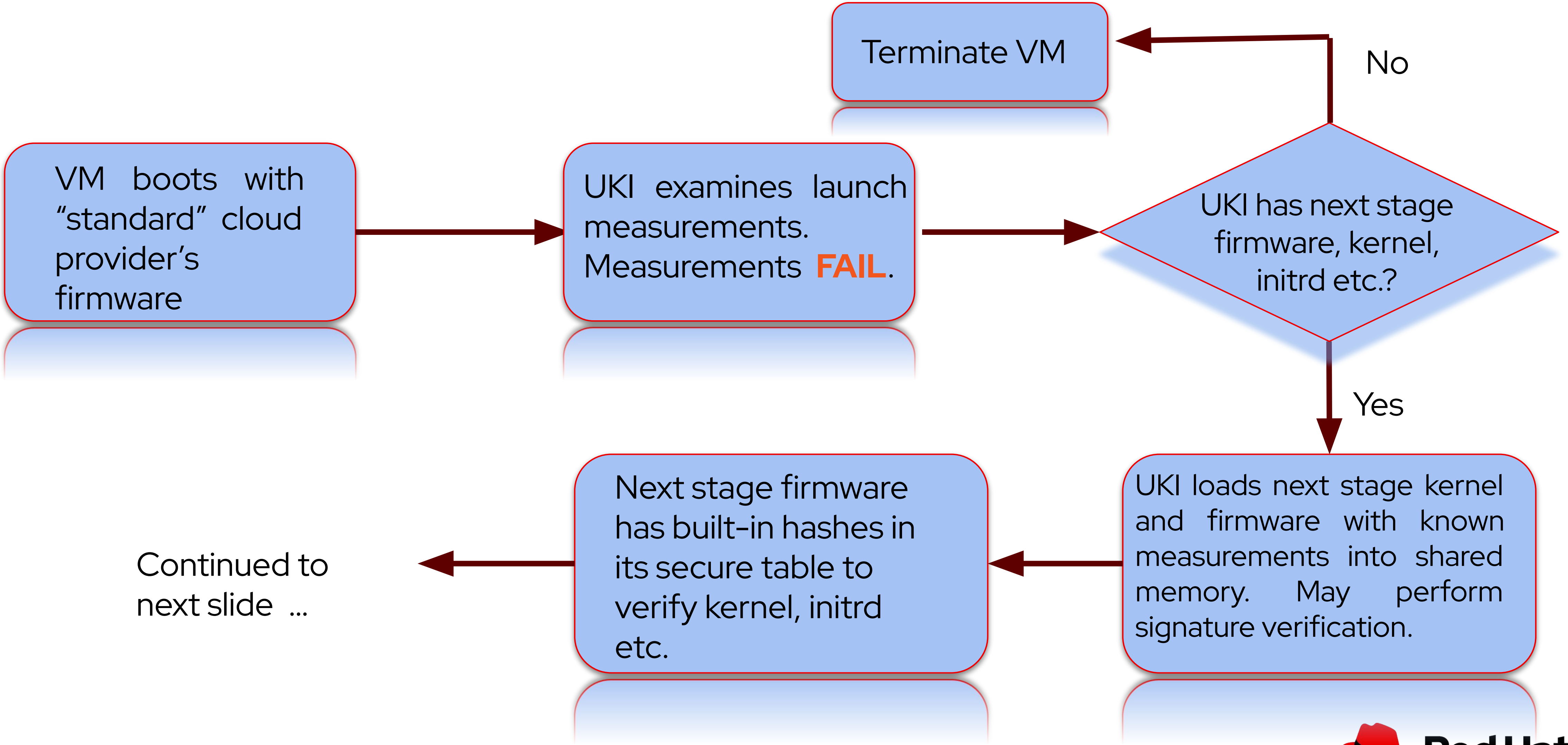
