Securing trusted boot of confidential VMs

Stefan Deml <Stefan.Deml@decentriq.com>
Andras Slemmer <Andras.Slemmer@decentriq.com>
Sep 21, 2021
Motivation for using Confidential Computing

Enable confidentiality and integrity of multi-party remote computations.

Required security guarantees:
- Memory confidentiality
- Memory integrity
- Memory freshness
- Code auditability
- Attestation of the TCB and software running remotely
- Control flow integrity:
  - Process based CC (SGX)
  - VM based CC (TDX, SEV)?
Attestation of remote software: Measured Boot

**Boot Process**

**Static: Large firmware**
- firmware || initramfs || kernel || cmdline
- 0xab12c3

**Static: Hash chain**
- firmware (measurement = 0xdea313)
- load measure assert
- initramfs || kernel || cmdline
- 0xdea313

**Dynamic: - Isolation (VMPL) - PCR like scoping**
- firmware
- load measure isolate
- initramfs || kernel || cmdline
- 0xdea313

**Attestation**

- 0xab12c3
- 0xef17c9
- CPU key

- 0xab12c3
- 0xef17c9
- CPU key

- 0xab12c3
- 0x8112b3
- fw key
- 0xef17c9
- CPU key

- 0xdea313
- 0x8112b3

- 0xab12c3
- hash digest
- signature
- key
Providing Control flow integrity (CFI)

- Firmware: **OVMF**
  - long-term CC support
- Hypervisor: **QEMU/KVM**
- VM parameters passed in:
  - memory size/e820
  - CPU Count
  - ACPI, ...
- Kernel: stripped down
  - Limit IO to VSOCK
- Virtualized devices
  - RNG
  - Time
  - Option ROM
  - Virtio PCI

**Problem:** How do we provide CFI using existing software (OVMF, linux)?
Thank You

www.decentriq.com