IoT devices: secure boot and sw maintenance.

Open IoT Summit
Igor Stoppa
Ostro Project

The topics and solutions discussed in this presentation stem from the Development of the Ostro Project.

Each topic is meaningful also per-se, applied to a generic Linux distro.

However, they coalesce and find mutual support under the broader umbrella of creating the Ostro OS.

https://ostroproject.org/
What is an “IoT device”?

“Physical object that features an IP address for internet connectivity and communicates with other similar objects and/or other internet-enabled devices and systems.”
IoT devices - old and new challenges

Usual maintenance tasks:

- Deploy a device
- Install/Remove/Update sw components.
- Manage its configuration.

Reliably and safely.
What makes the difference

Growing disparity between number of devices and administrators.

Critical services must work reliably, unattended ... 
... but new functionality / interactions must be easy to add.

As the same time, systems become more complex, what was running an RTOS, now runs Linux, their attack surface grows.

There is need for a simplified mechanisms to deliver/update SW components frequently.
“SW Update” is the sw management solution used by Clear Linux.

Clear Linux:
- Linux distribution maintained by Intel
- focused on Cloud
- showcase for new technologies
- specifically tweaked for Intel HW

https://clearlinux.org/
What’s wrong with packages?

- Typically, Linux distributions rely on SW packages.
- Packages have fine granularity: It takes several packages for deploying a macro functionality.
- Opens up possibility for package clashes or mismatches.

Doable, but not trivial and requires expertise.
What is a Bundle?

- A set of files, taken from an installation. It provides a high-level functionality, handled in one logical unit.

- Each Bundle is defined through a manifest. Each manifest contains a reference and a checksum, for each file that belongs to the bundle.

- Bundles can partially overlap, for example when including shared libraries.
Example of Bundles

- **Core OS Bundle (Unremovable)**: Bundle A
- **Optional Bundles (removable)**: Bundles B, C, D
- **Bundle 1**: A_i, D, C_j
- **Bundle 2**: B

A_i = cannot be removed
D, C_j = optional bundles
B = purged when both D, and C_j are removed

Bundle 3
SW Management - Bundles vs Files

- Files are hashed and handled through all the update process with their hash.
- The hash calculation depends also on the timestamp of the file and its attributes: both plain and extended.
- When a Bundle includes a file, the reference is placed in the manifest corresponding to the bundle.
- Using information from the manifests, it is possible to verify and restore the files belonging to the bundles installed.
SW Management - SW Updates

- Given two releases of the same SW, a binary diff-ing phase generates a set of deltas, containing all the changes detected.

- There is always an update available, from the “version 0” baseline. It means a full update and assumes that everything must be deployed anew.

- For each release, it is possible to define which others can update to it through binary delta updates. Lack of binary delta update means having to download the full version of those files that have changed.
● Each version provides a direct upgrade path from the previous 2
● Each version can also be achieved with a full update from “Version 0”
The Ostro Spin of SW Update

- Ostro provides:
  - Tools, sources, SSTATE cache to OEMs/ODMs for creating and signing own distro - Ex: Home automation system, Home Gateway, etc.
  - Reference set of images, for evaluation/early development. These images are built by Intel and are not meant for direct deployment to consumer devices.
  - The SSTATE from Ostro is not signed, so the OEM/ODM can reuse it and apply own signature to the swuppd stream.
Disk Image Layout

<table>
<thead>
<tr>
<th>Partition</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT</td>
<td></td>
</tr>
<tr>
<td>EFI 0xEF00</td>
<td></td>
</tr>
<tr>
<td>Alternate EFI 0x2700</td>
<td></td>
</tr>
<tr>
<td>RootFS 0x8300</td>
<td></td>
</tr>
<tr>
<td>Backup GPT</td>
<td></td>
</tr>
</tbody>
</table>

Partition Table

Active Boot Partition (Recognised by the BIOS)
Alternate Boot Partition (NOT recognised by the BIOS)
Active Boot Partition (Recognised by the BIOS)
Single EFI application: Typical System

- EFI BIOS
  - Load, Validate & Execute
- EFI BootLoader
  - Load & Validate
  - Load & Validate
- Kernel Command Line
- Initramfs
  - Pivot Root
  - Execute
- Rootfs
- Kernel
Single EFI application (Ostro)

EFI BIOS

Load, Validate & Execute

EFI Stub

Kernel Command Line

Single, signed UEFI application

Kernel

Initramfs

Rootfs

Root

Pivot

YOCTO LINUX KERNEL ZEPHYR INTEL GRAPHICS FOR LINUX QT DPDK BLUEZ OSTRO GNOME CHROMIUM IOTIVITY Wayland Soletta
Disk Image Layout - EFI App update - 1

- **GPT** (Partition Table)
- **EFI** (0xEF00)
- **Alternate EFI** (0x2700)
- **RootFS** (0x8300)
- **Backup GPT**

Contains the EFI application file tracked by swupd
Disk Image Layout - EFI App update - 2

GPT

EFI
0xEF00

Alternate EFI
0x2700

RootFS
0x8300

Partition Table

Still old EFI App

After SW Update pass, new App
Disk Image Layout - EFI App update - 3

<table>
<thead>
<tr>
<th>GPT</th>
<th>0x8300</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFI</td>
<td>0xEF00</td>
</tr>
<tr>
<td>Backup GPT</td>
<td></td>
</tr>
</tbody>
</table>

Partition Table

1) New App copied to Alternate EFI
2) Swap types of Alternate and “real” EFI partitions
3) reboot

After SW Update pass, new App
Next Steps

- Consolidate and merge upstream
  - Ostro image layout handling (to OE)
  - SW Update (separate layer in OE)
  - Handling of EFI single binary (to Clear Linux)

- Additional hardening (recover rootfs from initramfs)
Q/A
Thank You!