Wireless Network Exploitation

Capturing, Cracking, and Analyzing WPA2 Traffic

Kayvon Karimi

Cyber Proud Cybersecurity Pre-Apprenticeship Program

Instructor: Gabriel Longboy

4/26/2025

Supplemental Materials:

Video Walk Through

Power Point Slides

Table of Contents

Executive Summary

Wireless security is a vital component of modern cybersecurity, especially as the number of connected devices in home networks continues to grow. These networks, while convenient, often lack the visibility and safeguards found in enterprise environments. This project explores how a USB Wi-Fi adapter—configured for monitor mode in Kali Linux—can be used in two distinct but related security testing scenarios:

1. Attempting to crack the Wi-Fi password with a Wi-Fi adapter using *airmon-ng* and dictionary-based attacks.
2. Decrypting wireless traffic from other devices on the same network by entering the known WPA2 password into Wireshark.

By walking through the full setup and usage of a monitor-mode-compatible adapter, this project demonstrates how wireless traffic can be intercepted, decrypted, and analyzed—or potentially compromised—using open-source tools in a lab setting. These techniques mirror those used by penetration testers and ethical hackers and reinforce the importance of robust wireless security configurations.

Purpose

The purpose of this project is to gain practical experience in using a wireless network adapter for advanced traffic analysis and security testing. The key focus for this project is:

- Setting up the adapter to monitor all wireless traffic in the area.
- Decrypting WPA2 traffic on a known network using Wireshark.
- Capturing handshake packets to test Wi-Fi password cracking techniques.

Through these activities, the project provides insight into how encrypted wireless communication can still leak useful metadata, and how weak or reused passwords can put entire networks at risk. It also emphasizes ethical practices and responsible use of network monitoring tools.

Objectives

The primary objectives of this project are:

- ➢ Set up and configure a USB Wi-Fi adapter to operate in monitor mode on Kali Linux.
- ➢ Capture WPA2 4-way handshake with airmon-ng
- ➢ Perform a basic dictionary attack to crack the Wi-Fi password from captured handshakes.
- ➢ Decrypt WPA2-encrypted traffic in Wireshark using the known password.
- ➢ Capture and examine TLS-secured traffic.
- ➢ Recommend practical steps to improve home wireless network security.

⚠ Disclaimer: This experiment was performed solely on an owned home network and devices for educational purposes. No unauthorized access or testing was attempted on any network. ⚠

Scope

This project was conducted in a home lab environment using commonly available hardware and open-source tools. The central component is a **Panda PAU0D AC1200 USB Wi-Fi adapter**, chosen for its support of monitor mode and packet injection. The adapter was connected to a Kali Linux virtual machine running on a Lenovo ThinkPad with Windows 11.

This project involved two main scenarios:

1. **Cracking the Wi-Fi Password:**
   This project tested the adapter's ability to capture WPA2 handshake packets. After collecting handshakes, dictionary-based cracking techniques were explored using tools like aircrack-ng. This exercise simulated how an attacker might attempt to gain access to a network with a weak password.
2. **Decrypting Wireless Traffic:**
   Using a USB Wi-Fi adapter in monitor mode, I captured traffic from a WPA2-secured home network. After recovering the pre-shared key, I entered it into Wireshark to decrypt packets from a Dell Inspiron laptop (IP: 192.168.1.71, MAC: 74:13:EA:4F:06:D7). This enabled live traffic

analysis, including full HTTPS visibility by importing TLS session keys to decrypt previously encrypted web communications.

Glossary

Term	Description
IEEE 802.11	The set of standards that define wireless LAN (Wi-Fi) communication.
Airmon-ng	Part of the Aircrack-ng suite. Enables monitor mode on wireless adapters.
Aircrack-ng	A suite of tools used to assess Wi-Fi network security. It's commonly used for capturing and cracking WPA/WPA2 handshakes.
Brute Force Attack	Trial-and-error method of guessing credentials by systematically testing all possible combinations until the correct one is found.
BSSID	Basic Service Set Identifier; the MAC address of the wireless access point.
Deauthentication Attack	A type of attack that forces a device to disconnect and reconnect, enabling capture of a WPA2 handshake.
Dictionary Attack	A password-cracking method that tries each word from a predefined list (like rockyou.txt) to find a match.
EAPOL	Extensible Authentication Protocol over LAN; used during the WPA2 handshake to carry key exchange messages.
Haschat	An advanced password cracking tool.
Kali Linux	A Linux distribution tailored for penetration testing and ethical hacking.
Monitor Mode	A mode for Wi-Fi adapters that allows them to listen to all nearby wireless traffic without connecting to a network.
OSI Model	A conceptual framework that describes how data moves through a network in seven layers.
Packet Injection	The ability of a Wi-Fi adapter to send crafted packets, such as deauthentication frames.
rockyou.txt	A widely used password wordlist based on real passwords leaked from the 2009 RockYou breach.
SSID	Service Set Identifier; the public name of a Wi-Fi network.

TLS (Transport Layer Security)	A protocol that encrypts data over networks (e.g., HTTPS traffic) and operates above Wi-Fi encryption.
Wireshark	A network packet analyzer used to inspect and analyze captured network traffic.
WPA2 Handshake	A 4-step authentication exchange that happens when a client connects to a WPA2-protected Wi-Fi network.

Background Research / Setup Considerations

Technical Background

Before performing a Wi-Fi packet capture, or attempting a WPA2 password cracking, it's essential to understand the hardware requirements, wireless standards, and how these processes work within the OSI model. Wi-Fi communication follows the IEEE 802.11 standard, which primarily operates at two layers of the OSI model:

- **Layer 1 – Physical Layer**: Handles signal transmission over radio frequencies, including channel selection (2.4GHz/5GHz) and modulation. Monitor mode — the ability to listen to all nearby wireless traffic — is a function of this layer.
- **Layer 2 – Data Link Layer (MAC sublayer)**: Deals with 802.11 frame construction, MAC addressing, and control frames. Packet injections and WPA2 handshake capture occur at this layer.

Setup Requirements

To perform a successful wireless capture and password cracking attempt, your wireless adapter must meet the following minimum requirements:

- **Support for Monitor Mode** – to passively capture all wireless frames on a given channel.
- **Support for Packet Injection** – to actively send packets (e.g., de-authentication frames).
- **Support for both 2.4GHz and 5GHz bands** – for visibility across modern networks.
- **Access to higher channels (36 and above)** – commonly used by dual-band routers.

- **Driver compatibility with Kali Linux** – to ensure full functionality.

It is recommended to avoid Realtek-based adapters, as they often lack stable monitor mode and injection support, especially on 5GHz. Instead, choose adapters with chipsets that are well-supported in Kali and widely used for wireless testing:

- Atheros
- Ralink/MediaTek
- Broadcom (with limited but improving support)

WPA2 Handshake Capture

To attempt WPA2 password cracking, you must capture the 4-way handshake, which occurs when a client connects or reconnects to the access point. This handshake contains cryptographic material needed for offline attacks. However, handshake capture depends entirely on timing — if no client is actively connecting during your capture, no handshake will appear. In the event that there is not an active connection during the attempted capture, you can:

- Passively wait for a client to reconnect.
- Or use packet injections to send de-authentication packets, forcing a device to disconnect and reconnect, triggering the handshake.

To perform a basic deauthentication attack using, the following command can be used:

➢ *aireplay-ng --deauth 100 -a <AP MAC> -c <Client MAC> <interface>*

- --deauth 100: Sends 100 deauthentication frames.
- -a: Specifies the MAC address of the access point (target router).
- -c: Specifies the MAC address of the client device to disconnect.
- <interface>: Refers to the wireless adapter in monitor mode (e.g., wlan0mon).

Once captured, the handshake can be exported and used with tools like aircrack-ng for cracking the password.

Wordlists and Dictionary Attacks

A dictionary attack is a basic form of brute force attack where attackers try to crack a password by rapidly testing a list of common words, phrases, and number combinations until they find a match. Weak or reused passwords are one of the most common vulnerabilities in wireless security. According to Kaspersky, a Google study found that 65% of users reuse passwords across accounts, and 59% rely on personal details that are easy to guess or discover.[1] This creates ideal conditions for attackers to exploit—especially when passwords are short, predictable, or drawn from publicly known leaks.

In WPA2 cracking, attackers often rely on dictionary attacks, where each password in a wordlist is tested against a captured 4-way handshake to find the correct one. Tools like aircrack-ng automate this process by comparing each candidate password against the captured authentication data. In this project, I used the rockyou.txt wordlist[2]—an archive of millions of real-world passwords exposed in a 2009 data breach—to simulate this kind of attack. Because many users choose simple or common passphrases, tools like aircrack-ng paired with rockyou.txt can quickly crack poorly secured Wi-Fi networks.

Characteristics of a Strong Password

Strong passwords are essential for defending wireless networks against password-cracking tools such as Hashcat and Aircrack-ng. Weak passwords are easily broken when they have any of the following characteristics:

- Short length (e.g., fewer than 12 characters)
- Use of common dictionary words (e.g., "football", "qwerty")
- Simple patterns (e.g., "abc123", "password1")
- Repeating characters or predictable sequences (e.g., "aaaa1111", "1234567")

A strong password should include:

- Be long — ideally 16 characters or more
- Include a mix of uppercase and lowercase letters, numbers, and special characters
- Avoid recognizable words, dates, or common sequences
- Be randomly generated when possible, rather than human-created

---

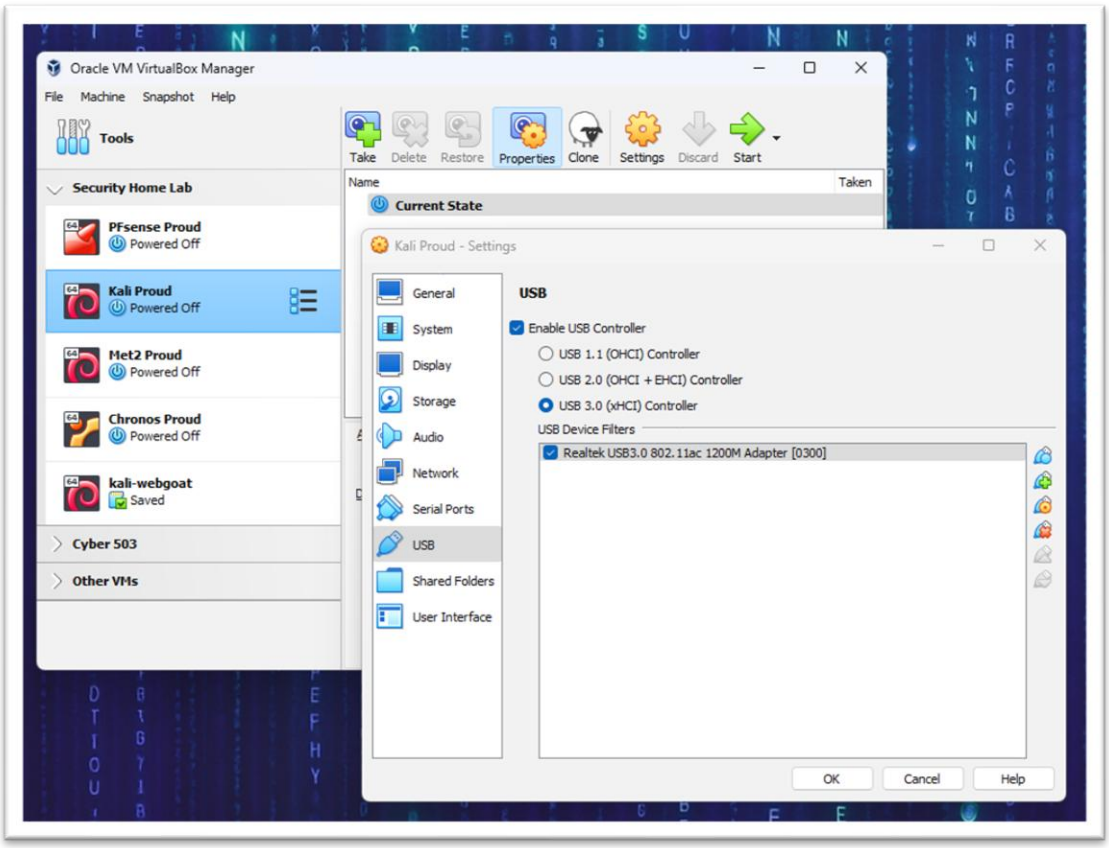
[1] Kaspersky. (n.d.). *What is a dictionary attack?* Kaspersky. https://www.kaspersky.com/resource-center/definitions/what-is-a-dictionary-attack

[2] Burns, W. J. (n.d.). *Common password list (rockyou.txt)* [Dataset]. Kaggle. https://www.kaggle.com/datasets/wjburns/common-password-list-rockyoutxt

Research shows that passwords built from random combinations are significantly harder to crack, especially against dictionary and rule-based attacks (*St. Cloud State University, 2024*). In wireless network security, random and complex passwords are the best defense against attacks using common wordlists like *rockyou.txt*.[3]

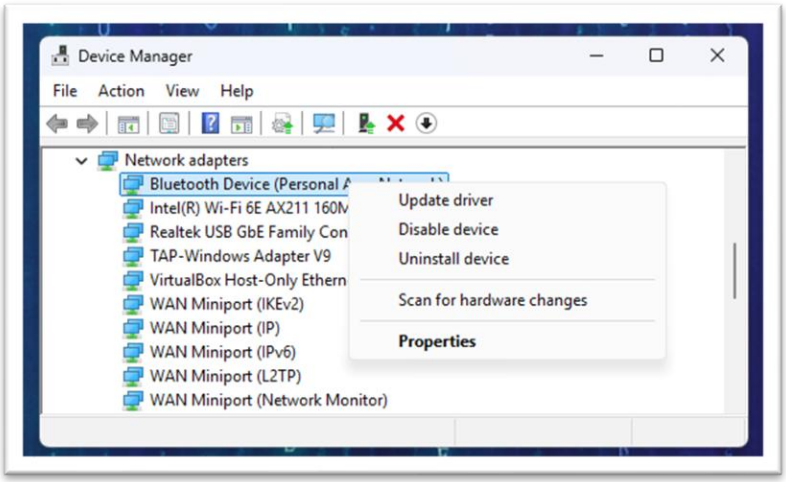
Setting Up the Adapter and wlan0 on Kali Linux

- Insert the Panda PAU0D USB Wi-Fi adapter into your host (Windows) machine. In VirtualBox, go to **Settings > USB** for your Kali VM and add the Panda adapter using the USB filter list. The name of your adapter shown may not match the retail brand name, it will often display the chipset manufacturer.



