Platform Security Summit 2019
Oct 1-3, 2019 · Redmond, WA

The Advanced Threats Evolution: REsearchers Arm Race

@matrosov

Chief Offensive Security REsearch at 

Former Principal Security Researcher @Cylance @Intel @ESET

Doing Security REsearch since 1997

How I got the idea to talk about evolution of persistence techniques?

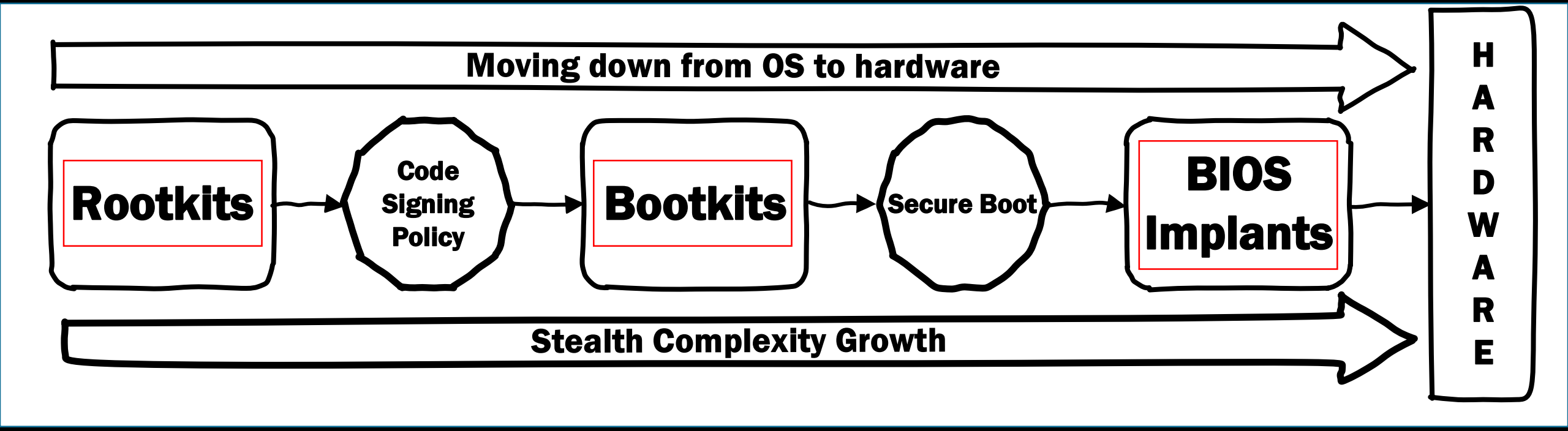
Security Industry Visibility Point

Modern Persistence Techniques

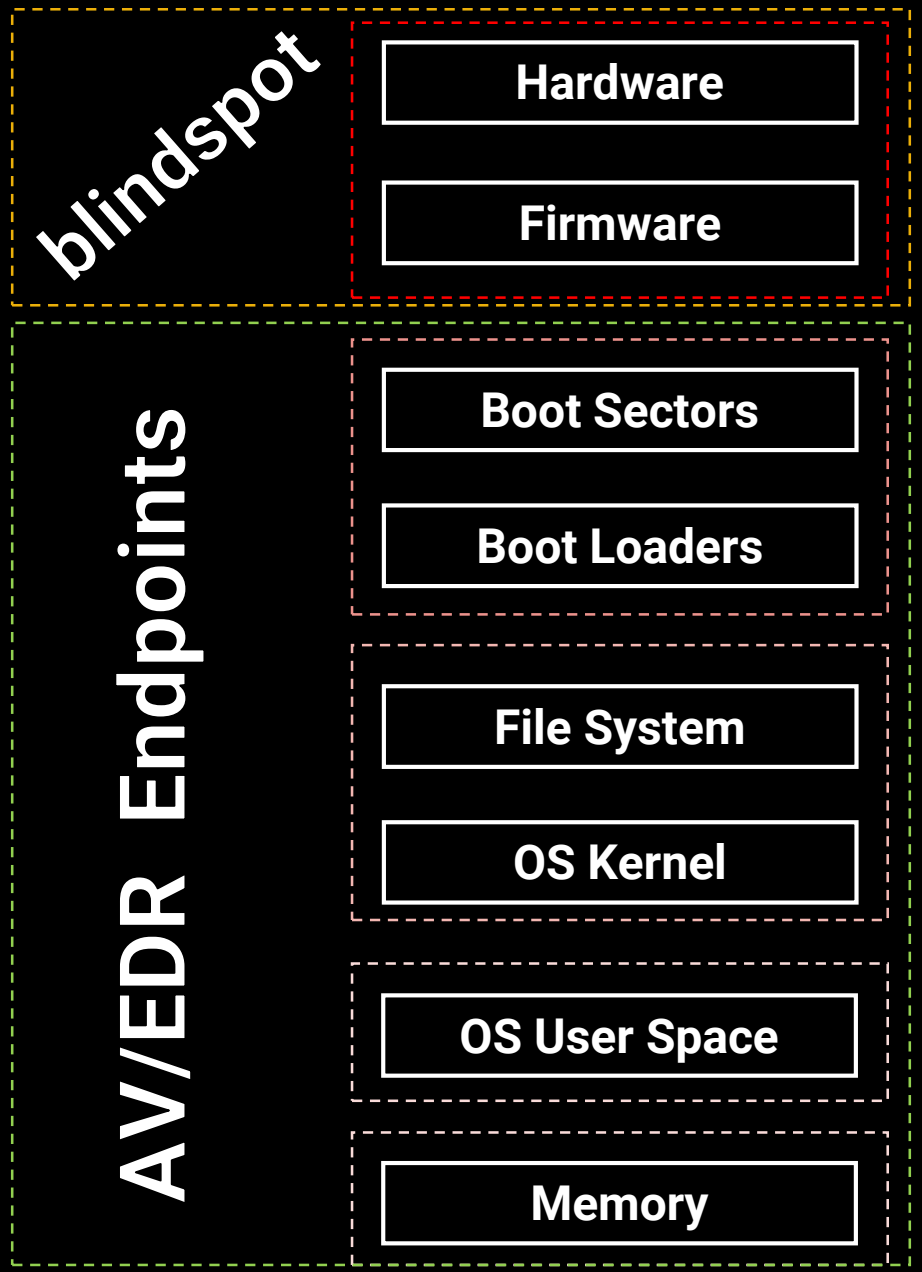
Evolution of persistence techniques

Mitigations against malware persistence techniques are raising the bar of complexity, but the bar is always covered only the most common ones.

More mitigations, more persistence complexity



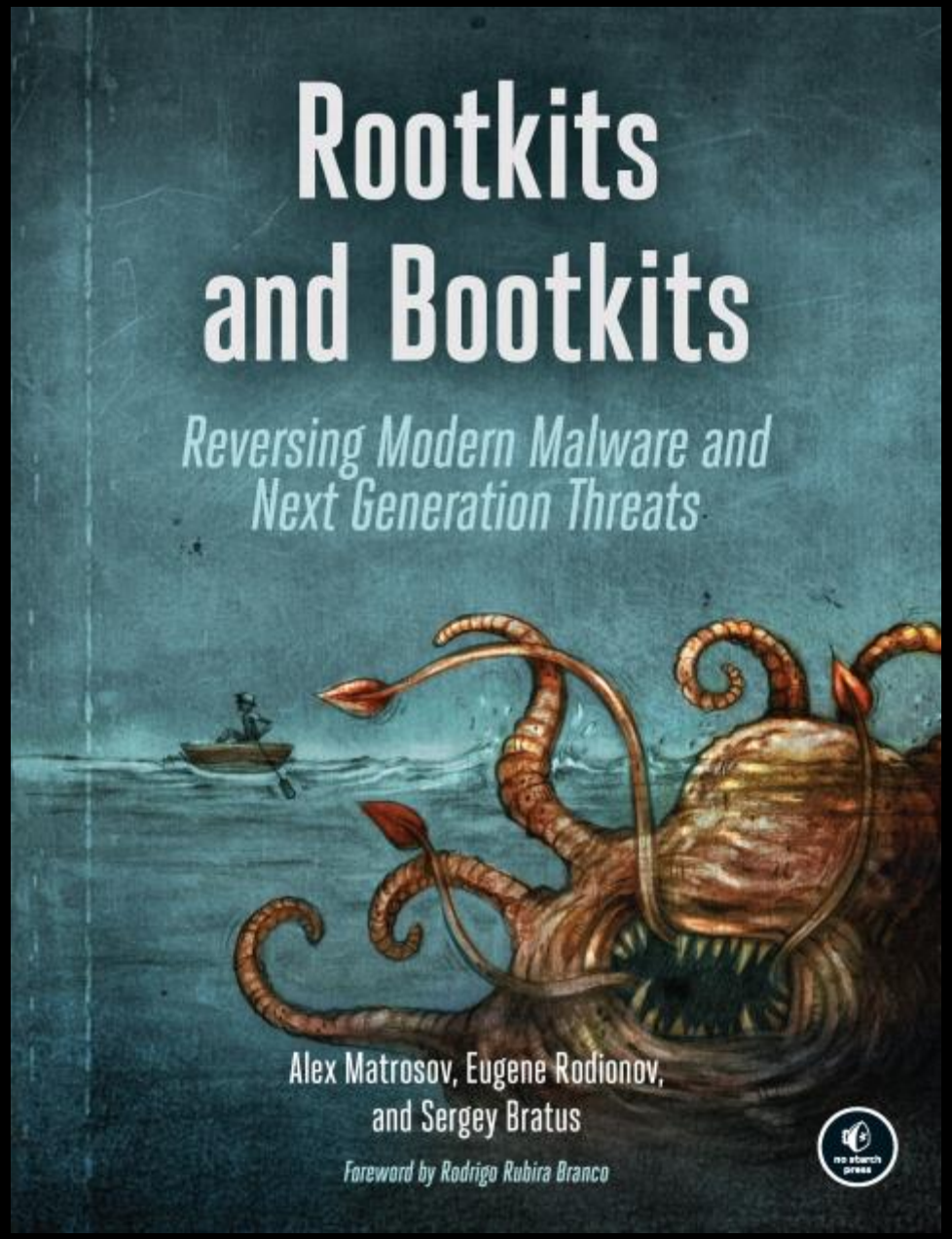
Types of Persistence



HW/OS change Persistence

Reboot/Shutdown Persistence

Sleep/Hibernate Persistence


Rootkits
and Bootkits
Reversing Modern Malware and
Next Generation Threats
Alex Matrosov, Eugene Rodionov,
and Sergey Bratus
Foreword by Rodrigo Rubira Branco
no starch
press

Rootkits

Golden Age of Rootkits (2006 - 2012)
Bootkits (2011 - 2015)

- ❑ Cybercrime actors persistence goals
  - ❑ Long term infection rate
    - ❑ Spam bots
    - ❑ DDoS bots

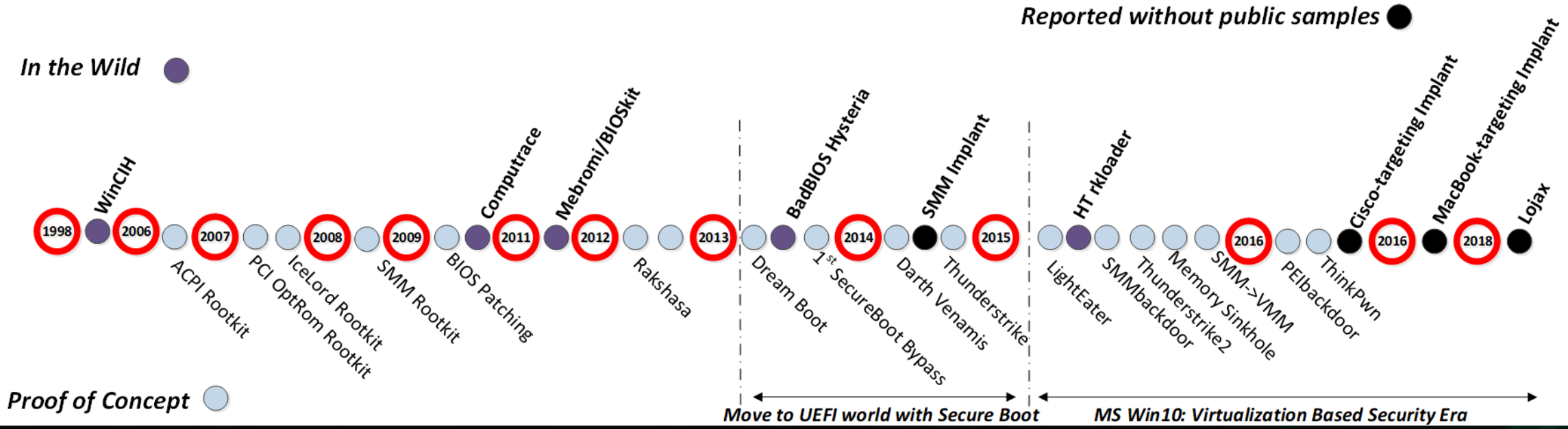
- ❑ State-sponsored actors persistence goals
  - ❑ Long term covered operations
    - ❑ Espionage, gathering data
    - ❑ Attack specific target in specific time

https://github.com/bootkitsbook/rootkits

https://github.com/bootkitsbook/bootkits

Golden Age of Firmware/Hardware Implants is happening right now!

