Is there an EFI monster inside your apple?

fG! @ SECUINSIDE 2015

Who am I?

- SECUINSIDE 2012.
- Messing around with Macs since 2007.
- Not a Mac Zealot!
- Love all kinds of rootkits.

SentinelOne

Whats Up Doc?

EFI Monsters?

- Introduction to EFI.
- How to
  - Reverse EFI binaries.
  - Search for EFI rootkits.

WE LIVE IN A CHANGING WORLD

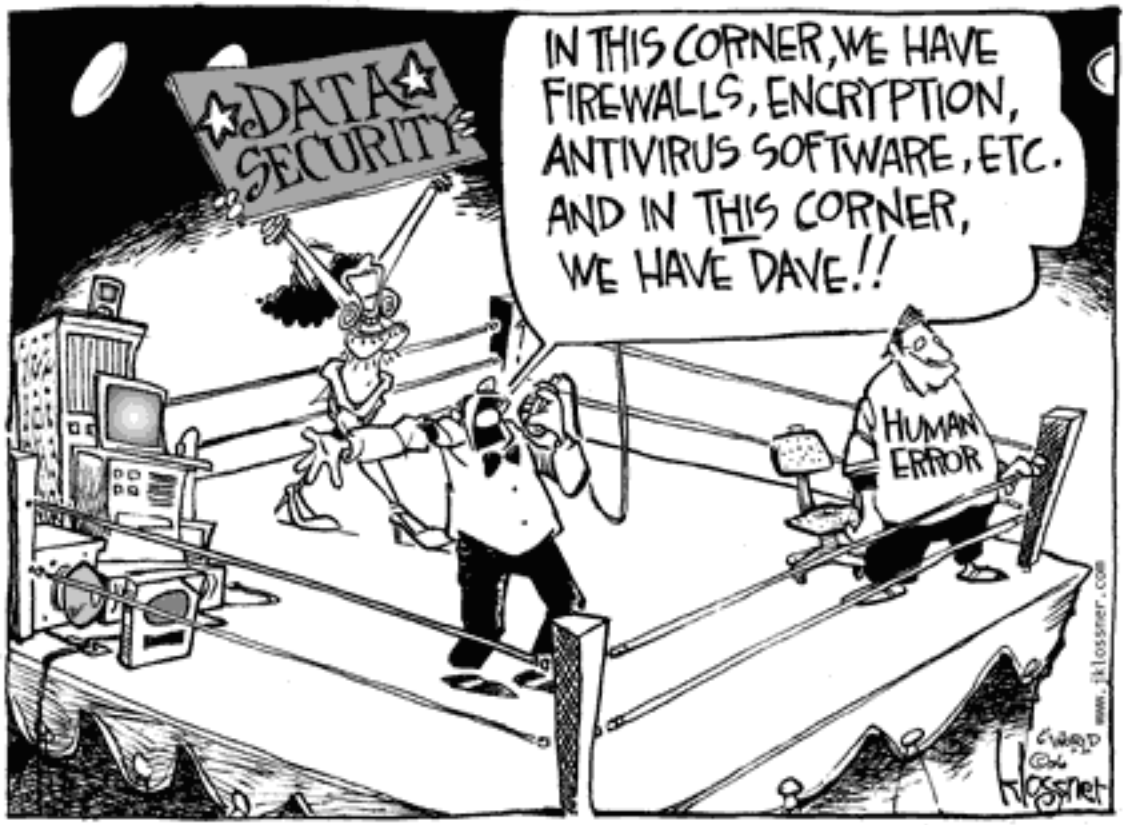
ASSUMPTIONS

"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 chipset only.
- No Core 2 Duo or older.

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
Klossner

Why EFI?

- BIOS replacement.
- Initially developed by Intel.
- Now UEFI (managed by UEFI consortium).
- Initializes your machine.
- Access to low level features.

What can we do?

What can we do?

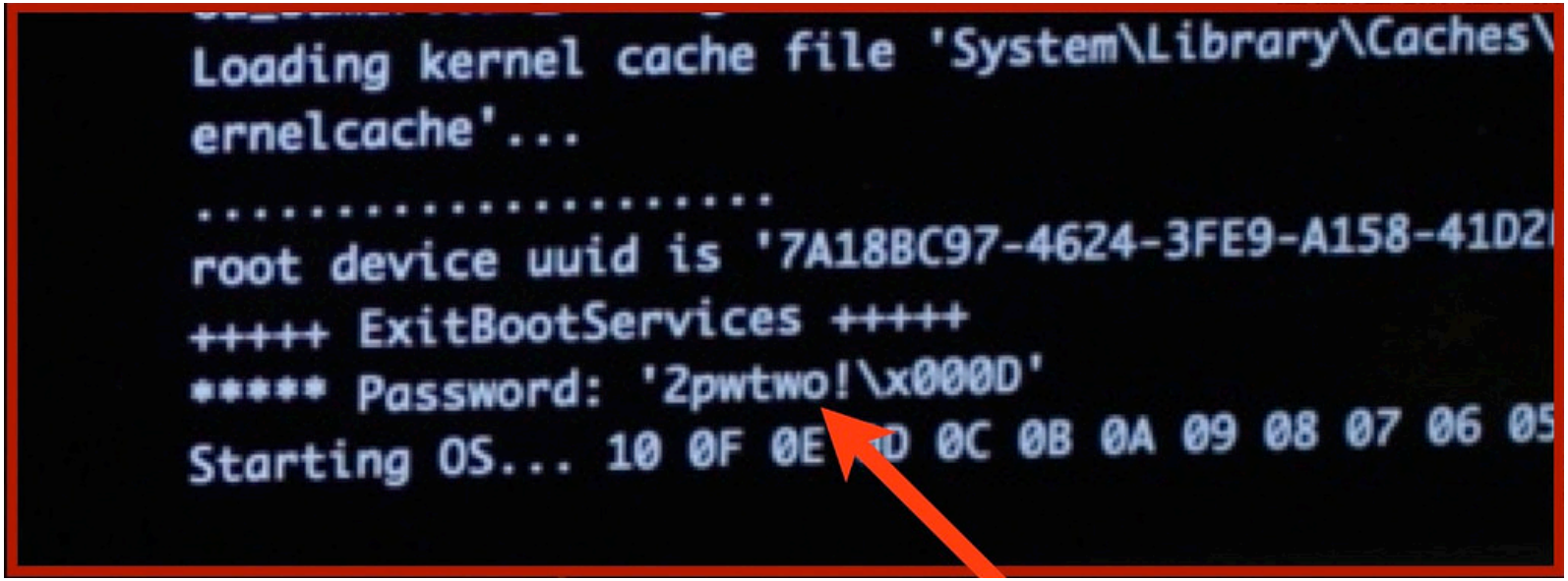
- Diskless rootkits.
- Persist across operating system reinstalls.
- Bypass full-disk encryption.
- And so on...

What can we do?

- HackingTeam built a UEFI rootkit.
  - https://github.com/hackedteam/vector-edk
  - https://github.com/informationextraction/vector-edk/blob/master/MdeModulePkg/Application/fsbg/fsbg.c
- Only for persistence across reinstalls.

What can we do?

- Full-disk encryption
  - Install a keylogger.
  - Recover FileVault2 unlock password.



What can we do?

- Attack "secure" operating systems
  - Tails.
  - Recover PGP keys/passphrases.
  - https://www.youtube.com/watch?v=sNYsfUNegEA.

What 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 weeks 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 machines are vulnerable.
- Haswell or newer are not vulnerable.
- 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			

