Hacking Into Someone’s Home using Radio Waves

Ethical Hacking of Securitas’ Alarm System

Axel Lindeberg, master thesis presentation

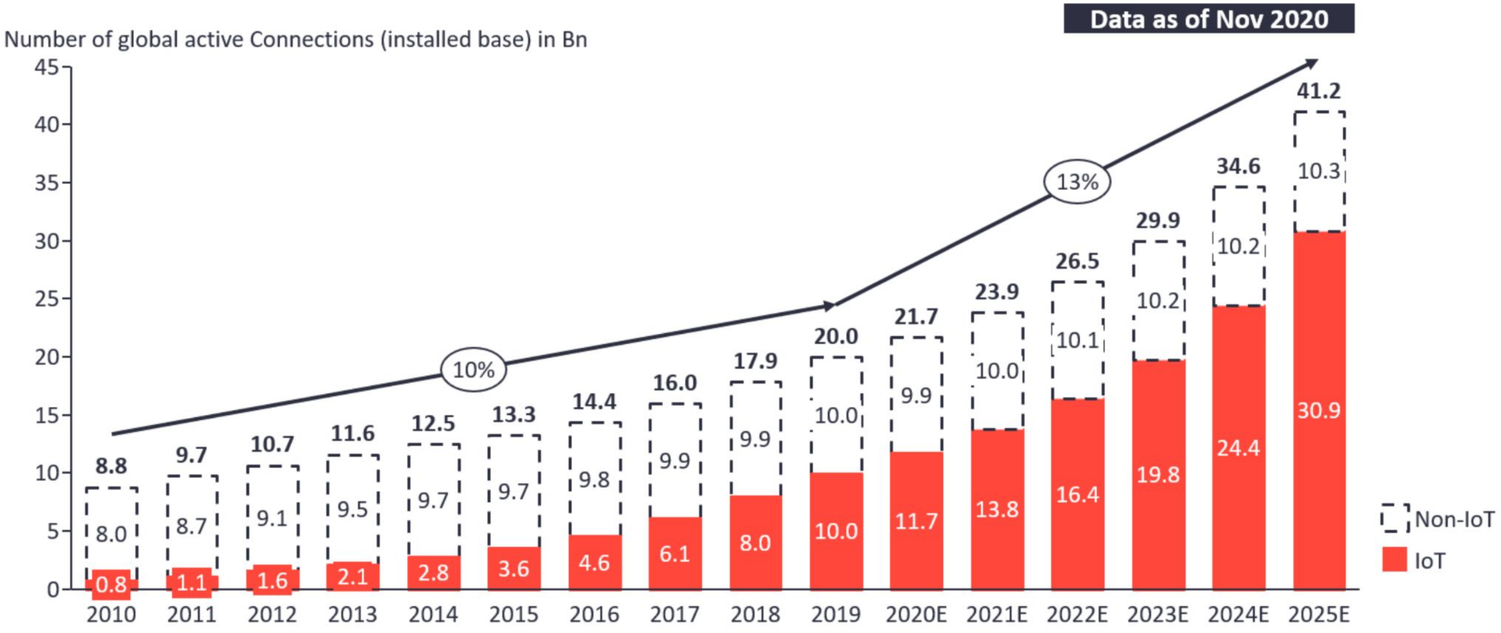
The Agenda for Today

1. Introduction
2. Method
3. The System under Consideration
4. Threat model
5. Penetration tests
   - 5.1. RF Replay attack
   - 5.2. RF Reverse Engineering
   - 5.3. RF Jamming Attack
   - 5.4. Insecure Network Services
6. Conclusions

The Rise of Connected IoT Devices

An estimated 30 billion connected IoT devices by 2025

Set to outnumber non-IoT devices to by 3:1 in 2025



The Rise of Smart Home Alarm Systems

Have become increasingly complex:

- Controlled remotely (mobile, web)
- Home Automation
- Smart speakers (Alexa, Google home)
- Smart locks

20% expected market growth in 2021.

Verisure alone is installed in 360 000 Swedish homes.

How secure are they against cyber attacks?

Method

Penetration testing methodology

Seven-step penetration testing methodology (Weidman, 2014)

1. Pre-engagement
2. Information Gathering
3. Threat Modeling
4. Vulnerability Analysis
5. Exploitation
6. Post Exploitation
7. Reporting

Threat modeling technique

A process to identify and analyze threats against a system.

A six-step process, specialized for IoT systems presented by Guzman et al. (2017).

1. Identify IoT assets
2. Create an architecture overview
3. Decompose the IoT system
4. Identify threats
5. Document threats
6. Rate the threats

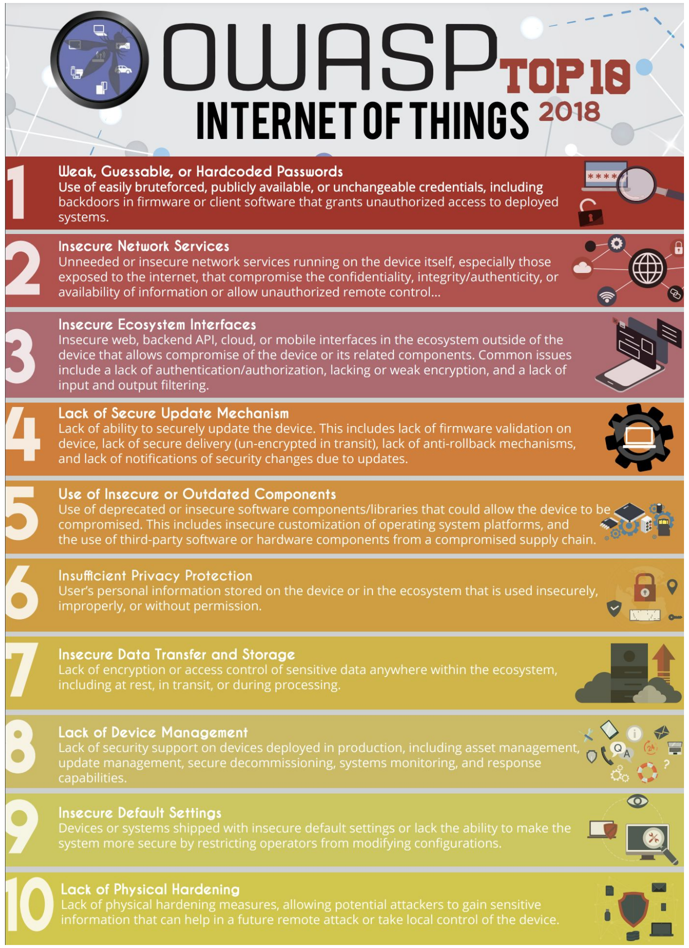
Related Work in Identifying Threats

- The STRIDE model of threats
- OWASP IoT Top 10
- ETSI EN 303 645 standard

STRIDE
METHODOLOGY
01
SPOOFING
02
TAMPERING
03
REPUDIATION
04
INFORMATION DISCLOSURE
05
DENIAL OF SERVICE (DOS)
06
ELEVATION OF PRIVILEGE

ETSI

The Standards People



OWASP TOP 10

INTERNET OF THINGS 2018

1 **Weak, Guessable, or Hardcoded Passwords**
Use of easily bruteforced, publicly available, or unchangeable credentials, including backdoors in firmware or client software that grants unauthorized access to deployed systems.

2 **Insecure Network Services**
Unneeded or insecure network services running on the device itself, especially those exposed to the internet, that compromise the confidentiality, integrity/authenticity, or availability of information or allow unauthorized remote control…

3 **Insecure Ecosystem Interfaces**
Insecure web, backend API, cloud, or mobile interfaces in the ecosystem outside of the device that allows compromise of the device or its related components. Common issues include a lack of authentication/authorization, lacking or weak encryption, and a lack of input and output filtering.

4 **Lack of Secure Update Mechanism**
Lack of ability to securely update the device. This includes lack of firmware validation on device, lack of secure delivery (un-encrypted in transit), lack of anti-rollback mechanisms, and lack of notifications of security changes due to updates.

5 **Use of Insecure or Outdated Components**
Use of deprecated or insecure software components/libraries that could allow the device to be compromised. This includes insecure customization of operating system platforms, and the use of third-party software or hardware components from a compromised supply chain.

6 **Insufficient Privacy Protection**
User's personal information stored on the device or in the ecosystem that is used insecurely, improperly, or without permission.

7 **Insecure Data Transfer and Storage**
Lack of encryption or access control of sensitive data anywhere within the ecosystem, including at rest, in transit, or during processing.

8 **Lack of Device Management**
Lack of security support on devices deployed in production, including asset management, update management, secure decommissioning, systems monitoring, and response capabilities.

9 **Insecure Default Settings**
Devices or systems shipped with insecure default settings or lack the ability to make the system more secure by restricting operators from modifying configurations.

