

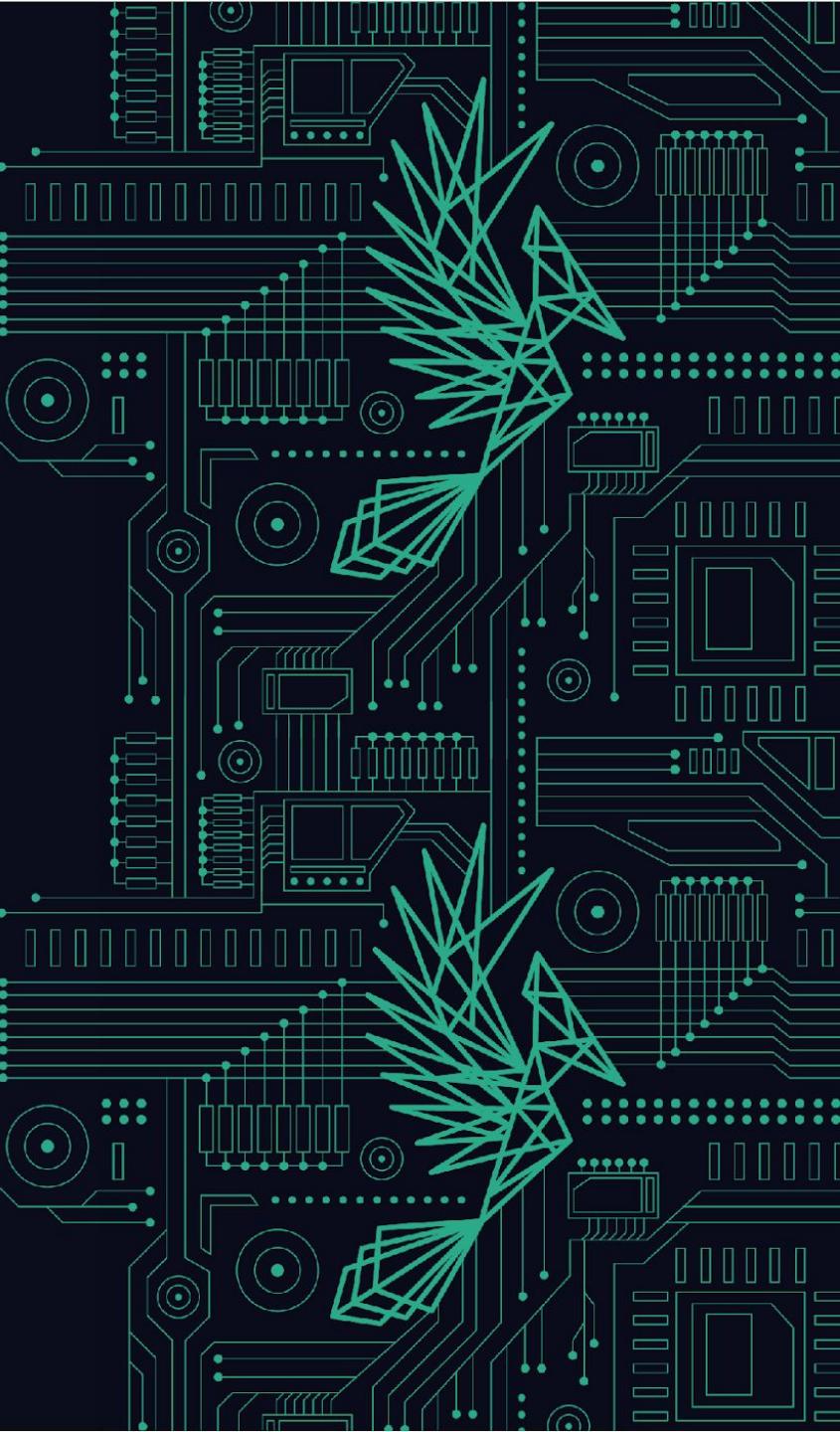
LABS CON

Signed and Dangerous: BYOVD Attacks on Secure Boot

Alex Matrosov, Fabio Pagani



binarly



Presenters



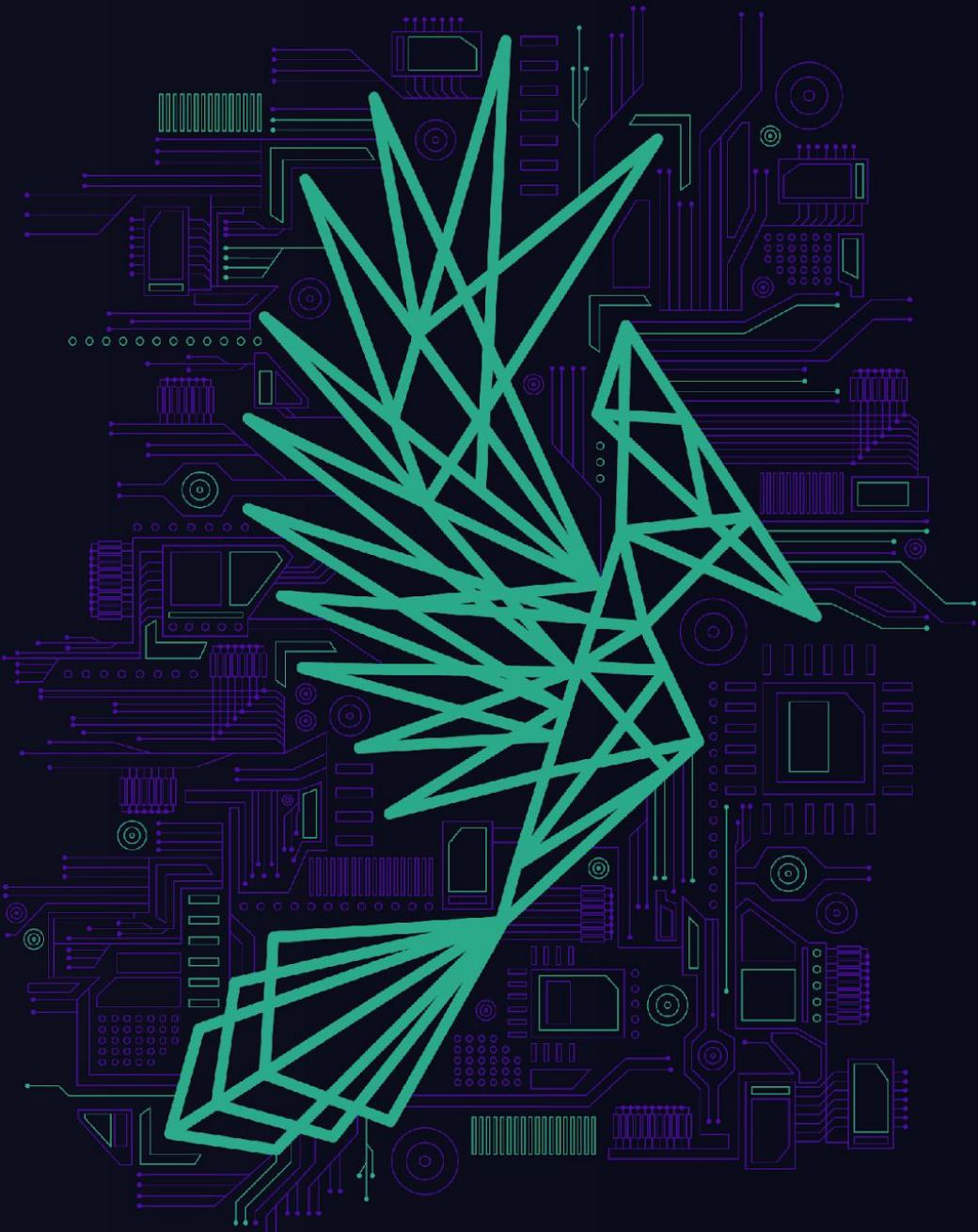
Alex Matrosov
CEO & Head
of Research

Fabio Pagani
Vulnerability
Research Lead

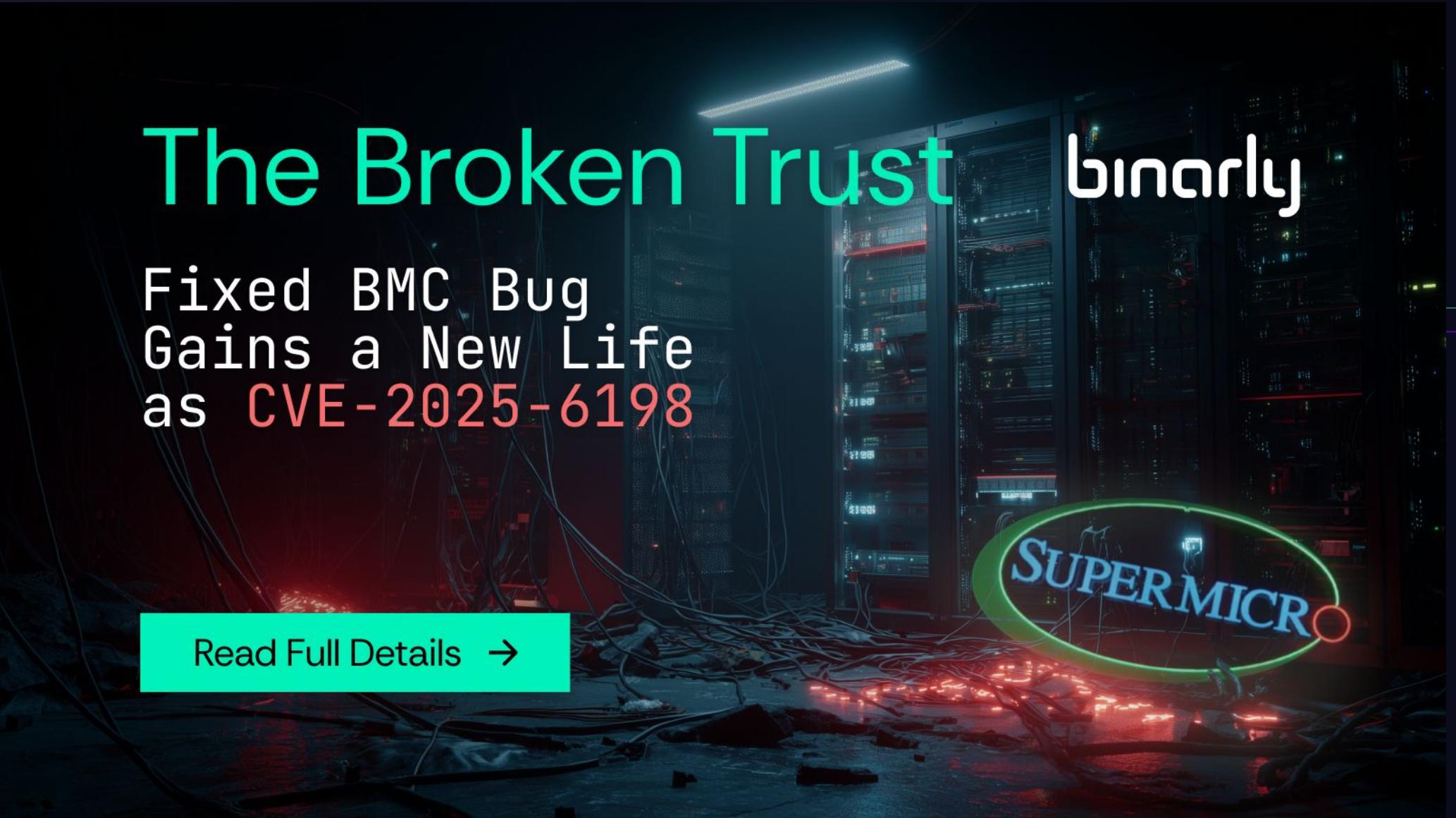


Agenda

- BYOVD Attacks (UEFI version)
- Taxonomy of Attacks against Secure Boot
- Finding Secure Boot bypasses
- Breaking BMC firmware validation
- Conclusions



