

002 - WPAPS2_Exploit_Walkthrough

So, I acquired the Alfa AWUS036ACM.

There is no set up required. We just need to change some settings in the hypervisor we are using.

For VirtualBox Users

1. **Enable USB 3.0 Support**
 - Go to **VirtualBox → Settings → USB**
 - Enable **USB Controller** and select **USB 3.0 (xHCI Controller)**
2. **Add the ALFA Adapter as a USB Device**
 - Click **Add USB Device Filter (+ Icon)**
 - Select **MediaTek MT7612U (or ALFA AWUS036ACM)**
3. **Start Kali and Check If It's Recognized**

Run inside Kali:

```
bash
lsusb
```

Copy Edit

For VMware Users

1. **Manually Attach the Adapter to Kali**
 - From VMware menu, go to:
Player → Removable Devices → (Your ALFA Adapter Name) → Connect (Disconnect from Host)
2. **Verify It in Kali**

Run inside Kali:

```
bash
lsusb
```

Copy Edit



At first, my kali was not recognizing it. I noticed that in the VM Setting, I had the option to chose between USB 2.0, 3.0, and 3.1. So, I changed to 3.1, and connected the device once again to my kali, and it recognized it.

```

(kali@kali)-[~]
$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 004: ID 0e0f:0008 VMware, Inc. Virtual Bluetooth Adapter
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 002: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 003 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub

(kali@kali)-[~]
$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 004: ID 0e0f:0008 VMware, Inc. Virtual Bluetooth Adapter
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 002: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 003 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 004 Device 002: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 004 Device 003: ID [REDACTED] MediaTek Inc. [REDACTED] 802.11a/b/g/n/ac Wireless Adapter

```

Now, when I try iwconfig, it does not show up in there.

Summary

- ✓ YES, it's normal that the device does not show up on your host system once it's attached to Kali.
- ✓ If Kali detects it (`lsusb` shows MediaTek MT7612U), you've passed it correctly.
- ✓ If it's not appearing in `iwconfig`, check drivers and restart network services.

The following command is to check what is going on after we issue "#sudo modprobe mt76x2u"

```

bash
dmesg | tail -50

```

Try resetting the USB controller to refresh communication:

```

bash
echo -n "0000:00:0c.0" | sudo tee /sys/bus/pci/drivers/xhci_hcd/unbind
sleep 2
echo -n "0000:00:0c.0" | sudo tee /sys/bus/pci/drivers/xhci_hcd/bind

```

Replace `0000:00:0c.0` with the correct PCI address for your system (get it from `lspci | grep -i usb`).

```
#lspci | grep -i usb
```

```
"echo -n "0000:00:0c.0" | sudo tee /sys/bus/pci/drivers/xhci_hcd/unbind sleep 2 echo -n "0000:00:0c.0" |
sudo tee /sys/bus/pci/drivers/xhci_hcd/bind"
```

The following command allow us to see the input of the wireless driver as it comes in, so lets try first running the command and then plugging in the driver.

```
douglas@Fr3shF0r3nsics -> sudo dmesg -w
```

[3.144809] piix4_smbus 0000:00:07.3: SMBus Host Controller not enabled!

The error message you're seeing suggests that the SMBus (System Management Bus) controller is not enabled on your system. The `piix4_smbus` driver is trying to access the SMBus, but it appears that the controller is not active or available.

To resolve this, you could try the following steps:

1. **Check BIOS/UEFI settings:** Sometimes, the SMBus controller is disabled in the BIOS/UEFI settings. Boot into your BIOS/UEFI and look for any settings related to SMBus or I2C controllers. Ensure that they are enabled.
2. **Load the SMBus driver:** Try manually loading the SMBus driver to see if it resolves the issue. You can do this by running:

```
sudo modprobe piix4_smbus
```

After running this command, check the `dmesg` logs again to see if the issue persists.

