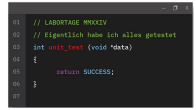
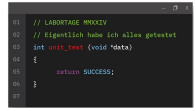
Look at ME in Rust!

Daniel Maslowski

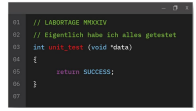


Agenda

Intel (CS)ME Platforms
Library and CLI
Fiedka the Firmware Editor



Intel (CS)ME Platforms

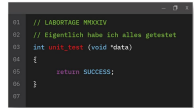


Knowledge

thanks to reverse engineering, presentations and tools

Alexander Tereshkin and Rafal Wojtczuk[1]
Igor Skochinsky[2]
Dmitry Sklyarov, Maxim Goryachy, Mark Ermolov[3]
Peter Bosch[4]
Plato Mavropoulos[5]

[1] https://invisiblethingslab.com/resources/bh09usa/Ring%20-3%20Rootkits.pdf
[2] https://github.com/skochinsky/papers
[3] https://github.com/ptresearch
[4] https://media.ccc.de/v/36c3-10694-intel_management_engine_deep_dive
[5] https://github.com/platomav/MEAnalyzer/



Knowledge

thanks to reverse engineering, presentations and tools

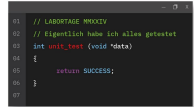
Alexander Tereshkin and Rafal Wojtczuk[1]
Igor Skochinsky[2]
Dmitry Sklyarov, Maxim Goryachy, Mark Ermolov[3]
Peter Bosch[4]
Plato Mavropoulos[5]

Variants

TXE - Trusted Execution Engine
(CS)ME - (Converged Security &)
Manageability Engine
SPS - Server Platform Services

---

[1] https://invisiblethingslab.com/resources/bh09usa/Ring%20-3%20Rootkits.pdf
[2] https://github.com/skochinsky/papers
[3] https://github.com/ptresearch
[4] https://media.ccc.de/v/36c3-10694-intel_management_engine_deep_dive
[5] https://github.com/platomav/MEAnalyzer/



Knowledge

thanks to reverse engineering, presentations and tools

Alexander Tereshkin and Rafal Wojtczuk[1]
Igor Skochinsky[2]
Dmitry Sklyarov, Maxim Goryachy, Mark Ermolov[3]
Peter Bosch[4]
Plato Mavropoulos[5]

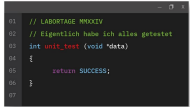
Variants

TXE - Trusted Execution Engine
(CS)ME - (Converged Security &) Manageability Engine
SPS - Server Platform Services

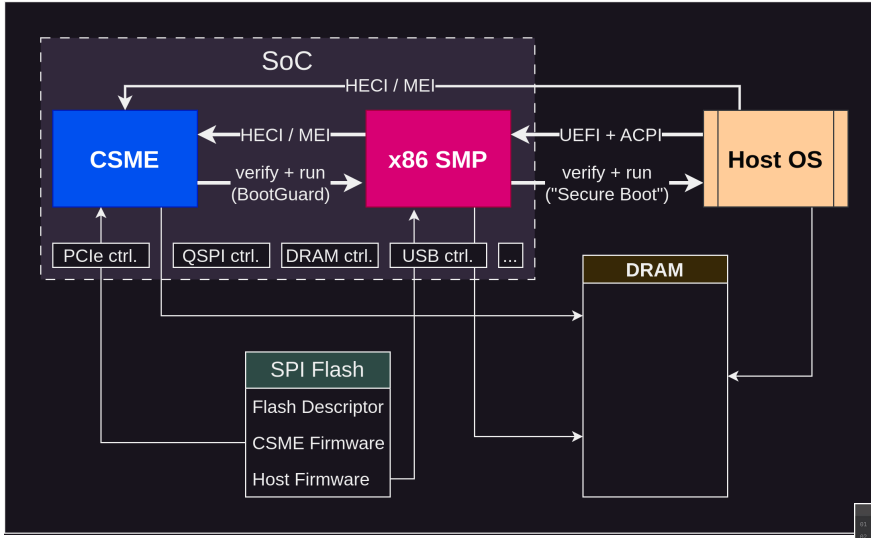
Generations / Versions

1st Gen: ME Ver 1-5, ARCTangent-A4
2nd Gen: ME Ver 6-10, ARC 600
3rd Gen: CSME Ver 11+, 486

