HAVE YOU SCANNED YOUR BIOS RECENTLY?

Aditya Kapoor, Cylance
akapoor@Cylance.com

Virus Bulletin Conference, 2017, Madrid.

CYLANCE

Who Am I

- 15 years in security Industry (ULL, McAfee, Intel ?, Cylance)
- Wrote malware signatures and cleaning back in the day.
- Primary contributor to rootkit scanner at MFE. POC Deepdefender, Intel Tech and Backend Infrastructre.
- Current interests Firmware Security, Static program anlysis and building product features.
- Loves playing Cricket and Tennis.
- Last VB talk in 2011 on rootkits.

What this talk is about

Coerce the AV industry to spend more money on firmware defense

CYLANCE

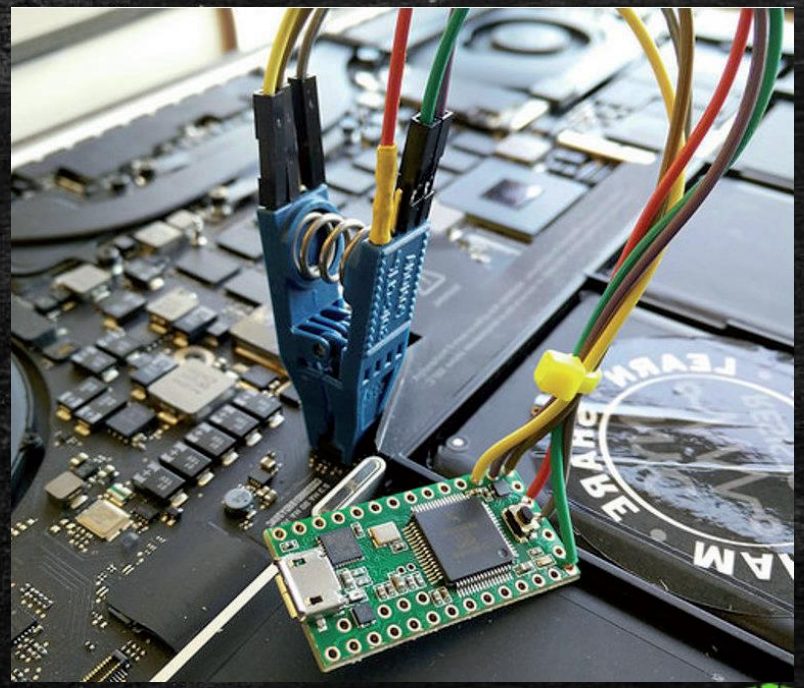
What this talk is (really) about

- UEFI (Unified Extensible Firmware Interface) Overview
- UEFI-attacks > UEFI-Defense
  - State of affairs of UEFI attack and Defense
  - Are UEFI attacks real/costly to build?
  - Time to turn the tables?
- UEFI Scanner – What/Why/Who?
  - AV Problem or Firmware/hardware vendors problem?
- And coercing … ☺

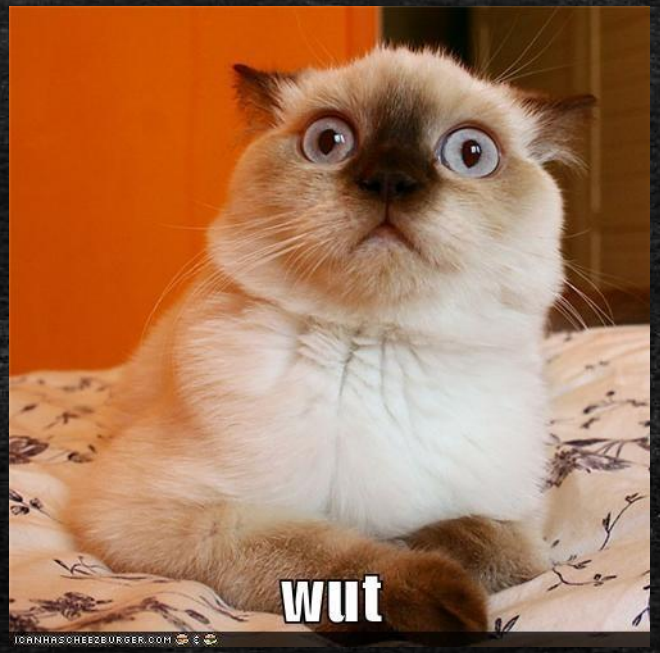
CYLANCE

What this talk is NOT about

- New L33t UEFI vulnerabilities (I leave that to experts)
- How to brick (er.. Debug) your hardware

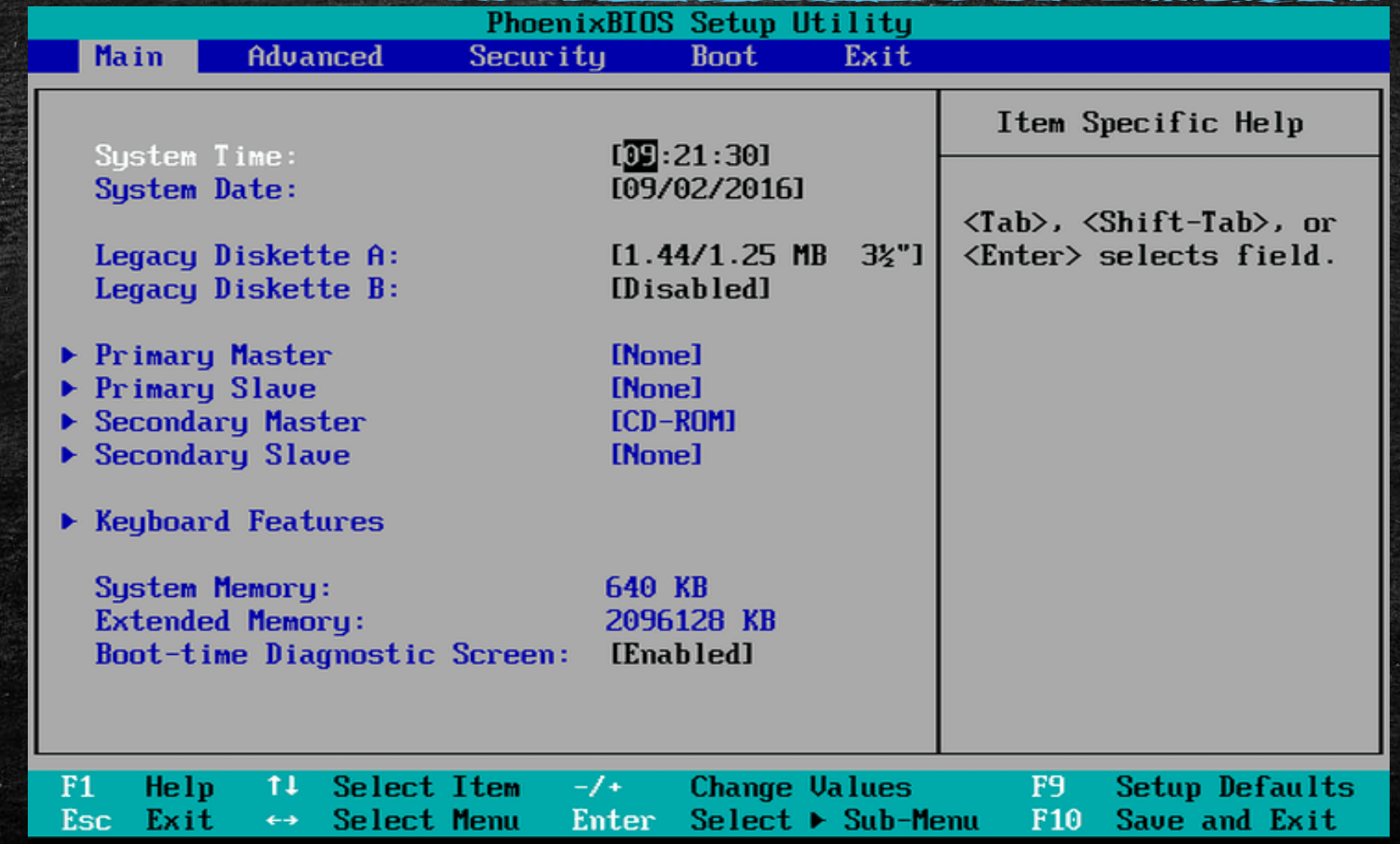


=



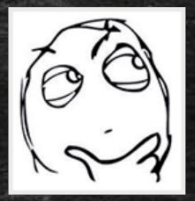
CYLANCE

Legacy Bios


PhoenixBIOS Setup Utility
Main Advanced Security Boot Exit
System Time: [09:21:30]
System Date: [09/02/2016]
Legacy Diskette A: [1.44/1.25 MB 3½"]
Legacy Diskette B: [Disabled]
▶ Primary Master [None]
▶ Primary Slave [None]
▶ Secondary Master [CD-ROM]
▶ Secondary Slave [None]
▶ Keyboard Features
System Memory: 640 KB
Extended Memory: 2096128 KB
Boot-time Diagnostic Screen: [Enabled]
Item Specific Help
<Tab>, <Shift-Tab>, or <Enter> selects field.
F1 Help ↑↓ Select Item -/+ Change Values F9 Setup Defaults
Esc Exit ↔ Select Menu Enter Select ▶ Sub-Menu F10 Save and Exit

Legacy Bios

- Created in 1975 by IBM
  - Hmm….. This should last us 2 years



- 

years later we are still discussing it.

- BIOS initializes CPU, RAM, does POST (Power on self test) and then it checks for any option ROMs (LAN, PCI), then passes the control to boot loader.

Legacy Bios Challenges

- Hardware dependent and inflexible updates.
  - Everytime a new hardware is introduced, new updates and workarounds needed to be added.
- Limited execution space in 16bit real mode
  - Small option ROMs
  - Assembly code for BIOS , complex updates.
  - Size of bootable devices is 2.2 TB max.
- Security challenges.
  - Minimal signed bios
  - No verification at the boot load time.

