Day One with a TTIG-868

Hardwear.io
2019

brian@security-bits.de ~~ www.security-bits.de ~~ @BadgeWizard

About Me

- Brian
  - @BadgeWizard
  - brian@security-bits.de

- Security Researcher / Hacker
  - Officially: “Incident Response”
- Hardware-, Embedded-, a bit of Telko-Security

“Day One” - A Single Day

- Security is
  - Expensive
  - Hard
  - Takes a long time
- Typical excuses why devices are not tested for security
- But...
  - Quick security checks, like presented here, are easy, simple and at least ensure a baseline

Department for
Digital, Culture,
Media & Sport

Guidance

Code of Practice for consumer IoT security

Published 14 October 2018

Meanwhile, in the UK

Requirements

- No default passwords
- Implement a vulnerability disclosure policy
- Keep software updated
- Securely store credentials and security-sensitive data
- Communicate securely
- Minimise exposed attack surfaces
- Ensure software integrity
- Ensure that personal data is protected
- Make systems resilient to outages
- Monitor system telemetry data
- Make it easy for consumers to delete personal data
- Make installation and maintenance of devices easy
- Validate input data

ToDo List For The Day

- Do homework
  - Search for documentation and understand it
- Read Manual
- Set up device (according to manual)
  - Document with screenshots and notes
- Use device
  - Sniff communication and evaluate
- Check communication for flaws
  - MitM, plaintext, attack where possible!
- Portscan
- Assess open ports
- Open device
  - Check for available debug ports
  - Extract and analyze data?
- Find firmware if available
  - Search for typical issues
- Write a documentation
- Share Results

Equipment

- Laptop
- VM with Wireshark, DNS, DHCP, NMAP, Burp
  - Basic networking
- WiFi Router
  - G.Li AR300m
- Logic analyzer
  - Saleae Logic Pro 16
  - A cheap one usually also does the job!
- Oscilloscope
  - Reichelt uni-T 2 channel digital scope
- Multimeter
- Soldering iron
- Microscope
- Camera
- SOIC Clips
  - They’re helpful but often cause problems and should be swapped regularly
- Cables, Solder, Tweezers and whatever :)

TTIG - 868

- The Things Industries LoRaWAN Indoor Gateway
  - Running on 868MHz for the European market
- Initially released Q1 2019 at a dev conference
  - Slowly but surely reaching the typical distributors since the Summer months
- ?First LoRa BTS using Semtech’s new Basic Station concept / model

Long Range

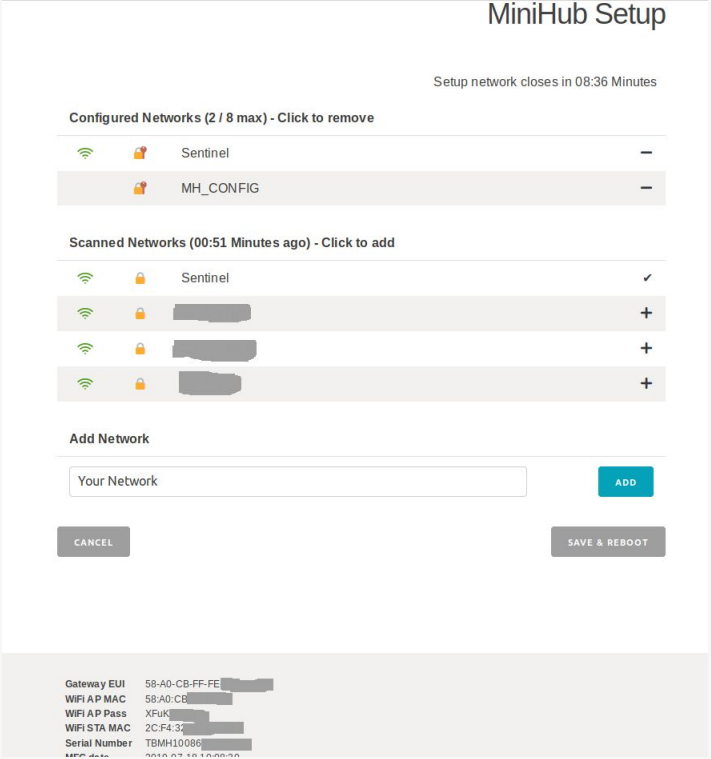
- Radio modulation / technology for long range, low power radio communication
  - (868MHz/Europe, 915MHz/US)
- Developed since 2008 (Cycleo)
- Now run by LoRa Alliance
- LoRa WAN -> Specific protocol designed upon LoRa

