RSAConference2022
San Francisco & Digital | June 6 – 9

TRANSFORM

SESSION ID: **HT-M01**

ESPecter: Showing the Future of UEFI Threats

Martin Smolár
Malware Researcher
ESET
@smolar_m

Jean-Ian Boutin
Director of Threat Research
ESET
@jiboutin

Disclaimer

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA® Conference, RSA Security LLC or any other co-sponsors. RSA Conference does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

So why are attackers interested in the Unified Extensible Firmware Interface (UEFI)?

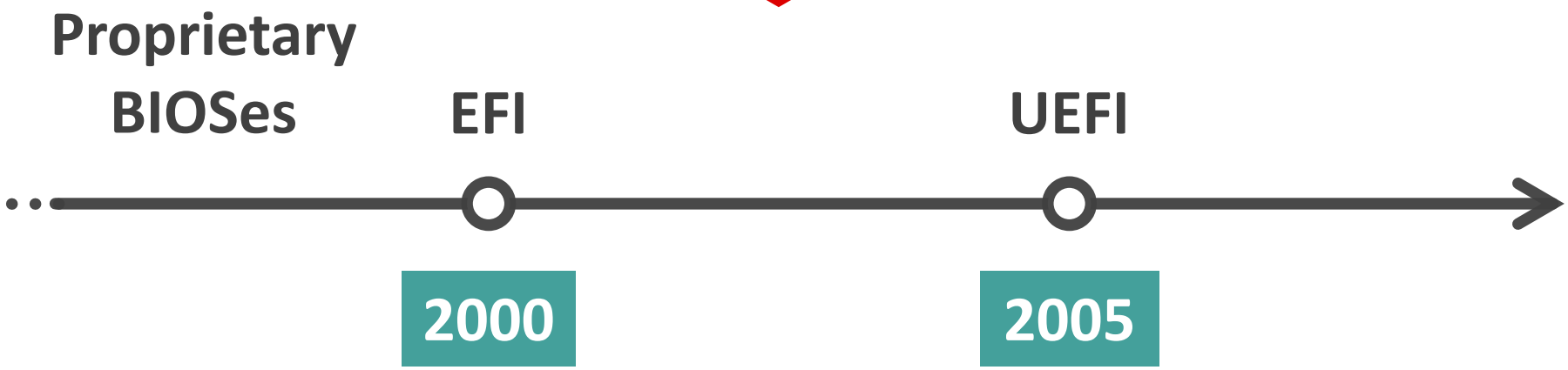
Detection is difficult

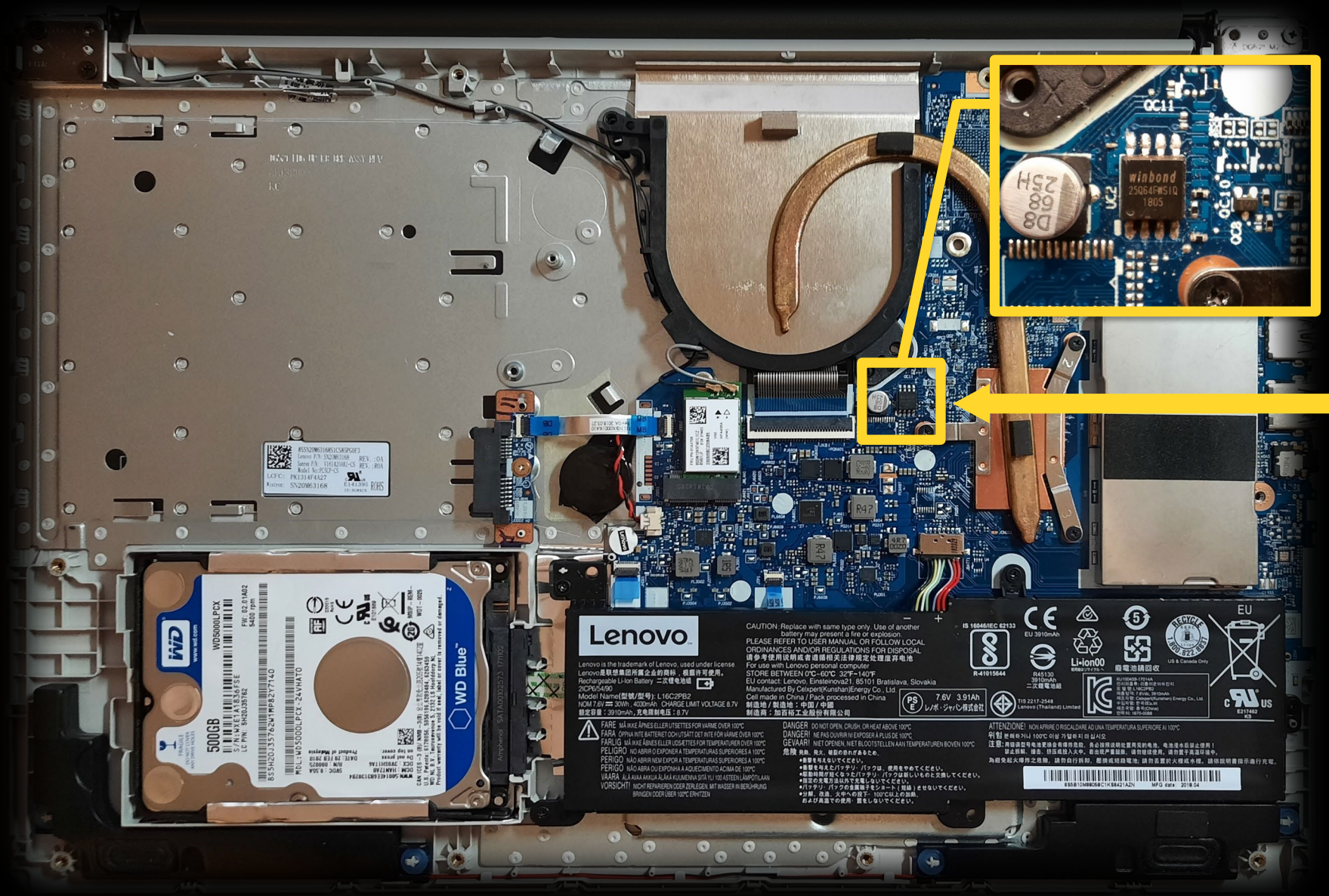
It can survive OS reinstallation

OS independent

UEFI Bootkit on ESP?

Can this trend TRANSFORM UEFI security?


Proprietary BIOSes
EFI
UEFI
2000
2005


winbond
25Q64FWSIQ
1805
UEFI Firmware location (SPI Flash chip)

Vault 7: CIA Hacking Tools Revealed

DerStarke 2.0

('toc' missing)

Hacking Team Uses UEFI BIOS Rootkit

By: Trend Micro
July 13, 2015
Read time: 2 min (747 words)

Subscribe

Authors

Trend Micro
Research, News, and Perspectives

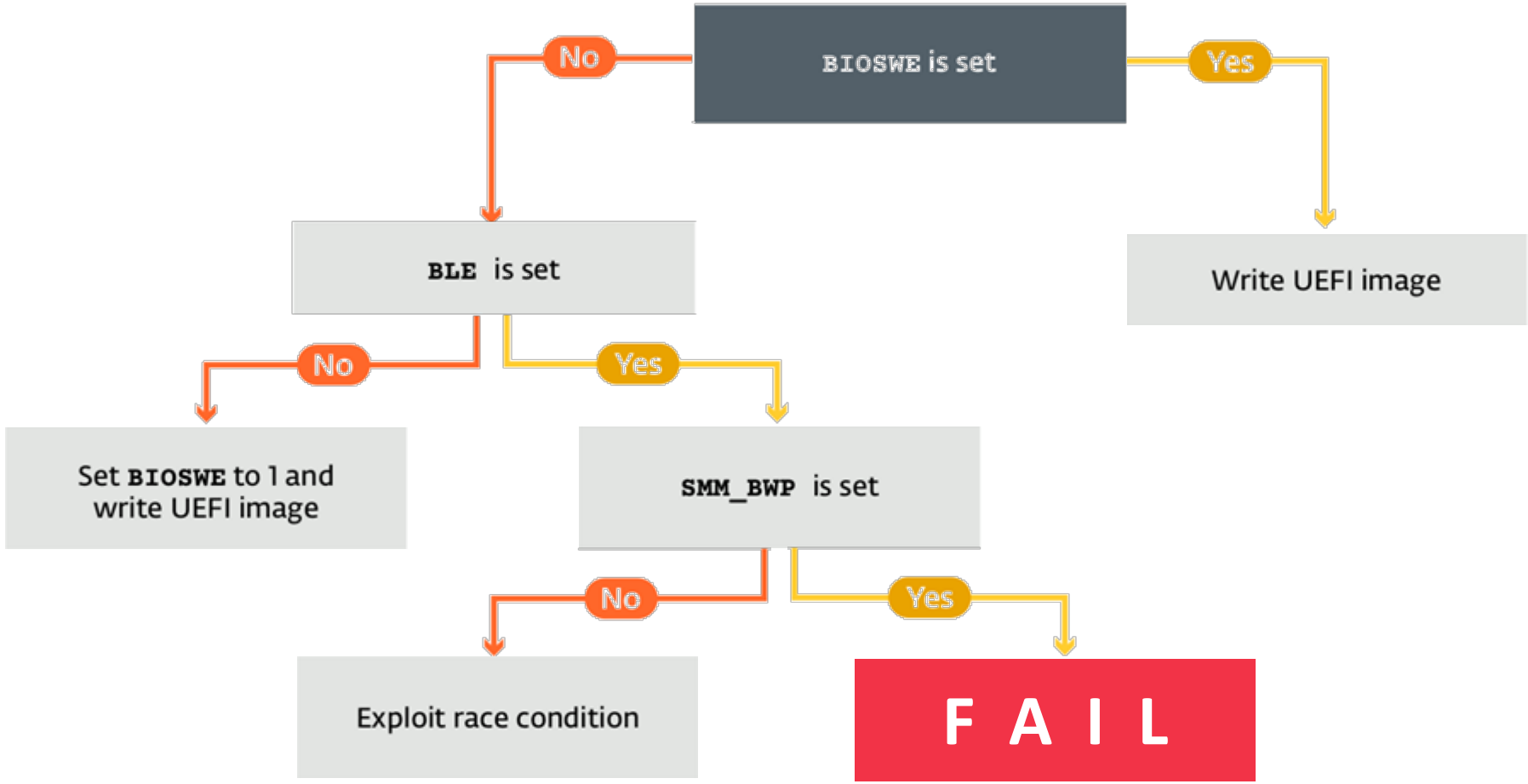
Contact Us

Subscribe

The dissection of the data from the Hacking Team leak has yielded another critical discovery: Hacking Team uses a UEFI BIOS rootkit to keep their Remote Control System (RCS) agent installed in their targets' systems. This means that even if the user formats the hard disk, reinstalls the OS, and even buys a new hard disk, the agents are implanted after Microsoft Windows is up and running. They have written a procedure specifically for Insyde BIOS (a very popular BIOS vendor for laptops). However, the code can very likely work on AMI BIOS as well. A Hacking Team slideshow presentation claims that successful infection requires physical access to the target system; however, we can't rule out the possibility of remote installation. An example attack scenario would be: The intruder gets access to the target computer, reboots into UEFI shell, dumps the BIOS, installs the BIOS rootkit, reflashes the BIOS, and then reboots the target system. We've found that Hacking


