Understanding Trust Relationships in Cloud-Based Confidential Computing

Anonymous Author(s)

***Abstract*—A major drawback of cloud computing used to be the lack of confidentiality and verifiability of computations, making it impossible to use public commercial clouds to work with sensitive code or data. With the availability of Trusted Execution Environments (TEEs) came the promise of enabling confidential computations in the cloud. A number of big Cloud Service Providers (CSP) now supports the deployment of Confidential Virtual Machines (CVMs) that can be attested remotely, supposedly guaranteeing verifiable isolation and integrity, and removing potentially compromised or malicious infrastructure from the system's Trusted Computing Base (TCB). In this paper, we investigate this claim and examine the CVM infrastructure provided by commercial CSPs regarding the attestability of the TEE hardware and the entire CVM software stack, and transparency regarding software provisioned by the CSP. We develop a hierarchy of attestation levels to explain our findings and trust limitations. For the services analysed, we observe that many attestation steps can only partially be verified by the CVM owner. Thus, running CVMs on these CSPs' infrastructure does not allow full TCB reduction through independently verifiable attestation but requires trust in the CSP to deploy secure software and to truthfully report attestation data. Complete protection from infrastructural threats is thus not provided.**

1. Introduction

Cloud computing is the backbone of digitalized societies, supporting today's cloud-native web service development paradigm. Increasingly, also industries that traditionally operated their own IT environments, e.g., telecommunication, manufacturing, and healthcare, are now moving to the cloud for better scalability, manageability, and cost reduction. However, this trend is accompanied by questions about the security and trustworthiness of *Cloud Service Providers (CSPs)* along with regulatory concerns about privacy, data protection, and data sovereignty: "moving to the cloud" involves moving large amounts of sensitive data, along with some of the responsibility to protect it, to a third party.

Different forms of *Confidential Computing (CC)* have surfaced to address such concerns, with *Trusted Execution Environments (TEEs)* [34] such as Intel SGX and TDX, AMD SEV-SNP, or ARM CCA promising to effectively take the cloud provider out of the *Trusted Computing Base (TCB)*. These TEEs provide secure compartments on cloud servers that can run applications while protecting data in use from infrastructural threats and enabling the "removal of even the cloud provider from the Trusted Computing Base" [12]. At the same time they provide *attestation* mechanisms to cryptographically verify that the unmodified application is indeed running in an authentic TEE. On paper, such a design allows for a drastic reduction of customers' dependency on cloud providers to adequately secure the cloud platform infrastructure – firmware, OS, and virtualization layers – and leaving only the hardware vendor that implements a TEE as a root of trust.

While early TEEs for server systems embedded *enclaves* into user processes, the current market trend are *Confidential Virtual Machines (CVMs)* that place an entire tenant VM into a TEE. While this approach increases the TCB within the TEE, it simplifies the deployment of software in TEEs and reduces system-call performance overheads [1] relative to enclaves, thus lowering the CC adoption hurdle for customers. Consequently, cloud providers such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP) have recently started marketing CVM solutions as an easy fix for the security, privacy, and trustworthiness challenges outlined above.

However, launching and attesting a CVM is a different beast than the enclave-based CC approach. A VM consists of layers of system software, including firmware, bootloader, kernel, and guest OS, as well as user space applications, with many components usually provided by the CSP. Given this large software stack running inside the CVM, the issue of trustworthiness comes again into focus. Effectively removing the infrastructure, i.e., the CSP, from the customer's TCB requires 1) an attestation infrastructure that allows to attest the authenticity of both the TEE hardware and *all* the software running inside it, and 2) transparency for software components provisioned by the CSP in the CVM, e.g., by supplying the underlying source code alongside a reproducible build process.

In this paper we examine CVM offerings on public clouds along these two axes in order to evaluate the level of trust in the CSP these CC solutions still require. In particular, we take a close look at the boot process and provided attestion mechanisms involved in the setup of AMD SEV-SNP CVMs on popular cloud providers. It turns out that there are shortcomings that prevent achieving confidential computing under infrastructural threats. In many cases this is due to missing attestation infrastructure or proprietary software components that need to be included in the CVM. We make the following contributions:

- We introduce a hierarchy of *attestation levels* with increasingly stronger guarantees for CVMs and showcase what could go wrong with partial attestation at the lower levels of this hierarchy;
- We highlight that, if some components cannot be verified by the CVM owner independently, some blind trust in the CSP is still required when deploying services in commercial clouds;
- We conduct a case study into AMD SEV-SNP as provided by popular commercial CSPs to assess how products meet our attestation levels, and we explain shortcomings in current offerings.

