



AUGUST 6-7, 2025
MANDALAY BAY / LAS VEGAS

Booting into Breaches

Hunting Windows SecureBoot's Remote Attack Surfaces

Azure Yang @ CyberKunlun



- Azure Yang @4zure9

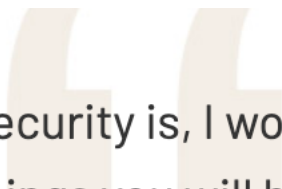
Security Researcher @ Cyber Kunlun | MSRC MVR(2022–2025)

- Started journey into Windows security from late 2021
- Discovered **79 public CVEs** in Windows security, specializing in bootloaders, remote vulnerabilities. Ranked **#5 on MSRC's 2024/2025 annual Windows Leaderboard** and **#2 in 2023Q4** for SecureBoot research.
- Retired CTF player, **DEF CON CTF Black Badge** owner.
- Blending offensive expertise into defensive evolution.

- **Background**
- Attack surface in bootloader
 - Network protocol
 - BCD Registry
 - Security Policy
 - Filesystem
 - Logic flaw
- How to fuzz
- Attack surface beyond bootloader
- Future Work & Take Aways

Why Explore SecureBoot?

- Exploring unknown area is attractive for researcher

A large, stylized, light-colored graphic of a double quote mark (") is positioned to the left of the quote text.

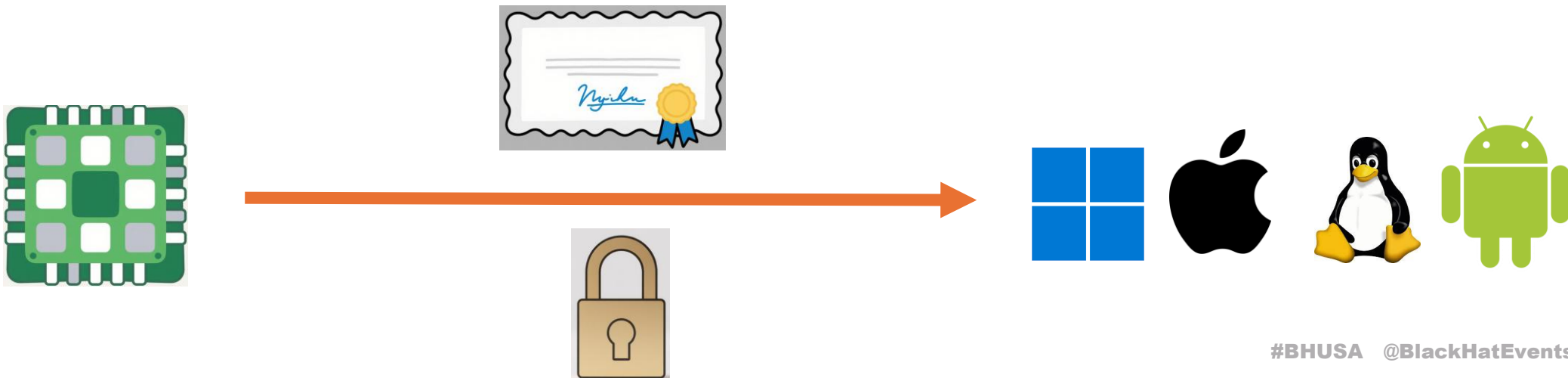
Security is, I would say, our top priority because for all the exciting things you will be able to do with computers - organizing your lives, staying in touch with people, being creative - if we don't solve these security problems, then people will hold back.

— Bill Gates

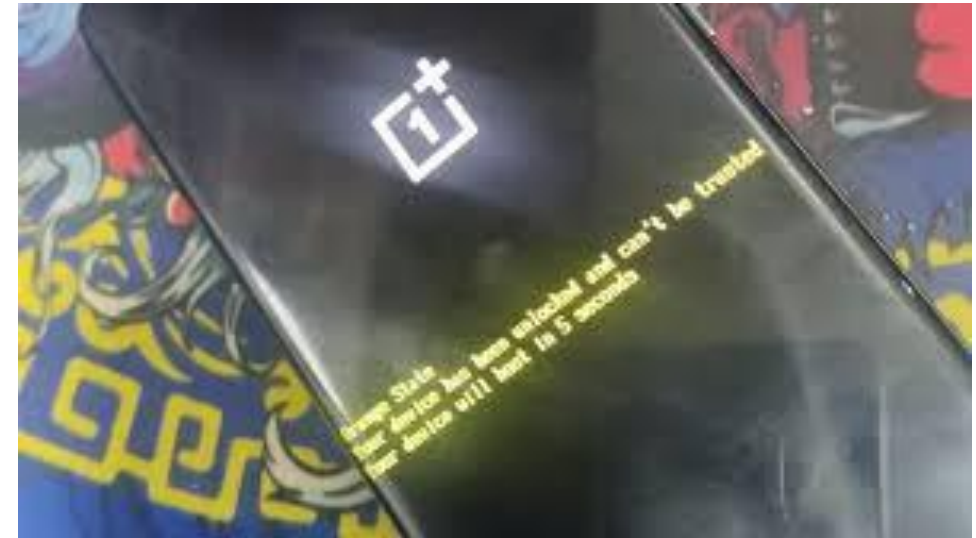
- The foundation of computer security starts with SecureBoot process
- SecureBoot vulnerabilities in Windows is rare in past decade.

SecureBoot – The bigger picture

- Mobile – Hardware lockout implementation
- PC – UEFI
- Using digital signatures and certificates to establishing a chain of trust from hardware to OS

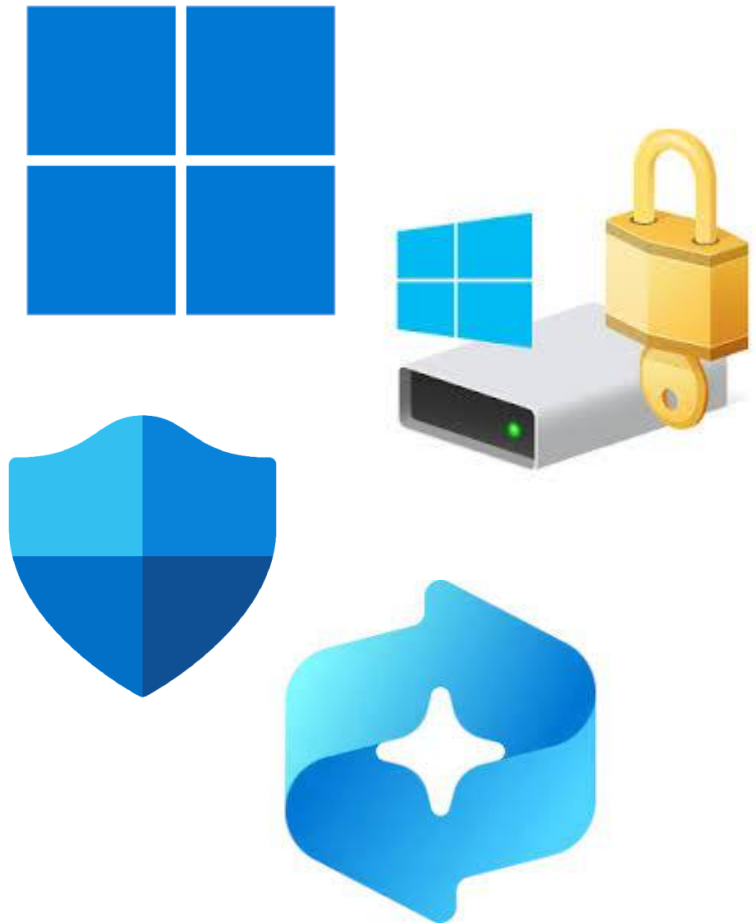


Mobile Secureboot



SecureBoot – Where is enforced

Windows Feature	Secure Boot Required?	Notes
Windows 11 Installation/Upgrade	Yes (capable)	Must be Secure Boot capable, recommended to enable
BitLocker Device Encryption	Strongly recommended	Protects boot chain integrity
Credential Guard	Yes	Depends on Secure Boot for trusted boot
Device Guard	Yes	Uses Secure Boot for code integrity
Early Launch Anti-Malware (ELAM)	Yes	Ensures trusted anti-malware drivers load first
Measured Boot	Yes	Relies on Secure Boot for integrity checks
Recall (Copilot+ PCs)	Yes	Requires Secure Boot, BitLocker, and Windows Hello



What makes the SecureBoot breaches

- Despite fixed in code and the updates has already been shipped, all my 32 Secure Boot Vulnerabilities findings still exploitable by default

- PCA2011 gets expired in 2026

- PCA2023

- UEFI var DBX 32K limit

- Compatibility issue

 Microsoft Root Certificate Authority 2010

 Microsoft Windows Production PCA 2011

 Microsoft Root Certificate Authority 2010

 Windows UEFI CA 2023

Issued to: Microsoft Windows Production PCA 2011

Issued by: Microsoft Root Certificate Authority 2010

Valid from 10/19/2011 **to** 10/19/2026

Golden Key's unlock attack

Fixed in	CVE	Title	In wild	Score	CVSS
2016-Jul	CVE-2016-3287	Secure Boot Security Feature Bypass Vulnerability	FALSE	6.2	CVSS:3.0/AV:P
2016-Aug	CVE-2016-3320	Secure Boot Security Feature Bypass Vulnerability	FALSE	6.6	CVSS:3.0/AV:P
2016-Nov	CVE-2016-7247	Secure Boot Component Security Feature Bypass Vulnerability	FALSE	6.2	CVSS:3.0/AV:P
2019-Sep	CVE-2019-1294	Windows Secure Boot Security Feature Bypass Vulnerability	FALSE	5.3	CVSS:3.0/AV:P
2019-Oct	CVE-2019-1368	Windows Secure Boot Security Feature Bypass Vulnerability	FALSE	4.9	CVSS:3.0/AV:P
2020-Feb	CVE-2020-0689	Microsoft Secure Boot Security Feature Bypass Vulnerability	FALSE	8.2	CVSS:3.0/AV:L
2022-Jan	CVE-2022-21894	Secure Boot Security Feature Bypass Vulnerability	FALSE	4.4	CVSS:3.1/AV:L
2023-May	CVE-2023-24932	Secure Boot Security Feature Bypass Vulnerability	TRUE	6.7	CVSS:3.1/AV:L

- About Attack vector

- (P)hysical
- (L)ocal
- (R)emote
- (A)djacent

Used by BlackLotus bootkit malware

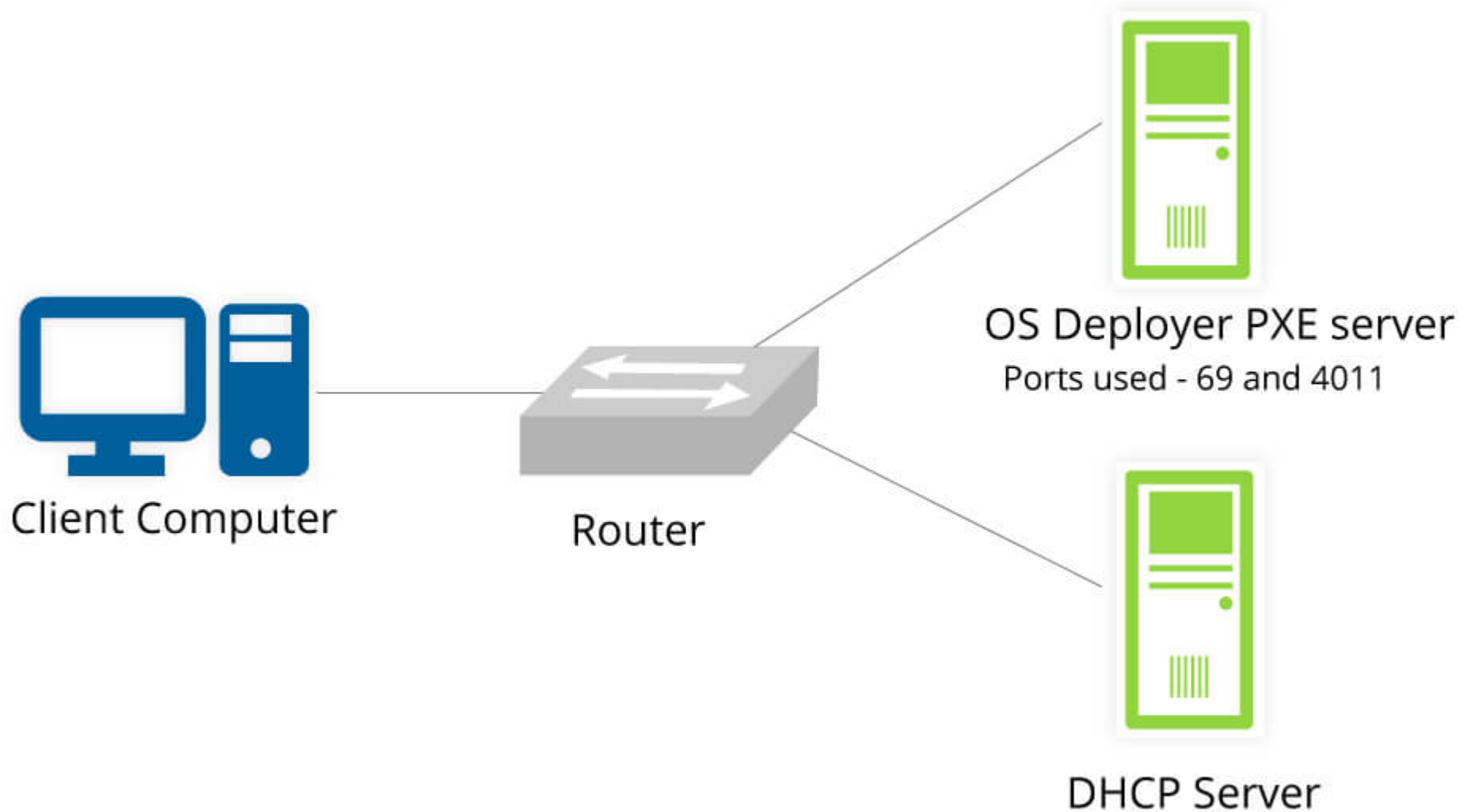
There's only 56 Secure Boot SFB from 2016-2025

1	2024-Apr	CVE-2024-20688	Secure Boot Security Feature Bypass Vulnerability	7.1	CVSS:3.1/AV:A/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
2	2024-Apr	CVE-2024-28896	Secure Boot Security Feature Bypass Vulnerability	7.5	CVSS:3.1/AV:A/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
3	2024-Apr	CVE-2024-28898	Secure Boot Security Feature Bypass Vulnerability	6.3	CVSS:3.1/AV:A/AC:H/PR:H/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
4	2024-Apr	CVE-2024-20689	Secure Boot Security Feature Bypass Vulnerability	7.1	CVSS:3.1/AV:A/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
5	2024-Apr	CVE-2024-26171	Secure Boot Security Feature Bypass Vulnerability	6.7	CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
6	2024-Apr	CVE-2024-26175	Secure Boot Security Feature Bypass Vulnerability	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
7	2024-Apr	CVE-2024-26180	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
8	2024-Apr	CVE-2024-26189	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
9	2024-Apr	CVE-2024-26240	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
10	2024-Apr	CVE-2024-28925	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
11	2024-Apr	CVE-2024-28897	Secure Boot Security Feature Bypass Vulnerability	6.8	CVSS:3.1/AV:A/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
12	2024-Apr	CVE-2024-29061	Secure Boot Security Feature Bypass Vulnerability	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
13	2024-Apr	CVE-2024-29062	Secure Boot Security Feature Bypass Vulnerability	7.1	CVSS:3.1/AV:A/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
14	2024-Apr	CVE-2024-28923	Secure Boot Security Feature Bypass Vulnerability	6.4	CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
15	2024-Jul	CVE-2024-28899	Secure Boot Security Feature Bypass Vulnerability	8.8	CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
16	2024-Jul	CVE-2024-37969	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
17	2024-Jul	CVE-2024-37970	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
18	2024-Jul	CVE-2024-37974	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
19	2024-Jul	CVE-2024-37981	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
20	2024-Jul	CVE-2024-37986	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
21	2024-Jul	CVE-2024-37987	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
22	2024-Jul	CVE-2024-26184	Secure Boot Security Feature Bypass Vulnerability	6.8	CVSS:3.1/AV:A/AC:H/PR:L/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
23	2024-Jul	CVE-2024-37971	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
24	2024-Jul	CVE-2024-37972	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
25	2024-Jul	CVE-2024-37973	Secure Boot Security Feature Bypass Vulnerability	8.8	CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
26	2024-Jul	CVE-2024-37975	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
27	2024-Jul	CVE-2024-37977	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
28	2024-Jul	CVE-2024-37978	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
29	2024-Jul	CVE-2024-37988	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
30	2024-Jul	CVE-2024-37989	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
31	2024-Jul	CVE-2024-38010	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C
32	2024-Jul	CVE-2024-38011	Secure Boot Security Feature Bypass Vulnerability	8	CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C



There are also two dozen fixes for security feature bypass (SFB) bugs, although I think we need to rename a component. Between 23 fixes in April and 20 more this month, I don't think we can really call it *Secure* Boot anymore. Even worse, all but two of these could be exploited by an Adjacent attacker with LAN access to the target. Oof. I'm calling this feature "Protected Boot" rather than "Secure Boot". The SFB bug in BitLocker requires physical access, but BitLocker is specifically designed to prevent this sort of attack, so...er...not good.

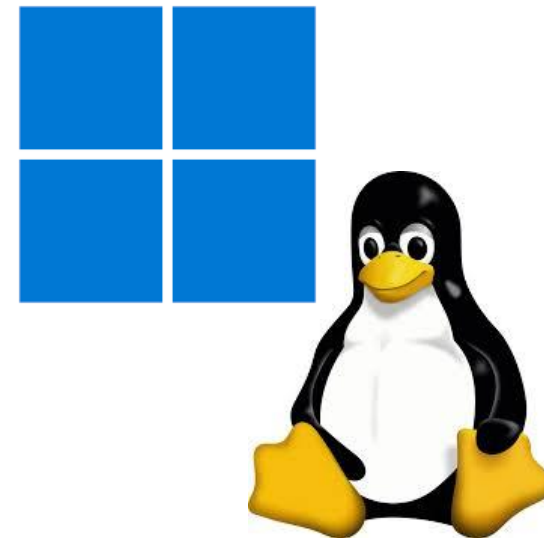
How AV:A possible?



Define Secure boot SFB in Windows

- Before secure boot, Microsoft doesn't acknowledge vulnerabilities in bootloader.
- With Secure Boot, bootloader issue can get a CVE
- Secure Boot is a security feature of Windows
- Vulnerabilities in Secure Boot are security feature bypass
 - It can be remote
 - It can be without user interaction
 - It can be preauth
 - It can be Remote Code Execution/Information Leak
 - It can't be Denial of Service

- Most of PC with UEFI Secure boot enabled
 - Linux
 - Windows
- B(ring) Y(our) O(wn) B(ootloader)
 - Can be from adjacent network
 - Can be preauth
 - Exploitable by default in many scenario
 - Exploitable until PCA2011 expire or added to DBX



Summary of my research

- 55 unique reports
- Duplicate cases are already removed
- All reported cases can be carried by **unauthenticated attacker from network.**
- By finding method
 - Audit: 35
 - Fuzzing: 20
- By attack surface
 - BCD Registry: 25
 - Filesystem: 16
 - Network protocol: 6
 - Windows Kernel: 5

ID	Attack Surface	Compomnet	method	CVE	Type
1	Network protocol	PXE Bootloader	Audit	CVE-2024-20688	Stack OOB W
2	Network protocol	PXE Bootloader	Audit	CVE-2024-20689	Stack OOB W
3	Network protocol	PXE Bootloader	Audit	-	DoS
4	Network protocol	PXE Bootloader	Audit	CVE-2024-28925	Recursive Calling
5	Network protocol	PXE Bootloader	Audit	CVE-2024-26180	Recursive Calling
6	BCD Element Processing	Bootloader	fuzzing	CVE-2024-26175	Heap OOB W
7	BCD Element Processing	Bootloader	fuzzing	CVE-2024-37971	Recursive Calling
8	BCD Element Processing	Bootloader	fuzzing	CVE-2024-37970	Heap OOB W
9	BCD Registry structer	Bootloader	fuzzing	CVE-2024-37972	Heap OOB W
10	BCD Element Processing	Bootloader	fuzzing	CVE-2024-28896	Stack OOB W
11	BCD Element Processing	Bootloader	Audit & fuzzing	CVE-2024-28897	Arbitrary Memory W
12	BCD Element Processing	Bootloader	fuzzing	CVE-2024-28923	Heap OOB W
13	BCD Element Processing	Bootloader	Audit & fuzzing	CVE-2024-37973	Recursive Calling
14	BCD Element Processing	Bootloader	Audit & fuzzing	CVE-2024-29061	Stack OOB W
15	BCD Element Processing	Bootloader	Audit	CVE-2024-37969	Info leak
16	BCD Element Processing	Bootloader	Audit	CVE-2024-37974	Heap OOB W
17	BCD Element Processing	Bootloader	Audit	?	Recursive Calling
18	BCD Element Processing	Bootloader	Audit	CVE-2024-26171	Heap OOB W
19	BCD Element Processing	Bootloader	Audit	CVE-2024-37970	Heap OOB W
20	Network protocol	Bootloader	Audit	CVE-2024-37975	Integer Overflow
21	BCD Element Processing	Bootloader	Audit	?	Arbitrary Memory W
22	BCD Element Processing	Bootloader	Audit	CVE-2024-37978	Heap OOB W
23	BCD Element Processing	Bootloader	Audit	CVE-2024-26240	Calling Stack
24	BCD Element Processing	Bootloader	Audit	CVE-2024-37981	Heap OOB W
25	BCD Element Processing	Bootloader	Audit	CVE-2024-26189	Recursive Calling
26	Security Policy	Bootloader	Audit	-	Logical
27	BCD Element Processing	Bootloader	Audit	CVE-2024-28897	Heap OOB W
28	BCD Element Processing	Bootloader	Audit	CVE-2024-26175	Heap OOB W
29	BCD Element Processing	Bootloader	Audit	CVE-2024-26175	Heap OOB W
30	BCD Element Processing	Bootloader	Audit	CVE-2024-26175	Heap OOB W
31	BCD Element Processing	Bootloader	Audit	CVE-2024-28898	Recursive Calling
32	BCD Element Processing	Bootloader	Audit	CVE-2024-37986	Heap OOB W
33	Architecture issue	Bootloader	Audit	-	Logical
34	WIM filesystem	Bootloader	fuzzing	CVE-2024-37987	Invalid Pointer Deref
35	WIM filesystem	Bootloader	Audit	CVE-2024-37988	Heap OOB W
36	WIM filesystem	Bootloader	Audit	CVE-2024-37989	Arbitrary Memory W
37	WIM filesystem	Bootloader	Audit	CVE-2024-38010	Heap OOB W
38	WIM filesystem	Bootloader	fuzzing	-	DoS
39	WIM filesystem	Bootloader	fuzzing	-	DoS
40	NTFS filesystem	Bootloader	fuzzing	-	DoS
41	NTFS filesystem	Bootloader	fuzzing	?	Recursive Calling
42	WIM filesystem	Bootloader	Audit	?	Arbitrary Memory W
43	NTFS filesystem	Bootloader	fuzzing	-	DoS
44	NTFS filesystem	Bootloader	fuzzing	-	DoS
45	NTFS filesystem	Bootloader	fuzzing	CVE-2024-38011	Heap OOB W
46	NTFS filesystem	Bootloader	fuzzing	CVE-2024-28899	Recursive Calling
47	NTFS filesystem	Bootloader	fuzzing	-	DoS
48	FAT filesystem	Bootloader	fuzzing	-	DoS
49	FAT filesystem	Bootloader	fuzzing	-	DoS
50	Architecture issue	Bootloader	Audit	CVE-2024-29062	Logical
51	Driver Config	Kernel	fuzzing	-	Heap OOB W
52	Sdb Parsing	Kernel	fuzzing	-	DoS
53	Driver Config	Kernel	Audit	-	Heap OOB W
54	Driver Config	Kernel	Audit	-	DoS
55	Driver Config	Kernel	Audit	-	Recursive Calling

My way to reduce duplication report

- On the developer view
 - Find a bug
 - Write a fuzzer to make it discoverable through fuzzing
 - Conduct hot patching on vulnerability
 - Find out if there are still any related crashes caused by the same rootcause.
 - Repeat

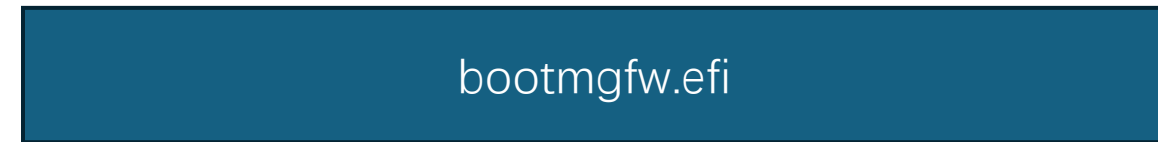
About the heap memory management

- No pageheap
- OOB write can happened silently without crash the bootloader
- The MmHapReportHeapCorruption itself has self recursive calling issue
- Allocate 0x20 at least, OOB write to block offset less than 0x20 is not a real vulnerability can be exploit, block is 0x20 aligned.