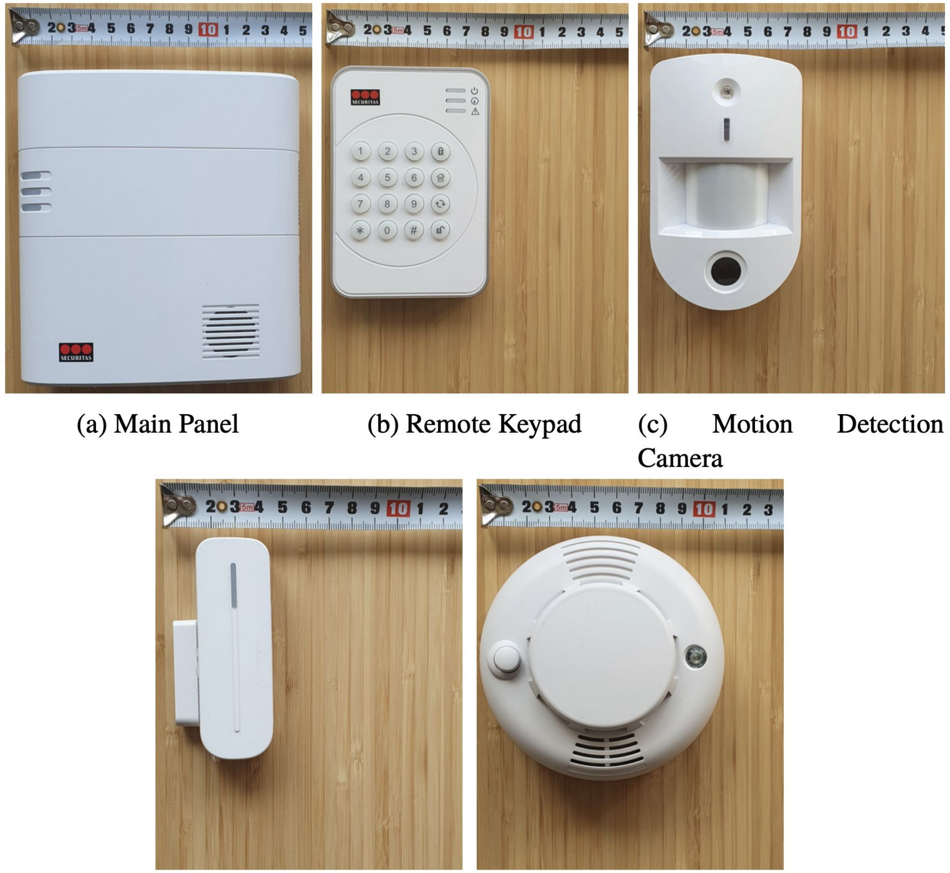
10 **Lack of Physical Hardening**
Lack of physical hardening measures, allowing potential attackers to gain sensitive information that can help in a future remote attack or take local control of the device.

The System under Consideration

The SecuritasHome Alarm System

The Hardware:
Five components, including a central panel that controls the whole system.

Manufactured by a Taiwanese company called *Climax Technology*



(a) Main Panel

(b) Remote Keypad

(c) Motion Detection Camera

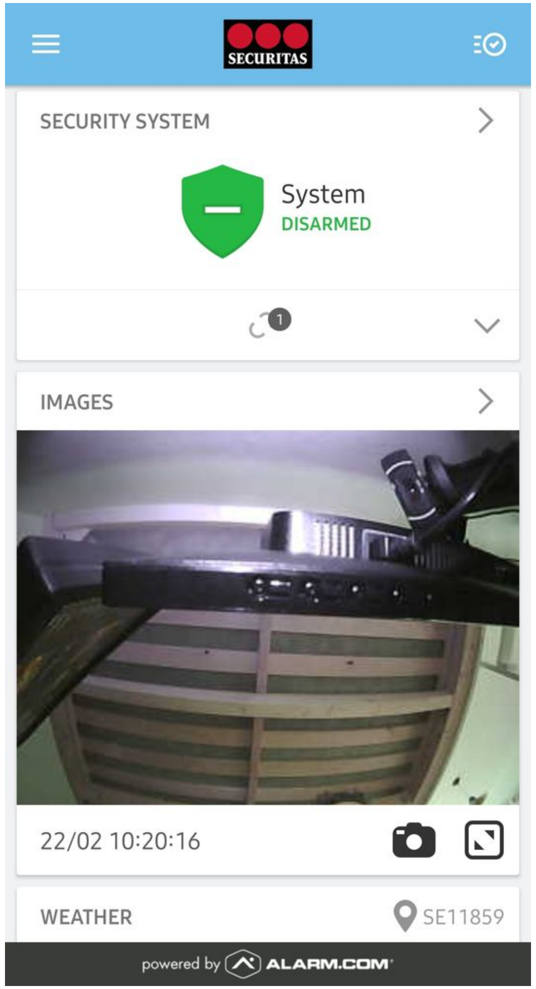
(d) Door Contact Sensor

(e) Smoke Detector

The SecuritasHome Alarm System

The Software:
A website and a mobile app. Allows the user to remotely control the system.

Developed by an American company called *Alarm.com*



Threat model

Identified Assets

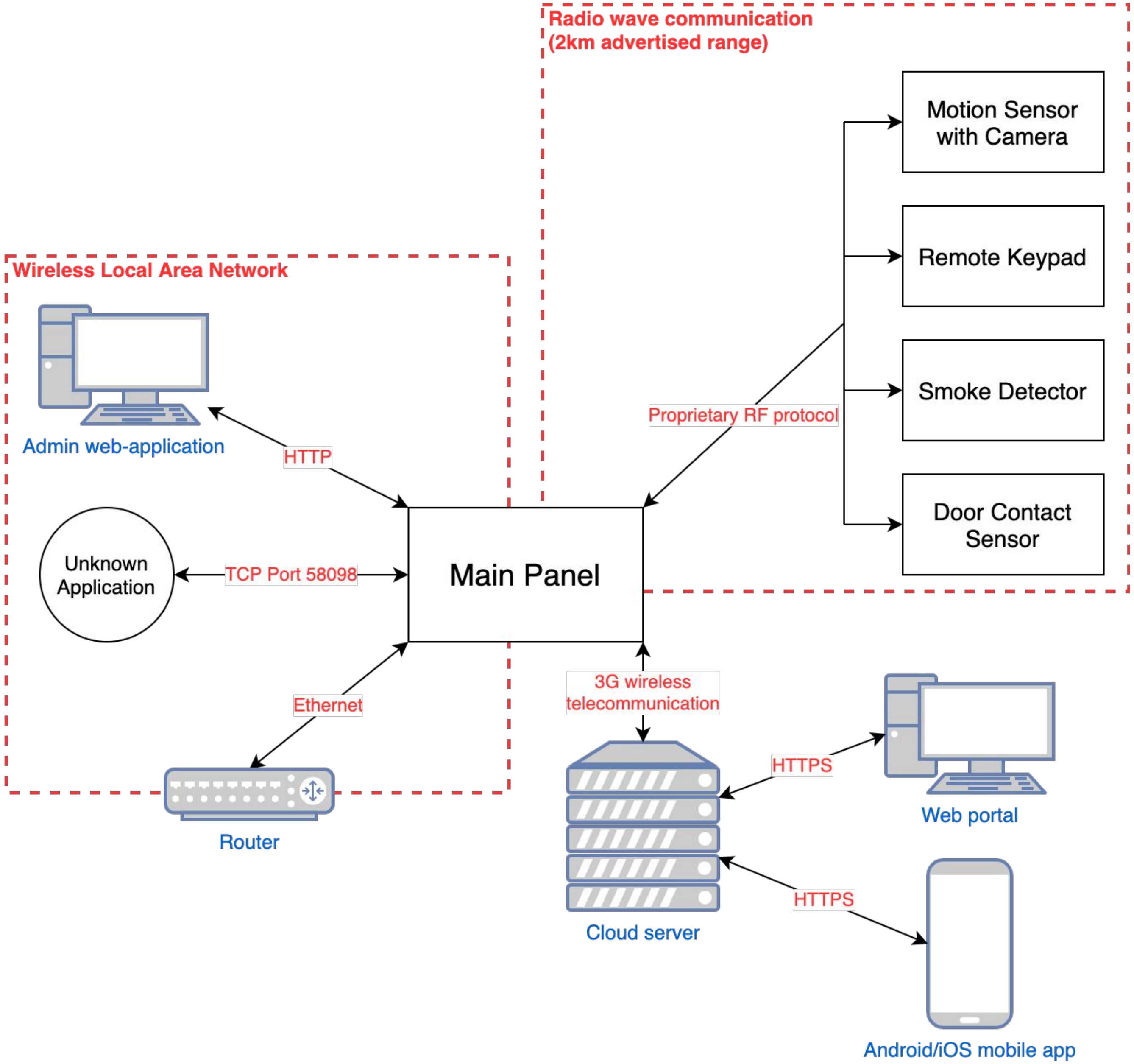
Initial step: Identify the assets of the system.

- Physical access to the house
- Personal four-digit pin
- Arm/disarm state of the system
- State of triggers, like the sabotage sensors
- Door contact sensor state
- Authentication to the admin web application
- Triggered alarm state
- Login credentials to the local webserver

Architecture Overview

Identified use-cases:

- The user arms/disarms the system via:
  - the remote keypad panel.
  - the web portal.
  - the mobile app.
- The user receives a notification about a state change in the system via:
  - a mobile notification.
  - an email.
- The user requests a photo be taken by the camera.