welivesecurity BY eset
Menu
LoJax: First UEFI rootkit found in the wild, courtesy of the Sednit group
ESET researchers have shown that the Sednit operators used different components of the LoJax malware to target a few government organizations in the Balkans as well as in Central and Eastern Europe
ESET Research
27 Sep 2018 - 11:57AM

LoJax SPI Flash writing decision tree



RSAConference2022

What's next?


UEFI Firmware location (SPI Flash chip)
Difficult access
ESP location (HDD/SSD)
Easy access
WD Blue
500GB
Lenovo

Your PC is blocked.
Now pay 250$ to safe-data.ru@outlook.com.

Any way to recover your files will me be data loss !

But, so careful my kid :).

Complexity of threat versus added benefit

ESP implants

- \+ "Only" 1 protection (SecureBoot)
- \+ Easy to deploy (less privileges required)
- − Easier to discover/detect/remove
- − Does not survive OS reinstall

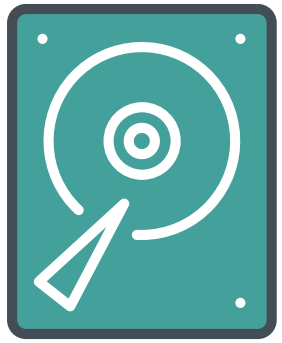
SPI Flash implants

- \+ Better persistence (survives OS reinstall)
- \+ Harder to discover/detect
- − Higher privileges required
- − Harder to achieve – multiple level protections
- − Mistakes in implant implementation can turn victim's device into a brick

RSAConference2022

ESPecter analysis

EFI System Partition (ESP)



GPT disk

Partition 1 – OS (ntfs, ext4, …)

Partition 2 – Data (ntfs, ext4, …)

ESP partition – **FAT32**

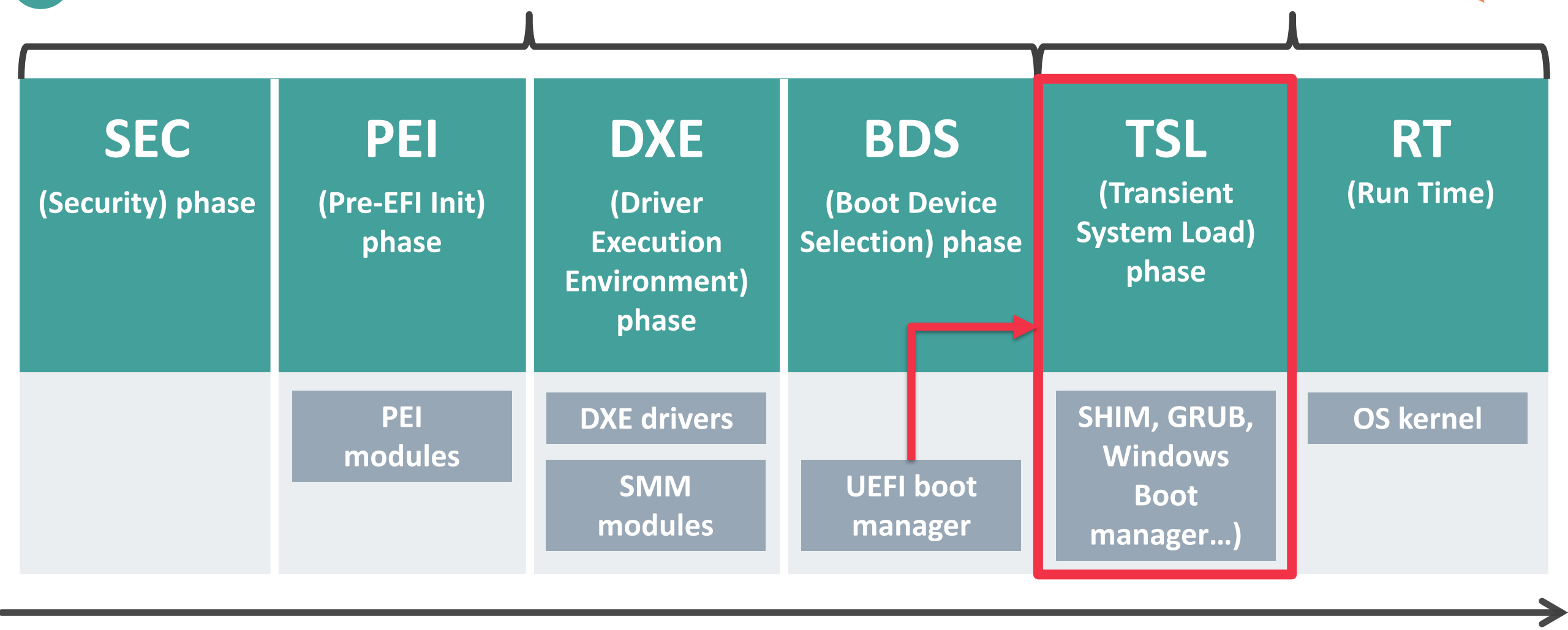
/**EFI**/**Microsoft**/**Boot**/bootmgfw.efi
/**ubuntu**/shimx64.efi
/**Vendor3**/bootx64.efi
…
/**EFI**/**Boot**/bootx64.efi

BootOrder = **0002 0001 0003**

Boot**0001** = **/EFI/Microsoft/boot/**bootmgfw.efi

Boot**0002** = **/EFI/ubuntu/**shimx64.efi

Boot**0003** = **/EFI/Vendor3/**bootx64.efi


SPI Flash firmware
3rd party apps/drivers
SEC
(Security) phase
PEI
(Pre-EFI Init) phase
DXE
(Driver Execution Environment) phase
BDS
(Boot Device Selection) phase
TSL
(Transient System Load) phase
RT
(Run Time)
PEI modules
DXE drivers
SMM modules
UEFI boot manager
SHIM, GRUB, Windows Boot manager…)
OS kernel
Boot process

ESPecter persistence

/EFI/Microsoft/Boot/**bootmgfw.efi**

/EFI/Boot/**bootx64.efi**

Inject extra section

/EFI/Microsoft/Boot/**bootmgfw.efi**

/EFI/Boot/**bootx64.efi**

```
Count of sections                5 | Machine                        AMD64
Symbol table  00000000[00000000]   |       Fri Jul 07 08:04:38 2017
Size of optional header       00F0 | Magic optional header          020B
Linker version               14.10 | OS version                     0.00
Image version                 0.00 | Subsystem version              1.00
Entry point                0001A730 | Size of code                0010EC00
Size of init data          00019E00 | Size of uninit data         00000000
Size of image              00166000 | Size of header              00000400
Base of code               00001000 |
Image base      00000000`10000000   | Subsystem                    EFI app
Section alignment          00001000 | File alignment              00000200
Stack           00000000`00100000   | Heap            00000000`00100000
Stack commit    00000000`00001000   | Heap commit     00000000`00001000
Checksum                   00130058 | Number of dirs                  16
Overlay         00128E00[00002198/8600/8,398 Kb]
```

```
Count of sections                6 | Machine                        AMD64
Symbol table  00000000[00000000]   |       Fri Jul 07 08:04:38 2017
Size of optional header       00F0 | Magic optional header          020B
Linker version               14.10 | OS version                     0.00
Image version                 0.00 | Subsystem version              1.00
Entry point                00166000 | Size of code                0010EC00
Size of init data          00019E00 | Size of uninit data         00000000
Size of image              00183000 | Size of header              00000400
Base of code               00001000 |
Image base      00000000`10000000   | Subsystem                    EFI app
Section alignment          00001000 | File alignment              00000200
Stack           00000000`00100000   | Heap            00000000`00100000
Stack commit    00000000`00001000   | Heap commit     00000000`00001000
Checksum                   00130058 | Number of dirs                  16
Overlay         00145800[00002198/8600/8,398 Kb]
```

