Linux on a Confidential VM in a cloud: where's the challenge?

FOSDEM2024

Vitaly Kuznetsov
vkuznets@redhat.com
Confidential VM types on public clouds

- **Google Cloud Platform**
  - C2D/N2D (AMD SEV), (AMD SEV) option announced in June, 2020.
  - N2D (AMD SEV-SNP), (AMD SEV-SNP) option private preview announced in April, 2023; public preview since January, 2024.

- **Microsoft Azure**

- **Amazon Web Services**
  - M6a, C6a, and R6a (AMD SEV-SNP), SEV-SNP feature GA announced in May, 2023.
Guarantees

- **Confidentiality guarantees provided by hardware:**
  - VM’s memory is encrypted.
    - Encryption is ‘transparent’ when observed from within the VM.
  - CPU state is encrypted (SEV-ES/SEV-SNP, TDX).
  - Memory integrity guarantees are provided (SEV-SNP, TDX).

- **Confidentiality guarantees NOT provided by hardware:**
  - Protection of data at rest (must be ensured on the guest level!).
  - Protection of data in transit (not specific to CVM).

- **Guarantees which CANNOT be provided with existing hardware:**
  - Non-disruption guarantees.
Protecting data “at rest”

- Must be done at the guest (not host!) level.
- OS data must be protected too:
  - Sensitive OS data (configuration files, private keys, random seed, etc.) must be fully protected (==encrypted)
  - The rest of the operating system (e.g. executable files) requires at least “write protection” from the host (encrypted and/or integrity checked)
- Standard tools (e.g. LUKS for encryption, dm-verity for integrity checking) can be used as the key in memory is protected.
  - ... but how does the guest get the right key/hash?
Linux cloud CVM: where’s the challenge?

Encryption

Platform firmware

Unencrypted part

Bootloader (shim, GRUB, sd-boot, …)

Linux kernel

Initramfs

Encrypted part

Operating system on the root volume

Standard Linux boot process

Root volume is encrypted, the key is provided at initramfs time.
Encryption

- Providing a key “manually” (e.g. by entering a password through console) is doomed to be not only inconvenient, but also insecure.

- Keys to sensitive data must be provided to the guest in an automated fashion only after checking that it is in “known good state”:
  - The guest is running in a genuine CVM.
  - All the code which was executed on the CPU (loaded from unencrypted part) is “trusted”.
  - These properties must be proved to a trusted third party holding the key.
vTPM can (sometimes) be used as a trusted “third party”

- Different implementations of vTPM
  - As part of firmware running in a different “trust level”.
  - As a separate “domain”.
  - As a separate “partition”.
  - As an emulated device on the host.

- Different types of vTPM:
  - Stateful
  - Stateless
Stateful vTPM

- Implementation examples:
  - Azure SEV-SNP, Azure TDX
  - AWS SEV-SNP
  - GCP SEV-SNP
- The “state” may be isolated from the host:
  - Azure SEV-SNP/TDX claim to provide the isolation.
  - Isolation claims cannot be proved from within the guest.
- When vTPM’s public key (e.g. SRK) is known, it allows to implement “pre-encryption” of the root volume.
- Self-encryption upon first boot can (in theory) be implemented without the need to know public key in advance.
Stateless (ephemeral) vTPM

- Implementation example: Azure TDX.
- In theory, allows for “zero trust” solutions.
- **Should** be implemented as part of firmware and thus can be measured/attested by the guest:
  - Simpler with SEV-SNP, harder with TDX.
  - In theory, can be brought by cloud user (“bring your own firmware”) but no real world implementations yet.
- Could not be used to store/protect secrets, an external attestation server is needed.
  - An intermediary key can be injected after successful attestation thus reproducing “stateful” experience.
Linux cloud CVM: where’s the challenge?