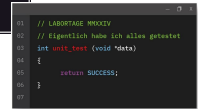
[1] https://invisiblethingslab.com/resources/bh09usa/Ring%20-3%20Rootkits.pdf
[2] https://github.com/skochinsky/papers
[3] https://github.com/ptresearch
[4] https://media.ccc.de/v/36c3-10694-intel_management_engine_deep_dive
[5] https://github.com/platomav/MEAnalyzer/



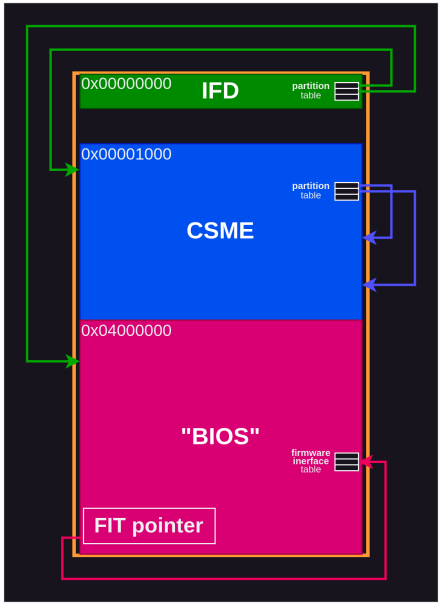
Hardware Platform and Flow[6]



[6] https://www.intel.com/content/dam/www/public/us/en/security-advisory/documents/intel-csme-security-white-paper.pdf



Flash Partitioning



Intel Flash Descriptor (IFD)

partitions whole flash

Flash Partition Table (FPT)

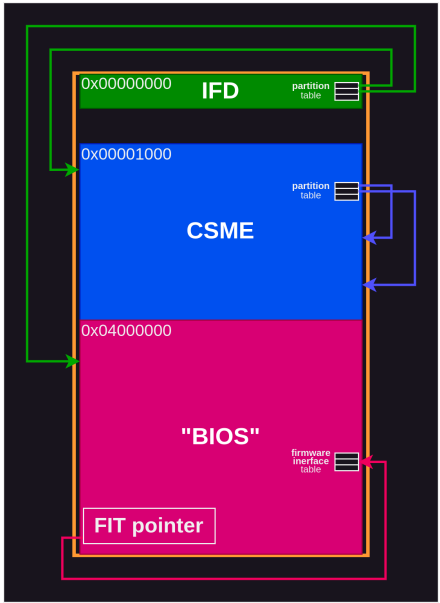
partitions ME firmware

Firmware Interface Table (FIT)

points to microcode updates, code modules, manifests and policies

```
01  // LABORTAGE MMXXIV
02  // Eigentlich habe ich alles getestet
03  int unit_test (void *data)
04  {
05      return SUCCESS;
06  }
07
```

Flash Partitioning



Intel Flash Descriptor (IFD)

partitions whole flash

Flash Partition Table (FPT)

partitions ME firmware

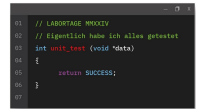
Firmware Interface Table (FIT)

points to microcode updates, code modules, manifests and policies

We are interested in FPT and FIT.

```
01
02  // LABORTAGE MMXXIV
03  // Eigentlich habe ich alles getestet
04  int unit_test (void *data)
05  {
06      return SUCCESS;
07  }
```

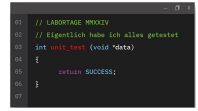
FPT (Flash Partition Table)



FPT (Flash Partition Table)