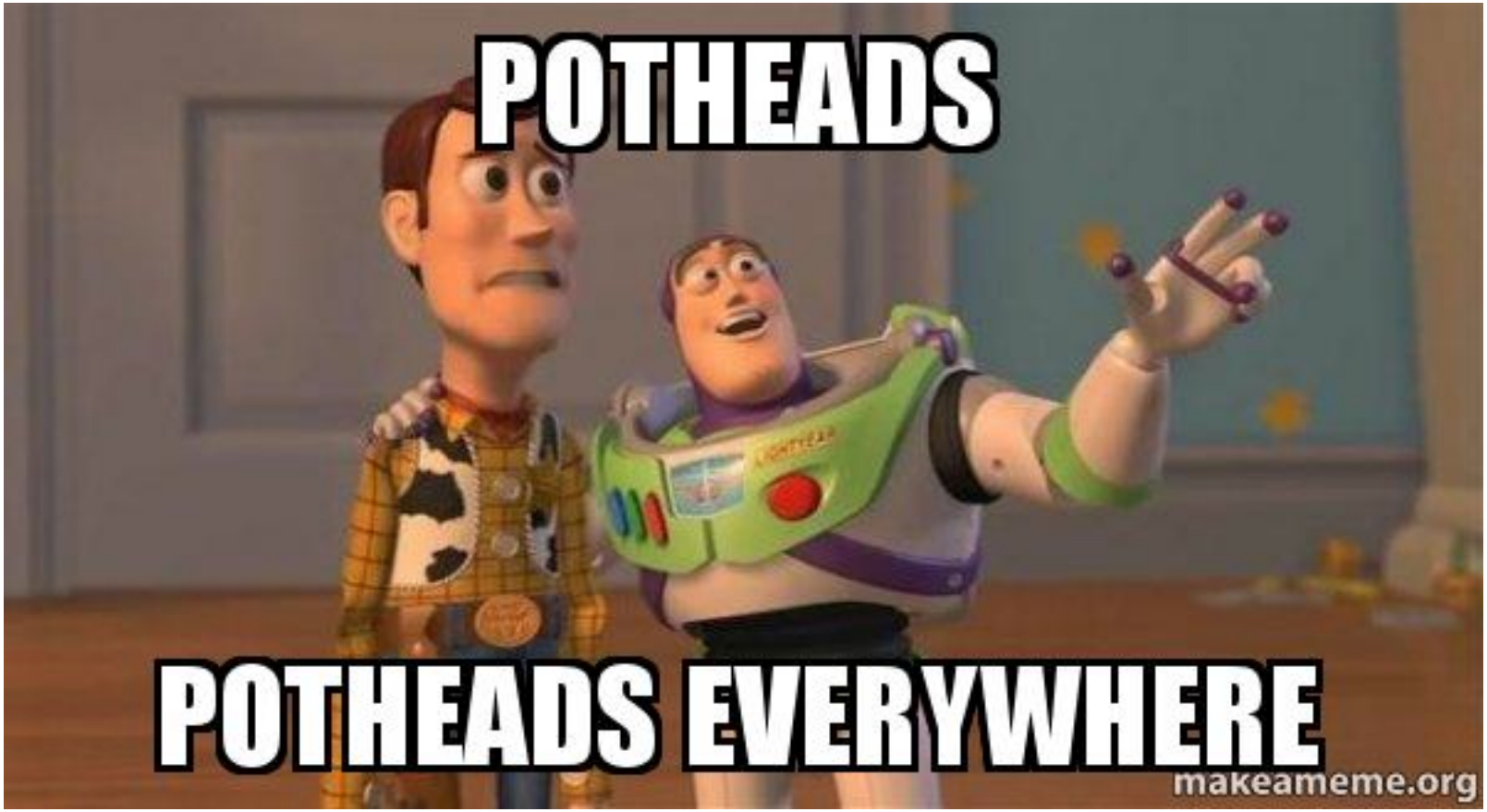
A zero day story...

- 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.

Apple …



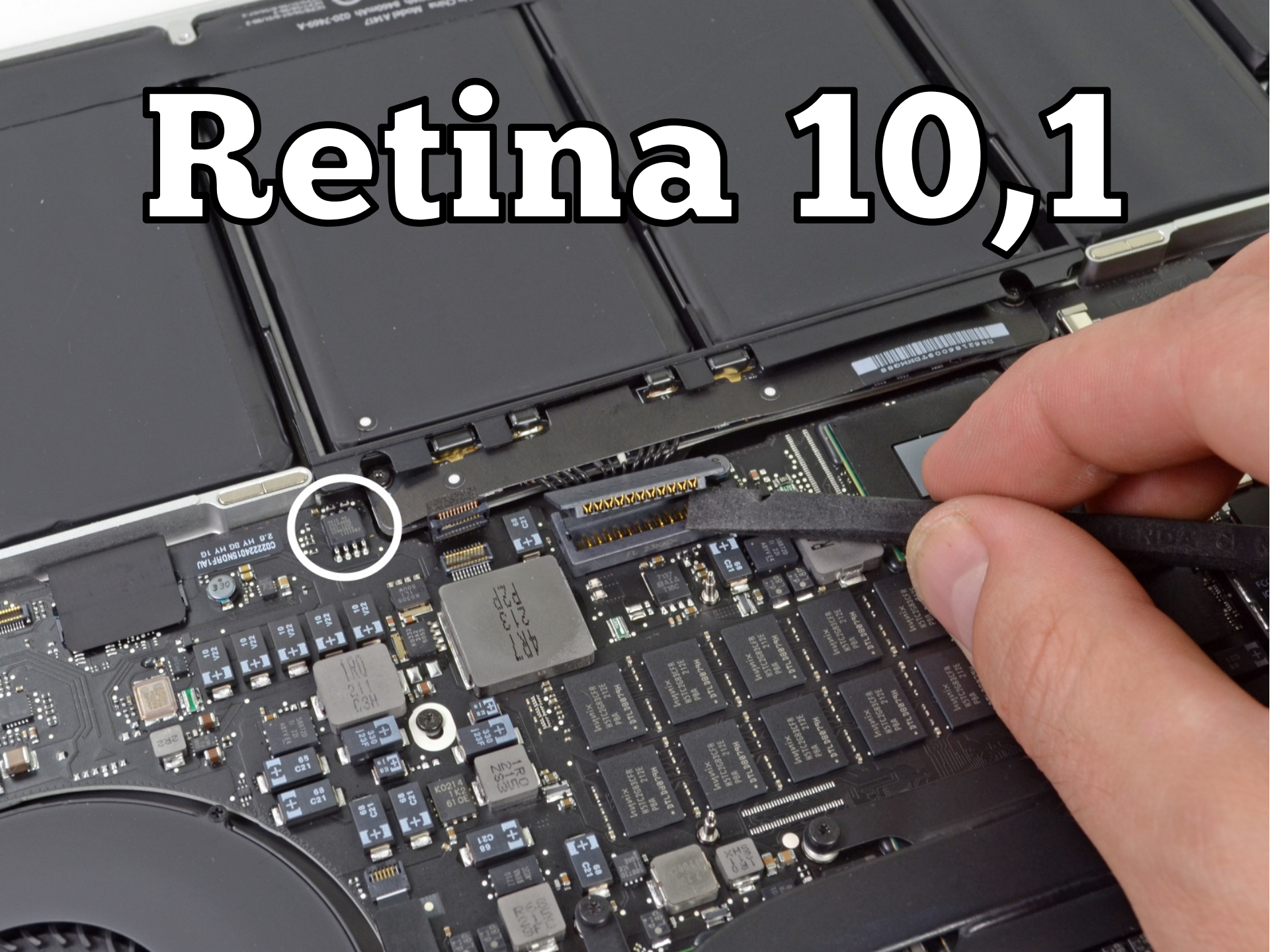

Where is
EFI?

Where's EFI

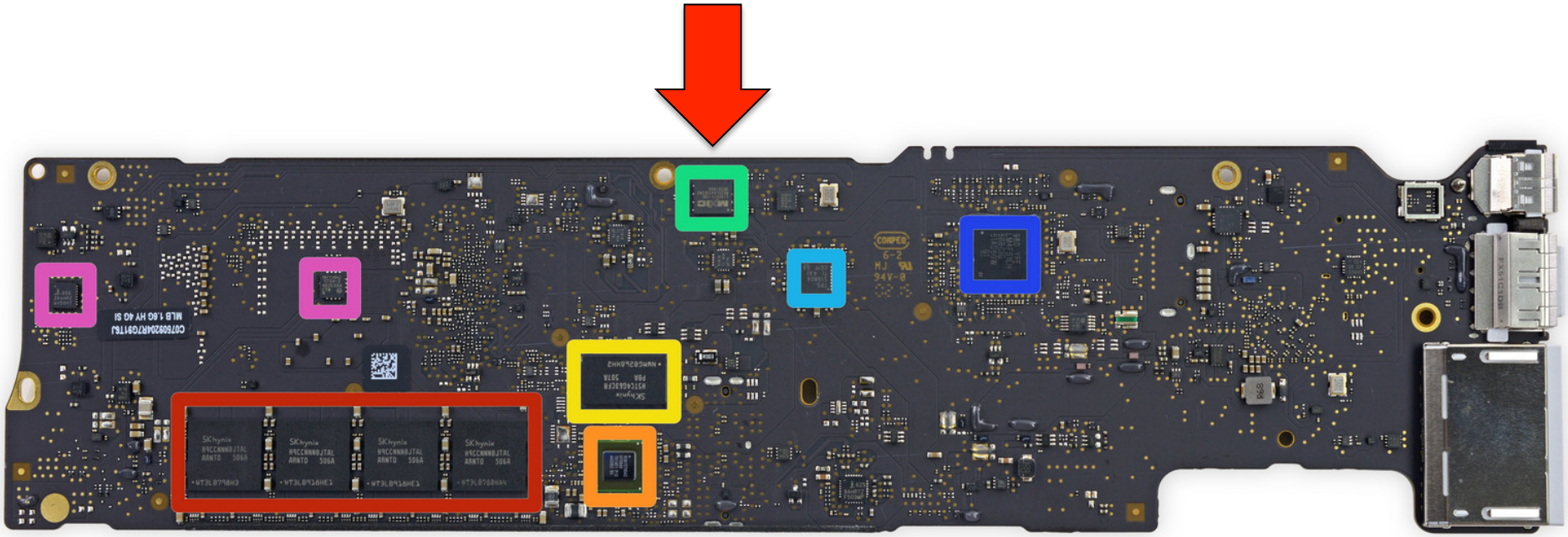
- Usually stored in a CMOS serial flash.
- Two popular chips
  - Macronix MX25L6406E.
  - Micron N25Q064A.

- SPI compatible.
- Almost all are 64 Mbits/8 Mbytes.

