THE LINUX FOUNDATION PROJECTS

CIVIL INFRASTRUCTURE PLATFORM

Secure Boot and Over-the-Air Updates – That's simple, no?

Jan Kiszka, Siemens AG

Embedded Linux Conference North America 2020, June 30th 2020

About the Presenter

Jan Kiszka <jan.kiszka@siemens.com>

- Working for Siemens Corporate Technology
- (In-house) Embedded Linux consultant & developer
- Member of CIP, isar-cip-core development, some CIP kernel backports
- Maintainer of and contributor to various OSS projects

Credits

Design

- Christian Storm (Siemens)

Integration

- Quirin Gylstorff (Siemens)
- Michael Adler (Siemens)
- Harald Seiler (Denx)

U-Boot integration

- Marek Vasut (Denx)

Agenda

- Motivation
- Concepts
  - Dual-copy update pattern
  - Basic embedded secure boot pattern
  - Designing in variability: secure dual-copy update
- Implementation aspects
  - Bootloaders
  - Kernel and initramfs
  - SWUpdate
- Pre-integration for CIP Core
- Summary

From ROM Firmware to Over-the-Air Updates

- Past embedded systems
  - Unconnected devices
  - “Never touch a running [embedded] system”
  - At most functional fixes
  - Manually applied updates (“please insert update medium”)
- Requirements today
  - Connectivity is standard
  - Security updates inevitable (mandated by IEC 62443 e.g.)
  - Unattended updates required
  - Robust updates (atomic, roll-back capable)

How Can CIP Help?

CIVIL
INFRASTRUCTURE
PLATFORM

Industrial grade

- Reliability
- Functional Safety
- Real-time capabilities

Sustainability

- Product life-cycles of decades
- Backwards compatibility
- Standards

Security

- Security & vunerability managment
- Firmware updates
- Minimize risk of regressions

CIP Software Update Workgroup

- Develop best-practice patterns
- Define requirements on CIP Core, ensure long-term maintenance
- Align with IEC 62443 certification work of Security WG
- Provide reference implementations on top of CIP Core
- Test the implementation on CIP reference hardware

1 SLTS kernel	2 Real-time	3 Testing	4 CIP Core	5 Security WG(*)	6 Software update WG	
✓	✓	✓	✓	✓	✓	Industrial grade
✓		✓	✓		✓	Sustainability
✓		✓	✓	✓	✓	Security

