Lightweight log-monitoring-based mitigation tool against WLAN attacks

Ramzi Saifan[1], Mohammad Radi[2], Hamsa Al-Dabbagh[2], Badr Mansour[2]
[1]Department of Computer Engineering, Faculty of Engineering, University of Jordan, Amman, Jordan
[2]Department of Computer Engineering, Princess Sumaya University for Technology, Amman, Jordan

Article Info

Article history:

Received Jan 29, 2024
Revised Mar 25, 2024
Accepted Mar 31, 2024

Keywords:

ARP poisoning
Defense tool
DHCP starvation
Log monitoring
Wi-Fi deauthentication

ABSTRACT

Wireless network attacks are some of the most common network security threats dealt with daily. Their ease of execution and effectiveness make them commonplace within most public networks. The goal of this paper is to develop a tool which provides defenses against these attacks, one which can also generate the attacks to test its own effectiveness in defending against them. The research involved the design, testing, and implementation of attacks/defenses tool, which benefits from a user-friendly user interface that simplifies the testing process. The attacks were generated using existing tools, linked to one central interface. The defense methodology was script-based and created entirely from scratch. It was also linked to a single interface which continuously monitors logs to detect and prevent attacks in an efficient timely manner. The results showed that the proposed defenses to the studied wireless attacks were effective at mitigation, or outright prevention. They were also more lightweight than existing solutions, making them more appealing for less powerful hardware.

Corresponding Author:

Ramzi Saifan
Department of Computer Engineering, Faculty of Engineering, University of Jordan
Amman, Jordan
Email: r.saifan@ju.edu.jo

1. INTRODUCTION

Wireless local area networks (WLANs) have become an integral part of our modern communication infrastructure, providing seamless connectivity and added convenience [1], [2]. WLANs are networks that utilize the air as their transmission medium. Hosts on these networks typically communicate with an access point (AP) which is connected to the Internet allowing the delivery of information back and forth. The advantages of WLANs are numerous, including ease of installation where there is no need to install a cable to every connected station, affordability as the access point is not expensive and wire costs are negligible, increased productivity because users can move freely while staying connected to the internet, and reasonable data rates which can reach up to 1 Gbps (with the release of recent standards).

However, the proliferation of wireless technology has also introduced new security challenges, as wireless networks are susceptible to various types of attacks that can compromise their integrity and availability [3], [4]. These security challenges stem from various factors: firstly, the wireless communication medium functions as a broadcast medium, allowing any station within proximity to intercept traffic from other stations. Secondly, early wireless networks had weak security protocols that have improved over time, yet many devices still employ outdated technologies with vulnerabilities. Thirdly, individuals often connect to available wireless networks without verifying their authenticity, especially when in unfamiliar or travel settings.

The primary wireless network technology is Wi-Fi, utilizing radio waves, with frequencies such as 2.4 GHz and 5 GHz. Each of these frequencies is presenting advantages and disadvantages [5], [6]. Wi-Fi networks typically feature an access point to provide wireless connectivity to other stations, forming a basic service set (BSS) that constitutes a wireless local area network.

This paper delves into WLAN security, more specifically, we develop defense tools aimed at safeguarding WLAN against three prominent attacks: dynamic host configuration protocol (DHCP) starvation attack [7], address resolution protocol (ARP) poisoning attack [8], and Wi-Fi deauthentication attack [9]. These attacks represent significant threats to WLAN security and have the potential to disrupt communication, compromise data confidentiality, and enable bad actors to gain unauthorized access.

Here is a brief about each protocol and attack: first, DHCP dynamically assigns IP addresses to hosts, while the starvation attack is when an attacker exhausts the available IP addresses on a DHCP server by requesting excessive leases. On the other hand, ARP maps IP addresses to media access control (MAC) addresses, while the poisoning attack is when an attacker sends falsified ARP messages to associate their own MAC address with the IP address of another device. Lastly, the deauthentication attack is when an attacker sends deauthentication frames to disconnect devices from a Wi-Fi network, disrupting their connectivity. Current research on DHCP starvation attacks in WLANs faces several drawbacks. Firstly, many studies primarily focus on traditional mitigation techniques, such as rate limiting or DHCP snooping, which may not adequately address sophisticated attack variants or evolving attacker strategies [10]. Additionally, most of the research tends to overlook the potential impact of these attacks on real-world networks, often relying on simulated environments that may not accurately replicate complex network dynamics [11].

Research on ARP spoofing attacks against WLANs encounters several limitations. Primarily, many studies focus on conventional countermeasures, such as static ARP table entries or ARP cache validation, which may not effectively mitigate sophisticated ARP spoofing techniques employed by attackers [12]. Furthermore, most of the research often relies on simplistic network topologies or simulated environments, which may not accurately reflect the complexities of real-world WLAN environments [13].

Research on Wi-Fi deauthentication attacks faces several limitations. Firstly, many studies primarily focus on conventional mitigation techniques, such as implementing intrusion detection systems or deploying wireless intrusion prevention systems, which may not effectively counter sophisticated deauthentication attack strategies [14]. Additionally, most of the research often relies on simplified network models or simulated environments, which may not accurately represent the complexities of real-world Wi-Fi networks [9].

Addressing these drawbacks is crucial for advancing WLANs and enhancing their resilience against attacks. Despite their long-standing presence, ongoing research continues to explore these vulnerabilities. Consequently, modern solutions coexist with effective traditional methods. This paper aims to design and implement a GUI-based WLAN defense/attack tool. It will apply defenses to shield the network and generate attacks for testing. The code is open source, facilitating customization by companies and enhancing usability. Additionally, it offers an extra layer of protection for regular users against network and device vulnerabilities. Detection of attacks relies on a novel method, monitoring DHCP server logs and router activity logs via firmware. The proposed tool efficiently detects and resolves attacks, validated through deployment and testing in a real environment rather than a simulator.

The remainder of the paper is organized as follows: section 2 reviews current research on mitigating these challenges. In section 3 introduces our comprehensive defense design. In section 4 presents tests assessing the performance of our defense tools under different conditions, offering insights into their effectiveness and limitations.

2. RELATED WORK

In this section, several related state-of-the-art papers are presented. These papers discuss the attacks we are handling in this project. The attacks are DHCP starvation, ARP poisoning, and Wi-Fi deauthentication.