2. Background

2.1. VM-based TEEs

Confidential VMs are enabled by recent technologies such as AMD Secure Encrypted Virtualization (SEV)-Secure Nested Paging (SNP) [3], Intel Trust Domain Extensions (TDX) [20], and ARM Confidential Compute Architecture (CCA) [4]. While the inner workings of these mechanisms may differ, they all allow to essentially wrap an entire VM, including its firmware, OS, and applications, into a TEE and also attest its authenticity to a third party via a cryptographic protocol. This contrasts greatly with enclave-based TEEs, such as Intel SGX, which provide an isolated execution environment for a trusted partition of a larger untrusted user space application.

On the other hand, VM-based TEEs are mostly transparent to the VM running inside it except for an adaptation layer, either in the guest system software or a paravisor [11]. Hence, they can provide CC to larger applications deployed as VMs without changing application code. Moreover, unmodified CC can be provided to containers via micro-VM container runtimes such as Kata [32]. Thus, CVMs are considered to be more suitable for cloud-based deployments. Nevertheless, this ease-of-use comes with an increased TCB in the TEE. This not only increases the attack surface and risk for vulnerabilities, but also complicates attestation. As measuring the complete VM along with the launch of the TEE is impractical, current CVM architectures follow a staged attestation approach that first creates a root of trust within the VM firmware and then integrates into the VM boot process, leveraging measured/secure boot technologies.

2.2. Measured Boot and Attestation

Even without consideration for TEEs, booting a VM is a complex process that is composed of several stages. The first components to run in such a Linux VM are its firmware, e.g., Open Virtual Machine Firmware (OVMF) and bootloader, e.g., GRUB. In some cases, such as with Direct Linux Boot [31], it is possible to skip the bootloader altogether and boot a kernel directly from the firmware. Once booted, the kernel extracts and mounts the initial RAM disk, e.g., *initramfs*, and executes the contained *init* script. Its sole purpose is to mount the actual root filesystem for the VM containing the OS and user space applications which is provided as a block device mapped into VM memory. After mounting this filesystem, the script changes the root directory to it and launches the OS, which in turn may start user applications.

The integrity and authenticity of the boot process is paramount to ensure the trustworthiness of any system. Measured boot [35] is a common technique to produce evidence that a computer system has booted securely. While its implementation may differ between hardware architectures, measuring the boot process usually relies on a fixed and trusted initial firmware stage and a tamperproof Root of Trust for Measurement (RoTM), such as a Trusted Platform Module (TPM). The initial firmware stage may cryptographically measure itself and load subsequent boot code which, in turn, may continue the measurement process and extend the produced measurement results. The measurement process and storage of results, typically hashes, is handled by the RoTM, and the authenticity of results can be verified at a later stage by a third party. For TEEs, the RoTM is integrated with the TEE implementation, e.g., in microcode for Intel SGX, or in a secure co-processor for AMD SEV-SNP, called AMD Secure Processor (SP). When setting up a TEE instance, the untrusted system interact with the RoTM through secure interfaces to create trustworthy measurements of the instance. For CVMs, measured boot is established by measuring the guest firmware along with the TEE itself during setup. For subsequent measurements, a local RoTM needs to be included to the firmware, e.g., via a virtual TPM (vTPM). Measurement results can then be certified by the RoTM and sent to a remote verifier for verification.

3. Problem Statement

This work examines trust relations involved in the CVM deployment process. To avoid potential confusion, we first delineate what we mean by "trust" and "trustworthiness" in this context.

Cloud service customers expect CSPs to run their cloud platform securely, e.g., that best practices are followed to prevent compromise by outsiders, customer data is sufficiently secured from leakage to other tenants, and policies are in place to curtail insider threats. Depending on their specific requirements, a tenant needs to perform a risk analysis before deploying their software at a given cloud platform given the information they have about it. This information is often incomplete as most CSPs run proprietary software to operate their platforms and customers must inherently rely on the promises of a CSP's marketing department for operational aspects that are opaque to them, while also soft metrics such as reputation, track record, or popularity may factor into their decision.

Accepting any residual risk, the tenant can then be said to *trust* the cloud platform. However, after making the decision, this trust becomes mandatory as the customer simply has to rely on the CSP to hold up their end of the bargain. This trust is also perpetuated *blindly* until objective evidence surfaces that it is no longer justified.

We posit that the dual of this blind trust is *trustworthiness*, which allows for continuous risk assessment by a cloud tenant through evidence of platform security obtained at run time. Such evidence may come in the form of attestation reports for software deployed in TEEs together with the possibility to independently verify the security of the TCB. The more such run-time evidence is provided, the greater the trustworthiness of an execution platform becomes, and the less blind trust is required.

