binarly

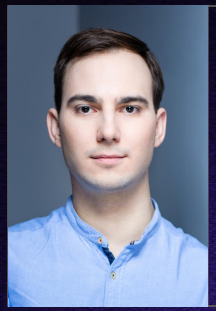
BINARLY 2026

Broken Trust: Firmware Bypass Chains, BMC Persistence, and EDR Evasion

Alex Matrosov, Fabio Pagani
@DistrictCon 1

binarly.io

Binarly REsearch Team



Anton Ivanov

@ant_av7

Alex Matrosov

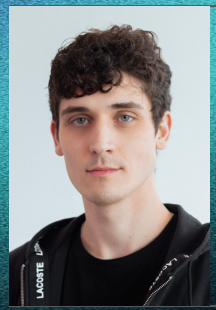
@matrosov

Fabio Pagani

@pagabuc

Sam L. Thomas

@xorpse



Yegor Vasilenko

@yeggorv

Introduction

The Invisible Foundation: Firmware is Everywhere

PERSONAL COMPUTING

Laptops, desktops

Enterprise servers

CORE INFRASTRUCTURE

Enterprise servers

Network appliances

CRITICAL SYSTEMS

ATMs

Voting machines

The Scale of Code in Modern Firmware

Project	# Lines of ASM*	# Lines of C*
SQLite	438k	183k
Linux kernel 6.18 (defconfig)	19.9M	6.5M
BMC (uboot + kernel + libs)	25.9M	8.9M
Laptop UEFI Firmware	30.1M	7.1M

* Counted using scc on the output of IDA Pro's 'Create ASM File' and 'Create C File'.

Firmware: A Reality Check

- Billions of heterogeneous devices across the entire computing stack
- Large codebase: millions of lines of C code
- Testing is non-trivial, highly hardware-dependent
- What can go wrong?

UEFI Vulnerabilities Are On The Rise

Vulnerability	Year	CVSS Score	CWEs
LogoFAIL	2023	6.7 (Medium) to 8.2 (High)	CWE-122: Heap-based Buffer Overflow CWE-125: Out-of-bounds Read CWE-190: Integer Overflow
PixieFail			
CVE-2025-33045	2025	6 (Medium)	CWE-123: Write-what-where Condition CWE-200: Sensitive Information Exposure
CVE-2025-4425	2025	8.2 (High)	CWE-121: Stack-based Buffer Overflow
CVE-2025-4422	2025	8.2 (High)	CWE-787: Out-of-bounds Write
BRLY-DVA-2025-011 (CVE-2025-7029)	2025	8.2 (High)	CWE-822: Untrusted Pointer Dereference

And hundreds more...

https://www.binarly.io/advisories
https://blog.quarkslab.com/pixiefail-nine-vulnerabilities-in-tianocores-edk-ii-ipv6-network-stack.html

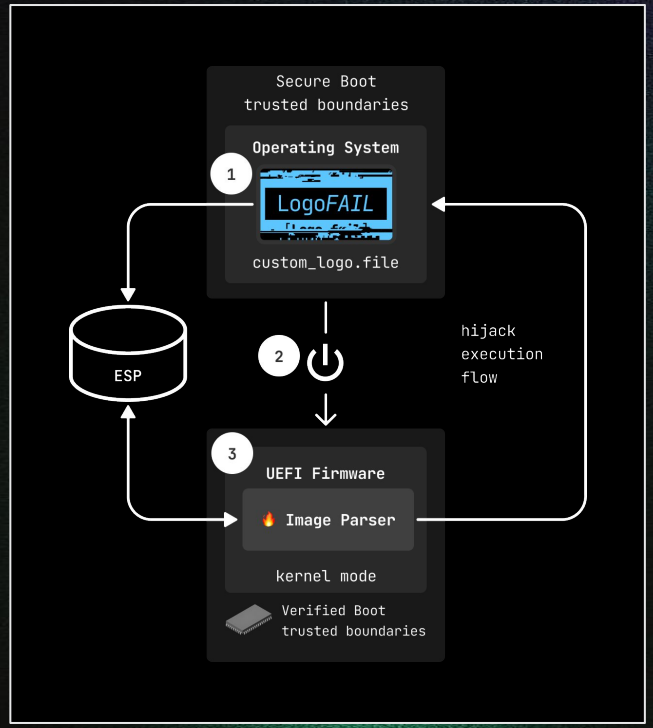
LogoFAIL

SUMMARY

- Majority of UEFI firmware contains image parsers
- Vendors allow customization of the logo displayed during boot
- Image parsers written in C, found crashes after seconds of fuzzing

DEVELOPING A POC

1. From the OS, store a malformed image on the ESP
2. Reboot the system
3. UEFI firmware parses the malformed image
4. Integer overflow to Heap overflow to DXE arbitrary code execution



https://www.binarly.io/blog/finding-logofail-the-dangers-of-image-parsing-during-system-boot
https://www.binarly.io/blog/inside-the-logofail-poc-from-integer-overflow-to-arbitrary-code-execution

Unknown Vulnerabilities Threatening the UEFI Ecosystem

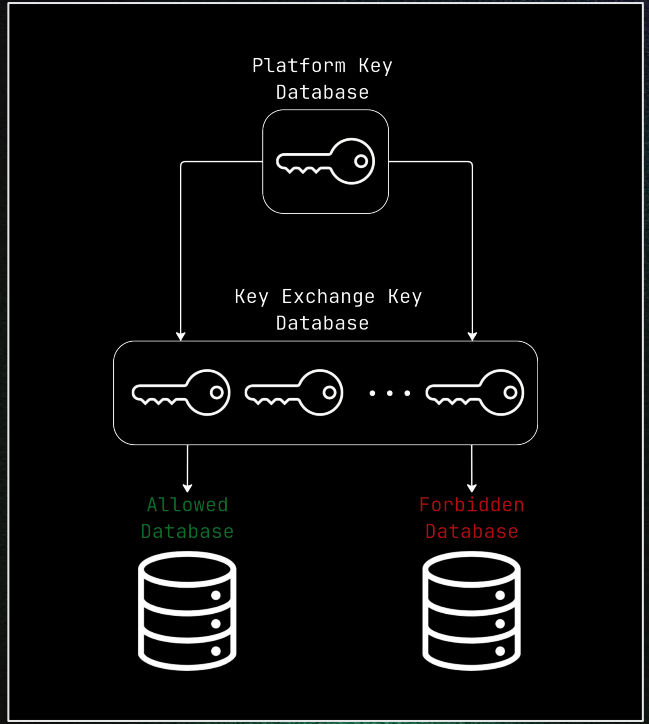
Vulnerability	CVSS Score	CWEs
DoubleGetVariable	8.2 (High)	CWE-787: Out-of-bounds Write
GetSetVariable	6.0 (Medium)	CWE-125: Out-of-bounds Read
PointerViaVariable (Memory Write)	8.2 (High)	CWE-787: Out-of-bounds Write CWE-822: Untrusted Pointer Dereference
PointerViaVariable (Function Call)	8.2 (High)	CWE-822: Untrusted Pointer Dereference CWE-829: Inclusion of Untrusted Functionality
SmmCommBuffer (Memory Write)	8.2 (High)	CWE-787: Out-of-bounds Write CWE-822: Untrusted Pointer Dereference
SmmCommBuffer (Callout)	8.2 (High)	CWE-822: Untrusted Pointer Dereference CWE-829: Inclusion of Untrusted Functionality
...		