- Background
- **Attack surface in bootloader**
 - Network protocol
 - BCD Registry
 - Security Policy
 - Filesystem
 - Logic flaw
- How to fuzz
- Attack surface beyond bootloader
- Future Work & Take Aways

Network protocol

- IPv4 DHCP PXE
- IPv6 DHCPv6 PXE
- HTTP
- WDS Multicast



IPv4/v6



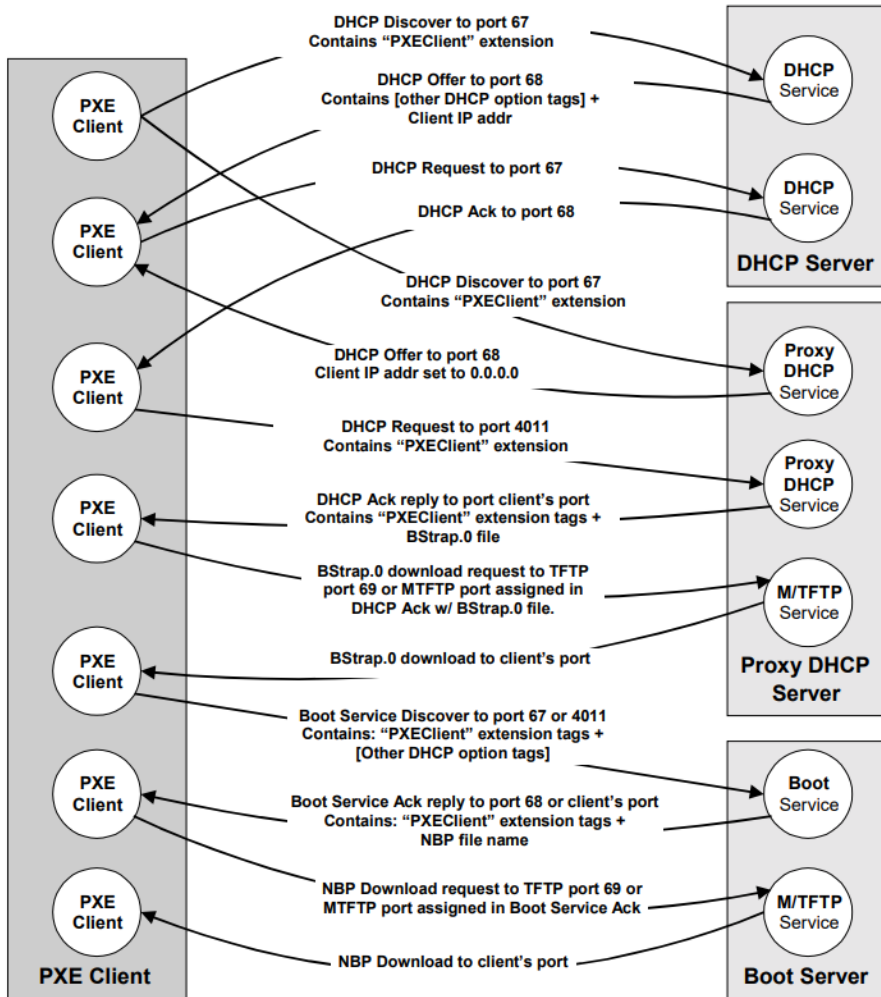


Figure 2-4 PXE Client Response to DHCP Server Supplying Boot Service Discovery Code

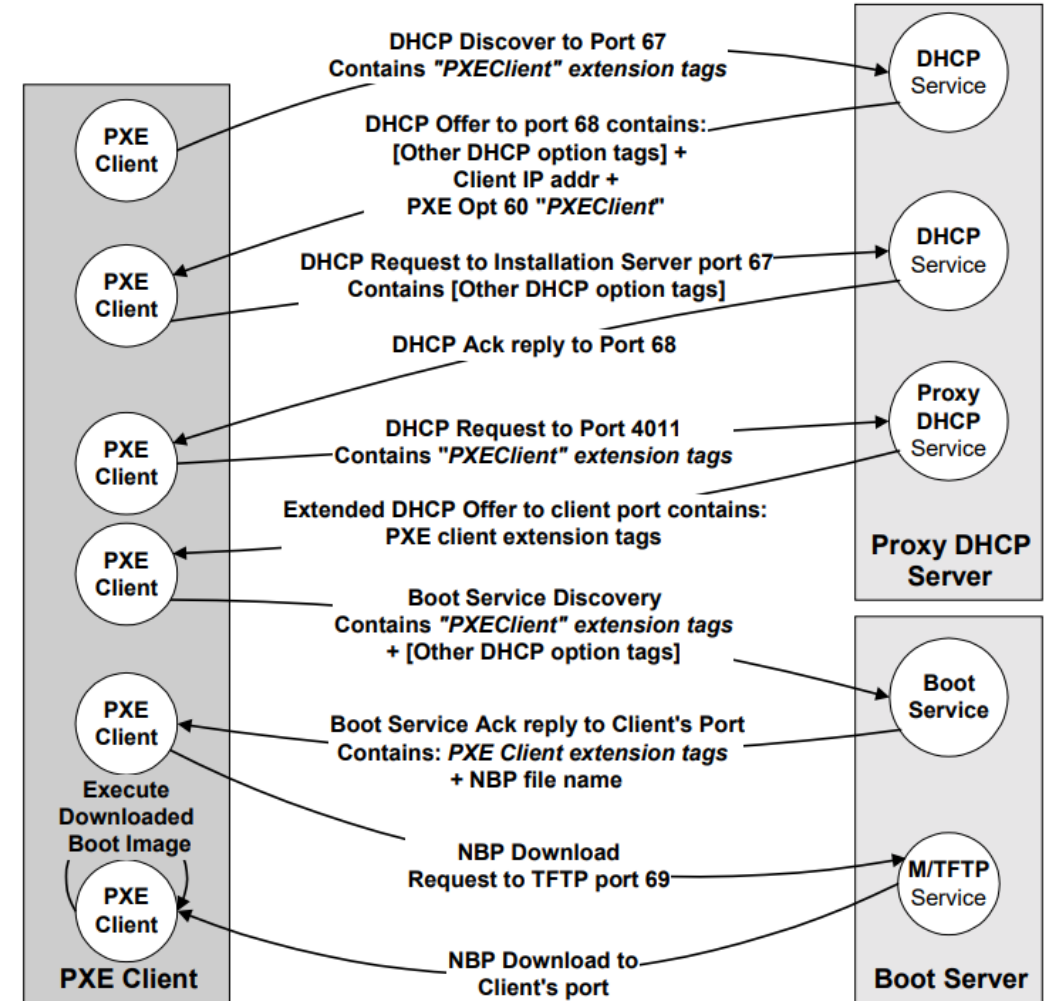


Figure 2-3 PXE Client Response to DHCP Server Containing a Proxy DHCP Service

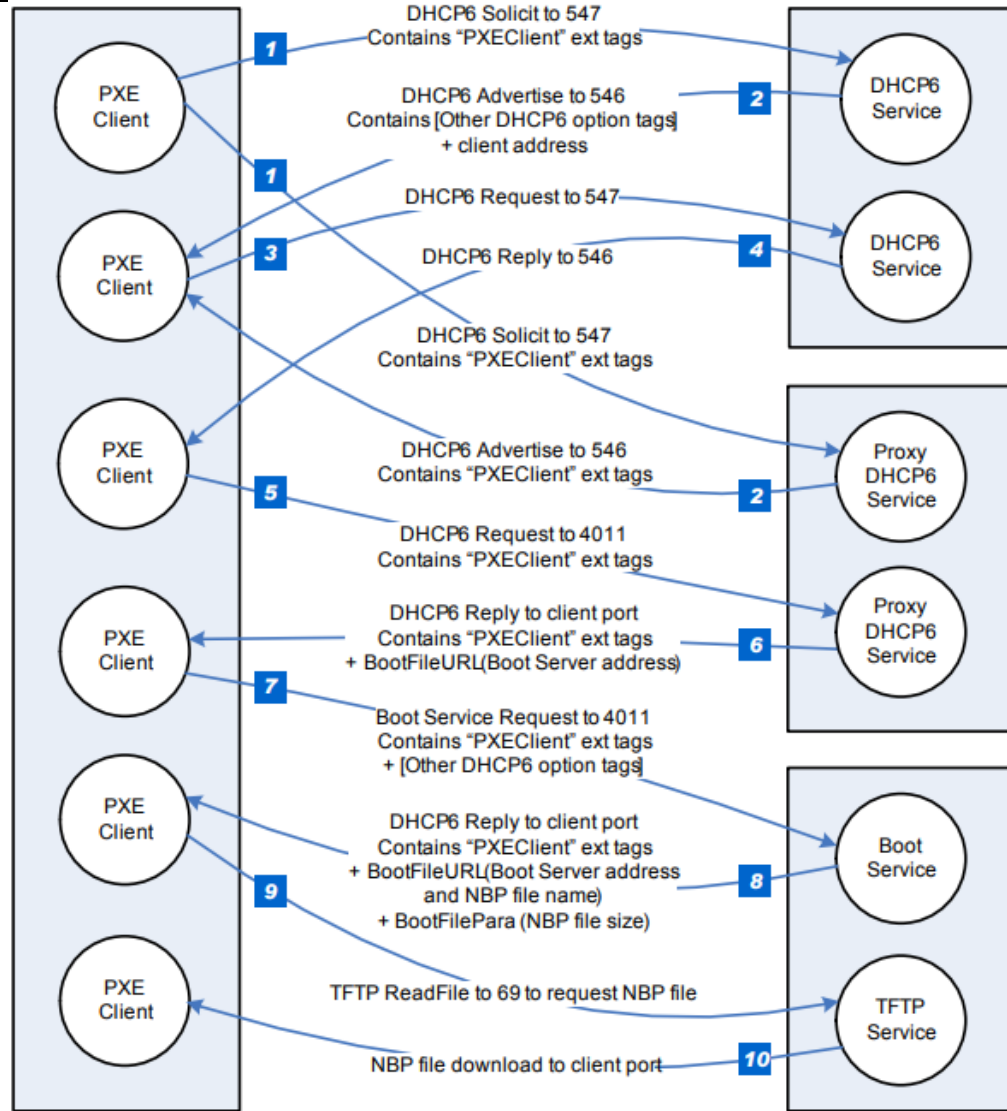


Figure 59. IPv6-based PXE boot (DHCP6 and ProxyDHCP6 reside on the different server)

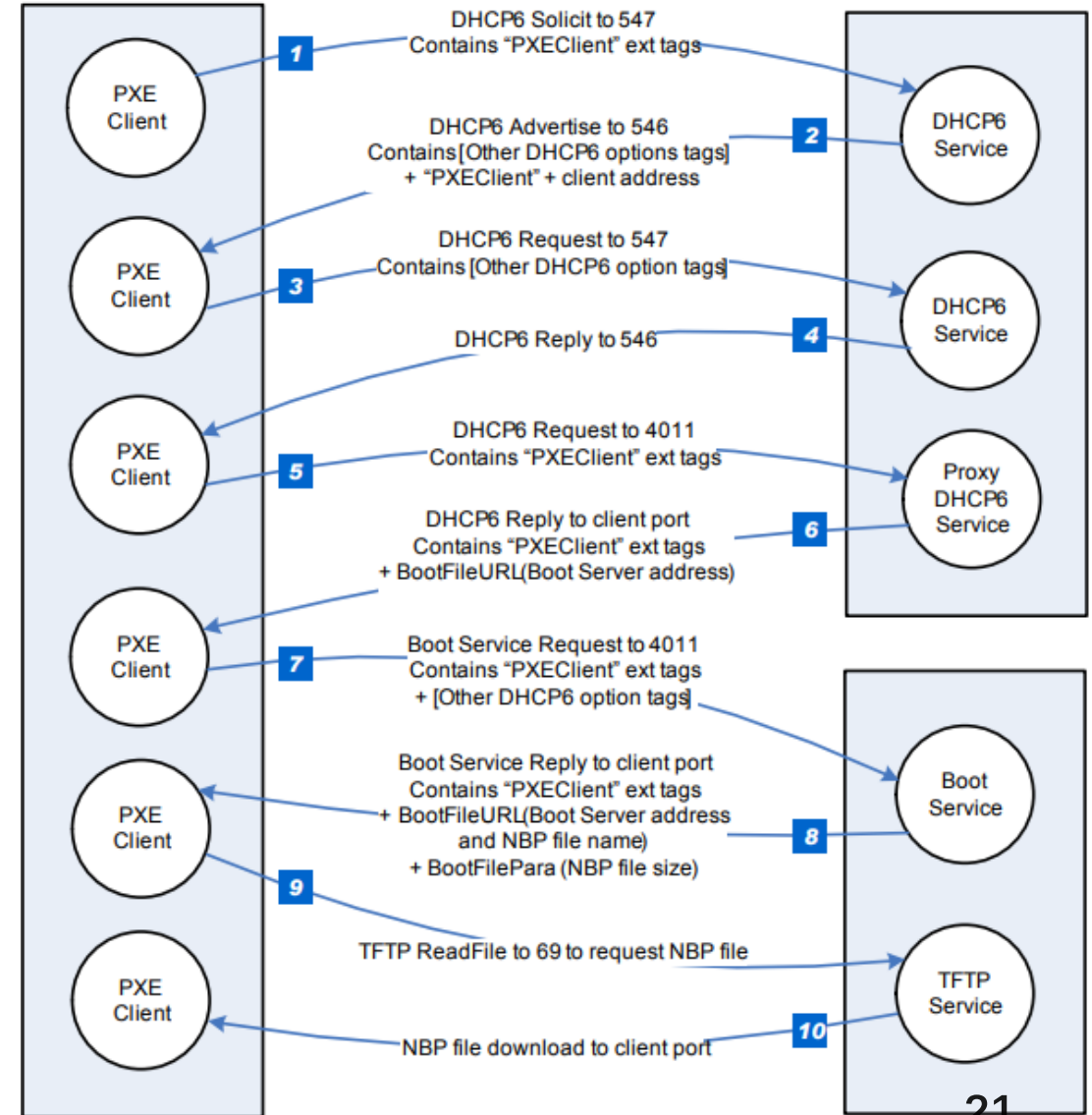
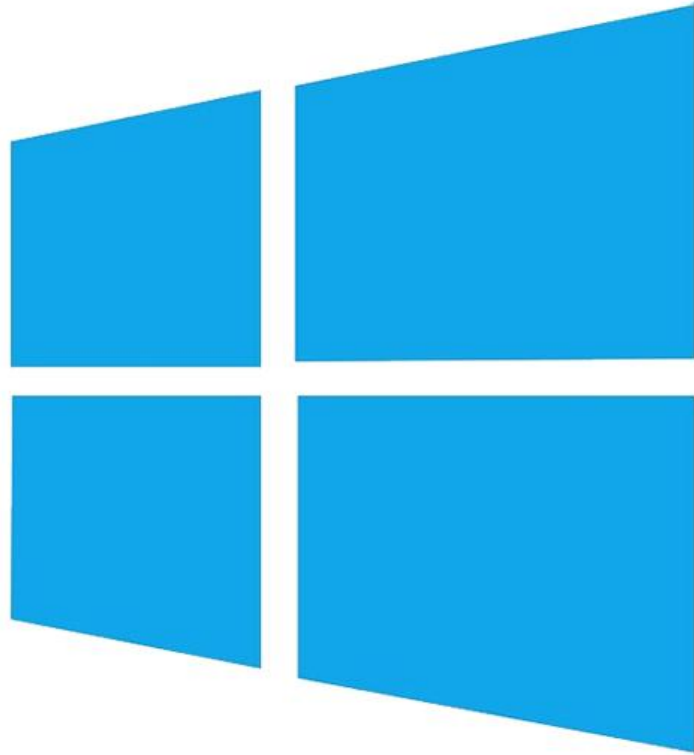


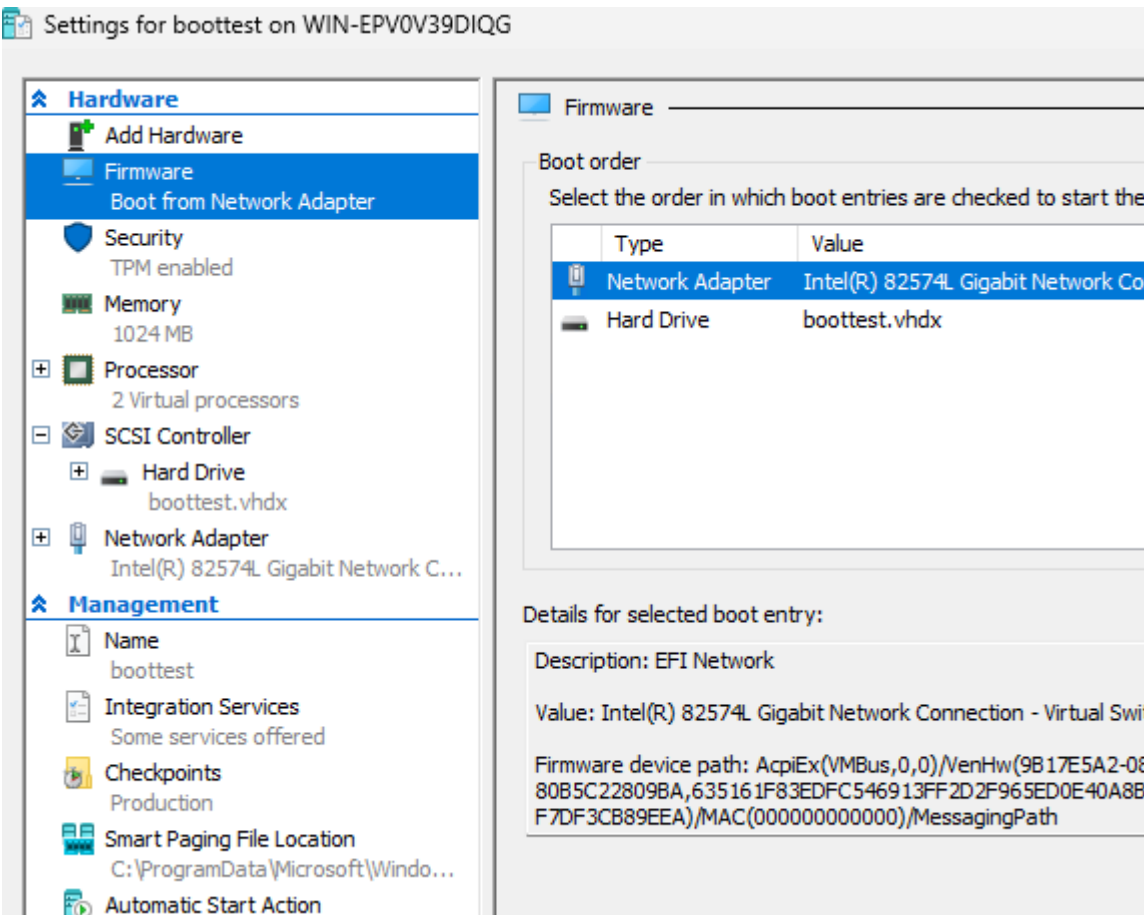
Figure 58. netboot6 (DHCP6 and ProxyDHCP6 reside on the same server)

- Hyper-V Gen2 VM is recommended



Microsoft
Hyper-V

Hyper-V Gen 2 VM Default Boot Settings



Support IPv6 boot in firmware,
Use powershell to enable IPv6 PXE booting.

Set-VMFirmware boottest -PreferredNetworkBootProtocol IPv6

```
>>Start PXE over IPv6...
Station IP address is 2001:DA8:FF:212:0:0:B362:C674
```

```
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
DXE driver
```

Hyper-V™

Why choose Hyper-V

```

70 v5 = v4->__vftable->TriageDump64::Initialize__MEMORY_DUMP_PARAMETERS__PTR(
71     v4,
72     (GuestCrashDumpWriter *)((char *)this + 56));
73 if ( v5 >= 0 )
74 {
75     LODWORD(v18) = 3;
76     v19 = 0i64;
77     v9 = (const WCHAR *)((char *)this + 8);
78     v10 = (const unsigned __int16 *)((char *)this + 8);
79     if ( *((_QWORD *)this + 4) >= 8ui64 )
80         v10 = *(const unsigned __int16 **)v9;
81     Vm1::VmFile::VmFile((Vm1::VmFile *)&v19, v10, v8, *((_DWORD *)this + 11));
82     LODWORD(v18) = 4;
83     v5 = v4->__vftable->TriageDump64::Write_Vm1::VmFile(v4, &v19); // Generate Dump, ?Write@TriageDump64@@UEAAJAEAVVmFile@Vm1@@@Z
84     Vm1::VmFile::Reset(&v19, v11);
85     if ( v5 >= 0 )
86     {
87         v4->__vftable->TriageDump64::GetBugcheckCode_uint(v4, (unsigned int *)&v18);
88         v20 = (struct _EVENT_DESCRIPTOR)MSVM_GUEST_CRASH_DUMP_SUCCESS;
89         Vm1::VmFile::Reset(&v19, v13);
90         v5 = 0;
91     }
92     else
93     {
94         LODWORD(v18) = v5;
95         if ( *((_QWORD *)this + 4) >= 8ui64 )
96             v9 = *(const WCHAR **)v9;
97         DeleteFileW(v9);
98         Vm1::VmFile::Reset(&v19, v12);

```

Bugcheck Analysis

KMODE_EXCEPTION_NOT_HANDLED (1e)

This is a very common BugCheck. Usually the exception address pinpoints the driver/function that caused the problem. Always note this address as well as the link date of the driver/image that contains this address.

