

BACHELOR'S THESIS COMPUTING SCIENCE



RADBOD UNIVERSITY NIJMEGEN

Hack my audio back

An analysis of practical attacks on two Bluetooth speakers.

Author:
Sam Haeck
s1040362

First supervisor/assessor:
Professor Joan Daemen

Second assessor:
Professor Katharina Kohls

June 14, 2022

Abstract

The purpose of this research is to analyze a Man-in-the-Middle attack on Bluetooth speakers. This is done by answering In what way it is possible to sever the connection between a users device and a Bluetooth speaker, or alter the information exchanged between them, by exploiting weaknesses in the Bluetooth protocol. 3 Different speakers were taken and connected to using a laptop. The Bluetooth traffic was then analyzed and checked for the possibility of a MITM attack. Since there is no indication provided to the user whether a connection is made to the correct speaker, in the implementation of Bluetooth in all three speakers, a MITM was possible. The attack made use of the tool btproxy and worked the same for all three speakers. No alterations had to be made due to differences in security. In conclusion, it became clear that the information between the host and controller could be changed by a Man-in-the-Middle attack. This attack worked, since the user does not know whether a connection is happening to the real host, or to an impersonator.

Contents

1	Introduction	4
2	Preliminaries	6
2.1	Bluetooth	6
2.1.1	General Description	6
2.1.2	Security	8
	BR/EDR and AMP security	9
	Secure Simple Pairing	9
	Secure Connections Only Mode	12
	LE security	12
2.2	Attacks	13
2.2.1	MITM	13
2.3	Tools	14
2.3.1	Wireshark	14
2.3.2	Xcode and additional tools	14
2.3.3	PyBluez	15
2.3.4	BTProxy	15
2.3.5	VirtualBox	17
2.3.6	Hcitrust	17
3	Research Methods	18
3.1	Capturing Bluetooth traffic	18
3.1.1	Setup	18
3.1.2	Experiment	18
3.1.3	Explanation of steps	20
3.1.4	Identifying Bluetooth version and models	23
3.1.5	Identifying different phases	24
3.2	MITM attack	24
3.2.1	Setup	24
	Setting up VirtualBox	25
	Setting up Kali Linux	25
	Setting up BT-Proxy	25
3.2.2	Performing the attack	26

4	Results	27
4.1	Capturing Bluetooth traffic	27
4.1.1	Bose Speaker	27
	Bluetooth model	27
	Device discovery	28
	Connection request	28
	Information and SDP	29
	Authentication	30
	Encryption	31
	Setting up communication	31
	Sending data	31
	Disconnection	32
4.1.2	JBL speaker	32
	Bluetooth model	32
	Device discovery	33
	Connection request	33
	Information, SDP and Authentication	34
	Encryption	34
	Final capture	35
4.1.3	Myriad speaker	35
4.2	MITM results	36
4.2.1	Flow	36
4.2.2	Example of changing name	36
5	Related Work	38
5.1	MITM on Bluetooth	38
5.2	Bluetooth weaknesses	39
6	Conclusions	40
7	Future work	41
7.1	Attack prevention	41
7.2	Bluez for non Linux users	41
7.3	Speeding up	41
7.4	Automation	42
A	Appendix	46
A.1	Packet Captures	46
A.1.1	Bose Box	46
A.1.2	JBL Box	46
A.1.3	Myriad Box	46
A.1.4	Kali linux prepared virtual Box	46
A.2	Hardware information	46
A.2.1	MacBook Pro	46

A.3 Bluetooth security figures	47
--	----

Chapter 1

Introduction

Nowadays, the communication protocol Bluetooth is present almost everywhere. Bluetooth is a wireless technology used for short-range communication between electronic devices. With the new trend on smart devices this protocol is increasingly important. However, the misuse of it can set some irritations. Consider for example the following situation: imagine arriving home and there is a party in your building. People set up multiple speakers and the music blasting through them is very loud and simply distasteful. Speaking with the host of the party got you nowhere and the situation is starting to get on your nerve, since sleeping is just impossible this way. However, there are multiple Bluetooth access points of the different speakers that could be tried to access to put on different noise or music. This sparked the question: Could the behavior of a Bluetooth speaker that is not yours be modified, by exploiting a weakness in the Bluetooth protocol?

This problem shows how a person can be set on a malicious path to alter and use technology of others for his own purposes. Bluetooth is used everywhere nowadays. Not only only for the previous example of playing music over speaker, but also for the exchange of different information. Bluetooth is for example used as a base protocol in Airdrop[4], which is used in the exchange of information in apple devices. Even doing a simple scan for accessible devices in your own building will have a probable outcome of multiple devices to select and many different Bluetooth versions to connect to. Compatibility is preferred to security[7], so different generations of Bluetooth in these devices can become a security risk. If Bluetooth is exploited by a person with bad intentions, problems would arise. That is why it is imperative to solve this problem or to the least perform an analysis of attacks on Bluetooth.

The goal of this research is to answer this question: In what way is it possible to sever the connection between a users device and a Bluetooth speaker, or alter the information exchanged between them, by exploiting weaknesses in the Bluetooth protocol? This question leads to the following sub-questions:

1. Is there such a difference in implementation of Bluetooth in different types of speakers, that the user notices these differences while using the speakers?
2. Is Bluetooth implemented in such a different way in each speaker that a different implementation of the attack had to be implemented per speaker?

A solution for the research question is needed to raise awareness on unsafe usage of Bluetooth and the and to demonstrate security risks that can come with the implementation of it. The solution will be an analysis of a Man-in-the-Middle attack (see chapter 2) between the Bluetooth controlling device of a user and Bluetooth speaker.

By explaining how the attack is performed and what issues are the cause of the vulnerabilities an intuition can be created to prevent these causes in future implementations and usage of Bluetooth.

Following next in this thesis are first the preliminaries(2), where the context and theory needed to understand this thesis is explained. After this the flow of the experiment and the setup will be explained in research methods(3). Then the results of the experiment will be showcased in results (4). Much research has already taken place on Bluetooth weaknesses and MITM attacks on Bluetooth, so to demonstrate the contribution of this thesis, the chapter related works (5), discusses this research. The conclusions are drawn in the chapter after that (6), finally, future work (7) is recommended.

Chapter 2

Preliminaries

The research was conducted with quite some tools and a great deal of knowledge not necessarily generally known. This is why the preliminaries will help understanding this knowledge.

2.1 Bluetooth

Bluetooth[7] is the key subject of this thesis. Especially the security aspects of it. This section explains the terms commonly used in this paper.

2.1.1 General Description

Bluetooth is a very robust technology[22] due to a frequency hopping technique. Depending on the implementation of Bluetooth, it does not consume a lot of power[17]. Many of the features in Bluetooth specifications are optional which allows differentiation in the product and therefore it can be implemented on all kinds of devices[8]. Optional in this case means that a device can have a Bluetooth implementation, without for example extra security features although they are available.

Due to the wide variety of devices that need communication protocols there are two forms or implementations of Bluetooth: Basic Rate (BR) and Low Energy (LE). Both systems include device discovery, connection establishment and connection mechanisms. Both can be implemented on the same device, but the two forms also work separately from each other.

The Basic Rate system includes optional Enhanced Data Rate (EDR) Alternate Media Access Control (MAC) and Physical (PHY) layer extensions. The Basic Rate can thus communicate data faster than the LE system, but uses more power. The LE system includes features used for products that require less power and lower complexity than BR/EDR. The LE system is

also designed for use cases and applications with lower data rates and has lower duty cycles. Depending on the specific use case, one system may work more optimally than the other.

Devices that implement both systems are able to communicate with other devices implementing both systems as well as devices implementing either system. Some use cases can only be supported by one system, for example a heart rate monitor, which needs to make use of low energy features.[30] Ideally, the device supports both systems, since it is then suited to communicate with most other devices.

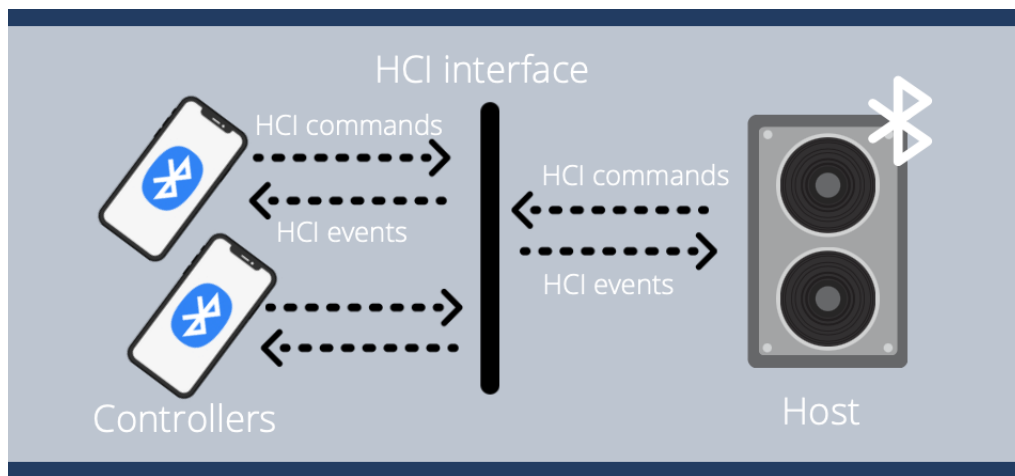


Figure 2.1: Host-controller interface

The Bluetooth core system consists of a Host and one or more Controllers. For example a Bluetooth speaker(Host) and two phones sending music requests(Controllers). A Host and the Controllers are able to interact through the Host Controller interface (HCI), see figure 2.1. HCI commands are sent to the interface and the events are sent from the interface to the Host/-Controller. Through HCI events the Host and Controllers are updated and informed by the HCI. Two types of Controllers are defined in this version of the specification: Primary Controllers and Secondary Controllers.

