# Open Source Foundries

## Zephyr and MCUBoot as Foundations for Secure IOT Products

Marti Bolivar
marti@opensourcefoundries.com



# Zephyr and MCUboot for the Impatient



- Batteries-included open source RTOS
- https://www.zephyrproject.org/

- Secure boot for 32-bit MCUs (mynewt, Zephyr, RIOT*)
- https://mcuboot.com/

* Applications only, not bootloader itself

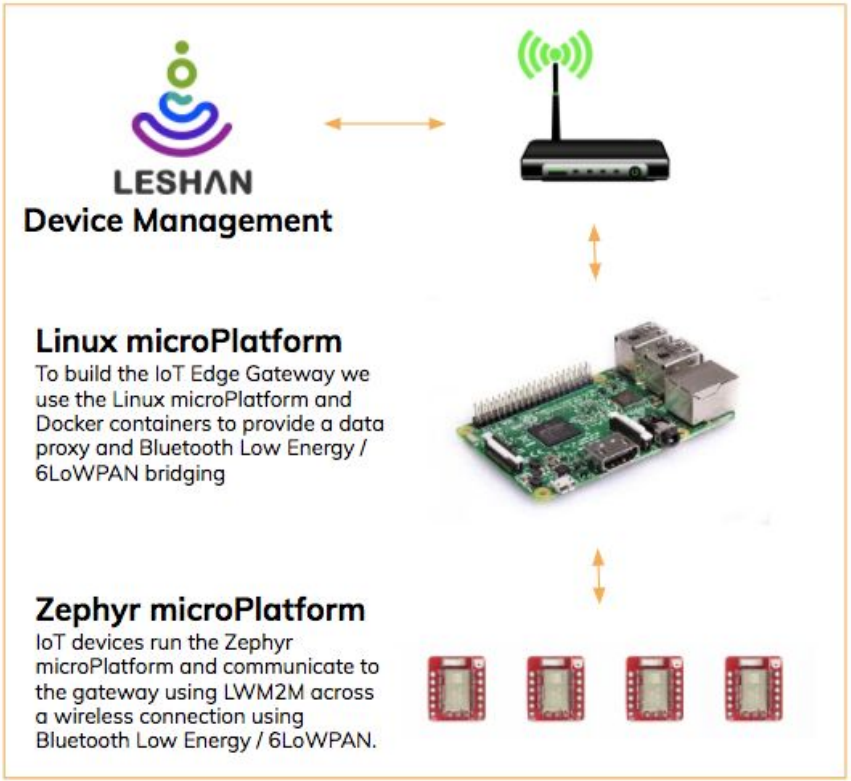
# What you'll get from this talk

- Information on how to combine Zephyr, MCUboot, and additional reference hardware and software as a basis for developing secure, updatable, connected devices.

- A walk-through of a concrete, open source IoT application based on Zephyr, MCUBoot, and LWM2M, as well as the developer tooling and security model underlying it.

  The emphasis is practical: what is it, how do you build it, features and limitations, etc.

