

Look at ME in Rust! 🦀

Daniel Maslowski

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

Agenda

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

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

Intel (CS)ME Platforms

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

Knowledge

thanks to reverse engineering, presentations and tools

Alexander Tereshkin and Rafal Wojtczuk¹

Igor Skochinsky²

Dmitry Sklyarov, Maxim Goryachy, Mark Ermolov³

Peter Bosch⁴

Plato Mavropoulos⁵

¹<https://invisiblethingslab.com/resources/bh09usa/Ring%20-3%20Rootkits.pdf>

²<https://github.com/skochinsky/papers>

³<https://github.com/ptresearch>

⁴https://media.ccc.de/v/36c3-10694-intel_management_engine_deep_dive

⁵<https://github.com/platomav/MEAnalyzer/>

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

Knowledge

thanks to reverse engineering, presentations and tools

Alexander Tereshkin and Rafal Wojtczuk¹

Igor Skochinsky²

Dmitry Sklyarov, Maxim Goryachy, Mark Ermolov³

Peter Bosch⁴

Plato Mavropoulos⁵

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>

⁴https://media.ccc.de/v/36c3-10694-intel_management_engine_deep_dive

⁵<https://github.com/platomav/MEAnalyzer/>

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

Knowledge

thanks to reverse engineering, presentations and tools

Alexander Tereshkin and Rafal Wojtczuk¹

Igor Skochinsky²

Dmitry Sklyarov, Maxim Goryachy, Mark Ermolov³

Peter Bosch⁴

Plato Mavropoulos⁵

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>

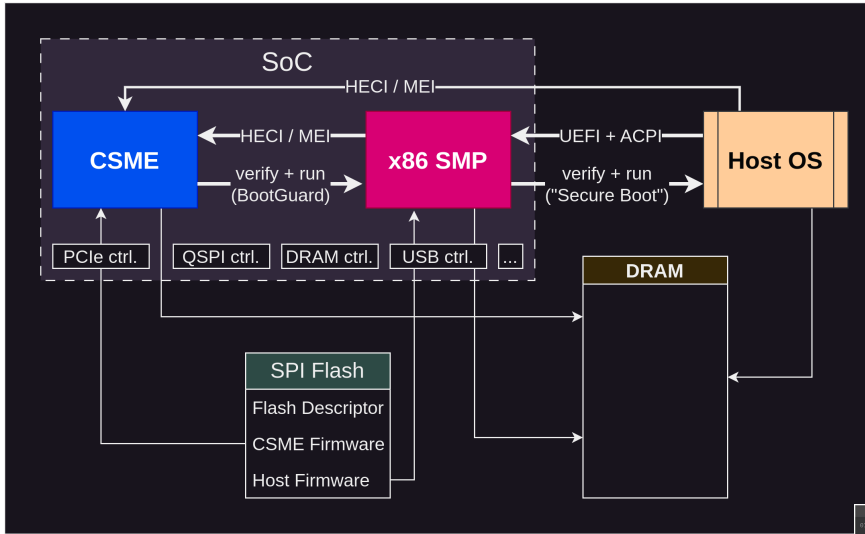
³<https://github.com/ptresearch>

⁴https://media.ccc.de/v/36c3-10694-intel_management_engine_deep_dive

⁵<https://github.com/platomav/MEAnalyzer/>

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

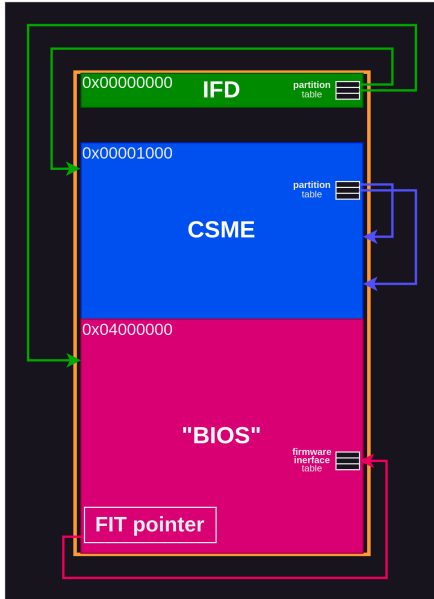
Hardware Platform and Flow⁶



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

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

Flash Partitioning



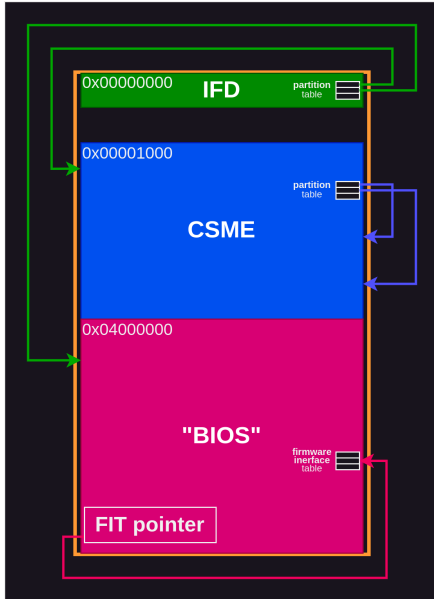
Intel Flash Descriptor (IFD)
partitions whole flash

