OFFENSIVE SECURITY

OSWP Exam Documentation

v.2.0

matteo.peruzzi96@gmail.com

Matteo Peruzzi

OSID: OS-74948

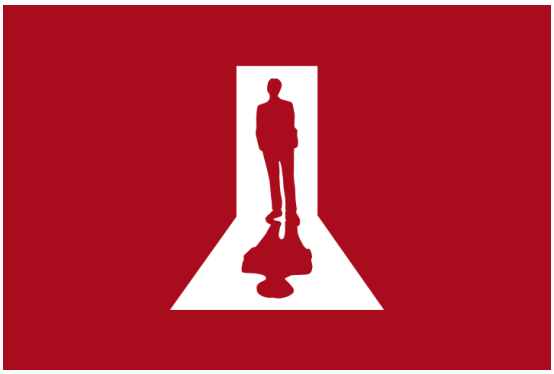

Table of Contents

1.0 Access Point 1: WPA-PSK (Severnaya)

While monitoring the wifi activity in the area, I discovered an Access Point called SWCC. After finding out the SWCC AP was using WPA-PSK, I exploited it intercepting some data containing the handshake – forced by deauthenticating a client. Once intercepted, I used aircrack-ng with a default kali password list to retrieve the key.

As final step, I logged in the wifi using the key found and retrieved *proof.txt*.

1.1 Proof

AP key: 999999999

Proof.txt: 73d8e398a11bb00a628018244321aaff

1.2 Screenshots

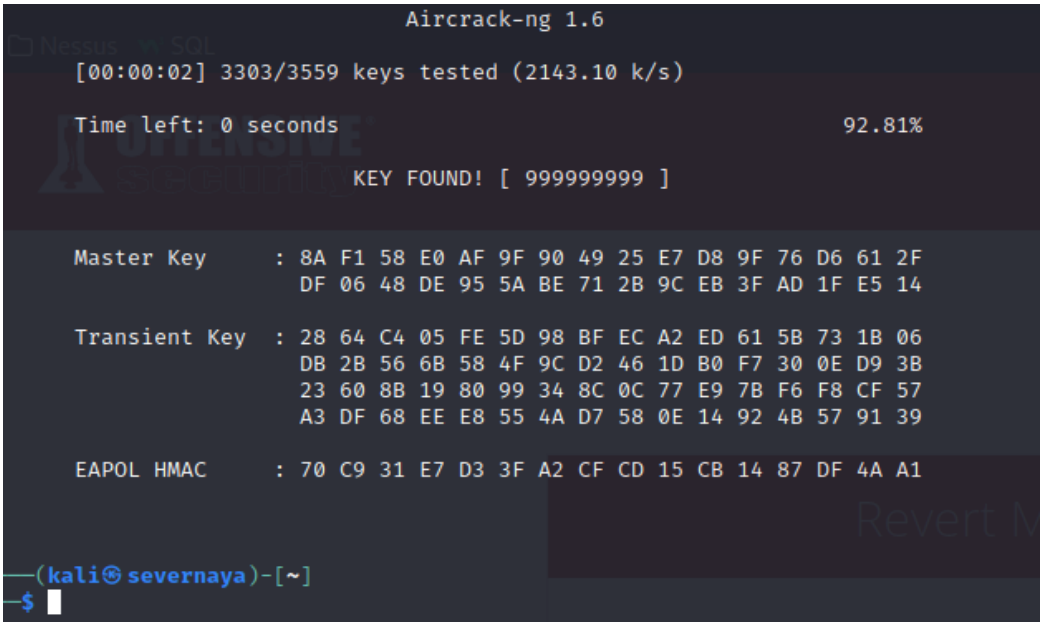

Figure 1: Aircrack-ng finding the wireless key.