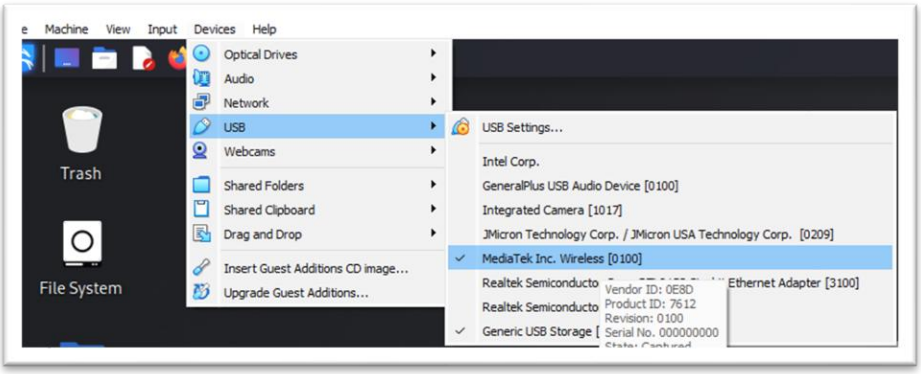
[3] St. Cloud State University. (2024). A Study on the Security of Password Hashing with GPU-Based Cracking Methods. St. Cloud State University Repository. https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1032&context=msia_etds

- Open **Device Manager** (Ctrl + c) on the host and disable the adapter under **Network Adapters** to prevent the host from taking control.

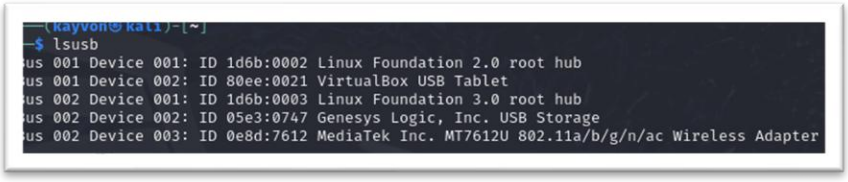


- Unplug the adapter and Launch the Kali Linux virtual machine BEFORE plugging the adapter back in.

- Once Kali is running, plug in the adapter and check if Kali recognizes the adapter through **Devices > USB** in the VirtualBox toolbar and ensure the adapter is selected (check-marked).



- Verify the adapter is detected by Kali, open a terminal in Kali and run:
  - *Lsusb*

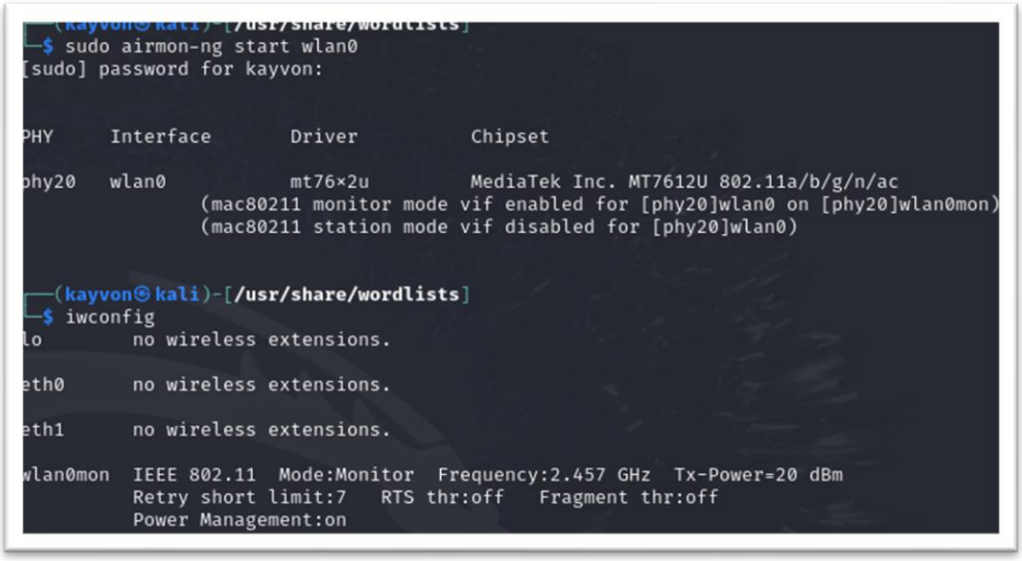


- Set the adapter to **monitor** mode by running:

➢ *sudo airmon-ng start wlan0*

❖ Confirm the interface is in monitor mode using:

➢ *Iwconfig*



This displays the USB Wi-Fi adapter is now set up on Wlan0 with monitor mode enabled.

Cracking the Wi-Fi Password

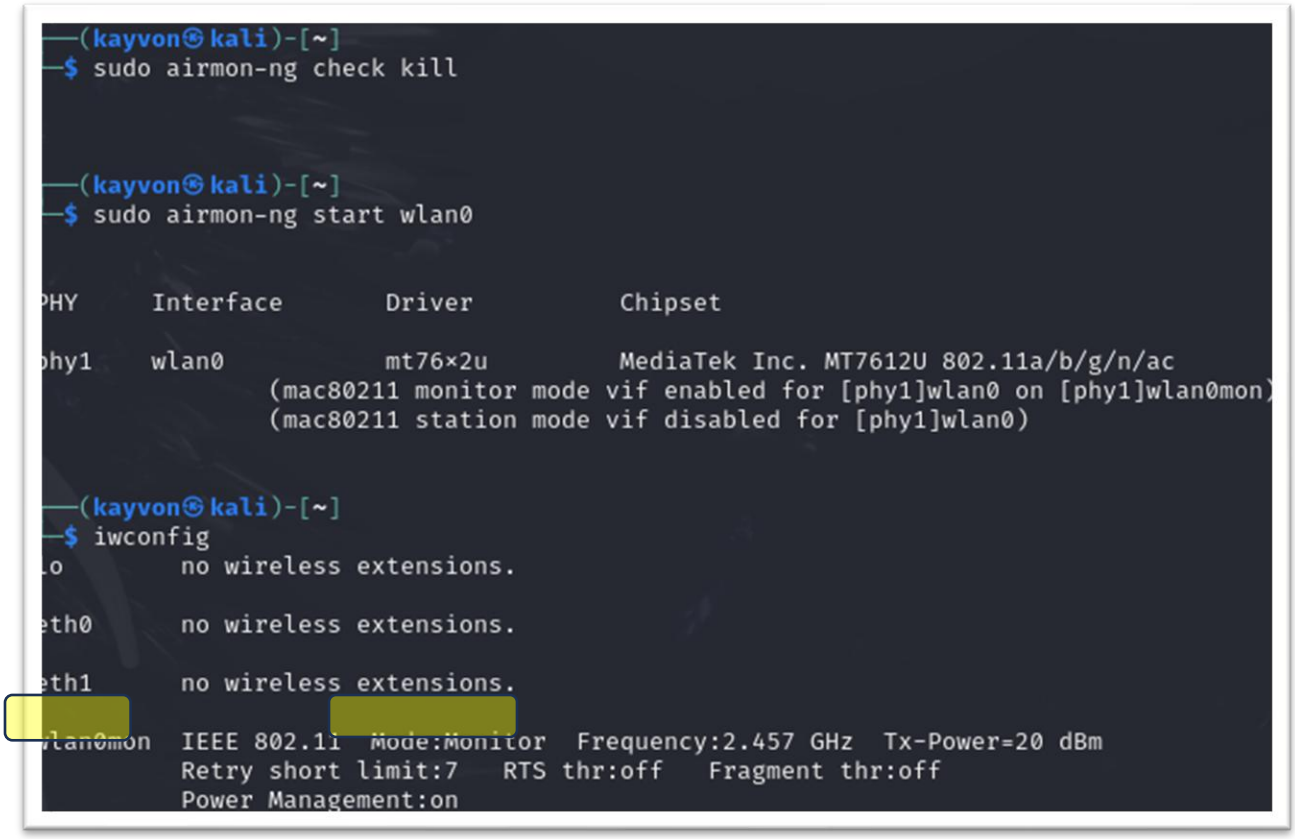
Before trying to crack any passwords, the USB Wi-Fi adapter was switched into monitor mode using *airmon-ng*, as shown in the previous step. This allows the device to listen to all wireless traffic on a specific channel without joining the network—basically just watching what's going on. Monitor mode is essential for picking up WPA2 handshakes, which happens when a device connects to a Wi-Fi network. With this setup, the project shows how an attacker could use free, open-source tools to capture a handshake and attempt a dictionary attack on it. The goal was to see if the home Wi-Fi password could be cracked using the common *rockyou.txt* wordlist, demonstrating how risky weak or reused passwords can be—even on WPA2-secured networks.

Steps to crack Wi-Fi password

1) Kill any airmon-ng processes to avoid conflict, turn the Wi-Fi adapter to monitor mode, and verify:

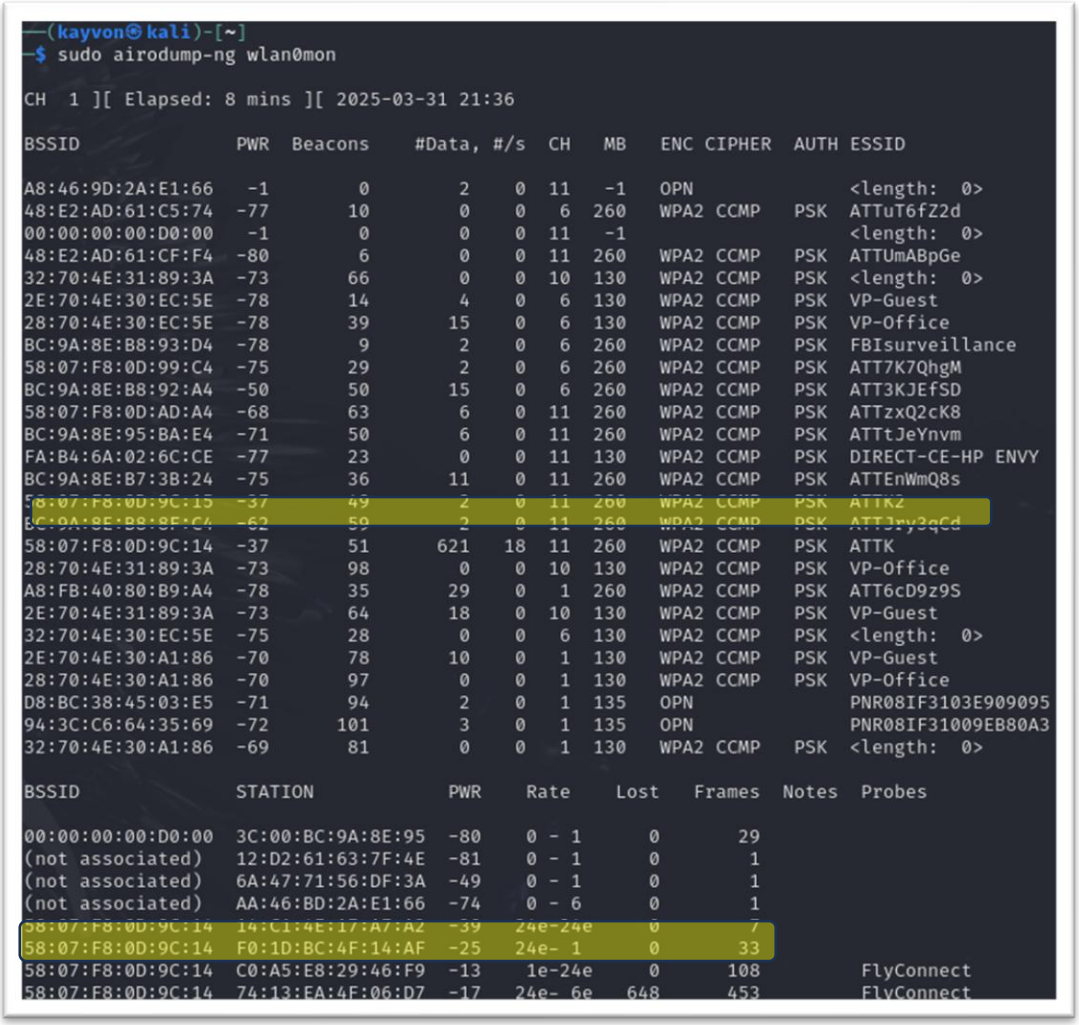
❖ *sudo airmon-ng check kill*

- *sudo airmon-ng start wlan0*
- *iwconfig*
- *(This will change your wlan0 to wlan0mon)*



2) Identify the access point of your target device

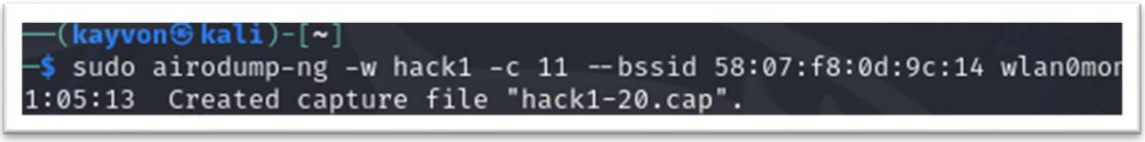
- run '*sudo airodump-ng wlan0mon*'
- Take note of the access point MAC address, BSSID
- We see in the frame below that the target device on ATTK is operating on channel 11 with BSSID number of 58:07:f8:0d:9c:14.

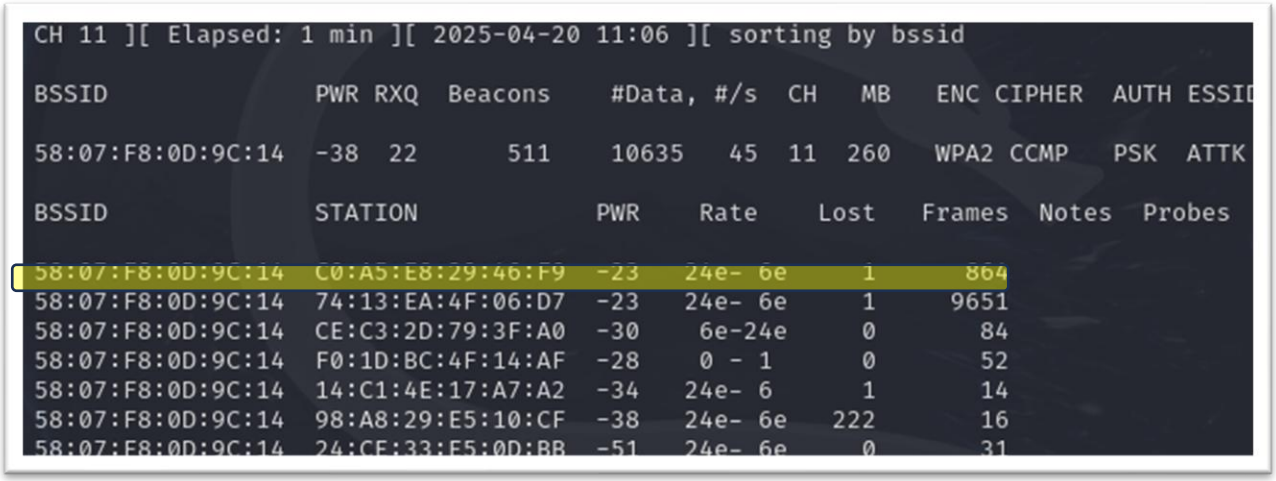


3) Start a Wi-Fi packet capture targeting the ATTK network (BSSID: 58:07:f8:0d:9c:14) on Channel 11, using monitor mode (wlan0mon).

