The Firmware Supply-Chain Security is broken!

Can we fix it?

SPEAKER

Alex Matrosov

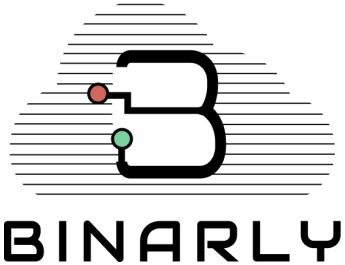
Alex Ermolov

Kai Michaelis

Richard Hughes

BINARLY - Pasadena, CA

https://www.binarly.io



Protect Devices from emerging firmware and hardware threats using modern artificial intelligence.

Provide an advanced analytics platform for Security Operations Center and Incident Response teams for enhanced visibility into the enterprise device infrastructure.

founders@binarly.io

IMMUNE - **Bochum, Germany**

https://www.immu.ne/

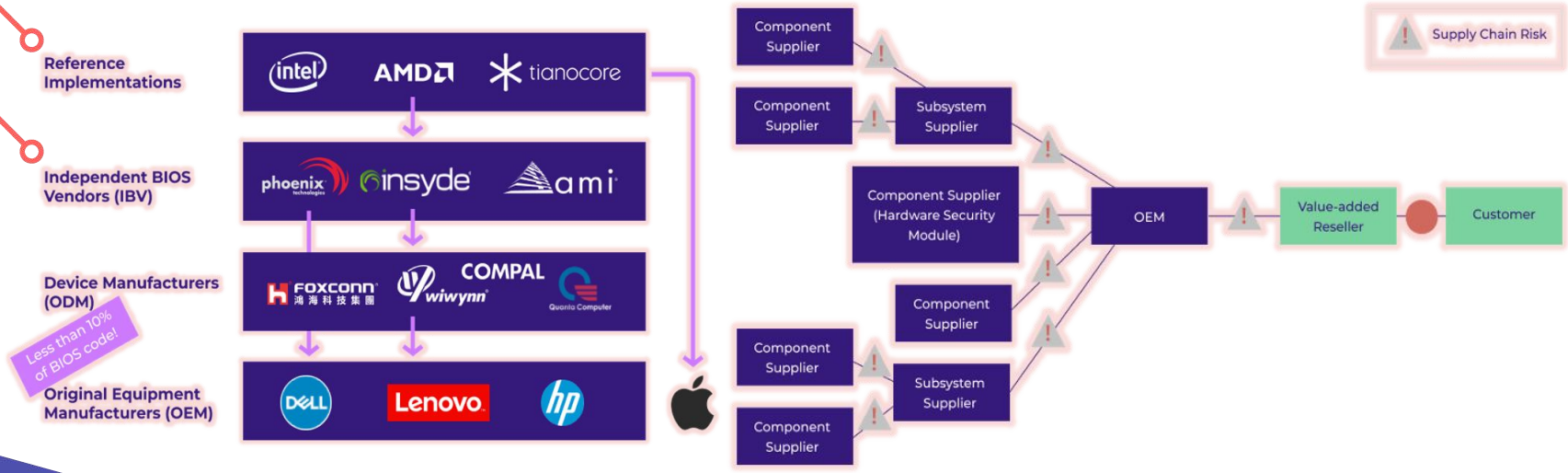
Provies **attestation as a service** for data center and edge compute.

In depth analysis of firmware, provisioned hardware configuration and boot chain.

Provides **firmware monitoring and risk assessment** across the whole server fleet.

contact@immu.ne

Supply Chain complexity keeps 1-days unfixed for years



Firmware Vulnerability Disclosure Time

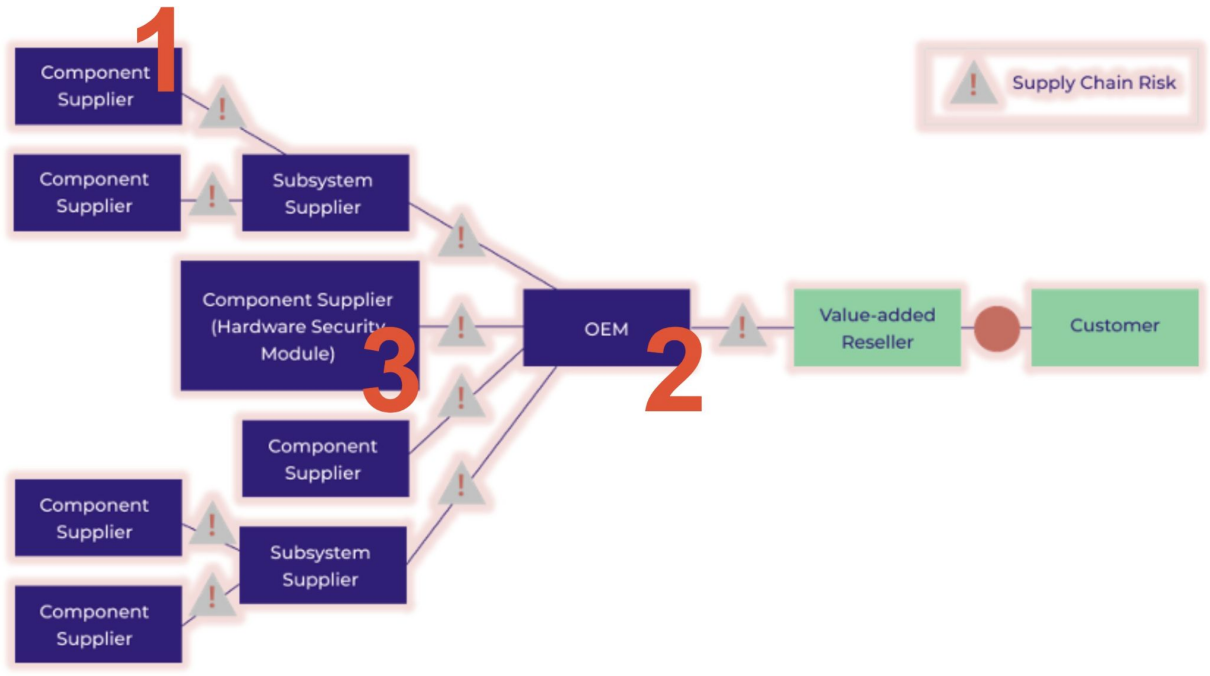
Combination of vulnerabilities creates a successful attack vector for compromising firmware:

1. **Privileges Escalation**
2. **FW storage write-protection bypass**
3. **HW-based trusted boot bypass**

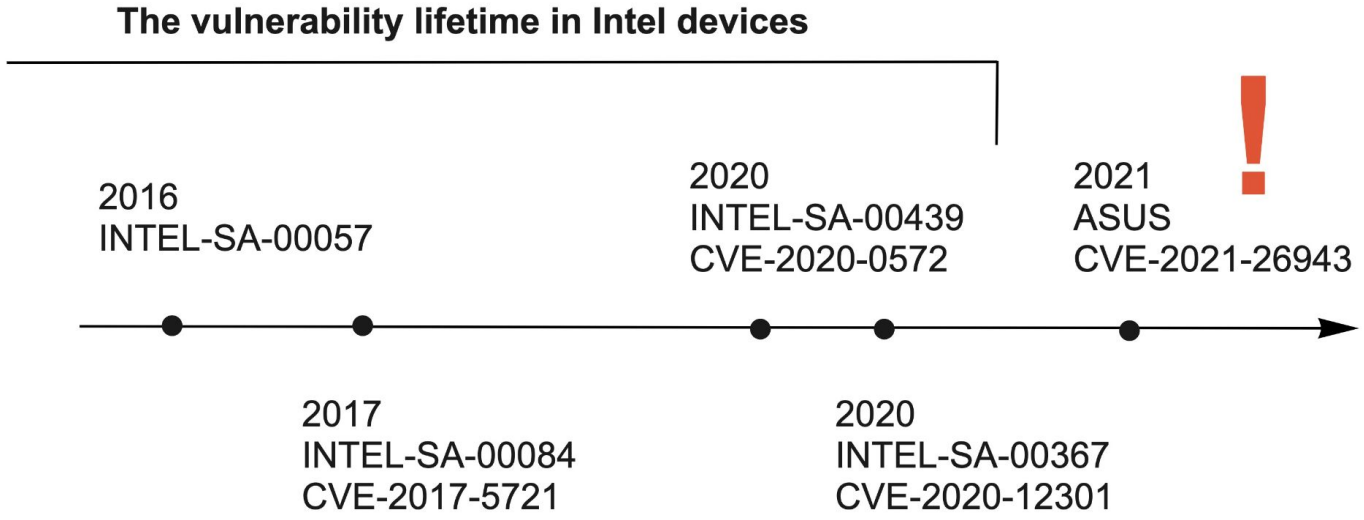
Patch delivery timeline could be different for each vulnerability in the attack vector.

A single vulnerability could be replaced by another one (not necessarily equal impact) to support the attack vector.

Supply Chain security failures points



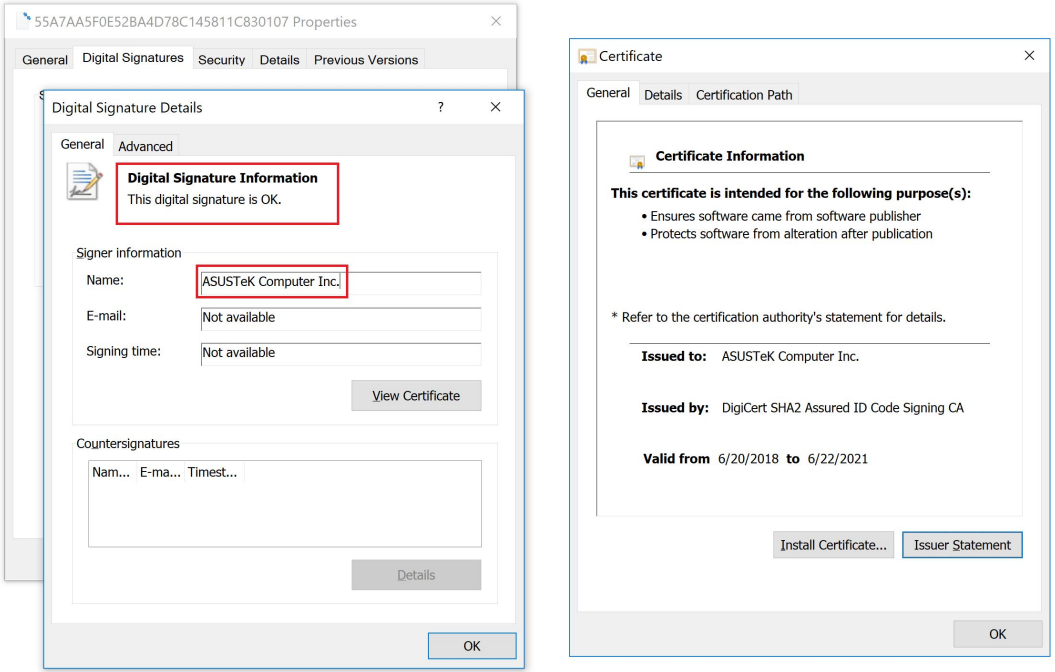
AMI UsbRt vulnerability is a perfect example of supply chain complexity



! Any device that uses the AMI UsbRt could be vulnerable to the very same bug

From reporting the firmware vulnerability to the fix usually takes around 6-9 months or more

Lack of transparency into supply chain



An Investigative Update of the Cyberattack

By Sudhakar Ramakrishna

May 7, 2021 | Security SolarFocus

Lack of security in the supply chain

Computer giant Acer hit by $50 million ransomware attack

By Lawrence Abrams

March 19, 2021 11:11 AM 0

Lack of security in the supply chain

> *"Acer routinely monitors its IT systems, and most cyberattacks are well defensed. Companies like us are constantly under attack, and we have reported recent abnormal situations observed to the relevant law enforcement and data protection authorities in multiple countries."*
>
> *"We have been continuously enhancing our cybersecurity infrastructure to protect business continuity and our information integrity. We urge all companies and organizations to adhere to cyber security disciplines and best practices, and be vigilant to any network activity abnormalities." - Acer.*

Lack of security in the supply chain

Gigabyte Breached by Ransomware Group AvosLocker – Data up for sale

By **Madeleine Hodson** in **Cyber Security News**

Published: **October 21, 2021**

Lack of transparency into supply chain

- Complex, intransparent firmware supply chain
  - Lots of 3rd party suppliers
  - Lack of widespread use of software BOM
- Extremely long disclosure timelines
  - From months to a year
- Security vs. regular support
  - No SLA: no updates
  - Limited ability to notify end users
- Needs outside effort to fix the update problem
  - LVFS finally creates concise story for non-Windows platforms

