BETRAYING THE BIOS:

WHERE THE GUARDIANS OF THE BIOS ARE FAILING

Alex Matrosov

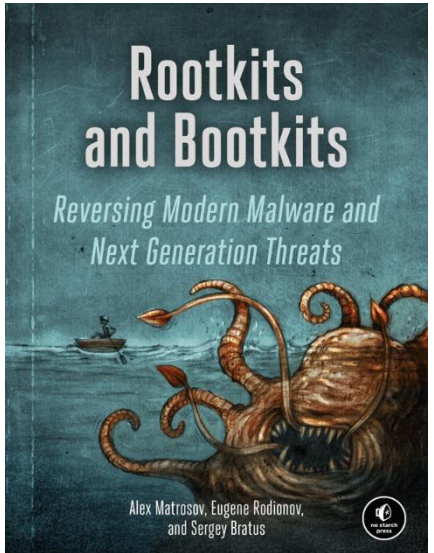
@matrosov

Have a lot of fun with UEFI Security and RE

Former Security Researcher @Intel

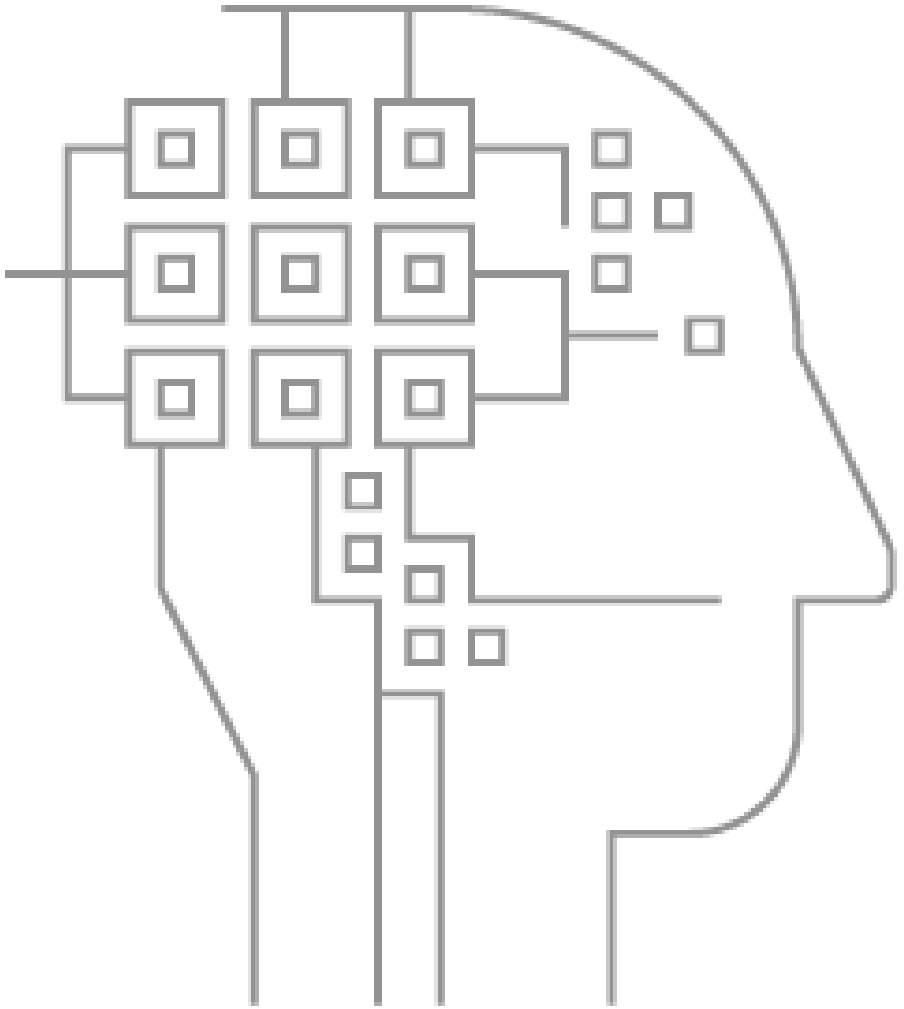
Reverse Engineering since 1997

Book co-author nostarch.com/rootkits

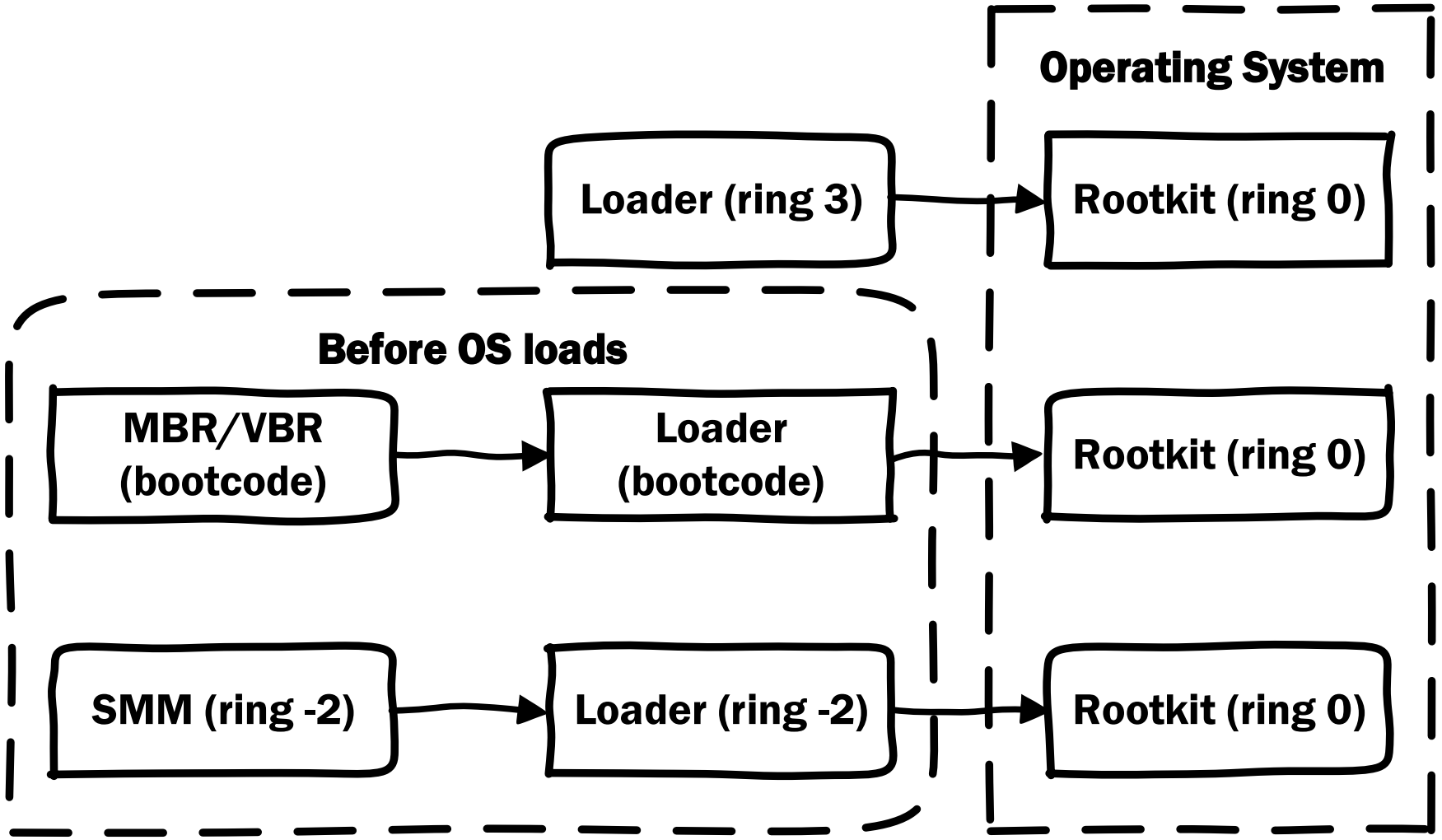


@matrosov

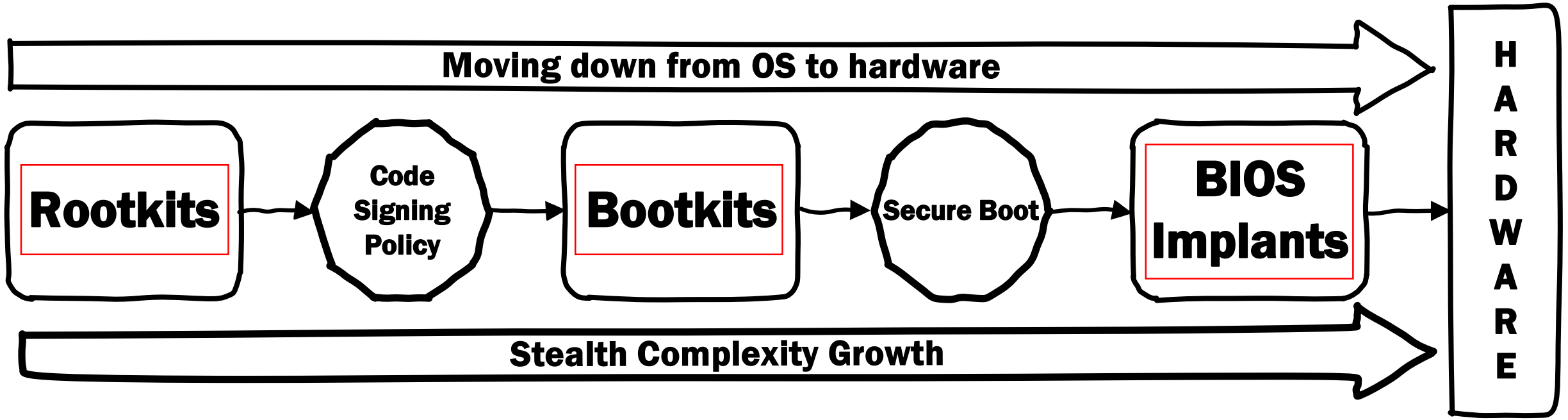
- **Intro**
- **Attacks on BIOS Updates**
  - ✓ Unsigned Updates
  - ✓ BIOS protection bits
  - ✓ SmiFlash and SecSmiFlash
- **Intel Boot Guard**
  - ✓ AMI implementation details
  - ✓ Discover ACM secrets
  - ✓ Vulns
  - ✓ Boot Guard Bypass!
- **Intel BIOS Guard**
  - ✓ AMI implementation details



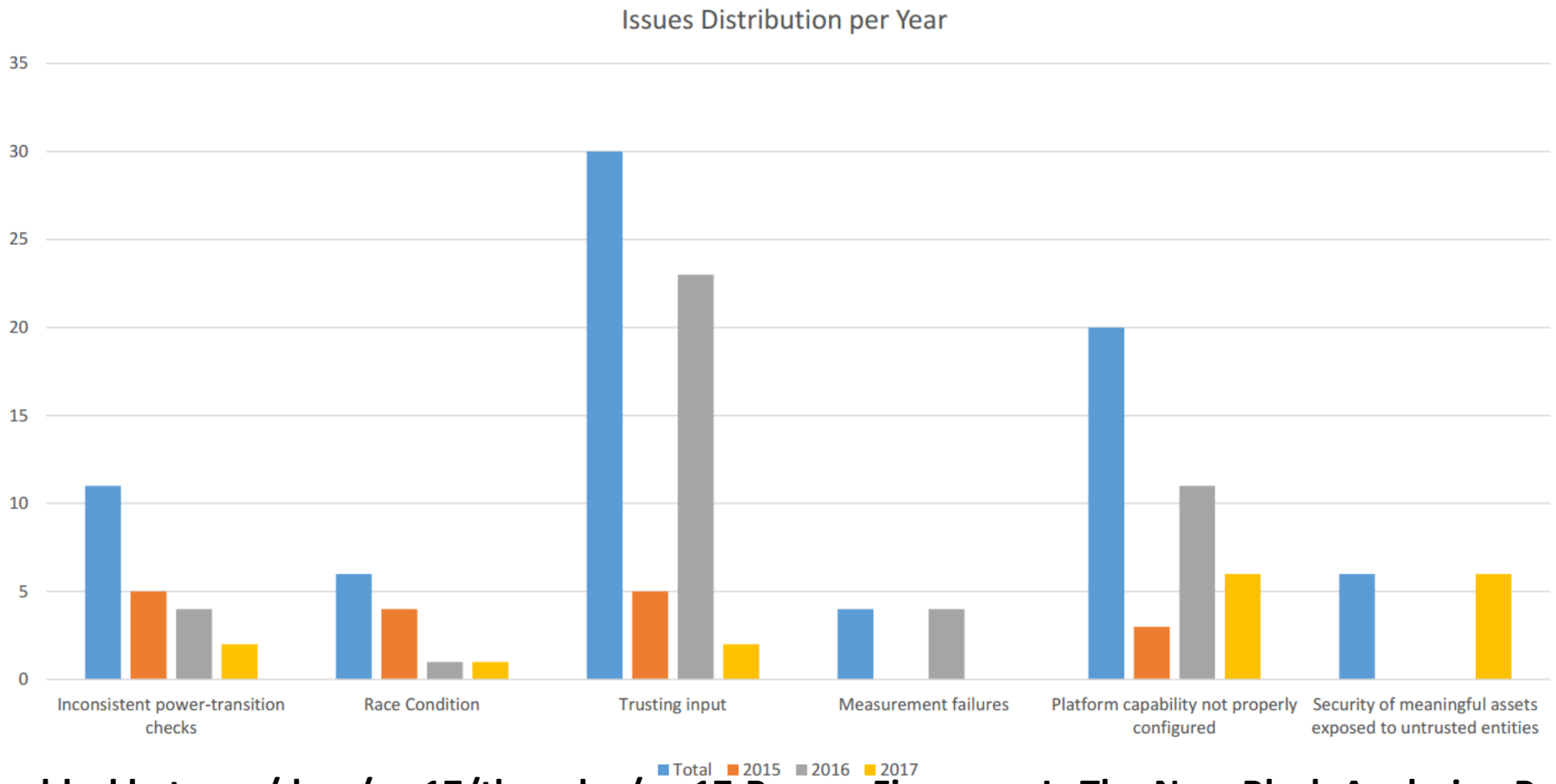
All rootkits want to get into Ring 0



More mitigations, more rootkits complexity

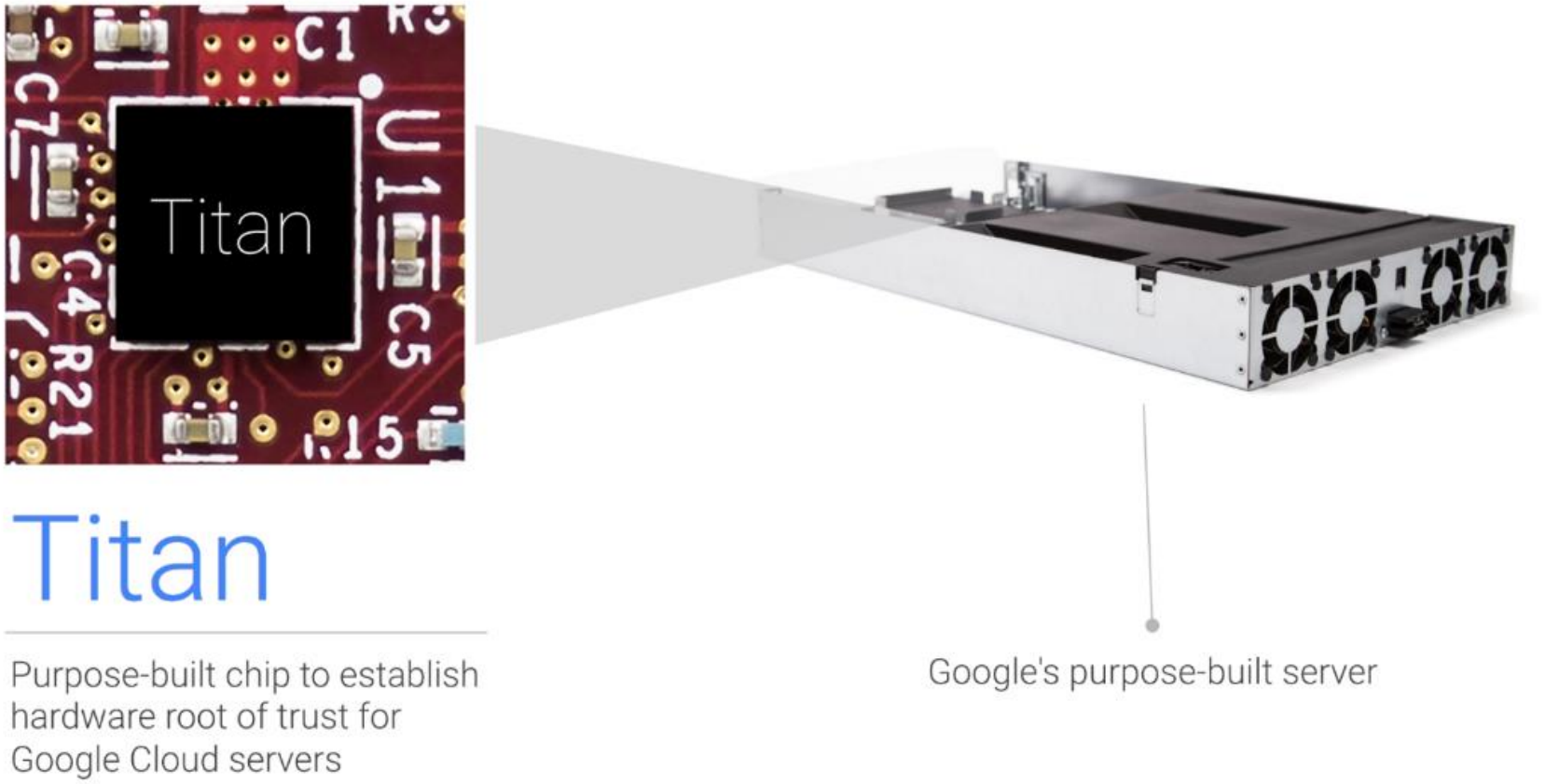


Growths of configuration based vulnerabilities



https://www.blackhat.com/docs/us-17/thursday/us-17-Branco-Firmware-Is-The-New-Black-Analyzing-Past-Three-Years-Of-BIOS-UEFI-Security-Vulnerabilities.pdf

