COCONUT SVSM

Early attestation to unlock persistent state

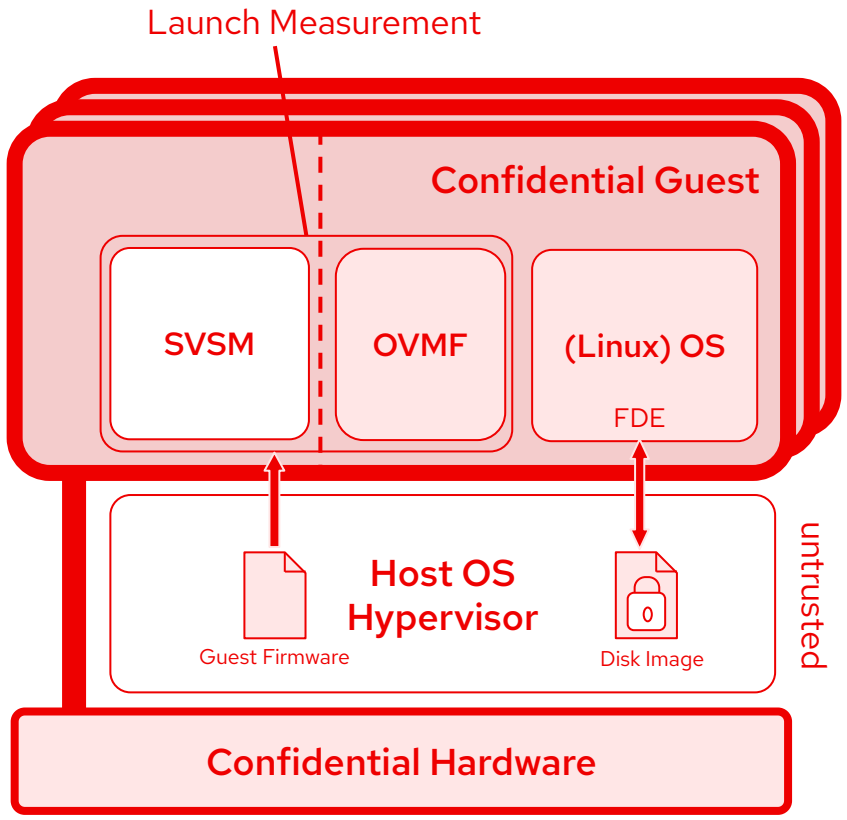
KVM Forum 2024

Oliver Steffen
<osteffen@redhat.com>

Stefano Garzarella
<sgarzare@redhat.com>

Red Hat

Quick SVSM intro



- Confidential Virtualization:

  Hardware is trusted, Host OS/cloud provider is not trusted
- Hardware guarantees confidentiality of guest memory and CPU state
- Remote Attestation -> confirm that running in valid CVM
- Launch Measurement covers initial state
- Data in use is protected

  Data at rest -> guest OS has to use disk encryption
- SVSM:
  - Paravisor running isolated from both guest OS and host
  - Provides security critical services to the guest, example: virtual TPM

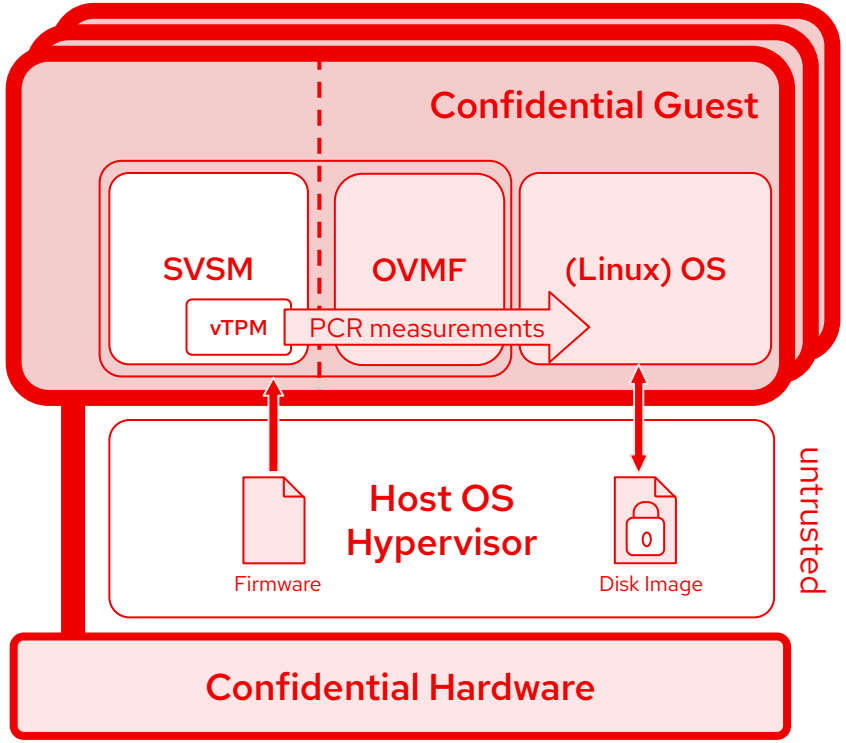
Red Hat

End User Goals

Support current Linux distros as confidential guests with as few changes as possible.

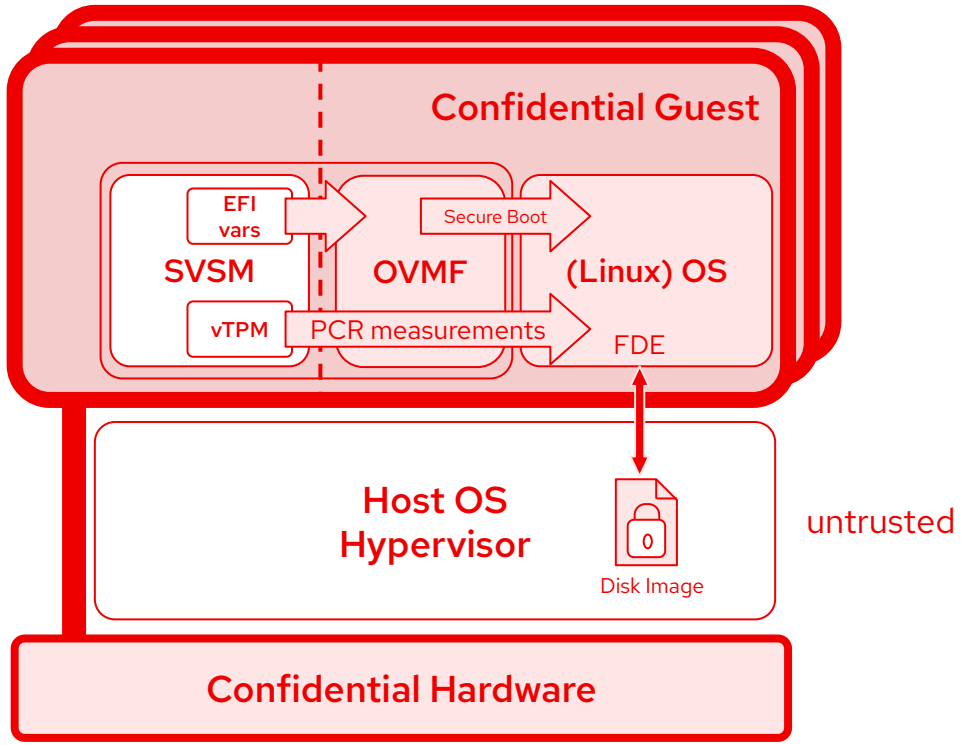
- Easy migration from regular VMs to CVMs
- Support long living guests
  - Stop + Restart
  - Persistent, mutable disk
  - Self-updating OS during runtime
- Guest OS should not have to deal with attestation
- Support measured boot & automatic FDE unlocking via TPM's PCR policy
- Support SecureBoot