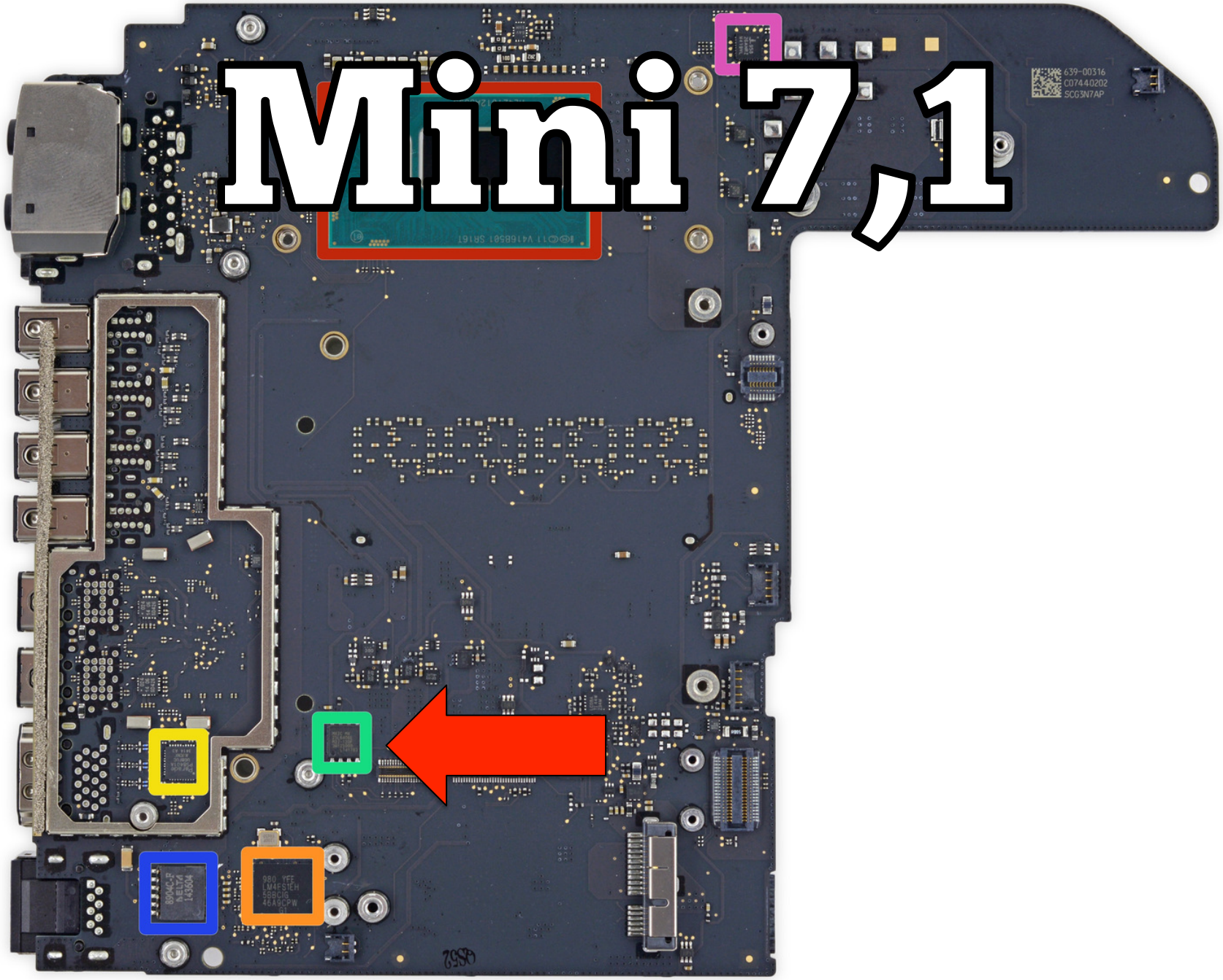
Retina 10,1



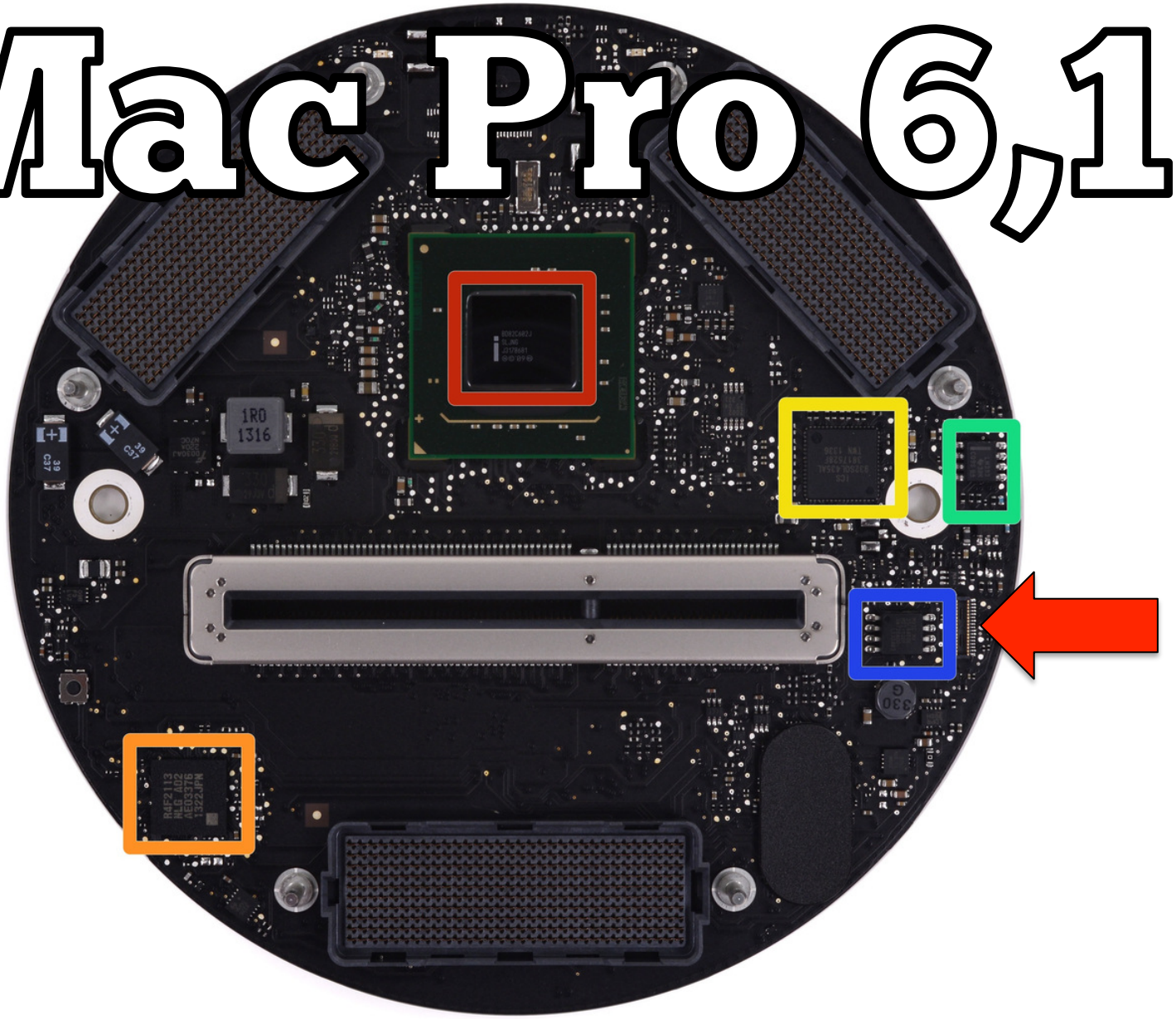
Air 7,2



Mini 7,1



Mac Pro 6,1



Where's 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.

Where's EFI

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

Where's EFI

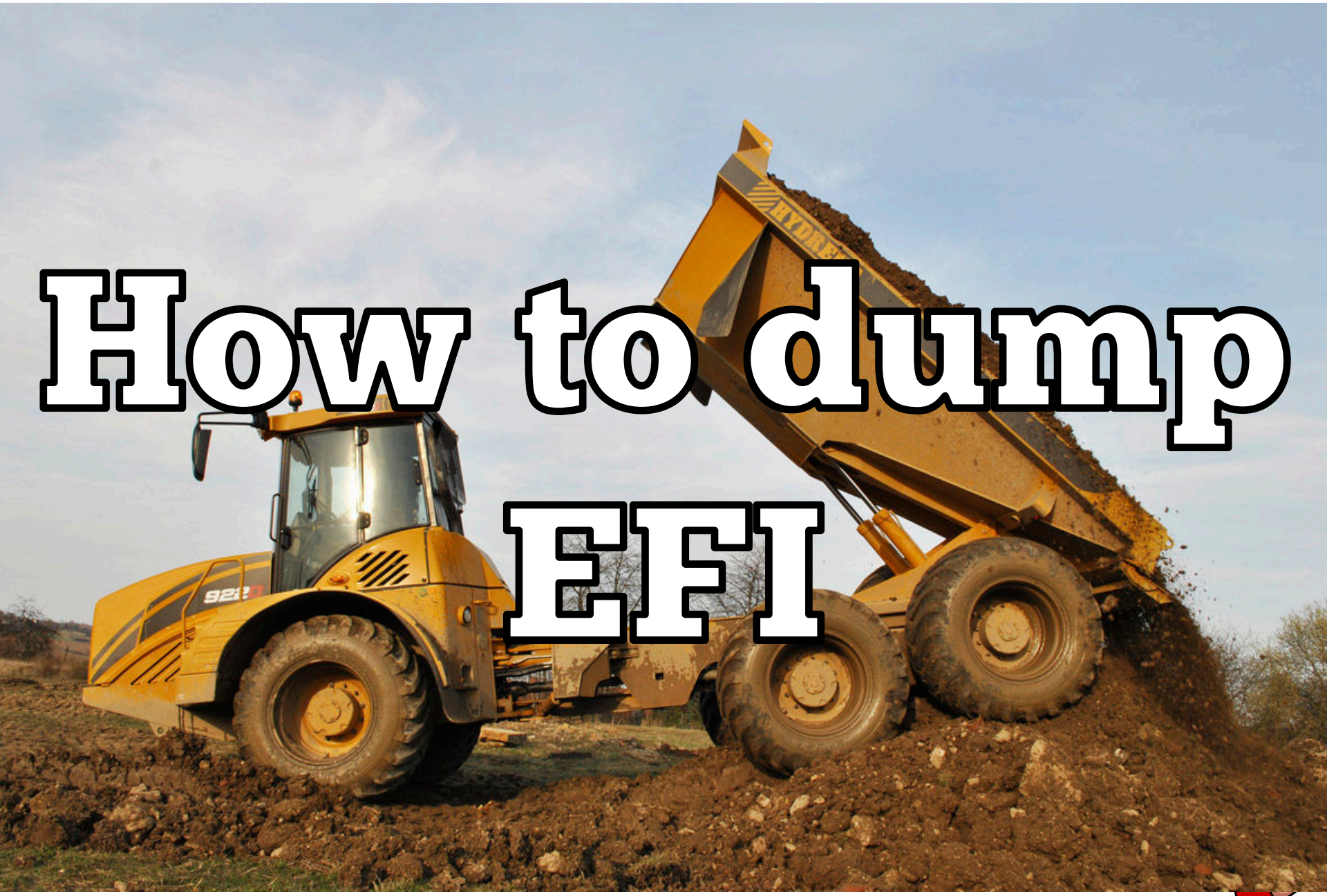
- Newer machines flash chip(s)
  - Winbond W25Q64FV.
- Chip list from EfiFlasher.efi:

