TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Systems Architecture, Chair of Privacy and Data Security

Intel Management Engine

Hauptseminar Technischer Datenschutz

Dominik Pataky

June 2017

Contents

Author	Dominik Pataky <dominik.pataky@tu-dresden.de>
	PGP 0x80BF7C9C5B62468F
Supervisor	Dr. Stefan Köpsell
Time	Summer 2017
Last update	Sunday 18th June, 2017 16:07
License	CC BY-SA 4.0

The Intel Management Engine is an independent processor with access to the CPU, RAM and network embedded into most modern Intel chipsets. This is a research report on the topic for the university module „Hauptseminar Technischer Datenschutz“ at the Chair of Privacy and Data Security, Faculty of Computer Science, TU Dresden. I will first describe what the ME is, what hardware capabilities it owns and what software stack is running inside the chip. In later sections I analyze problems with security and privacy and possible protection mechanisms. This report is bundled with a presentation, which is based on this paper.

1 Introduction

The Intel Management Engine (ME) is a hardware chip embedded on Intel motherboards in addition to the main CPU. The chip is integrated in most modern Intel chipsets. The ME is most commonly mistaken as vPro or AMT, but is in fact the hardware chip for out-of-band (OOB) control which other Intel vPro software products use for their purpose. One of these implementations using the Intel ME is the Advanced Management Technology (AMT). The ME also exists for Intels server boards, named the Server Platform Services (SPS), and for Atom System on a Chip (SoC) chips, called the Security Engine (SEC).

1.1 The Intel vPro product family

The Intel vPro product family is an umbrella term that includes multiple sub-technologies. Hardware with a CPU that is delivered with vPro includes all or a subset of these technologies. vPro specializes on features for remote management, virtualization and security. Figure 1 shows most of the vPro products and their „grouping“.

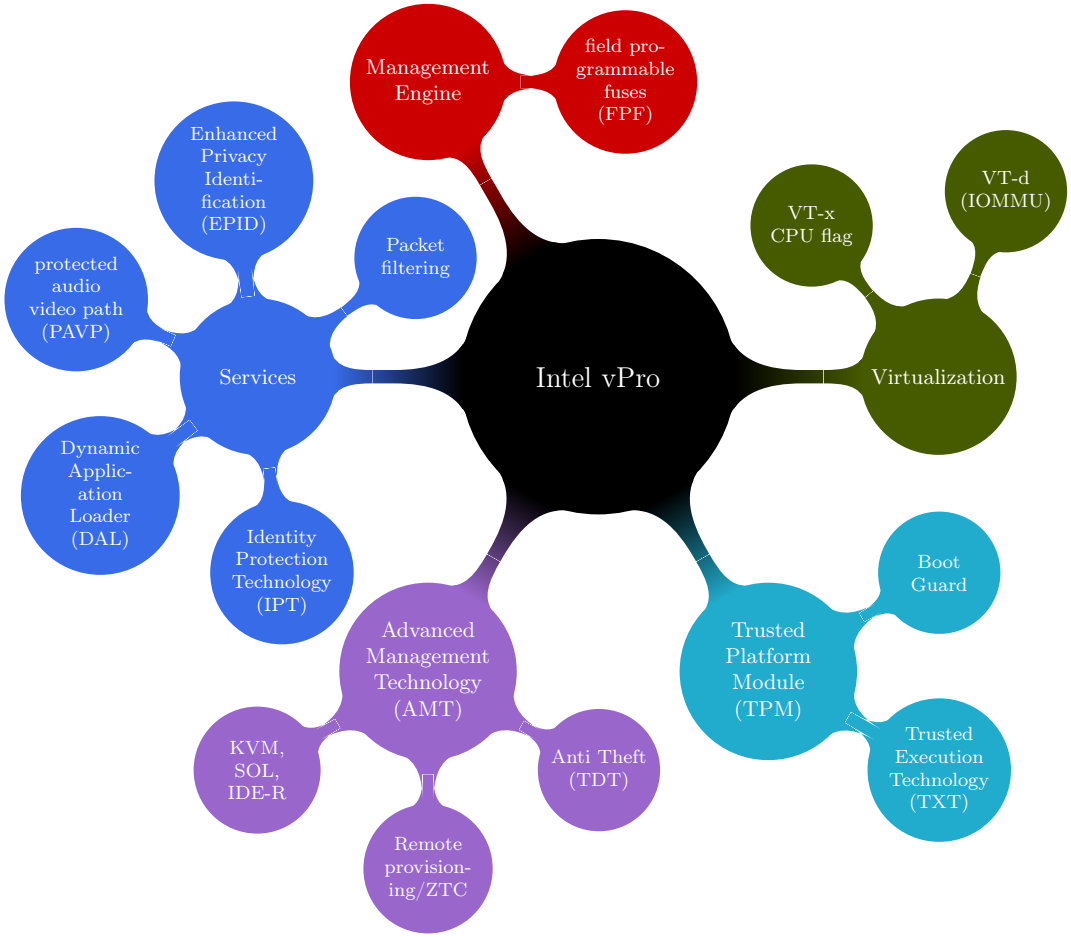

Figure 1: Partial overview of the main products that make up the Intel vPro product family.

1.2 The Management Engine

The Intel Management Engine (ME) is an out-of-band (which could be described as „outside of the normal flow paths“) 32-bit ARC microprocessor. It is integrated into the southbridge (see section 2 for a more detailed explanation of the hardware setup), has access to the CPU and to the dynamic random-access memory (DRAM). It uses a part of the DRAM to cache its own encrypted dynamic data. The chip is not dependent on the power state of the host machine (respectively the host operating system), since its use is to control computers which are stuck and need to be rebooted or start computers which are powered off. For reachability all the ME needs is a plugged in power supply and an Ethernet connection.

Access to the host system also includes the ability to capture network packets from the integrated network controller. As seen in [KGS09, chapter 10], AMT also provides a low-level firewall (packet filter). Additionally, the ME features an isolated cryptography chip, a secure storage and offers debugging capabilities. For more details about the software capabilities of the ME see section 3.

The following figure 2 shows the building blocks of the hard- and software of the Management Engine.