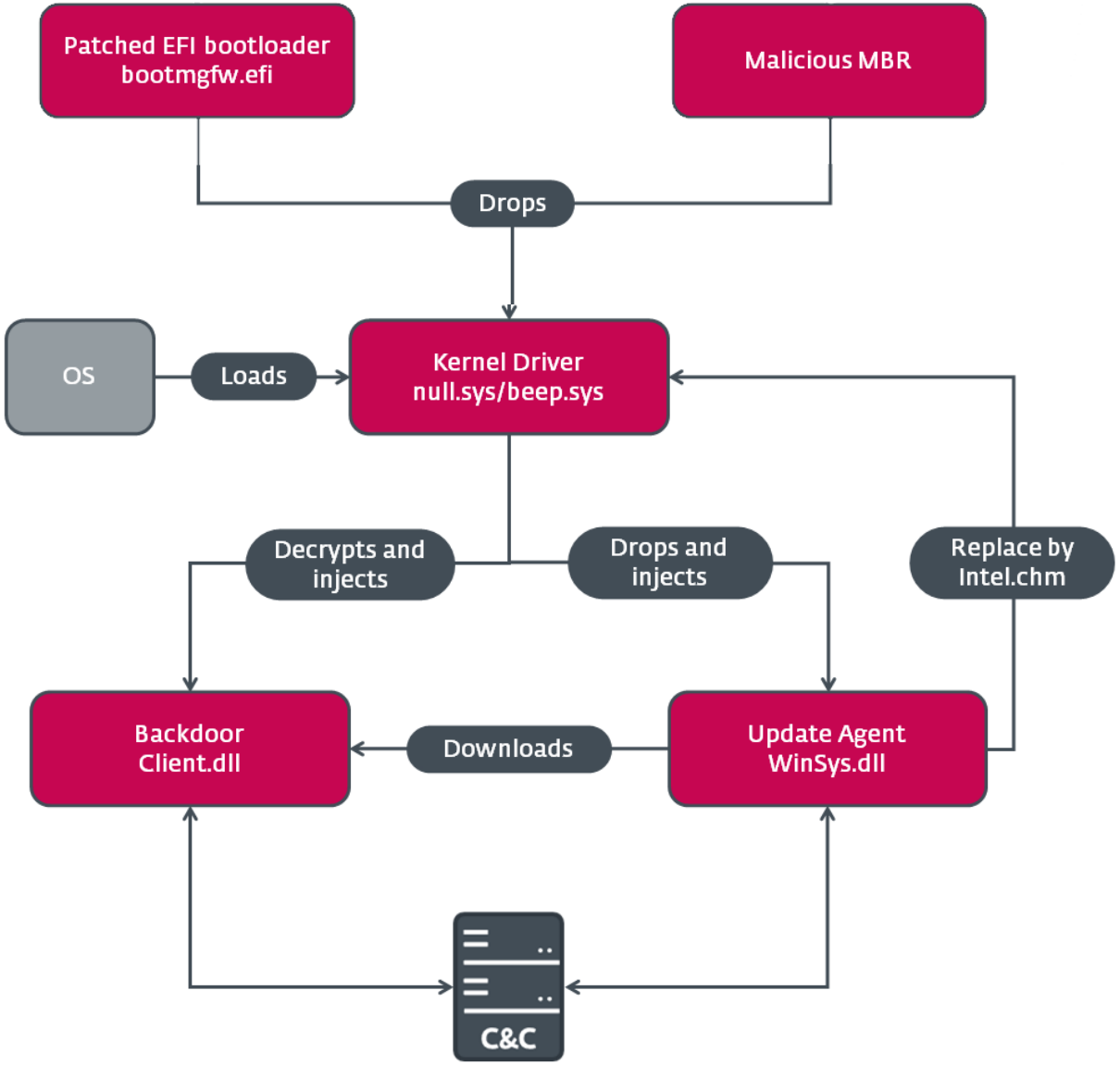
ESPecter overview

2012

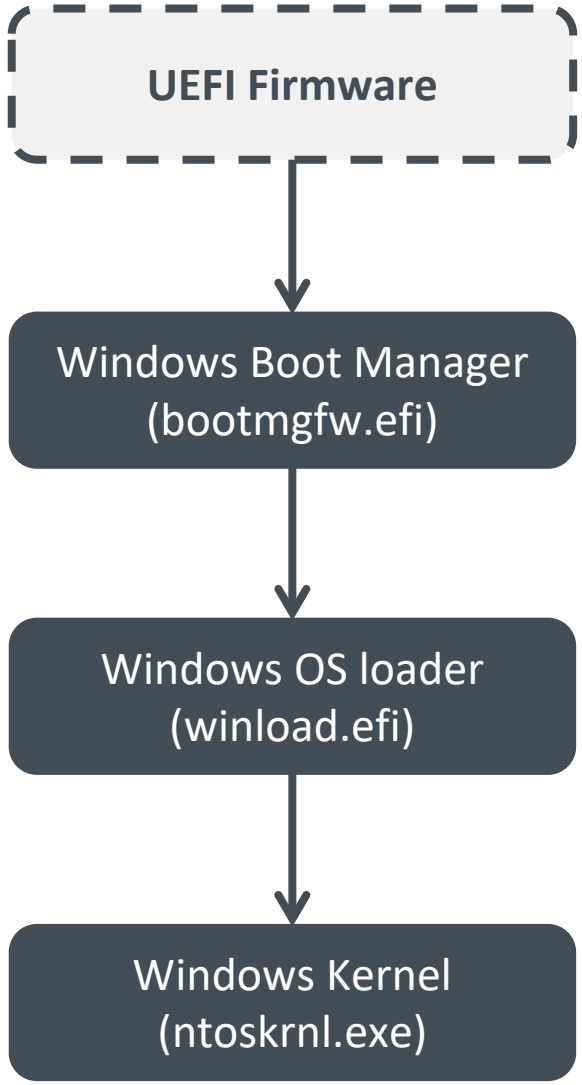
Legacy version

2020

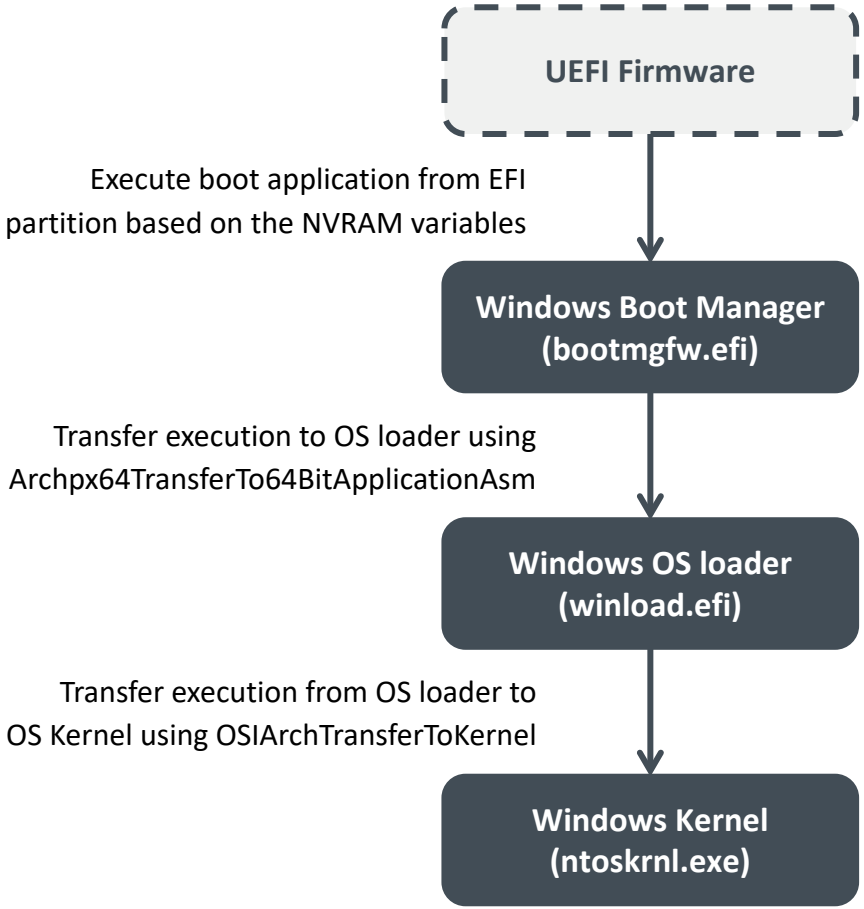
UEFI version

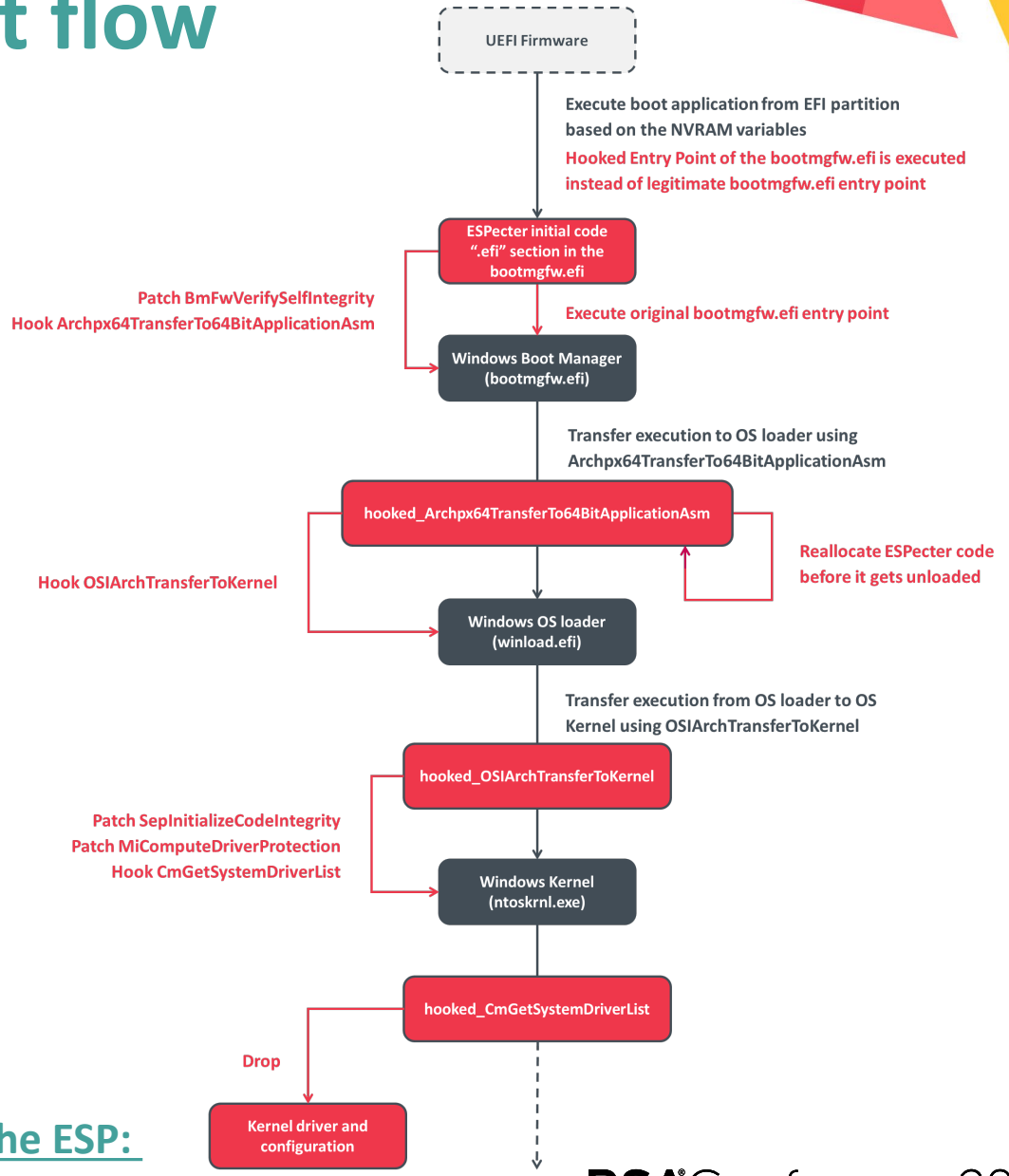


Typical Windows boot flow



Compromised Windows boot flow





Disable Boot Manager's self integrity check

```
if ( Patch_BmFwVerifySelfIntegrity_pattern1(mem, length) )// 33 FF 48
{
  if ( Patch_BmFwVerifySelfIntegrity_pattern2(mem, length) )// 33 FF 4
  {
    if ( Patch_BmFwVerifySelfIntegrity_pattern3(mem, length) )// 33 FF
    {
      if ( Patch_BmFwVerifySelfIntegrity_pattern4(mem, length) )// 33
      {
        if ( Patch_BmFwVerifySelfIntegrity_pattern5(mem, length) )// 3
```

```
  v2 = *(_BYTE *)mem == 0x33;
  mem = (_DWORD *)((char *)mem + 1);
  --v4;
}
while ( !v2 );
if ( !v2 )
  break;
v2 = *mem == 0x658348FF;
if ( *mem == 0x658348FF )
{
  v2 = mem[1] == 0x834800C8;
  if ( mem[1] == 0x834800C8 )
  {
    v2 = mem[2] == 0x83004865;
    if ( mem[2] == 0x83004865 )
    {
      // FOUND PATTERN   33 FF 48 83 65 C8 00 48 83 65 48 00 83

      // PATCH WITH
      // B8 00 00 00 00   mov eax, 0
      // C3               ret
      *((_BYTE *)mem - 23) = 0xB8;
      *(_DWORD *)((char *)mem - 22) = 0;
      *((_BYTE *)mem - 18) = 0xC3;
      result = 0i64;
```

Driver signature enforcement disable

Original

```
__int64 SepInitializeCodeIntegrity()
{
  unsigned int CiOptions; // edi
  _LIST_ENTRY *p_BootDriverListHead; // rbx
  _LOADER_PARAMETER_EXTENSION *Extension; // rcx
  _LOADER_PARAMETER_CI_EXTENSION *CodeIntegrityData; // rdx
  char *LoadOptions; // rcx

  CiOptions = 6;
  memset(&unk_140438464, 0, 0xC4ui64);
  p_BootDriverListHead = 0i64;
  SeCiCallbacks = 0xD0;
  qword_140438528 = 0xA000007i64;
  if ( KeLoaderBlock_0 )
