AMERICAN UNIVERSITY OF BEIRUT: DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Fingerprinting with DHCP: Attack and Detection tool

Hassan Nasser
Rashad Saab
Souraya Nasser
October 12, 2015

1 INTRODUCTION

DHCP is the Dynamic Host Configuration Protocol that is used to distribute IP addresses to users in the network. When the user wants to connect to the network the device has no IP. In addition to that, it doesn't have the IP of the DHCP server in order to request an IP from it. So the user sends a broadcast discover message to all the users in the network to discover the server. The server replies with a broadcast offer message offering an IP to the user. The user accepts the IP with a broadcast request message. The server then sends the configuration information in an acknowledgment message. This procedure has a lot of security implications. The messages sent are broadcast messages since the user has no IP which allows for others to sniff these messages and have access to the information they carry. This report will discuss how to use DHCP packets to learn the OS of the device as well as a way to detect this fingerprinting.

2 DHCP FINGERPRINTING

2.1 CONCEPT

The DHCP client send packets to the DHCP server to obtain a unique IP address and other important networking information that include information about the hardware or operating system of the device as well as other information. These packets are defined by RFC 2132.

The use of DHCP options is vendor, device, and OS dependent, which creates significant differences in the DHCP packets generated by various devices. Thus, these options constitute a fingerprint that can determine the OS not only for Linux, Android, and IOS but also for many others OS. Every device in dynamic IP mode starts its connection by DHCP request packet to acquire an IP. This made reading the (55 decimal) option in the DHCP Request packet a fingerprint of the operating system. Our software is based on this concept. It gives good results by distinguishing between Android, IOS, and Linux (including version of Linux). In order to get the fingerprint three steps should be implemented:

1. De-authentication Attack
2. Parsing The Packets
3. Compare Fingerprint to Those in the Table

2.2 ENVIRONMENT AND PROGRAMS USED

Two programs are implemented to execute these three steps. The program for the de-authentication attack is implemented in python and executed on the terminal with Linux as an Operating System. The command to execute it is

```
sudo python2 ~/Downloads/ScriptName.py
```

The second program is implemented in Microsoft Access Database. It has one table as well as a search function and a simple GUI to facilitate searching for fingerprints and OS's

2.3 DE-AUTHENTICATION ATTACK

The aim is to sniff the DHCP request packet. However, we can't know when will a user connect to the network. In addition to that, we cant read the previously sent request packets. Therefore, we need to force the user to request another IP and we do that by sending a de-authentication packet for all IPs on the network. That way, each device will send a DHCP Request packet including the 55 option which is OS dependent. Therefore, we will have access to the DHCP Request packet including the fingerprint.

The first program that is implements in python does the deauthentication attack and generates the fingerprint of the victim device. The program will operate in monitor mode on the most powerful wireless interface.After that it will look for all the access points and clients connected to these access points by hopping from one channel to another sequentially at a speed of 1 channel per second. While hopping from one channel to the other it identifies all the clients connected to it and it send three deauthentication packets.

1. From the Access Point to the client
2. From the client to the Access Point
3. From the Access Point sent as broadcast to de-authenticate all clients connected to the Access Point. This de-authentication packet can be ignored if the routers ignore broaodcast de-authentication packets (In this case sending it will be a waste of time).

The de-authentication packets will ignore the MAC address of the moniter mode wireless interface.