3. **Check for hardware compatibility:** Ensure that your hardware actually supports SMBus. If your system doesn't have an SMBus controller, this message might be benign.
4. **Update your kernel or drivers:** Make sure you're using the latest version of your kernel and drivers, as updates might fix compatibility issues.
5. **Disable SMBus in your system:** If SMBus is not essential for your system, you can try disabling it in the kernel module configuration. However, this might not be recommended unless you're sure you don't need it for hardware monitoring or other functions.

```
kali@kali: ~ × kali@kali: ~ ×
GNU nano 8.1 /etc/default/grub
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
# info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0 linuxprvh ADBreach OpacityRoom
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet mitigations=off"
GRUB_CMDLINE_LINUX=""

# If your computer has multiple operating systems installed, then you
# probably want to run os-prober. However, if your computer is a host
# for guest Oses installed via LVM or raw disk devices, running
# os-prober can cause damage to those guest Oses as it mounts
# filesystems to look for things.
#GRUB_DISABLE_OS_PROBER=false

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal
#GRUB_TERMINAL=console

# The resolution used on graphical terminal (in units of lines and columns)
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command 'vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY="true"

# Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"

h3aPayload passwords.txt AgentSudo Anthem
```

→ NO
CHANGES

Steps to Resolve/Workaround:

1. **Update BIOS/UEFI:** This type of mismatch is often caused by outdated firmware. Visit your motherboard's manufacturer's website and check for any BIOS/UEFI updates. Updating to the latest version could resolve this issue.
2. **Check for CPU Configuration Options in BIOS:** Sometimes, there may be settings related to CPU topology or APIC in the BIOS/UEFI. Ensuring these are set to default or optimized settings can help resolve the mismatch.
3. **Kernel Boot Parameters:** You can try adjusting certain boot parameters to work around this issue. For instance, you can try adding `noapic` or `nolapic` to your kernel boot parameters to bypass the APIC configuration. This can sometimes resolve conflicts, but it may impact performance or features that rely on APIC.

To do this:

- Edit the GRUB configuration file:

```
bash                                                                    Copy Edit

sudo nano /etc/default/grub
```

- Find the line starting with `GRUB_CMDLINE_LINUX_DEFAULT` and add `noapic` or `nolapic` at the end of the line:

```
bash                                                                    Copy Edit

GRUB_CMDLINE_LINUX_DEFAULT="quiet splash noapic"
```

- Update GRUB and reboot:

```
bash                                                                    Copy Edit

sudo update-grub
sudo reboot
```

4. **Ignore the Warning:** If your system seems to be running fine despite the messages, you may choose to ignore them. In many cases, the kernel can still operate normally even with this warning.

Changing the above file broke the boot process of my virtual kali machine. Luckily, I had screenshotted the default file and was able to change it back.

If yours is broke too (vmware), while the machine is booting your are going to be prompted to boot normally or in advanced mode, select advanced mode, then you want to boot in recovery mode. For me, it came a time where everything seemed to be frozen, I just clicked enter, and was able to boot into my machine. Once you are completely booted and logged in, you wanna go to the file you edited, and change back to the default or at least to the way it was before. After I "rolled back", I rebooted my machine, and it was fine.

Lets see if this works.

```
amd64-microcode-blacklist.conf ath9k_htc.conf intel-microcode-blacklist.conf kali-defaults.conf
(kali@kali)-[/etc/modprobe.d]
$ sudo apt upgrade
The following packages were automatically installed and are no longer required:
imagemagick-6.q16 libbftm9 libhdf5-hl-100t64 libtag1v5
libbftm1 libgl-mesa-dev libjxl0.9 libtag1v5-vanilla
libc++1-19 libgles-dev libmagickcore-6.q16-7-extra libtagc0
libc++abi1-19 libgles1 libmagickcore-6.q16-7t64 libunwind-19
libcapstone4 libglvnd-core-dev libmagickwand-6.q16-7t64 libwebRTC-audio-processing1
libconfig++9v5 libglvnd-dev libmbcrypto7t64 libx265-209
libconfig9 libgtksourcview-3.0-1 libpaper1 openjdk-23-jre
libdirectfb-1.7-7t64 libgtksourcview-3.0-common libqt5x11extras5 openjdk-23-jre-headless
libegl-dev libgtksourcviewmm-3.0-0v5 libsuperlu6 python3-appdirs
Use 'sudo apt autoremove' to remove them.

Summary:
Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0

(kali@kali)-[/etc/modprobe.d]
$ sudo apt update
Hit:1 http://http.kali.org/kali kali-rolling InRelease
All packages are up to date.

(kali@kali)-[/etc/modprobe.d]
$ echo 'options usbcore autosuspend=-1' | sudo tee /etc/modprobe.d/usbcore.conf
options usbcore autosuspend=-1

(kali@kali)-[/etc/modprobe.d]
$ sudo update-initramfs -u
update-initramfs: Generating /boot/initrd.img-6.11.2-amd64
```

