https://goo.gl/1EGWkr

BoF: Secure OTA Collaboration

Ricardo Salveti, Principal Engineer
Alan Bennett, VP Engineering

Monday, Oct 23 2017 - ELC-E

OPEN SOURCE FOUNDRIES

Background

Open Source Foundries - a new Startup productization of what we did @ Linaro

- Minimal, secure, open source, updateable ‘easy’ microPlatforms
- Ricardo Salveti, Tyler Baker, Marti Bolivar, Milo Casagrande, Michael Scott, Andy Doan
- Recent activity: LAVA-docker, KernelCI, Jobserv, OSLight, Anti-patterns, Zephyr LWM2M/FOTA Framework, OTA Collaboration / Security design

Now, let’s get technical

Goals of the BoF

- Early analysis pointed us to many ‘kinda-different, but open’ solutions
- Analyzed OTA systems, summarize, propose some collaboration steps
  - Security is hard, best to share open and common solutions when possible
- If we miss or get things wrong, speak up, don’t let this be a one-way talk

Not going to be covered in this BoF:

- Comparison between current major OTA solutions
  - Extensively covered at previous conferences (check references for the presentations as we only have 45 minutes!)
- Traditional package-based systems (rpm, deb, etc)

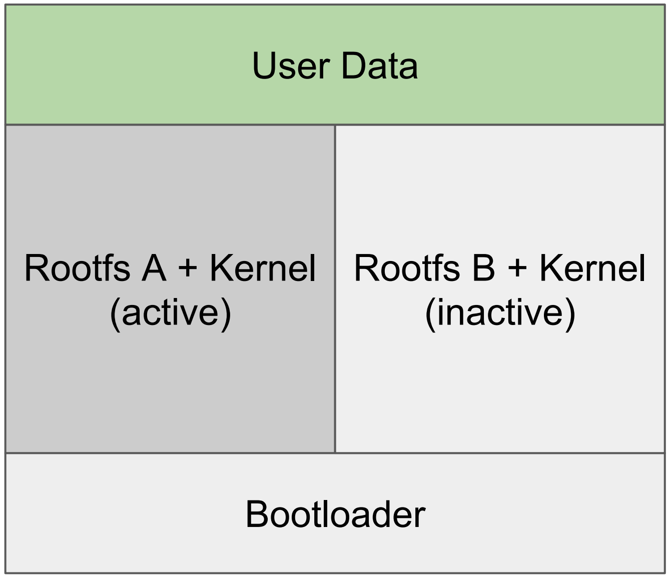
IoT Software Update Requirements

OPEN SOURCE FOUNDRIES

- Atomic updates
  - Stateless system
- Capable of updating bootloader, kernel, configuration and the rootfs
- Fail-safe, rollback previous software state
  - Boot / update monitoring (watchdog), with boot confirmation
- Secure download / verification of the image
- Easy to use / consume without vendor lock-in
  - Ideally supported by OpenEmbedded (external layers)
- Trusted boot and execution of software update in a trusted environment
  - Leveraging platform's hardware TPM and/or TEE features

Block-based Update Systems (1/2)

- Symmetric and/or Asymmetric
- Mostly dual bank (A/B) scenarios
- RW data in a separated partition
- Bootloader dependency
- Full rootfs update
  - Reboot required
- Safe and reliable process
  - For both update and rollback
- Easy to manage at the server side
- Image verification (key / cert)
- OE/Yocto layer usually available



Block-Based Update Systems (2/2)

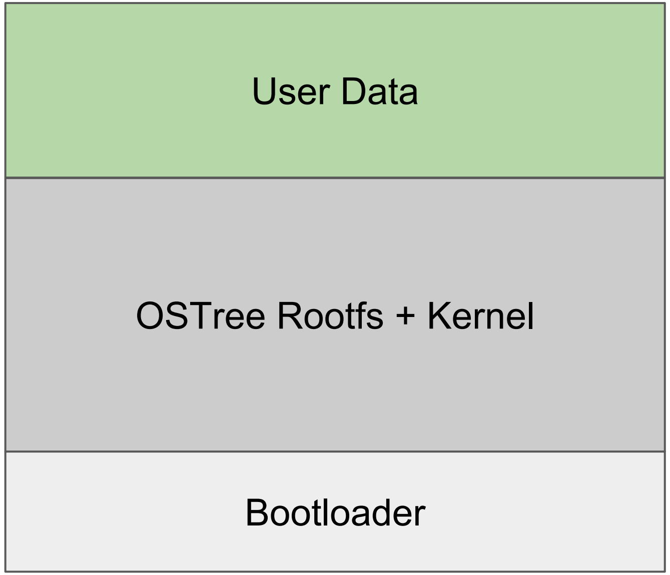
Main implementations:

- SWUpdate (GPLv2): https://github.com/sbabic/swupdate
- Mender (Apache v2.0): https://mender.io
- RAUC (LGPLv2.1): https://github.com/rauc/rauc
- ResinOS (Apache v2.0): https://resinos.io

Some more flexible than the others, some also offering Open Source server-side implementations (e.g. swupdate with hawkbit / mender).

File-based Update Systems (1/2)

- Updates to individual files / dirs
- Reboot may be optional (swupd)
- Simpler partition layout
- Fast download / update process
  - Worst case: full rootfs update
- Bootloader dependency
- Safe and reliable process
  - For both update and rollback
- Server side more complex
- Image verification (key / cert)
- OE/Yocto layer usually available



File-based Update Systems (2/2)

Main implementations:

- OSTree (LGPLv2): https://github.com/ostreedev/ostree
    - "Git for operating system binaries"
    - Used by several projects:
        - Gnome Continuous: https://wiki.gnome.org/Projects/GnomeContinuous
        - Project Atomic: https://github.com/projectatomic/rpm-ostree
        - QtOTA: http://doc.qt.io/QtOTA/
        - flatpak: https://github.com/flatpak/flatpak
        - Automotive Grade Linux: https://github.com/advancedtelematic/meta-updater
        - Endless OS: https://github.com/endlessm/eos-updater
- Swup (GPLv2): https://github.com/clearlinux/swupd-client

Problems Identified

- Secure / verified boot story still problematic
  - Usually hardware specific
- Trusted execution environment not widely used
  - Trusted execution of the OTA client (image update / swap)
  - Runtime integrity check
  - Trusted storage / eMMC
- Boot firmware updates
- Several OE Layers duplicating board specific logic
  - Mostly around bootloader patching and scripting
- Lack of threat models
  - Antipatterns in IoT: https://lwn.net/Articles/733512/
- Secure Software Distribution

Secure Software Distribution

- Main problem found with the current OTA systems:
  - HTTPS + Crypto (e.g. GnuPG) is not necessarily enough for a fully secure solution
  - System still considerably vulnerable against several other attacks:
    - Freeze, endless data, rollback, wrong software installation, malicious mirrors
- The Update Framework Specification (TUF)
  - Metadata for target files
  - Key features: multiple roles, data freshness, signed collection, key hierarchy, transparent key rotation and threshold (targets) signing
  - Projects implementing TUF: Docker (Notary), CoreOS, Python's pip, Ruby's gems
- AGL / ATS ahead of the game, TUF / Uptane implementation
  - Uptane is based on TUF but extended to better cover the automotive requirements

Suggestions for Collaboration

- Guidelines / reference implementation for secure boot
- Trusted execution environment (bootloader update, integrity checks)
- Bootloader rootfs image update process (image swap, boot count)
- Boot firmware update process
- Integration with different Open Source management servers
    - Mender support in SWUpdate?
- Watchdog best practices / boot image validation
- Secure software distribution (TUF) implementation

https://elinux.org/Secure_OTA_Update ?

References

- Yocto System Update Comparison Wiki
- Identifying secure firmware update mechanisms and open source options for embedded Linux devices (Alex Gonzalez - Digi International)
- [RFC] Device-side support for software update in AGL (Konsulko Group / ATS)
- Comparison of Linux Software Update Technologies (Matt Porter, Konsulko Group)
- Open Software Updates for IoT (Phil Wise, Advanced Telematic Systems)
- How we added software updates to AGL (Phil Wise, Advanced Telematic Systems)
- How do you update your embedded Linux devices? (Daniel / Keijiro, Toshiba)
- Secure boot Secure software update (Yannick Gicquel, iot.bzh)
- Surviving in the wilderness integrity protection and system update (Patrick, Intel)
- Secure Software Distribution in an Adversarial World (Diogo Mónica, Docker)
- The Update Framework Specification

Relevant Talks this week!

