Baltic J. Modern Computing, Vol. 13 (2025), No. 3, 592-623
https://doi.org/10.22364/bjmc.2025.13.3.03

Assessing Wi-Fi Security Protocols: A Study of Dictionary Attack Performance

Oleksii CHALYI

Institute of Social Sciences and Applied Informatics, Vilnius University, Muitines St 8, LT-44280 Kaunas, Lithuania

oleksii.chalyi@knf.vu.lt

ORCID 0009-0006-3536-9715

Abstract: This study evaluates the effectiveness of password recovery techniques targeting wireless network protocols, including WPA2, WPA3, and Wi-Fi Protected Setup. Several tools were tested for their ability to capture authentication data and perform offline dictionary attacks. The impact of system parameters was analyzed, including processor clock speed, memory size and channels, GPU usage, and backend frameworks. Results show that CPU architecture and frequency strongly affect performance, while memory configuration has negligible impact. Discrete GPUs significantly improve cracking speed, and OpenCL slightly outperformed CUDA. Combining the CPU with GPUs may reduce performance. GPU temperatures were also found to be much lower than CPU under load. A strong correlation was observed between benchmark scores and cracking performance. The study confirms that improper configurations, such as WPA3 compatibility mode, expose networks to downgrade attacks. These findings offer practical guidance on secure Wi-Fi setup and hardware selection for efficient password recovery.

Keywords: Wi-Fi security, WPA2, WPA3, WPS, dictionary attack, brute-force, password recovery, Wi-Fi handshake, ANOVA, Hashcat

1. Introduction

Since the development of smartphones with Wi-Fi connectivity, wireless technology has become widely integrated into everyday digital life (Kanagachidambaresan, 2022). To connect a smartphone or any other device that suppo rts wireless internet access, users typically purchase a Wi-Fi router and configure it to function as an access point (AP) (Dondyk and Zou, 2013). This setup is performed by accessing the router's administrative panel via an IP address (commonly 192.168.0.1) or a dedicated URL (e.g., miwifi.com for Xiaomi devices) (Ferriz, 2013). The default login credentials—usually “admin” for both username and password—are often printed on the router’s label (Niemietz and Schwenk, 2015). During the configuration process, users must select a security protocol for their AP (Potter and Fleck, 2003). Once a security protocol is chosen and a password is set, an Ethernet connection is established to the router, enabling Wi-Fi access for smartphones, tablets, and other devices (Jewell et al., 2015).

Due to the widespread adoption of Wi-Fi networks, the question is no longer whether wireless security protocols can be compromised, but rather how and under what

conditions they are most vulnerable (Pahlavan and Krishnamurthy, 2020). The original protocol, Wired Equivalent Privacy (WEP), was introduced in the IEEE 802.11 standard in 1997 (Pahlavan and Krishnamurthy, 2020). However, WEP was quickly rendered insecure due to serious limitations, including its use of the outdated RC4 encryption algorithm and insufficient key length (Jean-Philippe, 2024). In 2003, the Wi-Fi Protected Access (WPA) protocol was released, introducing the TKIP encryption algorithm as a temporary solution (Jiang and Garuba 2008). This protocol was also soon compromised (Moen et al., 2004). WPA2 followed in 2004, offering more robust security with AES encryption and becoming the most widely adopted wireless security standard (Moissinac et al., 2021). In 2018, WPA3 was introduced, providing stronger protections against brute-force attacks through individualized encryption and password salting techniques (Sagers, 2021).

Among the most effective techniques for breaching Wi-Fi networks are brute-force attacks, which systematically attempt all possible password combinations until the correct one is identified (Por et al., 2024; Chalyi and Kolomytsev, 2023). A more efficient variant, the dictionary attack, uses a predefined list of likely passwords instead of attempting every possible combination (Bosnjak et al., 2018). These wordlists may be manually created, downloaded from the internet, or derived from leaked password databases, such as the top one million most frequently used passwords. Although dictionary attacks significantly reduce the time needed compared to pure brute-force methods, their success depends entirely on whether the target password is included in the selected list (Narayanan and Shmatikov, 2005).

The aim of this research is to evaluate the efficiency of various password recovery tools targeting WPA2 and WPA3 security protocols, with a particular focus on dictionary-based and brute-force attacks. Unlike prior studies, this work systematically analyzes the impact of hardware characteristics, including memory size, memory channel configuration (single vs. dual), CPU clock speed, and architectural generation on password recovery performance.

In addition, this study explores GPU acceleration in offline password cracking, comparing integrated and discrete GPUs, backend frameworks (CUDA vs. OpenCL), and hybrid CPU + GPU configurations. A particularly novel finding is that combining a CPU with multiple GPUs can degrade overall performance, contradicting common assumptions about hybrid acceleration.

The study also evaluates memory usage across varying wordlist sizes, revealing that RAM consumption scales more with device power than with dictionary size. Moreover, thermal analysis showed that GPU temperatures remain significantly lower than CPU temperatures during sustained cracking operations—an important consideration for long-term attacks and hardware longevity.

The research also investigates the vulnerabilities associated with Wi-Fi Protected Setup (WPS) and the practical feasibility of WPA3 downgrade attacks, which remain significant risks in many wireless networks when improperly configured (Chalyi et al., 2025; Abasi-Amefon et al., 2020; Bruzgė et al., 2023).

2. Related Work

Moissinac et al., (2021) in their work, investigate wireless encryption and WPA2 weaknesses. They describe the workings of the WEP and WPA2 algorithms and examine

their key generation process. Using the Aircrack-ng tool and a Kali Linux 2020.1 virtual machine, they conducted experiments simulating three operations used for WPA2-PSK calculations. It took 33.415 seconds for 1,000 operations to find the key using the Aircrack-ng tool. To address the vulnerabilities of WPA2-PSK, they proposed a solution that increases the time required for key discovery by 2.9 times—up to 96.743 seconds. Their approach involved replacing the standard WPA2-PSK authentication mechanism with a combination of RADIUS authentication and an SSL handshake, which adds an additional layer of security. The experiments showed that these modifications significantly increased the computational overhead for an attacker attempting to crack WPA2-PSK. Additionally, their study highlighted the impact of increasing computational power on WPA2-PSK vulnerabilities. The results demonstrated that a higher computational cost could serve as a deterrent for brute-force attacks. However, the authors also acknowledged that their proposed approach does not fully mitigate dictionary attacks or precomputed hash-based exploits. They concluded that the adoption of WPA3, with its Simultaneous Authentication of Equals (SAE) handshake, could further enhance security by eliminating weaknesses inherent in WPA2-PSK. Their findings emphasize the necessity of continuous improvements in wireless encryption standards to keep pace with evolving attack methodologies.

Appel and Guenther (2022) investigate the improvements of WPA3 over WPA2. Their paper aims to provide an overview of the designs of WPA2 and WPA3, including their currently known vulnerabilities, and attempts to determine whether WPA3 is still a viable successor or if it has already been compromised beyond repair. Moreover, the paper highlights critical security weaknesses in WPA3, particularly the Dragonblood attacks, which expose vulnerabilities in the SAE handshake. These attacks include downgrade attacks that force clients to connect using WPA2, side-channel attacks leveraging memory access patterns or timing variations, and denial-of-service attacks that exploit the high computational cost of WPA3 authentication. The authors argue that while WPA3 addresses known vulnerabilities of WPA2, its flaws raise concerns about its long-term viability. They emphasize that the transition mode, intended to ease adoption, paradoxically weakens security by enabling attacks on WPA3 networks. Furthermore, the paper discusses the challenges in mitigating these vulnerabilities, as countermeasures often introduce additional computational overhead, making WPA3 adoption more difficult, especially for resource-constrained devices. The paper concludes that although WPA3 offers improvements, its current state requires further refinements or alternative solutions to ensure robust wireless security, highlighting the need for continued research and potential protocol modifications.

Moroz (2024), in her qualification paper, investigates the WEP and WPA1-3 security protocols and their resistance to brute-force and dictionary attacks. She also explores the relationship between the execution time required for a dictionary attack and the security protocol used. According to this work, as of May 2024, only 1.28% of all Wi-Fi networks worldwide operate using WPA3, the most secure known security protocol for wireless networks. The experiments were conducted on a locally installed Kali Linux 6.6.9-amd64 operating system. The tools Aircrack-ng, Rockyou, and Hashcat were used. An offline dictionary attack was executed on a Huawei Wi-Fi AX3 router using the Rockyou dictionary on supported security protocols. A total of 46 iterations were performed with three passwords. Appendix B of Moroz's work presents a table listing the passwords, security protocols, and the time required to perform the dictionary attack. The results show that for a simple password (eight digits), the required time for a

dictionary attack was the same across all WPA1-3 security protocols. However, when the password became more complex (14 digits), the attack time for WPA1 was lower than for WPA2. Notably, in this experiment, the time required to crack WPA3 was 16,524 times longer compared to WPA1, which could be the result of error. The experiment demonstrated that without exploiting vulnerabilities to downgrade the security protocol to WPA2-PSK in WPA2-PSK/WPA3-SAE mode, it is impossible to execute offline dictionary and brute-force attacks due to the complexity of the Dragonfly handshake mechanism. Additionally, the WPA3-SAE protocol is resistant to de-authentication attacks due to the Security Association (SA) mechanism, which prevents the control of management frames, such as unencrypted de-authentication frames.

Fatihah Wan Mustapha et al. (2020) proposed a WiFi approximated signal quality measurement method to be used with a Brute Force algorithm in looking for the best placement of AP in indoor locations. The result shows that the approximated signal quality generated by the proposed algorithm almost equals the actual strength measured with an acceptable error. Instead of performing multiple signal strength measurements at different AP locations, their method requires only a single initial measurement. The algorithm then approximates the expected signal strength at other locations, allowing for an optimized AP placement without extensive physical testing. This significantly reduces the time required for network planning while maintaining accurate predictions of signal distribution. Their approach aims to optimize AP placement to ensure both cost-effectiveness and sufficient coverage, addressing the common issue of inefficient AP deployment in indoor environments. The new placement of AP proposed by their algorithm also manages to ensure a minimum of 84% WiFi strength in each room if all 4 APs were used. Moreover, their findings indicate that even with a reduced number of APs, acceptable coverage can be maintained—2 APs are sufficient to achieve at least 72% WiFi signal quality in each room. The study highlights the efficiency of the Brute Force algorithm in systematically exploring placement options, ultimately leading to a configuration that balances network performance and installation cost.

