DISCOVERING VULNERABLE UEFI FIRMWARE AT SCALE

Yuriy Bulygin, Alex Bazhaniuk

Presented: September 14, 2017. Last Update: October 4, 2017

Motivation

- We tend to focus on new vulnerabilities in firmware of new systems
- Yet there are still many systems that don’t have basic firmware security “hygiene”
- And lots of old systems which are in use for years
- Tools like CHIPSEC can help with checking these problems on individual systems
- But can we understand the state of entire population of systems?

INTRO TO BASIC UEFI FIRMWARE SECURITY

System Flash

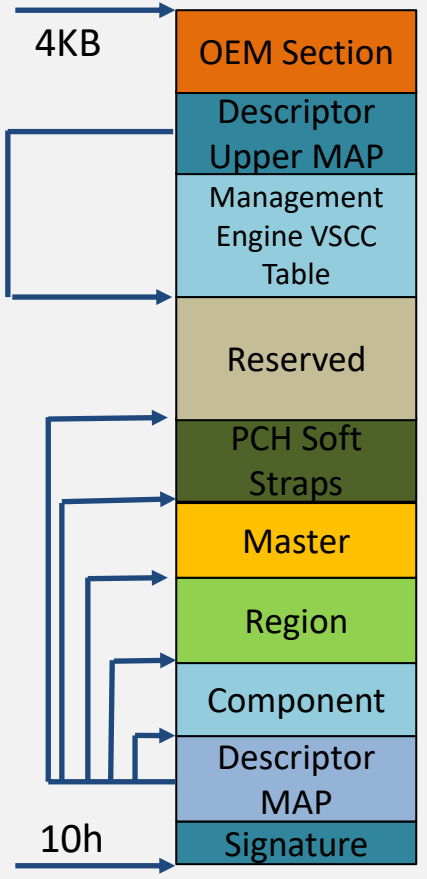
- SPI Flash Memory device(s) containing main UEFI firmware, Intel ME firmware, GBe persistent settings, EC firmware etc.
- Direct Access by software through physical address space
  - `0xFFFFFFFF` PA maps to `0xFFFFFF` Flash Linear Address
- Program Register Access by software via SPI MMIO registers
  - FLA are programmed explicitly
- Descriptor describes other regions

Region	Content
0	Flash Descriptor
1	BIOS
2	Intel Management Engine
3	Gigabit Ethernet
4	Platform Data

Intel 7 Series Chipset PCH datasheet

System Flash Descriptor

- Region 0 at FLA `0 – FFFh` (4 KB)
- Signature: **`0FF0A55Ah`** at 10h LBA
- Contains the following sections:
  - *Component*: flash device configuration
  - *Region*: describes other regions
  - *Master*: defines Rd/Wr Access Control table
- Access Control table defines which masters (CPU, ME, GbE) can access regions



Security of System Flash

1. Firmware update image must be signed
   - UEFI uses “capsules” for signed UEFI updates upon reboot or sleep
   - Capsule contains firmware volumes with firmware to be updated
   - Contains firmware executable that performs update
   - Boot-time firmware checks capsule signature before flashing
2. System flash must be read-only at run-time
   - Enforces secure update as SW cannot flash FW at run-time w/o signature checks
   - Some systems check signature & flash at run-time in SMM
3. Flash descriptor must be read-only
   - Programmed at manufacturing then never updated

Breaking a Whack-a-Mole

- Usual way to check if system firmware is protected is to run CHIPSEC
- But it requires testing on real hardware
- Sure enough, many platforms (even newest) are found to be vulnerable
  - Skylake based MSI
  - Gigabyte BRIX BIOS Write Protection is not enabled (CLVA-2017-01-002)
  - Coreboot
- Can we scale this analysis, at least for basic firmware protections, without testing every single system?

We have tons of UEFI update images from platform vendors. Let’s put them to work!

Can we find out which systems don’t protect their system flash based on just the update images?

On the shoulders of giants

- Amazing work by **Teddy Reed**
  - Analytics, and Scalability, and UEFI exploitation! (Infiltrate 2014)
  - UEFI Spider can crawl/download BIOS updates from OEM web-sites
- Tools that can parse UEFI firmware images or “capsules”
  - CHIPSEC
  - uefi-firmware-parser by Teddy Reed
  - UEFITool by Nikolaj Schlej

FINDING VULNERABLE PLATFORMS FROM UEFI UPDATES

What If?