Run the following command to write the scan to a file. Keep in mind that in this demonstration, I labeled the file ‘hack1’ for organizational purposes, but you can utilize any naming methodology, if preferred.

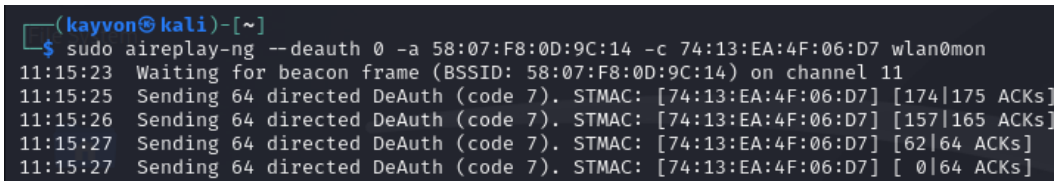
- *sudo airodump-ng -w hack1 -c 11 --bssid 58:07:f8:0d:9c:14 wlan0mon*

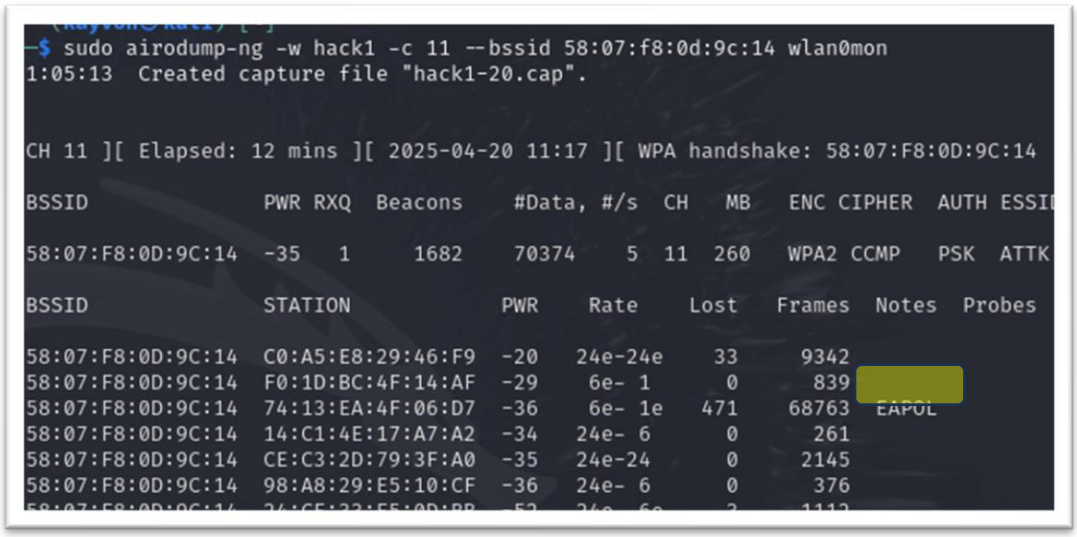




4) Capture the 4-way handshake from a NEW terminal by running:

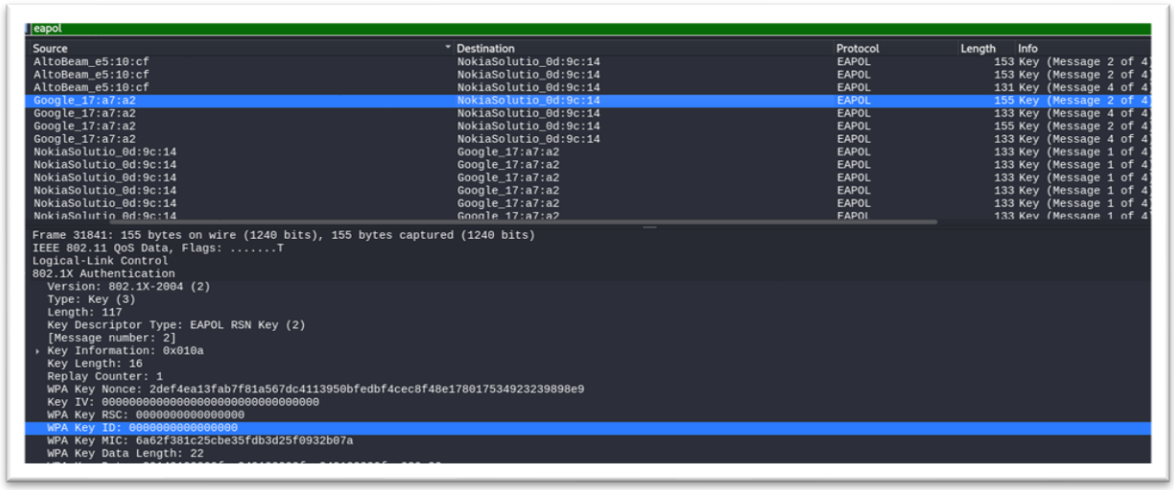
- *sudo aireplay-ng --deauth 0 -a 58:07:F8:0D:9C:14 -c 74:13:EA:4F:06:D7 wlan0mon*
- This attack temporarily disconnected all clients from the target Wi-Fi network, forcing them to reconnect. During this reconnection, you need to keep airodump-ng running to capture the WPA2 handshake.





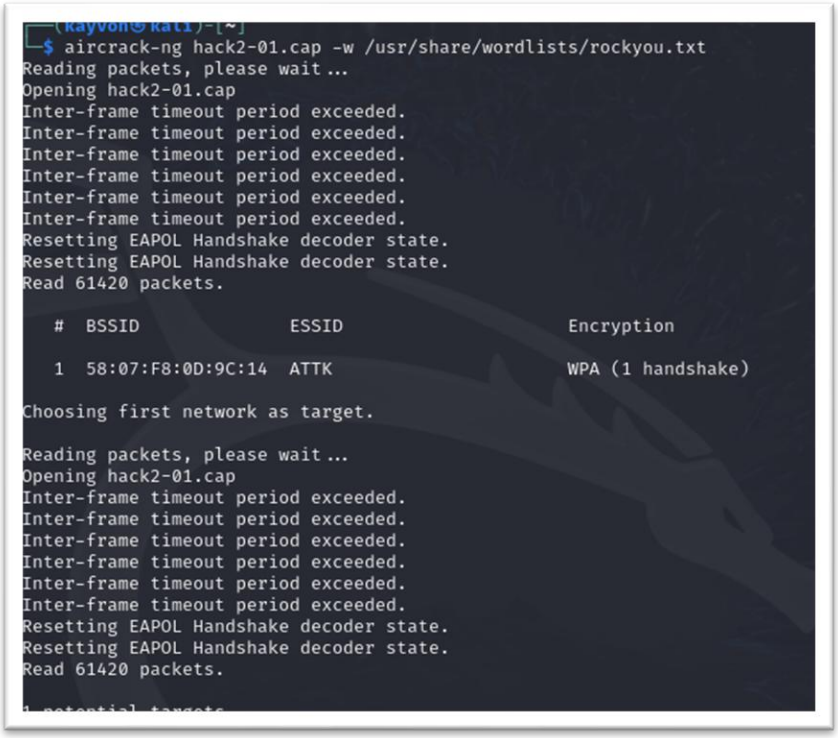
5) Open the file in Wireshark in the same active directory and inspect the EAPOL handshake packets.

- *wireshark hack1-20.cap* (For me it was the 20th scan, your first scan will be labeled *-01*)
- Enter 'eapol' in the filter to inspect handshake packets
- Expand the EAPOL packet labeled 'message 2 of 4'
- In the bottom panel, expand 802.1x Authentication and find 'WPA Key ID'
- This is what needs to be cracked!

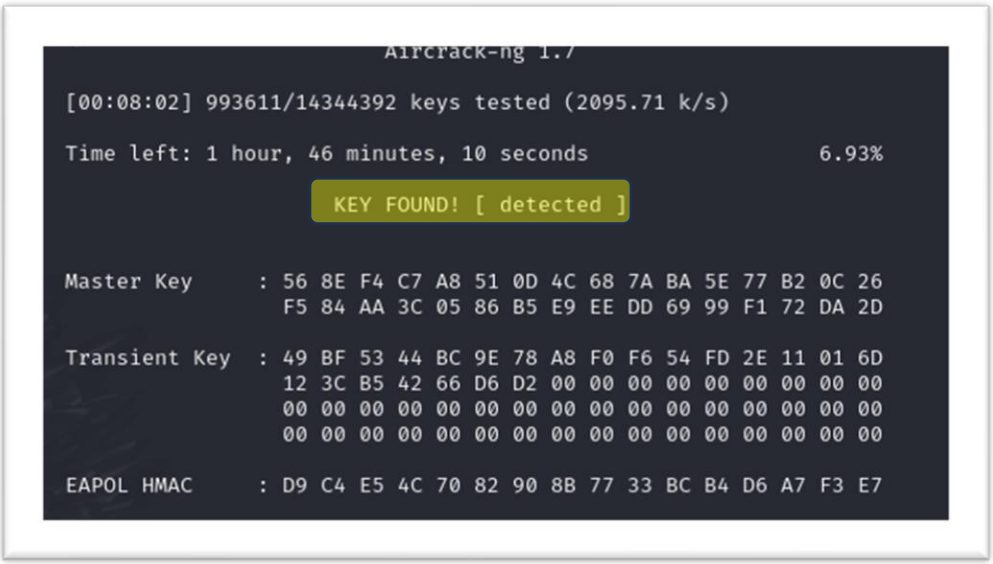


6) Extracting and cracking the WPA2 Handshake

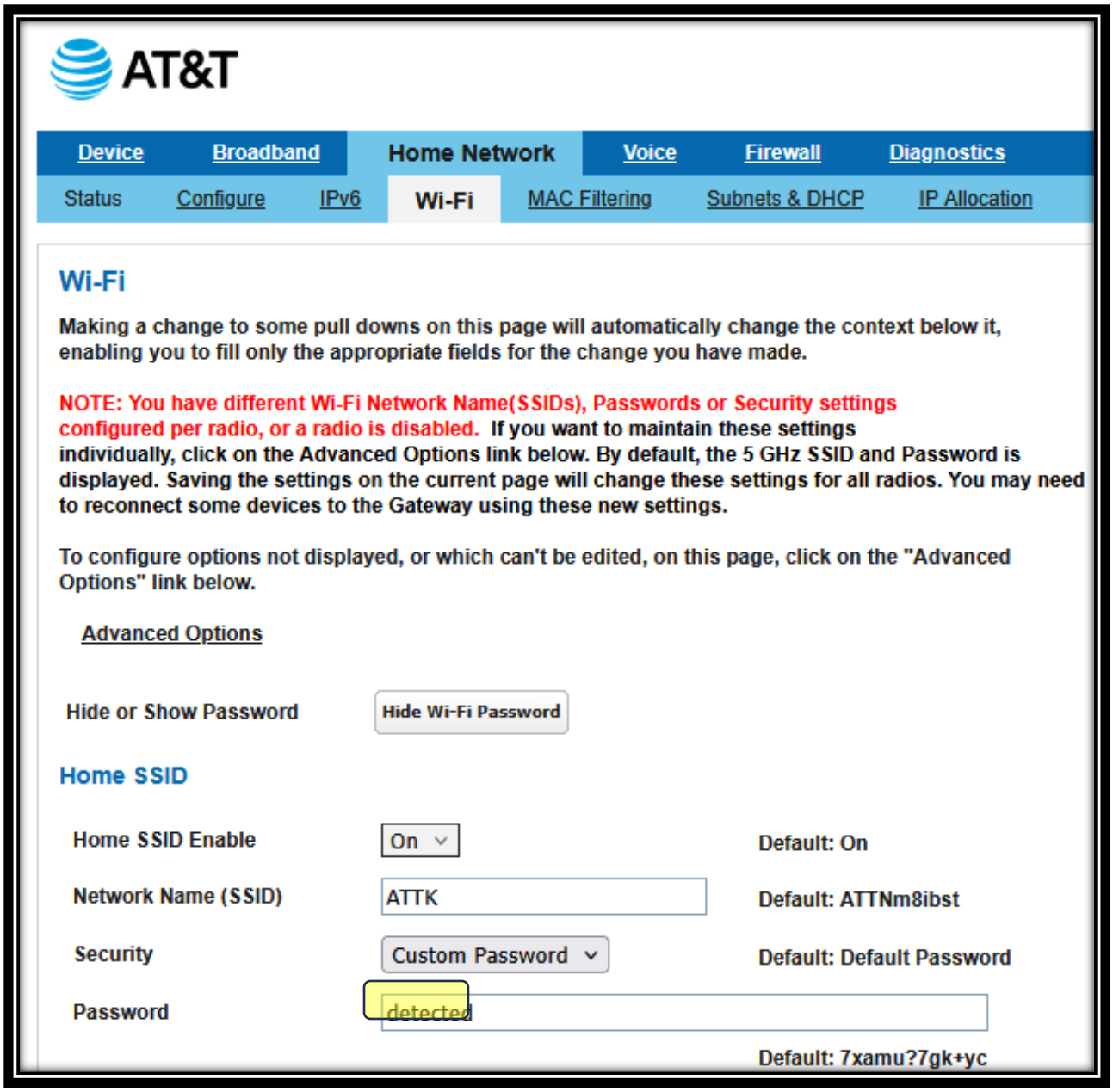
- Put adapter from monitor mode back to managed mode by running:
  - *sudo airmon-ng stop wlan0mon*
- Navigate to /usr/share/wordlists and verify the presence of the rockyou.txt
- Use aircrack-ng with the rockyou.txt wordlist to begin performing a dictionary attack for cracking
  - *aircrack-ng hack2-01.cap -w /usr/share/wordlists/rockyou.txt -b 58:07:F8:0D:9C:14*



- Aircrack-ng began testing each password in the wordlist against the captured handshake
- Cracking ran at ~2,000–2,900 keys/sec.
- After testing ~993,611 keys, the correct WPA2 passphrase was found:
  - KEY FOUND! [detected]



Given that this was conducted on my own home network, I'm able to confirm that the password for the target network was cracked. The image below from the router management console confirms this.