2.1. DHCP Starvation

To detect this attack, investigators usually resort to monitoring DHCP server logs [15] to narrow down the culprit of the attack. An intrusion detection system (IDS) can be used for such monitoring. Systems like these typically use two types of detection; the first of which is signature-based detection which will assess network traffic based on previously known attack methodology and notify a network administrator when a known attack may be happening on the network. The second type is anomaly detection which is far more complex. It will learn the normal activity to come to expect on the network over time based on legitimate users’ usage and traffic, then will be able to identify when abnormalities arise.

Tripathi and Hubballi [11], the paper delves into a smart anomaly-based IDS specifically intended for monitoring DHCP-related attacks. This system uses training and testing phases, much like a machine learning algorithm, in order to profile expected traffic related to DHCP training, then is fed abnormal cases where a starvation attack is targeting the test DHCP server testing. The testing phase can use one of the many attack generation tools (such as dhcpig [16]).

Tripathi and Hubballi [17], more advanced methods of the attack are discussed. These are targeted towards specific victims on the network as opposed to a network-wide attack that starves the DHCP pool entirely. Such an attack could be used to deny service from specific subnets on a network, those of which could be intended for critical functions. These methods include directly interrupting the discover-offer-request-acknowledge (DORA) message exchange by sending spoofed replies and, as a result, targeting the attack at specific victims (disallowing them to lease addresses). The more advanced detection methods mentioned in [11] are necessary for the detection of some of the newer iterations of the attack from [17].

To prevent attacks, implementing rate limiting is crucial. Huawei [18], rate limiting restricts the rate at which DHCP messages are processed on the server side, minimizing the effectiveness of targeted starvation attacks. This was achieved by implementing rate limiting on the WLAN device, such as a home router functioning as the DHCP server. By monitoring packet rates through each port, suspiciously high volumes trigger independent examination channels, preventing disruption to legitimate traffic and server functionality. While this method protects against server shutdowns, it doesn't prevent address pool starvation. Alternatively, rate limiting at the address level restricts specific MAC addresses from sending further requests if they exceed a preset packet volume threshold, ensuring that no attacks are allowed to occur.

Some of the referenced related work like [19] has tried (and mostly failed) to introduce authentication methods for DHCP message exchanges. Given that DHCP packets are of the UDP type, meaning they are connectionless and have no authentication, any client local to the network can send requests to the DHCP server and be leased an address without checking for legitimacy. This would seemingly be fixed by using an authentication method, but most WLAN implementations that have previously attempted this fell short of fully preventing starvation attacks with this method [19]. As an alternative to the covered existing research, we propose a method involving periodic ping sweeps to check the legitimacy of IP-leases. Accordingly, our tool will perform the necessary changes if suspicious activity is detected.

2.2. ARP poisoning

Normally, detecting ARP spoofing could be tricky as to where to look. Typically, a network analyzer such as “Wireshark” can help as using such programs can give the user alerts if an ARP poisoning attack is detected [20]. Atmojo *et al.* [21], a machine learning model is trained using features such as IP addresses, transmission control protocol (TCP) port, user datagram protocol (UDP) port, length, protocol, and MAC addresses. The model is trained and used to assess the test data and to classify each record if it is a normal activity or is an ARP poisoning attack.

Meghana *et al.* [22], the approach used to prevent ARP poisoning is to have a static ARP table, meaning manually configuring the bindings for each host in the network. This method has many disadvantages, primarily being that most networks are dynamic such that there is a DHCP server in the network and so mappings are not permanent. In addition, this method becomes impractical in larger networks. Another method would involve adding a layer of security using stateful ARP, such that a device would keep distinct records for ARP requests and ARP replies. Once the device receives an ARP reply, it first checks if this reply corresponds to a sent ARP request. If no request was sent, the host would suppose that the ARP reply received was in fact sent by an attacker and so would discard it and not update the ARP table. Despite that this solution would be effective, changing into a stateful ARP requires alteration of the already in use protocol.

Another technique to prevent ARP poisoning, which is discussed in [23], is to implement a client-side checker with a voting system. In the case of ARP tables, updating the system would wait for a random interval from 0 to 100 milliseconds, then sends a voting request for all other systems in the network to get the correct IP and MAC address pair. Upon the voting result, if a MAC address gets more than 50% of the votes it will be accepted, and the ARP table and cache will be updated. In our implementation, we propose incorporating a duplicate detection system to check for (incorrectly) repeated MAC addresses in any host’s ARP table. The tool would take this as a sign of malicious activity occurring during the communication between wireless hosts on the network; thus, potentially a man-in-the-middle (MiTM) attack!

2.3. Wi-Fi deauthentication

Typically, implementing security measures such as strong Wi-Fi network passwords, Wi-Fi protected access version 3 (WPA3) encryption [24], or intrusion detection systems can help mitigate the risk of this attack. Otherwise, an incident response team may need to sift through communication history, using network monitoring tools, to analyze the network traffic searching for unusual patterns.

Latha and Bommi [25], suggested using machine learning-based IDS alongside hardware, software, and upgrades to handle WLAN incursions. Experimental results showed the proposed machine learning-based IDS achieved a high detection capability, with a 96% detection rate for denial-of-service (DoS) attacks and a maximum classifier accuracy of 88.8%. However, utilizing machine learning for deauthentication attacks can be challenging due to their unpredictable nature. Training a model, particularly with numerical features, may yield inconsistent results, reflected in the research's accuracy below 90%, which is suboptimal for mitigating attacks affecting host device availability.

Khasanova [26], has attempted to tackle Wi-Fi attacks by examining the contents of dissociation and deauthentication frames. This approach can prove effective if there is a known signature causing suspicion of a potential attack. It is also useful to detect more dangerous attacks which may target the access point as well as the clients on the WLAN. All in all, this research’s methodology is less than ideal for any kind of mitigation as it involves a lot of sifting through captured traffic to find evidence and is rather reactionary.

The experimental and traditional findings demonstrate the effectiveness of defense strategies used against this attack. Unfortunately, they nearly always rely on a higher level of security or the aid of machine learning techniques, both of which may increase network-setup costs and may not be widely supported just yet. However, monitoring the order of log messages provides a consistent pattern whenever a legitimate device deauthentication happens. Using a series of conditional programming statements can help isolate incidents where bad actors may be involved in a wireless device dropping their connection suddenly.