Update image is not “signed capsule” & contains valid descriptor

Update is a full ROM image

Easily automatable

Suspected unsigned firmware update

Actually used as the update image

Not easily automatable

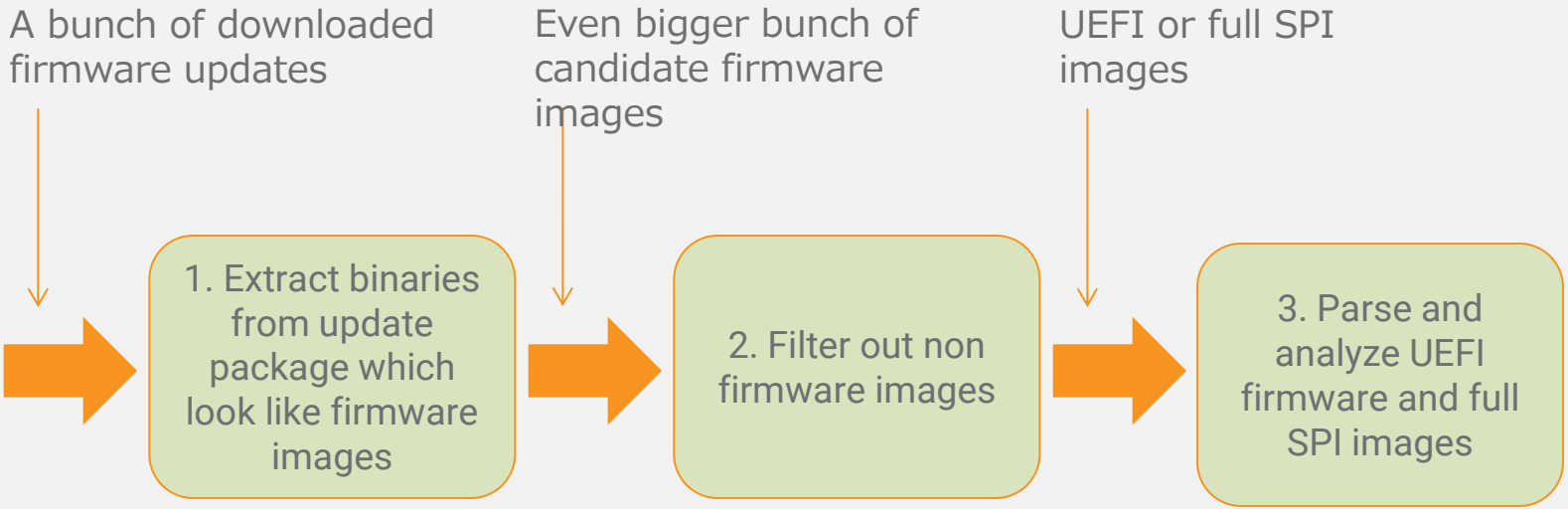
System flash is not protected

What this means exactly…

Every such ROM image would indicate that corresponding platform model [probably] has the following vulnerabilities

- UEFI firmware update are not signed
- System flash is writeable by software
- SPI flash descriptor is writeable by software

High-Level Process



Let’s get started…

- **32987** firmware updates packages from **9 platform vendors**
  - **Acer** 647, **ASRock** 306, **ASUS** 6871, **Dell** 9400, **Gigabyte** 2606, **HP** 3138, **Intel** 4408, **Lenovo** 2952, **MSI** 1813
- **44318** candidate images extracted
  - Does a binary look like UEFI image? (CHIPSEC, uefi-firmware-parser)
  - Other binary heuristics (known magic values etc.)
  - File extensions: ROM, BIN, IMG, BIO, CAP, IMA, FD, WPH, HDR, FL*…
- Parsed and analyzed **21204 unique UEFI firmware images** (extracted from **19150** update packages)
  - The rest are either legacy BIOSes or we couldn’t extract/parse

UEFI “update” must be generic?

- Vendor firmware update != UEFI image or UEFI update image
  - There’s no standard format of UEFI firmware updates
- Examples of what we saw in firmware update packages:
  - May contain update utilities for different OS (EFI, DOS, Windows, Linux)
  - UEFI Images may be encrypted inside updates
  - May contain multiple types of firmware images used in different cases
  - May have firmware images embedded into update tool executables

They should be easy to extract, right?

- Try common (de)compression utilities: `zip, 7z, zlib-flate`. If decompression doesn't work: `binwalk -e`
- Utilities required for update packages of specific vendors
  - `cabextract` (Lenovo)
  - `innoextract` (Lenovo)
  - `InsydeFlash.exe -cpf` (HP)
- Command-line arguments required for self-extracting update packages
  - `/writeromfile` (Dell)
  - `/VERYSILENT` (Lenovo)
- Can also try other things
  - Mount and extract firmware images if update package if ISO
  - Run update package and monitor file system (e.g. temp directories)

Searching for SPI descriptors…

- Exclude known UEFI “capsule” images (e.g. *.FL1/FL2/CAP files)
- Include only images with exact 2MB, 4MB, 8MB, 16MB size
- Include only images with valid SPI flash descriptor at offset `0x00`
- Include only images with Read/Write-able SPI descriptor

Valid Flash Descriptor

```
##############################################################
# SPI FLASH DESCRIPTOR
##############################################################

+ 0x0000 Reserved : FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
+ 0x0010 Signature: 0x0FF0A55A
[*] FLMAP0 = 0x00040003 << Flash Map 0 Register (FDBAR + 0x14)
    [00] FCBA              = 3 << Flash Component Base Address
    [08] NC                = 0 << Number of Components
    [16] FRBA              = 4 << Flash Region Base Address
[*] FLMAP1 = 0x58100208 << Flash Map 1 Register (FDBAR + 0x18)
    [00] FMBA              = 8 << Flash Master Base Address
    [08] NM                = 2 << Number of Masters
    [16] FPSBA             = 10 << Flash PCH Strap Base Address
    [24] PSL               = 58 << PCH Strap Length
[*] FLMAP2 = 0x00310330 << Flash Map 2 Register (FDBAR + 0x1C)
    [00] FCPUSBA           = 30 << Flash CPU Strap Base Address
    [08] CPUSL             = 3 << CPU Strap Length
```