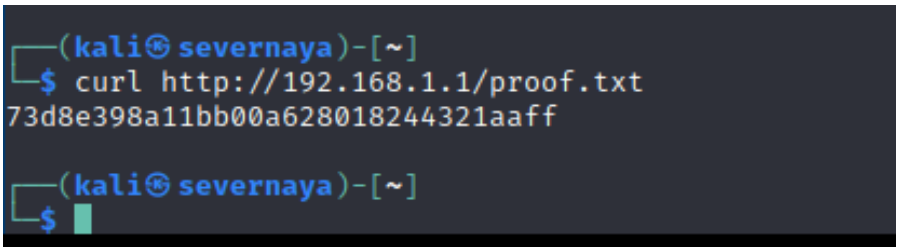

Figure 2: Proof of the flag retrieved after connecting to the AP.

1.2 Steps

After deploying and connecting to the remote machine (*severnaya*) through SSH, I set the *WLAN0* interface to monitor mode using the following command:

 # sudo airmon-ng start wlan0

Figure 3: Set the interface to monitor mode.

This command creates a *wlan0mon* interface in monitor mode, ready to scan the area.
As next step, I started scanning the nearby Wifi networks using airodump-ng like follows:

 # sudo airodump-ng wlan0mon

The output of the above command is shown in the picture below:

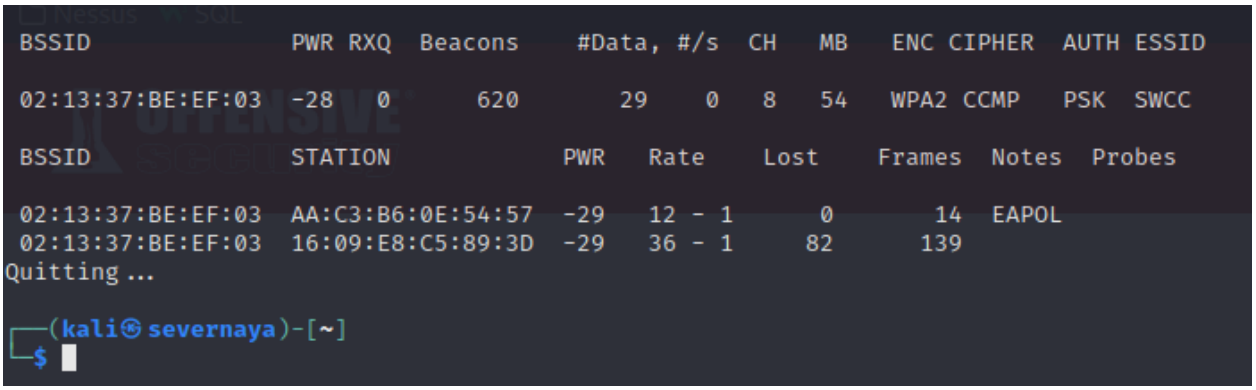

Figure 4: Output of airodump-ng.

From the output I noticed a network called SWCC using WPA-PSK. To get more details and to try and find the password, I performed a deauthentication attack against a client that was connected to the AP while saving the captured data to a file (*wpa-01.cap*) using the code below:

 # aireplay-ng -0 1 -a 02:13:37:BE:EF:03 -c D6:D5:6A:2B:2F:8E wlan0mon

While running the data dump command specifying the network I was interested in:

 # airodump-ng -c 8 -w wpa –essid SWCC –bssid 02:13:37:BE:EF:03 wlan0mon

This way, I have been able to save some useful data (containing the handshake) to then try and crack it locally. The process is shown in the following picture:

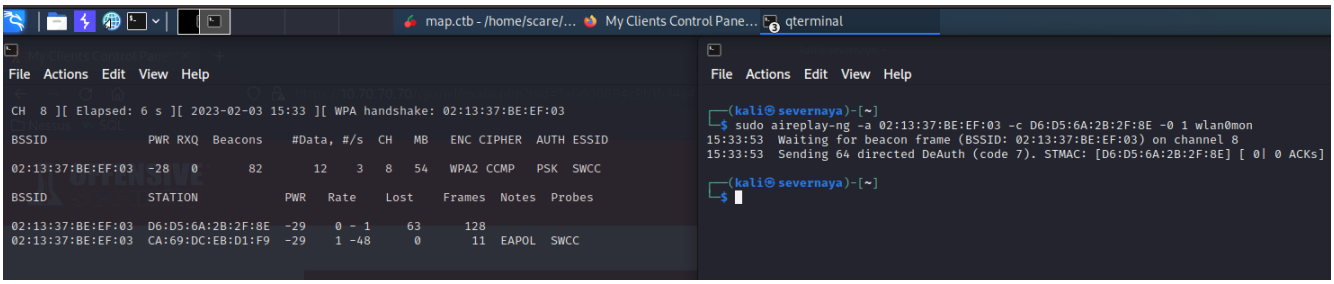

Figure 5: Deauthentication attack against a client to get the handshake in the data dump.