Prodani and Rista (2024), in their work, analyze the security of passwords in Wi-Fi networks using the Airgeddon tool and applying brute-force and dictionary attacks . By capturing handshakes on different Wi-Fi networks, they estimate the time needed to decrypt them while simultaneously analyzing the hardware performance and the effectiveness of brute-force and dictionary attacks. In their experiments, they used three hosts with different CPU, GPU, RAM, and storage types—HDD, SSD SATA3, and SSD NVMe. They launched 10 APs with different passwords, captured handshakes, and attempted to crack them using the three different hosts. Table 2 in their work presents the results, showing that hardware specifications significantly impact the cracking time, with Host 3 performing much better than Hosts 1 and 2. The results demonstrate that more powerful hardware can drastically reduce the time required for brute-force attacks. For example, for cracking their second SSID (with numeric passwords only), Host 1 took 1238 seconds, Host 2 took 467 seconds, and Host 3 only 5 seconds—2.65 and 247 times faster, respectively, compared to Host 1. Furthermore, the paper compares the effectiveness of brute-force and dictionary attacks for different passwords. The results indicate that while brute-force attacks require significant computational power and time, dictionary attacks can be much faster if the password exists in the predefined dictionary file. The findings suggest that adopting strong password practices, including complex combinations of characters, avoiding common phrases or acronyms, using long and

varied passwords, and avoiding frequent repetitions, improves the security of Wi-Fi networks.

3. WPA/WPA 2

The main attack on WPA/WPA2 security protocols in this research was a dictionary-based attack on the hashes contained in the captured handshake. The analyzed tools were divided into the following categories:

1. Tools for capturing handshakes.
2. Tools for brute-forcing captured hashes.
3. Universal tools that can capture handshakes and brute-force them.

3.1. Handshake Capture

Several tools were selected for evaluating WPA/WPA2 handshake capture efficiency: airodump-ng, hcxdumptool, besside-ng and Fluxion.

Airodump-ng, a component of the Aircrack-ng suite, is widely used in security-focused Linux distributions (e.g., Kali Linux, Parrot OS) (Kolev and Yordan, 2024). It enables monitor mode and captures handshake data from nearby access points. Combined with deauthentication attacks (e.g., via aireplay-ng), it facilitates the forced re-authentication of clients, allowing the handshake to be captured in *.cap* format.

Hcxdumptool, from the Hcxtools package, performs low-level Layer 2 attacks and captures handshake data in *.pcapng* format (WEB (a)). The captured data must be converted to the *.hc22000* format using *hcxpcapngtool* for compatibility with tools like Hashcat.

Besside-ng, also part of the Aircrack-ng suite, automates the process by enabling monitor mode, scanning for targets, and launching deauthentication attacks without manual configuration (Barybin et al., 2019). While convenient, its lack of fine-tuned control may lead to instability in multi-AP environments.

Fluxion differs significantly from other tools, as it uses social engineering rather than brute-force techniques (WEB (b)). After forcing disconnection from the target AP, it sets up a fake access point to phish the Wi-Fi credentials. Although it does not support hash cracking, Fluxion can verify handshake capture using internal modules such as Aircrack-ng or Cowpatty.

3.2. Brute-Force and Dictionary attack

Several tools were evaluated for their capabilities in performing brute-force and dictionary-based attacks on WPA/WPA2 handshake hashes. These included Aircrack-ng, John the Ripper, Cowpatty, and Hashcat.

Aircrack-ng is a widely used utility that supports both dictionary and brute-force attacks on *.cap* handshake files (Jain et al., 2022). Its integration into many security toolkits, including Kali Linux, makes it a common baseline for comparative testing. Aircrack-ng processes WPA handshake hashes directly and attempts to recover the passphrase using a supplied wordlist.

John the Ripper requires the *.cap* file to be converted to a compatible format using the *wpapcap2john* utility (WEB (c)). Once converted, the tool applies a wordlist attack

using its internal password-cracking engine. Its modular design allows it to support a variety of hash formats and attack strategies.

Cowpatty performs offline dictionary attacks against WPA/WPA2 networks using pre-shared key (PSK) authentication (WEB (d)). The tool accepts *.cap* files and a wordlist, and matches the computed hash values with the handshake contents. The attack performance is highly dependent on the presence of precomputed hash tables (e.g., rainbow tables) or optimized hash-caching mechanisms.

Hashcat is the most advanced tool in this evaluation, supporting GPU acceleration and distributed cracking across multiple devices (WEB (e)). It requires conversion of *.cap* files into *.hc22000* format, compatible with Hashcat's optimized WPA2 cracking routines (WEB (f)). Hashcat supports numerous attack modes, including dictionary, mask, hybrid, and rule-based approaches, and enables detailed benchmarking across different hardware configurations.

3.3. Universal Tools

Universal tools that integrate both handshake capture and password cracking functionalities were also evaluated. These included Airgeddon, Wifite, and Fern WiFi Cracker.

Airgeddon is a comprehensive Bash-based framework designed for wireless network auditing (WEB (g)). It integrates a variety of utilities, such as *airmon-ng*, *aireplay-ng*, and *Hashcat*, and supports both manual and semi-automated workflows. The tool provides a unified interface for capturing WPA/WPA2 handshakes and executing attacks using different cracking engines, including *aircrack-ng* and h*ashcat*. It also supports rule-based attacks and various deauthentication methods (e.g., *mdk4*, *aireplay-ng*). Although Airgeddon is not pre-installed in Kali Linux, it remains popular due to its modular structure and wide feature set.

Wifite is an automated wireless auditing tool that wraps multiple attack tools, including *aircrack-ng*, *pyrit*, *reaver*, and *tshark* (WEB (h)). It is designed for minimal user interaction, enabling fully automated WPA/WEP attack sequences. Wifite can capture handshakes, conduct deauthentication attacks, and apply dictionary-based password cracking. By default, it uses *aircrack-ng* as the cracking engine and stops execution after a predefined duration if no successful match is found.

Fern WiFi Cracker is a graphical wireless security auditing tool written in Python using the Qt framework. It supports WEP, WPA, and WPS key recovery and provides a GUI-based interface for scanning, capturing, and attacking Wi-Fi networks. Due to its reliance on native hardware interfaces, Fern WiFi Cracker operates effectively only in live environments and may not function properly in virtualized setups. Unlike command-line tools, its usability is geared toward educational or demonstration purposes rather than high-performance testing.

3.4. Creating Wordlists

Brute Force is a good attack for breaching short random passwords. On the other hand, Dictionary attacks are better for attacking long passwords based on real words or words from a specific dictionary, wordlists or a defined reference source (Hastings et al., 2013). Despite the fact that a brute-force attack can be more successful than a dictionary attack, it is much more time-consuming. Users can use pre-existing wordlists (Chalyi and

Stopochkina, 2024), such as the 10 million most popular passwords, or utilize the wordlist storage pre-installed in Kali Linux.

In addition to static lists, custom wordlists can be generated using tools such as Maskprocessor and Crunch, both of which allow for the creation of structured and exhaustive password combinations tailored to specific attack scenarios.

Maskprocessor is a high-performance tool that generates passwords based on user-defined masks. Each position in the generated password string can be constrained to specific character sets (e.g., lowercase, uppercase, digits, special characters), enabling targeted generation. This approach is particularly useful when the attacker has partial knowledge of the password structure.

Crunch provides similar capabilities, allowing users to define the minimum and maximum length, custom patterns, and character sets. In contrast to Maskprocessor, Crunch outputs entire wordlists as files, which can be useful for offline analysis or benchmarking.

Wordlist generation tools are essential when attacking strong but pattern-based passwords, such as those using repeated substitutions or corporate naming conventions. While these methods are computationally intensive, they significantly improve the likelihood of success in targeted attacks compared to using generic dictionaries.

4. WPS

WPS allows the exchange of keys to be done over-the-air between a wireless router and compatible end devices. This is usually implemented by way of a hardware ‘WPS’ button on the wireless router and a hardware or software ‘WPS’ button on the end device. The WPS PIN is an eight-digit number used to add a new client to the network (Sadeghian, 2013). When a client attempts to join the network using the WPS PIN, the access point will check the validity of the PIN separately in two halves. The first half consists of four digits (10,000 possibilities), and the second half consists of three usable digits and one checksum digit, resulting in three effective digits (1,000 possibilities). According to Wifite, three attacks could be conducted (WEB (h)):

1. **Pixie-Dust attack**. This attack exploits the fact that on some devices Enrollee Secret Nonce 1 (E-S1) and Enrollee Secret Nonce 2 (E-S2), which are two hashes of two halves of splitted PIN are generated using insecure pseudo-random number generators which have 32 bits of state and no external entropy and that each half of the PIN takes at most 10^4 guesses (WEB (i)).
2. **Brute-Force PIN attack**. A design flaw in the WPS specification for PIN authentication significantly reduces the time required to brute-force the entire PIN because it allows an attacker to know when the first half of the 8-digit PIN is correct (Jared, 2011).
3. **Null PIN attack**. This attack can be used against access points that do not follow the WPS checksum on the last digit of the PIN (WEB (j)).

The lack of a proper lockout policy after a certain number of failed attempts to guess the PIN on many wireless routers makes this brute-force attack much more feasible. Some routers have defense systems against WPS PIN attacks, which lock the WPS connection

after several failed WPS PIN attempts. Some routers can be locked for 1 or 2 minutes, while others are locked for a long time and are difficult to crack (Ye et al., 2024).

4.1. Tools for breaking WPS PIN

Multiple tools were evaluated for their ability to exploit known design flaws in Wi-Fi Protected Setup (WPS), particularly vulnerabilities related to PIN-based authentication mechanisms. The analyzed tools include Bully, Reaver, OneShot, Wifite, and Airgeddon.

Bully is a C-based WPS brute-force attack tool that improves upon the original Reaver implementation (WEB (k)). It offers enhanced performance through optimized memory and CPU usage, correct endianness handling, and reduced dependencies. Bully requires the BSSID of the target access point and supports multiple attack types including brute-force PIN cracking and Pixie-Dust-style attacks.

Reaver is a widely known tool for exploiting WPS PIN vulnerabilities (WEB (l)). It performs brute-force attacks and can recover WPA pre-shared keys (PSK) upon successful PIN discovery. Reaver also includes *wash*, a utility for identifying WPS-enabled access points. Like Bully, it requires prior knowledge of the target BSSID and operates primarily via CLI.

OneShot is a lightweight Python script designed specifically for executing Pixie-Dust attacks. Unlike other tools, it does not require switching to monitor mode and focuses solely on exploiting Pixie-Dust attack.

Wifite, originally developed with a focus on WPS exploitation, offers a high level of automation. It supports multiple attack types including Pixie-Dust, NULL PIN, and brute-force PIN attacks, utilizing both Bully and Reaver as backends.

Airgeddon integrates WPS attack capabilities within a broader Wi-Fi auditing framework. It supports both Bully and Reaver for brute-force and Pixie-Dust attacks, and allows selection of attack method via its modular script interface. Additionally, it includes support for database-based known-PIN attacks, making it suitable for extended testing scenarios.