R/W Access to Flash Descriptor

```
 Region                | FLREGx    | Base     | Limit
-------------------------------------------------------
0 Flash Descriptor     | 00000000  | 00000000 | 00000000
1 BIOS                 | 0FFF0700  | 00700000 | 00FFF000
2 Intel ME             | 06FF0003  | 00003000 | 006FF000
3 GBe                  | 00020001  | 00001000 | 00002000
4 Platform Data        | 00007FFF  | 07FFF000 | 00000000 (not used)
8 Embedded Controller  | 00007FFF  | 07FFF000 | 00000000 (not used)

+ 0x0080 Master Section:
=======================================================
+ 0x0080 FLMSTR0   : 0xFFFFFF00
+ 0x0084 FLMSTR1   : 0xFFFFFF00

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

R/W Flash Descriptor

Our Suspects

Unique UEFI Images	Analyzed	Full SPI Images
Acer	312	3 (1%)
ASRock	440	73 (16.6%)
ASUS	3697	629 (17%)
Dell	4673	78 (1.7%)
Gigabyte	1330	1117 (84%)
HP	1593	94 (5.9%)
Intel	4387	0
Lenovo	3053	75 (2.5%)
MSI	1719	1461 (85%)
Total	**21204**	**3530 (16.6%)**

False Positives

- Not all update packages containing full SPI images indicate that corresponding systems are vulnerable
- Some images can only be flashed from USB thumb drive during BIOS Setup (requires user interaction)
- Some updates packages include full SPI images along with signed capsules which may be used at manufacturing(?)

False Negatives

- Presence of signed capsule in the update package (or absence of full SPI image) does **not** mean system flash is protected
  - Example: ASUS P8Z77-PRO here
- Update packages may embed SPI images into executables of update utilities which we couldn’t extract
- We excluded images with Read-Only flash descriptor
