SO YOU WANT TO HACK RADIOS

A PRIMER ON WIRELESS REVERSE ENGINEERING
WHO ARE THESE GUYS

- Marc “mou$e whisperer” Newlin
  - Security Researcher @ Bastille
  - Discovered Mousejack vulnerability in 2016
  - Finished 2nd in DARPA Spectrum Challenge in 2013
  - Finished 3rd in DARPA Shredder Challenge in 2011

- Matt Knight
  - Software Engineer and Security Researcher @ Bastille
  - Reverse engineered the LoRa wireless protocol in 2016
  - BE & BA from Dartmouth

marc@Bastille.net
@marcnewlin

matt@Bastille.net
@embeddedsec
WHO IS THIS FOR?
WHY SHOULD YOU CARE?
WIRELESS SYSTEMS ARE EVERYWHERE
MOBILE
WIRELESS SYSTEMS ARE EVERYWHERE
MOBILE
WIRELESS SYSTEMS ARE EVERYWHERE
IOT
Fewer wires every year!
ABOUT THE INTERNET OF THINGS...

▸ Everyone’s Favorite Buzzword™
▸ What is it, actually?
   ▸ Sales and marketing speak for “connected embedded devices”
   ▸ “Smart” devices are usually pretty stupid
EMBEDDED REALITIES

- Embedded systems are built on compromise
  - Size and cost constraints
  - Battery powered
  - Challenging deployment scenarios
  - Difficult to patch

Vulnerable by Virtue of Being Constrained
ALARM SYSTEM VULNERABILITIES

- Discovered by Bastille’s Logan Lamb in 2014
- Legacy RF link between home alarm system sensors and control panel is vulnerable to:
  - Jamming (denying alarm reporting)
  - Command injection (trigger false alarms)
  - Eavesdropping (detect occupancy, monitor movement)
MOUSEJACK

- Discovered by Bastille's Marc Newlin in 2015
- RF link between non-Bluetooth wireless keyboards and mice (100MMs of devices) vulnerable to:
  - Command injection (running arbitrary commands at current permissions level)
  - Eavesdropping (sniffing passwords, credit card #s, etc.)
IOT VILLAGE FEEDBACK

- Interest in Software Defined Radio and RF systems is high
- RF is intimidating!
  - Too much EE for software people
  - Too academic!
NO PHD?
NO PROBLEM!
AGENDA

1. So you want to hack RF...
2. Introduce essential RF concepts
3. Introduce RF reverse engineering workflow that applies to all systems
4. Do it live!
   1. Wireless camera flash
   2. Wireless LED strip controller
   3. HP wireless keyboard

This is what it’s all about
WHAT WE WON’T COVER

Digital Signal Processing
SO YOU WANT TO HACK WIRELESS
BARRIERS TO ENTRY

- Lower than ever before
- Commodity hardware is:
  - Really powerful
  - Increasingly cheap
- Free (beer & liberty) software is abundant!
HARDWARE TOOLS

- Dedicated Radio Chipset (Hardware Defined Radio)
  - Does 1 protocol really well
  - **Pros**: single-protocol performance, cost, simplicity, low power
  - **Cons**: lack of flexibility

- Examples:
  - Ubertooth ($200)
  - RFCat / Yardstick One ($100)
  - nRF24 dongles ($35)
  - ApiMote ($90)
HARDWARE TOOLS

- Software Defined Radio (SDR)
  - Swiss army knife for most-things RF
  - **Pros**: flexibility (can implement *any* protocol)
  - **Cons**: cost, complexity, power, performance (software and RF)
- Examples:
  - Ettus USRP ($686—>$$$$$$)
  - HackRF ($300)
  - BladeRF ($420-$650)
FREE SOFTWARE

- **SDR:**
  - GNU Radio: open source digital signal processing suite
  - GNU Radio OOT Modules: third party plugins
    - gr-lora, gr-nordic
  - Baudline, Inspectrum, Fosphor: powerful analysis tools

- **HDR:**
  - Bluez, libubertooth, Killerbee
  - Marc’s nRF24 library
TOOLS ARE RIDICULOUS
RADIO CRASH COURSE
PHY LAYER

- Lowest layer in communication stack
- In wired protocols: voltage, timing, and wiring defining 1s and 0s
- In wireless: patterns of energy being sent over RF medium
WHAT IS RF?

