Cryptology - 695.641

John Bannon

F96B14

WPA2 and Dictionary Attacks: A practical attack against modern Wi-Fi networks

Introduction

Background

With the introduction of wireless networks, the need for a protocol that could secure communication and restrict access emerged. In 1997, WEP (Wired Equivalent Privacy) was introduced as a part of the original IEEE 802.11 standard[1]. While somewhat effective, it suffered from weak encryption and key security due to it's use of RC4 and key reuse policies[2]. It was quickly superseded by WEP2, which improved upon the poor key management and weak Initialization Vectors. However, since WEP was still based on the weak RC4 algorithm, it never saw official adoption.

In 2003, WPA (Wi-Fi Protected Access), developed by Wi-Fi Alliance, was introduced with 802.11g. It was intended to be an intermediate replacement for WEP while the full 802.11 standard developed[3]. It improved over WEP by moving away from RC4 and implementing TKIP (Temporal Key Integrity Protocol), which uses 64 or 128-bit encryption on a per-packet basis. Notably, it also introduced the 4-way handshake, a process which ensures both the client and AP (access point) have the correct Pre-Shared Key without requiring having to transmit it[3]. Despite other improvements like the inclusion of a Message Integrity Check, WPA's security was compromised in 2009 when Toshihiro Ohigashi and Masakatu Morii iterated on a previous attack strategy to be able to decrypt short packets in any WPA network[4].

WPA2 replaced WPA in 2004 with 802.11i. It replaces TKIP with AES-128-CCMP (Advanced Encryption Standard Counter Mode CBC-Mac Protocol)[3]. While WPA2 is considered secure and is still deployed in countless APs today, there are known vulnerabilities such as the KRACK attack and hash dictionary attacks. These have been patched for the most part through modern AP configurations, however.

Objective

This paper details the implementation of a practical attack on WPA. More specifically, a successful recovery of the PSK (Pre-Shared Key) from a captured PCAP (Packet Capture) file. During the process of implementation, the author decided that due to the nature of the tools

and methods used, it would be preferable to focus on WPA2 networks. Since the proposed method of attack is applicable to WPA and WPA2, the results of the experiment are still relevant to WPA. This decision was made as to forgo the creation of a private WPA network for live, active WPA2 networks given time constraints.

Description

The implemented practical attack on WPA2 involves capturing a pcapng file from a network and brute forcing the hash contained within to obtain the PSK via a dictionary attack. A dictionary attack is the process of testing various potential passwords against a captured hash until the correct password is found. The Bash script accompanying this paper uses aircrack-ng, hcxdumptool, hcxpcapngtool, hcxhashtool, and hashcat as the main tools for performing preparation, capture, conversion, cleaning, and cracking, respectively. This section will describe the hardware and operating system used, why the attack itself works, what functions and roles each tool serves in the attack, as well as why they were chosen over alternatives.

Hardware

All scans were performed using a Techkey AC-1200 USB Wi-Fi Adapter. This adapter was chosen over alternatives because of its USB 3.0 and 802.11ac capabilities. Meaning, it can scan the 2.4GHz and 5GHz band with high gain. The adapter was also selected because of its use of Realtek Wi-Fi chipset and drivers, which are known to be compatible with hcxdumptool[6]. Specifically, the RTL88x2B driver was used for this implementation[7].

Development, testing, and subsequent scans were performed on an Arch Linux installation with kernel version 6.19.10-arch1-1. All hash cracking using Hashcat was computed with an NVIDIA RTX 4090. While the Bash script should be compatible with any Linux distribution that can install the required dependencies, please note that this specific script will only work on Linux due to the inclusion of hcxdumptool. For Windows or MacOS alternatives, the aircrack-ng suite is a suitable replacement for hcxdumptool, at the cost of some features and conveniences.