Here, if it is a power issue, then we can do try the above. I am running out of ideas, so I am trying to see if something works.

It did not work for me, so I deleted the file.

Commented out the blacklist microcode on both files, and rebooted my system.

```
(kali@kali)-[~]
$ cd /etc/modprobe.d

(kali@kali)-[/etc/modprobe.d]
$ ls
amd64-microcode-blacklist.conf ath9k_htc.conf intel-microcode-blacklist.conf kali-defaults.conf

(kali@kali)-[/etc/modprobe.d]
$ cat amd64-microcode-blacklist.conf
# The microcode module attempts to apply a microcode update when
# it autoloading. This is not always safe, so we block it by default.
#blacklist microcode

(kali@kali)-[/etc/modprobe.d]
$ cat intel-microcode-blacklist.conf
# The microcode module attempts to apply a microcode update when
# it autoloading. This is not always safe, so we block it by default.
#blacklist microcode

(kali@kali)-[/etc/modprobe.d]
$ ls
amd64-microcode-blacklist.conf ath9k_htc.conf intel-microcode-blacklist.conf kali-defaults.conf

(kali@kali)-[/etc/modprobe.d]
$ sudo update-initramfs -u
update-initramfs: Generating /boot/initrd.img-6.11.2-amd64

(kali@kali)-[/etc/modprobe.d]
$
```

It looks like this might have done the trick. I do not want to jinx it.

```

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

eth0        no wireless extensions.

wlan0       IEEE 802.11  ESSID:off/any
            Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
            Retry short limit:7   RTS thr:off   Fragment thr:off
            Power Management:off

```

Alright, after a long troubleshooting, it comes the reward.

```

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

eth0        no wireless extensions.

wlan0       IEEE 802.11  ESSID:off/any
            Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
            Retry short limit:7   RTS thr:off   Fragment thr:off
            Power Management:off

(kali㉿kali)-[~]
$ airmon-ng check kill
Run it as root

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

Killing these processes:

    PID Name
    1688 wpa_supplicant

(kali㉿kali)-[~]
$ sudo su
(root㉿kali)-[/home/kali]
# 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)

```

We can see it says monitor mode enabled, and station mode disabled. This is what we want.

We can confirm these changes by running once again the iwconfig.

```

(root㉿kali)-[/home/kali]
# iwconfig
lo          no wireless extensions.

eth0        no wireless extensions.

wlan0mon    IEEE 802.11  Mode:Monitor  Frequency:2.457 GHz  Tx-Power=20 dBm
            Retry short limit:7   RTS thr:off   Fragment thr:off
            Power Management:on

```


We will be attacking WPAPS2. This attack relies on this poor password, so we can crack it and get in.

The tool we are going to use to make the attack is called #airodump-ng, and it is in the same suite as the tool we used to set up the monitor mode.

First, we need to see what is available, so we are going to issue:

```
#airodump-ng wlan0mon
```

And, we start collecting info on the available wifis.