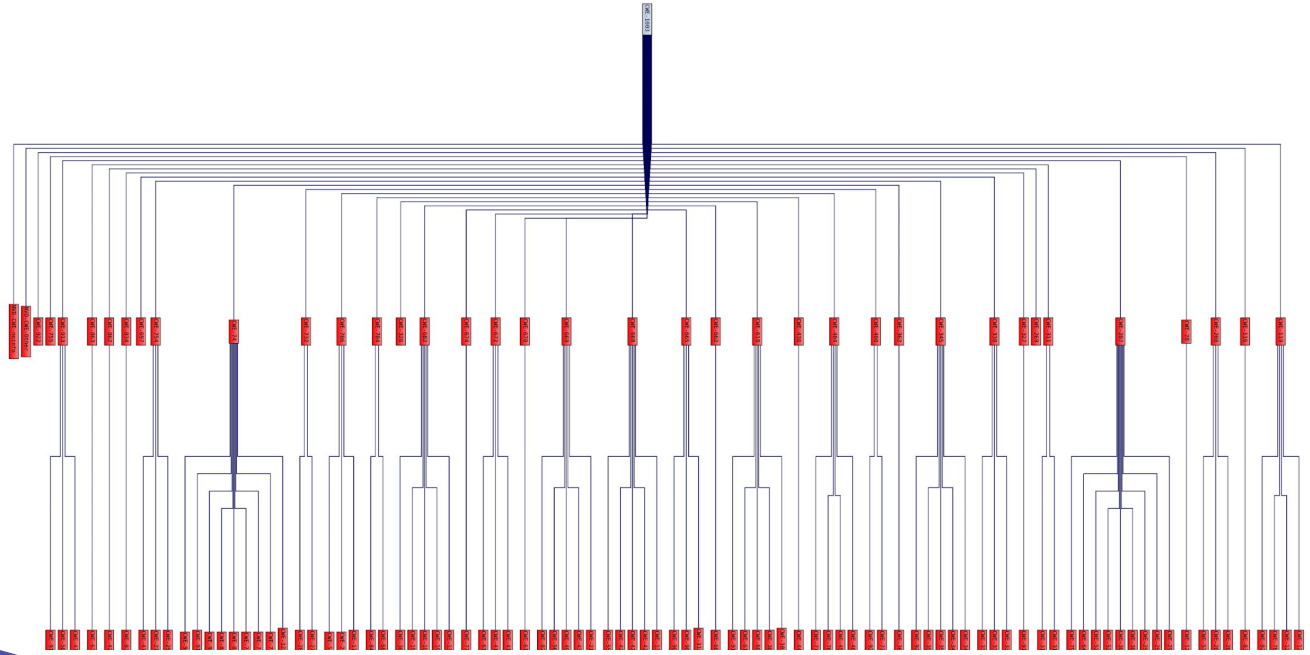
What are vulnerability classifications

- Enumeration of (common) vulnerabilities
  - Common Weakness Enumeration (CWE)
  - MITRE ATT&CK Framework
- Shared vocabulary
  - Exchange IOC, techniques
- Knowledge base
  - Red Teaming
  - Defence assessment
- Systematization
  - Research

What are vulnerability classifications

OPEN SOURCE FIRMWARE CONFERENCE

What are vulnerability classifications



Why Another Spec?

- CWE
  - CWE-1236: Improper Neutralization of Formula Elements in a CSV File
  - CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
  - CWE-1310: Missing Ability to Patch ROM Code

- MITRE ATT&CK:
  - T1495: Firmware Corruption
  - T1542: Pre-OS Boot

Why Another Spec?

- CWE
  - CWE-1236: Improper Neutralization of Formula Elements in a CSV File
  - CWE-79: Improper Neutralization of Input During Web Page

Modify System Image	Some vendors of embedded network devices provide cryptographic signing to ensure the integrity of operating system images at boot time. Implement where available, following vendor guidelines. [1]

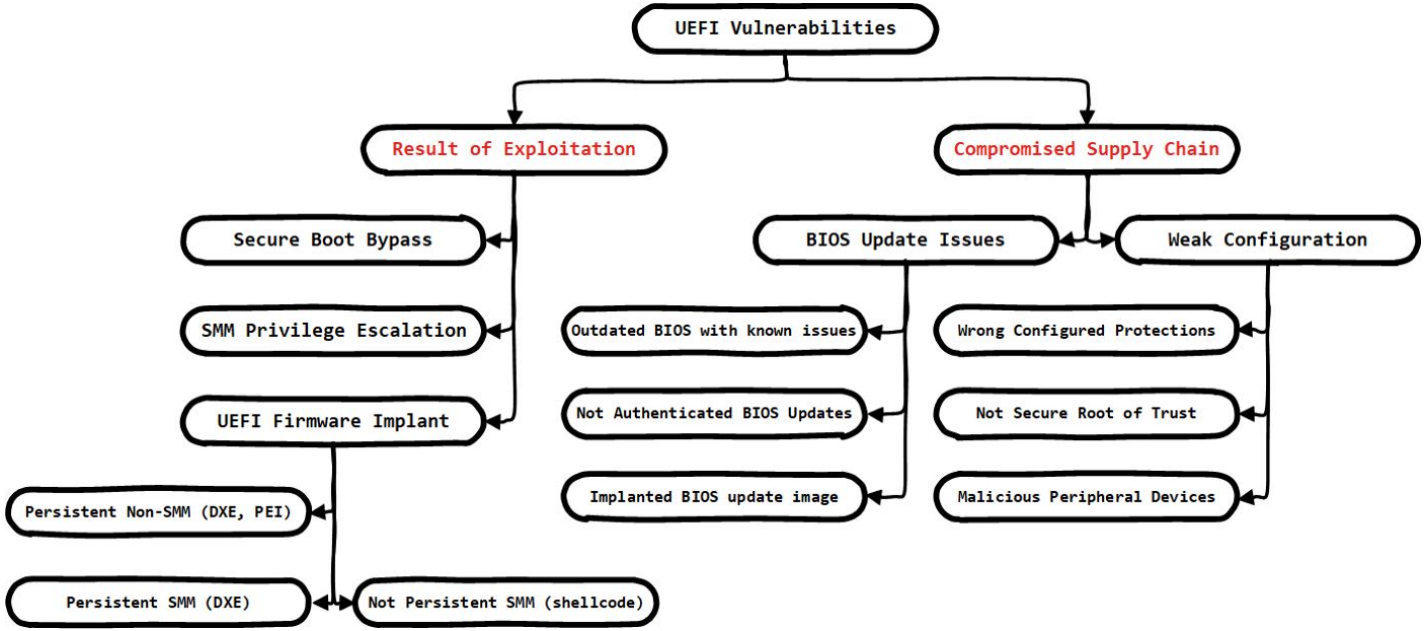
- MITRE ATT&CK:
  - T1495: Firmware Corruption
  - T1542: Pre-OS Boot
-

Why Another Spec?

