BETRAYING THE BIOS:

GOING DEEPER INTO BIOS GUARD IMPLEMENTATIONS

Alex Matrosov

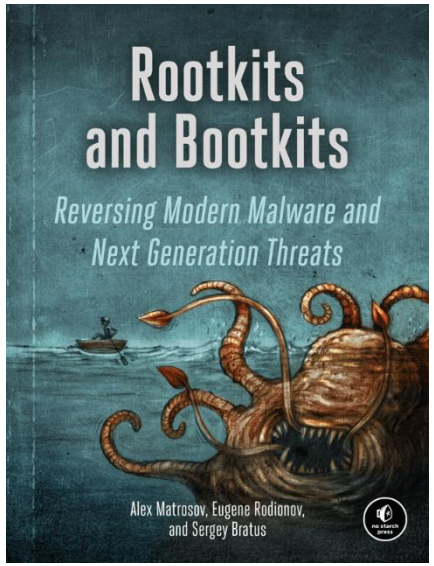
@matrosov

Leading Embedded Security at NVIDIA

Former Security Researcher @Cylance @Intel @ESET

Security REsearch since 1997

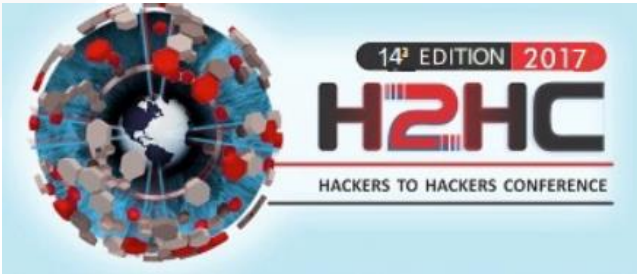
Book co-author nostarch.com/rootkits



@matrosov

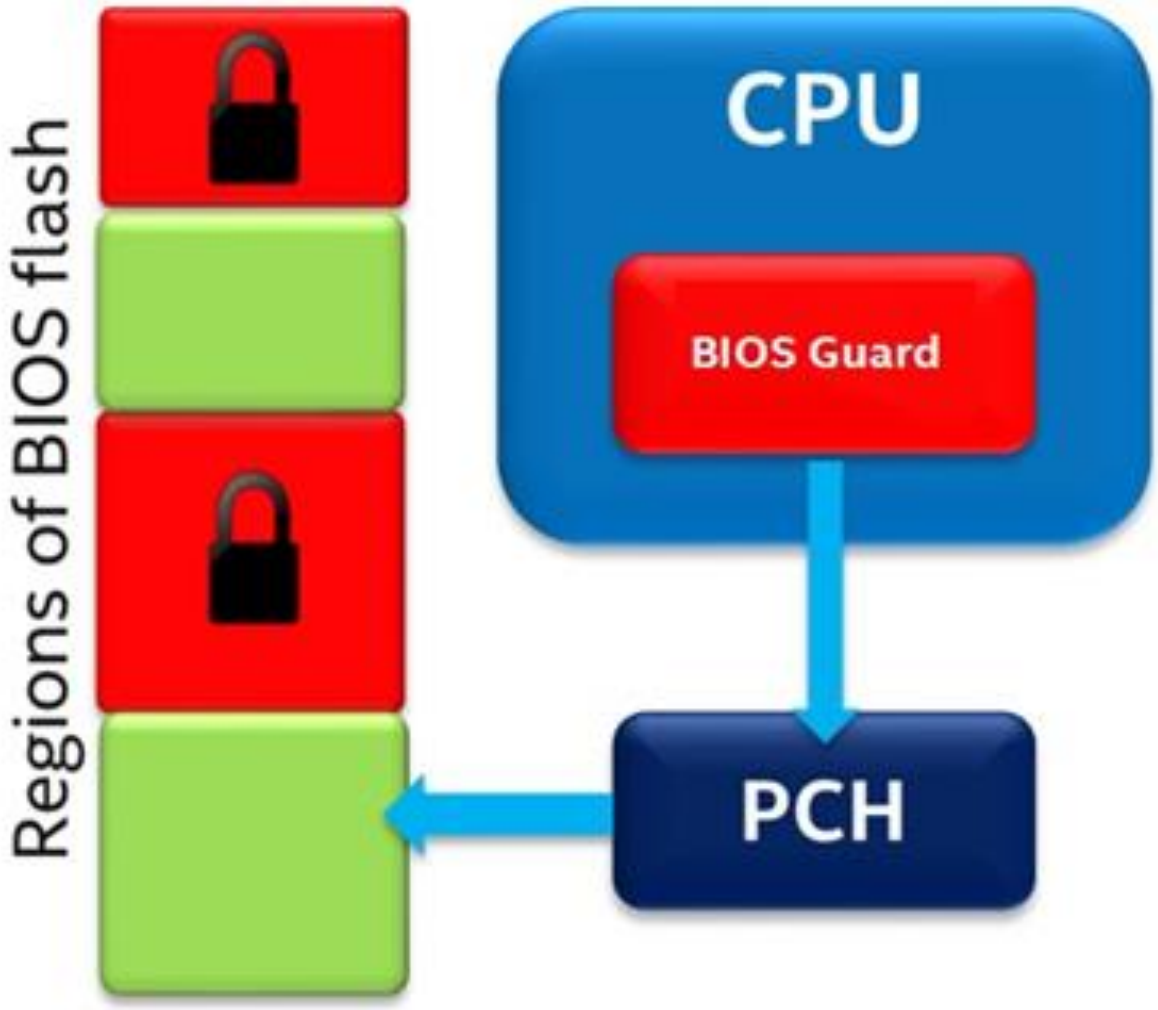
⌖Disclaimer⌖

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



WHAT’S NEW?

https://2017.zeronights.ru/wp-content/uploads/materials/zn2017_matrosov.pdf
https://www.youtube.com/watch?v=a2A-iuAGkSM

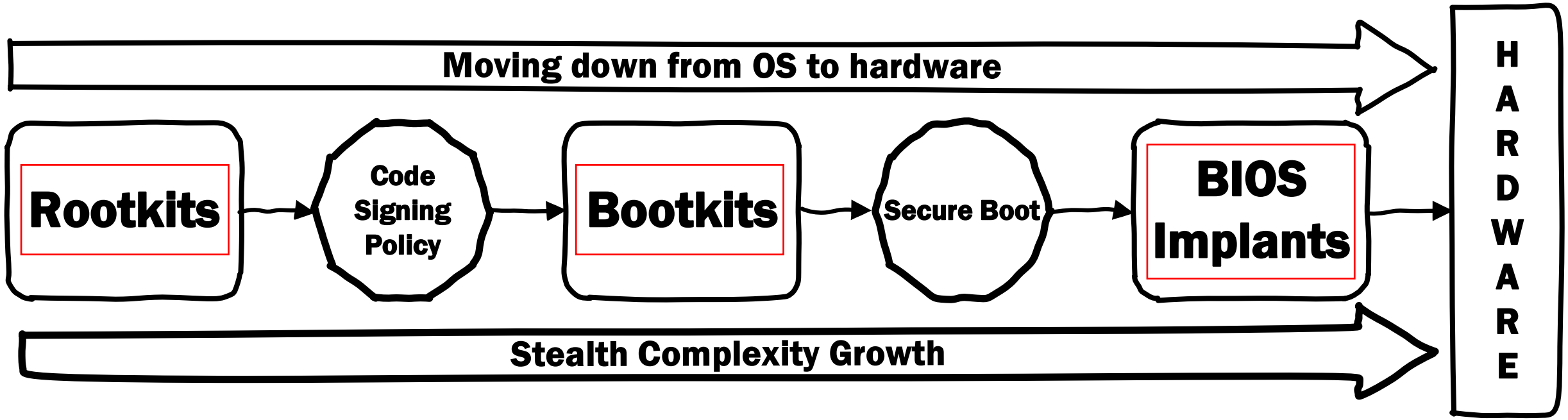


https://software.intel.com/en-us/articles/intel-xeon-processor-scalable-family-technical-overview

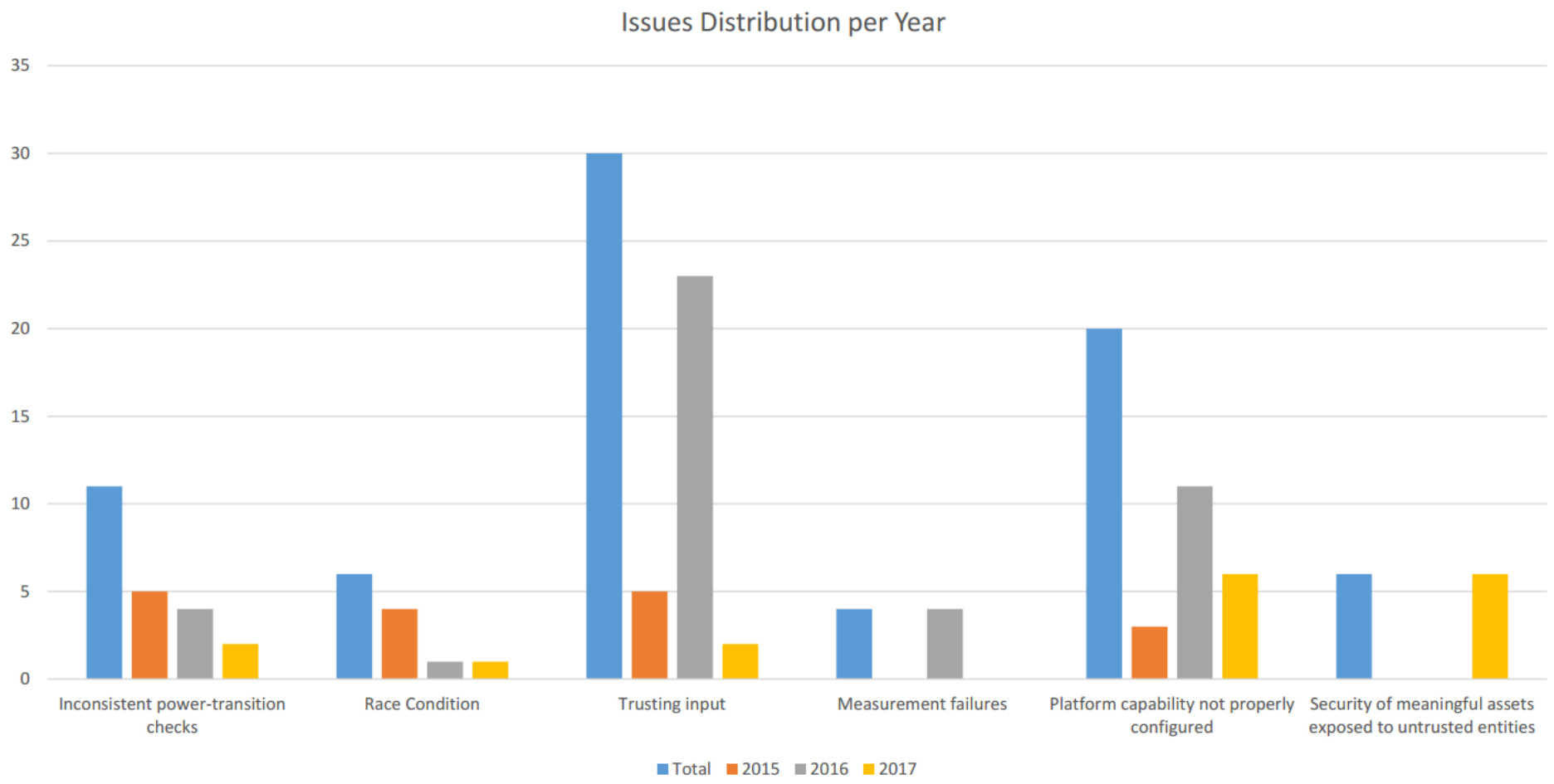
https://software.intel.com/en-us/articles/intel-xeon-processor-scalable-family-technical-overview

- Intro
- Breaking AMI BIOS Updates
  - ✓ Signed update tools fun
  - ✓ AMI updates authentication
- Intel BIOS Guard
  - ✓ AMI implementation details
  - ✓ BIOS Guard Script WTF?
- Intel Boot Guard
  - ✓ New Bypass Approaches OFFENSIVE
- ACM Dirty Secrets
  - ✓ Reversing ACM's

More mitigations, more rootkits complexity

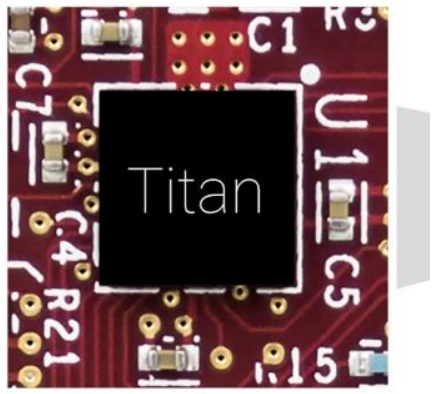


Growths of configuration based vulns



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

Locking Root of Trust in HW without Intel



Titan

Purpose-built chip to establish hardware root of trust for Google Cloud servers

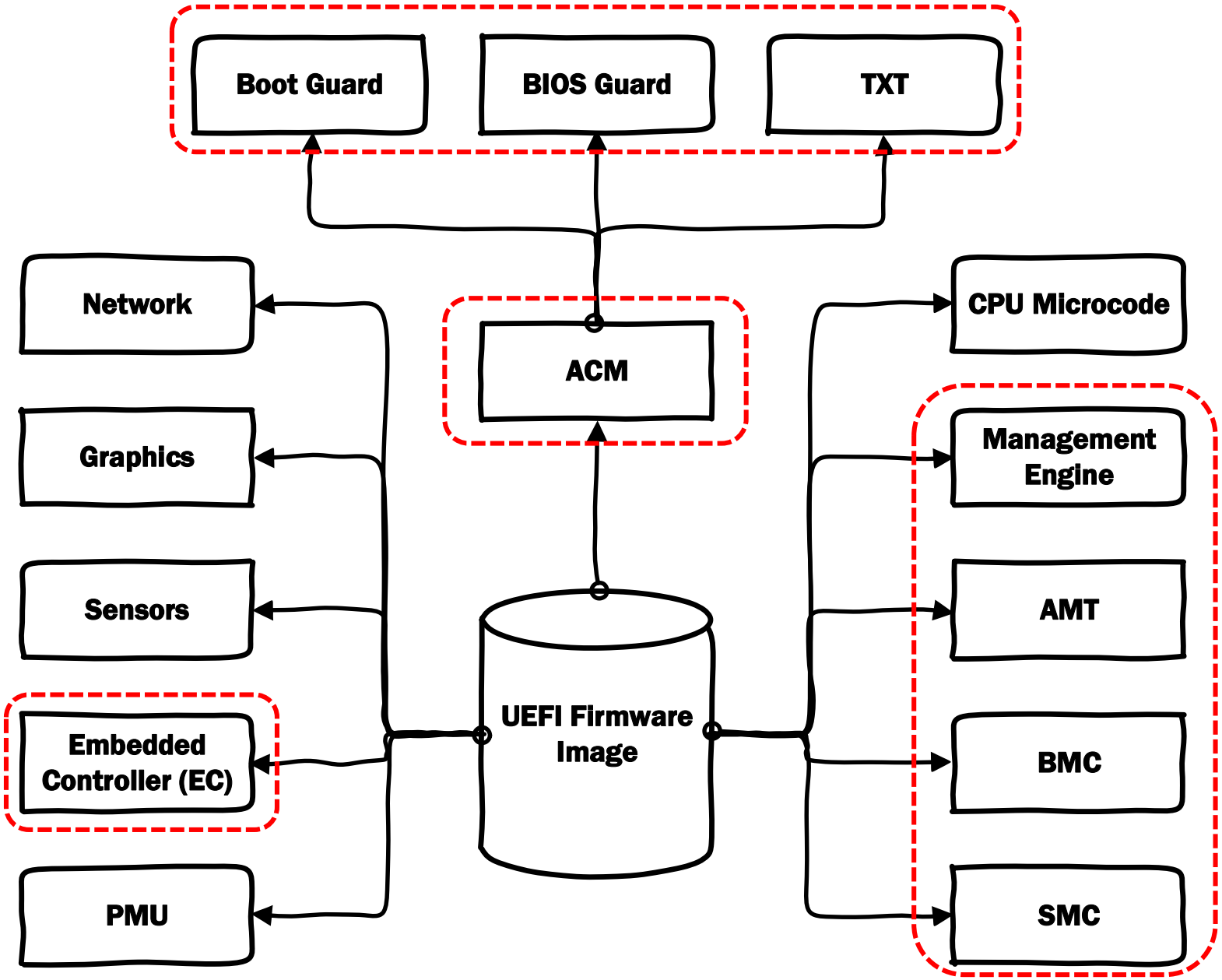
https://csrc.nist.gov/CSRC/media/Publications/sp/800-193/draft/documents/sp800-193-draft.pdf
https://cloudplatform.googleblog.com/2017/08/Titan-in-depth-security-in-plaintext.html
https://azure.microsoft.com/en-au/blog/microsofts-project-olympus-delivers-cloud-hardware-innovation-at-scale/
https://github.com/opencomputeproject/Project_Olympus/tree/master/Project_Cerberus
http://www.microchip.com/design-centers/security-ics/cryptoauthentication/cloud-authentication/
https://github.com/MicrochipTech/AWS-Secure-Insight