Current state of Coconut on AMD SEV-SNP



- Linux and OVMF support running under an SVSM already
- Coconut SVSM provides an ephemeral vTPM
  - vTPM is remanufactured at each boot
    - Primary keys change
    - NV storage is not preserved
    - Monotonic counters are not preserved
  - Allows PCR measurements
  - Unsealing secrets (-> FDE keys) does not work
- "In RAM" UEFI variable store
  - Volatile: User can't customize SecureBoot settings, boot options, etc.
  - Not possible to implement securely in OVMF due to lack of SMM and no storage device
- Open Questions
  - How to automatically unlock the root disk?
  - When and how to do remote attestation?

SVSM Persistent State



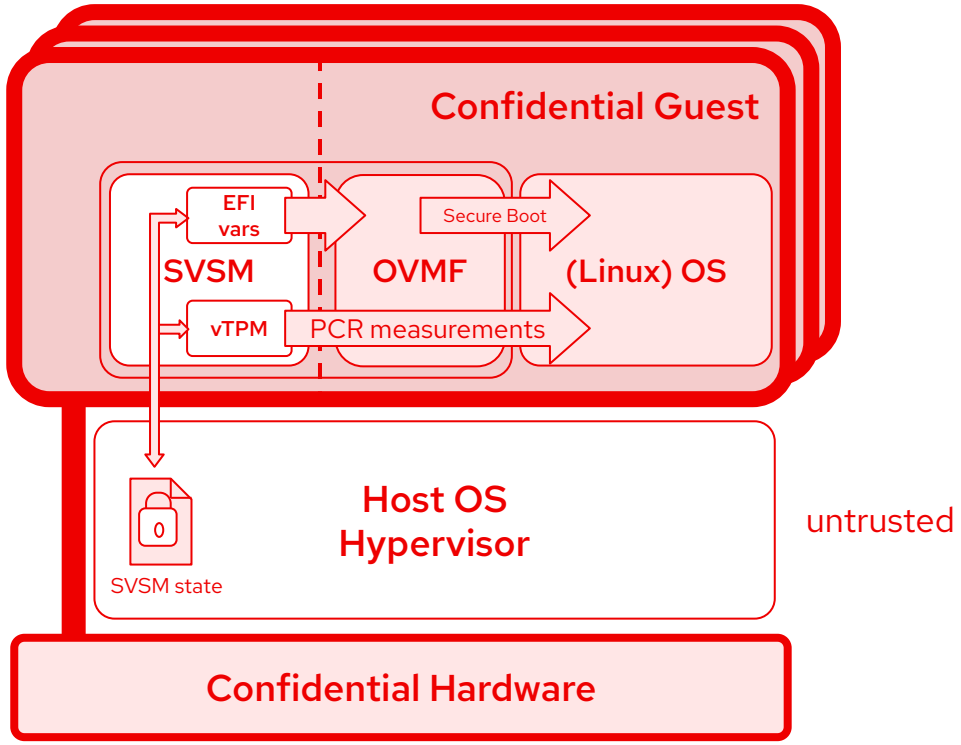
Goals:

- Provide a fully functional persistent vTPM
  - Preserve TPM identity, counters, and storage across reboots
- Provide a fully functional UEFI Storage
  - Implement a variable service in SVSM that OVMF can talk to
  - Manage access in SVSM
- This allows
  - Measured boot + disk unlocking via PCR policy
  - Configurable SecureBoot

To implement these features, we need **SVSM to support a persistent state** across reboots.

Red Hat

SVSM Persistent State



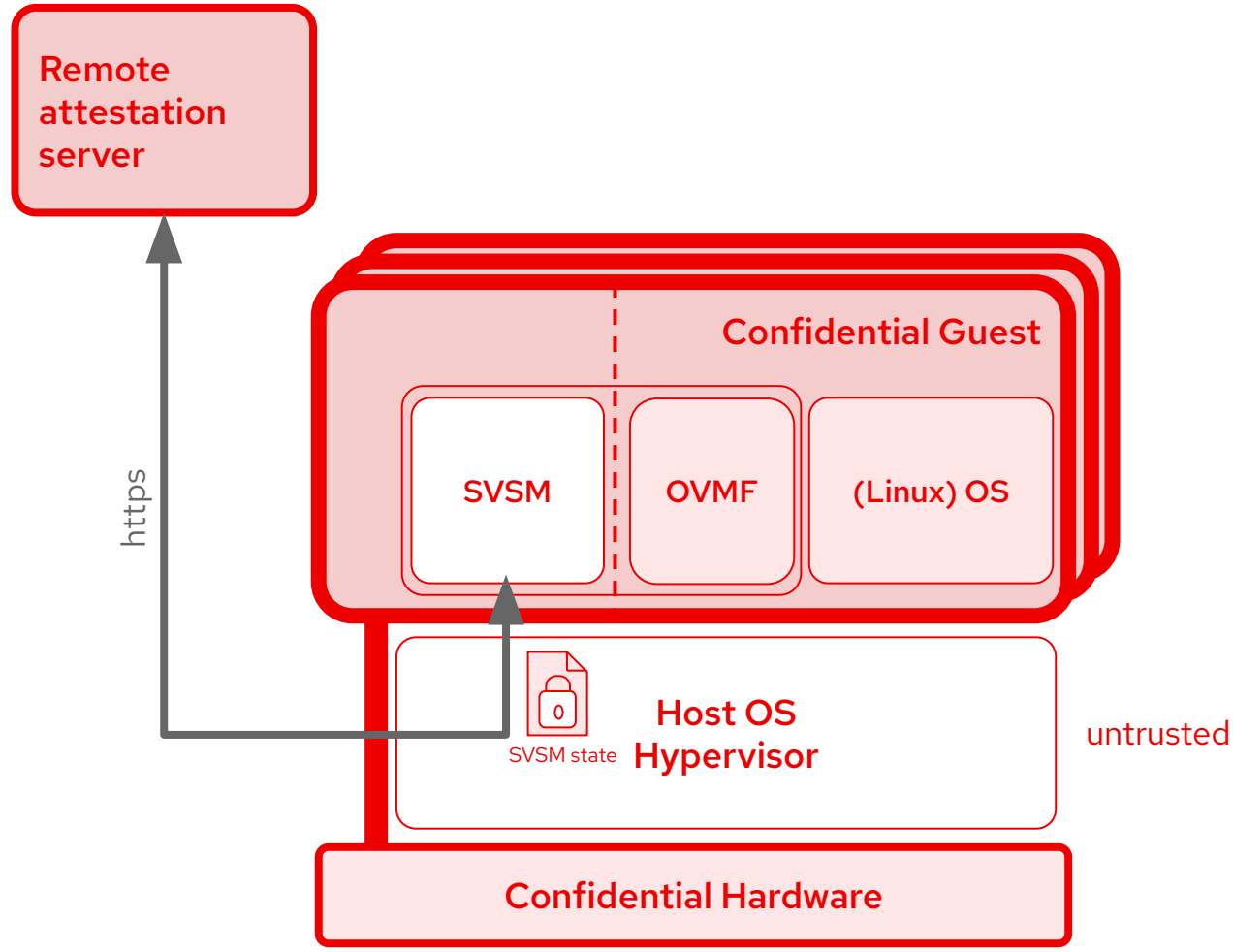
untrusted

- SVSM State = vTPM state + UEFI variables + ...
- Add a storage driver to SVSM
- Storage backend provided by the host
- Use encryption - the host is not trusted!
- Probably need to support multiple drivers for different hypervisors

How to decrypt the SVSM state ?

