Confidential VMs Explained: An Empirical Analysis of AMD SEV-SNP and Intel TDX

MASANORI MISONO and DIMITRIOS STAVRAKAKIS, Technical University of Munich, Germany
NUNO SANTOS, INESC-ID & Instituto Superior Técnico, University of Lisbon, Portugal
PRAMOD BHATOTIA, Technical University of Munich, Germany

Confidential computing is gaining traction in the cloud, driven by the increasing security and privacy concerns across various industries. Recent trusted hardware advancements introduce Confidential Virtual Machines (CVMs) to alleviate the programmability and usability challenges of the previously proposed enclave-based trusted computing technologies. CVM hardware extensions facilitate secure, hardware-isolated encrypted VMs, promoting programmability and easier deployment in cloud infrastructures. However, differing microarchitectural features, interfaces, and security properties among hardware vendors complicate the evaluation of CVMs for different use cases. Understanding the performance implications, functional limitations, and security guarantees of CVMs is a crucial step toward their adoption.

This paper presents a detailed empirical analysis of two leading CVM technologies: AMD Secure Encrypted Virtualization–Secure Nested Paging (SEV-SNP) and Intel Trust Domain Extensions (TDX). We review their microarchitectural components and conduct a thorough performance evaluation across various aspects, including memory management, computational performance, storage and network stacks, and attestation primitives. We further present a security analysis through a trusted computing base (TCB) evaluation and Common Vulnerabilities and Exposures (CVE) analysis. Our key findings demonstrate, among others, the effect of CVMs on boot time, memory management and I/O, and identify inefficiencies in their context switch mechanisms. We further provide insights into the performance implications of CVMs and highlight potential room for improvement.

CCS Concepts: • **General and reference → Surveys and overviews**; • **Security and privacy → Virtualization and security**.

Additional Key Words and Phrases: confidential computing, virtual machine, AMD SEV-SNP, Intel TDX

ACM Reference Format:
Masanori Misono, Dimitrios Stavrakakis, Nuno Santos, and Pramod Bhatotia. 2024. Confidential VMs Explained: An Empirical Analysis of AMD SEV-SNP and Intel TDX. *Proc. ACM Meas. Anal. Comput. Syst.* 8, 3, Article 36 (December 2024), 42 pages. https://doi.org/10.1145/3700418

1 Introduction

Context: Confidential computing in cloud environments. Confidential computing [39] has become a crucial component of cloud environments to ensure secure data and computations across various sectors, including finance [26, 131], healthcare [36, 200], and industrial applications [79, 145, 148, 181]. The initial commercially available confidential computing hardware extensions introduced application-level enclave-based Trusted Execution Environments (TEEs) [85, 116], such

Authors' Contact Information: Masanori Misono, masanori.misono@in.tum.de; Dimitrios Stavrakakis, dimitrios.stavrakakis@tum.de, Technical University of Munich, Munich, Germany; Nuno Santos, nuno.m.santos@tecnico.ulisboa.pt, INESC-ID & Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal; Pramod Bhatotia, Technical University of Munich, Munich, Germany, pramod.bhatotia@tum.de.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2024 Copyright held by the owner/author(s).
ACM 2476-1249/2024/12-ART36
https://doi.org/10.1145/3700418

as Intel SGX [108], RISC-V Keystone [122], and Arm TrustZone [24]. Despite their strong security properties, enclave-based TEEs were not overwhelmingly adopted by cloud users because of high programmability overheads, among other issues. In particular, enclave-based TEEs necessitate code modifications and application redesigns tailored to their architectural features [178, 192], posing programmability and usability challenges.

To overcome the limitations of enclave-based TEEs, major hardware vendors have proposed new trusted VM-level ISA extensions, e.g., Intel TDX [112], AMD SEV-SNP [21], and *upcoming* Arm v9.0 CCA [23], leading to the advent of Confidential Virtual Machines (CVMs) [11, 23, 91, 94, 112, 159, 172]. CVMs overcome the limitation of programmability and employability of application-level enclaves, such as Intel SGX, by proposing a new trustworthy virtual machine abstraction. Several major cloud providers are already offering CVM instances [78, 82, 93, 143, 158, 177], which highlights CVMs' importance as a fundamental building block for confidential computing in modern cloud environments.

Motivation: Same CVM abstraction, but different hardware and software properties. CVMs provide a confidential computing abstraction at the virtual machine level, allowing unmodified applications to run on it with existing software stacks while protecting data in use from unauthorized access, even from the hypervisor hosted by the cloud provider. Therefore, applications can be seamlessly deployed in CVMs, thus promoting programmability and enabling an easier adoption of trusted computing in the cloud.

On the other hand, CVMs, offered by various hardware vendors, provide fundamentally different microarchitectural features, hardware interfaces and security properties, despite their aim to offer the same VM-based abstraction for trusted computing. Such differences mandate new VM management schemes, leading to changes in the system software stack for both the host and guest environments. On top of that, for cloud users and developers, systematically evaluating the suitability of a CVM based on potential performance implications, supported functionalities, and security properties is challenging due to the diverse underlying hardware architectures. To this end, understanding the performance characteristics, functional limitations, and security features of CVMs is crucial for their adoption. Although several studies analyze CVMs, they mainly perform literature reviews [4, 35, 116, 171] or focus on evaluating a single CVM technology (e.g., AMD SEV(-SNP)) [5, 89, 123, 149, 196]. Our paper fills the gap by providing a practical and comprehensive empirical analysis of CVMs.

Our Approach: An Empirical CVM analysis of the hardware and software system stacks. We conduct a detailed empirical analysis of two widely-used commercial CVM technologies, AMD SEV-SNP, and Intel TDX. Initially, we thoroughly review their (micro)architectural components to highlight their functionalities and how they interact to achieve the security goals of CVMs. Following, we demonstrate our experimental results, obtained through a series of micro- and macrobenchmarks, to identify the performance characteristics and implications of CVMs on various workload scenarios and use cases. We examine several CVM aspects, including memory management, computational performance, storage and network stacks, and attestation primitives. Lastly, we analyze the security features of these CVM technologies and their trusted computing base (TCB) size and present our Common Vulnerabilities and Exposures (CVE) analysis. This analysis aims to showcase the suitability and caveats of AMD SEV-SNP and Intel TDX for security-critical workloads.

Our key findings: Through our extensive architectural analysis and experimental evaluation, we identify the following key findings for the examined CVM technologies:

- **Slow boot time:** Booting a CVM can take over twice as much as booting a standard VM. In addition to the additional procedure to launch a CVM, the host-side memory management predominantly affects boot time.
- **Memory allocation tax:** The CVM-specific *memory acceptance* operation is performance-heavy, and our memory allocation microbenchmarks show up to 92% increase for the *initial* memory allocation of the CVMs.
- **Costly context switch:** If an application relies on frequently repeating sleep and wake-up events, the vCPU-sleep (HLT) overhead of the CVM can cause a significant performance drop. Our benchmarks show up to 431% increase in processing in the worst case (e.g., NPB). We further show that enforcing guest-side polling can help mitigate this issue.
- **I/O overhead:** When the CPU utilization is high, the I/O overhead can be significant due to the internal implementations of the default I/O software stacks using bounce buffers. We observe up to 60% performance drop for heavy network processing benchmarks (iperf TCP).
- **Large TCB:** The TCB size of a CVM typically includes millions of Lines of Code (LoC) as it contains the full-fledged operating system (OS) of the guest, thus increasing the attack vectors.
- **New security attack vectors:** 39% of the CVEs related to AMD SEV-SNP and Intel TDX are attributed to improper validation mechanisms, while 54% of the CVEs are associated with vulnerabilities in the underlying firmware. Additionally, 8 CVEs refer to attacks from the guest to the host and from the host to the host.

Implications on the microarchitectural hardware and software stack. There is a need to reconsider fundamental confidential computing concepts even though the hardware, firmware, and supporting software stacks constantly evolve to support more functionalities and extend their security guarantees. Our study can serve as a stepping stone toward enhancing cloud environments' security, applicability, and performance utilizing CVMs. In particular, based on the outcomes of our study, we consider the following areas for improvement:

- **Reducing and optimizing VMEXIT impact:** In the context of CVMs, VMEXITs are expensive. Reducing the number of VMEXITs and minimizing VMEXIT processing time is critical for performance. From a software perspective, implementing a sophisticated and adaptable polling policy is desirable, while optimizing VMEXIT processing is essential from a hardware perspective.
- **Designing new boot scheme:** Improving the bootup times of CVMs through specially designed HW/SW co-designed system stacks is essential. Such an improvement will increase the applicability of CVMs in the cloud and facilitate further use cases for CVMs (e.g., serverless computing).
- **Optimizing I/O stacks:** I/O stacks are vulnerable points of CVMs, as all the data transfers must be checked and validated. However, current approaches incur significant performance degradation. Therefore, designing optimized CVM-aware I/O stacks should become a priority.
- **Unifying attestation primitives:** Each CVM technology has its own, sometimes quite convoluted, attestation primitives. A collaborative effort to unify the attestation process is necessary and would help to standardize a trustworthy CVM deployment process in the cloud.
- **Reducing and hardening TCB:** CVMs have inherently large TCBs, which can widen their attack surface due to the high number of inputs from the host. Hardening and reducing the TCB size is crucial to minimize security risks (e.g., hardened Linux kernel, minimal LibOS).
- **Open-sourcing firmware:** Making the platform firmware (ASP firmware / TDX module) ecosystem fully open-source would be beneficial for transparency and bug-fixing reasons. Furthermore, since it is written in C, it would be advantageous to use a memory-safe language such as Rust.
- **Testing new interfaces:** New CVM software stacks can introduce an attack from the guest to the host and the host to the host as well, and testing that interface is also essential.

Contributions. To the best of our knowledge, this paper is the first systematic study of modern confidential computing architectures that are widely used in current cloud infrastructures. Importantly, we conduct both an extensive architectural and security analysis of CVMs, and practical experiments on real hardware. More specifically, our paper makes the following contributions:

(1) We present in-depth the architectural characteristics of AMD SEV-SNP and Intel TDX, the core confidential virtual machine technologies of the x86-64 platforms (§ 3).
(2) We empirically evaluate the performance of AMD SEV-SNP and Intel TDX using real hardware (4th-generation AMD EPYC Processors, 5th-generation Intel Xeon Scalable Processors) across multiple dimensions to cover various use cases and application scenarios (§ 4).
(3) We present a thorough security analysis of AMD SEV-SNP and Intel TDX, examining the TCB size and the reported CVM-related CVEs (§ 5).
(4) We make our automated evaluation framework publicly available to facilitate further research endeavors in CVMs. The evaluation code is available at https://github.com/TUM-DSE/CVM_eval.

Disclaimer. This paper does not raise any ethical concerns. Intel Corporation provided us with the Intel TDX machine for our experiments and approved the results for publication. The text solely reflects the opinion of the authors.

2 Overview

2.1 Confidential Virtual Machines (CVMs)

Confidential virtual machines (CVMs) provide a confidential computing abstraction at the level of a virtual machine. The entire VM runs as a trusted execution environment (TEE), protecting code and data in use from unauthorized access, even from the hypervisor or the virtual machine monitor (VMM) that manages the CVM. The enlightened guest kernel operates within the VM and can run unmodified applications. These characteristics offer better programmability and usability than the traditional application-based TEEs [24, 43, 108, 122, 197]; applications can utilize the existing software stacks as in a normal VM. In addition, CVMs provide remote attestation functionalities, allowing a CVM owner to ensure that it is launched in the intended state and runs on the intended hardware.

Major CPU vendors introduce CVMs support, including AMD SEV-SNP [11], Intel TDX [112], IBM Z SE [94], OpenPower PEF [91], and ARM CCA [23], in addition to work on RISC-V such as Confidential VM Extensions (CoVE) [172], and Assured Confidential Execution (ACE) [159]. Among them, AMD SEV-SNP and Intel TDX are now available to the public, and major cloud providers have started providing instances based on them [78, 82, 93, 143, 158, 177].

2.2 Threat Model and Assumptions

We detail a general threat model for CVMs. In contrast to traditional VMs, CVMs do not trust the virtual machine monitor (VMM) or the host system. The host and the VMM can be malicious and try to access and compromise the guest memory or the CVM state, yet they manage CVMs. To protect CVMs in such an environment, the system platform offers trusted components and mechanisms that assure the confidentiality and integrity of CVMs, including their memory and register states. However, this does not directly protect I/O traffic outside of the CVMs, and the guest application must use appropriate encryption methods if necessary.

More specifically, adversaries may gain full control of the boot firmware, including the System Management Mode (SMM), the host operating system, the hypervisor, peripheral devices, and non-confidential legacy VMs. They can also have full control of inputs to the CVMs, including device I/O and interrupts. Physical attacks, such as fault injection attacks and roll-back attacks

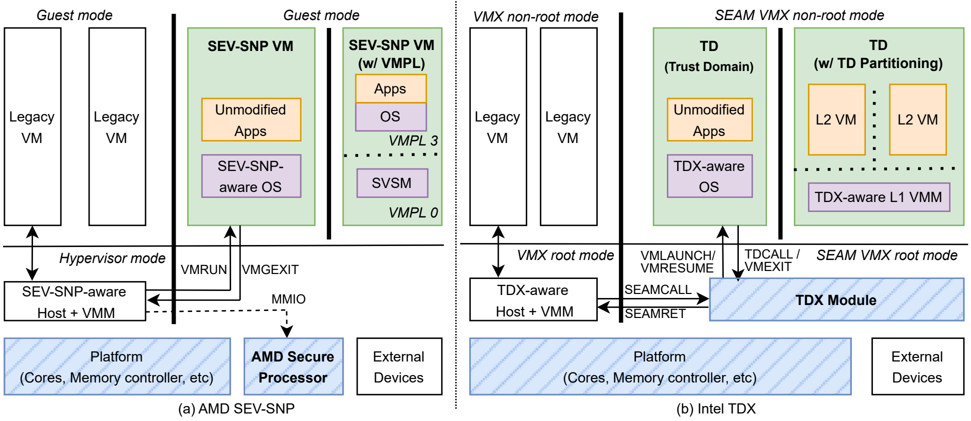


Fig. 1. The architecture of AMD SEV-SNP (left) and Intel TDX (right). The green regions denote CVMs, with unmodified software (orange) and modified software (purple). The blue-hatched regions are trusted components. The thick line indicates each CVM is isolated from the host and other VMs with encryption.

to DRAM regions, are considered out of the scope. However, CVMs can neutralize some physical attacks, such as cold-boot attacks [86].

Tenants trust the manufacturer and the system platform (processors, memory controllers, etc.). In addition, the software running in a CVM is part of the TCB. CVMs are responsible for protecting data in transit (network) and at rest (storage) using appropriate measures such as encryption protocols (e.g., TLS). Besides, protecting against side-channel attacks is out of the direct scope of CVMs. CVMs should take appropriate countermeasures against side-channel and speculation attacks due to their micro-architectural implementation if necessary [7, 14, 47, 97, 98, 186, 189].

Last but not least, the system does not guarantee the availability of CVMs, and denial of service attack (DoS) is out of the scope of the threat model. For example, a malicious VMM can choose not to schedule a CVM. In general, CVMs assume that the system is “benign but vulnerable” [11], and the availability during normal times is ensured. This assumption is reasonable given that cloud providers offer services to users under certain service level agreements (SLAs) and, therefore, have a strong incentive to maintain availability.

2.3 AMD SEV-SNP

AMD introduced AMD SEV in the 1st generation of EPYC processors, which only supports memory encryption [12] and is subject to several attacks including data extraction and code injection [32, 87, 126, 127, 129, 151, 152, 188, 190, 191]. Later, AMD added the VM register state encryption (AMD SEV-ES) [9], and memory integrity protection (AMD SEV-SNP) [11] along with several security improvements to address these issues. AMD SEV(-ES) and SEV-SNP have different ABI[1] to configure and manage VMs [10, 18]. This paper explicitly uses AMD SEV, AMD SEV-ES, and AMD SEV-SNP to refer to each version and focuses on the analysis of AMD SEV-SNP.

AMD Secure Processor (ASP). AMD SEV(-ES/SNP) uses a dedicated processor called AMD Secure Processor (ASP), also known as AMD Platform Security Processor or PSP, to manage its security features. ASP is an ARM-based processor separated from the main x86 cores but directly integrated into the CPU die, creating a hardware root-of-trust [2, 8, 20, 33]. The ASP securely

[1] AMD calls SEV API instead of ABI. In this paper, we use ABI to refer to the interface of ASP.

manages SEV-SNP VM encryption keys and the reverse map table (§ 3.3) to ensure the integrity of the guest address translation. The ASP also provides remote attestation functionalities.

Overview. Figure 1 (a) shows the overview of AMD SEV-SNP. An SEV-SNP VM runs in an isolated environment, where its memory and state are encrypted with an ASP-managed unique key. While the VMM allocates and manages the resources of an SEV-SNP VM, ASP is responsible for encrypting the guest state and protecting the guest address translation integrity. The host communicates with the ASP through an MMIO-based interface to configure SEV-SNP VMs [18]. During SNP VM execution, SMI (System Management Interrupt) is blocked, and there is no direct transition to System Management Mode (SMM) from the SNP VM. Even the SMM cannot access the SNP VM.

VMPL. AMD SEV-SNP introduces a new feature called VMPL (Virtual Machine Protection Level) [11]. VMPL provides an additional privilege level to the guest that is orthogonal to the traditional x86 ring protections. There are 4 VMPLs, and the higher VMPL can trap several events from the lower VMPL. AMD standardizes the software running in the highest VMPL level (VMPL0), called SVSM (Secure VM Service Module) [16]. SVSM is intended to offer several services that the VMM traditionally provides, including vTPM and live migration [13, 155].

2.4 Intel TDX

Intel introduced Intel TDX in the 4th generation of Xeon Scalable Processors as a private preview and made it publicly available in the 5th generation of Xeon Scalable Processors. One of the main architectural differences between Intel TDX and AMD SEV-SNP is that Intel TDX introduces a new CPU mode (SEAM) and runs a trusted software (TDX module) instead of using microcode with the help of an external secure processor to manage CVMs.

SEAM mode and TDX module. Intel introduces two new components: a new CPU operation mode *Secure Arbitration Mode (SEAM)*, and special trusted software the *TDX module*. A Trust Domain (TD) is a virtual machine that runs in a secure environment under the control of the TDX module. SEAM is subdivided into VMX SEAM root mode and VMX SEAM non-root mode. A TD operates in VMX SEAM non-root mode, while, the TDX module is running in VMX SEAM root mode.

Overview. Figure 1 (b) shows the overview of Intel TDX. A TD runs in an isolated environment, where its memory and state are encrypted with a unique key. The TDX module intermediates the communication between the host and TDs and performs necessary state encryption/decryption and restore operations. The host calls the TDX module using `SEAMCALL`, whereas the TD calls the TDX module using the `TDCALL` instruction. VMEXIT events also transfer the control to the TDX module, and then it transfers control to the host when necessary (the transition is called `SEAMRET`). Similarly to AMD SEV-SNP, there is no direct transition to System Management Mode (SMM) in the SEAM mode. Depending on the configuration, SMI causes VMEXIT events or is blocked.

TD partitioning. Intel TDX 1.5 introduces a new feature called TD-partitioning [106], which allows TD to have up to three nested VMs with one L1 VMM. Similar to VMPL, TD-partitioning aims to realize features that, traditionally, the VMM provides, such as vTPM and live migration [102, 103].

2.5 Software Stacks

CVMs require new software stacks for their management, communication, and ensuring security properties. First, the host kernel and the VMM are responsible for managing CVMs. The host interacts with the ASP or the TDX module with ABIs [18, 105]. Both AMD and Intel develop the host kernel and VMM based on Linux, KVM, and QEMU, the most widely used virtualization toolchain.

In turn, the guest firmware is responsible for initializing CVMs in the guest context, including configuring private memory and performing the appropriate measurements (§ 3.1). AMD and Intel develop the guest firmware with OVMF (UEFI) [60]. Intel calls this firmware TDVF. Intel also develops TD-shim [65], a dedicated minimal firmware for TDs. Furthermore, the guest kernel needs to be CVM-aware to communicate with the host using a dedicated communication protocol (§ 3.2). In addition, the guest must treat the inputs from the VMM as untrusted and validate them. The inputs include device I/Os, interrupts, certain CPU instructions (e.g., `cpuid`, `rdmsr`), and ACPI tables.

