

Bootkits: Past, Present & Future

Eugene Rodionov
@vradius

Alexander Matrosov
@matrosov

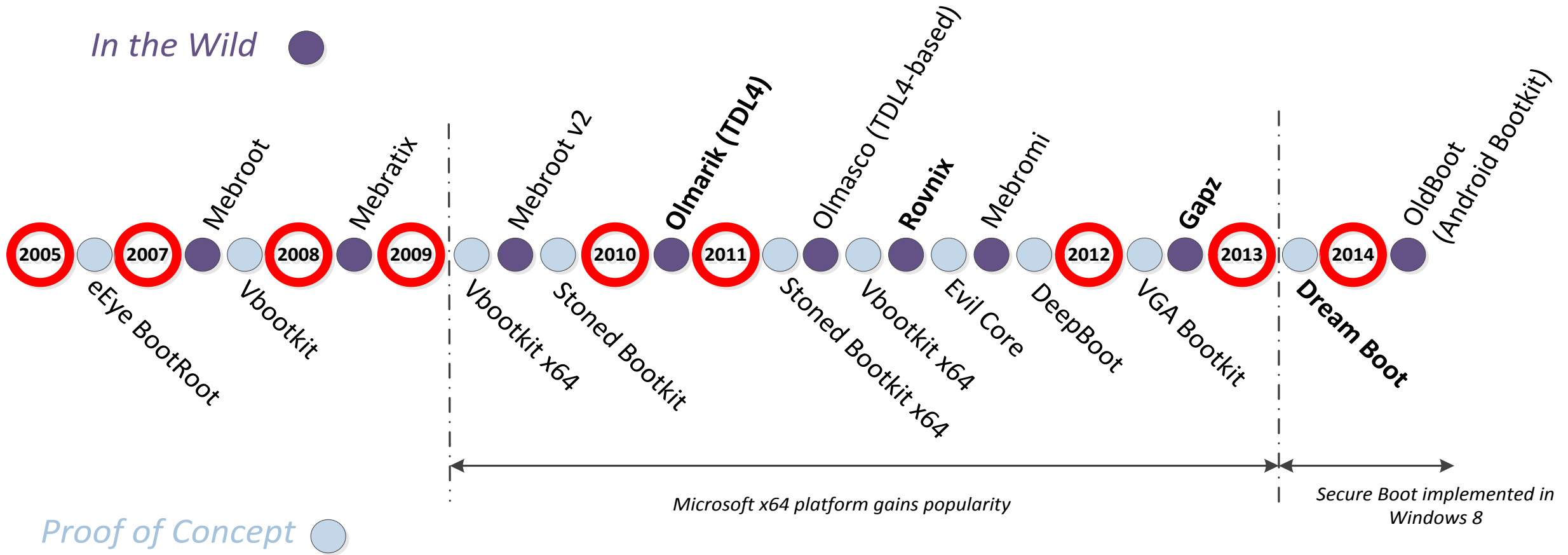
David Harley
@DavidHarleyBlog



Agenda

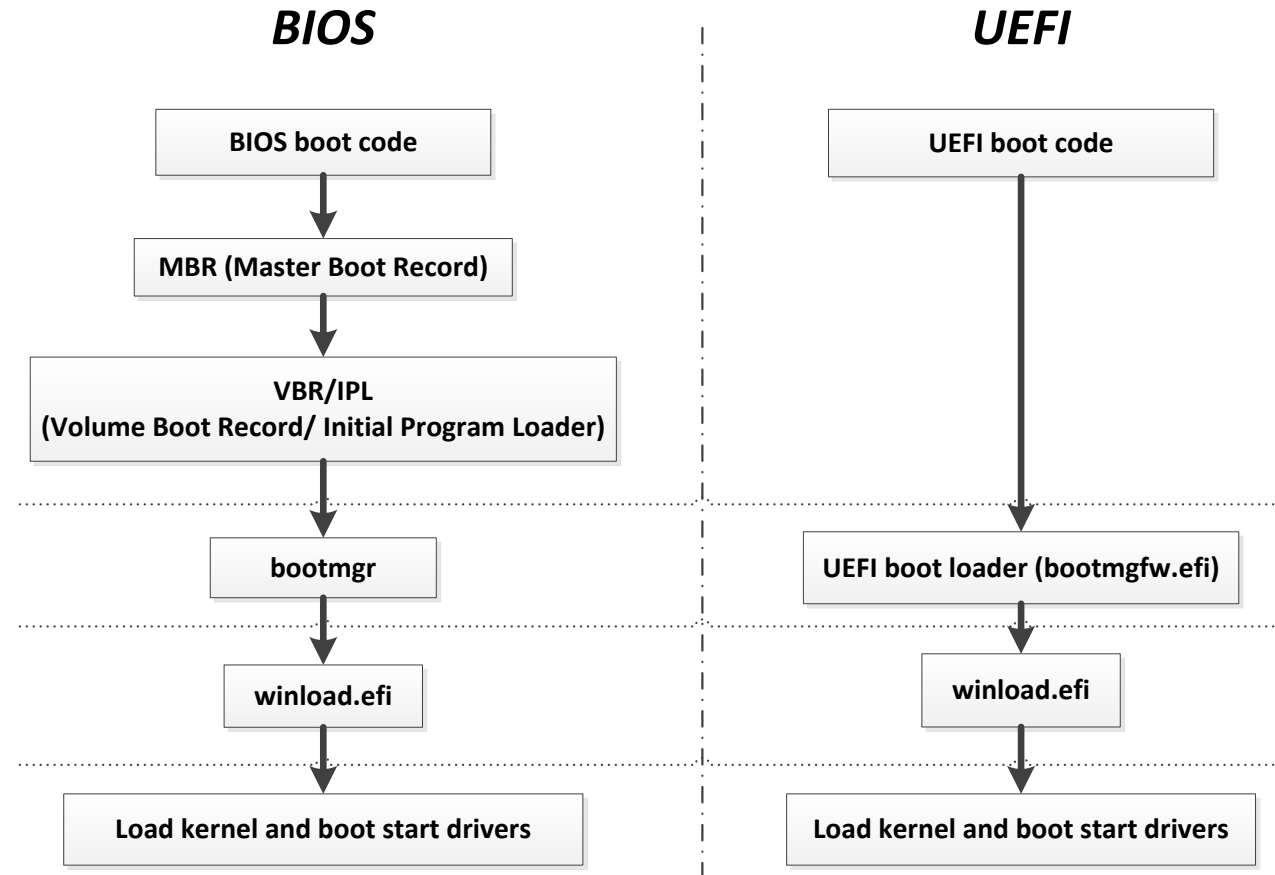
- Modern Bootkits History
 - ✓ Legacy BIOS vs. UEFI Boot Environment & Proof of Concept vs. In the Wild
 - ✓ Legacy BIOS Bootkit Classification
- UEFI Bootkits
 - ✓ Bootkit Implementation Strategies
- Attacks against Secure Boot
- Forensic Software
 - ✓ HiddenFsReader
 - ✓ CHIPSEC