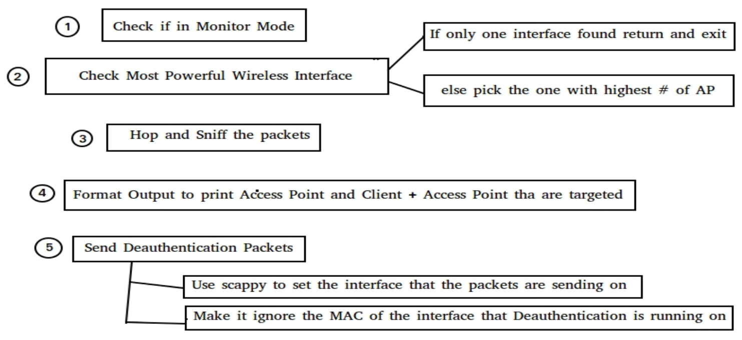
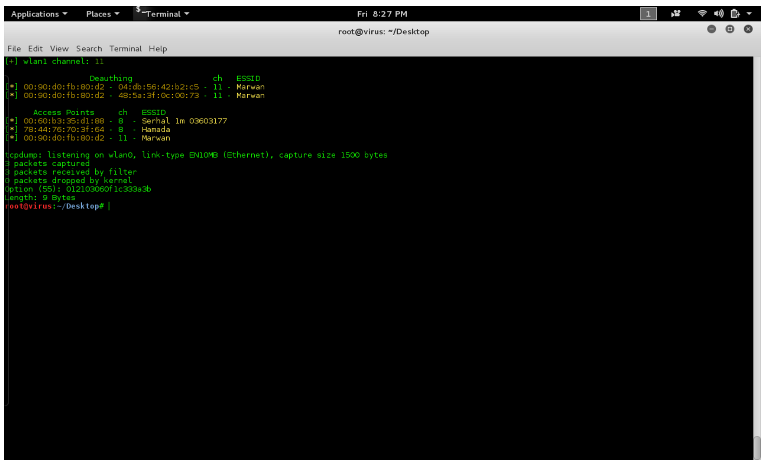

1
Check if in Monitor Mode
2
Check Most Powerful Wireless Interface
If only one interface found return and exit
else pick the one with highest # of AP
3
Hop and Sniff the packets
4
Format Output to print Access Point and Client + Access Point tha are targeted
5
Send Deauthentication Packets
Use scappy to set the interface that the packets are sending on
Make it ignore the MAC of the interface that Deauthentication is running on

Figure 2.1: Basic Steps Done By the Algorithm



```
[+] wlan1 channel: 11

                Deauthing                 ch   ESSID
[*] 00:90:d0:fb:80:d2 - 04:db:56:42:b2:c5 - 11 - Marwan
[*] 00:90:d0:fb:80:d2 - 48:5a:3f:0c:00:73 - 11 - Marwan

      Access Points     ch   ESSID
[*] 00:60:b3:35:d1:88 - 8  - Serhal 1m 03603177
[*] 78:44:76:70:3f:64 - 8  - Hamada
[*] 00:90:d0:fb:80:d2 - 11 - Marwan

tcpdump: listening on wlan0, link-type EN10MB (Ethernet), capture size 1500 bytes
3 packets captured
3 packets received by filter
0 packets dropped by kernel
Option (55): 012103060f1c333a3b
Length: 9 Bytes
root@virus:~/Desktop# |
```

Figure 2.2: Output of the Fingerprinting attack - Android Device

2.3.1 Parsing the Packet

After the clients are disconnected from the Access Points, they send request to the DHCP server to connect and the program sniffs and parses the requests for the option 55. After getting the DHCP request packet we parse it to reach the option field which has multiple options segments. Each option segment has variable length, this segment is formed by:

1. First Byte : option value that determines the option type. In our case the option of interest is option 55.

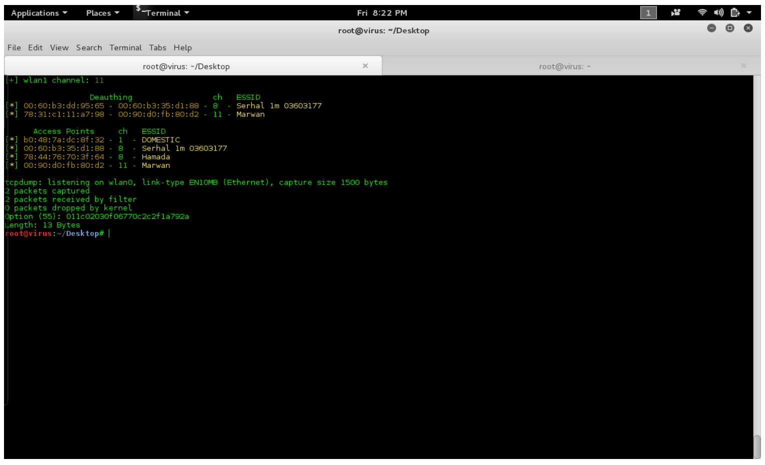

```
[+] wlan1 channel: 11

                 Deauthing                ch   ESSID
[*] 00:60:b3:dd:95:65 - 00:60:b3:35:d1:88 - 8  - Serhal 1m 03603177
[*] 78:31:c1:11:a7:98 - 00:90:d0:fb:80:d2 - 11 - Marwan

      Access Points     ch   ESSID
[*] b0:48:7a:dc:8f:32 - 1  - DOMESTIC
[*] 00:60:b3:35:d1:88 - 8  - Serhal 1m 03603177
[*] 78:44:76:70:3f:64 - 8  - Hamada
[*] 00:90:d0:fb:80:d2 - 11 - Marwan

tcpdump: listening on wlan0, link-type EN10MB (Ethernet), capture size 1500 bytes
2 packets captured
2 packets received by filter
0 packets dropped by kernel
Option (55): 011c02030f06770c2c2f1a792a
Length: 13 Bytes
root@virus:~/Desktop# |
```