Radio wave communication
(2km advertised range)
Motion Sensor with Camera
Remote Keypad
Smoke Detector
Door Contact Sensor
Proprietary RF protocol
Wireless Local Area Network
Admin web-application
HTTP
Unknown Application
TCP Port 58098
Main Panel
Ethernet
Router
3G wireless telecommunication
HTTPS
Web portal
HTTPS
Cloud server
Android/iOS mobile app

Identified Threats

The STRIDE model, OWASP IoT Top 10, and ETSI EN 303 645 standard were used to identify threats against the system:

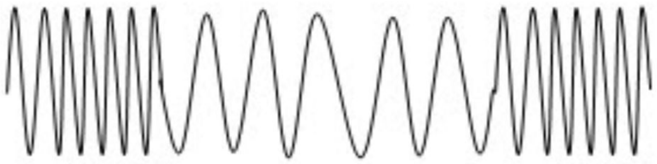
- Spoof hardware components via RF communication
- RF Replay attack
- Insecure or non-existent encryption in communication
- DoS/Jamming attack against the RF communication
- Insecure default credentials
- Insecure network services
- ... And more

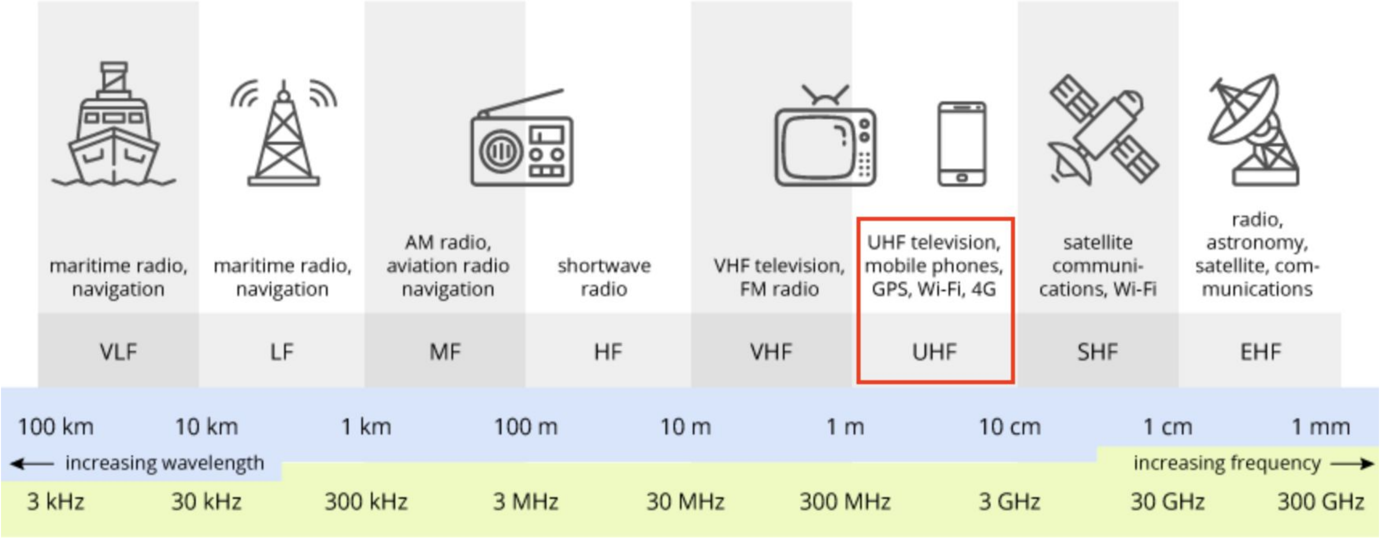
Penetration Tests

1. RF Replay attack
2. RF reverse engineering
3. RF jamming attack
4. Insecure Network services

RF Communication

- IoT Systems often use Radio Frequencies (RF) to communicate wirelessly.
- They communicate at a specific, predetermined frequency.
- *WiFi, Bluetooth, Zigbee, proprietary protocols*




maritime radio, navigation
maritime radio, navigation
AM radio, aviation radio navigation
shortwave radio
VHF television, FM radio
UHF television, mobile phones, GPS, Wi-Fi, 4G
satellite communi-cations, Wi-Fi
radio, astronomy, satellite, com-munications
VLF
LF
MF
HF
VHF
UHF
SHF
EHF
100 km
10 km
1 km
100 m
10 m
1 m
10 cm
1 cm
1 mm
← increasing wavelength
increasing frequency →
3 kHz
30 kHz
300 kHz
3 MHz
30 MHz
300 MHz
3 GHz
30 GHz
300 GHz

RF Hacking - Why?

“RF Hacking today is the same as Web Security in the 90s”

Harshit Agrawal, MIT Academy of Engineering

- Security is often overlooked/ignored by manufacturers.
- Riddled with trivial vulnerabilities, unencrypted traffic, etc.
- Lack of cybersecurity competence among RF engineers.

- Higher barrier to entry, requires specific hardware.
- Not common knowledge among pentesters like web security.

Software Defined Radios (SDR)

RF Hacking requires specific hardware, SDRs.

- Used to receive and transmit arbitrary RF signals.
- Controlled and tunable in software.
- Have become cheap and readily available in recent years.

RTL-SDR.COM
QUICKSTART SETUP GUIDE: RTL-SDR.COM/QSG
DVB-T+DAB+FM+SDR
RTL2832U R820T2 TCXO+BIAS T+HF
V.3

Penetration Tests

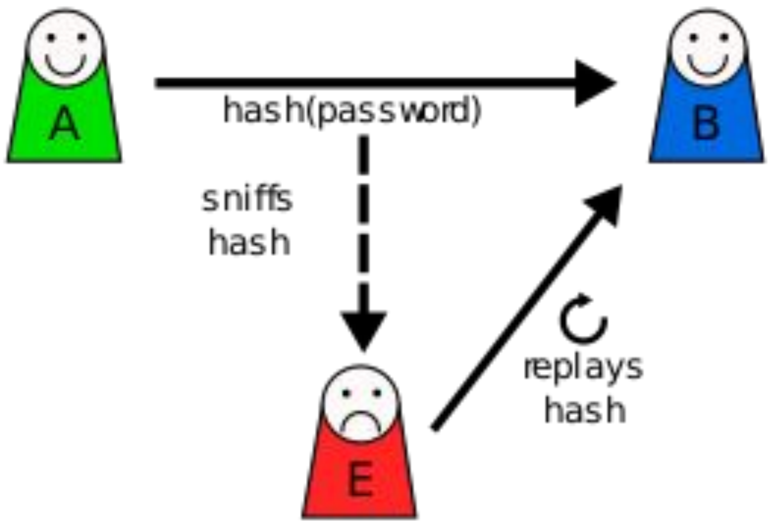
Task 1: RF Replay attack

Task 1: RF Replay attack

A general attack on network communication.

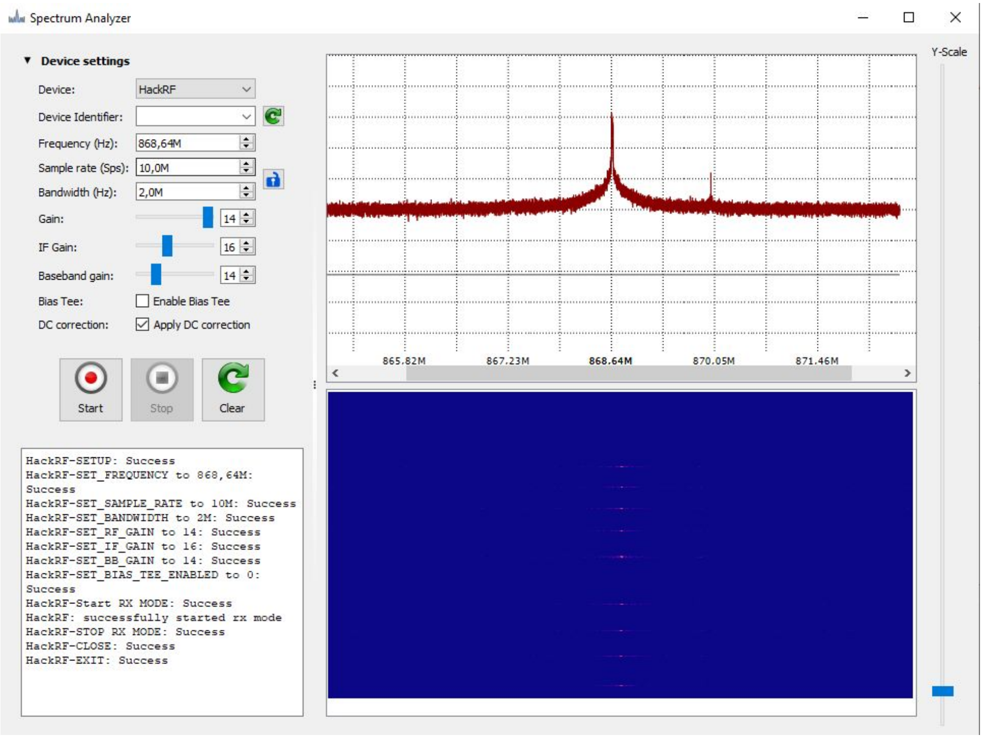
A message is recorded and then retransmitted later to achieve the same effect.