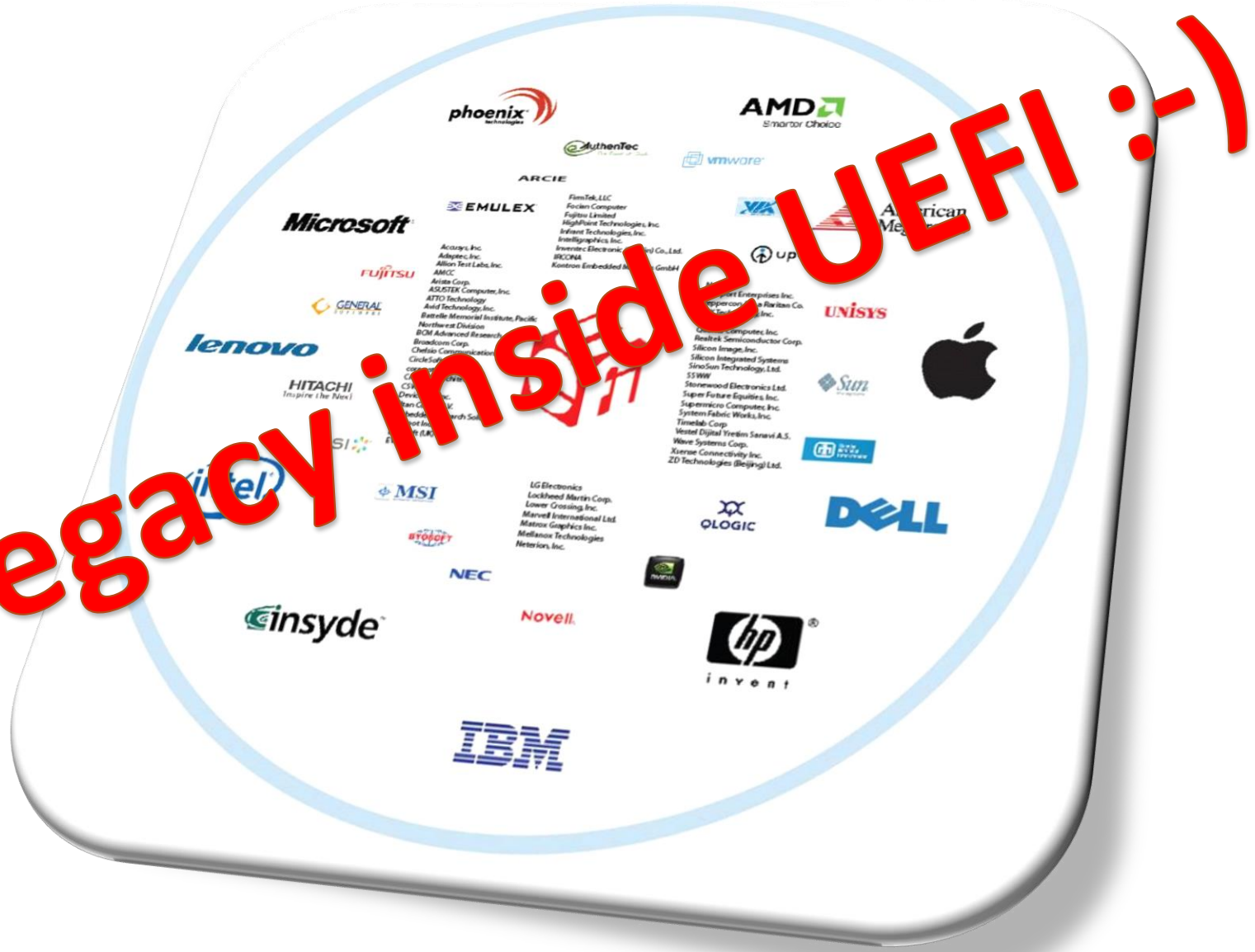
Google Titan Chip



https://cloudplatform.googleblog.com/2017/08/Titan-in-depth-security-in-plaintext.html

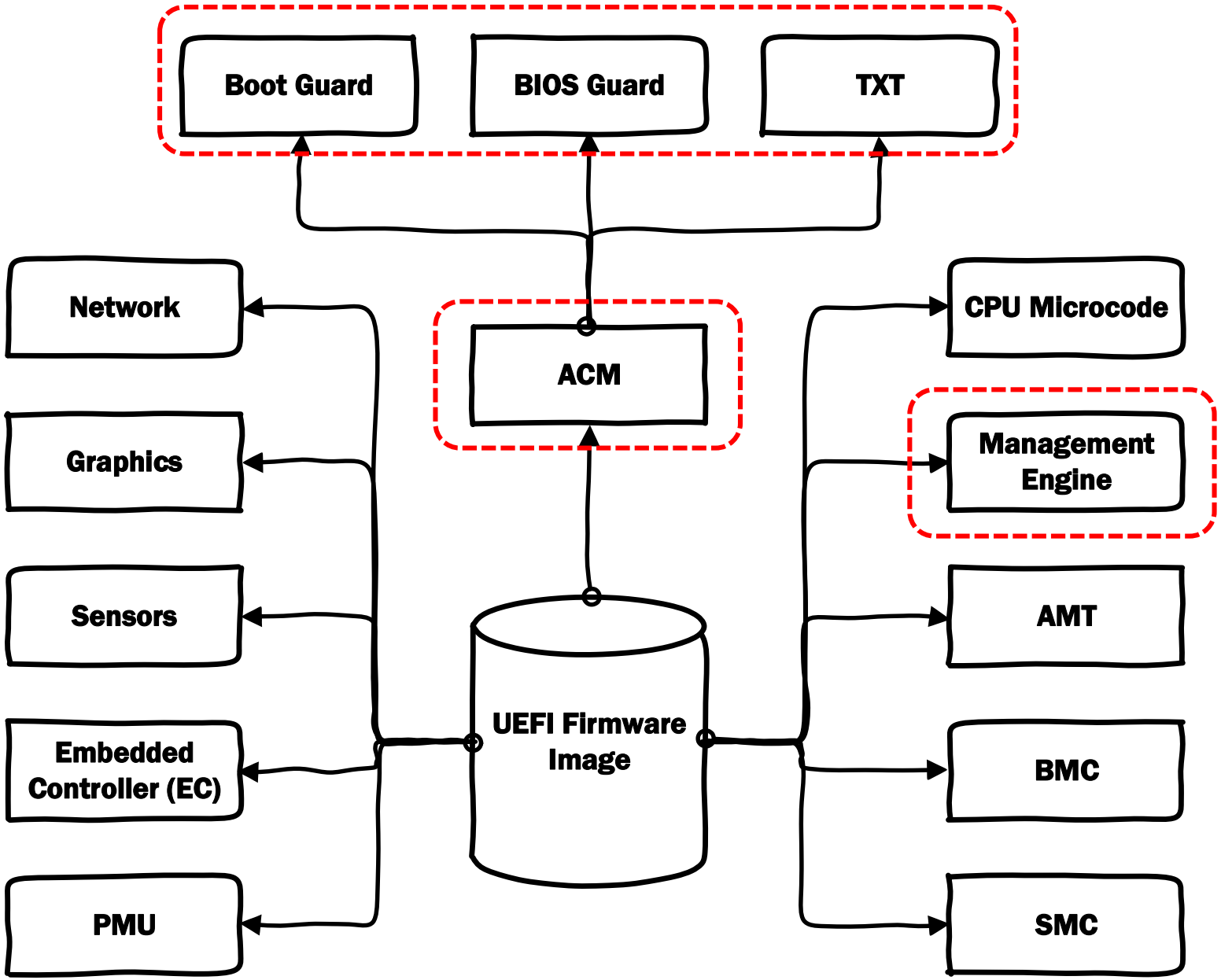
BIOS Update Issues

No more legacy! UEFI is everywhere!!



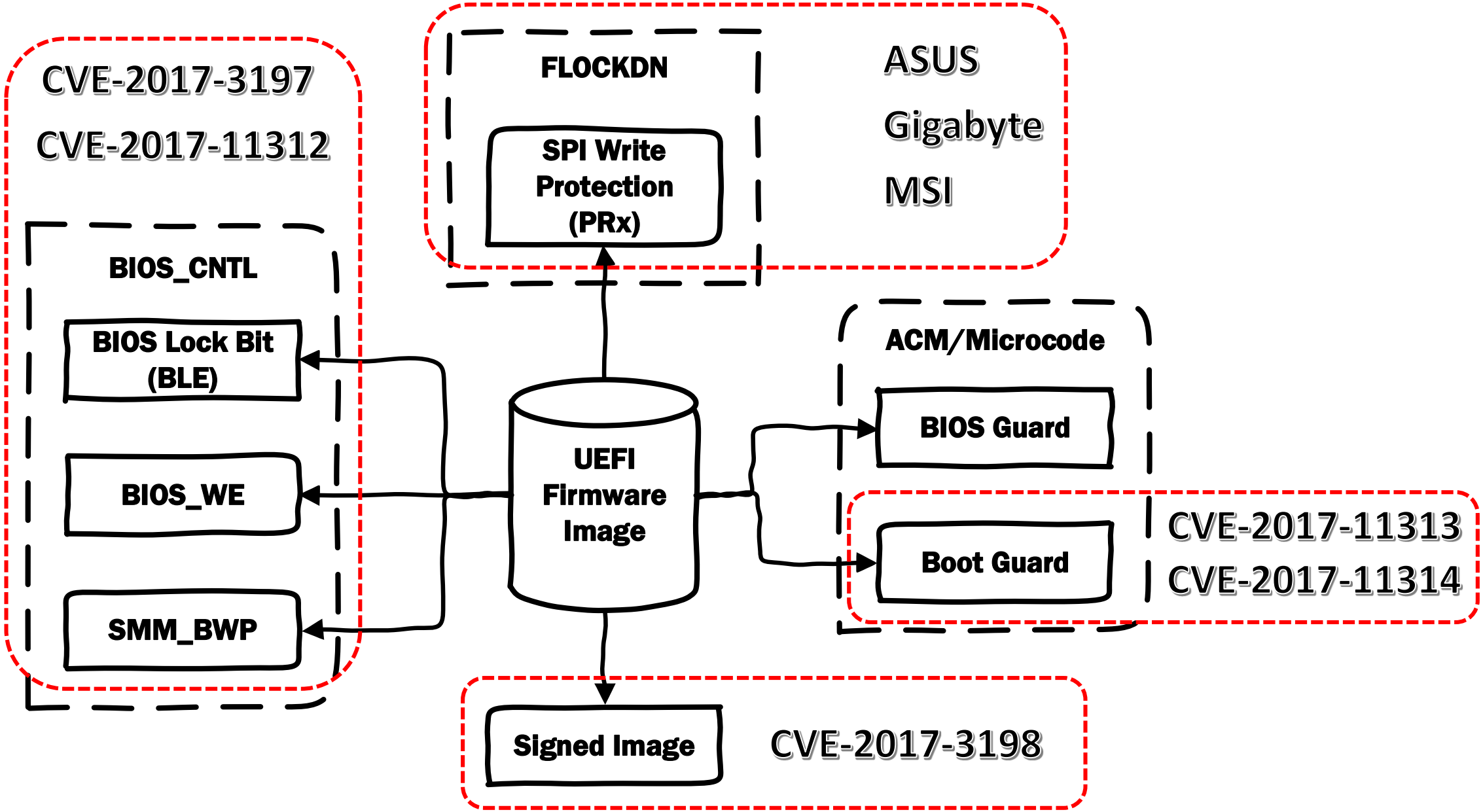
Now the legacy inside UEFI :-)

How many different firmware's inside BIOS update?



All the vulnerabilities mention in this research found inside AMI-based UEFI firmware’s

All Guardians of the BIOS on one slide



How different vendors care about security?

Vendor Name	BLE	SMM_BWP	PRx	Authenticated Update
ASUS	+	+	-	-
MSI	-	-	-	-
Gigabyte	+	+	-	-
Dell	+	+	-+	+
Lenovo	+	+	RP	+
HP	+	+	RP/WP	+
Intel	+	+	-	+
Apple	-	-	WP	+

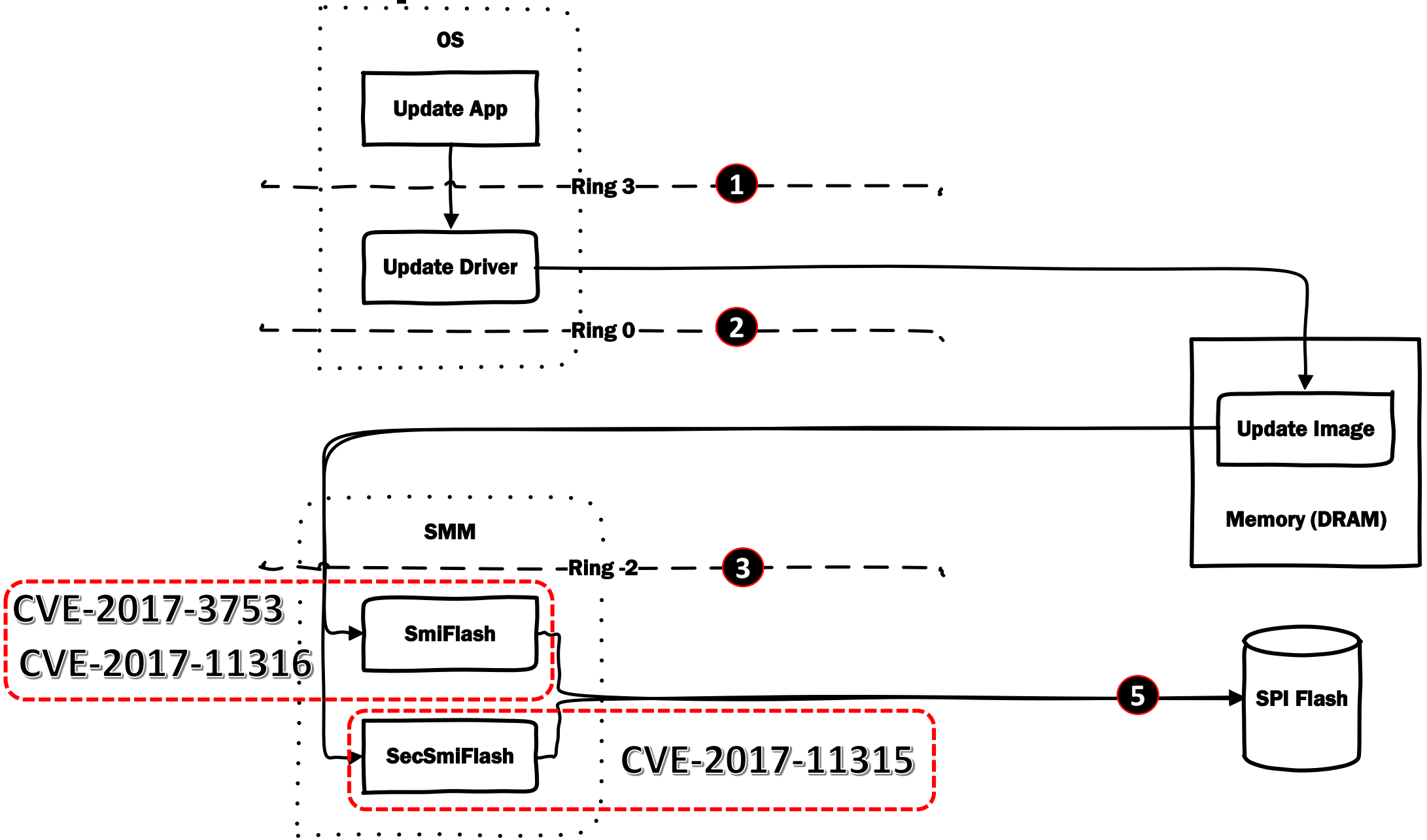
```
[x][ ================================================================
[x][ Module: BIOS Interface Lock (including Top Swap Mode)
[x][ ================================================================
[*] BiosInterfaceLockDown (BILD) control = 1
[*] BIOS Top Swap mode is disabled (TSS = 0)
[*] RTC TopSwap control (TS) = 0
[+] PASSED: BIOS Interface is locked (including Top Swap Mode)

[*] running module: chipsec.modules.common.bios_wp
[*] Module path: c:\Chipsec\chipsec\modules\common\bios_wp.pyc
[x][ ================================================================
[x][ Module: BIOS Region Write Protection
[x][ ================================================================
[*] BC = 0x08 << BIOS Control (b:d.f 00:31.0 + 0xDC)
    [00] BIOSWE           = 0 << BIOS Write Enable
    [01] BLE              = 0 << BIOS Lock Enable
    [02] SRC              = 2 << SPI Read Configuration
    [04] TSS              = 0 << Top Swap Status
    [05] SMM_BWP          = 0 << SMM BIOS Write Protection
[-] BIOS region write protection is disabled!

[*] BIOS Region: Base = 0x00A00000, Limit = 0x00FFFFFF
SPI Protected Ranges
------------------------------------------------------------
PRx (offset) | Value    | Base     | Limit    | WP? | RP?

PR0 (74)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR1 (78)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR2 (7C)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR3 (80)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR4 (84)     | 00000000 | 00000000 | 00000000 | 0   | 0

[!] None of the SPI protected ranges write-protect BIOS region
```

blackhat.com/docs/asia-17/materials/asia-17-Matrosov-The-UEFI-Firmware-Rootkits-Myths-And-Reality.pdf

Why so vulnerable?

- **BIOS LOCK (BLE) not enabled (CLVA-2016-12-001/CVE-2017-3197)**
  - ✓ Attacker is able to modify BIOSWE bit
  - ✓ Attacker can arbitrary write to SPI flash from OS
- **FW update process don't verify signature**
  - ✓ Attacker is able to abuse BIOS updater with signed driver
- **SmiFlash Handler multiple vulns (CLVA-2016-12-002/CVE-2017-3198)**
  - ✓ Attacker can elevate privileges to SMM (ring -2)

How BIOS Update Guardians Fail?