Modern Bootkit History

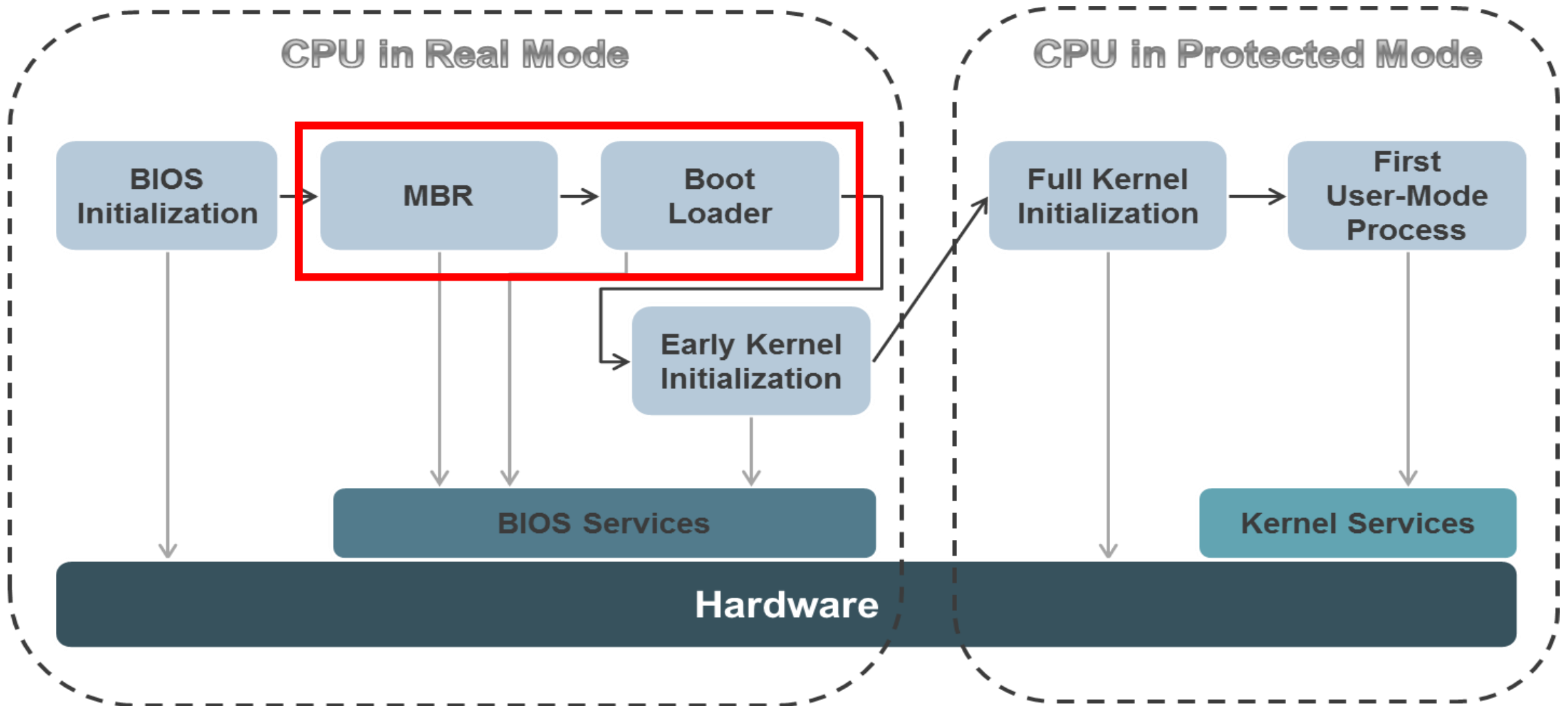


Legacy BIOS vs. UEFI

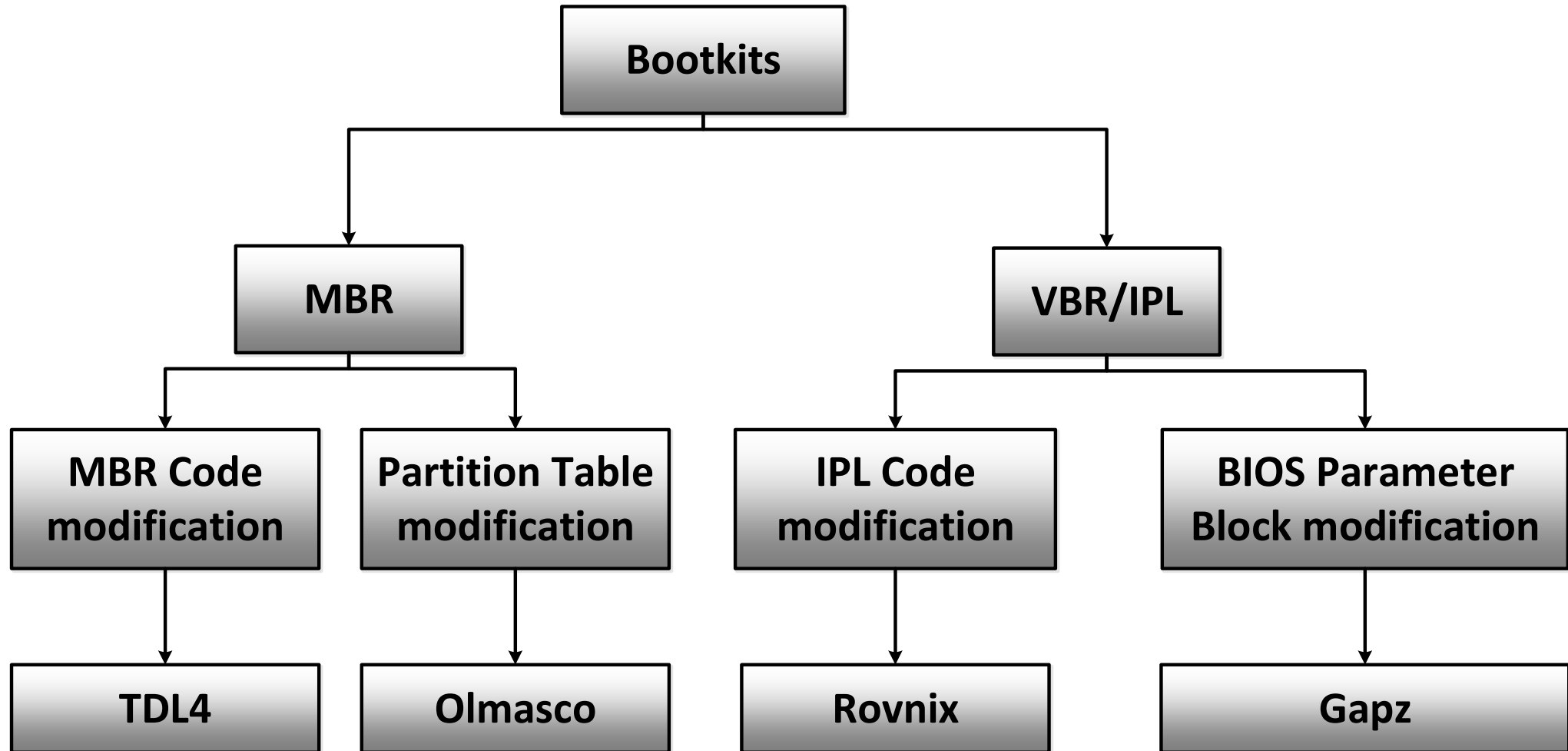
- No more MBR and VBR/IPL code
- Different hard drive partitioning scheme: GPT (GUID Partition Table)
- Secure Boot technology is implemented in Windows 8



The Target of Modern Bootkits (MBR/VBR)

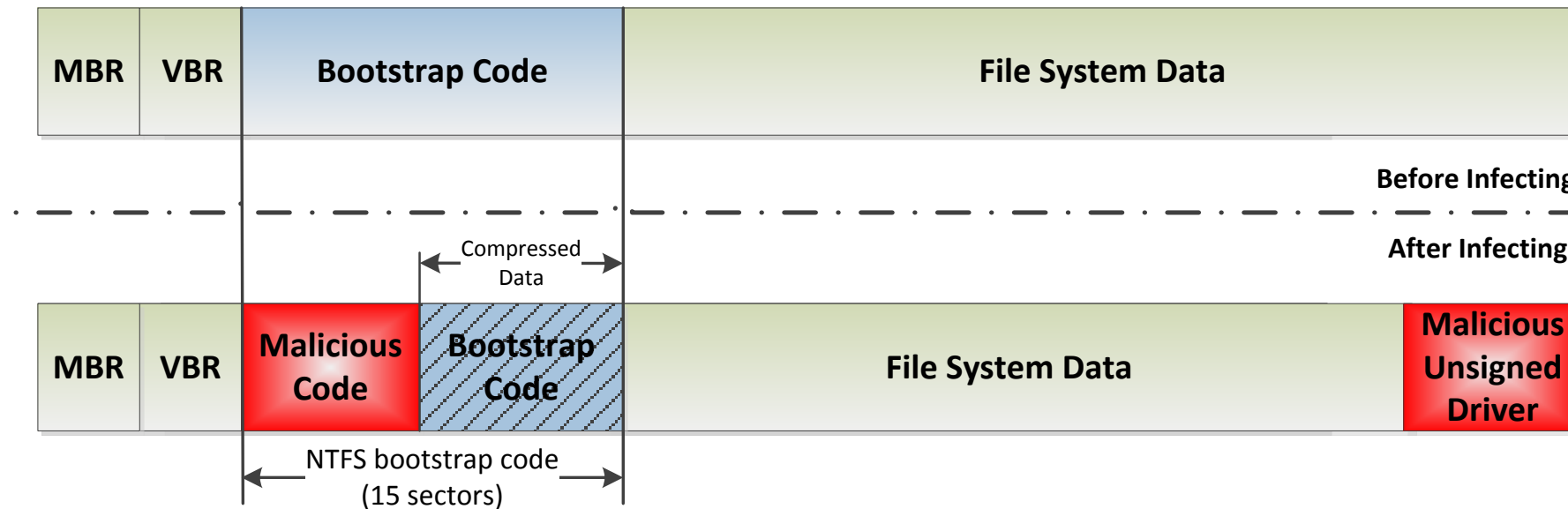


Classification of MBR/VBR Bootkits



IPL Code Modification: Rovnix

- Win64/Rovnix overwrites bootstrap code of the active partition



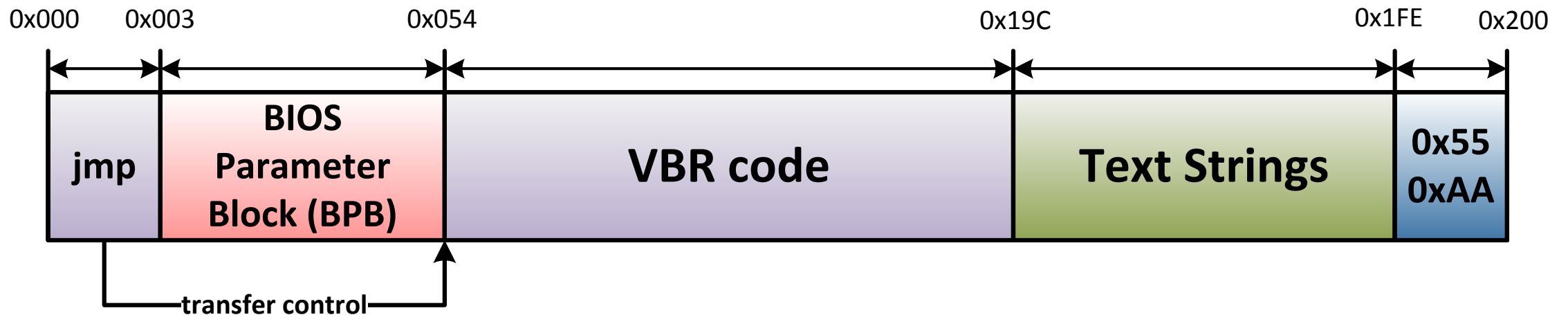
"Hasta La Vista, Bootkit: Exploiting the VBR"

<http://www.welivesecurity.com/2011/08/23/hasta-la-vista-bootkit-exploiting-the-vbr/>

Gapz VBR Bootkit

Main features:

- Relies on Microsoft Windows VBR layout
- The infections result in modifying only 4 bytes of VBR
- The patched bytes might differ on various installations



“Mind the Gapz: The most complex bootkit ever analyzed?”

<http://www.welivesecurity.com/wp-content/uploads/2013/04/gapz-bootkit-whitepaper.pdf>