A *“zero-knowledge”* attack.



Task 1: RF Replay attack - Method

- First, establish the center frequency.
- Used the open-source tool *Universal Radio Hacker* (URH).
- Communicates at **868,64 MHz**.



▼ Device settings

Device: HackRF
Device Identifier:
Frequency (Hz): 868,64M
Sample rate (Sps): 10,0M
Bandwidth (Hz): 2,0M
Gain: 14
IF Gain: 16
Baseband gain: 14
Bias Tee: □ Enable Bias Tee
DC correction: ☑ Apply DC correction

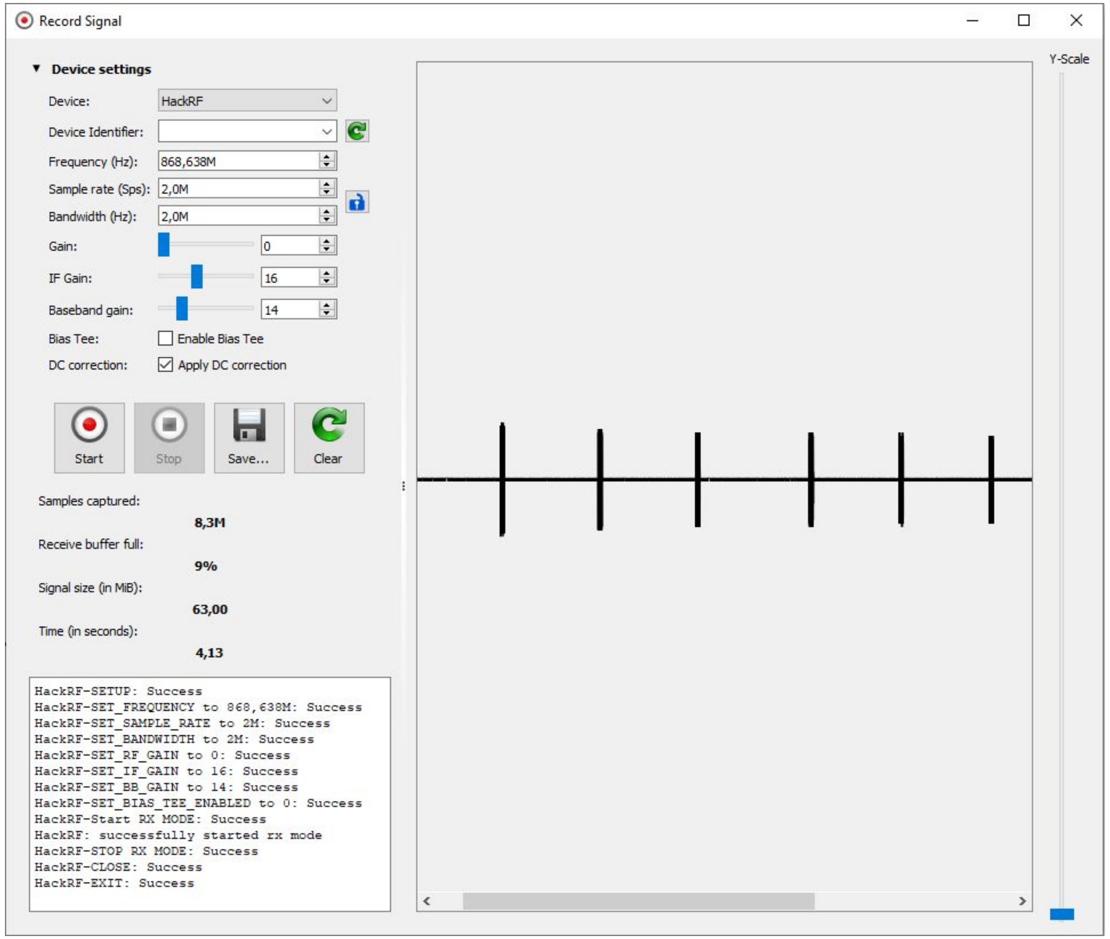
```
HackRF-SETUP: Success
HackRF-SET_FREQUENCY to 868,64M:
Success
HackRF-SET_SAMPLE_RATE to 10M: Success
HackRF-SET_BANDWIDTH to 2M: Success
HackRF-SET_RF_GAIN to 14: Success
HackRF-SET_IF_GAIN to 16: Success
HackRF-SET_BB_GAIN to 14: Success
HackRF-SET_BIAS_TEE_ENABLED to 0:
Success
HackRF-Start RX MODE: Success
HackRF: successfully started rx mode
HackRF-STOP RX MODE: Success
HackRF-CLOSE: Success
HackRF-EXIT: Success
```

Task 1: RF Replay attack - Method

Next, signals were recorded using URH.

These are saved as a file of raw IQ-data of signed bytes.

Was repeated for all identified RF endpoints.



▼ Device settings

Device: HackRF

Device Identifier:

Frequency (Hz): 868,638M

Sample rate (Sps): 2,0M

Bandwidth (Hz): 2,0M

Gain: 0

IF Gain: 16

Baseband gain: 14

Bias Tee: ☐ Enable Bias Tee

DC correction: ☑ Apply DC correction

Samples captured: 8,3M

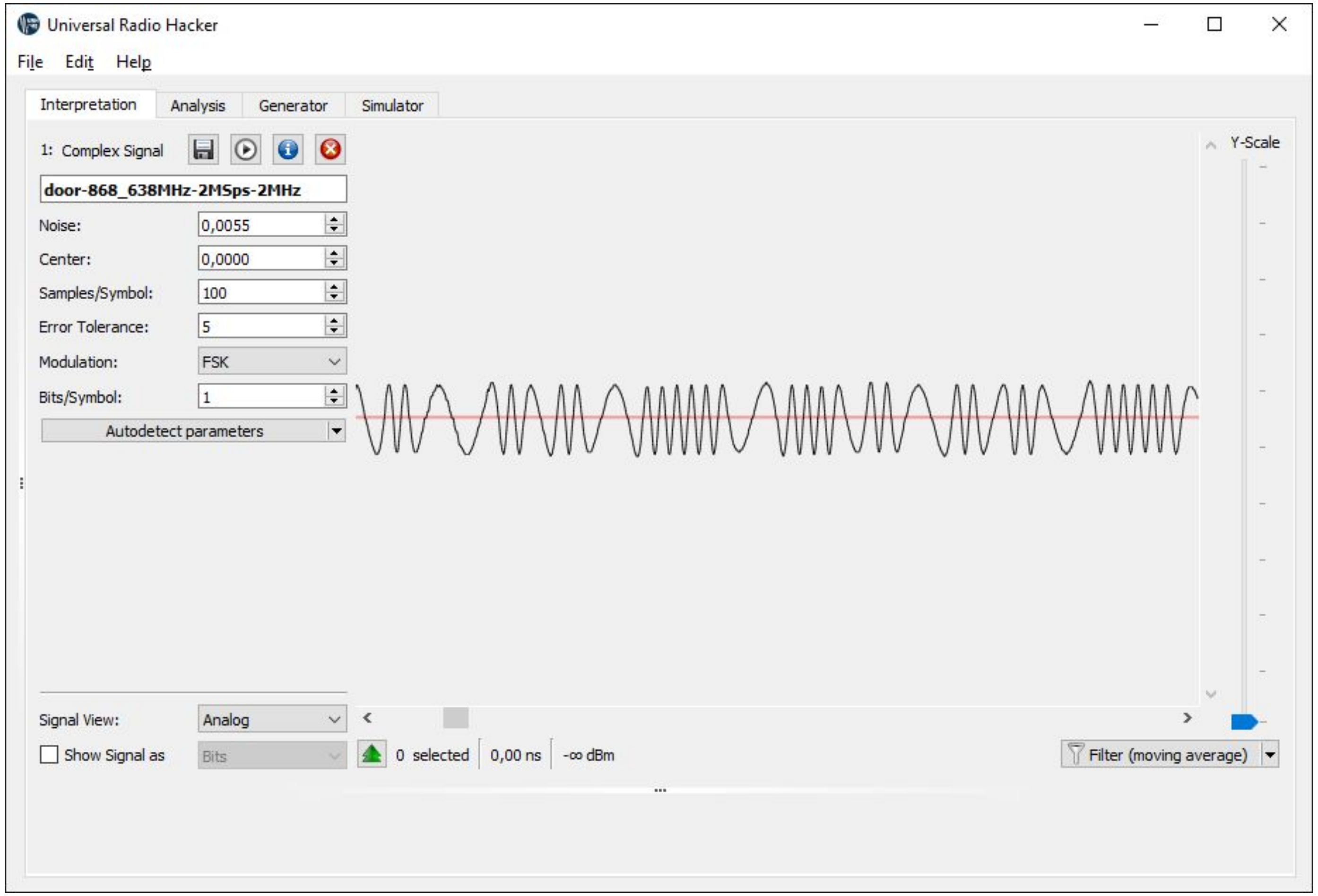
Receive buffer full: 9%

Signal size (in MiB): 63,00

Time (in seconds): 4,13

```
HackRF-SETUP: Success
HackRF-SET_FREQUENCY to 868,638M: Success
HackRF-SET_SAMPLE_RATE to 2M: Success
HackRF-SET_BANDWIDTH to 2M: Success
HackRF-SET_RF_GAIN to 0: Success
HackRF-SET_IF_GAIN to 16: Success
HackRF-SET_BB_GAIN to 14: Success
HackRF-SET_BIAS_TEE_ENABLED to 0: Success
HackRF-Start RX MODE: Success
HackRF: successfully started rx mode
HackRF-STOP RX MODE: Success
HackRF-CLOSE: Success
HackRF-EXIT: Success
```

