Enhancing Network Security: Mitigating ARP Spoofing Threats through Advanced Prevention Strategies

Himaneesh Yadala, Vaibhav Anurag, Preetha Sarkar

Department of Computer Science and Engineering
National Institute of Technology Karnataka
Surathkal, Mangalore, India

9739955000, 79799 15189, 6292253051

himaneeshyadala.221cs264@nitk.edu.in, anuragvaibhav.221cs258@nitk.edu.in,
preethasarkar.221cs236@nitk.edu.in

March 20, 2024

Abstract

Securing them from malicious attacks is crucial with the increasing reliance on interconnected computer networks. ARP spoofing, a method where attackers manipulate ARP tables for their benefit, poses a significant threat. Current preventative measures like static ARP entries, network segmentation, and secure ARP offer a fair amount of protection, but they can be inconvenient to manage across networks consisting of a large amount of devices or have other limitations.
This paper proposes a new approach to ARP spoofing prevention that addresses these challenges. We aim to actively detect and prevent spoofing attempts by constantly analyzing every ARP packet on the network. Our algorithm uses ICMP packets to get the physical address of the attacker and spoofed device. This method provides a more reliable and adaptable defense against ARP spoofing, strengthening overall network security.

1. Introduction

Network security is vital for protecting sensitive information and ensuring reliable communication. One significant threat to network security is Address Resolution Protocol (ARP) [1] spoofing, where attackers manipulate data to reroute traffic. ARP packets are extremely vulnerable as the responses are automated and can easily be faked. Efficient and robust ARP spoofing prevention methods are crucial to safeguard network integrity. Implementing such techniques minimizes security risks, creating a safer and more reliable user experience on the network.
This paper discusses the use of static MAC entries, this method maintains an ARP table [2] by manually configuring mappings between IP addresses and MAC addresses [3], unlike dynamic ARP entries that are automatically populated through ARP requests and responses, static entries remain unchanged until manually updated or removed. This method is simple to setup on small networks, however, this has a lack of scalability and requirement of considerable time and effort to manage entries on large networks.
This paper also mentions the use of a more sophisticated technique called secure ARP, it implements a novel approach by authenticating ARP messages using the Digital Signature Algorithm (DSA). This safeguard significantly enhances network security. Only devices with valid signing keys can send authenticated messages, making it much harder for attackers to spoof identities. Although, the way S-ARP works (using a PKI and a trusted server) makes things more complicated, it requires additional infrastructure to be set up on the network.

The rest of this paper is organised as follows. The first section goes over a few related works and publications in this domain and explains our motivation for producing this paper. Subsequently, section 2 introduces the proposed solution by utilising different modules. This is followed by section 3, where the application of algorithm is showcased in a simulated environment. Section 4 compares a variety of existing methods to our

technique according to different parameters. Finally, the paper concludes with the future work and scope of this implementation.

2. Related Works

2.1. Unequal Request Reply Algorithm:

Marco Carnut et. al. [4] proposed an algorithm, where the switch maintains a counter for ARP packets. The switch is supposed to maintain four counters.
Counter 1: gets appended whenever an ARP request is received at the port.
Counter 2: gets appended whenever ARP request is transmitted out.
Counter 3: gets appended whenever ARP reply is received.
Counter 4: gets appended whenever ARP reply is going out.
For a time interval, we will calculate delta values of counters 2 and 3, this imbalance (difference between the deltas of counter 2 and counter 3) will be analysed, let the imbalance of the ARP request and replies at a port p be denoted by im[p].

$$\text{im[p]} = \text{delta(counter2)-delta(counter3)}$$

Now, if the imbalance is positive, then the number of requests sent out of that particular time interval is greater than the replies received, which is alright. But when the imbalance becomes negative, it means that ARP replies are greater than ARP requests. This directly indicates that an ARP attack is going on.
As shown in figure-1, since the switch maintains counters for the ports, we are able to pinpoint the port where the attack is going on.
This method is easy to implement and gives the exact port location where ARP spoof is happening. However, maintaining a counter and analyzing it for every port for every ARP request and response adds significant overhead, also creating privacy issues because the network usage pattern of a user is being stored and analysed.

