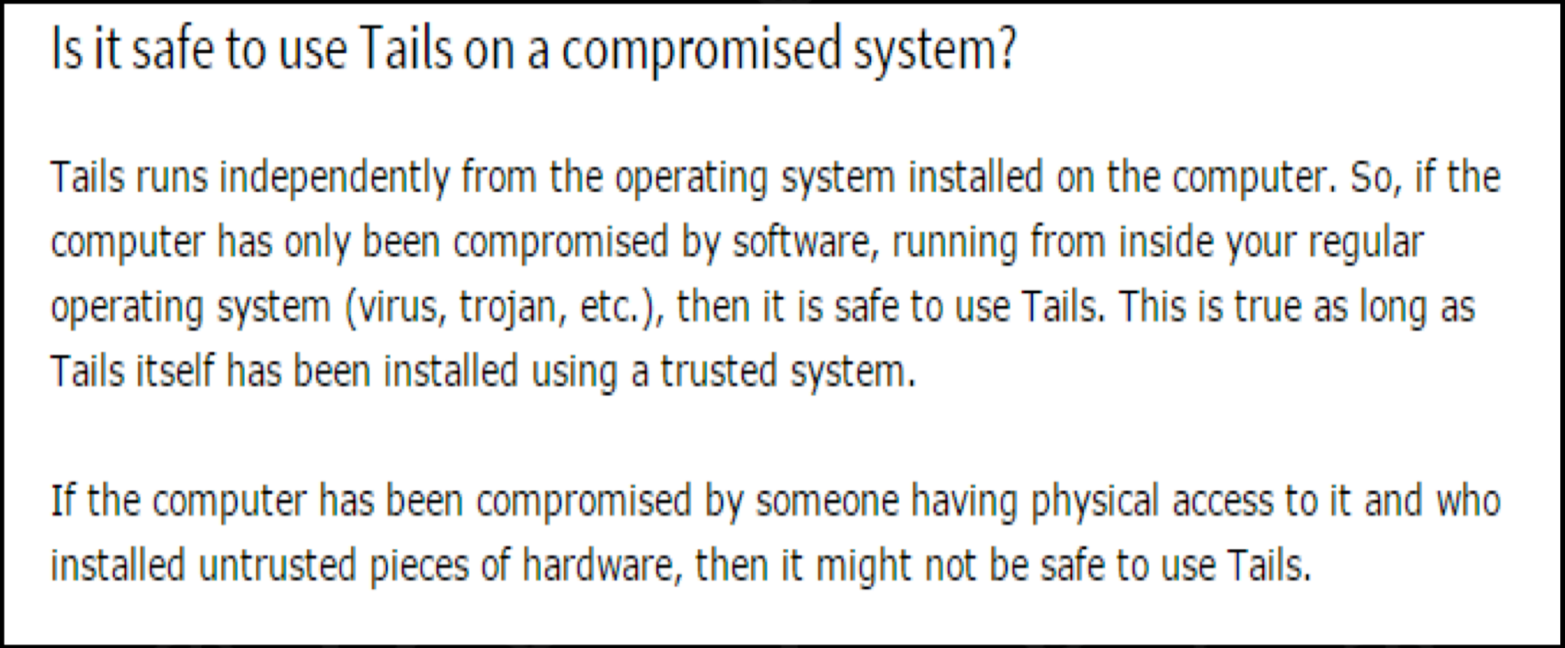
```
*******************Starting Wiping the memory, press Control-C to abort earlie
 Help: "/usr/bin/sdmem -h"
ipe mode is insecure (one pass with 0x00)
******************Starting Wiping the memory, press Control-C to abort earlier
Help: "/usr/bin/sdmem -h"
ipe mode is insecure (one pass with 0x00)
*********************Starting Wiping the memory, press Control-C to abort earl
r. Help: "/usr/bin/sdmem -h"
ipe mode is insecure (one pass with 0x00)
******************************************************************************
******************************************************************************
******************************************************************************
```

- Tails also attempts to erase memory to scrub any residual secrets that may be exposed to the main operating system



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00F005D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00F005E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00F005F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00F00600  63 61 6E 73 65 63 77 65 73 74 32 30 31 35 FF FF  cansecwest2015ÿÿ
00F00610  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00F00620  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00F006F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00F00700  4D 49 4D 45 2D 56 65 72 73 69 6F 6E 3A 20 31 2E  MIME-Version: 1.
00F00710  30 0D 0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A  0..Content-Type:
00F00720  20 74 65 78 74 2F 70 6C 61 69 6E 3B 20 63 68 61   text/plain; cha
00F00730  72 73 65 74 3D 75 74 66 2D 38 0D 0A 43 6F 6E 74  rset=utf-8..Cont
00F00740  65 6E 74 2D 54 72 61 6E 73 66 65 72 2D 45 6E 63  ent-Transfer-Enc
00F00750  6F 64 69 6E 67 3A 20 38 62 69 74 0D 0A 0D 0A 0A  oding: 8bit.....
00F00760  0A 2D 2D 2D 20 53 74 61 72 74 20 6F 66 20 50 47  .--- Start of PG
00F00770  50 2F 49 6E 6C 69 6E 65 20 65 6E 63 72 79 70 74  P/Inline encrypt
00F00780  65 64 20 64 61 74 61 20 2D 2D 2D 0A 54 68 69 73  ed data ---.This
00F00790  20 65 6E 63 72 79 70 74 65 64 20 6D 65 73 73 61   encrypted messa
00F007A0  67 65 20 77 69 6C 6C 20 73 65 6C 66 2D 64 65 73  ge will self-des
00F007B0  74 72 75 63 74 20 69 6E 20 33 2E 2E 2E 20 32 2E  truct in 3... 2.
00F007C0  2E 2E 0A 2D 2D 2D 20 45 6E 64 20 6F 66 20 50 47  ...--- End of PG
00F007D0  50 2F 49 6E 6C 69 6E 65 20 65 6E 63 72 79 70 74  P/Inline encrypt
00F007E0  65 64 20 64 61 74 61 20 2D 2D 2D 0A 0D 0A 0A 0A  ed data ---
```

- Our malware still has access to it, as we store the secrets to non-volatile storage so we can exfil at earliest convenience
  - So even if you were to use Tails in offline mode, to try to avoid exfiltration of secrets, you can still be owned



Is it safe to use Tails on a compromised system?

Tails runs independently from the operating system installed on the computer. So, if the computer has only been compromised by software, running from inside your regular operating system (virus, trojan, etc.), then it is safe to use Tails. This is true as long as Tails itself has been installed using a trusted system.

If the computer has been compromised by someone having physical access to it and who installed untrusted pieces of hardware, then it might not be safe to use Tails.

- Time to rethink this…

All fall before a LightEater

- The US Air Force made the “Lightweight Portable Security” (LPS) Live CD[1] with much the same purpose as Tails:
- “LPS differs from traditional operating systems in that it isn't continually patched. LPS is designed to run from read-only media and without any persistent storage. *Any malware that might infect a computer can only run within that session*.”
- “LPS-Public turns an untrusted system (such as a home computer) into a trusted network client. *No trace of work activity (or malware) can be written to the local computer*. Simply plug in your USB smart card reader *to access CAC- and PIV-restricted US government websites*.”
- Attackers that infect BIOS will always win against non-persistent OSes, because they can persist across reboots, and live in OS-independent SMM

[1] http://www.spi.dod.mil/lipose.htm

TELL EVERYONE THEIR COMPUTERS ARE ARCHITECTURALLY INSECURE AT THE LOWEST LEVELS AND NOBODY BATS AN EYE
STEAL ONE GPG KEY FROM MEMORY IN TAILS AND EVERYONE LOSES THEIR MINDS
memegenerator.net

Where's the architectural flaw?

