



Investigating Radio Frequency Vulnerabilities in IoT (Using a HackRF)

A thesis presented for the degree of Bachelor of Science

Author: Sam Mantle
Supervisor: Eirini S Anthi
Moderator: Kirill Sidorov

Cardiff University
Computer Science and Informatics
May 2019

Abstract

In this thesis, a HackRF One software defined radio was used to capture radio frequency (RF) signals from a number of IoT devices. The devices were used in two different environments; the Cardiff University COMSC IoT Lab and a home network which has been configured for these experiments, this includes a secured and non-secured access point. The non-secured access point ensured the communication is not encrypted at layer 2 for additional analysis. A custom made Faraday cage was also used to reduce background noise in the signals. The IoT devices which were used include two Amazon Echo Dots (2nd Generation) running different firmware versions, a TP-Link HS100 WiFi Smart Plug and a Withings Bluetooth Blood Pressure Monitor. RF signals were captured during normal typical usage of each device and while carrying out some common attacks against them. These attacks include port scanning, deauthentication, denial of service, radio frequency replay attacks, jamming and using various functions within an IoT toolkit developed for exploiting TP-Link devices.

The RF signals were captured to observe the signals of typical usage of a device and analyse them for any patterns or consistency, giving an indication on whether the devices can be fingerprinted based on RF behaviour and potentially identify them in other captures. The RF signals captured during attacks were analysed to determine if there are signs of the attacks taking place and whether it seems possible to detect attacks based on RF signals. The RF based replay attack involved replaying typical usage to or from the devices to repeat an action or transmitted data. A replay of DoS and deauthentication attacks were also conducted in an attempt to do these attacks again without using the hack tools. A small python script was also created to pull out the raw values out of the RF captures for analysis. The RF Signals were captured by using GNU Radio Companion and analysed using Universal Radio Hacker and the custom python script. Some not but all of the attacks carried out were successful, and evidence of the attacks taking place was very evident within the captured RF signals. There were also some similarities in the data during normal usage, however in the Bluetooth captures it was not possible to determine whether the patterns are meaningful due to issues capturing all of the data.

Acknowledgements

Firstly I would like to thank my supervisor, Irene Anthi, for her support and guidance throughout the project.

Secondly I would like to thank my friends and family for their unconditional love and support throughout the project.

Finally I would like to thank my work colleagues, who share the same passion as me, for sharing their knowledge and giving me the motivation to do my best.

Contents

1	Introduction	8
2	Background	9
3	Methodology	10
3.1	IoT Devices	10
3.1.1	Amazon Echo Dot 2nd Gen	10
3.1.2	TP-Link HS100 UK WiFi Wall Plug	10
3.1.3	Withings Bluetooth Blood Pressure Monitor	10
3.2	Environment Setup	12
3.2.1	Network	12
3.2.2	Hardware	13
3.2.3	Software	15
3.3	Using the IoT Devices	18
3.4	Capturing RF Traffic	18
3.4.1	Environment	19
3.4.2	Amazon Echo Dot	19
3.4.3	TP-Link HS100 UK WiFi Wall Plug	20
3.4.4	Withings Bluetooth Blood Pressure Monitor	21
3.5	Attacks	21
3.5.1	Amazon Echo Dot	21
3.5.2	TP-Link HS100 UK WiFi Wall Plug	25
3.5.3	Withings Bluetooth Blood Pressure Monitor	26
4	Results	29
4.1	Environment	29
4.2	Amazon Echo Dot	29
4.2.1	Usage	29
4.2.2	Nmap Port Scan	29
4.2.3	Deauthentication	31
4.2.4	Replay attack	31
4.2.5	Jamming	33
4.3	TP-Link HS100 UK WiFi Wall Plug	33
4.3.1	Usage	33
4.3.2	Nmap Port Scan	33
4.3.3	Deauthentication	34
4.3.4	Replay attack	34
4.3.5	Jamming	34
4.3.6	SI6 Network's IoT Toolkit	36
4.4	Withings Bluetooth Blood Pressure Monitor	36
4.4.1	Usage	36
4.4.2	BlueSmack/DoS Attack	37
4.4.3	Replay Attack	38

4.4.4	Jamming Attack	38
5	Future Work	39
6	Conclusion	40
7	Reflection	41

List of Tables

3.1	Amazon Echo Dot Specification (Home)	11
3.2	Amazon Echo Dot Specification (COMSC IoT Lab)	11
3.3	TP-Link HS100 Smart Plug Specification	11
3.4	Withings Blood Pressure Monitor Specification	12
3.5	Channel 6 frequency range in MHz	19
3.6	Amazon Echo Dot Alexa Commands	20
3.7	TP-Link HS100 Smart Plug Commands	20
3.8	Withings Blood Pressure Monitor Commands	21
3.9	Attacks against the Echo Dot and the time to record RF signals.	22
3.10	Attacks against the TP-Link Plug and the time to record RF signals.	25
3.11	Attacks against the Withings Blood Pressure Monitor and the time to record RF signals.	26

List of Figures

3.1	Lab network diagram	12
3.2	Home network diagram	13
3.3	Home network channel	13
3.4	MAC Address Whitelist	14
3.5	DIY Faraday cage	15
3.6	“project_vm_files” folder shared and accessible within the VM	16
3.7	hackrf.info command confirming HackRF is connected and ready.	18
3.8	GNU Radio flow graph capturing RF on 2437MHz at 20MHz sample Rate	19
3.9	GNU Radio File Sink: Complex data type	20
3.10	Nmap port scan against the Amazon Echo Dot.	22
3.11	Confirming the adaptor wlan0 is connected	22
3.12	Set the wlan0 adaptor to monitor mode	22
3.13	Confirming the adaptor is in monitor mode with new device ID (wlan0mon)	23
3.14	airodump-ng: Monitor network access points	23
3.15	airodump-ng: Find the Echo Dot communicating with the AP	23
3.16	Confirming the MAC address found by airodump.	23
3.17	aireplay-ng: Sending disassociate packets while GNU Radio records RF signals.	24
3.18	GNU Radio: Signal flow graph for replay attack	24
3.19	GNU Radio: Signal flow graph for jamming using Gaussian noise.	25
3.20	IoT Toolkit: iot-scan tool scanning for TP-Link devices.	26
3.21	IoT Toolkit: iot-tl-plugin tool get-info command.	26
3.22	IoT Toolkit: iot-tl-plugin tool toggle command.	26
3.23	Bluetoothctl scan showing devices and corresponding MAC Address.	27
3.24	L2ping sending 600 byte packets to Withings BPM	28
4.1	RF Signals from home and lab environment	30
4.2	Script comparing environment captures.	30
4.3	Large string of values being repeated by Echo Dot on Open WiFi	31
4.4	Result of nmap port scan against Amazon Echo Dot	31
4.5	Nmap port scan: Echo Dot	32
4.6	Echo Dot successful dauthentication	32
4.7	RF Signals show evidence of deauthentication attack (Echo Dot)	32
4.8	RF Signals produced by the Gaussian noise jamming attack.	33
4.9	TP-Link Kasa app does not allow open WiFi networks	34
4.10	3 captures taken turning the device on then off again.	35
4.11	Result of nmap port scan against TP-Link HS100 Plug	35
4.12	RF Signals show evidence of deauthentication attack (TP-Link HS100)	35
4.13	Deauthentication attack successful (TP-Link HS100)	36
4.14	SI6 IoT Toolkit get-info command results show JSON data	36
4.15	RF Signals from toggle attack	36
4.16	GQRX Waterfall showing frequency shifting/hopping	37
4.17	RF Signals from starting a blood pressure test	37

4.18 Withings BP Monitor disconnects after BlueSmack/DoS	38
4.19 RF Signals from BlueSmack/DoS attack	38

Chapter 1

Introduction

With the significant increase of homes and organisations adopting IoT devices into their ecosystem to simplify and automate every day tasks it also introduces a security complexities, which, in result increases their exposure and attack surface to a wide range of cyber attacks. IoT devices can be one of the weakest links in any infrastructure, making them a good target for cyber attacks. There appears to be some lack of coverage with IoT devices when it comes to protection and detection against these attacks, it can also be very difficult to detect or prevent attacks which occur at the radio frequency layer. This will also boost motive to use radio frequency based attacks or any attacks which can be difficult to detect with a typical intrusion detection system (which are either heuristic or signature based).

This project will focus on analysing radio frequency vulnerabilities within a number of popular IoT devices and explore the possibility of detecting or preventing these attacks. These devices can operating over WiFi and Bluetooth will be tested against a variety of common attacks while recording radio frequency signals in real-time for analysis. The radio signals will be analysed for any signs of patterns or repetition which may indicate poor security, they will also be analysed to observe any obvious change which could indicate abnormal activity. Identifying anomalies or unexpected behaviour within RF activity and being able to identify attacks in the RF signals could improve attack detection and fill the gap in the IoT security field.

The issue with this is that radio frequency signals can be unpredictable due to the variety of background noise or radiation from other equipment, the range of communication modulation types, wireless security protocols and encryption. Environments where there are a large number of devices and equipment may prove difficulty in detecting sudden abnormal activity and could result in a lot of false positives, however in some environments this may not be the case. IoT devices do not necessarily have to involve the internet, they can often be used in environments where there is very little equipment or technology in surrounding areas, including remote locations such as remote villages or forests, meaning there is a higher likelihood an attack could be detected compared to typical consistent usage.

This report will begin by looking at some background research in chapter 2, covering IoT in general, cyber attacks targeting IoT, research on RF attacks and RF signal analysis and finally any existing approaches to fingerprinting or identification through signal data. The methodology in chapter 3 will begin by outlining the project environment setup including the network topology and each piece of hardware and software that will be used throughout the project. The methodology will then cover how the devices will be used, how the data will be captured and what attacks will be conducted against each device. The attacks will include typical attacks over the network and RF specific attacks. The results in chapter 4 will then start off by comparing normal RF signal activity on both networks, followed by analysis of the RF signal captures of devices usage and whether there are any similarities/consistency in repeated captures. After that, the attack results will include the results from the attacks themselves (any indicators of success or compromise) along with the analysis of the captured radio signals during the attacks. This will look at whether attacks can be detected within the radio signal captures compared to typical usage. This will be followed by opportunities of future work in chapter 5 and what could be possible based on the results gathered. The overall conclusion of the project will be in chapter 6 and my personal reflection in chapter 7.

Chapter 2

Background

There are a wide range of IoT/smart devices out there which are used at home, in business industries and healthcare. According to intel [1], most IoT devices reside within businesses and health care due to the functionality they bring to the business such as tracking inventory, monitoring, managing machines or appliances and helping to improve efficiency. Some IoT devices are also used in an effort to improve security such as smart locks for easy managed access control, biometrics and facial recognition locks and remote sensors.

There are a large number of IoT devices out in the open world which could be open to attack, with so many devices being available they often become the target of botnet attacks, which are then used to target other services and conduct attacks such as distributed denial of service attacks as outlined here [2]. Due to the fact that businesses rely on these devices so much it is important to ensure they continue to operate as intended, businesses can be highly affected if their IoT devices become infected or become victim of Dos or DDos attacks. Due to the hardware and software limitations, these devices can often be vulnerable to DoS attacks and deauthentication.

There are also a number of techniques used to exploit functionality of smart devices, such as creating malicious apps and skills for the Amazon Alexa devices outlined in this article [3]. This article explains how malicious applications or skills which can be installed on the amazon echo take advantage of how words can sound similar, meaning users can accidentally trigger the malicious app instead of a legitimate one. This can include triggering banking apps which will redirect users to a phishing site to steal credentials.

There are a number of RF based attacks which have already been used to attack devices by using a software defined radio such as a HackRF. These attacks have targeted devices such as drones and vehicles as mentioned in [4] and [5]. Replay attacks can sometimes be conducted to gain access to vehicles by replaying key signals to the vehicle. Amplification attacks can also be used against key-less entry vehicles to gain entry and start the vehicles. A number of attacks have also been done against basic devices such as alarm systems, door bells and motion sensors as shown here [6]. There is not much material covering devices operating on high frequencies including Bluetooth and WiFi, possibly due to the security protocols that have been implemented and the amount of data that gets transmitted at those frequencies.

Chapter 3

Methodology

3.1 IoT Devices

3.1.1 Amazon Echo Dot 2nd Gen

The first device included in this project is the Amazon Echo Dot 2nd generation. There are in fact two of these devices available to me running 2 different versions of firmware. This will also be useful to test how consistent the communications are between the two devices or compare how different the devices are at the RF layer. The Amazon Echo Dot is a smart speaker with Amazon Alexa, a virtual assistant which is controlled by voice commands. The smart speaker can be linked with various other applications and services to be controlled remotely including music services, online stores, news sources and much more. The Echo Dot can also control other IoT devices including but not limited to smart plugs and smart light bulbs. The first Echo Dot is my own which will be used at home. The technical specification is in table 3.1. This information was gathered from the device itself and from the FCC technical sheet [7]. The second Echo Dot device is from the university's COMSC IoT Lab and the technical specification is the same as above apart from the MAC Address and firmware version. See table 3.2 for more details.

3.1.2 TP-Link HS100 UK WiFi Wall Plug

The TP-Link HS100 is a smart plug adaptor which is used to control any electrical appliance via the user's smartphone. The adaptor plugs into the socket like normal while the electrical device plugs into the smart plug, the plug is then connected to the WiFi network and controlled by the TP-Link Kasa app on iOS or Android. According to the product page [8], the plug can also be controlled by other smart devices and services including as Amazon Alexa and Google Assistant. The technical information in 3.3 was gathered from the device itself (printed on the label or by using the SI6 Networks IoT toolkit to carry out reconnaissance), and from the FCC documentation found here [9].