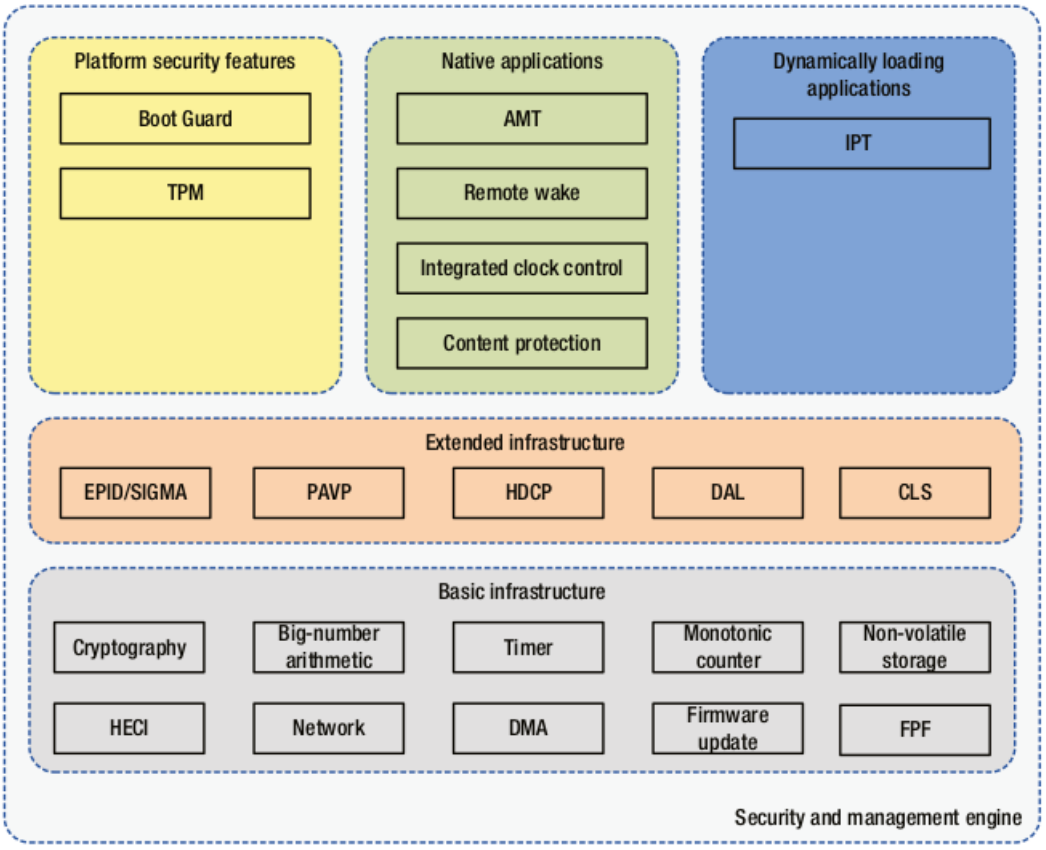

Figure 2: Intel ME components. At the bottom the underlaying hardware components provide the basic infrastructure. Above the extended infrastructure offers more special services, e.g. protected audio video path (PAVP) for secure display and Dynamic Application Loader (DAL) for applications loaded from external sources. Additional services are provided by e.g. the Trusted Platform Module (TPM), the Advanced Management Technology (AMT) and the Identity Protection Technology (IPT). Source: [Rua14, p. 235]

1.3 AMT version 11 (Series 200)

A revision of version 11 of the Management Engine, introduced in the Series 200 CPUs from this year (see section 2.2), brings a new chip and firmware design concept. As Sklyarov, Goryachy and Ermolov [SGE17] found out, the newest generation of the Intel ME chip will switch from an ARC architecture to x86. With this update the firmware layout changes as well, rendering the previous reverse engineering tools useless. There are traces of a new kernel, the open source Minix 3 microkernel.[1] [Rua14, p. 236] also mentiones that the newly developed Software Guard Extentions (SGX) technology makes use of the Management Engine.

1.4 Technologies similar to the Intel ME

The Intel ME as an implementation of an out-of-band processor beside the main CPU is only one of multiple variants currently on the market. Even though not all of the following products feature remote management, most of the hardware implementations are proprietary and can therefore never be fully audited for their full feature set.

Probably the best known remote management alternative is the **Intelligent Platform Management Interface (IPMI)**, which is an open standard with propietary and open source implementations[2]. The specification itself is lead by Intel since version 1.0 in 1998. The current version 2.0 was published in 2004 and was followed by a revision, version 1.1. The update included features like VLAN support and IPv6 addressing.[3] The implementations of the IPMI standard include the Dell DRAC feature [Del] and the HPE iLO products [Ent].[4] On the open source side there is for example GNU FreeIPMI[5] and OpenIPMI.[6]

Advanced Micro Devices, Inc. (AMD) as the main competitor to Intel has its own implementations for a secure side-band processor, the **AMD Secure Processor** (formerly known as Platform Security Processor (PSP)).[7][8] The Secure Processor is a dedicated processor implementing the TrustZone technology which is licensed from ARM Technologies. The core functionality of this processor however is not remote management but the execution of trusted (signed) software. But since AMD also states this technology enables „anti-theft software“ it is very likely that it also features some remote access capabilities. Bundled with its capability to control the data passed to the CPU (to differentiate between trusted and untrusted zones) it is indeed comparable to the Intel ME.

As mentioned, **ARM** as the licensor of the ARM processor family for embedded systems also features hardware implementations of secure environments. Their product „**TrustZone**“[9][10] is a hardware SoC component that divides software stacks into „worlds“ (which is another term for „domain“). This way, signed and trusted software can run in a secure world whereas any untrusted (unsigned) software is isolated in its own context.

It is also worth noting that there are multiple other hardware implementations for **Trusted Platform Module (TPM)** and **Digital Rights Management (DRM)** chips. The TPM technology is standardized as international ISO standards ISO/IEC 11889-1:2009[11] and ISO/IEC 11889-1:2015[12], written by the members of the Trusted Computing Group (TCG). This is also the case for DRM implementations. International members of the GlobalPlatform[13] comply to the Trusted Execution Environment (TEE) specifications and provide multiple commercial hardware variants.

[1] http://www.minix3.org/, last visit on 2017-05-22
[2] http://ipmiutil.sourceforge.net/docs/ipmisw-compare.htm, last visit on 2017-05-11
[3] http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-technical-resources.html, last visit on 2017-05-11
[4] http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-adopters-list.html, last visit on 2017-05-11
[5] https://www.gnu.org/software/freeipmi/, last visit on 2017-05-11
[6] http://openipmi.sourceforge.net/, last visit on 2017-05-11
[7] http://www.amd.com/en-us/innovations/software-technologies/security, last visit on 2017-05-11
[8] https://libreboot.org/faq.html#amd, last visit on 2017-05-11
[9] http://www.arm.com/products/security-on-arm/trustzone, last visit on 2017-05-11
[10] https://bits-please.blogspot.de/2016/06/trustzone-kernel-privilege-escalation.html, last visit on 2017-05-23
[11] https://www.iso.org/standard/50970.html, last visit on 2017-06-14
[12] https://www.iso.org/standard/66510.html, last visit on 2017-06-14
[13] https://www.globalplatform.org, last visit on 2017-06-14

2 Hardware capabilities and connections

The Intel Management Engine (ME) is an independent chip on Intel motherboards. This section explains Intels chipset design and the development and integration of the ME chip.

The diagram in fig. 3 shows the hardware connection in a Z87 chipset from Intel. Because Intel has its own chipset design which evolved from the standard northbridge-southbridge model, the following definitions are important to know:

- The **northbridge** is the CPU with some northbridge functionality embedded into the CPU. It is also known as the Memory Controller Hub (MCH) or Graphics and Memory Controller Hub (GMCH) if integrated graphics is present inside the CPU.
- The **southbridge** is named Platform Controller Hub (PCH). The earlier design before 2009 used the term I/O Controller Hub (ICH).

Both bridges connect different components of the overall hardware, depending on their speed. In further explanations I will use the terms northbridge and southbridge instead of the Intel specific names.

At the bottom the ME owned flash storage is connected via the Serial Peripheral Interface (SPI) bus to the southbridge. The northbridge is connected via the Direct Media Interface (DMI) 2.0 to the southbridge.

