Software Defined Radio Leaks

Kerryn Eer
E0014864
National University of Singapore
kerryn_eer@u.nus.edu

Ong Ai Hui
E0202723
National University of Singapore
aihui@u.nus.edu

Tan Wenjian
E0191462
National University of Singapore
e0191462@u.nus.edu

Ng Wei Xin
E0177126
National University of Singapore
e0177126@u.nus.edu

ABSTRACT

In this paper, we will be exploring information leakage via wireless transmission from wireless devices people use in their daily lives. For this project, we used a store-bought wireless keyboard. With the right Software Defined Radio (SDR) tools, we aim to sniff these wireless transmissions to make sense of them and obtain valuable information. As a result of this study, we can conclude if the keyboard's usage poses a threat to the user's information privacy.

General Terms

Privacy, Security, Confidentiality, Integrity

Keywords

HackRF One, GNU Radio, Digital Signal Processing, Side-channel attack, Signal-to-noise ratio, RF Technology, GFSK, Software Defined Radio

1. INTRODUCTION

With the advent of technology, many have made the switch from the traditional wired keyboard and mouse to wireless alternatives. A wireless keyboard has many conveniences, taking away the trouble of cumbersome wires and giving users the luxury typing comfortably whilst seated further away [1]. We conducted a survey with ten students from National University of Singapore to examine their usage of wireless keyboards. One student said, "I use my laptop in clamshell mode at home, along with an external monitor. It is very convenient to use a wireless keyboard and mouse with my set up".

The increasing popularity of wireless keyboards has led to concerns over its security. With the slew of information pertaining to the design and implementation of commercial keyboards online, they are extremely susceptible to side-channel-attacks. Regardless of the robustness of a system, if the wireless keyboards are compromised, personal information, passwords and credit card numbers can be sniffed, causing a breach in the user's data confidentiality. Furthermore, the wireless nature of these keyboards makes them extremely attractive to replay attacks, where RF signals from the keyboard are captured and retransmitted to its host device, compromising the integrity of packets being received by the computer. Hence, it is of upmost importance for these wireless keyboards to be secure, lest an attacker sniffs and/or tampers with the signals.

In this report, we will give an overview of the technology behind wireless communication (Section 2), followed by the specifications of the keyboards and our selected tools (Section 3). Then, we will dive into the implementation details of the sniffing attack (Section 4) and go through our findings (Section 5). Finally, we will touch on possible defense strategies (Section 6) and the challenges we faced over the course of this project (Section 7).

2. BACKGROUND

2.1 Wireless Keyboards

Most wireless keyboards communicate with their host devices via signals in the **I**ndustrial, **S**cientific, and **M**edical (ISM) bandwidth of 2.4GHz.

To form a connection over Radio Frequency (RF) technology, a pair of transceivers are required – one transmitter, one receiver. The transmitter can be found in the keyboard, while the receiver is connected to the host device. [2] This eliminates the need for a traditional physical connection.

2.2 Radio Frequency (RF) Wireless Technology

2.2.1 Basic Overview

The transmitter initiates the RF communication. The packet to be sent is encoded into the signal via a modulation technique. This signal is transmitted over the air to the receiver.

The receiver receives the signal from the transmitter and demodulates it to retrieve packet. The packet received may then be processed by other components. [3]

The quality of a signal is measured by the signal-to-noise ratio (SNR). A low SNR indicates a low-quality signal, thus poor communication, as the data transmitted may have been corrupted by the high noise levels.

The power level required for an RF signal to be successfully received by a receiver is known as the receive sensitivity. The lower the power level required, the higher the receive sensitivity.

2.2.2 Modulation and Demodulation

Modulation is the process of multiplying a signal of low frequency by a higher frequency (also known as the carrier). Its purpose is to enable the transmission of multiple data streams over the same channel without interference.

Conversely, demodulation is the process of converting the modulated signal to the original raw signal of low frequency. It does so by multiplying the modulated signal with the carrier. More processing of this baseband signal occurs by passing it through a low pass filter and amplifier for cleaning up and magnifying it. [4]

There are various modulation schemes and the one explored in this project is GFSK (Gaussian Frequency Shift Keying). GFSK is a frequency modulation (FM) scheme where digital information is transmitted through discrete frequency changes of a carrier signal with a Gaussian filter applied. [5]

3. TOOLS

3.1 HackRF One

3.1.1 Introduction

HackRF One is an open source Software Defined Radio (SDR) device from Great Scotts Gadgets. It is capable of transmitting and receiving radio signals from 1MHz to 6GHz at up to 20 million samples per second and has support from third party software. However, it has a maximum bandwidth of 20MHz, limiting the number of channels it can receive signals from at a given frequency, with respect to the transmitter's channel bandwidth. [6]

Figure 1: HackRF One from Great Scott Gadgets [28]

HackRF One, being an SDR, is essentially a radio communication device that makes heavy use of software instead of hardware systems, such as amplifiers and modulators, to process radio waveforms. This presents different functionalities and gives users a lot of flexibility as changes in the specifications can be made easier as compared to earlier types of radio.

3.1.2 Role in this project

In this project, we used HackRF One to sniff transmission data between the wireless keyboard and its receiver. By making use of its Analog to Digital Converter, we can convert the analog signals being transmitted into a series of bits as a digital representation of the signal. This digital representation can then be processed through a third-party software, such as GNU Radio, to obtain useful information.

3.2 Logitech K270 Wireless Keyboard

3.2.1 Introduction

Logitech K270 Wireless keyboard uses Logitech's proprietary Advanced 2.4GHz Technology for transmission between the keyboard and its receiver. The keyboard's receiver is the Logitech Unifying Receiver which came paired with the purchase of the keyboard.

Figure 2: Logitech Unifying Receiver [29]

The proprietary wireless communication has the following specifications (as of March 2, 2009):

	Logitech Advanced 2.4 GHz Technology
Radio frequency band	2.4 GHz ISM
Number of radio channels	24
Encryption scheme	128-bit AES symmetric encryption • Limited to the wireless link between the keyboard and receiver • Encryption keys are not transmitted over the air but are stored in a non-volatile memory area of the keyboard and the receiver • Keys are hardware encrypted hence, inaccessible to software
Frequency agility protocol	• Stays on the same channel as long as possible. • Changes channel only when the current channel becomes unusable.
Network topology	Star topology network • Receiver is the central point but not the master • Each device is a master and can transmit data at any time

Table 1: Logitech Advanced 2.4 GHz Technology Specifications [7]

3.2.2 Common Vulnerabilities and Exposures

Several Common Vulnerabilities and Exposures (CVEs) on Logitech's proprietary wireless communication, involving Unifying Receiver-compatible devices, were found over the years.