Understanding the RockYou.txt Wordlist

This section outlines the final phase of the WPA2 cracking process: using a captured handshake file to run an offline dictionary attack with aircrack-ng and the widely used rockyou.txt wordlist. The handshake was captured from a Dell device (74:13:EA:4F:06:D7) during an active session on the ATTK access point (58:07:F8:0D:9C:14). After forcing a reconnection via deauthentication, the resulting handshake was saved as hack1-20.cap, setting the stage for an offline password recovery attempt.

The objective was to replicate a realistic attack scenario where a client is kicked off a network, the handshake is intercepted, and an attacker attempts to break the encryption using a list of commonly used passwords. This mirrors a standard technique in Wi-Fi penetration testing aimed at revealing weak password practices in WPA2-secured environments.

The command used was:

aircrack-ng hack2-01.cap -w /usr/share/wordlists/rockyou.txt -b 58:07:F8:0D:9C:14

- hack2-01.cap — the capture file containing the WPA2 handshake
- -w — specifies the wordlist
- -b — targets the specific BSSID of the access point

Aircrack-ng then proceeded to systematically test each password in the wordlist against the handshake using the Pre-Shared Key (PSK) authentication challenge in the WPA2 protocol. To crack the WPA2 handshake, I used the rockyou.txt wordlist, one of the most widely used password dictionaries in penetration testing. Originally derived from real leaked passwords exposed in the RockYou.com data breach (2009), this list contains millions of commonly used, weak, and reused passwords. It's often used in brute-force and dictionary attacks due to its real-world relevance.

In Kali Linux, the wordlist is stored by default at:

- */usr/share/wordlists/rockyou.txt.gz*

Before using it, the file must be unzipped with the command:

- *gunzip /usr/share/wordlists/rockyou.txt.gz*

For the experiment above, the rockyou.txt file was already previously extracted. Once extracted, I used rockyou.txt with aircrack-ng to perform a dictionary attack against the WPA2 handshake.

- *aircrack-ng hack2-01.cap -w /usr/share/wordlists/rockyou.txt -b 58:07:F8:0D:9C:14*

The tool systematically tested each password in the list against the captured handshake until it found a match:

- *KEY FOUND! [detected]*

Once the handshake was saved to hack1-20.cap, the attack no longer required any connection to the access point or device. Using aircrack-ng, each password in the rockyou.txt wordlist was tested against the captured handshake offline. The correct Wi-Fi password was identified within minutes, confirming that the network was vulnerable due to its use of a weak, commonly used credential. This outcome reinforces a key point: WPA2 encryption alone doesn't guarantee security. If the password is predictable or reused, it can be cracked with basic tools and publicly available wordlists—no sustained access or advanced exploits are needed.

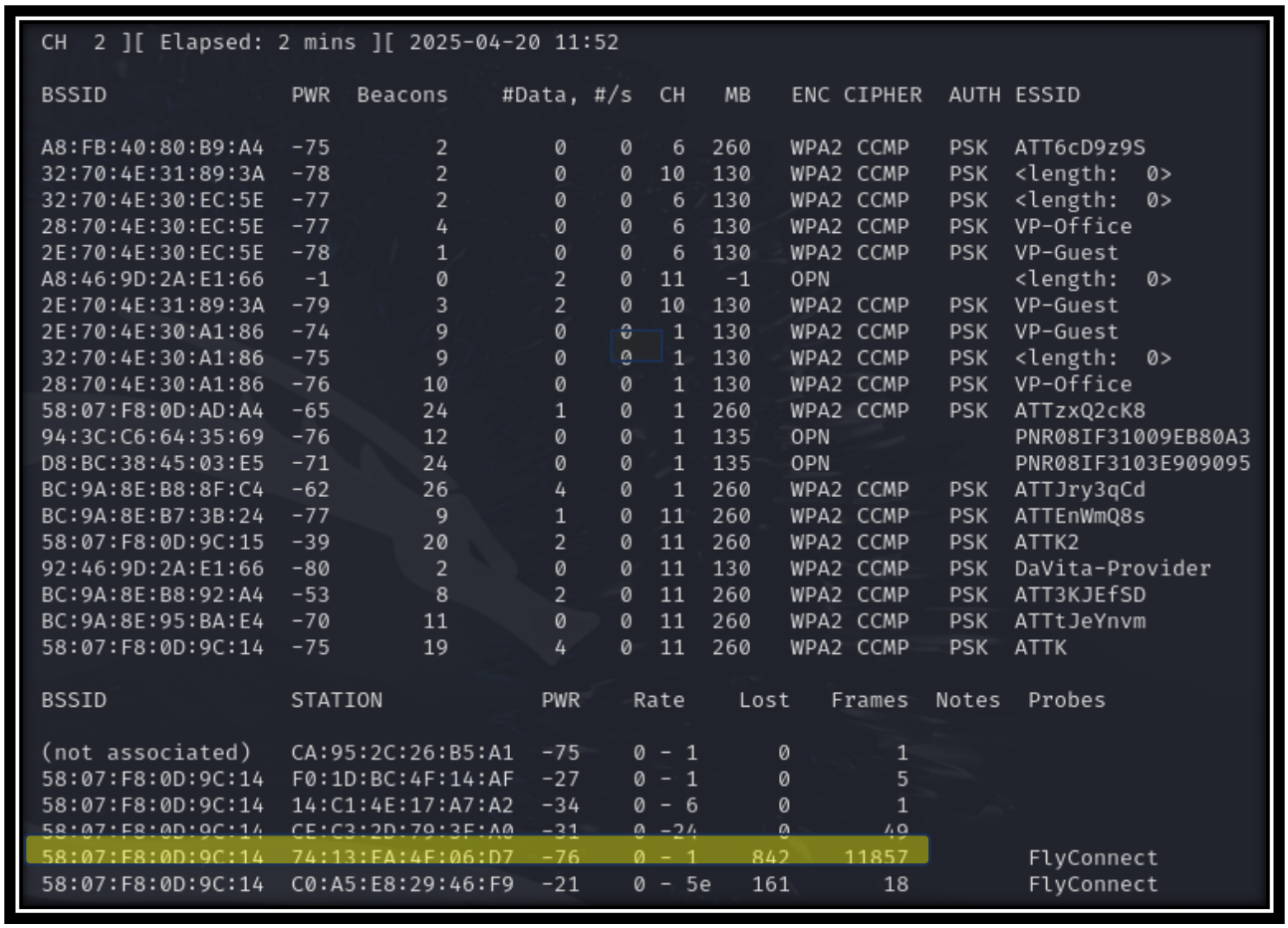
Decrypting Wi-Fi Traffic

After capturing the WPA2 handshake and recovering the correct Wi-Fi password, the next step was decrypting the wireless traffic in Wireshark. This process starts with identifying the target network, capturing the needed packets, and then loading the capture file into Wireshark for analysis. By entering the password in Wireshark's IEEE 802.11 settings, encrypted frames can be decrypted, making it possible to see device activity and uncover sensitive metadata.

This section shows how someone can decrypt wireless traffic by capturing the WPA2 handshake, using a known password, and working with a wireless adapter in monitor mode from the beginning. If you've already followed the previous steps to capture the handshake and obtain the password, you can skip ahead to Step 6, where the actual decryption in Wireshark begins.

Steps:

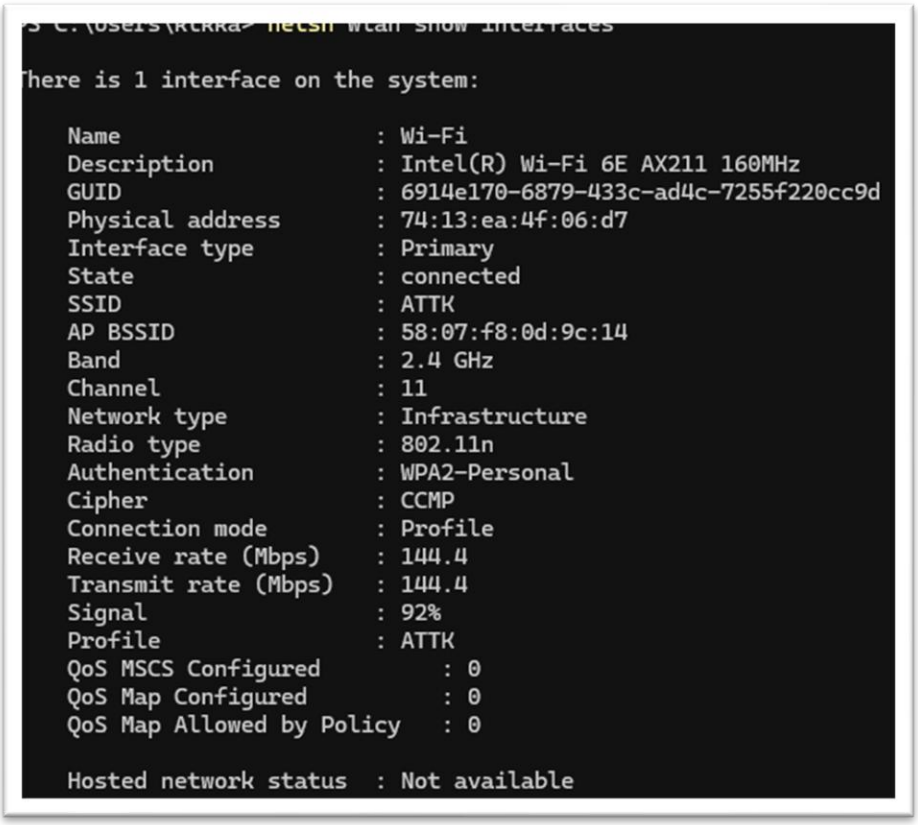
1) **Use Airodump-ng to scan nearby wireless networks and find target:**
   - *sudo airodump-ng wlan0*
   - It was confirmed that the target access point ATTK was operating on channel 11 and can confirm the BSSID and MAC address of the target device
   - BSSID: 58:07:F8:0D:9C:1C
   - MAC: 74:13:EA:4F:06:D7



```
CH  2 ][ Elapsed: 2 mins ][ 2025-04-20 11:52

BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

A8:FB:40:80:B9:A4  -75        2        0    0   6  260   WPA2 CCMP   PSK  ATT6cD9z9S
32:70:4E:31:89:3A  -78        2        0    0  10  130   WPA2 CCMP   PSK  <length:  0>
32:70:4E:30:EC:5E  -77        2        0    0   6  130   WPA2 CCMP   PSK  <length:  0>
28:70:4E:30:EC:5E  -77        4        0    0   6  130   WPA2 CCMP   PSK  VP-Office
2E:70:4E:30:EC:5E  -78        1        0    0   6  130   WPA2 CCMP   PSK  VP-Guest
A8:46:9D:2A:E1:66   -1        0        2    0  11   -1   OPN              <length:  0>
2E:70:4E:31:89:3A  -79        3        2    0  10  130   WPA2 CCMP   PSK  VP-Guest
2E:70:4E:30:A1:86  -74        9        0    0   1  130   WPA2 CCMP   PSK  VP-Guest
32:70:4E:30:A1:86  -75        9        0    0   1  130   WPA2 CCMP   PSK  <length:  0>
28:70:4E:30:A1:86  -76       10        0    0   1  130   WPA2 CCMP   PSK  VP-Office
58:07:F8:0D:AD:A4  -65       24        1    0   1  260   WPA2 CCMP   PSK  ATTzxQ2cK8
94:3C:C6:64:35:69  -76       12        0    0   1  135   OPN              PNR08IF31009EB80A3
D8:BC:38:45:03:E5  -71       24        0    0   1  135   OPN              PNR08IF3103E909095
BC:9A:8E:B8:8F:C4  -62       26        4    0   1  260   WPA2 CCMP   PSK  ATTJry3qCd
BC:9A:8E:B7:3B:24  -77        9        1    0  11  260   WPA2 CCMP   PSK  ATTEnWmQ8s
58:07:F8:0D:9C:15  -39       20        2    0  11  260   WPA2 CCMP   PSK  ATTK2
92:46:9D:2A:E1:66  -80        2        0    0  11  130   WPA2 CCMP   PSK  DaVita-Provider
BC:9A:8E:B8:92:A4  -53        8        2    0  11  260   WPA2 CCMP   PSK  ATT3KJEfSD
BC:9A:8E:95:BA:E4  -70       11        0    0  11  260   WPA2 CCMP   PSK  ATTtJeYnvm
58:07:F8:0D:9C:14  -75       19        4    0  11  260   WPA2 CCMP   PSK  ATTK

BSSID              STATION            PWR    Rate    Lost   Frames  Notes  Probes

(not associated)   CA:95:2C:26:B5:A1  -75    0 - 1      0        1
58:07:F8:0D:9C:14  F0:1D:BC:4F:14:AF  -27    0 - 1      0        5
58:07:F8:0D:9C:14  14:C1:4E:17:A7:A2  -34    0 - 6      0        1
58:07:F8:0D:9C:14  CE:C3:2D:79:3E:A0  -31    0 -24      0       49
58:07:F8:0D:9C:14  74:13:EA:4E:06:D7  -76    0 - 1    842    11857          FlyConnect
58:07:F8:0D:9C:14  C0:A5:E8:29:46:F9  -21    0 - 5e   161       18          FlyConnect
```

- Optional: Verify channel and BSSID on the target system using Command Prompt:

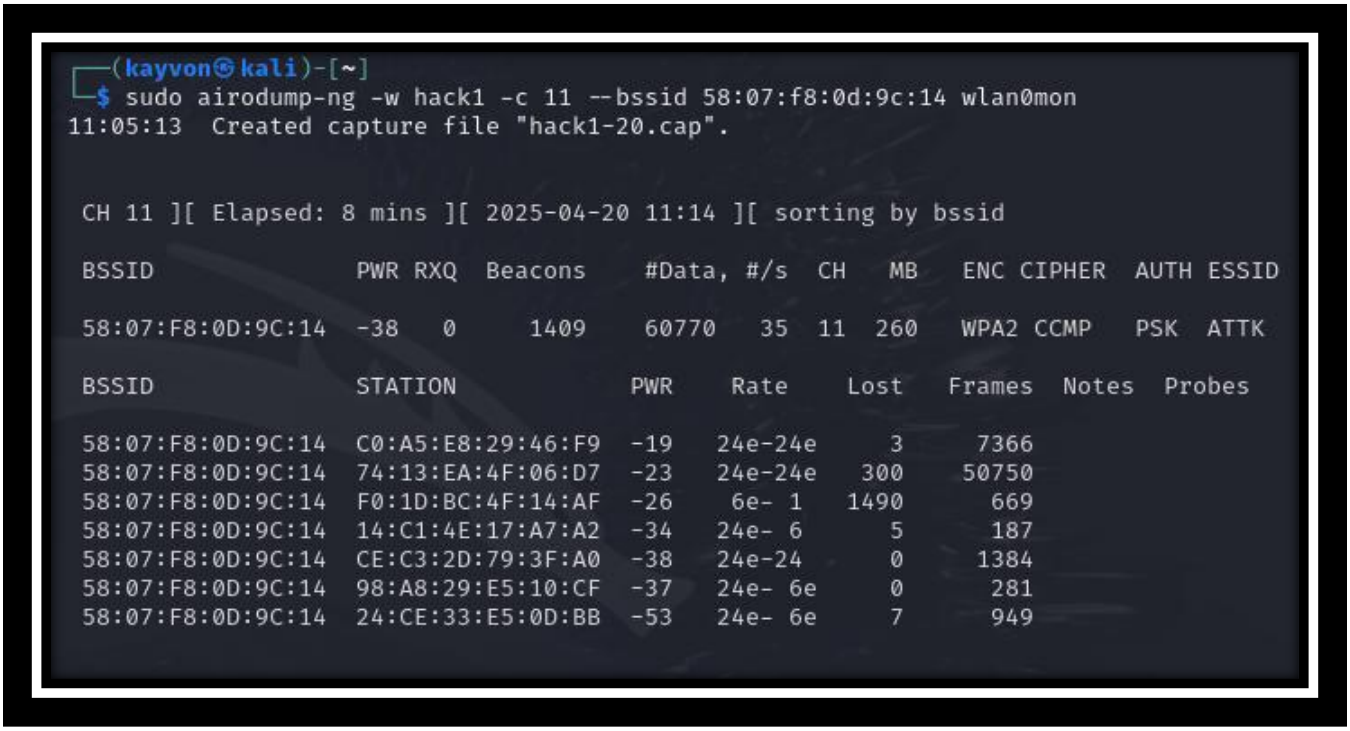
➢ *netsh wlan show interfa*



2) Run airodump-ng in targeted capture mode, specifying:

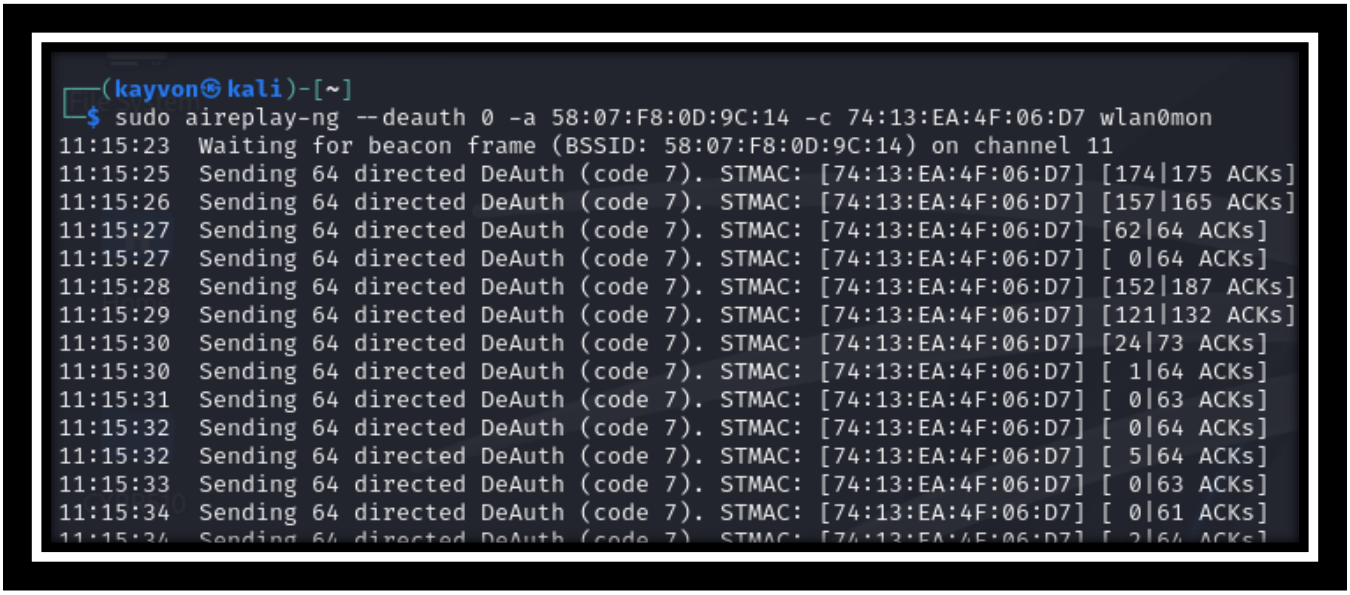
- The BSSID of your access point: (58:07:F8:0D:9C:14)
- The correct channel (11)
- A filename to save the capture (hack1.cap)
- Used wlan0mon as the monitoring interface (already in monitor mode)
- The command below creates the capture file.

➢ *sudo airodump-ng -w hack1 -c 11 --bssid 58:07:F8:0D:9C:14 wlan0mon*

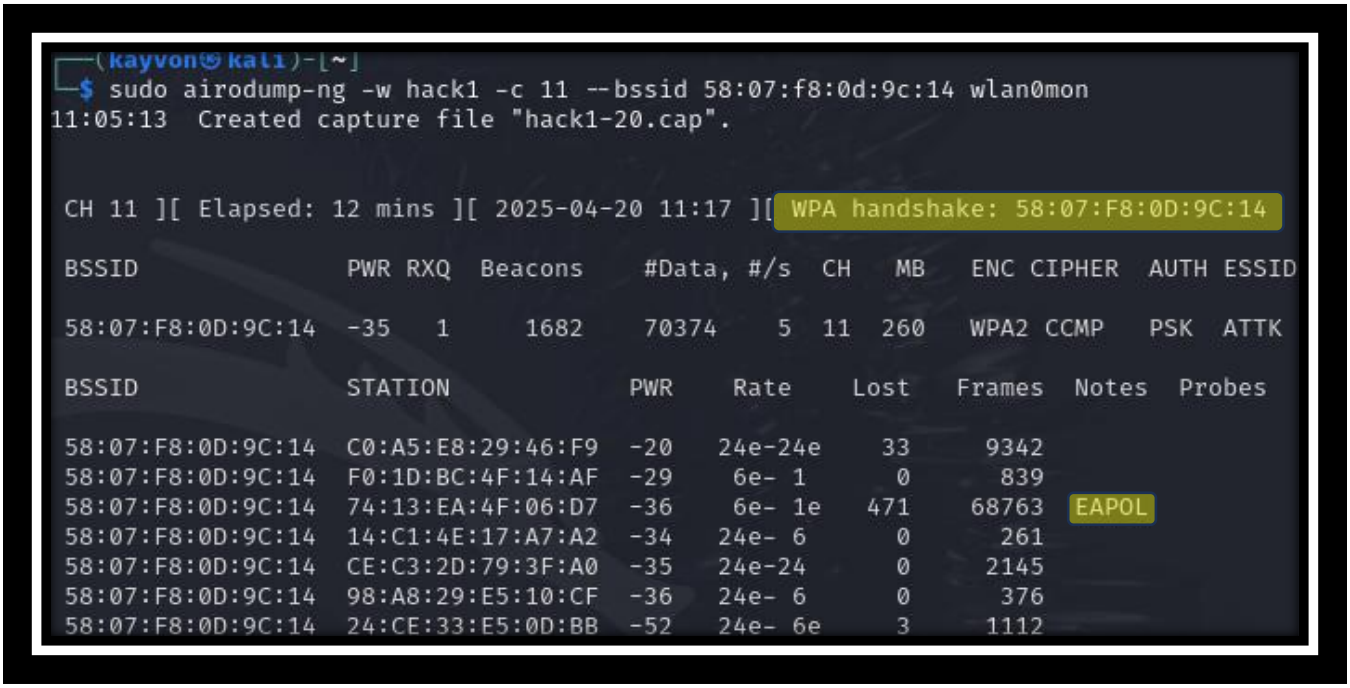


3) Confirm the 4-way handshake using a deauthentication attack:

- Use *aireplay-ng* to send deauthentication frames and force the target device to disconnect, opening vulnerabilities, triggering a handshake when it reconnects.
  - *sudo aireplay-ng --deauth 0 -a [BSSID] -c [Target MAC] wlan0mon*



- A successful handshake will result in an observed WPA handshake in the top right corner and will show EAPOL under Notes.



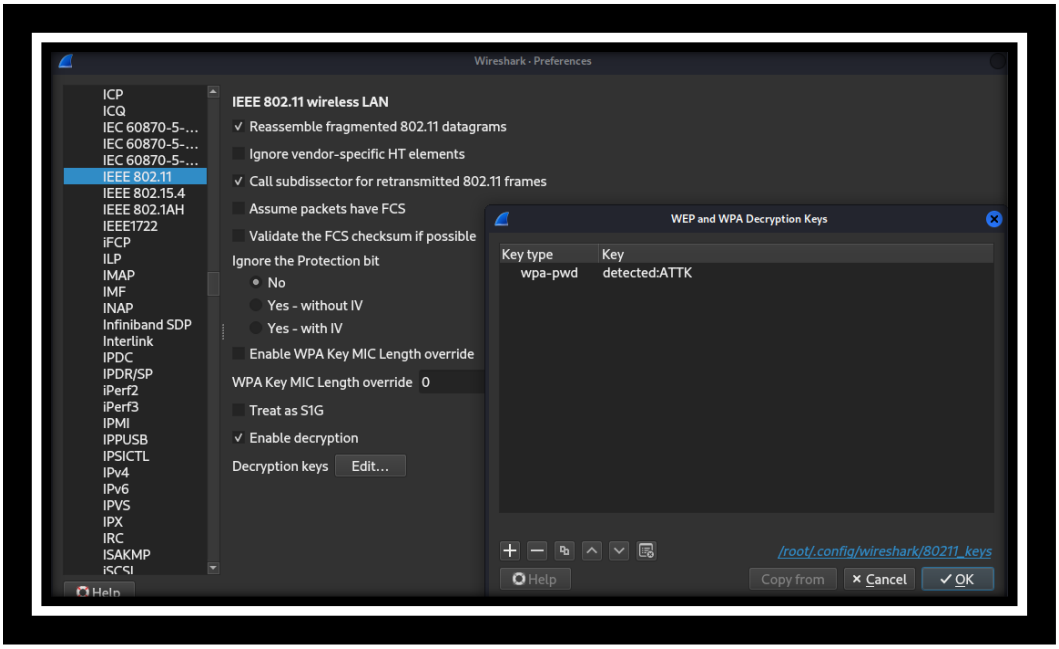
4) Stop the capture and open the capture file in Wireshark:

Once the handshake was captured and confirmed (indicated by the WPA handshake: message in airodump-ng), the capture session was stopped using Ctrl + C. This finalized the .cap file (e.g., dell-handshake-02.cap). The file was then opened in Wireshark, a packet analysis tool, to visually inspect the captured traffic. The .cap file is stored in the working directory where airodump-ng was executed.

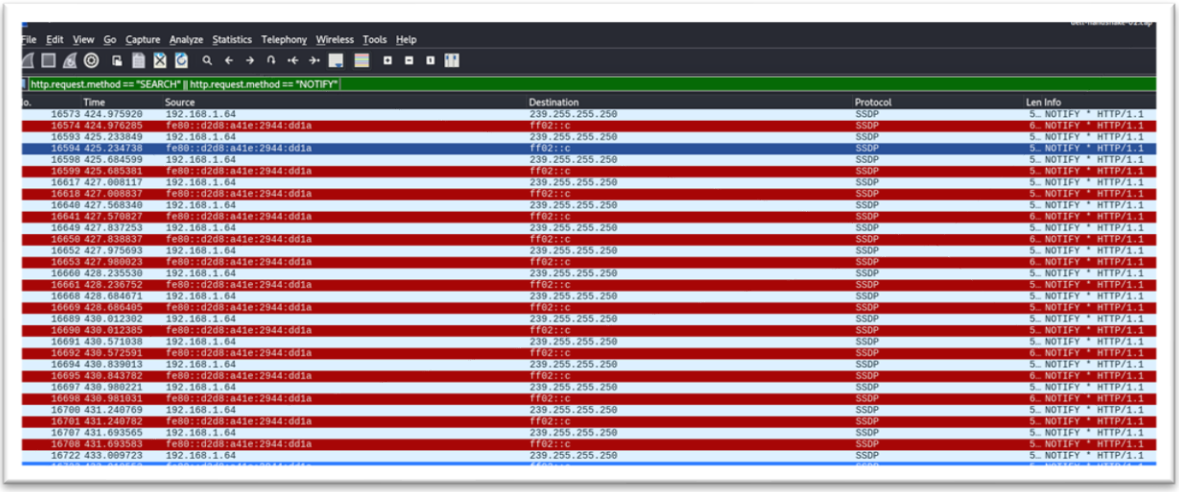
- Stop Session:
  - Ctrl + C
- Locate file in working directory:
  - *ls -l | grep.cap*
- Launch Wireshark through terminal:
  - *sudo wireshark*

5) Decrypting Captured Traffic in Wireshark:

- After opening the captured .cap file in Wireshark, the Wi-Fi password for the network (ATTK) was entered to enable decryption of encrypted 802.11 traffic.
- Edit → Preferences → Protocols → IEEE 802.11 → Decryption Keys → Edit
- A new key of type wpa-pwd was added in the format.
- Click OK.



If you see highlighted packets after inputting the WPA key like the image below, then you have successfully decrypted Wi-Fi traffic.



