DOI:10.1145/3624576

Article development led by acmqueue
queue.acm.org

Enabling confidential computing with AMD SEV-SNP technology.

BY DAVID KAPLAN

Hardware VM Isolation in the Cloud

WHEN YOU RENT an apartment, it is reasonable to have some expectation of privacy. The landlord provides you a place to stay in return for rent, and you do not expect the landlord to go snooping through your things or installing cameras in your bedroom. When you rent a virtual machine (VM) in the cloud, why should expectations be any different?

Public clouds provide an important service, offering compute capability in return for money. While cloud providers arguably do their best to keep your data isolated, especially from other tenants, there typically is not anything in hardware that keeps all your data isolated from the provider itself. Standard techniques, such as network encryption and encrypted hard disks, can protect data-in-transit and data-at-rest, respectively, but these do not protect data being actively processed (that is, data-in-use). A bug in the cloud provider's software, a malicious insider, or an errant crash dump could all expose your private data in ways that are often undetectable. These risks have kept many industries from moving to the cloud... until now, thanks to a new class of technology called "confidential computing."

Confidential computing[16] (CC) is a generic term for technology that is designed to protect workloads, such as VMs, from all other software, including the hypervisor (HV). It is a big change from the traditional trust model of CPUs. Historically, CPUs were built using a ring-based security model where higher-privileged rings had access to everything in the lower-privileged rings. This worked well for operating systems and many typical environments. With CC, this model is modified so that while administrator code, such as the HV, manages system resources and scheduling, that code no longer has complete access to the system. For example, the HV might choose whether to run one VM or another VM, and how much memory to assign each, but it cannot see or modify the contents of that memory.

CC is useful in environments where the owner of the workload differs from the owner of the physical machine. This may be because you are security-conscious or concerned about the risk of other tenants exploiting vulnerabilities in cloud infrastructure, or simply because of the type of data in the VM. Customer data, financial information, and health information, for example, are highly sensitive and often subject to tight regulatory or compliance controls. CC reduces the risk of that data leaking while in the cloud.

Overview of AMD SEV-SNP

AMD began offering CC support with the first generation of Secure Encrypted Virtualization (SEV)[12] technology in 2017. The SEV technology has evolved