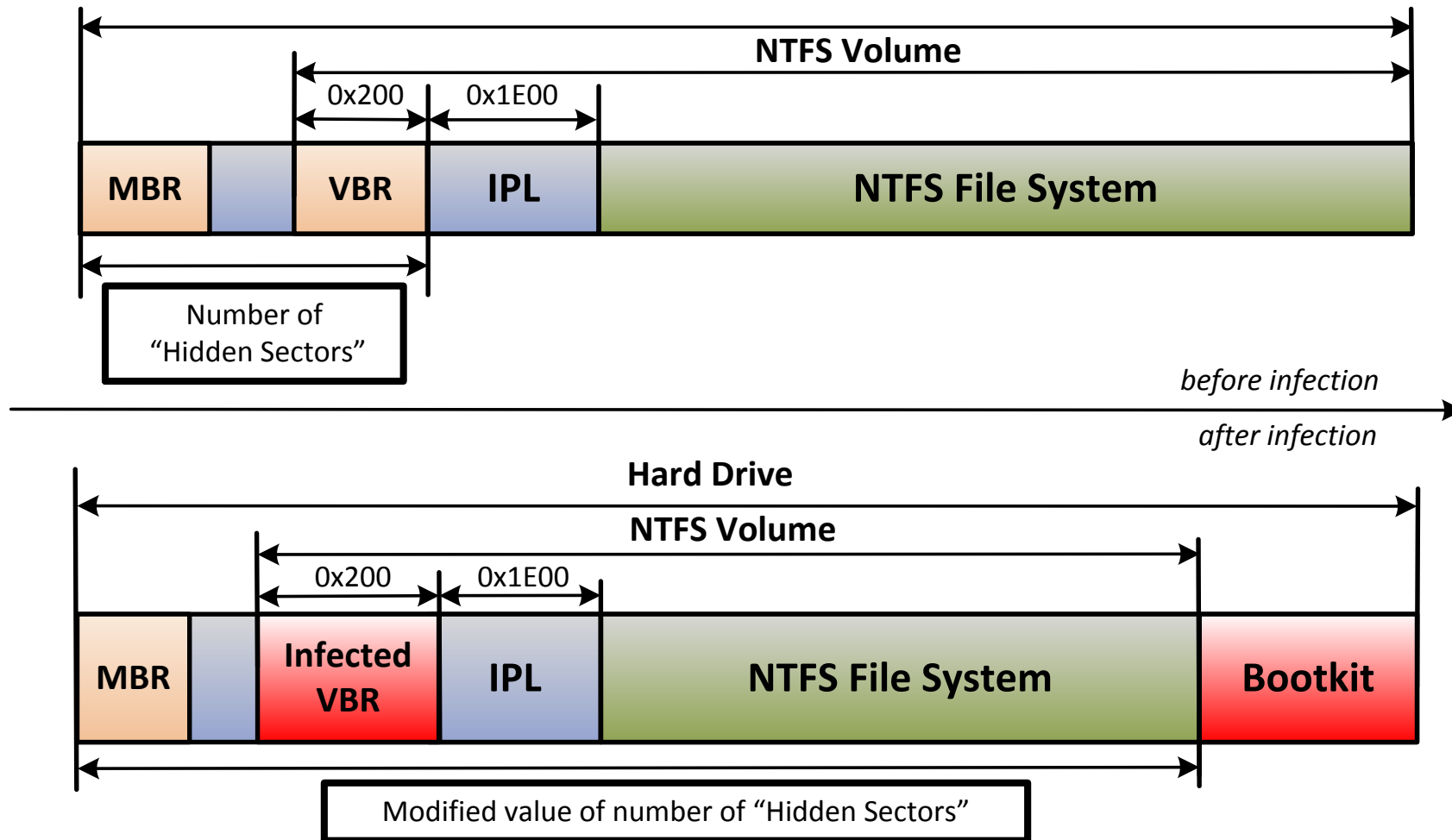
Gapz BPB Layout

00000000:	EB	52	90	4E-54	46	53	20-20	20	20	00-02	08	00	00
00000010:	00	00	00	00-00	F8	00	00-3F	00	FF	00-00	08	00	00
00000020:	00	00	00	00-80	00	80	00-FF	1F	03	00-00	00	00	00
00000030:	55	21	00	00-00	00	00	00-02	00	00	00-00	00	00	00
00000040:	F6	00	00	00-01	00	00	00-E6	94	34	C6-AD	34	C6	50
00000050:	00	00	00	00-FA	33	C0	8E-D0	BC	00	7C-FB	68	C0	07
00000060:	1F	1E	68	66-00	CB	88	16-0E	00	66	81-3E	03	00	4E
00000070:	54	46	53	75-15	B4	41	BB-AA	55	CD	13-72	0C	81	FB
00000080:	55	AA	75	06-F7	C1	01	00-75	03	E9	DD-00	1E	83	EC
00000090:	18	68	1A	00-B4	48	8A	16-0E	00	8B	F4-16	1F	CD	13
000000A0:	9F	83	C4	18-9E	58	1F	72-E1	3B	06	0B-00	75	DB	A3
000000B0:	0F	00	C1	2E-0F	00	04	1E-5A	33	DB	B9-00	20	2B	C8
000000C0:	66	FF	06	11-00	03	16	0F-00	8E	C2	FF-06	16	00	E8
000000D0:	4B	00	2B	C8-77	EF	B8	00-BB	CD	1A	66-23	C0	75	2D
000000E0:	66	81	FB	54-43	50	41	75-24	81	F9	02-01	72	1E	16
000000F0:	68	07	BB	16-68	70	0E	16-68	09	00	66-53	66	53	66
00000100:	55	16	16	16-68	B8	01	66-61	0E	07	CD-1A	33	C0	BF
00000110:	28	10	B9	D8-0F	FC	F3	AA-E9	5F	01	90-90	66	60	1E
00000120:	06	66	A1	11-00	66	03	06-1C	00	1E	66-68	00	00	00
00000130:	00	66	50	06-53	68	01	00-68	10	00	B4-42	8A	16	0E
00000140:	00	16	1F	8B-F4	CD	13	66-59	5B	5A	66-59	66	59	1F
00000150:	0F	82	16	00-66	FF	06	11-00	03	16	0F-00	8E	C2	FF
00000160:	0E	16	00	75-BC	07	1F	66-61	C3	A0	F8-01	E8	09	00
00000170:	A0	FB	01	E8-03	00	F4	EB-FD	B4	01	8B-F0	AC	3C	00
00000180:	74	09	B4	0E-BB	07	00	CD-10	EB	F2	C3-0D	0A	41	20
00000190:	64	69	73	6B-20	72	65	61-64	20	65	72-72	6F	72	20
000001A0:	6F	63	63	75-72	72	65	64-00	0D	0A	42-4F	4F	54	4D
000001B0:	47	52	20	69-73	20	6D	69-73	73	69	6E-67	00	0D	0A
000001C0:	42	4F	4F	54-4D	47	52	20-69	73	20	63-6F	6D	70	72
000001D0:	65	73	73	65-64	00	0D	0A-50	72	65	73-73	20	43	74
000001E0:	72	6C	2B	41-6C	74	2B	44-65	6C	20	74-6F	20	72	65
000001F0:	73	74	61	72-74	0D	0A	00-8C	A9	BE	D6-00	00	55	AA
00000200:	07	00	42	00-4F	00	4F	00-54	00	4D	00-47	00	52	00
00000210:	04	00	24	00-49	00	33	00-30	00	00	D4-00	00	00	24
00000220:	??	??	??	??-??	??	??	??-??	??	??	??-??	??	??	??

HiddenSectors field
of BPB

VBR of the
active partition

Gapz



Functionality	Gapz	Olmarik (TDL4)	Rovnix (Cidox)	Goblin (XPAJ)	Olmasco (MaxSS)
MBR modification	✓	✓	✗	✓	✓
VBR modification	✓	✗	✓	✗	✗
Hidden file system type	FAT32	custom	FAT16 modification	custom (TDL4 based)	custom
Crypto implementation	AES-256, RC4, MD5, SHA1, ECC	XOR/RC4	Custom (XOR+ROL)	✗	RC6 modification
Compression algorithm	✓	✗	aPlib	aPlib	✗
Custom TCP/IP network stack implementation	✓	✗	✓	✗	✗

HiddenFsReader as a Forensic Tool (MBR/VBR)



```
ESET Hidden File System Reader
1.0.3.1 (Apr 30 2013 16:31:34)
Copyright (c) 1992-2013 ESET, spol. s r.o. All rights reserved.

Processing... Please wait.
Parsing file systems...

"Gapz_VBR" file system found:
- vbr_original          md5: 32E746BECCA5C4CC2511CABFFE6B7310
- payload.bin           md5: 9DCFE30C707B0941EEECF51DA2DBBAA0
- cfg                   md5: 3DC93A2466B881E24912DCCF839FC4C8
- his                   md5: DF739CC8AA796A24FF10E57894F8864C
- overlord32.dll        md5: 3AEC40DE15B791B2DFA978DEDE7B0C89
- overlord64.dll        md5: F5358444F57E2849C73D9DD14EBB4FA4
- conf.z                md5: 7215EE9C7D9DC229D2921A40E899EC5F
- e59df022              md5: 74D9434F39779CB608D48D773F627287
- vbr_infected          md5: 115AB3FD466BEE136DE25A6CEB46E54C

File system(s) successfully exported!
```

HiddenFsReader as a Forensic Tool (MBR/VBR)

```
ESET Hidden File System Reader
1.0.3.1 <Apr 30 2013 16:31:34>

Copyright <c> 1992-2013 ESET, spol. s r.o. All rights reserved.

HfsReader.exe [params] [export_path]

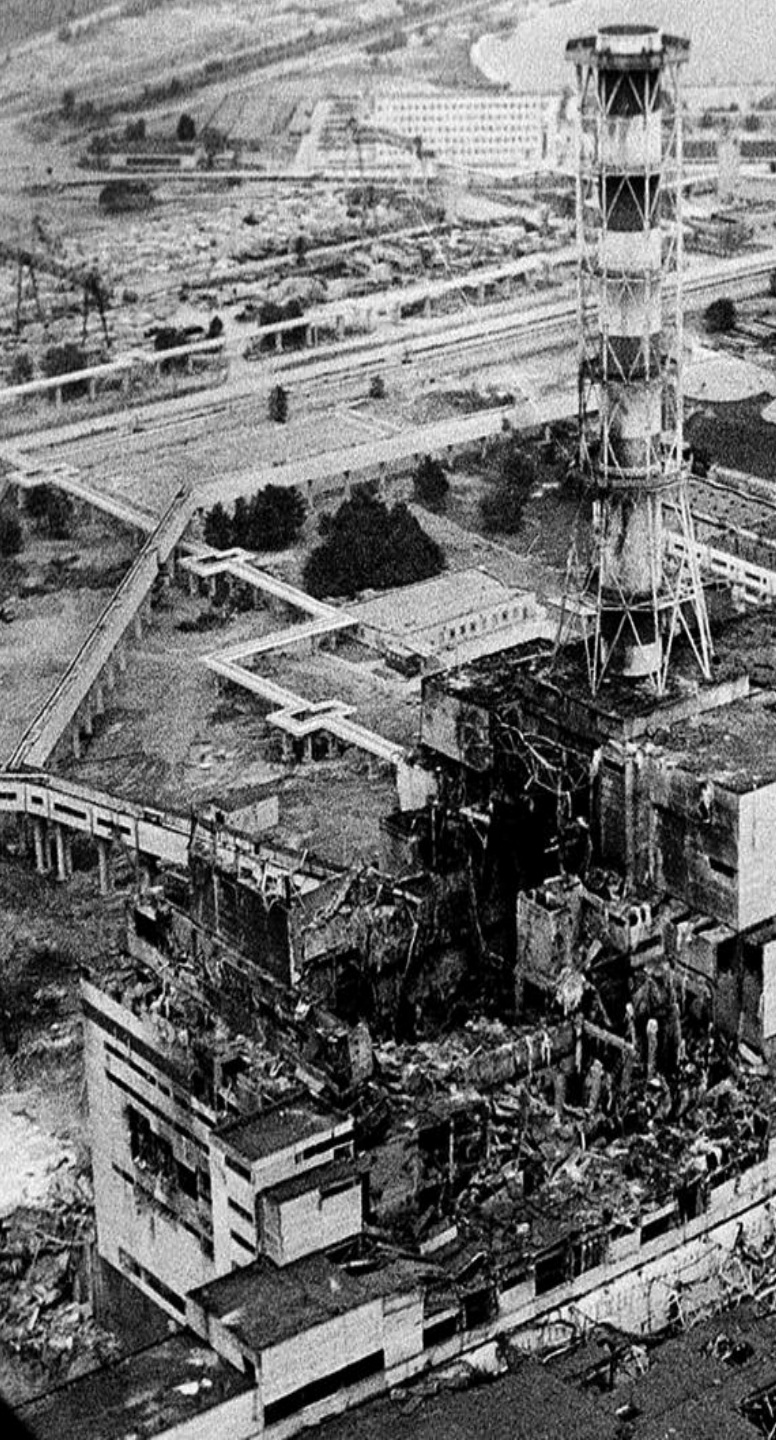
Params:
/help or /?      - print help message
/no-output       - no output to command line
/no-export       - do not export files from file system(s)
/export-txt      - export file list from file system(s) to text file
/mbr            - make mbr dump
/vbr            - make active drive vbr dump
/dump=<o>,<s>    - make hard drive dump
                  <o> - offset from beginning or "end"
                  <s> - size
                  Examples:
                      /dump=512,1024
                      /dump=end,4096
/zip             - pack all files into zip archive
/full           - create full analysis and pack results into zip archive

Supported Hidden File Systems:
Win32/Olmarik <TDL3/TDL3+/TDL4>
Win32/Olmasco <MaxXSS>
Win32/Sirefef <ZeroAccess>
Win32/Rovnix
Win32/Xpaj
Win32/Gapz
Win32/Flamer
Win32/Urelas <GBPBoot>
```