3. PROPOSED METHOD

Conventional cybersecurity defenses are largely rooted in wired techniques [27], often prove inadequate in mitigating the dynamic nature of wireless attacks. By integrating innovative, purpose-built scripts, and a seamlessly coordinated defense strategy, our approach seeks to strengthen wireless network security against a range of potential threats. For testing purposes, the attacks are generated in a controlled test environment. Moreover, the defense approach uses shell scripts to prevent attacks, overcoming router storage constraints and allowing users to customize parameters. Figure 1 shows the hardware setup of the WLAN test environment. The setup includes the following:

- An attack laptop booted into Kali Linux
- A Kali Linux virtual machine (VM)
- A defense laptop booted into Ubuntu Linux
- A victim Huawei mobile device (running Android 10)
- A router (running OpenWrt firmware)

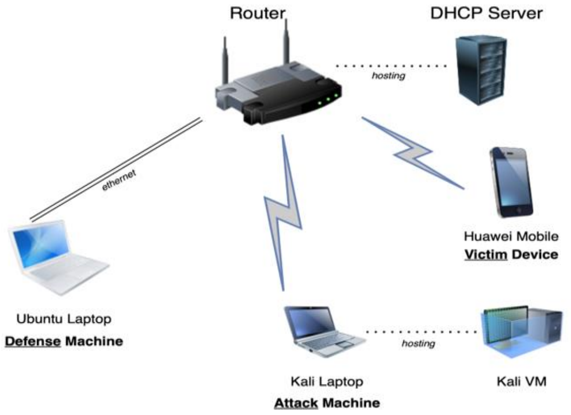


Figure 1. Test environment hardware setup

The Kali Laptop was utilized for ARP poisoning and deauthentication attacks, with a wireless interface card capable of switching to monitor mode. The Kali VM was needed for DHCP starvation, as it needed to randomize MAC addresses for DORA cycles. The defense machine hosted a graphical user interface (GUI) that shows logs, ARP table, and DHCP leases updated periodically. This gives the administrator the flexibility to monitor information about the network. Additionally, the defense machine was connected to the router via an ethernet cable to ensure stability, performance, and security. This was done to prevent deauthentication attacks on the logging system, preventing log loss.

In the proposed network structure, a TP-Link router was used which supports 2.4 GHz and has a custom OpenWrt firmware v18.06. OpenWrt firmware is chosen for its high customization, security, and stability. Moreover, the router had limited flash storage (4 megabytes) and approximately 70 kilobytes were accessible after installing the needed tools. Due to the limited storage, the newest firmware version was incompatible with the router. The firmware had to be Linux-based to run the shell scripts. In addition, it needed to support our desired customization for the logging system.

Logs are sent to an external server using TCP protocol for reliability. The logging server's IP is the same as the defense machine's and is a static lease on the router. The coreutils-nohup package is installed to run commands without hang-ups. Moreover, Rsyslog package was installed to ensure the logging server was up and running. We configured the logging server to run over TCP protocol. The Rsyslog offers high performance when it comes to sending and processing log messages because it utilizes multithreading. In addition, it offers the ability to configure the output format. In summary, the developed design is composed of two tools; attack and defense, which run independently. Both have a GUI which was built using Python with the Tkinter library to be user-friendly.

3.1. Attack suite

The attack design mainly consists of existing tools for performing the attacks in a controlled environment. This tests the defenses' effectiveness. We used several methods to perform the three attacks.

First, the DHCP starvation attack can be generated using many tools such as Hyenae, dhcpstarv, dhcpig, Gobbler, and Yersinia. We moved to a Python script written using the Scapy library found on GitHub [28] to perform the attack because these scripts are available to anyone and could be used unethically. Yet all the tools work on the same basis, which goes through the DORA cycle.

The DHCP starvation script works by sending the full DORA cycle using random MAC addresses to lease as many IP addresses as possible from the pool. The script initiates by generating a random MAC address and sending a DHCP discover packet with the spoofed MAC address. The script then iterates until receiving a DHCP offer packet or reaching a preset attempt limit. In persistent mode, it continuously sends DHCP discover packets if no offers are received; otherwise, it retries a number of times before concluding the attack. Subsequently, it transmits a DHCP request packet with the obtained IP and spoofed MAC address, ultimately executing the DHCP starvation attack by invoking the main function with specified arguments.

Secondly, regarding the ARP Poisoning attack, we had two options; the first is the Arpspoof tool, which is a command line-based tool that was fast and functioning as expected. The second was Ettercap, which has a GUI and functions correctly but is slower than Arpspoof, and we would rather not to deal with an additional, third-party GUI. Therefore, the ARP poisoning attack is orchestrated using Arpspoof to facilitate a MiTM scenario between two devices. Arpspoof tool leverages ARP spoofing, sending falsified ARP messages to the target hosts, thereby linking the attacker's MAC address with the IP address of the intended victim. With this manipulation established, Arpspoof intercepts network traffic between the victim and the router, enabling the attacker to potentially compromising sensitive information or launching further network attacks.

Lastly, for the Wi-Fi deauthentication attack, we may use airplay-ng or MDK4. Airplay-ng is compatible with many devices and easy to use and set up. MDK4 is widely supported, very stable, and reliable, but more challenging to set up and use. Nevertheless, both tools work similarly; by switching the adapter to monitor mode (promiscuous) and sending the packets to the targeted devices. The deauthentication attack was executed using Airodump-ng and Airplay-ng through the following steps. Firstly, monitor mode was initiated with the command "airmon-ng start interface-name". Then, the Airodump-ng tool was employed to scan the wireless medium and gather necessary data. Finally, the deauthentication attack was launched via the Airplay-ng command, targeting the identified basic service set identifier (BSSID) and station, and specifying the number of packets to be transmitted to forcefully break their connection.

3.2. The defense suite

The developed defense design consists of two main components. The first of which is the series of scripts we have written to apply defenses, and the other is the GUI from which the defenses can be applied, and shows recent logs, an ARP table, and DHCP leases updated frequently. The user can choose to enable the defense or not.

The defense approach is a lightweight solution to overcome the router's storage constraints. Specifically, the defense suite consists of scripts to apply defenses as shown in Figure 2, by verifying active IP addresses, preventing duplicate connections, and detecting and mitigating Wi-Fi deauthentication attacks. These scripts provide robust defense mechanisms against various network threats, contributing to overall network protection. More precisely, the defense tool uses five shell scripts executed using the /bin/sh interpreter. Three of them are attack-specific and two for generating and setting random MAC addresses.

The first script checks the DHCP lease-file for active IP addresses to test lease legitimacy. If a host is active and is currently using an IP address, it responds to echo request with an echo reply. If the host is inactive, it is assumed to be a false lease. Consequently, the host's address is reported and removed from the DHCP server pool. Figure 2(a) shows a detailed flowchart of the defense script for DHCP starvation.

The second script defends against ARP poisoning. It works by monitoring exchanged messages for any possible duplicate MAC addresses. If it detects duplicate MACs, it blocks that MAC from connecting to the AP. This modifies the wireless configuration to deny connections as the duplicate MAC address represents the attacker's MAC address.