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's EFI

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

How to dump EFI



How to dump EFI

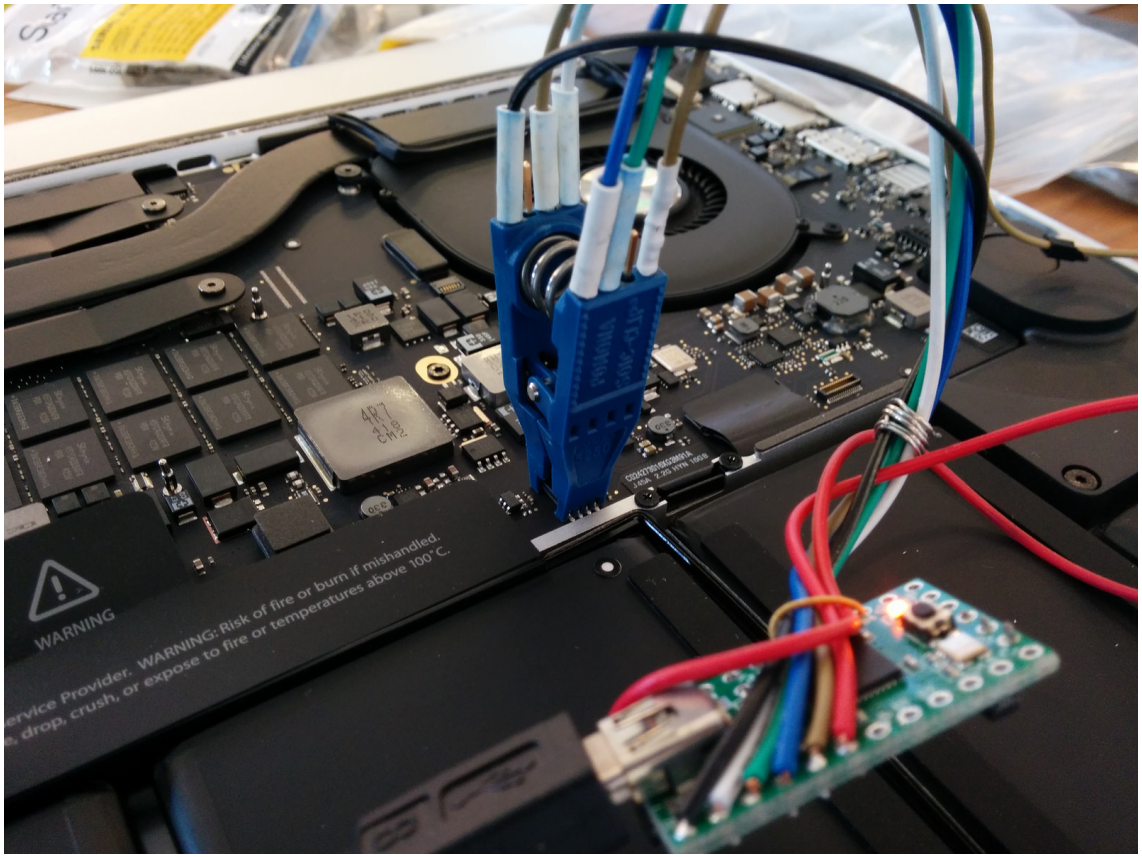
- Hardware
  - The best and most reliable way.
  - Trustable.
- Software
  - Possible if chip supported by flashrom.
  - Not (very) trustable.

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.



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   | O |   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   | O |   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
- Both available in DarwinDumper.
  - Apple trusts the packaged DirectHW.kext.

Software

- http://flashrom.org/Flashrom
- http://www.coreboot.org/DirectHW
- https://bitbucket.org/blackosx/darwindumper/downloads

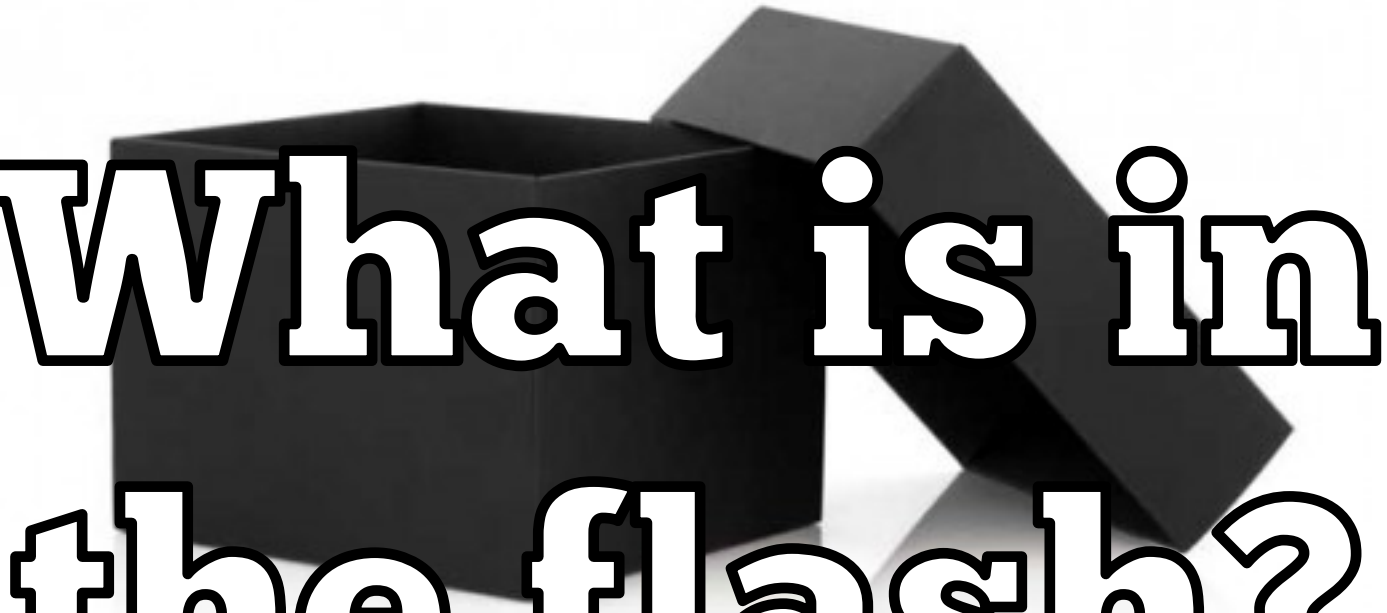
```
sh-3.2# kextload DirectHW.kext/

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

- Good enough to play around.
- Mostly useless to chase EFI rootkits.
- Unless rootkit is made by HackingTeam!
  - The leaked source code 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.
- Few presentations by Igor Skochinsky.
- Definitely requires more research!
- Unpacker
  - http://io.smashthestack.org/me/

Intel ME region

- http://me.bios.io/images/c/ca/Rootkit_in_your_laptop.pdf
- https://ruxconbreakpoint.com/assets/2014/slides/bpx-Breakpoint%202014%20Skochinsky.pdf
- http://recon.cx/2014/slides/Recon%202014%20Skochinsky.pdf

BIOS region

- Contains
  - EFI binaries for different phases.
  - NVRAM.
  - Microcode.
- 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
+----------------------------------------+
```

