Now You See It...

TOCTOU Attacks Against BootGuard

Peter Bosch & Trammell Hudson
and once that is done we have owned the system

Boot strapping slightly more secure systems

Trammell Hudson @grr

A6C7 4E34 1054 A169 CE52
BE5F B65B FE54 DDEF 86C0
Hardware & Firmware Threats
Platform Threats

- BIOS Malware
- UEFI Rootkits
- Bootkits
- SMM Rootkits
- Device FW Malware
- ACPI Rootkits
- Option ROM Malware
- Evil Maid
- HVM Rootkits (Blue Pill)
- HW Trojans

http://vzimmer.blogspot.com/2013/09/where-do-i-sign-up.html
Full Verified Boot Sequence

Intel® Device Protection Technology with Boot Guard


OEM PI Verification Using PI Signed Firmware Volumes
Vol 3, section 3.2.1.1 of PI 1.3 Specification

OEM UEFI 2.4 Secure Boot
Chapter 27.2 of The UEFI 2.4 Specification

http://vzimmer.blogspot.com/2013/09/where-do-i-sign-up.html
Chain of Trust (simplified)

- ME boot ROM
- Root of Trust
- ME region
- SPI FLASH
- X86 Microcode
- BIOS ACM
- SPI FLASH
- Reset & PEI
- SPI FLASH
- DXE regions
- SPI FLASH
- Bootloader
- On Disk
- OS kernel
- On Disk

Signed by Intel

Fused OEM key

Signed by IBV or OEM
Coreboot slide
Intel should be congratulated for taking steps to make it more difficult for attackers to compromise system firmware, but criticised for doing so in such a way that vendors are forced to choose between security and freedom. The ability to control the software that your system runs is fundamental to Free Software, ...
Don’t attack the algorithm, Attack the implementation
Safeguarding rootkits: Intel BootGuard

The issue

One day I found out that some systems have the SPI flash regions unlocked and the BootGuard configuration not set (nor enabled, nor disabled):

- All Gigabyte systems
- All MSI systems
- 21 Lenovo branded notebook machine types and 4 ThinkServer machine types
- other few vendors I cannot mention at the moment

That’s because of the close manufacturing fuse was not set at the end of the manufacturing line.
... if an attacker manages to delete the BootGuardDxe from the DXE volume, the protection of the DXE part will not work at all (there will be no code to check the results of the verification done by the IBB).

https://embedi.org/blog/bypassing-intel-boot-guard/
Slides "Modern Secure Boot Attacks: Bypassing Hardware Root of Trust from Software" from #BHASIA and #OPCDE2019 released! Lenovo keeps manufacturing mode Boot Guard "backdoor" to unlock DXE volume for arbitrary modifications. It fully breaks Secure Boot!

github.com/REhints/Public  ...
“Chain of Trust” is only as secure as every link in the chain.
Our Achievements

- Switched-on AMT on non-vPro systems
- Activated JTAG for Intel ME via the vulnerability
- Dumped starter code (aka ROM)
- Recovered complete Huffman code for ME 11
- Extracted Integrity and Confidentiality Platform Keys [FFS17]
- Bypassed Intel Boot Guard
### CVE-2018-9062

**Description**

In some Lenovo ThinkPad products, one BIOS region is not properly included in the checks, allowing injection of arbitrary code.

**References**

- URL: http://www.securityfocus.com/bid/105387

### CVE-2018-12169

**Description**

Platform sample code firmware in 4th Generation Intel Core Processor, 5th Generation Intel Core Processor, 6th Generation Intel Core Processor, 7th Generation Intel Core Processor and 8th Generation Intel Core Processor contains a logic error which may allow physical attacker to potentially bypass firmware authentication.

**References**

- URL: http://www.securityfocus.com/bid/105387
- CONFIRM: https://edk2-docs.gitbooks.io/security-advisory/content/unauthenticated-firmware-chain-of-trust-bypass.html

**Assigning CNA**

Intel Corporation
Unsigned firmware volumes!

Bootguard protected sections

Phoenix hash protected volumes

https://github.com/LongSoft/UEFITool
# C.3 OEM Profile Parameters

## Table C-3. Profile Parameters Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protect Bios Environment Enabled (PBE)</td>
<td>Platform manufacturer may want Initial boot block to be protected between verification/measurement and execution from attacks on buses and non-CPU components. Boot Guard accomplishes this by allowing the initial boot block to be verified and executed in LLC in NEM if PBE is enabled.</td>
<td>false - Take no actions to control the environment during execution of the BIOS components (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>true - Takes actions to control the environment during the execution of the BIOS components.</td>
</tr>
</tbody>
</table>
1. Start ACM
1. Start ACM
2. Verify IBB
What could go wrong?
Hashing and load into cache

