IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 7, 2022, accepted April 28, 2022, date of publication May 3, 2022, date of current version May 12, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3172329

D-ARP: An Efficient Scheme to Detect and Prevent ARP Spoofing

SABAH M. MORSY[1] **AND DALIA NASHAT**[2]
[1]Faculty of Computers and Information Technology, National Egyptian E-Learning University, Giza 12611, Egypt
[2]Department of Information Technology, Faculty of Computers and Information, Assiut University, Assiut 71515, Egypt

Corresponding author: Dalia Nashat (dnashat@aun.edu.eg)

ABSTRACT Nowadays, cyber-attack is a severe criminal violation, and it is one of the most active fields of research. A Man-in-the-middle attack (MITM) is a type of cyber-attack in which an unauthorized third party secretly accesses the communication between two hosts in the same network to read/modify the transferred data between them. ARP spoofing-based MITM attack exploits ARP protocol weakness where the attacker associates its MAC address with the IP address of an intended legitimate host. Although there are many defense approaches for ARP spoofing based-MITM attacks, these methods are uncompleted or have a performance overhead since they modify the original ARP protocol. Also, some of these approaches depend on the centralized server, which is a single point of failure. This paper presents a detection scheme for ARP spoofing-based MITM attack called D-ARP, which is compatible with the original ARP protocol. The main idea of D-ARP is to send an ARP packet signed with a key in parallel with the original ARP packets to make a correlation between requests and replies. Each host records all types of signing ARP packets in a log file. Based on this correlation, D-ARP matches the injected key to detect ARP spoofing if there is a duplicate or conflict in the MAC address. For more reliability, D-ARP uses the DHCP server and the Nmap feature to detect the MAC addresses of MITM attackers. Moreover, this scheme also offers a module for Admin to create a trusted list of hosts. The experimental results show that D-ARP is highly effective for detecting and preventing ARP spoofing with zero false positives and zero false negative probabilities.

INDEX TERMS Address resolution protocol, man-in-the-middle, ARP spoofing, anomaly detection.

I. INTRODUCTION

The number of internet users is growing too fast in many fields such as industry, education, marketing, social networking, and online shopping. The use of the Internet increased with the growth of using online communications in organizations by using wired and wireless networks and the widespread use of smartphones in social media. Nearly 4.66 billion people were active internet users in January 2021 [1]. Although The number of websites and applications increases rapidly, they expose to attacks. Man-in-the-Middle (MITM) is one of the most dangerous cyberattacks and is considered the main serious concern for many network security [2]. In 2017, American multinational consumer credit reporting agency (Equifax) withdrew its mobile applications after concerns that MITM identified and exploited their vulnerabilities [3].

The associate editor coordinating the review of this manuscript and approving it for publication was P. Venkata Krishna.

A MITM can secretly convince two hosts to believe that they are directly communicating with each other. But in fact, they share their transferred data and messages with the attacker as illustrated in figure1 [4]. MITM firstly targets data confidentiality which eavesdrops on the transferred data between two hosts. Then, it breaks data integrity by intercepting communication and modifying the transferred data between two hosts. Finally, it controls data availability by forcing one of the two hosts to stop communicating with another by modifying or destroying the transferred data [5], [6].

Spoofing-based MITM is one of the most common MITM attacks in which the attacker is a third party that involves intercepting the legitimate path between two hosts. These victim hosts are convinced that communicating with each other, not with an intruder. Many network protocols are vulnerable to spoofing based on MITM, such as Address Resolution Protocol (ARP), Dynamic Host Configuration Protocol (DHCP), Internet Protocol (IP), and Domain Name

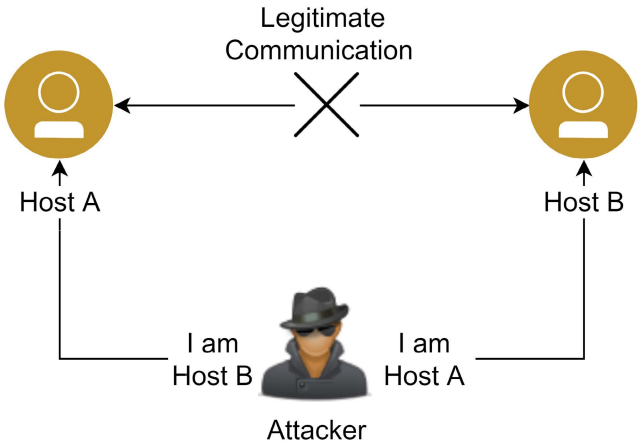

FIGURE 1. Man in the middle.

System (DNS). This work focuses on ARP spoofing based MITM due to the importance of ARP protocol since each frame must contain the destination Media Access Control (MAC) address to be transmitted [7]. Moreover, ARP spoofing attacks can be used to perform many other critical attacks such as Denial-of-Service (DDoS) and Session hijacking attacks.

ARP is one of the most essential protocols for LAN communication which exists in the data link layer. It resolves the network layer address (IP address) assigned by the Network layer to a data link layer address (MAC address). Also, ARP is crucial because any frame transmitted within a network must contain the destination MAC address. Each host has an ARP cache table that consists of <IP, MAC> pairs to communicate with another host to reduce the number of broadcast messages. This table contains two types of entries (i.e., static and dynamic). Static entries are permanent and reside in the ARP cache table, even if the system reboots. The dynamic entires only reside in the ARP cache table for a specific time (usually a few minutes). This time is determined according to the type of operating system. ARP is a stateless protocol, so it has a lacked caching system security. Therefore, it can easily spoof by a MITM attack. ARP spoofing-based MITM attack exploits the weakness of ARP protocol in which an unauthorized third party secretly accesses the communication between two hosts in the same network to read/modify the transferred data between them [8]–[11].

Three defense mechanisms against ARP spoofing are detection, prevention, and mitigation. Since ARP Spoofing aims to damage the <IP, MAC> correlation in the ARP cache table, the intruder is considered a third party in the communication. Defense mechanisms try to solve the vulnerabilities of sending and receiving ARP packets. To guarantee that the transmitted data reached a legitimate destination, not to the intruder. We detect the spoofing before the source communicates with an illegitimate destination to protect the data and prevent attacks.

This paper introduces a detection and defence scheme for ARP spoofing called D-ARP, which is compatible with the original ARP protocol. D-ARP is a new scheme to detect and prevent ARP spoofing-based MITM by sending an ARP packet signed with a key in parallel with the original ARP packets to associate a correlation between requests and replies. Each host records all types of signing ARP packets (i.e., in a log file). Based on this correlation, D-ARP matches the injected key to detect ARP spoofing if there is a duplicate or conflict in the MAC address.

The D-ARP scheme guarantees the following:

- There is no false positive or false negative probability.
- Fully automated process and there is no headache for Admin.
- No delay in sending or receiving packets.
- No additional overhead.
- No single point of failure.

This paper is organized as follows: Section II reviews the literature related to this work. Section III provides an overview of ARP spoofing. Section IV describes our proposed scheme in details. Experimental results are presented in section V. And finally, section VI concludes this paper and introduces the future directions.

II. RELATED WORK

It is very important to detect malicious nodes in the network [12], [13]. To detect ARP spoofing, sometimes some networks use software tools such as Arpwatch and Wireshark. Arpwatch [14] is used to detect ARP spoofing, but it depends on the administrator to determine the malicious pairs, and this leads to high false positives probability. Also, a Wireshark tool is used to detect spoofing by detecting the duplicate use of IP addresses, but it does not detect all forms of cache spoofing, and it isn't able to prevent ARP spoofing.

Therefore, there are many defense approaches for ARP spoofing attacks are proposed. These schemes can be classified into cryptographic solutions, server-based solutions, static entries solutions, voting-based solutions, and host-based solutions. Cryptographic solutions, which the most common solution to prevent attacks, such as [15]–[17] which based on authentication hash functions to generate keys (i.e., public and private keys), but these solutions almost need to update the ARP protocol itself.