▸ “One of the four fundamental forces of the universe” — Tom Rondeau, DARPA Program Manager, former GNU Radio lead

▸ “Radio Frequency”

▸ Electromagnetic waves

▸ Energy
Spectrogram

a.k.a. “waterfall”
MANIPULATING RF

- Done with a radio
- Hardware defined
  - RF and protocol in silicon
- Software defined radio (SDR)
  - Flexible silicon handles RF
  - Protocol-specific components implemented in software (CPU or FPGA)
PHY COMPONENTS

- Modulation
  - How digital values are mapped to RF energy
- RF parameters that can be modulated:
  - Amplitude
  - Frequency
  - Phase
  - some combination of the above
Modulators can modulate analog or digital information.

- Digital modulation
  - Symbols: discrete RF energy state representing some quantity of information
COMMON IOT PHYS

- Frequency Shift Keying: FSK, GFSK
  - RF energy alternates between two frequencies to signify digital values

- Amplitude Shift Keying: ASK, OOK
  - Changes in RF power on a certain frequency signify digital values

Symbols

http://3.bp.blogspot.com/-w6qwKucSjII/UgWKKmPUP2I/AAAAAAAAADYAJ9NMGYzcJ4/s1600/Screenshot-2013-08-03-04-41-52.png
MORE COMPLICATED IOT PHYS

- Spread spectrum
  - Data bits are encoded at a higher rate and occupy more spectrum
  - Resilient to RF noise
- Examples:
  - 802.15.4 (top)
  - LoRa (bottom)

125, 250, or 500 kHz
Radios can have two functions:

- Transmitting
- Receiving

If a radio can do both it is dubbed a transceiver
ON REVERSE ENGINEERING

- How does one reverse engineer an arbitrary wireless system?

- Main objective: figure out how data is mapped to symbols

- Reverse engineering boils down to building receivers
METHODOLOGY

WIRELESS REVERSE ENGINEERING

METHODOLOGY
[INTERACTIVE]
LET’S FORMALIZE THIS
RF REVERSE ENGINEERING METHODOLOGY

1. Characterize the channel
1. CHANNEL CHARACTERIZATION

- Things to identify:
  1. Where on the spectrum is it? i.e. what is its Center Frequency?
  2. How wide is the channel? (kHz or MHz)
  3. Is the channel static or does it hop? If latter, what pattern/timing?
RF REVERSE ENGINEERING METHODOLOGY

1. Characterize the channel
2. Identify the modulation
2. IDENTIFY THE MODULATION

- Defines how **data** is mapped to RF energy

- This is the scariest part!

- …until you realize that most modulations are variations on a theme

- How to identify:
  1. OSINT/Documentation
  2. Intuition!
RF REVERSE ENGINEERING METHODOLOGY

1. Characterize the channel
2. Identify the modulation
3. Determine the symbol rate
3. DETERMINE SYMBOL RATE

▸ How often does the symbol state change?

▸ How to identify:
  ▸ OSINT/Documentation
  ▸ Measurement (Baudline, Inspectrum)

**Time selection**

- Enable cursors: ✔
- Symbols: 66
- Rate: 291.036Hz
- Period: 3.436ms
- Symbol rate: 19.2084kHz
- Symbol period: 52.0606μs
RF REVERSE ENGINEERING METHODOLOGY

1. Characterize the channel
2. Identify the modulation
3. Determine the symbol rate
4. Synchronize
4. SYNCHRONIZE

- Things to identify:
  1. **Preamble**: pattern that tells receivers “data to follow”, clock recovery
  2. **Start of Frame Delimiter (SFD)**: tells receiver “preamble is over, data follows from here on out”

- These are present in essentially **ALL** digital communication schemes!
RF REVERSE ENGINEERING METHODOLOGY

1. Characterize the channel
2. Identify the modulation
3. Determine the symbol rate
4. Synchronize
5. Extract symbols
5. EXTRACT SYMBOLS

- De-map symbols into data based on the expected modulation topology
- Profit! (more on this later)
RF REVERSE ENGINEERING METHODOLOGY

1. Characterize the channel
2. Identify the modulation
3. Determine the symbol rate
4. Synchronize
5. Extract symbols
LET’S SEE IT IN ACTION
BUT FIRST
A word on OPEN SOURCE INTELLIGENCE
OPEN SOURCE INTELLIGENCE (OSINT)

- Information gleaned from public sources:
  - FCC/regulatory filing documents
  - Technical documentation (datasheets, application notes)
  - Patents
  - etc.