These tools were compared using the following criteria:

- **GUI** – Graphical User Interface of the tool; if the tool has only a command-line interface, the value is marked as CLI.
- **WPS Attack Type** – List of possible attacks (as described above) that can be performed using the selected tool.
- **Pre-installed** – Indicates whether the tool is pre-installed in Kali Linux.
- **Auto Monitor Mode** – Specifies whether the tool requires manual setup of the network interface in monitor mode.
- **Language** – Programming language used by the tool.
- **BSSID** – Indicates whether the tool requires specifying the target BSSID before performing the attack.

5. WPA 3

In addition to direct password attacks, a downgrade attack scenario was analyzed to evaluate the vulnerability of WPA3-enabled networks operating in mixed-mode

compatibility. This type of attack exploits the transition mode (WPA3/WPA2) available on many modern routers, allowing legacy clients to connect via WPA2. As a result, an attacker can coerce a WPA3-capable client into connecting to a rogue access point using WPA2, thereby exposing it to traditional handshake capture and password cracking techniques.

The downgrade attack is based on the Evil Twin technique (WEB (m)). A fake access point is deployed with the same SSID as the legitimate one, but only supporting WPA2 encryption. If the victim device has previously connected to the original network and transition mode is enabled, it may automatically attempt to reconnect to the fake WPA2 AP (WEB (n)). Upon reconnection, a standard WPA2 handshake can be captured and subjected to dictionary-based password recovery.

The experimental setup required two wireless interfaces: one to broadcast the rogue access point using *hostapd*, and another to capture the resulting handshake via *airodump-ng*. Subsequent dictionary attacks were performed using *aircrack-ng*. Router firmware was configured in WPA3/WPA2 compatibility mode to enable downgrade conditions. Devices using WPA3-only mode were not vulnerable under these circumstances, confirming that the attack vector is specific to transition configurations.

6. Experimental Setup and Variables

The following tools were used for the comparison of brute-force tools for cracking WPA2 handshakes: Aircrack-ng, John, Hashcat, and Cowpatty.

Airmon-ng was used as the tool for capturing handshakes due to the universality of its .cap files, which can be converted to .john or .hc22000 formats.

The experimental setup was designed to evaluate the influence of various parameters on the success of password dictionary-based attacks.

To assess dictionary-based attack performance, the attacker's hardware configuration was modified across several key variables:

- **RAM amount**: 16GB (single-channel), 2x8GB (dual-channel), 24GB (dual-channel).
- **RAM configuration**: single-channel and dual-channel setups.
- **GPU type**: integrated Radeon RX Vega 7.
- **CPU clock speed**: fixed at either 1.4 GHz, 1.7 GHz and 2.9 GHz

The complete **testbed specifications** were kept consistent for comparability and are listed below:

- **Host system**: Asus TUF A17 FA706II equipped with a Ryzen 7 4800H processor.
- **Memory modules**:
  - 2× Samsung DDR4 8 GB 3200 MHz.
  - 1× Juhor DDR4 16 GB 3200 MHz.
- **Wi-Fi adapters**: Intel AX210NGW (M.2 interface), Ralink MT7601U (USB interface, required for WPA3 testing).
- **Operating system**: Kali Linux Live Image.
- **Routers used as access points**:
  1. TP-Link WR843ND (Wi-Fi 4, 2.4 GHz, WPA2, WPS-enabled).

2. Xiaomi BE5000 (Wi-Fi 7, dual-band 2.4/5 GHz, WPA3).

The SSIDs the tested access points were modified to include the author's ORCID ID as a suffix (e.g., "BE5000_0009-0006-3536-9715"), thereby confirming ownership and allowing independent verification of legal use.

To analyze the impact of CPU architecture on WPA2 dictionary attack performance, a set of controlled experiments were conducted under fixed clock conditions. Specifically, the following CPUs were tested:

- Ryzen 7 7700 and Ryzen 7 4800H at 3.6 GHz (to isolate architectural differences with equal frequency and core/thread count).
- Ryzen 7 7700, Ryzen 7 4800H, and Pentium N3530 at 1.7 GHz (to compare across generations and classes of processors).

To evaluate the correlation between general-purpose CPU performance metrics and actual cracking throughput, these three CPUs were also tested at their maximum native performance settings. The resulting performance data was compared against publicly available PassMark CPU benchmark scores.

Additionally, the role of GPU acceleration in hash processing was examined by testing several configurations:

- Discrete GPUs: AMD Radeon RX 5700 XT and NVIDIA GTX 1650 Ti (evaluated under both CUDA and OpenCL backends).
- Integrated GPUs: AMD Radeon Graphics (Ryzen 7000 iGPU) and Radeon RX Vega 7.
- Hybrid setups combining both discrete and integrated GPUs.
- Full configurations with CPU + iGPU + dGPU (e.g., Ryzen 7700 + Radeon iGPU + RX 5700 XT).

To explore the relationship between GPU benchmark scores and hash processing performance, PassMark G3D scores were used for correlation analysis. For GPUs with multiple supported APIs (e.g., CUDA and OpenCL on the GTX 1650 Ti), only the configuration yielding the higher speed was used.

Finally, the **experimental design** ensured statistical consistency and reproducibility:

- Each configuration was tested in three repetitions for 2.9 GHz, two repetitions for 1.7 GHz, and one repetition for 1.4 GHz. This was done due to the wider use of attacking systems with 1.7 GHz and 1.4 GHz compared to 2.9 GHz, as well as the low result difference between iterations.
- The wordlist tool with the dictionary file *rockyou.txt*, which contains 14,344,392 entries, was used for all brute-force attempts. The chosen password was !!HELLOKITTY!!, which is located near the end of the list (entry #14,338,078), ensuring a longer dictionary attack time and making the performance differences more apparent.
- To evaluate the effect of dictionary volume on host RAM usage, two additional wordlists were used: *md5decryptor.txt*, containing 3,431,316 entries, and an expanded version of r*ockyou.txt* (×5), containing 71,721,955 entries. RAM consumption was measured across different devices for each of these wordlists.
- For dictionary attacks on WPA3, a modified *wordlist-probable.txt* was used.

- Environmental variables such as temperature and radio interference were held constant.
- One-way ANOVA was used to evaluate the significance of categorical variables. If the p-value is lower than 0.05, the categorical variable has a significant impact.
- In addition, CPU and GPU configurations were selected to represent a wide range of computing capabilities, from entry-level processors (e.g., Pentium N3530) to modern high-performance systems (e.g., Ryzen 7 7700 with discrete GPU). This allowed the experimental dataset to reflect real-world attack scenarios and hardware diversity. All measurements were verified for consistency across repeated runs.

6.1. Legal and Ethical Disclaimer:

All experiments were conducted exclusively on equipment and access points owned by the author. No tests were performed on networks or devices without explicit authorization. This study strictly adheres to ethical research standards and does not encourage unauthorized access to wireless networks. Readers are reminded to comply with applicable laws and obtain consent before conducting any similar experiments.

7. Results

7.1. WPA/WPA 2

The Cowpatty tool was excluded from the comparison after observing its speed at the highest possible frequency, shown in Figure 1.

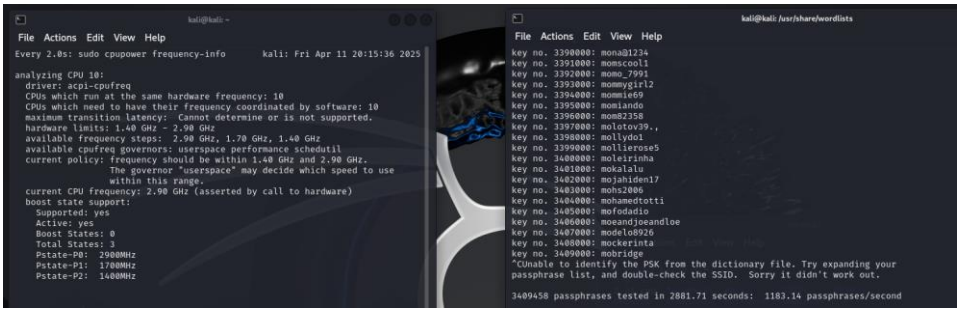

Figure 1. Cowpatty Speed at the Highest Frequency

Using the *matplotlib.pyplot* Python library, a figure demonstrating the average speed comparison between all tested tools was created, shown in Figure 2.

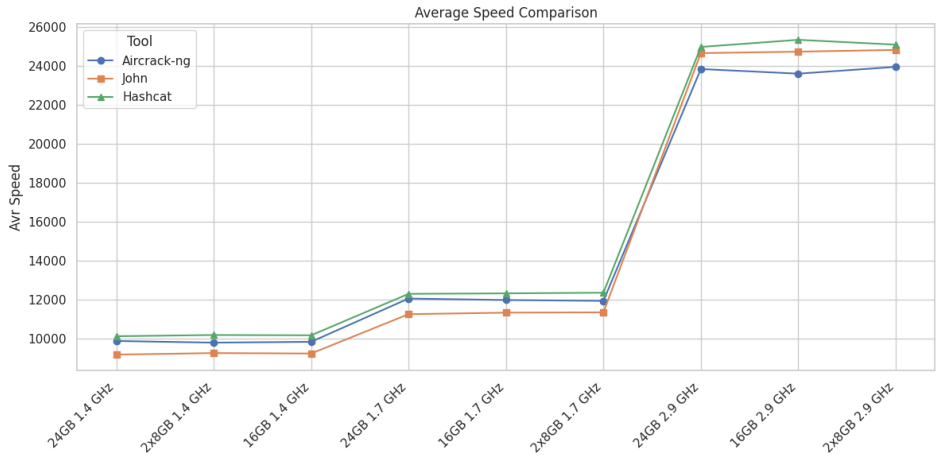

Figure 2. Average Speed Comparison of Brute-Force Tools

This figure demonstrates not only that Hashcat provides the highest speed among all tested tools, but also highlights a strong correlation between higher CPU clock speeds and brute-force performance. Additionally, it indicates that the amount of RAM and its channel configuration (dual or single) have minimal impact on brute-force speed. These observations were further confirmed using the one-way ANOVA method with the f_oneway function in Python, shown in Table 1.

Table 1. One-Way ANOVA Results for Estimating the Influence of Parameters on Brute-Force Speed

Tool	RAM F value	RAM p value	CPU clock F value	CPU clock p value
Aircrack-ng	0.0002	0.9998	13,529.81	1.09e-11
John	0.0001	0.9999	57,183.20	1.44e-13
Hashcat	0.0002	0.9998	15,946.85	6.65e-12

The results show that the p-value for RAM is nearly equal to 1 for each tool, indicating no significant influence on brute-force speed. In contrast, the p-value for CPU clock speed indicates a significant impact on performance. The data also reveal that John is more dependent on higher CPU clock speeds compared to the other tools. At 1.4 GHz and 1.7 GHz, Aircrack-ng outperforms John; however, at 2.9 GHz, John surpasses Aircrack-ng in speed.

Additionally, the time required to complete a dictionary-based attack attack was evaluated for each tool. The tools reported the duration in minutes, which was then converted into seconds for consistency and presented in a comparative figure across all three tools, shown in Figure 3.