The third script scans logs for Wi-Fi deauthentication events. The script continuously scans the logs for any "disconnect" message, which may be a possible attack. Particularly, an attack is detected if a certain pattern of logs appears which is the existence of a disconnect message that is not followed by a dis-association message. If an attack is detected, it generates a random MAC address, updates wireless configuration, and restarts the interface to prevent progress.

Changing the network's BSSID to a new randomly generated MAC address will nullify the attacker's attempt to deauthenticate WLAN hosts since the attack involves targeting a specific BSSID. A side effect of the script described is a very minor drop in connection for legitimate users while the BSSID is updated. However, from our testing, this drop is negligible and does not hurt availability for clients. This means that despite a minor lag that some may experience, their access to the network and anything they may be doing via the internet will not be negatively impacted. Figure 2(b) shows a flowchart of the defense script for Wi-Fi deauthentication.

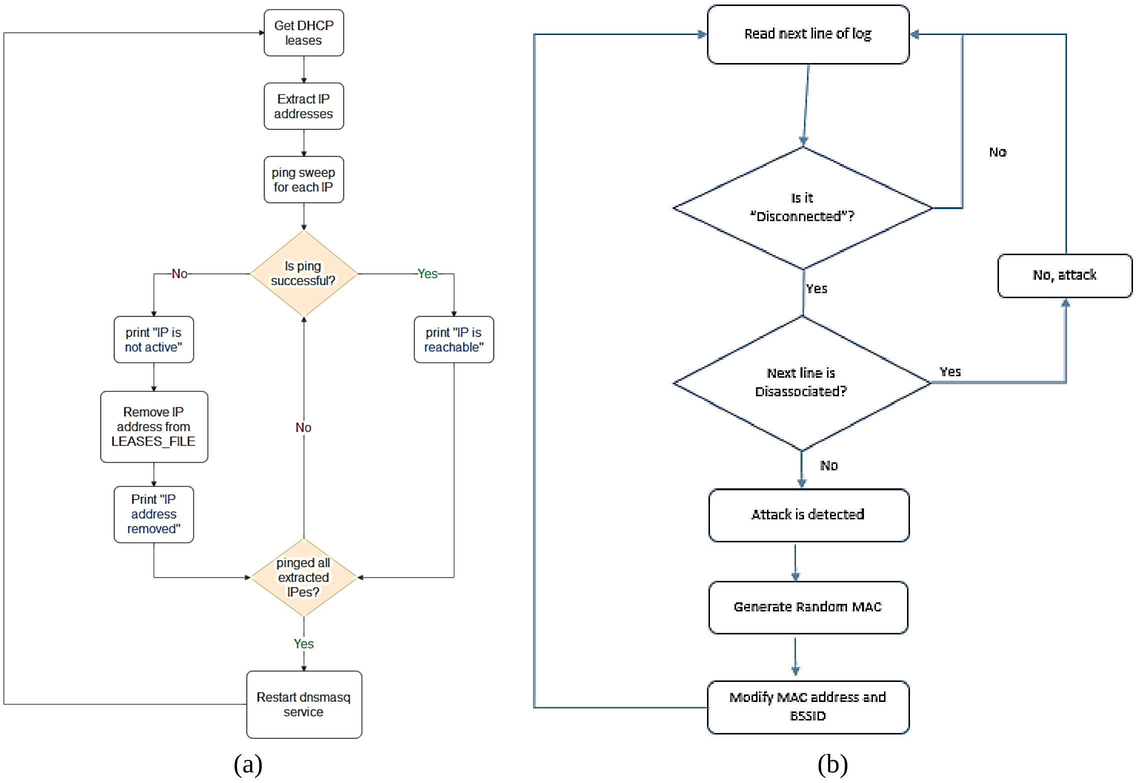


Figure 2. Proposed defenses flowcharts (a) DHCP starvation defense script and (b) Wi-Fi deauthentication defense script's

4. RESULTS AND DISCUSSION

In this section, we will show the results by displaying screen shots for scenarios before and after applying the defense scripts. First, Figure 3 shows the attack suite GUI interface. We show how to generate each attach, and how to defend against it.

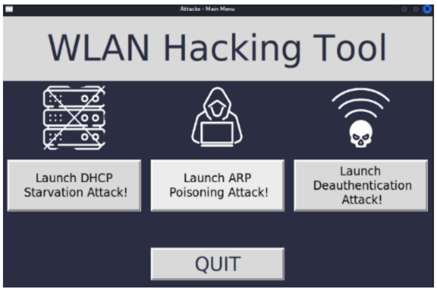
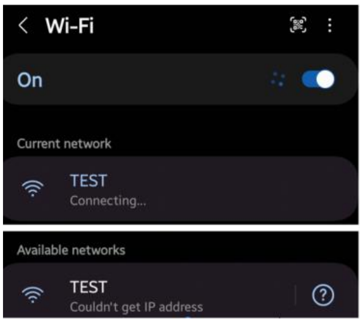

Attacks - Main Menu
WLAN Hacking Tool
Launch DHCP Starvation Attack!
Launch ARP Poisoning Attack!
Launch Deauthentication Attack!
QUIT

Figure 3. Attack main menu

4.1. Testing DHCP starvation defense

In order to set up the generation of the DHCP starvation attack, we needed to use the Kali VM. By default, the attack script would send a specific number of DHCP requests. To test the attack in this initial scenario, we reduced the DHCP pool size to accommodate the limited duration of the attack. After confirming that it was functioning as expected, we used the persistent mode of the script in order to keep the attack running indefinitely. This would result in the starvation of the entire pool, regardless of size, which would eventually be eventually successful.

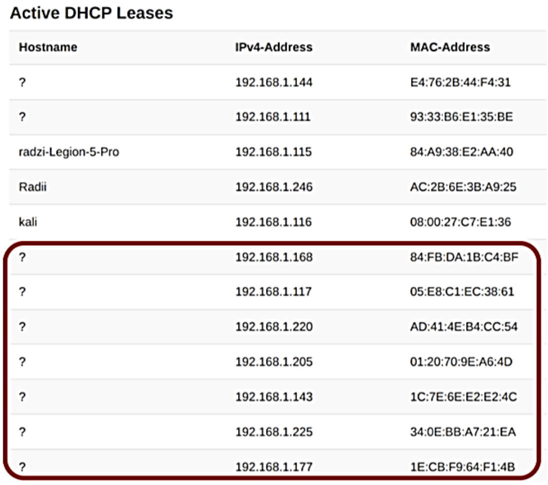
First, without defenses applied, the network was affected by the starvation attack quickly (in a matter of seconds), where it lost the ability to lease IP addresses to legitimate hosts. Had the pool size been bigger, the time needed for the attacker to complete starvation would have been slightly longer. As can be seen in Figure 4, the victim device attempting to connect to the Wi-Fi SSID "TEST" (the test network) would be unable to get past the "obtaining IP address" stage as the pool was all used up by the attacker's leases as shown in Figure 5.



