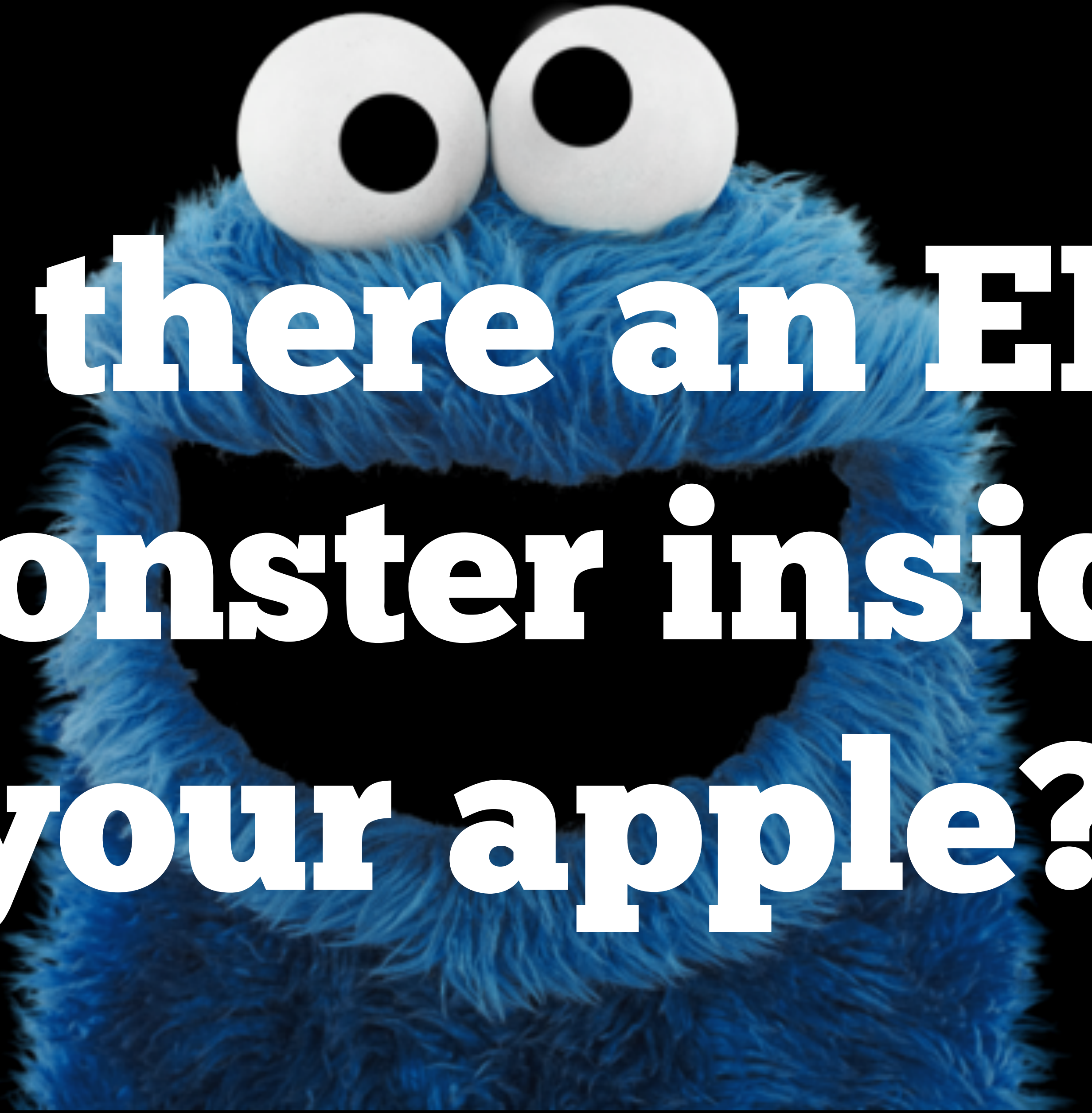


Is there an EFI monster inside your apple?

fG! @ CODE BLUE 2015

Who am I?

- An Economist.
- Who loves Human Behavior.
- And politics.
- Oh, and a bit of computers.

SentinelOne

Translators!

Whats Up Doc?

EFI Monsters?

- Introduction to EFI.
- How to
  - Reverse engineer (U)EFI binaries.
  - Search for (U)EFI rootkits.

WE LIVE IN A CHANGING WORLD

ASSUMPTIONS

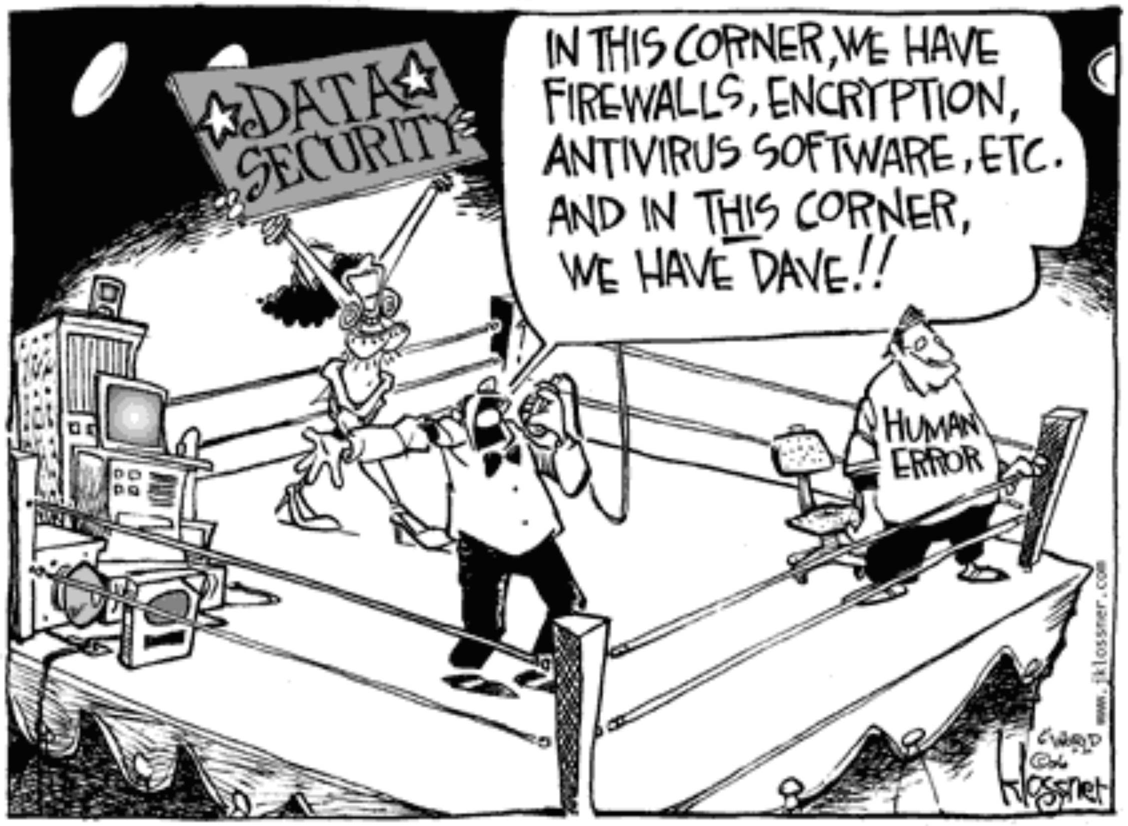
ROAD OUT
LANDSLIDE
Ted Key

"Relax! I know this road perfectly! I've been driving it all my life!"

Assumptions

- Reference machine
  - MacBook Pro Retina 10,1.
- 64-bit only OS X versions.
- Sandy Bridge or newer.

Why EFI?


DATA SECURITY
IN THIS CORNER, WE HAVE FIREWALLS, ENCRYPTION, ANTIVIRUS SOFTWARE, ETC. AND IN THIS CORNER, WE HAVE DAVE!!
HUMAN ERROR
www.jklossner.com
©WORLD ©06 Klossner

Why EFI?

- BIOS replacement.
- Initially developed by Intel.
  - http://www.intel.com/content/www/us/en/architecture-and-technology/unified-extensible-firmware-interface/efi-specifications-general-technology.html
- Now UEFI, managed by UEFI consortium.
  - http://www.uefi.org

Why EFI?

- Initializes your machine.
- Access to low level features.
- Modular.
- Feature rich.
- Rather easy development in C.

What evil things can we do?

What evil things can we do?

- Diskless kernel/userland rootkits
- Rootkit data stored in the flash chip.
- Unpack and patch kernel on boot.
- RAM only, never touch hard-disk.
- Check Snare's SyScan 2012 presentation.

What evil things can we do?

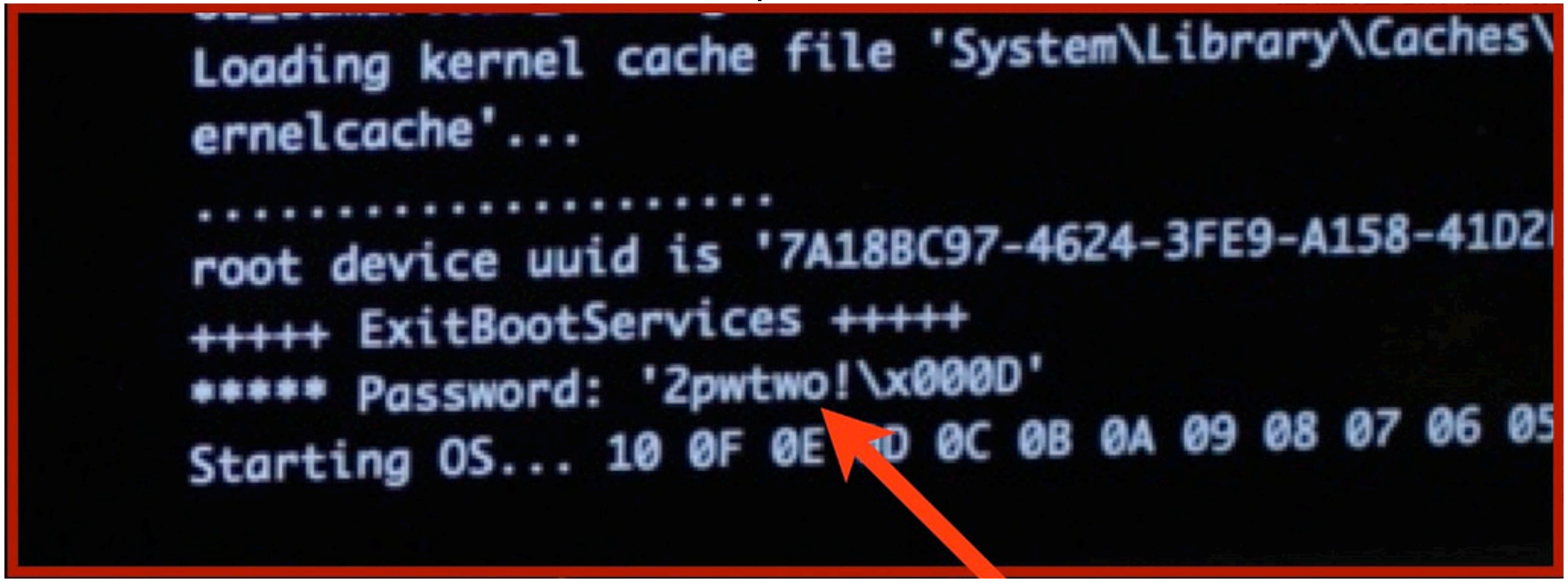
- Can be hard to detect.
- With regular available tools.
- And with some anti-forensics.
- For example anti-memory dumping.

What evil things can we do?

- Persistence across operating system installs
- HackingTeam built a UEFI rootkit.
  - https://github.com/hackedteam/vector-edk
  - https://github.com/informationextraction/vector-edk/blob/master/MdeModulePkg/Application/fsbg/fsbg.c

What evil things can we do?

- Attack full-disk encryption
- Install a keylogger.
- Recover FileVault2 password.



```
Loading kernel cache file 'System\Library\Caches\
ernelcache'...
.......................
root device uuid is '7A18BC97-4624-3FE9-A158-41D2
++++++ ExitBootServices ++++++
****** Password: '2pwtwo!\x000D'
Starting OS... 10 0F 0E 0D 0C 0B 0A 09 08 07 06 05
```

What evil things can we do?

- Attack “secure” operating systems
- For example, Tails.
- Recover PGP keys and/or passphrases.
- https://www.youtube.com/watch?v=sNYsfUNegEA.

What evil things can we do?

- Bootloader
  - Redirect to a custom bootloader.
- SMM backdoors
  - http://blog.cr4.sh/2015/07/building-reliable-smm-backdoor-for-uefi.html

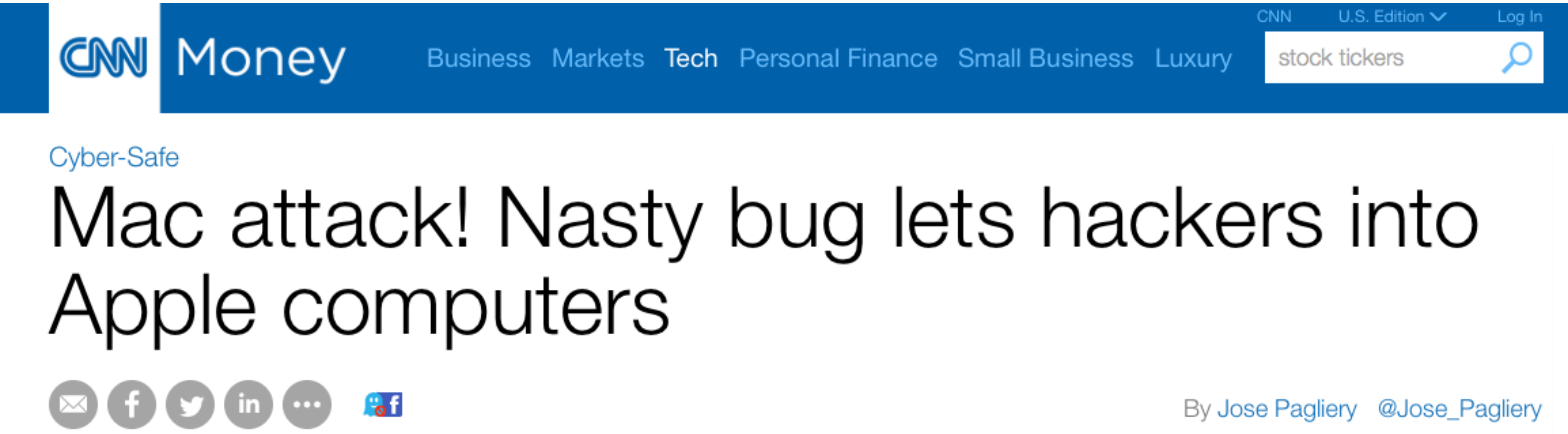
TL;DR

OWN EVERYTHING!

Once upon a time…

there was a...

a zero day!

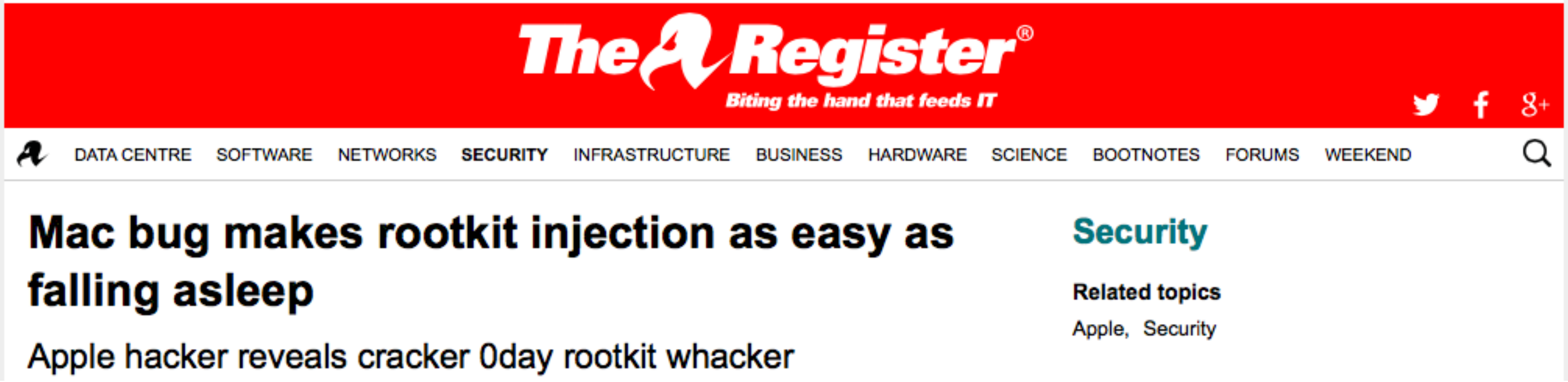


CNN Money — Business · Markets · Tech · Personal Finance · Small Business · Luxury — CNN · U.S. Edition · Log In — stock tickers

Cyber-Safe

Mac attack! Nasty bug lets hackers into Apple computers

By Jose Pagliery @Jose_Pagliery



The Register® — Biting the hand that feeds IT

DATA CENTRE · SOFTWARE · NETWORKS · SECURITY · INFRASTRUCTURE · BUSINESS · HARDWARE · SCIENCE · BOOTNOTES · FORUMS · WEEKEND

Mac bug makes rootkit injection as easy as falling asleep

Apple hacker reveals cracker 0day rootkit whacker

Security

Related topics

Apple, Security

A zero day story...

- Firmware related zero day.
- Disclosed a few months ago.
  - https://reverse.put.as/2015/05/29/the-empire-strikes-back-apple-how-your-mac-firmware-security-is-completely-broken/

A zero day story...

- Failure to lock the flash.
- Write to the flash from userland.
- Similar to Thunderstrike but better.
- Thunderstrike requires physical access.
- Prince Harming allows remote attack.

PERSISTENCE
FIRMWARE FLASH

- Hardware-specific, but it's always there
- Can modify everything
  - SEC, PEI, DXE, BDS, custom drivers, whatever
- Can be written to from the OS
- So awesome. 11/10 A++++ would buy again.

A zero day story…

- Extremely simple to trigger.
- Put machine to sleep.
  - Close, wait for fans to stop, and reopen.
  - Or force sleep with "pmset sleepnow".

A zero day story…

- Sandy Bridge and Ivy Bridge Macs are vulnerable.
- Haswell or newer are not.
- All older machines are vulnerable
  - Core 2 Duo or older.
  - No flash protections at all.

A zero day story…

- Available updates:

MacBook Air	MacBook Pro	Mac Mini	Mac Pro	iMac
4,1	8,1	5,1	6,1	12,1
5,1	9,1	6,1		13,1
6,1	10,1	7,1		14,1
7,1	10,2			14,2
	11,1			14,3
	11,2			14,4
	11,4			15,1
	12,1			

A zero day story…

- Reversing and understanding the vulnerability.
    - https://reverse.put.as/2015/07/01/reversing-prince-harmings-kiss-of-death/
- Contains links to relevant EFI documentation.

A zero day story…

- Venamis aka Dark Jedi was also patched.
  - http://events.ccc.de/congress/2014/Fahrplan/events/6129.html
  - http://blog.cr4.sh/2015/02/exploiting-uefi-boot-script-table.html
- Slightly more complex, same results.

A zero day story...

- The story doesn't end here.
- Check ThunderStrike 2 slides.
- Other unpatched vulnerabilities.
- Can be exploited with remote attack vectors.

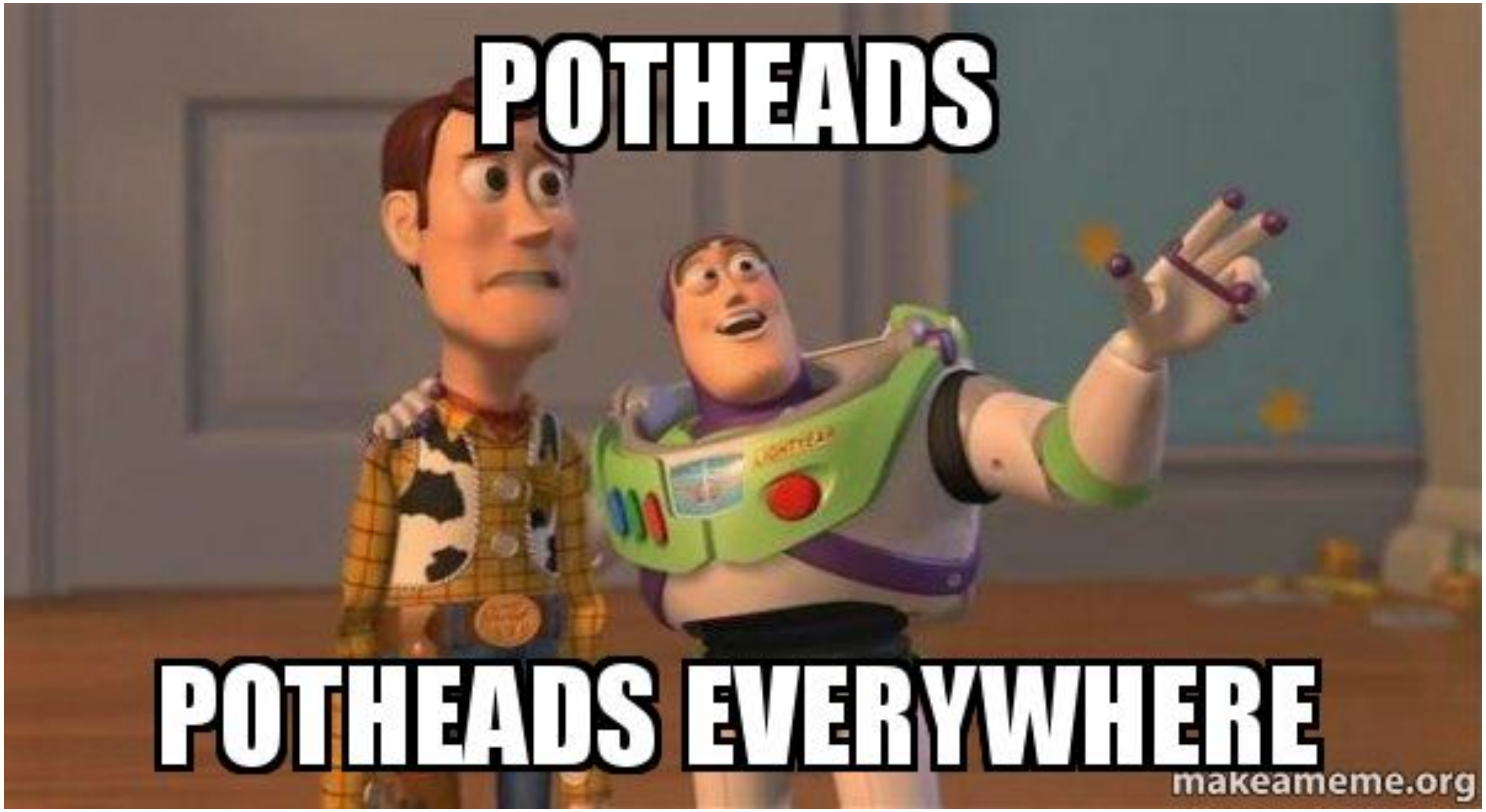
Old bugs, new platforms

Vulnerability	Private disclosure Public disclosure	Status on OSX
Snorlax/PrinceHarming VU #577140	August 2013 July 2015 / May 2015	Patched June 2015
Darth Venamis VU #976132	Sept 2014 Dec 2014	Partial Patch June 2015
SpeedRacer/BIOS_CTNL VU #766164	Dec 2013 Aug 2014	Vulnerable
King's Gambit VU #552286	Dec 2013 Aug 2014	Vulnerable (See HITB-GSEC 2015)
The Sicilian VU #255726	~May 2013 Sep 2013	Vulnerable
Setup UEFI Variable VU #758382	June 2013 Mar 2014	Not vulnerable

Reminder: This talk has 1 main point

- Apple has not been as responsive, or as accurate, as other PC vendors in responding to industry-wide notifications of firmware vulnerabilities. Consequently Mac users have been left vulnerable to attacks that have been fixed on other x86-based PCs.

Apple ...



Where is EFI?

Where is EFI?

- Usually stored in a CMOS serial flash.
- Two popular chips
  - Macronix MX25L6406E.
  - Micron N25Q064A.
- SPI compatible.
- Most are 64 Mbits/8 Mbytes.

Where is EFI?

- Newer machines flash chip(s)
  - Winbond W25Q64FV.
- Chip list from EfiFlasher.efi:

SST 25VF080	Macronix 25L1605	ST Micro M25P16	WinBond 25X32
SST 25VF016	Macronix 25L3205	ST Micro M25P32	Winbond 25X64
SST 25VF032	Macronix 25L6436E	Eon M25P32	Winbond 25X128
SST 25VF064	Atmel 45DB321	Eon M25P16	Numonyx N25Q064

Where is EFI?

- Most chips are 8 pin SOIC.
- SMD or BGA versions used?
  - Retinas 13"?
  - New MacBook 12"?

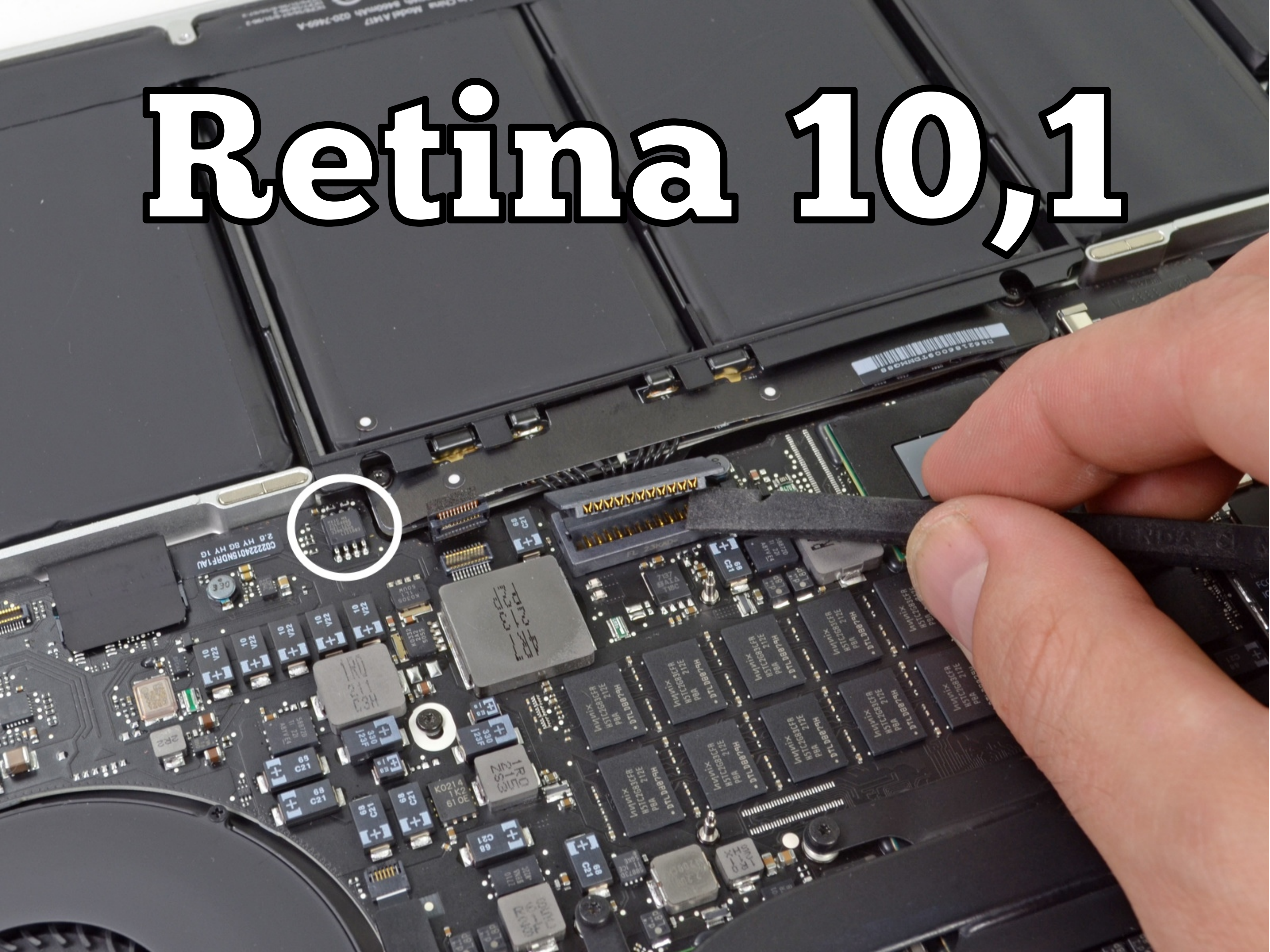
Where is EFI?