UEFI Bios


UEFI BIOS UTILITY – EZ MODE
Exit/Advanced Mode
09:46:40
Sunday[06/02/2013]
SABERTOOTH Z87 BIOS Version : 0801
CPU Type : Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz Speed : 3508 MHz
Total Memory : 16384 MB (DDR3 1333MHz)
English
CPU Information
Temp. +100.4°F/+38.0°C
Voltage 1.392V
Dram Information
DIMM_A1:N/A
DIMM_A2:Corsair 8192MB 1333Mhz
DIMM_B1:N/A
DIMM_B2:Corsair 8192MB 1333Mhz
Fan
CPU_FAN 1314RPM
Standard
CPU_OPT_FAN 1305RPM
CHA_FAN1 N/A
Standard
System Performance
Power Saving
Normal
ASUS Optimal
Quiet
Performance
Energy Saving
Boot Priority
Use the mouse to drag or keyboard to navigate to decide the boot priority.
Shortcut(F3)
Advanced Mode(F7)
Boot Menu(F8)
Default(F5)

UEFI Bios

- EFI-Extensible Firmware Interface
- Intel created it in early 2000's , since processors were 64 bit but bios were still 16 bit.
- 2005 -> Unified EFI was born.
- UEFI -> is actually just a specification, vendors can use the spec to create their own bios.
- Why EFI? Why not Legacy Bios?

UEFI Bios

- Benefits
  - Has programing language (C)
  - No Limits on option ROMs. (64kb, hardware dependent)
    - Replaced by drivers.
  - Supports HDD > 2.2TB
  - Supports modern hardware needs.
    - System management
    - Power management
    - Remote services
    - Enhanced security
    - Supports Secure boot, larger HD
    - Faster boot times.
- Post UEFI - Standardization
  - 330+ members of UEFI
  - Supports Intel/ARM and Windows, Ubuntu, RedHat etc.
  - Routers, Scada, Automotive and IOT devices.

CYLANCE

Bios attacks in the news

KIM ZETTER SECURITY 03.20.15 02:39 PM

HACKING BIOS CHIPS ISN'T JUST THE NSA'S DOMAIN ANYMORE

Karl M.
Vulnerability Researcher (wannabe)
Aug 18, 2016 • updated Aug 18, 2016 • last reply Aug 22, 2016 • 137 views

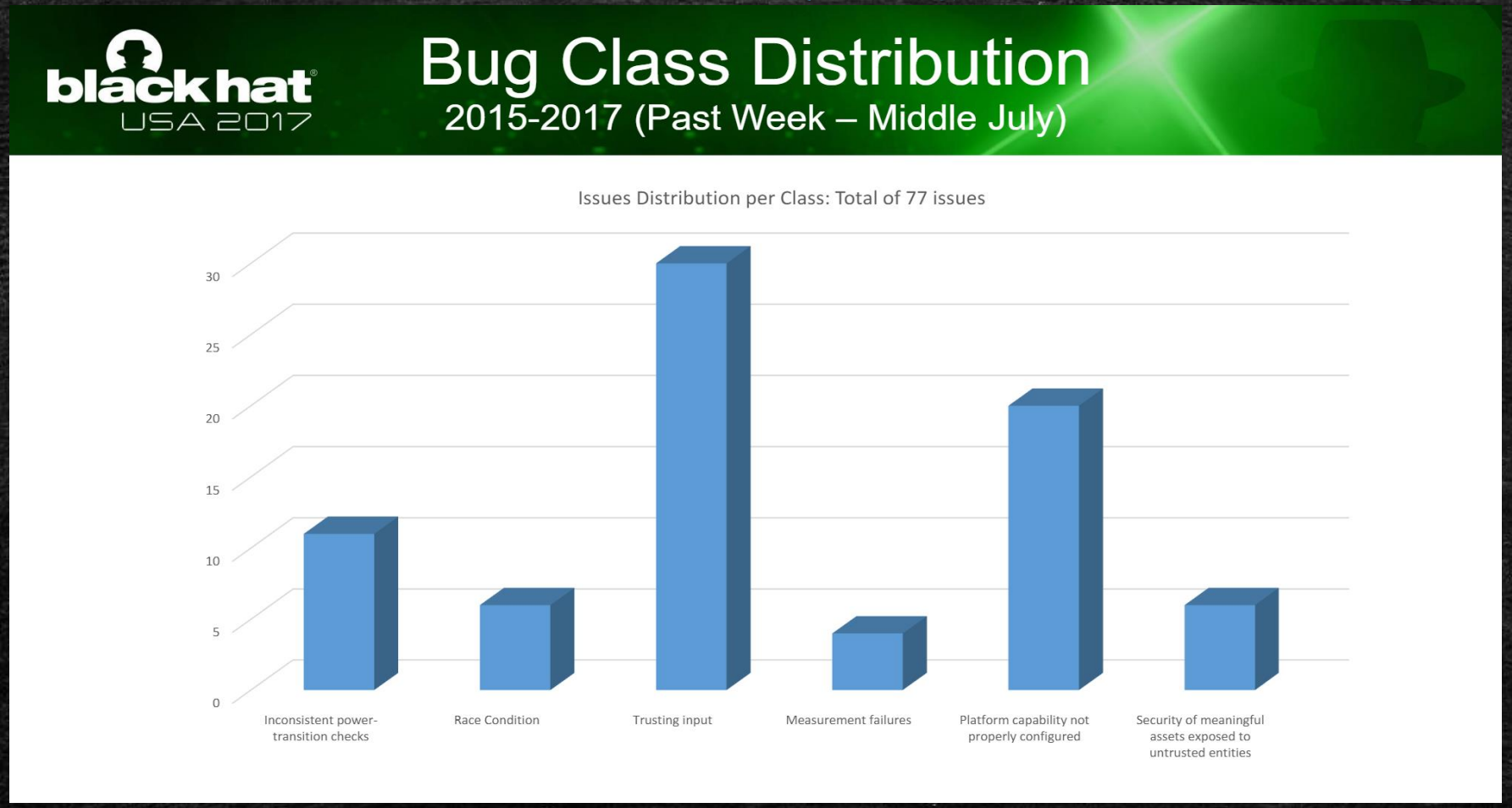
Follow

NSA has put multiple BIOS implants into their attack tools - equation group leak

Hacking Team Spyware preloaded with UEFI BIOS Rootkit to Hide Itself

Tuesday, July 14, 2015 Mohit Kumar

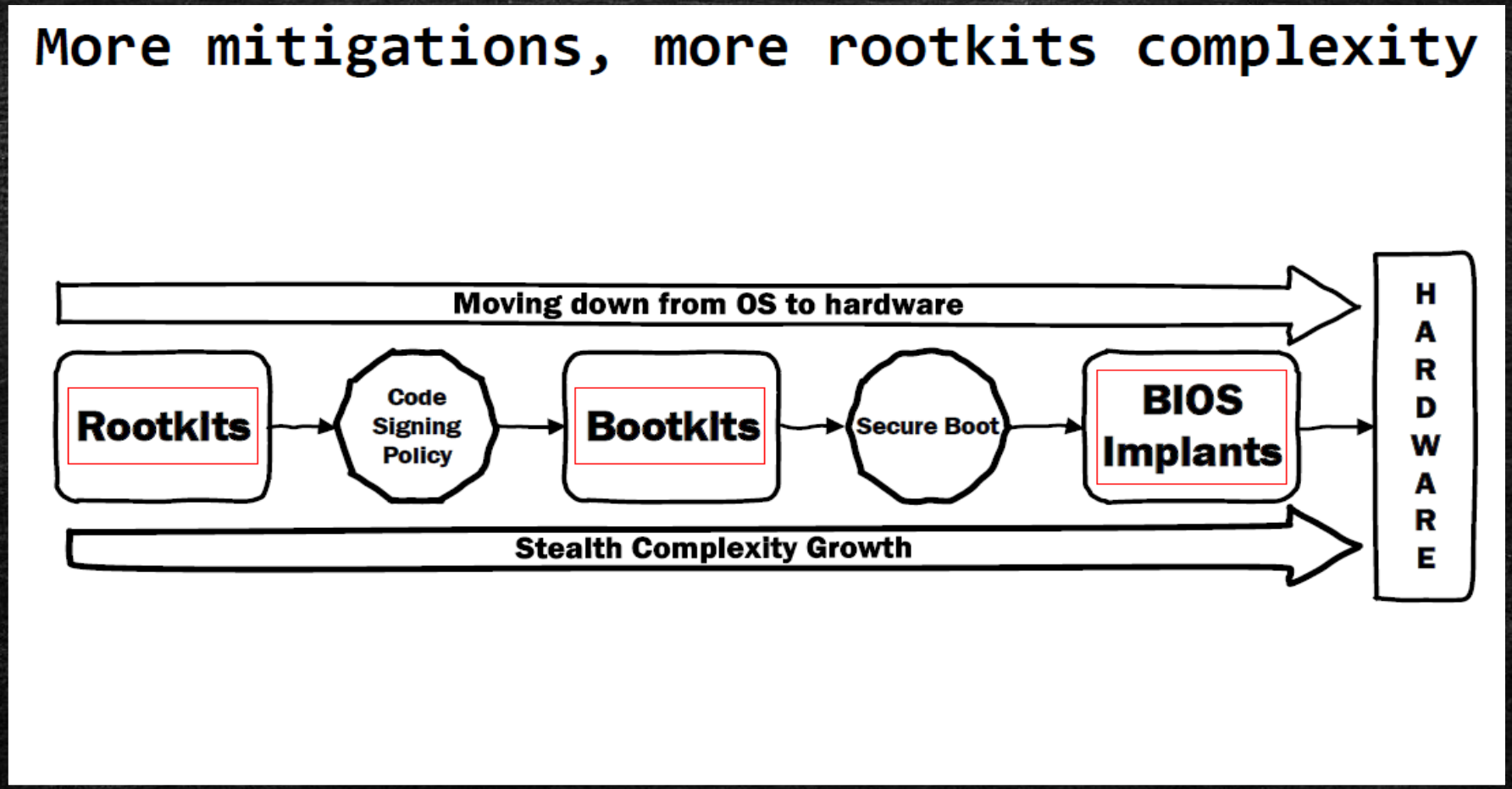
Current state of UEFI attacks



Intel provided Data at BH2017

CYLANCE

Why target UEFI?



