Fiedka the Firmware Editor
Advancing from CLIs to GUls

Daniel Maslowski
Agenda

- Introduction and Scope
- Current State
- Roadmap and Contributing
Introduction and Scope
Hello, I am Daniel :-) 

Work and education
- IT security and computer science
- software engineer
- infrastructure and web
- applications and UI

Open Source contributions
- hardware and firmware
- operating systems
- software distributions
- reverse engineering
Hello, I am Fiedka :-)  

Graphical Firmware Editor  
- UEFI/PSP/ME Filesystem Exploration  
- Flash Usage Visualization  
- TPM Event Log  
- Platform State Investigation  
- Secure Boot Key Management
Firmware

Firmware is software that is hard to get to. - Bryan Cantrill
UX/UI design and hardware design/manufacturing are both complex and still learning about agile and open processes.
UX/UI design and hardware design/manufacturing are both complex and still learning about agile and open processes.

We need more tools and feedback.
UX/UI design and hardware design/manufacturing are both complex and still learning about agile and open processes.

We need more tools and feedback.

- For education, to speed up understanding
- For research, to gain security through transparency
- For development, to support sustainability
Full firmware images differ depending on vendor, platform, OEM, and other factors.
Full firmware images differ depending on vendor, platform, OEM, and other factors.

Building tools is a huge investment.
Problem Statement

Full firmware images differ depending on vendor, platform, OEM, and other factors.

Building tools is a huge investment.

- Intel x86
  - IFD, ME, UEFI, …
- AMD x86
  - PSP, SMU, UEFI, …
- other platforms
  - ARM, RISC-V, Xtensa, DSPs, …
Other Tools

- AMI’s Module Management Tool (MMTool)
- RED BIOS EDITOR
- Intel FMMT
- UEFITool
- ifdtool
- uefi-firmware-parser
- PSPTool
- MFT (Mimoja’s Firmware Toolkit)
- MEAnalyzer
- mecleaner
- Fiano
- tpmtool
- Converged Security Suite
- efiXplorer
- ...

...
### Trees and Tables

<table>
<thead>
<tr>
<th>Name</th>
<th>Action</th>
<th>Type</th>
<th>Subtype</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>UEFI Image</td>
<td>Padding</td>
<td>Image</td>
<td>UEFI</td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td>Padding</td>
<td>Padding</td>
<td>Non-empty</td>
<td></td>
</tr>
<tr>
<td>1F0E069EC-6F0A-4B85-801E-9E218</td>
<td>Volume</td>
<td>Empty</td>
<td>(0x0FF)</td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td>Padding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NVMe store</td>
<td>Padding</td>
<td>NVMe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDC store</td>
<td>Padding</td>
<td>FDC</td>
<td>Empty</td>
<td>(0x0FF)</td>
</tr>
<tr>
<td>- EFISystemNVDataFvGuid</td>
<td>Volume</td>
<td>NVRAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- VSS store</td>
<td>VSS</td>
<td>Entry</td>
<td>Auth</td>
<td>PK</td>
</tr>
<tr>
<td>- EFIglobalVariableGuid</td>
<td>VSS</td>
<td>Entry</td>
<td>Auth</td>
<td>KEK</td>
</tr>
<tr>
<td>- EFIImageSecurityDatabase.</td>
<td>VSS</td>
<td>Entry</td>
<td>Auth</td>
<td>dbx</td>
</tr>
<tr>
<td>Free space</td>
<td>Free</td>
<td>space</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td>Padding</td>
<td>Padding</td>
<td>Non-empty</td>
<td></td>
</tr>
<tr>
<td>Intel microcode</td>
<td>Microcode</td>
<td>Intel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel microcode</td>
<td>Microcode</td>
<td>Intel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td>Padding</td>
<td>Padding</td>
<td>Non-empty</td>
<td></td>
</tr>
<tr>
<td>2F34A86B-3F2C-47EB-A9C3-B7A3</td>
<td>Volume</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td>Padding</td>
<td>Padding</td>
<td>Non-empty</td>
<td></td>
</tr>
</tbody>
</table>

**Found volume magic at 0x6a7000**

**Firmware Volume:** 8c8ce578-8a3d-4f1c-9935-896185c32dd3 attr 0x0004fe

**Firmware Volume Blocks:** (1617, 0x1000)

**File 0:** ffffffff-fffffff-fffffff-ffffffffffffff type 0x0f, attr 0x0000

**File 1:** 414d94ad-998d-47d2-bfcf-4e8b2241de32 type 0x0e, attr 0x00

**Section 0:** type 0x18, size 0x1014 (416 bytes) (Free-form GUID)

**File 2:** 05ca020b-07c1-1ad-c-9011-00173153e8b0 type 0x01, attr 0x00

**Firmware Volume:** 8c8ce578-8a3d-4f1c-9935-896185c32dd3 attr 0x0004fe

**Firmware Volume Blocks:** (32, 0x1000)

**File 0:** ffffffff-fffffff-fffffff-ffffffffffffff type 0x0f, attr 0x0000

**File 1:** 7bb2b999-61bb-1ld-9ad-0090273fc14d type 0x02, attr 0x0000

---

**ME Analyzer v1.160.0 r212**

<table>
<thead>
<tr>
<th>1.rom (1/1)</th>
<th>Family</th>
<th>ME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Family</td>
<td>ME</td>
</tr>
<tr>
<td>Version</td>
<td>4.1.3.1038</td>
<td></td>
</tr>
<tr>
<td>Release</td>
<td>Production</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Extracted</td>
<td></td>
</tr>
<tr>
<td>SKU</td>
<td>AMT</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>2008-12-17</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>0x1FB080</td>
<td></td>
</tr>
<tr>
<td>Chipset Support</td>
<td>ICH9M</td>
<td></td>
</tr>
<tr>
<td>Latest</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
General Concepts