Lastly, the guest applications must use appropriate measures to protect sensitive data outside of CVMs. For networking, the guest can use encrypted communication protocols such as TLS [95]. For storage, the guest has to encrypt data before writing to the external disk, with integrity and rollback protection if required. Linux's dm-crypt [53] is widely used as a full disk encryption. Linux's dm-integrity [54] (for read/write data) and dm-verity [55] (for read-only data) provide integrity protection at a block level using Merkle trees [142]. The guest can further leverage trusted remote services based on replication or append-only ledgers to ensure rollback protection [22, 71, 90, 133, 134, 156].

3 Architecture Analysis

In this section, we analyze the architectural details and highlight the main components that are crucial to understanding the characteristics of AMD SEV-SNP and Intel TDX.

3.1 VM Initialization and Boot

The main difference between booting CVMs and normal VMs is that the VMM explicitly adds the initial guest code, data and state using the ABI defined by the ASP or TDX module. ASP and TDX modules calculate the measurement (running hash) of added components, which is included in the attestation report (§ 3.5) for validation purposes. Once the CVM is launched, the VMM cannot directly access the guest memory or alter the guest state. After the CVM launch, the guest firmware and the kernel are responsible for initializing the guest. This includes validating the guest memory and enabling encryption by properly configuring the guest page tables (§ 3.3).

Measured boot. The guest launch measurement only covers the initial state. To compensate, the measured boot records loaded component measurements during boot for verification later. The Trusted Computing Group (TCG) standardizes measured boot protocol using TPM [83]. In a virtualized environment, a VM use a virtual TPM (vTPM). However, the traditional vTPM is inappropriate for CVMs since the VMM manages it [28]. Several works propose vTPMs for CVMs [144, 155, 163], or non-vTPM based measured boot [77, 84]. Additionally, both AMD SEV-SNP and Intel TDX provide their own way of performing measured boot.

AMD proposes a measured direct boot [153, 154], where the VMM additionally inserts the hash of the kernel, initrd, and kernel parameters into the initial guest memory. Thus, they become part of the measurement in the attestation report. This boot method is applicable if the kernel code and boot parameter are not confidential. If they are confidential, the guest must boot from an encrypted disk [30]. On the other hand, Intel TDX introduces new measurement registers, called Runtime Extendable Measurement Registers (RTMRs). The guest firmware and the OS can extend them using TDCALLs [96]. The RTMRs are then included in the attestation report.

3.2 VM-VMM Communication

The traditional VM-VMM communication relies on the VMM consulting the guest state upon VMEXIT, which is prohibited for CVMs. Therefore, AMD SEV-SNP and Intel TDX introduce a new communication protocol [17, 101], allowing the guest to explicitly pass the needed state

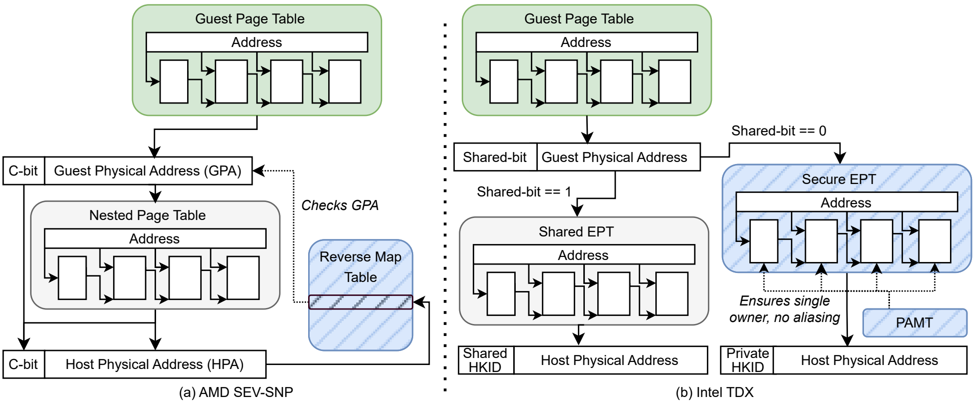


Fig. 2. Memory translation on AMD SEV-SNP and Intel TDX. The blue-hatched regions are trusted.

to the VMM when requesting an action. In particular, AMD SEV-SNP defines GHCB (Guest-Hypervisor Communication Block) format for the communication [17], and the guest uses the VMGEXIT instruction to call the VMM. On the other hand, Intel TDX Module ABI [105] defines the communication protocol for the TD, and the guest uses TDCALL for the communication. While VMGEXIT instruction always transfers the control to the VMM, some of the TDCALL calls are handled by the TDX module without switching the context to the VMM.

To minimize the modification to the guest software, certain traditional VMEXIT events raise an exception in the guest. Specifically, AMD SEV-SNP raises VMM Communication Exception (#VC). On the other hand, for Intel TDX, the TDX module first handles VMEXIT events and, if unable to do so, raises a Virtualization Exception (#VE) to the guest. The guest's exception handler explicitly calls the VMM when necessary. Some events unconditionally cause VMEXIT to the VMM as in a normal VM if the guest does not need to pass the state (e.g., external interrupts).

3.3 Memory Management

CVMs adapt the memory management mechanisms to ensure memory confidentiality and integrity.

Confidentiality. Figure 2 (a) shows the address translation flow in AMD SEV-SNP. AMD SEV-SNP bases its memory encryption on AMD Secure Memory Encryption (SME) [12]. The upper bit of the guest's physical address is a C-bit, controlling if the page is encrypted (C-bit is 1) or shared with the VMM (C-bit is 0). The host's physical address also has a C-bit. The guest's C-bit has higher priority than the host's C-bit. If C-bit is set only in the host, the memory is still shared but encrypted with the host key, which is effective in protecting against some types of attacks, such as cold boot attacks.

Figure 2 (b) shows the address translation flow in Intel TDX. Intel TDX uses two nested paging structures (EPTs) to manage address translation. One is shared EPT for shared memory, and the other is secure EPT for private memory. The upper bit of the guest's physical address (shared-bit) specifies if the page is shared or not. On the host, the memory is encrypted with Intel Total Memory Encryption–Multi-Key (TME-MK) [99], which uses the upper bits of the physical address as the key ID (HKID; Host Key ID). Intel TME-MK splits the HKIDs into two groups: private key IDs and shared key IDs. Shared HKIDs are only used for shared pages. When initializing the guest, the TDX module assigns a unique private key ID (HKID) to the TD. If the host BIOS enables *TME bypass*, it turns off the encryption for the HKID 0, thus removing the encryption for the normal host pages.

Integrity. To ensure memory integrity, AMD SEV-SNP introduces a reverse map table (RMP). The RMP records the mapping from the host physical address (HPA) to the guest physical address (GPA) with the owner. To maintain address translation integrity, the guest must explicitly *accept* a translation from GPA to HPA, before using that memory region. The memory controller uses the RMP to ensure that only a single GPA is mapped to a single HPA, and when accessing memory, the controller checks the RMP to ensure valid access. The results of RMP checks are cached in the TLB. ASP manages the RMP, and the VMM can update the RMP using the SEV-SNP firmware ABI [18].

A TD also explicitly accepts memory using TDCALL before use. Additionally, Intel TDX uses a Physical Address Metadata Table (PAMT) to ensure that a physical page in the secure memory is assigned to at most one TD with no aliasing. In addition, Intel TDX introduces one TD Owner bit for each cache line to protect memory integrity. The TD Owner bit is set to 1 when writing with a private HKID. When reading memory, the hardware checks if the TD Owner bit is set. Unauthorized write to the memory clears the TD Owner bit and the next read from this memory region in a TD will trigger a VMEXIT. In addition, by default, the cache lines are protected by a MAC (Message Authentication Code) [100], which protects against bit-flips (e.g., Rowhammer attacks) [118].

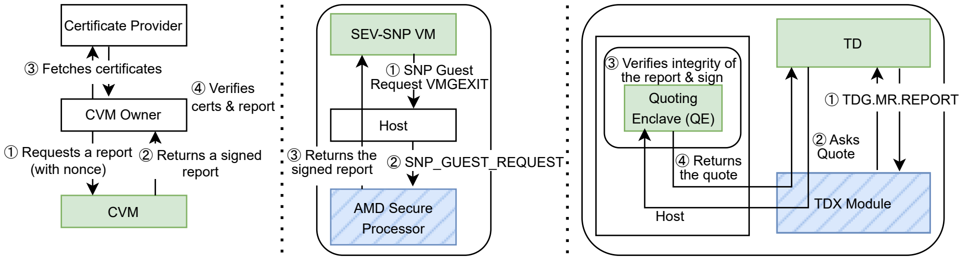
3.4 I/O Operations

I/O operations involve interaction with untrusted external entities and thus require special care to handle. There are three types of I/O operations: IO instruction (`in`/`out` instructions), memory-mapped IO (MMIO), and direct memory access (DMA). First, IO instructions and MMIO are only doable via the explicit VMM call (§ 3.2). Normal IO instructions cause #VC/#VE, and the VMM configures the MMIO region so that accessing it causes #VC/#VE. In addition, DMA requires further care as external devices cannot access the guest's private memory. Therefore, the guest uses a dedicated shared buffer (bounce buffer [169], also known as *s*wiotlb in Linux) for DMA. This means DMA operations have additional copies in the CVM environment, impacting their performance. The VMM manages and injects interrupts to the guest.

Input validation. All input from an external device should be treated as untrusted; therefore, the guest device driver must validate all input values and handle potential anomalies accordingly [166]. The validation includes (1) performing a range check to ensure the values are within the expected range, (2) having timeout if an input is not responsive for a certain amount of time, and (3) avoiding double fetch [184]. Several research works show that device drivers tend to have many validation misses from the external inputs [88, 146, 147, 174, 175, 179], showing the importance of device driver hardening. In the context of CVMs, the Linux kernel developers work on VirtIO hardening [170, 180]. In addition, Intel recommends having a device filter mechanism that white-lists the devices to be used by the guest [40].

Direct device assignment. The direct device assignment allows the VM to directly access devices, realizing full hardware performance. However, in the CVM trust model, untrusted devices cannot directly access the guest's private memory. Therefore, they must use a bounce buffer for communication.

To extend the trust to a device and remove bounce buffer overhead, there are ongoing efforts to enable direct device assignments for CVMs on top of TDISP (TEE Device Interface Security Protocol) [162] specified by PCI-SIG. TDISP utilizes the SPDM [73] protocol to attest the device and establish a secure connection, and PCIe IDE [161] to establish an encrypted and integrity-protected communication over the PCIe transaction layer. With TDISP, the trusted PCIe device can directly perform DMA to the CVM's private memory. The current 4th Generation EPYC Processors and 5th Generation Intel Scalable Processors do not support TDISP but future processors will incorporate this capability [15, 104]. Intel calls the direct device assignment feature with TDISP TDX-Connect [104], whereas AMD calls it SEV-TIO [15].



(a) Basic Attestation Flow (b) AMD SEV-SNP (c) Intel TDX

Fig. 3. Remote attestation. (a) Basic attestation flow. The CVM owner gets an attestation report from a CVM. The report is signed by the known key by AMD or Intel. The owner can verify the report by checking the certificate chain and its contents. (b), (c) The attestation report generation of AMD SEV-SNP and Intel TDX.

GPU TEEs. Several recent works propose GPU TEEs [25, 48, 70, 92, 115, 117, 132, 183, 198, 201]. While most of them do not directly target CVMs, NVIDIA H100 [70] is the first GPU that supports a confidential computing mode, offering trusted code execution for AMD SEV-SNP and Intel TDX. With this mode, a CVM and the GPU communicate with the SPDM protocol [73] and establish a secure connection, allowing the CVM to perform trusted code execution on the GPU while the host and other VMs cannot access the GPU memory. H100 itself has its own hardware root-of-trust and device attestation mechanism, through which a CVM owner can verify if the GPU is legitimate. Due to the lack of TDISP, there is additional overhead for communication; the CVM explicitly performs encryption and decryption and uses a bounce buffer for the DMA. Therefore, the CPU's encryption/decryption and copying speed limits the performance for I/O-heavy workload [70, 150].

3.5 Remote Attestation

A CVM owner can ensure that the CVM is launched in the intended state and runs on the intended hardware using the remote attestation process. Figure 3 (a) presents the basic attestation workflow. The CVM owner gets an attestation report from the CVM, which includes measurements of the launch and user-provided data, signed with the keys issued by CPU vendors. Next, the owner fetches the certificates and verifies the certificate chains and the report, and then checks and validates its content.

AMD SEV-SNP. Figure 3 (b) presents the overview of the attestation generation of AMD SEV-SNP. The AMD SEV-SNP VM obtains the attestation report using `SNP Guest Request`, which calls the VMM. The VMM, in turn, uses `SNP_GET_REQUEST` command to get the report from the ASP. The report is signed by VCEK (Vendor Chip Endorsement Key), which is unique for each CPU and derived from the ASP firmware and the microcode version. VCEK is signed by ASK (SEV Signing Key), and ASK is signed by AMD ARK (AMD Root Key). ARK is self-signed and consists of the root of trust of the certificate chain. AMD Key Distribution Service (KDS) provides the certifications for these keys. To verify the attestation report, the VM owner first fetches the corresponding certificates of ARK, ASK, and VCEK and verifies the certificate chains. If the fetched certificates are legitimate, then they can verify that the report is properly signed by VCEK. Finally, they can check the measurement and the user data to complete the attestation process.

Intel TDX. Figure 3 (c) shows the overview of the attestation report generation of Intel TDX. The TD obtains an attestation report from the TDX module using `TDCALL[TDG.MR.REPORT]`. The report is integrity-protected using a Message Authentication Code (MAC). To sign the report for remote attestation, Intel TDX reuses the SGX attestation mechanism [136]. It utilizes a special SGX enclave called Quoting Enclave (QE), developed and signed by Intel, and running on the same host but outside of the TD. The TD calls the QE using TDCALL or another transport mechanism, such as VSOCK. The QE first checks the integrity of the report and then signs it with the Attestation Key (AK), which is signed by Intel Provisioning Certification Key (PCK). Finally, it returns the result (quote) to the TD. The VM owner can verify the quote by consulting the certificate provided by the Intel SGX Provisioning Certification Service (PCS) and then checking and validating the measurement.

3.6 Miscellaneous Functionalities

In this section, we summarize miscellaneous features specific to CVMs.

Random number generation. The guests can use the `rdrand/rdseed` instructions to get random numbers on both SEV-SNP and TDX. The host cannot tamper with these values. However, there is a possibility of `rdrand/rdseed` instructions failing if there is enough pressure from the other cores. The host could intentionally cause this, resulting in a DoS attack to the guest, which is out of the scope of CVMs.

Time stamp counter (TSC). TDX module offers Trusted TSC [101] and handles the `rdtsc/rdtscp` instructions and returns trusted values. On the other hand, AMD SEV-SNP provides Secure TSC with the aid of ASP [44]. During boot-up, the guest queries TSC scale and offset from ASP. The guest gets a raw TSC value from a specific MSR, and then adjusts it using the scale and offset.

Performance counters. The current AMD SEV-SNP does not support secure performance counters. SEV-SNP guests can consult performance counters via VMGEXIT, but these values are host-controlled values. In addition, the host is able to observe the SEV-SNP VM state using the performance counters. Aegis [130] proposes mitigations against this issue by adding noise in the performance counters through the execution of certain instructions in the guest. AMD plans to introduce Performance Monitoring Counter Virtualization [19] to mitigate the issue. On the other hand, Intel TDX offers secure performance counter virtualization [101]. When enabled, the TDX module performs a context switch of performance counters when entering and exiting the TD. Thus, the host cannot alter the guest's performance counters or leverage them to observe the guest's state.

VM live migration. Existing live migration techniques for regular VMs are not applicable for CVMs as they rely on the hypervisor reading the guest state on the host platform. To support live migration, both ASP and TDX modules define ABIs that the guest can use to export its state and move to another machine while preserving the confidentiality and integrity properties.

Nested virtualization. The traditional nested virtualization is not available for AMD SEV-SNP and Intel TDX as it requires the VMM to be able to read the guest states and emulate the virtualization instructions [27]. While not perfectly aligned with the traditional nested virtualization model, AMD SEV-SNP provides VMPL to enforce isolation inside the CVM. Intel TDX 1.5 provides TD Partitioning, which realizes nested virtualization in the TD for up to three nested guests with one L1 VMM.

4 Performance Analysis

In this section, we present a detailed performance analysis of AMD SEV-SNP and Intel TDX compared to normal VMs across the following dimensions: boot time (§ 4.1), VM-VMM communication (§ 4.2), memory performance(§ 4.3), system and computational applications (§ 4.4), and I/O intensive applications (§ 4.5). We also show our conducted evaluation on the remote attestation primitives (§ 4.6). The goal of these experiments is to realize the differences compared to the normal VM counterpart and to understand the unique performance characteristics of the CVMs.

Table 1. Experimental environment

	AMD SEV-SNP	**Intel TDX**
CPU	4th Gen AMD EPYC 9654P 96-Cores ×1	5th Gen Intel Xeon Platinum 8570 56-Cores ×2
Memory	SK Hynix DDR5 4800 MT/s 64 GB × 12	Samsung DDR5 4800 MT/s 64 GB × 16
Storage*	KIOXIA CM7-R NVMe SSD 3 TB	Intel DC P3608 SSD 756 GB
Host OS	Linux 6.8 (AMDESE/linux cc25683†)	Linux 6.8 (canonical/tdx Ubuntu 24.04)
Guest OS	Linux 6.8 (torvalds/linux v6.8)	Linux 6.8 (torvalds/linux v6.8)
Hypervisor	QEMU 8.2 (AMDESE/qemu f246dd2)	QEMU 8.2 (tdx-noble-8.2.2)
Firmware	OVMF (AMDESE/ovmf 09fbe9)	TDVF (edk2-staging/TDVF c229fca)
Misc	SEV firmware 1.55.30	TDX module 1.5 (build_num 698)

*: For the storage performance evaluation.

†: This represents GitHub's repository name with 7-digits commit hash.

Experimental setup. Table 1 contains our experimental setup. We use 4th Gen AMD EPYC Processors for AMD SEV-SNP and 5th Gen Intel Xeon Scalable Processors for Intel TDX. At the time of writing, CVM software stacks are actively developed, and the full support of AMD SEV-SNP and Intel TDX is *not* upstreamed. Therefore, we use the patched software (host Linux, QEMU, and OVMF) provided by AMD and Intel for our evaluation. Note that the base version of the host Linux and QEMU are the same. Therefore, the main difference between AMD and Intel versions is the CPU-specific virtualization support. We use the same Linux 6.8 kernel for the guest.

We pin vCPUs to physical cores where caches are shared as much as possible, disable hyper-threading and CPU frequency scaling, and enable the default side-channel mitigations of Linux for both the host and the guest. We explicitly disable any memory encryption and CVM features for the evaluation of normal VMs. We allocate 1 GB of memory for the bounce buffer when used.

4.1 Boot Time Analysis

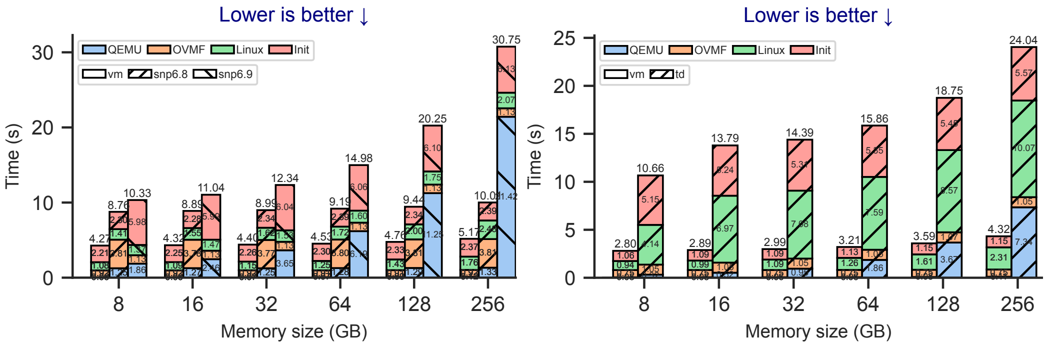
Methodology. We measure the boot time of Direct Linux Boot [62]. We use a compressed Linux kernel (bzImage) built with `defconfig` and `kvm_guest.config` with SNP or TDX support as an example of a general-purpose VM. We use an unencrypted NixOS disk image backed with virtio-blk as a root filesystem. We boot the VM without memory backend preallocation. For AMD SEV-SNP, we boot the VM using the normal direct boot, i.e, not measured direct boot, and we measure the boot time using the 6.9 kernel[2], which has a different memory initialization scheme.

