

# COCONUT SVSM

Early attestation to unlock  
persistent state

KVM Forum 2024

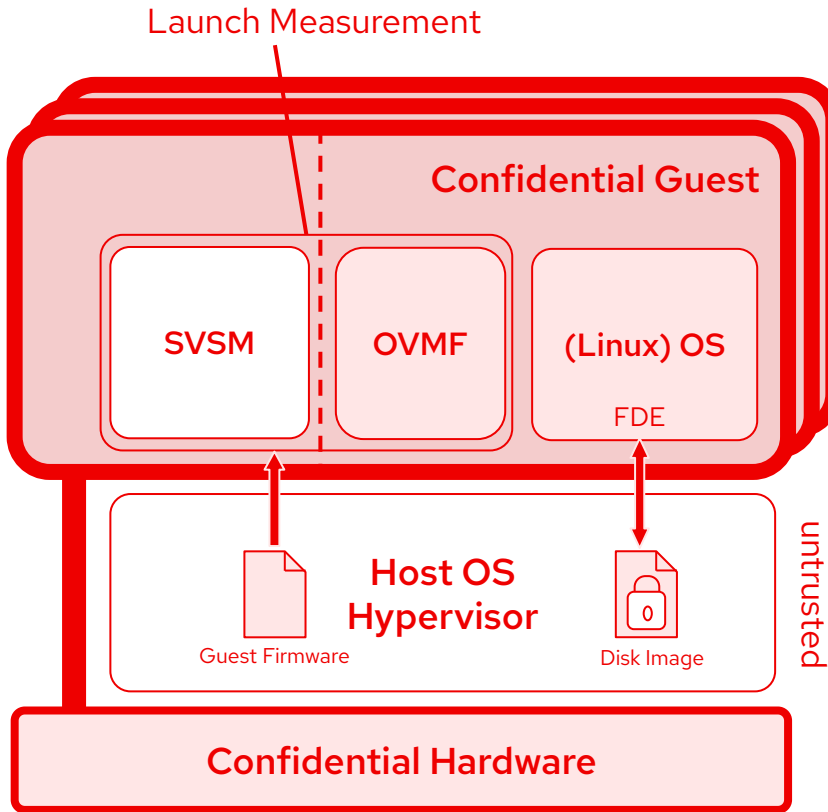
**Oliver Steffen**

<osteffen@redhat.com>

**Stefano Garzarella**

<sgarzare@redhat.com>

## Quick SVSM intro



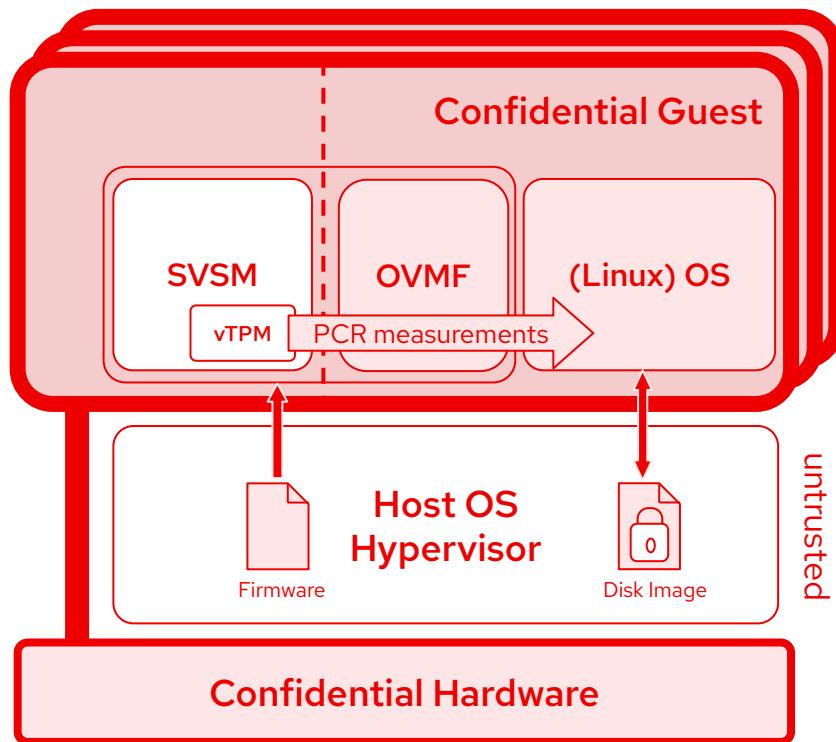
- Confidential Virtualization:
  - Hardware is trusted, Host OS/cloud provider is not trusted
- Hardware guarantees confidentiality of guest memory and CPU state
- Remote Attestation -> confirm that running in valid CVM
- Launch Measurement covers initial state
- Data in use is protected
- Data at rest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

# End User Goals

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

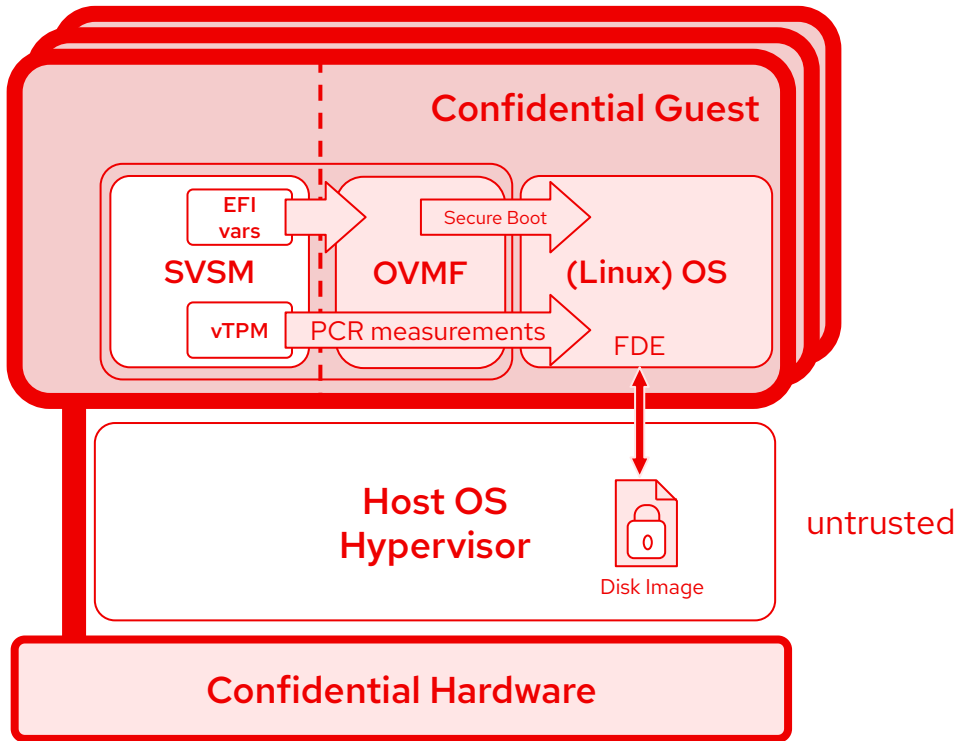
- Easy migration from regular VMs to CVMs
- Support long living guests
  - Stop + Restart
  - Persistent, mutable disk
  - Self-updating OS during runtime
- Guest OS should not have to deal with attestation
- Support measured boot & automatic FDE unlocking via TPM's PCR policy
- Support SecureBoot