- You can buy the chips bulk and cheap.
- Useful for flashing experiments.
- Good results from Aliexpress.com.
- Around $14 for 10 N25Q064A.
- Around $8 for 10 MX25L640E.

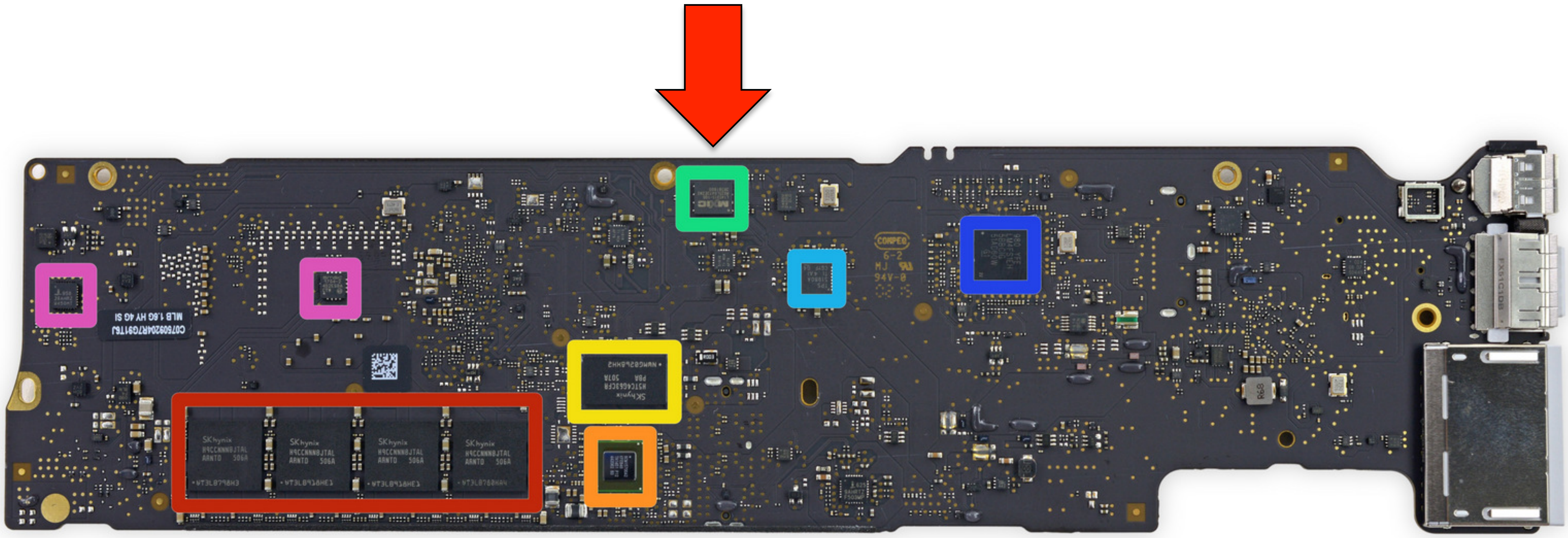
Where is EFI?

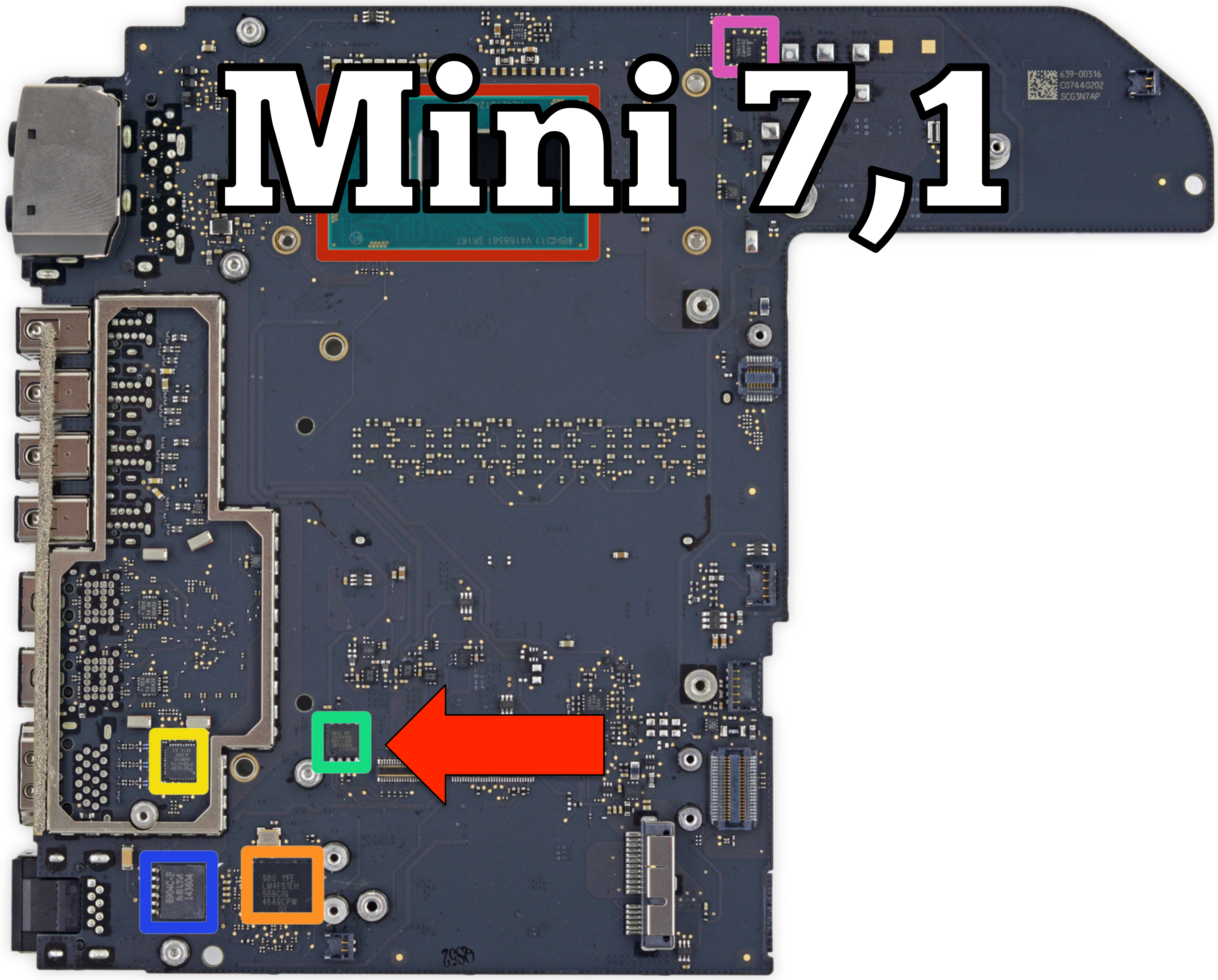
- Easy access on some models.
    - Retinas 15" are the easiest.
- Extensive disassembly required on others.
- Still, a MacBook Pro 8,1 can be disassembled in 5 mins or less.

Retina 10,1

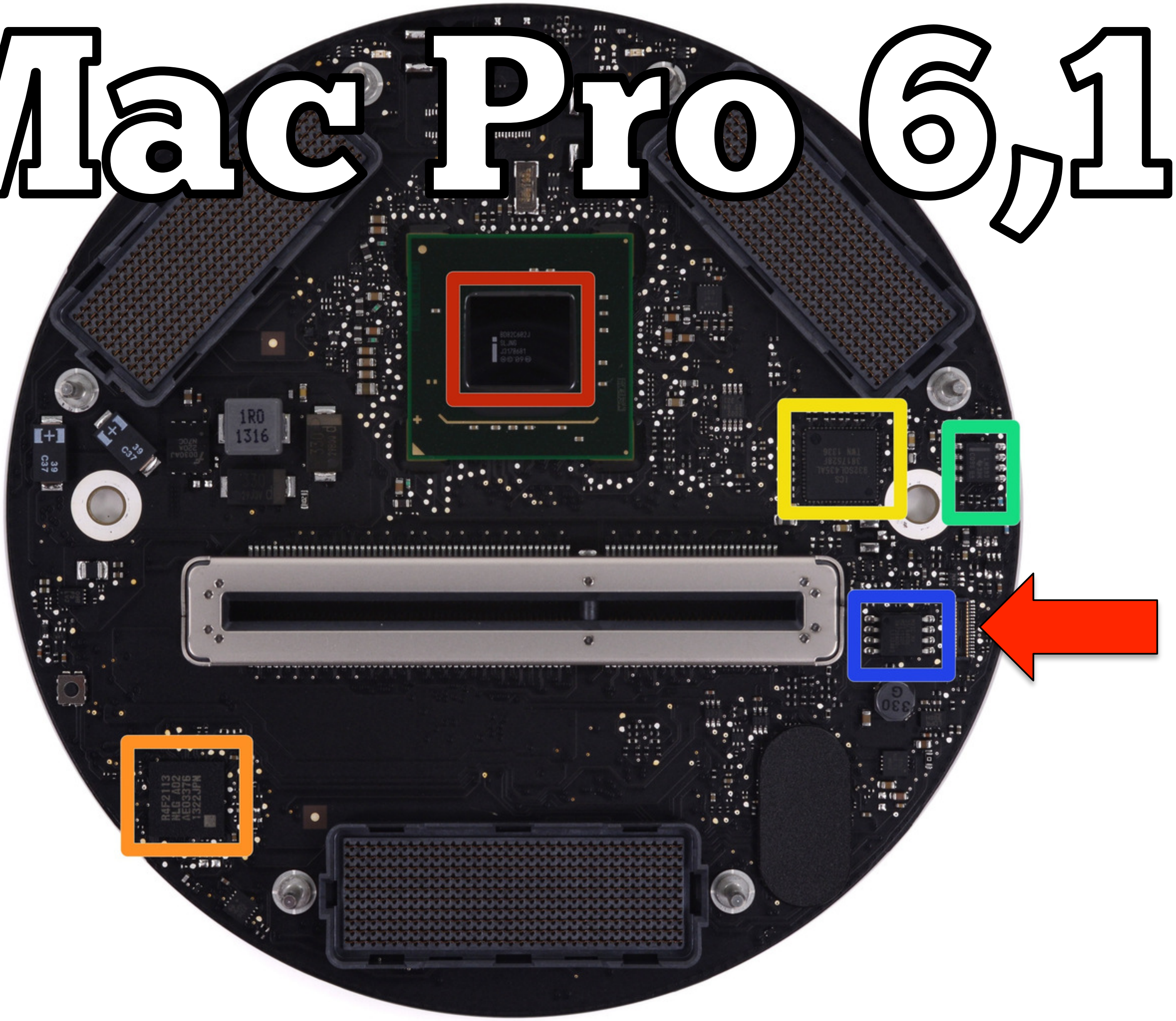


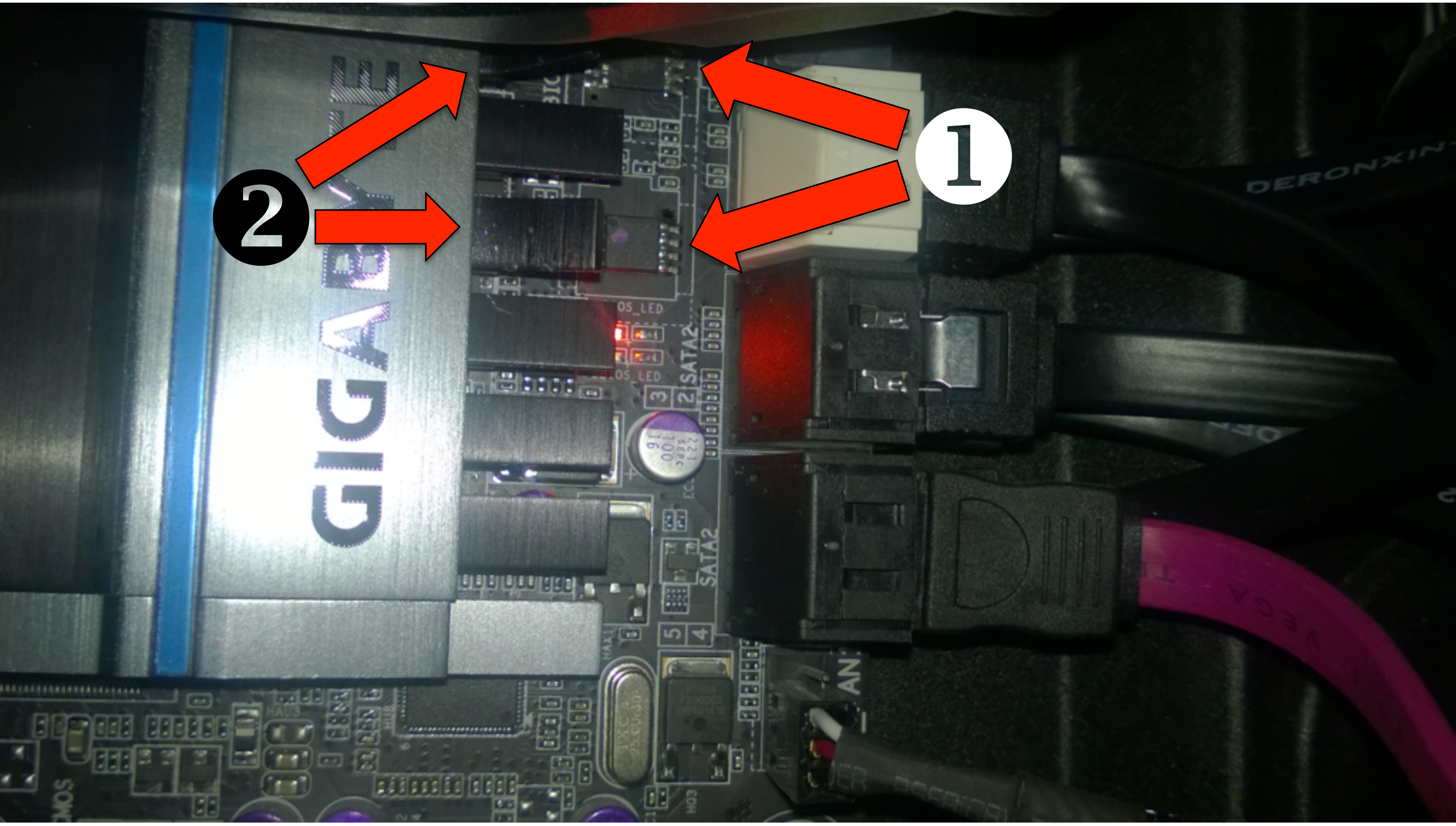
Air 7,2




Mini 7,1

Mac Pro 6,1




1
2
GIGABYTE
OS_LED
SATA2
SATA2

How to dump EFI

How to dump EFI

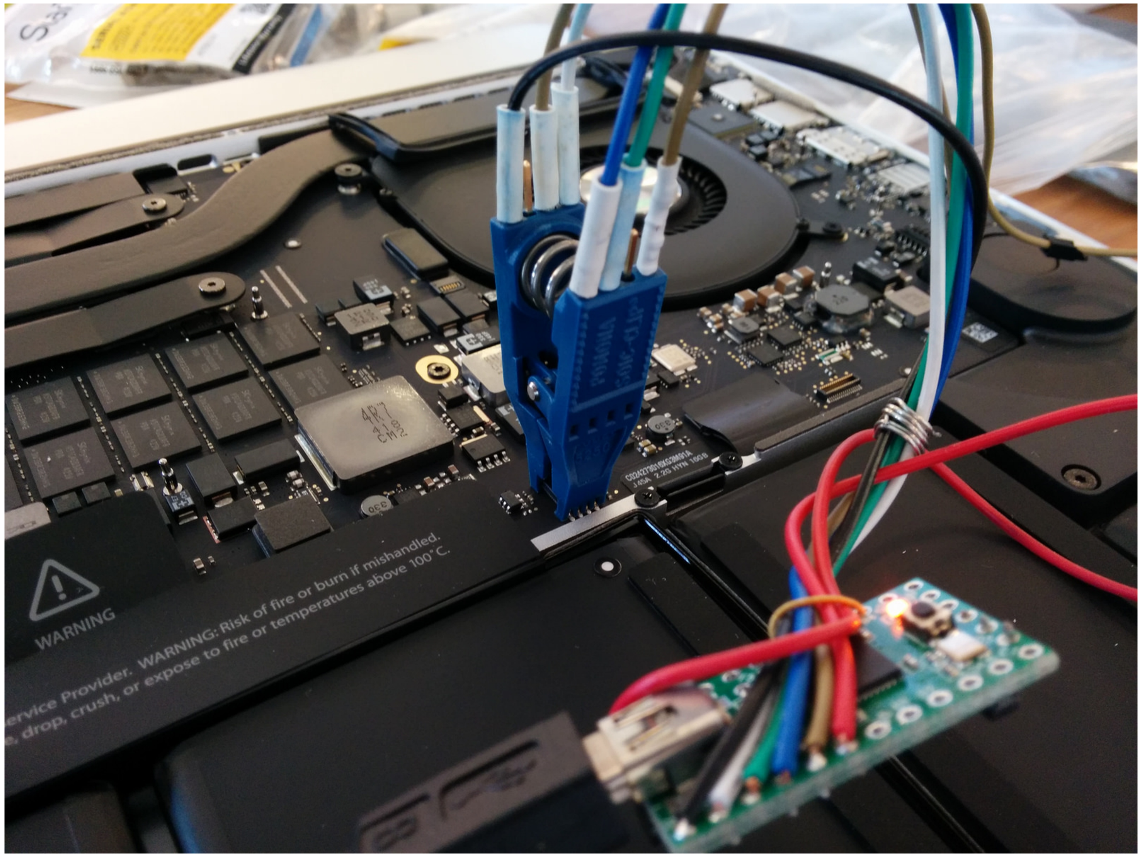
- Hardware
  - The best and most reliable way.
  - Trustable.
- Software
  - Possible if chip supported by flashrom.
  - Not (very) trustable.

Hardware

- Any SPI compatible programmer.
  - http://flashrom.org/Supported_programmers
- I use Trammell Hudson's SPI flasher.
  - https://trmm.net/SPI

Hardware

- Based on Teensy 2.0 or 3.x.


WARNING
ervice Provider. WARNING: Risk of fire or burn if mishandled.
, drop, crush, or expose to fire or temperatures above 100°C.

Hardware

- Easy to build.
- Cheap, ~ $30.
- Fast, dumps a 64Mbit flash in 8 mins.
- The Teensy 3 version is even faster.
- It just works!

Flash chip SPI pinout

```
                   +--------------+
 (WHITE) CS  -----| o            |----- VCC (RED)
                  |              |
 (BROWN) SO  -----|              |----- RST (YELLOW)
                  |              |
(ORANGE) WP  -----|              |----- SCLK (GREEN)
                  |              |
(BLACK) GND  -----|              |----- SI (BLUE)
                   +--------------+
```

Teensy 2.0 pinout

```
                     Teensy 2.0
                  +--+-------+--+
 (BLACK) GND -----|o |       | o|----- VCC (RED)
                  |  |  USB  |  |
  (WHITE) CS -----|o |       | o|
                  |  +-------+  |
(GREEN) SCLK -----|o           o|
                  |             |
   (BLUE) SI -----|o           o|
                  |             |
  (BROWN) SO -----|o           o|
                  |             |
                  |o +-------+ o|
                  |  |       |  |
                  |o |       | o|
                  |  +-------+  |
                  |o           o|
                  |             |
                  |o           o|
                  |             |
                  |o           o|
                  |    +---+    |
                  |o   | 0 |   o|
                  |    +---+    |
                  |o o o o o o o|
                  +----+--------+
                       |
                      VCC
                (YELLOW/ORANGE)
```

Teensy 2.0 pinout

- Teensy 2 default voltage is 5v.
- Flash chips are 3.3.v.
- Requires voltage regulator MCP1825.
- https://www.pjrc.com/store/mcp1825.html

Teensy 3.1 pinout

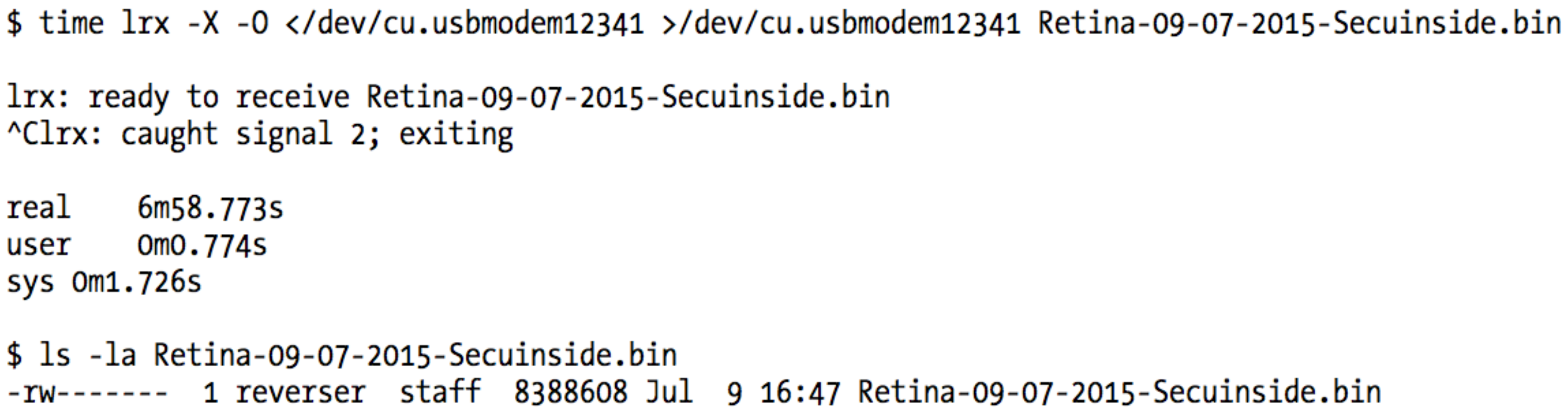
```
                    Teensy 3.1
                 +--+--------+--+
(BLACK) GND -----|o |        | o|
                 |  |  USB   |  |
                 |o |        | o|
                 |  +--------+  |
                 |o            o|----- VCC (RED)
                 |              |
                 |o            o|
                 |              |
                 |o            o|
                 |              |
                 |o +--------+ o|
                 |  |        |  |
                 |o |        | o|
                 |  +--------+  |
                 |o            o|
                 |              |
                 |o            o|
                 |              |
                 |o            o|
                 |     +----+   |
                 |o    | 0  |  o|
                 |     +----+   |
 (WHITE) CS -----|o            o|
                 |              |
 (BROWN) SO -----|o            o|
                 |              |
  (BLUE) SI -----|o o o o o o o |---- SCLK (GREEN)
                 +-----+--------+
                       |
                      VCC
                (YELLOW/ORANGE)
```

Tips & Tricks

- Shunt WP and RST pins to VCC.
- Different SPI pins names
  - SCLK, SCK, CLK.
  - MOSI, SIMO, SDO, DO, DOUT, SO, MTSR.
  - MISO, SOMI, SDI, DI, DIN, SI, MRST.
  - SS, nCS, CS, CSB, CSN, nSS, STE, SYNC.

Hardware

- How to read entire flash



```
$ time lrx -X -O </dev/cu.usbmodem12341 >/dev/cu.usbmodem12341 Retina-09-07-2015-Secuinside.bin

lrx: ready to receive Retina-09-07-2015-Secuinside.bin
^Clrx: caught signal 2; exiting

real    6m58.773s
user    0m0.774s
sys 0m1.726s

$ ls -la Retina-09-07-2015-Secuinside.bin
-rw-------  1 reverser  staff  8388608 Jul  9 16:47 Retina-09-07-2015-Secuinside.bin
```

Hardware

- How to write entire 64MB flash

```
spi
>Help:
i: print ID
r: read 16 bytes from address - r0<enter>
R: read XX bytes from address - R0 10<enter>
d: dump to console
w: write enable interactive
e: erase sector interactive
u: upload
b: upload bios area only
1: flash first ffs
2: flash second ffs
3: flash third ffs
x: download

u
>0 800000
(exit to shell)
# pv new-efi.bin > /dev/cu.usbmodem12341
```

Hardware

- Linux works best to write the flash.
- Some issues with OS X version.
- pv or serial driver issues?
    - http://www.ivarch.com/programs/pv.shtml

Software

- Requirements
  - Flashrom
  - DirectHW.kext
- Rwmem by Trammell also works.
- Or readphysmem.

Software

- DarwinDumper.
- Contains binary versions of flashrom and DirectHW.kext.
- Kernel extension is not code signed.
- (Still) Whitelisted by Apple.

Software

- http://flashrom.org/Flashrom
- http://www.coreboot.org/DirectHW
- https://bitbucket.org/blackosx/darwindumper/downloads
- https://github.com/osresearch/rwmem
- https://github.com/gdbinit/readphysmem

```
sh-3.2# kextload DirectHW.kext/

sh-3.2# ./flashrom -r bios_dump.bin -V -p internal
flashrom v0.9.7-r1711 on Darwin 14.4.0 (x86_64)
flashrom is free software, get the source code at http://www.flashrom.org

flashrom was built with libpci 3.1.7, LLVM Clang 6.0 (clang-600.0.56), little endian
Command line (5 args): ./flashrom -r bios_dump.bin -V -p internal
(...)
Found chipset "Intel HM77" with PCI ID 8086:1e57.
This chipset is marked as untested. If you are using an up-to-date version
of flashrom *and* were (not) able to successfully update your firmware with it,
then please email a report to flashrom@flashrom.org including a verbose (-V) log.
Thank you!
```

```
SPI Read Configuration: prefetching disabled, caching enabled, OK.
The following protocols are supported: FWH, SPI.
(..)
Probing for Micron/Numonyx/ST N25Q064..3E, 8192 kB: probe_spi_rdid_generic: id1 0x20, id2 0xba17
Found Micron/Numonyx/ST flash chip "N25Q064..3E" (8192 kB, SPI) at physical address 0xff800000.
Chip status register is 0x00.
Chip status register: Status Register Write Disable (SRWD, SRP, ...) is not set
Chip status register: Block Protect 3 (BP3) is not set
Chip status register: Top/Bottom (TB) is top
Chip status register: Block Protect 2 (BP2) is not set
Chip status register: Block Protect 1 (BP1) is not set
Chip status register: Block Protect 0 (BP0) is not set
Chip status register: Write Enable Latch (WEL) is not set
Chip status register: Write In Progress (WIP/BUSY) is not set
(...)
```

```
Found Micron/Numonyx/ST flash chip "N25Q064..3E" (8192 kB, SPI).
This chip may contain one-time programmable memory. flashrom cannot read
and may never be able to write it, hence it may not be able to completely
clone the contents of this chip (see man page for details).
Reading flash... done.
Restoring MMIO space at 0x10ae098a0
Restoring PCI config space for 00:1f:0 reg 0xdc

sh-3.2# ls -la bios_dump.bin
-rw-r--r--  1 root  staff  8388608 Jul  8 01:23 bios_dump.bin
```

Software

- AppleHWAccess.kext.
- readphysmem utility.
- Can read bios without external kext.
- Default on Mavericks and Yosemite.
- Not anymore on El Capitan.

Software

- Good enough to play around.
- Mostly useless to chase (U)EFI rootkits.
- Unless it is made by HackingTeam.
  - Their version makes no attempt to hide itself from software dumps.

What is in the flash?