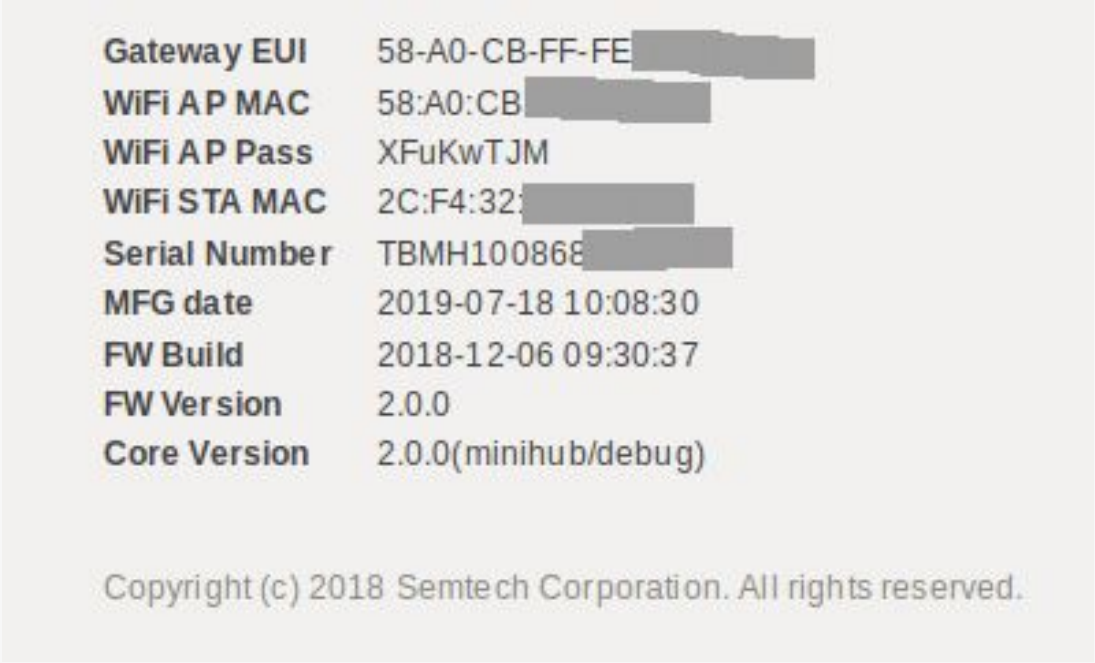
Publications on LoRa

- 2016, Syscan360: Robert Miller, MWR - LoRa Security: Building a secure LoRa solution
- 2016, GRCon16: Matt Knight, Bastille Research - Reversing and Implementing the LoRA PHY with SDR