Exploiting UEFI SMM Vulnerabilities For Persistent Implants, Nika Korchok Wakulich, OOTB2025BKK, https://www.youtube.com/watch?v=OzM2aGsiD1s

Secure Boot Vulnerabilities Impacting the Chain of Trust

Recent vulnerabilities impacting Secure Boot:

- PKfail
- Hydroph0bia (CVE-2025-4275)
- Broken dbx[1]
- Vulnerable signed module:
  - CVE-2025-3052 (Binarly)
  - CVE-2024-7344 (ESET)



1. https://www.binarly.io/blog/from-trust-to-trouble-the-supply-chain-implications-of-a-broken-dbx

PKfail


```
Version: 3 (0x2)
Serial Number:
    55:fb:ef:87:81:23:00:84:47:17:0b:b3:cd:87:3a:f4
Signature Algorithm: sha256WithRSAEncryption
Issuer: CN=DO NOT TRUST - AMI Test PK
Validity
    Not Before: Nov  8 23:32:53 2017 GMT
    Not After : Nov  8 23:32:52 2021 GMT
Subject: CN=DO NOT TRUST - AMI Test PK
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
        Modulus:
            00:e7:36:7b:20:92:ba:7f:aa:a3:f6:0e:49:08:87:
            f5:1c:11:33:ba:5d:f8:9b:5c:ed:c7:90:e4:f3:41:
...
```

Oh, hi! I am a private key, that's been available on GitHub for 6 months! 🤦

```
$ openssl pkcs12 -in FW_priKey.pfx -nodes
Enter Import Password:
```

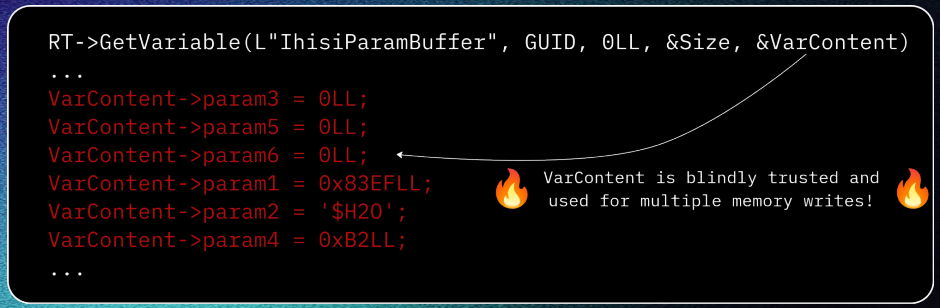
```
$ cat AmiTestKey.sdl | grep password -C3
TOKEN
        Name  = "FW_PFX_Password"
        Value  = "abcd"
        Help  = "Specifies the password to use when opening a PFX -
Private Key container file."
        TokenType = Expression
        TargetMAK = Yes
End
```

```
$ openssl x509 -noout -text -in FW_pubKey.cer  | rg "Issuer:|Subject:"
        Issuer: CN=DO NOT TRUST - AMI Test PK
        Subject: CN=DO NOT TRUST - AMI Test PK
```

https://www.binarly.io/pkfail

CVE-2025-3052

- Vulnerability found in module signed with Microsoft's third-party UEFI certificate ("*Microsoft Corporation UEFI CA 2011*")
- Secure Boot can be bypassed on any device trusting this key
- Microsoft added 14 new hashes to *dbx* as a mitigation during Patch Tuesday



Physical Attacks are (Sometimes) Out of Scope

Hello Alex,

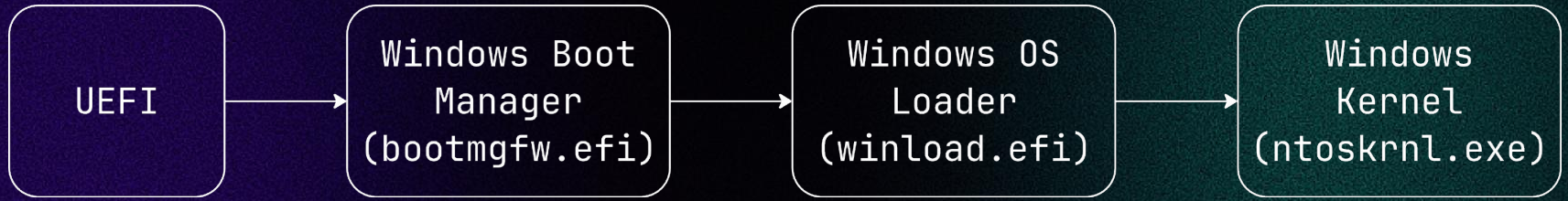
Thank you for your patience as our team diligently worked through this. After additional review and follow-up technical discussions, our product team stakeholders and PSIRT engineering concluded that the physical attack vector falls within the confines of a security weakness as opposed to a security vulnerability. The rational for that assessment is there would be persistent malicious code running in the BIOS, but not something that would be able to reach into the OS during boot handoff . The product teams will look into potential security hardening regarding this scenario, but at this time, our classification of these items will be considered as security weaknesses.

binarly

BINARLY 2026

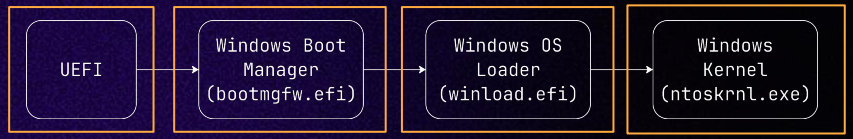
The Anatomy of UEFI Bootkits

High-Level Overview of the Windows Boot Process



https://www.binarly.io/blog/uefi-bootkit-hunting-in-depth-search-for-unique-code-behavior

The Anatomy of an UEFI Bootkit: redlotus-rs



01

Exploit a vulnerability in UEFI firmware, start bootkit infection chain

02

Load in memory Windows Boot Manager (bootmgfw.efi) and hook ImgArchStartBootApplication

03

During ImgArchStartBootApplication, hook BlImgAllocateImageBuffer OslFwpKernelSetupPhase1 and of Windows OS Loader (winload.efi)

04

During BlImgAllocateImageBuffer, allocate a buffer for the rootkit kernel driver

05

During OslFwpKernelSetupPhase1, patch a victim driver entrypoint to redirect execution to the rootkit driver

06

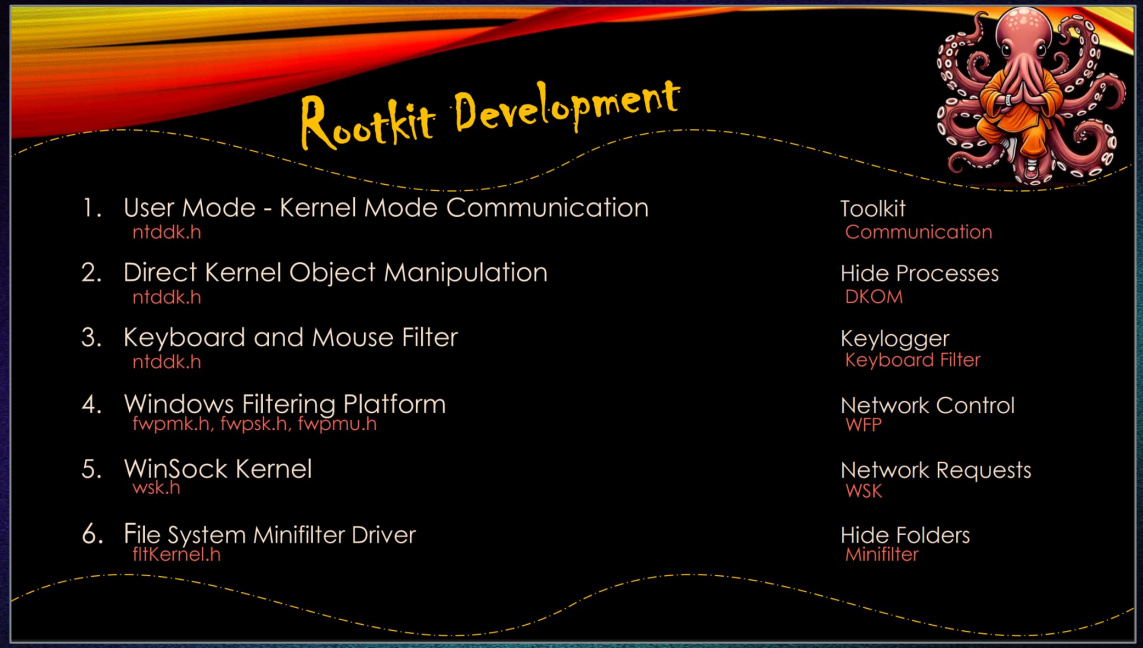
Execution is redirected to the rootkit kernel driver which hooks HalDispatchTable in the Windows Kernel (ntoskrnl.exe)