The 4-Way Handshake

To understand why capturing a part of a handshake or a PMKID is enough information for a dictionary attack, the 4-Way Handshake must be examined (See Fig. 1). First, both the client and the AP calculate a Pairwise Master Key (PMK). This key acts as the final check for later calculations in the handshake; if the client calculates the wrong PMK (by inputting the wrong password), subsequent calculations will be incorrect and cause the AP to reject the client**.** According to Adam Govier, "The PMK is created using the Password-Based Key Derivation Function #2 (PBKDF2), with the SHA1 hashing function used with HMAC as the message authentication code: $PMK = PBKDF2(HMAC\text{-}SHA1, PSK, SSID, 4096, 256)$. HMAC-SHA1 is the Pseudo Random Function used, whilst 4096 iterations of this function are used to create the 256 bit PMK. The SSID is used as a salt for the resulting key, and of course the PSK (passphrase in this instance) is used as the basis for this entire process"[13].

After both parties generate the PMK, the AP sends an ANonce, a random number, to the client. The client then generates it's own random number, the SNonce, and generates the Pairwise Transient Key (PTK) from both Nonces and MAC addresses[13]. It also generates a Message Integrity Check (MIC) from a portion of the PTK. The client sends it's SNonce and MIC to the AP. The AP calculates the PTK and generates the MIC. If the calculated MIC and the received MIC from the client are the same, then the client has the correct password and is allowed to connect[13].

The catch is that the first 2 messages of the handshake contain enough information to perform an attack. Since both Nonces, MAC addresses, and MICs are sent unencrypted, an attacker can capture that information and guess a password, derive a potential PMK and PTK, and calculate a MIC. If the calculated MIC is the same as the one captured from the handshake, the password is found. While this can work, the capture of the first 3 messages is preferable. The second message contains the client's calculated MIC. So, if the client used the wrong password, the attacker is working with junk data. Since the 3rd message contains the correct MIC calculated from the AP, the attacker knows that if they ever get a hit, it will be the correct password.

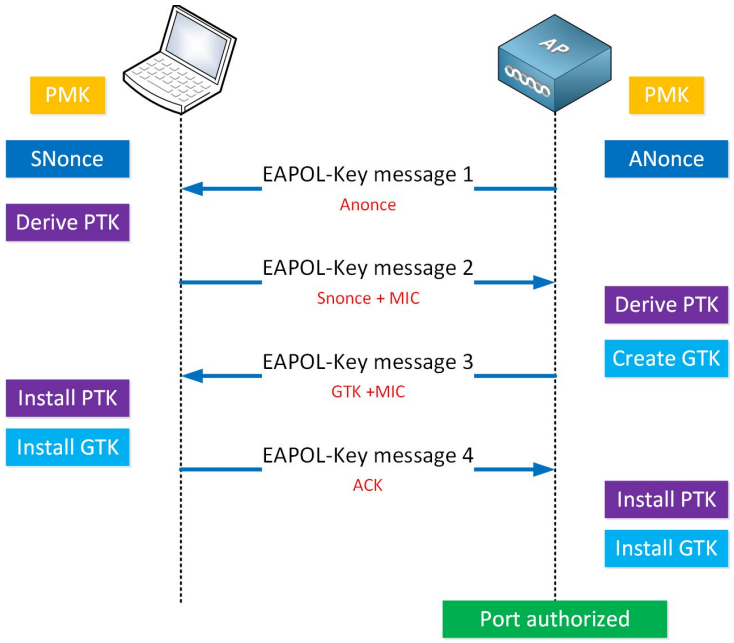


Figure 1: The 4-Way Handshake between a client and AP[12].

Capturing a PMKID allows for an attack in a similar fashion. A PMKID (Pairwise Master Key Identifier) is a hash that the AP will send to previously connected clients, forgoing the full authentication process. This key is sent in the first message of the handshake and is derived from the PMK: $PMKID = HMAC\text{-}SHA1\text{-}128(PMK, "PMK\ NAME" \ |MAC_{AP}| \ MAC_{STA})$[14]. Since both MAC addresses are known, the only unknown is the PMK, or the password. So, an attacker can perform dictionary attacks to repeatedly calculate PMKIDs until the hashes match.