- See Marc’s prior talks on OSINT from FCC filings
RF REVERSE ENGINEERING METHODOLOGY

0. Open-source intelligence research
1. Characterize the channel
2. Identify the modulation
3. Determine the symbol rate
4. Synchronize
5. Extract symbols
Frequency Shift Keying

CAMERA FLASH TRIGGER
Yongnuo YN560-TX Flash Controller

- Remotely triggers camera flashes
- Wireless configuration of flash settings
- Flashes can be grouped for granular control
- Let’s reverse it and see how it works!
RF REVERSE ENGINEERING METHODOLOGY

0. Open-source intelligence research
1. Characterize the channel
2. Identify the modulation
3. Determine the symbol rate
4. Synchronize
5. Extract symbols
0. OSINT

- No FCC ID silkscreen!

- Google:
  - "site:fccid.io YN560-TX"
SO YOU WANT TO HACK RADIOS // BASTILLE NETWORKS

FCC Test Report EUT Description

<table>
<thead>
<tr>
<th>Report No.: CST-TCB140624031</th>
</tr>
</thead>
<tbody>
<tr>
<td>I General Information</td>
</tr>
<tr>
<td>1.1 Description of Device (EUT)</td>
</tr>
<tr>
<td>Trade Name: YONGNUO</td>
</tr>
<tr>
<td>EUT                           : Manual Flash Controller</td>
</tr>
<tr>
<td>Model No.                     : YN560-TX</td>
</tr>
<tr>
<td>DIFF.                         : N/A</td>
</tr>
<tr>
<td>Type of Antenna               : PCB Antenna, Max. Gain: 1.5dBi</td>
</tr>
<tr>
<td>Operation Frequency           : 2402.5-2456.5MHz</td>
</tr>
<tr>
<td>Channel number                : 16</td>
</tr>
<tr>
<td>Modulation type               : FSK</td>
</tr>
</tbody>
</table>

Channel and modulation clues

Good start…
Let’s see what else we can find
FCC Test Report EUT Description

<table>
<thead>
<tr>
<th>Channel</th>
<th>Frequency</th>
<th>Channel</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH1</td>
<td>2402.5 MHz</td>
<td>CH9</td>
<td>2429.5 MHz</td>
</tr>
<tr>
<td>CH2</td>
<td>2405.5 MHz</td>
<td>CH10</td>
<td>2432.5 MHz</td>
</tr>
<tr>
<td>CH3</td>
<td>2408.5 MHz</td>
<td>CH11</td>
<td>2438.5 MHz</td>
</tr>
<tr>
<td>CH4</td>
<td>2411.5 MHz</td>
<td>CH12</td>
<td>2441.5 MHz</td>
</tr>
<tr>
<td>CH5</td>
<td>2414.5 MHz</td>
<td>CH13</td>
<td>2444.5 MHz</td>
</tr>
<tr>
<td>CH6</td>
<td>2417.5 MHz</td>
<td>CH14</td>
<td>2450.5 MHz</td>
</tr>
<tr>
<td>CH7</td>
<td>2420.5 MHz</td>
<td>CH15</td>
<td>2453.5 MHz</td>
</tr>
<tr>
<td>CH8</td>
<td>2426.5 MHz</td>
<td>CH16</td>
<td>2456.5 MHz</td>
</tr>
</tbody>
</table>

Detailed channel mapping
RF REVERSE ENGINEERING METHODOLOGY

1. Characterize the channel
2. Identify the modulation
3. Determine the symbol rate
4. Synchronize
5. Extract symbols

Open-source intelligence research

16 channels, center frequencies, MISSING: bandwidth
FSK

OSINT gets us on our way!
Bandwidth Measurement

500 kHz

Inspectrum

Baudline
RF REVERSE ENGINEERING METHODOLOGY

1. Open-source intelligence research
2. Characterize the channel
3. Identify the modulation
4. Determine the symbol rate
5. Synchronize