BIOS region

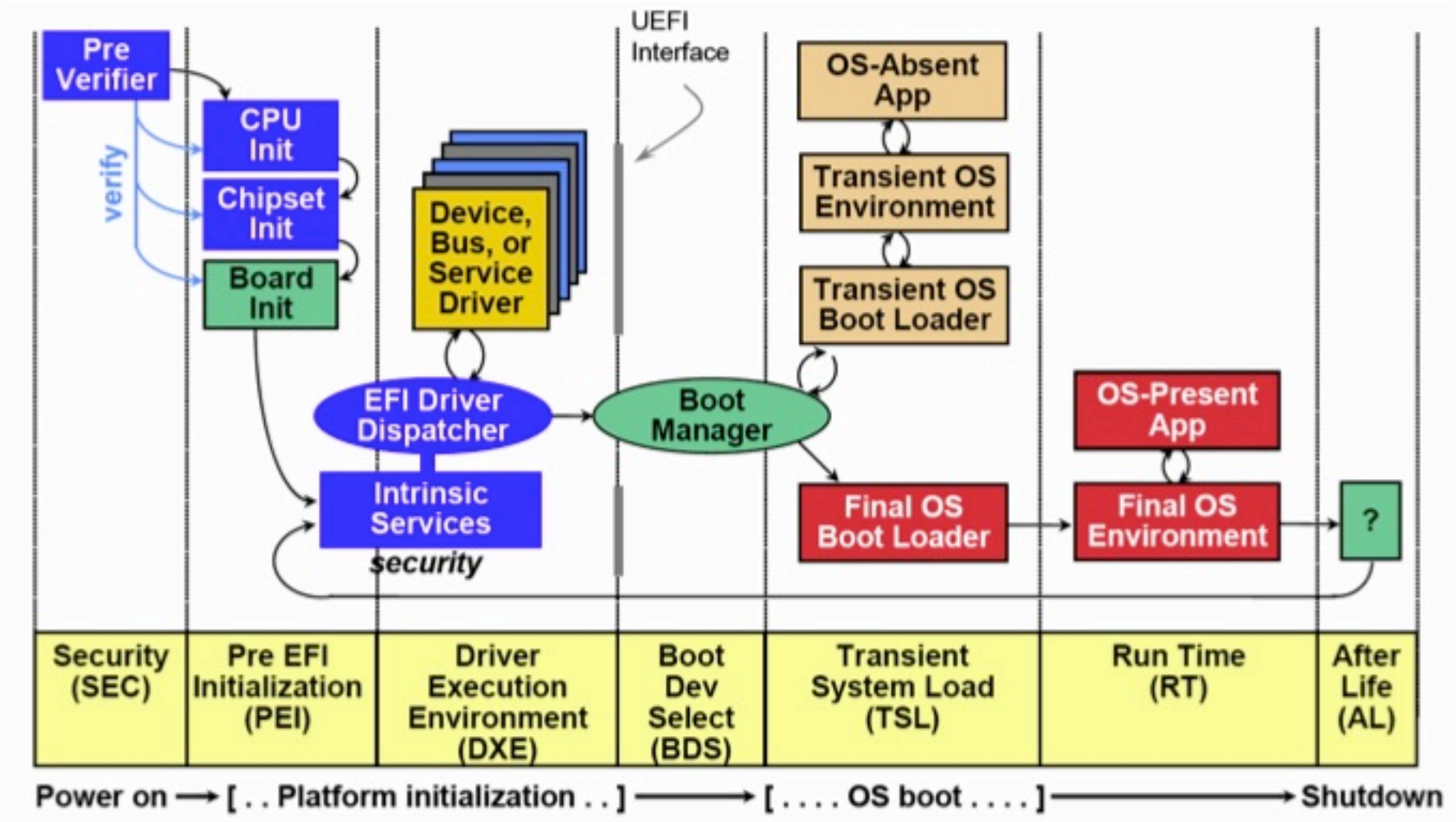
- Everything is labeled with a GUID.
- No filenames.
- Many GUID can be found in EFI specs.
- Others are vendor specific/private.

EFI Boot Flow

EFI Boot Phases

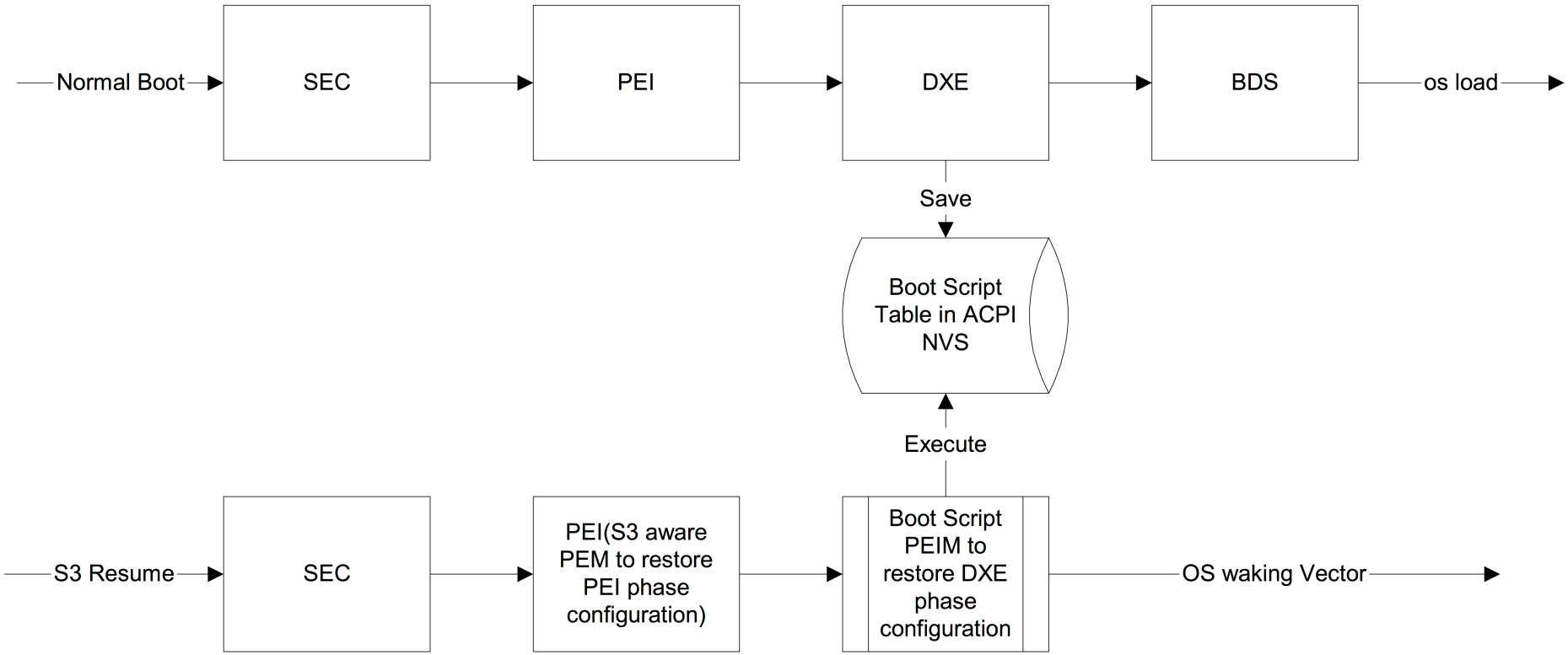
- Different initialization phases.
- Make resources available to next phase.
- Memory for example.

PI Boot Phases



EFI Phases

- Security (SEC).
- Pre-EFI Initialization (PEI).
- Driver Execution Environment (DXE).
- Boot Device Selection (BDS).
- Others...


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.

