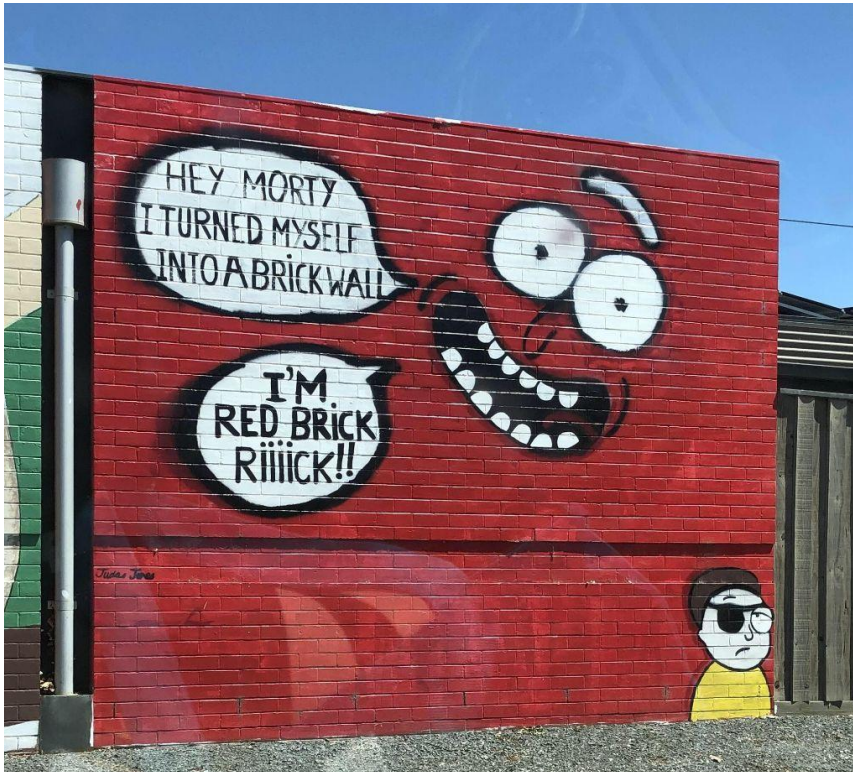
The UEFI Threat: Or How I Can "Permanently" Brick Your Computer



Paul Asadoorian - Shmoocon 2023

What are your requirements for purchasing a new computer?

Your computer comes with software (firmware) pre-installed

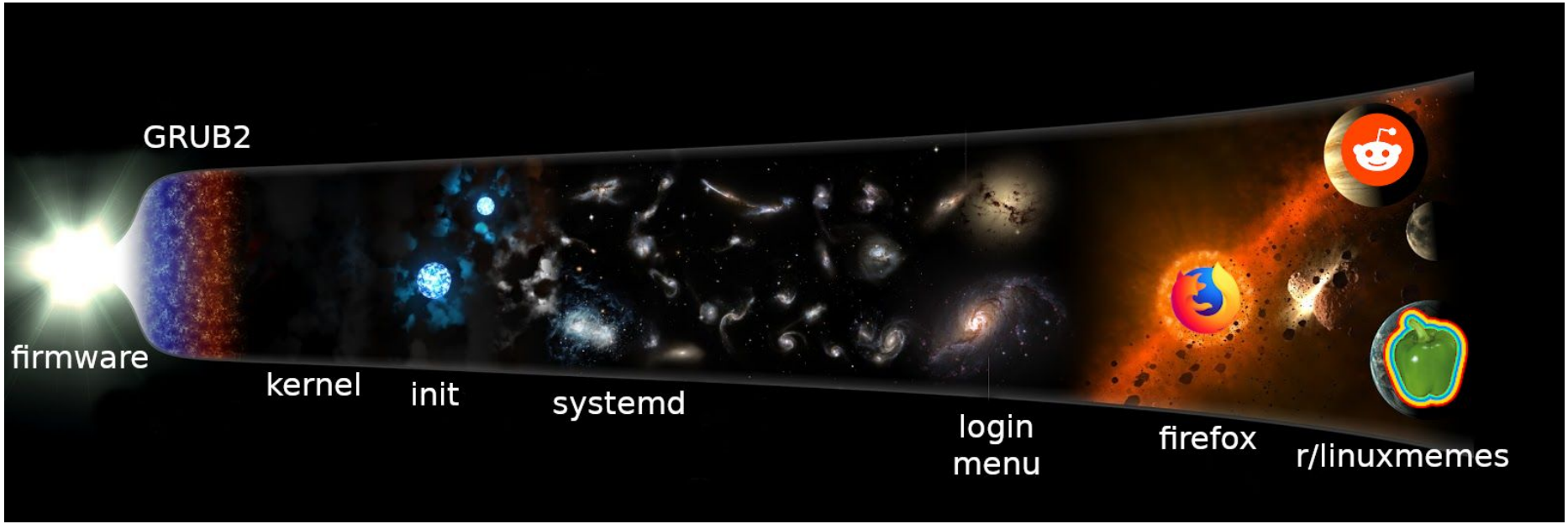
Unlike most other software (bootloader, kernel, operating system, or application software):

1 - It is not easy to modify or update

2 - It has the highest levels of privilege on your system

3- Most users don’t even know its there let alone interact/customize it

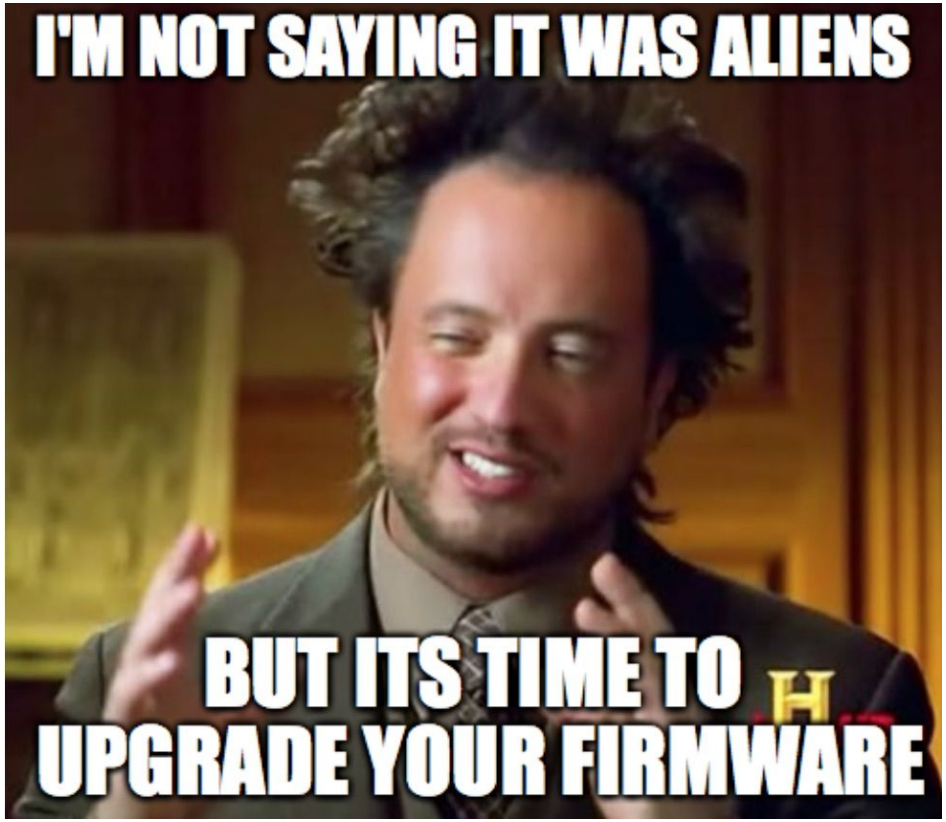
UEFI operates with the most privileges on your computer*.



