

# Provably curb man-in-the-middle attack-based ARP spoofing in a local network

Hiba Imad Nasser, Mohammed Abdulridha Hussain

Department of Computer Science, College of Education for Pure Science, University of Basrah, Basrah, Iraq

## Article Info

### Article history:

Received Mar 14, 2022

Revised May 9, 2022

Accepted Jun 20, 2022

### Keywords:

ARP poisoning

ARP spoofing

MITM

MITM sniffing

Network security

## ABSTRACT

Even today, internet users' data security remains a significant concern. One problem is ARP poisoning, otherwise referred to as ARP spoofing. Such attacks are intended to exploit the identified ARP protocol vulnerability. Despite no straightforward remedy for ARP spoofing being apparent, certain actions may be taken to maintain one's safety. The most basic and common defence against a poisoning attack is manually adding MAC and IP addresses to the static ARP cache table. However, this solution is ineffective for large networks where static entries require considerable time and effort to maintain, whether by human input or via special tools and settings for the static entries of network devices. Accordingly, this paper aimed to monitor network packet information and detect the behaviour of ARP poison attacks on operating systems, for instance Windows and Linux. The discovery and defence policy systematically and periodically check the MAC addresses in the ARP table, enabling alerts to be issued if a duplicate entry is detected. This enables the poison-IP address to be blocked before a reply is sent. Finally, the results showed that the superiority was successfully achieved in the detection, prevention and reporting mechanisms in the real-world environment.

This is an open access article under the [CC BY-SA](#) license.



## Corresponding Author:

Hiba Imad Nasser

Department of Computer Science, College of Education for Pure Science, University of Basrah

Basrah, Iraq

E-mail: eduppg.hiba.amad@uobasrah.edu.iq

## 1. INTRODUCTION

The Internet has emerged as a fundamental aspect of daily life. By January 2022, the number of Internet users amounted to 4.88 billion internationally, equating to almost 62% of the global population. This number continues to grow, with almost 257 million new Internet users being added last year; over 700,000 unique users are added daily [1]. The majority of people use it for communication and information exchange. Therefore, security is deemed a foremost threat to computer networks and their applications, requiring action to be initiated in order to safeguard networks and services against illegal activity [2]. Through such malicious attacks, computer systems and technology-dependent businesses are targeted. These assaults primarily involve the computer code twitch, the modification or deletion of data on computer systems, as well as other forms of illegal access. The most prevalent of these types of attacks are cyber-attacks, Man-in-the-middle (MITM), social engineering, replay, as well as denial of service (DoS) attacks. An MITM attack is a cyber-attack through which the attacker intercepts a two-party conversation, mirrors both parties and has access to information provided by both parties [3]. Whenever an MITM attack occurs, a malicious client places his computer in the path of two communicators' transmissions. The use of a packet sniffer afterwards enforces sniffing. To ensure that the communication appears unbroken, the unethical client's computer passes traffic between the unsuspecting clients, as presented in Figure 1.

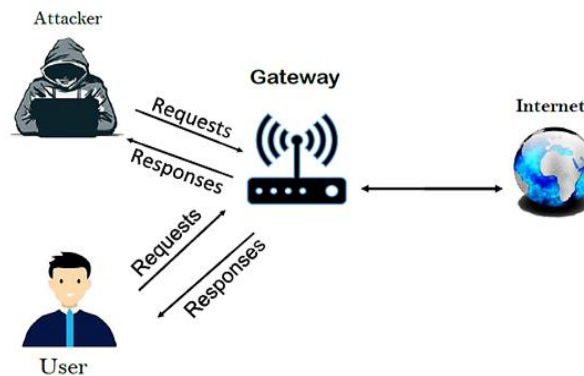


Figure 1. MITM attack technique

MITM attacks enable eavesdropping between people, clients and servers. This may include HTTPS connections to websites, other SSL/TLS connections, Wi-Fi connections and more [4]. Similar to an MITM attack, during a replay attack, the attacker captures data and then repeatedly sends it over the network without modifying it. The original data is valid (usually from an authorised user), meanings its transfer is treated by network security protocols as if it were an original data transfer. When replay attacks are used, hackers do not necessarily need to decrypt them. The attack network's security is similar to standard data transmission. A DoS attack involves blocking legitimate users' access to a network or web resource. Typically, this is performed by flooding the target (usually a web server) with traffic, or by sending malicious requests causing the target resource to break down or become completely unreachable. Social engineering techniques are used by hackers as an initial step to attack a system or network and steal sensitive data, or to distribute and spread malware within the network of an organisation in order to obtain unauthorised access to systems, networks or physical and vital sites, or for financial gain.

### 1.1. Problem statement

Issues with ARP and ARP Spoofing ARP has some basic security issues in that it modifies the host's ARP cache table in the case of trustworthy mutual-agreement operations during the request/response message delivery process. The initial step in performing an MITM attack involves poisoning the Address Resolution Protocol (ARP) tables in the target machine and the router. The MITM attack will target Layer 2 (OSI) networks by using ARP protocol weaknesses. In computer communication, an essential and most commonly adopted protocol is the ARP. ARP messages operate in a Local Area Network (LAN) by translating IP addresses to MAC addresses. Nevertheless, the ARP protocol possesses significant weaknesses making it susceptible and easy to abuse by attackers [5]. The most significant disadvantage is that the protocol is stateless, with no reliable technique existing for authenticating ARP responses in the network. The reply packets are always subject to spoofing by hostile entities on the same network. Regarding ARP Spoofing, this is the process of associating one client's MAC address with another client's IP address [6]. Moreover, ARP Spoofing may be adopted to cause a DOS attack on an LAN simply by intercepting or blocking packets and not forwarding them. In recent years, ARP poisoning has been markedly focused on in relevant research. Hackers engage in ARP poisoning to undertake an array of attacks, including but not limited to MITM and DoS. Consequently, the network's efficacy is undermined. ARP spoofing may be detected and mitigated using a variety of approaches [7]:

- a. Cryptographic methods
- b. Developing changes in the existing system kernel
- c. Packet filtering
- d. Using third-party applications
- e. A static ARP cache table is manually configured

First, the cryptographic technique involves encrypting the ARP protocol. This strategy's principal disadvantage is its incompatibility with the standard ARP protocol, while it also degrades the ARP protocol's performance. The second technique modifies the operating system kernel to avert ARP spoofing attacks. Nevertheless, not all operating systems—rather only open-source ones—can develop this method. Furthermore, it may be incompatible with the standard ARP protocol. ARP spoofing attacks may be avoided by scanning and filtering packets prior to arrival at their destination. This requires an active firewall for scanning all packets and greater processing power, which significantly affects the typical home user's performance. The fourth option is utilising third-party system to detect and protect users. However, other

*Provably curb man-in-the-middle attack-based ARP spoofing in a local network ... (Hiba Imad Nasser)*

research has evidenced the inefficiency of this method against a spoofing attack using the ARP protocol. The fifth option is the primary yet most effective method of preventing an ARP spoofing attack, which involves manually adding a MAC address to the static ARP cache table. Even so, the time-consuming nature of this approach means that not all network administrators will be prepared to exert themselves.

The aforementioned techniques do not always provide an efficient and appropriate solution for the standard ARP protocol. Independent of router support, our proposed solution enhances certain techniques adopted to detect and prevent ARP-spoofing. This avoids the task of creating a static ARP cache table and manual management, instead incorporating the ARP validation function as a means of automating the procedure. Therefore, the network security contributions made through the proposed method are:

- a. A simple method that is compatible with standard ARP protocol and does not alter old devices;
- b. A method that combines packet filtering with blocking and detection of ARP spoofing attacks;
- c. A method that defends against DOS attacks;
- d. A method that, independent of router capability, will implement dynamic table and dynamic entries for the device.

Organising this Paper, section 2 describes the extant literature related to this issue. Section 3 defines the problem addressed in this work, as well as explaining how to spoof the ARP signal. Section 4 presents and implements our proposed method in greater depth. Section 5 provides our research results. Section 6 details the security analysis and evaluation of our proposed method. In the final section, the conclusions are summarised.

## 2. RELATED WORK

The following provides a list of previously proposed methods for detecting and combatting ARP poisoning attacks, focused on a particular scenario in order to develop the requisite tools. Prabhadevi. B [8] GNS3, Wireshark, in addition to the packet analysis tools Ettercap and Wincap are discussed in this work. IP-Mac table comparisons for the ARP headers and Ethernet are undertaken using this framework, with information added to a fake list, if a fake entry is discovered. Messages are sent to the gateway to broadcast an alert regarding cache poisoning attacks, add fake information to the fake list, as well as to clean the cache every ten minutes. An updated ARP table is required per every new host on the network. This method is reliant upon data in the fake list in order to detect the attack, although the list's storage locations has not been discovered, which may make the system vulnerable if attacked, while also wasting time due to no confirmation being provided that it works with a real network.

Hijazi *et al.* [9] The researchers discuss various ARP issues, recommending several techniques for detecting and blocking these attacks from compromising the network's security. The proposed solution is based on a static entry mechanism for the ARP table, type comparison, in addition to input IP-Mac-based detection and prevention actions. The system is safe if the IP-Mac address in the ARP table matches. To continue working, the fake IP-Mac must be removed from your table. These operations are undertaken on the proxy server utilising the Patch file to execute, resolve and correct the ARP Security problems. The effectiveness of such a strategy has been confirmed, with it requiring no further resources and its application in a real-world local network providing practical solutions. This method may be used to implement static entries in the ARP tables, although dynamic entries are unsupported.

Majumdar *et al.* [10] By implementing static entries in the ARP tables, it is straightforward to detect ARP poisoning attacks. The researchers developed a tool to perform ARP poisoning and spoofing using the Python programming language. The approach to attack detection is a script written in Python and the Scapy library. Following verification of the request packet's contents and issuing of a warning message if the real mac does not match the response Mac, utilising a static entry only in a cache table acts as a preventative strategy. Dynamic entries are unsupported by the proposed technique. Rather than writing a spoofing script, kali Linux attacks and packet analysis may be undertaken using the already accessible tools.

Alsukkar *et al.* [11] The researchers advocate two strategies, namely the detection and defence against MITM attacks via the deployment of a programme that alerts the user to the existence of an ARP spoofing assault, by displaying the attacker's IP-Mac address. Additionally, as a protection measure, the original ARP table settings are restored and a ping ICMP packet is sent with the actual Mac address to the router for all network devices. The application is installed on both the end user's and administrator's PCs. Furthermore, the network is deployed to attack and poison the ARP table. Python is used to write both programs, which run solely on the Linux operating system, with their efficacy and functionality having been evidenced. However, the proposed study has not proven to be efficient on other operating systems. It creates high network traffic due to the numerous packets sent to all devices every second.

Mahendra [12] the researcher's approach provides a means of improving the ARP table static entry process, minimising the arduous manual entry procedures, as well as automatically verifying the validity of

data entered into the static table. The method proved effective by utilising the static record feature provided by certain operating systems, for example Linux and Windows, for creating a semi-static table for the cache. In a virtual network, packets are sent and the replies' validity is verified, prior to an ARPing tool being used to add them to the table. Thus, the recommended technique does not guarantee security for all network users; it must create a white list of trustworthy IP-MAC addresses that may be used for validating the ARPing tool's efficiency for adding to the static record.

Rupal *et al.* [13] this article clarifies how an authentication tool may also be adopted to detect and prevent ARP poisoning in a dynamic IP setup. IP-MAC pairs are authenticated using an ARP cache, which is then stored in a text file in a secondary ARP cache, with the primary ARP cache used to retain this information. One server configures the IP DHCP, the other authenticates users using MySQL and database, while the third detects and prevents cache poisoning attacks. An (ICMP) broadcast request and (ICMP) replies are sent to all devices in order to detect and prevent attacks. The IP-Mac is removed from both the secondary and primary ARP caches, since it failed to send a response. However, the system broadcasting requests make it inefficient, due to the network becoming overloaded and the server's requiring management by authorised persons and use of trustworthy storage locations.