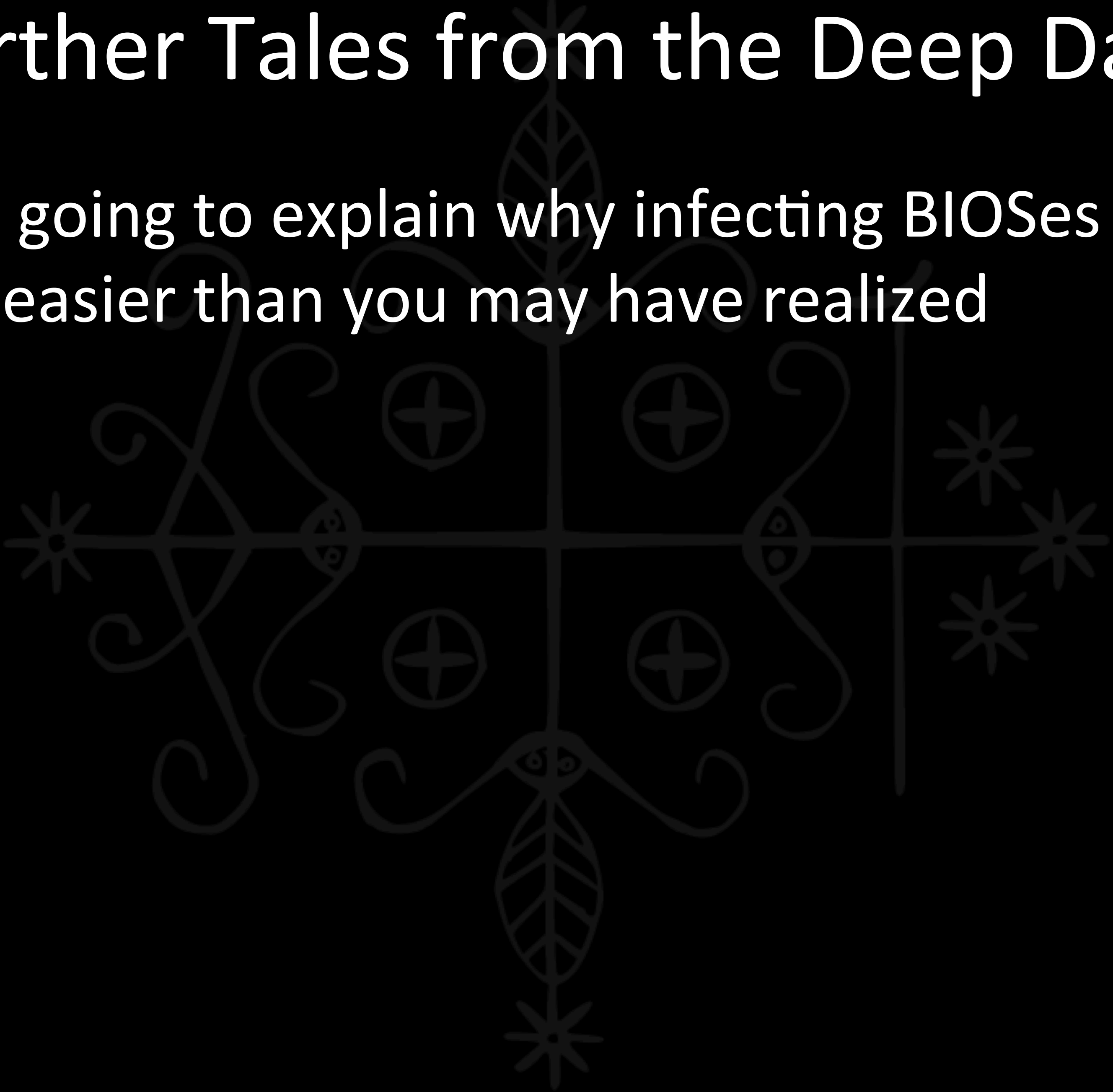
- The fact that SMM can read/write everyone's memory is an x86 architectural vulnerability
- No security system (virtualization, live CDs, normal OSes) is secure until this is fixed
  - We'll come back later to how we intend to fix it

This talk has 2 main points

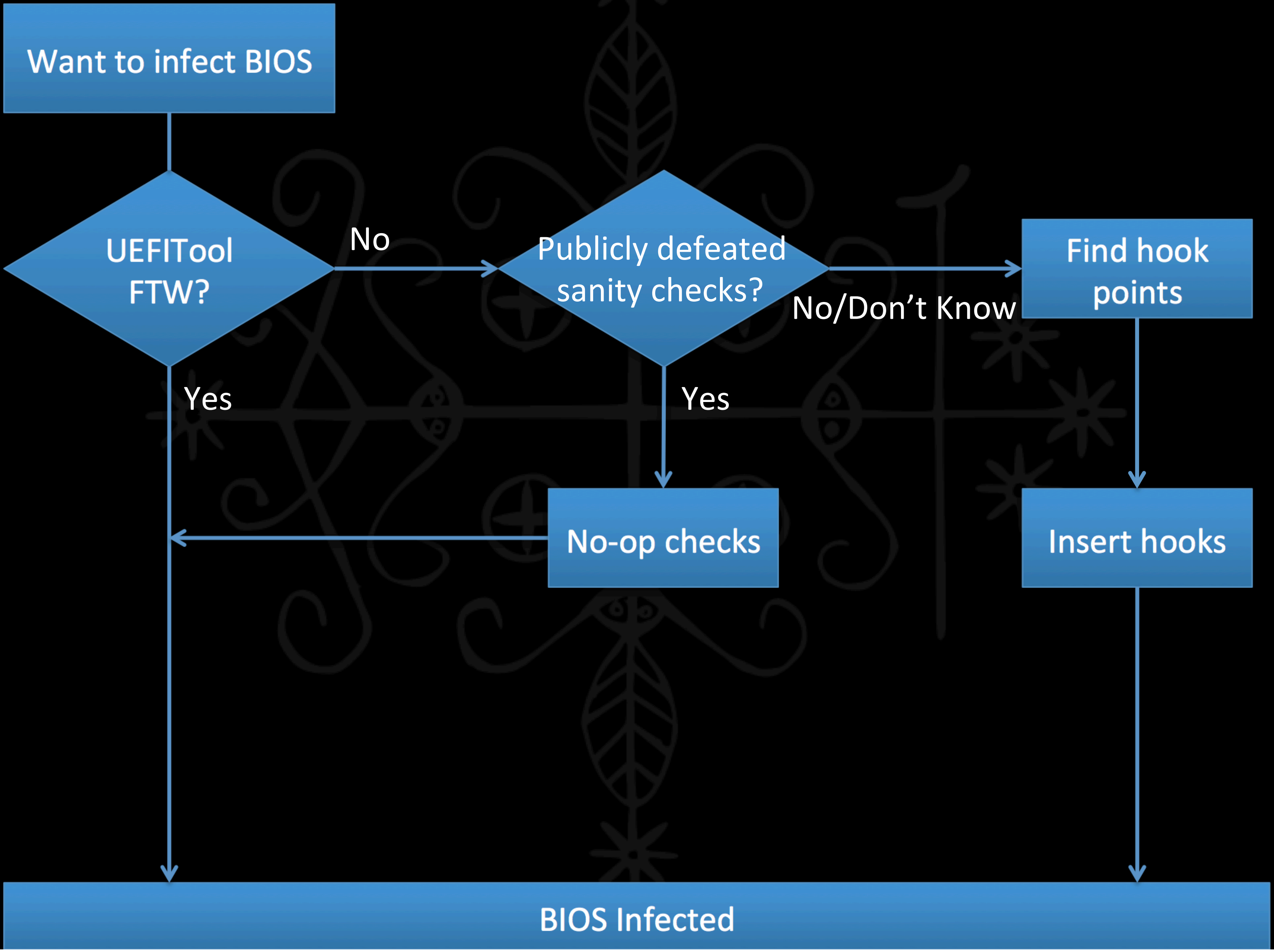
- Because almost no one applies BIOS patches, almost every BIOS in the wild is affected by *at least* one vulnerability, and can be infected
- The high amount of code reuse across UEFI BIOSes means that BIOS infection is automatable and reliable

Further Tales from the Deep Dark

- I'm going to explain why infecting BIOSes is *a lot* easier than you may have realized



Infection Decision Tree



Want to infect BIOS

UEFITool FTW?

No

Yes

Publicly defeated sanity checks?

No/Don’t Know

Yes

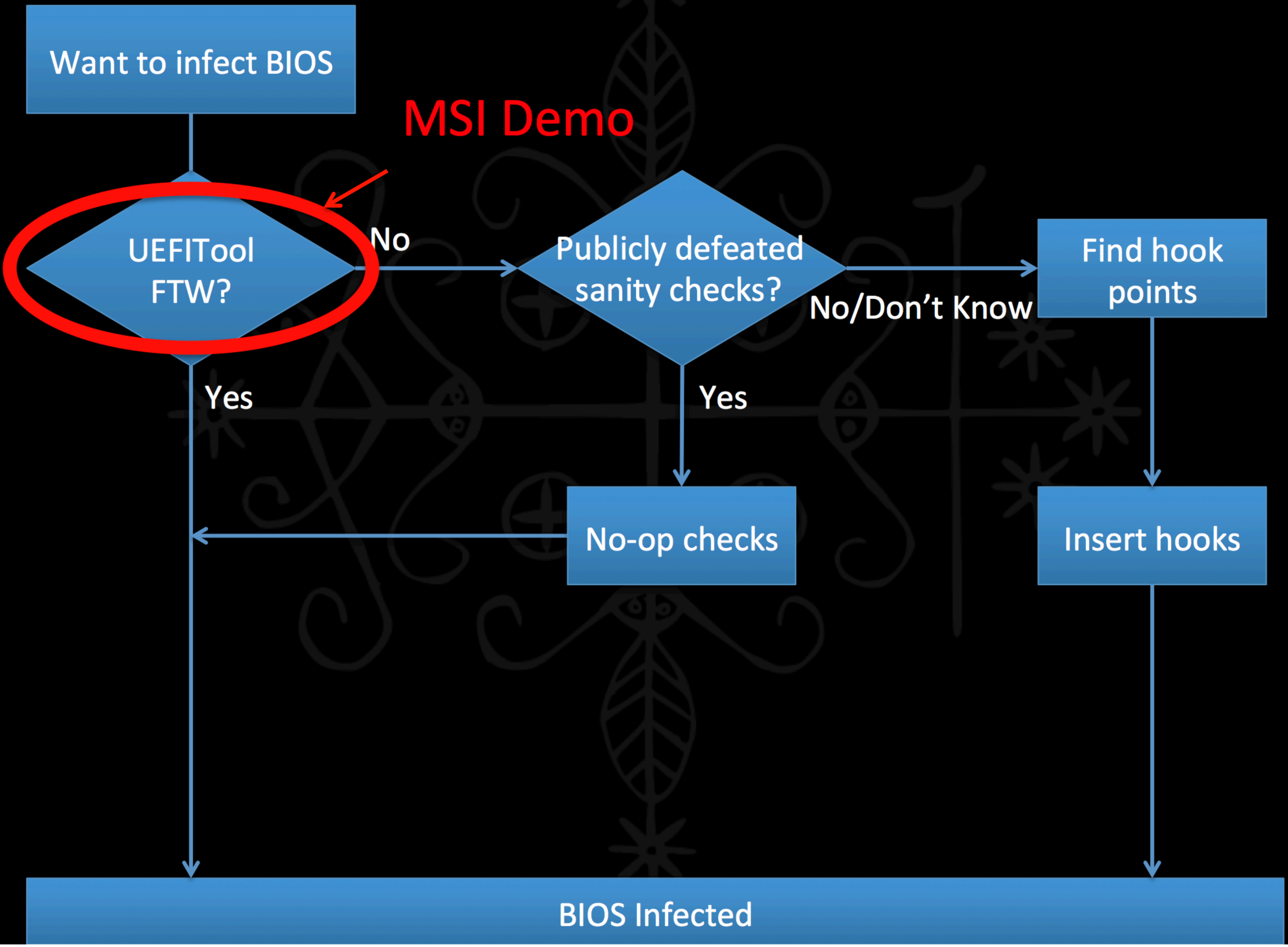
Find hook points

No-op checks

Insert hooks

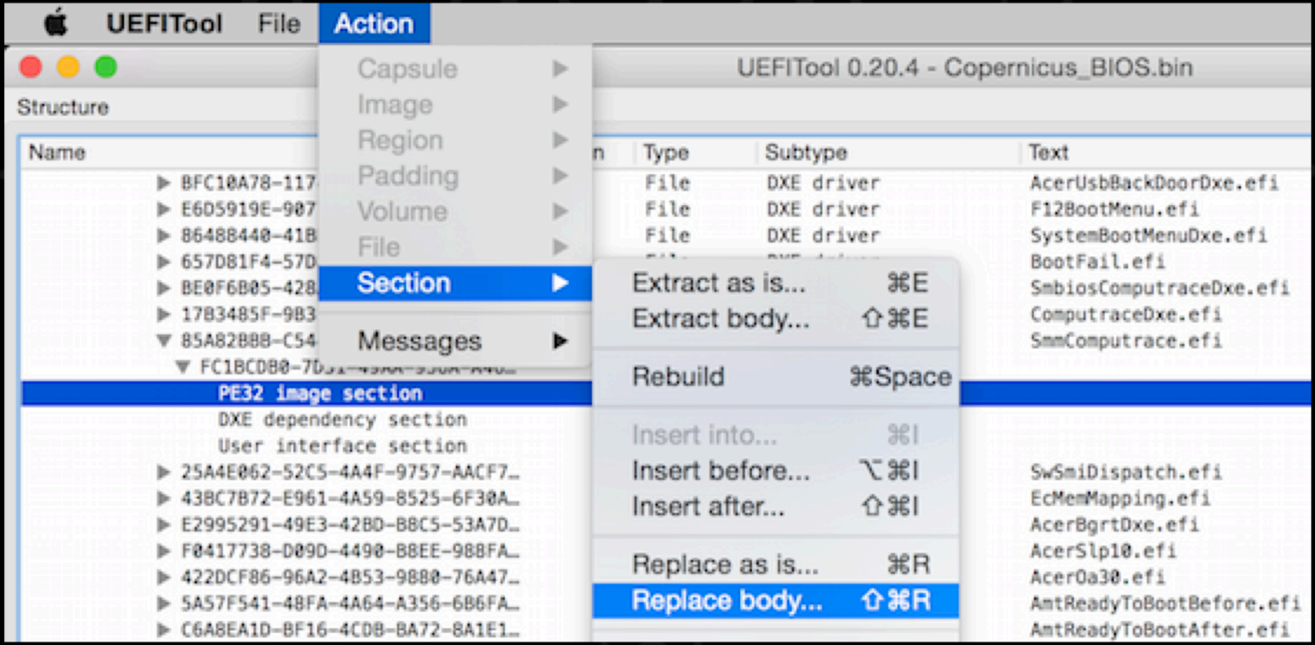
BIOS Infected

Infection Decision Tree



“UEFITool FTW” Infection

- As done on the MSI