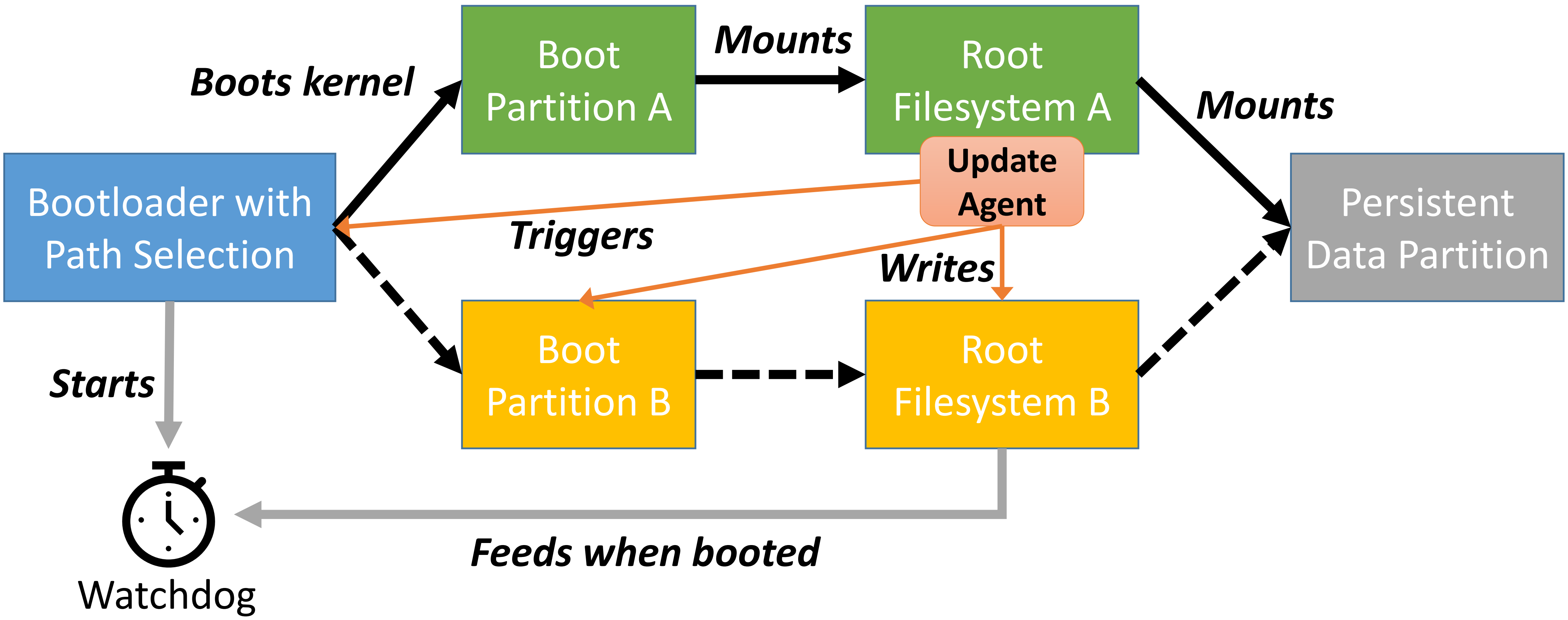
(*): Workgroup

CIP Projects and its scopes

Dual-Copy Updates

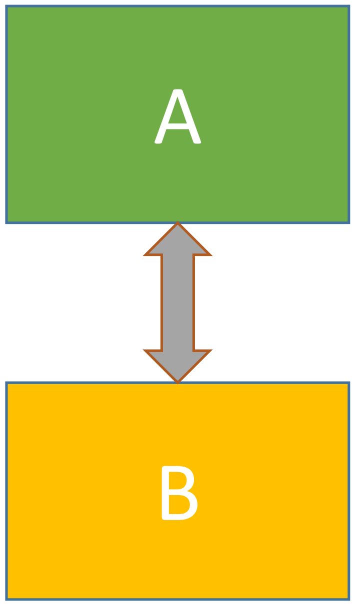
- Always have a full working instance of your software (“A” path)
- Update second instance (“B” path)
- Roll back to A if B does not boot/work
- Pros
  - Ensures that consistent images are used
  - Avoids single points of failures (filesystems, package set etc.)
  - Relatively simple
- Cons
  - Storage
  - Transfer size (can be mitigated with delta images)

Basic Dual-Copy Update Pattern



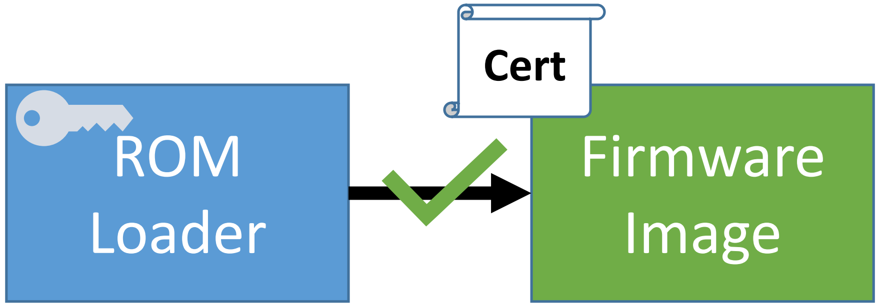
Dual-Copy Update Principles

- Do not touch the working boot path on updates!
  - No changes to partition tables
  - Separate boot partitions and filesystems
- Make update artifacts boot path agnostic
- Confirm update only after checking system state
  - Update service running?
  - Connectivity to update server OK?
  - Device functionality OK?
- Be careful with converting data partition content!