The Broken Trust



binarly

Fixed BMC Bug
Gains a New Life
as **CVE-2025-6198**

Read Full Details →

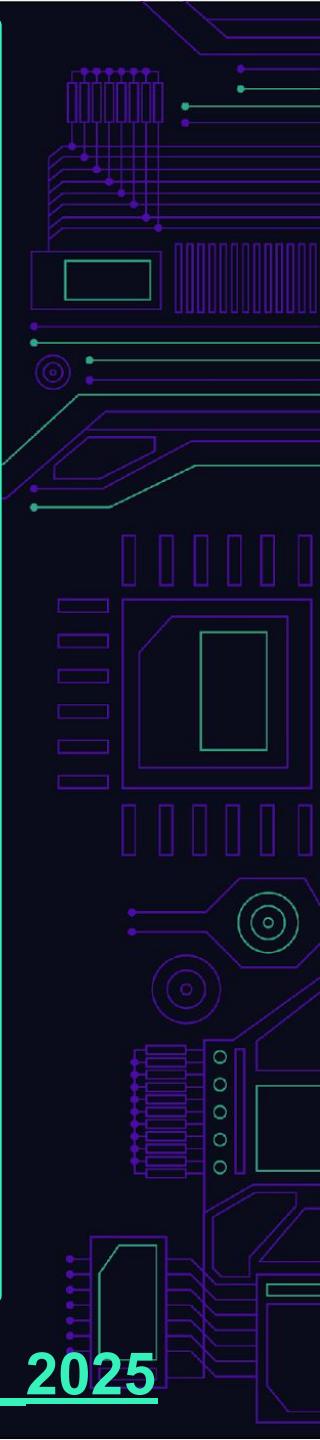
<https://www.binarly.io/blog/broken-trust-fixed-supermicro-bmc-bug-gains-a-new-life-in-two-new-vulnerabilities>

<https://www.binarly.io/advisories/brly-2025-020>

<https://www.binarly.io/advisories/brly-2025-021>

Two security issues have been discovered in select Supermicro boards. These issues may affect Supermicro BMC Firmware.