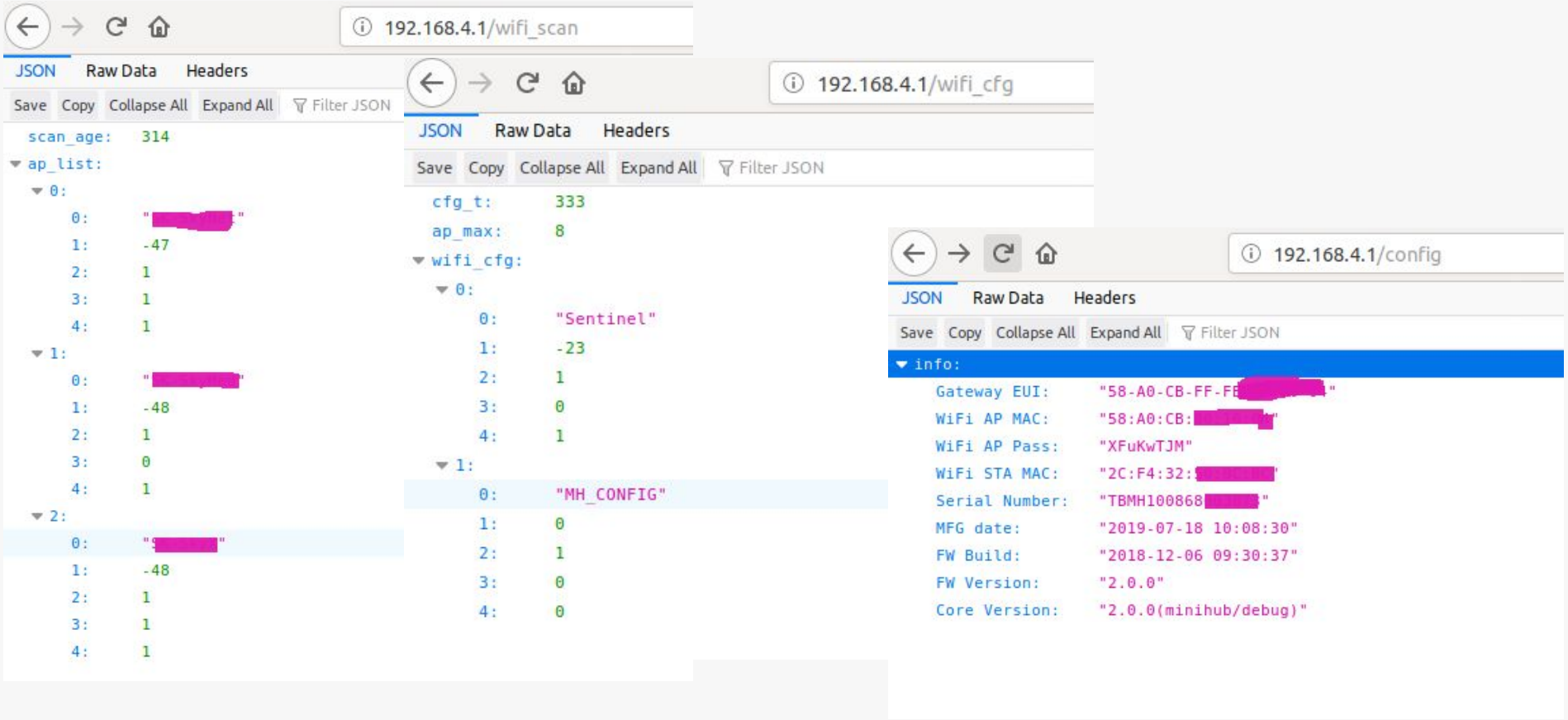
First Steps

- Runs from mains or USB-C
  - Sadly USB is dead
- Hold setup button to start WiFi AP mode
  - Config mode
- Connect to network
  - Network key is printed on back of device
  - Seems random (at least my two samples)
- http://192.168.101.4

