IOActive®

Research-fueled Security Services

\ WHITE PAPER \

Cross-Platform Feature Comparison

Commissioned by Intel
IOActive Research

May 2021

Contents

Abstract

For an Intel-commissioned study, IOActive compared security-related technologies from both 11th Gen Intel Core vPro mobile processors and AMD Ryzen PRO 4000 series mobile processors. Our comparison was based on a set of objectives bundled into five categories: Below the OS, Platform Update, Trusted Execution, Advanced Threat Protection, and Crypto Extension.

Based on our research, we conclude that AMD offers no corresponding technologies in the Below the OS, Platform Update, Advanced Threat Protection, or Crypto Extension categories, while Intel offers features in all of these categories. Intel and AMD have equivalent capabilities in the Trusted Execution category.

Disclosure

The research presented in this white paper was funded in part by Intel Corporation.

Management Summary

In this document, IOActive presents:

- A comparison of security features provided by the 11th Gen Intel Core vPro mobile processors and AMD's Ryzen PRO 4000 series mobile processors, as well as highlights from current academic research where applicable
- Test results of selected tests cases for Intel's technology

IOActive has compared security related technologies from both Intel and AMD using the 11th Gen Intel Core vPro mobile processors and Ryzen PRO 4000 series mobile (AMD) CPUs. Our comparison is based on a set of objectives bundled into five categories: Below the OS, Platform Update, Trusted Execution, Advanced Threat Protection (ATP), and Crypto Extension.

From our research, we conclude that AMD offers no technologies in the ATP or Platform Update categories. Intel offers BIOS Guard, Firmware Update Restart, Intel Control-flow Enforcement Technology (Intel CET), and Intel Threat Detection Technology (Intel TDT) in the ATP category. In the Below the OS category, AMD has no corresponding technology to Intel System Security Report. In the Crypto Extension category, AMD has no corresponding technology to Intel's AVX512-variant of AES. Based on our research, Intel and AMD have equivalent capabilities in the Trusted Execution category.

The following technologies are the most impactful platform differentiators for security:

1. Intel CET
2. Intel TDT

Additionally, the composites of the security technologies discussed in this document offer a compounded value that is greater than the sum of the parts.

Our research team ran a series of tests for Intel TDT, based on install and executable instructions provided by the Intel TDT team. The tests aimed to detect a curated selection of samples of cryptominers and known ransomware in various environments and on different platforms.

The test results show a detection rate of 100% for ransomware and 100% for cryptominers by Intel TDT. Also, to better mimic threats that are increasingly obfuscating in virtual machines (VMs), we ran comparisons to popular anti-virus (AV) software that is not enabled for CPU-based threat detection. In these cases, Intel TDT was able to detect 75% of obfuscated cryptominers, compared to 0% by AV software, which has a lack of visibility into these types of attacks due to its typical deployment in the host OS. Note: Intel TDT is not a standalone AV or EDR package; it is intended to integrate into these solutions to augment and improve threat detection efficacy.

Technical Summary

The Intel Technologies and AMD Technologies sections of this document quote from publicly available documents of both vendors and list particular security technologies. The Detailed Features Comparison section explains the commonalities and differences of the corresponding technologies from both vendors. IOActive developed a security feature model and used it to compare the two subject platforms as described in the Model and Comparison subsection.

The Intel Threat Detection Technology Tests section contains a detailed description of the setup for the tests for Intel TDT, the particular samples of ransomware and cryptominers that were used as test cases for the various tests, and the test results.

Model and Comparison

IOActive's approach for a comparison of features across vendor platforms started with the formulation of a security model. Our model consists of a carefully selected list of security objectives, bundled into categories. The objectives define goals and properties that provide security benefits to the customer. The categories relate to different execution stages of the CPU.

The following paragraphs define the categories and their objectives, list technologies or building blocks which can be used to achieve the objectives, and compare the corresponding technologies implemented by Intel and AMD.

The model we present here is not exhaustive. It is missing a complete inventory of use-cases and would benefit from a thorough threat model of the security features. IOActive adapted an organizational structure of objectives provided by Intel to create this model.

Each compared feature is presented as fulfilling, fulfilling with qualifications, or not fulfilling the associated objective.

Below the OS (Platform Integrity)

With the beginning of the boot sequence, the CPU must evaluate its hardware and firmware environment and ensure transition only to a verified bootloader. This integrity validation mitigates the risk of instruction tampering or interception by an adversary as well as providing a trustworthy baseline for the remainder of boot and operation.

In particular, the objectives include:

- Identify unauthorized changes to hardware and firmware
- Prevent malicious code injection in BIOS/UEFI memory
- Ensure OS and virtual environments are running directly on platform hardware (assuming no malicious code injection)
- Enforce OEM/IHV's provided policy and report on it

The main building blocks to achieve these objectives include:

- Secure Boot
  - Root of trust/chain of trust
  - Enforcement
- Measured Boot
  - Secure storage
  - Static root of trust measurement
  - Dynamic root of trust measurement
  - Attestation

Table 1. Below the OS solutions

Intel Solution	AMD Solution
Intel Boot Guard	AMD Secure Boot
Intel Trusted Execution Technology	AMD SKINIT + Secure Loader
Intel Runtime BIOS Resilience	AMD SMM Supervisor[1]
Intel System Security Report	No equivalent feature
Intel System Resources Defense	AMD SMM Supervisor[1]

Platform Update

This category is about mechanisms for trustworthy firmware updates. Each of these protections addresses the risk of replacement of some or all of the platform-defining software and firmware with malicious components.

- Objectives
  - Update firmware code with integrity check
  - Downgrade protection
  - Focus on BIOS for most recent and secure updates
- Building Blocks
  - Provide new environment so OEMs can perform more flexible and modular updates securely
  - Utilizes UEFI capsule architecture for driving better firmware updates

[1] As of publication, only a brief high-level description of this feature was available. The authors are provisionally accepting the assertions in this description.

Table 2. Platform Update solutions

Intel Solution	AMD Solution
Intel Firmware Guard	No equivalent feature[2]
Intel BIOS Guard	No equivalent feature[2]

[2] https://www.amd.com/system/files/documents/guardmi-infographic.pdf equates AMD secure boot to both of these Intel features, but this is not substantiated by public documentation, such as https://www.amd.com/system/files/TechDocs/40332.pdf

Trusted Execution/Application and OS Protection

This category includes objectives for protection and prevention mechanisms to ensure a trustworthy runtime environment. Building on the integrity measures above, these elements enhance the stability and safety of the platform's operation.

- Objectives
  - Prevent memory corruption and tampering attacks
  - Protect sensitive data from unauthorized access
  - Protect data and virtualized containers with hardware-enforced isolation and encryption
  - Improve performance of virtualized security workloads
- Building Blocks/Hypervisor Support
  - Virtualization instructions
    - VM extensions
    - Nested virtualization
  - Virtual I/O
  - Virtual interrupts
  - Memory protection/encryption
  - Hardware-based encryption and random number generation

Table 3. Trusted Execution solutions

Intel Solution	AMD Solution
Intel Virtualization Extensions (VT-x)	AMD-V
Intel Virtualization Technology for Directed I/O (VT-d)	AMD-Vi
Intel APIC Virtualization	Advanced Virtual Interrupt Controller (AVIC)
Intel Mode Based Execution Control	AMD-V with GMET
Intel Control-flow Enforcement Technology	Not implemented in Ryzen PRO 4000
Intel Total Memory Encryption	AMD SEV/SME[3]

Advanced Threat Protection

The objectives in this category do not protect or prevent attacks facilitate their detection.

[3] As of Ryzen 4000 Pro Series. 5000 Pro Series adds additional capabilities.

- Objectives
  - Increase security and performance via offload to GPU dedicated security workloads
  - Leverage hardware telemetry to help detect advanced threats such as ransomware and cryptomining attacks
- Building Blocks
  - Reference code components
  - Detectors

Table 4. ATP solutions

Intel Solution	AMD Solution
Intel TDT- Security Offload to iGPU (e.g., Memory Scanning)	No equivalent feature
Intel TDT - Advanced Platform Telemetry	No equivalent feature

Crypto Extension

This category lists objectives for hardware support for crypto primitives with specific properties.

- Objectives
  - Provide hardware implementations for certified crypto primitives
  - Mitigate side channels in the implementation of crypto primitives
  - Allow crypto operations without storing key where it can get lost
  - High speed and throughput
- Building Blocks
  - Good source of randomness (rdrand)
  - Efficient crypto primitives with side channel mitigation
  - Secure key storage (TPM, PKEY)
  - Parallelizable implementation of primitives
  - Hardware-assisted AES encryption
  - Hardware-assisted AES decryption
  - Hardware-assisted AES inverse mix column transformation
  - Hardware-assisted AES created round keys with key expansion schedule
  - Cryptographically secure enhanced non-deterministic random bit generator
  - Cryptographically secure deterministic random bit generator

Table 5. Crypto Extension solutions

Intel Solution	AMD Solution
Intel Advanced Encryption Standard New Instructions (Intel AES-NI)	AMD AES-NI
Intel Secure Key	AMD RNRAND
Key Locker	No equivalent feature

Intel Technologies

The following sections contain the description of technologies from Intel, based on publicly accessible documentation. The quotes in these sections are primarily from Intel document ID 631121 version 003.[4]

Intel Virtualization Technology

Intel Virtualization Technology (Intel VT) makes a single system appear as multiple independent systems to software. This allows multiple, independent OSs to run simultaneously on a single system. Intel VT comprises technology components to support Virtualization of platforms based on Intel architecture microprocessors and chipsets

Intel VT, Intel 64, and Intel architecture (Intel VT-x) added hardware support in the processor to improve the virtualization performance and robustness. Intel VT for Directed I/O (VT-d) extends Intel VT-x by adding hardware-assisted support to improve I/O device virtualization performance.

Intel VT-x provides hardware acceleration for virtualization of IA platforms. VM Monitor (VMM) can use Intel VT-x features to provide an improved reliable virtualization platform.

The processor supports the following added new Intel VT-x features:

- **Mode-based Execute Control for EPT (MBEC)**: A mode of EPT operation which enables different controls for executability of Guest Physical Address (GPA) based on Guest specified mode (User/Supervisor) of linear address translating to the GPA. When the mode is enabled, the executability of a GPA is defined by two bits in EPT entry. One bit for accesses to user pages and other one for accesses to supervisor pages.

Intel Virtualization Technology for Directed I/O

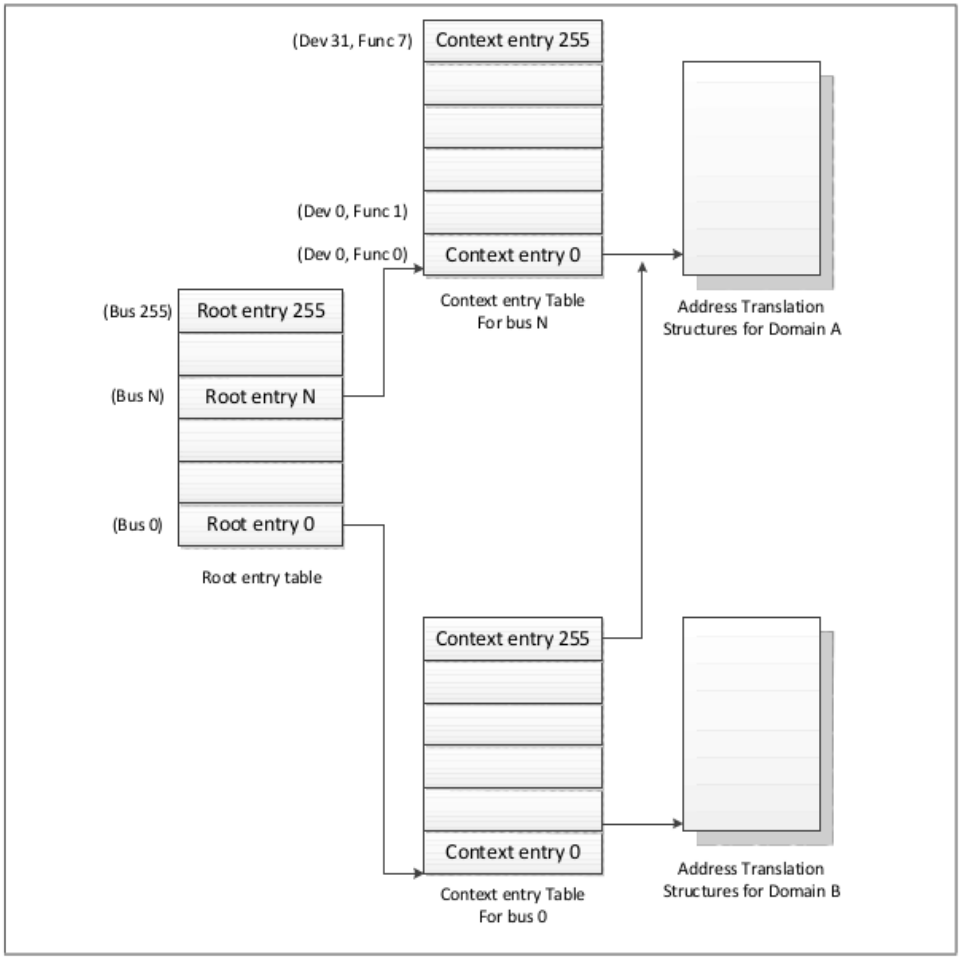
The key Intel Virtualization Technology for Directed I/O (Intel VT-d) objectives are domain-based isolation and hardware-based virtualization. A domain can be abstractly defined as an isolated environment in a platform to which a subset of host physical memory is allocated. Intel VT-d provides accelerated I/O performance for a virtualization platform and provides software with the following capabilities:

- **I/O Device Assignment and Security**: for flexibly assigning I/O devices to VMs and extending the protection and isolation properties of VMs for I/O operations.
- **DMA Remapping**: for supporting independent address translations for Direct Memory Accesses (DMA) from devices.

