

Hacking The IoT (Internet of Things) - PenTesting RF Operated Devices

Erez Metula

Application Security Expert

Founder, AppSec Labs

ErezMetula@AppSec-Labs.com



About Me



- 📖 Founder of AppSec Labs

- 📖 OWASP IL Board

- 📖 Application security expert

- 📖 Book author

 - 📖 Managed Code Rootkits (Syngress)

- 📖 Speaker & trainer

 - 📖 Presented at BlackHat, Defcon, RSA, OWASP USA, OWASP IL, etc..

 - 📖 Secure coding/hacking trainer



Agenda

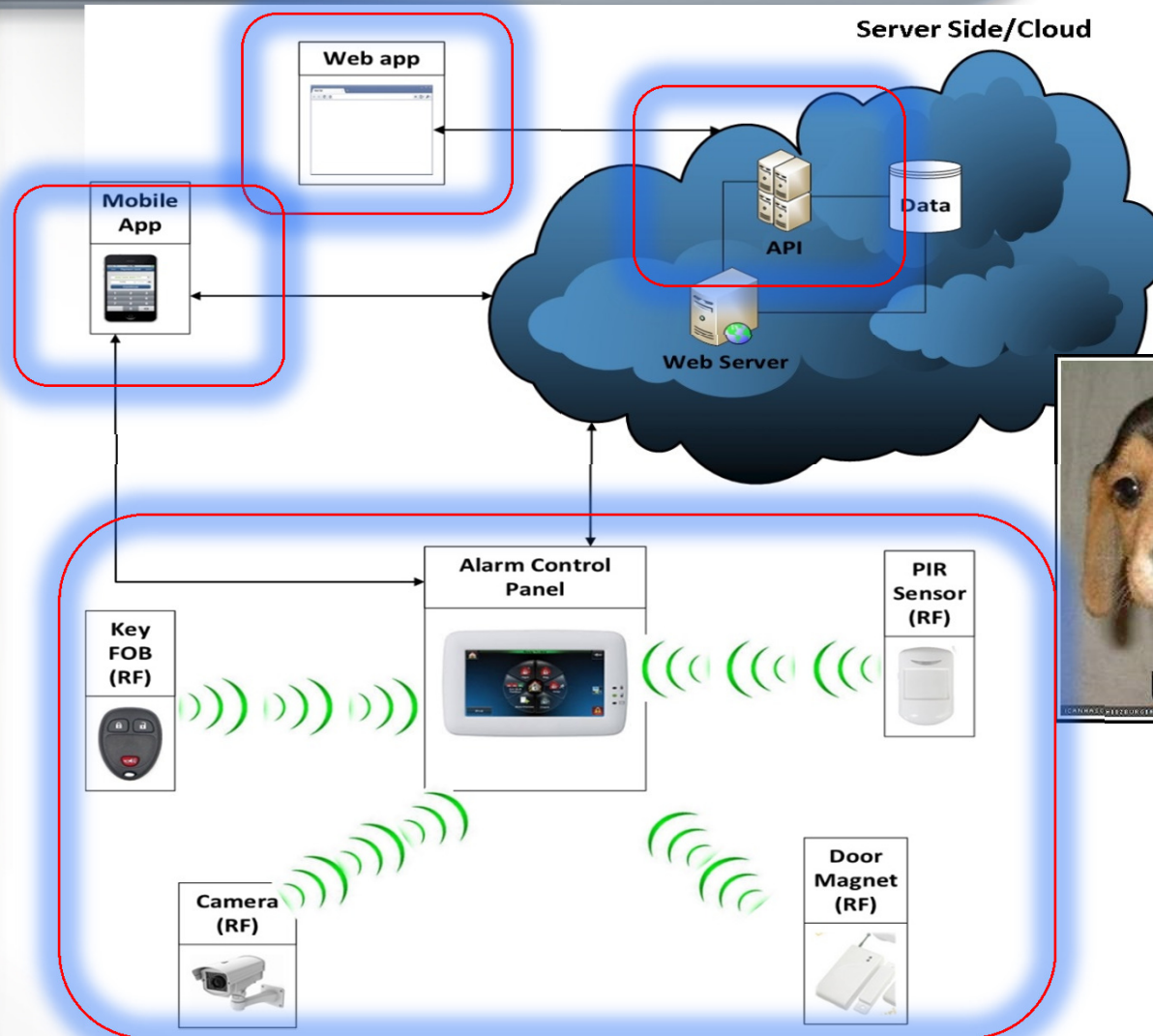


- 📄 Intro
- 📄 Basic RF terminology & Intro to SDR (Software Defined Radio)
- 📄 Capturing & replaying RF transmissions
- 📄 Reverse engineering unknown RF protocol: step-by-step
- 📄 Playing with an IoT wireless alarm system
- 📄 Breaking car key FOB (and RF operated devices in general)
- 📄 Replay, transmission and message tampering
- 📄 Jamming

Why Focus on RF?



Common IoT Architecture ..and the target system for today..



Common RF Protocols Used in IoT







- ▣ Bluetooth/BLE (2.4GHz)
 - ▣ Zigbee (2.4GHz)
 - ▣ Z-wave (900MHz)
 - ▣ Wifi (2.4GHz)
 - ▣ Cellular (900/1800/1900/2100MHz)
 - ▣ Custom RF (?)
-
- ▣ **The focus of this talk will be custom RF as it doesn't have any generic tool but rather a set of tools/techniques**

