

Untrusted Roots: exploiting vulnerabilities in Intel ACMs

Alexander Ermolov
[@flothrone](https://twitter.com/flothrone)

#WhoAmI

- Former team member at [Digital Security](#) and [Embedi](#)
- [ZeroNights](#) security conference
- Intel Management Engine
 - [Intel AMT. Stealth Breakthrough](#)
- Intel Boot Guard
 - [Safeguarding rootkits: Intel Boot Guard](#)
 - [Bypassing Intel Boot Guard](#)
- UEFI BIOS
 - [UEFI BIOS holes: So Much Magic, Don't Come Inside](#)
 - [NUClear explotion](#)
 - [Microcode downgrade](#)



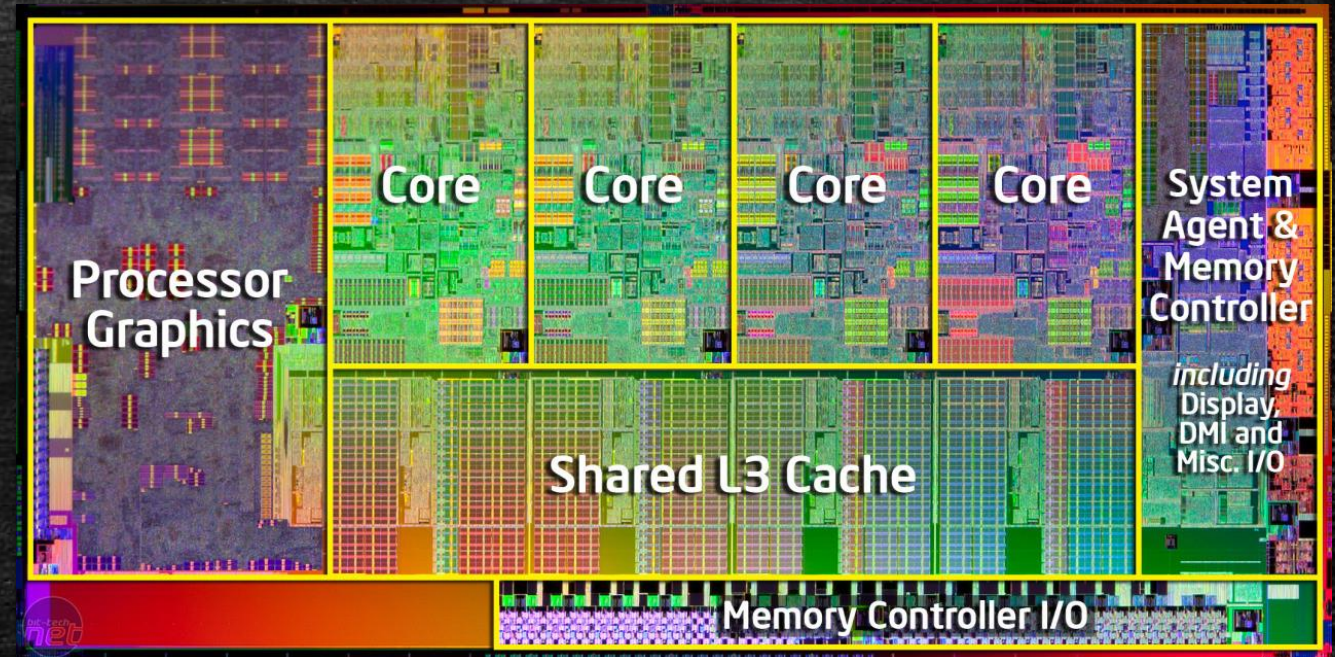
#Agenda

- Intel CPU Roots of Trust overview
 - Intel Microcode Updates
 - FIT
 - Intel ACMs
- Downgrading Intel Boot Guard ACM
- Downgrading Microcode and Intel TXT ACM
- Mitigations & Takeaways

Intel CPU Roots of Trust overview

Inside Intel CPU

- Processor cores
 - BSP (Bootstrap Processor)
 - APs (Application Processors)
- Graphics core
- IMC (Integrated Memory Controller)
- L3 cache
- I/O logic



Microcode

Control Unit has Microcode ROM that contains the CPU microcode - a program written in a hardware-level instructions to implement a higher-level instructions

For example, MOVS instruction implementation:

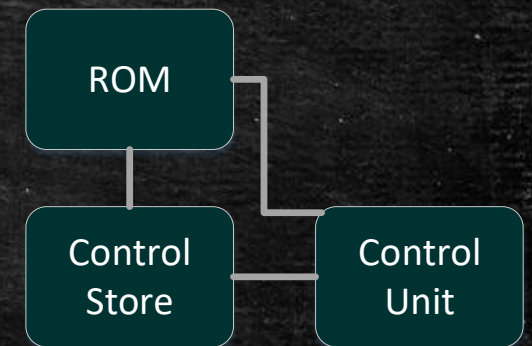
```
LLDF      ; load direction flag to latch in functional unit
OR ecx, ecx ; test if ECX is zero
JZ end    ; terminate string move if ECX is zero
```

loop:

```
MOVFM tmp0, [esi] ; move the data to tmp data from source and inc/dec ESI
MOVIM [edi], tmp0 ; move the data to destination and inc/dec EDI
EDECXJNZ loop    ; dec ECX and repeat until zero
```