CYLANCE

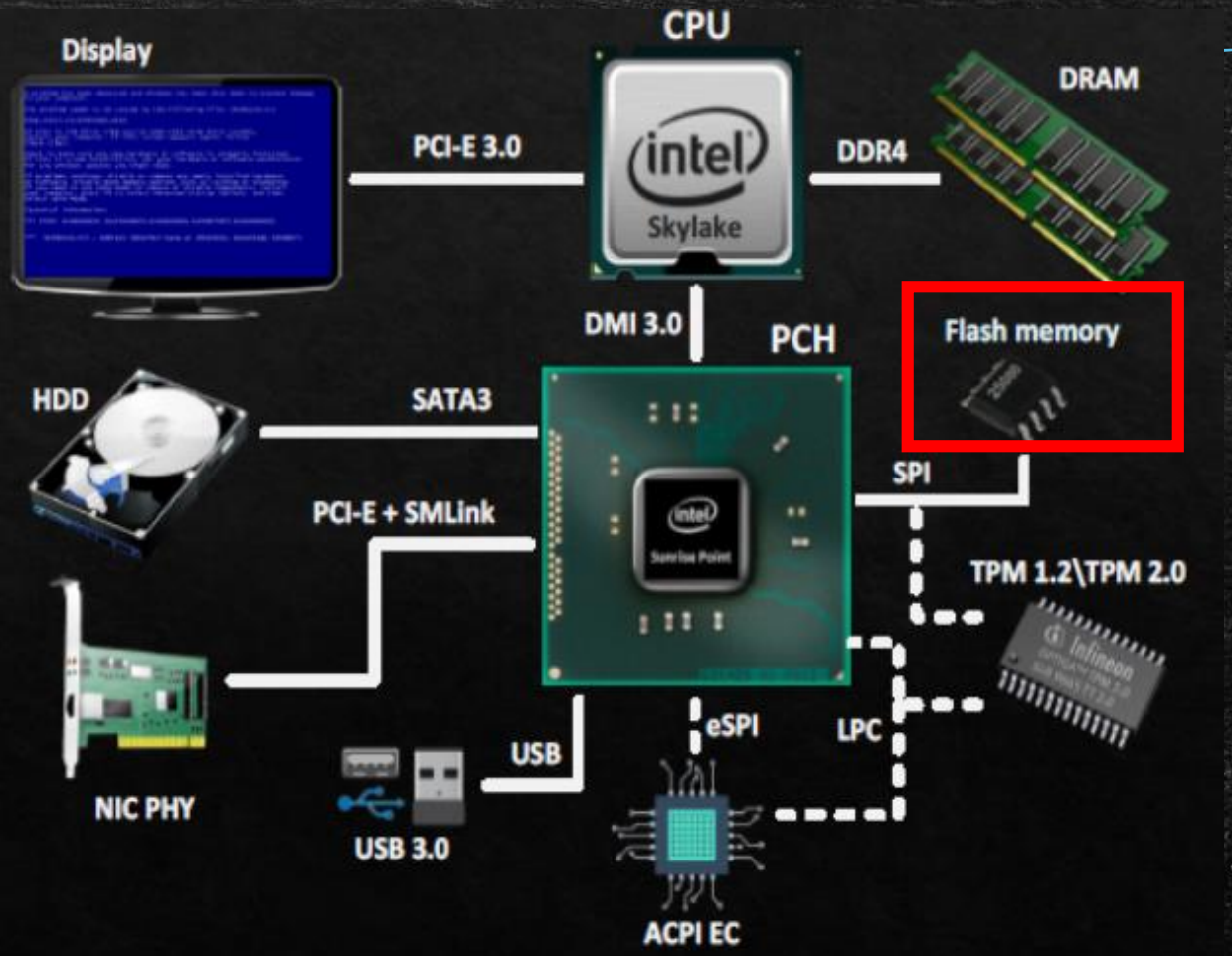
*Image Credits - https://github.com/REhints/BlackHat_2017/blob/master/Betraying the BIOS.pdf

System overview

Execution environments:

- CPU
- Chipset
- ACPI EC

The main part of platform firmware is stored on SPI flash memory

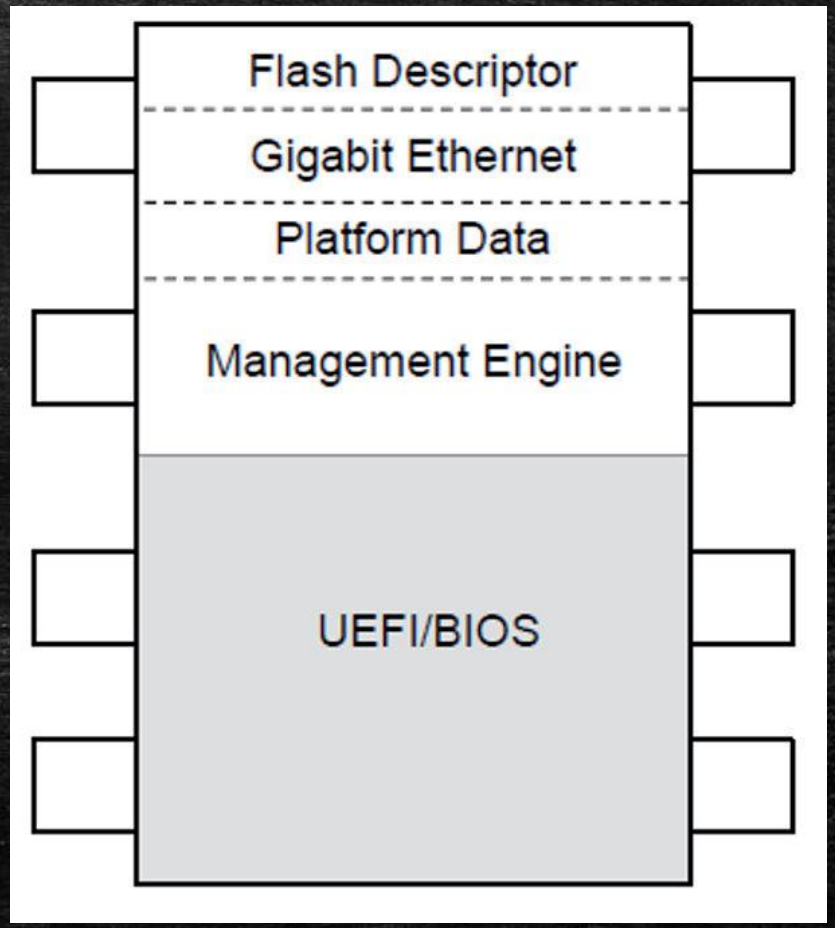


https://www.slideshare.net/DefconRussia/dcg29-safeguarding-rootkits-intel-boot-guard-part-ii

CYLANCE

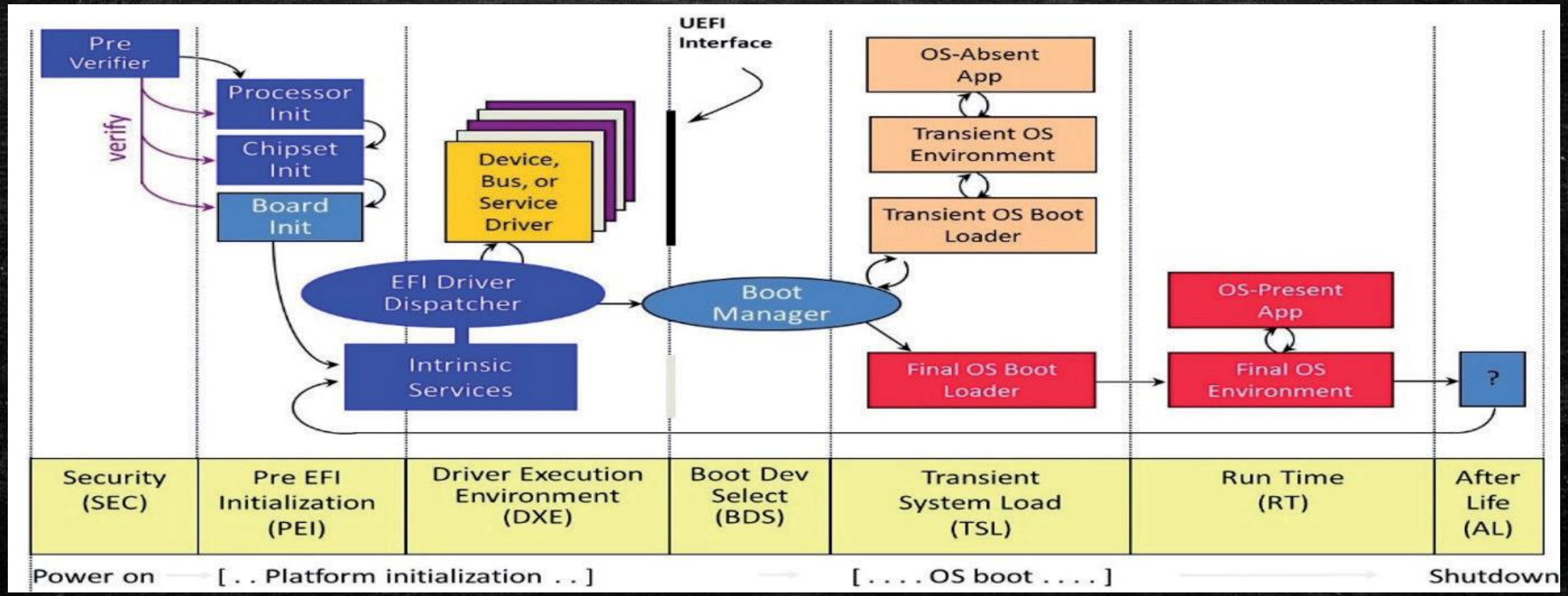
Uefi firmware storage layout

- Flash Storage is typically divided into five sections, the first of which describes the flash layout.
- The flash descriptor starts with the signature 0x0FF0A55A at offset 0x10 and contains various components like a descriptor map and region. The region contains offsets and the size of the BIOS region.
- The flash descriptor is always the first region and the BIOS is always the last one on the chip.