CVE Identifier	Description

CVE-2016-10761 [8]	Mousejacking attack - keystroke injection on Logitech unifying devices, bypassing encryption.
CVE-2019-13052 [9]	Live decryption if a connection between a Logitech Unifying device (keyboard) and receiver is sniffed.
CVE-2019-13054 [10]	Determination of the AES key used between the receiver and Logitech R500 presentation clicker, leading to keystroke injection.
CVE-2019-13055 [11]	Dumping of AES keys and addresses on certain Logitech Unifying devices enable live decryption of Radio Frequency transmissions, as demonstrated on a Logitech K360 keyboard.

Table 2: Past CVEs related to Logitech unifying compatible devices

3.2.3 *Role in this project*

In this project, we will attempt to sniff the transmission between this keyboard and its receiver and make sense of the transmitted data. We will capture the signal then filter and demodulate it to retrieve the transmitted data.

3.3 GQRX

3.3.1 *Introduction*

GQRX is an opensource software defined radio (SDR) receiver powered by GNU Radio and Qt graphical toolkit. It supports various SDR hardware like HackRF One. [12] It provides a myriad of functionalities; the most relevant ones are listed below:

- Discovering devices attached to the computer
- Processing I/Q data
- Changing frequency, gain and applying corrections
- FFT plot and waterfall
- Spectrum analyzer mode where all signal processing is disabled

3.3.2 *Role in this project*

GQRX is used when searching for the channel the keyboard and receiver are communicating over. With the found channel, we can proceed with other software for signal analysis. GQRX also helps verify the if the SDR device (HackRF One) is in proper working condition.

3.4 GNU Radio and GNU Radio Companion

3.4.1 *Introduction*

GNU Radio is a free and open-source software development toolkit used for signal processing and analysing on digital inputs for software radio implementations. [13]. The most common usage for GNU Radio would be recording RF signals and spectrum analysis. It is a framework meant for writing signal processing applications, wrapped in reusable blocks.

GNU Radio Companion is a graphical user interface (GUI) software for users to construct a flowgraph with, where nodes of the graph are called blocks, and edges represent a stream-like data flow. [14]

3.4.2 *How does the software work?*

GNU Radio Companion allows users to implement flowgraphs for signal processing. The flowgraphs can be used to capture, transmit, receive, replay, and even demodulate signals for analysis. In-built features include waveform generators, modulators, filters and Fourier analysis implementations that are peer-reviewed and optimized. There are GUI sinks such as frequency, constellation, time and waterfall sink available to help users better make sense of the signals that are captured.

Users without prior programming experience can use this application as there are prebuilt blocks, where only parameters have to be filled in. Once the flowgraph is implemented, actual python code can be generated through the software for interpretation and usage. Otherwise, for experienced users, blocks can be self-written to fit one's needs.

3.4.3 *Role in this project*

GNU Radio Companion was utilized to identify, isolate, filter, demodulate, and synchronize the signal source. Our flowgraph will generate a byte stream from the transmission signals captured, which then can be further processed for packet extraction.

4. IMPLEMENTATION

Sniffing data from the transmission between the wireless keyboard and its receiver requires a few steps. First, the channel used for transmission has to be identified. The signals captured from the channel are then filtered, demodulated and synchronized with the transmitter's data rate. These steps will result in a stream of binary data, which can be interpreted to find out what packets are being transmitted between the wireless keyboard and the receiver.

The setup for this project is as shown below:

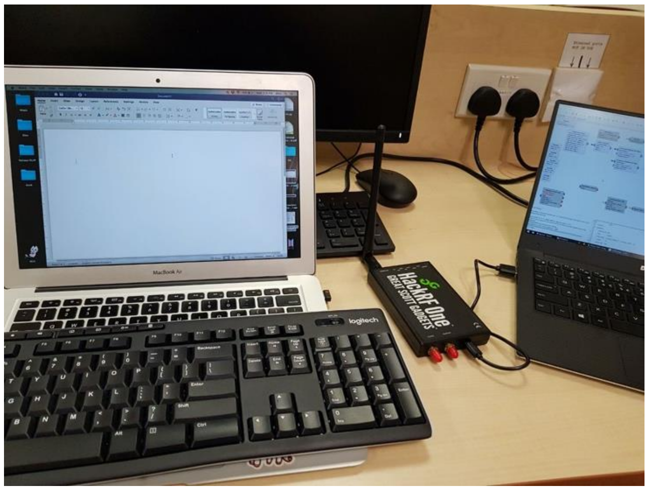


Figure 3: Setup of the demonstration of the attack

HackRF One, with an antenna attached, is connected to the attacker's computer (on the right). The victim's computer (on the left) is connected to the wireless keyboard using Logitech's unifying receiver.

4.1 Reading the Documentations

The simplest way to obtain information about the signal of interest is to read the fine prints. Every wireless device sold in the United States must be registered under the Federal Communications Commission (FCC). The Logitech Unifying Receiver that comes with the Logitech keyboard K270 has an FCC identification number (FCCID) of *JNZCU0007*. We referred to the public test report from FCC to obtain clues and information about its implementation and workings.

For the keyboard, we dismantled it and found out that a nRF24LE1H chip from Nordic Semiconductor was used for transmission from the keyboard to receiver. The chip, detailed in Section 4.1.1, falls under the nRF24LE1 category. We found a product specification datasheet for nRF24LE1 chips, revealing various important information which will help us in the later sections. [15]

4.1.1 Nordic Semiconductor nRF24LE1H chip

nRF24LE1 is a type of nRF24 intelligent 2.4 GHz RF Transceivers with embedded microcontrollers. It ensures tighter protocol timing, security and low power use with enhanced co-existence performance. [15]