Aircrack-ng

Aircrack-ng is a complete suite of tools to assess WiFi network security[5]. It contains all the tools necessary to perform a capture and dictionary attack against WPA2. For the purposes of this implementation, however, only airmon-ng is utilized. Airmon-ng, in conjunction with some system commands, are used to prepare the environment for capturing. Specifically, all processes that control or maintain the network (like NetworkManager and wpa_supplicant) are killed. Airmon-ng is used to double check if any processes were restarted and promptly kills them. This double verification is necessary as throughout testing, the author found that simply telling important network processes to stop was not enough to ensure they would not try to hijack the interface mid-scan.

Hcxdumptool

Hcxdumptool is a tool to capture packets from WLAN devices and to discover potential weak points within own WiFi networks by running layer 2 attacks against the WPA protocol[6]. Part of a larger collections of tools known as hcxtools, hcxdumptool is the heart of the attack. The tool was chosen over the more well-known Aircrack-ng due to it's modern capabilities such as client-less capture, raw sockets, and capture automation. The tool can request and capture a PMKID (Pairwise Master Key Identifier) during an initial association request with an AP, giving all the information needed to attempt an attack without a client needing to be connected to the AP[6]. It also actively sends deauthorization packets while scanning for networks, kicking clients off the AP to force a 4-way handshake that can be captured[6]. With aircrack-ng, a second terminal window is required to run aireplay-ng and send de-auth packets, which is troublesome to automate. Finally, hcxdumptool's use of Berkeley Packet Filters and raw sockets in kernel space that bypass the Linux network stack prevents junk traffic from reaching the CPU, reducing overhead[6]. By comparison, aircrack-ng processes packets in user space which can cause packet drops.

For this implementation, hcxdumptool is first used to prepare the interface by setting it to monitor mode. In monitor mode, a WiFi interface disconnects from all networks and accepts every packet on a chosen channel. This allows the tool to see important data, management and control frames that are usually invisible to Wi-Fi cards. This is actually a passive feature of the tool that happens when scanning is initiated. Then, after a channel(s) is chosen, active scanning begins. Upon detecting an AP, hcxdumptool will attempt an association request. If the AP supports PMKID caching and the setting is enabled, it will send its PMKID with the association response and the tool will automatically stop scanning. This is an ideal scenario since information to attempt an attack can be obtained anonymously and client-less. If a PMKID is not received, hcxdumptool will then send de-auth packets to clients connected to the AP. If a client disconnects and then reconnects, the 4-way handshake can be captured and an attack attempt can be made. Finally, if no de-auth packets reach a client, hcxdumptool will passively listen to see if a client naturally connects and will attempt to capture the handshake. If no PMKIDs are received, the tool will scan for 10 minutes before terminating.

Hcxpcapngtool

Another tool from hcxtools, hcxpcapngtool converts the raw packets, containing PMKIDs or handshakes, to hash files that can be read by Hashcat. Specifically, it converts .pcapng files to .hc22000. Additionally, the tool marks hashes that may require nonce-error correction due to higher-latency 5GHz captures. One large advantage of using this tool in conjunction with hcxdumptool is the ability to use the .hc22000 format. The format was specifically created by Hashcat developers for use with the hcxtools suite. By combining PMKIDs and EAPOL message pairs into a single file, the key derivation function (PBKDF2) can be efficiently reused which saves GPU cycles[8].

Hcxhashtool

This tool complements hcxpcapngtool by further refining the hash files it outputs. It can filter and output hashes in various configurations including: ESSID, MAC-AP, MAC-CLIENT, VENDOR-AP, VENDOR-CLIENT, EAPOL type, etc...[9] The feature pertinent to this implementation is the ability to filter out incomplete or corrupted handshakes that would otherwise waste GPU cycles. Furthermore, it's ability to group hashes by ESSID ensures that the expensive key derivation function only executes once per unique network name[9].