An implementation of Bluetooth has only one Primary Controller which may be one of the following:

- a BR/EDR Controller.
- an LE Controller.
- a combined BR/EDR Controller and LE Controller portion into a single Controller.

A Bluetooth core system may additionally have one or more Secondary Controllers, provided it has the following configuration:

- an Alternate MAC/PHY (AMP) Controller including an 802.11 Protocol Adaptation Layer, a protocol layer which makes Bluetooth very fast[10], 802.11 Medium access control(MAC) which is another protocol layer[19] and PHY or physical transport. AMP controllers are only used as secondary controllers.

2.1.2 Security

There are many security measures with the Bluetooth protocol: Pairing, Bonding, authentication, encryption,integrity. For supporting legacy communication, those different security measures can be implemented using different primitives:

1. Pairing: the initial phase in which one or more shared secret keys are created.
2. Bonding: this makes sure the devices 'get to know each other', so the keys created during pairing are stored for use in subsequent connections in order to form a trusted device pair.
3. Device authentication: verification that the two devices have the same keys
4. Encryption: message confidentiality, so encrypting the data send via the Bluetooth communication channel.
5. Message integrity: protection against message forgeries, using for example signed data[7].

BR/EDR and AMP security

Figure 2.2 shows the security key hierarchy of BR/EDR and MAC/PHYs (AMP). As can be observed the security architecture and methods used differ from each other since it depends on whether Secure Simple Pairing or Secure Connections algorithms and procedures are used. Secure Connections and Secure Simple Pairing are two different mechanisms which can be put in place. Secure Connections is the newest mechanism and upgrades the old Secure Simple Pairing mechanism. Keys are generated on the Controller

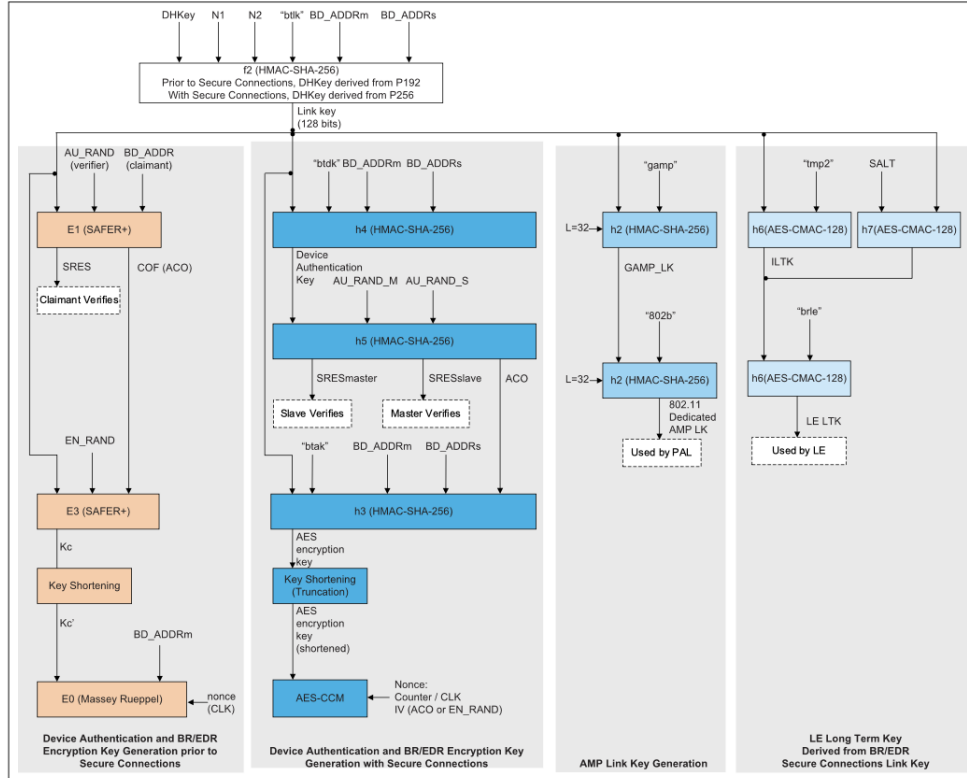


Figure 2.2: Security key hierarchy BR/EDR, AMP[7]

Secure Simple Pairing

The main goal of Secure Simple Pairing is to make it easy for the user to pair his device with the Bluetooth host device. Secondly, the security of Bluetooth should be increased by this mechanism, in comparison to the previous pairing method, Legacy pairing. The two main security goals are protection against passive eavesdropping and man-in-the-middle attacks. One important element that helps in increasing Bluetooth security is thus authentication of the controllers device to the host device and the other way around.

Secure Simple Pairing is able to make use of 4 different association models to provide protection. Some authentication models require human interaction and others do not. This is related to the device capability. Therefore the security of each method is not equivalent. These association models are, Numeric Comparison, Just Works, Out of Band and Passkey Entry. Figure 2.3 shows how these are used.

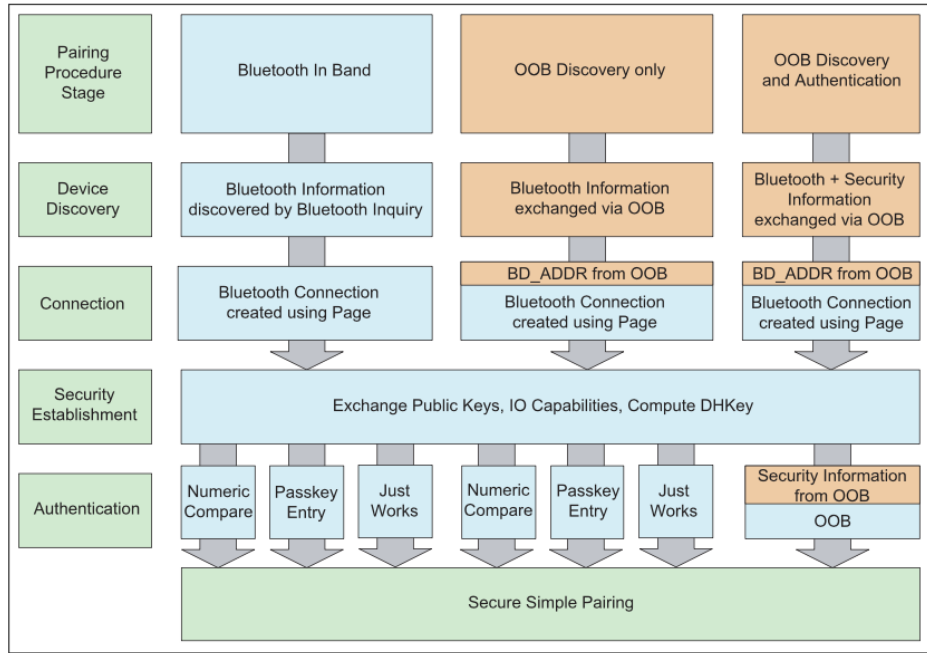


Figure 2.3: Association models[7]

Numeric Comparison

When the association model of numeric comparison is used both devices that are pairing are capable of displaying a 6 digit number to the users. Both devices also have the possibility to confirm whether the numbers are the same by entering ‘yes’ or ‘no’. An example use case for this is a phone pairing to a PC.

When trying to pair, the six digit number should be the same on both devices. If this is the case and both devices confirm, pairing is successful. Many Bluetooth devices do not have unique names. This procedure helps to identify accidental wrong connections. The procedure also helps to prevent man-in-the-middle attacks, since the user can see when a wrong code is provided. Passive eavesdropping is also becoming a lot harder. This is because the number is hard to guess. 6 Digits with each digit having 10 possibilities gives 10^6 possibilities to guess.

Just Works

This model uses the same model as Numeric Comparison, but the user does not see any number nor does the user have to enter a number. This model is used when one or more devices are not able to display a number. The user is simply asked to accept the connection. The exact implementation of Just Works is up to the end manufacturer. An example of this is a phone connecting to a wireless headphone.

Just Works provides less protection against MITM attacks than numeric comparison, since there is no method of confirmation for the user to check whether he pairs to the correct device. It does however protect a bit more against passive eavesdropping than when using no security code at all, since the same base model as Numeric Comparison is used[2]. So passive eavesdropping becomes harder because a 6 digit numeric code is still used while pairing.

Out of Band

OOB association model is used mainly in situations where an Out of Band mechanism is used for device discovery and an exchange in cryptographic numbers in the pairing process. An OOB mechanism provides an extra channel separate from the main channel which allows data sent via that channel to be separate from any other data. An example of the OOB model in practice is with a solution making use of Near Field Communication(NFC). In this case two devices will touch together and after this confirmation is asked on the device to pair. An exchange of information takes place, both device discovery information and cryptographic information. So names and services are exchanged but also the secret keys and encryption keys.[11]

This model is as secure as the OOB channel used[27]. If the OOB channel is not secure against MITM attacks then the security is compromised during authentication.

Passkey Entry

This model is used in a scenario where one device has input capabilities, but no output capabilities and the other device does have output capabilities. For example with a wireless keyboard and a PC.

On the device with a display a security code of 6 digits is shown. The device with only input capabilities should input this same code in order to pair. If the entered number is correct and the numbers correspond, the pairing is successful.

Since the security code is hard to guess there is some protection against passive eavesdropping and MITM attacks[29]. If a third party tries such an attack, it will have to guess the code. Otherwise the security code that is displayed is not valid, while it is entered on the Bluetooth device.