3.1.3 Withings Bluetooth Blood Pressure Monitor

The Withings Blood Pressure Monitor is a simple smart device controlled by the Withings Health Mate app [10]. This is designed for users to record their blood pressure and keep track of their health data. The device syncs with the user's smartphone over Bluetooth or via the USB port on the device. The information gathered from this product was from the device's product page at Withings [11], from the app communicating directly to the device and the FCC documentation for the device itself [12] along with the FCC documentation BLE Antenna used within the device [13]. The detailed specification is in table 3.4

Model	Echo Dot Gen 2
Firmware Version	628568520
FCC ID	2AHSE-2045
Wireless Features	WiFi and Bluetooth
Operating Frequencies	2.402-2.48 GHz (Bluetooth) 2.412-2.462 GHz (WiFi) 5.18-5.24 GHz (WiFi) 5.745-5.825 GHz (WiFi)
WiFi Modulation Type	OFDM (B-PSK / Q-PSK / 16QAM / 64QAM) (WiFi)
MAC Address	4C:EF:C0:03:79:58

Table 3.1: Amazon Echo Dot Specification (Home)

Model	Echo Dot Gen 2
Firmware Version	613507720
FCC ID	2AHSE-2045
Wireless Features	WiFi and Bluetooth
Operating Frequencies	2.402-2.48 GHz (Bluetooth) 2.412-2.462 GHz (WiFi) 5.18-5.24 GHz (WiFi) 5.745-5.825 GHz (WiFi)
Modulation Type	OFDM (B-PSK / Q-PSK / 16QAM / 64QAM) (WiFi)
MAC Address	40:B4:CD:D5:AA:D5

Table 3.2: Amazon Echo Dot Specification (COMSC IoT Lab)

Model	HS100
Firmware Version	1.2.5 Build 171213 Rel.101014
FCC ID	TE7HS100
Wireless Features	WiFi only
Operating Frequencies	test
Modulation Type	OFDM (WiFi)
MAC Address	50:C7:BF:66:99:2E

Table 3.3: TP-Link HS100 Smart Plug Specification

Model	BP-801
Firmware Version	Unknown
FCC ID	XNAWPM02
Wireless Features	Bluetooth only
Operating Frequencies	2402MHz - 2480MHz
Antenna Rated Frequency	2400MHz - 2483MHz
Modulation Type	GFSK
MAC Address	40:B4:CD:D5:AA:D5

Table 3.4: Withings Blood Pressure Monitor Specification

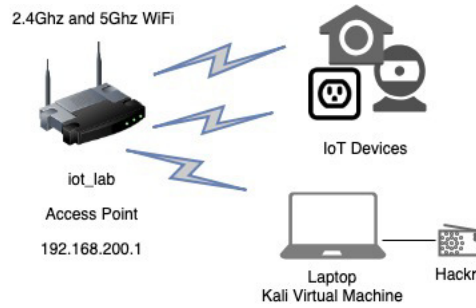


Figure 3.1: Lab network diagram

3.2 Environment Setup

3.2.1 Network

Lab Network

The experiments will be conducted in two different environments, the first environment will be within the COMSC IoT Lab where there is a test-bed of devices connected to an isolated network. The reason for this is that there are a range of devices connected to the network which can be tested, although the main focus will be on the devices mentioned above. The environment is also more realistic and true to a business environment as there are a lot of devices including end user devices, wireless access points for the university WiFi and appliances which may cause a lot of RF noise around the 2.4Ghz band. This meant that experiments can be conducted to see the changes in the RF signals and determine whether attacks can be easily detected within a high noise environment. The diagram for the network can be seen in figure 3.1

Home Network

For majority of the experiments and testing a low noise environment would be preferred, it would also be more beneficial to attempt some of the attacks with less noise and data in the RF signal captures. For this, an isolated virtual LAN network (VLAN) was created at home with a separate access point using an old router. This meant that the test devices can be connected to their own virtual network and attacks can be carried out without affecting the rest of the network. This also provided more control over the access point configuration including specifying exactly what channel and security the WiFi access point operates on to make the investigations easier. The network topology can be seen in figure 3.2.

The network monitor utility built into macOS was used to monitor the channels of WiFi access points in the area to determine what channel would be best to reduce interference and noise. Channel 6 was used

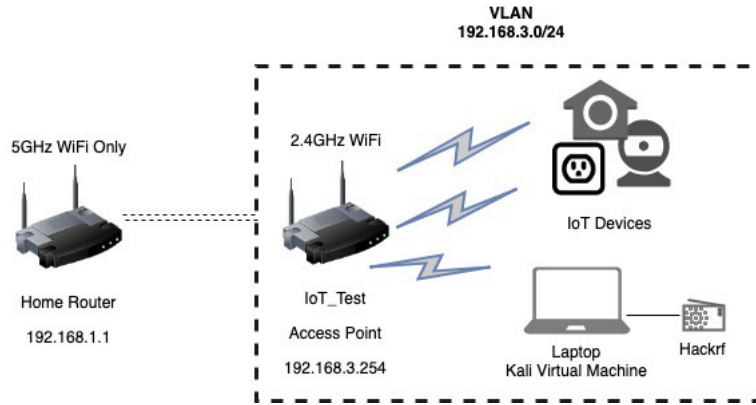


Figure 3.2: Home network diagram

Wireless Mode:	802.11n Only
Channel Bandwidth:	20 MHz
Radio Channel:	6

Figure 3.3: Home network channel

for the home IoT access point and the 2.4GHz WiFi on my main home router was disabled to reduce noise, leaving the 5GHz channel free for home devices which would not interfere with the experiments. The channel bandwidth was configured to use 20Mhz on 802.11n only as shown in figure 3.3 to ensure the HackRF can capture all the data.

The secure access point will be using WPA-2 Personal PSK security which uses AES encryption with a key length of 128 bits to encrypt the communication, meaning in theory the communication be different each time as although the key is the same, the IP header information within the packets are likely to be different each time due to various fields including the identification field, time to live (TTL), checksum and padding. This means that after encryption, the traffic will likely look very different from layer 3 onward regardless of the same commands being repeated. The one issue with this level of security is that the source and destination MAC Addresses can still be observed. With all this in mind, the RF signals may still show consistent patterns depending on the amount of traffic.

Throughout the project, the WiFi security may be removed in order to monitor the unencrypted traffic. Broadcasting an open WiFi access point with internet access from is not secure, therefore internet filters and a MAC address whitelist will be setup to ensure only authorised test devices have access to the network and any unknown devices will not be able to connect. The MAC addresses are unique to each device and can be specified in the access point configuration as shown in figure 3.4. The virtual machines on the laptop will also have the same MAC Address only when using the shared network adaptor (using the built-in WiFi). However a separate USB WiFi adaptor will be used for some of the experiments, meaning in some instances the virtual machine will show up as a different MAC address.

3.2.2 Hardware

Laptop (Macbook Pro 2017)

The laptop being used during the project is an Apple Macbook Pro 2017, due to limited ports on the laptop, USB to USB-C adaptors will be used for connecting peripherals when needed. The laptop is running MacOS Mojave and virtual machines will be used to carry out attacks and record the data which will be explained

MAC Filter Mode:	
Accept	
MAC Filter List:	
MAC Address:	Wireless Client Description:
<input type="text"/>	<input type="text"/>
8C8590B8BEB2	Laptop
00C0CA97CB7F	Alfa Network Adaptor
4CEFC0037958	Amazon Echo Dot 2
78009EF813D8	Samsung Phone
50c7bf66992e	TP Link HS100
40B4CDD5AAD5	Amazon Echo Dot
DCA904E2C4F6	iPhone

Figure 3.4: MAC Address Whitelist

below. VMware Fusion 11 will be used to virtualise Kali Linux and Parrot OS which will be explained in the software section below. The Macbook laptop has 16GB of memory (RAM) which will be useful for running the virtual machines analysing the captured RF signals, as the file sizes at a 20MHz sample rate can be large and loading them in software may require extensive memory.

HackRF and ANT500

The HackRF is a software defined radio by Great Scott Gadgets which can be used to receive and transmit radio signals from 1 MHz to 6 GHz[14]. This device is the key piece of hardware for the project as it will be used to capture and send packets to and from the IoT devices. The HackRF supports up to a 20MHz Sample rate which is enough for capturing a single 2.4Ghz WiFi channel, however Bluetooth may vary. The antenna is officially rated for up to 1100Mhz (1.1Ghz) however the HackRF and antenna will be close to the devices therefore shouldn't be an issue, although there may still be the possibility of losing data during the transmissions and captures. This factor will be taken into consideration when analysing the results.

iPhone 7 (iOS 12.2)

An Apple iPhone 7 running iOS 12.2 will be used to run the apps to control the IoT Devices. It also has the ability to disable either WiFi or Bluetooth separately when needed to avoid additional noise and interference at close range.

Samsung J3 (Android 5.1.1)

This android phone will be used to load an older version of the apps if needed. The apps will be downloaded from apkmirror [15] as an .apk file and loaded onto the phone.

ALFA USB 2.4GHz WiFi adaptor

Due to using virtual machines to carry out attacks, a separate USB WiFi adaptor will be needed which can be assigned strictly to the virtual machine to carry out WiFi reconnaissance and attacks. This ALFA Networks USB WiFi adaptor was chosen as it uses the Atheros AR9002U chipset [16] which is supported by Kali Linux and Parrot OS straight out of the box. This WiFi adaptor also supports monitoring mode and packet injection which is necessary for some of the experiments.

Homemade DIY Faraday Cage

In order to block some of the background RF signals and noise it was necessary to make a quick and easy DIY Faraday cage. This was made out of a cardboard box and several layers of foil. The idea is to put



Figure 3.5: DIY Faraday cage

a device and the HackRF ANT500 antenna inside the box which should shield them from the rest of the background noise or at least reduce the signal strength of the noise recorded by the HackRF. The box can be seen in figure 3.5. The hole for the HackRF antenna can be seen, the hole is small enough for the antenna to fit tightly to reduce the chance of signals getting in.

3.2.3 Software

Amazon Alexa iOS App

In order to use the Amazon Echo Dot device, the Amazon Alexa iOS app was installed to set up the device and connect it to the network. At the start of the experiment the app version which was installed was recorded to be 2.2.265941.0. This app will only be used to setup the device and check the firmware version, the commands given to the Echo Dot will only be done via voice commands directly to the device.

Kasa iOS App

Kasa is the app developed by TP-Link for connecting and managing TP-Link smart devices on the network [17]. This app will be used to connect to the TP-Link smart plug to set it up on each network and send commands to it locally. The app and plug will not be connected to the internet and will be used as a local only device. The iOS app version I have installed is 2.11.

Withings Health Mate iOS App

The Withings Health Mate App is a fitness, activity and health tracking app [10]. This will be used only to connect to the Withings Bluetooth blood pressure monitor and take readings. For ethical and privacy reasons, the information used to setup the application will be a test persona and not real personal information (such as name and email address). The device and app will only be used on myself to test the device and monitor the Bluetooth RF signals.

VMWare Fusion 11

VMWare Fusion 11 is a hypervisor built to run virtual machines on macOS [18]. This will be used to run the Kali Linux virtual machine which is explained below. Fusion also supports forwarding USB devices which is necessary for assigning the hackrf and USB WiFi adaptor to the virtual machine. It also supports shared folders therefore a folder will be 'mounted' within the virtual machine which is stored on the host laptop drive. This will be more efficient instead of storing the project files inside the virtual hard drive created by the virtual machine and copying them to the host. This shared folder setup can be seen in figure 3.6 and

is setup with read and write permissions. This will make it easier for accessing the files to analyse using software installed on the host macOS system instead of using the virtual machine.

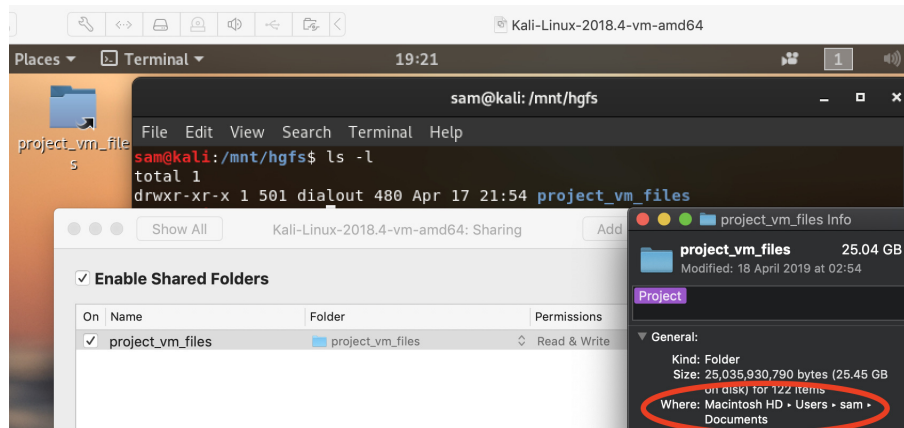


Figure 3.6: “project_vm_files” folder shared and accessible within the VM

Kali Linux (2018.4 VM amd64)

Kali Linux is a debian based linux operating system which is maintained and funded by Offensive Security [19]. This operating system is designed for security professionals for auditing, penetration testing and forensics. A custom VMWare virtual machine image will be used which was downloaded from the Offensive Security website [20], specifically version 2018.4 at the time of downloading. There are a range of security tools, drivers and device support included (including the HackRF), which makes this ideal for my experiments. In some instances additional applications may be installed. All the applications that will be used are mentioned further below.

Nmap

Nmap is a network mapping/discovery tool which is used to scan for devices on the network and check for open ports on available devices. This tool can also attempt to detect the device’s operating system and what services it supports [21]. This tool is an popular reconnaissance tool used by attackers and can generate a lot of network traffic.

Aircrack-ng Suite

Aircrack-ng is a set of tools used to assess WiFi network security [22]. This suite is included within Kali Linux and will be used for monitoring the WiFi networks and experimenting with some attacks. This tool will be used with the ALFA USB WiFi adaptor mentioned previously, this will allow me to use the airmon-ng and airodump-ng tools to set the adaptor to monitor mode and monitor WiFi devices in the area. The airplay-ng tool will then be used to send traffic to the devices and carry out attacks.