The retrospective of incidents with BIOS implants



Number of incidents with firmware increases every year

PC REVIEWS BEST PICKS HOW-TO NEWS SMART HOME BUSINESS SHOP

#Computex2019 #MotoZ4 #EchoShow5 #iPodTouch #PrimeDay

Business | IT Watch

Invisible Malware Is Here and Your Security Software Can't Catch It

Sophisticated attackers are now using "invisible malware," a new form of attack that your firewalls can't stop and your anti-malware software can't find nor remove. Here are steps you can take right now to protect your servers and network.

By Wayne Rash April 25, 2019 7:00AM EST

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

NO EASY CURE —

Supermicro hardware weaknesses let researchers backdoor an IBM cloud server

Other providers of bare-metal cloud computing might also be vulnerable to BMC hack.

DAN GOODIN - 2/26/2019, 5:00 AM

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

HIT THE ROAD, LOJAX —

First UEFI malware discovered in wild is laptop security software hijacked by Russians

"LoJax" repurposed LoJack anti-theft agent as rootkit that could survive OS re-installs.

SEAN GALLAGHER - 10/2/2018, 5:33 AM

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

ALIVE AND WELL —

Eight months after discovery, unkillable LoJax rootkit campaign remains active

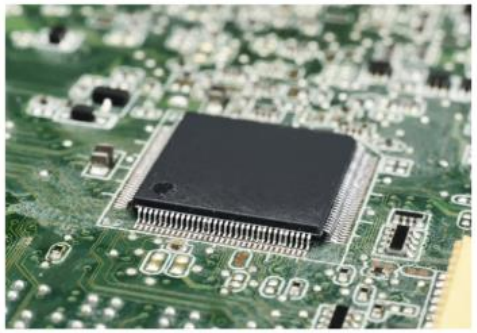
Control servers for Fancy Bear's UEFI-burrowing malware still responding to pings.

DAN GOODIN - 1/16/2019, 6:00 AM

threatpost Cloud Security | Malware | Vulnerabilities | Privacy | HackerOne Spotlight

Locky Variant Zepto Debuts with Big Spam Push — Chinese Ad Firm Raking in $3

Scope of ThinkPwn UEFI Zero Day Expands



The scope of the ThinkPwn UEFI vulnerability disclosed last week has grown past Lenovo and HP laptop firmware to motherboards sold by Gigabyte.

A serious hardware vulnerability, thought to be confined to UEFI drivers in Lenovo and HP laptops, has also been found in firmware running on motherboards sold by Gigabyte.

Some Motherboards Plagued by BIOS Firmware Implementation Flaws

By Catalin Cimpanu October 7, 2017 03:00 AM 0

Alex Matrosov, a security researcher for Cylance, has discovered several flaws in how some motherboard vendors implemented Intel's UEFI BIOS firmware into their products.

These flaws allow an attacker to bypass BIOS firmware protections, such as Intel Boot Guard and Intel BIOS Guard, to disable and alter UEFI BIOS firmware, such as placing a rootkit.

Matrosov presented his findings at the Black Hat USA 2017 security conference held in Las Vegas in August.

Bloomberg Businessweek

■ October 4, 2018, 2:00 AM PDT

The Big Hack: The Software Side of China's Supply Chain Attack

- It wasn't just hardware. An online portal for firmware updates hid and distributed malware.

Tech buyer rights raised in Cisco vulnerability

The seriousness of the Cisco vulnerability, Thrangycat, raises the question of tech buyers' rights when dealing with such a serious flaw in a vendor's hardware.

Antone Gonsalves Director of News 31 May 2019

A network vulnerability as hard to fix as the recently disclosed Thrangrycat flaw in 150 varieties of Cisco switches and routers raises the question of a tech buyer's rights when repairing a

"Firmware attacks are hard to detect by current antivirus and other security software because firmware resides at architectural levels of the device that are not usually accessible to current tools"

OS Kernel-Mode (Ring 0)

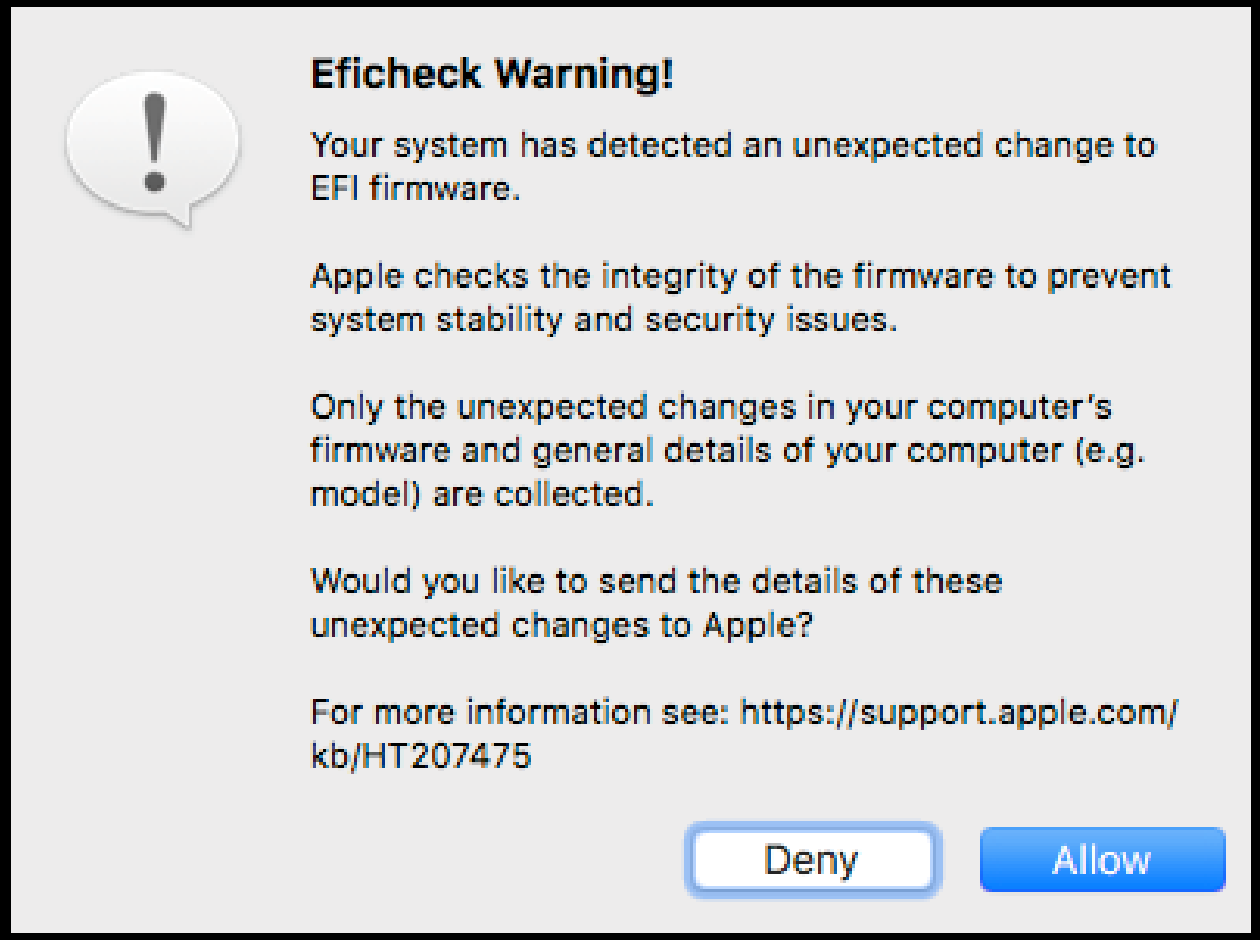
- **Mitigations:** PatchGuard, Code Signing Policy
- **Prevention:** AV HIPS, EDR

Boot code (MBR/VBR)

- **Mitigations:** Secure/Measured Boot, Boot Guard
- **Prevention:** AV HIPS, EDR

BIOS/UEFI Firmware SMM (Ring -2)

- **Mitigations: limited** (Intel BIOS/Boot Guard? STM?)
- **Prevention: not exist**



Integrity check works only with fixed dataset!

In large-scale integrity, pre-check/post-check has problems to detect supply chain attacks.

Scan completed: Threats found

Threats found: 2 (Cleaned: 0)
Detection Engine used: 18088 (20180921)

Scan Log

Version of detection engine: 18088 (20180921)

Date: 21-Sep-18 Time: 10:19:48 AM

Scanned disks, folders and files: Boot sectors/UEFI

\\Uefi Partition » UEFI » uefi:\\Volume 1\DXE Core {4A538818-5AE0-4EB2-B2EB-488B23657022}\Unnamed partition\Volume 1\

Number of scanned objects: 474

Number of threats found: 2

Number of cleaned objects: 0

Time of completion: 10:30:39 AM Total scanning time: 651 sec (00:10:51)

Dump and scan approach for UEFI images from OS has visibility limitations.

Most of AV endpoints just detect known UEFI threats and identify unusual DXE/PEI images by simple heuristics.

BIOS challenge for AV Endpoint solutions

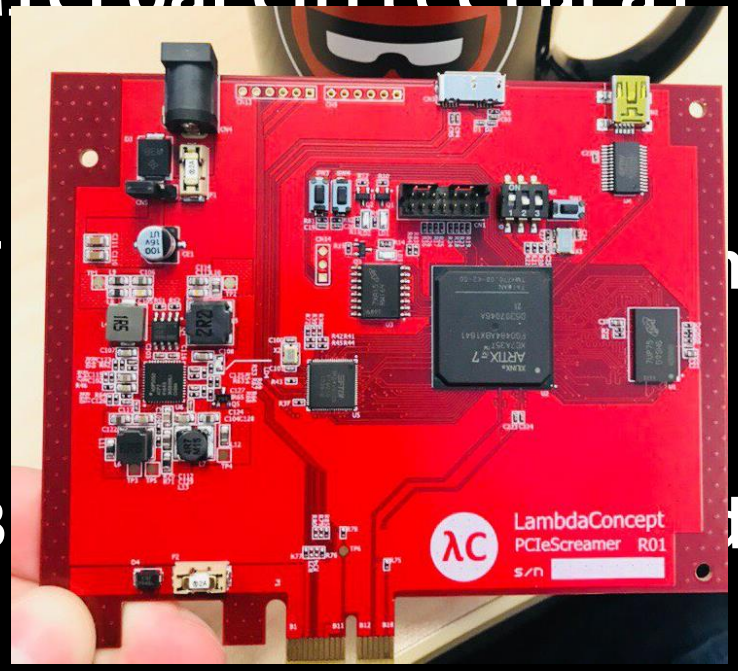
- ❑ Limits of gathering information after OS boot
  - ✓ Everything can be a fake after OS boot
  - ✓ BIOS Rootkit/Implant can disable any AV endpoint solution
  - ✓ All the BIOS updates can be reinfected or just blocked
- ❑ No trusted path between AV and UEFI Firmware
  - ✓ Only cooperation between Firmware and AV can help to protect
  - ✓ Hashing UEFI drivers or SPI flash dump don’t guarantee much
  - ✓ Who guarantee BIOS updates from hardware vendor not infected?
- ❑ Blind spot of supply chain attacks
  - ✓ Physical access not in scope of the most of security features

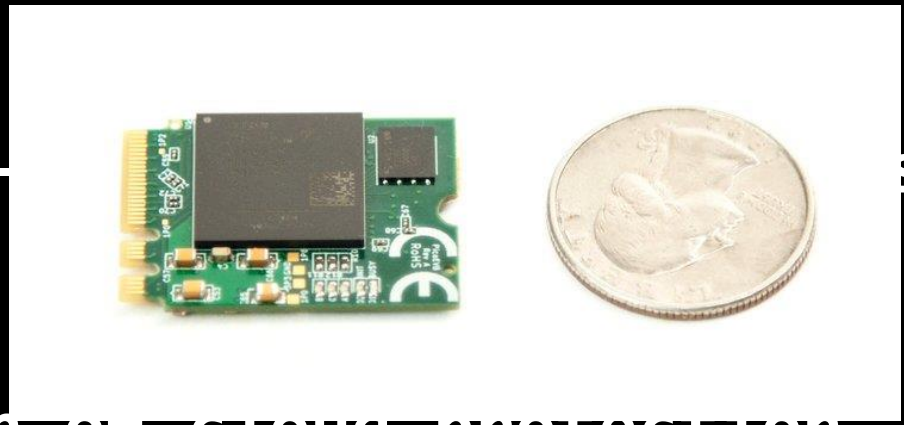
Who watch the watchers?

- ❑ Microarchitectural attacks – Hardware is new Software!
- ❑ Firmware is everywhere – Who cares about its security?!
- ❑ BIOS became a foundation of cloud protection – broken?
- ❑ Supply Chain attacks – become mainstream?

Who watch the watchers?

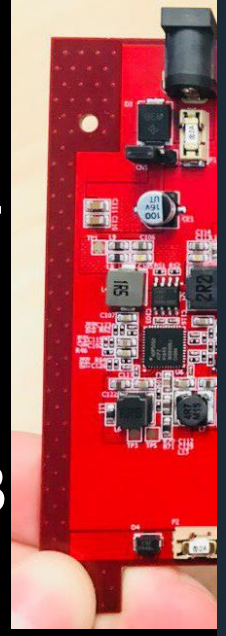
- ❑ Microarchitectural attacks – Hardware is new Software!
- ❑ F…here – …security?!
- ❑ B…dation of cloud protection – broken?
- ❑ Supply Chain attacks – become mainstream?