CVE ID	Severity	Issue Type	Description
<u>CVE-2025-7937</u>	Medium	Improper Verification of Cryptographic Signature	A crafted firmware image can bypass the Supermicro BMC firmware verification logic of RoT 1.0 to update the system firmware. The crafted image has a customized PDBA table of RoT 1.0 to redirect the program to the fake PDBA table in the unsigned region. Supermicro CVSSv3 score: 6.6 (AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H)
<u>CVE-2025-6198</u>	Medium	Improper Verification of Cryptographic Signature	A crafted firmware image can bypass the Supermicro BMC firmware verification logic of Signing Table to update the system firmware. The crafted image has a customized signing table to redirect the program to the fake signing table in the unsigned region. Supermicro CVSSv3 score: 6.4 (AV:N/AC:H/PR:H/UI:R/S:U/C:H/I:H/A:H)



Security risks arising from firmware developer and device vendor breaches

2022

- Intel PPAM expired certificate
- LC/FC data breach

2024

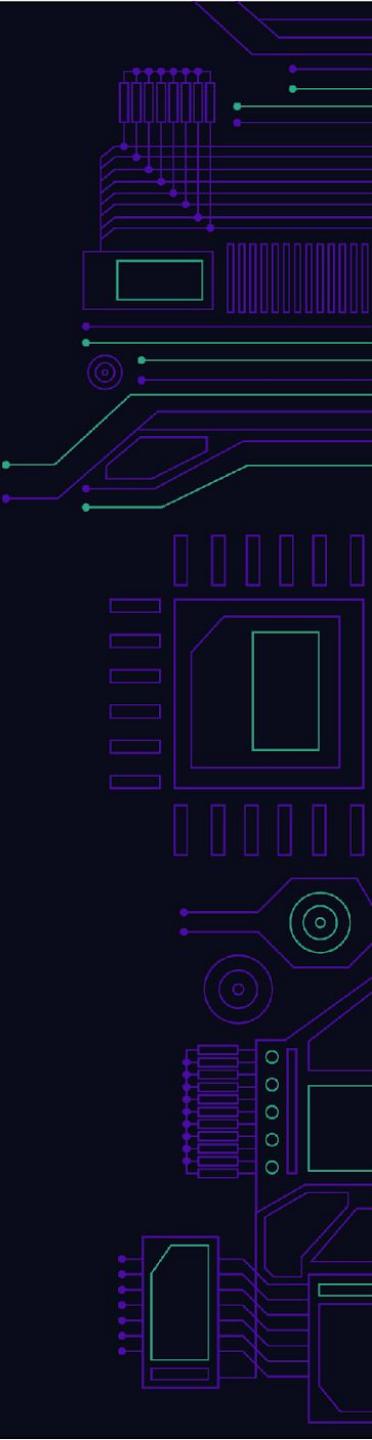
- PKfail + BlackLotus Demo
- Supermicro

2023

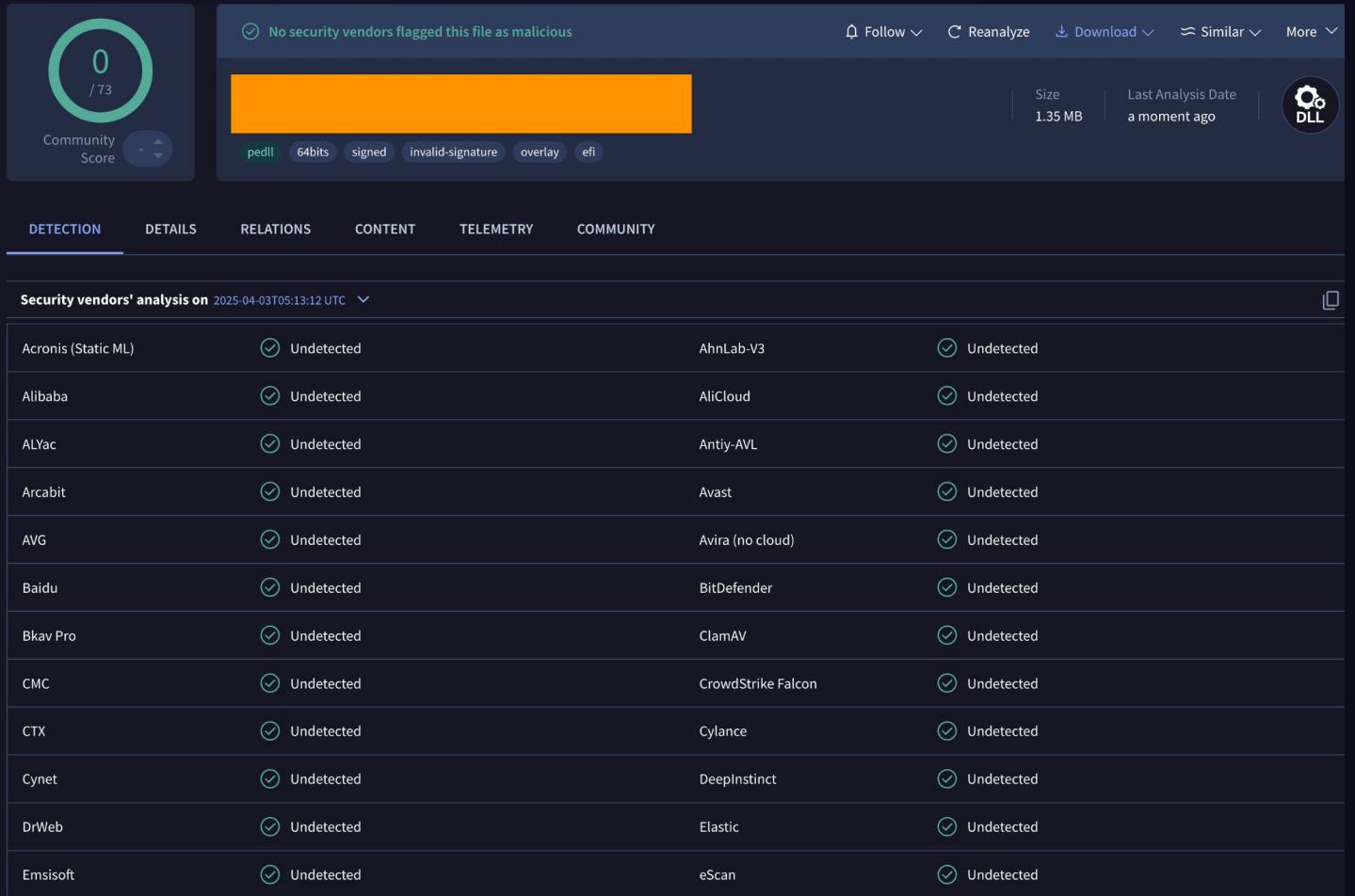
- MSI OEM data breach
- Intel BootGuard key leakage impact

