Trusted Computing: Security Challenges in Intel Trusted Execution Environment

Sean Carroll
Department of Electrical and Computer Engineering
Villanova University
Villanova, PA 19010
scarro15@villanova.edu

***Abstract*— The current state of technology is heavily built upon the concepts of security and trust. The definition of trust varies substantially, and this has resulted in varied designs in trust technologies and raised concerns for their security. In this paper, trust is initially defined to remove potential confusion. The paper proceeds with a discussion of Intel vPro technology, in the interest of creating a more secure Trusted Execution Environment (TEE). Intel Management Engine (ME) and Intel Software Guard Extensions (SGX) are highlighted as potential challenges in achieving this effort.**

***Index Terms*—Trust, Trusted Computing, Trusted Execution Environment, Secure Execution Environment, Trusted Computing Base, Intel Management Engine, Root of Trust, Digital Rights Management, Remote Attestation, Trustless Computing**

I. Introduction

It is necessary to begin with a clear understanding of what we term trusted and consider the scope of our research to be bound to. Referenced papers vary on the idea of what the objective of trust and secure design attempt to achieve, in consideration to the end-user, third-parties, and trusted entities. There are clear security implications that arise from bearing certain entities control over computing platforms to achieve a functional and sustainable platform.

Our objective is to analyze current implementations seen in industry to highlight the issues and tradeoffs that must be made to achieve the basic concept that can be unanimously agreed upon: the idea that, regardless of concerns for security and trust, a functional and sustainable design is always the goal.

Although this paper cannot be exhaustive in analysis of Trusted Environments seen in industry, we highlight specific topics that appear problematic to the interests of secure computing. Further research and brevity will be the topic of future papers and it is the hope of fostering new ideas for provable security approaches.

Section (II) explicitly terms trust and related terms to avoid confusion. Section (III) discusses challenges in Intel TEEs, specifically with Intel ME and Intel SGX. Section (IV) proposes further problems with Intel's TEE approach, with proposed ways to remediate security risks. Section (V) discusses various design schemes in platform design, and Digital Rights Management (DRM). Section (VI) includes the authors main contribution to research. Section (VII) discusses related works and Section (VIII) concludes the research.

II. Considering Trust

Trust is a unique concept. To trust oneself or other entity is to give full ability to destroy one self's current state or to allow the other entity to destroy one's current state, respectfully. If one of two interacting entities does not trust itself, we are acknowledging that both parties provide no basis for analysis, without an externality. In order to analyze trust, there is basis (the current state of oneself) that cannot declare guaranteed trust, and a fundamental assumption that oneself is trusted must be made. Logically, we must meet a balance between knowledge of identity and anonymity to communicate.

If human interaction plays a role in analysis of the topic, it brings a third-party viewpoint C if considering computer entity A, interacting with computer entity B. The system dynamic demands a trusted third entity, which poses an issue for parties attempting to be completely trustless. Zero-knowledge proofs limit sensitive data that is desired to be trusted by A, but not trusted by B, to only A. The same logic follows for B. However, there remains the task of interacting with a central point of verification by both parties, which still cannot be guaranteed. We still rely on statistical probability—definitive, guaranteed, and causal verification (which the author terms causal trust) is not feasible. Physically unclonable function (PUF) is based on the uniqueness of manufactured design, which allows us to attach an identity to an entity. Yet, the verification process still lends itself to statistical probability for assurance. Error cannot be neglected in either zero-knowledge proofs or PUF, and causal trust is desired, but logically infeasible.

It is our goal to limit the amount of trust given to the TEE, in the effort to push towards a Secure Execution Environment (SEE), acknowledging that a Secure TEE is likely be the industry compromise.

III. Challenges in Intel TEE

Our focus is rooted in a conundrum found in current secure logic that the author terms the *Insecure Sandwich.* It is evidence in industry, for example with Intel vPro Technology suite, whereby the execution environment is sandwiched between layers: closed-sourced hardware design and remote

attestation, which have the possibility of tampering with the contained environment, through higher privileges, but cannot be verified by each layer. If we are to be trustless and truly paranoid, this design is not sustainable moving forward.