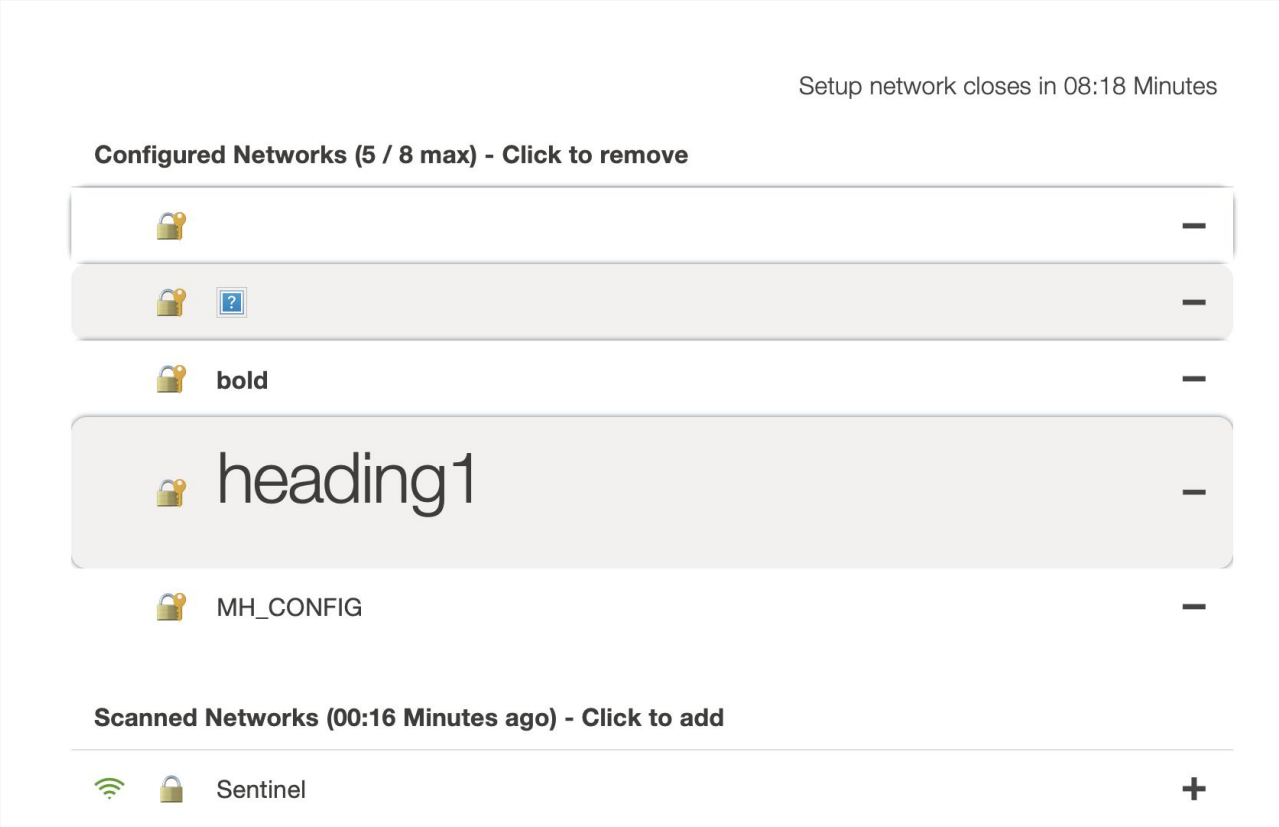




Config Menu - Version Information



Config Endpoints



Just a slight lack of input validation

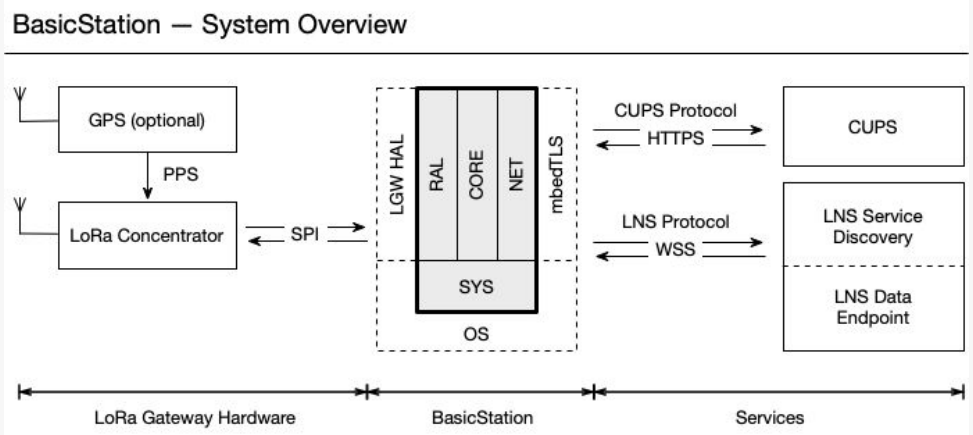
Portscan

- Only open port seems to be 80/TCP

- A ESP8266 does not like to be scanned!

LoRa Basic Station

- New approach for managing LoRa gateways and getting traffic from the field to the cloud
- Developed by Semtech
- Consists of 2 protocols
  - LNS
  - CUPS



https://lora-developers.semtech.com/resources/tools/basic-station/welcome-basic-station/

CUPS

- Configuration and Update Server
- Simple JSON based protocol
- Used to fetch configuration
  - Communication endpoints
  - Credentials
  - Certificates
  - Updates
- Information fetched based on
  - Router ID / model

Response

field	description
cupsUriLen	Length of CUPS URI (cun)
cupsUri	CUPS URI (cups.uri)
tcUriLen	Length of LNS URI (tun)
tcUri	LNS URI (tc.uri)
cupsCredLen	Length of CUPS credentials (ccn)
cupsCred	Credentials blob
tcCredLen	Length of LNS credentials (tcn)
tcCred	Credentials blob
sigLen	Length of signature for update blob
keyCRC	CRC of the key used for the signature
sig	Signature over the update blob
updLen	Length of generic update data (udn)
updData	Generic update data blob

CUPS Security

- 4 Options
- No authentication!
  - “All three files *.trust, *.cert, and *.key SHALL be missing or empty.”
- TLS Server Authentication
  - Server key stored in local .trust file
- TLS Server and Client Authentication
  - Using local .trust and client cert in .key file
- TLS Server Authentication and Client Token
  - Using .trust file and an Authorization header in .key file

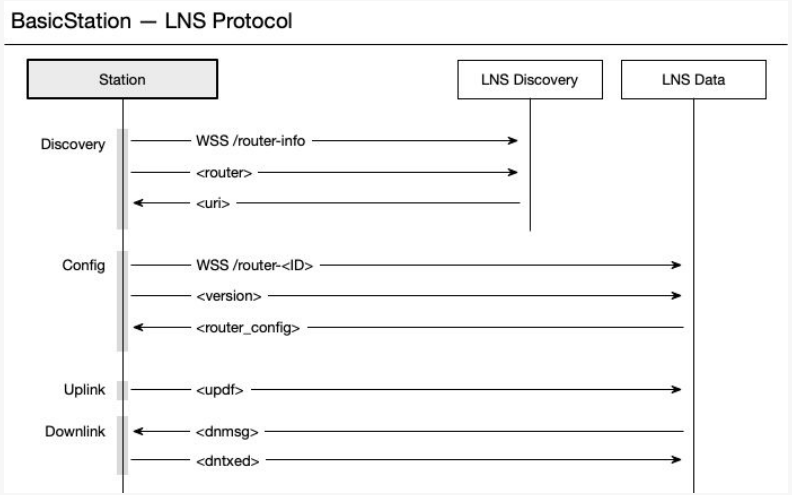
A Quote

“Station supports four different authentication modes. Each authentication mode is configured by providing specific files with credentials being defined by three types of files...”

https://lora-developers.semtech.com/resources/tools/basic-station/authentication-modes/

LNS Protocol

- LoRaWAN® Network Server Protocol
- Endpoint set by default or fetched via CUPS
- Same security measures as CUPS
- Used for
  - Radio Configuration
  - Transportation of payload data
  - Remote Shell
  - Time Synchronisation



https://lora-developers.semtech.com/resources/tools/basic-station/the-lns-protocol/

Router Config Message

```
{
  "msgtype"    : "router_config"
  "NetID"      : [ INT, .. ]
  "JoinEui"    : [ [INT,INT], .. ]  // ranges: beg,end inclusive
  "region"     : STRING             // e.g. "EU863", "US902", ..
  "hwspec"     : STRING
  "freq_range" : [ INT, INT ]       // min, max (hz)
  "DRs"        : [ [INT,INT,INT], .. ]   // sf,bw,dnonly
  "sx1301_conf": [ SX1301CONF, .. ]
  "nocca"      : BOOL
  "nodc"       : BOOL
  "nodwell"    : BOOL
}
```

