Zero-Trust vTPM for Confidential VMs

Claudio Carvalho
cclaudio@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

```
sha1:
 0 : 0x016D0C6607090DF56F0C0B5309D4829FC36BD2D0
 1 : 0x171C7AD21E3894793F06E82214DEF97665632015
 2 : 0xEC63A51BAAA287552E5AAF0A948CC4C85BE7E5A7
 3 : 0xB2A83B0EBF2F8374299A5B2BDFC31EA955AD7236
 4 : 0x1B806B4F8C9C34A0B74AC05DC4E1C6FB83FFD095
 5 : 0xEBBEA2E1B6C039484B0DD89F73CF34770C9B9D73
 6 : 0xB2A83B0EBF2F8374299A5B2BDFC31EA955AD7236
 7 : 0x0EA335A02680C87B5C19A232A8F6CAD8B4DC7F35
 8 : 0x5F033D3BC86210B51D83E7890FE89288FDF65E89
 9 : 0xFF5D221A16E56901FD30F6CC2A4A0B2DB8E1423C
 10: 0xB875FD5EA0E53E87CB89F26A204031DF2539D4F1
 11: 0x0000000000000000000000000000000000000000
 12: 0x0000000000000000000000000000000000000000
 13: 0x0000000000000000000000000000000000000000
 14: 0x6B03BDE55DC2938FB94317EB2169BCF88204A4B1
 15: 0x0000000000000000000000000000000000000000
 16: 0x0000000000000000000000000000000000000000
 17: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
 18: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
 19: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
 20: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
 21: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
 22: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
 23: 0x0000000000000000000000000000000000000000
```

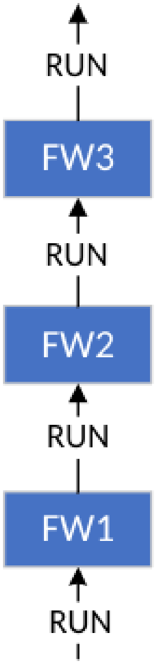
Platform Configuration Registers (PCR)

```
FW4_BLB: 0xA18BC18791AAD83BA3872A74AEB4F53358DF5AFC
FW4_CFG: 0x8628BEC2CD448684E0BA7AD222F644C4D3E73AEE

FW3_BLB: 0x068AA16114684203C91BA0C29E5485EEB9EC8746

FW2_BLB: 0x451903A2AF538C4E6ACA44189845C703B8656FE9
FW2_CFG: 0xB2A83B0EBF2F8374299A5B2BDFC31EA955AD7236
```

Event Log



PCR Extend operation

PCRnew := *HashAlgo*(PCR*current* || digest)

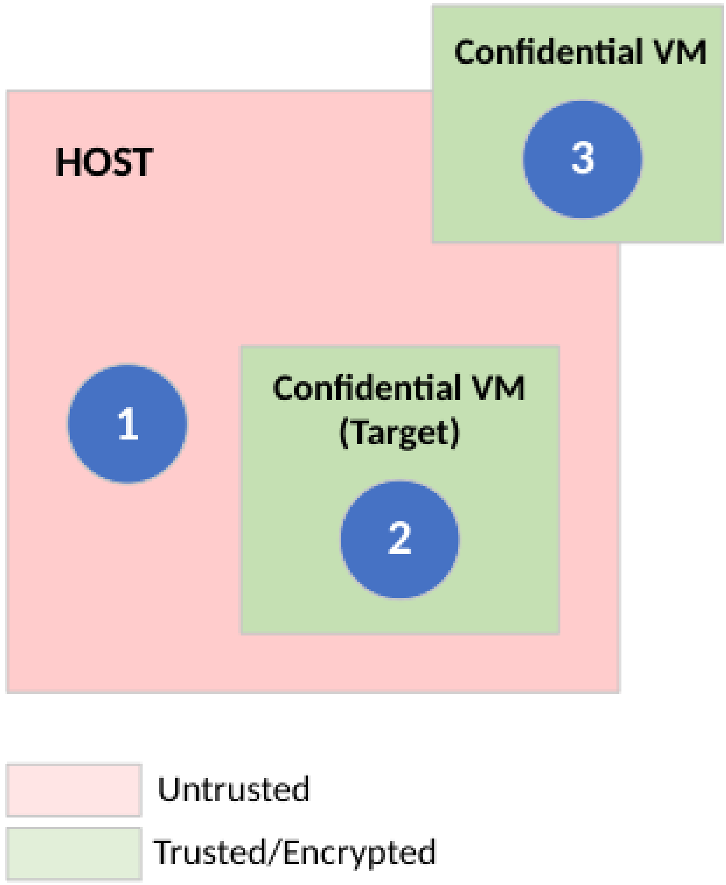
Zero-trust vTPM for attestation

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. <u>In the actual CVM</u>
3. In a separate CVM



Where should we run the vTPM?