TEST
Connecting...

TEST
Couldn't get IP address

Figure 4. Legitimate host unable to lease an IP address

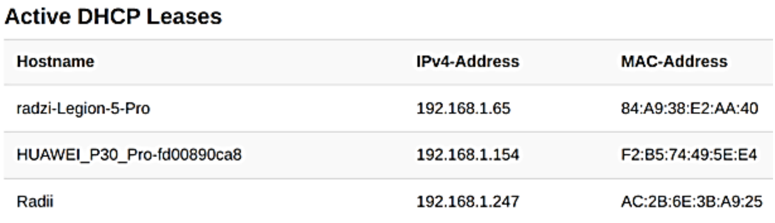


Active DHCP Leases

Hostname	IPv4-Address	MAC-Address
?	192.168.1.144	E4:76:2B:44:F4:31
?	192.168.1.111	93:33:B6:E1:35:BE
radzi-Legion-5-Pro	192.168.1.115	84:A9:38:E2:AA:40
Radii	192.168.1.246	AC:2B:6E:3B:A9:25
kali	192.168.1.116	08:00:27:C7:E1:36
?	192.168.1.168	84:FB:DA:1B:C4:BF
?	192.168.1.117	05:E8:C1:EC:38:61
?	192.168.1.220	AD:41:4E:B4:CC:54
?	192.168.1.205	01:20:70:9E:A6:4D
?	192.168.1.143	1C:7E:6E:E2:E2:4C
?	192.168.1.225	34:0E:BB:A7:21:EA
?	192.168.1.177	1E:CB:F9:64:F1:4B

Figure 5. Fully leased DHCP pool

If a successful DHCP starvation attack had already happened, and the defense was reactively applied, the network would recover as follows: first, the defense machine would perform ping sweeps at the interval already specified in the defense script. Secondly, the maliciously leased IP addresses wouldn't all be able to respond and the defense machine would, as a result, remove these from the lease file as shown in Figure 6. This would then allow legitimate hosts on the network to successfully lease IP addresses once again.



Active DHCP Leases

Hostname	IPv4-Address	MAC-Address
radzi-Legion-5-Pro	192.168.1.65	84:A9:38:E2:AA:40
HUAWEI_P30_Pro-fd00890ca8	192.168.1.154	F2:B5:74:49:5E:E4
Radii	192.168.1.247	AC:2B:6E:3B:A9:25

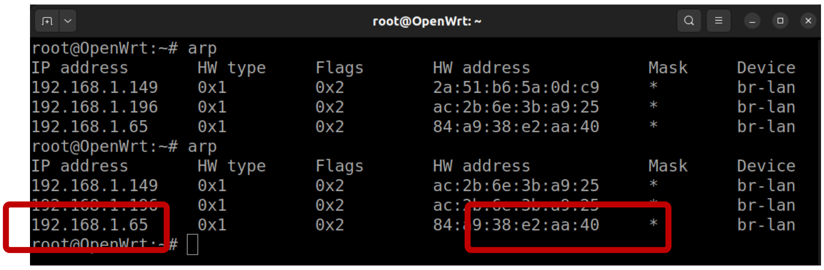
Figure 6. Recovered DHCP pool (functioning normally)

If the defense machine had the defense active, and the attack happened to occur at the same time, the malicious leases would not be effective at starving the pool. This is because the periodic pings to leased addresses would stay right on top of the attacker's leases and remove them from the lease file; resulting in the DHCP service's availability remaining intact. While the defense is applied, the attacker's leases would be recognized as malicious and removed from the lease file immediately.

4.2. Testing ARP poisoning defense

To generate the ARP poisoning MiTM attack, we used the Arpspoof tool. We tested on a Windows laptop first, but the security features of Windows 11 blocked the attacks as it would re-send an ARP request even after receiving a spoofed ARP reply. To get past this, we opted to use a device running an Android security patch level. This device fell victim to the fake ARP replies. At this point, the ARP tables of the router and Android device were successfully poisoned. We confirmed this as the Android device had lost connectivity to the router. With no ARP poisoning defense applied, an attacker can poison the ARP table of both the router and a victim. Consequently, the attacker can play MiTM between the router and the victim. By sending ARP reply packets to both devices, the changes would be accepted and saved to the ARP tables.

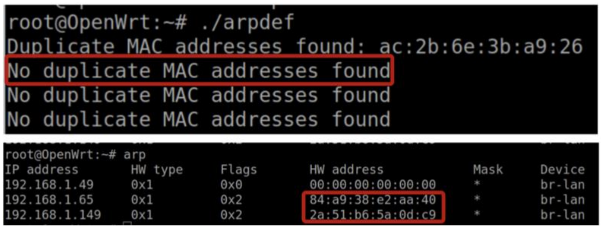
Figure 7 shows running the "arp" command before and after the attack is performed. We can see that the router's ARP table has changed after the attack and now there are two IP addresses with the same MAC address; one is the MAC address of the attacker and the other is for the victim (wrong pair with the IP). It is also noticed that the victim is unable to communicate with the router as the packet is sent to the router's IP, but on layer 2 level any packet would be destined to the attacker as it would have the MAC address of the attacker.



```
root@OpenWrt:~# arp
IP address       HW type     Flags       HW address            Mask     Device
192.168.1.149    0x1         0x2         2a:51:b6:5a:0d:c9     *        br-lan
192.168.1.196    0x1         0x2         ac:2b:6e:3b:a9:25     *        br-lan
192.168.1.65     0x1         0x2         84:a9:38:e2:aa:40     *        br-lan
root@OpenWrt:~# arp
IP address       HW type     Flags       HW address            Mask     Device
192.168.1.149    0x1         0x2         ac:2b:6e:3b:a9:25     *        br-lan
192.168.1.196    0x1         0x2         ac:2b:6e:3b:a9:25     *        br-lan
192.168.1.65     0x1         0x2         84:a9:38:e2:aa:40     *        br-lan
root@OpenWrt:~# 
```

Figure 7. Router's poisoned ARP table (contains duplicate MACs)

After having the attack done and the ARP tables poisoned, we ran the defense script. The defense script detects the poisoned ARP table as shown in Figure 8 top part. Therefore, it extracts the duplicate MAC address and adds it to the filtered MAC address list with the denied entry. It will also restart the network service to clear the ARP table so that it can be filled again with the correct MAC addresses; Figure 8 (lower part) shows this recovered ARP table. On the other hand, having the defense up and running all the time and continuously monitoring the router’s ARP table would rapidly isolate the attacker and deny them access. In such a case, the poisoned ARP table will be restored promptly with a minor delay in the next communicated packet.