Breaking AMI BIOS Updates

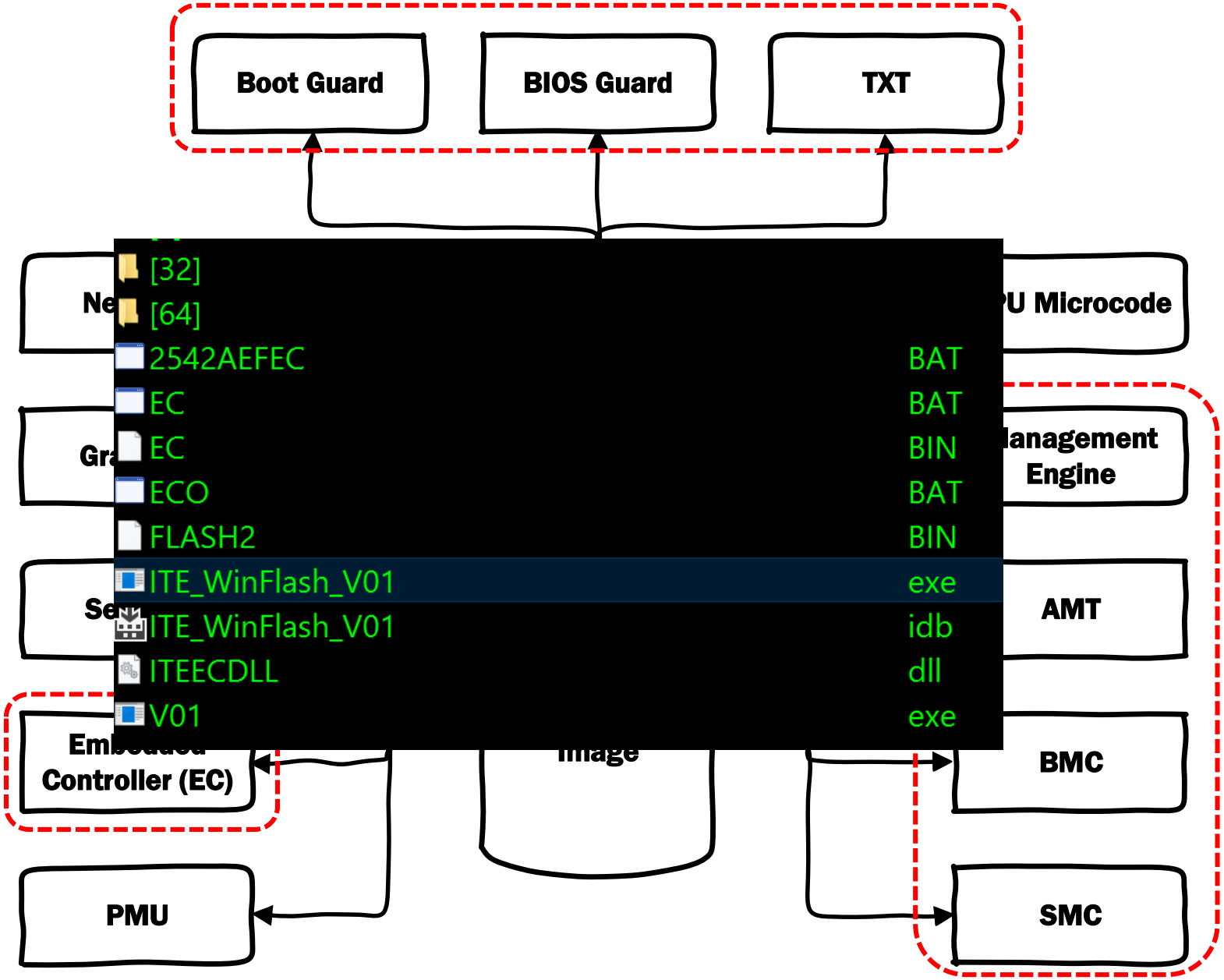
No more legacy! UEFI is everywhere!!

Now the legacy inside UEFI :-)

How many different firmware's inside BIOS update?



How many different firmware's inside BIOS update?



How many dif update?

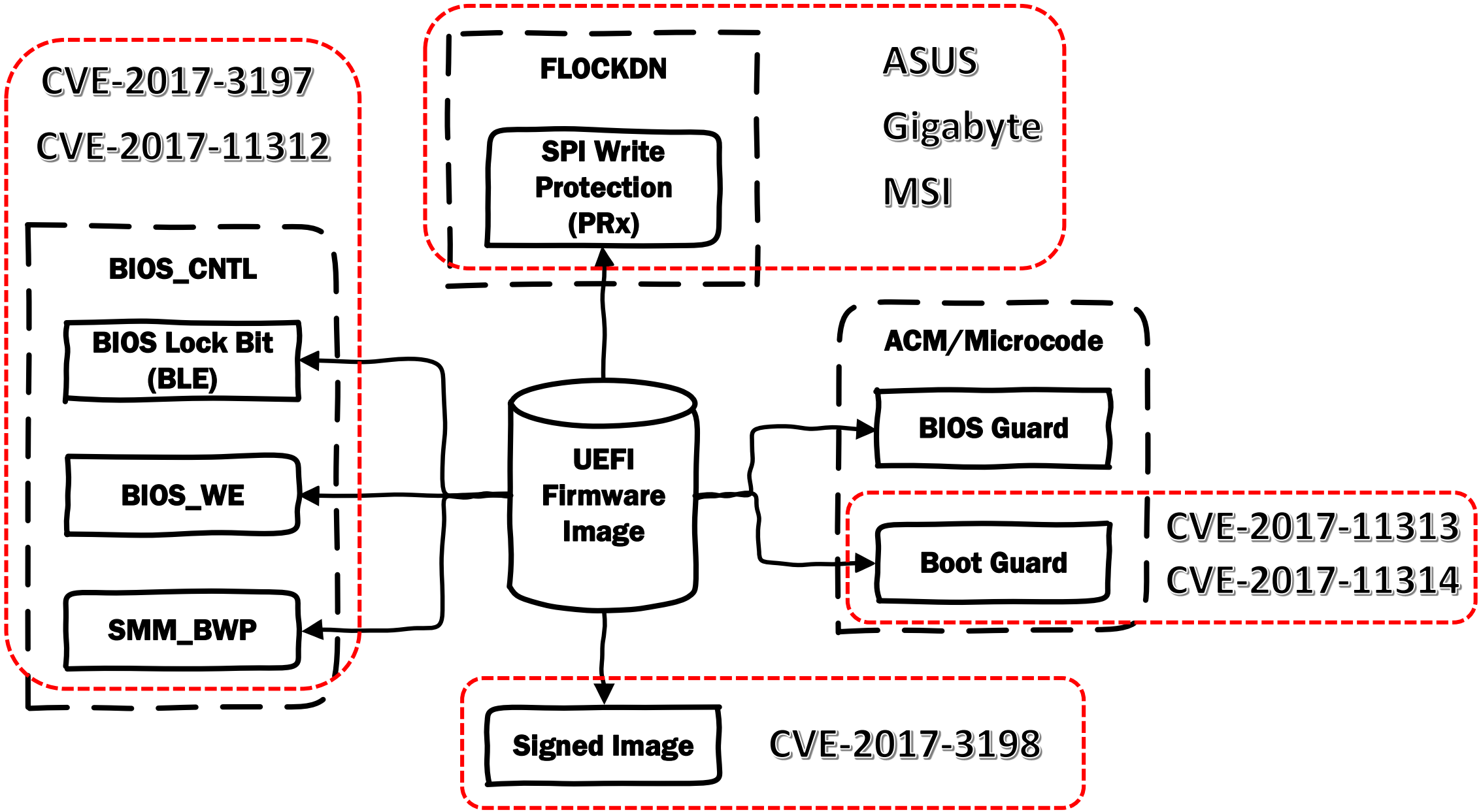
```
if ( dword_410FEC )
{
  write_port(dword_410FF4);
  printf("Send Erase Command...\n");
}
Sleep(0x64u);
printf("Erase Done\n");
if ( sub_401170() )
{
  printf("Return from Erase Checking: Done\n");
  if ( !dword_410FEC )
  {
    printf("Send Erase Command Again\n");
    write_port(dword_410FF4);
    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;
}
}
```

Never underestimate

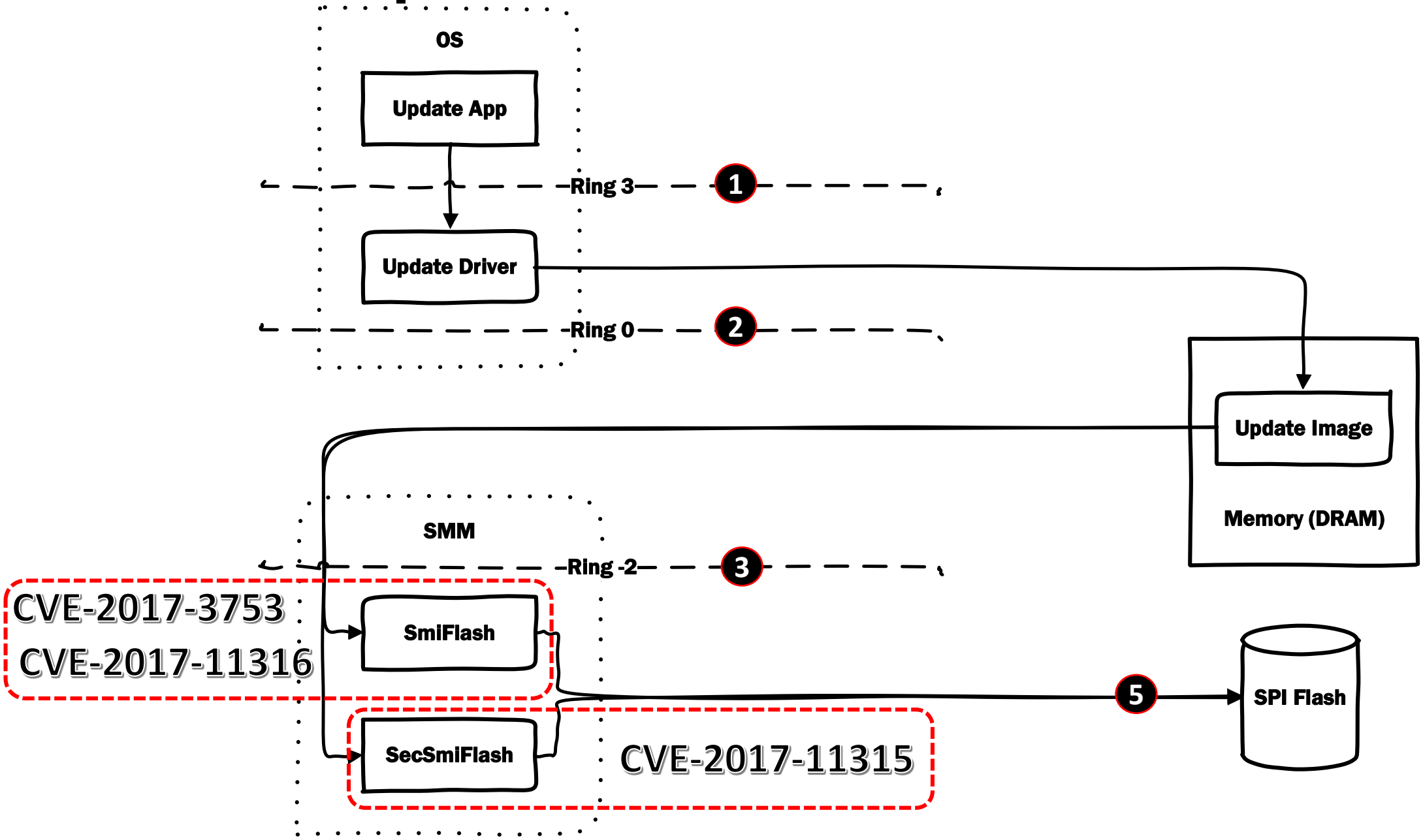
in your Threat Model!!

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

All Guardians of the BIOS on one slide



How BIOS Update Guardians Fail?



SMM SMI handlers extracted from SRAM dump

SW SMI HANDLERS:
0xbd541110: SMI = 0xb0, addr = 0xbd537c28, image = NbSmi
0xbd542910: SMI = 0xbb, addr = 0xbd52ed04, image = SbRunSmm
0xbd542210: SMI = 0xa0, addr = 0xbd525ce4, image = AcpiModeEnable
0xbd542010: SMI = 0xa1, addr = 0xbd525dd0, image = AcpiModeEnable
0xbd524b10: SMI = 0x55, addr = 0xbd5114d0, image = SmramSaveInfoHandlerSmm
0xbd4e6a10: SMI = 0x43, addr = 0xbd4e5360, image = AhciInt13Smm
0xbd4e6810: SMI = 0x44, addr = 0xbd4e07bc, image = MicrocodeUpdate
0xbd4e6610: SMI = 0x41, addr = 0xbd4dc9b8, image = OA3_SMM
0xbd4e6510: SMI = 0xdf, addr = 0xbd4dab54, image = OA3_SMM
0xbd4e6410: SMI = 0xef, addr = 0xbd4d89e0, image = SmiVariable
0xbd4e6310: SMI = 0x90, addr = 0xbd4d42dc, image = BiosDataRecordSmi
0xbd4cec10: SMI = 0x61, addr = 0xbd4cfde0, image = CmosSmm
0xbd4ce510: SMI = 0x42, addr = 0xbd4c4cd0, image = NvmeSmm
0xbd4ce110: SMI = 0x26, addr = 0xbd4ac32c, image = Ofbd
0xbd497c10: SMI = 0x20, addr = 0xbd4929bc, image = SmiFlash
0xbd497b10: SMI = 0x21, addr = 0xbd4929bc, image = SmiFlash
0xbd497a10: SMI = 0x22, addr = 0xbd4929bc, image = SmiFlash
0xbd497910: SMI = 0x23, addr = 0xbd4929bc, image = SmiFlash
0xbd497810: SMI = 0x24, addr = 0xbd4929bc, image = SmiFlash
0xbd497710: SMI = 0x25, addr = 0xbd4929bc, image = SmiFlash
0xbd497410: SMI = 0x35, addr = 0xbd48fe24, image = TcgSmm
0xbd472f10: SMI = 0x31, addr = 0xbd474ca8, image = UsbRtSmm
0xbd472b10: SMI = 0xbf, addr = 0xbd46ea48, image = CrbSmi
0xbd472710: SMI = 0x01, addr = 0xbd46d5e0, image = PiSmmCommunicationSmm
0xbd472010: SMI = 0x50, addr = 0xbd4671d4, image = SmbiosDmiEdit

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()

SecSMIFlash Handler Issues: ASUS

➢ S 2017-11315

[3

➢

```
switch(Data)
    {
        // 0x1d Load Flash Image to memory
        case Load_Image:
    if(!GuidCmp(&PubKey.Guid, &Guid)) {                    f);
            Cert = PubKey.Blob;
            Size = PubKey.Size;
        } else
        if(!GuidCmp(&PubKey.Guid, &Guid)) {
            MemSet(&HashCert, sizeof(HashCert), 0);
            MemCpy(HashCert.SignedHash, PubKey.Blob, 32);  Buff);
            Cert = &HashCert;
            Size = 32;
        }
    }
            Status = SetPolicy((FUNC_BLOCK *)pCommBuff);
            break;
    }
```

That's why BIOS Guard created

SecSMIFlash Handler Issues: ASUS

- S 2017-11315
  [3

```
switch(Data)
    {
        // 0x1d Load Flash Image to memory
        case Load_Image:
            Status = LoadImage((FUNC_BLOCK *)pCommBuff);
            break;

        // 0x1e Get Flash Update Policy
        case Get_Policy:
            Status = GetPolicy((FLASH_POLICY *)pCommBuff);
            break;

        // 0x1f Set Flash method
        case Set_Policy:
            Status = SetPolicy((FUNC_BLOCK *)pCommBuff);
            break;
    }
```

-

SecSMIFlash Handler Issues: ASUS

➢ S

[3

➢

2017-11315

```
switch(Data)
    {
        // 0x1d Load Flash Image to memory
        case Load_Image:
    if(!GuidCmp(&PubKey.Guid, &Guid)) {
            Cert = PubKey.Blob;
            Size = PubKey.Size;
        } else
        if(!GuidCmp(&PubKey.Guid, &Guid)) {
            MemSet(&HashCert, sizeof(HashCert), 0);
            MemCpy(HashCert.SignedHash, PubKey.Blob, 32);
            Cert = &HashCert;
            Size = 32;
        }
    }
            Status = SetPolicy((FUNC_BLOCK *)pCommBuff);
            break;
    }
```