- 16 channels, center frequencies, 500 kHz bandwidth
- FSK
Symbol Rate Measurement

**Time selection**

- Enable cursors: **✓**
- Symbols: 56
- Rate: 8.92857kHz
- Period: 112μs
- **Symbol rate:** 500kHz
- Symbol period: 2μs

**Inspectrum Cursors**
RF REVERSE ENGINEERING METHODOLOGY

1. Open-source intelligence research
2. Characterize the channel
3. Identify the modulation
4. Determine the symbol rate
5. Synchronize
6. Extract symbols
RF REVERSE ENGINEERING METHODOLOGY

1. Open-source intelligence research
2. Characterize the channel
3. Identify the modulation
4. Determine the symbol rate
5. Synchronize
6. Extract symbols

- GNU Radio Flowgraph to produce a stream of symbols
- Python scripting to parse symbols into data
Translate Params into GNU Radio Flowgraph

1. Channel
   - UHD: USRP Source
     - Device Arguments: nu...e=4096
       - Samp Rate (Sps): 2M
       - Ch0: Center Freq (Hz): 2.457G
       - Ch0: Gain Value: 0
       - Ch0: Antenna: TX/RX
   - Note: The 'rot... FSK signal.
   - Rotator
     - Phase Increment: 1.5708
   - Polyphase Arbitrary Resampler
     - Resampling Rate: 1
     - Taps: Number of Filters: 32
     - Stop-band Attenuation: 100
   - Low Pass Filter
     - Decimation: 1
     - Gain: 1
     - Sample Rate: 2M
     - Cutoff Freq: 500k
     - Transition Width: 10k
     - Window: Hamming
     - Beta: 6.76

2. Demodulation
   - Quadrature Demod
     - Gain: 10.1859
   - Note: The 'Qua...INT sources.
   - Clock Recovery MM
     - Omega: 4
     - Gain Omega: 7.65625m
     - Mu: 500m
     - Gain Mu: 175m
     - Omega Relative Limit: 5m

3. Symbol Rate
   - File Sink
     - File: ...ramdisk/yongnu_fifo
     - Unbuffered: On
     - Append file: Append
4. Synchronization and 5. Symbol Extraction

1. Look for preamble
   a. 0b010101...

2. Look for SFD to synchronize
   a. Empirically observed to be 0b00

3. Read out frame using approximate MTU size
   a. Dump N bits

4. Parse frame
Application Layer

Trigger Flash

- 1 broadcast command fires all camera flashes

Configuration

- Updates a single flash group’s settings
  - Target group, mode (On/Manual/Multi), flash power
  - 5 commands, only the 4th carries state information
Demo Time!
Bonus Content:

Blooper Reel!
On-Off Keying / Pulse-Width Modulation

RF LED CONTROLLER
So you want to hack radios // Bastille Networks

[404 Model Not Found]

- USB LED Strip
- RF Remote Control
- Unknown Vendor
- Unknown Model Number
- No FCC ID
RF LED Controller Reverse Engineering Workflow

0. Open-source intelligence research
1. Characterize the channel
2. Identify the modulation
3. Determine the symbol rate
4. Synchronize
5. Extract symbols
RF Wireless Remote

RGB LED Controller

Instructions

1. Turn On/Standby
Press this key to turn on unit or switch to standby mode. Unit will turn on and restore to previous status at powering on moment.

2/9. Dynamic Modes Adjust
Switch to dynamic mode from static color mode, or switch between dynamic modes.

3/8. Dynamic Speed Adjust
Adjust dynamic playing speed. Press Speed+ to increase speed and press Speed- to decrease speed. Unit will switch to dynamic mode if press this key at static color mode.

4/5. Static Color Adjust
Switch to static color mode from dynamic mode, or switch between different static colors.

6/7. Brightness Adjust
Adjust static color brightness. Press Bright+ to increase brightness and press Bright- to decrease. Unit will switch to static color mode if press this key at dynamic mode.

Functions

1.  
2.  
3.  
4.  
5.  
6.  
7.  
8.  
9.  
10.
Nothing Useful
10. Demo Mode
Press this key will switch to Demo mode. In Demo mode, it will loop play 9 dynamic modes, each mode repeat 3 times.