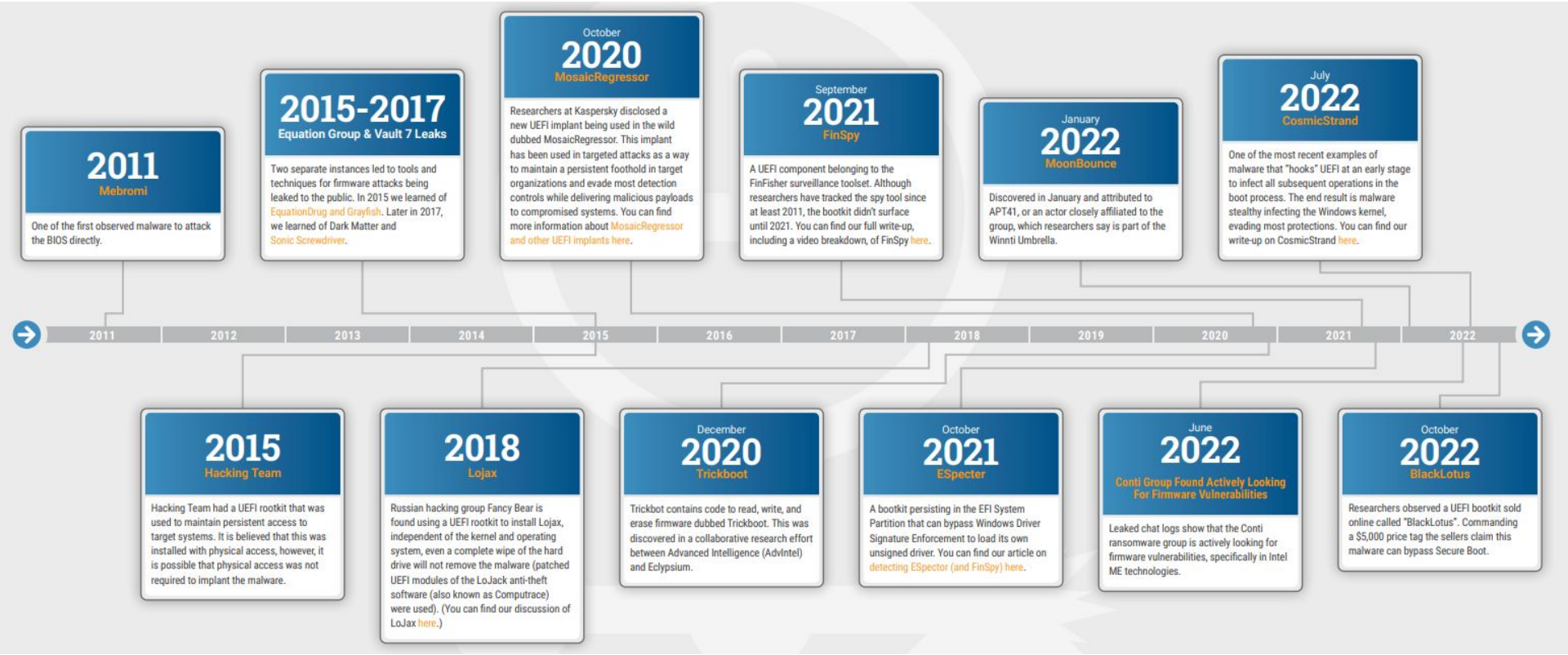
**Except for Intel CSME/ME…*

Firmware is just software that is difficult to program.

Like most other software it has security vulnerabilities and configuration issues

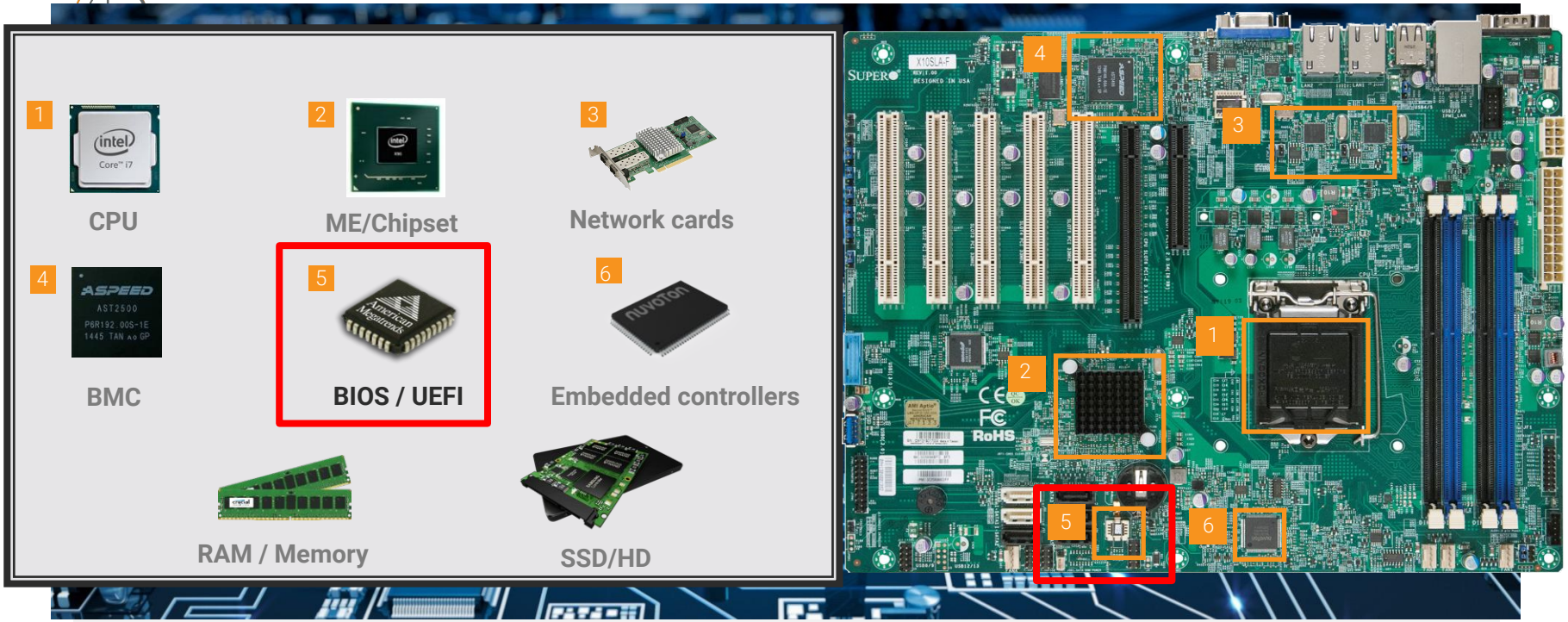


But you probably won’t…



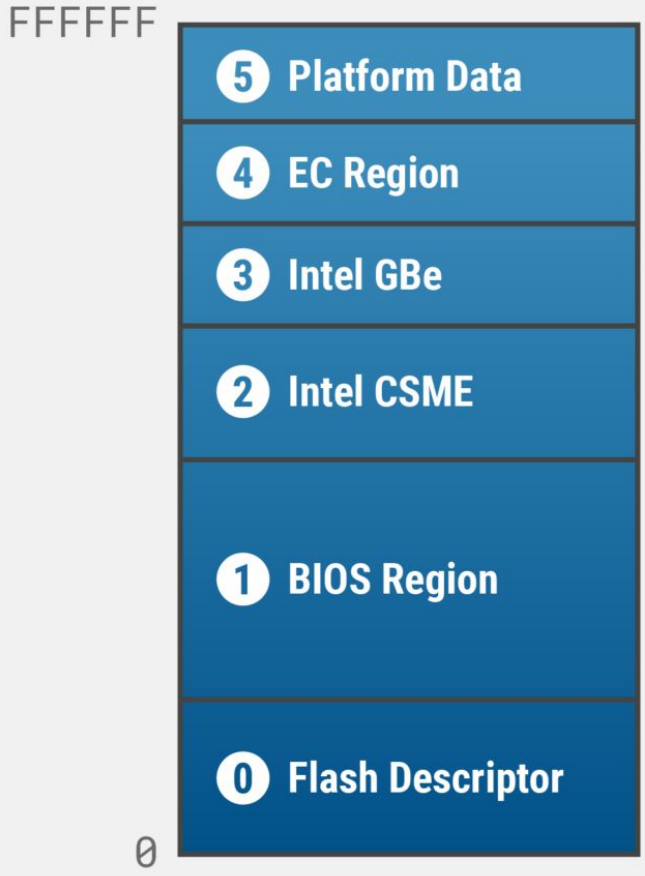
https://eclypsium.com/2022/10/20/firmware-attacks-an-endpoint-timeline/