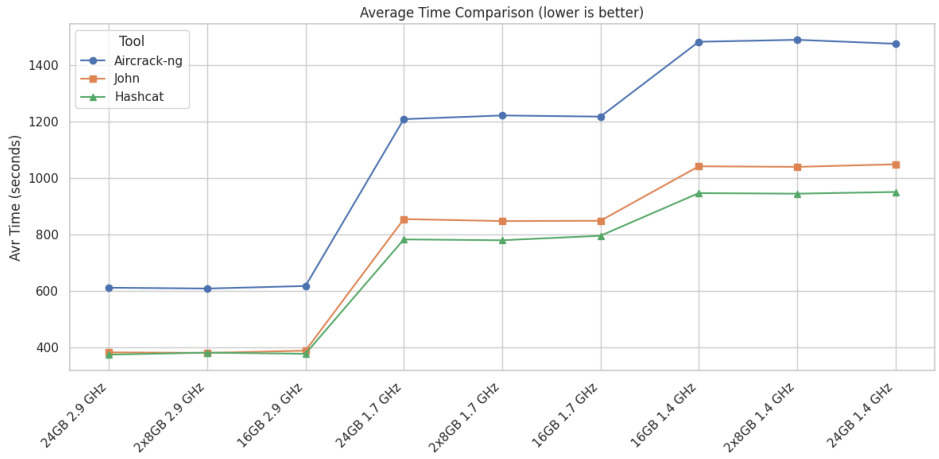

Figure 3. Average Time Comparison for Brute-Force Tools

A lower value indicates better performance, as less time is required to find the password. This figure demonstrates that at 2.9 GHz, the time required by John and Hashcat is nearly the same, whereas at lower frequencies, Hashcat performs better. The one-way ANOVA method was used to investigate the influence of CPU and RAM on the performance of brute-force tools for handshake hash cracking (Table 2).

Table 2. One-Way ANOVA Method for Estimating Parameters Influencing Brute-Force Time

Tool	RAM F value	RAM p value	CPU clock F value	CPU clock p value
Aircrack-ng	0.0003	0.9997	15,638.78	7.06e-12
John	0.0002	0.9998	19,151.01	3.84e-12
Hashcat	0.0002	0.9998	8,244.14	4.81e-11

It also shows that the RAM amount and channel usage have almost no effect on the required time, unlike the CPU clock. For time values, increasing the CPU clock provides a similar advantage for all tools. However, at 1.4 GHz, the time required by John and Hashcat was nearly the same, while at higher frequencies, the difference became more noticeable.

After completing the brute-force hash attack, each tool displays the elapsed time and speed (in passwords or hashes per second). Hashcat is the only tool that also reports the CPU temperature after finishing the attack. Based on its output, a figure illustrating the

dependence of temperature on CPU clock and RAM configuration was created, shown in Figure 4.

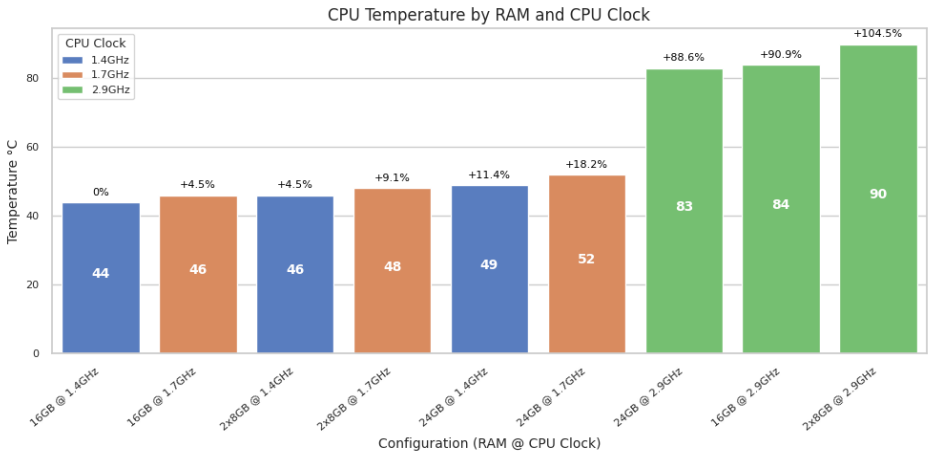

Figure 4. CPU Temperature Comparison

This figure shows that at higher frequencies, the CPU temperature increases significantly. It also indicates that when using 8 GB of RAM in dual-channel mode, the temperature was the highest, reaching 90 degrees Celsius. However, with 24 GB of RAM, the CPU temperature was highest at 1.4 GHz and 1.7 GHz. Additionally, the one-way ANOVA method was used to investigate the dependencies between temperature, RAM, and CPU clock (Table 3).

Considering that Hashcat consistently demonstrated the highest performance among all tested tools and fully supports GPU utilization for dictionary-based attacks, it was selected for all further experiments. Table 3 presents the results of WPA2 cracking performance across different CPU architectures at fixed clock speeds.

Table 3. CPU Architecture Comparison for WPA2 Dictionary Attack Performance (Fixed Clock Speed)

CPU	Clock	Architecture	Cores	Threads	L3 Cache (MB)	Hashcat Speed (pw/sec)
Pentium N3530	1.7	Silvermont	4	4	2	920
Ryzen 4800H	1.7	Zen 2	8	16	8	12288
Ryzen 7700	1.7	Zen 4	8	16	32	28040
Ryzen 4800H	3.6	Zen 2	8	16	8	29194
Ryzen 7700	3.6	Zen 4	8	16	32	60109

As shown in Table 3, CPU architecture significantly impacts Hashcat performance.

More modern CPU architectures provide considerably higher cracking speeds, even at the same clock frequency. Figure 5 illustrates this difference visually by comparing CPUs running at the same clock speed.

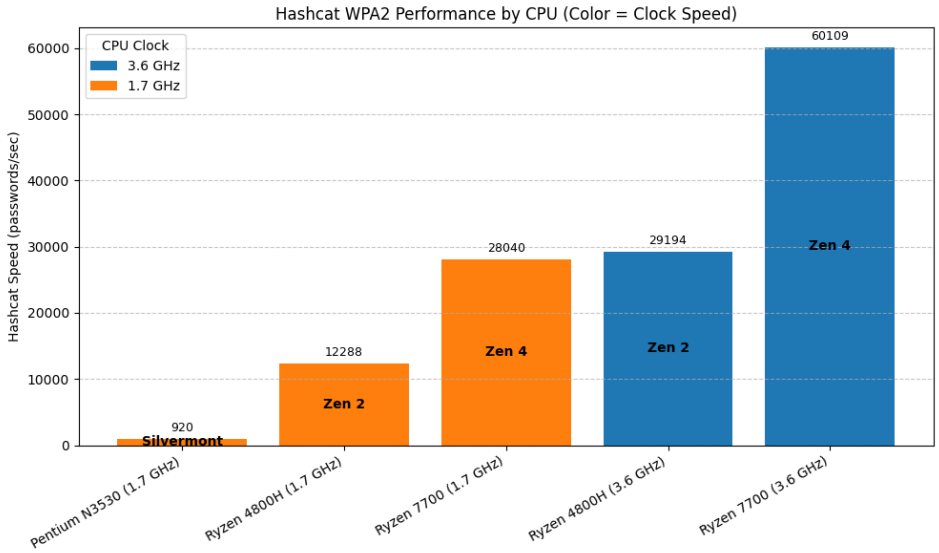

Figure 5. Hashcat Speed Comparison for CPUs at Equal Clock Frequency

The figure confirms that even under identical clock settings, more recent and powerful architectures (e.g., Zen 4) significantly outperform older or low-end designs (e.g., Silvermont).

To further investigate the relationship between overall CPU performance and dictionary attack speed, PassMark benchmark scores were used for comparison under default (unrestricted) CPU frequency settings. The results are shown in Table 4.

Table 4. Hashcat Performance and PassMark Scores at Default CPU Frequency

CPU	Avg Speed (pw/sec)	Avg Time (s)	PassMark Score
Ryzen 7700	110900	87	34457
Ryzen 4800H	29194	329.3	18258
Pentium N3530	1002	8940	1182

To evaluate the strength of this dependency, a correlation analysis was performed between PassMark scores and Hashcat speed. The result is presented in Figure 6.

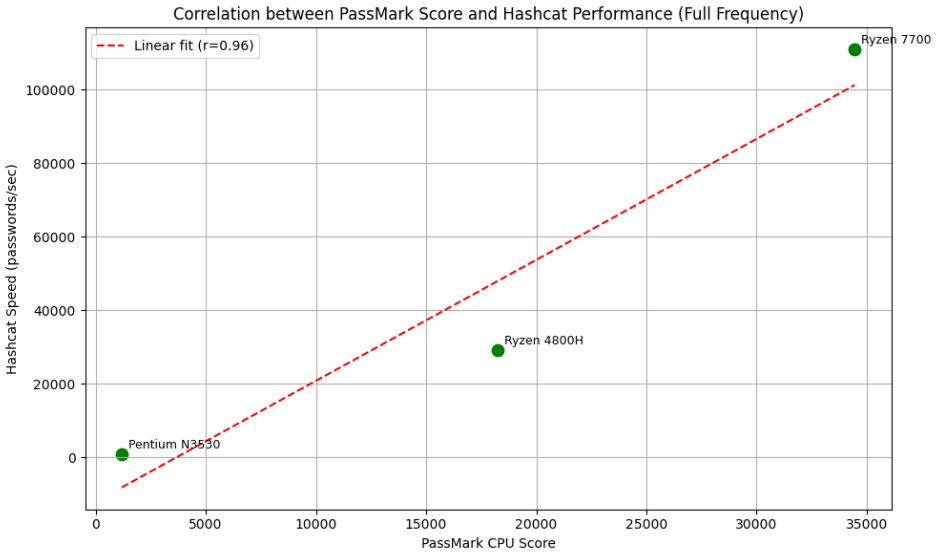

Figure 6. Correlation Between PassMark Score and Hashcat CPU Performance

The figure demonstrates a near-perfect linear correlation (r = 0.96) between PassMark CPU scores and measured WPA2 dictionary attack speed in Hashcat under default frequency conditions. This supports the hypothesis that synthetic CPU benchmarks can serve as reliable predictors of brute-force password cracking performance.

Hashcat fully supports GPU acceleration and can take advantage of both discrete and integrated graphics cards when performing dictionary attacks. Table 5 presents the comparison of different GPUs in WPA2 hash cracking performance using dictionary attacks.