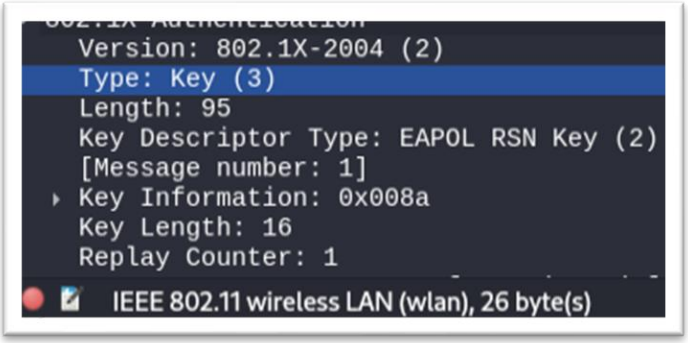
Analyzing Decrypted Traffic in Wireshark:

With the Wi-Fi password successfully recovered, the next phase involved analyzing the previously encrypted traffic using Wireshark. After entering the password into Wireshark's IEEE 802.11 decryption settings, the captured packets could be decrypted and viewed in plain text. Wireshark used

the WPA2 4-way handshake—captured during the initial packet collection—along with the correct passphrase to unlock the wireless data.

While this report does not focus on deep packet analysis, this section highlights some of the protocols observed in the decrypted traffic. It demonstrates how access to a single Wi-Fi password, combined with a captured handshake, can expose a significant amount of device activity and network metadata that would otherwise remain hidden.

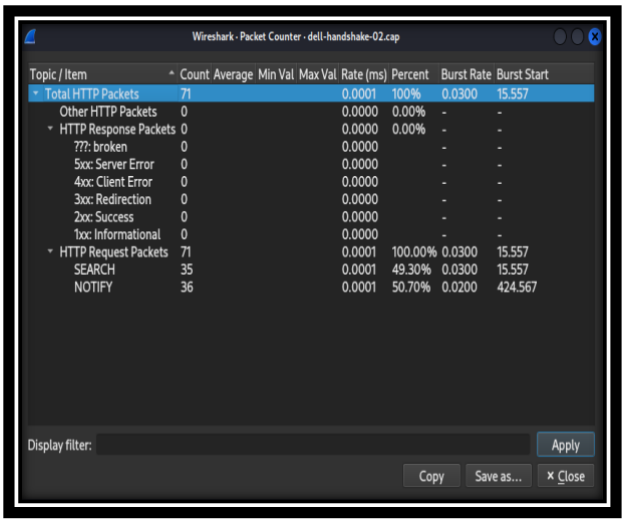
EAPOL Packet Breakdown: Message 1 of the WPA2 4-Way Handshake



This packet is part of the WPA2 4-way handshake, specifically Message 1, sent from the access point to the client. It contains the WPA Key Nonce used to generate encryption keys, and a zeroed-out MIC confirming it's the initial key exchange. This is the packet that initiated the handshake process and allowed Wireshark to decrypt the captured wireless traffic.

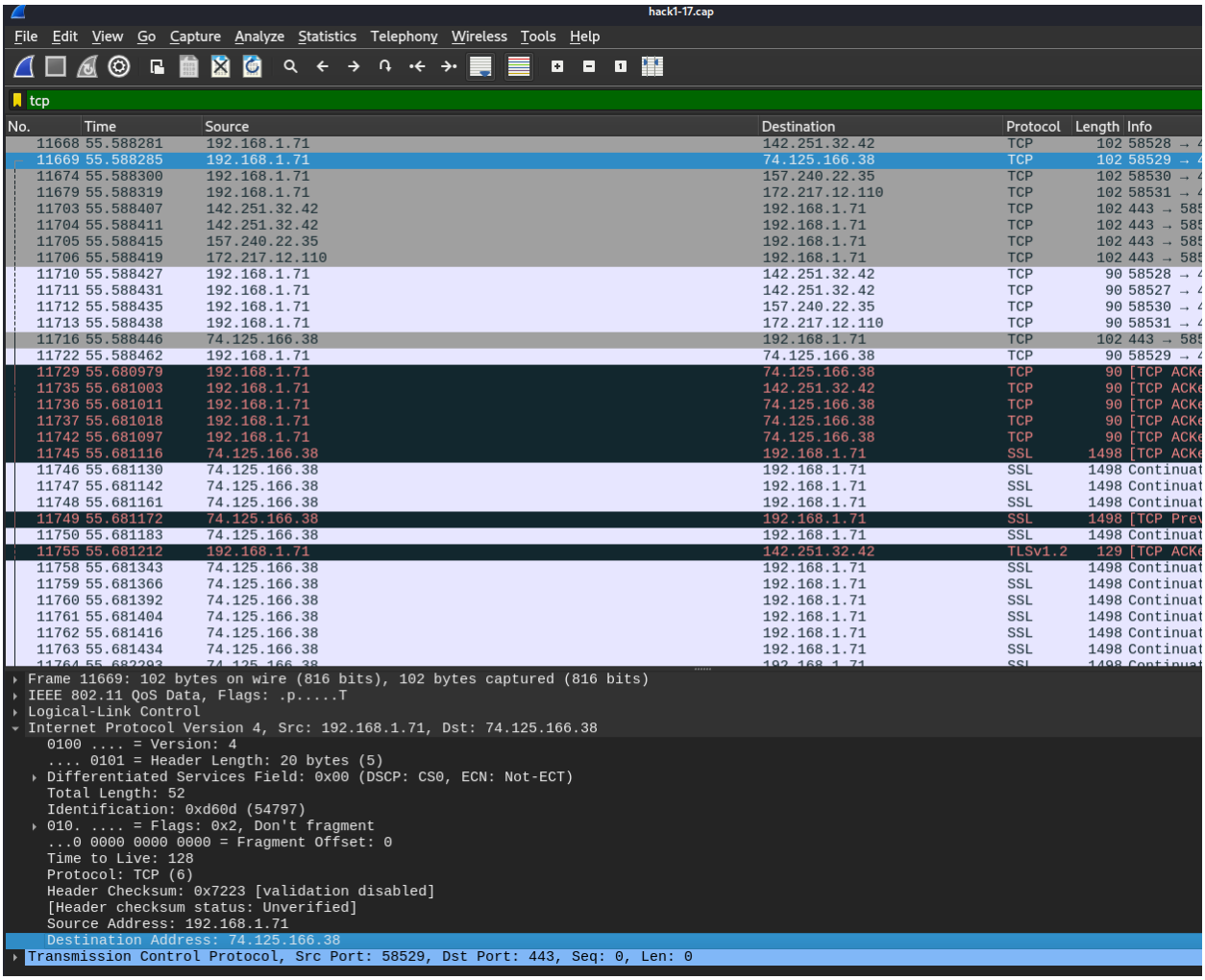
Wireshark Screenshot Analysis – TCP and TLS Traffic

The screenshot captures a section of a Wireshark packet capture, displaying a sequence of packets related to an HTTPS connection between an internal host and external servers. This snapshot offers insight into how encrypted web traffic flows from a client over a wireless network.



The packet analysis from dell-handshake-02.cap shows a total of 71 HTTP packets were captured. All of these were HTTP request packets, with 35 SEARCH requests and 36 NOTIFY messages. No HTTP response packets were recorded, including status codes such as 2xx (Success), 4xx (Client Error), or 5xx (Server Error). This pattern indicates device discovery and advertisement traffic, likely generated by services such as UPnP or similar protocols over HTTP.

Wireshark Capture Showing HTTPS Traffic: TCP 3-Way Handshake and TLS Session



- The internal IP address 192.168.1.71 is the source of most traffic, with multiple external IP addresses:
  - 74.125.166.38 (Google-owned)
  - 142.251.32.42 (also part of Google infrastructure)
- The destination port is 443, which is standard for HTTPS (secure web traffic).
- A typical TCP 3-way handshake is visible:
  - SYN from 192.168.1.71 to 74.125.166.38 (Frame 11669)
  - SYN-ACK back from the server
  - ACK from the client, completing the connection setup
- Following the handshake, the protocol switches to SSL/TLS, specifically TLSv1.2 in Frame 11750, indicating that the encrypted HTTPS session has begun.

- Multiple 1498-byte packets indicate a high volume of encrypted traffic, likely from a streaming service left active on the target device.

Why TLS Traffic Was Not Decrypted - and What It Would Take

While this project successfully decrypted WPA2 wireless traffic at the network level, the contents of many communications—such as web browsing—remained encrypted due to Transport Layer Security (TLS). TLS operates at the application layer (Layer 7 of the OSI model) and provides end-to-end encryption for protocols like HTTPS, meaning that even after Wi-Fi decryption, the payloads of TLS-encrypted sessions cannot be read without access to additional keys or exploits.

To decrypt TLS traffic, one of the following would generally be required:

1. Access to TLS Session Keys:

If you control the endpoint (such as the client device or server), you can configure it to export session keys. For example, a browser like Firefox or Chrome can be launched with an environment variable (SSLKEYLOGFILE) that saves session keys used in TLS connections. These keys can then be loaded into Wireshark to decrypt HTTPS traffic from that session. This method is only feasible in a lab environment where you control the client device.

2. Man-in-the-Middle (MitM) Attack with a Rogue Certificate:

A MitM attack can be attempted by setting up a rogue access point or proxy server and installing a fake root certificate on the victim's device. Tools like mitmproxy or Burp Suite can intercept and decrypt TLS traffic by re-signing it with the attacker’s certificate. However, this method only works if the client accepts the rogue certificate—usually requiring physical access or social engineering—and is blocked by techniques like certificate pinning.

3. Private Key Access (Server-side)

If the attacker has access to the private key of the web server (which should be secret), and if static RSA key exchange is used (which is now deprecated), then TLS traffic can be decrypted. However, modern TLS 1.2+ sessions typically use ephemeral keys (DHE/ECDHE), which generate a unique session key per connection and do not allow decryption even with the server's private key.

4. TLS Downgrade or Exploit

In rare cases, outdated or misconfigured servers may still support weak ciphers or older versions of TLS (e.g., TLS 1.0/1.1), which are vulnerable to downgrade or cryptographic attacks. Exploiting these

vulnerabilities is difficult, unreliable, and typically only relevant in enterprise red-team assessments of academic research.

TLS Decryption in Wireshark

In this project, TLS traffic was captured during the same session as the WPA2 handshake, using Wireshark on the wlan0mon interface. After successfully recording the 4-way EAPOL handshake from the Dell device, I continued monitoring the channel to capture the device's ongoing network activity. While the capture was still running, the Dell accessed secure websites, resulting in multiple encrypted TLS sessions being recorded in the same file (hack 1.cap). These sessions appear as:

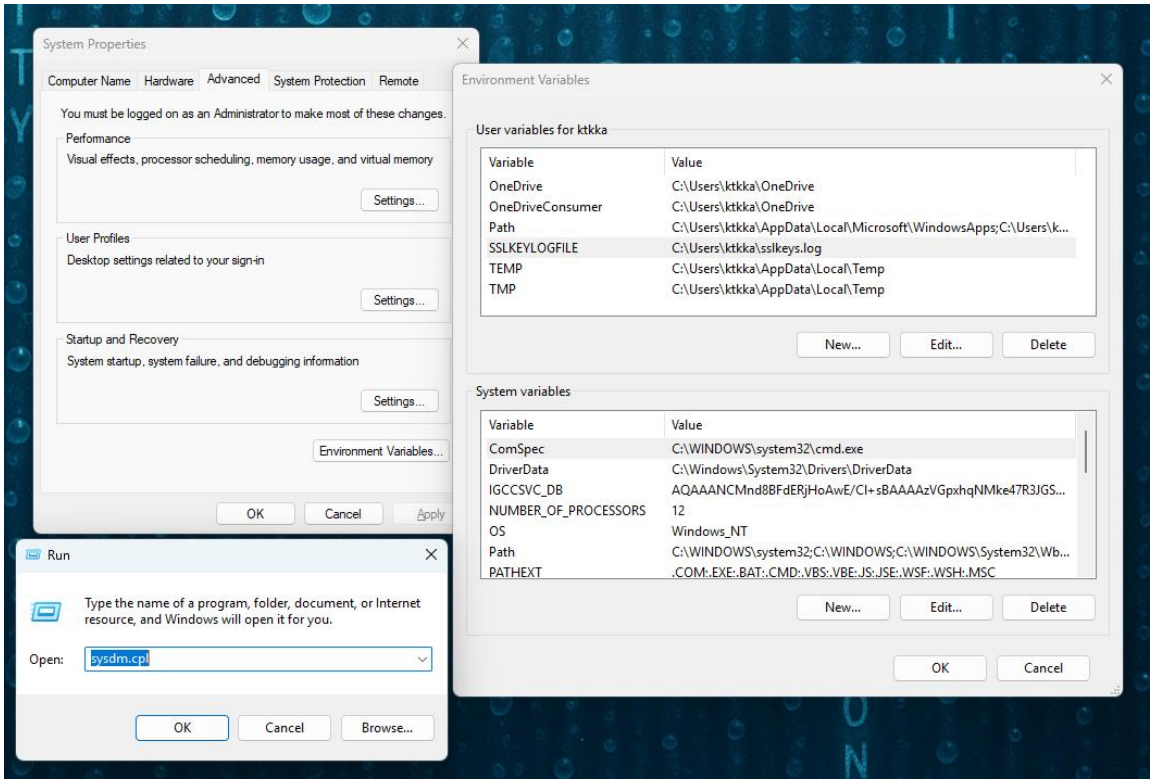
- TCP connections over port 443
- TLSv1.2 and TLSv1.3 handshakes
- Encrypted application-layer data typical of HTTPS

This confirmed that the capture contained not only the WPA2 handshake (needed for decrypting 802.11 traffic) but also full TLS sessions, complete with metadata such as source/destination IPs, SNI values, and encrypted payload lengths. However, the application data itself remained encrypted due to the absence of TLS session keys—highlighting the layered nature of modern encryption.

TLS Decryption with Session Keys

After capturing the WPA2 handshake and isolating Dell's traffic through earlier steps, the next phase involved setting up Firefox on the target device to export TLS session keys. This allowed for full decryption of HTTPS traffic within Wireshark.

1. **Enable TLS Session Key Logging on Target Device**
    - Create environment variable *SSLKEYLOGFILE* on Dell.
    - Set value to output TLS session keys to *C:\Users\YOUR_USERNAME\Desktop\sslkeys.log.*
    - Make sure Firefox was closed before setting the variable.
    - Relaunched Firefox so session keys would be captured.