Common Unlicensed Bands



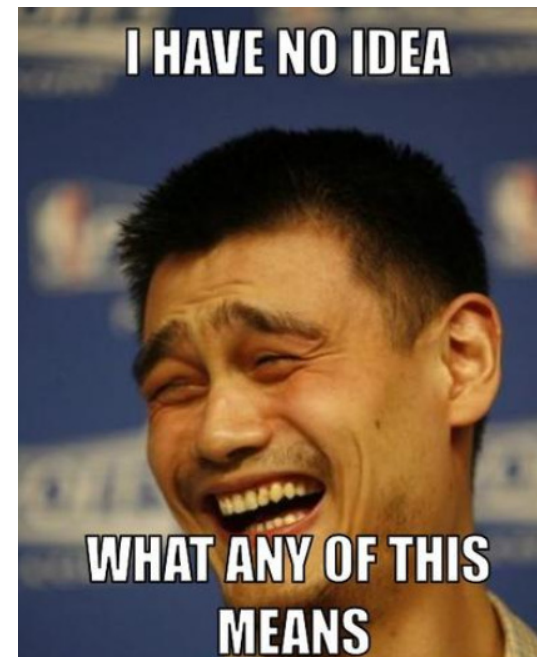
Common ISM bands used in IoT systems

-  315 MHz
-  433 MHz
-  902-928 MHz
-  863 – 870 MHz
-  2.4 GHz
-  5.8 GHz

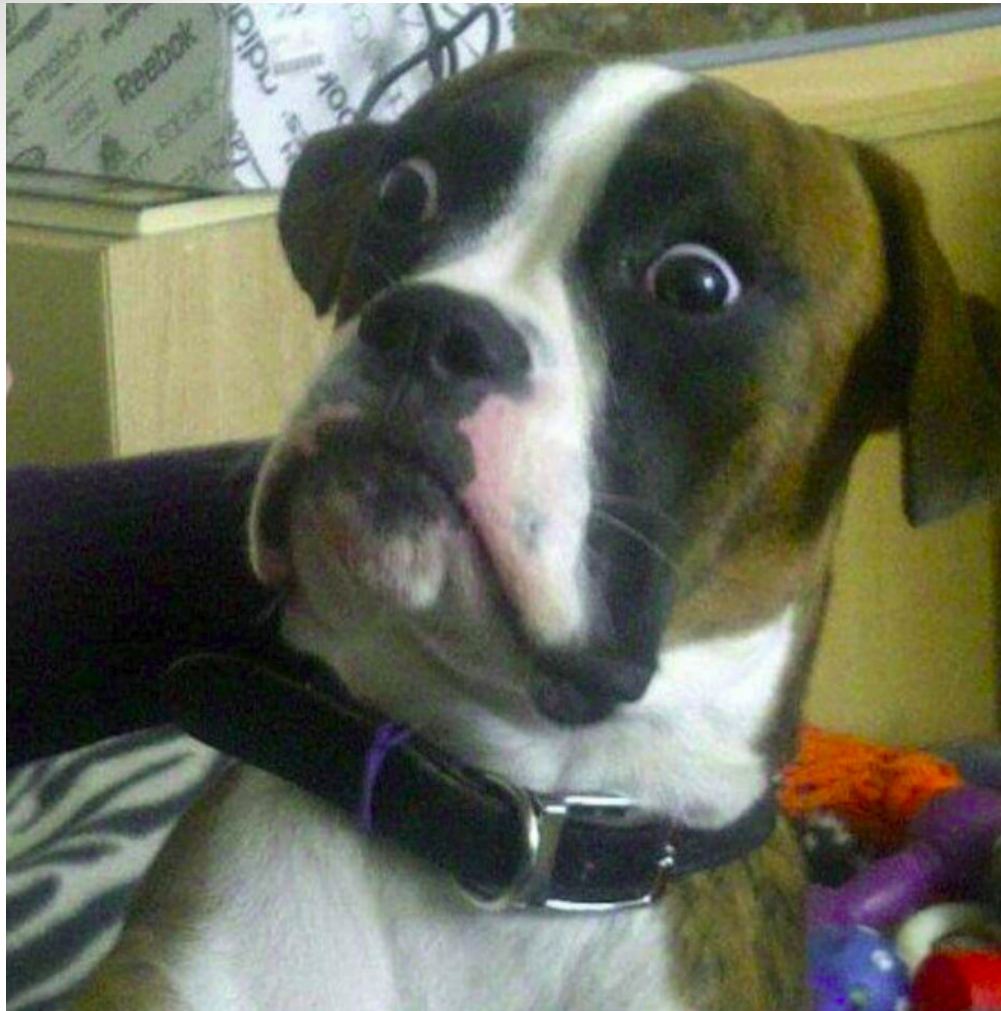
Reverse Engineering Unknown RF Transmissions

☐ Some basic questions we'll need to answer

- ☐ Frequency
- ☐ Channel width
- ☐ Modulation
- ☐ Bit rate
- ☐ Preamble
- ☐ Sync word
- ☐ CRC?
- ☐ Whitening



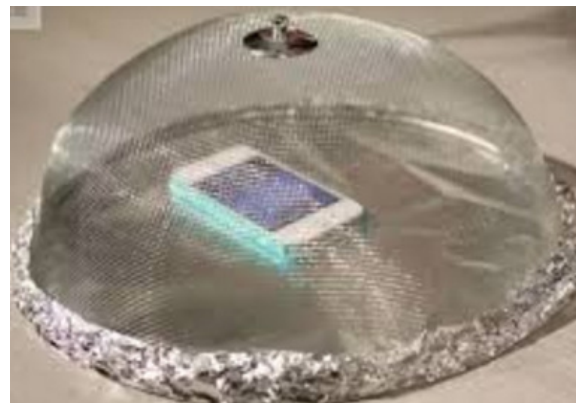
WARNING



© All rights reserved to AppSec Labs Ltd.