SMIFlash Handler Issues: Gigabyte, Lenovo, MSI

- SmiFlash HANDLERS (SMiFlash.efi)→ CVE-2017-3753, CVE-2017-11316

  [BC327DBD-B982-4f55-9F79-056AD7E987C5]

  - ✓ ENABLE 0x20
  - ✓ READ 0x21
  - ✓ ERASE 0x22
  - ✓ WRITE 0x23
  - ✓ DISABLE 0x24
  - ✓ GET_INFO 0x25

- No checks for the input pointers

  SmmIsBufferOutsideSmmValid()

SecSMIFlash Handler Issues: ASUS

- SecSmiFlash HANDLERS (SecSMiFlash.efi) → CVE-2017-11315
  [3370A4BD-8C23-4565-A2A2-065FEEDE6080]
  - ✓ LOAD_IMAGE 0x1d
  - ✓ GET_POLICY 0x1e
  - ✓ SET_POLICY 0x1f

- No checks for the input pointers
  SmmIsBufferOutsideSmmValid()

That's why BIOS Guard created

Responsible Disclosure Fun

- ✓ Discovery Date: 2017-04-20
- ✓ Intel PSIRT Notified: 2017-05-22
- ✓ All the Vendors Notified: 2017-05-26
- ✓ Disclosure Notification Date: 2017-05-30
- ✓ Lenovo Released a Patch: 2017-07-11
- ✓ ASUS Released a Patch: 2017-06-23
- ✓ MITRE Assign 6 CVE’s: 2017-07-13
- ✓ Gigabyte Released a Patch: 2017-07-25
- ✓ Public Disclosure Date: 2017-07-27

https://github.com/CylanceVulnResearch/disclosures

ASUS Responsible Disclosure Fun

Alex Matrosov
@matrosov

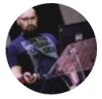
Bravo @ASUS! You silently patch 3 of my

Dear sender,

Thank you for the e-mail.
Please don't get us wrong, all of your findings are valuable and we deeply appreciate for the kindness sharing.

We would mention "Fixed UEFI and SMI vulnerability. Special thanks for Cylance" in the update BIOS, or it can be discussed if you have ideas of wording in mind.
Thank you

Best regards,
ASUS Security | (c)ASUSTeK Computer Inc.



Alex Matrosov @matrosov · Jul 14

Replying to @matrosov @ASUS

Finally ASUS agreed they patched my bugs. Good to know but I'm already confirmed this with simple check by BinDiff for patched SMM driver ;)

Intel Boot Guard

Different shades of Secure Boot

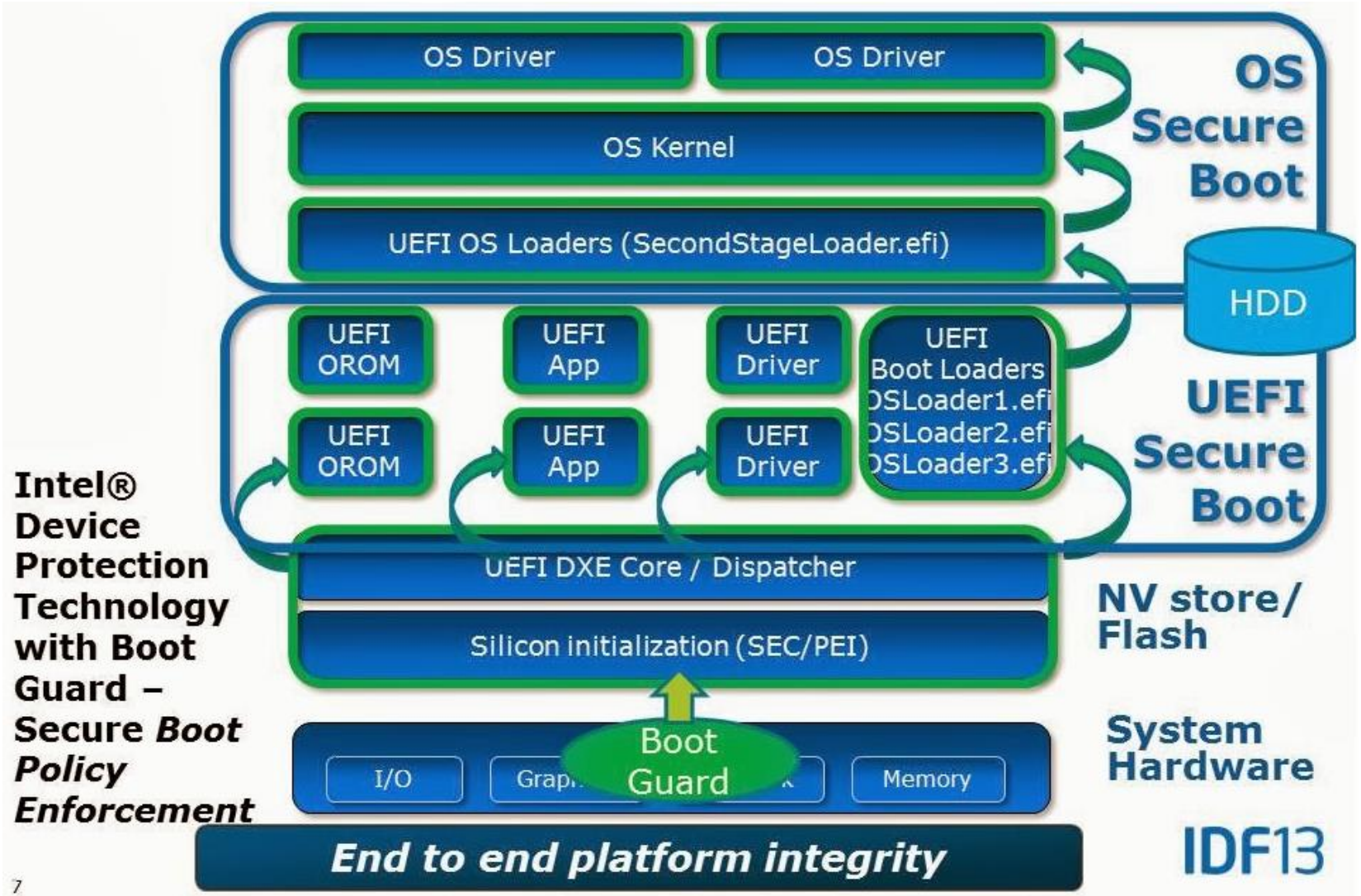
- **Secure Boot -> since 2012**
  - ✓ Root of Trust = Firmware -> BIOS
  - ✓ Attack Surface = Firmware
- **Measured Boot (Boot Guard) -> since 2013**
  - ✓ Root of Trust = Hardware -> Trusted Platform Module (TPM)
  - ✓ Attack Surface = Firmware
- **Verified Boot (Boot Guard) -> since 2013**
  - ✓ Root of Trust = Hardware -> Field Programming Fuse (FPF)->Locked
  - ✓ Attack Surface = Firmware + Hardware

First bypass today?!

Why Boot Guard has been created?

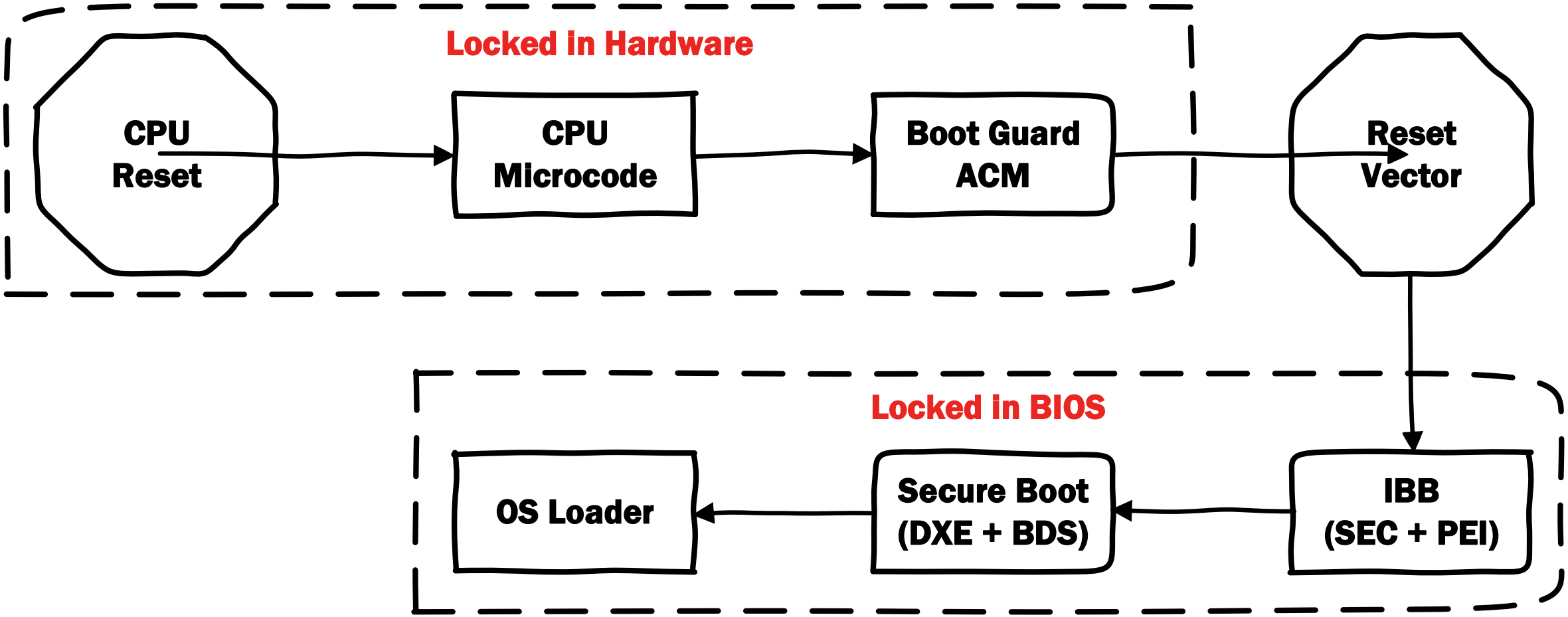
- **Secure Boot** starts from DXE phase and impacted with any SMM issues/implants
- No verification on early boot for SEC/PEI boot phases
- **Measured Boot** starts before PEI phase but also impacted with any SMM issues/implants
- The Root of Trust must be locked by hardware (**Verified Boot**)
- The first step of verification should rely on microcode authentication

Intel Boot Guard Technology



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

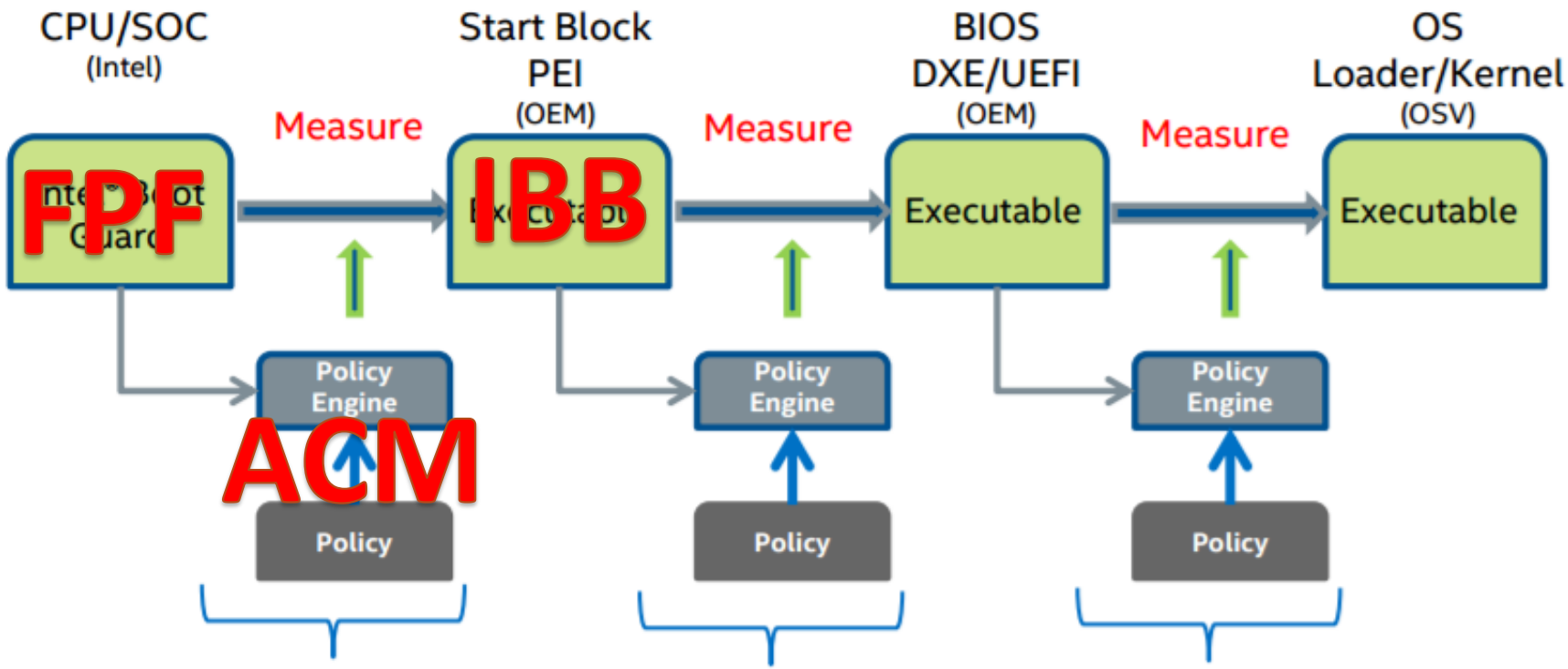
Boot Guard: Boot Flow



Intel Boot Guard operating modes

- Not Enabled
- Measured Boot (root of trust = TPM)
- Verified Boot (root of trust = FPF)
- Measured + Verified Boot (root of trust = FPF + TPM)

Demystifying Intel Boot Guard



Intel® Device Protection Technology with Boot Guard

http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/4th-gen-core-family-mobile-brief.pdf

OEM PI Verification Using PI Signed Firmware Volumes

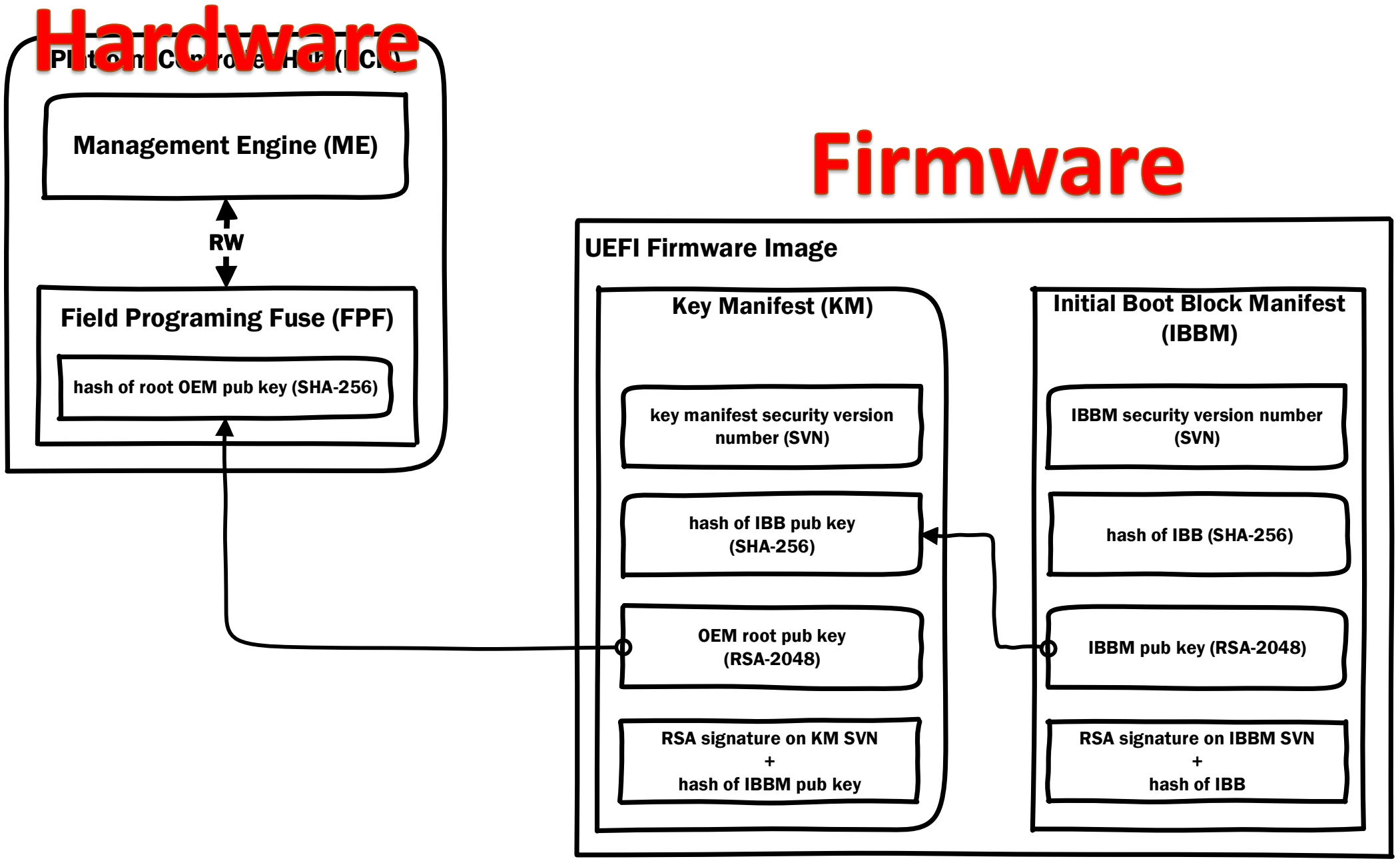
Vol 3, section 3.2.1.1 of PI 1.3 Specification