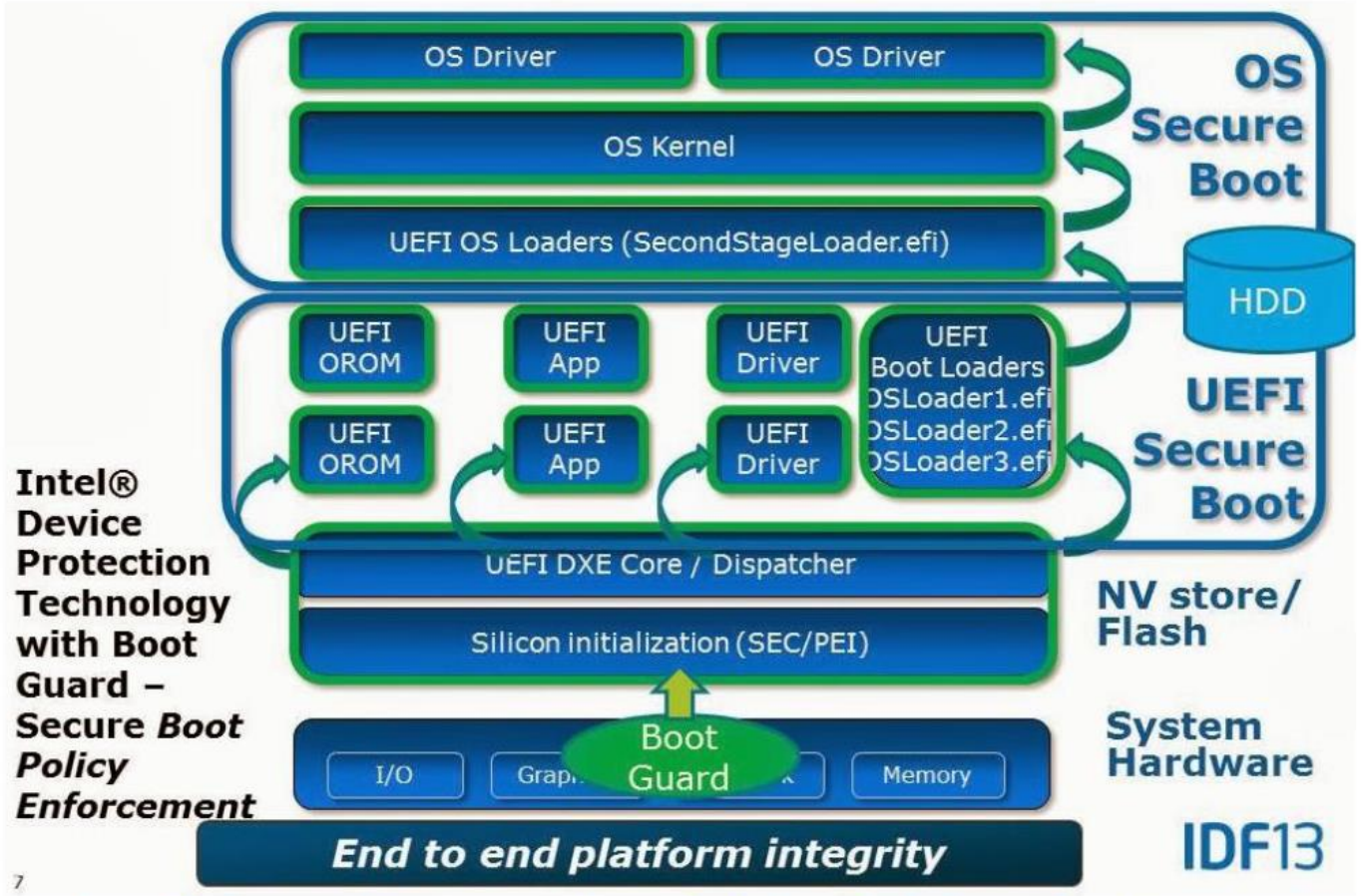


CYLANCE

Uefi boot process

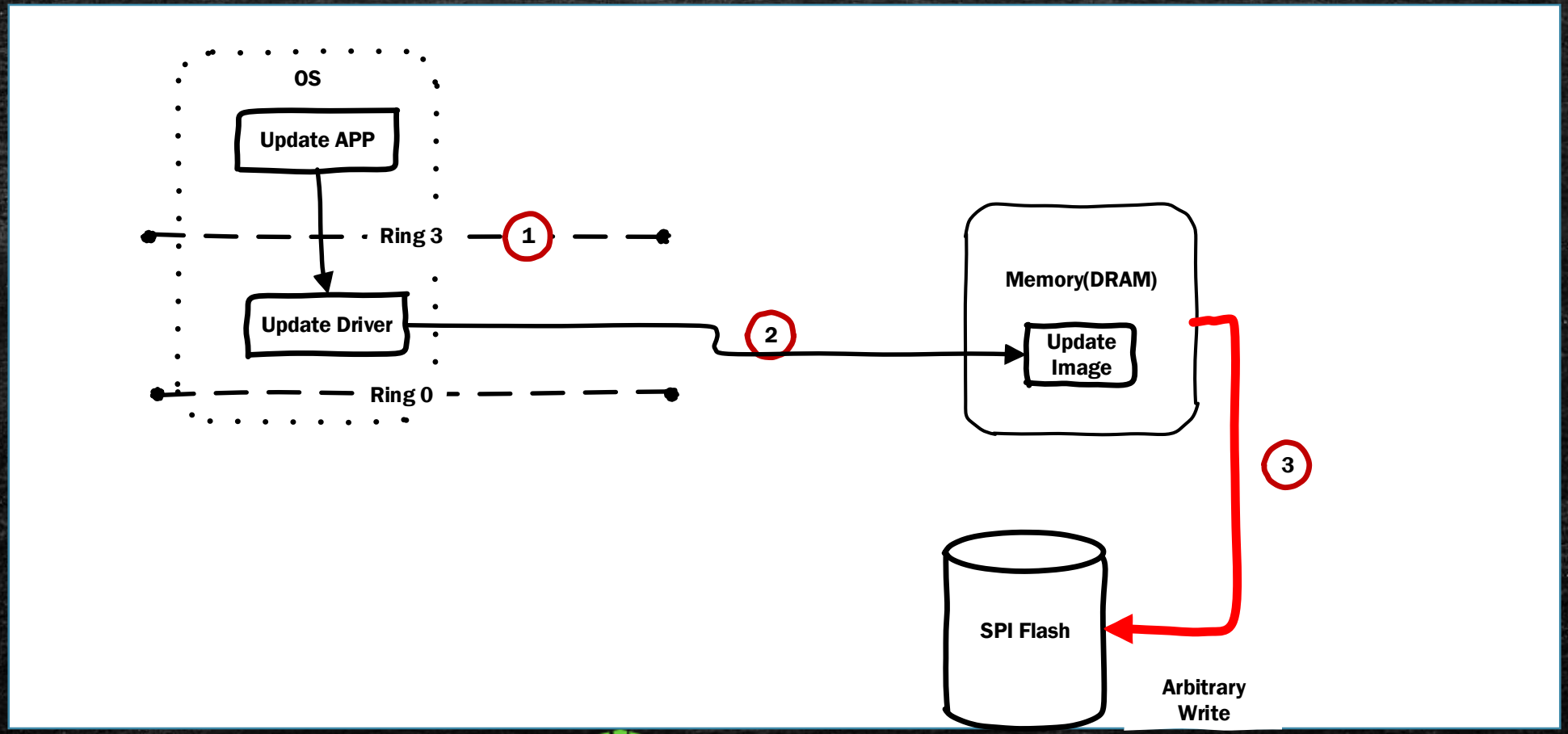


Intel boot guard , supplying root of trust.



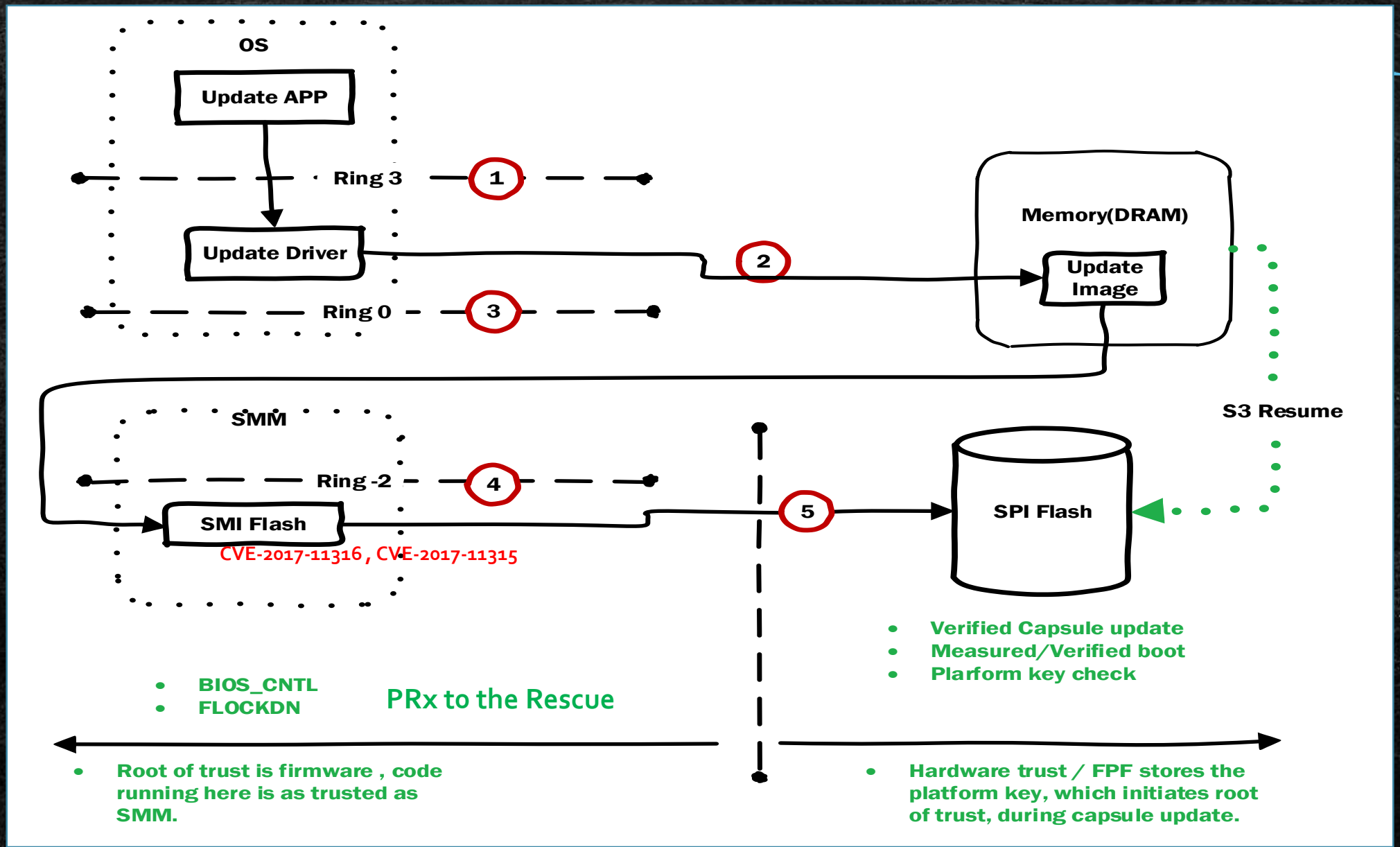
http://vzimmer.blogspot.com/2013/09/where-do-i-sign-up.html

Arbitrary writes to SPI flash – Worst case scenario



CYLANCE

Typical Bios update path with security enabled.