WARNING

- ☐ The spectrum is a limited public resource
- ☐ Playing with illegal radio might get you arrested...
 - ☐ It can interfere with critical systems, homeland security, army, etc.
- ☐ It is advised to use a Faraday cage (“RF cage”)



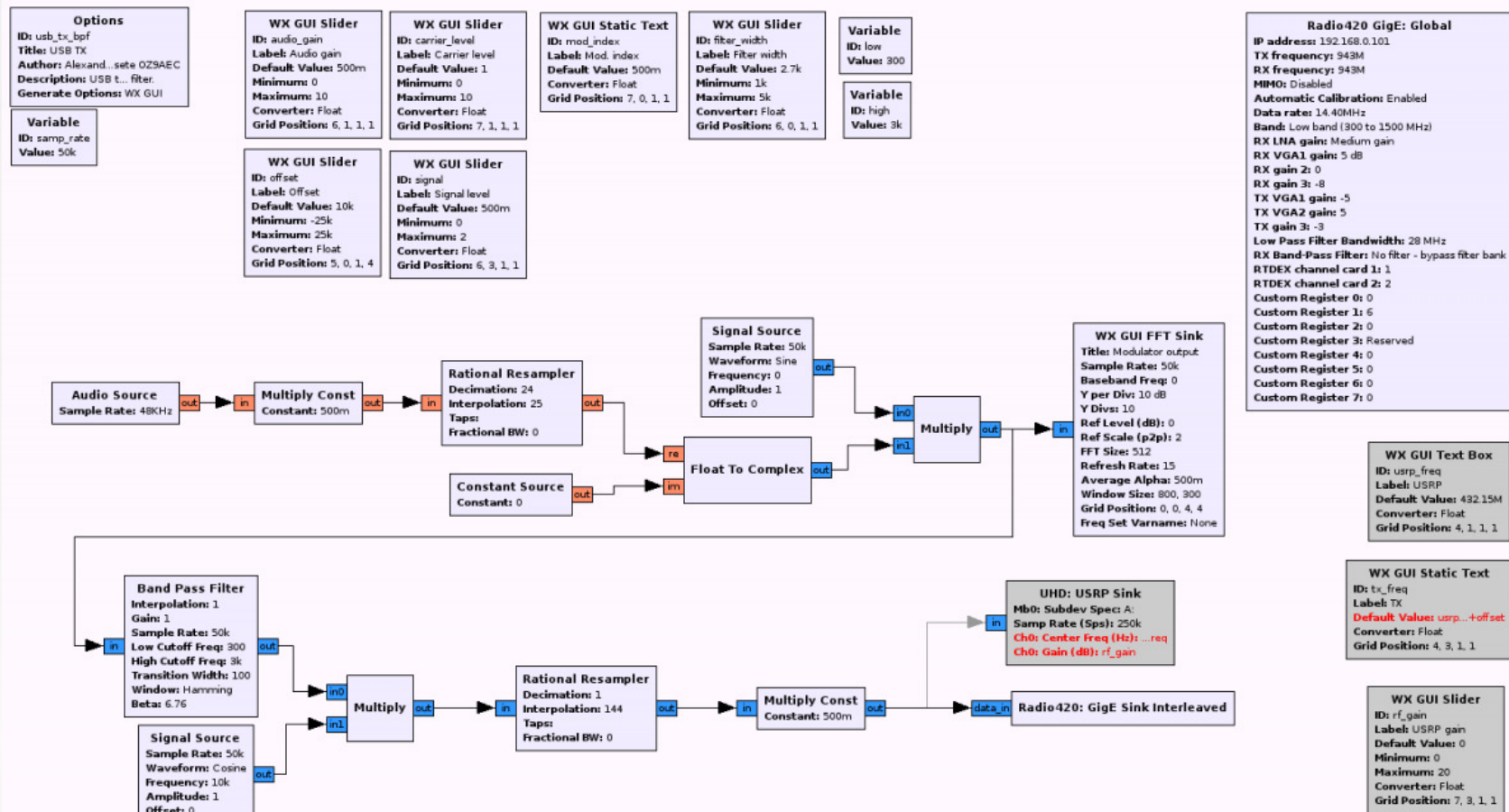
Software Defined Radio (SDR)



- ▢ a radio communication system where components that have been typically implemented in hardware (mixers, filters, amplifiers, modulators, etc.) are instead implemented by means of software
- ▢ Drag and drop components and create a flowgraph!

Software Defined Radio (SDR)

gnuradio



HackRF One

- ❏ A Software Defined Radio peripheral capable of transmission or reception of radio signals from 1 MHz to 6 GHz.
- ❏ One of the best peripherals that are out there
- ❏ Can receive and transmit
- ❏ Cost: 300\$



RTL-SDR

- ❏ RTL-SDR is a very cheap software defined radio that uses a DVB-T TV tuner dongle based on the RTL2832U chipset.
- ❏ Essentially, this means that a cheap [\\$20 TV tuner USB dongle with the RTL2832U chip](#) can be used as a computer based SDR.
- ❏ Drawback – receive only. But still great for analysis!



DEMO – Disarming an Alarm System Using Replay Attack



📄 Zero knowledge replay attack

📄 Record

📄 `hackrf_transfer -r 433780000.raw -f 433780000`

📄 Transmit

📄 `hackrf_transfer -t 433780000.raw -f 433780000 -x 20`

Replay Advantage/Disadvantage



▣ Advantage

- ▣ Zero knowledge
- ▣ Effective even if the message is encrypted

▣ Disadvantage

- ▣ Cannot create a valid message from scratch
- ▣ Cannot “play” with messages - many times you’d like to modify a message based on the original one
 - ▣ Tamper with ID
 - ▣ Tamper with command
 - ▣ Perform input validation attacks
 - ▣ Etc.

