MODERN SECURE BOOT ATTACKS: BYPASSING HARDWARE ROOT OF TRUST FROM SOFTWARE

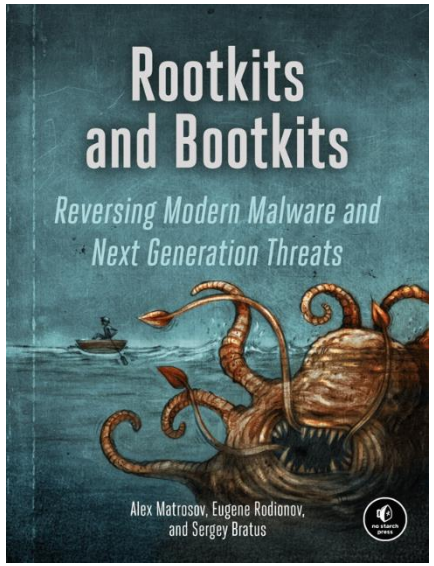
Alex Matrosov
@matrosov

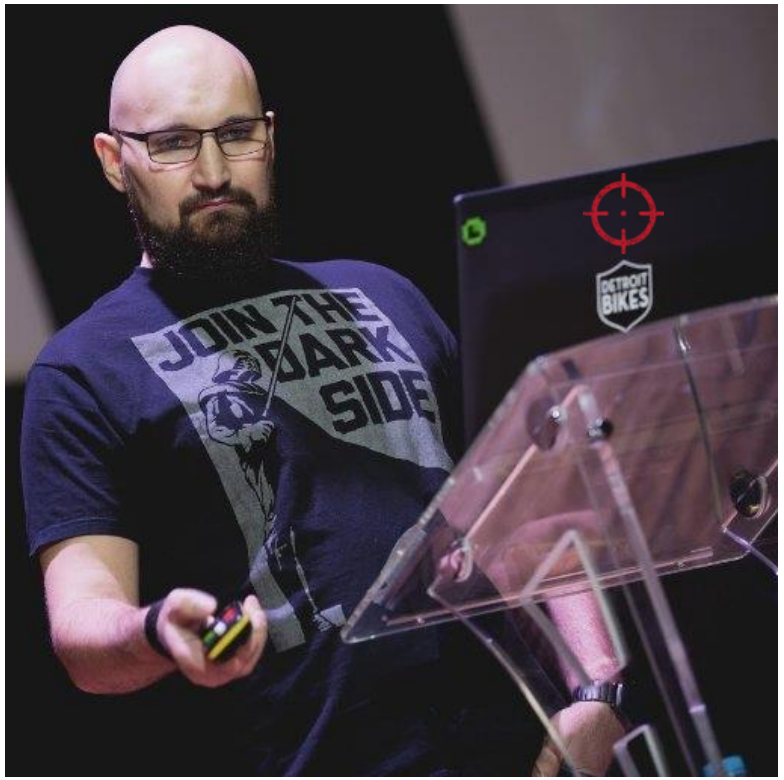
Leading Offensive Security REsearch at NVIDIA

Former Security Researcher @Cylance @Intel @ESET

Doing Security REsearch since 1997

Book co-author nostarch.com/rootkits





@matrosov

"Follow in the footsteps of professionals with a record of discovering advanced malware."
— Rodrigo Rubira Branco

Rootkits and Bootkits will teach you how to understand and counter sophisticated, advanced threats buried deep in a machine's boot process or UEFI firmware.

With the aid of numerous case studies and professional research from three of the world's leading security experts, you'll trace malware development over time from rootkits like TDL3 to present-day UEFI implants and examine how they infect a system, persist through reboot, and evade security software. As you inspect and dissect real malware, you'll learn:

- How Windows boots—including 32-bit, 64-bit, and UEFI mode—and where to find vulnerabilities
- The details of boot process security mechanisms like Secure Boot, including an overview of Virtual Secure Mode (VSM) and Device Guard
- Reverse engineering and forensic techniques for analyzing real malware, including bootkits like Rovnix/Carberp, Gapz, TDL4, and the infamous rootkits TDL3 and Festi
- How to perform static and dynamic analysis using emulation and tools like Bochs and IDA Pro
- How to better understand the delivery stage of threats against BIOS and UEFI firmware in order to create detection capabilities
- How to use virtualization tools like VMware Workstation to reverse engineer bootkits and the Intel Chipsec tool to dig into forensic analysis

Cybercrime syndicates and malicious actors will continue to write ever more persistent and covert attacks, but the game is not lost. Explore the cutting edge of malware analysis with *Rootkits and Bootkits*.

About the Authors

Alex Matrosov is an Offensive Security Research Lead at NVIDIA with over 20 years of experience in reverse engineering, advanced malware analysis, firmware security, and exploitation techniques. **Eugene Rodionov**, PhD, is a Security Researcher at Intel working in BIOS security for Client Platforms. **Sergey Bratus** is a Research Associate Professor in the Computer Science Department at Dartmouth College. He has previously worked at BBN Technologies on natural language processing research.

no starch press
THE FINEST IN GEEK ENTERTAINMENT™
www.nostarch.com

FSC FPO

"I LIE FLAT." This book uses a durable binding that won't snap shut.

Price: $49.95 ($65.95 CDN)
Shelve In: COMPUTERS/SECURITY
ISBN: 978-1-59327-716-1
54995
9 781593 277161

Rootkits and Bootkits
Reversing Modern Malware and Next Generation Threats
Matrosov, Rodionov, and Bratus
no starch press

Rootkits and Bootkits

Reversing Modern Malware and Next Generation Threats

Alex Matrosov, Eugene Rodionov, and Sergey Bratus

Foreword by Rodrigo Rubira Branco

no starch press

⌖Disclaimer⌖

I don’t speak for my employer.
All opinions, information here are mine responsibilities
😈 (include all bad jokes) 😈

REsearch Target

Lenovo P50

Gigabyte/ASUS/MSI don't care about security and it's too easy targets

- What is Hardware Root of Trust?

😈 Boot Guard Bypass 😈

- Computrace Never Dies
  - ✓ OS Enable/Disable
  - ✓ Permanent Disabling is a joke o_O
- SMI over WMI is too evil 😈
  - ✓ SMM communications without ring-0
  - ✓ WMI-based fileless FW rootkits?
- EC is not a security boundary 🤦

(*EC – Embedded Controller)

Hardware Root of Trust

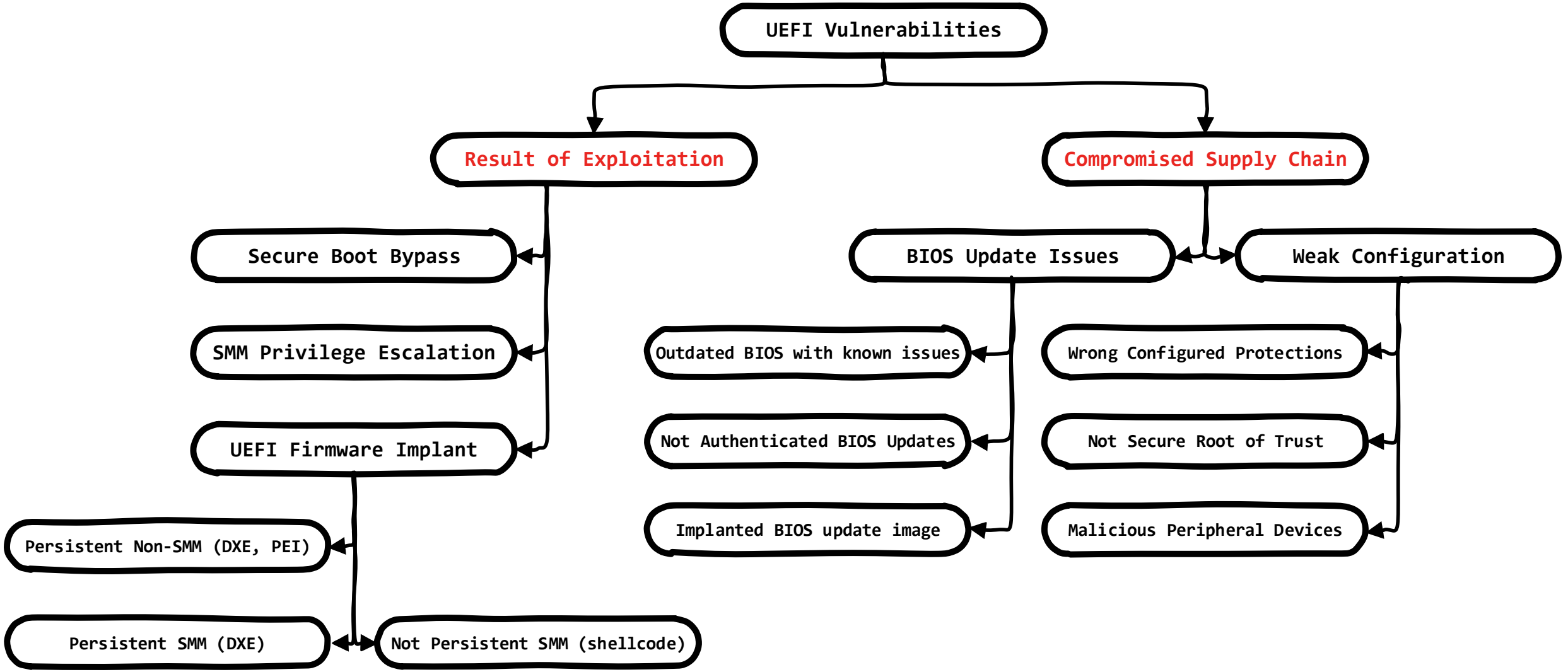
WTF Hardware Root of Trust?

- **Root of Trust baked in pure Hardware?**
  - ✓ Cant be extracted/modified from software (developed in RTL)?
    - ✓ not flexible with OEM’s
    - ✓ hard to support in the field (updates and etc.)
    - ✓ hard to implement secure way to cooperate with firmware on the same chip

- **In the most of the cases Hardware Root of Trust it’s a mix between firmware and locked in the FUSE value or by specific bit.**

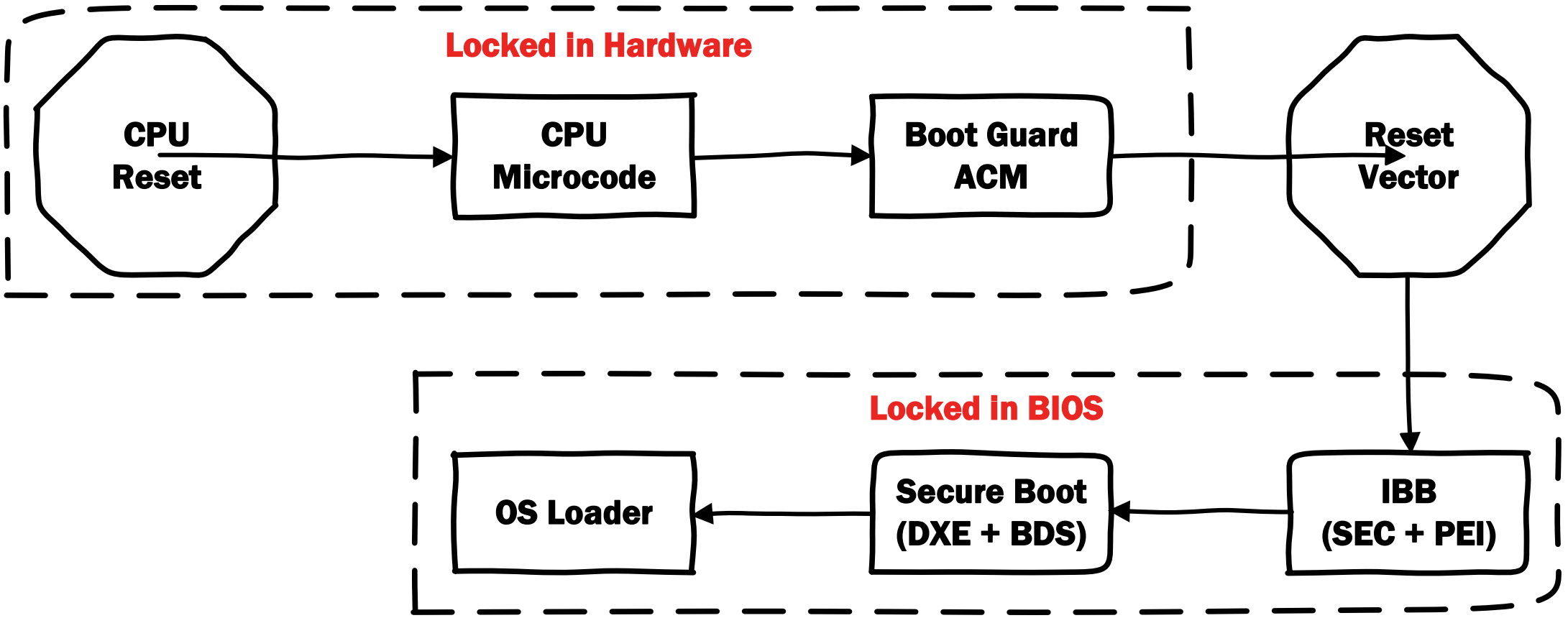
- **Secure state transition between hardware and firmware is hard. It’s always something missing.**

UEFI vulns classification



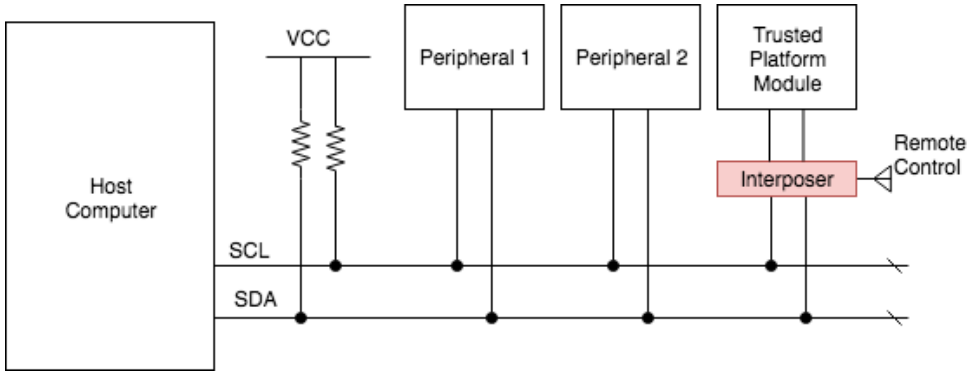
https://medium.com/@matrosov/uefi-vulnerabilities-classification-4897596e60af

Boot Guard: Boot Flow in Perfect World



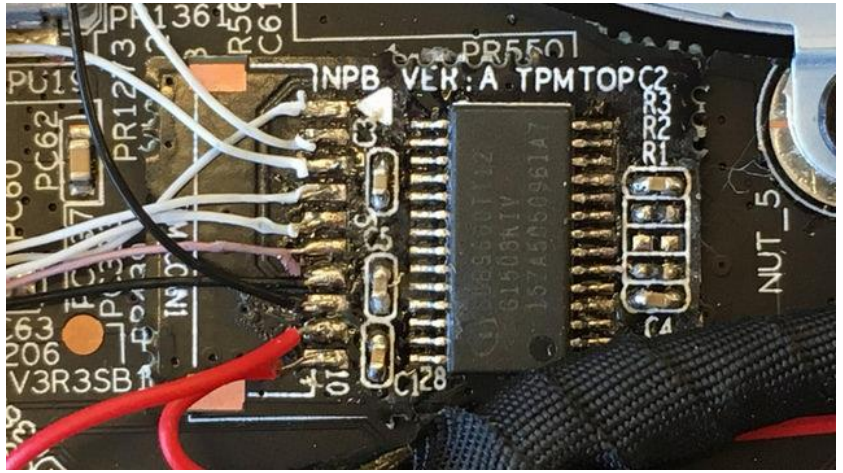
HW Root of Trust: TPM is broken?

