

# Look at ME!

## Investigating Intel ME Firmware

Daniel Maslowski



# Disclaimer







This is not about whether we should trust Intel or any (chip) vendor.

Many details about the ME are not public or scattered across the web.

I probably have errors in some places; please report them to me.



# Agenda

-  Introduction
-  Open Source Firmware
-  Intel x86 Hardware
-  Motivation
-  Firmware Analysis
-  Conclusion



# Introduction



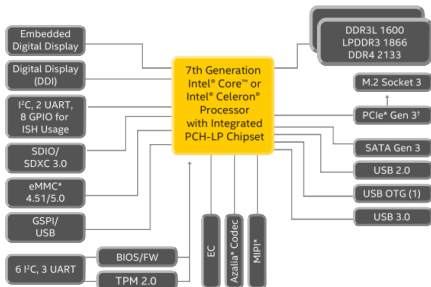


**NoT**  
Network  
of Things



# Microcontrollers and SoCs on your x86 mainboard

- Chipset (southbridge)
- Gigabit Ethernet (Gbe)
- USB controller
- PCI(e)
- SATA
- GPU
- HD Audio
- Bluetooth module
- Wi-Fi module
- ...



Kaby Lake U Mobile block diagram  
adapted from Intel specifications

## Critical Controllers

- Trusted Platform Module (TPM)
- Embedded Controller (EC)
- Baseboard Management Controller (BMC)





# Open Source Firmware





# Open Source Firmware projects



## Host (CPU, main SoC, chipset)

-  coreboot
-  LinuxBoot
  - ▶ Heads
  - ▶ u-root

## Embedded Controller (EC)

-  Chromium OS EC
-  System76 EC

## Baseboard Management Controller (BMC)

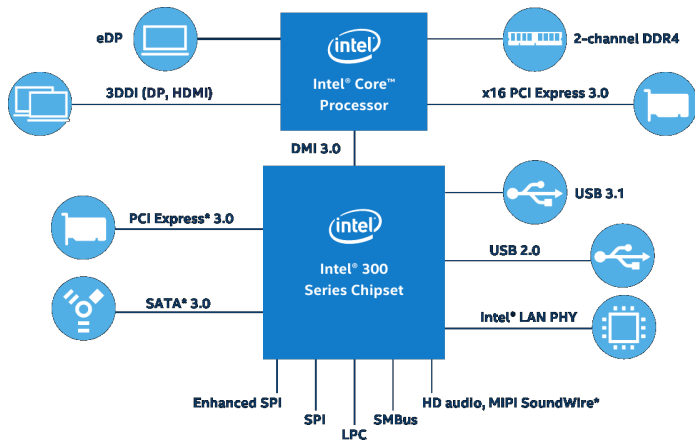
-  OpenBMC
-  u-bmc



# Intel x86 Hardware

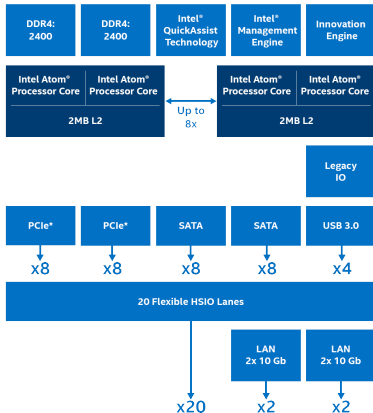


# Intel chipsets



# A closer look: Denverton platform

see Intel website and WikiChip



So what is this...?



Management Engine



Innovation Engine



# Intel Management Engine (today)



- 👤 microcontroller unit (MCU)
- 👤 part of chipset or System on Chip (SoC)
- 👤 connected to SPI flash, CPU, GbE
- 👤 started from Active Management Technology (AMT)
- 👤 may offer runtime services (more later)
- 👤 can verify host firmware (more later)



# Innovation Engine

## From Intel's website

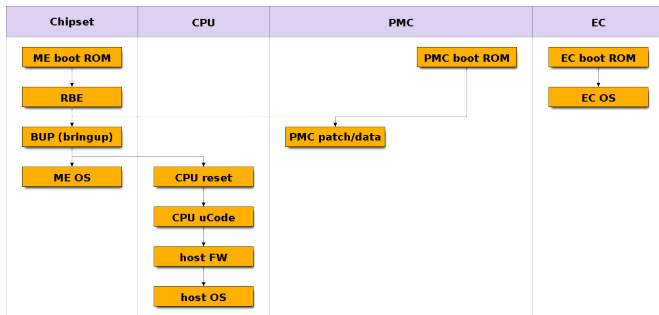
*Enables next-generation systems to customize solution firmware to drive greater operational efficiency, security, and predictive maintenance.*