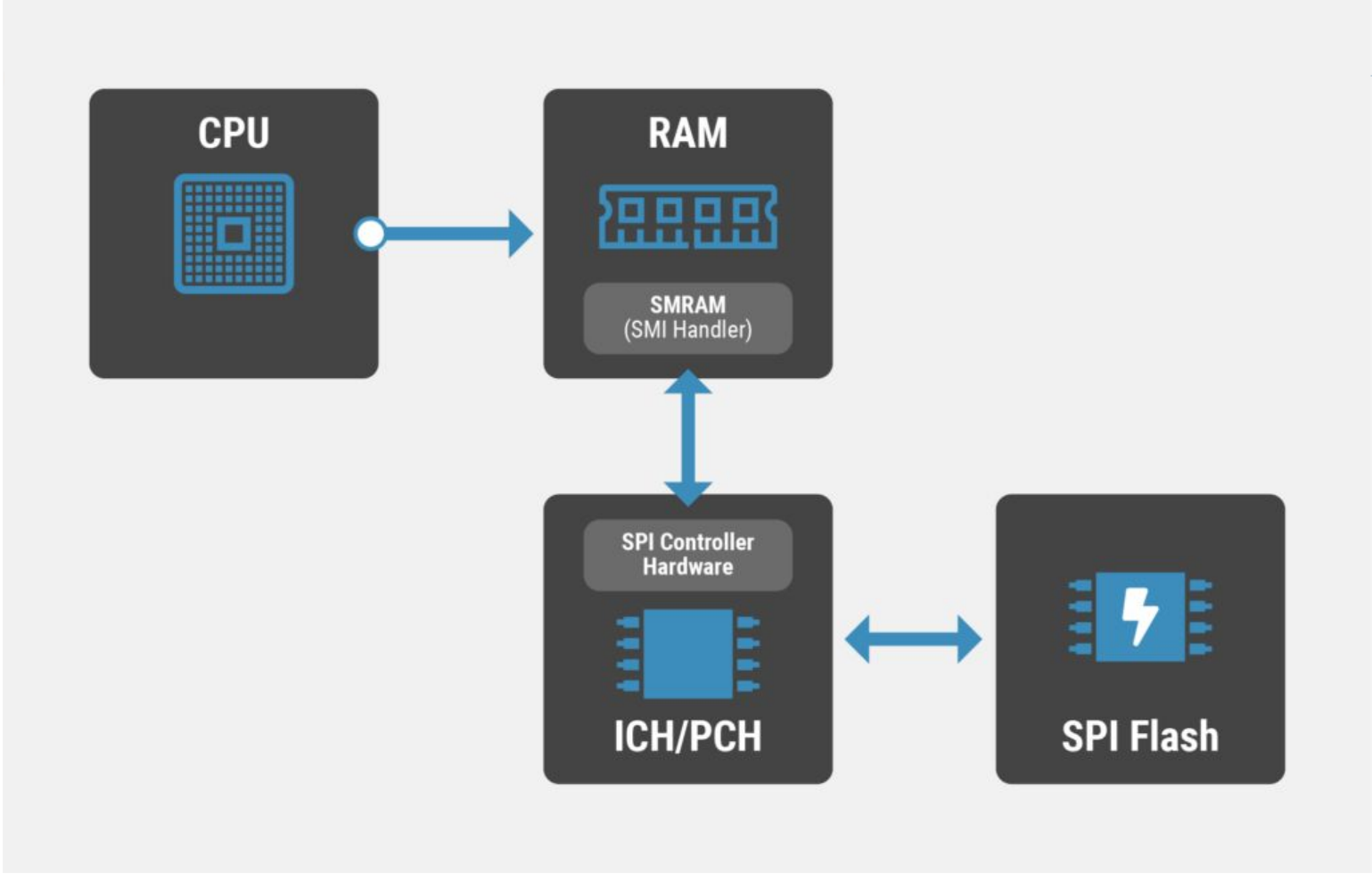
I don’t believe you. Isn’t this a manufacturer/OEM problem?



Which manufacturer and which firmware/software?

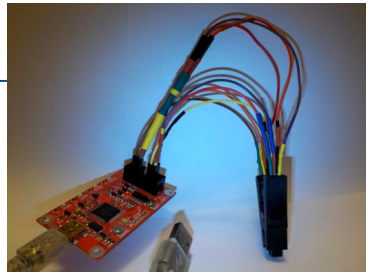
Typical 16MB total





https://opensecuritytraining.info/IntroBIOS_files/Day2_03_Advanced%20x86%20-%20BIOS%20and%20SMM%20Internals%20-%20SPI%20Flash%20Protection%20Mechanisms.pdf

Protecting System Firmware Storage



Protecting the contents of the SPI flash is tricky business as there are multiple mechanisms to control writes:

1. **All of the settings and registers could (and do) change depending on which chipset in in your computer**
2. “SMM BIOS Write Protection” (SMM_BWP) - writes can only allowed in SMM
3. Flash Descriptor region defines how the SPI flash is laid out and some access controls
4. Settings are controlled by the OEM (i.e. you typically can’t go trying to change these settings yourself as you will brick you own system or they will just have no affect)

https://eclypsium.com/2022/09/19/firmware-security-realizations-part-3-spi-write-protections/

SPI Flash Protections

These are four (more popular) ways to protect the SPI flash, depending on your hardware/chipset:

1. **The Flash Descriptor**
2. **Global Write Protections**
3. **BIOS Range Write Protection**
4. **Flash Configuration Lockdown**

```
$  sudo ./chipsec_util.py spi dump fd.bin
```

<- Dumping contents of SPI flash to file

```
################################################################
##                                                            ##
##  CHIPSEC: Platform Hardware Security Assessment Framework  ##
##                                                            ##
################################################################
[CHIPSEC] Version  : 1.10.0
[CHIPSEC] Arguments: spi dump fd.bin
```

#1 - The Flash Descriptor

```
****** Chipsec Linux Kernel module is licensed under GPL 2.0
[CHIPSEC] API mode: using CHIPSEC kernel module API
[CHIPSEC] OS      : Linux 5.4.0-125-generic #141-Ubuntu SMP Wed Aug 10 13:42:03 UTC 2022 x86_64
[CHIPSEC] Python  : 3.8.10 (64-bit)
[CHIPSEC] Helper  : LinuxHelper [redacted]/chipsec/helper/linux/chipsec.ko)
[CHIPSEC] Platform: Bay Trail SoC
[CHIPSEC]    CPUID: 30679
[CHIPSEC]      VID: 8086
[CHIPSEC]      DID: 0F00
[CHIPSEC]      RID: 11
[CHIPSEC] Executing command 'spi' with args ['dump', 'fd.bin']

[CHIPSEC] Dumping entire SPI flash memory to 'fd.bin'
[CHIPSEC] it may take a few minutes (use DEBUG or VERBOSE logger options to see progress)
[CHIPSEC] BIOS region: base = 0x00300000, limit = 0x007FFFFF
[CHIPSEC] Dumping 0x00800000 bytes (to the end of BIOS region)
[spi] reading 0x800000 bytes from SPI at FLA = 0x0 (in 131072 0x40-byte chunks + 0x0-byte remainder)
[CHIPSEC] Completed SPI flash dump to 'fd.bin'
[CHIPSEC] (spi) time elapsed 106.251
```

#1 - The Flash Descriptor

```
$ sudo ./chipsec_util.py spidesc fd.bin


Flash Regions
-------------------------------------------------------
 Region                | FLREGx    | Base     | Limit
-------------------------------------------------------
0 Flash Descriptor     | 00000000  | 00000000 | 00000000
1 BIOS                 | 07FF0300  | 00300000 | 007FF000
2 Intel ME             | 02FF0001  | 00001000 | 002FF000

+ 0x0060 Master Section:
=======================================================
+ 0x0060 FLMSTR1   : 0xFFFF0000
+ 0x0064 FLMSTR2   : 0xFFFF0000

Master Read/Write Access to Flash Regions
-------------------------------------------------------
 Region                | CPU       | ME
-------------------------------------------------------
 0 Flash Descriptor    | RW        | RW
 1 BIOS                | RW        | RW
 2 Intel ME            | RW        | RW
```

```
$ sudo ./chipsec_main.py -m common.bios_wp
<snip>
[x][ ================================================================================
[x][ Module: BIOS Region Write Protection
[x][ ================================================================================
[*] BC = 0x00000000 << BIOS Control (b:d.f 00:31.5 + 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
    [06] BBS              = 0 << Boot BIOS Strap
    [07] BILD             = 1 << BIOS Interface Lock Down
[-] BIOS region write protection is disabled!

[*] BIOS Region: Base = 0x00300000, Limit = 0x00FFFFFF
SPI Protected Ranges
------------------------------------------------------------
PRx (offset) | Value    | Base     | Limit    | WP? | RP?
------------------------------------------------------------
PR0 (84)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR1 (88)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR2 (8C)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR3 (90)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR4 (94)     | 00000000 | 00000000 | 00000000 | 0   | 0

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

[!] BIOS should enable all available SMM based write protection mechanisms.
[!] Or configure SPI protected ranges to protect the entire BIOS region.
[-] FAILED: BIOS is NOT protected completely
```

#2 - BIOS Write Protections

#3 - BIOS (and other) Range Write Protection

```
$ sudo ./chipsec_main.py -m common.spi_lock

[*] Running module: chipsec.modules.common.spi_lock
[x][ ================================================================================
[x][ Module: SPI Flash Controller Configuration Locks
[x][ ================================================================================
[*] HSFS = 0x3F00E800 << Hardware Sequencing Flash Status Register (SPIBAR + 0x4)
    [00] FDONE            = 0 << Flash Cycle Done
    [01] FCERR            = 0 << Flash Cycle Error
    [02] AEL              = 0 << Access Error Log
    [05] SCIP             = 0 << SPI cycle in progress
    [11] WRSDIS           = 1 << Write status disable
    [12] PR34LKD          = 0 << PRR3 PRR4 Lock-Down
    [13] FDOPSS           = 1 << Flash Descriptor Override Pin-Strap Status
    [14] FDV              = 1 << Flash Descriptor Valid
    [15] FLOCKDN          = 1 << Flash Configuration Lock-Down
    [16] FGO              = 0 << Flash cycle go
    [17] FCYCLE           = 0 << Flash Cycle Type
    [21] WET              = 0 << Write Enable Type
    [24] FDBC             = 3F << Flash Data Byte Count
    [31] FSMIE            = 0 << Flash SPI SMI# Enable
[+] SPI write status disable set.
[+] SPI Flash Controller configuration is locked
[+] PASSED: SPI Flash Controller locked correctly.
```

