Firmware Enumeration With Open Source Tools

Paul Asadoorian, Principal Security Evangelist

How many devices are in your devices?

1 

CPU

2 

ME/Chipset

3 

Network cards

4 

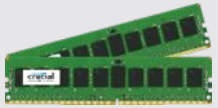
BMC

5 

BIOS / UEFI

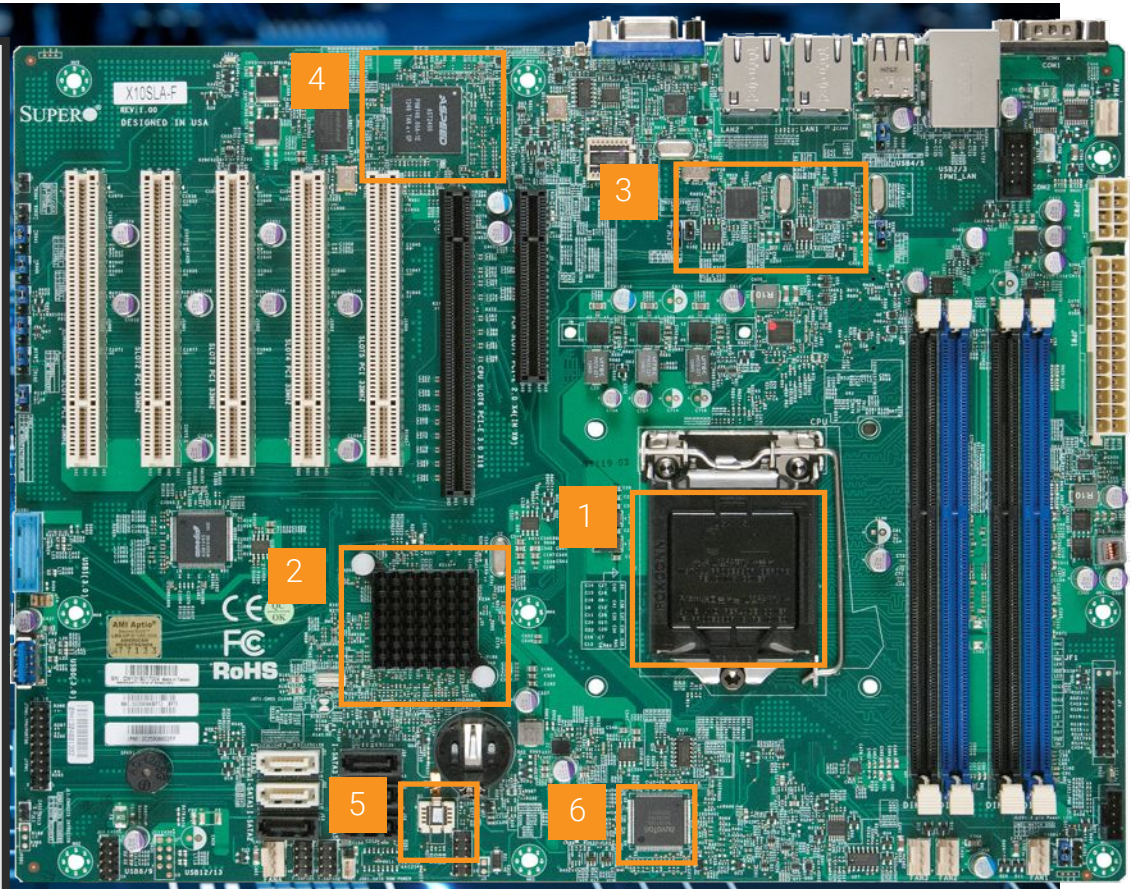
6 

Embedded controllers



RAM / Memory

SSD/HD



The hidden dangers inside the platform by Mickey Shkatov and Jesse Michael: https://www.youtube.com/watch?v=U37Z941pcME

What/Who Attacks UEFI Firmware?

- 2011 - Mebromi (BIOS)
- 2015 - Hacking Team - UEFI rootkit
- 2015-2117 - Equation Group and Vault 7 Leaks
- 2018 - LoJax, used by Fancy Bear
- 2020 - MosaicRegressor, origin unknown
- 2020 - Trickboot - Trickbot contains code to read, write, and erase firmware
- 2021 - FinSpy - A UEFI component belonging to the FinFisher surveillance toolset
- 2021 - ESpecter - A bootkit persisting in the EFI System Partition that can bypass Windows Driver Signature Enforcement
- 2022 - MoonBounce - Attributed to APT41, or an actor closely affiliated to the group, which researchers say is part of the Winnti Umbrella
- 2022 - Conti Group Found Actively Looking For Firmware Vulnerabilities
- 2022 - CosmicStrand - Malware that “hooks” UEFI at an early stage to infect all subsequent operations in the boot process
- *2022 - BlackLotus - Researchers observed a UEFI bootkit sold online called “BlackLotus”.