-  HP Enterprise is using it
-  just a copy of the ME MCU



# Intel platform boot sequence

## How it all works together



vPro



# What is this vPro thing?

- 👤 umbrella marketing term for a set of technologies
  - ▶ includes AMT (Active Management Technology)
- 👤 as per ARK, for some chips, there is no “eligibility”
  - ▶ the hardware still supports it though
  - ▶ Intel does not provide details on eligibility

People are trying to figure out technicalities and availability details.

*mzygowski: I think BtG is supported on all CSME/ME silicons. TXE may be different (SecureBoot). ApolloLake TXE should have BtG (saw some Intel docs about it). So the information is really scattered*



# vPro Platform Security Features

It is hard to sum up how the entire platform works because everything is related.

- 👤 SGX (Software Guard Extensions)
- 👤 TXT (Trusted Execution Technology)
  - ▶ PTT, a TPM 2.0 implementation
- 👤 BootGuard



# AMT, MEI and ISH

## Active Management Technology

- 👤 available through MEI driver
  - ▶ hardware monitoring
  - ▶ power control
  - ▶ OS updates
  - ▶ storage
  - ▶ proxy for KVM (keyboard, video, mouse)

## Management Engine Interface

- 👤 implemented in Linux kernel




## Integrated Sensor Hub

- 👤 dedicated low power co-processor
- 👤 implemented in Linux Kernel



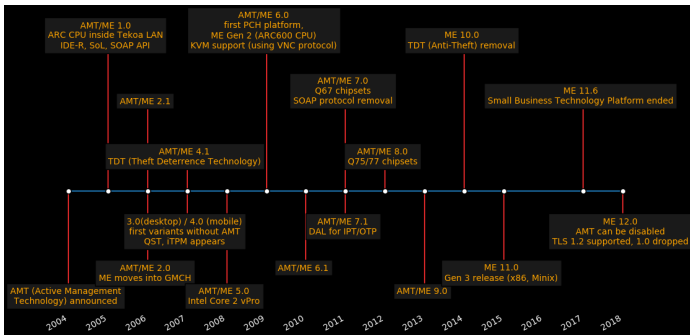
# MEBX

## Management Engine BIOS Extensions

-  configuration interface in host firmware
-  Ctrl + P or F6
-  default password is admin



# Once upon a time...



adapted from Igor Skochinsky - Intel ME Myths and Reality, Wikipedia and Intel



# Intel ME Version 12.0

- 👤 most recent documented release
- 👤 release notes are public
- 👤 supports TLS 1.2, dropped 1.0
- 👤 CIM\_Battery class
- 👤 AMT can be disabled
- 👤 category of “super\_critical” events

Version 14.0 has already been found in the wild in 2019.



# ME Firmware Variants

---

CON(S)	Consumer
COR(P)	Corporate
SLM(?)	Slim
SPS	Server Platform
	Services
IGN(?)	Ignition

---



# Motivation





# First public release of a redistributable ME firmware binary

## EDK II non-osi mailing list

*Ignition Firmware is a variant of ME firmware that is intended to provide lightweight chipset initialization. It does not contain all the features of the Intel® Server Platform Services (SPS) ME firmware. Ignition Firmware is consequently much smaller than Intel® SPS Firmware (~0.5 MB vs. ~3 MB).*

## Build and distribute full firmware images with binaries

-  Firmware Support Package (FSP) for host firmware
-  Ignition ME firmware for Cascade Lake / Purley



Follow the yellow brick road...



# ME Ignition Firmware License

*Redistribution and use in binary form, without modification, are permitted, provided that the following conditions are met:*

- 1. Redistributions must reproduce the above copyright notice and the following disclaimer in the documentation and/or other materials provided with the distribution.*
- 2. Neither the name of Intel Corporation nor the names of its suppliers may be used to endorse or promote products derived from this software without specific prior written permission.*
- 3. No reverse engineering, decompilation, or disassembly of this software is permitted.*



Pay no attention to that man behind the curtain!



