






Fiedka the Firmware Editor

Advancing from CLIs to GUIs

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*



Agile and Open

UX/UI design and hardware design/manufacturing are both complex and still learning about agile and open processes.



Agile and Open

UX/UI design and hardware design/manufacturing are both complex and still learning about agile and open processes.

We need more tools and feedback.



Agile and Open

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



Problem Statement

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



Problem Statement

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

Name	Action	Type	Subtype	Text
- UEFI Image		Image	UEFI	
Padding		Padding	Non-empty	
!FD0BACE-0F0A-40B5-901E-F6218...		Volume	FFSV2	
- EfiSystemNvDataFvGuid		Volume	NVRAM	
Padding		Padding	Non-empty	
FTW store		FTW store		
Padding		Padding	Empty (0xFF)	
- FDC store		FDC store		
- EfiSystemNvDataFvGuid		Volume	NVRAM	
- VSS store		VSS store		
EfiGlobalVariableGuid		VSS entry	Auth	PK
EfiGlobalVariableGuid		VSS entry	Auth	KEK
EfiImageSecurityDatabase..		VSS entry	Auth	db
EfiImageSecurityDatabase..		VSS entry	Auth	dbx
Free space		Free space		
Padding		Padding	Non-empty	
Intel microcode		Microcode	Intel	
Intel microcode		Microcode	Intel	
Padding		Padding	Non-empty	
!CF1406C5-3FEC-47EB-A6C3-B71A3...		Volume	FFSV2	
Padding		Padding	Non-empty	

```

Found volume magic at 0x6af000
Firmware Volume: 8c8ce578-8a3d-4f1c-9935-896185c32dd3 attr 0x0004fe
Firmware Volume Blocks: (1617, 0x1000)
File 0: ffffffff-ffff-ffff-ffff-ffffffffffff type 0xf0, attr 0x00
File 1: 414d94ad-998d-47d2-bfcd-4e882241de32 type 0x02, attr 0x00
  Section 0: type 0x18, size 0x1014 (4116 bytes) (Free-form GUID)
File 2: 05ca020b-0fc1-11dc-9011-00173153eba8 type 0x01, attr 0x04
Firmware Volume: 8c8ce578-8a3d-4f1c-9935-896185c32dd3 attr 0x00
Firmware Volume Blocks: (32, 0x1000)
File 0: ffffffff-ffff-ffff-ffff-ffffffffffff type 0xf0, attr
File 1: 7bb28b99-61bb-11d5-9a5d-0090273f14d type 0x02, attr
  
```

ME Analyzer v1.160.0 r212	
1.rom (1/1)	
Family	ME
Version	4.1.3.1038
Release	Production
Type	Extracted
SKU	AMT
Date	2008-12-17
Size	0x1FB000
Chipset Support	ICH9M
Latest	No



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 => **file systems**






metadata, properties, inputs to boot flow (much volatility!)

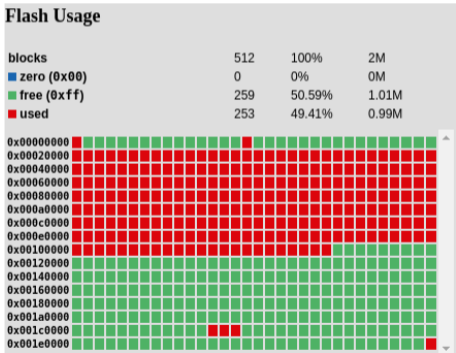




Current State



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



Demo



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

Fiedka - analyze a firmware image

Select file

UEFI
A52MIX_1.31 FA49 51C8 E527 03FE 5C60 4F1C 61C0 GUID or name substring

undefined bytes, 1 files
FA4974FC-AF1D-4E5D-BDC5-DACD6D27BAEC

undefined bytes, 6 files
51C81E99-DF1A-4DC5-9966-0B8D9AA09ACD (84366C9E-CD61-4B6E-8BC7-384541382842)

AndLegacyStorage LegacyRegion
CmsData CmsData
CmsVideo AM79EC-ABF1-4096-8E76-8E13847D993

undefined bytes, 20 files
E5274881-D739-4209-9882-7C564B19E263 (C0492C7F-75E2-487C-B2DD-000E6CF7B905)

Signature
path: A3F8E6A-1127-88F9-8E7C-6C8089FF8D
type: EFI_FV_FILETYPE_DRIVER
size:
blocks used: none

Signature
path: 0298FC7-6548-8888-624D-485A134E4F6A
type: EFI_FV_FILETYPE_DRIVER
size:
blocks used: none

Signature
path: 1A7E4490-2F52-A150-983C-61205E87622B
type: EFI_FV_FILETYPE_DRIVER
size:
blocks used: none

Signature
path: 52803F90-E8E9-4E73-41E2-8DF644050113
type: EFI_FV_FILETYPE_DRIVER
size:
blocks used: none

DeepEx
> EFI_PCD_PROTOCOL_GUID
path: A23F879-228D-4F4D-A437-988982C8E8BA
type: EFI_FV_FILETYPE_DRIVER
size:
blocks used: none

UnidPublicData
path: 89262F3A-22DC-49C8-8007-1F9101F3E25A
type: EFI_FV_FILETYPE_DRIVER
size:
blocks used: none

Flash Usage

Blocks	4096	100%	18M
zero (0x00)	21	0.51%	0.08M
free (0xFF)	1706	43.0%	8.06M
used	2269	55.86%	8.04M



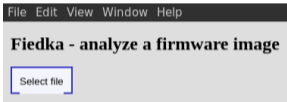
Integrations

- 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

EV_EFI_GPT_EVENT	
Event	"Disk Guid - 00000000-C5D3-98AD-A5D0-0844A28DF096"
Digests	SHA1: fdf06fbb281efabfe317ec56061b24798cda4c9d
	SHA256: 8d38c857cff8ff0b5e94abe94366f84dcddc7e9cbbd0c6d67e26e69fd4b4d4d
EFI boot services app	
Event	"Image loaded at address 0x3123249176 with 93054b"
Digests	SHA1: f19f8e338af1998b1933b5f7be28a80c1dc0c34fd
	SHA256: 7c1a8614c3e3d6a5b6c2897ea4e94c8892c658b6d16a7459fa038526f25f4cc3
EFI boot services app	
Event	"Image loaded at address 0x3093762072 with 9605120b"
Digests	SHA1: 2eb9a7daab81bbd9bfe258662e2b18b790714b6d
	SHA256: 7ae77c55430d6140e976a244de37241bbdbe46b67605f620df2b0683ee3b9ef9



Demo





Roadmap and Contributing







Next Milestones

1. integrate PSP, IFD/ME, CBFS parsing - back-end work required
2. edit/save; MVP: delete DXE in OVMF - utk already has that :)







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?)



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






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

```
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

<https://metaspora.org/fiedka-osfc2021.pdf>

Previous Talk

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

<https://metaspora.org/introducing-utk-web-rc3.pdf>

