Zero-Trust vTPM for Confidential VMs

Claudio Carvalho
ccclaudio@ibm.com

KVM Forum – June 14th, 2023
Outline

• Introduction
• Our goal
• TPM-based attestation
• Zero-trust vTPM
  • Where should we run it?
  • vTPM authenticity
  • vTPM ephemeral state
  • Architecture
  • Demo
• Conclusions
• Questions
Introduction

• Protecting sensitive data end-to-end
  • At rest, in-transit and in-use

• Trusted Execution Environment (TEE)

• Confidential Virtual Machines
  • AMD SEV-SNP, Intel TDX, ARM CCA, etc
  • Protect the VM data in-use from unauthorized access
    • E.g. hypervisor / CSPs
  • Various use cases e.g. Cloud, IoT and multi-party computation
Introduction

• CVM consumption model:
  • Attest the confidential VM first

• How do we attest a CVM?
  • Boot and runtime measurements + attestation
  • VM launch measurement covers only the initial VM state
  • Solutions proposed are not standard and/or rely on untrusted entities

• Trusted Platform Module
  • Industry standard for attestation
  • Existing TPM tooling could be reused
  • Allow use of advanced attestation techniques e.g. IMA
Our Goal

• Develop a zero-trust vTPM that allow TPM-based attestation for Confidential VMs

• Target platform: AMD SEV-SNP virtual machines
TPM-based Remote Attestation

Platform Configuration Registers (PCR)

PCR Extend operation
PCRnew := HashAlgo(PCRcurrent || digest)
Zero-trust vTPM for attestation

1. Where should we run the vTPM?
2. How do we prove the vTPM is authentic?
3. vTPM state injection/ejection
Where should we run the vTPM?

- vTPM does not run in a physical TPM chip
  - MS TPM 2.0 Ref. implementation
  - Must protect the vTPM data from unauthorized access e.g. host OS and guest OS

1. In the CVM host
2. In the actual CVM
3. In a separate CVM
Where should we run the vTPM?

- We run the vTPM in the Secure VM Service Module

- Why do we run it in the SVSM?
  - First module that runs in SEV-SNP guests
  - Provide runtime privileged services to the guest
    - Leverage VM Privilege Level (VMPL) for address space isolation

- vTPM
  - Enlightened TPM drivers for SVSM
    - https://lore.kernel.org/linux-coco/acb06bc7f329dfee21afa1b2ff080fe29b799021.camel@linux.ibm.com/
    - [NEW] vTPM protocol – SVSM spec draft v0.62 - linux-coco mailing list

- SVSM implementations
  - https://github.com/AMDESE/linux-svsm
  - https://github.com/coconut-svsm/svsm
SVSM-vTPM Authenticity

- Physical TPM
  - Endorsement Key (EK) - TPM identity
    - EKpriv never leaves the TPM
  - EKcert: Certificate of authenticity for the EK
    - Processes used for creating and protecting the key meets the necessary security criteria (TPM Arch spec §9.5.2)
  - TPM manufacturer ships the TPM with an EK and EKcert

- SVSM-vTPM
  - MS TPM 2.0 Ref
    - Entropy source: rand() calls the rdrand assembly instruction
    - Openssl 1.1.1q
  - Create EK (TPM2_CC_CREATEPRIMARY)
  - Call the AMD-Secure Processor(SP) to issue an certificate of authenticity for the EK
    - SNP_ATTESTATION_REPORT: AMD-SP signs the attestation report with a key that chains back to AMD root key
    - Bind EK to AMD: Provide a SHA512_hash(EKpub) in the report data
    - VMPL0 Attestation report can be requested only by the SVSM (VMPCK0 key access)
    - Hypervisor cannot decrypt SNP_ATTESTATION_REPORT requests (SEV-SNP ABI spec)
    - Authentic SVSM-vTPM

**Diagram:**
- AMD SEV-SNP Linux Guest Boot
  - Guest Userspace
    - Guest Kernel
      - vTPM Driver
        - VMPL1
        - OVMF
          - vTPM Driver
            - VMPL0
              - SVSM
                - vTPM
                - VMRUN
  - QEMU / KVM
  - AMD Secure Processor

KVM Forum 2023
SVSM-vTPM Ephemeral State

• On every boot, we:
  • Create the EK
  • Request VMPL0 report to bind the EK to the AMD SNP platform
  • Save the VMPL0 report in the TPM NVRAM

• Advantages
  • Simple design
  • Allow TPM-based boot and runtime attestation
  • No need to eject or inject state
    • No need to early attest the VM Launch Measurement before injecting the TPM state
    • No need to secure the TPM state at rest and in-transit
    • State injection requires more code and orchestration at early boot
    • Its attack surface is considerably smaller than persistent state
  • Simplified SVSM-vTPM migration

• Disadvantages
  • May have limited use cases
    • However, full disk encryption can be enabled by an intermediate storage key
SVSM-vTPM Architecture

- SVSM-vTPM proof-of-concept
  - [https://github.com/svsm-vtpm/linux-svsm](https://github.com/svsm-vtpm/linux-svsm)
  - Tested the SVSM-vTPM with keylime

- Contributing it to SVSM open source projects
SVSM-vTPM Demo

- TPM NVRAM: VMPL0 Attestation Report (signed with AMD VCEK)
  - report_data: SHA512_hash(ek-pub)
  - Vmpl: 0

- Validate SVSM-vTPM
  - Attestation report signature
    - Read VMPL0 Attestation Report saved in the TPM
    - Download the VCEK certificate from the AMD website
    - Validate the attestation report signature
  - EK pub
    - Read EKpub from the SVSM-vTPM
    - Check report_data == SHA512_hash(EKpub)
  - Check VMPL == 0
Conclusions

• vTPMs can be used in CVMs to extend the chain of trust up to the guest OS

• SVSM-vTPM:
  • Protected from the guest OS and Host OS
  • Its EK uniquely identify the CVM
  • Authentic: its EKpub is chained back to AMD hardware root-of-trust
  • It's state is ephemeral (simple design) and it allows use of the TPM-based attestation
  • With small TPM driver changes the existing TPM tooling can be reused

• We're contributing the SVSM-vTPM proof-of-concept to SVSM open source projects
Questions?

• How can we improve isolation between modules in the SVSM?

Thank you!