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

*Unconfirmed at this time

Check Your BIOS/UEFI Version Before You Go Any Further

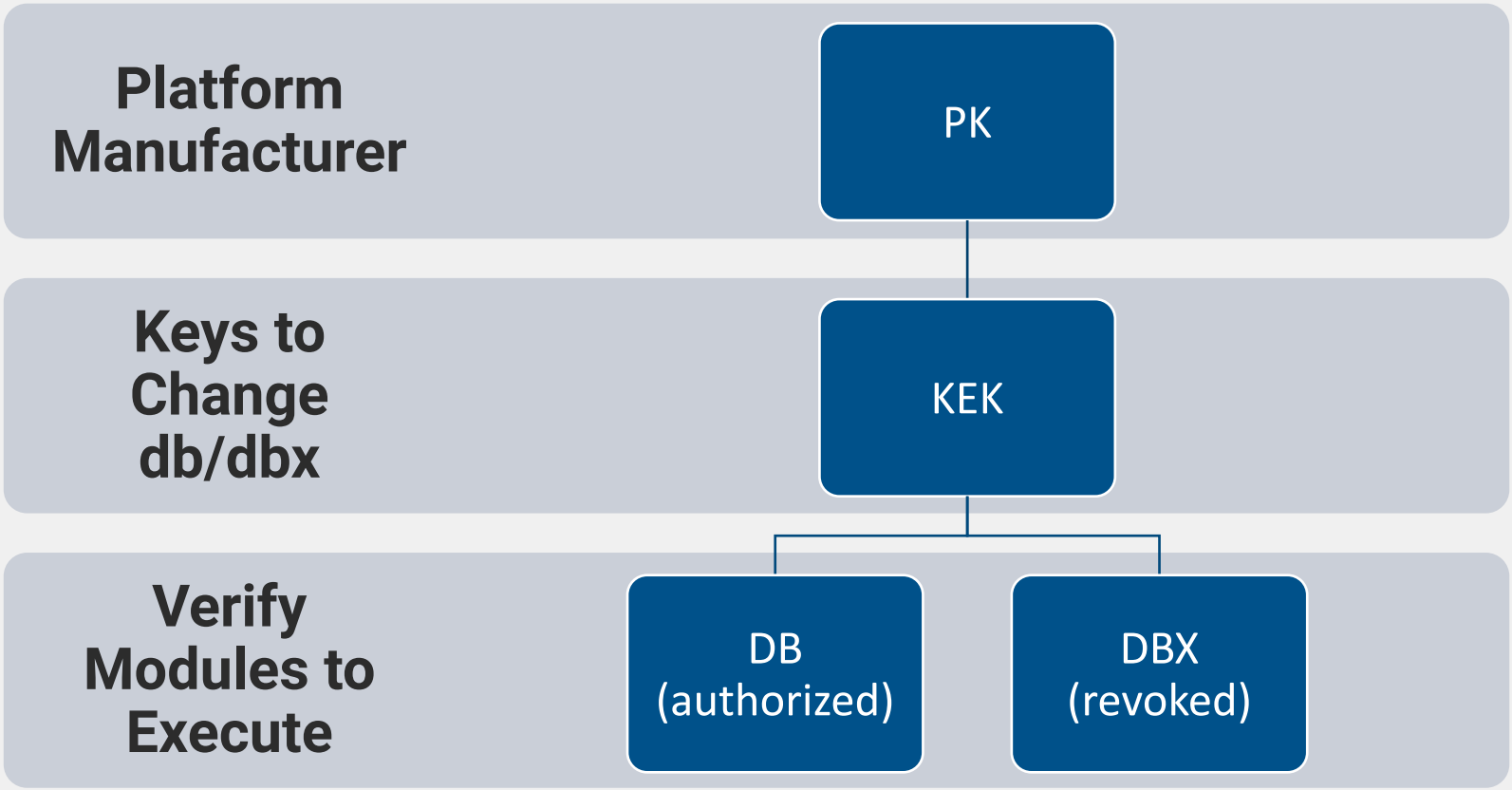
```
# For Linux systems
# sudo apt install dmidecode

$ sudo dmidecode -s bios-version
E16S3IMS.108

$ sudo dmidecode -s bios-release-date
11/18/2019
```

https://osxdaily.com/2022/02/02/find-mac-system-info-terminal-system-profiler/
https://windowsreport.com/check-bios-version-windows-10/
https://www.linuxtechi.com/dmidecode-command-examples-linux/

There is a potential workaround, that may help some people:

> This issue can be mitigated on some devices by updating the UEFI bios to the latest version before attempting to install KB5012170.

Secure Boot Certificate Hierarchy



Secure Boot Key Hierarchy

Platform Key (PK)

- Typically a single Platform Vendor Certificate
- It's primary job is to verify KEKs

Key Exchange Keys (KEKs)

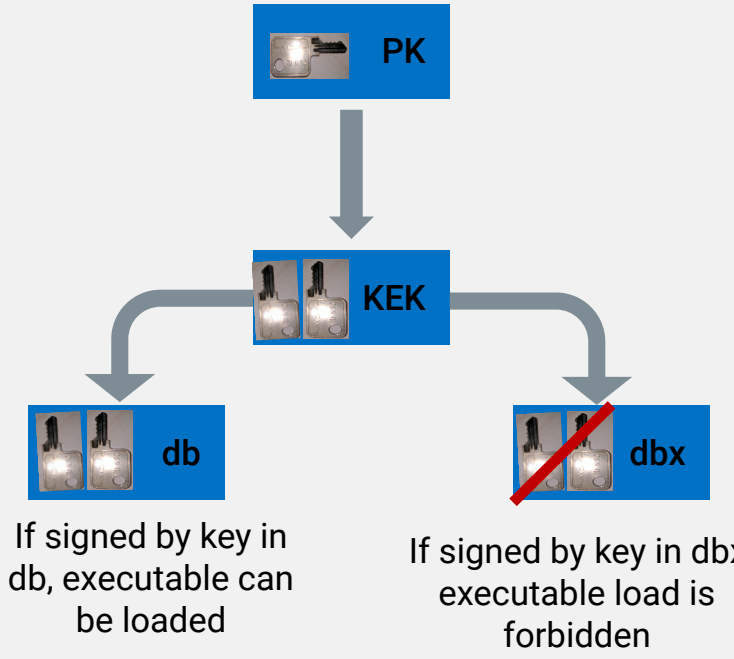
- The primary job is to verify db and dbx entries
- There can be multiple

Authorized Database (db)

- X509 certificates, SHA1/SHA256 hashes of allowed images

Forbidden Database (dbx)

- X509 certificates, SHA1/SHA256 hashes of revoked images (This gets checked first!)



Quick Check First: Am I Vulnerable To Boothole?

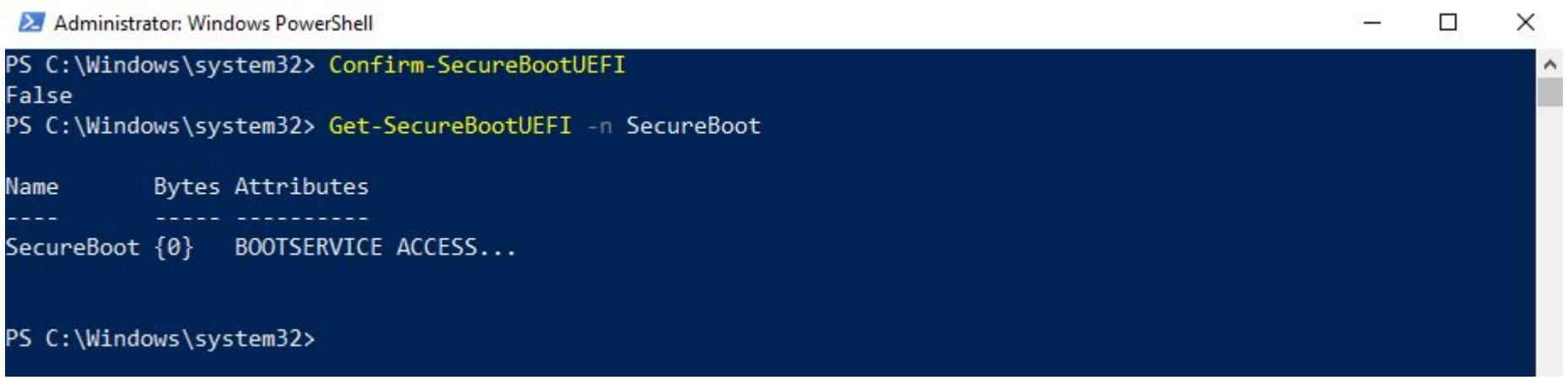
```
$ sudo bash BootHoleDetection.sh
Boot Hole Detection Bash Script
Copyright (C) 2020, Eclypsium, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it under certain conditions;

[*] mokutil:
SecureBoot disabled

[!] ESP is not protected


[+] No indication found that GRUB/Shim is vulnerable to BootHole
```