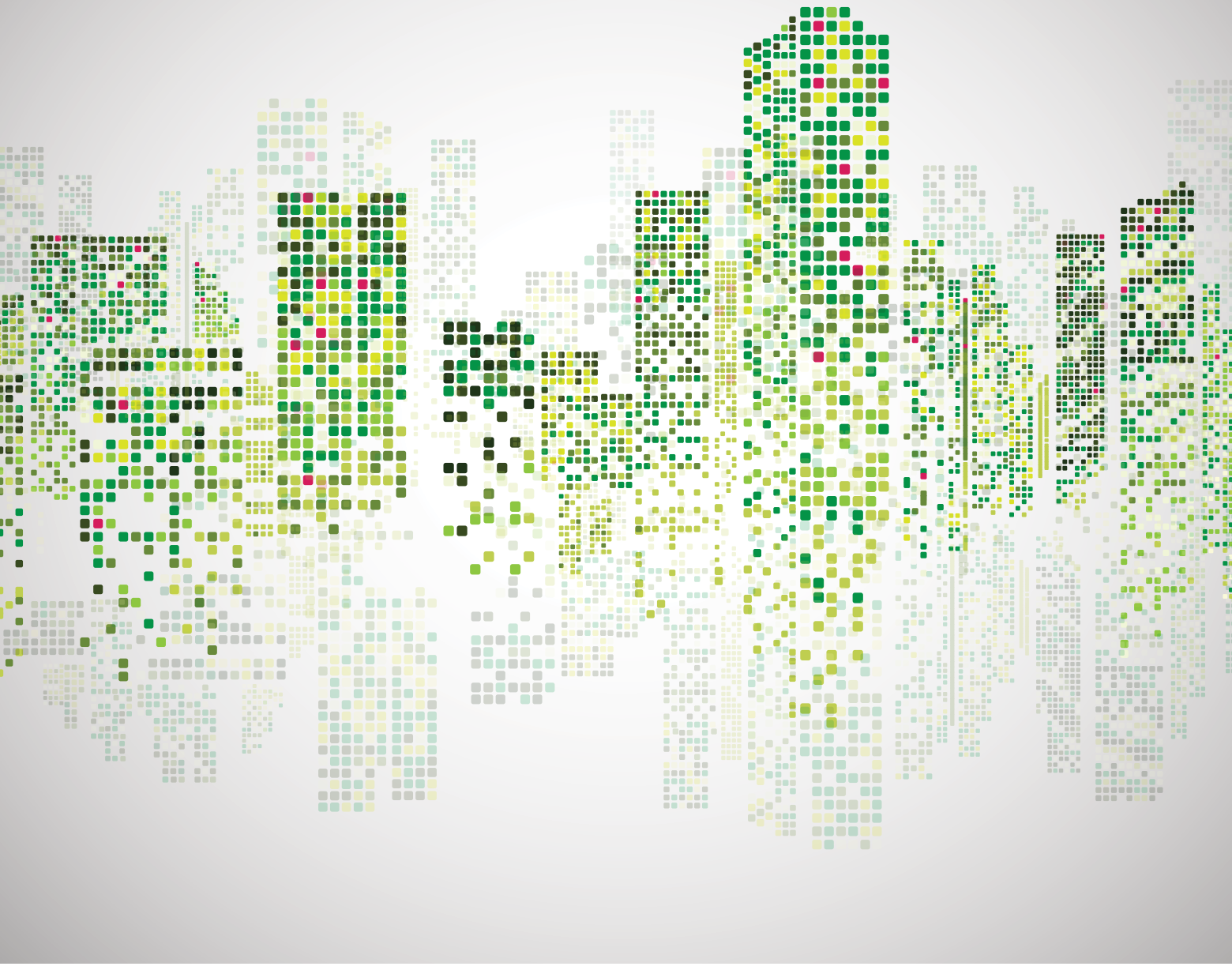


IMAGE FROM SHUTTERSTOCK.COM

over time to support more security and flexibility. This article focuses on the third generation of this technology, called SEV with Secure Nested Paging, or SEV-SNP. AMD SEV-SNP[1] is available in third generation and later AMD EPYC processors, which began shipping in early 2021.

SEV-SNP leverages existing AMD-V technology, which provides special hardware support and acceleration for virtualization. SEV-SNP is designed to protect a VM from all outside entities, including the bare-metal hypervisor, the basic input/output system (BIOS), other VMs, and even external I/O devices. While SEV-SNP requires special software support at the operating system and HV level, it does not require any changes to applications. A Web server that runs in a cloud VM today can be easily moved into an SEV-SNP VM without recompilation in what is often referred to as "lift-and-shift."

SEV-SNP basics. SEV-SNP supports several key CC principles, including the need for confidentiality, integrity, and attestation. The need for confidentiality should be obvious, as it is in the name "confidential computing." In practical terms, this means restricting access to the data inside an SEV-SNP VM. This is primarily accomplished using memory encryption, as shown in Figure 1. AMD EPYC processors include a high-bandwidth AES-128 (or AES-256 in fourth-generation AMD EPYC) encryption engine in each of the on-die memory controllers. This engine is designed to quickly encrypt/decrypt all data between the CPU and DRAM (dynamic random-access memory). Each SEV-SNP VM is assigned a unique random encryption key when the VM boots, and the use of that key is tightly controlled by hardware, ensuring plaintext is visible to only the correct VM.

While most data inside a VM typically should remain private, VMs do sometimes need to communicate with the HV or devices. In the SEV-SNP architecture, this is done using *shared memory pages*. A VM may choose which

memory pages are private or shared using a page-table bit (C-bit or encrypted-bit). Private pages (C-bit=1) are always encrypted with the VM-specific encryption key, while shared pages (C-bit=0) may be either unencrypted or encrypted using an HV key.

Another key principle of SEV-SNP is data integrity, which means ensuring that untrusted code, like the HV, may not modify guest memory in any way. Even though guest memory is encrypted, an attacker could attempt to corrupt it or perform a replay attack, using an old copy of ciphertext to replace a newer version so the guest appears to see an old version of data. The SEV-SNP architecture enforces data integrity on guest private pages using a structure called the reverse map table (RMP).