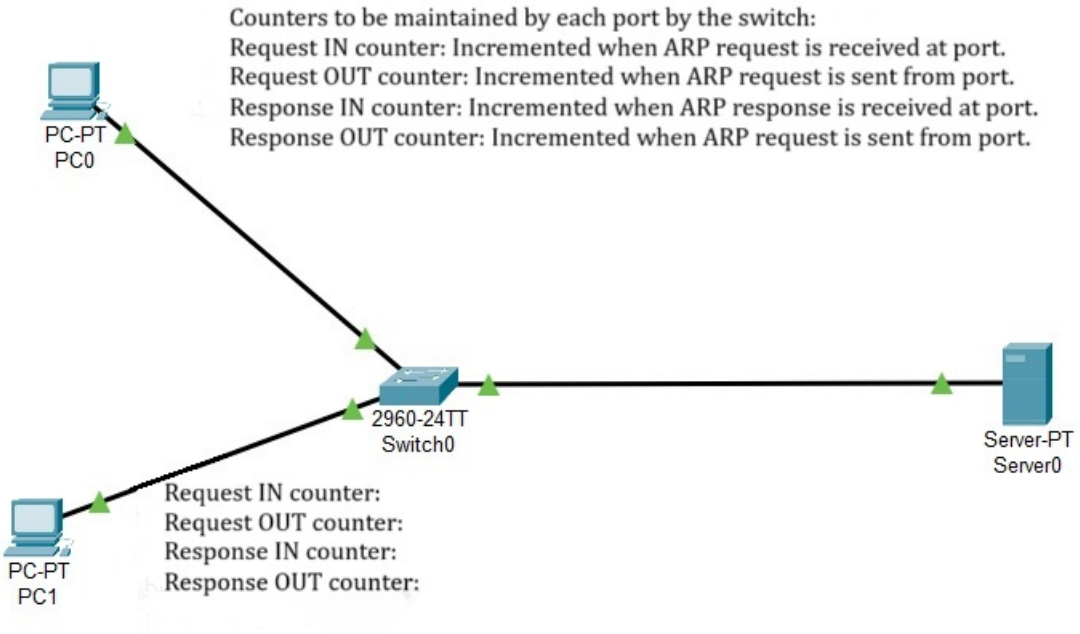

Figure 1: Working of Unequal Request Reply Algorithm

2.2. Static MAC Entries:

Khalid M. Amin et. al. [5] proposed a paper about maintaining **Static MAC Entries**. Instead of relying on dynamic look-ups, network administrators can create fixed entries within the ARP cache using a TCP/IP tool.

These static ARP caches can be used to facilitate ARP requests for frequently called IP addresses.
The association of these MAC addresses remains unaffected even after dynamic network changes or device movements. This is comparatively an effective solution to preventing ARP spoofing on small networks consisting of a small number of connected devices. This method bypasses the dynamic ARP process where devices broadcast requests and responses to discover each other's MAC addresses. [6]
By our understanding, static MAC entries could have been a potential solution. Their apparent simplicity and ease of implementation, particularly for smaller networks, has a certain appeal. Additionally, the potential for improved performance and predictability in specific scenarios seemed advantageous. However, all devices outside the pre-defined list are vulnerable to dynamic spoofing attacks.
The figure-2 shows a few static entries on the network our device is connected to, accessing the ARP entries can be done by the "TCP/IP Control Panel" on MAC OS and by the command "arp -a" on the command prompt of a device running on Windows OS.

This is a simple solution that skips dynamic ARP requests, which might offer slight speed improvements in specific scenarios. Additionally, manually setting entries is a straightforward process, especially for small networks with few devices.
However, attackers can exploit this technique by repeatedly spamming spoofed messages, overriding static entries before communication occurs, rendering them useless. As a network grows or changes, manually managing entries becomes complicated and error-prone, introducing new security risks.

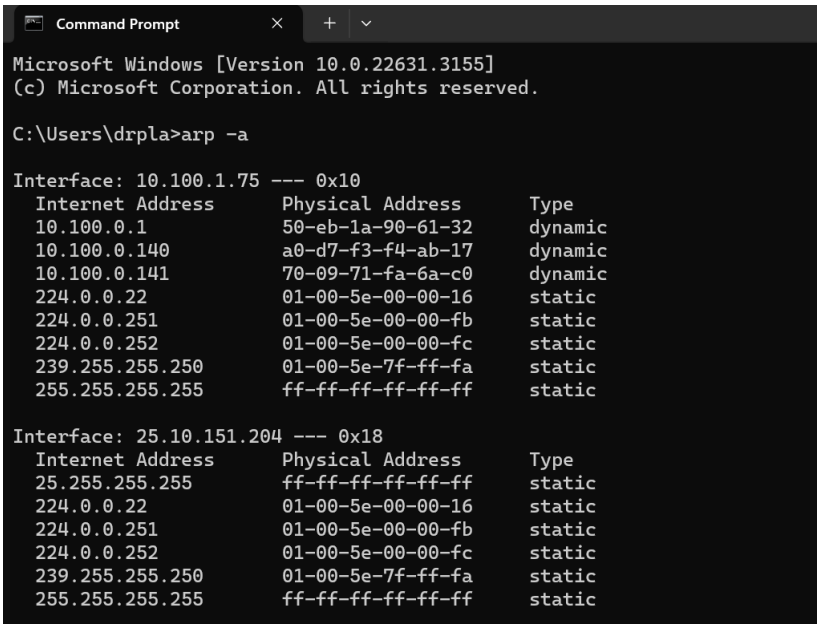

Figure 2: Static MAC Entries on a Network

2.3. Secure ARP

Shimpy Goyal et. al. [7] improved the previous technique's susceptibility to ARP spoofing attacks by using **Secure ARP**. This method operates by employing a public key infrastructure (PKI) to authenticate devices within the network. When a device seeks to communicate with another device, it transmits an S-ARP request comprising its MAC and IP addresses, alongside a digital signature generated using its private key. Upon receiving the request, the recipient device utilizes the sender's public key to validate the signature, ensuring the sender's claimed identity.
The figure-3 shows the working of secure ARP, the granular access control facilitated by S-ARP's digital signatures allows for the selective validation of device identities. This selective validation serves to build trust amongst authorized devices and effectively hinders unauthorized devices, consequently strengthening network security.
As per our understanding, traditional ARP lacks authentication, making it vulnerable to attackers forging messages and impersonating devices. Hence, S-ARP addresses this by adding digital signatures to ARP messages. Devices with valid signing keys can send authenticated messages, verifiable by others. Adding this authentication layer benefits overall network security, making it significantly harder for attackers to exploit ARP vulnerabilities and access sensitive data.