07

Windows completes the boot process with a rootkit installed


binarly
Combining a Secure Boot Bypass with a Bootkit on Windows 11

The Sky's the Limit

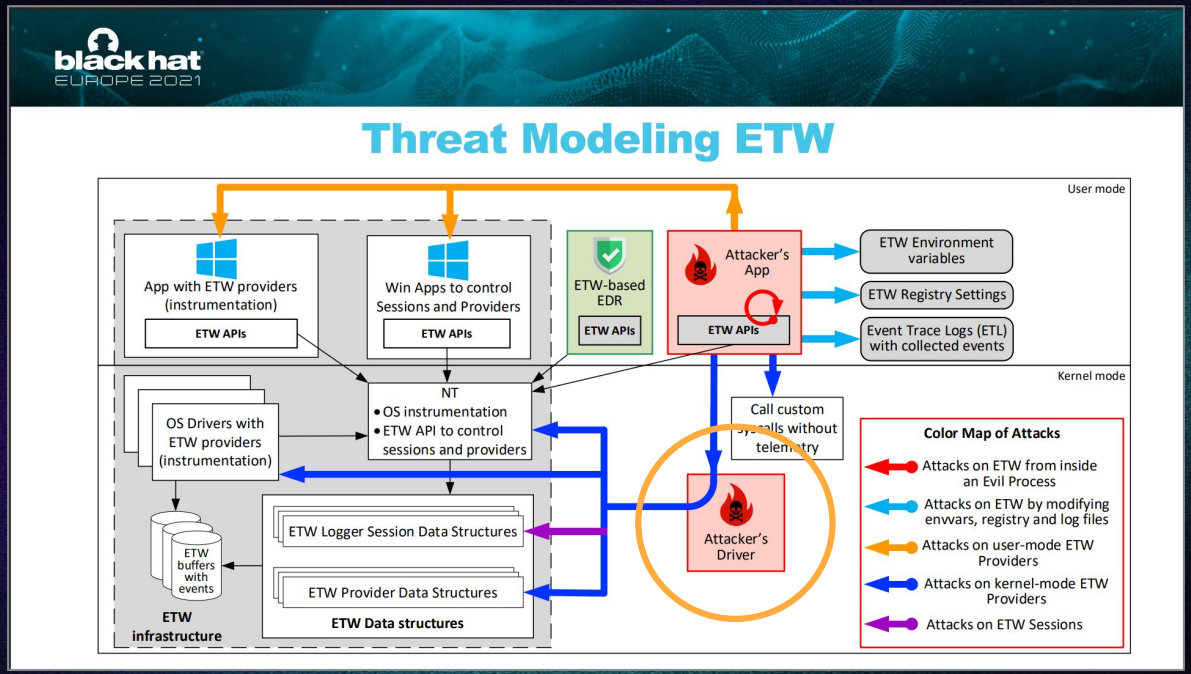


Infecting the Boot to Own the Kernel, Alejandro Vazquez, Maria San Jose, DEF CON 33, https://github.com/TheMalwareGuardian/Abyss

Event Tracing for Windows

- Event Tracing for Windows (ETW) is a native, high-performance Windows telemetry framework that records detailed kernel and user-mode system activity.
- EDR solutions leverage ETW to gain deep visibility into process execution, file operations, registry changes, and network activity using trusted OS-level signals.
- ETW is ideal for EDR because it provides telemetry, enabling real-time detection and forensic analysis without degrading system performance.

Event Tracing for Windows



Veni, No Vidi, No Vici: Attacks on ETW Blind EDR Sensors, Binarly REsearch Team, BH EU 2021

Event Tracing for Windows

CAN I WRITE MY OWN EVENTS?

- Make the EDR believe ‘things’ happened, for instance for impersonating attacks which are too risky or complicated to run.
- Use it offensively for creating distractions or spoofing events.
- Since most cloud based EDRs have caps on events, we potentially can create blind spots.

I'm in your logs now, deceiving your analysts and blinding your EDR, Olaf Hartong, BH US 2025