 Boot Priority



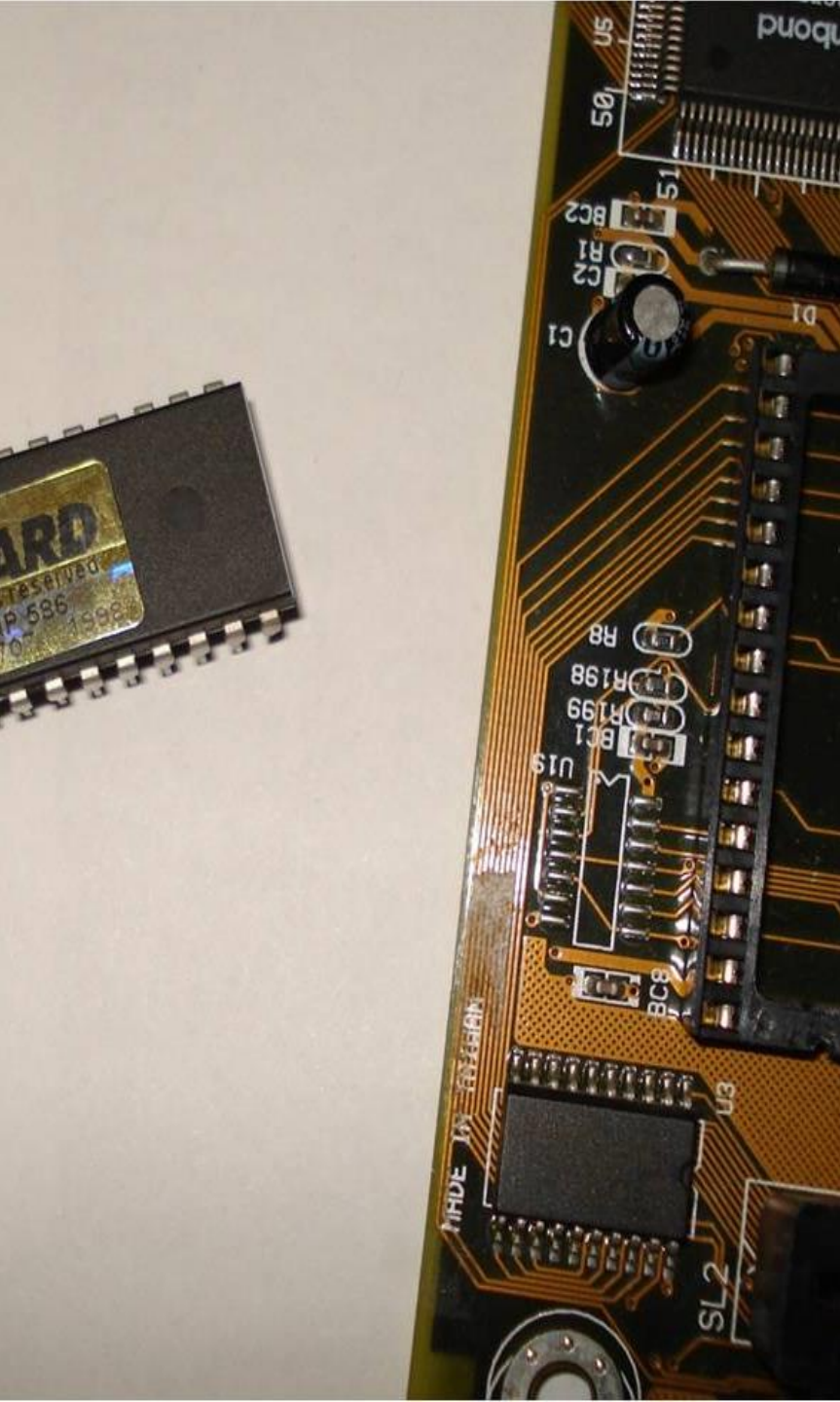
Then World Moved to UEFI..



In The Beginning...

In 1998-99 **CIH** (Chernobyl) **virus**
written by a student of Taipei Tatung
Institute of Technology in Taiwan
infected **~60 million** PCs

CIH (Chernobyl) **erased BIOS 'ROM'** boot
block and boot sectors on a hard drive
causing **~1B US dollars** in damage



Signed BIOS Updates Are Rare

- [Mebromi](#) malware includes BIOS infector & MBR bootkit components
- Patches BIOS ROM binary injecting malicious ISA Option ROM with legitimate BIOS image mod utility
- Triggers SW SMI 0x29/0x2F to erase SPI flash then write patched BIOS binary

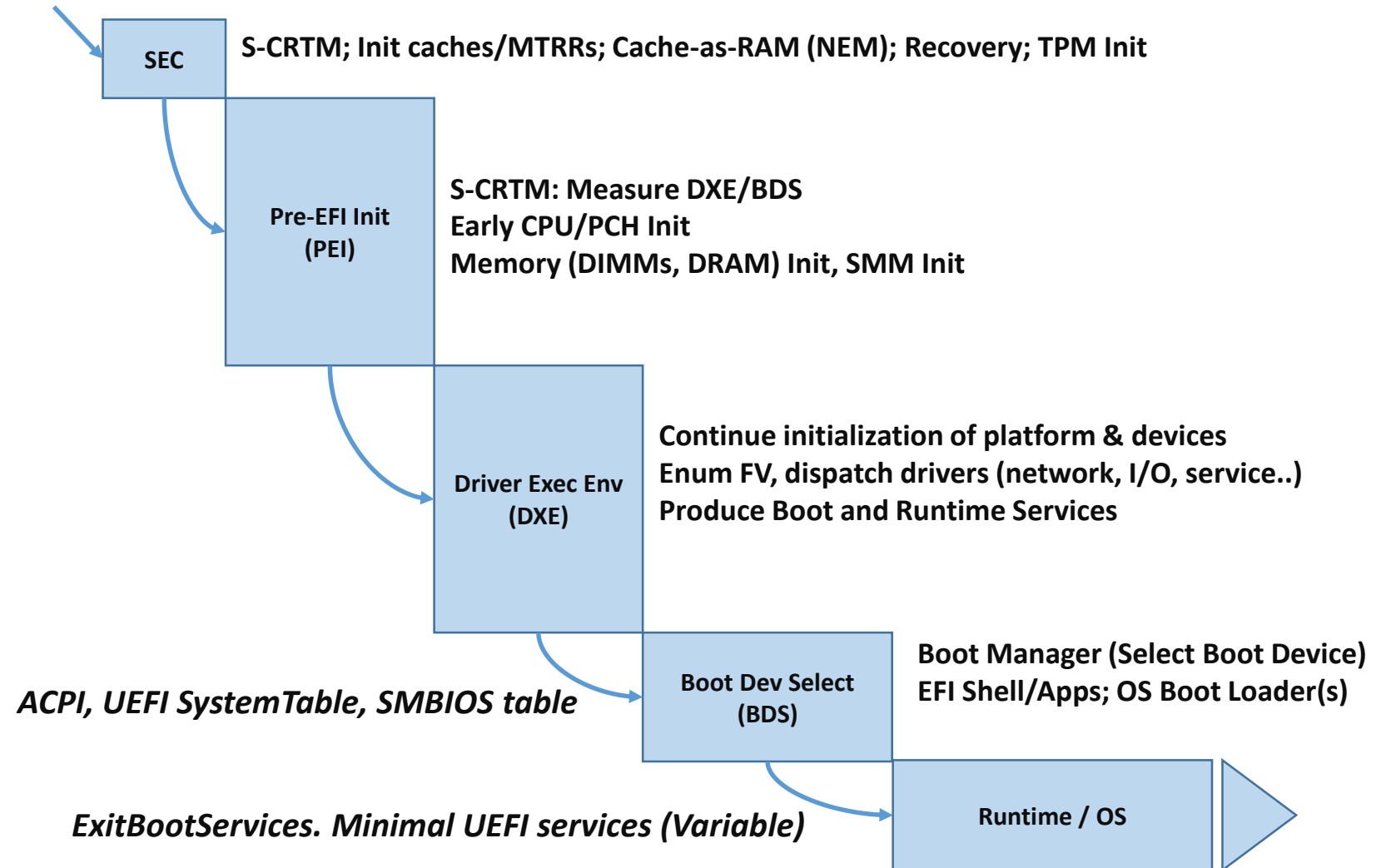
No Signature Checks of OS boot loaders (MBR/VBR)

- No concept of Secure or Verified Boot
- Wonder why **TDL4** and likes flourished?