  {
    Extension = KeLoaderBlock_0->Extension;
    if ( Extension )
    {
      CodeIntegrityData = Extension->CodeIntegrityData;
      if ( CodeIntegrityData )
        CiOptions = CodeIntegrityData->CodeIntegrityOptions;
    }
    LoadOptions = KeLoaderBlock_0->LoadOptions;
    if ( LoadOptions && SepIsOptionPresent(LoadOptions) )
      SeCiDebugOptions |= 1u;
    if ( KeLoaderBlock_0 )
      p_BootDriverListHead = &KeLoaderBlock_0->BootDriverList
  }
  return CiInitialize(CiOptions, p_BootDriverListHead, &SeCi
}
```

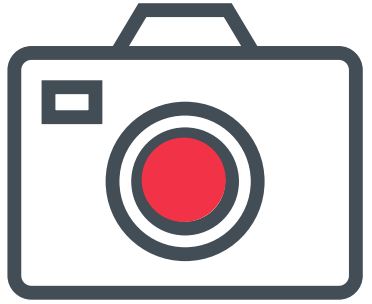
Patched

```
__int64 SepInitializeCodeIntegrity()
{
  unsigned int CiOptions; // edi
  _LIST_ENTRY *p_BootDriverListHead; // rbx
  _LOADER_PARAMETER_EXTENSION *Extension; // rcx
  _LOADER_PARAMETER_CI_EXTENSION *CodeIntegrityData; // r
  char *LoadOptions; // rcx

  CiOptions = 6;
  memset(&unk_140438464, 0, 0xC4ui64);
  p_BootDriverListHead = 0i64;
  SeCiCallbacks = 0xD0;
  qword_140438528 = 0xA000007i64;
  if ( KeLoaderBlock_0 )
  {
    Extension = KeLoaderBlock_0->Extension;
    if ( Extension )
    {
      CodeIntegrityData = Extension->CodeIntegrityData;
      if ( CodeIntegrityData )
        CiOptions = 0;
    }
    LoadOptions = KeLoaderBlock_0->LoadOptions;
    if ( LoadOptions && SepIsOptionPresent(LoadOptions) )
      SeCiDebugOptions |= 1u;
    if ( KeLoaderBlock_0 )
      p_BootDriverListHead = &KeLoaderBlock_0->BootDriver
  }
  return CiInitialize(CiOptions, p_BootDriverListHead, &S
}
```

PATCH CiOptions

ESPecter espionage



Attribution


```
                        ; DATA XREF: FullScreenshot+6E↑o
text "UTF-16LE", '截图全屏失败!',0Ah,0
align 10h

                        ; DATA XREF: ForegroundWindowScreenshot+63↑o
text "UTF-16LE", '截图前台窗口失败!',0Ah,0
align 10h
```