end:

```
EXIT
```

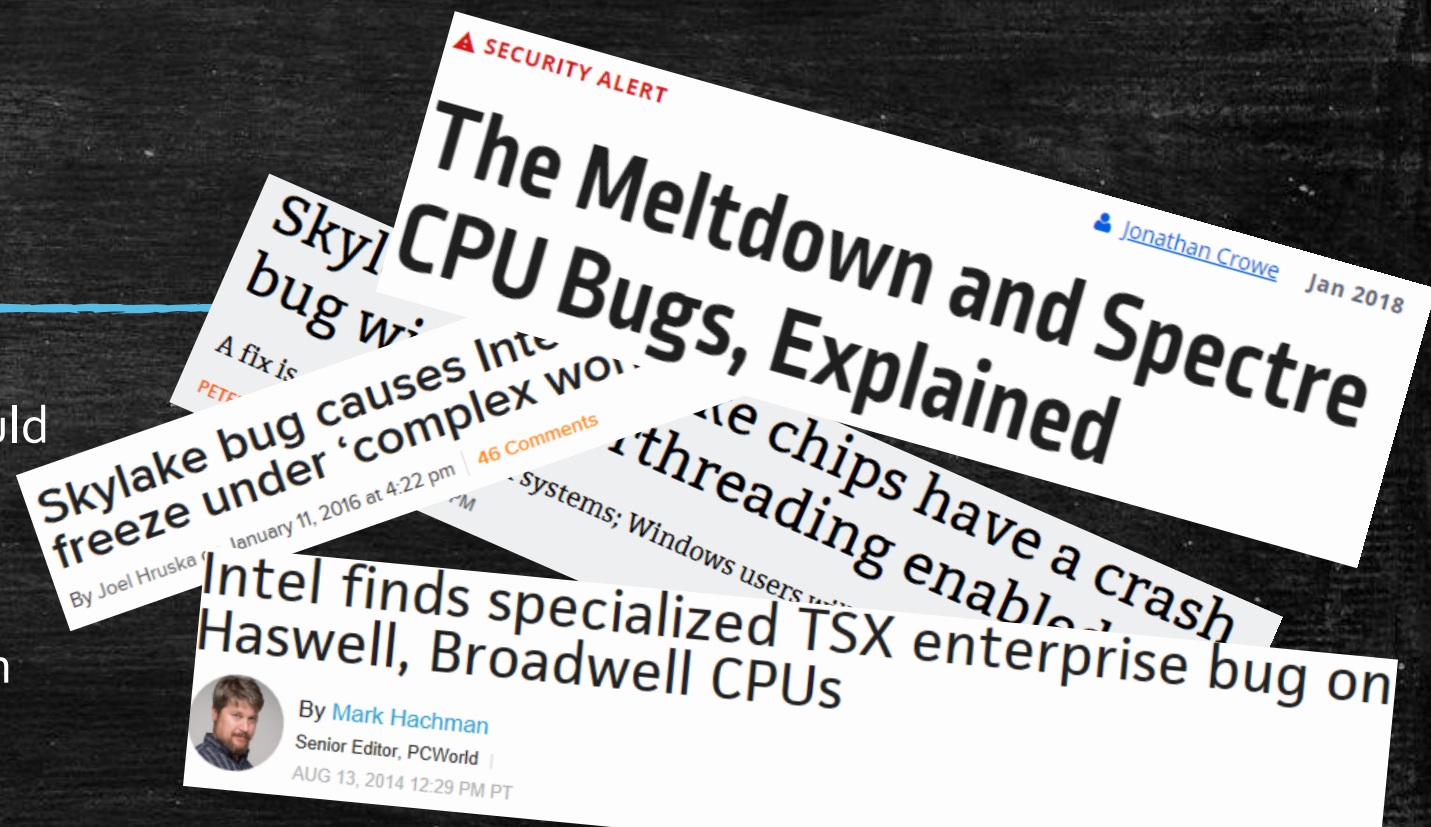


Security Analysis of x86 Processor Microcode
https://www.dcddcc.com/docs/2014_paper_microcode.pdf

Microcode update

Microcode can have bugs, so it should be updatable

The updated microcode has to be loaded into Control Store upon each CPU power on



	Address	Size	Version	Checksum	Type	
1	_FIT_	00000070h	0100h	00h	FIT Header	
2	00000000FFD70400h	00017C00h	0100h	00h	Microcode	CPUID: 000906EAh, Revision: 00000096h, Date: 02.05.2018
3	00000000FFD88000h	00018000h	0100h	00h	Microcode	CPUID: 000906EBh, Revision: 0000008Eh, Date: 24.03.2018
4	00000000FFDA0000h	00017800h	0100h	00h	Microcode	CPUID: 000906ECh, Revision: 00000084h, Date: 19.02.2018
5	00000000FFF10000h	00008000h	0100h	00h	BIOS ACM	LocalOffset: 00000018h, EntryPoint: 00003BD1h, ACM SVN: 0000h, Date: 09.02.2017
6	00000000FFFCDC80h	00000000h	0100h	00h	BootGuard Key Manifest	
7	00000000FFFC0000h	00000000h	0100h	00h	BootGuard Boot Policy	

Firmware Interface Table (FIT)

Intel image	Image	Intel
Descriptor region	Region	Descriptor
GbE region	Region	GbE
ME region	Region	ME
BIOS region	Region	BIOS
>FA4974FC-AF1D-4E5D-BDC5-DACD6D27BAEC	Volume	FFSv2
>FA4974FC-AF1D-4E5D-BDC5-DACD6D27BAEC	Volume	FFSv2
Padding	Padding	Empty (0xFF)
>4F1C52D3-D824-4D2A-A2F0-EC40C23C5916	Volume	FFSv2
>AFDD39F1-19D7-4501-A730-CE5A27E1154B	Volume	FFSv2
Pad-file	File	Pad
B52282EE-9B66-44B9-B1CF-7E5040F787C1	File	Raw
Pad-file		
>Microcode		
Pad-file		
BiosAc		
Volume free space		
>14E428FA-1A12-4875-B637-8B3CC87FDF07		
>61C0F511-A691-4F54-974F-B9A42172CE53		
OxFFFFFFFFC0		

NVRAM

DXE

FVDATA

PEI

PEI + SEC

Hex view: B52282EE-9B66-44B9-B1CF-7E5040F787C1

0000	5F	46	49	54	5F	20	20	20	07	00	00	00	00	01	00	00	_FIT_
0010	00	04	D7	FF	00	00	00	00	00	00	00	00	00	01	01	00	..xÿ.....
0020	00	80	D8	FF	00	00	00	00	00	00	00	00	00	01	01	00	..ÿ.....
0030	00	00	DA	FF	00	00	00	00	00	00	00	00	00	01	01	00	..ÿ.....
0040	00	00	F1	FF	00	00	00	00	00	00	00	00	00	01	02	00	..ñÿ.....
0050	80	DC	FC	FF	00	00	00	00	00	00	00	00	00	01	0B	00	..Üÿ.....
0060	00	CC	FC	FF	00	00	00	00	00	00	00	00	00	01	0C	00	..üÿ.....