Current state of UEFI security

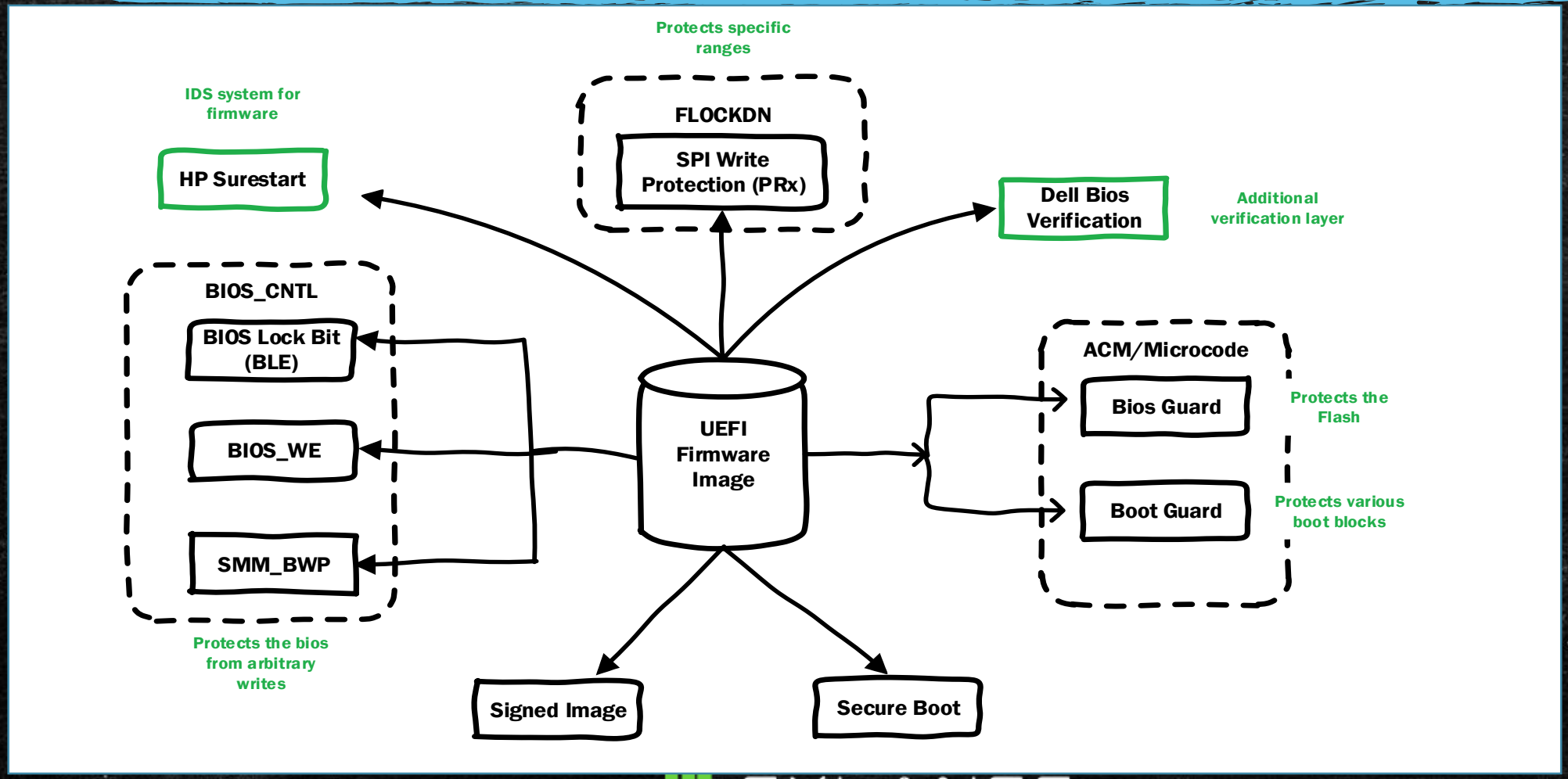
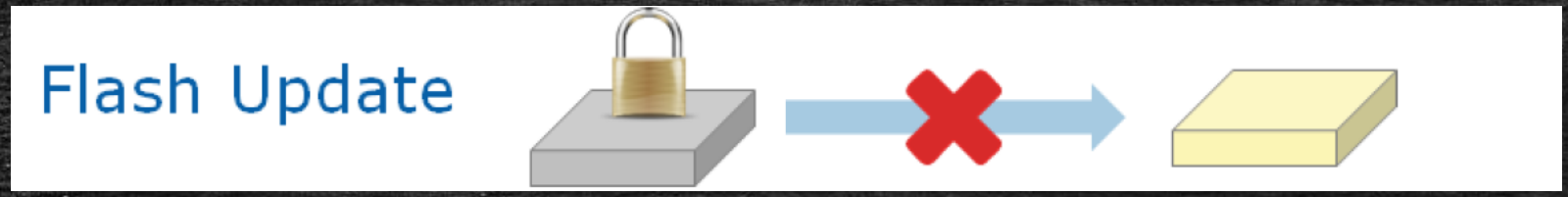
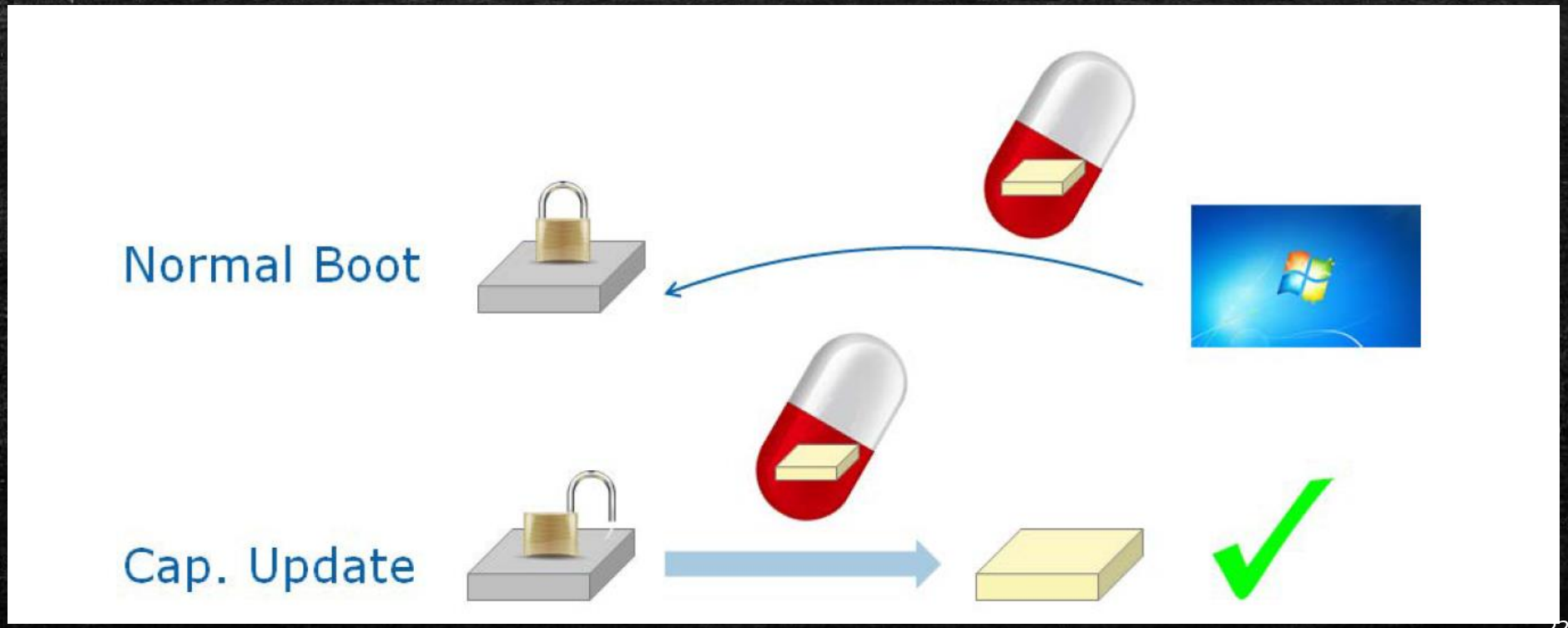


*Image Credits - https://github.com/REhints/BlackHat_2017/blob/master/Betraying the BIOS.pdf