SecSMIFlash Handler Issues: ASUS

➢ S 2017-11315

[3

➢

```
switch(Data)
    {
        // 0x1d Load Flash Image to memory
        case Load_Image:
    if(!GuidCmp(&PubKey.Guid, &Guid)) {                    f);
            Cert = PubKey.Blob;
            Size = PubKey.Size;
        } else
        if(!GuidCmp(&PubKey.Guid, &Guid)) {
            MemSet(&HashCert, sizeof(HashCert), 0);
            MemCpy(HashCert.SignedHash, PubKey.Blob, 32);  Buff);
            Cert = &HashCert;
            Size = 32;
        }
    }
            Status = SetPolicy((FUNC_BLOCK *)pCommBuff);
            break;
    }
```

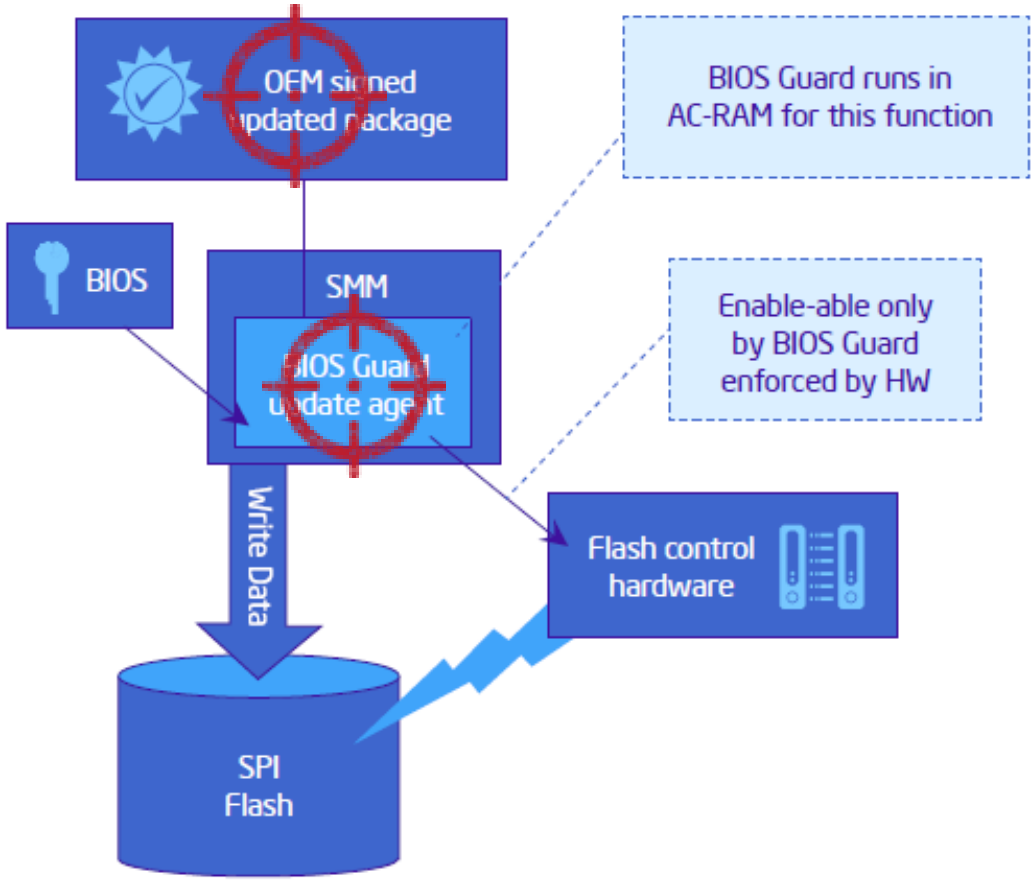
That's why BIOS Guard created

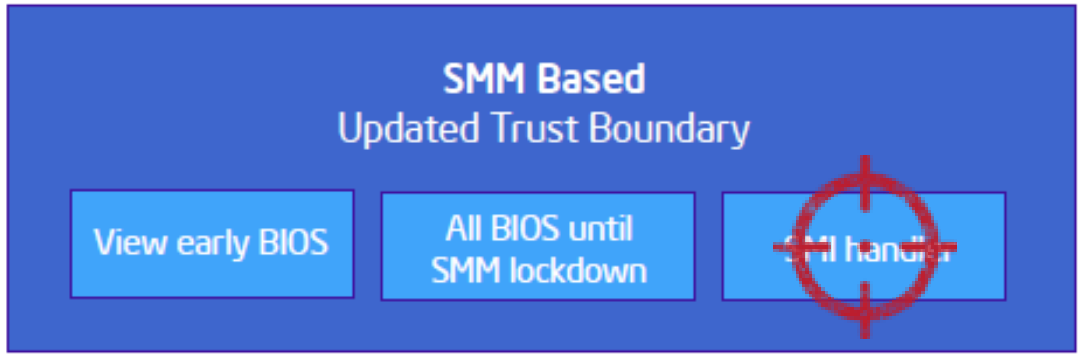
Intel BIOS Guard:
AMI Implementation

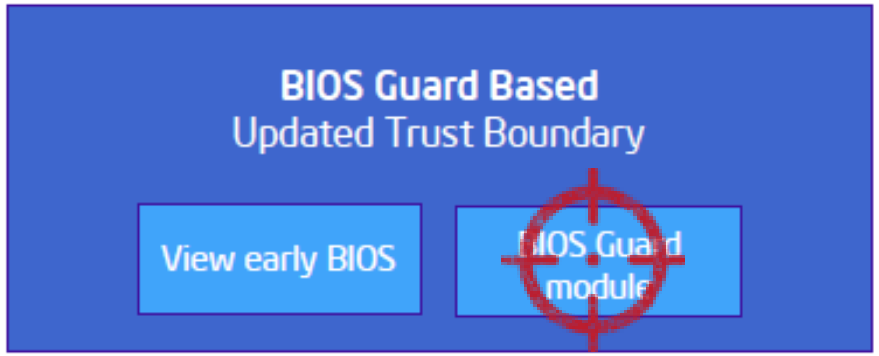
Intel BIOS Guard: Platform Flash Armoring technology (PFAT)

- **Armoring SPI Flash access**
  - ✓ Access controlled by BIOS Guard ACM
  - ✓ Partially implemented in Microcode, PCH, BIOS and EC
  - ✓ PCH locked SPI flash access without PFAT
- **BIOS update authentication**
  - ✓ Authenticated by BIOS Guard ACM
  - ✓
- **Gameover for malicious updates?**
  - ✓ Physical access + direct programming SPI flash still possible
  - ✓ Post update verification only rely on Intel Boot Guard integrity

Demystifying Intel BIOS Guard

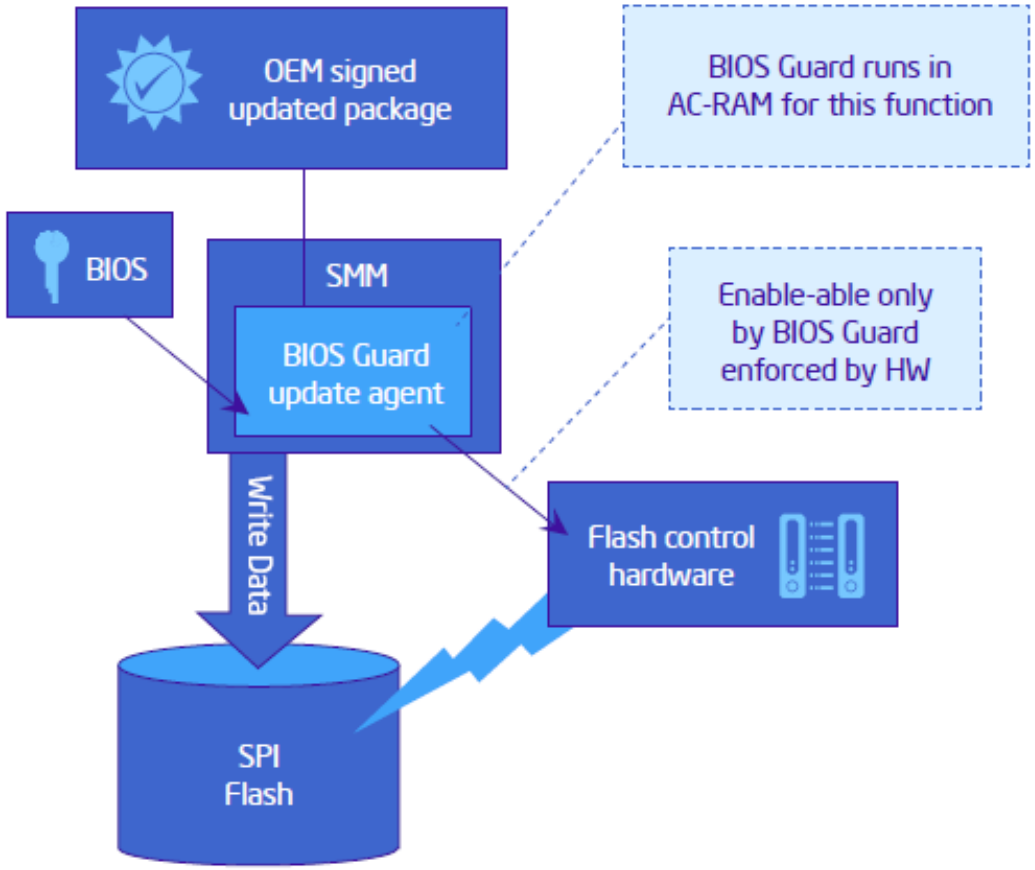


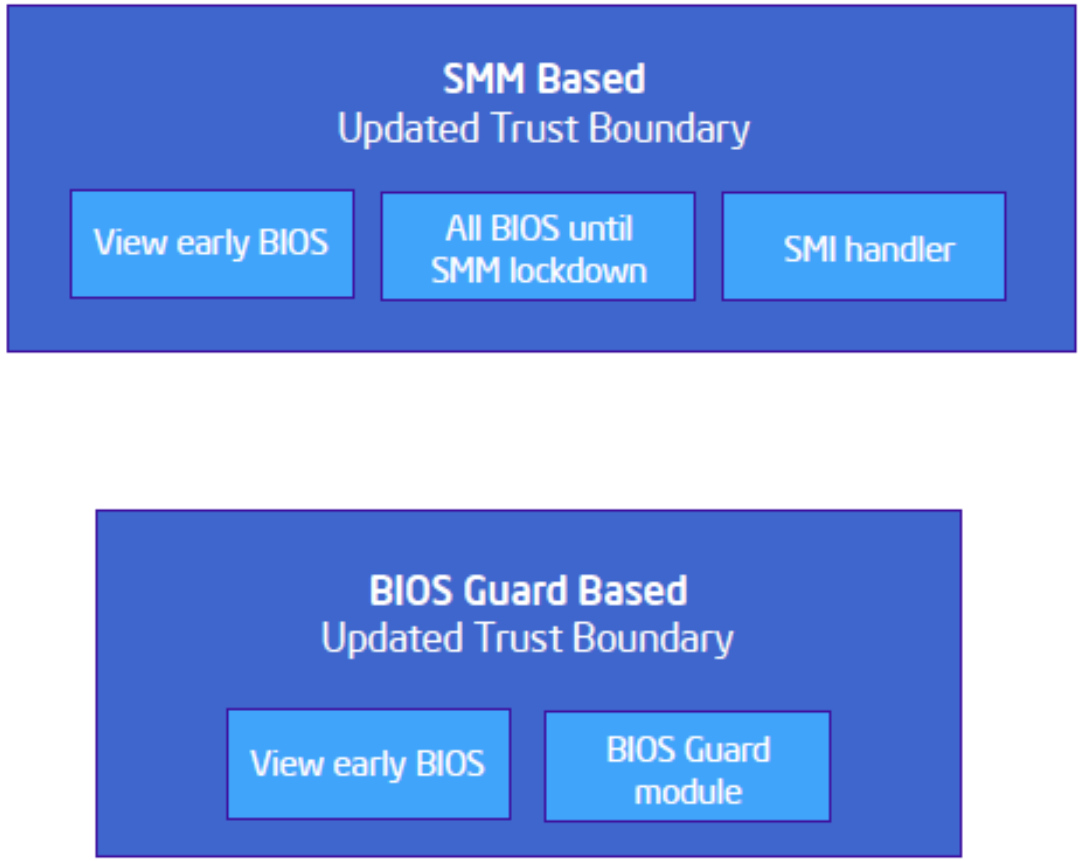




http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/security-technologies-4th-gen-core-retail-paper.pdf

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

AMI Boot Guard BIOS Components

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

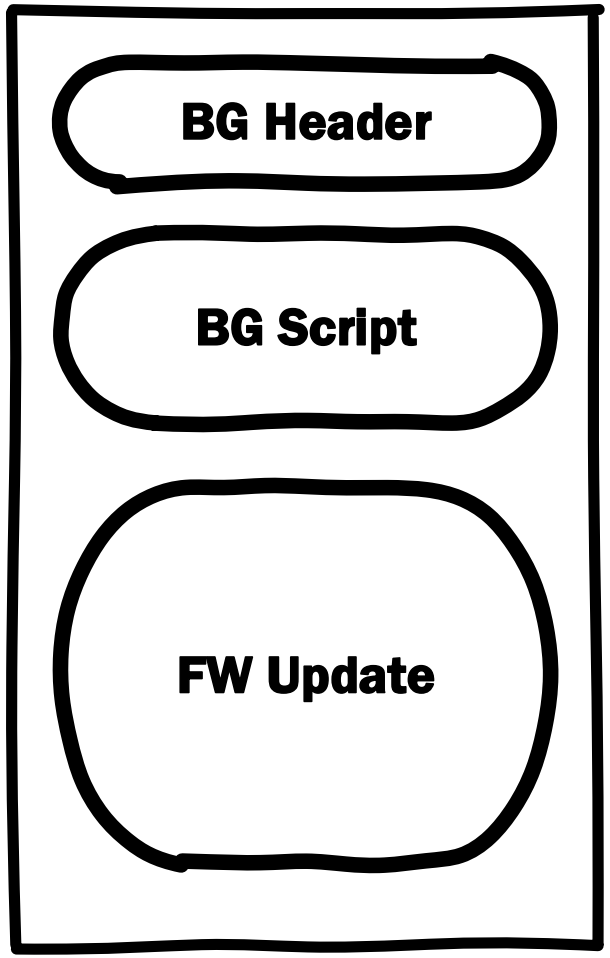
- **SMM**
  - **BiosGuardSmm** [44FE07D3-C312-4ad4-B892-269AB069C8E1]
  - **BiosGuardServices** [6D4BAA0B-F431-4370-AF19-99D6209239F6]
  - **BiosGuardFlashLib** [9a5768ad-497b-406e-a8ba-4d838389a6a1]

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

AMI BIOS Guard Artifacts if ENABLED

- Signatures is always good thing for RE
  - ✓ _AMIPFAT -
  - ✓ _AMIBSGD -
- UEFI Variable [BiosGuardCapsuleVariable]
- Intel BIOS Guard ACM
  - ✓ [7934156D-CFCE-460E-92F5-A07909A59ECA]

AMI BIOS Guard Update Format




```
// BIOS Guard update Package Header
typedef struct {
  UINT16 Version;
  UINT16 Unknown;
  UINT8  Unknown[16];
  UINT32 Unknown;
  UINT32 Unknown;
  UINT32 BgScriptSize;
  UINT32 UpdateSize;
  UINT32 FwSvn;
  UINT32 EcSvn;
  UINT32 Unknown;
} BGUP_HEADER;
```

Intel BIOS Guard Script

- Intel BIOS Guard Script
  - Turing complete language
  - Executed by ACM module
  - BiosGuardServices support BG Script execution
    - BIOSGUARD_PROTOCOL.Write()
    - BIOSGUARD_PROTOCOL.Erase()
    - BIOSGUARD_PROTOCOL.Execute()

  - BG Script Support operations with SPI Flash
    - BG_ERASE
    - BG_WRITE
    - BG_READ

    - Bytecode commands
      - Load/Store [Byte/Dword]
      - CMP, COPY, JMP
      - Conditional JMP
      - ReadKeyHash