- Use Nikolaj Schlej’s excellent UEFITool[1] to replace the module you care about with one that includes malicious functionality



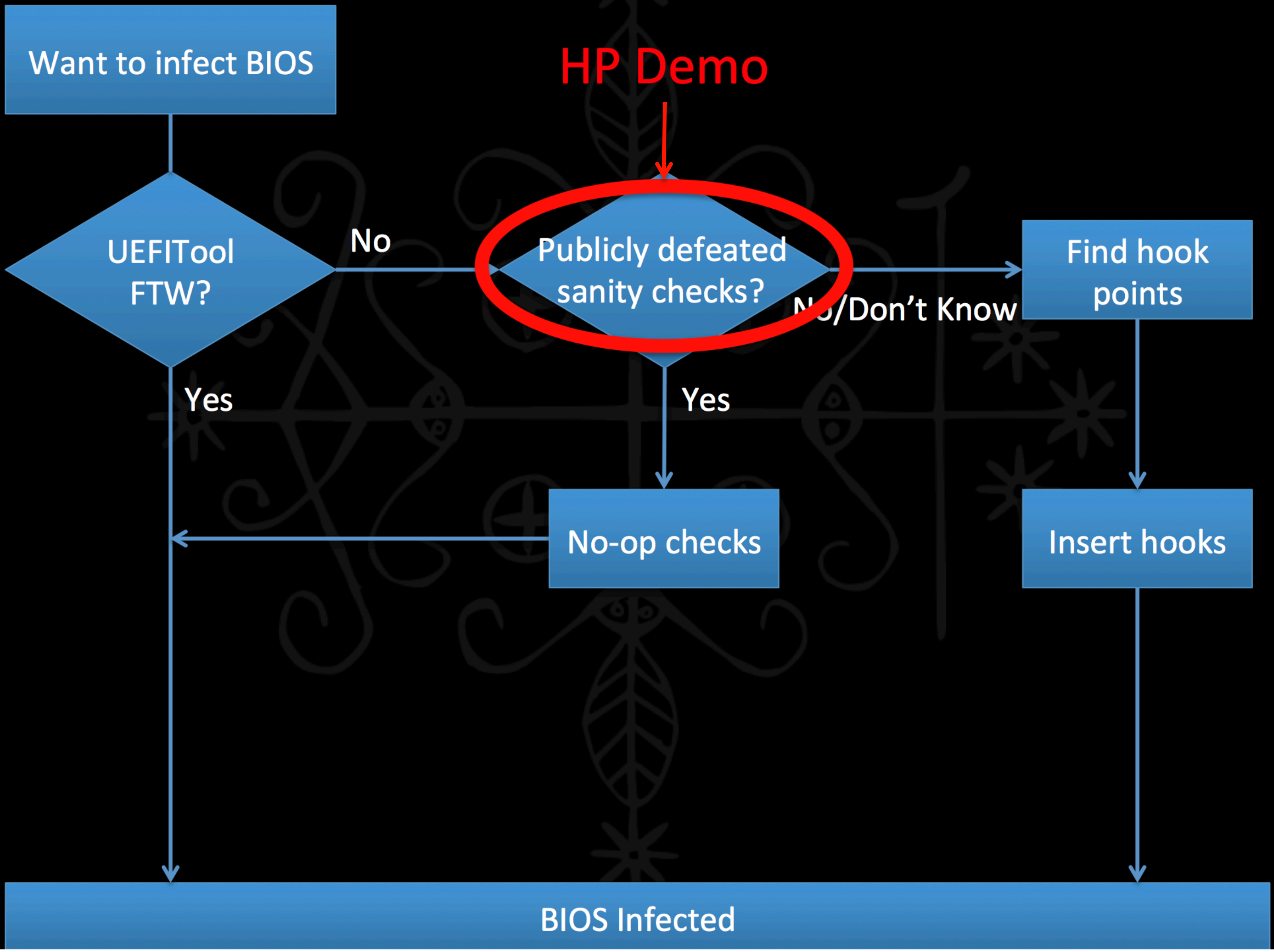
Name	Type	Subtype	Text
▶ BFC10A78-117	File	DXE driver	AcerUsbBackDoorDxe.efi
▶ E6D5919E-907	File	DXE driver	F12BootMenu.efi
▶ 86488440-41B	File	DXE driver	SystemBootMenuDxe.efi
▶ 657D81F4-57D			BootFail.efi
▶ BE0F6B05-428			SmbiosComputraceDxe.efi
▶ 17B3485F-9B3			ComputraceDxe.efi
▼ 85A82BBB-C54			SmmComputrace.efi
▼ FC1BCDB0-7D…			
PE32 image section			
DXE dependency section			
User interface section			
▶ 25A4E062-52C5-4A4F-9757-AACF7…			SwSmiDispatch.efi
▶ 43BC7B72-E961-4A59-8525-6F30A…			EcMemMapping.efi
▶ E2995291-49E3-42BD-B8C5-53A7D…			AcerBgrtDxe.efi
▶ F0417738-D09D-4490-B8EE-988FA…			AcerSlp10.efi
▶ 422DCF86-96A2-4B53-9880-76A47…			AcerOa30.efi
▶ 5A57F541-48FA-4A64-A356-6B6FA…			AmtReadyToBootBefore.efi
▶ C6A8EA1D-BF16-4CDB-BA72-8A1E1…			AmtReadyToBootAfter.efi

← hmm…

- Reflash w/ exploit FTW

[1] https://github.com/LongSoft/UEFITool

Infection Decision Tree



HP Demo

Want to infect BIOS

UEFITool FTW?

No

Publicly defeated sanity checks?