Acronym Gallery

- **AES:** Advanced Encryption Standard
- **ASID:** address space ID
- **ASP:** AMD security processor
- **DMA:** direct memory access
- **GPA:** guest physical address
- **GVA:** guest virtual address
- **HV:** hypervisor
- **IDE:** integrity and data encryption
- **IOMMU:** I/O memory management unit
- **KDS:** key distribution service
- **MMU:** memory management unit
- **OVMF:** open virtual machine firmware
- **RMP:** reverse map table
- **SEV:** secure encrypted virtualization
- **SEV-SNP:** SEV with secure nested paging
- **SEV-TIO:** SEV with trusted I/O
- **SmartNICs:** smart network interface cards
- **SPDM:** secure protocol and data model
- **SVSM:** secure VM service module
- **TDISP:** TEE Device Interface Security Protocol
- **TLB:** translation lookaside buffers
- **TPM:** trusted platform module
- **UEFI:** unified extensible firmware interface
- **VCEK:** versioned chip endorsement key
- **VM:** virtual machine
- **VMPL:** virtual machine privilege level
- **VMSA:** virtual machine save area
- **TPM:** virtual-TPM

Reverse Map Table

In an AMD system, memory is accessed through either CPU instructions or device I/O, known as direct memory access (DMA). CPU accesses use an internal memory management unit (MMU), while DMA uses an I/O memory management unit (IOMMU). Both the MMU and IOMMU translate virtual addresses into physical addresses using page tables before accessing memory. In the SEV-SNP architecture, the HV controls the page tables and mapping to addresses in the physical address space, but the RMP enforces access control in the physical address space.

The RMP is a large in-memory data structure that tracks the owner of each page of memory. Every 4KB page of DRAM is associated with a 16-byte RMP entry. That RMP entry indicates who is allowed to write to that page. At any given time, memory might be assigned to a specific guest or to the HV, or even reserved for hardware use. The RMP is never modified by software directly; it can be manipulated only by trusted means, such as special x86 CPU instructions.

The RMP is used to enforce access control to memory pages in the system. When the CPU or IOMMU translates a virtual address into a physical address, an RMP check is typically performed at the end of the translation. The physical address that the software or device wants to access is used to index into the RMP and then to verify that the assigned owner is the one trying to access the page. If this check fails, a fault is generated, and access is blocked.

For example, to add a page of memory to a guest, the HV would issue the RMPUPDATE x86 instruction. This instruction modifies the RMP so that a specific page of memory is now assigned to a specific guest at a specific guest physical address (GPA). Once this is done, the HV no longer can write that memory page. If it attempts to do so, the CPU will issue a page fault.

What makes the RMP a reverse map is that for guest memory, the RMP entry contains the GPA and address space ID (ASID) to which that page is assigned. This ensures a malicious HV cannot try to map a page into the wrong location in the guest address space.

When the CPU is running an SEV-SNP guest, after translating a guest virtual address (GVA) into a GPA and then into a system physical address using nested paging, the RMP entry is read and checked, as shown in Figure 2. The RMP entry must contain the correct GPA that was used during the page tablewalk, which verifies that the translation is correct.

Other Protection

Memory encryption and the RMP provide a lot of confidentiality and integrity protection for SEV-SNP VMs, but this is not the whole story. SEV-SNP is designed to protect VMs from a malicious HV and malicious devices, among others. Of course, cloud providers probably are not intending to be malicious, but the point of CC is to be protected in all cases. After all, benign software can become malicious via a simple remote code exploit.

One example of additional security to protect against potentially malicious hypervisors is the ability of SEV-SNP VMs to opt into various modes of interrupt security. Typically, interrupts that occur for things like emulated devices are injected into the guest VM by the HV, and this is perfectly normal. But what would happen if the HV maliciously injected an interrupt while a VM had interrupts masked? Certainly, that is not expected behavior, but "unexpected behavior" is what makes attackers lick their chops.