Secure Connections Only Mode

As stated earlier this mode upgrades the security of Secure Simple Pairing. This mode is used when there is a need for high security and the device requires to only have federal information processing standard FIPS[12] approved algorithms, particularly FIPS level 140-2[23]. As can be seen in the 2.2, the algorithms that are not FIPS[6] approved are replaced with FIPS approved algorithms. So elliptic curve P-256 is enforced by the host, AES-CCM is used for encryption and secure authentication sequences are used, meaning that both host and controller verify each other instead of having authentication happen only on one side.

The benefit of Secure Connections Only Mode is that the security of Bluetooth increases significantly. However, the downside is that devices that do not support this mode have no backwards compatibility to the devices that use this mode.

LE security

The security model of LE is LE Legacy Pairing and LE Secure Connections Only Mode. LE makes use of association models as well. These are similar as to BR/EDR Secure Simple Pairing association models. LE Legacy Pairing covers 3 out of 4 models and has no equivalent for Numeric Comparison.

The major difference in LE Legacy Pairing and BR/EDR Secure Simple Pairing is that LE Legacy Pairing does not provide protection against passive eavesdropping within association models Just Works and Passkey Entry. This is because Secure Simple Pairing uses Elliptic Curve-Diffie Hellman and LE Legacy Pairing does not.

Another difference is that the Host generates the keys instead of the Controller.

2.2 Attacks

There are many different ways and attack models that could be used to analyze Bluetooth security. For this research a focus will be placed on the Man-in-the-Middle attack model.

2.2.1 MITM

A man-in-the-middle attack[13] is a situation where a person with malicious intentions places his own device in between a sender and a receiver or in the midst of a communication channel, giving the attacker access to the traffic. The intention of the attacker is to do something with the traffic being send over the communication channel. This can be stealing, monitoring or altering data or impersonate one of the parties, ideally without any of the parties noticing odd behavior.

There are two phases the attacker has to go through in order to be successful, being interception and decryption.

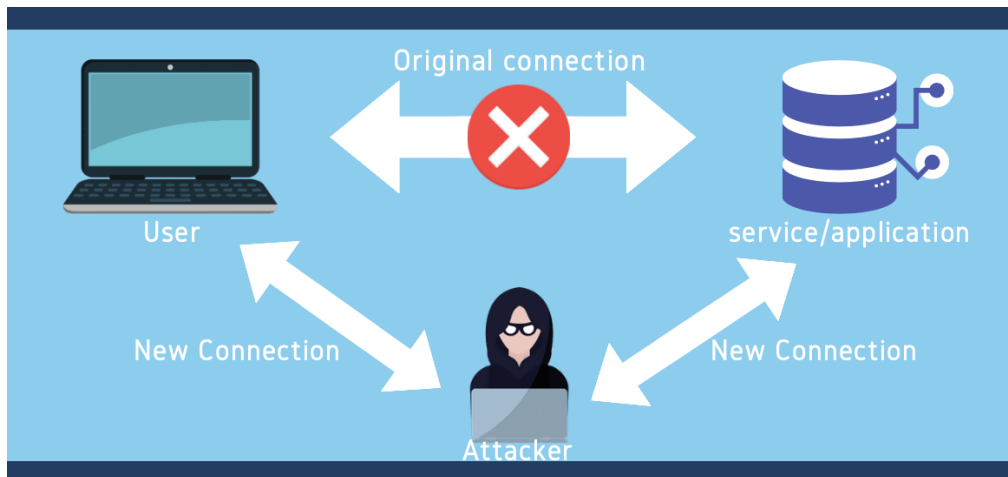


Figure 2.4: Man-in-the-middle attack

As can be seen in figure 2.4, the attacker impersonates both parties. The attacker can read and alter data as they please.

2.3 Tools

There are quite some ways to retrieve information on protocols and attack them. The tools used in this research are explained in this section.

2.3.1 Wireshark

Wireshark[31] is a tool that allows the user to capture packets of protocols used for communication, for example network protocols. In this way the user can analyze the contents of packets communicated through the protocol. Wireshark also allows the user to resend messages, alter them and set breakpoints at certain packets.

Captures of traffic can be saved and opened later to analyze the packets again. This is why traffic captured by other tools can also be opened in wireshark to analyze further.

2.3.2 Xcode and additional tools

Specifically for Apple products the Xcode toolset[5] was created. It is used to create apps for all kinds of Apple products. Development in user interface design, coding, testing and debugging is all integrated in this tool set.

Additionally there is an extra tool set for Xcode[14]. This set contains tools that make development easier, like a printer simulator. The set also contains a particular useful tool: PacketLogger. This tool allows the user to capture the packets of a specific Bluetooth device. Filters can be added and adjusted to analyze a specific part of the communication.

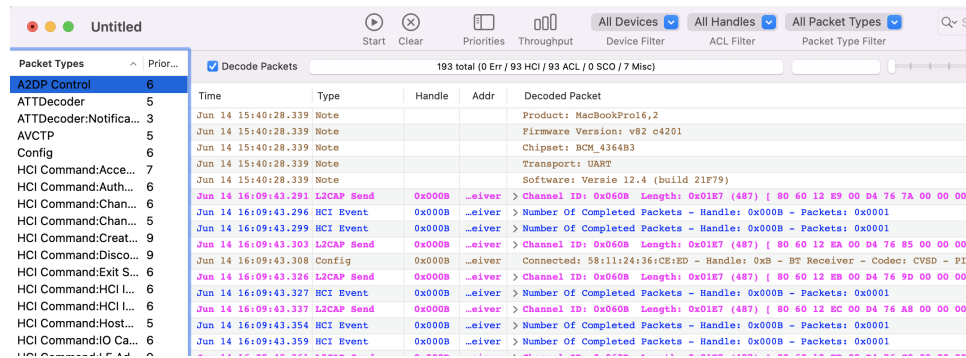


Figure 2.5: Packetlogger program

Figure 2.5 contains an example of what a capture could look like. As can be seen the network traffic is captured in an orderly color coded manner without any noise of other non Bluetooth traffic in between. However, analyzing a specific packet does not give as much information as in Wireshark.

A traffic capture can also be saved and be imported later to analyze or analyze further. The exported data is compatible with Wireshark, which makes analyzing packets more specifically also possible.

2.3.3 PyBluez

PyBluez is an extension module to the programming language Python. The module allows the programmer to access system Bluetooth resources. There are several useful functions, like `send()` which allows the programmer to send custom Bluetooth packets, or `discover_devices()` to send an inquiry on available Bluetooth devices.

Unfortunately, the integration for PyBluez on Mac has been deprecated, since the underlying library Bluez, only works on Linux[25]. The reason for this is that the module lightblue, used by Bluez, has to have the LightAquaBlue framework working on mac. This LightAquaBlue framework used deprecated function calls. Some of these functions can be transformed using XCode, but unfortunately not all[1].

2.3.4 BTProxy

The Bluetooth proxy can be used to perform a MITM attack on Bluetooth BR/EDR devices. The original project has been developed by Patrick Conor[24]. The proxy can only be used on Linux, since it makes use of the Bluez library. The tool was deprecated and could only use functions of Python 2. In order to enforce python 3 some changes have been made.[26]

After the install of the tool the attacker can simply run the proxy by running it in the terminal like so:

```
btproxy F1:64:F3:31:67:88 40:14:33:66:CC:FF
```

Where F1:64:F3:31:67:88 is the controllers mac address and 40:14:33:66:CC:FF the hosts mac address.

The proxy works by firstly scanning the specified host device for a name and device classes. After this it can make an accurate clone of the Bluetooth module of that device. For the purpose of this research the name of the clone gets `_btproxy` appended to it, but in a real attack that would not happen. Then the proxy scans for services on the host device. For each of the services a socket connection will be created and a listening port for the controller to connect to.

An example showing how services are cloned:

1. A clone is made on the device running the proxy, containing the name: 'Speaker_btproxy', and device class '0x340404'.
2. A scan for services is done by the proxy to the host and three services are returned:
 - Rendering
 - Object transfer
 - Audio
3. Listening ports are created by the proxy:
 - Proxy listening for connections for "Rendering"
 - Proxy listening for connections for "Object transfer"
 - Proxy listening for connections for "Audio"
4. The proxy connects to the host device per service:
 - Attempting connections with 3 services on slave.
 - Connected to service "Audio"
 - Connected to service "Object transfer"
 - Connected to service "Rendering"

The real host device is now connected to the clone, so only the clone appears on the list of available devices. As soon as the controller connects to the clone the MITM attack is successful.

Traffic received and sent further by the proxy can be modified by changing the implementation of the following functions:

```
# replace.py
def master_cb(req):
    """
        Received something from master, about to be sent to
        slave.
    """
    print '<< ', repr(req)
    open('mastermessages.log', 'a+b').write(req)
    return req
```

The argument **req** is the request that the controller has sent to the host. The return statement gives the information back so the proxy will send it further to the host. Suppose the use case for the MITM attack is to make sure nothing happens on the host, then packets can simply be dropped. This could be done by changing for example the last line to **return ""**.

```
def slave_cb(res):
    """
        Same as above but it's from slave about to be sent to
        master
    """
    print '>> ', repr(res)
    open('slavemessages.log', 'a+b').write(res)
    return res
```

This function can be used to modify the responses sent from host to controller. Any changes to `res` will change the response sent on further by the proxy.

2.3.5 VirtualBox

VirtualBox[21] is a tool that allows the user to run virtual machines on their device. The tool has the feature to import pre-build virtual machines and it is available for MacOS. With the use of VirtualBox, it will thus be possible to run Linux on MacOS.