Arguments:

Arg1: ffffffffcc0000005, The exception code that was not handled

Arg2: 000000003e915383, The address that the exception occurred at

Arg3: 0000000000000000, Parameter 0 of the exception

Arg4: 0000000046eece8, Parameter 1 of the exception

bootmgfw!WimpSearchForDirent+0x63:

00000000`3e915383 0fb74364 movzx eax,word ptr [rbx+64h] ds:0030:41414141`414141a5=????

kd> kf

#	Memory	Child-SP	RetAddr	Call Site
00		00000000`046eebb0	00000000`3e91461c	bootmgfw!WimpSearchForDirent+0x63
01	40	00000000`046eebf0	00000000`3e90f36a	bootmgfw!WimOpen+0x74
02	100	00000000`046eecf0	00000000`3e90f32c	bootmgfw!FileIoOpen+0x24e
03	a0	00000000`046eed90	00000000`3e90f32c	bootmgfw!FileIoOpen+0x210
04	a0	00000000`046eee30	00000000`3e90ee14	bootmgfw!FileIoOpen+0x210
05	a0	00000000`046eedd0	00000000`3e90da21	bootmgfw!BlpFileOpen+0xe8
06	160	00000000`046ef030	00000000`3ea6084f	bootmgfw!BlFileOpen+0x71
07	70	00000000`046ef0a0	00000000`3e9e903e	bootmgfw!SbeEnumerateFilesInDirectory+0x5f
08	70	00000000`046ef110	00000000`3e8a2214	bootmgfw!BlImgLoadBootApplication+0x21e
09	1b0	00000000`046ef2c0	00000000`3e8a30d0	bootmgfw!BmTransferExecution+0x84
0a	100	00000000`046ef3c0	00000000`3e8a1c46	bootmgfw!BmpLaunchBootEntry+0x25c

BUGCHECK_CODE: 1e

BUGCHECK_P1: ffffffffcc0000005

BUGCHECK_P2: 3e915383

BUGCHECK_P3: 0

BUGCHECK_P4: 46eece8

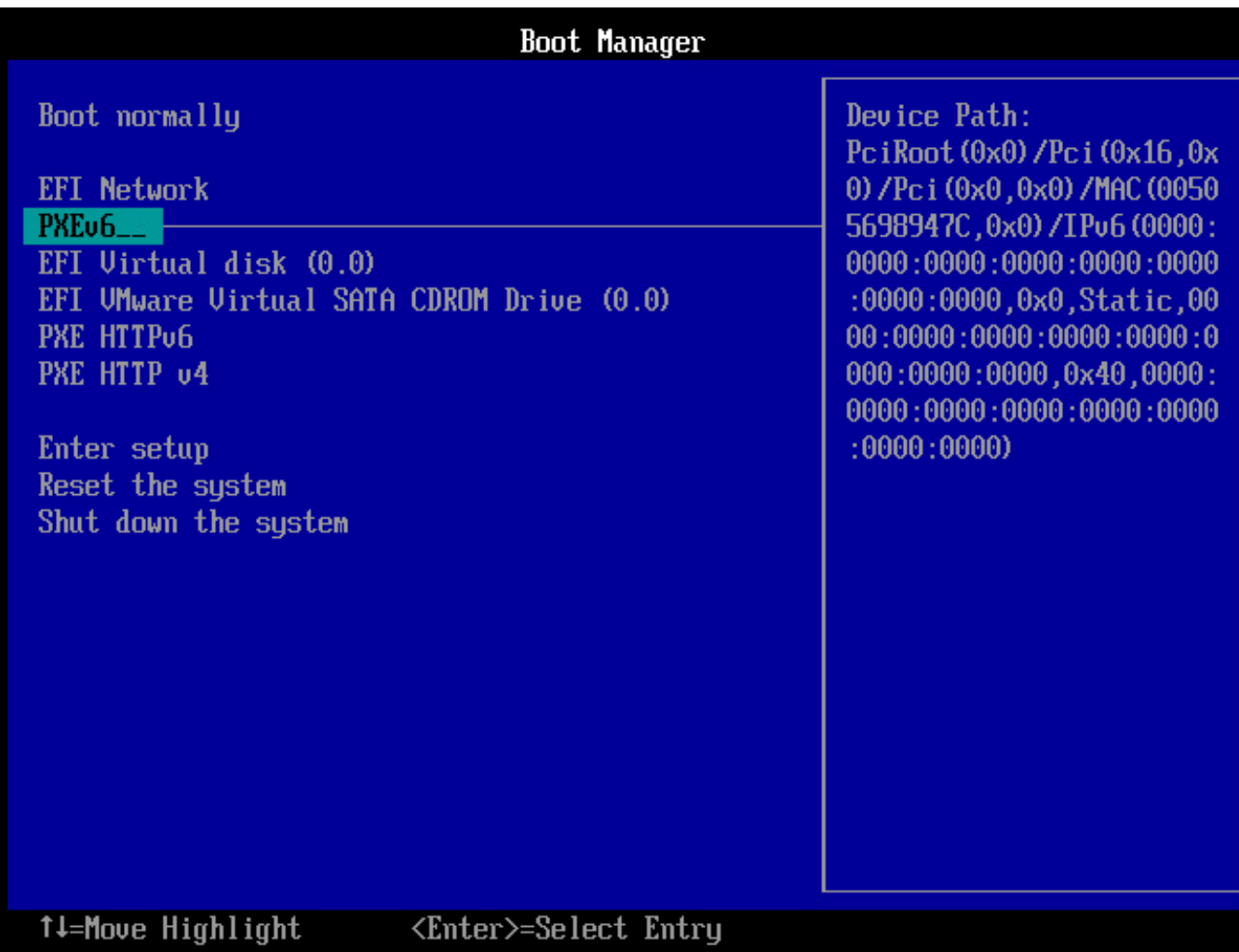
FILE_IN_CAB: Memory.dmp

EXCEPTION_PARAMETER1: 0000000000000000

EXCEPTION_PARAMETER2: 00000000046eece8

READ_ADDRESS: 00000000046eece8

VMware ESXi Firmware



Support IPv6 boot in firmware and can be configured in UEFI console.

▶ SnpDxe	File	DXE driver
▶ DpcDxe	File	DXE driver
▶ MnpDxe	File	DXE driver
▶ Ip4Dxe	File	DXE driver
▶ ArpDxe	File	DXE driver
▶ Udp4Dxe	File	DXE driver
▶ Dhcp4Dxe	File	DXE driver
▶ Mtftp4Dxe	File	DXE driver
▶ UefiPxeBcDxe	File	DXE driver
▶ Ip6Dxe	File	DXE driver
▶ Udp6Dxe	File	DXE driver
▶ Dhcp6Dxe	File	DXE driver
▶ Mtftp6Dxe	File	DXE driver
▶ TcpDxe	File	DXE driver
▶ DnsDxe	File	DXE driver
▶ HttpDxe	File	DXE driver
▶ HttpBootDxe	File	DXE driver
▶ HttpUtilitiesDxe	File	DXE driver

- CVE-2024-20688-PXE Bootloader BmpParseDhcpv6Packet ServerIdentifier stack out of bound write
- CVE-2024-20689-PXE Bootloader BmpParseDhcpv6Packet ClientIdentifier stack out of bound write

```

1  __int64 __fastcall BmpParseDhcpv6Packet(_DWORD *_pBuffer, unsigned int cbLimit)
2  {
3      // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5      memset(a3, 0, sizeof(DhcpPacket));
6      if ( cbLimit >= 4ui64 )
7      {
8          pBuffer = _pBuffer;
9          a3->word228 = (unsigned __int8)*pBuffer;
10         a3->dword330 = (*pBuffer >> 8) & 0FFFFFFF | a3->dword330 & 0xFF000000;
11         if ( PxeFindDhcpv6Option(1, 1, (__int64)pBuffer, cbLimit, &Src, v4) )
12         {
13             a3->word22A = v4[0]; // client identifier
14             memmove(&a3->pClientIdentifier, Src, v4[0]); // stack overflow
15         }
16         if ( PxeFindDhcpv6Option(2, 1, (__int64)pBuffer, cbLimit, &Src, v4) )
17         {
18             a3->word2AC = v4[0]; // server identifier
19             memmove(&a3->pServerIdentifier, Src, v4[0]); // stackoverflow
20         }
21     }
22 }

```

DhcpPacket binlpacket; // [rsp+E0h] [rbp-698h] BYREF

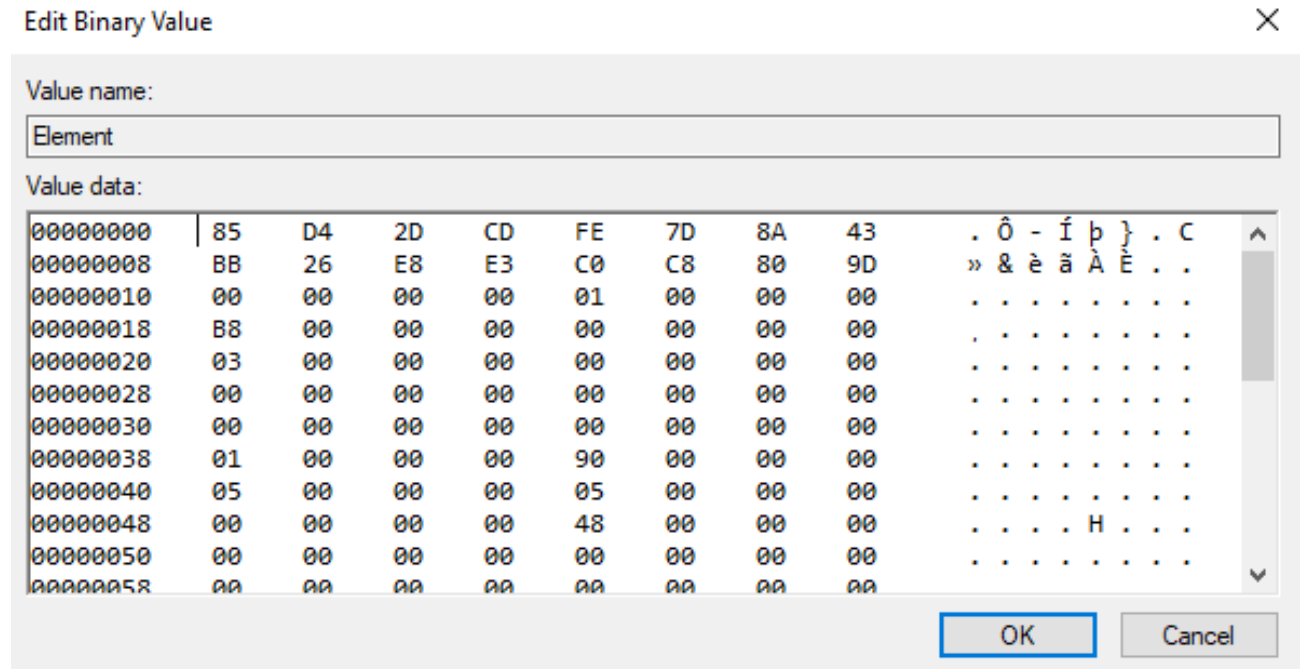
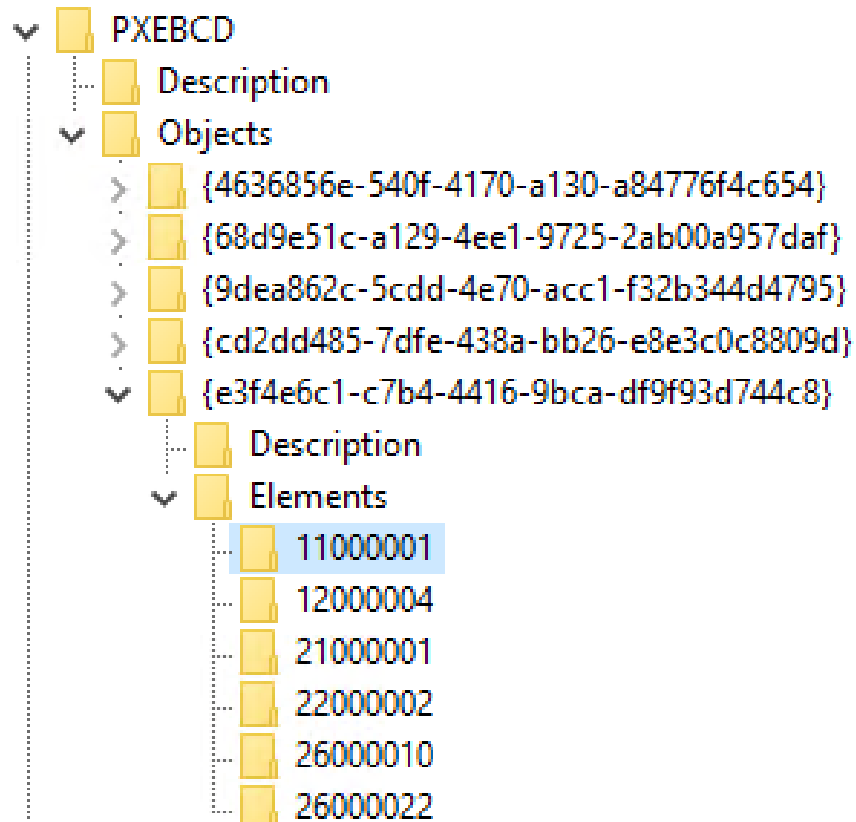
```

!!!! X64 Exception Type - 0D(#GP - General Protection) CPU Apic ID - 00000000 !!!!
ExceptionData - 0000000000000000
RIP - 4141414141414141, CS - 0000000000000038, RFLAGS - 0000000000000202
RAX - 00000000C0000240, RCX - 00000000C0000240, RDX - 0000000010133828
RBX - 0000000000000013, RSP - 000000003FE37510, RBP - 000000003FE375D0
RSI - 4242424242424242, RDI - 4242424242424242
R8 - 0000000000000000, R9 - 0000000000000000, R10 - 0000000010161170
R11 - 000000003FE36CA0, R12 - 0000000000000000, R13 - 0000000000000000
R14 - 0000000000000000, R15 - 0000000000000000
DS - 0000000000000030, ES - 0000000000000030, FS - 0000000000000030
GS - 0000000000000030, SS - 0000000000000030
CR0 - 0000000080010033, CR2 - 4141414141414141, CR3 - 000000003F801000
CR4 - 0000000000000068, CR8 - 0000000000000000
DR0 - 0000000000000000, DR1 - 0000000000000000, DR2 - 0000000000000000
DR3 - 0000000000000000, DR6 - 00000000FFFF0FF0, DR7 - 00000000000000400
GDTR - 000000003F5DC000 0000000000000047, LDTR - 0000000000000000
IDTR - 000000003EE5E018 00000000000000FF, TR - 0000000000000000
FXSAVE_STATE - 000000003FE37170

```

Register controlled by remote attacker.

- Registry Hive file
- Can be edit by regedit and API



- Bootloader CmpRemoveCellFromIndex Heap out of bound write

```

unsigned __int16 __fastcall CmpRemoveCellFromIndex(struct_CellPaged *CellPaged, unsigned __int16 CellIndex)
{
    __int64 v2; // r10
    unsigned __int16 result; // ax
    unsigned __int16 v4; // r8
    size_t v5; // r8
    struct_CellPaged *v6; // rdx
    struct_CellPaged *v7; // rcx
    unsigned __int16 v8; // dx

    v2 = CellIndex;
    result = CellPaged->word0 - 0x666C;
    if ( (result & 0xFDFF) != 0 )
    {
        v4 = CellPaged->CellCount - 1;
        CellPaged->CellCount = v4;
        if ( !v4 )
            return result;
        v5 = 4 * (v4 - (unsigned __int64)CellIndex);
        v6 = &CellPaged[CellIndex + 2];
        v7 = &CellPaged[v2 + 1];
    }
    else
    {
        v8 = CellPaged->CellCount - 1;
        CellPaged->CellCount = v8;
        if ( !v8 )
            return result;
        v5 = 8 * (v8 - v2);
        v6 = &CellPaged[2 * (unsigned int)(v2 + 1) + 1];
        v7 = &CellPaged[2 * v2 + 1];
    }
    return (unsigned __int16)memmove(v7, v6, v5);
}

```

`v5 = 4 * (v4 - (unsigned __int64)CellIndex);`

`return (unsigned __int16)memmove(v7, v6, v5);`

Say hello to bcdedit

BCDEDIT - Boot Configuration Data Store Editor

The Bcdedit.exe command-line tool modifies the boot configuration data store. The boot configuration data store contains boot configuration parameters and controls how the operating system is booted. These parameters were previously in the Boot.ini file (in BIOS-based operating systems) or in the nonvolatile RAM entries (in Extensible Firmware Interface-based operating systems). You can use Bcdedit.exe to add, delete, edit, and append entries in the boot configuration data store.

For detailed command and option information, type bcdedit.exe /? <command>. For example, to display detailed information about the /createstore command, type:

```
bcdedit.exe /? /createstore
```

For an alphabetical list of topics in this help file, run "bcdedit /? TOPICS".

Commands that operate on a store

=====

/store	Used to specify a BCD store other than the current system default.
/createstore	Creates a new and empty boot configuration data store.
/export	Exports the contents of the system store to a file. This file can be used later to restore the state of the system store.
/import	Restores the state of the system store using a backup file created with the /export command.
/sysstore	Sets the system store device (only affects EFI systems, does not persist across reboots, and is only used in cases where the system store device is ambiguous).

Commands that operate on entries in a store

=====

/copy	Makes copies of entries in the store.
/create	Creates new entries in the store.
/delete	Deletes entries from the store.
/mirror	Creates mirror of entries in the store.

Run bcdedit /? ID for information about identifiers used by these commands.

Commands that operate on entry options

=====

/deletevalue	Deletes entry options from the store.
/set	Sets entry option values in the store.

Run bcdedit /? TYPES for a list of datatypes used by these commands.

Run bcdedit /? FORMATS for a list of valid data formats.

Commands that control output

=====

/enum	Lists entries in the store.
/v	Command-line option that displays entry identifiers in full, rather than using names for well-known identifiers. Use /v by itself as a command to display entry identifiers in full for the ACTIVE type.

Running "bcdedit" by itself is equivalent to running "bcdedit /enum ACTIVE".

Commands that control the boot manager

=====

/bootsequence	Sets the one-time boot sequence for the boot manager.
/default	Sets the default entry that the boot manager will use.
/displayorder	Sets the order in which the boot manager displays the multiboot menu.
/timeout	Sets the boot manager time-out value.
/toolsdisplayorder	Sets the order in which the boot manager displays the tools menu.

Commands that control Emergency Management Services for a boot application

=====

/bootems	Enables or disables Emergency Management Services for a boot application.
/ems	Enables or disables Emergency Management Services for an operating system entry.
/emssettings	Sets the global Emergency Management Services parameters.

- Notes
 - DOS
 - Expression Web
 - Front Page
 - Independent Hardware Vendors
 - Independent Software Vendors
 - Office
 - Security
 - Web
 - Windows
 - Startup
 - Windows Vista and Higher
 - Boot Configuration Data
 - BCD Objects
 - BCD Elements**
 - The Advanced Boot Options Menu
 - The Edit Boot Options Menu
 - The Boot Status Data Log
 - Older Windows Versions
 - Kernel
 - Licensing
 - Internet Information Services (IIS)
 - Shell
 - Internet Explorer
 - Help
 - Debugging
 - Retro-Computing
 - Archive



indeed look to be complete, given its stated caveats, for Windows 8.

Global Elements

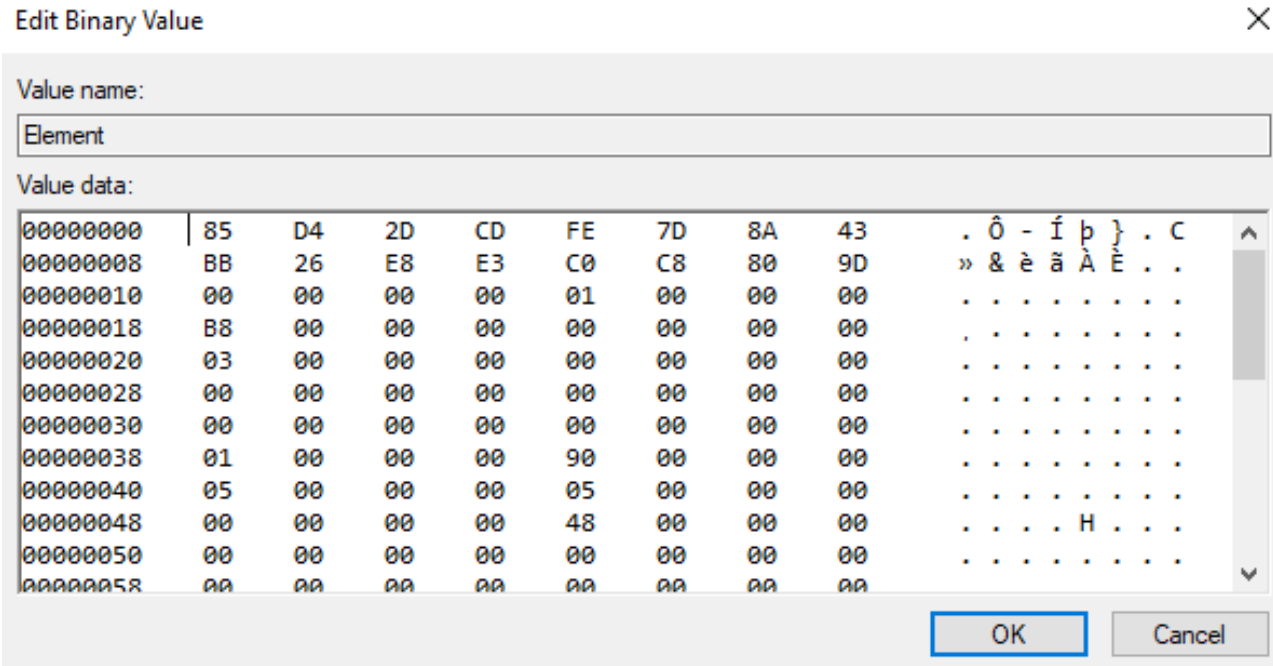
The following apply to all object types.

