Lab 5- Management Frame Packet Attack (De-Authentication)

Introduction

In this lab, we will be creating a management frame attack of deauthentication a client from the AP. Then, taking advantage of the client trying to reassociate with the AP and capturing the EAPOL packets that contain the password. This will bring us to the second part of the lab, where we will use the captured packets and bruteforce it using Aircrack-ng with a customized wordlist.

# Task 1 –

What is the BSSID of the access point?

**BSSID: C8:3A:35:02:76:30**

What is the channel of the access point?

**Channel: 6**

**AUTh = PSK**

What is the MAC address of the device that is already connected to the access point? Include screenshots of the results

**MAC Address: 0A:7C:AF:CA:B8:E6**

A screenshot of a computer

Description automatically generated

Include a screenshot of the command used to deauthenticate or the deauthentication process.

A screenshot of a computer

Description automatically generated

**Deauthentication command**

A screenshot of a computer

Description automatically generated

**Deauthentication Packets**

# Task 2 – Cracking Password

### Handshake

What does the first packet in the 4-way handshake authentication of WPA-PSK contain? Is it sent by the access point or the client device? What will the receiving device do with this information?

**The first packet is sent by the AP “TendaTec”. It contains the ANounce of the AP. The client receiving this will use it to construct a PTK.**

A screenshot of a computer

Description automatically generated

What does the second packet in the 4-way handshake authentication of WPA-PSK contain? What will the receiving device do with this information? Is it sent by the access point or the client device

**The second packet is sent from the client to the AP, it contains the SNounce and the MIC. The receiving AP will use it to construct the PTK.**

A screenshot of a computer

Description automatically generated

**Q 4.1.3 [10 pts]** What does the third packet in the 4-way handshake authentication of WPA-PSK contain? Is it sent by the access point or the client device? Include a screenshot of the packet.

**The third packet is sent by the AP to the client. It contains the GTK and MIC. “RSC” is the starting number of GTK.**

A screenshot of a computer

Description automatically generated

What does the fourth packet in the 4-way handshake authentication of WPA-PSK contain? Is it sent by the access point or the client device? Include a screenshot of the packet.

**The fourth packet is sent by the client to the AP. It is a ACK packet.**

A screenshot of a computer

Description automatically generated

### Password Cracking

What is the password? Include a screenshot of aircrack-ng with the password cracked.

Password: CY20sec23GMU

A computer screen shot of a computer

Description automatically generated

# Wordlist Generation

*In summary, generating all the possible combinations is the major idea. since we were given all the parameters, we just had to follow them to construct the wordlist. I first wrote all the words possible combinations of words. Sequentially, used a python code to generate all the possible combinations of upper and lower case for each letter - for each string. Further, used a python code that will incorporate the number through a double loop. Finally, tried brute forcing the pcap password with each wordlist I generated.*

Base Dictionary:

The password can be two of the three words; hence, it must be one of the following bases:

MASONCYSE

MASONGMU

CYSEMASON

CYSEGMU

GMUCYSE

GMUMASON

The math:

Since each letter can be either upper or lower case, then:

MASONCYSE = 9 letters

MASONGMU = 8 letters

CYSEMASON = 9 letters

CYSEGMU = 7 letters

GMUCYSE = 7 letters

GMUMASON = 8 letters

The total possibilities for upper and lower case for each base are the following:

2^9 = 512

2^8 = 256

2^7 = 128

2(512) + 2(256) +2(128) = 1792 total possibilities for upper and lower cases for the base words.

I used ChatGPT to write me the following code that will print me all the combinations for upper and loser cases. I ran the code for each base string.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Python file**:

casesLAB5.py

**Output files**:

masoncyse.txt

masongmu.txt

cysemason.txt

cysegmu.txt

After checking the outputs of each string, I checked that total output matches the predicted combinations calculations made earlier.

The words will always have these numbers:

1. 20
2. 21
3. 22
4. 23
5. 24

Now, since we have 512 variations for MASONCYSE upper and lower cases. We also must calculate the possibility of variation of each line with these numbers. Using premutation formula (order sensitive; meaning that (20,21) is a different variation from (21,20)), I found that there are 20 variations:

(20, 21)

(20, 22)

(20, 23)

(20, 24)

(21, 20)

(21, 22)

(21, 23)

(21, 24)

(22, 20)

(22, 21)

(22, 23)

(22, 24)

(23, 20)

(23, 21)

(23, 22)

(23, 24)

(24, 20)

(24, 21)

(24, 22)

(24, 23)

Now we consider the different index position in the base word index.

[1]M[2]A[3]S[4]O[5]N[6]C[7]Y[8]S[9]E[10]

For each number there are 10 different index values, but we have two numbers. So, the variations are much greater than that.

If we calculate the total number of these numbers with one variation of ‘MASONCYSE’ then the total possibilities will be 10,626,577,920.

We need to do the same for each of the 512 variations.

10,626,577,920 \* 512 = 5.4408079e+12

So, using ChatGPT, I used a python code that double loop around the string with the number’s permutations. I ran the strings lists for each base dictionary through this code to get the final list file for each string. The output from here will be the final wordlist used in Aircrack-ng.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

**Python file**:

numbersLAB5.py

**Output files:**

Cysegmu\_output.lst

Cysemason\_output.lst

Masoncyse\_output.lst

Masongmu\_output.lst

# Cracking Attempts

**String 1: Mason Cyse**

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

**String 2: Mason Gmu**

A computer screen shot of a computer code

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

**String 3: CYSEMASON**

A screen shot of a computer

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

**String 4: CYSEGMU**

A computer screen with white text

Description automatically generated

A computer screen shot of a computer

Description automatically generated

Password: CY20sec23GMU

Bonus – decrypting the captured packets using the password gained.

If the client had any communication, we would have been able to view them decrypt them and read them in plaintext. However, we did not continue to capture any data after getting the EAPOL. The device also didn’t have internet access, so we wouldn’t have captured any important data in this case. This might have been more beneficial in other scenarios.

A screenshot of a computer

Description automatically generated