OEM UEFI 2.4 Secure Boot

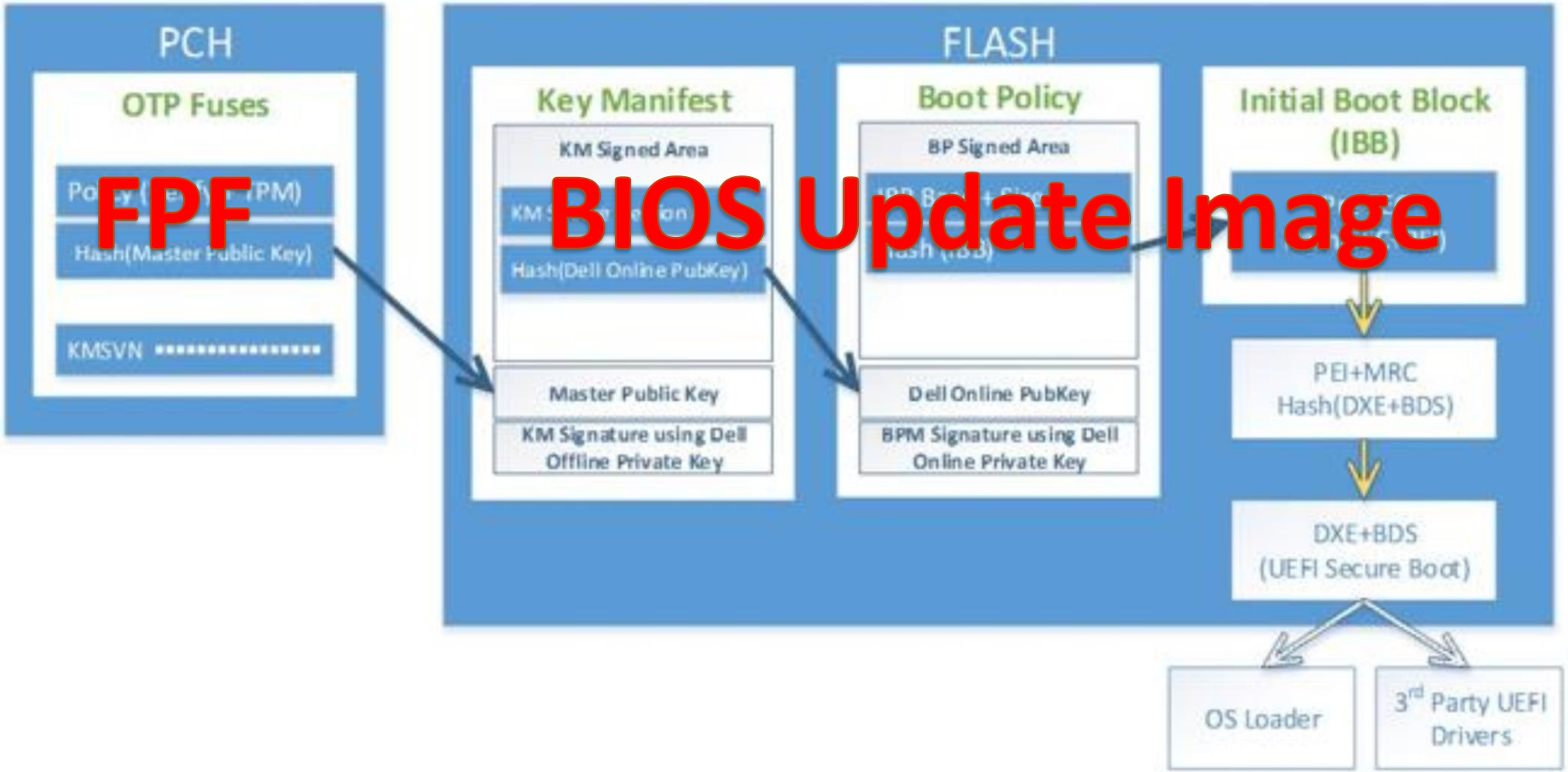
Chapter 27.2 of The UEFI 2.4 Specification

https://firmware.intel.com/sites/default/files/STTS003%20-%20SF15_STTS003_100f.pdf

Boot Guard: Chain of Trust



Demystifying Intel Boot Guard



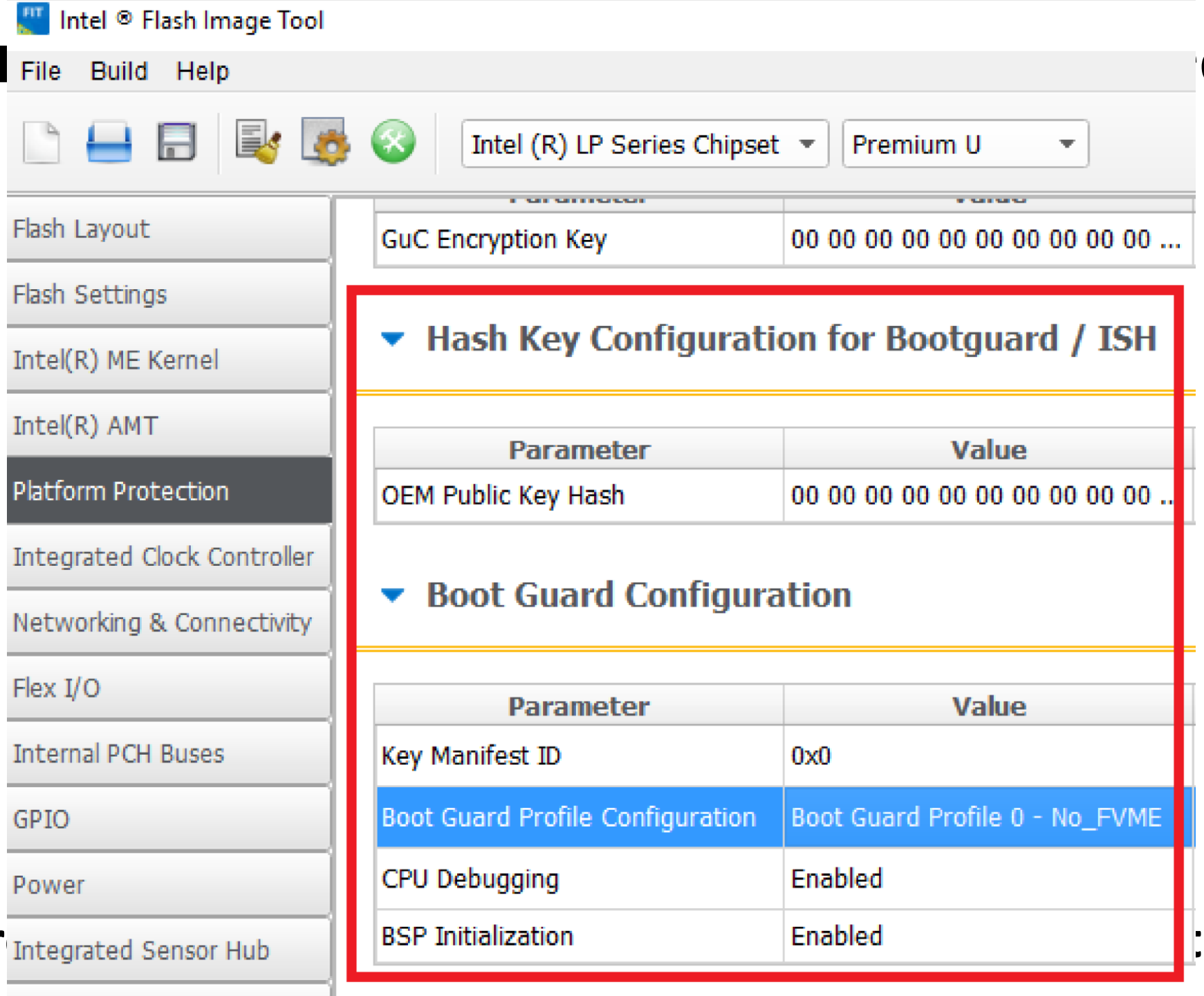
http://en.community.dell.com/techcenter/extras/m/white_papers/20444061

Guard's Configuration of Tested Hardware

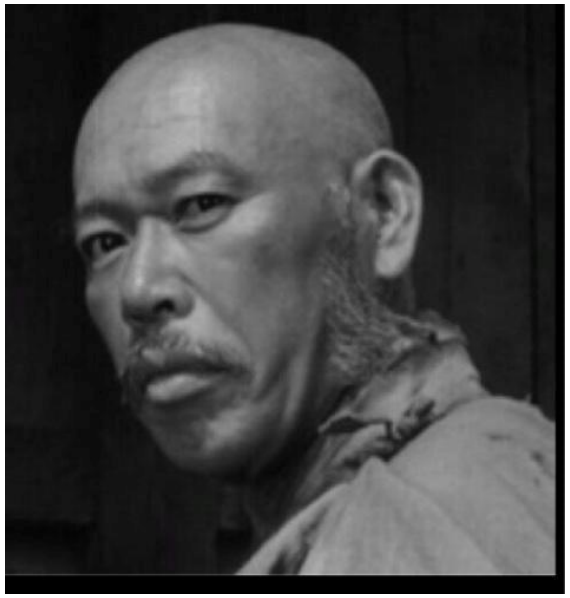
Vendor Name	ME Access	EC Access	CPU Debugging (DCI)	Boot Guard	Forced Boot Guard ACM	Boot Guard FPF	BIOS Guard
ASUS VivoMini	Disabled	Disabled	Enabled	Disabled	Disabled	Disabled	Disabled
MSI Cubi2	Disabled	Disabled	Enabled	Disabled	Disabled	Disabled	Disabled
Gigabyte Brix	Read/Write Enabled	Read/Write Enabled	Enabled	Measured Verified	Enabled (FPF not set)	Not Set	Disabled
Dell	Disabled	Disabled	Enabled	Measured Verified	Enabled	Enabled	Enabled
Lenovo ThinkCentre	Disabled	Disabled	Enabled	Disabled	Disabled	Disabled	Disabled
HP Elitedesk	Disabled	Disabled	Enabled	Disabled	Disabled	Disabled	Disabled
Intel NUC	Disabled	Disabled	Enabled	Disabled	Disabled	Disabled	Disabled
Apple	Read Enabled	Disabled	Disabled	Not Supported	Not Supported	Not Supported	Not Supported


TRUST
NO
ONE

Safegu d



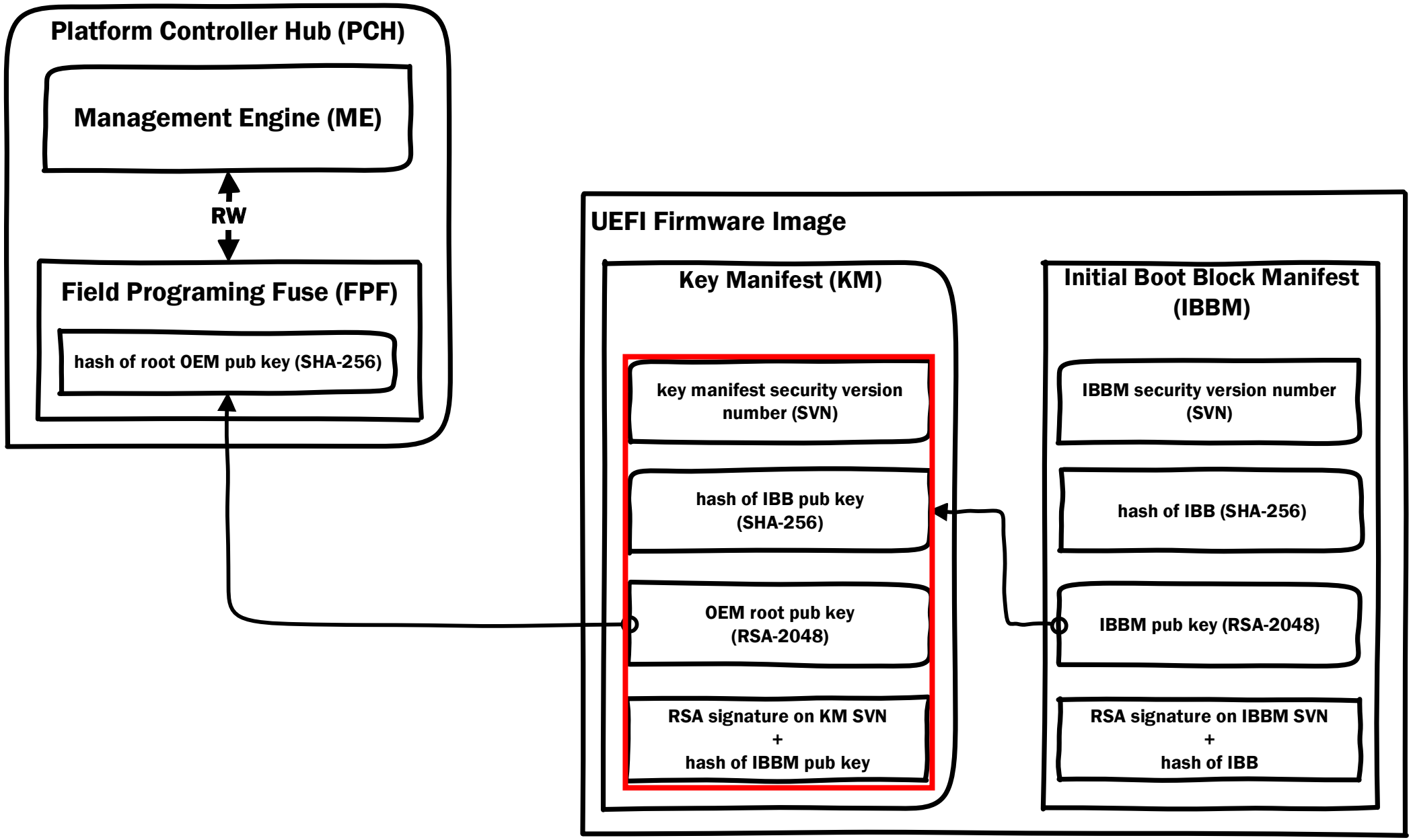
2016.zer :Guard.pdf



You never attack
the standard, you attack
the implementation, including the process