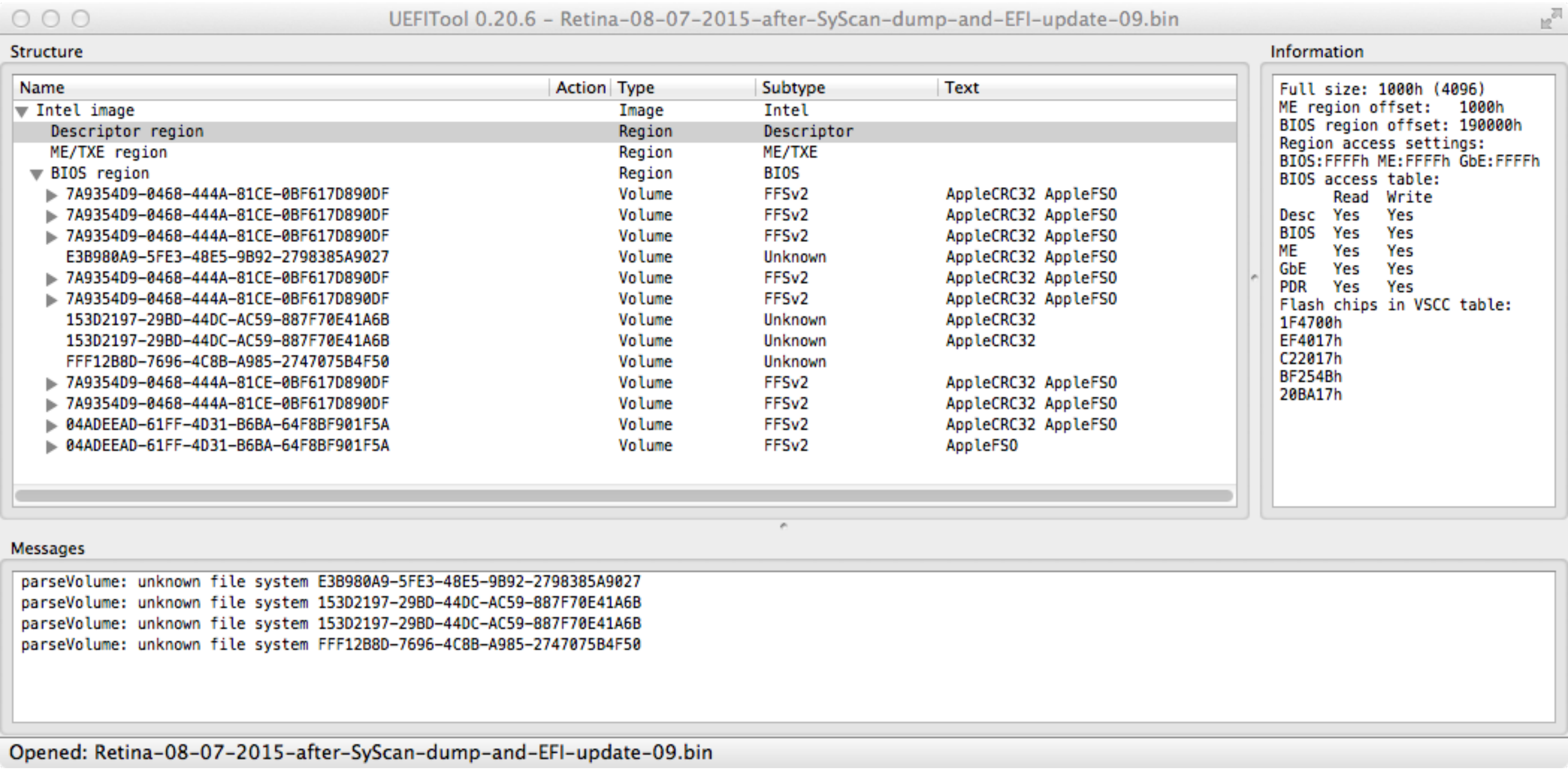
What’s in the flash

```
+-----------------------------------------+
|               Descriptor                |
|                 Region                  |
+-----------------------------[0x1000]----+
|                                         |
|                Intel                    |
|               Management                |
|                 Engine                  |  <- a big black box
|                  (ME)                   |           :-)
|                 Region                  |
|                                         |
+----------------------------[0x19000]----+
|                                         |
|                                         |
|                                         |
|                                         |
|                                         |
|                                         |
|                  BIOS                   |
|                 Region                  |  <- our playground!
|                                         |
|                                         |
|                                         |
|                                         |
|                                         |
|                                         |
+-----------------------------------------+
```

What's in the flash

```
+--------------------------------------+
|              Descriptor              |
|                Region                |
+--------------------------------------+
|                 GbE                  |
|                Region                |
+--------------------------------------+
|                  ME                  |
|                Region                |
+--------------------------------------+
|            Platform Data             |
|                Region                |
+--------------------------------------+
|                                      |
|                                      |
|                                      |
|                                      |
|                                      |
|                                      |
|                 BIOS                 |
|                Region                |
|                                      |
|                                      |
|                                      |
|                                      |
|                                      |
|                                      |
+--------------------------------------+
```

What's in the flash



UEFITool 0.20.6 – Retina-08-07-2015-after-SyScan-dump-and-EFI-update-09.bin

Structure

Name	Action	Type	Subtype	Text
▼ Intel image		Image	Intel	
Descriptor region		Region	Descriptor	
ME/TXE region		Region	ME/TXE	
▼ BIOS region		Region	BIOS	
▶ 7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2	AppleCRC32 AppleFSO
▶ 7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2	AppleCRC32 AppleFSO
▶ 7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2	AppleCRC32 AppleFSO
E3B980A9-5FE3-48E5-9B92-2798385A9027		Volume	Unknown	AppleCRC32 AppleFSO
▶ 7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2	AppleCRC32 AppleFSO
▶ 7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2	AppleCRC32 AppleFSO
153D2197-29BD-44DC-AC59-887F70E41A6B		Volume	Unknown	AppleCRC32
153D2197-29BD-44DC-AC59-887F70E41A6B		Volume	Unknown	AppleCRC32
FFF12B8D-7696-4C8B-A985-2747075B4F50		Volume	Unknown	
▶ 7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2	AppleCRC32 AppleFSO
▶ 7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2	AppleCRC32 AppleFSO
▶ 04ADEEAD-61FF-4D31-B6BA-64F8BF901F5A		Volume	FFSv2	AppleCRC32 AppleFSO
▶ 04ADEEAD-61FF-4D31-B6BA-64F8BF901F5A		Volume	FFSv2	AppleFSO

Information

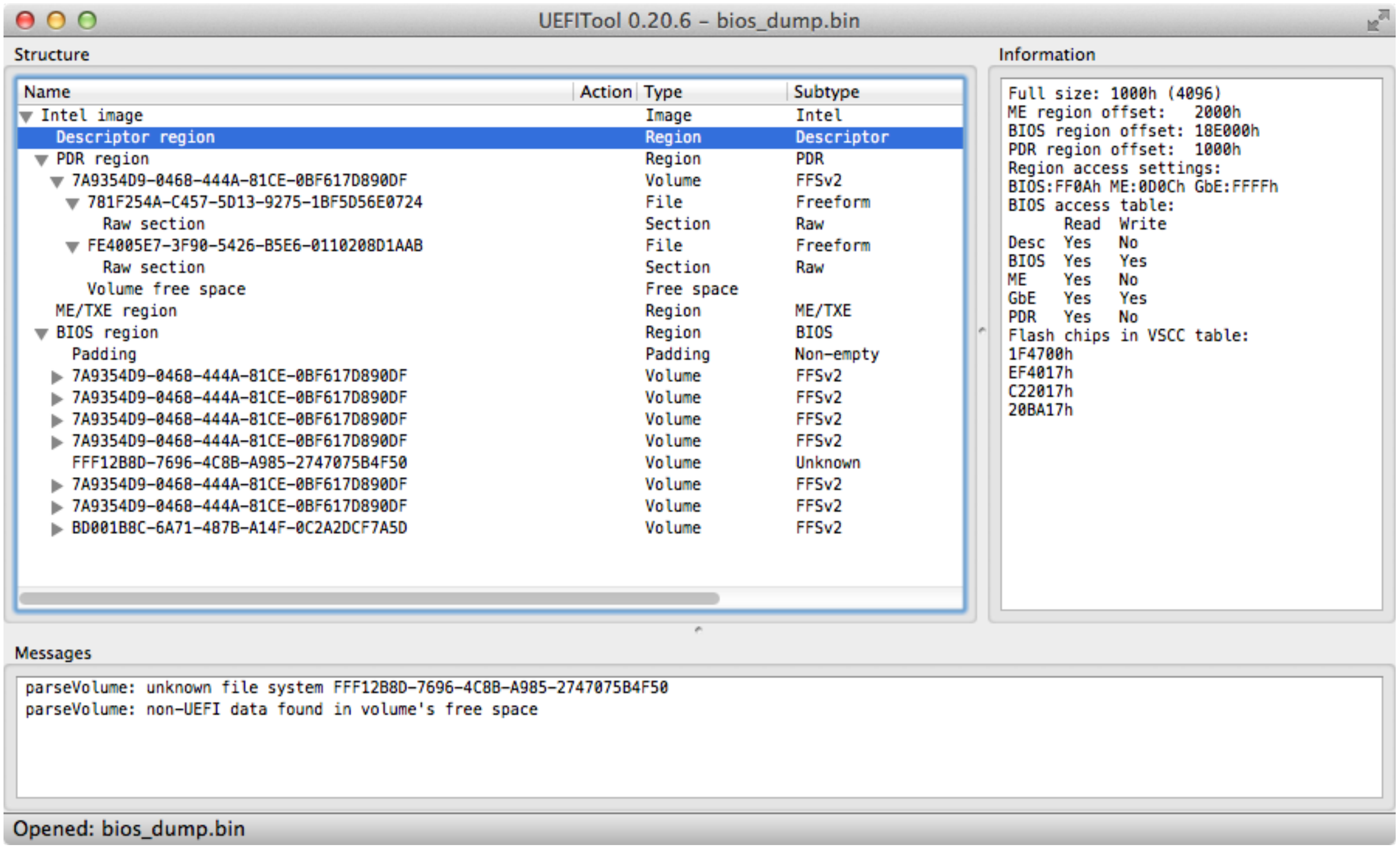
```
Full size: 1000h (4096)
ME region offset:   1000h
BIOS region offset: 190000h
Region access settings:
BIOS:FFFFh ME:FFFFh GbE:FFFFh
BIOS access table:
      Read  Write
Desc  Yes   Yes
BIOS  Yes   Yes
ME    Yes   Yes
GbE   Yes   Yes
PDR   Yes   Yes
Flash chips in VSCC table:
1F4700h
EF4017h
C22017h
BF254Bh
20BA17h
```

Messages

```
parseVolume: unknown file system E3B980A9-5FE3-48E5-9B92-2798385A9027
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system FFF12B8D-7696-4C8B-A985-2747075B4F50
```

Opened: Retina-08-07-2015-after-SyScan-dump-and-EFI-update-09.bin

What's in the flash



UEFITool 0.20.6 – bios_dump.bin

Structure

Name	Action	Type	Subtype
Intel image		Image	Intel
Descriptor region		Region	Descriptor
PDR region		Region	PDR
7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2
781F254A-C457-5D13-9275-1BF5D56E0724		File	Freeform
Raw section		Section	Raw
FE4005E7-3F90-5426-B5E6-0110208D1AAB		File	Freeform
Raw section		Section	Raw
Volume free space		Free space	
ME/TXE region		Region	ME/TXE
BIOS region		Region	BIOS
Padding		Padding	Non-empty
7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2
7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2
7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2
7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2
FFF12B8D-7696-4C8B-A985-2747075B4F50		Volume	Unknown
7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2
7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2
BD001B8C-6A71-487B-A14F-0C2A2DCF7A5D		Volume	FFSv2

Information

```
Full size: 1000h (4096)
ME region offset:   2000h
BIOS region offset: 18E000h
PDR region offset:  1000h
Region access settings:
BIOS:FF0Ah ME:0D0Ch GbE:FFFFh
BIOS access table:
      Read  Write
Desc  Yes   No
BIOS  Yes   Yes
ME    Yes   No
GbE   Yes   Yes
PDR   Yes   No
Flash chips in VSCC table:
1F4700h
EF4017h
C22017h
20BA17h
```

Messages

```
parseVolume: unknown file system FFF12B8D-7696-4C8B-A985-2747075B4F50
parseVolume: non-UEFI data found in volume's free space
```

Opened: bios_dump.bin

Descriptor region

- Location of other regions.
- Access permissions.
  - OS/BIOS shouldn't access ME region.
- VSCC configures ME flash access.

Intel ME region

- A CPU inside your CPU ☺.
- Runs Java.
- Can be active with system powered off.
- Out of band network access!
- No access from BIOS and OS.

Intel ME region

- Mostly a blackbox.
- Three presentations by Igor Skochinsky.
- Definitely requires more research!
- Unpacker
  - http://io.smashthestack.org/me/

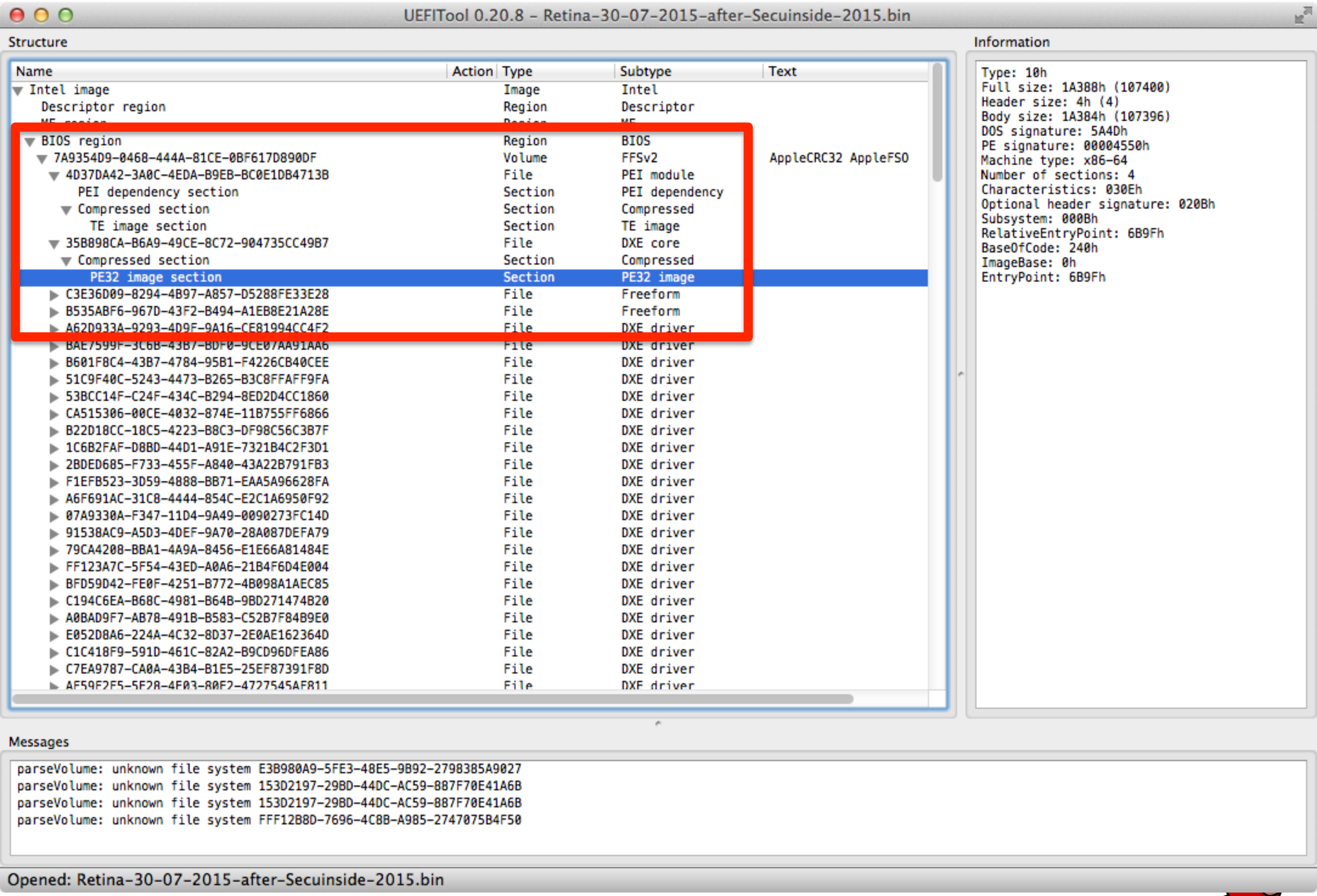
Intel ME region

- Rootkit in your laptop: Hidden code in your chipset and how to discover what exactly it does
- Intel ME Secrets
- Intel ME: Two years later
- https://github.com/skochinsky/papers

BIOS region

- Contains
  - EFI binaries for different phases.
  - NVRAM.
  - Microcode (not for some models).
- Each on its own firmware volume (FVH).

```
+----------------------------[0x19000]---+
|  7A9354D9-0468-444A-81CE-0BF617D890DF  |
+----------------------------------------+
|  7A9354D9-0468-444A-81CE-0BF617D890DF  |
+----------------------------------------+
|  7A9354D9-0468-444A-81CE-0BF617D890DF  |
+----------------------------------------+
|  E3B980A9-5FE3-48E5-9B92-2798385A9027  |
+----------------------------------------+
|  7A9354D9-0468-444A-81CE-0BF617D890DF  |
+----------------------------------------+
|  7A9354D9-0468-444A-81CE-0BF617D890DF  |
+----------------------------------------+
|  153D2197-29BD-44DC-AC59-887F70E41A6B  | <- Microcode
+----------------------------------------+
|  153D2197-29BD-44DC-AC59-887F70E41A6B  | <- Microcode
+----------------------------------------+
|  FFF12B8D-7696-4C8B-A985-2747075B4F50  | <- NVRAM
+----------------------------------------+
|  7A9354D9-0468-444A-81CE-0BF617D890DF  |
+----------------------------------------+
|  7A9354D9-0468-444A-81CE-0BF617D890DF  |
+----------------------------------------+
|  04ADEEAD-61FF-4D31-B6BA-64F8BF901F5A  | <- Boot Volume
+----------------------------------------+
|  04ADEEAD-61FF-4D31-B6BA-64F8BF901F5A  | <- Boot Volume
+----------------------------------------+
```



UEFITool 0.20.8 – Retina-30-07-2015-after-Secuinside-2015.bin

Structure

```
Name                                          Action  Type      Subtype          Text
▼ Intel image                                         Image     Intel
    Descriptor region                                 Region    Descriptor
  ▼ BIOS region                                       Region    BIOS
    ▼ 7A9354D9-0468-444A-81CE-0BF617D890DF            Volume    FFSv2            AppleCRC32 AppleFSO
      ▼ 4D37DA42-3A0C-4EDA-B9EB-BC0E1DB4713B          File      PEI module
          PEI dependency section                      Section   PEI dependency
        ▼ Compressed section                          Section   Compressed
            TE image section                          Section   TE image
      ▼ 35B898CA-B6A9-49CE-8C72-904735CC49B7          File      DXE core
        ▼ Compressed section                          Section   Compressed
            PE32 image section                        Section   PE32 image
      ▶ C3E36D09-8294-4B97-A857-D5288FE33E28          File      Freeform
      ▶ B535ABF6-967D-43F2-B494-A1EB8E21A28E          File      Freeform
      ▶ A62D933A-9293-4D9F-9A16-CE81994CC4F2          File      DXE driver
      ▶ BAE7599F-3C6B-43B7-BDF0-9CE07AA91AA6          File      DXE driver
      ▶ B601F8C4-43B7-4784-95B1-F4226CB40CEE          File      DXE driver
      ▶ 51C9F40C-5243-4473-B265-B3C8FFAFF9FA          File      DXE driver
      ▶ 53BCC14F-C24F-434C-B294-8ED2D4CC1860          File      DXE driver
      ▶ CA515306-00CE-4032-874E-11B755FF6866          File      DXE driver
      ▶ B22D18CC-18C5-4223-B8C3-DF98C56C3B7F          File      DXE driver
      ▶ 1C6B2FAF-D8BD-44D1-A91E-7321B4C2F3D1          File      DXE driver
      ▶ 2BDED685-F733-455F-A840-43A22B791FB3          File      DXE driver
      ▶ F1EFB523-3D59-4888-BB71-EAA5A96628FA          File      DXE driver
      ▶ A6F691AC-31C8-4444-854C-E2C1A6950F92          File      DXE driver
      ▶ 07A9330A-F347-11D4-9A49-0090273FC14D          File      DXE driver
      ▶ 91538AC9-A5D3-4DEF-9A70-28A087DEFA79          File      DXE driver
      ▶ 79CA4208-BBA1-4A9A-8456-E1E66A81484E          File      DXE driver
      ▶ FF123A7C-5F54-43ED-A0A6-21B4F6D4E004          File      DXE driver
      ▶ BFD59D42-FE0F-4251-B772-4B098A1AEC85          File      DXE driver
      ▶ C194C6EA-B68C-4981-B64B-9BD271474B20          File      DXE driver
      ▶ A0BAD9F7-AB78-491B-B583-C52B7F84B9E0          File      DXE driver
      ▶ E052D8A6-224A-4C32-8D37-2E0AE162364D          File      DXE driver
      ▶ C1C418F9-591D-461C-82A2-B9CD96DFEA86          File      DXE driver
      ▶ C7EA9787-CA0A-43B4-B1E5-25EF87391F8D          File      DXE driver
      ▶ AF59F2F5-5E28-4E03-80E2-4727545AF811          File      DXE driver
```

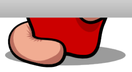
Information

```
Type: 10h
Full size: 1A388h (107400)
Header size: 4h (4)
Body size: 1A384h (107396)
DOS signature: 5A4Dh
PE signature: 00004550h
Machine type: x86-64
Number of sections: 4
Characteristics: 030Eh
Optional header signature: 020Bh
Subsystem: 000Bh
RelativeEntryPoint: 6B9Fh
BaseOfCode: 240h
ImageBase: 0h
EntryPoint: 6B9Fh
```

Messages

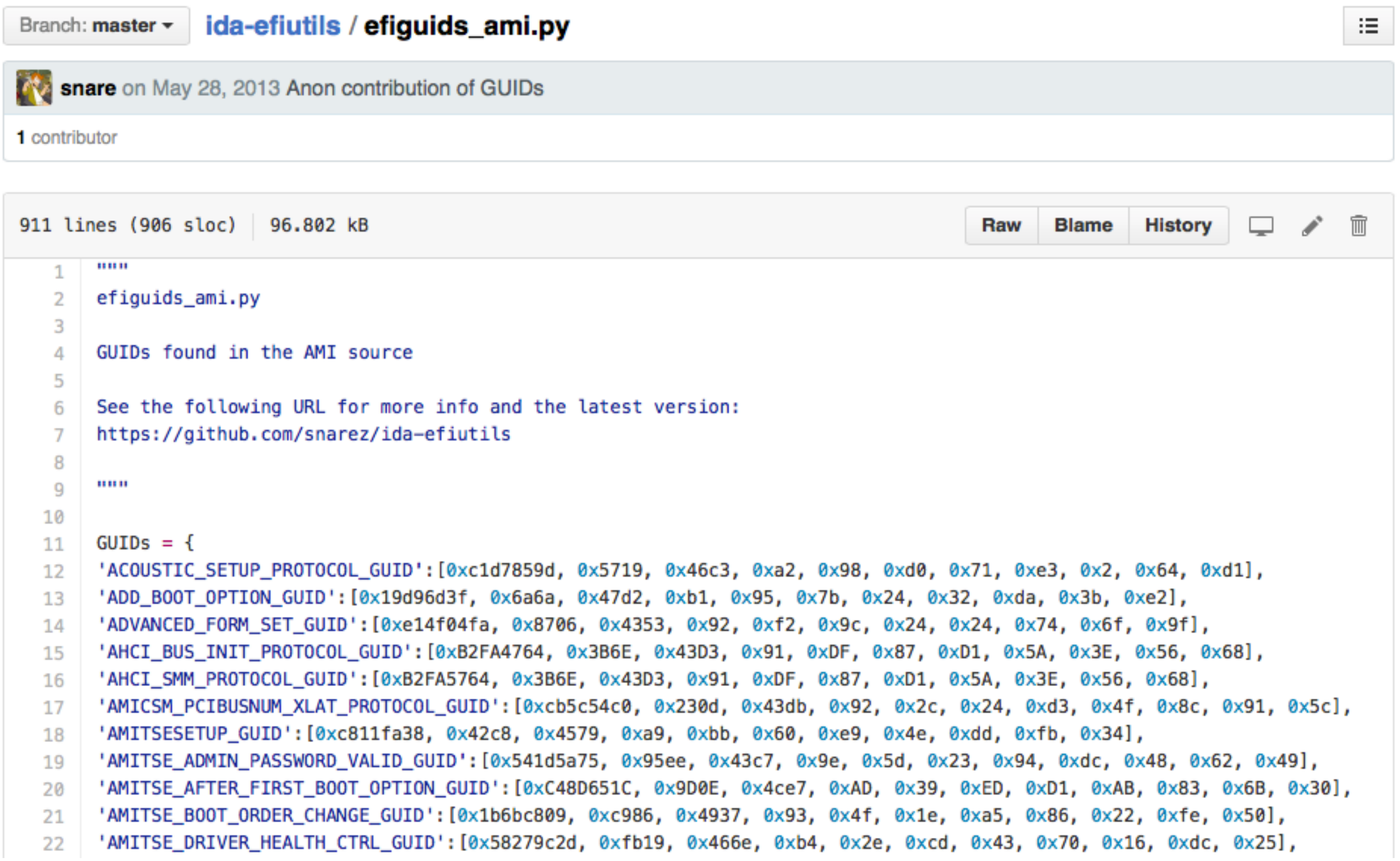
```
parseVolume: unknown file system E3B980A9-5FE3-48E5-9B92-2798385A9027
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system FFF12B8D-7696-4C8B-A985-2747075B4F50
```

Opened: Retina-30-07-2015-after-Secuinside-2015.bin



BIOS region

- Everything is labeled with a GUID.
- No filenames.
- Many GUID can be found in EFI specs.
- Others are vendor specific/private.
- Google and luck are your friends!