Figure 2.3: Output of the Fingerprinting attack - Linux Kali Device

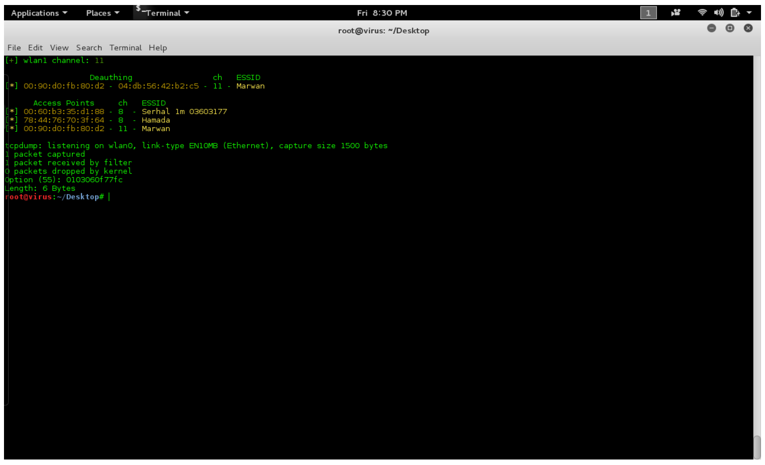
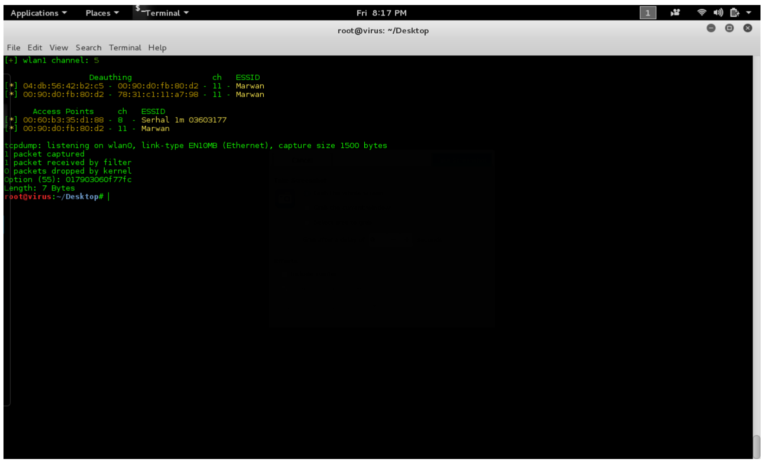

```
[+] wlan1 channel: 11

                 Deauthing                ch   ESSID
[*] 00:90:d0:fb:80:d2 - 04:db:56:42:b2:c5 - 11 - Marwan

      Access Points     ch   ESSID
[*] 00:60:b3:35:d1:88 - 8  - Serhal 1m 03603177
[*] 78:44:76:70:3f:64 - 8  - Hamada
[*] 00:90:d0:fb:80:d2 - 11 - Marwan

tcpdump: listening on wlan0, link-type EN10MB (Ethernet), capture size 1500 bytes
1 packet captured
1 packet received by filter
0 packets dropped by kernel
Option (55): 0103060f77fc
Length: 6 Bytes
root@virus:~/Desktop# |
```