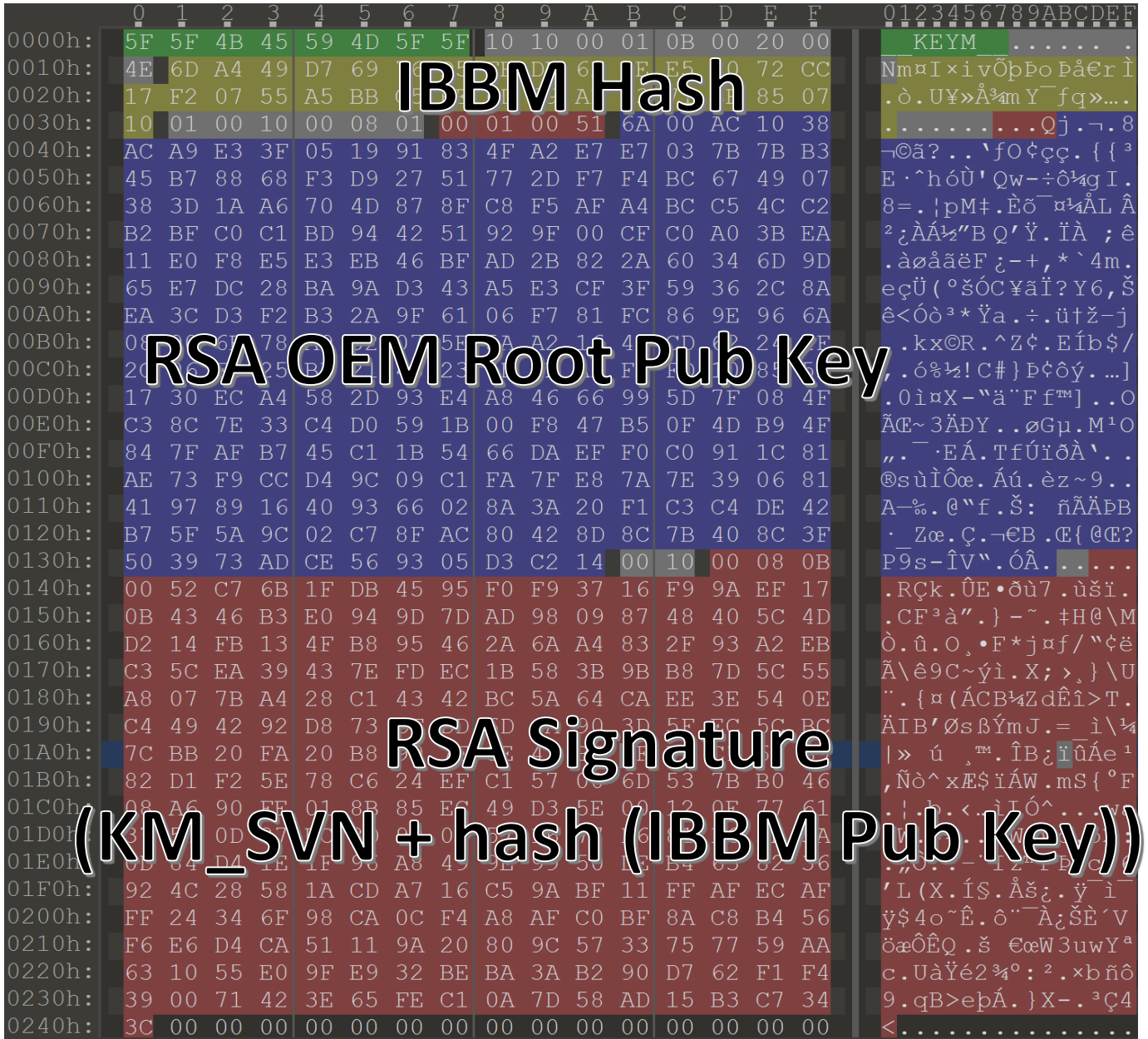
Grugq

Boot Guard: Chain of Trust

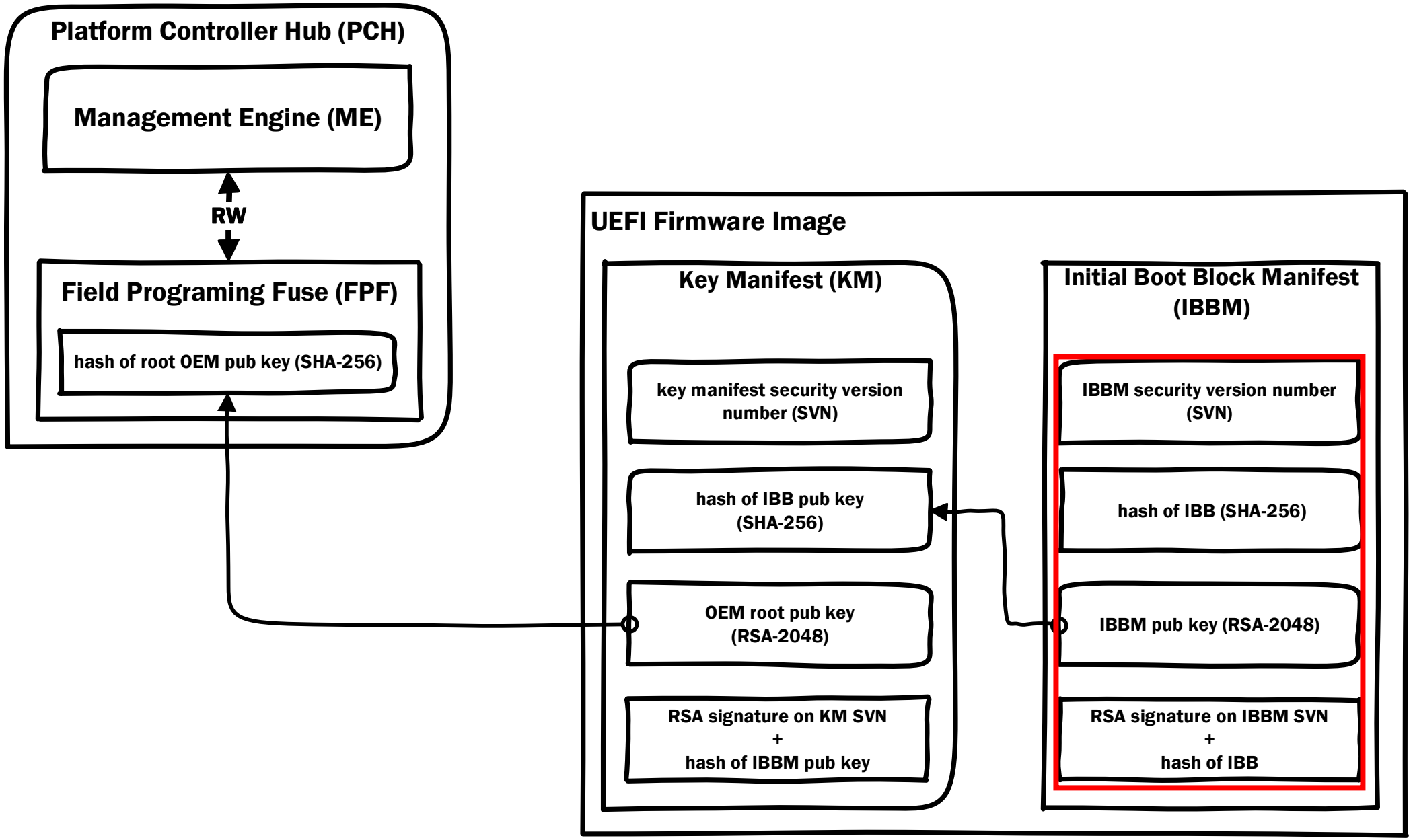


Boot Guard: Key Manifest (KM)

- struct BOOT_GUARD_KEY_MANIFEST BGKM
  - UBYTE Signature[8]
  - UBYTE Unknown
  - UBYTE Unknown1
  - UBYTE KmSvn
  - UBYTE Unknown2
  - UBYTE Unknown3
  - UINT16 Unknown4[0]
  - struct KEY_HASH IbbmKeyHash
  - UBYTE Unknown4[1]
  - UINT16 Unknown5
  - struct KEY_RSA OemPubKey
    - struct RSA_PUBLIC_KEY Key
      - UBYTE Unknown8
      - UINT16 Size
      - UINT32 Exp
      - UBYTE PubKey[256]
    - UINT16 Unknown16
    - struct RSA_SIGNATURE Signature
      - UINT16 KeySize
      - UINT16 Unknown16
      - UBYTE Signature[256]

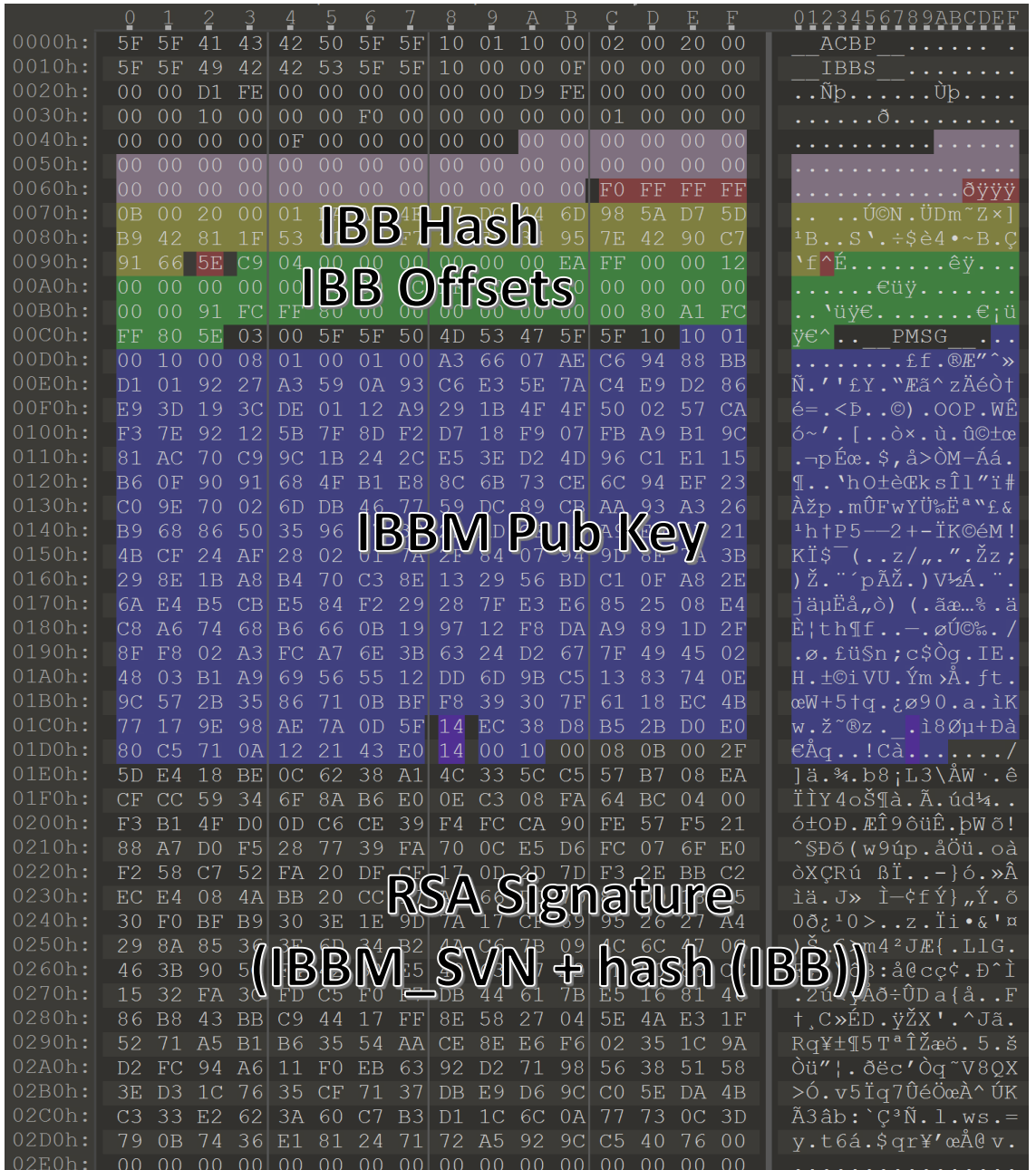


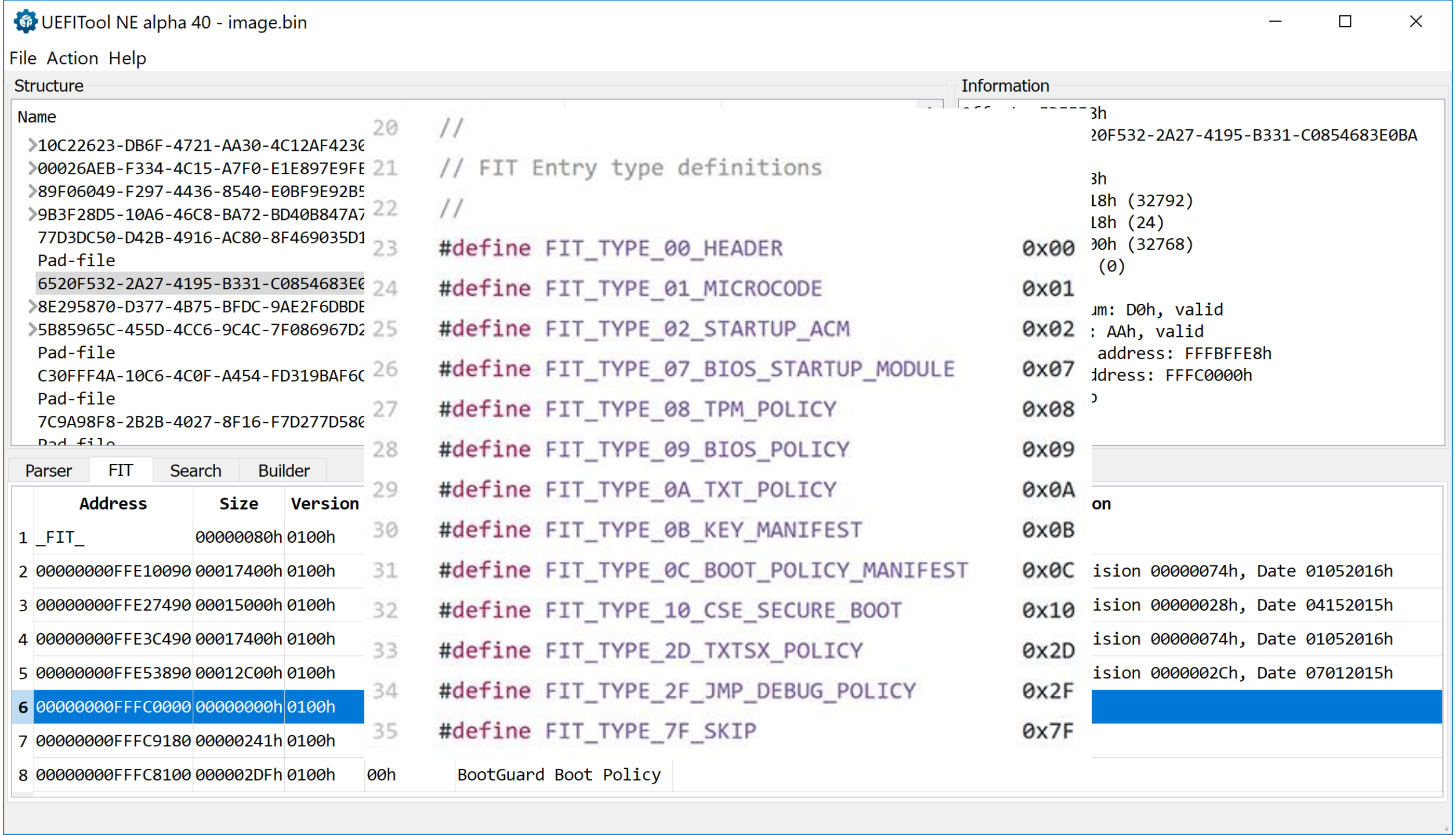
Boot Guard: Chain of Trust



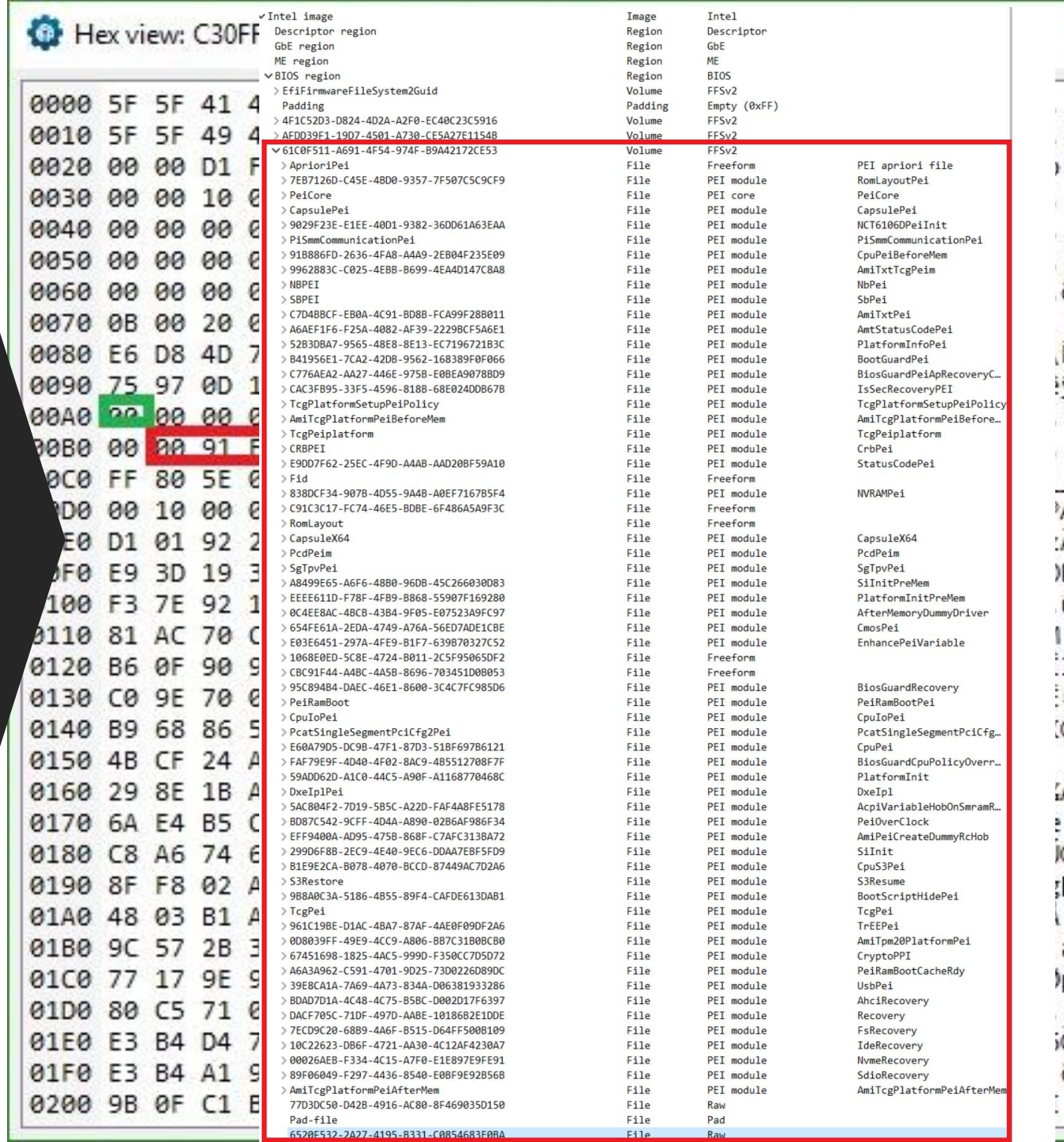
Boot Guard: Boot Policy Manifest (BPM)

- struct BOOT_POLICY_MANIFEST BPM
  - struct BOOT_POLICY_MANIFEST_HEADER Hdr
    - UBYTE Signature[8]
    - UBYTE Unknown
    - UBYTE Unknown2
    - UBYTE Unknown3
    - UBYTE Unknown4
    - UBYTE AcmSvn
    - UBYTE Unknown5
    - UINT16 Unknown6
  - struct IBB_ELEMENT IBBS
    - UBYTE Signature[8]
    - UBYTE Unknown
    - UBYTE Unknown1[2]
    - UBYTE Unknown2
    - UINT32 Unknown3
    - UINT64 Unknown4
    - UINT64 VtdBar
    - UINT32 Unknown5
    - UINT32 Unknown6
    - UINT64 Unknown7[2]
    - UINT16 Unknown8
    - struct KEY_HASH IbbHash
    - UINT32 EntryPoint
    - struct KEY_HASH SigHash
    - UBYTE SegmentNum
    - struct IBB_SEGMENT IbbSegment[4]
  - struct PLATFORM_MANUFACTURER PM
  - struct BOOT_POLICY_MANIFEST_SIGNATURE BPMS
    - UBYTE Signature[8]
    - UBYTE Version
    - struct RSA_SIGNATURE KeySignature