## 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

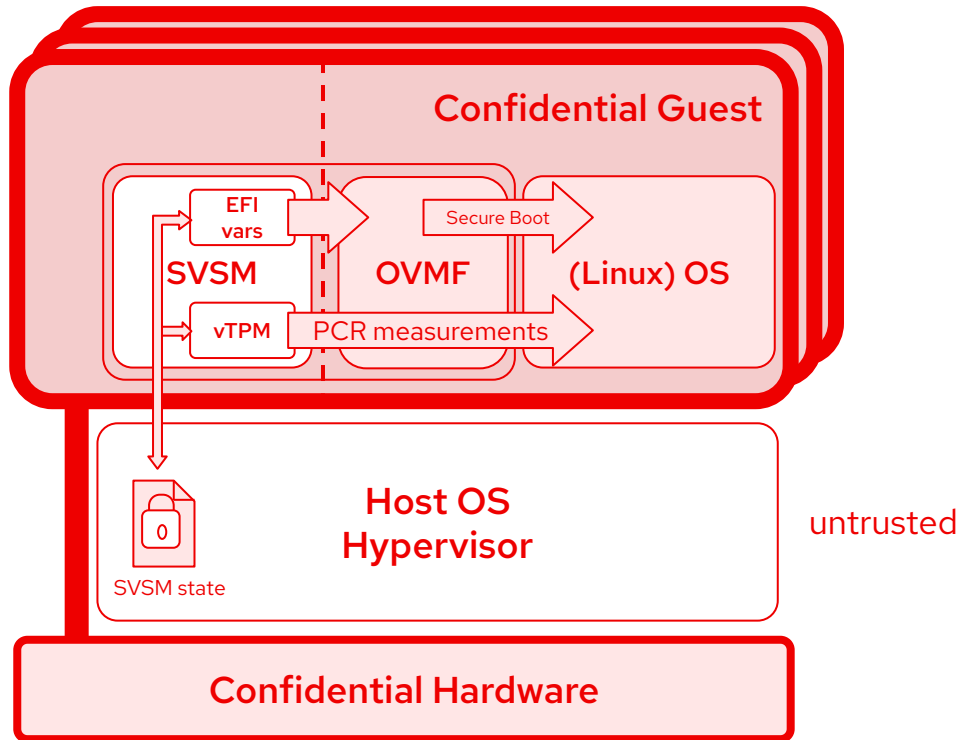


## Goals:

- 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.

