Unifying Confidential Attestation

Tobin Feldman-Fitzthum, Dov Murik
IBM
Set the stage

- VM-based Confidential computing
- Untrusted host/hypervisor
- VMs with encrypted memory
- Terms:
  - Guest/Attester
  - Host
  - Owner/Relying Party
What we will cover

• Overview of attestation mechanisms
• Approaches for unification
• Focus on guest
SEV(-ES)

- Pre-attestation driven by the host
- Secure channel established before boot
- Launch measurement covers:
  - Platform information
  - Launch digest (hash of VM firmware and initial vCPUs state)
- Secret injected using secure channel after measurement verification
SEV-SNP

• Driven by the guest
• Signed attestation report covers:
  – Platform information
  – Launch digest (hash of VM firmware and initial vCPUs state)
• VM separated to permission levels (VMPLs)
Driven by the guest

Signed attestation report covers:
- Platform information
- VM firmware, initial vCPUs state
- Can be extended using 4 RTMRs (run-time measurement registers)

Two phases: get report; then sign it
s390 Secure Execution

• Driven by the guest
  – Requires encrypted attestation request from owner (guest passes it to ultravisor)

• Signed attestation report covers:
  – Platform information
  – kernel + initrd + kernel command-line

• Optional – guest image is encrypted
• Measure the entire stack with every TEE
• Let the HW measure part of the stack
• Use software to measure the rest
• Where should we split between HW and SW?
Firmware (1)

- HW measures the firmware
- Firmware measures the rest of the stack by securely emulating a TPM
  - Secure vTPM can’t be tampered with by host or guest OS
- Secure vTPM generically exposes attestation to the guest OS and facilitates measurements
- vTPM provides a generic interface across all hardware and requires little to no modification of guests
  - Existing interface with kernel and measurement infrastructure
  - Shared between platforms and deployment types
- Can be implemented as an SVSM with SEV-SNP
  - Memory encryption prevents tampering by host
  - VMPLs prevent tampering by guest OS
How do you provision it?
- Inject identity into vTPM
- How do we securely manage these TPM identities?
  - TPM is unusable until identity has been provisioned.

Alternatively, use ephemeral vTPMs, which are manufactured for each guest
- Include the hash of the public EK in the attestation evidence
- Verifier of TPM quotes will have to validate evidence
- SVSM/VMPLs only supported with SEV-SNP
- # TDX RTMRs < # TPM PCRs
- If no VMPLs: how can we protect from a malicious kernel?
  - Maybe in another trusted VM
- vTPM is just one option for firmware-based measurement
• HW measures firmware and kernel
• Kernel measures the rest
  – Uses software kernel-mode TPM
• There are unification efforts in the kernel for other aspects of confidential computing
Initrd

- HW measures firmware, kernel, and initrd
  - Using measured direct boot (for SEV / SEV-SNP)
  - OVMF extends RTMRs during measured boot (for TDX)
- Attestation Agent in initrd measures the rest
  - Used in Confidential Containers to provide workload secrets/certificates
  - Could also be used for encrypted disks
**Initrd**

- Easier to implement
  - Modular user space process
- Non-standard initrd
  - Distros can add attestation agent

Diagram:

- Application
- Initrd
- Kernel
- Firmware
- HW

- KBC
- KBS
- Attestation Agent

Connections:
- get_secret
- get_resource

The Linux Foundation
• HW measures the entire stack
• Today, most TEEs don’t measure the whole stack
• HW is less flexible and standardized
• Extending HW measurement to the whole stack can have performance implications

• This is non-unified attestation
Attestation verification

- Verification follows from guest implementation
- Requires validation of hardware measurement
- Might also require validating a software measurement
- HW/SW validation could be split between multiple services
Further research

• Which approach should we take?
  – Are they interoperable?

• Supporting new architectures (ARM-CCA, RISC-V, …)

• Can we standardize future versions of hardware