Red Hat Enterprise Linux

Introducing F-UKI, Guest Firmware in a UKI for Confidential Cloud Deployments

Anirban (Ani) Sinha
Principal Software Engineer, Virtualization, Red Hat.
anisinha@redhat.com.

FOSDEM 2025, Brussels, Belgium Feb 1-2 2025

Red Hat

People who are involved

- Ani Sinha (Red Hat).
- Vitaly Kuznetsov (Red Hat).
- Alex Graf (AWS) (original idea).
- Paolo Bonzini (Red Hat).
- Gerd Hoffman (Red Hat).
- Harald Hoyer (Matter Labs).

Huge thanks  to Lennart Poettering (speaking here at FOSDEM) for helping us with UKI parts!

Alex and Vitaly are also speaking at *FOSDEM* this year! Please find them around!

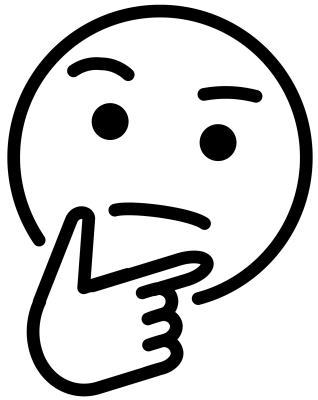
*F-UKI : UKI with Firmware, termed coined by Alex :-)

Focus of the talk

- Background
- What?
- Why?
- How?
- Using UKI to bring in firmware
- Current Status
- References