AMI BIOS Guard Commands

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

- SMM
  - BG_READ
  - BG_WRITE
  - BG_ERASE

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

ACM Dirty Secrets

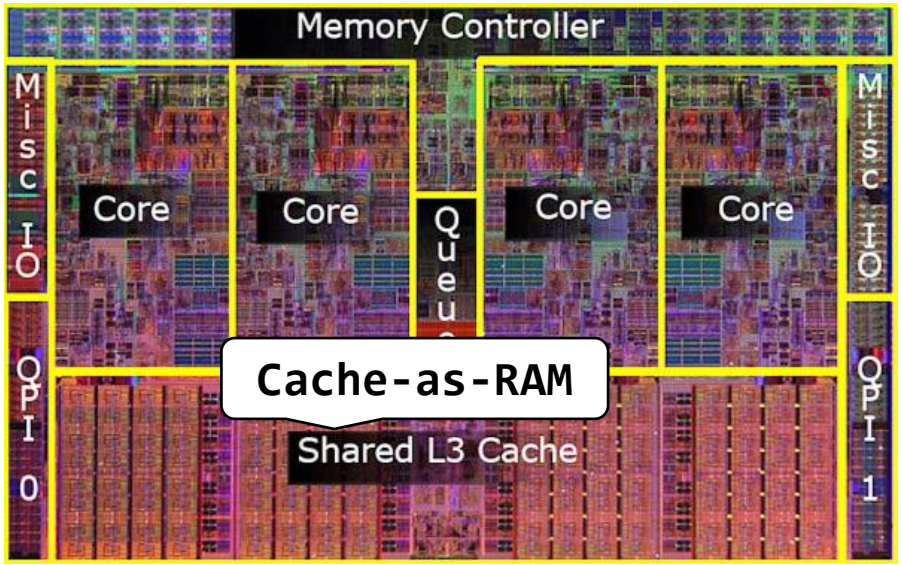
Authenticated Code Module (ACM)

- ACM is x86 (32-bit) code developed by Intel?
- ACM executes in AC-RAM (Cache-as-RAM or NEM)
- ACM can executes multiple times by request
- ACM has CPU microcode and Chipset specifics

```
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
```

ACM Runtime

- ACM executes in AC-RAM (Cache-as-RAM (CAR|NEM)
- Cache-as-RAM pre-DRAM memory on PEI phase
- Root of Trust locked in Intel Microcode
- Runtime is totally trusted no mitigations
- Developed on mix C and ASM => no static analysis

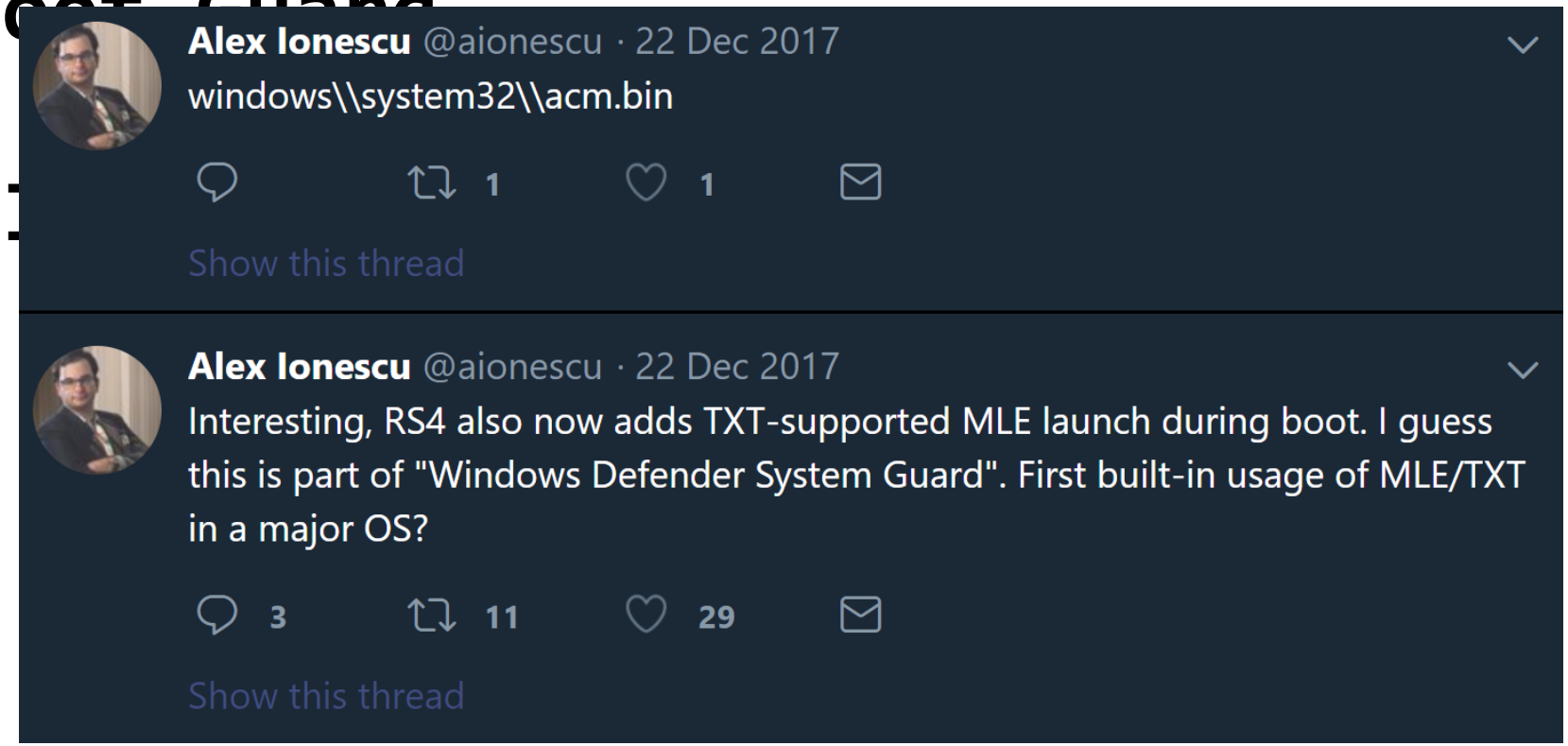


Known ACM modules

- Intel SINIT/SCLEAN (TXT related)
- Intel Boot Guard
- Intel BIOS Guard

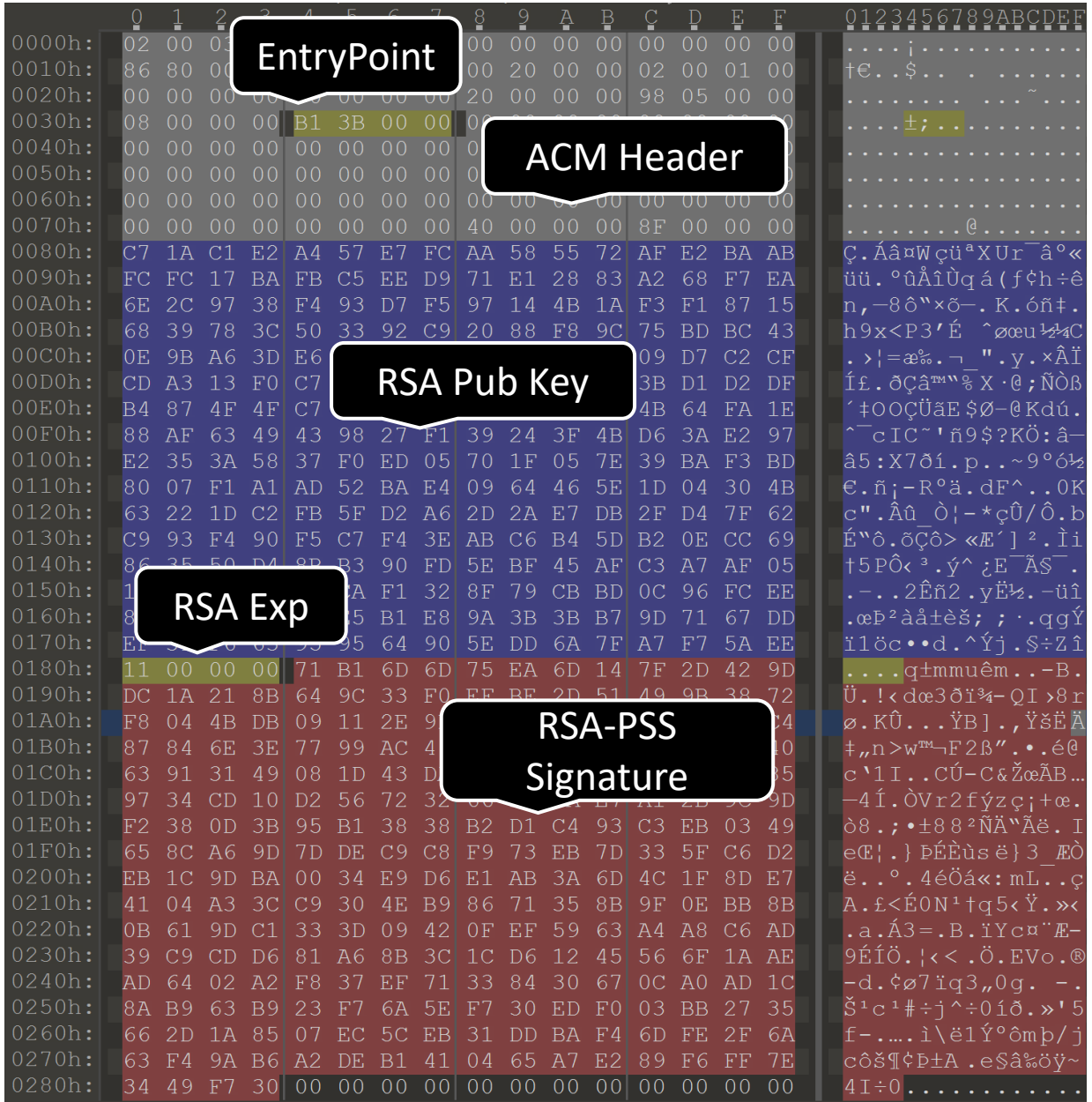
Known ACM modules

- Intel SINIT/SCLEAN (TXT related)
- Intel Boot Guard
- Intel BI



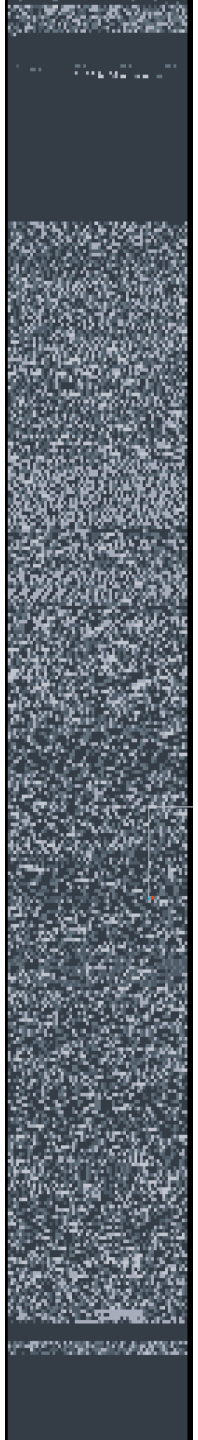
Boot Guard: Authenticated Code Module (ACM)

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]	



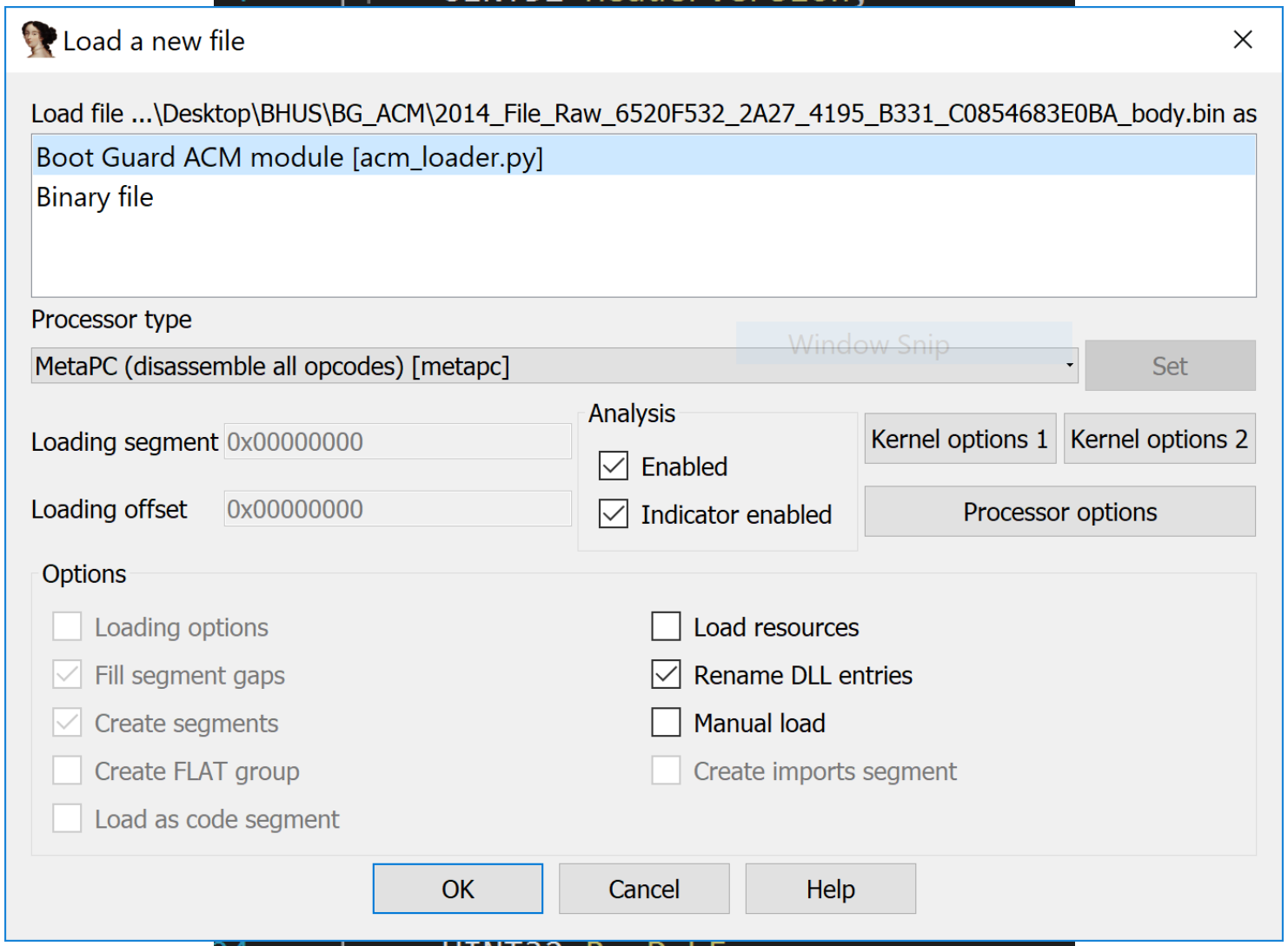
ACM Header

```
1   typedef struct INTEL_ACM_HEADER_ {
2       UINT32 ModuleType;
3       UINT32 HeaderType;
4       UINT32 HeaderVersion;
5       UINT16 ChipsetId;
6       UINT16 Flags;
7       UINT32 ModuleVendor;
8       UINT8  DateDay;
9       UINT8  DateMonth;
10      UINT16 DateYear;
11      UINT32 ModuleSize;
12      UINT16 AcmSvn;
13      UINT16 : 16;
14      UINT32 Unknown1;
15      UINT32 Unknown2;
16      UINT32 GdtMax;
17      UINT32 GdtBase;
18      UINT32 SegmentSel;
19      UINT32 EntryPoint;
20      UINT8  Unknown3[64];
21      UINT32 KeySize;
22      UINT32 Unknown4;
23      UINT8  RsaPubKey[256];
24      UINT32 RsaPubExp;
25      UINT8  RsaSig[256];
26  } INTEL_ACM_HEADER;
```



ACM Header

```
1  typedef struct INTEL_ACM_HEADER_ {
2      UINT32 ModuleType;
3      UINT32 HeaderType;
4      UINT32 HeaderVersion;
```




```
24     UINT32 RsaPubExp;
25     UINT8  RsaSig[256];
26 } INTEL_ACM_HEADER;
```