#4 - Flash Configuration Lockdown

And now, some research…

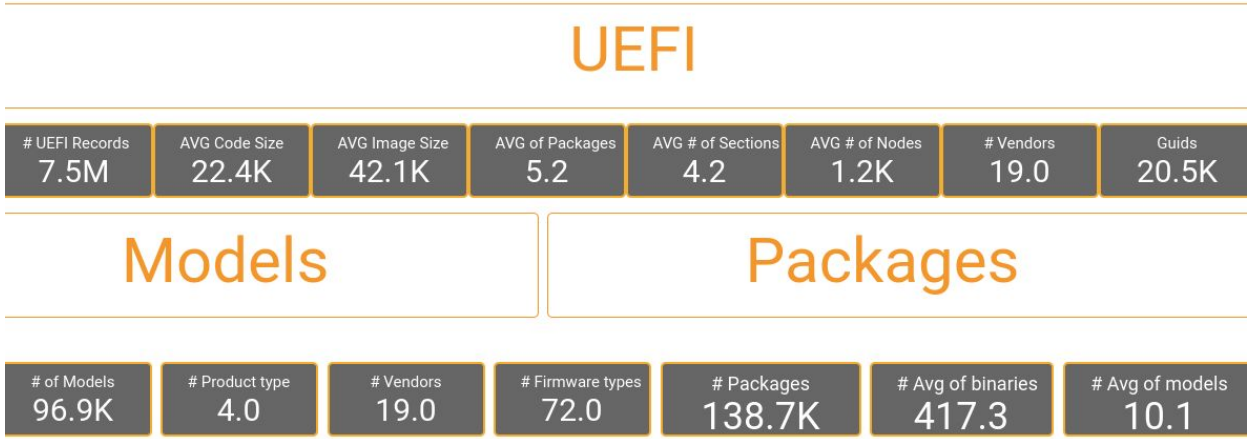
<- Me when I discovered some of the research that had already been done

Largest Dataset

Across 138,000+ firmware packages there exists about 198,000 CVEs

CWE-119: **Improper Restriction of Operations within the Bounds of a Memory Buffer** is the most popular CWE

This includes all firmware that we collect and analyze, not just UEFI…



2017 Research (Previously Unreleased)

The team analyzed 32,000+ firmware images and found:

~3,417 update images corresponding to ~502 models from 9 manufacturers appear to be lacking basic firmware protections

MSI & Gigabyte account for majority (2,578 images ~ 345 models)

It's trivial to install firmware implants or brick such systems (specific to SPI descriptor access checks)

Manufacturer	Vulnerable firmware images	Vulnerable models
Acer	0 - 2	0 – 2
ASRock	73	**~53** models (all older than Skylake)
ASUS	629	**~61** models (all older than Ivy Bridge)
Dell	51	**~11** models (Vostro and Inspiron older than 2014)
Gigabyte	1117 (345 Skylake+)	**~247** models including Skylake (6 Gen Intel Core) or newer
HP	11	**~6**
Intel	0	0
Lenovo	75	**~26** (ThinkServer TS150-550, ThinkCentre/IdeaCentre)
MSI	1461 (495 Skylake+)	**~98** models including Skylake (6 Gen Intel Core) or newer
Total	**3417 (16.1%)**	**~502 models**

2017 Research (Previously Unreleased)

More Updated research

We analyzed **15,083** LIVE unique devices running **781** different firmware versions across **335** different hardware models.

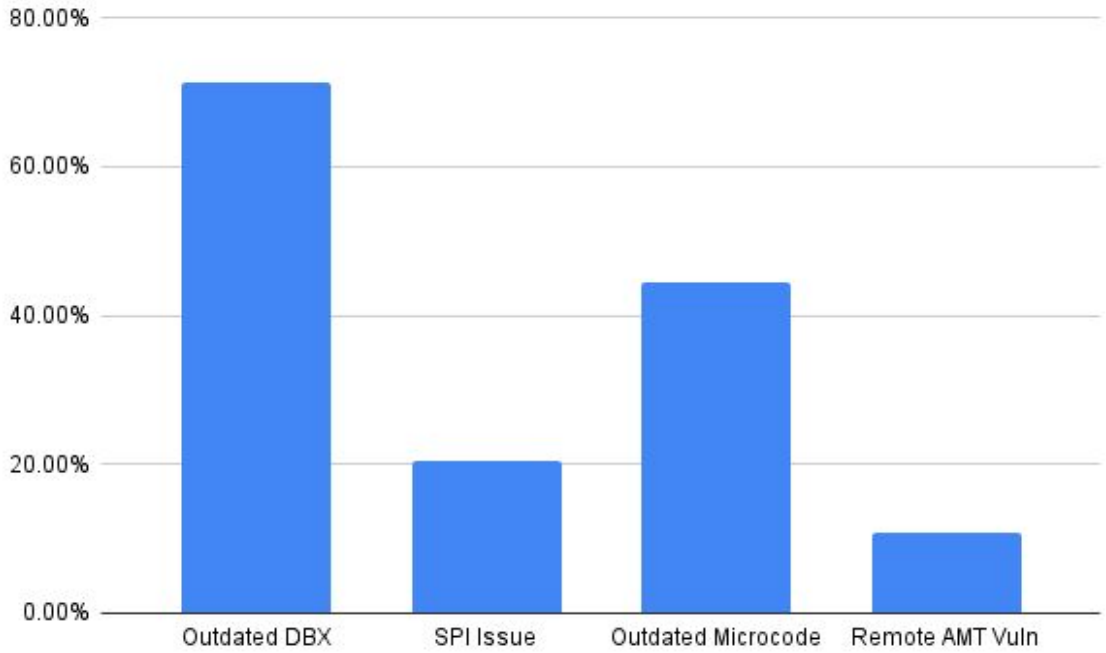
Security checks were performed on the devices at runtime and not by checking the static firmware image

In almost all of them we found a firmware-related vulnerability

Many vulnerabilities are remediated with a firmware update, proving that firmware updates are not being applied

Updated Research

We analyzed 900+ full firmware images (SPI dumps)



Digging a little deeper…

Vendor	# Models	# fw versions	# unique devices	**Top vulns!** (there are many more per vendor)
Dell	79	215	6643	• **BMC default password:** 1846 • **spi_desc:** 1719 • **Outdated DBX:** 1220 • **Debug interfaces enabled**: 1207 • **CVE-2020-0549, CVE-2020-0548, CVE-2020-0545**: >1000 • **Outdated microcode related vulns:** > 800 • etc…
Lenovo	77	274	3480	• **Outdated DBX**: >3000 • **Outdated microcode related vulns**: >2000 • **CVE-2020-24512** and **CVE-2020-24511** > 2500 • **IntelSA00391:** > 2500 • **IntelSA00295:** > 2100 • etc…

Numbers #2

Vendor	# Models	# fw versions	# unique devices	Top vulns!
HP	26	30	34	• **CVE-2020-0528:** 15 • **CVE-2019-11135:** 14 • **CVE-2019-0154:** 14 • etc…
Supermicro	45	143	4394	• **Outdated microcode related vulns:** > 1900 • **TPM_CVE_2017_15361:** 643 • **Pantsdown:** > 500 • **spi_desc:** 130 • **bios_wp**: 72, • **spi_lock:** 51, • etc…

Numbers #3

Vendor	# Models	# fw versions	# unique devices	Top vulns!
Apple	19	30	50	To be filled with data of the json like the others
Asus	11	12	13	To be filled
Microsoft	10	28	96	To be filled
Intel	4	10	13	To be filled
and another one…	and another one…	another one…	another one…	another ones…

Vendors

These are just the most significative samples. Full lists of vendors and vulnerabilities found are much larger.

Full list of vendors analyzed with at least a sample device and vulnerable:

- Lenovo
- Dell
- HP
- Apple
- Supermicro
- Microsoft
- Intel
- Acer
- Quanta
- MSI
- Fujitsu
- LG
- Abaco systems
- HPE
- IBM
- Inspur
- Toshiba
- Asus
- Samsung
- Maxsun
- Gigabyte
- MouseComputer

Can provide full analysis file if you wish (?)

Checking My Own Stuff

Purchased July 2020 and again in July 2021, same BIOS versions on each and no update from OEM!

```
[redacted]:~/chipsec$ sudo dmidecode -t bios
# dmidecode 3.2
Getting SMBIOS data from sysfs.
SMBIOS 3.0.0 present.

Handle 0x0000, DMI type 0, 24 bytes
BIOS Information
        Vendor: American Megatrends Inc.
        Version: 5.6.5
        Release Date: 12/26/2018
```

```
[*] Running module: chipsec.modules.common.spi_desc
[x][ ================================================================================
[x][ Module: SPI Flash Region Access Control
[x][ ================================================================================
[*] FRAP = 0x0000FFFF << SPI Flash Regions Access Permissions Register (SPIBAR + 0x50)
    [00] BRRA             = FF << BIOS Region Read Access
    [08] BRWA             = FF << BIOS Region Write Access
    [16] BMRAG            = 0 << BIOS Master Read Access Grant
    [24] BMWAG            = 0 << BIOS Master Write Access Grant
[*] Software access to SPI flash regions: read = 0xFF, write = 0xFF
[-] Software has write access to SPI flash descriptor

[-] FAILED: SPI flash permissions allow SW to write flash descriptor
[-] System may be using alternative protection by including descriptor region in SPI Protected Range Registers
```

Checking My Own Stuff

```
# dmidecode 3.2
Getting SMBIOS data from sysfs.
SMBIOS 2.8 present.

Handle 0x0000, DMI type 0, 26 bytes
BIOS Information
        Vendor: American Megatrends Inc.
        Version: 1.D0
        Release Date: 01/19/2021
```

Title	Version	Release Date
AMI BIOS	7B98v1E	2022-11-08

Description:
- Windows 11 Supported.
- Update CPU Micro code.
- Change the default setting of Secure Boot.
- Improve Intel DG2 VGA Card compatibility.