CH 3][Elapsed: 6 s][2025-02-08 16:16

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
[REDACTED]	-43	1	3 0	3	65	WPA2	CCMP	PSK	[REDACTED]
[REDACTED]	-33	3	0 0	3	360	WPA2	CCMP	PSK	[REDACTED]
[REDACTED]	-34	2	0 0	3	360	WPA2	CCMP	PSK	[REDACTED]
[REDACTED]	-43	3	67 0	3	65	WPA2	CCMP	PSK	[REDACTED]
[REDACTED]	-83	0	0 0	5	360	WPA2	CCMP	PSK	TP-Link_
[REDACTED]	-79	2	0 0	8	130	WPA2	CCMP	PSK	[REDACTED]
[REDACTED]	-65	3	0 0	8	270	WPA2	CCMP	PSK	[REDACTED]
[REDACTED]	-63	2	0 0	8	195	WPA2	CCMP	PSK	[REDACTED]
[REDACTED]	-77	2	0 0	7	260	WPA2	CCMP	PSK	[REDACTED]
[REDACTED]	-49	3	0 0	6	130	WPA2	CCMP	PSK	[REDACTED]
[REDACTED]	-70	1	1 0	2	130	WPA2	CCMP	PSK	[REDACTED]
[REDACTED]	-83	1	3 0	1	720	WPA2	CCMP	PSK	[REDACTED]
[REDACTED]	-73	0	0 0	1	360	WPA3	CCMP	SAE	[REDACTED]

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
[REDACTED]:D1:B8:[REDACTED]:B8:[REDACTED]	[REDACTED]:B8:[REDACTED]	-1	12e- 0	0	3		
[REDACTED]:D1:B8:[REDACTED]:B8:[REDACTED]	[REDACTED]:B8:[REDACTED]	-52	12e-12e	3	67		
[REDACTED]:DA:FF:[REDACTED]:DE:[REDACTED]	[REDACTED]:DE:[REDACTED]	-1	24e- 0	0	1		
[REDACTED]:E2:C1:[REDACTED]:5B:[REDACTED]	[REDACTED]:5B:[REDACTED]	-1	1e- 0	0	3		

I am going to be attacking my router, tp-ling.

This command, as you can see, allows us to see the BSSID, the ESSID, and the much more.

Access points BSSID is the MAC Address of the access point. PWR is the Power level, which tells us how close we are to the network. CH is the channel the network is using (this is what we are going to use to target the network), it shows how much data is going through the network, the type of connection, the CIPHER, the Authentication mechanism, and the ESSID.

The network we want to attack is on Channel 5.

We are going to issue:

```
#airodump-ng -c 5 --bssid MAC_Address -w capture wlan0mon
```

-w is going to write the output to a file named "capture".

CH 5][Elapsed: 48 s][2025-02-08 16:32

BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
██████████	-36	24	372	13 0	5	360	WPA2	CCMP	PSK	TP-Link_██████████
BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes			

This is how it looks like after running the command.

We are going to run an attack called de-authentication attack. This is to speed this process of capturing this hash so we can crack a password. This is basically kicking some device out of the network, and when the client reassociates, we are able to capture the handshake packet to crack the hash. For this attack, we need the mac address for a Station which for some reason we do not have captured one yet.

```
#aireplay-ng --deauth 10 -a <AP_MAC> wlan0mon
```

Lets try the above attack. This will attempt to deauth all devices, and see if we can get the handshake.

3. Check for Hidden Clients:

- Some clients may not appear in the `airodump-ng` list due to power-saving modes or low traffic.
- If you suspect this is the case, you can try sending a broadcast deauthentication packet to force all connected clients to reconnect:

```
bash
```

Copy

```
aireplay-ng --deauth 10 -a <AP_MAC> wlan0mon
```

- This command does not require a `Client_MAC` and will deauthenticate all clients connected to the AP.

Something happened, and the terminal in my kali linux just stopped responding. Not sure what happened there. We could have tested for the vulnerability before attacking it. Maybe 10 was too much.

4. Verify Your Wireless Adapter's Capabilities:

- Ensure your wireless adapter supports packet injection, which is required for deauthentication attacks.
- Test your adapter's injection capability using:

```
bash
```

Copy

```
aireplay-ng --test wlan0mon
```

- If the test fails, your adapter may not support injection, and you may need to use a different one.

Okay. Well, even though we haven't yet actually made this work, we learned a lot.

So, if we had the Client_MAC, we would be able to use aireplay-ng to do a deauth attack to a specific machine.

We would issue the following commands.