Since S-ARP utilizes digital signatures like DSA [8]to authenticate ARP messages, ensuring only authorized devices can communicate. This significantly reduces the success rate of spoofing attacks compared to traditional, trust-based ARP.
Althought, compared to standard ARP, S-ARP's reliance on a public key infrastructure (PKI) and a trusted third-party server introduces increased complexity and requires additional infrastructure. This can translate to higher costs and greater difficulty in managing and administering the network.

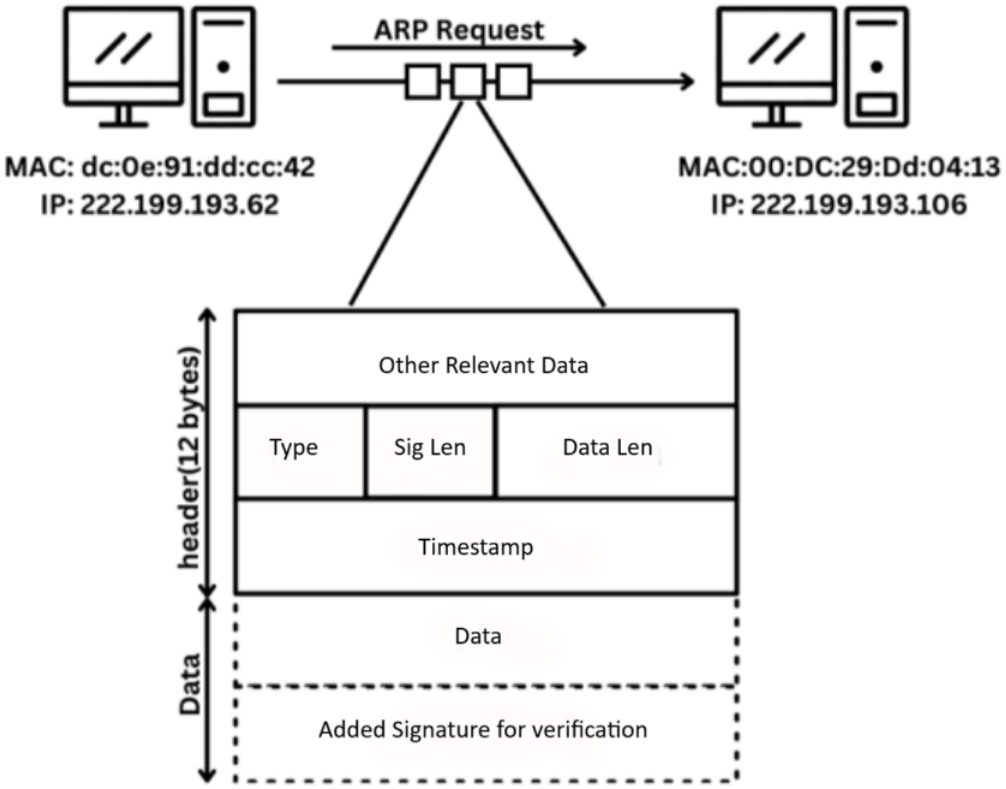

Figure 3: Working of Secure ARP

3.Design and Implementation of ARP-ICMP Detection Algorithm Design

3.1.1 Terms and Definitions:

1. **IP-MAC Address Mapping/Binding**: It is the process of linking a particular IP address with a particular MAC address such that it creates a one-one relationship between the physical address and logical address.

2. **DHCP Snooping Database**: A database which contains records of trusted IP and MAC address pairs.Also known as DHCP snooping binding table.

3. **Invalid ARP Packet**: If the destination IP and MAC addresses do not match with the corresponding IP and MAC address in the database the packet is considered invalid. Matching is done based on the IP addresses. These packets are ARP spoofed.

4. **Valid ARP Packet**: If the destination IP and MAC addresses match with the corresponding IP and MAC address in the database the packet is considered valid.

5. **Victim Host**: The device on the network who is targeted by the attacker.

6. **Detection Host**: A device on the network whose sole purpose is to continuously monitor the network. It prevents and intercepts any attack attempts on devices in the network.

7. **Attacking Host**: The device on the network that is trying to spoof using ARP aiming for attacks like Man-in-the-Middle Attack.

8. **ICMP Ping**: Ping is a tool utilized to ascertain the accessibility of a host on an Internet Protocol (IP) network and to calculate the round-trip time (RTT) for messages transmitted from the originating host to a target computer.

9. **RTT(Round Trip Time)** - The time taken by a data packet to travel from source computer to destination computer and back to source computer is known as RTT.

10. **DAI Enabled Switches**: It is a switch in the network which is configured with DAI to prevent an ARP attack. It allows only valid ARP packets through the network and drops an invalid ARP packet. A central switching device called hub is DAI enabled.

3.1.2 Algorithm Structure:

The provided algorithm can be divided into the six following structures each with its own specific set of functionalities:

1.ARP Packet Catcher Module: It captures all the ARP request and response packets generated within the network. The packets are then transferred to the Invalid Packet Detector Module to analyze the validity of the packets.

2.Invalid Packet Detection Module: This module is used to classify packets as valid or invalid.The result is sent to the response module. Packets with new IP-MAC addresses are forwarded to the ARP Spoof Verification Module.