Once captured, I tried to crack the key using aircrack-ng using the command below:

 # sudo aircrack-ng -w /usr/share/john/password.lst -e SWCC -b 02:13:37:BE:EF:03 wpa-01.cap

And the script successfully found the key:

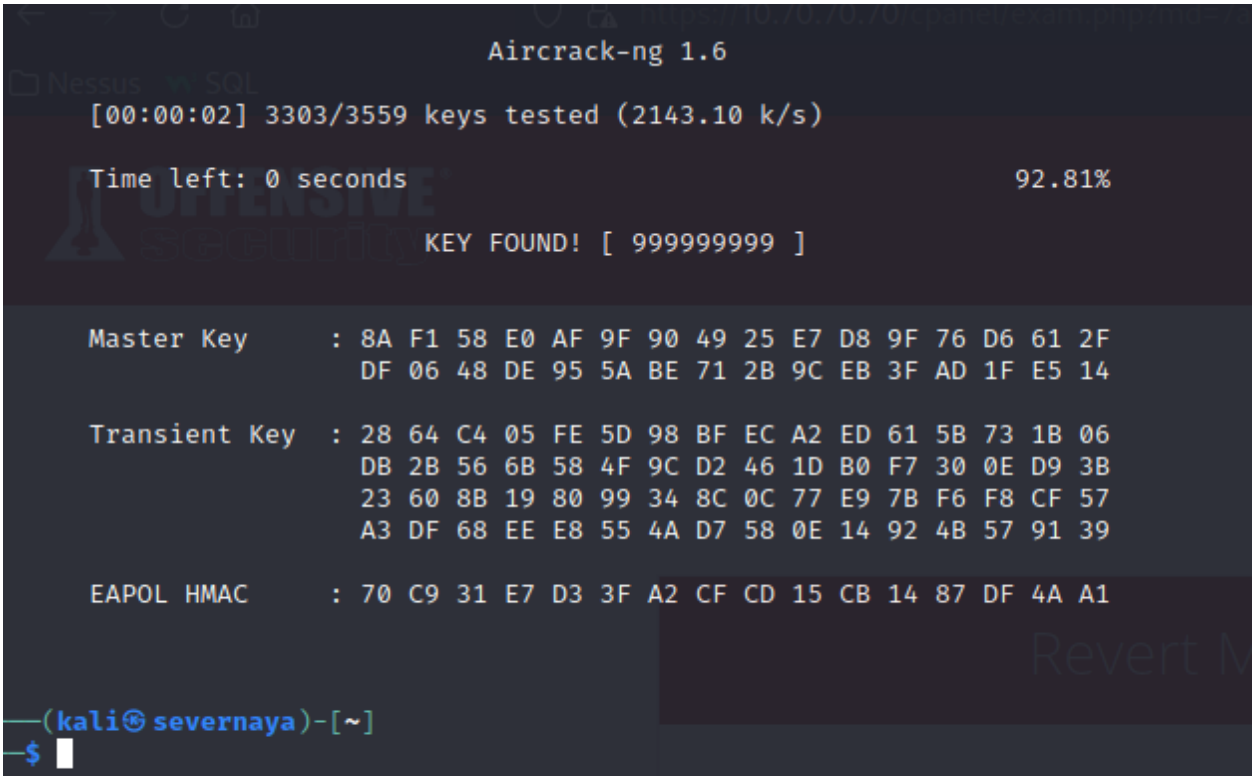

Figure 6: Aircrack-ng finding the key "999999999" .