The hard part
- phases/stages, payloads, chains
- (non-)volatility, static vs dynamic, persistency, reset…
General Concepts

The hard part
- phases/stages, payloads, chains
- (non-)volatility, static vs dynamic, persistency, reset...

The easy part
- storage, files \(\Rightarrow\) file systems
- metadata, properties, inputs to boot flow (much volatility!)
Current State
How it started

inspired by Ryan’s `iutk2`
render flash usage from `fmap` fixtures
`utk-web, rendering fixtures`

color palette suitable for color-blind people
blocks display usage: free, used, all-zero
How it’s going

Electron app with Go-based WASM back-end and file picker
UI components: Directory, Entry, Fmap, navigation, search
Storybook UI development environment
**Integrations**

- UEFI: parsing via Fiano’s utk pkg
- PSP: experimental branch using Converged Security Suite
- TPM: components to render a log (basics)

> fixtures from tpmtool, PRs open

<table>
<thead>
<tr>
<th>Event</th>
<th>Digests</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Disk Guid: 00000000-C5D3-90AD-A500-0044A280F096&quot;</td>
<td>SHA1: 1fd1961b281c7adde317e53663b2479d6a4f9d, SHA256: 8d3bc65776f705e545a5ed943687854c0d87f9cb80c5ed57e43eb9f4b4f6d</td>
</tr>
</tbody>
</table>

**EFI boot services app**

<table>
<thead>
<tr>
<th>Event</th>
<th>Digests</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Image loaded at address 0x3123249176 with 93854b&quot;</td>
<td>SHA1: 09f1833bbf1f9bbf9336f76eb2188b3c19c4fd, SHA256: 7c18614e3e3cfe5a3b6c2097eae494e8092c65f8d610a7459fasn8526/2554cc3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event</th>
<th>Digests</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Image loaded at address 0x3693762002 with 9665120b&quot;</td>
<td>SHA1: 2e9e67daebf818a9b6f259062e2b1b79b714b6c, SHA256: 7ace7c5433f6144e976a244de37241b1b6e46b67685762382f2b9683ee3b0ef9</td>
</tr>
</tbody>
</table>
Demo

Fiedka - analyze a firmware image

Select file
Roadmap and Contributing
Next Milestones

1. Integrate PSP, IFD/ME, CBFS parsing - back-end work required
2. Edit/save; MVP: delete DXE in OVMF - uǐk already has that :)
Ideas

Emulation

- dxelate, peimulate: wrap a DXE/PEIM in a monitored OVMF
- on click of a DXE/PEIM, create OVMF image and run QEMU
- likewise, leverage PSPEmu
- UEFI runtime service analysis (SMM?)
Ideas

Emulation
- dxelate, peimulate: wrap a DXE/PEIM in a monitored OVMF
- on click of a DXE/PEIM, create OVMF image and run QEMU
- likewise, leverage PSPEmu
- UEFI runtime service analysis (SMM?)

Bootable ISO
- create an image with Linux, Fiedka, u-root, CLI tools etc
- the firmware development counterpart to Kali
- possibly extend SystemRescueCD
System Integration

EFI variables and Secure Boot
- look into rhboot/efivar and canonical/go-efilib
- integrate with go-uefi

TPM / Integrity
- tpmtool
- go-attestation
- immune Guard agent
- health report
- tpm-js for simulation and debugging
System Integration

EFI variables and Secure Boot
- look into rhboot/efivar and canonical/go-efilib
- integrate with go-uefi

TPM / Integrity
- log display; tpmtool, go-attestation
- immune Guard agent health report
- tpm-js for simulation and debugging
Visualizations

- DMI, SMBIOS, etc
- ACPI tables, Device Tree
- IFR (Intermediate Forms Representation) - firmware menu
- platform and memory setup (page tables, interrupts, MSRs)
- integrate CPUID visualiser
Visualization API

Transforms

```javascript
export const flattenVolumes = (volumes) => {
    volumes.reduce((acc, curr) => {
        if (curr.Value.Files) {
            curr.Value.Files.forEach((f) => {
                if (f.Type === FILE_TYPE_FV_IMAGE) {
                    getFvsFromFile(f).forEach((fv) => {
                        acc.push({ parent: f, ...fv });
                    });
                }
            });
        }
        acc.push(curr);
    }, []);
    return acc;
};
```
Other ideas

Do you have any ideas? Please submit! :-)

https://github.com/fiedka/fiedka/issues
Thanks!
Links

Project Website
https://fiedka.app/

Slides

Previous Talk
Introducing utk-web - a web developer's view on firmware