For the CVM case this means that the CSP not only has to provide the necessary infrastructure to attest the authenticity of all boot stages, but also provide transparency for the software components provisioned by the CSP to allow security analysis of the remaining TCB.

This work aims to examine this issue more closely and proposes a hierarchy of attestation levels for the CVM that allow to obtain increasing amounts of runtime evidence for the integrity and authenticity of a CVM, thus allowing to increase trustworthiness. We then demonstrate the utility of this metric in an analysis of popular public cloud

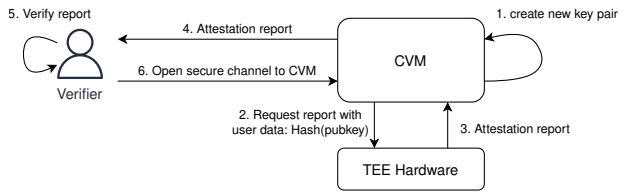

Figure 1: High-level attestation flow.

platforms offering AMD SEV-SNP functionality judging the maximum level of trustworthiness achievable by them.

4. System Model

In our system model, a guest owner deploys a CVM on an infrastructure where potentially other software is running at the same time, such as other VMs (confidential or not), monitoring software, etc. The infrastructure is managed by an infrastructure provider, and should provide the hardware and software primitives required to run the CVM in a TEE. Besides, a verifier performs attestation to make sure that the CVM was deployed and booted correctly; We refer to the IETF RATS architecture for a more comprehensive representation of the attestation infrastructure [8]. Furthermore, an end user interacts with the CVM to, e.g., deploy workloads and retrieve results. In general, some interaction between such actors is required and trust relationships between them may vary according to the use case. Besides, two or more logical actors can, in practice, be impersonated by the same party: For example, when running a CVM on a local machine, all actors may be played by the owner of that machine. When deploying a CVM in the cloud, however, all actors are likely unique. We will discuss this scenario in more detail in Sect. 5.

In this paper, we follow the standard threat model of TEEs: All software running outside of the CVM is potentially malicious, including the VMM/hypervisor, other VMs, etc. Note that this is true even when the guest owner and the infrastructure provider are the same party or trust each other, since the software running on the infrastructure might be compromised by remote attackers. Instead, the TEE manufacturer and any software and hardware components provided by them are assumed to be secure. The TEE threat model mainly focuses on confidentiality and integrity but leaves out availability. Additionally, side channels and physical attacks are also out of scope.

4.1. The Importance of Remote Attestation

When it comes to TEEs, remote attestation is an essential step to gain confidence in the deployed CVM, making sure that, after boot: 1) the CVM is running in a genuine and up-to-date TEE, and 2) the CVM is in an expected and good state, i.e., only trustworthy and measured software has been loaded. Moreover, attestation is often used to bind the identity of the CVM to a cryptographic credential, later used to establish a secure channel for provisioning secrets and deploying workloads, e.g., via SSH or TLS connections (Fig. 1). Usually, this is done by embedding some information about the credential in the attestation report, e.g., a hash of the public part of an ephemeral key.

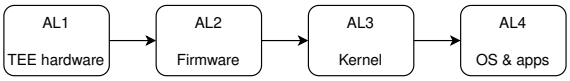

Figure 2: Hierarchy of attestation levels for CVMs.

Here, a nonce may also be used for freshness. This proves to the verifier that the key is *owned* by a CVM with that specific hardware and software configuration.

When attestation fails, this binding becomes compromised, and the verifier (or other entities) cannot be sure that they are communicating with a legitimate CVM. The problem is exacerbated by the complexity of CVM attestation that might lead to some components being accidentally skipped during verification. In the next sections, we describe all steps of CVM attestation and show what might happen when some of these steps are missed.