## SVSM Persistent State

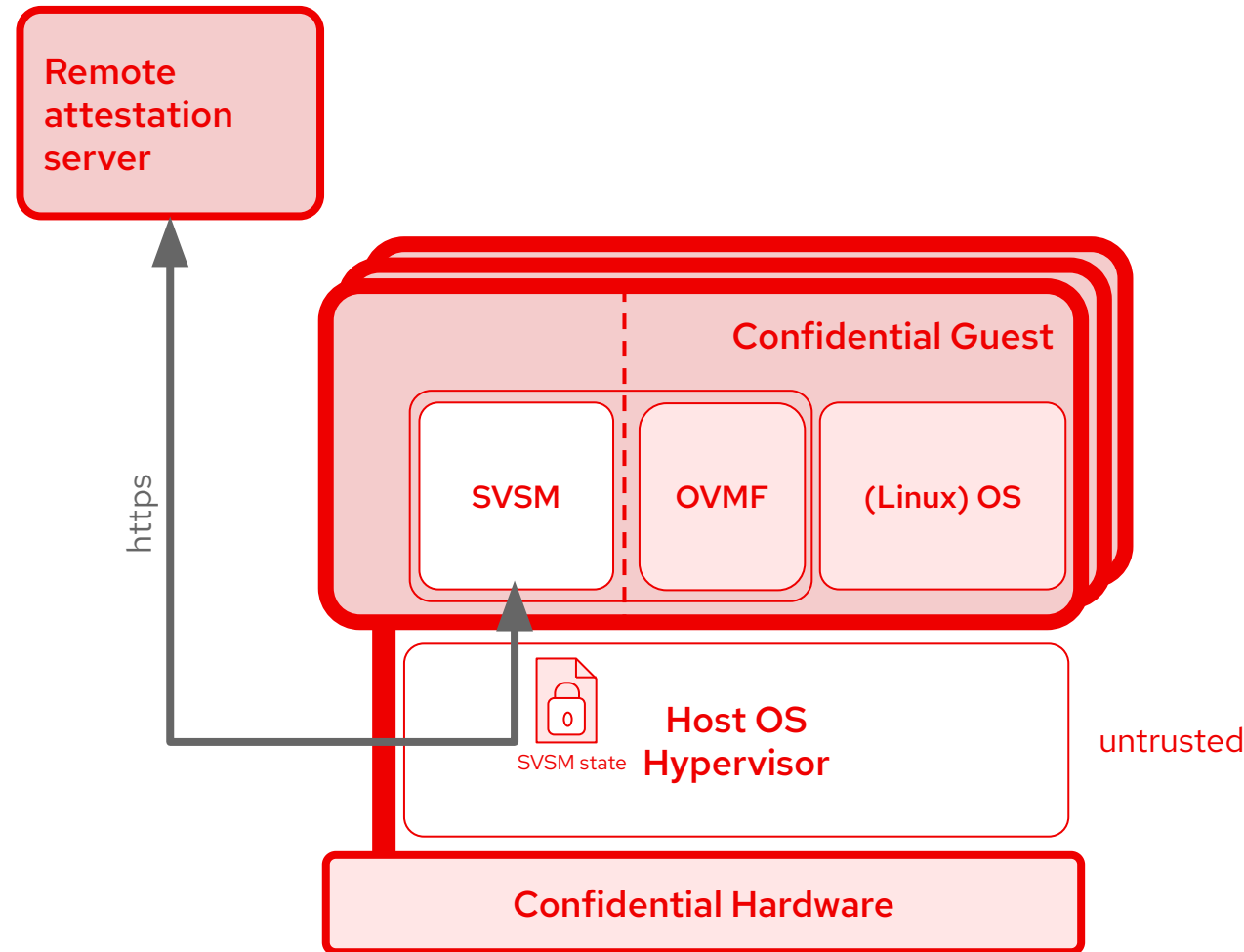


- SVSM State = vTPM state + UEFI variables + ...
- Add a storage driver to SVSM
- Storage backend provided by the host
- Use encryption - the host is not trusted!
- Probably need to support multiple drivers for different hypervisors

**How to decrypt the SVSM state ?**

# Early attestation in SVSM

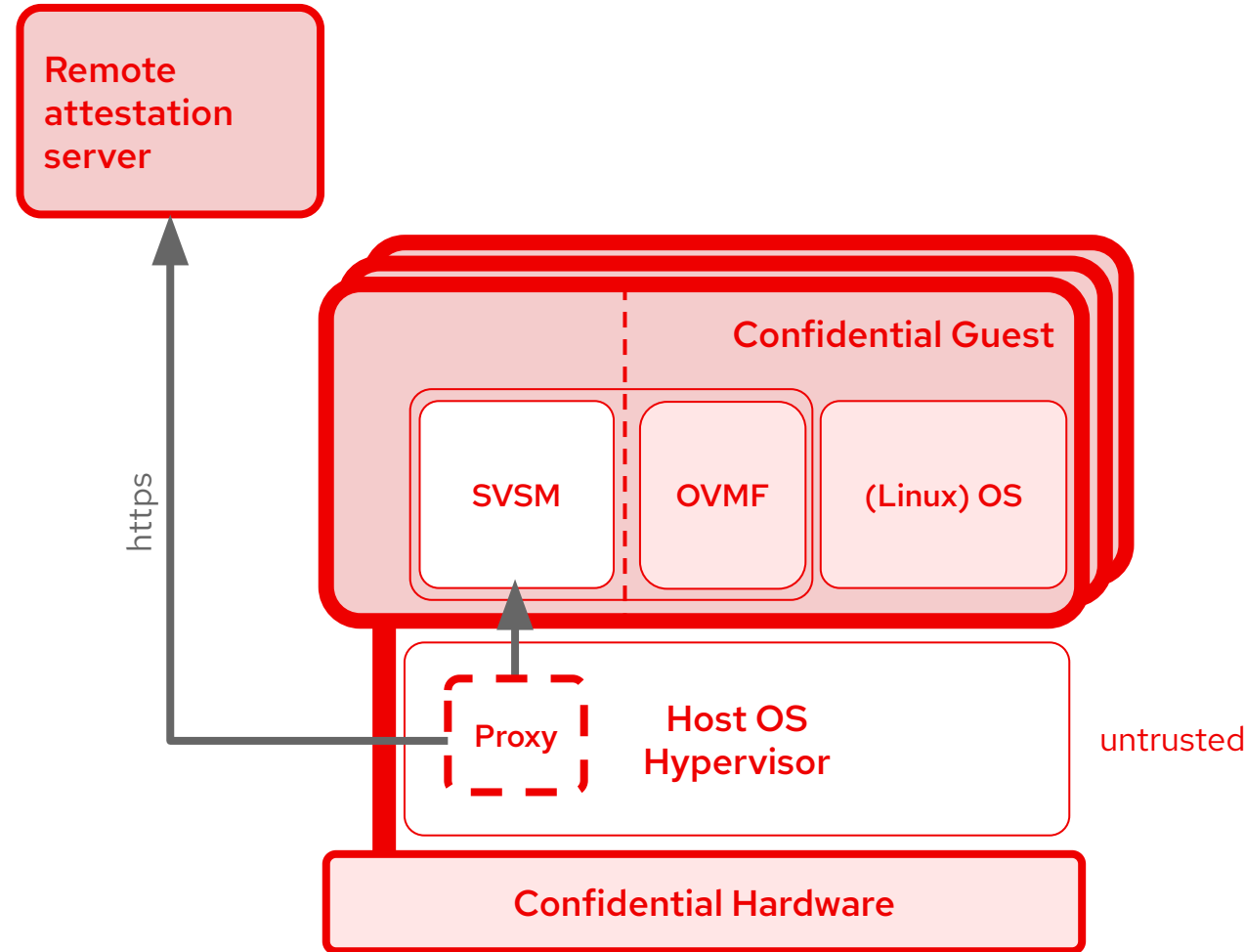
- Encrypted state
  - Unlocked only after a successful remote attestation
    - Early attestation in SVSM is needed
- Remote attestation
  - HW generates an attestation report (evidence)
    - signed by HW's vendor certificate
  - Remote server (trusted) checks the evidence
    - Expected SW running 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



7 SVSM community proposal for early attestation:  
[https://docs.google.com/document/d/11ZsxP8jsviP3ddp9Hrn0rf6inttNw\\_PbnzOpsXlxlPs](https://docs.google.com/document/d/11ZsxP8jsviP3ddp9Hrn0rf6inttNw_PbnzOpsXlxlPs)