L2ping

L2ping is a command line tool which can be used in a similar way to the standard ping command, it can discover and check whether a specific Bluetooth device is reachable. It is used to send L2CAP packets to the Bluetooth devices via their MAC address [23]. This L2Pping tool can be used to send a large number of packets to a Bluetooth device to flood it, which may result in communication disruption, the device disconnecting, the device crashing or the device shutting down. This is also known as a “denial of service” (DoS) attack.

HackRF command-line Tools

As part of the HackRF package, there are a variety of command line tools available to use, these tools are also already available in Kali Linux. Firstly, the `hackrf.info` tool is used to verify the HackRF device that is plugged in, it will also display all the relevant information about the HackRf including hardware version, firmware version and software version. This tool is used to verify it is up to date, plugged in and ready to use. Secondly, there is a `hackrf.transfer` tool which is used to send and receive data. Optional parameters can be set to change the sample rate, gain and input/output file information.

GNU Radio

GNU Radio is an open-source toolkit which is used to implement software radios [24]. GNU Radio can be used with radio hardware (such as the HackRF) to create a software-defined radio solution. GNU Radio also includes a GUI tool called GNU Radio Companion [25], this tool is used to create signal flow graphs using a drag and drop interface and automatically generate the flow graph code. This will be used to communicate with the HackRF hardware to capture, save and transmit RF signals to and from the IoT devices.

GQRX

As stated on the GQRX web page [26], “Gqrx is an open source software defined radio receiver (SDR) powered by the GNU Radio and the Qt graphical toolkit”. This Software will be useful for monitoring frequencies in real-time without saving to a file and displaying the data on a more detailed waterfall spectrum than the one available in GNU Radio Companion. This tool is also particularly useful for observing and pin-pointing the frequency in which data is transmitted.

Universal Radio Hacker (URH)

Universal Radio Hacker is a python based tool developed by Johannes Pohl which is used to investigate unknown wireless protocols [27]. This tool can be used to directly communicate with a software defined radio or simply used to load existing RF signal captures. This tool will be used to load the captured RF signals from GNU Radio Companion and analyse them in more detail. This tool will display the captured signal in a variety of formats (Analogue, Spectrogram and Demodulate) and allow me to attempt to demodulate the signals, exclude noise and decode the data. URH Supports ASK, FSK and PSK modulation types, the IoT devices being tested use either PSK or FSK modulation. The tool will be useful as a visual representation of the signals and check for patterns or any areas of consistency and any obvious changes during attacks.

SI6 Network’s IoT Toolkit

As described in this web page [28], The “SI6 Networks IoT toolkit is a set of IoT security assessment and trouble-shooting tools”. As part of this toolkit, there are tools for specifically gathering information and attacking TP-Link devices including the HS100 Smart Plug which will be tested. A presentation [29] created by Fernando Gont, the developer of the toolkit, covers attacks against TP-Link devices using the toolkit. The first tool that will be used from the toolkit is “`iotsan`”, this scans the network for any IoT devices and returns their device name and IP address. The second tool which will be used is “`iot-tl-plug`” which has a range of functions including sending a “`toggle`” command to switch the plug on and off, as well as a “`getinfo`” command which returns verbose information about the device including firmware version.

Python 3

Python 3 will be used to create a small script used to load the RF signal capture files to read the data and attempt to process them to investigate the raw numbers in detail. The file will look at the number of raw IQ values in the signal capture, do a unique count on those values, find how many of those values occur more than once and finally show the top 10 most occurring values from the capture.


```

sam@kali:~$ hackrf_info
hackrf_info version: unknown
libhackrf version: unknown (0.5)
Found HackRF
Index: 0
Serial number: 0000000000000000406464c823443a4b
Board ID Number: 2 (HackRF One)
Firmware Version: 2018.01.1 (API:1.02)
Part ID Number: 0xa000cb3c 0x005a474f

```

Figure 3.7: hackrf_info command confirming HackRF is connected and ready.

3.3 Using the IoT Devices

During the experiments some usage tests will be repeated to check for consistency and patterns. Tests will be carried out on the home network and at the IoT lab to compare results at different noise levels. Some tests will also involve the use of the Faraday cage. As explained previously, some tests will also involve connecting the devices to an open WiFi network in order to remove the transport encryption and observe any change in the signals.

In some cases GQRX will be used to monitor the device usage in real time to observe how much data is being transmitted and for long long. This will give me an indication on how long it will take to use the HackRF and capturing software to capture all the relevant signals. This will also give an indication on how the data is being sent across the frequency range, including pinpointing the centre frequency and observing any frequency shifting or frequency hopping.

3.4 Capturing RF Traffic

To get started, the HackRF was connected to the Kali Linux virtual machine and verified by using the “hackrf_info” command as shown in figure 3.7. The next step was to ensure other software such as QGRX and GNU Radio Companion was working with the HackRF correctly, this was confirmed by loading up the software and doing some quick test captures.

Both the “hackrf_transfer” command-line tool and GNU Radio Companion may be used to capture files. However, GNU Radio Companion will more likely be used as it supports capturing the raw I/Q data¹ and storing them as a 32-bit complex data type, whereas the hackrf_transfer tool will only save in 8bit signed integer values which means although the file sizes may be smaller and working with the data may be more efficient, it may cause issues during analysis. Universal Radio Hacker needs the values to be complex, therefore any raw I/Q values captured in 8-bit signed integer format from the HackRF will need to be converted. The UI in GNU Radio Companion is also easy to use and supports the ability to show graphs as the data is being captured in real-time which is useful.

To analyse the results, Universal Radio Hacker will be used to load the files and analyse. Universal Radio Hacker has a lot of features including viewing the signals in analogue form, demodulated or as a spectrogram. It also has the ability to adjust noise cancellation, modulation type and provides the ability to view the demodulated data as binary or hex values. This will be useful for attempting to analyse the signals for repetition or patterns. A simple python script will also be used to read how many complex IQ values are in the capture, how many unique values there are by counting duplicates and analysing what values occur the most. This will include the top 10 most occurring values.

¹I/Q is an abbreviation for “In-Phase and Quadrature” signals. Representing the sine and cosine waveforms

Lower Frequency	Center Frequency	Upper Frequency
2426	2437	2448

Table 3.5: Channel 6 frequency range in MHz

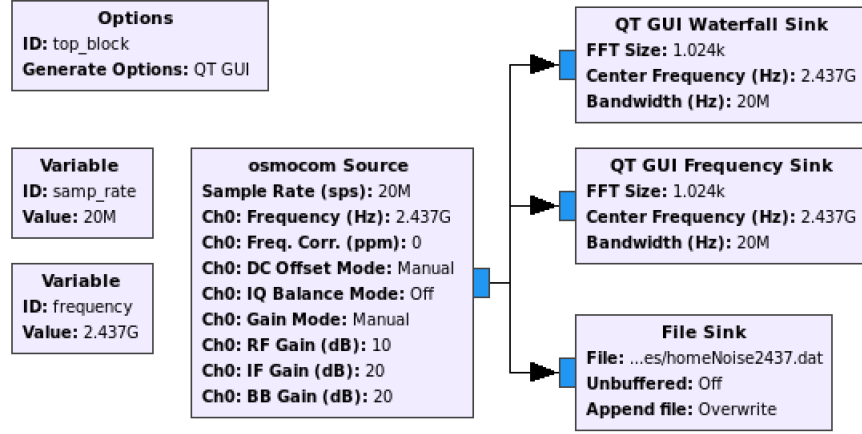


Figure 3.8: GNU Radio flow graph capturing RF on 2437MHz at 20MHz sample Rate

3.4.1 Environment

As an initial starting point, RF captures were taken of both the home environment and the IoT Lab environment. To determine what frequency had to be captured, the network monitor utility built into macOS was used to observe what channel the IoT Lab WiFi is using. This was not necessary for the home network as this was already configured to be on channel 6, the channel frequencies are in table 3.5.

GNU Radio Companion was used to capture the RF signals and store them as 32-bit complex I/Q values. The signal flow graph for capturing the data is shown in figure 3.8 and figure 3.9. This same flow will be used to capture traffic from all the devices, the only thing that will change if the output File Sink file name and the centre frequency when recording Bluetooth. A waterfall and frequency sink is used for visual representation of the data being captured to ensure it is working as it should. The sample Rate was set to 20MHz which is the maximum rate supported by the HackRF. This ensures as much data as possible is captured within the channel 6 frequency range. A sample was also taken inside the Faraday cage to compare results to see if there was any difference in using the Faraday cage to reduce noise.

3.4.2 Amazon Echo Dot

The experiments involving the Amazon Echo Dot will involve asking Alexa a typical command which does not involve any third party apps or skills. In testing, two different types of voice commands will be sent, one involving information retrieval and another one will be an action. There will also be a test using the iOS Amazon Alexa app to adjust the volume. Some tests are expected to cause more traffic, for example the information retrieval commands may produce more signals due to the device making additional outbound requests over the network. The table 3.6 shows the commands which will be given to the Echo during testing and how long it will take to record the signals the HackRF.

To capture the signals from the Amazon Echo Dot, the device was first connected to the secure WiFi access point and placed within the Faraday cage along with the antenna of the HackRF. GNU Radio Companion was used again to capture the signals while the commands in table 3.6 were completed. As explained previously, the tests will be completed more than once to compare for patterns and consistency. The flow diagram used to

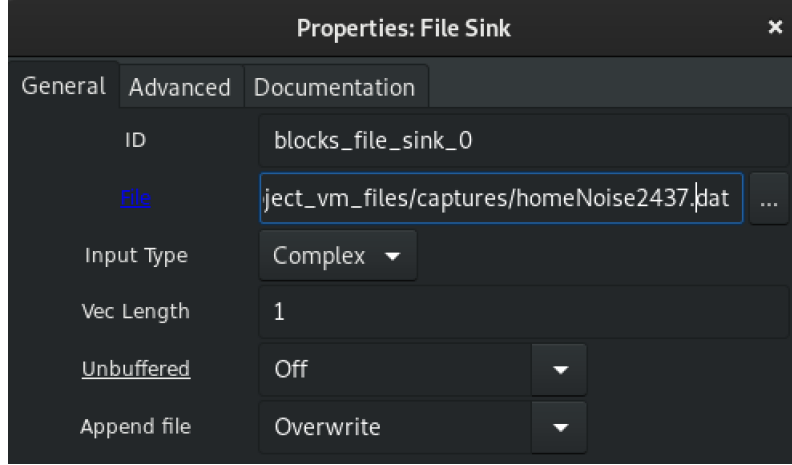


Figure 3.9: GNU Radio File Sink: Complex data type

Type of Command	Command	Time to Record Signal
Voice - Information Retrieval	“Alexa, what is the weather like in Cardiff?”	15 Seconds
Voice - Action	“Alexa, set a timer for 10 seconds”	8 Seconds
Alexa App Command	Turn the volume down by 1 mark	5 Seconds

Table 3.6: Amazon Echo Dot Alexa Commands

capture this data is the same as the one seen in figure 3.8 and figure 3.9, except the file name will be different.

Once the captures have been completed on the secure network, the Echo Dot was re-configured to connect to the open WiFi access point and more captures were taken doing the same tests.

3.4.3 TP-Link HS100 UK WiFi Wall Plug

The experiments involving the TP-Link Smart Plug will involve using the Kasa app to turn the device on and off. This is a simple device and doesn’t have much functionality, there is also very little delay while sending the commands over the network, meaning capturing the signals from this device will be much quicker than the other devices. The usage and the time taken to record the signal is outlined in table 3.7

Command	Time to Record Signal
Use app to toggle on then off	5 Seconds

Table 3.7: TP-Link HS100 Smart Plug Commands

Again, the method of capturing the signals was the same as the Amazon Echo Dot, however once the signals were captured on the secure network there was an issue connecting to the open WiFi network. Due to limitations with the Kasa app and device, it could not be connected to the open WiFi network. An older version of the Kasa Smart android app was installed onto the Samsung J3 android phone, this apk was downloaded from “apkmirror” [30]. The version downloaded was v1.7.7.638 uploaded May 16, 2017 at 11:18PM GMT. This app had the same limitations, therefore it was not possible to connect the device to the open WiFi network at all.

Command	Time to Record Signal
Start the blood pressure test (until it fails)	8 Seconds
Complete a blood pressure test	70 Seconds

Table 3.8: Withings Blood Pressure Monitor Commands

3.4.4 Withings Bluetooth Blood Pressure Monitor

The experiments involving the blood pressure monitor will include turning the device on and pairing with the phone. Once the devices are paired, a notification will appear on the phone requesting access to launch the Withings Health Mate app. The testing will also include starting a blood pressure test (until it fails due to not being on a body) and attempt to complete a full blood pressure reading. While the blood pressure monitor is being used, all other devices operating on the same frequency in the area will be turned off to minimise background noise. The usage and the time taken to record the signal is outlined in table 3.8.

The method for capturing the signals are again very similar to previously, however the frequency variable will also be changed to 2.442G (2442Mhz) due to the centre frequency being different. GQRX will also be used to observe the signals in real time due to the chance of frequency hopping and data spanning across a larger frequency range (83Mhz) where the HackRF is only capable of capturing 20Mhz.

3.5 Attacks

This experiments will look at some common basic attacks and attempting to progress through the first 4 stages of the cyber killchain. Reconnaissance will involve tasks such as scanning the network and devices gathering information and determining how to carry out an attack. Weaponisation and delivery will involve collecting, creating and sending attacks to the devices, while exploitation will take advantage of any potential vulnerabilities which may result in success. The attacks that will be carried out will not involve installation or post-compromise action. The idea is to carry out the initial attacks and attempt to gather data on the lower layers by capturing the RF signals and analysing the data to see if there is any signs of attack. This is important as the aim of intrusion detection is to detect these initial stages of attack before full compromise and post-compromise action on objectives. There will also be some common RF specific attacks, which are commonly done against basic or low power devices device such as vehicle remotes and alarm systems. This can include Jamming and RF replay attacks. Some attacks will generate more signals than others which may be easier to detect on the RF layer, for example, flood and brute force attacks.