Early attestation in SVSM

- Encrypted state
  - Unlocked only after a successful remote attestation
    - Early attestation in SVSM is needed
- Remote attestation
  - HW generates an attestation report (evidence)
    - signed by HW's vendor certificate
  - Remote server (trusted) checks the evidence
    - Expected SW running on a genuine HW
    - extra parameters needed for the attestation (e.g. nonce, identity)
  - Remote server sends back the SVSM state key
    - Unlock vTPM state, UEFI variable service, etc.
- Challenges
  - Network stack not available in SVSM
  - Support multiple remote attestation protocols

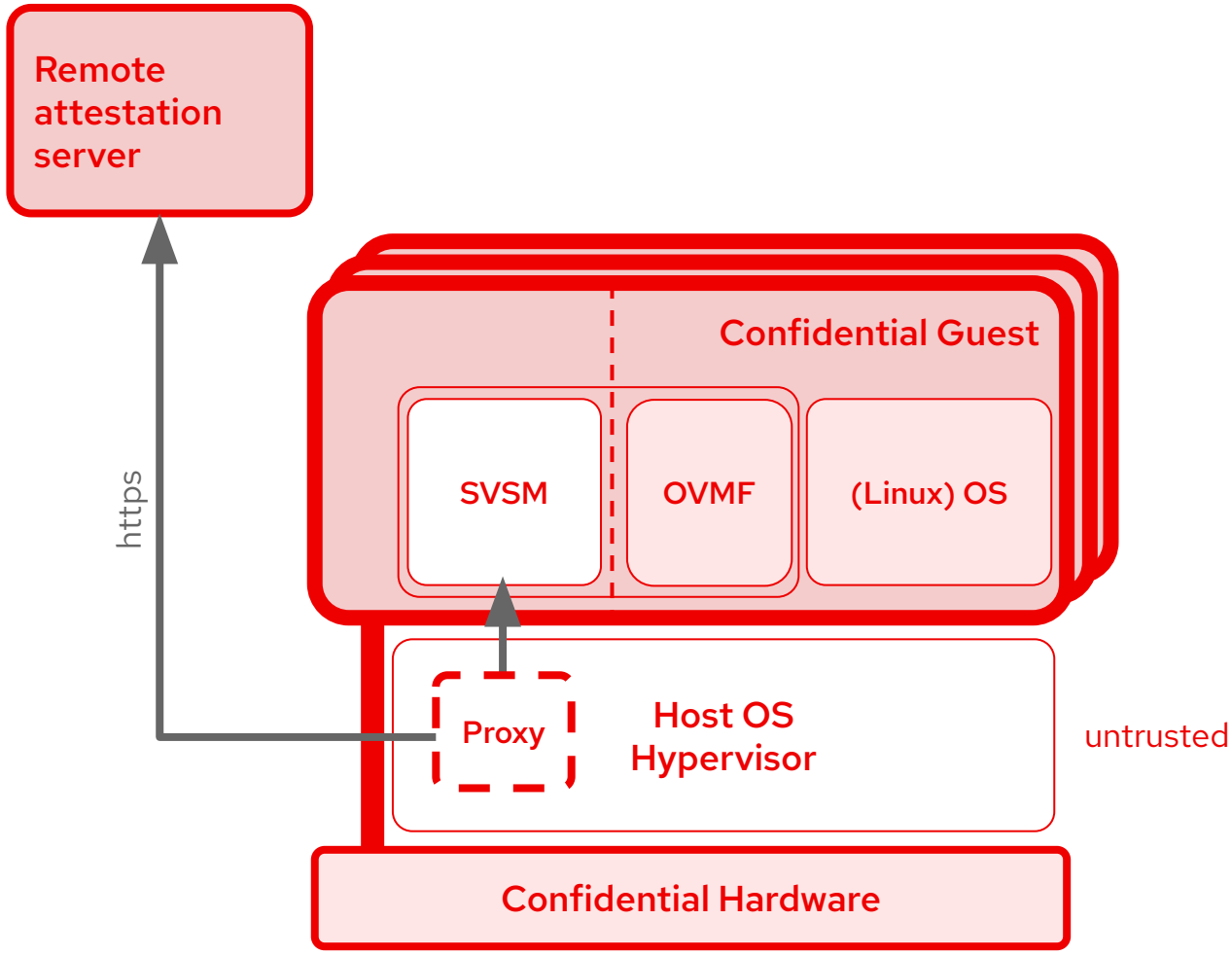


SVSM community proposal for early attestation:
https://docs.google.com/document/d/11ZsxP8jsviP3ddp9Hrn0rf6inttNw_Pbnz0psXlxlPs

Red Hat

Attestation proxy

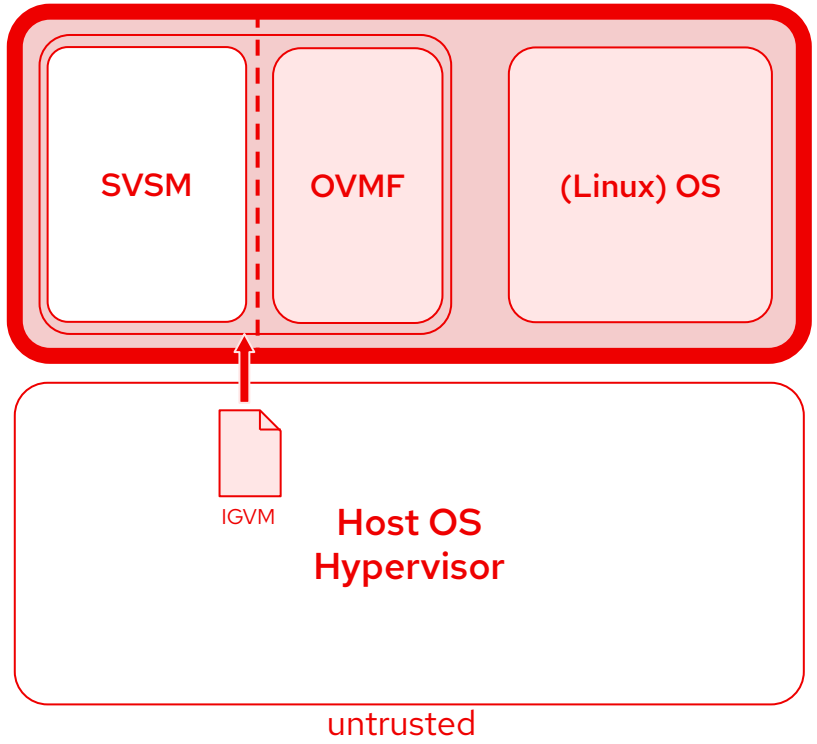
- Proxy application running on the host
  - Allow SVSM to talk to a remote attestation server
  - Forward SVSM request using an https channel
- Challenges
  - Application running on an untrusted host
    - MITM
      - SVSM must generate asymmetric key to receive secrets
    - DoS
      - Out of context for confidential computing
  - host <-> SVSM communication channel