There are strategies to mitigate tampering and challenge the other isolated layers, through designs such as Tails, Whonix, and Qubes OS. These platforms challenge the questioned layers with a state of temporal separation, in which tampering could be proven [3]. However, it is unrealistic to measure the tampering and pits the user into an environment with limited functionality due to separation-by-design. Open-source implementations are challenged by heavily funded and motivated, closed-source designs.

Bootstrapping on Intel x86 architecture uses a separate microkernel, and in doing so, benefits from limiting attack vectors for third-parties, but allows potential bootkits through design, and therefore, closed source designs are put into question. The proposal that a TEE can be ensured by an industry-trusted, yet not third-party verified Root of Trust (RoT) and separation environment, is questioned for its secure design and sustainability long-term.

Intel Management Engine subsystem, a technology part of blanket technologies of Intel vPro technology, raises security concerns, as every new chip since 2015 has the ability to run Intel ME and Intel Active Management Technology (AMT), even if not noted [9][10]. It is without question that Intel Software Guard Extensions (SGX) design has led researchers to several unresolved issues during hardening attempts [12].

The proposed strength during initial implementation of SGX was its ability to provide a compact and secure trusted computing base (TCB), however further development has led to increased size of enclaves, which moves away for the security ideology of the term TCB. The increase code base of Intel SGX over time raises concerns for its security, as more and more features are added within the encrypted enclave. Several entities have proposed disabling Intel ME, however, it is no longer feasible on the platform [2][4]. It appears unrealistic that growing proprietary innovations by Intel could be thwarted easily on all platforms [coreboot, OpenBIOS, etc.] without backlash to ensure its persistence.

This initially small code base of workloads in TEE inevitably creates vulnerabilities that can be exploited by attackers. Even with perfect isolation and secure architecture, we cannot prevent attacks from growing buggy code [8]. It is advantageous to seek other methods to employ trustless and provably secure execution environments, rather than sustain the model of TEE that is currently provided in industry.

Ring -3 bootkits have been demonstrated by using Intel ME [8]. Previously, SMM-based rootkits have been used as cyberweapons by the National Security Agency (NSA) [8]. In trend, we declare that running trusted coded in TEE raises a potential security concern. Within this mode we cannot currently:

- Verify the executed code given an identified open-source framework
- Verify the defense within the TEE
- Verify methods detecting compromised TEE

The concept of isolating hardware, and then due to its isolation, the isolated firmware are hard to verify and detect tampering. Patching low-level firmware form a hacked state by way of high-level software is not a reasonably secure procedure if we cannot snapshot the state of hardware form a remote high-level, isolated environment that logs all bands of data transmission. That is, for the reason that the internal low-level isolated environment are in emphasis—isolated—the internal components and operation bypass and neglect audit by the kernel tied to the user operating system (OS).

There is a clear interest within the system security community in the challenge of building secure environments upon hardware and firmware treated as untrusted [2][3]. Investigation into Tails OS has revealed irregularities. Testing has demonstrated in our experimentation unusual separations on the 1st install disk, which might demonstrate adversary interest in attacking base platform circumvention of control.

A topic of interest in SEE is the whitelisting approach used by Intel. The whitelisting approach prevents untrusted code from, theoretically, being executing on the machine. Allowing all code trusted or untrusted, to be executed on the machine, provides the advantage that we a user can realize a threat without doubt and obscurity, meaning a lack in need to trust Intel in this approach, but a moderating entity of choice.

Whitelisting ensures no unauthorized firmware flash modifications. The main problem with static-based trusted boot is related to the ever-long and complex chain of trust (CoT), which includes the entire BIOS with all its drivers and their subsystems, and also, the OS loaders. Intel Trusted Execution Technology (TXT) attempted to shorten this chain by excluding the BIOS, boot, and OS loader from the TCB. Whitelisting appears to create poor trust anchor design, as [2] states that Intel could provide whitelisting to any entity of choice, without our knowledge or consent.