Installing

1. Power Supply
This unit accepts DC 12V to 24V power supply. The inner pole polarity is positive and sleeve is negative. Please select proper power supply according to the LED application.

2. LED Output
This unit support common anode connection LED products. The mark “△” indicates the common connection node. The peak output current is 4A per channel, please reduce load if main unit is overheating.

CAUTION! Do not short circuit the LED output, this may lead to permanent damage!

3. Remote Control
Please pull out the insulate part before using. The RF wireless remote signal can pass through barrier, so it’s not necessary to aim at the main unit when operate. For proper receiving remote signal, do not install the main unit in closed metal parts. The remote battery is 3V CR2025 type, please only replace with same type battery.

Specification

<table>
<thead>
<tr>
<th>Dynamic mode</th>
<th>19 modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Color</td>
<td>20 colors</td>
</tr>
<tr>
<td>PWM Grade</td>
<td>256 levels</td>
</tr>
<tr>
<td>Brightness Grade</td>
<td>5 levels</td>
</tr>
<tr>
<td>Speed Grade</td>
<td>10 levels</td>
</tr>
<tr>
<td>Demo mode</td>
<td>Yes</td>
</tr>
<tr>
<td>Working Voltage</td>
<td>DC 12–24V</td>
</tr>
<tr>
<td>Output Current</td>
<td>3-way, peak 4A per channel</td>
</tr>
<tr>
<td>Remote frequency</td>
<td>433.92MHz</td>
</tr>
<tr>
<td>Remote distance</td>
<td>&gt; 30m at open area</td>
</tr>
</tbody>
</table>
10. Demo Mode
Press this key will switch to Demo mode. In Demo mode, it will loop play 9 dynamic modes, each mode repeat 3 times.

Installing

1. Power Supply
This unit accepts DC 12V to 24V power supply. The inner pole polarity is positive and sleeve is negative. Please select proper power supply according to the LED application.

2. LED Output
This unit support common anode connection LED products. The mark “A” indicates the common connection node. The peak output current is 4A per channel, please reduce load if main unit is overheating.
CAUTION! Do not short circuit the LED output, this may lead to permanent damage!

3. Remote Control
Please pull out the insulate part before using. The RF wireless remote signal can pass through barrier, so it’s not necessary to aim at the main unit when operate. For proper receiving remote signal, do not install the main unit in closed metal parts. The remote battery is 3V CR2025 type, please only replace with same type battery.

Specification

<table>
<thead>
<tr>
<th>Dynamic mode</th>
<th>19 modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Color</td>
<td>20 colors</td>
</tr>
<tr>
<td>PWM Grade</td>
<td>256 levels</td>
</tr>
<tr>
<td>Brightness Grade</td>
<td>5 levels</td>
</tr>
<tr>
<td>Speed Grade</td>
<td>10 levels</td>
</tr>
<tr>
<td>Demo mode</td>
<td>Yes</td>
</tr>
<tr>
<td>Working Voltage</td>
<td>DC 12–24V</td>
</tr>
<tr>
<td>Output Current</td>
<td>3-way peak 4A per channel</td>
</tr>
<tr>
<td>Remote frequency</td>
<td>433.92MHz</td>
</tr>
<tr>
<td>Remote distance</td>
<td>&gt;30m at open area</td>
</tr>
</tbody>
</table>

433.92 MHz Center Frequency
RF LED Controller Reverse Engineering Workflow

0. Open-source intelligence research

1. Characterize the channel
   - 433.92 MHz Center Frequency

2. Identify the modulation

3. Determine the symbol rate

4. Synchronize

5. Extract symbols
RF LED Controller Reverse Engineering Workflow

0. Open-source intelligence research
1. Characterize the channel
2. Identify the modulation
3. Determine the symbol rate
4. Synchronize
5. Extract symbols
Visually Determine the Modulation
Visually Determine the Modulation
Visually Determine the Modulation
Visually Determine the Modulation

Transmitter On

Transmitter Off
Visually Determine the Modulation

- Transmitter On
- Transmitter Off
- Transmitter On
Visually Determine the Modulation

Transmitter On
Transmitter Off
Transmitter On
Transmitter Off
Visually Determine the Modulation