# Attestation proxy

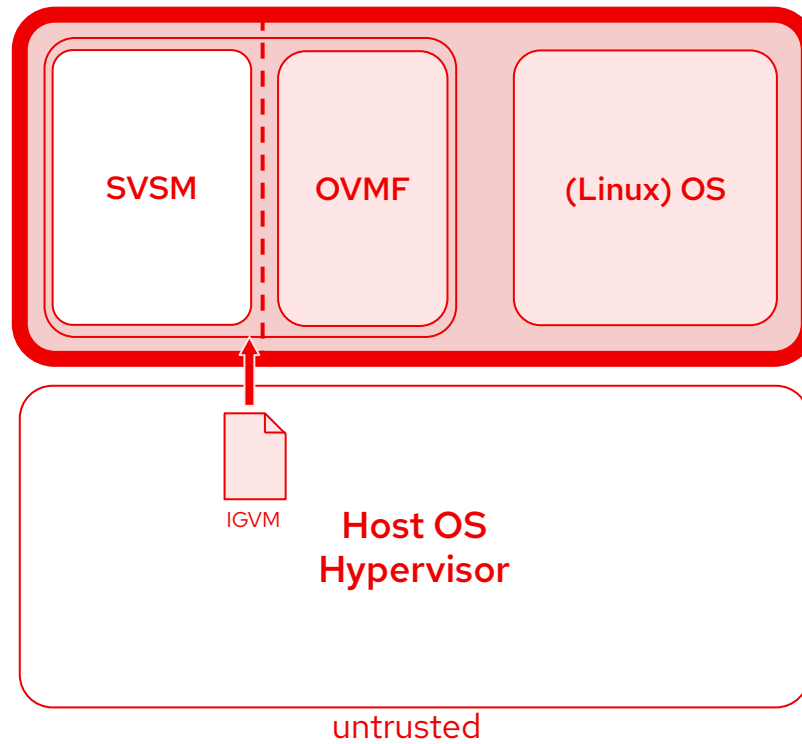
- Proxy application running on the host
  - Allow SVSM to talk to a remote attestation server
  - Forward SVSM request using an https channel
- Challenges
  - Application running on an untrusted host
    - MITM
      - SVSM must generate asymmetric key to receive secrets
    - DoS
      - Out of context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