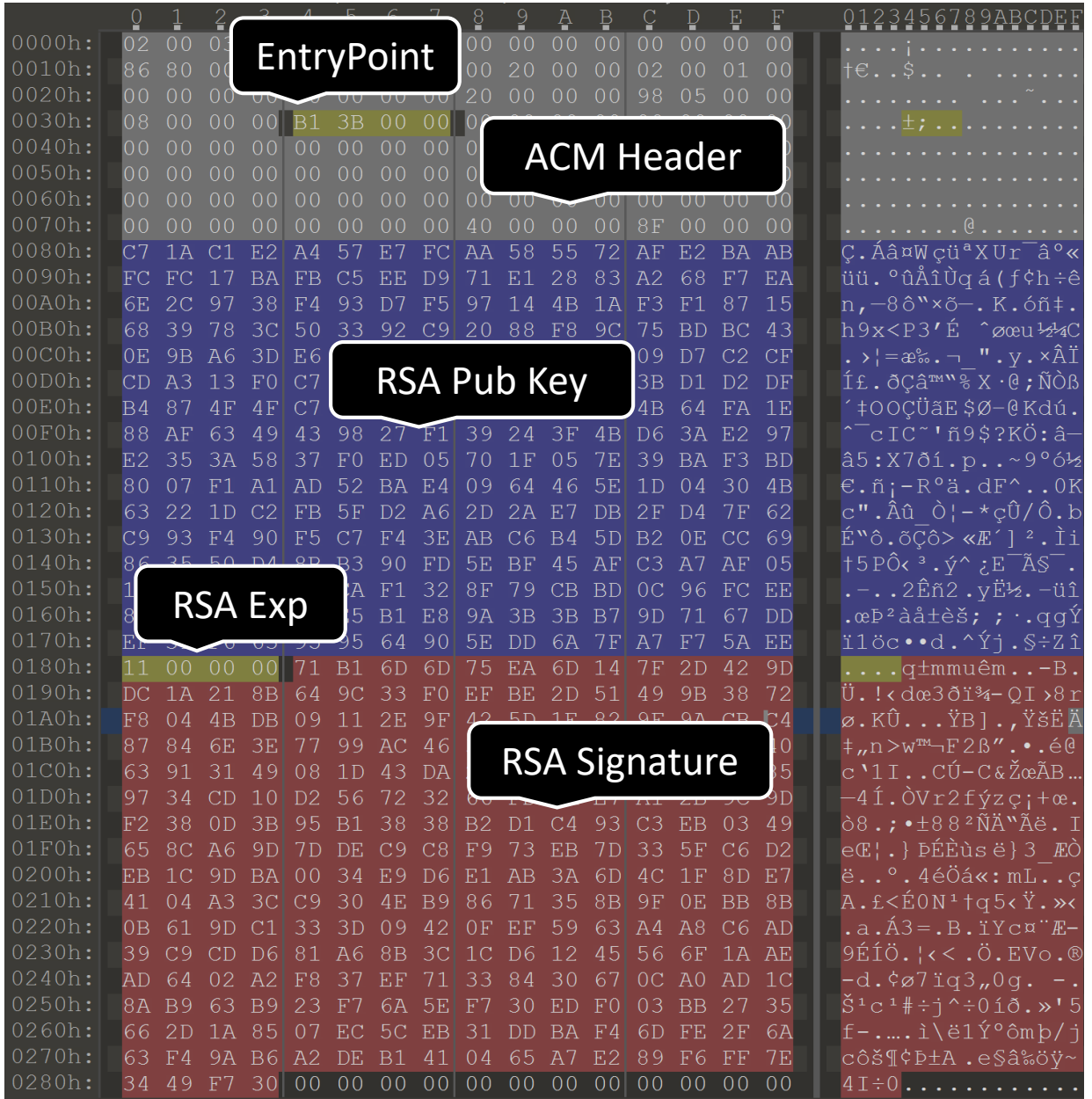

UEFITool NE alpha 40 - image.bin
File Action Help
Structure
Name
10C22623-DB6F-4721-AA30-4C12AF4236
00026AEB-F334-4C15-A7F0-E1E897E9FE
89F06049-F297-4436-8540-E0BF9E92B5
9B3F28D5-10A6-46C8-BA72-BD40B847A7
77D3DC50-D42B-4916-AC80-8F469035D1
Pad-file
6520F532-2A27-4195-B331-C0854683E0
8E295870-D377-4B75-BFDC-9AE2F6DBDE
5B85965C-455D-4CC6-9C4C-7F086967D2
Pad-file
C30FFF4A-10C6-4C0F-A454-FD319BAF6C
Pad-file
7C9A98F8-2B2B-4027-8F16-F7D277D580
Information
3h
20F532-2A27-4195-B331-C0854683E0BA
3h
L8h (32792)
L8h (24)
00h (32768)
(0)
um: D0h, valid
: AAh, valid
address: FFFBFFE8h
ddress: FFFC0000h
Parser
FIT
Search
Builder
Address
Size
Version
on
1 _FIT_ 00000080h 0100h
2 00000000FFE10090 00017400h 0100h
ision 00000074h, Date 01052016h
3 00000000FFE27490 00015000h 0100h
ision 00000028h, Date 04152015h
4 00000000FFE3C490 00017400h 0100h
ision 00000074h, Date 01052016h
5 00000000FFE53890 00012C00h 0100h
ision 0000002Ch, Date 07012015h
6 00000000FFFC0000 00000000h 0100h
7 00000000FFFC9180 00000241h 0100h
8 00000000FFFC8100 000002DFh 0100h 00h BootGuard Boot Policy
20 //
21 // FIT Entry type definitions
22 //
23 #define FIT_TYPE_00_HEADER 0x00
24 #define FIT_TYPE_01_MICROCODE 0x01
25 #define FIT_TYPE_02_STARTUP_ACM 0x02
26 #define FIT_TYPE_07_BIOS_STARTUP_MODULE 0x07
27 #define FIT_TYPE_08_TPM_POLICY 0x08
28 #define FIT_TYPE_09_BIOS_POLICY 0x09
29 #define FIT_TYPE_0A_TXT_POLICY 0x0A
30 #define FIT_TYPE_0B_KEY_MANIFEST 0x0B
31 #define FIT_TYPE_0C_BOOT_POLICY_MANIFEST 0x0C
32 #define FIT_TYPE_10_CSE_SECURE_BOOT 0x10
33 #define FIT_TYPE_2D_TXTSX_POLICY 0x2D
34 #define FIT_TYPE_2F_JMP_DEBUG_POLICY 0x2F
35 #define FIT_TYPE_7F_SKIP 0x7F

Boot Guard: Initial Boot Block (IBB)



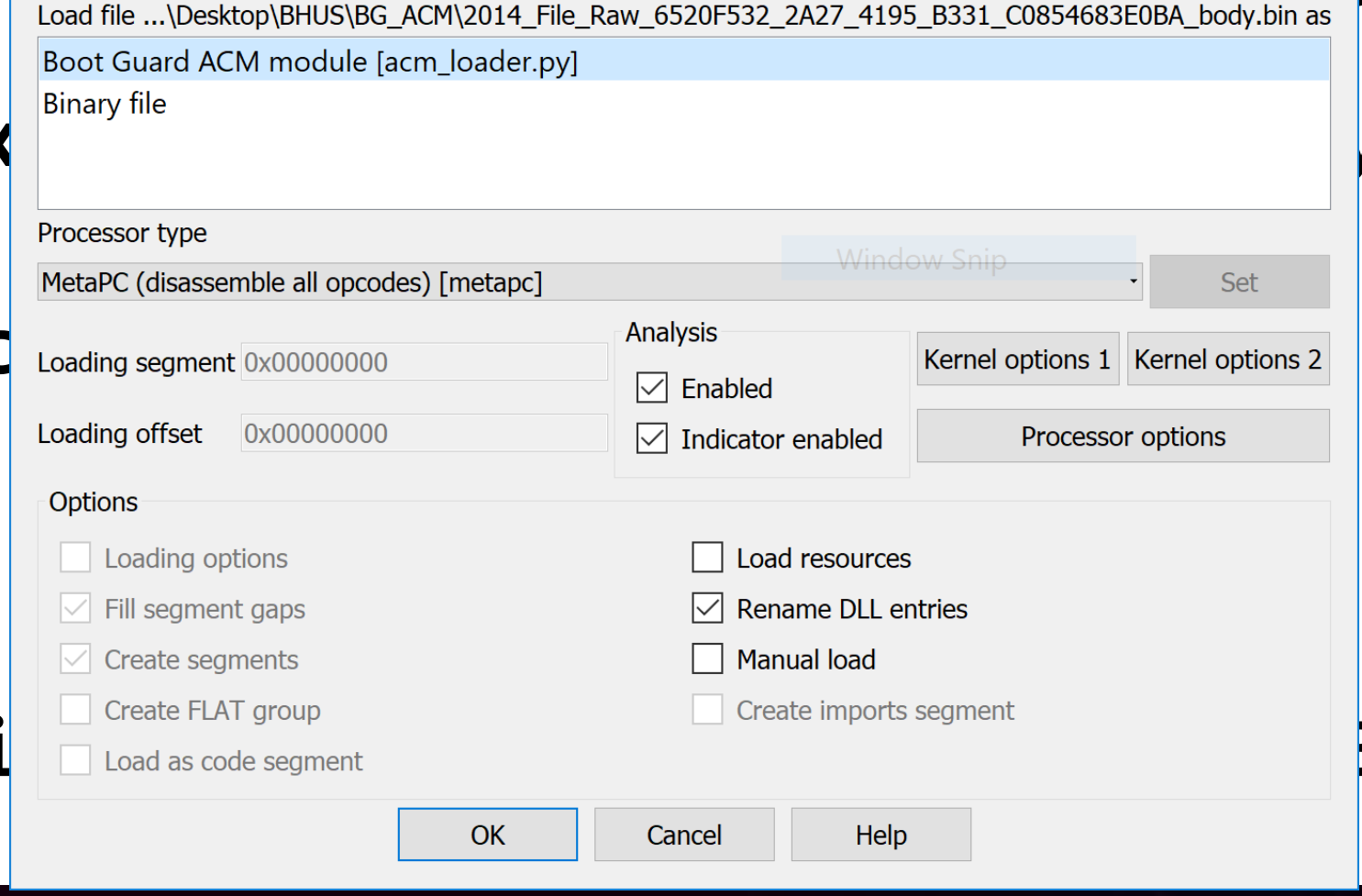
Boot Guard: Authenticated Code Module (ACM)

struct ACM_HEADER ACM	
UINT32 ModuleType	30002h
UINT32 HeaderType	A1h
UINT32 Unknown[2]	
UINT32 ModuleVendor	8086h
UINT32 Date	20150624h
UINT32 ModuleSize	2000h
UINT16 AcmSvn	2h
UINT16 Unknown1	1h
UINT32 Unknown2[5]	
UINT32 EntryPoint	3BB1h
UBYTE Unknown3[64]	
UINT32 KeySize	40h
UINT32 Unknown4	8Fh
UBYTE RsaPubKey[256]	
UINT32 RsaPubExp	11h
UBYTE RsaSig[256]	



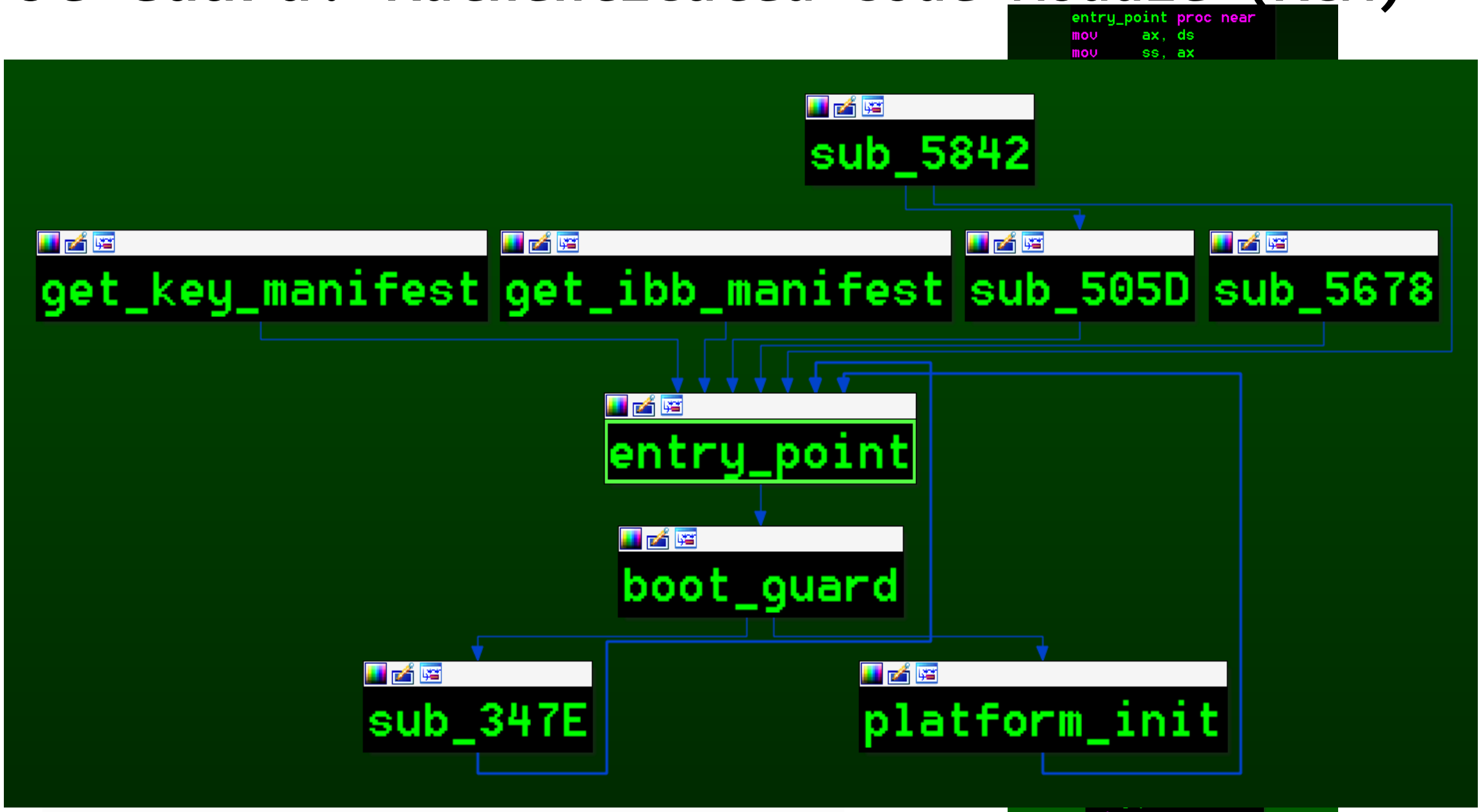
Boot Guar (ACM)

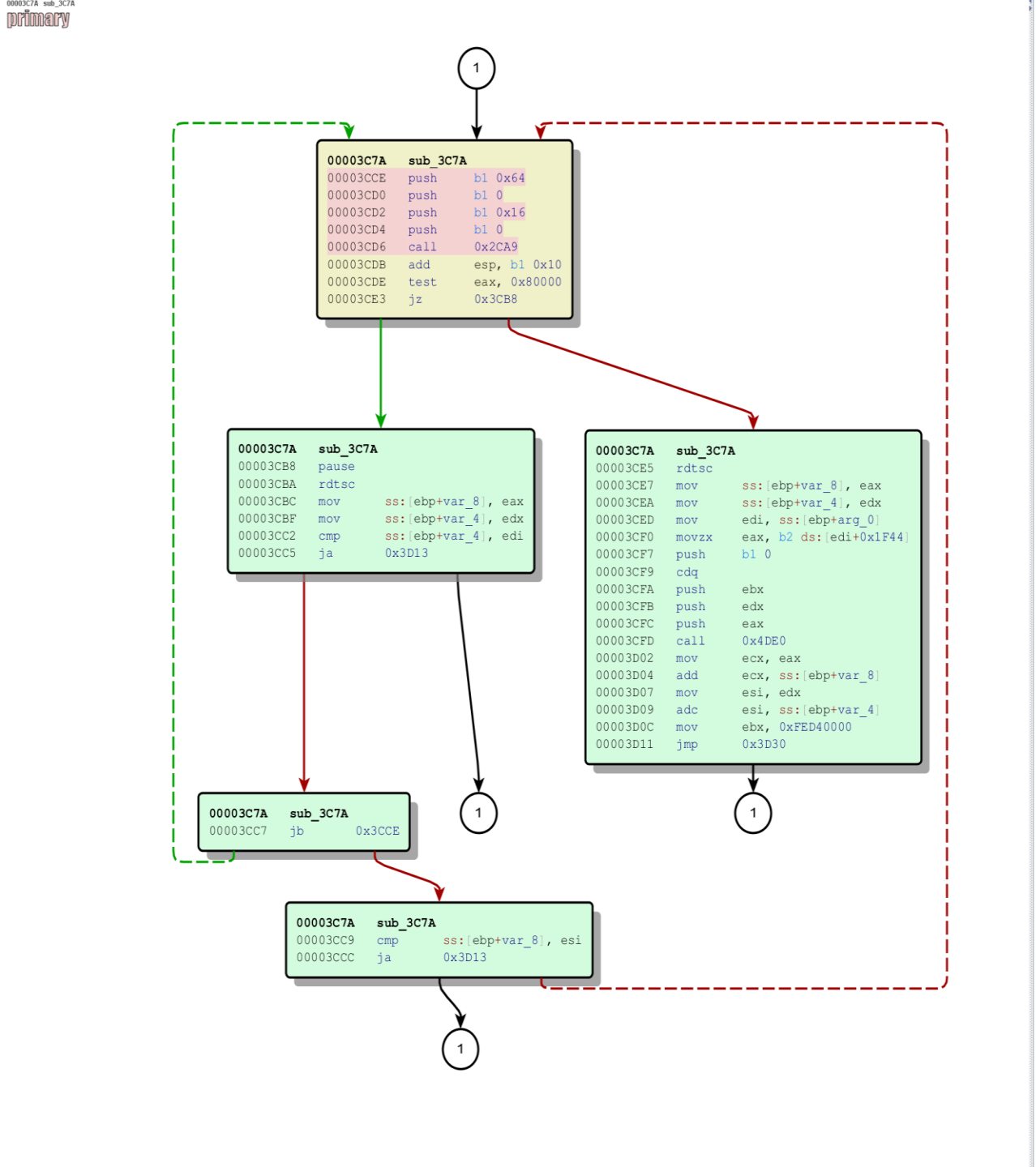
- ACM is x y Intel
- ACM exec or NEM)
- ACM has
- ACM veri BB (IBBM)