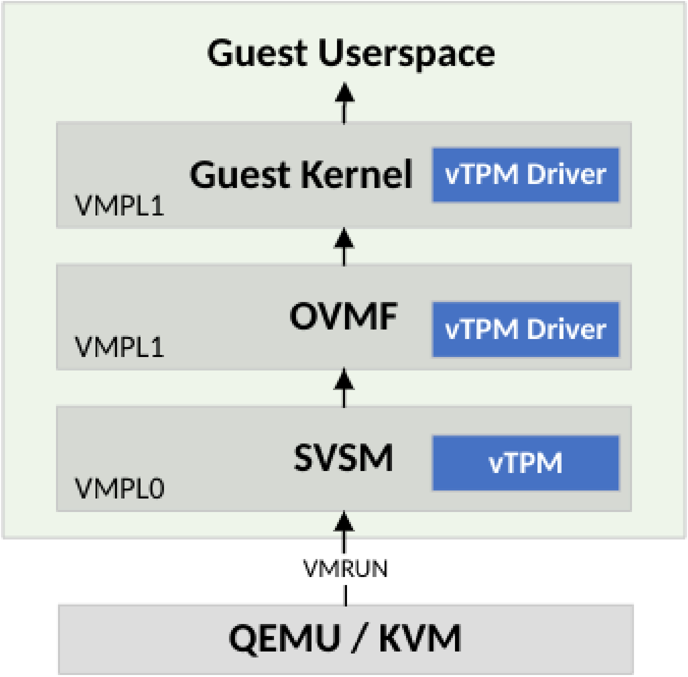
- We run the vTPM in the Secure VM Service Module

- Why do we run it in the SVSM?
  - First module that runs in SEV-SNP guests
  - Provide runtime privileged services to the guest
    - SVSM specification - https://www.amd.com/en/developer/sev.html
  - 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