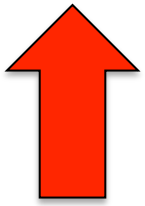
UEFITool 0.20.6 – bios_dump.bin

Structure

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

Information

```
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

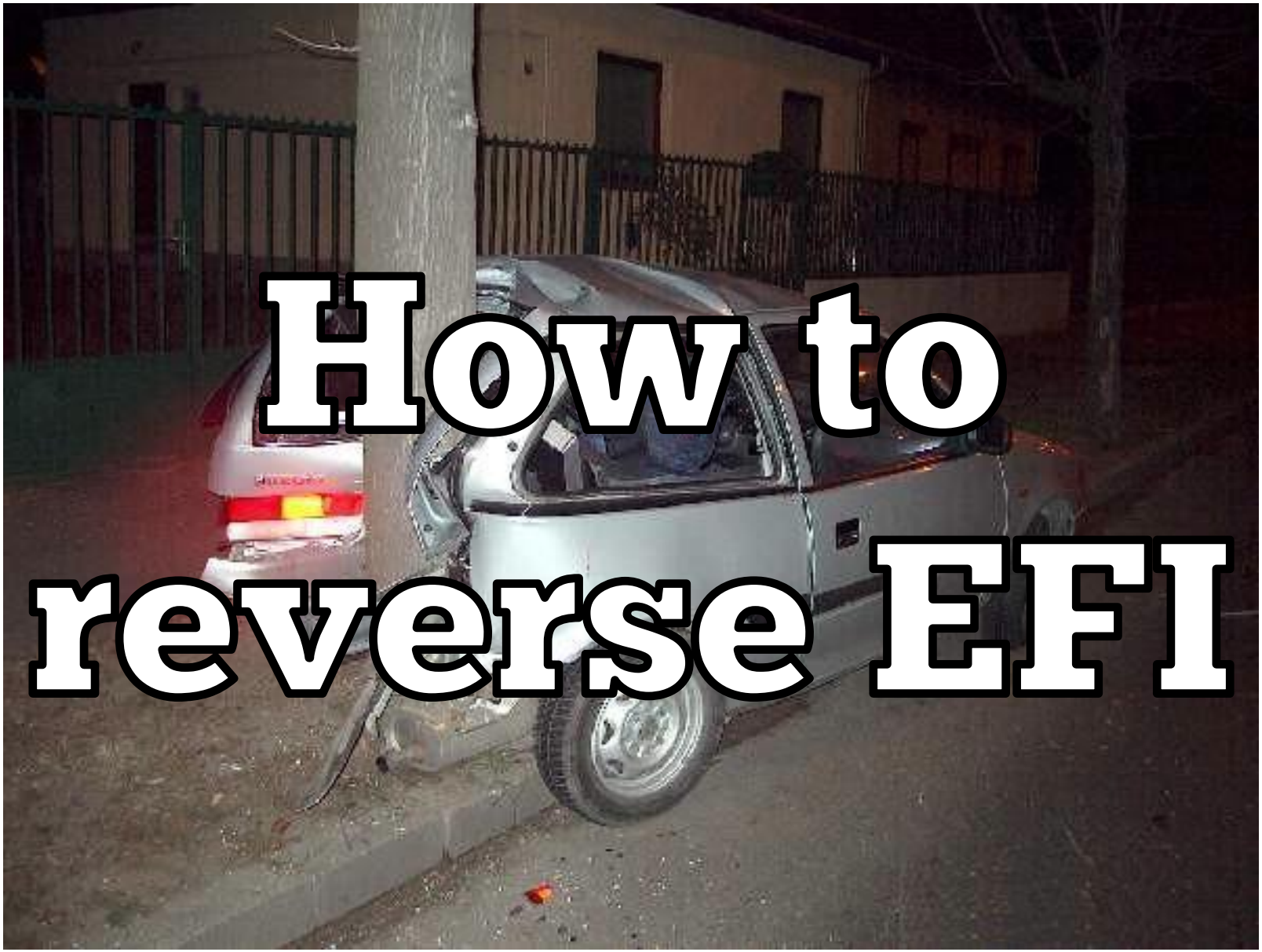
Name	Action	Type	Subtype
FC1BCDB0-7D31-49AA-936A-A4600D9DD083		Section	GUID defined
PE32 image section		Section	PE32 image
A210F973-229D-4F4D-AA37-9895E6C9EABA		File	DXE driver
FC1BCDB0-7D31-49AA-936A-A4600D9DD083		Section	GUID defined
PE32 image section		Section	PE32 image
025BBFC7-E6A9-4B8B-82AD-6815A1AEAF4A		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
```

gFrameworkEfiMpServiceProtocolGuid

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.
- Solution is to build your own TE loader.
- Easier than you think ☺.


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      FffsFindNextFile;      |
  EFI_PEI_FFS_FIND_SECTION_DATA   FffsFindSectionData;   |
  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

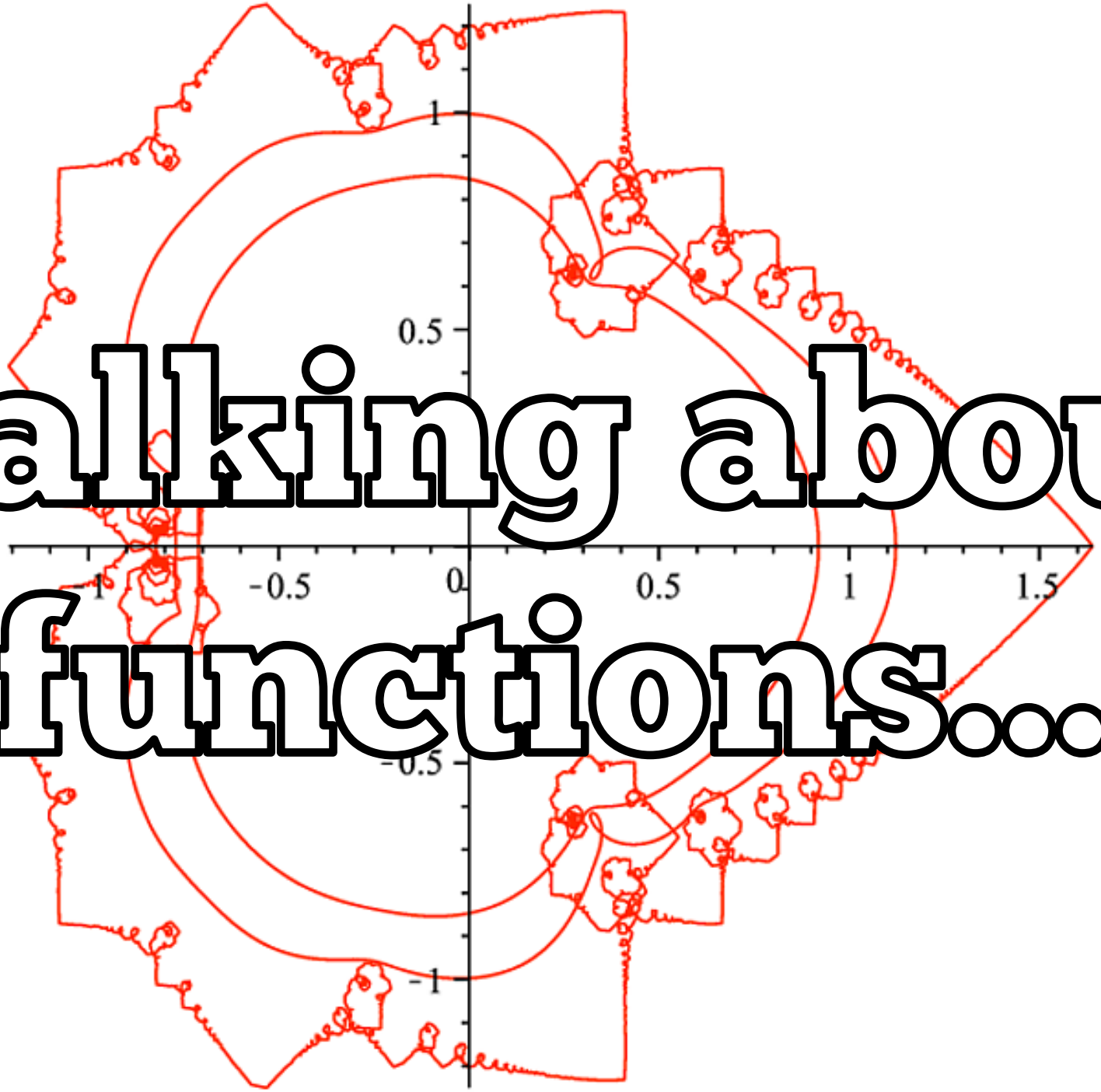
```
typedef struct {
  EFI_TABLE_HEADER Hdr;
  CHAR16 *FirmwareVendor;
  UINT32 FirmwareRevision;

  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.

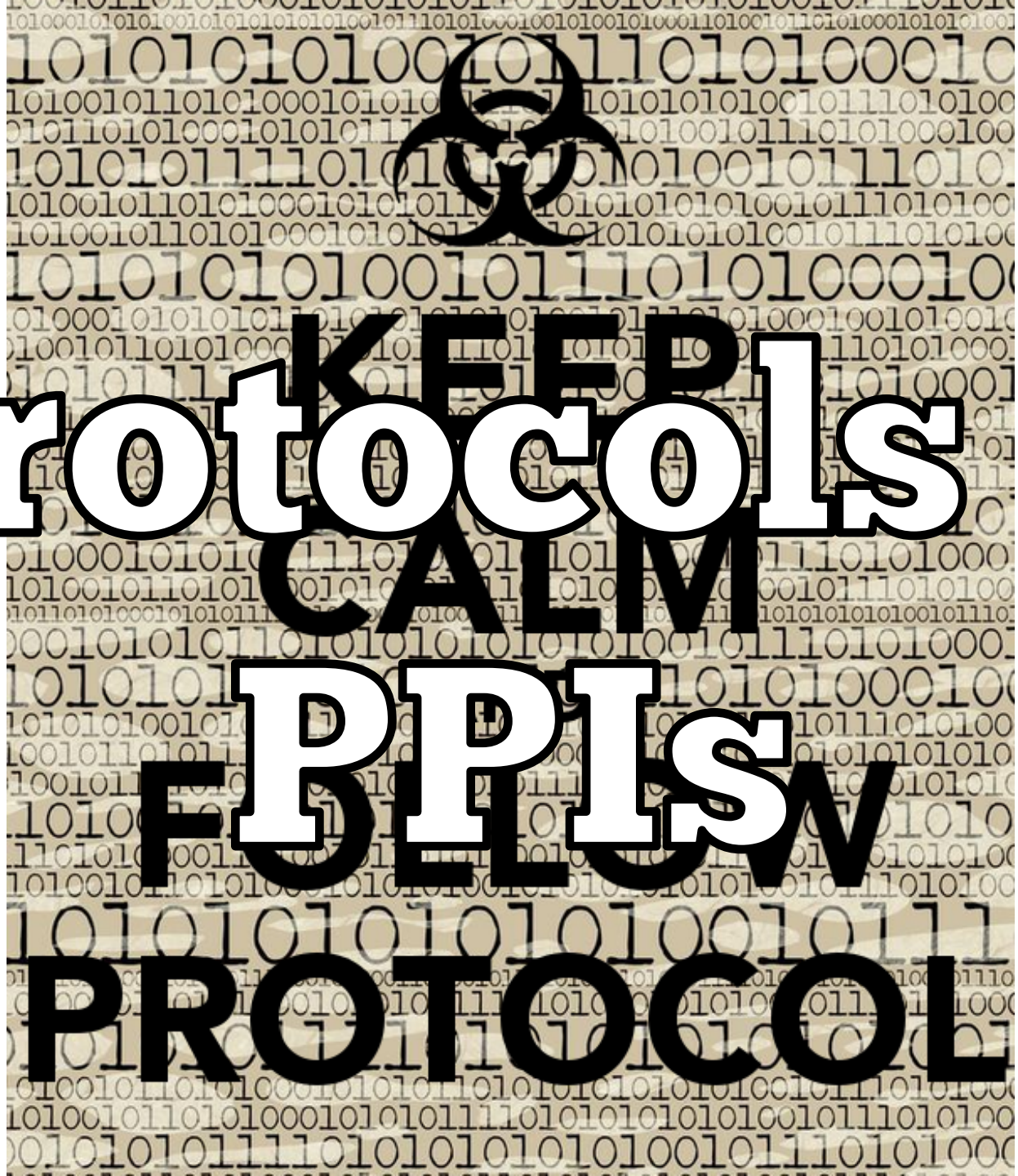
```
call    PeiPerfMeasure  ;     PEI_PERF_START (&PrivateData.PS,L"PreMem", NULL, mTick);
lea     eax, [ebp+var_C8]
mov     [esp+8], eax
lea     eax, [ebp-268h]
mov     [esp+4], eax
mov     [esp], edi
call    PeiDispatcher   ;   PeiDispatcher (PeiStartupDescriptor, &PrivateData, DispatchData);
cmp     [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 & FBPIs



