IOT Message Security

DI Markus Haslinger MSc

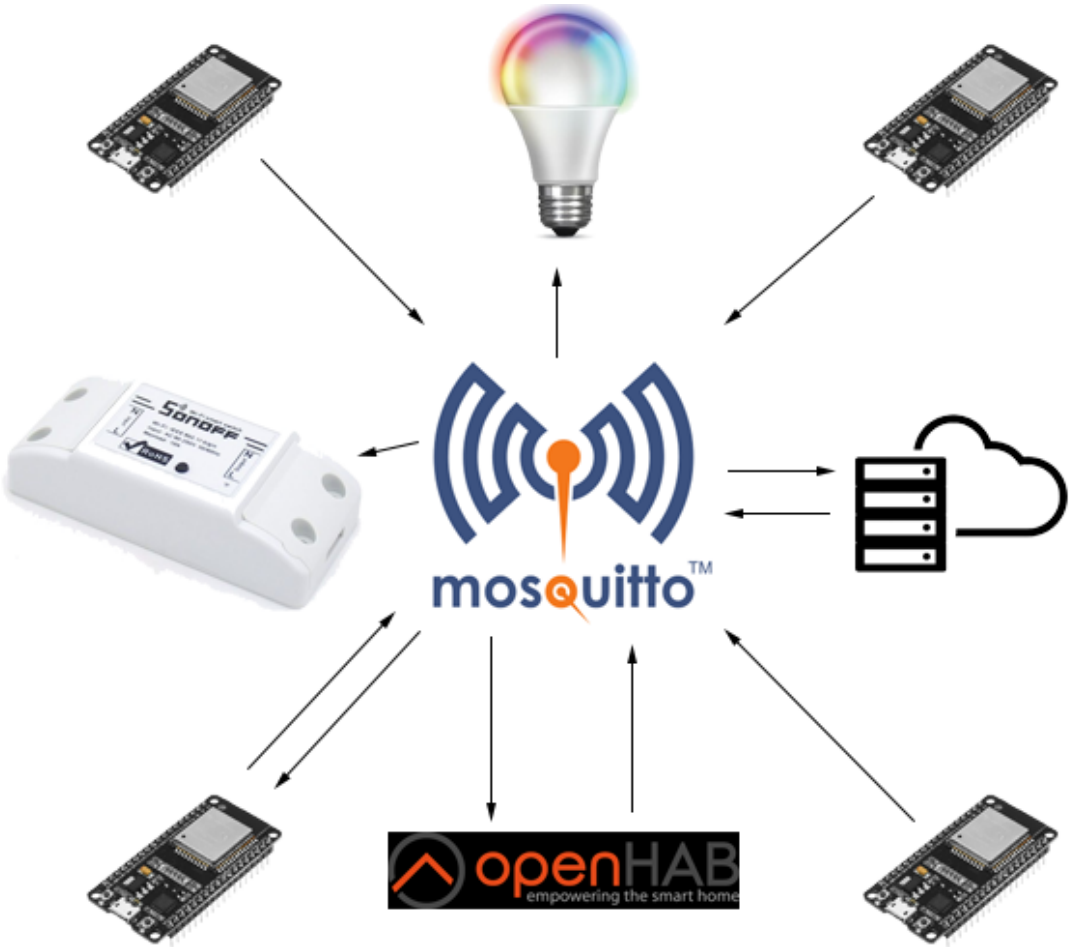
FF IOT

HTL LEONDING

Agenda

HTL LEONDING

Typical Setup



HTL LEONDING

Typical Setup

- Several, different devices send sensor data
- A broker receives and distributes the data
- Other devices and services
  - Evaluate the data
  - React on the data
- Messages are sent
  - In a binary or text format (e.g. JSON)
  - Unencrypted
    - To be consumed by a multitude of applications
    - (for performance reasons)

HTL LEONDING

Unencrypted messages

- Messages are usually sent as plain text or in an otherwise machine readable format
- This increases interoperability
- Encryption is a comperatively performance intensive operation
- IOT devices[1]
  - Are performance constrained
  - Are power constrained ⇒ even if the processing power is there it will drain batteries faster
- But the messages can be read by anyone

[1] despite being much more capable than a few years ago

HTL LEONDING

Unencrypted messages

- Sending messages unencrypted makes sense
- But the messages can be read by anyone
- Do we care?
  - You might not mind someone knowing the temperature of your living room
  - But what about a webcam feed of you in the pool?
  - Or a burglar checking when you are not home and a window is open?
- It seems wise to apply at least some level of protection

HTL LEONDING

Basic Security

- Let's think about home automation
- The easiest way is to leverage some available security
- Our devices are connected to our WLAN
- So enabling something like WPA2 will lock others out
  - Which you should do always, anyway
- We don't have to change anything at our IOT devices and the messages can't be read outside the network
- Are we done now?

HTL LEONDING

Issues with basic security

- Locking your WLAN might be enough for your home
- But we are training you to design business solutions
- Imagine a pipeline with temperature and leakage sensors
  - IOT devices spread over thousands of kilometers
  - Messages transfered via cellular networks
  - MQTT broker running in a server farm somewhere
  - Consumer applications located around the world
- What are you going to do?