2. Generate Browsing Activity

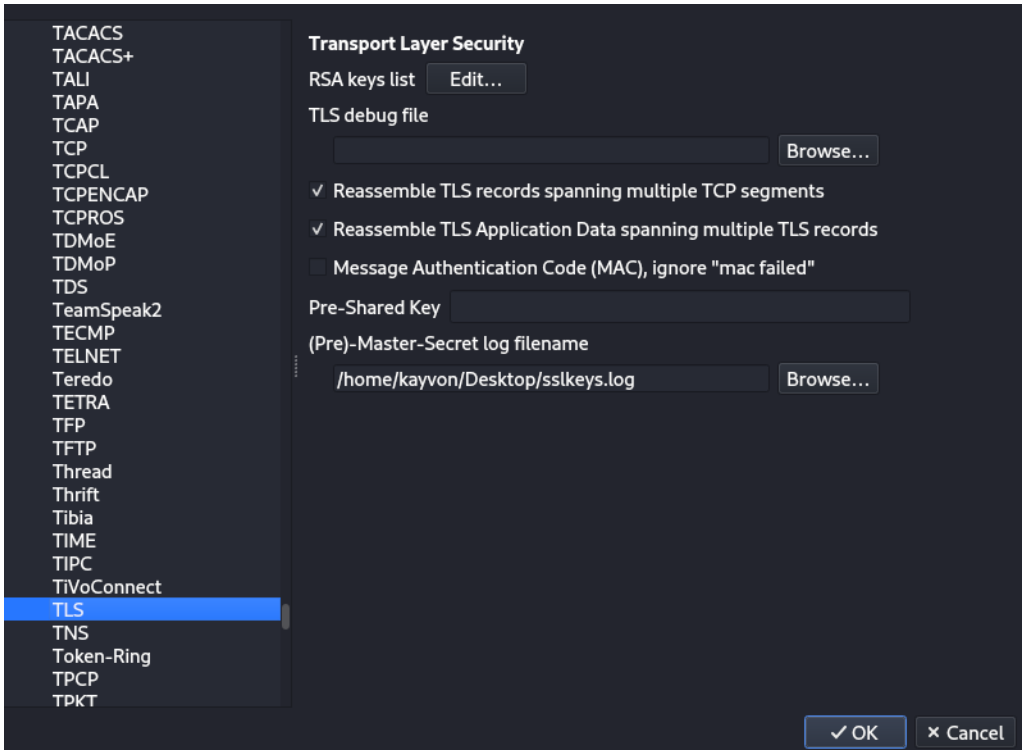
- ➢ Open Firefox on Dell
- ➢ Browse multiple HTTPS website (such as YouTube)
- ➢ Capture traffic in parallel on Kali Linux

3. Transfer TLS Session Key File from Target to Kali

- ➢ Use a usb flash drive to retrieve the sslkeys.log file from target device (Dell)
- ➢ Transfer the file onto Kali Linux for use in Wireshark analysis

4. Configure Wireshark to decrypt HTTPS Traffic

- ➢ Open the captured .cap file in Wireshark
- ➢ Navigate to Edit/Preferences/Protocols/TlS
- ➢ Set the (Pre)-Master-Secret log filename field to point to the transferred *sslkeys.log*



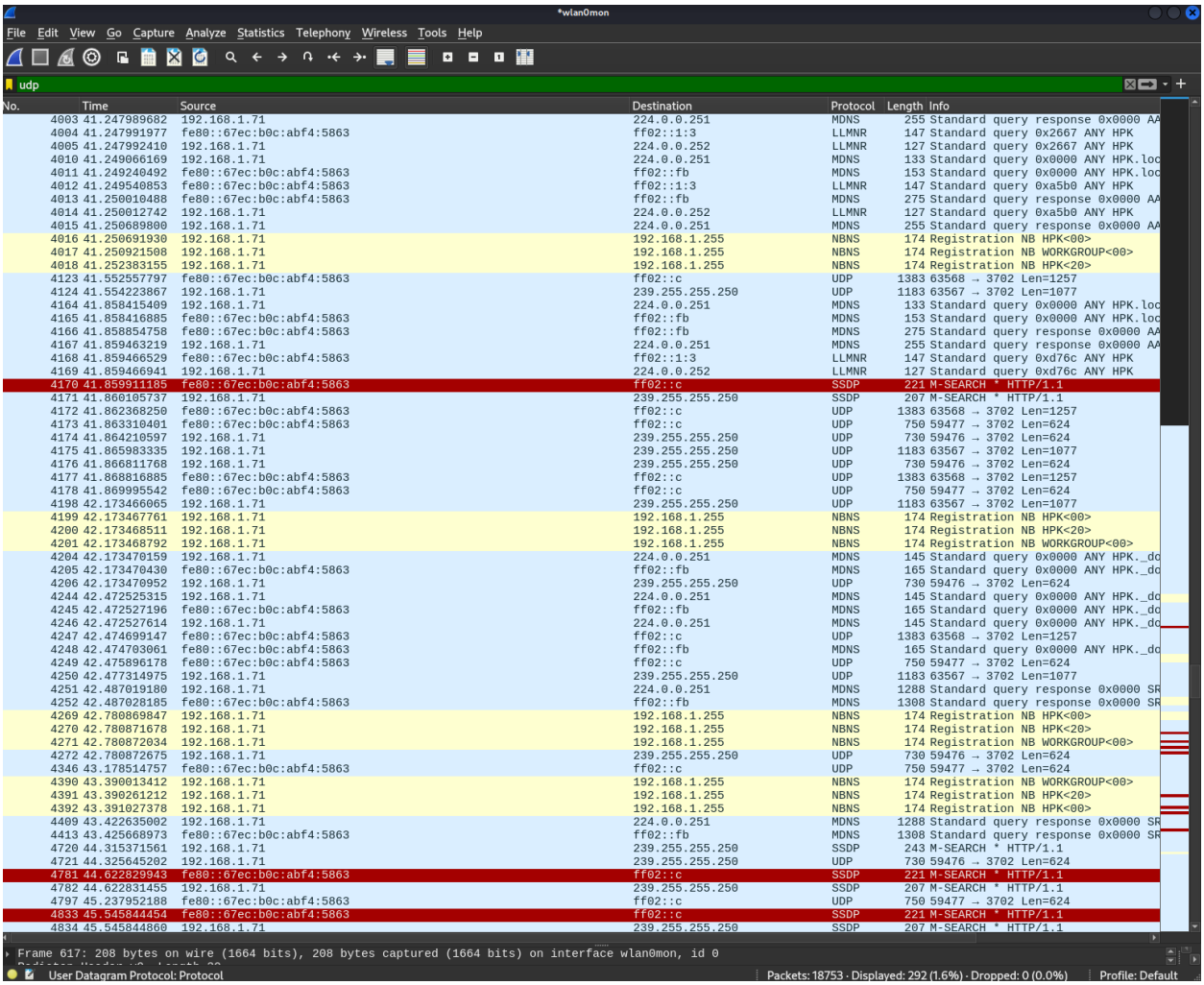
5. Analyze and Identify Decrypted Web Activity

- Apply Wireshark filter to locate Dell's browsing activity
- *'tls.handshake.type == 1'* for Client Hello packets
- *'dns'* to locate domain queries

```
DNS     1… Standard query response 0xbdca A gateway.instagram.com CNAME dgw.c10r.facebook.co
DNS     1… Standard query 0xc50b A pub-csm-usce-03-t.trouter.skype.com
DNS     1… Standard query 0x3ff4 SRV _ldap._tcp.dc._msdcs.attlocal.net
DNS     4… Standard query response 0xc50b A pub-csm-usce-03-t.trouter.skype.com CNAME partit
DNS     1… Standard query 0x7829 A us04zpns.zoom.us
DNS     1… Standard query response 0x7829 A us04zpns.zoom.us CNAME zpns.zoom.us CNAME zpns-b
DNS     1… Standard query 0x587c A teams.live.com
DNS     1… Standard query 0xd183 HTTPS teams.live.com
DNS     1… Standard query 0xe571 A ssl.gstatic.com
DNS     1… Standard query response 0xe571 A ssl.gstatic.com A 142.251.46.163
DNS     1… Standard query 0x14d7 A ssl.gstatic.com
DNS     1… Standard query 0xe4c8 AAAA ssl.gstatic.com
DNS     1… Standard query 0xcb76 A signaler-pa.clients6.google.com
DNS     1… Standard query 0x7720 A chat.google.com
DNS     1… Standard query response 0x4356 A tv.youtube.com A 142.251.32.46
DNS     1… Standard query 0x73f8 SRV _ldap._tcp.dc._msdcs.attlocal.net
DNS     2… Standard query response 0x583d A ic3.events.data.microsoft.com CNAME teams-events
DNS     2… Standard query response 0x0169 AAAA ic3.events.data.microsoft.com CNAME teams-eve
DNS     1… Standard query 0x3cfe A attlocal.net
DNS     1… Standard query 0x01b1 A www.msftconnecttest.com
DNS     1… Standard query response 0x3cfe A attlocal.net
DNS     1… Standard query 0x4c96 A www.facebook.com
DNS     1… Standard query 0x494f A ssl.gstatic.com
DNS     1… Standard query 0xf6f1 A chat.google.com
DNS     1… Standard query 0xf564 A tv.youtube.com
```

During the browsing session, DNS queries from the Dell device were recorded, showing attempts to resolve domains linked to major websites. The captured traffic included queries to subdomains of Facebook, Google, and YouTube, offering strong evidence of real-world web activity. These DNS lookups, collected while monitoring wireless traffic, confirm that the device actively interacted with popular services during the analysis period.

Wireshark Capture of Local Network Broadcast Traffic Including LLMNR, NBNS, and SSDP Protocols



This capture contains various types of background network traffic that occur as part of normal LAN device discovery and name resolution. The traffic is primarily UDP-based and includes the following protocols:

SSDP (Simple Service Discovery Protocol)

- Uses UDP port 1900.

- Part of the Universal Plug and Play (UPnP) suite.
- Used for discovering and advertising devices and services on the local network.
- Seen in the capture as HTTP-style SEARCH and NOTIFY messages.

LLMNR (Link-Local Multicast Name Resolution)

- Uses UDP port 5355.
- Allows devices to resolve hostnames on the same local subnet without a DNS server.
- Commonly used by Windows systems for fallback name resolution.

NBNS (NetBIOS Name Service)

- Uses UDP port 137.
- An older name resolution protocol used in Windows networks.
- Broadcast-based queries for device names and addresses.

mDNS (Multicast DNS)

- Uses UDP port 5353.
- Allows devices to perform DNS-like queries on the local network without a DNS server.
- Common in Apple devices and smart home/IoT environments.

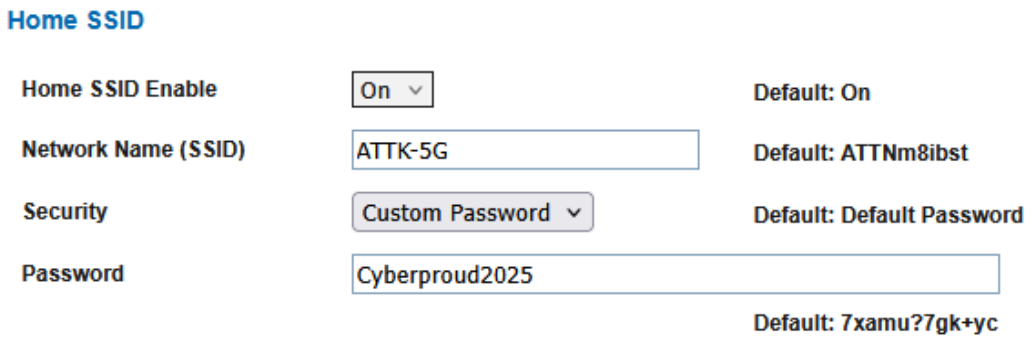
UDP (User Datagram Protocol)

- The transport layer protocol is used by all the above.
- Lightweight and suitable for broadcast and multicast traffic.
- Enables fast service and device discovery in local networks.
- These protocols generate traffic even when users are idle, as devices routinely announce their presence and listen for services. Capturing and analyzing this traffic helps map out active devices, services, and behaviors on a network. It also highlights how certain name resolution services (like LLMNR and NBNS) can be leveraged by attackers.

Decrypting the WPA2 traffic allowed me to view device activity, network metadata, and capture full TCP and TLS sessions. However, the actual contents of HTTPS traffic remained protected by TLS encryption, preventing full visibility into user data. By using session keys in a controlled environment, I was able to demonstrate how TLS traffic can be decrypted when the right keys are available.

Passkey Cracking Use Cases

To test stronger, more realistic passwords, I replaced the original weak WPA2 password with a tougher one: CyberProud2025 inside the router management console. This new password mixed uppercase and lowercase letters with numbers, making it much harder to crack using a basic dictionary attack.

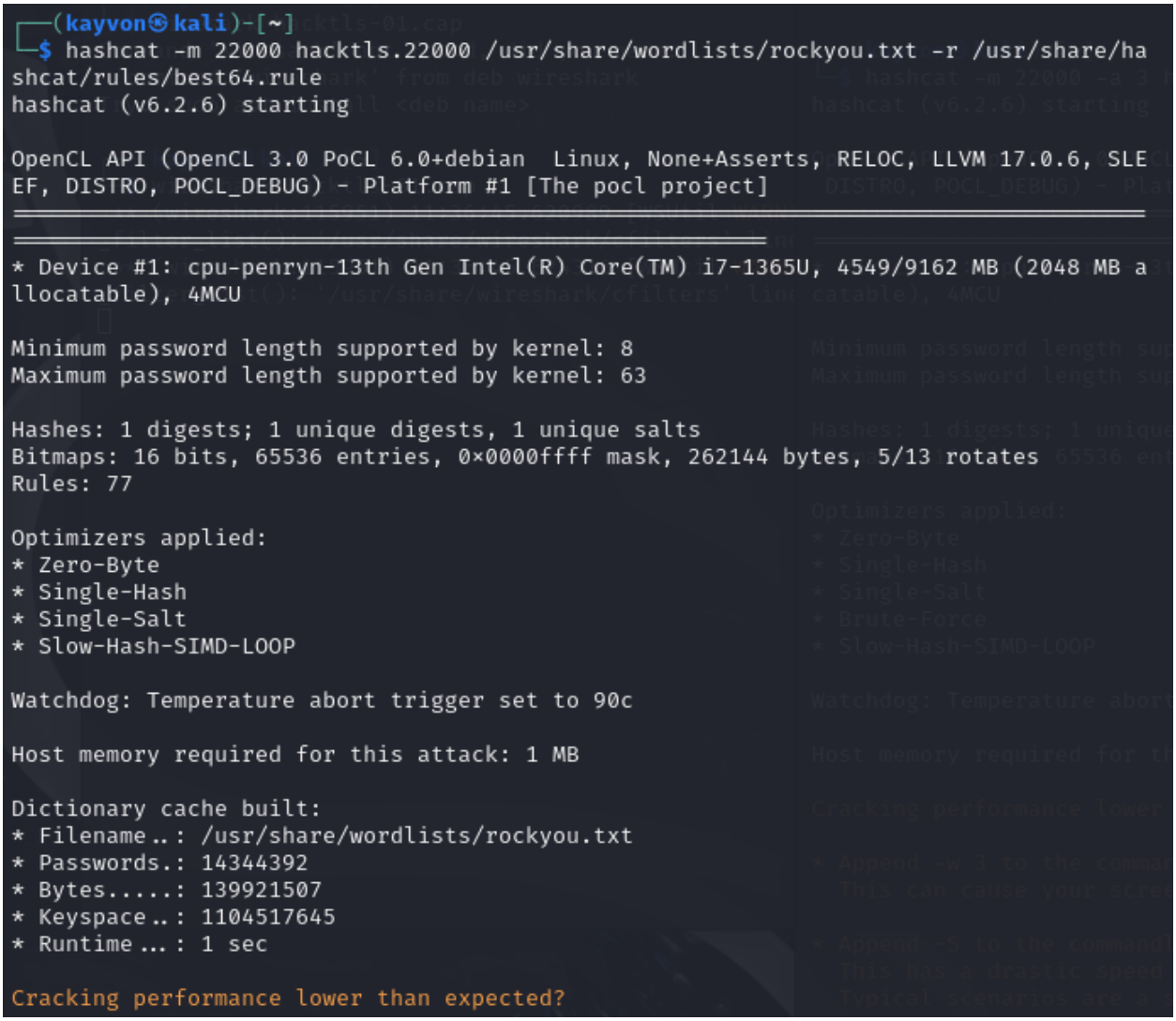


For my first cracking attempt, I used Aircrack-ng with the rockyou.txt wordlist, which is a go-to collection of commonly used passwords. The tool ran through over 14 million possible keys at a rate of around 2,000 keys per second. Despite letting it run for almost three hours, the result was clear: KEY NOT FOUND. The screenshot shows that the estimated time left ballooned into the absurd—over 113 million days—which just underscores how inefficient a dictionary attack becomes when the password isn't in the list. This test confirmed that CyberProud2025 wasn't part of rockyou.txt and that even moderately strong passwords can be very effective against standard, low-effort attacks.