Girdler *et al.* [14] proposed an active defensive mechanism for preventing ARP spoofing in a local network. Software-defined networking (SDN) is a somewhat novel technology that has swiftly gained traction in recent years, as a consequence of its numerous advantages for carrying out network administration. This paper initially implemented an SDN intrusion detection and prevention system (IDPs). Subsequently, it presented an ARP spoofing detection technique that utilises specifically created tools and libraries in order to check user input (Mac) at a specific moment, after which suspicious entries are added to a black list of Mac Addresses. The solution is efficient and rapidly responds to intrusion attempts in a real network environment. However, this approach may prove ineffectual; it requires customers to submit their IP and MAC addresses in advance and stores them in a database for detection. Ultimately, the server does not authenticate clients, rather it simply waits for the attack to occur before resolving it.

Alwazeh *et al.* [15] the researchers developed a novel model comprising two sender and receiver systems in order to counter MITM attacks, specifically using the public key environment and the private key to enhance the system's performance, as well as improve the efficiency, reliability, and confidentiality of the server-client data transfer. Public and private key creation and distribution is carried out using ECC and Diff-Helman cryptography, hybrid AES and Blowfish encryption, in addition to ECDS and SHA-256 authentication and integration. To reduce this attack, the system uses the most robust methods for cryptography, authentication and data transfer between the client and the server, alongside the optimal configuration of the client and server handshake mechanism. The principal problem with cryptographic strategies is that they do not work with the standard ARP protocol, thus slowing down their speed.

The comparison criteria include whether it is dependent on static ARP entries, works in the ARP protocol, is scalable, has manual or automatic configuration and implementation, is reliant on cryptography algorithms, consumes time, as well as whether it requires special or expensive hardware. Table 1 compares the proposed technique with previous techniques.

Table 1. Comparison among different methods using nine comparative attributes

Methods	Static ARP	ARP	Full Prevention of ARP spoofing	Scalability	Automation	Authentication	Cryptographic	Cost Effective	Time effective
Proposed	x	✓	✓	✓	x	x	x	✓	✓
Fake list-NS3 [7]	✓	✓	✓	✓	x	x	x	✓	x
Patch file[8]	✓	✓	x	✓	✓	x	x	x	x
Filter packet-tool[9]	✓	✓	x	✓	x	x	x	✓	✓
Defense tool[10]	x	✓	✓	✓	x	✓	x	✓	x
ARPing tool-semi Static[11]	✓	✓	✓	✓	✓	x	x	✓	x
IP configuration [12]	x	x	✓	✓	✓	✓	x	x	x
SDN (IDPs) [13]	x	x	✓	✓	x	✓	x	x	x
Sender&reciver system[14]	✓	x	x	✓	✓	✓	✓	✓	x

### 3. BACKGROUND

#### 3.1. Address resolution protocol

The goal of address resolution protocol (ARP), one of the main protocols in the TCP/IP family, is to map an IPv4 address to a physical address. To communicate with another device, network applications use IPv4 addresses in the application layer. On the other hand, the address in the data link layer is the MAC address, which is permanently burned into the network card. The goal of address resolution protocol (ARP) is to determine the MAC address of a device in your local area network (LAN) in order to identify the IPv4 address that the network application is trying to interact with [16]. As soon as a packet leaves the LAN, the MAC address is assigned to the next hop. To send a data frame across a LAN, the sender must know the receiver's MAC address. The ARP protocol is based on two types of messages, namely ARP request and ARP reply [17]. The target host's MAC address is included in the ARP request, with the MAC address linked with that IP address possible to view in the ARP reply.

For example, the source device uses ARP cache to determine whether it already knows the resolved MAC Address of the destination device while communicating with another device. For communication, it will utilise that MAC address. The source machine generates an ARP request message with its data link layer address (sender hardware address) and IPv4 address as the sender protocol address if ARP resolution is unavailable in the local cache [18]. In this case, the IPv4 address of the destination is used as the Target Protocol Address. No Target Hardware Address is apparent because the system is unable to discover it at this time. An ARP request message is sent by the source to the local network, requesting a new IP address. Because it is a broadcast, every device on the LAN receives the message. When a source wants to communicate with another device, each device checks the target device's IPv4 address against its own (IPv4 address). Those not matching will have the packet discarded. If the Target Protocol Address matches, the targeted device will send an Address Resolution Protocol (ARP) reply message. The data for the targeted hardware address and targeted protocol address is received from the ARP request message and utilised in the reply message. The destination device's ARP cache will be modified, given that it will shortly need to access the sender machine [19]. A unicast is used, rather than a broadcast, to send an ARP reply message to the target device. The sender hardware address stores the destination's layer two address; the source computer processes this as part of the ARP [20]. After receiving the ARP reply message, the source machine's ARP cache is updated with the sender hardware address and sender protocol address.

#### 3.2. ARP cache poisoning and spoofing attack

The principal objective of ARP spoofing is to exploit any ARP protocol authentication vulnerabilities, through sending spoofed ARP messages to the LAN [21]. In the majority of instances, the idea underpinning the attack is to connect the attacker's host MAC address to the target host's IP address, resulting in any communication sent for the target host instead being rerouted to the attacker's host. To evade detection, the attacker may snoop on packets while forwarding traffic to the real default destination, or amend the contents prior to forwarding it (an MITM attack) [22]. A DOS attack may be performed by dropping some or all of the packets on the network. ARP spoofing is used to capture bandwidth by preventing the communication of all other devices. Protocols such as ARP are stateless [23]. If a network host receives an ARP request without requesting it, the network host will automatically cache the response [24]. Even entries that have not expired are rewritten whenever an ARP reply packet arrives. A host has no means of authenticating an ARP protocol packet's source [25]. Two simple spoofing techniques may be used that exploit this vulnerability in the ARP protocol:

##### 3.2.1. ARP request spoofing

This spoofing approach is better comprehended by using an example MITM attack. In this case, the attacker deceives both the victim and the gateway. The attacker sends the victim an ARP request packet. To mislead the victim, this ARP request packet includes faked data. The victim believes that ARP request packet's sender was a gateway, meaning that it stores the information from the ARP request packet in its own ARP cache table; similarly, the gateway is spoofed. All gateway-victim traffic is forwarded to the attacker due to the poisoning. Subsequently, the perpetrator creates a path between them. Because the network connection is uninterrupted, the victim typically remains unaware of the attack.