ida-efiutils / efiguids_ami.py

snare on May 28, 2013 Anon contribution of GUIDs

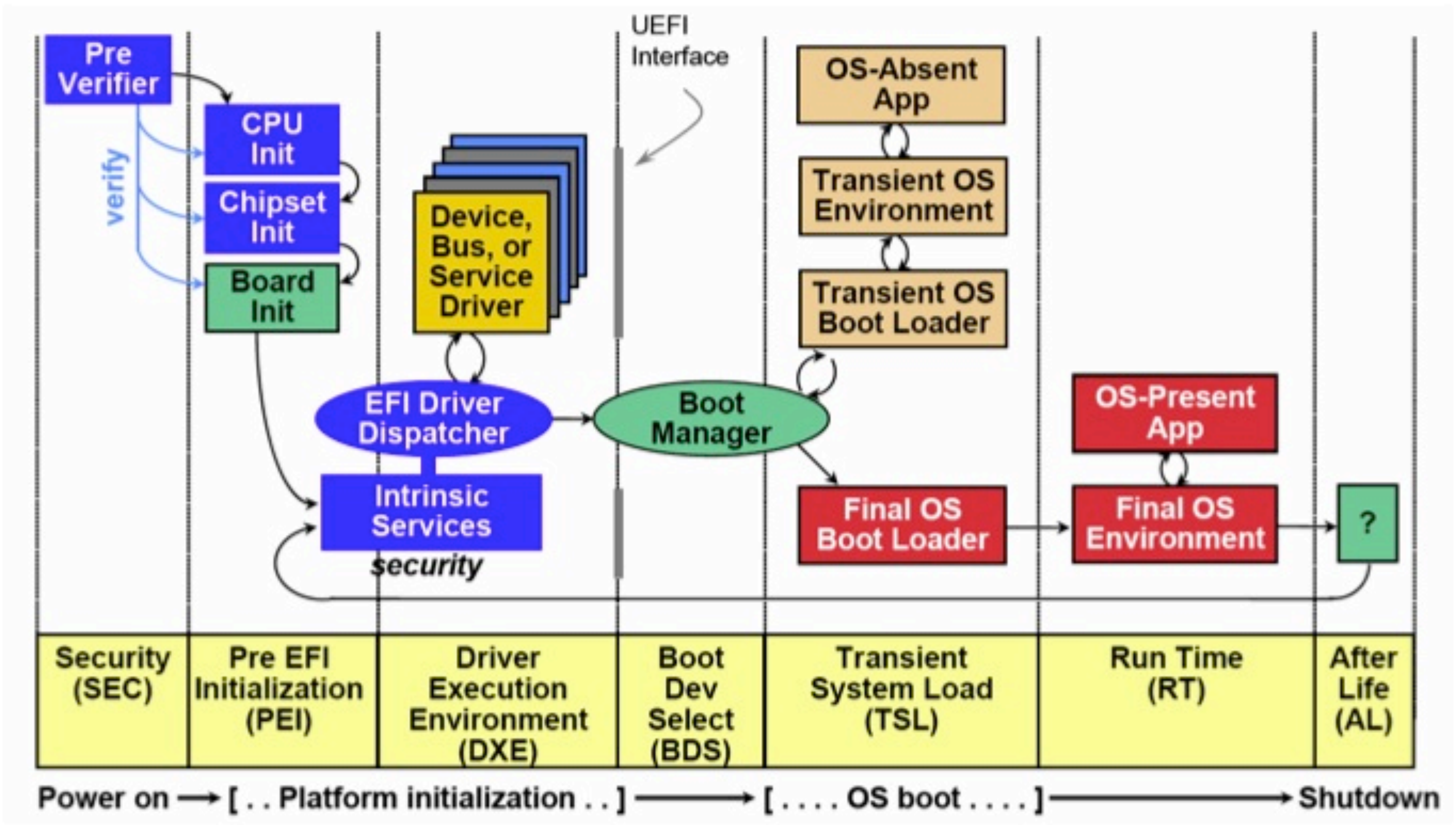
1 contributor

911 lines (906 sloc) | 96.802 kB

```
"""
efiguids_ami.py

GUIDs found in the AMI source

See the following URL for more info and the latest version:
https://github.com/snarez/ida-efiutils

"""

GUIDs = {
'ACOUSTIC_SETUP_PROTOCOL_GUID':[0xc1d7859d, 0x5719, 0x46c3, 0xa2, 0x98, 0xd0, 0x71, 0xe3, 0x2, 0x64, 0xd1],
'ADD_BOOT_OPTION_GUID':[0x19d96d3f, 0x6a6a, 0x47d2, 0xb1, 0x95, 0x7b, 0x24, 0x32, 0xda, 0x3b, 0xe2],
'ADVANCED_FORM_SET_GUID':[0xe14f04fa, 0x8706, 0x4353, 0x92, 0xf2, 0x9c, 0x24, 0x24, 0x74, 0x6f, 0x9f],
'AHCI_BUS_INIT_PROTOCOL_GUID':[0xB2FA4764, 0x3B6E, 0x43D3, 0x91, 0xDF, 0x87, 0xD1, 0x5A, 0x3E, 0x56, 0x68],
'AHCI_SMM_PROTOCOL_GUID':[0xB2FA5764, 0x3B6E, 0x43D3, 0x91, 0xDF, 0x87, 0xD1, 0x5A, 0x3E, 0x56, 0x68],
'AMICSM_PCIBUSNUM_XLAT_PROTOCOL_GUID':[0xcb5c54c0, 0x230d, 0x43db, 0x92, 0x2c, 0x24, 0xd3, 0x4f, 0x8c, 0x91, 0x5c],
'AMITSESETUP_GUID':[0xc811fa38, 0x42c8, 0x4579, 0xa9, 0xbb, 0x60, 0xe9, 0x4e, 0xdd, 0xfb, 0x34],
'AMITSE_ADMIN_PASSWORD_VALID_GUID':[0x541d5a75, 0x95ee, 0x43c7, 0x9e, 0x5d, 0x23, 0x94, 0xdc, 0x48, 0x62, 0x49],
'AMITSE_AFTER_FIRST_BOOT_OPTION_GUID':[0xC48D651C, 0x9D0E, 0x4ce7, 0xAD, 0x39, 0xED, 0xD1, 0xAB, 0x83, 0x6B, 0x30],
'AMITSE_BOOT_ORDER_CHANGE_GUID':[0x1b6bc809, 0xc986, 0x4937, 0x93, 0x4f, 0x1e, 0xa5, 0x86, 0x22, 0xfe, 0x50],
'AMITSE_DRIVER_HEALTH_CTRL_GUID':[0x58279c2d, 0xfb19, 0x466e, 0xb4, 0x2e, 0xcd, 0x43, 0x70, 0x16, 0xdc, 0x25],
```

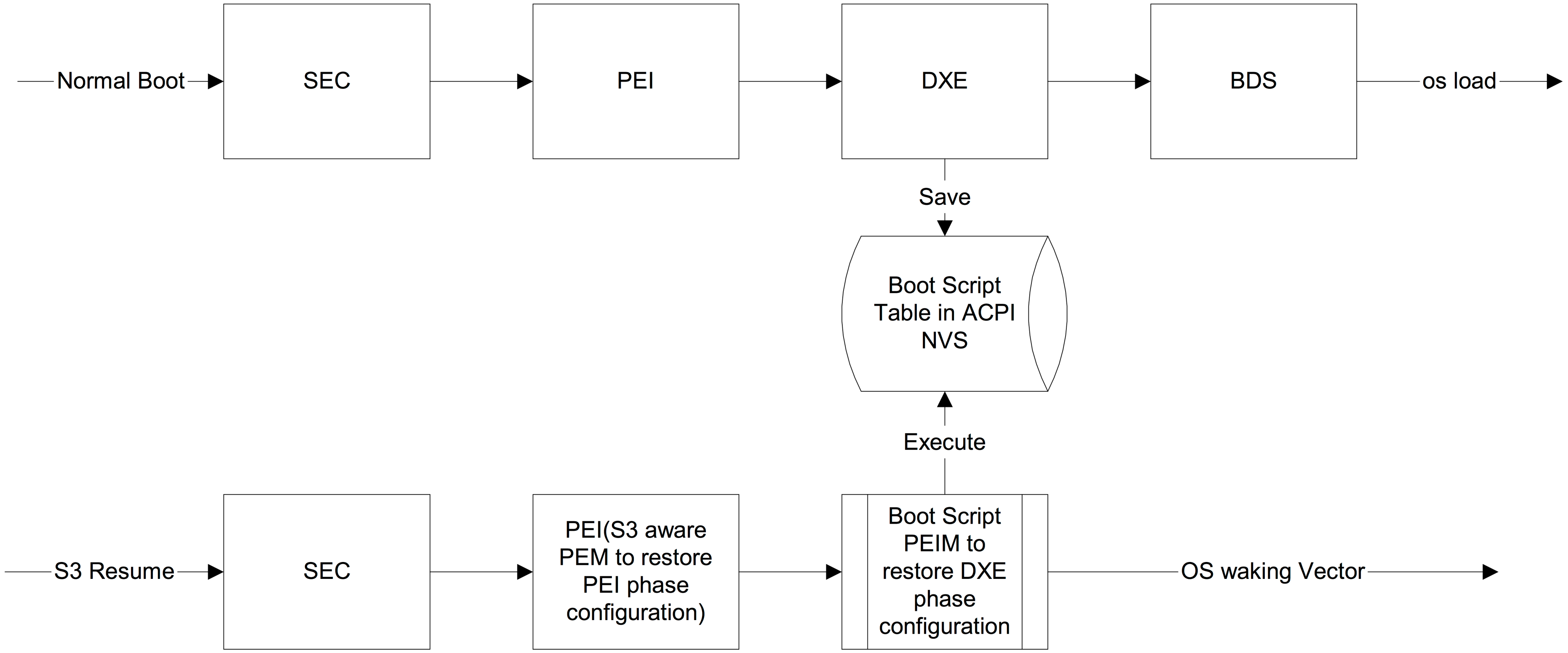
EFI Boot Flow

PI Boot Phases



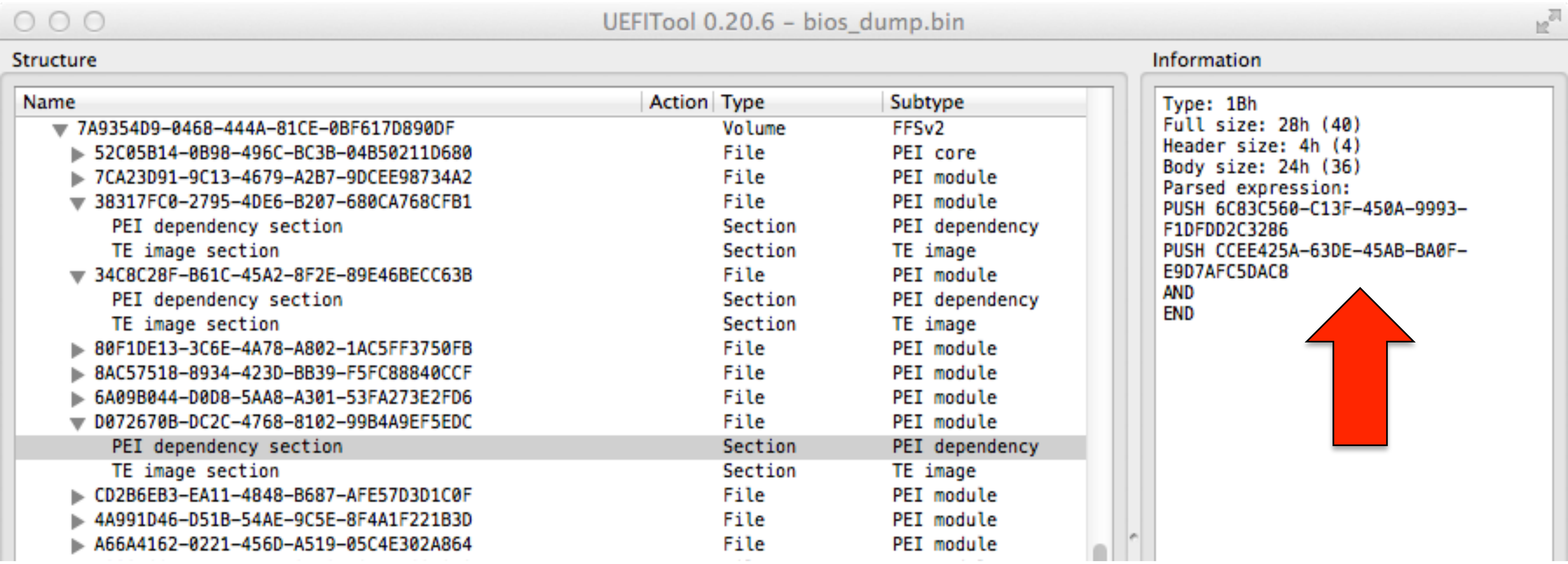
EFI Boot Phases

- Different initialization phases.
- Make resources available to next phase.
- Memory for example.


Normal Boot
SEC
PEI
DXE
BDS
os load
Save
Boot Script Table in ACPI NVS
Execute
S3 Resume
SEC
PEI(S3 aware PEM to restore PEI phase configuration)
Boot Script PEIM to restore DXE phase configuration
OS waking Vector

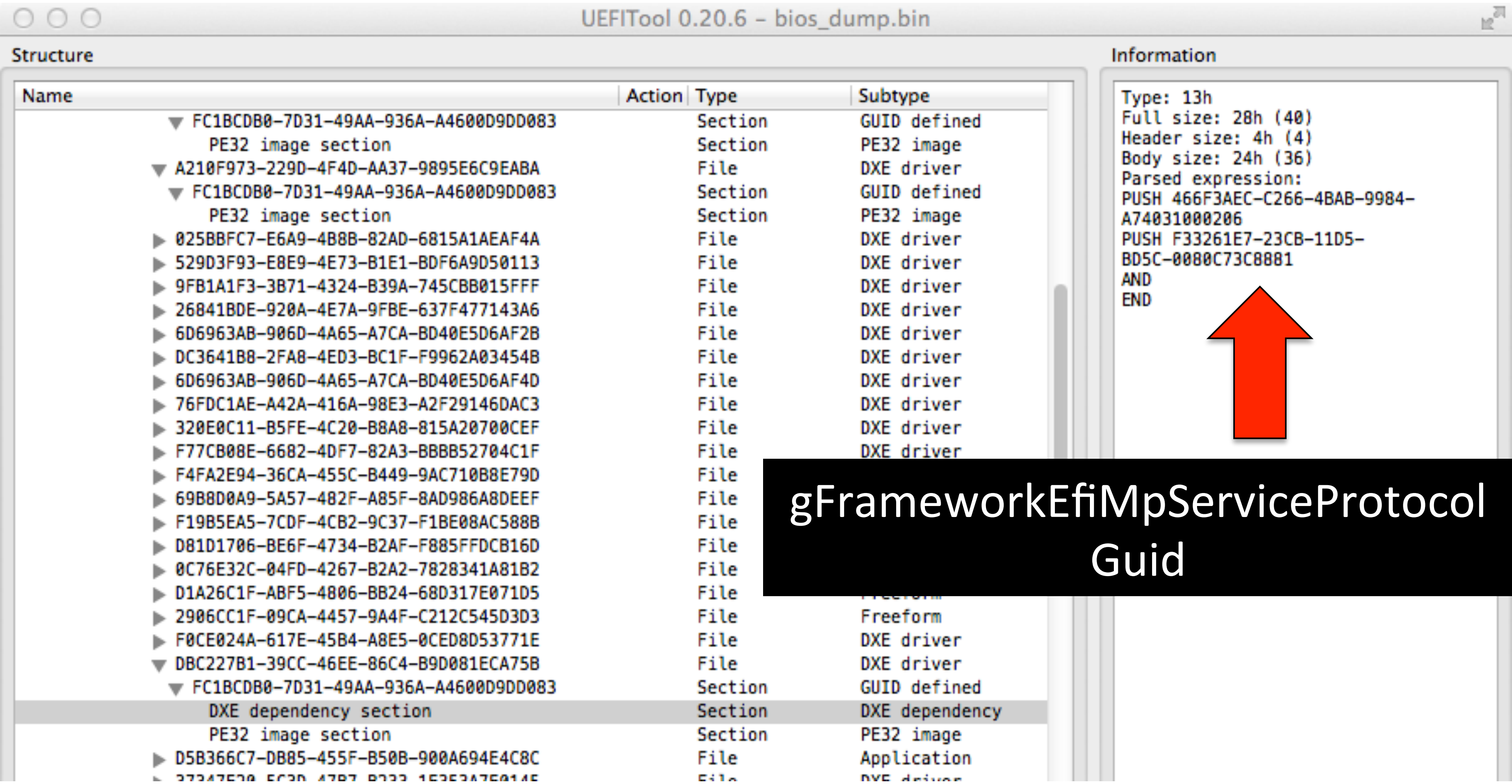
The PEI/DXE Dispatchers

- PEI and DXE phases have a dispatcher.
- Guarantees dependencies and load order.
- Dependency expressions.
- Available as a section.



```
Name                                          Action  Type      Subtype
▼ 7A9354D9-0468-444A-81CE-0BF617D890DF                Volume    FFSv2
  ▶ 52C05B14-0B98-496C-BC3B-04B50211D680              File      PEI core
  ▶ 7CA23D91-9C13-4679-A2B7-9DCEE98734A2              File      PEI module
  ▼ 38317FC0-2795-4DE6-B207-680CA768CFB1              File      PEI module
      PEI dependency section                          Section   PEI dependency
      TE image section                                Section   TE image
  ▼ 34C8C28F-B61C-45A2-8F2E-89E46BECC63B              File      PEI module
      PEI dependency section                          Section   PEI dependency
      TE image section                                Section   TE image
  ▶ 80F1DE13-3C6E-4A78-A802-1AC5FF3750FB              File      PEI module
  ▶ 8AC57518-8934-423D-BB39-F5FC88840CCF              File      PEI module
  ▶ 6A09B044-D0D8-5AA8-A301-53FA273E2FD6              File      PEI module
  ▼ D072670B-DC2C-4768-8102-99B4A9EF5EDC              File      PEI module
      PEI dependency section                          Section   PEI dependency
      TE image section                                Section   TE image
  ▶ CD2B6EB3-EA11-4848-B687-AFE57D3D1C0F              File      PEI module
  ▶ 4A991D46-D51B-54AE-9C5E-8F4A1F221B3D              File      PEI module
  ▶ A66A4162-0221-456D-A519-05C4E302A864              File      PEI module
```

```
Type: 1Bh
Full size: 28h (40)
Header size: 4h (4)
Body size: 24h (36)
Parsed expression:
PUSH 6C83C560-C13F-450A-9993-
F1DFDD2C3286
PUSH CCEE425A-63DE-45AB-BA0F-
E9D7AFC5DAC8
AND
END
```


UEFITool 0.20.6 – bios_dump.bin

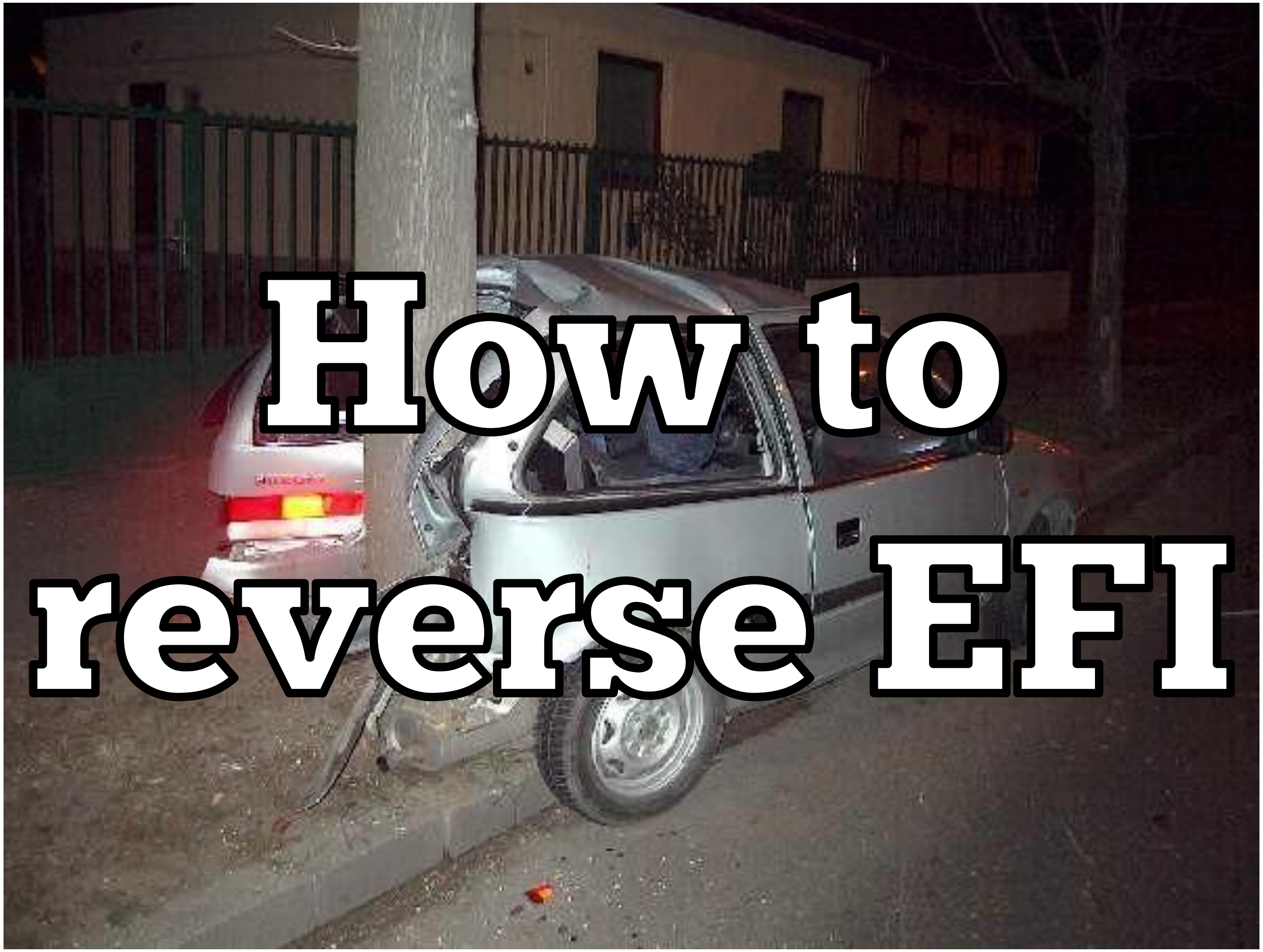
Structure

```
Name                                            Action  Type      Subtype
    ▼ FC1BCDB0-7D31-49AA-936A-A4600D9DD083              Section   GUID defined
        PE32 image section                              Section   PE32 image
  ▼ A210F973-229D-4F4D-AA37-9895E6C9EABA                File      DXE driver
    ▼ FC1BCDB0-7D31-49AA-936A-A4600D9DD083              Section   GUID defined
        PE32 image section                              Section   PE32 image
  ▶ 025BBFC7-E6A9-4B8B-82AD-6815A1AEAF4A                File      DXE driver
  ▶ 529D3F93-E8E9-4E73-B1E1-BDF6A9D50113                File      DXE driver
  ▶ 9FB1A1F3-3B71-4324-B39A-745CBB015FFF                File      DXE driver
  ▶ 26841BDE-920A-4E7A-9FBE-637F477143A6                File      DXE driver
  ▶ 6D6963AB-906D-4A65-A7CA-BD40E5D6AF2B                File      DXE driver
  ▶ DC3641B8-2FA8-4ED3-BC1F-F9962A03454B                File      DXE driver
  ▶ 6D6963AB-906D-4A65-A7CA-BD40E5D6AF4D                File      DXE driver
  ▶ 76FDC1AE-A42A-416A-98E3-A2F29146DAC3                File      DXE driver
  ▶ 320E0C11-B5FE-4C20-B8A8-815A20700CEF                File      DXE driver
  ▶ F77CB08E-6682-4DF7-82A3-BBBB52704C1F                File      DXE driver
  ▶ F4FA2E94-36CA-455C-B449-9AC710B8E79D                File
  ▶ 69B8D0A9-5A57-482F-A85F-8AD986A8DEEF                File
  ▶ F19B5EA5-7CDF-4CB2-9C37-F1BE08AC588B                File
  ▶ D81D1706-BE6F-4734-B2AF-F885FFDCB16D                File
  ▶ 0C76E32C-04FD-4267-B2A2-7828341A81B2                File
  ▶ D1A26C1F-ABF5-4806-BB24-68D317E071D5                File
  ▶ 2906CC1F-09CA-4457-9A4F-C212C545D3D3                File      Freeform
  ▶ F0CE024A-617E-45B4-A8E5-0CED8D53771E                File      DXE driver
  ▼ DBC227B1-39CC-46EE-86C4-B9D081ECA75B                File      DXE driver
    ▼ FC1BCDB0-7D31-49AA-936A-A4600D9DD083              Section   GUID defined
        DXE dependency section                          Section   DXE dependency
        PE32 image section                              Section   PE32 image
  ▶ D5B366C7-DB85-455F-B50B-900A694E4C8C                File      Application
```

Information

```
Type: 13h
Full size: 28h (40)
Header size: 4h (4)
Body size: 24h (36)
Parsed expression:
PUSH 466F3AEC-C266-4BAB-9984-
A74031000206
PUSH F33261E7-23CB-11D5-
BD5C-0080C73C8881
AND
END
```

gFrameworkEfiMpServiceProtocol Guid



How to reverse EFI

Tools

- UEFITool and UEFIExtract
  - https://github.com/LongSoft/UEFITool
- Snare's IDA EFI Utils
  - https://github.com/snare/ida-efiutils/
- UEFI Firmware parser
  - https://github.com/snare/ida-efiutils/
- CHIPSEC
  - https://github.com/chipsec/chipsec

EFI file types

- Two executable file types.
- PE32/PE32+ (as in Windows).
- TE – Terse Executable.
- 16/32/64 bit code, depending on phase.

TE file format

- TE is just a stripped version of PE.
- Unnecessary PE headers are removed.
- To save space.
- Used by SEC and PEI phase binaries.

TE file format

- IDA unable to correctly disassemble.
- Fails to parse the TE headers.
- Afaik, still not fixed in 6.8.
- Solution is to build your own TE loader.
- https://github.com/gdbinit/TELoader

Where is libc?

EFI Services

- No standard libraries to link against.
- Instead there are services.
- Basic functions made available on each phase.
- Access via function pointers.

EFI Services

```
typedef struct _EFI_PEI_SERVICES {
  EFI_TABLE_HEADER              Hdr;
  EFI_PEI_INSTALL_PPI           InstallPpi;         <----.
  EFI_PEI_REINSTALL_PPI         ReInstallPpi;            |
  EFI_PEI_LOCATE_PPI            LocatePpi;               |
  EFI_PEI_NOTIFY_PPI            NotifyPpi;               |
  EFI_PEI_GET_BOOT_MODE         GetBootMode;             |
  EFI_PEI_SET_BOOT_MODE         SetBootMode;             |
  EFI_PEI_GET_HOB_LIST          GetHobList;              |
  EFI_PEI_CREATE_HOB            CreateHob;               |
  EFI_PEI_FFS_FIND_NEXT_VOLUME  FfsFindNextVolume;       |
  EFI_PEI_FFS_FIND_NEXT_FILE    FfsFindNextFile;         |
  EFI_PEI_FFS_FIND_SECTION_DATA FfsFindSectionData;      |
  EFI_PEI_INSTALL_PEI_MEMORY    InstallPeiMemory;        |
  EFI_PEI_ALLOCATE_PAGES        AllocatePages;           |
  EFI_PEI_ALLOCATE_POOL         AllocatePool;            |
  EFI_PEI_COPY_MEM              CopyMem;                 |
  EFI_PEI_SET_MEM               CopyMem;                 |
  EFI_PEI_REPORT_STATUS_CODE    CopyMem;                 |
  EFI_PEI_RESET_SYSTEM          ResetSystem;             |
  EFI_PEI_CPU_IO_PPI            CpuIo;                   |
  EFI_PEI_PCI_CFG_PPI           PciCfg;             <----'
} EFI_PEI_SERVICES;
```

EFI Services

```
typedef struct {
  EFI_TABLE_HEADER                Hdr;
  EFI_GET_TIME                    GetTime;                      <----.
  EFI_SET_TIME                    SetTime;                           |
  EFI_GET_WAKEUP_TIME             GetWakeupTime;                     |
  EFI_SET_WAKEUP_TIME             SetWakeupTime;                     |
  EFI_SET_VIRTUAL_ADDRESS_MAP     SetVirtualAddressMap;              |
  EFI_CONVERT_POINTER             ConvertPointer;                    |
  EFI_GET_VARIABLE                GetVariable;                       |
  EFI_GET_NEXT_VARIABLE_NAME      GetNextVariableName;               |
  EFI_SET_VARIABLE                SetVariable;                       |
  EFI_GET_NEXT_HIGH_MONO_COUNT    GetNextHighMonotonicCount;         |
  EFI_RESET_SYSTEM                ResetSystem;                       |
  EFI_UPDATE_CAPSULE              UpdateCapsule;                     |
  EFI_QUERY_CAPSULE_CAPABILITIES QueryCapsuleCapabilities;           |
  EFI_QUERY_VARIABLE_INFO         QueryVariableInfo;            <----'
} EFI_RUNTIME_SERVICES;
```

EFI Services

- Each phase has different services.
- Entrypoint function contains a pointer to the tables.