# System Overview

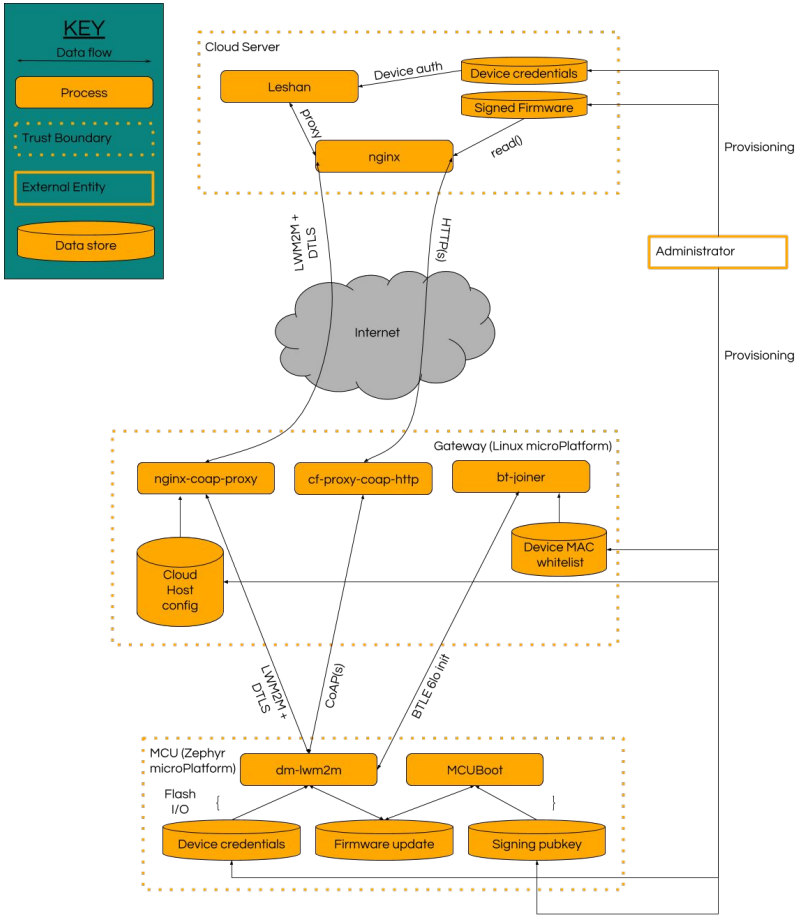



- Eclipse Leshan in a container for getting started with LWM2M on the cloud
- Gateway binary builds for RPi3 and other gateway boards, with sources for OE/Yocto layers to build and customize yourself
- Works with multiple Zephyr-supported devices and boards, including Nordic nRF52, NXP K64

# System Block Diagram

Dataflow diagram for threat modeling (and later reference).

We'll focus on the connected device at bottom; securing the server and the gateway are out of scope for this talk.



# Build Demo in a Docker container

```
# Dockerfile:
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y --no-install-recommends
software-properties-common
RUN add-apt-repository ppa:osf-maintainers/ppa \
    && apt-get update \
    && apt-get install -y --no-install-recommends zmp-dev \
    && pip3 install --user pyelftools cryptography intelhex
RUN git config --global user.name "Example Hacker" \
    && git config --global user.email "hacker@example.com"
COPY ./get-zmp.sh /tmp/get-zmp.sh

# get-zmp.sh contains:
mkdir zmp && cd zmp
repo init -u https://github.com/OpenSourceFoundries/zmp-manifest
repo sync
./zmp build -b nrf52_blenano2 zephyr-fota-samples/dm-lwm2m
```

# System Live Demo

# Extending the System

Many standard objects are already defined for LWM2M by OMA:
http://www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html

Objects are identified as **<OBJECT_ID>/<INSTANCE_ID>/<RESOURCE_ID>.**

Example OMA-defined objects:

- Security, access control
- Device attributes
- Connectivity monitoring
- Firmware update
- Location
- Lock and wipe
- ...

Example third-party and vendor-defined objects:

- Digital and analog I/O
- Numerous sensor and actuator types
- Power, voltage, current, frequency
- Accelerometers, magnetometers
- GPS location
- Acidity
- Alarm state
- Add your own here!

# Extending the System

To register your own objects:

1. (Optional) Configure Zephyr device driver for your hardware (many drivers available, more all the time)
2. Create an object instance using LWM2M core
3. Register callbacks with LWM2M core which implement resource handlers

Working example (without error handling) for GPIO-based light control to the right.

```
1  static struct device *led_dev;
2  static u8_t led_current;
3
4  static int on_off_cb(u16_t obj_inst_id, u8_t *data, u16_t data_len,
5                       bool last_block, size_t total_size)
6  {
7          u8_t led_val;
8
9          led_val = *data;
10         if (led_val != led_current) {
11                 gpio_pin_write(led_dev, LED_GPIO_PIN, led_val);
12                 led_current = led_val;
13                 lwm2m_engine_set_s32("3311/0/5852", 0);
14         }
15         return 0;
16 }
17
18 void init_light_control(void)
19 {
20         led_dev = device_get_binding(LED_GPIO_PORT);
21         gpio_pin_configure(led_dev, LED_GPIO_PIN, GPIO_DIR_OUT);
22         gpio_pin_write(led_dev, LED_GPIO_PIN, 0);
23
24         lwm2m_engine_create_obj_inst("3311/0");
25         lwm2m_engine_register_post_write_callback("3311/0/5850",
26                                                   on_off_cb);
27 }
```

# Extending the System

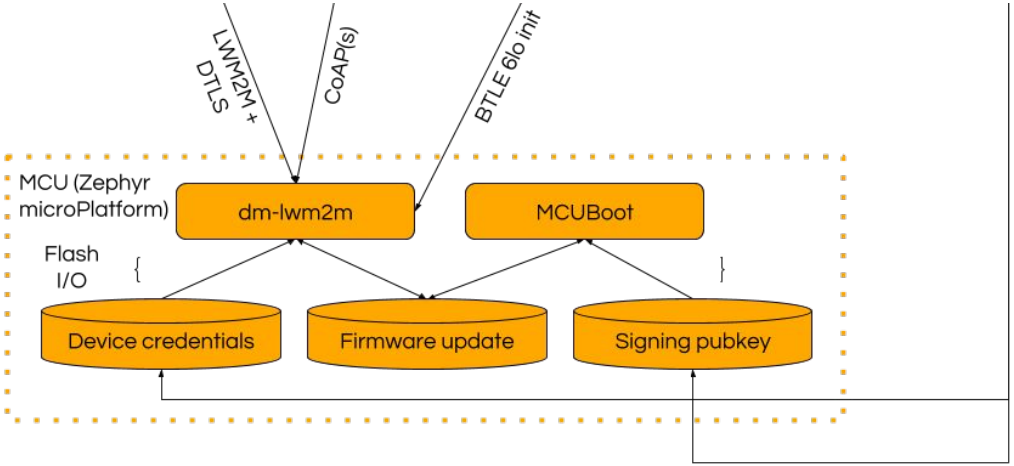
Other things to think about (other than gateway/cloud):