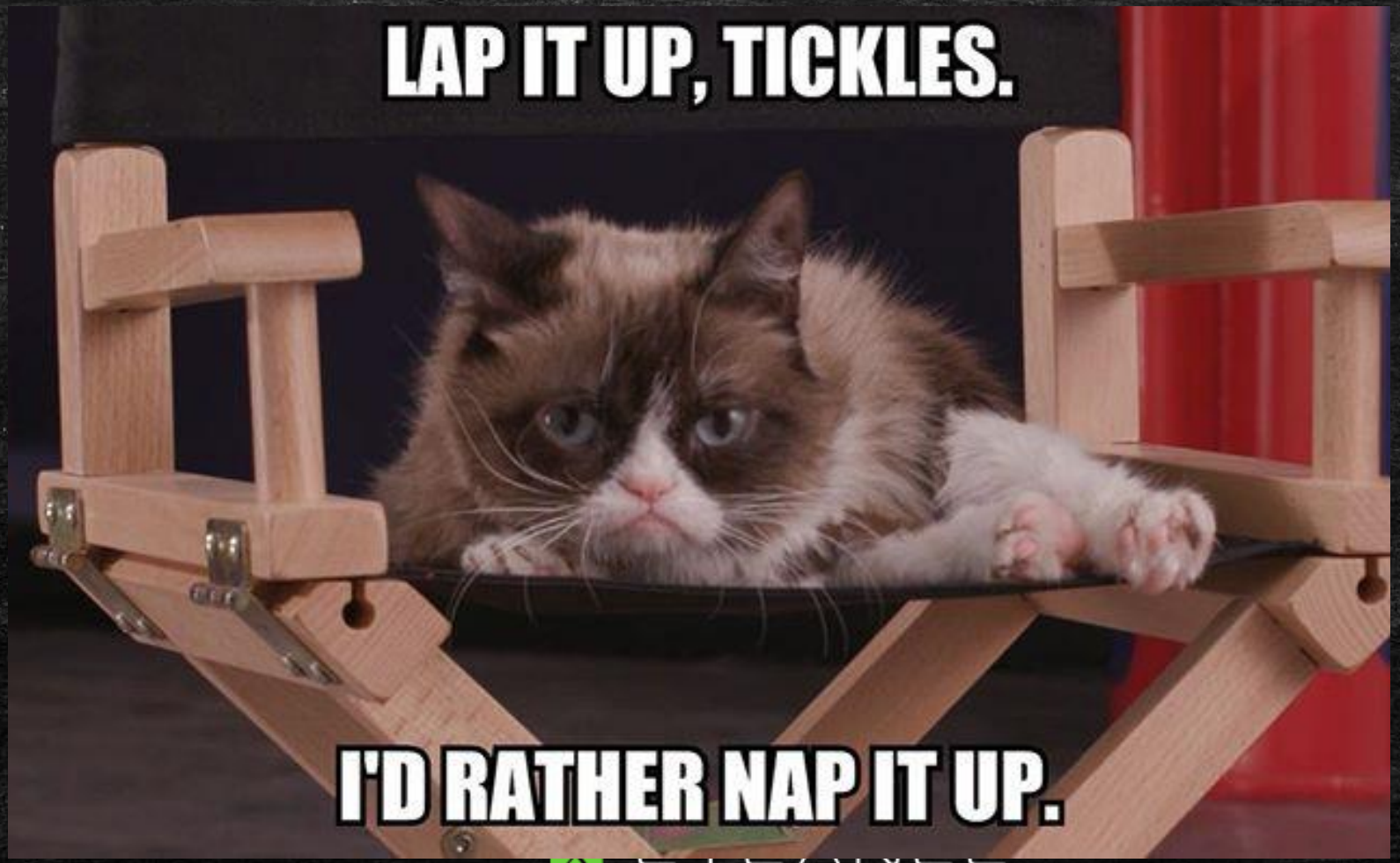
CYLANCE

Recommended flash update method





Who's with me?



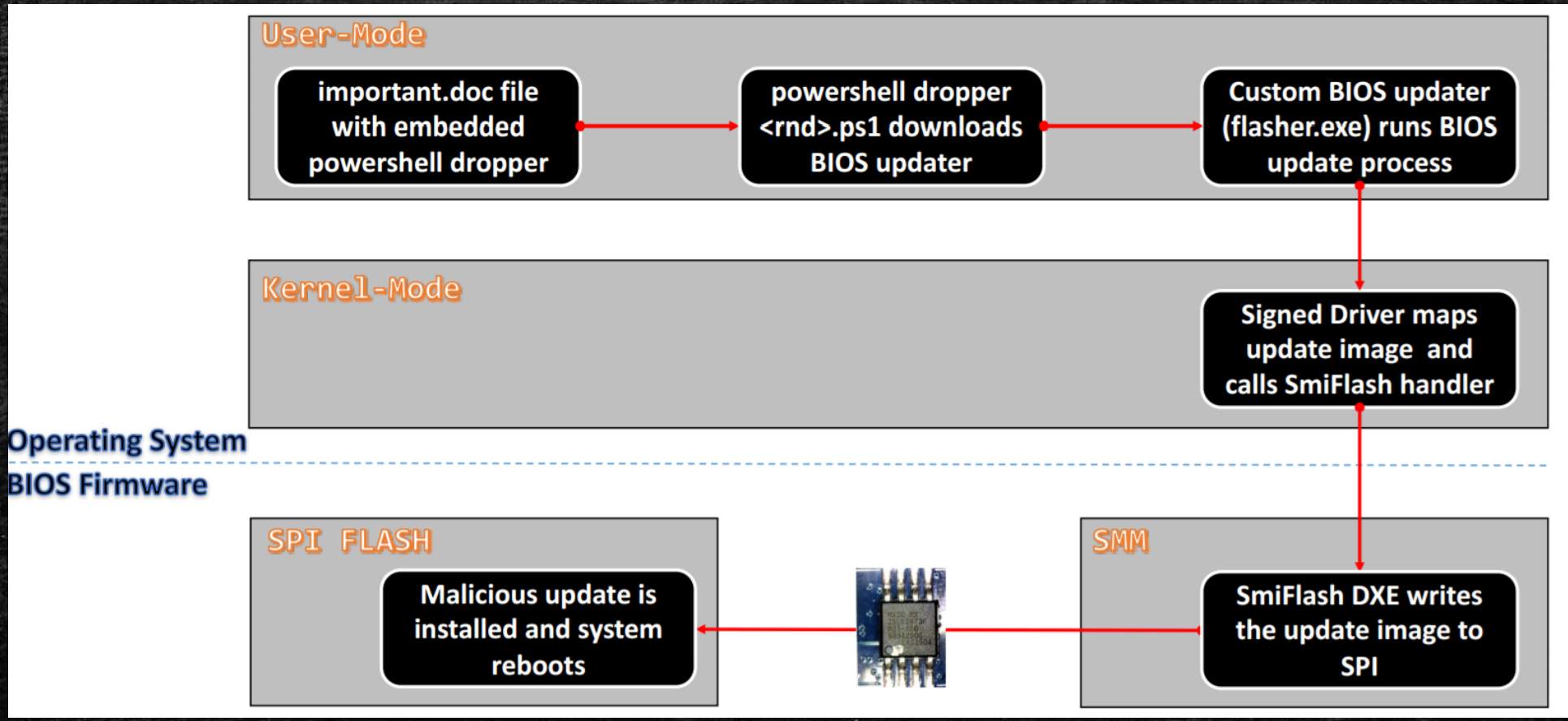
State of Security in various tested platforms.

