



Building A UEFI Security Test Strategy

Presented by Kevin Davis
VP, Insyde Software

presented by



Agenda



Introduction

Why Security Testing?

Areas to test

Available Test Sources

Next Steps

Q & A



Introduction



UEFI is everywhere

- Tablets, PCs, Servers, Industrial Controllers
- Booting in UEFI Mode
- UEFI Secure Boot since 2.3.1c and Windows 8.0
 - Trying to protect the layers before the OSs
- UEFI interfaces are exposed, no BIOS hiding in the CSM

Security Researchers

- UEFI provides well defined interfaces to the OSs
- All Modern Windows 8 systems have common UEFI specs
- Security Researchers have a good starting point at TianoCore.ORG

Why Security Testing?



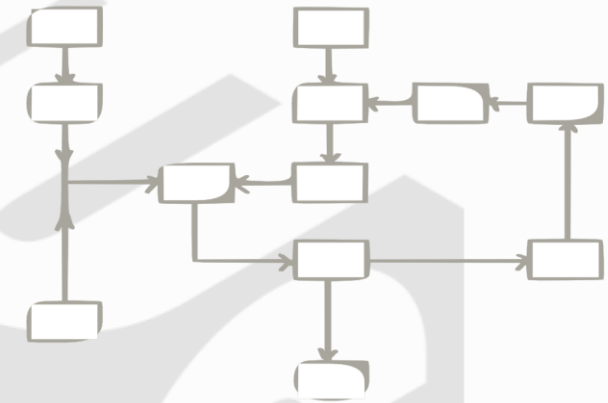
Systems are very complex

A single bit in a register can allow attacks

- If BIOS_Control bit 0 = 1, BIOS is writable
- If BIOS is writable, a hacker can insert code
- If the hacker is skilled, the code can do anything

This example can turn a system into a Brick!

OEM Warranty issue!



Areas to Test

Interfaces to the OS

Network interfaces

Interfaces to SMM / security co-processors

External port interfaces

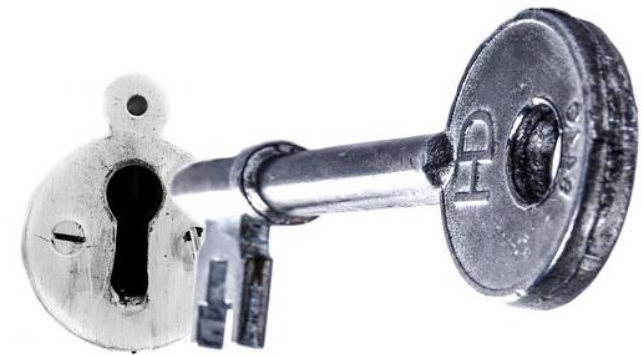
- Keyboards!, Mice!, USB ports, PCIe slots

ROM storage device registers

SMI Protection Control registers

Device update is critical

Some testing after machine is ready to ship



Available Test Sources



Silicon Vendors

- Open source tools
- NDA / Confidential tools

MITRE's Copernicus

BIOS Vendor tests

Hacking Tools



Intel Tools



Intel CHIPSEC tool – (github.com/chipsec/chipsec)

- Open source “engine”

- Open source test scripts

- NDA Test scripts

Intel SelfTest (NDA)

Intel BITS - (downloadcenter.intel.com/Detail_Desc.aspx?DwnldID=19763)

Most Intel tools identify Failures !!

CHIPSEC



A framework for analyzing security of PC platforms including hardware, system firmware and platform component configuration.