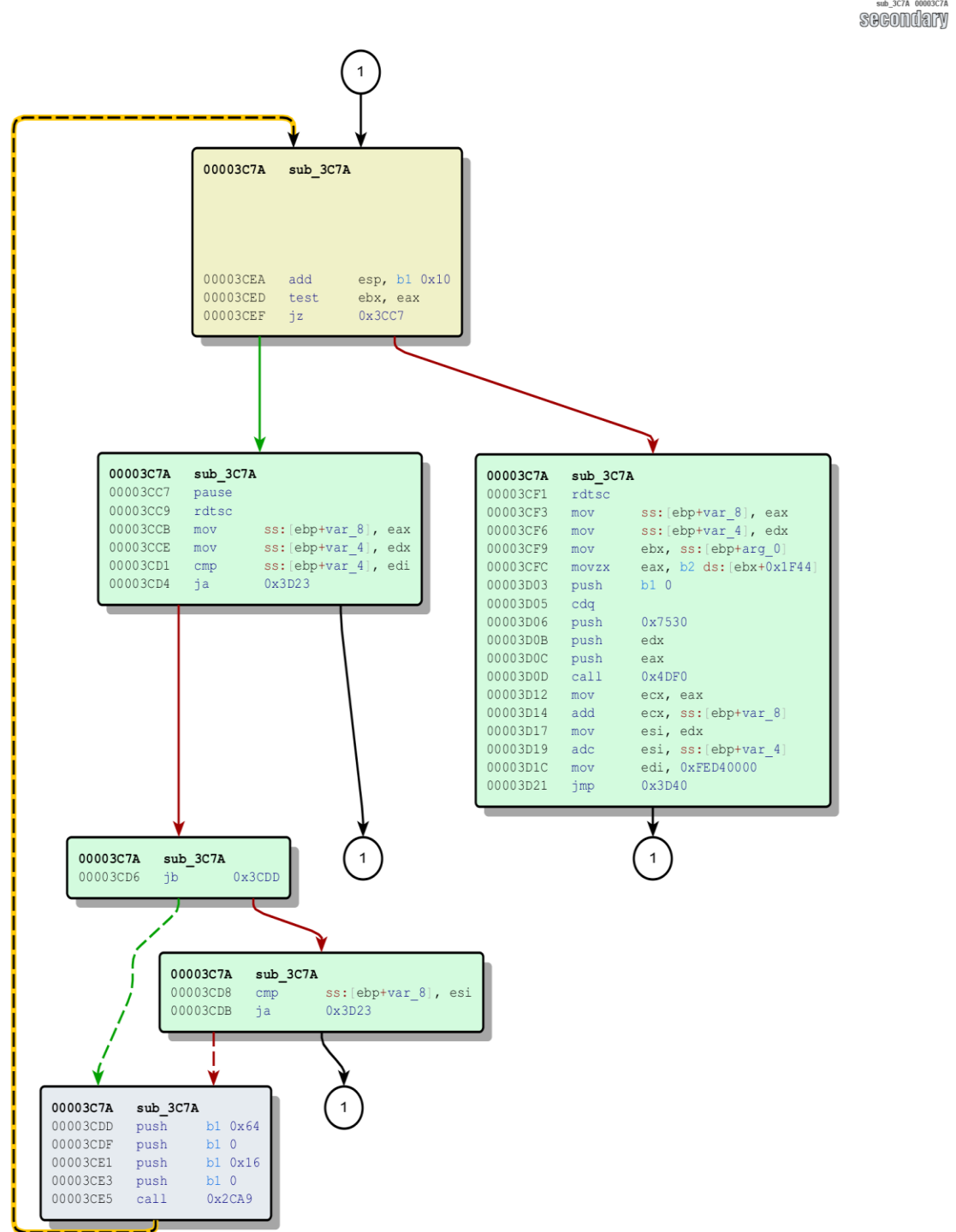



```
c:\Users\matrosov\Desktop\cpu_rec-1.0\cpu_rec-1.0>python cpu_rec.py -v BootGuard_ACM.bin
INFO : Default set of size 11 is read; 8 different CPUs known
INFO : ... MarkovCrossEntropy[2-grams;A] done in 1.294000s
INFO : ... MarkovCrossEntropy[3-grams;A] done in 1.796000s
BootGuard_ACM.bin                                          full(0x8000)   X86
INFO : ... window size 0x800 done in 0.340000s
chunk(0x4c00;19)    X86
```

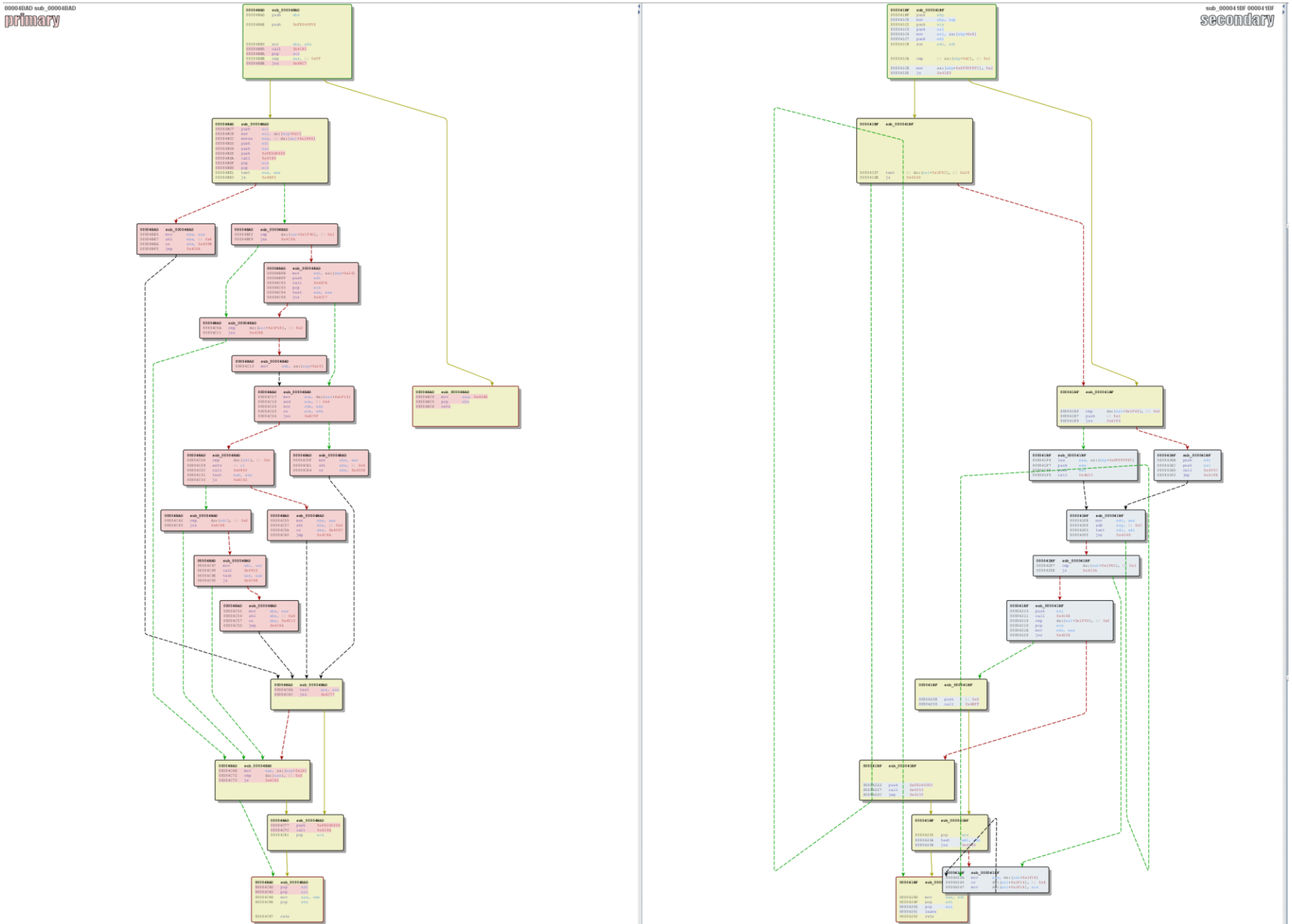
Boot Guard: Authenticated Code Module (ACM)




primary
00003C7A sub_3C7A
00003CCE push b1 0x64
00003CD0 push b1 0
00003CD2 push b1 0x16
00003CD4 push b1 0
00003CD6 call 0x2CA9
00003CDB add esp, b1 0x10
00003CDE test eax, 0x80000
00003CE3 jz 0x3CB8
00003C7A sub_3C7A
00003CB8 pause
00003CBA rdtsc
00003CBC mov ss:[ebp+var_8], eax
00003CBF mov ss:[ebp+var_4], edx
00003CC2 cmp ss:[ebp+var_4], edi
00003CC5 ja 0x3D13
00003C7A sub_3C7A
00003CE5 rdtsc
00003CE7 mov ss:[ebp+var_8], eax
00003CEA mov ss:[ebp+var_4], edx
00003CED mov edi, ss:[ebp+arg_0]
00003CF0 movzx eax, b2 ds:[edi+0x1F44]
00003CF7 push b1 0
00003CF9 cdq
00003CFA push ebx
00003CFB push edx
00003CFC push eax
00003CFD call 0x4DE0
00003D02 mov ecx, eax
00003D04 add ecx, ss:[ebp+var_8]
00003D07 mov esi, edx
00003D09 adc esi, ss:[ebp+var_4]
00003D0C mov ebx, 0xFED40000
00003D11 jmp 0x3D30
00003C7A sub_3C7A
00003CC7 jb 0x3CCE
00003C7A sub_3C7A
00003CC9 cmp ss:[ebp+var_8], esi
00003CCC ja 0x3D13
1
1
1
1
1


secondary
00003C7A sub_3C7A
00003CEA add esp, b1 0x10
00003CED test ebx, eax
00003CEF jz 0x3CC7
00003C7A sub_3C7A
00003CC7 pause
00003CC9 rdtsc
00003CCB mov ss:[ebp+var_8], eax
00003CCE mov ss:[ebp+var_4], edx
00003CD1 cmp ss:[ebp+var_4], edi
00003CD4 ja 0x3D23
00003C7A sub_3C7A
00003CF1 rdtsc
00003CF3 mov ss:[ebp+var_8], eax
00003CF6 mov ss:[ebp+var_4], edx
00003CF9 mov ebx, ss:[ebp+arg_0]
00003CFC movzx eax, b2 ds:[ebx+0x1F44]
00003D03 push b1 0
00003D05 cdq
00003D06 push 0x7530
00003D0B push edx
00003D0C push eax
00003D0D call 0x4DF0
00003D12 mov ecx, eax
00003D14 add ecx, ss:[ebp+var_8]
00003D17 mov esi, edx
00003D19 adc esi, ss:[ebp+var_4]
00003D1C mov edi, 0xFED40000
00003D21 jmp 0x3D40
00003C7A sub_3C7A
00003CD6 jb 0x3CDD
00003C7A sub_3C7A
00003CD8 cmp ss:[ebp+var_8], esi
00003CDB ja 0x3D23
00003C7A sub_3C7A
00003CDD push b1 0x64
00003CDF push b1 0
00003CE1 push b1 0x16
00003CE3 push b1 0
00003CE5 call 0x2CA9
1
1
1
1

Boot Guard ACM BinDiff: Broadwell vs Skylake



Boot Guard BIOS Components (AMI)

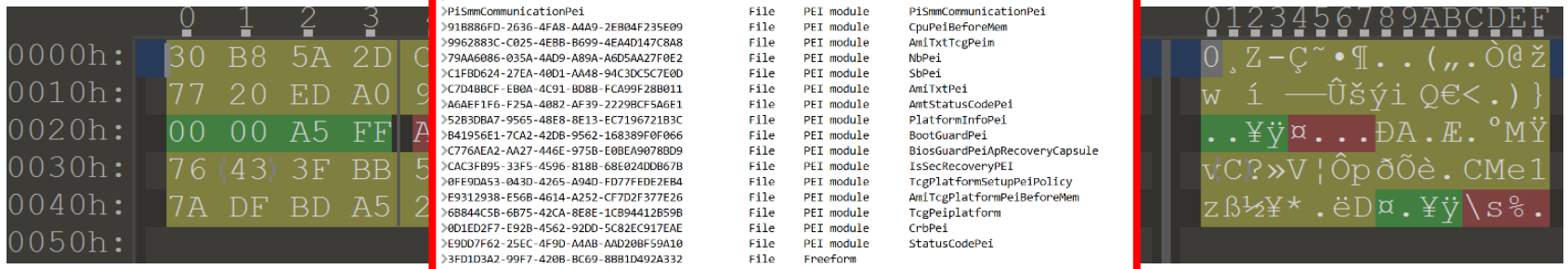
- **PEI**
  - **BootGuardPei** [B41956E1-7CA2-42db-9562-168389F0F066]
- **SMM**
  - **VerifyFwBootGuard** [EE89F590-A816-4ac5-B3A9-1BC759B12439]
- **DXE**
  - **BootGuardDxe** [1DB43EC9-DF5F-4cf5-AAF0-0E85DB4E149A]

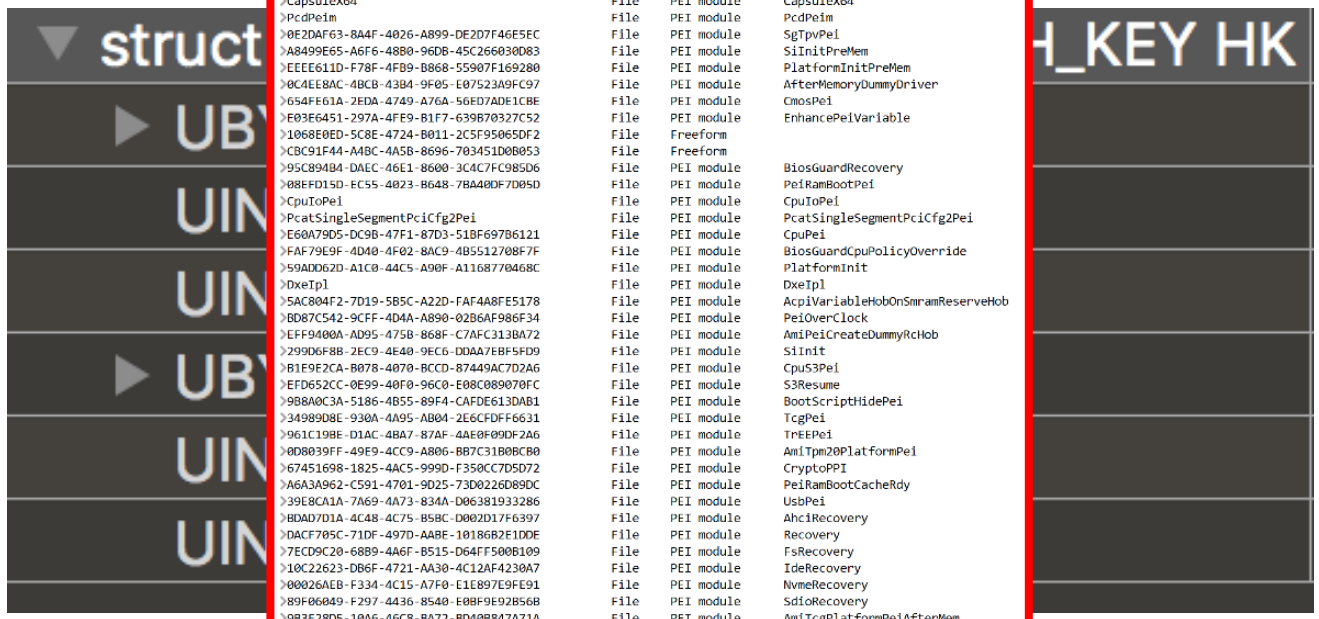
BootGuardPei Validation Flow

```
EFI_STATUSBootGuardPei(EFI_PEI_SERVICES **PeiServices, VOID *Ppi)
{
    ...

    Status = GetBootMode ();
    if ( EFI_ERROR( Status ) ) {
        return   Status;
    }

    ...

    if ( (BootMode == BOOT_IN_RECOVERY_MODE) || (BootMode == BOOT_ON_FLASH_UPDATE) || BootMode == BOOT_ON_S3_RESUME) {
        return   Status;
    }

    BootGuardVerifyTransitionPEItoDXEFlag = 0;

    ...

    CalculateSha256(BootGuardHashKeySegment0);
    CalculateSha256(CurrentBootGuardHashKey0);

    if ( !MemCmp(BootGuardHashKeySegment0, CurrentBootGuardHashKey0, 32) ) {
        BootGuardVerifyTransitionPEItoDXEFlag = 1;
    } else {
        BootGuardVerifyTransitionPEItoDXEFlag = 0;
        return   EFI_SUCCESS;
    }

    if ( !((BootGuardHashKeySegment1 == 0) {
        CalculateSha256 (BootGuardHashKeySegment1);
        CalculateSha256 (CurrentBootGuardHashKey1);

        if ( !MemCmp(BootGuardHashKeySegment1, CurrentBootGuardHashKey1, 32) ) {
            BootGuardVerifyTransitionPEItoDXEFlag = 1;
        } else {
            BootGuardVerifyTransitionPEItoDXEFlag = 0;
            return   EFI_SUCCESS;
        }
    }

    return   Status;
}
```

Boot Guard: PEI FV_HASH

➢ **FV_HASH_KEY** **451D0B053]**





VerifyFwBootGuard SMM Validation Flow (Intel ME communications over HECI)

- Find and Verify ACM
  - Verify ACM SVN
- Find and Verify Key Manifest (KM)
  - Verify KM SVN
- Find and Verify Boot Policy Manifest (BPM)
  - Verify BPM SVN
- If something wrong return EFI_SECURITY_VIOLATION

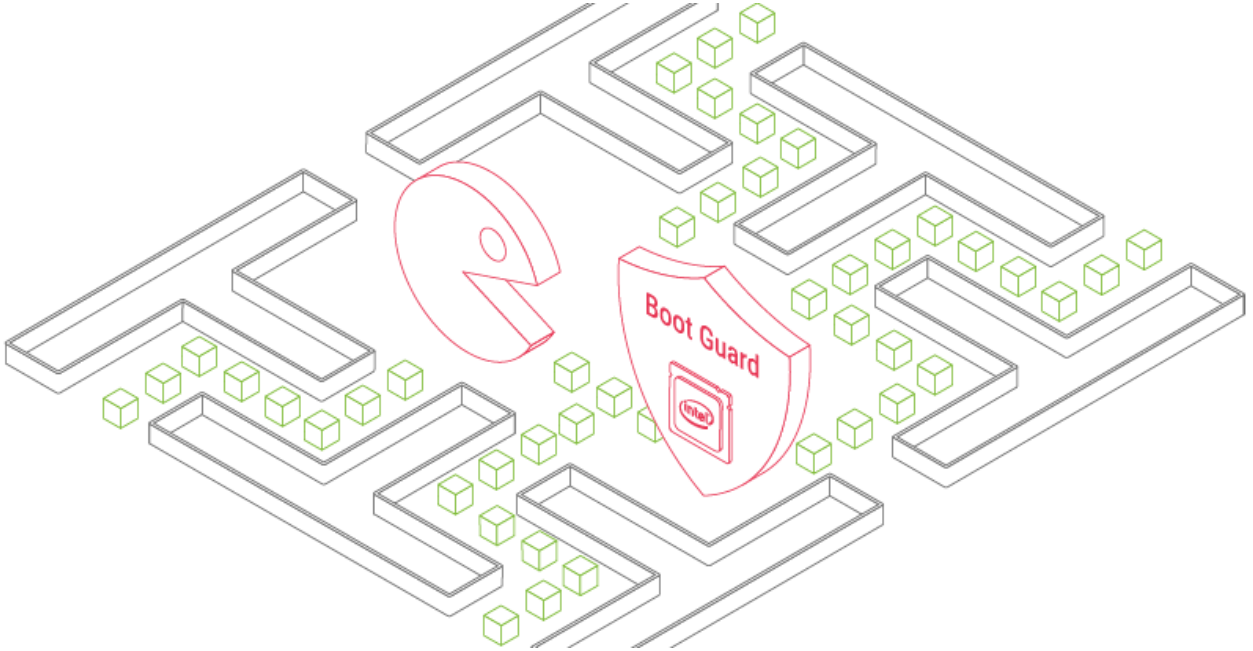
BootGuardDxe Validation Flow

```
EFI_STATUS BootGuardDxe(EFI_HANDLE ImageHandle, EFI_SYSTEM_TABLE *SystemTable)
{
    ...

    if ( BootGuardSupported() == FALSE ) {
        return   EFI_SUCCESS;
    }

    ...

    BootMode  = Ge[illegible]Mo[illegible]();
    if ( (Boo[illegible] == BOOT_IN_RECOVERY_MODE) || (BootMode == BOOT_ON_FLASH_UPDATE) ) {
      re[illegible] EFI_SUCCESS;
    }

    ...

    [redacted]{

    return   EFI_SUCCESS;
}
```

S3 rootkits coming :-)

← one more 0-day bug?