Vendor Name	BLE	SMM_BWP	PRx	Authenticated Update
ASUS	+	+	-	-
MSI	-	-	-	-
Gigabyte	+	+	-	-
Dell	+	+	-+	+
Lenovo	+	+	RP	+
HP	+	+	RP/WP	+
Intel	+	+	-	+
Apple	-	-	WP	+

```
[x][ ================================================================
[x][ Module: BIOS Interface Lock (including Top Swap Mode)
[x][ ================================================================
[*] BiosInterfaceLockDown (BILD) control = 1
[*] BIOS Top Swap mode is disabled (TSS = 0)
[*] RTC TopSwap control (TS) = 0
[+] PASSED: BIOS Interface is locked (including Top Swap Mode)

[*] running module: chipsec.modules.common.bios_wp
[*] Module path: c:\Chipsec\chipsec\modules\common\bios_wp.pyc
[x][ ================================================================
[x][ Module: BIOS Region Write Protection
[x][ ================================================================
[*] BC = 0x08 << BIOS Control (b:d.f 00:31.0 + 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
[-] BIOS region write protection is disabled!

[*] BIOS Region: Base = 0x00A00000, Limit = 0x00FFFFFF
SPI Protected Ranges
------------------------------------------------------------
PRx (offset) | Value    | Base     | Limit    | WP? | RP?

PR0 (74)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR1 (78)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR2 (7C)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR3 (80)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR4 (84)     | 00000000 | 00000000 | 00000000 | 0   | 0

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

CYLANCE

A crafted attack in our labs



CYLANCE

Demo

UEFI Scanner Components

- Pre-Infection checks
  - How many bug classes can it address?
  - Checks for incorrect configuration, variables?
  - Any dynamic runtime checks?
    - MMIOBAR
    - SMM protections and vulnerabilities etc.

- Post infection checks (UEFI static scanner)
  - Scan enterprise networks for known malicious or anomalous EFI binaries.
  - This is in the realm of traditional AV.

Uefi static scanner overview

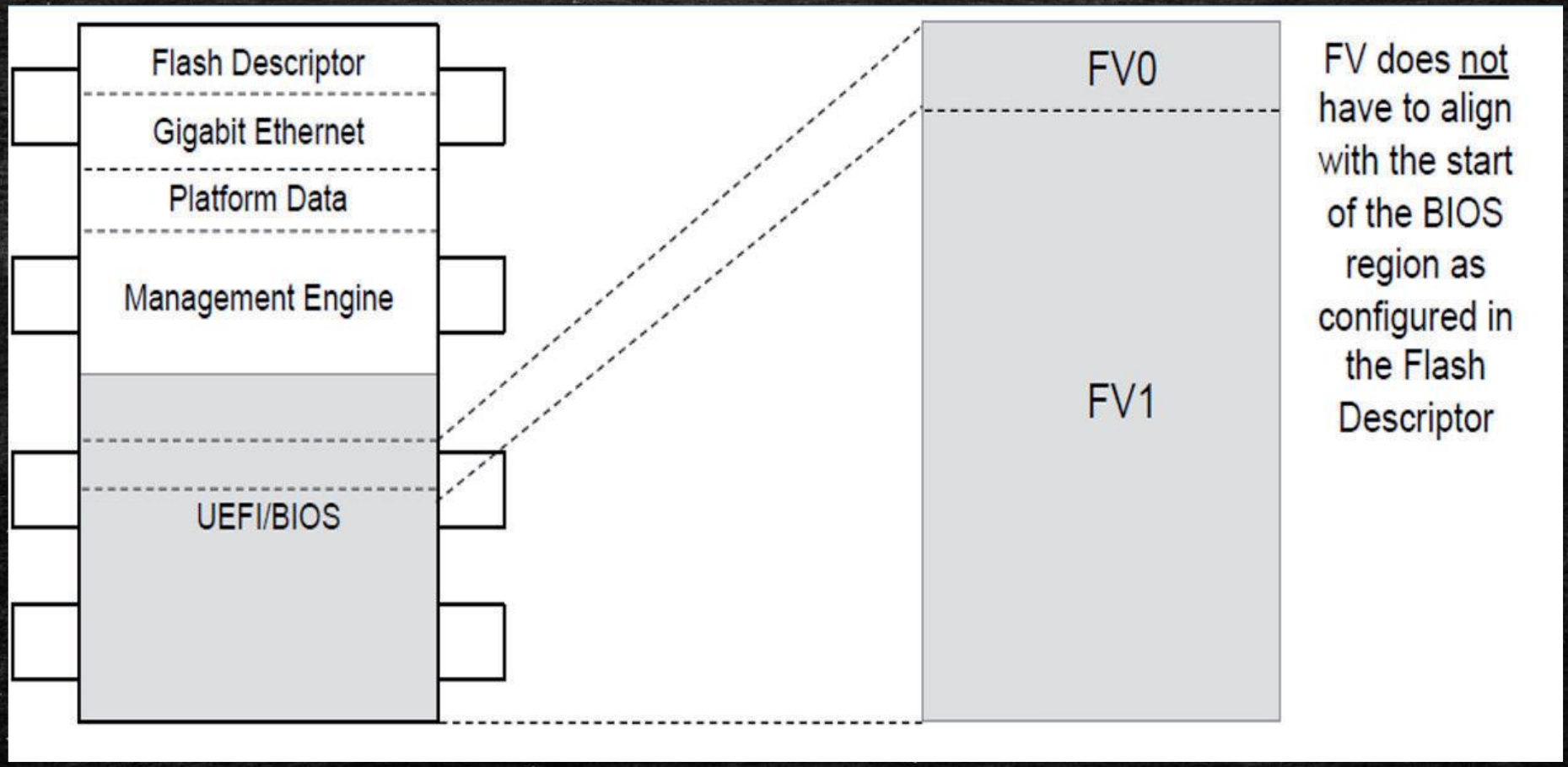
1. Extract UEFI BIOS from SPI flash storage.
2. Parse UEFI firmware file system.
3. Extract firmware volumes.
4. Extract file sections and categorize them.
5. Recursively extract UEFI drivers and applications from sections.

CYLANCE

SPI flash extraction

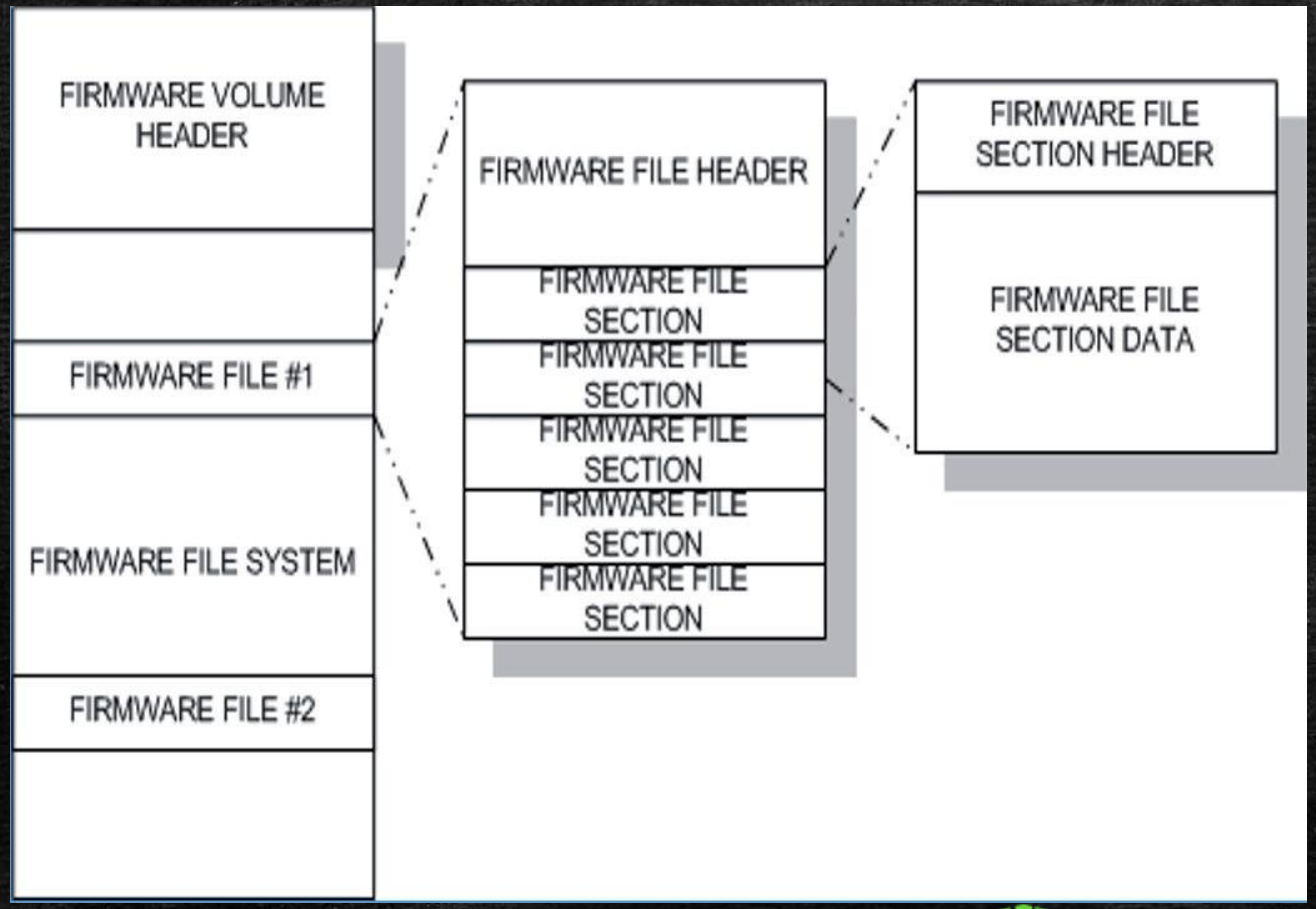
1. Determine the platform DeviceID and VendorID the application is running on by reading PCI Configuration via CONFIG_ADDRESS 0x0CF8 followed by STI instruction.
2. To get the VendorID/DeviceID, we need to pass the bus ,device, function and offset parameter values as 0. (The read_pci_reg function can be used as shown in Chipsec code)
3. Find the base address register of SPI using the same function but passing the correct (depends on the platform) bus, device, function and register parameters.
4. Set up hardware sequencing flash control registers(HSFC) with ‘Flash Cycle’ as Read for the CPU to read the SPI flash and copy it via the SPI memory mapped configuration register FDATA0.
5. The maximum size of data returned in the memory location pointed to by the FDATA0 register is 64 bytes, so the SPI flash dump would get saved in chunks of 64 bytes.

FFS (Firmware file system)



CYLANCE

Firmware volume layout

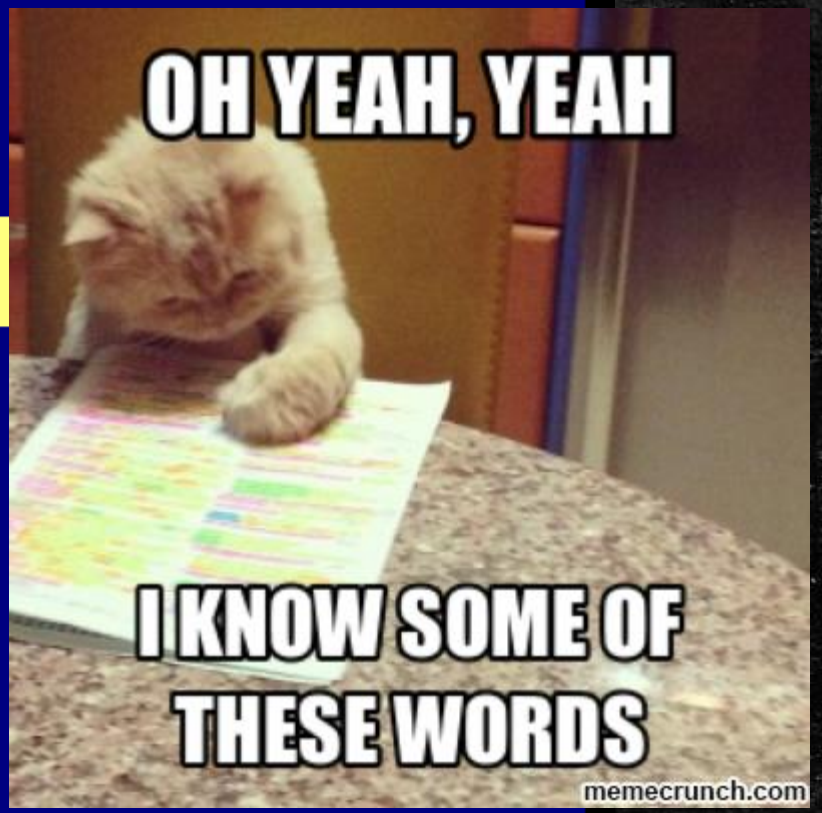



```
typedef struct {
    UINT8                 ZeroVector[16];
    EFI_GUID              FileSystemGuid;
    UINT64                FvLength;
    UINT32                Signature;
    EFI_FV_ATTRIBUTES     Attributes;
    UINT16                HeaderLength;
    UINT16                Checksum;
    UINT16                ExtHeaderOffset;
    UINT8                 Reserved[1];
    UINT8                 Revision;
    EFI_FV_BLOCK_MAP      BlockMap[];
} EFI_FIRMWARE_VOLUME_HEADER;
```

CYLANCE

Typical Firmware volume start

