Introduction of CIP Software Updates Working Group

Akihiro Suzuki, Toshiba
CIP Mini Summit 2019
Lyon, France
October 31, 2019
Who am I?

• Akihiro Suzuki@Toshiba
• Software Engineer since 2011
• The main part of my work
  • Customize and apply Linux to various industrial embedded products
• What’s my role in CIP?
  • CIP SW Updates WG leader
Table of contents

• Introduction of SW Updates WG background
  • Why this WG has been established?
• Introduction of current reference software update mechanism except for A/B update and binary delta update
  • SWUpdate and hawkBit
  • Safe update (signed update & encrypted update)
• Future work
• Summary
Introduction of SW Updates WG background
Why this WG has been established?

• SW Updates WG was established about an year ago

• Background
  • CIP aims to provide super long term support
  • It’s important for CIP to have a reference software update mechanism

• Goal
  • Provide CIP reference software update mechanism
    • Incorporate the mechanism into CIP Core
    • The mechanism should be tested by the testing platform provided by CIP Testing
Introduction of current software update mechanism
Essential requirements from CIP members

- [https://wiki.linuxfoundation.org/civilinfrastructureplatform/cip_software_updates_architecture#requirements](https://wiki.linuxfoundation.org/civilinfrastructureplatform/cip_software_updates_architecture#requirements)

  - Functionality requirements
    - Ability to update the kernel, rootfs, and applications
    - Easy to customize the update steps
  
  - Portability requirements
    - Independent of the image build system
    - Independent of the underlying filesystems
    - Minimize client program dependencies
    - Provide an interface to interact with bootloaders

  - Update media requirements
    - Ability to update from a network server
    - Ability to update from local media: USB, microSD, LAN

  - Resource requirements
    - Minimize network bandwidth usage
    - Minimize storage overhead on the client
    - Keep downtime below a few minutes
    - Minimize storage costs on the server

  - Security requirements
    - Signed updates (authentication, integrity)
    - Encrypted communication

  - Fail-safety requirements
    - Reliable against power loss (atomic updates)
    - Ability to roll back to a previous working image

Network server requirements

- Ability to see the update status
Non-detailed architecture

- [https://wiki.linuxfoundation.org/civilinfrastructureplatform/cip_software_updates_architecture#non-detailed_architecture](https://wiki.linuxfoundation.org/civilinfrastructureplatform/cip_software_updates_architecture#non-detailed_architecture)

**Build tool**
- builds operating system images
  - examples: deby, isar, debos, yocto/oe, live*wrapper, etc.

**Client**
- **Updater**
  - a daemon that accesses the server and performs the updates
  - verifies digital signatures
  - supports various bootloaders (u*boot, efibootguard, etc)
  - candidates: swupdate, rauc, mender, custom script
- **A/B updates**
  - each partition is duplicated (with exceptions such as the data partition)
  - enables lower downtime, rollback, and seamless updates
  - stream updates directly to avoid needing a cache
  - use the active partition as the seed to reduce bandwidth usage
  - For small storage devices jump to an update ramdisk
  - For local updates use a USB filesystem

**Server**
- **Storage**
  - stores operating system images and versions efficiently
  - candidates: casync, ostree
- **Delivery**
  - sends only data that has changed (deltas)
  - candidates: casync, courgette, ostree, rsync
- **Security**
  - guarantees encryption, authenticity and integrity of updates
  - candidates: digital signature (x509), https, delta hashes
- **Server application**
  - has an https REST API (requires a token on the client)
  - communicate status, send commands, download manifests
  - as a frontend to visualize update status and control updates
  - candidates: mender.io, hawkbit, custom(flask, django, expressjs, ..)
- **File*based vs block based**
  - block based updates ensure that any file attributes are updated
  - if the tool supports the necessary attributes file*based updates are possible too

---

CIP Mini Summit 2019

8
Initial prototype of the software update mechanism

• Block based update
  • It ensures that any file attributes are updated

• A/B update
  • It can rollback when the update fails

• OTA update
  • It can update remotely

• Binary delta update
  • It can reduce the consumption of a server storage and a network bandwidth between client and server

• Safe update (today, I’ll be focused on this)
  • The update will be done safely
Compare SW Updates tools

- [https://wiki.linuxfoundation.org/civilinfrastructureplatform/cip_comparison_report](https://wiki.linuxfoundation.org/civilinfrastructureplatform/cip_comparison_report)

- **Client software**
  - SWUpdate + librsync
  - RAUC + Casync
  - meta-updater

- **Server software**
  - hawkBit
  - Mender.IO server
  - HERE OTA community
  - Custom Http server