@uffeux

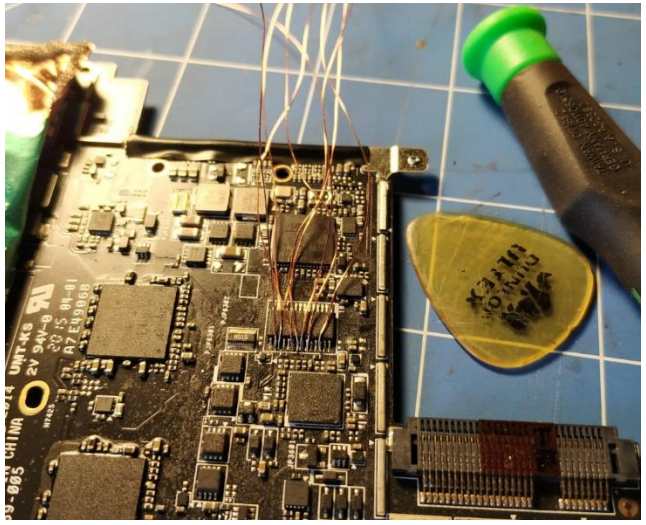


https://github.com/nccgroup/TPMGenie

@qrs

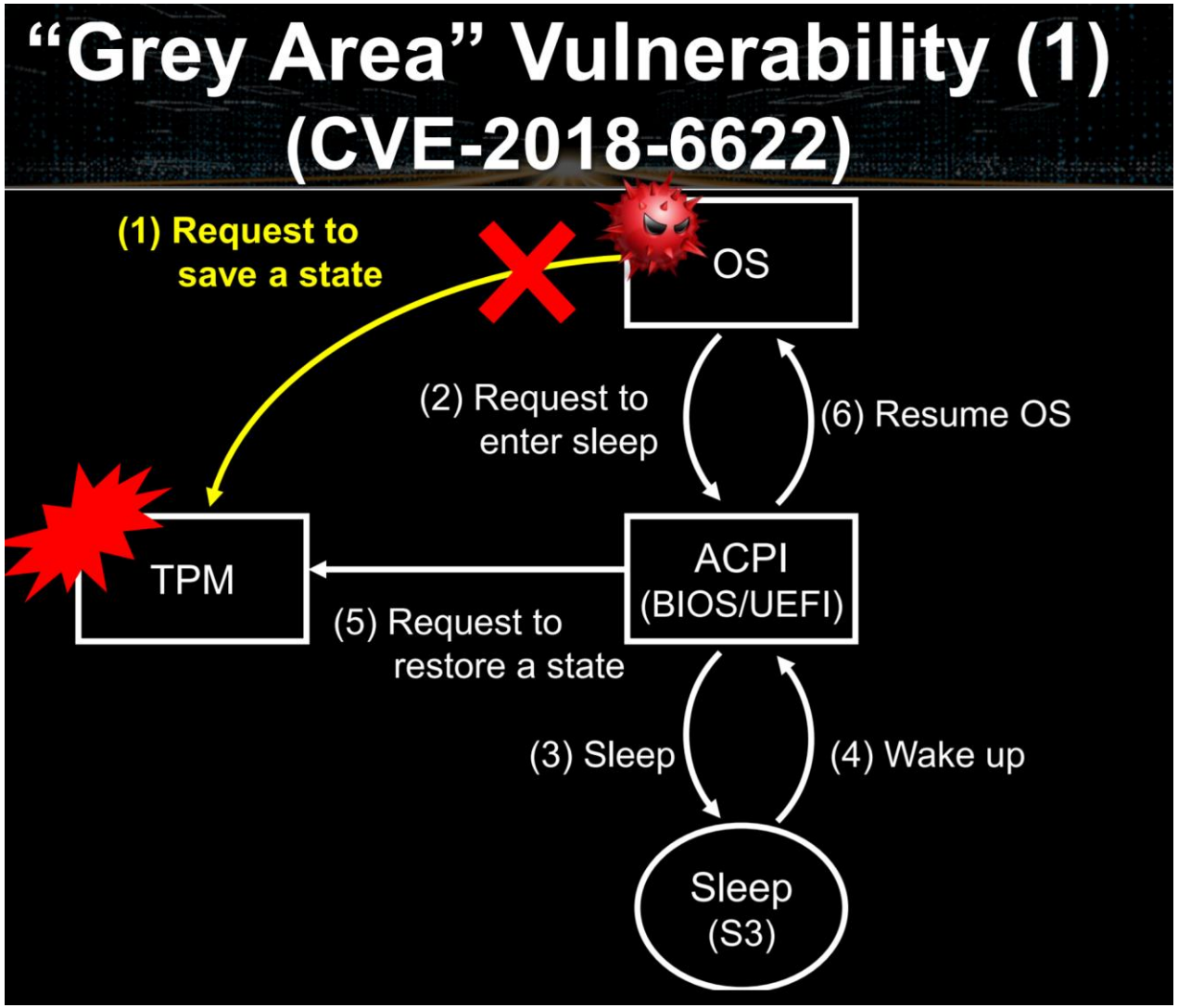


@0x446f49



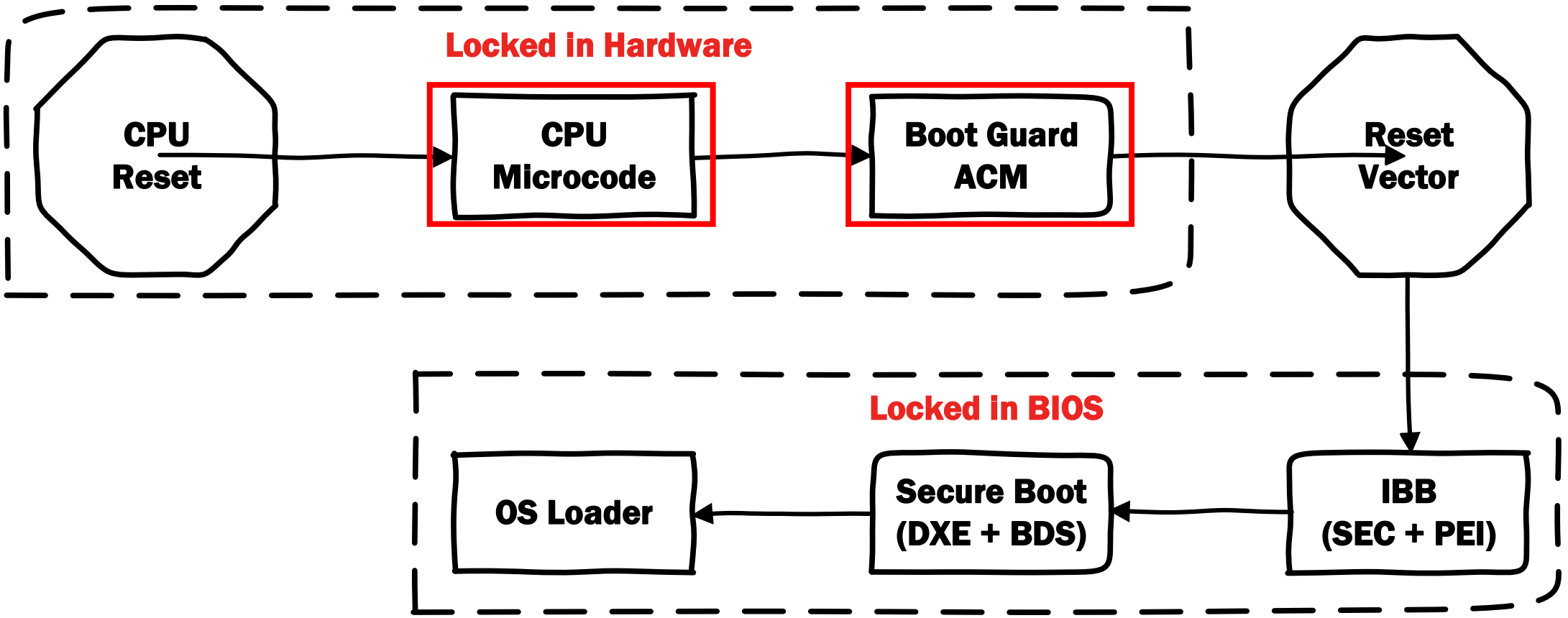
https://pulsesecurity.co.nz/articles/TPM-sniffing

HW Root of Trust: TPM is broken?



https://i.blackhat.com/asia-19/Thu-March-28/bh-asia-Seunghun-Finally-I-Can-Sleep-Tonight-Catching-Sleep-Mode-Vulnerabilities-of-the-TPM-with-the-Napper.pdf

Boot Guard: Boot Flow in REAL World



But world is not perfect :)

Microcode	File	Raw
Intel microcode	Microcode	Intel
Volume free space	Free space	
Padding	Padding	Non-empty

Parser | FIT | Security | Search | Builder

	Address	Size	Version	Checksum	Type	Information
1	_FIT_	00000080h	0100h	13h	FIT Header	
2	00000000FFE10060h	00018400h	0100h	00h	Microcode	CPUID: 000506E3h, Revision: 000000C6h, Date: 17.04.2018
3	00000000FFEB8000h	00008000h	0100h	00h	BIOS ACM	LocalOffset: 00008000h, EntryPoint: 00003BB1h, ACM SVN: 0002h, Date: 24.06.2015
4	00000000FFFE0000h	00002000h	0100h	00h	BIOS Init	
5	00000000FFEC0000h	00012000h	0100h	00h	BIOS Init	
6	00000000FFDD0000h	00003000h	0100h	00h	BIOS Init	
7	00000000FFEB5000h	00000241h	0100h	00h	BootGuard Key Manifest	LocalOffset: 00005000h, KM Version: 10h, KM SVN: 00h, KM ID: 01h
8	00000000FFEB2000h	000002D3h	0100h	00h	BootGuard Boot Policy	LocalOffset: 00002000h, BP SVN: 00h, ACM SVN: 02h

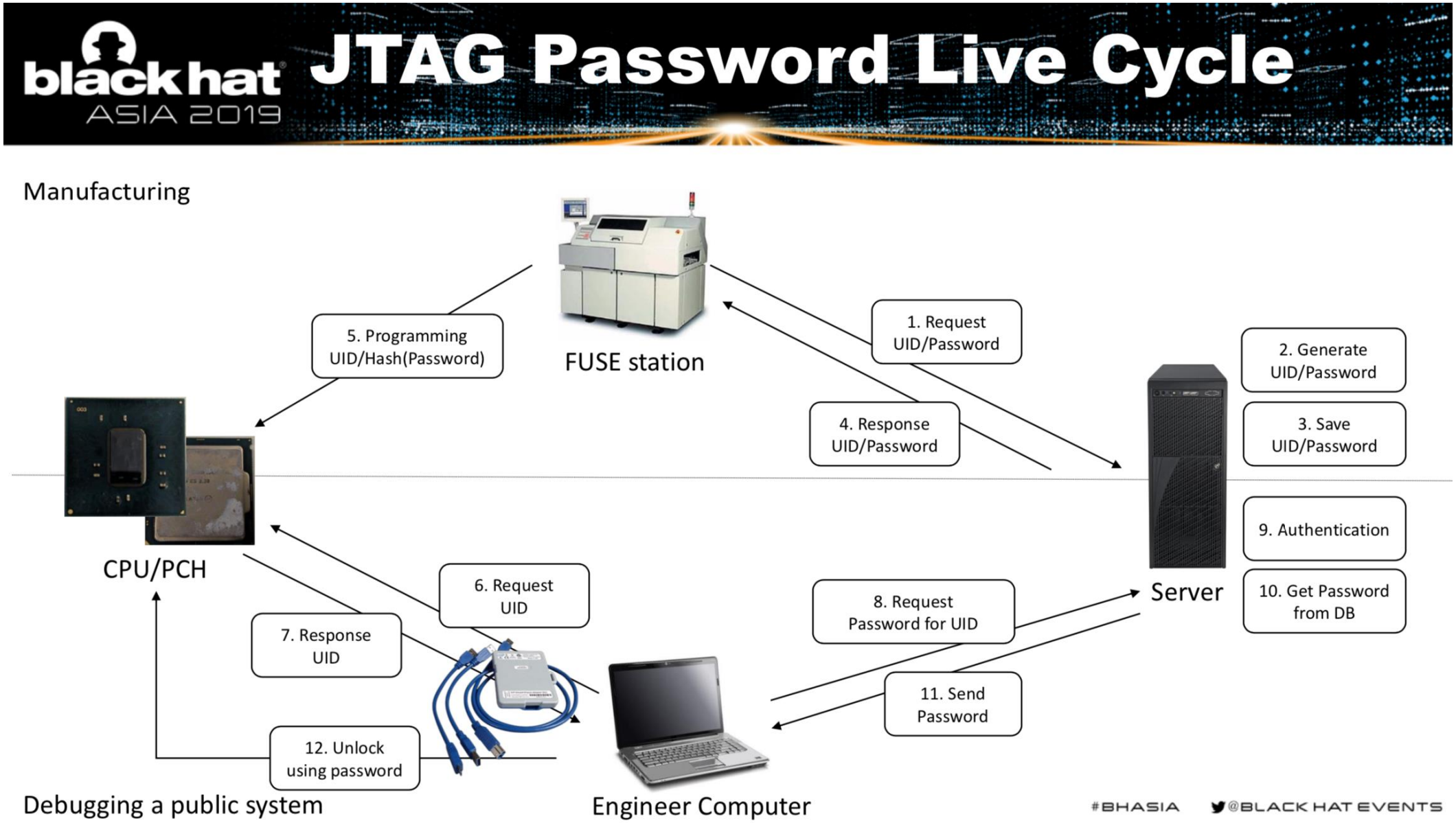
https://github.com/LongSoft/UEFITool

Why don’t lock everything in HW?

- **Hardware not flexible and expensive**
  - ✓ OEM’s don’t like locked secrets (supply chain)
  - ✓ The cost for the vulnerabilities very high (no updates)

- **All the vendors reducing HW locked secrets**
  - ✓ Even one locked bit in HW allow to say about HW locked feature
  - ✓ Mix Hardware + Firmware is common in actual implementation

HW manufacturing supply chain is very complex



https://www.blackhat.com/asia-19/briefings/schedule/index.html#intel-visa-through-the-rabbit-hole-13513

Intel Boot Guard:
New Ways to Bypass

oops I did it again

How HW-based Root of Trust become a SW

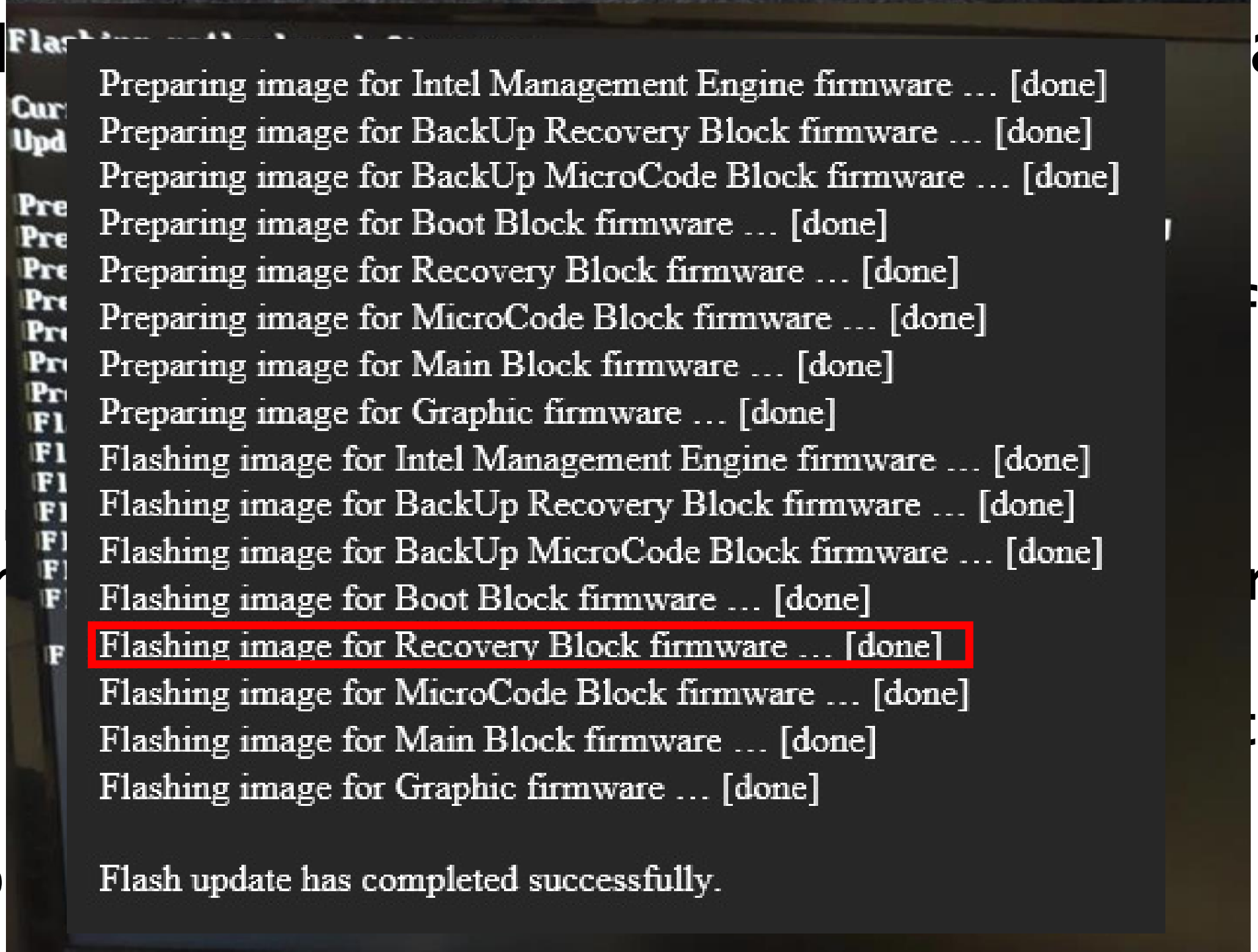
- Recovery mode is evil 😈
- Secure transition Chain of Trust on different boot stages is ~~slow~~ hard
- In most of the cases without hard reset Root of Trust moves to pure software for performance
- Enterprise hardware need remote update tools
- Nobody use Intel BIOS Guard even Intel :)

How H a SW

- Rec
- Sec ferent boot
- In Root of Tr rmance
- Ent tools
- Nob)

How H… a SW

- Rec…
- Sec… ferent boot
- In… Root of Tr… rmance
- Ent… tools
- Nob…)