- CWE
  - CWE-1236: Improper Neutralization of Formula Elements in a CSV File
  - CWE-79: Improper Neutralization of Input During Web Page
- T1495: Firmware Corruption
- T1542: Pre-OS Boot

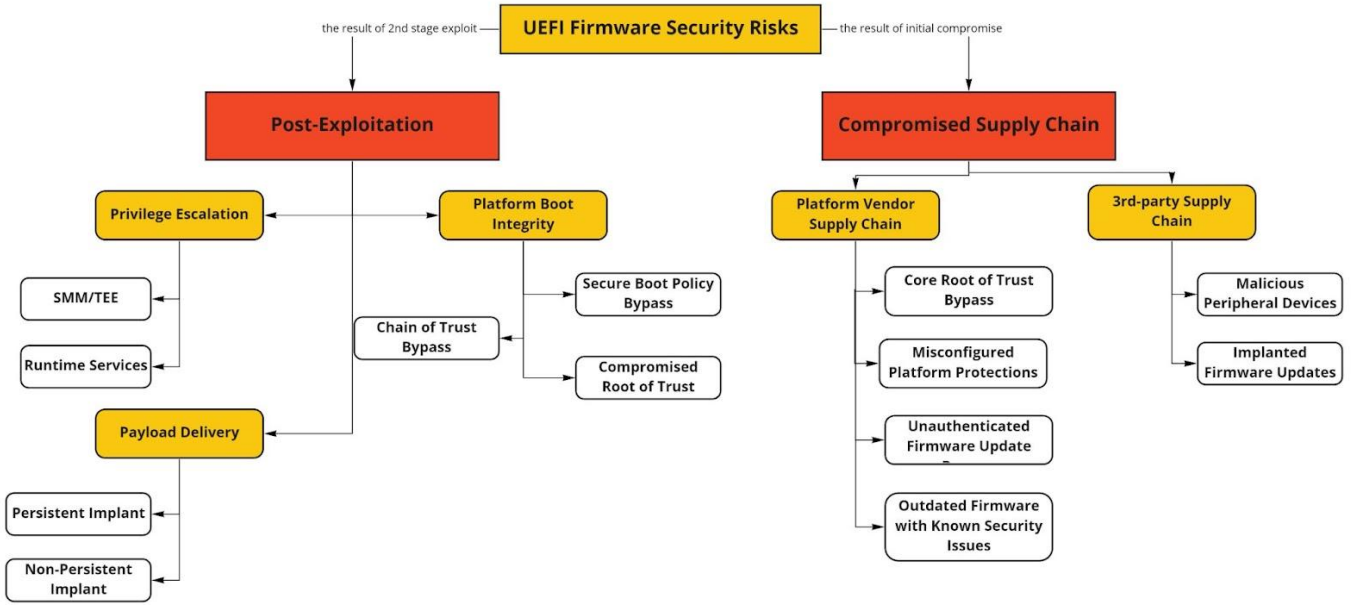
Local (L)	A vulnerability exploitable with only *local access* requires the attacker to have either physical access to the vulnerable system or a local (shell) account. Examples of locally exploitable vulnerabilities are peripheral attacks such as Firewire/USB DMA attacks, and local privilege escalations (e.g., sudo).
Adjacent Network (A)	A vulnerability exploitable with *adjacent network access* requires the attacker to have access to either the broadcast or collision domain of the vulnerable software. Examples of local networks include local IP subnet, Bluetooth, IEEE 802.11, and local Ethernet segment.
Network (N)	A vulnerability exploitable with *network access* means the vulnerable software is bound to the network stack and the attacker does not require local network access or local access. Such a vulnerability is often termed "remotely exploitable". An example of a network attack is an RPC buffer overflow.