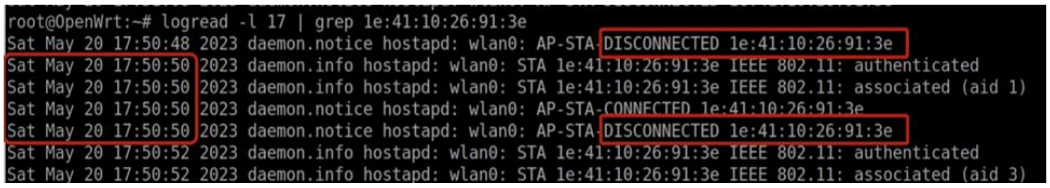
```
root@OpenWrt:~# ./arpdef
Duplicate MAC addresses found: ac:2b:6e:3b:a9:26
No duplicate MAC addresses found
No duplicate MAC addresses found
No duplicate MAC addresses found
```

```
root@OpenWrt:~# arp
IP address       HW type     Flags       HW address            Mask     Device
192.168.1.49     0x1         0x0         00:00:00:00:00:00     *        br-lan
192.168.1.65     0x1         0x2         84:a9:38:e2:aa:40     *        br-lan
192.168.1.149    0x1         0x2         2a:51:b6:5a:0d:c9     *        br-lan
```

Figure 8. Detection of attack and recovery of ARP table

4.3. Testing Wi-Fi deauthentication defense

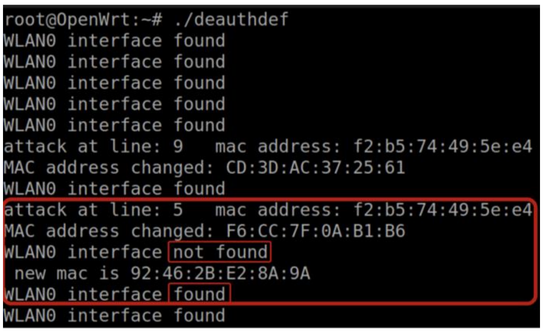
To test the deauthentication attack, we will first show what happens without the defense applied. The attacker successfully deauthenticates the victim, which leads to disruption of network connectivity. By forcing the victim’s device to disconnect from the network, the user will lose the ability to access the internet, communicate with other devices, or utilize network-dependent services. When the attack was successful, as we can see in Figure 9, the logs showed that the victim had a “disconnected” log from the hostapd service without “disassociated” and “deauthenticated” logs. In addition, if the victim attempts to reconnect while the attack is in progress, they will be deauthenticated once again and lose connectivity almost immediately.



```
root@OpenWrt:~# logread -l 17 | grep 1e:41:10:26:91:3e
Sat May 20 17:50:48 2023 daemon.notice hostapd: wlan0: AP-STA-DISCONNECTED 1e:41:10:26:91:3e
Sat May 20 17:50:50 2023 daemon.info hostapd: wlan0: STA 1e:41:10:26:91:3e IEEE 802.11: authenticated
Sat May 20 17:50:50 2023 daemon.info hostapd: wlan0: STA 1e:41:10:26:91:3e IEEE 802.11: associated (aid 1)
Sat May 20 17:50:50 2023 daemon.notice hostapd: wlan0: AP-STA-CONNECTED 1e:41:10:26:91:3e
Sat May 20 17:50:50 2023 daemon.notice hostapd: wlan0: AP-STA-DISCONNECTED 1e:41:10:26:91:3e
Sat May 20 17:50:52 2023 daemon.info hostapd: wlan0: STA 1e:41:10:26:91:3e IEEE 802.11: authenticated
Sat May 20 17:50:52 2023 daemon.info hostapd: wlan0: STA 1e:41:10:26:91:3e IEEE 802.11: associated (aid 3)
```

Figure 9. Logs indicating deauthentication attack

The next test was done after the occurrence of an attack, and the victim was not able to reconnect to the WLAN. Here we enabled the deauthentication defense, which will generate a random MAC address and change BSSID to that new random value; the defense script can be seen in action in Figure 10. The time to change BSSID and randomize MAC varies for different attack scenarios, but eventually, in all scenarios, the network is available for all legitimate users, and all of them will be promptly moved to the new BSSID and MAC. This is because the beacon messages are sent every 100 ms. Figure 10 also shows that the interface WLAN0 is initially not found. But, once the script runs, the interface is updated with the newly generated MAC, and the interface is found once again.



```
root@OpenWrt:~# ./deauthdef
WLAN0 interface found
WLAN0 interface found
WLAN0 interface found
WLAN0 interface found
WLAN0 interface found
attack at line: 9   mac address: f2:b5:74:49:5e:e4
MAC address changed: CD:3D:AC:37:25:61
WLAN0 interface found
attack at line: 5   mac address: f2:b5:74:49:5e:e4
MAC address changed: F6:CC:7F:0A:B1:B6
WLAN0 interface not found
 new mac is 92:46:2B:E2:8A:9A
WLAN0 interface found
WLAN0 interface found
```

Figure 10. The running deauthentication defense script

Finally, when performing the attack with the defense active, we can see that the victim will still be disconnected, but as the defense is actively scanning the logs, it will detect the attack instantly, generate a random BSSID and MAC and apply them right away to the router's configuration. This will restore connectivity to the victim soon enough without affecting other legitimate users. So, while the attack may still be going on, it would have zero effect on our router. In fact, it is attacking a BSSID that no longer exists!.

4.4. Discussion

Overall, the tests demonstrated effective defense mechanisms and user-friendly interfaces for executing and monitoring network actions. Table 1 summarizes the effects of generating the attacks while the defenses are applied to logs, legitimate users, attacker, victim, and router. The DHCP defense script performed a ping sweep on all the leases, for which we need to experiment with different ping counts. We started with four echo request/reply pairs; it worked successfully but needed some time to ping every device on the network hence slowing down the performance of the defense; for that reason, we reduced the count to one because all the devices are on our local network and theoretically no packets will be lost. That change succeeded in less time with reduced network traffic. Moreover, removing the leased IP record from the DHCP leases file in the router temporarily removed the lease, but when the devices communicated with the router, the entry would be inserted again. As a result, we needed to remove that record by restarting (committing) the DHCP service. This, however, didn't affect legitimate devices trying to obtain an IP address from the server.