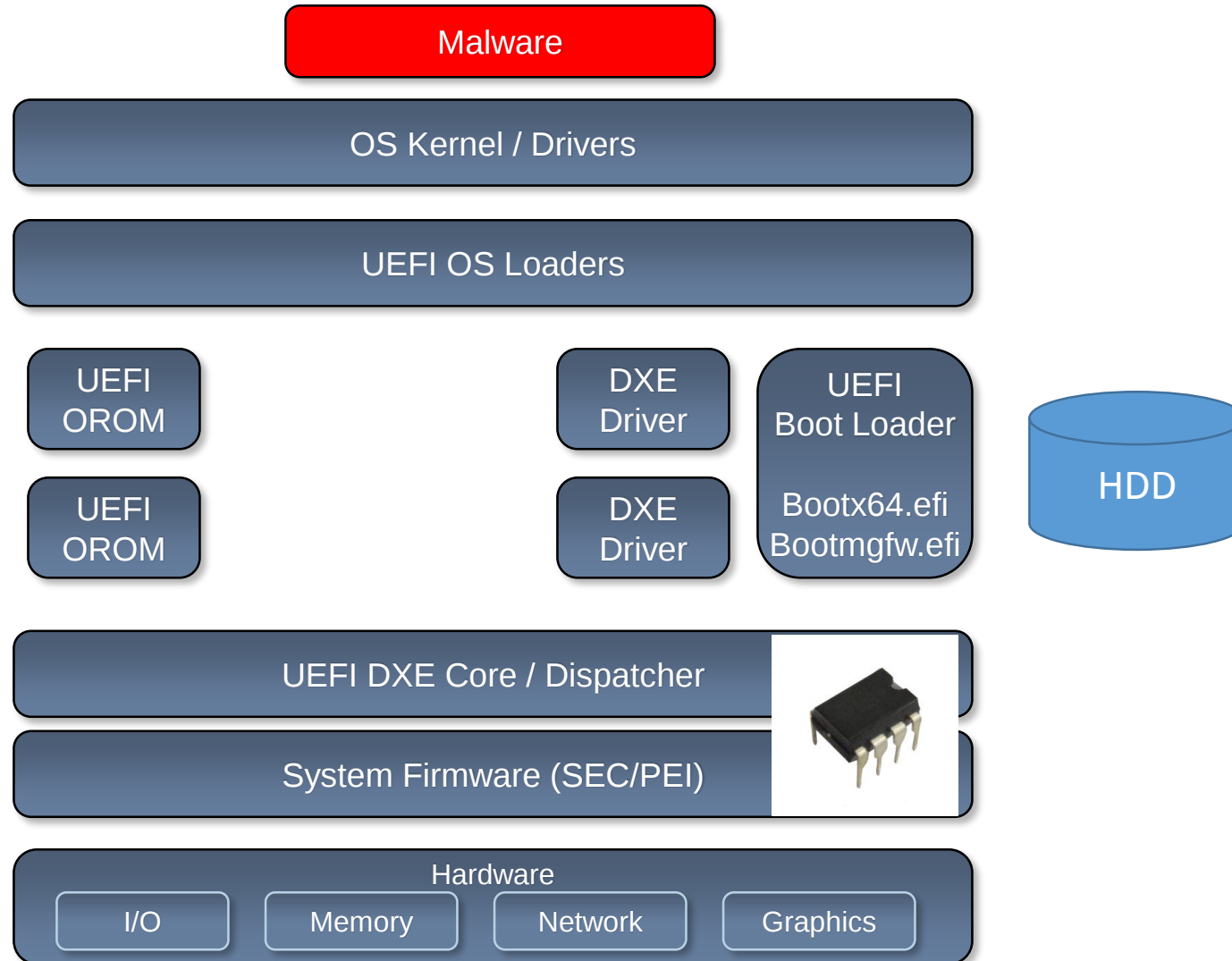


UEFI BIOS Firmware

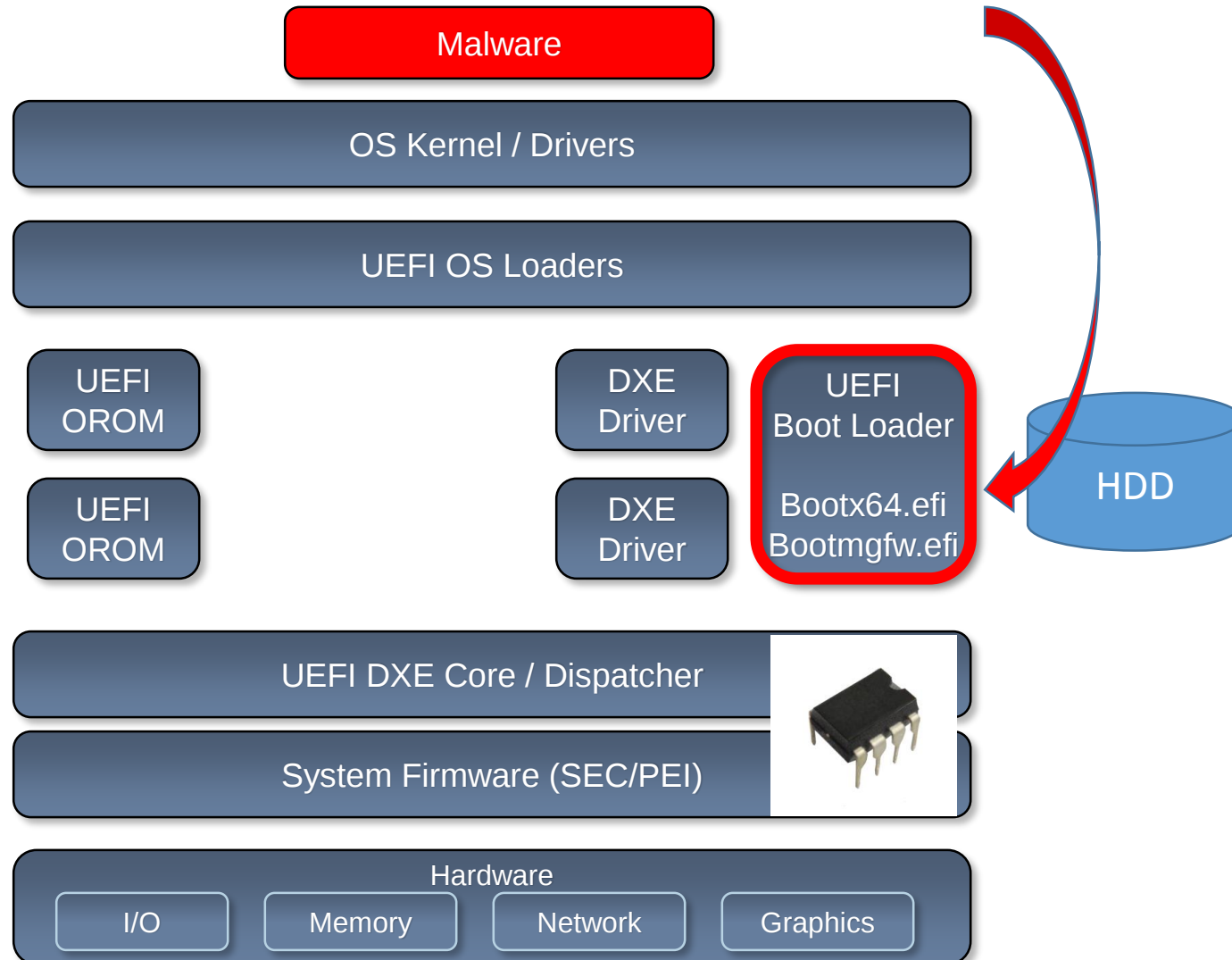
CPU Reset



UEFI Bootkits



UEFI Bootkits



UEFI Bootkits

Replacing Windows Boot Manager

EFI System Partition (ESP) on Fixed Drive

ESP\EFI\Microsoft\Boot\bootmgfw.efi

[UEFI technology: say hello to the Windows 8 bootkit!](#) by ITSEC

Replacing Fallback Boot Loader

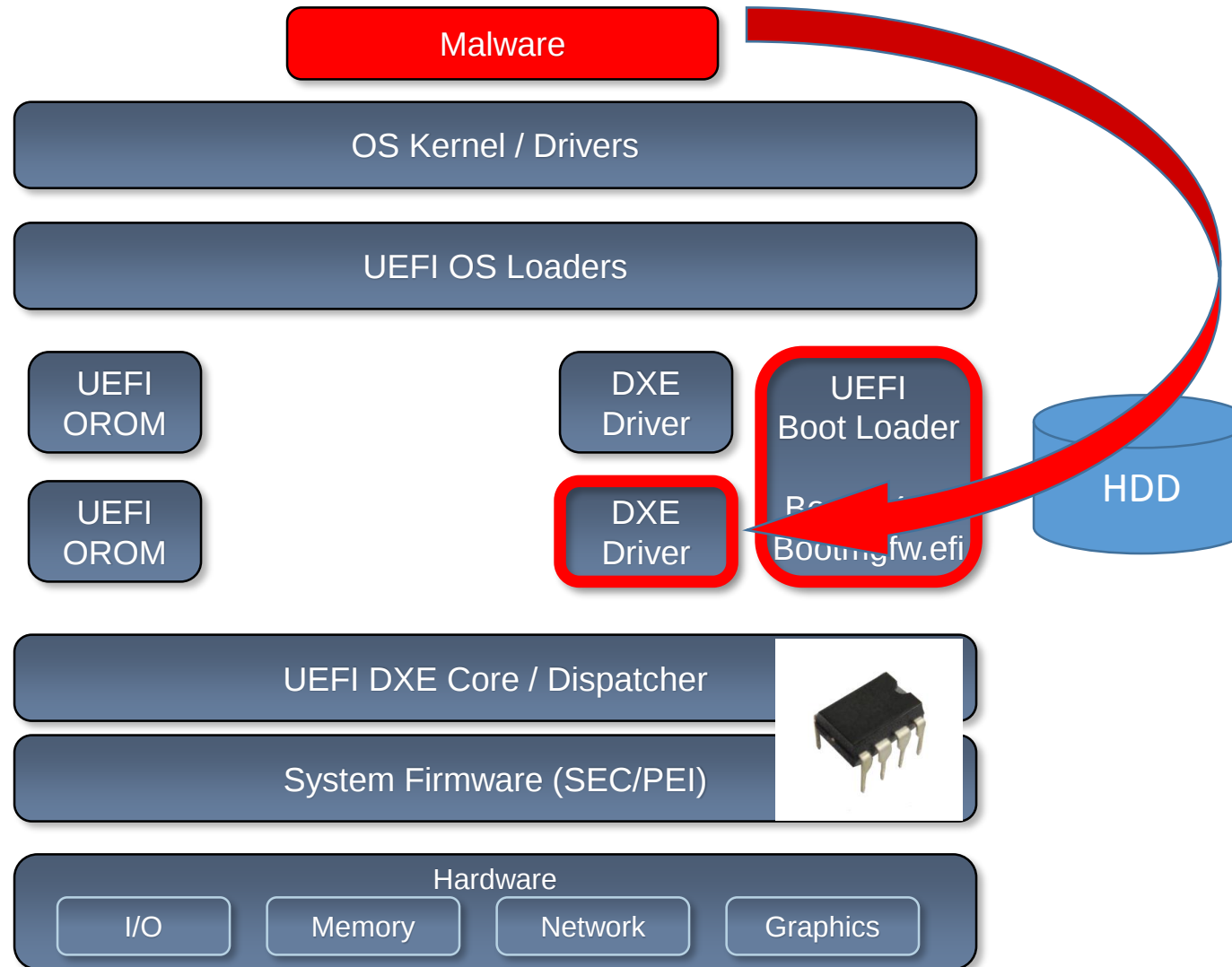
ESP\EFI\Boot\bootx64.efi

[UEFI and Dreamboot](#) by Sébastien Kaczmarek, QUARKSLAB

Adding New Boot Loader (bootkit.efi)