On for ~450us
Off for ~1200us

On for ~1200us
Off for ~450us
Visually Determine the Modulation

On for ~450us
Off for ~1200us

On for ~1200us
Off for ~450us
Visually Determine the Modulation

On for ~450us
Off for ~1200us

On for ~1200us
Off for ~450us
Visually Determine the Modulation

Pulse Width Modulation (On-Off Keying)

On for ~450us
Off for ~1200us

On for ~1200us
Off for ~450us
RF LED Controller Reverse Engineering Workflow

0. Open-source intelligence research
1. Characterize the channel
2. Identify the modulation: Pulse Width Modulation / On-Off Keying
3. Determine the symbol rate: ~600b/s (1650us symbol duration)
4. Synchronize
5. Extract symbols

Characterize the channel
Identify the modulation
Determine the symbol rate
Synchronize
Extract symbols
RF LED Controller Reverse Engineering Workflow

0. Open-source intelligence research
1. Characterize the channel
2. Identify the modulation
3. Determine the symbol rate
4. Synchronize
5. Extract symbols
Button Waveforms in Baudline

- Sync Field (4 bits)
- Address (16 bits)
- Button ID (4 bits)
- Trailer (1 bit)
RF LED Controller Packet Format

1100001000101100111111111101

- Sync Field(??) (4 bits)
- Address(??) (16 bits)
- Button ID (4 bits)
- Trailer (1 bit)
Packet Spacing

12ms gap
What do we know?

- 433.92 MHz center frequency [channel]
- Pulse width modulation [modulation]
  - 600b/s data rate [symbol timing]
  - Bit 1 is ~1200us on and ~450us off
  - Bit 0 us ~450us on and ~1200us off
- Packets are 25 bits long [synchronize]
  - 4 sync bits (??)
  - 16 address bits (??)
  - 4 button id
  - 1 trailer
- 12ms spacing between packets
TDMA Frequency Shift Keying

HP KEYBOARD
HP CLASSIC WIRELESS DESKTOP

- 2.4GHz Wireless Keyboard/Mouse
- OEM = ACROX
- Keyboard
  - FCC ID PRDKB14
- Mouse
  - FCC ID PRDMU26
- Dongle
  - FCC ID PRDRX02
# HP DONGLE TEST REPORT

<table>
<thead>
<tr>
<th>EUT</th>
<th>2.4GHz Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODEL NO.</td>
<td>MRN</td>
</tr>
<tr>
<td>FCC ID</td>
<td>PRDRX02</td>
</tr>
<tr>
<td>POWER SUPPLY</td>
<td>5Vdc (host equipment)</td>
</tr>
<tr>
<td>MODULATION TYPE</td>
<td>GFSK</td>
</tr>
<tr>
<td>DATA RATE</td>
<td>1M bit/sec.</td>
</tr>
<tr>
<td>OPERATING FREQUENCY</td>
<td>2403MHz–2480MHz</td>
</tr>
<tr>
<td>NUMBER OF CHANNEL</td>
<td>78</td>
</tr>
<tr>
<td>ANTENNA TYPE</td>
<td>Printed antenna</td>
</tr>
<tr>
<td>DATA CABLE</td>
<td>NA</td>
</tr>
<tr>
<td>I/O PORT</td>
<td>USB</td>
</tr>
<tr>
<td>ACCESSORY DEVICES</td>
<td>NA</td>
</tr>
</tbody>
</table>
HP KEYBOARD TEST REPORT

1.1.1 Product Details

The following brands are provided to this EUT.

<table>
<thead>
<tr>
<th>Brand Name</th>
<th>Model Name</th>
<th>Product Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACROX</td>
<td>KBIM, K2BM</td>
<td>HP Wireless Keyboard K2500</td>
<td>Marketing purpose</td>
</tr>
<tr>
<td>HP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.1.2 Specification of the Equipment under Test (EUT)

<table>
<thead>
<tr>
<th>RF General Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency Range (MHz)</td>
</tr>
<tr>
<td>2400-2483.5</td>
</tr>
</tbody>
</table>
SO YOU WANT TO HACK RADIOS // BASTILLE NETWORKS