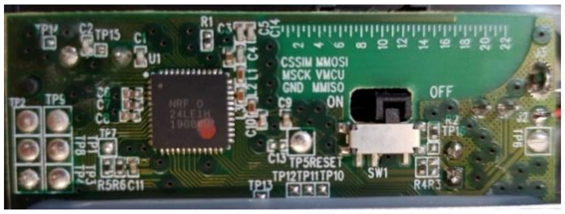


Figure 4: Internals of K270 Wireless Keyboard after dismantling

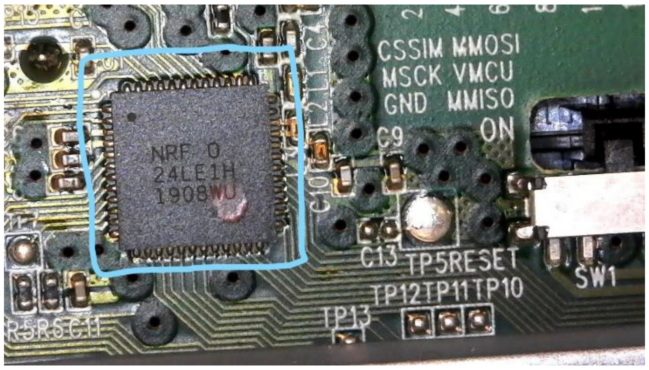
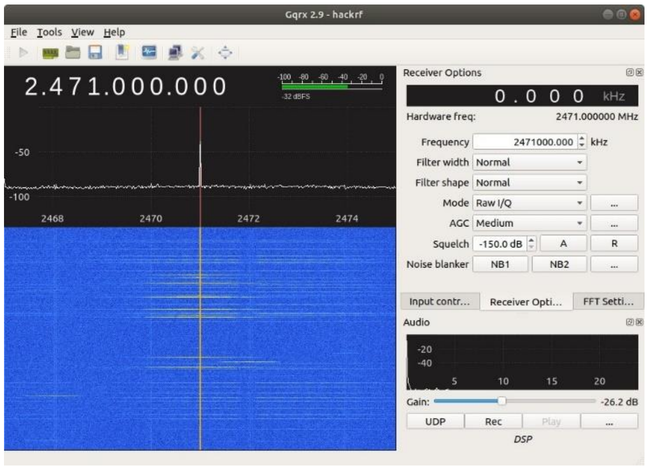


Figure 5: nRF24LE1H chip

4.2 Determining Channel Frequency and Bandwidth

From the FCC report datasheet, the operation range of the device was between 2.405 and 2.474 GHz, with 24 channels, each 2 MHz wide [16]. We verified this information with the help of GQRX. An example of a signal captured at 2.471GHz with a 2MHz bandwidth using GQRX is seen in Figure 6 below.



0.000 kHz
Hardware freq: 2471.000000 MHz
Frequency 2471000.000 kHz
Filter width Normal
Filter shape Normal
Mode Raw I/Q
AGC Medium
Squelch -150.0 dB
Noise blanker NB1 NB2
Gain: -26.2 dB

Figure 6: 2MHz bandwidth signal at 2.471GHz

It was found that the signal could be located on any of the 24 channels listed in Table 3 below. The signal hops a channel on every pairing, and when the channel is unusable due to interference (noise) [7].

Channel	**Freq. (MHz)**	**Channel**	**Freq. (MHz)**	**Channel**	**Freq. (MHz)**	**Channel**	**Freq. (MHz)**
1	2405	7	2423	13	2441	19	2459
2	2408	8	2426	14	2444	20	2462
3	2411	9	2429	15	2447	21	2465
4	2414	10	2432	16	2450	22	2468
5	2417	11	2435	17	2453	23	2471
6	2420	12	2438	18	2456	24	2474

Table 3: Channels and their respective frequencies [16]

4.3 Determining Modulation Scheme

In the FCC report referenced in Section 4.1, the modulation scheme used is GFSK (Gaussian Frequency Shift Keying). Furthermore, based on its product specifications, the nRF24LE1H chip discovered has a GFSK transceiver. The frequency found in the product specification was found to deviate ±320 kHz from what was stated in the report [15].

4.4 Determining Baud Rate

From the nRF24LE1 product specification, there are three possible data rate, 250 kbps, 1 Mbps, and 2 Mbps. [15] In addition, it was found that the packets being sent has either a *01010101* or *10101010* 8-bit preamble, depending on the which bit the address started with. With all the information, including those from earlier sections, and preliminary demodulation, the baud rate was determined to be 2 Mbps. From Figure 7 below, the time required to send the 8-bit preamble was about 4 µs, which is equal to a 2 Mbps data rate.

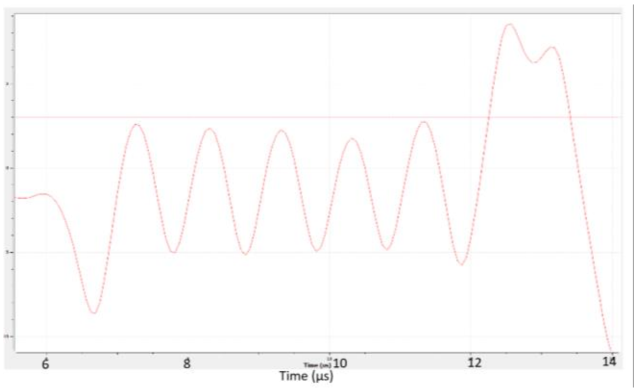


Figure 7: Evidence of 2Mbps data rate

4.5 Implementing the Flowgraph in GNU Radio Companion

With all the information gathered from the previous sections, the flowgraph for retrieving the raw data being transmitted between the keyboard and USB dongle in GNU Radio Companion can now be implemented (see Figure 9). Variables used for the flowgraph include:

- `samp_rate` = 6M
- `freq` = 2.474GHz (current channel identified using GQRX)
- `channel_width` = 2M
- `freq_offset` = 1.2M
- `fsk_deviation_hz` = 320k

The flowgraph starts with an `Osmocom Source` that reads in quadrature (I/Q) data from the HackRF One. The channel frequency is offset by 1.2 MHz to avoid the DC spike signal from the HackRF One. [17] The IQ data stream is then streamed to a `FXFF` (Frequency Xlating Finite impulse response Filter) to isolate the signal. This `FXFF` performs a frequency translation on the signal to center the original frequency of interest from the previous offset. Also, by using a low pass filter tap with a cutoff frequency of `channel_width/2` and a transition width of `channel_width/2*0.4`, the `FXFF` can filter out signals outside the cutoff frequency of ±1 MHz, since the signal bandwidth had been identified as 2 MHz. The data stream is then passed through a `Power Squelch` to remove any noise when the amplitude is too low, to help with data processing at the end. [18]

Once the IQ data has been isolated and filtered, the stream is then passed down to the `Quadrature Demod`. The `Quadrature Demod` block can be used to demodulate FM, FSK, GMSK, etc. The `Quadrature Demod` calculates the phase difference between 2 consecutive samples, which is the same as calculating the change in frequency between the 2 consecutive samples. The output stream of the `Quadrature Demod` is then streamed to a `Clock Recovery MM`, a `Binary Slicer`, and finally to a `File Sink`. The combination of the `Clock Recovery MM` and the `Binary Slicer` detects consistent high and low transitions, which then outputs the highs as a 0x01 byte and the lows as a 0x00 byte. The `Omega` parameter in the `Clock Recovery MM` is set to `samp_rate/data_rate`, which is the number of samples per symbol. The byte stream is then fed into a `File Sink`, to a FIFO (First In First Out) pipe file for live analysis. Analysis of the byte stream and framing the data to extract the packets can be done in GNU Radio Companion as well. However, it was a challenge for us, and so we decided to use the more traditional method of writing a script to process the data. [18][19][21]