##### 3.2.2. ARP reply spoofing

ARP reply packet spoofing is identical to ARP request spoofing. The ARP packet type is the sole aspect that differs. Despite the victim having never asked for it, the attacker directly sends an ARP reply. In such cases, the intrusion may be swiftly detected, because it is unusual for a host to receive an ARP reply without issuing an ARP request. Figure 2 shows how an ARP spoofing attack is performed.

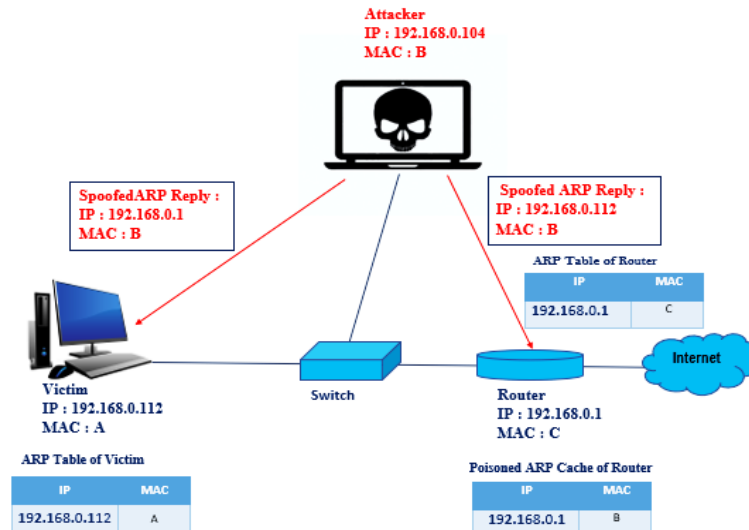


Figure 2. The ARP spoofing attacks technique

#### 4. THE SYSTEM MODEL

Our system model has been proposed with the aim of preventing ARP spoofing. The modification or implementation of a particular topology, is not required for ARP, nor does the protection strategy need any hardware installed on the network. Additionally, due to the ARP spoofing defence system not using any encrypted fee mechanism, it is lightweight, fast, as well as able to instantly detect an ARP spoofing attack. The Python programming language is adopted to implement the approach described in the application Layer-by-Layer Internet. The ARP protocol is not altered in any way. Given that the current standard ARP protocol is compatible with it, this solution is not reliant upon any alternations to extant LAN technology or hardware. In order to test this strategy, we utilised a local network of four computers, a router, a Windows 10 computer, as well as two virtual machines. A client and an attacker are the only two devices in this virtual environment—Kali Linux for the attacker and Ubuntu for the client. Figure 3 presents the network structure, with our default settings indicated. Each number is an Internet protocol address. An IP address is used in this architecture.

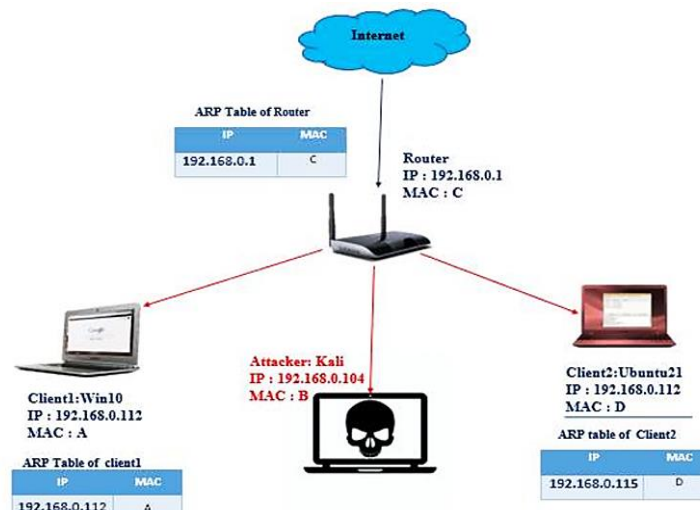


Figure 3. ARP network topology

The following process was used to evaluate the performance of our proposed methodology. For clients 1 and 2, we installed the application containing our approach, although not in the router. Subsequently, we executed an ARP spoofing attack using the Scapy library on this internal network from the

attacker node. Scapy is a particularly strong and flexible Python-based packet manipulation tool. Scapy is used to receive, sniff, analyse, and forge the network packets, while it can also sniff packets contained in a pcap file. Additionally, we used Scapy to efficiently undertake trace routing, probing, scanning, verification and validation and network discovery. Furthermore, we adopted Scapy to control the source and destination ARP headers' IP and MAC address, as well as to perform complete MITM attack scenarios and defensive measures on end-user PCs.

#### 4.1. Defence scenario

The suggestion shown in detail is communication between the source and destination hosts, namely the sending and receiving of ARP packets. The function examines the packet and saves the original MAC address as the "real mac" variable, while the response's MAC address is saved as the "response mac" variable. Following this, it compares the two values to determine whether they are identical; if not, this suggests that the values have been faked, with another function being invoked that presents a warning message to the user. This procedure may be repeated indefinitely in order to compare each device's genuine and simulated mac addresses. Having detected the attack and notified the victim of the attacker's IP address, the attacker's IP address is blocked from the firewall and all protocols and ports. This ensures that irrespective of whether an attack recurs, the attacker will still be prevented from accessing any of the victim's data on their device. The attacker's device may subsequently be attacked to destroy the service. The set of defence steps is presented in algorithm 1:

##### Algorithm 1: defence algorithm

###### BEGIN:

Input: ARP request packet.

