Getting started with RAUC

Kamel Bouhara
kamel.bouhara@bootlin.com
- Embedded Linux engineer at Bootlin
  - Embedded Linux **expertise**
  - Bootloader, Linux kernel, Yocto Project, Buildroot
  - Complete Linux BSP development
  - Strong open-source focus: upstreaming and contributions
- Significant experience in board bring up with NXP SoCs
- Small contributions to Yocto Project, Buildroot and Linux
- Living in **Lyon**, France
About this talk

- Brief introduction to RAUC
- Explore RAUC integration with Yocto/Buildroot, U-Boot/Barebox
- Share some insights and field experience
- Give some important keys to be ready to start with RAUC
What is RAUC?

- RAUC is an update system mechanism
- Developed by Pengutronix (Barebox, PTXDist)
- Licensed under LGPL-2.1
- [https://rauc.io/](https://rauc.io/)
Non exhaustive list of RAUC features

- Fail-safe and atomic
  - Using symmetric or asymmetric update (redundant A/B scheme, rescue system)
  - Update result is either valid or invalid (interrupt, corrupted bundle, device not compatible)

- Security
  - Using OpenSSL x.509 certificates
  - Using an Hardware Secure Module with PKCS11 keys

- Flexible and customizable update with bundle handlers
What's a RAUC bundle artifact?

Image updates deployed on each slot partitions

5 type of sections: update, bundle, hooks, handler and image

signed squashfs image (.raucb)

manifest file (.raucm)
(checksum can be used to decide if update is required with install-same=false)
Getting started with RAUC

RAUC and Build systems
Yocto integration

- meta-rauc layer
  - Mainly provide a generic class for bundle generation
  - A fragment file `rauc.cfg` for the kernel squashfs support
  - A basic example of bundle recipe `core-bundle-minimal.bb`
  - The rauc package provide support for both sysVinit and systemd
  - A script to generate development certificate with openssl
  - Some useful packages: dt-utils, casync, rauc-hawkbit ...
  - [https://github.com/rauc/meta-rauc](https://github.com/rauc/meta-rauc)
Yocto integration

- Add the `meta-rauc` layer to your `bblayers.conf`:

  ```
  $ git clone https://github.com/rauc/meta-rauc.git
  $ bitbake-layers add-layer meta-rauc
  ```

- Create a recipe `rauc\_%\..bbappend` to install your own rauc configuration:

  ```
  FILESEXTRAPATHS_prepend := "\${THISDIR}/files:
  SRC_URI_append := " file://system.conf"
  ```
Yocto integration

Setup a minimal rauc configuration through packageconfig:

```bash
PACKAGECONFIG_remove_pn-rauc = "service"
PACKAGECONFIG_remove_pn-rauc = "network"
PACKAGECONFIG_remove_pn-rauc = "gpt"
```

See `rauc.inc` for the exhaustive list of packageconfig
A bundle recipe

Create your bundle recipe demoboard-bundle.bb

Inherit bundle

RAUC_BUNDLE_COMPATIBLE ?= "Demo Board"
RAUC_BUNDLE_SLOTS ?= "rootfs"
RAUC_SLOT_rootfs ?= "core-image-minimal"
RAUC_IMAGE_FSTYPE = "ubifs"

RAUC_KEY_FILE = ":${YOCTOROOT}/meta-demoboard/keys/dev.key.pem"
RAUC_CERT_FILE = ":${YOCTOROOT}/meta-demoboard/keys/dev.cert.pem"

RAUC_BUNDLE_COMPATIBLE the target compatible
RAUC_BUNDLE_SLOTS list of partitions to update
RAUC_IMAGE_FSTYPE root filesystem (ubifs, squashfs, ext4, vfat, raw)
Building and Using

▶ Add the rauc package to your image

```
IMAGE_INSTALL_append = "rauc"
```

▶ Generate your bundle image

```
$ bitbake demoboard-bundle
```

▶ On the target, install the generated *.raucb using `rauc install` command

▶ Run `rauc status mark-good` to validate the boot on the new slot (shall be done by an initscript or systemd service)
Using RAUC with Buildroot

- Select `BR2_PACKAGE_RAUC=y`
- Options enabled as dependencies:

```bash
select BR2_PACKAGE_SQUASHFS # run-time dependency
select BR2_PACKAGE_UBOOT_TOOLS if BR2_TARGET_UBOOT
select BR2_PACKAGE_UBOOT_TOOLS_FWPRINTENV if BR2_TARGET_UBOOT
```