To take things further, I used Hashcat, a more advanced and flexible password cracking tool. Hashcat supports multiple attack types including dictionary, rule-based, brute-force, and hybrid. It can run on both CPUs and GPUs, and it's commonly used for WPA2 cracking when Aircrack-ng isn't enough. For this attempt, I ran a rule-based attack using the command below, which combines the rockyou.txt wordlist with the best64.rule:

- hashcat -m 22000 hacktls.22000 /usr/share/wordlists/rockyou.txt -r /usr/share/hashcat/rules/best64.rule

This setup applies common password mutations like capitalizing words, appending numbers, or replacing letters—designed to guess more realistic passwords based on patterns people often use.



Hashcat quickly processed over 14 million password guesses in just 1 second during the rule-based attack. The output shows that the session completed successfully but no password was cracked. This means that none of the entries in the rockyou.txt wordlist, even after applying the best64.rule mutations, matched the captured WPA2 handshake. So far, no successful results have been obtained. This highlights the difficulty of cracking a strong password with basic wordlists and simple rule-based modifications. If a password is recovered later, the findings will be updated.

In this experiment, Hashcat's performance was constrained by the absence of GPU acceleration. Operating solely on a CPU, even rapid rule-based attacks were unable to recover the password. While Hashcat quickly processed over 14 million guesses during the session, the lack of a graphics card made more intensive attacks, such as brute-force, inefficient and impractical. A GPU like the Nvidia RTX 4090 would significantly boost attack speed, handling hundreds of thousands of guesses per second.

Ultimately, this highlights that in real-world scenarios, both strong password practices and hardware resource constraints serve as critical defenses against unauthorized wireless exploitation. [4]

Recommendations

As home networks increasingly become the backbone of daily life - supporting remote work, smart home devices, education, and sensitive financial transactions - it is critical to ensure they are properly secured against evolving cyber threats. Weak wireless configurations can leave users vulnerable to risks such as unauthorized access, data interceptions, and targeted attacks.

The following recommendations provide practical, high-impact steps to strengthen the security of home Wifi environments. These measures are designed to minimize attack surfaces, reinforce authentication mechanisms, and enhance visibility and control across the network, creating a resilient foundation for both personal and professional use.

Home Network Recommendations

[4] Rudisail, B. (2022, November). *Tackling GPU-enabled password cracking*. Spiceworks. https://www.spiceworks.com/it-security/identity-access-management/articles/tackling-gpu-enabled-password-cracking/

- **Use Strong and Unique Wi-Fi Passwords**

  Weak or commonly used passwords remain one of the most exploited vulnerabilities in home networks.

  - ➢ Avoid dictionary words or predictable phrases.
  - ➢ Use a long, complex passphrase that includes upper and lowercase letters, numbers, and symbols.
  - ➢ This helps defend against both brute-force and dictionary-based attacks.

- **Upgrade to WPA3 if Available**

  WPA3 introduces stronger encryption methods and eliminates the vulnerability of capturing a handshake for offline cracking. Unlike WPA2, WPA3 uses Simultaneous Authentication of Equals (SAE) which is resistant to dictionary attacks.

  - ➢ Introduces Simultaneous Authentication of Equals (SAE), which prevents offline attacks like the one demonstrated in this project.
  - ➢ WPA3 support should be confirmed for both the router and connected devices.

- **Disable WPS (Wi-Fi Protected Setup)**

  WPS is known to be vulnerable to brute-force attacks. Even if WPA2 is properly configured, WPS can offer an easier point of entry for attackers.

  - ➢ It allows attackers to brute-force the setup PIN and gain access even if the main Wi-Fi password is strong.
  - ➢ Disabling WPS through router settings reduces this risk.

- **Segment Your Network with VLANs or Guest Wi-Fi**

  Use a guest network for devices that do not need access to your main network resources.

  - ➢ Minimizes the risk of lateral movement in case one device is compromised.
  - ➢ For advanced users, VLAN segmentation can offer even stronger isolation.

- **Enable Multi-Factor Authentication (2FA) on Routers**

  - ➢ Critical for preventing unauthorized configuration changes.
  - ➢ Protects access even if credentials are compromised.

- Enable automatic firmware updates or manually check them regularly.

- **Use MAC Address Filtering and Hide SSID**
  - MAC filtering allows only pre-approved devices to connect.
  - Hiding the SSID reduces visibility to attackers but should be relied upon as a primary defense.
  - These methods should be paired with WPA2/WPA3 encryption and strong passwords.

- **Keep Router Firmware Updated**
  - Manufacturers often release security patches to fix known vulnerabilities.
  - Keeping the router's firmware up to date ensures protection against newly discovered exploits.

- **Regularly Monitor Network Traffic and Devices**
  - Periodically check the list of connected devices to detect unknown or suspicious devices.
  - Tools like Wireshark, router logs, or even mobile apps provided by router manufacturers can help maintain visibility.

Recommended GPUs for Password Cracking

Top 2 Internal Graphics Cards:

- NVIDIA RTX 4090 (link)
  - currently the highest-performing GPU available for password cracking.
  - Extremely fast, capable of handling massive numbers of hash guesses per second.
- AMD Radeon RX 7900 XTX (link)
  - AMD's flagship GPU for performance-heavy tasks delivering high cracking speeds.
  - Generally cheaper than the RTX 4090 while offering strong performance.
  - Consumes less power compared to NVIDIA's top-end cards in many tasks.

Top 2 External GPU (eGPU) Solutions for laptops:

- Razer Core X Chroma (link)
  - A popular and reliable eGPU enclosure that supports large, high-power graphics cards like the RTX 4090 or 4080.
  - Connects via Thunderbolt 3 or 4, offering strong performance while maintaining laptop portability.
- Sonnet eGFX Breakway Box 750ex (link)
  - Provides excellent thermal management and power delivery, suitable for high-end GPUs.
  - Offers a slightly more affordable option compared to the Razer Core, with stable and efficient performance.

Note:

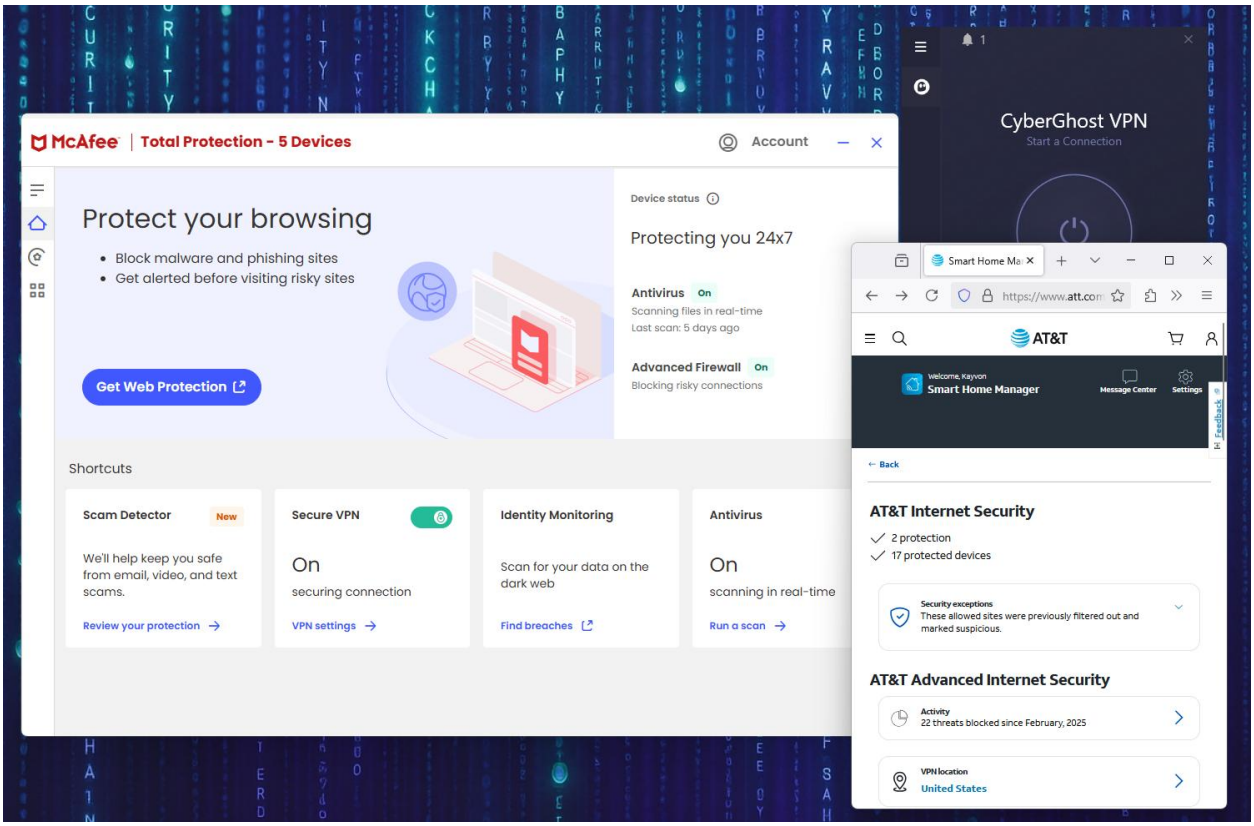
GPUs provide a major speed advantage over CPUs in password cracking by processing many computations at once. Using a GPU significantly reduces the time required for attacks like brute-force or rule-based methods.

Conclusion

This project simulated a real-world Wi-Fi attack using Kali Linux and tools like airmon-ng, airodump-ng, aircrack-ng, and Hashcat to capture and crack a WPA2 handshake. After collecting the handshake with a USB Wi-Fi adapter, I launched an offline dictionary attack with rockyou.txt. Once the password was recovered, I used Wireshark to decrypt wireless traffic. With access to the session keys, I was able to decrypt TLS sessions and view HTTPS traffic in full, showing how a weak Wi-Fi password can compromise even encrypted communications.

Through this process, it became clear that while CPU-based cracking is enough for weak passwords, stronger passphrases resist attacks without powerful hardware. Testing with Hashcat showed that using a GPU—rather than a CPU—would dramatically speed up password recovery, turning multi-year cracks into minutes. This highlighted how critical hardware is in real-world attack scenarios.

Applying these lessons, I strengthened my home network: I upgraded my WPA2 passphrase, used AT&T's Smart Home Manager for monitoring, isolated smart devices on a guest network, and secured public Wi-Fi use with VPNs. These steps create a layered defense, reducing the risks exposed in this project.



References:

Bombal, D. (2022, March). *Cracking WiFi WPA2 Handshake* [Video]. YouTube, https://www.youtube.com/watch?v=WfYxrLaqlN8

Burns, W. J. (n.d.). *Common password list (rockyou.txt)* [Dataset]. Kaggle. https://www.kaggle.com/datasets/wjburns/common-password-list-rockyoutxt

DarkAudax. (2010, March). *Cracking WPA/WPA2. Aircrack-ng Documentation*, https://www.aircrack-ng.org/doku.php?id=cracking_wpa

Ultimate Systems Blog. (2024, October). *External GPU vs. Internal GPU*. Ultimate Systems Blog. https://blog.usro.net/2024/10/external-gpu-vs-internal-gpu/

HackHunt. (2024, January). *How to Enable Monitor Mode on Kali Linux*. [Video] YouTube, https://www.youtube.com/watch?v=zdCGMIZbRr8&t=58s.

Inno TechTips. (2024, February). *How to sniff/capture WiFi traffic using Kali Linux* [Video]. YouTube, https://www.youtube.com/watch?v=Nut5yBmmMOY

ITPro. (2022, October). *Nvidia's RTX 4090 is a powerful password-cracking tool.* ITPro. https://www.itpro.com/hardware/components/369322/nvidias-rtx-4090-is-a-powerful-password-cracking-tool

Kaspersky. (n.d.). *What is a dictionary attack?* Kaspersky. https://www.kaspersky.com/resource-center/definitions/what-is-a-dictionary-attack

Offensive Security. (n.d.). *aircrack-ng*. Kali Linux Tools. https://www.kali.org/tools/aircrack-ng/

Rudisail, B. (2022, November). *Tackling GPU-enabled password cracking*. Spiceworks. https://www.spiceworks.com/it-security/identity-access-management/articles/tackling-gpu-enabled-password-cracking/

St. Cloud State University. (2024). Study on Password Hashing with GPU https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1032&context=msia_etds

Wright, G. (2024, February). *Dictionary attack*. TechTarget. https://www.techtarget.com/searchsecurity/definition/dictionary-attack

Supporting Materials:

Video Demonstration: Wireshark Network Exploitation: Cracking WPA2 & Decrypting Traffic

Supplementary Slides: Wireless Network Exploitation Project Slides.ppt