2.3.6 Hcitol

hcitol[16] is a Bluetooth tool making use of the Bluez library. It is used to configure Bluetooth connections and send some special command to Bluetooth devices. These commands use separate sub-functionalities of Bluetooth[28]. For example: `hcitol name 14:7D:DA:E3:F7:67` gives only the name of the Bluetooth device with address 14:7D:DA:E3:F7:67. In order to get the Bluetooth addresses this tool can be used by either of the following commands:

```
hcitol scan
hcitol inq
```

Chapter 3

Research Methods

This research has been conducted in the form of an experiment. Three Bluetooth devices will be taken: a Bose soundlink mini speaker, a JBL speaker and a Myriad speaker. These devices will connect to a MacBook Pro (See Appendix A for full specifications) with a Bluetooth connector integrated in the system. For the attack phase a Bluetooth controller was needed. An iPhone 10 was taken for this purpose.

3.1 Capturing Bluetooth traffic

The first phase of the experiment is capturing Bluetooth traffic. The experiment is the same for all three speakers.

3.1.1 Setup

Figure 3.1 shows the setup of the first part of the experiment. The laptop has a Bluetooth-controller, which makes it possible to connect from the laptop to the Bluetooth speaker. The programs on figure 3.1 are necessary for this experiment in order to form the connection, inspect the connection and send data over the connection.

3.1.2 Experiment

1. Start up the Bluetooth receiver.
2. Start up PacketLogger.
3. Start up Safari and search on Youtube for ‘Money for Nothing’ by ‘Dire Straits’.



Figure 3.1: Initial setup

4. Turn on the speaker.
5. Set the speaker to pairing mode. This means that the speaker allows incoming connections.
6. Start capturing packets.
7. Turn the Bluetooth receiver on.
8. Connect to the speaker, Packets should now be appearing in Packet-Logger.
9. Wait 10 seconds.
10. On Youtube, start the video.
11. Wait 15 seconds.
12. Pause the Youtube video.

13. Turn Bluetooth off.
14. Stop capturing packets.
15. Turn off the speaker.

3.1.3 Explanation of steps

- **Start up the Bluetooth receiver:** click on the search icon on the right top of the laptop screen. Type in Bluetooth and click on the result shown in the screenshot 3.2.

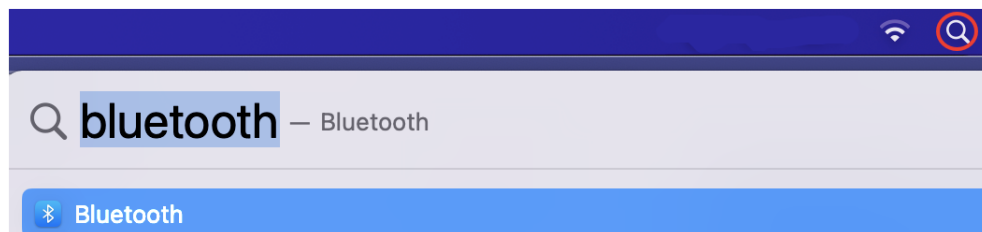


Figure 3.2: Bluetooth search screenshot

- **Start up PacketLogger:** click on the search icon on the right top of the laptop screen. Type in PacketLogger and click on the result shown in the screenshot 3.3.

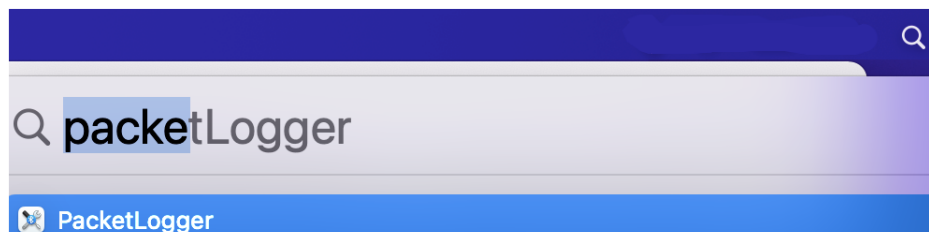


Figure 3.3: PacketLogger search screenshot

- **Start up Safari and search on Youtube for 'Money for Nothing' by 'Dire Straits':** Follow the same procedure as step 1 or 2, but search for 'Safari'. Then enter the following link:

`https://www.youtube.com/watch?v=wTP2RUD_cL0`

This link brings up the following Youtube page:
Pause the video for now by hitting 'space'.

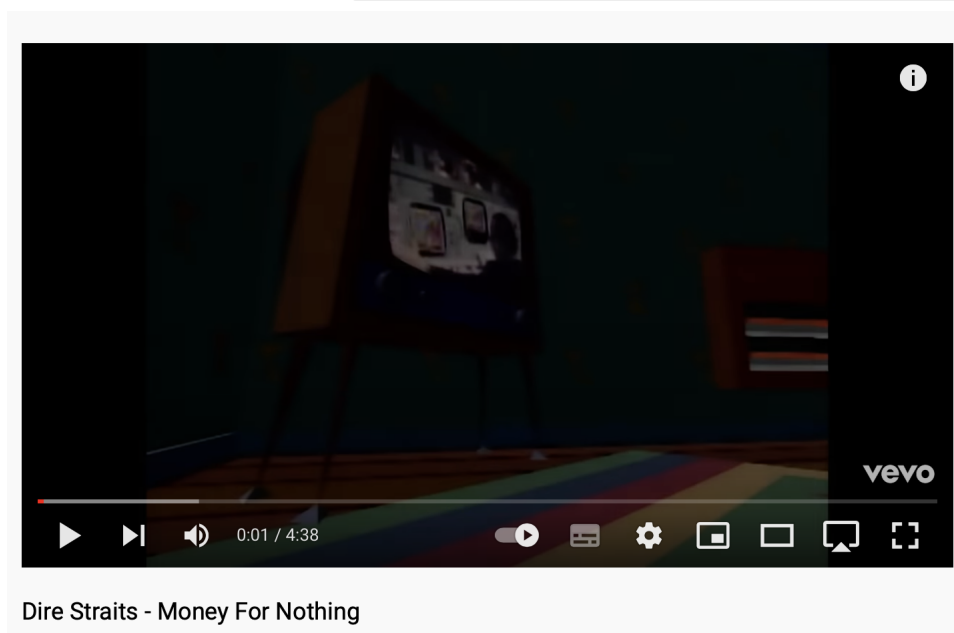


Figure 3.4: Youtube page screenshot



- **Turn on the speaker:** search for this icon and click it: .
- **Set the speaker to pairing mode:** search for this icon and click it: .
- **Start capturing packets:** In PacketLogger, click the start button.



Figure 3.5: Start logging packets

- **Turn the Bluetooth receiver on:** In the Bluetooth app, click the 'on' button.
- **Connect to the speaker:** The name of the speaker appeared in the list of connectable Bluetooth devices. Click on this name.
- **Start or pause a Youtube video:** This is simply done by clicking 'space'.

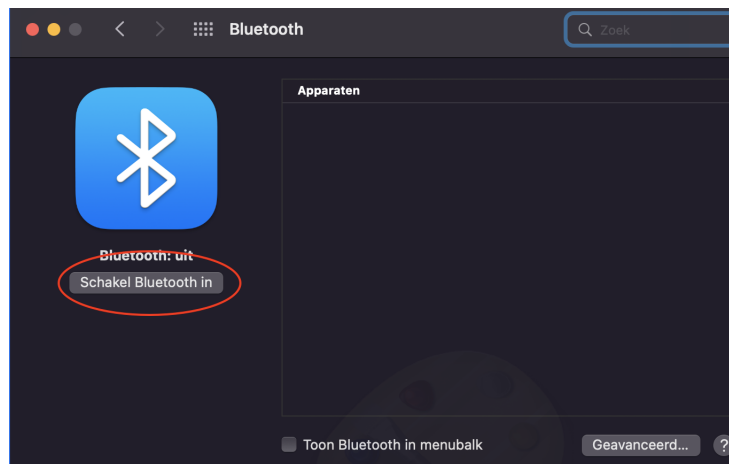



Figure 3.6: Turn on Bluetooth



Figure 3.7: Connect to the speaker

- **Turn Bluetooth off:** click on the same button in the Bluetooth app as used for turning it on.
- **Stop capturing packets:** In PacketLogger the button that said 'start' now says 'stop'. Click on this button.
- **Turn the speaker off:** search for this icon and click it: .

This experiment is to analyze whether a MITM attack would be possible. For the experiment all steps would be followed, a real user would exclude steps 2,6,9,11 and 14. The user has no interest in what the packets look like, just that the speaker plays the correct music.

An attacker is not able to do this experiment to retrieve all information, since he does not have access to the foreign speaker he wants to attack and the experiment takes too much time. This is why an attacker would use `hcitool` to retrieve the information needed. The attacker mainly needs the name and Bluetooth address. These can be obtained by running:

```
hcitool scan
Scanning ...
    00:80:25:2F:1E:49      SMA001d: 2140034381 SN2140034381
```


The example gives the device address: 00:80:25:2F:1E:49 and the name: SMA001d.

3.1.4 Identifying Bluetooth version and models

The results of the experiment will be a packet capture containing information about the Bluetooth connection. It will show the protocols used but also the version of Bluetooth BR/EDR or LE is used. The information that will be identified is this:

Bluetooth version:

The Bluetooth version is determined by looking at the inquiries. In the packet capture two different types of inquiries can be identified. The LE inquiry and the BR/EDR inquiry.

One of the inquiries only happens once and after that a connection follows. The other inquiry will happen multiple times more during the connection. Whichever version this inquiry shows is not the Bluetooth version. To give an example: a packet capture has been retrieved and during the connection the following packet appears multiple times:

HCI Command 0x0000 **LE** Set Scan Parameters - **Active** - 30/300 (ms) SEND

As can be seen, this inquiry is for Bluetooth LE. This means that the connection has been set up using BR/EDR and this is thus the version used.