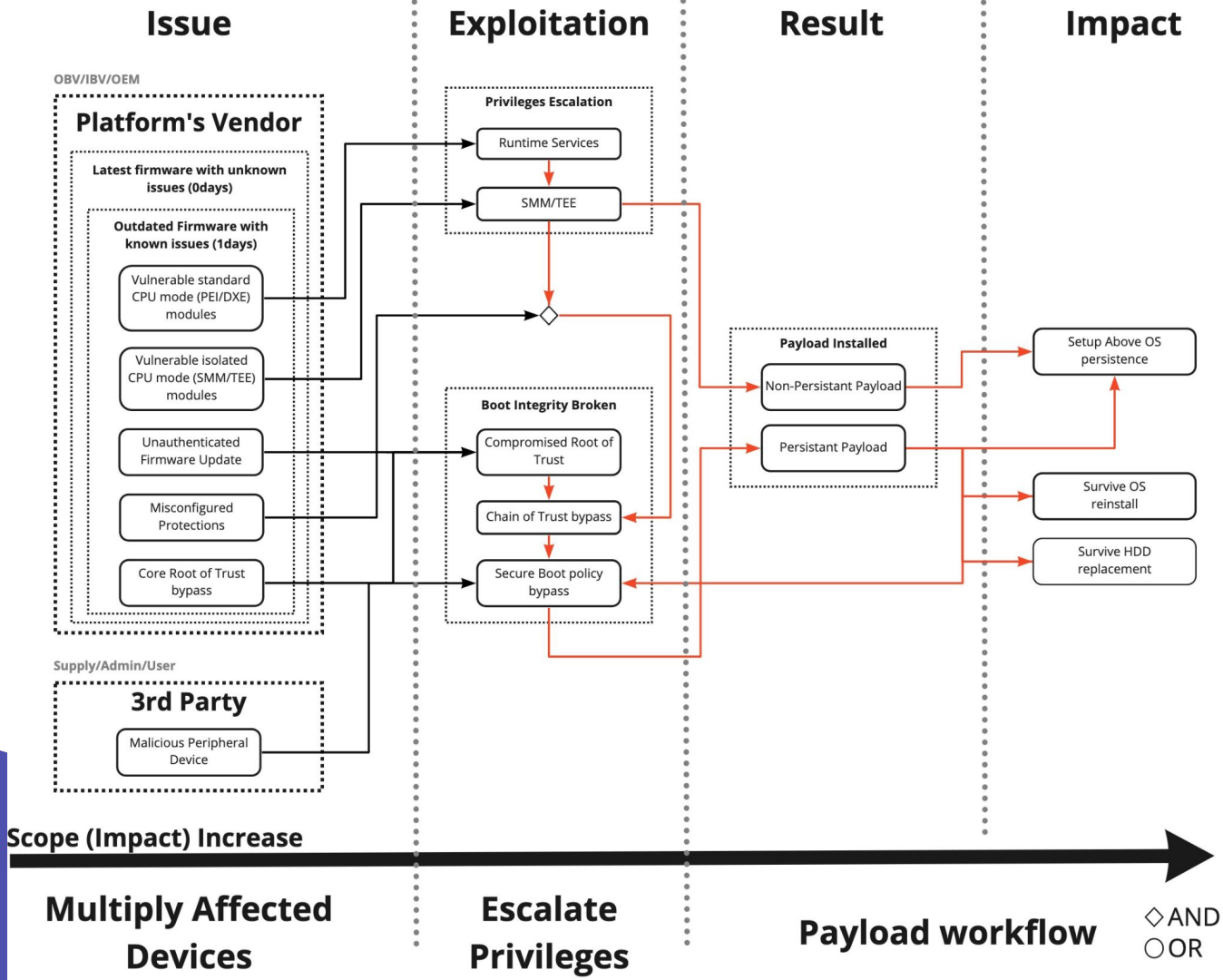
Firmware Vulnerabilities Classification: How It Started



https://medium.com/firmware-threat-hunting/uefi-vulnerabilities-classification-4897596e60af

Firmware Vulnerabilities Classification: Based on Impact

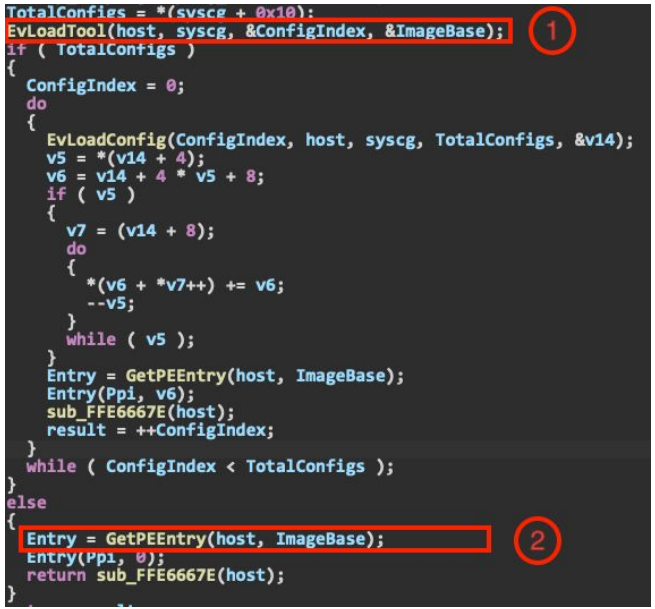




Disclosing and getting fixed the vulnerability is one side of the problem.

Another it's deliver this patch at scale to many systems in the field.

Intel BSSA DFT case study



- These hidden features of Intel BSSA were designed to run arbitrary unsigned code blobs stored in EFI variables.
- To make matters worse, Intel BSSA DFT is a part of the reference code.
- Intel BSSA DFT was intended to be used for debugging or testing purposes only.

https://www.binarly.io/posts/Attacking_(pre)EFI_Ecosystem

Vendor Name	Vendor Advisory	CVE	URL
Intel	INTEL-SA-00525	CVE-2021-0144	https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00525.html
Dell	DSA-2021-146	CVE-2021-0144	https://www.dell.com/support/kbdoc/en-us/000189473/dsa-2021-146-dell-client-platform-security-update-for-intel-bssa-vulnerability
Nvidia	NV-5213	CVE-2021-0144	https://nvidia.custhelp.com/app/answers/detail/a_id/5213
Lenovo	LEN-61893	CVE-2021-0144	https://support.lenovo.com/eg/en/product_security/ps500424-intel-bssa-dft-advisory
HP	HPSBHF03736	CVE-2021-0144	https://support.hp.com/za-en/document/ish_4168405-4168434-16/hpsbhf03736
HPE	HPESBHF04171	CVE-2021-0144	https://support.hpe.com/hpesc/public/docDisplay?docLocale=en_US&docId=hpesbhf04171en_us
Supermicro	no vendor ID	CVE-2021-0144	https://www.supermicro.com/en/support/security_Intel-SA-00525
F5	K08593253	CVE-2021-0144	https://support.f5.com/csp/article/K08593253

