Universal Radio Hacker
A Suite for Analyzing and Attacking Stateful Wireless Protocols

Johannes Pohl and Andreas Noack

University of Applied Sciences Stralsund

August 13, 2018
Proprietary wireless protocols everywhere

Smart Home

- Increase comfort through wireless sockets, door locks, valve sensors...
- Devices are designed under size and energy constraints
- Less resources for cryptography

Risks of Smart Home

- Manufactures design custom proprietary wireless protocols
- Hackers may take over households and e.g. break in without physical traces

How can we eavesdrop and manipulate the wireless communication between such devices to assess the security?

August 13, 2018 Johannes Pohl and Andreas Noack Universal Radio Hacker
Software Defined Radio

Why Software Defined Radios?

- Send and receive on nearly arbitrary frequencies
- Flexibility and extendability with *custom software*

\[\text{e.g. HackRF: 1 MHz - 6 GHz}\]
Software Defined Radios

Software Defined Radios are affordable

Last Checked: July 21, 2018

NooElec HackRF One Software Defined Radio (SDR)
by NooElec

$317.95
Get it by Wednesday, Jul 25
FREE Shipping on eligible orders

NooElec NESDR SMArt - Premium RTL-SDR Software Defined Radio
by NooElec

$23.95
Get it by Wednesday, Jul 25
FREE Shipping on eligible orders
Universal Radio Hacker

- HackRF One
- URH
- Universal Radio Hacker

Software Defined Radios

- Interpretation
- Analysis

Stateless

Format

Simulation

Stateful

Format

August 13, 2018 Johannes Pohl and Andreas Noack
Universal Radio Hacker

- **Software Defined Radios**

- **Universal Radio Hacker** Slide 5

- **Attacking a Door Lock**

- **Interpretation**
- **Analysis**
  - **Stateless**
    - **Format**
      - **Generation**
  - **101010**

- **August 13, 2018 Johannes Pohl and Andreas Noack**
Universal Radio Hacker

- **Introduction**
- **Attacking a Door Lock**
- **Conclusion**

**Software Defined Radios**

**Universal Radio Hacker Slide 5**

- **Interpretation**
- **Analysis**
  - 101010
  - **Stateless**
  - **Stateful**
- **Generation**
- **Simulation**

August 13, 2018
Johannes Pohl and Andreas Noack
Universal Radio Hacker
Setup

CCU  door lock  remote control
Overview

Pairing

- central (CCU)
- door lock
- remote control

AES-Key

OPEN Command

Challenge

Response

AES-Key

ACK

wireless socket

AES-Key
Overview

- **Pairing**
  - central (CCU)
  - door lock
  - remote control

- **AES-Key**
  - from central (CCU) to door lock
  - from remote control to door lock

- **OPEN Command**

---

August 13, 2018

Johannes Pohl and Andreas Noack

Universal Radio Hacker
Overview

- **Central (CCU)**
- **Door Lock**
- **Remote Control**

**Pairing**

**AES-Key**

**OPEN Command**

**Challenge**
Overview

- **Pairing**
  - central (CCU)
  - door lock
  - remote control

- **OPEN Command**

- **Challenge**

- **Response_{AES-Key}(Challenge)**
Overview

- **Pairing**
  - central (CCU)
  - door lock
  - remote control

- **OPEN Command**
  - Challenge
  - $\text{Response}_{\text{AES-Key}}(\text{Challenge})$
  - ACK

**Protocol**

- **Overview**
  - AES-Key
  - OPEN Command
  - Challenge
  - $\text{Response}_{\text{AES-Key}}(\text{Challenge})$
  - ACK
Overview

- **Central (CCU)**
  - **AES-Key**
- **Door Lock**
  - **AES-Key**
- **Remote Control**
  - **AES-Key**

**Pairing**

**OPEN Command**

1. **Challenge**
2. **Response_{AES-Key}(Challenge)**
3. **ACK**
Record and demodulate signal

Capture of door lock open communication

Zoom into start of second message
Demodulation and Signal Editing with URH

Further Interpretation Features

- **Synchronized selection** between demodulated and raw signal
- **Signal Editor** i.e. copy, paste, crop, mute signal selections
- Configurable moving average and bandpass filters
Analysis phase

In Analysis phase we reverse engineer the protocol format.

**Example format**

- Preamble
- Synchronization
- Length
- Source Address
- Destination Address
- Data
- Checksum

This includes

- **Decode** messages
- **Labeling** of protocol fields
- **Group** messages by assigning message types
What kind of decoding does the door lock use?

All messages are encoded in the following way:

1. Pseudo encryption
2. Data Whitening
3. (Modulation)
Pseudo Encryption

Code

```c
enc[0] = msg[0];
enc[1] = ~(msg[1]) ^ 0x89;
for (i = 2; i < NUM_BYTES; i++)
    enc[i] = (enc[i-1]+0xdc) ^ msg[i];
```

Use

- Does not increase the security
- Assumption: Obscure method for pseudo security
To increase transmission quality a data whitening is used.

XOR with each 8 LSB of a pseudo-random sequence generated by an LFSR represented by the polynomial $x^9 + x^5 + x^0$.

Initial state is 111111111

First eight states of the LFSR:

111111111 $\Rightarrow$ 011111111 $\Rightarrow$ 001111111 $\Rightarrow$ 000111111 $\Rightarrow$
000011111 $\Rightarrow$ 100001111 $\Rightarrow$ 110000111 $\Rightarrow$ 111000011
Decodings with URH

MyDecoding

Base Functions
- Edge Trigger
- Morse Code
- Substitution
- External Program

Additional Functions

Invert
- Differential Encoding
- Change Bitorder
- Remove Redundancy
- Remove Carrier
- Remove Data Whitening (CC1101)
- Wireless Short Packet (WSP)
- Cut before/after

Decoder

Signal

Invert
- Change Bitorder
- Invert #2

Information and Options

## DECODING PROCESS ##

Invert:
All bits are inverted, i.e. 0->1 and 1->0.

Decoded Bits

Signal (0,1):
- Test: 10010110

Decoded Bits:
- 01101001

[Decoding Errors = 0]
Result in URH after decoding and labeling

**Other Options:** Select all, Filter, Align

Assign manually or rule based

Check against configurable CRC
In Simulation phase we can work on the logical layer. URH takes care of **Modulation** and **Encoding** during simulation time.

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. [Trigger command]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1. IF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>item1.rc &gt; 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1.1. [Goto: item6]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. preamble synchronization length sequence number control type source address destination address cipher crc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. [Trigger command]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.1. IF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>item4.rc == 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.1.1. [Goto: item3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. preamble synchronization length sequence number control type source address destination address command checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. preamble synchronization length sequence number control type source address destination address command challenge magic checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. preamble synchronization length sequence number control type source address destination address command checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. preamble synchronization length sequence number control type source address destination address command authentication checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Demonstration Video
Summary and future work

Summary

- **Software Defined Radios** offer a high flexibility when investigating radio protocols
- Tools like **Universal Radio Hacker** abstract the required HF basics and enable analyzing such protocols without having to be a hardware expert
- Smart Home manufacturers have to react, Security by Obscurity is no longer an option

Ongoing work

- Rule based intelligence for automatic analysis phase
- Enhance accuracy of detecting interpretation parameters
- Support for more complex modulations e.g. 4-PSK
Contact

- E-Mail: Johannes.Pohl90@gmail.com
- E-Mail: Andreas.Noack@hochschule-stralsund.de
- GitHub: https://github.com/jopohl