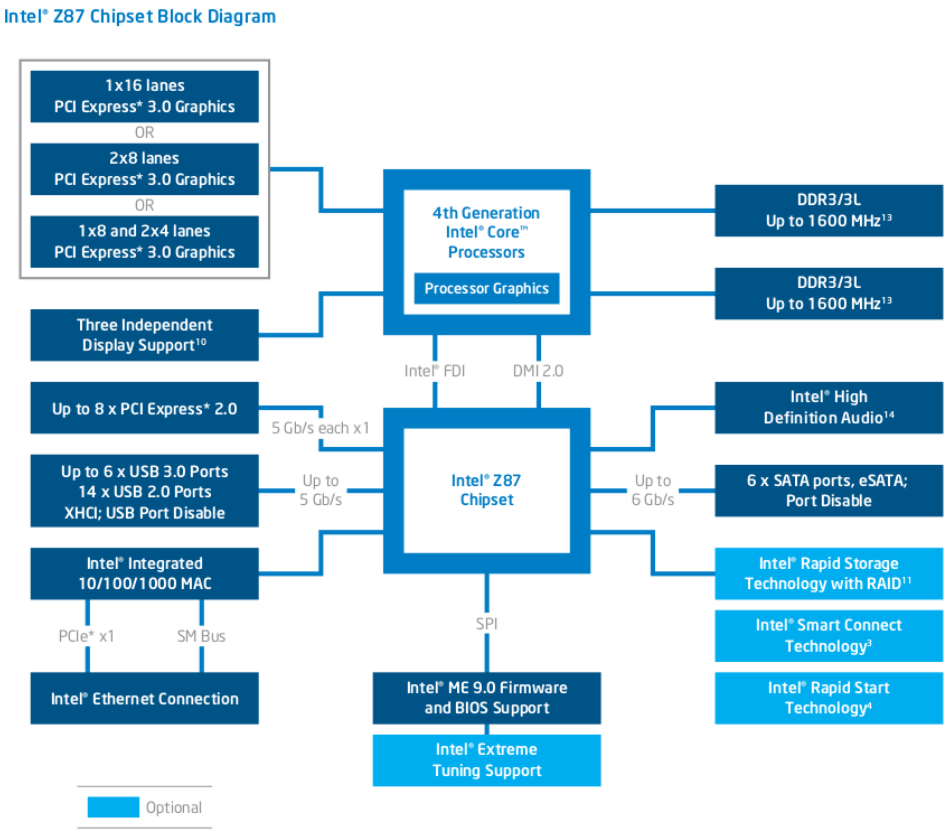

Figure 3: Overview of the Intel Z87 Series 8 chipset hardware. Source: http://www.intel.com/content/www/us/en/chipsets/z87-chipset-brief.html

The series 7 and 8 datasheets provide a more detailed information about the integration of the Intel ME chip into the southbridge.[14][15] In the first design from 2005 the ME microcontroller was integrated in the northbridge and additional hardware was embedded in the southbridge.

[14] http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/7-series-chipset-pch-datasheet.pdf, last visit on 2017-05-11

[15] http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/8-series-chipset-pch-datasheet.pdf, last visit on 2017-05-11

Since the Hub redesign in 2009, the Intel ME is embedded into the southbridge. The new design features better integration and isolation, a more robust security and the full „integration into the I/O subsystem“. The ROM, flash and the ME chip combine an independent isolated execution environment for secured applications. This security is ensured by memory enforcements, an established root of trust inside the Read-Only Memory (ROM) and further signed firmware parts that are only executed after verification with the key burnt into the ROM.

As mentioned above, the southbridge has access to the CPU via a DMI interface. This interface is used by certain modules of the ME to use the CPU for their own tasks. Secret data flowing through the external CPU is encrypted.

The coprocessor features isolated L1 and L2 caches and uses the hosts DRAM for L3 caching. The L1 cache is embedded inside the processor, whereas L2 caching is inside the engine and accessed over an unencrypted internal bus. Access to the DRAM of the host is made available via a designated direct memory access (DMA) component. The DMA is used for copying parts of the firmware, for data exchanged via wireless LAN and for encrypted video when DRM is used. Data saved in DRAM is encrypted and the integrity is checked with checksums.

The memory used for engine related tasks is managed by the memory manager. The memory manager keeps track of an overlay which divides the memory into task regions, called memory protection range (MPR). The regions are bundled with metadata and security rules, nonpriviledged tasks for example cannot access memory outside of their assigned region. Trying to access foreign memory results in exceptions. The copying of module code from the flash storage into the DRAM is managed by the loader.

The cryptography chip inside the ME offers a random number generator (RNG), hashing functions, AES, RSA and big number arithmetic. Furthermore the engine has dedicated hardware logic to handle Huffman compressed data.

2.1 Connections to integrated wired and wireless adapters

In earlier versions the component which is the ME today was originally integrated inside the network interface controller (NIC). When the feature set of this out-of-band (OOB) component grew, a redesign first integrated the functionality inside the northbridge and later in the southbridge. What is still one of the most important hardware integrations (primarily for the Advanced Management Technology (AMT)) is the direct connection to the NIC. Fig. 4 shows the flow of network packets through the NIC and the ME interface, resulting in either routing to the Operating System (OS) or OOB redirection to AMT.

For access to the network the ME can share the DHCP information and the MAC address with the host or use a configured IP address for communication.

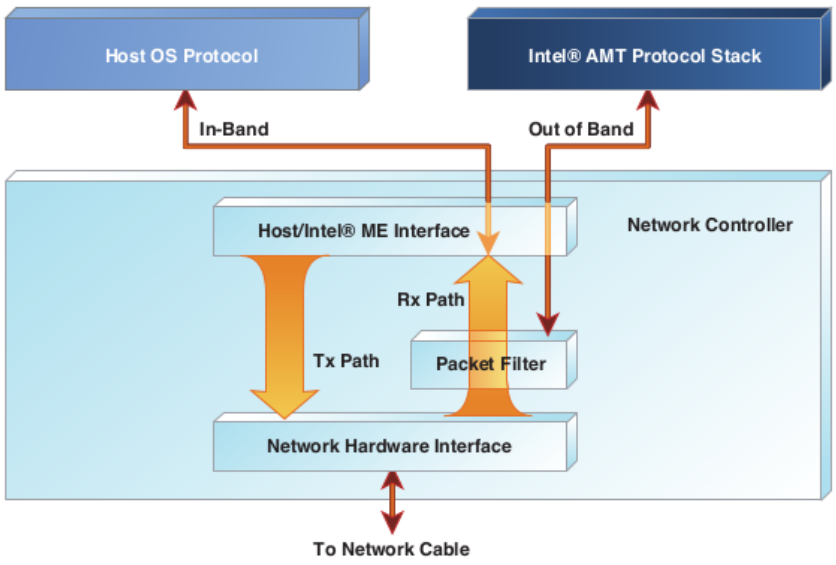

Figure 4: Diagram of the Ethernet controller integration with the embedded ME chip inside the southbridge chipset. Source: [KGS09, Fig. 7.6].

According to [Gar17], AMT also has access to wireless networks if configured explicitly by an administrator. There are two modes the communication between an external administrator and the ME over a wireless adapter works:

1. No OS is running, AMT uses the ME with configured credentials to connect to the network with the builtin wireless chip.
2. An OS is active and has control over the wireless adapter, AMT uses the guest driver to communicate over its designated ports. This requires proper implementation of this functionality inside the OS driver. This seems not to be the case in Linux, but in Windows.

2.2 Details about Intel chipset Series

The table in fig. 5 shows an overview over the different Intel chipset families – beginning with Intel Series 5, the first generation after the redesign in 2009 mentioned above. Each Series has multiple chip versions, for example Series 8 includes B85, Q87, H87 and Z87. All versions have different feature sets. They can be identified by the first number in the chipset name (Z87 being Series 8, Q250 being Series 200).

Series	Code name	Year	CPU code name	Core	LGA	ME
5	Ibex Peak	2009	Nehalem*	1^{st} gen	1156	6.0-6.2
6	Cougar Point	2011	Sandy Bridge	2^{nd} gen	1155	7.0-7.1
7	Panther Point	2012	Ivy Bridge	3^{rd} gen	1155	8.0-8.1
8	Lynx Point	2013	Haswell	4^{th} gen	1150	9.0-9.5
9	Wildcat Point	2014	Broadwell	5^{th} gen	1150	10.0
100	Sunrise Point	2015	Skylake	6^{th} gen	1151	11.0
200	Union Point	2017	Kaby Lake	7^{th} gen	1151	11.5-11.6

Figure 5: Overview of different Intel Series chipsets since Series 5 (1^{st} Generation Core). The column „ME“ shows the firmware version used in the corresponding chipset. The values in the column „Core family“ are used for marketing of the Intel i3/i5/i7 processors (not necessarily for Xeon and Celeron processors). „LGA“ defines the socket type of the motherboard using this chipset. All data was collected from https://ark.intel.com/, https://security-center.intel.com/advisory.aspx?intelid=INTEL-SA-00075 and https://en.wikipedia.org/wiki/List_of_Intel_chipsets including the various footnotes. *Note: for the first Core generation, the CPU code names were Lynnfield and Clarkdale, both based on the Nehalem microarchitecture

3 Software

This section is mostly based on Intel publications *Active Platform Management Demystified - Unleashing the power of Intel vPro Technology* by Kumar, Goel and Saint-Hilaire [KGS09] and *Platform Embedded Security Technology Revealed* by Ruan [Rua14], together with investigation results from the works of Skochinsky [Sko14] who reverse engineered parts of the firmware.