How HW-based Root of Trust become a SW

```
text:FFF145EA                 push    [esp+70h+Sha256Buffer]
text:FFF145EE                 call    Sha256Init
text:FFF145F3                 push    [esi+BOOT_GUARD_DXE_HASH_CONTAINER.Block1Size]
text:FFF145F6                 push    [esi+BOOT_GUARD_DXE_HASH_CONTAINER.Block1BaseAddress]
text:FFF145F9                 push    [esp+7Ch+Sha256Buffer]
text:FFF145FD                 call    Sha256Calc
text:FFF14602                 lea     eax, [esp+80h+Block1CalculatedHash]
text:FFF14606                 push    eax
text:FFF14607                 push    [esp+84h+Sha256Buffer]
text:FFF1460B                 call    Sha256Calc2
text:FFF14610                 push    ebx
text:FFF14611                 lea     eax, [esp+8Ch+Block1CalculatedHash]
text:FFF14615                 push    eax
text:FFF14616                 add     esi, BOOT_GUARD_DXE_HASH_CONTAINER.Block1Sha256Hash
text:FFF14619                 push    esi
text:FFF1461A                 call    Compare
text:FFF1461F                 add     esp, 24h
text:FFF14622                 test    eax, eax
text:FFF14624                 jnz     short ReturnError
text:FFF14626                 or      byte ptr [edi+(size EFI_HOB_GUID_TYPE)], 1 ; VerificationResult
text:FFF1462A
text:FFF1462A ReturnOk:                               ; CODE XREF: UnknownCallBack+14F↑j
text:FFF1462A                                         ; UnknownCallBack+154↑j
text:FFF1462A                 mov     eax, [esp+70h+ExitCode]
text:FFF1462E                 jmp     short exit
text:FFF14630 ; ---------------------------------------------------------------------------
text:FFF14630
text:FFF14630 ReturnError:                            ; CODE XREF: UnknownCallBack+146↑j
text:FFF14630                                         ; UnknownCallBack+190↑j
text:FFF14630                 and     byte ptr [edi+(size EFI_HOB_GUID_TYPE)], 10h ; VerificationResult
text:FFF14634                 call    GetPeiServices
text:FFF14639                 mov     ecx, [eax]
text:FFF1463B                 push    ebx             ; BootMode
text:FFF1463C                 push    eax             ; PeiServices
text:FFF1463D                 call    [ecx+EFI_PEI_SERVICES.SetBootMode]
text:FFF14640                 pop     ecx
text:FFF14641                 pop     ecx
text:FFF14642                 call    InstallBootInRecoveryModePpi
text:FFF14647                 xor     eax, eax
text:FFF14649
text:FFF14649 exit:                                   ; CODE XREF: UnknownCallBack+64↑j
text:FFF14649                                         ; UnknownCallBack+6F↑j ...
text:FFF14649                 pop     edi
text:FFF1464A                 pop     esi
text:FFF1464B                 pop     ebx
text:FFF1464C                 mov     esp, ebp
text:FFF1464E                 pop     ebp
text:FFF1464F                 retn
text:FFF1464F UnknownCallBack endp
```

https://embedi.org/blog/nuclear-explotion/

How HW-based Root of Trust become a SW

```
text:FFF145EA                 push    [esp+70h+Sha256Buffer]
text:FFF145EE                 call    Sha256Init
text:FFF145F3                 push    [esi+BOOT_GUARD_DXE_HASH_CONTAINER.Block1Size]
text:FFF145F6                 push    [esi+BOOT_GUARD_DXE_HASH_CONTAINER.Block1BaseAddress]
text:FFF145F9                 push    [esp+7Ch+Sha256Buffer]
text:FFF145FD                 call    Sha256Calc
text:FFF14602                 lea     eax, [esp+80h+Block1CalculatedHash]
text:FFF14606                 push    eax
text:FFF14607                 push    [esp+84h+Sha256Buffer]
text:FFF1460B                 call    Sha256Calc2
text:FFF14610                 push    ebx
text:FFF14611                 lea     eax, [esp+8Ch+Block1CalculatedHash]
```

```
text:FFF1460B         call    Sha256Calc2
text:FFF14610         push    ebx
text:FFF14611         lea     eax, [esp+8Ch+Block1CalculatedHash]
text:FFF14615         push    eax
text:FFF14616         add     esi, BOOT_GUARD_DXE_HASH_CONTAINER.Block1Sha256Hash
text:FFF14619         push    esi
text:FFF1461A         call    Compare
text:FFF1461F         add     esp, 24h
text:FFF14622         test    eax, eax
text:FFF14624         jnz     short ReturnError
text:FFF14626         or      byte ptr [edi+(size EFI_HOB_GUID_TYPE)], 1 ; VerificationResult
```

```
text:FFF14640                 pop     ecx
text:FFF14641                 pop     ecx
text:FFF14642                 call    InstallBootInRecoveryModePpi
text:FFF14647                 xor     eax, eax
text:FFF14649
text:FFF14649 exit:                                   ; CODE XREF: UnknownCallBack+64↑j
text:FFF14649                                         ; UnknownCallBack+6F↑j ...
text:FFF14649                 pop     edi
text:FFF1464A                 pop     esi
text:FFF1464B                 pop     ebx
text:FFF1464C                 mov     esp, ebp
text:FFF1464E                 pop     ebp
text:FFF1464F                 retn
text:FFF1464F UnknownCallBack endp
```

https://embedi.org/blog/nuclear-explotion/

How HW-based Root of Trust become a SW

```
text:FFF145EA                 push    [esp+70h+Sha256Buffer]
text:FFF145EE                 call    Sha256Init
text:FFF145F3                 push    [esi+BOOT_GUARD_DXE_HASH_CONTAINER.Block1Size]
text:FFF145F6                 push    [esi+BOOT_GUARD_DXE_HASH_CONTAINER.Block1BaseAddress]
text:FFF145F9                 push    [esp+7Ch+Sha256Buffer]
text:FFF145FD                 call    Sha256Calc
text:FFF14602                 lea     eax, [esp+80h+Block1CalculatedHash]
text:FFF14606                 push    eax
text:FFF14607                 push    [esp+84h+Sha256Buffer]
text:FFF1460B                 call    Sha256Calc2
text:FFF14610                 push    ebx
text:FFF14611                 lea     eax, [esp+8Ch+Block1CalculatedHash]
```

```
text:FFF1460B                 call    Sha256Calc2
```

```
if (memcmp(&HashContainer.BlockHash, &CalculatedHash, SHA256_DIGEST_SIZE))
      *(BootGuardPeiHob + 0x18) = 0;  // The stored value (verification result)
else                                  // is ignored!
      // Start Recovery!
```

```
text:FFF14626                 or      byte ptr [edi+(size EFI_HOB_GUID_TYPE)], 1 ; VerificationResult
text:FFF14640                 pop     ecx
text:FFF14641                 pop     ecx
text:FFF14642                 call    InstallBootInRecoveryModePpi
text:FFF14647                 xor     eax, eax
text:FFF14649
text:FFF14649 exit:                                   ; CODE XREF: UnknownCallBack+64↑j
text:FFF14649                                         ; UnknownCallBack+6F↑j ...
text:FFF14649                 pop     edi
text:FFF1464A                 pop     esi
text:FFF1464B                 pop     ebx
text:FFF1464C                 mov     esp, ebp
text:FFF1464E                 pop     ebp
text:FFF1464F                 retn
text:FFF1464F UnknownCallBack endp
```

https://embedi.org/blog/nuclear-explotion/

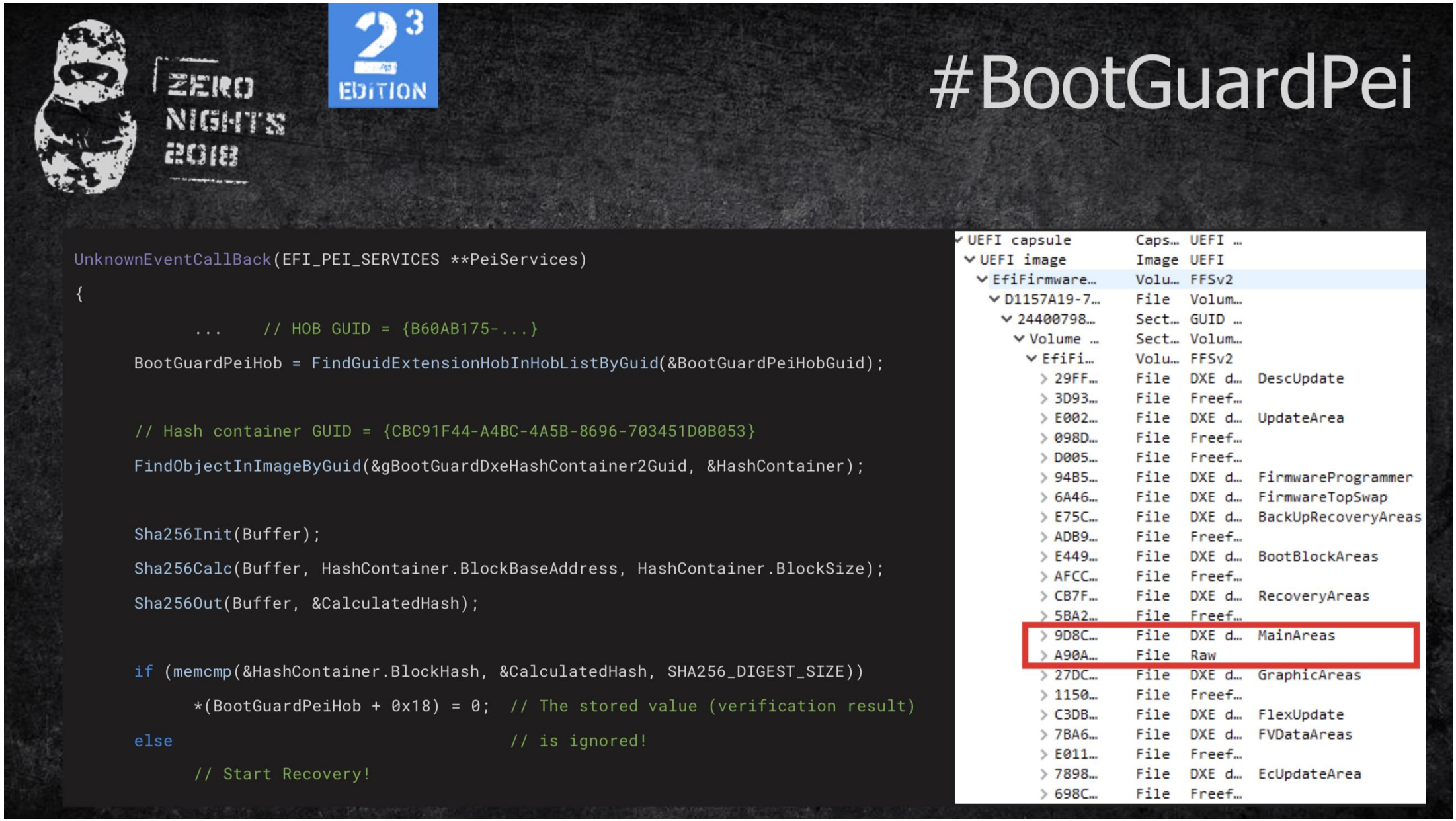
2³ EDITION

CVE-2018-12158

INTEL-SA-00168

A tribute to: What makes OS drivers dangerous for BIOS?
by Alex Matrosov @matrosov
https://medium.com/@matrosov/dangerous-update-tools-c246f7299459

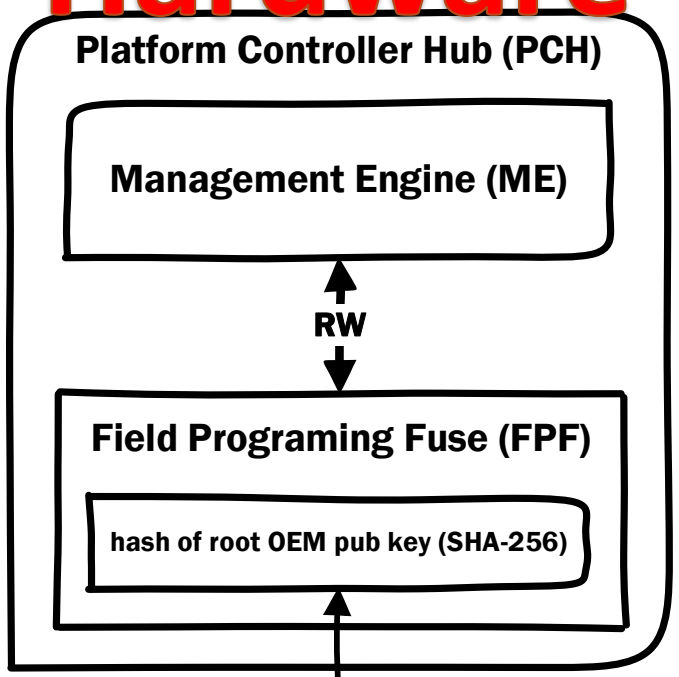
How HW-based Root of Trust become a SW



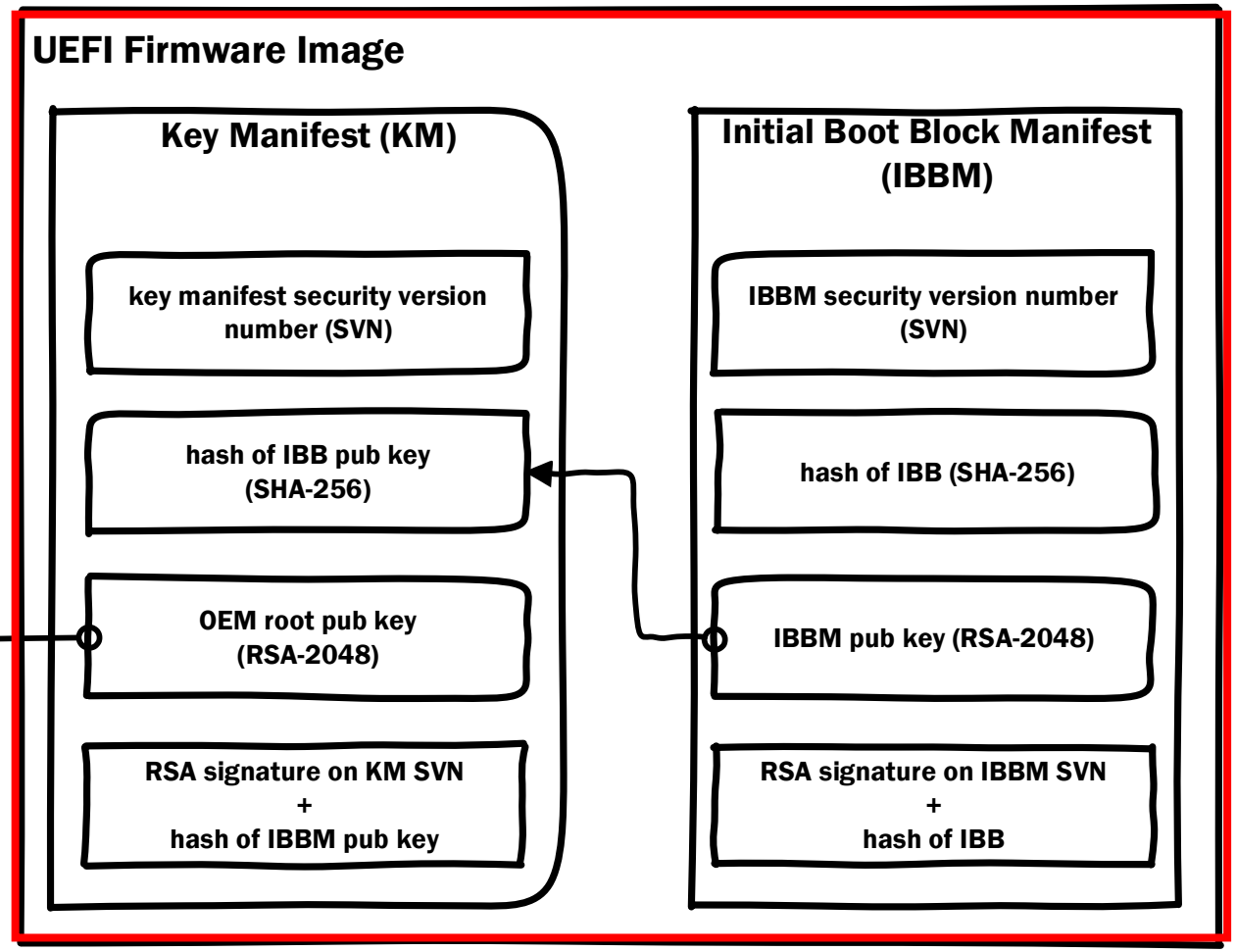
https://2018.zeronights.ru/en/wp-content/uploads/materials/06-NUClear-explotion.pdf