Task 1: RF Replay attack - Method



door-868_638MHz-2MSps-2MHz

Noise: 0,0055

Center: 0,0000

Samples/Symbol: 100

Error Tolerance: 5

Modulation: FSK

Bits/Symbol: 1

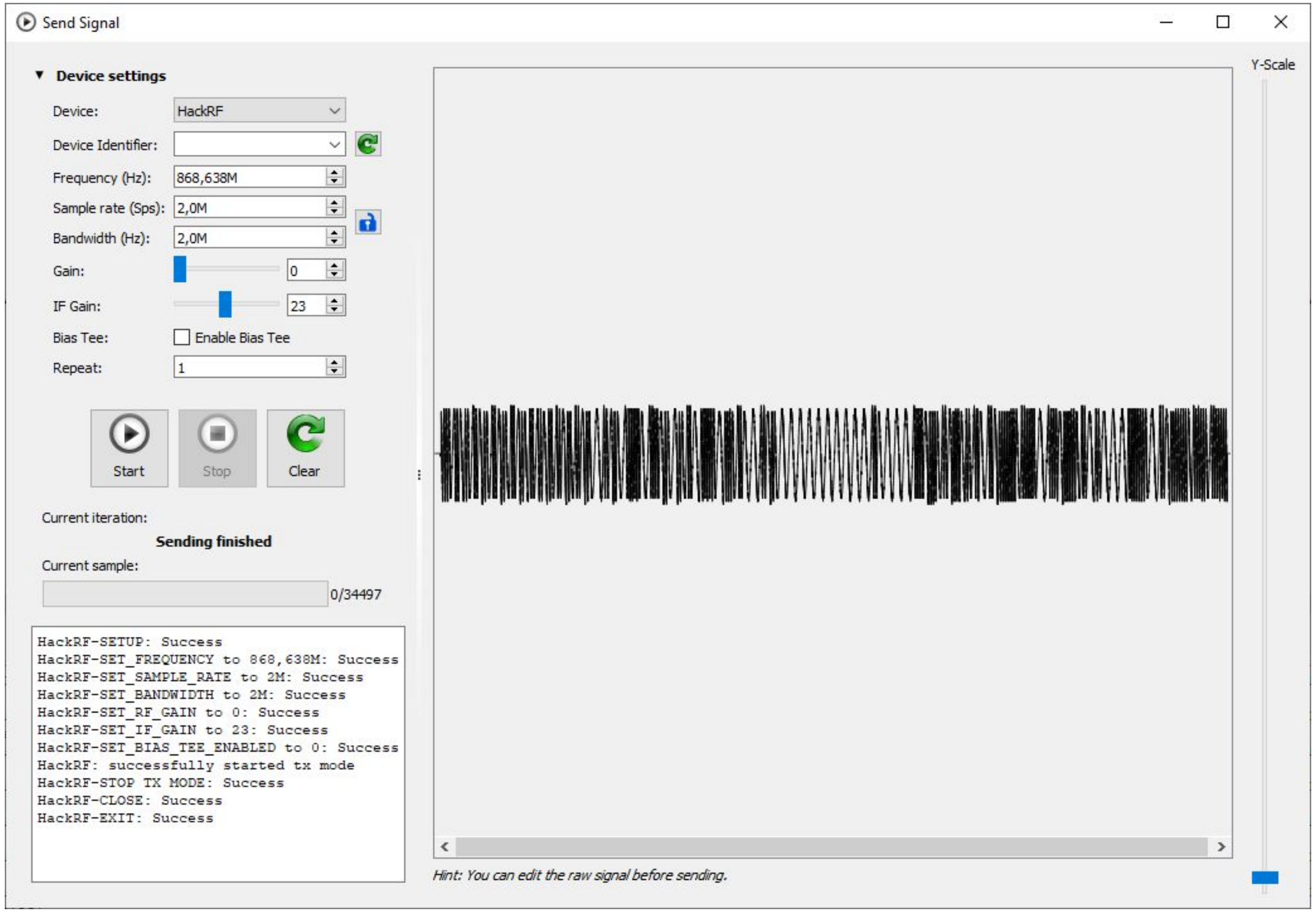
Signal View: Analog

□ Show Signal as Bits

0 selected | 0,00 ns | -∞ dBm

Task 1: RF Replay attack - Method

Lastly, the recorded signals were transmitted using URH.



▼ **Device settings**

Device: HackRF

Device Identifier:

Frequency (Hz): 868,638M

Sample rate (Sps): 2,0M

Bandwidth (Hz): 2,0M

Gain: 0

IF Gain: 23

Bias Tee: □ Enable Bias Tee

Repeat: 1

Current iteration:

Sending finished

Current sample: 0/34497

```
HackRF-SETUP: Success
HackRF-SET_FREQUENCY to 868,638M: Success
HackRF-SET_SAMPLE_RATE to 2M: Success
HackRF-SET_BANDWIDTH to 2M: Success
HackRF-SET_RF_GAIN to 0: Success
HackRF-SET_IF_GAIN to 23: Success
HackRF-SET_BIAS_TEE_ENABLED to 0: Success
HackRF: successfully started tx mode
HackRF-STOP TX MODE: Success
HackRF-CLOSE: Success
HackRF-EXIT: Success
```

Hint: You can edit the raw signal before sending.

Task 1: RF Replay attack - Results

All identified RF communication endpoints were deemed vulnerable to replay attacks.

Captured signals still worked months later.

An attacker can disarm an armed system, completely bypassing the systems functionality.

Task 1: RF Replay attack - Discussion

A critical mistake, compromising the entire security.

A well-known issue in RF communication.

Can be hard to protect against in some IoT systems.

- Timestamps
- One-time passwords
- Rolling codes

Does, however, require first recording the signal.

Penetration Tests

Task 2: Reverse engineering the RF protocol

Task 2: RF reverse engineering

The system uses a proprietary RF protocol.

Ideally, one would like to understand the RF protocol structure.

Can let an attacker send arbitrary RF messages.

Need to convert raw signal data back into binary data.

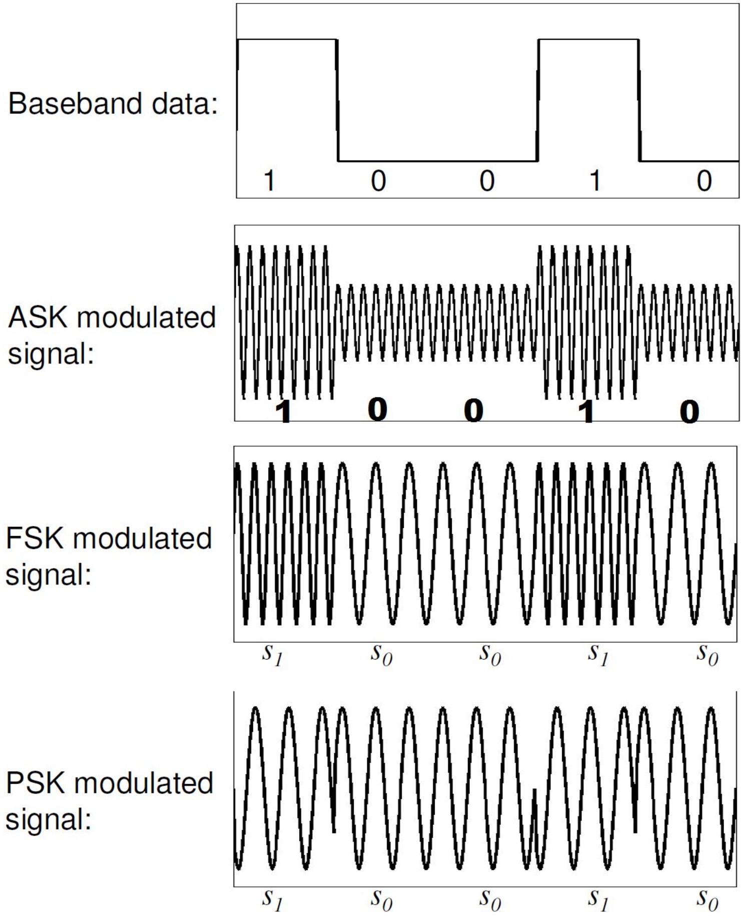
Task 2: Background - Modulation

Digital Modulation, encoding binary data in radio signals.