3.DHCP Snooping Database: A database created to store all the legitimate IP-MAC bindings, given to the devices in the network by DHCP.

4.ARP Spoof Verification Module: The new ARP packets received from the Invalid Packet Detector Module are analyzed by this module.

5.Response Module: Gives the appropriate response in correspondence with our result from the Invalid Packet detector Module and ARP Spoof verification Module. Packets are then either declared valid or invalid.

6.Periodic Network Monitor Module: Periodically checks if the network is infiltrated by outside means or not.

FLOWCHART

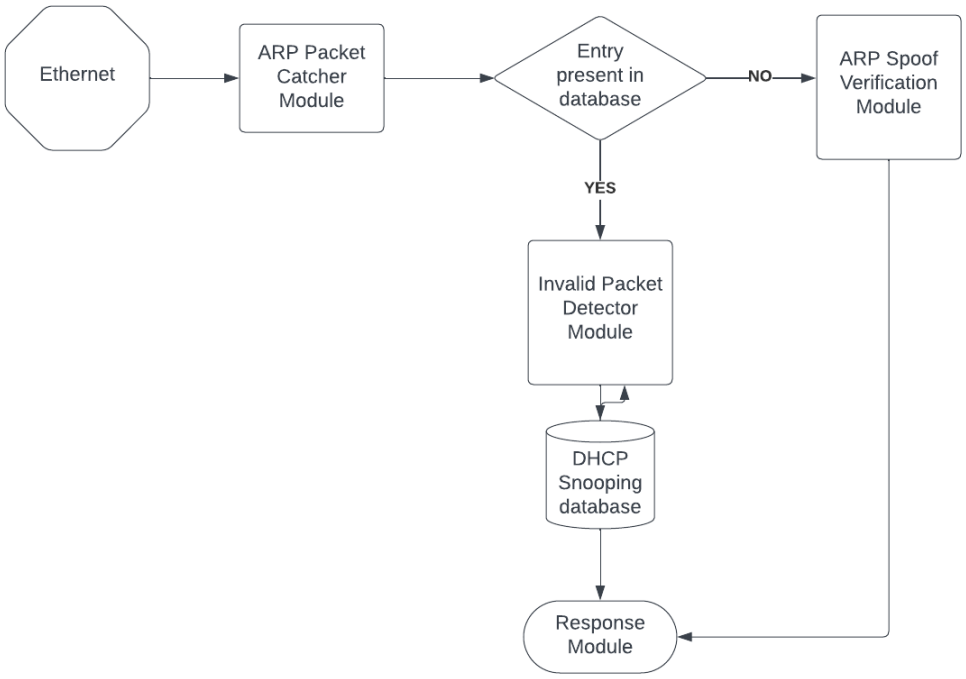

Figure 4: Working of ARP Spoofing Detection Model

3.1.3 Invalid Packet Detection Module:

The ARP header of an ARP packet contains the following addresses:
1. Source IP address
2. Source MAC address
3. Destination IP address
4. Destination MAC address

Figure 5 demonstrates a network, with a DHCP server responsible for managing a DHCP database. As shown in Figure 6, this database functions to retain the valid associations of IP-MAC addresses belonging to devices within the network infrastructure. In an invalid ARP packet, the destination IP address is of the victim host while the destination MAC address is of the attacker host. In order to detect this discrepancy, the MAC address in the ARP packet is compared with the MAC address stored in our DHCP Database. If addresses don't match, it implies the occurrence of an ARP attack.
As illustrated in Figure 7, a hub in the network is configured with DAI to drop the packet if it is spoofed.If the entries in the ARP header match with the database, the switch passes the packet to its required destination.
If IP-MAC address map in the ARP header has no entry in the database, the ARP spoof verification module is used to validate the packet.

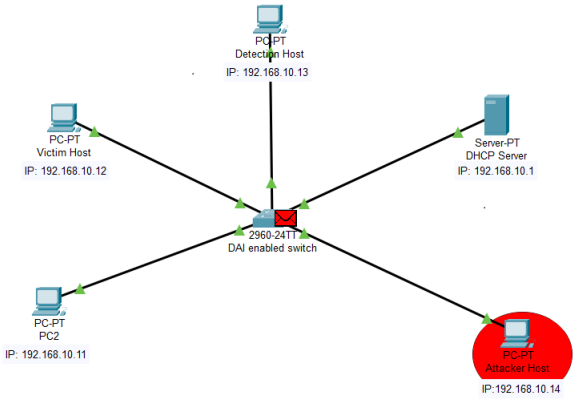

Figure 5: Simulation of DAI enabled network

```
Switch#show ip dhcp snooping binding
MacAddress          IpAddress        Lease(sec)  Type           VLAN  Interface
------------------  ---------------  ----------  -------------  ----  --------------------
00:02:16:59:10:57   192.168.10.11    86400       dhcp-snooping  1     FastEthernet0/2
00:90:0C:B0:78:D2   192.168.10.12    86400       dhcp-snooping  1     FastEthernet0/3
00:D0:97:B7:9A:2B   192.168.10.13    86400       dhcp-snooping  1     FastEthernet0/1
Total number of bindings: 3
```

Figure 6: DHCP Snooping Database

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Failed	ATTA...	PC1	ICMP		0.000	N	0	(edit)	(delete)

Figure 7: ICMP Ping Packet

3.1.4 ARP Spoof Verification Module:

We use ICMP probing in order to detect an attack. Since the possibility of an attack is unknown the result of this module can be divided into following three cases:
1. ARP attack with no IP routing enabled
2. ARP attack with IP routing enabled
3. No ARP attack(Valid MAC address)

Case 1: Detection host creates an ICMP echo request packet using the MAC and IP address provided in the ARP header.This packet is sent to the attacker host. In the attacker's computer the Data Link Layer accepts the packet(since MAC addresses are the same) while it is rejected by the Network Layer(due to different IP addresses). The packet is rejected and no ICMP echo response is received by the detection host. The ARP packet is considered invalid and dropped.