Data Partitions

lots of partitions; not investigated any further



FPT (Flash Partition Table)

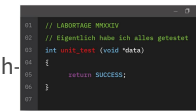
Data Partitions

lots of partitions; not investigated any further

Directories

flat layout
signature validated before execution
Gen 2: manifest first including entry count, then entries
Gen 3: Code Partition Directory (CPD), manifest in directory
Debug Launch Module (DLM): optionally run before other code

[7] https://www.blackhat.com/docs/eu-17/materials/eu-17-Sklyarov-Intel-ME-Flash-File-System-Explained.pdf



FPT (Flash Partition Table)

Data Partitions

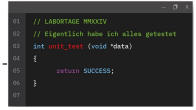
lots of partitions; not investigated any further

Directories

flat layout
signature validated before execution
Gen 2: manifest first including entry count, then entries
Gen 3: Code Partition Directory (CPD), manifest in directory
Debug Launch Module (DLM): optionally run before other code

Flash File System aka ME File System (MFS)[7]

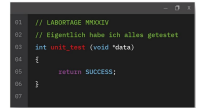
https://github.com/ptresearch/parseMFS
https://github.com/kakaroto/MFSUtil
is EFFS (partition) the same?



[7] https://www.blackhat.com/docs/eu-17/materials/eu-17-Sklyarov-Intel-ME-Flash-File-System-Explained.pdf

FIT (Firmware Interface Table)

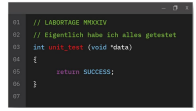
optional part of a full *secure boot* setup, for *OEMs*, not consumers



FIT (Firmware Interface Table)

optional part of a full *secure boot* setup, for *OEMs*, not consumers

The BIOS flash must include a Firmware Interface Table (FIT) with Type 0 (FIT Header) and Type 1 (Microcode Update) entries.
A microcode update must exist for every processor stepping supported by the platform.



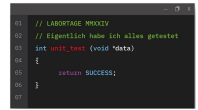
FIT (Firmware Interface Table)

optional part of a full *secure boot* setup, for *OEMs*, not consumers

The BIOS flash must include a Firmware Interface Table (FIT) with Type 0 (FIT Header) and Type 1 (Microcode Update) entries.
A microcode update must exist for every processor stepping supported by the platform.

Documentation

https://cdrdv2.intel.com/v1/dl/getContent/599500/view
https://doc.coreboot.org/soc/intel/fit.html
https://winraid.level1techs.com/t/fit-tool-csme-rebuild/33139



FIT (Firmware Interface Table)

optional part of a full *secure boot* setup, for *OEMs*, not consumers

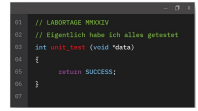
> *The BIOS flash must include a Firmware Interface Table (FIT) with Type 0 (FIT Header) and Type 1 (Microcode Update) entries.*
> *A microcode update must exist for every processor stepping supported by the platform.*

Documentation

https://cdrdv2.intel.com/v1/dl/getContent/599500/view
https://doc.coreboot.org/soc/intel/fit.html
https://winraid.level1techs.com/t/fit-tool-csme-rebuild/33139

Editing

Fiano? https://github.com/fiedka/fiedka/issues/64



FIT (Firmware Interface Table)

optional part of a full *secure boot* setup, for *OEMs*, not consumers

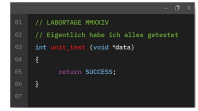
> *The BIOS flash must include a Firmware Interface Table (FIT) with Type 0 (FIT Header) and Type 1 (Microcode Update) entries.*
> *A microcode update must exist for every processor stepping supported by the platform.*

Documentation

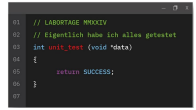
https://cdrdv2.intel.com/v1/dl/getContent/599500/view
https://doc.coreboot.org/soc/intel/fit.html
https://winraid.level1techs.com/t/fit-tool-csme-rebuild/33139