3.5.1 Amazon Echo Dot

There will be a variety of attacks completed against the Amazon Echo Dot, while the attacks are taking place, RF signals will be captured for analysis. The time of the capture will depend on the attack. RF Based attacks such as jamming and the replay attack will not be captured, as the attack itself involves transmitting a signal using the HackRF and it is not possible to record a signal while transmitting one simultaneously. Table 3.9 shows each attack with the corresponding time frame for the capture.

Nmap Port Scan

One of the attacks that that will be carried out against the Echo Dot is an nmap port scan. This is a very popular reconnaissance tactic used to discover all the open, filtered and closed ports on a given device. The tool can be used to scan a range of ports specified or a list of known popular ports (such as port 22 for SSH and port 80 for web servers). For this attack, the nmap scan tool will be used to scan the 1000 most common ports using the command **nmap 192.168.3.19 -Pn --top-ports 1000 -v** as seen in figure 3.10. The IP Address specified in the command is the Echo Dot, the **-Pn** flag is used to ignore host discovery and treat the device as online. This command is used when the device refuses to respond to a ping. The –

Attack	Time to Record Signal
Nmap Port Scan	10 Seconds
Deauthentication	10 Seconds
Replay Attack	N/A
Jamming	N/A

Table 3.9: Attacks against the Echo Dot and the time to record RF signals.

top-ports 1000 flag is specified to scan the top 1000 most common ports. This is enough to keep the scan running while the RF signals are captured. Finally, the **-v** flag is set for verbose output, this is used to see more information as the test is running to ensure it is working as expected.

```
sam@kali:~$ nmap 192.168.3.19 -Pn --top-ports 1000 -v
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-09 00:19 EDT
```

Figure 3.10: Nmap port scan against the Amazon Echo Dot.

Deauthentication Attack

The next attack that will be done against the Echo Dot is a deauthentication attack. According to [31], this is one of the most common DoS attacks on 802.11 networks. This attack will be done by using the aircrack-ng suite. The concept is to spoof the MAC address of a WiFi access point and send disassociate packets to the victim device which is connected to that access point. The device will then assume the packets are legitimate and disconnect from the access point. Once the victim disconnects, it will attempt to reconnect to that network SSID. This attack is often done by attackers in an attempt to force devices to disconnect from a network and reconnect to their own spoofed malicious network posing as the legitimate network. This can also be done as a DoS attack if the packets are sent on repeat, forcing the device to never reconnect.

To start off, the `iwconfig` command was used to check the ALFA USB WiFi adaptor connected and get the device ID as shown in figure 3.11 and continued by using the `airmon-ng` tool to put the WiFi adaptor into monitor mode as shown in figure 3.12. The `iwconfig` command was used again to check the device was put into monitor mode and confirm the new device ID, as this can change after the device is put into monitor mode as shown in figure 3.13.

```
sam@kali:~$ iwconfig
wlan0 IEEE 802.11 ESSID:off/any
        Mode:Managed Access Point: Not-Associated Tx-Power=20 dBm
        Retry short limit:7 RTS thr:off Fragment thr:off
        Power Management:off
```

Figure 3.11: Confirming the adaptor wlan0 is connected

```
sam@kali:~$ sudo airmon-ng start wlan0
```

Figure 3.12: Set the wlan0 adaptor to monitor mode

The next step was to use the `airodump-ng` tool command to monitor the area for network devices, find the IoT test access point along with its MAC Address and use the `airodump` again against the access point to find devices communicating with it, which should include the Echo Dot device along with its MAC address too. These commands are seen in figures 3.14 and 3.15. The MAC Addresses can also be confirmed using the

```
sam@kali:~$ iwconfig
wlan0mon IEEE 802.11 Mode:Monitor Frequency:2.457 GHz Tx-Power=20 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Power Management:off
```

Figure 3.13: Confirming the adaptor is in monitor mode with new device ID (wlan0mon)

website Macvendor [32], which is used to find the vendor in which the MAC address is associated with, in this case the website can be used to confirm the MAC Address connecting to the access point is an Amazon device. This can be seen in figure 3.16

```
sam@kali:~$ sudo airodump-ng wlan0mon
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
32:5A:3A:5D:1C:88	-55	19	0 0	6	130	WPA2	CCMP	PSK	IoT_Test_Secure
30:5A:3A:5D:1C:88	-55	17	1 0	6	130	OPN			IoT_Test(HACKING)

Figure 3.14: airodump-ng: Monitor network access points

```
sam@kali:~$ sudo airodump-ng wlan0mon --bssid 30:5A:3A:5D:1C:88 --channel 6
[sudo] password for sam:
```

BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
30:5A:3A:5D:1C:88	-72	43	295	91 0	6	130	OPN			IoT_Test(HACKING)

BSSID	STATION	PWR	Rate	Lost	Frames	Probe
30:5A:3A:5D:1C:88	4C:EF:C0:03:79:58	-44	0e-24e	1515	998	
30:5A:3A:5D:1C:88	78:00:9E:F8:13:D8	-55	0 - 1	0	6	

Figure 3.15: airodump-ng: Find the Echo Dot communicating with the AP

Enter a MAC Address

4C:EF:C0:03:79:58

Amazon Technologies Inc.

Figure 3.16: Confirming the MAC address found by airodump.

Once the devices and MAC Addresses have been found, the final step was to start the deauthentication attack using the aireplay-ng tool to send disassociate packets, this forces the client device (in this case the Echo Dot) to stop using the access point it is connected to. Once the packets start sending, GNU Radio Companion is used to capture the signals during the attack until the device is disconnected. The command used was **aireplay-ng -deauth 0 -c 4C:EF:C0:03:79:58 -a 30:5A:3A:5D:1C:88 wlan0mon**. The **-deauth** flag means deauthentication, **0** means send packets continuously, this is done because the device will attempt to reconnect once the attack has stopped. The **-a** and **-c** flags are used to specify the access point and client device respectively. Finally, **wlan0mon** is the WiFi interface to use to send the packets. This is shown in figure 3.17.

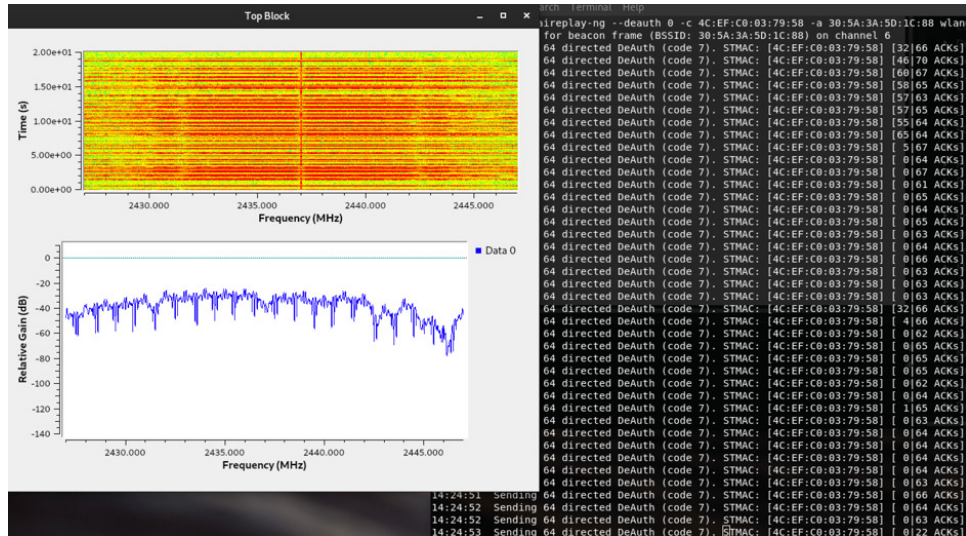


Figure 3.17: aireplay-ng: Sending disassociate packets while GNU Radio records RF signals.

Replay Attack

The next attack to carry out is a replay attack. The concept is to capture the RF signals while using the device as done previously, and re-transmit those signals in an attempt to replay a command to the device. In this case, GNU radio will be used to capture the signals and attempt to replay them to the device. The replay attack will be done with all 3 commands listed in table 3.6. The signals captured from using the Echo Dot previously will be used in a new flow created in GNU Radio Companion, the flow is essentially the reverse of the capture flow. This is shown in figure 3.18. Again, the waterfall and frequency sinks are used for visual representation of the data being sent to ensure it is working as it should. As a final test, a replay attack will be done using the deauthentication signal capture, this will be done to test whether a deauthentication attack can be achieved without using the aircrack-ng suite.

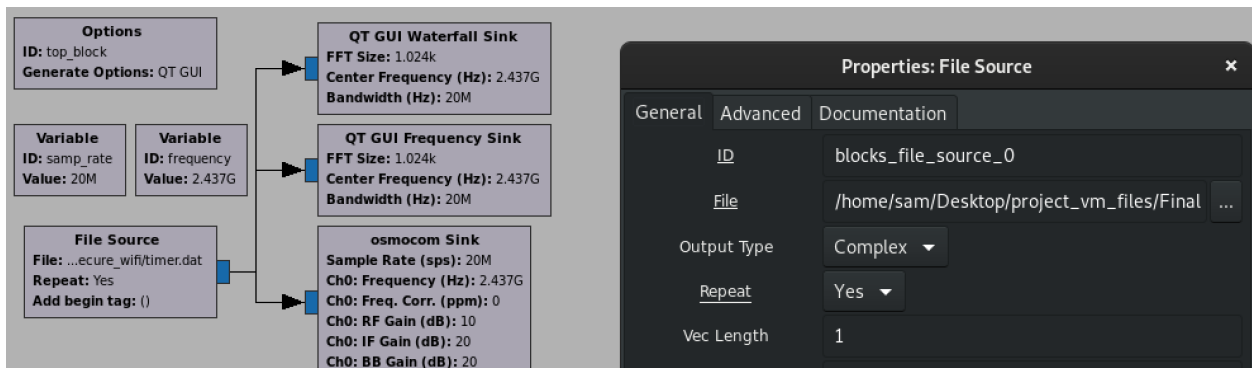


Figure 3.18: GNU Radio: Signal flow graph for replay attack

Jamming

This final attack is very simple, GNU Radio Companion will be used to transmit a large amount of random noise across the entire 20MHz frequency range in an attempt to cause disturbance such as the device failing to retrieve data or disconnecting from the network. While conducting this attack against the Echo Dot, it will play music from a radio station while the jamming noise signal is transmitted. Any lag in the radio playback will indicate potential disturbance in the device's connection. The GNU Radio Companion flow diagram is shown in figure

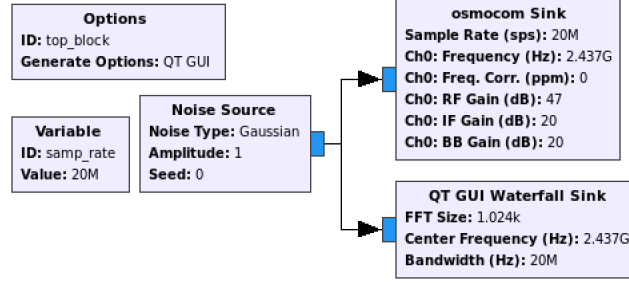


Figure 3.19: GNU Radio: Signal flow graph for jamming using Gaussian noise.

3.5.2 TP-Link HS100 UK WiFi Wall Plug

As with the Amazon Echo Dot, there will be similar attacks carried out against the TP-Link smart plug. Again, while the attacks are taking place, RF signals will be captured for analysis. Table 3.10 shows each attack with the corresponding time frame for the capture.

Attack	Time to Record Signal
Nmap Port Scan	10 Seconds
Deauthentication	10 Seconds
Replay Attack	N/A
Jamming	N/A
Toggle Command	5 Seconds.

Table 3.10: Attacks against the TP-Link Plug and the time to record RF signals.

Nmap Port Scan

The same Nmap port scan reconnaissance tool which was used against the Amazon Echo Dot will also be used against the TP-Link plug, the command will be the same except for the device IP address.

Deauthentication, Replay and Jamming Attacks

Both the deauthentication attack and the replay attacks which were done to the Amazon Echo Dot will also be done to the TP-Link plug, the only difference is that the commands being sent to the plug are different. It is also important to consider the commands being sent from the phone to trigger the on and off functions are being sent over the network local only, not over the internet. The attacks will also be carried out while connected to the secure access point, due to limitations and being unable to connect to the non-secure access point. This means that during the deauthentication attack, the access point MAC Address will be **32:5A:3A:5D:1C:88** instead of **30:5A:3A:5D:1C:88**. The jamming attack is also the same, to test for success the plug will be turned on and off via the Kasa app.

SI6 Networks IoT Toolkit

The SI6 Networks IoT Toolkit can be used to carry out some reconnaissance and basic attacks against the plug while on the same network. The first task will be to use the `iot-scan` tool to scan for the device on the network, this was done by using the command `sudo iot-scan -i wlan0 -Lcas` shown in figure 3.20. The `-i` flag was used to specify the wireless interface, in this case it was `wlan0`. The `-L` flag is used to specify “local network”, meaning the tool will scan the local subnet.

```
sam@kali:~/Desktop/iot-toolkit$ sudo iot-scan -i wlan0 -L
192.168.3.18 # IOT.SMARTPLUGSWITCH: TP-Link HS100(UK): Wi-Fi Smart Plug: "Plug"
```

Figure 3.20: IoT Toolkit: iot-scan tool scanning for TP-Link devices.

The next task was to use the iot-tl-plugin tool to get information from the device. This was done by using the command `sudo ./iot-tp-plugin -L -i wlan0 -c get-info -v` as shown in figure 3.21. As mentioned previously, the `-L` flag is used to specify the local network subnet and `-i` is used to specify the interface, which in this case is `wlan0`. The `-c` flag is used to specify a command which precedes it, in this case it is the `get-info` command, and finally the `-v` flag forced a verbose output.