- Capsule images may be unsigned

Vulnerable Systems

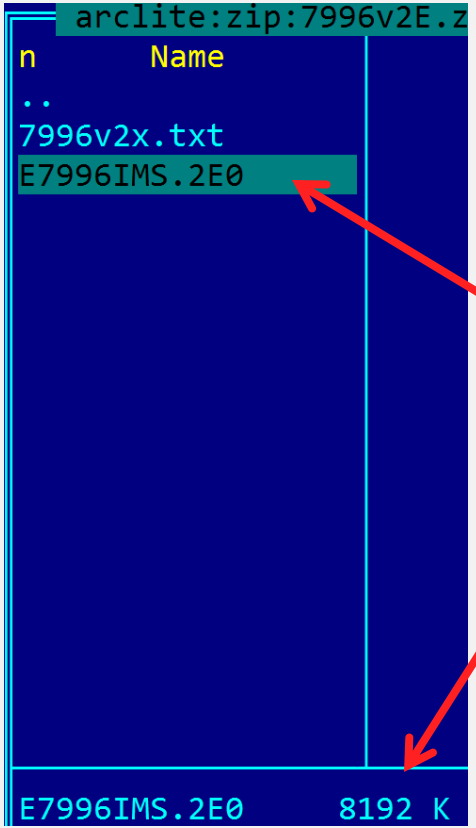
Manufacturer	Vulnerable firmware images	Vulnerable models
Acer	0 - 2	0 – 2
ASRock	73	**~53** models (all older than Skylake)
ASUS	629	**~61** models (all older than Ivy Bridge)
Dell	51	**~11** models (Vostro and Inspiron older than 2014)
Gigabyte	1117 (345 Skylake+)	**~247** models including Skylake (6 Gen Intel Core) or newer
HP	84	**~74** (business desktops older than Skylake)
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	**3490 (16.4%)**	**~570 models**

Results: MSI

- **1461 UEFI update images for ~98 models** appear to be vulnerable
- Including **496** Skylake (2015) and newer

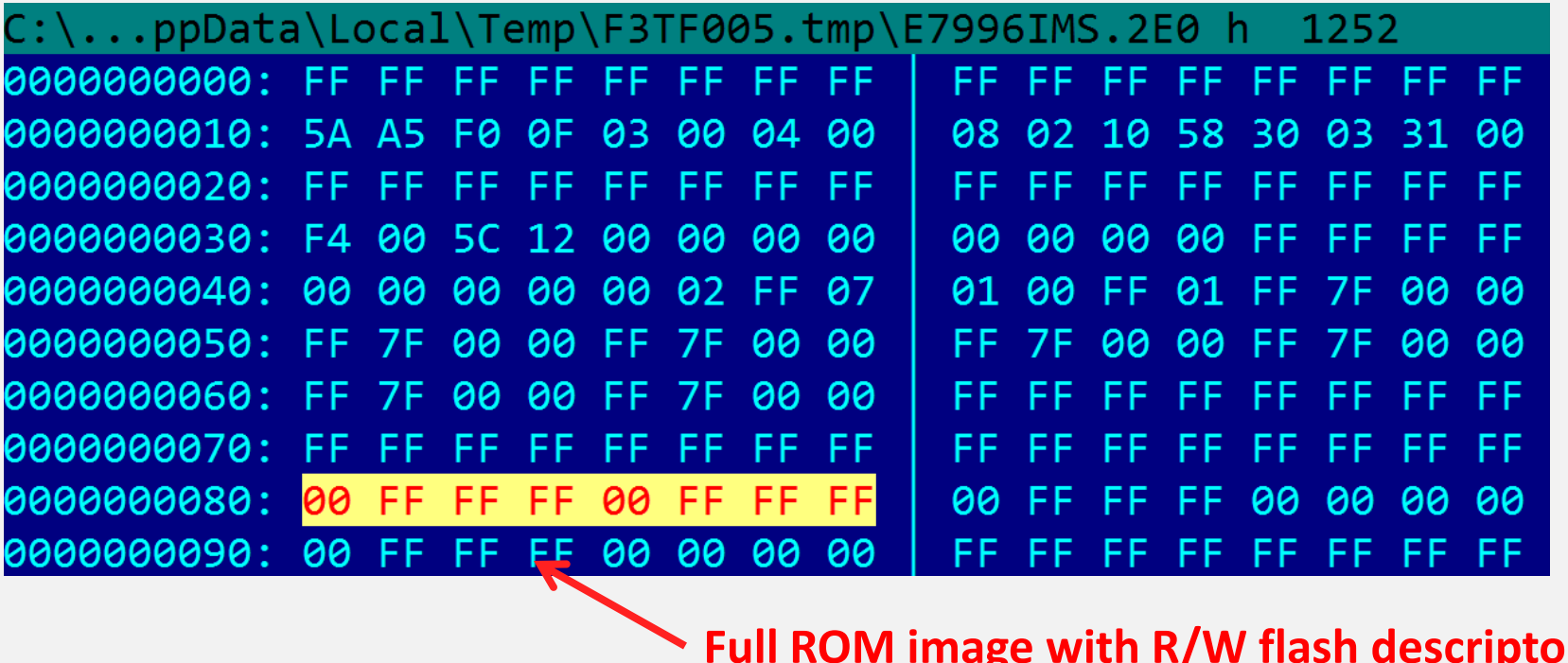
- Confirmed on some of MSI systems
- Example: **MSI H110 PRO-VD**

Example: MSI H110 PRO-VD (BIOS V2.E)



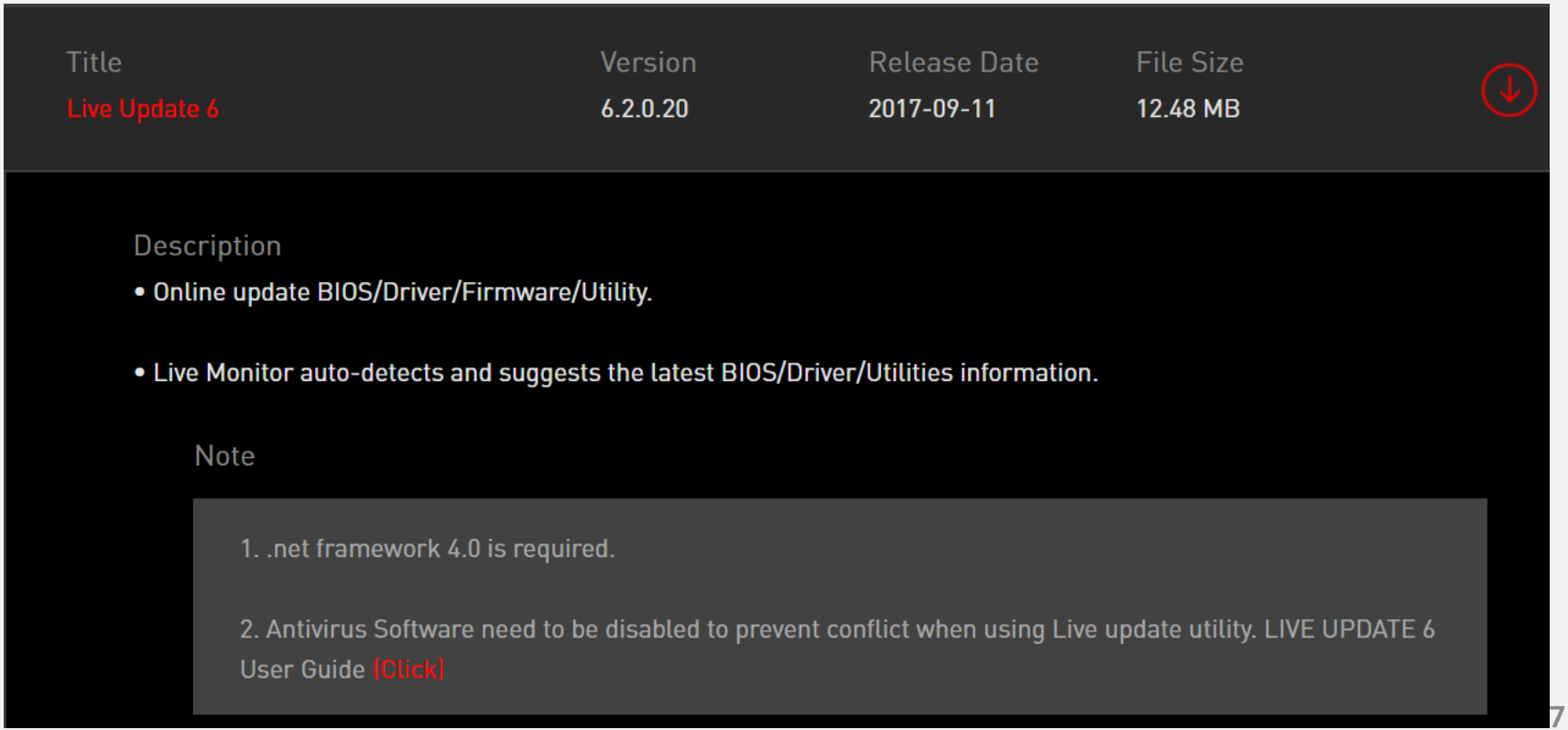
7996v2E UEFI update (V2.E) contains full ROM image intended for flashing under Windows, DOS or EFI shell with “flashers” and MSI Live Update

Example: MSI H110 PRO-VD (BIOS V2.E)



Full ROM image with R/W flash descriptor

MSI Live Update



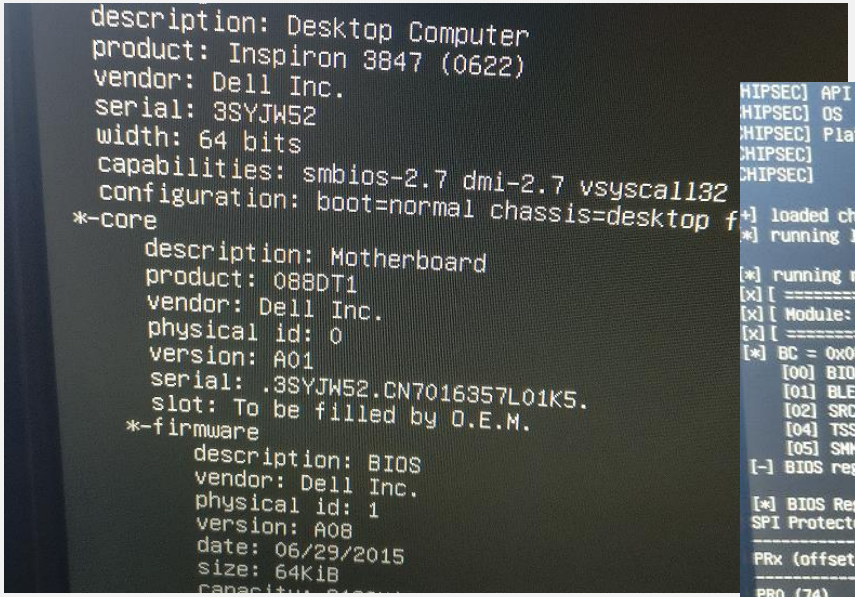
Results: Gigabyte

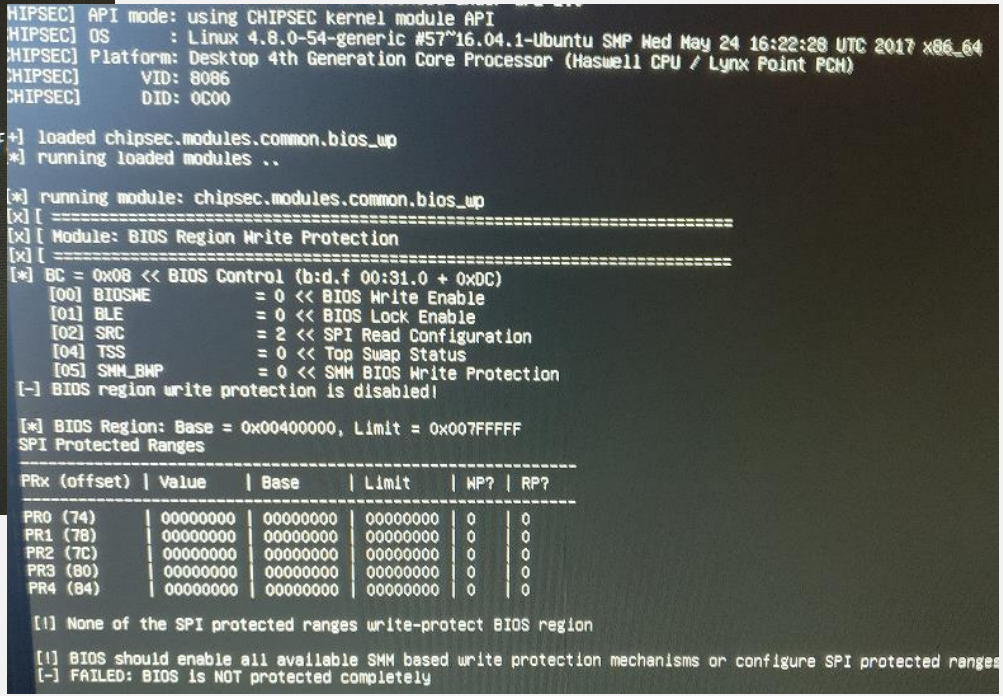
- **1117 UEFI update images for ~247 models** appear to be vulnerable
- Including **345** Skylake (2015) and newer

Results: Dell

- 78 UEFI update images corresponding to 24 models are suspects
- 13 update images for 4 models are false positives. Updates are using signed capsules but also includes full SPI images
- **51 update images for 11 models** appear to be vulnerable
  - Inspiron & Vostro 2011–2014 models with updates up to 2016