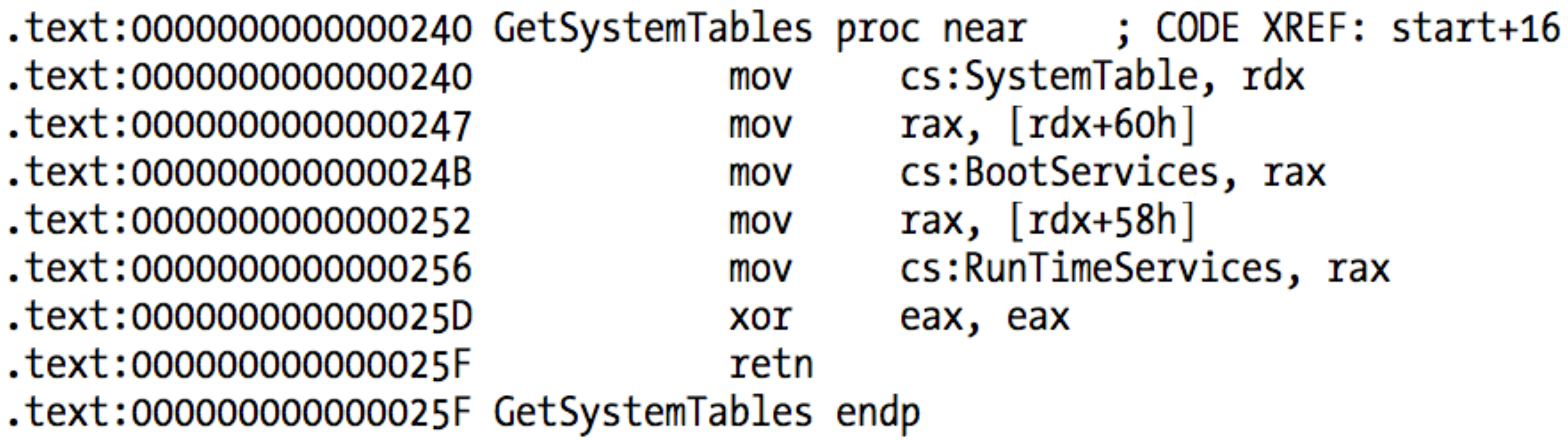
```
typedef
EFI_STATUS
 (*EFI_IMAGE_ENTRY_POINT)(
  IN EFI_HANDLE ImageHandle,
  IN EFI_SYSTEM_TABLE *SystemTable <------- this one
);
```

EFI Services

```
typedef struct {
  EFI_TABLE_HEADER Hdr;
  CHAR16 *FirmwareVendor;
  UINT32 FirmwareRevision;

  EFI_HANDLE ConsoleInHandle;
  EFI_SIMPLE_TEXT_INPUT_PROTOCOL *ConIn;
  EFI_HANDLE ConsoleOutHandle;
  EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL *ConOut;
  EFI_HANDLE StandardErrorHandle;
  EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL *StdErr;

  EFI_RUNTIME_SERVICES *RuntimeServices; <- EFI_RUNTIME_SERVICES
  EFI_BOOT_SERVICES *BootServices;       <- EFI_BOOT_SERVICES

  UINTN NumberOfTableEntries;
  EFI_CONFIGURATION_TABLE *ConfigurationTable;
} EFI_SYSTEM_TABLE;
```

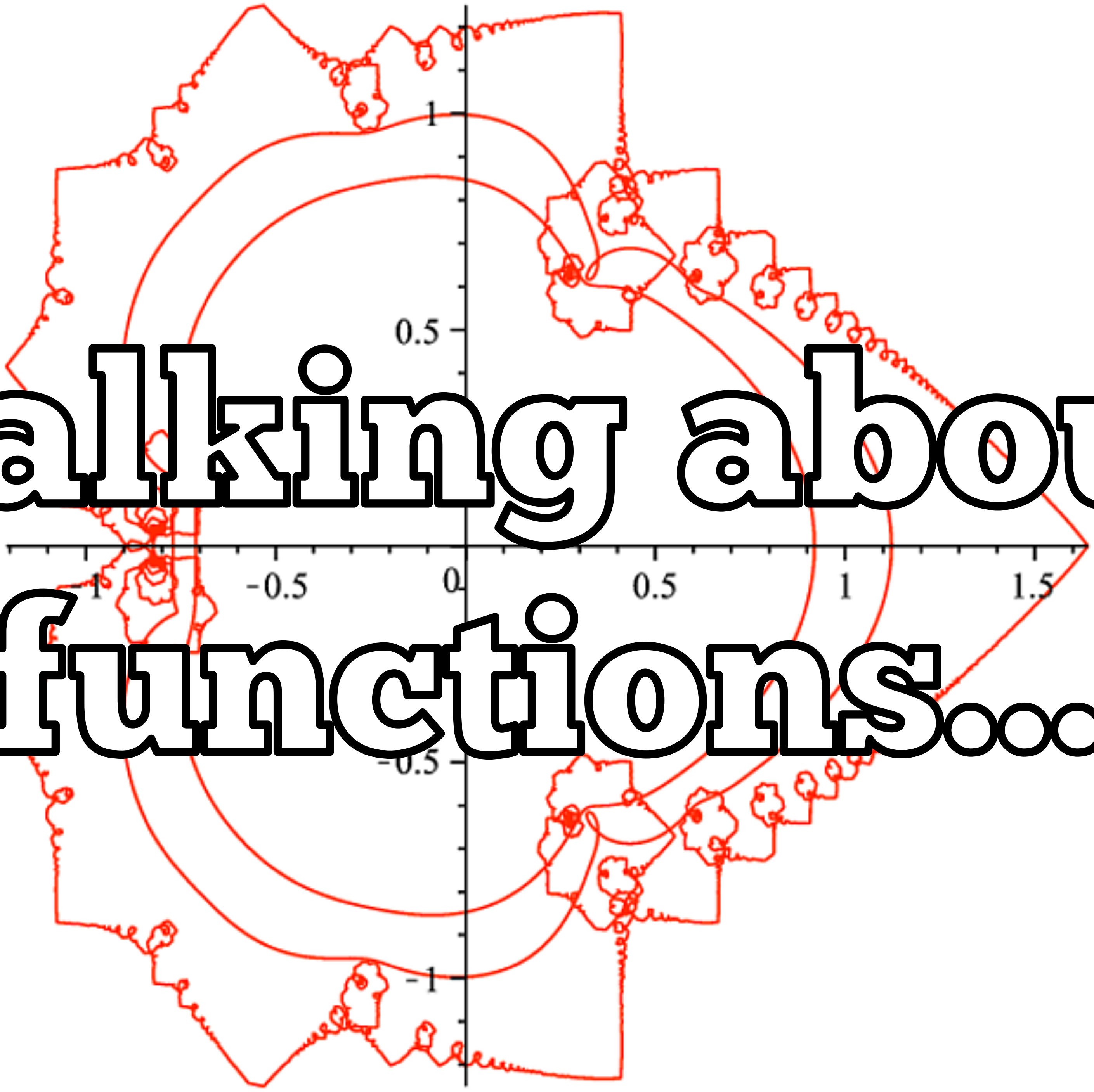
EFI Services

- Code that you often see in DXE drivers



```
.text:0000000000000240 GetSystemTables proc near     ; CODE XREF: start+16
.text:0000000000000240                 mov     cs:SystemTable, rdx
.text:0000000000000247                 mov     rax, [rdx+60h]
.text:000000000000024B                 mov     cs:BootServices, rax
.text:0000000000000252                 mov     rax, [rdx+58h]
.text:0000000000000256                 mov     cs:RunTimeServices, rax
.text:000000000000025D                 xor     eax, eax
.text:000000000000025F                 retn
.text:000000000000025F GetSystemTables endp
```

Talking about functions...



Calling conventions

- 32-bit binaries use standard C convention
  - Arguments passed on the stack.
  - SEC/PEI phase binaries.

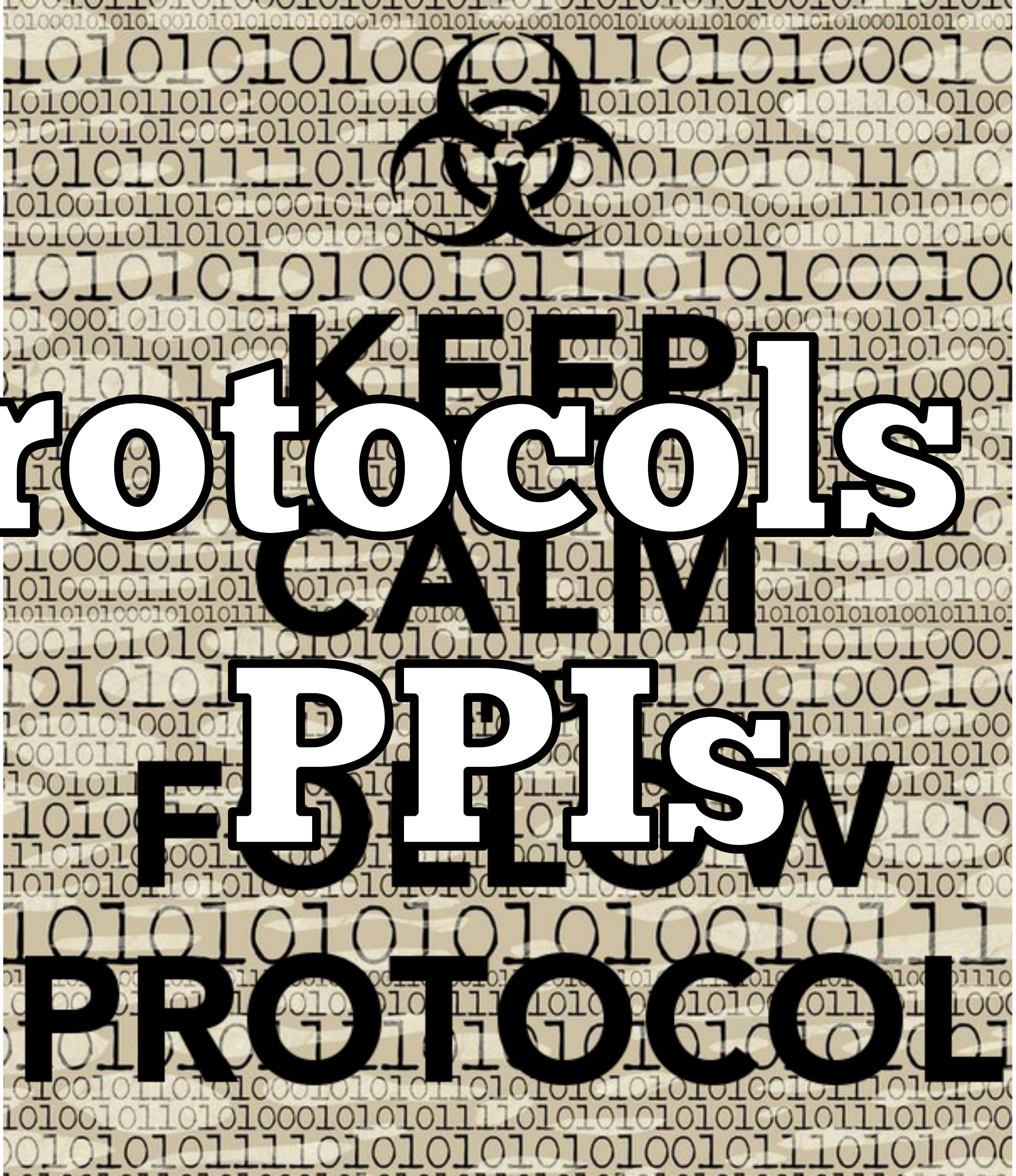
```
call    PeiPerfMeasure  ;     PEI_PERF_START (&PrivateData.PS,L"PreMem", NULL, mTick);
lea     eax, [ebp+var_C8]
mov     [esp+8], eax
lea     eax, [ebp-268h]
mov     [esp+4], eax
mov     [esp], edi
call    PeiDispatcher   ;   PeiDispatcher (PeiStartupDescriptor, &PrivateData, DispatchData);
cmp     [ebp+var_9B], 1
jz      short loc_FFEA736E
mov     [esp], esi
mov     dword ptr [esp+0Ch], offset aPrivatedata_pe ; "PrivateData.PeiMemoryInstalled == ((BOO"...
mov     dword ptr [esp+8], 16Ch
mov     dword ptr [esp+4], offset a_EdkFoundati_4 ; "./Edk/Foundation/Core/Pei/PeiMain/PeiMa"...
call    PeiDebugAssert  ;   PEI_ASSERT(&PrivateData.PS, PrivateData.PeiMemoryInstalled == TRUE);
```

Calling conventions

- 64-bit binaries use Microsoft's x64
  - First four arguments: RCX, RDX, R8, R9.
  - Remaining on the stack.
  - 32-byte shadow space on stack.
  - First stack argument starts at offset 0x20.
  - DXE phase binaries.

```
mov     rax, cs:1F688h
mov     dword ptr [rsp+28h], 2 <- 6th
mov     qword ptr [rsp+20h], 0 <- 5th
lea     rdx, qword_1D7A0       <- 2nd
lea     r8, [rbp+var_38]       <- 3rd
mov     rcx, rdi               <- 1st
xor     r9d, r9d               <- 4th
call    qword ptr [rax+118h]
```

Protocols & PPIs



Protocols & PPIs

- The basic services aren't enough.
- How are more services made available?
- Via Protocols and PPIs.
- Installed (published) by (U)EFI binaries.
- Others can locate and use them.

Protocols & PPIs

- Protocol (and PPI) is a data structure.
- Contains an identification, GUID.
- Optionally, function pointers and data.

```
[ Protocol ]
#define EFI_ACPI_S3_SAVE_GUID { 0x125f2de1, 0xfb85, 0x440c, 0xa5, 0x4c,
                                0x4d, 0x99, 0x35, 0x8a, 0x8d, 0x38 }


typedef struct _EFI_ACPI_S3_SAVE_PROTOCOL {
 EFI_ACPI_GET_LEGACY_MEMORY_SIZE GetLegacyMemorySize;
 EFI_ACPI_S3_SAVE S3Save;
} EFI_ACPI_S3_SAVE_PROTOCOL;

[ Function Pointers]
typedef
EFI_STATUS
(EFIAPI *EFI_ACPI_S3_SAVE)(
  IN EFI_ACPI_S3_SAVE_PROTOCOL      * This,
  IN VOID                           * LegacyMemoryAddress
  );

typedef
EFI_STATUS
(EFIAPI *EFI_ACPI_GET_LEGACY_MEMORY_SIZE)(
  IN  EFI_ACPI_S3_SAVE_PROTOCOL     * This,
  OUT UINTN                         * Size
);
```

Protocols & PPIs

- Protocols exist in DXE phase.
- PPIs exist in PEI phase.
- In practice we can assume they are equivalent.

Sample PPI usage

- First, locate the PPI.

```
EFI_STATUS      Status;
EFI_BOOT_MODE   BootMode;
PEI_CAPSULE_PPI *Capsule;

Status = (*PeiServices)->LocatePpi ((const EFI_PEI_SERVICES **)PeiServices,
                                    &gPeiCapsulePpiGuid,
                                    0,
                                    NULL,
                                    (VOID **)&Capsule
                                    );
```

Sample PPI usage

- Second, use it.

```
if (Status == EFI_SUCCESS) {
  if (Capsule->CheckCapsuleUpdate ((EFI_PEI_SERVICES**)PeiServices) == EFI_SUCCESS) {
    BootMode = BOOT_ON_FLASH_UPDATE;
    Status = (*PeiServices)->SetBootMode((const EFI_PEI_SERVICES **)PeiServices, BootMode);
    ASSERT_EFI_ERROR (Status);
  }
}
```

Sample Protocol usage

```
#define EFI_BOOT_SCRIPT_SAVE_GUID \
{ 0x470e1529, 0xb79e, 0x4e32, 0xa0, 0xfe, 0x6a,0x15, 0x6d, 0x29, 0xf9, 0xb2 }

typedef struct _EFI_BOOT_SCRIPT_SAVE_PROTOCOL {
    EFI_BOOT_SCRIPT_WRITE Write;
    EFI_BOOT_SCRIPT_CLOSE_TABLE CloseTable;
} EFI_BOOT_SCRIPT_SAVE_PROTOCOL;
```

```
.data:0000000000009D20 ; EFI_GUID gEfiBootScriptSaveProtocolGuid
.data:0000000000009D20 gEfiBootScriptSaveProtocolGuid dd 470E1529h
.data:0000000000009D20                                dw 0B79Eh
.data:0000000000009D20                                dw 4E32h
.data:0000000000009D20                                db 0A0h, 0FEh, 6Ah, 15h, 6Dh, 29h, 0F9h, 0B2h
```

```
locate_bootscript_save_protocol proc near ; CODE XREF: sub_180C+21
       push    rbp
       mov     rbp, rsp
       sub     rsp, 20h
       mov     rax, [rdx+60h] <- BootServices
       lea     rcx, gEfiBootScriptSaveProtocolGuid <- GUID to locate
       lea     r8, Boot_Script_Save_Interface <- store pointer to table
       xor     edx, edx
       call    qword ptr [rax+140h] <- BootServices->LocateProtocol()
       test    rax, rax
       jns     short loc_281
       mov     rcx, 8000000000000014h
       cmp     rax, rcx
       jz      short loc_281
       mov     cs:Boot_Script_Save_Interface, 0

loc_281:        ; CODE XREF: locate_bootscript_save_protocol+25
                ; locate_bootscript_save_protocol+34
       xor     eax, eax
       add     rsp, 20h
       pop     rbp
       retn
locate_bootscript_save_protocol endp
```

```
save_script_dispatch_opcode proc near    ; CODE XREF: sub_2D0F+6C
                                         ; sub_3C1A+83 ...
                push    rbp
                mov     rbp, rsp
                sub     rsp, 20h
                mov     r9, rdx  <- EntryPoint
                mov     rdx, 800000000000000Eh
                mov     rax, cs:Boot_Script_Save_Interface
                test    rax, rax <- NULL ptr?
                jz      short loc_3E1
                movzx   edx, cx  <- TableName
                mov     rcx, rax <- *This
                mov     r8d, 8   <- OpCode
                call    qword ptr [rax] <- BootScriptSave->Write()
                xor     edx, edx

loc_3E1:                                 ; CODE XREF: save_script_dispatch_opcode+1F
                mov     rax, rdx
                add     rsp, 20h
                pop     rbp
                retn
save_script_dispatch_opcode endp
```

Apple EFI customizations

Apple Computer Inc.

NOTICE OF PROPRIETARY PROPERTY

THE INFORMATION CONTAINED HEREIN IS THE PROPRIETARY PROPERTY OF APPLE COMPUTER, INC. THE POSSESSOR AGREES TO THE FOLLOWING

I TO MAINTAIN THE DOCUMENT IN CONFIDENCE
II NOT TO REPRODUCE OR COPY IT
III NOT TO REVEAL OR PUBLISH IN WHOLE OR PART

Apple EFI customizations

- Apple specific modifications.
- To reserved fields.
- Must be taken care of.
- Else bricked firmware.
- UEFITool v0.27+ handles everything.

EFI_FIRMWARE_VOLUME_HEADER

Summary

Describes the features and layout of the firmware volume.

Prototype

```
typedef struct {
  UINT8                     ZeroVector[16];
  EFI_GUID                  FileSystemGuid;
  UINT64                    FvLength;
  UINT32                    Signature;
  EFI_FVB_ATTRIBUTES_2      Attributes;
  UINT16                    HeaderLength;
  UINT16                    Checksum;
  UINT16                    ExtHeaderOffset;
  UINT8                     Reserved[1];
  UINT8                     Revision;
  EFI_FV_BLOCK_MAP          BlockMap[];
} EFI_FIRMWARE_VOLUME_HEADER;
```

Parameters

ZeroVector

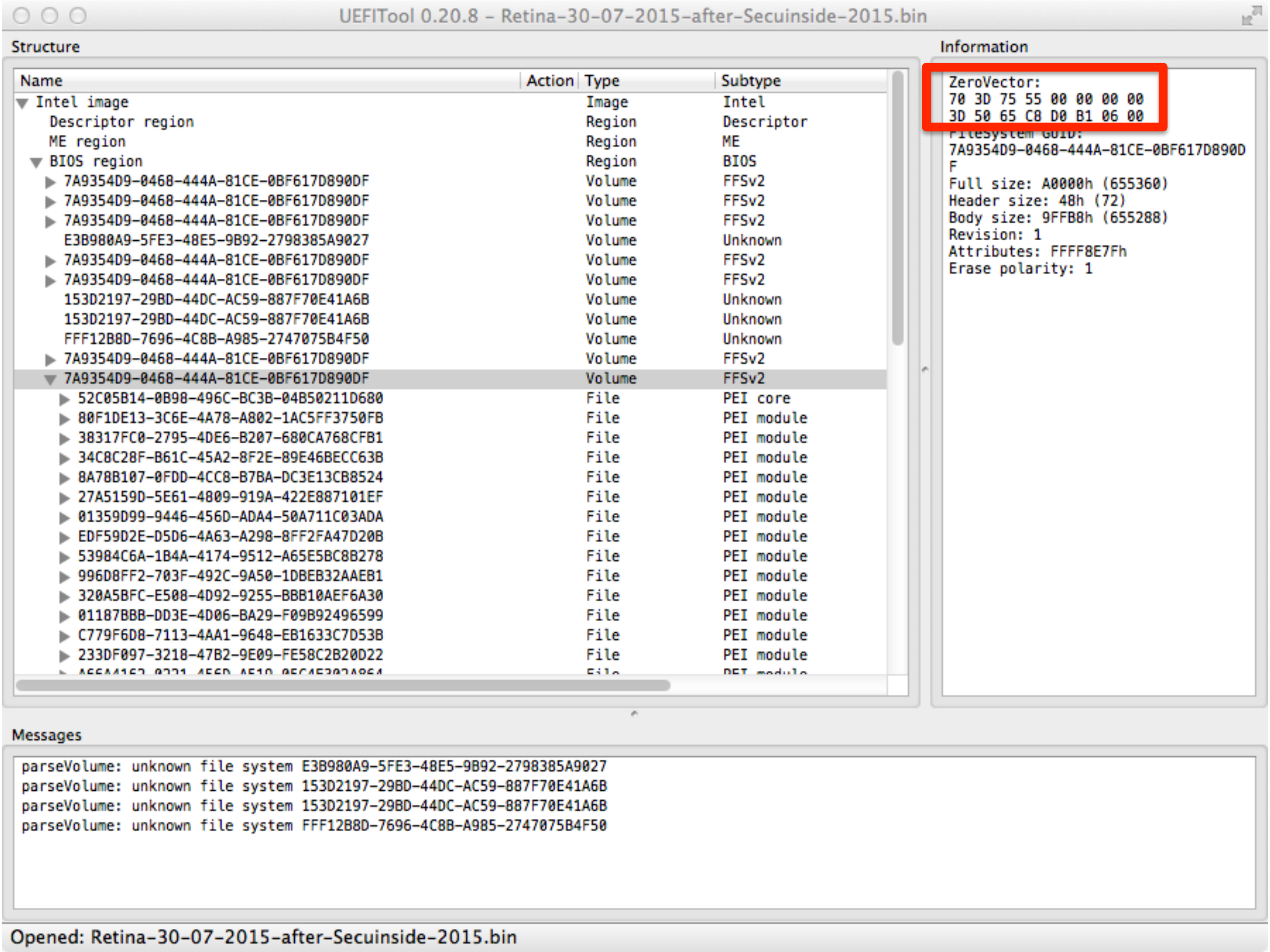
The first 16 bytes are reserved to allow for the reset vector of processors whose reset vector is at address 0.

Apple EFI customizations

- The first 8 bytes.
- Constant between firmware volumes with the same GUID.
- Changes between versions?
- Unknown meaning, doesn't seem relevant.

Apple EFI customizations

- Next 4 bytes.
- CRC32 value.
- Of the firmware volume contents.
- By spec, header got its own 16-bit checksum.



UEFITool 0.20.8 – Retina-30-07-2015-after-Secuinside-2015.bin

Structure

```
Name                                             Action  Type     Subtype
▼ Intel image                                            Image    Intel
    Descriptor region                                    Region   Descriptor
    ME region                                            Region   ME
  ▼ BIOS region                                          Region   BIOS
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF               Volume   FFSv2
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF               Volume   FFSv2
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF               Volume   FFSv2
      E3B980A9-5FE3-48E5-9B92-2798385A9027               Volume   Unknown
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF               Volume   FFSv2
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF               Volume   FFSv2
      153D2197-29BD-44DC-AC59-887F70E41A6B               Volume   Unknown
      153D2197-29BD-44DC-AC59-887F70E41A6B               Volume   Unknown
      FFF12B8D-7696-4C8B-A985-2747075B4F50               Volume   Unknown
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF               Volume   FFSv2
    ▼ 7A9354D9-0468-444A-81CE-0BF617D890DF               Volume   FFSv2
      ▶ 52C05B14-0B98-496C-BC3B-04B50211D680             File     PEI core
      ▶ 80F1DE13-3C6E-4A78-A802-1AC5FF3750FB             File     PEI module
      ▶ 38317FC0-2795-4DE6-B207-680CA768CFB1             File     PEI module
      ▶ 34C8C28F-B61C-45A2-8F2E-89E46BECC63B             File     PEI module
      ▶ 8A78B107-0FDD-4CC8-B7BA-DC3E13CB8524             File     PEI module
      ▶ 27A5159D-5E61-4809-919A-422E887101EF             File     PEI module
      ▶ 01359D99-9446-456D-ADA4-50A711C03ADA             File     PEI module
      ▶ EDF59D2E-D5D6-4A63-A298-8FF2FA47D20B             File     PEI module
      ▶ 53984C6A-1B4A-4174-9512-A65E5BC8B278             File     PEI module
      ▶ 996D8FF2-703F-492C-9A50-1DBEB32AAEB1             File     PEI module
      ▶ 320A5BFC-E508-4D92-9255-BBB10AEF6A30             File     PEI module
      ▶ 01187BBB-DD3E-4D06-BA29-F09B92496599             File     PEI module
      ▶ C779F6D8-7113-4AA1-9648-EB1633C7D53B             File     PEI module
      ▶ 233DF097-3218-47B2-9E09-FE58C2B20D22             File     PEI module
```

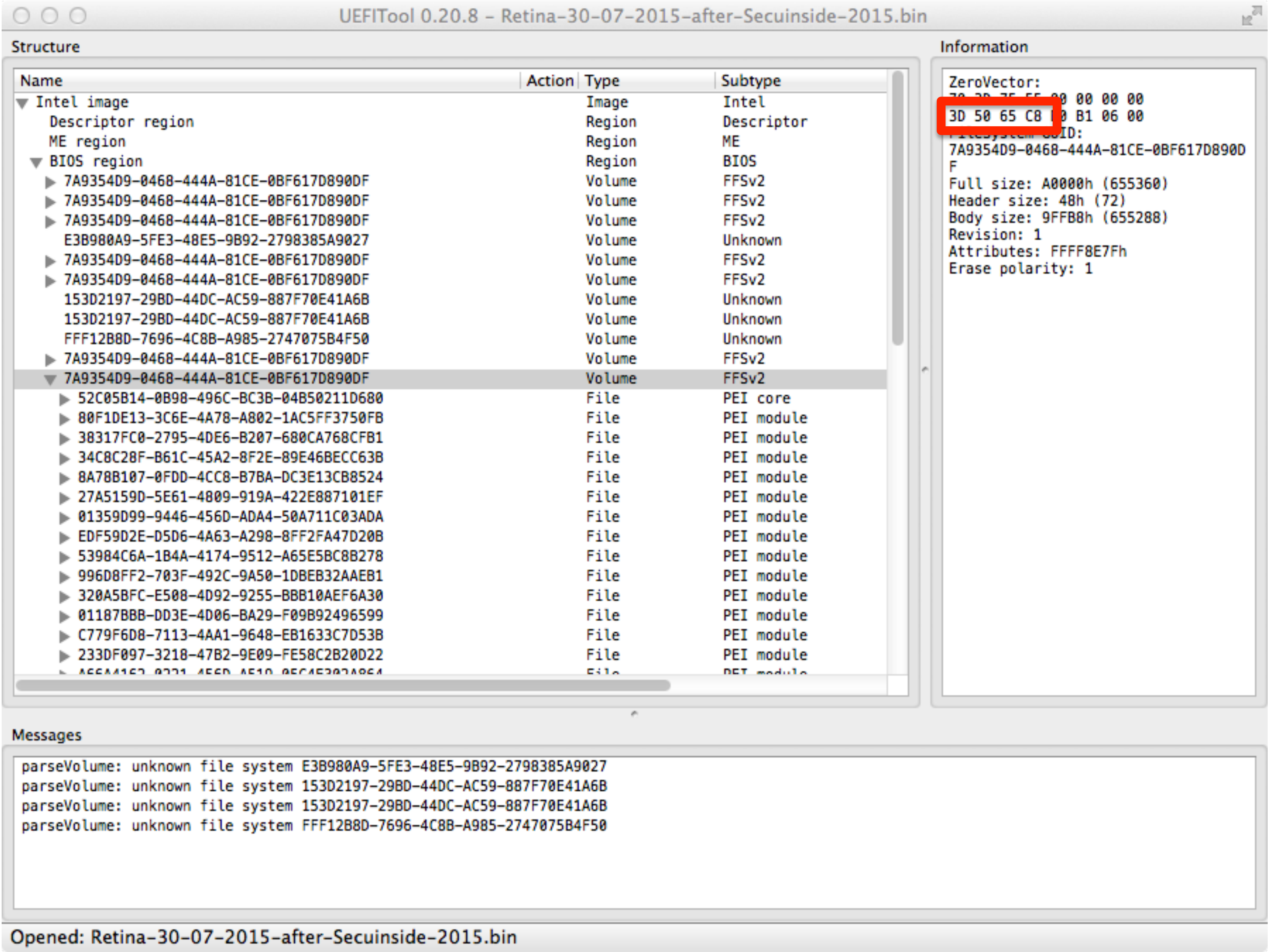
Information

```
ZeroVector:
70 3D 75 55 00 00 00 00
3D 50 65 C8 D0 B1 06 00
Filesystem GUID:
7A9354D9-0468-444A-81CE-0BF617D890D
F
Full size: A0000h (655360)
Header size: 48h (72)
Body size: 9FFB8h (655288)
Revision: 1
Attributes: FFFF8E7Fh
Erase polarity: 1
```