Binarly FwHunt vs Intel BSSA DFT

```
UncoreInitPeimRefCode:
  meta:
    name: INTEL-SA-00525 (CVE-2021-0144)
    namespace: Intel BSSA DFT detection tool
    url: https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00525.html
  guids:
    - 1:
      - name: EFI_PLATFORM_INFO_GUID
      - value: 1E2ACC41-E26A-483D-AFC7A056C34E087B
    - 2:
      - name: EFI_STATUS_CODE_SPECIFIC_DATA_GUID
      - value: 335984BD-E805-409A-B8F8D27ECE5FF7A6
    - 3:
      - name: EFI_STATUS_CODE_DATA_TYPE_DEBUG_GUID
      - value: 9A4E9246-D553-11D5-87E200062945C3B9
    - 4:
      - name: EFI_PEI_READ_ONLY_VARIABLE2_PPI_GUID
      - value: 2AB86EF5-ECB5-4134-B5563854CA1FE1B4
    - 5:
      - name: EFI_PEI_PCD_PPI_GUID
      - value: 01F34D25-4DE2-23AD-3FF336353FF323F1
  ppi:
    - 1:
      - name: EFI_PEI_READ_ONLY_VARIABLE2_PPI_GUID
      - value: 2AB86EF5-ECB5-4134-B5563854CA1FE1B4
      - service:
        - name: LocatePpi
  wide_strings:
    1: syscg
    2: toolh
  hex_strings:
    1: 56e8........593c01....80be....000000
      # 56                                    push    esi
      # E8 .. .. .. ..                       call    x_BiosSsaEnabled
      # 59                                    pop     ecx
      # 3C 01                                 cmp     al, 1
      # .. ..                                 jnz     short loc_FFDE86FD
      # 80 BE .. .. 00 00 00                  cmp     byte ptr [esi+81h], 0
      # .. ..                                 jz      short loc_FFDE86FD
    2: 6a006a0268be00000056e8
      # 6A 00                                 push    0
      # 6A 02                                 push    2
      # 68 BE 00 00 00                        push    0BEh
      # 56                                    push    esi
      # E8 .. .. .. ..                       call    SsaApi
```

```
                        ; const EFI_GUID gSsaBiosVariablesGuid
E8 FF EE 43 78 A9 DC 41+gSsaBiosVariablesGuid dd 43EEFFE8h              ; Data1
9D B6 54 C4 27 F2 7E 2A                                                 ; DATA XREF: EvLoadTool+65↓o
                                                                        ; EvLoadConfig+74↓o
                                                                        ; ExecuteTargetOnlyCmd+74↓o
                                                                        ; SsaBiosLoadStatus+83↓o
                                        dw 0A978h                       ; Data2
                                        dw 41DCh                        ; Data3
                                        db 9Dh, 0B6h, 54h, 0C4h, 27h, 0F2h, 7Eh, 2Ah; Data4
                        ; int dword_FFDEC918[4]
```

```
(*PeiServices)->LocatePpi(PeiServices, &gReadOnlyVariable2Guid, 0, 0, &ReadOnlyPpi);
ZeroMem(syscg_stack, 2048);
ReadOnlyPpi->GetVariable(ReadOnlyPpi, L"syscg", &gSsaBiosVariablesGuid, 0, &DataSize, syscg_stack);
syscg = AllocatePool(DataSize);
memcpy_0(syscg, syscg_stack, DataSize);
TotalConfigs = *(syscg + 0x10);
EvLoadTool(host, syscg, &ConfigIndex, &ImageBase);
```

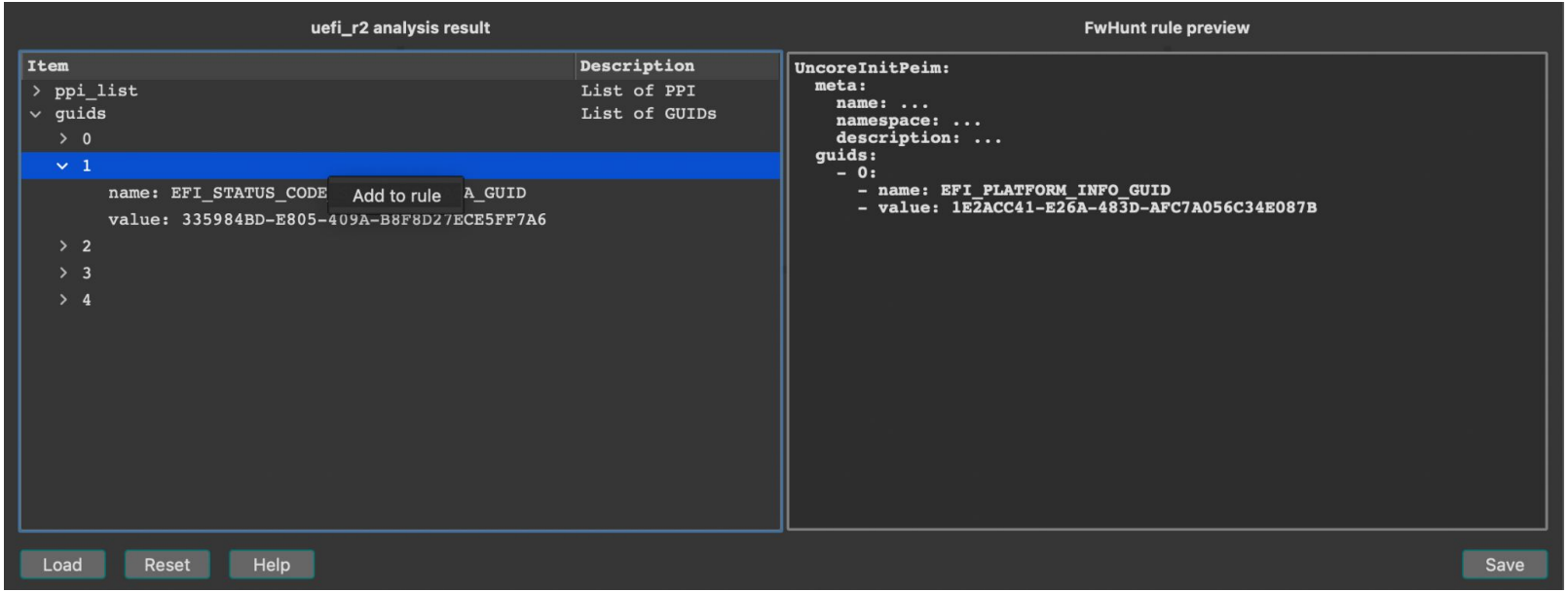
YARA rules that are NOT tailored to effectively cover UEFI firmware code specifics

The Firmware Hunt (FwHunt) rule format was designed to scan for known vulnerabilities and verify that an affected OEM vendor has patched the issue in its latest update.