4.2. Attestation Levels

Attesting a CVM is not an easy task since we need to ensure that every part of the boot process is measured and can be verified. This section identifies five stages of attestation of a CVM, called *Attestation Levels (ALs)*. These levels are incremental, meaning that the guarantees provided at a certain level build upon the levels below. A visual representation of such levels is provided in Fig. 2.
AL0: No attestation. This is the baseline level where no attestation is done. As a result, the verifier cannot make any claims on the state of the CVM and the threat model is the same as for a regular VM execution environment.
AL1: Attested TEE isolation. In this level, the verifier has access to the "raw" attestation report signed by the TEE manufacturer, and is able to independently verify its integrity and authenticity. This typically involves the verification of the report's signature and certificate chain, which goes up to a root certificate that is self-signed by the TEE manufacturer. Moreover, the report also contains some information about the platform's TCB (e.g., CPU model, microcode version, etc.), and freshness information provided by the verifier. This allows the verifier to attest that the CVM is indeed running in a genuine and up-to-date TEE, which reduces the threat surface significantly as the host firmware, OS, and virtualization layer can now be excluded from the TCB. The verifier can also be sure now that software running outside of the CVM cannot directly access data stored inside of it. However, threats specific to the CVM technology used, e.g., side channel leakage, or to the software running inside the CVM remain.
AL2: Measured firmware. As mentioned above, remote attestation is used to establish a secure channel into the CVM, but AL1 does not give any information about what software is running at the other end of that channel. A first step to alleviate this situation is verify the *launch measurement* contained in the attestation report, which reflects the memory layout of the CVM at boot time including the firmware, page metadata and CPU register state. To obtain evidence for the correct deployment of the firmware, the verifier needs to validate the launch measurement against a trusted reference value. That measurement in itself does not tell anything about the code and data running in the CVM, hence it is important that the firmware is publicly available and can be reproducibly built. After a successful

verification, it can be ruled out that the trusted firmware has been replaced by a malicious one.

AL3: Measured Kernel. To rule out compromise of later boot stages, the verifier needs to validate the entire boot chain that starts from the firmware and goes up to the kernel, (optionally) through a bootloader. This also includes the initial RAM disk and kernel command-line parameters. By default, these components are not measured by the TEE and thus not part of the launch measurement. However, there are several ways to make this attestation possible, some of which are discussed in Sect. 4.3.

AL4: Fully measured boot. The verification of the measurements done at AL3 stops at *early userspace*, i.e., right before the root filesystem is mounted to the CVM. At the very least, the root filesystem of the CVM containing OS and application data should be integrity protected, to prevent loading a malicious version. In case it contains secrets, it should also be encrypted. The challenge in this step is to securely provision keys to the CVM only after a successful AL3 attestation, to prevent leaking the keys to a compromised CVM. Again, possible implementation strategies are provided in Sect. 4.3.

After a successful AL4 attestation, the verifier has confidence that the desired system and application software has been deployed correctly in the CVM. At this point, the threat model is similar to running the guest system on a trusted dedicated server and run-time security of the CVM is under the guest owner's purview. In particular, the deployed software may employ security policies to sanitize untrusted inputs or prevent the loading of additional, untrusted software. Moreover, further attestation steps may be initiated by the guest owner, e.g., run-time attestation of application control-flow integrity, or attesting the authenticity of attached secure I/O devices.

4.3. Attestation Strategies

Typically, support for AL1 and AL2 attestation is provided by all TEE manufacturers, via a certificate chain and a public key infrastructure (for AL1) and a a measurement of the initial memory layout of the CVM (for AL2). In both cases, it is often sufficient to verify the attestation report that can be fetched by the CVM and sent to the verifier. To reach ALs 3 and 4, instead, additional steps are needed. Below, we discuss possible approaches.

AL3. A simple strategy to make this attestation possible leverages Direct Linux Boot to bind kernel measurements to the firmware, in such a way that the launch measurement in the attestation report also reflects the identity of kernel components [14]. This can be done by adding kernel measurements as firmware variables, which will be loaded into secure memory at boot and measured by the TEE. Then, the firmware will load and pass control to the kernel only if its measurements match with the expected ones stored as variables, aborting otherwise. A more involved solution involves relying on a vTPM for measuring the CVM boot chain. On AMD SEV-SNP, this functionality can be exposed thanks to Virtual Machine Privilege Levels (VMPLs) [13], where the vTPM executes in a higher privilege level than the rest of the system. To bind vTPM quotes to TEE attestation reports, the digest of the vTPM endorsement key is added as user data in the TEE report [30]. The verifier, then, has to check both the

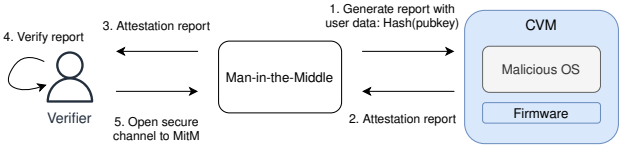

Figure 3: Possible Man-in-the-Middle attack with an unverified guest OS.

report and the quote to verify the trustworthiness of the CVM. On Intel TDX, instead, a vTPM can be exposed from a separate trust domain [19]. Additionally, Intel TDX also provides four Runtime Measurement Registers (RTMRs) whose value is reflected in the attestation report [21]. These registers are analogous to a TPM's PCRs, and can be similarly leveraged by a TDX-aware firmware to measure boot components.