Protocols & PPIs

- The basic services aren't enough.
- How are more services made available?
- Via Protocols and PPIs.
- Installed (published) by 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
```

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 available (yet).

How to find EFI monsters

- Only useful for machines with available updates.
- Newly released machines need to wait for a firmware update.
- Firmware & signatures vault
  - https://github.com/gdbinit/firmware_vault

How to find EFI monsters

- Two file formats used for updates.
- SCAP (most common).
- FD (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-AC59-8B7F70E41A6B

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	
1 ▼ Compressed section		Section	Compressed	
▼ 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	
2 ▼ Compressed section		Section	Compressed	
▼ 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	
3 ▼ Compressed section		Section	Compressed	
▼ 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

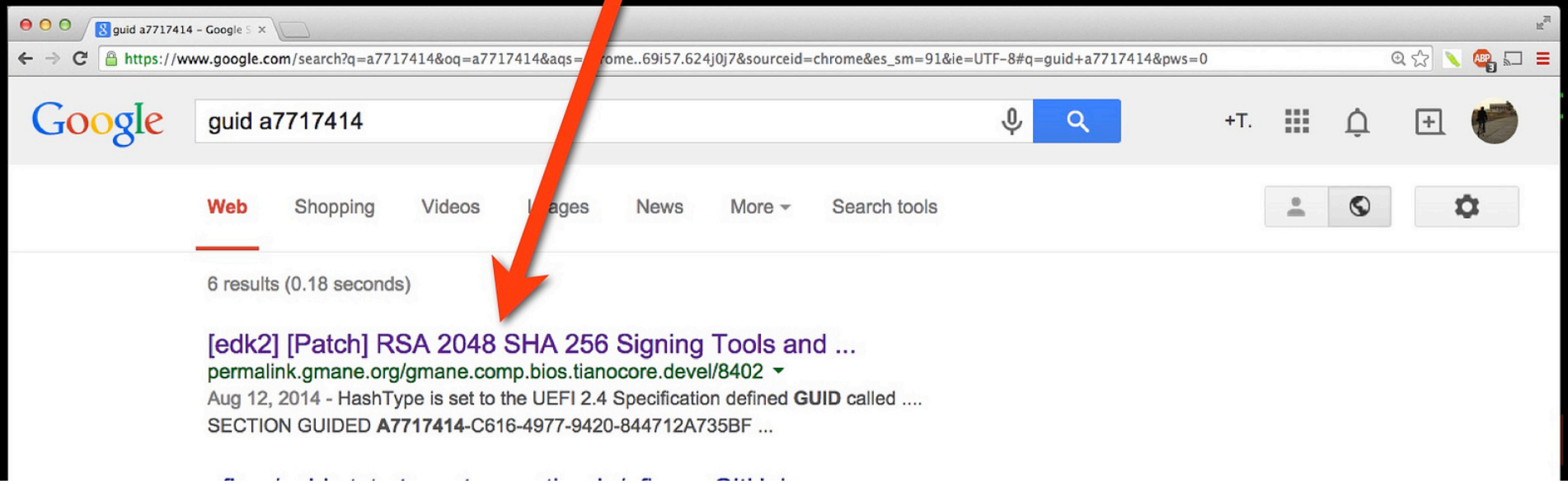
- ❶ is NVRAM region.
- ❷ is Microcode.
- ❸ is Boot volume.

SCAP

- SCAP is signed.
- RSA2048 SHA256.
- Apple backported from UEFI to EFI.
- 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 15 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 ef 59 9f be da ef 4d 6b 0c  ..u.u...Y....Mk.
```

GUID a7717414-c616-4977-94208447 12a735bf



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?

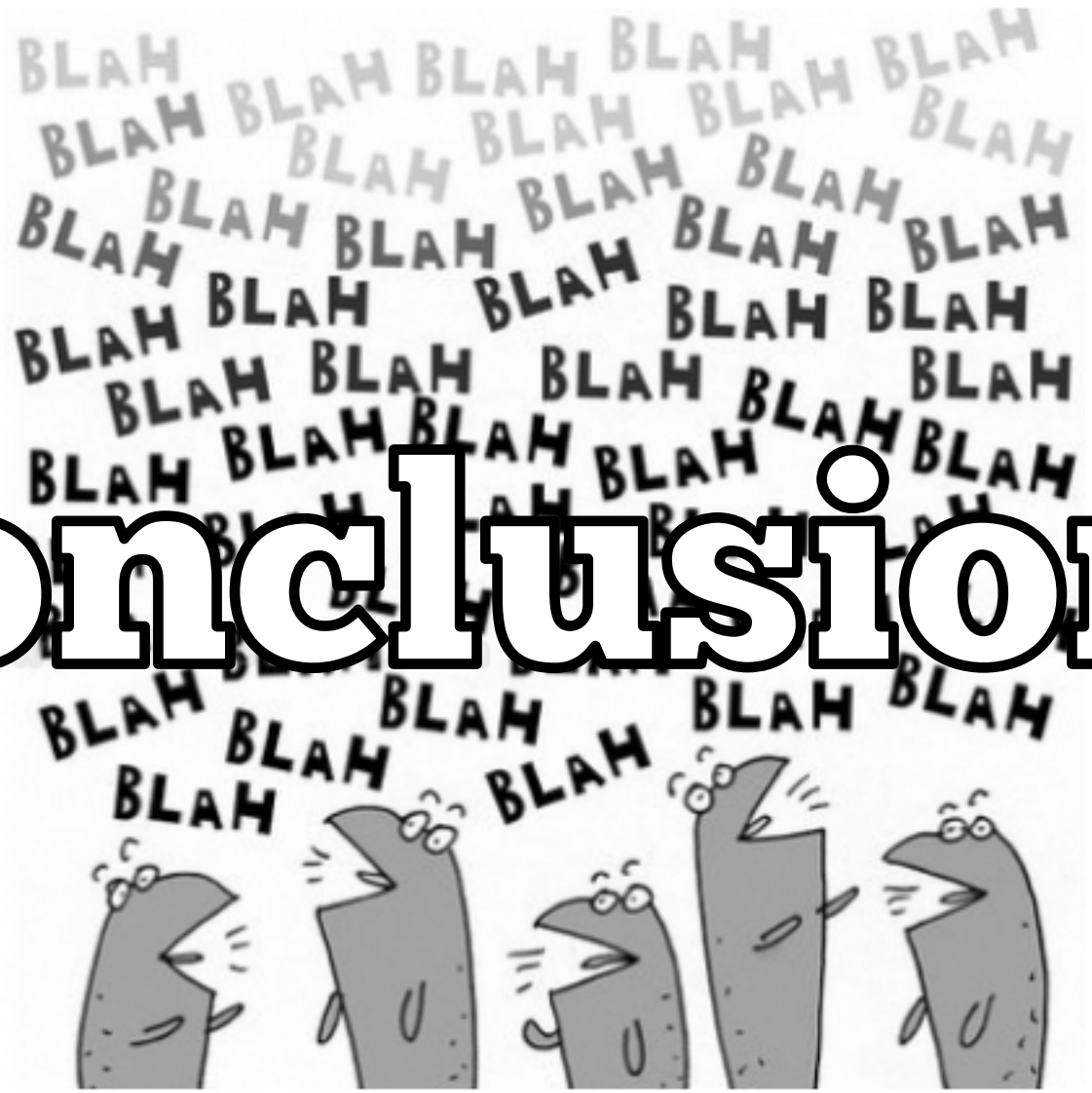
How to find EFI monsters

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

- EFI rootkits aren't unicorns.
- Although they are very rare.
- And we really don't know what's out there.
- HackingTeam developed one last year.
- Although it was too simple and not advanced.