Stateful vTPM with no explicit confidentiality guarantees

- Implementation example: AWS SEV-SNP, GCP SEV-SNP
- Can’t be used if isolation from the host is a must :-(
- Can simply be “ignored”:
  - An external attestation server is needed.
  - A non-vTPM unlocking method for root volume is required, no “standard” for that yet.
  - It’s unclear whether PCR measurements can still be used or not (implementation specific).
Verifying unencrypted part

- Traditionally, SecureBoot/Measured boot technologies are used for early boot integrity protection.
  - SecureBoot: all artifacts in the boot chain are signed by known keys
  - Measured boot: all important information about boot process is recorded in TPM PCRs
- To get initramfs under SecureBoot protection and get some ‘verifiable’ measurements, it must be built and signed by a trusted party (e.g. OS vendor).
- To use existing verification mechanisms, “Unified Kernel Image” concept is introduced:
UKI implications: static initramfs

- **Initramfs is static** and built at kernel package build time
- The list of drivers and tools is fixed by the OS vendor, the applicable scope must be defined.
  - E.g. Fedora/RHEL ship “kernel-uki-virt” package with drivers needed for popular virt/cloud environments (Virtio, VMBus, Xen, NVMe,...).
- Systemd **system extensions** mechanism can be used to extend initramfs (with limitations)
UKI implications: static cmdline

- **Kernel command line is static** and built at kernel package build time
- Must be “one size fits all” so e.g. kernel-uki-virt in Fedora/RHEL ship with “console=ttyS0 console=tty0” cmdline.
- Passing “root=UUID” is not possible:
  - Root volume must be auto discovered, e.g. with `systemd-gpt-auto-generator` systemd feature.
- “Signed extensions” mechanism for systemd-stub was recently added upstream.
  - Can be used both by the OS vendor and the instance owner (with limitations).
UKI implications: boot flow

- UKI is a UEFI binary and can be loaded:
  - Directly from firmware:
    - Signing key (vendor) must be in SecureBoot DB, revocations must use DBX.
  - By ‘shim’:
    - Signing key can be in SecureBoot DB, shim’s ‘vendor_cert’/’vendor_db’, MOK.
    - SBAT mechanism can additionally be used for revocations.

- No “bootloader UI” experience:
  - Fedora ships kernel-bootcfg (‘uki-direct’ package) for automatic UEFI boot variable management.
Attestation client/server

- Remote attestation **must** be used in all “Stateless vTPM”/”Untrusted vTPM”/”No vTPM” scenarios.
- The presented “evidence” can differ:
  - Different hardware technology (SEV-SNP, TDX).
  - Method to obtain measurements (directly from hardware, through vTPM,...)
  - vTPM/no-vTPM.
- No “standard” implementation for open-source attestation client/server atm
  - [KBS](#) project from CoCo looks promising!
Linux cloud CVM: where's the challenge?

Integrity checking

Platform firmware

Non-integrity-checked part (ESP)

Bootloader (shim, GRUB, sd-boot, ...)

Linux kernel

Initramfs

Integrity checked part

Operating system on the root volume

Standard Linux boot process

Root volume is integrity checked, the expected hash is known at initramfs time.
Integrity checking

- Must support runtime checking (e.g. dm-verity)
- Must be accompanied by a trusted kernel/initramfs (UKI)
- OS needs to know the expected root hash:
  - Can be built into UKI (not suitable for general purpose distro UKIs)
  - Can be a signed cmdline extension
  - Can be sourced from a signed file on ESP
- Can be accompanied by writeable overlay
  - All considerations for ‘encryption’ apply
Additional considerations: VM authenticity

- VM user needs a way to verify that they are connecting (‘ssh ...’) to their own CVM and that protection mechanisms (SecureBoot, encryption, integrity checking, ...) were actually used.
  - Customized (e.g. customer uploaded pre-encrypted) images can be tailored for the specific deployment and can contain pre-encrypted secrets.
  - Generalized (e.g. Marketplace) images normally support various types of deployment (different instance types, CVM/non-CVM, vTPM/no-vTPM, ...) and thus require additional attestation.
Additional considerations: image contents

- Runtime guest agents
  - Cloud-init, WALA,... are **not** isolated from the host as the host provides the (untrusted) data source.
  - Malicious host can try emulating any cloud data source.

- Virtual hardware
  - Malicious host can try attacking the guest by presenting any device which has corresponding guest driver.
  - Emulated hardware (e.g. serial console) should always be considered ‘insecure’; no sensitive data should appear in the output/input.
Additional considerations: VM storage

- Replay attacks
  - Malicious host can try presenting an older version of guest’s storage or some parts of it at any time.

- Source image integrity
  - Even when full disk encryption or integrity checking is in use, it is possible to present an older version of the source image.
  - A guest verifiable data about the source image must be conveyed.
  - No ‘standard’ way for doing this atm.
Thank you

Red Hat is the world’s leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.