Hashcat

According to the hashcat wiki, "hashcat is the world's fastest and most advanced password recovery tool"[8]. With a plethora of flags, configurations, and hash modes to choose from, virtually any type of established hashing algorithm can be attacked. For this implementation, the most impactful feature is GPU-accelerated hash processing. With aircrack-ng being a CPU-based cracking software that relies on the older .cap format, it can only process thousands of hashes per second. Hashcat, utilizing CUDA or OpenCL, can process millions of hashes per second. This, along with the added benefits that come from the .hc22000 and .pcapng formats discussed above, make hashcat the obvious choice for this implementation.

Once a 'cleaned' hash file is generated by hcxhashtool, a dictionary attack is executed against the hashes by utilizing hashcat's dictionary attack mode. The attack is ran against several word lists, namely rockyou.txt and breach.txt[10][11]. If the word list(s) is exhausted, another dictionary attack with rules will execute. Rules apply specific modifiers to words in a word list which dramatically increases word variation and consequently increases the number of possible passwords to be tested. If a hash is cracked, the hash and corresponding password will be output to the terminal.

Analysis

The results of the project were a successful recovery of the PSK from a .pcapng file captured from a live WPA2 network using dictionary attacks(See fig 2, 3, 5). It should be noted, for legal reasons, that the only network attacked during the course of this project was the author's personal home network in which they own. PMKIDs and partial handshakes from other networks were captured during testing but, none of these hashes were acted upon and were subsequently deleted. Furthermore, because the PSK of the author's network is not weak, the PSK was injected into a word list to show Hashcat's functionality (See fig 3, 4). While this means that the PSK recovered was not obtained 'naturally', it still proves that the script created for this project is capable of detecting, capturing, and cracking hashes to obtain a PSK.

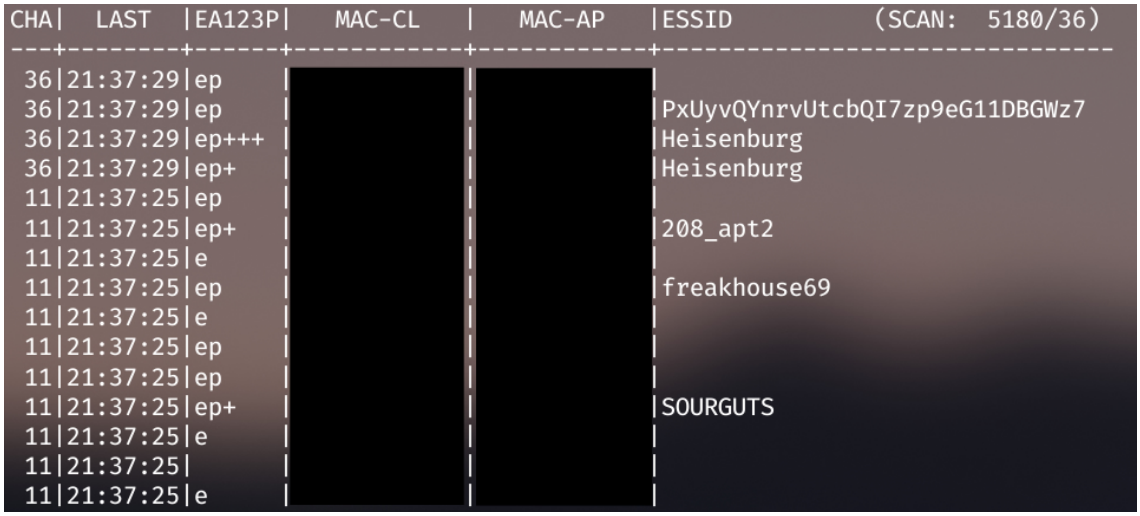


Figure 2: Active hcxdumptool scan with successful partial-handshake capture.