Three main types:

- *Amplitude-Shift keying* (ASK)
- *Frequency-Shift keying* (FSK)
- *Phase-Shift keying* (PSK)

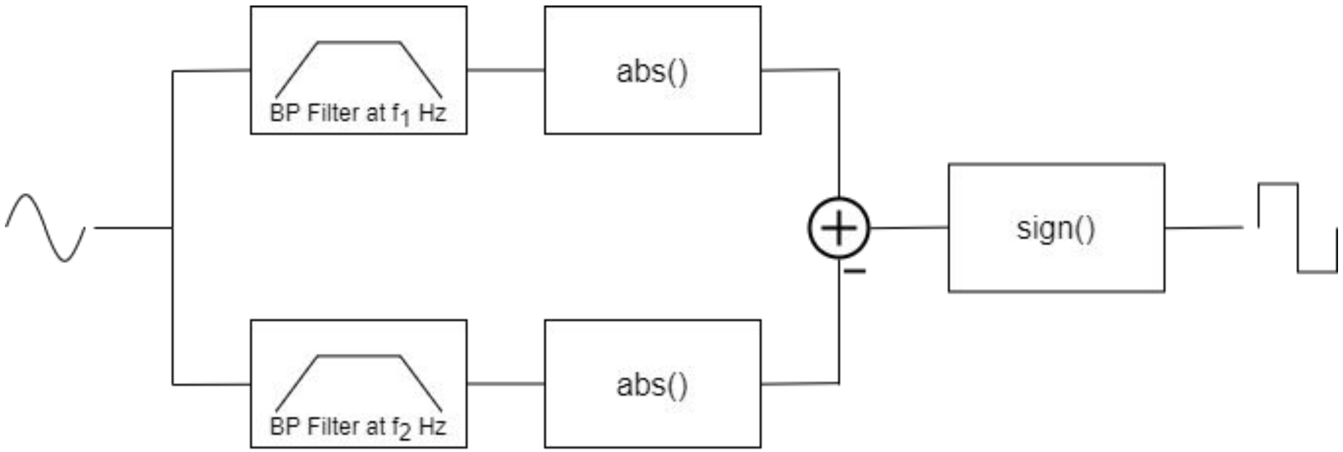
Many, more complicated schemes (*OOK, GFSK, MFSK, QAM, FHSS, ...*)



Task 2: Background - Demodulation

Digital Demodulation, is the process of converting a modulated signal back into binary data.

Usually done in hardware, using efficient specialized circuitry.

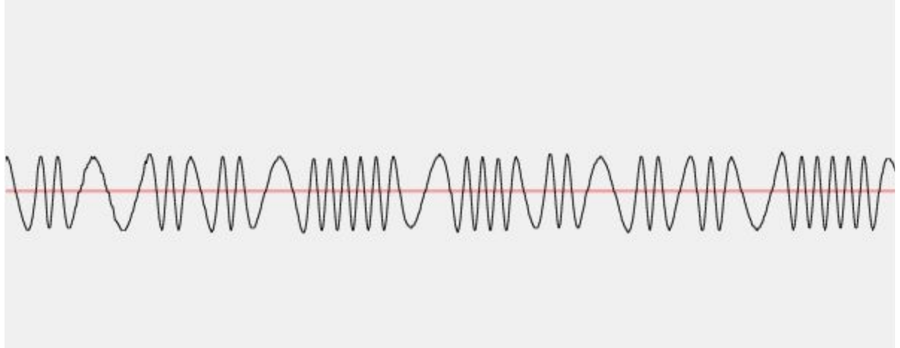


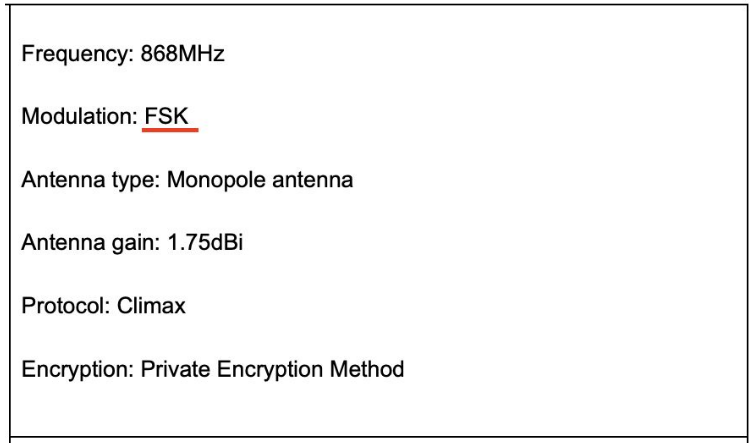
Task 2: Method

Signals were captured using the method described previously.

Next, we need to find the modulation scheme used.

Visually inspecting the signal we see it is *FSK*, also documented in the user manual.





Frequency: 868MHz

Modulation: FSK

Antenna type: Monopole antenna

Antenna gain: 1.75dBi

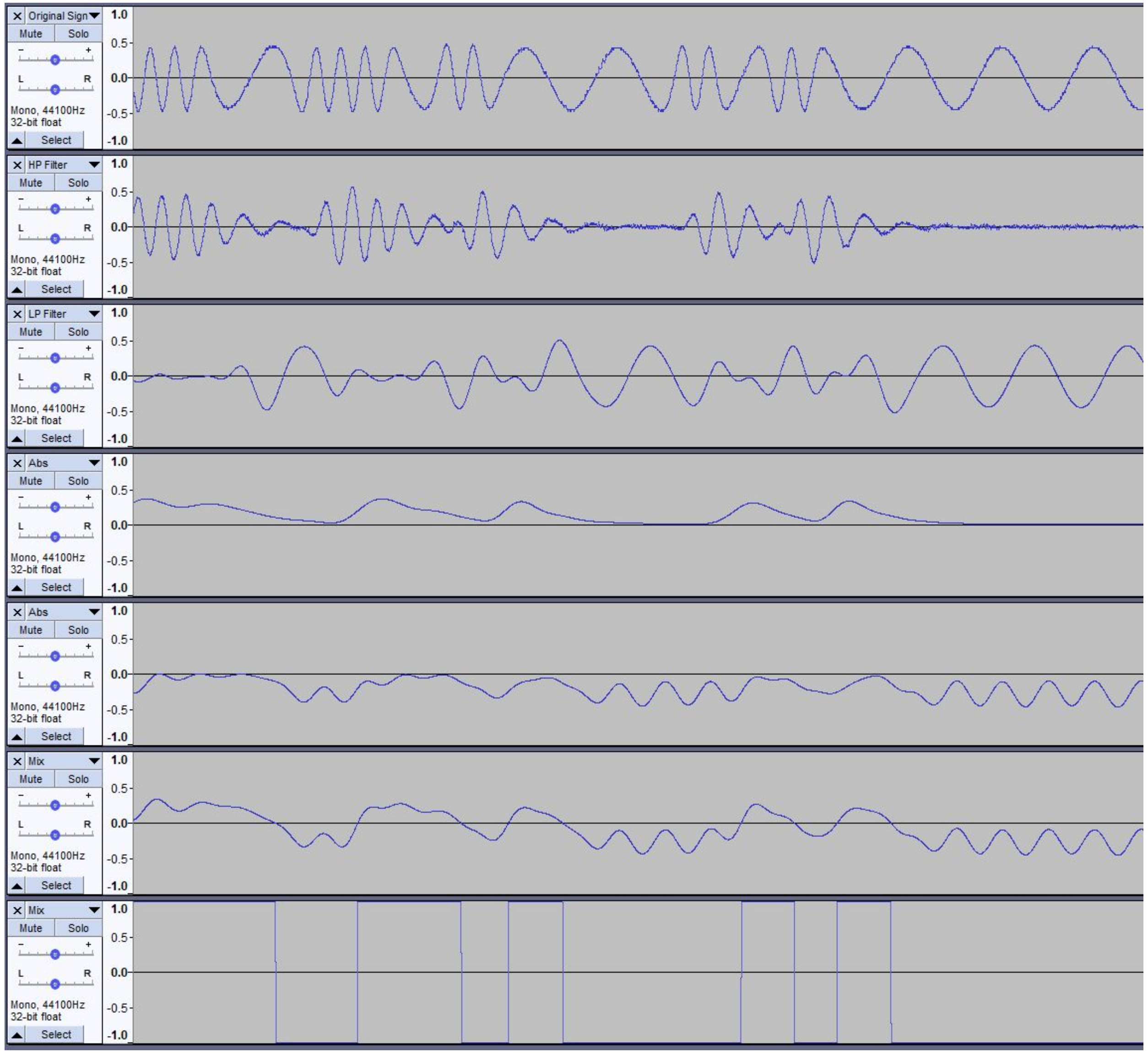
Protocol: Climax

Encryption: Private Encryption Method

Task 2: Method

URH tries to automatically demodulate, however, it failed for these signals.