Who wa

- ❑ Microa… … Software!
- ❑ F… … security?!
- ❑ B… … – broken?
- ❑ Supply …



Dmytro Oleksiuk
@d_olex

Following

Rogue PCI-E/FireWire/Thunderbolt/etc. device can exploit platform firmware vulns to execute arbitrary System Management Mode code [1/x]

```
:/vagrant_home/Documents/Xilinx/s6_pcie_uart/python$ sudo ./uefi_infect.
oor_X64.efi
e was not initialized by the host
e was not initialized by the host
with target is up
ewhere around 0x8b800000
 at 0x88dd0000
TABLE is at 0x8b295f18
RVICES is at 0x83808d60
RVICES.LocateProtocol() address is 0x83810b88
size is 0x15c0
RVA is 0x538
age driver at 0x10000...
rotocol(): 0x83810b88 -> 0x00010538

was planted, waiting for SMM exploit...
gement Mode payload was executed, collecting SMRAM dump...

 4d 53 33 5f 36 34 0c 1b 7e 8b 00 00 00 00 | SMMS3.64.....
 7b 8b 00 00 00 00 00 80 00 00 00 00 00 00 | ..............
 00 80 00 00 00 00 00 30 7c 8b 00 00 00 00 | .......0.....
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ..............
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ..............
 00 00 00 00 00 00 00 00 30 a7 7f 8b 00 00 | ........0.....
 41 3f e1 6a 2b 1c e2 de 43 1f e3 4a 29 3c | .A..j....C..J..
 63 1f c3 4a 09 3c 42 de e3 1f 43 4a 89 3c | ..c..J..B...CJ..
 0f 37 af 62 65 14 aa f6 0b 37 ab 62 61 14 | ...7.be....7.ba.
 2b 37 8b 62 41 14 0a f6 ab 37 0b 62 c1 14 | ...7.bA....7.b..
 ab 33 0b 66 c1 10 0b d2 aa 13 0a 46 c0 30 | ...3.f.......F.0
 ba 53 1a 06 d0 70 5b 92 fa 53 5a 06 90 70 | ...S...p...SZ..p
 f4 14 bb fc 86 54 8f ae f0 14 bf fc 82 54 | .....T.......T
 f0 14 bf fc 82 54 8f ae f0 14 bf fc 82 54 | .....T.......T
 f0 16 bf fe 82 56 8e ac f1 16 be fe 83 56 | .....V.......V
 e1 56 ae be 93 16 1e ec 61 56 2e be 13 16 | ...V.....aV....
```

```
c(320) : ******************************
c(321) :
c(322) :   UEFI backdoor loaded
c(323) :
c(324) : ******************************
c(327) : Image address is 0x10000
c(341) : Resident code base address is 0x8a1a7000
c(193) : BackdoorEntryResident()
c(207) : SMM communication protocol is at 0x8b2ad300
c(216) : Buffer for SMM communicate call is allocate
c(237) : Communicate() returned status 0xe, data siz
c(243) : Exploitation success!
```

Intel® N

7:41 PM - 21 Jun 2017

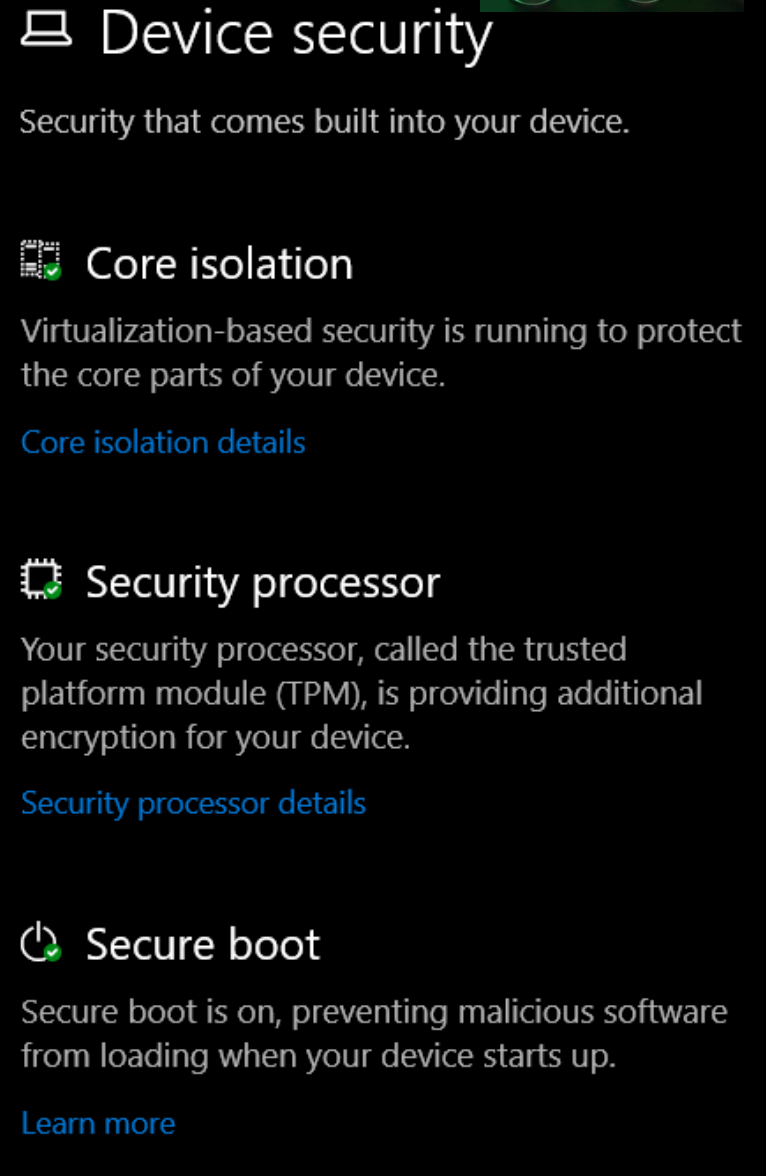
Win10 has active security measurement inside BIOS

❑ MS HW Security Testability Specification (HSTI)

- **HstiSiliconDxe –** Secure Boot, Signed Updates …
- **HstiPlatformDxe –** DMA, Rollback Protection …
- **HstiResultDxe –** store results in locked EFI variable

❑ MS Device Guard has direct FW feedback

- **DeviceGuardDxe –** runtime check for security features dependencies is enabled correctly in firmware



But Gigabyte/ASUS/MSI/ASROCK/Samsung don't care about security! 😈

```
[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
```

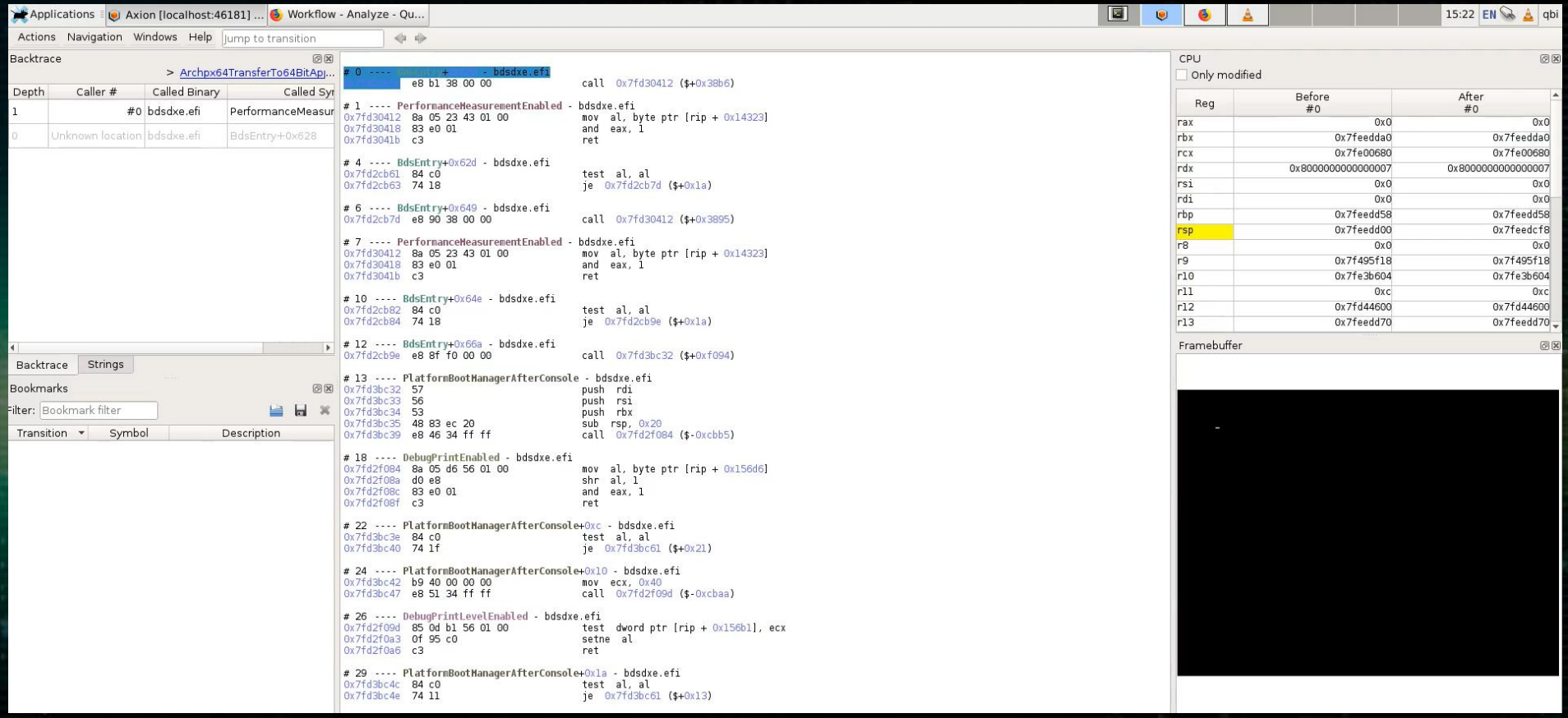
Tools limitations for UEFI firmware RE

- Full system simulation (like Simics) don’t provide hardware-vendor specific environment and EFI protocols which is create a lot of limitations.

- UEFI emulation (like QEMU) to execute specific DXE driver even with multiple stubs will have very limited code coverage. In some cases, it will be helpful but in most of them just a waste of time.

- Hardware-level debugging (over Intel DCI) isn’t available on the most of enterprise platforms (DCI unlock sometimes possible over vulnerabilities). Executing DXE driver on not native hardware create the same limitations as simulation or emulation.

Tools limitations for UEFI firmware RE



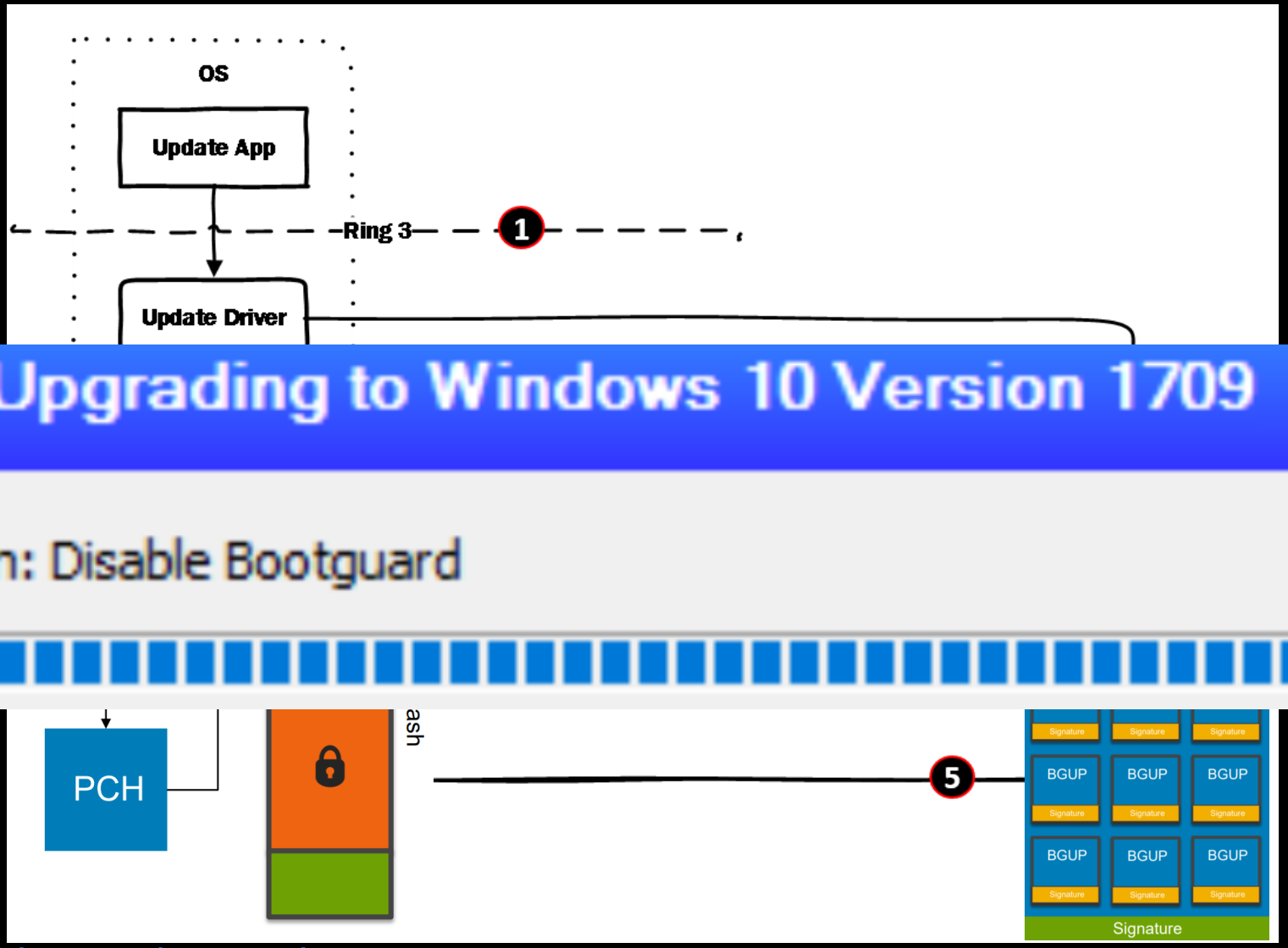
https://blog.tetrane.com/2019/From-UEFI-to-Windows-Boot.html