Red Hat

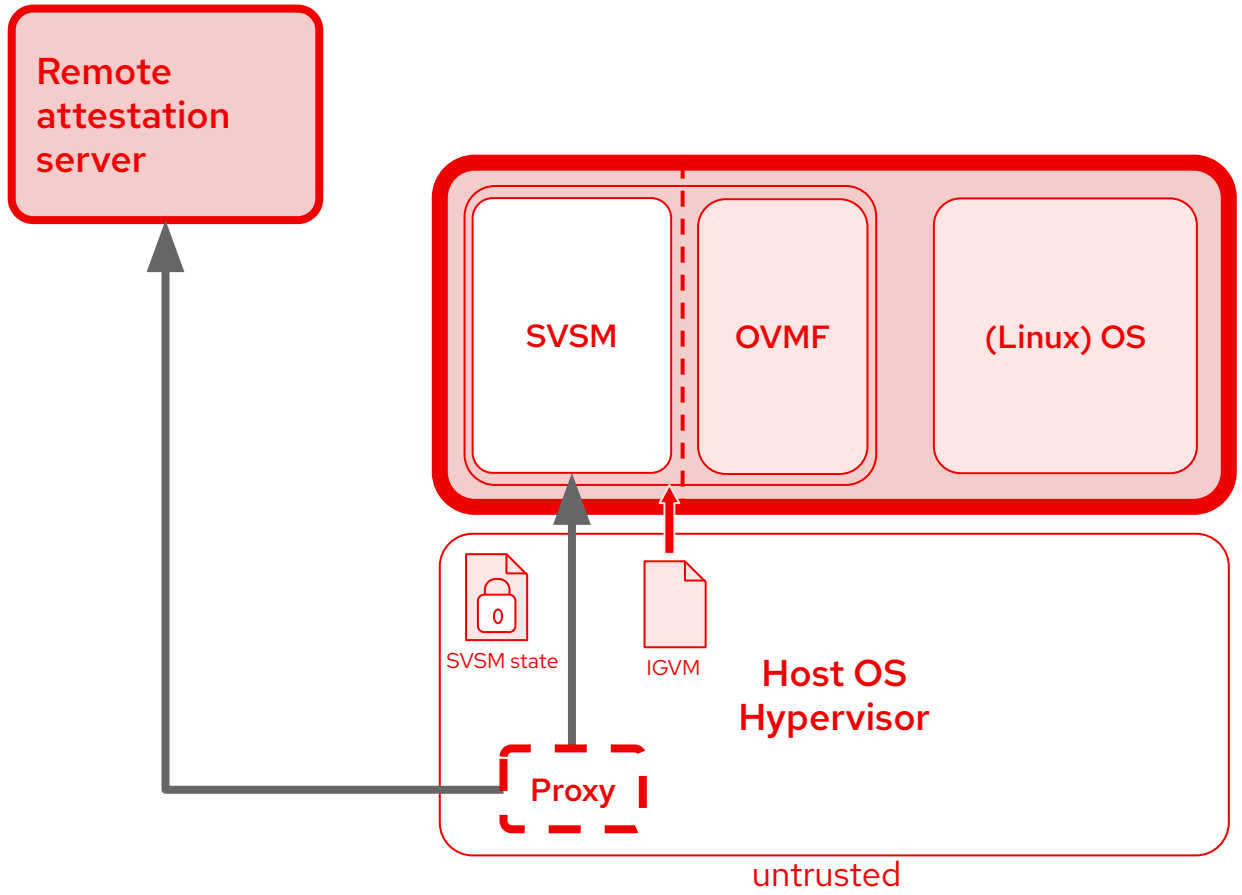
“How it all works”

- SVSM boots up from IGVM file



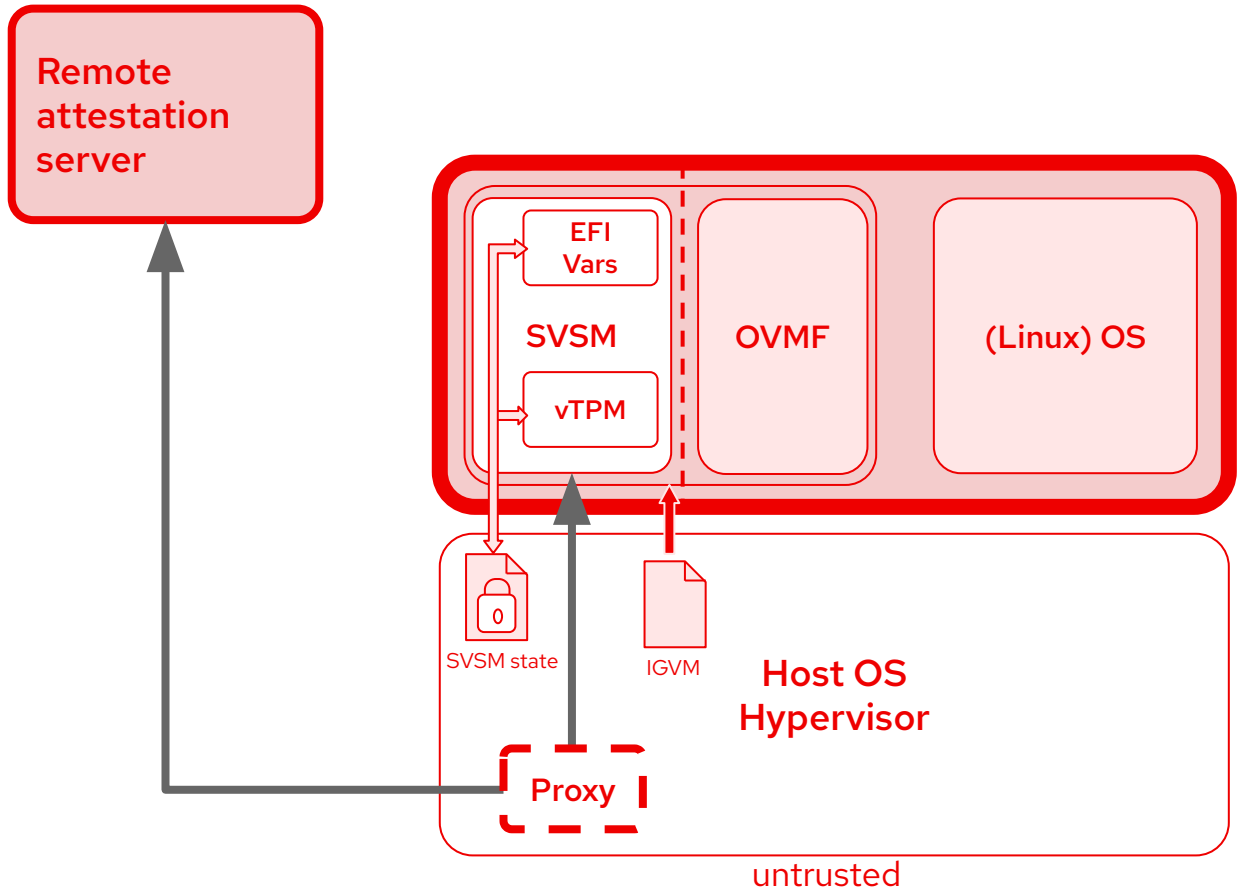
Red Hat

“How it all works”