```
sam@kali:~/Desktop/iot-toolkit$ sudo ./iot-tl-plugin -L -i wlan0 -c get_info -v
Got response from: 192.168.3.18, port 9999
```

Figure 3.21: IoT Toolkit: iot-tl-plugin tool get-info command.

The final attack will involve delivering a toggle command to the device using the same iot-tp-plugin tool as the previous attack. This is a payload which has already been constructed and is designed to toggle the device on and off repeatedly until the attacker stops the command from running in the terminal. This was done by using the command `iot-tl-plugin --toggle 192.168.3.18` as shown in figure 3.22, where the IP address in the command is the IP address of the TP-Link smart plug.

```
sam@kali:~/Desktop/iot-toolkit$ iot-tl-plugin --toggle 192.168.3.18
^C
```

Figure 3.22: IoT Toolkit: iot-tl-plugin tool toggle command.

3.5.3 Withings Bluetooth Blood Pressure Monitor

Due to this device operating on Bluetooth instead of WiFi there will be some differences. Firstly, the nmap scan will not be done against this device. Secondly, the deauthentication attack will actually be a DoS attack conducted with a different tool, although the objective is the same. There will also still be RF based attacks including replay attacks and jamming. The issue regarding these attacks is that the capture sample rate is only 20MHz, whereas the Bluetooth frequency range for the Blood Pressure Monitor is potentially 83MHz. This means a lot of data may be missed and it is less likely for some attacks to work. Table 3.11 shows each attack with the corresponding time frame for the capture.

Attack	Time to Record Signal
DoS (BlueSmack)	10 Seconds
Replay Attack	N/A
Jamming	N/A

Table 3.11: Attacks against the Withings Blood Pressure Monitor and the time to record RF signals.

BlueSmack/DoS Attack

Firstly, instead of a deauthentication attack, a similar goal can be achieved by conducting a common Bluetooth DoS (Denial of Service) attack known as the “BlueSmack” attack. As explained by [33], the

BlueSmack attack is where an attacker will use the L2Ping tool to send unusually large echo request packets to the victim Bluetooth device. Due to some devices being unable to handle packets over a certain size, the device may freeze and disconnect from the paired device. The end result is denial of service, as the Bluetooth device will no longer be connected or operational, this is different from the deauthentication attack as this attack is exploiting performance limitations within the Bluetooth protocol stack, whereas the deauthentication attack was telling devices to disconnect from the access point. This attack is also referred to as the “Ping of Death”, as it works in the same way, where an oversized ping request would cause operating systems or devices to crash.

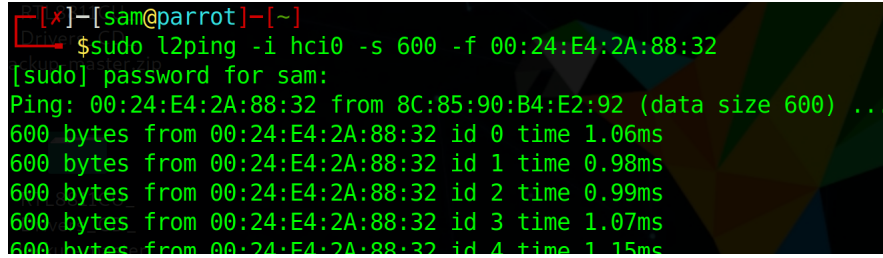
Due to unforeseen issues with the Kali Linux virtual machine, the Parrot Linux OS virtual machine was used for this attack instead. To start this attack, it was necessary to carry out some basic reconnaissance by using the Bluetoothctl utility to begin discovering devices. This was done by using the command-line tool **Bluetoothctl** followed by **scan on** as shown in figure 3.23. The scan will show the devices along with their MAC Address. To carry out the attack, the MAC address for the blood pressure monitor is needed so that the target address can be specified in the l2ping command. Figure 3.23 shows the device “AnonP”, which is the iPhone in which the blood pressure monitor is paired to, followed by the “Withings BPM 32” which is the blood pressure monitor. Once the MAC addresses were discovered, the command **l2ping -i hci0 -s 600 -f 00:24:E4:2A:88:32** was used to send the large L2CAP echo requests to the blood pressure monitor as shown in figure 3.24. Here, the **-i** flag is used to specify the Bluetooth adaptor ID which in this case, is **hci0**. The **-s** flag specifies the size of the packets to send, which in this case is 600 bytes, according to [33] this should be large enough for a successful DoS attack. Next, the **-f** flag is used to specify “flood”, meaning the delay between transmission is set to 0. This is then followed by the target device MAC address.

```
[sam@parrot]~$ sudo bluetoothctl
[sudo] password for sam:
Agent registered
[bluetooth]# scan on
Discovery started
[CHG] Controller 8C:85:90:B4:E2:92 Discovering: yes
[NEW] Device DC:A9:04:E2:C4:F7 DC-A9-04-E2-C4-F7
[CHG] Device DC:A9:04:E2:C4:F7 LegacyPairing: no
[CHG] Device DC:A9:04:E2:C4:F7 RSSI: 127
[CHG] Device DC:A9:04:E2:C4:F7 Name: AnonP
[CHG] Device DC:A9:04:E2:C4:F7 Alias: AnonP
[CHG] Device DC:A9:04:E2:C4:F7 LegacyPairing: yes
[CHG] Device DC:A9:04:E2:C4:F7 RSSI is nil
[NEW] Device 00:24:E4:2A:88:32 00-24-E4-2A-88-32
[CHG] Device 00:24:E4:2A:88:32 LegacyPairing: no
[CHG] Device 00:24:E4:2A:88:32 RSSI: 127
[CHG] Device 00:24:E4:2A:88:32 Name: Withings BPM 32
[CHG] Device 00:24:E4:2A:88:32 Alias: Withings BPM 32
```

Figure 3.23: Bluetoothctl scan showing devices and corresponding MAC Address.

Replay Attack

The replay attack will follow a similar method to the previous devices, where the actions mentioned in table 3.8 will be carried out and recorded with the HackRF via GNU Radio Companion using the flow graph in figure 3.8 and replayed using the flow graph seen in figure 3.18 (with the exception of the frequency being changed to 2442MHz in both flow graphs). The full blood pressure test capture will be replayed in an attempt to replay the same blood pressure data back to the phone, meaning repeated data should be recorded by the Withings Health Mate app. Starting the blood pressure test is a quick command and is more likely to be successful as it requires less data to be sent.

A terminal window with a dark background and green text. The prompt is [x]-[sam@parrot]-[~]. The command \$sudo l2ping -i hci0 -s 600 -f 00:24:E4:2A:88:32 is entered. The prompt changes to [sudo] password for sam:. The output shows a ping to 00:24:E4:2A:88:32 from 8C:85:90:B4:E2:92 (data size 600) ... followed by five lines of results: 600 bytes from 00:24:E4:2A:88:32 id 0 time 1.06ms, 600 bytes from 00:24:E4:2A:88:32 id 1 time 0.98ms, 600 bytes from 00:24:E4:2A:88:32 id 2 time 0.99ms, 600 bytes from 00:24:E4:2A:88:32 id 3 time 1.07ms, and 600 bytes from 00:24:E4:2A:88:32 id 4 time 1.15ms.

```
[x]-[sam@parrot]-[~]  
$sudo l2ping -i hci0 -s 600 -f 00:24:E4:2A:88:32  
[sudo] password for sam:  
Ping: 00:24:E4:2A:88:32 from 8C:85:90:B4:E2:92 (data size 600) ...  
600 bytes from 00:24:E4:2A:88:32 id 0 time 1.06ms  
600 bytes from 00:24:E4:2A:88:32 id 1 time 0.98ms  
600 bytes from 00:24:E4:2A:88:32 id 2 time 0.99ms  
600 bytes from 00:24:E4:2A:88:32 id 3 time 1.07ms  
600 bytes from 00:24:E4:2A:88:32 id 4 time 1.15ms
```

Figure 3.24: L2ping sending 600 byte packets to Withings BPM

Jamming Attack

The jamming attack will also be attempted using the same method as the previous devices as shown in figure 3.19, however this will be done on the centre frequency of 2442MHz. As mentioned previously, the fact that the frequency range is potentially across 83MHz, jamming 20MHz of bandwidth will likely be ineffective, the device will likely continue operating as normal outside of those frequencies. This is in fact part of the design, where Bluetooth devices will frequency hop to avoid interference.

Chapter 4

Results

4.1 Environment

There were RF captures done in the home network environment and the lab network environment. These captures were recorded for 5 seconds. The results from viewing the captures in Universal Radio Hacker shows that there is evidently a lot more noise in the lab environment compared to the home environment as expected. There does appear to be more Bluetooth frequency (2442MHz) noise in the home environment compared to channel 6 WiFi (2437MHz). These results are shown in figure 4.1. Universal radio hacker failed to auto-detect the parameters as shown on the left in figure 4.1 due to the insignificance of the data, it did however classify it as mostly noise. The results of testing the Faraday cage within Universal Radio Hacker was difficult to determine due to the y axis not scaling well, however the python script “checkCaptures.py” shows that there were less recorded values using the Faraday cage and majority of those values occurred more than once, meaning the weak signals were blocked out, compared to the normal noise capture where almost half of the values in the whole file only occurred once. This is shown in the comparison of figures 4.2a to 4.2c. These results show that in business environments there is a good chance of high noise which could cause difficulty detecting small attacks simply due to the amount of data being transmitted.

4.2 Amazon Echo Dot

4.2.1 Usage

During general usage there were some interesting observations in the captured RF signals. As expected, the tests which were done while connected to the secured network were very different and did not show any interesting results or patterns, this was the case for both versions of the Echo Dot. However during the tests on the non-secured network, some patterns and repetitive data was found in some of the same captures. For example, the test where Alexa was asked the voice command “What is the weather like in Cardiff?”, there was a long string of values which had been repeated once in the same capture after demodulation. It is typical to see short values being repeated, however in this case it was 703 bits which seemed unusual and definitely stands out. This could be benign data, or could be part of unencrypted communication such as analytics. This is shown in figure 4.3 represented in hexadecimal format. If this is a value commonly repeated and associated with the Echo Dot, this could be used to identify the Echo Dot device in an RF capture, which may be useful to attackers for carrying out RF based reconnaissance or as a way of fingerprinting the device signals.

4.2.2 Nmap Port Scan

The results from the nmap scan against the Amazon Echo Dot showed that there were no open ports which were part of the top 1000 most common port numbers. The results are shown in figure 4.4. The recorded RF signals also show signs of scan activity due to the high volume of repetitive looking signals. After

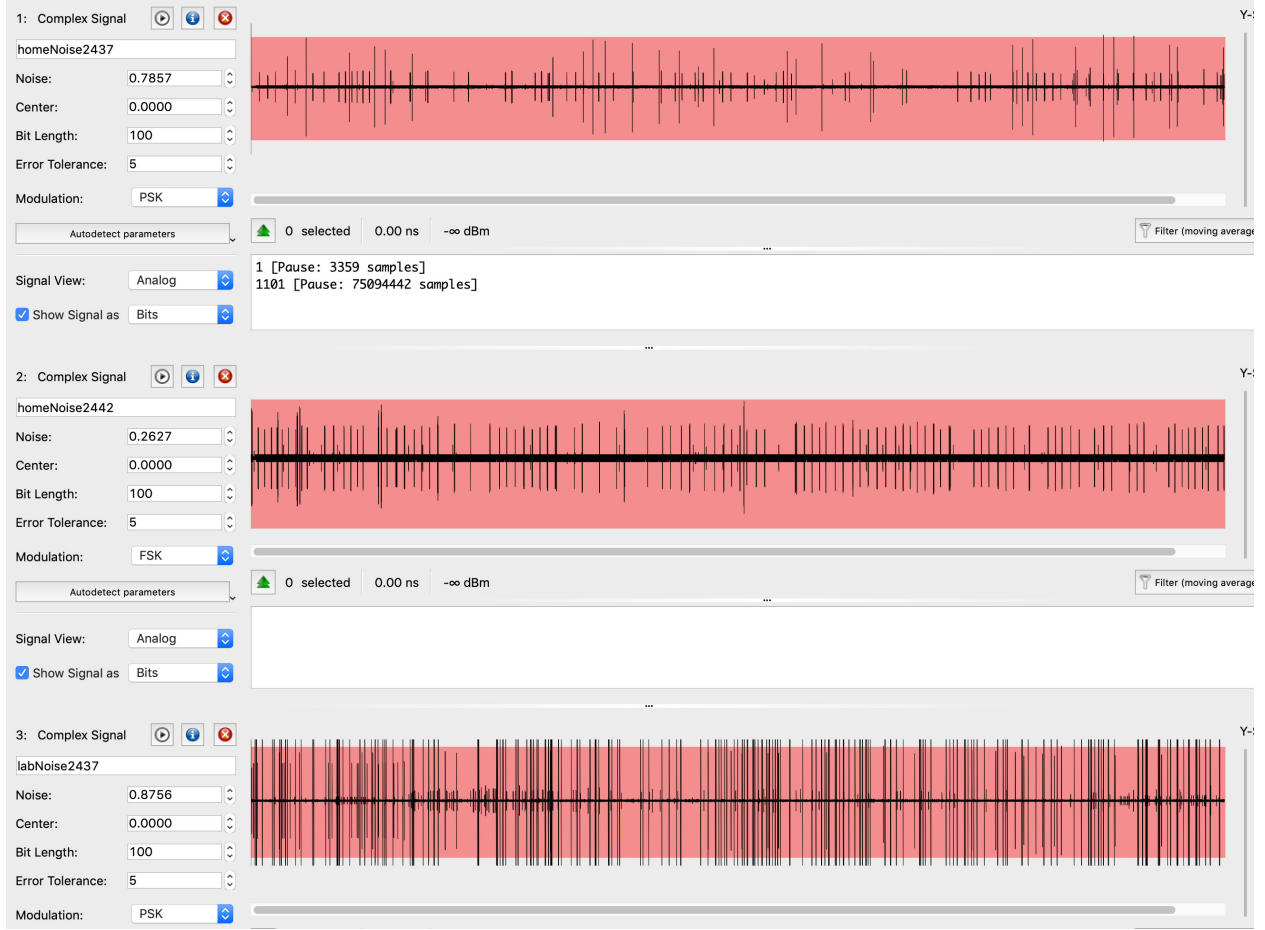


Figure 4.1: RF Signals from home and lab environment

Filename: homeNoise2437.dat	Filename: homeNoise2437_faraday.dat	Filename: labNoise2437.dat
Total No. of values: 70306755	Total No. of values: 32756678	Total No. of values: 112702824
No. of unique values (counted): 22997	No. of unique values (counted): 2824	No. of unique values (counted): 59873
No. of values which occur > once: 12289	No. of values which occur > once: 2417	No. of values which occur > once: 49944
Top 10 values and their count:	Top 10 values and their count:	Top 10 values and their count:
1.0842024178528004e-19: 17861772	1.0842024178528004e-19: 8835555	1.0842024178528004e-19: 35276792
1.0842024188625424e-19: 11069131	1.0842024188625424e-19: 5614378	1.0842024188625424e-19: 17528274
5.03559548e-315: 10212401	5.03559548e-315: 4975430	5.03559548e-315: 16157954
1.0842024168430584e-19: 6328810	2.77558189703169e-17: 2956479	1.0842024168430584e-19: 9102327
2.77558189703169e-17: 6040154	1.0842024168430584e-19: 2895489	2.77558189703169e-17: 8544225
5.056318096e-315: 5486984	5.056318096e-315: 2437453	5.056318096e-315: 7631707
2.775581922881084e-17: 3189253	2.775581922881084e-17: 1402761	2.775581922881084e-17: 4221553
5.014872865e-315: 3040701	5.014872865e-315: 1131254	5.014872865e-315: 4040030
2.775581871182296e-17: 1628562	2.775581871182296e-17: 558613	2.775581871182296e-17: 2060543
1.0842024193674133e-19: 713500	1.0842024193674133e-19: 173407	1.0842024193674133e-19: 885601
(a) homeNoise2437.dat	(b) homeNoise2437(Faraday).dat	(c) labNoise2437.dat

Figure 4.2: Script comparing environment captures.



Figure 4.3: Large string of values being repeated by Echo Dot on Open WiFi