We report the following time measurements: (A) VMM initialization time ("QEMU") (time between the start of `main()` in QEMU and first `ioctl(KVM_VCPU_RUN)`), (B) firmware initialization time ("OVMF") (time until `ExitBootServices()`), (C) Linux initialization time ("Linux") (time until `init_start()`), and (D) userspace initialization time ("init") (time until the all systemd services are dispatched). We perform measurements with 8 vCPUs while varying the VM memory size (8/16/32/64/128/256 GB). We perform our experiments 10 times and report the median time.

Result. Figure 4 shows the evaluation results. For 8 vCPUs and 64 GB memory, the boot time of the AMD SEV-SNP VM is 102% (kernel 6.8) and 231% (kernel 6.9) longer than the normal VM, whereas the TD is 394% longer. Notably, the SNP kernel 6.9 and TDX show an increase in boot time (especially for "QEMU") as the memory size increases. This happens because the current KVM separates the entire guest memory from the normal host memory before launching [41][3] and *not*

[2] Version: AMDESE/linux 05b1014, AMDESE/qemu fb924a5, tianocore/edk2 4b6ee06

[3] The KVM allocates memory as guest-first memory, which makes memory inaccessible from the host even in the absence of hardware encryption support. This enables additional isolation but comes with a certain management cost.



(a) AMD SEV-SNP

(b) Intel TDX

Fig. 4. Linux Direct Boot time. Changing memory size with 8 vCPUs.

because of the measurement of more memory pages as normal memory pages are later accepted during runtime.

Takeaway #1: The CVM boot time increases compared to the normal VM due to the additional initialization process. For 8 vCPUs and 64 GB memory, we observe up to 231% and 394% boot time increases for AMD SEV-SNP and Intel TDX, respectively. Notably, the host-side memory management of the guest memory affects the boot time of the CVMs.

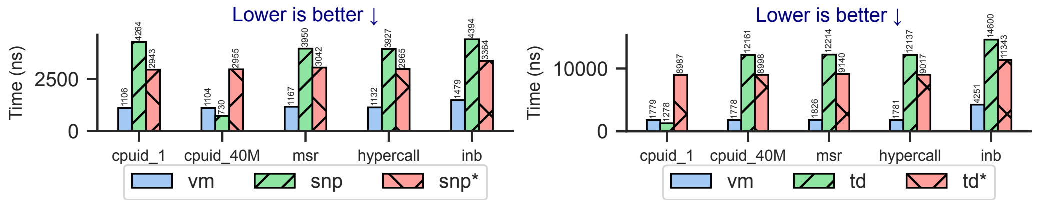
4.2 VM-VMM Communication Latency

Methodology. We design and execute a kernel module in the guest that triggers instructions typically resulting in a VMEXIT. Specifically, we measure the latency of CPUID (leaf 0x1 (cpuid_1) and 0x40000000 (cpuid_40M)), MSR read (`IA32_APIC_BASE` MSR (0x1b)), hypercall (`KVM_HC_MMU_OP` (deprecated), the KVM returns immediately), and PIO (read from I/O port 0x40, a status register of the PIT (Programmable Interval Timer)). We invoke instructions a hundred thousand times and calculate the average latency. We also measure the latency of CVM guests directly calling the host using VMGEXIT (AMD SEV-SNP) or TDCALL (Intel TDX), which eliminates the exception handling cost.

Result. Figure 5 shows the result of the VMEXIT latency evaluation. `snp*` and `td*` show the results when the guest explicitly calls the host. For AMD SEV-SNP, the latency of executing instructions that entail VMEXIT has an average of 240% increase compared to the normal VM, whereas the TD shows 472% increase, showing that Intel TDX has much higher VMEXIT latency than AMD-SNP. The intervention of the TDX module during the context switch is the most probable reason for this behavior. The SEV-SNP VM handles the cpuid leaf 0x40000000 in the #VC handler and does not call the VMM[4], and the TD handles the cpuid leaf 0x1 in the TDX module and does not call the VMM. Therefore, there is no VMEXIT in the execution of these instructions.

Takeaway #2: VMEXIT is a costly operation for CVMs. Instructions that involve VMEXIT to the VMM introduce 240% latency increase for SNP VMs on average, whereas the TD shows 472% increase. An SNP VM avoids the VMEXIT overheads for some CPUID instructions by using a special CPUID page, and a TD avoids that if the TDX module handles the instructions.

[4] During the launch, the VMM can add a special CPUID page to the guest, which ASP validates and ensures the trusted CPUID values. An SEV-SNP VM consults CPUID first in the #VC handler, and only calls the host if some fix is needed.



(a) AMD SEV-SNP

(b) Intel TDX

Fig. 5. Instruction latency. Instructions except cpuid_40M for SNP and cpuid_1 for TD results in VMEXIT through an explicit call to the host via #VE/#VC. SNP guest consults a CPUID page added during the launch for getting value cpuid_40M, and thus there is no VMEXIT. TDX module handles cpuid_1 and returns the value without injecting an exception. snp* and td* calls to the host directly, removing #VE/#VC overheads.

4.3 Memory Performance Analysis

4.3.1 *Memory Latency and Bandwidth.*

Methodology. We use the Intel Memory Latency Checker (MLC) [110] to measure memory latency and bandwidth. In addition, we use the entire Memory Test Suite of the Phoronix Test Suite [138, 139] except t-test1 (memory allocation benchmark) to evaluate memory performance. The suite includes MBW [58], Tinymembench [66], RAMSpeed/SMP [63], STREAM [135], and Cachebench [137].

Result. Intel MLC reports a 16ns increase for the random idle read latency on both AMD SEV-SNP and Intel TDX. For the peak injection bandwidth, AMD SEV-SNP VM reports 3.90, 3.98, 9.08, 0.416, 1.96% overhead for All reads, 3:1 RW, 2:1 RW, 1:1 RW, and stream-like bandwidth (average 2.88%), whereas a TD reports 10.04, 8.25, 7.20, 7.02, 7.03% overhead respectively (average 7.86%). Figure 6 shows the result of the Phoronix Test Suite memory benchmark. The SNP VM shows 7.29% overhead on average. The respective value for TDs is 4.06%.

4.3.2 *Memory Mapping Performance.*

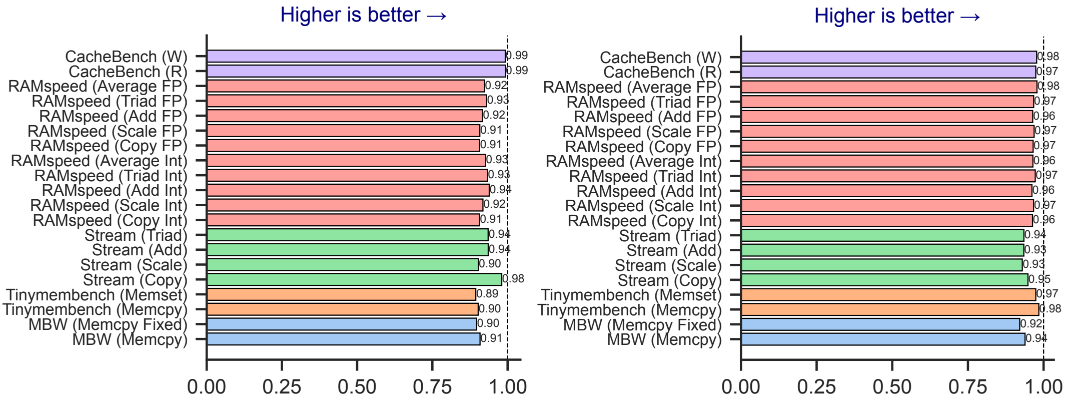
Methodology. We create a program that allocates a 32GB memory region using `mmap()` with the (`MAP_POPULATE`) flag. We measure the time of the first and second invocation of the program using a VM with 64 GB memory and 8 vCPUs. For CVMs, the initial memory allocation involves memory acceptance. We repeat the experiment 10 times and report the median values.

Result. The SNP VM takes 28.11 seconds to populate the 32GB memory region for the first time, which is 59% increase compared to the normal VM (17.64 s). The TD takes 29.05 seconds for its first mmap, which is 92% increase compared to the normal VM (15.15 s). In the second mmap invocation, the memory is already accepted. Thus, the memory population time on the CVMs is comparable to that on the normal VMs, with less than a 5% difference (15.38s and 12.97s for the SNP VM and the TD, respectively).

Takeaway #3: The CVMs introduce memory overhead in terms of latency and bandwidth due to continuous encryption and integrity protection. Phoronix Memory Test Suite shows 7.29% and 4.06% overhead for AMD SEV-SNP and Intel TDX, respectively. Further, memory acceptance takes time for the first use of a memory region. Once the memory is accepted, this overhead is eliminated.

4.4 System and Computational Application Performance Analysis

Methodology. We run the following applications to analyze application performance in CVMs: (1) Unixbench, a set of programs to evaluate various aspects of Unix-like systems, (2) NAS Parallel Benchmarks (NPB) [72], a set of programs to evaluate parallel computation performance, (3) a 3D image rendering with Blender [50], (4) image classification with PyTorch [61, 160], and (5) a large language model (BERT [69]) with TensorFlow [1, 67]. We choose these applications to reflect



(a) AMD SEV-SNP

(b) Intel TDX

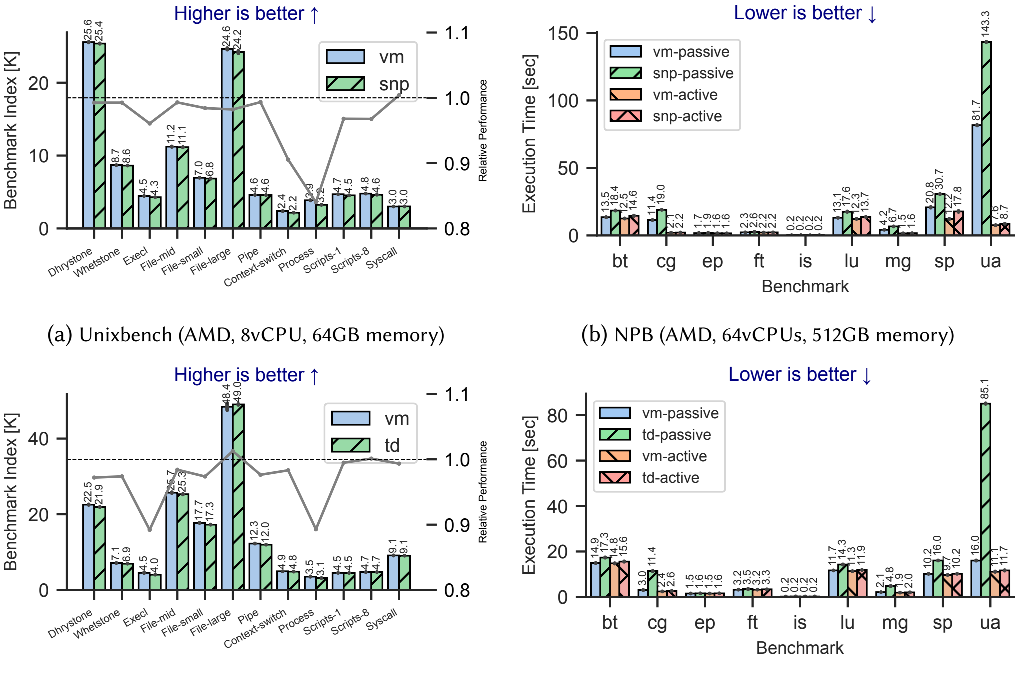
Fig. 6. Normalized overhead compared to a normal VM of Phoronix Memory Test Suite.

the diverse workloads of the cloud. Note that cloud providers envision that the entirety of cloud computing becomes confidential [168], highlighting the importance of evaluating the performance of CVMs with various workloads. We do not use any accelerators (e.g., GPUs) for computation; we run the applications directly on the CPU.

We run UnixBench 5.1.3 with 8 parallel copies with 8 vCPUs and 64 GB memory. We run NPB with 64 vCPUs and 512 GB memory for the AMD machine and 56 vCPUs and 256 GB memory for the Intel machine to evaluate the performance of a large VM. We use the NPB 3.4.1 (OpenMP version) and choose the size C. We run the benchmark with two settings: (1) with `OMP_WAIT_POLICY=PASSIVE` (threads are put to sleep when waiting) and (2) with `OMP_WAIT_POLICY=ACTIVE` (threads actively poll) [29]. We run each benchmark five times and present the average values. For Blender, PyTorch, and TensorFlow, we use four VM configurations: “S” (1 vCPUs, 8 GB memory), “M” (8 vCPUs, 64 GB memory), “L” (32 vCPUs, 256 GB memory for the AMD machine, 28 vCPUs, 128 GB memory for the Intel machine), and “X” (64 vCPUs, 512 GB memory for the AMD machine, 56 vCPUs, 256 GB memory for the Intel machine). For Blender, We measure the rendering time of a sample scene with a size of 175 KB, where the blender renders a 640x480 PNG image. For PyTorch, we measure the inference time of image classification using AlexNet [120] 1000 times. For TensorFlow (BERT), we measure the throughput of the inference using the BERT-Large model with squad dataset [167]. We configure the applications to have as many threads as the number of vCPUs in the VM.

Result. Figure 7 and Figure 8 show the application performance evaluation results. We observe that several benchmarks incur more than 10% overhead compared to the normal VM both on AMD SEV-SNP and Intel TDX, which is larger than the previously reported memory overhead (§ 4.3). These are (1) “Process” of UnixBench (16% and 11% overhead for the SEV-SNP VM and TD), (2) NPB with `OMP_WAIT_POLICY=PASSIVE`, (41% and 73% increase in execution time on average), and TensorFlow (BERT) results with a large VM (“X”) (16% and 18% overhead for SEV-SNP VM and TD).

Performance analysis. We analyze the performance drop of the TensorFlow (BERT) in the “X” on a TD. First, this workload does not fully use the CPUs even if the application has the same number of threads as the vCPUs assigned to the VM. The CPU utilization of the normal VM is ~88%, whereas the one of the TD is ~75%. Next, we monitor TDVMCALLs (`TDCALL[TDG.VP.VMCALL]`) during the workload and find that there are 99,450 TDVMCALLs per second on average (11,312 for HLT, 12,340 for RDMSR, and 75,797 for WRMSR for setting APIC timer). This indicates that the workload has many sleep and wake-up events, due to the synchronization events for the



(a) Unixbench (AMD, 8vCPU, 64GB memory)

(b) NPB (AMD, 64vCPUs, 512GB memory)

(c) Unixbench (Intel, 8vCPU, 64GB memory)

(d) NPB (Intel, 56vCPU, 256GB memory)

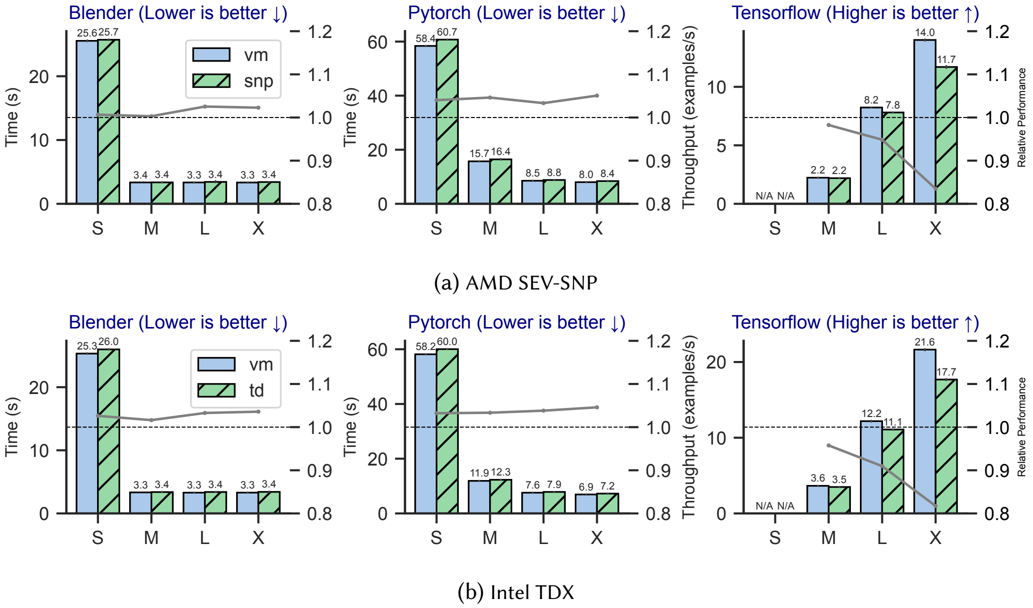
Fig. 7. Unixbench and NPB ("-passive" and "-active" are results with `OMP_WAIT_POLICY=PASSIVE/ACTIVE`.)

computation. We also observe similar VMEXITs on the AMD machine. While these VMEXITs occur in the normal VMs as well, one of the possible causes for the performance drop is the costly VMEXITs in the CVMs, as shown in § 4.2.

Mitigations. We find several mitigation techniques for this issue. The first one is using idle polling [195], which makes the vCPU busy wait when it is idle. This eliminates the VMEXITs due to the halt. With this option, the TensorFlow (BERT) overhead of TD compared to the normal VM (with idle poll) is 4.3% in the "X" VM on the Intel machine, and 2.3% in the "X" VM on the AMD machine. We also confirm that idle polling reduces the performance overhead for the "Process" of UnixBench. However, this is costly as the guest CPU utilization is always 100% from the host's point of view.

Linux also has *guest halt polling* [56], which makes the guest poll for a certain amount of time before executing HLT. Table 2 shows the relation between performance and CPU utilization of TensorFlow (BERT) with different halt polling values on the Intel machine. There is a trade-off between performance and an increase in CPU utilization due to the polling. Increasing the time of halt polling increases the throughput, but the CPU utilization also gets higher.

Application-side polling is yet another option. For the NPB benchmark, the result with the `OMP_WAIT_POLICY=ACTIVE` option shows a 12.2% and 5.9% average increase in execution time compared to the normal VM on the AMD and Intel machine, respectively.



(a) AMD SEV-SNP

(b) Intel TDX

Fig. 8. Application benchmarks (Blender, PyTorch (image classification), TensorFlow (BERT)).

Table 2. The relation between the throughput of TensorFlow (BERT) and the polling strategy in the "X" VM on the Intel machine. We report the guest userspace CPU utilization and host-side vCPU utilization (50% is the maximum as we allocate half of the pCPUs as vCPUs). haltpoll 200k means we set 200k for the `halt_poll_ns` parameter. We use the default values for other haltpoll parameters (`guest_halt_poll_grow=2`, `guest_halt_poll_shrink=2`, `guest_halt_poll_grow_start=50000`, and `guest_halt_poll_allow_shrink=Y`). "idle poll" means the guest uses idle polling.

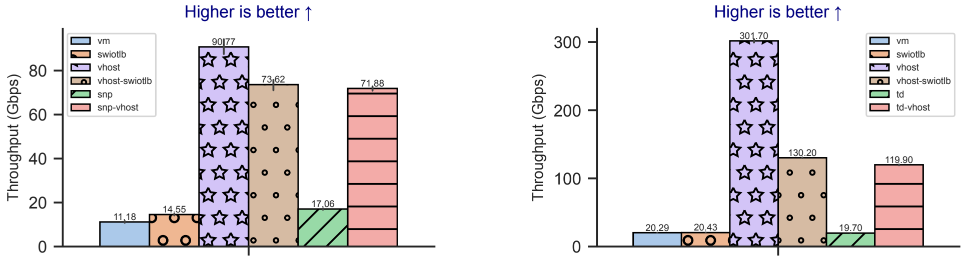
	Guest CPU util	**Host CPU util**	**Throughput (/s)**
VM	88.47	44.35	21.67
TD	74.35	37.93	17.51
TD (haltpoll 200k)	77.15	39.52	18.15
TD (haltpoll 1000k)	84.81	46.90	20.06
TD (haltpoll 10000k)	89.12	49.88	21.02
TD (idle poll)	89.64	50.00	20.63

Takeaway #4: Applications on CVMs show performance degradation mainly due to memory overheads. Additionally, if application threads (vCPUs) have many sleep and wake-up events, the vCPU-sleep (HLT) overhead of the CVM can be another significant factor for the performance drop. We observe up to 431% increase in execution time (NPB benchmark "ua" on a TD). The guest-side halt polling mitigates this overhead with the cost of higher CPU utilization.