3.1 Overall integration on the board

The ME microcontroller is embedded inside the southbridge since a redesign in 2009. It has access to the dynamic random-access memory (DRAM) via direct memory access (DMA), is always powered on and can intercept network packages from the integrated Gigabit Ethernet (GbE) network interface controller (NIC) without the CPU knowing it. The code the ME uses is split: a very small boot routine is burnt inside the ROM, more complex modules which are loaded after the initial boot are saved in a shared flash chip that is also used by the Basic Input/Output System (BIOS). The ROM also contains the hash of the Intel signature key.

The southbridge chipset reads the partition boundaries on the Serial Peripheral Interface (SPI) flash and enforces the security of the ME region, locking out the BIOS.

3.2 Software stack

The ME is a seperate full-stack „mini-computer“. The microcontroller hosts its own kernel, a ThreadX from Express Logic (as mentioned in [SGE17] there are hints of a Minix 3 kernel in ME version 11). The ThreadX is a Real Time Operating System (RTOS) kernel with special capabilities for real time applications. This allows the ME to intercept network packets, video signals and other streams that are later passed into the CPU or GPU, without a degradation of performance. Since the ThreadX kernel is a so called nanokernel, it includes only the most basic functionalities for memory management, scheduling, timing and event handling. All other software parts are built on top of this abstraction layer.

One of the biggest modules is the Advanced Management Technology (AMT). Besides that the ME also includes multiple other services including low-level firewalling („System Defense“, which is network packet filtering), IDE redirection (IDE-R) for booting from a remote ISO, Serial over LAN (SOL), Identity Protection Technology (IPT) with One Time Password (OTP) functionality for Two-factor authentication (2FA) and „Intel Anti Theft“ (Theft Deterrence Technology (TDT)) with capabilities to remotely wipe the hardware by periodically checking a remote server for validation. See the diagram in fig. 1 for a better overview.

3.3 Stack integrity and Boot Guard

All code running on the ME is signed with an Intel RSA key pair (the Firmware Signing Key (FWSK)) which underlies very strict security guidelines. The SHA-256 hash of this key is burnt into the ROM, and is used by the boot module inside the ROM to verify every part of code from the boot up. Every partition with its modules inside the firmware is independently signed. This way it is not possible to modify and re-sign parts of the firmware without having the secret key. Fig. 6 shows the integrity dependencies which play into the signing process for module code.

Additionally to securing the integrity of its own firmware the Management Engine can also use Intel Boot Guard to expand verification functionality onto the BIOS and the hosts OS. For this the OEM must implement the Authenticated Code Module (ACM) into the firmware, before the BIOS. From here two modes are available to be used:

1. measured boot with Intel Trusted Execution Technology (TXT): hashes („measures“) the initial boot block (after the ACM) and stores the value inside the Trusted Platform Module (TPM). Code which is executed later (BIOS, OS) is incrementally hashed and combined with already hashed values. This incremental procedure is called „extend“.
2. verified boot: this method can be used without the need of a TPM. In this case the field programmable fuses (FPF) are the root of trust for the initial boot block. It and all later executed parts verify the following part. The FPF are burnt by the OEM and checked by the fuse manager task.

3.4 Firmware partitions

The modifiable firmware of the ME resides inside the SPI flash, which is shared between the BIOS, the ME and the NIC (in contrast to the read-only parts residing in the burnt-in ROM). This firmware consists of multiple partitions, decleared in the firmware partition table (FPT). Those partitions are also called „modules“ and may be signed and encrypted with an unknown Huffman encoding. Figure 7 shows an overview over the structure of the firmware and its modules.

3.5 Shared DRAM

Because the integrated memory belonging to the ME is neither big nor fast enough to buffer all features of the ME, the ME uses a part of the host system RAM for its own purposes. When booting, the ME starts in a very limited mode which requests a part of the RAM from the BIOS. This is handled by initiating the Unified Memory Architecture (UMA). The BIOS then locks the reserved part of the RAM so the CPU has no access to it.

This technique has evolved during the development of the Series chipsets. Currently, all data that is transferred into the DRAM is encrypted and thoroughly checked. Newer attacks against this sourced out memory region, for example a direct attack on the UMA by Skochinsky [Sko14], failed. Though Skochinsky could extract code from the BUP and KERNEL modules in later research by analysing a different firmware (this alternative firmware was originally assembled for Intel server motherboards). Section 4.6 shows a successful attack against the UMA which worked in Series 3.

3.6 Tasks: isolation, privileges, assets

Processes running inside the ME are called tasks. With a growing feature set the ME is used by more and more specialized processes. [Rua14] mentions that with Haswell more than 10 applications are embedded inside the ME. Every task loaded from the firmware is signed and checked individually and categorized as privileged and non-privileged. Of course, only a very small subset of tasks – like the ROM loader and inner kernel – are categorized as priviledged tasks. All other modules fall into the nonpriviledged category. For further details related to memory see section 2.

Modules that are run in nonpriviledged mode do not have access to hardware features and have to communicate through the kernel to exchange data. This is enforced by the task isolation mechanism and the inter-task call management. Any data a module wants to exchange with another must notify the kernel, which then copies the related data (e.g. as request and response).

This security enforcement also includes asset protection. All features a module can use (memory, storage, devices, synchronization objects like semaphores) are specified as assets. If a module requests change of an asset this manipulation must be done through the kernel. The kernel checks the ownership of the asset by comparing the task ID with the ID inside the metadata of an asset and only fulfills the request if both IDs match.

3.7 DAL, the Java Virtual Machine

As seen above the firmware includes a module for the Dynamic Application Loader (DAL). By reverse engineering those parts Skochinsky [Sko14] discovered strings that hint to a Java Virtual Machine (JVM) inside the firmware. Further investigation showed that this partition is formatted with the ISO standardized JEFF[16][17] storage format. As the packed code is not further obfuscated it is possible to read the Java classes. They include libraries for cryptography, storage access and UI elements.

Most of the results of Skochinskys investigation in 2012 are confirmed in [Rua14] in 2014.

The DAL is able to load and run custom applets in runtime. Those applets are reviewed and signed by Intel and the signature is checked for validity before execution, locking out any non-approved code. The execution itself is containered inside a Trusted Execution Environment (TEE), namely Trusted Foundations developed by Trusted Logic (aquired by Gemalto). The TEE implementation is based on the ARM TrustZone design and modified for Intel. This implementation enforces a context separation between multiple applets running at the same time.

The application code for an applet is loaded from the harddisk by the OS via the Management Engine Interface (MEI) (see section 3.8), so the DAL is only usable when the OS is powered and fully booted. Applets running inside the DAL are isolated from each other. They can request access to hardware devices

[16] http://www.unicode.org/L2/L2002/02042-jeff-spec.pdf, last visit on 2017-05-22
[17] https://github.com/skochinsky/jeff-tools, last visit on 2017-05-22