SX1301CONF Object

```
{
  "radio_0": { .. } // same structure as radio_1
  "radio_1": {
    "enable": BOOL,
    "freq"  : INT
  },
  "chan_FSK": {
    "enable": BOOL
  },
  "chan_Lora_std": {
    "enable": BOOL,
    "radio": 0|1,
    "if": INT,
    "bandwidth": INT,
    "spread_factor": INT
  },
  "chan_multiSF_0": { .. }  // _0 .. _7 all have the same structure
  ..
  "chan_multiSF_7": {
    "enable": BOOL,
    "radio": 0|1,
    "if": INT
  }
}
```

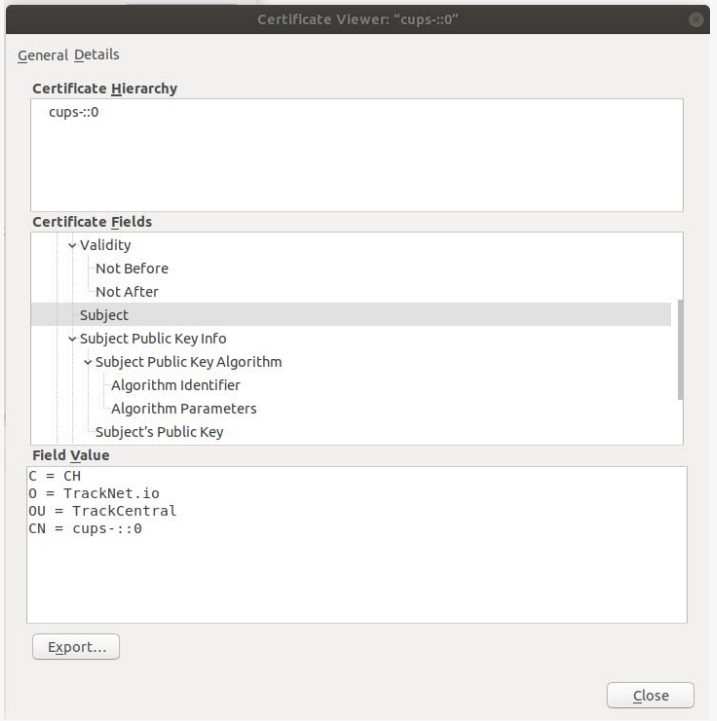
LNS Snippets

PCAP :)

MitM?

- CUPS cert is self signed
- LNS cert is a Let’s Encrypt Cert

- Created certificates with same settings
  - Redirected traffic
-



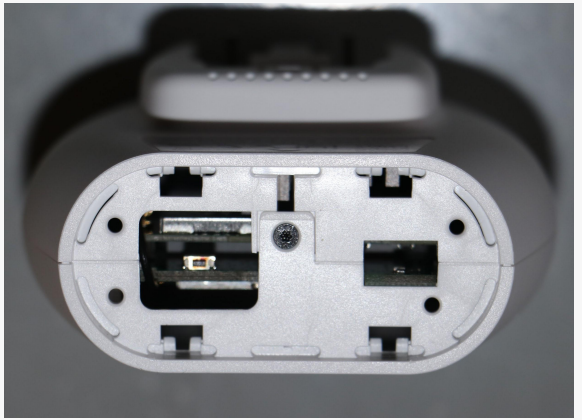
1970-01-01 00:00:08.389 [CUP:VERB] Retrieving update-info from CUPS https://rjs.sm.tc:9191...
1970-01-01 00:00:08.455 [AIO:DEBU] ssl_tls.c:4426 MBEDTLS[1]: x509_verify_cert() returned -9984 (-0x2700)
1970-01-01 00:00:08.460 [AIO:DEBU] ssl_tls.c:6849 MBEDTLS[1]: mbedtls_ssl_handshake() returned -9984 (-0x2700)
1970-01-01 00:00:08.465 [AIO:ERRO] [2] Send failed: X509 - Certificate verification failed, e.g. CRL, CA or signature check failed
1970-01-01 00:00:08.476 [AIO:DEBU] [2] HTTP connection shutdown...
1970-01-01 00:00:08.486 [SYS:INFO] sys_inState - Ignoring state transition: 5
1970-01-01 00:00:08.488 [CUP:INFO] Interaction with CUPS failed - retrying in 1m

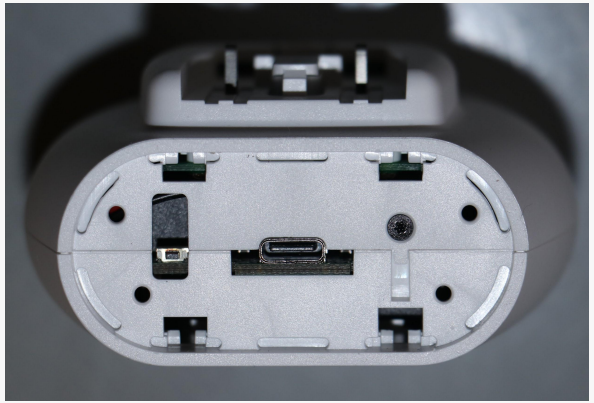
MitM Failed :(

Portscan

- No open ports to be found

- T ESP8266 still does not like to be scanned!