Figure 2.4: Output of the Fingerprinting attack - iOS 8 Device

2. Second Byte: option segment length L

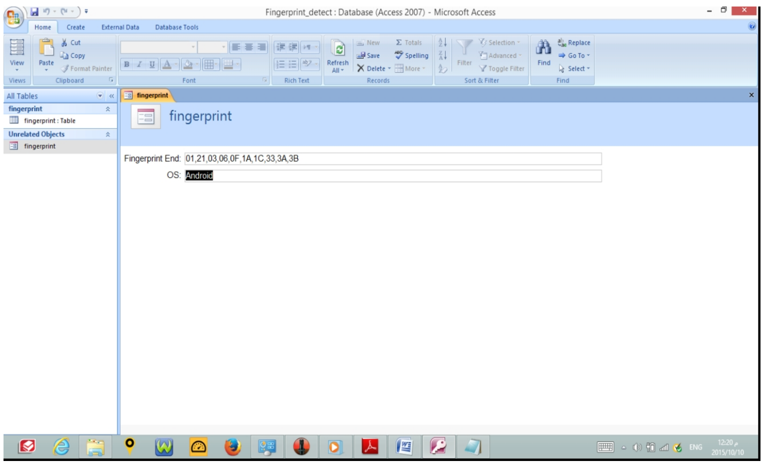
3. L Bytes: L bytes containing the option values

Option 55 is not always the first so the program should parse the document and discard everything that located before this option. Once this option is reached it reads the L Bytes. Finally the program formats the data and prints them to the terminal.Using another program, one can compares the option's end to the table of fingerprints provided in the Appendix. This program is implemented in Microsoft Access Database and contains a a table for the fingerprints as well as the corresponding OS. It has a simple search function that takes the fingerprint as an input and gives the corresponding OS of the device as well as the version if it exists. The input should have the following format: option 55 in two digit hexadecimal separated with commas without any space.



```
[+] wlan1 channel: 5

                Deauthing                ch   ESSID
[*] 04:db:56:42:b2:c5 - 00:90:d0:fb:80:d2 - 11 - Marwan
[*] 00:90:d0:fb:80:d2 - 78:31:c1:11:a7:98 - 11 - Marwan

      Access Points     ch   ESSID
[*] 00:60:b3:35:d1:88 - 8  - Serhal 1m 03603177
[*] 00:90:d0:fb:80:d2 - 11 - Marwan

tcpdump: listening on wlan0, link-type EN10MB (Ethernet), capture size 1500 bytes
1 packet captured
1 packet received by filter
0 packets dropped by kernel
Option (55): 017903060f77fc
Length: 7 Bytes
root@virus:~/Desktop#
```

Figure 2.5: Output of the Fingerprinting attack - iOS 9 Device



fingerprint

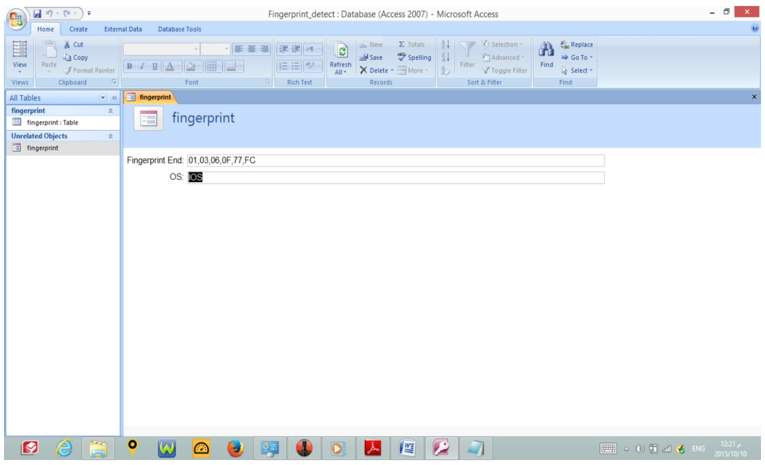
Fingerprint End: 01,21,03,06,0F,1A,1C,33,3A,3B

OS: Android

Figure 2.6: Table of Fingerprints - Android Device Detected

3 DETECTING DHCP FINGERPRINTING

This fingerprinting attack is based on the fact that de-authentication packets can be sent to all the clients in the network. So to detect this attack one can use a static IP on a laptop and run a small script using python that detects the de-authentication packets that are transmitted. The IP address of the device has to be static so that it is not affected by the de-authentication attack. In addition to that, there should be a way to determine the number of clients connected to the DHCP server so that this number is compared to the number of de-authenticated devices. This is tricky since there is no way of telling if de-authentication packets are real or are due to an attack. Also the number of users can change with time. One solution can be to find the number of connected clients every small interval of time (example: 1 second since its not so probable for the number of clients to change significantly in one second) and then when it