binarly

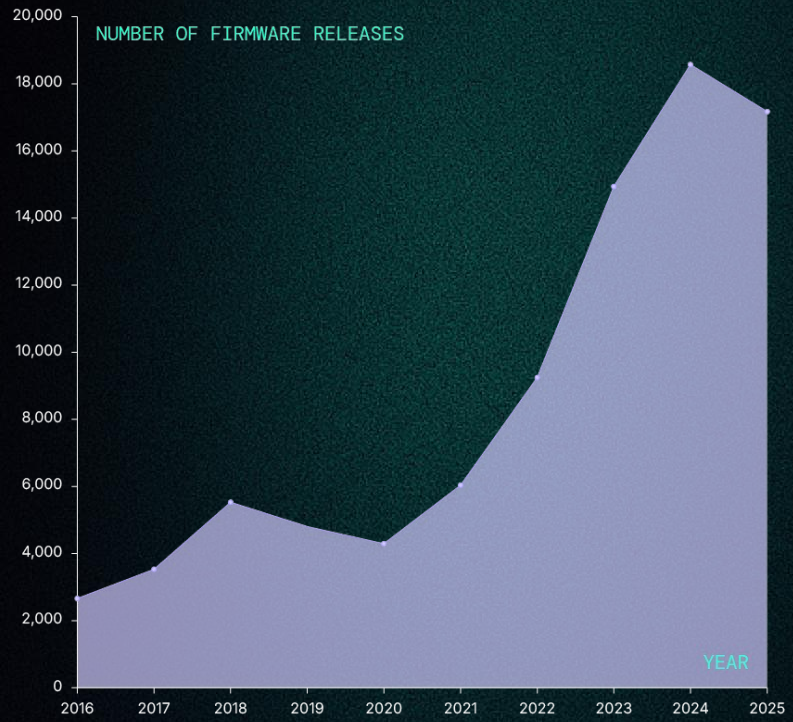
BINARLY 2026

A Look Inside the UEFI Ecosystem

Binarly's Dataset of UEFI Firmware

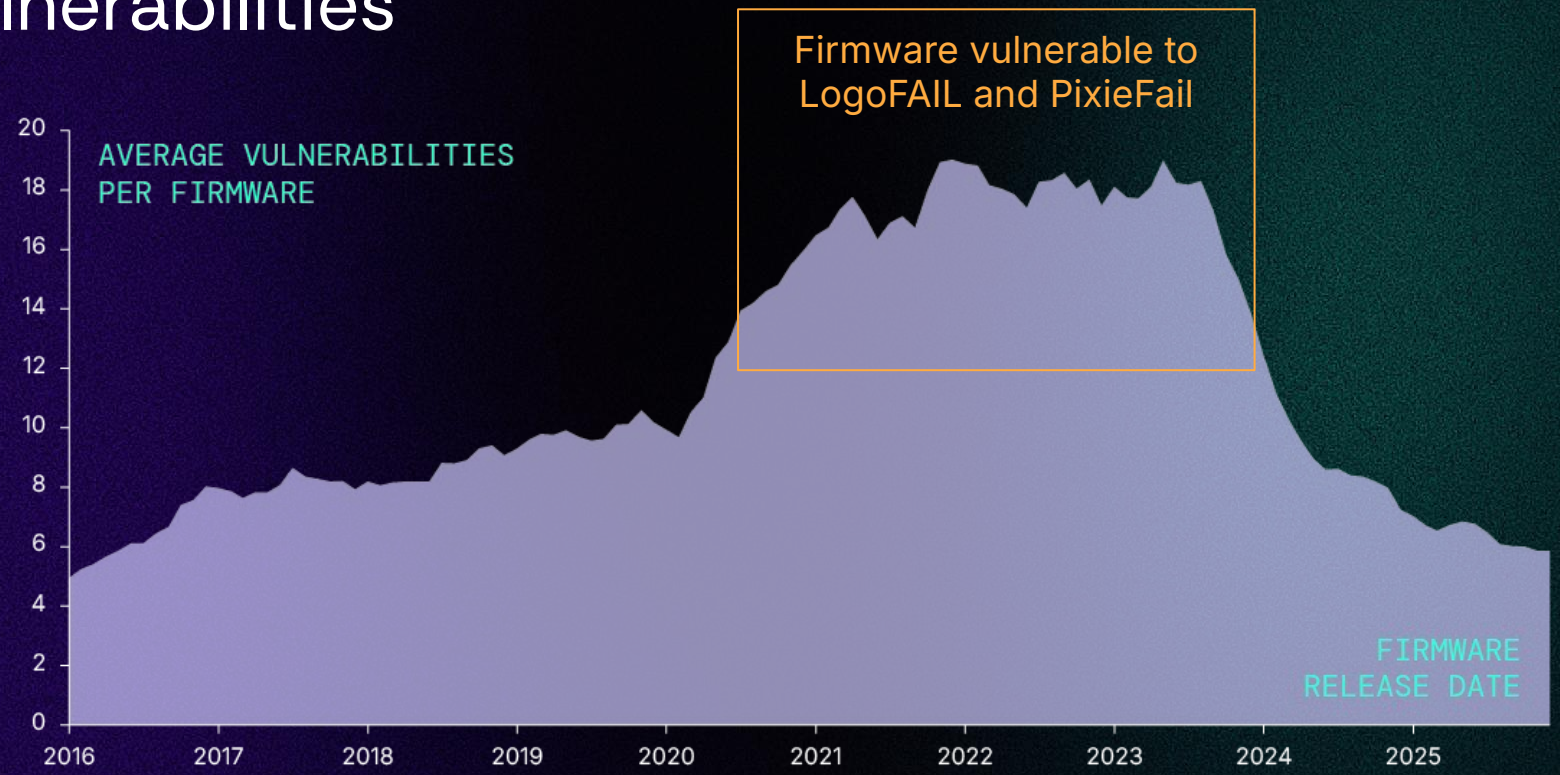
Dataset with 80,000 UEFI firmware images:

- Spanning over 10 years
- Includes every major vendor (Lenovo, Dell, HP, Intel..)
  - Tracking around 10,000 of recent device models
  - At least one firmware released in the past 4 years
  - 25% can be considered EOL (no firmware released in the last 2 years)

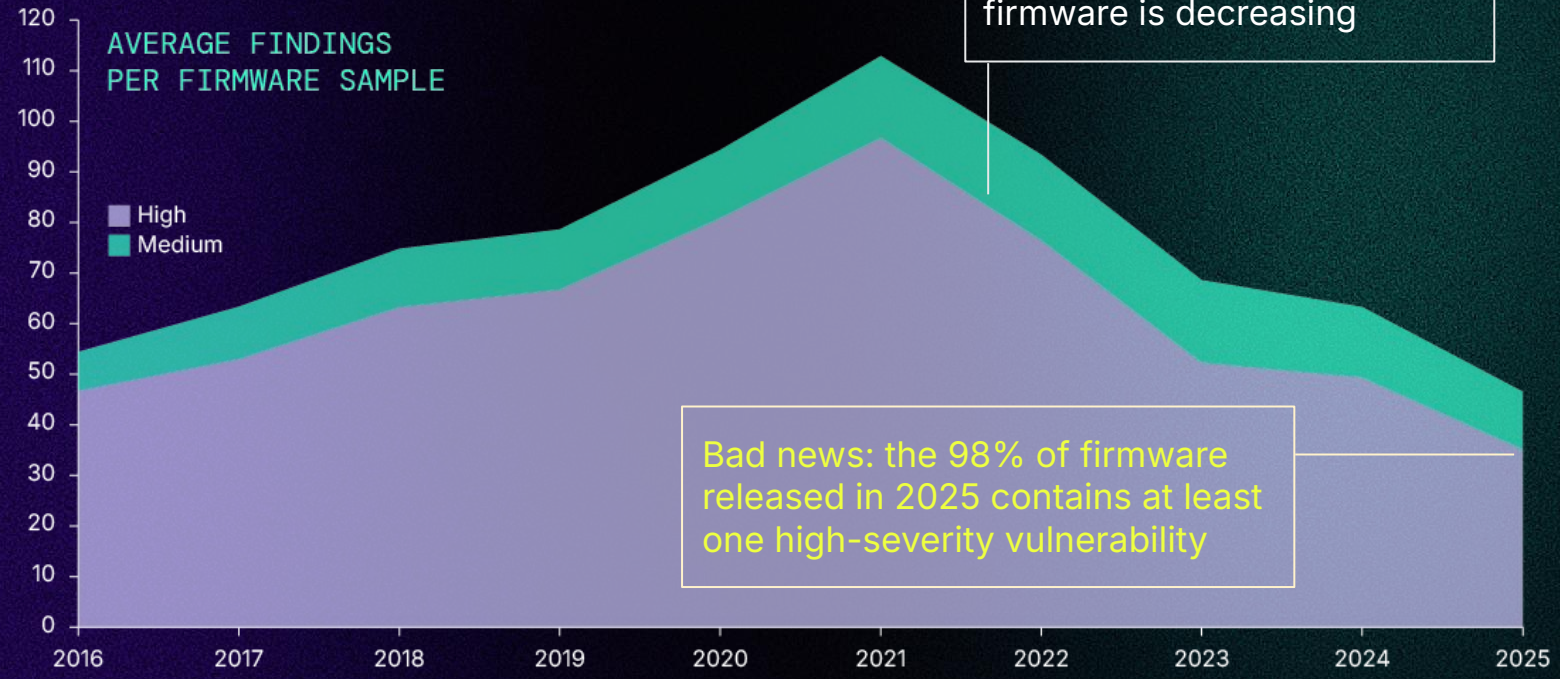


BINARLY 2026

Impact on Known Firmware Vulnerabilities



Impact on Unknown Firmware Vulnerabilities



Latest From the Trenches

ESET Research has discovered HybridPetya, on the VirusTotal sample sharing platform. It is a copycat of the **infamous Petya/NotPetya malware**, adding the capability of **compromising UEFI-based systems** and weaponizing CVE-2024-7344 to **bypass UEFI Secure Boot** on outdated systems.

binarly

BINARLY 2026

BMC REsearch

The long chain of Supermicro BMC firmware fixes

- It all started with CVE-2024-10237

CVE-2024-10237 Detail

AWAITING ANALYSIS

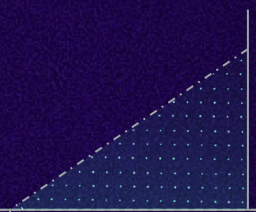
This CVE record has been marked for NVD enrichment efforts.