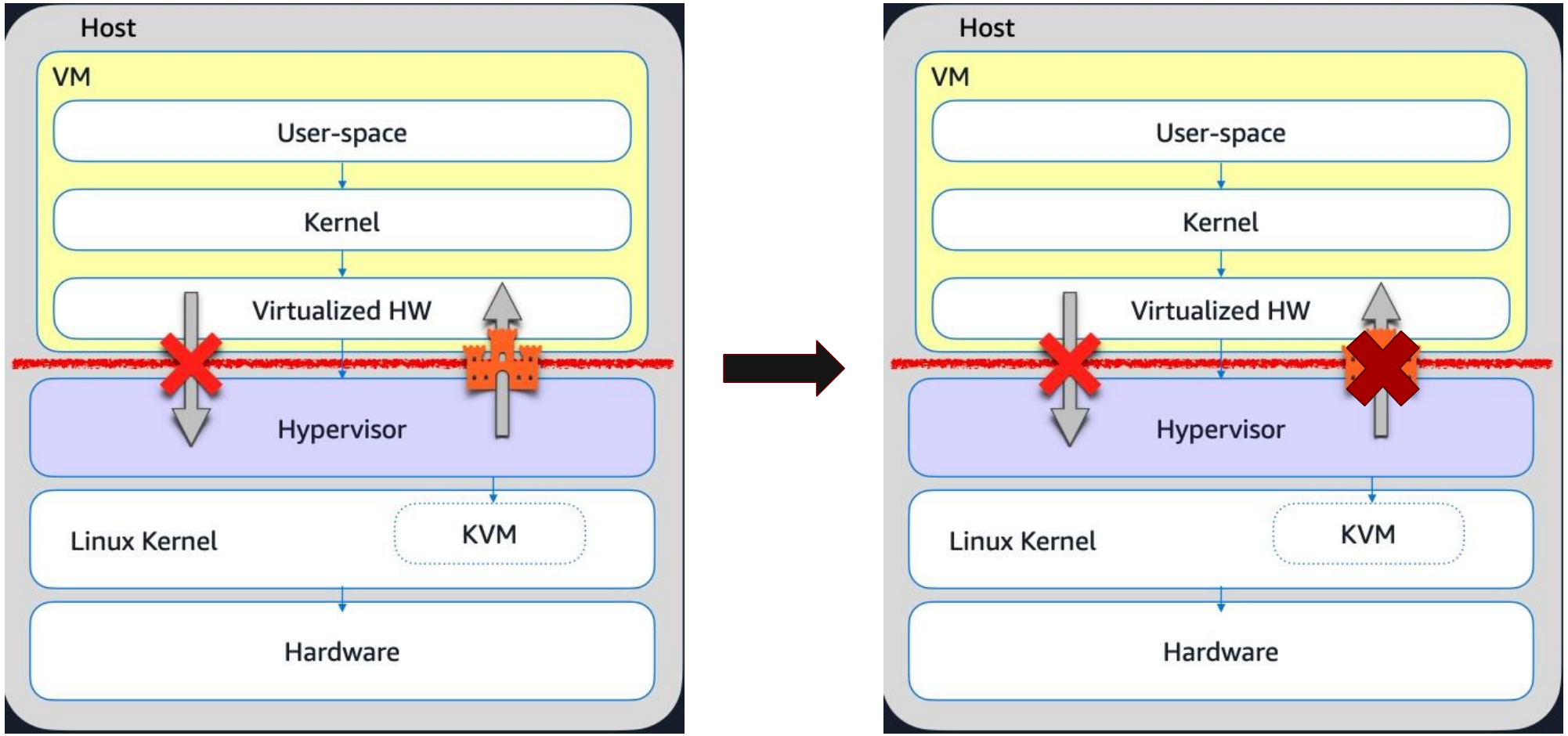
Background

- What?
- Why?
- How?
- Using UKI to bring in firmware
- Current Status
- References



Taken from Nicolas Saenz Julienne's KVM Forum 2024 presentation

So what is a Confidential VM?



Red Hat

So what is a Confidential VM?

Confidential VM provides protection from the host it runs on:

- "Protection" means strong security boundary for all **data** in the VM. Malicious hypervisor or an actor having access (even privileged!) to the host should not be able to get access to the data.
- The host is still able to disrupt execution of the VM, e.g. it can stop it.
- Hardware (AMD SEV-SNP, Intel TDX) is responsible for encrypting memory and CPU state.
- Storage encryption is necessary for security and must be done by the guest OS.

Verifying integrity at start - measured boot ... (1)

Measured Boot:

- It is the process of computing and securely recording hashes of code and critical data at each stage in the boot chain before the code/data is used.
- No validation. Simply a record of what code/critical-data was present on the system during boot.
- Platform Configuration Registers in TPM store the incremental hashes/measurements securely (can not be modified once the ***update*** operation has completed on the newly ***extended*** measured hash) that become part of the final launch digest.
- Intel TDX module provides measurement capability in HW (no need for TPM registers).
- The measurements can also be stored securely in memory.

Verifying integrity at start - measured boot ... (2)

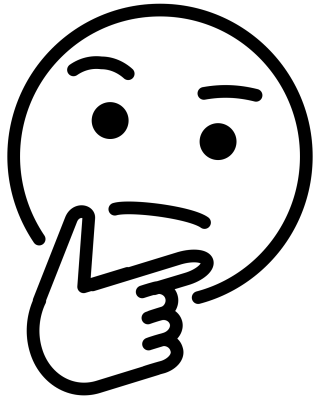
Measured Boot:

- **Firmware images must be measured.**
- Final launch digest is then sent to a remote attestation server that validates the digest and send keys to unlock secrets within the guest OS.
- The guest OS can then decrypt the disk with the secrets.

Our work focuses mostly on measured boot.

Agenda

- Background
- **What?**
- Why?
- How?
- Using UKI to bring in firmware
- Current Status
- References



Red Hat

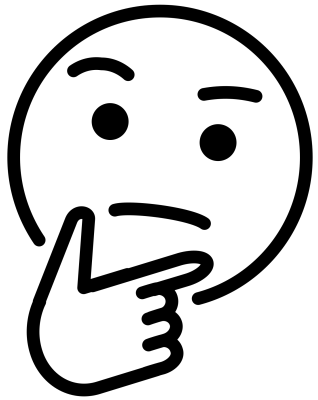
KVM Forum 2024 presentation

The talk is focused on in-guest firmware update mechanism

- The focus is on the situation when host administrator != guest tenant (e.g. ‘cloud’ use-case).
- The mechanism is mostly useful for Confidential VMs but in theory can work with traditional VMs too.

Agenda

- Background
- What?
- **Why?**
- How?
- Using UKI to bring in firmware
- Current Status
- References



KVM Forum 2024 presentation

Why guest tenants are interested in supplying their own firmware?

- Getting predictable, pre-calculated launch measurements
- Implementing exclusive per-guest features and configurations
  - Bring-your-own SecureBoot everything (and **trust** it!)
  - A stateful vTPM implementation with runtime attestations
  - ...
- and this is updateable during guest's lifecycle, not only upon creation.

KVM Forum 2024 presentation

Why host owners are interested in providing the option to do in-guest firmware update?