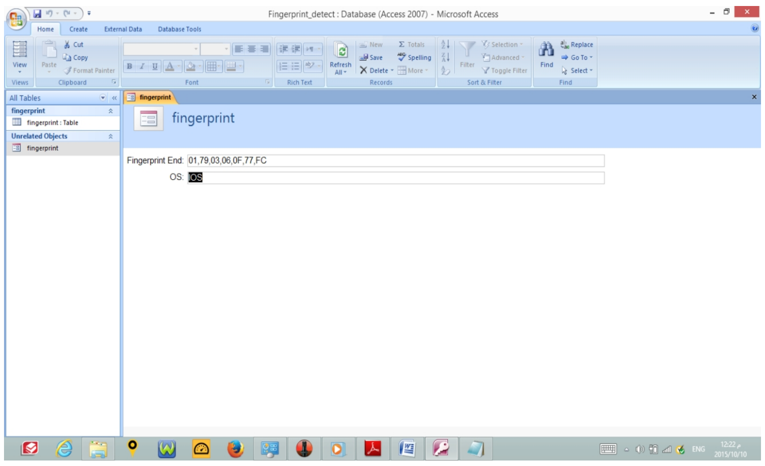

Fingerprint_detect : Database (Access 2007) - Microsoft Access

fingerprint

Fingerprint End: 01,03,06,0F,77,FC

OS: IOS

Figure 2.7: Table of Fingerprints - IOS Device Detected


Fingerprint_detect : Database (Access 2007) - Microsoft Access

fingerprint

Fingerprint End: 01,79,03,06,0F,77,FC

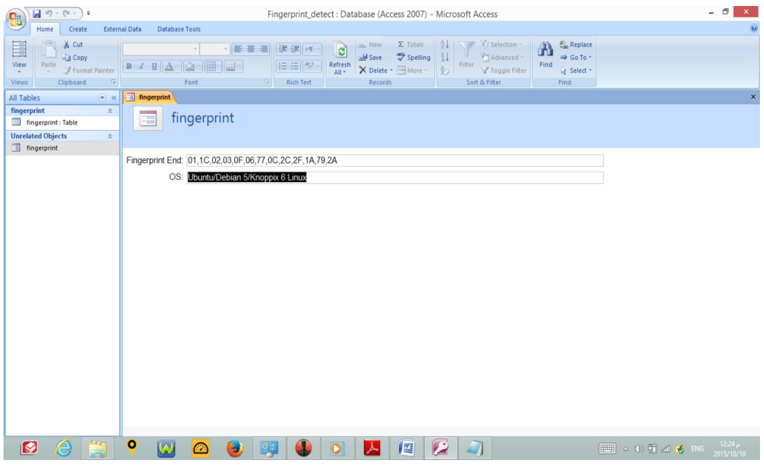
OS: IOS

Figure 2.8: Table of Fingerprints - IOS Device Detected

receives de-authentication packets it would find the number of devices disconnected and compare it to the number of clients. The numbers will not be equal but they will be so close if it was an attack. Therefore one can set a threshold for the de-authentication to be considered an attack (example: 99 percent of the clients are being de-authenticated from the network).

The script to find the number of clients connected on the Access Points is used in the script for the attack. It is the same code that is used to hop from one channel to the other discovering the devices but with a counter that increments every time it finds a client connected. Scapy was used in the python script to detect the deauthentication packets. The script for the detection tool is implemented in python and executed on the terminal with Linux as an Operating System. The command to execute it is

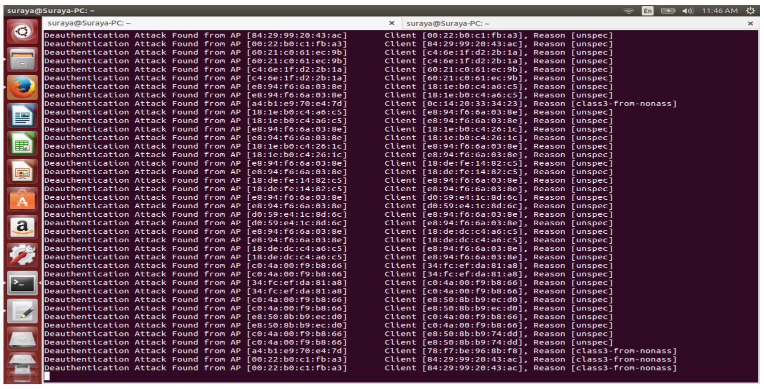
```
sudo python2 ~/Downloads/ScriptName.py
```