4.5 I/O Performance Analysis

4.5.1 Network Performance Analysis.

Methodology. We use a virtio-net [157] device with a local network. We use the multi-queue (8 queues) for TCP evaluation to stress the network. We use iperf3 [114] to measure the TCP/UDP



(a) AMD SEV-SNP. The CPU utilization: around 40% (“vhost”), 100% (“vhost-swiotlb” , and “snp-vhost”).

(b) Intel TDX. The CPU utilization: 100% for the “vhost”, “vhost-swiotlb”, and “td-vhost”.

Fig. 9. TCP Throughput. Using a local network, with multi queues (8 queues), 128K buffer size.

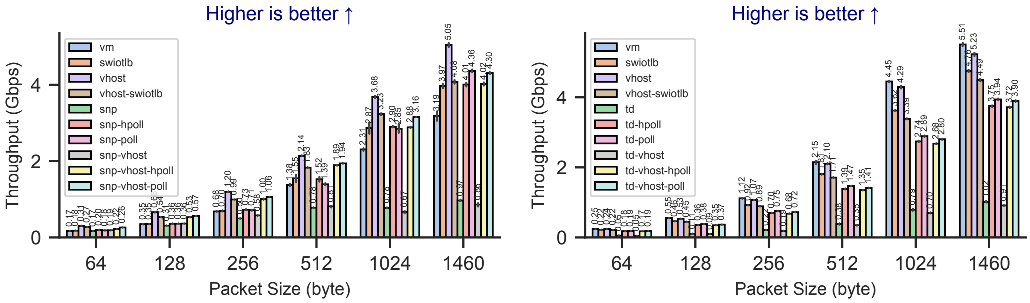
throughput. We run the iperf3 server in the guest and the client in the host. We pin the iperf3 client threads to different physical cores where vCPUs are pinned. We have 8 parallel connections of iperf3 except for the TCP measurement of the Intel machine, where we have 32 parallel connections to stress the server. We also measure the round-trip time (RTT) using ping. We measure the RTT 30 times and use the first three RTTs as warm-up and do not consider them in our reported results. We measure the performance of a normal VM (“vm”), a normal VM using bounce buffer (“swiotlb”) (§ 3.4), a normal VM using vhost (“vhost”), a normal VM with both vhost and swiotlb (“vhost-swiotlb“). For CVMs, we measure the performance with several options: vhost enabled (“-vhost”), guest halt polling enabled (“-hpoll”), and idle polling enabled (“-poll”). Lastly, we also report network I/O benchmarks with Nginx and Memcached.

Case #1: Performance under high CPU load (TCP). Figure 9 shows the network evaluation results using TCP with 128K buffer size. We observe that the bounce buffer overhead is considerable. On the AMD machine, the guest CPU utilization is around 40% for the “vhost” and achieves a throughput of 90 Gbps[5]. On the other hand, “vhost-swiotlb” and “snp-vhost” show 19% and 21% overhead compared to “vhost” with 100% CPU utilization. On the Intel machine, with the vhost enabled, the guest CPU is almost fully utilized, achieving 302 Gbps throughput. Under this condition, “vhost-swiotlb” shows 57% overhead compared to “vhost”, whereas “td-vhost” shows 60% overhead. These results demonstrate that under high CPU utilization, bounce-buffer overhead becomes significant, reducing the throughput and increasing the CPU utilization more if it is not saturated.

Case #2: Performance under non-CPU intensive case (UDP). Figure 10 shows the network performance evaluation using UDP with different packet sizes. In this case, CPU utilization is low. For instance, on the Intel machine, the normal VM reaches 5.51 Gbps with packet size 1460, while 37.49% CPU is idle. We observe a similar vCPU-sleep performance issue that occurred in the computational application evaluation. TD only achieves 1.02 Gbps with 65% CPU idle for the packet size 1460 (81% overhead). With we enable halt polling or idle polling, the TD achieves 3.74 Gbps or 3.94 Gbps (32% and 28% overhead, respectively). On the AMD machine, the normal VM achieves 3.19 Gbps, whereas SNP reaches only 0.97 Gbps (75% overhead). However, the halt polling and idle polling bring the performance to 4.01 Gbps and 4.36 Gbps, respectively (-2% and -11% overhead).

Case #3: Latency (ping). Figure 11 presents the network latency results using ping. An SNP VM and a TD exhibit 17 and 16 μs higher latency than a normal VM (w/o vhost). The idle polling

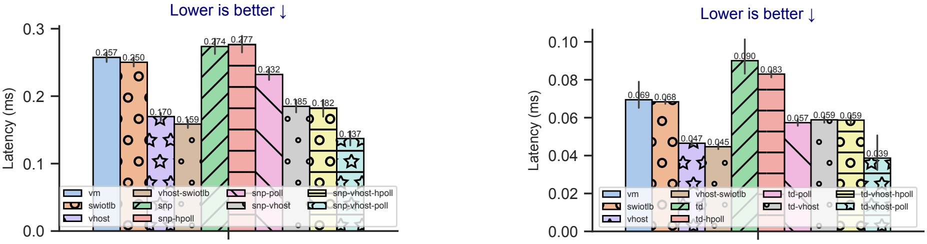
[5] We tried but could not utilize full CPUs on this evaluation.



(a) AMD SEV-SNP

(b) Intel TDX

Fig. 10. UDP Throughput. Using a local network, with a single queue, different packet sizes.



(a) AMD SEV-SNP

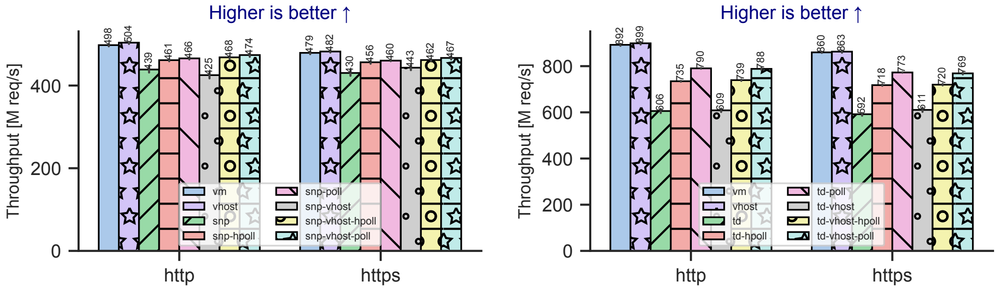
(b) Intel TDX

Fig. 11. Network Latency (Ping). Using a local network, with a single queue, 64-byte packet size.

reduces the latency by 42 µs (w/o vhost) and 48 µs (w/ vhost) on the SNP VM and 29 µs (w/o vhost) and 24 µs (w/ vhost) on the TD.

Case #4: Nginx and Memcached. We perform additional network I/O benchmarks with Nginx [59] and Memcached [57]. We use a VM with 8 vCPUs and 64 GB memory. For networking, we use the local network with virtio-net with a single queue. For Nginx, we use wrk [194] (with 8 threads, 300 connections) to stress the Nginx server and present the throughput measured in requests per second (RPS). The web server serves as a static web page that contains a "Benchmark" message. We evaluate the performance of both HTTP and HTTPS cases. For Memcached, we use memtier-benchmark [64] to measure the throughput of the Memcached server. The Memcached server is configured with 8 threads. The client uses 8 threads and 100 connections with pipeline size 40 to stress the Memcached server. We measure the performance of both TCP and TLS cases.

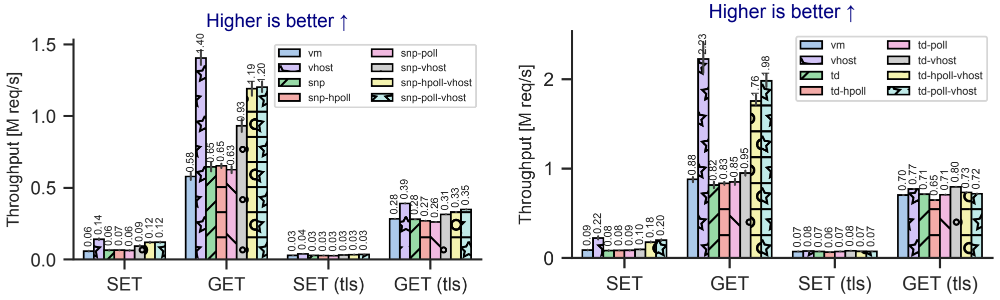
Figure 12 and Figure 13 show the respective results. We find similar characteristics to the iperf UDP evaluation in the Nginx results, though the performance drop is smaller than the iperf UDP evaluation, and idle polling mitigates the performance drop. For example, the SEV VM shows a 10% overhead compared to the normal VM for the HTTPS case (without vhost), whereas the SEV VM with idle polling shows a 4% overhead. For Intel TDX, the overheads for these cases are 31% and 10%, respectively. Regarding Memcached, for the TCP case, the SEV VM has 34% overhead (with vhost), whereas the SEV VM with the idle polling has 14% overhead for a GET operation. On the other hand, the respective overheads of the TD and the TD with the idle polling are 57% and 11%. However, for the TLS case, we find smaller performance differences. The SEV VM has a 20% overhead, and the SEV VM with the idle polling has a 10% overhead for a GET operation.



(a) AMD SEV-SNP

(b) Intel TDX

Fig. 12. Nginx performance with wrk. Using a local network, with a single queue.



(a) AMD SEV-SNP

(b) Intel TDX

Fig. 13. Memcached performance with memtier-benchmark. Using a local network, with a single queue.

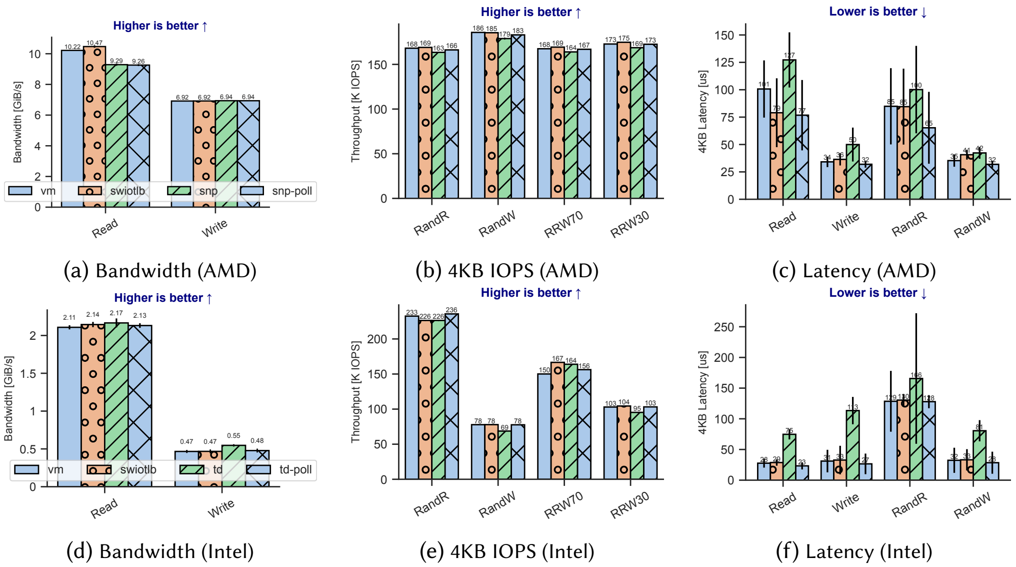
The overhead of the TD and the TD with the idle polling for the same operations are -3% and 6%, respectively. This could be attributed to the TLS processing overhead.

4.5.2 Storage Performance Analysis.

Methodology. We use a virtio-blk [157] backed by an entire NVMe device. QEMU creates an iothread and uses libaio with O_DIRECT to access the NVMe device. We use fio [52] to measure the bandwidth (128KB block size, 128 I/O depth, and 4 jobs for sequential read and write), IOPS (4 KB block size, 32 I/O depth, and 4 jobs for random read, random write, random read/write with 70% read and 30% write), and latency (4 KB block size, 1 I/O depth, and 1 job for read, write, random read, and random write) of the storage performance. For each test case, we use 20 seconds for the warm-up and 30 seconds for the measurement. We perform 10 runs and report the median result. Fio uses libaio ioengine in the guest and performs I/O operations with O_DIRECT to the entire virtio-blk device.

Result. Figure 14 shows the storage performance evaluation results. Notably, in our environment, fio does not reach the maximum performance of the device[6], and the operation is not CPU-bound. Therefore, both bandwidth and IOPS evaluations do not show a large difference between a normal VM and a CVM. An SEV-SNP VM has 4.6% and 2.9% overhead, and a TD has -9.7% and 2.5% overhead on average for bandwidth and IOPS compared to the normal VM. On the other hand, CVMs show

[6] Intel DC P3608 (intel machine): maximum bandwidth: 5 GB / 2 GB (Read / Write), IOPS: 850k / 150k (random 4k read/write). KIOXIA CM7-R (AMD machine): maximum bandwidth: 14 GB / 6.75 GB, IOPS: 2,700K / 310K (id.).



(a) Bandwidth (AMD) (b) 4KB IOPS (AMD) (c) Latency (AMD)

(d) Bandwidth (Intel) (e) 4KB IOPS (Intel) (f) Latency (Intel)

Fig. 14. Performance of a virtio-blk backed by a raw NVMe disk. "-poll" is a result with the idle polling.

higher latency than normal VMs. An SEV-SNP VM has 16 µs, and a TD has 54 µs average latency increase. If we enable idle polling, the CVMs have a similar performance as the normal VM.

Takeaway #5: If the guest CPU utilization is high for the I/O processing, the bounce buffer (memory copying) overhead dominates the performance drop of the CVMs. We observe 60% performance drop on iperf TCP measurement where CPU utilization is 100%. On the other hand, under low CPU utilization cases, such as iperf UDP and latency measurement, the VMEXIT overhead can become more dominant. The guest-side halt polling largely mitigates the performance drop in such cases as well.

4.6 Analysis of Attestation Primitives

Methodology. We evaluate the attestation process of AMD SEV-SNP and Intel TDX. Precisely, we measure the time of each attestation primitive shown in Table 3. For AMD SEV-SNP, we use snpguest [182] and measure the time of attestation primitives on the guest. For TDX evaluation, we use tdx-attest [109], which uses VSOCK to call the QE to measure the time of getting a signed quote. To measure the time of the certificate and quote verification, we use go-tdx-guest [80]. We use Ubuntu 24.04 as a guest and perform the certificate and the quote verification on the host. We repeat each measurement 10 times and report the average time.

Result. Table 3 shows the obtained results. Getting an attestation report takes the most time and is 6.19 ms for AMD SEV-SNP and 2.75 ms for Intel TDX on average.

Takeaway #6: The operation of attestation primitives only takes several milliseconds. In practice, the remote attestation is combined with other use cases, such as encrypted disk boot [30] and establishing a secure TLS connection [119], and operations like booting and communicating over network take more time than the attestation itself.

4.7 Discussion

Comparison of AMD SEV-SNP and Intel TDX. In general, our evaluation shows AMD SEV-SNP and Intel TDX have similar performance characteristics with a few exceptions.

Table 3. Attestation Primitives and their processing times

Primitive	AMD SEV-SNP [ms]	Intel TDX [ms]
Getting an attestation report (quote for TDX)	6.19	2.75 (getting a report: 0.022)
Verifying the certificates	1.04	0.67
Verifying the report (quote for TDX)	2.68	0.24

First, the boot time measurement shows that the TDX takes more time in both the VMM (QEMU) and the guest firmware on the Linux 6.8 kernel. On the other hand, the SEV-SNP results demonstrate that different software stacks can show quite different boot times. We presume that there are several possible optimizations for reducing the boot time.

Secondly, our VMEXIT latency measurement shows that Intel TDX has a longer average VMEXIT latency (472% increase compared to the normal VM) than AMD SEV-SNP (240%). This could be attributed to the intervention of the TDX module. However, more investigation is needed to identify the root cause. On the other hand, our application benchmarks show that the TDX (and SEV-SNP) performance degradation for normal applications is not as bad as the VMEXIT benchmark. This is because VMEXITs do not dominate the normal workload. In addition, if the TDX module handles VMEXIT (e.g., some CPUID leaves), then the VMEXIT latency can be smaller than that of the normal VM.

In some benchmarks, particularly NPB and iperf TCP, the performance overhead of TDX is higher compared to the normal VM, as opposed to SEV-SNP. However, we must note that in these evaluations, a normal VM on an Intel machine has much better performance than a normal VM on an AMD machine, which contributes to the larger performance overhead of TDX.

Performance characteristics. We find three main factors that affect application performance on CVMs. Firstly, for standard computational workloads (Unixbench, Blender, image classification with PyTorch), memory encryption and integrity protection are the main sources of the overhead. Secondly, for I/O intensive workloads under heavy CPU utilization (as demonstrated in iperf TCP), the bounce buffer overhead justifies the performance drop. Lastly, if the application has many sleep events due to the required synchronization for multithreaded applications (e.g., Bert with TensorFlow and NPB) or I/O wait for the application (e.g., iperf udp), then the HLT VMEXIT overhead cannot be ignored. We detail the possible performance optimization below.

Performance optimization. Our evaluation reveals several opportunities for promising performance optimizations.

First, our boot time evaluation demonstrates that the software stack largely affects the boot time. SEVeriFast [89] also analyzes the boot time of AMD SEV-SNP and proposes the optimized boot process for micro-VMs on AMD SEV-SNP. TD-shim [65] is a thin firmware specialized for TDs. Incorporating such optimized firmware could reduce the boot time for standard CVMs as well.

Secondly, our application evaluation shows that vCPU HLT VMEXIT can cause a significant overhead for the CVMs. The investigation of the root cause is left as future work. HotCalls [187] proposes shared-memory-based communication between an SGX enclave and the host OS to reduce the overhead of the context switch. Similar techniques could be applied to the CVMs to reduce the overhead of the VMEXIT. As a mitigation, we demonstrate that the guest-side halt polling can be effective. However, the guest-side halt polling has a trade-off with the CPU utilization, and finding the optimal configuration depends on the workload. Automated tuning of the polling policy would also be a path for future work.

Regarding the bounce buffer overhead, Bifrost [123] proposes optimized bounce buffer handling for the encrypted network connection by directly encrypting and decrypting data in the bounce buffer, removing the overhead of an additional data copy. We could use similar techniques to reduce the bounce buffer overhead for other devices, such as storage.

Lastly, exploring architectural improvements is also an important direction. For example, HPMP [75] proposes an efficient additional page-walk mechanism for memory protection on the RISC-V architecture.

Comparison with previous work. Akram et al. [5] conduct a performance evaluation on AMD SEV (not SNP) using NPB, reporting up to 3.4x slowdown on a NUMA machine with memory interleaving. However, they do not observe a performance drop regarding HLT. This could be attributed to the use of AMD SEV, which lacks state protection and, therefore, has lower VMEXIT latency. Also, no information on the wait policy is provided. Bifrost [123] focuses on network I/O on AMD SEV-ES, reporting a memory copy of the bounce buffer consumes more than 50% of the CPU cycles for CPU heavy workload, in line with our TCP evaluation. They also report frequent VM exits due to interrupts taking up more than 20%. However, they do not evaluate any impact of the halt polling.

Yan and Gopalan [196] evaluate AMD SEV-SNP performance with various benchmarks. They report up to 4.3% overhead for memory-intensive workloads and 36% TCP overhead using the local network, which roughly matches our evaluation. On the other hand, they report only a 3-4% performance drop on UDP measurements and around 47% overhead in disk I/O using virito-scsi, which we could not reproduce. The difference could be attributed to the use of different software stacks, devices, and configurations. They do not report I/O latency and CPU utilization metrics. In addition, they report measurements for NPB, showing ~3.5% overhead on ua (size D) on an SNP VM. It is possible that they use OpenMP's default wait policy (adoptive poll policy), although this information is not provided. Qiu et al. [165] show half of the IOPS performance with 24 local SSD disks on an SNP VM, indicating storage also can show a significant overhead when heavily used.