Binarly FwHunt vs Intel BSSA DFT

https://www.binarly.io/posts/Detecting_Firmware_vulnerabilities_at_scale_Intel_BSSA_DFT_case_study

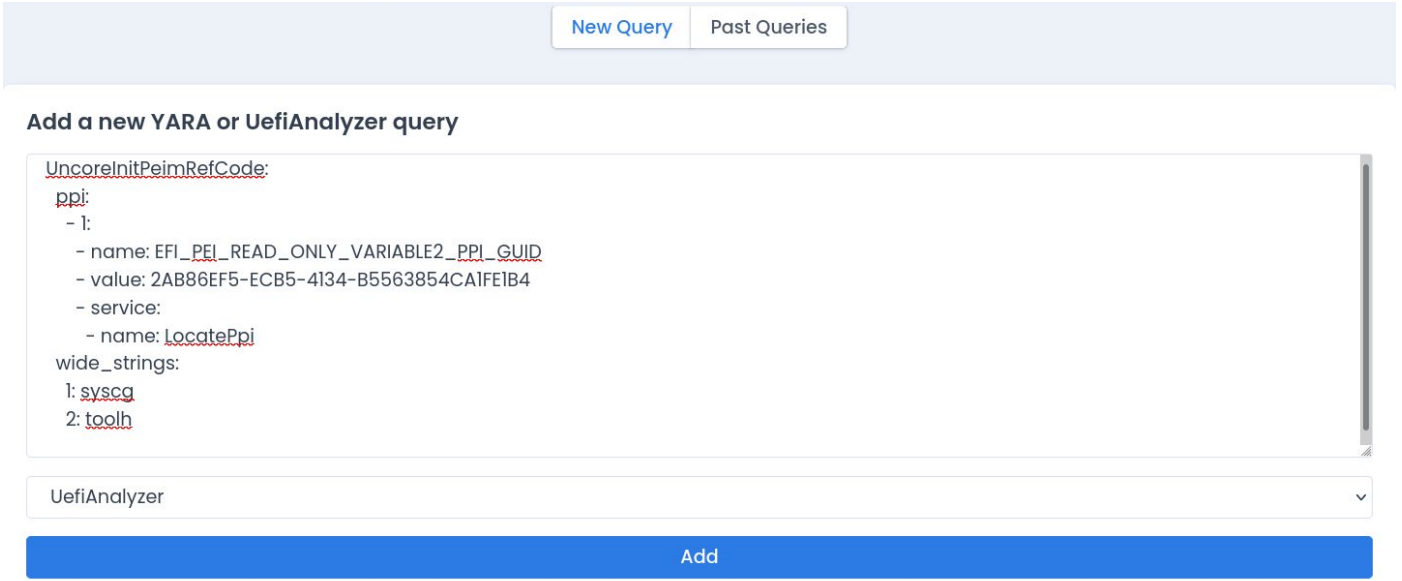
UEFI Firmware IOC hunt



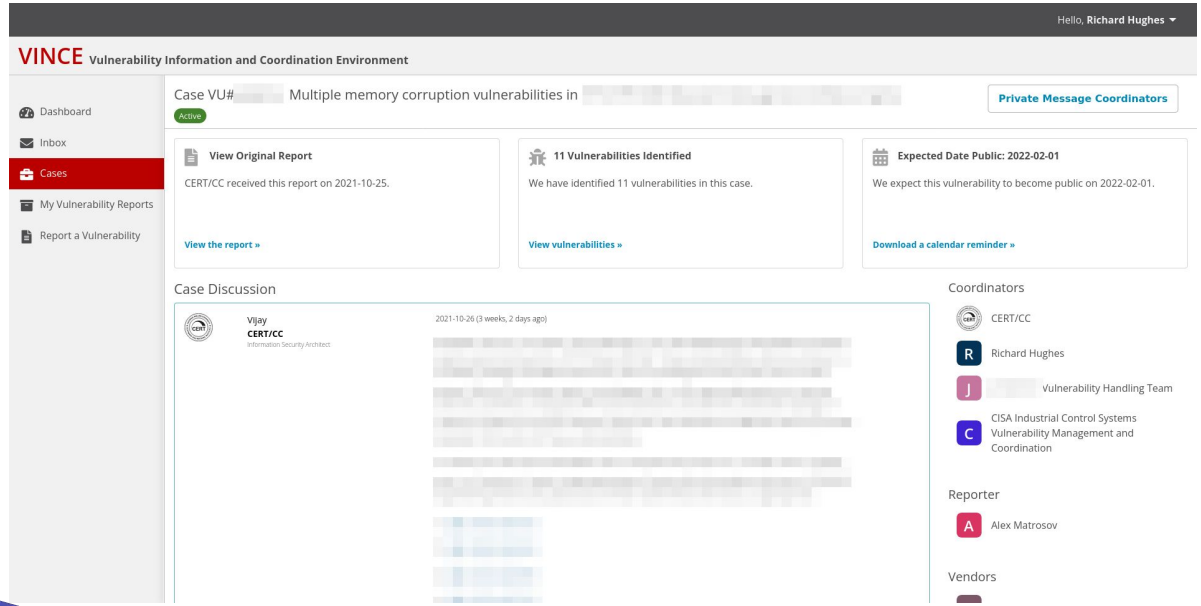
https://github.com/binarly-io/fwhunt-ida
https://www.youtube.com/watch?v=V0-Ie7z_ojE