A server-based solution such as [18] depends on passing packets via a trusted server to be analyzed. The main limitation of these schemes is that the server itself is considered as a single point of failure. Static entries methods are the simplest way to prevent ARP spoofing depending on assigning IP addresses manually, such as [19], the disadvantage of these methods is that they are not suitable for a dynamic environment and large scale networks. Also, voting-based techniques such as [20]–[23] provided an enhancement in ARP protocol which depend on other hosts to determine the fake <IP, MAC> mappings, but these have an additional overhead and must follow the new ARP protocol. The proportion of host-based solutions depends on

a duplicate of replies or machine learning features. The main drawback of these approaches is that they are not practical, and difficult to decide the level of trustiness and importance of each host.

S-ARP [15] is one of the most common cryptography-based approaches. This approach is considered backward compatible with ARP protocol which provides authentication of ARP replies through public-key cryptography. During the initial contact, all hosts generate public and private key pairs and transmit them to the Authoritative Key Distributor (AKD) with signed certificates. Then the server has been distributes these keys to hosts. By using these keys, any host could identify the legal users. However, S-ARP is the common scheme to prevent ARP spoofing based on authentication.

This scheme has a lot of drawbacks. Firstly, it depends on the Authoritative Key Distributor to authenticate replies, which is considered as a single point of failure in the network. Also, hosts need to communicate with AKD every time they receive an ARP reply in order to request the Public Keys of ARP request senders. If this distributor is down, ARP packets that are sent by a previously unknown host, cannot be verified by the host (i.e., the sender's public key is not in the destination's key ring). Secondly, even if the possibility that the AKD is working legitimately, an attacker can impersonate a host that goes down by cloning the hardware address of the host (but only until the cache entry of the host being impersonated, in the host being attacked, expires). Thirdly this scheme is that it does not support the dynamic assignment of IP addresses. Although S-ARP requires dealing with its ARP protocol, it also requires upgrading of the DHCP server called S-DHCP and incremental deployment is not easy, and a host may not accept ARP replies from non-S-ARP hosts. This means that S-ARP requires updates in two protocols. Fourth, one of the most crucial problems with this approach is the processing time to encrypt, decrypt, and send extra packets to get the public key or get the host verified, all this extra time processing is considered a performance overhead compared to the current ARP protocol. Finally, S-ARP will check the ARP replies only while the victims may be deceived by sending an ARP request [24]–[27].

Authors in [18] proposed a solution to overcome the authentication shortage and solve the voting method issue. The main idea of this solution is using a centralized server. This server has all <IP, MAC> pairs of each host inside this LAN. All Hosts have a table that contains a trusted <IP, MAC> pair, so the host can detect the attack easily and send everything about it to the centralized server.

Rupal et. al. [28] introduced a detection and prevention algorithm for ARP spoofing depending on hosts' authentication. This algorithm sends ICMP packets to construct a table that involves a combination of IP address and its corresponding MAC address for all hosts in the network, this table is called secondary cache. The main idea of this scheme is the duplication of MAC Address in ARP reply in the secondary cache, which is responsible for checking IP-MAC addresses respective the system in the network.

B. Prabadevi [29] presented a system based on analyzing packets to mitigate ARP spoofing using timestamps. It analyzed the contents of all ARP packets (request and replies) to check whether the Ethernet header is consistent with the ARP header or not. Then, a broadcast alert message is generated for ARP packets with a timestamp for these valid messages, and every 20 minutes the system clears the ARP cache.

Hijazi *et al.* [19] proposed a client/server solution. It does not need to install any special software since the server is responsible for detecting the attack. The solution depends on collecting all <IP, MAC> pairs automatically from ARP cache tables. After validations of <IP, MAC> Pairs, it constructed a table containing these pairs and then registered them in a proxy server which detected the attacker when it connected with the server with the same registered IP address.

A defense method presented in [30] depends on semi-static ARP entries to prevent ARP spoofing. Their main idea is to validate and manage the static entries of ARP and remove manual adding entries into the cache table. It does not need any modification of ARP protocol and devices. Also, it can not work in dynamically addressing environments and is not suitable for large-scale networks.

Xia *et al.* [31] proposed a defense solution for ARP spoofing oriented to the OpenFlow platform. This mechanism is implemented as a module of the POX controller, which exploits the SDN advantages to make OpenFlow defense against ARP spoofing. Active ARP inspection (AAI) is a centralized unit implemented on OpenFlow to manage its features. All ARP packets on the local subnet are forwarded to the controller to be processed by AAI subsequently. This mechanism is not flexible enough to integrate.

Singh *et al.* [32] presented a mechanism to validate the new association. Each host maintains a secondary table file to store the <IP, MAC> binding. When the host has received a new binding, it is validated by sending two ICMP probe packets one to the previous binding, and the other to the new one. For new entry of host without a previous entry in ARP cache, is validated by using ARP packets to find all the claiming hosts to that IP, used together with ICMP packet to provide a two-phase validation.

Authors in [33] presented a scheme in the SDN environment to detect and prevent spoofed ARP packets by extending the SDN controller to check every packet within networks. The mechanism depends on using the POX controller with the L2 learning module, an OvS switch, and a DHCP server which determines DHCP offer packets. On the connected switches, flow rules are established to forward all ARP and DHCP packets to the controller. POX also maintains a primary table containing IP to MAC address pairs for each device on the network, which is generated from received DHCP offer packets. POX tells switches to ignore received ARP packets with MAC addresses that aren't in the primary table, also a flow rule is then installed for the originating OvS port. The drawback of this solution appears when the network expands and the traffic increase.

The algorithm in [34] is used to detect Man-in-the-middle and its location which depends on studying the nature of ARP Spoofing itself. The algorithm is considered a switch-based detection approach. First, it used data structure to save data from ARP Packets. Then, the switch can determine the spoofing place.

A host-side solution based on ARP analysis to detect attacks is proposed in [35]. This solution conjugates machine learning and signal processing to detect MITM attacks which presents a ''statefulness'' into the ARP protocol by adding a padding layer to the frame and encoding a binary value and a sequence number to match a corresponding ARP request to its reply. The detection mechanism depends on the time taken for an ARP request to be sent directly from one host to another, which varies from the instance where an attacker intercepts the traffic.

Girdler *et al.* [36] proposed a solution to ARP Spoofing and Blacklist MAC address. This solution is a Software-Defined Networking (SDN)-based Intrusion Detection and Prevention System (IDPS). This is accomplished by dynamically adjusting SDN's operating parameters to detect malicious network traffic. For each incoming packet, after it is accepted by PacketIn events, it is processed by one of four functions (i.e., ARP Request Spoofing, ARP Reply Spoofing, ARP Reply Destination Spoofing, and Blacklisted MAC Addresses) to detect the type of spoofing. Also, the IDPS can identify devices whose MAC address has previously been blacklisted by the IDPS configuration tool.

Authors in [37] presented an algorithm that depends on sniffing and analyzing all incoming ARP packets. The basic step is to compare between the real MAC Address and the response MAC Address of the ARP Packet sniffed to detect conflicts. After capturing the ARP packet, it is analyzed to obtain the source MAC address of the sniffed ARP packet and the real physical address of the sender. If there is a conflict between the real MAC Address and the response MAC address, then there is an ARP attack. Also, this paper relies on static entries to prevent ARP spoofing.

A client/server-based intrusion detection system (CSIDS) was proposed in [38]. This protocol depends on a client/server model for exchanging and controlling the resolution messages between system members. CSIDS requires sending a control message to the server for every suspicious packet to check whether a host is in its cache table or not. If it is not in its cache table, CSIDS uses the voting technique for each host in the LAN, and upon receiving the answers, the server then replies to the requested client with a positive or negative packet. Although the system is effective, the voting process takes time, and this leads to an increase in the delay time between sending the query packet to the server and waiting for the response from the server.

III. ARP SPOOFING

A. ADDRESS RESOLUTION PROTOCOL

Each device connected to the internet has two addresses: an IP address and a MAC address. The IP address is assigned to each device whenever connecting to the internet or local LAN. Furthermore, it is a dynamic address that changes every time the device is connected to any network from a different place, while the MAC address is unique. The ARP protocol is used to find the MAC address according to the intended IP address [39].