Flash Partition Table (FPT)
partitions ME firmware

Firmware Interface Table (FIT)
points to microcode updates, code
modules, manifests and policies

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

FPT (Flash Partition Table)

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

FPT (Flash Partition Table)

Data Partitions

lots of partitions; not investigated any further

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

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>

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

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 EDFS (partition) the same?

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

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

FIT (Firmware Interface Table)

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

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

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.

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

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>

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

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>

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

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). :)

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

DEMO: FPT + FIT in CLI

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

Library and CLI

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

Background

analysis problem: covering all cases is complex

depth first approach with narrow focus: Intel ME 11

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

Background

analysis problem: covering all cases is complex

depth first approach with narrow focus: Intel ME 11

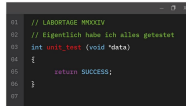
<https://github.com/skochinsky/me-tools>

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)
```



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

More Sources

BIOS modding community (Win-Raid forum⁸ 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

Fiano⁹ + CSS¹⁰ + Immune¹¹

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

Romulan¹²

- ▶ 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/>

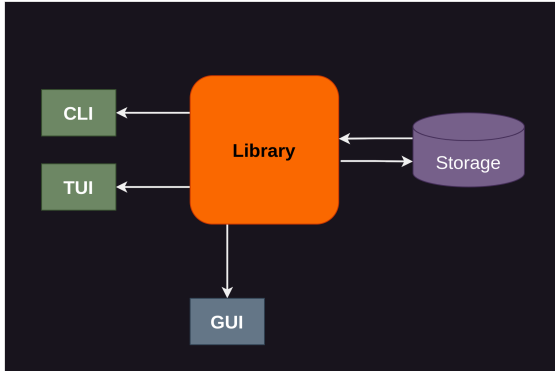
¹⁰<https://github.com/9elements/converged-security-suite/>

¹¹<https://github.com/immune-gmbh/guard-oss>

¹²<https://github.com/orangecms/romulan>

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

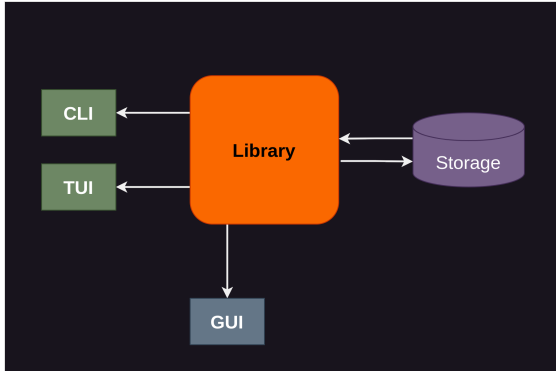
Trinity of Representations



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

```
01 // LABORTAGE MXXXIV
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

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

me_fs_rs

https://github.com/fiedka/me_fs_rs

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

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)

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

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

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

Let's have a look!

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

Fiedka the Firmware Editor

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

What is Fiedka again?

```
01 // LABORTAGE MXXXIV
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
firmware analyzer and **e**ditor.



<https://fiedka.app>

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

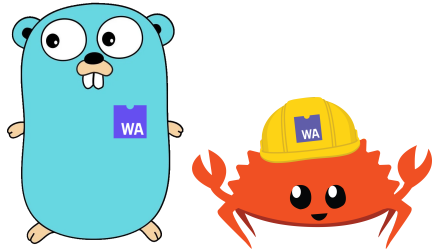
What is Fiedka again?

Fiedka is the graphical desktop
firmware analyzer and **e**ditor.



<https://fiedka.app>

The backend runs in WebAssembly.



```
01 // LABORTAGE MXXXIV
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
firmware analyzer and **e**ditor.