2025

- DBX inconsistency
- Intel BootGuard again?
- AMD Microcode validation is broken



[2025] SignedModule.efi found on VT



No security vendors flagged this file as malicious

Follow Reanalyze Download Similar More

Community Score: 0 / 73

File details: pedll 64bits signed invalid-signature overlay efi

Size: 1.35 MB | Last Analysis Date: a moment ago | DLL

DETECTION DETAILS RELATIONS CONTENT TELEMETRY COMMUNITY

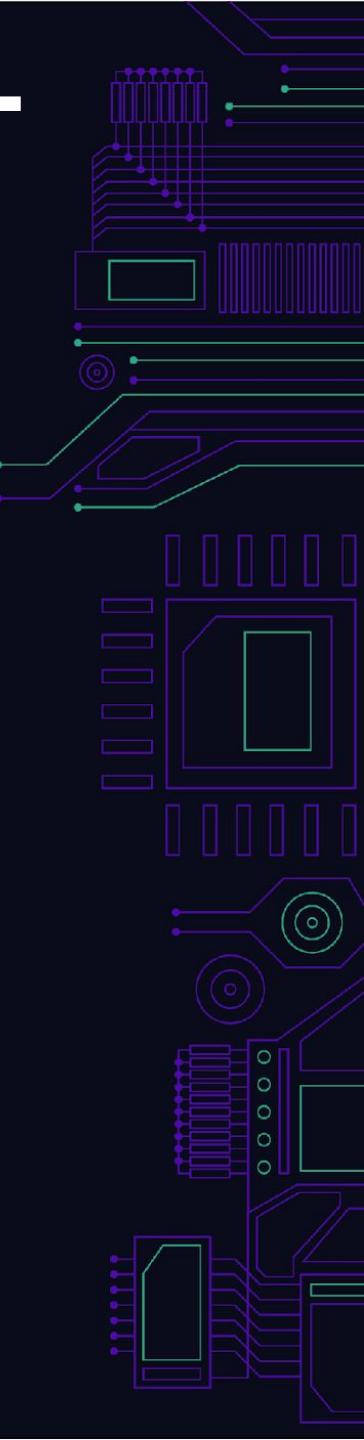
Security vendors' analysis on 2025-04-03T05:13:12 UTC

Vendor	Result	Vendor	Result
Acronis (Static ML)	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	AliCloud	Undetected
ALYac	Undetected	Antiy-AVL	Undetected
Arcabit	Undetected	Avast	Undetected
AVG	Undetected	Avira (no cloud)	Undetected
Baidu	Undetected	BitDefender	Undetected
Bkav Pro	Undetected	ClamAV	Undetected
CMC	Undetected	CrowdStrike Falcon	Undetected
CTX	Undetected	Cylance	Undetected
Cynet	Undetected	DeepInstinct	Undetected
DrWeb	Undetected	Elastic	Undetected
Emsisoft	Undetected	eScan	Undetected

<https://www.binarly.io/blog/another-crack-in-the-chain-of-trust>
<https://www.kb.cert.org/vuls/id/806555>

[2025] SignedModule.efi found on VT

```
$ authenticode-tool info Dtbios-efi64-71.22.efi
SHA-1: 7ec65bb912b1fdce514a1a5ff8cf2ed187eb8fa3
SHA-256: 6b4328ebcbe46ed9118ff2d4472de329d70ba83016df7a6f50f8af923883bc54
Signature 0:
Digest: 6b4328ebcbe46ed9118ff2d4472de329d70ba83016df7a6f50f8af923883bc54
Signer:
  Issuer: CN=Microsoft Corporation UEFI CA 2011,
            O=Microsoft Corporation,L=Redmond,ST=Washington,C=US
  Serial number: 33:00:00:00:4F:53:61:25:A6:D6:64:88:67:00:01:00:00:00:4F
Certificate 0:
  Issuer: CN=Microsoft Corporation UEFI CA 2011,
            O=Microsoft Corporation,L=Redmond,ST=Washington,C=US
  Subject: CN=Microsoft Windows UEFI Driver Publisher,
            O=Microsoft Corporation,L=Redmond,ST=Washington,C=US
  Serial number: 33:00:00:00:4F:53:61:25:A6:D6:64:88:67:00:01:00:00:00:4F
Certificate 1:
  Issuer: CN=Microsoft Corporation Third Party Marketplace Root,
            O=Microsoft Corporation,L=Redmond,ST=Washington,C=US
  Subject: CN=Microsoft Corporation UEFI CA 2011,
            O=Microsoft Corporation,L=Redmond,ST=Washington,C=US
  Serial number: 61:08:D3:C4:00:00:00:00:00:04
```



Introduction to BYOVD

- Technique that exploits vulnerabilities in legitimate Windows kernel drivers to gain privileged access
- The drivers are **signed and trusted** by the OS:
 - Attacker installs the vulnerable kernel driver
 - The vulnerability is exploited in kernel context
 - Profit (?)
- Historically used only by Advanced Persistent Threats (APTs),
BYOVD is now found in commodity threats too (ransomware)

<https://blog.talosintelligence.com/exploring-vulnerable-windows-drivers/>

BYOVD + UEFI = ?

- UEFI firmware also relies on signature verification when Secure Boot is active
- Secure Boot: only **trusted and verified** modules are allowed to be executed
- Determination based on the content of NVRAM variables:
 - **db** → allowed signatures
 - **dbx** → revoked signatures

What is the impact
of BYOVD on UEFI?