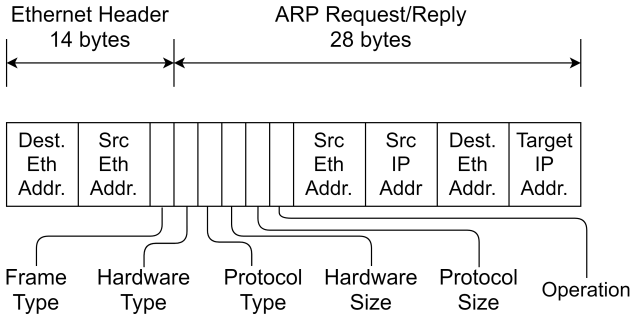

FIGURE 2. ARP Request/Reply format on Ethernet [39].

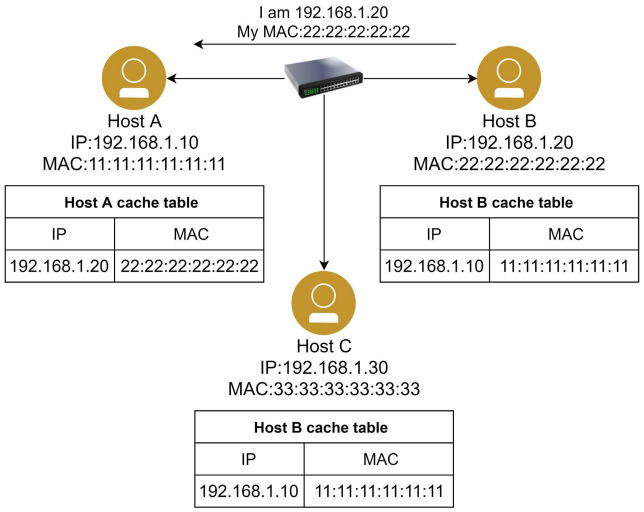

FIGURE 3. ARP cache table for Host A and B.

ARP packet is encapsulated in an Ethernet frame which is constructed whenever the sender needs the MAC address for the destination to communicate with it. By knowing the IP address of the destination, the ARP maps this IP address to the data link layer address (MAC address). Figure 2 represents the format of the ARP request/reply on the Ethernet frame [40].

ARP has been designed to work effectively under regular circumstances, not to deal with malicious hosts. In a normal case, if host A needs to communicate with host B in the same network and the MAC address of host B is unknown to host A. Host A will send an ARP broadcast request to all devices in the network to receive only one ARP unicast reply from host B, and this ARP reply contains the MAC address of host B. Each host has a cache table that recorded all IP and MAC pairs for all hosts' contacts with it before. Figure 3 illustrates an example of three devices cache tables after sending a broadcast ARP request from host A and receiving a unicast ARP reply from host B [24], [41], [42].

B. ARP SPOOFING-BASED MITM

ARP Spoofing is the most common type of spoofing, more damaging and easy to perform. ARP spoofing-based MITM attack exploits the weakness of ARP protocol in which an unauthorized third party secretly accesses the communication between two hosts in the same network. According to ARP protocol, operating systems maintain a cache table of ARP replies from different hosts, in order to minimize the number of ARP requests that are being broadcast. When a host receives any ARP packet, it will automatically update its ARP cache table with the new <IP, MAC> association entry. Note that the <IP, MAC> mapping that is received in the ARP reply should be used to update the ARP cache, even if that sender's IP address is already in the table [8], [43].

Since ARP is a stateless protocol, any host can be spoofed easily by forged messages, as long as there's no correlation between ARP requests and replies. This leads to untrusted <IP, MAC> pairs in the ARP cache table. In an ARP spoofing attack, the attacker can send ARP requests or replies with fake <IP, MAC> mappings (i.e., IP address of the destination and the attacker's MAC address) as an attempt to poison the other hosts' cache tables on the LAN. Therefore, packets redirect to the attacker instead of the legitimate destination. There are two general ways in which an ARP Poisoning attack can be accomplished: the attacker can either wait to see ARP requests for a specific target and send a reply, or transmits a gratuitous ARP reply, which is an unsolicited broadcast packet, which means it is possible to receive a reply without a request matched with it [44]–[46].

In an ARP spoofing scenario, suppose, we have next LAN: the attacker (IP = 192.168.1.30, MAC = 33:33:33:33:33:33), victim Host A (IP = 192.168.1.10, MAC = 11:11:11:11:11:11), and victim Host B (IP = 192.168.1.20, MAC = 22:22:22:22:22:22). For example, if host A wants to communicate with host B, for the first time, which means host A doesn't know Host B MAC address. Therefore host A sends a broadcast request to know the MAC address of host B. All hosts in this LAN update their cache tables with <IP, MAC> of host A. Then host B will send a unicast reply to host A containing its MAC address. To complete ARP spoofing-based MITM attack, The attacker sends an ARP reply message to host A saying that IP: 192.168.48.20 has a MAC address: 33:33:33:33:33:33. Hence, host A cache table is updated with the IP address of host B corresponding to the attacker's MAC address. When host A wants to send a message to Host B it will go to Attacker's MAC address 33:33:33:33:33:33, instead of host B's 22:22:22:22:22:22. Figure 4 illustrates the cache table for host A after receiving a fake reply.

Also, There are five different forms for ARP spoofing to poison a victim ARP cache table as a follows [40], [47]:

1) Unsolicited reply: the attacker sends a forged ARP reply without receiving any ARP request to poison the victim's cache table.

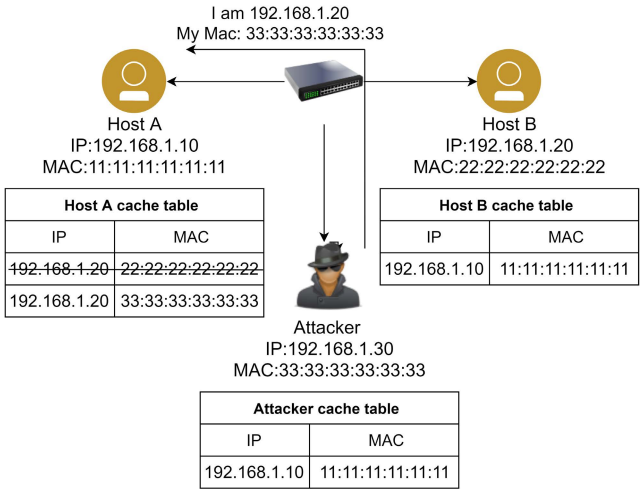

FIGURE 4. Forged ARP reply to host A.

2) Request: The attacker sends a forged broadcast ARP request to all hosts in the network to poison their cache tables.
3) Reply to a request: the attacker sends a forged ARP reply after receiving an ARP request to poison the victim's cache table.
4) Request and reply: the attacker sends a forged ARP request and reply corresponding to that request to poison the victim cache table.
5) Gratuitous ARP: the attacker can also poison the cache table by sending a gratuitous ARP request. The gratuitous ARP is used by a host to find out if another host on the LAN also has the same IP address. When a host gets an IP address from the DHCP server, it sends out a gratuitous ARP frame

IV. PROPOSED SCHEME

This section proposes the overall architecture of D-ARP scheme for detection and prevention of ARP spoofing, then introduces the three modules of D-ARP in detail (i.e., host, admin, and DHCP modules).

A. OVERALL SCHEME ARCHITECTURE

D-ARP is a detection and prevention scheme that is compatible with the original ARP protocol and follows its characteristics for sending a broadcast, receiving a reply, timeout, and cache [8]. As we mentioned in section III, the main problem in ARP protocol, there is no correlation between requests and replies. Therefore, the cache table can be easily updated by forged messages. The main idea of the D-ARP scheme is to solve the problem of replies duplication and prevent the five different forms to poison cache tables as mentioned before, without any changes in the original ARP protocol.