Chinese speaking TA

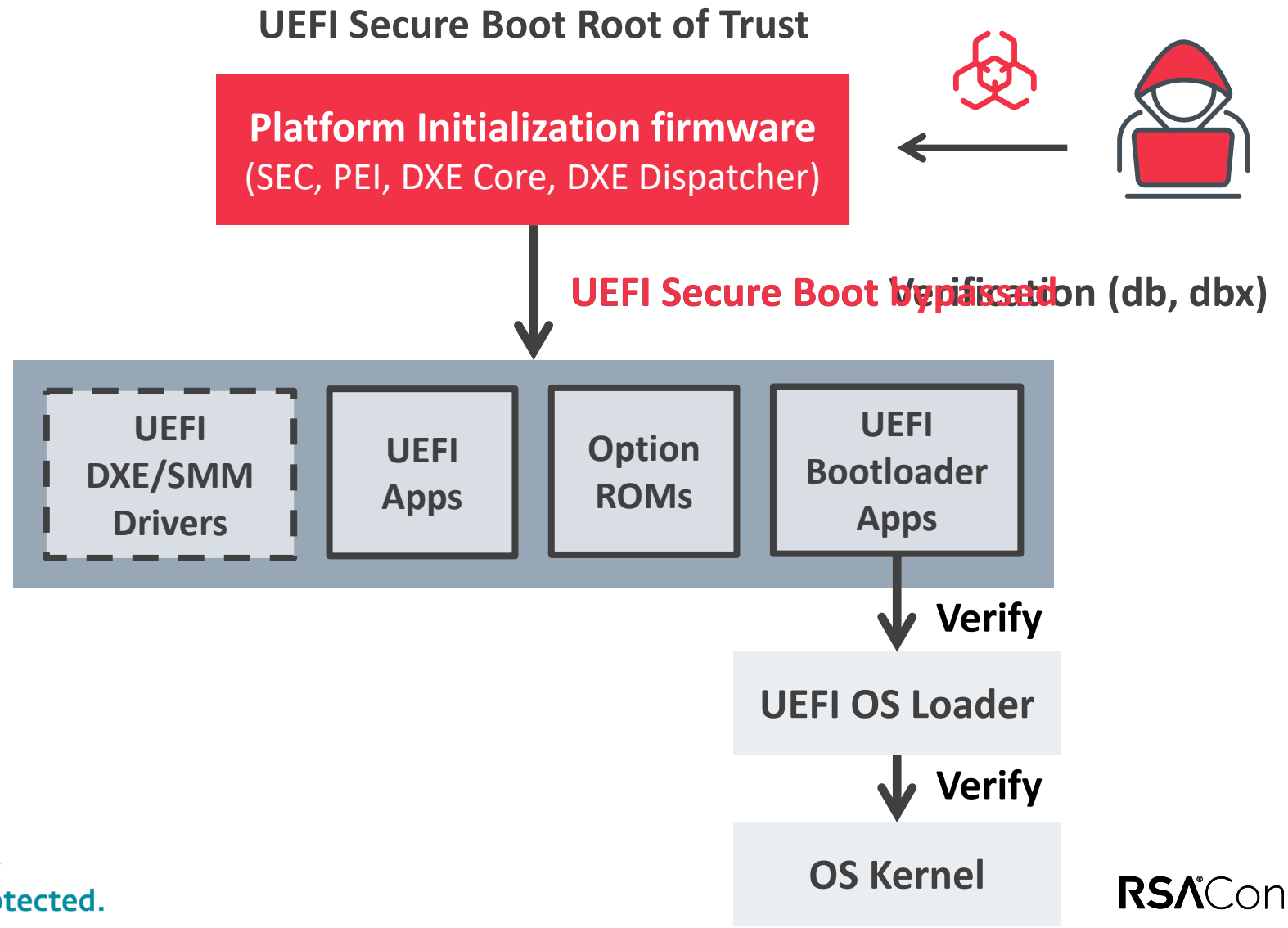
Low confidence

RSAConference2022

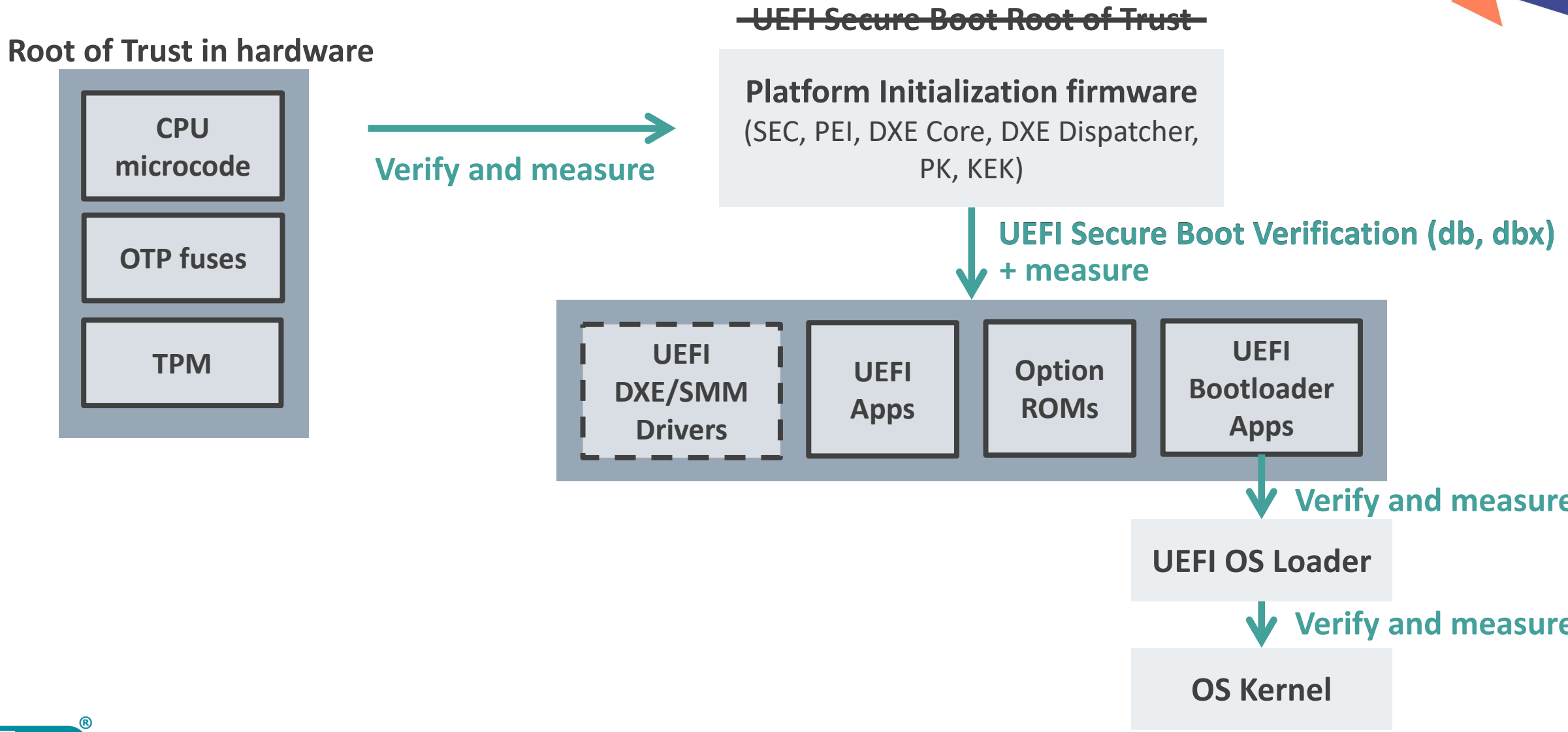
UEFI firmware security real-world problems

When standards meet reality

UEFI Secure Boot



Verified & Measured Boot

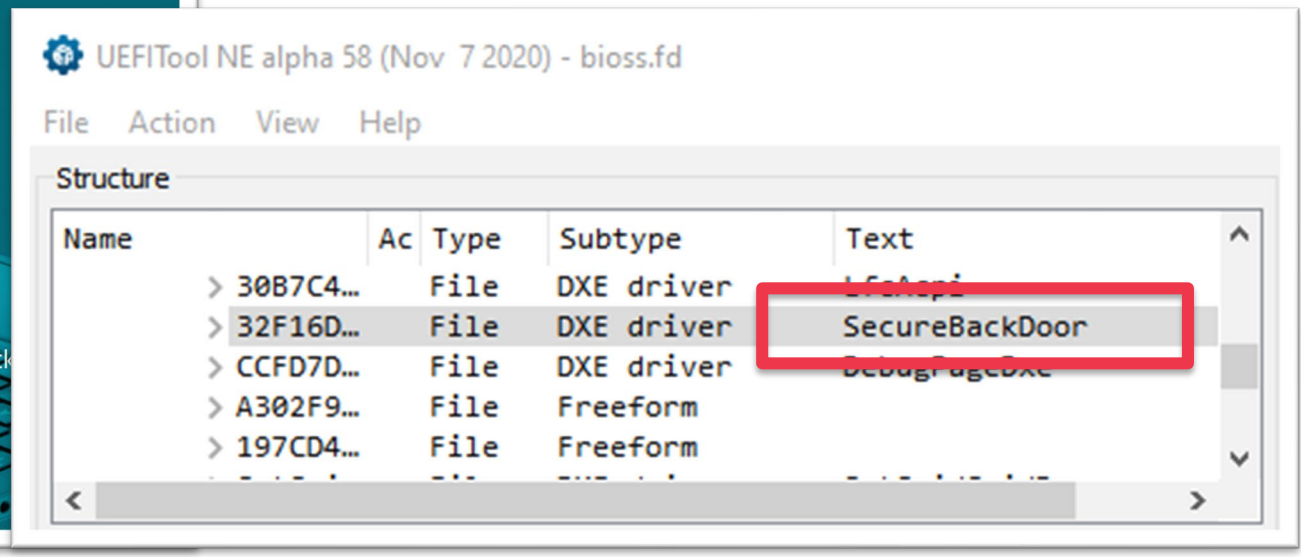


Protections work, BUT!

- World is full of **legacy and low-end devices**
- Misconfigurations
- Vulnerabilities + End Of Development Support = vulnerable forever

Our latest discovery – Lenovo UEFI vulnerabilities

- 100+ Lenovo consumer laptops affected
- Forgotten drivers from manufacturing process



Our latest discovery – Lenovo UEFI Vulnerabilities

- **Activated from the user mode process – just create UEFI Variable**
- CVE-2021-3971 **- disable SPI flash protections**
- CVE-2021-3972 – **disable UEFI Secure Boot** (and more)

Exploit CVE-2021-3971 → Deploy **LoJax/MoonBounce**

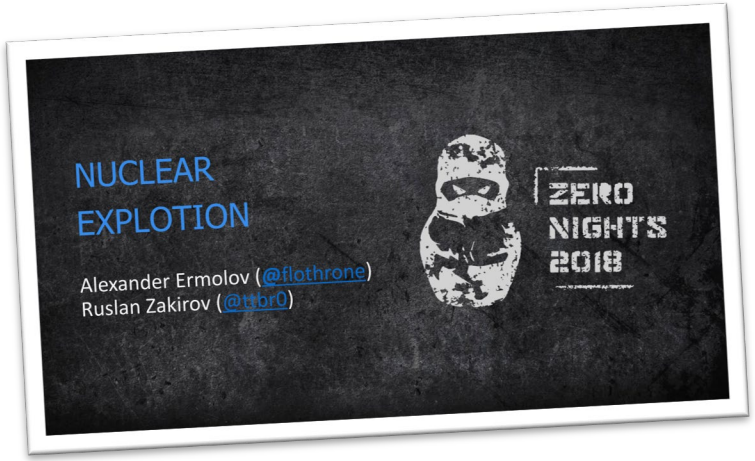
Exploit CVE-2021-3972 → Deploy **ESPecter/FinSpy bootkit**

UEFI firmware vulnerabilities

Another Brick in the Wall: Uncovering SMM Vulnerabilities in HP Firmware

ASSAF CARLSBAD / MARCH 10, 2022

By Assaf Carlsbad & Itai Liba



Repeatable Failures: AMI UsbRt - Six years later, firmware attack vector still affect millions of enterprise devices

Repeatable Firmware Security Failures: 16 High Impact Vulnerabilities Discovered in HP Devices

March 8, 2022

efiXplorer Team

An In-Depth Look at the 23 High-Impact Vulnerabilities

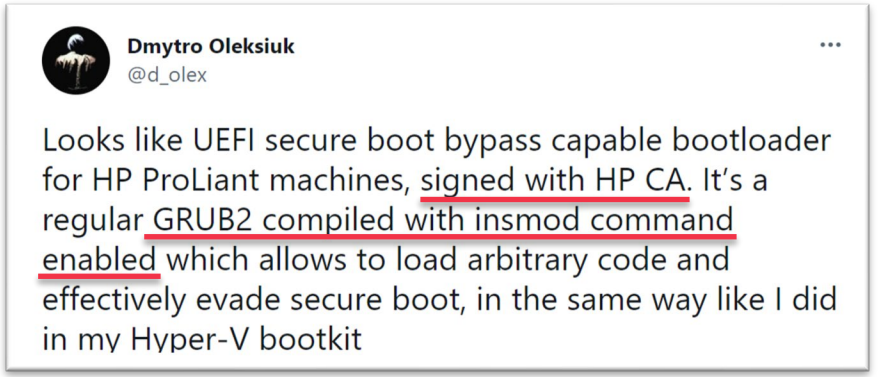
February 1, 2022

efiXplorer Team

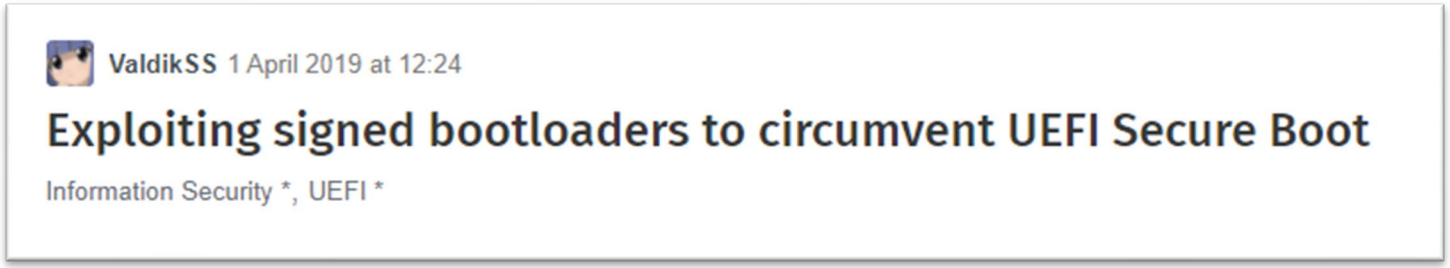
Vulnerable bootloaders – bypassing Secure Boot