No/Don't Know

Find hook points

Yes

Yes

No-op checks

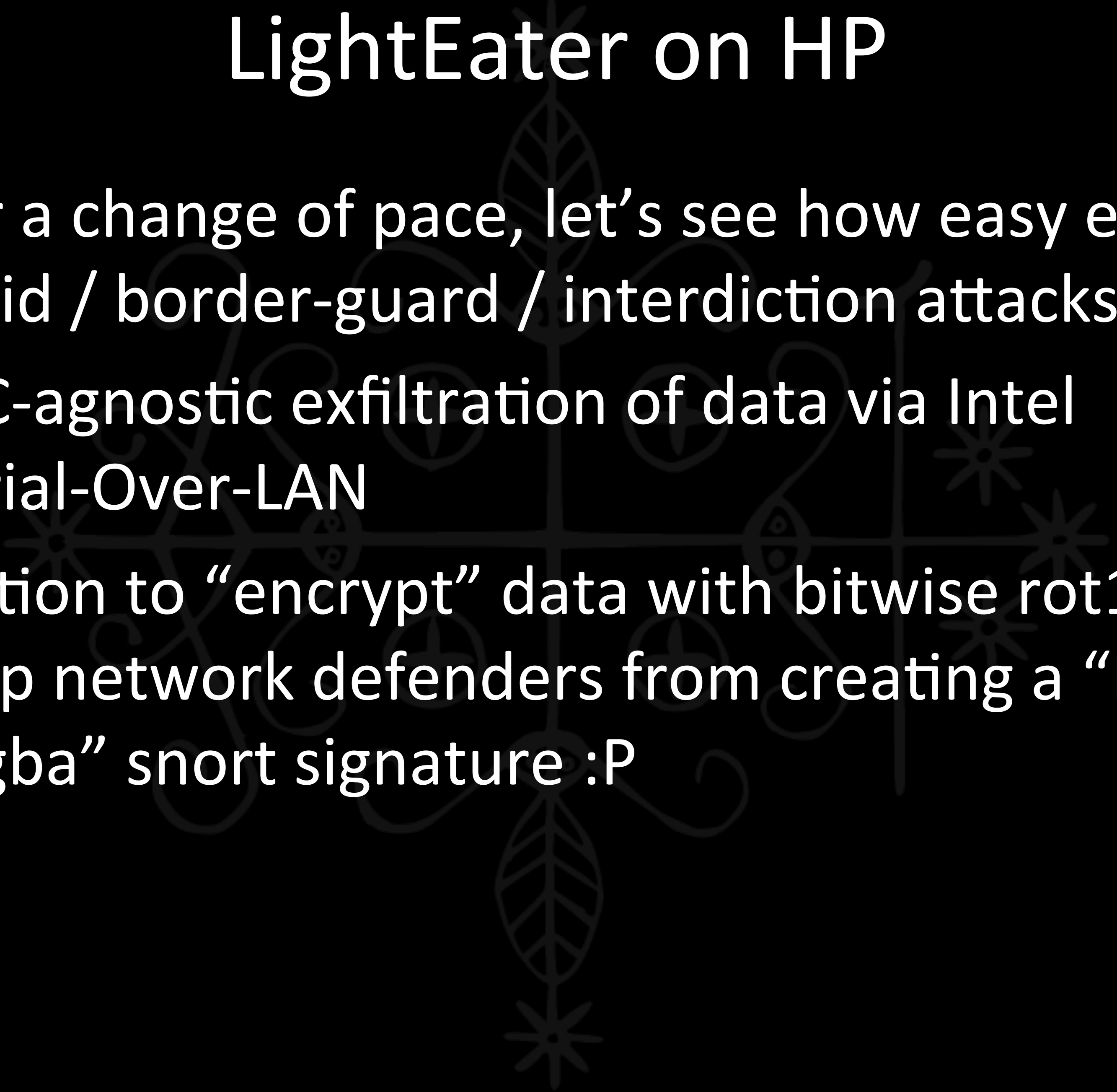
Insert hooks

BIOS Infected

Sanity Check Speed Bumps

- Some vendors like HP build in sanity checks
- Descriptions of bypasses can be easily found on the net, and would be developed quietly by anyone who actually cared enough
- We created a 9 byte signature for one HP sanity check by following the steps in a public blog post
  - And 2 variant signatures based on looking at a few models where the signature didn't fire
  - The 3 signature variants matched 570/789 HP BIOS images
    - Could be improved further, but we're just making a point
- If signature found, replace the last 2 bytes w/ 0x9090
- Goto previous slide

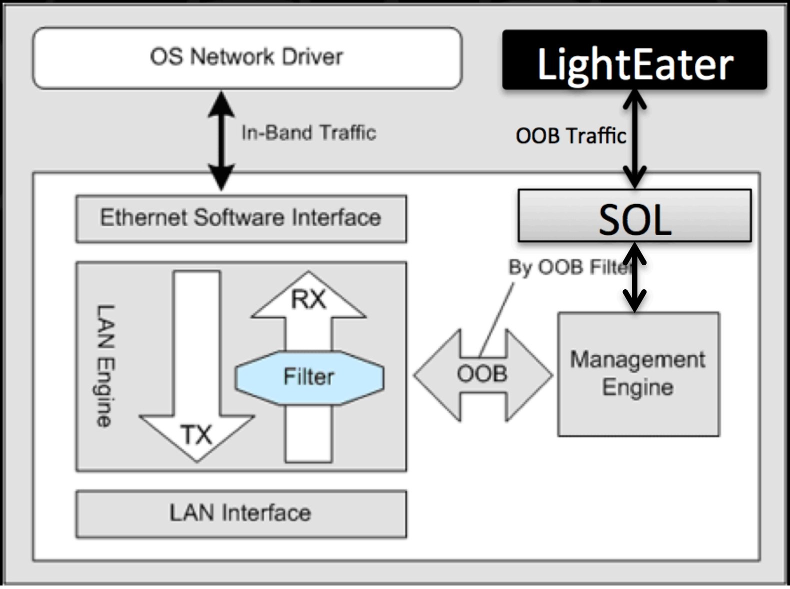
LightEater on HP



- For a change of pace, let’s see how easy evil-maid / border-guard / interdiction attacks are!
- NIC-agnostic exfiltration of data via Intel Serial-Over-LAN
- Option to “encrypt” data with bitwise rot13 to stop network defenders from creating a “Papa Legba” snort signature :P

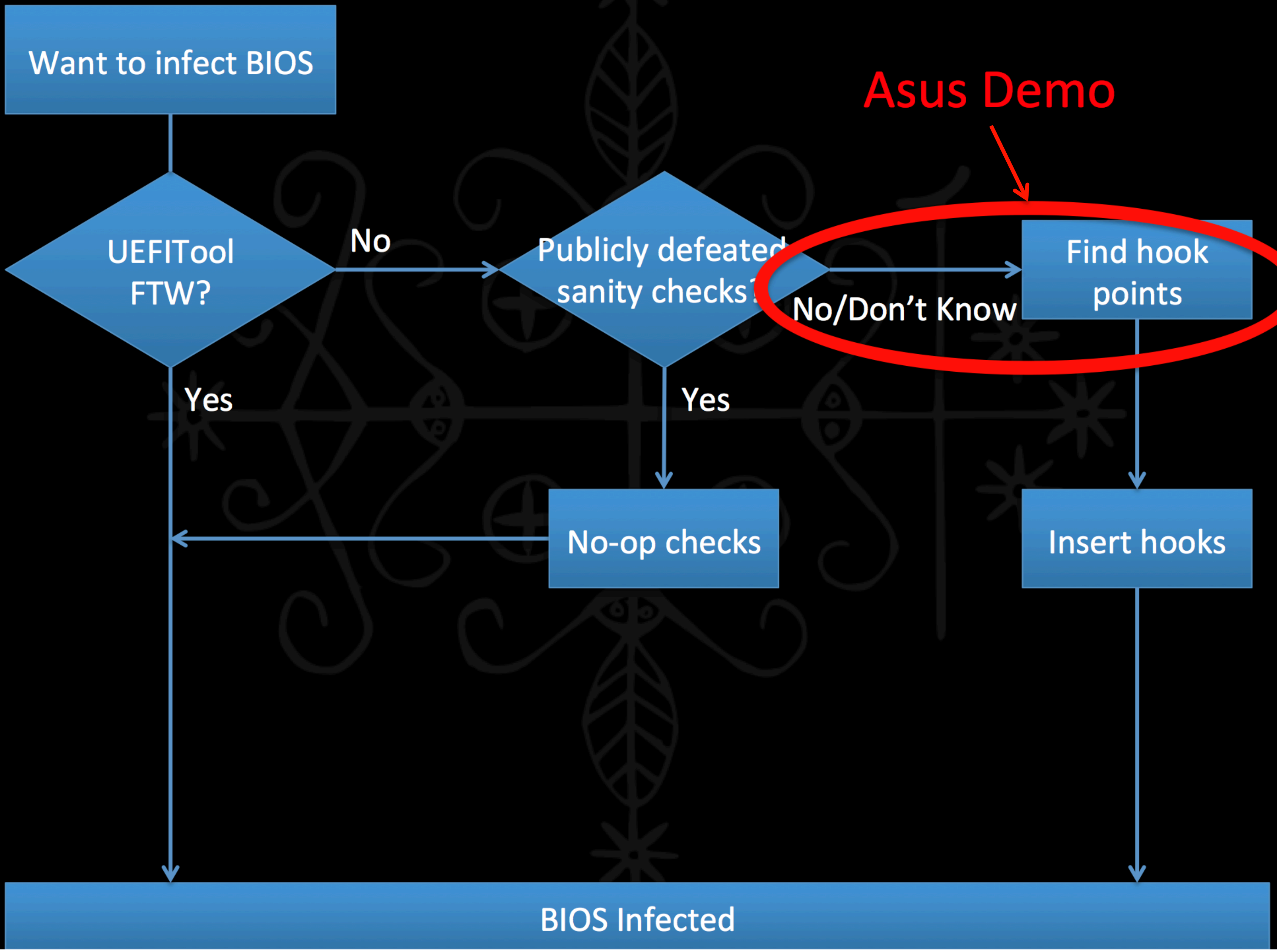
A word about AMT SOL

- Unlike past work for low level networking[10-12], we don't need to know anything about the NIC
- We write to a fake serial port that AMT creates
- AMT magically translates it to Intel's proprietary SOL protocol (that there's no wireshark dissector for)



https://software.intel.com/en-us/articles/architecture-guide-intel-active-management-technology

Infection Decision Tree

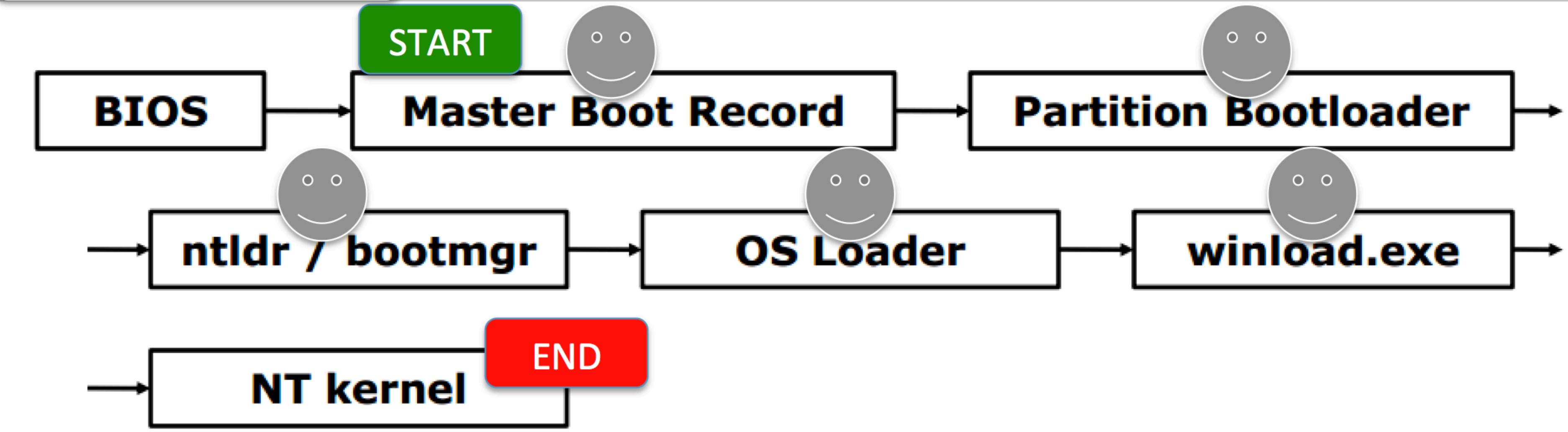


“BIOSkit” Infection

- Sometimes UEFITool doesn’t work, and you don’t care enough to RE why
- Fall back to the generic technique of “hook-and-hop”, just like a normal bootkit
  - Just starting earlier, and more privileged
- You’re more or less guaranteed that there’s an easily targeted, uncompressed, easy-to-hook starting location in the PEI core file

Reminder of how normal bootkits work

Windows Boot Process



`Ntldr` = 16-bit stub + `OS Loader` (just binary appended)
Windows Vista splits up `ntldr` into `bootmgr`, `winload.exe` and `winresume.exe`

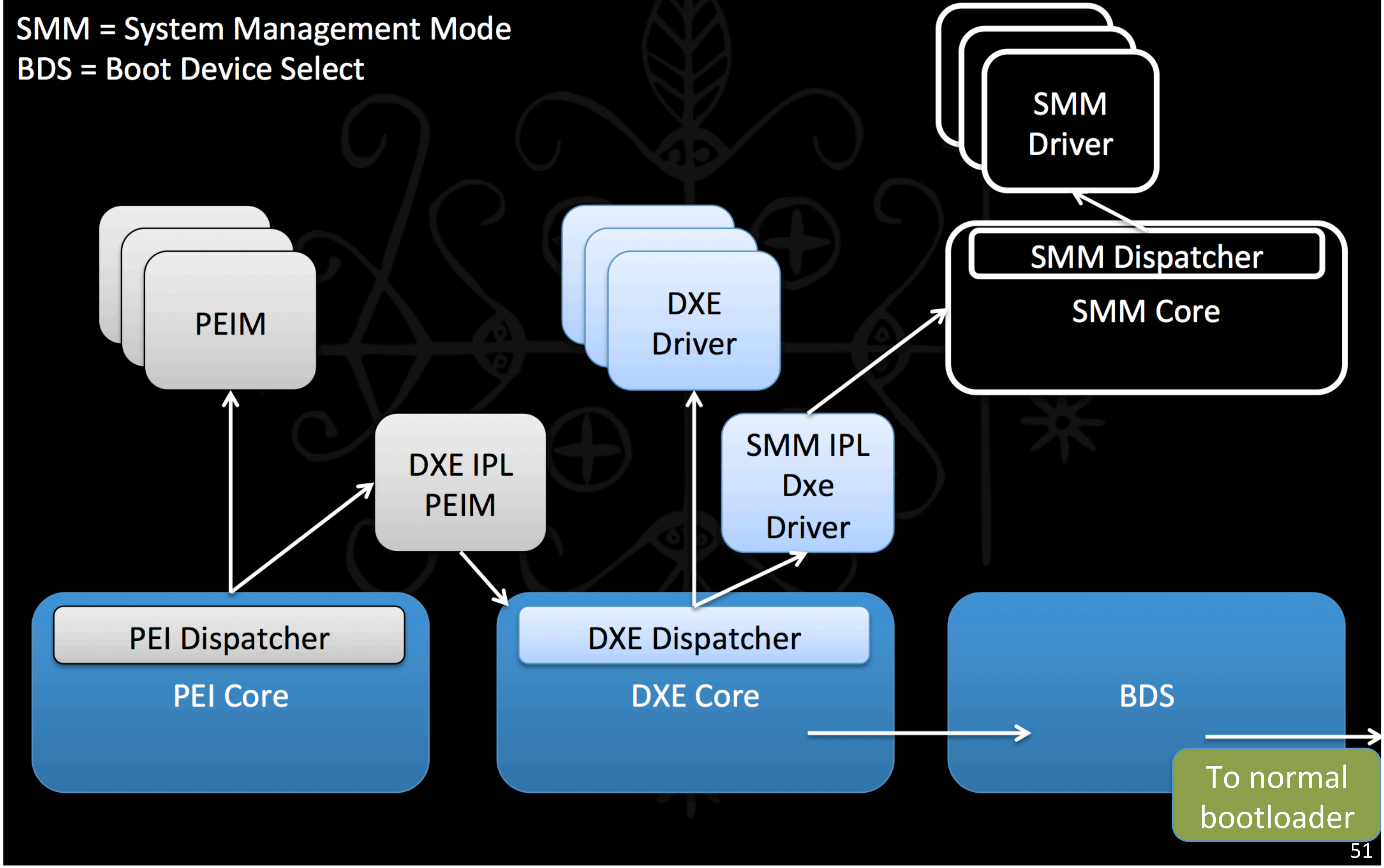
Windows XP	Windows Vista	Processor Environment
ntldr	bootmgr	Real Mode
OS Loader	OS Loader	Protected Mode
-	winload.exe	Protected Mode
NT kernel	NT kernel	Protected Mode + Paging

Modified from http://www.blackhat.com/presentations/bh-usa-09/KLEISSNER/BHUSA09-Kleissner-StonedBootkit-SLIDES.pdf

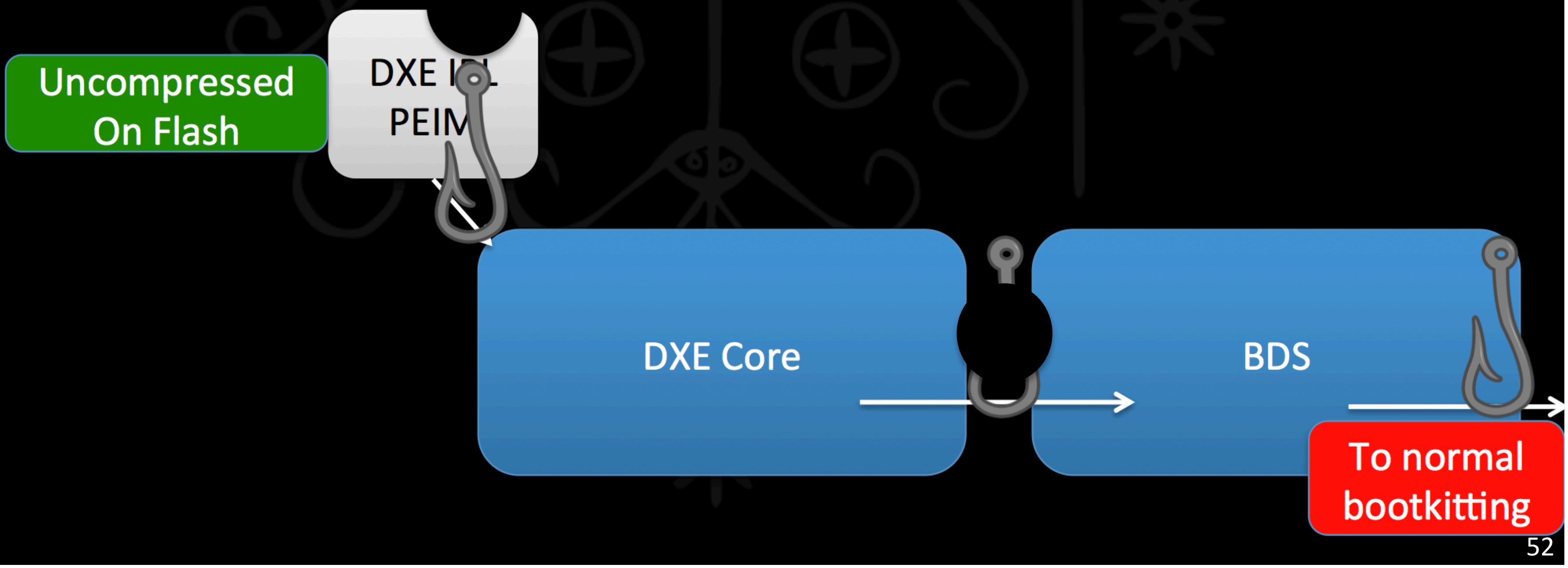
PEI = Pre EFI Initialization
PEIM = PEI Module
IPL = Initial Program Loader
DXE = Driver Execution Environment
SMM = System Management Mode
BDS = Boot Device Select

The UEFI skeleton