# Philosophy

taken from training slides for FSP by Intel

## Philosophy

### There are ...

- plenty of smart firmware engineers
- comprehensive specifications and standards
- successful implementation examples using various boot loaders.

### There isn't ...

- enough open technical information to program a new silicon

### Therefore ...

- Intel provides what Intel knows the best, and let the ecosystem do what they are the best at



Intel® Intelligent Systems Summit  
Intel® Intelligent Systems: A new era in embedded computing

intel  
Intelligent Systems



# Vendor perspective

taken from the FSP whitepaper

*Intel is working towards releasing as much source code as possible going forward. A binary component is still the best way to encapsulate the complex solution that developers may not necessarily need to bother about as long as the binary component does its job right.*



## Dexter's Law

*Only proprietary software vendors want  
proprietary software.*



# Spotting the issue

Attackers do not play by the rules






## First steps






# Previous work / existing resources




## Analysis

-  me\_cleaner and its wiki
-  Heads docs on ME cleaner
-  MEAnalyzer

## Reverse engineering

-  ROMP module reverse engineering effort by Youness Alaoui
-  Huffman decoders
-  tools by Positive Research

## More information

-  talks by Igor Skochinsky
-  Win-Raid Forum
-  talk by Intel at Black Hat USA 2019



# Plundervolt





*We build on the reverse engineering efforts of [64, 49, 57] that revealed the existence of an undocumented MSR to adjust operating voltage on Intel Core CPUs. To ensure reproducibility of our findings, we document this concealed interface in detail. All results were experimentally confirmed on our test platforms (cf. Table I).*





# Can we trust the vendors?

Trust is complicated and hard to define.

## Blind trust

-  security by obscurity
-  consumers “don’t care”

## Established trust

-  full insight
-  personal relationship

Why do I have to disclose if a cookie may contain traces of nuts, but not what hardware actually contains or when software may have flaws?



# Security has many dimensions.

- 👤 physical: voltages, hardware accessibility
  - ▶ see Plundervolt
  - ▶ see Cacheout
- 👤 computational: constant-time for crypto ops
  - ▶ see TPM Fail
- 👤 logical: programmatic flaws
  - ▶ happens all the time
  - ▶ can be fixed with updates *if supplied*



# Responsible and Coordinated Disclosure

CVEs happen, which closed models make worse.

Lots of highly severe CVEs regarding (CS)ME were disclosed lately.

More issues were announced.



# Hardware and Firmware in Combination

Intel researchers agree that common taxonomy is needed.



# Security Perspectives

Auditability is a requirement, fulfilled by open source.

## Theorem

*no audit => no trust*



# Firmware Analysis



# Firmware Partition Table

00000000:	e9eb	0f02	0000	0000	0000	0000	0000	0000	.....
00000010:	2446	5054	0a00	0000	2010	209c	ffff	ffff	\$FPT.....
00000020:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000030:	4654	5052	0000	0000	0010	0300	0000	0400	FTPR.....
00000040:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000050:	4654	5550	0000	0000	0000	0000	0000	0000	FTUP.....
00000060:	0000	0000	0000	0000	0000	0000	0000	00ff	.....
00000070:	444c	4d50	0000	0000	0090	0000	0080	0200	DLMP.....
00000080:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000090:	4d46	5300	0000	0000	0040	0000	0020	0000	MFS.....@
000000a0:	0000	0000	0000	0000	0000	0000	0100	0000	.....
000000b0:	524f	4d42	0000	0000	0010	0000	0000	0000	ROMB.....
000000c0:	0000	0000	0000	0000	0000	0000	0100	0000	.....
000000d0:	4650	5442	0000	0000	0010	0000	0010	0000	FPTB.....
000000e0:	0000	0000	0000	0000	0000	0000	0100	0000	.....
000000f0:	4d46	5342	0000	0000	0020	0000	0020	0000	MFSB.....
00000100:	0000	0000	0000	0000	0000	0000	0100	0000	.....
00000110:	464c	4f47	0000	0000	0060	0000	0010	0000	FLOG.....
00000120:	0000	0000	0000	0000	0000	0000	0100	0000	.....
00000130:	5554	4f4b	0000	0000	0070	0000	0020	0000	UTOK.....p
00000140:	0000	0000	0000	0000	0000	0000	0100	0000	.....
00000150:	4643	5000	0000	0000	0010	0700	0020	0000	FCP.....
00000160:	0000	0000	0000	0000	0000	0000	0100	0000	.....