Association models:

To determine which association model is used, the packet capture itself will be observed, searching for clues, but also the interaction with the Macbook Pro and the Bluetooth speaker will be looked at.

If for example a number has to be entered on the Macbook, because the speaker displayed this number on a led, it can be known that the association model Passkey Entry is used.

So after the connection has been initiated by clicking the name of the speaker in the Bluetooth list, attention is paid what is asked of the user:

- Is nothing further required? The association model is Just Works.
- Are numbers being displayed and does the user have to compare? The association model is Numeric Comparison.
- Does the user have to manually enter a number to confirm? The association model is Passkey entry?

- Does the user have to do anything different, like touching the devices?
The association model is Out of Band.

3.1.5 Identifying different phases

The packet capture will show all packets related to Bluetooth captured during the designated period. The list of packets is sorted on time, so the earliest packet is one top of the list and the latest at the bottom. This means that for each phase it can be seen exactly what happens during that period.

To identify such a phase it is important to identify when a phase changes. These changes are marked by packets that state ‘completed’ or ‘successful’ followed by a different kind of request.

The way these state changed will be identified is simply going through each packet and checking if it belongs to a sequence or that it starts a new sequence.

3.2 MITM attack

With the information from the previous phase, a Man-in-the-Middle attack could be implemented.

3.2.1 Setup

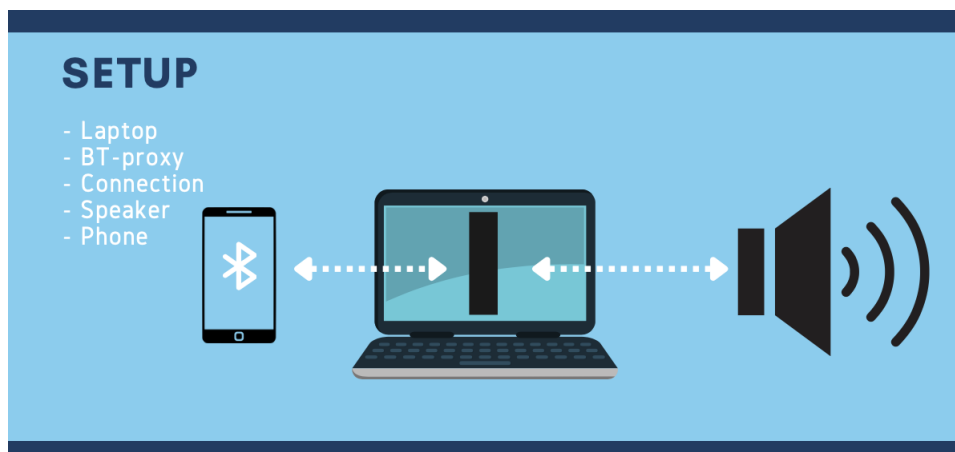


Figure 3.8: Setup for the attack

Figure 3.8 shows the setup for the attack. On the Macbook VirtualBox is running an instance of Kali Linux. On this instance the Bluetooth proxy is running. The controller is an iPhone 10, with address 72:5A:FD:3B:3C:9D.

The proxy stops incoming traffic from both the host and controller and based on the implementation either sends nothing, the right packets, or different packets through to the other entity.

Setting up VirtualBox

1. Download VirtualBox from this link[20], the OS X version.
2. Double click the .dmg file that has been downloaded.
3. Let the install program run and give the password of the Macbook when the install program asks for it.

Setting up Kali Linux

1. Start VirtualBox by opening finder→apps→VirtualBox and double clicking the icon.
2. Choose to import a virtual machine from the local file system.
3. Select either a new clean virtual box following this link[15] or import the virtual machine with BT-proxy already prepared A.

No further setup is required if the pre-installed bt-proxy box is chosen. Otherwise the following steps still have to be followed.

Setting up BT-Proxy

1. Start the VirtualBox and log in using username:kali and password:kali.
2. In the Virtual Machine, open Firefox and follow this link. Download the code.
3. Open the terminal:
Install pip:
`python3 -m ensurepip --default-pip`
Install the requirements:
`sudo apt-get install bluez bluez-tools libbluetooth-dev python-dev`
Uninstall setuptools due to an incompatibility error:
`sudo pip uninstall setuptools`
Install an earlier version of setuptools:
`pip install setuptools==57`
Now BT-proxy can be installed:
`sudo python btproxy/setup.py install`

All tools have now correctly been set up.

3.2.2 Performing the attack

To perform the attack the following steps were taken:

1. The virtual machine was started and logged into using username:kali and password:kali.
2. From the information retrieval phase the correct Bluetooth addresses of the Host and Controller were retrieved. These addresses could now be used in running the proxy:
`btproxy <master-bt-mac-address> <slave-bt-mac-address>`
3. Wait until the btproxy tool states: “Now you’re free to connect to...”
4. Turn on the speaker and make sure it is in pairing mode.
5. Navigate to the Bluetooth list on the iPhone.
6. Double click the speakers name in the Bluetooth list.
7. Play ‘Money for Nothing’, by ‘Dire Straits’ for 20 seconds.
8. Shut down the Bluetooth receiver and speaker.
9. Shut down the virtual machine.

Chapter 4

Results

4.1 Capturing Bluetooth traffic

Firstly the specific part of the results that shows the Bluetooth model will be looked at. After this is shown the different phases of the Bluetooth connection in the results will be walked through. For the full results see the Appendix A.

4.1.1 Bose Speaker

The different phases of the Bluetooth connection become clear from the capture. The initial phases that we see happening in the capture are mostly covered by the `HCI vendor specific` commands. This is the laptop simply initializing Bluetooth and preparing for a connection that might come in. After this the laptop starts scanning for devices it can connect to. In the capture this starts at:

Bluetooth model

The Bluetooth version used is BR/EDR. This is because packets doing a LE scan kept appearing:

```
Mar 11 10:50:00.792 HCI Command 0x0000 LE Set Scan Parameters - Active  
- 30/300 (ms) SEND (line 3183)
```

The association model used was Just Works. The user did not have to manually do anything else then simply selecting the device to connect to, this makes the connection vulnerable to MITM.

Device discovery

The device discovery phase starts at packet:

Mar 11 10:49:38.507 HCI Command 0x0000 **HCI Inquiry SEND** (line 153)

The result of the inquiry will then give the available devices and information about those devices. This information can be useful later. The result yields:

Mar 11 10:49:41.089 HCI Event 0x0000 08:E4:FD:B6:A7:D5 **Inquiry Result**

- **EIR**- 08:E4:FD:B6:A7:D5 - Audio : Headset - RSSI: -47 dBm RECV

Parameter Length: 255 (0xFF)

Num Responses: 0x01

Bluetooth Device Address: 08:E4:FD:B6:A7:D5

Page Scan Repetition Mode: 0x01

Reserved: 0x00

Class Of Device: 0x340404

Service Class: 0x01A0

Rendering

Object Transfer

Audio

Major Class: 0x0004

Audio

Minor Class: 0x0001

Headset

Clock Offset: 0x35E7

RSSI: -47 dBm

Local Name: S2025

(line 315)

The laptop kept on scanning for other devices, but the inquiry result shows that the Bose speaker has been found correctly.

Connection request

The user initiates the connection by clicking the button of the discovered device, which gives the following packet:

```

Mar 11 10:49:50.878 HCI Command 0x0000 08:E4:FD:B6:A7:D5 Create
Connection- 08:E4:FD:B6:A7:D5 SEND
  Opcode: 0x0405 (OGF: 0x01 OCF: 0x05)
  Parameter Length: 13 (0x0D)
  Bluetooth Device Address: 08:E4:FD:B6:A7:D5
  Packet Type: 0xCC18
  Page Scan Repetition Mode: 0x01
  Page Scan Mode: 0x00
  Clock Offset: 0xB5E7
  Allow Role Switch: 0x00
Mar 11 10:49:50.878 HCI Command 0x0000 00000000: 0504 ODD5 A7B6
FDE4 0818 CC01 00E7 B500 ..... SEND
(line 395)

```

Which succeeds:

```

Mar 11 10:49:51.021 HCI Event 0x000C 08:E4:FD:B6:A7:D5 Connection
Complete- 08:E4:FD:B6:A7:D5 RECV
  Parameter Length: 11 (0x0B)
  Status: 0x00 - Success
  Connection Handle: 0x000C
  Bluetooth Device Address: 08:E4:FD:B6:A7:D5
  Link Type: 0x01
  Encryption Mode: 0x00
line(405)

```

No indication is given that the user chose the right device. A MITM should be doable, the user will not know if the right host is connected to.