- SVSM boots up from IGVM file
  - Uses proxy to connect to attestation server
    - Sends attestation report
    - Receives key for SVSM state store
  - Unlocks state store

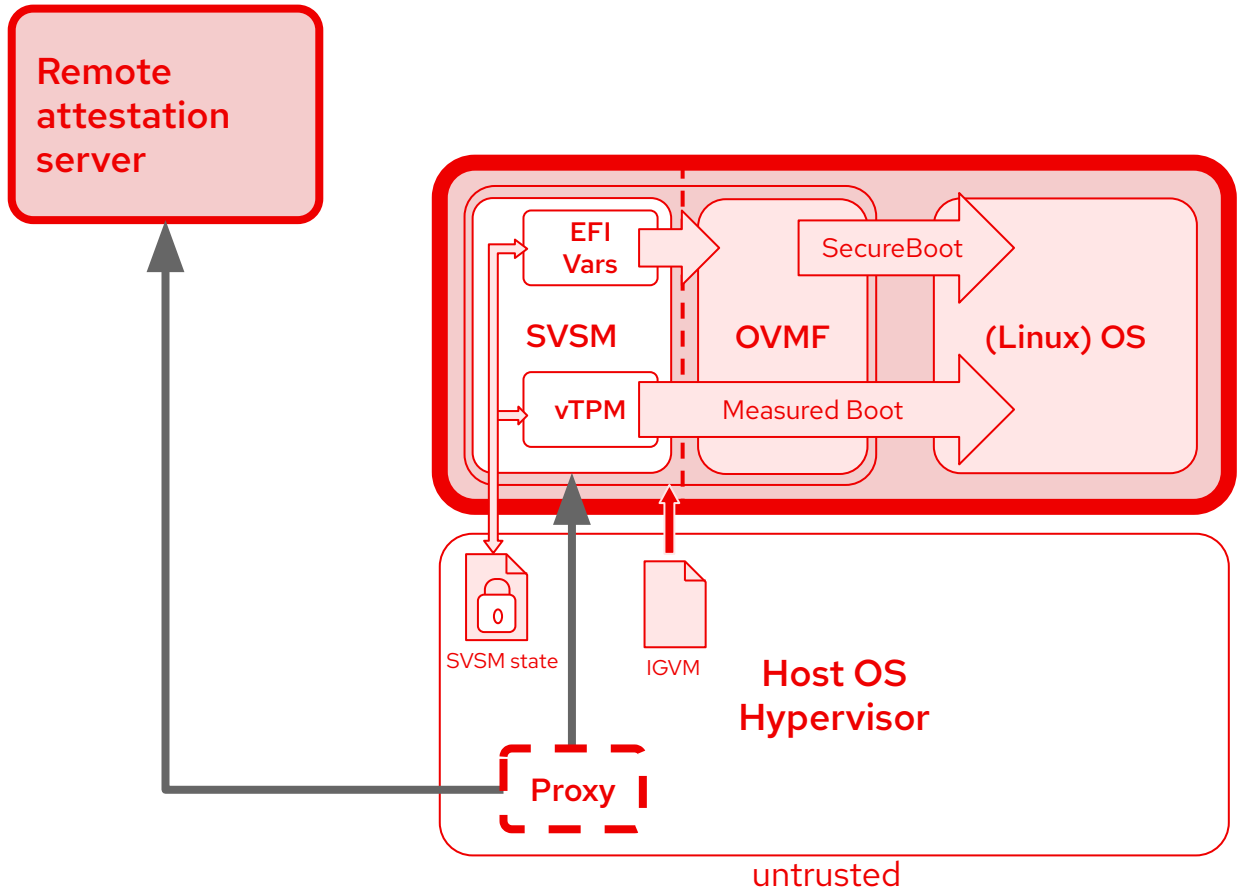
“How it all works”



- SVSM boots up from IGVM file
  - Uses proxy to connect to attestation server
    - Sends attestation report
    - Receives key for SVSM state store
  - Unlocks state store
  - Initialize vTPM and UEFI variable service from that
  - Continues boot process and launches OVMF