partition  
FTPR



offset  
0x31000






size  
0x40000



# Code Partition Directory

Each CPD entry can be either:

-  partition manifest (".man"), "old" generation 2 manifest
-  module metadata (".met"), also contains the module hash
-  module



# CPD data structure

see Win-Raid Forum

```
00031000: 2443 5044 0500 0000 0101 10b2 4654 5052 $CPD.....FTPR
00031010: 4654 5052 2e6d 616e 0000 0000 8800 0000 FTPR.man.....
00031020: f003 0000 0000 0000 7262 6500 0000 0000 .....rbe.....
00031030: 0000 0000 7005 0000 0090 0200 0000 0000 .....p.....
00031040: 7262 652e 6d65 7400 0000 0000 7804 0000 rbe.met.....x...
00031050: 7c00 0000 0000 0000 6d61 6e75 6600 0000 |.....manuf...
00031060: 0000 0000 7095 0200 0050 0000 0000 0000 .....p....P...
00031070: 6d61 6e75 662e 6d65 7400 0000 f404 0000 manuf.met.....
00031080: 7c00 0000 0000 0000 9400 0000 a100 0000 |.....
00031090: 0000 0100 0000 0000 8680 0000 1706 1920 .....
000310a0: fc00 0000 244d 4e32 0000 0000 0100 0000 ...$.MN2.....
000310b0: 0200 1d00 0100 0000 0000 0000 0000 0000 .....
000310c0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000310d0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000310e0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000310f0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
```



file

FTPR.man



offset

0x0088




size


0x03f0



# FTPR

 meaning unknown; could refer to *factory*, *partition*, *reset*


## files

 FTPR.man - FTPR manifest

 rbe

 rbe.met

 manuf

 manuf.met



# FTPR manifest

- 👤 seems to consist of three parts (lots of 0000 and ffff may be separators)
- 👤 header includes architecture (8086) and date (2019-06-17)
  - ▶ followed by the tag \$MN2
- 👤 more metadata? (FTPR itself, rbe, manuf)
- 👤 0x7c, 0x200200?



# Trailer?

rbe

```
7262 6500 0000 0000 0000 0000 0000 ffff 7c00 0000  
b5da a898 d17c c016 4c04 3b2c f141 c26b  
756a de87 dc2c 59b0 995a f551 ac0d e839
```

manuf

```
6d61 6e75 6600 0000 0000 0000 0000 ffff 7c00 0000  
9064 981d 6cf7 c15d 9a4a 64aa f081 58cc  
2619 a3ae 71ae 6230 8bdb 3694 a7cb 1b83
```

FTPR

```
0f00 0000 9c00 0000 4654 5052
```



## And almost the same thing again

rbe

```
7262 6500 0000 0000 0000 0000 0002 2000 7c00 0000  
b5da a898 d17c c016 4c04 3b2c f141 c26b  
756a de87 dc2c 59b0 995a f551 ac0d e839
```

manuf

```
6d61 6e75 6600 0000 0000 0000 0002 2000 7c00 0000  
9064 981d 6cf7 c15d 9a4a 64aa f081 58cc  
2619 a3ae 71ae 6230 8bdb 3694 a7cb 1b83
```




RCHA - what is that?

```
3200 0000 1000 0000 5243 4841 0000 0000
```



manuf

consists of three parts

-  bootpart
-  boot\_fpt
-  ftpr.mft



# x86 Instructions

manuf

```
00000000: 0fa0 66b8 3000 8ee0 b904 0000 0064 8b09
00000010: b800 0000 0064 8b00 ba04 0000 0064 8b12
```

PUSH FS ; segment register

MOV AX, 0x0030





MOV FS, AX

MOV ECX, 0x000004

MOV ECX,DWORD PTR FS:[ECX]


MOV EAX, 0x000000

## References

-  push onto stack
-  16-bit and 8-bit registers
-  single byte or small x86 opcodes
-  x86 assembler in 256 LOC





# PMC

 included twice, 65584 bytes - 64KB + 48B (3 \* 16B)

## Last three lines

```
00010000: 706d 635f 6677 5f6c 6267 5f62 302d 3138
          pmc_fw_lbg_b0-18
00010010: 7777 3334 6100 0000 0000 0000 0000 0137
          ww34a.....7
00010020: 0000 0100 0000 0000 0000 0000 0000 0000
          .....
```

 probably upper 64KB are actual image and last three lines are meta information

 pmc\_fw\_lbg\_b0-18ww34a looks like a version string



## Conclusion



# Security







All firmware has to be fully open source.