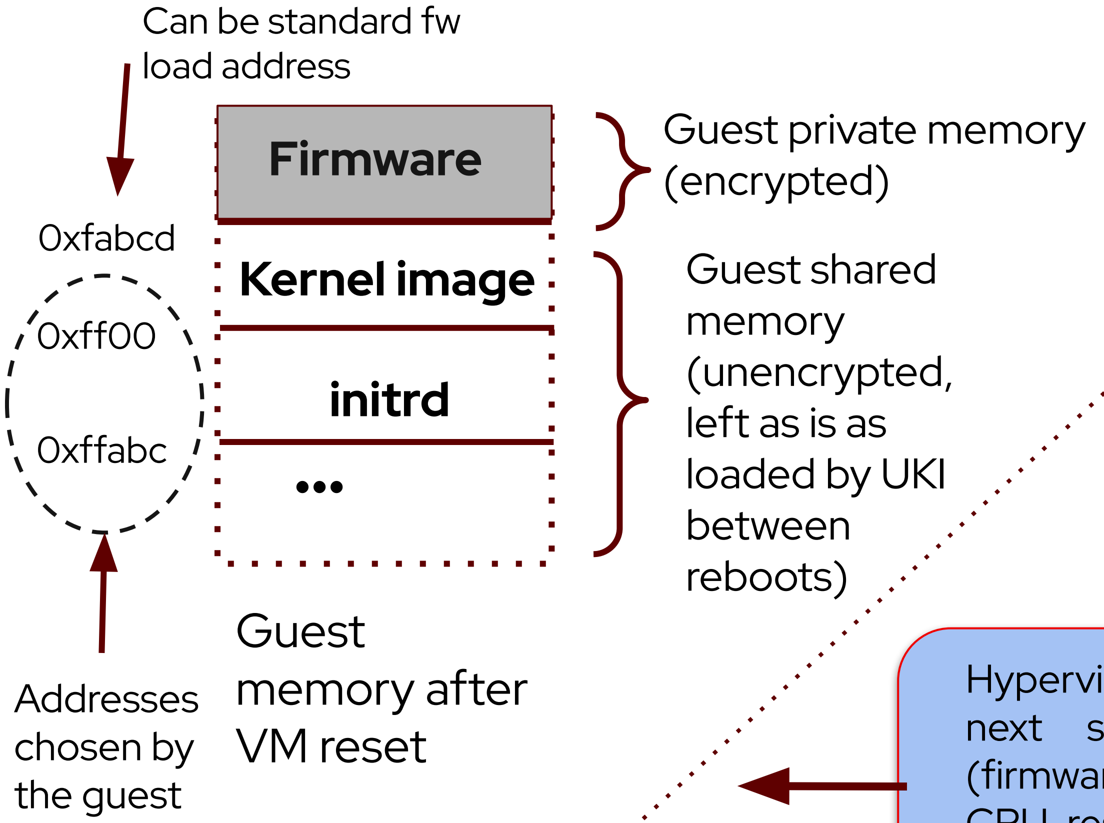
How?