HP DONGLE DMESG OUTPUT

```
[ +0.276333] usb 1-3.1: new full-speed USB device number 21 using xhci_hcd
[ +0.091959] usb 1-3.1: New USB device found, idVendor=3938, idProduct=1032
[ +0.000012] usb 1-3.1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ +0.000008] usb 1-3.1: Product: 2.4G RF Keyboard & Mouse
[ +0.000007] usb 1-3.1: Manufacturer: MOSART Semi.
[ +0.000470] usb 1-3.1: ep 0x81 - rounding interval to 64 microframes, ep desc says 80 microframes
[ +0.002402] input: MOSART Semi. 2.4G RF Keyboard & Mouse as /devices/pci0000:00/0000:00:14.0/usb1/1-3.1/3-1/input:
[ +0.054089] hid-generic 0003:3938:1032.0009: input,hidraw2: USB HID v1.10 Keyboard [MOSART Semi, 2]
[ +0.004330] input: MOSART Semi. 2.4G RF Keyboard & Mouse as /devices/pci0000:00/0000:00:14.0/usb1/1-3.1/3-1/input:
[ +0.055401] hid-generic 0003:3938:1032.000A: input,hiddev0,hidraw3: USB HID v1.10 Mouse [MOSART Semi, 2]
```
DONGLE IN BAUDLINE

- Always transmitting at 8ms intervals
- No channel hopping
- TDMA? (Time Division Multiple Access)
KEYBOARD IN BAUDLINE

- Keystrokes follow dongle packets by 2ms
- Keyboard transmits up to every 8ms
- Dongle behavior doesn’t change
KEYBOARD DEMOD FLOWGRAPH
GREP FOR PACKETS

xxd -p demod.out |
tr -d "\n" |
grep -Po "(00|ff|aa|55)+\.{8}" |
sort |
uniq -c |
sort -nr |
Head -n 10
GREP FOR PACKETS

```
xxd -p demod.out |
tr -d "\n" |
grep -Po "(00|ff|aa|55)+.{8}" |
sort |
uniq -c |
sort -nr |
Head -n 10
```
GREP FOR PACKETS

```
xxd -p demod.out |
tr -d "\n" |
grep -Po "(00|ff|aa|55)+.{8}" |
sort |
uniq -c |
sort -nr |
Head -n 10
```
GREP FOR PACKETS

```
xxd -p demod.out |
tr -d "\n" |
grep -Po "(00|ff|aa|55)+.{8}" |
sort |
uniq -c |
sort -nr |
Head -n 10
```
DONGLE PACKET BYTES

fffffaaaaaaaaaaaaaaaaaaadeddd4e8
sed s/[dongle packets]//g
KEYBOARD PACKET BYTES

text
GREP, GREP, AND GREP SOME MORE!

aaaaaa  ddd4e8  2e  db  3f  384a
aaaaaa  ddd4e8  2d  db  37  6092
aaaaaa  ddd4e8  28  db  3f  98f8
aaaaaa  ddd4e8  25  db  3f  c9ba
aaaaaa  ddd4e8  25  db  21  3649
aaaaaa  ddd4e8  21  db  27  30f5
aaaaaa  ddd4e8  20  db  3f  3951
GREP, GREP, AND GREP SOME MORE!

aaaaaa  ddd4e8  2e  db  3f  384a
aaaaaa  ddd4e8  2d  db  37  6092
aaaaaa  ddd4e8  28  db  3f  98f8
aaaaaa  ddd4e8  25  db  3f  c9ba
aaaaaa  ddd4e8  25  db  21  3649
aaaaaa  ddd4e8  21  db  27  30f5
aaaaaa  ddd4e8  20  db  3f  3951

preamble
GREP, GREP, AND GREP SOME MORE!

```
aaaaaaa ddd4e8 2e db 3f 384a
aaaaaaa ddd4e8 2d db 3f 98f8
aaaaaaa ddd4e8 25 db 3f c9ba
```

preamble address
GREP, GREP, AND GREP SOME MORE!

aaaaaa ddd4e8 2e db 3f 384a
aaaaaa ddd4e8 2d db 37 6092
aaaaaa ddd4e8 28 db 3f 98f8
aaaaaa ddd4e8 25 db 3f c9ba
aaaaaa ddd4e8 25 db 21 3649
aaaaaa ddd4e8 21 db 27 30f5
aaaaaa ddd4e8 20 db 3f 3951

preamble address sequence
GREP, GREP, AND GREP SOME MORE!

```
aaaaaa ddd4e8 2e db 3f 384a
aaaaaa ddd4e8 2d db 37 6092
aaaaaa ddd4e8 28 db 3f 98f8
aaaaaa ddd4e8 25 db 3f c9ba
aaaaaa ddd4e8 25 db 21 3649
aaaaaa ddd4e8 21 db 27 30f5
aaaaaa ddd4e8 20 db 3f 3951
```