Once the key was found, I configured a wifi.conf file containing the configuration to use to connect to the network using *wpa_supplicant*:

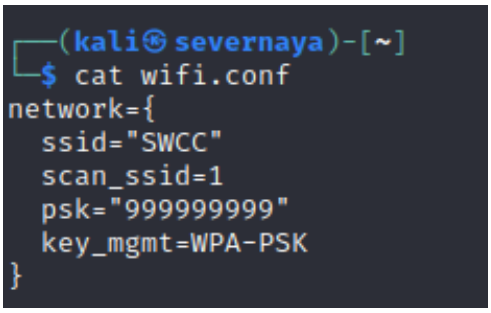

Figure 7: wpa_supplicant configuration used.

After stopping the monitor mode on the interface, running *wpa_supplicant* and *dhclient,* I have successfully connected to the wifi and retrieved the hash:

 # sudo airmon-ng stop wlan0mon

 # sudo wpa_supplicant -c wifi.conf -I wlan0 -B

 # sudo dhclient wlan0

 # curl http://192.168.1.1/proof.txt

Figure 8: Final steps to retrieve the hash.

2.0 Access Point 2: WPA-EAP (bagend)

While monitoring the wifi activity in the area, I discovered an Access Point called Bilbos. After further enumeration I discovered that Access Point was transmitting on channel 10 and using WPA2 with MGT, signs of a probable WPA-Enterprise. I proceeded to collect information by utilizing the deauthentication attack – like I did against Access Point 1 – to then study the packages and certificates dumped using Wireshark. After that, I created a fake AP really similar to the AP found before using *freeradious* and *hostapd-mana*. I waited for someone to connect to my AP, capture the credentials and crack them locally. Once captured, I used *asleep* to retrieve the key and used it to log into the network and dump the *proof.txt*.

2.1 Proof

AP key: marielle

Proof.txt: 552dcc0e40ef6535736636a3f26bf4d3

2.2 Screenshots

Figure 9: asleep finding the password to the network.

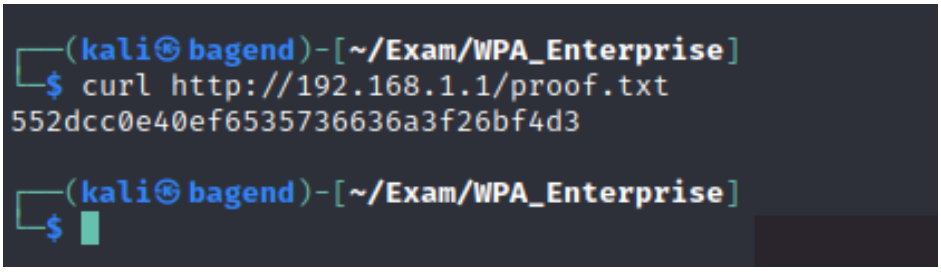

Figure 10: Retrieving the final Proof.txt.

2.3 Steps

NOTE: since the first part of the attack is the same as the one described above, I will quickly run through it to the focus on the different and more important part.

After deploying and connecting to the remote machine (*bagend*) through SSH, I noticed a WPA-Enterprise network called Bilbos. In order to get some details about the network, I run a deauthentication attack against a client connected to the AP while intercepting the data (with the handshake) and saving it to a local file (see attack against the WPA-PSK above to check details).

After the attack, I had some different data files to analyze, as shown in the picture below:

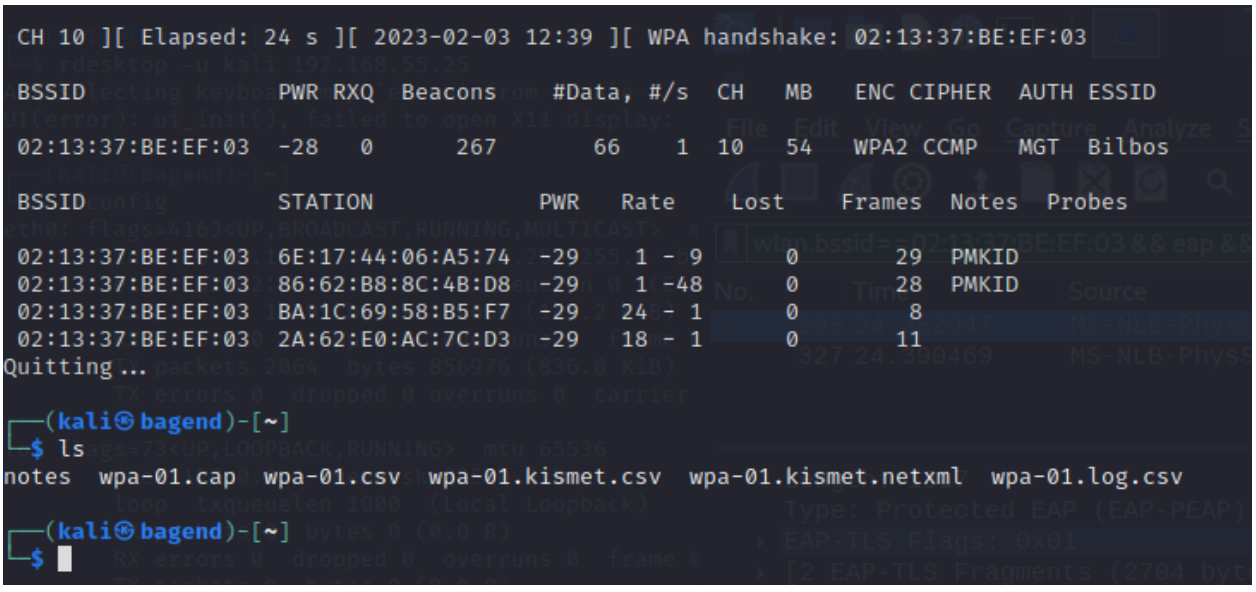

Figure 11: airodump-ng showing Bilbos AP and list of the files dumped.

I then connected to the remote machine (*bagend*) usiong *freerdp* using the command below:

 # freerdp -u kali 192.168.55.25

My aim was to analyze the captured packets using Wireshark, to see more information about the AP:

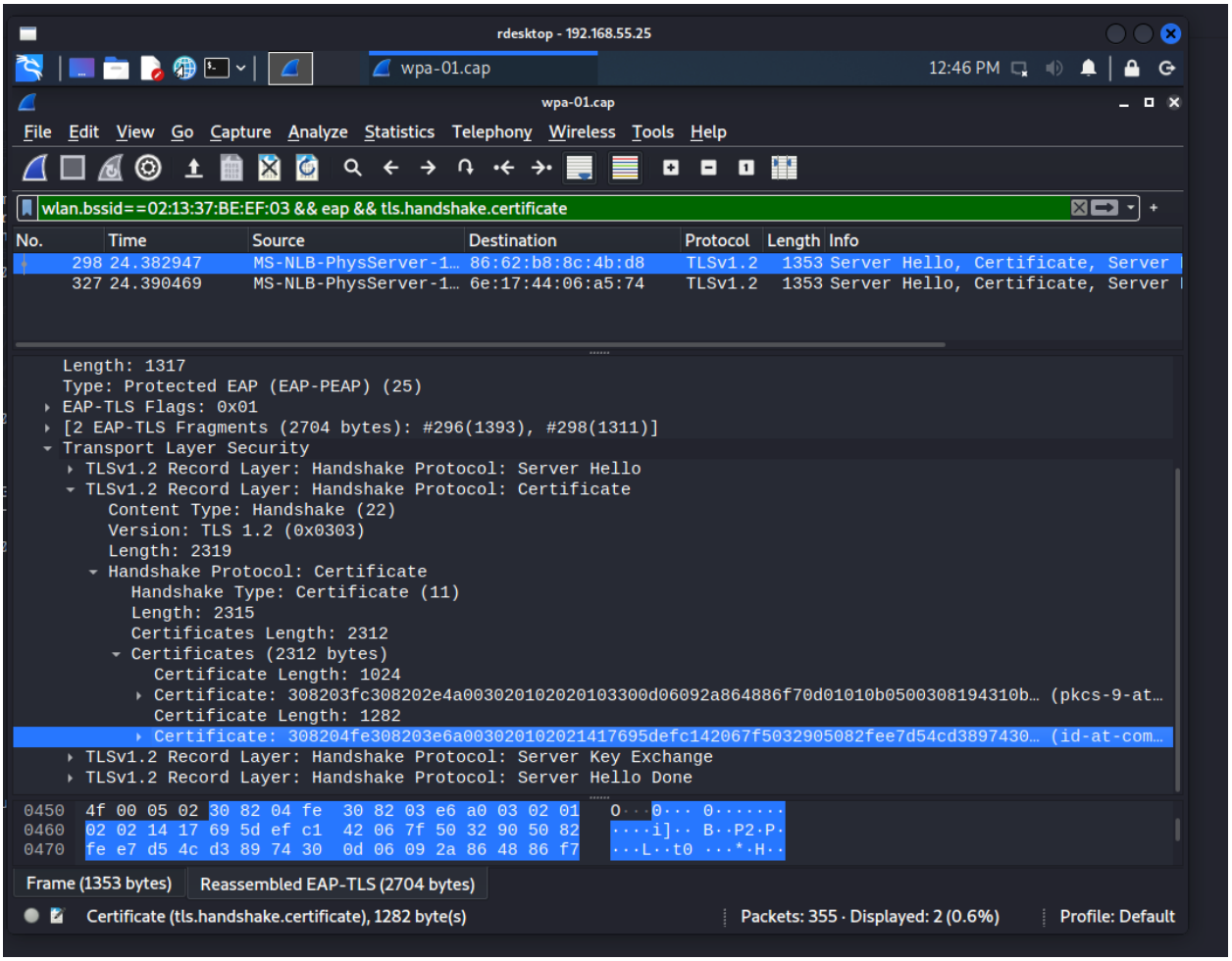

Figure 12: Wireshark analysis on the captured packages.

Apart from the other information like type of authentication method (EAP-PEAP) and encryption protocols (CCMP or/and TKIP), I filtered out information about the certificates using the following filter – as shown in Figure 12:

 # wlan.bssid==02:13:37:BE:EF:03 && eap && tls.handshake.certificate

I then saved the two certificates (in .der extension) right-clicking on each of them and selecting *Export Packet Bytes* to then check their information using the following command:

 # openssl x509 -inform der -in ca.der -text

After that, I started configuring the my AP: *freeradius* was already installed on the system, so I proceeded working on its settings, modifying /etc/freeradius/3.0/certs/ca.cnf (under [certificate_authority]) and /etc/freeradius/3.0/certs/server.cnf (under [server]) like follows:

```
countryName             = US
stateOrProvinceName     = Radius
localityName            = Somewhere
organizationName        = Bilbos Inc.
emailAddress            = admin@bilbos.org
commonName              = "Bilbos Certificate Authority"
```

Figure 13: ca.cnf and server.cnf settings modified.

Once done, I proceeded building the certificates, regenerating the *dh* and running the following commands as root:

 # cd /etc/freeradius/3.0/certs/

 # rm dh

 # make

```
┌──(root💀bagend)-[/etc/freeradius/3.0/certs]
└─# ls
bootstrap  ca.cnf  client.cnf  dh  inner-server.cnf  Makefile  README  server.cnf  xpextensions

┌──(root💀bagend)-[/etc/freeradius/3.0/certs]
└─# rm /etc/freeradius/3.0/certs/dh

┌──(root💀bagend)-[/etc/freeradius/3.0/certs]
└─# make
openssl dhparam -out dh -2 2048
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
.....+.....+.....
.....+.....+.....
.....+.....+.....
.....
.....+.....+.....
.....+.....+.....
.....
.....+.....+.....
```

Figure 14: certificate creation process.

I was able to correctly create the CA certificate, the server certificate and the Diffie-Hellman parameters.