Modified BootOrder / Boot#### EFI variables

UEFI Bootkits



UEFI Bootkits

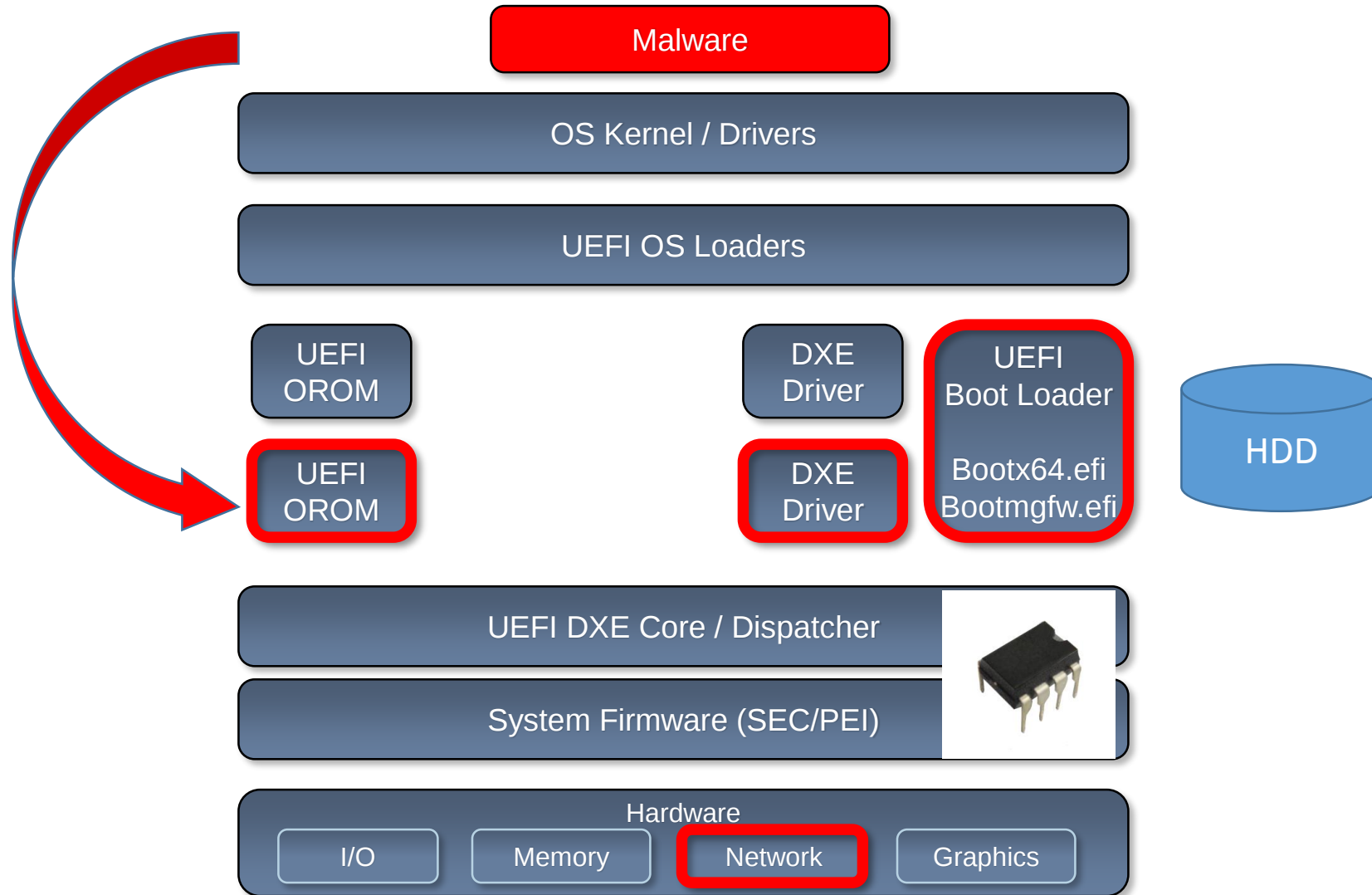
Adding/Replacing DXE Driver

Stored on Fixed Drive

Not embedded in Firmware Volume (FV) in ROM

Modified DriverOrder + Driver#### EFI variables

UEFI Bootkits



UEFI Bootkits

Patching UEFI “Option ROM”

UEFI DXE Driver in Add-On Card (Network, Storage..)

Non-Embedded in FV in ROM

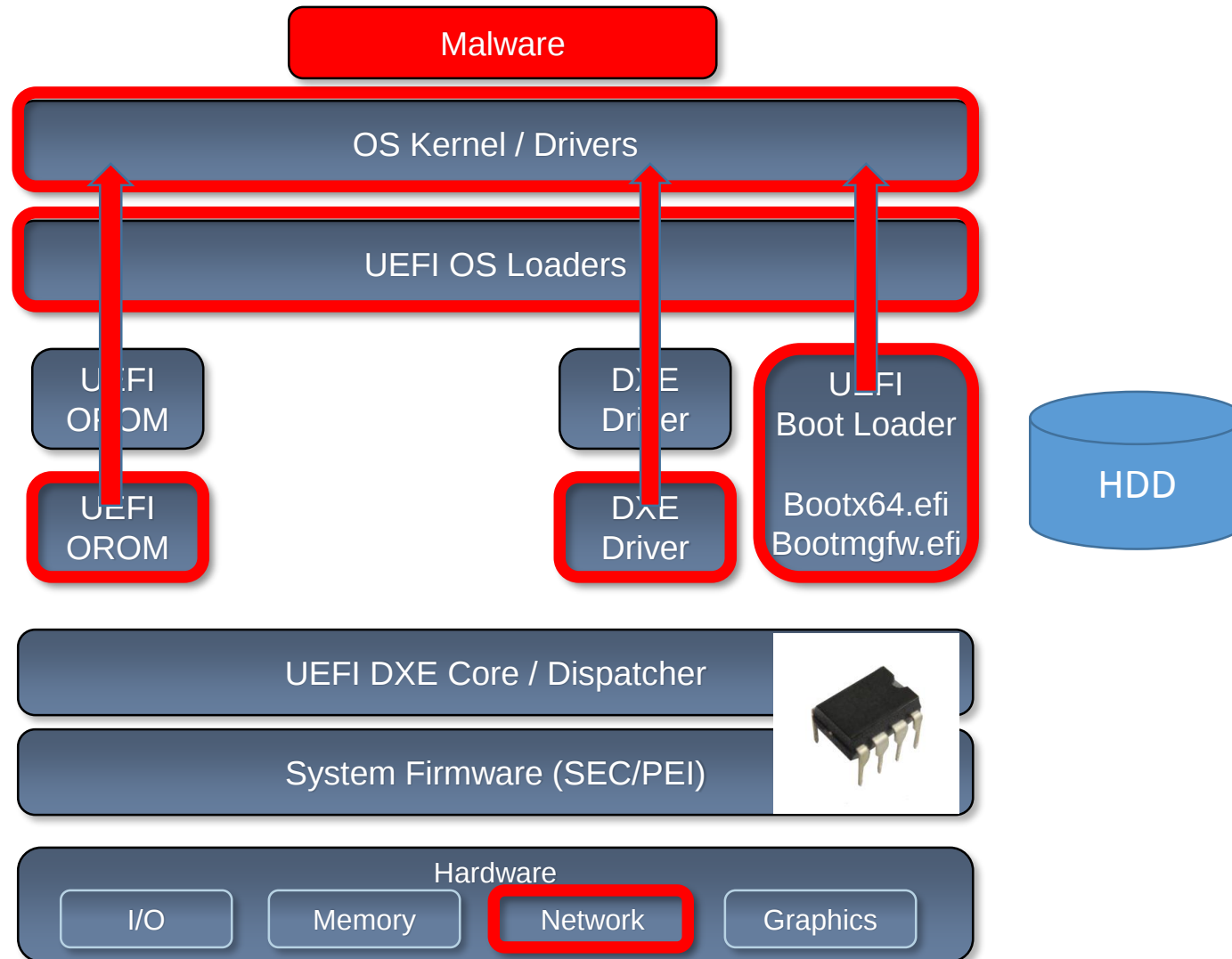
[Mac EFI Rootkits](#) by @snare, Black Hat USA 2012

UEFI Bootkits

Replacing OS Loaders (winload.efi, winresume.efi)