Securing Embedded System Images

- The "ideal world"
  - Monolithic static image
  - Signed by device manufacturer
  - Validated by device ROM prior booting



- Real-world complications
  - Multi-stage boot process, multiple artifacts (bootloader, kernel, rootfs, …)
  - Changes needed during runtime (configuration, logging, …)
  - Vendor-specific security mechanisms and formats
  - …
  - …and updating all of that

Basic Embedded Secure Boot Pattern

- Bootloader protected by hardware
- Bootloader loads and validates
  - Kernel
  - Initramfs
  - [Device tree]
- Initramfs validates static rootfs (dm-verity)
- Data partition handling
  - Consistency check at application-level (i.e open partition)
  - Signing or encrypting via device secret (requires trust anchor)

Challenges with Secure Boot

- Bootloaders must be locked-down
  - Extra, possibly unvalidated boot paths
  - Runtime parameters
  - Interactive sessions
  - …
- …but update state must remain modifiable
- Kernel command line parameters must be locked-down
- Plan for key updates (new keys, revocations)!

Implementation Aspects

- SWUpdate
- Bootloaders
- Kernel containers
- Initramfs logic

SWUpdate as Update Manager

- Versatile tool to manage and perform embedded system updates on the device
- Writes artifacts, controls bootloader
- Supports various input modes
  - Local invocation
  - Integrated webserver
  - Remote server download
  - hawkBit connector
- Can also handle application packages, peripheral firmware, FPGA bitstreams etc.

https://github.com/sbabic/swupdate

Round-Robin SWUpdate Pattern

- Default SWUpdate pattern
  - Hard-coded image target paths
  - Would mean shipping two images (A vs. B path)
- Better solution
  - Use embedded lua scripting to identify target path
  - Script implements round-robin for A vs. B

```
software =
{
   version = "0.2";
   name = "secure boot update"
   images: ({
         filename = "rootfs.img4.ext4.gz";
         device = "fedcba98-7654-3210-cafe-5e0710000001,
                  fedcba98-7654-3210-cafe-5e0710000002";
         type = "roundrobin";
         compressed = true;
         filesystem = "ext4";
   });
   files: ({
         filename = "linux.signed.efi";
         path = "linux.signed.efi";
         type = "kernelfile";
         device = "sda2,sda3";
         filesystem = "vfat";
   })
}
```

CIVIL INFRASTRUCTURE PLATFORM

Bootloader: EFI Boot Guard

- Started by Siemens with two goals
  - Robust boot path selection on UEFI targets
  - Early watchdog enabling
- Replaces grub-efi, systemd-boot etc. – as long as they lack features
- Supported by SWUpdate
- Maintains state in two (or more) FAT partitions
  - UEFI executable to start
  - Parameters to pass to executable
  - Watchdog timeout
  - State version and flags

```
# update-images
# bg_setenv --update
# reboot
…
# bg_setenv --confirm
```