In order to correlate requests and replies, an encrypted key will be injected into the D-ARP requests, and these requests will be sent in parallel with the original one. Therefore, D-ARP should be installed on all hosts within

a LAN. For each host, all signed ARP requests and replies will be recorded in a log file, which is used to correlate signed requests and replies. All requests and replies will also be recorded in the ARP cache table after three levels of validations (i. e., DHCP server, Nmap feature, and log file). Furthermore, the network administrator can create a trusted list that can be inserted automatically into all hosts' cache tables. The main D-ARP steps:

1) The DHCP server is used to grant IP addresses to all hosts within the LAN.
2) Using DHCP server and Nmap feature as an additional two levels of validations to validate all packets' sender and destination <IP, MAC> pairs.
3) This scheme provides a monitoring function for the admin that allows him/her to see all ARP requests and replies packets that are transmitted within the LAN.
4) The admin creates a trusted list which contains static entries on both the DHCP server and ARP cache table trusted entries, and this list can be modified by the admin only. Once any of the hosts in the LAN is active, its ARP cache table will be updated automatically as in the trusted list. It contributes to the reduction of network congestion. For example, host A will directly connect with host B without sending a broadcast ARP request if host B's IP and MAC addresses are in the trusted list.
5) For hosts that are not in the trusted list, they send an ARP request or reply in parallel with the original one.
6) Unsigned ARP replies will automatically be discarded, while signed ones will be recorded in a log file as a third level of validations to automatically detect any form of the five forms to poison the ARP cache table as mentioned above.
7) Once a detection occurred, D-ARP informs the victim and the admin about attack details and provides the evidence to expose the attacker with the three levels of validation. Also, the ARP cache table for the victim will be removed and updated again with the trusted list.

D-ARP consists of three modules host, DHCP server, and admin module in detail.

B. HOST MODULE

D-ARP is a host-based solution, so it manages all ARP packets to detect and prevent ARP spoofing. Figure 5 illustrates the host GUI which is divided into two parts (i.e., left and right). The left part contains:

- IP and MAC addresses of the host itself.
- Interface used to pass a packet.
- The IP address of the destination host which used in sending a broadcast packet.
- Send ARP request button.

The right part contains all received requests and replies information. Here, there are two types of replies:

1) Original reply without key.
2) D-ARP reply with a key.

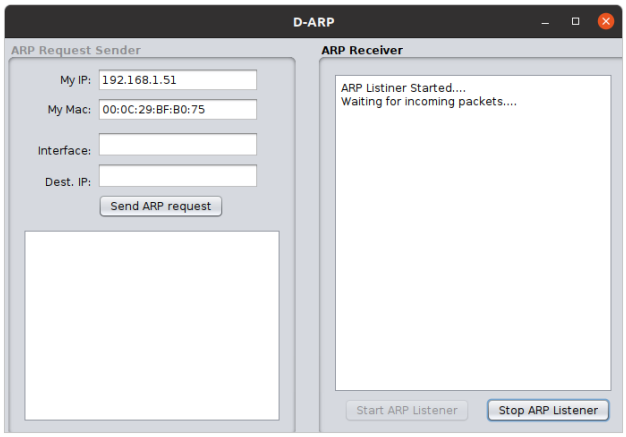

FIGURE 5. Host interface.

1) KEY GENERATION

Key generation is an essential step in the scheme implementation. Many cryptographic hash functions are used to generate digital signatures for preventing attacks, such as the Secure Hash Algorithm (SHA)-3 and Rivest Shamir Adleman (RSA) algorithm [48], [49].

The main purpose for using the key is to bind a request with its reply and to distinguish (i.e., differentiate) between different requests, so this key should be very lightweight, as described before. To achieve this target we use a simple function for light calculations and make the host itself immune to other attacks. Also, simple calculations lead to saving time and avoiding delays in sending requests [50], [51].

Each host generates a different key for each broadcast request using a simple function. The key generates from a collection of random characters consisting of letters, numbers, and symbols, then the output of this function is encrypted by the Cipher algorithm [52], which is a simple cryptographic hash function that doesn't need either complex calculations or time. The complexity of the key generator is constant. The key consists of eight characters such as f#QK2bM, f6Zdv@cl, 9ZWn$q6, and !@NTeK@O. The default size of all ARP packets (i.e., requests or replies) is 28 bytes, while the size of D-ARP packets after adding a key field is 36 bytes as shown in figure 6. Therefore the length of each key is 8 bytes since it is available to extend the ARP packet size to 60 bytes [35] for experimental purposes.

Since D-ARP uses a maximum of 8 bytes generated keys, and this limits the key space to (26 alphabets + 10 numbers + 8 symbols = 44, then $44^8 = 1.4048224e + 13$) possible unique keys. Therefore, each host in the proposed scheme may generate a $1.4048224e+13$ key to increase the variety of generations and reduce the chance of repeating the key. The verification of D-ARP requests depends on the sender's IP address, key, and MAC address together from the Log file.

Another more secure way is to apply a cryptography technique to generate a digital signature for each request's key. We use the RSA cryptosystem combined with the Secure

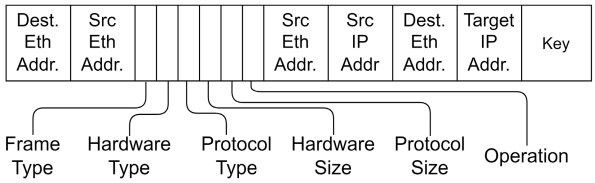

FIGURE 6. D-ARP packet header.

Hash Algorithm (SHA-256) in the proposed scheme [53]. We use (SHA-256) hash function to generate a random key for each request packet then the RSA algorithm is used to encrypt it.

RSA is a secure asymmetric cryptography algorithm for the reliability of data transmission, which uses a public key for encryption and requires another key (i.e., private) known only to the intended destination to be kept secret for decryption [54]. RSA can be used either for encryption purposes (i.e., ensuring that the attacker cannot read packets that sends to the intended destination from a specific source) or for signing requests to guarantee that they are transmitted from a legitimate host. Thus, we ensure that spoofing can't be accomplished by sending a request or gratuitous request.

The proposed scheme aims to protect ARP cache entries from attacks by generating a signature for signing each request. Also, the proposed scheme requires generating public and private keys and encrypting them using RSA for each host. Keys are generated and distributed by the administrator via secure channels. On the sender side, the sender encrypts the signature with its private key because it doesn't know the destination and then sends the broadcast request with a signature. On the destination side, the host decrypts the signature with the sender's public key to send a unicast reply. Therefore, the intended host only can send a unicast encrypted reply and the attack can be easily detected.

2) REQUEST/REPLY SCENARIO

By using the "Send ARP request" button, we can send a D-ARP broadcast packet to all hosts in this LAN. The details of the request will appear at the bottom of the left-hand side. At the same time, other hosts' cache tables will be updated with <IP, MAC> pairs after validation from the DHCP-Nmap list to be compatible with the original ARP protocol. The pseudocode of the proposed sending a request in the sender-side is given in the algorithm 1.

Figures 7 and 8 illustrated a normal case of sending and receiving packets using cipher algorithm and RSA algorithm respectively.

When the destination host receives request packets, it checks the validation of <IP, MAC> pair before update the cache table. Each host has a small file called a log file that contains specific information in the D-ARP header packet such as key, type of packet request or reply, sender IP, sender MAC, destination IP, and destination MAC. After checking the validity of the request, it sends a D-ARP reply and updates the log file with request details. In case of any mismatch,

Algorithm 1 Sending a Request Algorithm (Sender Side)

Input Destination IP address
Output key, ARP broadcast request

```
boolean isExist=checkARPCacheTable(destIP)
if !isExist then
    key=generatekey()
    appendInLogFile(sndrIP,sndrMac, destIP, destMac, key)
    ARPpkt=structARPRequestpkt(sndrIP, sndrMac, destIP, destMac, key)
    sendARPpkt(ARPpkt)
    updateGUI(ARPpkt)
else
    ARPpkt=getInformationFromARPCacheTable()
    sendARPpkt(ARPpkt)
    appendInLogFile(ARPpkt)
    updateGUI(ARPpkt)
end if
```

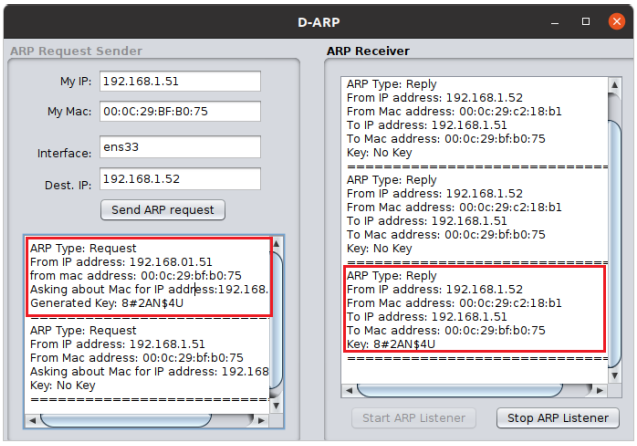

FIGURE 7. Normal case using cipher algorithm.

