SURVIVING IN THE WILDERNESS INTEGRITY PROTECTION AND SYSTEM UPDATE

Patrick Ohly, Intel Open Source Technology Center

Preliminary version

MOTIVATION FOR THE TALK

- Why bother?
- Why yet another talk?
- What’s my background?

PERSONAL BACKGROUND

- Security and update in Ostro™ OS
    - meta-intel-iot-security/meta-integrity: IMA
    - meta-swupd: Clear Linux* update mechanism
- Supporting an update mechanism in the Yocto Project? Comparison in the Yocto Wiki.
- Integrating dm-verity and whole-disk encryption into IoT OS Reference Kit for Intel® Architecture

* other names and brands may be claimed as the property of others

WHY BOTHER?

- Surviving…
  - Harden before shipping.
  - Update once deployed to fix new vulnerabilities.
- … in the wilderness
  - Hostile environment: unauthorized users may be able to access, modify and boot a device.
  - Integrity protection must ensure that a device only runs unmodified software, in an unmodified configuration.

CONTENT OF THE TALK

- Taxonomy of update mechanisms
- Interaction between system update and integrity protection
- Hands-on part with IoT Refkit

TAXONOMY OF UPDATE MECHANISMS

CANDIDATES COMPARED FOR YOCTO

- swupd
- OSTree
- swupdate
- mender.io

https://wiki.yoctoproject.org/wiki/System_Update

KEY CRITERIA

- Block based vs. file based
- Partition layout
- Integration with boot process
- Integration with update server for over-the-air (OTA) updates

BLOCK VS. FILE UPDATE

- Block based: update partitions (swupdate, mender.io)
    - Reboot required
    - Partition size fixed
    - Rewrite entire partitions
- File based: update individual files and directories (swupd, OSTree)
    - Reboot may be optional (swupd)
    - Same update stream can be applied to devices with different disk sizes
    - Very efficient

PARTITION LAYOUT

- A/B setup: “live” partition and second partition that gets updated
  - mender.io relies on this
  - Supported by swupdate
  - Could be done with OSTree and swupd
- Single partition
  - Supported by swupdate
  - Default mode of operation for OSTree and swupd
- Updating content outside of the rootfs partition?

INTEGRATION WITH BOOT PROCESS

- Choose what to boot into
  - OSTree bind-mounts actual rootfs
  - mender.io and swupdate set u-boot variables
- Rescue mode
  - swupdate has recipe for fallback initramfs

INTEGRATION WITH UPDATE SERVER

- Clients pull anonymously, need additional telemetry
  - OSTree
  - swupd
- Dedicated update server
  - mender.io, including hosted service
  - swupdate supports hawkbit

INTEGRITY PROTECTION

AVAILABLE OPTIONS

- Linux Integrity Measurement Architecture (IMA) with Extended Verification Module (EVM)
- Whole-disk encryption with per-machine secret key
- dm-verity

IMA/EVM

- Originally designed for remote attestation based on measurements
- Extended to enforce locally the integrity of file content (IMA) and attributes (EVM)
- EVM tied to per-machine key
- Changes file system semantic:
  - Data and xattr must match to make file usable, but get flushed independently (breaks sqlite, increases risk in case of power loss).
- Does **not** protect integrity of directory content and therefore **susceptible to offline attacks**:
  - Disable services by removing files
  - Replace trusted content with symlinks to untrusted content

WHOLE-DISK ENCRYPTION

- Integrity protection a side effect:
  attacker cannot modify files without knowing the secret key
- Offline modifications result in scrambled blocks, which may or may not be detected by the filesystem
- Key (pun intended) problem: creating and securing a per-machine encryption key

DM-VERITY

- Originally designed for Chrome OS, also supported by Android
- Verifies integrity of each block in a read-only partition, modifications immediately lead to read error
- Boot process must verify integrity of short root hash
- Partition also usable without dm-verity

COMPATIBILITY BETWEEN UPDATE AND INTEGRITY

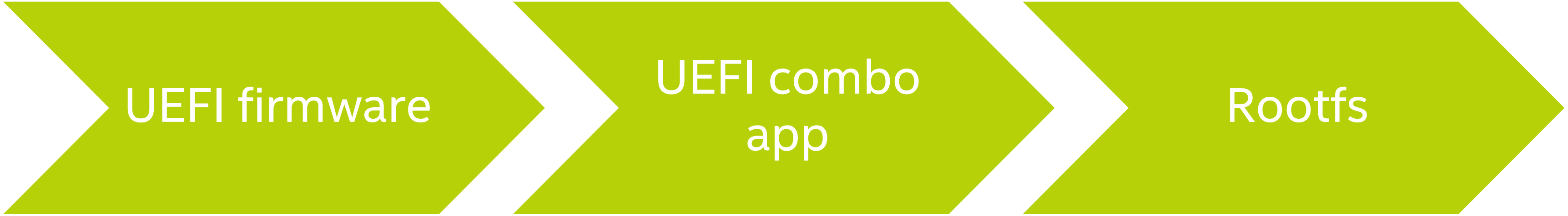
	swupd	OSTree	mender.io	swupdate
IMA/EVM	✓	✓	✗	✗
Encryption	✓	✓	✓	✓
dm-verity	✗	✗	✓	✓

- EVM needs per-machine key and writable rootfs, not compatible with block-based update
- swupd and OSTree need writable rootfs, not compatible with dm-verity

CASE STUDY

dm-verity and LUKS+TPM in IoT Reference OS Kit

ARCHITECTURE



UEFI firmware

UEFI combo app

Rootfs

- flash

- VFAT partition
- kernel + initramfs + systemd-boot EFI stub + boot parameters

- ext4, optionally with encryption or dm-verity

TARGET MACHINE

- qemu
    - swtpm + qemu-tpm patches
    - MACHINE=intel-corei7-64
    - TianoCore/ovmf as firmware
- Fictional device with custom keys enrolled

SYSTEM COMPONENTS: INSTALLER IMAGE

- Contains whole-disk images as input
- Production image free of installer components
- `image-installer` script
  - generic part in `image-installer.bbclass`
  - refkit part in `refkit-installer-image.bb`
- Installation: partition target disk, optional: set up whole-disk encryption with new key in TPM NVRAM, copy files
- Built with wic and new `dm-verity.py` source plugin which creates partition with hash data

SYSTEM COMPONENTS: INITRAMFS

- Based on `initramfs-framework` (OE-core)
- New:
  - initramfs-framework-refkit-dm-verity
  - initramfs-framework-refkit-luks
- Same `refkit-initramfs` for all images, parameterized with per-image boot parameters

HOWTO

TODO: adding layers, reconfiguring distro, building

DEMO: initializing TPM, starting swtpm, booting installer image, booting installed image, updating that image with swupd

OPENS

- Integration of UEFI signing
- A/B partition setup with swupd and/or OSTree
- Stateless rootfs
- Editing boot parameters or at least automatically adapting them to the current machine (serial port)

QUESTIONS?

LINKS

TODO