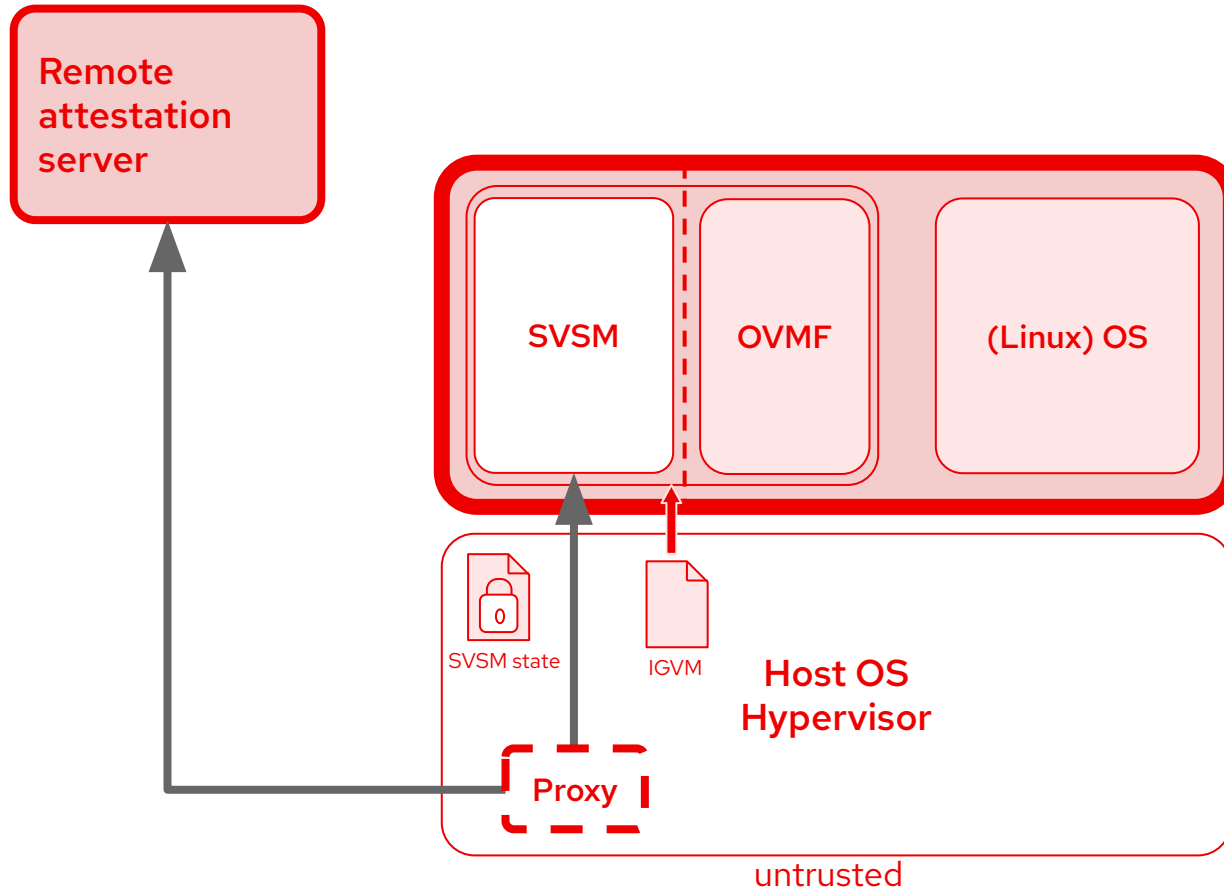


# “How it all works”

- SVSM boots up from IGVM file

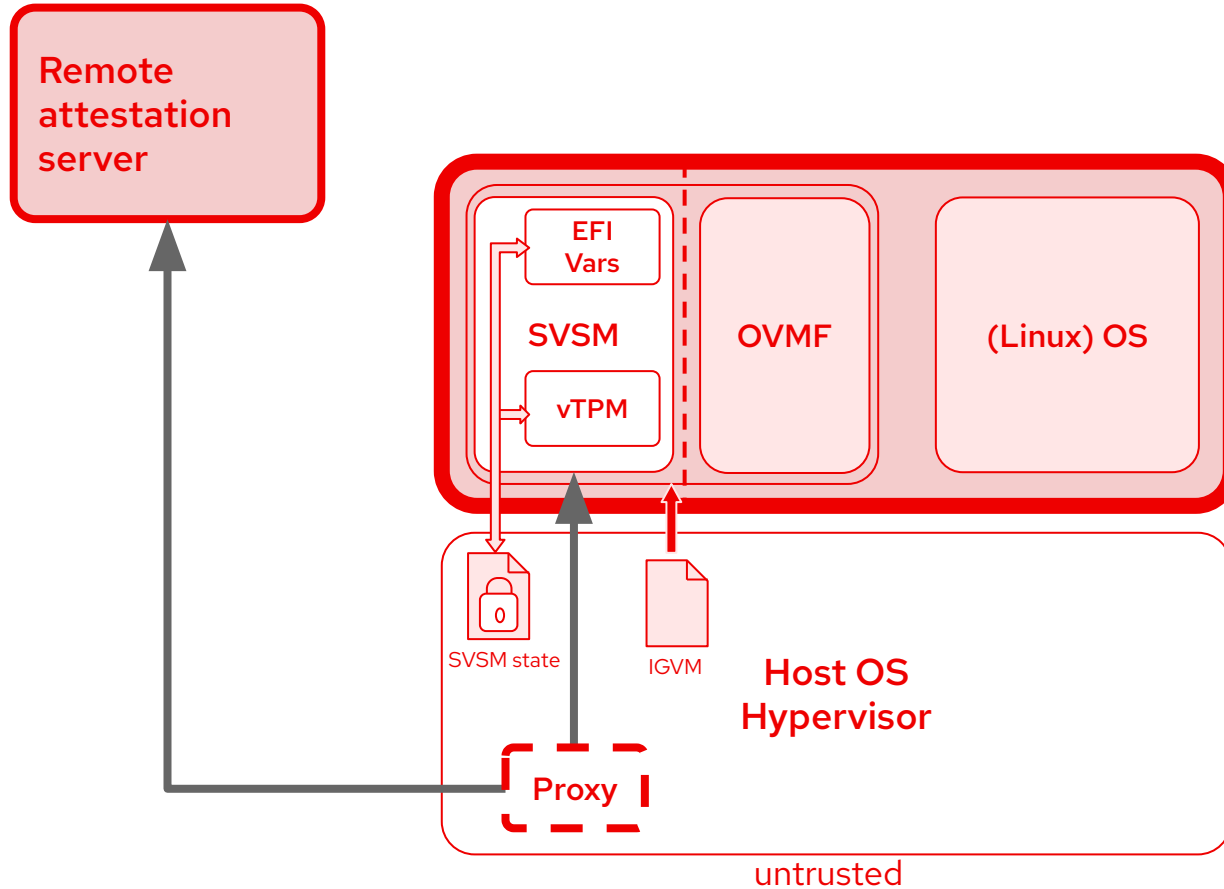


## “How it all works”



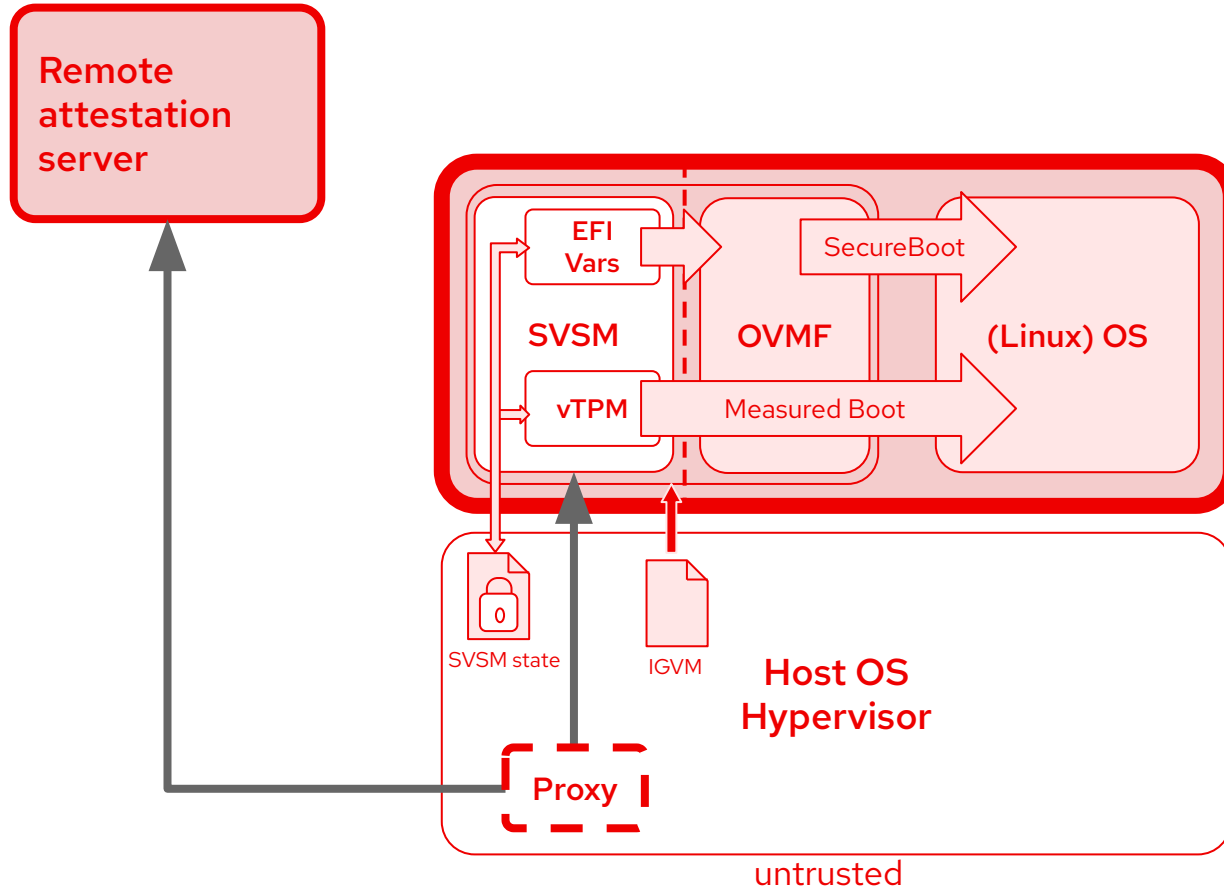
- SVSM boots up from IGVM file
  - Uses proxy to connect to attestation server
    - Sends attestation report
    - Receives key for SVSM state store
  - Unlocks state store