- For Confidential VMs, updating firmware can't go unnoticed by the guest tenant
  - Getting rid of the responsibility for non-trivial software which runs **inside** the confidential guest.
  - Guests may break because of new, unexpected launch measurements.
  - Tenants may be interested in what changed and in case of e.g. embargoed CVEs the information cannot be shared.

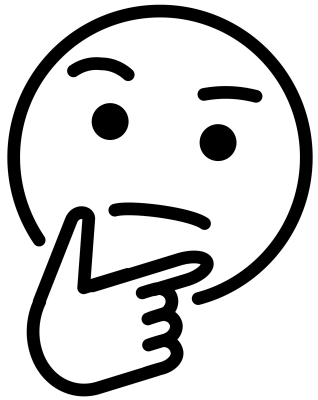
KVM Forum 2024 presentation

But why not just supply the firmware externally, as part of guest VM image or separately?

- The firmware will require a storage if supplied separately and this external storage will have to be linked to the VM's lifecycle.
- Storing the firmware as part of guest image (e.g. a file on ESP, separate partition,...) can be problematic:
  - The host may not have access to guest storage at all (e.g. NVME passthrough with acceleration card).

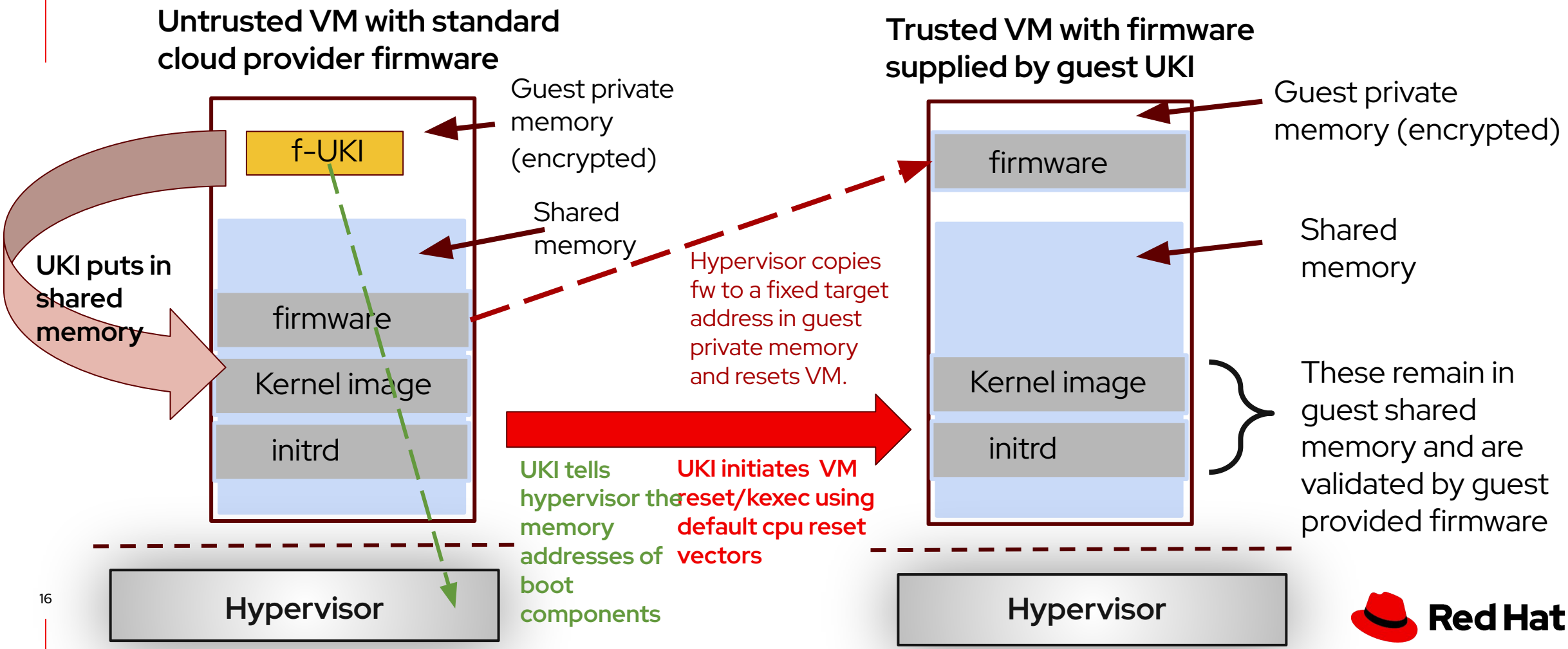
Agenda

- Background
- What?
- Why?
- **How?**
- Using UKI to bring in firmware
- References