Case 2: Detection host creates an ICMP echo request packet using the MAC and IP address provided in the ARP header. This packet is sent to the attacker host(because of the MAC address). In the attacker host's computer, it sees that the MAC address of the packet matches its own MAC address but the IP address does not match. Since , IP routing is enabled, it routes the packet to the device with the correct IP address.
Now, when the device receives the Echo request packet, it responds to it. After this, the detector module must sniff for the packet with the same identifier and sequence number as that of the request packet. When found, it compares the MAC address of source in the echo response packet with that in the ARP packet. If they don't match then, we are sure that an attack is taking place. The program then adds the attacker's MAC address to the untrusted database of the switch, blocks the port where the attack was taking place, and notifies the administrator about the attack.

Case 3: If the host we suspected was legitimate, then the IP-MAC address pair in the sniffed ARP packet must be correct. So, the detection host sends the ICMP request packet to the IP-MAC address in the ARP packet. Then, the suspected host receives the packet and responds with an echo reply packet. If the host was legitimate, the IP-MAC of source in the echo reply packet will be same as that in the IP-MAC of the ARP

packet. In this case the program will add this binding as a valid one in the DHCP database.

FLOWCHART

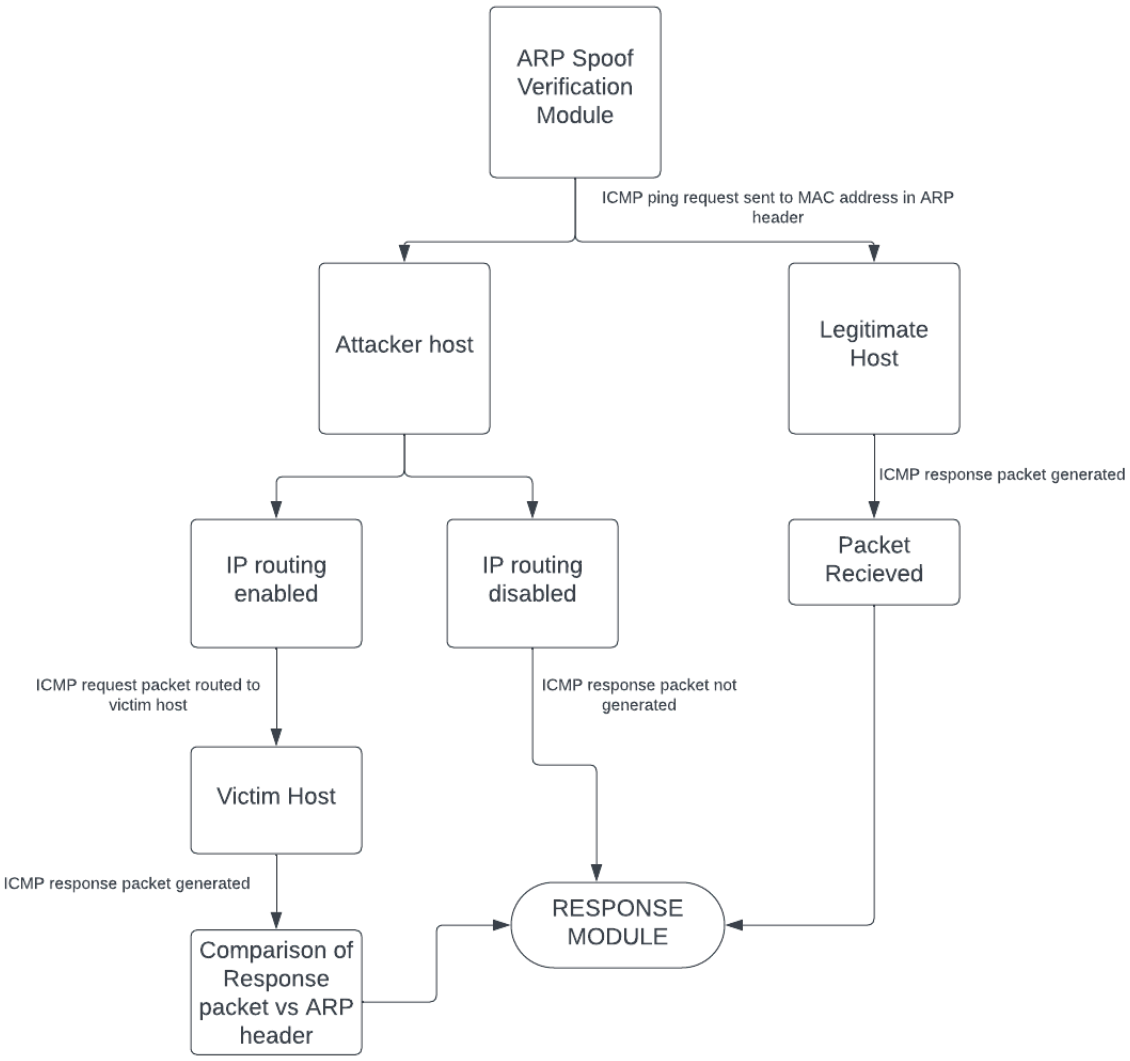

Figure 8: Working of ARP Spoof Verification Module

3.1.5 Periodic network monitor module:

There might be some advanced attackers that still bypass our security system. For that case, we will monitor the network periodically by sending Echo request packets to every device from the detection host. The detection host will have a database of the devices MAC, IP and average RTT. We will define a threshold limit for every device in the network.
Let the average RTT value for a device be RTT(avg), the received RTT be RTT(curr), and the threshold limit be L.