- preamble
- address
- sequence
- frame type
SO YOU WANT TO HACK RADIOS // BASTILLE NETWORKS

GREP, GREP, AND GREP SOME MORE!

```
aaaaaa  ddd4e8  2e  db  3f  384a
aaaaaa  ddd4e8  2d  db  37  6092
aaaaaa  ddd4e8  28  db  3f  98f8
aaaaaa  ddd4e8  25  db  3f  c9ba
aaaaaa  ddd4e8  25  db  21  3649
aaaaaa  ddd4e8  21  db  27  30f5
aaaaaa  ddd4e8  20  db  3f  3951
```

preamble  address  sequence  frame type  keystroke
GREP, GREP, AND GREP SOME MORE!

AAAAAA  ddd4e8  2e  db  3f  384a
AAAAAA  ddd4e8  2d  db  37  6092
AAAAAA  ddd4e8  28  db  3f  98f8
AAAAAA  ddd4e8  25  db  3f  c9ba
AAAAAA  ddd4e8  25  db  21  3649
AAAAAA  ddd4e8  21  db  27  30f5
AAAAAA  ddd4e8  20  db  3f  3951

preamble  address  sequence  frame type  keystroke  crc16
tl;dr
smarter people than me made that easy
Common Threads

Methodology Revisited
Reverse Engineering Methodology

0. Open-source intelligence research
1. Characterize the channel
2. Identify the modulation
3. Determine the symbol rate
4. Synchronize
5. Extract symbols
1. Channel Characterization

All 3 PHYs share a common notion of a channel

<table>
<thead>
<tr>
<th>Camera Flash Controller</th>
<th>RF LED Strip</th>
<th>Keyboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>+/- 250 kHz in 2.4 GHz</td>
<td>433.92 MHz</td>
<td>2416 MHz</td>
</tr>
</tbody>
</table>
2. Identify Modulation

Modulation is the biggest variable
(but OSINT makes identifying it easy)

<table>
<thead>
<tr>
<th>Camera Flash Controller</th>
<th>RF LED Strip</th>
<th>Keyboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency Shift Keying</td>
<td>Pulse-Width Modulation / On-Off Keying</td>
<td>TDMA Frequency Shift Keying</td>
</tr>
</tbody>
</table>
3. Symbol Rate Recovery

All 3 PHYs share a common notion of discrete symbol timing

<table>
<thead>
<tr>
<th>Camera Flash Controller</th>
<th>RF LED Strip</th>
<th>Keyboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>500,000 symbols/s</td>
<td>600 symbols/s</td>
<td>1,000,000 symbols/s</td>
</tr>
</tbody>
</table>
4. Synchronization

All 3 PHYs contain *synchronization features*
(preamble and/or Start of Frame delimiter)

<table>
<thead>
<tr>
<th>Camera Flash Controller</th>
<th>RF LED Strip</th>
<th>Keyboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0b01010101...00)</td>
<td>Sync Field (assumed)</td>
<td>Preamble (0xaa..aa)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SFD (3 byte address)</td>
</tr>
</tbody>
</table>
5. Symbol Extraction

Once you get here it’s just bits on a disk
Reverse Engineering Methodology

0. Open-source intelligence research
1. Characterize the channel
2. Identify the modulation
3. Determine the symbol rate
4. Synchronize
5. Extract symbols

Same process for 3 different PHYs!
Conclusions

Disparate wireless systems can be rationalized via process.

OSINT will help you skip the complex/domain-specific radio parts.

Once you demodulate, you have bits on a disk which you can handle any way you please.

One last thought to leave you with...
The IoT is full of holes...
It’s up to you to find them!

marc@Bastille.net
@marcnewlin

matt@Bastille.net
@embeddedsec
Thanks!

marc@Bastille.net
@marcnewlin

matt@Bastille.net
@embeddedsec
Questions?