You can also debug the UEFI firmware part of the boot process or even custom UEFI modules with source level debugging. Please check our XNU kernel debugging howto for more details on this feature.

https://hex-rays.com/products/ida/7.3/index.shtml

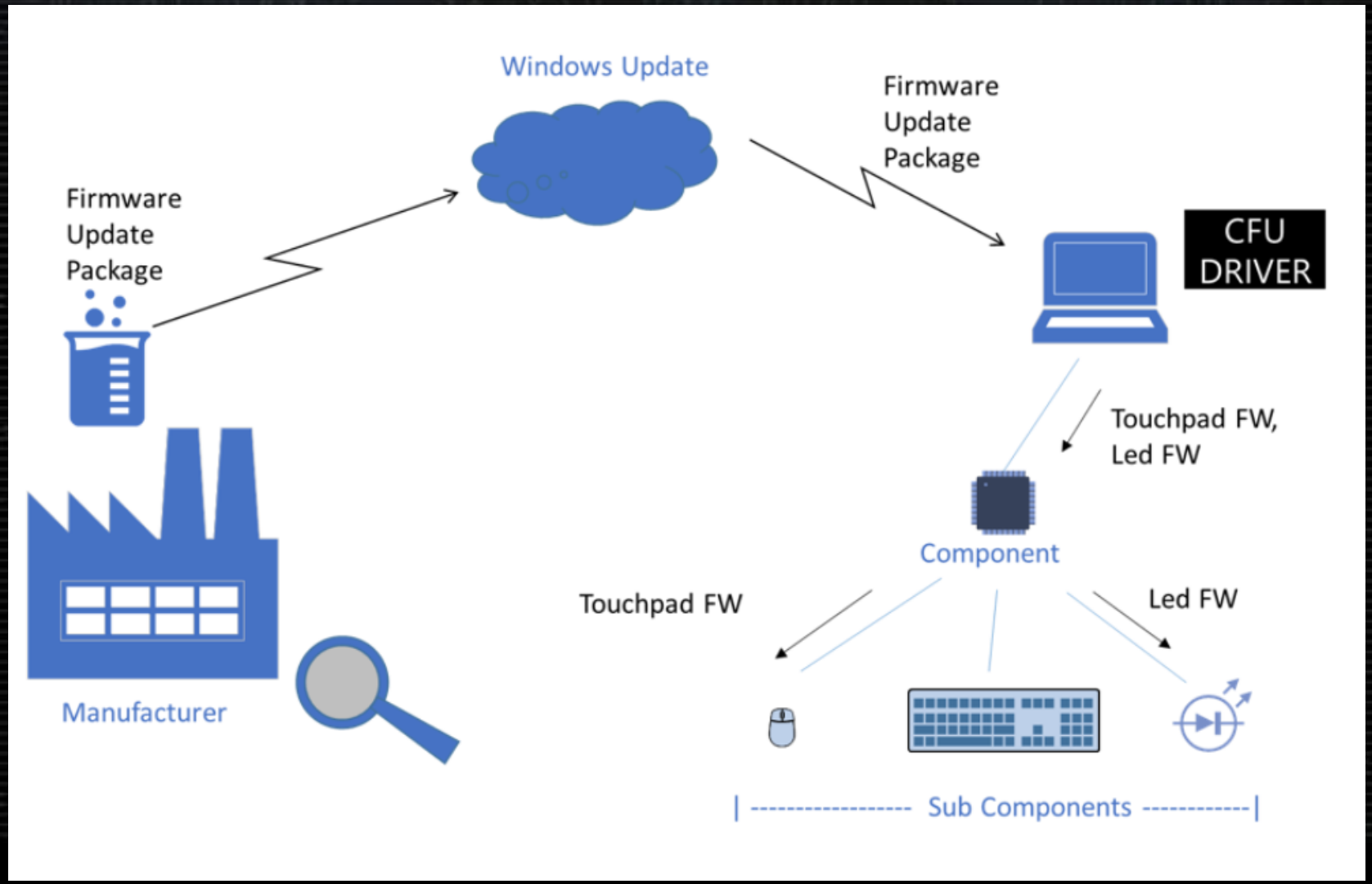

BRAVE
NEW
WORLD

UEFI update process chaos: too many tools, too many problems



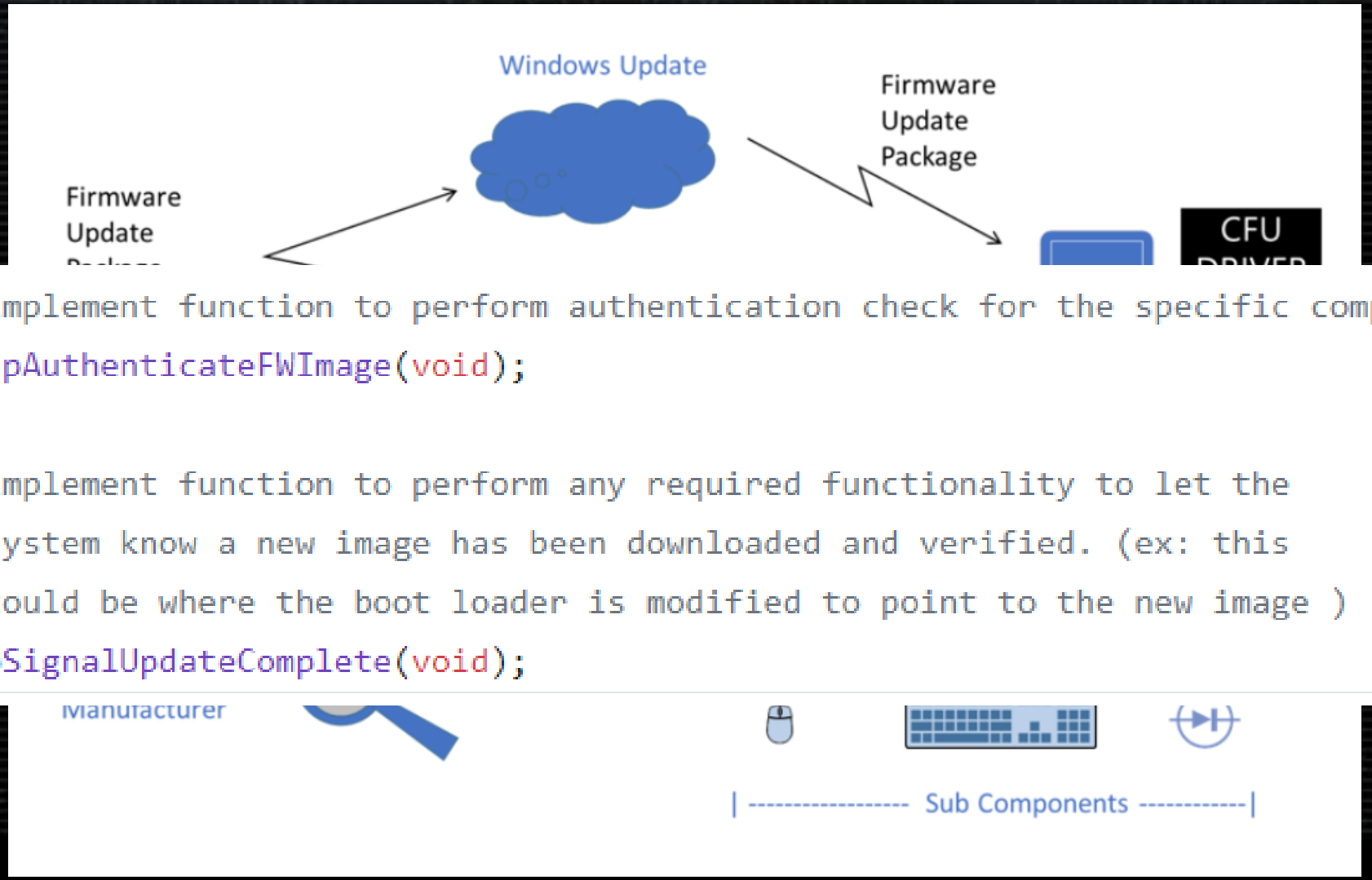
https://embedi.org/blog/nuclear-explotion/
https://medium.com/@matrosov/dangerous-update-tools-c246f7299459
https://www.blackhat.com/us-19/briefings/schedule/#breaking-through-another-side-bypassing-firmware-security-boundaries-from-embedded-controller-15902

MS Component Firmware Update



https://github.com/Microsoft/CFU

MS Component Firmware Update

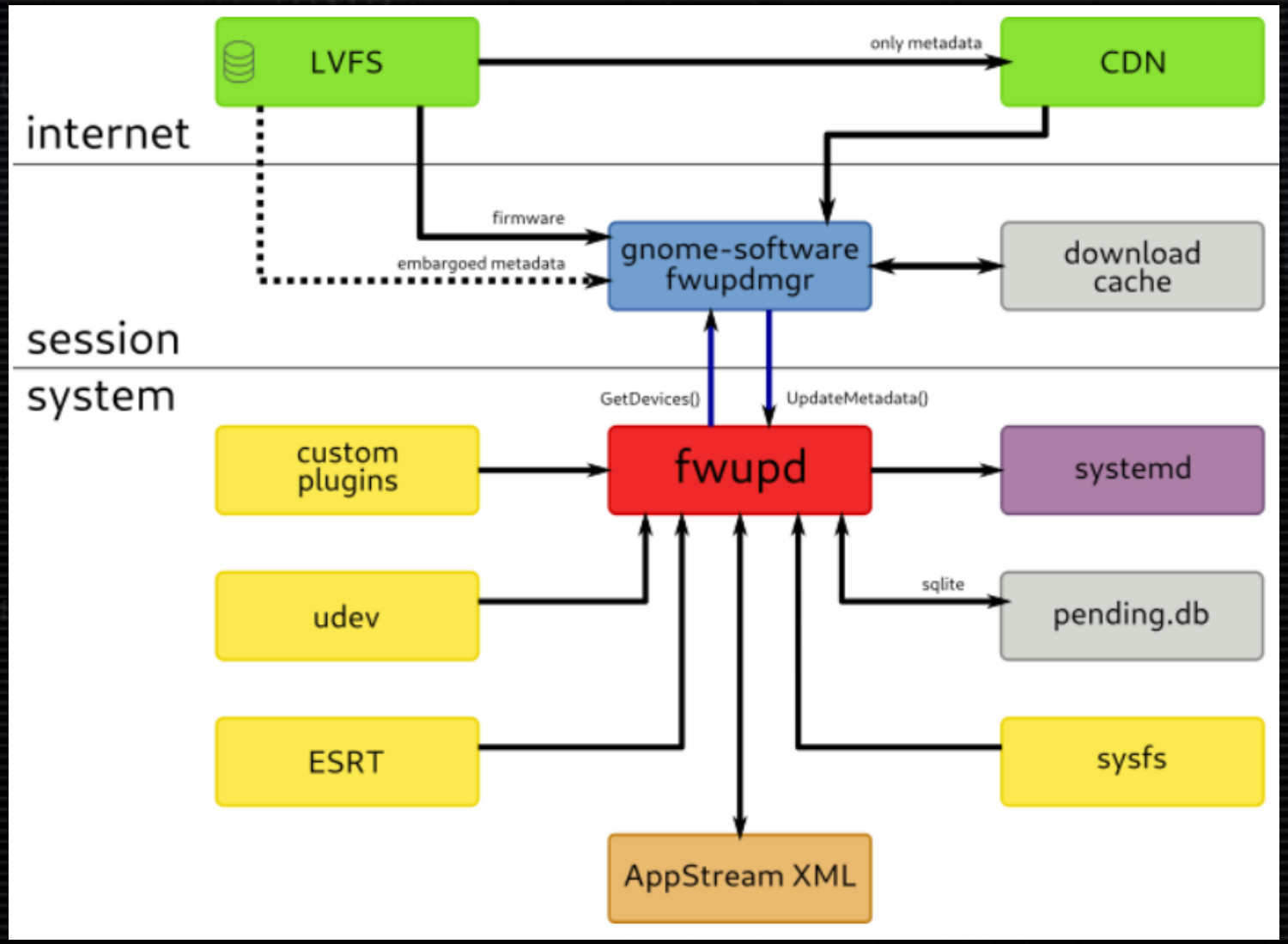



```
// Developer TODO - implement function to perform authentication check for the specific component image
INT32 ICompFwUpdateBspAuthenticateFWImage(void);

// Developer TODO - implement function to perform any required functionality to let the
//                  system know a new image has been downloaded and verified. (ex: this
//                  could be where the boot loader is modified to point to the new image )
void ICompFwUpdateBspSignalUpdateComplete(void);
```