[4] https://cdrdv2.intel.com/v1/dl/getContent/631121

- **Interrupt Remapping**: for supporting isolation and routing of interrupts from devices and external interrupt controllers to appropriate VMs.
- **Reliability**: for recording and reporting to system software DMA and interrupt errors that may otherwise corrupt memory or impact VM isolation.

Intel VT-d accomplishes address translation by associating transaction from a given I/O device to a translation table associated with the Guest to which the device is assigned. It does this by means of the data structure in the following illustration. This table creates an association between the device's PCI Express Bus/Device/Function (B/D/F) number and the base address of a translation table. This data structure is populated by a VMM to map devices to translation tables in accordance with the device assignment restrictions above and to include a multi-level translation table (VT-d Table) that contains guest-specific address translations.



Intel APIC Virtualization Technology

Advanced Programmable Interrupt Controller Virtualization (APICv) is a collection of features that can be used to support the virtualization of interrupts and the APIC.

When APICv is enabled, the processor emulates many accesses to the APIC, tracks the state of the virtual APIC, and delivers virtual interrupts—all in VMX non-root operation without a VM exit.

The following are the VM-execution controls relevant to APICv and virtual interrupts:

- **Virtual-interrupt Delivery**: This controls enable the evaluation and delivery of pending virtual interrupts. It also enables the emulation of writes (memory-mapped or MSR-based, as enabled) to the APIC registers that control interrupt prioritization.
- **Use TPR Shadow**: This control enables emulation of accesses to the APIC's task-priority register (TPR) via CR8 and, if enabled, via the memory-mapped or MSR-based interfaces.
- **Virtualize APIC Accesses**: This control enables virtualization of memory-mapped accesses to the APIC by causing VM exits on accesses to a VMM-specified APIC-access page. Some of the other controls, if set, may cause some of these accesses to be emulated rather than causing VM exits.
- **Virtualize x2APIC Mode**: This control enables virtualization of MSR-based accesses to the APIC.
- **APIC-register Virtualization**: This control allows memory-mapped and MSR-based reads of most APIC registers (as enabled) by satisfying them from the virtual-APIC page. It directs memory-mapped writes to the APIC-access page to the virtual-APIC page, following them by VM exits for VMM emulation.

Intel Trusted Execution Technology

Intel Trusted Execution Technology (Intel TXT) defines platform-level enhancements that provide the building blocks for creating trusted platforms.

The Intel TXT platform helps to provide the authenticity of the controlling environment such that those wishing to rely on the platform can make an appropriate trust decision. The Intel TXT platform determines the identity of the controlling environment by accurately measuring and verifying the controlling software.

Another aspect of the trust decision is the ability of the platform to resist attempts to change the controlling environment. The Intel TXT platform will resist attempts by software processes to change the controlling environment or bypass the bounds set by the controlling environment.

Intel TXT is a set of extensions designed to provide a measured and controlled launch of system software that will then establish a protected environment for itself and any additional software that it may execute.

These extensions enhance two areas:

- The launching of the Measured Launched Environment (MLE)
- The protection of the MLE from potential corruption

The enhanced platform provides these launch and control interfaces using Safer Mode Extensions (SMX).

The SMX interface includes the following functions:

- Measured/Verified launch of the MLE
- Mechanisms to ensure the above measurement is protected and stored in a secure location
- Protection mechanisms that allow the MLE to control attempts to modify itself

For the above features, BIOS should test the associated capability bit before attempting to access any of the above registers.

As called out by security researchers Joanna Rutkowska and Rafal Wojtczuk in 2008-2009, "TXT is essentially useless without protection against SMM-originating attacks."[5] At the time this was considered a controversial statement, but it has turned out to be an astute one. As such TXT on its own has limited value without protecting SMM-based attacks. Intel implements this in three different technologies:

- Intel Runtime BIOS Resilience
- Intel System Resources Defense
- Intel System Resource Report

All of these (Intel TXT+IRBR+ISRD+ISRR) make up pieces of a larger story that addresses the concerns raised by Rutkowska and Wojtczuk over a decade ago. These three technologies are summarized below.

Intel Runtime BIOS Resilience

Intel IRBR helps to prevent malicious software injection into SMM and to prevent SMM being used to access trusted environment memory, by locking down the CPU paging structures used by SMM.

Intel System Resources Defense

Intel ISRD is a BIOS hardening technology that splits System Management Interrupt (SMI) handlers into Ring 3 and Ring 0 portions.

Supervisor/user paging on the smaller Ring 0 portion will enforce access policy for all the Ring 3 code with regard to the SMM state save, MSR registers, IO ports, and other registers.

[5] https://www.blackhat.com/presentations/bh-dc-09/Wojtczuk_Rutkowska/BlackHat-DC-09-Rutkowska-Attacking-Intel-TXT-slides.pdf

The Ring 0 portion can perform save/restore of register context to allow the Ring 3 section to make use of those registers without having access to the OS context or the ability to modify the OS context.

The Ring 0 portion is signed and provided by Intel. This portion is attested by the processor.

Intel System Security Report

Intel System Security Report consists of the Platform Properties Assessment Module (PPAM) which is measured by SINIT into PCR17 and signed by Intel. PPAM creates a report of hardware access policies used by SMM and will be consumed by the MLE. While PPAM re-uses some of the infrastructure that was initially created for an SMI Transfer Monitor (STM), it is not an STM, it only collects and reports information about platform configuration.

Intel Advanced Encryption Standard New Instructions (Intel AES-NI) and PCLMULQDQ

The processor supports Intel Advanced Encryption Standard New Instructions (Intel AES-NI) that are a set of Single Instruction Multiple Data (SIMD) instructions that enable fast and secure data encryption and decryption based on the Advanced Encryption Standard (AES). Intel AES-NI is valuable for a wide range of cryptographic applications, such as applications that perform bulk encryption/decryption, authentication, random number generation, and authenticated encryption. AES is broadly accepted as the standard for both government and industrial applications and is widely deployed in various protocols.

Intel AES-NI consists of six Intel SSE instructions. Four instructions, AESENC, AESENCLAST, AESDEC, and AESDELAST facilitate high-performance AES encryption and decryption. The other two, AESIMC and AESKEYGENASSIST, support the AES key expansion procedure. Note that the VEX prefix can be used with these instructions, adding vector extensions.

The processor supports the carry-less multiplication instruction, PCLMULQDQ. PCLMULQDQ is a Single Instruction Multiple Data (SIMD) instruction that computes the 128-bit carry-less multiplication of two 64-bit operands without generating and propagating carries. Carry-less multiplication is an essential processing component of several cryptographic systems and standards. Hence, accelerating carry-less multiplication can significantly contribute to achieving high-speed secure computing and communication.

Intel Secure Key

The processor supports Intel Secure Key, a software visible random number generation mechanism supported by a high-quality entropy source. This capability is available to programmers through the RDSEED and RDRAND instructions. The resultant random number generation capability is designed to comply with existing industry standards in this regard (ANSI X9.82 and NIST SP 800-90).

Some possible usages of the RDRAND instruction include cryptographic key generation as used in a variety of applications, including communication, digital signatures, secure storage, and so on.

Intel Boot Guard Technology

Intel Boot Guard technology is a part of boot integrity protection technology. Intel Boot Guard can help protect the platform boot integrity by preventing the execution of unauthorized boot blocks. With Intel Boot Guard, platform manufacturers can create boot policies such that invocation of an unauthorized (or untrusted) boot block will trigger the platform protection per the manufacturer's defined policy.

With verification based in the hardware, Intel Boot Guard extends the trust boundary of the platform boot process down to the hardware level.

Intel Boot Guard accomplishes this by:

- Providing of hardware-based Static Root of Trust for Measurement (S-RTM) and the Root of Trust for Verification (RTV) using Intel architectural components
- Providing of architectural definition for platform manufacturer Boot Policy
- Enforcing of manufacture provided Boot Policy using Intel architectural components

Benefits of this protection are that Intel Boot Guard can help maintain platform integrity by preventing re-purposing of the manufacturer’s hardware to run an unauthorized software stack.

Intel Supervisor Mode Execution Protection

Intel Supervisor Mode Execution Protection (SMEP) is a mechanism that provides the next level of system protection by blocking malicious software attacks from user mode code when the system is running in the highest privilege level. This technology helps to protect from virus attacks and unwanted code from harming the system.

Intel Supervisor Mode Access Protection

Intel Supervisor Mode Access Protection (SMAP) is a mechanism that provides next level of system protection by blocking a malicious user from tricking the OS into branching off user data. This technology shuts down very popular attack vectors against OSs.

Intel Secure Hash Algorithm Extensions (Intel SHA Extensions)Intel Secure Hash Algorithm Extensions (Intel SHA Extensions)is one of the most commonly employed cryptographic algorithms. Primary usages of Intel SHA Extensions include data integrity, message authentication, digital signatures, and data de-duplication. As the pervasive use of security solutions continues to grow, Intel SHA Extensions can be seen in more applications now than ever. The Intel SHA Extensions are designed to improve the performance of these compute-intensive algorithms on Intel architecture-based processors.

The Intel SHA Extensions are a family of seven instructions based on the Intel Streaming SIMD Extensions (Intel SSE)that are used together to accelerate the performance of processing SHA-1 and SHA-256 on Intel architecture-based processors. Given the growing importance of Intel SHA Extensions in our everyday computing devices, the new instructions are designed to provide a needed boost of performance to hashing a single buffer of data. The performance benefits will not only help improve responsiveness and lower power consumption for a given application, but they may also enable developers to adopt Intel SHA Extensions in new applications to protect data while delivering to their user experience goals. The instructions are defined in a way that simplifies their mapping into the algorithm processing flow of most software libraries, thus enabling easier development.

User Mode Instruction Prevention

User Mode Instruction Prevention (UMIP) provides additional hardening capability to the OS kernel by allowing certain instructions to execute only in supervisor mode (Ring 0).

If the OS opts to use UMIP, the following instructions are enforced to run in supervisor mode:

- SGDT - Store the GDTR register value
- SIDT - Store the IDTR register value
- SLDT - Store the LDTR register value
- SMSW - Store Machine Status Word
- STR - Store the TR register value

An attempt at such execution in user mode causes a general protection exception (#GP).

Intel Total Memory Encryption (Intel TME)

Intel Total Memory Encryption (Intel TME) technology encrypts the platform's entire memory with a single key. Intel TME, when enabled via BIOS configuration, ensures that all memory accessed from the Intel processor is encrypted.

Intel TME encrypts memory accesses using the AES XTS algorithm with 128-bit keys. The encryption key used for memory encryption is generated using a hardened random number generator in the processor and is not exposed to software.

Data in-memory and on the external memory buses is encrypted and exists in plain text only inside the processor. This allows existing software to operate without any modification while protecting memory using Intel TME. Intel TME does not protect memory from modifications.

Intel TME allows the BIOS to specify a physical address range to remain unencrypted. Software running on an Intel TME enabled system has full visibility into all portions of memory that are configured to be unencrypted by reading a configuration register in the processor.

Intel TME is aimed at providing protection from tampering attacks such as so called 'Cold Boot' attacks.

Intel Control-flow Enforcement Technology (Intel CET)

Return-oriented Programming (ROP), and similarly CALL/JMP-oriented programming (COP/JOP), has been the prevalent attack methodology for stealth exploit writers targeting vulnerabilities in programs.

Intel Control-flow Enforcement Technology (Intel CET) provides the following components to defend against ROP/JOP style control-flow subversion attacks:

Shadow Stack

A shadow stack is a second stack for the program that is used exclusively for control transfer operations. This stack is separate from the data stack and can be enabled for operation individually in user mode or supervisor mode.

The shadow stack is protected from tampering through extensions to the page table protections such that regular store instructions cannot modify the contents of the shadow stack. To provide this protection the page table protections are extended to support an additional attribute for pages to mark them as “Shadow Stack” pages. When shadow stacks are enabled, control transfer instructions/flows such as near call, far call, call to interrupt/exception handlers, and so on. store their return addresses to the shadow stack. The RET instruction pops the return address from both stacks and compares them. If the return addresses from the two stacks do not match, the processor signals a control protection exception (#CP). Stores from instructions such as MOV, XSAVE, and so on are not allowed to the shadow stack.

Indirect Branch Tracking

The ENDBR32 and ENDBR64 (collectively ENDBRANCH) are two new instructions that are used to mark valid indirect CALL/JMP target locations in the program. This instruction is a NOP on legacy processors for backward compatibility.