Output: ARP reply packet.

Step1: Before sending ARP Reply Frame, the dest. host will

Step2: check the IP address of the source host in its firewall.

Step3: if (firewall contains IP address), then

Step4: A message will display directly to the dest. host

Step5: show alert message this IP is blocked

Step6: else

Step7: checking for any types of attacks ARP Request packet

Step8: If ( the packet is an ARP packet and ARP reply (op=2), then

Step9: get the real MAC address of the sender and response MAC from the ARP Reply packet

Step10: if they're different, definitely there is an attack

Step11: If (NOT match), then

Step12: show alert message in dest. host "under attack" and IP the attacker

Step13: blocking this attacker from all protocols in dest. host

Step14: block this IP

Step15: do reverse attack on attacker host to cut service

Step16: reverse attack on attacker

Step17: else

ARP Reply Frame will be sent to the source host.

END: //end of the procedure

The Figure 4 flowchart presents the diagram detection and prevention procedures for the proposed mechanism. The flow chart clarifies the defence procedure. As long as no interference is detected, the process will use the sleep() method to stop before the central loop is repeated. Although this step is optional, it is recommended in order to reduce pressure on system resources. Indeed, if an attack is detected, the instructions included in the "if statement" will run without delay, in order to successfully defend against the attacking computer.

#### 4.2. Experiment attack scenario

We performed experiments with an ARP spoofing attack scenario for three components, namely two basic PCs and an internet-connected portal. After using the tools Nmap and Net discover in Kali Linux to determine which network devices, ports and protocols may be attacked, we executed an ARP poisoning attack on a PC running Windows. First, we verified the internet connections for the victim's and attacker's respective computers. After starting the attack tool, the attacker will modify the gateway's MAC address, input the victim's IP address, select the type of interface, then click the START button to begin the attack.

Thus far, two IP addresses have the same MAC address. This is a scenario in which an attacker's IP address 192.168.10.104 and the gateway's IP address 192.168.0.1 have the same mac address.

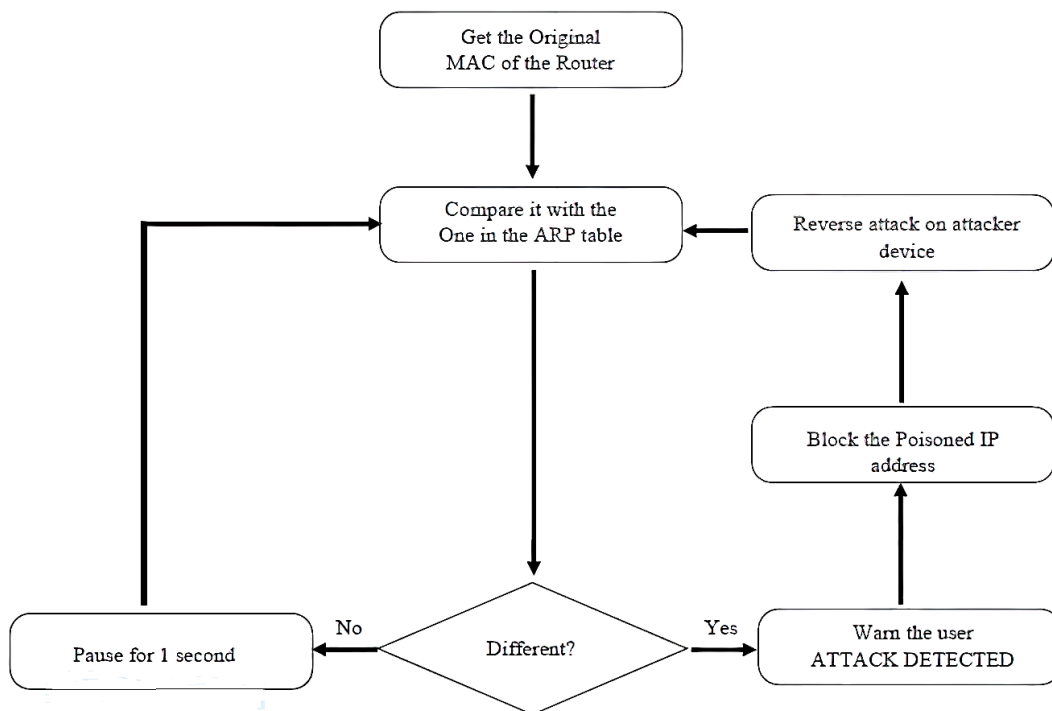


Figure 4. proposal method flowchart

## 5. RESULTS AND DISCUSSION

The tools that we have created will be presented in this section. To launch an attack or build a defence, they have a simple user interface. All tests were performed on a computer with Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz, 2.60 GHz, with 4.0 GB RAM and a 64-bit Windows 10 operating system.

### 5.1. Output for a defence scenario

Figure 5 presents the Defence Tool's primary screen window. Initially, you choose the network interface. It is not necessary to input the IP address of the gateway; it will be automatically discovered. Subsequently, click the "Start" button to initiate the defence. Upon discovery of an attack event, the attackers' IP addresses will be presented in the Status box. When an attack occurs, the attacker's IP address and a warning message will be provided to the target user. Once an attack has been blocked, a reverse attack can be initiated to eliminate the attacker's own device. Following an attack's discovery, the table will be reset and assigned a new IP/MAC address. The first scenario is illustrated in Figures 6 and 7. The results indicate that well-defined techniques place a premium on dynamic ARP entries while safely building the ARP cache table. This solution achieves the following: detection and prevention of attacks via ARP spoofing.