Library Elements

Before Windows 10, all the elements that can be in all types of object are Library elements.

Constant	Symbolic Names	Friendly Name	Format or Value	Versions
0x11000001	BcdLibraryDevice_ApplicationDevice BCDE_LIBRARY_TYPE_APPLICATION_DEVICE	device	device	6.0 and higher
0x12000002	BcdLibraryString_ApplicationPath BCDE_LIBRARY_TYPE_APPLICATION_PATH	path	string	6.0 and higher
0x12000004	BcdLibraryString_Description BCDE_LIBRARY_TYPE_DESCRIPTION	description	string	6.0 and higher
0x12000005	BcdLibraryString_PREFERREDLocale BCDE_LIBRARY_TYPE_PREFERRED_LOCALE	locale	string	6.0 and higher
0x14000006	BcdLibraryObjectList_InheritedObjects BCDE_LIBRARY_TYPE_INHERIT	inherit	GUID list	6.0 and higher
0x15000007	BcdLibraryInteger_TruncatePhysicalMemory BCDE_LIBRARY_TYPE_TRUNCATE_PHYSICAL_MEMORY	truncatememory	integer	6.0 and higher
0x14000008	BcdLibraryObjectList_RecoverySequence BCDE_LIBRARY_TYPE_RECOVERY_SEQUENCE	recoverysequence	GUID list	6.0 and higher
0x16000009	BcdLibraryBoolean_AutoRecoveryEnabled BCDE_LIBRARY_TYPE_AUTO_RECOVERY_ENABLED	recoveryenabled	boolean	6.0 and higher
0x1700000A	BcdLibraryIntegerList_BadMemoryList BCDE_LIBRARY_TYPE_BAD_MEMORY_LIST	badmemorylist	integer list	6.0 and higher
0x1600000B	BcdLibraryBoolean_AllowBadMemoryAccess BCDE_LIBRARY_TYPE_ALLOW_BAD_MEMORY_ACCESS	badmemoryaccess	boolean	32 and higher

- REG_BINARY
- Valid when unpacking from REG_BINARY format.
- Size gets checked
- Content in it is not checked



- BL_DEVICE_TYPE
 - DiskDevice = 0x0
 - LegacyPartitionDevice = 0x2
 - SerialDevice = 0x3
 - UdpDevice = 0x4
 - BootDevice = 0x5
 - PartitionDevice = 0x6
 - VmbusDevice = 0x7
 - LocateDevice = 0x8
 - UriDevice = 0x9
 - CompositeDevice = 0xA
 - CimfsDevice = 0xB

BiSanitizeRamdiskDevicesInDevice

CVE	Attack Surface	Finding method	Report title
CVE-2024-28896	BCD Element Processing	fuzzing	Bootloader BiSanitizeRamdiskDevicesInDevice RamDiskDevice stack OOB Write preauth RCE
CVE-2024-28897	BCD Element Processing	Audit	Bootloader BiSanitizeRamdiskDevicesInDevice LocateDevice invalid ParentOffset arbitrary memory write preauth RCE
CVE-2024-29061	BCD Element Processing	Audit	Bootloader BiSanitizeRamdiskDevicesInDevice FileDevice stack OOB Write preauth RCE
CVE-2024-26175	BCD Element Processing	Audit	Bootloader BiSanitizeRamdiskDevicesInDevice CimfsDevice Heap OOB write preauth RCE
CVE-2024-26175	BCD Element Processing	Audit	Bootloader BiSanitizeRamdiskDevicesInDevice RamDiskDevice Heap OOB write preauth RCE
CVE-2024-26175	BCD Element Processing	Audit	Bootloader BiSanitizeRamdiskDevicesInDevice CompositeDevice Heap OOB write preauth RCE

```
1 __int64 __fastcall __spoils<> BiSanitizeRamdiskDevicesInDevice(BL_DEVICE_DESCRIPTOR *Device)
2 {
3     // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5     v1 = 0;
6     for ( i = 0; ; ++i )
7     {
8         if ( !Device )
9             return v1;
10        if ( i == 8 )
11            return 0xC0000000;
12        DeviceType = Device->DeviceType;
13        if ( Device->DeviceType > 6 )
14            break;
15        if ( DeviceType != PartitionDevice )
16        {
17            if ( DeviceType == DiskDevice )
18                // 0
19            {
20                switch ( Device->field_10 )
21                {
22                case 3:
23                    Device->Flags |= 1u;
24                    *((_QWORD *)&Device->field_14 = 0i64; // oob 0x14
25                    *((_QWORD *)&Device->field_1C = 0i64; // oob 0x1c
26                    Device->field_24 = 0; // oob 0x24
27                    v8 = &Device->field_28;
28                    if ( v8 && *((_DWORD *)v8 + 1) > 0xCu )
29                    {
30                        Device = (BL_DEVICE_DESCRIPTOR *)((char *)v8 + 0xC); //
31                        // add offset 0x34, min size 0x48, align 0x60
32                        // add offset 0x34+0x34, min size 0x7c, align 0x80
33                    }
34                    continue;
35                case 5: // FileDevice
36                    if ( Device->ParentOffset < 0x10u )
37                        return 0xC0000000;
38                    Device = (BL_DEVICE_DESCRIPTOR *)((char *)Device + 0x20); // add offset 0x20, oob 0x34 0x3c 0x44
39                    continue;
40                case 6:
41                    Device = (BL_DEVICE_DESCRIPTOR *)((char *)Device + 0x28); // add offset 0x28 - oob 0x3c, 0x44, 0x4c
42                    continue;
43                }
44            LABEL_20:
45                Device = 0i64;
46                continue;
47            }
48            v4 = DeviceType - 1; // 2
49            if ( !v4 )
50                return v1; // 1
```

```
60     diskinfo = *(ramdisk *)&Device->field_20; // 2,6
61     type = _mm_cvtsi128_si32((__m128i *)&diskinfo.type);
62     if ( type == 3 ) // RamDiskDevice
63     {
64         Device = (BL_DEVICE_DESCRIPTOR *)&diskinfo.field_24;
65         v7 = diskinfo.field_1C <= 0xCu;
66     LABEL_24:
67         if ( v7 )
68             Device = 0i64;
69         continue;
70     }
71     if ( type == 5 ) // FileDevice
72     {
73         if ( diskinfo.field_8 < 0x10u )
74             return 0xC0000000;
75         v7 = diskinfo.field_8 <= 0xCu;
76         Device = (BL_DEVICE_DESCRIPTOR *)&diskinfo.field_10; // take stack addr as Device Descriptor Pointer
77         goto LABEL_24;
78     }
79     if ( type != 6 )
80         goto LABEL_20;
81     Device = (BL_DEVICE_DESCRIPTOR *)((char *)Device + 0x38); // next node, heap oob access
82 LABEL_35:
83     ;
84 }
85 switch ( DeviceType )
86 {
87     case VmbusDevice: // 7,8,9
88         return v1;
89     case LocateDevice: // locate device
90         if ( Device->field_10 != 1 || !Device->ParentOffset )
91             return v1;
92         if ( (BL_DEVICE_DESCRIPTOR *)((char *)Device + (unsigned int)Device->ParentOffset) < Device )
93             return 0xC0000000;
94         Device = (BL_DEVICE_DESCRIPTOR *)((char *)Device + (unsigned int)Device->ParentOffset);
95         goto LABEL_35;
96     case UriDevice:
97         return v1;
98 }
99 f ( (unsigned int)(DeviceType - 10) > 1 )
100     return 0xC0000000;
101     Device->Flags |= 1u;
102     f ( DeviceType == 0xA )
103         *((_QWORD *)&Device->field_24 = 0i64; // oob write to heap object
104     else
105         Device->field_14 = 0;
106     return v1;
107 }
```

Secure Boot Security Feature Bypass Vul

CVE-2024-26175
Security Vulnerability

Released: Apr 9, 2024

Assigning CNA: Microsoft

CVE.org link: [CVE-2024-26175](#)

Impact: Security Feature Bypass Max Severity: Important

Weakness: [CWE-125: Out-of-bounds Read](#)

CVSS Source: Microsoft

Vector String: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:

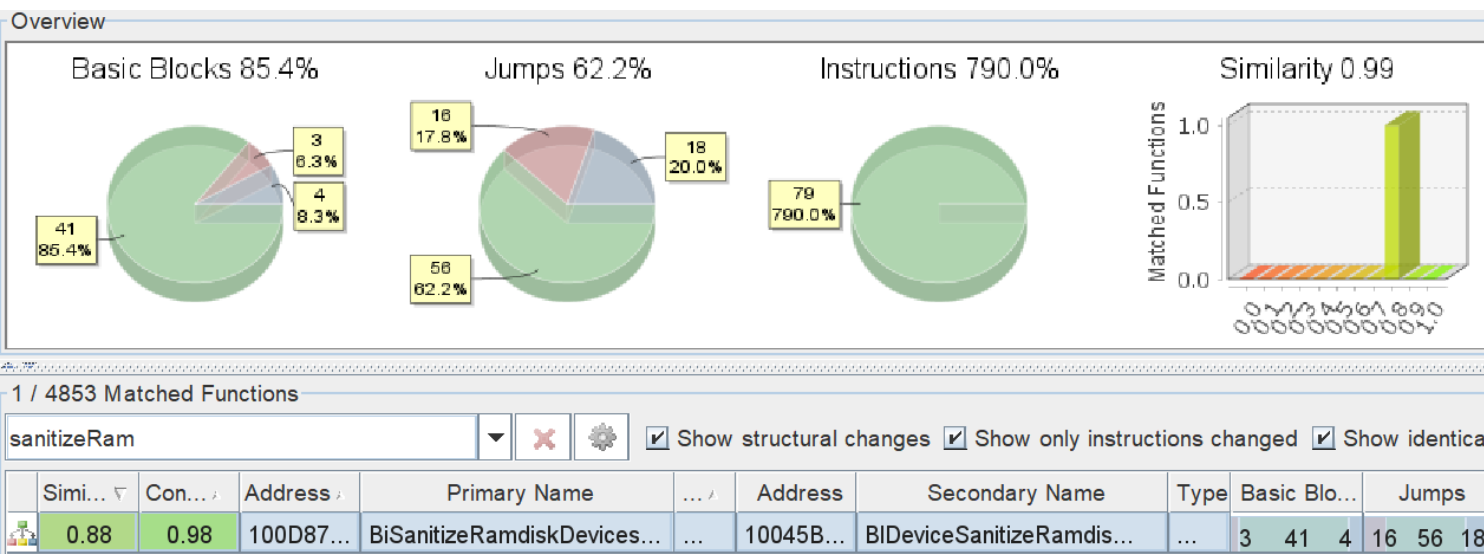
Metrics: CVSS:3.1 7.8 / 6.8 ⓘ

A design level change to fix(?) them all

Subject: RE: MSRC Case 83872 CRM:0022036386

Hi Azure,

The cases listed were deemed duplicates of an internally reported case. We determine duplicate by a few different criteria. In this case, a design level change was implemented to address them all and therefore deemed duplicate.



Define design level change:

Before: BiSanitizeRamdiskDevicesInDevice

After: BIDeviceSanitizeRamdiskDevicesInDevice

A design level change to fix(?) them all

CimfsDevice Heap OOB write

Release Date:

09/04/2024

Version:

OS Builds 22621.3447 and 22631.3447

```
kd> !mDvmbootmgfw
Browse full module list
start      end      module name
00000000`10000000 00000000`101e0000 bootmgfw C (pdb symbols) C:\Program Files\Windows Kits\10\Debuggers\
Loaded symbol image file: bootmgfw.efi
Mapped memory image file: C:\Program Files\Windows Kits\10\Debuggers\x64\sym\bootmgfw.efi\C5D5959B1e0000\bootmgfw
Image path: bootmgfw.efi
Image name: bootmgfw.efi
Browse all global symbols functions data
Image was built with /Brepro flag.
Timestamp: 00000000 (This is a reproducible build file hash, not a timestamp)
Checksum: 00000000
ImageSize: 001E0000
File version: 10.0.22621.3447
Product version: 10.0.22621.3447
File flags: 0 (Mask 3F)
File OS: 40004 NT Win32
File type: 1.0 App
File date: 00000000.00000000
Translations: 0409.04b0
Information from resource tables:
CompanyName: Microsoft Corporation
ProductName: Microsoft® Windows® Operating System
InternalName: bootmgr.exe
OriginalFilename: bootmgr.exe
ProductVersion: 10.0.22621.3447
FileVersion: 10.0.22621.3447 (WinBuild.160101.0800)
FileDescription: Boot Manager
LegalCopyright: © Microsoft Corporation. All rights reserved.

kd> r
rax=0000000000000000 rbx=0000000000000083
rdx=00000000000083af20 rsi=0000000000000000
rip=0000000000100c24ab rsp=00000000000046e2
r8=0000000000000000 r9=0000000000000000
r11=00000000000046e23d8 r12=0000000000000000 r13=0000000000000000
r14=000000000000838610 r15=0000000000000006
iopl=0         nv up ei pl nz na pe nc
cs=0028  ss=0008  ds=0030  es=0030  fs=0030  gs=0030             efl=00000202
bootmgfw!MmHapReportHeapCorruption+0x37:
00000000`100c24ab cc      int      3

kd> ki
#      Memory      Child-SP      RetAddr      Call Site
00 00000000`046e23e0 00000000`0083af20 bootmgfw!MmHapReportHeapCorruption+0x37
```

ProductVersion: 10.0.22621.3447
FileVersion: 10.0.22621.3447

bootmgfw!MmHapReportHeapCorruption+0x37:
00000000`100c24ab cc int

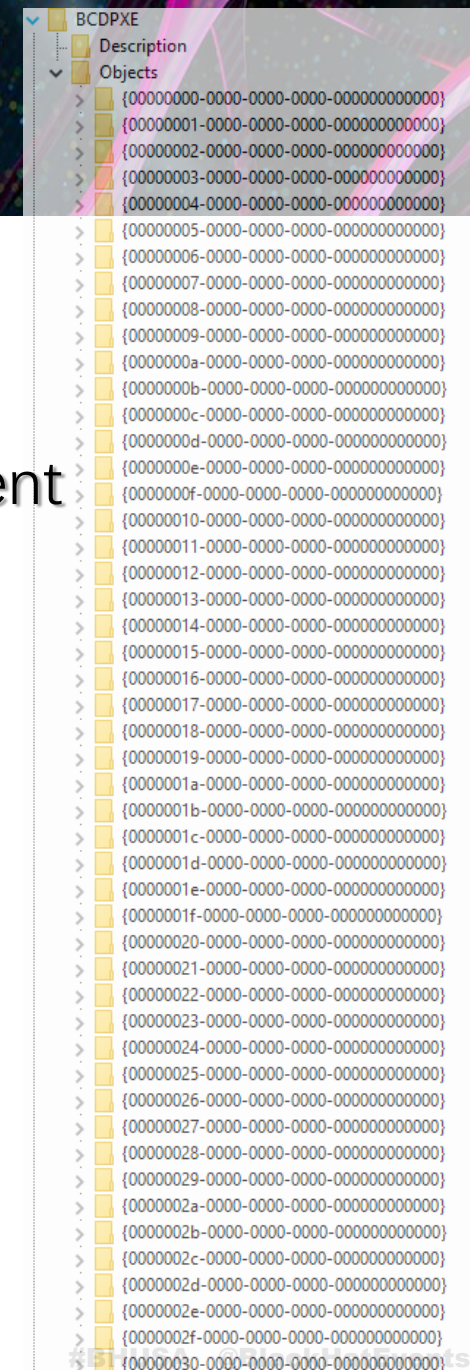
bootmgfw!MmHapReportHeapCorruption+0x37:
00000000`100c24ab cc int 3

It just need boundary check,
Microsoft won't miss it

They did fix it right?

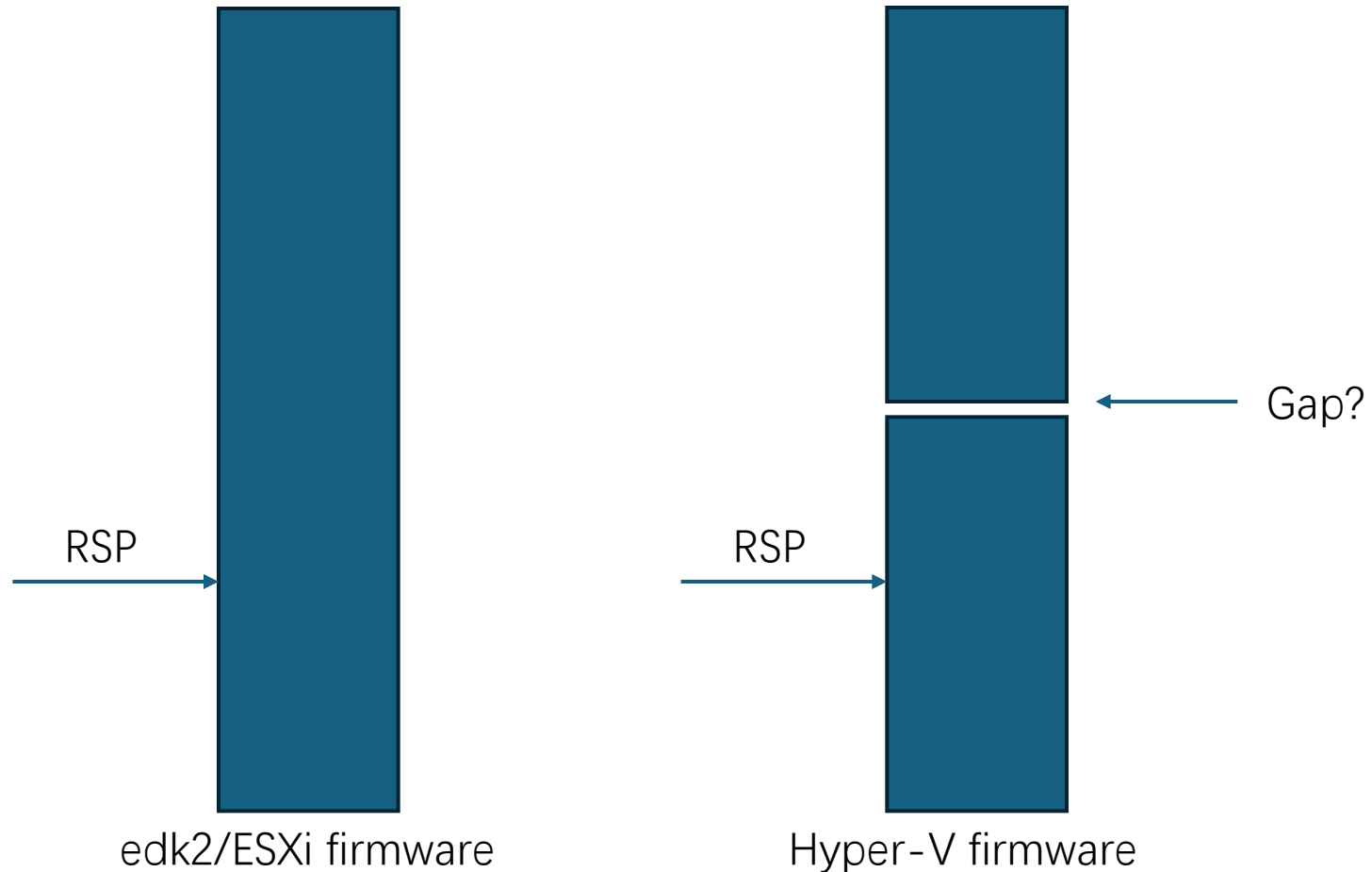
When bcdedit cry

- CVE-2024-28898
 - Recursive calling when enumerate the GUID element contained by the first 16 bytes GUID in the device element



About Recursive calling

- Stack memory layout



About Recursive calling

```
> 0xfecf8b1:  mov     DWORD PTR [r13+0x0],r12d
```

```
0xfecf8b5:  call    0xfecf9d7
```

```
0xfecf8ba:  mov     eax,r12d
```

```
0xfecf8bd:  add     rsp,0x20
```

```
0xfecf8c1:  pop     r12
```

```
[-----stack-----]
0000| 0xffbaf60 --> 0x0 (0x00000000)
0004| 0xffbaf64 --> 0x0 (0x00000000)
0008| 0xffbaf68 --> 0x2 --> 0x0 (0x00000000)
0012| 0xffbaf6c --> 0x0 (0x00000000)
0016| 0xffbaf70 --> 0xfe223818
0020| 0xffbaf74 --> 0x0 (0x00000000)
0024| 0xffbaf78 --> 0xffbb0fc --> 0x5 --> 0x0 (0x00000000)
0028| 0xffbaf7c --> 0x0 (0x00000000)
```