When IBT is enabled, any indirect JMP or CALL instruction must land on one of the ENDBRACH instructions or a control protection fault (#CP) will be triggered. This ensures that control flow will follow what the developer (and compiler) intended.

KeyLocker Technology

KeyLocker technology provides a method to allow using long-term keys without exposing them. This protects against vulnerabilities when keys can be exploited and used to attack encrypted data such as disk drives.

An instruction (LOADIWKEY) allows the OS to load a random wrapping value (IWKey). The IWKey can be backed up and restored by the OS to/from the PCH in a secure manner.

The software can wrap its own key via the ENCODEKEY instruction and receive a handle. The handle is used with the AES*KL instructions to handle encrypt and decrypt operations. Once a handle is obtained, the software can delete the original key from memory

AMD Technologies

The following sections describe security and virtualization related technologies from AMD, quoted from publicly accessible documents.

AMD Secure Processor Technology

"Under the memory encryption extensions defined here, each SEV-enabled guest VM is associated with a memory encryption key, and the SME mode (if used, see Section 7.10 on page 220) is associated with a separate key. Key management for the SEV feature is not handled by the CPU but rather by a separate processor known as the AMD Secure Processor (AMD-SP) which is present on AMD SOCs. A detailed discussion of AMD-SP operation is beyond the scope of this manual."[6]

AMD Secure RNG Library

"The AMD hardware RNG design allows access to output registers that enable reading the random values generated by the hardware. These registers are accessible to software through an x86user-level instruction. The x86 instructions are:

- RDRAND - Returns a 16-bit, 32-bit, or 64-bit random value
- RDSEED - Returns a 16-bit, 32-bit, or 64-bit conditioned random value

Accessing the random values using these low-level instructions can be cumbersome in high-level applications. Also, most applications would need a stream of random numbers which means multiple calls to RDRAND/RDSEED instructions."[7]

"During operation, 512 total bits of noise samples are collected and fed into an AES-256 CBC-MAC construct as specified in NIST SP 800-90Bsection 6.4.2. According to NIST's guidance, this construct produces 128-bits of "full entropy" since the input string was considered to have 256 (2*128) bits of assessed entropy. This entire process is repeated 3 times to generate a 384-bit seed to be used by the CTR_DRBG."[8]

AMD Virtualization Technology

Secure Virtual Machine

"SVM provides additional hardware support that is designed to facilitate the construction of trusted software systems. While the security features described in this section are orthogonal to SVM's virtualization support (and are not required for processor virtualization), the two form building blocks for trusted systems.

---

[6] AMD64 Architecture Programmer's Manual, https://www.amd.com/system/files/TechDocs/40332.pdf

[7] http://developer.amd.com/wordpress/media/2013/12/56310-AMD-Secure-Random-Number-Generator-Library.pdf

[8] https://www.amd.com/system/files/TechDocs/amd-random-number-generator.pdf

The AMD VM architecture is designed to provide:

- A guest/host tagged TLB to reduce virtualization overhead
- External (DMA) access protection for memory
- Assists for interrupt handling, virtual interrupt support, and enhanced pause filter
- The ability to intercept selected instructions or events in the guest
- Mechanisms for fast world switch between VMM and guest"[9]

SVM hardware extensions can be grouped into the following categories:

- State Switch: VMRUN, VMSAVE, VMLOAD instructions, global interrupt flag (GIF), and instructions to manipulate the latter (STGI, CLGI)[10]
- Intercepts: Allow the VMM to intercept sensitive operations in the guest[11]
- Interrupt and APIC Assists: physical interrupt intercepts, virtual interrupt support, APIC.TPR virtualization[12]
- SMM intercepts and assists[13]
- External (DMA) access protection[14]
- Nested paging support for two levels of address translation[15]
- Security—SKINIT instruction[16]

Secure Encrypted Virtualization

"The *Secure Encrypted Virtualization (SEV)* feature allows the memory contents of a VM to be transparently encrypted with a key unique to the guest VM. The memory controller contains a high-performance encryption engine which can be programmed with multiple keys for use by different VMs in the system. The programming and management of these keys and secure data transfer between host hypervisor and guest VM memory is handled by the SEV firmware running AMD Secure Processor. The API available to the host hypervisor for these operations is the focus of this document."[17]

"Secure Encrypted Virtualization (SEV) is available when the CPU is running in guest mode utilizing AMD-V virtualization features. SEV enables running encrypted VMs in which the

[9] AMD64 Architecture Programmer's Manual, https://www.amd.com/system/files/TechDocs/40332.pdf
[10] AMD64 Architecture Programmer's Manual, Section 15.5, Section 15.5.2, and Section 15.17
[11] AMD64 Architecture Programmer's Manual, Sections 15.7 - 15.14
[12] AMD64 Architecture Programmer's Manual, Section 15.17 and Section 15.21
[13] AMD64 Architecture Programmer's Manual, Section 15.22
[14] AMD64 Architecture Programmer's Manual, Section 15.24
[15] AMD64 Architecture Programmer's Manual, Section 15.25
[16] AMD64 Architecture Programmer's Manual, Section 15.27
[17] https://www.amd.com/system/files/TechDocs/55766_SEV-KM_API_Specification.pdf

code and data of the VM are secured so that the decrypted version is available only within the VM itself. Each VM may be associated with a unique encryption key so if data is Secure Virtual Machine accessed by a different entity using a different key, the SEV encrypted VM's data will be decrypted with an incorrect key, leading to unintelligible data. *It is important to note that SEV mode therefore represents a departure from the standard x86 virtualization security model, as the hypervisor is no longer able to inspect or alter all guest code or data.* The guest page tables, managed by the guest, may mark data memory pages as either private or shared, thus allowing selected pages to be shared outside the guest. Private memory is encrypted using a guest-specific key, while shared memory is accessible to the hypervisor."[18]

Secure Memory Encryption

"Software running in non-virtualized (native) mode can utilize the Secure Memory Encryption (SME) feature to mark individual pages of memory as encrypted through the page tables. A page of memory marked encrypted will be automatically decrypted when read by software and automatically encrypted when written to DRAM. SME may therefore be used to protect the contents of DRAM from physical attacks on the system. All memory encrypted using SME is encrypted with the same AES key which is created randomly each time a system is booted. The memory encryption key cannot be read or modified by software."[19]

Nested Virtualization

"Hardware support for improved performance of nested virtualization, which is the act of running a hypervisor as a guest under a higher-level hypervisor, is provided through the features described here. These relieve the top-level hypervisor from performing certain common, high-overhead operations that can occur with nested virtualization."[20]

Secure Nested Paging (SEV-SNP)

"The SEV-SNP features enable additional protection for encrypted VMs designed to achieve stronger isolation from the hypervisor. SEV-SNP is used with the SEV and SEV-ES features described in Section 15.34 and Section 15.35 respectively and requires the enablement and use of these features. Primarily, SEV-SNP provides integrity protection of VM memory to help prevent hypervisor-based attacks that rely on guest data corruption, aliasing, replay, and various other attack vectors. To achieve this, a new system-wide data structure called the Reverse Map Table (RMP) is used to perform additional security checks on memory access as described in Section 15.36.3. In addition to memory protection, SEV-SNP also includes several security features including a new VM Privilege Level (VMPL) architecture, interrupt

[18] AMD64 Architecture Programmer's Manual, https://www.amd.com/system/files/TechDocs/40332.pdf
[19] AMD64 Architecture Programmer's Manual
[20] AMD64 Architecture Programmer's Manual

injection restrictions, and side- channel protection. These features are designed to enable additional use models and enhanced security protections."[21]

Encrypted State

"Encrypted VMs that use the SEV feature described in Section 15.34 may additionally use the SEV-ES feature to protect guest register state from the hypervisor. An SEV-ES VM's CPU register state is encrypted during world switches and cannot be directly accessed or modified by the hypervisor. This is designed to protect against attacks such as exfiltration (unauthorized reading of VM state) and control flow attacks (modifying VM state) including rollback attacks (restoring an earlier VM register state)."[22]

Advanced Virtual Interrupt Controller

"The AMD Advanced Virtual Interrupt Controller (AVIC) is an important enhancement to AMD Virtualization Technology (AMD-V). In a virtualized environment, AVIC presents to each guest a virtual interrupt controller that is compliant with the local Advanced Programmable Interrupt Controller (APIC) architecture. See Chapter 16, "Advanced Programmable Interrupt Controller (APIC)," on page 583 for a detailed description of APIC."[23]

Secure Boot/SKINIT

"The SKINIT instruction is one of the keys to creating a "root of trust" starting with an initially untrusted operating mode. SKINIT reinitializes the processor to establish a secure execution environment for a software component called the secure loader (SL) and starts execution of the SL in a way that cannot be tampered with. SKINIT also copies the secure loader executable image to an external device, such as a Trusted Platform Module (TPM) for verification using unique bus transactions that preclude SKINIT operation from being emulated by software in a way that the TPM could not readily detect. (Detailed operation is described in Section 15.27.4.)"[24]

Secure Loader

"A secure loader (SL) typically initializes SVM hardware mechanisms and related data structures, and initiates execution of a trusted piece of software such as a VMM (referred to as a Security Kernel, or SK, in this document), after first having validated the identity of that software.

(It also sets up DEV (Device Exclusion Vector) for external access protection.)"[25]

---

[21] AMD64 Architecture Programmer's Manual https://www.amd.com/system/files/TechDocs/40332.pdf
[22] AMD64 Architecture Programmer's Manual
[23] AMD64 Architecture Programmer's Manual
[24] AMD64 Architecture Programmer's Manual
[25] AMD64 Architecture Programmer's Manual https://www.amd.com/system/files/TechDocs/40332.pdf

AMD SMM Supervisor

As previously mentioned in this document, a DRTM technology has limited value without addressing SMM originating attacks. The SMM supervisor is a technology from AMD that addresses this. Unfortunately, there is very little known or documented about it at the time of writing, except for two high-level statements (one in a blog post by Microsoft, and one in a blog post by AMD). The following is described:

- Installed during UEFI boot phase
- Authenticated by security processor at time of DRTM launch
- Deprivileges SMM handlers to Ring 3
- Installs Ring 0 kernel (signed by AMD) that regulates access to privileged resources
  - Based on a deny-list policy
- Policy is reported and verified

Shadow Stacks

This feature is not available before Ryzen 5000.

"The shadow stack mechanism facilitates protection against a common form of computer exploit known as Return Oriented Programming (ROP). ROP exploits utilize intentionally corrupted stack frames to divert normal processor control flow into short fragments of existing executable code, which ultimately end with a RET instruction. These fragments are then chained together using return addresses previously written to the stack by the attacker.

(Enabled through CET bit in CR4; see page 501 of the AMD64 Architecture Programmer's Manual)

Intel Control-flow Enforcement Technology (Intel CET). Bit 23. Setting this bit enables the shadow stack feature. This feature ensures that return addresses read from the stack by RET and IRET instructions originated from a CALL instruction or similar control transfer. See Section 18 "Shadow Stacks," on page 619 for more information. Before setting this bit, CR0.WP must be set to 1, otherwise a #GP fault is generated.

(See also chapter 6.7)"[26]

Supervisor Shadow Stack

This feature is not available before Ryzen 5000.

"The Supervisor Shadow Stack (SSS) feature is an extension to nested paging which allows a hypervisor to restrict which guest physical addresses may be used for a guest supervisor

[26] AMD64 Architecture Programmer's Manual https://www.amd.com/system/files/TechDocs/40332.pdf

shadow stack. Supervisor shadow stack accesses made by the guest to pages not designated as SSS pages in the nested page tables result in a #VMEXIT(NPF)."[27]

Guest Mode Execute Trap Extension

"The Guest Mode Execute Trap (GMET) extension allows a hypervisor to cause nested page faults on attempts by a guest to execute code at CPL0, 1 or 2 from pages designated by the hypervisor. The presence of the GMET extension is indicated by CPUID Fn8000_000A EDX[17]=1. The GMET mode is selected for a targeted guest by setting bit 3 of VMCB offset 090h to 1. For processors that don't support GMET this bit is ignored.

On GMET capable processors, when this bit is set to 1 on a VMRUN, the processor changes how the U/S bit in the nested page table is interpreted. The NX bit still prohibits execution of code at any privilege level when set to 1. However, with GMET enabled and the effective NX bit =0, if the effective U/S bit =1 and the page is being accessed for execution at CPL0, 1 or 2, a nested page fault #VMEXIT(NPF) is generated. If the effective NX bit =0 and the effective U/S bit =0 then the translation is allowed for the code page. The following table summarizes the behavior when GMET is enabled."[28]

External Access Protection

"By securing the virtual address translation mechanism, the VMM can restrict guest CPU accesses to memory. However, should the guest have direct access to DMA-capable devices, an additional protection mechanism is required. SVM provides multiple protection domains which can restrict device access to physical memory on a per-page basis. This is accomplished via control logic in the northbridge's host bridge which governs any external access port (e.g., PCI or HyperTransport technology interfaces)."[29]

I/O Memory Management Unit

"The I/O Memory Management Unit (IOMMU) extends the AMD64 system architecture by adding support for address translation and system memory access protection on DMA transfers from peripheral devices. IOMMU also helps filter and remap interrupts from peripheral devices.