Boot Guard Bypass

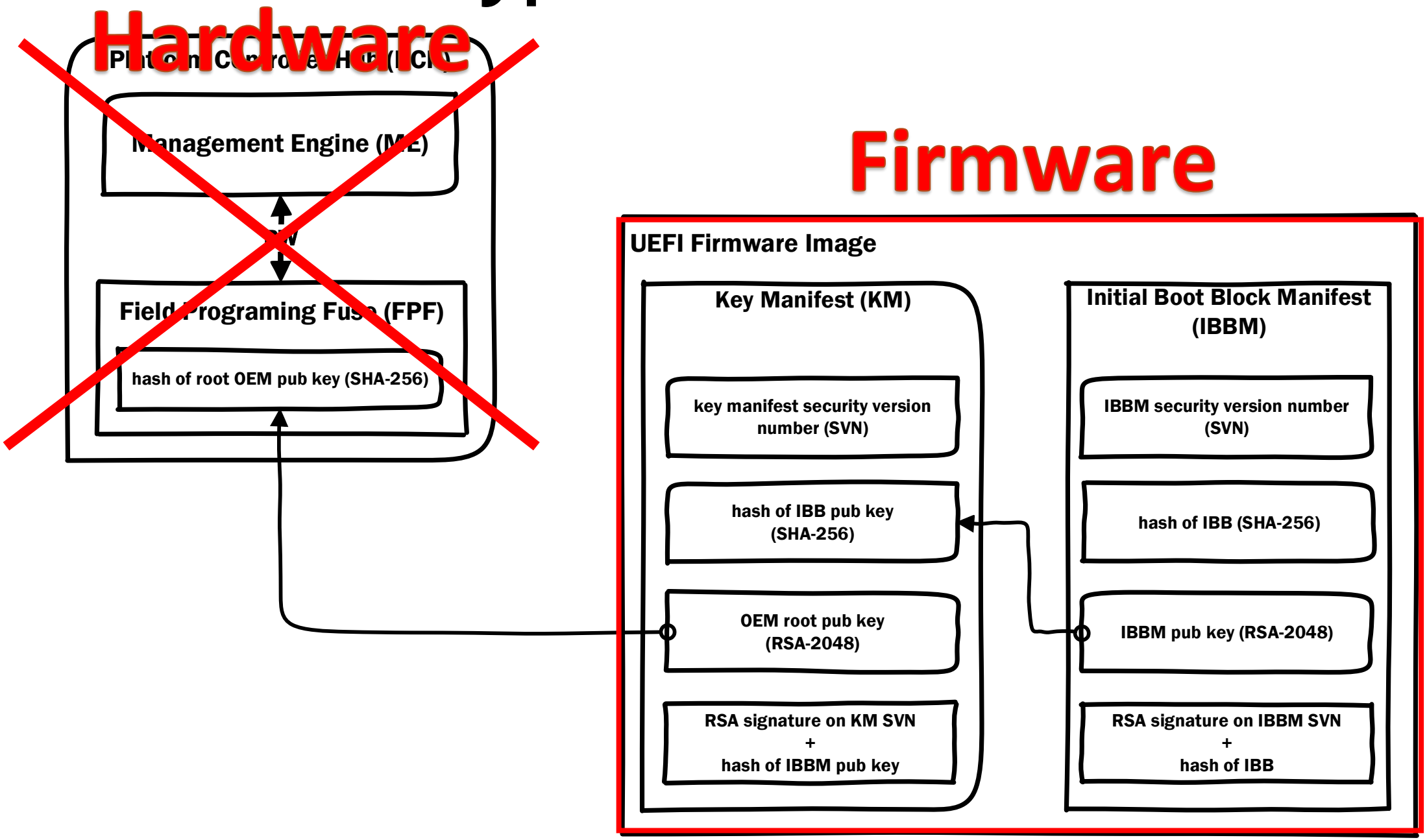
Hardware



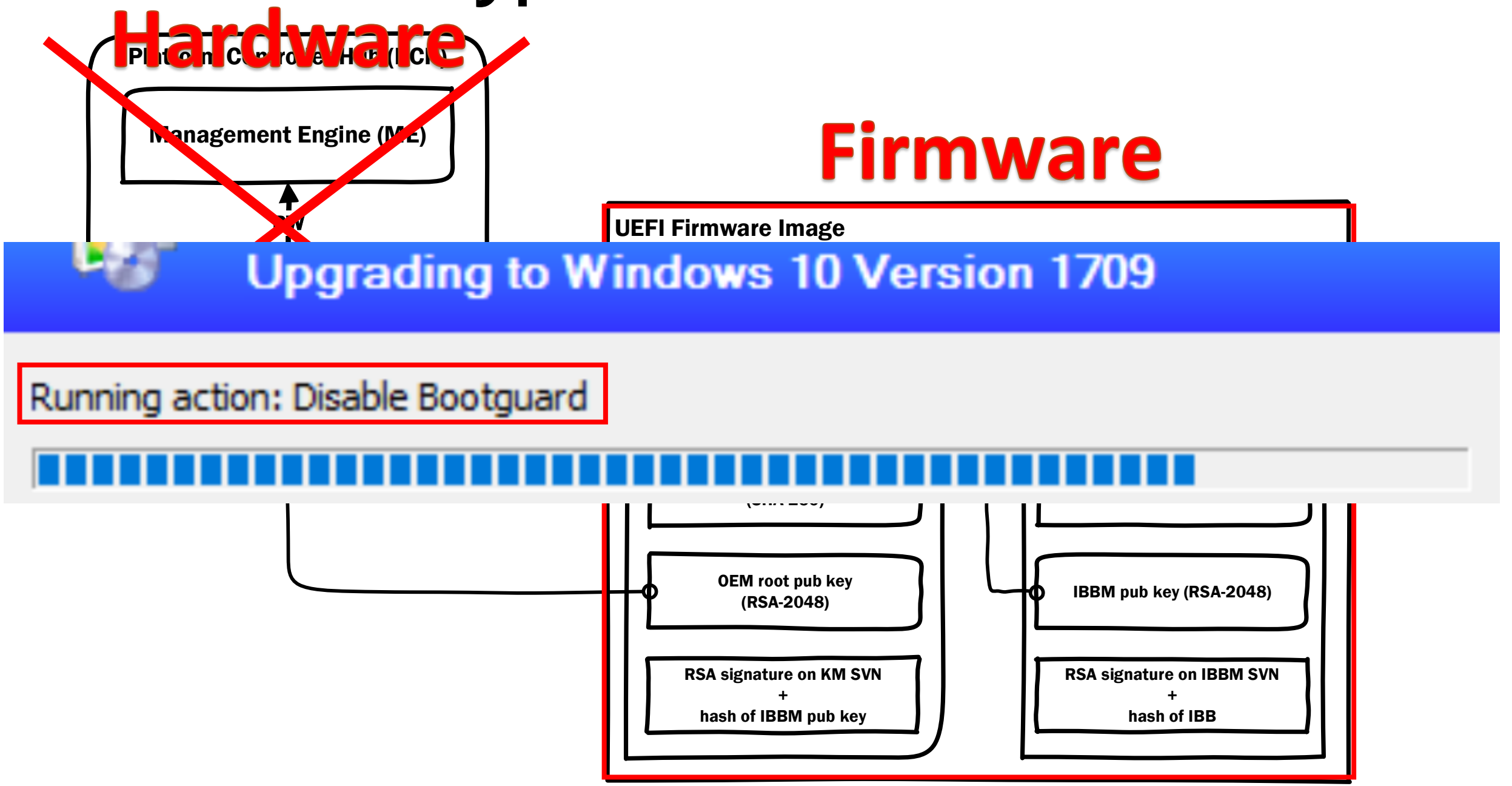
Firmware



Boot Guard Bypass

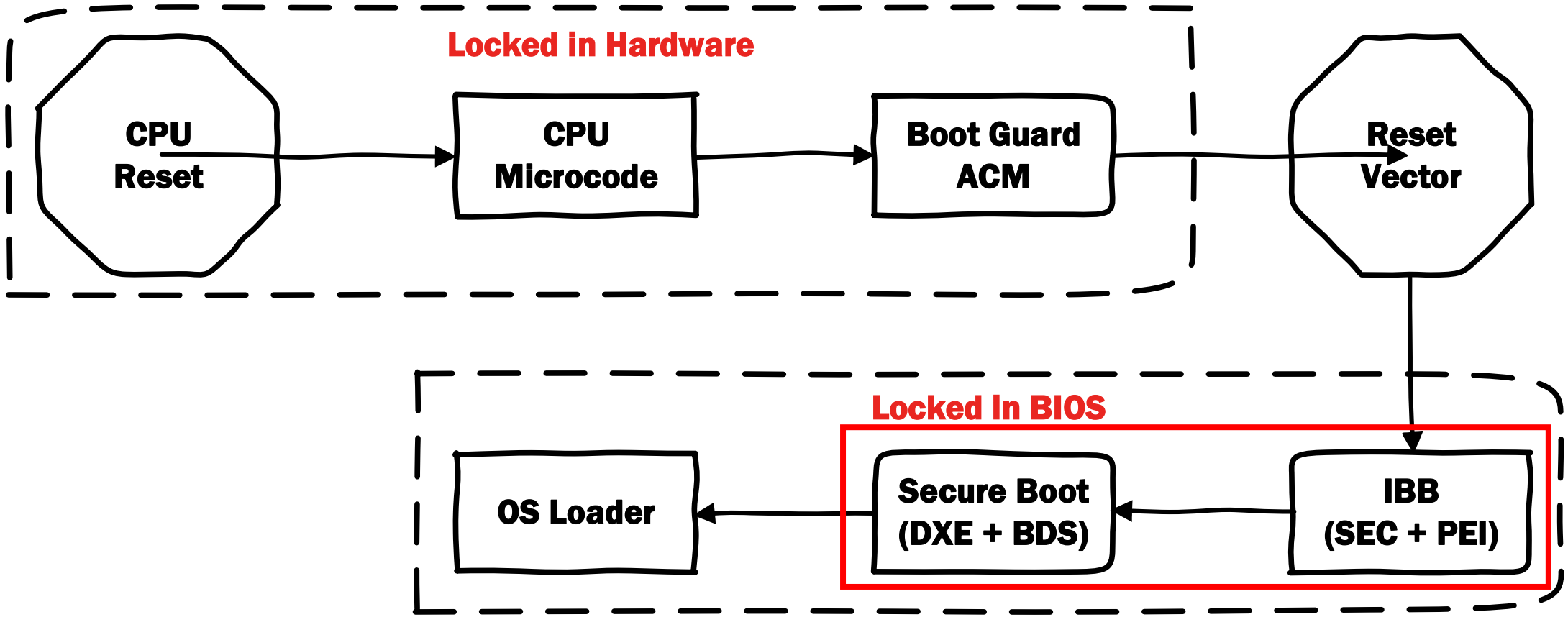


Boot Guard Bypass

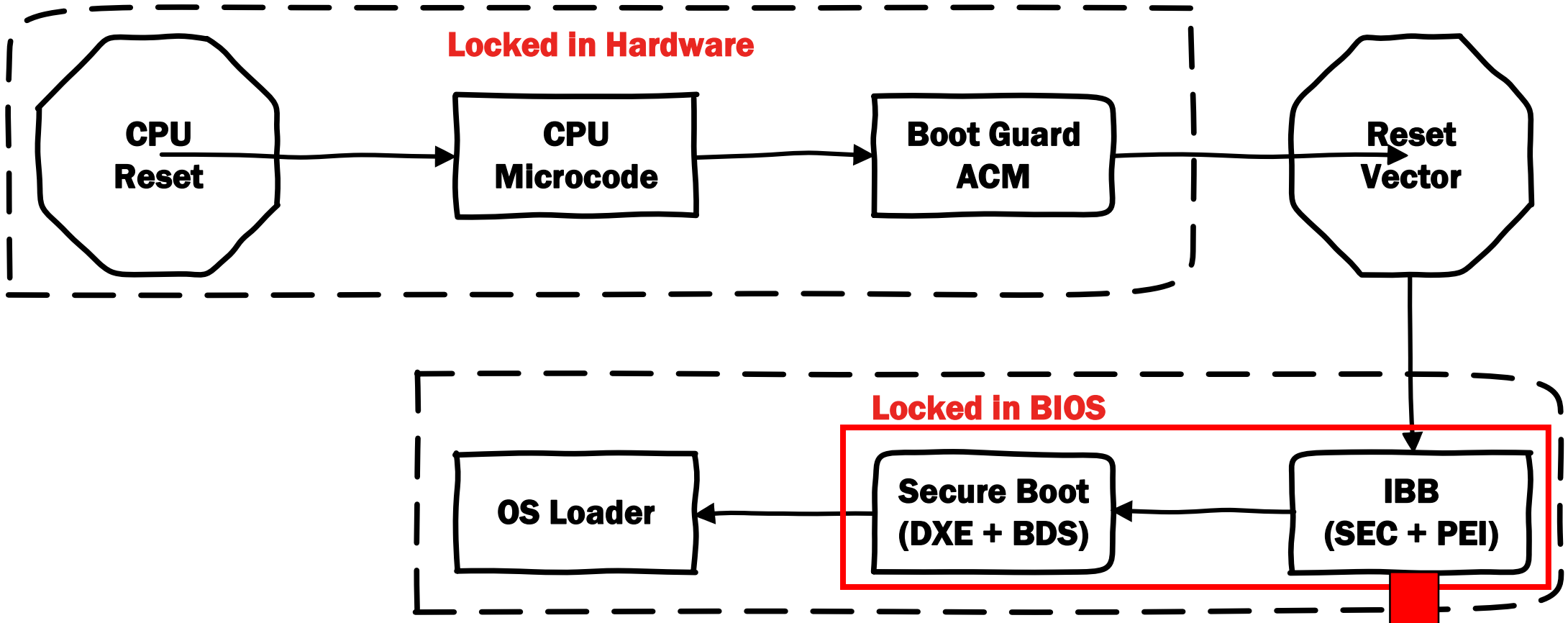


Boot Guard Bypass: LenovoPcdInit

Boot Guard: Boot Flow in ACTIVE manufacturing mode



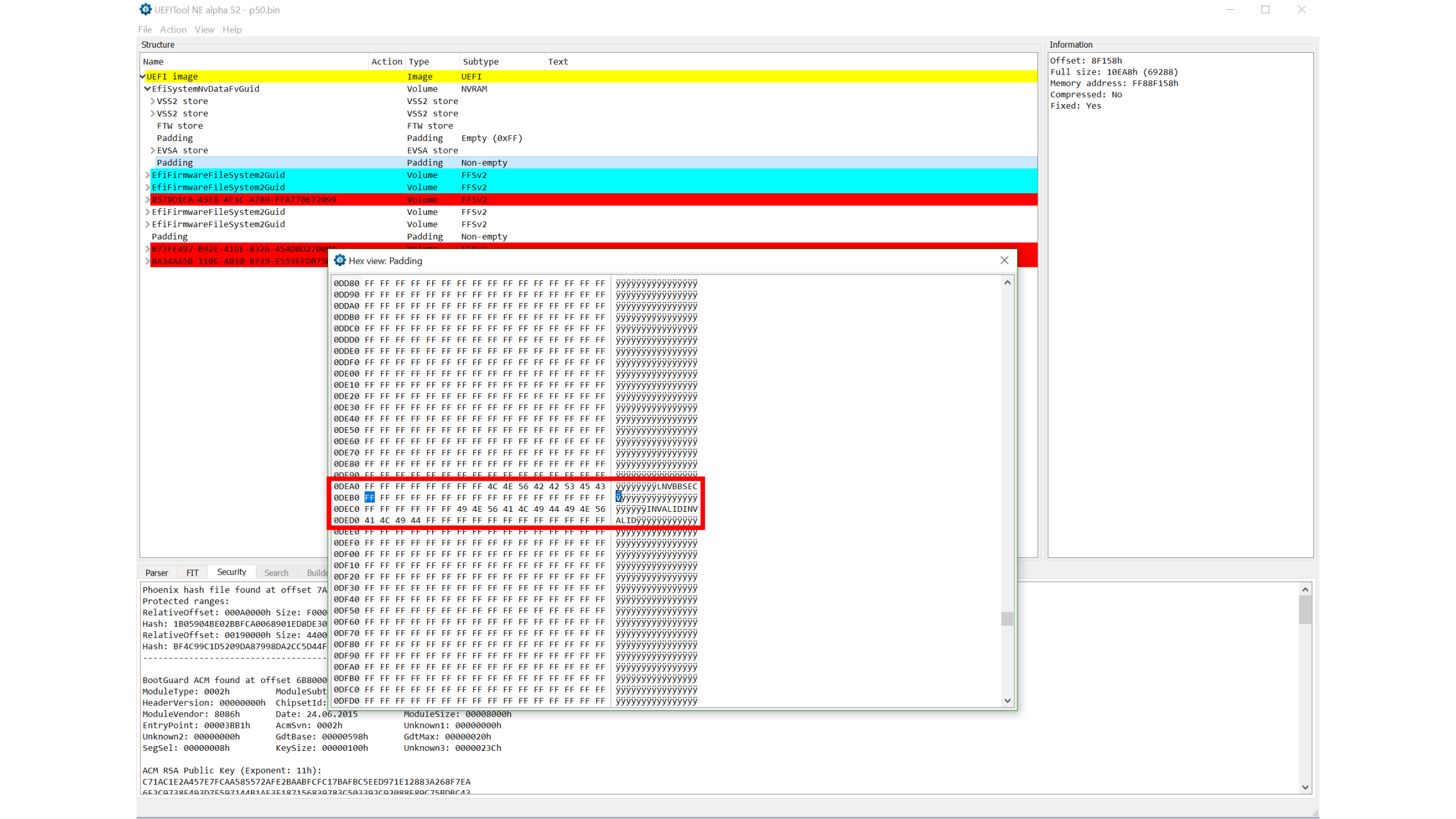
Boot Guard: Boot Flow in ACTIVE manufacturing mode



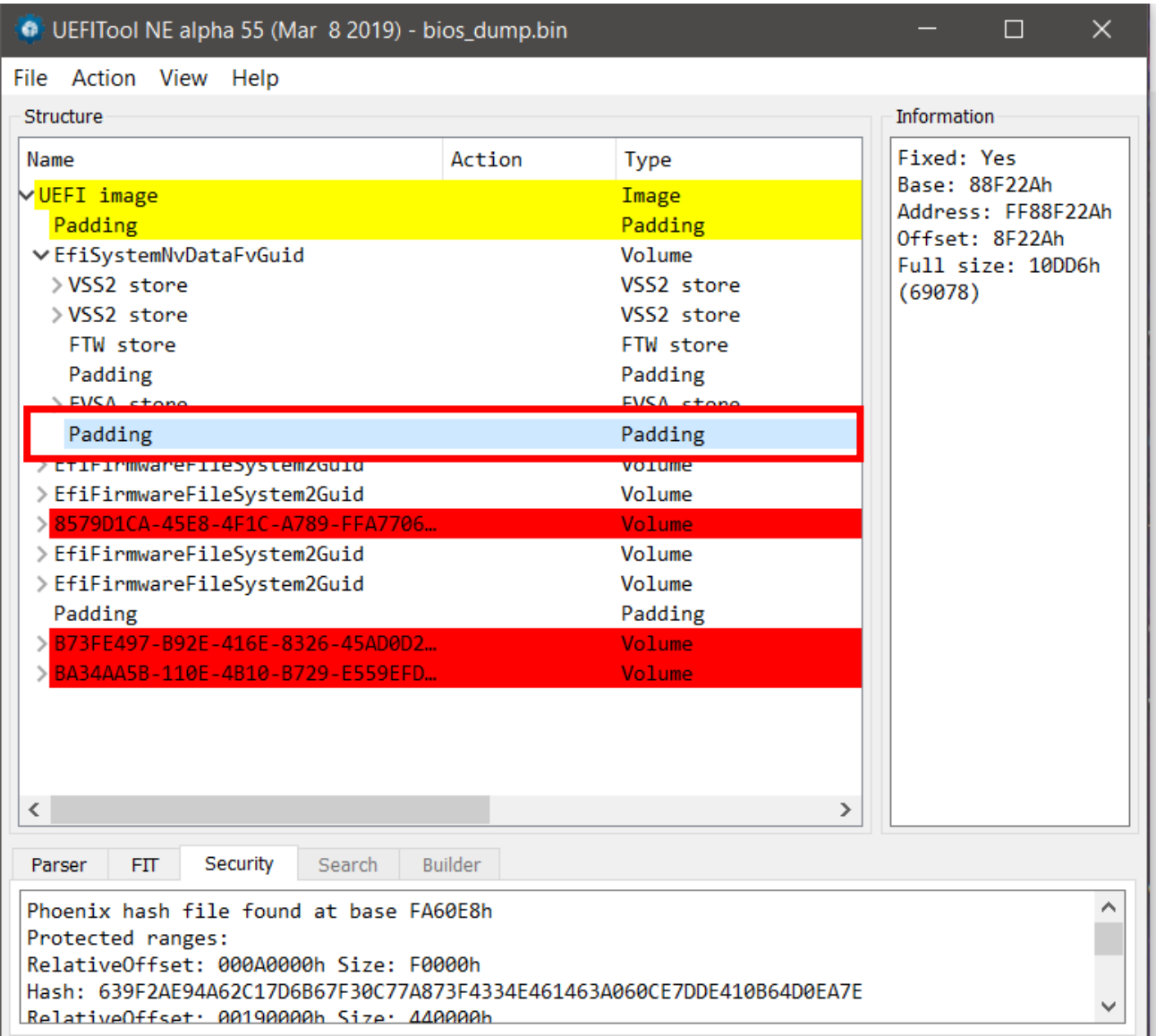

```
// IF manufacturing mode ACTIVE, skip verificaiton Boot Guard IBB for  DXE Verification.

if (PcdManufacturingMode == TRUE) {
  return EFI_SUCCESS;
}
```