Firmware Interface Table (FIT)

- Is a required element for Intel 64 architecture since introduction of Boot Guard technology
- Can point to microcode update (MCU) binaries
- CPU can load microcode updates from FIT prior to execution of BIOS and before starting Intel Boot Guard

5	00000000FFDD6EB0h	00012C00h	0100h	00h	Microcode	CPUID: 000506E2h, Revision: 0000002Ch, Date: 01.07.2015
6	00000000FFDE9AB0h	00017800h	0100h	00h	Microcode	CPUID: 000806E9h, Revision: 00000042h, Date: 02.10.2016
7	00000000FFE012B0h	00017800h	0100h	00h	Microcode	CPUID: 000906E9h, Revision: 00000042h, Date: 02.10.2016
8	00000000FFE18AB0h	00017800h	0100h	00h	Microcode	CPUID: 000506E8h, Revision: 00000034h, Date: 10.07.2016
9	00000000FFFE0000h	00008000h	0100h	00h	BIOS ACM	LocalOffset: 00000018h, EntryPoint: 00003BB1h, ACM SVN: 0002h, Date: 07.02.2016
10	00000000FFFD4C80h	00000241h	0100h	00h	BootGuard Key Manifest	LocalOffset: 00000018h, KM Version: 10h, KM SVN: 00h, KM ID: 0Fh
11	00000000FFFD3C00h	000002DFh	0100h	00h	BootGuard Boot Policy	LocalOffset: 00000018h, BP SVN: 00h, ACM SVN: 02h

Microcode Update binary main header

Microcode Update binary starts with the main header followed by an extended header and update data

```
typedef struct MICROCODE_UPDATE_HEADER {  
    unsigned long header_version;        // 1  
    unsigned long update_revision;  
    unsigned long date;                  // BCD format  
    unsigned long processor_signature;    // CPUID  
    unsigned long checksum;  
    unsigned long loader_revision;  
    unsigned long processor_flags;  
    unsigned long data_size;             // in bytes  
    unsigned long total_size;            // in bytes  
    unsigned char reserved[0x0C];  
};
```

0000h:	01 00 00 00	A6 00 00 00	16 20 21 08	E3 06 05 00!....!.ã...
0010h:	1F 67 51 E9	01 00 00 00	36 00 00 00	D0 7B 01 00	.gQé.....6...Đ{...
0020h:	00 7C 01 00	00 00 00 00	00 00 00 00	00 00 00 00
0030h:	00 00 00 00	A1 00 00 00	01 00 02 00	A6 00 00 00i..... ...
0040h:	06 00 00 00	51 5E 00 00	21 08 16 20	E1 17 00 00Q^...!... á...
0050h:	01 00 00 00	E3 06 05 00	00 00 00 00	00 00 00 00ã.....
0060h:	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0070h:	00 00 00 00	36 00 00 00	04 00 00 00	00 00 00 006.....
0080h:	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0090h:	C2 F3 14 67	67 C0 32 94	6F 7C DF 6D	49 3C EB C3	Âó.ggÀ2"o BmI<ëÃ
00A0h:	DC 5B 0D 20	E5 63 FA 8C	58 7A 7E A4	A2 6F 80 D4	Ů[. âcúEXz~HcoœÔ
00B0h:	A7 A0 92 D1	11 94 5F 4F	6B 9C AF E4	62 CE 00 7A	\$ 'Ñ."_Okæ-abî.z
00C0h:	A4 42 6D 81	7B 27 3E 07	03 63 AE B7	67 0D 63 9B	Hm.{ ' > ..c@·g.c>
00D0h:	C3 5F 0E C2	40 31 58 20	37 39 FC 12	9C B6 B6 C6	Ã_.Ã@1X 79ü.œqgÆ
00E0h:	6C 17 2B 8D	AC 09 2F E3	FC EC 36 2C	64 32 A8 A9	l.+.-./ãüi6,d2"©
00F0h:	99 B6 35 1F	AA E5 B8 14	CA 21 12 7F	B0 0B 41 F5	mq5.·ð.Ê!...°Að

Microcode Update binary extended header

```
typedef struct MICROCODE_UPDATE_EXTENDED_HEADER {
    unsigned short module_type;           // 0
    unsigned short module_subtype;        // 0
    unsigned long  header_size;            // in dwords
    unsigned long  header_version;         // 0x20001
    unsigned long  update_revision;
    unsigned long  unknown[2];
    unsigned long  date;                   // BCD format
    unsigned long  update_size;            // in dwords
    unsigned long  svn;
    unsigned long  processor_signature;
    unsigned long  unknown2[0x0E];
    unsigned char  update_hash[0x20];     // SHA256 hash of the decrypted update data
    unsigned char  rsa_mod[0x100];        // RSA 2048 public key modulus
    unsigned long  rsa_exp;                // RSA 2048 public key exponent
    unsigned char  signature[0x100];      // RSA 2048 signature of the header
};
```

0000h:	01 00 00 00	A6 00 00 00	16 20 21 08	E3 06 05 00!.ă...
0010h:	1F 67 51 E9	01 00 00 00	36 00 00 00	D0 7B 01 00	.gQé....6...Đ{..
0020h:	00 7C 01 00	00 00 00 00	00 00 00 00	00 00 00 00
0030h:	00 00 00 00	A1 00 00 00	01 00 02 00	A6 00 00 00i..... ...
0040h:	06 00 00 00	51 5E 00 00	21 08 16 20	E1 17 00 00Q^...!.. á...
0050h:	01 00 00 00	E3 06 05 00	00 00 00 00	00 00 00 00ă.....
0060h:	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0070h:	00 00 00 00	36 00 00 00	04 00 00 00	00 00 00 006.....
0080h:	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0090h:	C2 F3 14 67	67 C0 32 94	6F 7C DF 6D	49 3C EB C3	Âó.ggÀ2"o BmI<ěĂ
00A0h:	DC 5B 0D 20	E5 63 FA 8C	58 7A 7E A4	A2 6F 80 D4	Ů[. ěcúEXz~ꝛcoEO
00B0h:	A7 A0 92 D1	11 94 5F 4F	6B 9C AF E4	62 CE 00 7A	\$ 'Ñ."_Okœābî.z
00C0h:	A4 42 6D 81	7B 27 3E 07	03 63 AE B7	67 0D 63 9B	ꝛBm.{'>..c@·g.c>
00D0h:	C3 5F 0E C2	40 31 58 20	37 39 FC 12	9C B6 B6 C6	Ă_.Ă@1X 79ü.œŧŧE
00E0h:	6C 17 2B 8D	AC 09 2F E3	FC EC 36 2C	64 32 A8 A9	1.+.-./ăüi6,d2"©
00F0h:	99 B6 35 1F	AA F5 B8 14	CA 21 12 7F	B0 0B 41 F5	ꝛŧ5.ăö..Ê!...°.Aö