Table 1. Attacks' effect on the network

	Without defenses applied			With defenses applied		
	DHCP starvation	ARP poisoning	Wi-Fi deauthentication	DHCP starvation	ARP poisoning	Wi-Fi deauthentication
Effect on router	DHCP pool is filled with inactive leases	Unable to resolve address with victim host	Normal operation	Normal operation	Network restarted	BSSID changes and network interface is toggled down/up
Effect on logs	The logs are flooded with DORA messages	Duplicate MAC addresses for different IP bindings	Abnormal management frames; "Disconnected" without "Dissociated"	The logs are flooded with DORA messages	Duplicate MAC's access is blocked	Abnormal management frames initially, followed by hostapd service up/down toggle, then normal operation restored
Effect on legitimate users	No available IP leases for legitimate hosts	Unable to resolve address with router (hence, can't reach other networks)	Dropped connection from the network	New hosts can join the network and obtain an IP address normally	Minor network drop	Temporary delay before reconnection
Attacker's end result	Successfully starve the network of IP leases	Successfully poisoned the ARP table and established MiTM connection	Successfully deauthenticate the victim	Their attack failed	Their attack failed	Their attack only succeeded once, then was no longer effective

During testing, we noticed that the ARP table was not being periodically cleared. As a result, we needed an ARP script to restart the network upon completion of each loop iteration to apply its changes if an attack was detected. This restart was critical for the stability of the network after suffering an ARP poisoning attack. It also helped ensuring that there were no residual invalid ARP table entries.

The deauthentication defense script needed a new random MAC address to be applied to the new BSSID. This MAC would be applied to the router's wireless interface to broadcast the BSSID; however, for this change to be applied, we needed to reinitialize the wireless interface, but reinitializing it would sometimes fail. We also tried bringing the interface down and back up again, which also failed sometimes. To ensure that the change would be stable, we concluded that the only way the interface would operate again is by applying a new MAC, which is done by having flags in the script and running a script if needed.

It is worth mentioning that the newest generation of wireless security protocols, known as WPA3, is intended to increase the security of Wi-Fi networks. Although WPA3 primarily focuses on enhancing encryption and authentication processes, it does provide some features that are able to mitigate the studied attacks. However, not all devices or routers support WPA3.

5. CONCLUSION

In this paper, we have developed and tested defense mechanisms against three critical wireless network attacks: DHCP starvation, ARP poisoning, and Wi-Fi deauthentication. Our study demonstrates the efficiency of our innovative defense strategies in mitigating these threats. Through careful design and scripting, we have successfully mitigated the disruptions caused by these attacks. Our defense mechanisms effectively restore DHCP pool functionality, detect and prevent ARP poisoning, and rapidly restore connectivity after most Wi-Fi deauthentication incidents. As wireless security evolves, our research underscores the importance of adaptive defense mechanisms. By proactively addressing vulnerabilities and deploying agile countermeasures, we contribute to the ongoing effort to ensure secure and reliable wireless network operations. The future work could include the addition of a report generating feature for the defense GUI that contains the current status of the router as well as the latest attacks performed on it recently. This would help provide incident response teams with a clear breakdown of what happened and how it was addressed by the defense tool. It would also be useful in trying to understand & respond to more modern attacks that may occur in the future.

REFERENCES

[1] M. A. M. Albreem, “5G wireless communication systems: vision and challenges,” in *I4CT 2015 - 2015 2nd International Conference on Computer, Communications, and Control Technology, Art Proceeding*, Apr. 2015, pp. 493–497, doi: 10.1109/I4CT.2015.7219627.

[2] G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz, “A survey on 5G networks for the internet of things: communication technologies and challenges,” *IEEE Access*, vol. 6, pp. 3619–3647, 2017, doi: 10.1109/ACCESS.2017.2779844.

[3] Y. Liu, H. H. Chen, and L. Wang, “Physical layer security for next generation wireless networks: theories, technologies, and challenges,” *IEEE Communications Surveys and Tutorials*, vol. 19, no. 1, pp. 347–376, 2017, doi: 10.1109/COMST.2016.2598968.

[4] H. Boland and H. Mousavi, “Security issues of the IEEE 802.11B wireless LAN,” in *Canadian Conference on Electrical and Computer Engineering*, 2004, vol. 1, pp. 0333–0336, doi: 10.1109/ccece.2004.1345023.

[5] D. Duong, Y. Xu, and K. David, “Comparing the performance of Wi-Fi fingerprinting using the 2.4 GHz and 5 GHz signals,” in *IEEE Vehicular Technology Conference*, Jun. 2018, vol. 2018-June, pp. 1–5, doi: 10.1109/VTCSpring.2018.8417848.

[6] I. Dolinska, M. Jakubowski, and A. Masiukiewicz, “Interference comparison in Wi-Fi 2.4 GHz and 5 GHz bands,” in *Proceedings of the International Conference on Information and Digital Technologies, IDT 2017*, Jul. 2017, pp. 106–112, doi: 10.1109/DT.2017.8024280.

[7] N. Hubballi and N. Tripathi, “A closer look into DHCP starvation attack in wireless networks,” *Computers and Security*, vol. 65, pp. 387–404, Mar. 2017, doi: 10.1016/j.cose.2016.10.002.

[8] G. Al Sukkar, R. Saifan, S. Khwaldeh, M. Maqableh, and I. Jafar, “Address resolution protocol (ARP): spoofing attack and proposed defense,” *Communications and Network*, vol. 08, no. 03, pp. 118–130, 2016, doi: 10.4236/cn.2016.83012.

[9] D. Schepers, A. Ranganathan, and M. Vanhoef, “On the robustness of Wi-Fi Deauthentication countermeasures,” in *WiSec 2022 - Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, May 2022, pp. 245–256, doi: 10.1145/3507657.3528548.

[10] H. A. S. Adjei, M. T. Shunhua, G. K. Agordzo, Y. Li, G. Peprah, and E. S. A. Gyarteng, “SSL stripping technique (DHCP snooping and ARP spoofing inspection),” in *International Conference on Advanced Communication Technology, ICACT*, Feb. 2021, vol. 2021-February, pp. 187–193, doi: 10.23919/ICACT51234.2021.9370460.