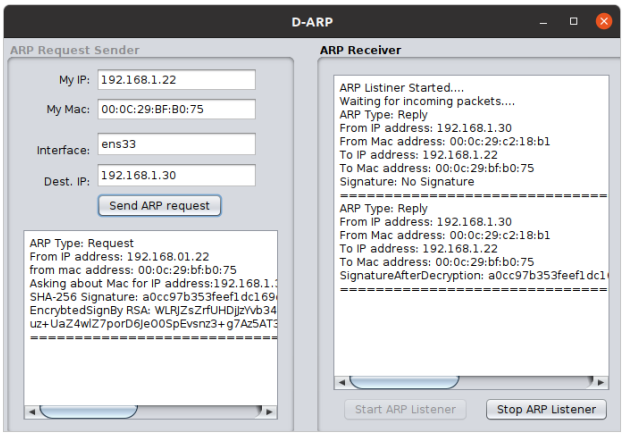

FIGURE 8. Normal case using RSA algorithm.

it drops these packets, which means it is a detection of forged requests. The pseudocode of the proposed destination side algorithm when the host receives a request and detects spoofing from forged requests is given in Algorithm 2.

Algorithm 2 Receiving Requests and Detecting Spoofing (Destination Side)

Input receiving request packet, host IP address
Output update GUI and cache table, send ARP reply

```
ARP pkt = recievedARPpkt()
updateGUI(pkt)
if !!pkt.getKey().isEmpty() then  If pkt.isRequest()
    if pkt.getDestIP() == getMyIP() then
        if isValidated(pkt.getsndrIP(),pkt.getSndrMAC() then
            appendInLogFile(pkt)
            ARPpkt = structAndSendARPReppkt(pkt)
        else
            Alert:Spoof detected Forged Request
            removeFromCacheTableandLog-file(spoofedpkt)
            informAdmin(spoofedpkt)
            isSpoofed = true
        end if
    end if
else
    if isValidated(pkt.getsndrIP(),pkt.getSndrMAC()) then
        appendInLogFile(pkt)
    else
        Alert:Spoof detected Forged Request
        removeFromCacheTableandLog file(pkt)
        informAdmin(pkt)
        isSpoofed = true
    end if
end if
```

After recording signed packets in a log file, D-ARP will check <IP, MAC> pair and keys for all received replies. If there is any conflict in MAC addresses, the host notifies that there is an attacker. Also, an alert message appears containing attacker information, and the host removes these packets from a log file. Then the spoofing data is sent to the admin, and the attacker is blocked automatically.

We classified the ARP spoofing detection into two cases:

- case1 D-ARP detects the attacker: when the host receives a request or a reply and its IP and MAC pair differ from which exists in the DHCP-Nmap list. After detecting the spoofing, we prevent the attacker by blocking the MAC address.
- case2 D-ARP detects the attacker: When the host receives more than one reply, and it detects a conflict or duplication of MAC addresses for the same IP or duplication of keys. After detecting the spoofing, D-ARP collects all MAC addresses from the replies. Through the DHCP-Nmap list, D-ARP easily determines the forged MAC address and prevents it.

The pseudocode of the proposed algorithm in case of receiving replies is given in algorithm 3. When the spoofing

Algorithm 3 Algorithm of Detection of ARP Spoofing on Host Side

Input ARP Pckts
Output Detect the spoofing

```
ARP pkt = recievedARPPkt()
updateGUI(pkt)
signedpkt=pktcontainskey
appendInLogFile(signedpkt)
ARPreq=getARPReqByIPFromLogFile(pkt.SndrIP())
ARP[]reps=getARPRepByIPFromLogFile(pkt.SndrIP())
numOfreps = 0
boolean isSpoofed = false
if !req.isNull() AND !isSpoofed then
    for <ARPrep:reps> do
        if req.getSndrIP()==rep.getDestinationIP() AND req.getSndrMAC()! =rep.getDestinationMAC() then
            numberOfreps++
        end if
    end for
    if numOfreps>2ANDreq.getSign()==rep.getSign() then
        Alert:Spoof detected two reps with same pass-word
        removeFromCacheTableandLogfile(Spoofedreq, Spoofedreps)
        informAdmin(Spoofedreq, Spoofedreps)
        isSpoofed = true
    else if numOfreps>2 AND rep.getSign()! = req.getSign() then
        Alert:Spoof detected two reps with different keys
        removeFromCacheTableandLogfile(Spoofedreq, Spoofedreps)
        informAdmin((Spoofedreq, Spoofedreps)
        isSpoofed = true
    end if
else
    Alert:Spoof detected rep without req
    removeFromCacheTableandLogfile(Spoofedreq, Spoofedrep)
    informAdmin(Spoofedreq, Spoofedrep)
    isSpoofed = true
end if
if !isSpoofed then
    if isValidated(pkt.getSndrIP(), pkt.getSndrMAC()) then
        updateGUIAndARPCacheTable(pkt)
    else
        Alert:Spoof detected Forged rep
        removeFromCacheTableandLogfile(Spoofedreq, Spoofedreps)
        informAdmin(Spoofedreq, Spoofedreps)
        isSpoofed = false
    end if
end if
```

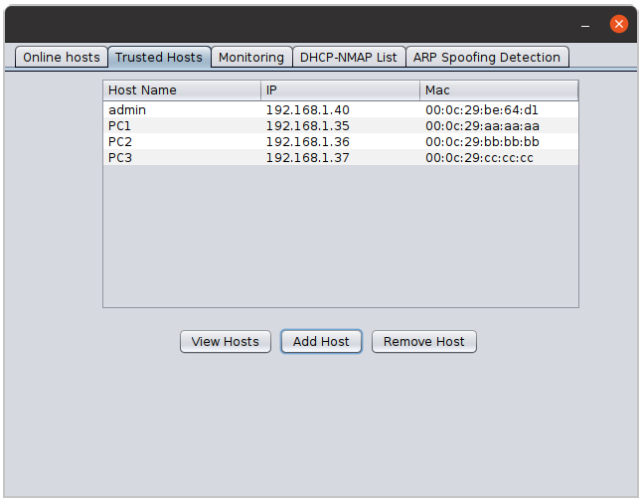

FIGURE 9. Trusted list.

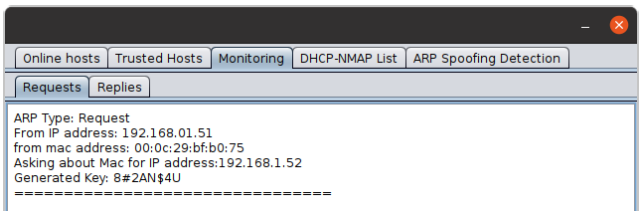

FIGURE 10. Admin monitoring requests.

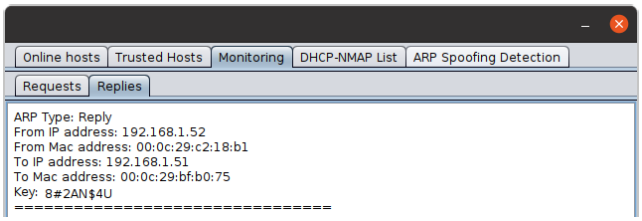

FIGURE 11. Admin monitoring replies.

has occurred, hosts are notified and information will send to the admin who decides the attacker.

C. DHCP SERVER MODULE

The DHCP server is responsible for assigning an IP address to all hosts in the LAN automatically. DHCP has a configuration file which contains static entries. The network administrator will use these entries to create a trusted list. Also, the DHCP server helps to solve the poison of the ARP cache table by gratuitous requests because it guarantees each host has a distinct IP address. In Admin module can view the online hosts, which takes an IP from the DHCP server and at the same time uses the D-ARP scheme. This module helps to detect the attacker, it will be discussed later in detection cases.