  - Confirmed on **Dell Inspiron 3847** desktop (circa 2013, UEFI firmware release 06/2015)
- Investigating 14 update images for 8 models
  - Full SPI images with R/W descriptors via option `/writeromfile`

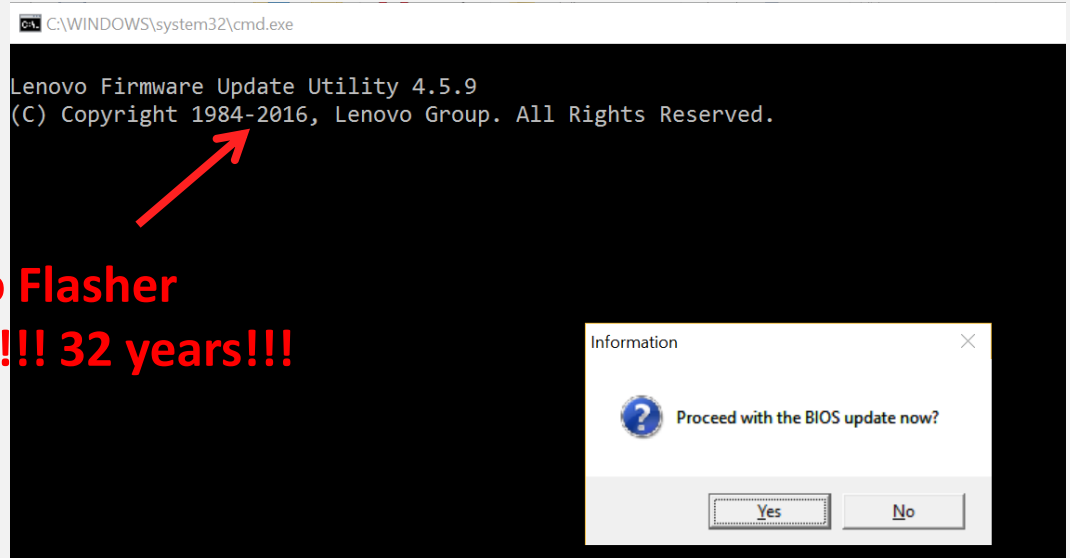
Example: Dell Inspiron 3847





Results: Lenovo

- **Up to 75 UEFI update images for 26 models** appear to be vulnerable (based on the analysis of update packages/images)
- Investigating systems which don't seem to protect UEFI firmware: ThinkServer (TS 150 - 550), ThinkCentre and IdeaCentre



Lenovo Flasher
1984 – 2016!!! 32 years!!!

Results: HP

- **Up to 84 UEFI update images for ~74 models** appear to be vulnerable (based on the analysis of update packages/images)
  - HP/Compaq business desktops (2011 - 2014)
  - All images are older than Skylake (< 2016)
- Examples:
  - Workstation Z2x0/Z420/Z820
  - Workstation Z1 G1/G2
  - ProDesk/EliteDesk G1/G2, ProOne/EliteOne G1
  - Signage Player MP8/MP9, RPx Retail System, t820 …
- 7 update packages are false positives. Include full SPI image
- Around SoftPaq SP77xxx switched to signed updates?

Results: ASUS

- **629 UEFI update images for ~61 models** appear to be vulnerable
- All vulnerable systems are older than Ivy Bridge (< 2013)
- Starting Ivy Bridge ASUS appears to have switched to using signed UEFI capsules

Results: ASRock

- Only have small pool of downloaded update packages (440)
- **73 UEFI update images for ~53 models** appear to be vulnerable
- All vulnerable systems are older than Skylake (< 2016)

ANALYZING UEFI UPDATES FOR DEFENSIVE PURPOSES

How to build “white-list” for UEFI?

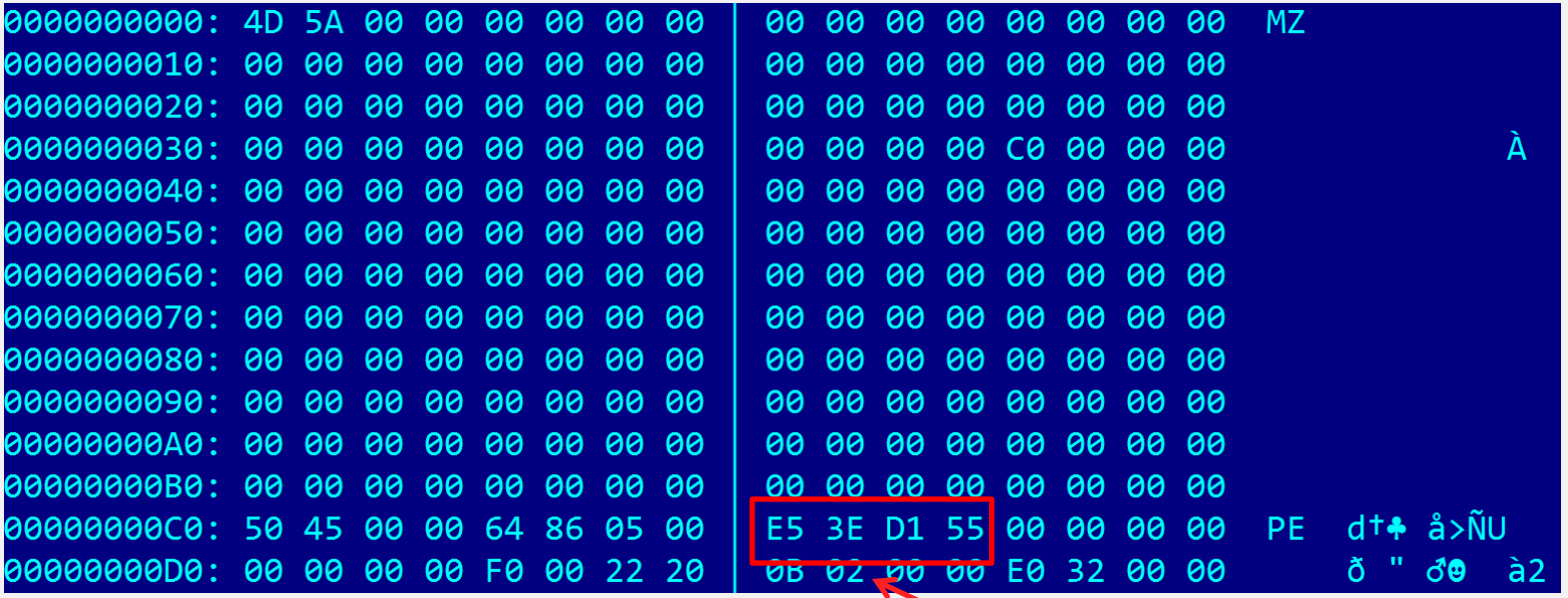
- We cannot just collect hashes of entire ROM images
