Prevention techniques against ARP spoofing

**Jai Kumawat, Digvijay Singh**

**Students at**

**Bennett University**

**Greater Noida, UP, India**

**Abstract**— Securing the user’s data that pass through the internet is a pervasive challenge until today. One of the threats is ARP poisoning or ARP spoofing. Those attacks are meant to exploit the vulnerability of the ARP protocol. There usually isn’t a quick fix to help identify and combat ARP spoofing, but there are ways to protect yourself and stay proactive about your security. One of the most basic but effective solution to mitigate this type of attack is adding MAC and IP address to the static ARP cache table manually. While these static entries provide some security against spoofing, but the maintenance takes a lot of effort. This research proposed a method to tackle ARP spoofing through static entries. This proposed method is simple and compatible with standard ARP protocol. It does not require any modification of the standard ARP protocol nor replacing old devices. We successfully proved the effectiveness of our proposed method.

**Index Terms**—ARP, static entries, MiMT, Spoofing, cache

—————————— ◆ ——————————

# 1 Introduction

**Internet**, a system architecture that has revolutionized communications and methods of commerce by allowing various computer networks around the world to interconnect. By 2020, approximately 4.5 billion people, or more than half of the world’s population, were estimated to have

access to the Internet. Most of those users use the internet for communication and sharing information. Therefore, securing the user’s data that pass through the internet is a very important challenge until today. Internet use packet switching communication model. It breaks down the data into smaller chunks then sent it in the form of a discrete packet that follows different channels in a sequence over time and re-joins at the destination node. One of the threats is ARP poisoning or ARP spoofing. Those attacks are meant to exploit the vulnerability of the ARP protocol. This attack exploits the vulnerability of ARP protocol that translates logical address to the physical address of a device.

For decades, many researches proposed several approaches

to mitigate the ARP spoofing attacks. **There are four categories of this mitigation approaches:**

* **Modifying ARP using cryptographic techniques;**
* **Setup packet filtering**
* **ARP spoofing attack detection and protection using external software; and**
* **Manually configuring static ARP cache table.**

The first solution protects the ARP protocol by adding a cryptographic function. Protection against data espionage can be promised by some encryption techniques and certificates for authentication. If an attacker only catches encoded data, the worst case is limited to a denial of service by discarding data packets. But reliable data encryption must be implemented consistently. The major drawback of this approach is the incompatibility with the standard ARP protocol and affect the ARP protocol performance. The second solution is packet filtering and inspection can help catch poisoned packets before they reach their destination. It can filter and block malicious packets that show any conflicting source information. This will require an active firewall which scans all the packets. This will require more processing power and will call performance hindrance for general home user. The third solution use external software to detect and protect clients. Majority of these software are paid and are meant for industry use. The fourth solution is to use static ARP table which will allow us to communicate only with the devices in the table and eliminate poisoning. This feature is not present in affordable or legacy router, despite this static ARP table can be added to every device.

**Our proposed method combines the 3rd and 4rth solution. This will implement open source version for spoof detection and static table for device regardless of router support.**

This proposed method is simple and compatible with standard ARP protocol. It does not require any modification of the standard ARP protocol nor replacing old devices. We organize this paper as follows. In section II, we describe the basic ARP spoofing technique and attack. In section III, we describe the detail explanation and implementation of our proposed method. We provide our result of our research in section IV. In the last section, we wraps-up the conclusion of this research.

# 2 ARP SPOOFING ATTACK

Every computer or device on the internet has two types of addresses: its physical address and its internet address. The IP address is a logical address of a device. IP address used to identify the location of a device that connected to the internet. It will change dynamically every time the user connects to the internet at a different location. On the other hand, the MAC address is a physical address that stored inside the network interface card. The MAC address is unique and unchangeable. MAC address is necessary for the internet protocol. MAC address used to identify the location of a device in a Local Area Network (LAN). Both MAC addresses and IP addresses are meant to identify a network device, but in different ways. A MAC address is responsible for local identification and an IP address for global identification. The MAC address is only significant on the LAN to which a device is connected, and it is not used or retained in the data stream once packets leave that network.