In figure 2, multiple networks are shown. MAC addresses have been obscured. Varying levels of the 4-way handshake have been captured. In the EA123P column, the levels of capture are shown. The 'E' column indicates whether a network is encrypted (e) or open (o)[9]. The 'A' column represents Authentication and Key Management[9]. A 'p' means the network uses a PSK[9]. The remaining columns indicate the exchanges which have been captures in a 4-way handshake[9]. For example, 'ep+++' indicates that messages 1, 2, and 3 of a handshake between an encrypted, password-protected network and an AP have been captured. It can be seen that hcxdumptool has captured 3 messages from Heisenburg (author's network).

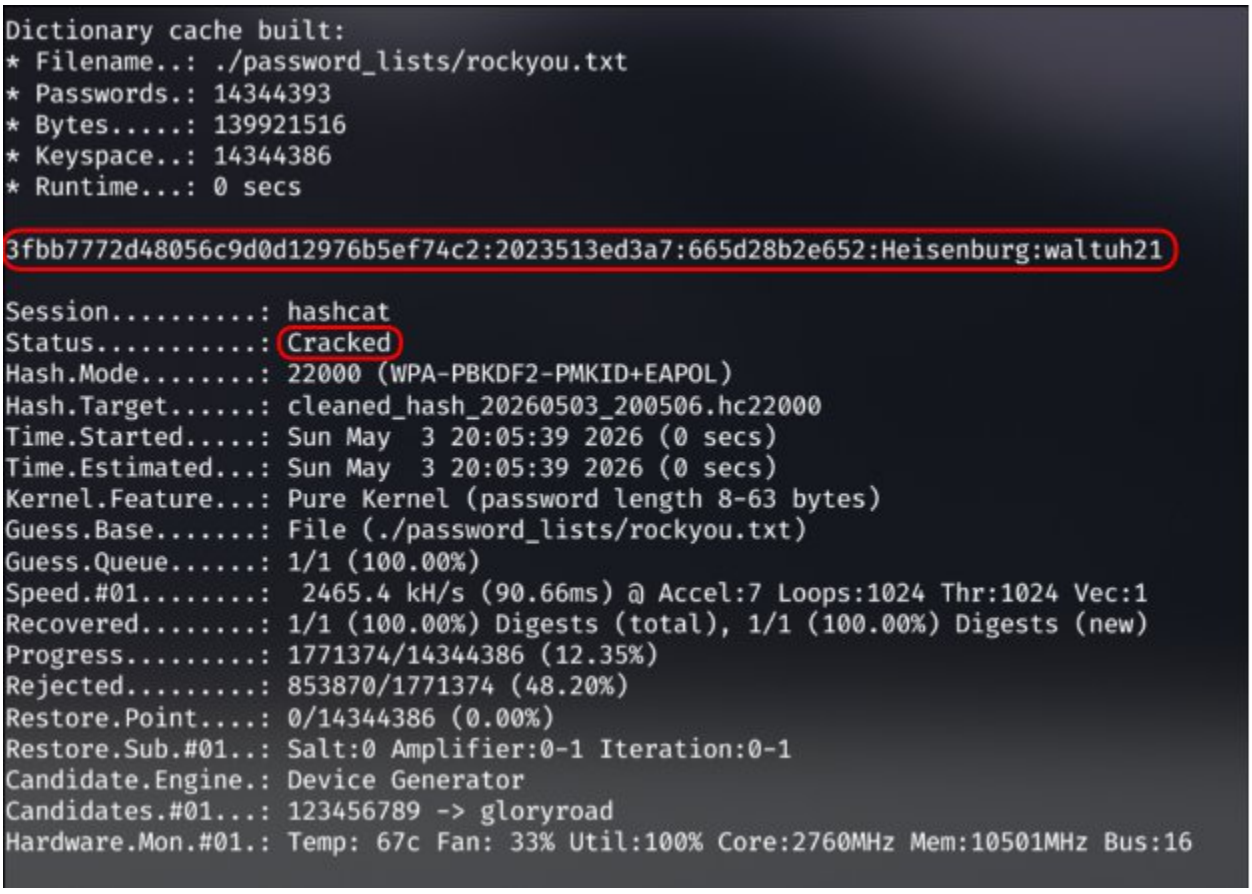


Figure 3: Successful hash crack using hashcat

Figure 3 shows a successful cracking of the hash obtained from Heisenburg. The highlighted portion along the top shows the actual hash, the network name, and the recovered password which can be seen as 'waltuh21'.