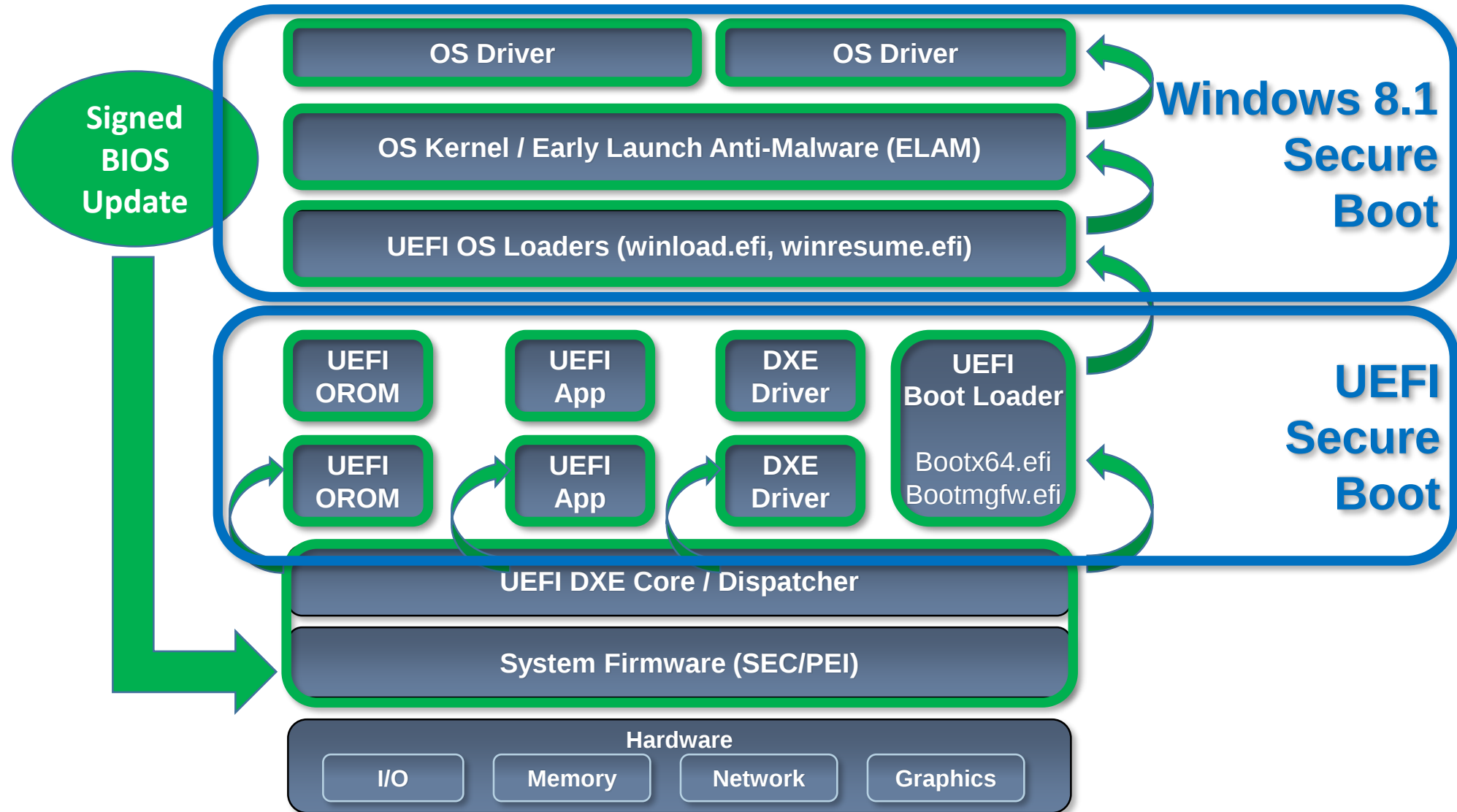
Patching GUID Partition Table (GPT)

UEFI Bootkits

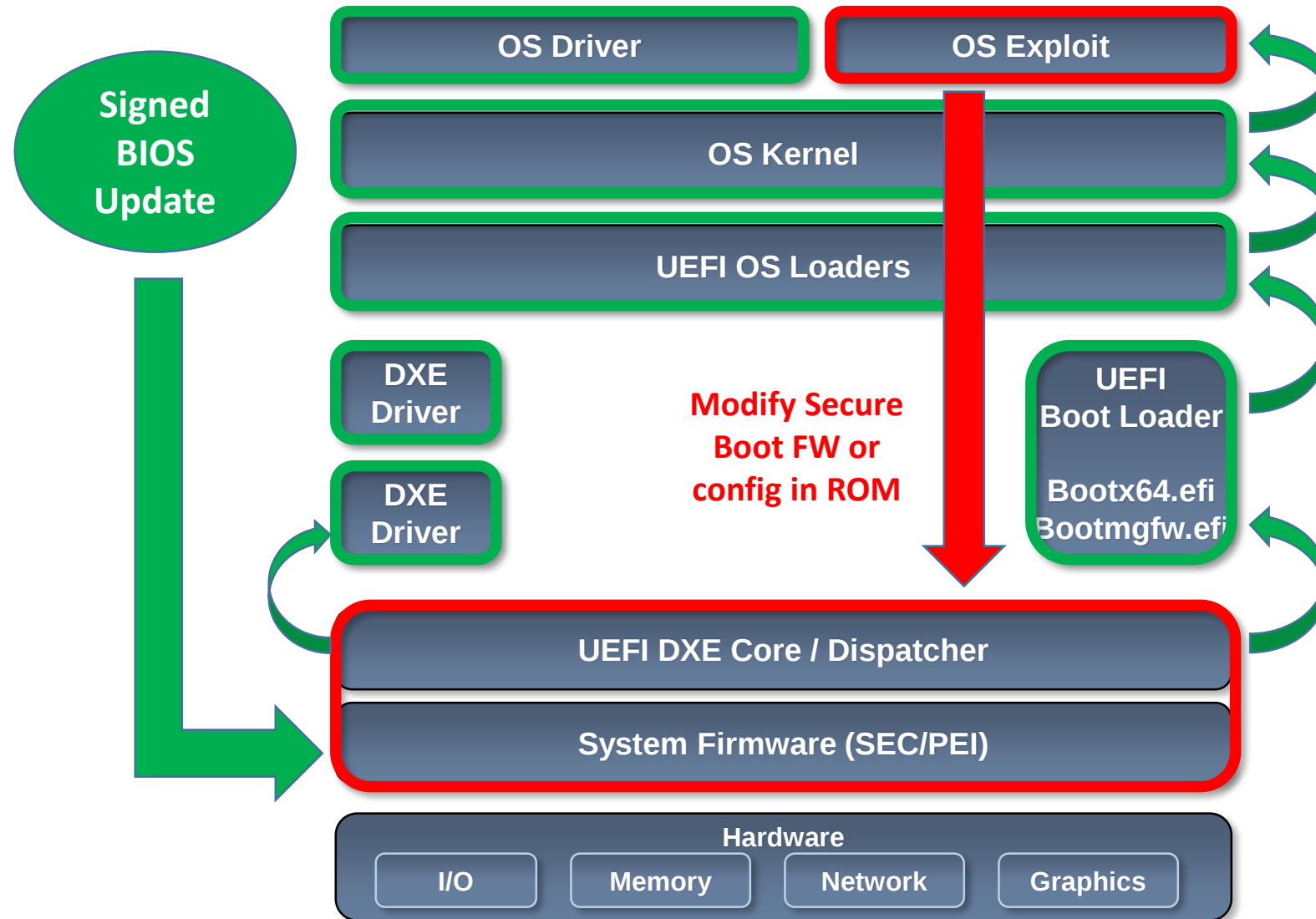


What about Secure Boot?

Secure Boot on MS Windows 8.1



Secure Boot bypass possible?



First Public Windows 8 Secure Boot Bypass (Aug 2013)

```
BIOS Exploit

[+] loaded exploits.bios.bh2013
[+] imported chipsec.modules.exploits.bios.bh2013
[*] BIOS Region: Base = 0x00200000, Limit = 0x007FFFFFFF

[*] Reading 0x80 bytes from BIOS region in ROM (address 0x20F000)..
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |

[+] Checking protection of UEFI BIOS region in ROM..
[spi] UEFI BIOS write protection enabled but not locked. Disabling..
[!] UEFI BIOS write protection is disabled
[*] Writing payload to BIOS region (address 0x20F000)..

[*] Reading BIOS back (address 0x20F000)..
20 20 49 4e 20 59 4f 55 52 20 42 49 4f 53 20 20 | IN YOUR BIOS
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 |
20 20 44 4f 4e 27 54 20 57 4f 52 52 59 21 20 20 | DON'T WORRY!
59 4f 55 52 20 4f 53 20 42 4f 4f 54 20 48 41 53 | YOUR OS BOOT HAS
20 20 42 45 45 4e 20 53 45 43 55 52 45 44 20 20 | BEEN SECURED
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 |
20 42 4c 41 43 4b 20 48 41 54 20 32 30 31 33 20 | BLACK HAT 2013
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |
```

[A Tale Of One Software Bypass Of Windows 8 Secure Boot](#)



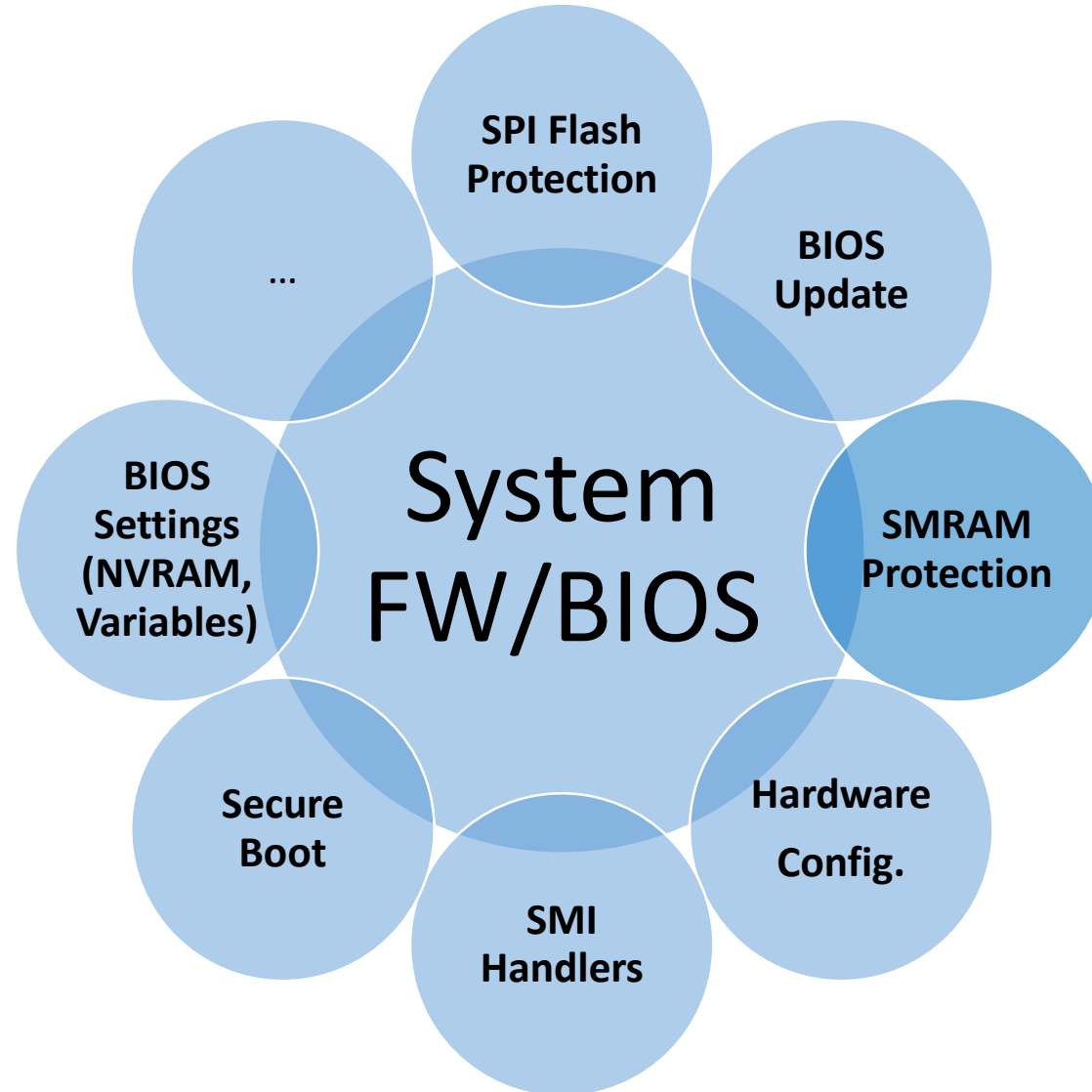
By booting this system up you agree
to have **no** expectation of **privacy**
in any communications or data,
transiting or stored on this system.
any communications or data may be
monitored, intercepted, recorded
and may be **disclosed** for any purpose.
press any key to continue...