Conclusions

- Chasing them requires hardware.
- Disassembling Macs monthly is not scalable.
- How to deal with this at enterprise level?

Conclusions

- Vendors are slow to release updates.
- If they ever release them.
- Check legbacore.com work.
- Apple has a great opportunity here.

Conclusions

- SMC is another interesting chip.
- Alex Ionescu and Andrea Barisani did some work in this area.
- There's SMC firmware update in a EFI driver.

Conclusions

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

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)
(鋰聚合物電池組/锂聚合物电池组) Rechargeable Li-polymer Battery Pack
(電壓/电压) Rating: 11.25V ⎓ (容量/容量) 3990mAh,45Wh
CAUTION: Risk of explosion if battery is replaced by an incorrect type. Dispose of used batteries according to the instructions. Risk of fire and burns. Do not open, crush, heat above (manufacturer's specified maximum temperature) or incinerate. Follow manufacturer's instructions. Charging current 1.7A / voltage 13.05V.
Max. operation temperature is 40°C.
ACHTUNG: Bei Verwendung anderer. Batterien besteht, Feuer oder Explosionsgefahr, Siehe die Vorsichtsmaßregeln in der Bedienungsanleitung. Wenn Sie Fragen oder Kommentare bezüglich der Akkubatterie haben, wenden Sie sich bitte an den Computerhersteller.
ATTENTION! A remplacer que par une autre batterie de meme type ou de meme qualite recommandée par le constructeur. Mettre au rebut les batteries usagées conformément aux instructions du fabricant.
危険: バッテリパックを分解、改造、火中に投入、ショート、あるいは指定された充電方法以外では充電しないでください。守らないと、火災、破裂、発熱の原因となります。
注意事項: 請參閱說明書的安全指示使用電池。如有問題請與電腦供應商聯絡。使用其他電池替換，將可能引起安全問題。
注意事项: 请参阅说明书的安全指示使用电池。如有问题请与电脑供应商联络。使用其他电池替换，将可能引起安全问题。
TIS 22[illegible]-254[illegible]
Acer Computer Co., Ltd
CE
MADE IN CHINA
EU 3920mAh
Acer Italy s.r./Via Lepetit, 40, 20020 Lainate (MI) Italy
RECOGNIZED COMPONENT
Intertek
CONFORMS TO ANSI/UL STD. 60950-1
CERTIFIED TO CAN/CSA STD. C22.2 NO. 60950-1
使用後は
リサイクルへ
Li-ion00
PSE
日本エイサー株式会社
11.25V 3920mAh
YU12001-13016
A/S: (02)3775-1516

Conclusions

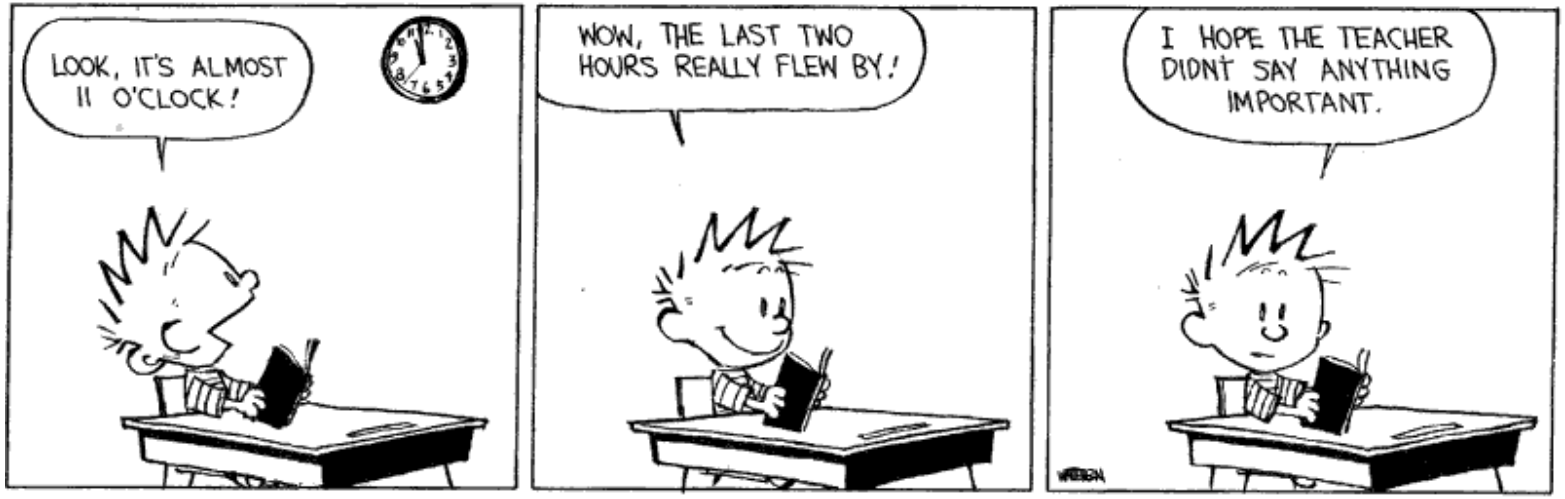
- Acer C720 & C720P Chromebook.
  - https://www.chromium.org/chromium-os/developer-information-for-chrome-os-devices/acer-c720-chromebook
- #7 is a write-protect screw.

Conclusions

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

Greetings

- SECUINSIDE team, Snare, Trammell, Xeno, Corey, Saure.



https://reverse.put.as

https://github.com/gdbinit

reverser@put.as

@osxreverser

#osxre @ irc.freenode.net

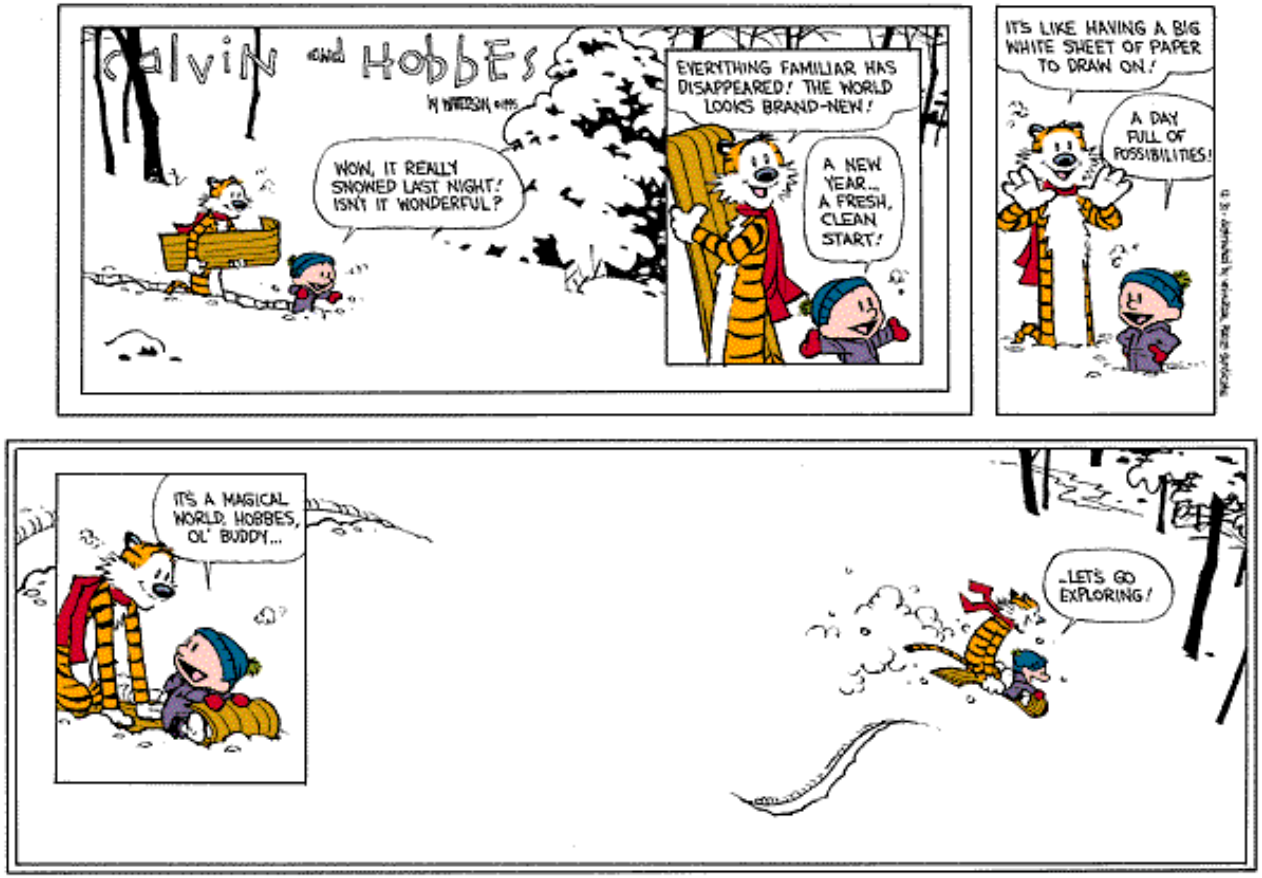
PGP key

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!



Let's go exploring!

References

- Images from images.google.com. Credit due to all their authors.
- Thunderstrike presentation
  - https://trmm.net/Thunderstrike_31c3
- 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