Latest research here: https://eclypsium.com/2022/08/11/vulnerable-bootloaders-2022/

Get it here: https://github.com/eclypsium/BootHole
There's A Hole In The Boot (Eclypsium Research, July 2020)

Is Secure Boot Enabled?

```
$ sudo fwts uefidump - | grep Secure
Name: SecureBoot
  Value: 0x00 (Secure Boot Mode Off)
Name: SecureBootSetup
  Data: 0340: 00 ff ff ff 83 07 53 65 63 75 72 65 42 6f 6f 74  ......SecureBoot


$ mokutil --sb-state
SecureBoot disabled
```



References:

https://support.apple.com/en-us/HT208198
https://www.linux.org/docs/man1/mokutil.html
https://wiki.ubuntu.com/FirmwareTestSuite/Reference
https://www.windowscentral.com/how-enable-secure-boot-pc-install-windows-11

What's In Those Variables?

```
$ efi-readvar -v PK -o old_PK.esl
$ efi-readvar -v KEK -o old_KEK.esl


$ sig-list-to-certs old_PK.esl oldpk
$ sig-list-to-certs old_KEK.esl oldkek


$ openssl x509 -in oldpk-0.der -inform der -noout -text
$ openssl x509 -in oldkek-0.der -inform der -noout -text
```

- Read the PK and KEK variables
- Convert them to DER format
- Use openssl to convert the DER formatted certificate to text.

What's In Those Variables?

```
$ openssl x509 -in oldkek-0.der -inform der -noout -text | grep Issuer
        Issuer: CN = ASUSTeK MotherBoard KEK Certificate

$ openssl x509 -in oldkek-1.der -inform der -noout -text | grep Issuer
        Issuer: C = US, ST = Washington, L = Redmond, O = Microsoft Corporation, CN =  Microsoft
Corporation Third Party Marketplace Root
                CA Issuers - URI:http://www.microsoft.com/pki/certs/MicCorThiParMarRoo_2010-10-05.crt

$ openssl x509 -in oldkek-2.der -inform der -noout -text | grep Issuer
        Issuer: C = GB, ST = Isle of Man, L = Douglas, O = Canonical Ltd., CN =  Canonical Ltd. Master
Certificate Authority
```

How Do You Know If Your DBX Is Out-Of-Date?

Updating The DBX Manually (Use caution!)

```
$ sudo apt install efitools

$ efi-readvar -v dbx -o existing_dbx.esl
Variable dbx, length 3724

$ dbxtool -d existing_dbx.esl -l | tail -1
240: {microsoft} {sha256} 540801dd345dc1c33ef431b35bf4c0e68bd319b577b9abe1a9cff1cbc39f548f

$ wget https://uefi.org/sites/default/files/resources/dbxupdate_x64.bin

$ sudo cp dbxupdate_x64.bin /usr/share/secureboot/updates/dbx/dbxupdate_x64.bin

$ sudo /usr/bin/sbkeysync -no-default-keystores -keystore /usr/share/secureboot/updates -verbose

$ efi-readvar -v dbx -o updated_dbx.esl

$ dbxtool -d updated_dbx.esl -l | tail -1
271: {microsoft} {sha256} af79b14064601bc0987d4747af1e914a228c05d622ceda03b7a4f67014fee767
```

NOTE: This is the old method, LVFS now supports a better way!

Firmware Security Realizations - Part 1 - Secure Boot and DBX

Using LVFS To Check For Firmware Updates

https://fwupd.org/

- The Linux Vendor Firmware Service (LVFS) provides Linux users with a facility to:
  - Discover devices/components on your system that have firmware
  - Download updates if new firmware is available
  - Apply firmware updates
- Vendors must submit firmware updates to LVFS

```
# fwupdmgr get-updates
Devices with no available firmware updates:
 • System Firmware
 • Thunderbolt host controller
 • UEFI dbx
 • WDC PC SN730 SDBPNTY-1T00-1032
No updatable devices
```

```
$ fwupdmgr get-devices
System Product Name
│
└─┐
  └─UEFI dbx:
        Device ID:          362301da643102b9f38477387e2193e57abaa590
        Summary:            UEFI revocation database
        Current version:    77
        Minimum Version:    77
        Vendor:             UEFI:Linux Foundation
        Install Duration:   1 second
        GUIDs:              fda6234b-adcb-5105-8515-9af647d29775
                            f8ff0d50-c757-5dc3-951a-39d86e16f419
                            c6682ade-b5ec-57c4-b687-676351208742
                            f8ba2887-9411-5c36-9cee-88995bb39731
                            7d5759e5-9aa0-5f0c-abd6-7439bb11b9f6
                            0c7691e1-b6f2-5d71-bc9c-aabee364c916
        Device Flags:       • Internal device
                            • Updatable
                            • Needs a reboot after installation
                            • Only version upgrades are allowed
```

Could we detect an outdated DBX and then use LVFS to update DBX entries? As of 1.8.5, YES!

https://blogs.gnome.org/hughsie/2020/08/17/updating-secure-boot-dbx-with-fwupd-and-the-lvfs/

Version 217:

Warning: This firmware is in the testing state and may not be suitable for production systems.