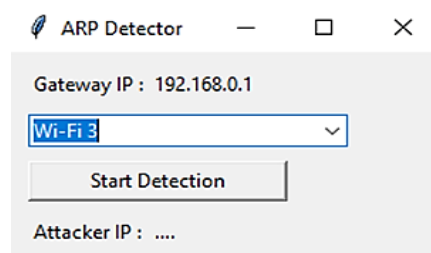


Figure 5. GUI ARP defense tool



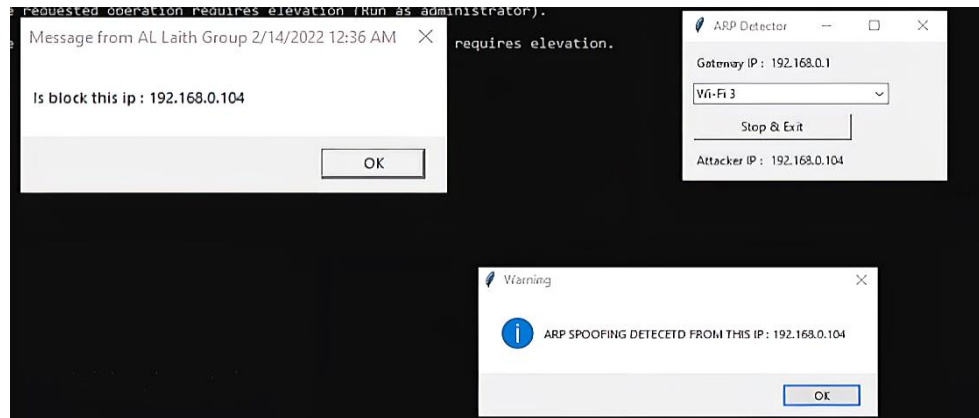


Figure 6. ARP defense tool result

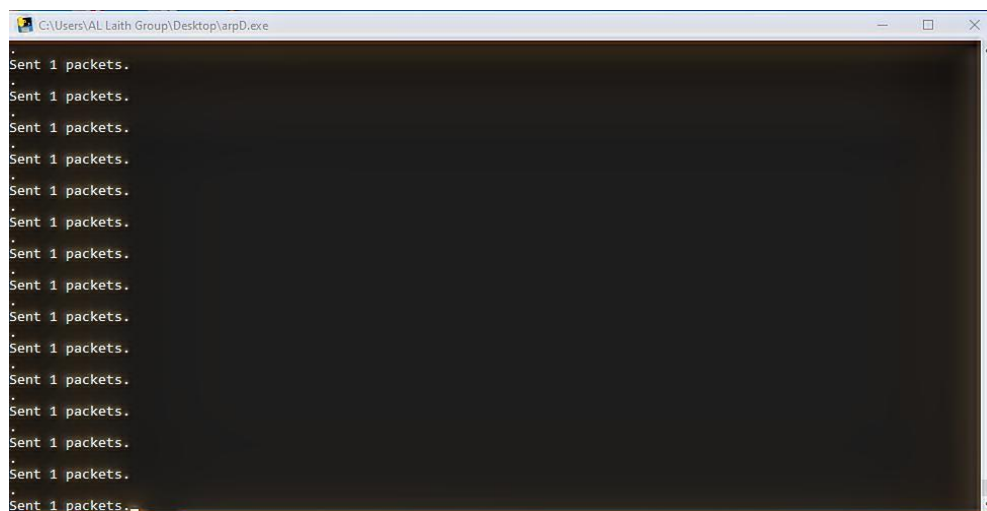


Figure 7. ARP defense tool result

## 5.2. Output for an attack scenario

The GUI was created to serve as a front-end for pre-developed attacks, as depicted in Figure 8. Prior to initiating the attack, we searched the network for devices associated with it using special tools in Kali Linux. This provided us with a list of all connected devices (apart from the attacker's device) with all the protocols and ports open on each device. Completing the scanning procedure is somewhat time consuming. Following its completion, the victim device's IP address must be inputted. Having made all the required settings, the "Start" button is clicked; if the attack is successful, the spoofed packets will be sent to the victim's machine. To cease the attack, the "Stop" button can be pressed, which stops the current attack and allows you to restart it.

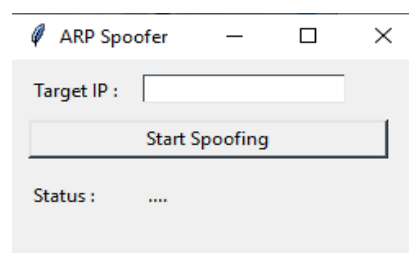


Figure 8. GUI ARP spoofing tool

### 5.3. Attack detection time

The attack detection time is the time interval between when the spoofed packet is transmitted to the victim's system as a legitimate packet and when it is validated, and the spoofing is detected. Figure 9 shows that the ARP response detection time for a faked packet is approximately 5 seconds. The ARP request and the ARP answer took over 0.38 seconds to complete.

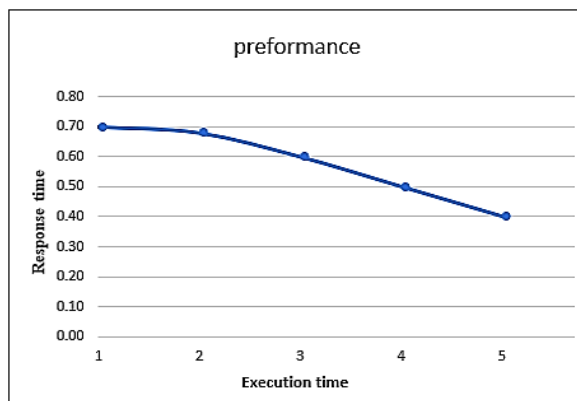


Figure 9. Performance time

### 5.4. Discussion

The algorithm successfully filtered all forms of ARP spoofing packets identified so far (Abnormal). Only normal ARP traffic was sent to the destination (the target host), while ignoring suspicious ARP packets. On the other hand, this technology had little impact on network performance because ARP packets contain no data and are very small compared to TCP/IP packets. Here is a quick explanation of the objectives of the study and the best expectations for the use and implementation of the algorithm we proposed:

- Deny all types of malicious ARP traffic detected and identified.
- Protection from MITM attacks made possible by fake ARP packets.
- Prevent DoS attacks using fake ARP packets by malicious users.