Microcode Update binary data

- The main part in MCU binary is Data (encrypted, the decryption key is hardcoded into CPU)
- Hash of RSA public key to authenticate the MCU is also hardcoded into CPU
- So no one knows exactly what Microcode is capable of

Known facts about Microcode

- Implements instructions
- Configures the execution logic on the line (that's how side-channels are fixed)
- Implements some startup behavior (like FIT parsing)
- Loads MCU from FIT
- Loads and executes Intel Authenticated Code Modules (ACMs) (from FIT or not)

Authenticated Code Modules (ACMs)

- Signed and sometimes encrypted Intel code modules
 - Loaded and executed from L3 cache (sometimes called AC RAM)
 - Serve as a Root-of-Trusts and a core of implementation for technologies:
 - Intel Boot Guard
type/subtype: 2.3, **not encrypted**, signed with **KEY₂**
 - Intel Trusted Execution Technology (TXT)
type/subtype: 2.0, **not encrypted**, signed with **KEY₃**
 - Intel BIOS Guard (PFAT)
type/subtype: 1.1, **encrypted**, signed with **KEY₁**
- FYI: microcode update binary
type/subtype: 1.0, **encrypted**, signed with **KEY₁**

CPU Roots of Trust inside UEFI BIOS

➤4F1C52D3-D824-4D2A-A2F0-EC40C23C5916	Volume	FFSv2	
▼AFDD39F1-19D7-4501-A730-CE5A27E1154B	Volume	FFSv2	
Pad-file	File	Pad	
B52282EE-9B66-44B9-B1CF-7E5040F787C1	File	Raw	FIT
Pad-file	File	Pad	
➤Microcode	File	Raw	Microcode Updates
Pad-file	File	Pad	
BiosAc	File	Raw	Intel TXT ACM
Volume free space	Free space		
▼61C0F511-A691-4F54-974F-B9A42172CE53	Volume	FFSv2	
➤AprioriPei	File	Freeform	PEI apriori file
0A602CEB-95A0-40C4-8181-EDCD801D0003	File	Freeform	PEI module
Pad-file	File	Pad	
C30FFF4A-10C6-4C0F-A454-FD319BAF6CE6	File	Raw	
Pad-file	File	Pad	
7C9A98F8-2B2B-4027-8F16-F7D277D58025	File	Raw	
➤8E295870-D377-4B75-BFDC-9AE2F6DBDE22	File	Freeform	
7934156D-CFCE-460E-92F5-A07909A59ECA	File	Raw	Intel BIOS Guard ACM
Pad-file	File	Pad	
6520F532-2A27-4195-B331-C0854683E0BA	File	Raw	Intel Boot Guard ACM
➤Non-empty pad-file	File	Pad	
TxtPeiAp	File	Raw	
Pad-file	File	Pad	
➤Volume Top File	File	SEC core	Volume Top File

Intel Boot Guard ACM downgrading

Intel Boot Guard protection area

>4F1C52D3-D824-4D2A-A2F0-EC40C23C5916	Volume	FFSv2	
▼AFDD39F1-19D7-4501-A730-CE5A27E1154B	Volume	FFSv2	
Pad-file	File	Pad	
B52282EE-9B66-44B9-B1CF-7E5040F787C1	File	Raw	FIT
Pad-file	File	Pad	
>Microcode	File	Raw	Microcode Updates
Pad-file	File	Pad	
BiosAc	File	Raw	Intel TXT ACM
Volume free space	Free space		
▼61C0F511-A691-4F54-974F-B9A42172CE53	Volume	FFSv2	
>AprioriPei	File	Freeform	PEI apriori file
▼0A603CEB-05A0-40C4-0181-EDCD801D0000	File	DET module	
Pad-file	File	Pad	
C30FFF4A-10C6-4C0F-A454-FD319BAF6CE6	File	Raw	
Pad-file	File	Pad	
7C9A98F8-2B2B-4027-8F16-F7D277D58025	File	Raw	
>8E295870-D377-4B75-BFDC-9AE2F6DBDE22	File	Freeform	
7934156D-CFCE-460E-92F5-A07909A59ECA	File	Raw	Intel BIOS Guard ACM
Pad-file	File	Pad	
6520F532-2A27-4195-B331-C0854683E0BA	File	Raw	Intel Boot Guard ACM
>Non-empty pad-file	File	Pad	
TxtPeiAp	File	Raw	
Pad-file	File	Pad	
>Volume Top File	File	SEC core	Volume Top File

Marked blue = protected area by Intel Boot Guard

Intel ACM header

```
typedef struct ACM_HEADER {
    unsigned short module_type;
    unsigned short module_subtype;
    unsigned long header_size;           // in dwords
    unsigned long header_version;
    unsigned short chipset_id;
    unsigned short flags;
    unsigned long module_vendor;        // 0x8086
    unsigned long date;                 // BCD format
    unsigned long module_size;          // in dwords
    unsigned short acm_svn;
    unsigned short se_svn;




















    ...

    unsigned char  rsa_mod[0x100];      // RSA 2048 public key modulus
    unsigned long  rsa_exp;              // RSA 2048 public key exponent
    unsigned char  signature[0x100];    // RSA 2048 signature of the header + data
};
```