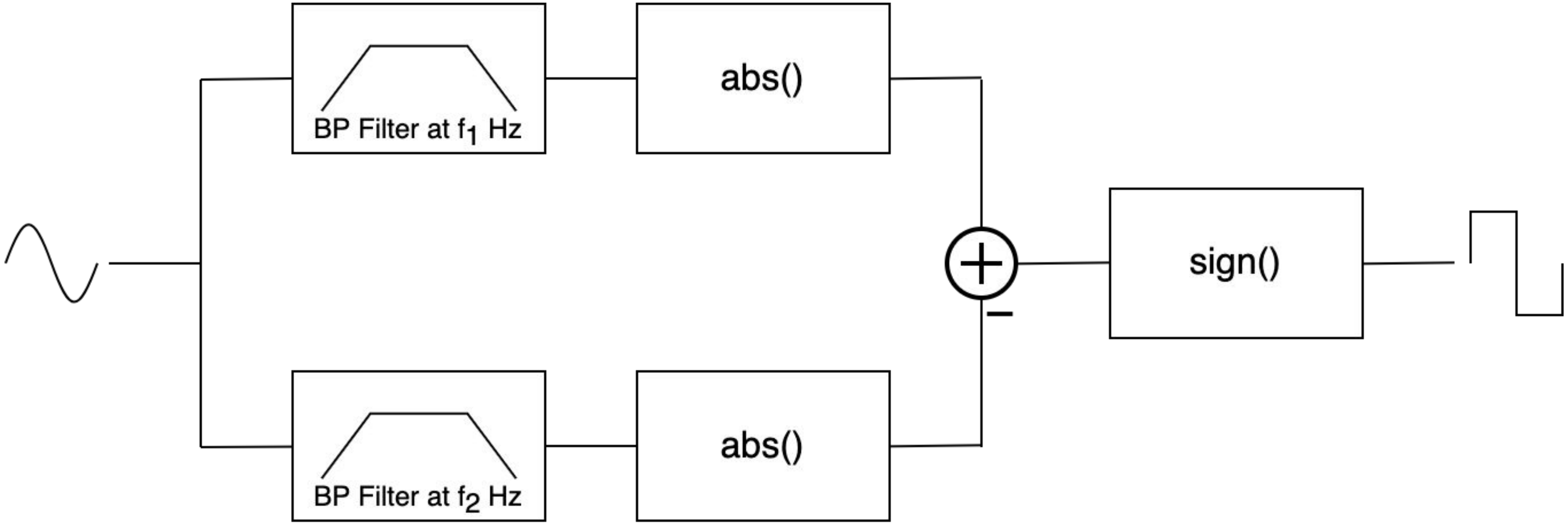
Was instead demodulated by hand using *Audacity.*



Task 2: Method

Complicated process involving many steps.

Essentially, implements this simplified circuit:

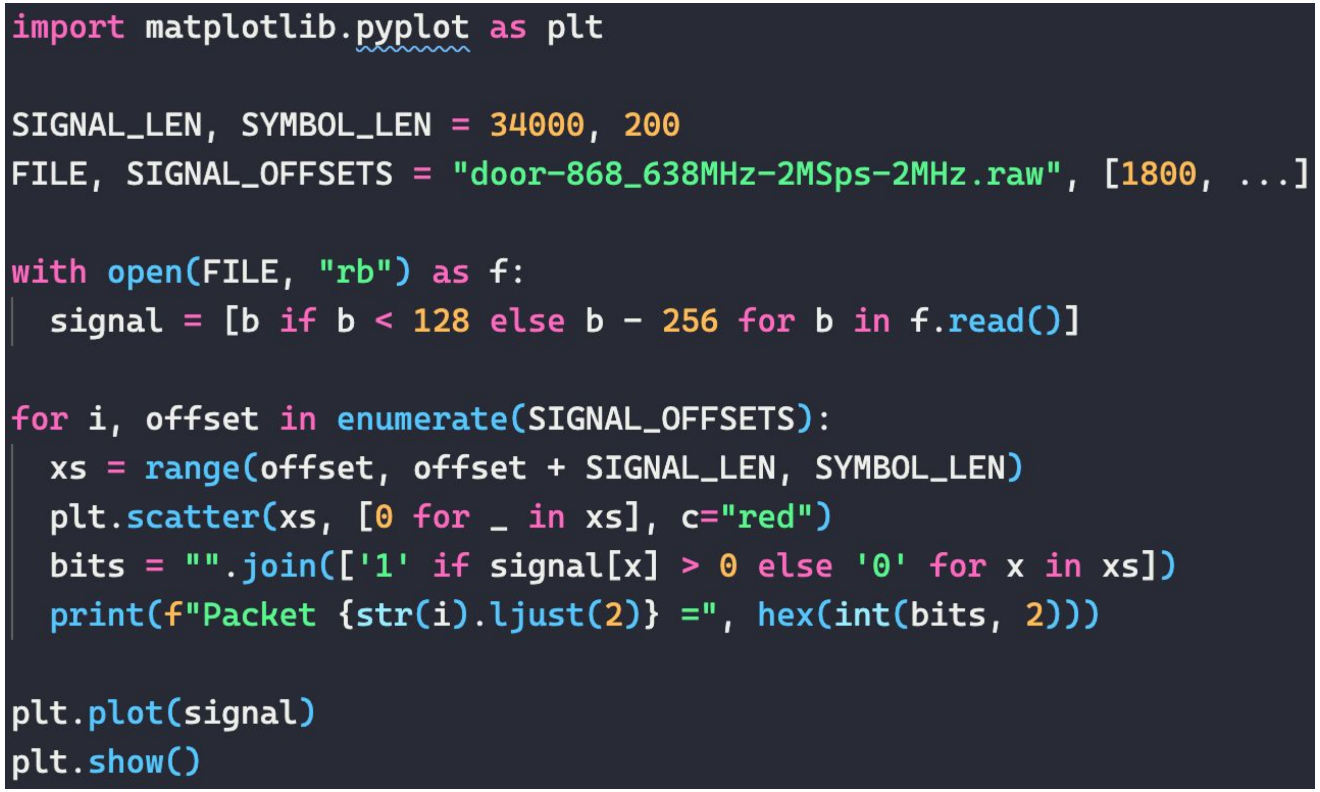


Task 2: Method

Lastly, the binary wave had to be decoded into binary data.

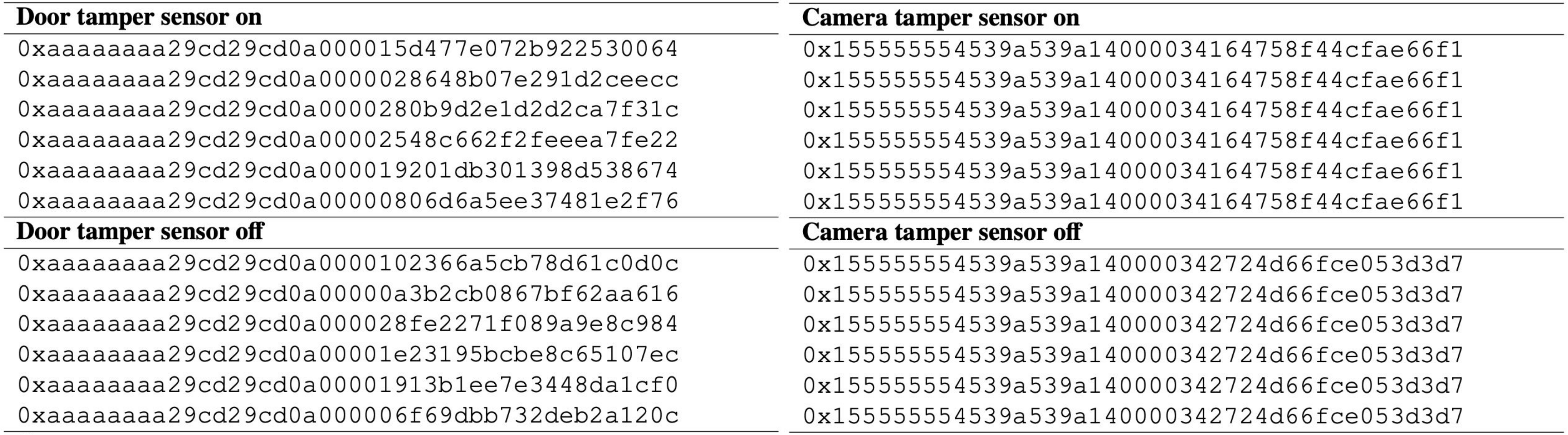
This was done using a small python script.