```

sam@kali:~$ nmap 192.168.3.19 -Pn --top-ports 1000 -v
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-09 00:19 EDT
Initiating Parallel DNS resolution of 1 host. at 00:19
Completed Parallel DNS resolution of 1 host. at 00:19, 13.01s elapsed
Initiating Connect Scan at 00:19
Scanning 192.168.3.19 [1000 ports]
Completed Connect Scan at 00:19, 18.43s elapsed (1000 total ports)
Nmap scan report for 192.168.3.19
Host is up (0.020s latency).
All 1000 scanned ports on 192.168.3.19 are filtered (960) or closed (40)

```

Figure 4.4: Result of nmap port scan against Amazon Echo Dot

demodulation there appears to be some values being repetitive, especially the value “80”. These results are shown in figure 4.5.

4.2.3 Deauthentication

The deauthentication attack against the Echo Dot was successful. The device remained disconnected from the network while the attack continued, it then re-connected back to the network shortly after the attack was stopped. The figure 4.6 shows the device with the red ring after being asked a command, indicating it was not connected to the network. As with the nmap scan, the RF captures for the deauthentication attack also clearly indicate the attack took place. As shown in figure 4.7, repetitive high volume bursts can be seen in consistent intervals, the demodulated data also indicates some repetition, where the values often begin with “ff” followed by relatively consistent pauses.

4.2.4 Replay attack

The replay attack against the Echo Dot was unsuccessful after attempting to replay every command detailed in the methodology while connected to the secure and non-secure network, this includes attempting the attack after using the “Alexa” wake word and ensuring the phone sending the volume adjustment commands was also connected to the non-secure network. However, when replaying the signals captured from the deauthentication attack seen in figure 4.7, the device did in fact disconnect from the network, indicating that replaying the RF signals for the disassociation packets was successful. This could be due to the fact that

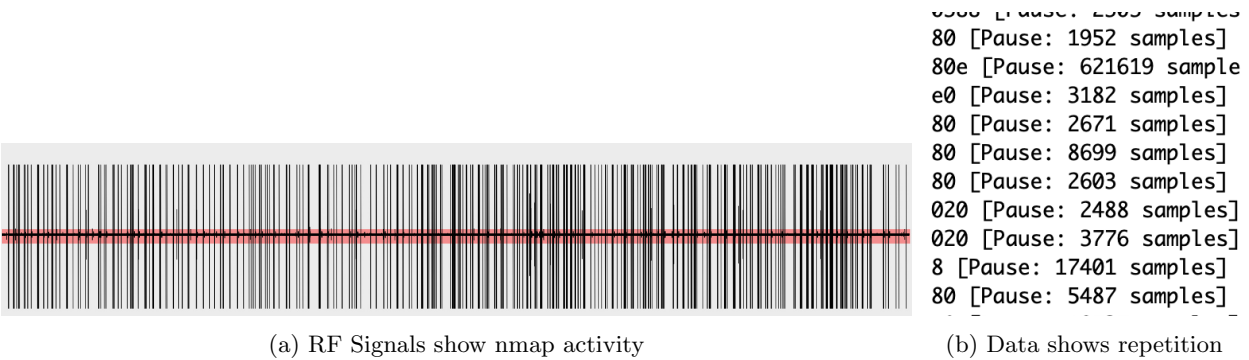


Figure 4.5: Nmap port scan: Echo Dot

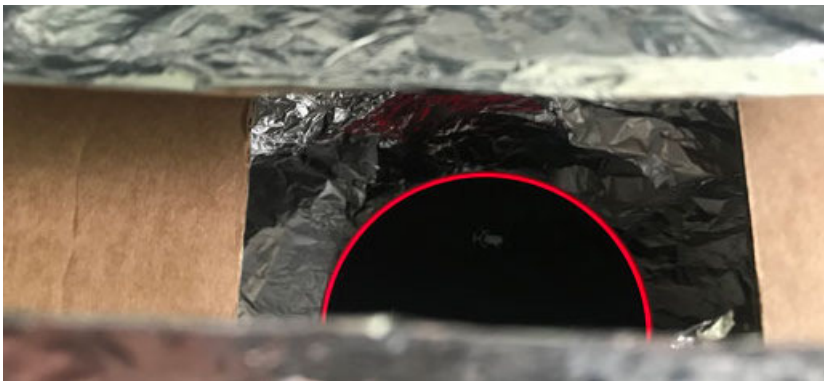


Figure 4.6: Echo Dot successful dauthentication

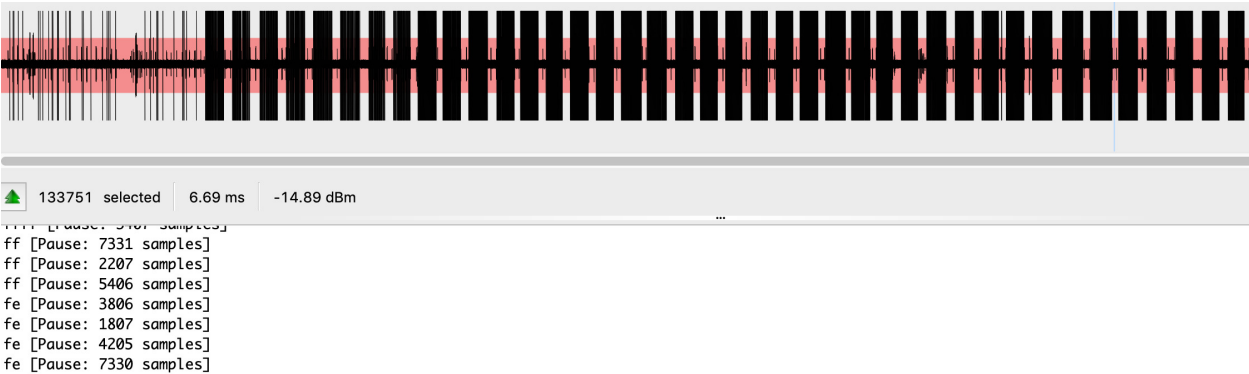


Figure 4.7: RF Signals show evidence of deauthentication attack (Echo Dot)

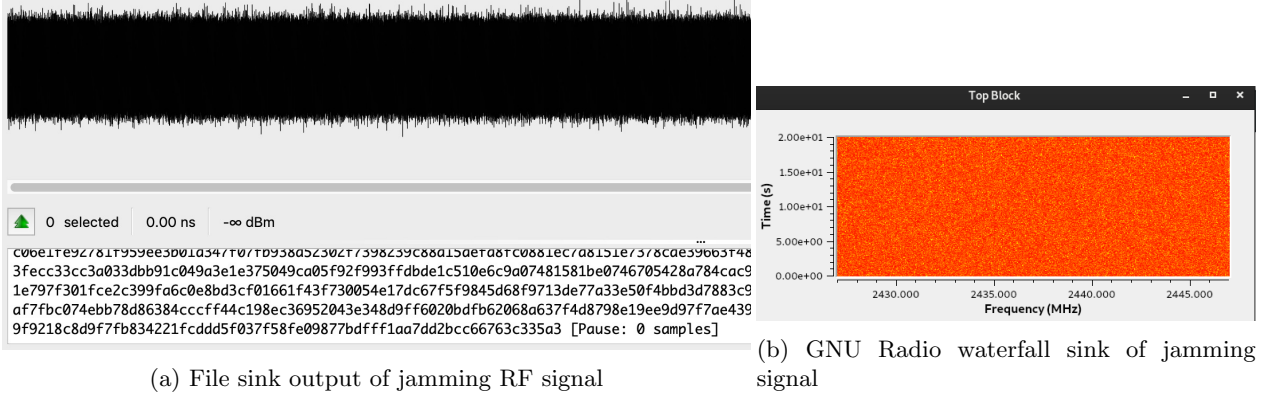


Figure 4.8: RF Signals produced by the Gaussian noise jamming attack.

the deauthentication attack works on a lower layer compared to general usage of the device at the application layer. Replaying the deauthentication attack will only work against the target device which was originally targeted in the captured deauthentication attack, this is due to the attack targeting a specific MAC address. A deauthentication attack can be done against all clients on a network, however this is often not as effective as targeting a specific device address.

4.2.5 Jamming

The jamming attack against the Echo Dot was also unsuccessful. It appears the noise source does not create enough interference to interrupt a signal. The attack may require adjustment including the gain settings for the HackRF, however due to time and hardware constraints the attack was not continued. For demonstration purposes, a waterfall sink was added to the GNU Radio Companion flow graph along with a file sink that the file can be analysed in Universal Radio Hacker. The figure 4.8 shows the output RF signals generated by the jamming attack. Figure 4.8a shows the random data being sent without pause.

4.3 TP-Link HS100 UK WiFi Wall Plug

4.3.1 Usage

General usage of the plug involved turning it on and off using the Kasa app. The plug had to be connected to the secure network due to limitations with the device and applications as explained in the methodology. The limitations involved the device or app not allowing users to configure it to connect to an open network, when attempting to do so, a message will appear detailing why it is not allowed. This was an interesting result and was not expected. An older version of the app was downloaded for android as explained previously but this still did not work, the application would force the user to chose which security protocol the access point uses and there was no option to chose an open network. A final attempt was made by connecting the device to a secured network and removing the security afterwards, however the device disconnected and refused to re-connect. These app screen shots can be seen in figure 4.9. 3 captures were taken using the device as normal by turning it on and off. The captures, as explained previously, were around 5 seconds long. They did appear to have some visual similarities but demodulated it seems different enough to prove the data is encrypted. The results from these are shown in Universal Radio Hacker in figure 4.10.

4.3.2 Nmap Port Scan

The results from the nmap port scan against the tp-link smart plug indicates port 9999 is open, which already explained in the SI6 Networks presentation [29] regarding attacks against this device. This port is used by the Kasa app to discover, connect and control the device on the network. The results from the nmap

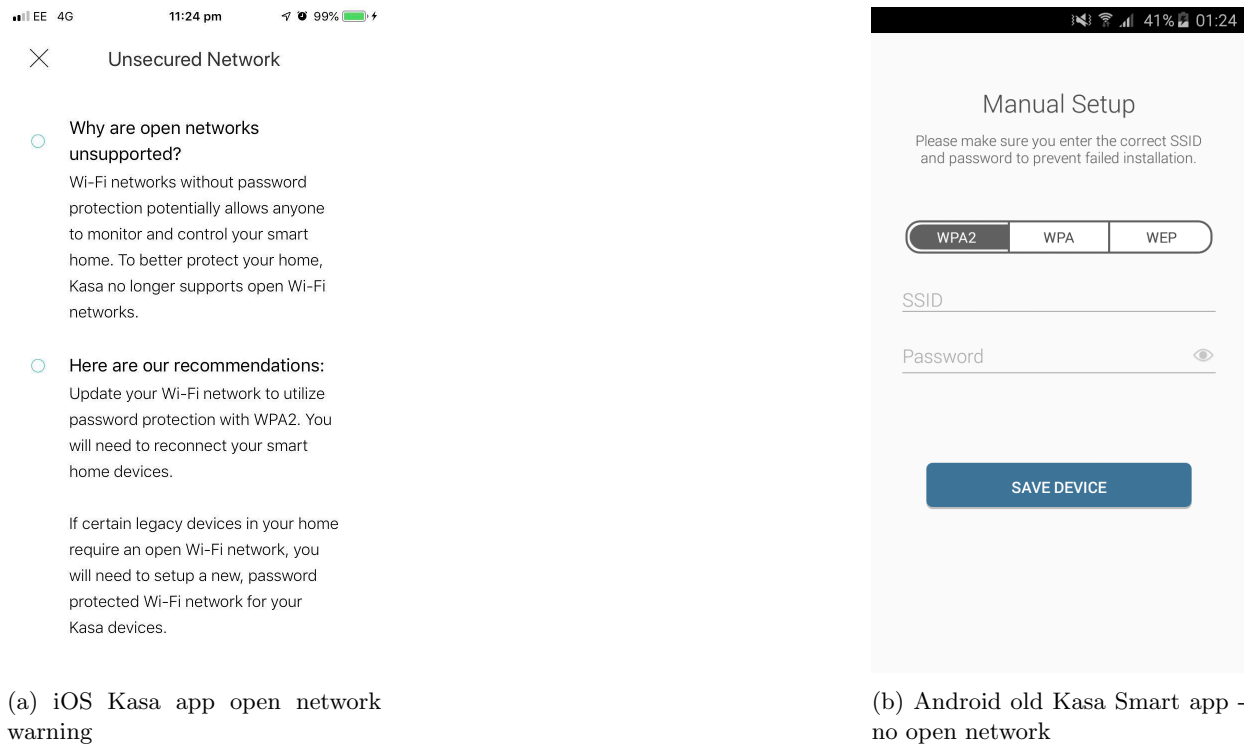


Figure 4.9: TP-Link Kasa app does not allow open WiFi networks

port scan can be seen in figure 4.11. The captured RF signals are very similar to the ones seen captured while scanning the Echo Dot, even though this device was connected to the secure network, the volume and consistency of the signals and pauses were very similar.

4.3.3 Deauthentication

The deauthentication attack against the plug was also successful, the same result as the Echo Dot. The plug disconnected from the network and remained disconnected until the attack was stopped. This was evident by the flashing LED on the WiFi logo on the device indicating it is attempting to establish a connection. This can be seen in figure 4.13. The captured RF signals were also very similar and consistent as with the results from attacking the Echo Dot. Although this was done on a secure network instead of the open network, the data looks very similar and the demodulated data also showed the same values beginning with “f” being repeated with consistent pauses as shown in figure 4.12.

4.3.4 Replay attack

The play attacks against the smart plug were unsuccessful as with the Echo Dot, this included repeating the captures 3 times which recorded signals from the Kasa app turning the device on and off and then replaying the toggle command signals recorded from the IoT toolkit during the toggle attack. As with the Echo Dot, replaying the signals from the deauthentication attack also worked against the smart plug, the plug disconnected and remained disconnected while the RF signals were being repeated.

4.3.5 Jamming

The outcome from the jamming attack was the same as when tested against the Echo Dot, broadcasting the Gaussian noise did not disrupt or delay the ability to turn the device on and off via the Kasa app.



Figure 4.13: Deauthentication attack successful (TP-Link HS100)

4.3.6 SI6 Network's IoT Toolkit

The results from using the IoT toolkit against the smart plug showed that the **get-info** command was able to return a large JSON response containing technical information including device type, model, IP address, MAC address, hardware version, lat/long coordinates and much more. This is shown in figure 4.14. The toggle attack was also successful, there was no verbose output from this result to indicate success, the only indication was the plug turning on and off repeatedly. The RF signals captured from the attack does show some repetitive signals as shown in figure 4.15.