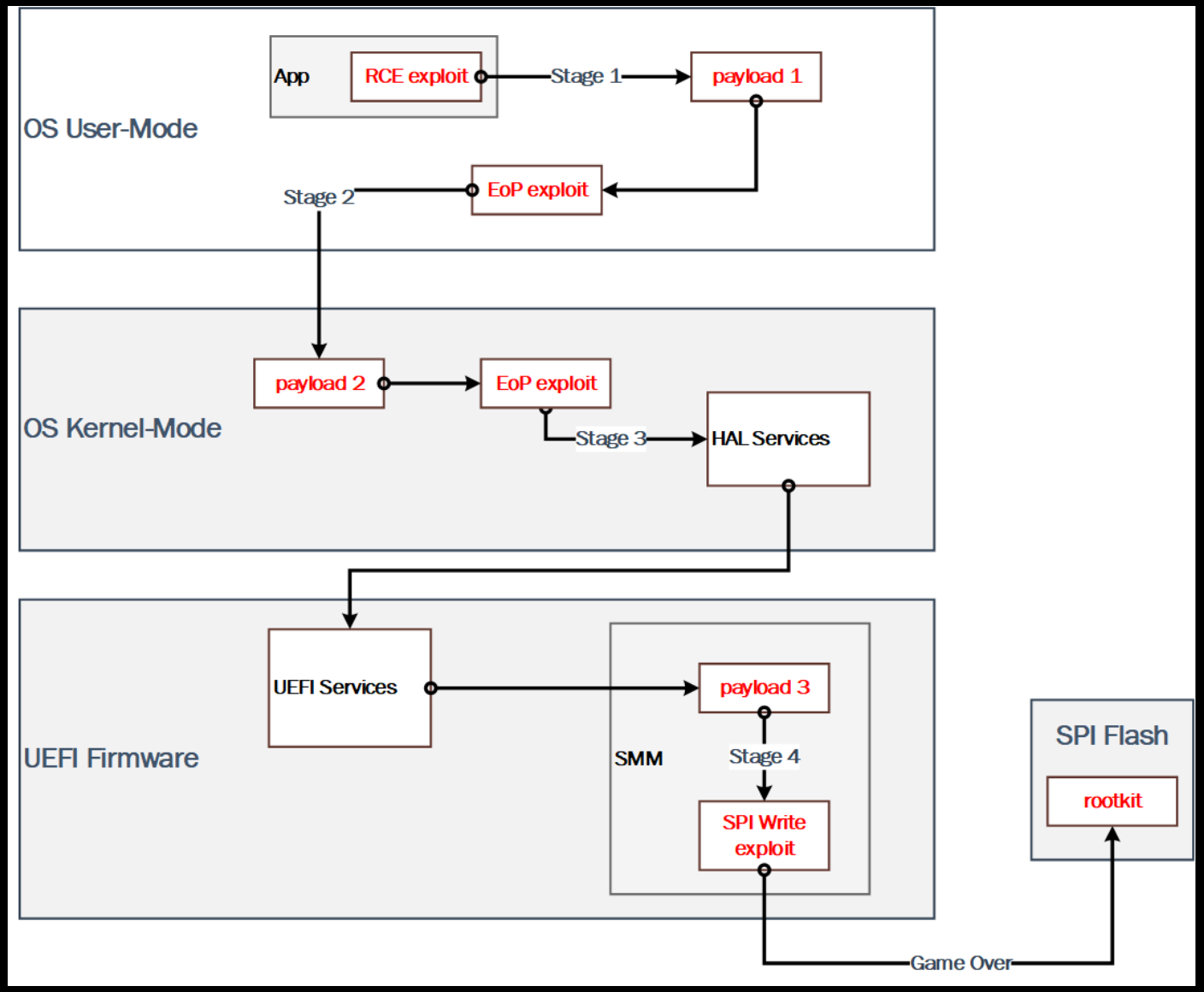
https://github.com/Microsoft/CFU

Linux Vendor Firmware Service



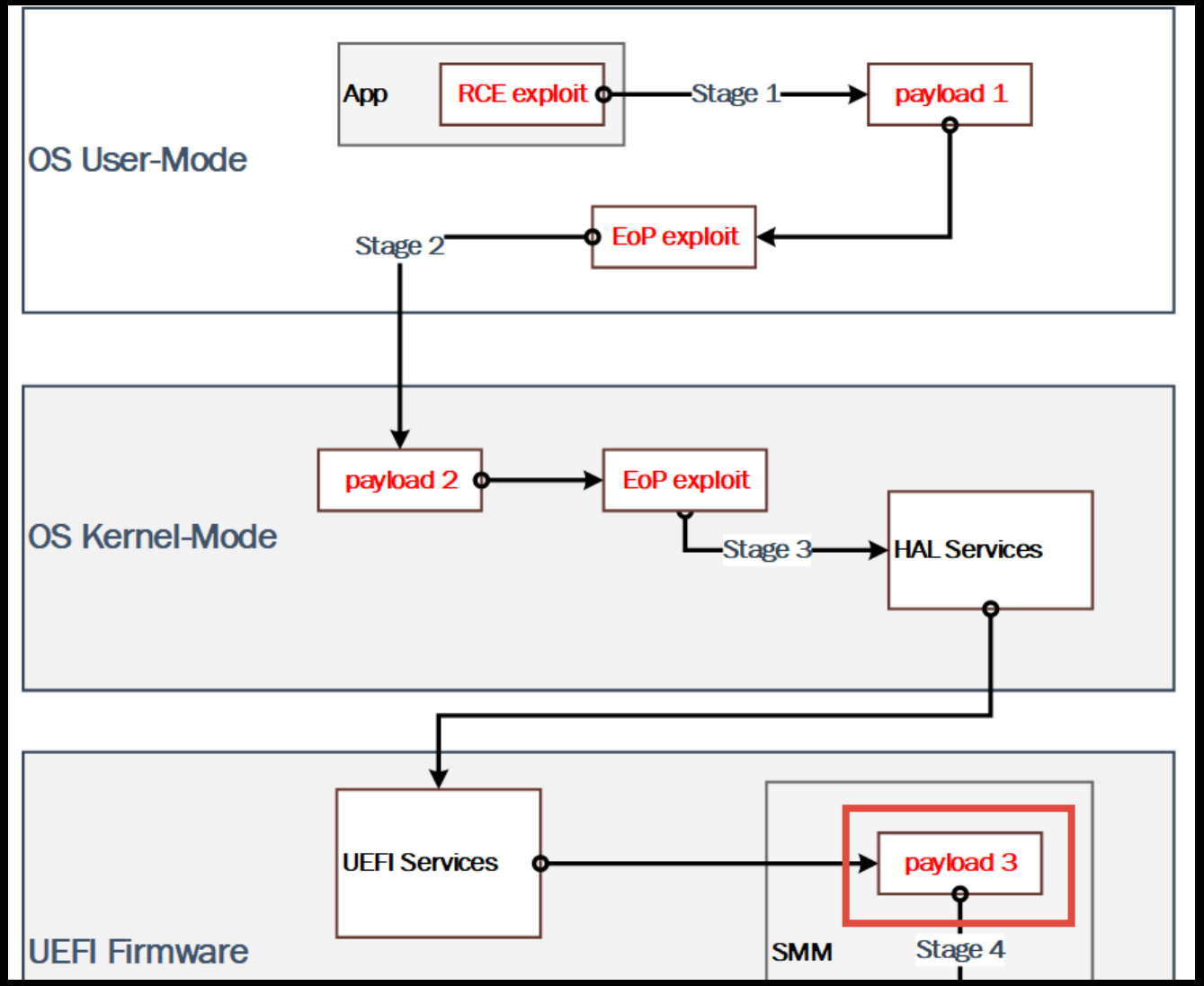
https://github.com/hughsie/fwupd

How many steps you need to gain BIOS persistence?



https://medium.com/@matrosov/dangerous-update-tools-c246f7299459

How many steps you need to gain BIOS persistence?



SMM Runtime Persistence

https://medium.com/@matrosov/dangerous-update-tools-c246f7299459

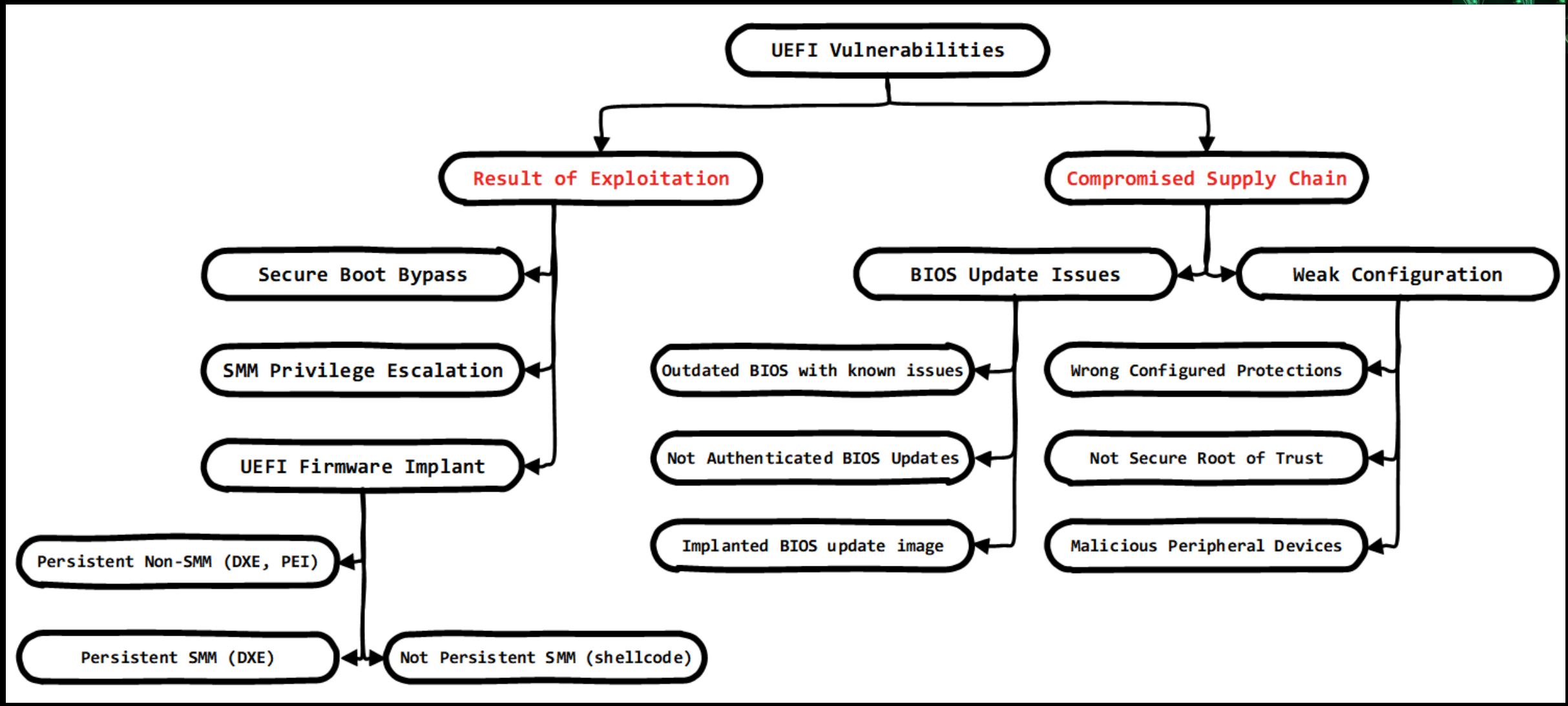
Why the golden age of Firmware/Hardware Implants is happening right now?!

Why firmware is a new big thing for the attacker?

- Firmware was not considered as a critical security asset for a long time
  - Firmware is patchable and can be updated in many cases in the field
    - The cost for firmware update is much lower vs hardware recall
  - Bring-up of the new hardware is hard. Many vendors by mistake misconfigured security features in the production stage.
- Everything goes to the cloud include firmware and hardware
  - In most of the cases cloud providers don't control firmware and hardware
    - Supply Chain attacks become a huge problem (nobody control all HW components)
    - Isolated VM instances can be attacked too (persistence in the guest instances)

http://2015.zeronights.ru/assets/files/10-Matrosov.pdf

UEFI persistence classification



https://medium.com/@matrosov/uefi-vulnerabilities-classification-4897596e60af

Average time of vulnerability disclosure in firmware 6-9 months!

(1-day exploit lifetime almost a year or longer!!)

Challenge of understanding attacker tactics or creating the right mitigations is related to mindset difference between attacker and architect.