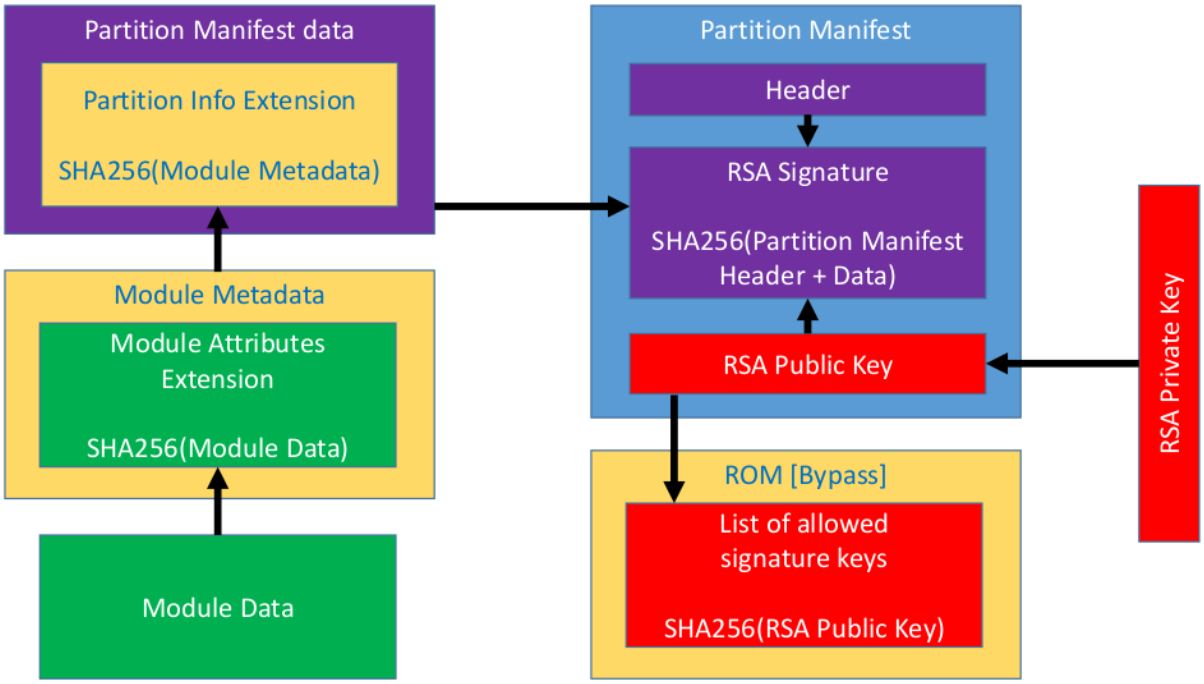

Figure 6: The Intel ME firmware integrity dependencies. On the left is the integrity path for each module, on the right the combination with the whole firmware parition inside the flash. The ROM part is the burnt-in read-only key list. The RSA private key on the right is a secret key that only Intel has access to. Source: [SGE17].

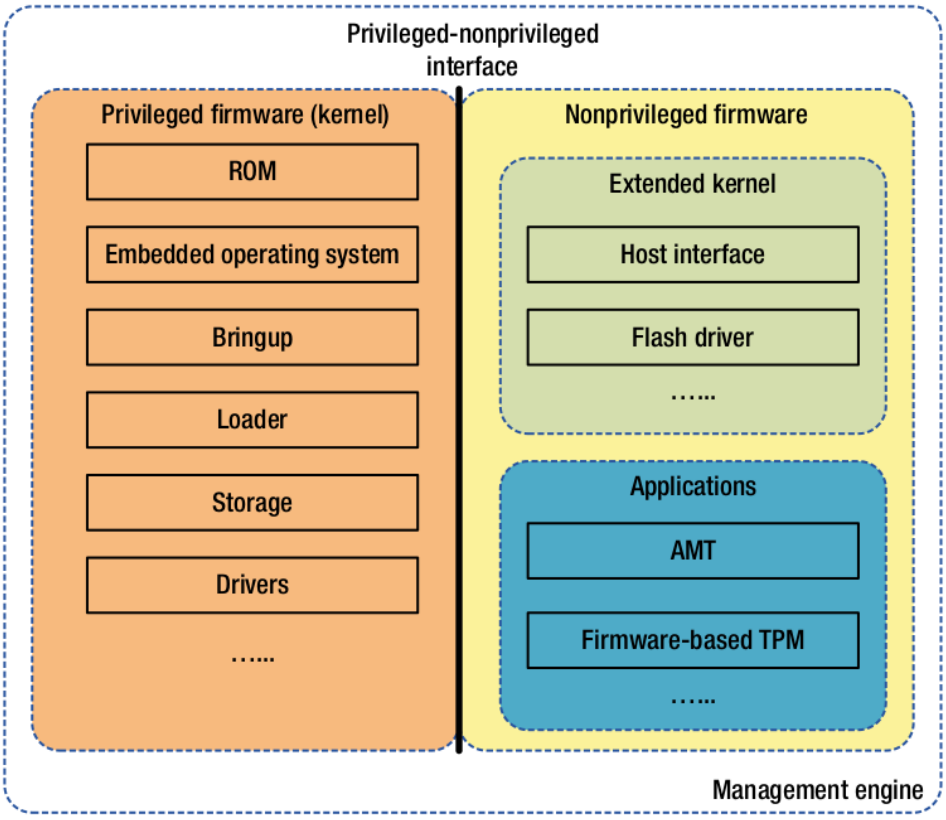

Figure 7: Overview over the different modules inside the firmware. For futher explanation of the priviledged and nonpriviledged differentiation see section 3.6. Source: [Rua14].

like the cryptography engine and the protected clock and can use other tasks like PAVP and EPID. The DAL also maintains a database of loaded applets with their version metadata. If a new applet is loaded, the security version is stored inside the database and all future runs of this applet are checked against the stored value. If the security level of an applet is lower than the stored value, the applet will not be executed. Extra data an applet needs to store for future use is encrypted and saved on the harddisk.

3.8 Bidirectional communication between OS and ME

After booting the Operating System (OS) which is installed on the host system a Management Engine is integrated in, the ME can be accessed via the Management Engine Interface (MEI) (in earlier versions this interface was called the Host Embedded Controller Interface (HECI)). For this, the OS needs compatible drivers. The free kernel Linux has this driver since version 3.5, released in 2012.[18][19] In Microsoft Windows, the OS needs a driver package supplied by Intel for its Identity Protection Technology (IPT) product[20]. It includes DLLs and the JOM DAL Host Interface (JHI) service that is running in the background.[21] [22]

After successful initialization of the interface, Linux has the device `/dev/mei`[23] and Windows the JHI service available. The JHI service is bundled with the Local Manageability Service (LMS) which routes packets sent from inside the OS to localhost into the JHI towards the ME. These interfaces are then used to send packets via the MEI toward the DAL task. The DAL receives the data, parses it and passes the stream to the applet running inside the JVM. Since the communication channel is bidirectional, the reply from the applet follows the same way – only backwards.

3.9 PAVP and DRM

The combination of technologies the Management Engine offers – the protected audio video path (PAVP), the direct memory access (DMA) and the Trusted Platform Module (TPM) – makes up a fully functioning Digital Rights Management (DRM) service. With DRM content producers use the Trusted Computing Group (TCG) TPM standard to secure protected multimedia content on the consumers machine. This includes checks for ownership, licensing of content and the restriction of content copying.

The chapter „Unleashing Premium Entertainment with Hardware-Based Content Protection Technology“ in [Rua14, p. 181] has detailed information about the implementation found inside the Management Engine.

Intel co-developed a technique called „UltraViolet“ as a member of the Digital Entertainment Content Ecosystem (DECE) group. UltraViolet is a cloud-based licensing service which checks the correctness of licenses a consumer holds via the internet. It is not a DRM scheme itself but offers its verification services for other DRM schemes like Sony OMA.

Since this method uses a crackable software protection mechanism which must be verified with technologies like Intel Boot Guard, Intel developed a solution to move this software solution into secure hardware. The current state deploys the above mentioned combination with PAVP to meet this requirements. Encrypted content that is delivered to the client is copied by the OS into the Graphics Processing Unit (GPU), the decryption key is handled by the Management Engine and sent directly to the GPU. The GPU then decrypts the content and starts a secured playback.

PAVP can optionally use the Enhanced Privacy Identification (EPID) task, external displays employ the Intel-developed High-bandwidth Digital Content Protection (HDCP) to secure the connection via cable.