## 6. Security analysis and evaluation

This section analyses our proposed method and explains how our approach may effectively resist well-known malicious techniques. Additionally, the suggested technique is proven to be secure. Furthermore, a comparative examination of some closely related tools is presented.

### 6.1. Theorem 1. Our proposed scheme can resist an arp poisoning attack

**Evidence-Proof.** In a local network, the attacker spoofs his Mac and uses it to make fake queries to any device on the network and obtain legitimate responses, obtaining access to the victim's sensitive information. Before providing replies to requests sent over the network, the suggested method checks and filters them to determine their authenticity. When the system detects fraudulent requests, the victim is warned that an attack has occurred, as shown in step (12) of the algorithm (1) and Figure 6.

### 6.2. Theorem 2. Our proposed scheme can resist a blocking attack

**Evidence-Proof.** After taking complete control of the access point and attempting to attack the victim to obtain their data, the attacker will be unable to do, given that the proposed system will block the attacker's IP address in addition to all protocols and ports. This is seen in step (13) of the algorithm (1) and Figure 6.

### 6.3. Theorem 3. Our proposed scheme can resist a mitm attack

**Evidence-proof.** MITM attacks were analyzed using our algorithms. According to the attack scenario shown in Figure 2, half of the attack target can be achieved by poisoning the cache of the victim's device. As long as the fake attacker doesn't detect it, he won't be able to get any information about the victim. Therefore, MITM does not occur as defined by algorithm (1) in steps (12, 13). Thus, such an attack is unlikely. In other words, if the attacker succeeds, he can still do the same thing he would have done without our technique. Because of this, the scale of the threat is relatively less.

#### 6.4. Theorem 4. Our proposed scheme can resist DoS attack

Evidence-proof. After obtaining the router's IP-MAC address, the attacker uses it to send fake traffic or huge resource requests, causing the server to crash. However, the method mentioned here checks and filters all network traffic related to the ARP table attack, as well as blocks the fake IP from all protocols and ports that the attacker might exploit in order to perform a DoS attack against the victim, thus preventing them from accessing their IP address. In Table 2, the chart compares security features with charts similar to ours.

Remark: After analysing the security of these methods in the study [11], we achieved equivalent results. The technology and programming language used to create them are the sole differences. In our research, we utilised the Python language and the Scapy library, whereas research [11] used the VB.net and MySQL languages to create the database and run queries to detect and resist attacks.

Table 2. Security features comparisons Schemes

security features	Schemes			proposed
	[9]	[10]	[11]	
ARP poisoning	Yes	Yes	Yes	Yes
Blocking attack	No	No	Yes	Yes
MITM	No	No	Yes	Yes
DOS	No	Yes	Yes	Yes

## 7. CONCLUSION

There are severe repercussions from attacks exploiting ARP, which is a non-stateless protocol. By interfering with the users' connections, an attacker is able to steal their data and even reroute their traffic to sites other than those the victim intended to visit. Those responsible for networks should show greater awareness and preparedness for these types of attacks. Consequently, users should develop greater awareness about them and implement security measures to protect their personal information. This study has explained and implemented an ARP spoofing and MITM attack after describing the various kinds and categories of vulnerabilities apparent in the ARP protocol. This was followed by the development of a defensive system that operates on end-user devices to maintain their security against these kinds of attacks. We successfully demonstrated our proposed strategy's effectiveness by executing the completed approach on the user's system. Our proposed method may prevent these attacks, as its effectiveness against them has been confirmed. No alterations are required for extant ARP protocol standards or devices in order to achieve this. Moreover, it is straightforward to implement in any host environment. Due to its limitations, the ARP protocol will soon become obsolete. Nevertheless, there are effective workarounds that will be available in future.

## FUTURE WORK

Our study's use of machine learning will enable us to train the system to reliably predict attacks, permitting us to test a larger variety of measurements.




## REFERENCES

- [1] "Global regional internet penetration rate 2022 | Statista". <https://www.statista.com/statistics/269329/penetration-rate-of-the-internet-by-region/> (accessed Mar. 12, 2022).
- [2] A. I. Abdulsada, D. G. Honi, and S. Al-Darraj, "Efficient multi-keyword similarity search over encrypted cloud documents", *Indones. J. Electr. Eng. Comput. Sci.*, vol. 23, no. 1, p. 510, Jul. 2021, doi: 10.11591/ijeecs.v23.i1.pp510-518.
- [3] M. Alzuwaini and A. Yassin, "An Efficient Mechanism to Prevent the Phishing Attacks", *Iraqi J. Electr. Electron. Eng.*, vol. 17, no. 1, pp. 1–11, Jun. 2021, doi: 10.37917/ijeec.17.1.15.
- [4] A. Mallik, A. Ahsan, M. M. Z. Shahadat, and J. C. Tsou, "Man-in-the-middle-attack: Understanding in simple words", *Int. J. Data Netw. Sci.*, vol. 3, no. 2, pp. 77–92, 2019, doi: 10.5267/j.ijdns.2019.1.001.
- [5] Z. Trabelsi and K. Shuaib, "Spoofed ARP packets detection in switched LAN networks", *SECURITY 2006 - Int. Conf. Secur. Cryptogr. Proc.*, pp. 40–47, 2006, doi: 10.5220/0002102400400047.
- [6] M. Anathi and K. Vijayakumar, "An intelligent approach for dynamic network traffic restriction using MAC address verification", *Comput. Commun.*, vol. 154, no. July 2019, pp. 559–564, 2020, doi: 10.1016/j.comcom.2020.02.021.
- [7] J. S. Meghana, T. Subashri, and K. R. Vimal, "A survey on ARP cache poisoning and techniques for detection and mitigation", in *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, Mar. 2017, pp. 1–6, doi: 10.1109/ICSCN.2017.8085417.
- [8] B. Prabadevi and N. Jeyanthi, "A framework to mitigate ARP sniffing attacks by cache poisoning", *Int. J. Adv. Intell. Paradig.*, vol. 10, no. 1/2, p. 146, 2018, doi: 10.1504/ijaip.2018.10010532.
- [9] S. Hijazi and M. S. Obaidat, "A New Detection and Prevention System for ARP Attacks Using Static Entry", *IEEE Systems Journal*, vol. 13, no. 3, pp. 2732–2738, 2019, doi: 10.1109/JSYST.2018.2880229.