Title	Version	Release Date
AMI BIOS	7B98v1D	2021-02-08

Description:
- Support Re-size BAR function to enhance GPU performance.
- Support new audio device.

Purchased July 2021, still in manufacturing mode!

```
[*] Running module: chipsec.modules.common.me_mfg_mode
[x][ ================================================================================
[x][ Module: ME Manufacturing Mode
[x][ ================================================================================
[-] FAILED: ME is in Manufacturing Mode
```

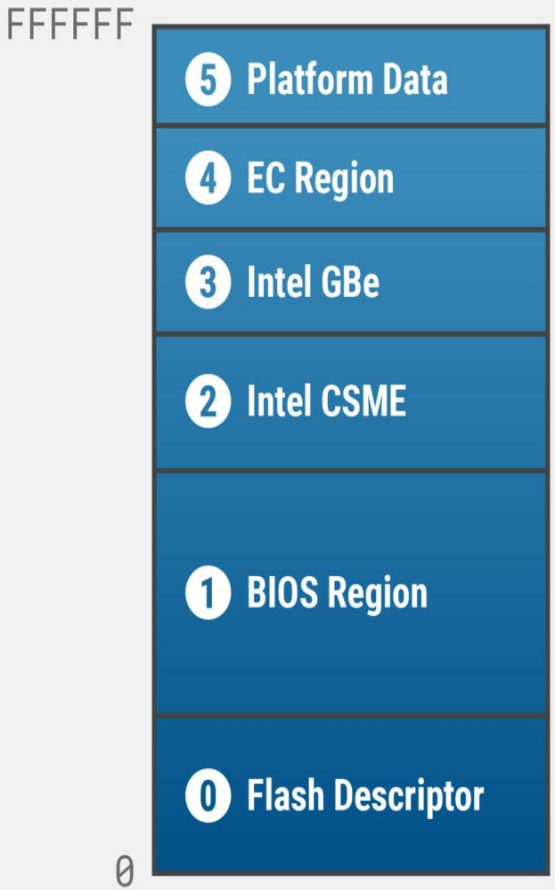
Bad Combination…

```
[*] BC = 0x00000888 << BIOS Control (b:d.f 00:31.5 + 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
    [06] BBS              = 0 << Boot BIOS Strap
    [07] BILD             = 1 << BIOS Interface Lock Down
[-] BIOS region write protection is disabled!

[*] BIOS Region: Base = 0x00300000, Limit = 0x00FFFFFF
SPI Protected Ranges
------------------------------------------------------------
PRx (offset) | Value    | Base     | Limit    | WP? | RP?
------------------------------------------------------------
PR0 (84)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR1 (88)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR2 (8C)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR3 (90)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR4 (94)     | 00000000 | 00000000 | 00000000 | 0   | 0

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

[-] BIOS should enable all available SMM based write protection mechanisms.
[-] Or configure SPI protected ranges to protect the entire BIOS region.
[-] FAILED: BIOS is NOT protected completely
```

```
SPI Flash Region Access Permissions
------------------------------------------------------------

BIOS Region Write Access Grant (00):
  FREG0_FLASHD: 0
  FREG1_BIOS  : 0
  FREG2_ME    : 0
  FREG3_GBE   : 0
  FREG4_PD    : 0
  FREG5       : 0
BIOS Region Read Access Grant (00):
  FREG0_FLASHD: 0
  FREG1_BIOS  : 0
  FREG2_ME    : 0
  FREG3_GBE   : 0
  FREG4_PD    : 0
  FREG5       : 0
BIOS Region Write Access (FF):
  FREG0_FLASHD: 1
  FREG1_BIOS  : 1
  FREG2_ME    : 1
  FREG3_GBE   : 1
  FREG4_PD    : 1
  FREG5       : 1
BIOS Region Read Access (FF):
  FREG0_FLASHD: 1
  FREG1_BIOS  : 1
  FREG2_ME    : 1
  FREG3_GBE   : 1
  FREG4_PD    : 1
  FREG5       : 1
[*] Software has write access to Platform Data region in SPI flash (it's platform specific)
WARNING: Software has write access to GBe region in SPI flash
[-] Software has write access to SPI flash descriptor
[-] Software has write access to Management Engine (ME) region in SPI flash
[-] FAILED: SPI Flash Region Access Permissions are not programmed securely in flash descriptor
[-] System may be using alternative protection by including descriptor region in SPI Protected Range Registers
[-] If using alternative protections, this can be considered a WARNING
```

Let's talk about “bad things”

Flash Descriptor Region

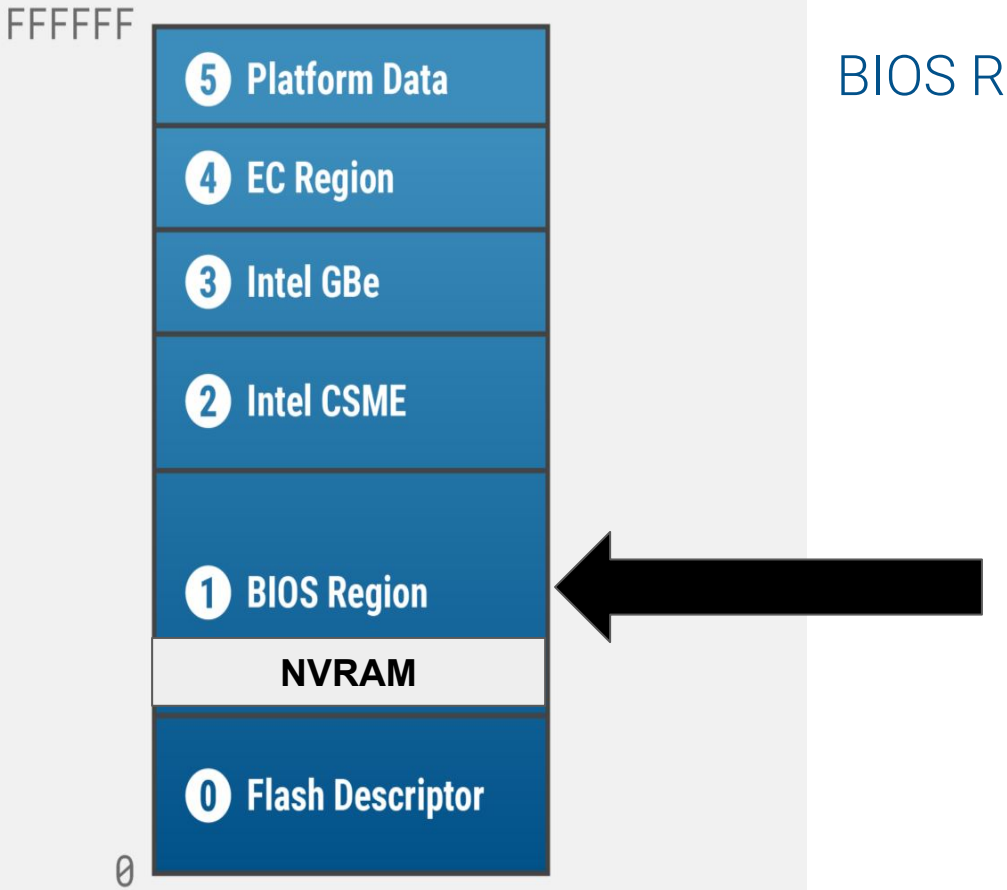


Contains the “map” for the other regions

Also controls a set of permissions for writing to the regions, including itself.

If left writable, you can make your own rules!

Also corrupting or overwriting it would be bricking-level bad.