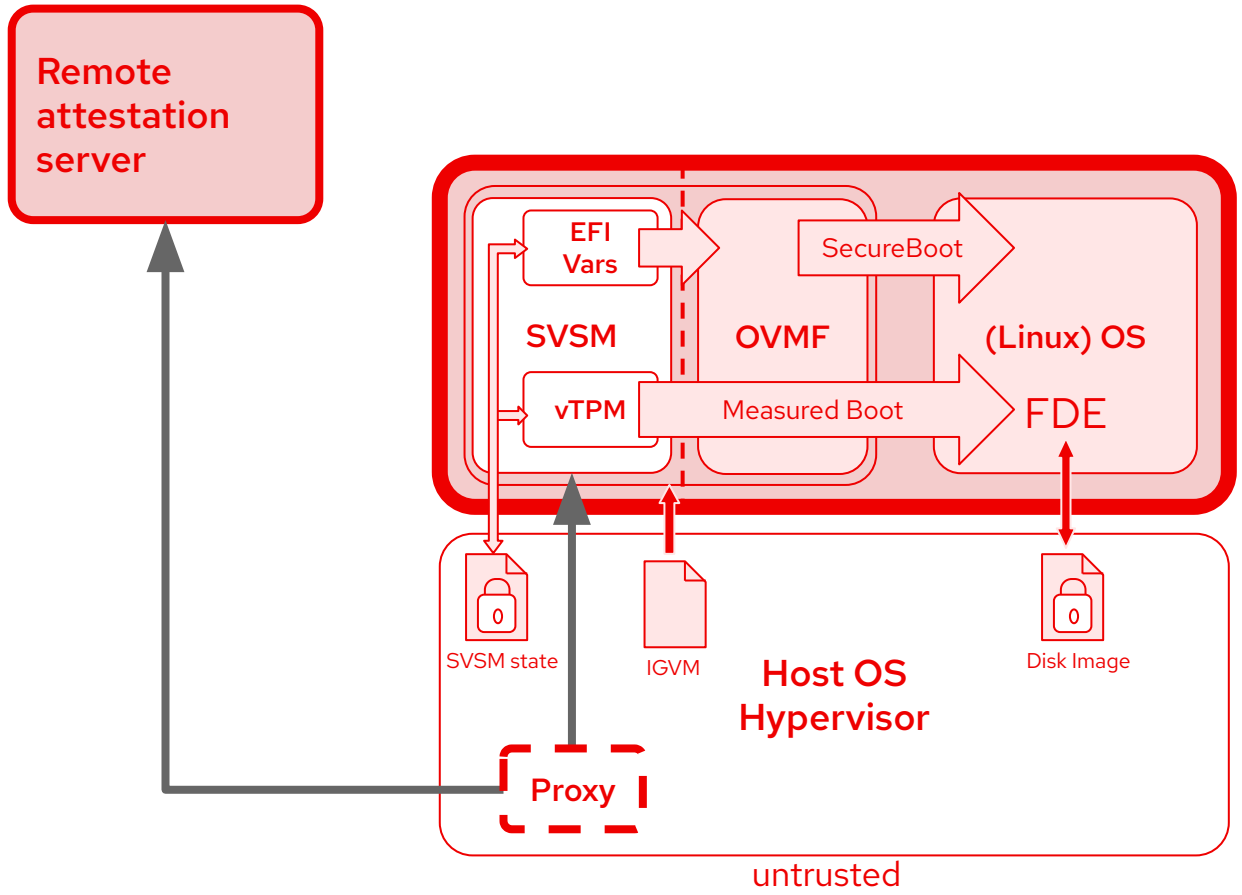
Red Hat

“How it all works”

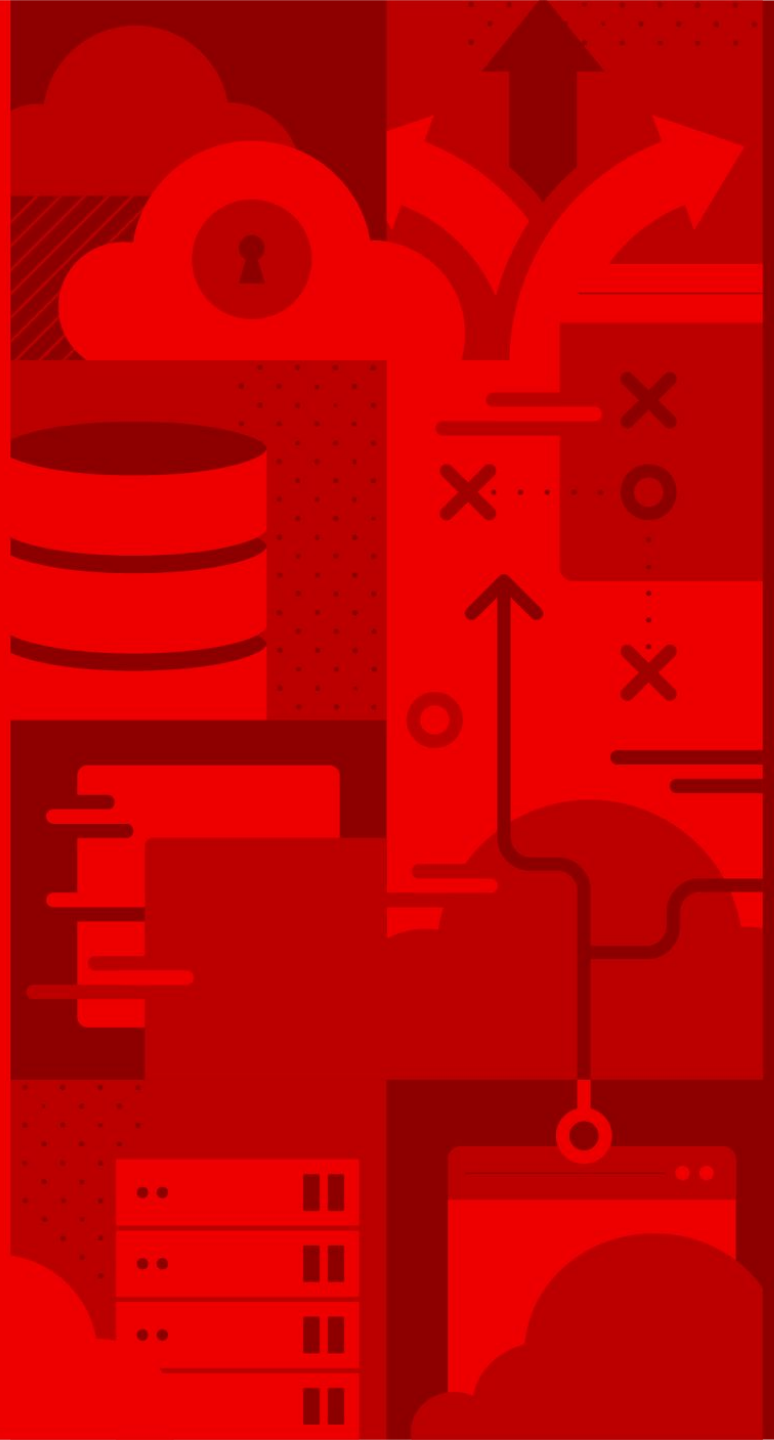


- SVSM boots up from IGVM file
  - Uses proxy to connect to attestation server
    - Sends attestation report
    - Receives key for SVSM state store
  - Unlocks state store
  - Initialize vTPM and UEFI variable service from that
  - Continues boot process and launches OVMF
- OVMF launches OS using SB and measured boot

“How it all works”



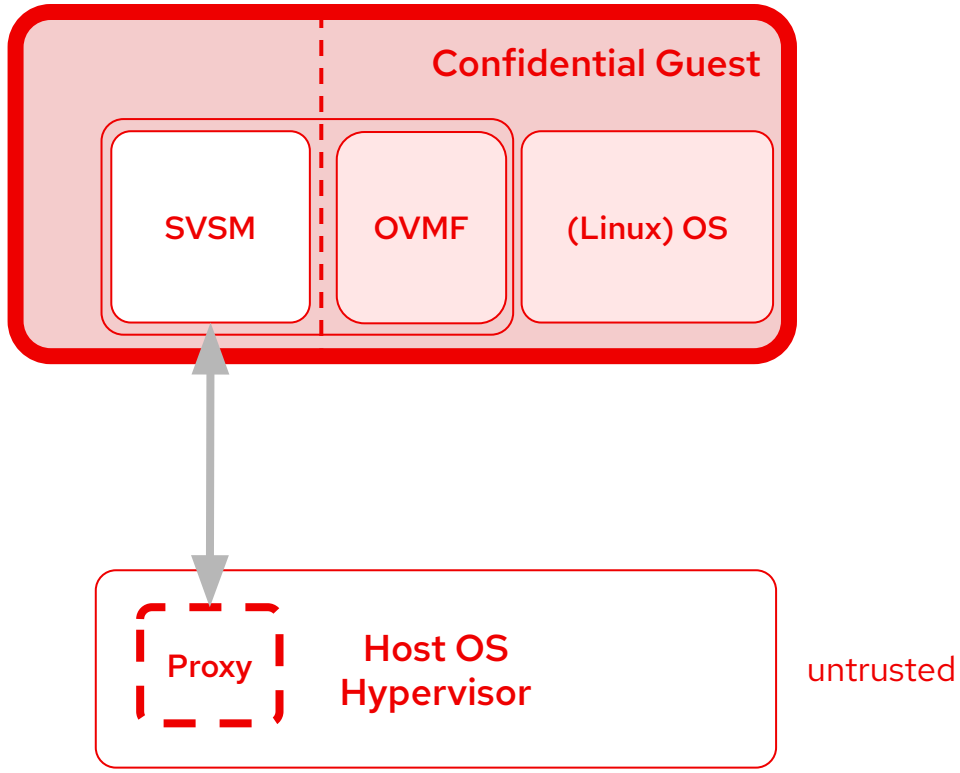
- SVSM boots up from IGVM file
  - Uses proxy to connect to attestation server
    - Sends attestation report
    - Receives key for SVSM state store
  - Unlocks state store
  - Initialize vTPM and UEFI variable service from that
  - Continues boot process and launches OVMF
- OVMF launches OS using SB and measured boot
- OS is able to unlock FDE via TPM’s PCR policy
  - Boot continues