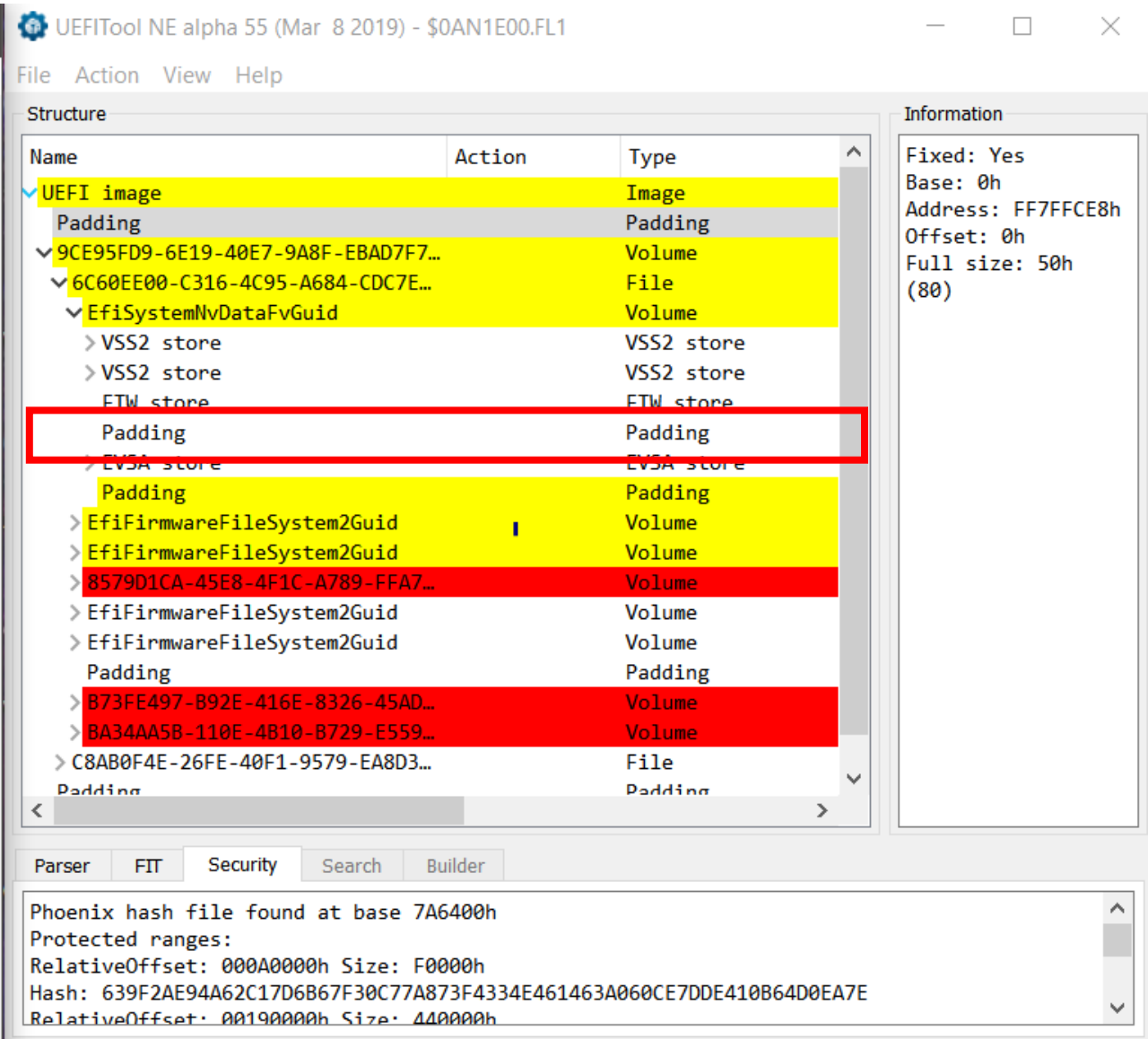
Boot Guard Bypass: Where Lenovo PCD stored?

UEFI image	Image	UEFI
EfiSystemNvDataFvGuid	Volume	NVRAM
VSS2 store	VSS2 store	
Free space	Free space	
VSS2 store	VSS2 store	
Free space	Free space	
FTW store	FTW store	
Padding	Padding	Empty (0xFF)
EVSA store	EVSA store	
Padding	Padding	Non-empty
EfiFirmwareFileSystem2Guid	Volume	FFSv2
EfiFirmwareFileSystem2Guid	Volume	FFSv2
8579D1CA-45E8-4F1C-A789-FFA7706…	Volume	FFSv2
EfiFirmwareFileSystem2Guid	Volume	FFSv2
EfiFirmwareFileSystem2Guid	Volume	FFSv2
Padding	Padding	Non-empty
B73FE497-B92E-416E-8326-45AD0D2…	Volume	FFSv2
BA34AA5B-110E-4B10-B729-E559EFD…	Volume	FFSv2


UEFITool NE alpha 52 - p50.bin
File Action View Help
Structure
Name Action Type Subtype Text
UEFI image Image UEFI
EfiSystemNvDataFvGuid Volume NVRAM
VSS2 store VSS2 store
VSS2 store VSS2 store
FTW store FTW store
Padding Padding Empty (0xFF)
EVSA store EVSA store
Padding Padding Non-empty
EfiFirmwareFileSystem2Guid Volume FFSv2
EfiFirmwareFileSystem2Guid Volume FFSv2
8579D1CA-45E8-4F1C-A789-FFA770672099 Volume FFSv2
EfiFirmwareFileSystem2Guid Volume FFSv2
EfiFirmwareFileSystem2Guid Volume FFSv2
Padding Padding Non-empty
B73FE497-B92E-416E-8326-45AD0D270091
BA34AA5B-110E-4B10-B729-E559EFD075
Information
Offset: 8F158h
Full size: 10EA8h (69288)
Memory address: FF88F158h
Compressed: No
Fixed: Yes
Hex view: Padding
0DD80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DD90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DDA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DDB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DDC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DDD0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DDE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DDF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DE00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DE10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DE20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DE30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DE40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DE50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DE60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DE70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DE80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DE90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DEA0 FF FF FF FF FF FF FF 4C 4E 56 42 42 53 45 43 ÿÿÿÿÿÿÿLNVBBSEC
0DEB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DEC0 FF FF FF FF FF FF 49 4E 56 41 4C 49 44 49 4E 56 ÿÿÿÿÿÿINVALIDINV
0DED0 41 4C 49 44 FF FF FF FF FF FF FF FF FF FF FF FF ALIDÿÿÿÿÿÿÿÿÿÿÿÿ
0DEE0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DEF0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DF00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DF10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DF20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DF30 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DF40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DF50 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DF60 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DF70 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DF80 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DF90 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DFA0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DFB0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DFC0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0DFD0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
Parser FIT Security Search Builde
Phoenix hash file found at offset 7A
Protected ranges:
RelativeOffset: 000A0000h Size: F000
Hash: 1B05904BE02BBFCA0068901ED8DE30
RelativeOffset: 00190000h Size: 4400
Hash: BF4C99C1D5209DA87998DA2CC5D44F
BootGuard ACM found at offset 6B8000
ModuleType: 0002h ModuleSubt
HeaderVersion: 00000000h ChipsetId:
ModuleVendor: 8086h Date: 24.06.2015 ModuleSize: 00008000h
EntryPoint: 00003BB1h AcmSvn: 0002h Unknown1: 00000000h
Unknown2: 00000000h GdtBase: 00000598h GdtMax: 00000020h
SegSel: 00000008h KeySize: 00000100h Unknown3: 0000023Ch
ACM RSA Public Key (Exponent: 11h):
C71AC1E2A457E7FCAA585572AFE2BAABFCFC17BAFBC5EED971E12883A268F7EA

Boot Guard Bypass: Going deeper with SPI dump





Padding_Non-empty_Padding1.pad ×

Edit As: Hex Run Script Run Template

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
DD40h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DD50h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DD60h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DD70h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DD80h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DD90h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DDA0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DDB0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DDC0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DDD0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DDE0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DDF0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE00h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE10h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE20h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE30h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE40h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE50h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE60h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE70h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE80h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ

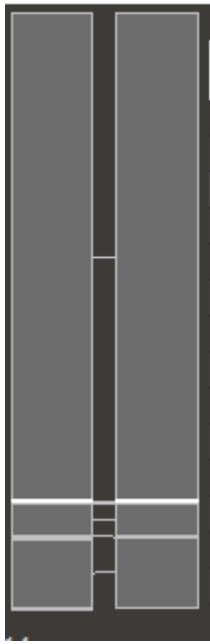
Padding_Non-empty_Padding.pad ×

Edit As: Hex Run Script Run Template

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
DD40h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DD50h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DD60h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DD70h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DD80h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DD90h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DDA0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DDB0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DDC0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DDD0h:	FF	FF	FF	FF	FF	FF	4C	4E	56	42	42	53	45	43	FB	FF	ÿÿÿÿÿÿLNVBBSECûÿ
DDE0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DDF0h:	FF	FF	FF	FF	49	4E	56	41	4C	49	44	49	4E	56	41	4C	ÿÿÿÿINVALIDINVAL
DE00h:	49	44	1A	A4	03	BF	56	DB	F4	61	17	06	FF	FF	FF	FF	ID.¤.¿VÛôa..ÿÿÿÿ
DE10h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE20h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE30h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE40h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE50h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
DE60h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ

Selected: 21 [15h] bytes (Range: 56799 [DDDFh] to 56819 [DDF3h])

Compare



C:\...\Padding_Non-empty_Padding1.pad vs. C:\...\Padding_Non-empty_Padding.pad

Result	Address A	Size A	Address B	Size B
Match	0h	DDD6h	0h	DDD6h
Only in B			DDD6h	9h
Match	DDD6h	15h	DDDFh	15h
Only in B			DDF4h	18h
Match	DDEBh	BDh	DE0Ch	BDh
Only in A	DEA8h	8h		
Match	DEB0h	16h	DEC9h	16h
Only in A	DEC6h	Eh		
Match	DED4h	EF7h	DEDFh	EF7h
Only in B			EDD6h	22h
Match	EDCBh	DDh	EDF8h	DDh
Only in A	EEA8h	22h		
Match	EECAh	1F01h	EED5h	1F01h
Only in A	10DCBh	DDh		

Why vendors leave this “backdoors”?

- Creating recover process for broken BIOS updates possible (even remotely).

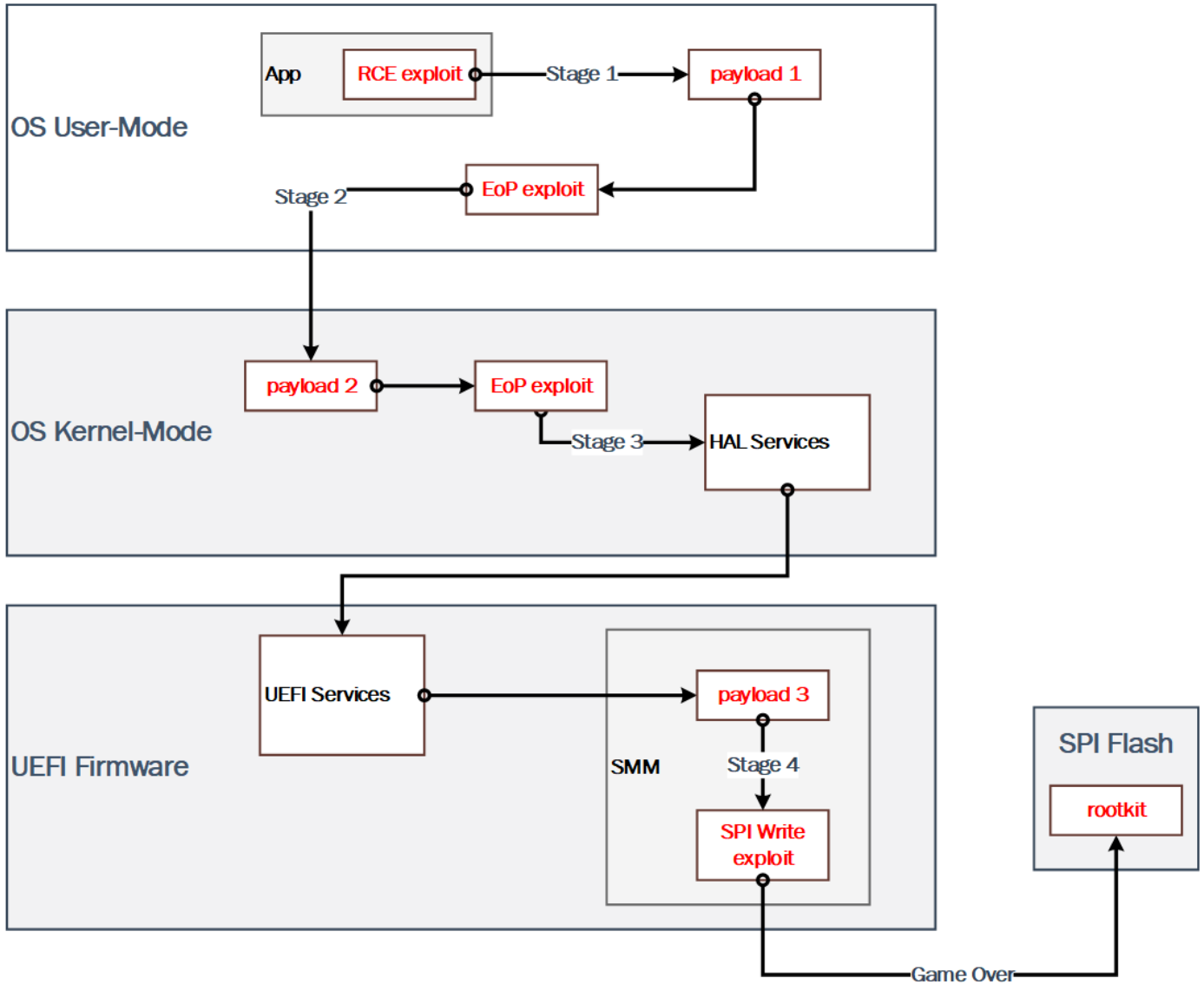
- But leaving “backdoors” is always create another problems even more serious.

- Enterprise market need stable solutions right? ☺

- Replace broken HW is expensive way but only one which guarantees security process for system recovery

SMI over WMI is evil

How many exploits you need?



https://medium.com/@matrosov/dangerous-update-tools-c246f7299459

How this REsearch get started?

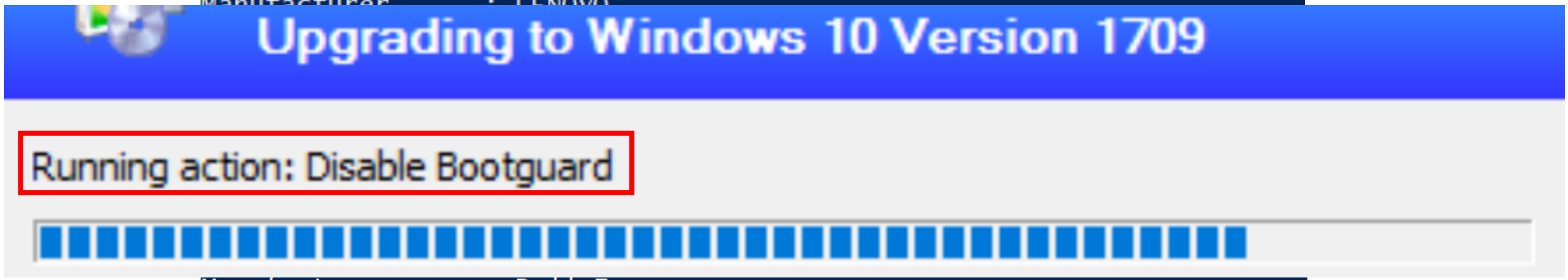
```
PS C:\Users\[redacted]>  Get-WmiObject -Query "Select * from Win32_Bios"


SMBIOSBIOSVersion : N1EET79W (1.52 )
Manufacturer      : LENOVO
Name              : N1EET79W (1.52 )
SerialNumber      : PC0B7VJT
Version           : LENOVO - 1520
```

```
PS C:\Users\[redacted]> Get-WmiObject -Query "Select * from Win32_Bios"


SMBIOSBIOSVersion : 1.11.0
Manufacturer      : Dell Inc.
Name              : 1.11.0
SerialNumber      : 70BJMH2
Version           : DELL   - 1072009
```

https://docs.microsoft.com/en-us/windows/desktop/cimwin32prov/win32-bios

How this REsearch get started?