Table 5. Hashcat GPU Performance for WPA2 Dictionary Attacks

GPU	Type	Core Clock (MHz)	Mem Clock (MHz)	Avg Speed (kpw/sec)	Avg Time (s)	PassMark G3D Score
Radeon Graphics (Ryzen 7700)	Integrated	2200	2900	28.48	337.5	1758
RX Vega 7	Integrated	1600	1600	45.62	211.33	3336
GTX 1650 Ti (CUDA)	Discrete	1820	6120	147	65.67	7534
GTX 1650 Ti (OpenCL)	Discrete	1820	6120	153.87	63.33	7534
RX 5700 XT	Discrete	1820	1742	411.2	22.67	16324

As shown in this table, discrete GPUs significantly outperform integrated GPUs in dictionary attacks. The RX 5700 XT, for example, achieved over 9× the speed of RX Vega 7. Figure 7 provides a visual comparison of GPU performance by type.

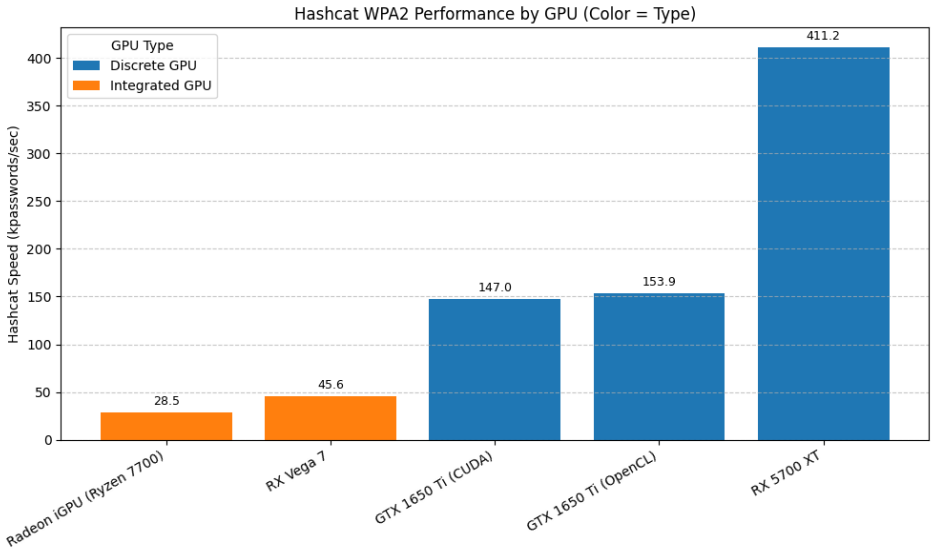

Figure 7. Hashcat Speed Comparison: Discrete vs. Integrated GPUs

This figure also shows that the OpenCL backend on the GTX 1650 Ti slightly outperformed the CUDA version in this experiment. While the difference is minor, it highlights how backend implementation and driver maturity may influence the results.

To explore the relationship between GPU benchmark scores and actual cracking performance, a correlation analysis was conducted using PassMark G3D scores. The results are shown in Figure 8.

This figure reveals an almost perfect linear correlation ($r = 0.99–1.00$) between PassMark G3D scores and Hashcat cracking speed, supporting the conclusion that general-purpose GPU benchmarks can reliably predict dictionary attack efficiency in WPA2 scenarios.

Hashcat also supports the ability to combine multiple GPUs for both dictionary and brute-force attacks, as well as integrate CPU with multiple GPUs. During the attack process, Hashcat reports the cracking speed for each device separately, along with the combined total speed. Table 6 presents the results of using two GPUs (integrated and discrete), as well as the combination of CPU + 2 GPUs.

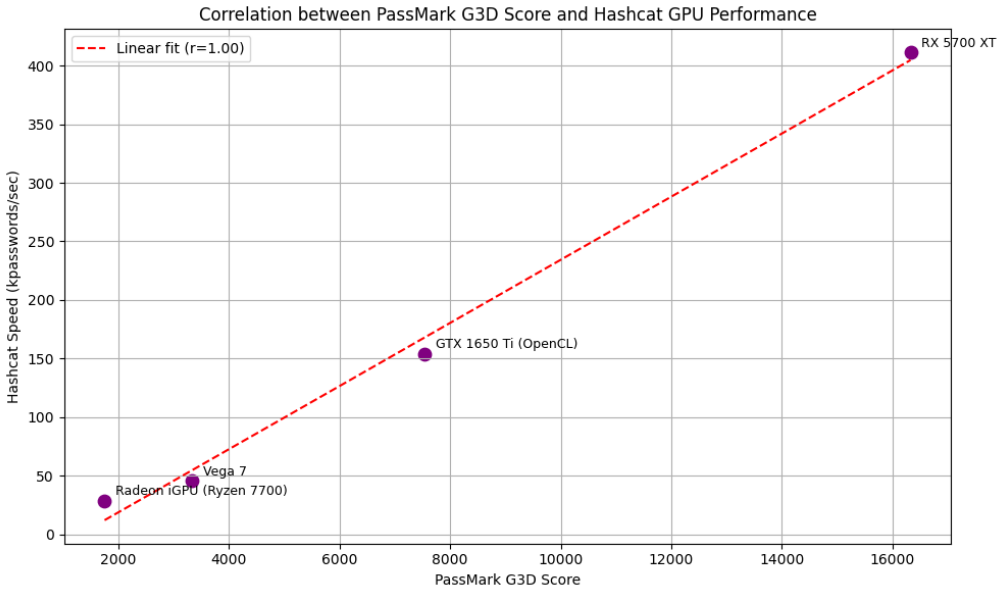

Figure 8. Correlation Between GPU Hashcat Performance and PassMark G3D Scores

Table 6. Hashcat Speed Comparison for 2 GPU and CPU + 2 GPU Combinations

Configuration	Avg Speed (kpw/sec)	Avg Time (s)
RX 5700 XT + Radeon iGPU	421.7	22.67
GTX 1650 Ti + RX Vega 7	185.5	52.33
Ryzen 7700 + RX 5700 XT + iGPU	353.83	27.33
Ryzen 4800H + 1650 Ti + RX Vega 7	162	59.33

This table demonstrates that combining an integrated and discrete GPU does not provide a significant performance boost, as indicated by the time required for hash cracking. However, the total speed of the attack increases, showing that using multiple powerful GPUs together can significantly improve the overall cracking speed by summing up their individual performances. Figure 9 visualizes these results.

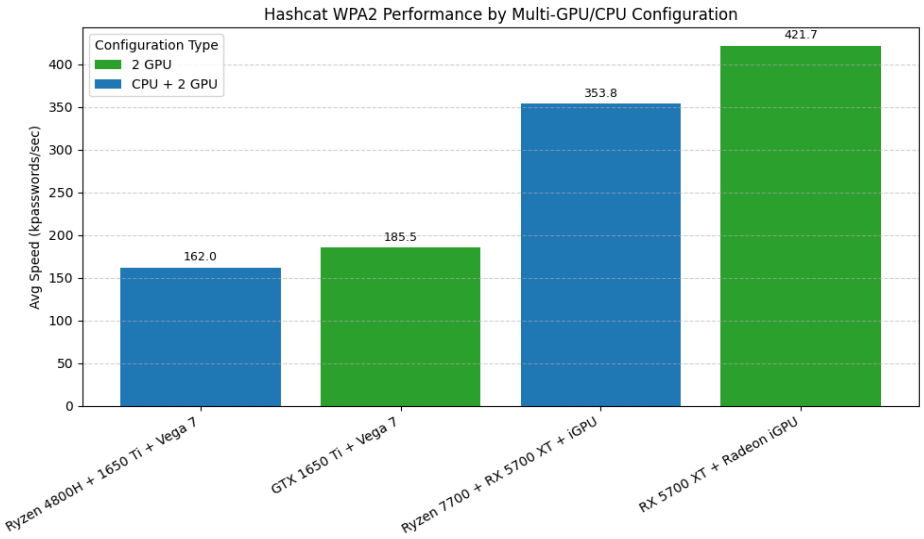

Figure 9. Hashcat Speed Comparison: Two GPUs vs CPU + Two GPUs

As shown in Figure 9, combining the CPU with both GPUs resulted in a lower performance compared to using both GPUs alone, which suggests that the CPU may not contribute effectively to GPU-accelerated tasks in this setup.

While previous experiments showed that RAM has minimal impact on dictionary attack performance using the rockyou.txt list, additional tests were conducted to determine whether this finding holds true for both larger and smaller wordlists. Table 7 presents the host memory consumption across different devices using three wordlists of varying sizes.

Table 7. RAM Usage (MB) for Different Wordlists per Device

Wordlist / Device	Ryzen 7 7700	Ryzen 7 4800H	RX 5700 XT	GTX 1650 Ti
Rockyou.txt	507 MB	520 MB	1 783 MB	1 284 MB
Md5decryptor.txt	500 MB	509 MB	1 777 MB	1 265 MB
Rockyou.txt × 5	551 MB	543 MB	1 801 MB	1 293 MB

This table demonstrates that RAM usage correlates weakly with the size of the wordlist. While RAM consumption increases slightly with larger dictionaries, the overall difference remains minimal. Figure 10 visualizes these results.

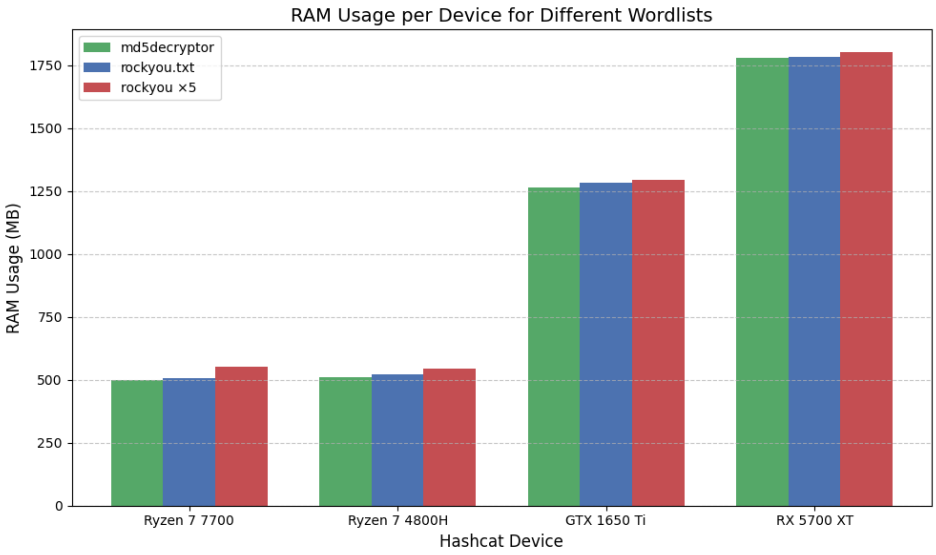

Figure 10. RAM Usage by Hashcat Devices with Different Wordlists

The figure illustrates that devices with higher cracking throughput (e.g., RX 5700 XT) also tend to exhibit greater RAM usage, though this increase is not directly proportional to the wordlist size. This confirms that dictionary size has limited influence on RAM requirements in practice.