Taxonomy of Attacks Against Secure Boot

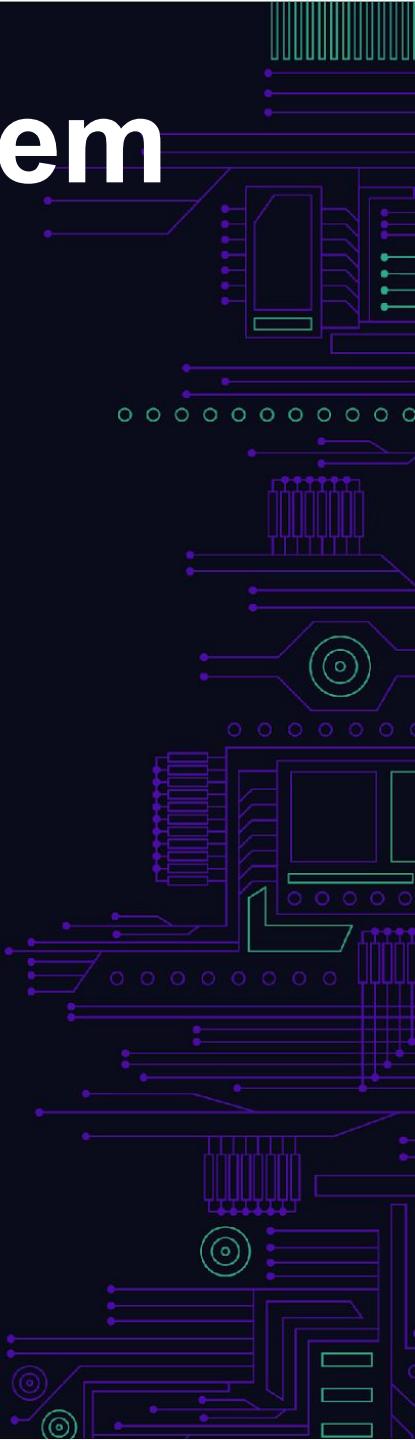
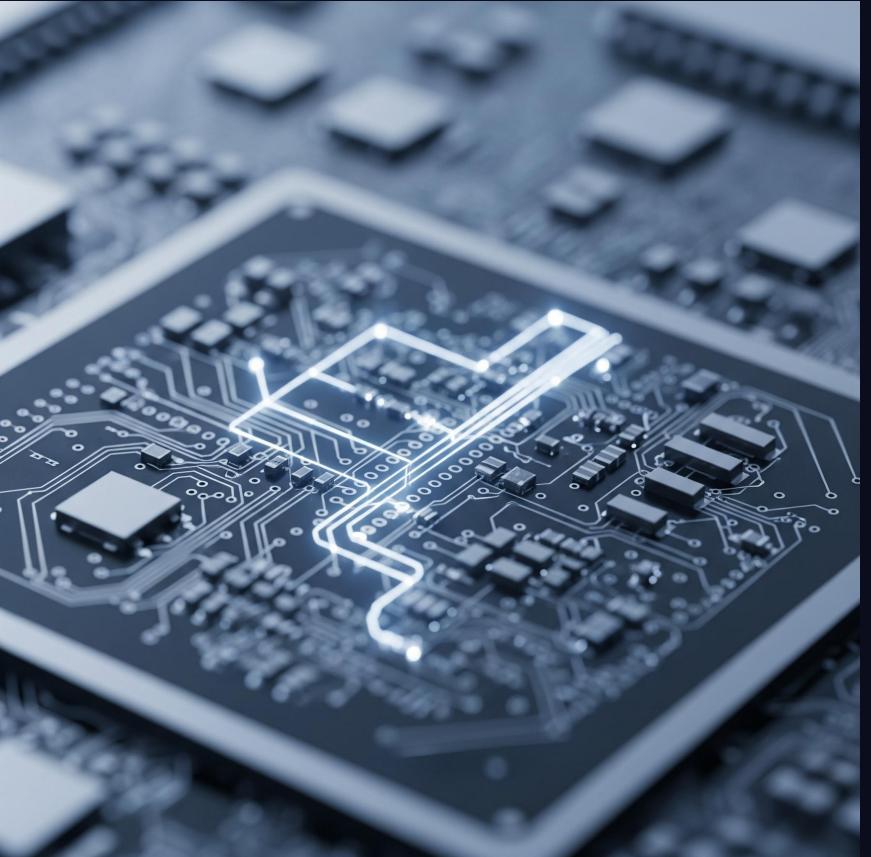
1. **Double-use modules**: Trusted programs exposing a functionality that can be misused to run untrusted code (e.g. the UEFI Shell)
2. **Trusted but vulnerable modules**: Trusted programs that contain exploitable vulnerabilities (e.g. CVE-2025-3052)
3. Leaked private keys: Keys used in authentication that are compromised, allowing attackers to sign malicious modules (e.g. PKfail)
4. **Verification logic bugs**: Bugs in the verification process itself that allows an attacker to bypass verification (e.g. CVE-2025-6198)
5. Debug or incomplete features: Features intended for debugging end up in production devices and allow to bypass authentication (e.g. CVE-2021-0114)

<https://www.binarly.io/blog/signed-and-dangerous-byovd-attacks-on-secure-boot>

Identify BYOVD in the UEFI ecosystem

High-level plan to identify **double-use** and **trusted but vulnerable** modules:

1. Collect a comprehensive dataset of UEFI modules
2. Determine which modules are trusted by real-world firmware
3. Scan trusted modules to detect double-use and trusted but vulnerable modules



Large database of UEFI modules

- Sources:
 - Internal collection of UEFI firmware (gathered over 5+ years)
 - Private telemetry data ([pk.fail](#) detector)
 - Public threat intelligence feeds (VirusTotal)
- Indexed over **10 million modules**

	name	guid	hex(hash)	authenticode	length(cert)
1	RealtekUndiDriver	e88db748-a947-46cf-ab6f-5c99b6c6c4b8	E70AD86ED34F1E7948253B4AB7F18...	FC5C7711F42C178A03C2B5067DED60C96BD9...	8672
2	RealtekPxe	1be14579-d805-4c3b-8874-410b818674e9	A4782AD88B9AA789F2C6421FB0C90...	BFD73544D17BEAB0ABB26C28335D3141C403...	8640
3	InfineonTpmUpdateDxe	8900e28f-de99-4fc4-894b-6f41cd139a48	CE383755FB2B13984C6750791495A...	E39214F6C5F4E1C7653640B3D25DE9036837...	8632
4	A8DAFB9B-3529-4E87-8584-ECDB6A5B78B6	a8dafb9b-3529-4e87-8584-ecdb6a5b78b6	46244EE2B5FDC63A0DD05C021A6EA...	B9CE1967709E788BC85D709F9A324D7C54E5...	8552
5	RtkUsbUndiDxe	3ed432c9-5f9d-415d-a1c3-2b0427a90758	ACB9A6CDDC57B623AD939891C9C06...	E822EE1DB8F068696FD106295EADC7F5393...	8552
6	7C0B621C-118C-49F3-BA6A-003244829342	7c0b621c-118c-49f3-ba6a-003244829342	5CDF3D75C0EC0800B9692AEDEF195...	3789CA5B6CCD21A528374F0FB85958516966...	1424
7	RtkUndiDxe	b7b82ad8-3349-4968-a940-7b8c265ff9b4	1E8ABB2E42F4F9D041CCC71DB642A...	1ABC75968C86E2DA5F9EAE4187A689D3EE47...	8744
8	AEB1671D-019C-4B3B-BA00-35A2E6280436	aeb1671d-019c-4b3b-ba00-35a2e6280436	36D5DD7D857FF7A9CBCE64EEEAFB6...	B09EAAADCE7C95318364D4A0103EAB08DEFC...	20760
9	Rtk8111UndiBin	2851e234-20fd-4d1e-9041-dcb8f3025cae	6E2DD29F159EDF01187FB6B518DBA...	F27308D9AB25BEADD7413A19E7E5232B5DF2...	9624
10	EzFlashInterfaceBin	d1531968-e138-4e2e-8f7e-383307169276	C33B9914C7D8FB5767B733FE121C5...	0FAC038F39EC874CF1D5CB56E188806B21A2...	1408