```
sam@kali:~/Desktop/iot-toolkit$ sudo ./iot-tl-plug -L -i wlan0 -c get_info -v
Got response from: 192.168.3.18, port 9999
{"system":{"get_sysinfo":{"err_code":0,"sw_ver":"1.2.5 Build 171213 Rel.101014","hw_ver":"1.0","type":"IOT.SMARTPLUGSWITCH",
"model":"HS100(UK)","mac":"50:C7:BF:66:99:2E","deviceId":"8006CD365E08D86A1843292F90BBBA18186A9B67","hwId":"522CBD4857687F1A
E7D7EE9B440BF26B","fwId":"00000000000000000000000000000000","oemId":"9F26B8A5A0266993C796D0BF593B8415","alias":"Plug","dev_n
ame":"Wi-Fi Smart Plug","icon_hash":"","relay_state":1,"on_time":116,"active_mode":"schedule","feature":"TIM","updating":0,"
rssi":-86,"led_off":0,"latitude":0,"longitude":0}},"emeter":{"err_code":-1,"err_msg":"module not support"}}
```

Figure 4.14: SI6 IoT Toolkit get-info command results show JSON data

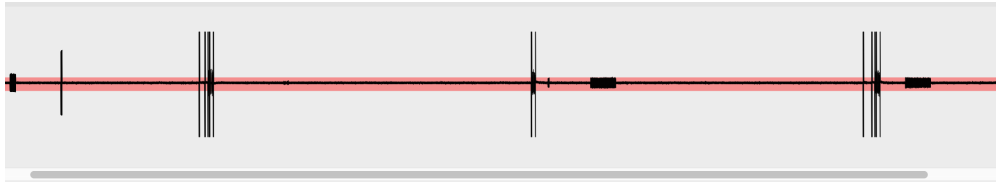


Figure 4.15: RF Signals from toggle attack

4.4 Withings Bluetooth Blood Pressure Monitor

4.4.1 Usage

The usage results from the blood pressure monitor were not very conclusive, as mentioned previously there was a risk that only part of the data would be captured by the HackRF due to the maximum support sample rate of 20MHz. The waterfall output from GQRX used to monitor the signals each time the device turned on showed signals hopping across bandwidth instead of remaining on or close to the 2442 centre frequency. This indicates there is a chance that signals are being sent outside the 20MHz bandwidth monitored by the HackRF due to the blood pressure monitor operating over an 83MHz range. The output from GQRX can be seen in figure 4.16. The signals recorded from starting the blood pressure test were difficult to analyse as the signals appear to be very fragmented and inconsistent, meaning visually there appears to be some patterns although this could be a combination of partial data being captured and noise. The demodulated data appears to only show similar values such as "ffffffc" being repeated with inconsistent pauses, this

could be due to the data not being demodulated properly or being incomplete. The modulation type for the Bluetooth communication is different to the WiFi devices which were tested, the blood pressure monitor uses FSK modulation whereas the Echo Dot and TP-Link plug uses a variant of PSK which is also multiplexed. Universal Radio Hacker did auto-detect the signals as being FSK modulation. The RF Signals captured are shown in figure 4.17.

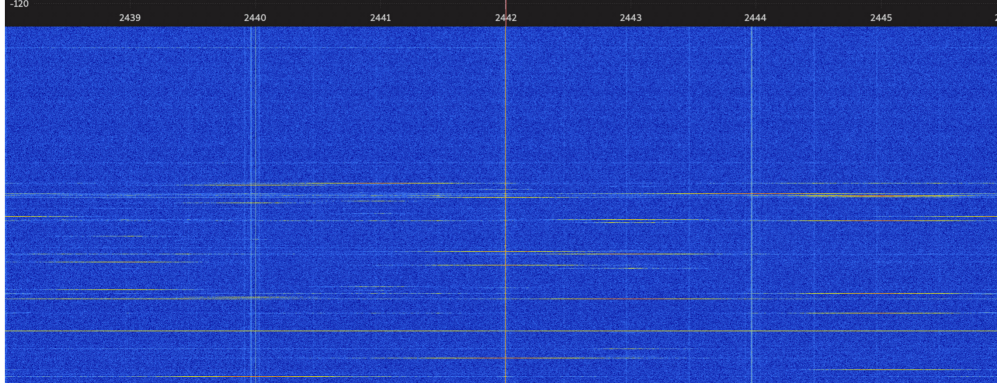


Figure 4.16: GQRX Waterfall showing frequency shifting/hopping

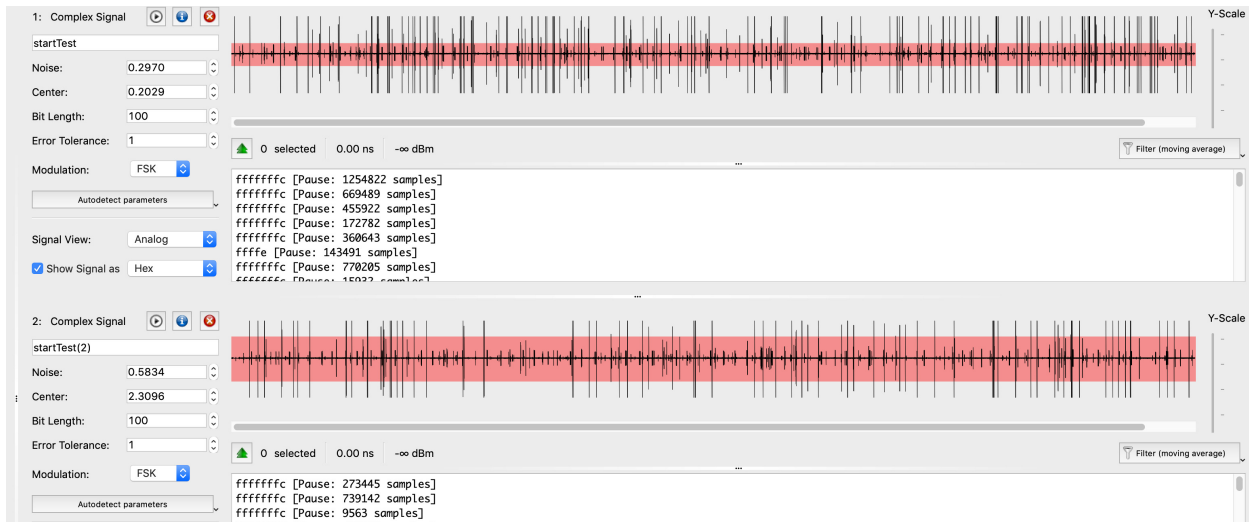


Figure 4.17: RF Signals from starting a blood pressure test

4.4.2 BlueSmack/DoS Attack

The BlueSmack DoS attack against the blood pressure monitor was successful, this was tested while connected to the device and a second time while completing a full blood pressure test. Once the L2Ping requests started sending, the device immediately stopped inflating and disconnected from the phone. The Withings Health Mate app gave instructions to turn the device on even though it was still on. While trying to reconnect to the device using the iPhone settings, it could not reconnect (as shown in figure 4.18a). The Bluetoothctl utility also showed the device being removed after it was discovered as shown in figure 4.18b.

The RF Signals which were captured during this attack are shown in figure 4.19. These signals show evidence of high traffic volume in one particular area, however this does not continue throughout the capture. This could be for a number of reasons; firstly it could be due to the target requests being sent outside of the

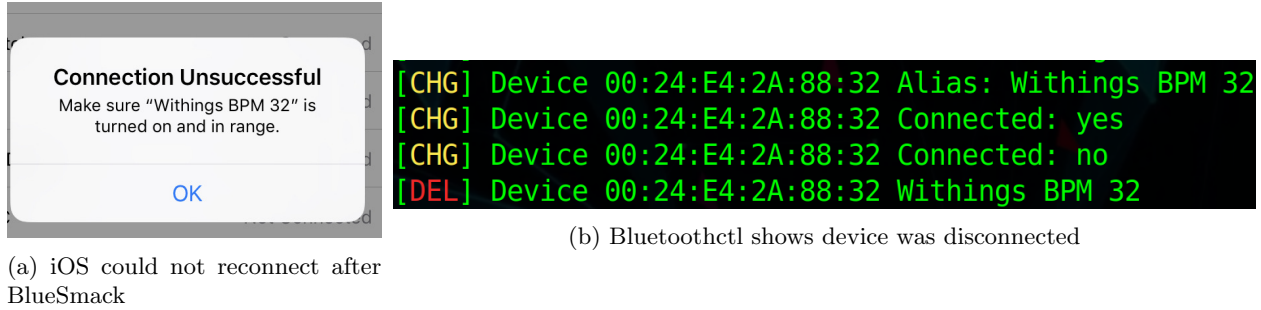


Figure 4.18: Withings BP Monitor disconnects after BlueSmack/DoS

20MHz sample rate and only a portion of the attack was captured, secondly it could be due to the packets not being sent as the device disconnected in time, however the L2ping tool showed the packets as being sent without stopping. Lastly, it could be that the high volume of RF signals captured at that time actually has nothing to do with the attack and can be considered as benign.

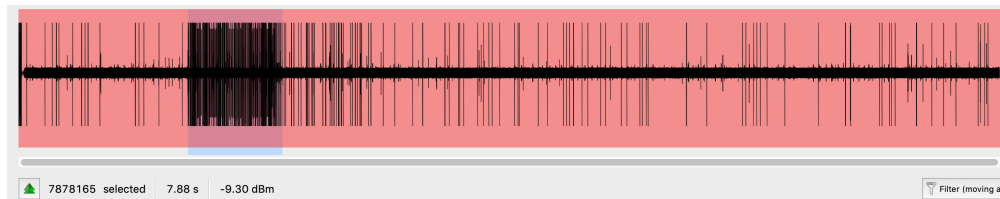


Figure 4.19: RF Signals from BlueSmack/DoS attack

4.4.3 Replay Attack

As with the previous devices, each replay attack involving replaying a command or data was unsuccessful as the device did not respond to starting the blood pressure test and replaying blood pressure test results being sent to the phone did not change the results the second time. Replaying the RF signals captured from the BlueSmack DoS attack was also unsuccessful. This could be due to only a small portion of the attack being captured shown in figure 4.19.

4.4.4 Jamming Attack

The Jamming attack using the Gaussian noise transmitted on the 2442MHz centre frequency was also unsuccessful, the noise needs to be transmitted across a larger range in order to disrupt the Bluetooth signals for the blood pressure monitor. Due to the results mentioned previously regarding frequency hopping and operating on a wide range, 20MHz of noise is not enough for this attack as the device remained fully operational including taking a full blood pressure test while carrying out the attack.

Chapter 5

Future Work

To improve on this project it would be ideal to test more devices against a larger range of attacks. For example, TP-Link have a range of devices which use the same Kasa app to communicate including cameras and light bulbs which would be interesting to investigate. Usually a smart home has more than one smart device, and they are often linked together to create a smart ecosystem, therefore linking the devices together to carry out other usage scenarios would be more realistic and provide more useful data. Some of the smart devices can also be controlled by the Amazon Echo Dot by using third party Alexa skills [34] and telling Alexa to control the device (such as turn on a light), it would be interesting to investigate the communication between these two devices over the local network. Additionally it would be useful to try and obtain IoT smart devices which communicate on other protocols other than WiFi and Bluetooth. Some devices may still use lower frequency protocols to communicate over shorter distances, for example a remote control operating on a 433MHz frequency, in these scenarios there is less security and less data to capture meaning replay attacks can be much easier to carry out. There are also other wireless protocols such as Zigbee, according to [35], Zigbee uses much shorter frequency bandwidth for their channels meaning capturing the data will require a smaller sample rate than 20MHz which also means smaller file sizes for recording data and attempting RF based attacks.

Further improvement would also include further investigation into fingerprinting the devices based on the data recorded, this could include looking at existing fingerprinting algorithms, including those used in multimedia comparison such as Shazam's fingerprinting algorithm. Another option would be possibly converting the RF I/Q Values to unique pixels to create an image, where image comparison and fingerprinting techniques could be used to identify possible attacks. Repeating frequency values such as those seen in the attacks could result in an image with a high number of pixels of a specific colour.

Further attacks could also be carried out, including device specific exploits, password brute force attacks either over SSH or a web server running a device. None of the devices tested in this project had a web server with an accessible login page, therefore it would be good to try attacks against these as the web servers are the most likely thing to get attacked if found by someone on the network.

Chapter 6

Conclusion

One of the aims of this thesis was to investigate the radio frequency signals produced by IoT devices during usage and analyse those signals for signs of patterns before progressing onto carrying out several attacks against each device. The devices were used in two different environments; the Cardiff University COMSC IoT Lab and a home network which has been configured for these experiments, this includes a secured and non-secured access point. The non-secured access point ensured the communication is not encrypted at layer 2 for additional analysis. A custom made Faraday cage was also used to reduce background noise in the signals, this Faraday cage was successful and was evident when analysing and comparing the background noise captured from each environment. The IoT devices which were used include two Amazon Echo Dots (2nd Generation) running different firmware versions, a TP-Link HS100 WiFi Smart Plug and a Withings Bluetooth Blood Pressure Monitor. The purpose of the attacks against these devices was to not only investigate whether the devices are vulnerable to certain attacks but to also record the RF signals for analysis and understand how the devices react to the attacks. Only some attacks were successful, it appeared that the replay attacks involving replaying normal usage did not work regardless of the device and what network they were connected to, however the deauthentication and DoS attacks were successful, and replay attacks involving replaying the deauthentication and DoS attacks were also successful. This means that IoT devices generally do not cope well with lower level attacks. The attacks against the TP-Link device using the IoT Toolkit was successful as expected, as there was already documentation outlining the success of the attacks from this toolkit.

Overall, after analysing the RF signal captures it appears they show obvious signs of attack due to the volume and pattern of signals observed. Although the nmap scan and the deauthentication attack generate high volumes of traffic, the RF signals appeared to be very different but still show signs of attack. This indicates that it may be possible to detect these attacks on the RF layer. This could lead to development of an RF based intrusion detection system that could detect attacks and anomalies within typical RF signals.

Chapter 7

Reflection