D. ADMIN MODULE

As shown in figure 9 Admin GUI has five tabs.

1) **Online hosts:** it is discussed in details in subsection IV-C.
2) **Trusted hosts:** they are added by the network administrator, and it has three fields host name, IP, and MAC addresses. The administrator has privileges to create, add or remove any entry in the trusted list. These entries are inserted into hosts' cache tables automatically while added to the list and removed when removed from the list.
3) **Monitoring:** it contains two tabs, one for requests and the other for replies which carry all information of D-ARP requests and replies packets such as sender IP and MAC. The monitoring category in admin modules collects these packets for checking as shown in figure 10 and figure 11
4) **DHCP-NMAP list:** it has a two-dimensional DHCP and Nmap. Nmap is a Linux feature used to view all up hosts in the LAN with their pairs of <IP, MAC>. The second dimension is the DHCP server, which shows all hosts get their IP addresses from it. As shown in figure 12.
5) **ARP Spoofing detection:** after detecting the spoofing in the host, this tab displays all details about the spoofing, such as spoofing type, when the spoofing occurs, request details, and reply details.

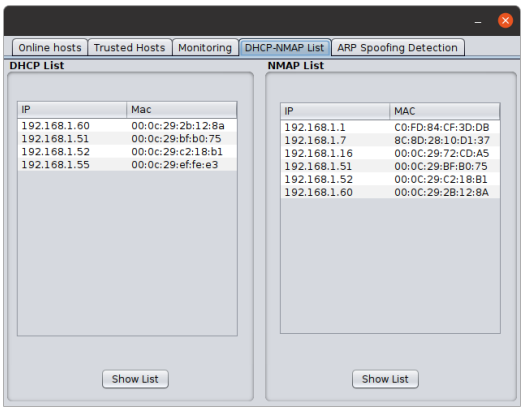

FIGURE 12. DHCP-Nmap lists.

TABLE 1. Network components.

Name	IP Address	MAC Address
Host A	192.168.1.51	00:0c:29:bf:b0:75
Host B	192.168.1.52	00:0c:29:c2:18:b1
Host C	192.168.1.60	00:0c:29:2b:12:8a
Admin	192.168.1.40	00:0c:29:be:64:d1
DHCP Server	192.168.1.16	00:0c:29:72:cd:a5
Attacker	192.168.1.55	00:0c:29:ef:fe:e3

V. EXPERIMENT AND RESULTS

D-ARP is implemented via a virtual workstation pro. The virtual workstation consists of six virtual machines: DHCP server, Admin, Host A, Host B, Host C, and an Attacker as illustrated in figure 13. Ubuntu desktop version 20.04.2.0 is the virtual machines operating system. To verify the effectiveness and creditably of the proposed scheme. The pairs of IP and MAC addresses are shown in table 1.

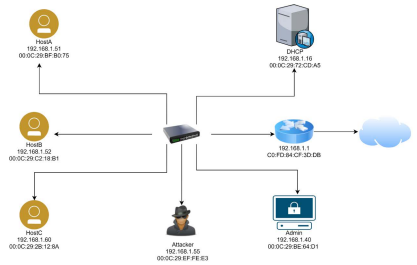

FIGURE 13. Network topology.

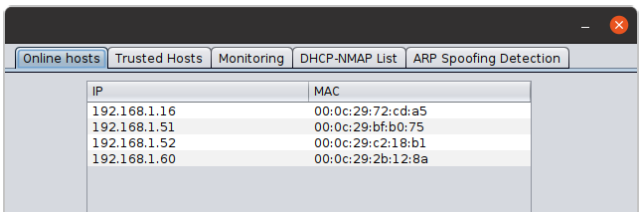

FIGURE 14. Online hosts.

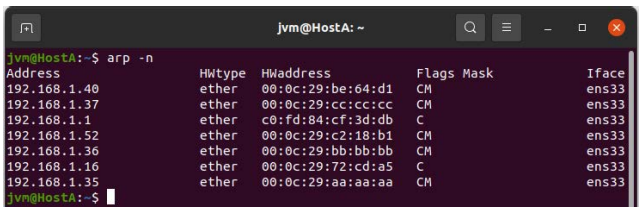

FIGURE 15. Host A cache table.

As shown in table 1, the DHCP server assigns IP addresses to hosts on this LAN from 192.168.1.30 to 192.168.1.150. It is available to increase or decrease this range.

After each host takes its IP address and starts the listener in the D-ARP scheme interface, this will appear on online hosts on the admin interface as shown in figure14.

In order to reduce the number of broadcast requests in the LAN, admin creates a trusted list which is inserted automatically in all hosts cache tables within LAN. We verify it by adding hosts PC1, PC2 with IP addresses 192.168.1.35, 192.168.1.36, 192.168.1.37 and MAC addresses 00:0c:29:aa:aa:aa, 00:0c:29:bb:bb:bb, 00:0c:29:cc:cc:cc respectively. As shown in figure 15, all hosts in the trusted list in figure 9 are inserted in the cache table of host A.

D-ARP displays every ARP packet transmitted within this LAN and takes them into consideration to be compatible with the original ARP.

Then we have the detection scenarios:

A. FIRST SCENARIO

If the victim host received two replies for the same request with the same key and different MAC addresses, an alarm message will be appeared to inform there is an attacker and its MAC address, as shown in figure 16.

Then information in the replies packets will be sent automatically to the admin to identify the attacker itself. As shown in figure17.

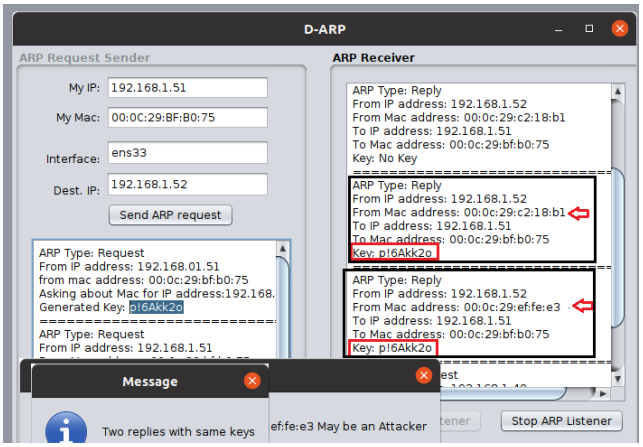

FIGURE 16. Scenario 1: two replies with same key.

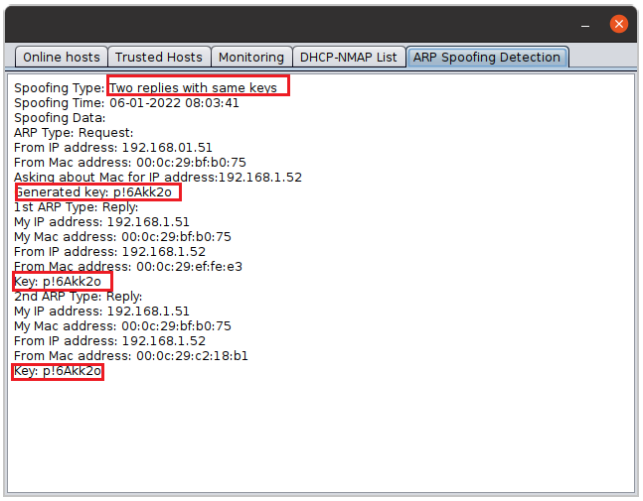

FIGURE 17. Admin: two replies with same key.