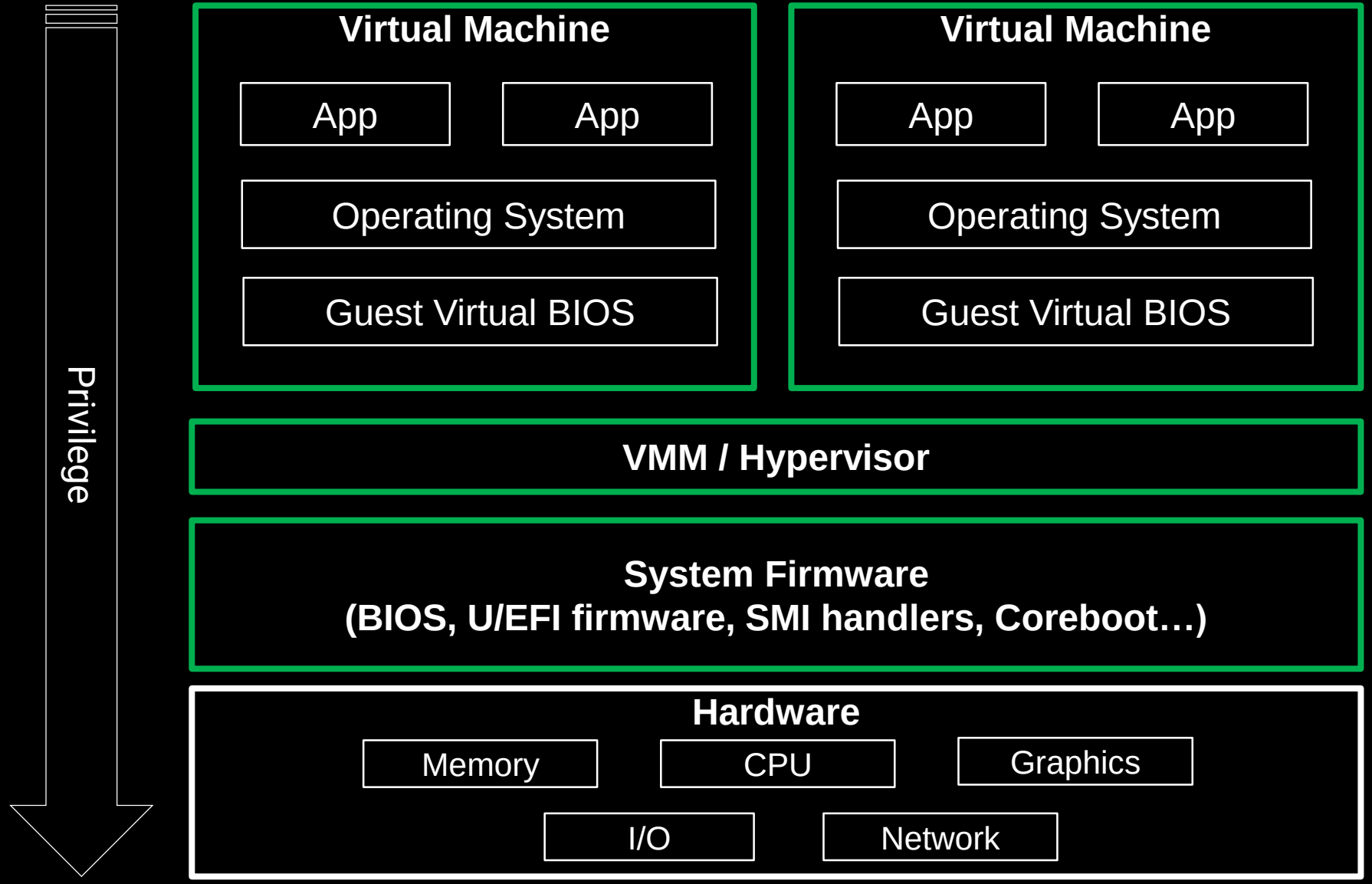
Offensive Research != Security Research

Mitigations Design != Security Architecture

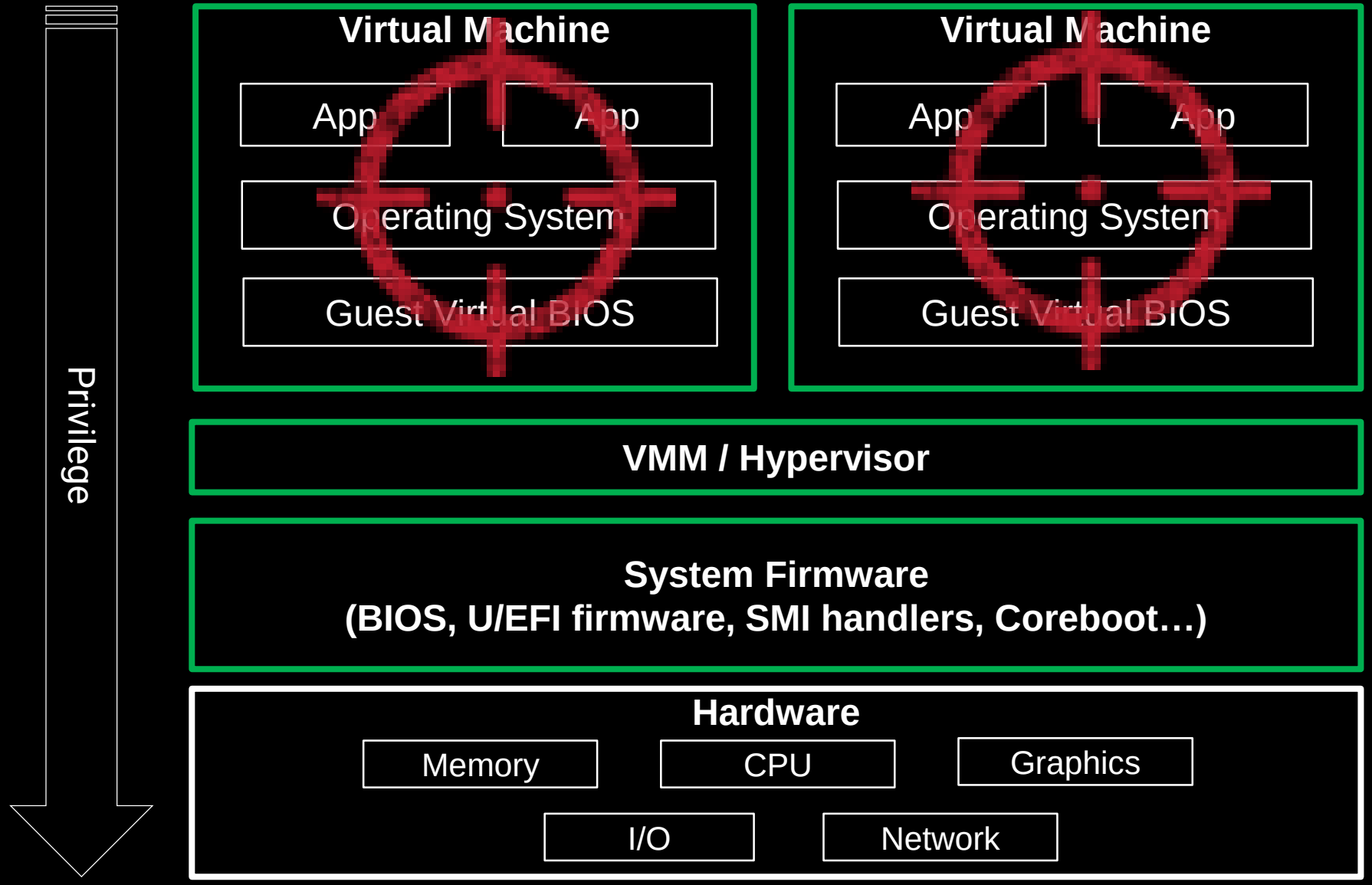
It’s why is important to have an internal offensive research team to uncover the real reality, not the one which is created by architects of mitigations.

Let’s go back to the cloud 😈

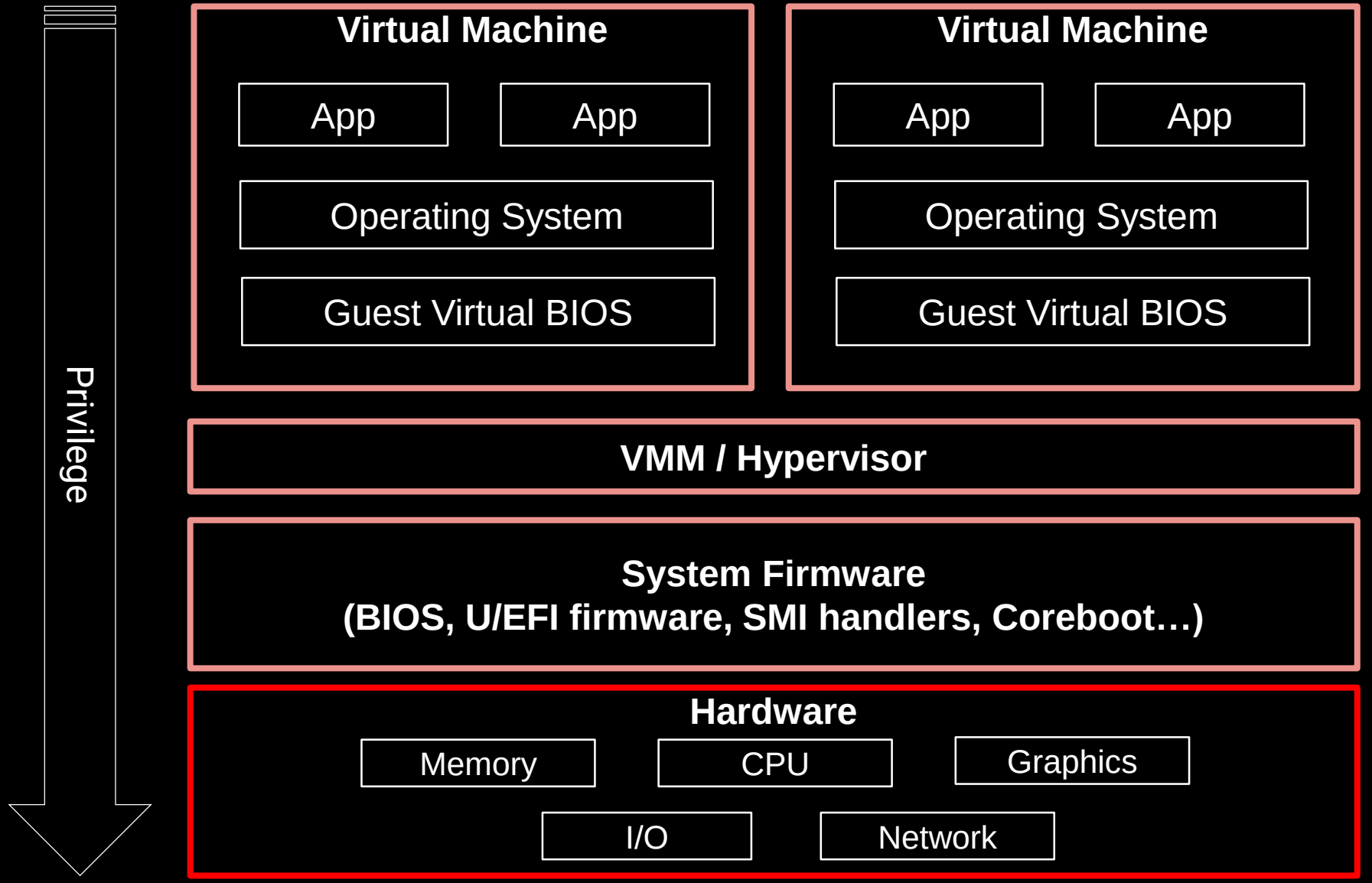
What can go wrong with the cloud?



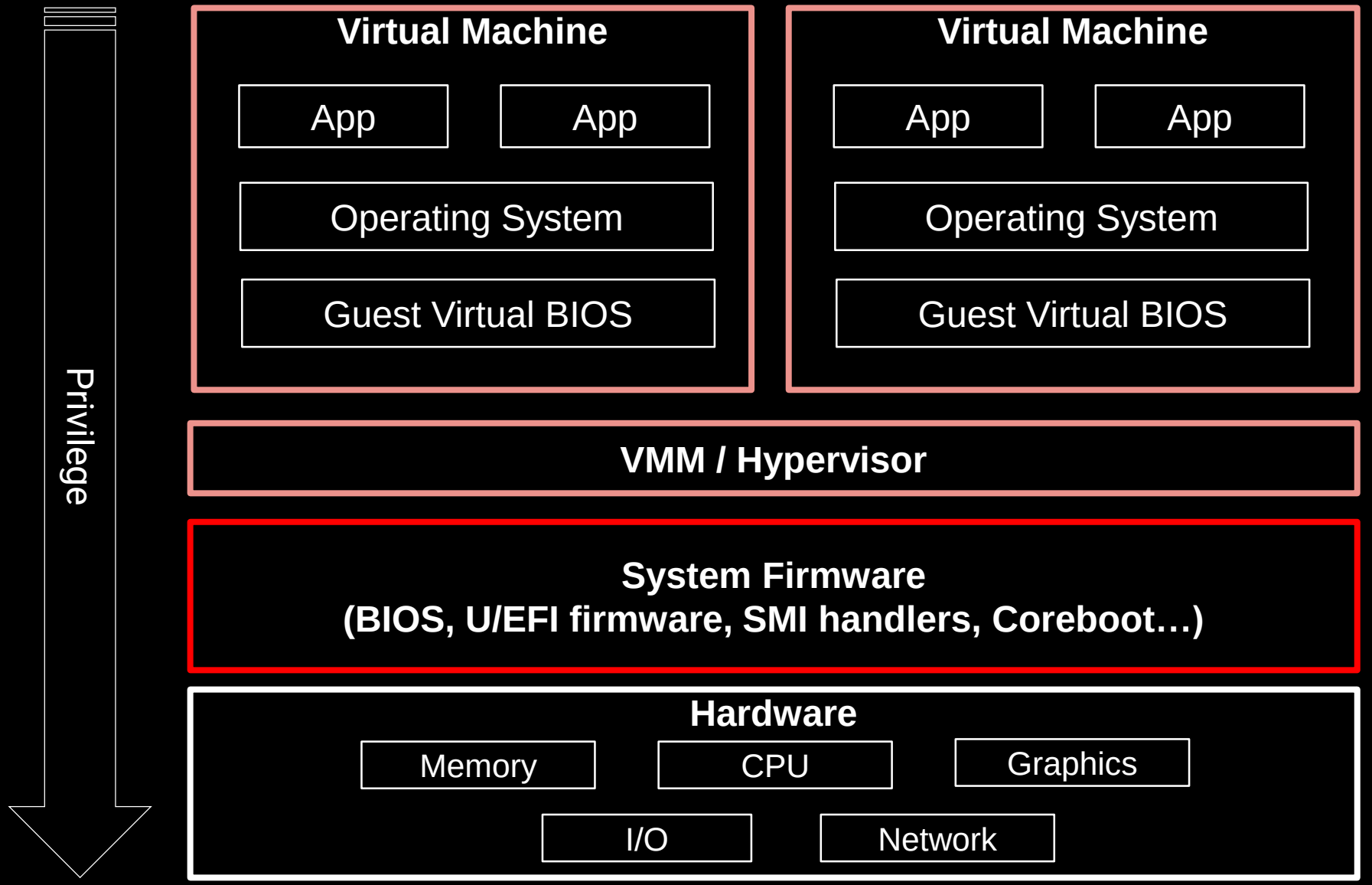
What can go wrong with the cloud?