Deauth attack:

```
root@kali: ~  
File Edit View Search Terminal Help  
C7:BF:8A:00:73 root@kali:~# aireplay-ng -0 1 -a 50:C7:BF:8A:00:73 -c 3C:F0:11:22:DB:E3 wlan0mon
```

Handwritten annotations:
- **AP_MAC** (above -a)
- **Client MAC** (above -c)
- **Number of times Attack runs** (below -0 1)

```
root@kali: ~  
File Edit View Search Terminal Help  
CH 2 ][ Elapsed: 24 s ][ 2019-12-22 00:33 ][ WPA handshake: 50:  
BSSID PWR RXQ Beacons #Data, #/s CH MB ENC  
50:C7:BF:8A:00:73 -10 76 195 87 38 2 195 WPA2  
BSSID STATION PWR Rate Lost Fram  
50:C7:BF:8A:00:73 3C:F0:11:22:DB:E3 -32 1e-48e 32 19  
0
```

Handwritten annotation: **WPA** (above WPA handshake)

Then, we can Ctrl+C, and crack the hash.

So, the command we can run to crack the file with the handshake we captured:

```
"#aircrack-ng -w wordlist.txt -b AP_MAC capture.cap
```

Now, we have a couple options from here. We can let this run indefinitely, and hope we capture some traffic with Client_MAC, or try to perform other attacks or perhaps the same steps with other tools.

Heath says to be patient, not sure how patient he is talking about.

7. Use a Different Tool:

- If `airodump-ng` is not showing clients, you can try using other tools like `Kismet` or `Wireshark` to capture Wi-Fi traffic and identify connected clients.

3. Handshake Capture with `hcxumptool`

- `hcxumptool` is another great tool for capturing handshakes, specifically designed to capture WPA handshakes more effectively. It can also collect PMKID hashes, which are another type of key that can be used in password cracking attempts.
- Example to start capturing handshakes:

CSS

Copy Edit

```
sudo hcxumptool -i wlan0mon --disable_platform
```

- You can use `hcxpcapngtool` (another tool from the same suite) to convert the captured data into a format that can be cracked with tools like Hashcat.

5. Reaver (if using WPS)

- Reaver is useful if the target router has WPS (Wi-Fi Protected Setup) enabled. It can exploit weaknesses in WPS and recover the WPA password, potentially without needing a handshake.
- Example of usage:

CSS

Copy Edit



```
sudo reaver -i wlan0mon -b <BSSID> -c <channel> -vv
```

6. Kismet

- Kismet is a wireless network detector, sniffer, and intrusion detection system. It's more advanced than airodump-ng in some ways, and you can use it to capture handshakes or network traffic in a more detailed manner. It's useful when you want to capture and analyze network traffic in addition to just cracking WPA.

To use Kismet:

```
nginx
```

 Copy  Edit



```
sudo kismet
```

Each tool has its strengths, so you may need to experiment with a combination of them depending on the situation. Wifite and hcxdumpool are especially good if you want something more streamlined and automated.

2. Wifite

- Wifite is a more automated tool for capturing handshakes and attacking WPA/WPA2 networks. It can capture handshakes from multiple access points simultaneously, and it also has the option to perform deauthentication attacks automatically.
- Example of usage:

```
nginx
```

 Copy  Edit

```
sudo wifite
```

Wifite will scan for networks and attempt to capture the handshakes from the networks in range.

Here are some other ways we can do this:




1. Evil Twin Attack (Rogue Access Point)

- **What it is:** You create a fake access point (AP) with the same SSID as the target network. Clients may connect to your rogue AP, allowing you to capture their traffic or perform further attacks.
 - **How it works:**
 - Use tools like `airbase-ng` or `hostapd` to create the rogue AP.
 - Deauthenticate clients from the legitimate AP (if possible) to force them to reconnect to your rogue AP.
 - **Tools:**
 - `airbase-ng` (part of the Aircrack-ng suite)
 - `hostapd`
 - `Karma` (for automated Evil Twin attacks)
-

2. KRACK Attack (Key Reinstallation Attack)

- **What it is:** Exploits a vulnerability in the WPA2 protocol to intercept and decrypt traffic between a client and an AP.
 - **How it works:**
 - Forces the reinstallation of an already-in-use encryption key, allowing the attacker to decrypt packets.
 - **Requirements:**
 - The target network must use WPA2 (WPA3 is not vulnerable).
 - You need to be within range of both the AP and the client.
 - **Tools:**
 - `krackattack` (proof-of-concept scripts available on GitHub)
-