- To deploy rauc files on target use `BR2_ROOTFS_OVERLAY`

```bash
board/.../demoboard/rootfs-overlay/
  etc
    rauc
      dev.cert.pem
      dev.key.pem
      system.conf
```
Generate the bundle with a post image script using BR2_ROOTFS_POST_IMAGE_SCRIPT

```bash
#!/bin/bash
...
cat >> ${BINARIES_DIR}/temp-dir/manifest.raucm

[update]
compatible=demo-board
version=${VERSION}

[image.rootfs]
filename=rootfs.ext4

EOF

${HOST_DIR}/bin/rauc --cert ${BOARD_DIR}/dev.cert.pem
    --key ${BOARD_DIR}/dev.key.pem
    bundle ${BINARIES_DIR}/temp-dir/
    ${BINARIES_DIR}/bundle.raucb
```
Getting started with RAUC

RAUC and Bootloaders
Barebox and RAUC: Pre-requisites

Hardware pre-requisite:

- A non-volatile memory with ~200 KBytes of dedicate space (not updated with Barebox)

Software pre-requisite:

- Install `dt-utils` on your filesystem from: https://git.pengutronix.de/cgit/tools/dt-utils
- If using EEPROM backend make sure you have the following kernel patch (nvem core): https://lkml.org/lkml/2020/4/6/445
Barebox and RAUC: In brief

- Enable bootchooser and barebox state support:
  
  ```
  CONFIG_STATE_DRV=y
  CONFIG_STATE=y
  CONFIG_BOOTCHOOSER=y
  CONFIG_CMD_STATE=y
  CONFIG_CMD_BOOTCHOOSER=y
  ```

- The **Bootchooser** is the algorithm implemented in Barebox to provide a mean to work with abstract boot targets.

- The **State** allows storing the set of variables required by RAUC.
Example of A/B update scenario setup:

$ tree arch/arm/boards/demoboard/env/nv/ |grep boot
  bootchooser.disable_on_zero_attempts
  bootchooser.reset_attempts
  bootchooser.reset_priorities
  bootchooser.retry
  bootchooser.state_prefix
  bootchooser.system0.boot
  bootchooser.system0.default_attempts
  bootchooser.system0.default_priority
  bootchooser.system1.boot
  bootchooser.system1.default_attempts
  bootchooser.system1.default_priority
  bootchooser.targets
  boot.default

▶ Double check barebox dts and state.prefix above!
U-Boot and RAUC: Pre-requisites

Hardware pre-requisites:
- A non-volatile memory with ~200 KBytes of dedicate space (not updated with u-boot)

Software pre-requisites:
- Install U-boot *fw-utils* on your filesystem, define u-boot environment offset in */etc/fw_env.config*
- Use *CONFIG_ENV_IS_IN_* and/or *CONFIG_SYS_REDUNDAND_ENVIRONMENT=y* when updating u-boot
Updating with U-Boot

▶ Example of boot script:

https://github.com/rauc/rauc/blob/master/contrib/uboot.sh

▶ Mainly based on three variables:

- **BOOT_ORDER** Which slot to boot first
- **BOOT_*_LEFT** Counters for boot attempts on A/B slots

▶ On target, load and run the boot script:

```bash
setenv loadscript "fatload mmc ${mmcdev}:${mmcpart} ${loadaddr} ${script};"
run loadscript;
source ${loadaddr}:${script};
```
Basic update scenarios
Update with a rescue system or asymmetric update

- Good solution for devices with minimal storage resource
- No fallback possible, require to define a backup plan when update failed
- Several reboots required to achieve the update
Example of rauc usage from initramfs

```bash
#!/bin/sh

PATH=/sbin:/bin:/usr/sbin:/usr/bin
USB=/mnt

# Sync clock on RTC
hwclock -s

# Attach ubi rootfs volume
ubiattach /dev/ubi_ctrl -m 2

...  

# Install the new system image
rauc install "\${USB}/demoboard-bundle-demo-board.raucb"

# Change the active boot slot and reboot on the main system
rauc status mark-active rootfs.1

reboot -f

exit 0
```

▶ Make sure the system clock is correctly setup (use RTC/hwclock -s or NTP)

LastError: signature verification failed: Verify error:self signed certificate
Installing `/media/demoboard-bundle-demo-board.raucb` failed
Good solution for devices with large size storage
Use the *parent* entry to bind all slots together in a single bundle update
Depending on the application it can be a complex scenario, use post-install script handlers wisely
Conclusion
What you need to know

- Both Yocto or Buildroot fully support RAUC
- Well integrated in Barebox (developed by the same Pengutronix folks)
- With Barebox you don’t need to directly deal with environment variables
- U-Boot is good enough for a simple redundant A/B scenario
- More complex scenario need modification in rauc bootchooser code
  (`uboot_get_state/uboot_set_state`)
- Make sure your device is well sized for your update strategy and application requirements
- New RAUC version 1.5 supporting the verity format for verified boot
Questions? Suggestions? Comments?

Kamel Bouhara
kamel.bouhara@bootlin.com

Slides under CC-BY-SA 3.0
https://bootlin.com/pub/conferences/2021/lee/bouhara-rauc