The author outlines an issue where Intel firmware verifies authorized access, given a reference lookup that it already has access to, but the other side [OS, user, client] does not, inherently making the process insecure by trusting itself, but no other entity. Mathematically, the concept of zero knowledge proofs, or Physically Unclonable functions (PUF) appear to be headed in the proper direction, in which provable verification of security does not necessarily require a trusted environment or trusted interaction.

The trusted boot scheme, or dual-execution environment requires every part [BIOS, GRUB, OS kernel, etc.] to be perfectly secure to operate securely. Conceptually, in a large chain of trust, there are many touchpoints in between processes. More touchpoints increase risk of security vulnerabilities. Intel is dual-in-part component: in which intel is responsible, in the above-considered trusted boot scheme, for not providing backdoor access, and not making the mistake of coding a design flaw. The overly trusted entity raises the need for provability, and transparency in trust added to this design to absolve security concerns.

Our research highlights a vulnerability in Intel design that has never been successfully addressed. We consider case; the

BIOS is hacked. It is wise to initially not trust the BIOS initially, however, let us proceed in this case. Intel TXT, or blanket technology, is patched with the SMM unit, or in technical terms: the SMI handler. This process does not work in practice with provable guarantee, for if the BIOS was pre-conditionally hacked, that hacked BIOS would boot the SMM. Patching a firmware is proven to be a very difficult task if the system's trusted base is hacked [8][10]. Furthermore, Future changes to the Intel SMI Transfer Monitor (STM) are not provisioned by Intel, making the vulnerability persistent [2].

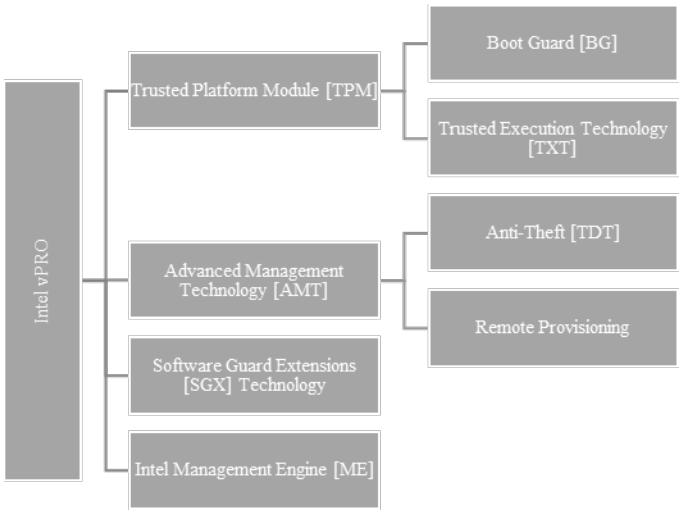

Fig 1. Topology of Intel vPro Technology Suite. This not inclusive of all technologies and only includes subsystems technologies of interest to our research.

IV. TEE Design Approach

Let us point out that SGX is thought to be implemented with the help of Intel ME, leading one to believe that in order to implement a SEE, we must protect against Intel ME. Many researchers advocate open-source BIOS, such as coreboot, OpenBIOS, OpenSGX, or seemingly better, open-sourcing closed-source proprietary Intel firmware related to their trusted boot platform.

However, open-sourcing does not appear to be a more secure solution. Research has shown open-source BIOS to be possibly less secure due to its lack of maintenance and review [14]. Furthermore, Google Chromebooks have open-source embedded controllers (EC), however, there still does not exist a reliable way for the average user to easily verify and ensure its authenticity. Like Qubes OS, the Chromebook pits a desired-to-be untrusted entity into an environment in which its trustworthiness could be measured but is seldom measured.

Open-sourcing Intel code and hardware proprietary information does nothing to ensure security if a provably secure framework cannot be easily implemented. Tamper-evident devices, do not address security concerns if a form of transparency and verification is not provided with reasonable ease-of-use.