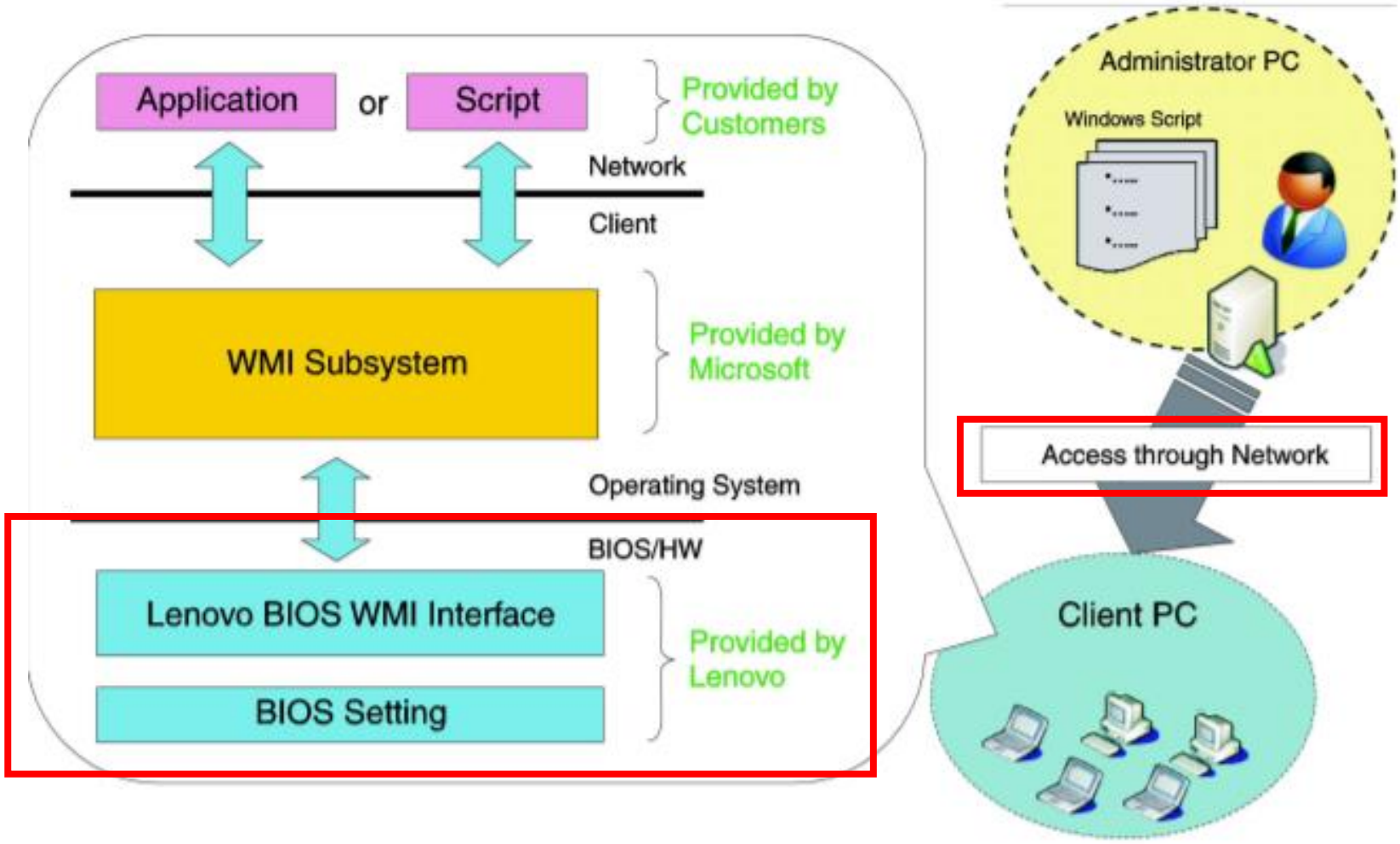
```
PS C:\Users\[redacted]>  Get-WmiObject -Query "Select * from Win32_Bios"

SMBIOSBIOSVersion : N1EET79W (1.52 )
Manufacturer      : LENOVO
```




```
Manufacturer      : Dell Inc.
Name              : 1.11.0
SerialNumber      : 70BJMH2
Version           : DELL   - 1072009
```

https://docs.microsoft.com/en-us/windows/desktop/cimwin32prov/win32-bios

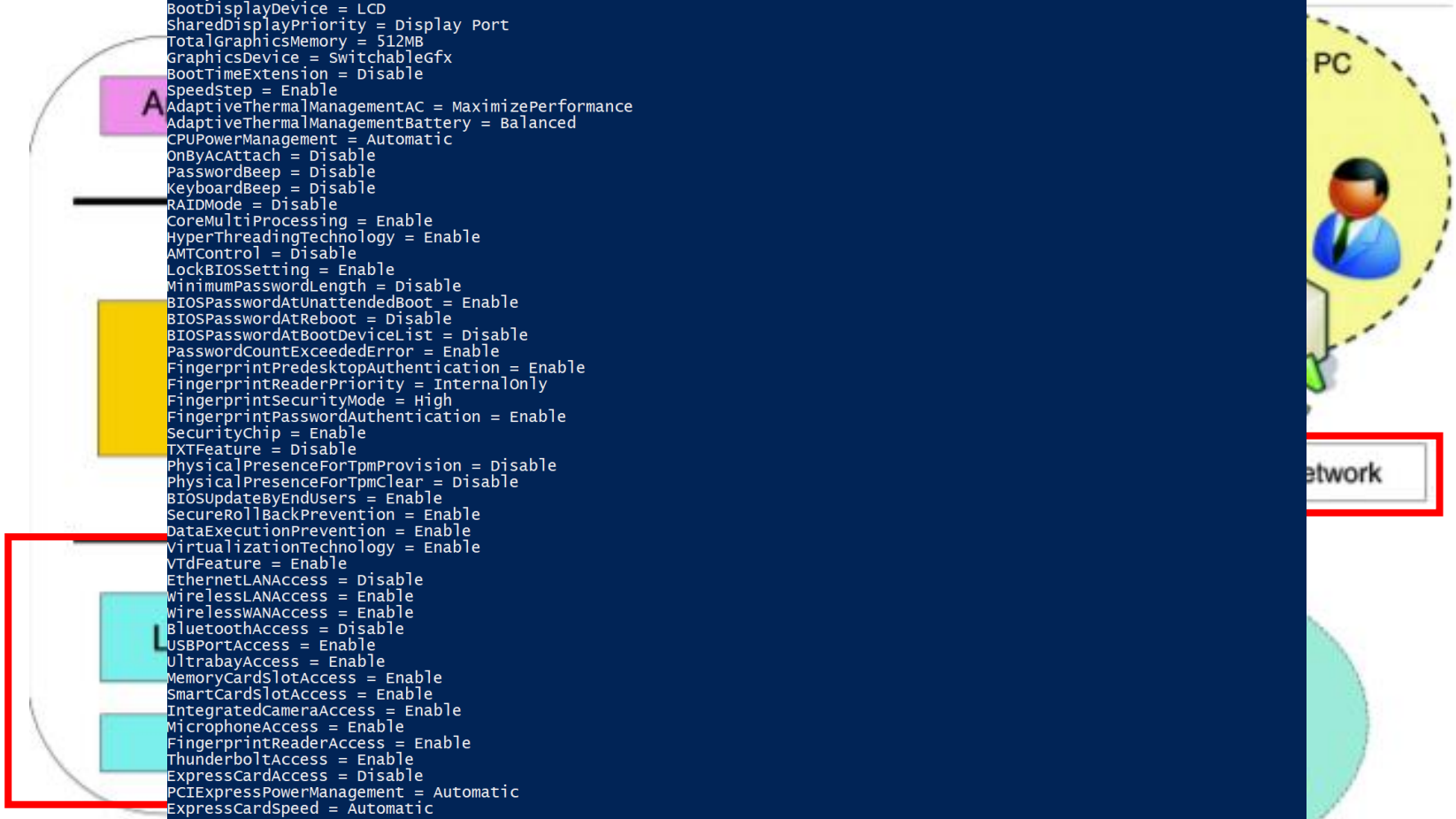
SMI over WMI is evil



https://download.lenovo.com/pccbbs/mobiles_pdf/kbl-r_deploy_01.pdf

SMI over

```
PS C:\WINDOWS\system32> gwmi -class Lenovo_BiosSetting -namespace root\wmi | ForEach-Object {if ($_.CurrentSetting -ne "
") {Write-Host $_.CurrentSetting.replace(",","=")}}
WakeOnLAN = Disable
EthernetLANOptionROM = Enable
IPv4NetworkStack = Disable
IPv6NetworkStack = Disable
UefiPxeBootPriority = IPv4First
Reserved = Disable
USBBIOSSupport = Disable
AlwaysOnUSB = Disable
TrackPoint = Automatic
TouchPad = Automatic
FnCtrlKeySwap = Disable
FnSticky = Disable
FnKeyAsPrimary = Disable
BootDisplayDevice = LCD
SharedDisplayPriority = Display Port
TotalGraphicsMemory = 512MB
GraphicsDevice = SwitchableGfx
BootTimeExtension = Disable
SpeedStep = Enable
AdaptiveThermalManagementAC = MaximizePerformance
AdaptiveThermalManagementBattery = Balanced
CPUPowerManagement = Automatic
OnByAcAttach = Disable
PasswordBeep = Disable
KeyboardBeep = Disable
RAIDMode = Disable
CoreMultiProcessing = Enable
HyperThreadingTechnology = Enable
AMTControl = Disable
LockBIOSSetting = Enable
MinimumPasswordLength = Disable
BIOSPasswordAtUnattendedBoot = Enable
BIOSPasswordAtReboot = Disable
BIOSPasswordAtBootDeviceList = Disable
PasswordCountExceededError = Enable
FingerprintPredesktopAuthentication = Enable
FingerprintReaderPriority = InternalOnly
FingerprintSecurityMode = High
FingerprintPasswordAuthentication = Enable
SecurityChip = Enable
TXTFeature = Disable
PhysicalPresenceForTpmProvision = Disable
PhysicalPresenceForTpmClear = Disable
BIOSUpdateByEndUsers = Enable
SecureRollBackPrevention = Enable
DataExecutionPrevention = Enable
VirtualizationTechnology = Enable
VTdFeature = Enable
EthernetLANAccess = Disable
WirelessLANAccess = Enable
WirelessWANAccess = Enable
BluetoothAccess = Disable
USBPortAccess = Enable
UltrabayAccess = Enable
MemoryCardSlotAccess = Enable
SmartCardSlotAccess = Enable
IntegratedCameraAccess = Enable
MicrophoneAccess = Enable
FingerprintReaderAccess = Enable
ThunderboltAccess = Enable
ExpressCardAccess = Disable
PCIExpressPowerManagement = Automatic
ExpressCardSpeed = Automatic
RAIDStorage = SATAHDD
BottomCoverTamperDetected = Disable
InternalStorageTamper = Disable
ComputraceModuleActivation = Disable
SecureBoot = Enable
SGXControl = Enable
DeviceGuard = Disable
BootMode = Quick
StartupOptionKeys = Enable
BootDeviceListF12Option = Enable
BootOrder = USBCD:USBHDD:USBFDD:NVMe0:NVMe1:HDD0:HDD1:HDD2:HDD3:PCILAN
NetworkBoot = PCILAN
BootOrderLock = Enable
```

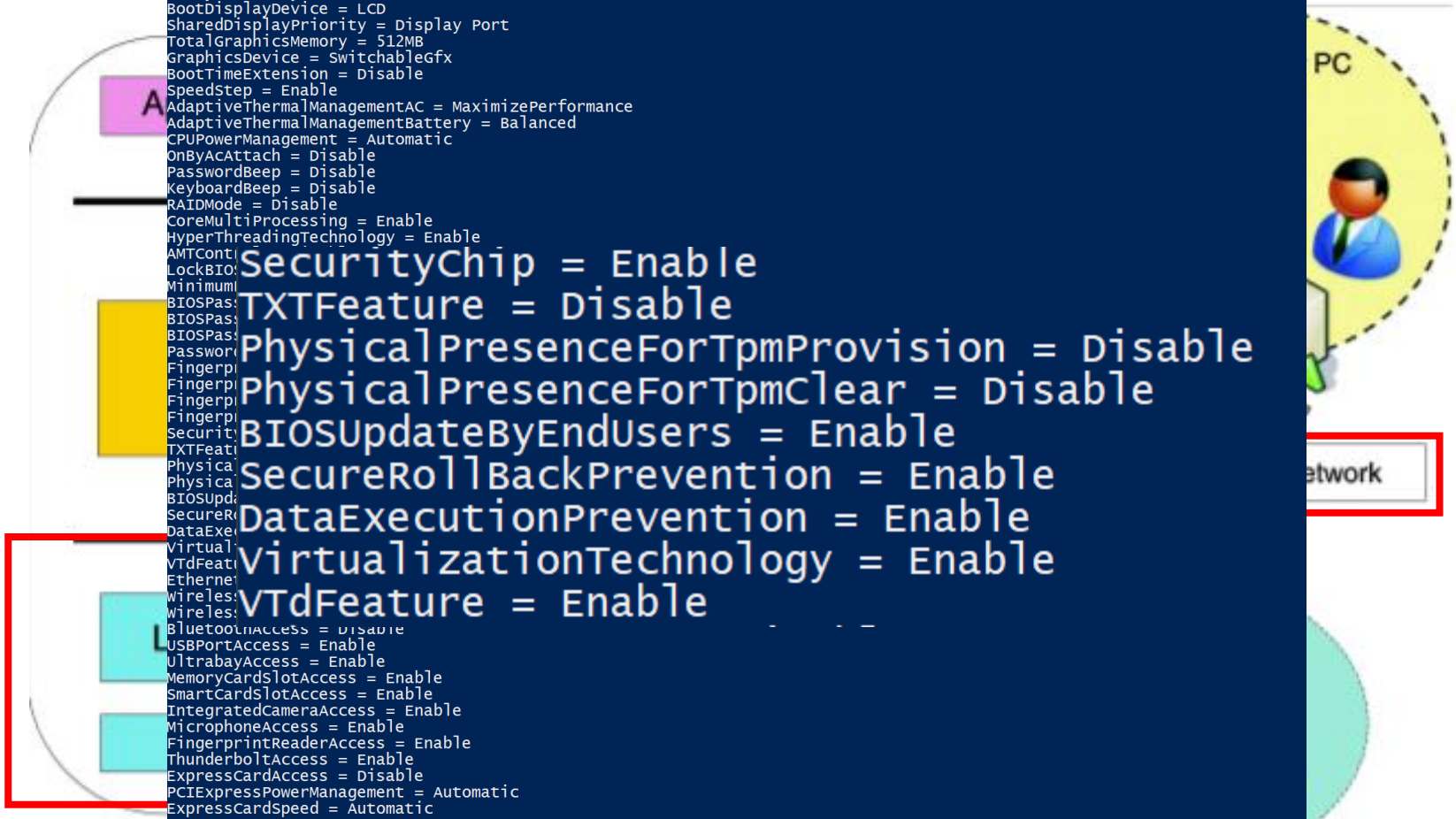


https://download.lenovo.co

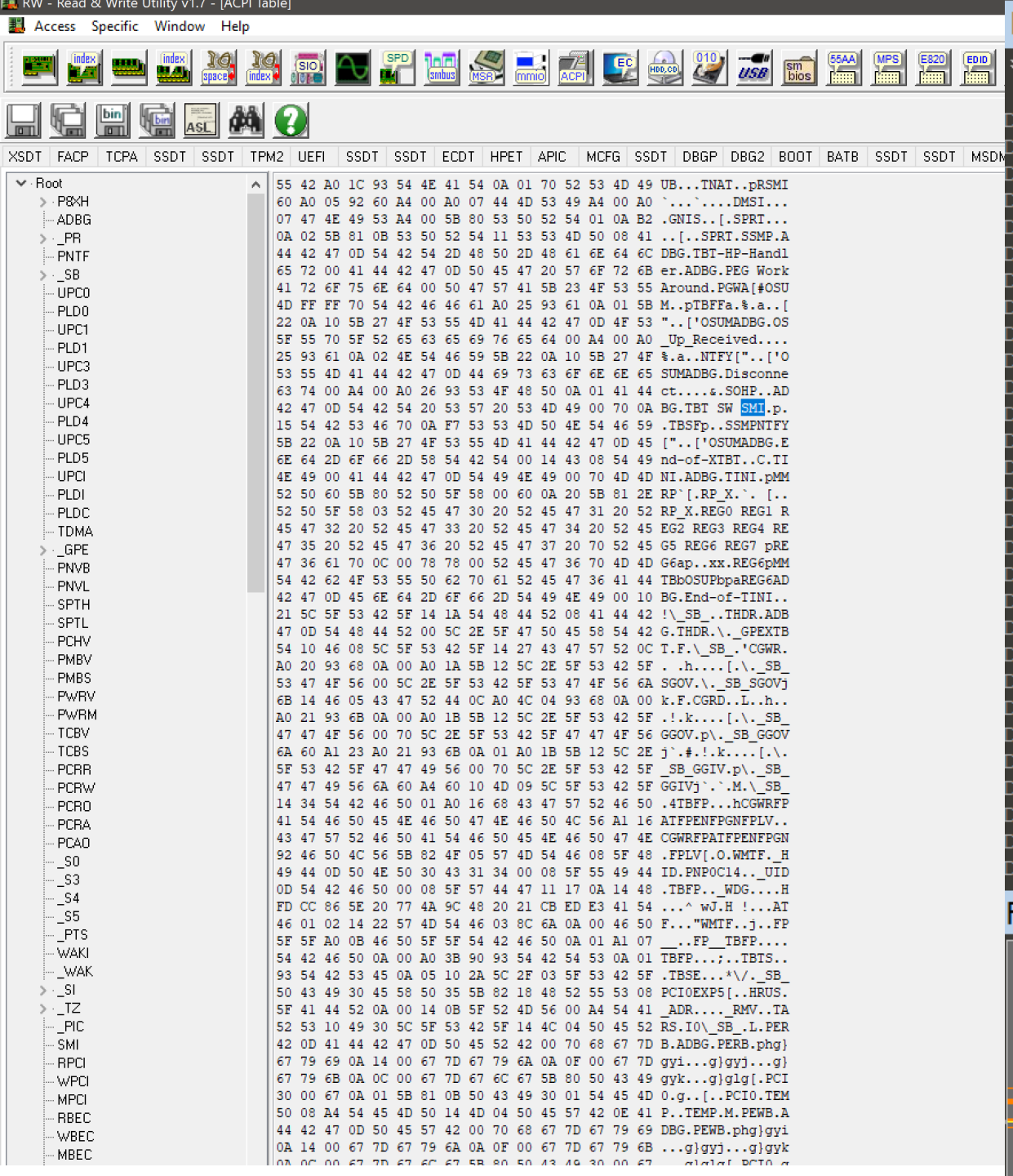
SMI over

```
PS C:\WINDOWS\system32> gwmi -class Lenovo_BiosSetting -namespace root\wmi | ForEach-Object {if ($_.CurrentSetting -ne
'') {Write-Host $_.CurrentSetting.replace(",","=")}}
WakeOnLAN = Disable
EthernetLANOptionROM = Enable
IPv4NetworkStack = Disable
IPv6NetworkStack = Disable
UefiPxeBootPriority = IPv4First
Reserved = Disable
USBBIOSSupport = Disable
AlwaysOnUSB = Disable
TrackPoint = Automatic
TouchPad = Automatic
FnCtrlKeySwap = Disable
FnSticky = Disable
FnKeyAsPrimary = Disable
BootDisplayDevice = LCD
SharedDisplayPriority = Display Port
TotalGraphicsMemory = 512MB
GraphicsDevice = SwitchableGfx
BootTimeExtension = Disable
SpeedStep = Enable
AdaptiveThermalManagementAC = MaximizePerformance
AdaptiveThermalManagementBattery = Balanced
CPUPowerManagement = Automatic
OnByAcAttach = Disable
PasswordBeep = Disable
KeyboardBeep = Disable
RAIDMode = Disable
CoreMultiProcessing = Enable
HyperThreadingTechnology = Enable
AMTCont
LockBIO
Minimum
BIOSPas
BIOSPas
BIOSPas
Passwor
Fingerp
Fingerp
Fingerp
Fingerp
Securit
TXTFeat
Physica
Physica
BIOSUpd
SecureR
DataExe
Virtual
VTdFeat
Ethernet
Wireless
Wireless
Bluetooth
USBPortAccess = Enable
UltrabayAccess = Enable
MemoryCardSlotAccess = Enable
SmartCardSlotAccess = Enable
IntegratedCameraAccess = Enable
MicrophoneAccess = Enable
FingerprintReaderAccess = Enable
ThunderboltAccess = Enable
ExpressCardAccess = Disable
PCIExpressPowerManagement = Automatic
ExpressCardSpeed = Automatic
RAIDStorage = SATAHDD
BottomCoverTamperDetected = Disable
InternalStorageTamper = Disable
ComputraceModuleActivation = Disable
SecureBoot = Enable
SGXControl = Enable
DeviceGuard = Disable
BootMode = Quick
StartupOptionKeys = Enable
BootDeviceListF12Option = Enable
BootOrder = USBCD:USBHDD:USBFDD:NVMe0:NVMe1:HDD0:HDD1:HDD2:HDD3:PCILAN
NetworkBoot = PCILAN
BootOrderLock = Enable
```