Description

There is a vulnerability in the BMC firmware image authentication design at Supermicro MBD-X12DPG-OA6 . An attacker can modify the firmware to bypass BMC inspection and bypass the signature verification process

The long chain of Supermicro BMC firmware fixes

- It all started with CVE-2024-10237
- It took Supermicro one year and three release cycles to resolve the issues
- Fixes for CVE-2025-12006 and CVE-2025-12007 were released in January 2026



JANUARY 2025

The fix for **CVE-2024-10237** was released

MAY 2025

Binarly REsearch bypasses **CVE-2024-10237** fix and finds another vulnerability in the alternative logic

SEPTEMBER 2025

Fixes for **CVE-2025-7937** and **CVE-2025-6198** were released

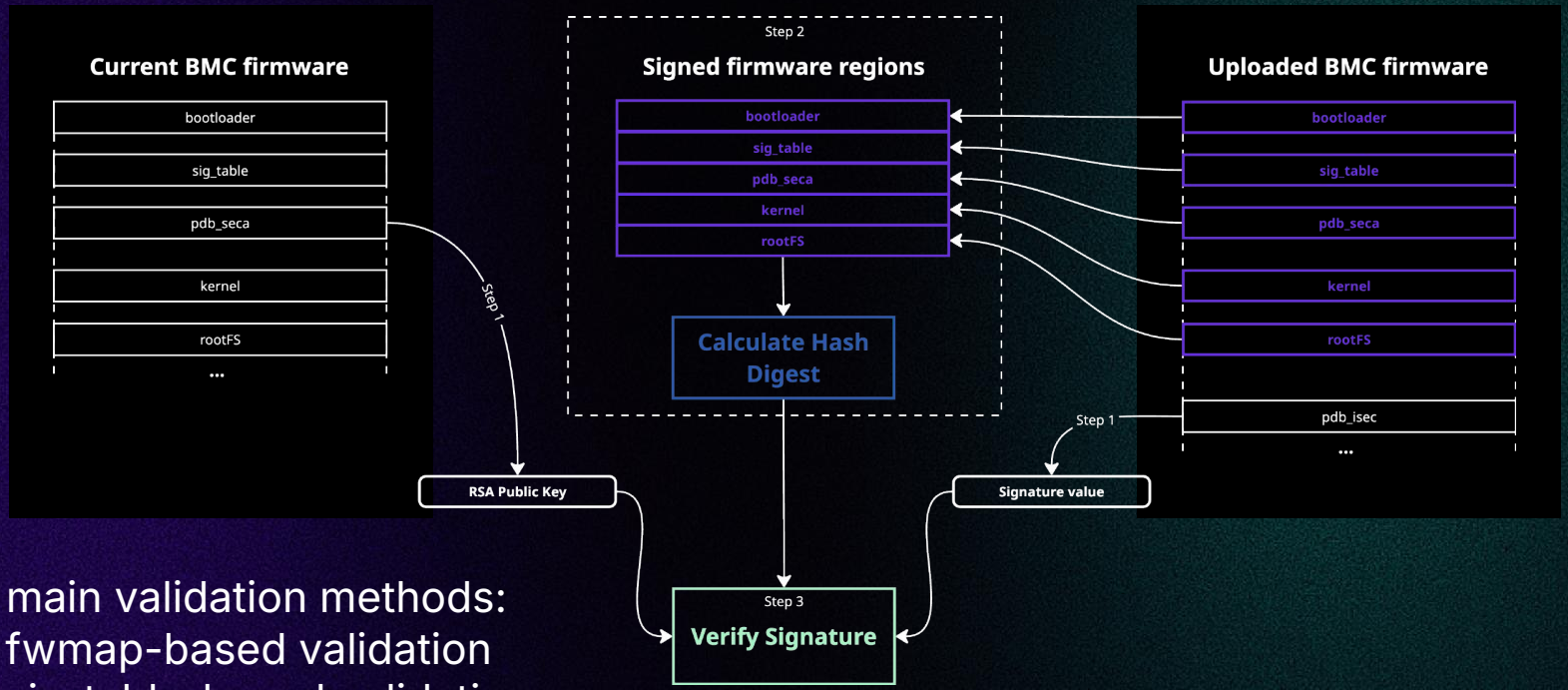
OCTOBER 2025

Binarly REsearch bypasses both **CVE-2025-7937** and **CVE-2025-6198** fixes

JANUARY 2026

Fixes for **CVE-2025-12006** and **CVE-2025-12007** were released

Supermicro BMC validation



Two main validation methods:

- fwmap-based validation
- sig_table-based validation

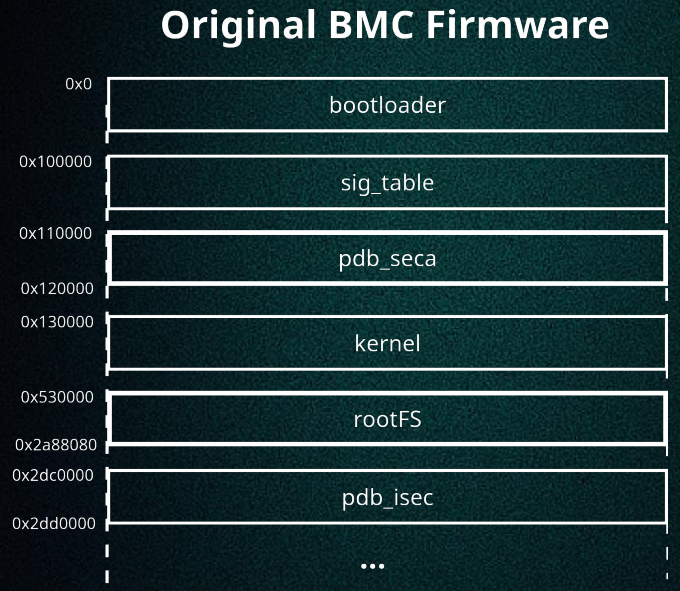
fwmap-based validation

fwmap table contains information about the firmware regions:

- offset
- size
- attributes (e.g. whether the region is signed or not)