3. Fragmentation Attack

- **What it is:** Exploits the way some devices handle fragmented packets to recover parts of the WPA/WPA2 handshake.
 - **How it works:**
 - Sends specially crafted fragmented packets to the AP or client to capture parts of the handshake.
 - **Tools:**
 - `aireplay-ng` (with fragmentation options)
 - Custom scripts for packet crafting.
- 

4. PMKID Attack

- **What it is:** Exploits a feature in WPA2/WPA3 to capture the PMKID (Pairwise Master Key Identifier) from the AP without requiring client interaction.
 - **How it works:**
 - Captures the PMKID from the AP's first message in the 4-way handshake.
 - Uses offline brute force or dictionary attacks to crack the password.
 - **Tools:**
 - `hcxdumpool` (to capture the PMKID)
 - `hashcat` (to crack the PMKID)
-

5. Downgrade Attack (WPA3 to WPA2)

- **What it is:** Forces a WPA3 network to fall back to WPA2, which may have weaker security.
 - **How it works:**
 - Exploits misconfigurations or vulnerabilities in the AP or client to force a downgrade.
 - Once downgraded, you can use WPA2-specific attacks (e.g., KRACK or deauthentication).
 - **Tools:**
 - Custom scripts or tools like `wpa_supplicant` for testing downgrade scenarios.
-

6. Beacon Flooding

- **What it is:** Overwhelms the target network with fake beacon frames, causing disruption or forcing clients to disconnect.
 - **How it works:**
 - Sends a large number of beacon frames with random SSIDs to confuse clients or the AP.
 - **Tools:**
 - `mdk4` (with beacon flooding mode)
 - `aireplay-ng`
-

7. ARP Spoofing (Man-in-the-Middle Attack)

- **What it is:** Intercepts traffic between a client and the AP by spoofing ARP messages.
 - **How it works:**
 - Tricks the client into sending traffic to your machine instead of the AP.
 - Requires being on the same network segment as the target.
 - **Tools:**
 - `ettercap`
 - `arpspoof` (part of the dsniff suite)
-

8. WPS PIN Brute-force Attack

- **What it is:** Exploits the Wi-Fi Protected Setup (WPS) feature to brute-force the PIN and gain access to the network.
 - **How it works:**
 - Uses tools to guess the WPS PIN, which is often weak or predictable.
 - **Requirements:**
 - WPS must be enabled on the target AP.
 - **Tools:**
 - `reaver`
 - `bully`
-

9. EAP-MD5 or LEAP Exploitation

- **What it is:** Exploits weaknesses in older EAP (Extensible Authentication Protocol) methods like EAP-MD5 or LEAP.
 - **How it works:**
 - Captures authentication frames and cracks the hashes offline.
 - **Tools:**
 - `asleap` (for LEAP attacks)
 - `eapmd5pass` (for EAP-MD5 attacks)
-

10. Wi-Fi Jamming (Denial of Service)

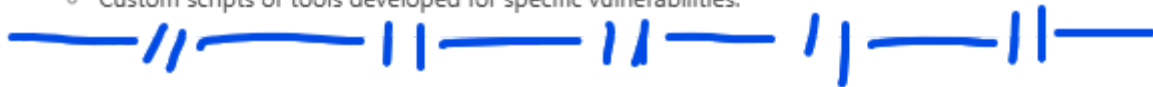
- **What it is:** Disrupts the target network by jamming the Wi-Fi signal.
 - **How it works:**
 - Sends continuous noise or deauthentication packets to prevent legitimate communication.
 - **Tools:**
 - `mdk4` (with deauthentication or jamming modes)
 - Custom RF jamming hardware (note: illegal in most jurisdictions).
-

11. Captive Portal Exploitation

- **What it is:** Exploits vulnerabilities in captive portals (e.g., in public Wi-Fi networks) to bypass authentication or capture credentials.
 - **How it works:**
 - Intercepts or manipulates HTTP traffic to bypass the portal or steal credentials.
 - **Tools:**
 - `Wireshark` (for traffic analysis)
 - `Burp Suite` (for HTTP manipulation)
-

12. Zero-Day Exploits

- **What it is:** Exploits unknown vulnerabilities in Wi-Fi hardware, firmware, or protocols.
- **How it works:**
 - Requires advanced knowledge of Wi-Fi protocols and reverse engineering.
 - Often involves discovering and exploiting flaws in specific devices or software.
- **Tools:**
 - Custom scripts or tools developed for specific vulnerabilities.