# “How it all works”



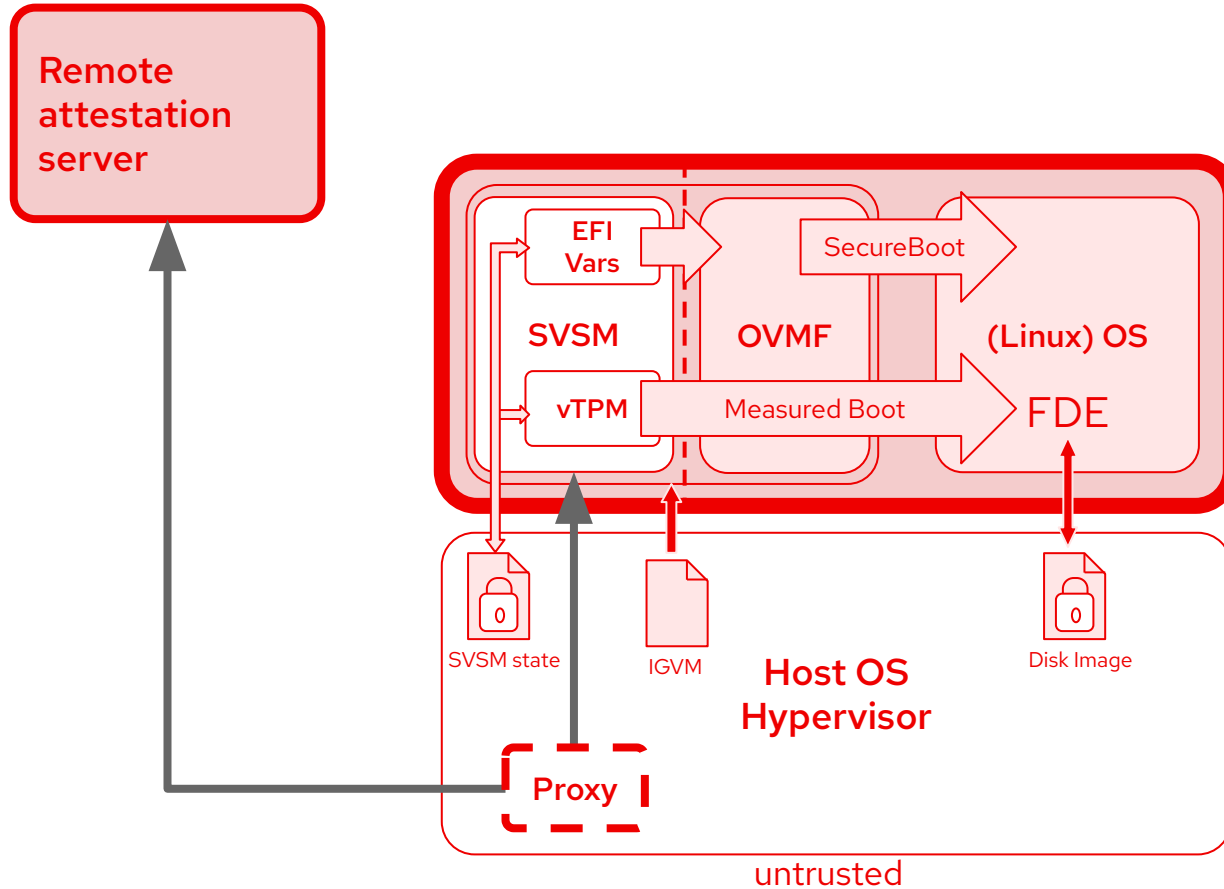
- SVSM boots up from IGVM file
  - Uses proxy to connect to attestation server
    - Sends attestation report
    - Receives key for SVSM state store
  - Unlocks state store
  - Initialize vTPM and UEFI variable service from that
  - Continues boot process and launches OVMF

# “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”

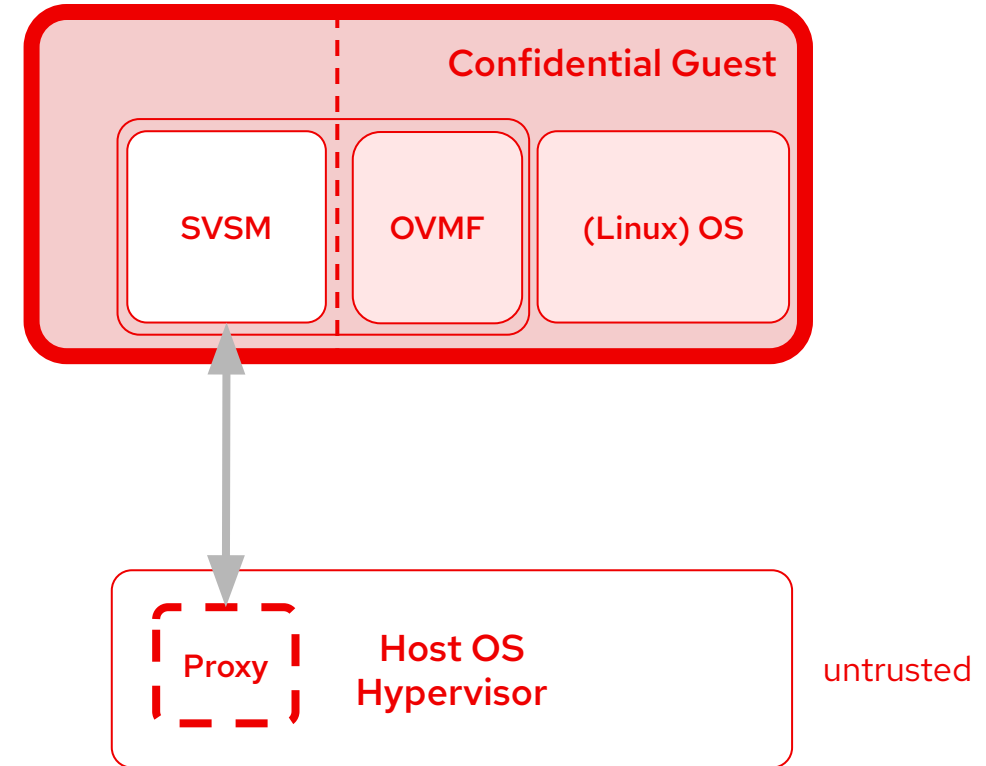


- SVSM boots up from IGVM file
  - Uses proxy to connect to attestation server
    - Sends attestation report
    - Receives key for SVSM state store
  - Unlocks state store
  - Initialize vTPM and UEFI variable service from that
  - Continues boot process and launches OVMF
- OVMF launches OS using SB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