Released	2022-08-23 10:12:10
State	testing
Urgency	high
License	Proprietary, distributed by agreement
Filename	DBXUpdate-20220812-x64.cab
Description	This updates the dbx to the latest release from Microsoft which adds insecure versions of grub and shim to the list of forbidden signatures due to multiple discovered security updates.

https://fwupd.org/lvfs/devices/org.linuxfoundation.dbx.x64.firmware

Use LVFS To Update The DBX

```
$ fwupdmgr --version

fwupdmgr --version
runtime   org.freedesktop.fwupd       1.8.5
runtime   org.freedesktop.fwupd-efi   1.3
compile   org.freedesktop.gusb        0.4.0
runtime   com.dell.libsmbios          2.4
runtime   org.kernel                  5.15.65-1-MANJARO
runtime   com.hughsie.libjcat         0.1.11
compile   com.hughsie.libjcat         0.1.11
compile   org.freedesktop.fwupd       1.8.5
runtime   org.freedesktop.gusb        0.4.0

# NOTE: You have to run the following command before any updates show up in the UI:

$ fwupdmgr get-updates

Firmware metadata has not been updated for 30 days and may not be up to date.

Update now? (Requires internet connection) [y|N]: y
Updating lvfs
Downloading…             [****************************************]
Downloading…             [****************************************]
Downloading…             [****************************************]
Successfully downloaded new metadata: 1 local device supported
Devices with no available firmware updates:
 • System Firmware
 • Thunderbolt host controller
 • WDC PC SN730 SDBPNTY-1T00-1032
```

Use LVFS To Update The DBX

I've done this on my 3 primary systems and found that it works great!

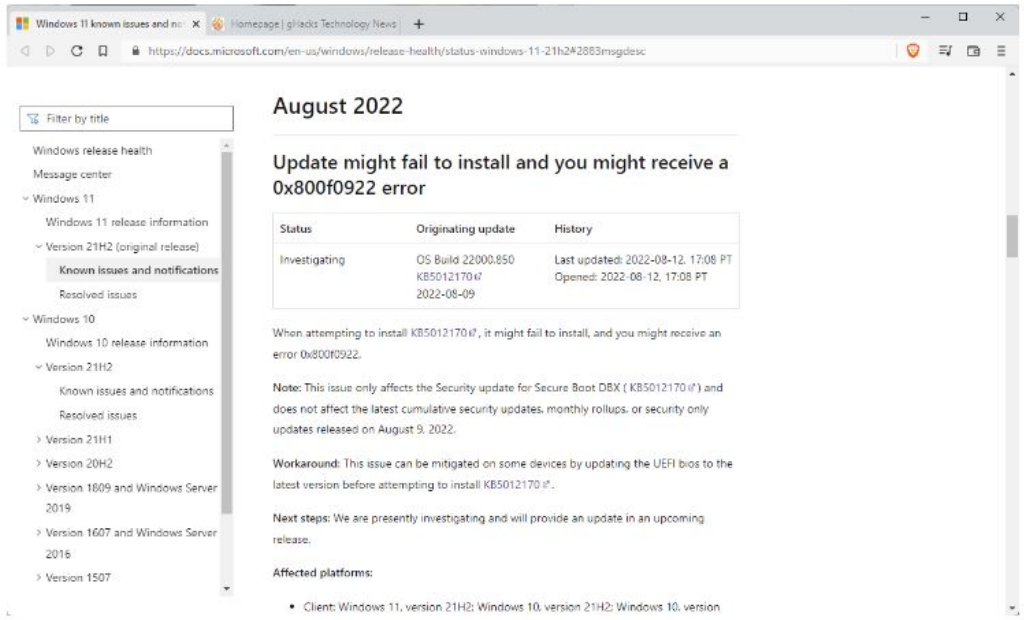
```
[paulda@gibson ~]$ fwupdmgr update
Devices with no available firmware updates:
 • System Firmware
 • Thunderbolt host controller
 • WDC PC SN730 SDBPNTY-1T00-1032

 Upgrade UEFI dbx from 211 to 217?

 This updates the dbx to the latest release from Microsoft which adds
 insecure versions of grub and shim to the list of forbidden signatures due
 to multiple discovered security updates.

 Before installing the update, fwupd will check for any affected executables
 in the ESP and will refuse to update if it finds any boot binaries signed
 with any of the forbidden signatures. If the installation fails, you will
 need to update shim and grub packages before the update can be deployed.

 Once you have installed this dbx update, any DVD or USB installer images
 signed with the old signatures may not work correctly. You may have to
 temporarily turn off secure boot when using recovery or installation media,
 if new images have not been made available by your distribution.


Perform operation? [Y|n]: Y
Downloading…             [****************************************]
Decompressing…           [****************************************]
Authenticating…          [****************************************]
==== AUTHENTICATING FOR org.freedesktop.fwupd.update-internal-trusted ====
Authentication is required to update the firmware on this machine
Authenticating as: Paul Asadoorian (paulda)
Password:
==== AUTHENTICATION COMPLETE ====
Waiting…                 [****************************************]
Writing…                 [****************************************]
Waiting…                 [****************************************]
Waiting…                 [****************************************]
Successfully installed firmware

An update requires a reboot to complete. Restart now? [y|N]:
```

Advice: Update Firmware…

KB5012170: Windows update error 0x800f0922, UEFI Bios update may resolve it

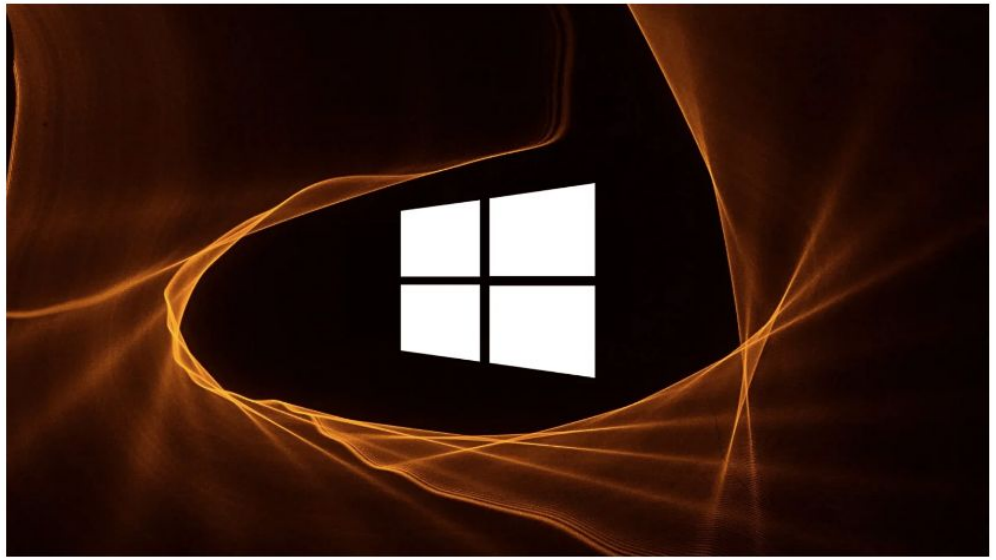
MARTIN BRINKMANN Aug 15, 2022 **Windows Updates** | 8

Microsoft released cumulative updates for all supported versions of Windows on the August 2022 Patch Day. The company did release a second security update for Windows at the day to address issues in Secure Boot DBX.