Known ACM Vulnerabilities

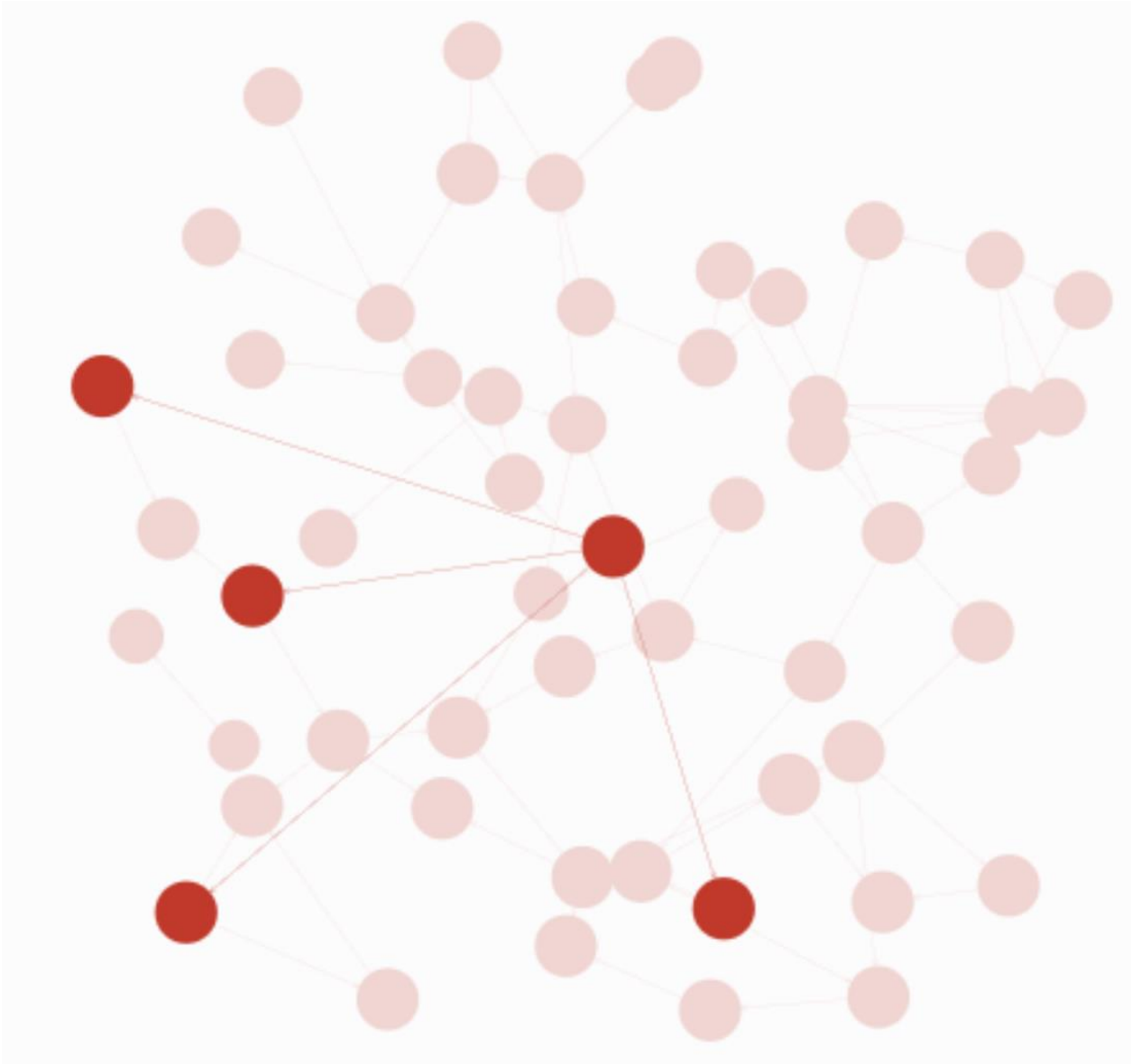
- INTEL-SA-00021 (2009)- SINIT ACM Privilege Escalation
- INTEL-SA-00030 (2011)- SINIT ACM Privilege Escalation
- INTEL-SA-00035 (2013)- SINIT ACM Privilege Escalation

http://invisiblethingslab.com/resources/2011/Attacking_Intel_TXT_via_SINIT_hijacking.pdf
https://software.intel.com/en-us/articles/intel-trusted-execution-technology/

Boot Guard: Authenticated Code Module (ACM)

- ACM is x86 (32-bit) code developed by Intel
- ACM executes in AC-RAM (Cache-as-RAM or NEM)
- ACM has CPU and Chipset specifics
- ACM verifies Key Manifest (KEYM) + IBB (IBBM)

Boot Guard: Authenticated Code Module (ACM)



```
entry_point proc near
mov     ax, ds
mov     ss, ax
mov     es, ax
mov     fs, ax
mov     gs, ax
mov     esp, ebp
add     esp, 1000h
mov     eax, ebp
add     eax, 4C8h
lidt    fword ptr [eax]
push    ebp
call    boot_guard
mov     ebx, eax
mov     edx, 0
mov     eax, 3
getsec
```

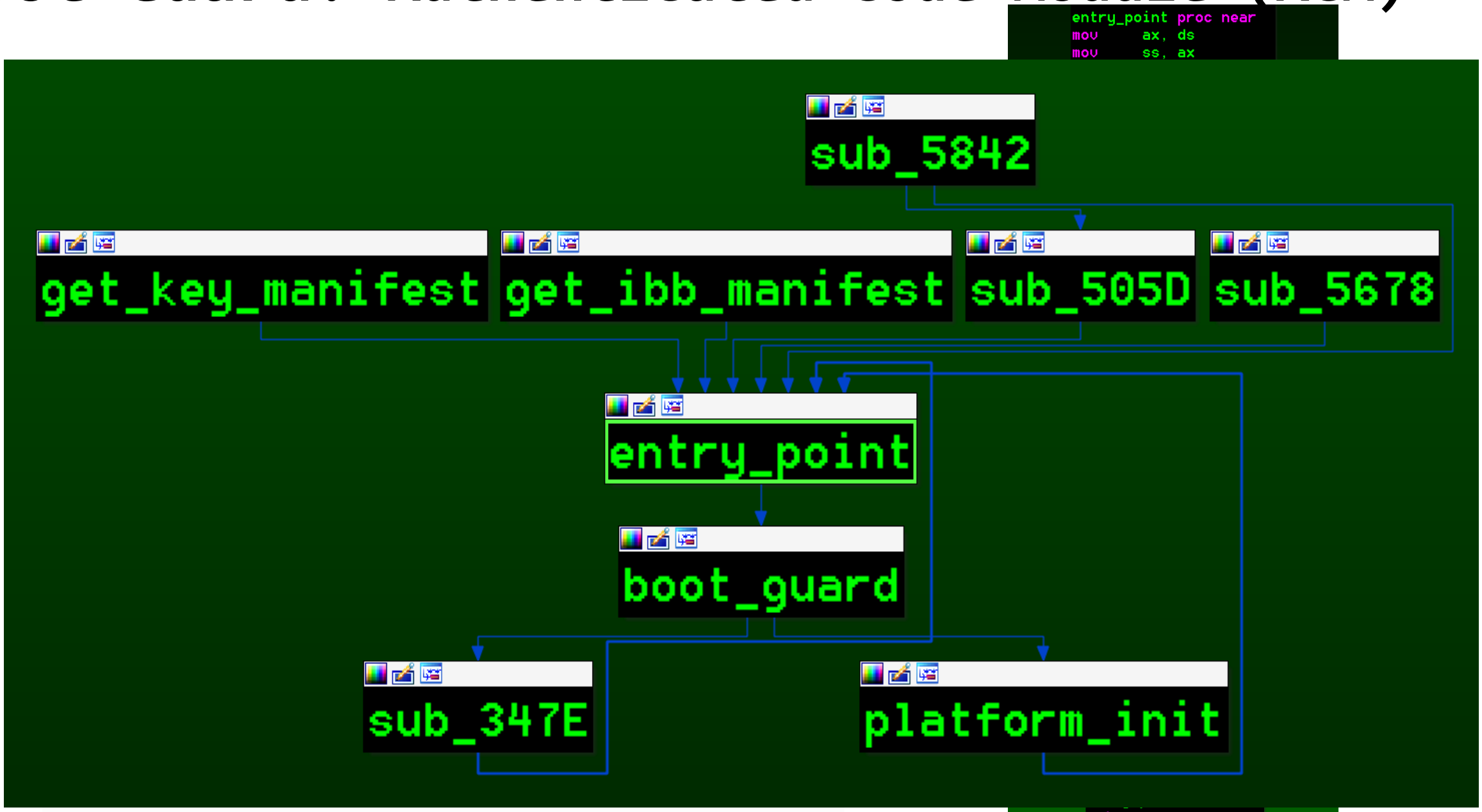
```
loc_3BE6:
push    ebp
mov     ebp, esp
cmp     dword ptr [ebp+14h], 0
mov     eax, [ebp+8]
jz      short loc_3C06
```

```
mov     ecx, [ebp+10h]
sub     ecx, eax
```

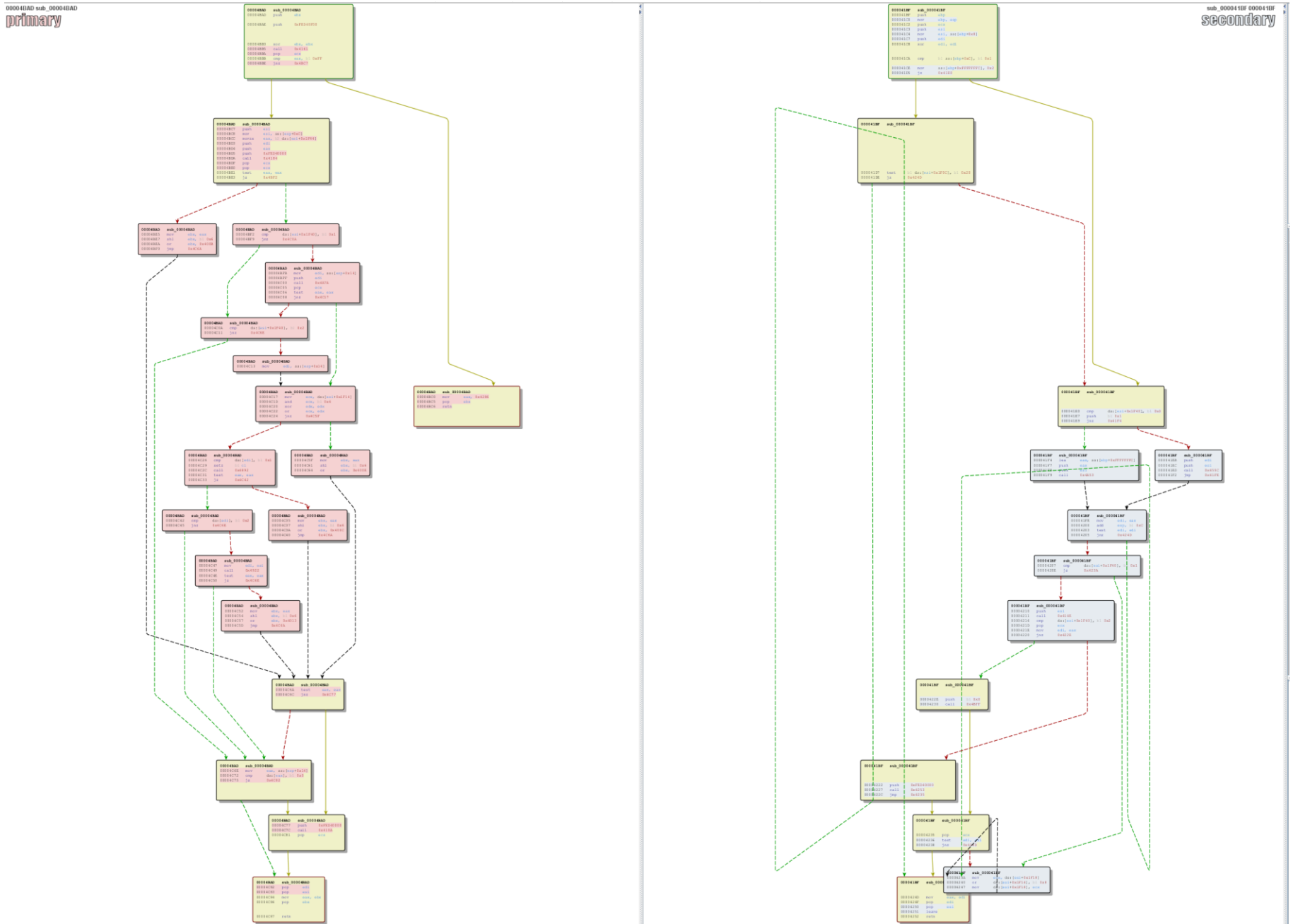
```
loc_3BF7:
mov     dl, [ecx+eax]
dec     dword ptr [ebp+14h]
mov     [eax], dl
inc     eax
cmp     dword ptr [ebp+14h], 0
jnz     short loc_3BF7
```

```
loc_3C06:
pop     ebp

public entry_point_1
entry_point_1:
retn
entry_point endp
```

Boot Guard: Authenticated Code Module (ACM)