Allows creating security test suites, security assessment tools for low level components and interfaces.

```
[x][ =====  
[x][ Module: SMI Events Configuration  
[x][ =====  
[-] SMM BIOS region write protection has not been enabled (SMM_BWP is not used)  
[*] PMBASE (ACPI I/O Base) = 0x0400  
[*] SMI_EN (SMI Control and Enable) register [I/O port 0x430] = 0x00022033  
    [13] TCO_EN (TCO Enable) = 1  
    [00] GBL_SMI_EN (Global SMI Enable) = 1  
[+] All required SMI events are enabled  
[*] TCOBASE (TCO I/O Base) = 0x0460  
[*] TCO1_CNT (TCO1 Control) register [I/O port 0x468] = 0x0A00  
    [12] TCO_LOCK = 0  
[-] TCO SMI event configuration is not locked. TCO SMI events can be disabled  
[*] GEN_PMCON_1 (General PM Config 1) register [BDF 0:31:0 + 0xA0] = 0x0E18  
    [04] SMI_LOCK = 1  
[+] SMI events global configuration is locked  
[-] FAILED: Not all required SMI sources are enabled and locked!
```

**BIOS Region Write
Protection not enabled !!**

Intel's SelfTest



windump.st - SelfTest 6.3.4

File Tools Help

Intel Confidential

Register Details

Name: [5] SMM BIOS write protect disable (SMM_BWP)

Actual:

Expected:

Comment
This bit field is not checked.

Description
Attribute: R/W/O
Default: 1
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.

BIOS Write Protect is Disabled !!

Left sidebar tree view:
+ B8h GPIO_ROUT - GPIO Routing Control
+ D0h BIOS_SEL1 - BIOS Select 1 Register
+ D4h BIOS_SEL2 - BIOS Select 2 Register
+ D8h BIOS_DEC_EN1 - BIOS Decode Enable
- DCh BIOS_CNTL - BIOS Control Register
 [7:6] Reserved
 [5] SMM BIOS write protect disable (SMM_BWP)
 [4] Top Swap Status (TSS)
 [3:2] SPI Read Configuration (SRC)
 [1] BIOS Lock Enable (BLE)
 [0] BIOS Write Enable (BIOSWE)
+ E0h FDCAP - Feature Detection Capability
+ E2h FDLEN - Feature Detection Capability
+ E3h FDVER - Feature Detection Version
+ E4h FVECIDX - Feature Vector Index
+ E7h FVECD - Feature Vector Data
+ F0h RCBA - Root Complex Base Address

MITRE's Copernicus



" [The BIOS] is a great place to persist indefinitely without fear of detection" – MITRE Copernicus_July-2013.pptx

BIOS Access Control checking

- Run collection tool

- Run Python analysis tools

- Review results

Failures are not identified !!



MITRE's Copernicus Log – limited chipsets



```
MITRE Copernicus Loading

Thank you for using Copernicus!
If you'd like to help us build a master list of vulnerable BIOSes,
please email your .csv file to copernicus@mitre.org

Allocating memory for ICH Parameters object
Allocating memory for MCH Parameters object
Allocating memory for Flash Chip object
Allocating memory for SMBIOS Parameters object
Initializing ICH parameters
Email the following to copernicus@mitre.org so we can look into adding support for this architecture.
Copernicus Error: Unidentified IO Controller Hub vendor=8086, device=27bc
Memory Controller: vendor=8086, device=a010
Error initializing ICH parameters
Cleaning up allocated memory.
Freeing memory for Flash Chip
Freeing memory for MCH Parameters object
Freeing memory for ICH Parameters object
Freeing memory for SMBIOS Parameters object
Freeing memory for Host Data
One or more elements of Copernicus Failed. Return Status 0x00000001
MITRE Copernicus Unloading
```

Unknown Device !!

MITRE's Copernicus – Error?



```
E:\Copernicus>c:python protections.py per-version -id -o output_csv StrawberryMountain
COUNT BIOS_VENDOR PRODUCT_NAME BIOS_VERSION SMRAM_UNLOCKED BIOS_UNLOCKED
1
                                0
                                1

1 CSV files successfully processed
0 out of 1 (0.0%) unlocked SMRAM
1 out of 1 (100.0%) unlocked BIOS
0 out of 1 (0.0%) SMI_LOCK not set
1 out of 1 (100.0%) PR write protection used
0 out of 1 (0.0%) SMM_BWP used when supported
0 out of 1 (0.0%) vulnerable to CERT VU#255726
0 out of 1 (0.0%) vulnerable to CERT VU#912156
0 out of 1 (0.0%) vulnerable to CERT VU#552286
```

Failing BIOS - Unlocked !!
YES it is a failure !!