Once done, I started working on the httpd-mana configuration, creating the hostapd-mana.conf file and modifying the *ssid* field to match the AP essid and the certificates fields to match the certificates we just created:

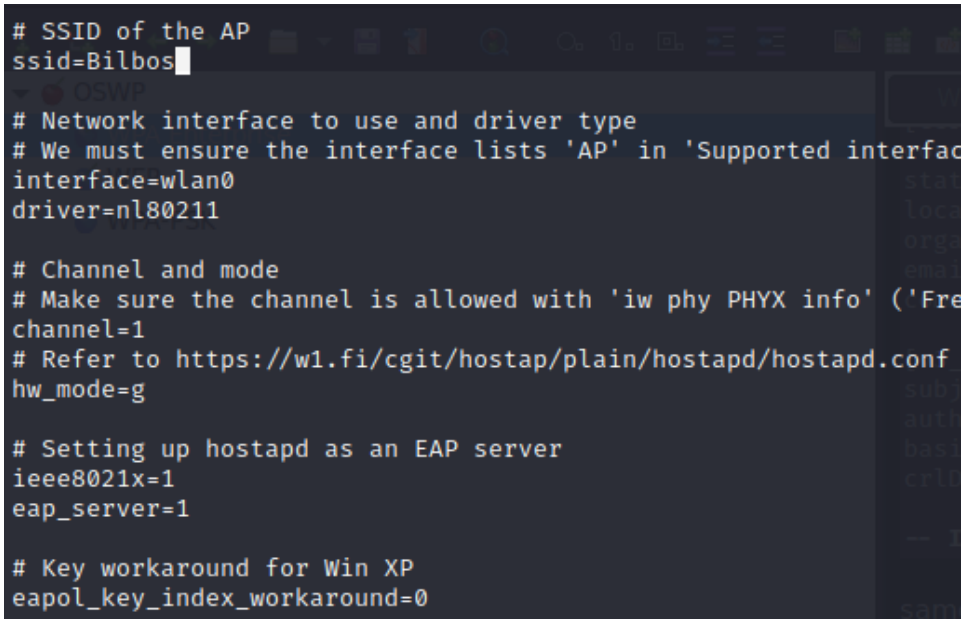

Figure 15: initial part of the httpd-mana.conf file.

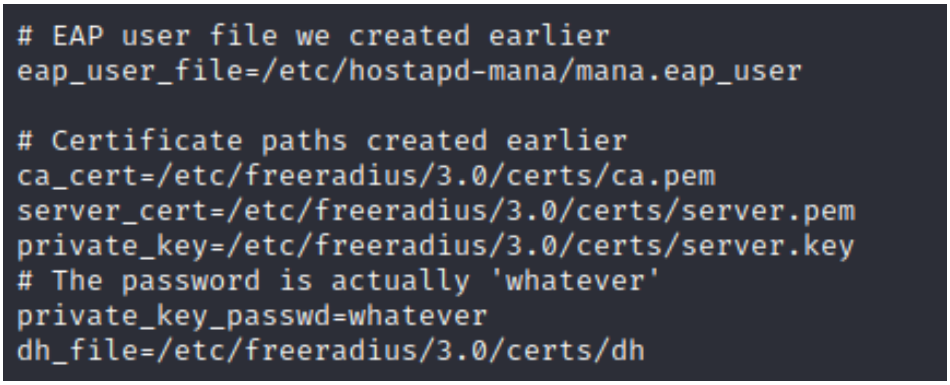

Figure 16: certificates part of hostapd-mana.conf

As one of the final configuration steps, I created the */etc/hostapd-mana/mana.eap_user* file containing the following:

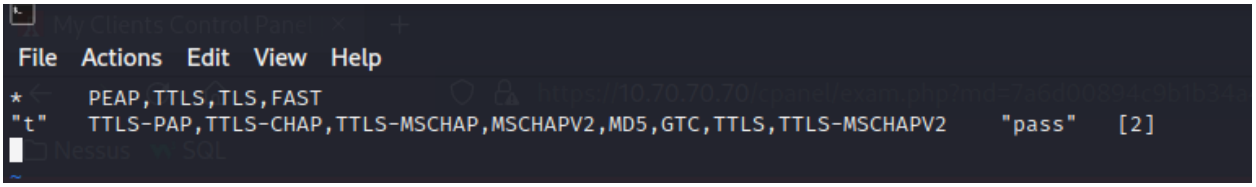

Figure 17: mana.eap_user configuration file.