The IOMMU enables several significant system-level enhancements:

- Legacy 32-bit I/O device support on 64-bit systems (generally without requiring bounce buffers and expensive memory copies).
- More secure user-level application access to selected I/O devices.
- More secure VM guest OS access to selected I/O devices.

[27] AMD64 Architecture Programmer's Manual
[28] AMD64 Architecture Programmer's Manual
[29] AMD64 Architecture Programmer's Manual

The IOMMU can be used to:

- Replace the existing Graphics Address Remapping Table (GART) mechanism.
- Remap addresses above 4GB for I/O devices that do not support 64-bit addressing.
- Allow a guest OS running on a VM to have direct, assigned control of a device.
- Provide page granularity control of device access to system memory.
- Allow a device direct access to user space I/O.
- Allow direct delivery of interrupts to a guest OS.
- Filter and remap interrupts.
- Share process virtual address space with selected peripheral devices.
- Isolate/sandbox devices to prevent malicious DMA accessing security sensitive OS and user data in memory.
- Enforce OS security policies for data access

The IOMMU can be thought of as a generalization of two facilities included in the AMD64 architecture: the GART and the Device Exclusion Vector (DEV). The GART provides address translation of I/O device accesses to a small range of the system physical address space, and the DEV provides a limited degree of I/O device classification and memory protection. With appropriate software support, the IOMMU can emulate the capabilities of the GART or DEV. IOMMU optionally provides the capability to remap peripheral interrupt vectors."[30]

AMD SMM Supervisor

"Since the SMI handler is typically provided by a developer different then the OS and SMM handler code running at a higher privilege has access to OS/Hypervisor Memory & Resources. Exploitable vulnerabilities in SMM code leads to compromise of Windows OS/HV & Virtualization Based Security (VBS). To help isolate SMM, AMD introduces a security module called AMD SMM Supervisor that executes immediately before control is transferred to the SMI handler after an SMI has occurred. AMD SMM Supervisor resides in AMD DRTM service block and the purpose of AMD SMM Supervisor is to:

- Block SMM from being able to modify Hypervisor or OS memory. An exception is a small coordinate communication buffer between the two.
- Prevent SMM from introducing new SMM code at run time
- Block SMM from accessing DMA, I/O, or registers that can compromise the Hypervisor or OS"[31]

[30] https://www.amd.com/system/files/TechDocs/48882_IOMMU_3.05_PUB.pdf

[31] https://community.amd.com/t5/amd-business-blog/amd-and-microsoft-secured-core-pc/ba-p/418204

Detailed Features Comparison

Hardware-assisted AES Instruction Set

Both Intel and AMD processors provide instructions to accelerate symmetric block cipher encryption/decryption of 128-bit data blocks using the AES specified by the NIST publication FIPS 197.

Initially the instructions were introduced as AES-NI which use the 128-bit SIMD/XMM registers. With the addition of AVX2 extensions, 256-bit vector register (YMM) support was added as VEX- encoded instructions. More recently, with the introduction of AVX-512 extensions, support for 512-bit vector registers (ZMM) were added with the EVEX- encoded instructions. The implementation of these instructions offer increased efficiency of execution compared to software implementations.

AES Encryption

Instruction	Description	Intel Supported	AMD Supported
AESENC	128-bit AES-NI	Y	Y
AESENCLAST	128-bit AES-NI	Y	Y
VEX-VAESENC	256-bit AVX	Y	Y
VEX-VAESENCLAST	256-bit AVX2	Y	Y
EVEX-VAESENC	512-bit AVX-512	Y	N
EVEX-VAESENCLAST	512-bit AVX-512	Y	N

AES Decryption

Instruction	Description	Intel Supported	AMD Supported
AESDEC	128-bit AES-NI	Y	Y
AESDECLAST	128-bit AES-NI	Y	Y
VEX-VAESDEC	256-bit AVX	Y	Y
VEX-VAESDECLAST	256-bit AVX2	Y	Y
EVEX-VAESDEC	512-bit AVX-512	Y	N
EVEX-VAESDECLAST	512-bit AVX-512	Y	N

AES Inverse Mix Column Transformation

Instruction	Description	Intel Supported	AMD Supported
AESIMC	128-bit AES-NI	Y	Y
VAESIMC	256-bit AVX2	Y	Y

Instruction	Description	Intel Supported	AMD Supported
EVEX-VAESIMC	512-bit AVX-512	Y	N

AES Create Round Keys with Key Expansion Schedule

Instruction	Description	Intel Supported	AMD Supported
AESKEYGENASSIST	128-bit AES-NI	Y	Y
VEX-VAESKEYGENASSIST	256-bit AVX2	Y	Y
EVEX-VAESKEYGENASSIST	512-bit AVX-512	Y	N

Cryptographically Secure Random Number Generator

The Intel Secure Key technology provides two instructions which are supported on both Intel and AMD processors. On AMD this feature is known as RNRAND.

Both the Intel and AMD documentation explicitly mention compliance to the NIST SP800-90A/SP800-90B/SP800-90C standards.

Cryptographically Secure Deterministic Random Bit Generator

Instruction	Description	Intel Supported	AMD Supported
RDRAND	CSDRBG	Y	Y

Both the Intel and the AMD implementation of the RDRAND instruction uses a Cryptographically Secure Deterministic Random Bit Generator (DRBG) designed to meet the NIST SP800-90A standard.

Cryptographically Secure Enhanced Non-Deterministic Random Bit Generator

Instruction	Description	Intel Supported	AMD Supported
RDSEED	CS Enhanced NDRBG	Y	Y

Intel Secure Key implementation of the RDSEED instruction uses an Cryptographically Secure Enhanced Non-Deterministic Random Bit Generator (NRBG) designed to meet the NIST SP 800-90B and NIST SP800-90C standards. According to AMD documents they follow NIST SP 800-90B standard.

Intel Virtualization Technology (Intel VT-x) vs. AMD-V

Intel VT-x is fundamentally the same technology that AMD calls AMD-V.

On a technical level the names of the CPU feature-flags and the names of the new instructions differ, but the concepts are the same.

Note that the actual meaning of the term has changed over the years as additional features (e.g., Extended Page Tables) have been added.

In our estimation, these are not security but performance improvements. They add hardware support for running VM kernels in a separate “Ring 0” that looks and feels like a Ring 0 but cannot harm the hypervisor or other guests. This functionality previously had to be emulated in software, and it was (i.e. in VMware or QEMU).

The performance gain is potentially large, but the overall security does not improve (emphasis added):

> Virtualization technologies are dependable on the specific set of instructions in the CPU, even when on the same architecture. There are many hardware virtualization technologies, e. g., Intel Vt-X, AMD-V, ARM-VEx, but the hypervisor core cannot support simultaneously more than one, as a consequence of the instruction set. **For this reason, just one technology was chosen, the Intel VT-x, mainly because of the higher market share of its CPU, better architecture documentation, and better support for nested hypervisor in industry virtualization solutions.** The last criteria mean that the proposed solution can run in environments already virtualized, where the OS is at the top of Type 1 or 2 hypervisor. Along with Intel’s Virtual Machine Extensions (VMX), from Intel Vt-X, the hypervisor will also use the following Intel extensions: EPT, for Second Layer Address Translation(SLAT) and Vt-D, for I/O Memory Management Unit (IOMMU).[32]

Intel VT-d vs AMD-Vi (IOMMU)

Intel VT-d is fundamentally the same technology that AMD calls AMD-Vi (or just IOMMU).

On a technical level it lets a hypervisor grant access to an I/O device to a guest. Where previously every hardware access from the guest had to be intercepted and remapped by the hypervisor, this remapping can now be done in hardware, reducing the number of VMEXIT traps and improving performance.

In our estimation these are not security but performance improvements. The functionality previously had to be emulated in software, as VMware and QEMU did, or one would have to use a paravirtualized OS.

The performance gain is potentially large, but the overall security does not improve.

There have been security issues discovered in the past, both in Intel VT-d/x and AMD-Vi. A paper by Mengyuan et al. describes a previously unexplored security issue of AMD SEV—the unprotected I/O operations of SEV-enabled guest VMs:

> “The root cause of the problem is the incompatibility between AMD-V’s I/O virtualization with SEV’s memory encryption scheme. The paper demonstrates that the unprotected I/O

[32]. Silva, Otávio, & P. Lício de Geus. 2020. “Lokke, a hybrid security hypervisor.” *Universidade Estadual de Campinas*. https://www.lasca.ic.unicamp.br/paulo/papers/2020-SBSeg-otavio.silva-lokke.hybrid.supervisor.pdf.

operations could also be exploited to construct powerful attack primitives, enabling the adversary to perform arbitrary memory encryption and decryption."[33]

A paper by Coppolino et al. refers to weaknesses on both platforms detected in the past:

"An additional attack to IOMMU came from Morgan et al. who showed that it can be violated exploiting a weakness in the typical design of Intel VT-d and AMD-Vi. The weakness is related to the configuration tables of the IOMMU, which are initialized in a DRAM region which is not protected from DMA accesses. A malicious peripheral may benefit from this weakness to modify these tables in memory just before the hardware setup of the IOMMU."[34]

Discrepancy in Kernel DMA protection

Windows' Kernel DMA protection protects against malicious DMA by devices connected to easily accessible internal/external DMA-capable ports, such as M.2 PCIe slots and Thunderbolt3, during OS runtime. This was first implemented in Windows 1803, specifically to protect against Thunderclap. This was later expanded on in Windows 1903, to protect against internal devices as well.

Thunderclap is a Thunderbolt-based attack that takes advantage of the fact that the IOMMU is page aligned, whereas drivers that want to map memory are not aware of this, because of DMA API abstractions. If a driver ends up mapping memory that is not page-aligned, the delta will still be accessible to Thunderbolt devices and can be tampered with (see thunderclap.io for more details). This is where the kernel DMA protection seems to originate, note that the resolution is more sophisticated than simply enabling IOMMU; that is already a given, and Windows already had solutions in place for this, in particular to protect BitLocker attack scenarios. The extra sophistication is the use of bounce buffers if mappings are not page aligned.

In order to determine the PCI root ports on which to apply this, a new ACPI object is needed for placement there via UEFI platform code. This object must have the property name ExternalFacingPort.[35] This functionality is also implemented in the Linux kernel, where the Linux VT-d driver looks for it, and if found, makes use of bounce buffers. The AMD-Vi driver on Linux does not implement this, nor does this seem supported on the Ryzen PRO 4000 machines we used for testing. The reason it turns out, is quite obvious: these AMD machines do not support Thunderbolt.

Kernel DMA protection was then expanded in Windows 1903 to also protect some internal PCIe (such as M.2), presumably those that are easy to connect devices with. In order to do

---

[33] Mengyuan, Li, Y. Zhang, Z. Lin, & Y. 2016 "Exploiting Unprotected I/O Operations in AMD's Secure Encrypted Virtualization." *Proceedings of the 28th USENIX Security Symposium*. https://www.usenix.org/system/files/sec19-li-mengyuan_0.pdf.

[34] Coppolino Luigi, S. D'Antonio, G. Mazzeo, & L. Romano. 2019. "A comprehensive survey of hardware-assisted security: From the edge to the cloud." *Internet of Things*, Vol. 6. doi.org/10.1016/j.iot.2019.100055.

[35] https://docs.microsoft.com/en-us/windows-hardware/drivers/pci/dsd-for-pcie-root-ports

this, another ACPI object named DmaProperty is required. We assume it applies the same bounce buffer logic for those devices. It must be noted that this is currently not implemented in the Linux VT-d driver, although it has been discussed on public Linux kernel developer mailing lists.[36] While it would not make much sense for AMD to implement the "ExternalFacingPort" ACPI object at this time (because they do not use Thunderbolt) it would make sense for them to implement the DmaProperty ACPI object. Currently this does not appear to be implemented in their UEFI platform code; this was tested on two different AMD Ryzen PRO 4000 laptops, and neither supported this.

Intel APICv vs AMD AVIC

This is fundamentally the same technology that AMD calls AVIC (or IOMMU). On a technical level, it allows a hypervisor to let the hardware route interrupts directly to guests where previously software emulation was necessary.

We consider these performance optimizations, not security improvements; the performance gain is potentially large, but the overall security does not improve.

As noted above, in their paper Mengyuan et al. show an incompatibility of AMD-V's I/O virtualization and AMD SEV, leading to a severe breach of security.

Intel TXT vs AMD SKINIT

Intel TXT is an implementation of the Trusted Computing Group's Dynamic Root of Trust for Measurement (DRTM) concept. Instead of measuring a chain of trust rooted in hardware and including all components of the boot process, DRTM adds hardware support to allow a piece of code to establish itself as trustworthy, regardless of the previous environment.

The piece of code that wants to establish trust for itself is called Measured Launch Environment (MLE).

The loader code loads the MLE and the ACM into memory and executes a special new CPU instruction which will copy the ACM into a separate in-CPU memory area and validate and execute it there to prevent outside tampering. The ACM will then take a measurement (in the TPM sense) of the MLE code and put the resulting hash value into the TPM, where MLE can read it.

Intel TXT prevents I/O device-based interference with the measurement by configuring the IOMMU/VT-d to disallow I/O access to the memory, and it prevents other threads of execution by forcing all processors into a well-defined initialization state.