Early-stage software and hardware. We experience several issues during our evaluation, including unknown firmware errors and performance degradation due to configuration changes. Our boot time evaluation demonstrates that the different software versions can show significantly different performance characteristics. Additionally, software support for VM live migration, TD-partitioning, and VMPL is still limited and only partially available. The ecosystem of CVMs is still in the early stages, highlighting the importance of continuous evaluation and improvement.

5 Security Analysis

This section evaluates the security of AMD SEV-SNP and Intel TDX. We mainly focus on the following aspects: TCB size and disclosed vulnerabilities.

5.1 TCB Analysis

Methodology. We measure the software TCB by counting the binary size and the LoC of the guest Linux kernel, the guest firmware, and the platform firmware (ASP firmware and the TDX module) using cloc [45]. We use their default configuration to build the binary. We report the size of uncompressed `vmlinux` as the binary size of the Linux kernel. We only count the files actually used in the build process by monitoring the file accesses. For the platform firmware, we report the size and LoC of the platform firmware that Intel and AMD provide on their website as a reference due to the limited availability of the source and build environment.

Result. Table 4 shows the TCB sizes of AMD SEV-SNP and Intel TDX. Linux kernel is the largest component in the TCB, followed by the guest firmware and the platform firmware.

5.1.1 Reducing and Hardening TCB. To improve the security of CVMs, minimizing and hardening the TCB is crucial. There are several approaches to reduce and harden the TCB of CVMs.

Intra-CVM isolation. One approach to reduce the TCB is having intra-CVM isolation. AMD SEV-SNP provides VMPL, and Intel TDX 1.5 provides TD-partitioning, allowing the guest OS to be partitioned into multiple isolated components. However, software support for these features is premature, and further research is needed to utilize these features effectively.

There are several works on utilizing VMPL. One notable existing work is COCONUT-SVSM [51], which is the secure monitor implementation for AMD SEV-SNP in accordance with the SVSM [16]. Hecate [76] and Azure paravisor [164] use VMPL to run an unmodified guest kernel in AMD SEV-SNP VMs. While the main focus of these works is achieving backward compatibility, they can be used as a base to implement a secure monitoring mechanism. VEIL [3] also uses VMPL to have secure enclaves within the AMD SEV-SNP VMs, while Cabin [140] protects the guest OS from the untrusted guest application using VMPL.

Reducing codebase. Another approach to reduce the TCB is to reduce the codebase of the guest OS. Several works aim to reduce the TCB of the CVMs by designing a dedicated software for CVMs [49, 65, 74, 121]. Intel develops a specialized firmware mainly for the container environment called TD-shim [65]. Some works propose a minimal OS based on library OS or designing a new OS for CVMs. For instance, Gramine-TDX [121] is using the Gramine LibOS [34] ported to TDX. Specializing in running a specific application, Gramine-TDX successfully reduces the TCB and minimizes the attack surface. Another example is Asterinas [49], an operating system designed for CVM environments aiming for a minimal TCB with Linux ABI compatibility in Rust. Mushroom [74] is another dedicated kernel that aims to run unmodified Linux applications with integrity protection utilizing the VMPL.

Hardening TCB. Hardening TCB so that the CVMs can resist attacks from the outside is also crucial. Table 5 summarizes the main inputs from the VMM to the CVM. We can separate inputs into four categories: (1) boot inputs, which the VMM passes when launching the CVM, (2) device I/O performed by the guest, (3) external inputs, and (4) certain CPU instructions. CVMs, ideally, must validate these inputs before using them. Regarding boot inputs, calculating the measurement of not only static inputs (the initial guest code, data, and state) but also dynamically calculating loaded components is crucial to ensure the guest state. For other inputs, the CVM guest kernel should carefully examine if the input values are legitimate by checking input ranges or using whitelisting. As we show in § 5.2, many real-world attacks are related to insufficient input validation.

There is a constant effort to harden the interfaces with the untrusted world in the kernel [170, 180]. However, the complexity of the Linux kernel and the abundance of its features mandate a lot of manual, non-trivial analysis and the development of dedicated tools to provide a good level of confidence in its hardening. This makes it quite difficult for the hardening effort to be accepted upstream. To this end, Intel releases TDX hardening documentation [111], which summarizes the design guideline of how the guest OS validates the inputs from the VMM. Additionally, several works perform fuzz testing to test the input validation of the CVMs [88, 111].

Code provenance. Last but not least, a mechanism that records the origin and history of code is important for usability. For instance, Code Transparency Service [46] proposes an architecture to track code provenance and to hold code providers accountable, preventing supply chain attacks.

> ***Takeaway #7***: Due to the nature of the CVMs, the TCB size can be more than millions of LoC as it includes the full-fledged OS of the guest VM. There are several research works that aim to reduce the TCB size and harden the TCB to improve the security of CVMs.

5.2 Disclosed Vulnerabilities

Methodology. We search the MITRE CVE database [42] with the keywords “SEV”, “SNP”, and “TDX”, and extract the CVEs related to AMD SEV(-ES/SNP) and Intel TDX. We summarize the

Table 4. Software TCB Sizes. "-" indicates that the respective information is not publicly available.

CVM	Component	Binary (B)	KLoC
AMD SEV-SNP	Linux 6.8 (w/ defconfig, kvm_guest.config, SNP support)	13M	2,307
	OVMF (w/ SNP support)	4.0M	206
	ASP SEV firmware	93K*	29†
Intel TDX	Linux 6.8 (w/ defconfig, kvm_guest_config, TDX support)	13M	2,304
	TDVF	4.0M	764
	TDX module 1.5	-	45
	Quoting Enclave, SEAM Loader‡	-	-

*: SEV firmware 1.55.30 [hex 1.37.1E], †: SEV firmware 1.55.25 [hex 1.37.19]. ‡: Component to load the TDX module.

Table 5. External CVM inputs from the VMM.

Input Source	Input Types	Notes
Boot inputs	Firmware code	Measured statically during initialization.
	UEFI variable, ACPI tables	Possibly measured dynamically using RTMR (TDX) or vTPM.
	PCI option ROM, OS loader	Possibly measured dynamically using RTMR (TDX) or vTPM.
	Kernel code, cmdline parameters	Possibly measured dynamically using RTMR (TDX) or vTPM. Possibly measured statically using Direct Measured Boot (SNP).
Device I/O	PIO	Use VMGEXIT / TDCALL.
	MMIO	Use VMGEXIT / TDCALL.
	DMA	Via unencrypted shared buffer.
External Inputs	Interrupts and exceptions	The VMM can inject. TDX prohibits exception injection #0-#31.
CPU instructions	Hypercalls	Use VMGEXIT, TDCALL.
	MSR (RDMSR, WDMSR)	TDX module virtualizes some of MSRs.
	CPUID	TDX module virtualizes most of CPUIDs. ASP provides trusted CPUID pages for SNP guests.
	RDPMC	TDX supports trusted performance counter virtualization.

CVEs up to June 2024. Regarding AMD SEV(-ES/SNP), we divide the CVEs into those applicable to AMD SEV-SNP and those that are not. We consider CVEs as applicable to AMD SEV-SNP if the description explicitly mentions SNP. If the CVEs are applicable to both AMD SEV-SNP and SEV(-ES), we classify them as AMD SEV-SNP. If CVE's explanations are unclear, we classify them as AMD SEV(-ES).

We categorize the CVEs into four types depending on the attacker's position and target: host-to-guest, host-to-host, guest-to-guest, and guest-to-host. We further categorize them into (1) software bugs, including validation bugs and functional bugs, (2) hardware bugs, including microarchitectural side-channel issues, and (3) design issues in the CVMs mechanism.

Result. Table 6 shows the summary of CVEs related to AMD SEV(-ES/SNP) and Intel TDX. In total, there are 49 CVEs related to AMD SEV(-ES/SNP) and 9 CVEs to Intel TDX. Among them, 81% of the CVEs are host-to-guest attacks, 5% are guest-to-guest attacks, 5% are guest-to-host attacks, and 9% are host-to-host attacks. 22 CVEs (39%) are attributed to improper validation mechanisms, while 31 (54%) are associated with vulnerabilities in the underlying firmware. Notably, AMD SEV(-ES/SNP) predates Intel TDX and, therefore, has more reported CVEs. Further, AMD SEV-SNP fixes many design issues in AMD SEV and SEV-ES [11]. The detailed list of CVEs is provided in the Appendix D. We describe the representative CVEs in the following sections.

5.2.1 Host-to-guest Attacks.

Table 6. CVEs of AMD SEV(-ES/SNP) and Intel TDX up to June 2024. Attack types are classified into host-to-guest, guest-to-guest, guest-to-host, and host-to-host attacks, depending on the attacker's position and the target. For example, host-to-guest attacks are attacks where the host attacks the guest. For software bugs, if the firmware update fixes the issue, we classify them as a firmware bug. We list full CVEs in the appendix.

Attack Type	CVM	Software bugs kernel / firmware	Hardware bugs	Design issues	Total
Host to Guest	AMD SEV-SNP	1 / 17	3	4	25
	AMD SEV/-ES	2 / 7	2	2	13
	Intel TDX	3 / 5	0	1	9
Guest to Guest	AMD SEV-SNP	0 / 1	1	0	2
	AMD SEV/-ES	0 / 0	1	0	1
	Intel TDX	0	0	0	0
Guest to Host	AMD SEV-SNP	1 / 0	0	0	1
	AMD SEV/-ES	2 / 0	0	0	2
	Intel TDX	0	0	0	0
Host to Host	AMD SEV-SNP	1 / 0	0	0	1
	AMD SEV/-ES	3 / 1	0	0	4
	Intel TDX	0	0	0	0

Software bugs. Software bugs are the most common in CVEs and are the main source of host-to-guest attacks. Notably, the majority of the software bugs are due to improper input validation (11 out of 18 for AMD SEV-SNP, 7 out of 9 for AMD SEV(-ES), and 4 out of 8 for Intel TDX). Further, many software bugs are in the firmware and their fixing requires firmware updates. For AMD SEV-SNP, 17 out of 18; for AMD SEV(-ES), 7 out of 9; and for Intel TDX, 5 out of 8 bugs are related to the firmware.

Hardware bugs. Some CVEs occur due to the improper implementation of the hardware. One category of these bugs is the improper implementation of the hardware primitives, such as in CVE-2021-26342, where the hardware does not properly handle TLB flush on AMD SEV(-ES), leading to information leakage. Side-channel attacks constitute another category. This includes the power-glitch side-channel (CVE-2023-20575) and the cipher text side channels (CVE-2023-46744) on AMD SEV-SNP.

Design issues. Several CVEs are due to design issues in the CVM mechanisms. The lack of the RMP allows arbitrary code execution on the guest through remapping attack on AMD SEV(-ES) (CVE-2020-12967 and CVE-2021-26311). AMD SEV-SNP and Intel TDX do not face these issues.

Some CVEs are related to interrupt and exception handling of CVMs. Heckler [175] (CVE-2024-25743, CVE-2024-25744) shows that the ability of the VMM to inject exceptions can cause unexpected behavior by invoking an exception handler unexpectedly for both AMD SEV-SNP and Intel TDX. WeSee [174] (CVE-2024-25742) shows that an injection of #VC also causes a similar problem for AMD SEV-SNP(/ES). Intel TDX does not allow injecting interrupts below vector 32, mitigating the issue. These CVEs highlight the importance of input validation and input filtering from the VMM to reduce security problems.

5.2.2 *Other Attacks.*

Guest-to-guest attacks. Vulnerabilities in the platform firmware can also lead to guest-to-guest attacks. CVE-2023-31346 allows an SEV-SNP guest to leak information in the ASP using a crafted request to the ASP. This is caused by the improper input validation in the ASP firmware.

Guest-to-host attacks. While the majority of the CVEs are host-to-guest attacks, there are some CVEs regarding guest-to-host attacks. For example, CVE-2023-4155, CVE-2021-47008, and

CVE-2021-4093 are related to improper hypercall (VMGEXIT) handling on the host side, which can result in the guest leaking information on the host or causes DOS on the AMD SEV platform.

Host-to-host attacks. There are also CVEs related to host-to-host attacks. Some of them are due to the improper implementation of the VMM. For example, CVE-2023-47026 is caused by a NULL pointer dereference in the incorrect KVM API implementation. The others are due to erroneous implementations in the ASP firmware where the malicious host can send a crafted request to the ASP firmware to execute a DOS attack on the host (CVE-2024-26696, CVE-2022-0171, and CVE-2021-47389).

Takeaway #8: Among the reported CVEs (up to June 2024), 47 CVEs (81%) are host-to-guest attacks introduced by both software and hardware bugs. 22 CVEs (39%) are attributed to improper validation mechanisms, while 31 (54%) are associated with vulnerabilities in the underlying firmware. On the other hand, there are 3 CVEs on guest-to-host and 5 CVEs host-to-host attacks, highlighting the complexity of the CVM software stacks and emphasizing the importance of testing their interfaces.

6 Related Work

Architecture analysis on CVMs. Guanciale et al. [85] review the architectures of AMD-SEV, ARM CCA, IBM PEF, and Intel TDX and make comparisons about their features, security guarantees, and usability. Jauernig et al. [116] compare traditional enclaves (Intel SGX, Arm TrustZone, Sanctum [43], Sanctuary [31]) and AMD SEV. Akram et al. [4] summarize existing TEEs, including AMD SEV-SNP and Intel TDX, in the context of HPC. Sahita et al. [171] analyze the ISA and attestation protocol of Intel TDX and describe possible threats. Cheng et al. [35] summarize Intel TDX architecture in a top-down manner based on the existing specifications. Li et al. [125] discuss the design choices of TEEs, including AMD SEV and Intel TDX. Wang and Oswald [185] summarize GPU TEEs and their relation to the CPU TEEs includig AMD SEV and Intel TDX. These works are complementary to our work.

Security analysis on AMD SEV-SNP and Intel TDX. Google presents a security review on AMD SEV-SNP [37] and Intel TDX [6]. Buhren et al. [33] analyze the attestation mechanism of AMD SEV, and Sardar et al. [173] perform formal verification on the attestation mechanism of Intel TDX. Ménétrey et al. [141] compare the attestation mechanisms of several TEEs, including SEV. Witharana et al. [193] verify the memory confidentiality and integrity property of Intel TDX using model checking. These works mainly focus on security analysis and are orthogonal to our work.

Performance evaluation on AMD SEV(-SNP) and Intel TDX. Intel publishes performance evaluations on TME [113] and Intel TDX on 4th generation Xeon processors [107]. AMD presents a performance evaluation on AMD SEV-SNP of 3rd generation EPYC processors using CPU-intensive workloads [38]. Mofrad et al. [149] and Göttel et al. [81] evaluate the memory encryption performance of AMD SEV. Akram et al. [5] perform performance evaluation on AMD SEV, Bifrost [123] evaluates network I/O, and Qiu et al. [165] evaluate database workloads on AMD SEV-ES/SNP. Yan and Gopalan [196] evaluate AMD SEV-SNP performance with several benchmarks. Segarra et al. [176] evaluate the performance of SEV in the context of serverless computing, and SEVeriFast [89] examines the boot performance of SEV-SNP VMs and proposes an optimized boot process for microVMs.

Compared to these works, this paper presents a thorough evaluation of both AMD SEV-SNP and Intel TDX in a unified manner using real hardware (4th generation EPYC Processors and 5th Gen Xeon Scalable processors). We detail notable points compared to previous work in § 4.7.

7 Conclusion

This paper provides a comprehensive empirical analysis of two prominent CVM technologies: AMD SEV-SNP and Intel TDX. While CVMs offer substantial benefits for trusted cloud computing, our study identifies areas requiring reconsiderations and optimizations to enhance their performance and security properties. Our detailed performance evaluation highlights significant impacts on boot time, memory management overheads, I/O performance implications, and context-switching inefficiencies. Our security analysis, based on TCB evaluation and CVE analysis, also mandates robust validation mechanisms and proper firmware testing and development. Overall, our findings provide valuable insights that contribute to the improvement and broader adoption of CVMs in cloud infrastructures. Future efforts should address the CVM inefficiencies by refining the underlying hardware and software stacks.

Software artifact & additional material. The evaluation code is available at https://github.com/TUM-DSE/CVM_eval. We present additional material in the § Appendix.

Acknowledgement. We thank our shepherd, Prof. Adwait Jog, and the anonymous reviewers for their helpful comments. We thank Intel for providing them with access to a TDX enabled machine and thank Benny Fuhry for the assistance in the TDX machine access. We thank Shiny Sebastian for valuable technical discussions on Intel TDX. We thank Robert Schambach and Luca Mathias for their initial contribution to the experiments. This work was partially supported by an ERC Starting Grant (ID: 101077577) and the Chips Joint Undertaking (JU), European Union (EU) HORIZON-JU-IA, under grant agreement No. 101140087 (SMARTY). The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the programme of "Souverän. Digital. Vernetzt.". Joint project 6G-life, project identification number: 16KISK002. This work was also supported by the Fundação para a Ciência e Tecnologia (FCT) under grant UIDB/50021/2020, and by IAPMEI under grant C6632206063-00466847 (SmartRetail).

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[2] John Abbott. 2017. Trusting in the CPU: Getting to the Roots of Security. Retrieved 2024-07-24 from https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/Trusting-in-the-CPU.pdf

[3] Adil Ahmad, Botong Ou, Congyu Liu, Xiaokuan Zhang, and Pedro Fonseca. 2024. VEIL: A Protected Services Framework for Confidential Virtual Machines. In *Proceedings of the 29th International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM. https://doi.org/10.1145/3623278.3624763

[4] Ayaz Akram, Venkatesh Akella, Sean Peisert, and Jason Lowe-Power. 2022. SoK: Limitations of Confidential Computing via TEEs for High-Performance Compute Systems. In *Proceedings of the 2022 IEEE International Symposium on Secure and Private Execution Environment Design*. IEEE. https://doi.org/10.1109/SEED55351.2022.00018

[5] Ayaz Akram, Anna Giannakou, Venkatesh Akella, Jason Lowe-Power, and Sean Peisert. 2021. Performance Analysis of Scientific Computing Workloads on General Purpose TEEs. In *Proceedings of the 2021 IEEE International Parallel and Distributed Processing Symposium*. IEEE. https://doi.org/10.1109/IPDPS49936.2021.00115

[6] Erdem Aktas, Cfir Cohen, Josh Eads, James Forshaw, and Felix Wilhelm. 2023. Intel Trust Domain Extensions (TDX) Security Review April 2023. Retrieved 2024-07-24 from https://services.google.com/fh/files/misc/intel_tdx_-_full_report_041423.pdf

[7] A K M Mubashwir Alam and Keke Chen. 2023. Making Your Program Oblivious: A Comparative Study for Side-channel-Safe Confidential Computing. In *Proceedings of the IEEE 16th International Conference on Cloud Computing*. IEEE. https://doi.org/10.1109/CLOUD60044.2023.00040

[8] AMD. 2013. AMD Security and Server innovation. Presented at 2013 UEFI Spring PlugFest. Retrieved 2024-07-24 from https://uefi.org/sites/default/files/resources/UEFI_PlugFest_AMD_Security_and_Server_innovation_AMD_March_2013.pdf

[9] AMD. 2017. Protecting VM Register State with SEV-ES. Retrieved 2024-07-24 from https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/Protecting-VM-Register-State-with-SEV-ES.pdf

[10] AMD. 2020. AMD Secure Encrypted Virtualization API Version 0.24. Retrieved 2024-07-24 from https://www.amd.com/content/dam/amd/en/documents/epyc-technical-docs/programmer-references/55766_SEV-KM_API_Specification.pdf

[11] AMD. 2020. AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More. Retrieved 2024-07-24 from https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf

[12] AMD. 2021. AMD Memory Encryption. Retrieved 2024-07-24 from https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/memory-encryption-white-paper.pdf

[13] AMD. 2022. Confidential Guest Services with Secure VM Service Module on SEV-SNP. Retrieved 2024-07-24 from https://kvm-forum.qemu.org/2022/SEV-SNP-Confidential-Guest-Services-with-SVSM.pdf Presented at KVM Forum 2022.