One interrupt security mode offered by SEV-SNP is "Restricted Injection." In this mode, the HV is prevented from injecting any interrupts or exceptions into the guest other than one specific vector called #HV. When a #HV is injected, the guest uses a special protocol[3] to find out what underlying interrupt or exception the hypervisor wanted to send. The guest may then decide to handle that event immediately or defer it until later. By doing so, the guest can effectively emulate its own interrupt controller, ensuring that interrupts are handled at the right time without relying on the HV.

Side-channel attacks such as Spectre are another potential threat that can break the isolation promises of technologies such as SEV-SNP. Conse-

quently, SEV-SNP offers special modes, for example, to isolate the branch predictor so VMs can be protected from certain types of side-channel attacks.

Of course, CC is not a panacea for all side-channel protection. Best practices such as constant-time crypto and avoiding secret-dependent branches are still required for the best level of security.

Attestation

Technologies such as SEV-SNP can isolate a VM, ensuring data confidentiality and integrity... but how do you know that the cloud has turned the feature on? And how can you know if the cloud provider has installed the most recent firmware or microcode fixes on its system? This is where attestation comes into play.

Attestation is a fancy word for proof—it is proof that you are who you say you are, or that something is in fact what it looks like. It involves a trusted someone certifying ("attesting") that someone else is secure.

In CC technologies such as SEV-SNP, the cloud provider is untrusted, so the only entity that can attest to the security of a VM is the silicon provider itself. In SEV-SNP, this service is provided by an on-chip entity called the AMD security processor (ASP).

The ASP is a dedicated microcontroller that is part of the processor but separate from the x86 CPU. It has its own resources, including fuses, SRAM, and crypto hardware. The ASP runs firmware that AMD signs and provides several services to help manage the life cycle of SEV-SNP VMs. These services are documented in a public specification[4] and are used to create/destroy SEV-SNP VMs, perform tasks such as memory swapping or live migration, and—of course—to provide attestation.

In SEV-SNP, the initial code and data associated with a VM is not secret. To create a secure VM, the HV provides the ASP with this initial code and data, which is typically a guest BIOS image such as open virtual machine firmware (OVMF). Typically, a secure VM uses an encrypted hard disk, so to boot successfully, it will need to unlock access to that disk. The key to this disk is likely on another key server somewhere. This is where attestation can come into play.

To request the disk decryption key, the new VM must prove itself to the key server. It must not only show that the VM is what it says it is, it must prove that it is running in a secure environment and that its image has not been tampered with in any way.

The ASP generates an attestation report when requested by an SEV-SNP guest VM, illustrated in Figure 3. This attestation report includes lots of information about the VM and the platform itself, such as the measurement (that is, hash) of the initial code and data of the VM, identity information

Figure 1. Memory encryption.

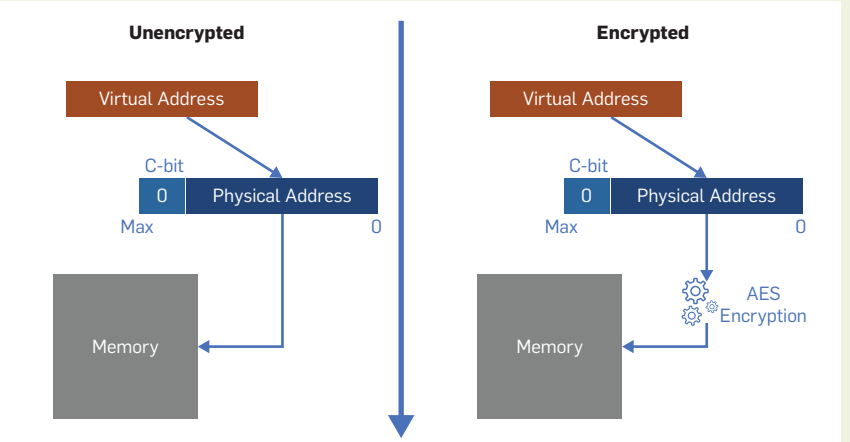


Figure 2. Using the reverse map table.