Intel TXT is used in the boot process to establish trust for the initial loading of the hypervisor or OS kernel.

[36] https://lkml.org/lkml/2020/7/6/1020

AMD's alternative to Intel TXT is called SKINIT after the new CPU instruction used to execute it. While SKINIT achieves the same goal, it comes with less options and instead utilizes a design that chooses simplicity over configurability.

As previously stated, Intel TXT and SKINIT are of limited value without addressing SMM originating attacks.

SMM Defense Technologies

The x86 architecture has a little-known feature called System Management Mode. It is meant for platform specific things like Power Management, is triggered externally (for example if the power cable is unplugged on a laptop) and has full system access. The code comes from the BIOS, so it is generally assumed to be trusted, with the Static Root of Trust to ensure the BIOS is not manipulated.

However, with Intel TXT promised to be useful even in the absence of the static root of trust, a solution to system management mode is necessary. Intel TXT disables interrupts in the MLE, so SMM interference is already impossible until MLE turns interrupts back on. AMD SKINIT does the same.

Unfortunately, the SMM code is in its own special memory area, called SMRAM, which is not readily accessible by the regular system code running on the CPU. This means that the MLE code cannot simply verify the data in the SMRAM.

Different approaches have been proposed for this problem. The first proposal was to have an STM (SMM Transfer Monitor), which would be somewhat like a hypervisor around the SMM code. Its function would be to restrict what the SMM code could access and do.

Intel's first iteration of the solution, with the 9th Gen Intel Core vPro processors, was Intel Runtime BIOS Resilience, which added UEFI code to make sure SMM code starts with a page table that restricts write access to pages comprising the page table, removes write access to code pages, sets the no-execute bit for data pages, and uses the CPU to lock the page table. That way, if an attacker manages to exploit a bug in the SMI handler code, they will not be able to write to any memory. If a platform needs access to some additional piece of memory, the attacker can put that in a special table used to set up the page table.

Intel's second iteration, with the 8th Gen Intel Core vPro processors, is Intel System Security Report. It adds a piece of signed Intel code, the PPAM, which can measure the contents of the configuration tables. PPAM itself is verified by the CPU itself. The result of PPAM's analysis is exported via an ephemeral PCR in the TPM and can be queried by the MLE.

Intel's third iteration, with the 10th Gen Intel Core vPro processors, is Intel System Resources Defense, which adds some more Intel signed code to the SMM, which is executed before the rest of the SMM code. It sets up a small Ring 0 kernel and runs the rest of the SMM code in Ring 3. That way, when the SMM code tries to do something it should not, the Ring 0 code can disallow it. Basically, this Ring 0 kernel enforces the policy configuration tables.

In essence, BIOS resilience sets up a restricted environment for SMM handlers, Intel System Security Report measures this, and Intel System Resources Defense enforces that SMM handlers cannot and will not change their own restricted environment. Combined, these

technologies form the Intel Hardware Shield, and they all are MLE code for finding out if the SMM code is enforcing the right kind of policy.

AMD's solution, while different, appears to solve very similar problems. A SMM Supervisor is introduced during the UEFI boot phase. It is AMD signed, and the Platform Security Processor (PSP) (their version of the Management Engine) will check its signature. The PSP also has a mechanism to prevent rollback attacks on the firmware, so an attacker cannot flash a new firmware without the SMM Supervisor.

The supervisor introduces a new SMM entry routine that will set up a Ring-0-and-Ring-3 environment for the SMM code, and it also has a policy (a deny list) that it checks against. AMD claims to also provide a way to measure the policy after SKINIT (their equivalent of the MLE), described as "SMM secure policy is reported to and verified by OS secure loader during DRTM event." There is extraordinarily little documentation available about the AMD SMM supervisor at the time of writing[37][38] and we can only go by that existing documentation.

On the surface, Intel and AMD appear to have feature parity. At a high level, the feature claims seem to be equivalent. Intel's measurement of the configuration table by the PPAM is very explicit, is a feature on its own, and is seen as one explicit part in a larger story. The AMD set of technologies, in particular, related to measurement, are not explicit: measurement of policy is only mentioned in a single sentence, in a blog post by Microsoft, without any further explanation.

Given the lack of documentation on the AMD side, it is hard to make a comparison and call out differentiators. One can either take AMD's claim (through a single sentence on Microsoft's blog) at their word or take a very critical look at this single-sentence claim and question whether or not it holds up. However, it is outside the scope of this effort to validate that the respective signed code blobs provide the promised protection, as this would require substantial reverse engineering and pretty elaborate and low-level OS-like code to inspect.

The obvious and clear differentiator here is the level of documentation. On the Intel side, there are not only documents provided under a non-disclosure agreement, but also public patents that describe Intel TXT+ISSR+ISRD+DGR. Although patent parlance is not trivial to decipher, you can make the argument that it is there. On the AMD side, there appears to virtually no documentation besides the one sentence.

Platform Differentiation

There appear to be two differentiators when comparing feature documentation between platforms. While both Intel and AMD use paging and further deprivilege the SMM handlers to Ring 3, there is a documented implementation difference in the Ring 0 code.

[37] https://www.microsoft.com/security/blog/2020/11/12/system-management-mode-deep-dive-how-smm-isolation-hardens-the-platform/

[38] https://community.amd.com/t5/amd-business-blog/amd-and-microsoft-secured-core-pc/ba-p/418204

On the Intel side, there is an exception handler that catches privileged operations such as an MSR write and then compares this against the OEM/IHV provided policy.

From the little documentation that is available, the AMD SMM supervisor does not appear to work in the same way. According to the aforementioned Microsoft blog post, “SMM Supervisor provides *syscall* interface to allow third-party SMI handlers to make such requests. The backend of the *syscall* interface, which resides in SMM supervisor, is controlled by SMM secure policy.” Instead of allowing the hardware to generate exceptions, the Ring 3 code will call into the supervisor with a *syscall* and ask it to perform a privileged action. While this might provide feature parity, it adds complexity and attack surface, since the Ring 0 code now has to perform non-trivial input parsing from untrusted Ring 3 code.

Absent code review and/or reverse engineering, it is difficult to determine the significance of differences in attack surface complexity between these platforms.

The second implementation differentiator is in the handling of policies.

AMD’s policy is deny-list-based, as described in the Microsoft blog post: “said policy is a deny list that can be customized per platform to determine which MSRs, IOs, or memory regions can be accessed from CPL3.”

Intel does not have a deny list for their policy, and instead the policy is declarative: each OEM specifies their policy. Anything outside of the OEM-specified policy is denied by default, although BIOS implementations can change this behavior.

While ultimate responsibility for a secure security policy lies with the OEMs, the platform can try to provide a mechanism for policy creation that is as secure as possible. In that respect the declarative policy is inherently more secure than a deny list.

Considerations

Further improvements could be made on Intel’s part regarding transparency and getting community support and buy-in for these features:

- Clear, detailed, accurate public documentation that rises to the level of internal dev-design specs
- An open-source implementation that can be objectively assessed without having to take anyone’s word for it; make the platform code (and other relevant code) publicly available on GitHub
- A detailed set of steps that describe “if you use compiler X, version Y with options Z the compiled blob will match byte-for-byte with the intel signed blob running on machines (minus the actual signature, for obvious reasons)”

Intel TME vs AMD SEV/SME

This feature is a mechanism to encrypt the memory contents in the memory controller, so that the data in the memory is encrypted but encryption and decryption are transparent to the software.

Since decryption is transparent to the software, this does not prevent any kind of software-based attack trying to read or write memory. However, it does help in the scenario where the attacker has physical access to the machine, removes the DRAM from the victim machine, and puts it into a different machine. If this switch is done quickly (the time window can be extended by cooling the chips), then the old data from the victim machine will still be stored in the DRAM and can be read out in the new host. Memory encryption would prevent this attack.

Please note that other physical attacks are possible and cannot be prevented by Intel TME. An obvious example would be an "evil maid" attack[39].

Memory encryption could also be useful for a new class of memory devices like Intel Xpoint, where you plug in persistent storage through a DRAM slot. In that case memory encryption would act like a full disk encryption on a regular storage device. However, for that to be useful the key would have to be persistent as well, making it subject to attack.

Another potential hardware attack prevented by memory encryption would be an "evil DRAM" or a debugging/reverse engineering device like a DDR interposer.

Multiple academic papers describe severe flaws in AMD SEV. In addition to the severe breach of security described in the paper from Mengyuan, Morbitzer, et al. present an optimized attack for a hypervisor to extract secrets from encrypted VMs (emphasis ours):

> *AMD SEV is a hardware extension for main memory encryption on multi-tenant systems. SEV uses an on-chip coprocessor, the AMD Secure Processor, to transparently encrypt virtual machine memory with individual, ephemeral keys never leaving the coprocessor. The goal is to protect the confidentiality of the tenants' memory from a malicious or compromised hypervisor and from memory attacks, for instance via cold boot or DMA. The SEVered attack has shown that it is nevertheless possible for a hypervisor to extract memory in plaintext from SEV-encrypted virtual machines without access to their encryption keys. However, the encryption impedes traditional virtual machine introspection techniques from locating secrets in memory prior to extraction. This can require the extraction of large amounts of memory to retrieve specific secrets and thus result in a time-consuming, obvious attack.* ***We present an approach that allows a malicious hypervisor quick identification and theft of secrets, such as TLS, SSH or FDE keys, from encrypted virtual machines on current SEV hardware.*** *We first observe activities of a virtual machine from within the hypervisor in order to infer the memory regions most likely to contain the secrets. Then, we systematically extract those memory regions and analyze their contents on-the-fly.* ***This allows for the efficient retrieval of targeted secrets, strongly increasing the chances of a fast, robust and stealthy theft.***[40]

[39] https://en.wikipedia.org/wiki/Evil_maid_attack

[40] Morbitzer, Mathias, M. Huber, & J. Horsch. 2019. "Extracting Secrets from Encrypted Virtual Machines." *CODASPY*. https://arxiv.org/pdf/1901.01759.pdf.

Intel CET

This consists of two hardware-assisted exploit mitigation techniques. While these techniques will not prevent security vulnerabilities in code, they will make exploiting them more difficult. These are shadow stacks and Indirect Branch Tracking (IBT).

The technical idea for the shadow stack feature is to have a second (shadow) stack in a protected place in memory where not even the kernel can manipulate it. Then, when an attacker uses a memory corruption bug to overwrite the call stack to gain control of the flow of execution, the hardware can detect the mismatch between the call stack and the shadow stack and kill the process.

IBT attempts to prevent successful memory corruption exploitation by making ROP/JOP (return oriented programming/jump oriented programming) exploitation more difficult. ROP/JOP cobbles together the code that an attacker wants to execute from partial bits and pieces that are already available in a program, often by calling or jumping to a place close to a function return, as to only execute a small set of desired instructions. When IBT is enabled, any indirect JMP or CALL instruction must land on one of the ENDBRACH instructions or a control protection fault (#CP) will be triggered. This ensures that control flow will follow what the developer (and compiler) intended.

It must be noted that neither of these remove the bug or the exploit, instead turning a potential code execution vulnerability into a definite denial-of-service vulnerability, which is usually seen as lower severity.

Both Intel and AMD have implemented this technology, but the AMD implementation is only now hitting the market with the recent Ryzen 5000 series chips. This comparison is still against Ryzen 4000 series, which does not have this technology, so the advantage lies with Intel here.

By the time of broad adoption of this feature, AMD will also include it in their mainstream offerings.

Note that this feature is opt-in, so only a few applications currently use it.

Li, et al. highlight the lack of measurement of efficacy and robustness of various Control-Flow integrity mechanisms:

> *We evaluated 12 most recent open-source CFI mechanisms and discovered 10 flaws in most CFI mechanisms or implementations. For some CFIs, their security policies or protected ICT sets do not match what they claimed. Some CFIs even expand the attack surface (e.g., introducing unintended targets). To facilitate a deeper understanding of CFI,*

we summarize the flaws into 7 common pitfalls which cover the whole lifetime of CFI mechanisms and reveal issues that affect CFI mechanisms in practical security.[41]

Intel KeyLocker Technology

These new instructions provide a way to use hardware-accelerated AES encryption and decryption without having to have the key in memory. The key still must be in memory initially for the upper protocol handling code and for telling the hardware what it is, but after that the code can use a "handle" instead of the actual key to do the encryption and decryption.

We believe this is a good idea, as it decreases the window of opportunity for extracting the key from the memory of the process. KeyLocker protects keys from other software elements once they have been placed in the locker, and as such it is meant as a defense-in-depth-capability.

AMD currently has no equivalent technology, and as far as we are aware has not announced any.

Intel TDT

The idea behind this technology is to use hardware support for performance profiling to extract a "code execution fingerprint" from running software, and then to use that for detecting malware using machine learning to compare this fingerprint to a library of known malware.

This unique blend of hardware telemetry features (utilizing profiling hooks leveraging a very rich set of events) and software is an innovative approach that adds another signal to defensive technologies. Due to the use of profiling events, Intel TDT can even perform threat detection for virtualized guests without requiring an agent running inside of a guest or performing complex hypervisor introspection.