BootGuardDxe Validation Flow



- https://embedi.com/blog/bypassing-intel-boot-guard
- Intel NUC Boot Guard Bypass CVE-2017-5722 kudos to Alex Ermolov
- https://security-center.intel.com/advisory.aspx?intelid=INTEL-SA-00084

```
Intel(R) MEInfo Version: 9.1.20.1020
Copyright(C) 2005 - 2014, Intel Corporation. All rights reserved.

GBE Region does not exist.
Intel(R) Manageability and Security Application code versions:

BIOS Version:
MEBx Version:
Gbe Version:
VendorID:
PCH Version:
FW Version:
LMS Version:
MEI Driver Version:

FW Capabilities:

    Intel(R) Small Busin
    Intel(R) Anti-Theft
    Intel(R) Capability
    Protect Audio Video
    Intel(R) Dynamic App
    Service Advertisemen

TLS:
Last ME reset reason:
Local FWUpdate:
BIOS Config Lock:
Host Read Access to ME:
Host Write Access to ME:
SPI Flash ID #1:
SPI Flash ID VSCC #1:
SPI Flash BIOS VSCC:
BIOS boot State:
OEM Id:
Capability Licensing Ser
OEM Tag:
Slot 1 Board Manufacture
Slot 2 System Assembler:
Slot 3 Reserved:
M3 Autotest:                    Disabled
Localized Language:             English
Independent Firmware Recovery:  Enabled
```

OEM Public Key Hash	EE 7D B6 9F 8B 18 F5 41 F6 46 ...


```
B18F541F6467089E8A4A0388EA0E259284CC42CAD7CEA3AF3BA7260

0000000000000000000000000000000000000000000000000000

                ME
                --
                Enabled
                Enabled
                Enabled
                Enabled
                Enabled
                Enabled
                0x1
                0x3
                Enabled
ed
```

Boot Guard Config

Parameter	Value
Key Manifest ID	0x0001
Boot Guard Profile Configuration	Boot Guard Profile 5 - FVME
CPU Debugging	Enabled

314)

(CVE-2017-11313)

copy from Gigabyte official website

Vertical Markets

- School
- University computer labs
- Libraries
- Hospital / Medical equipment
- Governmental

Powerful Commercial Applications

- Factory testing machine
- Bank ATM system
- Gaming equipment
- Vending machine
- Security system

Five steps to bypass Boot Guard

1) Modify UEFI firmware update image with rootkit/implant
or
Disable Intel Boot Guard

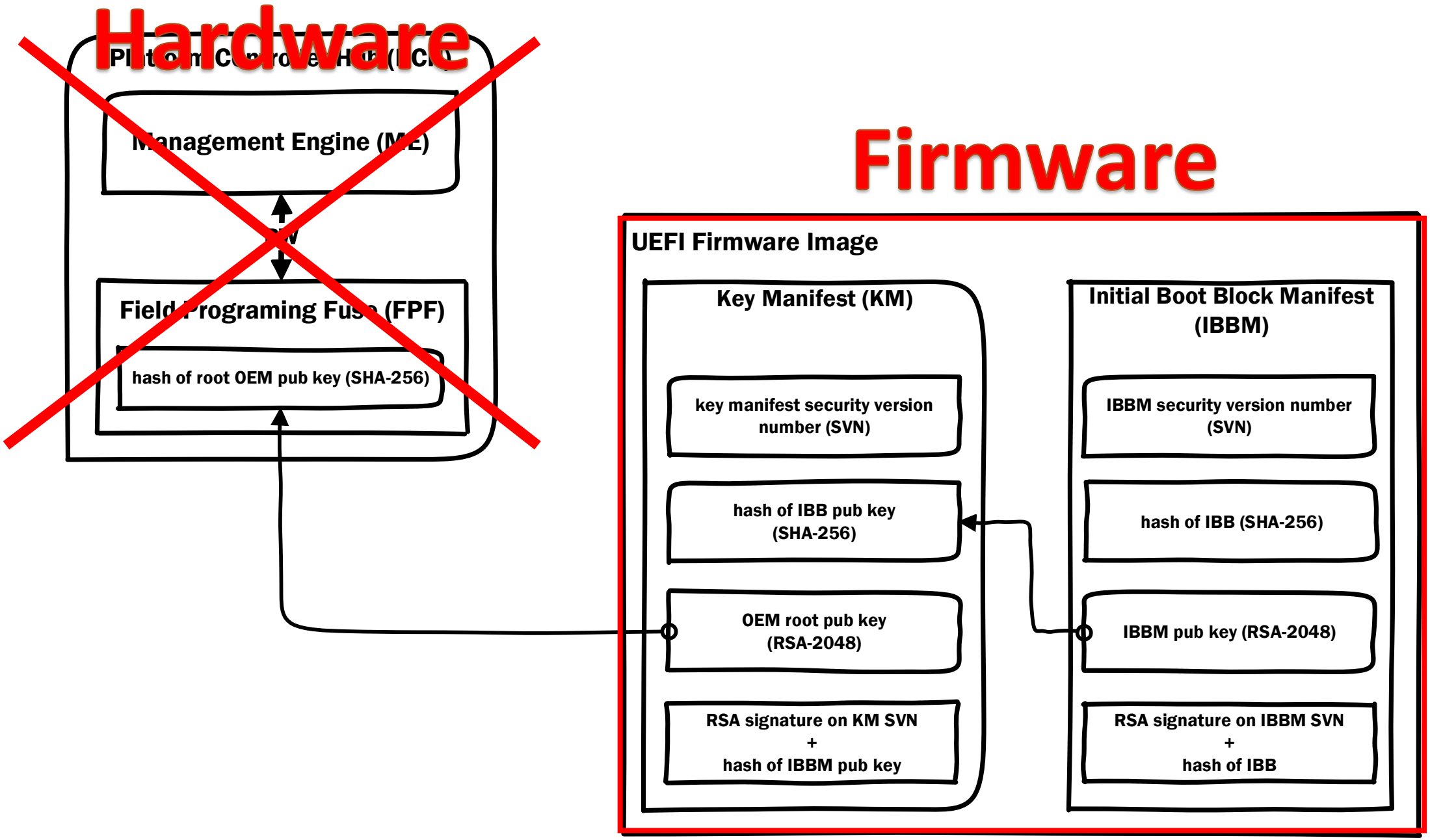
2) **Initial Boot Block (IBB)**
- ✓ Recalculate signature on 2048-bit RSA key pair for IBB
- ✓ Modify IBB manifest inside UEFI firmware update file
- ✓ Recalculate signature for IBB manifest with different 2048-bit RSA key pair

3) **Modify Root Key manifest**
- ✓ Recalculate SHA256 hash of the public key from Root Key Manifest

4) **Modify ME region with new key (CVE-2017-11314)**
- ✓ Modify Boot Guard configuration with active verified boot policy

5) **Lock Boot Guard configuration with by FPF (CVE-2017-11313)**

Boot Guard: Chain of Trust



Intel Statement

“Intel provides a 6th and 7th generation Core Platforms Secure Configuration Specification, which covers how to securely configure the platform. Additionally, Intel makes available a utility that our ecosystem partners can use to test and identify potential configuration issues.”

Gigabyte Statement

“For FPF issue, we discuss with internal the BIOS don’t need any update but we will add ME Lock tool to our production process soon, the new production ship will include ME Lock.”

UEFITool for Intel Boot Guard visual validation

https://github.com/LongSoft/UEFITool/releases/tag/A43

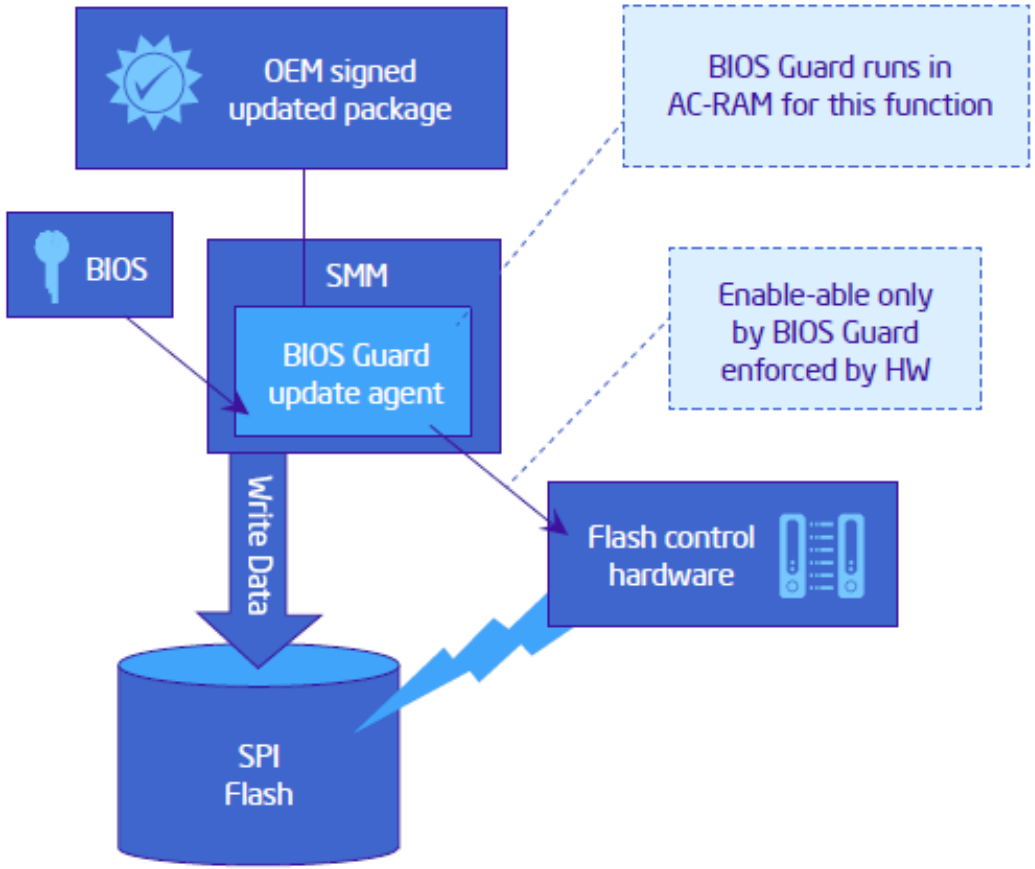
https://medium.com/@matrosov/bypass-intel-boot-guard-cc05edfca3a9

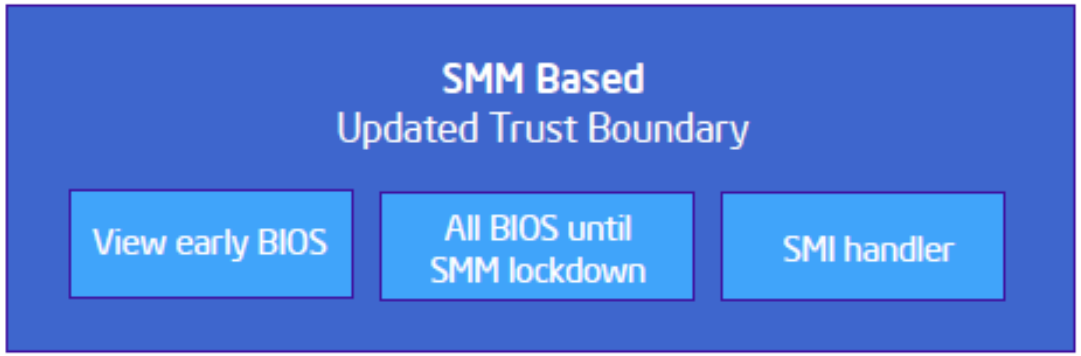
Intel BIOS Guard

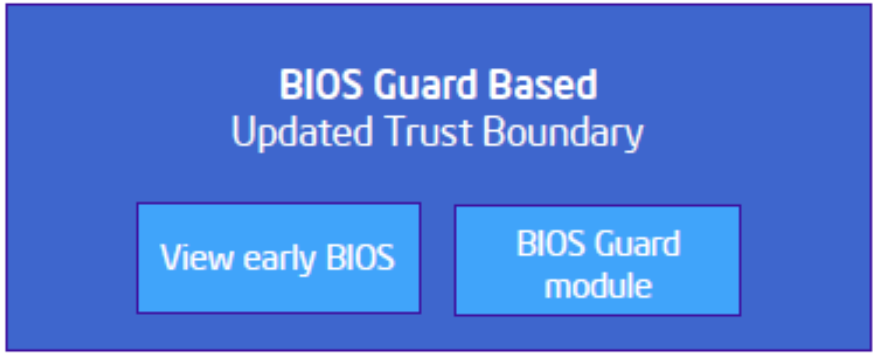
Intel BIOS Guard

- **Armoring SPI Flash access**
  - ✓ Access controlled by BIOS Guard ACM
  - ✓ Attack Surface = Firmware
- **BIOS update authentication**
  - ✓ Root of Trust = Hardware -> Trusted Platform Module (TPM)
  - ✓ Attack Surface = Firmware
- **Verified Boot -> since 2013**
  - ✓ Root of Trust = Hardware -> Field Programming Fuse (FPF)->Locked
  - ✓ Attack Surface = Firmware + Hardware

Demystifying Intel BIOS Guard







http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/security-technologies-4th-gen-core-retail-paper.pdf

Boot Guard BIOS Components (AMI)

- **PEI**
  - **BiosGuardPeiApRecoveryCapsule** [C776AEA2-AA27-446e-975B-E0BEA9078BD9]
  - **BiosGuardRecovery** [95C894B4-DAEC-46E1-8600-3C4C7FC985D6]
  - **BiosGuardCpuPolicyOverride** [FAF79E9F-4D40-4F02-8AC9-4B5512708F7F]

- **SMM**
  - **BiosGuardSmm** [44FE07D3-C312-4ad4-B892-269AB069C8E1]
  - **BiosGuardServices** [6D4BAA0B-F431-4370-AF19-99D6209239F6]

- **DXE**
  - **BiosGuardDxe** [6D1D13B3-8874-4e92-AED5-22FC7C4F7391]
  - **BiosGuardNvs** [17565311-4B71-4340-88AA-DC9F4422E53A]

Boot Guard BIOS Components (AMI)

- **PEI**
  - **BiosGuardPeiApRecoveryCapsule – AMI Capsule Update Validation**
  - **BiosGuardRecovery – Recovery Update Image parser**
  - **BiosGuardCpuPolicyOverride**
    - ✓ **Find Public Key**
    - ✓ **Find and Load BIOS Guard ACM**

- **SMM**
  - **BiosGuardSmm - Recovery SMI Handlers**

- **DXE**
  - **BiosGuardDxe - Recovery helper for update process**
    - ✓ **UEFI variable cleanup**
  - **BiosGuardNvs – ACPI helper for update process**
    - ✓ **AMI Capsule validation**

BIOS Guard Commands (AMI)

- PEI
  - BG_READ
  - BG_WRITE
  - BG_ERASE
  - BG_WRITE_ENABLE
  - BG_WRITE_DISABLE

- SMM
  - BG_READ
  - BG_WRITE
  - BG_ERASE

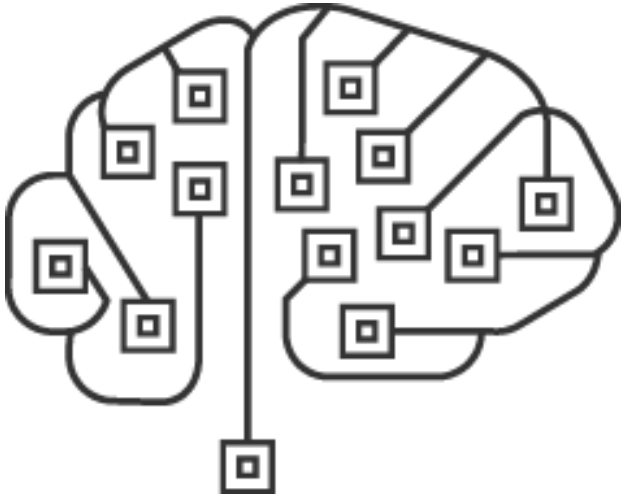
ZeroNights HackQuest starting at 10/23

http://hackquest.zeronights.org/


16-17 NOVEMBER 2017
ZERO
NIGHTS

All the stuff will be released on public

save the link:



https://github.com/REhints/BlackHat_2017

Thank you for your attention!

Alex Matrosov
@matrosov