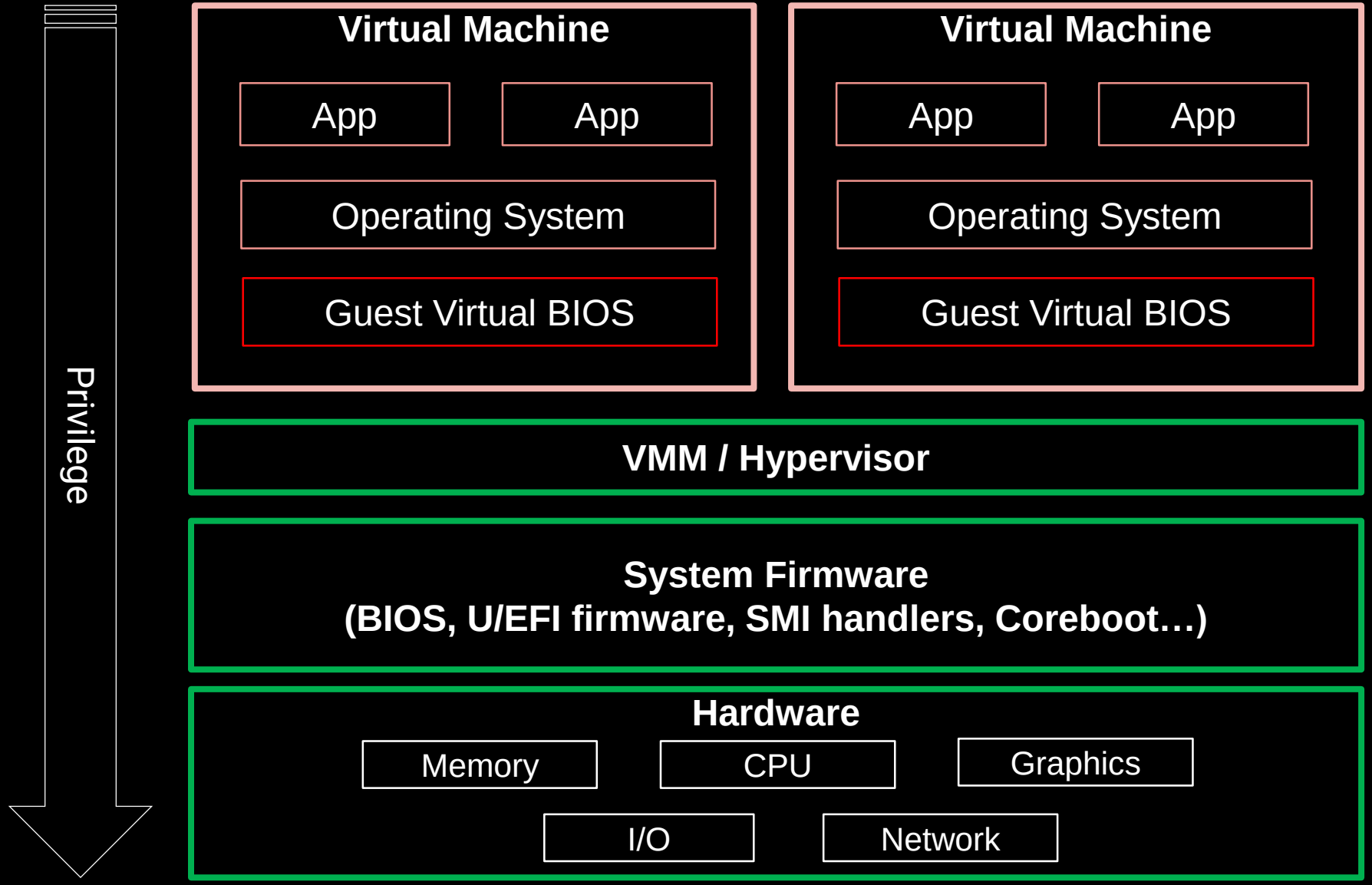
If BIOS is compromised it's ~~game~~ cloud over



If BIOS is compromised it's ~~game~~ cloud over



What if attacker gain persistence only in guest BIOS?



SeaBIOS/Coreboot in the cloud (in some cases)

```
←[34m[x][ ============================================================
[x][ Module: SPI Flash Descriptor Security Override Pin-Strap
[x][ ============================================================←[0m
←[39m[*] HSFS = 0x0000 << 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
    [03] BERASE             = 0 << Block/Sector Erase Size
    [05] SCIP               = 0 << SPI cycle in progress
    [13] FDOPSS             = 0 << Flash Descriptor Override Pin-Strap Status
    [14] FDV                = 0 << Flash Descriptor Valid
    [15] FLOCKDN            = 0 << Flash Configuration Lock-Down ←[0m
←[31m[-] FAILED: SPI Flash Descriptor Security Override is enabled←[0m
```

SeaBIOS/Coreboot in the cloud (in some cases)

```
-[34m[x][ ============================================================================
x][ Module: BIOS Region Write Protection
x][ ============================================================================←[0m
-[39m[*] BC = 0x00 << BIOS Control (b:d.f 00:31.0 + 0xDC)
   [00] BIOSWE           = 0 << BIOS Write Enable
   [01] BLE              = 0 << BIOS Lock Enable
   [02] SRC              = 0 << SPI Read Configuration
   [04] TSS              = 0 << Top Swap Status
   [05] SMM_BWP          = 0 << SMM BIOS Write Protection ←[0m
-[31m[-] BIOS region write protection is disabled!←[0m
-[39m
*] BIOS Region: Base = 0x00000000, Limit = 0x00000FFF←[0m
-[39mSPI Protected Ranges←[0m
-[39m----------------------------------------------------------------←[0m
-[39mPRx (offset) | Value    | Base     | Limit    | WP? | RP?←[0m
-[39m----------------------------------------------------------------←[0m
-[39mPR0 (74)     | 00000000 | 00000000 | 00000000 | 0   | 0 ←[0m
-[39mPR1 (78)     | 00000000 | 00000000 | 00000000 | 0   | 0 ←[0m
-[39mPR2 (7C)     | 00000000 | 00000000 | 00000000 | 0   | 0 ←[0m
-[39mPR3 (80)     | 00000000 | 00000000 | 00000000 | 0   | 0 ←[0m
-[39mPR4 (84)     | 00000000 | 00000000 | 00000000 | 0   | 0 ←[0m
-[39m←[0m
-[31m[!] None of the SPI protected ranges write-protect BIOS region←[0m
-[39m←[0m
-[31m[!] BIOS should enable all available SMM based write protection mechanisms or configure SPI protected ranges
-[31m[-] FAILED: BIOS is NOT protected completely←[0m
```

SeaBIOS/Coreboot in the cloud (in some cases)

```
[*] running module: chipsec.modules.common.spi_access←[0m
←[34m[x][ ================================================================================
[x][ Module: SPI Flash Region Access Control
[x][ ================================================================================←[0m
←[39mSPI Flash Region Access Permissions←[0m
←[39m----------------------------------------------------------------←[0m
←[39m[*] FRAP = 0x00000000 << SPI Flash Regions Access Permissions Register (SPIBAR + 0x50)
    [00] BRRA             = 0 << BIOS Region Read Access
    [08] BRWA             = 0 << BIOS Region Write Access
    [16] BMRAG            = 0 << BIOS Master Read Access Grant
    [24] BMWAG            = 0 << BIOS Master Write Access Grant ←[0m
←[39m←[0m
←[39mBIOS Region Write Access Grant (00):←[0m
←[39m  FREG0_FLASHD: 0←[0m
←[39m  FREG1_BIOS  : 0←[0m
←[39m  FREG2_ME    : 0←[0m
←[39m  FREG3_GBE   : 0←[0m
←[39m  FREG4_PD    : 0←[0m
←[39m  FREG5       : 0←[0m
←[39m  FREG6       : 0←[0m
←[39mBIOS Region Read Access Grant (00):←[0m
←[39m  FREG0_FLASHD: 0←[0m
←[39m  FREG1_BIOS  : 0←[0m
←[39m  FREG2_ME    : 0←[0m
←[39m  FREG3_GBE   : 0←[0m
←[39m  FREG4_PD    : 0←[0m
←[39m  FREG5       : 0←[0m
←[39m  FREG6       : 0←[0m
←[39mBIOS Region Write Access (00):←[0m
←[39m  FREG0_FLASHD: 0←[0m
←[39m  FREG1_BIOS  : 0←[0m
←[39m  FREG2_ME    : 0←[0m
←[39m  FREG3_GBE   : 0←[0m
←[39m  FREG4_PD    : 0←[0m
←[39m  FREG5       : 0←[0m
←[39m  FREG6       : 0←[0m
←[39mBIOS Region Read Access (00):←[0m
```

Big cloud providers like Amazon/Google/MS are not that bad

Shielded VM can help you protect your system from attack vectors like:

- Malicious guest OS firmware, including malicious UEFI extensions
- Boot and kernel vulnerabilities in guest OS
- Malicious insiders within your organization

To guard against these kinds of advanced persistent attacks, Shielded VM uses:

- **Unified Extensible Firmware Interface (UEFI)**: Ensures that firmware is signed and verified
- **Secure and Measured Boot**: Help ensure that a VM boots an expected, healthy kernel
- **Virtual Trusted Platform Module (vTPM)**: Establishes a root-of-trust, underpins Measured Boot, and prevents exfiltration of vTPM-sealed secrets
- **Integrity Monitoring**: Provides tamper-evident logging, integrated with Stackdriver, to help you quickly identify and remediate changes to a known integrity state

https://cloud.google.com/blog/products/identity-security/shielded-vm-your-ticket-to-guarding-against-rootkits-and-exfiltration
https://docs.microsoft.com/en-us/windows-server/security/guarded-fabric-shielded-vm/guarded-fabric-and-shielded-vms

VM guest BIOS persistence out of scope for existing security solutions!

BMC is a key from Data Center internal network

BMC in many cases expose different path to update BIOS

Targeting firmware update

Firmware update

- Complex file format parsing
- Various signature checks
- A vulnerability might allow to install a backdoored firmware

Accessible from both the host and the web server

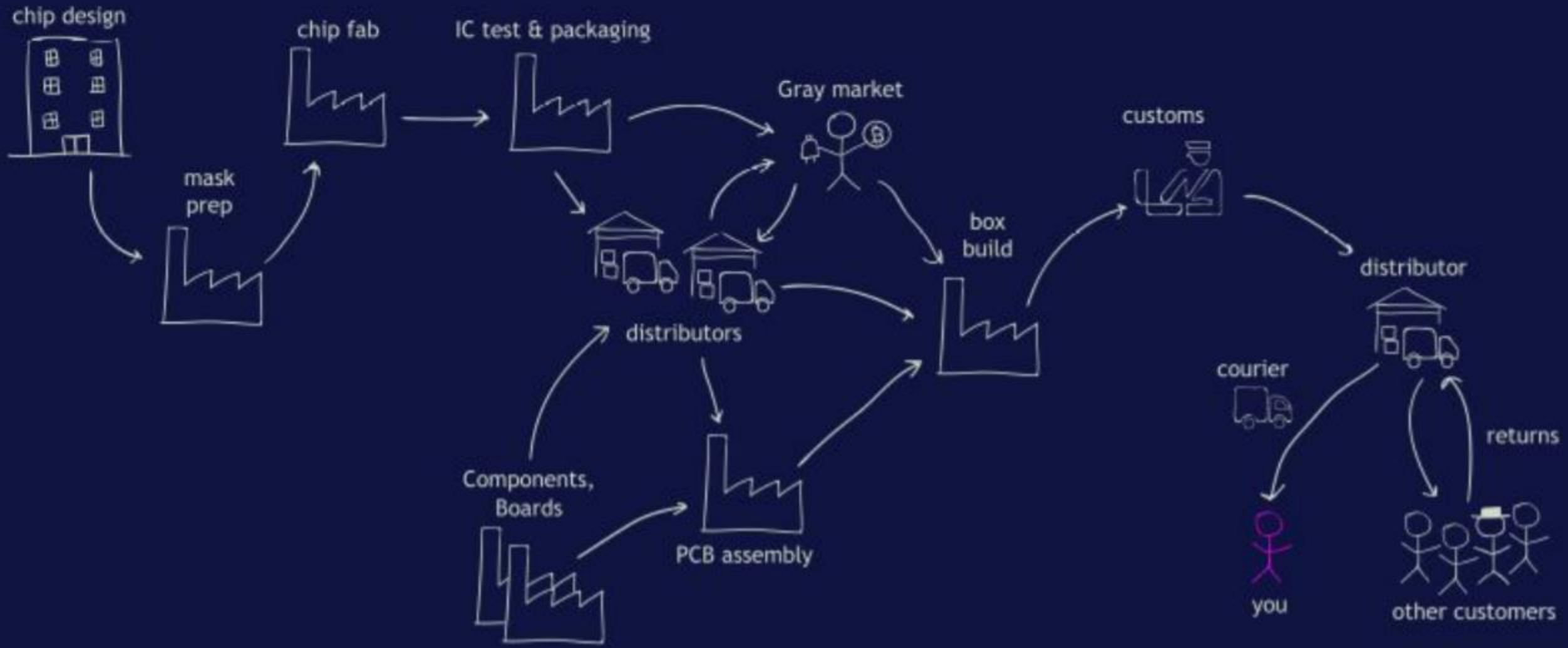
https://2018.zeronights.ru/wp-content/uploads/materials/01-Turning-your-BMC-into-a-revolving-door.pdf

https://www.sstic.org/media/SSTIC2019/SSTIC-actes/iDRACKAR/SSTIC2019-Slides-iDRACKAR-iooss.pdf

https://www.immunityinc.com/downloads/The-Unbearable-Lightness-of-BMC-wp.pdf

Supply Chain Attacks on HW/FW:
growing problems

It's A Huge Attack Surface...



https://www.bluehatil.com/abstracts#collapseSupply

Halvar Flake's wisdom

Scaling, firmware engines, and inspectability

Nobody has a good way to assure that a given device is reset into a "known-good" state, **especially** if the hardware was under physical attacker control.