Windows KB5012170 Secure Boot DBX update may fail with 0x800f0922 error

By Ionut Ilascu August 15, 2022 11:41 AM 3

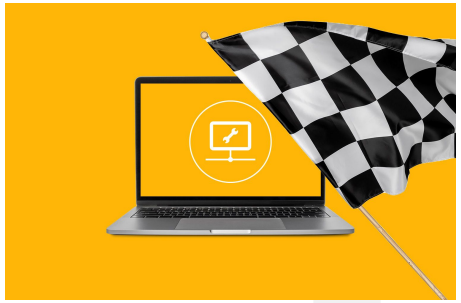


Intel ME/CSME

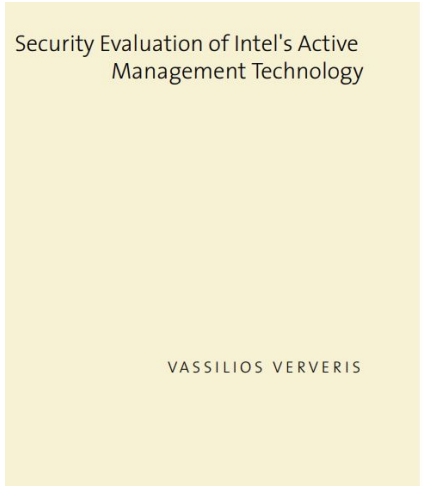
Intel CSME (Converged Security and Manageability Engine) Quick Rundown

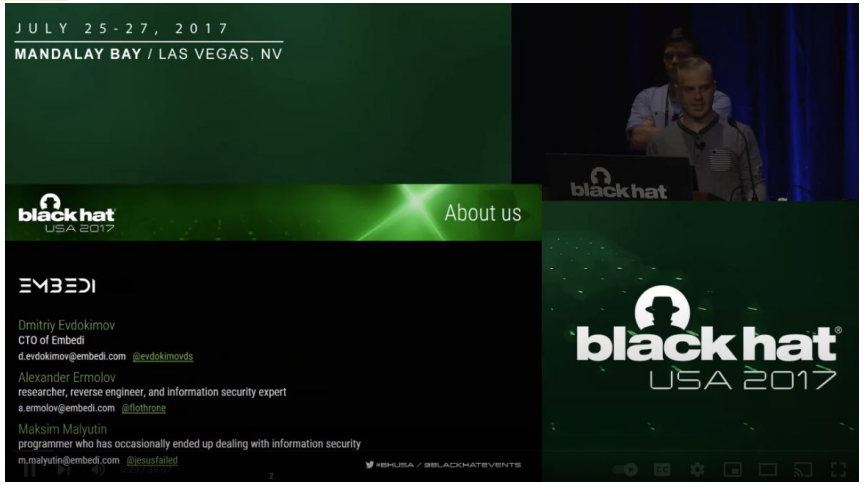
- Introduced by Intel in 2006 (if you have an Intel-based computer, you likely have it in some capacity)
- Runs on a separate CPU and SRAM with its own micro-kernel (uses SPI flash for storage)
- Introduces “Ring -3” level access to your computer, direct access to all components as conceivably the lowest level, but highest privilege level
- In most cases it cannot be completely removed or disabled, some functionality is required for your computer to boot and operate correctly
- Some manufacturers will reduce its functionality (e.g. Apple, System76)
- AMT (Active Management Technology) is a feature set that can be included in ME

[Firmware Security Realizations – Part 2 – Start Your Management Engine]



Security Researchers Have Found A Few Issues…





Links to all the research I could find on this topic is referenced here: [Firmware Security Realizations – Part 2 – Start Your Management Engine]

Check For “Silent Bob Is Silent”

```
$ git clone https://github.com/intel/INTEL-SA-00075-Linux-Detection-And-Mitigation-Tools
$ cd INTEL-SA-00075-Linux-Detection-And-Mitigation-Tools/
$ make
$ sudo ./INTEL-SA-00075-Discovery-Tool -d /dev/mei0

INTEL-SA-00075-Discovery-Tool – Release 1.0
Copyright (C) 2003-2012, 2017 Intel Corporation. All rights reserved


———————Vulnerability Status———————
System is not Vulnerable, no further action needed.

————————————————————
```

(Authentication bypass in Intel AMT discovered in 2017)

Intel Manufacturing Mode

- Typically enabled during the manufacturing and/or assembly phases on the PC
- The OEM uses this mode to set keys and other settings, then lock them by turning off manufacturing mode
- If its enabled, an attacker can set keys or other settings, **and then lock them!**
- Intel-based Apple laptops had this problem for a period of time: https://www.zdnet.com/article/some-apple-laptops-shipped-with-intel-chips-in-manufacturing-mode/

"Manufacturing Mode allows for the configuration of critical platform settings, such as those for BootGuard, a technology available with Intel's chips that can verify the boot process. These settings are stored in one-time-programmable memory (FUSEs), and some of them are called Field Programmable Fuses (FPFs) FPFs are typically used to store platform parameters. Setting FPFs requires Intel's ME to be in the Manufacturing Mode"

https://www.tomshardware.com/news/intel-me-cpu-undocumented-manufacturing-mode,37883.html

Matthew Garrett's AMT Check

```
$ git clone https://github.com/mjg59/mei-amt-check
$ cd mei-amt-check/
$ make
$ sudo ./mei-amt-check

Error: Management Engine refused connection. This probably means you don't
have AMT
```

As AMT is a feature, and more specifically a set of modules that can be included in ME's firmware, its nice to be able to check for it. Typically an Intel PC labeled with vPro will support AMT.