The concept of owning physical hardware, without the ability to boot without trusting the provider, is effectively the same as renting the device. Although not encompassing all security concerns, the combination of stateful storage and hardware places the user in a situation where they are persistently vulnerable to attack, regardless of the realization.

The author has previously examined a Mac OS X system that appeared to be tampered with, at the BIOS level. The BIOS could not be flashed, and new security concerns now prevent client laptops from installing OS via external hardware devices. The CMOS, BIOS, and other EC chips had rendered the device unusable.

The risks in trusting the supply chain is of equal concern to the chip manufacture and the end-user or purchasing client-base. Intel has acknowledged that an open-source plan is in slow integration, yet also claiming that business drivers keep the memory initialization logic closed [16]. Intel has received the argument that they are not making a platform under minPlatform philosophy [16].

The IP protected initialization in a well-defined blob does not appear to contradict an open-development environment, as their security philosophy appears also to be a defense mechanism. It appears virtualization is the next step, as renting hardware is now becoming almost the same concept.

The internet began as a US government-funded project for research. There have been, and remain, intellectual safeguards to protect hardware and software for the interest of the US government, that interested often being ensuring the safety of the American people.

There are generally two types of centralized security methods commonly seen in industry. One involves hiding proprietary knowledge and protecting for its escape, the trade secret approach. Closed-designed systems and hardware rely on the idea that others will not be able to, one, know about a design, but two, how to replicate and exploit it. This method could be conceptualized as a fish without its school in a large ocean. The potential threats will likely follow the school, thus ensuring survival.

The second method involves open-sourcing information, design, and implementation. This method enables more trust and transparency for the end-user, that is, if the end-user verifies the process themselves. Conceptually, a fish with its school in a large ocean can benefit if threats can be identified. However, if they are not identified by the fish or school, the potential threats are more likely to occur than if ignorant and isolated.

Depending on how each method manifest; the difficulty is not a single-factor decision. Of these two cybersecurity methods, commerce, national government interest, fundamental platform sustainability and growth, among other aspects, come into consideration when criticizing the practice of either method.

V. Platform Design and DRM

In platform design, the architects must consider whether a TEE is one that is desired. Often in product design or software engineering, a group is employed with designing a solution that is desired, without proper prerequisite information or a defined

solution in mind. This makes this procedure a good practice, but not a fool-proof solution to the problem of trust in SEE.

A replay attack on persistence storage could prove evidence of tampering activity, and thus it is unlikely that host memory pages are being access by Intel. However, it is better to assume the risk is existing, and very complex, yet not active.

Virtualization does not absolve risk, nor does emulation. The idea of separation through chaining a connection farther away form a threat could make it more secure, but not necessarily. In consideration of Tails, there is a finite limit of separation that remediates risk, by making the challenge more difficult, yet pragmatically the threat could be an externality out of user scope.

Opponents of TCB argue that DRM is no longer an issue, but an excuse for operating proprietary code to verify system tampering on remote devices. The argument for implementing Intel SGX in 2015 was heavily weighted on the need to enforce DRM policies, and this platform providing a mechanism to curb piracy [14]. The streaming of services has drastically affected the increase in piracy, in which it has gone down [14].

The argument that future piracy rates could potentially increase is supported, and if streaming or renting of services becomes too costly, there is future concern and need for enforcing DRM policies [16][14]. However, industry has not implemented the combination of technologies provided within Intel as expected, but in use of other tools and mechanisms.

The current Intel software is believed to include Intel ME, Boot Guard, and the GPU to decrypt content and ensure secure playback. DRM policies ensure verification of ownership and licensing [10][14]. There is further concern in buying preconfigured hardware, in specific, refurbished technologies, for the potential lack of control the new user has been granted.

The reconfigurations and specifications of the previous owner could affect the new user. This outlines great risk, especially if anti-theft software in BIOS is not fully wiped. The author has tested these ideas on multiple computers and found difficulty with persistence of anti-theft technologies.