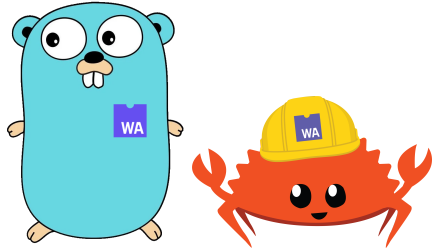
<https://fiedka.app>

Rust support is being added using

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

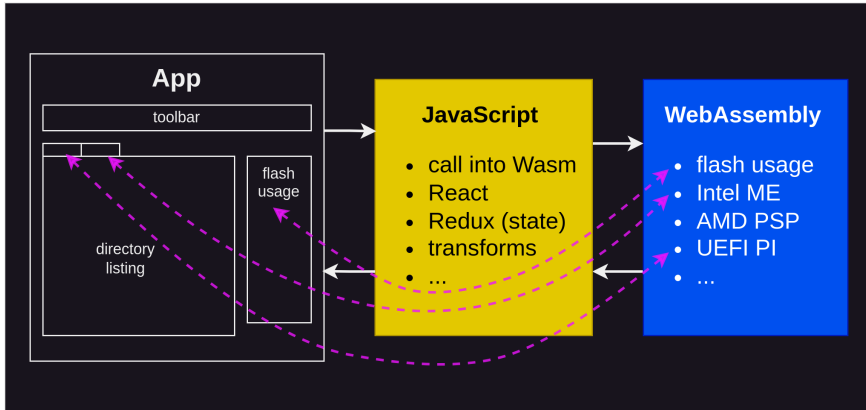
me_fs_rs: https://github.com/fiedka/me_fs_rs/

The backend runs in WebAssembly.



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

App Architecture



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

Fiedka in Action

Fiedka

File Edit View Window Help

analyze a firmware image

Select file Reanalyze Save LinuxBoot FAUCET Export Outline Feedback

Intel ME

Intel (CS)ME

x270.rom

undefined, 47 files

FTPR

FTPR.man

compression none
address 0x478
size 0xbec
blocks used: 1

rbe

compression none
address 0x2003a80
size 0x5000
blocks used: 6

rbe.met

compression none
address 0x1064
size 0x96
blocks used: 1

fptemp

compression none
address 0x6a80
size 0x2000
blocks used: 3

fptemp.met

compression none
address 0x10fa
size 0x38
blocks used: 1

kernel

compression none
address 0x2008a80
size 0x14000
blocks used: 21

kernel.met

compression none
address 0x1132
size 0x8e
blocks used: 1

syslib

compression none
address 0x2018c00
size 0x17000
blocks used: 24

syslib.met

compression none
address 0x11c0

bup

compression none
address 0x202ad40

Flash Usage

	blocks	4096	100%	16M
zero (0x00)	41	1%	0.16M	
free (0xff)	2295	56.03%	8.96M	
used	1760	42.97%	6.88M	

0x00000000 0x00020000 0x00040000 0x00060000 0x00080000 0x000a0000 0x000c0000 0x000e0000 0x00100000 0x00120000 0x00140000 0x00160000 0x00180000 0x001a0000 0x001c0000 0x001e0000 0x00200000 0x00220000 0x00240000 0x00260000 0x00280000 0x002a0000 0x002c0000 0x002e0000 0x00300000 0x00320000 0x00340000 0x00360000 0x00380000 0x003a0000 0x003c0000 0x003e0000 0x00400000 0x00420000 0x00440000 0x00460000 0x00480000 0x004a0000 0x004c0000 0x004e0000 0x00500000 0x00520000 0x00540000 0x00560000 0x00580000 0x005a0000 0x005c0000 0x005e0000 0x00600000 0x00620000 0x00640000

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

DEMO: ME Firmware in Fiedka

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

Related

Look at ME! - Intel ME firmware investigation (FOSDEM 2020)

https://archive.fosdem.org/2020/schedule/event/firmware_lam/

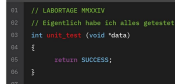
Fiedka the Firmware Editor (OSFC 2021)

<https://www.osfc.io/2021/talks/fiedka-the-firmware-editor/>

Rust WebAssembly in Electron (Labortage 2023)

[https://wiki.das-](https://wiki.das-labor.org/w/Labortage_2023#Rust_WebAssembly_in_Electron)

[labor.org/w/Labortage_2023#Rust_WebAssembly_in_Electron](https://wiki.das-labor.org/w/Labortage_2023#Rust_WebAssembly_in_Electron)



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

Thank you! :)

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

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>

License: CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>

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

BootGuard

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

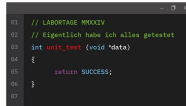
<https://edc.intel.com/content/www/us/en/design/ipla/software-development-platforms/client/platforms/alder-lake-desktop/12th-generation-intel-core-processors-datasheet-volume-1-of-2/010/boot-guard-technology/>

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

<https://eclipsium.com/blog/the-keys-to-the-kingdom-and-the-intel-boot-process/>

<https://eclipsium.com/blog/firmware-security-realizations-part-2/deguard>

<https://codeberg.org/libreboot/deguard>



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