Additionally, Intel TDT is designed to be offloaded to the GPU. GPUs are commonly underutilized in the vPro business case, and as such Intel TDT has found a clever way to be able to perform the collection and computation it needs to perform without having to take up many CPU cycles. Please note that this is the default, Intel TDT can also perform its work from the CPU.

Because it utilizes very specific Intel profiling events, Intel TDT only runs on Intel hardware. AMD does not offer a product like this.

[41] Li, Yuan, M. Wang, C. Zhang, X. Chen, S. Yang, & Y. Liu. 2020. "Finding Cracks in Shields: On the Security of Control Flow Integrity Mechanisms." *CCS '20: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. https://dl.acm.org/doi/10.1145/3372297.3417867.

The tested samples against this version of Intel TDT (see Intel Threat Detection Technology Tests) reflect current commonly seen ransomware strains as reported on popular anti-malware blogs and websites.[42] [43] [44] [45]

[42] https://www.datto.com/blog/common-types-of-ransomware

[43] https://www.cloudwards.net/ransomware-statistics/

[44] https://cybriant.com/top-ransomware-threats-of-2020/

[45] https://resources.infosecinstitute.com/topic/top-6-ransomware-strains-to-watch-out-for-in-2020/

Intel Threat Detection Technology Tests

System Specification

10th Gen Intel Core vPro processors

Item	Value
OS Name	Microsoft Windows 10 Enterprise
Version	10.0.19041 Build 19041
System Manufacturer	Lenovo
System Type	X64-based PC
System Model	81UE
System SKU	LENOVO_MT_81UE_BU_idea_FM_YOGA C640-13IML
Processor	Intel Core i7-10510U CPU @ 1.80GHz, 2301 MHz, 4 Core(s), 8 Logical Processor(s)

11th Gen Intel Core vPro mobile processors

Item	Value
OS Name	Microsoft Windows 10 Pro
Version	10.0.19041 Build 19041
System Manufacturer	Intel Corporation
System Type	X64-based PC
System Model	11th Gen Intel Core vPro mobile processors
System SKU	0101100100010000
Processor	11th Gen Intel Core i7-1185G7 @ 3.00GHz, 2995 MHz, 4 Core(s), 8 Logical Processor(s)

AMD

Item	Value
OS Name	Microsoft Windows 10 Pro
Version	10.0.18363 Build 18363
System Manufacturer	Lenovo
System Type	x64-based PC
System Model	20UD000GUS
System SKU	LENOVO_MT_20UD_BU_Think_FM_ThinkPad T14 Gen 1
Processor	AMD Ryzen 7 PRO 4750U with Radeon Graphics, 1700 MHz, 8 Core(s), 16 Logical Processor(s)

Test Artifacts

Cryptominer Binaries

Tool ID	Name	Version	SHA256SUM
#T0	Xmrig	5.5.0 win32	363841b14e9048fd50a012f2a3e04c3f86312fbcd3c1f4a837a102fe7e258ca7[46]
#T1	Xmrig	6.5.3 win64	bb5732cbc2515326b1f2df5e834223594dfd3c41b26bc5d06b80d30fe2870efd
#T2	Xmr-Stak-Rx	1.0.5 win64	f9bbb938a6190a03b15a643bc546bf13ee7cf460954665b4de88691589710090
#T3	Xmr-Stak	2.10.8 win64	1cd3333a4dd24ca3a6674ac3efac8cb3faf4094d39128f808971fcda2a47401b[47]

Obfuscated Cryptominer Binaries

Tool ID	Name	Version	SHA256SUM
#OT0	Xmrig	5.5.0 win32	16820f806ca7b071161c653fadf37e03f4a7fe5372d9739e34352d4b7bd7c262
#OT1	Xmrig	6.5.3 win64	0709a863b9389bbbe9b25ab1b6c3c84daacaf8274eaea437529d067ba24b9e87
#OT2	Xmr-Stak-Rx	1.0.5 win64	8c6eeb1ee5adc229b7c9ece51f3b87d5678778d00fa1b25875d473d44c8ba7ff
#OT3	Xmr-Stak	2.10.8 win64	7024e43610c4b0e83809a2842ecfb0c0e39e0fec3063e8e3dea79a78ffbae81b

[46] Obtained by IOActive from Xmrig GitHub. Last known release that supports 32-bit architecture, which is required according to the Intel TDT Test Plan.

[47] Obtained by IOActive from Xmr-Stak GitHub. The -Rx release provided by Intel only contains RandomX algorithms, but Intel TDT Test Plan also needs Cryptonight, which is contained in this release.

Cryptominer Configs

Filename	SHA256SUM
config_cn0.json	3f97c19d535b8b2f139ee765fa028de0b9cb1278f4b4c62ad694f506927bcf7a
config_cn1.json	8af6bca0d4b622560ac14f1548b3fc1bef419ccd11baa8a4e365373679785845
config_cn2.json	3a2270fffb0612f61174c6bab5f9b54011b23a15a087729df145eec749bd78a7
config_cn-lite0.json	03e2bbf312e3f508d52c42114f7539b7752c3e8d2a07ce4c15af5294cea1e072
config_cn-lite1.json	c6b77c712ac2524a3e45ed56d8a0656197aa7738e683ec55d894bfdc9c7fed58
config_rx0.json	568b4cc93c6952205b3173f07a82aba72342782981d888d2600b60d013e8ea41
config_rxarq.json	12188a9e522e754696b97cc002025fce1cb4e9087f09225cb35cee9e483204e1
config_rxkeva.json	13a4decc037adf7ce360acc44932d370246557a27d67abbb37f3263063930ed3
config_rxsfx.json	41bb1a1f328d2261d949262b3875cdb46f2911c9aba883c5e357132735515554
config_rxwow.json	5c391b8ca54fcff0fa14817da7654e3edfbf6fcdaec422c467d1082189fa4ae0

Cryptominer Samples

Note that samples #C0A to #C13 are 64-bit Xmrig. #C14 to #C19 are 64-bit Xmr-Stak-Rx which supports RandomX algorithm. #C1A is 64-bit Xmr-Stak which supports Cryptonight algorithm.

Sample ID	Tool Ref	Algorithm	NOTE
#C0A	#T1	RandomX (Monero)	`-c config_rx0.json`
#C0B	#T1	RandomARQ	`-c config_rxarq.json`
#C0C	#T1	RandomKEVA	`-c config_rxkeva.json`
#C0D	#T1	RandomSFX	`-c config_rxsfx.json`
#C0E	#T1	RandomWOW	`-c config_rxwow.json`
#C0F	#T1	Cryptonight (Original)	`-c config_cn0.json`
#C13	#T1	Cryptonight Lite (Variant 1)	`-c config_cn-lite1.json`
#C14	#T2	RandomARQ	`--currency arqma`
#C15	#T2	RandomKEVA	`--currency keva`
#C16	#T2	RandomXL	`--currency loki`
#C17	#T2	RandomX (Monero)	`--currency monero`
#C18	#T2	RandomSFX	`--currency safex`
#C19	#T2	RandomWOW	`--currency wownero`

Obfuscated Cryptominer Samples

The following samples are packed with UPX from their original cryptominer samples in the above tables.

Sample ID	Tool Ref	Algorithm	NOTE
#OC0A	#OT1	RandomX (Monero)	`-c config_rx0.json`
#OC0D	#OT1	RandomSFX	`-c config_rxsfx.json`
#OC14	#OT2	RandomARQ	`--currency arqma`
#OC17	#OT2	RandomX (Monero)	`--currency monero`

Ransomware Samples

The following samples were downloaded from VirusTotal.

Sample ID	Common Name	Hash
#R00	Robinhood	cc899c93b11eb911182636130d16f8b58665ad2156cb12be7947d85c1c190cc6
#R01	Robinhood	597ff7cdb2e435c1ea7adb6ae65bfd33078e19308515397a18a9c2bef3e3566d
#R02	Maze	153defee225de889d2ac66605f391f4aeaa8b867b4093c686941e64d0d245a57c
#R03	Maze	dee863ffa251717b8e56a96e2f9f0b41b09897d3c7cb2e8159fcb0ac0783611b
#R04	WastedLocker	8897db876553f942b2eb4005f8475a232bafb82a50ca7761a621842e894a3d80
#R05	WastedLocker	bcdac1a2b67e2b47f8129814dca3bcf7d55404757eb09f1c3103f57da3153ec8
#R06	Netwalker	34dffdb04ca07b014cdaee857690f86e490050335291ccc84c94994fa91e0160
#R07	Netwalker	246aea5a28ed117238ed0da8e6c96a9a9f1c627613d0f9f57da3e819f57231eb
#R08	Snake	E5262db186c97bbe533f0a674b08ecdafa3798ea7bc17c705df526419c168b60
#R09	Sodinokibi	0fa207940ea53e2b54a2b769d8ab033a6b2c5e08c78bf4d7dade79849960b54d
#R0A	Crysis	4b8271802c7cfec3b5258b581f4cb871edcc0c7bfb3bb7621707bdca094049a0
#R0B	Wannacry	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
#R0C	SamSam	0f2c5c39494f15b7ee637ad5b6b5d00a3e2f407b4f27d140cd5a821ff08acfac
#R0D	Shade	a8b27aa4fe7df15a677f9ab9b62764d557525059a9da5f4196f1f15049e2b433

Obfuscated Ransomware Samples

The following samples are packed with UPX from their original ransomware samples in the above tables.

Sample ID	Common Name	Hash
#OR00	Robinhood	`b4edc2a1234be734917bc8612f81d318bf8bac3f61e261205b050cc34e195da5`
#OR02	Maze	`0abf59650fe06c483e68e3a42980fbefe05ae67fb98367a0c6e3c605ab60cb2b`
#OR06	Netwalker	`b065a36d0c209628bf842d97e229aaada63bf239eff9535e01a70f91eccace64`
#OR09	Sodinokibi	`0637ed84a75943ba4ce8edade0bd0bba6d30bda31f8cd7ca5d91ec633b168415`
#OR0A	Crysis	`15bd8c86231250d060edabb4b9ccf2f700ac85a50ec8107f7d5fdd45dae76ae2`
#OR0D	Shade	`ca78ff3423479bf88489fcff1aaca85ae5de47da495308efd2b610da292b70bf`

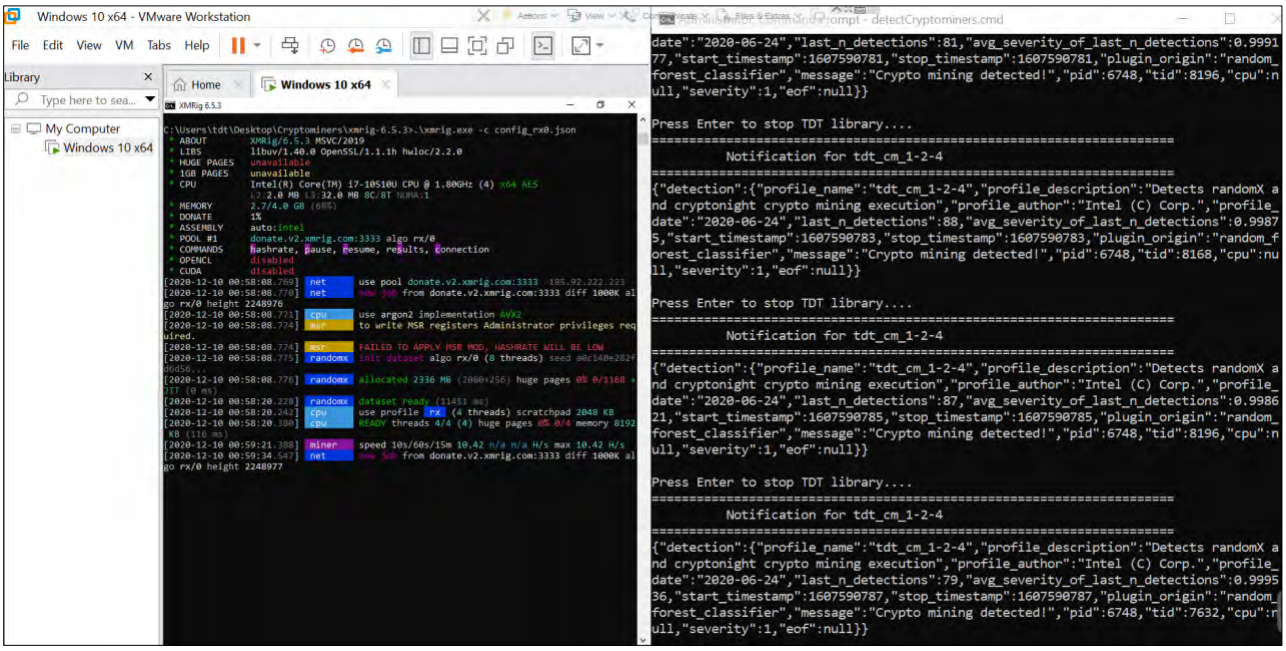
Test 1. Cryptomining in VM: 10th Gen Intel Core vPro processors with Intel TDT vs AMD

The following table shows where the Intel TDT Test App or a leading endpoint detection and response tool (EDRX) successfully identified Monero miners running inside a VMware workstation VM. This was performed on both 10th Gen Intel Core vPro processors and AMD hardware.