VI. Main Contribution

With exception to my discussion of TEE above, a major part of my research included the modification of computer hardware and firmware to limit Intel ME functionality. This section will be dedicated to my research into Intel ME partial disablement

A. Design Decisions

Several design considerations were taken into account when attempting to limit Intel ME firmware functionality. We desired an Intel motherboard with Haswell microarchitecture, and Intel ME version 9.X or lower, as this would create more fruitful discussion, if the experiment was efficacious. A motherboard from an ASUS G550JK was used for experimental testing. The motherboard had specifications:

- Intel Core i7 4th Gen 4710HQ (Haswell)
- Intel HM86 Chipset (Lynx Point)
- Intel ME v9.0.30.1482 (Generation 2)

The motherboard was sent to a third-party to replace the BIOS chip and verify the functionality and integrity of all parts on the chip. The CPU and all components were re-soldered, reassembled, and verified for functionality.

The default BIOS firmware was installed, along with Ubuntu 16.04 LTS. As expected from research, the computer immediately turned off after about 30 minutes after boot. Several other operating systems, including Windows 7 64-bit and Tails OS were installed to test this behavior. In all preliminary trials, the computer shut off around 30 minutes after boot.

The original firmware image was extracted using a USB CH341A Programmer and a DIP clip, with provided software. Two binaries were extracted, and both were verified to be the same, ensuring the quality of the binary.

A python script called me_cleaner was used in order to modify the Intel ME firmware image and prevent its interaction with the system [6]. The firmware is edited as to only allow Intel ME to be active at boot time. The process is illustrated in Figure 2.

After retrieving the modified binary file, the default OEM software flashing tool was used to flash the modified firmware. Booting the computer no longer shut off 30 minutes after boot. It was concluded that Intel ME played a role in the shut-down process.

B. Discussion of Results

This experiment was designed to demonstrate one way that Intel ME functionality could be limited. In this experiment, we chose the design specifications to highlight an important point that must be noted. The specific motherboard included a Haswell microarchitecture, and although we could prevent Intel ME from functioning all the time, the Intel ME binary blob is still required during the boot stage. Earlier processors can fully disable Intel ME during every state of execution. Newer processors demand Intel ME functionality always, as it cannot be disabled by design.

In the progression above, it is important to note that future Intel Designs have limited opensource implementations and we are now reliant on closed-sourced firmware.

The Intel TDT is thought to cause the immediate shutdown after 30 minutes. However, Intel ME must have played a role in the execution and or verification process that caused failed verification, and therefore, shutdown. Overall, this experiment attempted to expose the resistance end-users are facing from Intel when attempting to personally limit trust and become more secure.