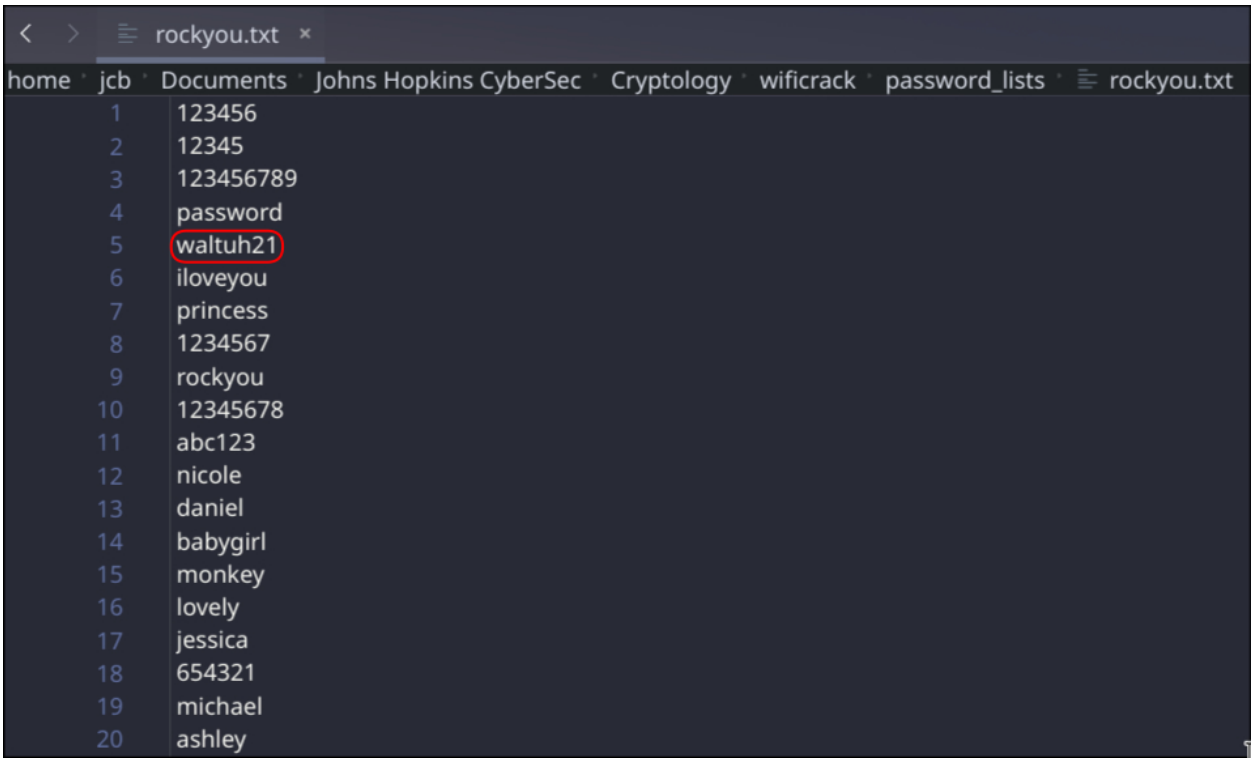
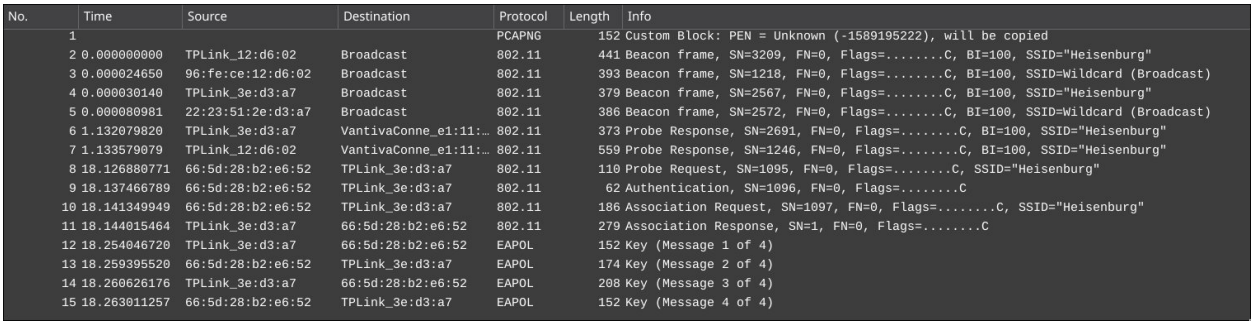