Chipsec Can Be Used To Enumerate ME's Manufacturing Mode

```
## Download and install the latest version of Chipsec
## Note: The instructions that follow were tested on Ubuntu 20.04.4

$ sudo apt install python3 python3-setuptools nasm build-essential linux-headers-$(uname -r)

$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python3 1

$ git clone https://github.com/chipsec/chipsec.git

$ cd chipsec

$ python setup.py build_ext -i
```

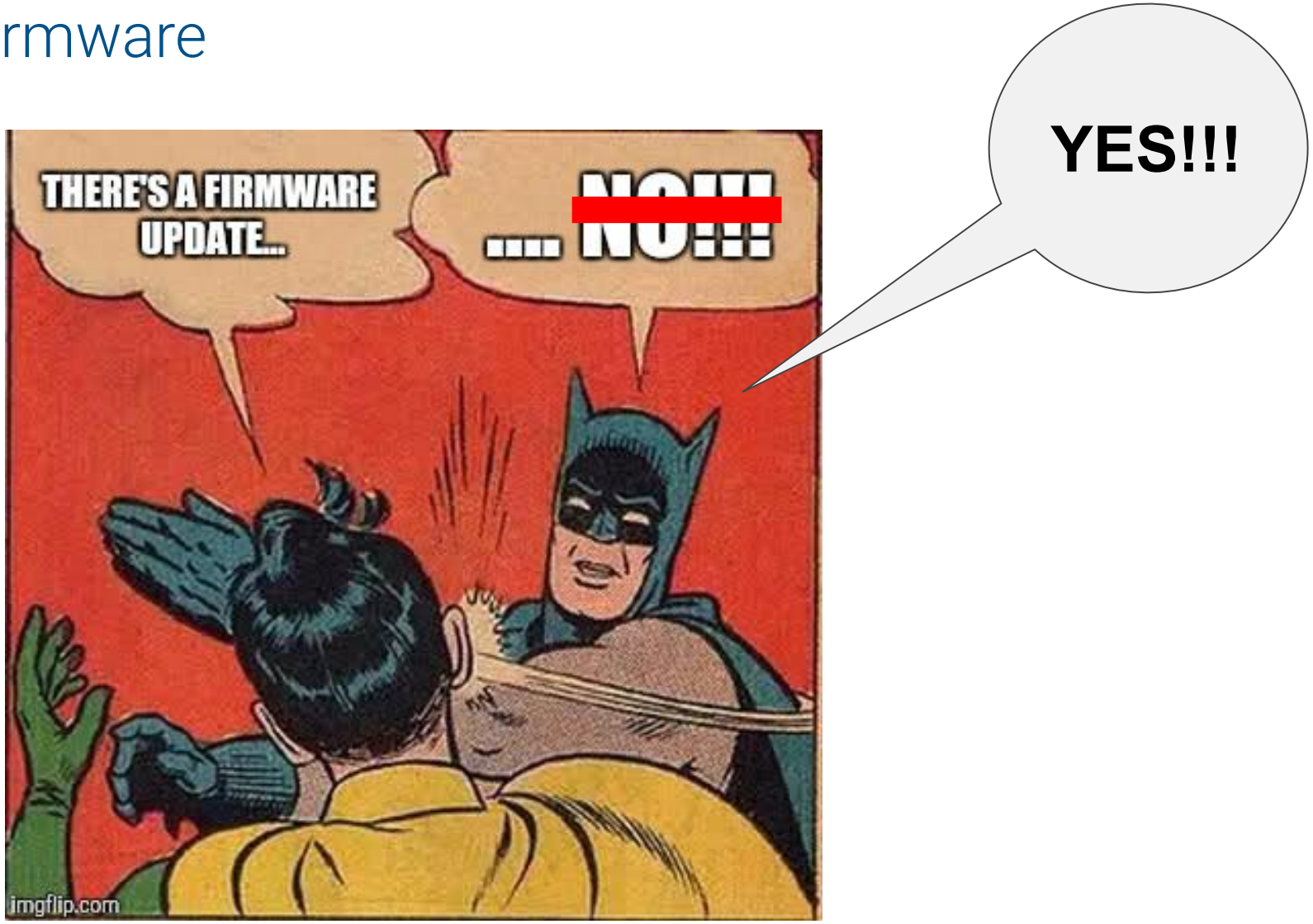
The Phantom Menace: Intel ME Manufacturing Mode (HiTHB 2018)

```
$ sudo python chipsec_main.py -m common.me_mfg_mode
################################################################
##                                                            ##
##  CHIPSEC: Platform Hardware Security Assessment Framework  ##
##                                                            ##
################################################################
[CHIPSEC] Version 1.8.6
[CHIPSEC] Arguments: -m common.me_mfg_mode

****** Chipsec Linux Kernel module is licensed under GPL 2.0
[CHIPSEC] API mode: using CHIPSEC kernel module API
[CHIPSEC] OS      : Linux 5.15.0-41-generic #44~20.04.1-Ubuntu SMP Fri Jun 24 13:27:29 UTC 2022 x86_64
[CHIPSEC] Python  : 3.8.10 (64-bit)
[CHIPSEC] Helper  : LinuxHelper (/home/paulda/chipsec/chipsec/helper/linux/chipsec.ko)
[CHIPSEC] Platform: Desktop 8th Generation Core Processor (CoffeeLake S 8 Cores)
[CHIPSEC]      VID: 8086
[CHIPSEC]      DID: 3E30
[CHIPSEC]      RID: 0D
[CHIPSEC] PCH     : Intel Z390 (300 series) PCH
[CHIPSEC]      VID: 8086
[CHIPSEC]      DID: A305
[CHIPSEC]      RID: 10

[+] loaded chipsec.modules.common.me_mfg_mode
[*] running loaded modules ..

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

```
## Note: You make have to install the required dependencies:
## sudo apt install libpci-dev zlib1g-dev

$ git clone -depth=1 https://review.coreboot.org/coreboot
$ cd coreboot/util/intelmetool
$ make
$ sudo ./intelmetool -m