[18] https://www.kernel.org/doc/Documentation/misc-devices/mei/mei.txt, last visit on 2017-05-23
[19] https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=ffc282, last visit on 2017-05-23
[20] https://downloadcenter.intel.com/download/25145/Intel-Identity-Protection-Technology-with-Public-Key-Infrastructure?product=46611, last visit on 2017-05-23
[21] http://www.shouldiremoveit.com/Intel-Management-Engine-Components-5127-program.aspx, last visit on 2017-05-23
[22] http://www.file.net/process/jhi_service.exe.html, last visit on 2017-05-23
[23] which identifies itself as a PCI device, for example `00:16.0 Communication controller: Intel Corporation 6 Series/C200 Series Chipset Family MEI Controller #1 (rev 04)`

4 Problems for security and privacy

Because of its invasive power, the Intel Management Engine (ME) is often criticized as a problem for security and privacy. The main problem of the hardware implementation is the loss of control by the user. In this section I am going to pick up some related topics, mixed with personal opinions.

4.1 Silent Bob is Silent (May 1st 2017)

As a good example of the risks the Intel Management Engine (ME) and Intel Advanced Management Technology (AMT) introduce to a computing platform we will now look further into a security incident on May 1st 2017. It is identified by INTEL-SA-00075[24] and CVE-2017-5689[25]. It was discovered by Maksim Malyutin, a security researcher at Embedi[26], who gave it the name „Silent Bob is Silent“[27].

The problem this incident brought to light is a programming error in the authentication algorithm used by the webinterface of AMT. The webinterface offers administrators easy access to the features of AMT on ports 16992 and 16993. Of course – since the interface gives full remote control over the AMT, which itself uses the ME to out-of-band control the hardware of the target computer – the interface is secured behind a login.

But the implementation of the administrative login, using a challenge-response digest authentication algorithm, is broken. Embedi shows in their report that while reverse engineering the AMT firmware they discovered that the function which checks the administrators response uses the function `strncmp(computed_response, user_response, response_length)` – but instead of the using the length of the computed response it uses the length of the user input. This means if the system waits for any response code but the attacker sends an empty string instead, the function passes because the length that is checked is zero.

The affected systems range from AMT version 6 to the latest 11 (known as ME Generation 2), starting in the 1^{st} Generation Core i3/i5/i7 series „Nehalem“ (Series 5), which introduced a hardware redesign in 2009. See section 2 for more details. Intel has released patches for the affected firmwares for OEMs, who work on the integration for their platforms.[28][29][30] There are multiple tools which help discovering the vulnerability.[31][32] There are also tools to disable AMT[33], but it is still unclear if tools from inside the computers OS can disable the ME completely.[34] This bug is even more problematic in combination with wireless enabled AMT setups. As described in 2.1 if the AMT is running with a Windows OS, it has access to the wireless controller. This feature cannot be disabled and even passes the Windows firewall [Gar17]. The result: any vulnerable computer connecting to a public WLAN can be „managed“ by an attacker who exploits the bug described in this section.

4.2 PLATINUM attack against SOL (June 7th 2017)

Another good example of a real world misuse of ME features is the SOL file-transfer exploit used by the PLATINUM hacking group. The exploit was discovered by the Microsoft Malware Protection Center (MMPC) in June 2017.[35]

For this attack to work a Windows host OS must have already been compromised and set up with AMT and Serial over LAN (SOL) credentials. After successfully enabling SOL (which is by default deactivated) with credentials, malware running on the Windows host can receive and send data over the SOL redirection driver. A remote attacker can connect to this hidden channel and send encrypted commands toward the malware. Because of the out-of-band (OOB) packet flow the Windows firewall is not able to filter out the attackers data stream.

[24] https://security-center.intel.com/advisory.aspx?intelid=INTEL-SA-00075, last visit on 2017-05-15
[25] https://nvd.nist.gov/vuln/detail/CVE-2017-5689, last visit on 2017-05-15
[26] https://www.embedi.com/news/mythbusters-cve-2017-5689, last visit on 2017-05-15
[27] https://www.embedi.com/files/white-papers/Silent-Bob-is-Silent.pdf, last visit on 2017-05-15
[28] https://support.lenovo.com/de/en/product_security/len-14963, last visit on 2017-05-15
[29] https://www.asus.com/News/uztEkib4zFMHCn5r, last visit on 2017-05-15
[30] http://h22208.www2.hpe.com/eginfolib/securityalerts/CVE-2017-5689-Intel/CVE-2017-5689.html, last visit on 2017-05-15
[31] https://downloadcenter.intel.com/download/26755, last visit on 2017-05-15
[32] https://github.com/mjg59/mei-amt-check, last visit on 2017-05-15
[33] https://mattermedia.com/blog/disabling-the-intel-management-engine/, last visit on 2017-05-03
[34] https://software.intel.com/en-us/forums/intel-business-client-software-development/topic/563988, last visit on 2017-05-15
[35] https://blogs.technet.microsoft.com/mmpc/2017/06/07/platinum-continues-to-evolve-find-ways-to-maintain-invisibility/, last visit on 2017-06-10

According to Microsoft it can not be determined if PLATINUM has found a way to remotely enable and configure SOL. This attack is the first known usage of SOL by malware.

4.3 Patching a firmware is difficult

Even though a lot of security holes can be resolved by patching the firmware, this process is difficult to automate. As seen in the aftermath of *4.1 Silent Bob is Silent (May 1st 2017)* patching the ME requires firmware updates from hardware vendors and cannot be done by Intel directly. After the vendor releases the patch, the owner of the hardware has to manually process the installation of the update. This can be a hard task[36] for end-users without enough experience. And even if the end-user is an administrator who originally had the intention to use AMT for administration it will take time and effort. And this all presumes the vendor does indeed release an update for vulnerable products.

Since people prefer to „never touch a running system“ than to patch regularly this is an obstacle which must be lowered or even automated. But who really wants unsupervised firmware updates for a component like the Management Engine? Unreviewed patches could possibly introduce new attack vectors, e.g. if the update mechanism is exploited to spread malware.

4.4 Risk of buying preconfigured used hardware

Leased business hardware which is returned for sale after the leasing contract expires is prone to preconfigured management solutions. AMT and products like the ones mentioned in 6.3 are targeted at these business users and their IT departments.

What really is a problem in this case are the decreased control possibilities a new user of this hardware is granted. A swap of the harddisk or a reinstallation of the OS is not sufficient to protect against exploits or usage errors. What if the hardware is still „phoning home“ to the old owner? What if the old owner or the leasing company did not wipe preconfigured anti-theft software that is installed inside the BIOS? A user without the knowledge of this low level software stack has very limited chance to completely own the bought hardware.

4.5 Breaking the Intel private key

As explained in the previous sections, code running inside the ME – especially dynamic code inside the Dynamic Application Loader (DAL) – is approved and signed with a secret Intel private key. The public key the engine uses to verify a signature is burnt into the ROM, so it cannot be tempered with. This security model has a problem which every key pair faces in the next decades: the increase in computing power might one day be enough to break the key in a reasonable timespan.

But what would this mean for hardware which is on the market now? The ROM cannot be upgraded, so the hash of the public key will forever be embedded into hardware released on the market at this moment. Once the key is brute-forced, everyone will be able to run „signed“ applets.

But why wait for better calculators – what if the secret key is plainly stolen from Intel? It can be assumed the secret key used in the manufacturing process is one of the most well guarded secrets in Intels facilities. And yet, leaks and break-ins happen repeatedly in this industry...

4.6 Planting rootkits inside the firmware

In 2009 Tereshkin and Wojtczuk [TW09] demonstrated a „Ring -3“ rootkit at the Black Hat USA conference. The term „Ring -3“ was defined as the level below the System Management Mode (SMM) (Ring -2), the Hypervisor (Ring -1) and the kernel (Ring 0). Because of the control the ME has over the CPU any rootkits running inside the ME have to be considered even lower than anything running inside the processor.

The attack affected the Q35 chipset and was executed by a memory remapping attack. It was possible to remap the secured memory region which was reserved for the ME firmware into a CPU readable memory region. This way the OS could access the memory area and modify the code residing in the firmware. The fix was deployed via BIOS upgrades, but malware was able to automatically downgrade BIOS firmware so this fix did not resolve the issue.