fwmap from Supermicro X12STW-F

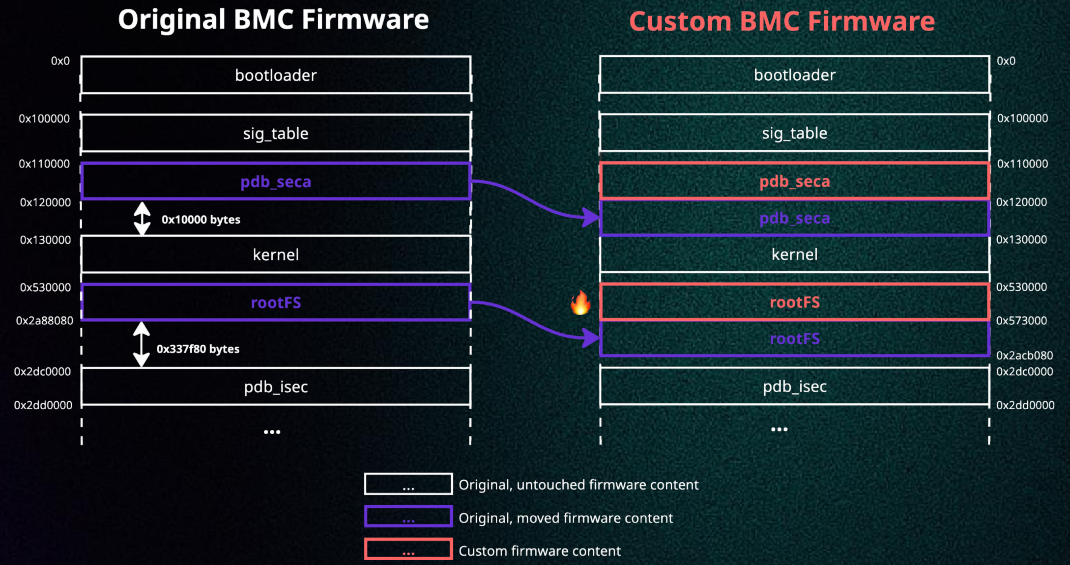
1. offset: `0x0000000`, size: `0x00a5400`, signed: `true` - `bootloader`
2. offset: `0x0100000`, size: `0x0001000`, signed: `true` - `sig_table`
3. offset: `0x0110000`, size: `0x0010000`, signed: `true` - `pdb_seca`
4. offset: `0x0130000`, size: `0x0325a00`, signed: `true` - `kernel`
5. offset: `0x0530000`, size: `0x2558080`, signed: `true` - `rootFS`
6. offset: `0x2dc0000`, size: `0x0010000`, signed: `false` - `pdb_isec`



CVE-2024-10237: PoC

Custom *fwmap*

1. offset: `0x0000000` , size: `0x00a5400` , signed: `true` - `bootloader`
2. offset: `0x0100000` , size: `0x0001000` , signed: `true` - `sig_table`
3. offset: `0x0120000` , size: `0x0010000` , signed: `true` - `pdb_seca`
4. offset: `0x0130000` , size: `0x0325a00` , signed: `true` - `kernel`
5. offset: `0x0573000` , size: `0x2558080` , signed: `true` - `rootFS`
6. offset: `0x2dc0000` , size: `0x0010000` , signed: `false` - `pdb_isec`



CVE-2024-10237: Demo

```
[FWUP]D[dump_signdata_cb]: Entry
[FWUP]D[dump_signdata_cb]: Data::(3dd1a008, 00000000) not signed, bypass.
[FWUP]D[fwmap_read_by_index]: FWMAP has 10 entries.
[FWUP]D[fwmap_parser]: callback on FwMap().
[FWUP]D[dump_signdata_cb]: Entry
[FWUP]D[dump_signdata_cb]: Data::(3ae9a008, 00000000) not signed, bypass.
[FWUP]D[fwmap_read_by_index]: FWMAP has 10 entries.
[FWUP]W[fwmap_read_by_index]: Index out of limit (10/10)!
[FWUP]W[fwmap_parser]: Get FwMap[10] failed, rc = -2!
[FWUP]D[fwmap_parser]: Done with rc = 0.
[FWUP]D[bmc_validation_check]: signdata_bio: 0x29848.
[FWUP]D[SignedFileSignatureValidation]: Entry.
[FWUP]D[DataSignatureValidation]: Entry.
[FWUP]D[ValidationPkcs7]: Entry.
[FWUP]D[VerifyPkcs7Data]: Entry.
[FWUP]D[VerifyPkcs7Data]: Verifying begin...
[FWUP]D[VerifyPkcs7Data]: Verify Pass.
```

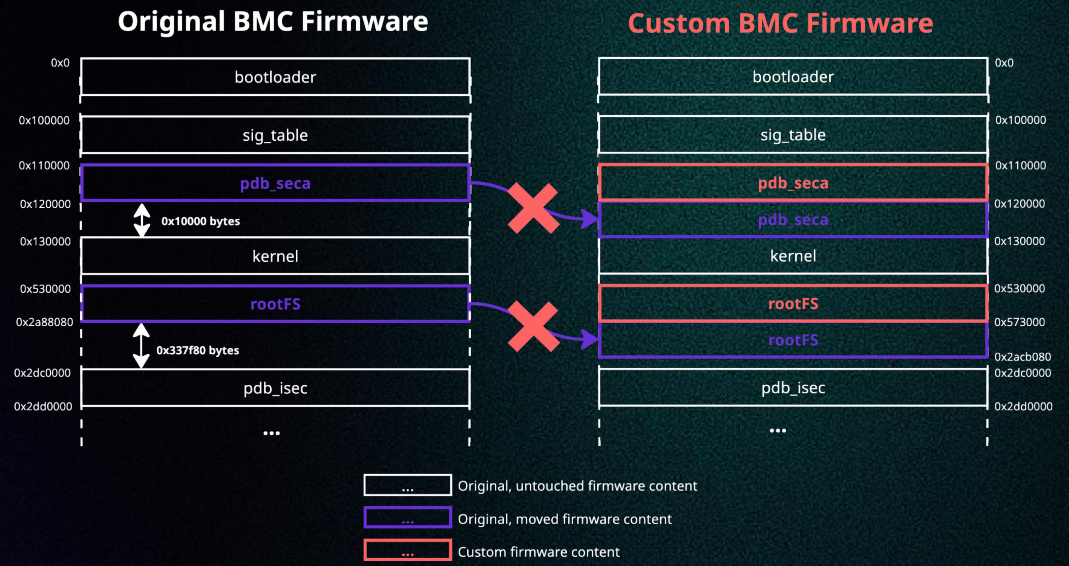
```
[    1.419853] peci-aspeed 1e78b000.peci-bus: peci bus 0 registered, irq 61
[    1.421382] ipip: IPv4 and MPLS over IPv4 tunneling driver
[    1.426010] NET: Registered protocol family 10
[    1.430252] Segment Routing with IPv6
[    1.432220] sit: IPv6, IPv4 and MPLS over IPv4 tunneling driver
[    1.434068] NET: Registered protocol family 17
[    1.434883] 8021q: 802.1Q VLAN Support v1.8
[    1.435293] Registering SWP/SWPB emulation handler
[    1.436223] Loading compiled-in X.509 certificates
[    1.442064] printk: console [netcon0] enabled
[    1.442285] netconsole: network logging started
[    1.442840] hctosys: unable to open rtc device (rtc0)
[    1.454587] VFS: Mounted root (squashfs filesystem) readonly on device 31:2.
[    1.492460] Freeing unused kernel memory: 1024K
[    1.557538] Checked W+X mappings: passed, no W+X pages found
[    1.557897] Run /sbin/init as init process
BINARLY RESEARCH
```

CVE-2024-10237: Supermicro's Patch

- No custom region offsets in *fwmap*, only **whitelisted** offsets can be used
- Only certain regions can have ***is_signed*** flag

Custom *fwmap*

1. offset: `0x0000000` , size: `0x00a5400` , signed: `true` - `bootloader`
2. offset: `0x0100000` , size: `0x0001000` , signed: `true` - `sig_table`
3. offset: ~~`0x0120000`~~ , size: `0x0010000` , signed: `true` - `pdb_seca`
4. offset: `0x0130000` , size: `0x0325a00` , signed: `true` - `kernel`
5. offset: ~~`0x0573000`~~ , size: `0x2558080` , signed: `true` - `rootFS`
6. offset: `0x2dc0000` , size: `0x0010000` , signed: `false` - `pdb_isec`