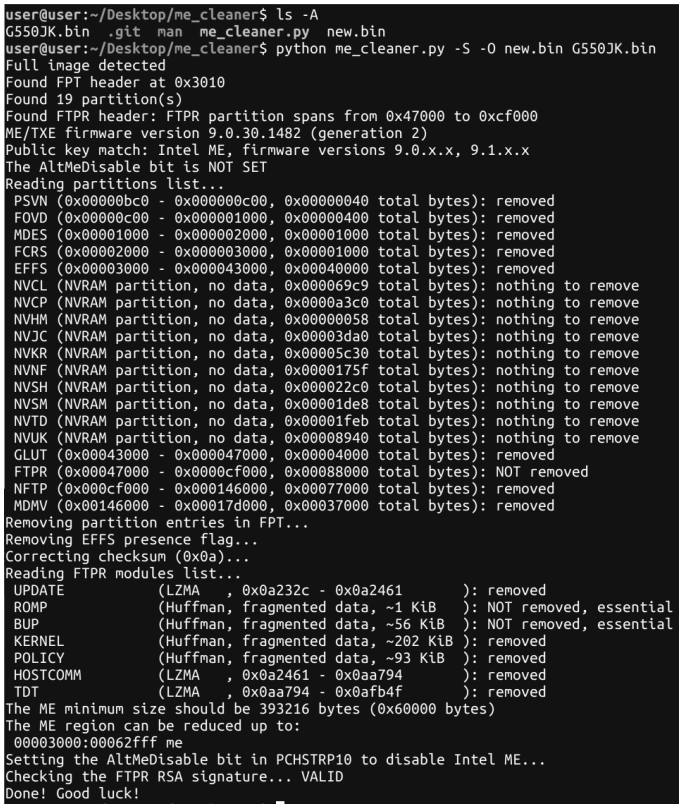

Fig 2. Python script me_cleaner.py run in terminal window on Ubuntu 18.04.1 LTS to remove Intel ME access.

VII. Related Work

Current research on Intel TEE design varies drastically, based on the objectives of the author. In the interest of security, it is wise to attempt to design a SEE, over an TEE. Intel's approach to security is to assume that that the user should not be trusted, and only Intel should be trusted. This has created a great amount of research into trustless computing. Most referenced works highlight potential attack vectors and vulnerabilities in Intel's current design. Some argue that Intel has discredited themselves as a trusted entity in computing [9].

One researcher held that Intel could not have provided a widespread trusted platform adoption, without their design approaches [12]. Our statement in the introduction acknowledges that functionality and adoption is often favored over security.

Rutkowska [2][3] outlines countless design flaws with Intel x86, and these papers show the preamble to the development of the Qubes OS. Her distaste and intolerance for trust sets a paranoid tone to the research, which benefits in being extensive in consideration of vulnerabilities. However, my contribution highlights that there is no good final security verification for her stateless design proposal. Furthermore, a lack of detailed reasons for discredidation of the Tails OS platform resulted in the author testing Tails OS, which found that installation demands two USBs, with security abnormalities found in the first USB. The Qubes or Whonix have not been tested in our research.

There appeared to be a drastic change in architecture technologies starting in 2015, and research before 2013 does not include any references to the current Intel SGX Technology. This is considered in our research.

Due to the rapid changes happening in Intel's design platforms, it appears very difficult for researchers to stay current. One researcher noted that the Intel highly complex integrated security architectures (ISA), alongside the large amount of knowledge required in many security areas, clearly prevented adoption of the technology [12]. This complexity could be a security measure by Intel. Meng argues that the complexity of security architecture causes vulnerabilities, not security [11].

Due to the extensive coverage in referenced works on the attacks on Intel ME, Intel SGX, and Intel SMM, the author chose not to include them in previous discussion.

Researchers have highlighted several issues with the security and complexity of Intel SGX [4][8][11]. Hardening Intel SGX has appeared to be a challenge [12]. Side-channel attacks, timing attacks, and further Intel SGX enclave vulnerabilities are reported [8][11]. Ring -3 rootkits in Intel ME from Intel AMT, the DAGGER attack on Intel ME, and SMM-based rootkits used by the NSA evidence the security threats to Intel's TEE [8]. Furthermore CVE-2017-5689 presented in [7] outlines an attack where Industrial Control Systems (ICS) were susceptible to the vulnerability.

It is important to highlight some bias and inaccuracies in research that were found. These were taken into consideration during the author's research. Ruan [14] claims that Intel ME isolation techniques make it immune to vulnerabilities and ideal for DRM. Patakay [10] references Ruan [14] for research and comments that the platform is built upon the insecure Intel ME subsystem.

D. Eek and B. Rothstein [13] discuss the relationship between vertical (political) and horizontal (social) trust and their research is considered important to the author's discussion of trust. The results of their research reveal that it is more advantageous to increase vertical trust, as this causes an increase in horizontal (social) trust [13]. Social trust is often seen as increasing social capital, and thus, social well-being. If citizens view authorities as untrustworthy, such as Intel, it has similarly negative effects on horizontal trust [13]. Furthermore, in a trustless state, vertical trust should be emphasized to increase horizontal trust [13]. In the scope of this research, trust is decreased in Intel over security concerns, causing decreased trusted in horizontal entities. However, as [13] has shown, it is more advantageous for both the end-users of Intel's products and for Intel to makes strides in assuring their trust. Strides towards verifiable and transparent execution in Intel TEEs will ensure a more trustworthy, secure, and sustainable model for long-term social and capital growth.