Which UEFI Modules Are Trusted?

- Selected 4000 recent firmware images, covering most OEMs
- Identified which modules from the database are trusted by the selected firmware images
- Results:
 - Discovered 7157 unique modules trusted by recent firmware
 - On average, firmware trusts 1500 modules, with peaks over 4000 modules

A vulnerability in any trusted module can be used to bypass Secure Boot on the device



Trusted but Vulnerable Modules

- Scanned modules with our platform to uncover issues in NVRAM variable handling and beyond
- Automatically identified one vulnerability (CVE-2025-3052) in a module signed with the Microsoft's third-party UEFI certificate
- June Patch Tuesday: Microsoft added 14 modules to dbx

```
RT->GetVariable(L"IhisiParamBuffer", GUID, 0LL, &Size, &VarContent)
...
VarContent->param3 = 0LL;
VarContent->param5 = 0LL;
VarContent->param6 = 0LL;
VarContent->param1 = 0x83EFLL;
VarContent->param2 = '$H20';
VarContent->param4 = 0xB2LL;
...

```

VarContent is blindly trusted and used for multiple memory writes!

Double-Use Modules

- Focus on UEFI Shell: isolated incidents or **ecosystem-wide** issue?
- Large attack-surface, dangerous commands (*mm*) and scripts executed at startup (*startup.nsh*)

```
Shell> dmem 0x11223344 20
Memory Address 0000000011223344 20 Bytes
11223344: 00 00 00 00 00 00 00 00-00 00 00 00 00 *.....*
11223354: 00 00 00 00 00 00 00 00-00 00 00 00 00 *.....*

Shell> mm 0x11223344 DDCCBBAA -w 4

Shell> dmem 0x11223344 20
Memory Address 0000000011223344 20 Bytes
11223344: AA BB CC DD-00 00 00 00-00 00 00 00 *.....*
11223354: 00 00 00 00 00 00 00 00-00 00 00 00 *.....*
```

Double-Use Modules

- Focus on UEFI Shell: isolated incidents or **ecosystem-wide** issue?
- Large attack-surface, dangerous commands (*mm*) and scripts executed at startup (*startup.nsh*)
- Discovered **30 UEFI shells trusted by hundreds of devices**
 - 29 shells are signed with an OEM certificate present in **db**
 - 1 shell is trusted because its Authenticode hash was added to **db**
- Disclosure with CERT/CC is ongoing, **stay tuned** for more details!



From Trusted Shell to Untrusted Code Execution

- Core idea: use the *mm* command to overwrite *gSecurity2*

```
if (gSecurity2 != NULL) {  
    //  
    // Verify File Authentication through the Security2 Architectural Protocol  
    //  
    SecurityStatus = gSecurity2->FileAuthentication (  
        gSecurity2,  
        OriginalFilePath,  
        FHand.Source,  
        FHand.SourceSize,  
        BootPolicy  
    );
```

When gSecurity2 is NULL, Secure Boot is not enforced!

From Trusted Shell to Untrusted Code Execution

We developed and tested a PoC:

1. From a privileged OS shell:

- Copy the trusted UEFI shell and a *startup.nsh* script to the EFI System Partition
- Place a second unsigned UEFI module (the payload) on the partition
- Configure the Boot Manager to run the UEFI shell before the unsigned module



From Trusted Shell to Untrusted Code Execution

We developed and tested a PoC:

2. After rebooting the device:

- The Boot Manager runs the UEFI shell
- The UEFI shell automatically executes *startup.nsh*, which issues an *mm* command to zero *gSecurity2*
- The unsigned module containing the malicious payload executes successfully



binarly

Combining a Secure Boot Bypass with a Bootkit on Windows 11

Secure Boot Bypass
Windows 11 Bootkit

Windows 11 Secure Boot Bypass
Windows 11 Bootkit

Windows 11 Secure Boot Bypass
Windows 11 Bootkit

Windows 11 Secure Boot Bypass
Windows 11 Bootkit

Windows 11 Secure Boot Bypass

Windows 11 Bootkit

Windows 11 Secure Boot Bypass

Windows 11 Bootkit

Windows 11 Secure Boot Bypass
Windows 11 Bootkit

Windows 11 Secure Boot Bypass

Windows 11 Bootkit

Windows 11 Secure Boot Bypass
Windows 11 Bootkit

Windows 11 Secure Boot Bypass

Windows 11 Bootkit

Windows 11 Secure Boot Bypass
Windows 11 Bootkit

Windows 11 Secure Boot Bypass

Windows 11 Bootkit

Windows 11 Secure Boot Bypass
Windows 11 Bootkit

Windows 11 Secure Boot Bypass

Windows 11 Bootkit

Windows 11 Secure Boot Bypass
Windows 11 Bootkit

Windows 11 Secure Boot Bypass

Windows 11 Bootkit

Windows 11 Secure Boot Bypass
Windows 11 Bootkit

Windows 11 Secure Boot Bypass

Windows 11 Bootkit

Windows 11 Secure Boot Bypass
Windows 11 Bootkit

Windows 11 Secure Boot Bypass

Windows 11 Bootkit

Pull Request on Tianocore EDK2 repo

mediouni-m commented 4 days ago

When UEFI Secure Boot is on, we should not allow for the UEFI shell to be used. As such, disabling it in that scenario.

- Breaking change?
 - **Breaking change** - Does this PR cause a break in build or boot behavior?
 - Examples: Does it add a new library class or move a module to a different repo.
- Impacts security?
 - The UEFI shell is known as insecure.
- Includes tests?
 - **Tests** - Does this PR include any explicit test code?
 - Examples: Unit tests or integration tests.