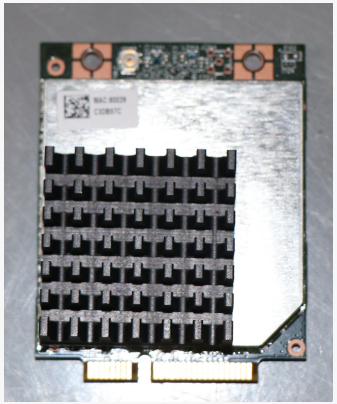


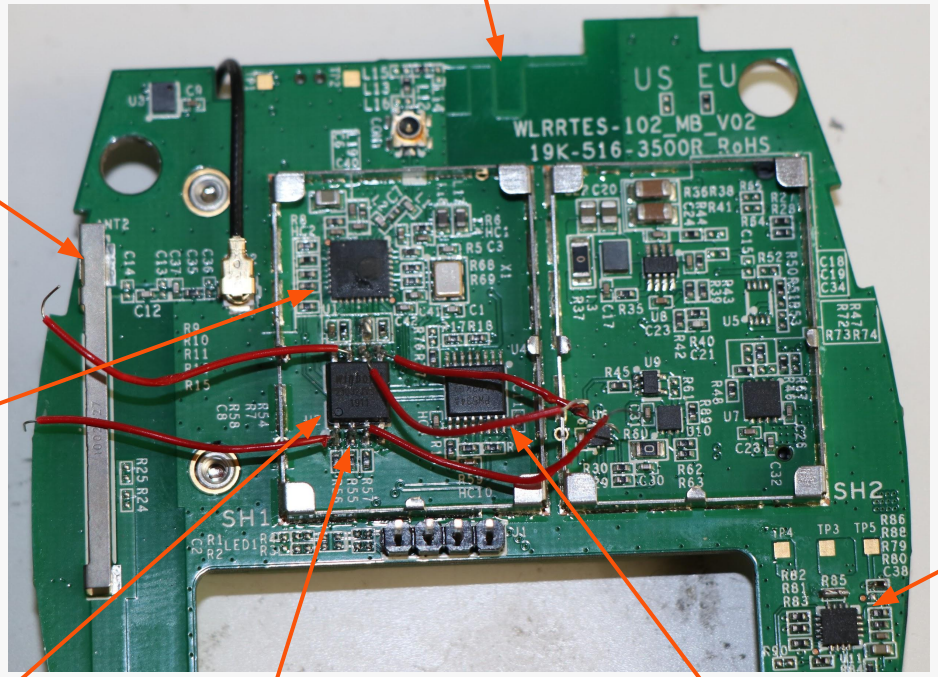
Teardown

The Open Device

Device Overview

- Based on an ESP8266
  - Own circuit, not a module
  - 4MB SPI memory, Winbond 25Q32
  - Hidden under removable shields
- UART Header
- LoRa module in mSATA format
  - Semtech SX1308




WiFi Antenna
LoRa Antenna
ESP8266
Winbond 25Q32
UART
I2C Port Expander
CP2101n USB-UART
US EU
WLRRTES-102_MB_V02
19K-516-3500R_RoHS
SH1
SH2

UART

- Prints boot log, various status information
- RX is sadly down :-(

→ Looking at a UART sniffs in slides isn't fun :)

```
1970-01-01 00:00:00.006 [SYS:DEBU] ======= VER
======
1970-01-01 00:00:00.008 [SYS:DEBU] Station Version
2.0.0(minihub/debug)
1970-01-01 00:00:00.010 [SYS:DEBU] Version Commit
e17c5af
1970-01-01 00:00:00.014 [SYS:DEBU] Station Build
2018-12-06 09:30:37
1970-01-01 00:00:00.020 [SYS:DEBU] Firmware Version
2.0.0
1970-01-01 00:00:00.025 [SYS:DEBU] FW Flavor ID
semtech0
1970-01-01 00:00:00.031 [SYS:DEBU] Model
minihub
1970-01-01 00:00:00.039 [SYS:DEBU] ======= SYS ======
1970-01-01 00:00:00.041 [SYS:DEBU] CPU Freq         80 /
80000000 / 80000000
1970-01-01 00:00:00.048 [SYS:DEBU] Random Number
896671054
1970-01-01 00:00:00.053 [SYS:DEBU] Reset cause      0
1970-01-01 00:00:00.058 [SYS:DEBU] Booting USER_BIN  1
1970-01-01 00:00:00.063 [SYS:DEBU] FW start addr
0x00001000
1970-01-01 00:00:00.069 [SYS:DEBU] SDK version
2.0-dev(9ec59b5)
1970-01-01 00:00:00.075 [SYS:DEBU] Free Heap Startup
56160 bytes
```