Boot Guard ACM BinDiff: Broadwell vs Skylake



Intel Boot Guard:
New Ways to Bypass

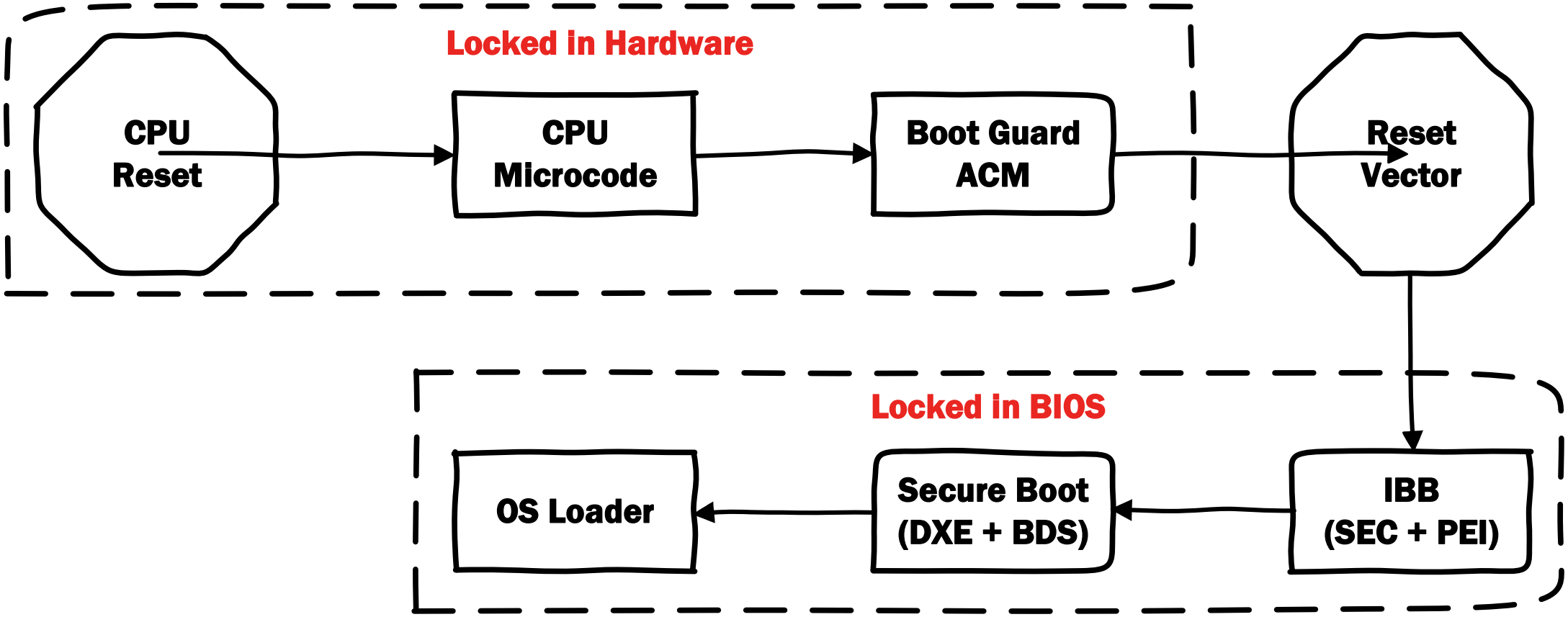
Different shades of Secure Boot

- **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

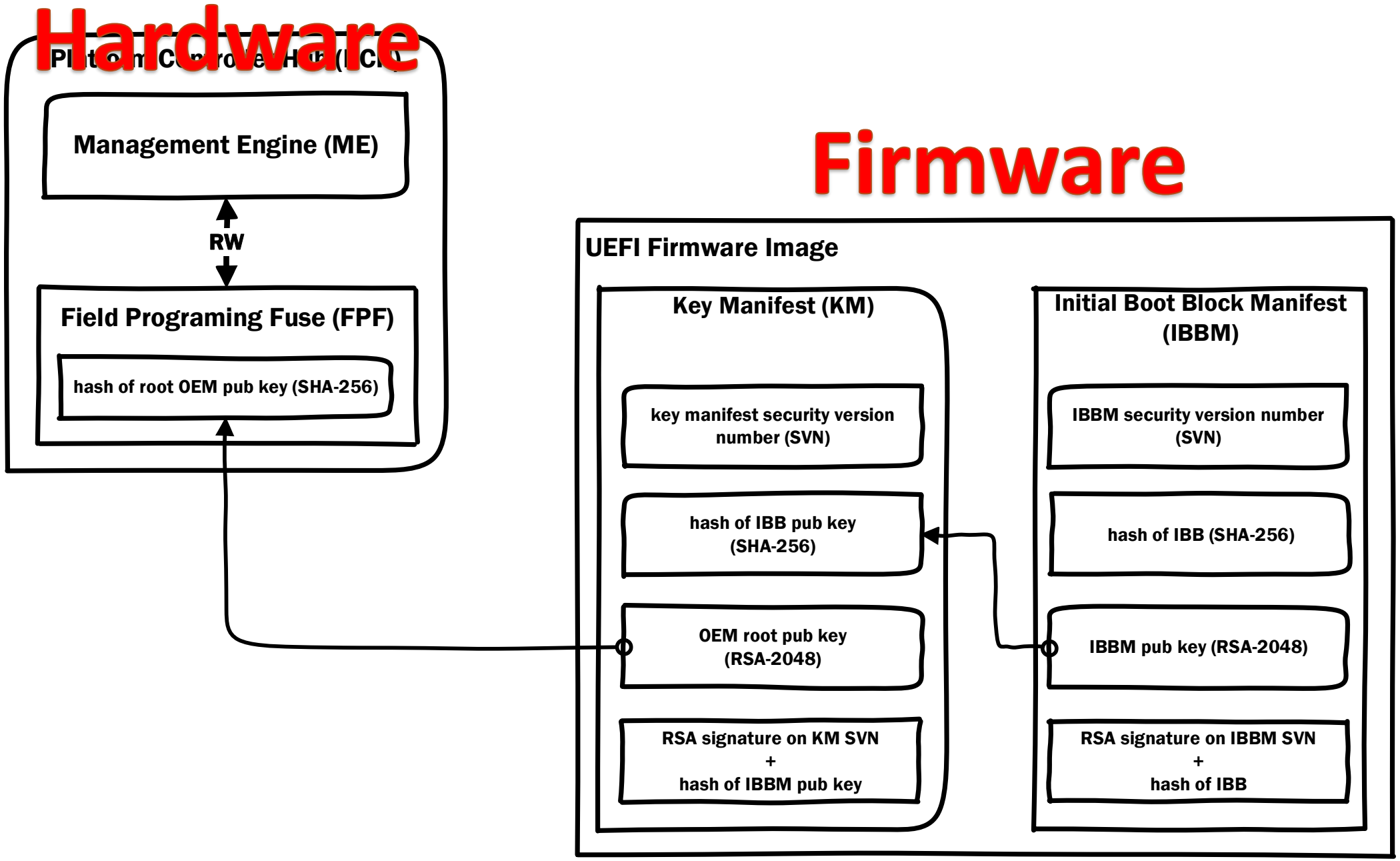
Intel Boot Guard operating modes

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

Boot Guard: Boot Flow



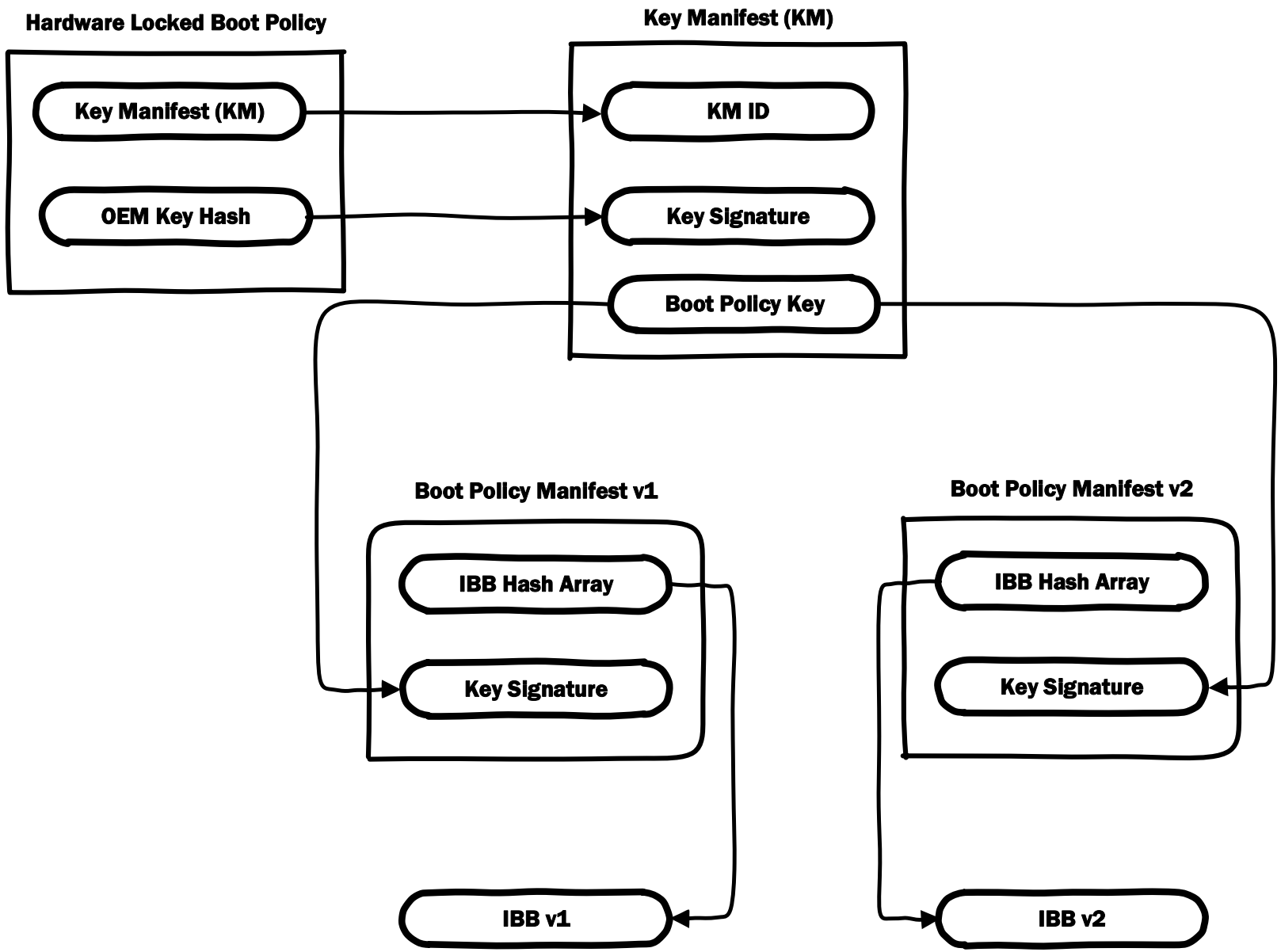
Boot Guard: Chain of Trust



VerifyFwBootGuard SMM Validation Flow (Intel ME communications over HECI)

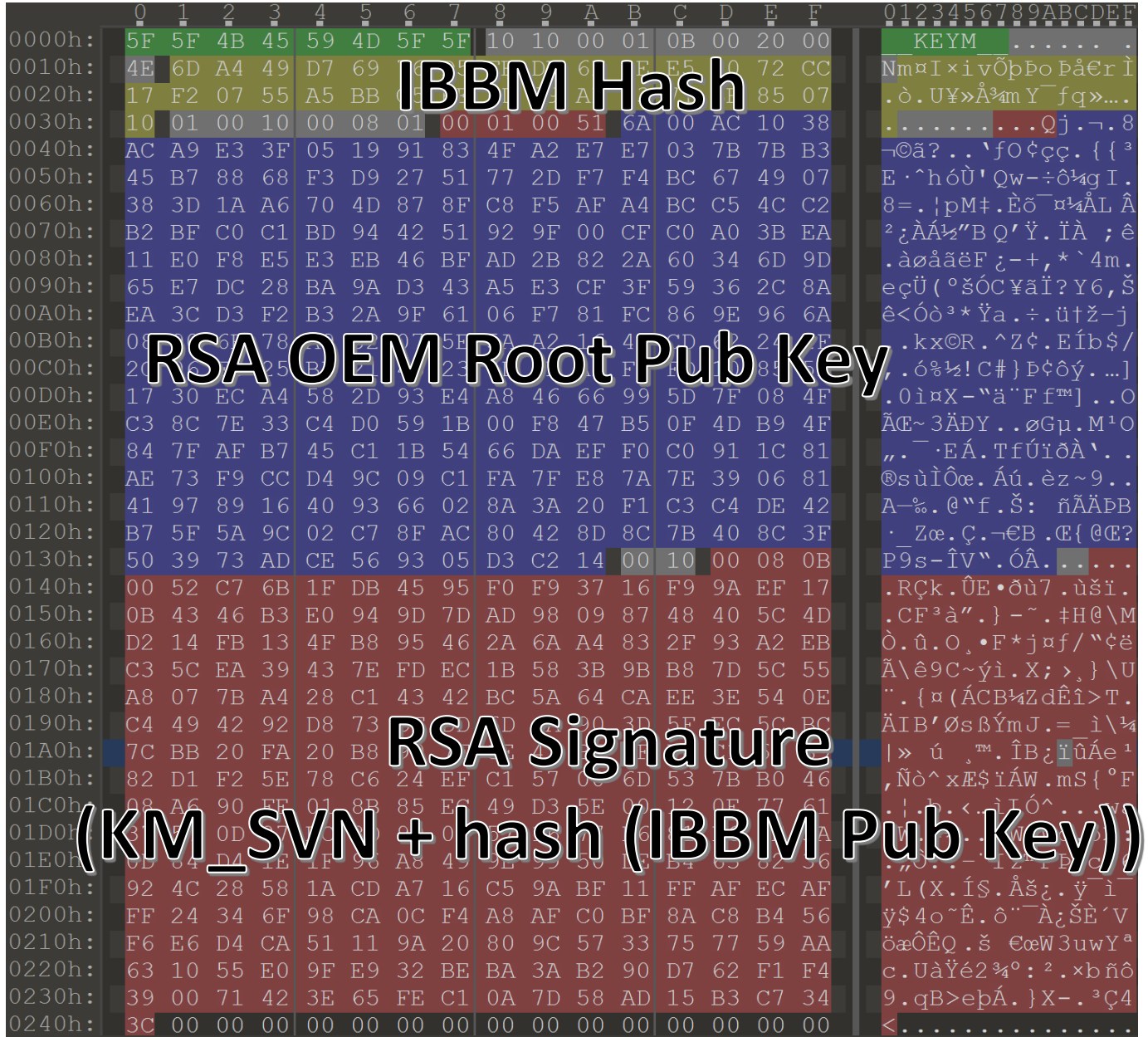
- Find and Verify ACM
  - Verify ACM SVN
- Find and Verify Key Manifest (KM)
  - Verify KM SVN
- Find and Verify Boot Policy Manifest (BPM)
  - Verify BPM SVN
- If something wrong return EFI_SECURITY_VIOLATION

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: PEI FV_HASH

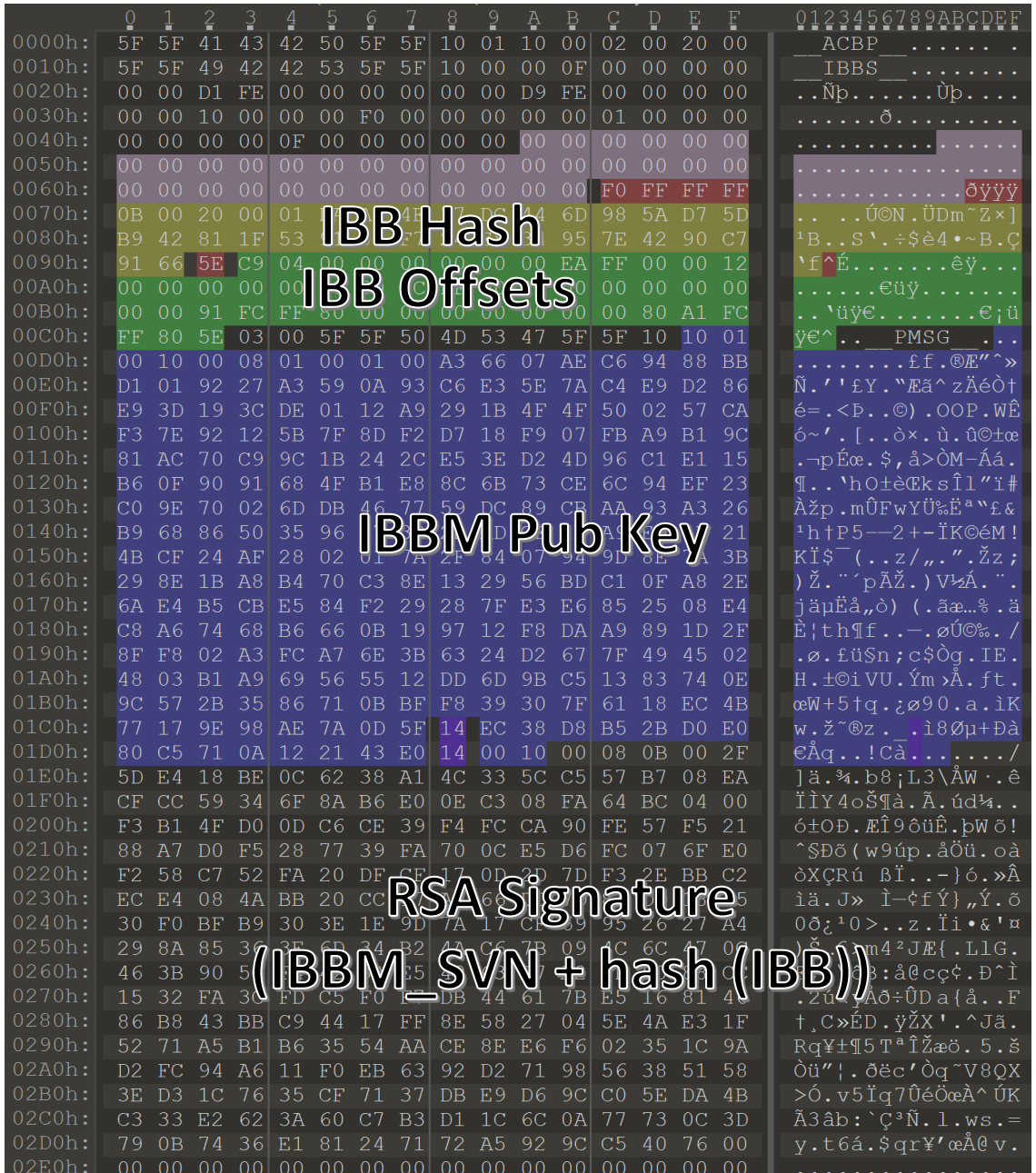
- **FV_HASH_KEY [CBC91F44-A4BC-4A5B-8696-703451D0B053]**

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    0123456789ABCDEF
0000h: 30 B8 5A 2D C7 98 95 B6 05 0C 28 84 0C D2 40 9E    0¸Z-Ç˜•¶..(„.Ò@ž
0010h: 77 20 ED A0 97 97 DB 9A FD 69 51 80 3C 18 29 7D    w í ——ÛšýiQ€<.)}
0020h: 00 00 A5 FF A4 00 00 00 D0 41 10 C6 02 B0 4D 9F    ..¥ÿ¤...ÐA.Æ.°MŸ
0030h: 76 43 3F BB 56 A6 D4 70 F0 D5 E8 0E 43 4D 65 31    vC?»V¦ÔpðÕè.CMe1
0040h: 7A DF BD A5 2A 03 EB 44 A4 10 A5 FF 5C 73 25 00    zß½¥*.ëD¤.¥ÿ\s%.
0050h:
```

```
struct BOOT_GUARD_MAIN_HASH_KEY HK
    UBYTE HashKey0[32]
    UINT32 SegmentBase0
    UINT32 SegmentSize0
    UBYTE HashKey1[32]
    UINT32 SegmentBase1
    UINT32 SegmentSize1
```

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



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

Grugq

BootGuardDxe Validation Flow

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

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

    ...

    BootMode  = GetBootMode();
    if ( (BootMode == BOOT_IN_RECOVERY_MODE) || (BootMode == BOOT_ON_FLASH_UPDATE) ) {
        return   EFI_SUCCESS;
    }

    ...

    if ( BootGuardVerifyTransitionPEItoDXEFlag == 0 ) {
        BootGuardRegisterCallBack();
    }

    return   EFI_SUCCESS;
}
```

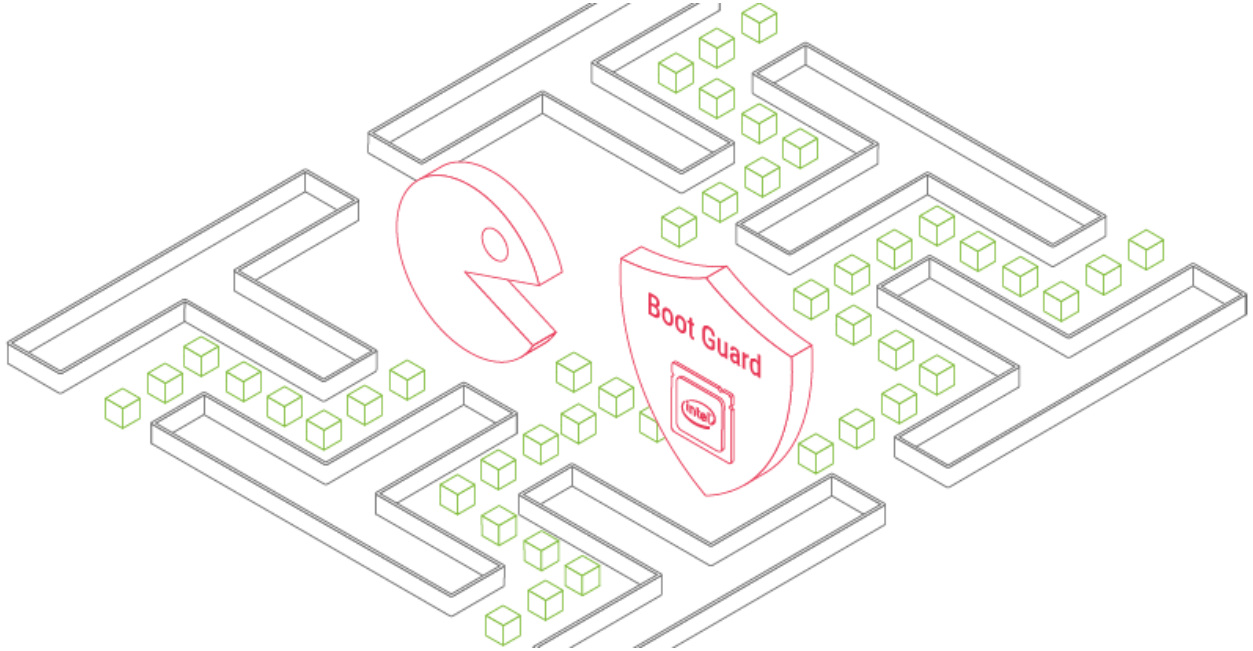
← one more 0-day bug?