Across all experiments, Hashcat consistently delivered the highest performance among the tested tools. CPU clock speed and architecture were the primary factors influencing cracking speed, while RAM amount and configuration had negligible effect. Among GPU-based configurations, discrete GPUs significantly outperformed integrated ones. However, combining the CPU with GPUs did not always yield improved results, suggesting that Hashcat may have limitations in parallel workload distribution across heterogeneous hardware.

7.2. WPS

Based on the tool descriptions, the following table (Table 8) was created, which demonstrates the comparison of selected tools for WPS vulnerability attacks.

Table 8. Comparison of Tools for Attacking WPS Vulnerabilities

Tool	GUI	Attack type	Pre-installed	Auto Monitor Mode	Language	BSSID
Bully	No	PIN Bruteforce, Pixiedust	Yes	No	C	Yes
Reaver	No	PIN Bruteforce, Pixiedust	Yes	No	C	Yes
Oneshot	No	Pixiedust	No	Not required	Python	Yes
Wifite	CLI	PIN Bruteforce, Pixiedust, Null Pin	Yes	Yes	Python	No
Airged don	CLI	PIN Bruteforce, Pixiedust, Null Pin	No	Yes	Python	No

In this table, Bully and Reaver are standalone command-line tools that perform brute-force and Pixie-Dust attacks on WPS-enabled access points. Wifite and Airgeddon act as frameworks that automate these attacks by leveraging tools like Bully and Reaver internally. Both Wifite and Airgeddon support additional attack types such as the Null PIN attack and offer user-friendly terminal interfaces.

Unlike Airgeddon and Oneshot, Wifite is pre-installed in Kali Linux and automatically handles monitor mode setup, making it more accessible for quick deployment. OneShot, while not requiring monitor mode, is limited to Pixie-Dust attacks and is not bundled with Kali Linux by default.

Based on this comparison, Wifite appears to be the most versatile and user-friendly option for launching WPS-related attacks, especially in penetration testing environments where ease of use and automation are valuable.

7.3. WPA 3

First, the user needs to enable monitor mode for both cards using airmon-ng and kill any processes that may affect the results, such as NetworkManager and wpa_supplicant.

To run the fake AP using hostapd via *hostapd config_name*, the following configuration was used:

interface=wlan0mon
ssid=BE5000_0009-0006-3536-9715
hw_mode=g
channel=5
auth_algs=1
wpa=2
wpa_passphrase=LETMEIN1234
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP

where *interface* is the name of the interface in monitoring mode, and *hw_mode=g* means the fake AP works on 2.4 GHz (the user should replace *g* with *a* to make the fake AP operate on 5 GHz). The ssid should have the same ESSID (name) as the target AP.

The user needs to start the command *airodump-ng --band abg interface2_name --essid target_essid -w name_of_cap_file* for direct target ESSID monitoring, shown in Figure 11.

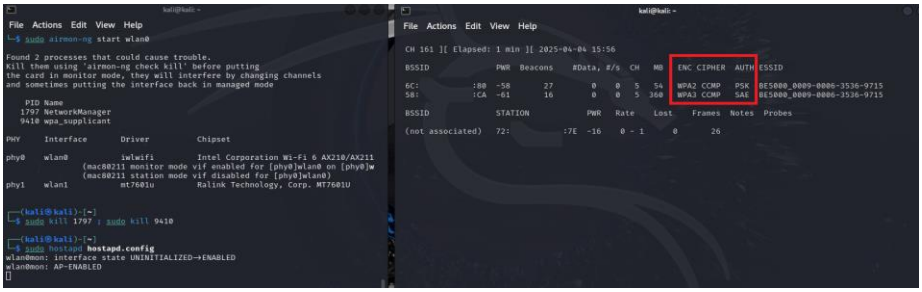

Figure 11. Result of Launching Fake AP via Hostapd

If hostapd is launched, the user will see both networks with the same ESSID but with different Encryption and Authentication. Next, the user needs to wait until the victim device connects to the fake AP; after that, the WPA handshake can be captured, shown in Figure 12.

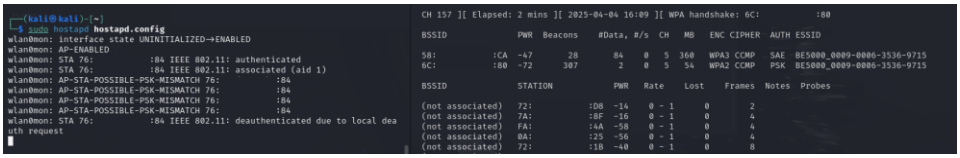

Figure 12. Captured Hhandshake using Fake AP

Additionally, it was checked whether having the fake AP on a different channel from the target AP would influence handshake capturing, shown in Figure 13.

Figure 13. Fake AP with Different Channel

The experiment has shown that it doesn't influence handshake capturing. After capturing the handshake using Aircrack-ng, the password attack can be conducted, shown in Figure 14.

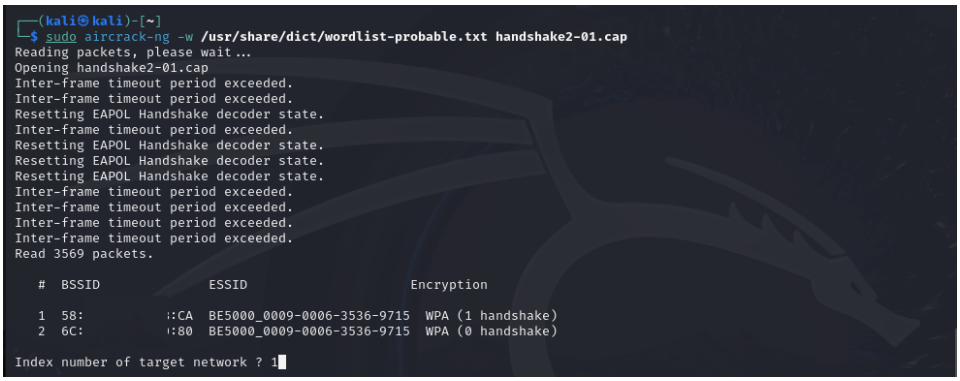

Figure 14. Aircrack-ng for Breaking Captrued Handshake

As Aircrack-ng shows in Figure 15, the handshake has two BSSIDs, but only one has the handshake (the target AP). The user needs to choose the one with the handshake and start the password attack.

Figure 15. Target AP Key Found

As a result, if the wordlist contains the correct password, the attack will be successful.

7.4. Key Findings

Cowpatty shows the worst result for WPA2 password attacks, with only 1,184 passphrases per second, making it time-consuming when using a wordlist with a large number of words.

The tools Aircrack-ng, John, and Hashcat were selected to measure the dependencies between brute-force speed, execution time, and RAM usage. The results showed that the amount and channel configuration of RAM had no significant impact, as confirmed by one-way ANOVA ($p \approx 1.0$). In some cases, single-channel 16 GB RAM even slightly outperformed dual-channel setups.

The dependence between brute-force speed, execution time, and CPU clock was measured. Three clock speeds were chosen: 1.4 GHz, 1.7 GHz, and 2.9 GHz. The results showed that with higher CPU clock speeds, the brute-force speed of each tool increased, and the required time decreased. This was also confirmed by the one-way ANOVA method, where the p-value was much less than 0.05, indicating a significant impact on brute-force speed and execution time.

Based on the lowest and highest experimental results, it was found that increasing the CPU clock from 1.4 GHz to 2.9 GHz provided the following improvements:

- For Aircrack-ng: a 245% speed boost and a 245% reduction in required time.
- For John: a 271% speed boost and a 276% reduction in password attack time.
- For Hashcat: a 251% speed boost and a 254% reduction in required time.

These results indicate that John has the highest dependence on CPU clock among all three tools, while Aircrack-ng has the lowest.

At higher CPU clocks, Hashcat outperforms Aircrack-ng by 6% and John by 2% in brute-force speed. It was also found that at 2.9 GHz, Hashcat requires only 57% of the time compared to Aircrack-ng, and 90% compared to John. Considering this data, Hashcat is the best solution among these tools for cracking handshake hashes, even though it requires converting the airodump cap file into the hc22000 format.

Comparing CPUs at equal frequency further confirmed architectural influence:

- At 3.6 GHz, Ryzen 7700 outperformed Ryzen 4800H by 205%, despite identical core/thread counts.
- At 1.7 GHz, Ryzen 7700 outperformed the 4800H by 228%, and Pentium N3530 by 3047%, illustrating how newer architectures significantly outperform older or entry-level designs.

Hashcat's support for GPU computation further improved performance. The discrete RX 5700 XT outperformed the integrated RX Vega 7 by 901%, and Radeon Graphics (Ryzen 7700) by 1443%. Among backends, OpenCL on GTX 1650 Ti outperformed CUDA by 4.7%, highlighting slight variations due to driver or API differences.

Combining two GPUs also showed improvement: using GTX 1650 Ti + Vega 7 increased cracking speed by 20% compared to GTX 1650 Ti alone. However, adding CPU to a dual-GPU setup (Ryzen 4800H + GTX 1650 Ti + Vega 7) reduced performance by 14.5%, suggesting that CPU contribution may become negligible or even limiting in mixed workloads.

Correlation analysis using PassMark benchmark scores showed strong alignment between synthetic performance metrics and Hashcat results, CPU: $r = 0.96$ and GPU: $r = 0.99$. These results confirm that benchmark scores are reliable predictors of real-world password cracking performance in dictionary attacks.

In the experiment on RAM usage dependence on wordlist size, it was found that dictionary volume has minimal impact on host memory consumption within the same Hashcat device. For example, *md5decryptor.txt*, which is approximately 21 times smaller than the 5× expanded *rockyou.txt*, showed only a 5% average difference in RAM usage across all four tested devices. However, differences in RAM usage became more apparent between devices. For instance, the RX 5700 XT, which demonstrated the highest cracking speed, consumed on average 344% more memory than the Ryzen 7 7700.

The analysis of tools targeting WPS vulnerabilities reveals that most rely on the same underlying brute-force and Pixie-Dust techniques, often implemented through shared components such as Reaver or Bully. Wifite stands out as the most comprehensive and accessible tool, supporting multiple attack types, automating setup, and being pre-installed in Kali Linux. Additionally, Wifite provides the best balance between functionality, ease of use, and availability.

WPA3 offers a high level of security against brute-force and dictionary attacks. However, using a downgrade attack, by creating a fake access point with WPA2 security and tricking the target into connecting to it, can provide an opportunity to capture the handshake of the target AP. Once the handshake is captured, it can be brute-forced using Aircrack-ng or another tool. However, executing this attack can be a challenging task, as it requires several conditions to be met: the target AP must support WPA2 compatibility mode, the user must connect to the fake AP, and the target AP’s password must be weak or present in the wordlist.