Legend: code, data, rodata, value

Stopped reason: SIGINT

0x00000000fecf8b1 in ?? ()

gdb-peda\$ info reg r13

r13 0xfe223818 0xfe223818

gdb-peda\$ x/10gx 0xfe223818

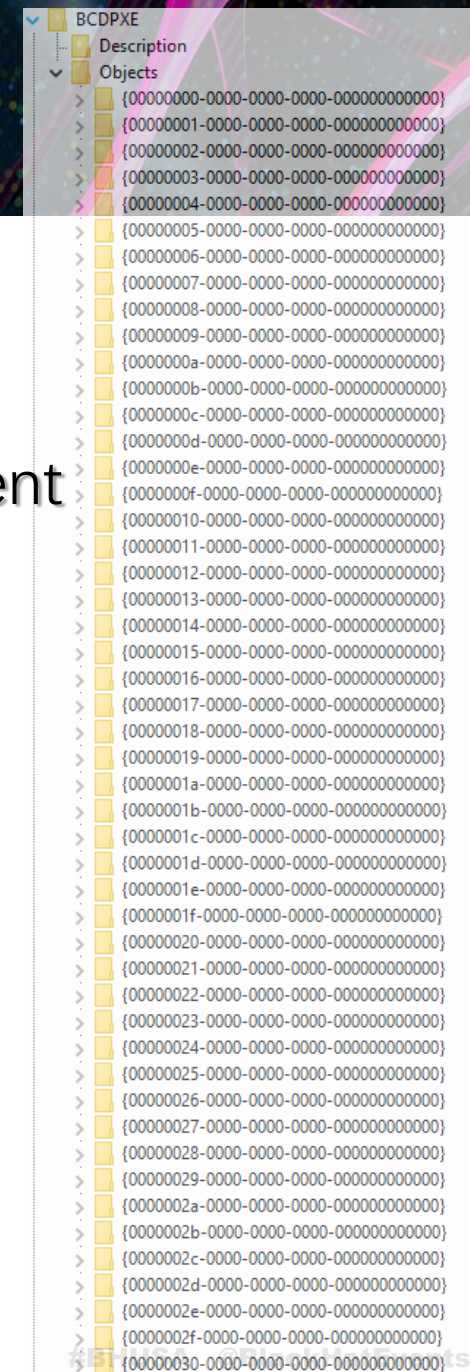
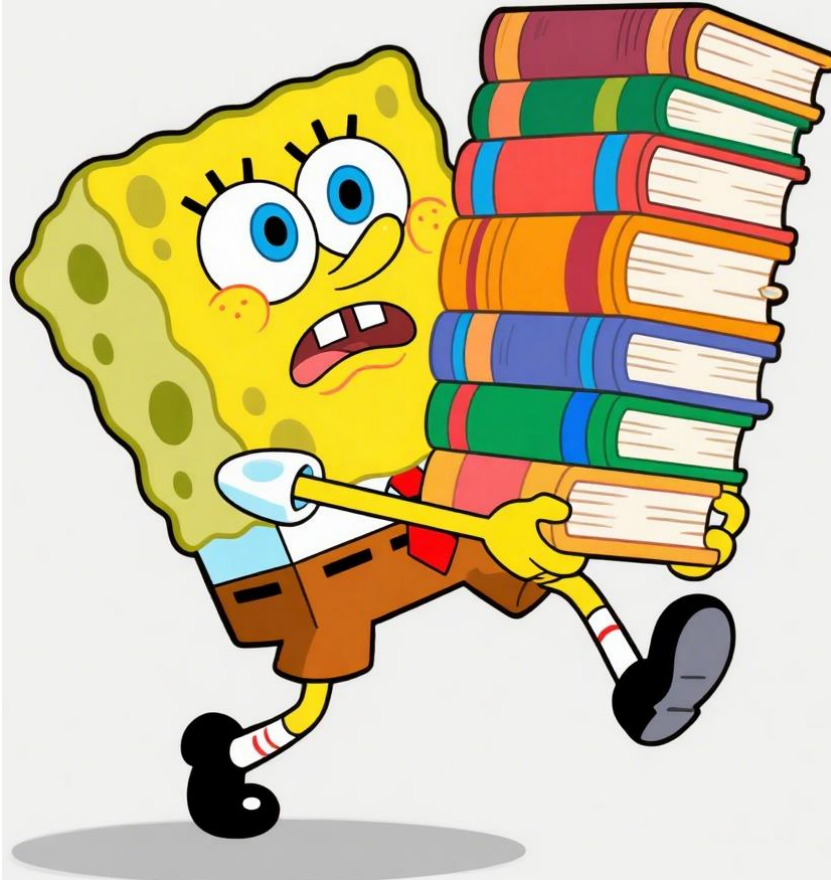
0xfe223818: Cannot access memory at address 0xfe223818

```
-----]
T direction overflow)
-----]
```

PROT 64-bit fluff=0000
L0 PROT 64-bit fluff=0000
PROT 64-bit fluff=0000

When bcdedit cry

- CVE-2024-28898
 - Recursive calling when enumerate the GUID element contained by the first 16 bytes GUID in the device element



When bcdedit cry

```
HKEY hiveKey;
LONG result = RegLoadKey(HKEY_USERS, "BCD123", hiveFilePath.c_str());
if (result != ERROR_SUCCESS) {
    std::cout << "Failed to load the registry hive. Error code: " << result << std::endl;
    getchar();
    return 1;
}

GUID guid = stringToGuid("{00000000-0000-0000-0000-000000000000}");

int cc = atoi(argv[2]);
for (int i=0;i<cc;++i)
{
    printf("i: %d\n", i);
    BCDCreateElement(HKEY_USERS, guid, i);
    puts("set done");
}

// Unload the registry hive
result = RegUnLoadKey(HKEY_USERS, "BCD123");
if (result != ERROR_SUCCESS) {
    std::cout << "Failed to unload the registry hive. Error code: " << result << std::endl;
    return 1;
}

std::cout << "Registry hive loaded, key value set, and hive unloaded successfully." <<
```

```
void BCDCreateElement(HKEY hRootKey, GUID NodeGUID, int i)
{
    HKEY hElementRoot;
    HKEY hElementElements;
    HKEY hElementElementsSubElement;
    HKEY hElementDescription;

    // Format BCD123\Objects\{s}\Elements\11000001
    memset(data1, 0, 0x100);
    NodeGUID.Data1 = i;
    sprintf((char*)&data1, "BCD123\\Objects\\%s", guidToString(&NodeGUID));
    puts((char *)data1);
    RegCreateKeyA(hRootKey, (LPCSTR)data1, &hElementRoot);

    sprintf((char*)&data1, "BCD123\\Objects\\%s\\Elements", guidToString(&NodeGUID));
    RegCreateKeyA(hRootKey, (LPCSTR)data1, &hElementElements);

    sprintf((char*)&data1, "BCD123\\Objects\\%s\\Description", guidToString(&NodeGUID));
    RegCreateKeyA(hRootKey, (LPCSTR)data1, &hElementDescription);

    sprintf((char*)&data1, "BCD123\\Objects\\%s\\Elements\\%x", guidToString(&NodeGUID), 0x11000001);
    // Create the sub element REG_BINARY
    RegCreateKeyA(hRootKey, (LPCSTR)data1, &hElementElementsSubElement);

    data1[0x110] = 0x5;
    data1[0x114] = 1;
    data1[0x118] = 0xC;
    *(DWORD *)&data1[0x100] = i+1;
    RegSetValueEx(hElementElementsSubElement, "Element", 0, REG_BINARY, &data1[0x100], 0x1C);

    RegCloseKey(hElementElementsSubElement);
    RegCloseKey(hElementDescription);
    RegCloseKey(hElementElements);
    RegCloseKey(hElementRoot);
}
```

HTTP Protocol

- Can be set by BCD element
- Not vulnerable when use firmware HTTP function
- Vulnerability exists when using firmware TCP and a hand-made HTTP parser in the bootloader.

- Bootloader HttppGetResponseTcp Integer overflow preauth RCE

```

65  v19 = httpTcp4_recvdata.DataLength - 3; // respond 2byte, integer overflow
66  if ( httpTcp4_recvdata.DataLength == 3 )
67      goto LABEL_40;
68  do
69  {
70      LODWORD(v19) = RtlCompareMemory((char *)Heap + v18, "\r\n\r\n", 4u);
71      if ( v19
72      {
73          *((_BYTE *)Heap + v18) = 0;
74          DataLength = httpTcp4_recvdata.DataLength;
75          v16 = (char *)Heap + v18 + 4;
76          v15 = (const char *)Heap;
77          v17 = httpTcp4_recvdata.DataLength - v18 - 4;
78      }
79      else
80      {
81          DataLength = httpTcp4_recvdata.DataLength;
82      }
83      ++v18;
84      v19 = DataLength - 3;
85  }
86  while ( v18 < (unsigned int)v19 );
87  if ( !v15 )
88  {

```

00194A82 HttppGetResponseTcp:68 (10195682)

- Golden Key's unlock attack
 - CVE-2016-3287 / CVE-2016-3320

A debug policy that was shipped with the HoloLens SDK was used in attack
RS1 and later is Secure, only down level operating systems are vulnerable
Must be an admin and have physical access to exploit the bug

Microsoft UEFI Security Updates

UEFI US Fall Plugfest – September 20 - 22, 2016
Presented by Microsoft

Scott Anderson, Suhas Manangi, Nate Nunez, Jeremiah Cox, Michael Anderson

- Private Key was not leaked
- This issue has no impact on Encryption or Bitlocker

And what it is

- For RT we had a debug policy to unlock individual devices for development
- The mechanism for debug policies was changed to simplify debug policies
- A design issue allowed the new policies to unlock old devices/OS versions
- A debug policy that was shipped with the HoloLens SDK was used in attack
- RS1 and later is Secure, only down level operating systems are vulnerable
- Must be an admin and have physical access to exploit the bug

- Case 83787

- Logic, by design, can be attack carried by unauthenticated attacker in network
- Ability to put everyone uses PXE boot at risk
- It only works in theory; this is one of only two cases among my submissions where I cannot bypass secure boot when I submitted.

Thank you again for submitting this issue to Microsoft. We determined that this behavior is considered to be by design. This case does not demonstrate a successful exploit with Secure Boot enabled.

We have closed this case.

If you have any questions, or additional information related to this report, please reply on this case thread.

Thank you very much for working with us.

Regards,

MSRC

- CVE-2024-29062

BmFwVerifySelfIntegrity SFB

- Exist because bootloader fetch bootmgfw.efi for verification from the bootdevice.

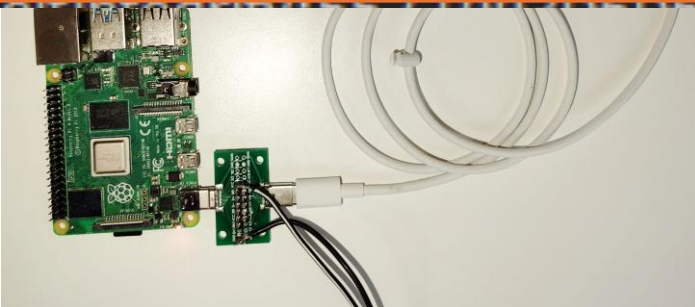
```

40 ApplicationEntry = B1GetApplicationEntry();
41 BootOptionDevice = B1GetBootOptionDevice(
42     ApplicationEntry->BcdData,
43     BCDE_LIBRARY_TYPE_APPLICATION_DEVICE,
44     &Device,
45     0i64);
46 v6 = _Device;
47 BootOptionString = BootOptionDevice;
48 if ( BootOptionDevice >= 0 )
49 {
50     if ( !_Device )
51         return (unsigned int)BootOptionString;
52     OpenFlags = 3i64;
53     if ( _Device->DeviceType != UdpDevice )
54         OpenFlags = 1i64;
55     BootOptionString = B1DeviceOpen(_Device, OpenFlags, &DeviceId);
56     if ( BootOptionString < 0 )
57     {
58         v8 = B1GetApplicationEntry();

```

```
BootOptionString = B1DeviceOpen(_Device, OpenFlags, &DeviceId);
```

```
BootOptionString = B1FileReadAtOffsetEx(FileId, image size, 0i64, (
```



CVE-2021-40045 – By taszk

```

71 BootOptionString = B1ImgAllocateImageBuffer(
72     (NET_FILE *)&pBuffer,
73     LODWORD(image_size.FileSize),
74     0xD000000A,
75     0,
76     0,
77     0);
78 if ( BootOptionString < 0 )
79 {
80     ImageBase = pBuffer;
81 }
82 else
83 {
84     ImageBase = pBuffer;
85     BootOptionString = B1FileReadAtOffsetEx(FileId, image size, 0i64, (int64)pBuffer, 0);

```

Windows Boot Code: Extensibility

- Another major threat for boot code is extensibility.
- For example, did you know some variants of boot manager support 10+ unique filesystems?
- Why do we expose this by default?

```
const PFILESYSTEM_TABLE FsTable[] = {  
    &NetRegisterFunctionTable,  
    &CompositeFsRegisterFunctionTable,  
    &VmbfsRegisterFunctionTable,  
    &CimFSRegisterFunctionTable,  
    &NtfsRegisterFunctionTable,  
    &EfiFsRegisterFunctionTable,  
    &FatRegisterFunctionTable,  
    &RefsRegisterFunctionTable,  
    &FppRegisterFunctionTable,  
    &WimRegisterFunctionTable,  
    &UdfsRegisterFunctionTable,  
    &EtfRegisterFunctionTable,  
    NULL  
};
```

blackhat® BRIEFINGS Bad Fixup

```
1 int __thiscall WimpFixupRoot(WIM_STRUCTURE_CONTEXT *WimContext)
2 {
3     // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5     if ( WimContext->BootMetaDataLen < 8u )
6         return 0xC0000098;
7     BootMetaData = (SecurityBlockDisk *)WimContext->BootMetaData;
8     SecurityBlockLength = BootMetaData->TotalLength;
9     if ( !BootMetaData->TotalLength )
10         SecurityBlockLength = 8;
11     if ( SecurityBlockLength < 8 || RtlULongPtrAdd((ULONG_PTR)BootMetaData, SecurityBlockLength, &uAugend) < 0 )
12         return 0xC0000098;
13     WimContext->RootDirEntry = (uAugend + 7) & 0xFFFFFFF8;
14     return 0;
15 }
```

```
int __thiscall WimpFixupRoot
```

```
void __fastcall __spoils<> FixupDirEntry(
```

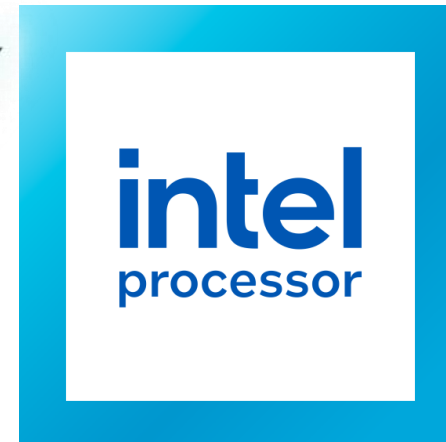
```
void __fastcall __spoils<> FixupDirEntry(WIM_STRUCTURE_CONTEXT *WimContext, DIRENTRY *DirEntry)
2 {
3     // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5     MetadataLimit = WimContext->BootMetaData + (unsigned int)WimContext->BootMetaDataLen;
6     if ( DirEntry->flags >= 0 )
7     {
8         EndOfLengthFieldPointer = &DirEntry->dwAttributes;
9         if ( &DirEntry->dwAttributes >= (_DWORD *)DirEntry && (unsigned __int64)EndOfLengthFieldPointer <= Me
10     {
11         liLength = DirEntry->liLength; // WimDirEntry should be at least 0x68 in size
12         if ( DirEntry->liLength )
13         {
14             if ( !HIDWORD(DirEntry->liLength) && (unsigned int)(liLength - 0x68) <= 0xFF97 )
15             {
16                 _NextEntry = (DIRENTRY *)((char *)DirEntry + liLength);
17                 if ( _NextEntry >= DirEntry && (unsigned __int64)_NextEntry <= MetadataLimit )
18                 {
19                     if ( DirEntry->wStreams )
20                     {
21                         wStreams = (unsigned __int16)DirEntry->wStreams;
22                         *(_QWORD *)&DirEntry->field_20 = _NextEntry; // pointer write at offset 0x20, takes 8 bytes
23                         fixedLimit = WimContext->BootMetaData + (unsigned int)WimContext->BootMetaDataLen;
24                         while...
25                     }
26                     else // no wStreams
27                     {
28                         *(_QWORD *)&DirEntry->field_20 = 0i64; // clear flags? not expected?
29 LABEL_21:
30                         DirEntry->liLength = (_int64)_NextEntry;
31                         if ( !_NextEntry->liLength )
32                             DirEntry->liLength = 0i64;
33                         if ( (*EndOfLengthFieldPointer & 0x10) == 0 ) // FILE_ATTRIBUTE_DIRECTORY
34                             goto complete;
35                         if ( !LODWORD(DirEntry->liSubdirOffset) )
36                         {
37                             DirEntry->liSubdirOffset = 0i64;
38 complete:
39                             DirEntry->flags |= 0x80000000; // oob write here
40                             return;
41                         }
42                         NextEntry = (DIRENTRY *) (WimContext->BootMetaData + LODWORD(DirEntry->liSubdirOffset));
43                         if ( (unsigned __int64)NextEntry >= WimContext->BootMetaData
44                             && (unsigned __int64)NextEntry < MetadataLimit
45                             && &NextEntry->dwAttributes >= (_DWORD *)NextEntry //
46                             // this check is wrong, dwAttribute offset is 8
47                             // however, the subsequent code will write to 0x24
48                             && (unsigned __int64)&NextEntry->dwAttributes <= MetadataLimit )
49                         {
50                             DirEntry->liSubdirOffset = (DIRENTRY *)((unsigned __int64)NextEntry & -(int64)NextEntry
500A456C FixupDirEntry:1 (100A516C)
```

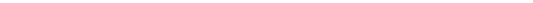
CVE	method	Title
CVE-2024-37987	fuzzing	FixupDirEntry wStream invalid 64bit pointer
CVE-2024-37988	audit	WimpFixupRoot invalid WIM SecurityBlock size check heap OOB write
CVE-2024-37989	audit	WimpFixupRoot lack of Directory attribute check arbitrary memory write
CVE-2024-38010	audit	FixupDirEntry Directory File NextEntry invalid check heap OOB write
	- fuzzing	WimpRead invalid chunk error deadlock preauth DoS
	- fuzzing	WimpReadResource div zero preauth DoS
	?audit	WimpFixupRoot lack of flags check arbitrary memory write

- AFLplusplus
 - NYX mode
- AFL++ - Free mutator
- NYX – Fast snapshot
- Intel PT – Code Coverage



NYX



- 

Payload segment

Tips to fuzz filesystem

- Filesystem itself is a code coverage amplifier
 - fuzzing use code basic block bitmap to collect coverage
 - To reach same logic in code, all roads can lead you to Rome

- Fuzzing approach
 - Reversing
 - Understanding
 - Fuzzing
 - Conduct hot patching on vulnerability
 - Repeat

- Result: 16 reports in 5 days, 5 by audit, 11 by fuzzing

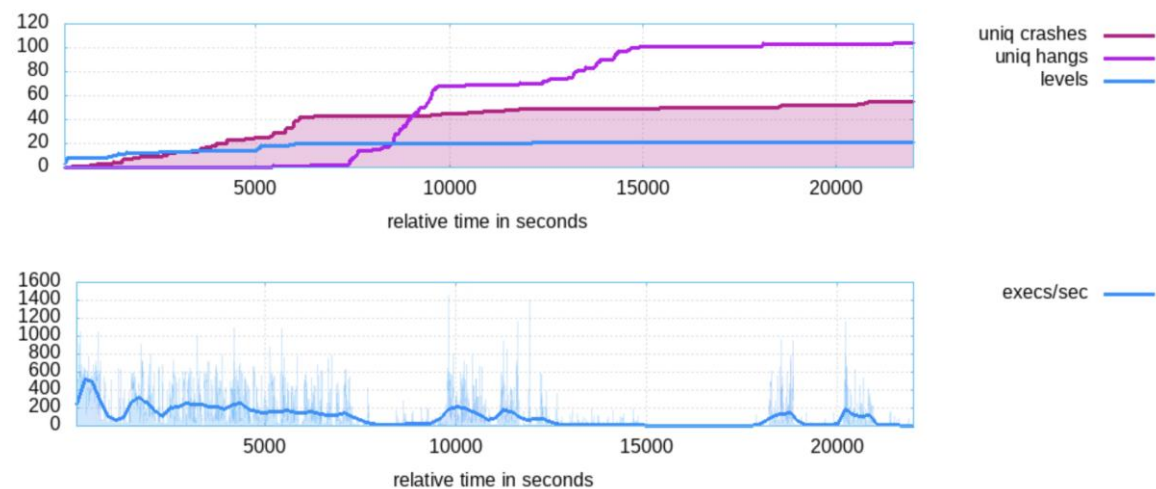
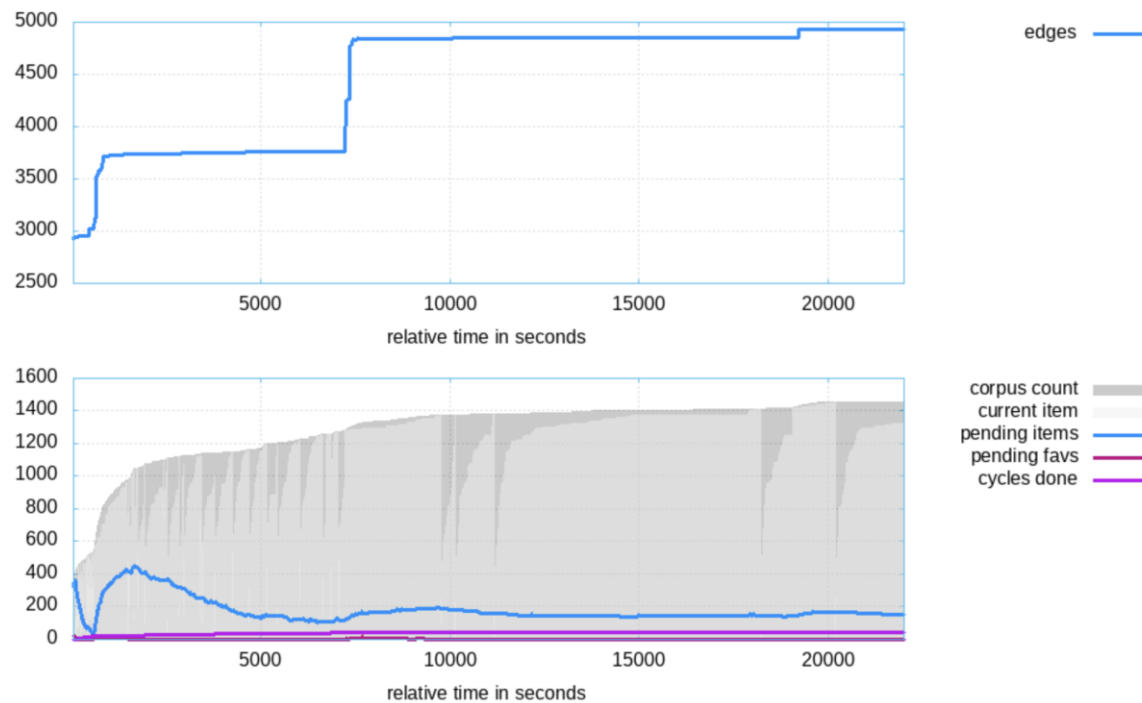