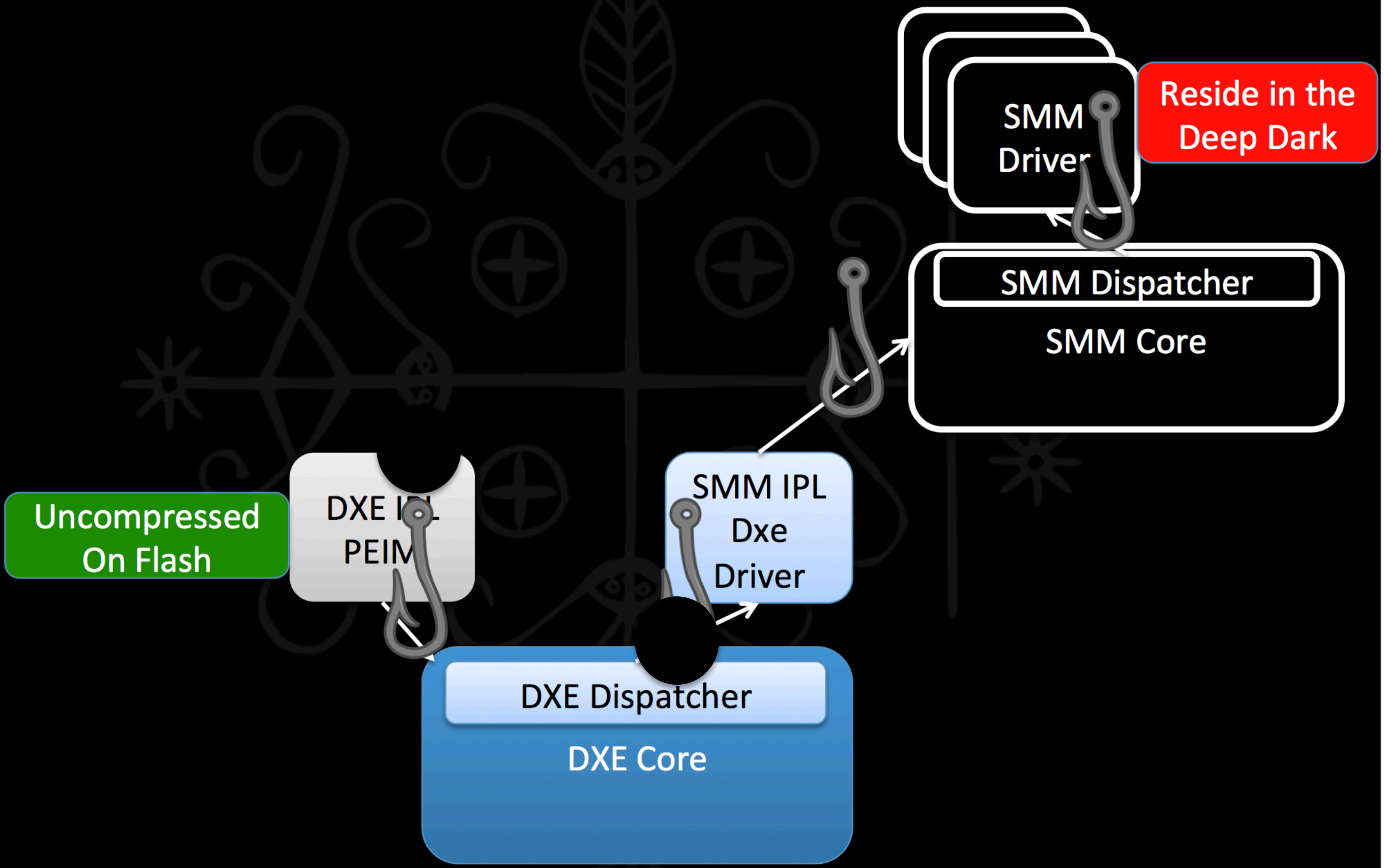
(that all vendors just add their own meat to)



Minimal hook paths in UEFI



Minimal hook paths in UEFI



LightEater on ASUS

- Uses hook-and-hop from DXE IPL to SMM
- From SMM attacks Windows 10
- Gets woken up every time a process starts, prints information about the process

Evidence of Scalability of Infection

- We wanted to show that the code an attacker wants to find can easily be identified with *simple and stupid* byte signatures
- Only took a couple days to develop

Example: DXE to BDS transition

- EDK open source code for DXE -> BDS transition
- DxeMain.c

- Equivalent exact assembly found in *6 separate vendors' BIOSes*

```
//
// Transfer control to the BDS Architectural Protocol
//
gBds->Entry (gBds);
```

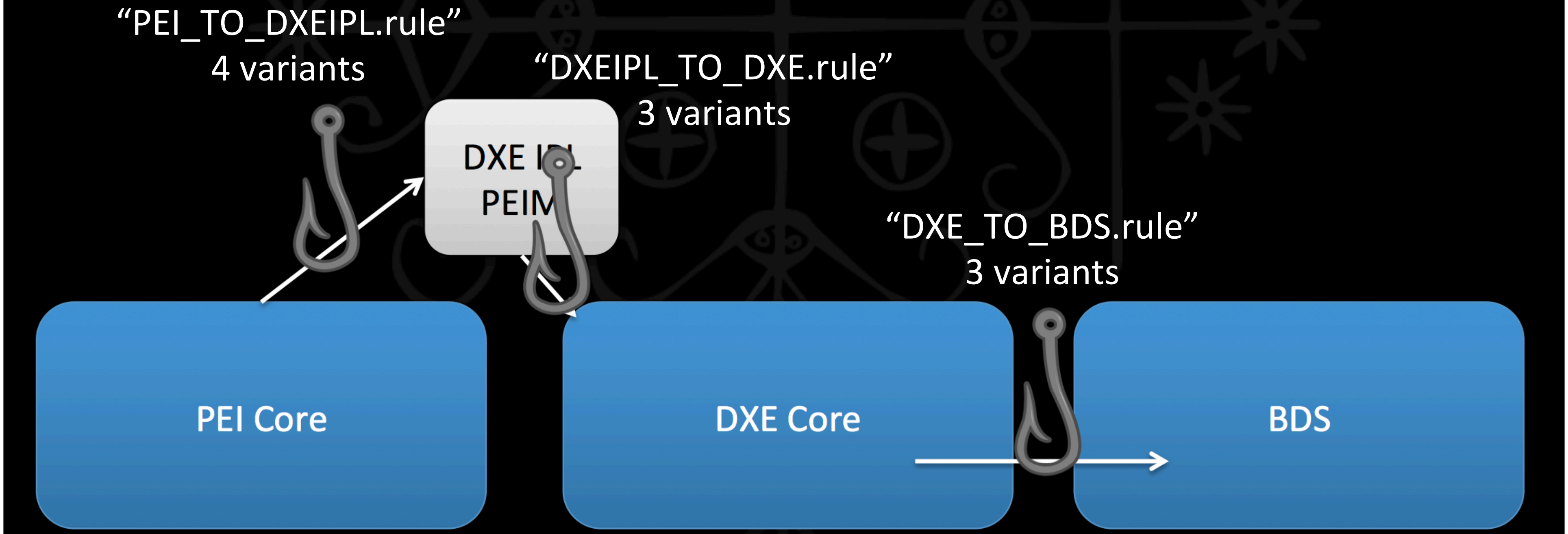
```
4C 8B 1D 8A AF 00 00   mov     r11, cs:gBDS
49 8B CB               mov     rcx, r11
41 FF 13               call    qword ptr [r11]
```

Yara rule = {4C 8B 1D [4] 49 8B CB 41 FF 13}

(yes, I know, I obviously should technically make it register-independent, but I don't care because it worked well enough as you'll see in a second :P)

Analysis targets

- Created YARA signatures from what the code looked like on 9 systems
- Key for next slides: “1,1,2” = “PEI_TO_DXEIPL variant 1, DXEIPL_TO_DXE variant 1, and DXE_TO_BDS variant 2 matched for this system”



Some Analysis Results

- Teddy Reed graciously provided the data set from his 2014 Infiltrate talk[1]
- 2158 BIOS images spidered from Lenovo, HP, Dell, Gigabyte, & Asrock’s websites
  - Haven’t counted how many individual models yet
- Signature scanning results:
  - PEI_TO_DXEIPL: 3 misses (1 model)
  - DXEIPL_TO_DXE: 0 misses
  - DXE_TO_BDS: 4 misses (2 models)

[1] “Analytics, and Scalability, and UEFI Exploitation (Oh My)” – Teddy Reed
http://prosauce.org/storage/slides/Infiltrate2014-Analytics-Scalability-UEFI-Exploitation.pdf

For reading at your leisure

(from Teddy Reed's data set)

(2158 images, 7 misses)

Lenovo (442 images)
- PEI_TO_DXEIPL: 0 misses
- DXEIPL_TO_DXE: 0 misses
- DXE_TO_BDS: 2 misses

HP (388 images)
- PEI_TO_DXEIPL: 0 misses
- DXEIPL_TO_DXE: 0 misses
- DXE_TO_BDS: 0 misses

Dell (381 images)
- PEI_TO_DXEIPL: 3 misses
- DXEIPL_TO_DXE: 0 misses
- DXE_TO_BDS: 2 misses

Gigabyte (347 images)
- PEI_TO_DXEIPL: 0 misses
- DXEIPL_TO_DXE: 0 misses
- DXE_TO_BDS: 0 misses

Asrock (596 images)
- PEI_TO_DXEIPL: 0 misses
- DXEIPL_TO_DXE: 0 misses
- DXE_TO_BDS: 0 misses

For reading at your leisure

(from a completely different LegbaCore data set)
(1003 images, 5 misses)

Lenovo (213 images)
- PEI_TO_DXEIPL: 0 misses
- DXEIPL_TO_DXE: 0 misses
- DXE_TO_BDS: 0 misses

HP (401 images)