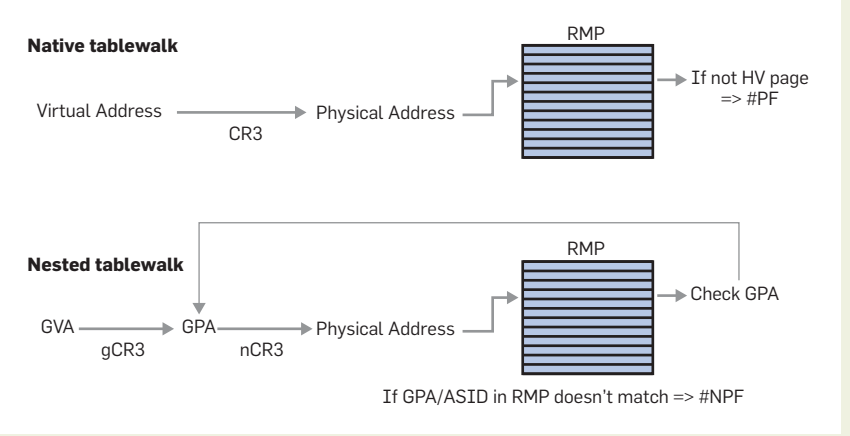
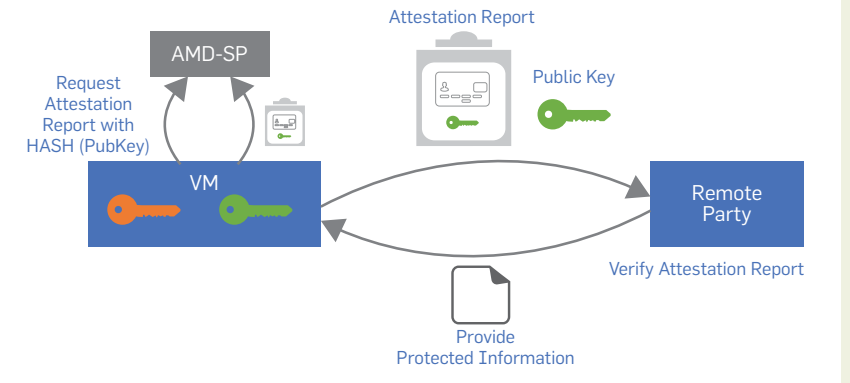


Figure 3. Attestation report cycle.



about who the VM is, version numbers associated with the currently loaded firmware, and so on.

The entire report is signed using a special key called a versioned chip endorsement key (VCEK). The VCEK is unique to each AMD CPU and is cryptographically derived using the current version of mutable ASP firmware and CPU microcode. This is built so someone looking at a report can be assured that this report was signed by a specific version of firmware, not an older version trying to pretend to be a newer one. This is especially important since when researchers find bugs in features such as SEV-SNP, AMD will often issue a firmware or CPU microcode update to address them.

How does this all fit together to talk to a key server to do a secure VM boot? One of the fields in the attestation report is a 512-bit field containing arbitrary guest-supplied data. A typical use case might involve a guest VM that creates a public/private key pair and provides a hash of its public key to be included in the attestation report. The verifying party can then use that public key to communicate securely with the guest if it passes all the required security checks in the attestation report.

Besides verifying the data in the attestation report, the verifying party must check the signature of the report. It should be signed with the appropriate VCEK—but how does the verifier know it can trust that VCEK? To assist with this, AMD created the Key Distribution Service (KDS), which provides certificates that can prove a VCEK is authentic and valid. The KDS is integrated into AMD's manufacturing flow so the AMD signs certificates only for authentic AMD parts. (For more details about this, see the VCEK Certificate and KDS Interface Specification.)[5] An example tool that can be used to assist with SEV-SNP attestation can be found on GitHub.[18]

Confidential computing is a security model that fits well with the public cloud.

Security Within a VM

So far, this article has focused on protecting a VM from the hypervisor and other host software. But VMs often need to enforce their own internal security, too. For example, on a bare-metal system unified extensible firmware interface (UEFI) secure boot may use a trusted platform module (TPM) chip to calculate measurements of the boot flow to ensure protection from things like rootkits. In a virtualized system, a TPM can be emulated by the HV. How can this work in a confidential environment where the HV cannot be trusted to emulate a TPM?