Information and SDP

The next couple of packets are to retrieve more information about the host. Information is requested and received with L2CAP and after this information is send over with SDP:

```

Mar 11 10:49:51.025 L2CAP Send 0x000C 08:E4:FD:B6:A7:D5 Information
Request SEND
Mar 11 10:49:51.029 L2CAP Receive 0x000C 08:E4:FD:B6:A7:D5 Information
Response RECV
Mar 11 10:49:51.030 L2CAP Send 0x000C 08:E4:FD:B6:A7:D5 Connection
Request - SDP SEND
Mar 11 10:49:51.035 L2CAP Receive 0x000C 08:E4:FD:B6:A7:D5 Connection
Response - SDP - Connection Successful RECV
Mar 11 10:49:51.035 L2CAP Send 0x000C 08:E4:FD:B6:A7:D5 Configuration
Request - SDP SEND
Mar 11 10:49:51.036 L2CAP Receive 0x000C 08:E4:FD:B6:A7:D5 Configuration
Request - SDP RECV
Mar 11 10:49:51.036 L2CAP Send 0x000C 08:E4:FD:B6:A7:D5 Configure
Response - SDP SEND
Mar 11 10:49:51.041 L2CAP Receive 0x000C 08:E4:FD:B6:A7:D5 Configure
Response - SDP RECV
(line 422)

```

Authentication

In the following lines authentication is requested:

```

Mar 11 10:49:51.079 HCI Command 0x000C 08:E4:FD:B6:A7:D5 Authentication
Requested- Connection Handle: 0x000C SEND
Mar 11 10:49:51.079 HCI Event 0x0000 Command Status - Authentication
Requested RECV
Mar 11 10:49:51.079 HCI Event 0x0000 08:E4:FD:B6:A7:D5 Link Key
Request - 08:E4:FD:B6:A7:D5 RECV
Mar 11 10:49:51.080 HCI Command 0x0000 08:E4:FD:B6:A7:D5 Link Key
Request Reply - 08:E4:FD:B6:A7:D5 SEND
    Opcode: 0x040B (OGF: 0x01 OCF: 0x0B)
    Parameter Length: 22 (0x16)
    Bluetooth Device Address: 08:E4:FD:B6:A7:D5
    Link Key: 0x50DC4972DD85CC12F1066561DE449DCD
Mar 11 10:49:51.080 HCI Command 0x0000 00000000: 0B04 16D5 A7B6
FDE4 08CD 9D44 DE61 6506 .....D.ae. SEND
Mar 11 10:49:51.081 HCI Event 0x0000 08:E4:FD:B6:A7:D5 Command Complete
[040B] - Link Key Request Reply - 08:E4:FD:B6:A7:D5 RECV
Mar 11 10:49:51.087 HCI Event 0x000C 08:E4:FD:B6:A7:D5 Authentication
Complete RECV
(line 465)

```


Encryption

The packets that make sure the messages will be encrypted:

```
Mar 11 10:49:51.087 HCI Command 0x000C 08:E4:FD:B6:A7:D5 Set Connection
Encryption- 0x01 - Connection Handle: 0x000C SEND
Mar 11 10:49:51.087 HCI Event 0x0000 Command Status - Set Connection
Encryption RECV
Mar 11 10:49:51.106 HCI Event 0x000C 08:E4:FD:B6:A7:D5 Encryption
Change Complete- Encryption Enabled RECV
Mar 11 10:49:51.106 HCI Command 0x0000 Read Encryption Key Size
SEND
(line 472)
```

Setting up communication

The next part is where communication channels are being set up and multi-media communication is also being initiated. This is seen by the RFCOMM packets and the A2DP packets in, for example, this part:

```
Mar 11 10:49:51.298 RFCOMM Send 0x000C 08:E4:FD:B6:A7:D5 UIH 6 Bytes
Of Data For Channel 0x04 [OK] SEND
Mar 11 10:49:51.303 A2DP Control Re 0x000C 08:E4:FD:B6:A7:D5 Open:
Accept-Response RECV
Mar 11 10:49:51.303 L2CAP Send 0x000C 08:E4:FD:B6:A7:D5 Connection
Request - AVDTP SEND
(line 545)
```

Sending data

After all the setting up is done, data can be send over the connection. The following packets show how this is done:

```
Mar 11 10:49:54.857 A2DP Audio Send 0x000C 08:E4:FD:B6:A7:D5 SBC
(Payload Not Logged) - Sequence 1 - Size: 608 (0x260) - Frames:
5 - Bitpool: 53 SEND
Mar 11 10:49:54.865 A2DP Audio Send 0x000C 08:E4:FD:B6:A7:D5 SBC
(Payload Not Logged) - Sequence 2 - Size: 608 (0x260) - Frames:
5 - Bitpool: 53 SEND
Mar 11 10:49:54.888 HCI Event 0x000C 08:E4:FD:B6:A7:D5 Number Of
Completed Packets - Handle: 0x000C - Packets: 0x0001 RECV
Mar 11 10:49:54.892 HCI Event 0x000C 08:E4:FD:B6:A7:D5 Number Of
Completed Packets - Handle: 0x000C - Packets: 0x0001 RECV
(line 2166)
```

As can be seen, the audio data is send in sequence, so the packets are numbered. Also for each packet send, a confirmation is received.

Disconnection

Finally to tear down the connection a sequence is set in motion:

```
Mar 11 10:50:01.536 L2CAP Send 0x000C 08:E4:FD:B6:A7:D5 Disconnection
Request - AVDTP SEND
Mar 11 10:50:01.536 L2CAP Send 0x000C 08:E4:FD:B6:A7:D5 Disconnection
Request - AVCTP SEND
Mar 11 10:50:01.566 L2CAP Receive 0x000C 08:E4:FD:B6:A7:D5 Disconnection
Response - AVDTP RECV
Mar 11 10:50:01.566 L2CAP Send 0x000C 08:E4:FD:B6:A7:D5 Disconnection
Request - AVDTP SEND
Mar 11 10:50:01.570 L2CAP Receive 0x000C 08:E4:FD:B6:A7:D5 Disconnection
Response - AVCTP RECV
Mar 11 10:50:01.576 L2CAP Send 0x000C 08:E4:FD:B6:A7:D5 Disconnection
Request - RFCOMM SEND
Mar 11 10:50:01.577 L2CAP Receive 0x000C 08:E4:FD:B6:A7:D5 Disconnection
Request - RFCOMM RECV
Mar 11 10:50:01.577 L2CAP Send 0x000C 08:E4:FD:B6:A7:D5 Disconnection
Response - RFCOMM SEND
Mar 11 10:50:01.730 HCI Event 0x000C 08:E4:FD:B6:A7:D5 Disconnection
Complete - Success RECV
(line 3156)
```

It can be seen that two disconnection requests are send for AVDTP, but only one response is received. This means that one packet got lost in transmission and that is why the controller sends another one. This one does get a response.

4.1.2 JBL speaker

The JBL speaker follows the same phases as the Bose speaker. For the full results, see the Appendix A.

Bluetooth model

Like the Bose speaker the JBL speaker has the BR/EDR version of Bluetooth. The LE scans kept happening during the connection. Also the association model used is Just Works, since there was no authentication action required of the user.

Device discovery

Like the Bose speaker and inquiry is send. The result of this inquiry is:

May 27 09:47:36.121 HCI Event 0x0000 70:99:1C:B5:31:21 **Inquiry Result-**

EIR - 70:99:1C:B5:31:21 - Audio : Loudspeaker- RSSI: -61 dBm RECV

Parameter Length: 255 (0xFF)

Num Responses: 0x01

Bluetooth Device Address:

70:99:1C:B5:31:21 Page Scan Repetition Mode: 0x01

Reserved: 0x00

Class Of Device: 0x240414

Service Class: 0x0120

Rendering

Audio

Major Class: 0x0004

Audio

Minor Class: 0x0005

Loudspeaker

Clock Offset: 0x5F8E

RSSI: -61 dBm

Local Name: JBL GO 2+

Transmit Power: 0

16 Bit UUIDs: 0X111E 0X1108 0X110B 0X110D 0X110E

(packet 290)

A minor difference can be observed opposed to the Bose speaker. The class of the device is a Loudspeaker instead of headset.

Connection request

A connection request is send:

May 27 09:47:37.486 HCI Command 0x0000 70:99:1C:B5:31:21 **Create**

Connection- 70:99:1C:B5:31:21 SEND

Opcode: 0x0405 (OGF: 0x01 OCF: 0x05)

Parameter Length: 13 (0x0D)

Bluetooth Device Address: 70:99:1C:B5:31:21

Packet Type: 0xCC18

Page Scan Repetition Mode: 0x01

Page Scan Mode: 0x00

Clock Offset: 0xDF8F

Allow Role Switch: 0x00

May 27 09:47:37.581 HCI Event 0x000B 70:99:1C:B5:31:21 **Connection**

Complete - 70:99:1C:B5:31:21 RECV

(packet 327)

Information, SDP and Authentication

Nothing different happens compared to the Bose speaker capture.

Encryption

Now something interesting happens, the JBL speaker seems to have more security implemented. Following the packets there can be seen that more traffic happens opposed to the Bose speaker:

May 27 09:47:37.687 HCI Command 0x0000 70:99:1C:B5:31:21 Link Key Request Negative Reply - 70:99:1C:B5:31:21 SEND
May 27 09:47:37.688 HCI Event 0x0000 70:99:1C:B5:31:21 Command Complete [040C] - Link Key Request Negative Reply - 70:99:1C:B5:31:21 RECV
May 27 09:47:37.689 HCI Event 0x0000 70:99:1C:B5:31:21 IO Capability Request - 70:99:1C:B5:31:21 RECV
May 27 09:47:37.689 HCI Command 0x0000 70:99:1C:B5:31:21 IO Capability Request Reply - 70:99:1C:B5:31:21 SEND
May 27 09:47:37.689 HCI Event 0x0000 Command Complete [042B] - IO Capability Request Reply RECV
May 27 09:47:37.696 HCI Event 0x0000 70:99:1C:B5:31:21 IO Capability Response - 70:99:1C:B5:31:21 RECV
May 27 09:47:37.878 HCI Event 0x0000 70:99:1C:B5:31:21 User Confirmation Request - 70:99:1C:B5:31:21 RECV
May 27 09:47:37.878 HCI Command 0x0000 70:99:1C:B5:31:21 User Confirmation Request Reply - 70:99:1C:B5:31:21 SEND
May 27 09:47:37.932 HCI Event 0x0000 Command Complete [042C] - User Confirmation Request Reply RECV
May 27 09:47:37.932 HCI Command 0x0000 70:99:1C:B5:31:21 User Confirmation Request Reply - 70:99:1C:B5:31:21 SEND
May 27 09:47:37.933 HCI Event 0x0000 Command Complete [042C] - User Confirmation Request Reply RECV May 27 09:47:38.373 HCI Event 0x0000 70:99:1C:B5:31:21 Simple Pairing Complete- 70:99:1C:B5:31:21 RECV
May 27 09:47:38.381 HCI Event 0x0000 70:99:1C:B5:31:21 Link Key Notification - 70:99:1C:B5:31:21 RECV May 27 09:47:38.467 HCI Command 0x000B 70:99:1C:B5:31:21 Set Connection Encryption - 0x01 - Connection Handle: 0x000B SEND May 27 09:47:38.468 HCI Event 0x0000 Command Status - Set Connection Encryption RECV May 27 09:47:38.482 HCI Event 0x000B 70:99:1C:B5:31:21 Encryption Change Complete - Encryption Enabled RECV
(packet 521)

As can be seen there are more security implementations, because of secure simple pairing that has been implemented correctly. However, a MITM will still work since the user is not provided with any information indication this is the correct device to connect to, other then the name.