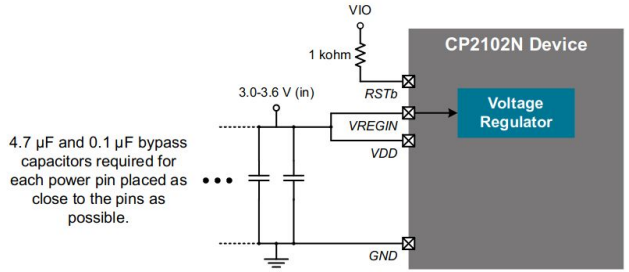


Figure 2.3. Connection Diagram with Voltage Regulator Not Used

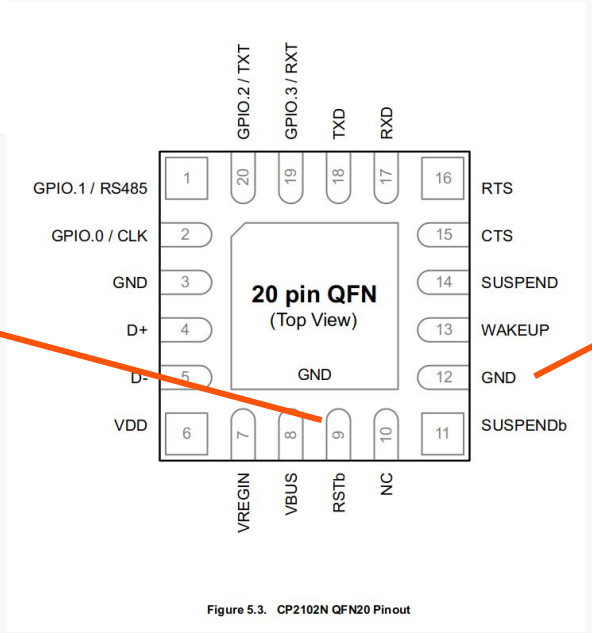
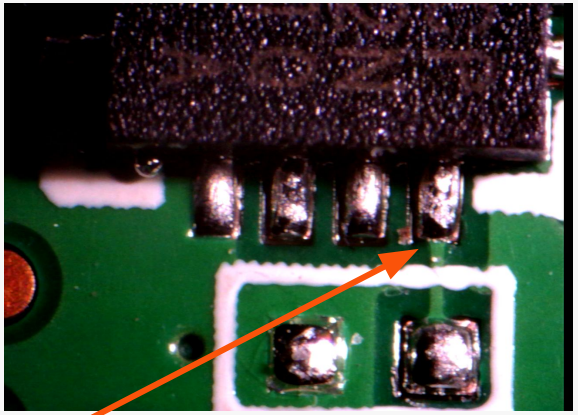
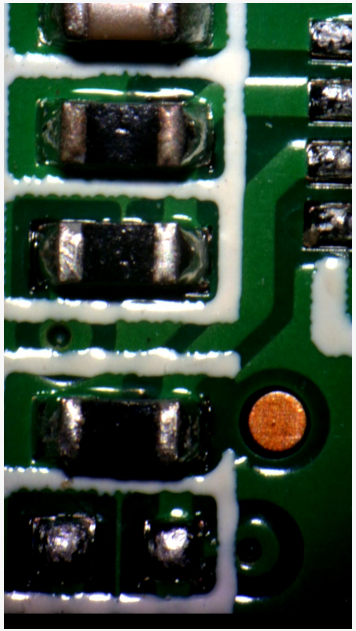


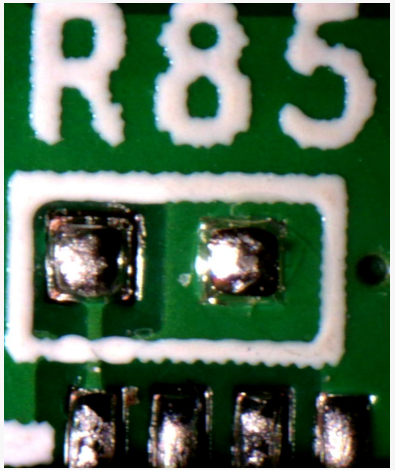
Figure 5.3. CP2102N QFN20 Pinout





GND

VCC



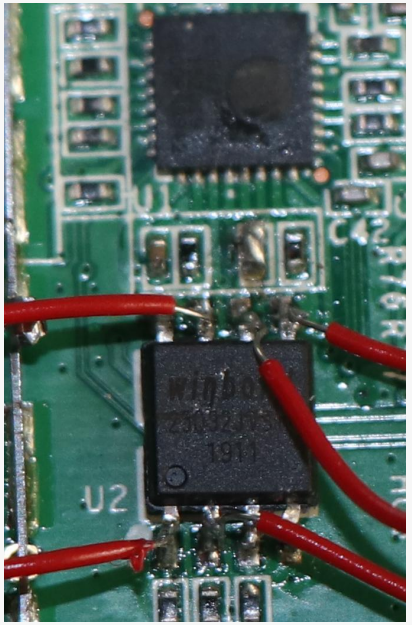
A few “mistakes” on the USB circuit

Flash Memory

- Typical approach: Saleae & SniffROM
  - Connect SOIC clip
  - Sniff communication with Saleae LA
  - Use SniffROM to reconstruct memory content
  - → No soldering
- Failed! :-(
  - Signals on LA looked good
  - Device didn’t boot anymore
    - Status LED only glimed
- Hooked up the Scope
  - Failed again

Flash Memory

- EPS8266 has 200R resistors on the SPI lines
- Removed the resistor, add a piece of wire



- Works! :)