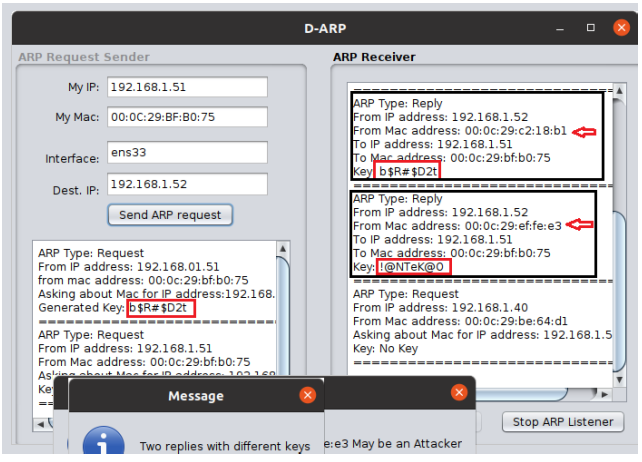

FIGURE 18. Scenario 2: two replies with different keys.

B. SECOND SCENARIO

If the victim host received more than one reply for the same request with different keys and different MAC addresses, an alarm will be appeared to inform there is an attacker and its MAC address as shown in figure18. Also, this information is sent to admin in figure 19.

TABLE 2. Comparison between D-ARP and related methods.

Name	S-ARP [15]	Ref. [55]	Ref. [29]	Ref. [36]	D-ARP
Performance overhead	Yes	Yes	Yes	Yes	No
Static/Dynamic environment	Yes	Yes	Yes	Yes	Yes
Single point of failure	Yes	Yes	No	No	No
Compatibility with original ARP	Yes	Yes	Yes	Yes	Yes
Pre-conditions and drawbacks	Modify in ARP, DHCP protocols, and key distributed	Install server and Increase network traffic	Maintenance cost and construction cost	Modify in SDN parameters, not flexible and complex	Install D-ARP
Defense against gratuitous ARP	N/A	No	No	No	Yes

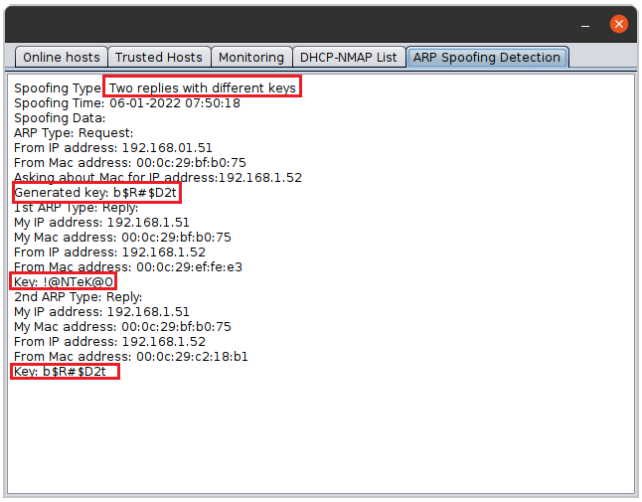

FIGURE 19. Admin: two replies with different keys.

In table 2, we compare between our proposed scheme D-ARP and related methods. This table shows clearly that D-ARP proved its efficiency as it outperforms the previous prevention methods S-ARP [15], [29], [55] and [36].

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose a D-ARP scheme for detecting and preventing ARP spoofing-based MITM which can be installed easily on all hosts. We successfully proved the efficiency and credibility of the D-ARP scheme. It is considered a complete solution to detect all forms of ARP spoofing and identify the attacker without any changes in ARP protocol. D-ARP is effective to reduce network congestion since D-ARP injects a trusted list in hosts cache tables. Finally, D-ARP detects the attacks immediately without any false negative or false positive and prevents them according to the trusted list and DHCP-Nmap list. In future work, we will improve our scheme to detect DDoS and Hijacking attacks and implement it on other platforms.

REFERENCES

[1] J. Johnson, "Internet users in the world 2021," Tech. Rep., 2021. [Online]. Available: https://www. statista.com/statistics/617136/digitalpopulation-worldwide
[2] B. Bhushan, G. Sahoo, and A. K. Rai, "Man-in-the-middle attack in wireless and computer networking—A review," in *Proc. 3rd Int. Conf. Adv. Comput., Commun. Automat. (ICACCA) (Fall)*, Sep. 2017, pp. 1–6.
[3] C. G. Weissman, "Here's why equifax yanked its apps from apple and google last week," Tech. Rep., Sep. 2017.
[4] T. Jabeen, H. Ashraf, A. Khatoon, S. S. Band, and A. Mosavi, "A lightweight genetic based algorithm for data security in wireless body area networks," *IEEE Access*, vol. 8, pp. 183460–183469, 2020.
[5] J.-Y. Kim and H.-J. Kim, "Defining security primitives for eliciting flexible attack scenarios through CAPEC analysis," in *Proc. Int. Workshop Inf. Secur. Appl.*, 2014, pp. 370–382.
[6] I. Green, "DNS spoofing by the man in the middle," Tech. Rep., 2005.
[7] A. M. AbdelSalam, A. B. El-Sisi, and R. K. Vamshi, "Mitigating ARP spoofing attacks in software-defined networks," in *Proc. 25th Int. Conf. Comput. Theory Appl. (ICCTA)*, Alexandria, Egypt, Oct. 2015, pp. 126–131.
[8] C. P. David, *An Ethernet Address Resolution Protocol*, document RFC 826, 1982.
[9] N. Saxena and N. S. Chaudhari, "Secure-AKA: An efficient AKA protocol for UMTS networks," *Wireless Pers. Commun.*, vol. 78, no. 2, pp. 1345–1373, Sep. 2014.
[10] V. Ramachandran and S. Nandi, "Detecting ARP spoofing: An active technique," in *Information Systems Security*, eds. Berlin, Germany: Springer, 2005, pp. 239–250.
[11] G. Jinhua and X. Kejian, "ARP spoofing detection algorithm using ICMP protocol," in *Proc. Int. Conf. Comput. Commun. Informat.*, Jan. 2013, pp. 1–6.
[12] M. Sirajuddin, C. Rupa, and A. Prasad, "A trusted model using improved-AODV in MANETS with packet loss reduction mechanism," *Adv. Model. Anal. B*, vol. 61, no. 1, pp. 15–22, Mar. 2018.
[13] M. Sirajuddin, C. Rupa, C. Iwendi, and C. Biamba, "TBSMR: A trust-based secure multipath routing protocol for enhancing the QoS of the mobile ad hoc network," *Secur. Commun. Netw.*, vol. 2021, pp. 1–9, Apr. 2021.
[14] L. N. R. Group *et al.*, *Arpwatch, the Ethernet Monitor Program; for Keeping Track of Ethernet/IP Address Pairings*. Accessed: Apr. 17, 2019.
[15] D. Bruschi, A. Ornaghi, and E. Rosti, "S-ARP: A secure address resolution protocol," in *Proc. 19th Annu. Comput. Secur. Appl. Conf.*, Dec. 2003, pp. 66–74.
[16] M. Sirajuddin, C. Rupa, and A. Prasad, "An innovative security model to handle blackhole attack in MANET," in *Proc. Int. Conf. Comput. Intell. Data Eng.* N. Chaki, A. Cortesi, and N. Devarakonda, Eds. Singapore: Springer, 2018, pp. 173–179.
[17] V. K. Tchendji, F. Mvah, C. T. Djamegni, and Y. F. Yankam, "E2BaSeP: Efficient Bayes based security protocol against ARP spoofing attacks in SDN architectures," *J. Hardw. Syst. Secur.*, vol. 5, no. 1, pp. 58–74, Mar. 2021.
[18] D. Srinath, S. P. S. Panimalar, A. J. Simla, and J. D. J. Deepa, "Detection and prevention of ARP spoofing using centralized server," *Int. J. Comput. Appl.*, vol. 113, no. 19, pp. 26–30, Mar. 2015.
[19] S. Hijazi and M. S. Obaidat, "A new detection and prevention system for ARP attacks using static entry," *IEEE Syst. J.*, vol. 13, no. 3, pp. 2732–2738, Sep. 2019.
[20] S. Y. Nam, D. Kim, and J. Kim, "Enhanced ARP: Preventing ARP poisoning-based man-in-the-middle attacks," *IEEE Commun. Lett.*, vol. 14, no. 2, pp. 187–189, Feb. 2010.
[21] S. Y. Nam, S. Djuraev, and M. Park, "Collaborative approach to mitigating ARP poisoning-based man-in-the-middle attacks," *Comput. Netw.*, vol. 57, no. 18, pp. 3866–3884, Dec. 2013.