AL4. To ensure the integrity of the root filesystem, a common approach is to extend the vTPM measurements up to the user space with Linux Integrity Measurements Architecture (IMA) [33]. IMA is a kernel subsystem responsible for measuring all binaries that are loaded at runtime, and it supports remote attestation. Measurements are stored in a measurement file which itself is integrity-protected by a measurement stored in the TPM, to prevent tampering from a privileged adversary. When adopting IMA, it is important to be aware that runtime measurements may be susceptible to time-of-check-time-of-use attacks [9], and that there exist so-called *measurement gaps*, i.e., some components are not measured.

Another approach relies on protecting the integrity of the whole filesystem at rest. To this extent, a popular software solution is `dm-verity` [25], which provides integrity protection by verifying the data blocks in a filesystem against pre-computed hash values, stored as a Merkle tree on a separate disk. The root of the tree (called root hash), guarantees integrity of the whole filesystem, and must protected from tampering. In our attestation flow, the root hash can be provided as a kernel parameter such that its integrity can be verified with a AL3 attestation. Filesystems using `dm-verity` are mounted as read-only; To enable both read and write, `dm-integrity` [24] can be leveraged instead to (re-)compute HMAC tags for each sector of the block device. In this case, however, a key should be separately provisioned to the kernel during early userspace. In our CVM scenario, this means that attestation should be performed during boot. Finally, `dm-integrity` can be combined with `dm-crypt` [23] to additionally provide encryption.

4.4. Partial Attestation

It should now be clear that attesting a CVM is not a trivial task. Unlike process-based TEEs such as Intel SGX, verifying the TEE attestation report alone is necessary but *not sufficient* to cover the whole boot process of the confidential workload. Failing to properly verify one or more boot stages might cause some attacks to go undetected, such as injecting malicious code or backdoors that would compromise the integrity and confidentiality of the CVM. These kind of attacks can be either local (e.g., a compromised hypervisor tampering with the root

filesystem) or remote (e.g., a malicious kernel image downloaded from an untrusted registry). Therefore, partial attestations might give a false sense of security to guest owners and end users, who expect that their code and data is protected when, in reality, this might not be true.

To showcase the severity of partial attestations, we consider the scenario where the verifier only attests the CVM up to AL2. Here, a Man-in-the-Middle (MitM) attack is possible if the CVM loads a malicious OS (Fig. 3): As the CVM can produce attestation reports with arbitrary user data, a MitM that controls the CVM OS can bind the report with a credential owned by the MitM (step 1). The verifier, when receiving the attestation report (step 3), can successfully verify that the report comes from a genuine CVM with the expected firmware (step 4), but cannot make any claims on the running OS. Failing to recognize this issue might cause the verifier to bind the CVM identity to the credential owned by the MitM, leading to the verifier possibly opening a channel with the MitM and leak secrets (step 5). What is worse, the MitM does not even need to run in a CVM, meaning that all data will be processed in clear. Therefore, although the verifier has performed a successful AL2 attestation, in practice the security guarantees obtained are basically the same as without attestation.

5. CVM Attestation in Public Clouds

Above, we have discussed the different attestation levels a verifier can achieve when evaluating the authenticity and integrity of a deployed CVM. However, who exactly plays the role of verifier depends greatly on the application scenario and the trust relations between the involved parties. In this work, we focus on a scenario where a tenant wants to deploy a CVM as guest owner on a cloud platform, acting as the verifier in the attestation process. Afterwards, attestation results may also be exposed to end users as well [15]. While different tenants may have different trust relations with a given CSP and lower attestation levels may suffice for them in practice, we assume here that the tenant wants to reduce the required trust in the infrastructure provider as much as possible and achieve a high level of trustworthiness for the CVM solution via attestation.

Here, the measurement reports used during attestation should ideally be verifiable independently of the CSP. In particular, attestation reports produced by the TEE hardware need to be available to the verifier. In addition, the verifier needs to be able to identify trusted software deployed in the TEE, i.e., they need to know the corresponding reference measurements for the firmware, kernel, user applications, etc. A CSP may provide these reference values so that the authenticity of the deployed software can be established. However, this results in a notion of verifiability that still relies on trust in the CSP and in the security of the deployed software. In principle, a trusted third party such as Intel Trust Authority [22] could certify the security of that software and publish the corresponding reference values, but that just shifts the required trust to a different entity.

A stronger *verifiability without trust* entails that the verifier can inspect the source code of software deployed in the TEE and obtain the reference measurements via a reproducible build process. Having access to the source code allows the verifier to conduct their own security analysis of the code and judge its trustworthiness. Of course, if the code is publicly available, this review can also be performed by the open source community, but this would again introduce required trust into the picture.