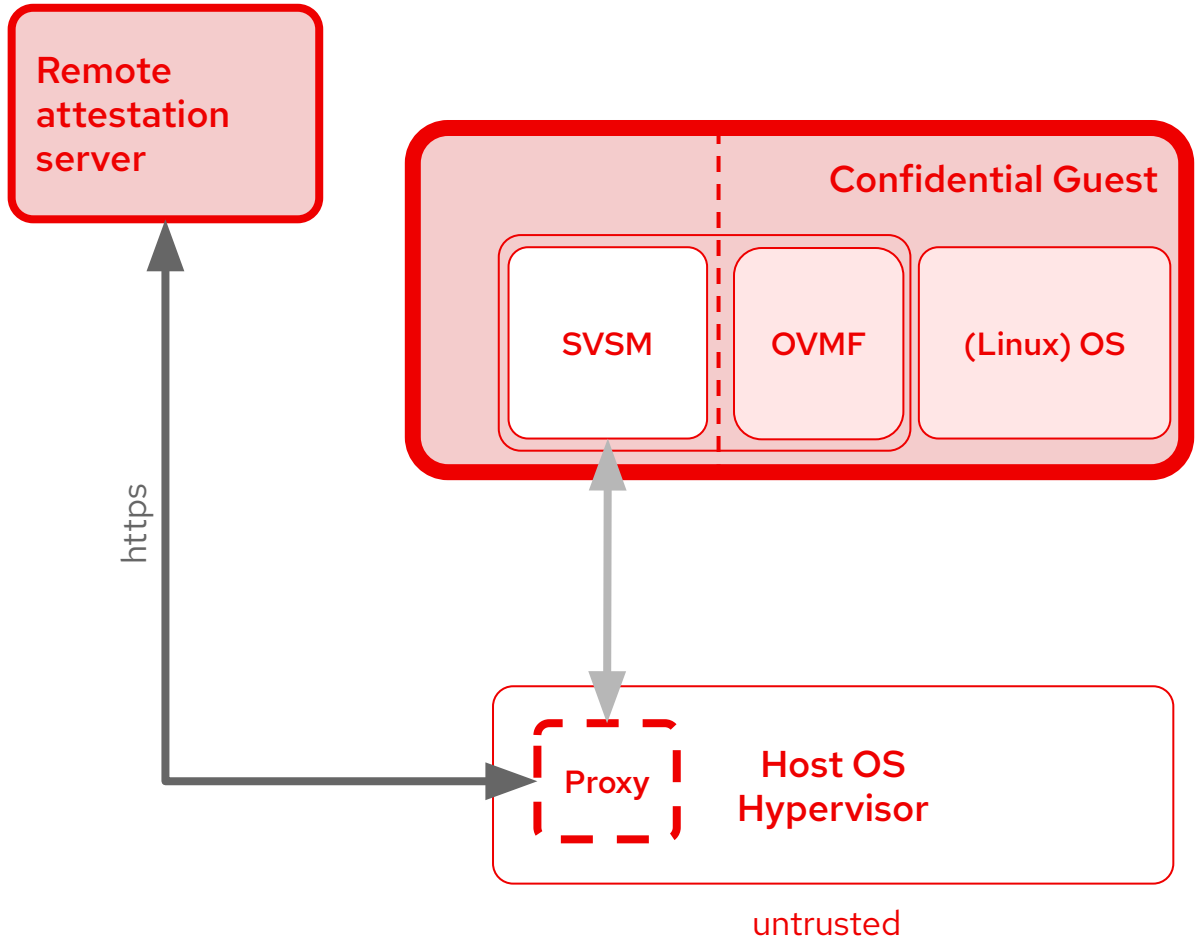
Details & Challenges

host <-> SVSM communication channel

- Communication channel between SVSM and the proxy
  - Device from the hypervisor
    - Virtual ISA serial port
      - easy to support
      - can be slow
    - virtio-vsock over MMIO
      - more code in SVSM
      - virtio-mmio transport reusable for storage
    - Other hypervisor-specific channel
  - Driver in SVSM
  - Communication protocol
    - Coconut SVSM specific protocol since it's not defined by any specification



Remote attestation protocol

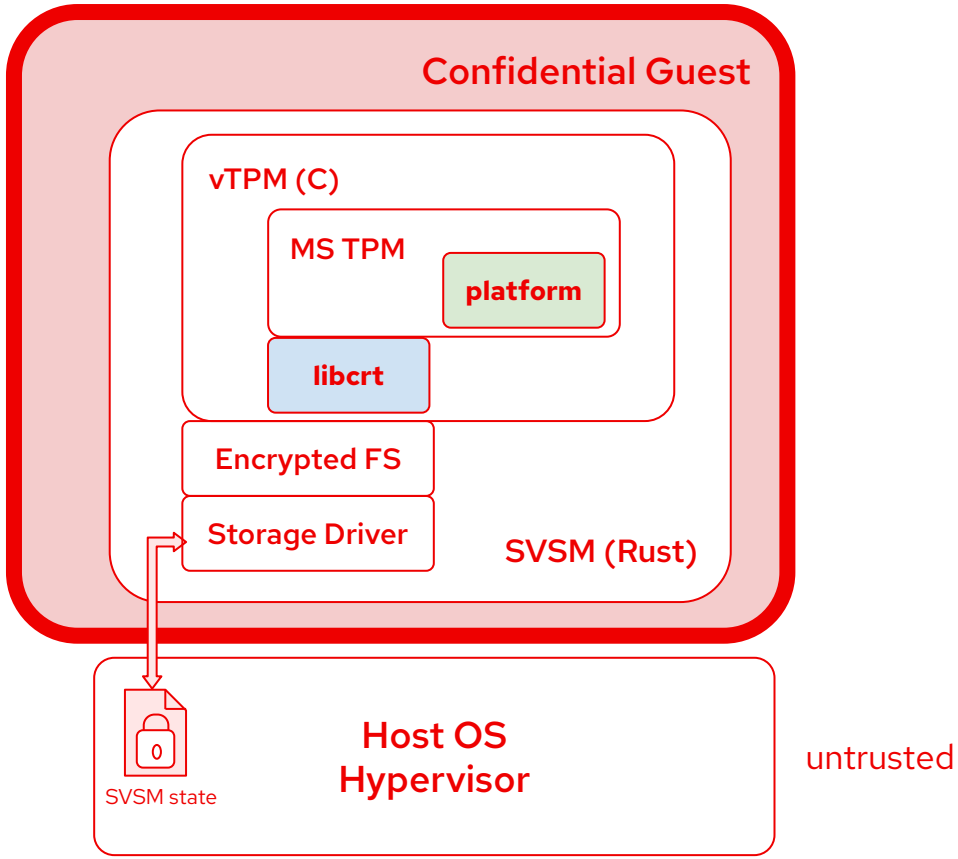


- Multiple attestation protocols
  - Keylime TEE boot attestation
- Where to implement it?
  - Implemented in SVSM
    - pro
      - host proxy and communication protocol very simple
      - SVSM can implement any type http-based protocol
    - cons
      - Supporting a new attestation protocol, we have to re-build SVSM
        - launch measurement will change
  - Implemented in the host proxy
    - pro
      - host can implement any remote attestation protocol without changing SVSM
    - cons
      - communication protocol between SVSM and the proxy can get complicated