Editing

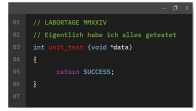
Fiano? https://github.com/fiedka/fiedka/issues/64
We have another implementation in Rust (WIP). :)



DEMO: FPT + FIT in CLI

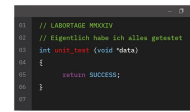


Library and CLI



Background

analysis problem: covering all cases is complex
depth first approach with narrow focus: Intel ME 11



Background

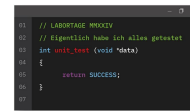
analysis problem: covering all cases is complex
depth first approach with narrow focus: Intel ME 11

https://github.com/skochinsky/me-tools
MEAnalyzer:

- written in Python
- 14k lines of code, thousands of lines of spaghetti logic
- 200+ classes/structs
- ~80 helpers, 20+ for analysis:
  `fd_anl_init`, `ext_anl`, `mod_anl`, `mfs_anl`, …

(line breaks added to fit on slide)

```
fd_exist,reading,file_end,start_man_match,end_man_match,
start_fd_match,end_fd_match,fd_count,fd_comp_all_size,
fd_is_ich,fd_is_cut,reading_msg = \
fd_anl_init(reading,file_end,
            start_man_match,end_man_match)
```



More Sources

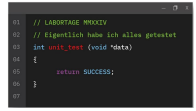
BIOS modding community (Win-Raid forum[8] etc)
https://github.com/corna/me_cleaner/wiki/How-does-it-work%3F
https://github.com/mostav02/Remove_IntelME_FPT
https://github.com/hardenedlinux/firmware-anatomy/blob/master/hack_ME/me_info.md
Fiano[9] + CSS[10] + Immune[11]

- written in Go
- CSS uses Fiano (some code moved over)
- special purposes covered, not for more general use

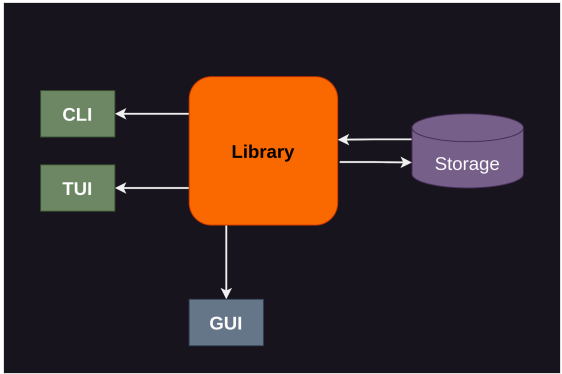
Romulan[12]

- written in Rust ;)
- forked from Jeremy Soller / System76

[8] https://winraid.level1techs.com/t/intel-converged-security-management-engine-drivers-firmware-and-tools-2-15/30719
[9] https://github.com/linuxboot/fiano/
[10] https://github.com/9elements/converged-security-suite/
[11] https://github.com/immune-gmbh/guard-oss
[12] https://github.com/orangecms/romulan



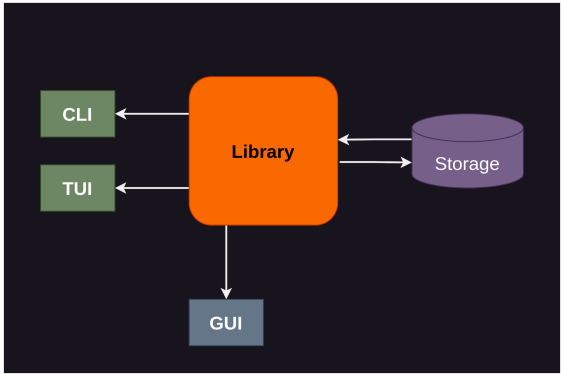
Trinity of Representations



internal
external (API, JSON)
binary (memory, file)

```
01  // LABORTAGE MMXXIV
02  // Eigentlich habe ich alles getestet
03  int unit_test (void *data)
04  {
05      return SUCCESS;
06  }
07
```