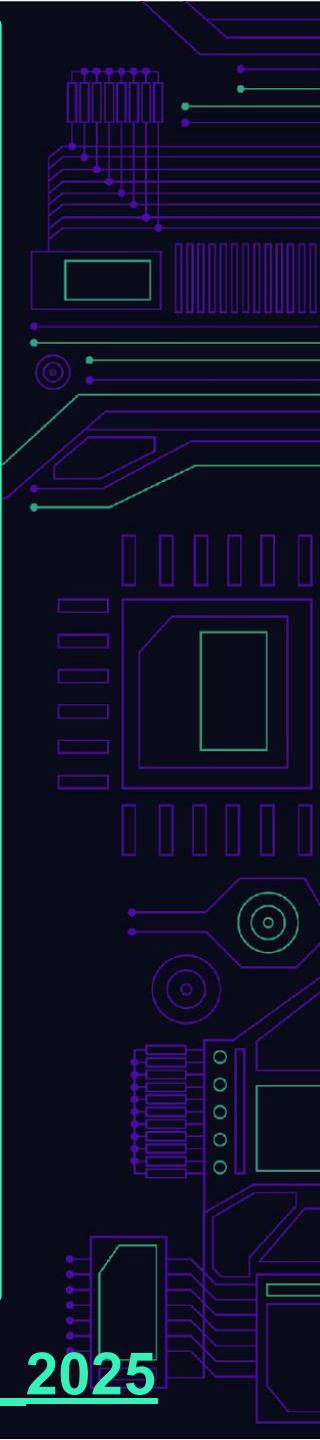
How This Was Tested

Has been on AWS since a while, see <https://github.com/aws/uefi/blob/5c3ac896feea3923/stable202211/0032-edk2-stable202211-uefi-shell-Disable-the-shell-when-UEFI-Secure-Bo>

```
ShellPkg/Application/Shell/Shell.c @@ -358,6 +358,17 @@ UefiMain (
358   358     EFI_HANDLE           ConInHandle;
359   359     EFI_SIMPLE_TEXT_INPUT_PROTOCOL *OldConIn;
360   360     SPLIT_LIST             *Split;
361 + 361     UINT8                 *SecureBoot;
362 +
363 + // If Secure Boot is enabled, do not launch the UEFI shell
364 + 364     SecureBoot = NULL;
365 + 365     GetEfiGlobalVariable2 (EFI_SECURE_BOOT_MODE_NAME, (VOID **) &SecureBoot, NULL);
366 + 366     if ((SecureBoot != NULL) && (*SecureBoot == SECURE_BOOT_MODE_ENABLE)) {
367 + 367         FreePool (SecureBoot);
368 + 368         return EFI_SECURITY_VIOLATION;
369 + 369     } else if (SecureBoot != NULL) {
370 + 370         FreePool (SecureBoot);
371 + 371     }
372
373     if (PcdGet8 (PcdShellSupportLevel) > 3) {
374         return (EFI_UNSUPPORTED);
375     }
376 }
```

Two security issues have been discovered in select Supermicro boards. These issues may affect Supermicro BMC Firmware.

CVE ID	Severity	Issue Type	Description
<u>CVE-2025-7937</u>	Medium	Improper Verification of Cryptographic Signature	A crafted firmware image can bypass the Supermicro BMC firmware verification logic of RoT 1.0 to update the system firmware. The crafted image has a customized PDBA table of RoT 1.0 to redirect the program to the fake PDBA table in the unsigned region. Supermicro CVSSv3 score: 6.6 (AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H)
<u>CVE-2025-6198</u>	Medium	Improper Verification of Cryptographic Signature	A crafted firmware image can bypass the Supermicro BMC firmware verification logic of Signing Table to update the system firmware. The crafted image has a customized signing table to redirect the program to the fake signing table in the unsigned region. Supermicro CVSSv3 score: 6.4 (AV:N/AC:H/PR:H/UI:R/S:U/C:H/I:H/A:H)



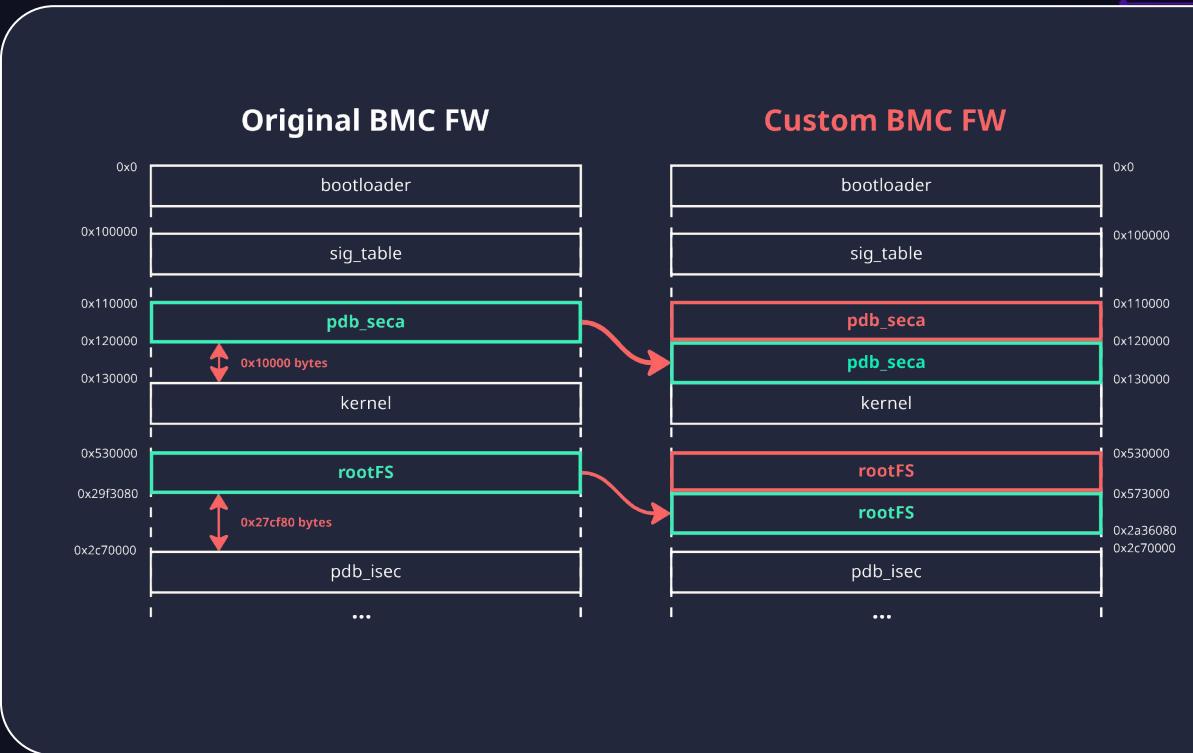
Firmware Validation Logic Bugs

- NVIDIA Offensive Security Research Team disclosed **2 stack overflows** and **1 design flaw** (CVE-2024-10237) in Supermicro BMC firmware validation process
- Validation based on the *fwmap* table + signature stored in the firmware image:

```
1. offset: 0x0000000, size:..., signed: true - bootloader  
2. offset: 0x0100000, size:..., signed: true - sig_table  
3. offset: 0x0110000, size:..., signed: true - pdb_seca  
4. offset: 0x0130000, size:..., signed: true - kernel  
5. offset: 0x0530000, size:..., signed: true - rootFS  
6. offset: 0x2dc0000, size:..., signed: false - pdb_isec
```