[11] N. Tripathi and N. Hubballi, “A probabilistic anomaly detection scheme to detect DHCP starvation attacks,” in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems, ANTS 2016*, Nov. 2017, pp. 1–6, doi: 10.1109/ANTS.2016.7947848.

[12] M. Data, “The defense against ARP spoofing attack using semi-static ARP cache table,” in *3rd International Conference on Sustainable Information Engineering and Technology, SIET 2018 - Proceedings*, Nov. 2018, pp. 206–210, doi: 10.1109/SIET.2018.8693155.

[13] J. Bhardwaj, V. K. Yadav, M. C. Trivedi, and A. K. Sen, “ARP cache poisoning: detection, mitigation and prevention schemes,” *International Journal of Computational Vision and Robotics*, vol. 11, no. 4, pp. 357–373, 2021, doi: 10.1504/IJCVR.2021.116547.

[14] S. C. Sethuraman, S. Dhamodara, and V. Vijayakumar, “Intrusion detection system for detecting wireless attacks in IEEE 802.11 networks,” *IET Networks*, vol. 8, no. 4, pp. 219–232, Jul. 2019, doi: 10.1049/iet-net.2018.5050.

[15] Hostiadi, D. Pramana, M. Sudarma, and I. Swamardika, “Server log monitoring based on running services system provider,” *ICSGTEIS*, 2014.

[16] GitHub, *kamorin/DHCPig: DHCP exhaustion script written in python using scapy network library*.

[17] N. Tripathi and N. Hubballi, “Exploiting DHCP server-side IP address conflict detection: a DHCP starvation attack,” in *International Symposium on Advanced Networks and Telecommunication Systems, ANTS*, Dec. 2016, vol. 2016-February, pp. 1–3, doi: 10.1109/ANTS.2015.7413661.

[18] Huawei, “DHCP Security - WLAN Security Hardening Guide.” https://support.huawei.com/enterprise/en/doc/EDOC1100096305/40410024/dhcp-security.

[19] S. Syed, F. Khuhawar, S. Talpur, A. A. Memon, M. A. Luque-Nieto, and S. Narejo, “Analysis of dynamic host control protocol implementation to assess DoS attacks,” in *2022 Global Conference on Wireless and Optical Technologies, GCWOT 2022*, Feb. 2022, pp. 1–7, doi: 10.1109/GCWOT53057.2022.9772887.

[20] R. Kaur, J. Punjab, E. Gurjot Singh, and S. Khurana, “A security approach to prevent ARP poisoning and defensive tools,” *International Journal of Computer and Communication System Engineering (IJCCSE)*, vol. 2, no. 3, pp. 431–437, 2015,

[21] Y. P. Atmojo, I. M. D. Susila, I. B. Suradarma, L. Yuningsih, E. S. Rini, and D. P. Hostiadi, “A new approach for ARP poisoning attack detection based on network traffic analysis,” in *2021 4th International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2021*, Dec. 2021, pp. 18–23, doi: 10.1109/ISRITI54043.2021.9702860.

[22] J. S. Meghana, T. Subashri, and K. R. Vimal, "A survey on ARP cache poisoning and techniques for detection and mitigation," in *2017 4th International Conference on Signal Processing, Communication and Networking, ICSCN 2017*, Mar. 2017, pp. 1–6, doi: 10.1109/ICSCN.2017.8085417.

[23] P. Arote and K. V. Arya, "Detection and prevention against ARP poisoning attack using modified ICMP and voting," in *Proceedings - 1st International Conference on Computational Intelligence and Networks, CINE 2015*, Jan. 2015, pp. 136–141, doi: 10.1109/CINE.2015.34.

[24] G. Sagers, "WPA3: the greatest security protocol that may never be," in *Proceedings - 2021 International Conference on Computational Science and Computational Intelligence, CSCI 2021*, 2021, pp. 1360–1364, doi: 10.1109/CSCI54926.2021.00273.

[25] R. Latha and R. M. Bommi, "Detection of deauthentication threats in Wi-Fi channels using machine learning strategies," in *2022 International Conference on Data Science, Agents and Artificial Intelligence, ICDSAAI 2022*, Dec. 2022, pp. 1–6, doi: 10.1109/ICDSAAI55433.2022.10028874.

[26] A. M. Khasanova, "Detection of attacks on Wi-Fi access points," in *Proceedings of the 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, ElConRus 2021*, Jan. 2021, pp. 28–31, doi: 10.1109/ElConRus51938.2021.9396420.

[27] A. Waqar, "Comparison between wired and wireless network and security risk in wireless network," 2011.

[28] GitHub, "DHCP starvation script by yoelbassin," *GitHub*, 2021. https://github.com/yoelbassin/DHCP-starvation.

BIOGRAPHIES OF AUTHORS

Ramzi Saifan received his B.Sc. and M.S. degrees in Computer Engineering from Jordan University of Science and Technology, Jordan, in 2003 and 2006 respectively, and received his Ph.D. degree in Computer Engineering from Iowa State University in 2012. Since Jan 2013, he joined as a faculty member in the Department of Computer Engineering at the University of Jordan. His current research interests include computer networks, network and information security, cognitive radio networks, machine learning, and image processing. Saifan published several papers in peer reviewed journals and conferences. He can be contacted at email: r.saifan@ju.edu.jo.

Mohammad Radi received his B.Sc. in Network Information and Security Engineering from Princess Sumaya University for Technology, Jordan, in 2023. Enriching his career path in cybersecurity, information security, and compliance fields. His research is devoted to new security solutions, new threats, and new defense methodologies. He has also partnered up with Hamsa Al-Dabbagh and Badr Mansour to publish a paper in machine learning. He can be contacted at email: moh20180187@std.psut.edu.jo.

Hamsa Al-Dabbagh received her B.Sc. in Network Information and Security Engineering from Princess Sumaya University for Technology, Jordan, in 2023. Taking the path of Governance, Risk, and compliance in the cybersecurity field to grow her career path. Her research is centered on modern technologies lacking standards, policies and security frameworks. She has also partnered up with Mohammad Radi and Badr Mansour to publish a paper in machine learning. She can be contacted at email: ham20180206@std.psut.edu.jo.

Badr Mansour received his B.Sc. in Network Information and Security Engineering from Princess Sumaya University for Technology, Jordan, in 2023. Pursuing his career path in information security and governance fields. His research is specialized in mitigating security risks and threats. He has also partnered up with Hamsa Al-Dabbagh and Badr Mansour into publishing paper in machine learning. He can be contacted at email: bad20180009@std.psut.edu.jo.