Proposed launch digest update mechanism



Proposed launch digest update mechanism (contd ...)



Continued from previous slide ...

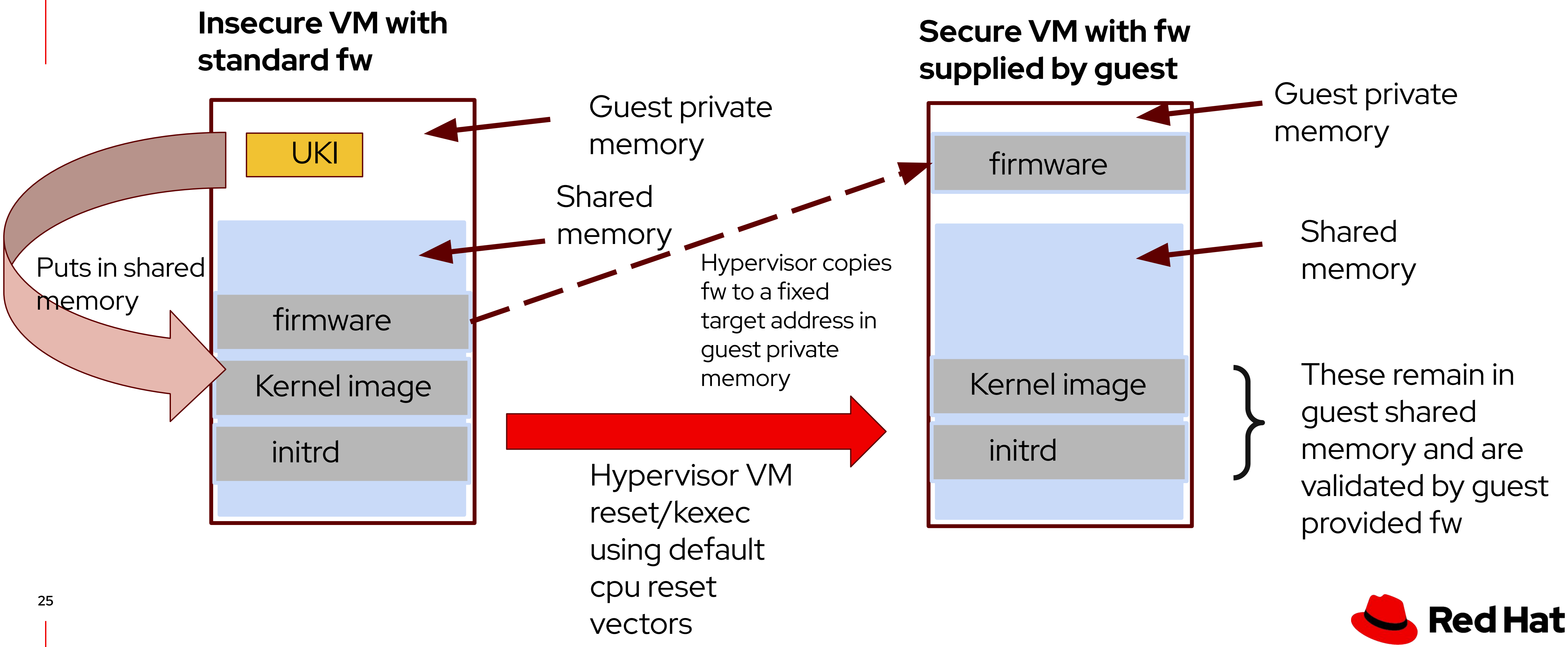
UKI uses hypervisor interface (fw_cfg) to pass address range of next stage firmware where it loaded in (shared memory) along with other next stage boot components.

Hypervisor copies the firmware to the target private address in the guest. This address is a fixed address for a particular architecture.

Hypervisor resets the VM with the next stage boot components (firmware, kernel etc) using default CPU reset vectors. Marks private memory as private. Generates new SEV context.