Parameters were measured by hand, using a *matplotlib* graph.



```
import matplotlib.pyplot as plt

SIGNAL_LEN, SYMBOL_LEN = 34000, 200
FILE, SIGNAL_OFFSETS = "door-868_638MHz-2MSps-2MHz.raw", [1800, ...]

with open(FILE, "rb") as f:
  signal = [b if b < 128 else b - 256 for b in f.read()]

for i, offset in enumerate(SIGNAL_OFFSETS):
  xs = range(offset, offset + SIGNAL_LEN, SYMBOL_LEN)
  plt.scatter(xs, [0 for _ in xs], c="red")
  bits = "".join(['1' if signal[x] > 0 else '0' for x in xs])
  print(f"Packet {str(i).ljust(2)} =", hex(int(bits, 2)))

plt.plot(signal)
plt.show()
```

Task 2: RF Reverse engineering - Result

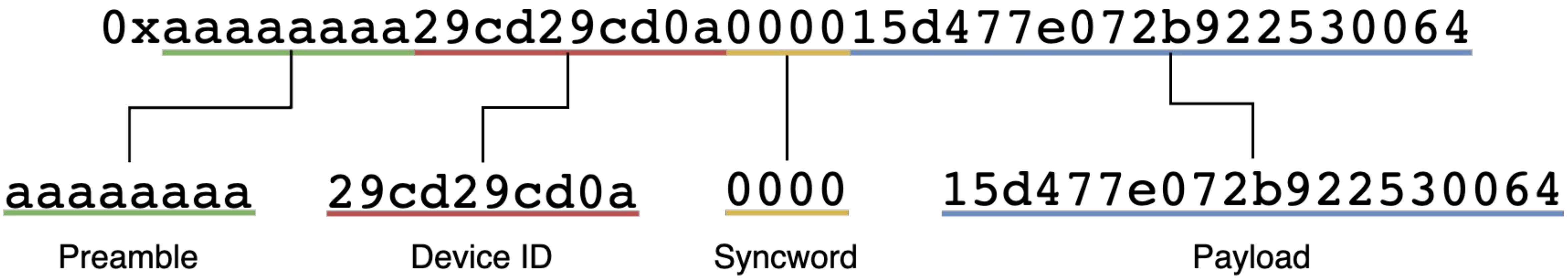


Door tamper sensor on	Camera tamper sensor on
0xaaaaaaaa29cd29cd0a000015d477e072b922530064	0x155555554539a539a14000034164758f44cfae66f1
0xaaaaaaaa29cd29cd0a0000028648b07e291d2ceecc	0x155555554539a539a14000034164758f44cfae66f1
0xaaaaaaaa29cd29cd0a0000280b9d2e1d2d2ca7f31c	0x155555554539a539a14000034164758f44cfae66f1
0xaaaaaaaa29cd29cd0a00002548c662f2feeea7fe22	0x155555554539a539a14000034164758f44cfae66f1
0xaaaaaaaa29cd29cd0a000019201db301398d538674	0x155555554539a539a14000034164758f44cfae66f1
0xaaaaaaaa29cd29cd0a00000806d6a5ee37481e2f76	0x155555554539a539a14000034164758f44cfae66f1
Door tamper sensor off	**Camera tamper sensor off**
0xaaaaaaaa29cd29cd0a0000102366a5cb78d61c0d0c	0x155555554539a539a140000342724d66fce053d3d7
0xaaaaaaaa29cd29cd0a00000a3b2cb0867bf62aa616	0x155555554539a539a140000342724d66fce053d3d7
0xaaaaaaaa29cd29cd0a000028fe2271f089a9e8c984	0x155555554539a539a140000342724d66fce053d3d7
0xaaaaaaaa29cd29cd0a00001e23195bcbe8c65107ec	0x155555554539a539a140000342724d66fce053d3d7
0xaaaaaaaa29cd29cd0a00001913b1ee7e3448da1cf0	0x155555554539a539a140000342724d66fce053d3d7
0xaaaaaaaa29cd29cd0a000006f69dbb732deb2a120c	0x155555554539a539a140000342724d66fce053d3d7

We now have a bunch of bits!

One can clearly see a structure in the messages.

Task 2: RF Reverse engineering - Result



0xaaaaaaaa29cd29cd0a000015d477e072b922530064

aaaaaaaa

Preamble

29cd29cd0a

Device ID

0000

Syncword

15d477e072b922530064

Payload

Task 2: Discussion

The messages follow a classic structure for RF protocols.

The payload is encrypted, or at least obfuscated.

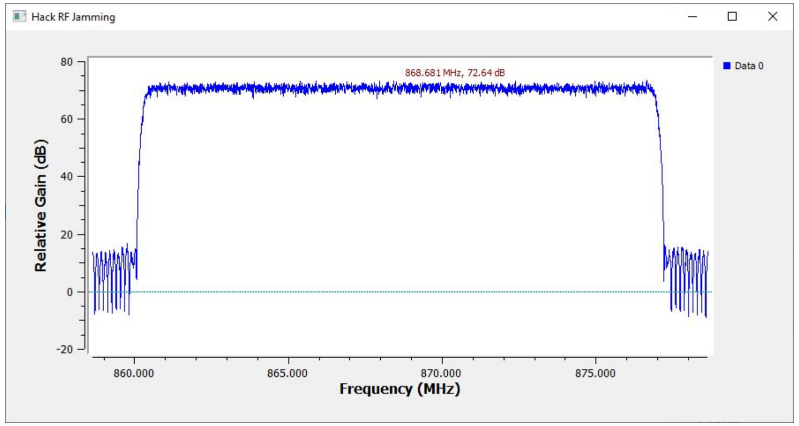
Further investigation would most likely require reverse engineering the firmware.

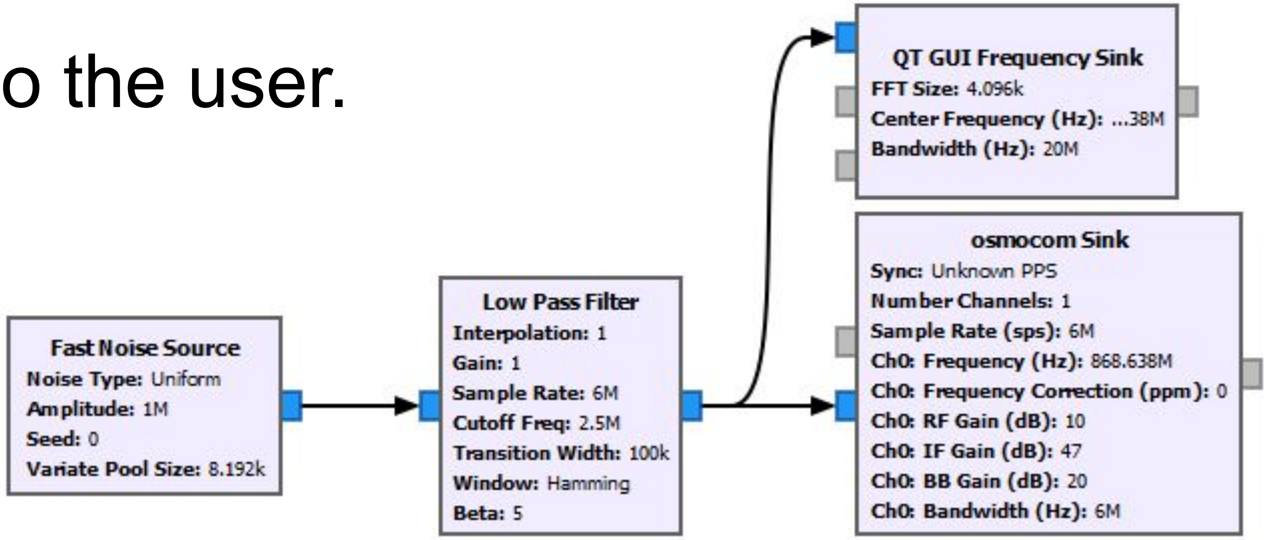
Task 3: RF Jamming Attack

The RF equivalent of a *DoS* attack.
Almost impossible to protect against.

This is detected by the system.

However, it is not reported to the user.





Fast Noise Source
Noise Type: Uniform
Amplitude: 1M
Seed: 0
Variate Pool Size: 8.192k

Low Pass Filter
Interpolation: 1
Gain: 1
Sample Rate: 6M
Cutoff Freq: 2.5M
Transition Width: 100k
Window: Hamming
Beta: 5

QT GUI Frequency Sink
FFT Size: 4.096k
Center Frequency (Hz): ...38M
Bandwidth (Hz): 20M

osmocom Sink
Sync: Unknown PPS
Number Channels: 1
Sample Rate (sps): 6M
Ch0: Frequency (Hz): 868.638M
Ch0: Frequency Correction (ppm): 0
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 47
Ch0: BB Gain (dB): 20
Ch0: Bandwidth (Hz): 6M

Task 4: Insecure Network Services

Very common source of vulnerabilities (OWASP IoT #2).
Three services on the system, found via *Nmap* port scanning.

53/tcp, 53/udp	(DNS)
80/tcp	(HTTP)
58098/tcp	(Backdoor?)

No additional vulnerabilities found. However, the services are all seemingly unnecessary and the last one is very *suspicious*.

Conclusion

Conclusion

The system does a lot of things right:

- GSM telecommunication
- Backup battery
- Tamper sensors on all devices
- Encryption in most communication channels

However, in cybersecurity one mistake can be all it takes.

Among other smaller vulnerabilities, a glaring security flaw in the RF protocol was found. This allows an attacker to disarm an armed system.

Thanks for Listening!

Questions?