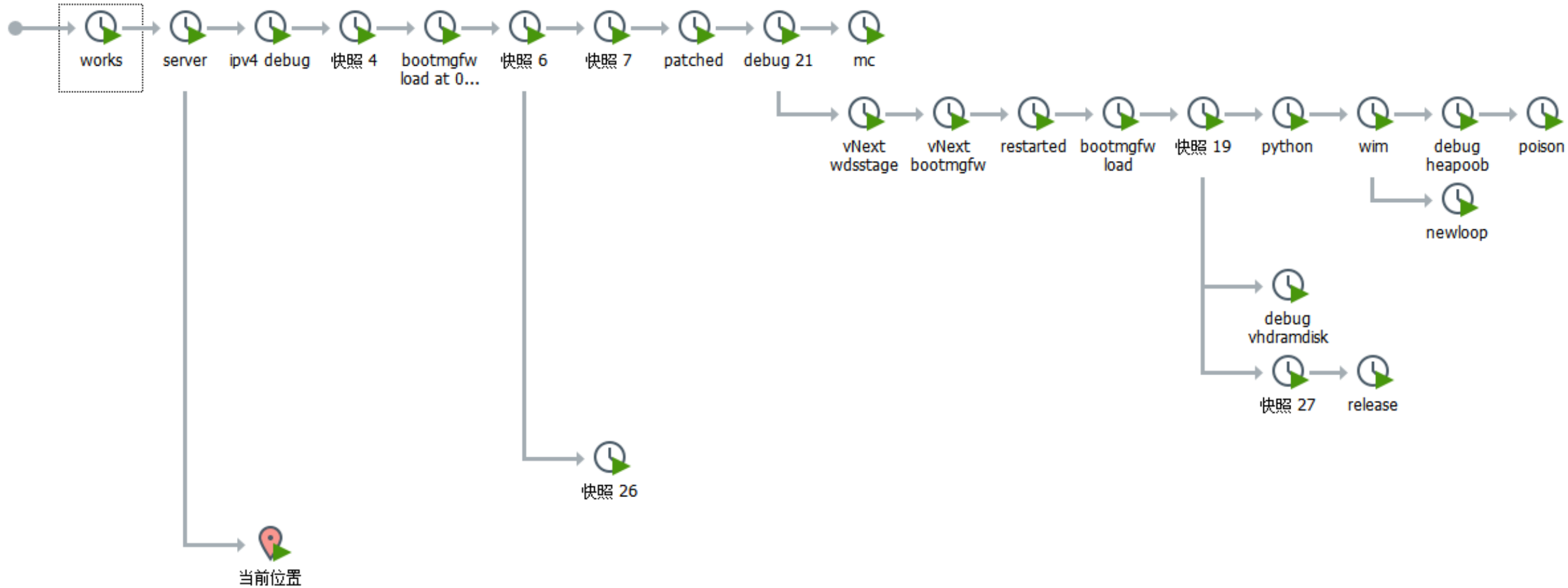
```
american fuzzy lop ++4.09a {0} (./nyx_mode/efi/ntfs) [fast] - Nyx
process timing      overall results
  run time : 23 days, 3 hrs, 30 min, 31 sec    cycles done : 477
  last new find : 0 days, 10 hrs, 47 min, 19 sec corpus count : 956
  last saved crash : 13 days, 3 hrs, 20 min, 50 sec saved crashes : 4
  last saved hang : 3 days, 10 hrs, 6 min, 0 sec   saved hangs : 20
cycle progress      map coverage
now processing : 682.419 (71.3%)    map density : 2.24% / 3.38%
runs timed out : 0 (0.00%)         count coverage : 3.17 bits/tuple
stage progress      findings in depth
now trying : splice 9              favored items : 171 (17.89%)
stage execs : 40/86 (46.51%)       new edges on : 235 (24.58%)
total execs : 234M                 total crashes : 15 (4 saved)
exec speed : 153.6/sec             total tmouts : 523 (0 saved)
fuzzing strategy yields      item geometry
bit flips : disabled (default, enable with -D) levels : 14
byte flips : disabled (default, enable with -D) pending : 0
arithmetics : disabled (default, enable with -D) pend fav : 0
known ints : disabled (default, enable with -D) own finds : 416
dictionary : n/a                imported : 538
havoc/splice : 239/81.6M, 181/152M stability : 100.00%
py/custom/rq : unused, unused, unused, unused
trim/eff : disabled, disabled
strategy: exploit      state: in progress      [cpu000: 10%]
```

```
american fuzzy lop ++4.09a {0} (./nyx_mode/efi/ntfs) [fast] - Nyx
process timing      overall results
  run time : 23 days, 3 hrs, 28 min, 46 sec    cycles done : 402
  last new find : 0 days, 18 hrs, 49 min, 15 sec corpus count : 1291
  last saved crash : 13 days, 12 hrs, 36 min, 26 sec saved crashes : 3
  last saved hang : 0 days, 20 hrs, 19 min, 7 sec   saved hangs : 55
cycle progress      map coverage
now processing : 1160.131 (89.9%)    map density : 1.91% / 3.38%
runs timed out : 0 (0.00%)         count coverage : 3.80 bits/tuple
stage progress      findings in depth
now trying : splice 8              favored items : 158 (12.24%)
stage execs : 16/172 (9.30%)       new edges on : 234 (18.13%)
total execs : 148M                 total crashes : 4 (3 saved)
exec speed : 210.6/sec             total tmouts : 572 (0 saved)
fuzzing strategy yields      item geometry
bit flips : disabled (default, enable with -D) levels : 34
byte flips : disabled (default, enable with -D) pending : 0
arithmetics : disabled (default, enable with -D) pend fav : 0
known ints : disabled (default, enable with -D) own finds : 1289
dictionary : n/a                imported : 0
havoc/splice : 806/52.4M, 486/96.5M stability : 100.00%
py/custom/rq : unused, unused, unused, unused
trim/eff : disabled, disabled
strategy: exploit      state: in progress      [cpu001: 8%]
```

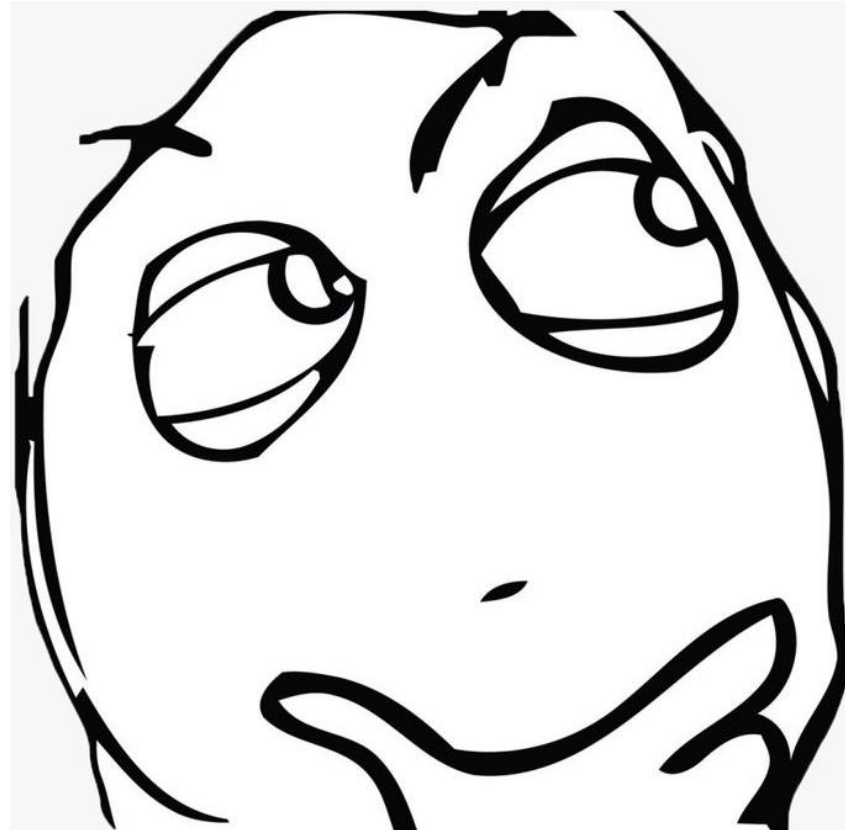
```
american fuzzy lop ++4.09a {0} (./nyx_mode/efi/ntfs) [fast] - Nyx
process timing      overall results
  run time : 23 days, 3 hrs, 28 min, 6 sec     cycles done : 358
  last new find : 0 days, 12 hrs, 29 min, 9 sec corpus count : 1139
  last saved crash : 20 days, 19 hrs, 16 min, 26 sec saved crashes : 2
  last saved hang : 1 days, 16 hrs, 5 min, 37 sec   saved hangs : 82
cycle progress      map coverage
now processing : 1072.75 (94.1%)    map density : 2.25% / 3.38%
runs timed out : 0 (0.00%)         count coverage : 3.43 bits/tuple
stage progress      findings in depth
now trying : splice 12             favored items : 159 (13.96%)
stage execs : 6/14 (42.86%)       new edges on : 232 (20.37%)
total execs : 125M                 total crashes : 13 (2 saved)
exec speed : 2.53/sec (zzzz...)    total tmouts : 1738 (0 saved)
fuzzing strategy yields      item geometry
bit flips : disabled (default, enable with -D) levels : 24
byte flips : disabled (default, enable with -D) pending : 0
arithmetics : disabled (default, enable with -D) pend fav : 0
known ints : disabled (default, enable with -D) own finds : 1137
dictionary : n/a                imported : 0
havoc/splice : 723/44.2M, 416/81.4M stability : 100.00%
py/custom/rq : unused, unused, unused, unused
trim/eff : disabled, disabled
strategy: exploit      state: in progress      [cpu003: 9%]
```

```
american fuzzy lop ++4.09a {0} (./nyx_mode/efi/ntfs) [fast] - Nyx
process timing      overall results
  run time : 23 days, 3 hrs, 28 min, 8 sec     cycles done : 420
  last new find : 0 days, 9 hrs, 31 min, 4 sec  corpus count : 1100
  last saved crash : 12 days, 4 hrs, 5 min, 7 sec saved crashes : 4
  last saved hang : 2 days, 21 hrs, 9 min, 20 sec   saved hangs : 18
cycle progress      map coverage
now processing : 407.331 (37.0%)    map density : 2.30% / 3.37%
runs timed out : 0 (0.00%)         count coverage : 3.39 bits/tuple
stage progress      findings in depth
now trying : splice 11             favored items : 167 (15.18%)
stage execs : 81/129 (62.79%)     new edges on : 230 (20.91%)
total execs : 130M                 total crashes : 10 (4 saved)
exec speed : 155.2/sec             total tmouts : 286 (0 saved)
fuzzing strategy yields      item geometry
bit flips : disabled (default, enable with -D) levels : 27
byte flips : disabled (default, enable with -D) pending : 0
arithmetics : disabled (default, enable with -D) pend fav : 0
known ints : disabled (default, enable with -D) own finds : 1098
dictionary : n/a                imported : 0
havoc/splice : 684/45.7M, 418/84.4M stability : 100.00%
py/custom/rq : unused, unused, unused, unused
trim/eff : disabled, disabled
strategy: exploit      state: in progress      [cpu002: 9%]
```





- Why do I need an infoleak to exploit the vulnerabilities?
 - Because there's ASLR on bootmgfw.efi



- What if I can bypass ASLR as if it does not exist from the start?

```
STACK_TEXT:
00000000`046f06d8 00000000`10062e01 : 00000000`0072fe00 00000000`0072fe00 00000000`0072fe40 00000000`0072b5f0 : bootmgfw!memcpy+0x203
00000000`046f06e0 00000000`1005549c : 00000000`0072fe00 00000000`046f0830 00000000`102a5550 00000000`0072fe40 : bootmgfw!UriOpen+0x69
00000000`046f0710 00000000`10053f5f : 00000000`00000000 00000000`00000001 00000000`00000000 00000000`00000000 : bootmgfw!BlpDeviceOpen+0x240
00000000`046f0780 00000000`1003ed83 : 00000000`00000001 00000000`00000000 00000000`00000000 00000000`00000000 : bootmgfw!BlDeviceOpen+0x5b
00000000`046f07b0 00000000`1003f0b4 : 00000000`00000001 00000000`00000000 00000000`ffffffff 00000000`00000000 : bootmgfw!BmpSecureBootInitializePolicy+0x7b
00000000`046f0860 00000000`10033683 : 00000000`00738f00 00000000`00000001 00000000`00000000 00000000`00000000 : bootmgfw!BmSecureBootInitializeMachinePolicy+0x60
00000000`046f08d0 00000000`100330ad : 00000000`102a5760 00000000`3f058300 00000000`00000000 00000000`00738f60 : bootmgfw!BmMain+0x427
00000000`046f0a90 00000000`046fb11f : 00000000`00000000 00000000`3f058818 00000000`00000001 00000000`03058001 : bootmgfw!EfiEntry+0x1d
00000000`046f0ac0 00000000`00000000 : 00000000`3f058818 00000000`00000001 00000000`03058001 00000000`00000000 : 0x46fb11f
```

SYMBOL_NAME: ANALYSIS_INCONCLUSIVE

MODULE_NAME: [Unknown Module](#)

IMAGE_NAME: Unknown_Image

STACK_COMMAND: .thread ; .cxr ; kb

FAILURE_BUCKET_ID: INVALID_KERNEL_CONTEXT_0x1E_c0000005_R

OSPLATFORM_TYPE: x64

OSNAME: Windows 8.1

FAILURE_ID_HASH: {7d47b74f-ccf1-b9cf-626b-4a8a12b8bf54}

Followup: MachineOwner

kd> lmDvmbbootmgfw

[Browse full module list](#)

start	end	module name
00000000`10000000	00000000`102cb000	bootmgfw C

(pdb symbols)

C:\Program Files\Windows Kits\10\Debuggers\x64\sym\bootmgfw.pdb\8EF01D7F670B27C1727E5CF57C

Unauthenticated Attacker
from network



Security boot chains remains intact and complete



UEFI PXE

Firmware bootloader
verification



Microsoft signed Bootloader

Protocol
vulnerabilities

BCD Element Processing
vulnerabilities

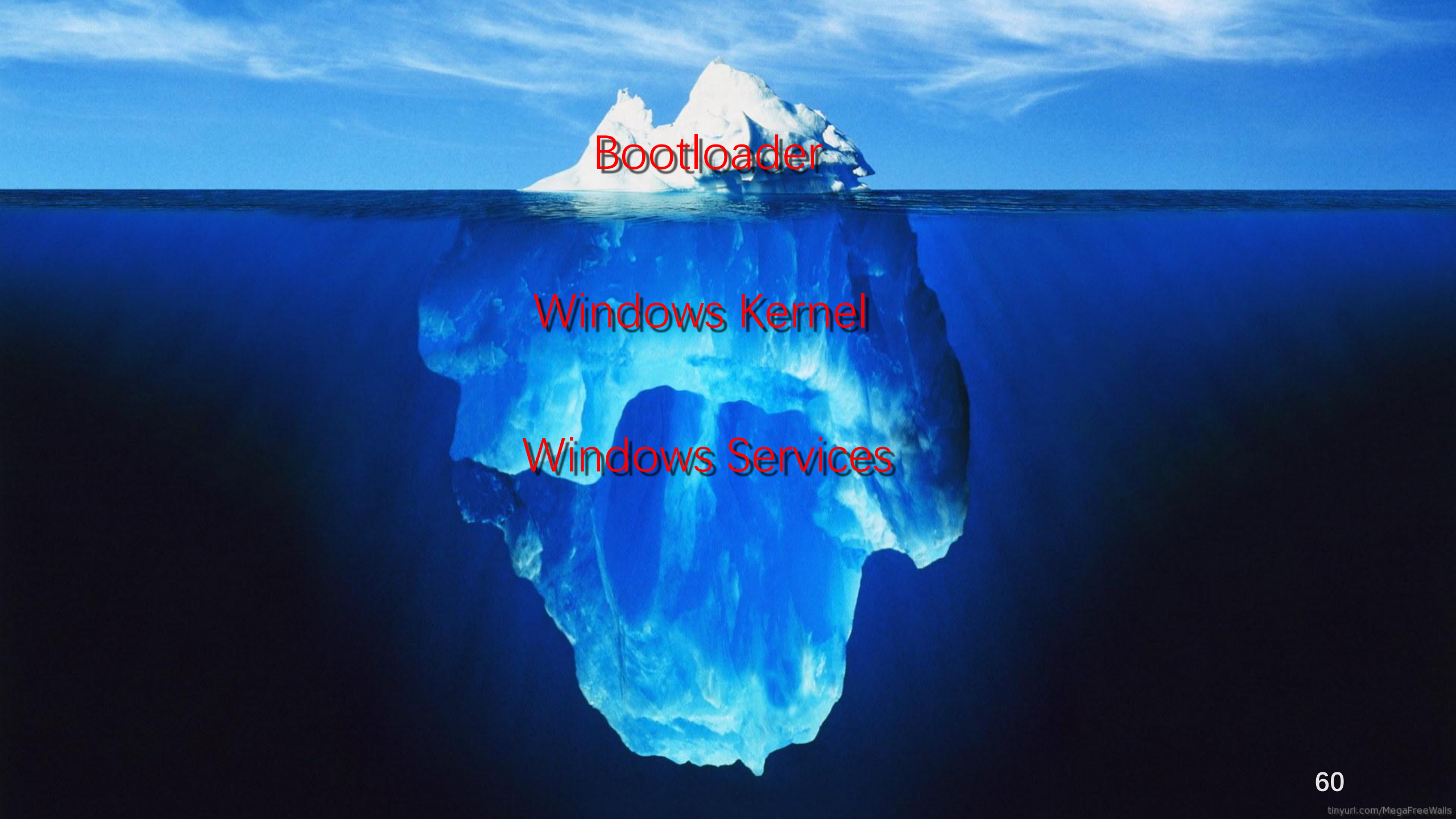
Filesystem
vulnerabilities

Memory corruption

Remote code execution

Security feature bypass

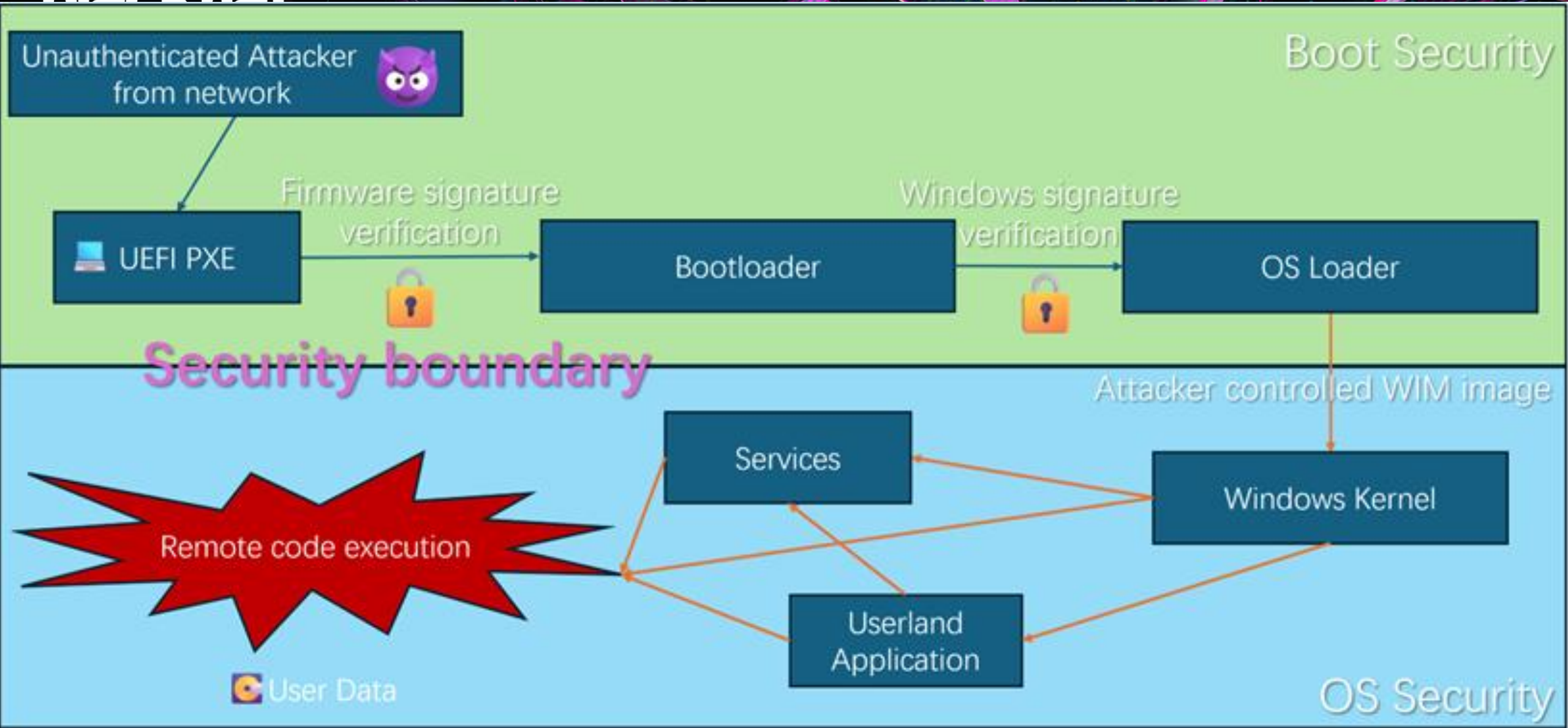
- Background
- Attack surface in bootloader
 - Network protocol
 - BCD Registry
 - Security Policy
 - Filesystem
 - Logic flaw
- **Attack surface beyond bootloader**
- Future Work & Take Aways



