System Security (201700086) IoT #3 - (bonus) - build your own secure firmware

Æde Symen Hoekstra, Luigi Coniglio (Group 24)

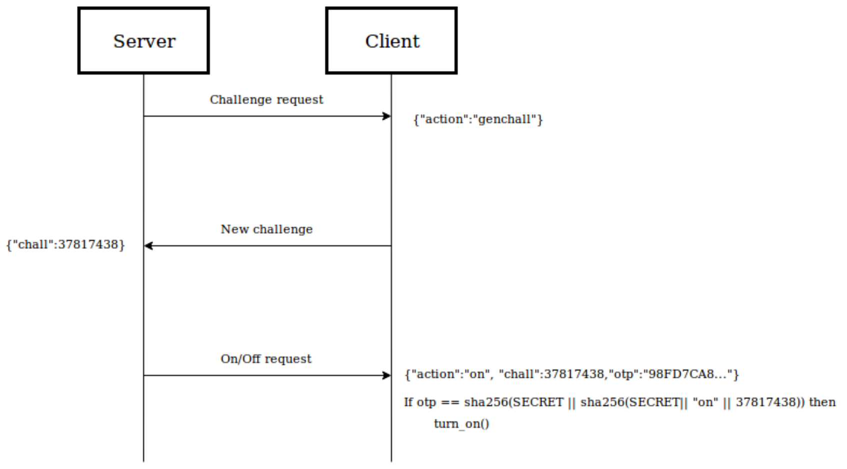
May 31, 2018

1 Setup

For this assignment we implemented a simple micropython-based firmware for the esp8266, which connects to a server and receives authenticated commands. The client code can be run on the esp8266 as well as the micropython "unix" port (however the user needs to specify the target interpreter setting the global variable *__esp8266__*).

2 Protocol

Figure 1 illustrates the main steps of the protocol used by the server to "submit" a turn on/off request to the client. In order to ask to the client to turn on/off the server will first request a challenge (which consist of a random 32 bit integer), it will then generate a one time password (otp) and send the actual command. The otp is generated as an HMAC on the shared secret value and challenge. Upon reception the client verifies the correctness of the otp and, in case of success, it executes the command sent by the server.
In order for an attacker to send a valid command he should be able to craft a valid otp, however given the size of the key it is practically impossible to craft a valid request. Each challenge is invalidated after every usage attempt (thus avoiding replay attacks and random guessing).



```
{"action":"genchall"}
```

```
{"chall":37817438}
```

```
{"action":"on", "chall":37817438,"otp":"98FD7CA8..."}
If otp == sha256(SECRET || sha256(SECRET|| "on" || 37817438)) then
    turn_on()
```

Figure 1: Secure communication protocol