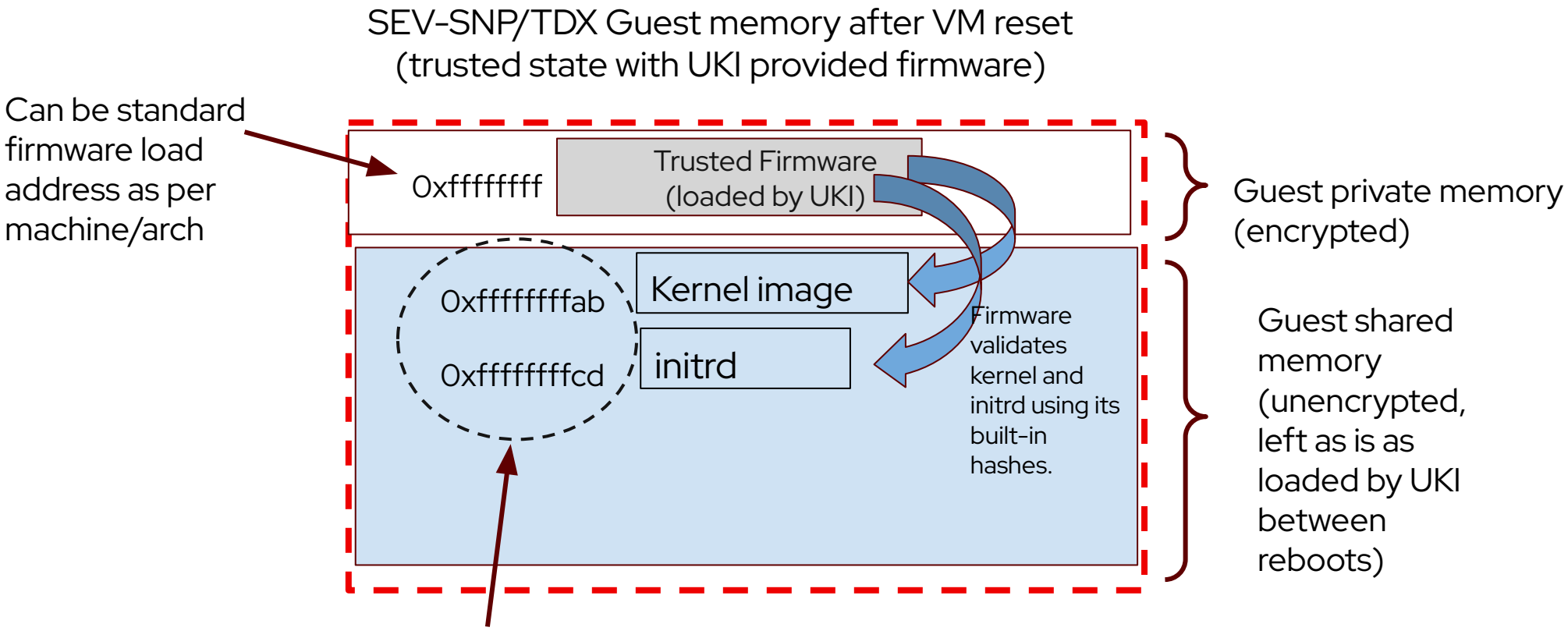
Red Hat

Firmware update mechanism using UKI … (1)



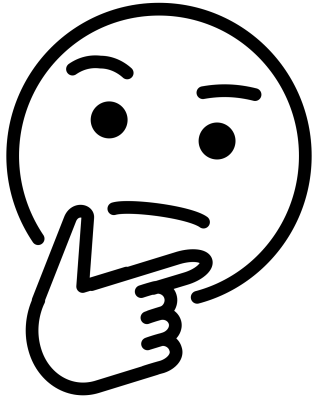
Red Hat

Firmware update mechanism using UKI … (2)



Red Hat

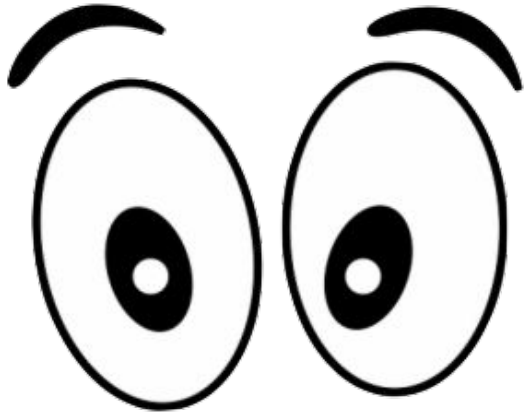
- Background
- What?
- Why?
- How?
- **Using UKI to bring in firmware**
- Current Status
- References



But why UKI for firmware? Or Why f-UKI?