If RTT(curr) > RTT(avg) + L, we respond by notifying the administrator. Then we compare the ARP cache table of the expectedly attacked device and compare the entries with that of the DHCP cache table. If the entries are the same, we respond that no attack is taking place. If any differences are found, we block the port, add the MAC address of the attacker to the untrusted database of the switch, and block the port where the attacker was connected. Then the program notifies the administrator about the attack.

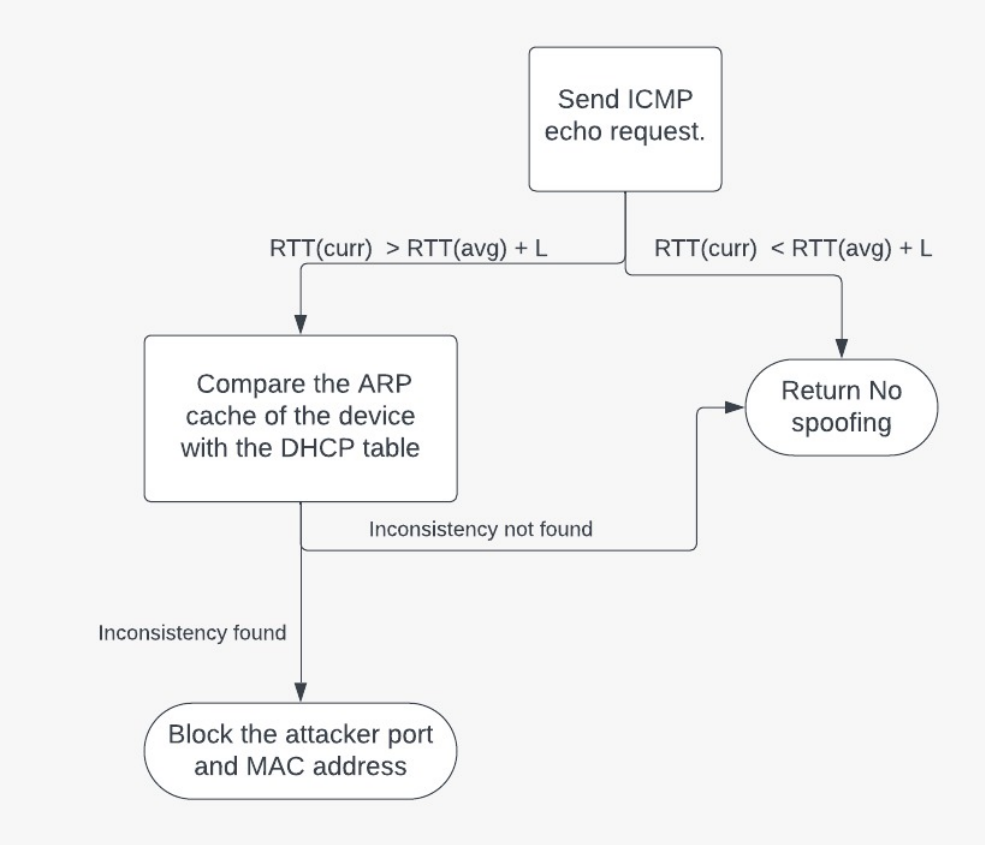

Figure 9: Working of Periodic Network Monitor Module

Implementation

This project uses Scapy, a Python library used for crafting, manipulating, and transmitting network packets. Scapy's interface allows users to work directly with network protocols at the fundamental packet level. These features made Scapy an ideal tool for sniffing network traffic and analyzing ARP spoofing attacks within this project.

3.2.1 Invalid packet Detection Module

Algorithm:

1. The function takes a packet as input.

2. If it is an ARP packet,Scapy is used to check the operation code(op) of the packet. If packet[ARP].op=1 it is a request packet, if packet[ARP].op=2 it is a response packet

3. It checks if packet[ARP].psrc and packet[ARP].hwsrc form a valid entry in the DHCP database or not.

4. If packet[ARP].psrc and packet[ARP].hwsrc do not form a valid entry the packet is declared invalid (Figure 10).

3.2.2 ARP Spoof Verification Module

Algorithm:

1. The physical address and hardware address is extracted from the ARP header and stored in two variables.

2. The following code snippet shows the extraction and storing explained in Step 1.

arp_src_ip=packet[ARP].psrc

arp_src_mac=packet[ARP].hwsrc

3. AN ICMP echo request packet is created using arp_src_mac=packet[ARP].hwsrc

4. If no ICMP response is received the packet is considered invalid (Figure 11).

5. If ICMP response is received, the MAC address and IP address is again extracted from the response packet. The values are compared with the IP and MAC address stored in arp_src_ip and arp_src_mac respectively.