Figure 4: Author's PSK manually injected into rockyou.txt



No.	Time	Source	Destination	Protocol	Length	Info
1				PCAPNG	152	Custom Block: PEN = Unknown (-1589195222), will be copied
2	0.000000000	TPLink_12:d6:02	Broadcast	802.11	441	Beacon frame, SN=3209, FN=0, Flags=........C, BI=100, SSID="Heisenburg"
3	0.000024650	96:fe:ce:12:d6:02	Broadcast	802.11	393	Beacon frame, SN=1218, FN=0, Flags=........C, BI=100, SSID=Wildcard (Broadcast)
4	0.000030140	TPLink_3e:d3:a7	Broadcast	802.11	379	Beacon frame, SN=2567, FN=0, Flags=........C, BI=100, SSID="Heisenburg"
5	0.000080981	22:23:51:2e:d3:a7	Broadcast	802.11	386	Beacon frame, SN=2572, FN=0, Flags=........C, BI=100, SSID=Wildcard (Broadcast)
6	1.132079820	TPLink_3e:d3:a7	VantivaConne_e1:11:…	802.11	373	Probe Response, SN=2691, FN=0, Flags=........C, BI=100, SSID="Heisenburg"
7	1.133579079	TPLink_12:d6:02	VantivaConne_e1:11:…	802.11	559	Probe Response, SN=1246, FN=0, Flags=........C, BI=100, SSID="Heisenburg"
8	18.126880771	66:5d:28:b2:e6:52	TPLink_3e:d3:a7	802.11	110	Probe Request, SN=1095, FN=0, Flags=........C, SSID="Heisenburg"
9	18.137466789	66:5d:28:b2:e6:52	TPLink_3e:d3:a7	802.11	62	Authentication, SN=1096, FN=0, Flags=.......C
10	18.141349949	66:5d:28:b2:e6:52	TPLink_3e:d3:a7	802.11	186	Association Request, SN=1097, FN=0, Flags=........C, SSID="Heisenburg"
11	18.144015464	TPLink_3e:d3:a7	66:5d:28:b2:e6:52	802.11	279	Association Response, SN=1, FN=0, Flags=........C
12	18.254046720	TPLink_3e:d3:a7	66:5d:28:b2:e6:52	EAPOL	152	Key (Message 1 of 4)
13	18.259395520	66:5d:28:b2:e6:52	TPLink_3e:d3:a7	EAPOL	174	Key (Message 2 of 4)
14	18.260626176	TPLink_3e:d3:a7	66:5d:28:b2:e6:52	EAPOL	208	Key (Message 3 of 4)
15	18.263011257	66:5d:28:b2:e6:52	TPLink_3e:d3:a7	EAPOL	152	Key (Message 4 of 4)

Figure 5: A pcapng file showing a successful handshake capture with Heisenburg

Figure 5 shows a pcapng file that contains a full 4-way handshake between Heisenburg and a client that was captured. Hcxdumptool's Probe Request and Association Request can be seen in line 8 and 10, telling the AP to wake up and start a handshake. All 4 EAPOL handshake messages in lines 12-15 can be seen. This is where information like Nonces and MICs can be found to construct a hash.