[BootHole](#)

[Mickey Shkatov and Jesse Michael](#)



[@d_olex](#)

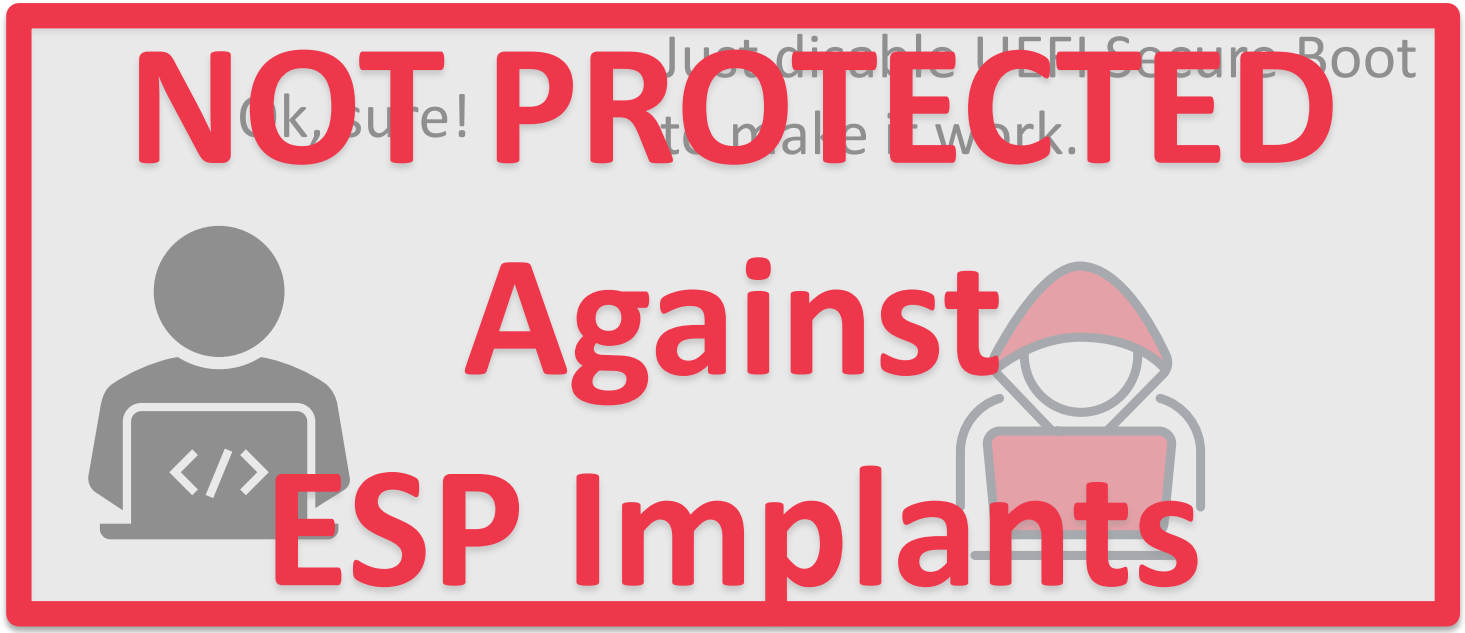


[ValdikSS](#)

The human element – SPI flash vs ESP implants

Once Intel® Boot Guard is **enabled**,
there **MUST be no way to disable** it

The human element – SPI flash vs ESP implants



One can be tricked to disable UEFI Secure Boot

The human element – ESP implants

Firestar

I am now using Windows 11 with Secure Boot disabled and I have no problem with dual-booting Manjaro.

Info:

Today I wanna share a cheat like some years ago where the people share a cheat and it works for more than a week

The cheat is loaded with my **modified version of EFI mapper**
Basically I'm not loading more drivers when I reach to the system and I do the **communication directly with the EFI driver**

General Notes:

When the cheat get finally detected I will probably share the source, **Driver source finally released (not exactly the last version): EFI Driver Access**
Never use public cheat like this one in your main account, every computer could have different detection flags, and you can get ban any time in any moment!
Aimbot button is MOUSE2 (Right Click) and you must hold the Right click!
You can use the injector that you want but must be as manual map and leave the section as rwx
UEFI MODE Windows Required
1* Some of you will need go to the BIOS and Other have the possibility of Boot Menu, Some of you have to enable keyboard bios mode if your keyboard don't work in bios and disable **Secure Boot**

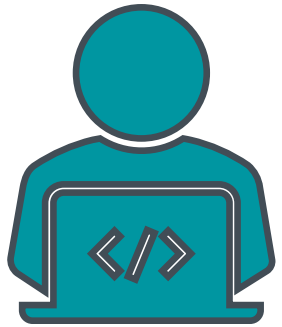
UEFI Bootkits – future of the UEFI threats?

ESP
easy access

+

Powerful threat

+



The human element

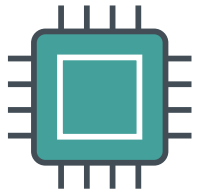
RSAConference2022

Protection & Detection (ESP implants)