Fingerprint End: 01,1C,02,03,0F,06,77,0C,2C,2F,1A,79,2A

OS: Ubuntu/Debian 5/Knoppix 6 Linux

Figure 2.9: Table of Fingerprints - Ubuntu/Debian 5/Knoppix 6 Linux Device Detected



```
Deauthentication Attack Found from AP [84:29:99:20:43:ac]       Client [00:22:b0:c1:fb:a3], Reason [unspec]
Deauthentication Attack Found from AP [00:22:b0:c1:fb:a3]       Client [84:29:99:20:43:ac], Reason [unspec]
Deauthentication Attack Found from AP [60:21:c0:61:ec:9b]       Client [c4:6e:1f:d2:2b:1a], Reason [unspec]
Deauthentication Attack Found from AP [60:21:c0:61:ec:9b]       Client [c4:6e:1f:d2:2b:1a], Reason [unspec]
Deauthentication Attack Found from AP [c4:6e:1f:d2:2b:1a]       Client [60:21:c0:61:ec:9b], Reason [unspec]
Deauthentication Attack Found from AP [c4:6e:1f:d2:2b:1a]       Client [60:21:c0:61:ec:9b], Reason [unspec]
Deauthentication Attack Found from AP [e8:94:f6:6a:03:8e]       Client [18:1e:b0:c4:a6:c5], Reason [unspec]
Deauthentication Attack Found from AP [e8:94:f6:6a:03:8e]       Client [18:1e:b0:c4:a6:c5], Reason [unspec]
Deauthentication Attack Found from AP [a4:b1:e9:70:e4:7d]       Client [0c:14:20:33:34:23], Reason [class3-from-nonass]
Deauthentication Attack Found from AP [18:1e:b0:c4:a6:c5]       Client [e8:94:f6:6a:03:8e], Reason [unspec]
Deauthentication Attack Found from AP [18:1e:b0:c4:a6:c5]       Client [e8:94:f6:6a:03:8e], Reason [unspec]
Deauthentication Attack Found from AP [e8:94:f6:6a:03:8e]       Client [18:1e:b0:c4:26:1c], Reason [unspec]
Deauthentication Attack Found from AP [e8:94:f6:6a:03:8e]       Client [18:1e:b0:c4:26:1c], Reason [unspec]
Deauthentication Attack Found from AP [18:1e:b0:c4:26:1c]       Client [e8:94:f6:6a:03:8e], Reason [unspec]
Deauthentication Attack Found from AP [18:1e:b0:c4:26:1c]       Client [e8:94:f6:6a:03:8e], Reason [unspec]
Deauthentication Attack Found from AP [e8:94:f6:6a:03:8e]       Client [18:de:fe:14:82:c5], Reason [unspec]
Deauthentication Attack Found from AP [e8:94:f6:6a:03:8e]       Client [18:de:fe:14:82:c5], Reason [unspec]
Deauthentication Attack Found from AP [18:de:fe:14:82:c5]       Client [e8:94:f6:6a:03:8e], Reason [unspec]
Deauthentication Attack Found from AP [18:de:fe:14:82:c5]       Client [e8:94:f6:6a:03:8e], Reason [unspec]
Deauthentication Attack Found from AP [e8:94:f6:6a:03:8e]       Client [d0:59:e4:1c:8d:6c], Reason [unspec]
Deauthentication Attack Found from AP [e8:94:f6:6a:03:8e]       Client [d0:59:e4:1c:8d:6c], Reason [unspec]
Deauthentication Attack Found from AP [d0:59:e4:1c:8d:6c]       Client [e8:94:f6:6a:03:8e], Reason [unspec]
Deauthentication Attack Found from AP [d0:59:e4:1c:8d:6c]       Client [e8:94:f6:6a:03:8e], Reason [unspec]
Deauthentication Attack Found from AP [e8:94:f6:6a:03:8e]       Client [18:de:dc:c4:a6:c5], Reason [unspec]
Deauthentication Attack Found from AP [e8:94:f6:6a:03:8e]       Client [18:de:dc:c4:a6:c5], Reason [unspec]
Deauthentication Attack Found from AP [18:de:dc:c4:a6:c5]       Client [e8:94:f6:6a:03:8e], Reason [unspec]
Deauthentication Attack Found from AP [18:de:dc:c4:a6:c5]       Client [e8:94:f6:6a:03:8e], Reason [unspec]
Deauthentication Attack Found from AP [c0:4a:00:f9:b8:66]       Client [34:fc:ef:da:81:a8], Reason [unspec]
Deauthentication Attack Found from AP [c0:4a:00:f9:b8:66]       Client [34:fc:ef:da:81:a8], Reason [unspec]
Deauthentication Attack Found from AP [34:fc:ef:da:81:a8]       Client [c0:4a:00:f9:b8:66], Reason [unspec]
Deauthentication Attack Found from AP [34:fc:ef:da:81:a8]       Client [c0:4a:00:f9:b8:66], Reason [unspec]
Deauthentication Attack Found from AP [c0:4a:00:f9:b8:66]       Client [e8:50:8b:b9:ec:d0], Reason [unspec]
Deauthentication Attack Found from AP [c0:4a:00:f9:b8:66]       Client [e8:50:8b:b9:ec:d0], Reason [unspec]
Deauthentication Attack Found from AP [e8:50:8b:b9:ec:d0]       Client [c0:4a:00:f9:b8:66], Reason [unspec]
Deauthentication Attack Found from AP [e8:50:8b:b9:ec:d0]       Client [c0:4a:00:f9:b8:66], Reason [unspec]
Deauthentication Attack Found from AP [c0:4a:00:f9:b8:66]       Client [e8:50:8b:b9:74:dd], Reason [unspec]
Deauthentication Attack Found from AP [c0:4a:00:f9:b8:66]       Client [e8:50:8b:b9:74:dd], Reason [unspec]
Deauthentication Attack Found from AP [a4:b1:e9:70:e4:7d]       Client [78:f7:be:96:8b:f8], Reason [class3-from-nonass]
Deauthentication Attack Found from AP [00:22:b0:c1:fb:a3]       Client [84:29:99:20:43:ac], Reason [class3-from-nonass]
Deauthentication Attack Found from AP [00:22:b0:c1:fb:a3]       Client [84:29:99:20:43:ac], Reason [class3-from-nonass]
```