Always Follow Security Error Messages

```
loc_44A:                                ; CODE XREF: BootGuardCheck+CE↓j
                cmp     cx, 4
                jz      short loc_463

loc_450:                                ; CODE XREF: BootGuardCheck+DC↓j
                                        ; BootGuardCheck+E2↓j
                movzx   ecx, [rax+EFI_HOB_GENERIC_HEADER.HobLength]
                add     rax, rcx

loc_457:                                ; CODE XREF: BootGuardCheck+B8↑j
                movzx   ecx, [rax+EFI_HOB_GENERIC_HEADER.HobType]
                cmp     cx, r9w
                jnz     short loc_44A
                mov     rax, rbx

loc_463:                                ; CODE XREF: BootGuardCheck+BE↑j
                cmp     rax, rbx
                jz      short ReturnError
                cmp     r8, qword ptr [rax+EFI_HOB_GUID_TYPE.Name.Data1]
                jnz     short loc_450
                cmp     rdx, qword ptr [rax+EFI_HOB_GUID_TYPE.Name.Data4]
                jnz     short loc_450
                cmp     rax, rbx
                jz      short ReturnError
                cmp     [rax+(size EFI_HOB_GUID_TYPE)], bl ; VerificationResult
                jnz     short exit
                lea     rax, [rsp+48h+buffer]
                xor     r9d, r9d        ; NotifyContext
                lea     r8, PrintFailMessageAndResetSystem ; NotifyFunction
                mov     [rsp+48h+var_28], rax
                mov     rax, cs:gEfiBootServices
                lea     edx, [r9+8]     ; NotifyTpl
                mov     ecx, 200h       ; Type
                call    [rax+EFI_BOOT_SERVICES.CreateEvent]
                cmp     rax, rbx
                jl      short exit
                mov     rax, cs:gEfiBootServices
                mov     rdx, [rsp+48h+buffer] ; Event
                lea     r8, [rsp+48h+Registration] ; Registration
                lea     rcx, gBdsAllDriversConnectedProtocolGuid ; Protocol
                call    [rax+EFI_BOOT_SERVICES.RegisterProtocolNotify]
                jmp     short exit
; ---------------------------------------------------------------------------
ReturnError:                            ; CODE XREF: BootGuardCheck+D6↑j
                                        ; BootGuardCheck+E7↑j
                mov     rbx, 800000000000000Eh
                jmp     short exit
; ---------------------------------------------------------------------------
ReturnOk:                               ; CODE XREF: BootGuardCheck+65↑j
                                        ; BootGuardCheck+85↑j ...
                xor     ebx, ebx

exit:                                   ; CODE XREF: BootGuardCheck+EC↑j
                                        ; BootGuardCheck+118↑j ...
                mov     rax, rbx
                add     rsp, 40h
                pop     rbx
                retn
BootGuardCheck  endp
```

```
; void __cdecl PrintFailMessageAndResetSystem(EFI_EVENT Event, void *Context)
PrintFailMessageAndResetSystem proc near
                                        ; DATA XREF: BootGuardCheck+F6↑o

arg_10          = byte ptr  18h
arg_18          = qword ptr  20h

                push    rbx
                sub     rsp, 20h
                mov     rax, cs:gEfiSystemTable
                mov     rbx, rcx
                mov     rdx, [rax+EFI_SYSTEM_TABLE.ConOut]
                mov     rcx, rdx
                call    qword ptr [rdx+30h]
                mov     r11, cs:gEfiSystemTable
                lea     rdx, aBootGuardVerif ; "Boot Guard verified DXE that is fail\n\"
                mov     rax, [r11+EFI_SYSTEM_TABLE.ConOut]
                mov     rcx, rax
                call    qword ptr [rax+8]
                mov     r11, cs:gEfiSystemTable
                lea     rdx, aSystemWillShut ; "System will shutdown\n\r"
                mov     rax, [r11+EFI_SYSTEM_TABLE.ConOut]
                mov     rcx, rax
                call    qword ptr [rax+8]
                mov     r11, cs:gEfiSystemTable
                lea     rdx, aPressAnyKey ; "Press any key\n\r"
                mov     rax, [r11+EFI_SYSTEM_TABLE.ConOut]
                mov     rcx, rax
                call    qword ptr [rax+8]

GetChar:                                ; CODE XREF: PrintFailMessageAndResetSystem+7C↓j
                mov     rax, cs:gEfiSystemTable
                lea     rdx, [rsp+28h+arg_10]
                mov     r8, [rax+EFI_SYSTEM_TABLE.ConIn]
                mov     rcx, r8
                call    qword ptr [r8+8]
                test    rax, rax
                jnz     short GetChar
                lea     ecx, [rax+2]    ; ResetType
                mov     rax, cs:gEfiRuntimeServices
                xor     r9d, r9d        ; ResetData
                xor     r8d, r8d        ; DataSize
                xor     edx, edx        ; ResetStatus
                call    [rax+EFI_RUNTIME_SERVICES.ResetSystem]
                mov     [rsp+28h+arg_18], 0

loc_580:                                ; CODE XREF: PrintFailMessageAndResetSystem+A4↓j
                mov     rax, [rsp+28h+arg_18]
                test    rax, rax
                jz      short loc_580
                mov     rax, cs:gEfiBootServices
                mov     rcx, rbx        ; Event
                call    [rax+EFI_BOOT_SERVICES.CloseEvent]
                add     rsp, 20h
                pop     rbx
                retn
PrintFailMessageAndResetSystem endp
```

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

TRUST
NO
ONE

```
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:                          F1
MEBx Version:                          10.0.0.0007
Gbe Version:                           Unknown
VendorID:                              8086
PCH Version:                           5
FW Version:                            9.1.20.1035
LMS Version:                           Not Availab
MEI Driver Version:                    11.0.0.1157

FW Capabilities:                       0x4910196C

    Intel(R) Small Business Technology - PRESENT/EN
    Intel(R) Anti-Theft Technology - PRESENT/ENABLE
    Intel(R) Capability Licensing Service - PRESENT
    Protect Audio Video Path - PRESENT/ENABLED
    Intel(R) Dynamic Application Loader - PRESENT/E
    Service Advertisement & Discovery - PRESENT/ENA

TLS:                                   Disabled
Last ME reset reason:                  Power up
Local FWUpdate:                        Enabled
BIOS Config Lock:                      Enabled
Host Read Access to ME:                Enabled
Host Write Access to ME:               Enabled
SPI Flash ID #1:                       C22018
SPI Flash ID VSCC #1:                  20452045
SPI Flash BIOS VSCC:                   20452045
BIOS boot State:                       Post Boot
OEM Id:                                00000000-0000-0000-0000-000000000000
Capability Licensing Service:          Enabled
OEM Tag:                               0x00000000
Slot 1 Board Manufacturer:             Unused
Slot 2 System Assembler:               Unused
Slot 3 Reserved:                       Unused
M3 Autotest:                           Disabled
Localized Language:                    English
Independent Firmware Recovery:         Enabled
```

```
EPID Group ID                          0xF9C
LSPCON Ports                           None
5K Ports                               None
OEM Public Key Hash FPF                Not set
OEM Public Key Hash ME                 EE7DB69F8B18F541F6467089E8A4A0388EA0E259284CC42CAD7CEA3AF3BA7260
ACM SVN FPF                            0x2
KM SVN FPF                             0x0
BSMM SVN FPF                           0x0
GuC Encryption Key FPF                 Not set
GuC Encryption Key ME                  0000000000000000000000000000000000000000000000000000000000000000

                                       FPF                     ME
                                       ---                     --
Force Boot Guard ACM                   Not set                 Enabled
Protect BIOS Environment               Not set                 Enabled
CPU Debugging                          Not set                 Enabled
BSP Initialization                     Not set                 Enabled
Measured Boot                          Not set                 Enabled
Verified Boot                          Not set                 Enabled
Key Manifest ID                        Not set                 0x1
Enforcement Policy                     Not set                 0x3
PTT                                    Not set                 Enabled
EK Revoke State                        Not Revoked
PTT RTC Clear Detection FPF            Not set
```

CVE-2017-11314)

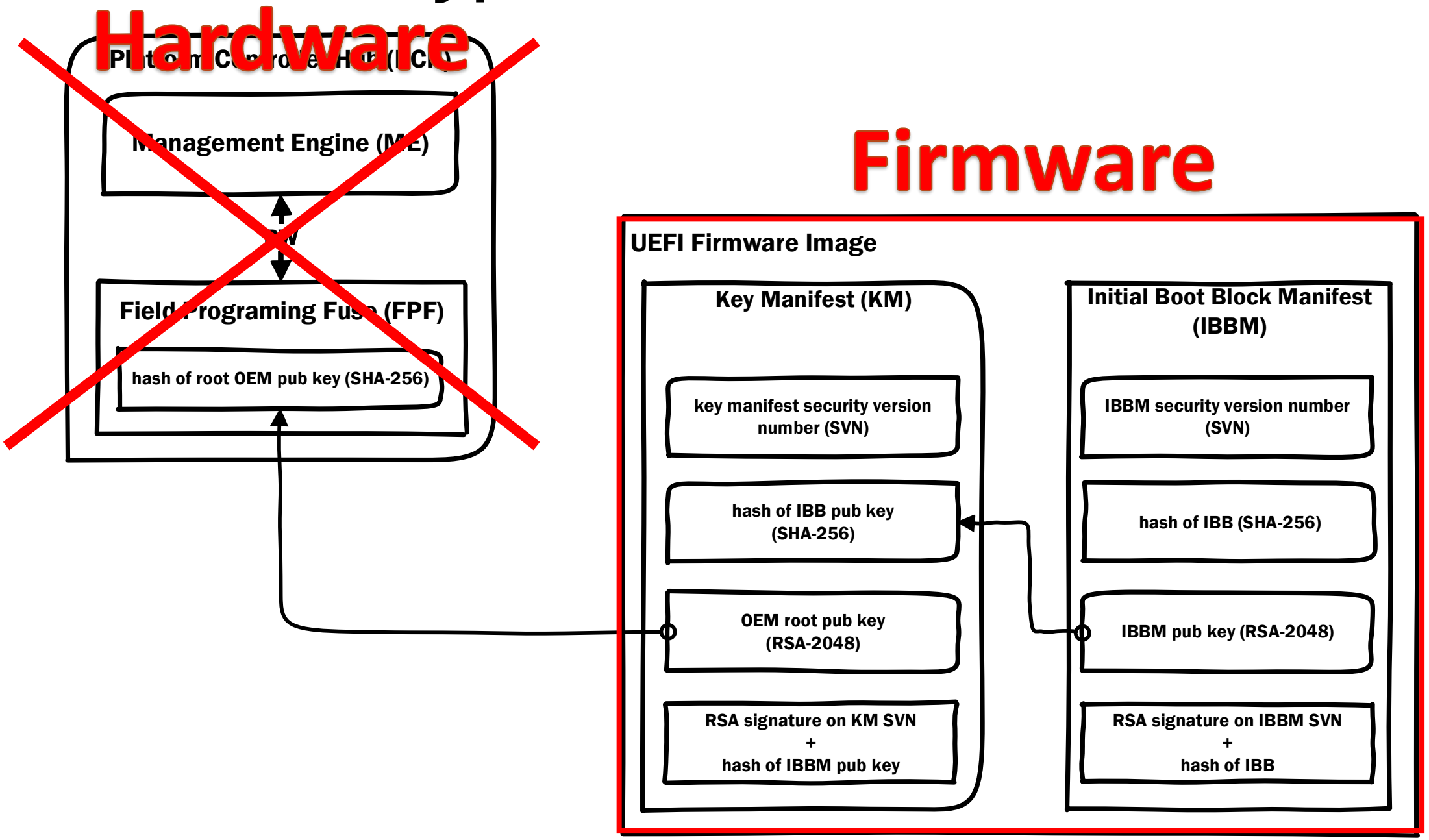
not Locked (CVE-2017-11313)

Target Platform

- **Gigabyte (GB-BSi7HA-6500)**
  - ✓ Intel 6th generation Core i7 CPU (Skylake) with vPro
  - ✓ Intel Boot Guard – ENABLED
  - ✓ Intel BIOS Guard – NOT ENABLED

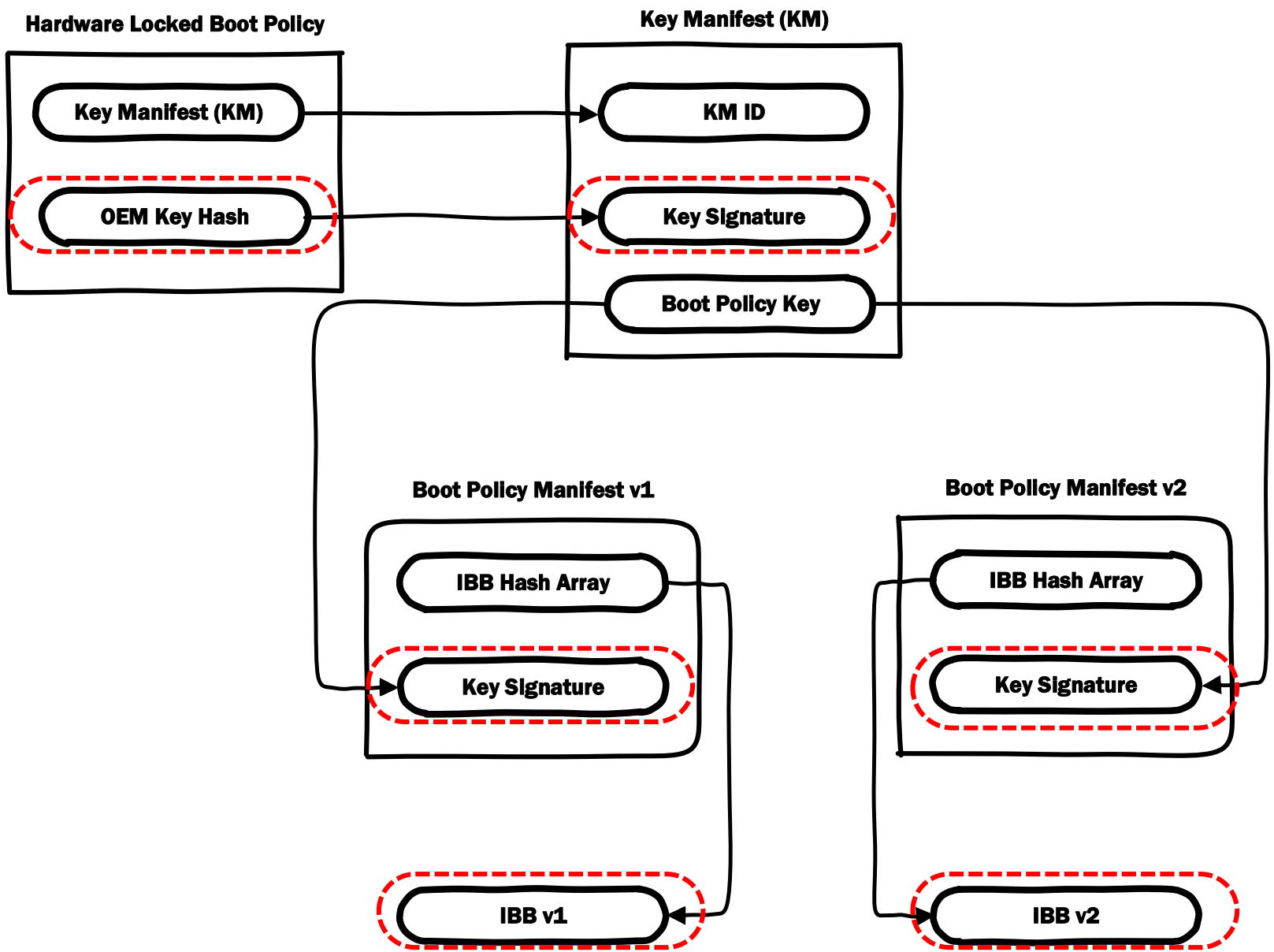
- **Vulnerabilities**
  - ✓ Host Write/Read Access to ME (CVE-2017-11314)
  - ✓ Intel Boot Guard Configuration not Locked (CVE-2017-11313)