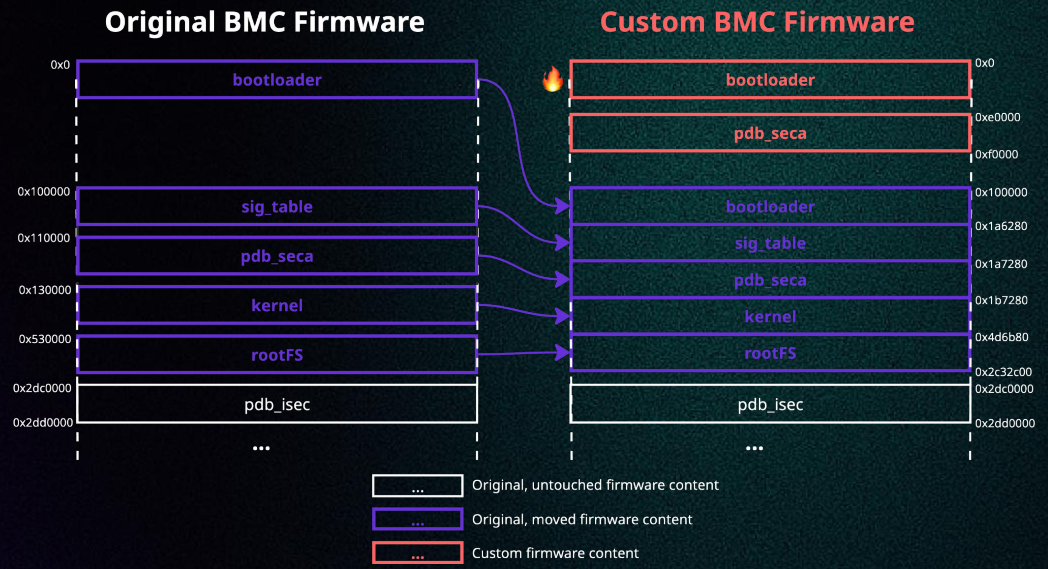


CVE-2024-10237: Bypassing the patch

- Move all the signed regions at whitelisted offset 0x100000
- Add entry in the custom *fwmap* and name it bootloader

Custom *fwmap*

```
1. offset: 0x100000 , size: 0x2b32c00 , signed: true - bootloader
```

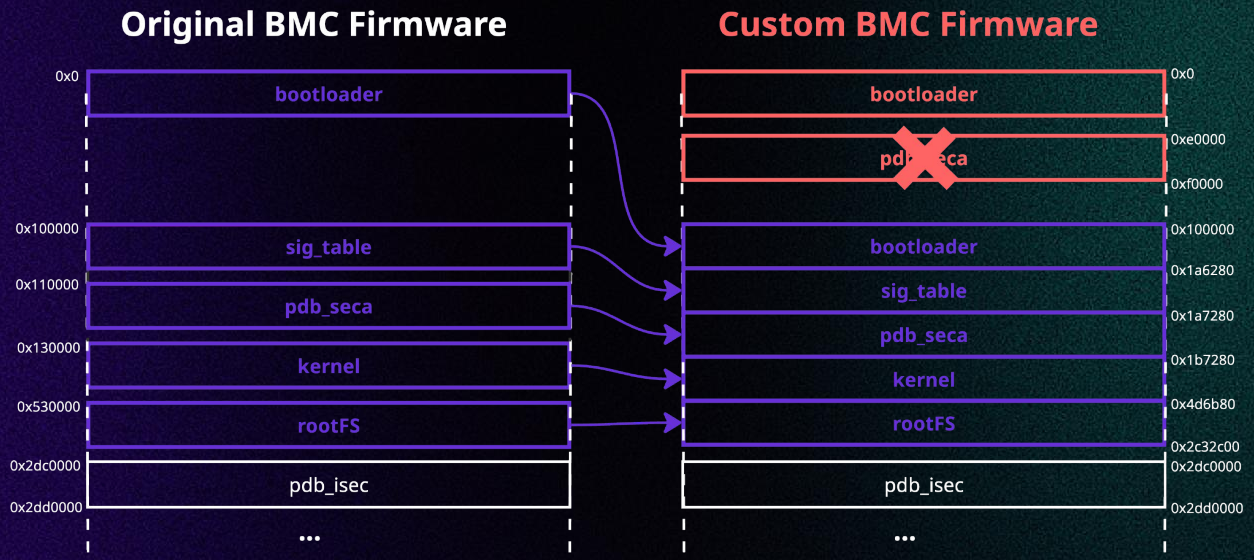


CVE-2025-7937: Demo


```
BP0c00


U-Boot 2019.04 (BINARLY RESEARCH)

SOC: AST2600-A3
PWM1: Enable fan0 and fan1
Hit any key to stop autoboot:  1    Trying 'kernel@1' kernel subimage
     Description:  Linux kernel
     Type:         Kernel Image
     Compression:  uncompressed
+ OK
## Loading fdt from FIT Image at 20130000 ...
   Using 'conf@aspeed-ast2600a1-evb.dtb' configuration
     Description:  Flattened Device Tree blob
[    1.127936] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHC[    1.142070] i2c /dev entries driver
[    1.147445] i2c_new_aspeed 1e78a080.i2c-bus: NEW-I2C: i2c-bus[    1.201031] i2c_new_aspeed 1e78a380.i2c-bus: NEW-I2C: i2c-busz] mode [2]
[    1.233330] i2c_new_aspeed 1e78a500.i2c-bus: NEW-I2C: i2c-bus [10]: adapter [100 khz] mode [2]
80000000, resource_size=0x1f000000, PAGE_SHIFT macro=0xc
 controller MIC: DEV 1e6e0000.sdram (INTERRUPT)
[    1.327206] ASPEED RSA Accelerator successfully registered
[    1.340179] usbhid: USB HID core driver
[    1.351795] peci_aspeed 1e78b000.peci-bus: Expect frequency: registered as minor 0
[    1.371226] peci_aspeed 1e78b000.peci-bus: peci bus 0 registe[    1.379486] ipip: IPv4 and MPLS over IPv4 tunneling driver
NU/Linux
BusyBox v1.35.0 (2025-06-21 00:11:57 PDT) multi-call binary.

ODE     Creation mode (default a=rw)
TYPE:
        b       Block device
        c or default a=rw)
TYPE:
        b       Block device
        c or u  Character device
        p       Named pipe (MAJOR MINOR must be omitted)
BusyBox v1.35.0 (2025-06-21 00:11:57 PDT) multi-call binary.
```

CVE-2025-7937: Supermicro's patch

- Checks that offset of processed *pdb_seca* is 0x110000
- *fwmap* must contain a region where offset <= *pdb_seca offset* < *offset + size*