Apply
UEFI Secure Boot with TPM auditing

UEFI
Secure Boot **ON**

+



TPM ready

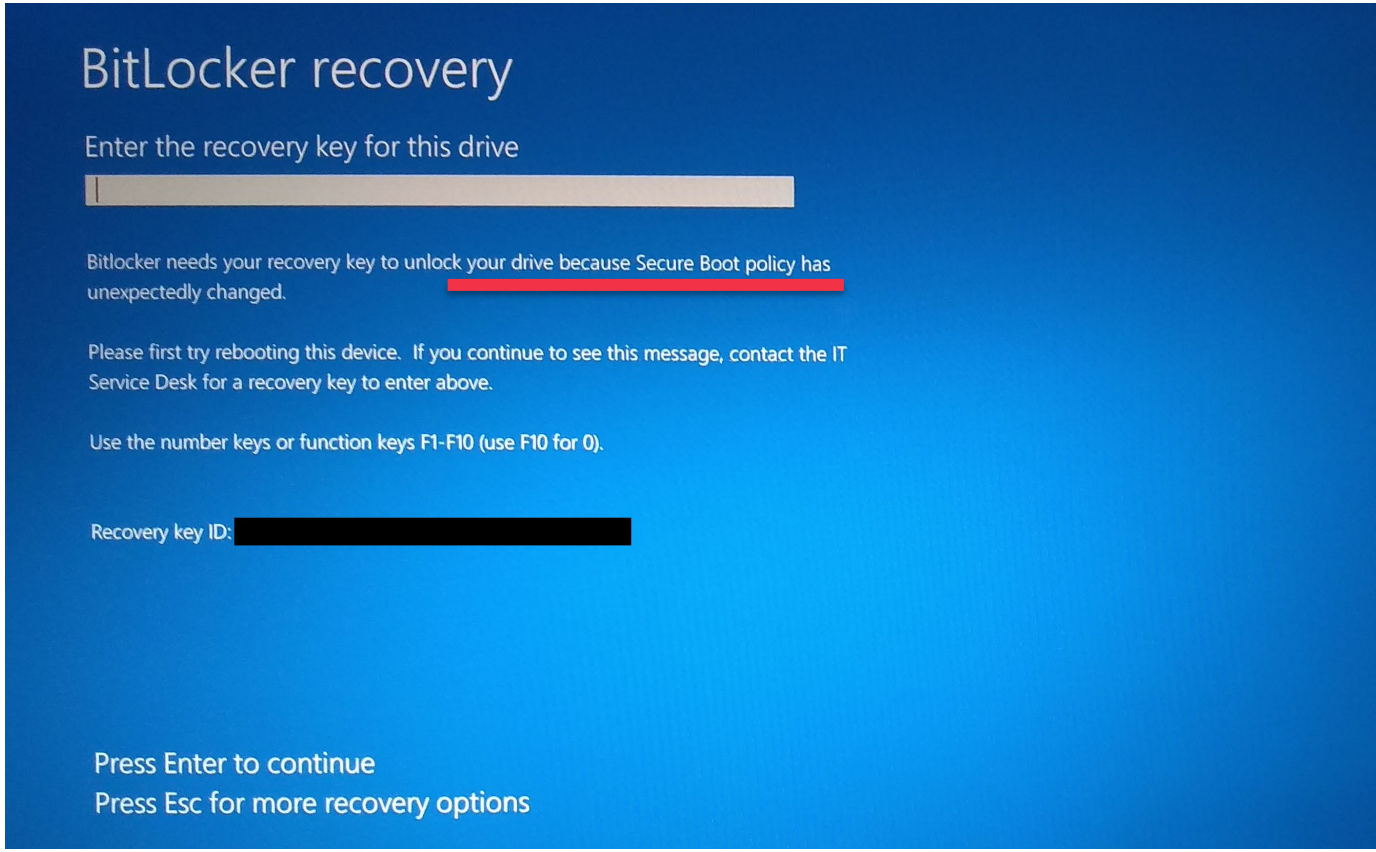
+

Enable Full Disk Encryption
with TPM support
on the Windows volume

Apply
UEFI Secure Boot with TPM auditing

UEFI
Secure Boot **OFF**

Deploy UEFI Bootkit



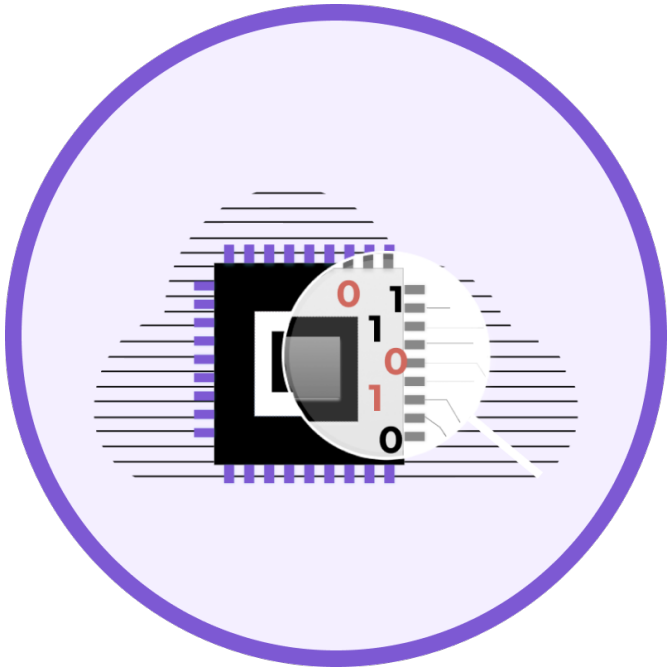
Apply

Defenders – Assessing your platform security

CHIPSEC

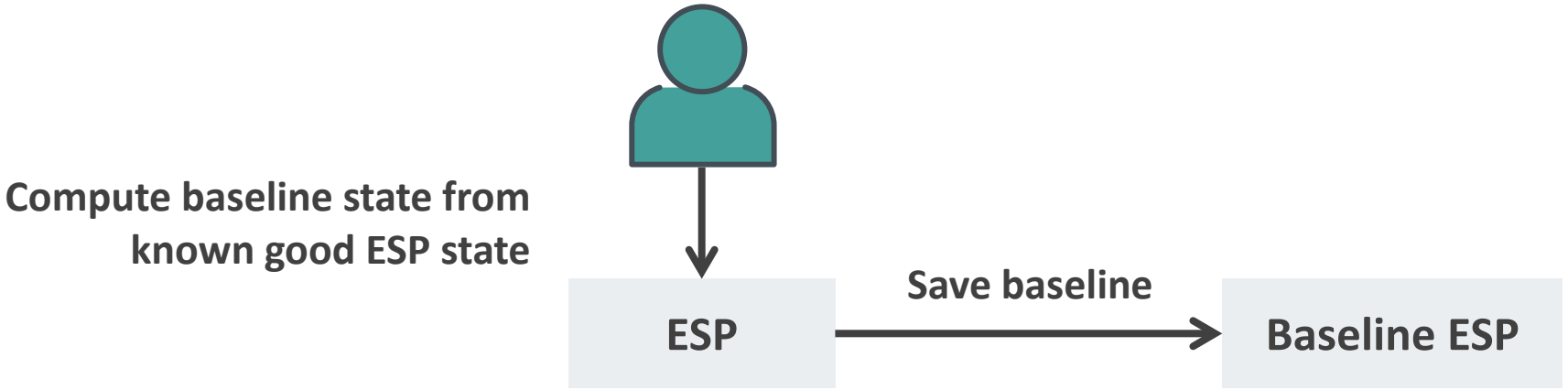
version 1.7.3

Platform Security Assesment Framework



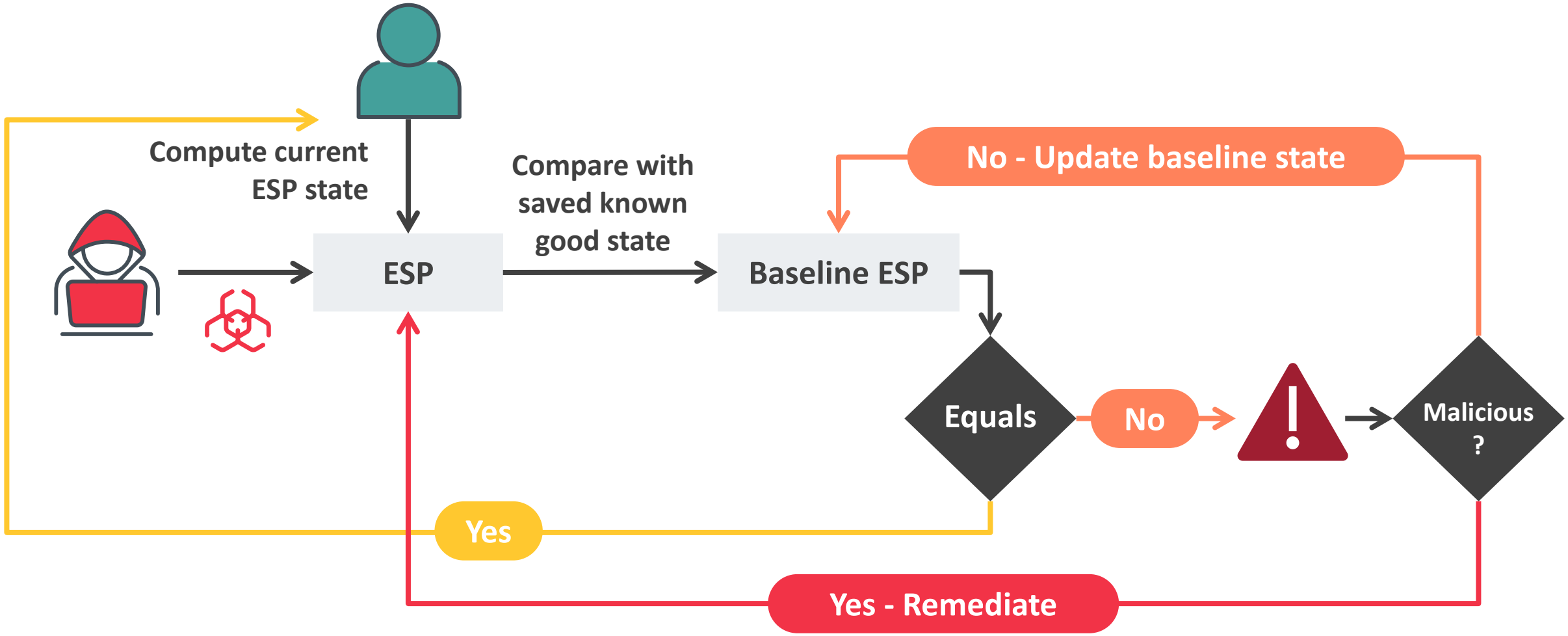
FwHunt & uefi_r2

Apply
Defenders - ESP integrity monitoring

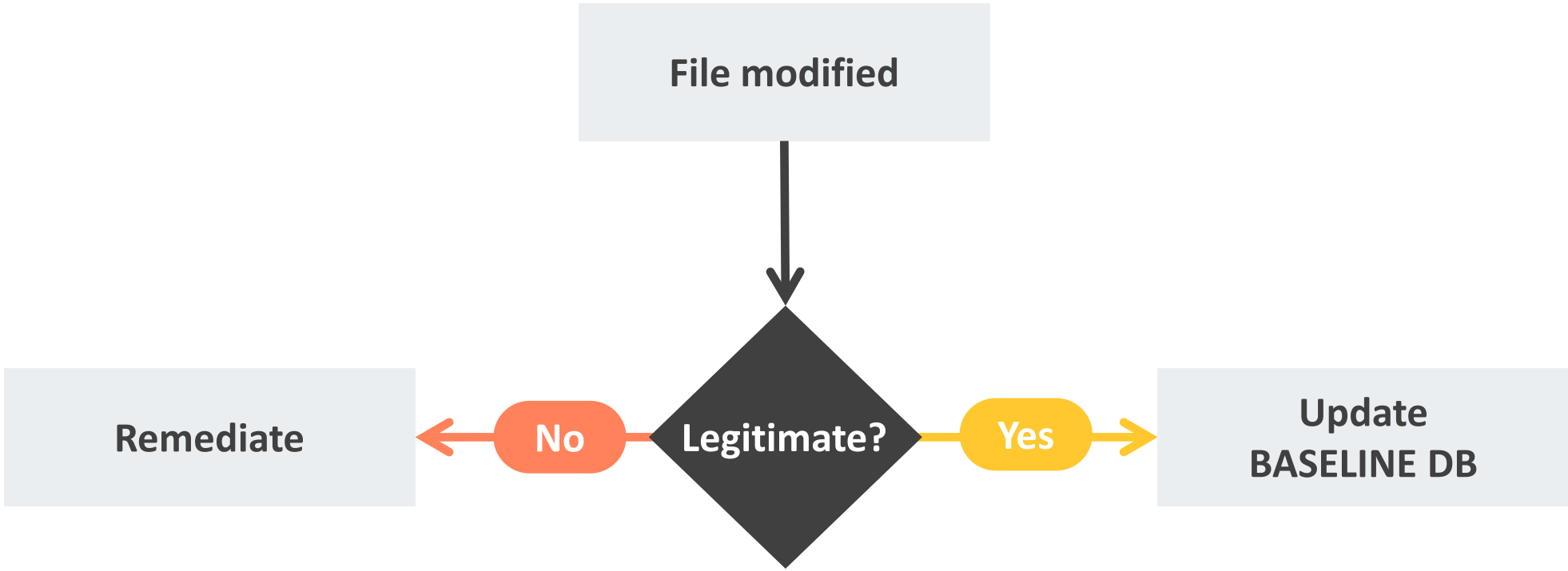


Apply

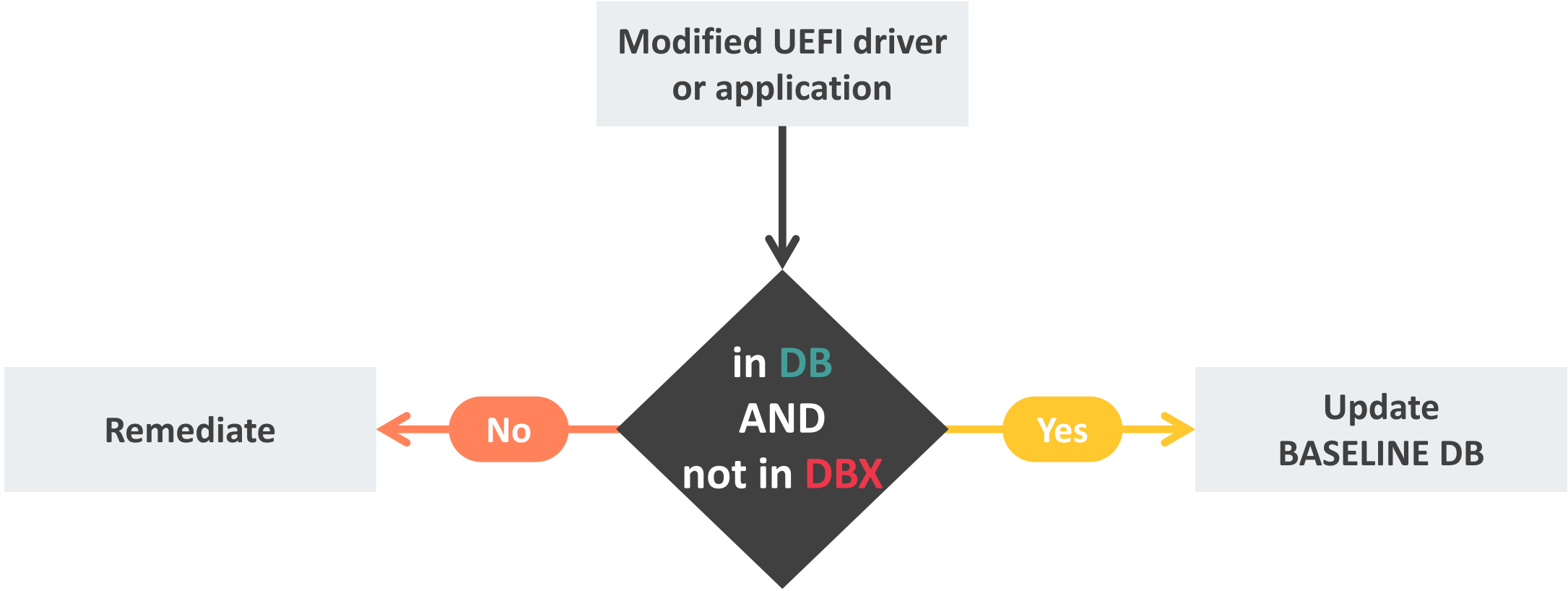
Defenders - ESP integrity monitoring



Apply
Defenders - ESP integrity monitoring



Apply
Defenders - ESP integrity monitoring



Apply

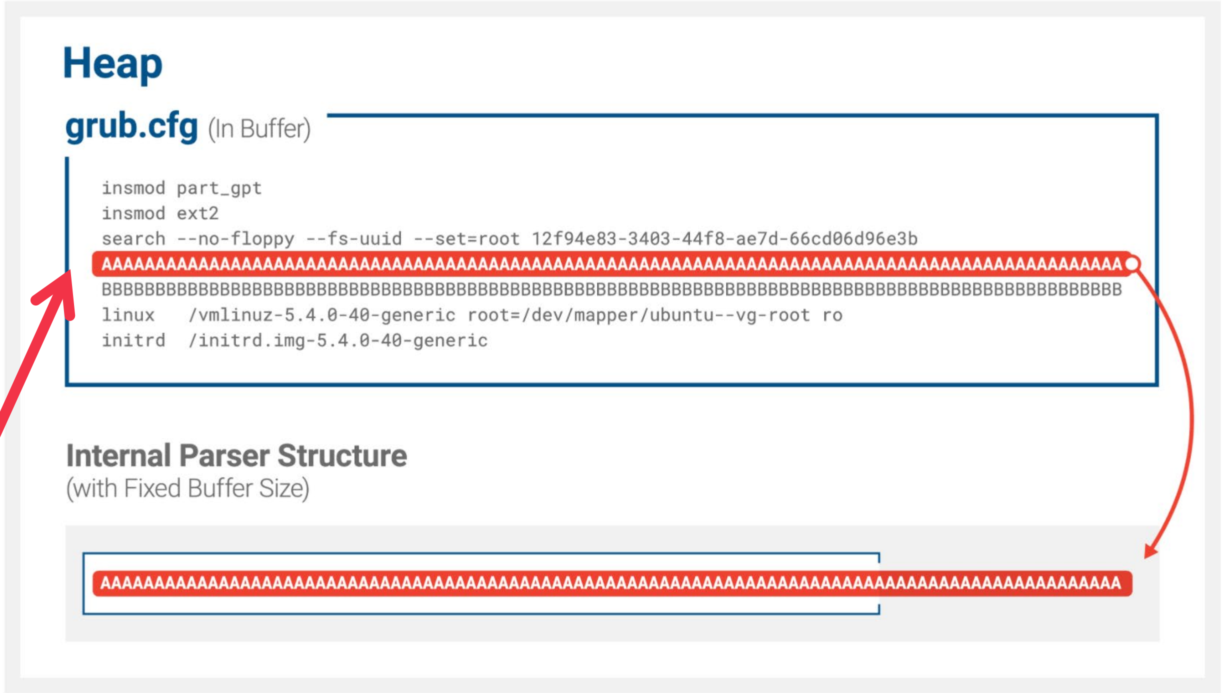
Defenders - ESP integrity monitoring

Legitimate configuration files modification (e.g. grub.cfg)

Load malicious chainloader module

```
rmmod chain
insmod /boot/grub/chain.mod
chainloader /EFI/Microsoft/Boot/bootmgfw_orig.efi /EFI/Microsoft/Boot/backdoor.efi
boot
```

[*Hyper-V Backdoor by @d_olex*]



Look for anomalies

[*BootHole*]

[*Mickey Shkatov and Jesse Michael*]

RSAConference2022