[14] AMD. 2022. TECHNICAL GUIDANCE FOR MITIGATING EFFECTS OF CIPHERTEXT VISIBILITY UNDER AMD SEV REVISION 5.10.22. Retrieved 2024-07-24 from https://www.amd.com/system/files/documents/221404394-a_security_wp_final.pdf

[15] AMD. 2023. AMD SEV-TIO: Trusted I/O for Secure Encrypted Virtualization March 2023. Retrieved 2024-07-24 from https://www.amd.com/system/files/documents/sev-tio-whitepaper.pdf

[16] AMD. 2023. Secure VM Service Module for SEV-SNP Guests Guest Communication Interface Revision: 1.00. Retrieved 2024-07-24 from https://www.amd.com/content/dam/amd/en/documents/epyc-technical-docs/specifications/58019.pdf

[17] AMD. 2023. SEV-ES Guest-Hypervisor Communication Block Standardization. Retrieved 2024-07-24 from https://www.amd.com/content/dam/amd/en/documents/epyc-technical-docs/specifications/56421.pdf

[18] AMD. 2023. SEV Secure Nested Paging Firmware ABI Specification Revision: 1.55. Retrieved 2024-07-24 from https://www.amd.com/content/dam/amd/en/documents/epyc-technical-docs/specifications/56860.pdf

[19] AMD. 2024. AMD64 Architecture Programmer's Manual Volume 2: System Programming Revision 3.42. Retrieved 2024-07-24 from https://www.amd.com/content/dam/amd/en/documents/processor-tech-docs/programmer-references/24593.pdf

[20] AMD. [n. d.]. AMD PRO Security. Retrieved 2024-07-24 from https://www.amd.com/en/technologies/pro-security

[21] AMD. [n. d.]. AMD Secure Encrypted Virtualization (SEV). Retrieved 2024-07-24 from https://www.amd.com/en/developer/sev.html

[22] Sebastian Angel, Aditya Basu, Weidong Cui, Trent Jaeger, Stella Lau, Srinath Setty, and Sudheesh Singanamalla. 2023. Nimble: Rollback Protection for Confidential Cloud Services. In *Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association. https://www.usenix.org/conference/osdi23/presentation/angel

[23] ARM. [n. d.]. Arm Confidential Compute Architecture. Retrieved 2024-07-24 from https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture

[24] ARM. [n. d.]. Arm TrustZone for Cortex-M. Retrieved 2024-07-24 from https://www.arm.com/technologies/trustzone-for-cortex-m

[25] Raad Bahmani, Ferdinand Brasser, Ghada Dessouky, Patrick Jauernig, Matthias Klimmek, Ahmad-Reza Sadeghi, and Emmanuel Stapf. 2021. CURE: A Security Architecture with CUstomizable and Resilient Enclaves. In *Proceedings of the 30th USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/usenixsecurity21/presentation/bahmani

[26] Zijian Bao, Qinghao Wang, Wenbo Shi, Lei Wang, Hong Lei, and Bangdao Chen. 2020. When Blockchain Meets SGX: An Overview, Challenges, and Open Issues. *IEEE Access* (2020). https://doi.org/10.1109/ACCESS.2020.3024254

[27] Muli Ben-Yehuda, Michael D. Day, Zvi Dubitzky, Michael Factor, Nadav Har'El, Abel Gordon, Anthony Liguori, Orit Wasserman, and Ben-Ami Yassour. 2010. The Turtles Project: Design and Implementation of Nested Virtualization. In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association. https://www.usenix.org/conference/osdi10/turtles-project-design-and-implementation-nested-virtualization

[28] Stefan Berger, Ramon Caceres, Kenneth A. Goldman, Ronald Perez, Reiner Sailer, and Leendert van Doorn. 2006. vTPM: Virtualizing the Trusted Platform Module. In *Proceedings of the 15th USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/15th-usenix-security-symposium/vtpm-virtualizing-trusted-platform-module

[29] OpenMP Architecture Review Board. [n. d.]. OPENMP API Specification: Version 5.0 November 2018 – OMP_WAIT_POLICY. Retrieved 2024-07-24 from https://www.openmp.org/spec-html/5.0/openmpse55.html

[30] James Bottomley and Brijesh Singh. 2021. Encrypted Virtual Machine Images for Confidential Computing. Presented at KVM Forum 2021. Retrieved 2024-07-24 from https://research.ibm.com/publications/encrypted-virtual-machine-images-for-confidential-computing

[31] Ferdinand Brasser, David Gens, Patrick Jauernig, Ahmad-Reza Sadeghi, and Emmanuel Stapf. 2019. SANCTUARY: ARMing TrustZone with User-space Enclaves. In *Proceedings of the 2019 Network and Distributed System Security Symposium*. Internet Society. https://www.ndss-symposium.org/ndss-paper/sanctuary-arming-trustzone-with-user-space-enclaves/

[32] Robert Buhren, Shay Gueron, Jan Nordholz, Jean-Pierre Seifert, and Julian Vetter. 2017. Fault Attacks on Encrypted General Purpose Compute Platforms. In *Proceedings of the 7th ACM on Conference on Data and Application Security and Privacy*. ACM. https://doi.org/10.1145/3029806.3029836

[33] Robert Buhren, Christian Werling, and Jean-Pierre Seifert. 2019. Insecure Until Proven Updated: Analyzing AMD SEV's Remote Attestation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3319535.3354216

[34] Chia che Tsai, Donald E. Porter, and Mona Vij. 2017. Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX. In *Proceedings of the 2017 USENIX Annual Technical Conference*. USENIX Association. https://www.usenix.org/conference/atc17/technical-sessions/presentation/tsai

[35] Pau-Chen Cheng, Wojciech Ozga, Enriquillo Valdez, Salman Ahmed, Zhongshu Gu, Hani Jamjoom, Hubertus Franke, and James Bottomley. 2024. Intel TDX Demystified: A Top-Down Approach. *ACM Comput. Surv.* (mar 2024). https://doi.org/10.1145/3652597

[36] Shekha Chenthara, Khandakar Ahmed, Hua Wang, and Frank Whittaker. 2019. Security and Privacy-Preserving Challenges of e-Health Solutions in Cloud Computing. *IEEE Access* 7 (2019).

[37] Cfir Cohen, James Forshaw, Jann Horn, and Mark Brand. 2022. AMD Secure Processor for Confidential Computing Security Review May 2022. Retrieved 2024-07-24 from https://storage.googleapis.com/gweb-uniblog-publish-prod/documents/AMD_GPZ-Technical_Report_FINAL_05_2022.pdf

[38] Lynn Comp. 2021. Microsoft Azure Confidential Computing Powered by 3rd Gen EPYC CPUs – AMD Community Blog. Retrieved 2024-07-24 from https://community.amd.com/t5/epyc-processors/microsoft-azure-confidential-computing-powered-by-3rd-gen-epyc/ba-p/497796

[39] The Confidential Computing Consortium. 2022. Confidential Computing: Hardware-Based Trusted Execution for Applications and Data: November 2022, V1.3. Retrieved 2024-07-24 from https://confidentialcomputing.io/wp-content/uploads/sites/10/2023/03/CCC_outreach_whitepaper_updated_November_2022.pdf

[40] Jonathan Corbet. 2021. A Firewall for Device Drivers [LWN.net]. Retrieved 2024-07-24 from https://lwn.net/Articles/865918/

[41] Jonathan Corbet. [n. d.]. Guest-first memory for KVM [LWN.net]. Retrieved 2024-07-24 from https://lwn.net/Articles/949277/

[42] The MITRE Corporation. [n. d.]. CVE. Retrieved 2024-07-24 from https://cve.mitre.org/

[43] Victor Costan, Ilia Lebedev, and Srinivas Devadas. 2016. Sanctum: Minimal Hardware Extensions for Strong Software Isolation. In *Proceedings of the 25th USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/costan

[44] Nikunj A. Dadhania. 2021. Secure TSC for AMD SEV-SNP Guests. Retrieved 2024-07-24 from https://lpc.events/event/17/contributions/1525/attachments/1351/2702/Secure%20TSC%20for%20AMD%20SEV-SNP%20guests.pdf

[45] Al Danial. [n. d.]. CLOC: Count Lines of Code. Retrieved 2024-07-24 from https://github.com/AlDanial/cloc

[46] Antoine Delignat-Lavaud, Cédric Fournet, Kapil Vaswani, Sylvan Clebsch, Maik Riechert, Manuel Costa, and Mark Russinovich. 2023. Why Should I Trust Your Code? Confidential Computing Enables Users to Authenticate Code Running in TEEs, but Users Also Need Evidence This Code is Trustworthy. *Queue* 21, 4 (2023). https://doi.org/10.1145/3623460

[47] Sen Deng, Mengyuan Li, Yining Tang, Shuai Wang, Shoumeng Yan, and Yinqian Zhang. 2023. CipherH: Automated Detection of Ciphertext Side-channel Vulnerabilities in Cryptographic Implementations. In *Proceedings of the 32nd USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/usenixsecurity23/presentation/deng-sen

[48] Yunjie Deng, Chenxu Wang, Shunchang Yu, Shiqing Liu, Zhenyu Ning, Kevin Leach, Jin Li, Shoumeng Yan, Zhengyu He, Jiannong Cao, and Fengwei Zhang. 2022. StrongBox: A GPU TEE on Arm Endpoints. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3548606.3560627

[49] Asterinas Developers. 2024. Asterinas. Retrieved 2024-07-24 from https://asterinas.github.io/

[50] Blender Developers. [n. d.]. blender.org - Home of the Blender project - Free and Open 3D Creation Software. Retrieved 2024-07-24 from https://www.blender.org/

[51] COCONUT-SVSM Developers. [n. d.]. COCONUT Secure VM Service Module. Retrieved 2024-10-06 from https://github.com/coconut-svsm/svsm

[52] Fio Developers. [n. d.]. Flexible I/O Tester. Retrieved 2024-07-24 from https://github.com/axboe/fio

[53] Linux Kernel Developers. [n. d.]. dm-crypt. Retrieved 2024-07-24 from https://www.kernel.org/doc/html/latest/admin-guide/device-mapper/dm-crypt.html

[54] Linux Kernel Developers. [n. d.]. dm-integrity. Retrieved 2024-07-24 from https://www.kernel.org/doc/html/latest/admin-guide/device-mapper/dm-integrity.html

[55] Linux Kernel Developers. [n. d.]. dm-verity. Retrieved 2024-07-24 from https://www.kernel.org/doc/html/latest/admin-guide/device-mapper/verity.html

[56] Linux Kernel Developers. [n. d.]. Guest halt polling – The Linux Kernel Documentation. Retrieved 2024-07-24 from https://docs.kernel.org/virt/guest-halt-polling.html

[57] Memcached Developers. [n. d.]. Memcached - A Distributed Memory Object Caching System. Retrieved 2024-07-24 from https://www.nginx.com/

[58] MBW Developers. [n. d.]. raas/mbw: Memory Bandwidth Benchmark. Retrieved 2024-07-24 from https://github.com/raas/mbw

[59] Nginx Developers. [n. d.]. Advanced Load Balancer, Web Server, & Reverse Proxy – NGINX. Retrieved 2024-07-24 from https://www.nginx.com/

[60] OVMF Developers. [n. d.]. OVMF. Retrieved 2024-07-24 from https://github.com/tianocore/tianocore.github.io/wiki/OVMF

[61] PyTorch Developers. [n. d.]. PyTorch. Retrieved 2024-07-24 from https://pytorch.org/

[62] QEMU Developers. [n. d.]. Direct Linux Boot. Retrieved 2024-07-24 from https://www.qemu.org/docs/master/system/linuxboot.html

[63] RamSpeed Developers. [n. d.]. cruvolo/ramspeed-smp: RAMspeed/SMP, a Cache and Memory Benchmarking Tool. Retrieved 2024-07-24 from https://github.com/cruvolo/ramspeed-smp

[64] Redis Developers. [n. d.]. memtier_benchmark: A High-Throughput Benchmarking Tool for Redis & Memcached. Retrieved 2024-07-24 from https://redis.io/blog/memtier_benchmark-a-high-throughput-benchmarking-tool-for-redis-memcached/

[65] TD-shim Developers. 2024. TD-shim - Confidential Containers Shim Firmware. Retrieved 2024-07-24 from https://github.com/confidential-containers/td-shim/

[66] Tinymembench Developers. [n. d.]. ssvb/tinymembench: Simple Benchmark for Memory Throughput and Latency. Retrieved 2024-07-24 from https://github.com/ssvb/tinymembench

[67] TensorFlow Developers. [n. d.]. TensorFlow. Retrieved 2024-07-24 from https://www.tensorflow.org/

[68] UnixBench developers. [n. d.]. UnixBench. Retrieved 2024-07-24 from https://github.com/kdlucas/byte-unixbench

[69] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

[70] Gobikrishna Dhanuskodi, Sudeshna Guha, Vidhya Krishnan, Aruna Manjunatha, Michael O'Connor, Rob Nertney, and Phil Rogers. 2023. Creating the First Confidential GPUs: The Team at NVIDIA Brings Confidentiality and Integrity to User Code and Data for Accelerated Computing. *Queue* 21, 4 (2023). https://doi.org/10.1145/3623393.3623391

[71] Baltasar Dinis, Peter Druschel, Rodrigo Rodrigues, and Superior Técnico. 2023. RR: A Fault Model for Efficient TEE Replication. In *Proceedings of the 31st Annual Network and Distributed System Security Symposium*. Internet Society. https://www.ndss-symposium.org/ndss-paper/rr-a-fault-model-for-efficient-tee-replication/

[72] NASA Advanced Supercomputing (NAS) Division. [n. d.]. NAS Parallel Benchmarks. Retrieved 2024-07-24 from https://www.nas.nasa.gov/software/npb.html

[73] DMTF. [n. d.]. Security Protocols and Data Models. Retrieved 2024-07-24 from https://www.dmtf.org/standards/spdm

[74] Tom Dohrmann. 2024. Integrity Protect Workloads with Mushroom. Retrieved 2024-07-24 from https://fosdem.org/2024/schedule/event/fosdem-2024-2461-integrity-protect-workloads-with-mushroom/

[75] Dong Du and Bicheng Yang. 2023. Accelerating Extra Dimensional Page Walks for Confidential Computing. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM. https://doi.org/10.1145/3613424.3614293

[76] Xinyang Ge, Hsuan-Chi Kuo, and Weidong Cui. 2022. Hecate: Lifting and Shifting On-Premises Workloads to an Untrusted Cloud. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3548606.3560592

[77] Peter Gonda. 2022. Using DICE Attestation for SEV and SNP Hardware Rooted Attestation. Presented at Linux Plumbers Conference 2022. Retrieved 2024-07-24 from https://lpc.events/event/16/contributions/1319/attachments/1117/2144/Plumbers%20CC%20DICE.pdf

[78] Google. 2019. Google Cloud Confidential Computing. Retrieved 2024-07-24 from https://cloud.google.com/confidential-computing/

[79] Google. [n. d.]. Confidentail Computing | Google Cloud. Retrieved 2024-07-24 from https://cloud.google.com/security/products/confidential-computing

[80] Google. [n. d.]. google/go-tdx-guest: TDX Guest. Retrieved 2024-07-24 from https://github.com/google/go-tdx-guest

[81] Christian Göttel, Rafael Pires, Isabelly Rocha, Sébastien Vaucher, Pascal Felber, Marcelo Pasin, and Valerio Schiavoni. 2018. Security, Performance and Energy Trade-Offs of Hardware-Assisted Memory Protection Mechanisms. In *Proceedings of the 37th Symposium on Reliable Distributed Systems*. IEEE. https://doi.org/10.1109/SRDS.2018.00024

[82] Alibaba Group. [n. d.]. Build a TDX Confidential Computing Environment – Elastic Compute Service – Alibaba Cloud Documentation Center. Retrieved 2024-07-24 from https://www.alibabacloud.com/help/en/ecs/user-guide/build-a-tdx-confidential-computing-environment

[83] Trusted Computing Group. 2014. TCG EFI Platform Specification For TPM Family 1.1 or 1.2 Specification Version 1.22 Revision 15. Retrieved 2024-07-24 from https://trustedcomputinggroup.org/wp-content/uploads/TCG_EFI_Platform_1_22_Final_-v15.pdf

[84] Trusted Computing Group. 2018. Hardware Requirements for a Device Identifier Composition Engine Family 2.0 Level 00 Revision 78. Retrieved 2024-07-24 from https://trustedcomputinggroup.org/wp-content/uploads/Hardware-Requirements-for-Device-Identifier-Composition-Engine-r78_For-Publication.pdf

[85] Roberto Guanciale, Nicolae Paladi, and Arash Vahidi. 2022. SoK: Confidential Quartet - Comparison of Platforms for Virtualization-Based Confidential Computing. In *Proceedings of the 2022 IEEE International Symposium on Secure and Private Execution Environment Design*. IEEE. https://doi.org/10.1109/SEED55351.2022.00017

[86] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. 2008. Lest We Remember: Cold-Boot Attacks on Encryption Keys. In *Proceedings of the 17th USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/17th-usenix-security-symposium/lest-we-remember-cold-boot-attacks-encryption-keys

[87] Felicitas Hetzelt and Robert Buhren. 2017. Security Analysis of Encrypted Virtual Machines. In *Proceedings of the 13th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. ACM. https://doi.org/10.1145/3050748.3050763

[88] Felicitas Hetzelt, Martin Radev, Robert Buhren, Mathias Morbitzer, and Jean-Pierre Seifert. 2021. VIA: Analyzing Device Interfaces of Protected Virtual Machines. In *Proceedings of the 2021 Annual Computer Security Applications Conference*. ACM. https://doi.org/10.1145/3485832.3488011

[89] Benjamin Holmes, Jason Waterman, and Dan Williams. 2024. SEVeriFast: Minimizing the Root of Trust for Fast Startup of SEV microVMs. In *Proceedings of the 29th International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM. https://doi.org/10.1145/3620665.3640424

[90] Heidi Howard, Fritz Alder, Edward Ashton, Amaury Chamayou, Sylvan Clebsch, Manuel Costa, Antoine Delignat-Lavaud, Cédric Fournet, Andrew Jeffery, Matthew Kerner, Fotios Kounelis, Markus A. Kuppe, Julien Maffre, Mark Russinovich, and Christoph M. Wintersteiger. 2023. Confidential Consortium Framework: Secure Multiparty Applications with Confidentiality, Integrity, and High Availability. *Proc. VLDB Endow.* 17, 2 (2023). https://doi.org/10.14778/3626292.3626304

[91] Guerney D. H. Hunt, Ramachandra Pai, Michael V. Le, Hani Jamjoom, Sukadev Bhattiprolu, Rick Boivie, Laurent Dufour, Brad Frey, Mohit Kapur, Kenneth A. Goldman, Ryan Grimm, Janani Janakirman, John M. Ludden, Paul Mackerras, Cathy May, Elaine R. Palmer, Bharata Bhasker Rao, Lawrence Roy, William A. Starke, Jeff Stuecheli, Enriquillo Valdez, and Wendel Voigt. 2021. Confidential Computing for OpenPOWER. In *Proceedings of the 16th European Conference on Computer Systems*. ACM. https://doi.org/10.1145/3447786.3456243

[92] Tyler Hunt, Zhipeng Jia, Vance Miller, Ariel Szekely, Yige Hu, Christopher J. Rossbach, and Emmett Witchel. 2020. Telekine: Secure Computing with Cloud GPUs . In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association. https://www.usenix.org/conference/nsdi20/presentation/hunt

[93] IBM. 2019. Confidential computing on IBM Cloud. Retrieved 2024-07-24 from https://www.ibm.com/cloud/confidential-computing

[94] IBM. [n. d.]. IBM Documentation – Introducing IBM Secure Execution for Linux. Retrieved 2024-07-24 from https://www.ibm.com/docs/en/linux-on-systems?topic=virtualization-secure-execution

[95] IETF. [n. d.]. The Transport Layer Security (TLS) Protocol Version 1.3. Retrieved 2024-07-24 from https://datatracker.ietf.org/doc/html/rfc8446

[96] Intel. [n. d.]. Runtime Integrity Measurement and Attestation in a Trust Domain. Retrieved 2024-07-24 from https://www.intel.com/content/www/us/en/developer/articles/community/runtime-integrity-measure-and-attest-trust-domain.html

[97] Intel. 2018. Host Firmware Speculative Execution Side Channel Mitigation. Retrieved 2024-07-24 from https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/technical-documentation/host-firmware-speculative-side-channel-mitigation.html

[98] Intel. 2022. Guidelines for Mitigating Timing Side Channels Against Cryptographic Implementations. Retrieved 2024-07-24 from https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/secure-coding/mitigate-timing-side-channel-crypto-implementation.html