A later investigation however showed that the Q45 chipset introduced new memory security techniques. Those are most probably still active in the newest chipsets and are described in section 3.

[36] https://download.lenovo.com/pccbbs/mobiles/n1qrg14w.txt, last visit on 2017-05-23

5 Protection

Protecting owned hardware against the threats mentioned in the previous section is a very difficult task. When looking at possible threats and attack vectors one must make a differentiation between threats coming from Intel as the developer and manufacturer of the Management Engine and outside attackers exploiting bugs of the implementation.

Related to product design one thing is important to note: Intel integrates the ME into its southbridge on all produced chipsets. It does not make a difference if AMT is enabled or not, the microcontroller with all its features and capabilities is embedded into the hardware and cannot be disabled by a hardware switch or by manually removing the chip.

5.1 Disabling the Management Engine

It is not possible to disable the Intel Management Engine (ME). The hardware is deeply embedded inside the chipset and only offers software switches for management of enabled features. Even though parts of the software stack can be set inactive, the integrated Gigabit Ethernet (GbE) network interface controller (NIC) forwards packets toward the ME for further processing.

What can be disabled however are the activated features running inside the firmware.

- The most basic approach to this is entering the management menu for Intel ME by either accessing it via the BIOS setup or by pressing CTRL+P at boot.
- To overwrite features, consider the removal of firmware parts with tools like the me_cleaner[37].
- If you do not want to touch the firmware, using an alternate BIOS like Libreboot[38] is an option.

According to [per16] it is possible to completely erase the firmware on Series 4 chipsets, but not later. Later chipsets need more work with external SPI flashing hardware and a rewrite of the firmware flash image. However it is possible to minimize the firmware to very limited functionality on newer chipsets, from Nehalem up to the newest generation, Kaby Lake (AMT 11).[39]

5.2 Filtering traffic targeting the ME

The ports the ME listens on (ports the ME filters packages from to reroute them into OOB hardware) are IANA approved and span the range 16992-16995:

- 16992: HTTP access to AMT webinterface
- 16993: HTTPS access to AMT webinterface
- 16994: Unencrypted Serial over LAN (SOL) and IDE redirection (IDE-R)
- 16995: SOL and IDE-R over TLS
- 5900: (optionally) Virtual Network Computing (VNC)

Using a firewall in front of the computer with a ME, packets that might be received by the ME on these ports can be filtered out. This way no packets are ever flowing into the embedded packet filter inside the network interface controller (NIC) and ME – reducing the attack surface.

5.3 Monitoring BIOS

According to [Hof13] the feature set of AMT depends on the OEMs BIOS implementation and differs from machine to machine, given they have different manufacturers.

As seen in previous chapters the ME firmware and the computers BIOS are an important software bundle in every AMT setup. Because the ME firmware is thoroughly signed and protected, the BIOS is a component that is left for the hardware vendor to modify and deploy. Any custom implementations and even the integration of the ME BIOS extensions (MEBX) increase the risk of new attack vectors.

It is therefore not only necessary to regularly patch the BIOS (presuming the vendor publishes upgrades) but also to monitor it for possible modification by malware.

[37] https://github.com/corna/me_cleaner, last visit on 2017-05-23
[38] https://libreboot.org, last visit on 2017-05-11
[39] https://github.com/corna/me_cleaner/wiki/me_cleaner-status, last visit on 2017-05-23

5.4 Buying from another CPU manufacturer

Since Intel decided that all modern CPUs leaving their manufacturing plants should also have an embedded, uncontrollable and very powerful out-of-band processor inside their chipset, the only solution to minimize this attack surface is switching to another manufacturer.

But because Intel – together with AMD – dominate the CPU market for business and consumer devices (Intel with around 80% and AMD 20% according to Passmark[40]) this is a very problematic choice. The next big names, ARM, as a licensor of embedded CPU System on a Chip (SoC) architectures and others as manufacturers do not share the x86 instruction set and therefore are not compatible with modern operating systems and software.

There are however approaches like the OpenCores project[41] which aim to open source the development of general purpose CPUs. These attempts might be a possible solution to proprietary hardware implementations which cannot be reviewed trustfully. Of course this introduces new challenges, for example „how can I as a customer check if the open design is indeed the implemented hardware running inside my computer?“.

On a personal note: of course, if you wish to use the ME with its provided services and features, the ME is a very mature and securely designed platform. My criticism mainly aims at the disproportionately wide feature set that is included in every chipset version, both in hardware and in software. A consumer of DRM-protected content for example does not need the feature set of AMT!

5.5 Pressure Intel into changing product design

Intel as the designer, developer and manufacturer of the chipsets with an integrated Management Engine is a legal entity that must abide laws and regulations. As a competing company on a free worldwide market it also cares about investors and customers. As [EP17] put it:

> The design choice of putting a secretive, unmodifiable management chip in every computer was terrible, and leaving their customers exposed to these risks without an opt-out is an act of extreme irresponsibility.

This leads to three actors who can actively influence Intel and trigger a change in future chipset designs:

- Regulatory entities: a change in market laws, new guidelines from NGOs and standardisation organisations and product requirements from hardware vendors can introduce legal constraints Intel can operate in.
- Investors and customers of Intel can influence design decisions by using management participation and switching to other products available on the market.
- As mentioned above, open source hardware is an alternative to proprietary implementations which can be reviewed independently by the customer. Such projects often have people willing to work but not the financial resources to allow its members to work full time.

5.6 Use GPL everywhere

Intel is not the only developer of the ME internal firmware components. As seen above Intel uses licensed software like the ThreadX kernel which comes from an external developer. A solution that matters in this context would be to put all components that are licensed to Intel under the GNU General Public License (GPL). The GPL would introduce the right to view implemented and modified code to the customer. This could potentially lead to an open sourced and freely modifiable firmware – but this contrasts the closed security model the Management Engine was designed for.

[40] https://www.cpubenchmark.net/market_share.html, last visit on 2017-05-28
[41] http://opencores.org/, last visit on 2017-05-23

6 Projects and products based on AMT

6.1 Developed at Intel

Meshcommander

Meshcommander is a web interface for remote management of computers with enabled Advanced Management Technology (AMT). It features support for Keyboard-Video-Mouse (KVM), Serial over LAN (SOL), IDE redirection and access to configuration settings and logs. It replaces the Manageability Developer Tool Kit (MDTK).[42]

MeshCentral

MeshCentral 2[43] is an open source NodeJS package[44] that runs on a host computer with Node installed and lets other computers with AMT enabled connect to this control machine. It also features the full control set like Meshcommander (remote management, configuration) and is targeted at administrators. Ylian Saint-Hilaire, co-author of [KGS09], seems to be one of the main developers of this project.[45][46][47][48]

6.2 Open sourcing the AMT

Parts of earlier AMT components were open sourced and published on SourceForge.[49] The developers Jerry Yu, Anas Nashif[50] and Tomas Winkler[51] were most likely Intel developers. The project received only one recent update since 2008, which was a modification of the Local Manageability Service (LMS) in January 2017.