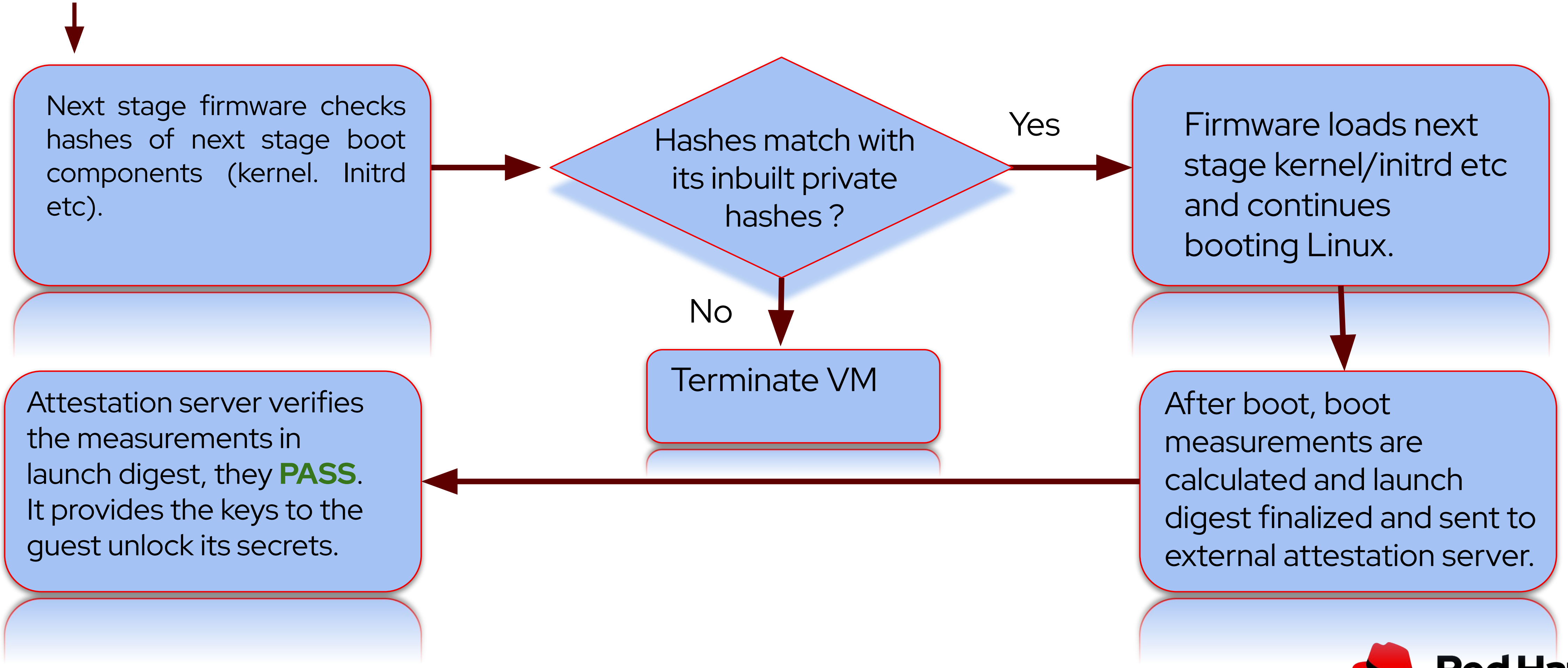
Continued to next slide ...

Proposed launch digest update mechanism (contd ...)



Proposed launch digest update mechanism (contd …)

Continued from previous slide …



Salient points ...

- The guest chooses where to place firmware, kernel and inirtd blobs.
    - Guest to guest ABI.
- UKI need to support “firmware” section (along with kernel, initrd, command line etc).
- Ukify.py ensures next stage firmware only loads trusted kernel and initrd etc by installing their hashes in the firmware’s secure hash table
- We are using the guest shared memory (shared with the hypervisor) as a data plane to pass the initial launch digests (firmware, kernel, initrd etc). The memory comes from the guest. No separate hypervisor memory allocation required.

Salient points ...

- Private launch digests (eg. next stage firmware) are copied from the shared memory to guest private memory by the hypervisor before restarting and regenerating the VM context.
- Systemd checks platform/capability bits to make sure we are loading the correct firmware version for the correct platform.
- If we use default CPU reset register values, no need to pass initial CPU states.
- Next stage firmware validates kernel, initrd etc since their hashes are stored in a hash table inside a secure page in the firmware.
  - There is no need to generate measurements for these components.
  - Signature verification also becomes optional.

Salient points ...

- The firmware itself is validated by the launch measurements that are sent to the external attestation server.
- To make sure that secure VM remains secure after updating the firmware, we also have a provision for implementing a “kill switch”.
    - Once the firmware/kernel etc are updated, no more updates are allowed using the hypervisor interface.

Brief overview of major stack components involved

- **QEMU**:
  - Hypervisor/guest interface in QEMU (fw_cfg based).
  - Guest reset mechanism for secure VMs
    - Currently in QEMU, resettings CPUS is not allowed - a reboot terminates the guest.
    - Shared memory needs to be preserved across reset.
    - New SEV context needs to be generated after reset.
  - Machine changes to make sure loader correctly loads firmware to the right address.

- **Systemd**:
  - Support for guest/hypervisor interface in systemd-boot.
  - Check platform/capabilities to make sure correct firmware is loaded.
  - Support for loading launch digests, using fw_cfg interface to pass digest information to hypervisor.
  - Trigger reset.

- **Firmware** (EDK2):
  - Fw_cfg changes to read platform/capability bits.
  - Scan fw_cfg vmfwupdate_blobs to find the kernel/initrd addresses and load linux from there.