OPEN SOURCE FIRMWARE CONFERENCE

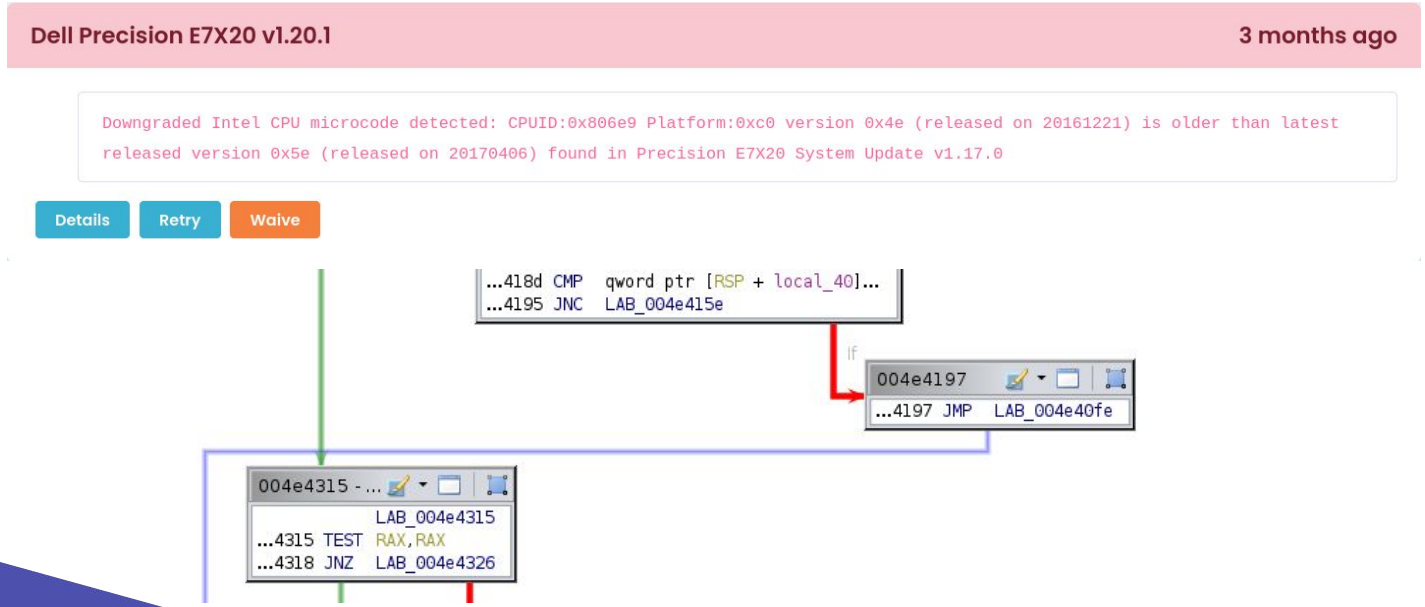
LVFS Yara and UEFI R2 scanning



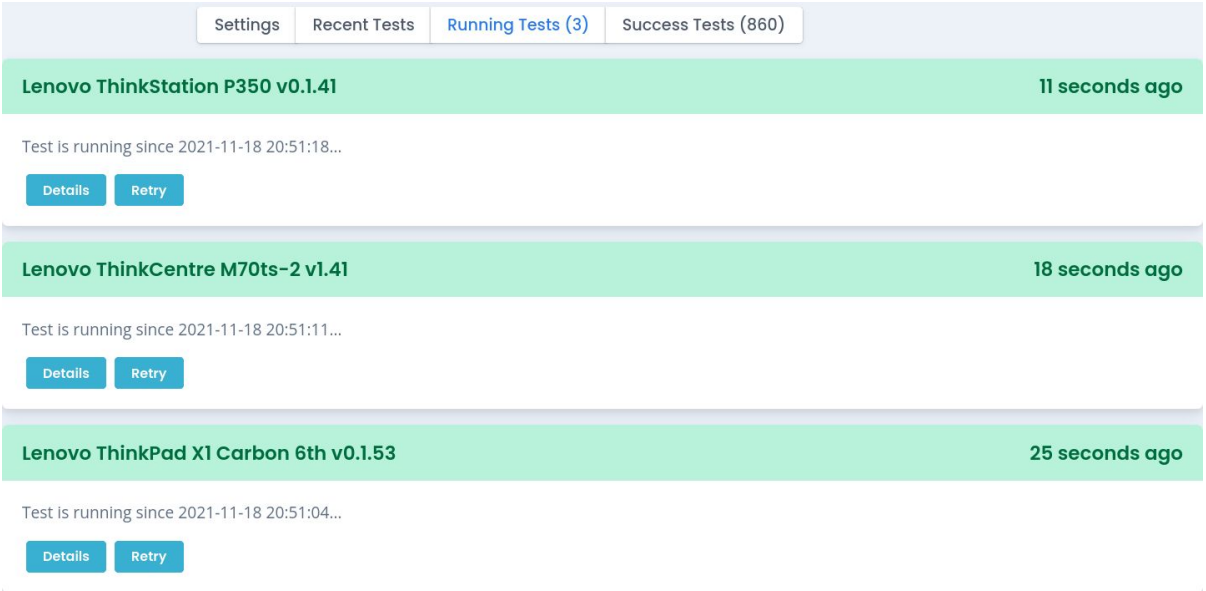
Use LVFS to find affected vendors & devices



Detect µcode downgrade, and **CVEs too?**



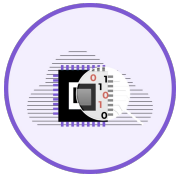
The LVFS run all rules on all firmware

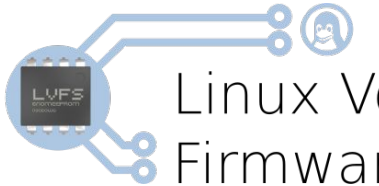


Open Source Tooling

- **Converged Security Suite**
  https://github.com/9elements/converged-security-suite
- **fwhunt and uefi_r2**
  https://github.com/binarly-io/uefi_r2
- **efiXplorer**
  https://github.com/binarly-io/efiXplorer
- **LVFS (Linux Vendor Firmware Service)**
  https://fwupd.org

CONVERGED SECURITY SUITE





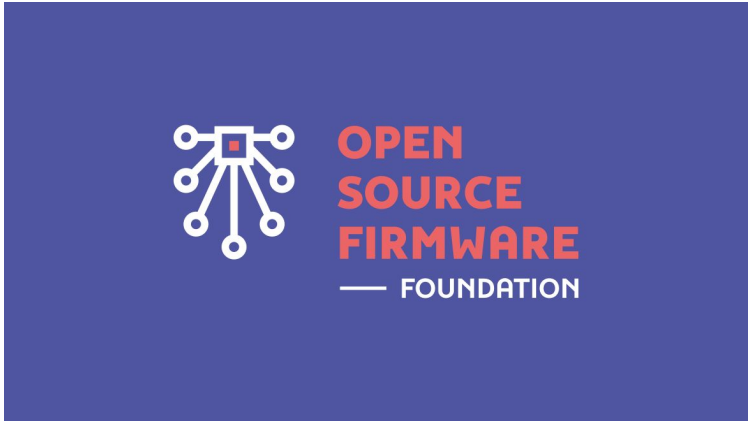
Linux Vendor Firmware Service

OPEN SOURCE FIRMWARE CONFERENCE

How to get involved?

- Firmware Security Workstream
  - Specification and Guidelines
  - Incident Response and Disclosure recommendations
  - Supporting the vendors
    - Open Source Tooling
    - Documentation

[Join us now]



opensourcefirmware.foundation

OPEN SOURCE FIRMWARE CONFERENCE