- [10] A. Majumdar, S. Raj, and T. Subbulakshmi, "ARP Poisoning Detection and Prevention using Scapy", *J. Phys. Conf. Ser.*, vol. 1911, no. 1, p. 012022, May 2021, doi: 10.1088/1742-6596/1911/1/012022.
- [11] G. Al Sukkar, R. Saifan, S. Khwaldeh, M. Maqableh, and I. Jafar, "Address Resolution Protocol (ARP): Spoofing Attack and Proposed Defense", *Commun. Netw.*, vol. 08, no. 03, pp. 118–130, 2016, doi: 10.4236/cn.2016.83012.
- [12] M. Data, "The Defense Against ARP Spoofing Attack Using Semi-Static ARP Cache Table", in *2018 International Conference on Sustainable Information Engineering and Technology (SIET)*, Nov. 2018, pp. 206–210, doi: 10.1109/SIET.2018.8693155.
- [13] Sri Venkateshwara College of Engineering. Department of Electronics and Communication Engineering, Institute of Electrical and Electronics Engineers. Bangalore Section, Institute of Electrical and Electronics Engineers. Bangalore Section. COM Chapter, Sri Venkateshwara College of Engineering, and Institute of Electrical and Electronics Engineers, *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT) : proceedings : 20-21 May 2016, Bengaluru, India.* .
- [14] T. Girdler and V. G. Vassilakis, "Implementing an intrusion detection and prevention system using Software-Defined Networking: Defending against ARP spoofing attacks and Blacklisted MAC Addresses", *Comput. Electr. Eng.*, vol. 90, Mar. 2021, doi: 10.1016/j.compeleceng.2021.106990.
- [15] M. Alwazzeh, S. Karaman, and M. N. Shamma, "Man in The Middle Attacks Against SSL/TLS: Mitigation and Defeat", *J. Cyber Secur. Mobil.*, Jul. 2020, doi: 10.13052/jcsm2245-1439.933.
- [16] S. Singh and D. Singh, "ARP Poisoning Detection and Prevention Mechanism using Voting and ICMP packets", *Indian J. Sci. Technol.*, vol. 11, no. 22, pp. 1–9, Jun. 2018, doi: 10.17485/ijst/2018/v11i22/92337.
- [17] Vaigai College of Engineering and Institute of Electrical and Electronics Engineers, *Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2020) : 13-15 May, 2020.* .
- [18] K. Cheng, M. Gao, and R. Guo, "Analysis and research on HTTPS hijacking attacks", in *NSWCTC 2010 - The 2nd International Conference on Networks Security, Wireless Communications and Trusted Computing*, 2010, vol. 2, pp. 223–226, doi: 10.1109/NSWCTC.2010.187.
- [19] M. S. Hossain, A. Paul, M. H. Islam, and M. Atiquzzaman, "Survey of the Protection Mechanisms to the SSL-based Session Hijacking Attacks", *Network Protocols and Algorithms*, vol. 10, no. 1, p. 83, 2018, doi: 10.5296/npa.v10i1.12478.
- [20] A. Susanto and W. K. Raharja, "Simulation and Analysis of Network Security Performance Using Attack Vector Method for Public Wifi Communication", *Int. J. Informatics Comput. Sci.*, vol. 5, no. 1, 2021, doi: 10.30865/ijics.v5i1.2764.
- [21] S. D. and R. K. J. Singh, "A Detailed Survey of ARP Poisoning Detection and Mitigation Techniques". .
- [22] S. Venkatramulu and C. V. G. Rao, "Various Solutions for Address Resolution Protocol Spoofing Attacks", *Int. J. Sci. Res. Publ.*, vol. 3, no. 7, pp. 1–12, 2013.
- [23] V. Ramachandran and S. Nandi, "Detecting ARP spoofing: An active technique", *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3803 LNCS, no. May, pp. 239–250, 2005, doi: 10.1007/11593980\_18.
- [24] V. Rohatgi and S. Goyal, "A detailed survey for detection and mitigation techniques against ARP spoofing", in *Proceedings of the 4th International Conference on IoT in Social, Mobile, Analytics and Cloud, ISMAC 2020*, Oct. 2020, pp. 352–356, doi: 10.1109/I-SMAC49090.2020.9243604.
- [25] P. Pandey, "Prevention of ARP spoofing: A probe packet based technique", *Proceedings of the 2013 3rd IEEE International Advance Computing Conference, IACC 2013*, pp. 147–153, 2013, doi: 10.1109/IAdCC.2013.6514211.

## BIOGRAPHIES OF AUTHORS



**Hiba Imad Nasser Al-Kanaani**    holds a Bachelor's degree in Computer Science from the University of Basra, Iraq in 2009. She is currently working as a teacher teaching computers at the General Directorate of Education in Basra. Her research interests include information security, cyber security, machine learning, and the internet of things. She can be contacted at email: eduppg.hiba.amad@uobasrah.edu.iq.



**Mohammed Abdulridha Hussain**    received his Bachelor degree in Computer Engineering in 2004 from University of Basrah, Basrah, Iraq. He received his M.Tech. Degree in Computer Science and Engineering in 2009 from GGS Indraprastha University, Delhi, India. Ph.D degree from Huazhong University of Science and Technology, Wuhan, China. His research interests include network, network security, cloud security, web application security and Wireless Sensor Network. He is currently working as an assistant Professor in Basrah University. He can be contacted at email: mohammed.abdulridha@uobasrah.edu.iq.