Conclusion

This project aimed to implement a practical attack on WPA/WPA2. This involved recovery of a PSK from a captured PCAP file. Through the development of a Bash script that integrates several tools into a automated pipeline, this goal was achieved. The script successfully detected the author's network, interacted with the AP to start a handshake, captured relevant EAPOL messages, constructed a hash, cleaned broken or dirty information, and cracked the hash to reveal the PSK. To clarify, this script is for educational purposes only and the author does not condone it's use for malicious intent. Unauthorized access to wireless networks is illegal.

The conclusions of this project prove that while the barrier of entry to advanced security auditing tools are not as high as some may think, the experience needed to properly operate said tools should not be understated. Significant effort went into understanding, troubleshooting, and implementing these tools in a generalized way. Specifically, finding a suitable USB Wi-Fi adapter and driver that supported hcxdumptool's advanced capabilities.

The current implementation only uses two word lists to check passwords against. Given more time, further research into optimal word lists and hashcat rules may increase success rates with arbitrary hashes. Deeper research into the possible configurations of hcxdumptool and hashcat may allow for easier capturing of handshakes / PMKIDs.

References

[1] Wikimedia Foundation. (2026, April 21). Wired equivalent privacy. Wikipedia. https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy

[2] Parker, J. W. S. (2025, April 12). Weaknesses of WEP encryption. Cyberly. https://www.cyberly.org/en/what-are-the-weaknesses-of-wep-encryption/index.html

[3] Wikimedia Foundation. (2026a, April 14). Wi-Fi Protected Access. Wikipedia. https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access

[4] Ohigashi, T., & Morii, M. (n.d.). A practical message falsification attack on WPA. https://dl.packetstormsecurity.net/papers/wireless/A_Practical_Message_Falsification_Attack_On_WPA.pdf

[5] Documentation. Aircrack. (n.d.). https://www.aircrack-ng.org/index.html

[6] ZerBea. (n.d.). Zerbea/hcxdumptool: Small tool to capture packets from WLAN devices. GitHub. https://github.com/ZerBea/hcxdumptool

[7] RinCat. (n.d.). RINCAT/RTL88x2BU-Linux-Driver: Realtek RTL88x2BU WIFI USB driver for linux. GitHub. https://github.com/RinCat/RTL88x2BU-Linux-Driver

[8] Hashcat advanced password recovery. cracking_wpawpa2 [hashcat wiki]. (n.d.). https://hashcat.net/wiki/doku.php?id=cracking_wpawpa2

[9] hcxhashtool man. hcxhashtool(1) - hcxtools - Debian testing - Debian Manpages. (n.d.). https://manpages.debian.org/testing/hcxtools/hcxhashtool.1.en.html

[10] Danielmiessler. (n.d.). SecLists/passwords at master · danielmiessler/seclists. GitHub. https://github.com/danielmiessler/SecLists/tree/master/Passwords

[11] breach.txt. Weakpass.com. (n.d.). https://weakpass.com/wordlists/breach.txt

[12] Molenaar, R. (2026, April 14). WPA and WPA2 4-Way Handshake. NetworkLessons.com. https://networklessons.com/wireless/wpa-and-wpa2-4-way-handshake

[13] Govier, A. (n.d.). Understanding WPA/WPA2 pre-shared-key cracking | exploitr. Exploitr. https://exploitr.com/articles/understanding-wpa-wpa2-psk-cracking/

[14] Kumar, M. (n.d.). WIFI Security Analysis: PMKID attack method and hash cracking. Blog. https://blog.geekinstitute.org/2025/05/wifi-security-analysis-pmkid-attack-method-and-hash-cracking.html

[15] Santos, R. (2026, February 15). WIFI hacking 101: WPA/WPA2 Cracking, PMKID, and WPS (part 2). WiFi Hacking 101: WPA/WPA2 Cracking, PMKID, and WPS (Part 2). https://kayssel.substack.com/p/wifi-hacking-101-wpawpa2-cracking