  - Contain modifiable data: NVRAM settings, ACPI tables, x509 certificates etc.
- UEFI firmware volumes contain PE/COFF or TE executables
  - **45 – 90** unique executables per UEFI firmware **update** image on average
  - **100 - 300** executables within full UEFI firmware image on a system
- We can build a list of hashes of known UEFI executables

Collecting UEFI hashes…

- Calculating hashes
  - Plain hash over entire PE/COFF image
  - Authenticode compliant hashes
- Most platform vendors use Authenticode hashes for (U)EFI binaries
  - TPM and UEFI Secure Boot use Authenticode hashes
- All of the above?
  - **~ 1.9M** plain or Authenticode compatible hashes
  - **~ 1M** Authenticode hashes with masked TimeDateStamp field

Authenticode Hashes

- Authenticode hash calculation for PE/COFF executables
  1. Hash PE header omitting the file's **Checkum** and the **Certificate Table** entry in optional **Header Data Directories**
  2. Hash PE sections
  3. Exclude **Attribute Certificate Table** from the hash calculation and hash any remaining data
- Open source Authenticode implementations
  - https://github.com/anthrotype/verify-sigs
  - https://github.com/illphil/authenticode



Masking **TimeDateStamp** field

Unique UEFI Hashes

Unique Hashes	Plain	Authenticode	Authenticode and TimeDateStamp=0
Acer	37292	35104	23231 (62%)
ASRock	26168	26170	1671 (6%)
ASUS	559857	549175	171948 (31%)
Dell	485970	476519	234135 (48%)
Gigabyte	168119	158328	109873 (65%)
HP	102631	97524	82632 (80%)
Intel	106924	98562	63363 (59%)
Lenovo	166212	150313	140038 (84%)
MSI	271365	257461	192731 (71%)
Total	**1910649**	**1849156**	**1034661 (54%)**

False Positives