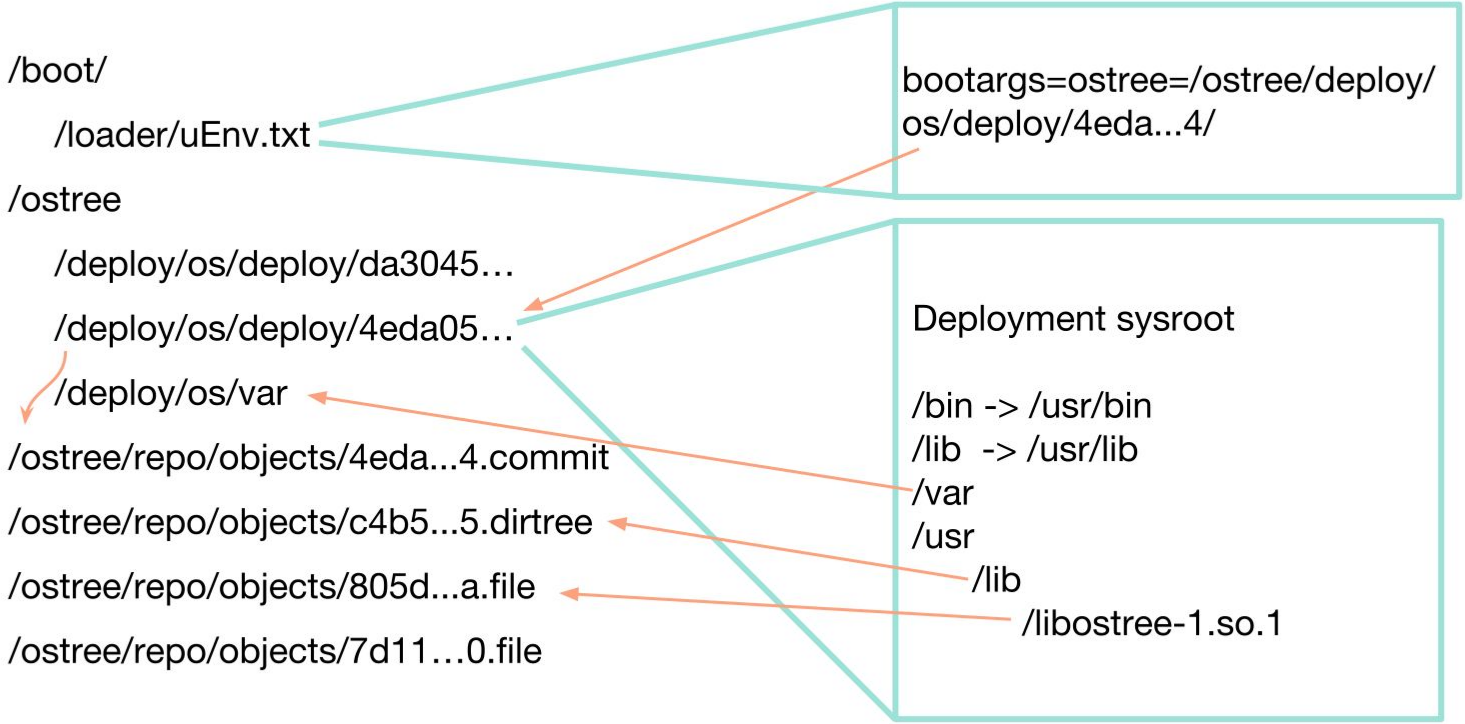
- Tuesday, October 24 • 11:45 - 12:25 - Protecting Your System from the Scum of the Universe - Gilad Ben-Yossef, Arm Holdings
- Tuesday, October 24 • 14:05 - 14:45 - Orchestrated Android-Style System Upgrades for Embedded Linux - Diego Rondini, Kynetics
- Wednesday, October 25 • 15:05 - 15:45 - Updating an Embedded System with SWUpdate Framework - Stefano Babic, DENX Software Engineering Gmbh