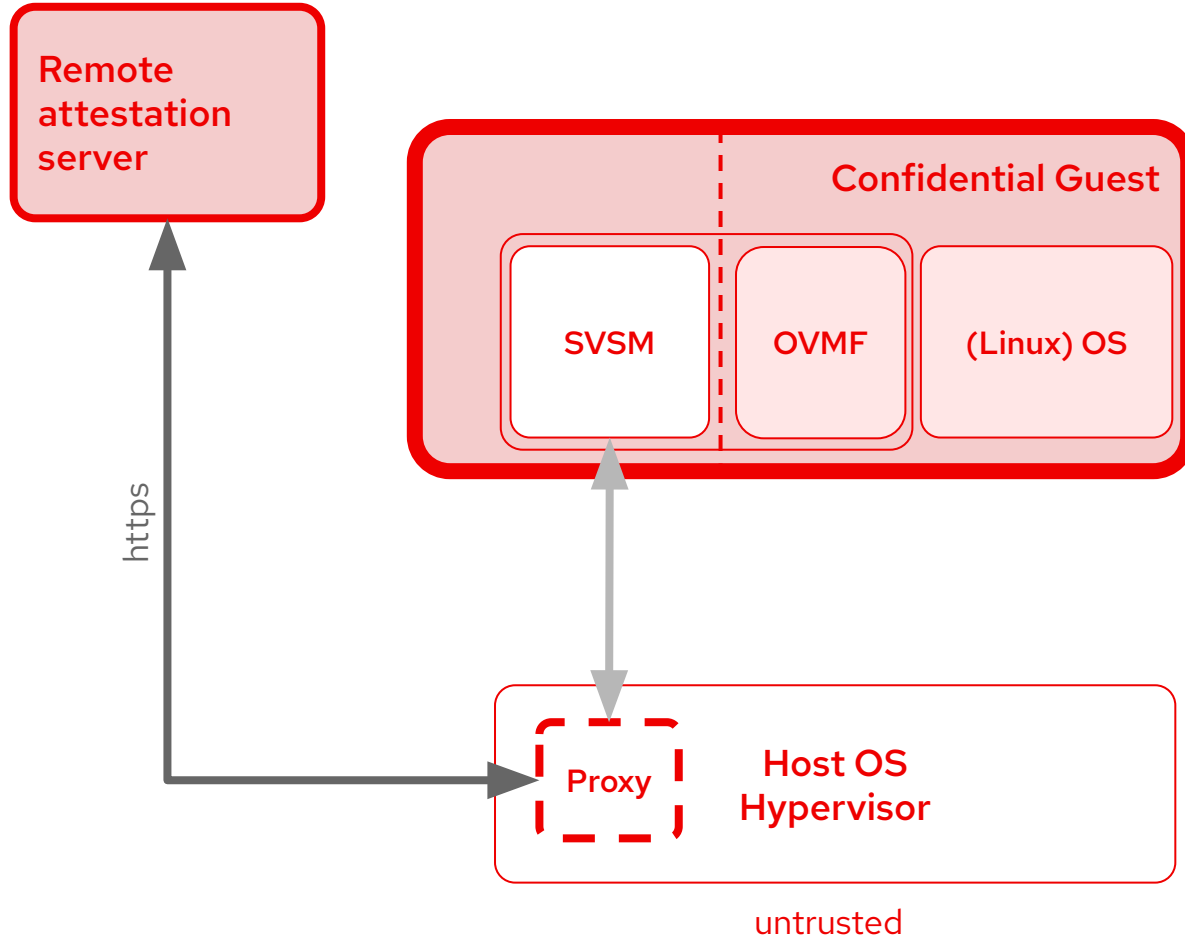
# Details & Challenges

## host $\leftrightarrow$ 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

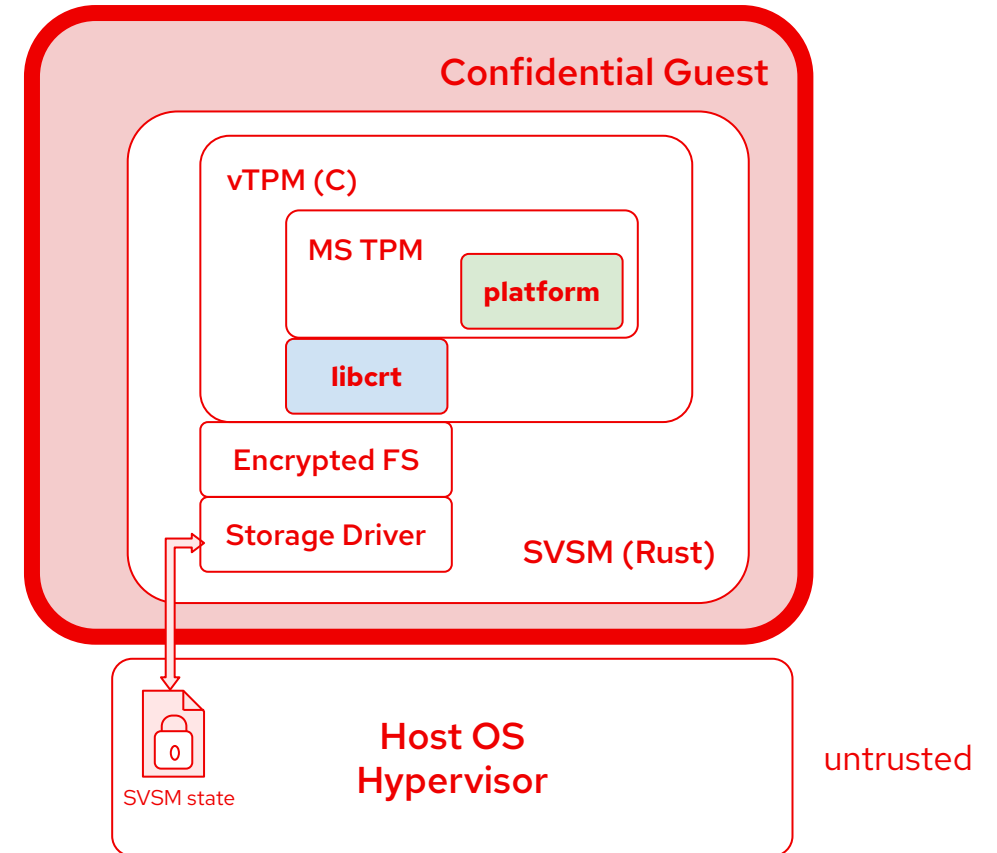


- Multiple attestation protocols
  - [Keylime TEE boot attestation](#)
- Where to implement it?
  - Implemented in SVSM
    - pro
      - host proxy and communication protocol very simple
      - SVSM can implement any type http-based protocol
    - cons
      - Supporting a new attestation protocol, 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