```
chipsec_main.py -i -n -m tools.uefi.whitelist -a
check,efi_lenovo.json,lenovo_t430.bin
```

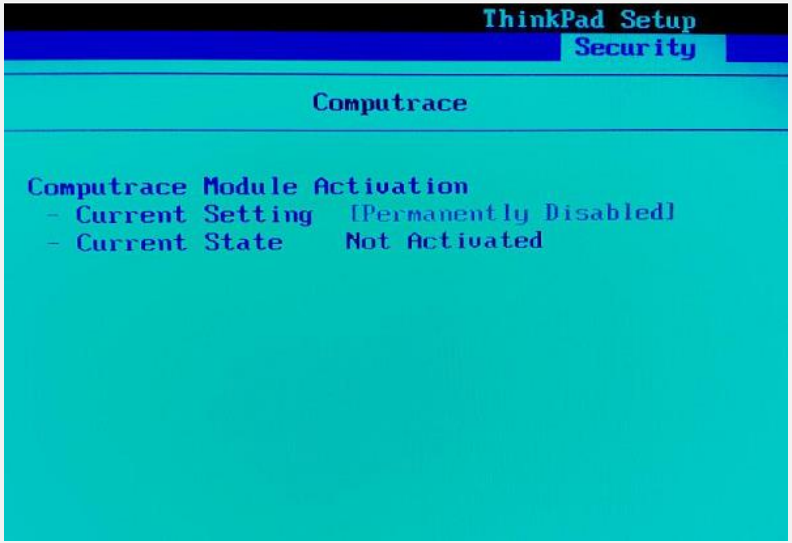
```
[+] loaded chipsec.modules.tools.uefi.whitelist
[*] running loaded modules ..