[99] Intel. 2022. Intel Architecture Memory Encryption Technologies Specification Revision 1.4. Retrieved 2024-07-24 from https://cdrdv2-public.intel.com/679154/multi-key-total-memory-encryption-spec-1.4.pdf

[100] Intel. 2022. White Paper Intel® Trust Domain Extensions. Retrieved 2024-07-24 from https://cdrdv2-public.intel.com/690419/TDX-Whitepaper-February2022.pdf

[101] Intel. 2023. Architecture Specification: Intel® Trust Domain Extensions (Intel® TDX) Module Version 1.0. Retrieved 2024-07-24 from https://cdrdv2.intel.com/v1/dl/getContent/733568

[102] Intel. 2023. intel/MigTD. Retrieved 2024-07-24 from https://github.com/intel/MigTD

[103] Intel. 2023. intel/vtpm-td. Retrieved 2024-07-24 from https://github.com/intel/vtpm-td

[104] Intel. 2023. Intel® TDX Connect Architecture Specification March 2023. Retrieved 2024-07-24 from https://www.intel.com/content/www/us/en/content-details/773614/intel-tdx-connect-architecture-specification.html

[105] Intel. 2023. Intel® TDX Module Architecture Application Binary Interface (ABI) Reference Specification. Retrieved 2024-07-24 from https://www.intel.com/content/www/us/en/content-details/795381/intel-tdx-module-architecture-application-binary-interface-abi-reference-specification.html

[106] Intel. 2023. Intel® TDX Module v1.5 TD Partitioning Architecture Specification November 2023. Retrieved 2024-07-24 from https://www.intel.com/content/www/us/en/content-details/795474/intel-tdx-module-v1-5-td-partitioning-architecture-specification.html

[107] Intel. 2023. Performance Considerations of Intel® Trust Domain Extensions on 4th Generation Intel® Xeon® Scalable Processors. Retrieved 2024-07-24 from https://www.intel.com/content/www/us/en/developer/articles/technical/trust-domain-extensions-on-4th-gen-xeon-processors.html

[108] Intel. [n. d.]. Intel Software Guard Extensions. Retrieved 2024-07-24 from https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html

[109] Intel. [n. d.]. intel/SGXDataCenterAttestationPrimitives – tdx_attest. Retrieved 2024-07-24 from https://github.com/intel/SGXDataCenterAttestationPrimitives/tree/main/QuoteGeneration/quote_wrapper/tdx_attest

[110] Intel. [n. d.]. Intel® Memory Latency Checker (Intel® MLC). Retrieved 2024-07-24 from https://www.intel.com/content/www/us/en/download/736633/intel-memory-latency-checker-intel-mlc.html

[111] Intel. [n. d.]. Intel® Trust Domain Extension Guest Kernel Hardening Documentation. Retrieved 2024-07-24 from https://intel.github.io/ccc-linux-guest-hardening-docs/index.html

[112] Intel. [n. d.]. Intel® Trust Domain Extensions (Intel TDX). Retrieved 2024-07-24 from https://www.intel.com/content/www/us/en/developer/articles/technical/intel-trust-domain-extensions.html

[113] Intel. [n. d.]. Runtime Encryption of Memory with Intel Total Memory Encryption – Multi-Key (IntelTME-MK). Retrieved 2024-07-24 from https://www.intel.com/content/www/us/en/developer/articles/news/runtime-encryption-of-memory-with-intel-tme-mk.html

[114] iPerf Developers. [n. d.]. iPerf - The ultimate speed test tool for TCP, UDP and SCTP. Retrieved 2024-07-24 from https://iperf.fr/

[115] Insu Jang, Adrian Tang, Taehoon Kim, Simha Sethumadhavan, and Jaehyuk Huh. 2019. Heterogeneous Isolated Execution for Commodity GPUs. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM. https://doi.org/10.1145/3297858.3304021

[116] Patrick Jauernig, Ahmad-Reza Sadeghi, and Emmanuel Stapf. 2020. Trusted Execution Environments: Properties, Applications, and Challenges. *IEEE Security & Privacy* 18, 2 (2020), 56–60.

[117] Jianyu Jiang, Ji Qi, Tianxiang Shen, Xusheng Chen, Shixiong Zhao, Sen Wang, Li Chen, Gong Zhang, Xiapu Luo, and Heming Cui. 2022. CRONUS: Fault-isolated, Secure and High-performance Heterogeneous Computing for Trusted Execution Environment. In *Proceedings of the 55th IEEE/ACM International Symposium on Microarchitecture*. IEEE. https://doi.org/10.1109/MICRO56248.2022.00019

[118] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping Bits in Memory without Accessing Them: An Experimental Study of DRAM Disturbance Errors. In *Proceedings of the 2014 ACM/IEEE 41st International Symposium on Computer Architecture*. IEEE. https://doi.org/10.1109/ISCA.2014.6853210

[119] Thomas Knauth, Michael Steiner, Somnath Chakrabarti, Li Lei, Cedric Xing, and Mona Vij. 2019. Integrating Remote Attestation with Transport Layer Security. arXiv:1801.05863 [cs.CR] https://arxiv.org/abs/1801.05863

[120] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of Advances in Neural Information Processing Systems 25*. Curran Associates, Inc. https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[121] Dmitrii Kuvaiskii, Dimitrios Stavrakakis, Kailun Qin, Cedric Xing, Pramod Bhatotia, and Mona Vij. 2024. Gramine-TDX: A Lightweight OS Kernel for Confidential VMs. In *Proceedings of the 31st ACM Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3658644.3690323

[122] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: An Open Framework for Architecting Trusted Execution Environments. In *Proceedings of the 15th European Conference on Computer Systems*.

ACM. https://doi.org/10.1145/3342195.3387532

[123] Dingji Li, Zeyu Mi, Chenhui Ji, Yifan Tan, Binyu Zang, Haibing Guan, and Haibo Chen. 2023. Bifrost: Analysis and Optimization of Network I/O Tax in Confidential Virtual Machines. In *Proceedings of the 2023 USENIX Annual Technical Conference*. USENIX Association. https://www.usenix.org/conference/atc23/presentation/li-dingji

[124] Mengyuan Li, Luca Wilke, Jan Wichelmann, Thomas Eisenbarth, Radu Teodorescu, and Yinqian Zhang. 2022. A Systematic Look at Ciphertext Side Channels on AMD SEV-SNP. In *Proceedings of the 43rd IEEE Symposium on Security and Privacy*. IEEE. https://doi.org/10.1109/SP46214.2022.9833768

[125] Mengyuan Li, Yuheng Yang, Guoxing Chen, Mengjia Yan, and Yinqian Zhang. 2024. SoK: Understanding Design Choices and Pitfalls of Trusted Execution Environments. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3634737.3644993

[126] Mengyuan Li, Yinqian Zhang, and Zhiqiang Lin. 2021. CrossLine: Breaking "Security-by-Crash" Based Memory Isolation in AMD SEV. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3460120.3485253

[127] Mengyuan Li, Yinqian Zhang, Zhiqiang Lin, and Yan Solihin. 2019. Exploiting Unprotected I/O Operations in AMD's Secure Encrypted Virtualization. In *Proceedings of the 28th USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/usenixsecurity19/presentation/li-mengyuan

[128] Mengyuan Li, Yinqian Zhang, Huibo Wang, Kang Li, and Yueqiang Cheng. 2021. CIPHERLEAKS: Breaking Constant-time Cryptography on AMD SEV via the Ciphertext Side Channel. In *Proceedings of the 30th USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/usenixsecurity21/presentation/li-mengyuan

[129] Mengyuan Li, Yinqian Zhang, Huibo Wang, Kang Li, and Yueqiang Cheng. 2021. TLB Poisoning Attacks on AMD Secure Encrypted Virtualization. In *Proceedins of the 2021 Annual Computer Security Applications Conference*. ACM. https://doi.org/10.1145/3485832.3485876

[130] Xiaoxuan Lou, Kangjie Chen, Guowen Xu, Han Qiu, Shangwei Guo, and Tianwei Zhang. 2024. Protecting Confidential Virtual Machines from Hardware Performance Counter Side Channels. In *Proceedings of the 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE. https://doi.org/10.1109/DSN58291.2024.00031

[131] Abhishek Mahalle, Jianming Yong, Xiaohui Tao, and Jun Shen. 2018. Data Privacy and System Security for Banking and Financial Services Industry based on Cloud Computing Infrastructure. In *Proceedings of the IEEE 22nd International Conference on Computer Supported Cooperative Work in Design*. https://doi.org/10.1109/CSCWD.2018.8465318

[132] HaoHui Mai, Jiacheng Zhao, Hongren Zheng, Yiyang Zhao, Zibin Liu, Mingyu Gao, Cong Wang, Huimin Cui, Xiaobing Feng, and Christos Kozyrakis. 2023. Honeycomb: Secure and Efficient GPU Executions via Static Validation. In *Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association. https://www.usenix.org/conference/osdi23/presentation/mai

[133] André Martin, Cong Lian, Franz Gregor, Robert Krahn, Valerio Schiavoni, Pascal Felber, and Christof Fetzer. 2021. ADAM-CS: Advanced Asynchronous Monotonic Counter Service. In *Proceedings of the 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. https://doi.org/10.1109/DSN48987.2021.00053

[134] Sinisa Matetic, Mansoor Ahmed, Kari Kostiainen, Aritra Dhar, David Sommer, Arthur Gervais, Ari Juels, and Srdjan Capkun. 2017. ROTE: Rollback Protection for Trusted Execution. In *Proceedings of the 26th USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/matetic

[135] John D. McCalpin. [n. d.]. STREAM: Sustainable Memory Bandwidth in High Performance Computers. Retrieved 2024-07-24 from https://www.cs.virginia.edu/stream/

[136] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. 2013. Innovative Instructions and Software Model for Isolated Execution. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*. ACM. https://doi.org/10.1145/2487726.2488368

[137] Phoronix Media. [n. d.]. CacheBench Benchmark – OpenBenchmarking.org. Retrieved 2024-07-24 from https://openbenchmarking.org/test/pts/cachebench

[138] Phoronix Media. [n. d.]. Memory Test Suite Test Suite Collection – OpenBenchmarking.org. Retrieved 2024-07-24 from https://openbenchmarking.org/suite/pts/memory

[139] Phoronix Media. [n. d.]. Phoronix Test Suite – Linux Testing & Benchmarking Platform, Automated Testing, Open-Source Benchmarking. Retrieved 2024-07-24 from https://www.phoronix-test-suite.com/

[140] Benshan Mei, Saisai Xia, Wenhao Wang, and Dongdai Lin. 2024. Cabin: Confining Untrusted Programs within Confidential VMs. In *Proceedings of the 26th International Conference on Information and Communications Security*. Springer Nature Singapore.

[141] Jämes Ménétrey, Christian Göttel, Anum Khurshid, Marcelo Pasin, Pascal Felber, Valerio Schiavoni, and Shahid Raza. 2022. Attestation Mechanisms for Trusted Execution Environments Demystified. In *Proceedings of the 2022 Distributed Applications and Interoperable Systems*. Springer International Publishing. https://doi.org/10.1007/978-3-031-16092-9_7

[142] Ralph C. Merkle. 1988. A Digital Signature Based on a Conventional Encryption Function. In *Advances in Cryptology — CRYPTO '87*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[143] Microsoft. 2019. Azure Confidential Computing. Retrieved 2024-07-24 from https://azure.microsoft.com/en-us/solutions/confidential-compute

[144] Microsoft. 2023. Virtual TPMs in Azure confidential VMs. Retrieved 2024-07-24 from https://learn.microsoft.com/en-us/azure/confidential-comp

[145] Microsoft. [n. d.]. Common Azure confidential computing scenarios and use cases | Microsoft Learn. Retrieved 2024-07-24 from https://learn.microsoft.com/en-us/azure/confidential-computing/use-cases-scenarios

[146] Masanori Misono, Toshiki Hatanaka, and Takahiro Shinagawa. 2022. DMAFV: Testing Device Drivers against DMA Faults. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. ACM. https://doi.org/10.1145/3477314.3507082

[147] Masanori Misono, Masahiro Ogino, Takaaki Fukai, and Takahiro Shinagawa. 2018. FaultVisor2: Testing Hypervisor Device Drivers Against Real Hardware Failures. In *Proceedings of the 2018 IEEE International Conference on Cloud Computing Technology and Science*. IEEE. https://doi.org/10.1109/CloudCom2018.2018.00048

[148] Fan Mo, Zahra Tarkhani, and Hamed Haddadi. 2024. Machine Learning with Confidential Computing: A Systematization of Knowledge. *ACM Comput. Surv.* 56, 11 (2024).

[149] Saeid Mofrad, Fengwei Zhang, Shiyong Lu, and Weidong Shi. 2018. A Comparison Study of Intel SGX and AMD Memory Encryption Technology. In *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*. ACM. https://doi.org/10.1145/3214292.3214301

[150] Apoorve Mohan, Mengmei Ye, Hubertus Franke, Mudhakar Srivatsa, Zhuoran Liu, and Nelson Mimura Gonzalez. 2024. Securing AI Inference in the Cloud: Is CPU-GPU Confidential Computing Ready?. In *Proceedings of the IEEE 17th International Conference on Cloud Computing*. IEEE. https://doi.org/10.1109/CLOUD62652.2024.00028

[151] Mathias Morbitzer, Manuel Huber, Julian Horsch, and Sascha Wessel. 2018. SEVered: Subverting AMD's Virtual Machine Encryption. In *Proceedings of the 11th European Workshop on Systems Security*. ACM. https://doi.org/10.1145/3193111.3193112

[152] Mathias Morbitzer, Sergej Proskurin, Martin Radev, Marko Dorfhuber, and Erick Quintanar Salas. 2021. SEVerity: Code Injection Attacks against Encrypted Virtual Machines. In *Proceedings of the 2021 IEEE Security and Privacy Workshops*. https://doi.org/10.1109/SPW53761.2021.00063

[153] Dov Murik. 2023. [edk2-devel] [PATCH v3 0/2] OvmfPkg: Enable Measured Direct Boot on AMD SEV-SNP. Retrieved 2024-07-24 from https://patchew.org/EDK2/20230302091532.1985238-1-dovmurik@linux.ibm.com/

[154] Dov Murik and Hubertus Franke. 2021. Securing Linux VM Boot with AMD SEV Measurement. Presented at KVM Forum 2021. Retrieved 2024-07-24 from https://kvmforum2021.sched.com/event/ke4h/securing-linux-vm-boot-with-amd-sev-measurement-dov-murik-hubertus-franke-ibm-research

[155] Vikram Narayanan, Claudio Carvalho, Angelo Ruocco, Gheorghe Almasi, James Bottomley, Mengmei Ye, Tobin Feldman-Fitzthum, Daniele Buono, Hubertus Franke, and Anton Burtsev. 2023. Remote Attestation of Confidential VMs Using Ephemeral vTPMs. In *Proceedings of the 2023 Annual Computer Security Applications Conference*. ACM. https://doi.org/10.1145/3627106.3627112

[156] Jianyu Niu, Wei Peng, Xiaokuan Zhang, and Yinqian Zhang. 2022. NARRATOR: Secure and Practical State Continuity for Trusted Execution in the Cloud. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3548606.3560620

[157] OASIS Open. [n. d.]. Virtual I/O Device (VIRTIO) Version 1.2. Retrieved 2024-07-24 from https://docs.oasis-open.org/virtio/virtio/v1.2/csd01/virtio-v1.2-csd01.html

[158] Oracle. 2019. Oracle Cloud Infrastructure Documentation – Confidential Computing. Retrieved 2024-07-24 from https://docs.oracle.com/en-us/iaas/Content/Compute/References/confidential_compute.htm

[159] Wojciech Ozga, Guerney D. H. Hunt, Michael V. Le, Elaine R. Palmer, and Avraham Shinnar. 2023. Towards a Formally Verified Security Monitor for VM-Based Confidential Computing. In *Proceedings of the 12th International Workshop on Hardware and Architectural Support for Security and Privacy*. ACM. https://doi.org/10.1145/3623652.3623668

[160] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of the Advances in Neural Information Processing Systems 32*. Curran Associates, Inc. https://papers.neurips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

[161] PCI-SIG. 2020. Integrity and Data Encryption (IDE) ECN Deep Dive. Retrieved 2024-07-24 from https://pcisig.com/sites/default/files/files/PCIe%20Security%20Webinar_Aug%202020_PDF.pdf

[162] PCI-SIG. 2022. TEE Device Interface Security Protocol (TDISP) Specification Revision 5.x. Retrieved 2024-07-24 from https://pcisig.com/tee-device-interface-security-protocol-tdisp

[163] Joana Pecholt and Sascha Wessel. 2022. CoCoTPM: Trusted Platform Modules for Virtual Machines in Confidential Computing Environments. In *Proceedings of the 38th Annual Computer Security Applications Conference*. ACM. https://doi.org/10.1145/3564625.3564648

[164] Caroline Perez-Vargas. 2023. Confidential VMs on Azure. Retrieved 2024-07-24 from https://techcommunity.microsoft.com/t5/windows-os-platform-blog/confidential-vms-on-azure/ba-p/3836282

[165] Lina Qiu, Rebecca Taft, Alexander Shraer, and George Kollios. 2024. The Price of Privacy: A Performance Study of Confidential Virtual Machines for Database Systems. In *Proceedings of the 20th International Workshop on Data Management on New Hardware*. ACM. https://doi.org/10.1145/3662010.3663440

[166] Martin Radev and Mathias Morbitzer. 2021. Exploiting Interfaces of Secure Encrypted Virtual Machines. In *Proceedings of the 2021 Reversing and Offensive-Oriented Trends Symposium*. ACM. https://doi.org/10.1145/3433667.3433668

[167] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://doi.org/10.18653/v1/D16-1264

[168] Mark Russinovich. [n. d.]. Azure AI Confidential Inferencing: Technical Deep-Dive. Retrieved 2024-10-06 from https://techcommunity.microsoft.com/t5/azure-confidential-computing/azure-ai-confidential-inferencing-technical-deep-dive/ba-p/4253150

[169] Marta Rybczyńska. 2019. Bounce Buffers for Untrusted Devices [LWN.net]. Retrieved 2024-07-24 from https://lwn.net/Articles/786558/

[170] Marta Rybczyńska. 2021. Hardening VirtIO [LWN.net]. Retrieved 2024-07-24 from https://lwn.net/Articles/865216/

[171] Ravi Sahita, Dror Caspi, Barry Huntley, Vincent Scarlata, Baruch Chaikin, Siddhartha Chhabra, Arie Aharon, and Ido Ouziel. 2021. Security Analysis of Confidential-compute Instruction Set Architecture for Virtualized Workloads. In *Proceedings of the 2021 International Symposium on Secure and Private Execution Environment Design*. IEEE Computer Society. https://doi.org/10.1109/SEED51797.2021.00024

[172] Ravi Sahita, Vedvyas Shanbhogue, Andrew Bresticker, Atul Khare, Atish Patra, Samuel Ortiz, Dylan Reid, and Rajnesh Kanwal. 2023. CoVE: Towards Confidential Computing on RISC-V Platforms. In *Proceedings of the 20th ACM International Conference on Computing Frontiers*. ACM. https://doi.org/10.1145/3587135.3592168

[173] Muhammad Usama Sardar, Saidgani Musaev, and Christof Fetzer. 2021. Demystifying Attestation in Intel Trust Domain Extensions via Formal Verification. *IEEE Access* 9 (2021). https://doi.org/10.1109/ACCESS.2021.3087421

[174] Benedict Schlüter, Supraja Sridhara, Andrin Bertschi, and Shweta Shinde. 2024. WESEE: Using Malicious #VC Interrupts to Break AMD SEV-SNP. In *Proceedings of the 45rd IEEE Symposium on Security and Privacy*. IEEE. https://doi.org/10.1109/SP54263.2024.00262

[175] Benedict Schlüter, Supraja Sridhara, Mark Kuhne, Andrin Bertschi, and Shweta Shinde. 2024. HECKLER: Breaking Confidential VMs with Malicious Interrupts. In *Proceedings of the 33rd USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/usenixsecurity24/presentation/schl%C3%BCter

[176] Carlos Segarra, Tobin Feldman-Fitzthum, Daniele Buono, and Peter Pietzuch. 2024. Serverless Confidential Containers: Challenges and Opportunities. In *Proceedings of the 2nd Workshop on SErverless Systems, Applications and MEthodologies*. ACM. https://doi.org/10.1145/3642977.3652097

[177] Amazon Web Services. 2019. AMD SEV-SNP - Amazon Elastic Compute Cloud. Retrieved 2024-07-24 from https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/sev-snp.html

[178] Kripa Shanker, Arun Joseph, and Vinod Ganapathy. 2020. An Evaluation of Methods to Port Legacy Code to SGX Enclaves. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, New York, NY, USA. https://doi.org/10.1145/3368089.3409726

[179] Satoru Takekoshi, Takahiro Shinagawa, and Kazuhiko Kato. 2016. Testing Device Drivers against Hardware Failures in Real Environmentsi. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM. https://doi.org/10.1145/2851613.2851740

[180] Michael S. Tsirkin and Stefan Hanjnoczi. 2023. Trust, Confidentiality, and Hardening The VirtIO Lessons. Retrieved 2024-07-24 from https://vmsplice.net/~stefan/stefanha-lpc-2023.pdf

[181] Dalton Cézane Gomes Valadares, Newton Carlos Will, Marco Aurélio Spohn, Danilo Freire de Souza Santos, Angelo Perkusich, and Kyller Costa Gorgônio. 2022. Confidential Computing in Cloud/fog-based Internet of Things Scenarios. *Internet of Things* 19 (2022).