Messages

```
parseVolume: unknown file system E3B980A9-5FE3-48E5-9B92-2798385A9027
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system FFF12B8D-7696-4C8B-A985-2747075B4F50
```

Opened: Retina-30-07-2015-after-Secuinside-2015.bin



UEFITool 0.20.8 – Retina-30-07-2015-after-Secuinside-2015.bin

Structure

```
Name                                          Action  Type      Subtype
▼ Intel image                                         Image     Intel
    Descriptor region                                 Region    Descriptor
    ME region                                         Region    ME
  ▼ BIOS region                                       Region    BIOS
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF            Volume    FFSv2
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF            Volume    FFSv2
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF            Volume    FFSv2
      E3B980A9-5FE3-48E5-9B92-2798385A9027            Volume    Unknown
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF            Volume    FFSv2
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF            Volume    FFSv2
      153D2197-29BD-44DC-AC59-887F70E41A6B            Volume    Unknown
      153D2197-29BD-44DC-AC59-887F70E41A6B            Volume    Unknown
      FFF12B8D-7696-4C8B-A985-2747075B4F50            Volume    Unknown
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF            Volume    FFSv2
    ▼ 7A9354D9-0468-444A-81CE-0BF617D890DF            Volume    FFSv2
      ▶ 52C05B14-0B98-496C-BC3B-04B50211D680          File      PEI core
      ▶ 80F1DE13-3C6E-4A78-A802-1AC5FF3750FB          File      PEI module
      ▶ 38317FC0-2795-4DE6-B207-680CA768CFB1          File      PEI module
      ▶ 34C8C28F-B61C-45A2-8F2E-89E46BECC63B          File      PEI module
      ▶ 8A78B107-0FDD-4CC8-B7BA-DC3E13CB8524          File      PEI module
      ▶ 27A5159D-5E61-4809-919A-422E887101EF          File      PEI module
      ▶ 01359D99-9446-456D-ADA4-50A711C03ADA          File      PEI module
      ▶ EDF59D2E-D5D6-4A63-A298-8FF2FA47D20B          File      PEI module
      ▶ 53984C6A-1B4A-4174-9512-A65E5BC8B278          File      PEI module
      ▶ 996D8FF2-703F-492C-9A50-1DBEB32AAEB1          File      PEI module
      ▶ 320A5BFC-E508-4D92-9255-BBB10AEF6A30          File      PEI module
      ▶ 01187BBB-DD3E-4D06-BA29-F09B92496599          File      PEI module
      ▶ C779F6D8-7113-4AA1-9648-EB1633C7D53B          File      PEI module
      ▶ 233DF097-3218-47B2-9E09-FE58C2B20D22          File      PEI module
```

Information

```
ZeroVector:
78 3D 75 FF 00 00 00 00
3D 50 65 C8 [illegible] B1 06 00
FileSystem GUID:
7A9354D9-0468-444A-81CE-0BF617D890D
F
Full size: A0000h (655360)
Header size: 48h (72)
Body size: 9FFB8h (655288)
Revision: 1
Attributes: FFFF8E7Fh
Erase polarity: 1
```

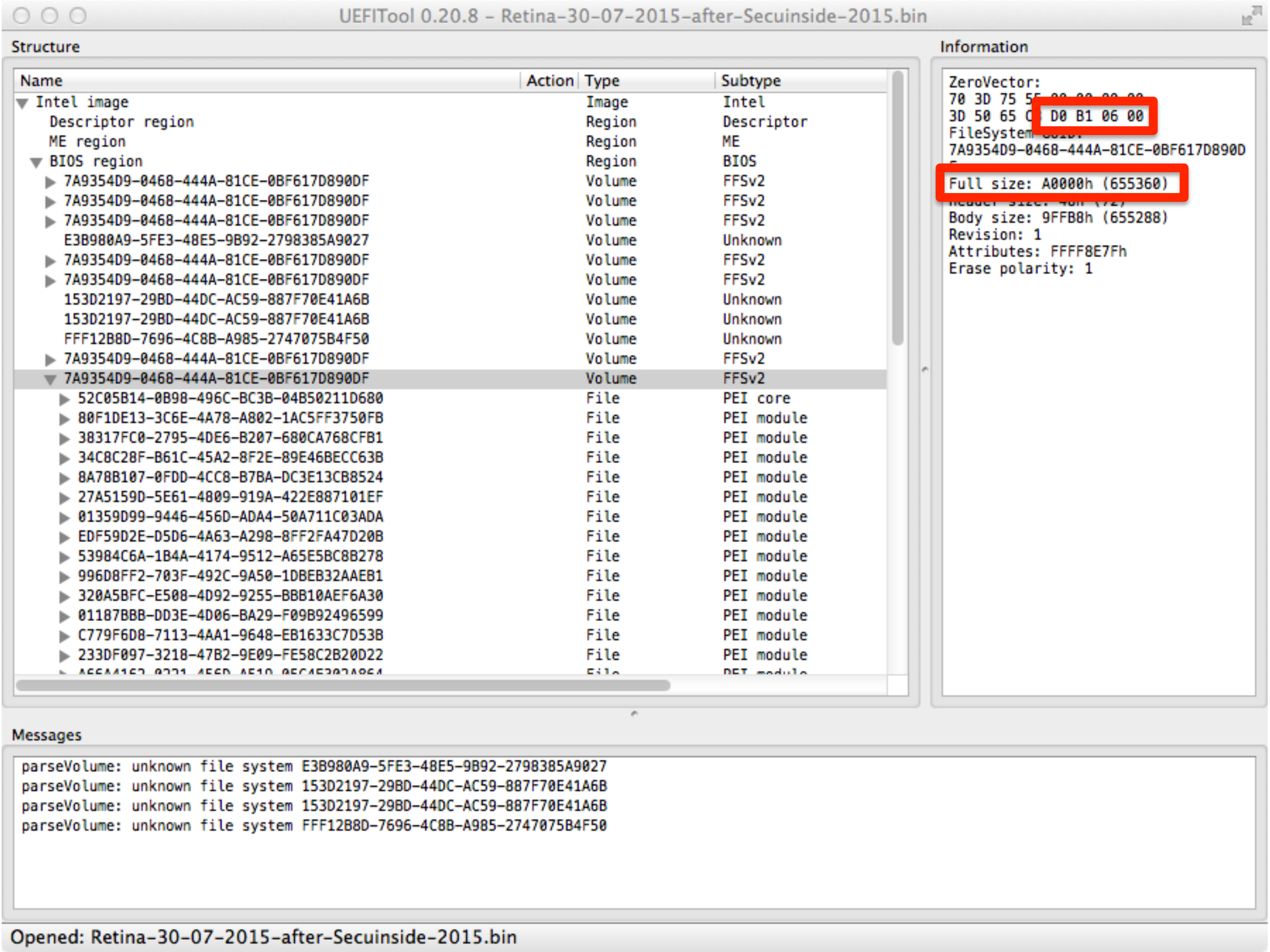
Messages

```
parseVolume: unknown file system E3B980A9-5FE3-48E5-9B92-2798385A9027
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system FFF12B8D-7696-4C8B-A985-2747075B4F50
```

Opened: Retina-30-07-2015-after-Secuinside-2015.bin

Apple EFI customizations

- Last 4 bytes.
- Total space used by firmware files.
- Must be updated if there are any modifications to volume free space.
- Bricked firmware if wrong.



UEFITool 0.20.8 – Retina-30-07-2015-after-Secuinside-2015.bin

Structure

```
Name                                              Action  Type     Subtype
▼ Intel image                                             Image    Intel
    Descriptor region                                     Region   Descriptor
    ME region                                             Region   ME
  ▼ BIOS region                                           Region   BIOS
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF                Volume   FFSv2
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF                Volume   FFSv2
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF                Volume   FFSv2
      E3B980A9-5FE3-48E5-9B92-2798385A9027                Volume   Unknown
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF                Volume   FFSv2
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF                Volume   FFSv2
      153D2197-29BD-44DC-AC59-887F70E41A6B                Volume   Unknown
      153D2197-29BD-44DC-AC59-887F70E41A6B                Volume   Unknown
      FFF12B8D-7696-4C8B-A985-2747075B4F50                Volume   Unknown
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF                Volume   FFSv2
    ▼ 7A9354D9-0468-444A-81CE-0BF617D890DF                Volume   FFSv2
      ▶ 52C05B14-0B98-496C-BC3B-04B50211D680              File     PEI core
      ▶ 80F1DE13-3C6E-4A78-A802-1AC5FF3750FB              File     PEI module
      ▶ 38317FC0-2795-4DE6-B207-680CA768CFB1              File     PEI module
      ▶ 34C8C28F-B61C-45A2-8F2E-89E46BECC63B              File     PEI module
      ▶ 8A78B107-0FDD-4CC8-B7BA-DC3E13CB8524              File     PEI module
      ▶ 27A5159D-5E61-4809-919A-422E887101EF              File     PEI module
      ▶ 01359D99-9446-456D-ADA4-50A711C03ADA              File     PEI module
      ▶ EDF59D2E-D5D6-4A63-A298-8FF2FA47D20B              File     PEI module
      ▶ 53984C6A-1B4A-4174-9512-A65E5BC8B278              File     PEI module
      ▶ 996D8FF2-703F-492C-9A50-1DBEB32AAEB1              File     PEI module
      ▶ 320A5BFC-E508-4D92-9255-BBB10AEF6A30              File     PEI module
      ▶ 01187BBB-DD3E-4D06-BA29-F09B92496599              File     PEI module
      ▶ C779F6D8-7113-4AA1-9648-EB1633C7D53B              File     PEI module
      ▶ 233DF097-3218-47B2-9E09-FE58C2B20D22              File     PEI module
```

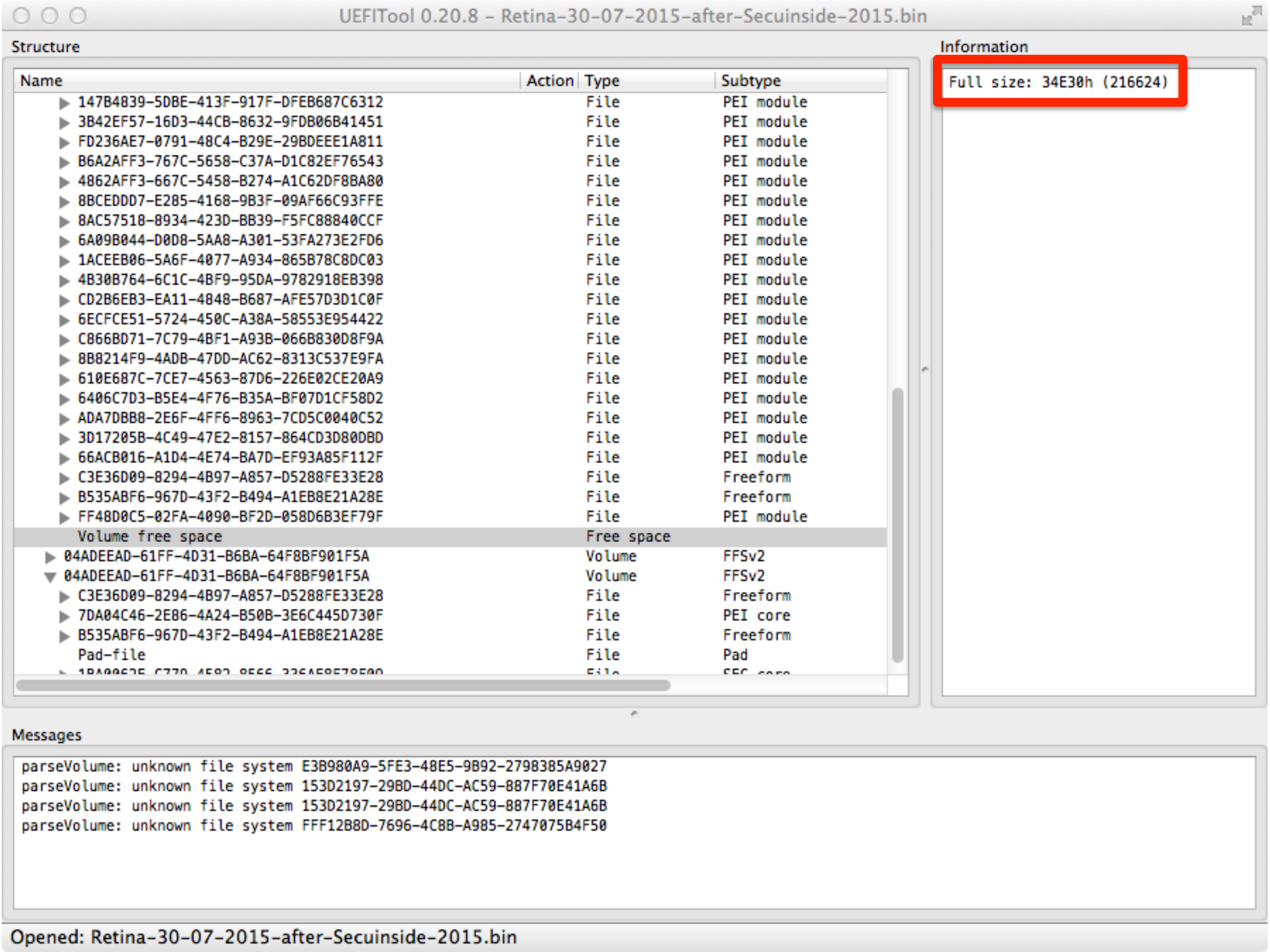
Information

```
ZeroVector:
70 3D 75 5
3D 50 65 C  D0 B1 06 00
FileSystem
7A9354D9-0468-444A-81CE-0BF617D890D
Full size: A0000h (655360)
Body size: 9FFB8h (655288)
Revision: 1
Attributes: FFFF8E7Fh
Erase polarity: 1
```

Messages

```
parseVolume: unknown file system E3B980A9-5FE3-48E5-9B92-2798385A9027
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system FFF12B8D-7696-4C8B-A985-2747075B4F50
```

Opened: Retina-30-07-2015-after-Secuinside-2015.bin



UEFITool 0.20.8 – Retina-30-07-2015-after-Secuinside-2015.bin

Structure

```
Name                                        Action  Type        Subtype
    ▶ 147B4839-5DBE-413F-917F-DFEB687C6312          File        PEI module
    ▶ 3B42EF57-16D3-44CB-8632-9FDB06B41451          File        PEI module
    ▶ FD236AE7-0791-48C4-B29E-29BDEEE1A811          File        PEI module
    ▶ B6A2AFF3-767C-5658-C37A-D1C82EF76543          File        PEI module
    ▶ 4862AFF3-667C-5458-B274-A1C62DF8BA80          File        PEI module
    ▶ 8BCEDDD7-E285-4168-9B3F-09AF66C93FFE          File        PEI module
    ▶ 8AC57518-8934-423D-BB39-F5FC88840CCF          File        PEI module
    ▶ 6A09B044-D0D8-5AA8-A301-53FA273E2FD6          File        PEI module
    ▶ 1ACEEB06-5A6F-4077-A934-865B78C8DC03          File        PEI module
    ▶ 4B30B764-6C1C-4BF9-95DA-9782918EB398          File        PEI module
    ▶ CD2B6EB3-EA11-4848-B687-AFE57D3D1C0F          File        PEI module
    ▶ 6ECFCE51-5724-450C-A38A-58553E954422          File        PEI module
    ▶ C866BD71-7C79-4BF1-A93B-066B830D8F9A          File        PEI module
    ▶ 8B8214F9-4ADB-47DD-AC62-8313C537E9FA          File        PEI module
    ▶ 610E687C-7CE7-4563-87D6-226E02CE20A9          File        PEI module
    ▶ 6406C7D3-B5E4-4F76-B35A-BF07D1CF58D2          File        PEI module
    ▶ ADA7DBB8-2E6F-4FF6-8963-7CD5C0040C52          File        PEI module
    ▶ 3D17205B-4C49-47E2-8157-864CD3D80DBD          File        PEI module
    ▶ 66ACB016-A1D4-4E74-BA7D-EF93A85F112F          File        PEI module
    ▶ C3E36D09-8294-4B97-A857-D5288FE33E28          File        Freeform
    ▶ B535ABF6-967D-43F2-B494-A1EB8E21A28E          File        Freeform
    ▶ FF48D0C5-02FA-4090-BF2D-058D6B3EF79F          File        PEI module
      Volume free space                             Free space
  ▶ 04ADEEAD-61FF-4D31-B6BA-64F8BF901F5A            Volume      FFSv2
  ▼ 04ADEEAD-61FF-4D31-B6BA-64F8BF901F5A            Volume      FFSv2
    ▶ C3E36D09-8294-4B97-A857-D5288FE33E28          File        Freeform
    ▶ 7DA04C46-2E86-4A24-B50B-3E6C445D730F          File        PEI core
    ▶ B535ABF6-967D-43F2-B494-A1EB8E21A28E          File        Freeform
      Pad-file                                      File        Pad
```

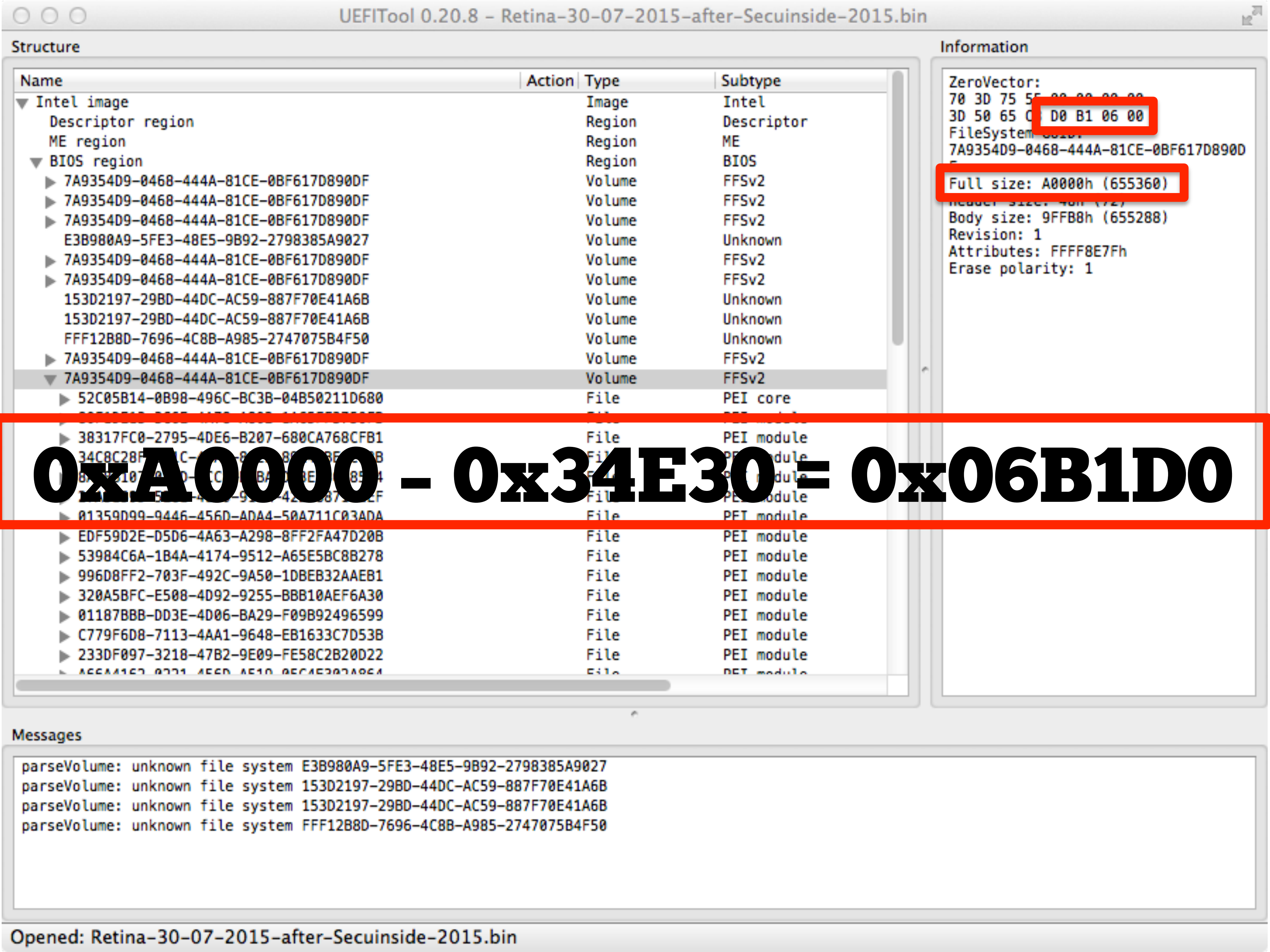
Information

```
Full size: 34E30h (216624)
```

Messages

```
parseVolume: unknown file system E3B980A9-5FE3-48E5-9B92-2798385A9027
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system FFF12B8D-7696-4C8B-A985-2747075B4F50
```

Opened: Retina-30-07-2015-after-Secuinside-2015.bin



UEFITool 0.20.8 – Retina-30-07-2015-after-Secuinside-2015.bin

Structure

```
Name                                            Action  Type     Subtype
▼ Intel image                                           Image    Intel
    Descriptor region                                   Region   Descriptor
    ME region                                           Region   ME
  ▼ BIOS region                                         Region   BIOS
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF              Volume   FFSv2
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF              Volume   FFSv2
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF              Volume   FFSv2
      E3B980A9-5FE3-48E5-9B92-2798385A9027              Volume   Unknown
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF              Volume   FFSv2
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF              Volume   FFSv2
      153D2197-29BD-44DC-AC59-887F70E41A6B              Volume   Unknown
      153D2197-29BD-44DC-AC59-887F70E41A6B              Volume   Unknown
      FFF12B8D-7696-4C8B-A985-2747075B4F50              Volume   Unknown
    ▶ 7A9354D9-0468-444A-81CE-0BF617D890DF              Volume   FFSv2
    ▼ 7A9354D9-0468-444A-81CE-0BF617D890DF              Volume   FFSv2
      ▶ 52C05B14-0B98-496C-BC3B-04B50211D680            File     PEI core
      ▶ [illegible]                                     File     PEI module
      ▶ 38317FC0-2795-4DE6-B207-680CA768CFB1            File     PEI module
      ▶ 34C8C28F[illegible]                             File     PEI module
      ▶ [illegible]                                     File     PEI module
      ▶ [illegible]                                     File     PEI module
      ▶ 01359D99-9446-456D-ADA4-50A711C03ADA            File     PEI module
      ▶ EDF59D2E-D5D6-4A63-A298-8FF2FA47D20B            File     PEI module
      ▶ 53984C6A-1B4A-4174-9512-A65E5BC8B278            File     PEI module
      ▶ 996D8FF2-703F-492C-9A50-1DBEB32AAEB1            File     PEI module
      ▶ 320A5BFC-E508-4D92-9255-BBB10AEF6A30            File     PEI module
      ▶ 01187BBB-DD3E-4D06-BA29-F09B92496599            File     PEI module
      ▶ C779F6D8-7113-4AA1-9648-EB1633C7D53B            File     PEI module
      ▶ 233DF097-3218-47B2-9E09-FE58C2B20D22            File     PEI module
```

Information

```
ZeroVector:
70 3D 75 5[illegible]
3D 50 65 C[illegible] D0 B1 06 00
FileSystem GUID:
7A9354D9-0468-444A-81CE-0BF617D890D
Full size: A0000h (655360)
Header size: 48h (72)
Body size: 9FFB8h (655288)
Revision: 1
Attributes: FFFF8E7Fh
Erase polarity: 1
```

0xA0000 – 0x34E30 = 0x06B1D0

Messages

```
parseVolume: unknown file system E3B980A9-5FE3-48E5-9B92-2798385A9027
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
parseVolume: unknown file system FFF12B8D-7696-4C8B-A985-2747075B4F50
```

Opened: Retina-30-07-2015-after-Secuinside-2015.bin

How to find EFI monsters

How to find EFI monsters

- Dump the flash contents.
  - Via hardware, if possible.
- Have a known good image.
  - A previously certified/trusted dump.
  - Or firmware updates.

How to find EFI monsters

- Firmware updates available from Apple.
- Direct downloads.
  - https://support.apple.com/en-us/HT201518
- Or combined with OS installer or updates.
- No hashes from Apple available (yet).

How to find EFI monsters

- Only useful for machines with available updates.
- Newly released machines need to wait for a firmware update.

How to find EFI monsters

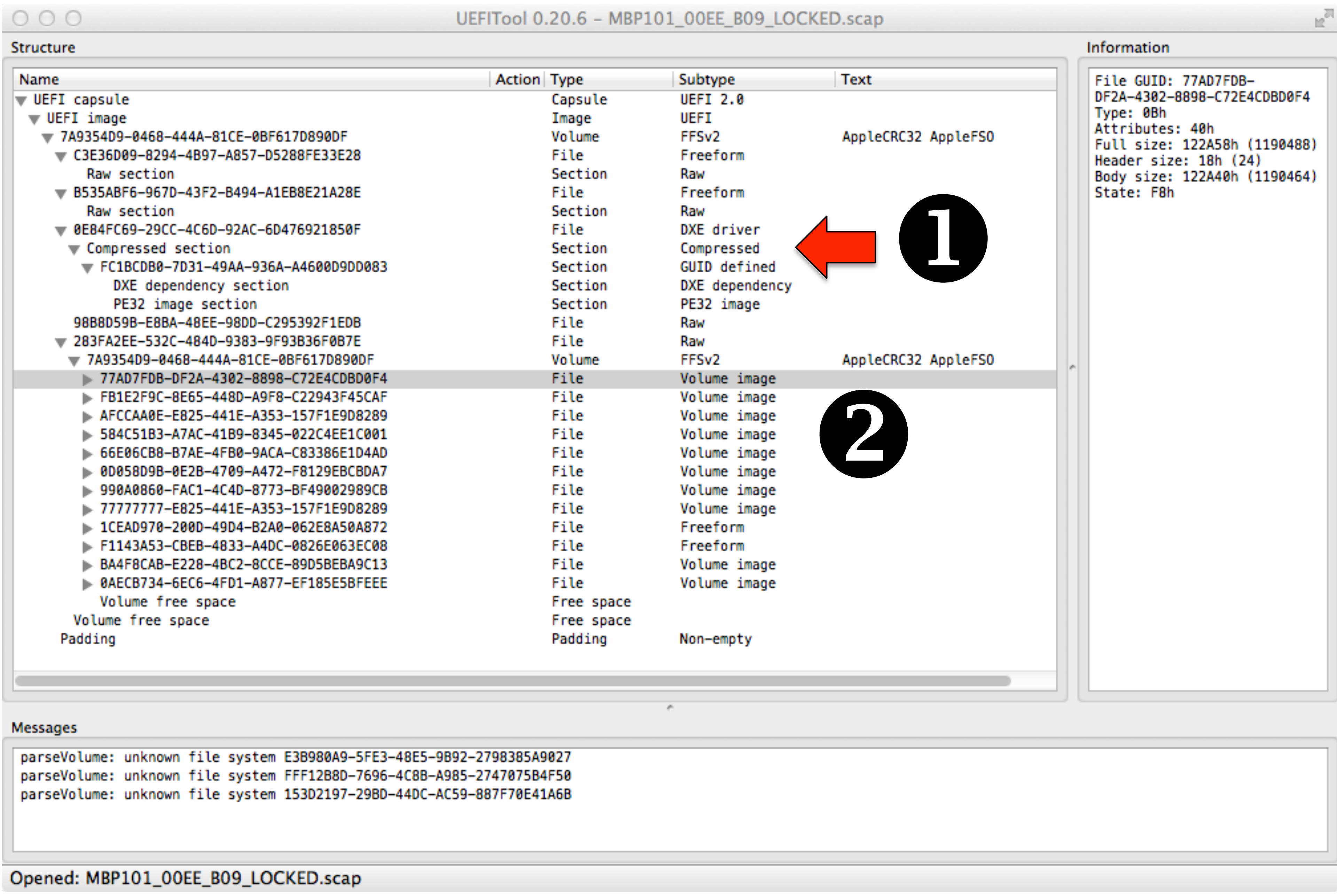
- Firmware & signatures vault
  - https://github.com/gdbinit/firmware_vault
- Signed by my PGP key.
- Extracted from available Apple updates.
- Soon, the SMC updates.