HTL LEONDING

Increasing security

- Assume we cannot fully control who listens to the network we use to send our messages
- The next logical step is to encrypt the messages themselves
- But we want to preserve interoperability
- So encrypting the communcation channel is better than encrypting the content
- Using something widespread and well tested to do that sounds reasonable

HTL LEONDING

Example – TLS for MQTT

- Following is a short demo on how to enable TLS for the Mosquitto MQTT broker
- We are then connecting an IOT device flashed with the popular Tasmota firmware
- Using two kinds of certificates:
  1. Self Signed
  2. Let's Encrypt
- Finally we'll hook up Node-RED as well

HTL LEONDING

Initial Setup

- We will start with self signed certificates
- The example assumes the following setup:
  - IOT Stack for Raspberry Pi
    - Portainer
    - Mosquitto
    - Node-RED
  - Sonoff POW ready to be flashed with Tasmota firmware
- **A step-by-step tutorial including console commands is provided separately**

HTL LEONDING

Preparing the server

1. Create CA & server certificates (e.g. using openssl)
2. Create a folder for the certificates in the IOT stack mosquitto directory
3. Update docker compose
   - Map the certificates folder
   - Add a port mapping for MQTT secure port 8883
4. Update mosquitto.conf
   - Add a listener for port 8883
   - Setup paths to certificate files
   - Select TLS version 1.2[2]

[2] 1.3 not yet supported by Tasmota

HTL LEONDING

Tasmota Setup

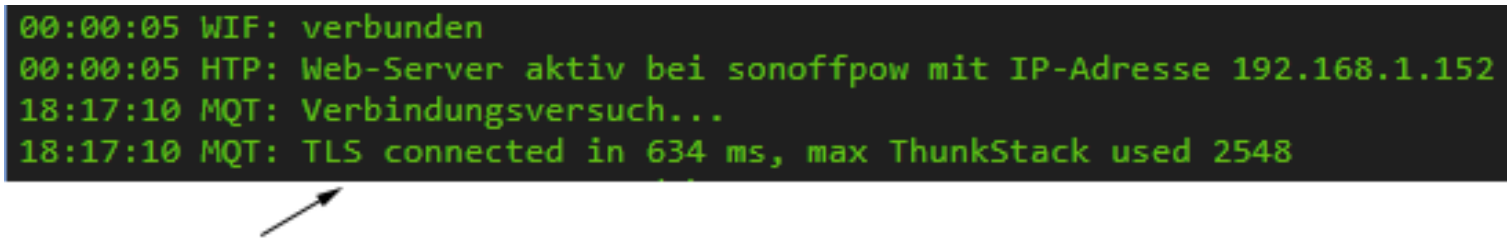
1. Enabling TLS requires us to compile the firmware ourselves with some additional flags
   - Clone from Github[3]
   - Compile using Visual Studio Code + PlatformIO
2. In the `user_config_override.h` file
   - add `#define USE_MQTT_TLS`
   - add `#undef MQTT_PORT`
   - add `#define MQTT_PORT 8883`
3. In the `platformio_override.ini` file
   - enable `-DUSE_CONFIG_OVERRIDE` build flags
4. Flash newly compiled firmware to device

[3] `https://github.com/arendst/Tasmota`

HTL LEONDING

Tasmota Setup

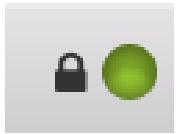
- If setup correctly the Tasmota console should show something like this:



HTL LEONDING

Client Setup

- For MQTT.FX
  - Switch port to 8883
  - In the SSL/TLS Tab `enable SSL/TLS`
  - Use `CA certificate file` and point to your created CA certificate
  - You should now see a lock symbol at the connection status:



- For Node-RED
  - Update mqtt-broker-config
  - Select `enable secure connection`
  - Update/Add TLS configuration
  - **Remove** `verify server certificate`
  - Change port to 8883

HTL LEONDING

Pros & Cons

- Can be done entirely on your own
- Works within any network
- Messages are sent encrypted
- Computation and memory usage is usually acceptable
  - If the environment warrants an encrypted transmission
- You need to recompile the firmware yourself
  - But support is integrated into Tasmota and activation is relatively easy
- Certificates cannot be validated

HTL LEONDING

General

- I suppose you are familiar with Let's Encrypt
- Instead of creating our own fake CA we can use Let's Encrypt
- The process is similar to using self signed certificates
- But we need a domain (at least DynDNS)
- In return we get certficates which can be validated
- Tasmota supports Let's Encrypt certificates

HTL LEONDING

Retrieving the certificate

- Let's Encrypt performs its 'challenge' via port 80, so that has to be available as well
  - At least temporarily
- Use certbot: `sudo certbot certonly --standalone`
- This will yield you four files:
  - cert.pem (needed)
  - chain.pem (needed)
  - privkey.pem (needed)
  - fullchain.pem (ignore)
- Use the three files instead of the self signed ones

HTL LEONDING

Tasmota Setup

- Once again we need to compile the firmware ourselves using yet another flag
- Add to the `user_config_override.h` file
  - `#define USE_MQTT_TLS_CA_CERT`
- Use the domain name as host
- Use the actual port (8883 or a forwarded one)

HTL LEONDING

Client Setup

- For MQTT.FX
  - Change broker host to domain
  - (Change port)
  - Switch `CA signed server certificate`
- For Node-RED
  - Change server to domain
  - (Change port)
  - Change TLS config:
    - Remove the CA certificate
    - Tick `verify server certificate`

HTL LEONDING

Pros & Cons

- Similar to self signed certificates
- But proper certificates which can be validated by devices
- Not yet an automated process for certficate renewal (valid for 3 months)
- Requires a domain name
- Might be more trouble than worth in a local network

HTL LEONDING

What we are still missing

- Our messages are now transported over an encrypted connection
  - $\Rightarrow$ Eavesdropping has become much more difficult
- Yet our clients are still not authenticated
  - $\Rightarrow$ Intruders can still send malicious messages
  - Fixing that would be the next step
  - But has to be supported by device firmware

HTL LEONDING