At this point, I started running *hostapd-mana* with the configuration just created and after a while, I've intercepted a victim attempt to authenticate:

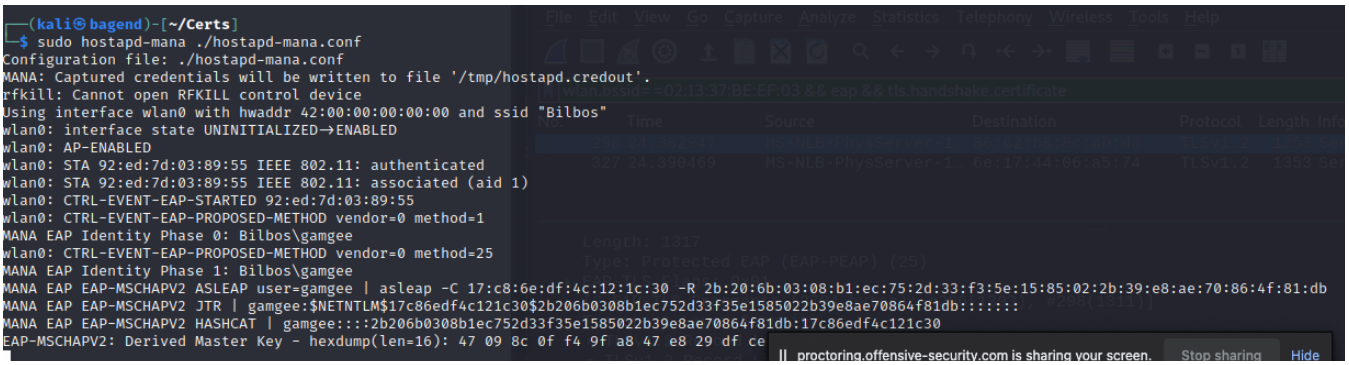

Figure 18: hostapd-mana running and showing an authentication attempt.

From the picture above, we can see a user *gamgee* attempting to connect. Hostapd-mana gives us the command to run to crack the key, so I just copied it and run it specifying my interface and password list:

Figure 19: asleep cracking the network key "marielle".

I finally retrieved the key to access the network. Next step I've took was to create a configuration file to use with *wpa_supplicant* to connect to the network and retrieve the final *proof.txt.* To do so, I used the following configuration in wifi-client.conf:

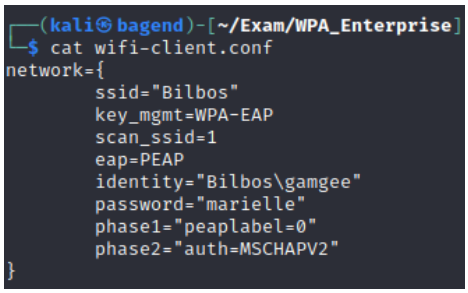

Figure 20: wifi-client.conf configuration file.

I then run *wpa_supplicant* using the configuration just created and, after successfully connecting to the network and used *dhcpclient* to get an IP, I retrieved the *proof.txt*.

I've performed the actions explained above using the following commands:

 # sudo wpa_supplicant -i wlan0 -c wifi-client.conf -B

 # sudo dhclient wlan0

#curl http://192.168.1.1/proof.txt

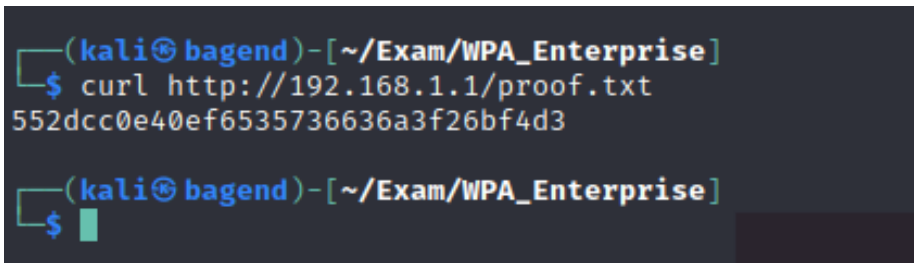

Figure 21: final proof.txt retrieved.