- ▣ We must be able to analyze the signal so we can recreate it manually “from scratch”

Planning

1. Information gathering
2. Frequency
3. Modulation
4. Deviation
5. Preamble/syncword
6. Symbol rate
7. Transmission!

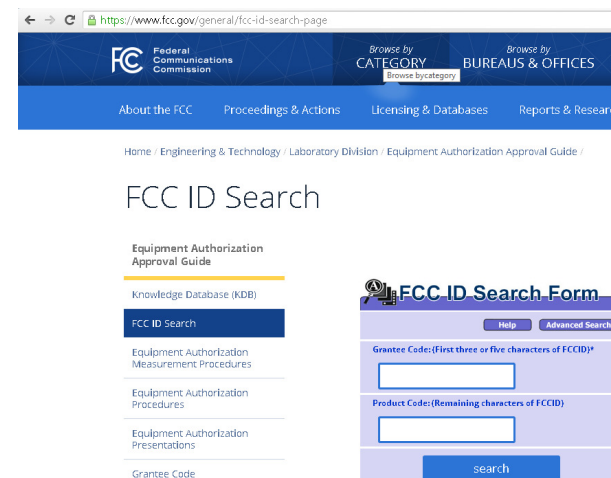
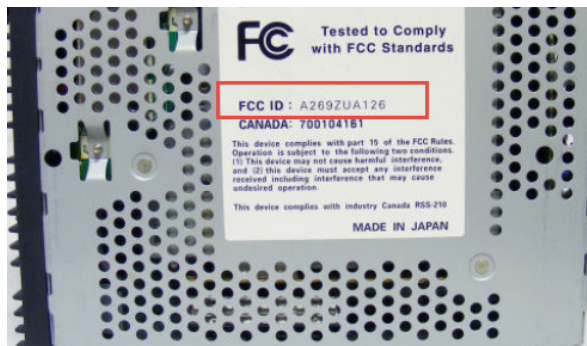


Step 1 – Information Gathering

📁 A good starting point – if you have some luck – search for the FCC ID

📁 <https://www.fcc.gov/general/fcc-id-search-page>

📁 <https://fccid.io/> demo: <https://fccid.io/Y8PFJ17-1>

A screenshot of the FCC ID Search page. The page header includes the FCC logo and navigation links. The main content area is titled 'FCC ID Search' and includes a sidebar with links to 'Equipment Authorization Approval Guide', 'Knowledge Database (KDB)', 'FCC ID Search', 'Equipment Authorization Measurement Procedures', 'Equipment Authorization Procedures', 'Equipment Authorization Presentations', and 'Grantee Code'. The main search area is titled 'FCC ID Search Form' and contains input fields for 'Grantee Code' and 'Product Code', along with a 'Search' button.

Step 1 – Information Gathering

Information extracted from the FCC site

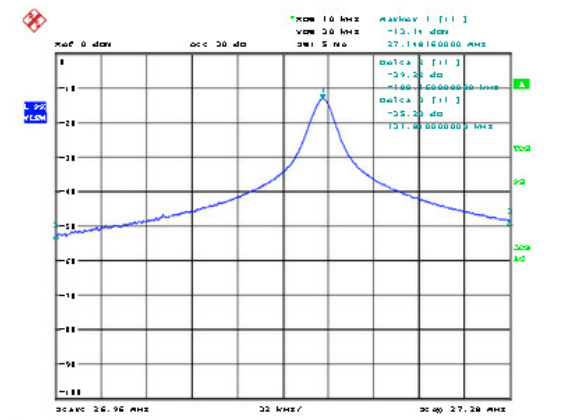
External photos	External Photos Adobe Acrobat PDF (131 kB)
Antenna spec	Operational Description Adobe Acrobat PDF (2693 kB)
RF Test Report	Test Report Adobe Acrobat PDF (1248 kB)
LTC Letter	Cover Letter(s) Adobe Acrobat PDF (89 kB)
label and Label location	ID Label/Location Info Adobe Acrobat PDF (540 kB)
Block Diagram	Block Diagram Adobe Acrobat PDF (434 kB)
RF Test Set-up Photos	Test Setup Photos Adobe Acrobat PDF (331 kB)
POA	Cover Letter(s) Adobe Acrobat PDF (99 kB)
Internal photos	Internal Photos Adobe Acrobat PDF (1222 kB)

The miscellaneous information includes details of the test procedure and measured bandwidth / calculation of factor such as pulse desensitization and averaging factor (calculation and timing diagram).

8.1 Measured Bandwidth






The plot shows the fundamental emission is confined in the specified band. And it also shows that the emission is at least 35.20 dB below the carrier level at the band edge (26.96 and 27.28 MHz). It meets the requirement of Section 15.227(b).

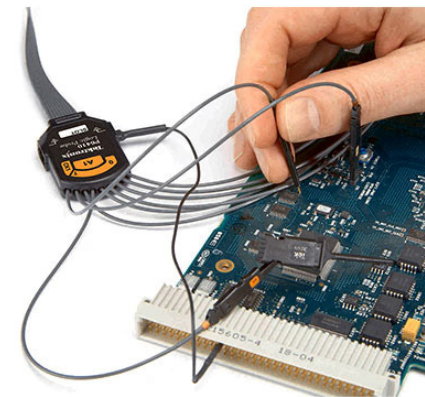
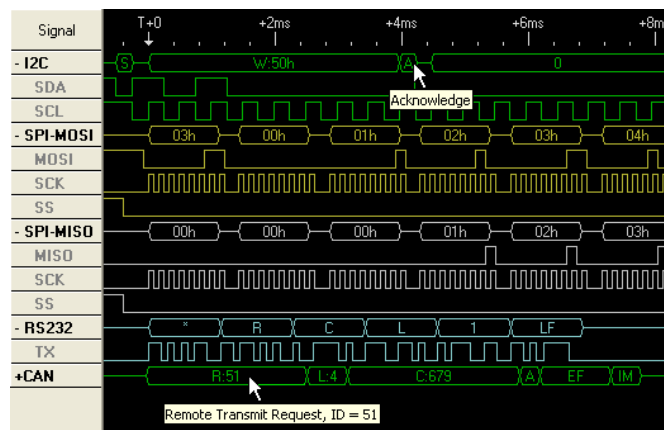
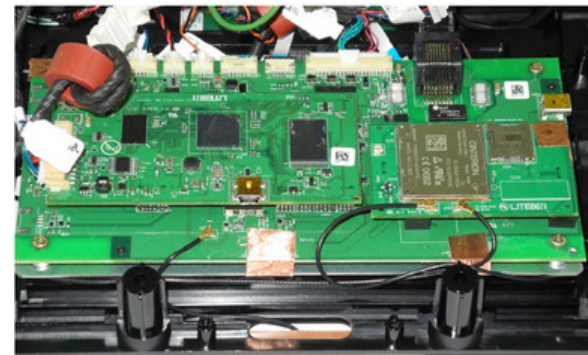
Pursuant to FCC Part 15 Section 15.215(c), the 20dB bandwidth of the emission was contained within the frequency band designed (mentioned as above) which the EUT operated. The effects, if any, from frequency sweeping, frequency hopping, other modulation techniques and frequency stability over expected variations in temperature and supply voltage were considered.