0000	02 00	03 00	A1 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00j.....
0010	86 80	00 00	30 04 19 20	00 20 00 00	02 00	02 00	00 00	..0..
0020	00 00	00 00	00 00 00 00	20 00 00 00	98 05	00 00	00 00
0030	08 00	00 00	D1 3B 00 00	00 00 00 00	00 00	00 00	00 00Ñ;.....
0040	00 00	00 00	00 00 00 00	00 00 00 00	00 00	00 00	00 00
0050	00 00	00 00	00 00 00 00	00 00 00 00	00 00	00 00	00 00
0060	00 00	00 00	00 00 00 00	00 00 00 00	00 00	00 00	00 00
0070	00 00	00 00	00 00 00 00	40 00 00 00	8F 00	00 00	00 00@...Å...
0080	C7 1A	C1 E2	A4 57 E7 FC	AA 58 55 72	AF E2	BA AB	00 00	Ç.ÂâµWçüªXUr`âª«
0090	FC FC	17 BA	FB C5 EE D9	71 E1 28 83	A2 68	F7 EA	00 00	üü.ºÜÃîÜqá(¢h÷ê
00A0	6E 2C	97 38	F4 93 D7 F5	97 14 4B 1A	F3 F1	87 15	00 00	n, 8ô xõ .K.ón .
00B0	68 39	78 3C	50 33 92 C9	20 88 F8 9C	75 BD	BC 43	00 00	h9x<P3 É ¢ u%C
00C0	0E 9B	A6 3D	E6 89 0C AC	5F 22 17 79	09 D7	C2 CF	00 00	. '=æ .-_" .y.ªÃİ
00D0	CD A3	13 F0	C7 E2 99 93	25 58 B7 40	3B D1	D2 DF	00 00	İ£.ðÇâ %X·@;ÑÒß
00E0	B4 87	4F 4F	C7 DC E3 45	24 D8 96 40	4B 64	FA 1E	00 00	`OOÇÜãE\$ø @Kdú.
00F0	88 AF	63 49	43 98 27 F1	39 24 3F 4B	D6 3A	E2 97	00 00	`cIC 'ñ9\$?KÖ:â
0100	E2 35	3A 58	37 F0 ED 05	70 1F 05 7E	39 BA	F3 BD	00 00	â5:X7ðí.p...~9ºó%
0110	80 07	F1 A1	AD 52 BA E4	09 64 46 5E	1D 04	30 4B	00 00	.ñj Rªâ.dF^..0K
0120	63 22	1D C2	FB 5F D2 A6	2D 2A E7 DB	2F D4	7F 62	00 00	c".Âû_ò `-ªçÜ/ô b
0130	C9 93	F4 90	F5 C7 F4 3E	AB C6 B4 5D	B2 0E	CC 69	00 00	É ôÊðÇô>«Æ`]².İi
0140	86 35	50 D4	8B B3 90 FD	5E BF 45 AF	C3 A7	AF 05	00 00	5PÔ ¢Éý^¿E`Ã\$`.
0150	13 96	12 17	32 CA F1 32	8F 79 CB BD	0C 96	FC EE	00 00	. ..2Êñ2ÂyÊ%. üi
0160	81 9C	DE B2	E0 E5 B1 E8	9A 3B 3B B7	9D 71	67 DD	00 00	ü p²ââ±è ;;·*qgY
0170	EF 31	F6 63	95 95 64 90	5E DD 6A 7F	A7 F7	5A EE	00 00	İlôc dÉ^Yj \$÷Zİ
0180	11 00	00 00	25 0B 98 5C	69 BB 4D 7B	96 9A	88 44	00 00%. \i»M{ D
0190	1D 73	5A 51	0D BF C4 CF	AD B0 38 EC	E0 FB	E4 D6	00 00	.sZQ.¿Ãİ º8iàûäÖ
01A0	C9 C5	04 4F	5A 06 09 3A	6C 6F 71 8E	E1 CB	6A 8A	00 00	ÉÅ.OZ...loq áËj
01B0	BF 09	7A BD	F8 79 71 90	92 A7 8B DA	64 D9	A1 61	00 00	¿.z%øyqÉ \$ UdÜja
01C0	D2 35	8A 93	D5 4D 09 C0	1A 7D AE DB	BC 0E	3A AE	00 00	05 òM.À.}ªÜ%.:ª
01D0	ED 5E	79 94	C5 1A EB 72	7C 57 16 54	6E C4	43 EB	00 00	í^y Å.ër W.TnÄCë
01E0	AD 4C	9F 4D	91 12 3F 01	B9 65 89 98	A9 51	9A CB	00 00	L M .?.¹e @Q È

Intel Boot Guard ACM

- 32KB binary
- Hash of ACM's RSA public key is locked in Microcode
- SVN value protects from downgrading

	2013-04-30 BootGuardAcm.bin
	2013-04-30 BootGuardAcm.bin.idb
	2014-05-01 BootGuardAcm.bin
	2014-05-01 BootGuardAcm.bin.idb
	2015-05-14 BootGuardAcm.bin
	2015-05-14 BootGuardAcm.bin.idb
	2015-06-24 BootGuardAcm.bin
	2015-06-24 BootGuardAcm.bin.idb
	2016-02-07 BootGuardAcm.bin
	2016-02-07 BootGuardAcm.bin.idb
	2016-08-28 BootGuardAcm.bin
	2016-08-28 BootGuardAcm.bin.idb
	2017-02-09 BootGuardAcm.bin
	2017-02-09 BootGuardAcm.bin.idb
	2018-08-01 BootGuardAcm.bin
	2018-08-01 BootGuardAcm.bin.idb
	2019-04-30 BootGuardAcm.bin
	2019-04-30 BootGuardAcm.bin.idb
	BootGuardAcm.idc




















Intel Boot Guard ACM

- 32KB binary
- Hash of ACM's RSA public key is locked in Microcode
 - the key hasn't changed in a years!
- SVN value protects from downgrading

2013-04-30 BootGuardAcm.bin
2013-04-30 BootGuardAcm.bin.idb
2014-05-01 BootGuardAcm.bin
2014-05-01 BootGuardAcm.bin.idb
2015-05-14 BootGuardAcm.bin
2015-05-14 BootGuardAcm.bin.idb
2015-06-24 BootGuardAcm.bin
2015-06-24 BootGuardAcm.bin.idb
2016-02-07 BootGuardAcm.bin
2016-02-07 BootGuardAcm.bin.idb
2016-08-28 BootGuardAcm.bin
2016-08-28 BootGuardAcm.bin.idb
2017-02-09 BootGuardAcm.bin
2017-02-09 BootGuardAcm.bin.idb
2018-08-01 BootGuardAcm.bin
2018-08-01 BootGuardAcm.bin.idb
2019-04-30 BootGuardAcm.bin
2019-04-30 BootGuardAcm.bin.idb
BootGuardAcm.idc

Intel Boot Guard ACM

- 32KB binary
- Hash of ACM's RSA public key is locked in Microcode
 - the key hasn't changed in a years!
- SVN value protects from downgrading
 - changes infrequently!