The `Complex to Mag^2`, `Multiply Const`, `Add Const` and `QT GUI Time Sink` blocks were used during preliminary demodulation to help with visual analysis. This can be seen in Figure 8 below where channel 0 (blue data) are the `Mag`^{2} values and channel 1 (red data) are the float values after `Quadrature Demod`. By triggering on channel 0 when they are above a certain amplitude, visual analysis of channel 1 in the time sink can be performed while the signal is strong, which is indicative of when packets are being transmitted. [27]

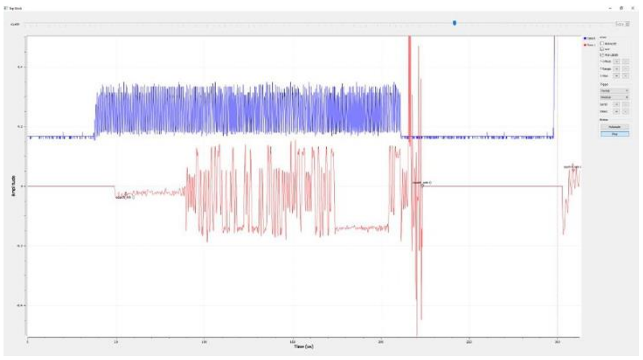
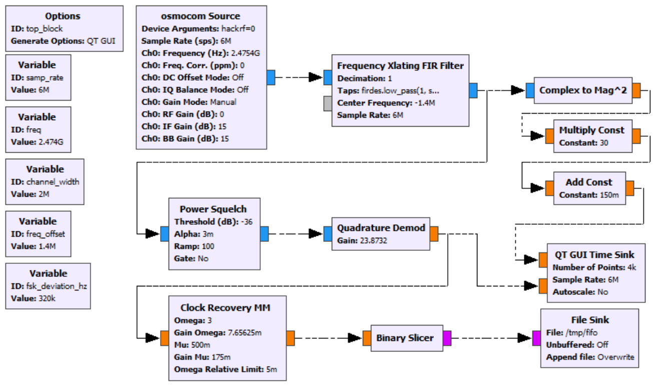


Figure 8: Visual analysis of preliminary demodulation



Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 6M

Variable
ID: freq
Value: 2.474G

Variable
ID: channel_width
Value: 2M

Variable
ID: freq_offset
Value: 1.4M

Variable
ID: fsk_deviation_hz
Value: 320k

osmocom Source
Device Arguments: hackrf=0
Sample Rate (sps): 6M
Ch0: Frequency (Hz): 2.4754G
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Off
Ch0: IQ Balance Mode: Off
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 15
Ch0: BB Gain (dB): 15

Frequency Xlating FIR Filter
Decimation: 1
Taps: firdes.low_pass(1, s...
Center Frequency: -1.4M
Sample Rate: 6M

Complex to Mag^2

Multiply Const
Constant: 30

Add Const
Constant: 150m

Power Squelch
Threshold (dB): -36
Alpha: 3m
Ramp: 100
Gate: No

Quadrature Demod
Gain: 23.8732

QT GUI Time Sink
Number of Points: 4k
Sample Rate: 6M
Autoscale: No

Clock Recovery MM
Omega: 3
Gain Omega: 7.65625m
Mu: 500m
Gain Mu: 175m
Omega Relative Limit: 5m

Binary Slicer

File Sink
File: /tmp/fifo
Unbuffered: Off
Append file: Overwrite

Figure 9: GNU Radio Companion flowgraph

4.6 Packet Recovery

In order to frame and packetize the data, the structure of the packet must be known. From the nRF24LE1 product specification datasheet, our packets appear to have an Enhanced ShockBurst (ESB) packet format. The format of the ESB packet is shown in Figure 10 below. The Packet Control Field (PCF) consists of a 6-

bit payload length, 2-bit packet ID (PID) and a 1-bit NO_ACK flag. [15]

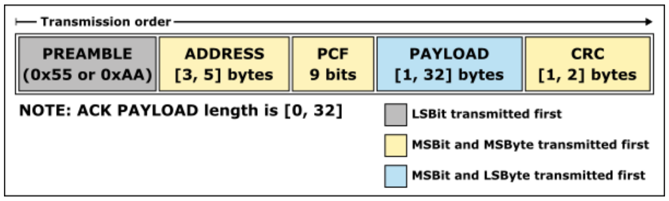
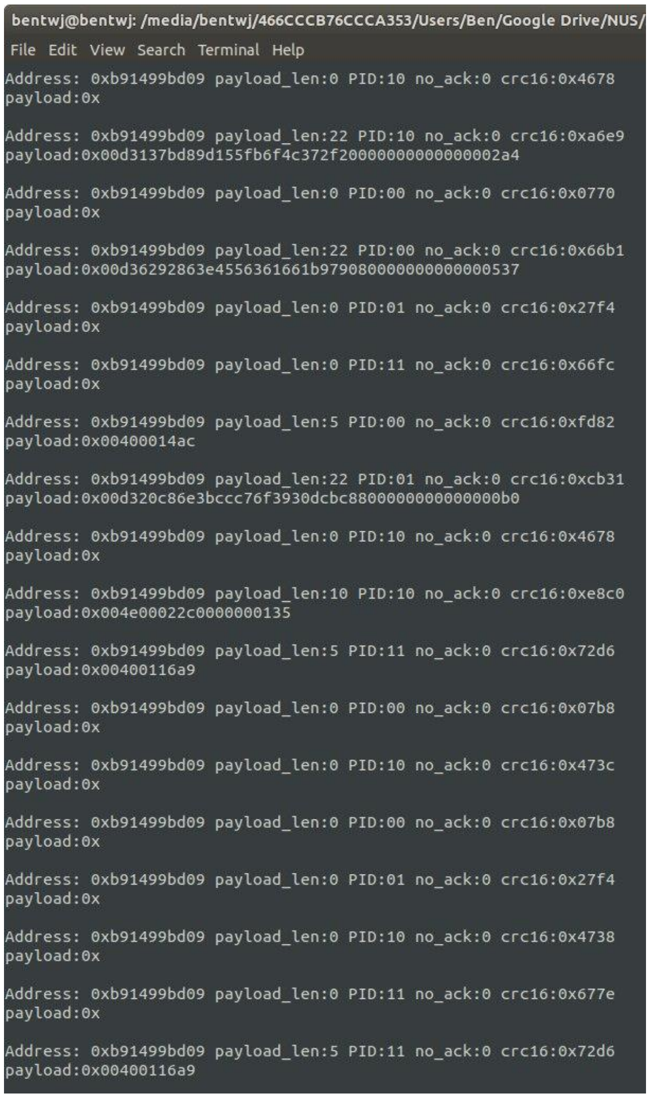