Overall, for our cloud CVM scenario, the tenant wants to achieve verifiability without trust for as high an attestation level as possible. Even if a high *nominal* AL can be achieved with the help of the CSP, still being forced to trust the CSP blindly for the measurement values and deployed software reduces the trustworthiness. We introduce the term *trustworthy AL* to denote the maximum, effective attestation level the tenant can achieve as a verifier without trust into the public cloud provider.

	AWS	Azure	GCP
TEE	●	◐	●
Firmware	●	◐	◐
Kernel	◐	◐	◐
Root FS	◐	◐	◐
Nominal AL	**4**	**4**	**4**
Trustworthy AL	**2**	**0**	**1**

● = Verifiable w/o trust; ◐ = Verifiable w/ trust; ○ = Not verifiable

TABLE 1: AMD SEV-SNP offerings on AWS, Azure and GCP. This table reflects attestation levels described in Sect. 4.2 and verifiability with or without trust, resulting in the maximum nominal AL (with trust in the CSP) and trustworthy AL (without trust). Results dated 2024-03-26.

6. Exploring the Cloud Landscape

We investigated current CC offerings on public clouds, focusing on SEV-SNP as the most widely available VM-based TEE. Our evaluation is solely based on commercial cloud features under general availability or public preview. We found that only the three major CSPs, i.e., Microsoft Azure, GCP and AWS, offer to deploy SEV-SNP CVMs on demand. We looked for VM-based confidential computing support in other CSPs, but they either supported plain SEV or only offered single-tenant bare-metal servers with SEV-SNP hardware. While plain SEV does not provide integrity protection nor flexible attestation capabilities, bare-metal servers come at much higher costs and typically require a fixed monthly subscription. Instead, we were interested in the multi-tenant scenario where customers can deploy CVMs on demand on a shared infrastructure, with little to no control over the hypervisor.

As discussed in Sect. 5, our evaluation tried to determine the level of *verifiability without trust* that can be achieved on such CSPs. Besides, we also considered the model where the verifier (partially) trusts the CSP and relies on provided services and data, such as a custom attestation report or a proprietary attestation service.

Results are shown in Tab. 1 and based both on public documentation and direct experiments. Our evaluation was made on 2024-03-26 and some of our findings may not hold in the future. However, our evaluation framework is still useful to evaluate future offers in the CVM area.

6.1. Results

In our experiments on AWS and GCP, we were able to fetch a raw attestation report from the AMD SP via the `/dev/sev-guest` device, allowing to independently verify that the guest VM is indeed running in a up-to-date SEV-SNP TEE. Azure CVMs, instead, only provide access to an attestation report generated at boot time and stored in vTPM non-volatile memory. As this report does not contain any freshness information chosen by the verifier, the latter cannot distinguish a legitimate attestation report from a replay attack and must rely on the CSP to attest that the VM is indeed using SEV-SNP. We tried to boot a SNP-aware kernel built from the AMDSEV repository [2], yet we still could not access the AMD SP.

Regarding AL2, only AWS offers an open source firmware that can be reproducibly built [7]. We successfully checked that the attestation report indeed reflects the correct measurement. Instead, Azure and GCP guests boot a proprietary firmware whose code cannot be audited. To the best of our knowledge, firmware reference measurements are also not provided, meaning that the verifier cannot even compare the measurement in the attestation report against the expected value. However, we observed that both Azure and GCP have a secure boot option that prevents CVMs from booting if the firmware is corrupted [26], [18], though we could obviously not experimentally verify this claim. Yet, tenants that are willing to trust their CSP can be confident that a running firmware was not tampered with by an outsider attacker.

All CSPs allow extending the measurements up to the kernel via a vTPM. However, on AWS [6] and GCP [16] the vTPM is implemented in the hypervisor, greatly increasing the TCB. Azure, instead, exposes a vTPM from within the CVM firmware [29] which, as mentioned before, cannot be reviewed. Thus, no CSP currently supports trustworthy attestation of kernel components.

After kernel verification, the chain of measurements can be extended to the root filesystem via different approaches (Sect. 4.3). Hence, AL4 is a natural extension of AL3, provided that the CSP allows for the customization or upload of VM images by the guest owner. We experimentally verified that the former is indeed possible by installing a custom bootloader and kernel, as well as changing kernel command-line parameters, e.g., to enable Linux IMA. Uploading images seems also feasible according to public documentation [5], [28], [17], but we did not verify this. Still, for trustworthy attestation, the integrity of the root filesystem ultimately relies on the integrity of the boot chain up to the kernel: If the latter cannot be verified independently of the CSP, neither can the former.