Figure 3.1: Output of the Fingerprinting attack - Android Device

3.1 Conclusion

DHCP has a lot a vulnerabilities that can be used by attackers. The way DHCP works by sending broadcast messages allows others to sniff them and collect valuable information about the OS of the devices. The attack is definitely powerful since there is no good way to prevent it except by using static IP's. The detection method is not capable of detecting the attack accurately since there is no information that can tell if the traffic was real or if it was coming from an attack. Further studies of the network traffic and on how to know if it is due to an attack might help in making this detection tool more accurate.

4 Appendix

Fingerprint End	OS
1,121,33,3,6,15,28,51,58,59	Android
1,121,33,3,6,15,28,51,58,59,119	Android
1,3,6,15,28,33,51,58,59,121	Android
1,33,3,6,15,26,28,51,58,59	Android
1,33,3,6,15,28,51,58,59	Android
6,3,1,15,12,42	Android
1,121,33,3,6,12,15,26,28,51,54,58,59,119	Chrome OS Linux
1,121,33,3,6,12,15,26,28,51,54,58,59,119,252	Chrome OS Linux
1,33,3,6,28,51,54,58,59	Chrome OS Linux
1,28,2,121,15,6,12,40,41,42,26,119,3,121,249,33,42	Debian-based Linux
1,28,2,3,15,6,119,12	Debian-based Linux
1,28,2,3,15,6,119,12,44,47,26	Debian-based Linux
1,28,2,3,15,6,119,12,44,47,26,121,42,121,249,33,252,42	Debian-based Linux
1,28,2,3,15,6,12	Debian-based Linux
1,28,2,3,15,6,12,121,249,33,252,42	Debian-based Linux
1,28,2,3,15,6,12,42	Debian-based Linux
1,28,2,3,15,6,12,44,47	Debian-based Linux
1,28,2,3,15,6,12,44,47,26	Debian-based Linux
1,28,3,121,26,12,15,119,6,40,41,87,85,86,44,45,46,47,42,121,249,33,252,42	Debian-based Linux
3,6,15,28,12,7,9,42,48,49	Debian-based Linux
1,28,2,121,15,6,12,40,41,42,26,119,3	Fedora 14 based distro
1,28,2,121,15,6,12,40,41,42,26,119,3,121,249,252,42	Fedora 15 or 16 based distro Linux
1,121,33,3,6,12,15,28,42,51,54,58,59,119	Generic Linux
1,15,3,6,44,46,47,31,33,249,43,0,128,112	Generic Linux
1,28,2,121,15,6,12,40,41,42,26,119,3,121,249,33,252,42	Generic Linux