Bootloader

Windows Kernel

Windows Services



Up-to date
OS Binary



Default
Configuration

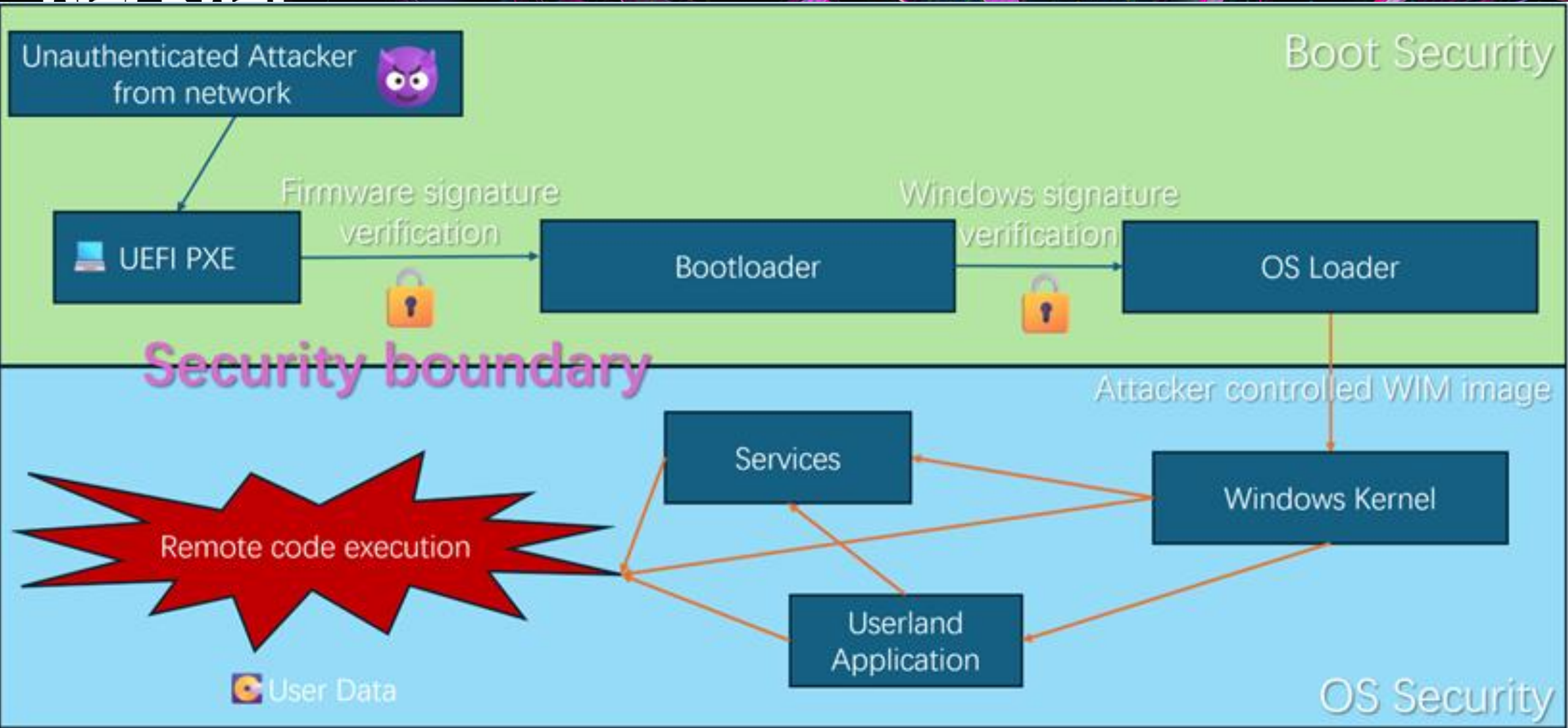


Up-to date
OS Binary

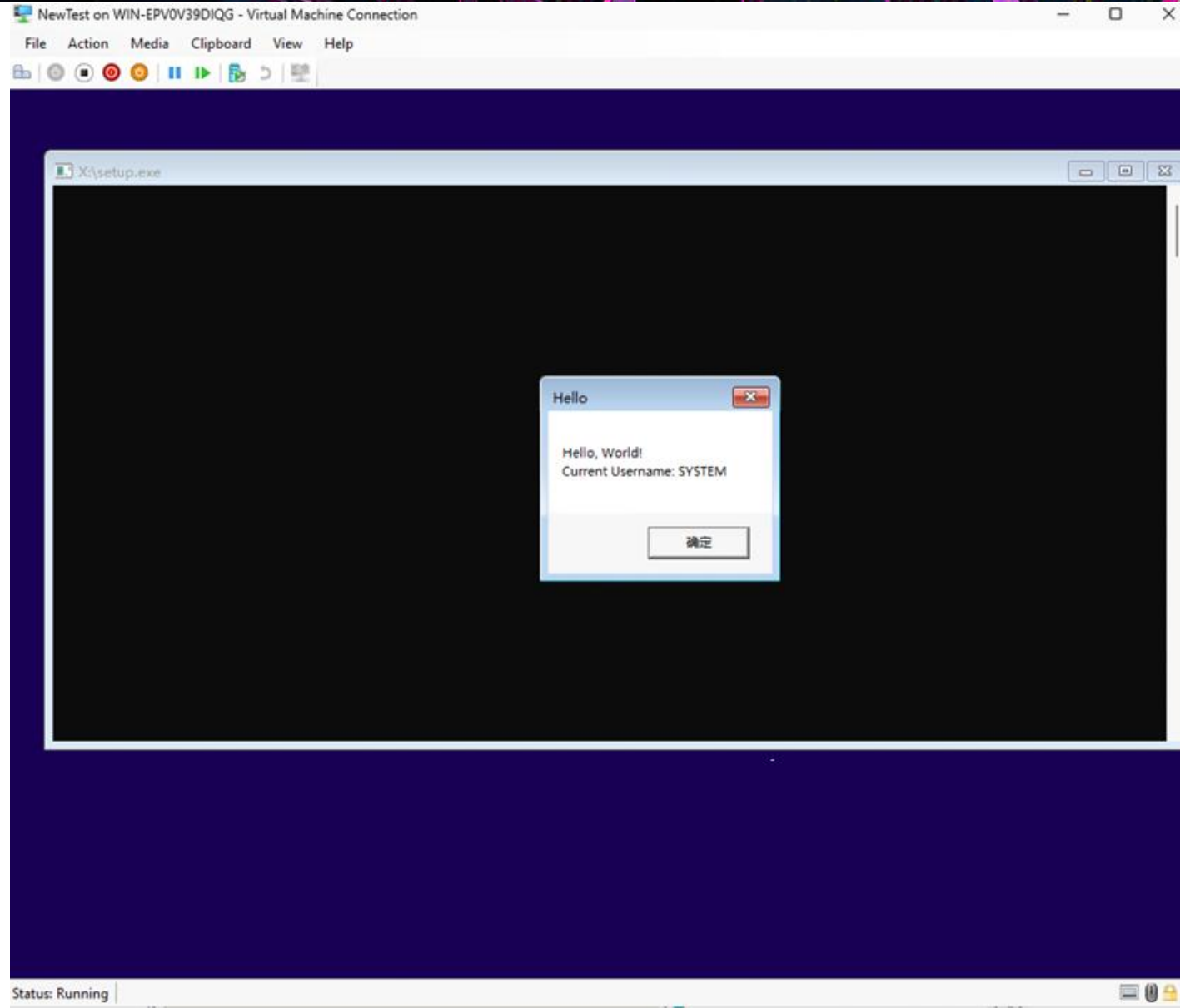


Attacker compromised
Configuration

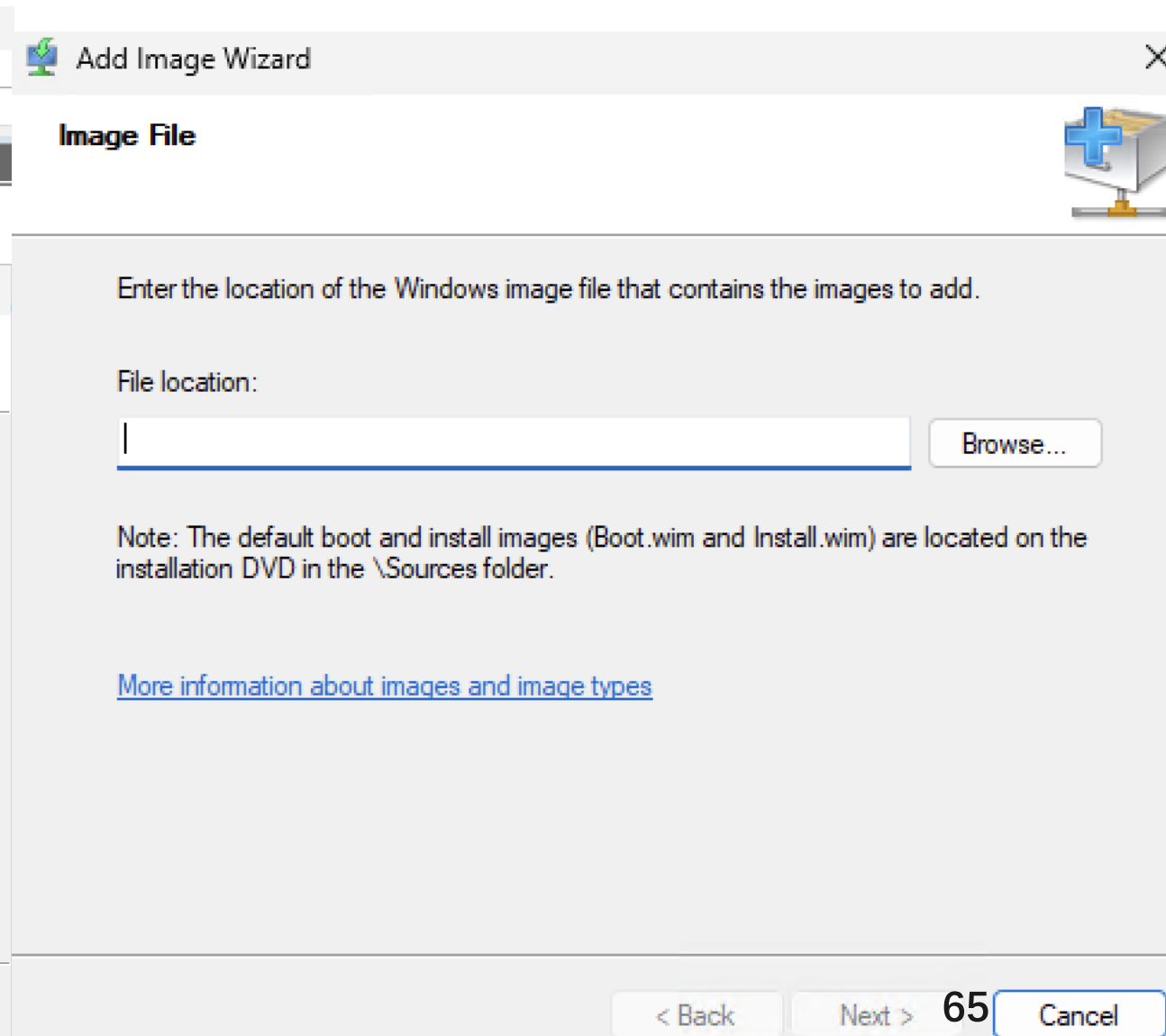
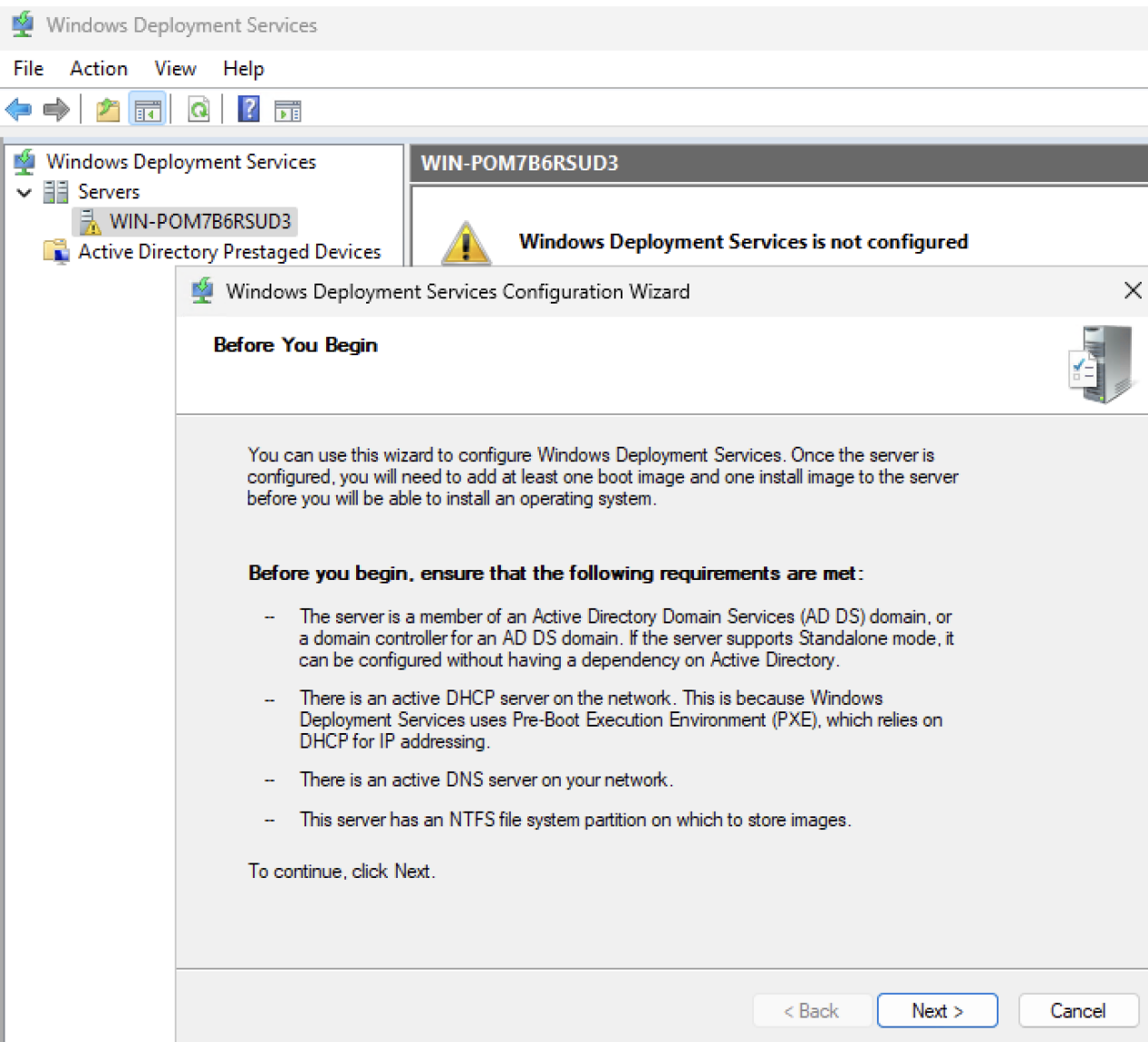




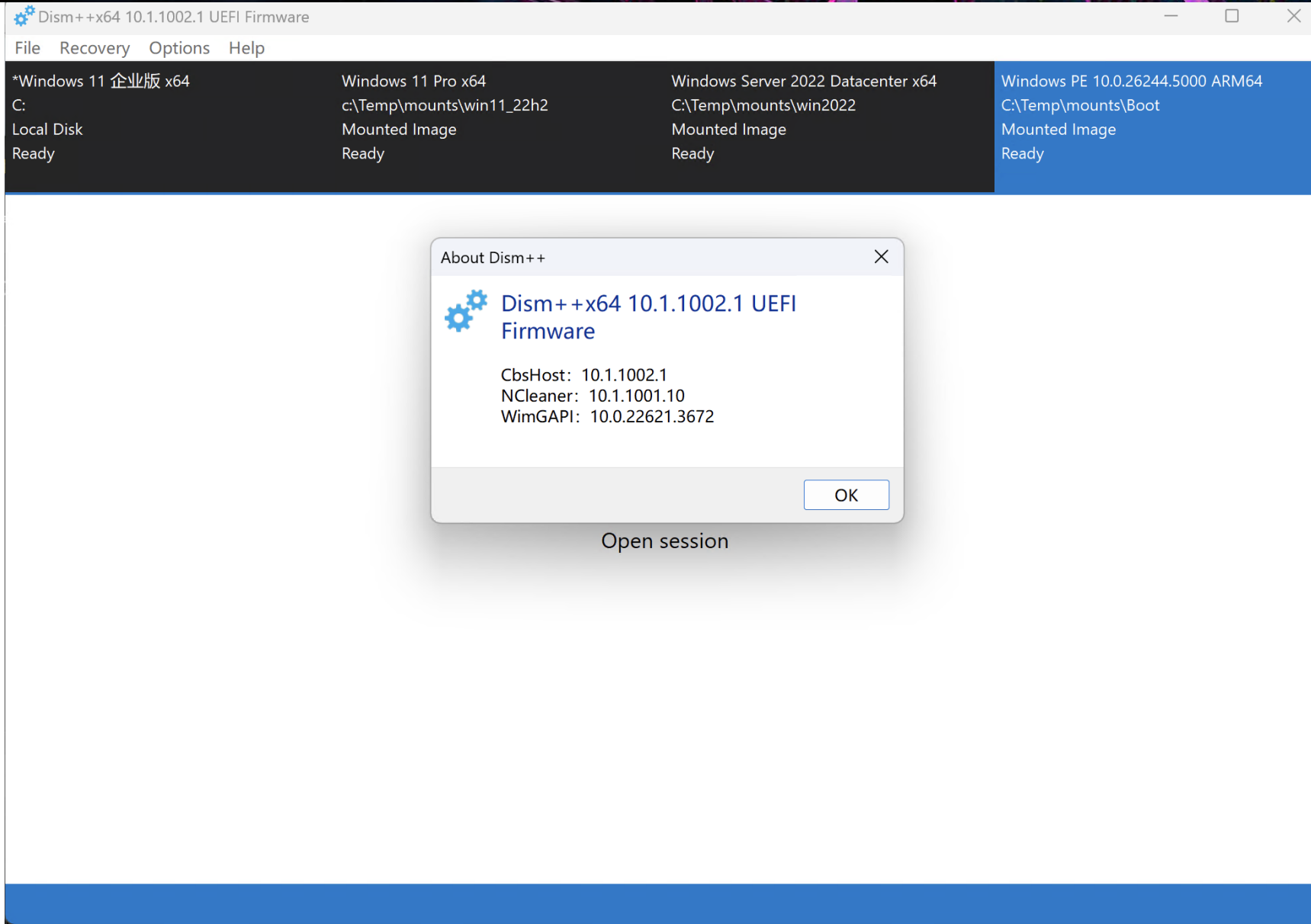
Your PC is under risk



How to make this attack



Introduce a handful tool



Introduce a handful tool

Mount Image-Max



Name	Value	
Image Name	Microsoft Windows Setup (arm64)	
Image Descri...	Microsoft Windows Setup (arm64)	
Edition	2	
Architecture	ARM64	
Created	2024/6/22 16:05:24	
Expanded Sp...	1.51 GB	
OS Version	10.0.26244.5000	

Target Image:

2: Microsoft Windows Setup (arm64)(Bootable)



C:\Temp\images\boot.wim



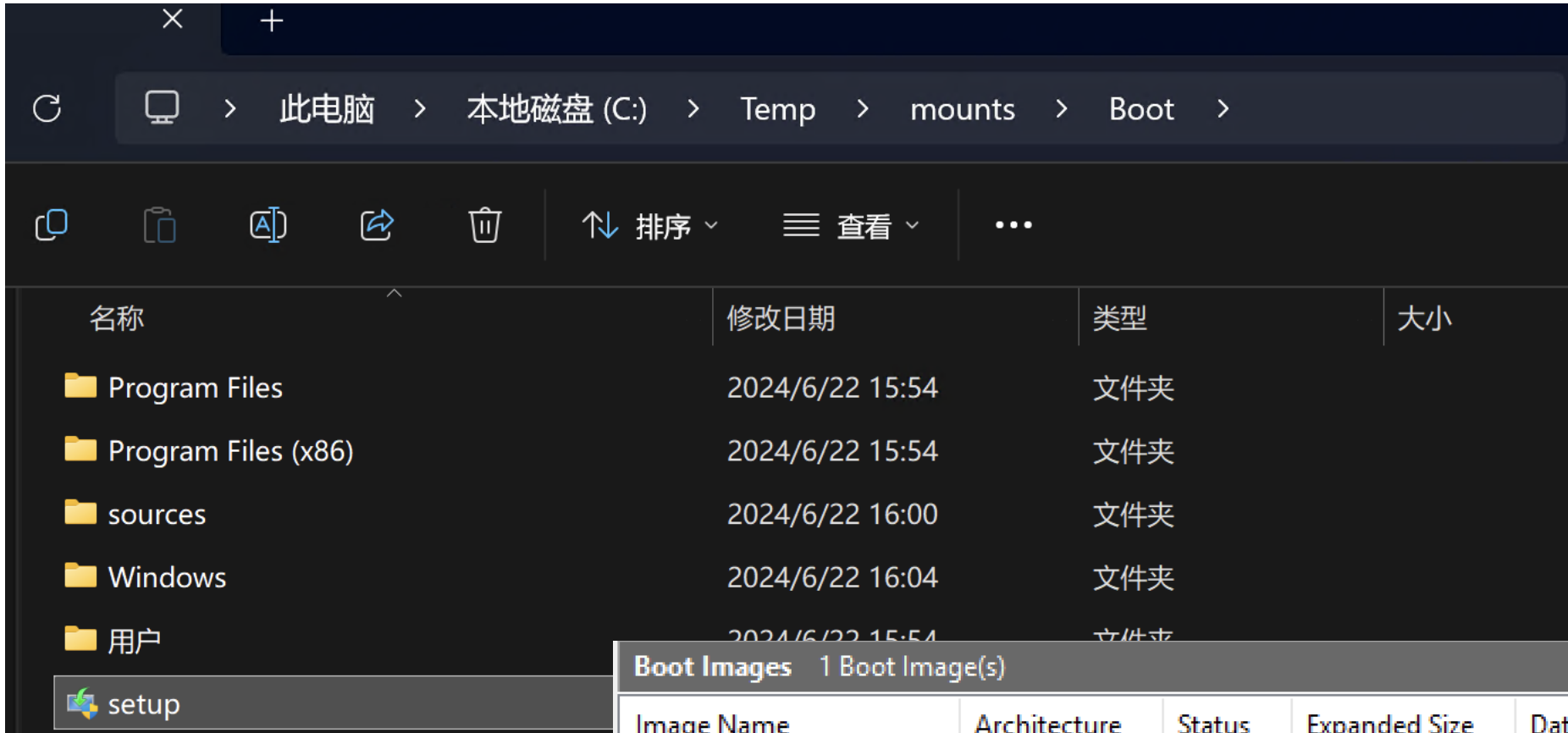
Browse

C:\Temp\mounts\Boot



Browse

Replace the setup.exe



Boot Images 1 Boot Image(s)							
Image Name	Architecture	Status	Expanded Size	Date	OS Version	Priority	
 Microsoft Windows S...	x64	Online	2295 MB	9/11/...	10.0.22621	500000 ...	