In summary, all CSPs provide the infrastructure to perform a full attestation (*nominal AL4*), but they rely on either proprietary software or components outside the TEE TCB, or both. On Azure, a verifier can obtain an attestation report but cannot verify its freshness, hence replay attacks are possible and the CSP must be trusted for all attestation levels (*trustworthy AL0*). On GCP, one can fetch dynamic reports, but the firmware cannot be independently verified (*trustworthy AL1*). Only AWS allows to attest an auditable firmware, yet extending the chain of measurements to later boot stages uses a hypervisor-based vTPM outside the TEE (*trustworthy AL2*).

6.2. Discussion

We showed that guest owners still require a significant amount of trust in the CSPs when using CVM solutions in the cloud. This goes against the classic TEE threat model, where the infrastructure is considered untrusted. There are two reasons for this: 1) CSP code is running inside the TEE boundary, with no possibility for review by the remote verifier, and 2) the attestation process involves CSP-managed components. While running CSP code in the CVM is sometimes necessary (e.g., to configure network interfaces, mount devices, etc.), it is certainly possible to increase transparency towards the verifier, as demonstrated by AWS' reviewable firmware. The second problem is mostly caused by a lack of customization options offered by CSPs. This is somewhat understandable since most cloud customers are looking for ease of use and "one-click compliance", where they are willing to trust the CSP to perform security checks in their place. For advanced users, however, more customizability is desirable.

Our evaluation did not focus on userspace software that runs after booting the root filesystem, such as `cloud-init` [10] or CSP-managed software like Azure Linux VM Agent [27]. These agents take care of VM configuration and can execute privileged commands, like installing packages or updating the trusted SSH identities. Any such operation, if compromised, can thwart the security guarantees of CVMs. Hence, it is equally important to implement hardening measures in the root filesystem to, e.g., check `cloud-init` configuration files before executing them or disable unnecessary software.

Finally, we note that CC is still a relatively new technology and support for recent TEEs such as SEV-SNP in the cloud has not yet reached full maturity. On the evaluated CSPs, CVMs are currently only available in select regions and, on GCP, support is still in public preview. Despite existing limitations, we see many efforts from CSPs to improve their offering, driven in part by the open source community and standardization bodies.

7. Conclusions

Being marketed as a solution to "remove the CSP from the TCB", which has caught the attention of customers who need an additional layer of protection for intellectual property, sensitive data, or for regulatory compliance in data protection, confidential computing is becoming a mainstream technology for cloud-based services. However, our research shows that, today, the promised security guarantees are far from reality as cloud providers still play a crucial role in the deployment and management of confidential workloads, especially regarding the latest trend with confidential virtual machines. Yet, customers that are willing to trust the CSPs can still benefit from current offerings as a defense-in-depth mechanism for strong hardware-rooted isolation from other tenants and the hypervisor. We also notice a joint effort between cloud providers and organizations such as the Confidential Computing Consortium and the IETF, as well as the open source community, to enhance current confidential computing solutions. Here, researchers can play an important role and nudge actors towards more principled security.

References

[1] A. Akram, A. Giannakou, V. Akella, J. Lowe-Power, and S. Peisert, "Performance Analysis of Scientific Computing Workloads on General Purpose TEEs," in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2021, pp. 1066–1076.

[2] AMD, "AMD Secure Encrypted Virtualization," https://github.com/AMDESE/AMDSEV/tree/snp-latest, Accessed on March 23, 2024.

[3] ——, "AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More," White Paper, January 2020.

[4] ARM, "Learn the architecture - Introducing Arm Confidential Compute Architecture," Specification, June 2023, version 3.0.

[5] AWS, "Importing a VM as an image using VM Import/-Export," https://docs.aws.amazon.com/vm-import/latest/userguide/vmimport-image-import.html, Accessed on March 26, 2024.

[6] ——, "NitroTPM on AWS to measure kernel," https://github.com/aws/uefi/issues/13, Accessed on March 26, 2024.

[7] ——, "UEFI," https://github.com/aws/uefi, Accessed on March 23, 2024.

[8] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, "Remote ATtestation procedureS (RATS) Architecture," Internet Engineering Task Force (IETF), RFC 9334, 01 2023.

[9] F. Bohling, T. Mueller, M. Eckel, and J. Lindemann, "Subverting Linux' integrity measurement architecture," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ser. ARES '20. Association for Computing Machinery, 2020.

[10] Canonical, "cloud-init," https://cloud-init.io/, Accessed on March 26, 2024.