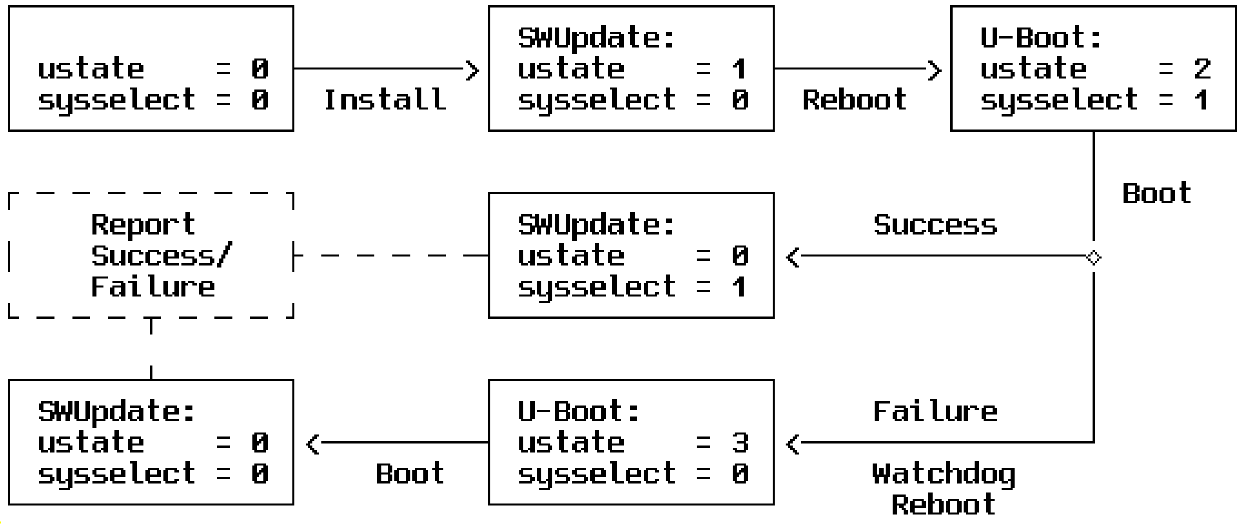
https://github.com/siemens/efibootguard

Secure Boot with EFI Boot Guard

- Fully rely on UEFI
  - Install public key in UEFI firmware
  - Sign bootloader and started executable
  - UEFI will validate artifacts before running them
- Challenge: unprotected state (environment)
- Solution: use unified kernel image
  - Embeds kernel parameters (and ignores passed ones)
  - Embeds initramfs
- Other state variables are all uncritical (worst-case attack: denial of service)

Bootloader: U-Boot

- De-facto standard on embedded devices
- Required features for software update widely available
  - Scripted boot path
  - Watchdog framework & drivers
  - Secure boot features
  - Upcoming: UEFI
- Update via 2 environment vars
  - ustate: idle, try update, booted update, failed
  - sysselect: confirmed A/B path



Secure Boot with U-Boot & Updates

- Sign U-Boot according to SoC needs
- Generate & sign FIT image with kernel, initramfs & DT
- Lock down U-Boot configuration,
  see e.g. https://labs.f-secure.com/publications/u-booting-securely
- Challenge: How to manage update state variables?
- Approach
  - Store in external environment
  - Control that only those 2 variables are read from there
  - Enforce type-checking
  - Lock down all other variables via built-in environment
  - Patches by Marek Vasut pending

Booting The Right rootfs

- Normal kernel parameter: root=/dev/partition (or PARTUUID or …)
- But we need to be A/B agnostic
- Options
  - Filesystem UUID – not always available
  - Write new partition UUIDs on each update – not always available
- Chosen approach: self-made filesystem UUID
  - Write new UUID to custom var in /etc/os-release
  - Embed UUID into initramfs
  - Patch to Debian initramfs selects target partition based on UUID