VIII. Conclusions

In this paper, we have discussed the growing trends in Intel vPro technology, for the interest in improving the security of

TEEs. We outlined and defined the unique concept of trust and its implications for secure execution environments. Furthermore, we highlighted various security concerns pertaining to Intel ME. We also discussed Intel SGX's issue with growing enclaves and their divergence from minPlatform design. We demonstrated an experiment that modifies BIOS firmware to restrict the access Intel ME under certain environments. Lastly, we discussed related works to our research paper.

It appears that there is no simple solution to solving the problem of security in trusted computing. Fully open-source design does not solve security issues raised by Intel's closed-source firmware implementations. However, steps to increase transparency, verifiability, and auditability in Intel's Trusted Computing technologies, as well as in other trusted hardware implementations, would create drastically more secure, and trustworthy TEEs.

REFERENCES

[1] V. Bashun, A. Sergeev, V. Minchenkov, and A. Yakovlev, "Too young to be secure: Analysis of UEFI threats and vulnerabilities," in *14th Conference of Open Innovation Association FRUCT*, 2013, pp. 16–24.

[2] J. Rutkowska, "Intel x86 considered harmful," p. 56, 2015.

[3] J. Rutkowska, "State considered harmful," 2015.

[4] C. Shepherd *et al.*, "Secure and Trusted Execution: Past, Present, and Future - A Critical Review in the Context of the Internet of Things and Cyber-Physical Systems," in *2016 IEEE Trustcom/BigDataSE/ISPA*, Tianjin, China, 2016, pp. 168–177.

[5] A. Y. Chernov, A. S. Konoplev, and D. V. Reshetov, "The task of building a trusted computing environment on the Intel hardware platform," *Automatic Control and Computer Sciences*, vol. 51, no. 8, pp. 844–847, Dec. 2017.

[6] M. Ermolov and M. Goryachy, "Disabling Intel ME 11 Via Undocumented Mode," *Positive Technologies*, 2017.

[7] D. Evdokimov, "Intel AMT vulnerability. Life after CVE-2017-5689," *Black Hat USA 2017*, pp. 1–27, 2017.

[8] Z. Ning, F. Zhang, W. Shi, and W. Shi, "Position Paper: Challenges Towards Securing Hardware-assisted Execution Environments," in *Proceedings of the Hardware and Architectural Support for Security and Privacy on ZZZ - HASP '17*, Toronto, ON, Canada, 2017, pp. 1–8.

[9] A. Ogolyuk, A. Sheglov, and K. Sheglov, "UEFI BIOS and Intel Management Engine Attack Vectors and Vulnerabilities," in *Proceedings of the 20th Conference of Open Innovations Association FRUCT*, 2017, vol. 776, pp. 657–662.

[10] D. Pataky, "Intel Management Engine," p. 19, 2017.

[11] D. Meng *et al.*, "Security-first architecture: deploying physically isolated active security processors for safeguarding the future of computing," *Cybersecurity*, vol. 1, no. 1, Dec. 2018.

[12] M. Plauth, F. Teschke, D. Richter, and A. Polze, "Hardening Application Security Using Intel SGX," in *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Lisbon, 2018, pp. 375–380.

[13] D. Eek and B. Rothstein, "Exploring a Causal Relationship between Vertical and Horizontal Trust," 2006.

[14] X. Ruan, Platform Embedded Security Technology Revealed: Safeguarding the Future of Computing with Intel Embedded Security and Management Engine. Apress, 2014.

[15] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted Execution Environment: What It is, and What It is Not," in 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 2015, pp. 57–64.

[16] V. Zimmer, "Open-Source Host Firmware Directions," Platform Security Summit, 2018