[22] 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.
[23] Y. Zhao, R. Guo, and P. Lv, "ARP spoofing analysis and prevention," in *Proc. 5th Int. Conf. Smart Grid Electr. Autom. (ICSGEA)*, Jun. 2020, pp. 572–575.
[24] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2027–2051, 3rd Quart., 2016.
[25] A. S. A. M. S. Ahmed, R. Hassan, and N. E. Othman, "IPv6 neighbor discovery protocol specifications, threats and countermeasures: A survey," *IEEE Access*, vol. 5, pp. 18187–18210, 2017.
[26] O. S. Younes, "Securing ARP and DHCP for mitigating link layer attacks," *Sādhanā*, vol. 42, no. 12, pp. 2041–2053, Dec. 2017.
[27] D. J. Tian, K. R. B. Butler, J. I. Choi, P. McDaniel, and P. Krishnaswamy, "Securing ARP/NDP from the ground up," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 9, pp. 2131–2143, Sep. 2017.
[28] D. R. Rupal, D. Satasiya, H. Kumar, and A. Agrawal, "Detection and prevention of ARP poisoning in dynamic IP configuration," in *Proc. IEEE Int. Conf. Recent Trends Electron., Inf. Commun. Technol. (RTEICT)*, May 2016, pp. 1240–1244.
[29] B. Prabadevi and N. Jeyanthi, "A mitigation system for ARP cache poisoning attacks," in *Proc. 2nd Int. Conf. Internet things, Data Cloud Comput.*, Mar. 2017, pp. 1–7.
[30] M. Data, "The defense against ARP spoofing attack using semi-static ARP cache table," in *Proc. Int. Conf. Sustain. Inf. Eng. Technol. (SIET)*, Nov. 2018, pp. 206–210.
[31] J. Xia, Z. Cai, G. Hu, and M. Xu, "An active defense solution for ARP spoofing in OpenFlow network," *Chin. J. Electron.*, vol. 28, no. 1, pp. 172–178, Jan. 2019.
[32] S. Singh, D. Singh, and A. M. Tripathi, "Two-phase validation scheme for detection and prevention of ARP cache poisoning," in *Progress Advanced Computing and Intelligent Engineering*. Cham, Switzerland, 2019, pp. 303–315.
[33] H. Y. Ibrahim, P. M. Ismael, A. A. Albabawat, and A. B. Al-Khalil, "A secure mechanism to prevent ARP spoofing and ARP broadcasting in SDN," in *Proc. Int. Conf. Comput. Sci. Softw. Eng. (CSASE)*, Apr. 2020, pp. 13–19.
[34] M. Ren, Y. Tian, S. Kong, D. Zhou, and D. Li, "An detection algorithm for ARP man-in-the-middle attack based on data packet forwarding behavior characteristics," in *Proc. IEEE 5th Inf. Technol. Mechatronics Eng. Conf. (ITOEC)*, Jun. 2020, pp. 1599–1604.
[35] J. J. Kponyo, J. O. Agyemang, and G. S. Klogo, "Detecting end-point (EP) man-in-the-middle (MITM) attack based on ARP analysis: A machine learning approach," *Int. J. Commun. Netw. Inf. Secur.*, vol. 12, no. 3, pp. 384–388, 2020.
[36] 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, Art. no. 106990.
[37] A. Majumdar, S. Raj, and T. Subbulakshmi, "ARP poisoning detection and prevention using Scapy," in *Proc. J. Phys., Conf.*, vol. 1911, 2021, Art. no. 012022.
[38] H. Salim and Z. Li, "A precise model to secure systems on Ethernet against man-in-the-middle attack," *IT Prof.*, vol. 23, no. 1, pp. 72–85, Jan. 2021.
[39] J. S. Meghana, T. Subashri, and K. R. Vimal, "A survey on ARP cache poisoning and techniques for detection and mitigation," in *Proc. 4th Int. Conf. Signal Process., Commun. Netw. (ICSCN)*, Mar. 2017, pp. 1–6.
[40] S. G. Bhirud and V. Katkar, "Light weight approach for IP-ARP spoofing detection and prevention," in *Proc. 2nd Asian Himalayas Int. Conf. Internet (AH-ICI)*, Nov. 2011, pp. 1–5.
[41] Z. Trabelsi and W. El-Hajj, "On investigating ARP spoofing security solutions," *Int. J. Internet Protocol Technol.*, vol. 5, nos. 1–2, pp. 92–100, 2010.
[42] S. Sun, X. Fu, B. Luo, and X. Du, "Detecting and mitigating ARP attacks in SDN-based cloud environment," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 659–664.
[43] Y. Wang, X. Feng, Y. Chen, L. Zhou, Y. Zhu, and J. Wu, "A dual detection method for Siemens inverter motor modbus RTU attack," *J. Comput. Commun.*, vol. 9, no. 7, pp. 91–108, 2021.
[44] S. Frankel, B. Eydt, L. Owens, and K. Scarfone, "Establishing wireless robust security networks: A guide to IEEE 802.11 i," NIST Special Publication, Gaithersburg, MD, USA, Tech. Rep., 2007, pp. 97–800.
[45] S. Mandal, D. A. Khan, and S. Jain, "Cloud-based zero trust access control policy: An approach to support work-from-home driven by COVID-19 pandemic," *New Gener. Comput.*, vol. 39, nos. 3–4, pp. 599–622, Nov. 2021.
[46] S. Hijazi and M. S. Obaidat, "Address resolution protocol spoofing attacks and security approaches: A survey," *Secur. Privacy*, vol. 2, no. 1, p. e49, Dec. 2018.
[47] M. V. Tripunitara and P. Dutta, "A middleware approach to asynchronous and backward compatible detection and prevention of ARP cache poisoning," in *Proc. 15th Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Dec. 1999, pp. 303–309.
[48] M. J. Dworkin *et al.*, "SHA-3 standard: Permutation-based hash and extendable-output functions," Tech. Rep., 2015.
[49] P. Barrett, "Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor," in *Advances in Cryptology CRYPTO*, A. M. Odlyzko, ed. Berlin, Germany: Springer, 1987.
[50] D. Mohamed, "Defense against distributed denial of service attacks in computer networks," Ph.D. dissertation, Graduate School Inf. Sci., Tohoku Univ., Sendai, Japan, 2010.
[51] A. Bremler-Barr and H. Levy, "Spoofing prevention method," in *Proc. IEEE 24th Annu. Joint Conf. IEEE Comput. Commun. Societies*, vol. 1, Feb. 2005, pp. 536–547.
[52] C. Madson and N. Doraswamy, *The ESP DES-CBC Cipher Algorithm With Explicit IV*, document RFC 2405, Nov. 1998.
[53] *FIPS 180-2 Secure Has Standard*, Federal NIST, Gaithersburg, MD, USA, 2002.
[54] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, "PKCS# 1: RSA cryptography specifications version 2.2," in *Internet Engineering Task Force, Request for Comments*, vol. 8017. 2016, p. 72.
[55] S. Kumar and S. Tapaswi, "A centralized detection and prevention technique against ARP poisoning," in *Proc. Title: Int. Conf. Cyber Secur., Cyber Warfare Digit. Forensic (CyberSec)*, Jun. 2012, pp. 259–264.

SABAH M. MORSY was born in Sohag, Egypt. She received the B.S. degree in computer science from the Faculty of Science, Assuit University, Assuit, Egypt, in 2011, and the Diploma degree in cloud computing from the Information Technology Institute, Assuit, in 2013. Since 2014, she has been a Teaching Assistant with the Information Technology Department, National Egyptian E-Learning University. Her research interests include security fields, machine learning, and artificial intelligence.

DALIA NASHAT received the Ph.D. degree in networks security from Tohoku University, Japan, in 2010. She was an Assistant Professor in computer science with the Faculty of Science, Assuit University, Assuit, Egypt, from 2010 to 2020, where she is currently an Assistant Professor with the Department of Information Technology, Faculty of Computers and Information. She was an Assistant Professor with Taif University, Saudi Arabia, from 2011 to 2014. Her research interests include networks security, intrusion detection, and signal processing.