- Portable executable that can be executed by UEFI bios in a guest agnostic manner.
  - Its already widely used in various distributions to load kernel, initrd etc.
- UKI add-on allows customizations - for example separate initrd or kernel command line for separate systems based on use-case.
- UKIs can be signed and UEFI bios can only load signed UKIs and UKI add-ons from trusted vendors.
- UKIs can be obsoleted/revoked using SBAT mechanism.
- UKIs containing updated kernel/initrd etc can be installed using standard package management tools.
- All of the above nicely applies for updating firmware images.

So lets build a f-UKI ...



```
$ ukify.py --help | grep efifw
  --efifw DIR           Directory with efi firmware binary file [.efifw section]
```

```
$ ukify.py build --efifw=/home/anisinha/ovmf-x86-smm-secure --output=/tmp/test.uki
```

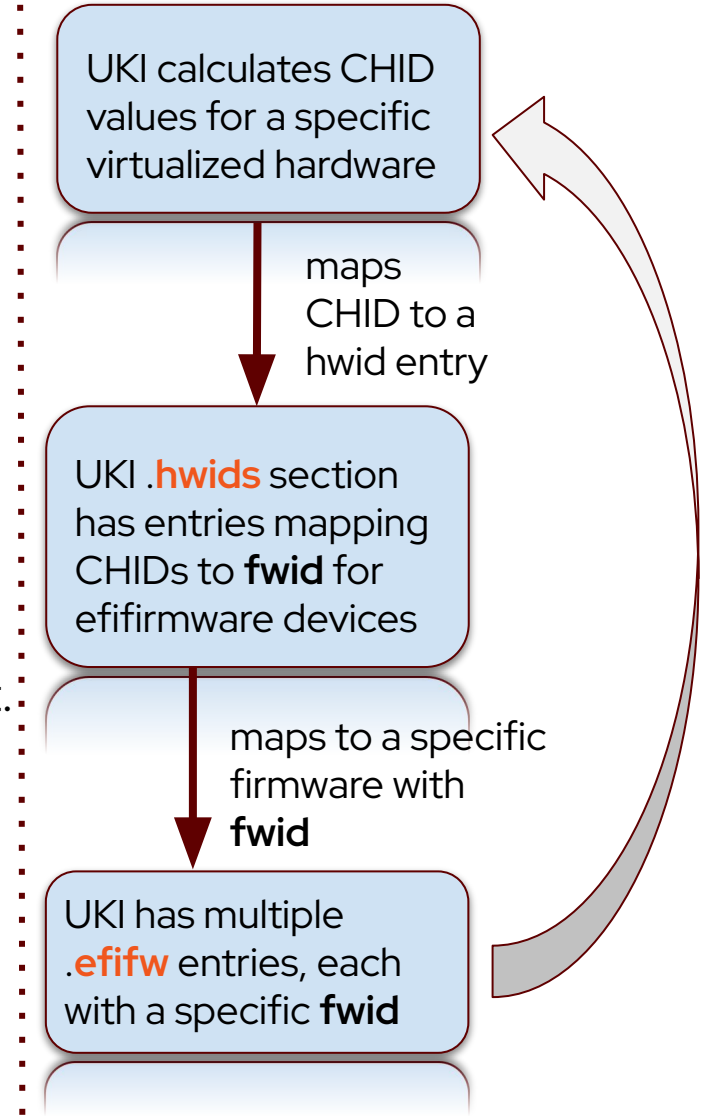
The name of the directory is used as an unique identifier for the firmware image

Directory containing a single ovmf firmware image

```
$ ukify.py build --efifw=/home/anisinha/ovmf-x86 --efifw=/workspace/firmware/ovmf-aarch64-smm
--output=/tmp/test.uki
```

f-UKI bits ... (1)