[182] VirTEE. [n. d.]. virtee/snpguest: A CLI tool for interacting with SEV-SNP guest environment Resources. Retrieved 2024-07-24 from https://github.com/virtee/snpguest

[183] Stavros Volos, Kapil Vaswani, and Rodrigo Bruno. 2018. Graviton: Trusted Execution Environments on GPUs. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association. https://www.usenix.org/conference/osdi18/presentation/volos

[184] Pengfei Wang, Jens Krinke, Kai Lu, Gen Li, and Steve Dodier-Lazaro. 2017. How Double-Fetch Situations turn into Double-Fetch Vulnerabilities: A Study of Double Fetches in the Linux Kernel. In *Proceedings of the 26th USENIX*

Security Symposium. USENIX Association. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-pengfei

[185] Qifan Wang and David Oswald. 2024. Confidential Computing on Heterogeneous CPU-GPU Systems: Survey and Future Directions. arXiv:2408.11601 [cs.CR] https://arxiv.org/abs/2408.11601

[186] Wubing Wang, Mengyuan Li, Yinqian Zhang, and Zhiqiang Lin. 2023. PwrLeak: Exploiting Power Reporting Interface for Side-Channel Attacks on AMD SEV. In *Proceedings of the 20th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer-Verlag. https://doi.org/10.1007/978-3-031-35504-2_3

[187] Ofir Weisse, Valeria Bertacco, and Todd Austin. 2017. Regaining Lost Cycles with HotCalls: A Fast Interface for SGX Secure Enclaves. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ACM. https://doi.org/10.1145/3079856.3080208

[188] Jan Werner, Joshua Mason, Manos Antonakakis, Michalis Polychronakis, and Fabian Monrose. 2019. The SEVerESt Of Them All: Inference Attacks Against Secure Virtual Enclaves. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. ACM. https://doi.org/10.1145/3321705.3329820

[189] Jan Wichelmann, Anna Pätschke, Luca Wilke, and Thomas Eisenbarth. 2023. Cipherfix: Mitigating Ciphertext Side-Channel Attacks in Software. In *Proceedings of the 32nd USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/usenixsecurity23/presentation/wichelmann

[190] Luca Wilke, Jan Wichelmann, Mathias Morbitzer, and Thomas Eisenbarth. 2020. SEVurity: No Security Without Integrity : Breaking Integrity-Free Memory Encryption with Minimal Assumptions. In *Proceedings of the 41st IEEE Symposium on Security and Privacy*. IEEE. https://doi.org/10.1109/SP40000.2020.00080

[191] Luca Wilke, Jan Wichelmann, Florian Sieck, and Thomas Eisenbarth. 2021. undeSErVed trust: Exploiting Permutation-Agnostic Remote Attestation. In *Proceedings of the 2021 IEEE Security and Privacy Workshops*. IEEE Computer Society. https://doi.org/10.1109/SPW53761.2021.00064

[192] Newton C. Will and Carlos A. Maziero. 2023. Intel Software Guard Extensions Applications: A Survey. *ACM Comput. Surv.* 55, 14s (2023). https://doi.org/10.1145/3593021

[193] Hasini Witharana, Debapriya Chatterjee, and Prabhat Mishra. 2024. Verifying Memory Confidentiality and Integrity of Intel TDX Trusted Execution Environments. In *Proceedings of the 2024 IEEE International Symposium on Hardware Oriented Security and Trust*. IEEE Computer Society. https://doi.org/10.1109/HOST55342.2024.10545349

[194] wrk Developers. [n. d.]. wg/wrk: Modern HTTP Benchmarking Tool. Retrieved 2024-07-24 from https://github.com/wg/wrk

[195] Rafael J. Wysocki. [n. d.]. CPU Idle Time Management – The Linux Kernel Documentation. Retrieved 2024-07-24 from https://www.kernel.org/doc/html/v6.8/admin-guide/pm/cpuidle.html

[196] Mingjie Yan and Kartik Gopalan. 2023. Performance Overheads of Confidential Virtual Machines. In *Proceedings of the 31st International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. IEEE. https://doi.org/10.1109/MASCOTS59514.2023.10387607

[197] Jason Zhijingcheng Yu, Shweta Shinde, Trevor E. Carlson, and Prateek Saxena. 2022. Elasticlave: An Efficient Memory Model for Enclaves. In *Proceedings of the 31st USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/usenixsecurity22/presentation/yu-jason

[198] Ardhi Wiratama Baskara Yudha, Jake Meyer, Shougang Yuan, Huiyang Zhou, and Yan Solihin. 2022. LITE: a Low-cost Practical Inter-operable GPU TEE. In *Proceedings of the 36th ACM International Conference on Supercomputing*. ACM. https://doi.org/10.1145/3524059.3532361

[199] Ruiyi Zhang, Lukas Gerlach, Daniel Weber, Lorenz Hetterich, Youheng Lü, Andreas Kogler, and Michael Schwarz. 2024. CacheWarp: Software-based Fault Injection using Selective State Reset. In *Proceedings of the 33rd USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/usenixsecurity24/presentation/zhang-ruiyi

[200] Rui Zhang and Ling Liu. 2010. Security Models and Requirements for Healthcare Application Clouds. In *Proceedings of the IEEE 3rd International Conference on Cloud Computing*. https://doi.org/10.1109/CLOUD.2010.62

[201] Jianping Zhu, Rui Hou, XiaoFeng Wang, Wenhao Wang, Jiangfeng Cao, Boyan Zhao, Zhongpu Wang, Yuhui Zhang, Jiameng Ying, Lixin Zhang, and Dan Meng. 2020. Enabling Rack-scale Confidential Computing using Heterogeneous Trusted Execution Environment. In *Proceedings of the 41st IEEE Symposium on Security and Privacy*. IEEE. https://doi.org/10.1109/SP40000.2020.00054

Appendix

A Terminology Comparison

Table 7 shows the summary of terminology comparison between AMD SEV-SNP and Intel TDX.

Table 7. Terminology comparison between AMD SEV-SNP and Intel TDX

Terminology	AMD SEV-SNP	Intel TDX
Second-level address translation table	Nested page table (NPT)	Extended Page Table (EPT)
Host physical address	System physical address (SPA)	Host physical address (HPA)
CPU mode for vmm	Host mode / Hypervisor mode	VMX root-mode
CPU mode for a normal VM	Guest mode	VMX non-root mode
CPU mode for a CVM	Guest mode	SEAM VMX non-root mode
vCPU state data structure	VMCB (Virtual Machine Control Block)	VMCS (Virtual Machine Control State)
Name of a CVM domain	SEV-SNP VM	Trust domain (TD)
Exception to handle VMEXIT in CVMs	#VC (VMM Communication Exception)	#VE (Virtualization Exception)
Instruction to enter to a normal VM	VMRUN	VMLAUNCH / VMRESUME
Instruction to enter to a CVM	VMRUN	SEAMCALL [TDH.VP.ENTER]
Hypercall from a normal VM	VMMCALL	VMCALL
Hypercall from a CVM	VMGEXIT	TDCALL [TDG.VP.VMCALL]

B Support Status

Table 8 summarizes the status of the supported features on Intel TDX. Table 9 shows the information about the processors that support AMD SEV(-ES/SNP), and Table 10 compares the features among AMD SEV, SEV-ES, and SEV-SNP.

Table 8. Supported features and processor versions on Intel TDX. At the time of writing, there is no publicly available concrete implementation of live migration and TD-partitioning.

Feature	Required TDX Module Version	First Support Processor
Intel TDX	TDX 1.0	4th Gen Xeon Scalable Processors (private preview)
Live Migration	TDX 1.5	5th Gen Xeon Scalable Processors
TD-Partitioning	TDX 1.5	5th Gen Xeon Scalable Processors
TEE-IO	TDX 2.0	Future processors

Table 9. Processors that support AMD SEV(-ES/SNP).

Feature	First Support Processor
AMD SEV	1st Gen EPYC Processors
AMD SEV-ES	2nd Gen EPYC Processors
AMD SEV-SNP	3rd Gen EPYC Processors

C Evaluation Details

C.1 Benchmark Details

C.1.1 Phoronix Test Suite Memory Benchmarks. Table 11 summarizes the Memory Test Suite of Phoronix Test Suite [139].

Table 10. Feature comparison among AMD SEV, SEV-ES, and SEV-SNP. Note that at the time of writing (July 2024), (1) processors (and hardware) supporting SEV-TIO are not available yet, and (2) there is no publicly available concrete software implementation of SEV-SNP live migration.

Feature	**AMD SEV**	**AMD SEV-ES**	**AMD SEV-SNP**
VM Memory Encryption	✓	✓	✓
VM Register State Encryption	✗	✓	✓
Integrity Protection	✗	✗	✓
VMPL	✗	✗	✓
Live Migration	✓	✓	✓
SEV-TIO	✗	✗	Future processors will support

Table 11. The Memory Test Suite of Phoronix Test Suite

Benchmark	**Details**
MBW [58]	Measure memory bandwidth using memcpy. The test measures the memory copy time of 1024 MB data size by (1) directly calling memcpy for the entire data and (2) using a 256 KB block as a memory copy unit (noted as “Fixed”).
Tinymembench [66]	Measure memory bandwidth of sequential memory access. The test measures memcpy and memset time for 32MB data.
RamSpeed/SMP [63]	Measure memory bandwidth of (1) Copy (`dst=src`), (2) Scale (`dst=const*src`), (3) Add (`dst=src1+src2`), and (4) Triad (`dst=const*src1+src2`) operations. The test measures the performance of these operations for both integer (Int) and floating point (FP) types.
STREAM [135]	Measure the memory bandwidth of Copy, Scale, Add, and Triad operations.
Cachebench [137]	Measure bandwidth of read and write operations. The test performs multiple memory operations using a 64KB size buffer intended to measure cache bandwidth.

C.1.2 NAS Parallel Benchmarks (NPB).

Table 12 shows the NAS Parallel Benchmarks (NPB) [72] used in the evaluation.

Table 12. The NAS Parallel Benchmarks (NPB)

Benchmark	**Details**
IS	Integer Sort, random memory access
EP	Embarrassingly Parallel
CG	Conjugate Gradient, irregular memory access and communication
MG	Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive
FT	Discrete 3D fast Fourier Transform, all-to-all communication
BT	Block Tri-diagonal solver
SP	Scalar Penta-diagonal solver
LU	Lower-Upper Gauss-Seidel solver
UA	Unstructured Adaptive mesh, dynamic and irregular memory access

C.1.3 Unixbench.

Table 13 shows the Unixbench [68] benchmarks used in the evaluation.

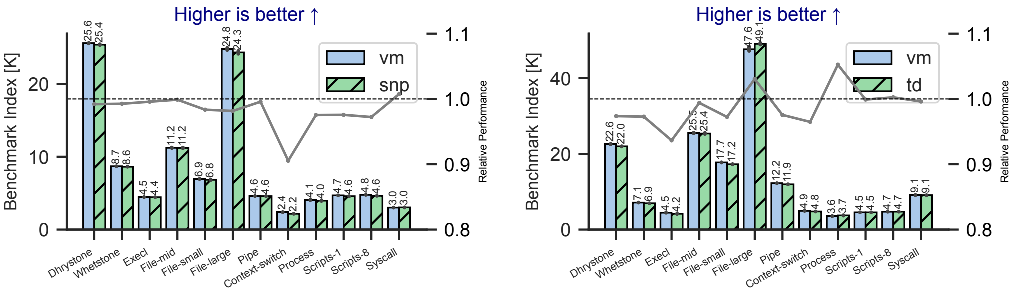
C.2 Additional Results

C.2.1 Application Benchmarks with Idle Polling.

Figure 15 and Figure 16 show the results of Unixbench and application benchmarks with idle polling (for both CVMs and VM). Notably, with the idle polling, we do not see more than 10% performance degradation in “Process” of Unixbench and TensorFlow (BERT) in the “X” VM.

Table 13. Unixbench

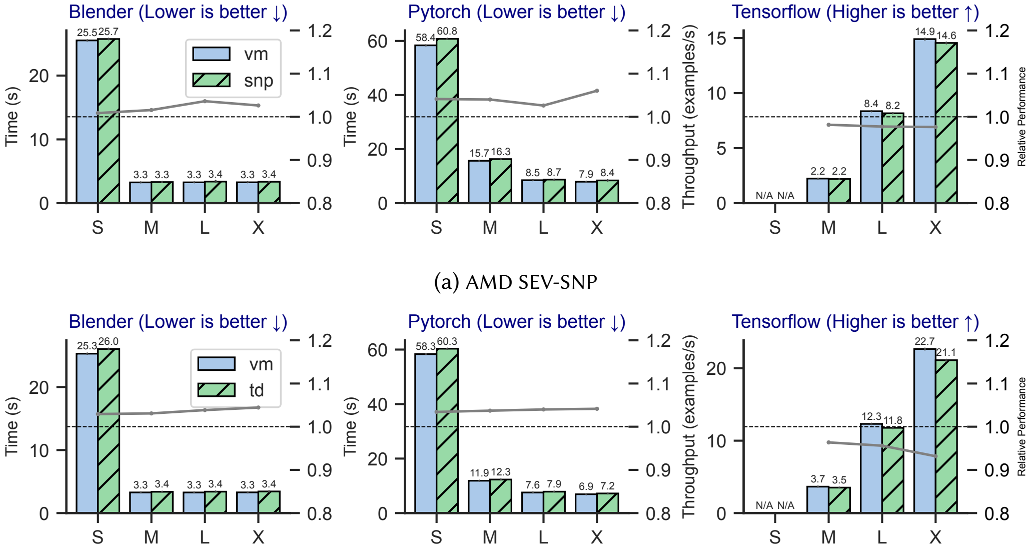
Benchmark	**Details**
Dhrystone	String handling without floating-point operations
Whetstone	A mix of operations typically performed in scientific applications
Execl	Count execl system call performed per second, which replaces process images
File-(s/m/l)	File copy performance with various buffer sizes (256B, 1KB, 4KB)
Pipe	Frequency of writing and reading 512 bytes through a pipe per second
Context Switching	Exchange rate of integers between two processes through a pipe
Process	Rate of process creation and reaping of child processes
Scripts(-1/8)	Measure how many times a shell script starts and reap (Scripts-8 has 8 concurrent copies)
System Call	Estimates the overhead cost of entering and exiting the operating system kernel



(a) Unixbench (AMD, 8vCPU, 64GB memory)

(b) Unixbench (Intel, 8vCPU, 64GB memory)

Fig. 15. Unixbench with idle polling.



(a) AMD SEV-SNP

(b) Intel TDX

Fig. 16. Application benchmarks with idle polling.

D Details of CVEs

Table 14 shows the full list of CVEs of AMD SEV(-ES/SNP) and Intel TDX up to June 2024 we summarize in the § 5.2.

Table 14. Full list of CVEs. Bug types: A: kernel, B: firmware, C: hardware, D: design issue. Attack types: H→G: Host to Guest, G→H: Guest to Host, G→G: Guest to Guest, H→G: Host to Host. "-" indicates no enough information is available. We classify CVEs as due to validation misses if the detail explicitly mentions validation.

CVE	CVM	Bug type	Attack type	Cause	Misc
CVE-2024-36936	TDX	A	H→G	Implementation	
CVE-2024-35939	TDX	A	H→G	Validation	
CVE-2024-25744	TDX	D	H→G	Design flaw	Hecler [175]
CVE-2023-52874	TDX	A	H→G	Validation	
CVE-2023-47855	TDX	B	H→G	Validation	
CVE-2023-45745	TDX	B	H→G	Validation	
CVE-2023-32666	TDX	B	H→G	-	
CVE-2023-22655	TDX	B	H→G	-	
CVE-2022-41804	TDX	B	H→G	-	
CVE-2024-25744	SNP	D	H→G	Design flaw	Heckler [175]
CVE-2024-25743	SNP	D	H→G	Design flaw	Heckler [175]
CVE-2024-25742	SNP	D	H→G	Design flaw	WeSee [174]
CVE-2023-52659	SNP	A	H→G	Implementation	
CVE-2023-4155	SNP	A	G→H	Validation	
CVE-2023-20573	SNP	D	H→G	Design flaw	Missing delivery of debug exceptions
CVE-2023-20566	SNP	B	H→G	Validation	
CVE-2023-20519	SNP	B	H→G	Implementation	
CVE-2022-23830	SNP	B	H→G	Implementation	
CVE-2022-23818	SNP	B	H→G	Validation	
CVE-2022-23813	SNP	B	H→G	Implementation	
CVE-2021-26409	SNP	B	H→G	Validation	
CVE-2021-26396	SNP	B	H→G	Validation	
CVE-2021-26349	SNP	B	H→G	Implementation	
CVE-2021-26328	SNP	B	H→G	Implementation	
CVE-2021-26327	SNP	B	H→G	Validation	
CVE-2021-26326	SNP	B	H→G	Validation	
CVE-2021-26325	SNP	B	H→G	Validation	
CVE-2021-26324	SNP	B	H→G	Implementation	
CVE-2021-26323	SNP	B	H→G	Validation	
CVE-2020-12966	SNP	C	H→G	Side-channel	Cipherleaks [128]
CVE-2024-26695	SNP	A	H→H	Implementation	
CVE-2023-46813	SNP	C	G→G	Race condition	
CVE-2023-31347	SNP	B	H→G	Implementation	
CVE-2023-31346	SNP	B	G→G	Implementation	
CVE-2023-20592	SNP	C	H→G	Implementation	CacheWarp [199]
CVE-2021-46744	SNP	C	H→G	Side-channel	Ciphertext side-channel [124]
CVE-2021-26321	SNP	B	H→G	Validation	
CVE-2021-26320	SNP	B	H→G	Validation	
CVE-2023-20575	SEV	C	H→G	Side-channel	Power side-channel
CVE-2022-0171	SEV	A	H→H	Implementation	
CVE-2021-47389	SEV	A	H→H	Implementation	
CVE-2021-47228	SEV	A	H→G	Implementation	
CVE-2021-47062	SEV	A	H→H	Implementation	
CVE-2021-47008	SEV	A	G→H	Implementation	

CVE-2021-46768	SEV	B	H→H	Validation	
CVE-2021-4093	SEV	A	G→H	Validation	
CVE-2021-26408	SEV	B	H→G	Validation	
CVE-2021-26406	SEV	B	H→G	Validation	
CVE-2021-26404	SEV	B	H→G	Validation	
CVE-2021-26403	SEV	B	H→G	Validation	
CVE-2021-26402	SEV	B	H→G	Validation	
CVE-2021-26342	SEV	C	H→G	Implementation	Improper TLB flush implementation
CVE-2021-26340	SEV	C	G→G	Implementation	Improper TLB flush implementation
CVE-2021-26332	SEV	B	H→G	Implementation	
CVE-2021-26311	SEV	D	H→G	Implementation	Lack of RMP
CVE-2020-36311	SEV	A	H→G	Implementation	
CVE-2020-12967	SEV	D	H→G	Implementation	Lack of RMP
CVE-2019-9836	SEV	B	H→G	Implementation	

Received August 2024; revised September 2024; accepted October 2024