SecurityChip = Enable
TXTFeature = Disable
PhysicalPresenceForTpmProvision = Disable
PhysicalPresenceForTpmClear = Disable
BIOSUpdateByEndUsers = Enable
SecureRollBackPrevention = Enable
DataExecutionPrevention = Enable
VirtualizationTechnology = Enable
VTdFeature = Enable



https://download.lenovo.co



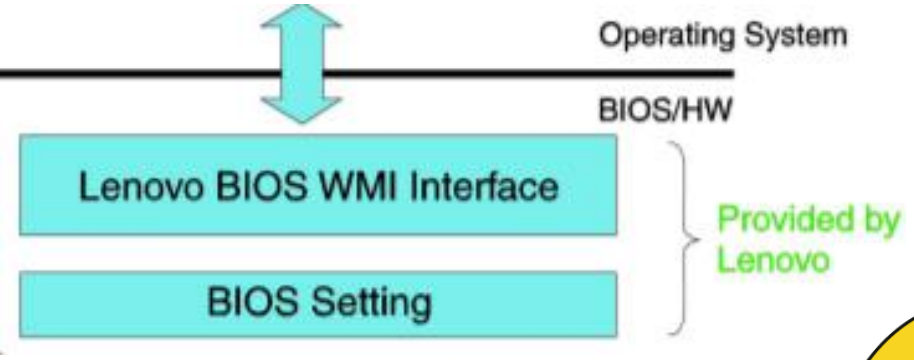
Find Results

Address	Value
D04Dh	WMI
D05Eh	WMI
D063h	WMI
D1CBh	WMI
D1D0h	WMI
D1DAh	WMI
D1ECh	WMI
D1F1h	WMI

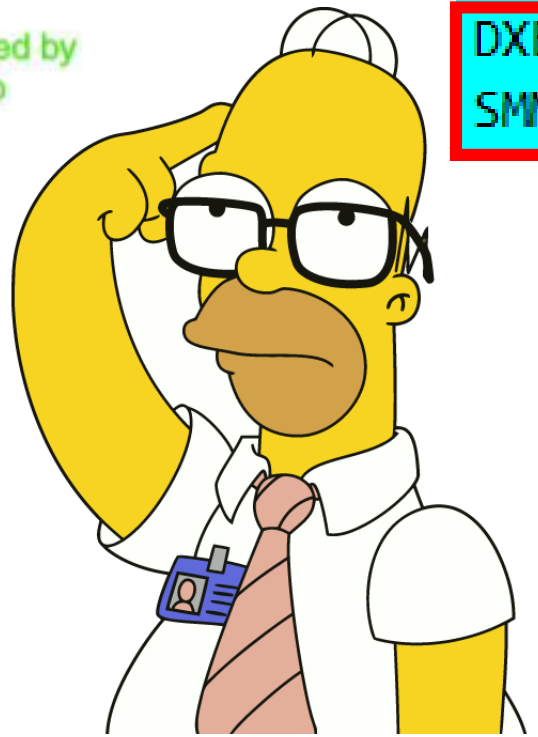
```
'
' Update Admnistrator Password
'
On Error Resume Next
Dim colItems

If WScript.Arguments.Count <> 3 Then
    WScript.Echo "SetSupervisorPassword.vbs [old Password] [new Password] [encoding]"
    WScript.Quit
End If

strRequest = "pap," + WScript.Arguments(0) + "," + WScript.Arguments(1) + "," + WScript.Arguments(2) +

strComputer = "LOCALHOST"      ' Change as needed.
Set objWMIService = GetObject("WinMgmts:" _
    &"{ImpersonationLevel=Impersonate}!\\" & strComputer & "\root\wmi")
Set colItems = objWMIService.ExecQuery("Select * from Lenovo_SetBiosPassword")

strReturn = "error"
For Each objItem in colItems
    ObjItem.SetBiosPassword strRequest, strReturn
Next

WScript.Echo " SetBiosPassword: "+ strReturn
```

BRAVE NEW WORLD

How this REsearch get started?



DXE driver	LenovoRemoteConfigUpdateDxe
SMM module	LenovoSetupAutomationSmm
DXE driver	LenovoSetupUnderOsDxe
SMM module	LenovoSetupUnderOsSmm



WTF LenovoSetupUnderOs (Smm/Dxe)?

- **LenovoSetupUnderOsDxe** (0D648466-36BD-42c6-B287-7C3BAA2575C0)
  - ✓ Communicate with LenovoPasswordManagerDxe
- **LenovoSetupUnderOsSmm** (65A72030-B02E-4bf3-8424-BA5F2FC56DE7)
  - Multiple WSMI Handlers (~12 SMI handlers):
    - ✓ Get/Set BiosPassword
    - ✓ Get/Set BiosSettings
- **LenovoHiddenSetting**
  - ✓ ComputraceDisable
  - ✓ CpuDebugEnable

Setup Automation SMI?

- ChangeConfiguration 0x04
- ChangePassword 0x81
- ChangeBootOrder 0xA7
- SecureBootConfiguration 0xAE

- It’s more: 0x0f, 0x80, 0x82, 0x9F, 0xB4/B6/B8

Setup Automation SMI?

- ChangeConfiguration 0x04
- ChangePassword 0x81
- ChangeBootOrder 0xA7
- SecureBootConfiguration 0xAE

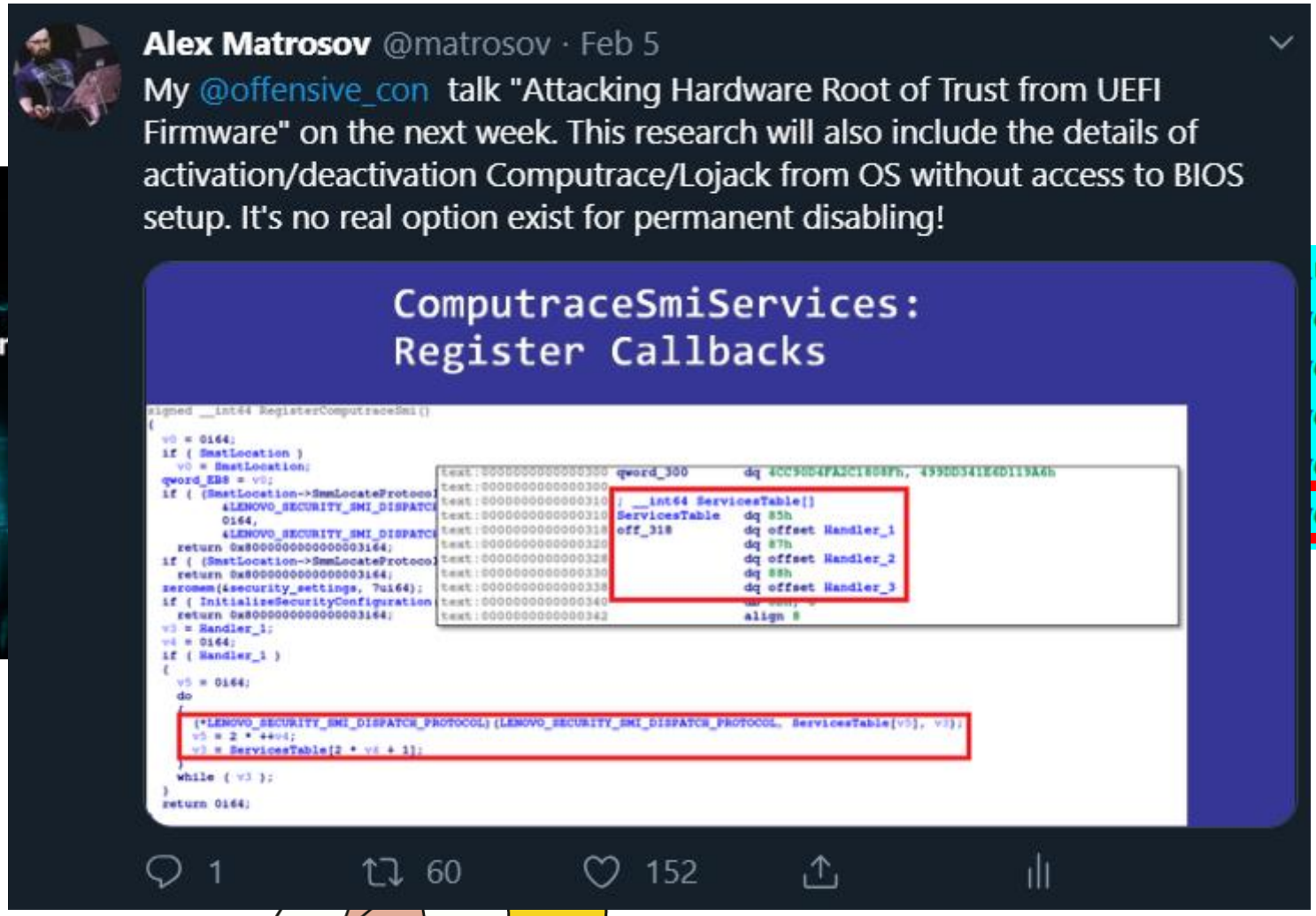
- It’s more: 0x0f, 0x80, 0x82, 0x9F, 0xB4/B6/B8

Computrace Never Dies

How I back to my old Computrace REsearch

Application	AbsoluteComputraceInstaller
DXE driver	LenovoComputraceEnablerDxe
DXE driver	LenovoComputraceLoaderDxe
SMM module	LenovoComputraceSmiServices
SMM module	LenovoSecuritySmiDispatch
DXE driver	LenovoRemoteConfigUpdateDxe

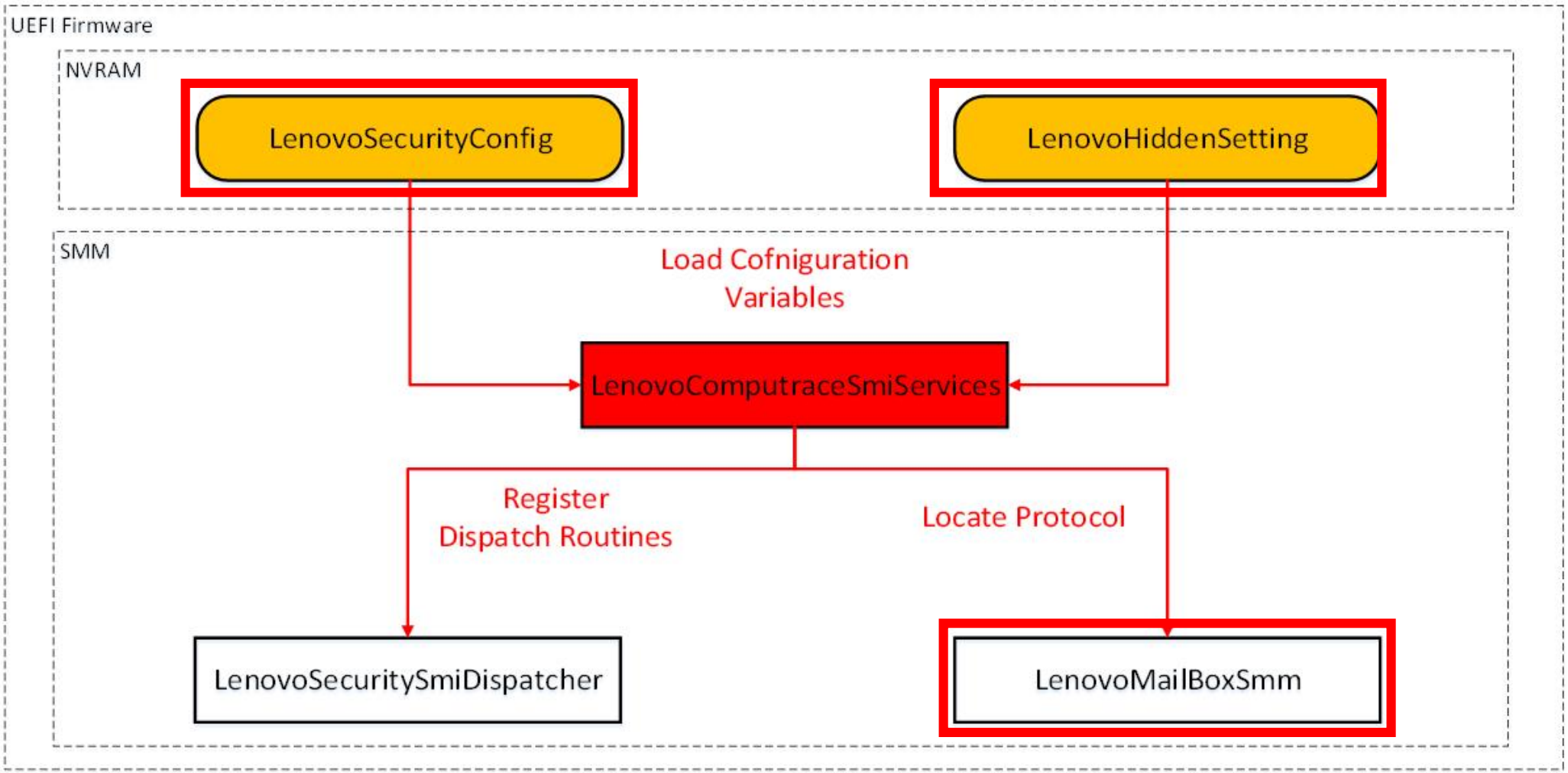
How I back to my old Computrace REsearch



uteComputraceInstaller
oComputraceEnablerDxe
oComputraceLoaderDxe
oComputraceSmiServices
oSecuritySmiDispatch
oRemoteConfigUpdateDxe

https://github.com/REhints/Publications/tree/master/Conferences/UEFI%20Firmware%20Rootkits%20Myths%20and%20Reality

ComputraceSmiServices: Components



https://github.com/REhints/Publications/tree/master/Conferences/UEFI%20Firmware%20Rootkits%20Myths%20and%20Reality

Lenovo security configs

```
typedef struct {

  UINT8   Unknown1[12]
  UINT8   IntelTXT;
  UINT8   Unknown2[5];
  UINT8   UserFlashUpdate;
  UINT8   Unknown3[15];
  UINT8   AccessToCamera;
  UINT8   AccessToMicrophone;
  UINT8   Unknown4[2];
  UINT8   Computrace;
  UINT8   Unknown5[2];
  UINT8   IntelVT;
  UINT8   IntelVTD;
  UINT8   Unknown6[2];
  UINT8   SecureBoot;
  UINT8   RollBackPrevention;
  UINT8   Unknown7[2];
  UINT8   IntelFTPM;
  UINT8   Unknown8;
  UINT8   PwdCountError;
  UINT8   Unknown9[6];
  UINT8   DeviceGuard;
  UINT8   Unknown10[80];

} LENOVO_SECURITY_CONFIG;
```

ComputraceSmiServices->Register Callbacks

```
signed __int64 RegisterComputraceSmi()
{
  v0 = 0i64;
  if ( SmstLocation )
    v0 = SmstLocation;
  qword_EB8 = v0;
  if ( (SmstLocation->SmmLocateProtocol(
          &LENOVO_SECURITY_SMI_DISPATCH_PROTOCOL_GUID,
          0i64,
          &LENOVO_SECURITY_SMI_DISPATCH_PROTOCOL) & 0x8000000000000000ui64) != 0i64 )
    return 0x8000000000000003i64;
  if ( (SmstLocation->SmmLocateProtocol(&LENOVO_MAILBOX_PROTOCOL_GUID, 0i64, &LENOVO_MAILBOX_PROTOCOL) & 0x8000000000000000ui64) != 0i64 )
    return 0x8000000000000003i64;
  zeromem(&security_settings, 7ui64);
  if ( InitializeSecurityConfiguration(v2) < 0 )
    return 0x8000000000000003i64;
  v3 = Handler_1;
  v4 = 0i64;
  if ( Handler_1 )
  {
    v5 = 0i64;
    do
    {
      (*LENOVO_SECURITY_SMI_DISPATCH_PROTOCOL)(LENOVO_SECURITY_SMI_DISPATCH_PROTOCOL, ServicesTable[v5], v3);
      v5 = 2 * ++v4;
      v3 = ServicesTable[2 * v4 + 1];
    }
    while ( v3 );
  }
  return 0i64;
}
```

https://github.com/REhints/Publications/tree/master/Conferences/UEFI%20Firmware%20Rootkits%20Myths%20and%20Reality