Remediation

Apply
Remediation

ESP

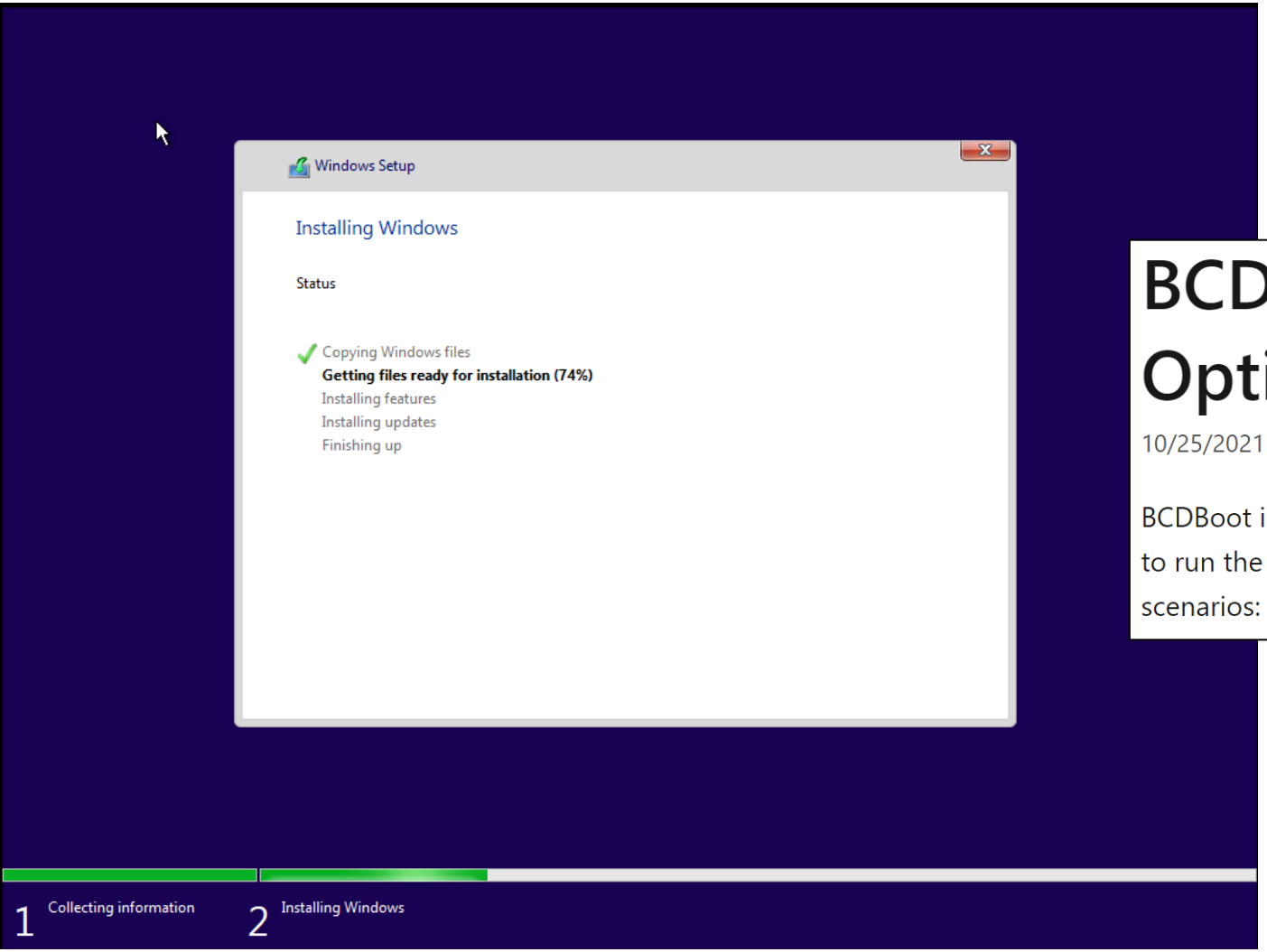
easy to modify

Apply
Remediation

ESP

easy to recover

Apply Remediation



BCDBoot Command-Line Options

10/25/2021 • 7 minutes to read • +1 Is this page helpful?

BCDBoot is a command-line tool used to configure the boot files on a PC or device to run the Windows operating system. You can use the tool in the following scenarios:

Apply
Remediation - ESPecter

Malicious files

- /EFI/Microsoft/Boot/**bootmgfw.efi**
- /EFI/Boot/**bootx64.efi**

Cleanup

1. Boot into Windows recovery command line (from installation DVD/USB)
2. Identify windows installation volume
3. Delete malicious files on ESP
4. Recover boot files with BCDBoot.exe

```
bcdboot C:\Windows /p /v
```

RSAConference2022

Takeaways

Takeaways

SPI implants:
detection is difficult, but so is implementation and deployment

ESP Implant:
easy deployment
Require “only” SecureBoot bypass
Less persistent, but still control the early phase of the boot process

Martin Smolár
Malware Researcher
@smolar_m

Jean-Ian Boutin
Director of Threat Research
@jiboutin

Martin Smolár

Malware Researcher

@smolar_m

Anton Cherepanov

Senior Malware Researcher

@cherepanov74

Jean-Ian Boutin

Director of Threat Research

@jiboutin