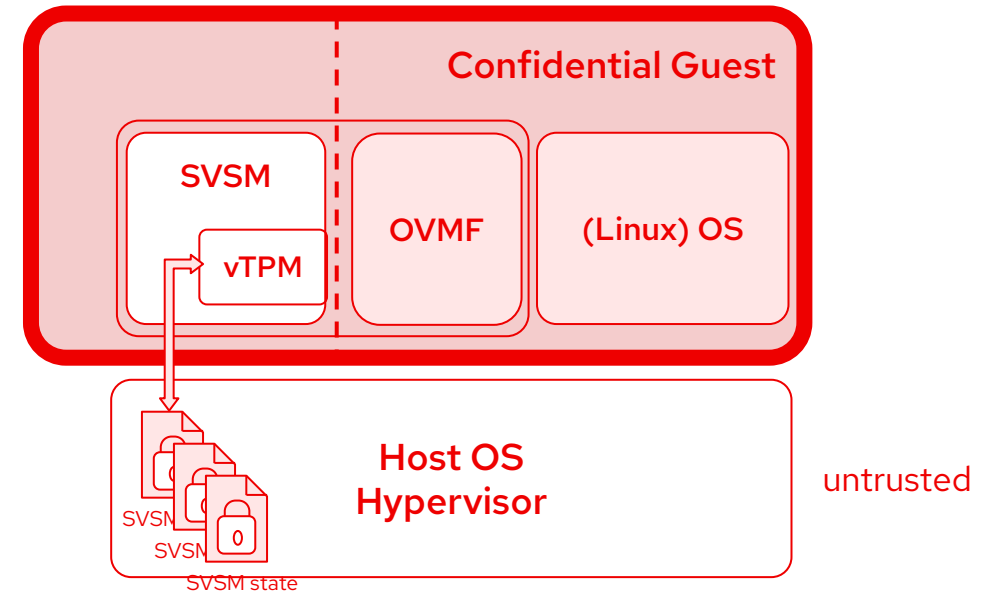
# MS TPM support for SVSM persistent storage

- 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



# 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



## 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: back 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>

```
[ OK ] Mounted sys-kernel-config... Kernel Configuration File System.
[ OK ] Started plymouth-start.service - Show Plymouth Boot Screen.
[ OK ] Started systemd-ask-passwords to Plymouth Directory Watch.
[ OK ] Reached target paths.target - Path Units.
[ 3.964473] sd 0:0:0:0: [sd] 419430M 512-byte logical blocks: (21.5 GiB/26.0 GiB)
[ 3.965368] memfd create() without MFD_EXEC nor MFD_NOEXEC_SEAL, pid=1 'systemd'
[ 3.965581] sd 0:0:0:0: [sd] Write Protect is off
[ 3.967512] sd 0:0:0:0: [sd] write cache: enabled, read cache: enabled, doesn't support DPO or FUA
[ 3.968933] sd 0:0:0:0: Attached scsi generic sg0 type 0
[ 3.970938] sda: sda1 sda2 sda3
[ 3.972402] sd 0:0:0:0: [sd] Attached SCSI disk
[ OK ] Found device dev-disk-by-label-fc0 device - QEMU HARDDISK 3.
Starting system-cryptsetup_899-658e-4621-897c-6413697acfd...
[ 4.338466] e1000 0000:00:02:0: eth0: [PCI:330Hz:32-bit] 52:54:00:12:34:56
[ 4.332745] e1000 0000:00:02:0: eth0: Intel(R) PRO/1000 Network Connection
[ 4.338236] e1000 0000:00:02:0: vepo2: renamed from eth0

Please enter passphrase for disk QEMU_HARDDISK (luks-846c4099-658e-4621-897c-6413697acfd): QEMU: Terminated

# sgarzare @ virtlab1028 in ~/repos/snp-svsm-efi-secret on git:main @ [16:53:03]
$ # LIKS key is wrong, so the guest OS can't mount the rootfs, let's fix it!
# sgarzare @ virtlab1028 in ~/repos/snp-svsm-efi-secret on git:main @ [16:53:22]

# From: options - SAMEDR103H
Rocket has launched from http://0.0.0.0:8000
+ pushd /home/sgarzarella/repos/snp-svsm-efi-secret/keys/abc
~/repos/snp-svsm-efi-secret/keys/abc ~~/repos/snp-svsm-efi-secret
+ HOST_ID=$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 10 | xargs | sha256sum | cut -d ' ' -f 1)
+ cargo run --example=svsm-proxy -- --unix /home/sgarzarella/repos/snp-svsm-efi-secret/svsm-proxy.sock --url http://localhost:8000 -f
Flashed dev (unoptimized + debuginfo) target(s) in 0.08s
Running target/debug/examples/svsm-proxy --unix /home/sgarzarella/repos/snp-svsm-efi-secret/svsm-proxy.sock --url http://localhost:8000 -f
GET /
>>> No sessions found for GET /
>>> AME catcher registered. Using Rocket default.
>>> Response succeeded.
2024-01-29T15:49:46Z INFO svsm_proxy: Starting HTTP proxy for http://localhost:8000
POST /keys/auth/application/json
>>> Matched: [auth] POST /keys/auth/application/json
>>> Outcome: Success
>>> Response succeeded.
2024-01-29T15:50:32Z INFO svsm_proxy: Client disconnected!
POST /keys/register-workload/application/json
>>> Matched: [auth] POST /keys/register-workload/application/json
>>> Outcome: Success
>>> Response succeeded.
2024-01-29T15:51:22Z INFO svsm_proxy: Starting HTTP proxy for http://localhost:8000
POST /keys/auth/application/json
>>> Matched: [auth] POST /keys/auth/application/json
>>> Outcome: Success
>>> Response succeeded.
POST /keys/attest/application/json
>>> Matched: [attest] POST /keys/attest/application/json
>>> Outcome: Success
>>> Response succeeded.
GET /keys/keys/keys
>>> Matched: [key] GET /keys/keys/keys
>>> Outcome: Success
>>> Response succeeded.
2024-01-29T15:53:01Z INFO svsm_proxy: Client disconnected!
```

# 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](https://www.linkedin.com/company/red-hat)

 [facebook.com/redhatinc](https://www.facebook.com/redhatinc)

 [youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)

 [twitter.com/RedHat](https://twitter.com/RedHat)