ASUS



BIOS Attack Surface



Summary of Attacks Against BIOS and Secure Boot

Subzero Security Patching

“1-days from Hell... get it?”

```
141c142,144
<  if ( sub_FFC40CE8(0x60u) != -1 || sub_FFC40CE8(0x64u) != -1 )
---
>  sub_FFC40D21(0xCF8u, 0x8000F8DC);
>  sub_FFC40D0F(0xCFCu, 2u);
>  if ( sub_FFC40D08(0x60u) != -1 || sub_FFC40D08(0x64u) != -1 )
```

From [Analytics, and Scalability, and UEFI Exploitation](#) by Teddy Reed

Patch attempts to enable BIOS write protection (sets BIOS_CONTROL[BLE]). Picked up by [Subzero](#)



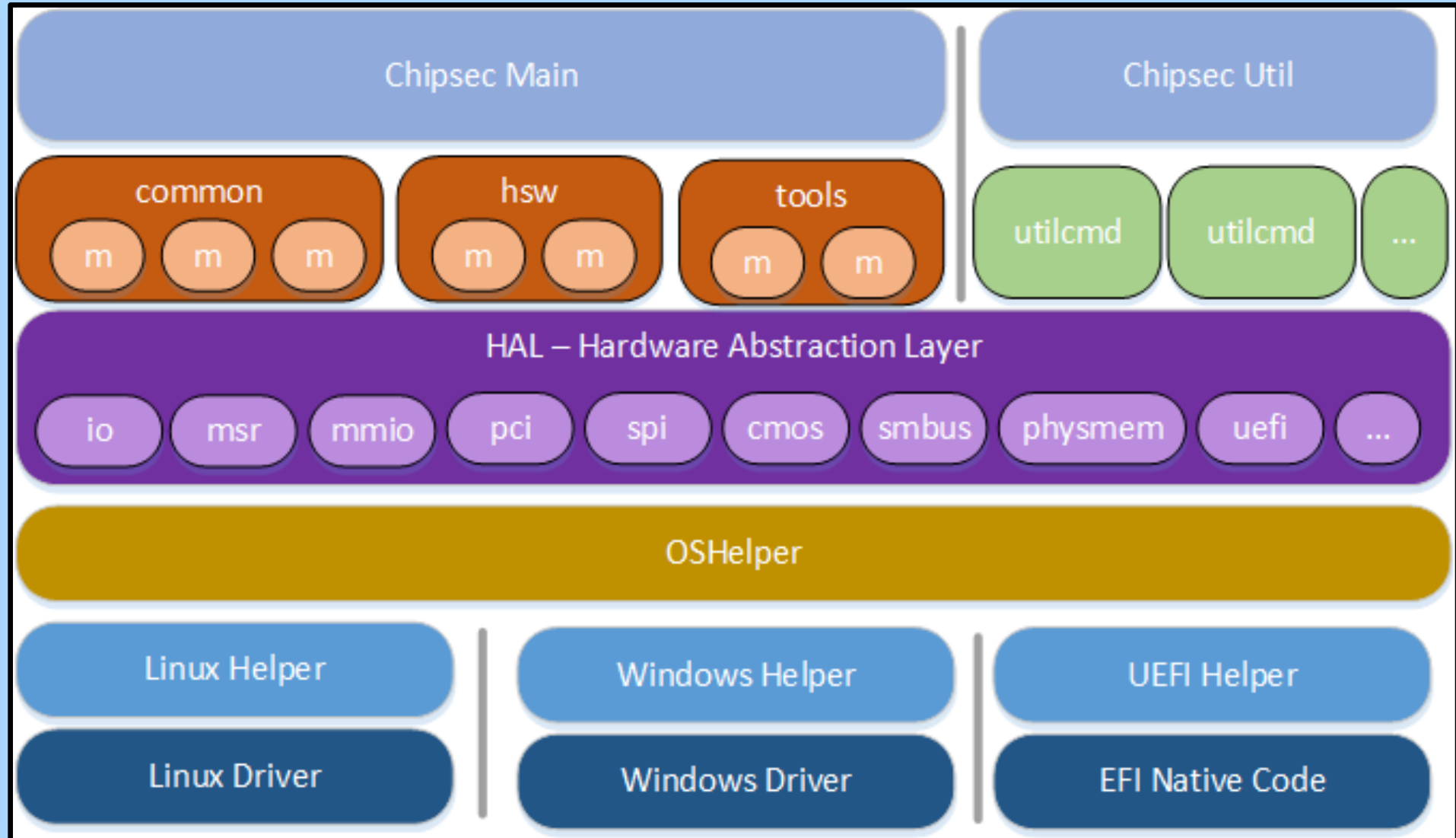
CHIPSEC

Platform Security Assessment Framework

<https://github.com/chipsec/chipsec>
@CHIPSEC



CHIPSEC: Platform Security Assessment Framework





CHIPSEC: Platform Security Assessment Framework

`chipsec_main.py`

runs modules (see modules dir below)

`chipsec_util.py`

runs manual utilities (see utilcmd dir below)

`/chipsec`

`/cfg`

platform specific configuration

`/hal`

all the HW stuff you can interact with

`/helper`

support for OS/environments

`/modules`

modules (tests/tools/PoCs) go here

`/utilcmd`

utility commands for chipsec_util

Known Threats and CHIPSEC modules

Issue	CHIPSEC Module	References
SMRAM Locking	common.smm	CanSecWest 2006
BIOS Keyboard Buffer Sanitization	common.bios_kbrd_buffer	DEFCON 16 2008
SMRR Configuration	common.smrr	ITL 2009 CanSecWest 2009
BIOS Protection	common.bios_wp	BlackHat USA 2009 CanSecWest 2013 Black Hat 2013 NoSuchCon 2013 Flashrom
SPI Controller Locking	common.spi_lock	Flashrom Copernicus
BIOS Interface Locking	common.bios_ts	PoC 2007
Access Control for Secure Boot Keys	common.secureboot.keys	UEFI 2.4 Spec
Access Control for Secure Boot Variables	common.secureboot.variables	UEFI 2.4 Spec

BIOS/Firmware Forensics

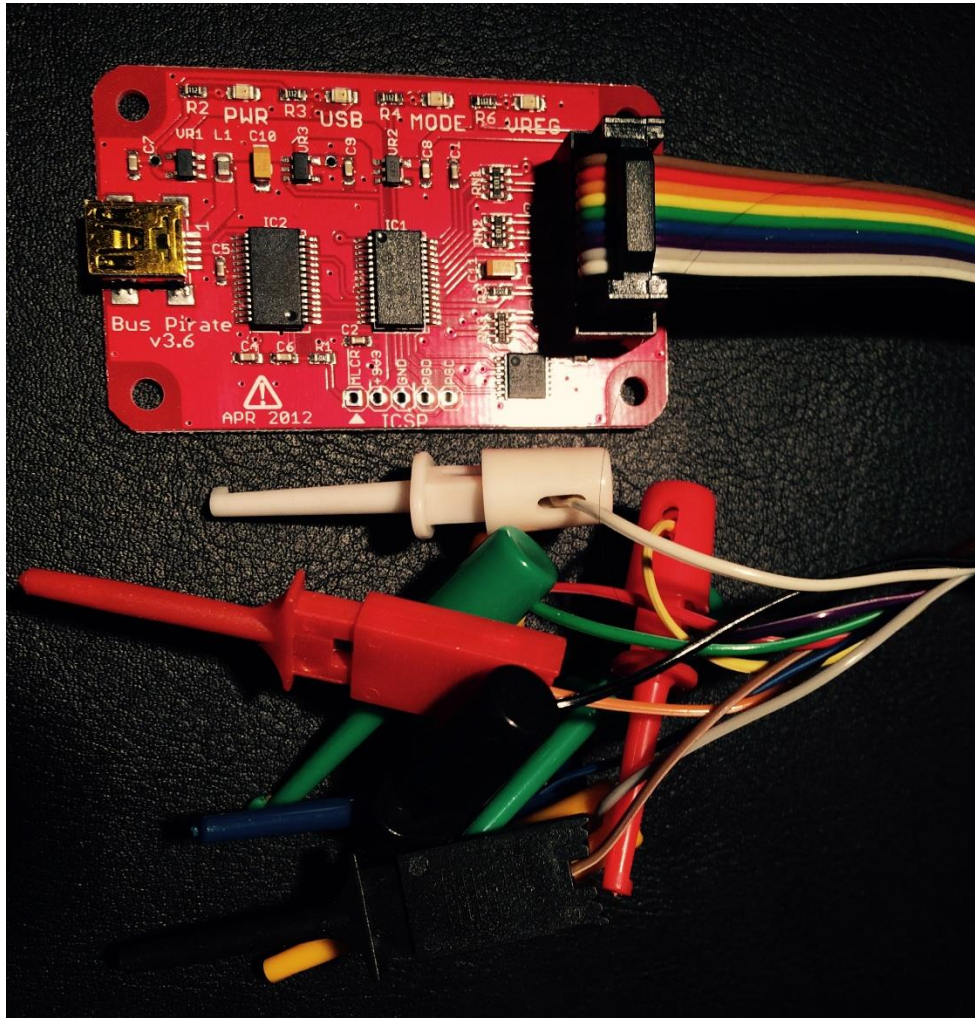
Live system firmware analysis

```
chipsec_util spi info
chipsec_util spi dump rom.bin
chipsec_util spi read 0x700000 0x100000 bios.bin
chipsec_util uefi var-list
chipsec_util uefi var-read db
                        D719B2CB-3D3A-4596-A3BC-DAD00E67656F db.bin
```

Offline system firmware analysis

```
chipsec_util uefi keys PK.bin
chipsec_util uefi nvram vss bios.bin
chipsec_util uefi decode rom.bin
chipsec_util decode rom.bin
```

How to dump BIOS firmware directly from chip?



How to dump BIOS firmware directly from chip?



DEMO TIME



Advanced Malware Analysis



Book is coming in 2015!

Stay Tuned ;)

Thank you for your attention!

Alexander Matrosov
@matrosov

Eugene Rodionov
@vradius

David Harley
@DavidHarleyBlog



2014
SEATTLE 
24 - 26 September 2014