How to find EFI monsters

- Two file formats used for updates.
- SCAP (most common).
- FD (some newer and older models).
- UEFITool can process both.

SCAP

- EFI Capsule.
- Used to deliver updates.
- Recommended delivery mechanism.
- Composed by firmware volumes.
- Flash dumps parser can be reused.



UEFITool 0.20.6 – MBP101_00EE_B09_LOCKED.scap

Structure

Name	Action	Type	Subtype	Text
▼ UEFI capsule		Capsule	UEFI 2.0	
▼ UEFI image		Image	UEFI	
▼ 7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2	AppleCRC32 AppleFSO
▼ C3E36D09-8294-4B97-A857-D5288FE33E28		File	Freeform	
Raw section		Section	Raw	
▼ B535ABF6-967D-43F2-B494-A1EB8E21A28E		File	Freeform	
Raw section		Section	Raw	
▼ 0E84FC69-29CC-4C6D-92AC-6D476921850F		File	DXE driver	
▼ Compressed section		Section	Compressed	
▼ FC1BCDB0-7D31-49AA-936A-A4600D9DD083		Section	GUID defined	
DXE dependency section		Section	DXE dependency	
PE32 image section		Section	PE32 image	
98B8D59B-E8BA-48EE-98DD-C295392F1EDB		File	Raw	
▼ 283FA2EE-532C-484D-9383-9F93B36F0B7E		File	Raw	
▼ 7A9354D9-0468-444A-81CE-0BF617D890DF		Volume	FFSv2	AppleCRC32 AppleFSO
▶ 77AD7FDB-DF2A-4302-8898-C72E4CDBD0F4		File	Volume image	
▶ FB1E2F9C-8E65-448D-A9F8-C22943F45CAF		File	Volume image	
▶ AFCCAA0E-E825-441E-A353-157F1E9D8289		File	Volume image	
▶ 584C51B3-A7AC-41B9-8345-022C4EE1C001		File	Volume image	
▶ 66E06CB8-B7AE-4FB0-9ACA-C83386E1D4AD		File	Volume image	
▶ 0D058D9B-0E2B-4709-A472-F8129EBCBDA7		File	Volume image	
▶ 990A0860-FAC1-4C4D-8773-BF49002989CB		File	Volume image	
▶ 77777777-E825-441E-A353-157F1E9D8289		File	Volume image	
▶ 1CEAD970-200D-49D4-B2A0-062E8A50A872		File	Freeform	
▶ F1143A53-CBEB-4833-A4DC-0826E063EC08		File	Freeform	
▶ BA4F8CAB-E228-4BC2-8CCE-89D5BEBA9C13		File	Volume image	
▶ 0AECB734-6EC6-4FD1-A877-EF185E5BFEEE		File	Volume image	
Volume free space		Free space		
Volume free space		Free space		
Padding		Padding	Non-empty	

Information

```
File GUID: 77AD7FDB-
DF2A-4302-8898-C72E4CDBD0F4
Type: 0Bh
Attributes: 40h
Full size: 122A58h (1190488)
Header size: 18h (24)
Body size: 122A40h (1190464)
State: F8h
```

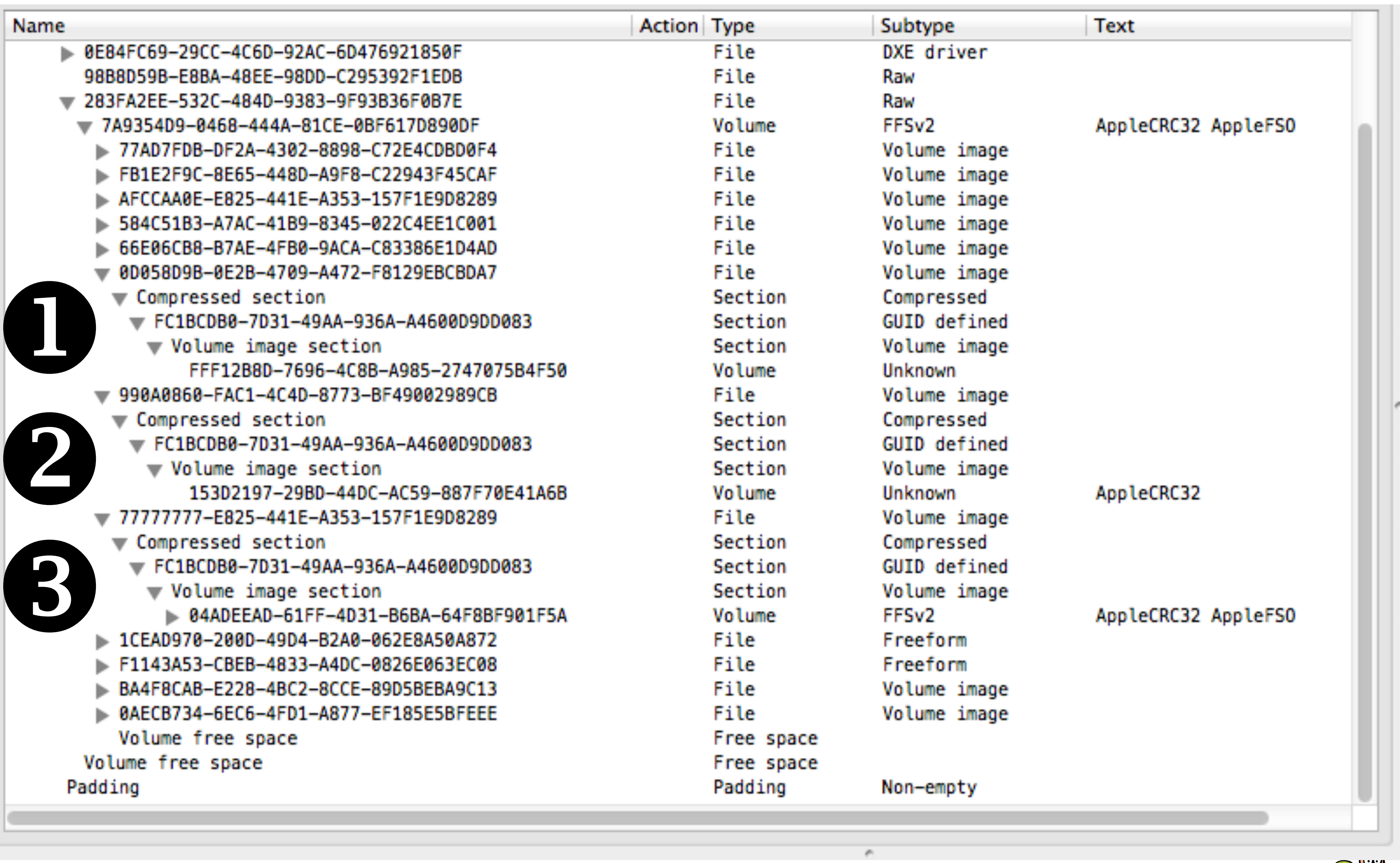
Messages

```
parseVolume: unknown file system E3B980A9-5FE3-48E5-9B92-2798385A9027
parseVolume: unknown file system FFF12B8D-7696-4C8B-A985-2747075B4F50
parseVolume: unknown file system 153D2197-29BD-44DC-AC59-887F70E41A6B
```

Opened: MBP101_00EE_B09_LOCKED.scap

SCAP

- ❶ is the EfiFlasher.efi or also known as UpdateDriverDxe.
- ❷ are the BIOS region contents.
- Encapsulated on different GUIDs.



```
Name                                              Action  Type        Subtype        Text
    ▶ 0E84FC69-29CC-4C6D-92AC-6D476921850F                File        DXE driver
      98B8D59B-E8BA-48EE-98DD-C295392F1EDB                File        Raw
    ▼ 283FA2EE-532C-484D-9383-9F93B36F0B7E                File        Raw
      ▼ 7A9354D9-0468-444A-81CE-0BF617D890DF              Volume      FFSv2          AppleCRC32 AppleFSO
        ▶ 77AD7FDB-DF2A-4302-8898-C72E4CDBD0F4            File        Volume image
        ▶ FB1E2F9C-8E65-448D-A9F8-C22943F45CAF            File        Volume image
        ▶ AFCCAA0E-E825-441E-A353-157F1E9D8289            File        Volume image
        ▶ 584C51B3-A7AC-41B9-8345-022C4EE1C001            File        Volume image
        ▶ 66E06CB8-B7AE-4FB0-9ACA-C83386E1D4AD            File        Volume image
        ▼ 0D058D9B-0E2B-4709-A472-F8129EBCBDA7            File        Volume image
          ▼ Compressed section                            Section     Compressed
❶           ▼ FC1BCDB0-7D31-49AA-936A-A4600D9DD083        Section     GUID defined
              ▼ Volume image section                      Section     Volume image
                  FFF12B8D-7696-4C8B-A985-2747075B4F50    Volume      Unknown
        ▼ 990A0860-FAC1-4C4D-8773-BF49002989CB            File        Volume image
          ▼ Compressed section                            Section     Compressed
❷           ▼ FC1BCDB0-7D31-49AA-936A-A4600D9DD083        Section     GUID defined
              ▼ Volume image section                      Section     Volume image
                  153D2197-29BD-44DC-AC59-887F70E41A6B    Volume      Unknown        AppleCRC32
        ▼ 77777777-E825-441E-A353-157F1E9D8289            File        Volume image
          ▼ Compressed section                            Section     Compressed
❸           ▼ FC1BCDB0-7D31-49AA-936A-A4600D9DD083        Section     GUID defined
              ▼ Volume image section                      Section     Volume image
                ▶ 04ADEEAD-61FF-4D31-B6BA-64F8BF901F5A    Volume      FFSv2          AppleCRC32 AppleFSO
        ▶ 1CEAD970-200D-49D4-B2A0-062E8A50A872            File        Freeform
        ▶ F1143A53-CBEB-4833-A4DC-0826E063EC08            File        Freeform
        ▶ BA4F8CAB-E228-4BC2-8CCE-89D5BEBA9C13            File        Volume image
        ▶ 0AECB734-6EC6-4FD1-A877-EF185E5BFEEE            File        Volume image
          Volume free space                               Free space
      Volume free space                                   Free space
    Padding                                               Padding     Non-empty
```

SCAP

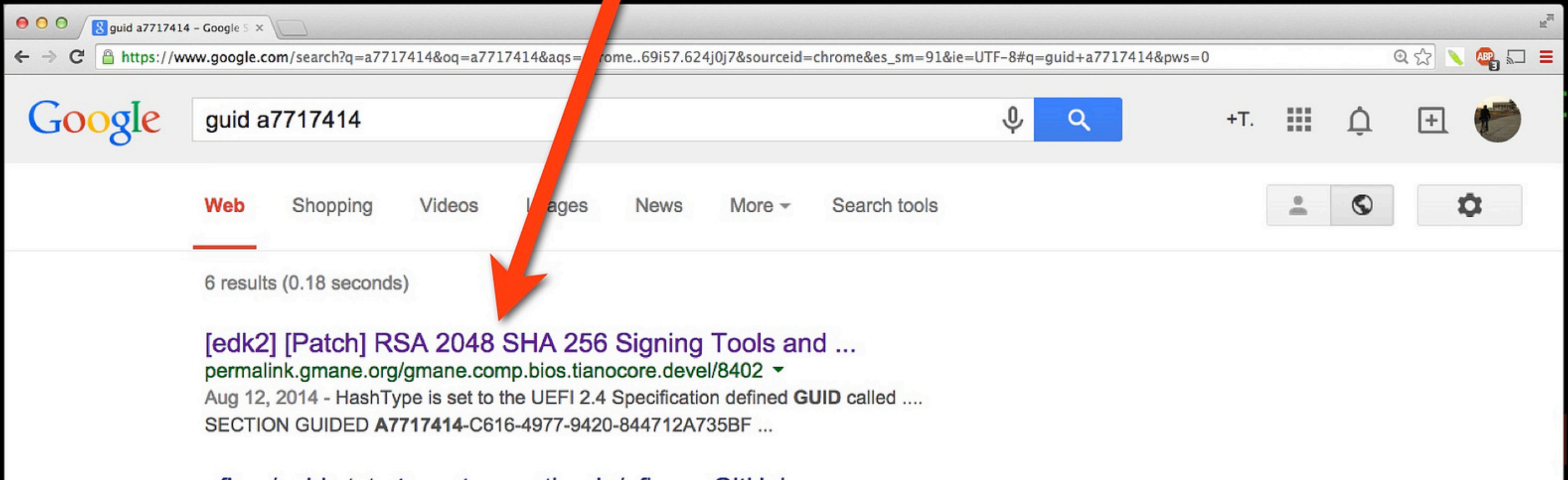
- ➊ is NVRAM region.
- ➋ is Microcode.
- ➌ is Boot volume.

SCAP

- SCAP is signed.
- RSA2048 SHA256.
- Apple backported from UEFI.
- First reported by Trammell Hudson.

```
% xxd -g 1 MBP101_00EE_B02_LOCKED.scap | tail -40 | head
0810030: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
0810040: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
0810050: 14 74 71 a7 16 c6 77 49 94 20 84 47 12 a7 35 bf  .tq...wI. .G..5.
0810060: cf fd 3e 6b fe 66 ec 75 f4 4b 7e 2e 0e d2 63 98  ..>k.f.u.K~...c.
0810070: 08 a9 8d 10 ac 37 8e 55 1c aa 0e 1c 1d 85 ef 6c  .....7.U.......l
0810080: d5 1c 75 8c 75 18 16 [illegible] 59 9f be da ef 4d 6b 0c  ..u.u...Y....Mk.
```

GUID a7717414-c616-4977-942084471 2a735bf



guid a7717414 - Google S ×

https://www.google.com/search?q=a7717414&oq=a7717414&aqs=chrome..69i57.624j0j7&sourceid=chrome&es_sm=91&ie=UTF-8#q=guid+a7717414&pws=0

guid a7717414

Web Shopping Videos Images News More Search tools

6 results (0.18 seconds)

[edk2] [Patch] RSA 2048 SHA 256 Signing Tools and ...
permalink.gmane.org/gmane.comp.bios.tianocore.devel/8402
Aug 12, 2014 - HashType is set to the UEFI 2.4 Specification defined **GUID** called ....
SECTION GUIDED **A7717414**-C616-4977-9420-844712A735BF ...

How to find EFI monsters

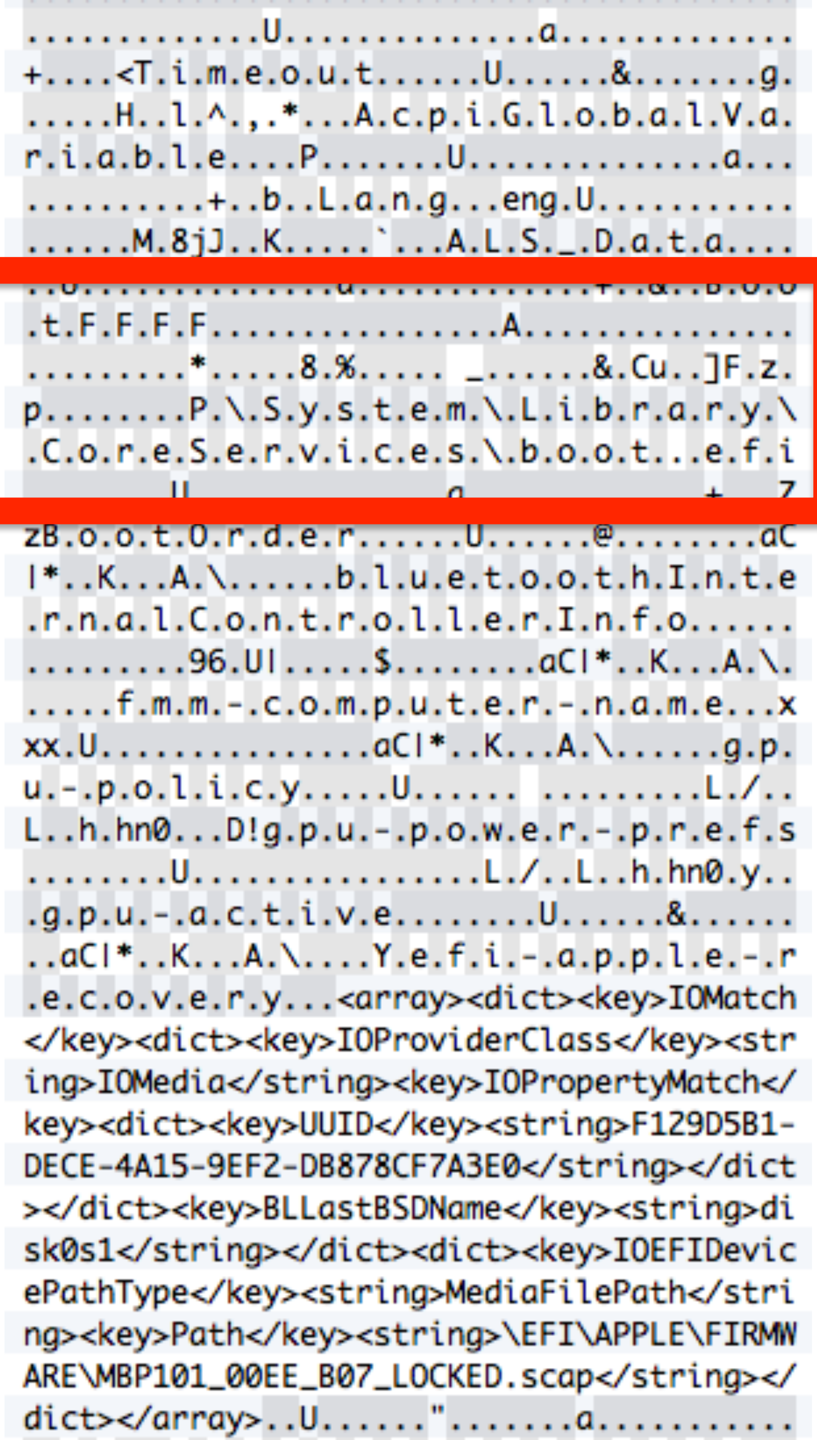
- Compare the flash dump against SCAP.
- Locate all EFI binaries in the dump.
- Checksum against SCAP contents.

How to find EFI monsters

- We also need to verify:
  - New files.
  - Missing files.
  - Free/padding space?

How to find EFI monsters

- Verify NVRAM contents!
- Boot device is stored there.
- HackingTeam had a new variable there.
  - A simple "fuse" to decide to infect or not target system.



```
.............U..............a............
+....<T.i.m.e.o.u.t......U......&.......g.
.....H..l.^.,.*...A.c.p.i.G.l.o.b.a.l.V.a.
r.i.a.b.l.e....P.......U..............a...
..........+..b..L.a.n.g...eng.U...........
......M.8jJ..K.....`...A.L.S._.D.a.t.a....
..U..............a.............+..&..B.o.o
.t.F.F.F.F...............A..............
.........*.....8.%..... _.......&.Cu..]F.z.
p........P.\.S.y.s.t.e.m.\.L.i.b.r.a.r.y.\
.C.o.r.e.S.e.r.v.i.c.e.s.\.b.o.o.t...e.f.i
........U..............a.............+...Z
zB.o.o.t.O.r.d.e.r......U......@.........aC
|*..K...A.\......b.l.u.e.t.o.o.t.h.I.n.t.e
.r.n.a.l.C.o.n.t.r.o.l.l.e.r.I.n.f.o......
.........96.U|.....$........aC|*..K...A.\.
.....f.m.m.-.c.o.m.p.u.t.e.r.-.n.a.m.e...x
xx.U...............aC|*..K...A.\......g.p.
u.-.p.o.l.i.c.y.....U...... .........L./..
L..h.hn0...D!g.p.u.-.p.o.w.e.r.-.p.r.e.f.s
........U................L./..L..h.hn0.y..
.g.p.u.-.a.c.t.i.v.e........U......&......
..aC|*..K...A.\....Y.e.f.i.-.a.p.p.l.e.-.r
.e.c.o.v.e.r.y...<array><dict><key>IOMatch
</key><dict><key>IOProviderClass</key><str
ing>IOMedia</string><key>IOPropertyMatch</
key><dict><key>UUID</key><string>F129D5B1-
DECE-4A15-9EF2-DB878CF7A3E0</string></dict
></dict><key>BLLastBSDName</key><string>di
sk0s1</string></dict><dict><key>IOEFIDevic
ePathType</key><string>MediaFilePath</stri
ng><key>Path</key><string>\EFI\APPLE\FIRMW
ARE\MBP101_00EE_B07_LOCKED.scap</string></
dict></array>..U......".......a..........
```

```
BOOLEAN
EFIAPI
CheckfTA()
{
    EFI_STATUS                    Status = EFI_SUCCESS;

    UINTN  VarDataSize;
    UINT8  VarData;

    VarData=0;
    VarDataSize=sizeof(VarData);
    Status=gRT->GetVariable(L"fTA", &gEfiGlobalFileVariableGuid, NULL, &VarDataSize, (UINTN*)&VarData);

    if(Status!=EFI_SUCCESS || VarData==0)
    {
#ifdef FORCE_DEBUG
        Print(L"Devo Infettare\n");
#endif
        return FALSE;
    }

#ifdef FORCE_DEBUG
    Print(L"NON Devo Infettare\n");
#endif
    return TRUE;
}
```

INFECT SYSTEM

DO NOT INFECT SYSTEM

How to find EFI monsters

- Don't forget boot.efi.
- Not very stealth.
- Always keep in mind that sophistication is not always required!
- If it works, why not?

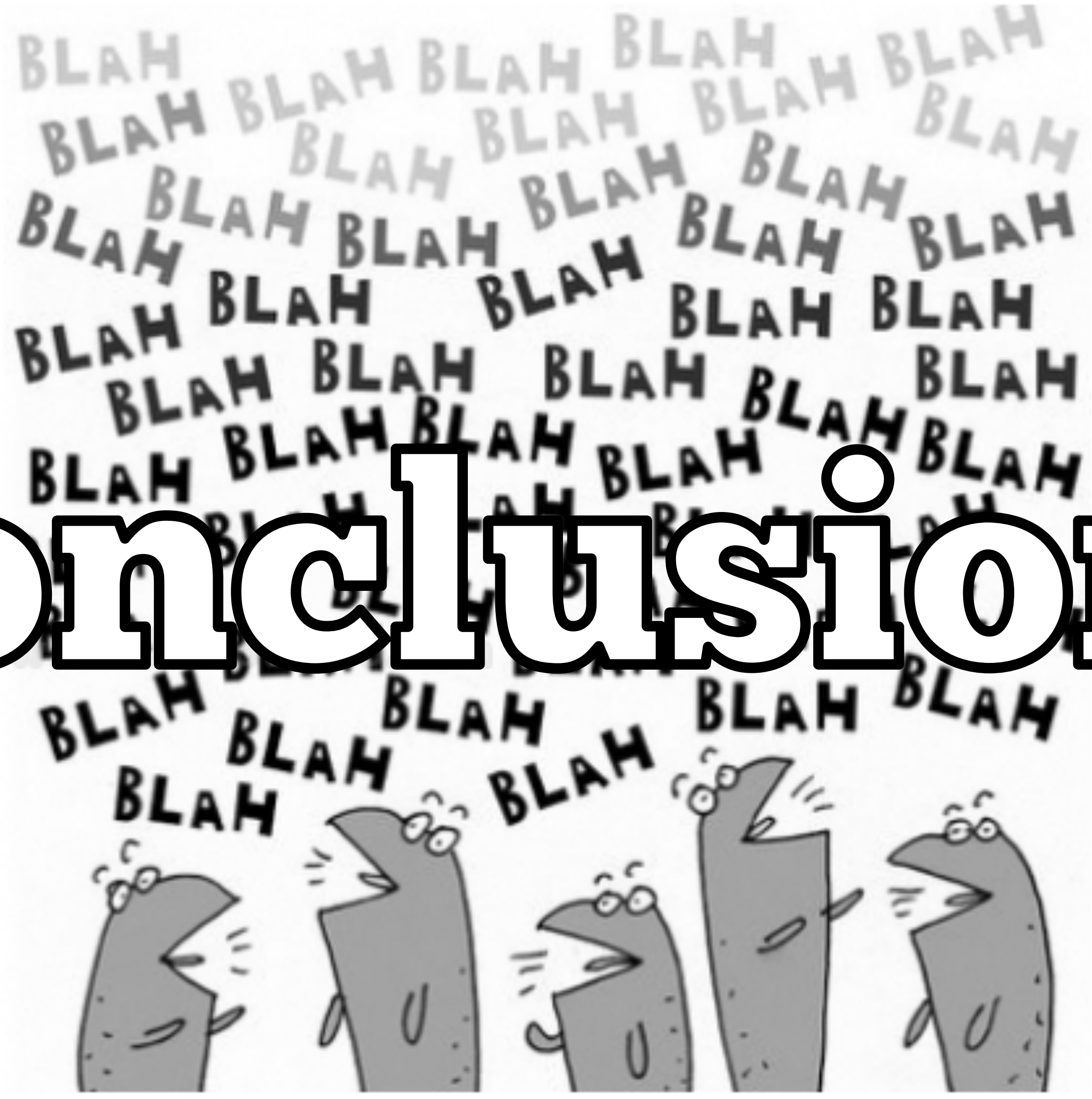
How to find EFI monsters

- SCAP is used by EfiFlasher.
- We can stitch our own firmware.
- Extract files from SCAP and build it.
- Reflash via SPI.
- Assumption that SCAP is legit.

How to find EFI monsters

- Stitch utility still in TODO list.
- Potential issues:
  - NVRAM contents?
  - Serial numbers?
- Use current dump and just replace binaries?

Conclusions



Conclusions

- (U)EFI rootkits aren't unicorns.
- Although they are very rare.
- Honestly, we don't know what's out there.
- HackingTeam developed one in 2014.
- Although it was too simple and not advanced.

Conclusions

- Chasing them requires hardware assistance.
- Disassembling computers monthly is not scalable/efficient/viable.
- How to deal with this at enterprise level?

Conclusions

- Vendors are usually slow releasing updates.
- If they ever do it.
- Check legbacore.com work.

Conclusions

- SMC is another interesting chip.
- Alex Ionescu and Andrea Barisani did some work in this area.
- Great rootkit possibilities?

Conclusions

- Intel Management Engine (ME).
- Big Pandora Box?
- Security researchers should have easier access to it.