Some of Insyde's test tools



Private Interface Tester

Tests SMI input parameters – buffers

InsydeSecurityTestToolX64.efi

InsydeSecurityTestToolIA32.efi

Insyde Toolbox - H2OITB

Around 20 separate tests

Variables, CMOS, variable reclaim testing



Private Interface Tester



Insyde Security Test Tool V1.0

Copyright (c) 2014, Insyde Software Corporation. All Rights Reserved.

```
Function 54h test....[Pass]
Function 55h test....[Pass]
Function 56h test....[Pass]
Function 57h test....[Pass]
Function 58h test....[Pass]
Function 59h test....[Pass]
Function 5Ah test....[Pass]
Function 5Bh test....[Failed]
```

The range of input buffer memory can overlap SMRAM.

Insyde Toolbox examples



- **Variable reclamation check [count]**
If variable size is not full, continue to write variables to cause reclaim on next boot
- **Variable rebuild check [count]**
Erase all variables and FV header with fill data 0xFF or 0x00
- **Variable boundary check [count]**
Write variables and try to overflow the space
Write variables and try to exactly fill the space
- **Variable usage [count]**
Check available space to check if BIOS updated any variables
- **Dump Variables to a file**
- **Perform malicious change to variables**
- **Partially erase variables**



Hacker tools



UEFITool - UEFI firmware image viewer and editor

<https://github.com/LongSoft/UEFITool>

Universal-IFR-Extractor - extract IFR from UEFI modules

<https://github.com/donovan6000/Universal-IFR-Extractor>

HxD - Hex Editor and Disk Editor

<http://mh-nexus.de/en/hxd/>

Binwalk - Firmware Analysis Tool

<https://github.com/devttys0/binwalk>

Search for signatures inside binary files

<http://freecode.com/projects/signsrch>

firmware-mod-kit - tools for easy deconstruction and reconstruction of firmware images

<https://code.google.com/p/firmware-mod-kit/>



Next Steps



Decide how much testing is required

Test in multiple project phases

- Design phase = What will you test?
- Development phase = Is everything ready to lock down?
- Manufacturing phase = Is the system ready to ship?

Don't ship with back doors! Perform final locks!

Expect future attacks, prepare for field updates



Contact Kevin Davis or Alan Lo
for demos

Insyde Suites #902 & #904

Building A UEFI Security Test Strategy

Demo



More reading ...



- Intel CHIPSEC
<https://github.com/chipsec/chipsec>
- MITRE Copernicus tool
<http://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/copernicus-question-your-assumptions-about>
- Signed BIOS Attack
http://www.ekoparty.org/archive/2013/charlas/Kallenberg/DefeatingSignedBios-EkoParty_2013_v1.pptx
- BIOS Attack Summary
<https://www.defcon.org/images/defcon-22/dc-22-presentations/Bulygin-Bazhaniul-Furtak-Loucaides/DEFCON-22-Bulygin-Bazhaniul-Furtak-Loucaides-Summary-of-attacks-against-BIOS-UPDATED.pdf>
- NIST BIOS Protection Guidelines (SP 800-147 and SP 800-147B)
<http://csrc.nist.gov/publications/nistpubs/800-147/NIST-SP800-147-April2011.pdf>
http://csrc.nist.gov/publications/drafts/800-147b/draft-sp800-147b_july2012.pdf
- IAD BIOS Update Protection Profile
https://www.niap-ccevs.org/pp/pp_bios_v1.0.pdf
- Windows Hardware Certification Requirements
<http://download.microsoft.com/download/A/D/F/ADF5BEDE-C0FB-4CC0-A3E1-B38093F50BA1/windows8-hardware-cert-requirements-system.pdf>
- UEFI Forum sub-teams: USST (UEFI Security) and PSST (PI Security)

For more information on
the Unified EFI Forum and
UEFI Specifications, visit
<http://www.uefi.org>



presented by