ComputraceSmiServices->Register Callbacks

```
signed __int64 RegisterComputraceSmi()
{
  v0 = 0i64;
  if ( SmstLocation )
    v0 = SmstLocation;
  qword_EB8 = v0;
  if ( (SmstLocation->SmmLocateProtocol
          &LENOVO_SECURITY_SMI_DISPATCH
          0i64,
          &LENOVO_SECURITY_SMI_DISPATCH
    return 0x8000000000000003i64;
  if ( (SmstLocation->SmmLocateProtocol
    return 0x8000000000000003i64;
  zeromem(&security_settings, 7ui64);
  if ( InitializeSecurityConfiguration(
    return 0x8000000000000003i64;
  v3 = Handler_1;
  v4 = 0i64;
  if ( Handler_1 )
  {
    v5 = 0i64;
    do
    {
      (*LENOVO_SECURITY_SMI_DISPATCH_PROTOCOL)(LENOVO_SECURITY_SMI_DISPATCH_PROTOCOL, ServicesTable[v5], v3);
      v5 = 2 * ++v4;
      v3 = ServicesTable[2 * v4 + 1];
    }
    while ( v3 );
  }
  return 0i64;
}
```

```
text:0000000000000300 qword_300       dq 4CC90D4FA2C1808Fh, 499DD341E6D119A6h
text:0000000000000300
text:0000000000000310
text:0000000000000310 ; __int64 ServicesTable[]
text:0000000000000310 ServicesTable   dq 85h
text:0000000000000318 off_318         dq offset Handler_1
text:0000000000000320                 dq 87h
text:0000000000000328                 dq offset Handler_2
text:0000000000000330                 dq 88h
text:0000000000000338                 dq offset Handler_3
text:0000000000000340                 db 0Bh, 0
text:0000000000000342                 align 8
```

https://github.com/REhints/Publications/tree/master/Conferences/UEFI%20Firmware%20Rootkits%20Myths%20and%20Reality

Computrace SMI Handlers

- ComputraceEnable = 0x85
- ComputraceDisable = 0x87
- ComputraceState = 0x88

- ComputraceEnableAction = 0x8d
- ComputraceDisableAction = 0x8e

ComputraceSmiServices->Register Callbacks

```
typedef struct {

  UINT8    Unknown1[4];
  UINT8    ComputraceState;
  UINT8    Unknown1[44];

} LENOVO_SCRATCH_DATA;
```

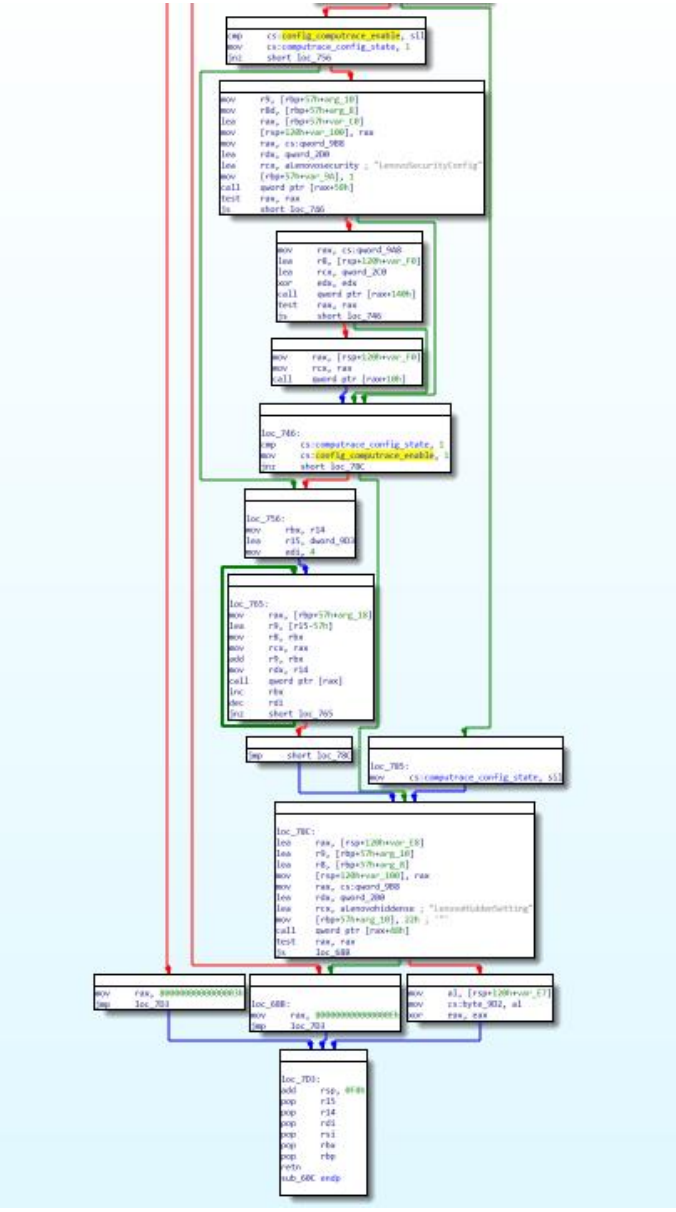
```
Variable NV+RT+BS '67C3208E-4FCB-498F-9729-0760BB4109A7:LenovoScratchData' DataSize = 30
  00000000: 00 00 00 00 00 00 00 00-00 00 00 00 01 00 00 01  *................*
  00000010: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000020: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
Variable NV+RT+BS '67C3208E-4FCB-498F-9729-0760BB4109A7:LenovoFlashScratch1' DataSize = 1
  00000000: 00                                               *.*
Variable NV+RT+BS '2A4DC6B7-41F5-45DD-B46F-2DD334C1CF65:LBL' DataSize = 1
  00000000: 00                                               *.*
Variable NV+RT+BS '67C3208E-4FCB-498F-9729-0760BB4109A7:MailBoxQ' DataSize = 4C
  00000000: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000010: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000020: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000030: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000040: 00 00 00 00 00 00 00 00-00 00 00 00              *............*
```

ComputraceSmiServices->Register Callbacks

```
typedef struct {

  UINT8   Unknown1[12]
  UINT8   IntelTXT;
  UINT8   Unknown2[5];
  UINT8   UserFlashUpdate;
  UINT8   Unknown3[15];
  UINT8   AccessToCamera;
  UINT8   AccessToMicrophone;
  UINT8   Unknown4[2];
  UINT8   Computrace;
  UINT8   Unknown5[2];
  UINT8   IntelVT;
  UINT8   IntelVTD;
  UINT8   Unknown6[2];
  UINT8   SecureBoot;
  UINT8   RollBackPrevention;
  UINT8   Unknown7[2];
  UINT8   IntelFTPM;
  UINT8   Unknown8;
  UINT8   PwdCountError;
  UINT8   Unknown9[6];
  UINT8   DeviceGuard;
  UINT8   Unknown10[80];

} LENOVO_SECURITY_CONFIG;
```

```
Variable NV+RT+BS 'A2C1808F-0D4F-4CC9-A619-D1E641D39D49:LenovoSecurityConfig' DataSize = 8B
  00000000: 01 00 00 00 01 01 00 00-01 00 01 00 00 00 00 00  *................*
  00000010: 00 00 00 00 01 01 01 01-01 01 01 01 01 01 01 01  *................*
  00000020: 01 01 01 01 01 00 00 01-00 01 00 00 01 00 00 00  *................*
  00000030: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000040: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000050: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000060: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000070: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000080: 00 00 00 00 00 00 00 00-00 00 01                 *...........*
```

ComputraceSmiServices->Register Callbacks



```
//0 = Disable 1 = Enable 2 = Permanent Disable
  if (SecurityConfig.Computrace == 1) {
    ComputraceState->Enable = TRUE;
  else
    ComputraceState->Enable = FALSE;
  }
```

SmiComputraceEnable = 0x85

```
#define COMPUTRACE_STATE_DISABLED       0x20000000
#define COMPUTRACE_STATE_ENABLED        0x40000000
#define COMPUTRACE_STATE_NOTSUPPORTED   0x80000000

  if (Computrace->State == FALSE) {
    reg_EAX |= COMPUTRACE_STATE_NOTSUPPORTED;
    return EFI_SUCCESS;
  }

  if (Computrace->State == TRUE) {
    reg_EAX |= COMPUTRACE_STATE_ENABLED;
    return EFI_SUCCESS;
  }
```

SmiComputraceDisable = 0x87

```
typedef struct _COMPUTRACE_STATE {
  BOOLEAN   Enabled;
  BOOLEAN   Active;
  BOOLEAN   Disabled;
  UINT8     DisableSecretKey[4];
} COMPUTRACE_STATE;
```

```
key_byte = cpu_regs->EBX;

ComputraceState.Active          = TRUE;
ComputraceState.DisableSecretKey[0] = key_byte & 0xff;
ComputraceState.DisableSecretKey[1] = (key_byte & 0xff00) >> 8;
ComputraceState.DisableSecretKey[2] = (key_byte & 0xff0000) >> 16;
ComputraceState.DisableSecretKey[3] = (key_byte & 0xff000000) >> 24;
```

SmiComputraceDisable = 0x87

```
key_match = TRUE;
for (i = 0; i < 4; i++) {
  if (Key[i] != ComputraceState.DisableKey[i]) {
    key_match = FALSE;
    break;
  }
}
if (key_match == FALSE) {

  DisableRetryCount++;

  cpu_regs->EAX |= COMPUTRACE_WRONG_KEY;
  return EFI_SUCCESS;
}
```

<- not constant time

Brutforce Lenovo Computrace Disable Key

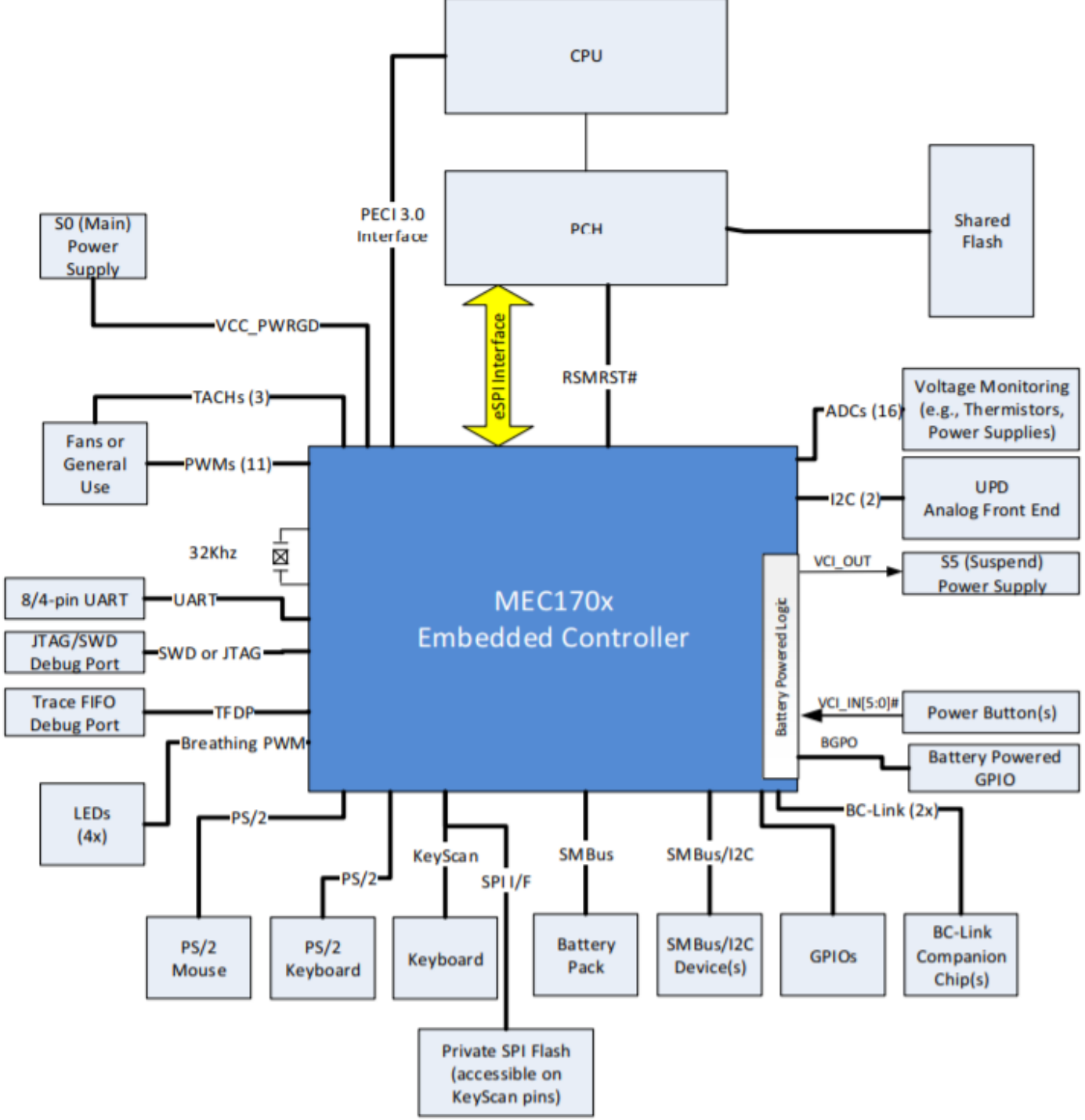
- **Computrace Disable Secret Key**
  - ✓ 1 BYTE secret value ☺ stored in SPI flash (NVRAM)
  - ✓ Can be different by laptop model line
    (my sweet victims p50 and t540p has a different keys)

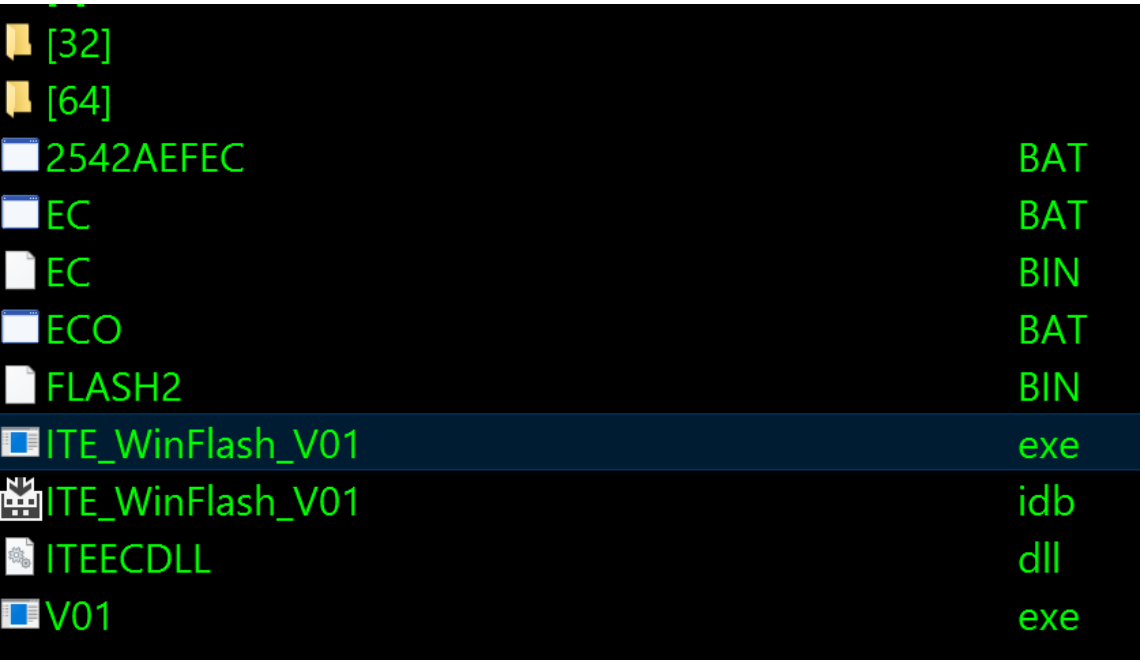
for i in range(0,256):
chipsec_util smi 0x0 0x85 0x0 hex(i)

Fuzz->Check->Repeat->Profit!

DisableSecretKey == 0x57 o_O

Embedded Controller is not a security boundary


CPU
PCH
Shared Flash
PECI 3.0 Interface
eSPI Interface
RSMRST#
S0 (Main) Power Supply
VCC_PWRGD
TACHs (3)
Fans or General Use
PWMs (11)
32Khz
8/4-pin UART
UART
JTAG/SWD Debug Port
SWD or JTAG
Trace FIFO Debug Port
TFDP
Breathing PWM
LEDs (4x)
MEC170x
Embedded Controller
Battery Powered Logic
ADCs (16)
Voltage Monitoring (e.g., Thermistors, Power Supplies)
I2C (2)
UPD Analog Front End
VCI_OUT
S5 (Suspend) Power Supply
VCI_IN[5:0]#
Power Button(s)
BGPO
Battery Powered GPIO
BC-Link (2x)
PS/2
PS/2
KeyScan
SPI I/F
SMBus
SMBus/I2C
PS/2 Mouse
PS/2 Keyboard
Keyboard
Battery Pack
SMBus/I2C Device(s)
GPIOs
BC-Link Companion Chip(s)
Private SPI Flash (accessible on KeyScan pins)




```
  if ( dword_410FEC )
  {
    write_port(dword_410FF4);
    printf("Send Erase Command...\n");
  }
  Sleep(0x64u);
  printf("Erase Done\n");
  if ( sub_401170() )
  {
    printf("Return from Erase Checking: Done\n");
    if ( !dword_410FEC )
    {
      printf("Send Erase Command Again\n");
      write_port(dword_410FF4);
      Sleep(0x64u);
    }
    dword_410FE4 = 0;
    while ( !sub_401220() )
    {
      printf("Programming the EC Firmware now.....\n");
      ++dword_410FE4;
      read_port();
      read_port();
      write_port(dword_410FF4);
      Sleep(0x64u);
    }
    printf("The EC Firmware Programmed Done & Verification Success.\n");
    ++dword_410FF4;
  }
  else
  {
    printf("Return from CheckDataFF: false\n");
    ++dword_410FF4;
  }
}
```

😈 More EC fun coming this summer 😈

Stay tuned!!

Summary:

- The usability in enterprise world in many cases the main enemy of security
- The vendors understand “Permanent Disable” option differently
- When Hardware-based Root of Trust transfer the state of Chain of Trust to software, it’s not hardware anymore

Thank you for your attention!

@matrosov