Figure 10: Format of ESB packet [20]

Using the ESB packet format as the first step, our script reads in the byte stream from the FIFO file and looks for the starting preamble of *01010101* or *10101010*. Once a preamble has been found, the address, PCF, payload, and Cyclic Redundancy Check (CRC) are read and stored in a packet format, which are printed out. An example run is shown in figure 11 below.


bentwj@bentwj: /media/bentwj/466CCCB76CCCA353/Users/Ben/Google Drive/NUS/
File Edit View Search Terminal Help

```
Address: 0xb91499bd09 payload_len:0 PID:10 no_ack:0 crc16:0x4678
payload:0x

Address: 0xb91499bd09 payload_len:22 PID:10 no_ack:0 crc16:0xa6e9
payload:0x00d3137bd89d155fb6f4c372f20000000000000002a4

Address: 0xb91499bd09 payload_len:0 PID:00 no_ack:0 crc16:0x0770
payload:0x

Address: 0xb91499bd09 payload_len:22 PID:00 no_ack:0 crc16:0x66b1
payload:0x00d36292863e4556361661b979080000000000000537

Address: 0xb91499bd09 payload_len:0 PID:01 no_ack:0 crc16:0x27f4
payload:0x

Address: 0xb91499bd09 payload_len:0 PID:11 no_ack:0 crc16:0x66fc
payload:0x

Address: 0xb91499bd09 payload_len:5 PID:00 no_ack:0 crc16:0xfd82
payload:0x00400014ac

Address: 0xb91499bd09 payload_len:22 PID:01 no_ack:0 crc16:0xcb31
payload:0x00d320c86e3bccc76f3930dcbc8800000000000000b0

Address: 0xb91499bd09 payload_len:0 PID:10 no_ack:0 crc16:0x4678
payload:0x

Address: 0xb91499bd09 payload_len:10 PID:10 no_ack:0 crc16:0xe8c0
payload:0x004e00022c0000000135

Address: 0xb91499bd09 payload_len:5 PID:11 no_ack:0 crc16:0x72d6
payload:0x00400116a9

Address: 0xb91499bd09 payload_len:0 PID:00 no_ack:0 crc16:0x07b8
payload:0x

Address: 0xb91499bd09 payload_len:0 PID:10 no_ack:0 crc16:0x473c
payload:0x

Address: 0xb91499bd09 payload_len:0 PID:00 no_ack:0 crc16:0x07b8
payload:0x

Address: 0xb91499bd09 payload_len:0 PID:01 no_ack:0 crc16:0x27f4
payload:0x

Address: 0xb91499bd09 payload_len:0 PID:10 no_ack:0 crc16:0x4738
payload:0x

Address: 0xb91499bd09 payload_len:0 PID:11 no_ack:0 crc16:0x677e
payload:0x

Address: 0xb91499bd09 payload_len:5 PID:11 no_ack:0 crc16:0x72d6
payload:0x00400116a9
```

Figure 11: Packets sent by the keyboard from keypresses of 'q'

5. FINDINGS

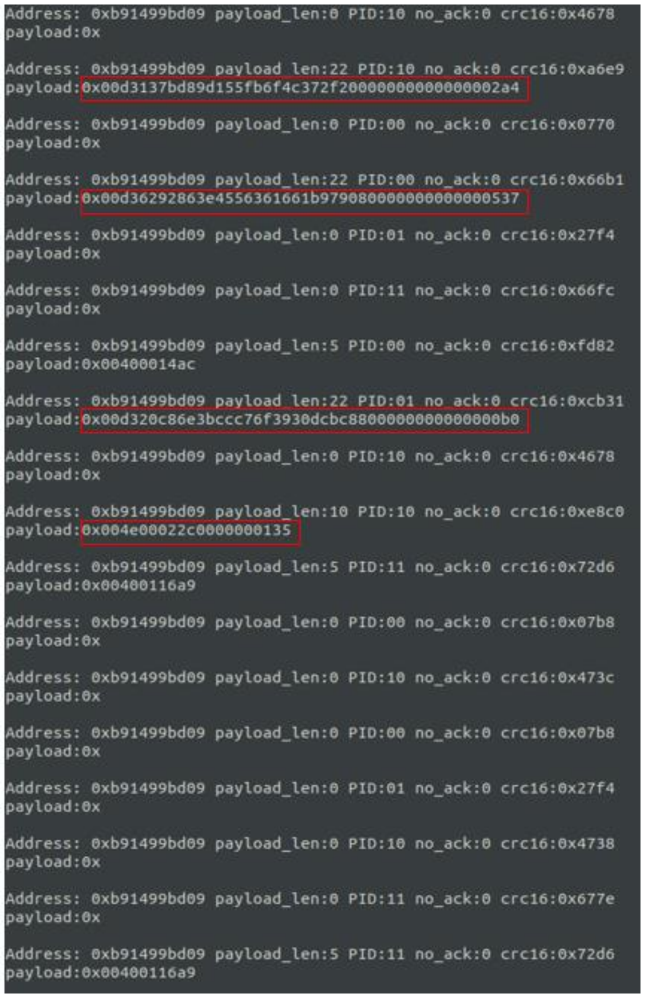
5.1 Results and Discussion

After detailed examinations and complete testing, the preamble and address of the packets were found to be *10101010* and '0xb91499bd09' respectively. Furthermore, the CRC has a length of 2 bytes. With these values, the packets were recovered. Some of our key findings are listed below:

- Payload length is either 0, 5, 10 or 22 bytes
- Payloads with lengths 0 and 5 appears to be 'keep-alive' signals
- 'Keep-alive' packets occur 4 times per second
- Payloads with lengths 22 appears to represent keypresses
- Payloads of length 10 are likely to be 'key-up' signals
- Pressing the same key again results in a different payload

We noticed that during the same paired session, the same keys when pressed have different payloads values and payload lengths.