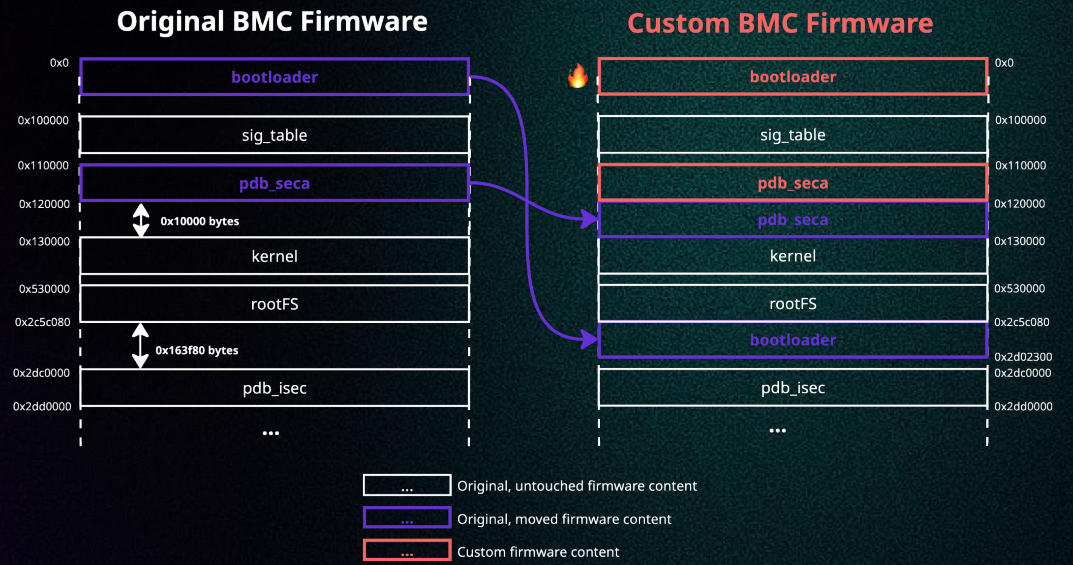


CVE-2025-7937: Bypassing the second patch

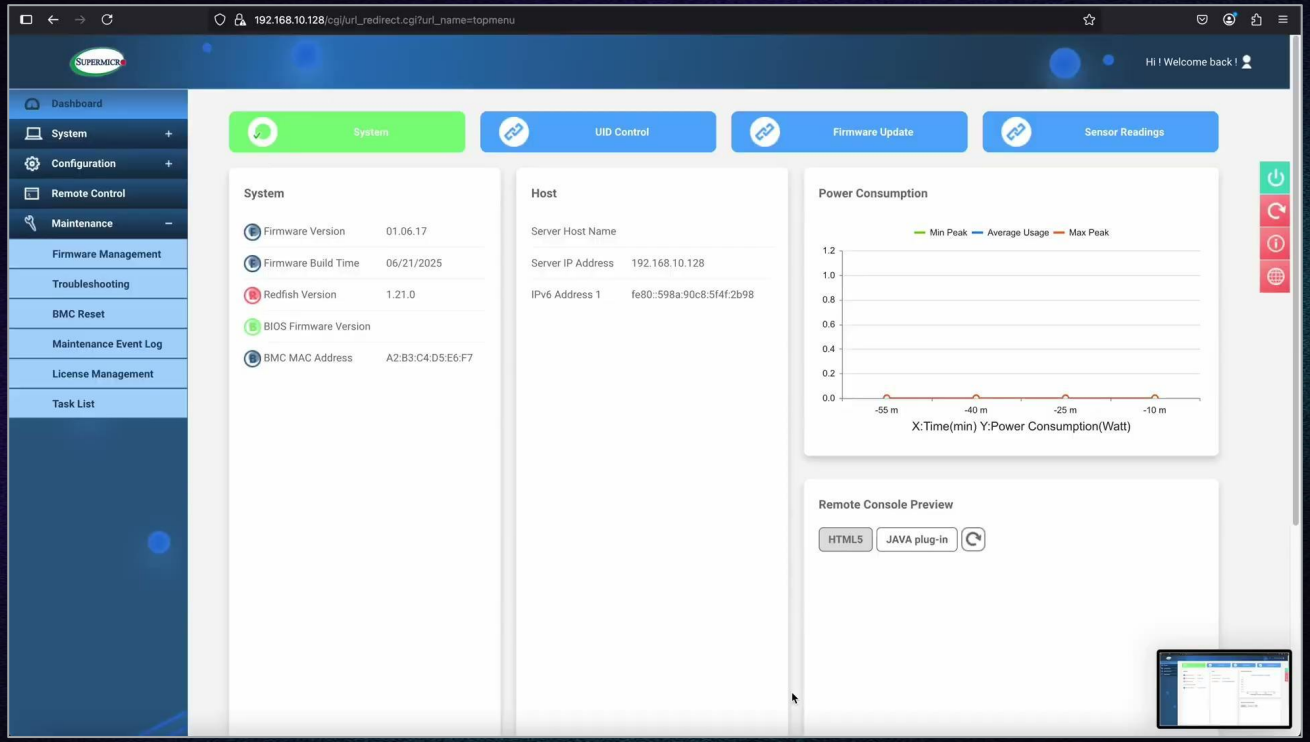
- Previously implemented checks were removed :)
  - We can add *fwmap* entries at custom offsets again!

Custom *fwmap*

```
1. offset: 0x2c5c080 , size: 0x00a6280 , signed: true - bootloader
2. offset: 0x0100000 , size: 0x0001000 , signed: true - sig_table
3. offset: 0x0120000 , size: 0x0010000 , signed: true - pdb_seca
4. offset: 0x0130000 , size: 0x031f880 , signed: true - kernel
5. offset: 0x0530000 , size: 0x272c080 , signed: true - rootFS
6. offset: 0x2dc0000 , size: 0x0010000 , signed: false - pdb_isec
7. offset: 0x2dd0000 , size: 0x0000000 , signed: false - nvram1
8. offset: 0x2e80000 , size: 0x0000000 , signed: false - uboot_env
9. offset: 0x0110000 , size: 0x0000001 , signed: false - nvram
```

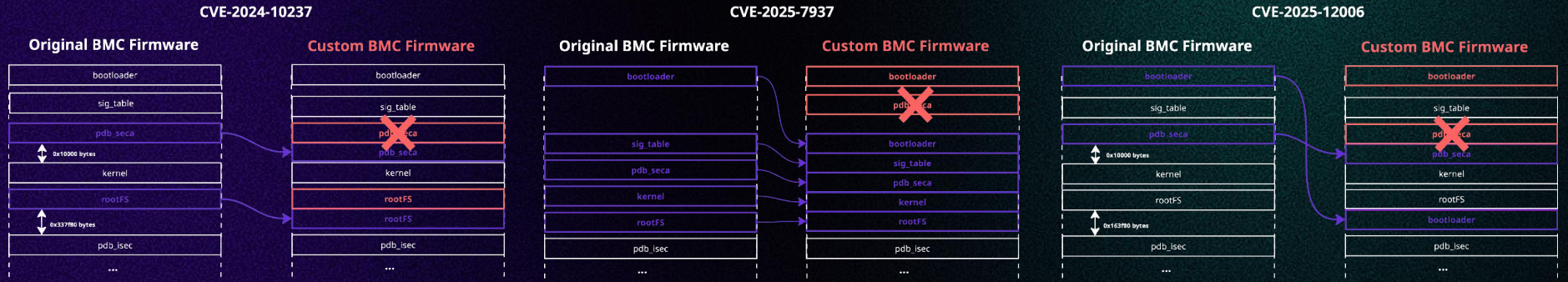


CVE-2025-12006: Demo



CVE-2025-12006: Supermicro’s final patch

- Offset of parsed pdb_seca should be equal to 0x110000
- For pdb_seca region defined in *fwmap*:
  - offset should be 0x110000
  - size should be 0x10000
  - it should have is_signed attribute
- Other fwmap regions should be located at only allowed offsets
  - For some regions, their size and attributes are also checked



CVE-2025-12006: Supermicro’s final patch

Fixes provided with the latest firmware release mitigate the issues, **but**:

- For both X12STW-F (*fwmap*) and X13SEM-F (*sig_table*), RSA keys used for image signing were not rotated
  - Firmware downgrade is not possible due to other changes, but may arise in the future
- For X13SEM-F, the required validation logic was added to the *libipmi.so* library, but before it was executed in the *OP-TEE* environment
  - Potential attackers with root privileges to the BMC system could bypass the introduced checks

What about *sig_table*-based validation?

- Similar logic, similar problems – CVE-2025-6198, CVE-2025-12007
- Blogpost coming soon, stay tuned!

Conclusions

- Firmware is ubiquitous, complex and not tested enough
- Number of bugs in the UEFI ecosystem are declining, still almost every firmware out there has 1+ high-severity bug
- Bugs in UEFI can impact the boot process and OS integrity
- BMC firmware validation is not a trivial task