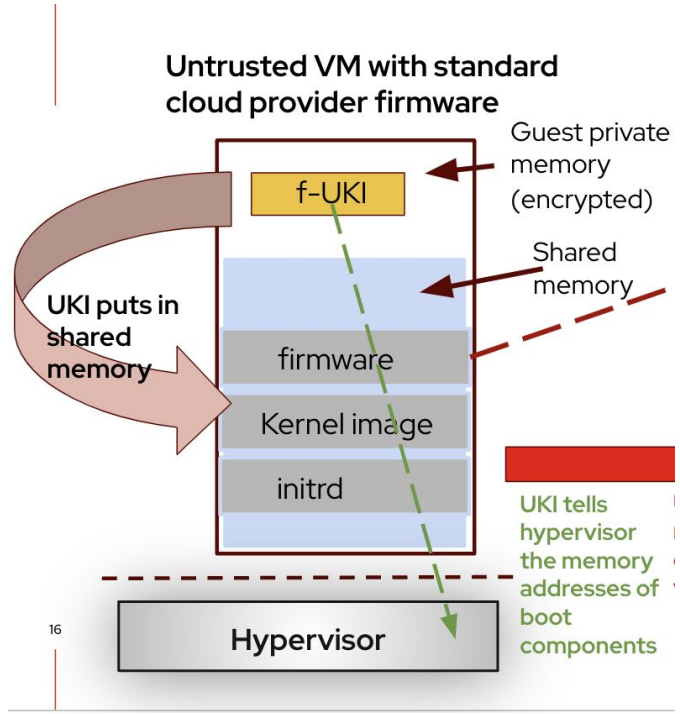
- Multiple firmware images can be bundled in the same UKI.
  - UKI has a mechanism to load the correct one that is compatible with the current hardware.
  - Calculates the "*computer hardware IDs*" (CHID) for the present hardware and matches a particular firmware (or say devicetree) with it.
- The selected firmware can then load **trusted** kernel/initrd and command line whose hashes are known to the firmware as trustworthy.
  - When building the UKI, *ukify* must install the hashes in the firmware or they may be installed by a different tool and packaged together later using *ukify*.



Red Hat

f-UKI bits ... (2)

- Through a well known hypervisor interface (see green line in diagram), the UKI then convey the hypervisor the memory locations where these boot components (firmware, kernel, initrd, command line etc) are loaded.
- UKI initiates reset of the guest to activate the packaged firmware and other boot components.
- The structure of the "*opaque blob*" in the hypervisor interface is known to UKI and packaged firmware.
  - The UKI conveys information on where it loaded the boot bits into memory for the packaged firmware to consume after reset.



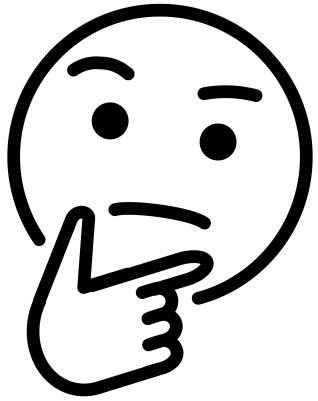
Red Hat

f-UKI bits ... (3)

- How to build the support for hypervisor interface into UKI in a hypervisor agnostic manner?
  - We prototyped and built patches only for QEMU/edk.
  - The demo was a prototype work - patches are rough work.
  - We are requesting inputs from the community to help us in this effort.
- QEMU uses fw-cfg. We need to build support/write a driver for fw-cfg in UKI. How to design that?
- Open question: What happens when there multiple initrd and kernel command lines are in UKI?

Agenda

- Background
- What?
- Why?
- How?
- Using UKI to bring in firmware
- **Current Status**
- References



Red Hat

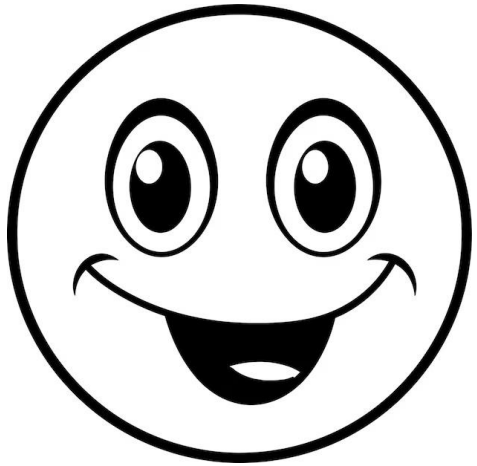
Progress so far ...