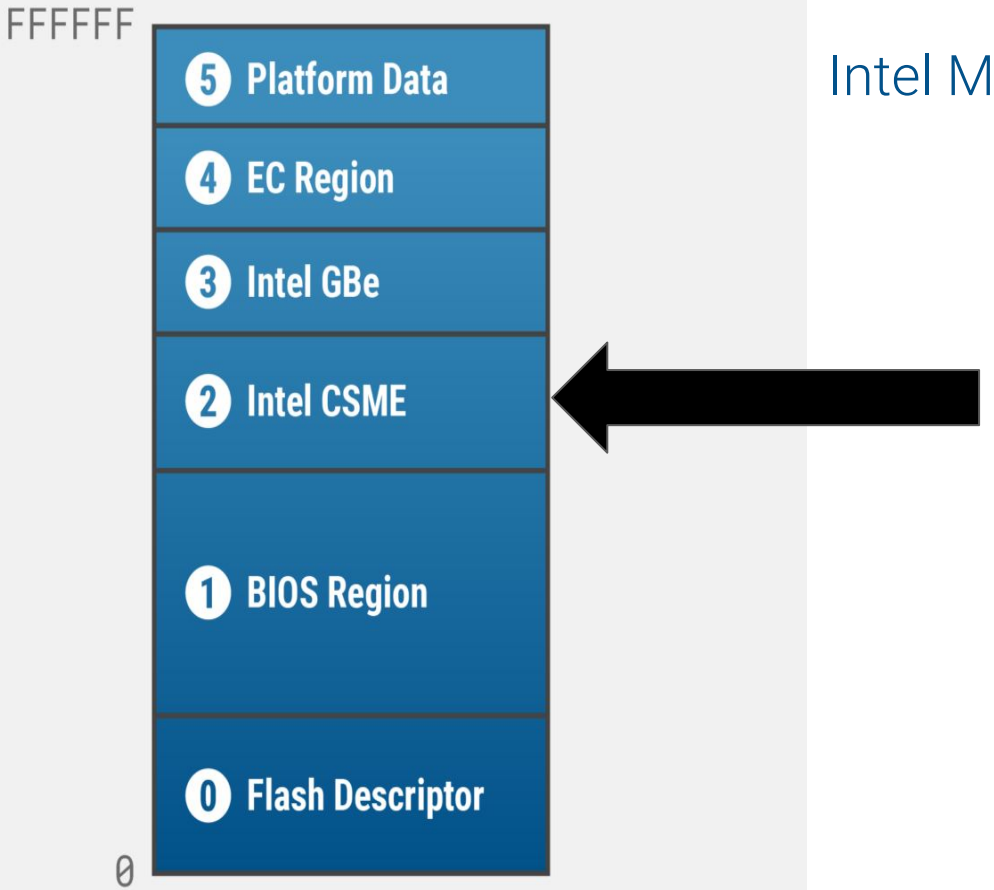


BIOS Region

Typically has permissions and rules to prevent overwriting or tampering, but has to remain writable in select sub-regions.

Variables stored in NVRAM can have different attributes to control access/writing (Runtime, Authenticated, etc…).

Modifying or deleting variables is sometimes all it takes to “brick” a system (depending on the attributes)

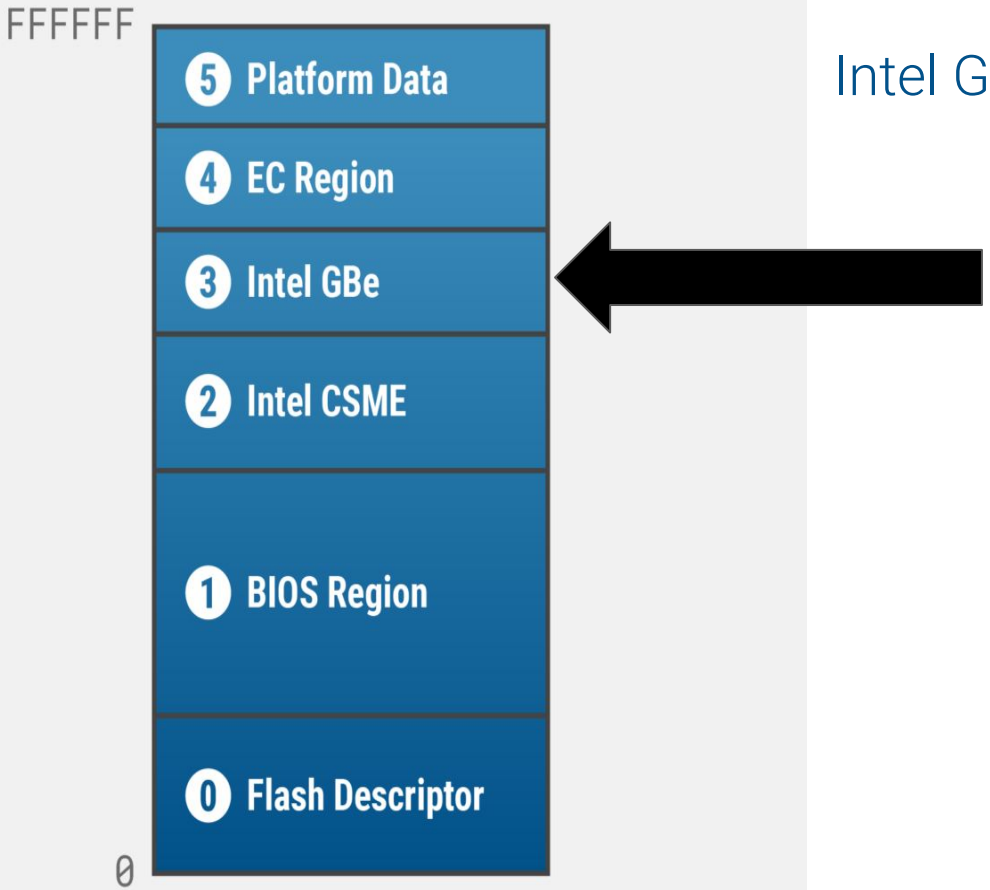


Intel ME/CSME

Not only for management, controls system critical functionality.

Modification is possible as each individual firmware module is signed. Some pair this down to remove functionality (I do not recommended)

Modifying or deleting will most likely “brick” the system, or at the very least cause unexpected behavior (like turning off after 30 minutes)



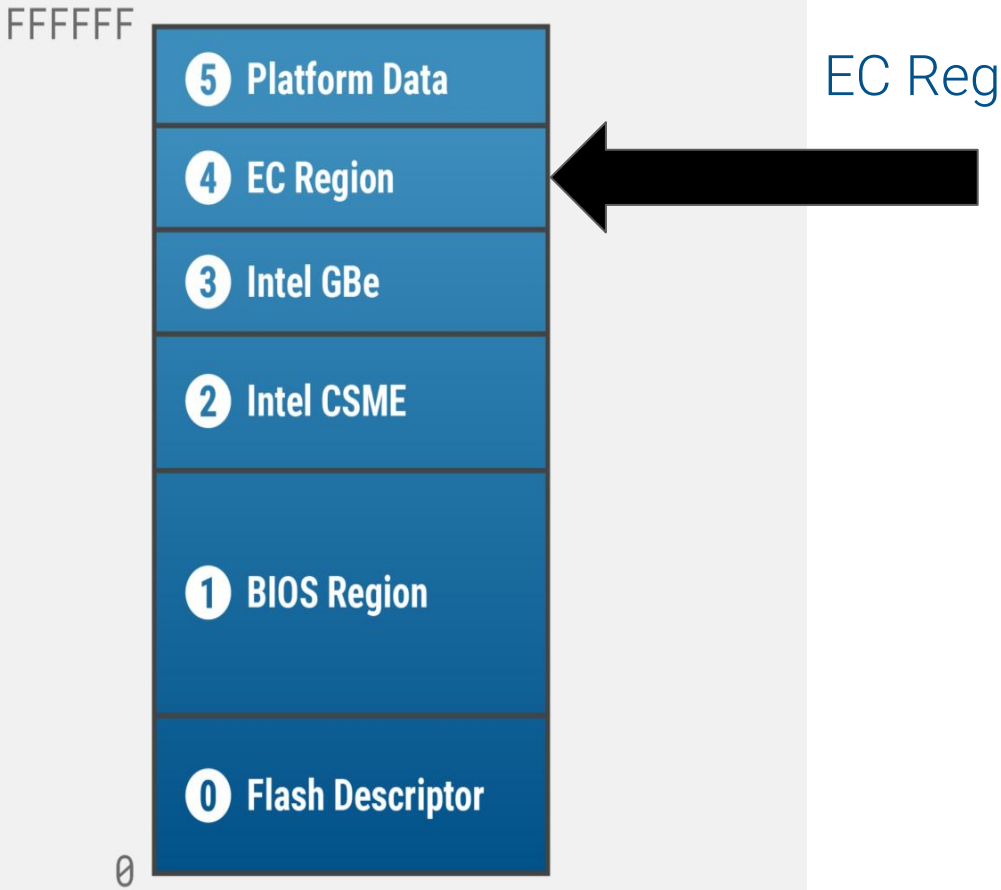
Intel GBe Region

Stores configuration information for Ethernet adapters (e.g. your MAC address).

If you wipe it out, your Ethernet adapter may not work properly or at all.

Curious if this will also impact AMT, as you could disrupt all ability to connect remotely to a machine.

Recovery would involve the OEM (or a backup if you have one or can get one from the vendor).