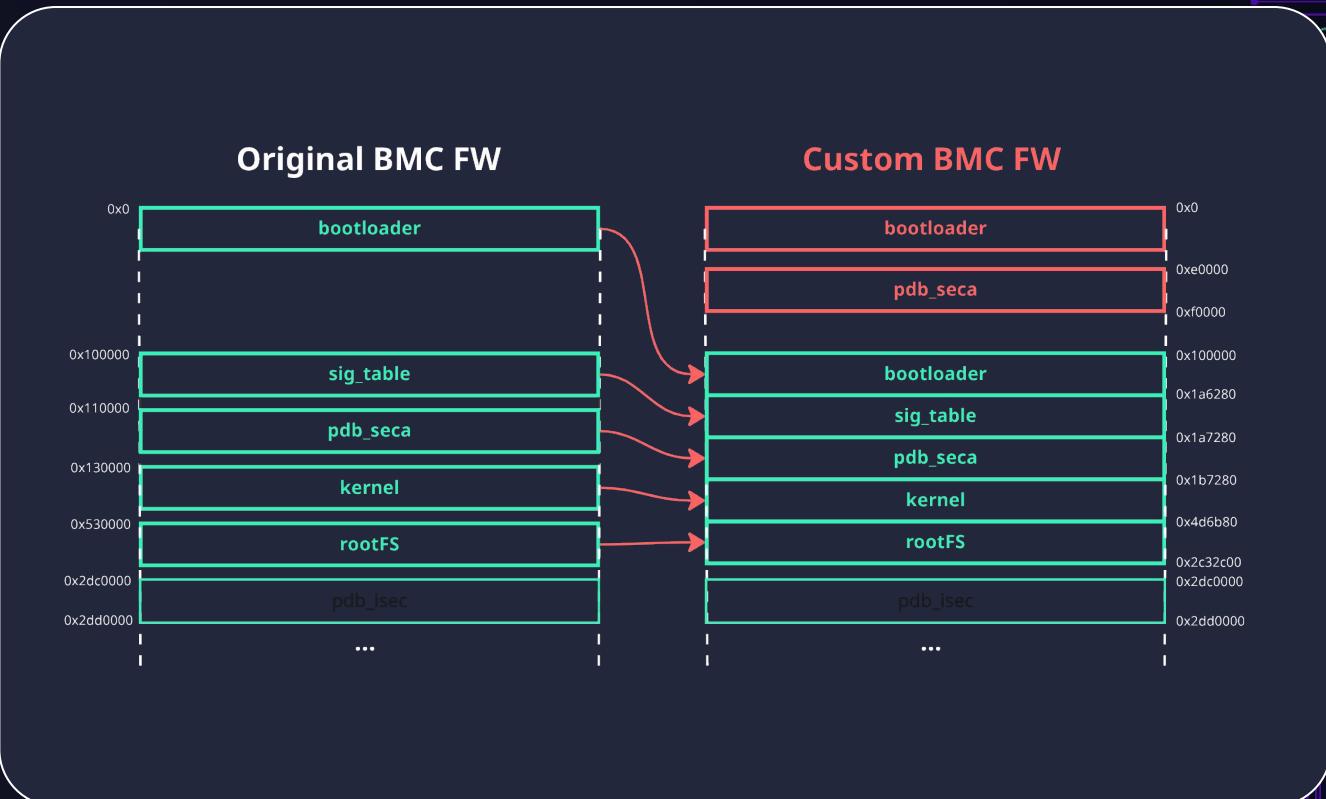
- Attack found by NVIDIA OSRT: move sections in the firmware image and update the *fwmap*:

```
1. offset: 0x0000000, size:..., signed: true - bootloader  
2. offset: 0x0100000, size:..., signed: true - sig_table  
3. offset: 0x0120000, size:..., signed: true - pdb_seca  
4. offset: 0x0130000, size:..., signed: true - kernel  
5. offset: 0x0573000, size:..., signed: true - rootFS  
6. offset: 0x2dc0000, size:..., signed: false - pdb_isec
```



FW Validation Bugs (CVE-2025-7937)

- Supermicro added checks on the offsets and attributes allowed in the *fwmap*
- Can these checks still be bypassed?
- CVE-2025-7937: Add a custom *fwmap* before the original one containing a **single element** (concatenation of all the regions) and swap the bootloader with a malicious one
 1. offset: 0x100000, size:..., signed: true - bootloader



<https://www.binarly.io/blog/broken-trust-fixed-supermicro-bmc-bug-gains-a-new-life-in-two-new-vulnerabilities>
<https://www.binarly.io/advisories/brly-2025-020>
<https://www.binarly.io/advisories/brly-2025-021>

Step 1: Select Type

Choose your type:



- Insert Component
- Insert Sub
- Insert Sub-Sub

Next

Conclusions

- Verification of firmware images is complex
- Secure Boot is a last line of defense against firmware-level threats
- Large number of signed modules in the wild → enrolling custom certificates if Secure Boot is a critical component
- Are UEFI-level threats coming?



ESET Research

Introducing HybridPetya: Petya/NotPetya copycat with UEFI Secure Boot bypass

UEFI copycat of Petya/NotPetya exploiting CVE-2024-7344 discovered on VirusTotal

Martin Smolár

12 Sep 2025 • 14 min. read



hasherezade
@hasherezade

I've got some really cool gift recently... UEFI Petya PoC:
youtube.com/watch?v=dMOiyp... 😊

 youtube.com
UEFI Petya PoC

2:17 PM · Sep 9, 2025 · 6,301 Views

<https://www.welivesecurity.com/en/eset-research/introducing-hybridpetya-petya-notpetya-copycat-uefi-secure-boot-bypass/>
<https://x.com/hasherezade/status/1965389009175412769>

LABS CON

Thank you



binarly