6. If the values match the packet is considered valid.

7. If the values do not match the packet is considered invalid.

4. Result and Analysis

We will be using Wireshark and Scapy library in Python, to demonstrate our algorithm in action. Through Scapy we create and send ARP and ICMP packets on our designated network.
To simulate an ARP spoofing attack, a custom spoofed ARP response packet is transferred over the network. The algorithm creates a DHCP database and generates ICMP packets to validate all the incoming ARP responses.

```
preetha@preetha-IdeaPad-3-15ALC6-Ub:~/ARP$ sudo python3 arp_final.py
MAC Address: 72:8b:3f:cc:4c:5f --> IP Address: 192.168.156.179
MAC Address: 30:03:c8:09:cd:4b --> IP Address: 192.168.156.28
Wrote DHCP database

ARP Request: 192.168.156.28 is asking about 192.168.156.78
MAC Address: 30:03:c8:09:cd:4b
ARP Response: 192.168.156.78 is sending to 192.168.156.28
MAC Address: 00:45:e2:d6:4c:05
Valid response,ARP packet is valid

ARP Response: 192.168.156.28 is sending to 192.168.156.78
MAC Address: 00:11:22:33:44:55
Invalid Response,ARP packet is invalid

ARP Request: 192.168.156.161 is asking about 192.168.156.78
MAC Address: 62:5c:81:31:d4:78
ARP Response: 192.168.156.78 is sending to 192.168.156.161
MAC Address: 00:45:e2:d6:4c:05
Valid response,ARP packet is valid

^CExiting
```

Figure 10: Invalid Packet Detection Module in Action

```
preetha@preetha-IdeaPad-3-15ALC6-Ub:~/ARP$ sudo python3 arp_final.py
Wrote DHCP database

ARP Response: 192.168.156.28 is sending to 192.168.156.78
MAC Address: 00:11:22:33:44:55
Entry not present in database
Begin emission:
Finished sending 1 packets.

Received 20 packets, got 0 answers, remaining 1 packets
No ICMP echo reply received
 ARP packet is invalid
```

Figure 11: ARP Spoof Verification Module in Action

An ARP spoofed packet is sent over the network to test the working of our algorithm. Figure 12 depicts the capture of the ARP spoofed packet by the detection host.

```
preetha@preetha-IdeaPad-3-15ALC6-Ub:~$ arp -a
_gateway (192.168.156.161) at 62:5c:81:31:d4:78 [ether] on wlp1s0
? (192.168.156.28) at 30:03:c8:09:cd:4b [ether] on wlp1s0
preetha@preetha-IdeaPad-3-15ALC6-Ub:~$ arp -a
_gateway (192.168.156.161) at 62:5c:81:31:d4:78 [ether] on wlp1s0
? (192.168.156.28) at 00:11:22:33:44:55 [ether] on wlp1s0
preetha@preetha-IdeaPad-3-15ALC6-Ub:~$
```

Figure 12: ARP Spoofing Verification

All the valid and invalid ARP packets are demonstrated on wireshark to monitor the movement from one device to another (Figure 13). While Wireshark captures all packets, our algorithm excels in validating them, a capability that sets it apart.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	62:5c:81:31:d4:78	CloudNetwork_09:cd:…	ARP	42	Who has 192.168.156.28? Tell 192.168.156.161
2	0.000069	CloudNetwork_09:cd:…	62:5c:81:31:d4:78	ARP	42	192.168.156.28 is at 30:03:c8:09:cd:4b
3	18.739403	62:5c:81:31:d4:78	CloudNetwork_09:cd:…	ARP	42	Who has 192.168.156.28? Tell 192.168.156.161
4	18.739446	CloudNetwork_09:cd:…	62:5c:81:31:d4:78	ARP	42	192.168.156.28 is at 30:03:c8:09:cd:4b
5	48.639331	62:5c:81:31:d4:78	CloudNetwork_09:cd:…	ARP	42	Who has 192.168.156.28? Tell 192.168.156.161
6	48.639398	CloudNetwork_09:cd:…	62:5c:81:31:d4:78	ARP	42	192.168.156.28 is at 30:03:c8:09:cd:4b
7	62.698603	CloudNetwork_09:cd:…	Broadcast	ARP	42	Who has 192.168.156.78? Tell 192.168.156.28
8	62.704580	CyberTANTech_d6:4c:…	CloudNetwork_09:cd:…	ARP	42	192.168.156.78 is at 00:45:e2:d6:4c:05
9	62.706348	CloudNetwork_09:cd:…	CyberTANTech_d6:4c:…	ARP	42	192.168.156.28 is at 00:11:22:33:44:55
10	63.958200	62:5c:81:31:d4:78	CloudNetwork_09:cd:…	ARP	42	Who has 192.168.156.28? Tell 192.168.156.161
11	63.958238	CloudNetwork_09:cd:…	62:5c:81:31:d4:78	ARP	42	192.168.156.28 is at 30:03:c8:09:cd:4b
12	66.175951	CloudNetwork_09:cd:…	Broadcast	ARP	42	Who has 192.168.156.78? Tell 192.168.156.28
13	66.357812	CyberTANTech_d6:4c:…	CloudNetwork_09:cd:…	ARP	42	192.168.156.78 is at 00:45:e2:d6:4c:05
14	66.359812	CloudNetwork_09:cd:…	CyberTANTech_d6:4c:…	ARP	42	192.168.156.28 is at 00:11:22:33:44:55
15	79.872234	62:5c:81:31:d4:78	CloudNetwork_09:cd:…	ARP	42	Who has 192.168.156.28? Tell 192.168.156.161
16	79.872291	CloudNetwork_09:cd:…	62:5c:81:31:d4:78	ARP	42	192.168.156.28 is at 30:03:c8:09:cd:4b
17	80.893237	CyberTANTech_d6:4c:…	Broadcast	ARP	42	Who has 192.168.156.28? Tell 192.168.156.78
18	80.893274	CloudNetwork_09:cd:…	CyberTANTech_d6:4c:…	ARP	42	192.168.156.28 is at 30:03:c8:09:cd:4b
19	100.351289	62:5c:81:31:d4:78	CloudNetwork_09:cd:…	ARP	42	Who has 192.168.156.28? Tell 192.168.156.161
20	100.351326	CloudNetwork_09:cd:…	62:5c:81:31:d4:78	ARP	42	192.168.156.28 is at 30:03:c8:09:cd:4b