- PEI_TO_DXEIPL: 0 misses
- DXEIPL_TO_DXE: 0 misses
- DXE_TO_BDS: 0 misses

Dell (348 images)
- PEI_TO_DXEIPL: 0 misses
- DXEIPL_TO_DXE: 0 misses
- DXE_TO_BDS: 0 misses

LG (13 images)
- PEI_TO_DXEIPL: 0 misses
- DXEIPL_TO_DXE: 0 misses
- DXE_TO_BDS: 1 misses

Asus (13 images)
- PEI_TO_DXEIPL: 2 misses
- DXEIPL_TO_DXE: 0 misses
- DXE_TO_BDS: 2 misses

Acer (15 images)
- PEI_TO_DXEIPL: 0 misses
- DXEIPL_TO_DXE: 0 misses
- DXE_TO_BDS: 0 misses

HP Example

HP EliteBook 750 G1 Notebook PC (ENERGY STAR)

1,1,2

HP EliteBook 755 G2 Notebook PC (ENERGY STAR)

1,1,2

HP EliteBook 740 G1 Notebook PC (ENERGY STAR)

1,1,2

HP EliteBook 745 G2 Notebook PC (ENERGY STAR)

1,1,2

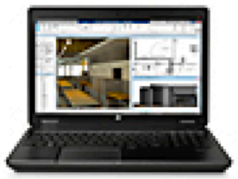
HP EliteBook 720 G1 Notebook PC (ENERGY STAR)

1,1,2

HP EliteBook 725 G2 Notebook PC (ENERGY STAR)

1,1,2

HP ZBook 15 G2 Mobile Workstation (ENERGY STAR)



3,1,3

HP ZBook 17 G2 Mobile Workstation (ENERGY STAR)

3,1,3

HP EliteBook 840 G1 Notebook PC (ENERGY STAR)

1,1,2

HP EliteBook 850 G1 Notebook PC (ENERGY STAR)

1,1,2

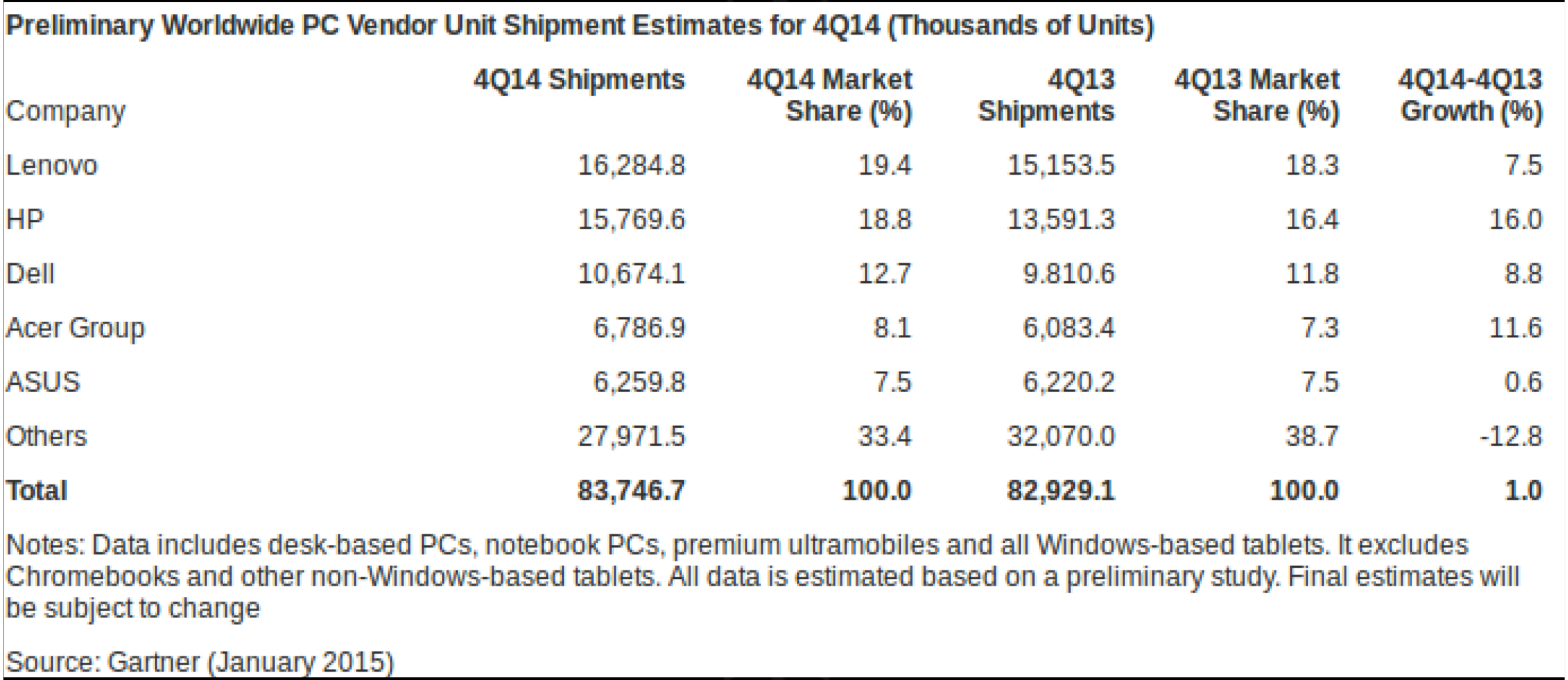
HP ZBook 14 Mobile Workstation (ENERGY STAR)

1,1,2

HP EliteBook 820 G2 Notebook PC (ENERGY STAR)

1,1,2

Extrapolation to millions



Preliminary Worldwide PC Vendor Unit Shipment Estimates for 4Q14 (Thousands of Units)

Company	4Q14 Shipments	4Q14 Market Share (%)	4Q13 Shipments	4Q13 Market Share (%)	4Q14-4Q13 Growth (%)
Lenovo	16,284.8	19.4	15,153.5	18.3	7.5
HP	15,769.6	18.8	13,591.3	16.4	16.0
Dell	10,674.1	12.7	9.810.6	11.8	8.8
Acer Group	6,786.9	8.1	6,083.4	7.3	11.6
ASUS	6,259.8	7.5	6,220.2	7.5	0.6
Others	27,971.5	33.4	32,070.0	38.7	-12.8
Total	**83,746.7**	**100.0**	**82,929.1**	**100.0**	**1.0**

Notes: Data includes desk-based PCs, notebook PCs, premium ultramobiles and all Windows-based tablets. It excludes Chromebooks and other non-Windows-based tablets. All data is estimated based on a preliminary study. Final estimates will be subject to change

Source: Gartner (January 2015)

From https://www.gartner.com/newsroom/id/2960125

Extrapolation to millions

- So if almost no one applies BIOS vulnerability patches…
- And if my tiny set of signatures can reliably find hook points and disable sanity checks in the machines HP is currently selling…
- And if HP shipped ~15M PCs in Q4 2014…
- Then we would understand that millions of these BIOSes could be reliably infected, yes?

Acer

Aspire S7-392

2,2,1

TravelMate B113

2,2,1

TravelMate P245

2,2,1

Veriton Z26600G

2,2,1

TravelMate P255

2,2,1

TravelMate P455

2,2,1

TravelMate P645

2,2,1

Veriton Z4810G

2,2,1

Veriton M4630G

2,2,1

Veriton X2630G

2,2,1

Veriton M2631

2,2,1

Veriton N4630G

2,2,1