Fingerprint End	OS
1,28,2,3,15,6,12,121,249,252,42	Generic Linux
1,3,6,12,15,23,28,29,31,33,40,41,42	Generic Linux
1,3,6,12,15,23,28,29,31,33,40,41,42,9,7,200,44	Generic Linux
1,3,6,12,15,28,40,41,42,119	Generic Linux
3,6,12,15,17,23,28,29,31,33,40,41,42,119	Generic Linux
3,6,12,15,17,23,28,29,31,33,40,41,42,9,7,200,44	Generic Linux
1,121,33,3,6,12,15,26,28,42,51,54,58,59,119	Gentoo Linux
1,121,33,3,6,12,15,28,40,41,42,51,58,59,119	Gentoo Linux
1,3,6,12,15,17,23,28,29,31,33,40,41,42	Gentoo Linux
1,3,6,12,15,17,23,28,29,31,33,40,41,42,119	Gentoo Linux
58,59,1,28,121,33,3,12,119,15,6,40,41,42,26	Gentoo Linux
3,6,15,119,2	IOS
3,6,15,119,252	IOS
3,6,15,119,78	IOS
252,3,42,15,6,1,12	Meego Netbook Linux
58,59,1,28,121,33,3,12,119,15,6,40,41,42,26,17,120	Puppy Linux 4.x
1,28,2,3,15,6,12,40,41,42	RedHat/Fedora-based Linux
1,28,2,3,15,6,12,40,41,42,26	RedHat/Fedora-based Linux
1,28,2,3,15,6,12,40,41,42,26,119	RedHat/Fedora-based Linux
28,2,3,15,6,12,40,41,42	RedHat/Fedora-based Linux
1,28,2,121,15,6,12,40,41,42,26,119,3,121,249,42	RHEL 6.4 or Centos6.4 Linux
1,28,3,26,12,15,6,40,41,87,85,86,44,45,46,47,42	Suse Linux Enterprise Desktop 11
1,28,2,3,15,6,12,40,41	SUSE Linux/Novell

Fingerprint End	OS
1,3,6,12,15,17,23,28,29,31,33,40,41,42,9,7,200,44	SUSE Linux/Novell Desktop
1,3,6,12,15,17,23,28,29,31,33,40,41,42,9,7,44,45,46,47	SUSE Linux/Novell Desktop
1,3,6,12,15,17,23,28,29,31,33,40,41,42,9,7,44,45,46,47,119	SUSE Linux/Novell Desktop
1,28,2,3,15,6,119,12,44,47,26,121,42,121,249,252,42	Ubuntu 11.04 Linux
58,59,1,28,121,33,3,12,119,15,6,26,17,120	Ubuntu Linux Server 10.04 LTS
1,28,2,3,15,6,119,12,44,47,26,121	Ubuntu/Debian 5/Knoppix 6 Linux
1,28,2,3,15,6,119,12,44,47,26,121,42	Ubuntu/Debian 5/Knoppix 6 Linux

5 References

1. http://www.ietf.org/rfc/rfc2132.txt
2. http://danmcinerney.org/how-to-kick-everyone-around-you-off-wifi-with-python/
3. https://github.com/DanMcInerney/wifijammer/blob/master/wifijammer.py
4. https://hakfive.wordpress.com/2010/03/24/episode-706/
5. http://www.ietf.org/rfc/rfc2132.txt