Figure 13: ARP Packet Captured

Comparison Between Existing Methods and Our Algorithm

Comparison Table				
Metrics	Secure-ARP	Unequal Request Reply Algorithm	Static MAC Entries	ARP-ICMP Detection Algorithm
Resource Requirements	Requires more resources than static entry approach. Cryptographic keys or certificates can add some memory overhead.	Generates significant overhead to switches. Leads to increased latency.	Minimal. Each static address requires very less memory.	ICMP packet produced only when IP-MAC biding is not found in DHCP database. So it has reduced latency.
Scalability	The processing overhead can become noticeable in large networks with high traffic volume, potentially impacting overall network performance.	For large networks, maintaining the counters becomes very costly.	Managing large amounts of entries can become is difficult to do manually, even with centralized management tools.	It works efficiently only on DHCP enabled network. In other networks, the latency is too high.
Complexity of implementation	More complex. Requires understanding the specific protocol and configuring authentication settings (keys, certificates, etc.). Usually involves more steps and specialized knowledge.	Simple to implement. Maintaining four counters for each port is easy.	Relatively simple. Requires manually adding entries for each device, but some switches offer batch configuration or centralized management tools.	Complex. It requires understanding of different layers of the network, how ARP protocol work and requires extracting MAC address from ICMP packets is complex.
Troubleshooting	More complex. Requires understanding of authentication errors, certificate issues, and potential compatibility problems. Troubleshooting logs and debugging tools become crucial.	Difficult. Gives information only about the number of ARP requests and responses.	Relatively straightforward. Identify incorrect entries or device MAC address issues and rectify the mistakes.	Straightforward. It gives out the MAC address of the victim and attacker. This enables us to look deeper into the points of breach in our network.

Table 1: Comparison Between the Existing Methods

This paper presents a working algorithm that manages to capture and analyze the ARP packets on the network and locates the actual source of the invalid packet, and discloses the attacker's mac address efficiently and robustly.

5. Conclusion and Future Work

The algorithm outlined in this paper effectively identifies ARP spoofing attacks within the network. As depicted in Figures 10 and 11, the algorithm can detect invalid packets regardless of whether the IP and MAC addresses are listed in the DHCP database. However, the problem arises if the attacker sends fake DHCP data packets to manipulate our database itself.

Our algorithm actively inspects all the ARP packets on the network, and in case of any possibility of attack, it uses ICMP packets to identify the attacker. By using the DHCP database, we significantly decrease the time taken to validate each ARP packet.
Overall, our algorithm enhances network security by making sure that every ARP packet on the network is valid, providing us with details of the attacker, and is efficient and scalable.

References

[1] Jaideep singh, Sandeep Dhariwal, and Rajeev Kumar. A detailed survey of arp poisoning detection and mitigation techniques. *International Journal of Control Theory and Applications*, 9, 02 2017.

[2] Huixing Xi. Research and application of ARP protocol vulnerability attack and defense technology based on trusted network. In *Advances in Materials, Machinery, Electronics (AMME 2017)*, volume 1820 of *American Institute of Physics Conference Series*, page 090019, March 2017.

[3] Wen Zeng, Junsheng Zhang, Ying Li, and Peng Qu. The study on media access control protocol for wireless network in library. *International Journal of Distributed Sensor Networks*, 2015:1–10, 08 2015.

[4] Marco Carnut and Joao Gondim. Arp spoofing detection on switched ethernet networks: A feasibility study. 11 2003.

[5] Khalid M. Amin Ahmed M.Abdel Salam, Wail S. Elkilani. An automated approach for preventing arp spoofing attack using static arp entries. *(IJACSA) International Journal of Advanced Computer Science and Applications*, 2014.

[6] Cisco. Static mac address support on service instances.

[7] Shimpy Goyal Vaishnavi Rohatgi. A detailed survey for detection and mitigation techniques against arp spoofing. *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2020.

[8] A. Kaur R. Kaur. Digital signature, in: 2012 international conference on computing sciences, pp. 295–301. 2012.

****** END ******