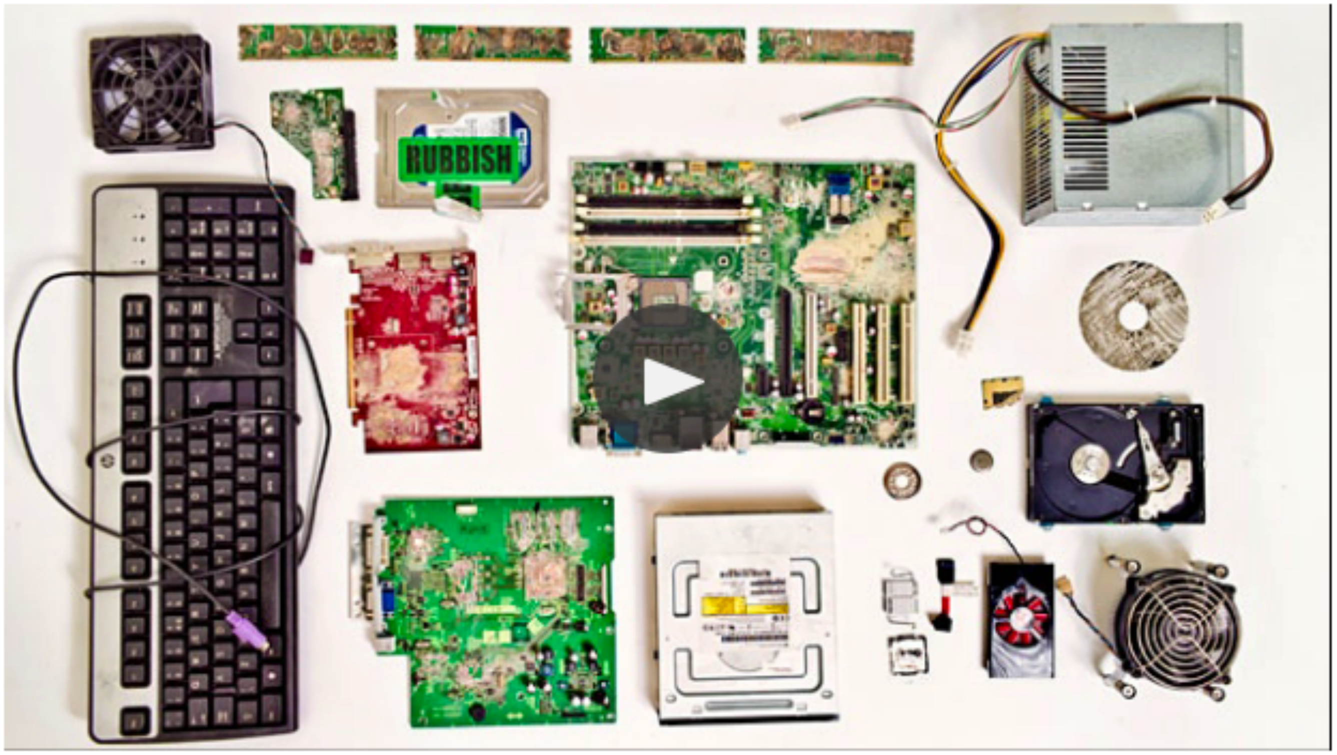
Conclusions

- Option ROMs.
- Still an issue with Apple's EFI implementation.
- No SecureBoot (signed OptionROMs).
- Check Thunderstrike 2 OptionROM worm.

Footage released of Guardian editors destroying Snowden hard drives

GCHQ technicians watched as journalists took angle grinders and drills to computers after weeks of tense negotiations

- Watch the footage of the hard drives being destroyed



New video footage has been released for the first time of the moment Guardian editors destroyed computers used to store top-secret documents leaked by the NSA whistleblower Edward Snowden.

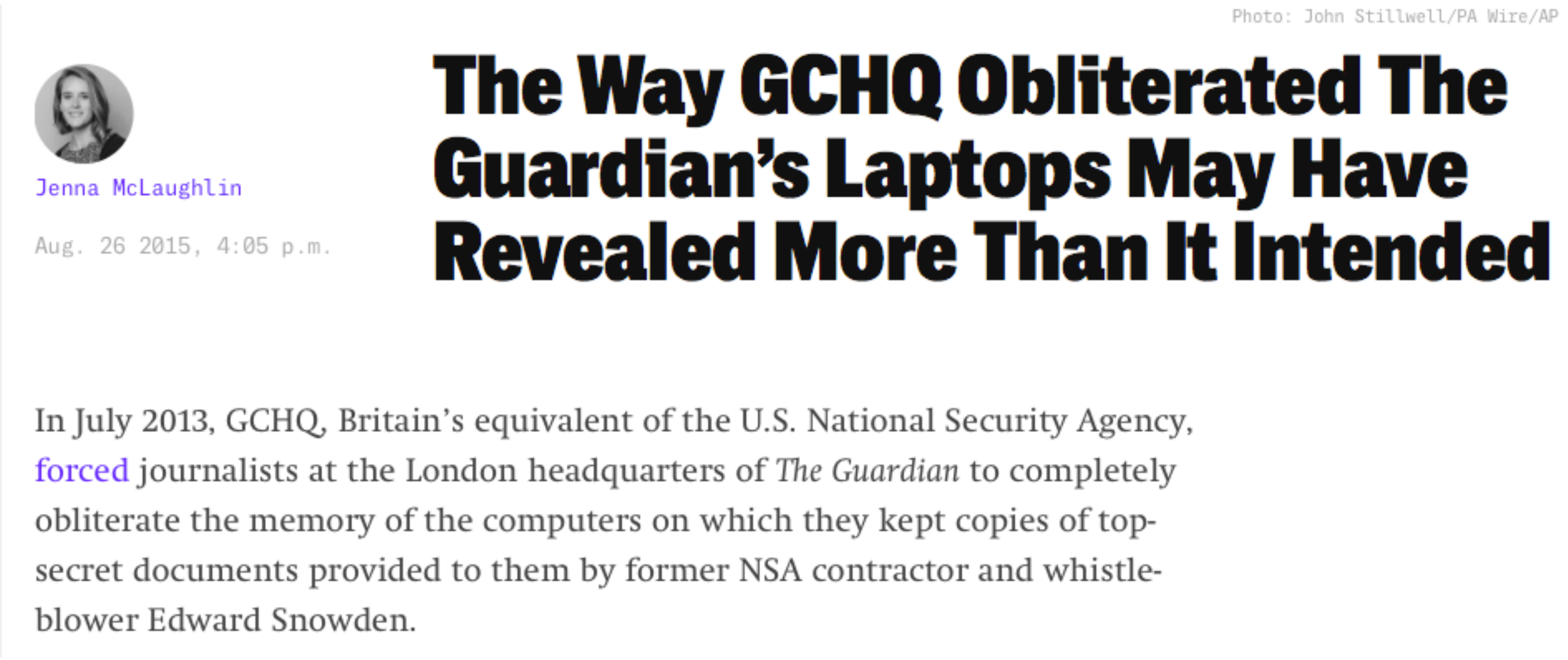


Photo: John Stillwell/PA Wire/AP

The Way GCHQ Obliterated The Guardian's Laptops May Have Revealed More Than It Intended

Jenna McLaughlin

Aug. 26 2015, 4:05 p.m.

In July 2013, GCHQ, Britain's equivalent of the U.S. National Security Agency, forced journalists at the London headquarters of *The Guardian* to completely obliterate the memory of the computers on which they kept copies of top-secret documents provided to them by former NSA contractor and whistle-blower Edward Snowden.

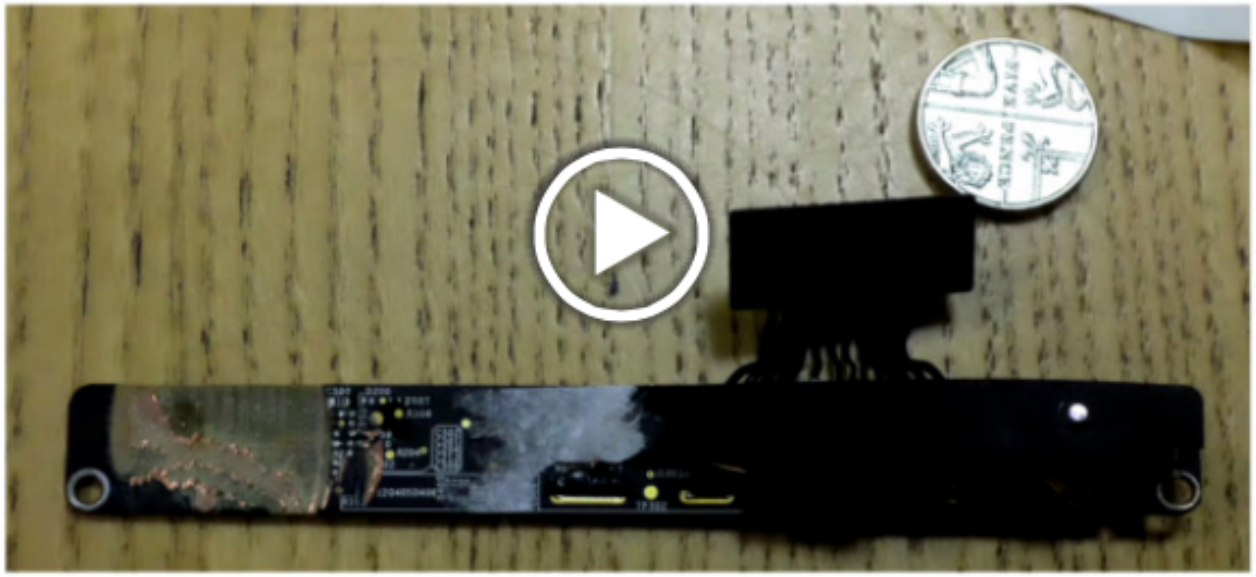
How to Destroy a Laptop with Top Secrets

How did GCHQ do it to the Guardian's copy of Snowden's files?

Mustafa Al-Bassam and Richard Tynan

Video | Audio | Download | Share

Power Controller



58 min | 2015-08-17 | 29209 | events.ccc.de

Conclusions

- Trolling?
- Real?
- Maybe a mix of both.
- Check Apple logic board schematics.
- There's a ton of interconnected stuff.

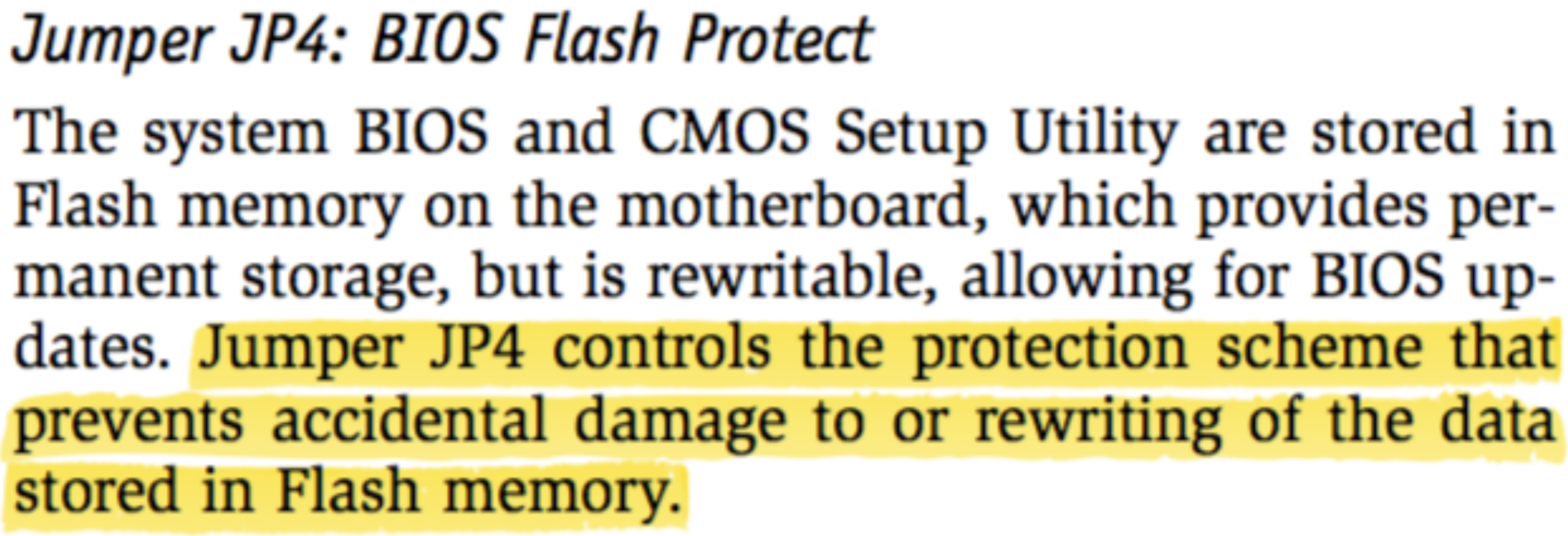
Conclusions

- We need trusted hardware solutions.
- If we can't trust hardware we are wasting a lot of time solving some software problems.

Conclusions

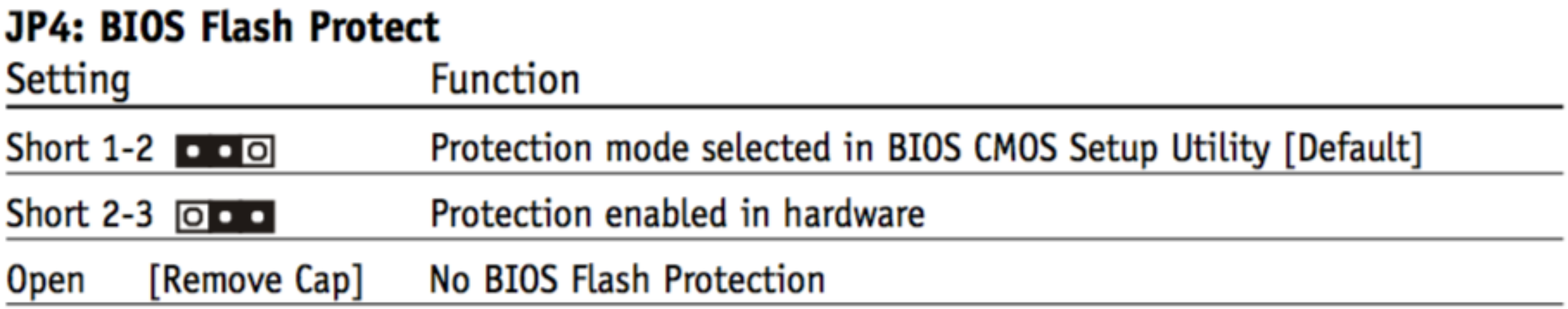
- Bring back physical protections?
- Switches to enable:
  - Flash writes.
  - MIC.
  - Camera.
  - Etc...

Conclusions



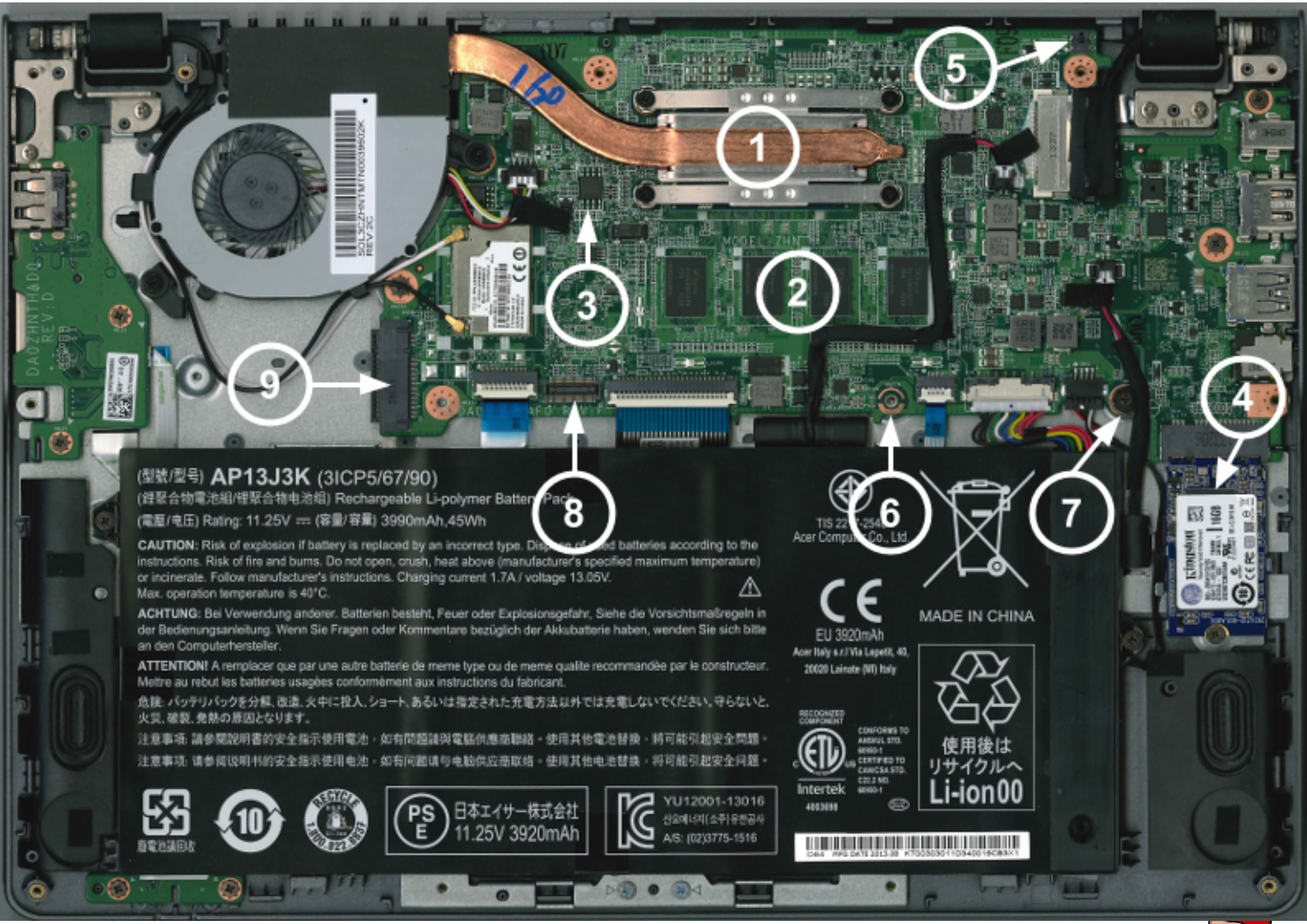
Jumper JP4: BIOS Flash Protect

The system BIOS and CMOS Setup Utility are stored in Flash memory on the motherboard, which provides permanent storage, but is rewritable, allowing for BIOS updates. Jumper JP4 controls the protection scheme that prevents accidental damage to or rewriting of the data stored in Flash memory.



JP4: BIOS Flash Protect

Setting	Function
Short 1-2 ●●○	Protection mode selected in BIOS CMOS Setup Utility [Default]
Short 2-3 ○●●	Protection enabled in hardware
Open [Remove Cap]	No BIOS Flash Protection


1
2
3
4
5
6
7
8
9
(型號/型号) AP13J3K (3ICP5/67/90)
Rechargeable Li-polymer Battery Pack
(電壓/电压) Rating: 11.25V ⎓ (容量/容量) 3990mAh,45Wh
MADE IN CHINA
EU 3920mAh
11.25V 3920mAh
Li-ion00

Conclusions

- Acer C720 & C720P Chromebook.
  - https://www.chromium.org/chromium-os/developer-information-for-chrome-os-devices/acer-c720-chromebook
- #7 is a write-protect screw.

Conclusions

- Might require new hardware design?
- NVRAM needs to be writable.
- An independent flash chip for writable regions?
- BOM/space restrictions?

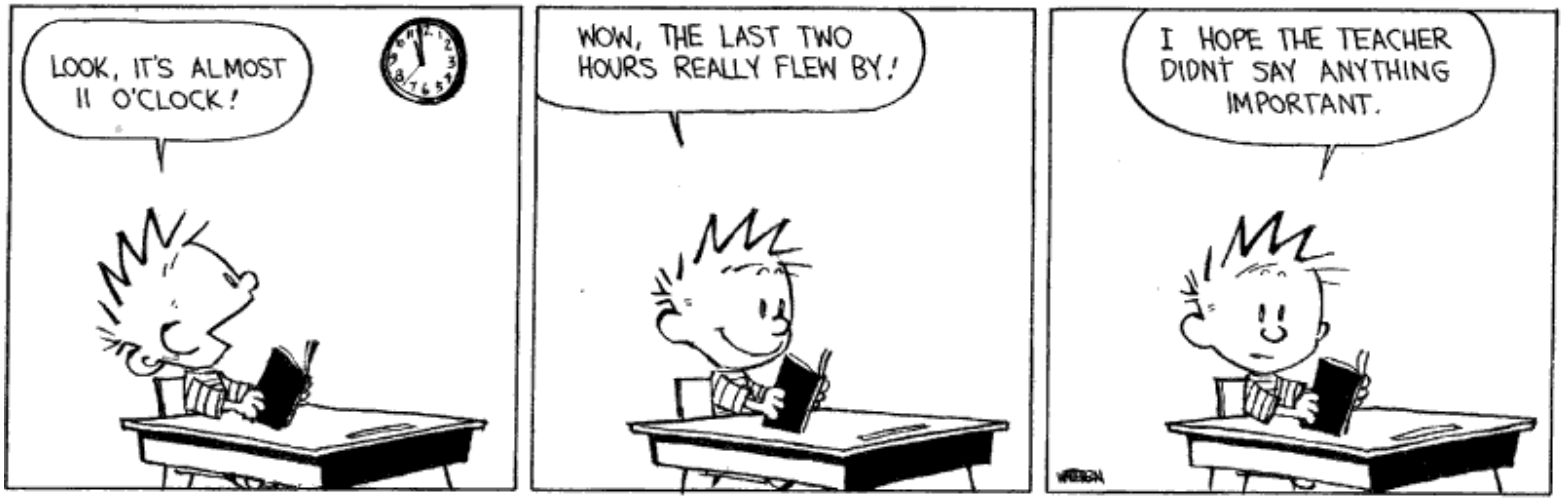
Conclusions

- Apple has a great opportunity here.
- Full control of design and supply chain.
- Can improve designs.
- Can force faster updates.
- Only matched by Chromebook?

FIRMWARE SECURITY IS CRITICAL!

Greetings

- CODE BLUE team, Snare, Trammell, Xeno, Corey, Saure, cr4sh.


LOOK, IT'S ALMOST 11 O'CLOCK!
WOW, THE LAST TWO HOURS REALLY FLEW BY!
I HOPE THE TEACHER DIDN'T SAY ANYTHING IMPORTANT.

https://reverse.put.as

https://github.com/gdbinit

reverser@put.as

@osxreverser

#osxre @ irc.freenode.net

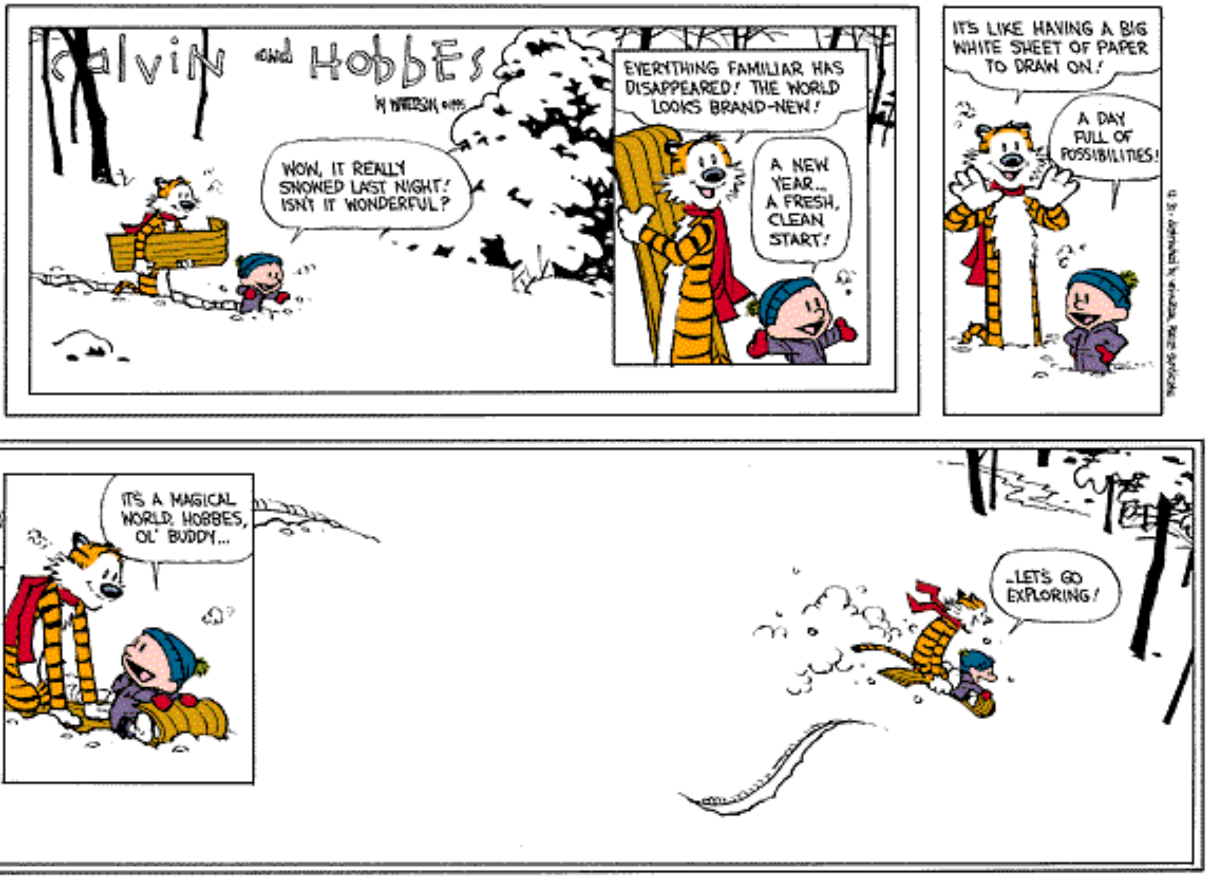
PGP key

https://reverse.put.as/wp-content/uploads/2008/06/publickey.txt

PGP Fingerprint

7B05 44D1 A1D5 3078 7F4C E745 9BB7 2A44 ED41 BF05

A day full of possibilities!


Calvin and Hobbes
by WATTERSON
WOW, IT REALLY SNOWED LAST NIGHT! ISN'T IT WONDERFUL?
EVERYTHING FAMILIAR HAS DISAPPEARED! THE WORLD LOOKS BRAND-NEW!
A NEW YEAR... A FRESH, CLEAN START!
IT'S LIKE HAVING A BIG WHITE SHEET OF PAPER TO DRAW ON!
A DAY FULL OF POSSIBILITIES!
IT'S A MAGICAL WORLD, HOBBES, OL' BUDDY...
...LET'S GO EXPLORING!

Let's go exploring!

References

- Images from images.google.com. Credit due to all their authors.
- Thunderstrike presentation
  - https://trmm.net/Thunderstrike_31c3
- Thunderstrike 2 presentation
  - https://trmm.net/Thunderstrike_2
- Snare EFI rootkits presentations
  - https://reverse.put.as/wp-content/uploads/2011/06/De_Mysteriis_Dom_Jobsivs_-_Syscan.pdf
  - https://reverse.put.as/wp-content/uploads/2011/06/De_Mysteriis_Dom_Jobsivs_Black_Hat_Slides.pdf
- Legbacore.com papers and presentations
  - http://legbacore.com/Research.html

References

- Alex Ionescu, Ninjas and Harry Potter: "Spell"unking in Apple SMC Land
  - http://www.nosuchcon.org/talks/2013/D1_02_Alex_Ninjas_and_Harry_Potter.pdf
- Alex Ionescu, Apple SMC The place to be definitely For an implant
  - https://www.youtube.com/watch?v=nSqpinjjgmg
- Andrea Barisani, Daniele Bianco, Practical Exploitation of Embedded Systems
  - http://dev.inversepath.com/download/public/embedded_systems_exploitation.pdf

References

- fG!, The Empire Strikes Back Apple – how your Mac firmware security is completely broken
    - https://reverse.put.as/2015/05/29/the-empire-strikes-back-apple-how-your-mac-firmware-security-is-completely-broken/
- fG!, Reversing Prince Harming's kiss of death
    - https://reverse.put.as/2015/07/01/reversing-prince-harmings-kiss-of-death/
- Cr4sh, Exploiting UEFI boot script table vulnerability
    - http://blog.cr4.sh/2015_02_01_archive.html

References

- Cr4sh, Building reliable SMM backdoor for UEFI based platforms
  - http://blog.cr4.sh/2015/07/building-reliable-smm-backdoor-for-uefi.html
- Firmware papers and presentations timeline
  - http://timeglider.com/timeline/5ca2daa6078caaf4
- Archive of OS X/iOS and firmware papers & presentations
  - https://reverse.put.as/papers/
- Intel ATR - Black Hat 2015 / Def Con 23 - Firmware rootkit
  - https://www.youtube.com/watch?v=sJnliPN0104&app=desktop