8. Discussion

8.1. Comparison with Previous Studies

Compared to earlier research that evaluated tools such as Airgeddon on three separate hosts with differing specifications (Prodani and Rista, 2024), the present study was conducted using not only separate hosts but also a single host system with controlled hardware variations. This approach allowed for more accurate attribution of performance differences to specific parameters, rather than being influenced by architectural disparities between machines. The comparison confirms, as noted in their Table 3, that dictionary attack performance is highly dependent on the host's configuration. The current results indicate that CPU clock speed has the most significant impact on brute-force performance. In contrast, factors such as RAM size, memory channel configuration, and GPU presence showed negligible influence, consistent with the fact that GPU acceleration is not utilized by the tested tools during password recovery — unlike in this study, where a discrete GPU such as the RX 5700 XT had a significant impact on dictionary attack speed.

Differences between this study and prior works (Moissinac et al., 2021 ; Moroz, 2024) are expected, as hardware specifications, CPU models, and environmental conditions vary across experiments. Even when using identical clock speeds, the results may differ due to architectural differences between processors, as confirmed in this study by comparing the Ryzen 7 7700 and Ryzen 7 4800H at the same clock speeds. Notably, in the referenced works, the evaluation was limited to Aircrack-ng. The present study expands this comparison by including Hashcat and John the Ripper, demonstrating

that Aircrack-ng is the slowest among the three tools tested in terms of brute-force performance.

Contrary to claims made in earlier research (Fatihah Wan Mustapha et al., 2020), parameters such as access point channel, frequency band, and physical distance have minimal impact on the password cracking process itself. These variables primarily affect the ability to capture the handshake, as signal strength degrades with distance. The farther the host is from the access point, the lower the signal-to-noise ratio, making handshake capture more difficult—but once the handshake is acquired, the cracking speed is unaffected by these factors.

The downgrade attack described in (Appel and Guenther, 2022) was also validated in this research. The results confirm that such an attack can be effective against WPA3, but only under specific conditions—namely, when the access point is operating in WPA3 compatibility mode (i.e., allowing WPA2 connections). If strict WPA3-only mode is enforced, the attack fails, and handshake acquisition becomes infeasible using conventional methods. Once the handshake is successfully captured, any of the evaluated tools, such as Aircrack-ng, Hashcat, or John can be employed for password recovery.

8.2. Recommendations

The attack on the WPA2 security protocol is divided into two steps: handshake capture and password attack. Each step was described, and several tools were analyzed. For handshake capture, users can use any tool capable of capturing a handshake; however, airodump-ng is recommended due to its pre-installation in most security-focused Linux distributions and its effectiveness.

During the experiments, it was determined that a possible mitigation method against handshake capture is connecting the AP user to another secure access point. After deauthentication, the device will attempt to reconnect to the closest remembered network. Therefore, it is recommended to authenticate the device in advance to both 5GHz and 2.4GHz networks, as they are likely to be closer, and the device will reconnect to them rather than to a fake AP (for WPA3) or a deauthenticated AP. This approach can increase the difficulty of successfully capturing the handshake.

Based on experimental results, Hashcat is the recommended tool for WPA2 password attacks, as it demonstrated the highest password-per-second performance and the shortest execution time compared to John the Ripper and Aircrack-ng. In contrast, Cowpatty is not recommended due to its significantly lower brute-force speed.

One of Hashcat's key advantages is its support for GPU acceleration, particularly discrete GPUs, which can increase cracking speed several-fold compared to CPU-only execution. Using multiple GPUs can further enhance performance. However, combining CPU and GPU in the same attack process may reduce overall speed, likely due to resource balancing limitations within Hashcat.

During the experiments, it was also observed that GPUs operated at significantly lower temperatures, with a maximum of 65 °C, compared to CPU temperatures reaching up to 93 °C under full load. This suggests that GPUs offer not only higher performance but also greater thermal efficiency in password cracking tasks.

Additionally, the choice of backend—CUDA or OpenCL—can slightly affect performance. In the tested configuration, OpenCL outperformed CUDA by a small margin (~4.7%). Although the difference is minor, it indicates that backend selection and driver optimization can influence final results, particularly on mid-range hardware.

Benchmark scores, such as those from PassMark, can be used to estimate expected performance in Hashcat. The experiments confirmed a strong correlation between benchmark scores and cracking speed: devices with higher PassMark scores consistently achieved better performance. Moreover, investing in a higher-end GPU yields greater performance gains than investing in a higher-end CPU. For those seeking to use multiple discrete GPUs, it is recommended to use an ATX motherboard with more than one PCIe x16 slot to ensure full bandwidth and compatibility.

It was also found that increasing RAM capacity or changing RAM channel configuration does not significantly affect brute-force speed. Instead, system performance is more strongly influenced by CPU frequency and the efficiency of the CPU's architecture. Therefore, efforts should be focused on selecting a modern high-performance CPU, rather than optimizing memory configurations.

Additionally, RAM usage showed minimal dependence on wordlist size when testing on the same Hashcat device. However, when using more powerful devices with higher password-per-second throughput, RAM consumption increases accordingly. It is recommended to use systems with at least 16 GB of RAM to ensure that performance is not limited by memory constraints.

For optimal performance, it is not recommended to run attacks from a live Linux image, as this setup can limit both CPU and GPU utilization. A better approach is to install a dedicated security-focused Linux distribution (e.g., Kali Linux or Parrot OS) directly on the system. Alternatively, Windows users can also run Hashcat with full GPU support, provided the necessary drivers and libraries (e.g., OpenCL or CUDA) are properly installed.

For beginners, a universal tool like Airgeddon can be used for attacking WPA2 access points. This tool is recommended because it features a command-line menu interface and integrates multiple tools for both handshake capture and hash brute-forcing. Its main drawback is that it is not pre-installed in Kali Linux. Wifite can be a good alternative to Airgeddon, although it requires a bit more knowledge to use effectively.

For a dictionary attack, a user needs to specify the wordlist containing passwords. It is recommended to use the wordlists folder provided by Kali Linux or use open dictionaries with the most popular passwords. If a user needs to create a universal wordlist, crunch is the optimal solution, as it displays the number of password lines and required disk space.

Despite the fact that WPS introduces a potential vulnerability to WiFi routers and is still generally recommended to be disabled, the analysis suggests it could remain enabled for the convenience of easier connection—provided certain precautions are taken. If the user chooses to keep WPS enabled, it is recommended to test the WPS locking time after a PIN brute-force attempt using Wifite (as recommended in the comparison) or another suitable tool. If the WPS remains locked for an extended period (e.g., several days), it may be considered safe to keep it enabled.

For access points using the WPA3 security protocol, it is not advisable to enable WPA3/WPA2 compatibility mode for password encryption. This mode allows downgrade attacks, exposing the network to password brute-force vulnerabilities. It is strongly recommended to disable compatibility mode and use WPA3 encryption only. In this configuration, any handshake captured through downgrade attempts will be encrypted with the WPA3 protocol, making it unusable by tools like Aircrack-ng. Because some older devices do not support WPA3, it is recommended to configure the

5GHz network with WPA3-only encryption and the 2.4GHz network with WPA3/WPA2 compatibility mode to maintain support for legacy devices.

The recommendation to use a complex password also applies here, as it remains the most effective method for mitigating brute-force attacks. A strong password should be longer than 8 characters (24+ is ideal), and include uppercase and lowercase letters, numbers, and special characters. It is also recommended to avoid using guessable or commonly used passwords that may appear in wordlists.

8.3. Study Limitations

WPS brute-force testing was excluded due to device-side lockout mechanisms. On the tested routers, the WPS PIN interface became unresponsive for several days after as few as three failed attempts, making realistic benchmarking of PIN attacks impractical.

WPA3 downgrade attacks were only evaluated under controlled lab conditions. Real-world feasibility depends on specific router settings (e.g., WPA2 compatibility mode), device behavior, and environmental factors such as channel overlap or roaming aggressiveness.

For CPU-based cracking performance, only a subset of architectural configurations was tested, with emphasis on AMD processors. Results may not directly extrapolate to Intel CPUs of similar clock speed or core count due to differences in instruction sets, caching strategies, and power management.

9. Conclusions

This study evaluated the efficiency of brute-force and dictionary attacks against WPA2 and WPA3 wireless security protocols using various tools and system configurations. The experimental results confirmed that CPU clock speed has a significant impact on brute-force performance, while RAM size and memory channel configuration have negligible effect. Statistical analysis using one-way ANOVA supported these findings and confirmed the dominant role of CPU frequency.

In tool comparisons, Hashcat consistently demonstrated the highest cracking speed and lowest execution time, especially at higher CPU frequencies. Additionally, it was shown that CPU architecture plays a critical role, as newer generations (e.g., Zen 4) significantly outperformed older ones (e.g., Zen 2 or Silvermont), even at identical clock speeds. A strong correlation ($r = 0.96$) was observed between PassMark CPU scores and actual performance in Hashcat.

The study also confirmed that GPU acceleration provides a substantial performance boost, particularly when using discrete GPUs. The RX 5700 XT, for instance, outperformed integrated graphics by over 900%. When using two GPUs simultaneously, additional speed gains were observed, although combining a CPU with dual GPUs did not improve performance and in some cases led to slight degradation. Thermal measurements also showed that GPUs operate significantly cooler than CPUs during cracking, which may benefit stability during extended attacks. GPU backend choice (CUDA vs. OpenCL) also had a small but measurable impact on performance, with OpenCL outperforming CUDA by ~4.7% on GTX 1650 Ti.

While RAM amount had negligible influence on brute-force speed, memory usage increased with larger dictionaries and more powerful cracking devices. However, even

between the smallest and largest tested wordlists (a 21× size difference), memory usage differed by only ~5%, indicating efficient memory management in Hashcat. Devices with higher cracking throughput exhibited proportionally higher RAM usage, suggesting a scaling relationship based on processing rate rather than dictionary size alone.

The findings also highlighted that WPS remains a critical vulnerability, especially on systems where lockout mechanisms are misconfigured or ineffective. Tools like Wifite and Airgeddon proved to be the most versatile for WPS-based attacks due to their support for multiple techniques and automation.

Furthermore, the analysis of WPA3 showed that while the protocol is highly resistant to direct brute-force and dictionary attacks, the presence of compatibility mode with WPA2 opens the door to downgrade attacks. Successful execution of such attacks depends on several factors, including user behavior and router configuration.

These results highlight the importance of secure configuration practices, including disabling WPS, avoiding compatibility modes, and enforcing strong password policies. This study contributes practical guidance for penetration testers and security professionals, as well as recommendations for users seeking to strengthen the security posture of personal or corporate Wi-Fi networks.

Aknowledgments

I am extremely grateful to Dr. Kęstutis Driaunys (ORCID 0000-0002-8456-123X) for reviewing and validating this research. His advice and recommendations significantly enhanced the novelty and overall quality of this article.