		2013-04-30 BootGuardAcm.bin	
		2013-04-30 BootGuardAcm.bin.idb	
		2014-05-01 BootGuardAcm.bin	
		2014-05-01 BootGuardAcm.bin.idb	
		2015-05-14 BootGuardAcm.bin	
		2015-05-14 BootGuardAcm.bin.idb	
		2015-06-24 BootGuardAcm.bin	
		2015-06-24 BootGuardAcm.bin.idb	
		2016-02-07 BootGuardAcm.bin	
		2016-02-07 BootGuardAcm.bin.idb	
		2016-08-28 BootGuardAcm.bin	
		2016-08-28 BootGuardAcm.bin.idb	
!		2017-02-09 BootGuardAcm.bin	
		2017-02-09 BootGuardAcm.bin.idb	
		2018-08-01 BootGuardAcm.bin	
		2018-08-01 BootGuardAcm.bin.idb	
		2019-04-30 BootGuardAcm.bin	
		2019-04-30 BootGuardAcm.bin.idb	
		BootGuardAcm.idc	

Intel Boot Guard ACM 2016

```
// Validate KEYM manifest
if ( *(_DWORD *)(a1 + 0x15B0) != 'EK__'
    || *(_DWORD *)(a1 + 0x15B4) != '___MY'
    || *(_BYTE *)(a1 + 0x15B8) != 0x10
    || *(_BYTE *)(a1 + 0x15BB) & 0xF0
    || *(_BYTE *)(a1 + 0x15BA) & 0xF0
    || !ValidateBgRsaEntry((BG_RSA_ENTRY *)(a1 + 0x15E0))
    || *(_WORD *)(a1 + 0x15BC) != 0xB
    || *(_WORD *)(a1 + 0x15BE) != 0x20 )
{
    return 0x12C5;
}

// Verify RSA signature
status = Rsa(a1, a1 + 0x15B0, 0x30, (BG_RSA_ENTRY *)(a1 + 0x15E0), a1 + 0x1E38);
```


Intel Boot Guard ACM 2016

```
// Verify SHA256
if ( Sha256((int)a4->RsaPubKey.Modulus, 0x100, (int)v9, a1 + 0x100, a1) )
    return 1;
if ( !CompareMemory((int)v9, a5, 0x20u) )
    return 2;

// Verify signature
if ( sub_1FE7(0x800, a1 + 0x5B0) )
    return 3;
if (!sub_3018(a1 + 0x330, 0x40, 1, &a4->RsaPubKey.Exponent))
    return 4;
if (!sub_3018(v5 + 0x370, 0x240, 0x40, a4->RsaPubKey.Modulus);
    return 5;
if ( sub_1E6D(v7, v12, v5 + 0x5B0) )
    return 6;
if ( sub_2FB6(a2, a3, a4->RsaSig.Sig, &v11, v5 + 0x5B0, v10, v5) )
    return 7;
```


Intel Boot Guard ACM 2017

```
// Validate KEYM manifest
if ( *(_DWORD *)(a1 + 0x15B0) != 'EK__'
    || *(_DWORD *)(a1 + 0x15B4) != '__MY'
    || *(_BYTE *)(a1 + 0x15B8) != 0x10
    || *(_BYTE *)(a1 + 0x15BB) & 0xF0
    || *(_BYTE *)(a1 + 0x15BA) & 0xF0
    || !ValidateBgRsaEntry((BG_RSA_ENTRY *)(a1 + 0x15E0))
    || *(_WORD *)(a1 + 0x15BC) != 0xB
    || *(_WORD *)(a1 + 0x15BE) != 0x20 )
{
    return 0x12C5;
}

// Verify RSA signature
status = Rsa(a1, a1 + 0x15B0, 0x30, (BG_RSA_ENTRY *)(a1 + 0x15E0), a1 + 0x1E38, TRUE); <-- Flag was
added
```


Intel Boot Guard ACM 2017

```
// Verify SHAH256
if ( Sha256Init(a1 + 0x100) )
    return 1;
if ( Sha256Calc((int)a4->RsaPubKey.Modulus, 0x100, v7, a1)
    || Flag == 1 && Sha256Calc((int)&a4->RsaPubKey.Exponent, 4, v7, a1) <-- RSA exponent hashing
    || Sha256Out((int)v12, v7, a1) )
{
    return 1;
}
if ( !CompareMemory((int)v12, a5, 0x20u) )
    return 2;

// Verify signature
...
```


Downgrading microcode

[INTEL-SA-00264](#)

Updating Microcode in UEFI BIOS

- Updates are to improve stability, performance and apply security fixes
- Updates should be loaded each time CPU is powered on, this means after S₃ (Sleep) / S₄ (Hibernation) / S₅ (Shutdown) modes
- Far not always updates can be loaded by CPU from FIT
- Updates that requires something special (like initialized DRAM) has to be loaded by the BIOS as early as possible from the moment conditions are satisfied
- Updates should be loaded on each CPU core separately

Microcode Update loading process

update_microcode:

```
mov rcx, 79h      ; IA32_BIOS_UPDATE_TRIGGER in RCX
xor rax, rax      ; clear RAX
xor rbx, rbx      ; clear RBX
mov rax, MicrocodeUpdate ; Linear address of the microcode update
add rax, 48h      ; Offset of Update Data in the Update
xor rdx, rdx      ; Zero RDX
wrmsr             ; trigger the microcode update
```

check_update_revision:

```
mov rcx, 08bh     ; IA32_BIOS_SIGN_ID
rdmsr             ; read MSR, Update Revision will be in RDX
```


Normal Boot. Step 1. CpuPei

```
// Find the appropriate MCU in FIT
MicrocodeAddr = FindMCUinFIT ();

if (MicrocodeAddr != NULL) {
    MicrocodeSize = ((MICROCODE_UPDATE_HEADER *) MicrocodeAddr)->TotalSize;

    // Copy the MCU from the mapped SPI flash memory into RAM
    Status = (*PeiServices)->AllocatePages ( ... , EFI_SIZE_TO_PAGES (MicrocodeSize), &MicrocodeBuffer);
    if (!EFI_ERROR (Status)) {
        (*PeiServices)->CopyMem (MicrocodeBuffer, MicrocodeAddr, MicrocodeSize);

        // Save this pointer into a HOB
        Status = (*PeiServices)->CreateHob ( ... , &UcodeHob);
        if (!EFI_ERROR (Status)) {
            AmiUcodeHobGuid = EFI_GUID ("94567C6F-F7A9-4229-1330-FE11CCAB3A11");
            memcpy (&UcodeHob->EfiHobGuidType.Name, &AmiUcodeHobGuid, sizeof(EFI_GUID));

            UcodeHob->UcodeAddr = MicrocodeBuffer;
        }
    }
}
```