```
00000000:  50 46 53 2E-48 44 52 2E-01 00 00 00-A3 97 C4 00  PFS.HDR.⊖   úù─
00000010:  B0 C2 C6 7E-E3 3F A0 42-A3 16 22 DD-05 17 C1 E8  ░┬╞~π?áBú▬"▌♣↨┴◍
00000020:  01 00 00 00-41 4E 20 20-0A 00 00 00-00 00 00 00  ⊖   AN  ◙
00000030:  00 00 00 00-00 00 00 00-00 00 60 00-00 01 00 00                `   ⊖
00000040:  2C 01 00 00-00 01 00 00-B0 4C 30 08-90 C1 79 7D  ,⊖   ⊖  ░L0▪É┴y}
00000050:  ED 19 89 51-AC F4 25 AD-00 00 00 00-00 00 00 00  ф↓ëQ¼⌠%¡
00000060:  00 00 00 00-00 00 00 00-D9 54 93 7A-68 04 4A 44          ┘Tôzh♦JD
00000070:  81 CE 0B F6-17 D8 90 DF-00 00 02 00-00 00 00 00  ü╬♂÷↨╪É■  ☻
00000080:  5F 46 56 48-FF 8E FF FF-48 00 92 3C-00 00 00 01  _FVH Ä  H Æ<   ⊖
00000090:  20 00 00 00-00 10 00 00-00 00 00 00-00 00 00 00     ►
000000A0:  A3 B9 F5 CE-6D 47 7F 49-9F DC E9 81-43 E0 42 2C  ú╣│╬mG⌂Iƒ▄ΘüCαB,
000000B0:  36 5A 01 00-B8 FF 01 F8-4E 56 41 52-11 0E FF FF  6Z⊖ ╕ ⊖°NVAR◄♫
000000C0:  FF 83 00 53-74 64 44 65-66 61 75 6C-74 73 00 4E   â StdDefaults N
000000D0:  56 41 52 EC-0C FF FF FF-83 00 53 65-74 75 70 00  VAR∞♀   â Setup
000000E0:  01 00 00 20-00 00 00 00-00 01 37 37-00 00 05 64  ⊖        ⊖77  ♣d
000000F0:  00 00 00 02-00 00 01 00-00 00 00 00-00 01 00 00     ☻  ⊖      ⊖
00000100:  01 01 00 00-01 00 00 00-00 00 00 00-00 01 01 00  ⊖⊖  ⊖        ⊖⊖
00000110:  01 01 02 01-00 00 00 02-00 00 01 01-01 00 01 01  ⊖⊖☻⊖   ☻  ⊖⊖⊖ ⊖⊖
00000120:  01 00 01 00-01 00 01 00-00 01 00 00-01 01 01 01  ⊖ ⊖ ⊖ ⊖  ⊖  ⊖⊖⊖⊖
00000130:  01 01 01 01-04 04 04 04-04 04 04 04-00 00 00 00  ⊖⊖⊖⊖♦♦♦♦♦♦♦♦
00000140:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000150:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000160:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```



CYLANCE

Firmware file and sections.

```
typedef struct _EFI_FFS_FILE_HEADER {
EFI_GUID Name;
EFI_FFS_INTEGRITY_CHECK IntegrityCheck;
EFI_FV_FILETYPE Type;
EFI_FFS_FILE_ATTRIBUTES Attributes;
UINT8 Size[3];
EFI_FFS_FILE_STATE State;
} EFI_FFS_FILE_HEADER,
*PEFI_FFS_FILE_HEADER;

typedef struct _EFI_FFS_FILE_HEADER2 {
EFI_GUID Name;
EFI_FFS_INTEGRITY_CHECK IntegrityCheck;
EFI_FV_FILETYPE Type;
EFI_FFS_FILE_ATTRIBUTES Attributes;
UINT8 Size[3];
EFI_FFS_FILE_STATE State;
UINT32 ExtendedSize;}
EFI_FFS_FILE_HEADER2,
*PEFI_FFS_FILE_HEADER2;
```

Encapsulation Section Types

SECTION_COMPRESSION 0x1

SECTION_GUID_DEFINED 0x2

Leaf Section Types

SECTION_PE32 0x10

SECTION_PIC 0x11

SECTION_TE 0x12

SECTION_DXE_DEPEX 0x13

SECTION_VERSION 0x14

SECTION_USER_INTERFACE 0x15

SECTION_COMPATIBILITY16* 0x16 SECTION_FIRMWARE_VOLUME_IMAGE 0x17

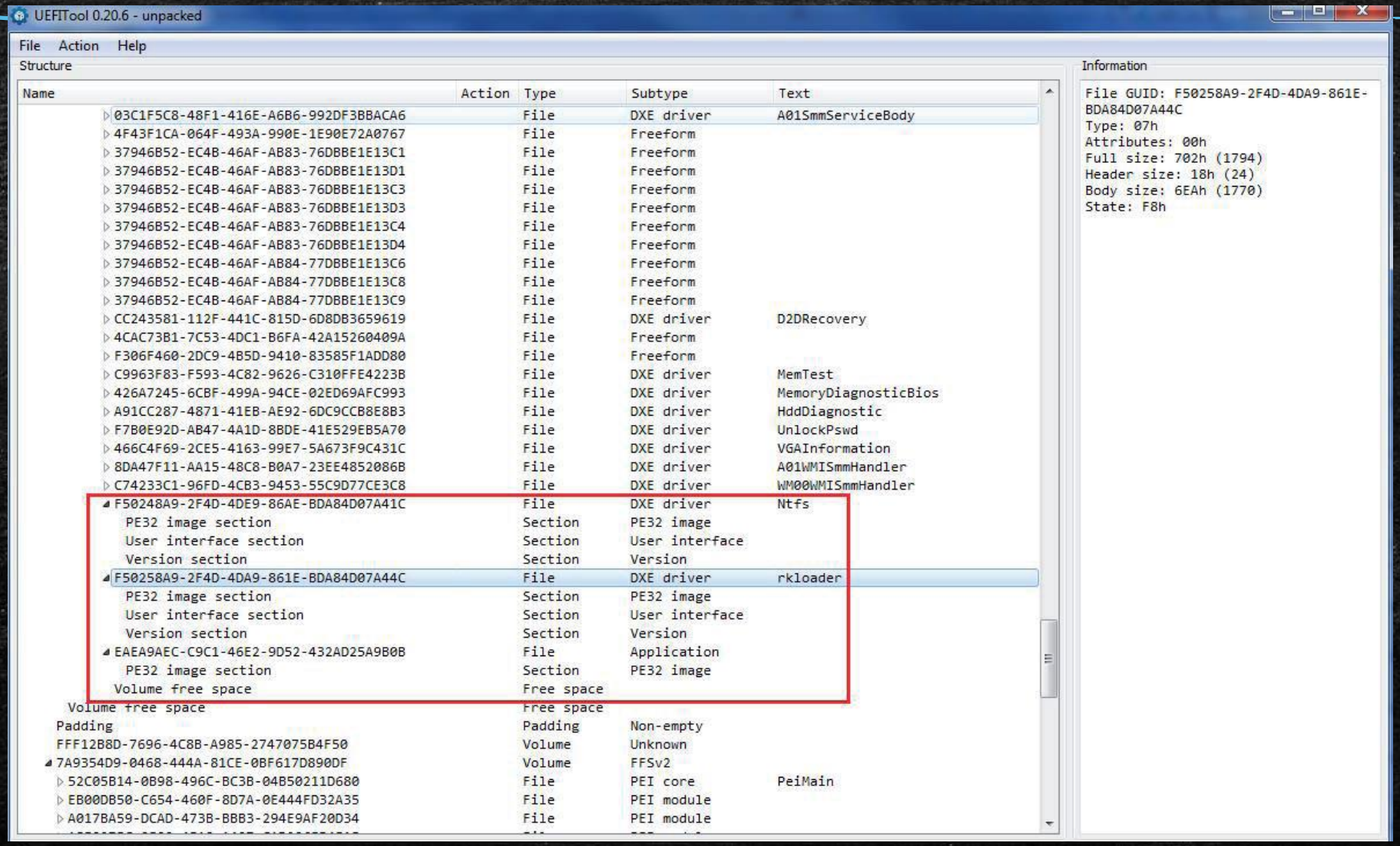
SECTION_FREEFORM_SUBTYPE_GUID 0x18

SECTION_RAW 0x19

SECTION_PEI_DEPEX 0x1b

**(Compatibility with legacy BIOS. Some UEFI-based PCs contain a Compatibility Support Module (CSM) that emulates earlier BIOS, providing more flexibility and compatibility for end users. To use the CSM, Secure Boot must be disabled)*

UEFI tool and hacking team bios implant



Cost of building a working UEFI bypass.

- Depends how secure the UEFI implementation is, how many check boxes are checked.
- In a poorly safeguarded system approx. cost could run in thousands to tens of thousands of USD to build a full stack attack with SPI flash persistence.
- In a system where we need to find zero days in each layer of attack it might run in millions.
- Unfortunately too many systems are unguarded.

Analysis done by Firmware experts – 44Con

- **~3,490 update images** corresponding to **~570 models** from 9 manufacturers were found **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 have had basic firmware protections for a while. Still older systems are often forgotten (older than 2016)
- Some manufacturers started recently (Ivy Bridge or Skylake systems)
- Some manufacturers yet to start protecting firmware on their systems

https://github.com/abazhaniuk/Publications/blob/master/2017/44CON_2017/Bulygin_Bazhaniuk_44con.pdf

Conclusion

- Understanding UEFI threat landscape and current security solutions is a bit of a learning curve.
- Cooking up an attack on one of the current consumer devices is within reach of many established malware groups.
- UEFI scanning is probably the quickest thing that AV-industry can contribute their expertise in.
- Firmware attack surface scanning.
  - Any dynamic checks and misconfiguration checks are very useful to know , but how will we act on them?

CYLANCE

Thanks for listening

- Email : akapoor@Cylance.com

- References for this presentation can be found in the VB paper.

CYLANCE