Summary

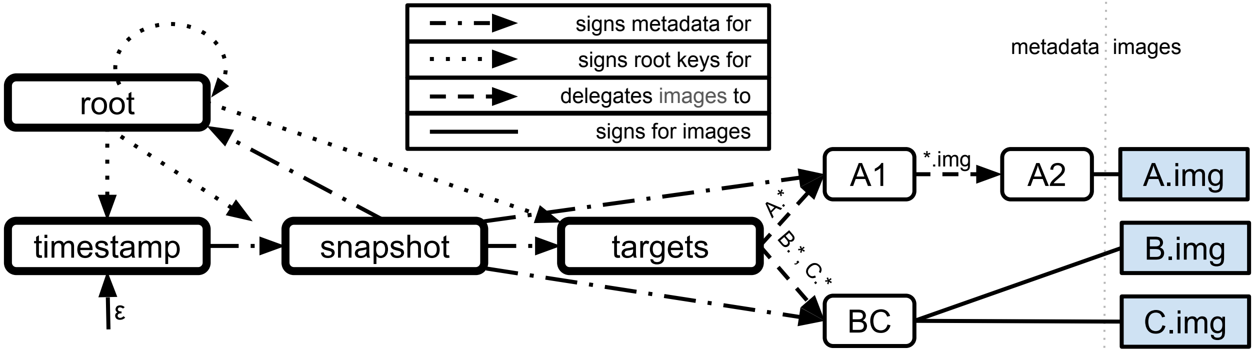
Thanks!

OPEN SOURCE
FOUNDRIES

OSTree basics: sysroot



Design principles for a repository

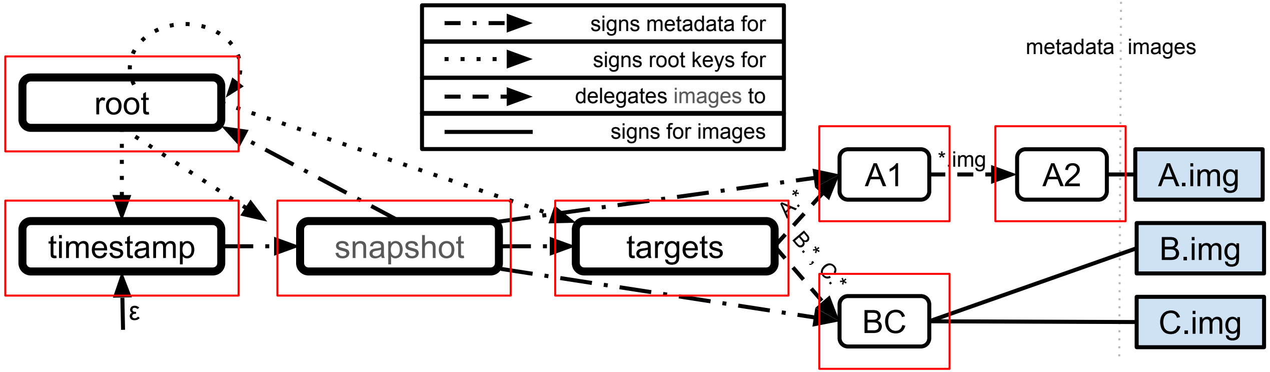


Design principles:

1. Separation of duties.
2. Threshold signatures.
3. Explicit and implicit revocation of keys.
4. Minimized risk through use of offline keys.

[SOTA #5: Uptane Design Overview]

Separation of duties

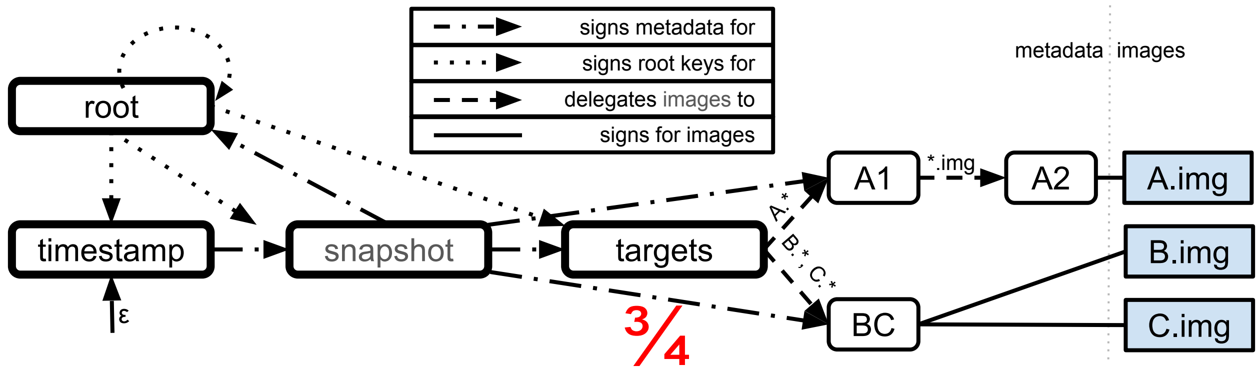


Design principles:

1. **Separation of duties.**
   - Sign different types of metadata using different keys.
   - Metadata about images (self-contained archives of code+data for ECUs), or other metadata files.