- Customizing the network configuration to your needs
- Setting up BT MAC addresses and other identifiers
- Bluetooth security configuration
- Key management for MCUboot and DTLS
- Porting to a different…
  - SoC
  - Board
  - Network device
- Certification

(All of these are up to you, but have configuration settings or OSF/vendor/Zephyr/MCUboot documentation.)

# Threat Modeling

# Threat Model: Attack Surfaces




- Physical access (during provisioning, in the field, etc.)
- Bluetooth link (network link initialization, usage, etc.)
- LwM2M and CoAP block transfer (device to cloud)

# Threat Model: Physical Access

- Difficult to make portable assumptions here; this is mostly left to the user
- Arm v8-M support is being added to Zephyr, which will improve this situation on some targets
- Example threats to consider:
  - Debugger-driven dumping (or modifying) stored keys on flash: at the very least, make sure you don't share PSKs across multiple devices, but also investigate any fuses etc. you can blow to disable ROM dumping, breakpoint debugging, etc.
  - Debugger-driven modifications to MCUboot / other trust anchors you may implement
  - CoAP block transfers for firmware updates are not secure by default: LWM2M PSKs are stored in a separate partition that never needs updating or sending across the cloud, but consider additional mitigations to avoid ROM dumping, other issues.
  - Taking advantage of hardware-specific key storage, if available

# Threat Model: Bluetooth Link

- For portability and general robustness, the link is untrusted
- Future work will continue using this assumption
- Zephyr does support Bluetooth security features, consider these and any others which may add to your defenses

# Threat Model: Network Communications

- Standard end-to-end web security model based on DTLS
  - You can provision per-unit keys on each device and in the cloud
  - Documentation is provided for setting this up
  - Crypto library is based on mbedTLS
- Default demo driver users a null entropy source to feed mbedTLS for portability, application developers will need to integrate with Zephyr entropy and random subsystem drivers for their platform
- Current implementation uses CoAP to gateway for firmware block transfers, making information disclosure, etc. possible with physical access to the device. This is being worked on.

(Adapted from IETF SUIT work in progress, see draft-moran-suit-architecture-01.)

# Some threats and mitigations to OTA update

| Threat | Mitigation | Notes |
|---|---|---|
| Payload type mismatch | Full binary OTA only | Substituting a .hex for a .bin in a system could cause DoS. We currently limit to full binary update only for this reason. Future work on differential updates planned, additional mitigations will be necessary. |
| Payload installation location | Immutable flash partitioning | Systems with configurable update installation locations need to verify targets. Flash partitioning is one-time per device to avoid this. |
| Payload not verified on boot | MCUboot checks image signatures | MCUboot is configured to check all image signatures by default on reset, using up to date algorithms (RSA PKCS#1 v2.2 with 2048 bit keys by default) |
| Unauthenticated updates | DTLS server auth | Update URIs are set via an end-to-end secure channel. However, gateway CoAP block transfer is not secure by default. Also consider redirects. |

(Adapted from IETF SUIT work in progress, see draft-moran-suit-architecture-01.)

# Additional Threats to Consider

| Threat | Notes |
|---|---|
| Rollback to old firmware | MCUboot doesn't currently provide anti-rollback mechanisms, making this an application-level issue. This is being worked on. |
| Offline device + old firmware | Freshness guarantees generally rely on a secure source of time, which aren't currently supported by MCUboot. Users with RTCs etc. may want additional protections here. |
| Mismatched firmware | MCUboot doesn't support robust vendor, device, deployment context, etc. model for firmware images. Users must prevent deployment of mismatched firmware. |
| Unqualified firmware | MCUboot's permissions model is simple, and doesn't natively support a permissions model with multiple roles (OEM versus network operator, e.g.), delegation, thresholding, etc. |
| URI redirection | This is potentially a concern at the gateway/device CoAP block transfer layer. Updates can cross internet via HTTPs, but currently are proxied via CoAP after that. |

# Security-Related Roadmap Items

- Arm v8-M: initial support is now part of Zephyr 1.11
  - Core support for Cortex-M23, Cortex-M33 CPUs
  - Additional secure element targeted at use cases like secure storage
  - SoC support for nRF91, implementing Cortex-M33, is expected
- Zephyr has initial user-space support, designed for memory isolation of application threads from kernel objects.
  - On x86, uses the MMU
  - On ARM and others, uses MPU

# Thank you


OPEN SOURCE
FOUNDRIES


Embedded Linux
Conference


OpenIoTSummit