Final capture

With the setting up of communication, the sending data and the disconnection there are no real differences to the earlier shown capture.

4.1.3 Myriad speaker

The Myriad speaker was similar in terms of Bluetooth implementation like the JBL speaker. Once again the BR/EDR version of Bluetooth is used and the association model Just Works. From the capture follows that the Bluetooth adress of the speaker is [DF:A0:3F:95:12:B8](#).

4.2 MITM results

With the results of the information retrieval phase the attack could successfully be implemented. The results of the attack follow in this section, but the full results are in the Appendix A.

4.2.1 Flow

The command that was run:

```
sudo btproxy 72:5A:FD:3B:3C:9D 08:E4:FD:B6:A7:D5
```

Gave as result:

Running proxy on master 72:5A:FD:3B:3C:9D and slave 08:E4:FD:B6:A7:D5

Using shared adapter

Slave adapter: hci0

Master adapter: hci0

Looking up info on slave (08:E4:FD:B6:A7:D5)

Looking up info on master (72:5A:FD:3B:3C:9D)

Spoofing master name as s2025_btproxy

Running inquiry scan

paired

Spoofing master name as s2025_btproxy

Proxy listening for connections for "Serial Port Server Port 1"

Proxy listening for connections for "Audio Remote Control"

Attempting connections with 2 services on slave

Connected to service "Audio Remote Control"

Connected to service "Serial Port Server Port 1"

Now you're free to connect to "s2025_btproxy" from master device.

The setup took about 25 seconds. After the connection request from the controller is send:

Accepted connection from (72:5A:FD:3B:3C:9D , 1)

b'\x00\x00\x00\x00\x00\x00\x00\x00\x00'

4.2.2 Example of changing name

For the purpose of this research, the name of the Bluetooth host is changed, so that the attacker can see that it was working. This results in figure 4.1. In a real a attack the name would be kept the same.

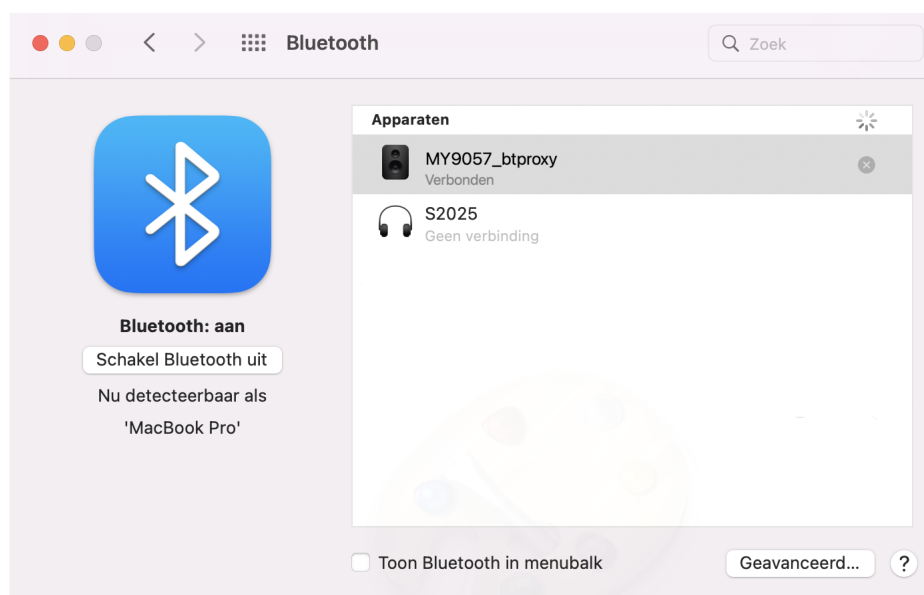


Figure 4.1: An example of how the proxy changed the Bluetooth connection.

Chapter 5

Related Work

5.1 MITM on Bluetooth

The research conducted in this bachelor thesis mainly describes the practical Man-in-the-Middle attack on the Bluetooth connection to speakers. Bluetooth has been subject to MITM attacks before and quite some research has been conducted on these offensives.

An active man in the middle attack on Bluetooth smart devices

Tal Melamed showed the effectiveness of such an MITM attack[18]. With his research he performed a practical attack on a BLE smart device. Not only was he successful in sending commands and thus taking control of the Bluetooth smart devices, which was a smart sports bracelet, in his own words: "by performing a MitM attack, hackers can even control from remote the mobile device used to communicate with the Bluetooth smart device." This thus means that not only the smart device was at risk. Also the controllers device could be taken control of. The end conclusions of the research is that there could definitely be some more improvements to security of BLE.

A take-away from this paper is that MITM attacks are feasible in practice. Although the researchers main purpose is to attack BLE, this shows promise that MITM attacks are possible on Bluetooth. What our research contributes to the paper is that instead of smart devices, specifically speakers is the area of focus. Also, on the contrary of just Bluetooth Low Energy, every Bluetooth version was possible, which in the end meant that BR/EDR was extensively tested and an MITM attack is possible.

5.2 Bluetooth weaknesses

Important in the research of this bachelor thesis are the weaknesses of Bluetooth that can be exploited. Earlier research papers indicate that these weaknesses are certainly present not only in the Bluetooth protocol itself, but also in the implementation of it in devices.

Happy MitM

Such implementation flaws have been investigated extensively by Jiska Classen and Matthias Hollick[9]. Their paper describes the issue of major Bluetooth stacks not warning the user of reusing earlier determined shared keys. Bluetooth specification clearly states that such warnings should be in place. This flaw of implementation sets the user at risk to "recently published and potential future security issues in Bluetooth authentication and encryption." The research paper thoroughly describes the possibility for a MITM attack on each major Bluetooth stack. What our research paper contributes to the research conducted earlier is that the main focus lies on audio devices, specifically speakers. Instead of analysing the specific weakness, the practical attack is analysed upon further.

BLURtooth: Exploiting Cross-Transport Key Derivation in Bluetooth Classic and Bluetooth Low Energy

As the title states a certain weakness in the implementation of Cross-Transport Key Derivation (CTKD) within BT and BLE can be exploited[3]. Keys can be overwritten mid transport, so again the MITM principle gets another functionality, namely, exploiting the CTKD weakness. Like in Happy MITM, Daniele Antonioli et al., are researching whether the found weakness is exploitable on the Bluetooth chips from popular vendors.

The research showed once more that Bluetooth is not perfect and the implementation of it has vulnerabilities that can be exploited. This thesis is different from the research, since this research main goal is a practical attack, while BLURtooth focusses more on the vulnerability of behind the attack. Furthermore, our research contributes by having a more specific target, instead of targeting all major systems like BLURtooth.

Chapter 6

Conclusions

In this research it can be seen that a practical attack on the Bluetooth speakers is indeed possible. Chapter 4 shows a successful attack on each of the speakers. The aim of this research was to find out in what way it was possible to sever the connection between a users device and a Bluetooth speaker, or alter the information exchanged between them, by exploiting weaknesses in the Bluetooth protocol.

The first sub goal was to discover whether there was such a difference in the implementation of Bluetooth in various types of speakers, that the user notices these differences while using the speakers. Following the capture of the speakers, some minor distinctions can be noticed. For example the JBL speaker has more security features implemented. However, these differences were not significant enough to distinct behavior per speaker. All speakers followed the same association model, which made the differences unnoticeable in behavior.

The weakness of the implementation of the speakers was that the user had no indication, but the name, whether the correct host was connected to. This allowed the same MITM attack to be possible on three speakers, which means that the attack script worked for each of the speakers and no alterations had to be made to the script per speaker.

To conclude, the information between the host and controller could be changed by a Man-in-the-Middle attack. This attack worked, since the user does not know whether a connection is happening to the real host, or to an impersonator.

Chapter 7

Future work

With the conclusions drawn from this research more subjects become interesting to investigate in the future. This chapter describes interesting works to look upon further.

7.1 Attack prevention

This research paper describes a weakness that makes it easy to perform the MITM attack. An interesting subject to investigate in the future would be to experiment with new security implementations in Bluetooth, for example setting passwords, to make this attack harder, or even render the weakness obsolete. This can be done by taking for example popular MITM prevention methods and testing different implementations in Bluetooth by letting real people use the implementation. The flow of usage could then be analyzed to check if such a prevention method would work for Bluetooth.

7.2 BlueZ for non Linux users

The MITM attack could only be used on Linux, due to deprecated libraries and tools on MacOS and Windows. Even on Linux, the version of BlueZ had to be set back in order to make the tool work. Interesting work to be looked at, would be to upgrade the Bluetooth proxy to be able to use the newer Bluetooth tools. Another thing to look at would be to make BlueZ work again on MacOS or to create a BlueZ version for windows. This would be partly research, but mostly a development project.