# Fully own your hardware

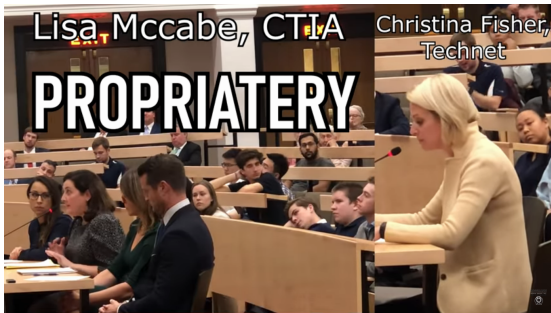
Run Linux everywhere?

Prerequisite: Code execution possible, preferably early, e.g., in mask ROM.

-  x86: LinuxBoot
-  BMCs: OpenBMC, u-bmc
-  routers: OpenWrt
-  iPhone: <http://iokit.racing/oneweirdtrick.pdf>
-  AMD PSP: exploits published at 36C3
-  Intel ME: ?



# Right to repair movement



vendors still propose security by obscurity

- ▶ although known to be pointless against sophisticated attackers
- ▶ repair technicians suffer from proprietary information
  - ▶ consumers and researchers alike



# Obtaining ME firmware images

- 👤 Win-Raid Forum
- 👤 dump your own devices' images



# Obtaining ME firmware images from Lenovo

- 👤 download update, e.g.,  
`https://support.lenovo.com/us/de/downloads/ds503998`
- 👤 run `innoextract [file] => app/` directory with files
- 👤 one for consumer and one for corporate version, `Me_xx.x_Coxx.bin` :)



# Obtaining ME firmware images from HP

- 👤 download update, e.g.,  
`h30318.www3.hp.com/pub/softpaq/sp99501-100000/sp998`
- 👤 run `7z x [file]` (in a new directory) => many files,  
we want `Q72_xxxxxx.bin`
- 👤 `xxd Q72_xxxxxx.bin | grep "\$FPT"` (extract  
line with FPT tag)
- 👤 note down address at beginning without 0 at the  
end, minus 1
- 👤 `dd if=Q72_xxxxxx.bin bs=16  
skip=0x[beginning] count=0x1000 of=me.bin`
- 👤 run MEA.py over it: `MEA.py me.bin`
- 👤 check expected length, try higher count for dd in  
case of error
- 👤 correct the count and dd again => there we go,  
`me.bin` :)



# Abbreviations and Acronyms

---





PMC	Power Management Controller
MSR(1)	Model-Specific Register
MSR(2)	Machine Status Register
PCR	Platform Configuration Register
FIT(C)	Flash Image Tool
FPT	Firmware Partition Table
CPD	Code Partition Directory
RBE	ROM Boot Extension
DAL	Dynamic Application Loader
PTT	Platform Trust Technology
FPF	Field Programmable Fuse

---





# Resources on AMT

## ISVs providing software for vPro

-  Solarwinds Dameware
-  Kaseya
-  Galaxy
-  42 Gears






## Additional resources

-  AMT Demystified Book (free)
-  AMT Case Studies



# Related work

## Talks from Black Hat USA 2019

-  Firmware Cartography: Charting the Course for Modern Server Compromise
-  Behind the scenes of iOS and Mac Security
-  Inside the Apple T2
-  Breaking Through Another Side: Bypassing Firmware Security Boundaries from Embedded Controller
-  Breaking Samsung's ARM TrustZone

## Talks by Alexander Ermolov

-  Safeguarding rootkits: Intel BootGuard



Kudos



Chaosdorf



LABOR.



coreboot



LinuxBoot



Thanks!



# Questions?

<https://github.com/orangecms/look-at-me>

<https://metaspora.org/look-at-me-fosdem20.pdf>

