Multi-Profile UKIs

and Other Ways to Supercharge Your Unified Kernel Images

Lennart Poettering
FOSDEM 2025
Brussels, Belgium

20 min

UKIs? What's that again?

UKIs → "Unified Kernel Images"

Single UEFI PE binary consisting of:

`systemd-stub` EFI stub +

Kernel image +

initrd image +

Kernel command line +

Devicetree +

Boot splash +

…

→ https://uapi-group.org/specifications/specs/unified_kernel_image/

Benefits

Robust: one file the boot loader needs to read

Secure: one file that can be signed + measured as whole

Problem Statement

Not trivially locally modifiable, uniform image, everywhere

Solution #1: **EFI add-ons** (authenticated via SecureBoot/shim), covering initrd, devicetree, kernel cmdline, CPU µcode, …

Solution #2: **systemd credentials** (authenticated via TPM), for parameterization of system and services

Solution #3: `systemd-confext` + `systemd-sysext` images (authenticated via kernel keyring), for extending `/etc/` and `/usr/`

Conceptually all three are “side-cars”: files dropped next to kernel that extend UKI in a flexible fashion

Multi-Profile UKIs

Solution #4 (new)

A single UKI – but with multiple profiles

One UKI, with multiple *alternative* sections for kernel command line, initrd, and so on.

Not a sidecar.

Limited flexibility, only a few blessed configurations.

Primary use-case: *one* UKI with multiple different kernel command lines, e.g. one for regular boots, one for recovery mode, one for factory reset, one for storage target mode, and similar.

Multi-Profile UKIs, Part #2

`systemd-boot` has been updated to understand profiles

One UKI, multiple menu items

Profile choice is measured to TPM PCR

Authentication by SecureBoot + PCRs just like any other UKI

Measurement only covers sections of chosen profile

`systemd-measure` + `ukify` natively support multi-profile UKIs

Profiles carry extensible, descriptive metadata (used for menu item strings)

Other Ways to Supercharge UKIs

Automatic choice of Devicetree blob

Include multiple `.dtbauto` sections

Include `.hwids` section that maps MSFT CHID → Devicetree “compatible” string

→ Devicetree is automatically selected at boot, by `systemd-stub`

Soon: Bring-Your-Own-Firmware

Automatic choice of firmware update

Usecase: BYOF cloud systems

Include one or more `.efifw` sections (containing name + UEFI capsule)

Include a `.hwids` section that maps MSFT CHID → `.efifw` firmware name

Firmware is automatically installed at boot when needed, by `systemd-stub`, followed by reboot

NB: qemu can nowadays directly boot into UKI (no boot loader, no systemd-stub necessary for any of this)

(See other FOSDEM talk by Anhi: https://fosdem.org/2025/schedule/event/fosdem-2025-4661-introducing-fuki-guest-firmware-in-a-uki-for-confidential-cloud-deployments/)

Hypercharged UKIs

Embed a whole OS into a UKI → USI (“Unified System Image”)

Never transition into any other file system

Whole OS runs from the initrd cpio

Conceptually from PoV of kernel: system never leaves the initrd

Conceptually from userspace PoV: system never goes through initrd

Example: diskomator (https://github.com/poettering/diskomator)

How to Build Supercharged UKIs + USIs?

Manually: `systemd-measure` + `ukify`

Or more comprehensively: `mkosi`

The End