EC Region

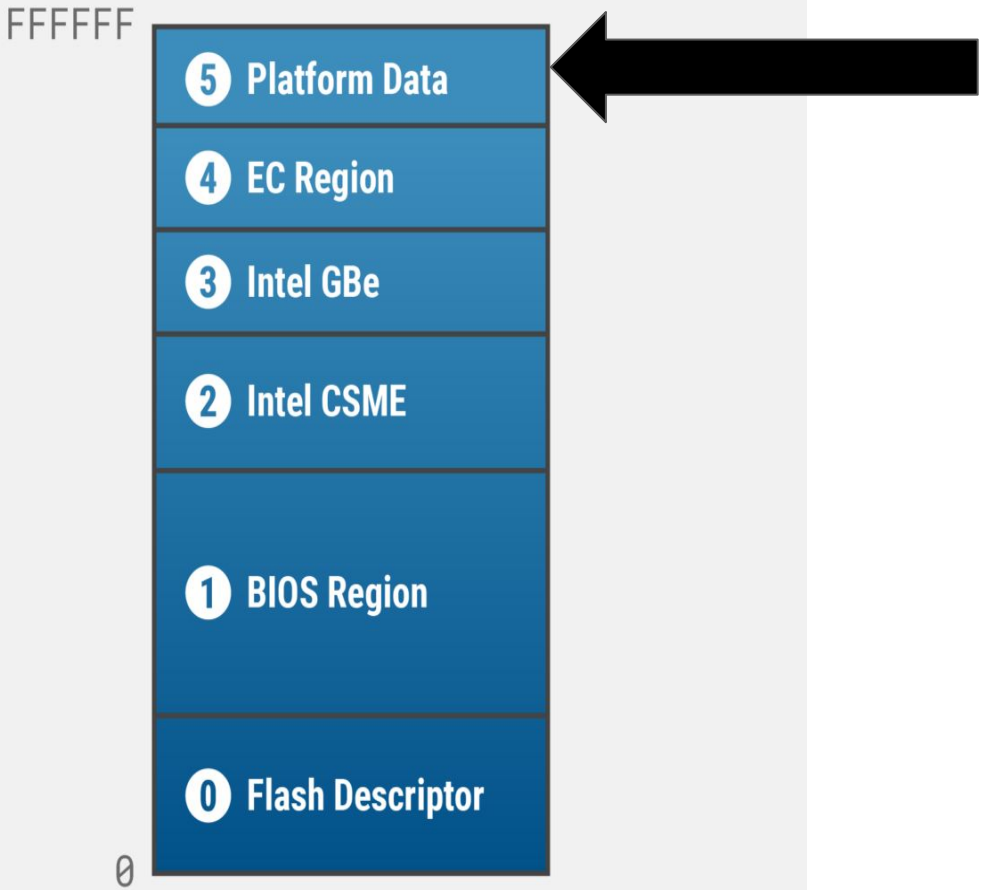
Stores EC-firmware (Embedded Controller) for things like your keyboard and sound/monitor controls.

I have not found a ton of security research on this region (If you are interested in more: https://www.youtube.com/watch?v=g-1Y466rDaI)

Curious if this will also impact AMT, as you could disrupt all ability to connect remotely to a machine.

Recovery would involve the OEM (or a backup if you have one).

Platform Data Region



Stores configuration for the “platform”, which could be anything else that is used by the OEM.

Implementation specific, and I was not able to find much information, let alone security research, about it.

Warning: Flashing the BIOS is a dangerous activity that may render your motherboard inoperable!

“Bad Things”

Using **Chipsec**, either built-in functionality or just reversing a read to a write (e.g. Erase what’s at the reset vector 4096 bytes, last 4096 reset) (https://github.com/chipsec/chipsec)

flashrom - Take an image and re-write it

RWEverything - (Read & Write Everything) - Also used by attackers!

Some systems have a recovery image, some store this on a separate SPI Flash

Warning: If you have a laptop/notebook/netbook, please do NOT try flashrom because interactions with the EC on these machines might crash your machine during flashing. flashrom tries to detect if a machine is a laptop, but not all laptops follow the standard, so this is not 100% reliable.[1]

https://wiki.archlinux.org/title/Flashing_BIOS_from_Linux

Why Can’t We Just Lock Everything Down?


IT LOOKS LIKE THERE WERE
TOO MANY CHEFS IN THE KITCHEN. YEEEAAAAAAHH.

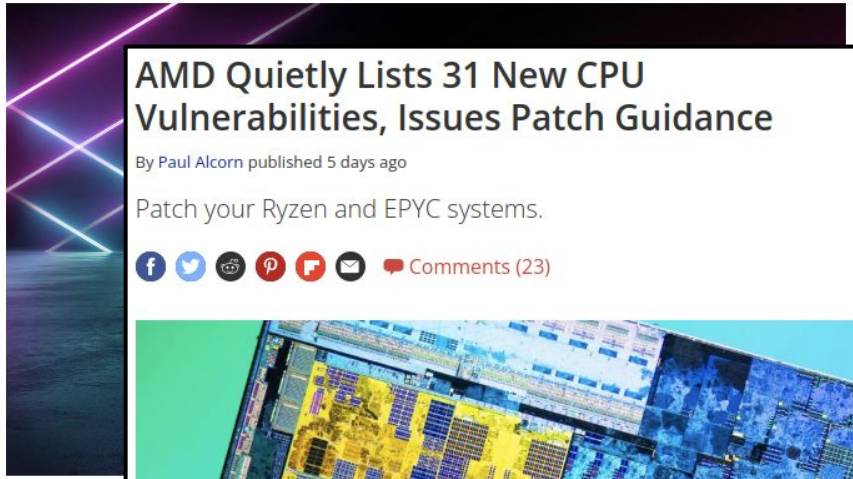

Microsoft
DELL
intel
phoenix
technologies
FOXCONN
msi

OEMs Make Mistakes…

MSI accidentally breaks Secure Boot for hundreds of motherboards

By Bill Toulas

January 16, 2023 05:02 PM 5



Over 290 MSI
settings that al
signature.

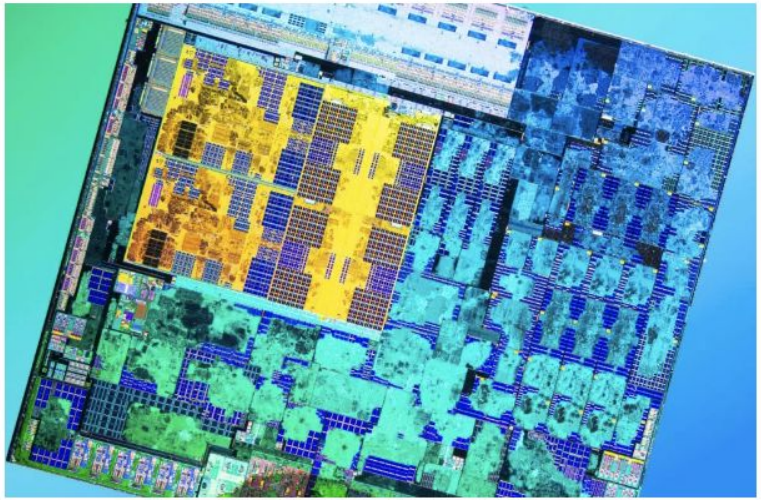
This discovery
receive a respo

AMD Quietly Lists 31 New CPU Vulnerabilities, Issues Patch Guidance

By Paul Alcorn published 5 days ago

Patch your Ryzen and EPYC systems.

Comments (23)



Intel confirms leaked Alder Lake BIOS Source Code is authentic

By Lawrence Abrams

October 9, 2022 08:53 PM 1

DR Tech | 3 MIN READ | DR TECHNOLOGY

Latest Firmware Flaws in Qualcomm Snapdragon Need Attention

The issue concerns the boot layer of ARM chips, which are driving a low-power mobile ecosystem that includes 5G smartphones and base stations.

Agam Shah
Contributing Writer

January 09, 2023

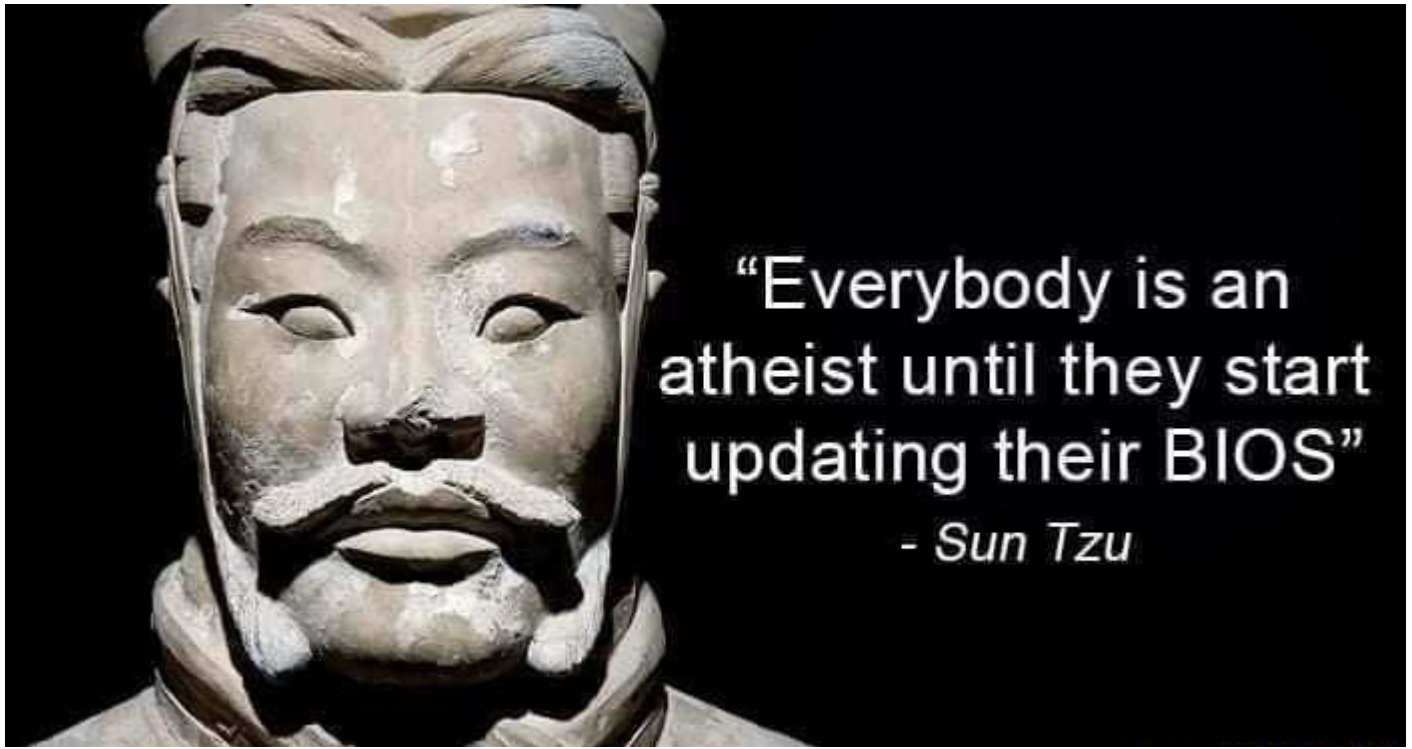
Meme unavailable due to supply chain issues. Thank you for understanding.