6.3 Private companies, cloud services

Since the AMT is made for businesses with their own IT departments the control of assets is in the hand of the hardware owner. This means that the customer buying Intel vPro enabled hardware is able to configure AMT to connect to his own servers. The anti theft technology can be used to track and remotely disable a computer with AMT enabled. This feature is used by companies which offer this service to non-business end users:

- SafeFrontier. „Remotely Repair, Track & Manage Computers With Missing or Corrupt Operating System or When Computer is Off“ (last update 2013)[52]
- „LoJack for Laptops“ (earlier „Computrace“) is a BIOS extension (which is deployed by the hardware vendor) that geotracks hardware assets and offers remote lock and wipe.[53] Computrace was target of successful attacks and could be modified to work as a rootkit.[54]

[42] http://www.meshcommander.com/, last visit on 2017-05-11
[43] https://twitter.com/MeshCentral, last visit on 2017-05-11
[44] https://registry.npmjs.org/meshcentral/-/meshcentral-0.0.4-c.tgz, last visit on 2017-05-11
[45] https://www.youtube.com/user/ysainthilaire/videos, last visit on 2017-05-11
[46] https://www.youtube.com/user/AAlliieennX/videos, last visit on 2017-05-11
[47] http://www.intel.com/software/ylian, last visit on 2017-05-11
[48] https://www.npmjs.com/~ysainthilaire, last visit on 2017-05-11
[49] https://sourceforge.net/projects/openamt/, last visit on 2017-05-10
[50] https://twitter.com/anasnashif, last visit on 2017-05-11
[51] https://lkml.org/lkml/2017/3/14/275, last visit on 2017-05-11
[52] http://safefrontier.com/, last visit on 2017-05-28
[53] https://lojack.absolute.com/en/store/lojack/laptops, last visit on 2017-05-28
[54] https://exploiting.wordpress.com/2009/09/11/138/, last visit on 2017-05-28

References

[Bar] Eddie Barcellos. *Disabling Intel AMT on Windows (and a simpler CVE-2017-5689 Mitigation Guide)*. URL: https://mattermedia.com/blog/disabling-intel-amt/ (visited on 11/05/2017).

[Del] Dell. *IPMI Configuration on Ninth-Generation Dell PowerEdge Servers*. URL: http://www.dell.com/downloads/global/power/ps3q06-20050317-Zhuo.pdf (visited on 11/05/2017).

[Doc17] Cory Doctorow. *Intel declared war on general purpose computing and lost, so now all our computers are broken*. 9th May 2017. URL: https://boingboing.net/2017/05/09/management-engine.html (visited on 11/05/2017).

[Ent] Hewlett Packard Enterprise. *HPE iLO IPMI User Guide*. URL: http://www.hpe.com/support/ilo4_ipmi_ug_en (visited on 11/05/2017).

[EP17] Peter Eckersley and Erica Portnoy. *Intel's Management Engine is a security hazard, and users need a way to disable it*. 8th May 2017. URL: https://www.eff.org/deeplinks/2017/05/intels-management-engine-security-hazard-and-users-need-way-disable-it (visited on 11/05/2017).

[Gar17] Matthew Garrett. *Intel AMT on wireless networks*. 9th May 2017. URL: http://mjg59.dreamwidth.org/48837.html (visited on 11/05/2017).

[Hof13] Gael Hofemeier. *Intel Active Management Technology (Intel AMT) Start Here Guide (Intel AMT 9.0)*. 2013. URL: https://software.intel.com/sites/default/files/article/393789/amt-9-start-here-guide.pdf (visited on 11/05/2017).

[KGS09] Arvind Kumar, Purushottam Goel and Ylian Saint-Hilaire. *Active Platform Management Demystified - Unleashing the power of Intel vPro Technology*. Intel Press, 2009.

[per16] persmule. *Neutralize ME firmware on SandyBridge and IvyBridge platforms*. 17th Nov. 2016. URL: https://hardenedlinux.github.io/firmware/2016/11/17/neutralize_ME_firmware_on_sandybridge_and_ivybridge.html (visited on 11/05/2017).

[Rua14] Xiaoyu Ruan. *Platform Embedded Security Technology Revealed*. Apress, 2014. ISBN: 978-1-4302-6571-9.

[Rut15] Joanna Rutkowska. *Intel x86 considered harmful*. 27th Oct. 2015. URL: https://blog.invisiblethings.org/2015/10/27/x86_harmful.html (visited on 04/05/2017).

[SGE17] Dmitry Sklyarov, Maxim Goryachy and Mark Ermolov. *Intel ME: The Way of the Static Analysis*. 2017. URL: https://www.troopers.de/downloads/troopers17/TR17_ME11_Static.pdf (visited on 23/05/2017).

[Sko14] Igor Skochinsky. *Intel ME Secrets – Hidden code in your chipset and how to discover what exactly it does*. 2014. URL: https://recon.cx/2014/slides/Recon%5C%202014%5C%20Skochinsky.pdf (visited on 11/05/2017).

[TW09] Alexander Tereshkin and Rafal Wojtczuk. *Introducing Ring -3 Rootkits*. 2009.

[Ver10] Vassilios Ververis. *Security Evaluation of Intel's Active Management Technology*. KTH Information and Communication Technology, 2010.

[Zam16] Damien Zammit. *Intel x86s hide another CPU that can take over your machine (you can't audit it)*. 15th June 2016. URL: http://boingboing.net/2016/06/15/intel-x86-processors-ship-with.html (visited on 11/05/2017).

List of Figures

Glossary

applet Very small application written for a very specific task. 10

measured boot Incremental integrity check. A TPM hashes the initial boot component and checks the value against a stored value. The next component (BIOS, boot loader, ...) is then hashed and the new hash combined with the old hash from before, extending the row of measured components. 9

root of trust The root of trust of the ME is the boot routine and the Firmware Signing Key (FWSK) which are burnt into the ROM. 7

task A module loaded into RAM, running as a process inside the engine. 10

verified boot Verifies the integrity of the initial boot block which then verifies the BIOS. Can work without TPM. 9

world Term used by ARM in their TrustZone implementation as an alias for „domain“. 5

Acronyms

2FA Two-factor authentication. 9

ACM Authenticated Code Module. 9
AMD Advanced Micro Devices, Inc.. 5
AMT Advanced Management Technology. 3, 4, 7, 9, 13, 17
ARM ARM processors. 5

BIOS Basic Input/Output System. 9

DAL Dynamic Application Loader. 3, 4, 10, 14
DMA direct memory access. 7, 9, 12
DMI Direct Media Interface. 6
DRAM dynamic random-access memory. 4, 9
DRM Digital Rights Management. 5, 12

EPID Enhanced Privacy Identification. 3, 12

FPF field programmable fuses. 3, 9
FPT firmware partition table. 10
FWSK Firmware Signing Key. 9, 19

GbE Gigabit Ethernet. 9, 15
GMCH Graphics and Memory Controller Hub. 6
GPL GNU General Public License. 16
GPU Graphics Processing Unit. 12

HDCP High-bandwidth Digital Content Protection. 12
HECI Host Embedded Controller Interface. 12

ICH I/O Controller Hub. 6
IDE-R IDE redirection. 9, 15
IPMI Intelligent Platform Management Interface. 5
IPT Identity Protection Technology. 3, 4, 9, 12

JHI JOM DAL Host Interface. 12
JVM Java Virtual Machine. 10

KVM Keyboard-Video-Mouse. 17

LMS Local Manageability Service. 12, 17

MCH Memory Controller Hub. 6
ME Intel Management Engine. 3, 4, 6, 13, 15, 18
MEBX ME BIOS extensions. 15
MEI Management Engine Interface. 10, 12
MPR memory protection range. 7

NIC network interface controller. 7, 9, 15

OOB out-of-band. 3, 7, 13
OS Operating System. 7, 12
OTP One Time Password. 9

PAVP protected audio video path. 3, 4, 12
PCH Platform Controller Hub. 6
PSP Platform Security Processor. 5

RNG random number generator. 7
ROM Read-Only Memory. 7
RTOS Real Time Operating System. 9

SEC Security Engine. 3
SGX Software Guard Extentions. 5
SMM System Management Mode. 14
SoC System on a Chip. 3, 5, 16
SOL Serial over LAN. 9, 13, 15, 17
SPI Serial Peripheral Interface. 6, 9
SPS Server Platform Services. 3

TCG Trusted Computing Group. 5, 12
TDT Theft Deterrence Technology. 9
TEE Trusted Execution Environment. 5, 10
TPM Trusted Platform Module. 3–5, 9, 12
TXT Trusted Execution Technology. 3, 9

UMA Unified Memory Architecture. 10

VNC Virtual Network Computing. 15