[11] COCONUT, "COCONUT-SVSM Github Repository," https://github.com/coconut-svsm/svsm, Accessed on March 26, 2024.

[12] C. C. Consortium, "Confidential Computing: Hardware-Based Trusted Execution for Applications and Data," 11 2022, version 1.3.

[13] A. M. Devices, "SEV Secure Nested Paging Firmware ABI Specification," Specification, September 2023, version 1.55.

[14] Dov Murik and Hubertus Franke, "Securing Linux VM boot with AMD SEV measurement," https://static.sched.com/hosted_files/kvmforum2021/ed/securing-linux-vm-boot-with-amd-sev-measurement.pdf, Accessed on March 26, 2024.

[15] A. Galanou, K. Bindlish, L. Preibsch, Y.-A. Pignolet, C. Fetzer, and R. Kapitza, "Trustworthy confidential virtual machines for the masses," in *Proceedings of the 24th International Middleware Conference*, ser. Middleware '23. Association for Computing Machinery, 2023, p. 316–328.

[16] Google Cloud, "Confidential VM attestation," https://cloud.google.com/confidential-computing/confidential-vm/docs/attestation, Accessed on March 26, 2024.

[17] ——, "Create custom Confidential VM images," https://cloud.google.com/confidential-computing/confidential-vm/docs/create-custom-confidential-vm-images, Accessed on March 26, 2024.

[18] ——, "What is Shielded VM? - Secure Boot," https://cloud.google.com/compute/shielded-vm/docs/shielded-vm#vtpm, Accessed on March 26, 2024.

[19] Intel, "Virtual TPM in Intel TDX," https://github.com/intel/vtpm-td, Accessed on March 26, 2024.

[20] ——, "Intel Trust Domain Extensions," White Paper, February 2023.

[21] ——, "Intel Trust Domain Extensions (Intel TDX) Module Base Architecture Specification," Specification, November 2023, version 348549-003US.

[22] Intel, "Put Zero Trust Within Reach and Get Public Cloud Flexibility with Private Cloud Security," White Paper, 09 2023, version 355888-001US.

[23] kernel.org, "dm-crypt," https://www.kernel.org/doc/html/latest/admin-guide/device-mapper/dm-crypt.html, Accessed on March 26, 2024.

[24] ——, "dm-integrity," https://docs.kernel.org/admin-guide/device-mapper/dm-integrity.html, Accessed on March 26, 2024.

[25] ——, "dm-verity," https://docs.kernel.org/admin-guide/device-mapper/verity.html, Accessed on March 26, 2024.

[26] Microsoft Azure, "About Azure confidential VMs - Attestation and TPM," https://learn.microsoft.com/en-us/azure/confidential-computing/confidential-vm-overview#attestation-and-tpm, Accessed on March 26, 2024.

[27] ——, "Azure Linux VM Agent overview," https://learn.microsoft.com/en-us/azure/virtual-machines/extensions/agent-linux, Accessed on March 26, 2024.

[28] ——, "How to create a custom image for Azure confidential VMs," https://learn.microsoft.com/en-us/azure/confidential-computing/how-to-create-custom-image-confidential-vm, Accessed on March 26, 2024.

[29] ——, "Virtual TPMs in Azure confidential VMs," https://learn.microsoft.com/en-us/azure/confidential-computing/virtual-tpms-in-azure-confidential-vm, Accessed on March 26, 2024.

[30] V. Narayanan, C. Carvalho, A. Ruocco, G. Almasi, J. Bottomley, M. Ye, T. Feldman-Fitzthum, D. Buono, H. Franke, and A. Burtsev, "Remote attestation of confidential VMs using ephemeral vTPMs," in *Proceedings of the 39th Annual Computer Security Applications Conference*, ser. ACSAC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 732–743.

[31] QEMU, "Direct Linux Boot," https://qemu-project.gitlab.io/qemu/system/linuxboot.html, Accessed on March 26, 2024.

[32] A. Randazzo and I. Tinnirello, "Kata Containers: An Emerging Architecture for Enabling MEC Services in Fast and Secure Way," in *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, 2019, pp. 209–214.

[33] R. Sailer, X. Zhang, T. Jaeger, and L. Van Doorn, "Design and implementation of a TCG-based integrity measurement architecture," in *USENIX Security symposium*, vol. 13, no. 2004, 2004, pp. 223–238.

[34] M. Schneider, R. J. Masti, S. Shinde, S. Capkun, and R. Perez, "Sok: Hardware-supported trusted execution environments," *arXiv preprint arXiv:2205.12742*, 2022.

[35] Trusted Computing Group, "TCG EFI Platform Specification," Specification, January 2014, version 1.22.