From the start I was enthusiastic about this project as I was hoping to do something IoT based. I have never done anything regarding radio frequencies or used a HackRF one before, although I heard about the research and attacks being done in that field. I was looking forward to taking something on that was relatively new for me which also involved carrying out attacks which I was already familiar with but never had the opportunity to explore how they impact IoT devices. After speaking with my supervisor about the initial proposal and discussing the options of approach to the project I was keen to get started. I never used any of the smart devices before personally, which is why it was useful to have the devices straight away so that I had time to set them up, learn how they work and understand what kind of actions I could capture as RF signals and what attacks could be carried out. Being in the security field and working as a security analyst part time meant that I had some existing technical skills and knowledge of what tools I could use and where to look for help when needed. I also already had the Kali Linux and Parrot OS virtual machines setup with snapshots, therefore there was less work involved getting setup, I also already had existing network knowledge and the hardware necessary to make an isolated network at home with an open and secured access point. Initially it skipped my mind that traffic on a secure network is encrypted which would mean the results in the RF signals would potentially be different, so the open network was setup a little while after the I started the project.

Due to my limitations in knowledge regarding radio frequency analysis I had to learn more about how the HackRF works, how radio signals work in general, how data is modulated and sent over WiFi and how it can affect the results. Most information online and tutorials appeared to cover lower frequency devices but nothing over 1GHz, this meant that it was difficult for me to analyse the data initially as the signals are far more dense and carry more data. Some attacks involving lower frequency hardware involved demodulating the data by eye bit-for-bit, which appeared to be impossible with the 2.4GHz signals. Due to time constraints and lack of knowledge in the area I didn't fully understand how the OFDM (orthogonal frequency-division multiplexing) [36] modulation worked, however the FCC reports for the devices mentioned a variant of PSK being used for modulation, therefore I chose to use PSK demodulation during the analysis of the RF signals. Initially I began capturing signals using a sample rate set to 2 Million, however later I discovered this meant I was only capturing 2MHz bandwidth of signals, meaning most of the signals were not being captured, once I discovered this I re-done all of the captures using a 20M sample rate which was the maximum the HackRF would support. This issue then led me to discover the complications with the Bluetooth devices operating across a 83MHz frequency range and how that would affect my captures, attacks and results.

My initial plan was to try and develop a way to fingerprint devices or identify attacks using similar methods to Shazam, however after researching the topic and attempting to understand how their algorithm works at a level where it could be adapted to suit my project I decided it was not something I was interested in pursuing in this project due to skill and time limitations. I used my initiative to leave it behind and focus on carrying out attacks and gathering data before attempting to come back to the fingerprinting idea afterwards if I had time. There was also the option of attempting machine learning with my python script to use the raw I/Q values to detect attacks and classify the data, however due to me not having a lot of knowledge and skills in implementing machine learning applications I decided this would also be put aside

until there is time at the end.

Regarding my overall performance throughout the project, I regret not testing more devices in the IoT lab and gathering more data, including devices such as a TP-Link Camera which would have been a good comparison against the smart plug, however due to the setup at the lab and technical issues trying to setup the device for myself to use while testing, I decided to focus on the devices I had at the time instead. While doing the investigations and recording the signals I made sure to organise the captures in sub folders for each device, name the files according to what action they correspond with, and take notes for how long the capture took in seconds. Being organised and having the notes for each capture along with screen shots of the attacks being carried out made it easier later on when writing the report. I learnt a lot during this project, my technical skills in a variety of areas have improved including radio signal frequencies and processing, attack techniques, reading I/Q data in python and some technical knowledge in the multimedia field regarding image and audio fingerprinting. I enjoyed the project as a whole and I was intrigued by some of the results, it was interesting carrying out attacks in a more realistic scenario and understanding how these smart devices were effected.

Bibliography

- [1] *A Guide to the Internet of Things Infographic*. URL: <https://www.intel.co.uk/content/www/uk/en/internet-of-things/infographics/guide-to-iot.html> (visited on 02/04/2019).
- [2] [1702.03681] *Turning Internet of Things(IoT) into Internet of Vulnerabilities (IoV) : IoT Botnets*. URL: <https://arxiv.org/abs/1702.03681> (visited on 10/05/2019).
- [3] *Researchers show Alexa “skill squatting” could hijack voice commands — Ars Technica*. URL: <https://arstechnica.com/information-technology/2018/08/researchers-show-alexa-skill-squatting-could-hijack-voice-commands/> (visited on 10/05/2019).
- [4] *PowerPoint Presentation*. URL: https://www.owasp.org/images/5/5e/OWASP201604_Drones.pdf (visited on 10/05/2019).
- [5] *Hacking Car Key Fobs with SDR - Lufsec*. URL: <https://www.lufsec.com/hacking-car-key-fobs-with-sdr/> (visited on 10/05/2019).
- [6] *consolecowboys: Hacking Everything with RF and Software Defined Radio - Part 1*. URL: <http://console-cowboys.blogspot.com/2017/10/hacking-everything-with-rf-and-software.html> (visited on 10/05/2019).
- [7] *Altocumulous Digital Media Receiver 2045 FCC ID 2AHSE-2045*. URL: <https://fccid.io/2AHSE-2045> (visited on 02/05/2019).
- [8] *HS100 — Kasa Smart Wi-Fi Plug — TP-Link United Kingdom*. URL: <https://www.tp-link.com/uk/home-networking/smart-plug/hs100/> (visited on 02/05/2019).
- [9] *FCC ID TE7HS100*. URL: <https://fcc.report/FCC-ID/TE7HS100> (visited on 03/05/2019).
- [10] *Health Tracker App — Fitness Tracker — Withings Health Mate*. URL: <https://www.withings.com/uk/en/health-mate> (visited on 01/05/2019).
- [11] *Wireless Blood Pressure Monitor - BPM — Withings*. URL: <https://www.withings.com/uk/en/blood-pressure-monitor> (visited on 02/05/2019).
- [12] *WPM02 Blood Pressure Monitor Test Report Withings*. URL: <https://fccid.io/XNAWPM02/Test-Report/Test-Report-2072132> (visited on 02/05/2019).
- [13] *WPM02 Blood Pressure Monitor Cover Letter Embedded 1001312 BT 2.0x1.2 110810 Withings*. URL: <https://fccid.io/XNAWPM02/Letter/Antenna-Spec-2072133> (visited on 02/05/2019).
- [14] Great Scott Gadgets. *HackRF One*. URL: <https://greatscottgadgets.com/hackrf/> (visited on 01/05/2019).
- [15] *APKMirror - Free APK Downloads - Download Free Android APKs #APKPLZ*. URL: <https://www.apkmirror.com/> (visited on 01/05/2019).
- [16] *ALFA Network AWUS036NHA - WikiDevi*. URL: https://wikidevi.com/wiki/ALFA_Network_AWUS036NHA (visited on 01/05/2019).
- [17] *KASA*. URL: <https://www.tp-link.com/us/kasa-smart/kasa.html> (visited on 01/05/2019).
- [18] *Run Windows on Mac — Virtual Machine for Mac — VMware Fusion*. URL: <https://www.vmware.com/products/fusion.html> (visited on 01/05/2019).
- [19] *About Kali Linux*. URL: <https://www.kali.org/about-us/> (visited on 01/05/2019).

- [20] *Kali Linux Custom Image Downloads - Offensive Security*. URL: <https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/> (visited on 01/05/2019).
- [21] *Nmap: the Network Mapper - Free Security Scanner*. URL: <https://nmap.org/> (visited on 01/05/2019).
- [22] *Main [Aircrack-ng]*. URL: <https://www.aircrack-ng.org/doku.php> (visited on 01/05/2019).
- [23] *l2ping(1) - Linux man page*. URL: <https://linux.die.net/man/1/l2ping> (visited on 01/05/2019).
- [24] *About GNU Radio · GNU Radio*. URL: <https://www.gnuradio.org/about/> (visited on 01/05/2019).
- [25] *GNURadioCompanion - GNU Radio*. URL: <https://wiki.gnuradio.org/index.php/GNURadioCompanion> (visited on 01/05/2019).
- [26] *Gqrx SDR - Open source software defined radio by Alexandru Csete OZ9AEC*. URL: <http://gqrx.dk/> (visited on 01/05/2019).
- [27] *GitHub - jopohl/urh: Universal Radio Hacker: investigate wireless protocols like a boss*. URL: <https://github.com/jopohl/urh> (visited on 02/05/2019).
- [28] *SI6 Networks*. URL: <https://www.si6networks.com/tools/iot-toolkit/> (visited on 02/05/2019).
- [29] *fgont-troopers2017-iot-tp-link-hacking.pdf*. URL: <https://www.si6networks.com/presentations/troopers2017/fgont-troopers2017-iot-tp-link-hacking.pdf> (visited on 02/05/2019).
- [30] *Kasa Smart 1.7.7.638 APK Download by TP-LINK Research America - APKMirror*. URL: <https://www.apkmirror.com/apk/tp-link-research-america/kasa-for-mobile/kasa-for-mobile-1-7-7-638-release/kasa-mobile-1-7-7-638-android-apk-download/> (visited on 03/05/2019).
- [31] *Hacking & Solutions: 802.11 Protocol Attacks, Deauthentication*. URL: <https://www.cwnp.com/hacking-solutions-802-11-protocol-attacks-deauthentication/> (visited on 03/05/2019).
- [32] *Home — MAC Vendor Lookup Tool & API — MACVendors.com*. URL: <https://macvendors.com/> (visited on 03/05/2019).
- [33] John P Dunning. *Bluetooth Threat Taxonomy*. 2018. URL: https://vtechworks.lib.vt.edu/bitstream/handle/10919/76883/etd-10242010-163002_Dunning_JP_T_2010.pdf.
- [34] *Amazon.co.uk Help: Enable Alexa Skills*. URL: https://www.amazon.co.uk/gp/help/customer/display.html/?ie=UTF8&nodeId=201848700&ref_=sv_a2s_6 (visited on 10/05/2019).
- [35] *Zigbee Networking Basics (35,000ft view) - Acuity Support*. URL: <https://acuitysupport.zendesk.com/hc/en-us/articles/225413967-Zigbee-Networking-Basics-35-000ft-view-> (visited on 10/05/2019).
- [36] *What is orthogonal frequency-division multiplexing (OFDM)? - Definition from WhatIs.com*. URL: <https://searchnetworking.techtarget.com/definition/orthogonal-frequency-division-multiplexing> (visited on 10/05/2019).