Finding Vulnerabilities and Potential Solutions

```
[redacted]:~$ fwupdmgr security --force
Host Security ID: HSI:0! (v1.7.9)

HSI-1
✔ CSME override:                Locked
✔ CSME v0:12.0.70.1652:         Valid
✔ Intel DCI debugger:           Disabled
✔ SPI write:                    Disabled
✔ UEFI platform key:            Valid
✘ CSME manufacturing mode:      Unlocked
✘ SPI BIOS region:              Unlocked
✘ SPI lock:                     Disabled
✘ TPM v2.0:                     Not found

HSI-2
✔ Intel BootGuard:              Enabled
✔ Intel DCI debugger:           Locked
✘ IOMMU:                        Not found
✘ Intel BootGuard ACM protected: Invalid
✘ Intel BootGuard OTP fuse:     Invalid
✘ Intel BootGuard verified boot: Invalid

HSI-3
✘ Intel BootGuard error policy: Invalid
✘ Intel CET Enabled:            Not supported
✘ Pre-boot DMA protection:      Invalid
✘ Suspend-to-idle:              Disabled
✘ Suspend-to-ram:               Enabled

HSI-4
✔ Intel SMAP:                   Enabled
✘ Encrypted RAM:                Not supported

Runtime Suffix -!
✔ fwupd plugins:                Untainted
✘ Linux kernel:                 Tainted
✘ Linux kernel lockdown:        Disabled
✘ Linux swap:                   Unencrypted
✘ UEFI secure boot:             Disabled
```

```
[redacted]:~$ fwupdmgr security --force
Host Security ID: HSI:1! (v1.7.9)

HSI-1
✔ CSME manufacturing mode:      Locked
✔ CSME override:                Locked
✔ CSME v0:12.0.81.1753:         Valid
✔ Intel DCI debugger:           Disabled
✔ SPI BIOS region:              Locked
✔ SPI lock:                     Enabled
✔ SPI write:                    Disabled
✔ TPM empty PCRs:               Valid
✔ TPM v2.0:                     Found
✔ UEFI platform key:            Valid

HSI-2
✔ Intel BootGuard:              Enabled
✔ Intel BootGuard ACM protected: Valid
✔ Intel BootGuard OTP fuse:     Valid
✔ Intel BootGuard verified boot: Valid
✔ Intel DCI debugger:           Locked
✔ TPM PCR0 reconstruction:      Valid
✘ IOMMU:                        Not found

HSI-3
✔ Intel BootGuard error policy: Valid
✔ Pre-boot DMA protection:      Enabled
✘ Intel CET Enabled:            Not supported
✘ Suspend-to-idle:              Disabled
✘ Suspend-to-ram:               Enabled

HSI-4
✔ Intel SMAP:                   Enabled
✘ Encrypted RAM:                Not supported

Runtime Suffix -!
✔ Linux kernel:                 Untainted
✔ Linux kernel lockdown:        Enabled
✔ fwupd plugins:                Untainted
✘ Linux swap:                   Unencrypted

This system has HSI runtime issues.
 » https://github.com/fwupd/fwupd/wiki/Host-security-ID-runtime-issues


Upload these anonymous results to the Linux Vendor Firmware Service to help other users? [y|N]:
```

Update Your Firmware



If you don’t, could someone “blow up” your computer?

PMFault: Faulting and Bricking Server CPUs through Management Interfaces, Or: A Modern Example of Halt and Catch Fire (January 13, 2023, https://arxiv.org/pdf/2301.05538.pdf)

“we show how remotely exploitable software weaknesses in the BMC (or other processors with PMBus access) can be used to access the PMBus and then perform hardware-based fault injection attacks”

*“...we experimentally show that overvolting outside the specified range has the potential of **permanently damaging Intel Xeon CPUs, rendering the server inoperable**.”*

The Future…

Upload your SPI flash image, backend analyzes it, you get a report

Scan the firmware update for “weird things” - should not break the platform, make it insecure, or brick the system (OEMs tend to care most about the latter)

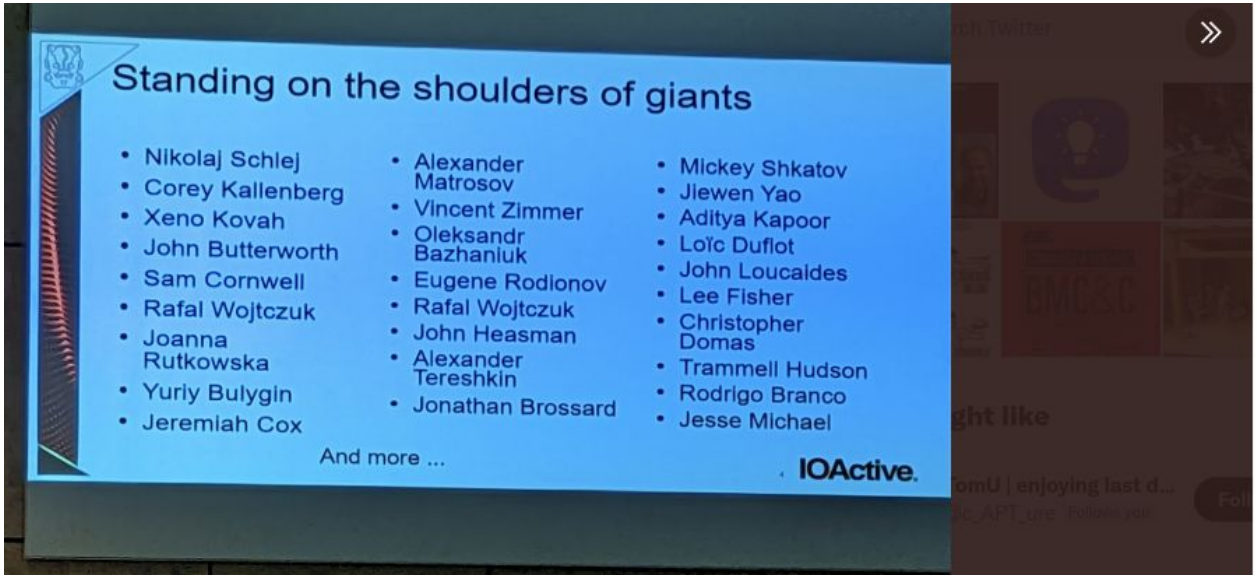
Others then get to evaluate some aspects of supply chain and firmware security pre-purchase

Some previous attempts, none still available today (to my knowledge)

Huge Thanks!

My Co-workers: Alex Bazhaniuk, Yuriy Bulygin, John Loucaides, Federico “Fede” Perez, Mickey Shkatov, Jesse Michael, Vladyslav Babkin, Nate Warfield and more!

About Me: Podcast host for Paul’s Security Weekly (https://securityweekly.com), Principal Security Evangelist for Eclypsium, and Eclypsium Podcast host (new!)



Content Leading Up To This Talk

Firmware Enumeration with Open Source Tools (Video/Webinar)

BHIS | Firmware Enumeration Using Open Source Tools | Paul Asadoorian | 1-Hour (Video/Webinar)

Firmware Security Realizations – Part 1 – Secure Boot And Dbx (Blog post)

Firmware Security Realizations – Part 2 – Start Your Management Engine (Blog Post)

Firmware Security Realizations – Part 3 – Spi Write Protections (Blog Post)

UEFI & SMM Vulnerabilities - Jesse Michael - PSW #764 (Video/Podcast)

Not-So-Secure Boot - Jesse Michael, Mickey Shkatov - PSW #751 (Video/Podcast)