Trinity of Representations

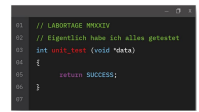


internal
external (API, JSON)
binary (memory, file)

Example

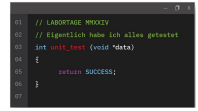
https://github.com/oxidecomputer/amd-apcb

`src/{ondisk,serialization}.rs`



me_fs_rs

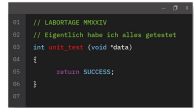
https://github.com/fiedka/me_fs_rs



me_fs_rs

https://github.com/fiedka/me_fs_rs

- library for parsing ME file systems
- FIT
- FPT (partitions)
- manifests
- Gen 2 directories
- Gen 3 CPDs
- MFS (WIP)



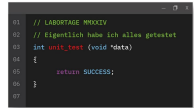
me_fs_rs

https://github.com/fiedka/me_fs_rs

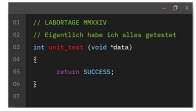
- library for parsing ME file systems
  - FIT
  - FPT (partitions)
  - manifests
  - Gen 2 directories
  - Gen 3 CPDs
  - MFS (WIP)

Crates

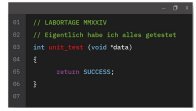
- `zerocopy` for parsing
- `serde` for JSON serialization
- `clap` for CLI



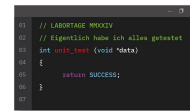
Let’s have a look!



Fiedka the Firmware Editor



What is Fiedka again?



What is Fiedka again?

Fiedka is the graphical desktop **fi**rmware analyzer and **ed**itor.

https://fiedka.app

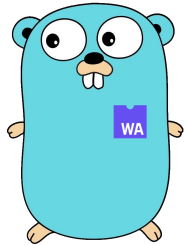
```
01  // LABORTAGE MMXXIV
02  // Eigentlich habe ich alles getestet
03  int unit_test (void *data)
04  {
05      return SUCCESS;
06  }
07
```

What is Fiedka again?

Fiedka is the graphical desktop **fi**rmware analyzer and **ed**itor.

https://fiedka.app

The backend runs in WebAssembly.



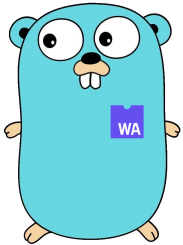

```
01  // LABORTAGE MMXXIV
02  // Eigentlich habe ich alles getestet
03  int unit_test (void *data)
04  {
05      return SUCCESS;
06  }
07
```

What is Fiedka again?

Fiedka is the graphical desktop **fi**rmware analyzer and **ed**itor.

https://fiedka.app

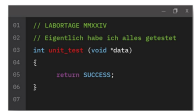
The backend runs in WebAssembly.