7.3 Speeding up

The tool now takes quite some time to create the proxy, which means the user is very likely to be already connected before the proxy is actually set

up. The MITM tool could definitely benefit from future work researching how actions could be sped up.

7.4 Automation

In order to run the btproxy tool, the Bluetooth addresses have to be known to the attacker. The attacker has to send a separate inquiry to retrieve those addresses. This costs time during the attack, which could be saved by creating an automation script that performs the inquiry automatically and then runs the tool with the information that the inquiry provided. Future work could be looking at developing such a script, testing what is most efficient.

Another automation option would be to provide automatic input from audio streams played on the proxy's device. The data that comes in from the speakers should be automatically be replaced by such an audio stream. An interesting future research to implement more functionality and automation to the btproxy tool.

Bibliography

- [1] agarcia. Modernize LightAquaBlue.xcodeproj · agarcia-rd/pybluez@f7e2398. <https://github.com/agarcia-rd/pybluez/commit/f7e2398e2bfe5a7ff3b04c5042c61fe782bcd9bfc>, January 2019.
- [2] Marwan Ali Albahar. EBSCOhost | 119727681 | TOWARDS ENHANCING JUST WORKS MODEL IN BLUETOOTH PAIRING., 2016.
- [3] Daniele Antonioli. BLURtooth | Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, January 2022.
- [4] Apple. AirDrop gebruiken op iPhone, iPad of iPod touch. <https://support.apple.com/nl-nl/HT204144>, January 2022.
- [5] Apple. Xcode - Apple Developer. <https://developer.apple.com/download/all/?q=xcode>, March 2022.
- [6] Axway. Advertised ciphers and cipher suites in FIPS mode. https://docs.axway.com/bundle/SecureTransport_54_AdministratorGuide_allOS_en_HTML5/page/Content/AdministratorsGuide/FIPS_transfer_mode/rest_required_ciphers_cipher_suites.htm, January 2022.
- [7] Bluetooth. Core Specifications | Bluetooth® Technology Website. <https://www.bluetooth.com/specifications/bluetooth-core-specification/>, December 2019.
- [8] Bluetooth. Bluetooth Technology Overview, 2022.
- [9] Jiska Classen and Matthias Hollick. Happy MitM: Fun and Toys in Every Bluetooth Device. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 72–77, June 2021. arXiv:2108.07190 [cs].
- [10] John Cox. How Bluetooth got as fast as Wi-Fi, April 2019.

- [11] Keijo Haataja and Pekka Toivanen. Practical Man-in-the-Middle Attacks Against Bluetooth Secure Simple Pairing. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–5, October 2008. ISSN: 2161-9654.
- [12] HYPR. Whats is the Federal Information Processing Standard (FIPS)? <https://www.hypr.com/federal-information-processing-standard/>, March 2020.
- [13] Imperva. What is MITM (Man in the Middle) Attack | Imperva. <https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/>, January 2022.
- [14] Apple Inc. Xcode. <https://developer.apple.com/xcode/>, March 2022.
- [15] kali. Get Kali. <https://www.kali.org/get-kali/>, January 2022.
- [16] Linux. hcitool(1): configure Bluetooth connections - Linux man page. <https://linux.die.net/man/1/hcitool>.
- [17] Chendong Liu, Yilin Zhang, and Huanyu Zhou. A Comprehensive Study of Bluetooth Low Energy. *Journal of Physics: Conference Series*, 2093(1):012021, November 2021.
- [18] Tal Melamed. An active man-in-the-middle attack on bluetooth smart devices. *International Journal of Safety and Security Engineering*, 8:200–211, February 2018.
- [19] John M Meredith. Specification # 25.321, April 2017.
- [20] oracle. Downloads – Oracle VM VirtualBox. <https://www.virtualbox.org/wiki/Downloads>, January 2022.
- [21] oracle. Oracle VM VirtualBox. <https://www.virtualbox.org/>, January 2022.
- [22] Aleksandr Ovechkin, Tim Claeys, Dries Vanoost, John F. Dawson, Guy A. E. Vandenbosch, and Davy Pissoort. Characterizing the Robustness of Wi-Fi and Bluetooth against Continuous Wave EM Disturbances inside a Reverberation Chamber. In *2021 IEEE International Joint EMC/SI/PI and EMC Europe Symposium*, pages 1031–1036, July 2021.
- [23] John Padgett, John Bahr, Mayank Batra, Marcel Holtmann, Rhonda Smithbey, Lily Chen, and Karen Scarfone. Guide to Bluetooth Security. Technical report, National Institute of Standards and Technology, January 2022.
- [24] Conor Patrick. Btproxy. <https://github.com/conorpp/btproxy>, May 2022. original-date: 2015-08-20T02:33:16Z.

- [25] PyBluez. PyBluez — PyBluez master documentation. <https://pybluez.readthedocs.io/en/latest/>, January 2019.
- [26] SamHaeck. Btproxy. <https://github.com/SamHaeck/btproxy>, April 2022. original-date: 2022-04-25T15:48:28Z.
- [27] K. Saravanan, L. Vijayanand, and R. K. Negesh. A Novel Bluetooth Man-In-The-Middle Attack Based On SSP using OOB Association model, March 2012. Number: arXiv:1203.4649 arXiv:1203.4649 [cs].
- [28] Fazli Subhan, Halabi Hasbullah, Azat Rozyyev, and Sheikh Tahir Bakhsh. Analysis of Bluetooth signal parameters for indoor positioning systems. In *2012 International Conference on Computer & Information Science (ICCIS)*, volume 2, pages 784–789, June 2012.
- [29] Da-Zhi Sun, Yi Mu, and Willy Susilo. Man-in-the-middle attacks on Secure Simple Pairing in Bluetooth standard V5.0 and its countermeasure. *Personal and Ubiquitous Computing*, 22(1):55–67, February 2018.
- [30] karl torvmark. Three flavors of Bluetooth®: Which one to choose? page 6, 2014.
- [31] Wireshark. Wireshark · Go Deep. <https://www.wireshark.org/>, March 2022.

Appendix A

Appendix

A.1 Packet Captures

A.1.1 Bose Box

download BOSE packet capture
download BOSE packet capture text file

A.1.2 JBL Box

download JBL packet capture text file download JBL packet capture text file

A.1.3 Myriad Box

download Myriad packet capture text file download Myriad packet capture text file

A.1.4 Kali linux prepared virtual Box

vmware kali.ova

A.2 Hardware information

A.2.1 MacBook Pro

Hardwareoverzicht:

Modelnaam:	MacBook Pro
Model-ID:	MacBookPro16,2
Processornaam:	Quad-Core Intel Core i5
Processorsnelheid:	2 GHz
Aantal processors:	1
Totale aantal cores:	4
L2-cache (per core):	512 KB


```

L3-cache:      6 MB
Hyperthreading-technologie:  Ingeschakeld
Geheugen:      16 GB
Versie systeemfirmware:      1715.81.2.0.0 (iBridge:
    19.16.10744.0.0,0)
Versie OS-lader:      540.80.2~11
Serienummer (systeem):      C02DX1KTML7L
Hardware-UUID:      B1D91248-81D9-5CA8-A470-FBC5E2E29BDE
UDID voor provisioning:      B1D91248-81D9-5CA8-A470-
    FBC5E2E29BDE
Status activeringsslot:      Ingeschakeld

Bluetooth-controller:
Adres:          14:7D:DA:E3:F7:67
Status:         Aan
Chipset:        BCM_4364B3
Detecteerbaar:  Uit
Firmwareversie: v75 c4195
Productcode:    0x0001
Ondersteunde apparaten:      0x382039 < HFP AVRCP A2DP HID
    Braille AACP GATT Serial >
Transport:      UART
Fabrikantcode:  0x004C (Apple)
Gekoppelde Bluetooth-apparaten:
    S2025:
        Adres:      08:E4:FD:B6:A7:D5
        Subtype:     Headset

```

A.3 Bluetooth security figures

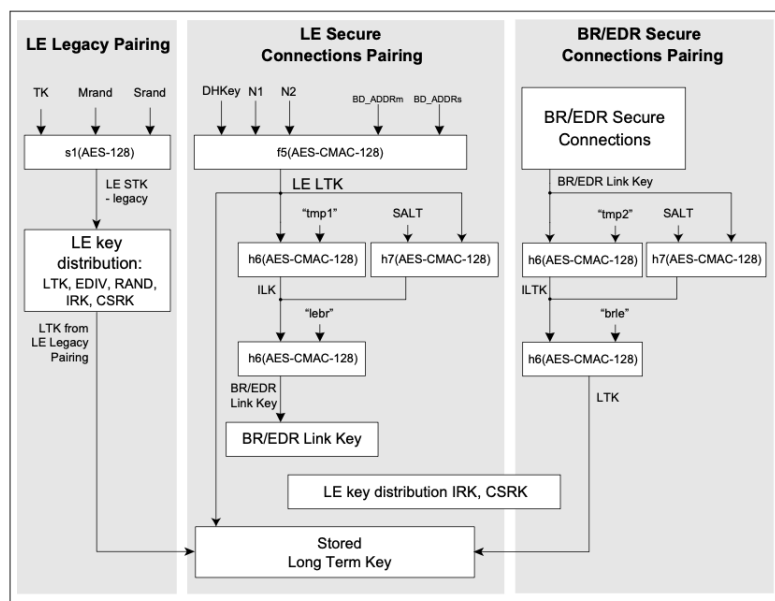


Figure A.1: Low Energy keys