Re-read code from flash
0x009ac0  SiInitPreMem-pe32  +4324 15
0x009a80  SiInitPreMem-pe32  +42e4 18
0x009ac0  SiInitPreMem-pe32  +4324 16
0x009a80  SiInitPreMem-pe32  +42e4 19
0x0f16cc0 SiInitPreMem-pe32  +51524 2
0x0f9e200 TraceHubStatusCodeHandlerPef-pe32  +2044 1
0x0ffcc40 SecCore-pe32  +1434 1
0x0ffcc80 SecCore-pe32  +1474 1
0x0fe0570 PeiCore-pe32  +454 0
0x0fe0340 PeiCore-pe32  +224 1
0x0fe057c PeiCore-pe32  +460 0
0x0fe0440 PeiCore-pe32  +324 1
0xfa9940 SystemErrorLogPef-pe32  +2a4 1
0xfa9900 SystemErrorLogPef-pe32  +264 1
SecCore::PeiTemporaryRamDone

FFFFCC42  mov    ecx,  IA32_MTRR_DEF_TYPE
FFFFCC47  rdmsr
FFFFCC49  and    eax, ~IA32_MTRR_ENABLE
FFFFCC5A  mov    ecx,  IA32_MTRR_DEF_TYPE
FFFFCC5F  wrmsr

- **E (MTRRs enabled) flag, bit 11** — MTRRs are enabled when set; all MTRRs are disabled when clear, and the UC memory type is applied to all of physical memory. When this flag is set, the FE flag can disable the fixed-range MTRRs; when the flag is clear, the FE flag has no effect. When the E flag is set, the type specified in the default memory type field is used for areas of memory not already mapped by either a fixed or variable MTRR.
Early Boot: ACM, Sec and PEI Phases

PEI
- Load RAM Init Module
- Init DRAM
- Load/Verify Modules

Sec/IBB
- Load PeiCore
- Disable CAR

ACM
- Verify ROM hash

Code in CAR   RAM   XIP
Using an FPGA to record the flash memory accesses during boot shows some interesting patterns of re-reading the same data from the UEFI BIOS region multiple times.
No soldering required
Series resistor
PCH !CS output
SPI flash
!CS input
PCH !CS output
Modchips of the state

Technical feasibility of the Bloomberg/Supermicro hardware implants

Trammell Hudson, Two Sigma
@qrs

https://trmm.net/Modchips
Response and Mitigations
| **Status:** | CONFIRMED |
| **Alias:** | None |
| **Product:** | TianoCore Security Issues |
| **Component:** | Security Issue |
| **Version:** | unspecified |
| **Hardware:** | All, All |
| **Importance:** | Normal |
| **Assignee:** | |
| **URL:** | |
| **Keywords:** | |
| **Personal Tags:** | |
| **Depends on:** | |
| **Blocks:** | |

**Reported:** 2019-03-12 01:28 EDT
**Modified:** 2019-05-07 07:38 EDT
**CC List:** 1 user including you
**Ignore Bug Mail:** (never email me about this bug)

**See Also:**
- EDK II Trunk
- UDK 2018
- UDK 2017
- UDK 2015
- UDK 2014 SP1

**Release(s) the issue is observed:**
- EDK II Trunk
- UDK 2018
- UDK 2017
- UDK 2015
- UDK 2014 SP1

**The OS the target platform is running:**
- ---

**Package:**
- IntelFsp2WrapperPkg
- IntelFspPkg
- IntelFspPkg
- IntelSiliconPkg
- MdeModulePkg

**Release(s) the issues must be fixed:**
- EDK II Trunk
- UDK 2018
- UDK 2017
- UDK 2015
- UDK 2014 SP1

**Intel CVE-2019-11098**
In addition to the call stack, the PEI Foundation will copy the following from temporary to permanent memory:

- PEI Foundation private data
- PEI Foundation heap
- HOB list
- Installed Firmware Volumes

Any permanent memory consumed in this fashion by the PEI Foundation will be described in a HOB, which the PEI Foundation will create.

The PEI Foundation will copy any installed firmware volumes from the temporary memory location to a permanent memory location with the alignment specified in the firmware volume header. Any *uncompressed* PE32 or TE sections within PEIMs in these firmware volumes will be fixed up. This ensures any static **EFI_PEI_PPI_DESCRIPTORs** or PPI interface pointers in these PEIMs point to the permanent memory addresses.

In addition, if there were any **EFI_PEI_PPI_DESCRIPTORs** created in the temporary memory heap or declared statically in PEIMs, their respective locations have been translated by an offset equal to the difference between the original location in temporary memory and the destination location in permanent memory. In addition to this heap copy, the PEI Foundation will traverse the PEI PPI database. Any references to **EFI_PEI_PPI_DESCRIPTORs** that are in temporary memory are translated to permanent memory addresses.
Why open source firmware is important

Jessie Frazelle - @jessfraz

https://blog.jessfraz.com/post/why-open-source-firmware-is-important-for-security/

https://coreboot.org/
https://www.linuxboot.org/
http://osresearch.net/
SPISpy coming soon!

Peter Bosch
@peterbjornx

Trammell Hudson
@qrs