[SOTA #5: Uptane Design Overview]

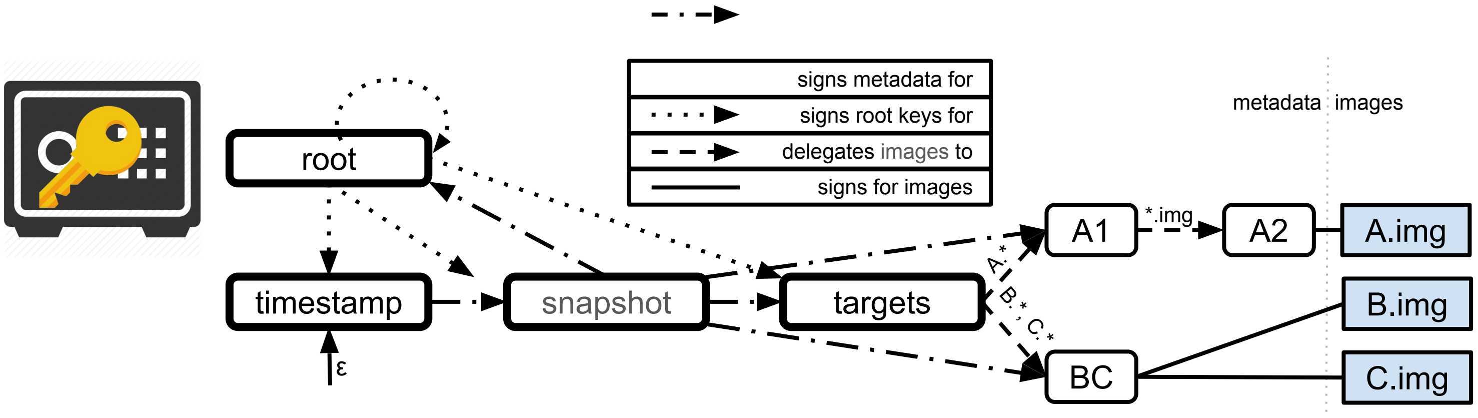
Threshold signatures



Design principles:

1. Separation of duties.
2. **Threshold signatures.**

Minimizing risk with offline keys

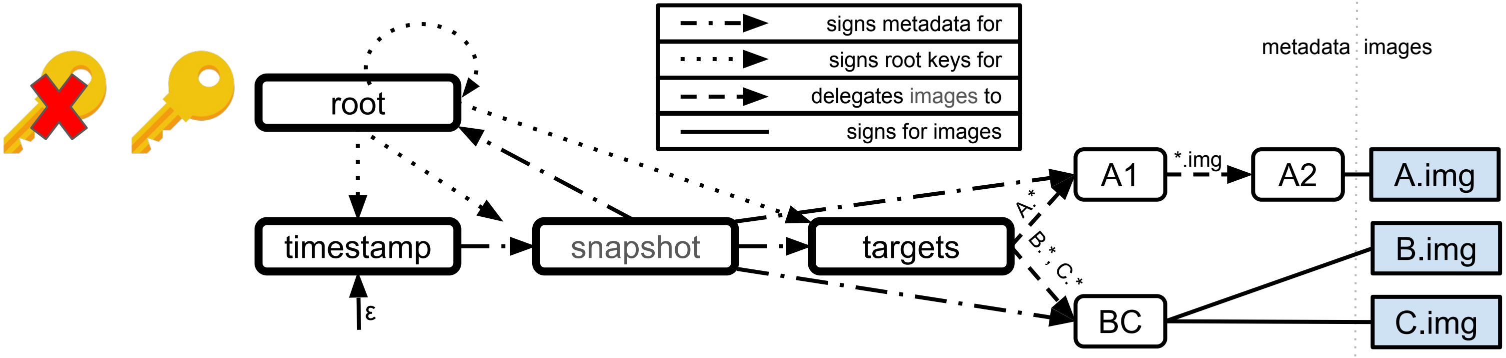


Design principles:

1. Separation of duties.
2. Threshold signatures.
3. Explicit and implicit revocation of keys.
4. **Minimized risk through use of offline keys.**

[SOTA #5: Uptane Design Overview]

Explicit & implicit revocation of keys



Design principles:

1. Separation of duties.
2. Threshold signatures.
3. **Explicit and implicit revocation of keys.**

[SOTA #5: Uptane Design Overview]