Asus

ASUSPRO ADVANCED B53S

miss,2,1

BU201

2,2,1

B451JA

2,2,1

B400A

2,2,1

ASUSPRO ESSENTIAL P53E

miss,2,1

PU551JH

2,2,1

P751JA

2,2,1

P55VA

2,2,1

ESC2000 G2

2,2,1

ESC4000 G2

2,2,1

TS500-E8-PS4

4,2,miss

RS500-E8-RS4

4,2,miss

LG

PC Gram 13Z940

2,1,1

PC Gram 14Z950

4,1,3

PC Gram 15Z950

4,1,3

Ultra PC 14U530

2,1,1

Ultra PC 15U530

2,1,1

Ultra PC 15U340

4,1,1

15N540

2,1,1

22V240

4,1,1

Tab Book 10T550

4,1,miss

Tab Book 11T740

2,1,1

A75CV

2,2,1

A75PS

2,2,1

It was about this time I got really tired of making these slides and manually downloading BIOSes ;)

A little good news before we go

- Were working with vendors like Dell to do security assessments to find and fix issues before they ship on new systems. Lenovo and others are also on the list.

A little good news before we go

- We're also working with Intel to try to create the first commercial-grade SMM isolation
- Intel has the ability for their hardware virtualization to jail SMM
- We will then work with BIOS vendors to incorporate the technology into shipping systems
- End result will be that even if attackers break into SMM, they can't read/write arbitrary memory
  - And we could detect attackers through measurements.

Conclusions

THE DARK DIMENSION'S GRIP IS TIGHT.
PAPA LEGBA SEES ME THROUGH-- BARELY.

This talk has 2 main points

- Because almost no one applies BIOS patches, almost every BIOS in the wild is affected by *at least* one vulnerability, and can be infected
- The high amount of code reuse across UEFI BIOSes means that BIOS infection is automatable and reliable

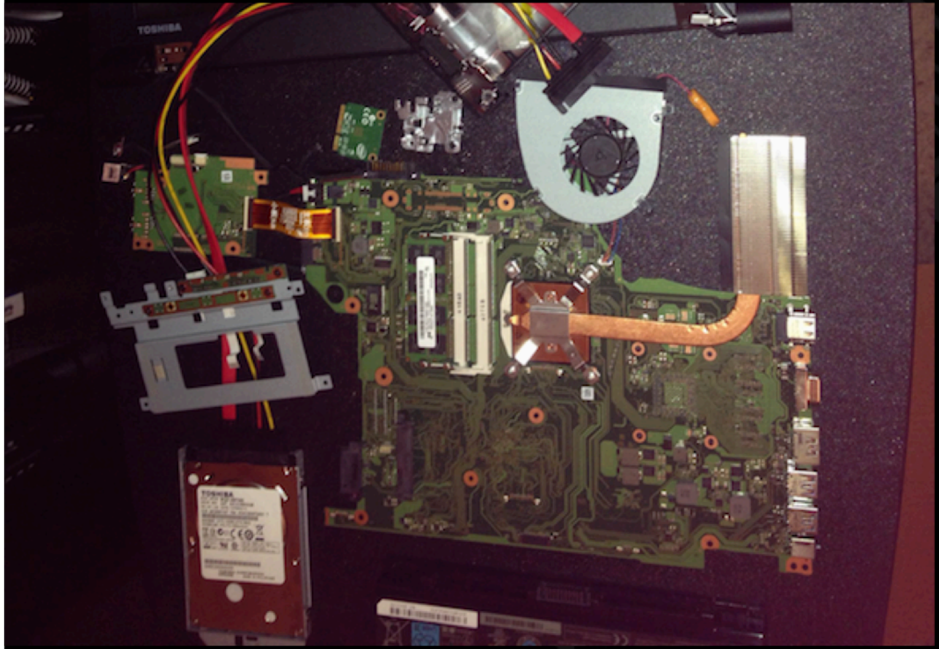
What we showed

- All systems we have looked at contain Incursion vulnerabilities that allow breaking in to SMM
- Incursion vulnerabilities can be found programatically
- The LightEater SMM attacker can perform *any attack* that is available to lesser attackers
  - We showed stealing GPG keys/messages from memory (on MSI), data exfiltration over the network (on HP), Windows kernel rootkit behavior (on Asus)
- Showed how a physical BIOS attack can be done in 2 minutes by an unskilled accomplice (maid/border guard)
- Homogeneity in UEFI BIOSes for the things an attacker cares about. Creating signatures from ~10 BIOSes is sufficient to find matches on thousands of images (which relate to millions of shipped machines)

Conclusions

- 2 guys + 4 weeks + $2k = Multiple vendors' BIOSes infected, with multiple infection capabilities
- One hand (purposely) tied behind our backs: *Didn't use special debug hardware. Serial prints only!*
- Do you really think that Five Eyes are the only ones who have developed capabilities in this space?