MEI found: [8086:a360] Cannon Lake PCH HECI Controller

ME Status   : 0x90000255
ME Status 2 : 0x6f10506

ME: FW Partition Table      : OK
ME: Bringup Loader Failure  : NO
ME: Firmware Init Complete  : YES
ME: Manufacturing Mode      : YES
ME: Boot Options Present    : NO
ME: Update In Progress      : NO
ME: Current Working State   : Normal
ME: Current Operation State : M0 with UMA
ME: Current Operation Mode  : Normal
ME: Error Code              : No Error
ME: Progress Phase          : ROM Phase
ME: Power Management Event  : Pseudo-global reset
ME: Progress Phase State    : (null)

ME: Extend Register not valid

ME: Firmware Version 12.0.1652.70 (code) 12.0.1652.70 (recovery) 12.0.1652.70 (fitc)
```

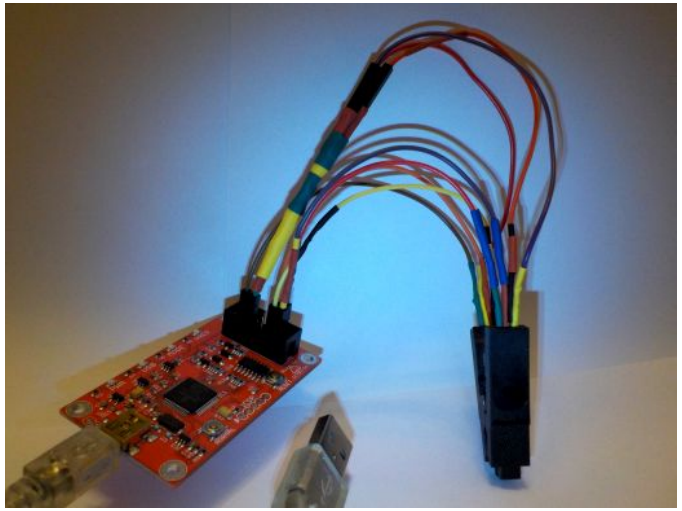
```
$  wget https://downloadmirror.intel.com/28632/CSME_Version_Detection_Tool_Linux.tar.gz
$  mkdir intel_csme
$  cd intel_csme/
$  tar zxvf ../CSME_Version_Detection_Tool_Linux.tar.gz
$  sudo python3 ./intel_csme_version_detection_tool

Intel(R) CSME Version Detection Tool
Copyright(C) 2017-2022, Intel Corporation, All rights reserved.

Application Version: 7.0.2.0
Scan date: 2022-07-14 17:03:03 GMT

*** Host Computer Information ***
Name: gibson
Manufacturer: Micro-Star International Co., Ltd.
Model: Prestige 15 A10SC
Processor Name: Intel(R) Core(TM) i7-10710U CPU @ 1.10GHz
OS Version: Ubuntu 20.04.4 LTS (5.15.0-41-generic)