Step 1 – Information Gathering

Do some hardware research (optional)

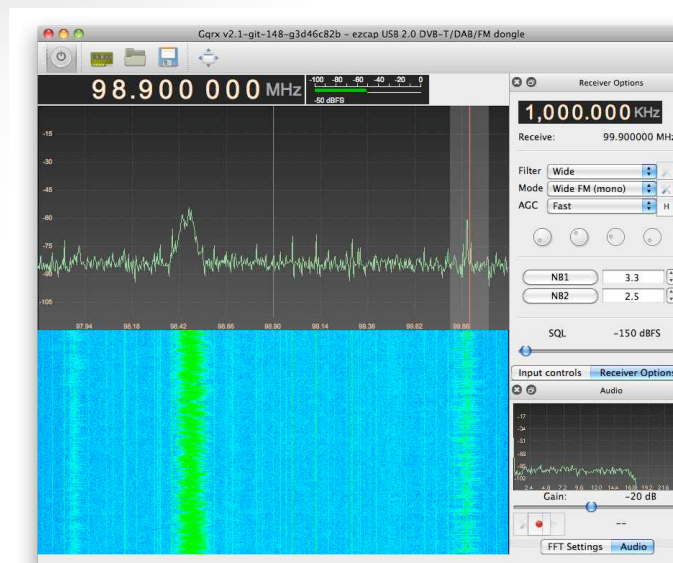
-  Open the device, and look at the PCB
-  Identify major electric components, mainly the microcontroller and the RF module
-  Look for UART/serial/etc.
-  Use a logical analyzer
-  Etc.



Step 2 - Frequency

 Use a spectrum analyzer

 Gqrx:



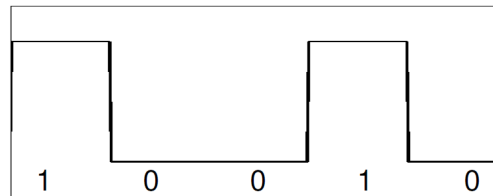
 You can then dump the signal to a file:

```
rtl_sdr -f 433950000 -s 2000000 -g 20 captured.cu8
```

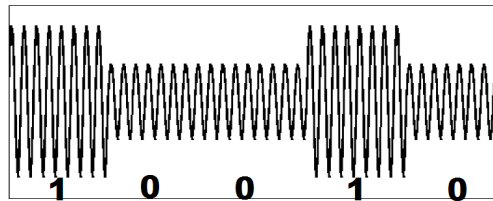
Step 3 - Modulation

Representing digital data as variations in the carrier wave

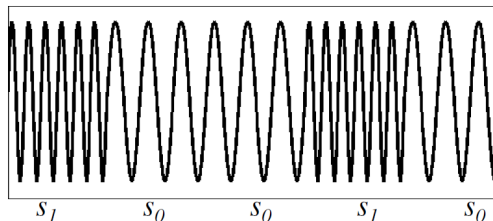
Baseband data:



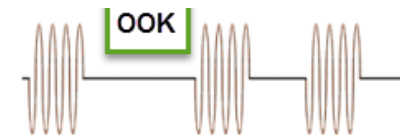
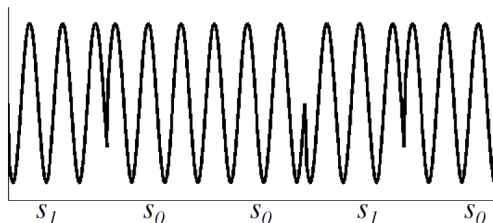
ASK modulated signal:



FSK modulated signal:

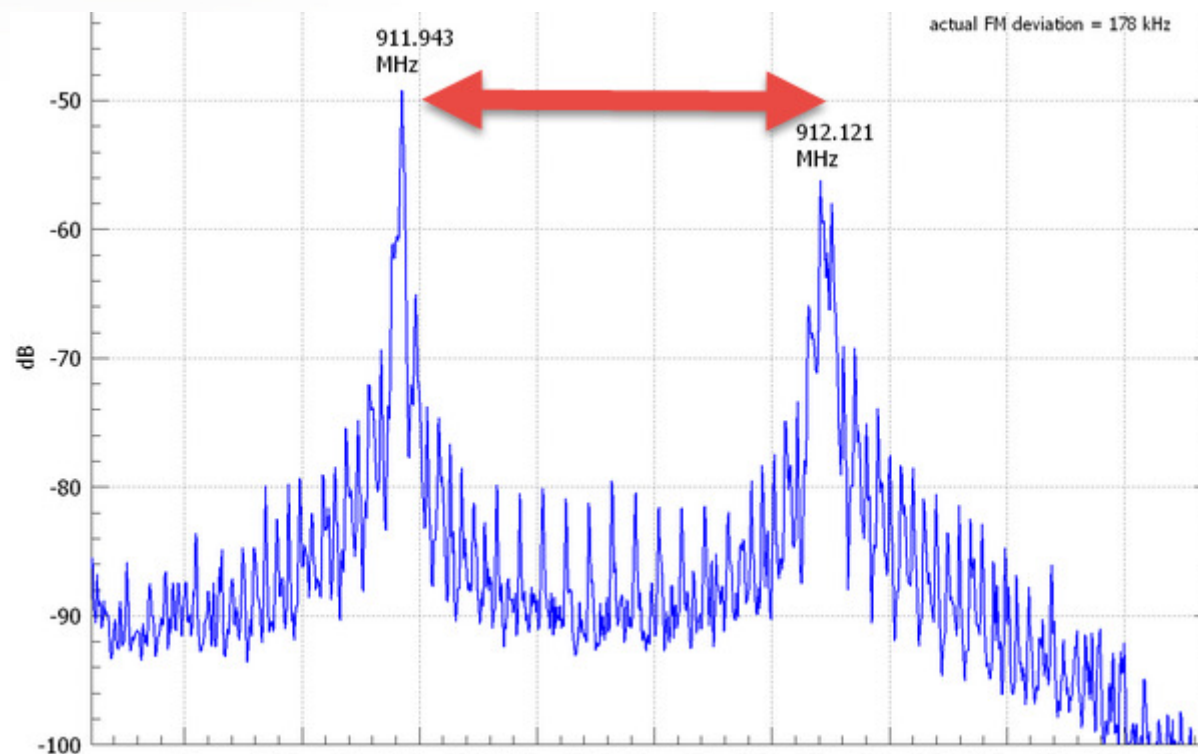


PSK modulated signal:



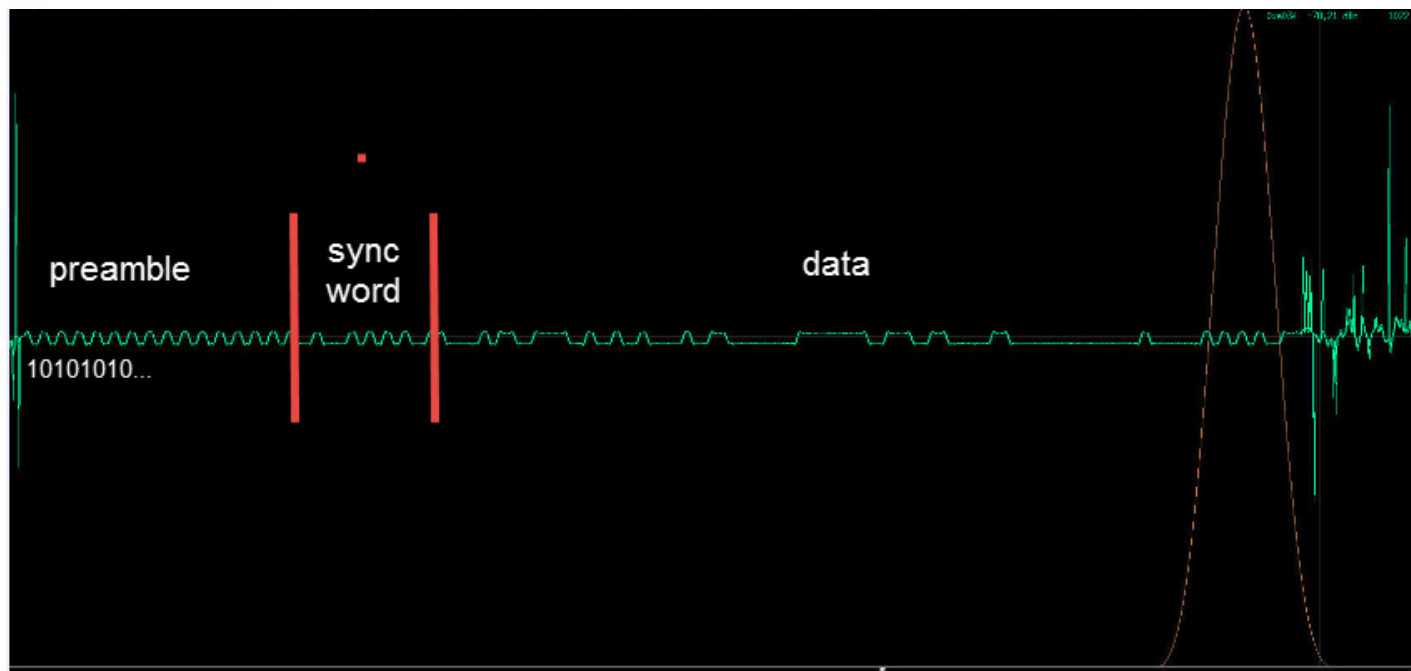
Step 4 - Deviation

 Often relevant for FSK



Step 5 - Preamble/Sync Word

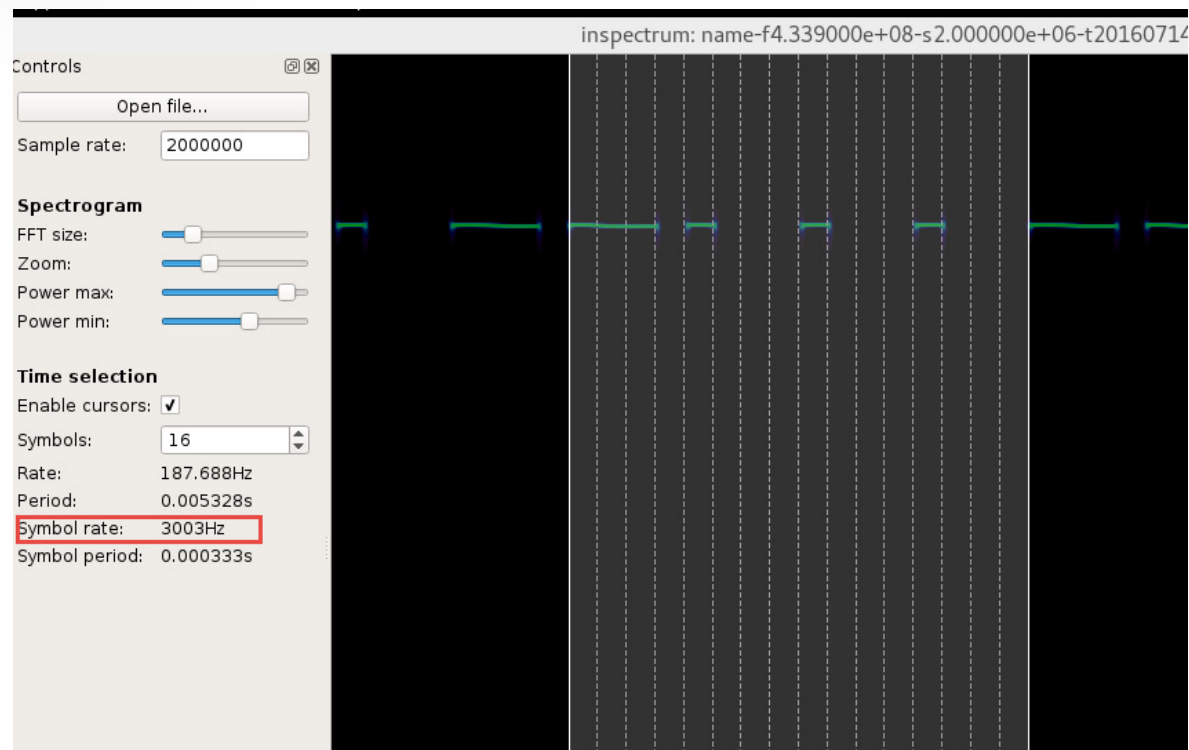
- ▣ Preamble = start of message
- ▣ Sync word = start of data



Step 6 - Symbol Rate

 Symbol = 0 or 1

 Symbol rate = symbols per second



DEMO – Analyzing the Signal



- ▣ Capture a signal (“disarming”)
- ▣ Analyze the signal
- ▣ Extract all the required info

Checklist - What We Know So Far..

- ☐ Frequency = 433.92 MHz
- ☐ Modulation = ASK/OOK
- ☐ Bit rate = 3000 bits per second
- ☐ Sync word = no
- ☐ Preamble = no
- ☐ Message structure = System ID, Command

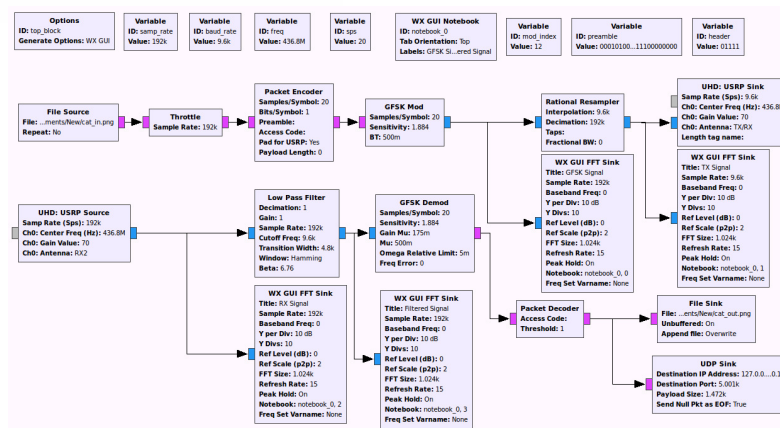


Step 7 – Transmission!

Now it's time to generate the message from scratch

Option 1 – use a flow graph

Too complicated, not interactive



Option 2 - use a commandline RF tool

Simple to use

Interactive!

Yardstick One and RFCat

- ▣ Yardstick one - a sub-1 GHz wireless test tool (developed by Michael Ossmann)
- ▣ Rfcats – Python-based firmware for *transmitting and receiving RF* (Developed by atlas)



```
'RfCat, the greatest thing since Frequency Hopping!'

Research Mode: enjoy the raw power of rflib

currently your environment has an object called "d" for dongle.  this is how
you interact with the rfcats dongle:
>>> d.ping()
>>> d.setFreq(433000000)
>>> d.setMdmModulation(MOD_ASK_00K)
>>> d.makePktFLEN(250)
>>> d.RFxmmit("HALLO")
>>> d.RFrecv()
>>> print d.reprRadioConfig()

In [1]:
```

DEMO – Disarming an Alarm System by Rebuilding the Transmission From Scratch



- ▣ Setup rfcats with all required info
- ▣ Send our own transmissions!

So we know..



- ▣ What is the actual message
- ▣ The message structure
 - ▣ 20 bit system ID, 4 bit command ID
- ▣ We can brute force ANY system in about 2 hours
- ▣ Demo – if time permits
 - ▣ Brute force calculation
 - ▣ Command ID fuzzing

Jamming

- ▣ A device that deliberately interferes with authorized wireless communications
- ▣ Resembles DoS
- ▣ JAMMING IS ILLEGAL
- ▣ Disable any commands that are sent to the system
 - ▣ Example: prevent the alarm system from arming
- ▣ Disrupt any information that is sent between the components
 - ▣ Example: disrupt the PIR movement sensor
- ▣ Attack is against the receiver (though logically it is against the sender)



Demo – Simple “Jamming”



 No more motion detection..