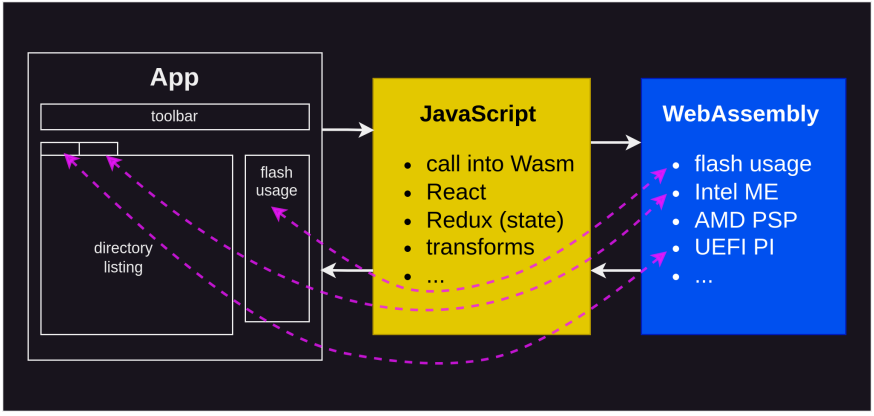
Rust support is being added using

Romulan: https://github.com/system76/romulan/

`me_fs_rs`: https://github.com/fiedka/me_fs_rs/

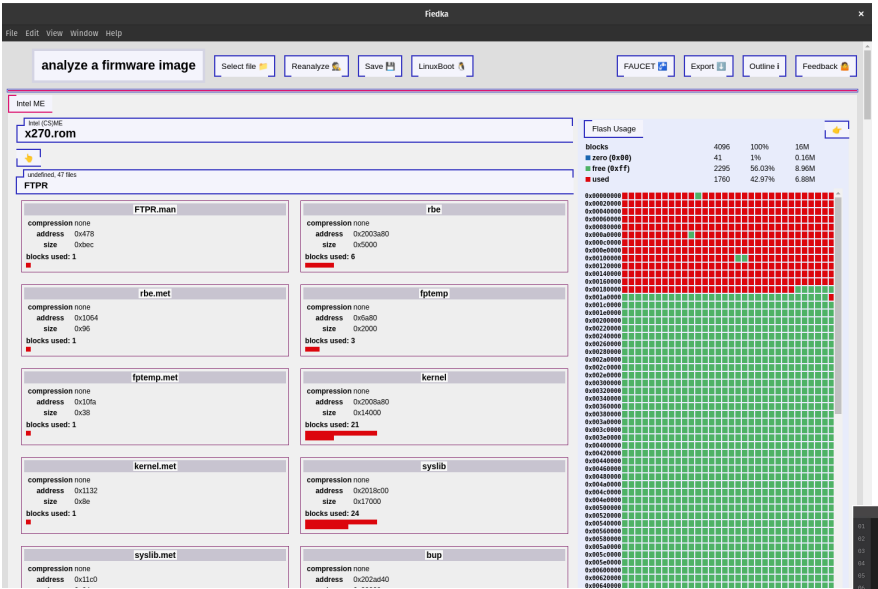


App Architecture

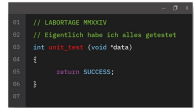



```
01  // LABORTAGE MMXXIV
02  // Eigentlich habe ich alles getestet
03  int unit_test (void *data)
04  {
05      return SUCCESS;
06  }
07
```

Fiedka in Action



DEMO: ME Firmware in Fiedka

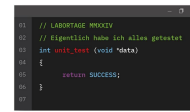


Related

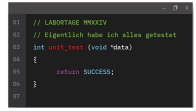
Look at ME! - Intel ME firmware investigation (FOSDEM 2020)
https://archive.fosdem.org/2020/schedule/event/firmware_lam/

Fiedka the Firmware Editor (OSFC 2021)
https://www.osfc.io/2021/talks/fiedka-the-firmware-editor/

Rust WebAssembly in Electron (Labortage 2023)
https://wiki.das-labor.org/w/Labortage_2023#Rust_WebAssembly_in_Electron



Thank you! :)

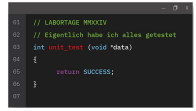

// LABORTAGE MMXXIV
// Eigentlich habe ich alles getestet
int unit_test (void *data)
{
return SUCCESS;
}

Follow Me

Daniel Maslowski

https://github.com/orangecms
https://twitter.com/orangecms
https://mastodon.social/@cyrevolt
https://youtube.com/@cyrevolt
https://twitch.tv/cyrevolt

https://metaspora.org/look-at-me-in-rust-labortage2024.pdf



BootGuard

https://designintools.intel.com/intel-bootguard-info.html

https://edc.intel.com/content/www/us/en/design/ipla/software-development-platforms/client/platforms/alder-lake-desktop/12th-generation-intel-core-processors-datasheet-volume-1-of-2/010/boot-guard-technology/

https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/resources/key-usage-in-integrated-firmware-images.html

https://eclypsium.com/blog/the-keys-to-the-kingdom-and-the-intel-boot-process/

https://eclypsium.com/blog/firmware-security-realizations-part-2/

deguard

https://codeberg.org/libreboot/deguard