*** Intel(R) ME Information ***
Engine: Intel(R) Converged Security and Management Engine
Version: 14.0.0.1061

*** Risk Assessment ***
Based on the analysis performed by this tool: This system is vulnerable.
Explanation:
  The detected version of the Intel(R) Converged Security and Management Engine firmware
  has a vulnerability listed in one or more of the public Security Advisories.
  Contact your system manufacturer for support and remediation of this system.
```

Advice: Update Firmware



Protecting System Firmware Storage

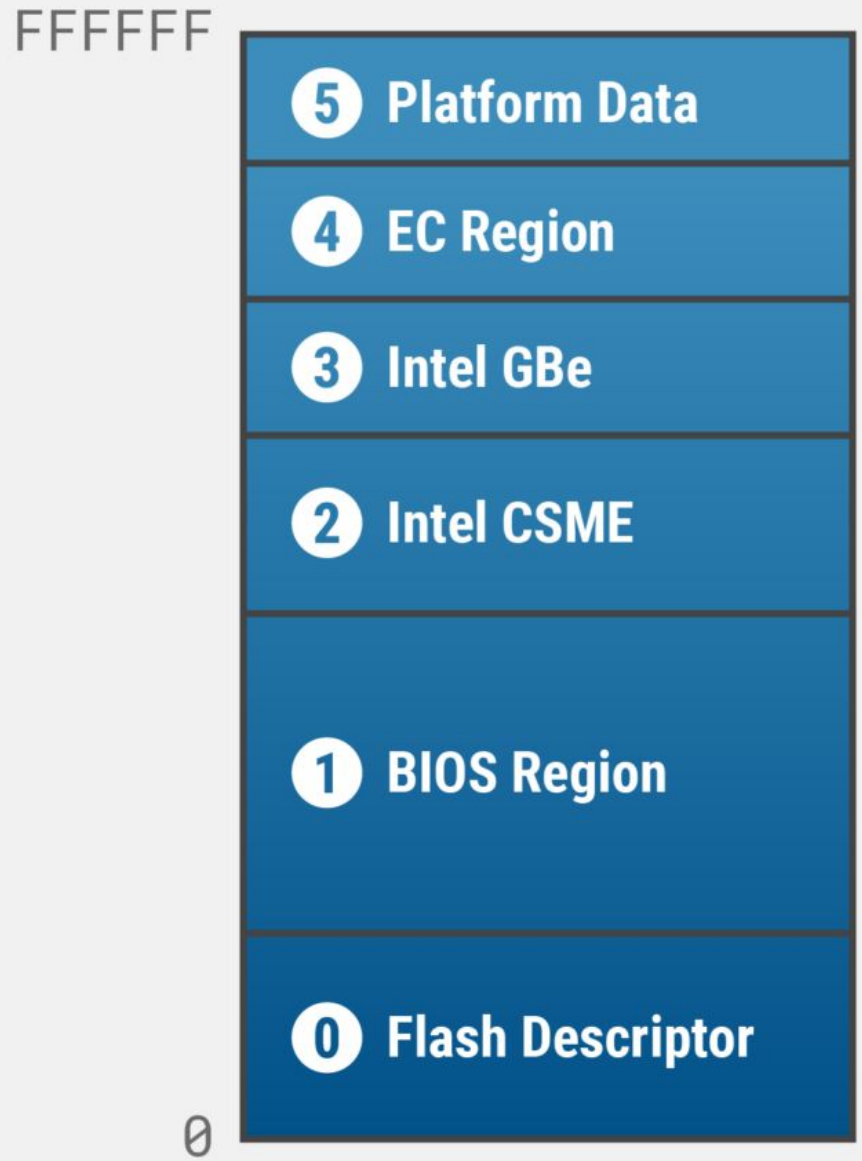
- Protecting the contents of the SPI flash is tricky business as there are multiple mechanisms to control writes:
  - The OEM can require that the system be in SMM before allowing SPI flash writes.
  - **All of the settings and registers could (and do) change depending on which chipset in in your computer**
  - BIOS Control Register “SMM BIOS Write Protection” (SMM_BWP) when set, writes can only be allowed by code executing in SMM
  - The Flash Descriptor region defines how the SPI flash is laid out and some access controls
  - If you allow writes to the Flash Descriptor region then an attacker can essentially change the rules!

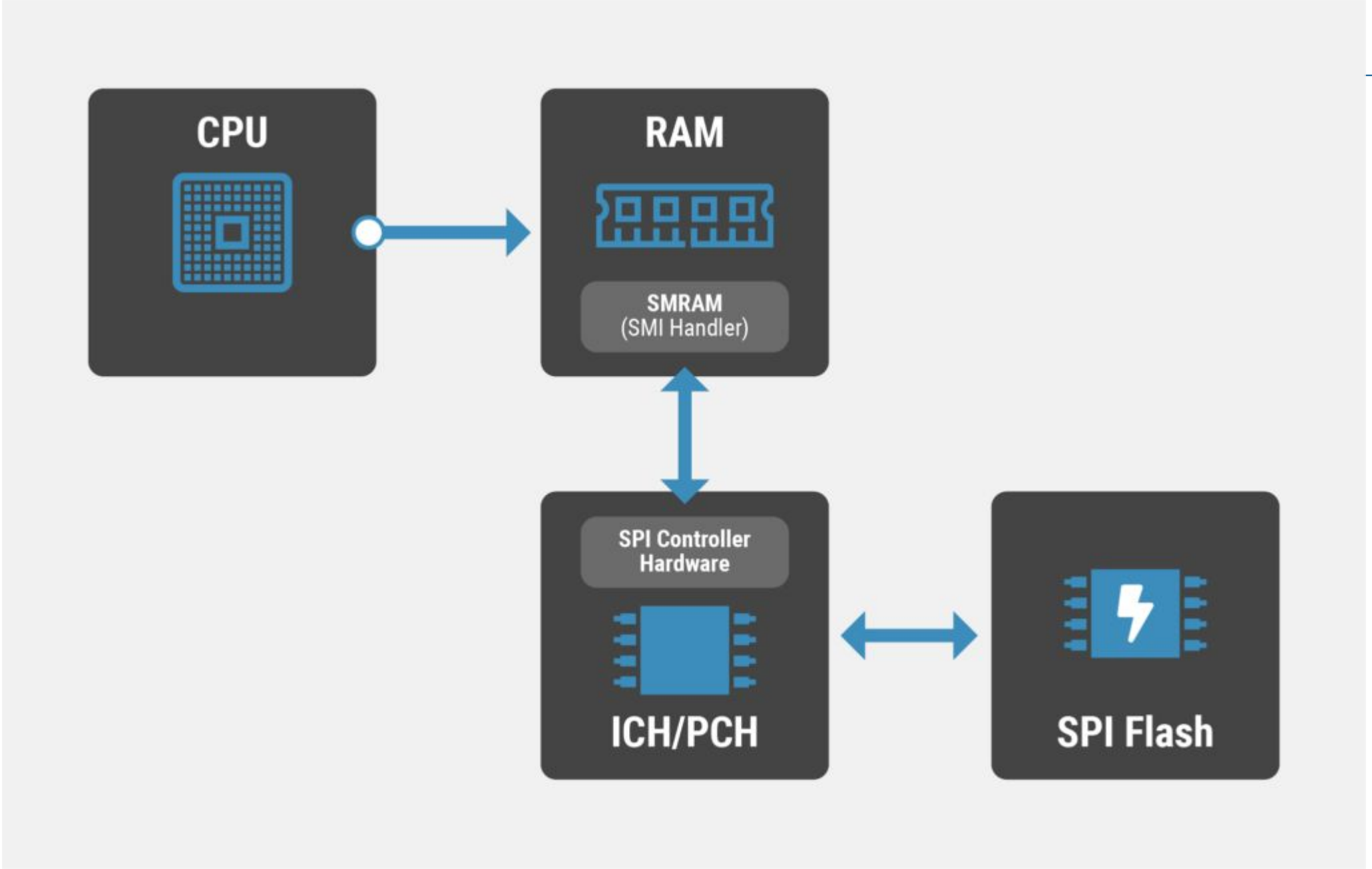


http://dangerousprototypes.com/docs/Flashing_a_BIOS_chip_with_Bus_Pirate

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

https://eclypsium.com/2019/10/23/protecting-system-firmware-storage/





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

SPI Flash Protections

There are four (in general) 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

#1 - The Flash Descriptor

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

################################################################
##                                                            ##
##  CHIPSEC: Platform Hardware Security Assessment Framework  ##
##                                                            ##
################################################################
[CHIPSEC] Version : 1.8.6
[CHIPSEC] OS      : Linux 5.15.0-41-generic #44~20.04.1-Ubuntu SMP Fri Jun 24 13:27:29 UTC 2022 x86_64
[CHIPSEC] Python  : 3.8.10 (64-bit)

****** Chipsec Linux Kernel module is licensed under GPL 2.0
[CHIPSEC] API mode: using CHIPSEC kernel module API
[CHIPSEC] Helper  : LinuxHelper (/home/paulda/chipsec/chipsec/helper/linux/chipsec.ko)
[CHIPSEC] Platform: Desktop 8th Generation Core Processor (CoffeeLake S 8 Cores)
[CHIPSEC]      VID: 8086
[CHIPSEC]      DID: 3E30
[CHIPSEC]      RID: 0D
[CHIPSEC] PCH     : Intel Z390 (300 series) PCH
[CHIPSEC]      VID: 8086
[CHIPSEC]      DID: A305
[CHIPSEC]      RID: 10
[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 = 0x00FFFFFF
[CHIPSEC] Dumping 0x01000000 bytes (to the end of BIOS region)
[spi] reading 0x1000000 bytes from SPI at FLA = 0x0 (in 262144 0x40-byte chunks + 0x0-byte remainder)
[CHIPSEC] Completed SPI flash dump to 'fd.bin'
[CHIPSEC] (spi) time elapsed 34.729
```

#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 = 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
```

#2 - Global Write Protections

#3 - BIOS 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

Advice: Update Firmware

Questions?

paul.asadoorian@eclypsium.com