After this research, The next tool we are going to be learning will be either Kismet or Wifite.

<https://www.winmill.com/wi-fi-penetration-testing/>

Go check it out their website.

Comprehensive Stages of Wi-Fi Penetration Testing

1. Reconnaissance – The Art of Information Gathering



Figure 2: Surveying Wireless Networks

Tool Highlight: War-driving with Kismet

Capabilities: Kismet excels in passively collecting packets and detecting hidden networks and clients. Its real power lies in its ability to map the network environment without alerting the target.

Practical Usage: Deploy Kismet to survey the target area. Identify SSIDs, MAC addresses of access points, and types of encryption in use. Pay special attention to any unusual network behaviors or unknown SSIDs that might indicate rogue access points.

War-driving: War-driving is a methodical approach for assessing the security of wireless networks. It involves physically moving through an area – typically in a vehicle, hence the term “driving” – to detect and evaluate the security of wireless networks in that vicinity. Kismet excels in this role (Figure-3).



Figure 3: Kismet – Basic Setup for Wardriving with Kismet

2. Scanning and Enumeration – Unveiling the Network



Figure 4: Enumeration

Kismet for Deeper Insights: Beyond initial reconnaissance, use Kismet to uncover the network structure, identify connected devices, and even pinpoint physical locations of devices using signal strength (Figure-5).

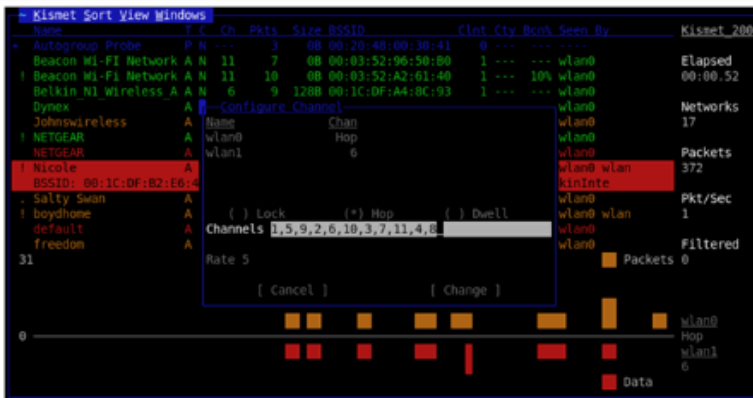


Figure 5: Kismet Detailed Screen

Analyzing Data: This phase is about interpreting the data gathered by Kismet. Look for outdated firmware, weak encryption methods, or excessive data transmission from a particular device.