To help with such a task, SEV-SNP provides a capability called Virtual Machine Privilege Level (VMPL), which allows for additional security controls that can protect memory inside the guest from other code within that same guest. Each SEV-SNP guest may have up to four VMPLs, where VMPL0 is the most privileged and VMPL3 is the least. Each hardware context—called a Virtual Machine Save Area (VMSA)—is associated with a VMPL, and every memory page assigned to a guest may have different permissions based on VMPL. VMPL0 always has full access to every page in the guest address space, but it may configure some pages not to be accessible at, say, VMPL1, or perhaps allow read-only access.

One use case of VMPLs is to run what AMD calls a Secure VM Service Module (SVSM) at VMPL0: a small piece of code designed to provide security services to the rest of the guest. For example, COCONUT[8] is an SVSM written in Rust, which runs in VMPL0 while the rest of the Linux OS runs at VMPL1.

One thing an SVSM can do is provide a virtual-TPM (vTPM) service to a guest. The vTPM appears as a normal TPM to the OS in the guest and enables features such as UEFI secure boot. But because the vTPM runs at VMPL0, it can use memory that is protected from the guest OS so any malware in the guest cannot compromise vTPM state. And let's not forget that the entire guest is running under SEV-SNP, so it's also protected from the HV or other VMs trying to compromise its state.

Services such as vTPM are also useful since they can prove the security of a VM to users who connect to that VM. The vTPM can attest to the security of the software running in the VM, while the SEV-SNP attestation report attests to the security of the vTPM itself and the VM configuration.

AMD is working with the open-source community to further SVSM development and eventually support features such as accelerated live migration and potentially other services via the SVSM.

Finally, the SEV-SNP architecture

also supports the concept of a *paravisor*, which can assist with running a guest OS that knows almost nothing about SEV-SNP. This is a small but trusted layer that sits between a mostly unmodified legacy guest OS and the HV. The paravisor handles all the new security protocols associated with SEV-SNP, even when the guest OS does not really know anything about CC.

For example, Microsoft uses a custom paravisor in Azure that leverages SEV-SNP VMPL technology to support customers moving existing workloads into a confidential cloud with minimal effort.[14] This paravisor supports compatibility with both Windows and Linux and provides secure boot services, secure interrupt delivery, and more. Technologies like this expand the reach of CC by making security easier to adopt without requiring changes to customer software stacks.

Cost of Security

Security never comes for free, but CC technologies such as SEV-SNP try to minimize the overhead of security as much as possible. SEV-SNP uses high-bandwidth Advanced Encryption Standard (AES) memory-encryption engines, multilevel translation lookaside buffers (TLBs), and other microarchitectural optimizations to reduce the performance impact of the additional security checks required to maintain a CC environment.

As a result, the performance cost of SEV-SNP is minimal in many workloads. For example, third-generation AMD EPYC processors demonstrated a delta of only approximately 4% on estimated SPECrate 2017_int_base,[17] 2% on server-side Java performance benchmarks, and 2% on Black-Scholes benchmarks.[9] These costs are primarily a result of the impact of AES memory encryption and RMP security checks. Over time, AMD expects these performance deltas will likely be reduced further. Of course, actual performance of real-world workloads depends on many factors, including hardware configuration, placement of the VM and its resources, and software optimizations.

Securing Device I/O

With each generation of SEV, the technology has evolved to add more security features, capabilities, and improved performance. AMD recently announced the next major evolution for SEV with the new SEV with Trusted I/O (SEV-TIO) technology.[2] SEV-TIO provides support for direct device assignment to SEV-SNP VMs, allowing secure VMs and physical devices to communicate in a fast and secure manner.

SEV-TIO is built atop several industry standards, including PCIe's TEE Device Interface Security Protocol (TDISP),[15] Integrity and Data Encryption (IDE),[11] Secure Protocol and Data Model (SPDM),[10] and others. It provides a secure link between a device, such as a virtual function on a device, and a specific SEV-SNP VM.

This secure link means that communication between the VM and the device has both confidentiality and integrity. It provides a way for a device to attest to a guest VM, so guest VMs can avoid interacting with malicious devices. Most importantly, it enables high-performance DMA without bounce buffering.