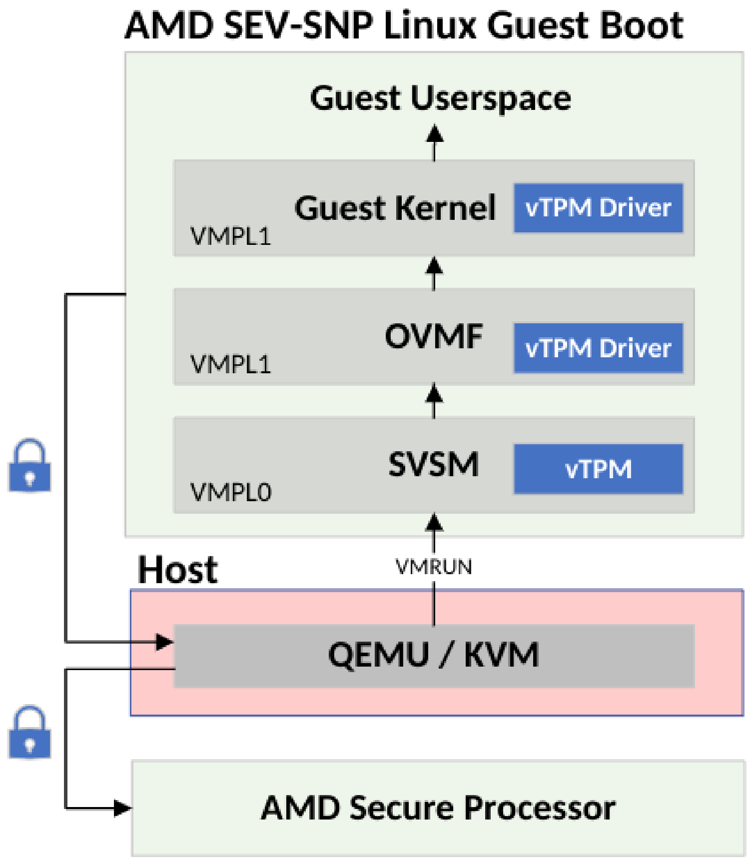
AMD SEV-SNP Linux Guest Boot



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



SVSM-vTPM Ephemeral State

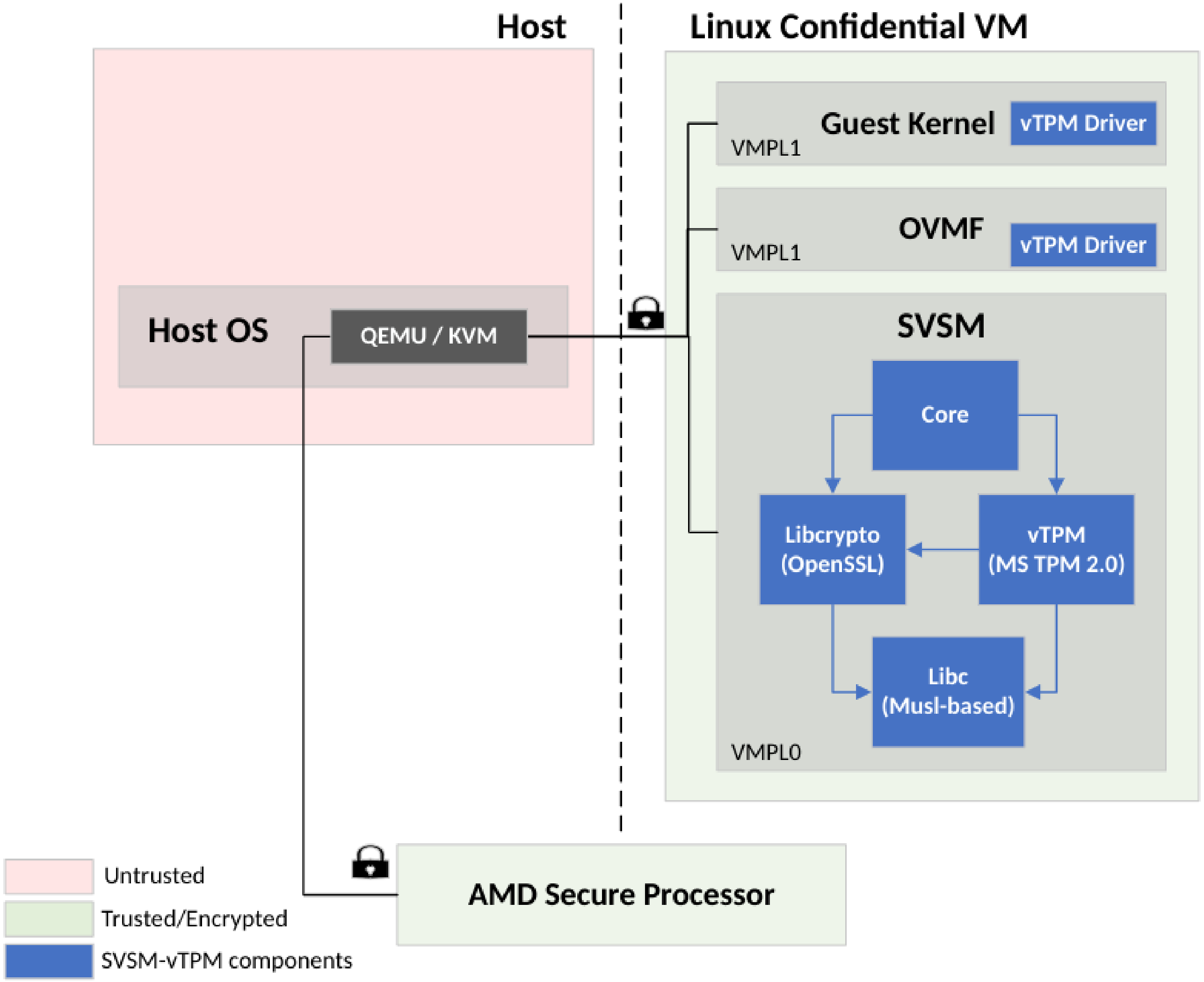
- On every boot, we:
    - Create the EK
    - Request VMPL0 report to bind the EK to the AMD SNP platform
    - Save the VMPL0 report in the TPM NVRAM

- 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
  - 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!