- "Absence of evidence is not evidence of absence"
- *It's time to start checking your firmware*
  - Stop giving firmware attackers a free pass and indefinite invisibility
- *It's time to start patching your BIOSes*
  - Demand the capability from your patch management software
- *It's time to demand better BIOS security from your OEM*
  - We'll eventually make a name-and-shame list of vendors who are perennially leaving their customers open to BIOS attacks

Pour a 40 on the curb for the PCs we've lost…



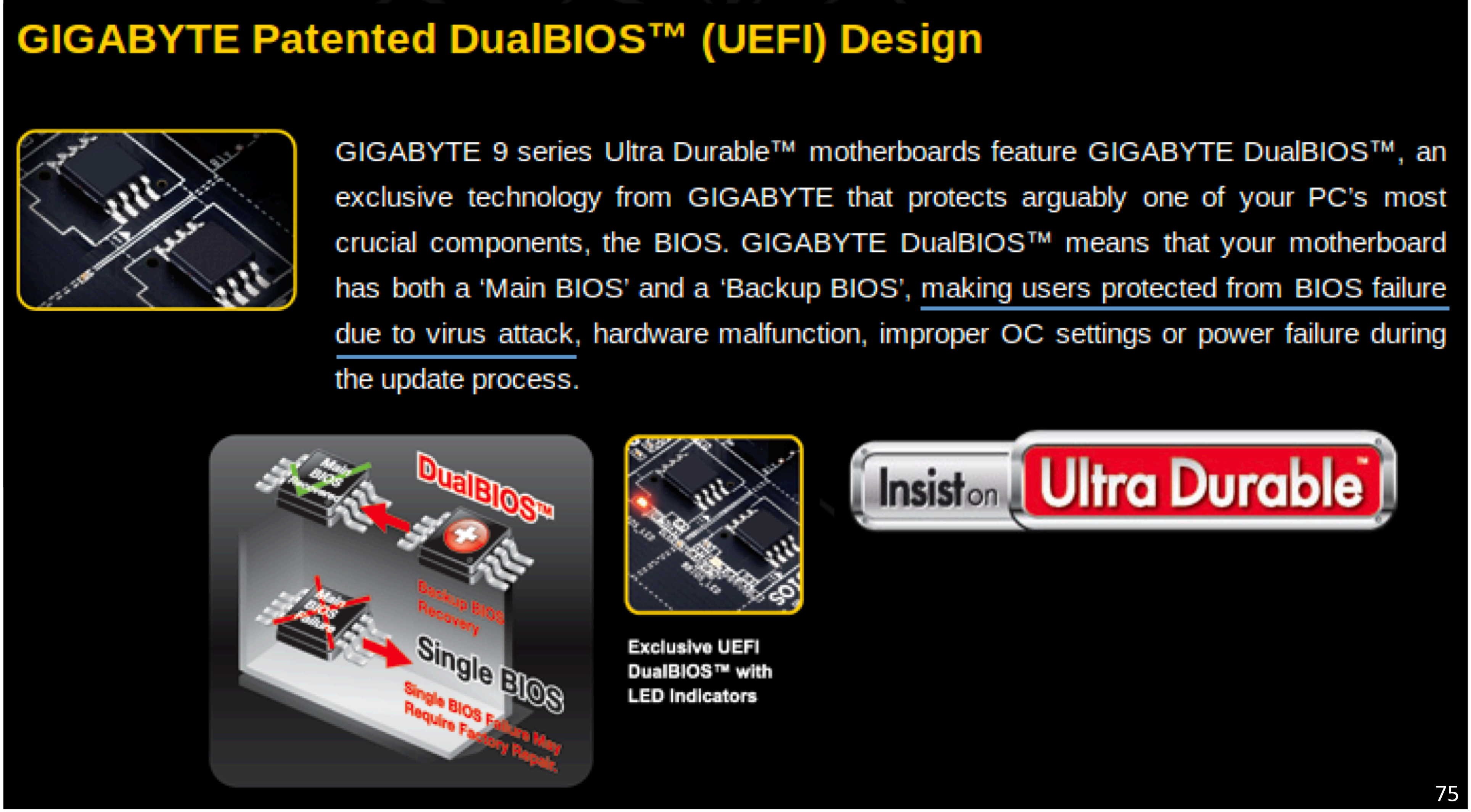
Toshiba Tecra…
Short circuited during disassembly
Rest in pieces buddy

Contact

- Twitter: @coreykal, @xenokovah, @legbacore
- Email: {corey, xeno}@legbacore.com
- http://legbacore.com/Contact.html for our GPG keys

- As always, go check out OpenSecurityTraining.info for the free classes from Corey and Xeno on x86 assembly & architecture, binary executable formats, stealth malware, and exploits.
- Then go forth and do cool research for us to read about!

Throwaway Demo



GIGABYTE Patented DualBIOS™ (UEFI) Design

GIGABYTE 9 series Ultra Durable™ motherboards feature GIGABYTE DualBIOS™, an exclusive technology from GIGABYTE that protects arguably one of your PC's most crucial components, the BIOS. GIGABYTE DualBIOS™ means that your motherboard has both a 'Main BIOS' and a 'Backup BIOS', making users protected from BIOS failure due to virus attack, hardware malfunction, improper OC settings or power failure during the update process.

Exclusive UEFI DualBIOS™ with LED Indicators

Verdict

--THEY ARE VULNERABLE TO THE LANGUAGE OF NULL.
RAK-AK-AK-AK
AK-AK-AK-AK

References

[0] Attacking Intel BIOS – Rafal Wojtczuk and Alexander Tereshkin, July 2009
http://invisiblethingslab.com/resources/bh09usa/Attacking%20Intel%20BIOS.pdf

http://www.kb.cert.org/vuls/id/127284 (CERT never posted?!)

[1] Defeating Signed BIOS Enforcement – Kallenberg et al., Sept. 2013
http://conference.hitb.org/hitbsecconf2013kul/materials/D1T1%20-%20Kallenberg,%20Kovah,%20Butterworth%20-%20Defeating%20Signed%20BIOS%20Enforcement.pdf

http://www.kb.cert.org/vuls/id/912156

http://www.kb.cert.org/vuls/id/255726 (CERT hasn't posted yet despite request)

[2] All Your Boot Are Belong To Us (MITRE portion) – Kallenberg et al. – Mar. 2014, delayed from publicly disclosing potential for bricking until HITB at Intel's request
https://cansecwest.com/slides/2014/AllYourBoot_csw14-mitre-final.pdf

http://www.kb.cert.org/vuls/id/758382

[3] All Your Boot Are Belong To Us (Intel portion) – Bulygin et al. – Mar. 2014
https://cansecwest.com/slides/2014/AllYourBoot_csw14-intel-final.pdf

[4] Setup for Failure: Defeating UEFI Secure Boot - Kallenberg et al., Apr. 2014
http://syscan.org/index.php/download/get/6e597f6067493dd581eed737146f3afb/SyScan2014_CoreyKallenberg_SetupforFailureDefeatingSecureBoot.zip

http://www.kb.cert.org/vuls/id/291102 (CERT hasn't posted yet despite request)

References

[5] Extreme Privilege Escalation on UEFI Windows 8 Systems – Kallenberg et al., Aug. 2014
https://www.blackhat.com/docs/us-14/materials/us-14-Kallenberg-Extreme-Privilege-Escalation-On-Windows8-UEFI-Systems.pdf

http://www.kb.cert.org/vuls/id/766164

[6] Attacks against UEFI Inspired by Darth Venamis and Speed Racer – Wojtczuk & Kallenberg, Dec. 2013
https://bromiumlabs.files.wordpress.com/2015/01/attacksonuefi_slides.pdf
http://www.kb.cert.org/vuls/id/533140

[7] Speed Racer: Exploiting an Intel Flash Protection Race Condition – Kallenberg & Wojtczuk, Dec. 2013
https://frab.cccv.de/system/attachments/2565/original/speed_racer_whitepaper.pdf

http://www.kb.cert.org/vuls/id/912156

[8] Attacking UEFI Boot Script – Wojtczuk & Kallenberg, Dec. 2013

https://frab.cccv.de/system/attachments/2566/original/venamis_whitepaper.pdf

http://www.kb.cert.org/vuls/id/552286

[9] "Snorlax" bug – Cornwell, et al., Dec. 2013

https://frab.cccv.de/system/attachments/2566/original/venamis_whitepaper.pdf

http://www.kb.cert.org/vuls/id/577140 (CERT hasn't posted yet despite request)

References

[10] Deeper Door – Embleton & Sparks, Jul. 2008 – http://clearhatconsulting.com/DeeperDoor

[11] Can you still trust your network card? - Duflot, et al., Mar. 2010 - http://www.ssi.gouv.fr/uploads/IMG/pdf/csw-trustnetworkcard.pdf

[12] The Jedi Packet Trick takes over the Deathstar - Arrigo Triulzi, Mar. 2010 www.alchemistowl.org/arrigo/Papers/Arrigo-Triulzi-CANSEC10-Project-Maux-III.pdf

[13] "Mebromi: the first BIOS rootkit in the wild" http://www.webroot.com/blog/2011/09/13/mebromi-the-first-bios-rootkit-in-the-wild/

[14] "NSA Speaks Out on Snowden Spying", Dec 2012 http://www.cbsnews.com/news/nsa-speaks-out-on-snowden-spying/

[15] "To Protect And Infect" - Jacob Applebaum, Dec. 2012 https://www.youtube.com/watch?v=vILAlhwUgIU (contains leaked classified NSA documents)

[16] "U.S. Gas, Oil Companies Targeted in Espionage Campaigns", Jan. 2013 http://threatpost.com/u-s-gas-oil-companies-targeted-in-espionage-campaigns/103777

References

[X] See all the related work we're aware of here: http://timeglider.com/timeline/5ca2daa6078caaf4

Backup

- “Should you worry when the skullhead is in front of you? Or is it worse because it’s always waiting, where your eyes don’t go?”
  - They Might Be Giants