18	SDIO_DATA_2	I/O	Connect to SD_D2 (Series R: 200Ω); SPIHD; HSPIHD; GPIO9
19	SDIO_DATA_3	I/O	Connect to SD_D3 (Series R: 200Ω); SPIWP; HSPIWP; GPIO10
20	SDIO_CMD	I/O	Connect to SD_CMD (Series R: 200Ω); SPI_CS0; GPIO11
21	SDIO_CLK	I/O	Connect to SD_CLK (Series R: 200Ω); SPI_CLK; GPIO6
22	SDIO_DATA_0	I/O	Connect to SD_D0 (Series R: 200Ω); SPI_MISO; GPIO7
23	SDIO_DATA_1	I/O	Connect to SD_D1 (Series R: 200Ω); SPI_MOSI; GPIO8

Flash Memory

- ESP8266 has no internal memory
  - No secure storage, all assets are on the flash
  - ESP32 in contrast has internal memory for keys
    - Have a look at the MINiBREW Craft System in the HardPwn Corner
- Flash is run in QuadSPI
  - But...my decoder went on strike :-(

Firmware

- Sadly firmware isn’t Open Source
  - But there is a reference / test implementation
  - https://github.com/lorabasics/basicstation
- Available code contains
  - C code for Basic Station
  - (partially Python) Code for eval / test environment
- Perfect for future test benches

Basic Station is a LoRaWAN Gateway implementation, including features like

- Ready for LoRaWAN Classes A, B, and C
- Unified Radio Abstraction Layer supporting Concentrator Reference Designs v1.5 and v2
- Powerful Backend Protocols (read here and here)
  - Centralized update and configuration management
  - Centralized channel-plan management
  - Centralized time synchronization and transfer
  - Various authentication schemes (client certificate, auth tokens)
  - Remote interactive shell
- Lean Design
  - No external software dependencies (except mbedTLS and libloragw/-v2)
  - Portable C code, no C++, dependent only on GNU libc
  - Easily portable to Linux-based gateways and embedded systems
  - No dependency on local time keeping
  - No need for incoming connections

Lame Code Analysis

- Detailed code analysis takes a long time
  - But there always is a compromise one can take
- Quickly grepping for bad/risky functions
  - Or using an applicable tool helps
- Flawfinder
  - Simple python code analysis tool

Flawfinder Output

```
./fs.c:282:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to
destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy
(warning: strncpy
  easily misused).
        strcpy(wb, cwd);
./fs.c:645:  [4] (race) access:
  This usually indicates a security flaw. If an attacker can
change anything
  along the path between the call to access() and the file's
actual use
  (e.g., by moving files), the attacker can exploit the race
condition
  (CWE-362/CWE-367!). Set up the correct permissions (e.g.,
using setuid())
  and try to open the file directly.
        return access(fn, mode);
```

Flawfinder Results

- Found 100 issues
  - Memcpy, strcpy, statically sized arrays, issues with not \0 terminated String
- Obviously “issues” are just potentials
  - I.e. strcpy issues can be prevented by proper validation of data
- Manually checked quite a few of them
  - All looked fine

Flawfinder Output

```
./fs.c:282:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to
destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy
(warning: strncpy
  easily misused).
      strcpy(wb, cwd);
./fs.c:645:  [4] (race) access:
  This usually indicates a security flaw. If an attacker can
change anything
  along the path between the call to access() and the file's
actual use
  (e.g., by moving files), the attacker can exploit the race
condition
  (CWE-362/CWE-367!). Set up the correct permissions (e.g.,
using setuid())
  and try to open the file directly.
      return access(fn, mode);
```

Requirements

- No default passwords
  - Check, except for the WiFi password which should be acceptable
- Implement a vulnerability disclosure policy
  - ?
- Keep software updated
  - At least they can
- Securely store credentials and security-sensitive data
  - Well….probably not
- Communicate securely
  - Check
- Minimise exposed attack surfaces
  - Check
- Ensure software integrity
  - Done during the update
- Ensure that personal data is protected
  - Hum, hard to say
- Make systems resilient to outages
  - Out of scope
- Monitor system telemetry data
  - Backend, so didn’t test
- Make it easy for consumers to delete personal data
  - Reset button
- Make installation and maintenance of devices easy
  - Yep
- Validate input data
  - Not yet perfect

Summary

- TTIG - 868 is a typical, simple IoT Device
  - No notable physical protection measures
- Configuration WiFi is done nicely
  - Unique key
  - Shutdown after 15 minutes
  - Output of SSID needs to be cleaned though
- Basic Station Protocol implements all necessary security options
  - Support for TLS
  - TLS actually works :)
  - No authentication without TLS
- Parsing might cause issues on other implementations

It’s not insecure!

A single day?

- Admittingly I spread a days work over multiple days
- But all in all I only took me about 10h

- Quick tests are easily possible, when you have your bits together

Outlook

- I need to fix and publish my QuadSPI decoder
  - Change the CUPS Server on the TTIG
- ...give a bunch of other IoT devices a single day pentest

Thanks for your time

Questions?

brian@security-bits.de ~~ www.security-bits.de ~~ @BadgeWizard