Boot Guard Bypass: Full Chain of Trust



```
Last ME reset reason                    Global system reset
Local FWUpdate                          Enabled
BIOS Config Lock                        Enabled
GbE Config Lock                         Enabled
Host Read Access to ME                  Disabled
Host Write Access to ME                 Disabled
Host Read Access to EC                  Disabled
Host Write Access to EC                 Disabled
SPI Flash ID 1                          EF4019
SPI Flash ID 2                          Unknown
BIOS boot State                         Post Boot
OEM ID                                  68853622-eed3-4e83-8a86-6cde315f6b78
Capability Licensing Service            Enabled
OEM Tag                                 0x00000000
Slot 1 Board Manufacturer               0x00001028
Slot 2 System Assembler                 0x00000000
Slot 3 Reserved                         0x00000000
M3 Autotest                             Enabled
C-link Status                           Enabled
Independent Firmware Recovery           Disabled
EPID Group ID                           0xFF5
LSPCON Ports                            None
5K Ports                                None
OEM Public Key Hash FPF                 963847BF4FA55A02B326C561C56BCE68820041CAC285B43AE1F1B1FEB3FED4A4
OEM Public Key Hash ME                  963847BF4FA55A02B326C561C56BCE68820041CAC285B43AE1F1B1FEB3FED4A4
ACM SVN FPF                             0x2
KM SVN FPF                              0x0
BSMM SVN FPF                            0x0
GuC Encryption Key FPF                  0000000000000000000000000000000000000000000000000000000000000000
GuC Encryption Key ME                   0000000000000000000000000000000000000000000000000000000000000000

                                        FPF                     ME
                                        ---                     --
Force Boot Guard ACM                    Enabled                 Enabled
Protect BIOS Environment                Enabled                 Enabled
CPU Debugging                           Enabled                 Enabled
BSP Initialization                      Enabled                 Enabled
Measured Boot                           Enabled                 Enabled
Verified Boot                           Enabled                 Enabled
Key Manifest ID                         0xF                     0xF
Enforcement Policy                      0x3                     0x3
```

Boot Guard Bypass: Don’t checks for Vendor Hash

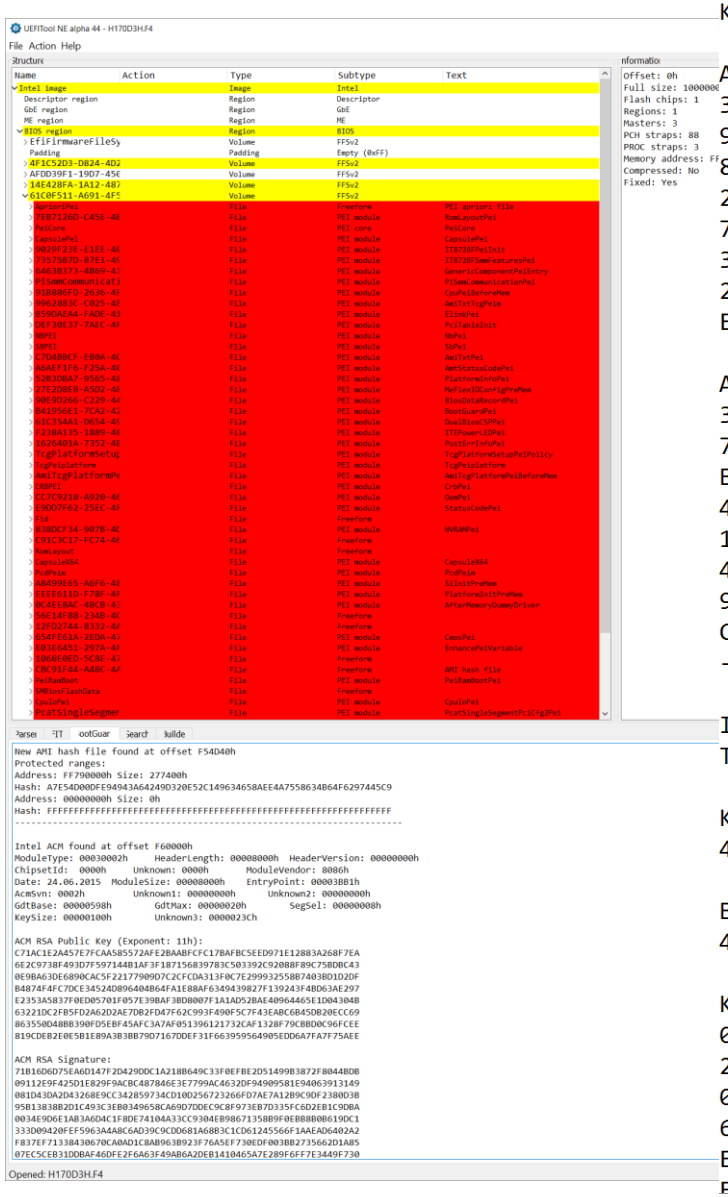


UEFITool: Boot Guard visual validation

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

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

UEFITool: Bo lidation




```
Parser   FIT   BootGuard   Search   Builder

Intel ACM found at offset 1E08000h
ModuleType: 00030002h     HeaderLength: 00008000h  HeaderVersion: 00000000h
ChipsetId:  0000h     Flags: 8000h       ModuleVendor: 8086h
Date: 09.02.2017  ModuleSize: 00008000h    EntryPoint: 00003BD1h
AcmSvn: 0000h         Unknown1: 00000000h      Unknown2: 00000000h
GdtBase: 00000598h        GdtMax: 00000020h        SegSel: 00000008h
KeySize: 00000100h        Unknown3: 0000023Ch

ACM RSA Public Key (Exponent: 11h):
3B355F85FE925740531002649FC6BC1A31EC188430FC551D3223BF2D6A24E1CC
9A6DECD046890ED9286CDBCA0CBEE26D66E37555E94138CD79F0B604F48F445B
8F0F53758BB7D6013C1A7C3A8B8AA6BCDA97884EE4F70FDE43D239AB61634531
20C21852766B9962C8F1A7ECDF5B1792029EF360E98876815B3D40754CAD169F
7B9996A2349898460A4B68D62122205E3866CBD10A73422F15A7D0676322743D
34F94AB305473D0139899ACCD8F0DD27B8423D011C595185FEB6B762D720F193
2489054A9E03825A9DF643BB8BD42B57E8B03860E18D659F01D811116F5E266A
E848BB0BD94A6179D257380EAA294BD6A87A9DFAAC93DB575A77E4EDB42AC8AB

ACM RSA Signature:
33BAADCA1B4148DC5FABE281508AD63F9635633E8E10B523F20D53E23021032F
73E82E8D7A4A160BA484AFC14ED8558B1574B6BC955F30302CCD2FEAFD72EA2E
B6D3AC283C552B820E8EB837571B51CA49F911DA34A1E0266A5D1423F0910E2B
409DDC8B84785E811DBB7B6158B53BB4185717715 3688E7E243FF3E2CD2830C1
19DC9D4E5D8FD6FB1C35460A1C38FC9A3AE02E34C7C879D2DE0C9A39F6A95C01
4655F8E53884141B363687818CEBB1D82467E534D7C6B0B1C244E42F9B91EC3A
93FED0727C1641CB1F558D20A104C0A8C43F37B884B468FF4A0D5BFE7D02AF82
C5B39B52D196D9974849AD2944C02A1C5354F7316EDF5BF2D14C2F361FD875A0
-----------------------------------------------------------------------

Intel BootGuard Key manifest found at offset 1E00500h
Tag: __KEYM__ Version: 10h KmVersion: 10h KmSvn: 00h KmId: 01h

Key Manifest RSA Public Key Hash:
47B3693AF671BACC3410345192A4B819B1FDE81B3D0C444209A46C440054FA57

Boot Policy RSA Public Key Hash:
40246C1C7F36AC1C812DF84F404DE0A8819DBEA3F013B902BAF0D587AE384B4A

Key Manifest RSA Public Key (Exponent: 10001h):
0B6FF5F14E8B40D1A69743A513D73B60E6C9729E30A56294DE932A06256322AA
25A8F1BA2B09D561F55153917E5AFB325BD494E1AF37195636E64D62427AED1B
08798A4190185AC8C97CB8D645ADC4C197B0D25FAE1C500B398DF1D08228D936
67AE9425CB1F99429F93A982DC8A29B2268094740706C6941F7D9382935293E3
E52162B24EBF4210C776E8BF6610A11D37F83AB58839EE9FF3399085D7E821EC
E6886816A8426DD20544D96673B165E5544366857316824 60E78EABDCA14B470
```

EFITool/releases/tag/A43

ov/bypass-intel-boot-guard-

Boot Guard Bypass: Not protected PEI Raw/Pad

UEFITool NE alpha 46 - outimage.bin

File Action View Help

Structure

Name	Action	Type	Subtype	Text
Padding		Padding	Empty (0xFF)	
8579D1CA-45E8-4F1C-A789-FFA770672099		Volume	FFSv2	
Pad-file		File	Pad	
2D27C6AA-7DCD-41F5-BB10-21166BE7E143		File	Raw	
Pad-file		File	Pad	
2D27C655-7DCD-41F5-BB10-21166BE7E143		File	Raw	
Pad-file		File	Pad	
B42C5280-D6AD-4FDF-871A-77E9DF40F30C		File	Raw	
Pad-file		File	Pad	
90BF2BFB-F998-4CBC-AD72-008D4D047A4B		File	PEI module	PeiTbtInit
Pad-file		File	Pad	
3EAF5E3A-E4B2-48E6-A9F1-B75CF204BCC8		File	PEI module	PeiBoardConfigInit
Pad-file		File	Pad	
59ADD62D-A1C0-44C5-A90F-A1168770468C		File	PEI module	PlatformInit
Pad-file		File	Pad	
FspsWrapperPeim		File	PEI module	FspsWrapperPeim
Pad-file		File	Pad	
960241A5-5CE6-4EBF-B9C4-E37D294BEBA9		File	PEI module	FspWrapperPeim
Volume free space		Free spa…		
CF1406C5-3FEC-47EB-A6C3-B71A3EE00B95		Volume	FFSv2	
AprioriPei		File	Freeform	PEI apriori file
Pad-file		File	Pad	
FB3D20EE-CB5D-45EE-BBAE-C5AAA9C0FDFA		File	PEI module	ChipsetSvcPei
Pad-file		File	Pad	
PcatSingleSegmentPciCfg2Pei		File	PEI module	PcatSingleSegmentPciCfg2Pei
Pad-file		File	Pad	
FirmwarePerformancePei		File	PEI module	FirmwarePerformancePei
CapsulePei		File	PEI module	CapsulePei
Pad-file		File	Pad	
CpuIoPei		File	PEI module	CpuIoPei
Pad-file		File	Pad	
PiSmmCommunicationPei		File	PEI module	PiSmmCommunicationPei
Pad-file		File	Pad	
22A5BB4D-F962-4D33-9CD6-FDAD39ACD153		File	PEI module	PlatformBoardIdPei
Pad-file		File	Pad	
EB00DB50-C654-460F-8D7A-0E444FD32B45		File	PEI module	DebugMaskPei
Pad-file		File	Pad	
PeiVariablePei		File	PEI module	PeiVariable
Pad-file		File	Pad	
C2239ACD-21D0-4CB0-B7DB-6D35EE7B0CC1		File	PEI module	FirmwareRevisionSyncPei

Boot Guard Bypass: Not protected PEI Raw/Pad

Hex view: 2D27C6AA-7DCD-41F5-BB10-21166BE7E143

```
0000 AA C6 27 2D CD 7D F5 41 BB 10 21 16 6B E7 E1 43   ªÆ'-Í}õA».!.kçáC
0010 5D AA 01 00 D8 0E 00 F8 5F 46 49 54 5F 20 20 20   ]ª..Ø..ø_FIT_
0020 06 00 00 00 00 01 80 0A 60 00 B6 FF 00 00 00 00   ...... .`.¶ÿ....
0030 00 00 00 00 00 01 01 00 60 78 B7 FF 00 00 00 00   ........`x·ÿ....
0040 00 00 00 00 00 01 01 00 00 80 E0 FF 00 00 00 00   ......... àÿ....
0050 00 80 00 00 00 01 02 00 00 05 E0 FF 00 00 00 00   . ........àÿ....
0060 41 02 00 00 00 01 0B 00 00 09 E0 FF 00 00 00 00   A.........àÿ....
0070 BB 02 00 00 00 01 0C 00 20 20 20 20 20 20 20 20   ».......
0080 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0090 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00A0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00B0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00C0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00D0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00E0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00F0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0100 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0110 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0120 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0130 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0140 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0150 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0160 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0170 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0180 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0190 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
01A0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
01B0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
01C0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
01D0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
01E0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
01F0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0200 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0210 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
```

Information

```
Offset: 1E000E8h
File GUID: 2D27C6AA-7DCD-41F5-BB10-21166BE7E143
Type: 01h
Attributes: 00h
Full size: ED8h (3800)
Header size: 18h (24)
Body size: EC0h (3776)
Tail size: 0h (0)
State: F8h
Header checksum: 5Dh, valid
Data checksum: AAh, valid
Header memory address: FFE000E8h
Data memory address: FFE00100h
Compressed: No
Fixed: Yes
```

Pad-file	File	Pad	
PeiVariablePei	File	PEI module	PeiVariable
Pad-file	File	Pad	
C2239ACD-21D0-4CB0-B7DB-6D35EE7B0CC1	File	PEI module	FirmwareRevisionSyncPei

Boot Guard Bypass: PEI Driver Loader specifics

```
EFI_STATUS
PeiLoadImage (
  IN      CONST EFI_PEI_SERVICES        **PeiServices,
  IN      EFI_PEI_FILE_HANDLE           FileHandle,
  IN      UINT8                         PeimState,
  OUT     EFI_PHYSICAL_ADDRESS          *EntryPoint,
  OUT     UINT32                        *AuthenticationState
```

← Always FALSE and never used for PEI Loader

https://github.com/tianocore/edk2/blob/master/MdeModulePkg/Core/Pei/Image/Image.c

Boot Guard ader specifics

```
EFI_STATUS
PeiLoadImage (
  IN      CONST
  IN      EFI_PE
  IN      UINT8
  OUT     EFI_PH
  OUT     UINT32                                  nState
```

```
do {
  PpiStatus = PeiServicesLocatePpi (
                &gEfiPeiLoadFilePpiGuid,
                Index,
                NULL,
                (VOID **)&LoadFile
                );
  if (!EFI_ERROR (PpiStatus)) {
    Status = LoadFile->LoadFile (
                       LoadFile,
                       FileHandle,
                       &ImageAddress,
                       &ImageSize,
                       EntryPoint,
                       AuthenticationState
                       );
    if (!EFI_ERROR (Status) || Status == EFI_WARN_BUFFER_TOO_SMALL) {
      //
      // The shadowed PEIM must be relocatable.
      //
      if (PeimState == PEIM_STATE_REGISITER_FOR_SHADOW) {
        IsStrip = RelocationIsStrip ((VOID *) (UINTN) ImageAddress);
        ASSERT (!IsStrip);
        if (IsStrip) {
          return EFI_UNSUPPORTED;
        }
      }
```

← **Always FALSE and never used for PEI Loader**

https://github.com/tianoco ge.c

Boot Guard Bypass: Inject DXE Payload

```
typedef struct {
  EFI_TABLE_HEADER                Hdr;
  CHAR16                          *FirmwareVendor;
  UINT32                          FirmwareRevision;
  EFI_HANDLE                      ConsoleInHandle;
  EFI_SIMPLE_TEXT_INPUT_PROTOCOL  *ConIn;
  EFI_HANDLE                      ConsoleOutHandle;
  EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL *ConOut;
  EFI_HANDLE                      StandardErrorHandle;
  EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL *StdErr;
  // A pointer to the EFI Runtime Services Table.
  EFI_RUNTIME_SERVICES            *RuntimeServices;
  // A pointer to the EFI Boot Services Table.
  EFI_BOOT_SERVICES               *BootServices;
  UINTN                           NumberOfTableEntries;
  EFI_CONFIGURATION_TABLE         *ConfigurationTable;
} EFI_SYSTEM_TABLE;
```

← Find:

```
define EFI_BOOT_SERVICES_SIGNATURE   SIGNATURE_64 ('B','O','O','T','S','E','R','V')
define EFI_BOOT_SERVICES_SIGNATURE   SIGNATURE_64 ('B','O','O','T','S','E','R','V')
```

1) BootServices → LocateProtocol
2) Hook LocateProtocol
3) Inject malicious DXE Driver

https://github.com/tianocore/edk2/... /MdePkg/Include/Uefi/UefiSpec.h

EVERYTHING IS BROKEN

OFFENSIVECON

Thank you for your attention!

@matrosov