I also wish to express my heartfelt gratitude to my mother and grandmother for their unwavering support and invaluable assistance throughout the preparation of this work. Their encouragement and insightful ideas were instrumental in shaping the final outcome.

Furthermore, I sincerely thank the Baltic Journal of Modern Computing (BJMC) for the opportunity to present my research in their esteemed publication.

Special thanks are extended to the Editorial Office and anonymous reviewers of BJMC for their constructive feedback, which helped to refine the article and improve its scientific rigor.

List of Abbreviations

AP	Access Point
URL	Uniform Resource Locator
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access
TKIP	Temporal Key Integrity Protocol
AES	Advanced Encryption Standard
SSL	Secure Sockets Layer
SAE	Simultaneous Authentication of Equals
RAM	Random Access Memory
CPU	Central Processing Unit
GPU	Graphics Processing Unit

HDD	Hard Disk Drive
SSD	Solid-State Drive
OS	Operating System
BSSID	Basic Service Set Identifier
GUI	Graphical User Interface
CLI	Command Line Interface
iGPU	Integrated Graphics Processing Unit
dGPU	Discrete Graphics Processing Unit
CUDA	Compute Unified Device Architecture

References

Abasi-amefon O., A., Matulevicius, R., Nolte, A. (2020). Security Risk Management in E-commerce Systems: A Threat-driven Approach. *Baltic Journal of Modern Computing, 8(2).* https://doi.org/10.22364/bjmc.2020.8.2.02

Appel, M.L., Guenther, S. (2020). WPA 3 - Improvements over WPA 2 or broken again? *Computer Science, Engineering*. https://www.semanticscholar.org/paper/WPA-3-Improvements-over-WPA-2-or-broken-again-Appel-Guenther/934dceb22eb034d615ba34203c6208c641f36b90

Barybin, O., Zaitseva, E., Brazhnyi, V. (2019). Testing the Security ESP32 Internet of Things Devices. *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*. https://doi.org/10.1109/picst47496.2019.9061269

Bosnjak, L., Sres, J., Brumen, B. (2018). Brute-force and dictionary attack on hashed real-world passwords. *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).* https://doi.org/10.23919/mipro.2018.8400211

Bruzgė, R., Černevičienė, J., Šapkauskienė, A., Mačerinskienė, A., Saulius Masteika, Kęstutis Driaunys. (2023). Stylized Facts, Volatility Dynamics and Risk Measures of Cryptocurrencies. *Journal of Business Economics and Management*, 24(3), 527–550. https://doi.org/10.3846/jbem.2023.19118

Chalyi, O., Driaunys, K., Rudžionis, V. (2025). Assessing Browser Security: A Detailed Study Based on CVE Metrics. *Future Internet, 17(3), 104–104*. https://doi.org/10.3390/fi17030104

Chalyi, O., Kolomytsev, M. (2023). Comparison of Tools for Web-Application Brute Forcing. *Theoretical and Applied Cybersecurity, 4(1)*. https://doi.org/10.20535/tacs.2664-29132022.1.274117

Chalyi, O., Stopochkina, I. (2024). Information Retrieval and Deanonymization in the Tasks of Early Detection of Potential Attacks on Critical Infrastructure. *Cybersecurity Education Science Technique, 2(26), 305–322*. https://doi.org/10.28925/2663-4023.2024.26.694

Dondyk, E., Zou, C. C. (2013). Denial of convenience attack to smartphones using a fake Wi-Fi access point. *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*. https://doi.org/10.1109/ccnc.2013.6488441

Fatihah Wan Mustapha, W. N., Abdul Aziz, M. A., Masrie, M., Sam, R., Tan, M. N. M. (2020). WiFi Approximated Strength Measurement Method With Brute Force Algorithm for a Minimum Number of AP and Maximum WiFi Coverage, *2020 IEEE 10th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, Malaysia, 2020, 180-185. https://doi.org/10.1109/ISCAIE47305.2020.9108833

Ferriz, O. I. (2013). Router security configuration towards LAN networks. *Upc.edu*. http://hdl.handle.net/2099.1/19299

Hastings, J., Laverty, D., Morrow, D. J. (2013). A smart grid information system for demand side participation: Remote control of domestic appliances to balance demand. *2013 48th International Universities' Power Engineering Conference (UPEC), 1–5.* https://doi.org/10.1109/upec.2013.6714949

Jain, S., Pruthi, S., Yadav, V., Sharma, K. (2022, March 1). Penetration Testing of Wireless EncryptionProtocols. *IEEE Xplore*. https://doi.org/10.1109/ICCMC53470.2022.9754042

Jared, A. (2011). Vulnerability Note VU#723755 - WiFi Protected Setup PIN brute force vulnerability. *Vulnerability Notes Database.*

Jean-Philippe, A. (2024). Serious Cryptography: A Practical Introduction to Modern Encryption, 2nd Edition. *NO STARCH PRESS, INC*.

Jewell, M. O., Costanza, E., Kittley-Davies, J. (2015). Connecting the things to the internet. *EPrints Soton (University of Southampton)*. https://doi.org/10.1145/2750858.2807535

Jiang, L., Garuba, M. (2008). Encryption as an Effective Tool in Reducing Wireless LAN Vulnerabilities. *Fifth International Conference on Information Technology: New Generations.* https://doi.org/10.1109/itng.2008.221

Kanagachidambaresan, G. R. (2022). Wireless Connectivity in IoT. *Apress EBooks*, 163–185. https://doi.org/10.1007/978-1-4842-8108-6_4

Kolev, K., Yordan S. (2024). WIRELESS SECURITY ISSUES. *Vide. Tehnologija. Resursi/Environment. Technology. Resources*, 4, 150–154. https://doi.org/10.17770/etr2024vol4.8186

Moen, V., Raddum, H., Hole, K. J. (2004). Weaknesses in the temporal key hash of WPA. ACM SIGMOBILE Mobile Computing and Communications Review, 8(2), 76–83. https://doi.org/10.1145/997122.997132

Moissinac, K., Ramos, D., Rendon, G., Elleithy, A. (2021, January 1). Wireless Encryption and WPA2 Weaknesses. *IEEE Xplore*. https://doi.org/10.1109/CCWC51732.2021.9376023

Moroz, V.R., (2024). Comparative analysis of the security of wireless networks with WEP, WPA, WPA2, WPA3 security standards. *Sumy State University.* http://essuir.sumdu.edu.ua:8080/handle/123456789/96660?locale=en

Nabaa Alaa, Al-Shareefi, F. (2024). A Comparative Study Between Two Cybersecurity Attacks: Brute Force and Dictionary Attacks. *Journal of Kufa for Mathematics and Computer, 11(2), 133–139.* https://doi.org/10.31642/jokmc/2018/110216

Narayanan, A., Shmatikov, V. (2005). Fast dictionary attacks on passwords using time-space tradeoff. *Proceedings of the 12th ACM Conference on Computer and Communications Security - CCS '05*. https://doi.org/10.1145/1102120.1102168

Niemietz, M., Schwenk, J. (2015). Owning Your Home Network: Router Security Revisited. *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1506.04112

Nur, W., Aziz, A., Marianah Masrie, Sam, R., Mohd. (2020). WiFi Approximated Strength Measurement Method With Brute Force Algorithm for a Minimum Number of AP and Maximum WiFi Coverage. *IEEE 10th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 180–185. https://doi.org/10.1109/iscaie47305.2020.9108833

Pahlavan, K., Krishnamurthy, P. (2020). Evolution and Impact of Wi-Fi Technology and Applications: A Historical Perspective. *International Journal of Wireless Information Networks, 28.* https://doi.org/10.1007/s10776-020-00501-8

Por, L., Y., Ng, I., O., Chen, Y., -L., Yang, J., Ku., C., S. (2024). A Systematic Literature Review on the Security Attacks and Countermeasures Used in Graphical Passwords. *IEEE Access*. https://doi.org/10.1109/access.2024.3373662

Potter, B., Fleck, B. (2003). 802.11 Security. *O'reilly*.

Prodani, F., Rista, A. (2024). Analyzing Password Security in Wi-Fi Networks Using The Airgeddon Script. *2024 International Workshop on Quantum & Biomedical Applications, Technologies, and Sensors (Q-BATS)*, 93–98. https://doi.org/10.1109/q-bats63267.2024.10873935

Sadeghian, A. (2013). Analysis of WPS Security in Wireless Access Points. *ResearchGate*. https://doi.org/10.13140/2.1.1869.2164

Sagers, G. (2021, December 1). WPA3: The Greatest Security Protocol That May Never Be. *IEEE Xplore*. https://doi.org/10.1109/CSCI54926.2021.00273

Ye, J., De, X., Zhao, L., Zhang, M., Wu, L., Zhang, W. (2024). Exposed by Default: A Security Analysis of Home Router Default Settings. *ASIA CCS '24.* https://doi.org/10.1145/3634737.3637671

WEB (a). ZerBea. (2023, July 18). hcxdumptool. GitHub. https://github.com/ZerBea/hcxdumptool

WEB (b). fluxion. (n.d.). Fluxion. https://fluxionnetwork.github.io/fluxion/

WEB (c). Openwall. (2020, September 25). openwall/john. GitHub. https://github.com/openwall/john

WEB (d). Cowpatty | Kali Linux Tools. (n.d.). Kali Linux. https://www.kali.org/tools/Cowpatty/

WEB (e). Hashcat/Hashcat. (2020, November 9). GitHub. https://github.com/Hashcat/Hashcat

WEB (f). Hashcat hcxpcapngtool - advanced password recovery. (n.d.). Hashcat.net. https://Hashcat.net/cap2Hashcat/

WEB (g). v1s1t0r. (2021, December 12). airgeddon. GitHub. https://github.com/v1s1t0r1sh3r3/airgeddon

WEB (h). wifite | Kali Linux Tools. (n.d.). Kali Linux. https://www.kali.org/tools/wifite/

WEB (i). wiire-a. (2017, December 26). FAQ. GitHub. https://github.com/wiire-a/pixiewps/wiki/FAQ

WEB (j). t6x. (2017, June 28). Introducing a new way to crack WPS: Option p with an Arbitrary String. GitHub. https://github.com/t6x/reaver-wps-fork-t6x/wiki/Introducing-a-new-way-to-crack-WPS:-Option--p-with-an-Arbitrary-String

WEB (k). bully | Kali Linux Tools. (n.d.). Kali Linux. https://www.kali.org/tools/bully/

WEB (l). reaver | Kali Linux Tools. (n.d.). Kali Linux. https://www.kali.org/tools/reaver/

WEB (m). WPA2 vs WPA3 (Full 2024 Comparison & Differences). (2024, June 12). https://www.stationx.net/wpa2-vs-wpa3/

WEB (n). WPA3 Downgrade attack. (2022, August 14). Netprojnetworks.com. https://wwJuly 31, 2025w.netprojnetworks.com/wpa3-downgrade-attack/

Received June 23, 2025, accepted July 31, 2025