Just wait for fish to bite



If you don't want to wait

- You might exploit a remote DoS to force the victim to reboot

Physical Attack
SecureBoot



Local Attack
SecureBoot



Remote Attack
SecureBoot

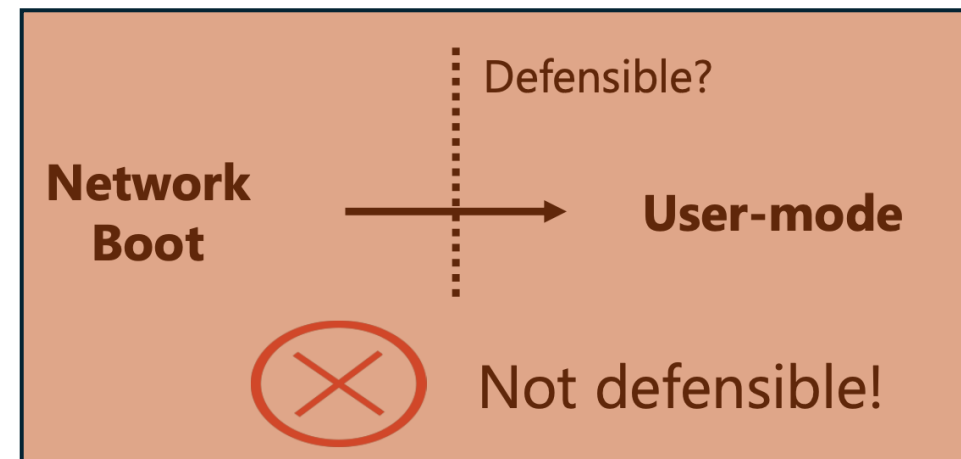
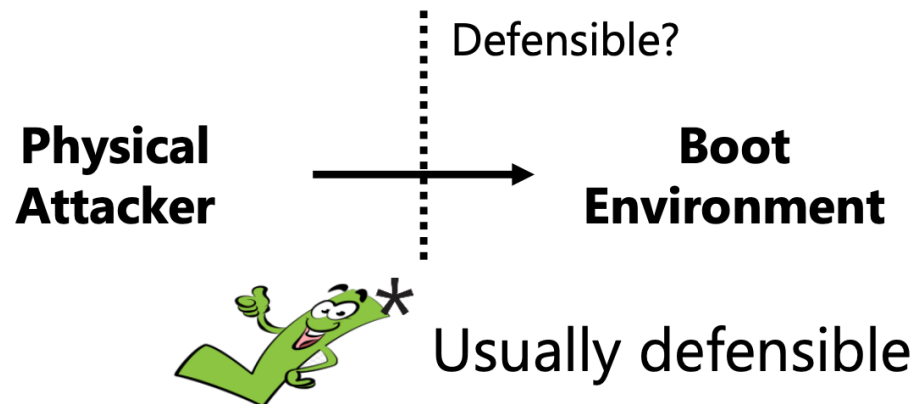
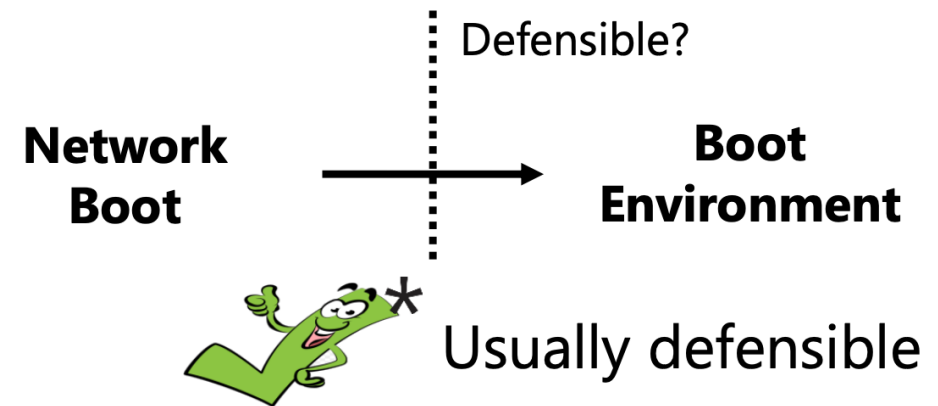
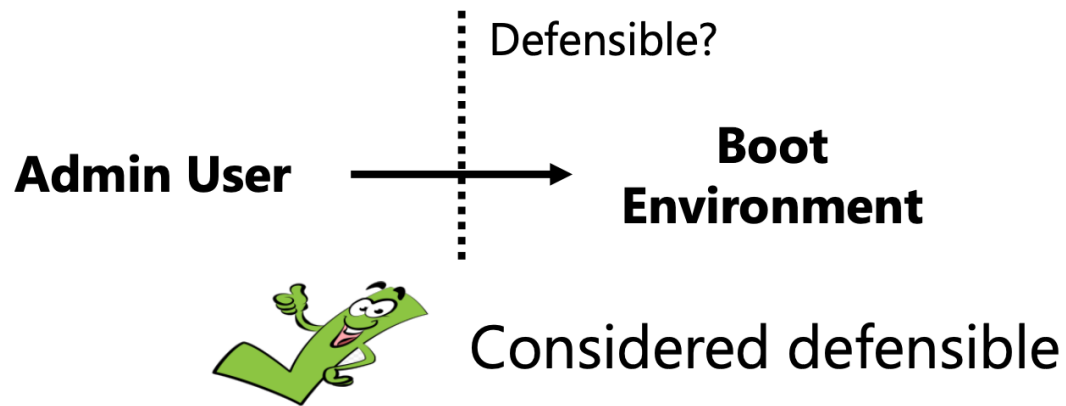


Remote DoS
Based Attack
SecureBoot



Where Does Microsoft Draw the Line?

Can an attacker achieve the same outcome by design?



- \WINDOWS\inf\errata.inf

```

63  if ( OslGetErrataFileNameFromRegistry(a4, &DestinationString) )
64  {
65      a2->Buffer[v7] = 0;
66      a2->Length = Length;
67      if ( B1AppendUnicodeToString(a2, L"inf\\") && B1AppendUnicodeToString(a2, DestinationString.Buffer) )
68      {
69          if ( OslIsTcbLaunchEnabled() )
70              v13 = B1ldrPreloadFile(a3, a2->Buffer);
71          else
72              v13 = B1ImgLoadImageWithProgress2(
73                  a3,
74                  0xE0000014LL,
75                  a2->Buffer,
76                  &a1->BasicData.Extension->EmInfFileImage,
77                  &a1->BasicData.Extension->EmInfFileSize,
78                  0x20000,
79                  0,
80                  0,
81                  0,
82                  0LL,
83                  0LL);

```

0000A88F OslpLoadMiscModules:63 (18000B78F)

NT Kernel

```

00000000 struct _LOADER_PARAMETER_EXTENSION // sizeof
00000000 {
00000000     unsigned int Size;
00000004     _PROFILE_PARAMETER_BLOCK Profile;
00000014     // padding byte
00000015     // padding byte
00000016     // padding byte
00000017     // padding byte
00000018     void *EmInfFileImage;
00000020     unsigned int EmInfFileSize;

```

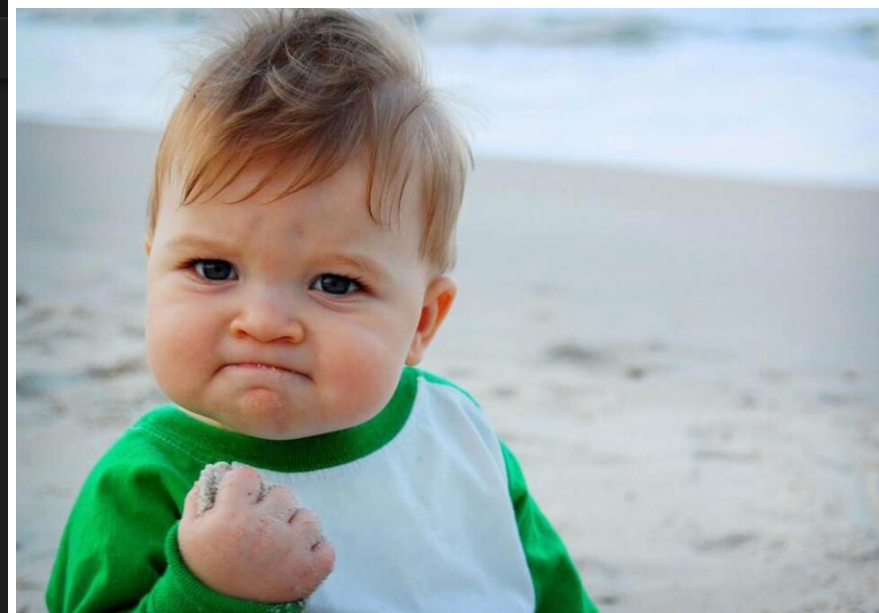
What's the file looks like

```
errata.inf x
errata.inf
1  ;/*++
2  ;
3  ;Copyright (c) Microsoft Corporation. All rights reserved.
4  ;
5  ;Module Name:
6  ;
7  ;   ERRATA.INF
8  ;
9  ;Abstract:
10 ;   INF file for the Errata Manager Database
11 ;
12 ;--*/
13 ;
14
15 ;=====
16 ;===== Declare the Target Rules =====
17 ;=====
18 ;
19 ;Specify the rules that the clients can register for notifications
20 ;Also need to specify the necessary string parameters if required
21 ;
22 ;N.B. The rule names must have been defined in the [RuleNameGuidDef] Section
23 ;   Declared in [RuleDef] Section and implemented in [Rule] Section
24 ;=====
25 [TargetRuleDef]
26
27 ACPISLPWorkAround = {FACP.ACER_OEMID.FACP.M25D_TableId}, \      ;ACERM25D02/25/00
28                     {FACP.COMPAQ_OEMID.FACP.LAREDO_TableId}, \    ;COMPAQLAREDO07/05/99
29                     {FACP.DELL_OEMID.FACP.WS210_TABLEID}, \      ;DellPrecisionWS210
30                     {FACP.DELL_OEMID.FACP.WS410_TABLEID}, \      ;DellPrecisionWS410
31                     {FACP.DELL_OEMID.FACP.WS610_TABLEID}, \      ;DellPrecisionWS610
32                     {FACP.DELL_OEMID.FACP.PE1300_TABLEID}, \      ;DellPowerEdge1300
```

- Attacker controlled
- Standard INF file
- Parsing in Kernel
- No common API



```
hook.cpp M  kdvm.cpp 4 X
VirtualKD-Redux > Lib > kdvmguestlib > kdvm.cpp > ChannelHelper<DefaultRPCChannel>
270
271     static NTSTATUS KdDebuggerInitialize0(PVOID lpLoaderParameterBlock)
272     {
273     #ifdef VKD_EXPERIMENTAL_PACKET_POLL_DIVIDER_SUPPORT
274     |         s_PacketPollRequestsToSkip = 0;
275     #endif
276
277     |         //PVOID hMod = KernelGetModuleBase("ntoskrnl.exe");
278     |         PVOID hMod = GetModuleBaseAddress(&IofCallDriver);
279     |         setup((size_t)hMod);
280     |         return STATUS_INVALID_PARAMETER;
281     }
```



Start fuzzing harness code from opensource code base

Choose Advanced Options for: Disable Signature Enforcement Manually!!! (Press F8) [VKD-Redux]

(Use the arrow keys to highlight your choice.)

• St

```
$ ../packer/qemu_tool.sh create_snapshot vnext.img 2048 ./ntos_fuzz
```

```
CREATE_SNAPSHOT
```

```
qemu-system-x86_64: warning: host doesn't support requested feature: CPUID.07H:EBX.hle [bit 4]
```

```
qemu-system-x86_64: warning: host doesn't support requested feature: CPUID.07H:EBX.rtm [bit 11]
```

```
[!] qemu-nyx: preparing to create pre image...
```

```
Creating pre image snapshot[qemu-nyx] switching to secondary CoW buffer
```

Enable Boot Logging

Enable low-resolution video

Last Known Good Configuration (advanced)

Debugging Mode

Disable automatic restart on system failure

Disable Driver Signature Enforcement

Disable Early Launch Anti-Malware Driver

Start Windows Normally

Description: Allows drivers containing improper signatures to be loaded.

ENTER=Choose

ESC=Cancel

Install VirtualKD-Red

This program will prepare
be updated accordingly.

☒ Create a new boot en

Disable Signature Enfo

☐ Use existing entry "W

☒ Set VirtualKD-Redux

☒ Replace kdcom.dll (r

☐ Patch winload to forc

☐ Install VisualDDK Lat

Install

id_000000,sig_00,src_000419,time_306686,execs_34066,op_havoc,rep_2

```

1 |1: kd> .trap 0xffffffff8864ca06f50
2 NOTE: The trap frame does not contain all registers.
3 Some register values may be zeroed or incorrect.
4 rax=0000000000000000 rbx=0000000000000000 rcx=fffffffffffffffff8
5 rdx=ffff9c0e1f960600 rsi=0000000000000000 rdi=0000000000000000
6 rip=fffff80235986c69 rsp=fffff8864ca070e0 rbp=00000000000000101
7 r8=ffff9c0e1f99a150 r9=0000000000000002 r10=00000000000000543
8 r11=ffff9c0e1f9a9fa0 r12=0000000000000000 r13=0000000000000000
9 r14=0000000000000000 r15=0000000000000000
10 iopl=0         nv up ei pl nz na pe nc
11 nt!EmpParseRules+0x275:
12 fffff802`35986c69 483919          cmp     qword ptr [rcx],rbx ds:ffffffffff`fffff8=???????????????
13 1: kd> kf
14 *** Stack trace for last set context - .thread/.cxr resets it
15 #   Memory   Child-SP          RetAddr          Call Site
16 00           fffff886`4ca070e0 fffff802`35985f77 nt!EmpParseRules+0x275
17 01          70 fffff886`4ca07150 fffff802`359cec43 nt!EmpParseInfDatabase+0x97
18 02          40 fffff886`4ca07190 fffff802`3597ce6b nt!EmInitSystem+0x12b
19 03         240 fffff886`4ca073d0 fffff802`354992a3 nt!Phase1InitializationDiscard+0xe63
20 04         1a0 fffff886`4ca07570 fffff802`35263a2a nt!Phase1Initialization+0x23
21 05          40 fffff886`4ca075b0 fffff802`3546e2d4 nt!PspSystemThreadStartup+0x5a
22 06          50 fffff886`4ca07600 00000000`00000000 nt!KiStartSystemThread+0x34

```

Do RE job, find RCE

```

44 Pool2 = (GUID *)ExAllocatePool2(0x100ui64, (int)(8 * SectionLineIndexValueCount + 0x38), 'time');// oob
45 oob_object_size_10 = (__int64)Pool2;
46 if ( !Pool2 )
47     return 0xC000009A;
48 GuidFromName = EmpInfParseGetGuidFromName((__int64)a1, (__int64)"CallbackGuidDef", KeyName, Pool2);
49 if ( GuidFromName < 0 )
50     break;
51 if ( (__int64)EmpSearchCallbackDatabase((__QWORD *)oob_object_size_10) )
52     goto LABEL_13;
53 *(_DWORD *)(oob_object_size_10 + 0x40) = v7 - 2;
54 SectionLineIndex = (const char *)CmpGetSectionLineIndex(a1, "CallbackDef", v2, 0);
55 if ( !SectionLineIndex )
56     break;
57 *(_DWORD *)(oob_object_size_10 + 0x38) = strtoul(SectionLineIndex, 0i64, 10);
58 v12 = (const char *)CmpGetSectionLineIndex(a1, "CallbackDef", v2, 1u);
59 if ( !v12 )
60     break;
61 *(_DWORD *)(oob_object_size_10 + 0x3C) = strtoul(v12, 0i64, 10);
62 for ( i = 2; i < v7; ++i )
63 {
64     v14 = (
65     GuidFrom
66     if ( Gu
67     {
68         ++v2;
69         ExFre
70         goto
71     }
72     v16 = i
73     *(_QWORD
74     }
75     *(_QWORD
76     *(_QWORD
77     ++v2;
78     *(_DWORD
79     *(_QWORD
80     ++EmpNumb
81     *(_QWORD *) (oob_object_size_10 + 0x28) = EmpCallbackListHead;
82     EmpCallbackListHead = oob_object_size_10 + 0x28;
83 }
84 v10 = (GUID *)oob_object_size_10;
85 LABEL_13:
86 ExFreePoolWithTag(v10, 'time');
87 goto LABEL_5;
88 }

```

Windows Kernel EmpParseCallbacks Heap Out-of-Bounds Write

The first Windows kernel memory corruption I've discovered in my career.

```
ExAllocatePool2(0x100ui64, (int)(8 * SectionLineIndexValueCount + 0x38),
```

```
*(_DWORD *) (oob_object_size_10 + 0x38) = strtoul(SectionLineIndex,
```

```
v12 = (const char *)CmpGetSectionLineIndex(a1, "CallbackDef", v2,
```

```
if ( !v12 )
```

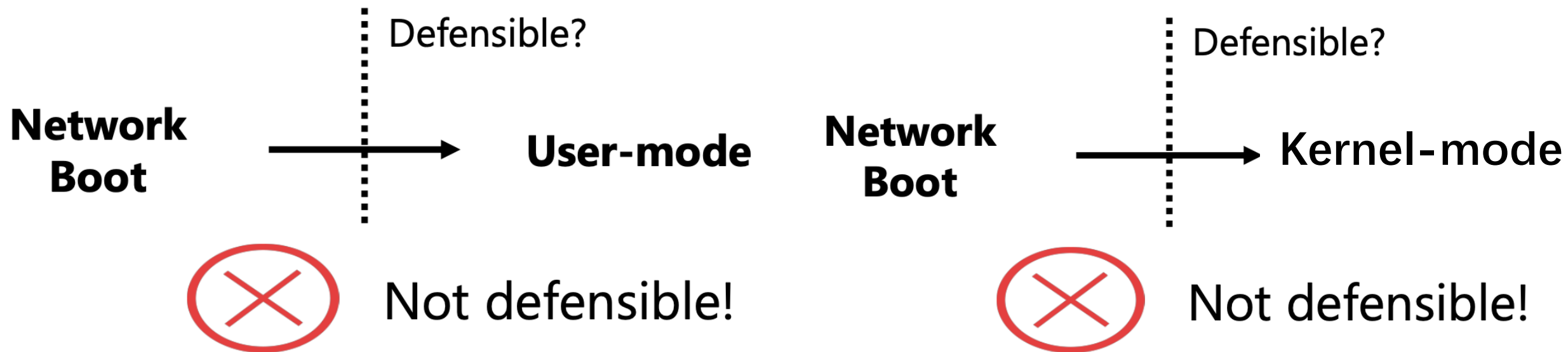
```
break;
```

```
*(_DWORD *) (oob_object_size_10 + 0x3C) = strtoul(v12, 0i64, 10);
```

```
7 f.write(b", ".join([b"0"]*0xffff))
```

```
8 f.write(a[111939+45:])
```

What's Actually Going On?



We have determined that the behavior described in your report is by design.

This case has been determined to be a moderate severity defense in depth issue and will not require a security update.



Status changed from **Review / Repro** to **Complete**

- Background
- Attack surface in bootloader
 - Network protocol
 - BCD Registry
 - Security Policy
 - Filesystem
 - Logic flaw
- Attack surface beyond bootloader
- **Future Work & Take Aways**

- Continue research on bootmgfw.efi on other attack surface
- Winload.efi
 - Hardware specific firmware (etc. HSP on AMD platform)
- Resume.efi
- Hyper-V bootloader
- Research on Windows kernel and userland service code that is invisible to remote attacker from normal boot

- Looking at Microsoft's patch, there's multiple branch, PCA2011 and PCA2023. Code before 26100 and code after 26100.

BCD Element Processing	24-Apr_PCA2023	CVE-2024-28923	BiConvertLocateDeviceElement LocateDevice invalid LocateIdentifier Size Integer overflow Heap OOB write
BCD Element Processing	24-Jul_PCA2023	CVE-2024-37973	BcdGetElementDataWithFlags Recursive calling stack OOB
BCD Element Processing	24-Apr_PCA2023	CVE-2024-28897	BiConvertLocateDeviceElement LocateDevice invalid ParentOffset uninitialized memory Heap OOB write
BCD Element Processing	24-Apr_PCA2023	CVE-2024-28898	BiEnumerateElements Recursive calling stack OOB
Architecture issue	24-Apr_PCA2023	CVE-2024-29062	BmFwVerifySelfIntegrity SFB

- You really should take the update guide manually. It's not only for DBX update, also to switching your bootloader to PCA2023 branch.

Booting into breaches

- PCA2011 Time breaches
- Patch branch breaches



- B(ring) Y(our) O(wn) B(ootloader) to archive AV:A in secureboot attack
- Small function with sanitize in its name could be very vulnerable
- Recursive calling could be exploitable to RCE in UEFI environment
- Check twice after your patch release, especially when you have found vulnerabilities in same component at a very large volume, don't be lazy.
- Take closer look at the code if fuzzer can generate DoS.
- Out of scope vulnerabilities could also be interesting in real world.
- Take further action immediately to fix these SecureBoot vulnerabilities.

Boot Security

Unauthenticated Attacker from network



Recheck for multiple WDS server



UEFI PXE

Firmware signature verification



Bootloader

Windows signature verification



OS Loader

Security boundary

Attacker controlled WIM image



Remote code execution

Services

Windows Kernel

Userland Application

User Data

OS Security

Z1UFPJ0LM6mUMe1t4UX8U3XZ/hoa5sc+wLWxLLB2PK6zHeJtUamU+gZGhSHNXIyIyUten168tu2YPM1ahgKx8bnKHKHlr14Vr-bL5mLet0LoU/565U
Sywy6PQsT1swCdEF3WiPSyZhUE+n1a3QNQeY38W681C+Gda76C8iEGFMlyheaswddq1Pt3M20UstdDcto5WogDs1sK7u53zWldoVolbEEV5DCj0C8
wLh+726qXptp6GXdAIxuvDNo9Y1xLHAN06fvEh+qaPtiwRPWZRc08LmhERAC9esLChIRpP9AVLArmnIyf9LhQJiFCbv5ukP/jTvNdrYSD9sgbLqRK
eXnTlzm5D9Vx/7mN8W+iTJft7d9flsJSy8qfyf2IjjXHjmeUusBcmdNXqyQbskxvzLPCwgs+2LMZPPnQ5jtpM8VvyF5fuCkJwNdZNFJC465hZ0f36cs
MX1NH4sZ+TP47eWYDoncV80HgPF1fkujTWqwTI38wen1k0v3EPyx0RptOXyjlGdLvUpmKCD/1Z6tWYdIsybmsnGzpWsB8k3KdUZvx1fCJua++9QRD
i5GcUeRLr7htQMOLtoRn4oAeRZOf0993YkN1fe/3CFcbrOfmeaorApRrCFauTLlQFF5yd+X32591vLrC919uHQki51/dHTOyPmNgfMbe/YmJ/xMT/9
gJoP0m179M0k9V0ZxBZMwPhSegd3bK1h4fQP9LQffPBS4ubq6/fACfo761oDh1B38ecwNaOL0a+d/4Vmom5VuYt36EYq4K0Qp5j4P/H/d666/u9eD
Ou/8X8AVLIyWQdwAAA=

This project is part of the Black Hat USA research "Booting into Breaches: Hunting Windows SecureBoot's Remote Attack Surfaces". It helps you check if your system is affected by the 31 SecureBoot vulnerabilities discovered by Azure Yang and patched in 2024. The tool collects anonymous data for presentation in the final Black Hat talk.

VIEW BLACK HAT BREACHING

SHOW AFFECTED VULNERABILITIES

A tool to detect secureboot status

SecureBoot, designed to protect against firmware-level tampering, has long been dismissed as a "local-only" attack surface. This research shatters that assumption, exposing systemic flaws that enable remote exploitation of SecureBoot—critical in Pre-Auth RCE on fully patched systems. With 31 CVEs discovered and fixed in Microsoft's SecureBoot implementation, we reveal how attackers can weaponize bootloader components (network stacks, BCD registries, filesystems) to bypass critical security guarantees.

Select Your Operating System:

WINDOWS

LINUX

Windows Instructions

Please run the following PowerShell commands as Administrator:

PowerShell

Copied!

```
<#
.SYNOPSIS
    Efficiently compresses and Base64 encodes Secure Boot UEFI database (db/dbx) in binary format.
.DESCRIPTION
```

ANALYZE

Analysis Results

✓ SecureBoot is enabled
Your system has SecureBoot enabled, which helps protect against boot-level malware.

✓ Windows PCA 2023 Certificate detected
Your Boot certificate database is up to date.

✓ Not vulnerable to SecureBoot Breaches
Your PC is not affected by the 31 vulnerabilities disclosed in this research.

✓ Your PC SecureBoot is fully functional and is up to date



Help us improve by taking a quick survey about your system

Thanks!
X: @4zure9