Pre-integrations for CIP Core

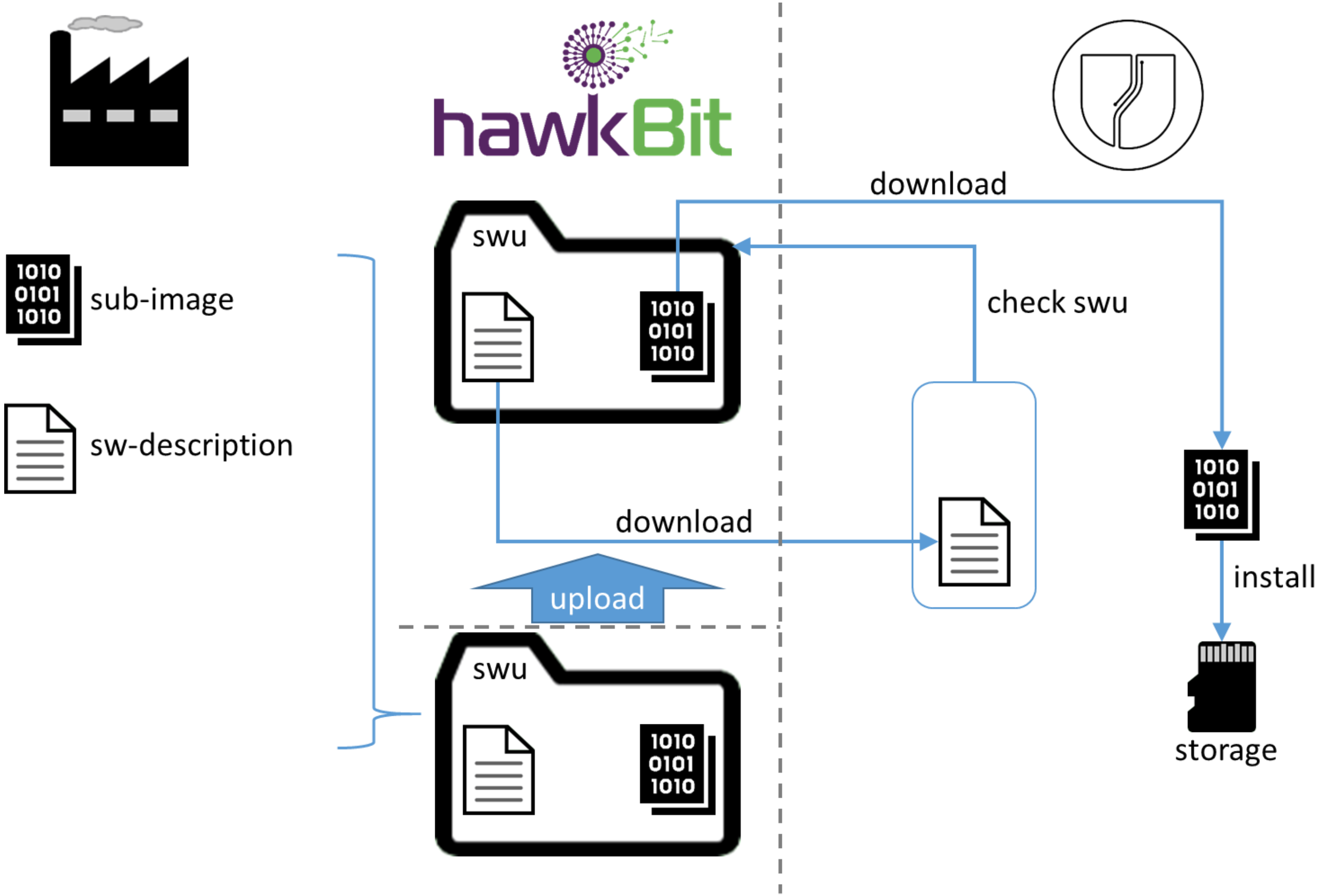
- Initially targeting Isar/Debian (isar-cip-core layer)
- First target: QEMU x86 with UEFI secure boot
- Contains recipes and configs for
  - A/B disk image
  - swu update container
  - SWUpdate (and deps)
  - EFI Boot Guard (with wic plugin for image installation)
  - rootfs-selecting initramfs
  - Signing of artifacts
- https://gitlab.com/Quirin.Gy/isar-cip-core/-/commits/feat/cip-secure-boot, see also try-out instructions

Next Steps

- Add rootfs validation
- Add U-Boot pattern
  - Round-robin lua handler extension
  - Signing of fit images
  - Patches (until merged) and reference configs for QEMU target
- @Siemens: Consolidate projects/products over isar-cip-core
- Explore OP-TEE based secured storage -> data partition protection
- Provide meta-debian support (Yocto/OE style)
- Update “full-stack” demo (real device, hawkBit backend)

Connecting a Backend

- Beyond the scope of this talk
- See Akihiro Suzuki's [talk at CIP Mini Summit 2019]



Summary

- Secure boot + robust software updates = no rocket science!

  …but also nothing for a long afternoon

- Most pieces are available and OSS

  …integration and configuration is the key

- CIP Software Update WG is providing / organizing
  - Blueprints / pre-integrations
  - Testing & long-term maintenance
- Contact us: cip-dev@lists.cip-project.org
- Let's make these features commodity!

Questions

CIVIL
INFRASTRUCTURE
PLATFORM