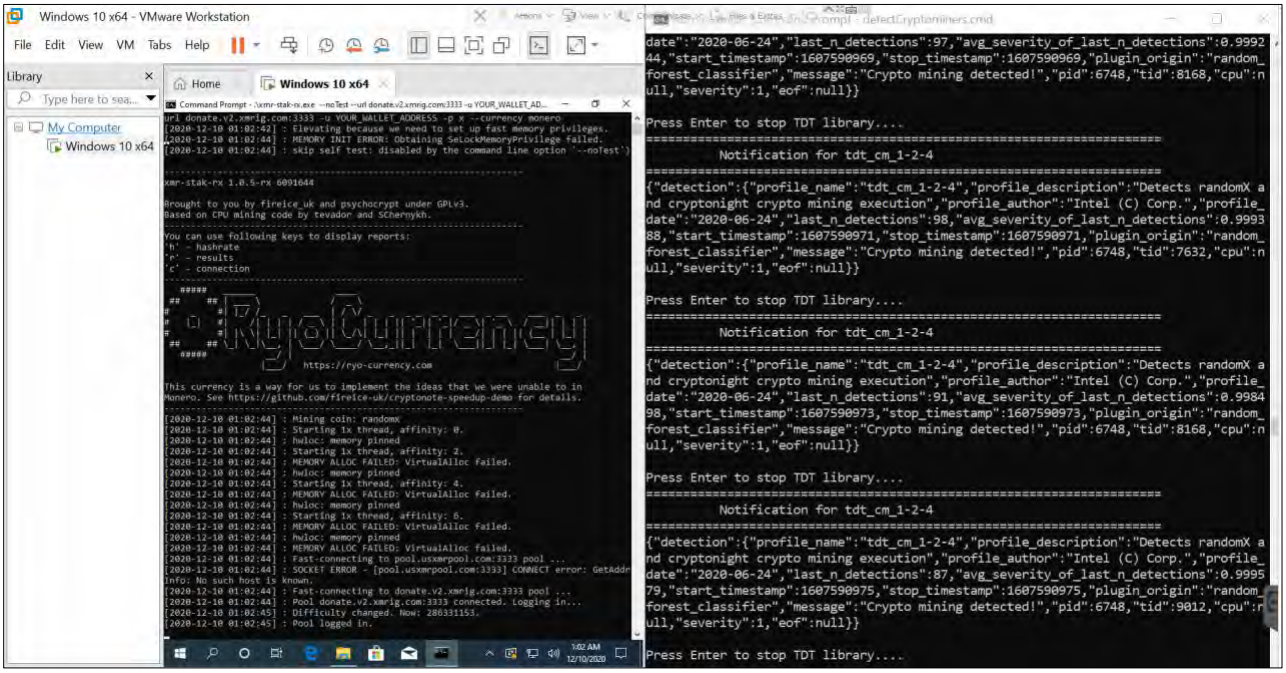
At the time these tests were performed, the test team did not have EDRX and Intel TDT Test App cannot configure the PMU on AMD, so no tests for this platform could be performed.

Sample ID	10th Gen Intel Core vPro processors	AMD
#C0A	Y	EDRX Not Available
#C17	Y	EDRX Not Available

The following screenshot shows a positive detection while running Sample #C0A:



The following screenshot shows another positive detection while running Sample #C17:



Test 2a. Cryptomining on Host: 10th Gen Intel Core vPro processors with Intel TDT

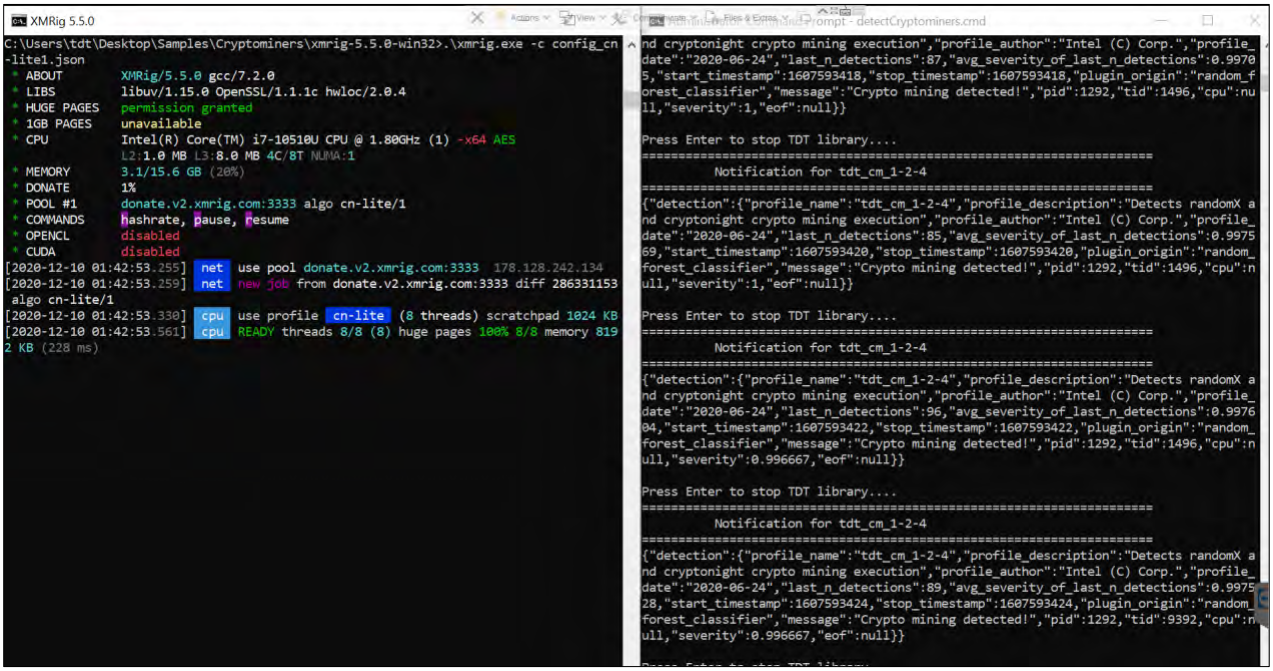
The following table shows where the Intel TDT Test App successfully identified cryptomining samples running on the same 10th Gen Intel Core vPro processors host.

Sample ID	10th Gen Intel Core vPro processors with TDT
#C0A	Y
#C0B	Y
#C0C	Y*
#C0D	Y*
#C0E	Y*
#C0F	Y
#C13	Y
#C14	Y
#C15	Y
#C16	Y
#C17	Y*
#C18	Y
#C19	Y

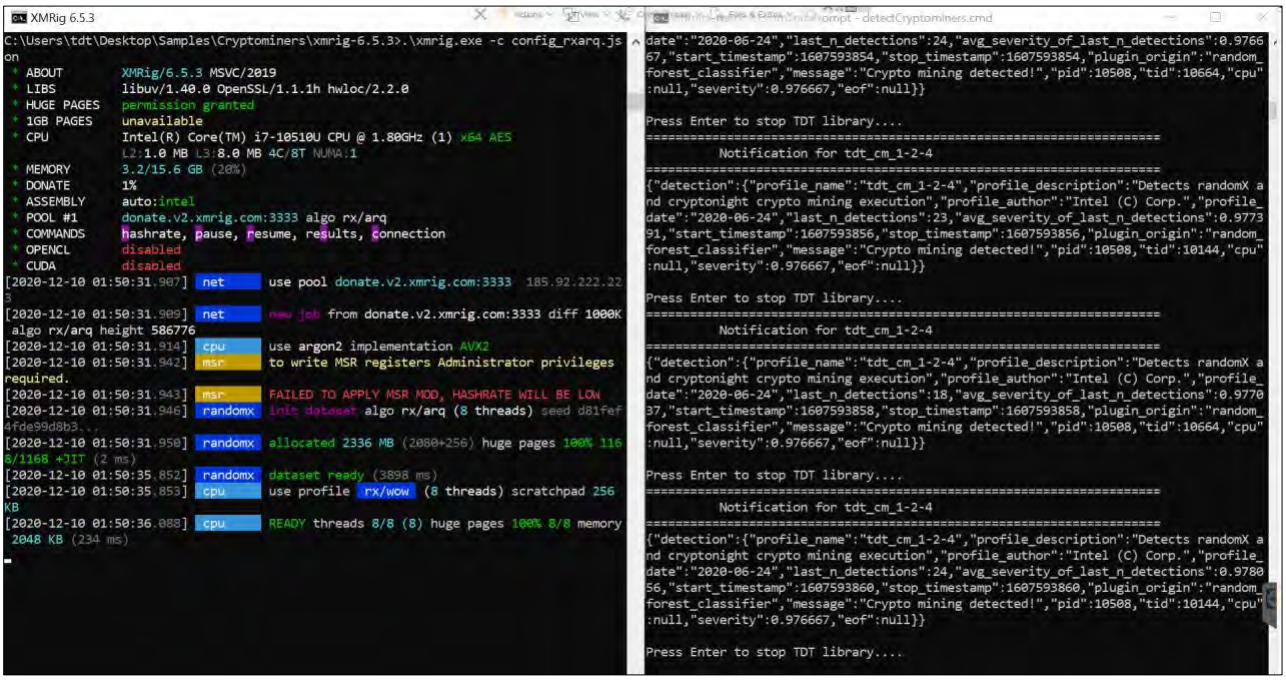
* These tests passed after allocation of sufficient CPU resources; however, the release version of Intel TDT will offload to the GPU, and therefore CPU resources should not hinder reliable detection.

Of particular importance in the above table is sample #C09, which is the only 32-bit Xmrig config which is successfully identified. In addition, there is sample #C17, which was detected within a VM, but not from the host.

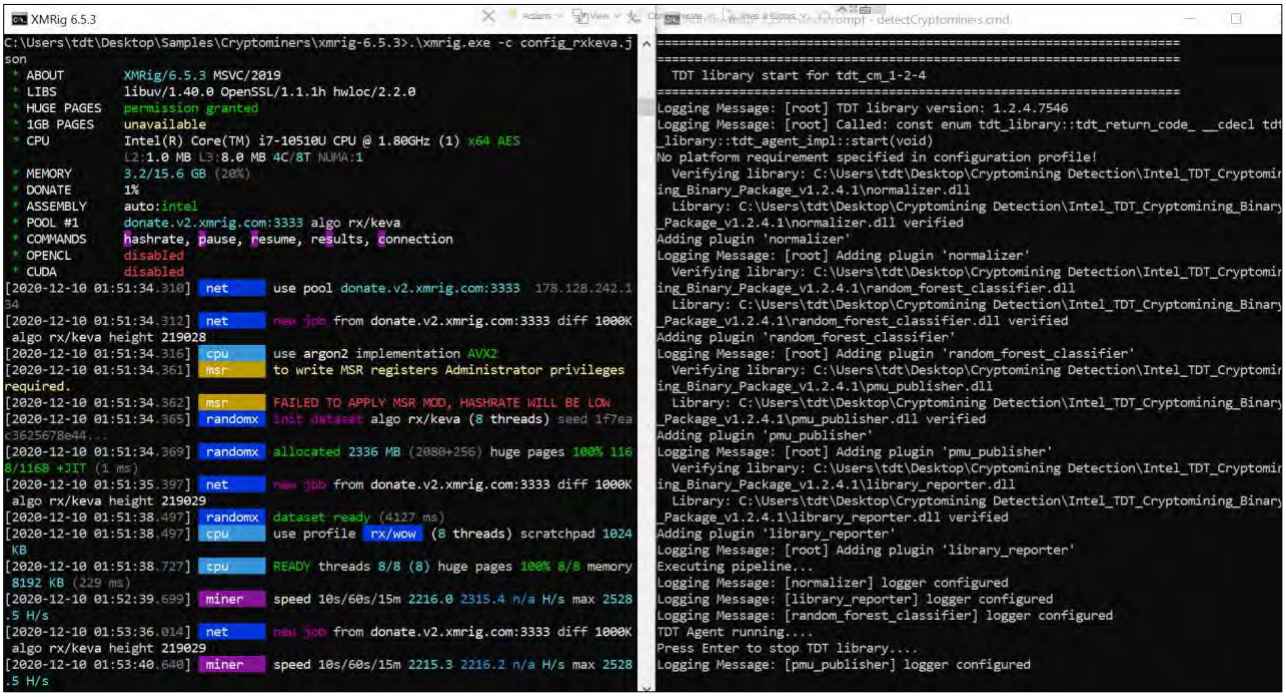
The following screenshot shows a positive detection while running Sample #C09:



The following screenshot shows a positive detection while running Sample #C0B:



The following screenshot shows the absence of any detection while running Sample #C0C:

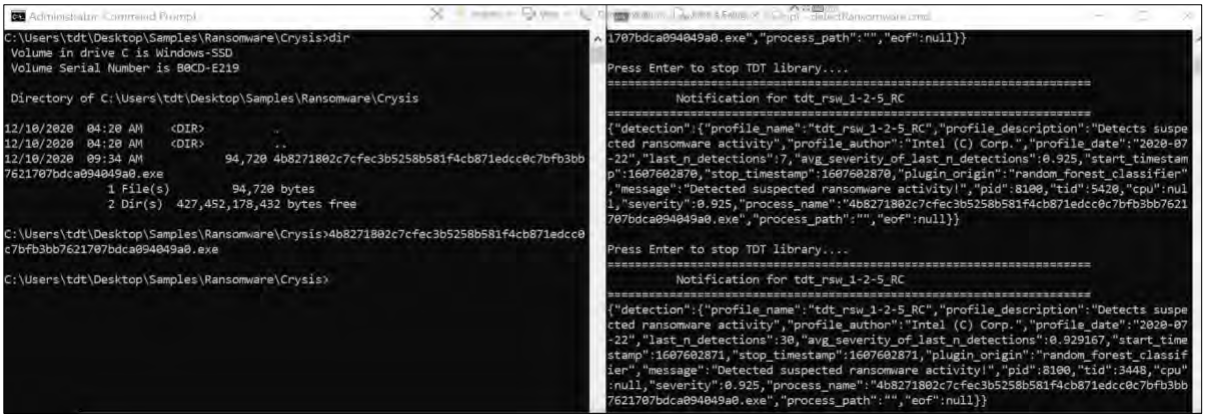


Test 2b. Ransomware on Host: Intel CML with Intel TDT

The following table shows where the Intel TDT Test App successfully found ransomware samples running on the same Intel CML host.