Select for our initial prototype
SWUpdate (Client)

- Software update tool for embedded system
- Repository
  - https://github.com/sbabic/swupdate
- Documentation
- Support functions we want to integrate into our update mechanism
  - Update from a server (also from a local media)
  - Easy to customize the update steps using handlers
  - binary delta update using librsync
  - safe update using “signed update” and “encrypted update”
SWUpdate - Update image (swu)

- sw-description
- sw-description.sig (for signed update)
- sub-images

```plaintext
software = {
    version = "0.1.0";
    hardware-compatibility: ["1.0"];  
    images: ({
        filename = "cip-core-image-cip-core-bbb.ext4.img.enc";
        device = "mmcblk0p2,mmcblk0p3";
        type = "roundrobin";
        sha256 = "92847698c23408bd7ec34a4d9295ca5366d15a3...";
        encrypted = true;
    });
}(e.g. raw update with encrypted update)
```
hawkBit (Server)

- Domain independent back end solution for software update
- Repository
  - https://github.com/eclipse/hawkbit
- Documentation
  - https://www.eclipse.org/hawkbit/
  - https://projects.eclipse.org/projects/iot.hawkbit
- Support functions we want to integrate into our software update mechanism
  - It has a REST API to communicate with target devices
  - It has a dashboard and you can see the update progress and status on that
Update image types and security options

• Update image types (You have to select 1 type)
  • raw update
    • The update using whole partition image.
    • Pros: You don’t have to concern about the state of the inactive partition
    • Cons: The update image is big
  • binary delta update
    • The update using delta image by librsync
    • Pros/Cons: The opposite of raw update’s pros/cons

• Security options (You can enable both of them at the same time)
  • signed update
  • encrypted update
Basic update overview - Build system

sub-image

sw-description

SWU
Basic update overview - Server and Client

- Server and Client

1. Download
2. Check SWU
3. Install
4. Storage
Basic update overview - Attack

Intrudes and replace a sub-image to an arbitrary one
Basic update overview - Attack
Signed update overview

- **write hash value**
- **sub-image**
- **public key**
- **check swu**
- **download**
- **check hash value**
- **install**
- **storage**

Diagram:
- swu
- sw-description
- sw-description.sig
- download
- upload
- same
Signed update overview - Developer

private key

public key
Signed update overview - Build system

write hash value

private key

sw-description

sw-description.sig

sub-image

swu
Signed update overview - Server and Client
Signed update overview - Attack

Attacker Intrudes and replace a sub-image to an arbitrary one.
Signed update overview - Attack

**Attacker**

- Intrudes and replaces a sub-image to an arbitrary one.
- Calculates the hash value and writes it to `sw-description`.
Signed update overview - Attack

- Attack
- public key
- same
- storage
Signed update overview - Attack

private key

calculate the hash value and write it to sw-description

Intrudes and replace a sub-image to an arbitrary one

Attacker
Encrypted update overview
Encrypted update overview - Developer

symmetric key

CIP Mini Summit 2019
Encrypted update overview - Build system

Symmetric key

Sub-image

Sub-image.enc

Sw-description

Swu
Encrypted update overview - Server and Client
Encrypted update overview - Attack

Attacker

See the contents

SWU

hawkBit
Safe update overview:

- **symmetric key**
- **sub-image**
- **write hash value**
- **sub-image. enc**
- **sw-description**
- **private key**
- **sw-description. sig**
- **hawkBit**
- **download**
- **public key**
- **check swu**
- **check hash value**
- **upload**
- **same**
- **symmetric key**
- **sub-image**
- **install**
- **storage**

The diagram illustrates the process of a safe update, including key interactions, file processing, and hash checks to ensure integrity and security.
Future work
2nd roadmap iteration

(from the beginning of Aug 2019 to the end of Jan 2020)

1. Integrate safe update features into the software update mechanism
   • It has been already done

2. Make it easier to contribute to our WG
   • Clarify current our tasks
   • Provide how to prepare an environment for developing and testing

3. Work on remaining tasks for the software update mechanism

4. Try to integrate the software update mechanism into CIP Core and CIP Testing properly
   • For CIP Core: add a recipe to build an update image
   • For CIP Testing: test the software update mechanism on several reference boards continuously
Summary
Summary

• SW Updates WG has been established to provide the CIP reference software update mechanism
• As an initial prototype, we select SWUpdate and hawkBit
• It supports the following functions
  • raw and binary delta update
  • Safe update by signed and encrypted update
• Future work
  • Clarify remaining tasks and work on them
  • Try to integrate the software update mechanism into CIP Core and CIP Testing properly
Thank you!