Normal Boot. Step 2. PlatformInit

- Later the microcode update loader finds this HOB
- Retrieves the MCU buffer address
- Updates CPU microcode with it

Normal Boot. Step 3. CpuSpSmi

```
// Find the MCU HOB and retrieve a saved MCU address
UcodeHob = (AMI_UCODE_HOB *) GetEfiConfigurationTable (pSystemTable, &HobListGuid);

if (UcodeHob != NULL) {
    Status = FindNextHobByGuid (&gAmiUcodeHobGuid, &UcodeHob);

    if (Status == EFI_SUCCESS && UcodeHob->UcodeAddr != NULL && UcodeHob->UcodeAddr != 0xFFFF) {
        gMicrocodeStart = UcodeHob->UcodeAddr;
    }
}

...

// Copy the applied MCU into SMRAM (to protect it from being replaced by OS)
if (gMicrocodeStart != NULL && ((MICROCODE_UPDATE_HEADER *) gMicrocodeStart)->HeaderVersion == 1) {
    UcodeSize = ((MICROCODE_UPDATE_HEADER *) gMicrocodeStart)->TotalSize;

    Status = pSmst->SmmAllocatePages ( ... , EFI_SIZE_TO_PAGES (UcodeSize), &SmramUcodeAddr);
    if (!EFI_ERROR (Status)) {
        memcpy (SmramUcodeAddr, gMicrocodeStart, UcodeSize);
    }
}
```


Normal Boot. Step 3. CpuSpSmi

```
gIntUcodeVarGuid = EFI_GUID ("eda41d22-7729-5b91-b3ee-ba619921cefa");
```

```
...
```

```
// Save its address into the 'IntUcode' EFI variable
```

```
IntUcodeVarData.Version = 1;
```

```
IntUcodeVarData.UcodeAddr = SmmUcodeAddr;
```

```
IntUcodeVarData.Unknown = 0;
```

```
IntUcodeVarData.Unknown2 = 0;
```

```
Status = pRuntimeServices->SetVariable (L"IntUcode", &gIntUcodeVarGuid,  
                                         EFI_VARIABLE_NON_VOLATILE |  
                                         EFI_VARIABLE_BOOTSERVICE_ACCESS |  
                                         EFI_VARIABLE_RUNTIME_ACCESS,  
                                         sizeof(IntUcodeVarData), &IntUcodeVarData);
```


Waking from S3. Step 1. CpuPei

```
// Instead of searching for the MCU again, get the pointer from 'IntUcode' EFI variable
Status = ReadOnlyVariable2->GetVariable ( ... , "IntUcode", & gIntUcodeVarGuid, NULL,
                                          &VarSize, &IntUcodeVarData);

if (!EFI_ERROR (Status)) {
    MicrocodeAddr = IntUcodeVarData.UcodeAddr;

    Status = (*PeiServices)->CreateHob ( ... , &UcodeHob);
    if (!EFI_ERROR (Status)) {
        AmiUcodeHobGuid = EFI_GUID ("94567C6F-F7A9-4229-1330-FE11CCAB3A11");
        memcpy (&UcodeHob->EfiHobGuidType.Name, &AmiUcodeHobGuid, sizeof(EFI_GUID));

        UcodeHob->uCodeAddr = MicrocodeAddr;
    }
}
```


Waking from S3. Step 2. PlatformInit

- Later the microcode update loader finds this HOB
- Retrieves the MCU buffer address
- Updates CPU microcode with it

Microcode Downgrade

This specific allows an attacker:

- to load an old microcode update capsule into memory
- make the 'IntUcode' EFI variable to point to it
- perform Sleep/Wake-up cycle

The system will be booted with the attacker-provided microcode (if it was valid and passed the integrity check, of course)

Microcode Downgrade

- 2019 version of MCU of CPU ID 0x806EA

CPU ID	N/A	000806EA
PATCH ID	N/A	00000096

- Downgraded to 2018 version

CPU ID	N/A	000806EA
PATCH ID	N/A	00000084

Side channel attacks

- Get rid of fixes (side channel attacks)
- Most of these attacks – extremely hard to apply in the wild
- Have never been spotted, however there's not much of detection tools:
 - [SCADET](#) by Majid Sabbagh
- [Introduction to software-based microarchitectural side-channel attacks](#) by Alexander Romyantsev
- [A New Memory Type Against Speculative Side-Channel Attacks](#) by [@IntelSTORM](#)

Debug capabilities

- Unlock debug capabilities
- Get rid of [INTEL-SA-00073](#) fix (CVE-2017-5684)
- [Intel DCI Secrets](#) by Maxim Goryachy and Mark Ermolov

Downgrading ACMs

- The ACM authentication is performed by a Microcode
- Older Microcode versions load older ACM (with reduced SVN)
- Downgraded ACM has exploitable 1days which makes vulnerable the technology they support

<https://twitter.com/matrosov/status/1139491430110584832>

**Alex Matrosov**
@matrosov

Follow

Intel microcode downgrade is a huge supply-chain problem. Even after the patch problem still exists in many platforms. Btw ACM's downgrade is also possible (a bit more tricky but downgrade both Microcode + ACM is a key to success).
Great job @flothrone and the team!

Alexander Ermolov @flothrone
Our team (@ttbr0 , @undermarble and me) walks through UEFI BIOS again, as a result:
- 6 Escalation of Privileges to SMM
- microcode downgrade vulnerability, allowing to bypass hardware root-of-trusts.
Details coming soon!
[Show this thread](#)

4:15 AM - 14 Jun 2019

Downgrading ACMs. Intel Boot Guard

- Not encrypted, binary diffing is applicable to find 1 days
- Executed only on startup (prior to BIOS) upon CPU is powered on and released from the RESET state
- ACM does not verify BIOS when waking from S₃ (performance optimizations) except each 12 boot

The implementation of vendor part of trusted boot is a target here.
Plenty of techniques are already in public

Downgrading ACMs. Intel BIOS Guard

- Encrypted, extremely hard to find a fixed issue
- Triggered to run SPI flash operations via CPU MSRs from SMM
- Downgrade is possible if SPI flash write access is gained (at which point further attack is unnecessary)