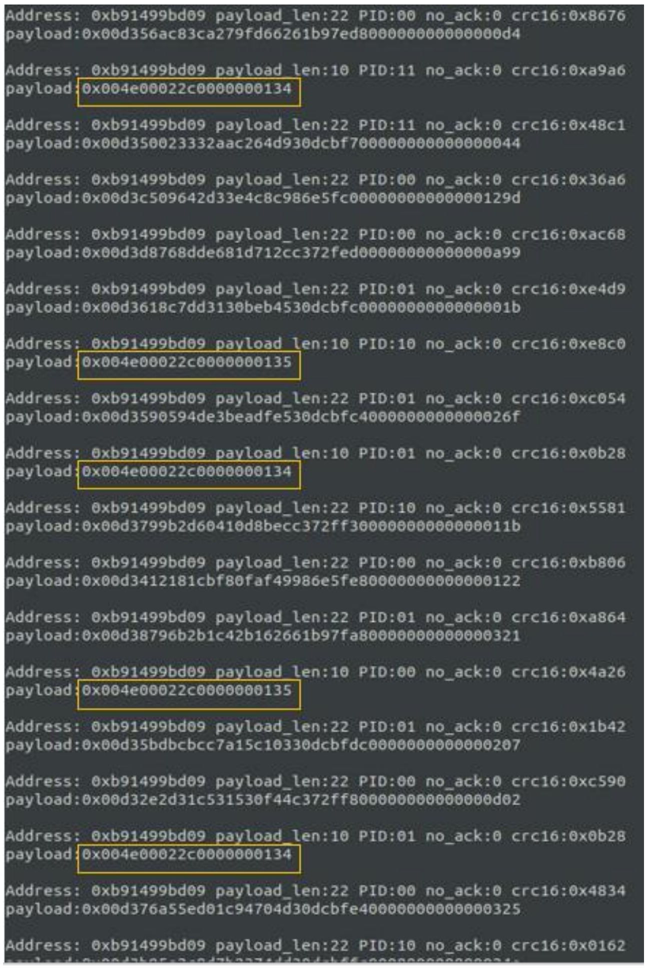
For example, keypresses of the character 'q' resulted in packets with payloads length of 22 and 10. This can be seen in Figure 12 below where these packets are highlighted in red.



```
Address: 0xb91499bd09 payload_len:0 PID:10 no_ack:0 crc16:0x4678
payload:0x

Address: 0xb91499bd09 payload_len:22 PID:10 no_ack:0 crc16:0xa6e9
payload:0x00d3137bd89d155fb6f4c372f20000000000000002a4

Address: 0xb91499bd09 payload_len:0 PID:00 no_ack:0 crc16:0x0770
payload:0x

Address: 0xb91499bd09 payload_len:22 PID:00 no_ack:0 crc16:0x66b1
payload:0x00d36292863e4556361661b979080000000000000537

Address: 0xb91499bd09 payload_len:0 PID:01 no_ack:0 crc16:0x27f4
payload:0x

Address: 0xb91499bd09 payload_len:0 PID:11 no_ack:0 crc16:0x66fc
payload:0x

Address: 0xb91499bd09 payload_len:5 PID:00 no_ack:0 crc16:0xfd82
payload:0x00400014ac

Address: 0xb91499bd09 payload_len:22 PID:01 no_ack:0 crc16:0xcb31
payload:0x00d320c86e3bccc76f3930dcbc8800000000000000b0

Address: 0xb91499bd09 payload_len:0 PID:10 no_ack:0 crc16:0x4678
payload:0x

Address: 0xb91499bd09 payload_len:10 PID:10 no_ack:0 crc16:0xe8c0
payload:0x004e00022c0000000135

Address: 0xb91499bd09 payload_len:5 PID:11 no_ack:0 crc16:0x72d6
payload:0x00400116a9

Address: 0xb91499bd09 payload_len:0 PID:00 no_ack:0 crc16:0x07b8
payload:0x

Address: 0xb91499bd09 payload_len:0 PID:10 no_ack:0 crc16:0x473c
payload:0x

Address: 0xb91499bd09 payload_len:0 PID:00 no_ack:0 crc16:0x07b8
payload:0x

Address: 0xb91499bd09 payload_len:0 PID:01 no_ack:0 crc16:0x27f4
payload:0x

Address: 0xb91499bd09 payload_len:0 PID:10 no_ack:0 crc16:0x4738
payload:0x

Address: 0xb91499bd09 payload_len:0 PID:11 no_ack:0 crc16:0x677e
payload:0x

Address: 0xb91499bd09 payload_len:5 PID:11 no_ack:0 crc16:0x72d6
payload:0x00400116a9
```

Figure 12: Packets with payloads of length 10 and 22 are likely to be caused by the keypresses

To help us analyse the packet payloads, in hopes of spotting some patterns, we modified our python script to only display packets with payload of length 10 and 22 which are highly likely to be the keypresses. The result of keypresses of the same letter ‘q’ is shown in Figure 13 below.



```
Address: 0xb91499bd09 payload_len:22 PID:00 no_ack:0 crc16:0x8676
payload:0x00d356ac83ca279fd66261b97ed800000000000000d4

Address: 0xb91499bd09 payload_len:10 PID:11 no_ack:0 crc16:0xa9a6
payload:0x004e00022c0000000134

Address: 0xb91499bd09 payload_len:22 PID:11 no_ack:0 crc16:0x48c1
payload:0x00d350023332aac264d930dcbf700000000000000044

Address: 0xb91499bd09 payload_len:22 PID:00 no_ack:0 crc16:0x36a6
payload:0x00d3c509642d33e4c8c986e5fc0000000000000129d

Address: 0xb91499bd09 payload_len:22 PID:00 no_ack:0 crc16:0xac68
payload:0x00d3d8768dde681d712cc372fed00000000000000a99

Address: 0xb91499bd09 payload_len:22 PID:01 no_ack:0 crc16:0xe4d9
payload:0x00d3618c7dd3130beb4530dcbfc0000000000000001b

Address: 0xb91499bd09 payload_len:10 PID:10 no_ack:0 crc16:0xe8c0
payload:0x004e00022c0000000135

Address: 0xb91499bd09 payload_len:22 PID:01 no_ack:0 crc16:0xc054
payload:0x00d3590594de3beadfe530dcbfc4000000000000026f

Address: 0xb91499bd09 payload_len:10 PID:01 no_ack:0 crc16:0x0b28
payload:0x004e00022c0000000134

Address: 0xb91499bd09 payload_len:22 PID:10 no_ack:0 crc16:0x5581
payload:0x00d3799b2d60410d8becc372ff30000000000000011b

Address: 0xb91499bd09 payload_len:22 PID:00 no_ack:0 crc16:0xb806
payload:0x00d3412181cbf80faf49986e5fe80000000000000122

Address: 0xb91499bd09 payload_len:22 PID:01 no_ack:0 crc16:0xa864
payload:0x00d38796b2b1c42b162661b97fa80000000000000321

Address: 0xb91499bd09 payload_len:10 PID:00 no_ack:0 crc16:0x4a26
payload:0x004e00022c0000000135

Address: 0xb91499bd09 payload_len:22 PID:01 no_ack:0 crc16:0x1b42
payload:0x00d35bdbcbcc7a15c10330dcbfdc0000000000000207

Address: 0xb91499bd09 payload_len:22 PID:00 no_ack:0 crc16:0xc590
payload:0x00d32e2d31c531530f44c372ff800000000000000d02

Address: 0xb91499bd09 payload_len:10 PID:01 no_ack:0 crc16:0x0b28
payload:0x004e00022c0000000134

Address: 0xb91499bd09 payload_len:22 PID:00 no_ack:0 crc16:0x4834
payload:0x00d376a55ed01c94704d30dcbfe40000000000000325

Address: 0xb91499bd09 payload_len:22 PID:10 no_ack:0 crc16:0x0162
```

Figure 13: Keypresses of ‘q’ with ‘keep alive’ signals removed and ‘key-up’ signals being highlighted in yellow.