Red Hat

MS TPM support for SVSM persistent storage

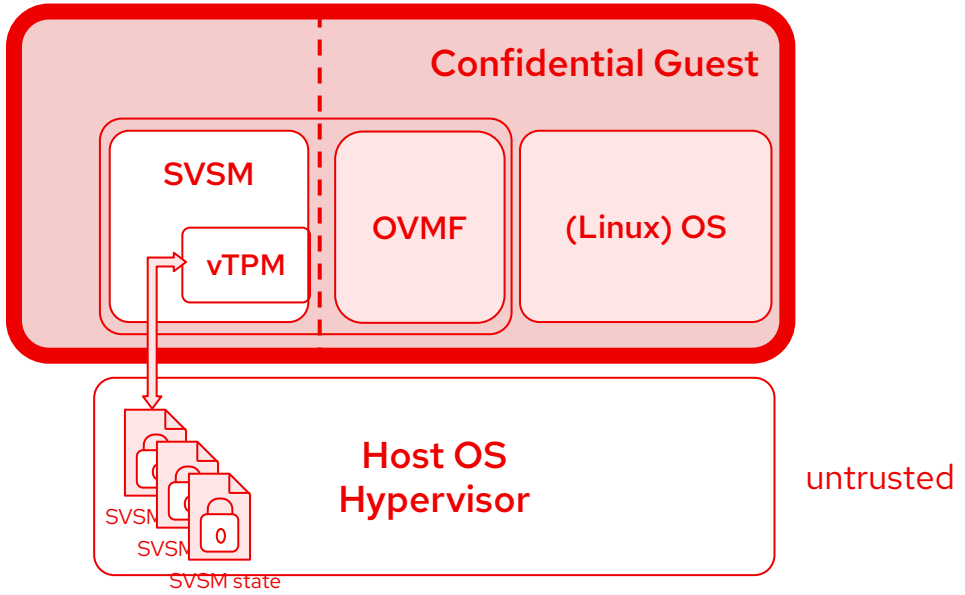
- TPM emulation in SVSM (libmstpm)
  - ms-tpm-20-ref
  - libcrt: C Run-Time (CRT) Library
  - openssl
- How to support SVSM storage API
  - **extending libcrt**
    - implement `fopen/fwrite/fread/etc.` on top of SVSM storage API
    - MS TPM without changes
    - Using simulator code, including storage support
  - **adding a new platform in MS TPM**
    - following `ms-tpm-20-ref/Samples`
    - implement `_plat__NvMemoryRead/_plat__NvMemoryWrite/etc.` on top of SVSM storage API



Red Hat

SVSM state: Rollback and clone attacks mitigation

- Malicious host could perform 2 attacks with the persistent state of SVSM
  - Rollback: reuse an old state
    - TPM monotonic counters could be unreliable
    - SecureBoot updates can be undone
  - Clone: spawn a copy
    - Same TPM identity for different instances
- How to mitigate these attacks
  - Rollback: boot counter
    - Released by remote attestation server
    - Stored in the encrypted SVSM state
  - Clone: only one successful attestation per boot request
    - The user must prime the attestation server before each boot
    - Server will only release the secret once and then wait for the next boot request



Red Hat

SVSM state: storage device

Considered options:

- Qemu pflash (CFI)
  - Used by OVMF for code and nvram
  - Simple
  - Slow: writes limited to 4 bytes at a time

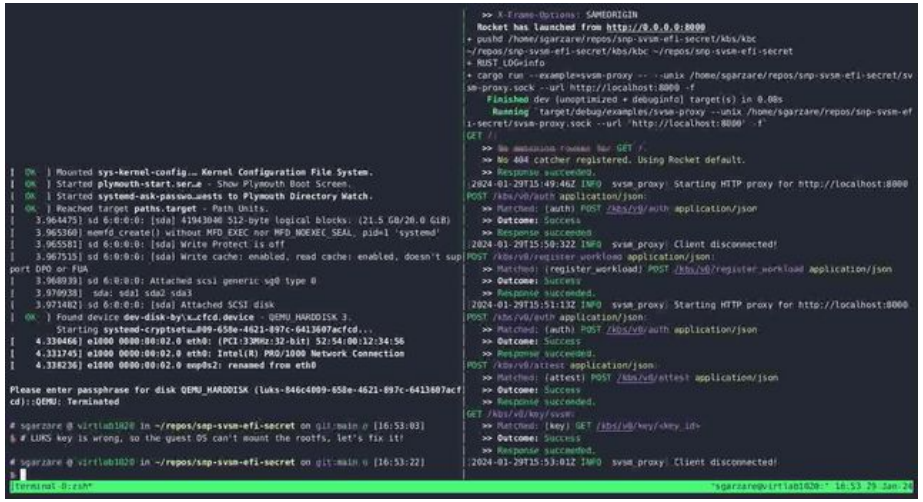
    Might cause delays if full vTPM state is written
- Virtio-blk via MMIO
  - Fast
  - Support for other devices (vsock,...) comes for (almost) free
  - Larger codebase
  - Patch for Qemu: enable virtio-mmio for Q35

- Goal: bock device abstraction layer to allow hypervisor specific implementations
- Or do we need a unified interface for all hypervisors?
- Draft PR: adding rcore-os/virtio-drivers crate
  - Crate is no_std, easy to integrate
  - Host-facing code: Needs review, tests, fuzzing

https://github.com/rcore-os/virtio-drivers/
https://github.com/coconut-svsm/svsm/pull/343

Demo

- Demo available: https://github.com/stefano-garzarella/snp-svsm-vtpm
  - Includes
    - Remote attestation via host proxy
    - Encrypted SVSM persistent state (virtio-blk)
      - Unlocked after successful attestation
    - Loading of MS TPM state from the virtio-blk device
    - Secret sealing/unsealing with TPM's PCR policy



https://red.ht/svsm

Next Steps

- Storage
  - Add a simple filesystem / partitioning scheme
  - Add protections
    - Integrity
    - Rollback
    - Clone
  - Decide on device type
    - Flash device
    - Virtio-blk
    - Something else?
- Remote attestation
  - Implement the attestation Client in the proxy instead of the SVSM?
  - Communication channel:
    Replace serial port with vsock?

Thank you!

Oliver Steffen
<osteffen@redhat.com>

Stefano Garzarella
<sgarzare@redhat.com>

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat