Immutable OS images

MiniDebConf Hamburg 2026

Manuel Traut, manut@debian.org

#Debian #LinuxOnMobile #EmbeddedLinux

May 10, 2026

Agenda

- Goal
- Target group
- Concept
- Components
- Additional requirements
- Existing solutions
- Considerations

What's the goal?

- Increase the reliability of a system
- Provide a secure run-time environment for services
- Protect secrets
- Fullfil requirements of regulations like EU-CRA
- Avoid execution of unsigned code, eg. due to modification

Who would like to have this?

- Physical server machines (hosting VMs, k8s, ..)
- Network equipment like routers
- Smart things like TVs, phones, vacuum cleaners, ..
- Automation control devices
- Measurement equipment
- I don't want to have this on my laptop as of today

What's the concept?

- Verified, secure boot chain
- Atomic (image based) signed upgrades
- Limited access and modifications during runtime (even if you are root)

What components are involved?

Verified boot chain

- Firmware
- Boot-menu
- Kernel cmdline
- Kernel, initrd, (devicetree)
- OS / userspace
- Applications
- Data
- Configuration

Verified boot chain

Firmware

Eg. EFI with own keys

- uEFI: barebox, coreboot, EDKII, u-boot
- Secure storage for keys e.g. RPMB
- Secure storage key derived from eg. SoC Machine ID

Verified boot chain

Boot-menu

Eg. grub or systemd-boot

- Rollback protection
- Recovery
- Factory reset

Verified boot chain

Kernel cmdline

- Not editable by user
- Compiled in only
- Part of UKI, FIT, DTB

Verified boot chain

Kernel, initrd, devicetree

- FIT or UKI
- Multiple signed artifacts

Verified boot chain

OS / userspace

- Image based; either just /usr or /
- Read-only FS, dm-verity
- OR libostree, composefs

Verified boot chain

Applications

- flatpak, snap
- systemd portable-service
- podman, docker

Verified boot chain

Data volumes

- IPE to avoid execution of binaries

```
CONFIG_IPE_BOOT_POLICY="ipe.txt"

policy_name=Ex_Policy policy_version=0.0.0
DEFAULT action=DENY

op=EXECUTE  boot_verified=TRUE action=ALLOW
op=FIRMWARE boot_verified=TRUE action=ALLOW
op=KMODULE  boot_verified=TRUE action=ALLOW

op=EXECUTE  dmverity_signature=TRUE action=ALLOW
op=FIRMWARE dmverity_signature=TRUE action=ALLOW
op=KMODULE  dmverity_signature=TRUE action=ALLOW
```

Verified boot chain

Configs

- Symlinks to rw storage
- systemd-confext
- systemd tmpfiles.d to init /etc from /usr

What else is needed?

- Roles and authorities
- Keys and revocation policies
- Secure update of all components
- Build; test; sign; deploy

What is available?

Live 'CDs'

- Simple, approved
  - No protection during runtime
  - Configurations lost by reboot
  - No update mechanism

What is available?

btrfs snapshots

- openSusSE MicroOS
  - \+ Transactional, differential upgrades
  - \- No fs integrity during runtime

What is available?

bootc + libostree (git for filesystems)

- Apertis, EndlessOS, Fedora/RH Atomic Spins, TorizonOS, webOS
  - \+ Transactional, differential upgrades
  - \+ Layered images
  - \+ FS integrity with composefs

What is available?

uEFI + systemd-boot + UKI + A/B dm-verity-sig

- bengalOS, GNOME OS, particleOS, postmarketOS
  - \+ Atomic upgrades eg. with systemd-sysupgrade
  - \+ Block-layer integrity with dm-verity and IPE
  - \+ OS=/usr or OS=/ is possible

More challenges?!

- User creation
  - systemd-homed-firstboot.service
  - phosh-first-boot
- Disk encryption
  - During first-boot, key in TPM2
- machine-id on read-only RFS
  - Derive from firmware
    systemd.machine_id=firmware
- ssh hostkeys
  - Generate during firstboot
  - Located on a shared r/w partition
    override sshd_keygen.service
- dist-upgrades (cfg format changes, db upgrades, ..)

Thank you!

See you soon at MiniDebConf CH Winterthur

- Debian can be used to build immutable images \o/
- Try BengalOS :)
  https://salsa.debian.org/BengalOS-team/bengalos-recipes