We observed that packets of payloads length 10 appear after those of length 22. We tested this theory thoroughly and came to the conclusion that payloads of length 10 are ‘key-up’ signals. These ‘key-up’ signals usually alternate between these two values: ‘0x004e00022c0000000134’ and ‘0x004e00022c0000000135’.

However, these packets are not captured all the time as it is a challenge for us to capture all packets that are sent by the keyboard.

While there is a pattern from payloads of length 10, there is no noticeable correlation among the payloads with length 22. These values may be different due to the data being encrypted using AES-128-bit counter mode, while key-up signals are not encrypted.

Further analysis needs to be performed to verify if sniffing each keypress can be done accurately and reliably. Finally, CRC verification needs to be done to ensure that the packets analyzed are valid.

5.2 Breakthroughs

We were able to detect activity on the keyboard since the keyboard sends radio signals when user is pressing on the keys. These radio signals can be detected by the attacker.

This leaks information that the user is at home or at his office and the attacker can possibly carry out other attacks with this knowledge.

We have also successfully processed the Radio Frequency signal captured and identified the preamble and payload of the packet amidst the noise.

5.3 Further enhancements and extensions

There are many more areas for exploration and extensions to the current state of the project.

5.3.1 Complete breakthrough with successful decryption

With more time and resources, we could explore the possibility of decrypting the data transmitted even it uses the AES 128-bit symmetric encryption scheme.

AES 128-bit encryption is known to be secure and there is no way to decrypt it unless the key is known or if there are implementation errors.

However, over the past years, there have been CVEs regarding the retrieval of the secret key used for the symmetric encryption. These are retrieved during the pairing session between the keyboard and Logitech unifying receiver [9]. [22]

With the key, this attack would be largely complete. This would enable attackers to sniff, demodulate and decrypt the transmitted signals to obtain the exact keystrokes made by the user.

5.3.2 Attacks for Bluetooth devices

Other wireless keyboards, such as Bluetooth keyboards, are also gaining popularity.

Bluetooth operates over the 2.4GHz frequency band (2402 to 2480 MHz). It is a globally unlicensed and unregulated short-range radio, leveraging on frequency-hopping. There are two radio versions - Bluetooth Low Energy (BLE) and Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR).

We attempted to sniff several of Bluetooth devices:

1. Targus KB55 Multi-platform Bluetooth Keyboard (uses BR/EDR)
2. Careeach wp-809 fitness tracker (uses BLE)
3. Aukey EP-B4 Bluetooth earphones (uses BLE)

We tried sniffing and performing Man-in-the-Middle attacks on the Bluetooth keyboard using Ubertooth [25] and btproxy [30]. However, these attacks were not successful, likely because Ubertooth was unable to keep up with the DR/EDR device’s fast channel hopping. It is also likely that these attacks techniques were designed for only for specific devices and are not applicable to the Targus KB55 Multi-platform Bluetooth Keyboard.

Thus, we switched over to BLE devices like the fitness tracker and Bluetooth wireless earphones. To attack these devices, we installed GATTacker [31] and followed tutorials on learning to use GATTacker to perform an attack [32]. Unfortunately, we were only able to sniff advertisement packets sent by the device and unable to retrieve much useful information.

6. DEFENCES

Wired keyboards are not likely to be subjected to the same kind of attack described in this report as the data is not transmitted over the air as radio frequency signals.

Nevertheless, to enjoy the convenience of a wireless keyboard, if we were to get one, it is paramount that the keyboard uses a secure encryption scheme (e.g. AES 128-bit encryption) as evident from the attack demonstrated. Since the encryption scheme is known to be secure, this means there are no known attacks possible without the secret key. This ensures that most of the time, the attacker will not be able to make sense of the information that is transmitted over since the secret key is usually unavailable and a separate attack (during pairing of receiver and keyboard) must be carried out. When wireless keyboards were new in the market, they use unencrypted radio signals which were subjected to sniffing and even spoofing attacks. [23] The data transmitted could be easily available to attackers without much effort.

7. CHALLENGES

7.1 Picking up relevant skills and carrying out in-depth research

Coming into this project, neither of us had any prior knowledge. This project looks into a niche and budding area, where there is insufficient and unclear documentation available online. It was difficult to pick up the skills demanded within a short period of time.

The learning curve was steep as we had to master the various tools used – GQRX with HackRF for channel finding and GNURadio Companion for filtering and demodulating the signals captured. Online resources were very technical and demanded an in-depth knowledge on radio waves and digital signal processing.

We followed a series of online tutorials by Michael Ossmann's to learn the fundamentals of digital signal processing and using GNU Radio [24].

7.2 Antenna issues

The strength of the antenna affects the results we receive significantly, especially when we are unable differentiate signals from the keyboard keypress and surrounding noise.

We used GQRX extensively when exploring the signal strengths and found a suitable antenna that will help us distinguish noise from the actual keypresses.

7.3 Channel hopping of the Logitech K270

The greater the amount of interference in the surroundings, the more frequent and likely it is for the channels to hop. It was tedious and cumbersome to find the right channel from which the transmission is being done. Finding the right channel can is time-consuming, making this a tedious and inefficient process.

We overcame this challenge by capturing the signals and replaying the saved results. Although it is possible to write a script to detect valid packets while switching channels, we did not have sufficient time to implement this well.

7.4 Calculating the CRC checksum

The CRC of the ESB packet is calculated over the address, PCF, and payload. However, due to the structure of the ESB, the length of the PCF is 9 bits. This results in unaligned bytes hence we were unsure on how to proceed with calculating the CRC. Furthermore, the details of the 2 bytes CRC used for the ESB were scarce, with only the polynomial ('0x1021') and initial value ('0xFFFF') known [15]. Due to the lack of time, we couldn't make much progress on recomputing the CRC value to filter out valid packets.