Prior to AMD SEV-TIO, all device communication had to go through shared (unencrypted) memory pages. This meant that a VM would have to take the data it wanted to send to a device, copy it into a C-bit=0 page, and then ask the device to grab it from there. That extra memory copy can be expensive if you are doing a lot of I/O. With SEV-TIO, once the guest has verified the attestation of the device and decided it is trustworthy, it can allow the device to DMA directly to guest private (encrypted) memory; no more memory copies required.

Additionally, because any data in C-bit=0 pages is visible to the HV, any secrets the VM wants to send to a device must be encrypted in software before being sent over DMA. With SEV-TIO, this software encryption is no longer strictly required as the data being read and written by devices can target C-bit=1 pages.

SEV-TIO is designed to work with any device that is compliant with TDISP and related standards. This might include smart network interface cards (SmartNICs), which can offload network and storage work; AI training accelerators; and more. While not every workload may require the kind of I/O security and performance offered by direct device assignment, SEV-TIO expands the reach of CC to I/O-sensitive workloads.

Conclusion

Confidential computing is a security model that fits well with the public cloud. It enables customers to rent VMs while enjoying hardware-based isolation that ensures a cloud provider cannot purposefully or accidentally see or corrupt their data. SEV-SNP was the first commercially available x86 technology to offer VM isolation—including confidentiality and integrity—for the cloud and is deployed in Microsoft Azure,[7] AWS,[6] and Google Cloud.[19] As CC technologies such as SEV-SNP develop, CC is likely to simply become the default trust model for the cloud. C

References

1. AMD. AMD SEV-SNP: Strengthening VM isolation with integrity protection and more. *AMD White Paper*, 2020; https://bit.ly/46cydC5.
2. AMD. AMD SEV-TIO: trusted I/O for secure encrypted virtualization. *AMD White Paper*, 2023; https://bit.ly/3EzG1Sr.
3. AMD. SEV-ES guest hypervisor communication block standardization, 2023.
4. AMD. SEV secure nested paging firmware ABI specification, 2022; https://www.amd.com/system/files/TechDocs/56860.pdf.
5. AMD. Versioned chip endorsement key (VCEK) certificate and KDS interface specification, 2023; https://www.amd.com/system/files/TechDocs/57230.pdf.
6. AWS. Amazon EC2 now supports AMD SEV-SNP, 2023; https://go.aws/3ZlqqzG.
7. Cai, R. Azure confidential VMs using SEV-SNP (DCasv5/ECasv5) are now generally available. *Microsoft Azure Confidential Computing Blog*, 2022; https://bit.ly/45KHd1x.
8. COCONUT Secure VM Service Module; https://github.com/coconut-svsm/svsm.
9. Comp, L. Microsoft Azure confidential computing powered by 3rd gen EPYC CPUs. *AMD*, 2021; https://bit.ly/45M8q3S.
10. DMTF. All published versions of DSP0274, 2023; https://www.dmtf.org/dsp/DSP0274.
11. Harriman, D. Integrity and Data Encryption and IO security updates. *PCI-SIG*, 2022; https://pcisig.com/blog/integrity-and-data-encryption-ide-and-io-security-updates.
12. Kaplan, D., Powell, J., Woller, T. AMD memory encryption. *AMD White Paper*, 2021; https://bit.ly/45NI01C.
13. Linux SVSM; https://github.com/AMDESE/linux-svsm.
14. Perez-Vargas, C. Confidential VMs on Azure. *Microsoft Windows OS Platform Blog*, 2023; https://bit.ly/45LD9xH.
15. PCI-SIG. TEE Device Interface Security Protocol, 2022; https://pcisig.com/tee-device-interface-security-protocol-tdisp.
16. Russinovich, M., et al. Toward confidential cloud computing. *Commun. ACM 64*, 6 (June 2021), 54–61; https://dl.acm.org/doi/10.1145/3453930.
17. See https://www.spec.org for more information about SPEC® benchmarks.
18. VirTEE; https://github.com/virtee/.
19. Young, J. Oh SNP! VMs get even more confidential. *Google Cloud Blog*, 2023; https://bit.ly/3P7egW8.

David Kaplan is a senior fellow at AMD, Austin, TX, USA, where he is the lead architect for the AMD Secure Encrypted Virtualization technology as part of the AMD Product Security Office. Kaplan has more than 17 years of experience at AMD and has worked in confidential computing for the past 10 years.