[*] running module: chipsec.modules.tools.uefi.whitelist
[*] Module arguments (3):
['check', 'efi_lenovo.json', 'lenovo_t430.bin']
[x][ ================================================================================
[x][ Module: simple white-list generation/checking for (U)EFI firmware
[x][ ================================================================================
[*] reading firmware from 'lenovo_t430.bin'...
[*] checking EFI executables against the list '/home/virvdova/chipsec/chipsec/efi_lenovo.json'
[*] found 414 EFI executables in UEFI firmware image 'lenovo_t430.bin'
[!] found EFI executable not in the list:
    9894c265bdd79a01ef94734cc576e0dd13f21854d8b1e23ba362349b6728ce21 (sha256)
    f22ce4c0081ef57d81dbfcd8ac2360ecec19b73a (sha1)
    {5920F406-5868-44F5-A9B9-6D4031481CC9}
    LenovoOemSecPei.efi
[!] WARNING: found 1 EFI executables not in the list '/home/virvdova/chipsec/chipsec/efi_lenovo.json'

[CHIPSEC] ***************************  SUMMARY  ***************************
```

This module has never been part of any update

We can also gather statistic on components/ features supported by various systems

Example: Absolute Computrace

- UEFI based Anti-Theft technology
- Contains UEFI firmware and OS level components
- Absolute Computrace Revisited



We decided to apply complex heuristic…

…search for “computrace” in the name

AbsoluteComputraceInstaller
AbsoluteComputraceInstallerWin8
BdsComputrace
BdsSmmComputrace
Computrace
ComputraceComponents
ComputraceDriver
CompuTraceDriver
ComputraceDxe
ComputraceDXE
ComputraceEnablerDxe
ComputraceLoader
ComputraceSMI
ComputraceSmm
ComputraceVariableInitDxe
DellDxeComputrace
DellSmmComputrace
DellSmmComputraceAcpiMode
DellSmmComputracePreInit
H19ComputraceRuntimeDxe
H19ComputraceSmm
HPComputrace
HPComputracePrivateSrc
L05Computrace
L05CompuTraceDxe
L05ComputraceEfi
L05SmmComputrace
LenovoComputraceEnablerDxe
LenovoComputraceLoaderDxe
LenovoComputraceSmiServices
LoadComputraceImage
SmbiosComputraceDxe
smmcomputrace
SmmComputrace
UEFIComputrace
UEFIComputraceDriver
UEFIEfiSmmComputrace
UEFIL05Computrace
UEFIL05SmmComputrace

Results

Vendor	*Computrace* Modules	Unique UEFI Images
Acer	146	57
ASRock	0*	
ASUS	624	312
Dell	6103	3262
Gigabyte	0*	
HP	2567	1365
Intel	0*	
Lenovo	8065	2231
MSI	0*	
Total	**17506**	**7228**

* Modules weren't found but may still be present under different names

Future Improvements

- Current heuristic “detect full ROM image vs capsule” is imprecise
- Explicitly detect capsules in update packages
  - firmware update DXE driver FV (`SysFirmUpdate.efi`)
  - SystemFirmwareDescriptor PEIM
  - Signature in `EFI_FIRMWARE_IMAGE_AUTHENTICATION` block
- Detect that ROM images inside update packages support signed capsule (FmpAuthenticationLib) & secure update (SecSMIFlash, PchBiosWrireProtect)
- Currently, we cannot answer this question: “Did particular system start protecting firmware with some update?”

Conclusions

- **~3,490 update images** corresponding to **~570 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
- Some manufacturers had basic firmware protections for a while. Yet older systems may be forgotten
- Some manufacturers started recently (> Ivy Bridge or Skylake)
- Some manufacturers yet to start protecting UEFI firmware

Conclusions

- Offline analysis of updates can help us understand overall health of firmware across entire population of systems
  - No need to test each individual system
  - Can detect systems lacking basic firmware security protections
  - Can be used to scan updates for other more complex problems
  - Not perfect, needs improvements
- Helps vendors understand which systems they forgot to fix
- Can also help us build global database of known firmware binaries
- This is an ongoing study. We'll keep updating it with further results…

THANK YOU!