Crypto Challenges



- ❑ Encryption is less used (compared to traditional “modern” applications)
- ❑ Hardware challenges
 - ❑ CPU
 - ❑ Memory
 - ❑ Flash space (example: 2k RSA key)
 - ❑ Battery power
 - ❑ Lack of receiving ability (i.e. transmitter only hardware)
- ❑ We'll often encounter
 - ❑ No encryption at all
 - ❑ Lack of challenge response
 - ❑ Weak crypto algorithms

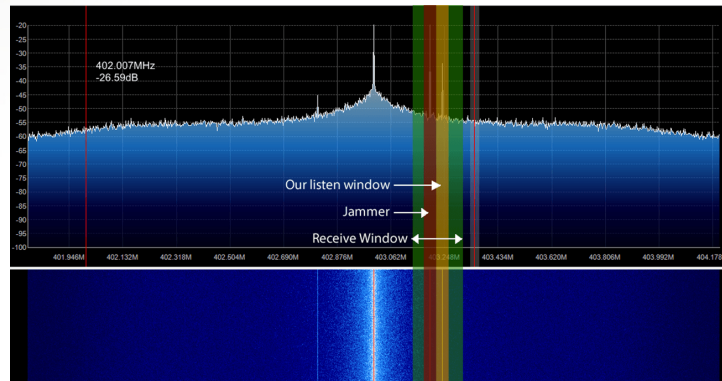
Rolling Code ("Hopping Code")



- ❏ Prevents [replay attacks](#) – attacker might record a fixed code transmission and replays it at a later time to cause the receiver to 'unlock'
- ❏ Useful in 1-way only communications systems
- ❏ Common implementation
 - ❏ Common PRNG — in both transmitter and receiver
 - ❏ Transmitter sends 'next' code in sequence
 - ❏ Receiver compares 'next' to its calculated 'next' code. A tolerance of next 256 codes is accepted in case receiver missed some transmitted keypresses.

Rolling Code Attacks

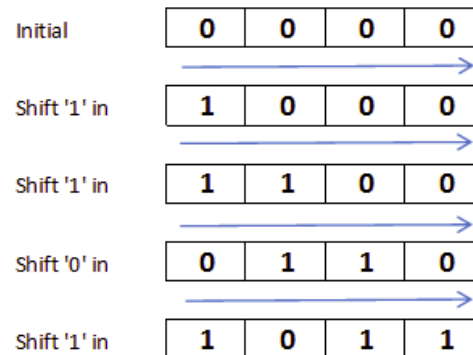
- It was demonstrated in 2015 by [Samy Kamkar](#) how a simple but clever attack can break rolling codes



Steps

- The device transmits a jamming signal to block the vehicle's reception of rolling code signals sent by the victim
- The victim tries again, sending the 2nd code
- The device records these signals from both 1st and 2nd attempts needed to unlock the vehicle.
- The recorded 1st code is forwarded to the vehicle only when the owner makes the 2nd attempt
- The recorded 2nd code is retained for future use

Shift-Registers



- It is common for bits that are read from the air to be stored in a **shift-register**
- A portion of the stream (“window”) is examined each time
- Each output is connected to the next input
- A circuit that shifts the 'bit array' stored in it by one position

The Meaning

- ☐ If there are some extra bits before/after the real value, it will be accepted by the receiver!
- ☐ We can sent 13 bits to test **two** 12-bit codes instead of sending a full 24 bits

0001011100

10 bit

=

0000010101010101111110100

24 bit

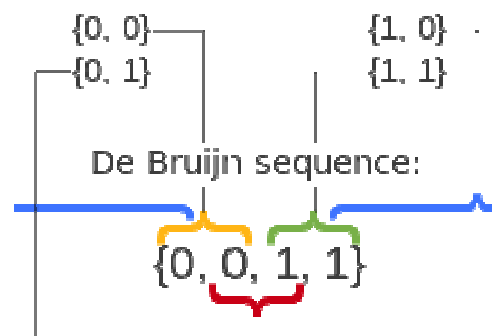


De-Bruijn Sequences

- ☐ a cyclic sequence in which every possible length- n string on A occurs exactly once as a substring
- ☐ Such a sequence is denoted by $B(k, n)$ and has length k^n , which is also the number of distinct substrings of length n on A ; De Bruijn sequences are therefore optimally short

Alphabet: $\{0, 1\}$
Subsequence length: 2

Subsequences:



De-Bruijn Sequences

There are $\frac{(k!)^{k^{n-1}}}{k^n}$ distinct De Bruijn sequences $B(k, n)$

Example –

- The pin code is composed of 0-9 digits, length = 4
- Trying all codes separately would require 4 (byte) × 10000 (possibilities) = 40000 byte.
- Using De-Bruijn we have $B(10, 4)$ solutions, with length 10000. Therefore, only at most $10000 + 3 = 10003$ (as the solutions are cyclic) bytes are needed to open the lock.

Demo – if Time Permits



```
def de_bruijn(k, n):  
    """  
    De Bruijn sequence for alphabet k  
    and subsequences of length n.  
    """  
    try:  
        # let's see if k can be cast to an integer;  
        # if so, make our alphabet a list  
        _ = int(k)  
        alphabet = list(map(str, range(k)))  
  
    except (ValueError, TypeError):  
        alphabet = k  
        k = len(k)  
  
    a = [0] * k * n  
    sequence = []  
  
    def db(t, p):  
        if t > n:  
            if n % p == 0:  
                sequence.extend(a[1:p + 1])  
            else:  
                pass
```

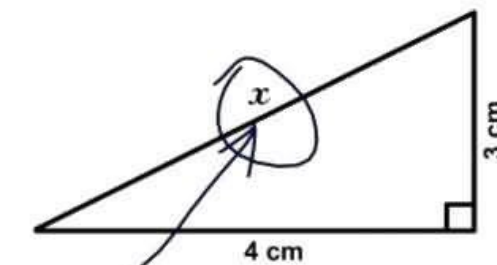
Summary



- 📄 IoT security requires you to look beyond the server side and mobile app security
- 📄 For simple a replay, a good SDR device will just do
- 📄 It is advised to analyze the transmissions and reverse engineer them
- 📄 “security by obscurity” is often encountered
- 📄 Now go hack the RF world.. 😊

QUESTIONS ?

3. Find x .



Here it is

Thank You!

...now go to see Tal's talk..

Erez Metula

Chairman & Founder, AppSec Labs

ErezMetula@AppSec-Labs.com