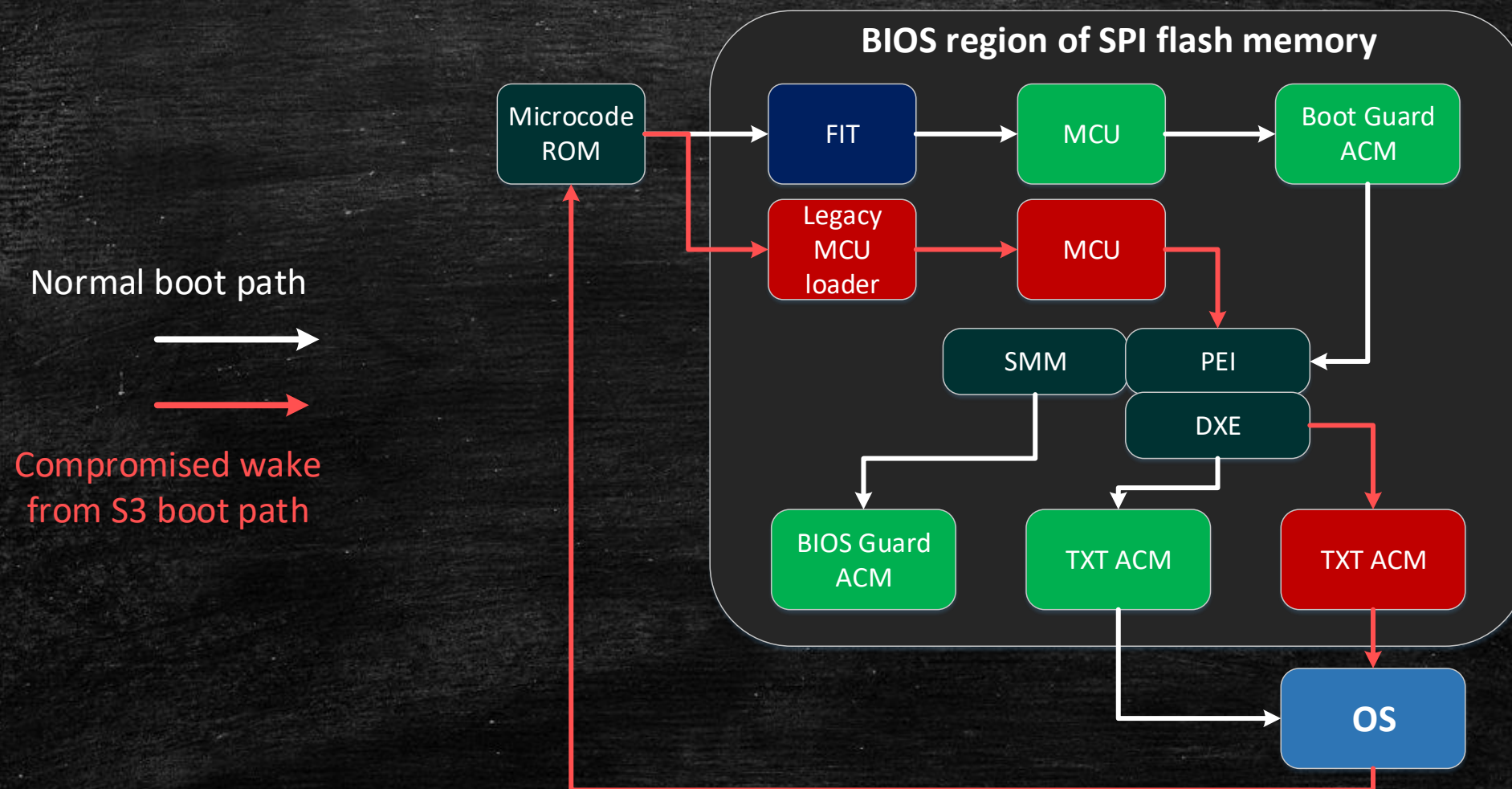
First bypass is already in public:

[Breaking Through Another Side: Bypassing Firmware Security Boundaries from Embedded Controller](#) by Alex Matrsov

Downgrading ACMs. Intel TXT

- Not encrypted, binary diffing is applicable to find a 1 days
- SINIT ACM is a target
- Triggered via GETSEC instruction from BIOS / OS to measure boot chain components
- Address of this ACM is specified in EBX register
- Address doesn't change from boot to boot, so downgrade is possible just by replacing this ACM in memory!
- [INTEL-SA-00035](#)

Downgrading ACMs. Intel TXT



Rewriting memory from S3 bootscript

[INTEL-SA-00264](#)

SmmPcieSataController

```
// Parse the communication buffer
ptr = *(unsigned long long *) CommBuffer;

addr = 0;
data = 0;

if ( ptr )
{
    i = 0;
    entries_num = *(unsigned int *)(*CommBufferSize + ptr - 4);

    if ( entries_num > 0 )
    {
        struct = ptr + 0x10;
        ... // parse the input structure in a loop
    }
}
```


SmmPcieSataController

```
// Parse the input structure in a loop
do
{
    if ( (unsigned int) addr == 'SBSP')
        break;

    command = *(unsigned char*)(struct - 0x10);
    addr     = *(unsigned long long*)(struct - 8);
    width    = *(unsigned long long*)struct;
    data     = *(unsigned long long*)(struct + 8);

    // Add MEMWRITE command to S3 bootscript
    if ( command == 1 )
        gEfiS3SmmSaveStateProtocol->Write(gEfiS3SmmSaveStateProtocol, 4, width, addr, 1, &data);

    // Add PCICFGWRITE command to S3 bootscript
    if ( command == 2 )
        gEfiS3SmmSaveStateProtocol->Write(gEfiS3SmmSaveStateProtocol, 2, width, addr, 1, &data);

    ++i;
    struct += 0x20;
}
while ( i < entries_num );
```


#Mitigations

- Intel SGX
 - does not check MCU SVN when leaving S3
- Better use of SVN values
- Protect 'IntUcode' EFI variable (mark as read-only and close from runtime access)
 - Could be bypassed if an attacker manages to run arbitrary code in SMM
- Make an OS to update the Microcode to the latest version
 - Process could be already compromised at the moment of validating the update version
- Supply only the updates which could be loaded from FIT

#Takeaways

- Supply chain problems
- The problem in a basic component compromises all technologies it serves as a Root-of-Trust
- The full impact is yet to discover

Thank you