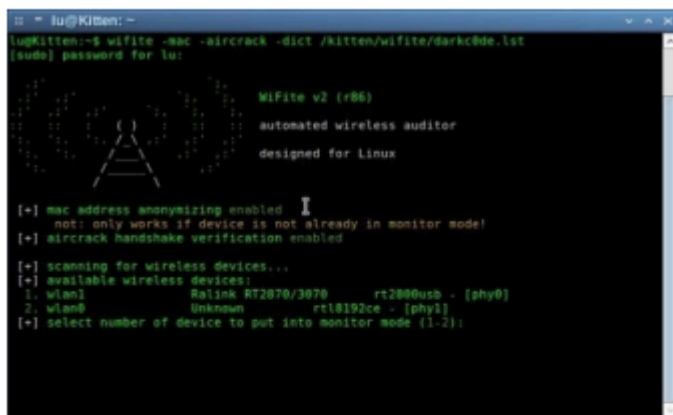
3. Gaining Access – Cracking the Code



Figure 6: Cracking the Code

Tool Focus: Wifite

Capabilities: Wi-Fite is an automated tool that simplifies the process of cracking WEP, WPA/WPA2, and WPS encrypted networks (Figure-7).



```
lu@Kitten:~$ wifite -mac -aircrack -dict /kitten/wifite/darkc0de.lst
[sudo] password for lu:

WiFi v2 (r86)
automated wireless auditor
designed for Linux

[+] mac address anonymizing enabled
not: only works if device is not already in monitor mode!
[+] aircrack handshake verification enabled

[+] scanning for wireless devices...
[+] available wireless devices:
1. wlan1 Ralink RT2870/3070 rt2800usb - [phy0]
2. wlan0 Unknown rtl8192ce - [phy1]
[+] select number of device to put into monitor mode (1-2):
```

Figure 7: Wifite Automated Wi-Fi Assessment Tool

Execution Strategy: For WPA/WPA2, Wi-Fite can use various methods like evil twin attacks, dictionary attacks, and brute force attacks. Remember, success largely depends on the strength of your word lists and computational power.

4. Maintaining Access and Eavesdropping – The Silent Observer



Figure 8: Sniffing the Network

Beyond Access: Once inside the network, it's crucial to maintain a low profile. Use tools like Wireshark in tandem with Kismet to monitor network traffic and sniff out sensitive information (Figure-9).

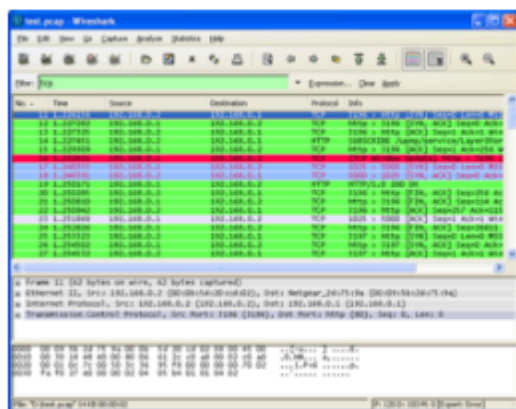


Figure 9: Sniffing with Wireshark

Objective: Look for unencrypted transmissions of sensitive data, such as passwords or personal information, that can be leveraged for further penetration or reported for strengthening network security.

5. Analysis and Reporting – The Finale



Figure 10: Analysis and Reporting Phase

Critical Analysis: This stage involves a thorough analysis of the data collected. Identify patterns, potential security flaws, and areas for improvement.

Reporting: Present your findings in a detailed report, highlighting vulnerabilities, the methods used to exploit them, and recommended actions for remediation.



6. Best Practices and Ethical Considerations



Figure 11: Ethical Hacking and the Law

- **Legal Compliance:** Ensure you have explicit permission and legal clearance before commencing any penetration test.
- **Stay Informed:** Keep abreast of the latest developments in Wi-Fi security to refine your testing methods.
- **Responsible Reporting:** Your report should not just outline vulnerabilities but also provide comprehensive solutions for mitigating these risks.

7. A Deep Dive into Real-World Application

Let's consider a hypothetical scenario. You're tasked with testing the Wi-Fi network of a large corporation. You start with Kismet, identifying an array of devices and noticing a few unexpected SSIDs, potentially indicating rogue access points.

Moving to Wi-Fite, you target the WPA2-encrypted employee network. Using a customized dictionary attack, you manage to crack the password, gaining access to the network. Inside, you employ a combination of Kismet and Wireshark to monitor traffic, discovering several unencrypted email transmissions containing confidential information. In your detailed report, you highlight these findings, stressing the need for stronger encryption, better password policies, and increased awareness about secure transmission of sensitive data.

8. Wrapping Up

In conclusion, Wi-Fi penetration testing is a sophisticated and essential aspect of network security. Tools like Kismet and Wi-Fite are invaluable in this process, but they require skilled handling and ethical application. Remember, the ultimate goal is to strengthen the network's defenses, not just to expose its weaknesses. Stay ethical, stay dedicated, and continue pushing the boundaries of what's possible in cybersecurity.