<- THIS!

We can't check the transistors.

We can't check the firmware origin.

Establishing "who is in control" is near-impossible against strong adversaries.

https://docs.google.com/presentation/d/17bKudNDduvN-7hWv7S84MiHUj2AnOPNbwjTM8euDC8w

Why for hardware vendors security is not on the first place?

WTF Hardware Root of Trust?

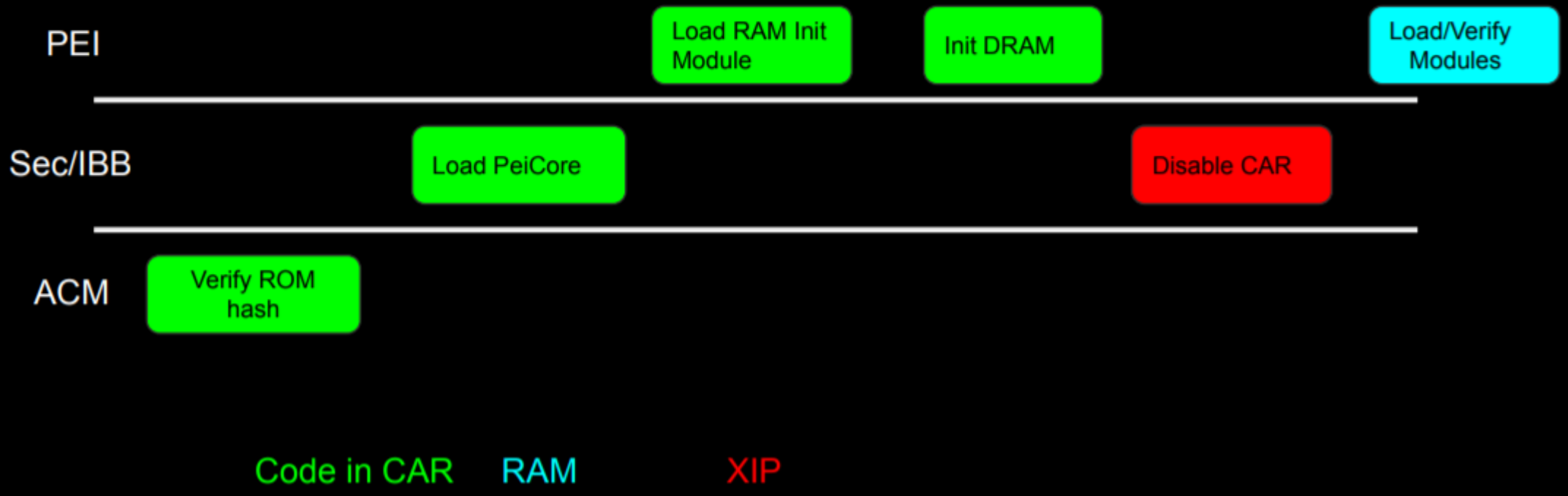
- Root of Trust baked in pure Hardware?
  - Cant be extracted/modified from software (developed in RTL)?
    - not flexible with OEM’s
    - hard to support in the field (updates and etc.)
    - hard to implement secure way to cooperate with firmware on the same chip

- In most of the cases Hardware Root of Trust it’s a mix between firmware and locked in the FUSE value or by specific bit.

- Secure state transition between hardware and firmware is hard. It’s always something missing.

TOCTOU on SPI Flash open the doors

Early Boot: ACM, Sec and PEI Phases



https://conference.hitb.org/hitbsecconf2019ams/sessions/now-you-see-it-toctou-attacks-against-secure-boot-and-bootguard/

Intel ACM's and Microcode downgrade

UEFITool NE alpha 55 (Mar 8 2019) - lenovo-p50.bin

File Action View Help

Structure

Name	Action	Type	Subtype
UEFI image		Image	UEFI
EfiSystemNvDataFvGuid		Volume	NVRAM
EfiFirmwareFileSystem2Guid		Volume	FFSv2
EfiFirmwareFileSystem2Guid		Volume	FFSv2
8579D1CA-45E8-4F1C-A789-FFA7706…		Volume	FFSv2
EfiFirmwareFileSystem2Guid		Volume	FFSv2
7934156D-CFCE-460E-92F5-A0790…		File	Raw
Volume free space		Free space	
EfiFirmwareFileSystem2Guid		Volume	FFSv2
Microcode		File	Raw
Volume free space		Free space	
Padding		Padding	Non-empty
B73FE497-B92E-416E-8326-45AD0D2…		Volume	FFSv2
BA34AA5B-110E-4B10-B729-E559EFD…		Volume	FFSv2

Parser | FIT | Security | Search | Builder

	Address	Size	Version	Checksum	Type
1	_FIT_	00000080h	0100h	13h	FIT Header
2	00000000FFE10060h	00018400h	0100h	00h	Microcode
3	00000000FFEB8000h	00008000h	0100h	00h	BIOS ACM
4	00000000FFFE0000h	00002000h	0100h	00h	BIOS Init
5	00000000FFEC0000h	00012000h	0100h	00h	BIOS Init
6	00000000FFDD0000h	00003000h	0100h	00h	BIOS Init
7	00000000FFEB5000h	00000241h	0100h	00h	BootGuard Key Manifest
8	00000000FFEB2000h	000002D3h	0100h	00h	BootGuard Boot Policy

https://github.com/LongSoft/UEFITool

Intel ACM's and Microcode downgrade

Alex Matrosov @matrosov · Jun 14

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

> **Alexander Ermolov** @flothrone · Jun 13
>
> Our team (@ttbr0 , @undermarble and me) walks through UEFI BIOS again, as a result:
> - 6 Escalation of Privileges to SMM
> - microcode downgrade vulnerability, allowing to bypass hardware root-of-trusts.
>
> Details coming soon!
> intel.com/content/www/us...

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

Halvar Flake's wisdom

Scaling, firmware engines, and inspectability

Scaling has yielded the performance gains of the last decades, but scaling made physical inspection impossible.

Universal computation has replaced many formerly-simpler components in your computer with full CPUs + firmware - usually without mechanism for inspection.

Current approach for firmware security is based on "ensuring nobody can get in" (code signing), but transient faults can be locally induced to bypass, and signing keys get stolen with regularity.

<- THIS!

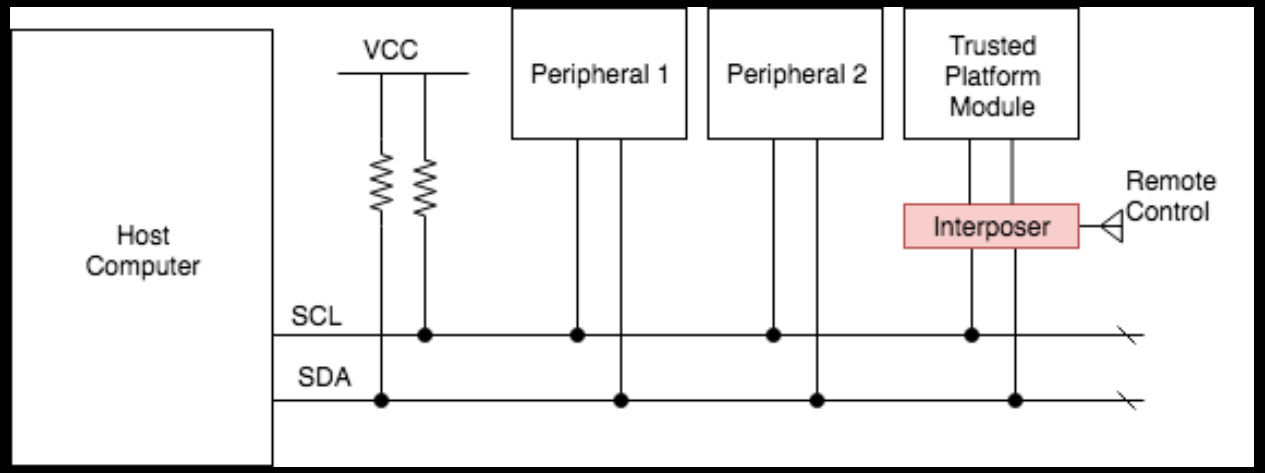
https://docs.google.com/presentation/d/17bKudNDduvN-7hWv7S84MiHUj2AnOPNbwjTM8euDC8w

3rd party components is part of Supply Chain hell

Supply Chain attack vectors extend attack surface which always been out of scope for HW/FW vendors.

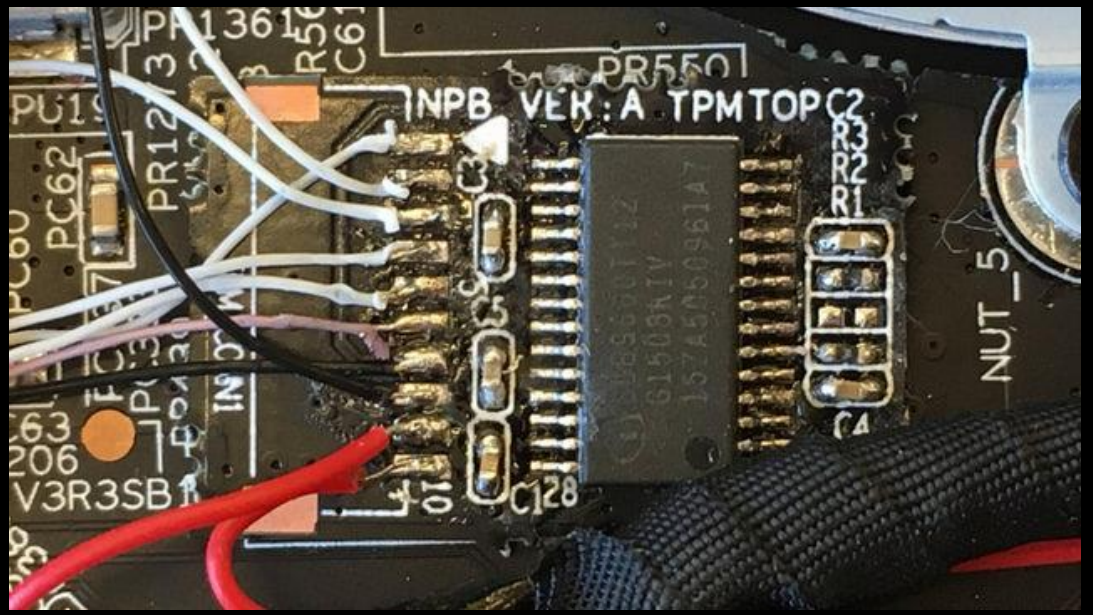
TPM – Root of ~~Trust~~ Problems

@uffeux



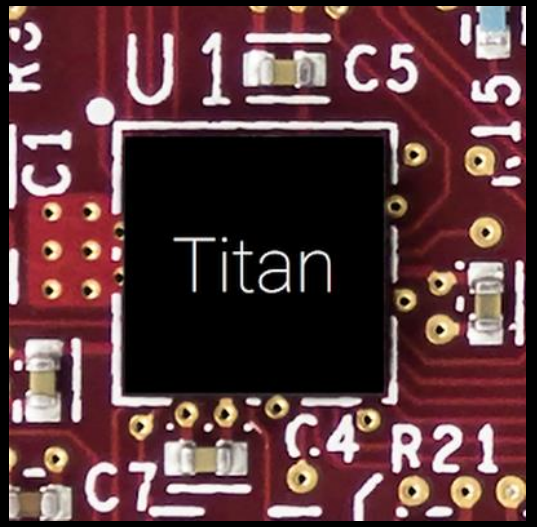
https://github.com/nccgroup/TPMGenie

@qrs



Major vendors trying to fix the Root

Google Titan



Apple T2

Microsoft Cerberus

Amzon Greengrass

https://cloud.google.com/blog/products/gcp/titan-in-depth-security-in-plaintext

https://www.apple.com/mac/docs/Apple_T2_Security_Chip_Overview.pdf

https://github.com/opencomputeproject/Project_Olympus/tree/master/Project_Cerberus

https://aws.amazon.com/greengrass/

Any of hardware vendors doesn’t have a full supply chain control

(a lot of 3rd party FW come as binary blobs)

Operation ShadowHammer (ASUS Live Update compromise) demonstrate security gaps in current whole update delivery process

https://securelist.com/operation-shadowhammer

https://securelist.com/operation-shadowhammer-a-high-profile-supply-chain-attack

REsearchers Arm Race never stops!

Thank you!

Platform Security Summit 2019

Oct 1-3, 2019 · Redmond, WA