QEMU:

- Some code reorgs needed towards enabling reset in CoCo **merged**!
- QEMU hypervisor interface design proposal patch posted. ✅
  - Link to google doc where we had elaborate discussions **later in references**.
  - See @*qemu-devel* link **later in references** for the latest patch.

Systemd:

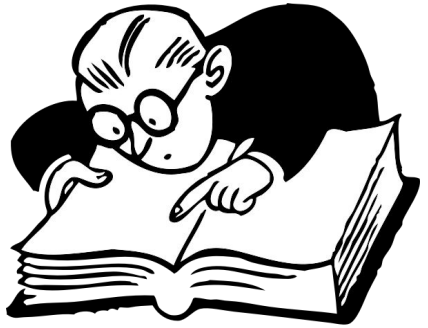
- Changes towards matching a firmware to a specific hardware **merged**!
- WIP: adding a "***efifw***" UKI section to contain the UEFI firmware.
  - WIP**:** update related documentations. ✅



We need your interest and help in getting all the bits of work into upstream.

Agenda

- Background
- What?
- Why?
- How?
- Using UKI to bring in firmware
- Current Status
- **References**



Red Hat

Upstream (QEMU, Systemd) activities

- Hypervisor interface design doc:
  - https://docs.google.com/document/d/14P5L2mwaGcfsKKnDkQL5dxi7j1Mc1rzXtbuvilA5xfU/edit?usp=sharing
- QEMU hypervisor interface patch on @*qemu-devel*:
  - https://mail.gnu.org/archive/html/qemu-devel/2025-01/msg05693.html
- Systemd PRs:
  - https://github.com/systemd/systemd/pull/35091 (WIP)
  - Doc update: https://github.com/uapi-group/specifications/pull/131 (WIP)
  - https://github.com/systemd/systemd/pull/35747 (**merged**!)

KVM Forum 2024

https://kvm-forum.qemu.org/2024/

- Talk page: https://pretalx.com/kvm-forum-2024/talk/HJSKRQ/
- Talk recording: https://youtu.be/VCMBxU6tAto?feature=shared
- Demo: https://people.redhat.com/~anisinha/BYOF-demo.mp4
- PDF slides:

  https://kvm-forum.qemu.org/2024/BYOF_-_KVM_Forum_2024_iWTioIP.pdf

Can also access these from my Red Hat page:
https://people.redhat.com/~anisinha/

Photo credit: Anthony Davis, Red Hat.

Red Hat

Links to talks, specs and documentations ...

- *UKI spec*: https://github.com/uapi-group/specifications/blob/main/specs/unified_kernel_image.md
- Emanuele's FOSDEM 2024 talk on UKI/secure boot work Red Hat is doing https://archive.fosdem.org/2024/events/attachments/fosdem-2024-2024-uki-addons-and-extensions-safely-extending-ukis-kernel-command-line-and-initrd/slides/22125/Uki_addons_O97iYns.pdf
- Lennart's FOSDEM 2024 talk on systemd-boot and UKI https://archive.fosdem.org/2024/events/attachments/fosdem-2024-1987-systemd-boot-systemd-stub-ukis/slides/22834/systemd-boot_systemd-stub_UKIs_mNuvmv0.pdf
- SHIM source: https://github.com/rhboot/shim/
- SBAT details: https://github.com/rhboot/shim/blob/main/SBAT.md

Links to talks, specs and documentations …

- Computer hardware IDS (CHID): https://learn.microsoft.com/en-us/windows-hardware/drivers/dashboard/using-chids
- Secure boot and measured boot Microsoft page: https://learn.microsoft.com/en-us/windows/security/operating-system-security/system-security/secure-the-windows-10-boot-process
- Controlling secure boot https://www.rodsbooks.com/efi-bootloaders/controlling-sb.html
- Measured boot design https://trustedfirmware-a.readthedocs.io/en/v2.11/design_documents/measured_boot.html
- Remote attestation using ephemeral TPM https://dl.acm.org/doi/pdf/10.1145/3627106.3627112
- *Intel TDX whitepaper* https://cdrdv2.intel.com/v1/dl/getContent/690419
- *AMD Secure Encrypted Virtualization* https://www.amd.com/en/developer/sev.html

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.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat