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 at LegbaCore
- 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
```
Root exploit
Remote code can escalate to 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

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 028cc: 0002d0 / 001204

Write to Option ROM
Search PCIe bus for removable devices
Thunderbolt adapter is now infected
Option ROM contains Thunderstrike 2
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: preferences_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 0x0001CB8 bytes copied to 7AFD7600
00: 663CEC8353565755
08: F008FED1F80405C7
10: 01CEE87AFD75D0A1
18: 00001C92C3810000

***** entry point 0x7AFD74FC=0000FFE9

***** Keystrokes: '\x0000\x00DD1p'
Starting OS... 10 0 F

---

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
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
YOU ARE NOT A BEAUTIFUL AND UNIQUE SNOWFLAKE.
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)
Case study 1: Speed Racer
### BIOS_CNTL (LPC I/F—D31:F0)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:2</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>
| 1   | **BIOS Lock Enable (BLE).** Once set, this bit can only be cleared by a PCIRST#.  
   
   1 = Setting the BIOSWE bit will cause SMI
   
   0 = Setting the BIOSWE will not cause SMI. |
| 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. |

Offset Address: 4E–4Fh  
Default Value: 0000h  
Lockable: No  
Attribute: R/W  
Size: 16 bits  
Power Well: Core
Case study 1: Speed Racer

VU #766164

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

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
### 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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:6</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
| 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

- BIOS_CNTL.BLE bit
- BIOS_CNTL.SMM_BWP bit
- Protected Range Registers

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

Flash Addr.
- 18E000
- 190000
- 610000
- 632000
- 7FFFFFF

UEFITool says "padding"?
ACCESS CONTROLLED BY SMM
ACCESS DENIED EXCEPT TO SMM
ACCESS DENIED EVEN TO SMM
ACCESS DENIED EVEN TO SMM
OS-resident Attacker
ACCESS CONTROLLED BY SMM
ACCESS DENIED EXCEPT TO SMM
ACCESS DENIED EVEN TO SMM
ACCESS DENIED EVEN TO SMM

Recommended:
BIOS_CNTL=0x1A
### Case study 1: Speed Racer

#### Vendor Information

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Status</th>
<th>Date Notified</th>
<th>Date Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>American Megatrends Incorporated (AMI)</td>
<td>Affected</td>
<td>12 Sep 2014</td>
<td>29 Dec 2014</td>
</tr>
<tr>
<td>Lenovo</td>
<td>Affected</td>
<td>12 Sep 2014</td>
<td>23 Jul 2015</td>
</tr>
<tr>
<td>Phoenix Technologies Ltd.</td>
<td>Affected</td>
<td>12 Sep 2014</td>
<td>17 Dec 2014</td>
</tr>
<tr>
<td>Apple Inc.</td>
<td>Not Affected</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBM Corporation</td>
<td>Not Affected</td>
<td>12 Sep 2014</td>
<td>16 Dec 2014</td>
</tr>
<tr>
<td>Insysde Software Corporation</td>
<td>Not Affected</td>
<td>12 Sep 2014</td>
<td>03 Feb 2015</td>
</tr>
<tr>
<td>Intel Corporation</td>
<td>Not Affected</td>
<td>12 Sep 2014</td>
<td>06 Jan 2015</td>
</tr>
<tr>
<td>AsusTek Computer Inc.</td>
<td>Unknown</td>
<td>12 Sep 2014</td>
<td></td>
</tr>
<tr>
<td>Gateway</td>
<td>Unknown</td>
<td>12 Sep 2014</td>
<td></td>
</tr>
<tr>
<td>Hewlett-Packard Company</td>
<td>Unknown</td>
<td>12 Sep 2014</td>
<td></td>
</tr>
<tr>
<td>Sony Corporation</td>
<td>Unknown</td>
<td>12 Sep 2014</td>
<td></td>
</tr>
<tr>
<td>Toshiba</td>
<td>Unknown</td>
<td>12 Sep 2014</td>
<td></td>
</tr>
</tbody>
</table>

*If you don’t hold your vendor accountable: silence*

*No penalty for being wrong...*
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.
• 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!

Protected Range Registers

Firmware

UEFITool says "padding"?

Code & Stuff

EFI Variables

Code & Stuff

OS-resident Attacker

ACCESS DENIED EVEN TO SMM

Flash Addr.

18E000
190000
610000
632000
7FFFFFF
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 defeated 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

Normal Boot:
SEC → PEI → DXE → BDS → os load

S3 Resume:
SEC → PEI/S3 aware PEM to restore PEI phase configuration
→ Boot Script Table in ACPI NVS
→ Execute
→ 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_SCRIPTSTALL_OPCODE       0x07
#define EFI_BOOT_SCRIPT_DISPATCH_OPCODE   0x08
```

Version 0.91
April 1, 2004
Case study 2: Darth Venamis

<table>
<thead>
<tr>
<th>BIOS_CNTL.BLE bit</th>
<th>BIOS_CNTL. SMM_BWP bit</th>
<th>Protected Range Registers</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>UEFITool says &quot;padding&quot;?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>18E000</td>
</tr>
</tbody>
</table>

OS-resident Attacker

ACCESS DENIED EVEN TO SMM
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.

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Status</th>
<th>Date Notified</th>
<th>Date Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>American Megatrends Incorporated (AMI)</td>
<td>Affected</td>
<td>15 Sep 2014</td>
<td>10 Dec 2014</td>
</tr>
<tr>
<td>Dell Computer Corporation, Inc.</td>
<td>Affected</td>
<td>15 Sep 2014</td>
<td>22 Jan 2015</td>
</tr>
<tr>
<td>Insyde Software Corporation</td>
<td>Affected</td>
<td>-</td>
<td>03 Feb 2015</td>
</tr>
<tr>
<td>Intel Corporation</td>
<td>Affected</td>
<td>15 Sep 2014</td>
<td>29 Dec 2014</td>
</tr>
<tr>
<td>Lenovo</td>
<td>Affected</td>
<td>-</td>
<td>21 Jan 2015</td>
</tr>
</tbody>
</table>
Case study 2: Darth Venamis

- 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.

Physical access is no longer required!
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.

Afer 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
@k6em10

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

RETWEETS 3
FAVORITES 9

12:14 PM - 3 Jun 2015
"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
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. This 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
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

(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

- 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 OS X 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!
Case study 4: Option ROMs

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

2. 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

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

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!