8. CONCLUSION

A sniffing attack of wireless keyboards is one that is realistic and dangerous if successful. One must be cautious about the wireless keyboard he or she is buying and ensures it is up to date with the latest security specifications and has implemented the firmware updates to patch fixes from past CVEs.

With the new wave of Software Defined Radio technologies, it is even more crucial for users to be wary when using radio-based devices. The attacks may be evolving at a faster pace than the defenses and we may not know what the next possible vulnerability and attack vector for wireless keyboards may be. When dealing with high security data, it may be more advisable to use a wired keyboard.

Besides keyboards, there is a myriad of other wireless devices such as clickers and mouse that could be possibly subjected to such attacks with the use of an SDR. Hence, when it comes to wireless devices, users should take extra care in checking their security specifications before purchasing them.

9. ACKNOWLEDGEMENTS

This project is part of the curriculum for CS3235 Computer Security and has the support of the module coordinator, Associate Professor Hugh Norman Anderson. Tools listed were provided generously by the module coordinator.

10. REFERENCES

[1] Wireless Keyboards – Advantages and Disadvantages of a Wireless Keyboard. (2016, May 25). Retrieved from https://pcdreams.com.sg/wireless-keyboards-advantages-and-disadvantages-of-a-wireless-keyboard.)

[2] Tyson, J., Wilson, T. V., & Pollette, C. (2000, November 21). How Computer Keyboards Work. Retrieved from https://computer.howstuffworks.com/keyboard5.htm.

[3] How Are RF Signals Transmitted? Let's Talk Equipment. (2018, March 26). Retrieved from https://blog.aerohive.com/how-are-rf-signals-transmitted-lets-talk-equipment/.

[4] How Does Signal Modulation Work? (2016, September 1). Retrieved from https://community.keysight.com/community/keysight-blogs/oscilloscopes/blog/2016/09/01/how-does-signal-modulation-work.

[5] Gerez, S. H. (2016). Implementation of Digital Signal Processing: Some Background on GFSK Modulation. Retrieved from http://sabihgerez.com/ut/sendfile/sendfile.php/gfsk-intro.pdf?sendfile=gfsk-intro.pdf

[6] HackRF One. (n.d.). Retrieved from https://greatscottgadgets.com/hackrf/one/.

[7] Logitech Whitepaper: Logitech Advanced 2.4 GHz Technology . (2009). Retrieved from https://www.logitech.com/images/pdf/roem/Logitech_Adv_2_4_Ghz_Whitepaper_BPG2009.pdf

[8] CVE-2016-10761 Detail. (2019, June 29). Retrieved from https://nvd.nist.gov/vuln/detail/CVE-2016-10761.

[9] CVE-2019-13052 Detail. (2019, June 29). Retrieved from https://nvd.nist.gov/vuln/detail/CVE-2019-13052.

[10] CVE-2019-13054 Detail. (2019, June 29). Retrieved from https://nvd.nist.gov/vuln/detail/CVE-2019-13054.

[11] CVE-2019-13055 Detail. (2019, June 29). Retrieved from https://nvd.nist.gov/vuln/detail/CVE-2019-13055.

[12] Welcome to gqrx. (n.d.). Retrieved from http://gqrx.dk/.

[13] About GNU Radio · GNU Radio. (n.d.). Retrieved from https://www.gnuradio.org/about/.

[14] TutorialsWritePythonApplications. (n.d.). Retrieved from https://wiki.gnuradio.org/index.php/TutorialsWritePythonApplications.

[15] nRF24LE1 Ultra-low Power Wireless System On-Chip Solution Product Specification v1.6. (2010). Retrieved from https://infocenter.nordicsemi.com/pdf/nRF24LE1_PS_v1.6.pdf

[16] FCC ID JNZCU0007 Logitech Far East Ltd 2.4GHz Transceiver CU0007. (2010). Retrieved from https://fccid.io/JNZCU0007

[17] Mossmann. (n.d.). mossmann/hackrf. Retrieved from https://github.com/mossmann/hackrf/wiki/FAQ).

[18] Zlata. (2014, May 15). GRC Transmission Analysis: Getting To the Bytes. Retrieved from https://www.inguardians.com/grc-transmission-analysis-getting-to-the-bytes/.

[19] Avian's Blog. (2015, March 12). Retrieved from https://www.tablix.org/~avian/blog/archives/2015/03/notes_on_m_m_clock_recovery/.

[20] Intro to ShockBurst/Enhanced ShockBurst. (2015, November 8). Retrieved from https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/intro-to-shockburstenhanced-shockburst.

[21] RF Testing Methodology. (n.d.). Retrieved from https://nccgroup.github.io/RFTM/fsk_receiver.html.

[22] mame82. (n.d.). mame82/misc. Retrieved from https://github.com/mame82/misc/blob/master/logitech_vuln_summary.md.

[23] Fox-Brewster, T. (2016, March 8). 'MouseJack' Attacks Hack Wireless Keyboards And Mice From 100 Meters. Retrieved from https://www.forbes.com/sites/thomasbrewster/2016/02/23/bastille-mousejack-attacks-hack-keystrokes/#b63ab93527ca.

[24] Software Defined Radio with HackRF. (n.d.). Retrieved from https://greatscottgadgets.com/sdr/.

[25] Greatscottgadgets. (n.d.). greatscottgadgets/ubertooth. Retrieved from https://github.com/greatscottgadgets/ubertooth/wiki.

[26] Newlin, M. (n.d.). Compromising Computers through Wireless Mice and Keyboards. Retrieved from https://hackcon.org/uploads/333/11 - Marc Newlin.pdf.

[27] Arnaud, Martin, Alexandre, & Castro, J. M. (2015, June 8). Sniffing Crazyflie's radio with HackRF blue. Retrieved from https://www.bitcraze.io/2015/06/sniffing-crazyflies-radio-with-hackrf-blue/.

[28] HackRF One Bundle. (n.d.). Retrieved from https://hackerwarehouse.com/product/hackrf-one-kit/.

[29] Logitech Unifying receiver. (2019, October 3). Retrieved from https://en.wikipedia.org/wiki/Logitech_Unifying_receiver#/media/File:Logitech_Unifying_Receiver_USB.jpg.

[30] Conorpp. (2018, January 30). conorpp/btproxy. Retrieved from https://github.com/conorpp/btproxy.

[31] Securing. (2018, October 3). securing/gattacker. Retrieved from https://github.com/securing/gattacker.

[32] Anand. (2018, March 2). Exploiting Bluetooth Low Energy using Gattacker for IoT - Step-by-Step Guide. Retrieved from https://blog.attify.com/hacking-bluetooth-low-energy/.