Sample ID	10th Gen Intel Core vPro processors with TDT
#R00	Y
#R01	Y
#R02	Y
#R03	Y
#R04	Y
#R05	Y
#R06	Y
#R07	Y
#R08	Y
#R09	Y
#R0A	Y
#R0B	Y
#R0C	Y
#R0D	Y

The following screenshot shows a positive detection while running Sample #R0A:



Test 3a Cryptomining on Host: Intel TDT vs. Current AV

The following table shows a comparison of successful detections of obfuscated cryptominer samples between Intel TDT and an industry-leading anti-malware product (X).

Sample ID	TDT	X
#OC0A	Y	N
#OC0D	N	N
#OC14	Y	N
#OC17	Y	N

Of particular importance in the above table is sample #OC17, which was detected but was not previously detected as part of Test 2a #C17.

Test 3b. Ransomware on Host: Intel TDT vs. Current AV

The following table shows a comparison of successful detections of obfuscated ransomware samples between Intel TDT (with an industry-leading anti-malware product disabled) and an industry-leading anti-malware product (X) (with Intel's TDT Test App disabled).

Sample ID	TDT	X
#OR00	Y	N
#OR02	Y	Y
#OR06	Y	Y
#OR09	Y	Y
#OR0A	Y	Y
#OR0D	Y	Y

Test 4. CML: Alternative to Test 3a and Test 3b

N/A (intentionally left blank)

Test 5.

N/A (intentionally left blank)

Test 6.

N/A (intentionally left blank)

About IOActive

IOActive is a comprehensive, high-end information security services firm with a long and established pedigree in delivering elite security services to its customers. Our world-renowned consulting and research teams deliver a portfolio of specialist security services ranging from penetration testing and application code assessment through to semiconductor reverse engineering. Global 500 companies across every industry continue to trust IOActive with their most critical and sensitive security issues. Founded in 1998, IOActive is headquartered in Seattle, USA, with global operations through the Americas, EMEA and Asia Pac regions. Visit www.ioactive.com for more information. Read the IOActive Labs Research Blog: http://blog.ioactive.com. Follow IOActive on Twitter: http://twitter.com/ioactive.

Addendum: Ryzen Pro 5000 Series Comparison

Management Summary

In this Intel-commissioned document, IOActive, Inc. (IOActive) presents a comparison of the security features provided and publicly documented by Intel® Corporation (Intel) in the 11th Gen Intel® Core™ vPro® mobile processors with AMD's Ryzen 5000 Pro mobile processor series based on public documentation from AMD.

Our comparison is based on a set of objectives bundled into five categories: Below the OS, Platform Update, Trusted Execution, Advanced Threat Protection (ATP), and Crypto Extension.

Intel vs AMD

In the Below the OS category, the subject AMD platform has no corresponding technology to Intel® System Security Report but does offer analogous capabilities to other Intel technologies.

In the Platform Update category, the subject AMD platform does not offer comparable capabilities to Intel BIOS Guard or Firmware Update Restart.

In the Trusted Execution category, 11th Gen Intel® Core™ vPro® mobile processors and the subject AMD architecture have equivalent capabilities, based on our research.

In the ATP category, we observed that the subject AMD platform does not offer comparable capabilities to Intel® Threat Detection Technology (Intel® TDT) in our comparative analysis. *We consider Intel TDT an impactful platform differentiator in this comparison*.

In the Crypto Extension category, the subject AMD platform has no corresponding technology to Intel's AVX512-variant of AES.

The composites of the security technologies discussed in this document offer a compounded value that is greater than the sum of the parts.

Our research team ran a series of tests for Intel TDT, based on install and executable instructions provided by the Intel TDT team. The tests aimed to detect a curated selection of samples of cryptominers and known ransomware in various environments and on different platforms.

The test results found Intel TDT had a detection rate of 100% for ransomware and 100% for cryptominers. Also, to better mimic threats that are increasingly obfuscating in VMs, we ran comparisons to popular anti-virus (AV) software that are not enabled for CPU-based threat detection. In these cases, Intel TDT was able to detect 75% of obfuscated cryptominers, compared to 0% detected by AV software, which has a lack of visibility into these types of attacks due to its typical deployment in the host OS.

Note: Intel TDT is not a standalone AV or EDR package; it is intended to integrate into these solutions to augment and improve threat detection efficacy.

Technical Summary

IOActive's analysis is based on publicly available documents from both Intel and AMD describing particular security technologies for the subject mobile PC-targeted architectures. We developed and used a security feature model to compare the two subject platforms, as described in the following Model and Comparison subsection.

Model and Comparison

Our approach for a comparison of features across vendor platforms started with the formulation of a security model. Our model consists of a carefully selected list of security objectives, bundled into categories. The objectives define goals and properties that provide security benefits to the customer. The categories relate to different execution stages of the CPU.

The following paragraphs define the categories and their objectives. They also list technologies or building blocks which can be used to achieve the objectives. Finally, a comparison of the corresponding technologies implemented by Intel and AMD is provided.

The model we present here is not exhaustive. It is missing a complete inventory of use-cases and would benefit from a thorough threat-model of the security features. IOActive adapted an organizational structure of objectives provided by Intel to create this model.

Each compared feature is presented as fulfilling, fulfilling with qualifications, or not fulfilling the associated objective.

Below the OS (Platform Integrity)

With the beginning of the boot sequence, the CPU must evaluate its hardware and firmware environment and ensure transition only to a verified bootloader. This integrity validation mitigates the risk of instruction tampering or interception by an adversary as well as providing a trustworthy baseline for the remainder of boot and operation.

In particular, the objectives include:

- Identify unauthorized changes to hardware and firmware
- Prevent malicious code injection in BIOS/UEFI memory
- Ensure OS and virtual environments are running directly on platform hardware (assuming no malicious code injection)
- Enforce OEM/IHV's provided policy and report on it

The main building blocks to achieve these objectives include:

- Secure Boot
    - Root of trust/chain of trust
    - Enforcement
- Measured Boot
    - Secure storage

- Static root of trust measurement
- Dynamic root of trust measurement
- Attestation

Table 6. Below the OS solutions

Intel Solution	AMD Solution
Intel® Boot Guard	AMD Secure Boot
Intel® Trusted Execution Technology	AMD SKINIT + Secure Loader
Intel® Runtime BIOS Resilience	AMD SMM Supervisor[48]
Intel® System Security Report	No equivalent feature
Intel® System Resources Defense	AMD SMM Supervisor[48]

Platform Update

This category is about mechanisms for trustworthy firmware updates. Each of these protections addresses the risk of replacement of some or all of the platform-defining software and firmware with malicious components.

- Objectives
  - Update firmware code with integrity check
  - Downgrade protection
  - Focus on BIOS for most recent and secure updates
- Building Blocks
  - Provide new environment so OEMs can perform more flexible and modular updates securely
  - Utilizes UEFI capsule architecture for driving better firmware updates

Table 7. Platform Update solutions

Intel Solution	AMD Solution
Intel® Firmware Guard	No equivalent feature[49]
Intel® BIOS Guard	No equivalent feature[49]

[48] As of publication, only a brief high-level description of this feature was available. The authors are provisionally accepting the assertions in this description.

[49] https://www.amd.com/system/files/documents/guardmi-infographic.pdf equates AMD secure boot to both of these Intel features, but this is not substantiated by the public documentation such as https://www.amd.com/system/files/TechDocs/40332.pdf

Trusted Execution/Application and OS Protection

In this category we list objectives for protection and prevention mechanisms to ensure a trustworthy runtime environment. Building on the integrity measures above, these elements enhance the stability and safety of the platform's operation.

- Objectives
  - Prevent memory corruption and tampering attacks
  - Protect sensitive data from unauthorized access
  - Protect data and virtualized containers with hardware-enforced isolation and encryption
  - Improve performance of virtualized security workloads
- Building Blocks/Hypervisor Support
  - Virtualization instructions
    - VM extensions
    - Nested virtualization
  - Virtual I/O
  - Virtual interrupts
  - Memory protection/encryption
  - Hardware-based encryption and random number generation

Table 8. Trusted Execution solutions

Intel Solution	AMD Solution
Intel® Virtualization Extensions (VT-x)	AMD-V
Intel® Virtualization Technology for Directed I/O (VT-d)	AMD-Vi
APIC Virtualization	Advanced Virtual Interrupt Controller (AVIC)
Mode Based Execution Control	AMD-V with GMET
Intel® Control-flow Enforcement Technology (Shadow Stack and Indirect Branch Tracking)	Shadow Stack Only
Intel® Total Memory Encryption	AMD SEV/SME[50]

[50] Multiple VMs can only share *unencrypted* memory (accessible to their common hypervisor).
See https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf, page 4

Advanced Threat Protection

The objectives in this category do not protect or prevent attacks, but allow detection.

- Objectives
  - Increase security and performance via offload to GPU dedicated security workloads
  - Leverage hardware telemetry to help detect advanced threats such as ransomware and cryptomining attacks
- Building Blocks
  - Reference code components
  - Detectors

Table 9. ATP solutions

Intel Solution	AMD Solution
Intel® TDT- Security Offload to iGPU (e.g., Memory Scanning)	No equivalent feature
Intel TDT - Advanced Platform Telemetry	No equivalent feature

Crypto Extension

This category lists objectives for hardware support for crypto primitives with specific properties.

- Objectives
  - Provide hardware implementations for certified crypto primitives
  - Mitigate side channels in the implementation of crypto primitives
  - Allow crypto operations without storing key where it can get lost
  - High speed and throughput
- Building Blocks
  - Good source of randomness (rdrand)
  - Efficient crypto primitives with side channel mitigation
  - Secure key storage (TPM, PKEY)
  - Parallelizable implementation of primitives
  - Hardware-assisted AES encryption
  - Hardware-assisted AES decryption
  - Hardware-assisted AES inverse mix column transformation
  - Hardware-assisted AES created round keys with key expansion schedule
  - Cryptographically secure enhanced non-deterministic random bit generator

- Cryptographically secure deterministic random bit generator

Table 10. Crypto Extension solutions

Intel Solution	AMD Solution
Intel® Advanced Encryption Standard New Instructions (Intel® AES-NI)	AMD AES-NI[51]
Intel® Secure Key	AMD RNRAND
Key Locker	No equivalent feature

[51] AMD does not appear to have a corresponding technology to Intel's AVX512-variant of AES

Appendix A: References

Coppolino Luigi, S. D'Antonio, G. Mazzeo, & L. Romano. 2019. "A comprehensive survey of hardware-assisted security: From the edge to the cloud." *Internet of Things*, Vol. 6. doi.org/10.1016/j.iot.2019.100055.

Li, Yuan, M. Wang, C. Zhang, X. Chen, S. Yang, & Y. Liu. 2020. "Finding Cracks in Shields: On the Security of Control Flow Integrity Mechanisms." *CCS '20: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. https://dl.acm.org/doi/10.1145/3372297.3417867.

Mengyuan, Li, Y. Zhang, Z. Lin, & Y. 2016 "Exploiting Unprotected I/O Operations in AMD's Secure Encrypted Virtualization." *Proceedings of the 28th USENIX Security Symposium*. https://www.usenix.org/system/files/sec19-li-mengyuan_0.pdf.

Morbitzer, Mathias, M. Huber, & J. Horsch. 2019. "Extracting Secrets from Encrypted Virtual Machines." *CODASPY*. https://arxiv.org/pdf/1901.01759.pdf.

Silva, Otávio, & P. Lício de Geus. 2020. "Lokke, a hybrid security hypervisor." *Universidade Estadual de Campinas*. https://www.lasca.ic.unicamp.br/paulo/papers/2020-SBSeg-otavio.silva-lokke.hybrid.supervisor.pdf.

https://www.amd.com/system/files/TechDocs/40332.pdf

https://www.amd.com/system/files/TechDocs/55766_SEV-KM_API_Specification.pdf

https://cdrdv2.intel.com/v1/dl/getContent/631121

https://lkml.org/lkml/2020/7/6/1020