When sending a frame of data in a local arena network, the

sender must know the MAC address of the receiver. The

sender uses the ARP protocol to translate the IP destination

inside the frame to a MAC address of the destination device.

ARP protocol comprised of two type of messages, they are

ARP request and ARP reply. ARP request specifies the IP

address of the target host MAC address. ARP reply specifies

the MAC address associated with that IP address. For example, When a source device want to communicate with another device, source device checks its Address Resolution Protocol (ARP) cache to find it already has a resolved [MAC Address](https://www.omnisecu.com/tcpip/media-access-control-mac-addresses.php) of the destination device. If it is there, it will use that [MAC Address](https://www.omnisecu.com/tcpip/media-access-control-mac-addresses.php) for communication. If ARP resolution is not there in local cache, the source machine will generate an Address Resolution Protocol (ARP) request message, it puts its own [data link layer address](https://www.omnisecu.com/tcpip/media-access-control-mac-addresses.php) as the Sender Hardware Address and its own [IPv4 Address](https://www.omnisecu.com/tcpip/internet-layer-ip-addresses.php) as the Sender Protocol Address. It fills the destination [IPv4 Address](https://www.omnisecu.com/tcpip/internet-layer-ip-addresses.php) as the Target Protocol Address. The Target Hardware Address will be left blank, since the machine is trying to find Target Hardware Address. The source [broadcasts](https://www.omnisecu.com/tcpip/what-is-limited-broadcast-in-ipv4.php) the Address Resolution Protocol (ARP) request message to the local network.

The message is received by each device on the [LAN](https://www.omnisecu.com/basic-networking/lan-and-wan-local-area-network-and-wide-area-network.php) since it is a [broadcast](https://www.omnisecu.com/cisco-certified-network-associate-ccna/unicast-multicast-broadcast.php). Each device compare the Target Protocol Address ([IPv4 Address](https://www.omnisecu.com/tcpip/internet-layer-ip-addresses.php) of the machine to which the source is trying to communicate) with its own Protocol Address ([IPv4 Address](https://www.omnisecu.com/tcpip/internet-layer-ip-addresses.php)). Those who do not match will drop the packet without any action.

When the targeted device checks the Target Protocol Address, it will find a match and will generate an Address Resolution Protocol (ARP) reply message. It takes the Sender [Hardware Address](https://www.omnisecu.com/tcpip/media-access-control-mac-addresses.php) and the Sender [Protocol Address](https://www.omnisecu.com/tcpip/internet-layer-ip-addresses.php) fields from the Address Resolution Protocol (ARP) request message and uses these values for the Targeted Hardware Address and Targeted Protocol Address of the reply message. The destination device will update its Address Resolution Protocol (ARP) cache, since it need to contact the sender machine soon. Destination device send the Address Resolution Protocol (ARP) reply message and it will NOT be a [broadcast](https://www.omnisecu.com/cisco-certified-network-associate-ccna/unicast-multicast-broadcast.php), but a [unicast](https://www.omnisecu.com/cisco-certified-network-associate-ccna/unicast-multicast-broadcast.php). The source machine will process the Address Resolution Protocol (ARP) reply from destination, it store the Sender Hardware Address as the [layer 2 address](https://www.omnisecu.com/tcpip/media-access-control-mac-addresses.php) of the destination. The source machine will update its Address Resolution Protocol (ARP) cache with the Sender Hardware Address and Sender Protocol Address it received from the Address Resolution Protocol (ARP) reply message.

**The basic principle behind ARP spoofing is to exploit the lack of authentication in the ARP protocol by sending**[**spoofed**](https://en.wikipedia.org/wiki/Spoofing_attack)**ARP messages onto the LAN**. Generally, the goal of the attack is to associate the attacker's host MAC address with the IP address of a target [host](https://en.wikipedia.org/wiki/Host_(network)), so that any traffic meant for the target host will be sent to the attacker's host. The attacker may choose to inspect the packets (spying), while forwarding the traffic to the actual default destination to avoid discovery, modify the data before forwarding it ([man-in-the-middle attack](https://en.wikipedia.org/wiki/Man-in-the-middle_attack)), or launch a [denial-of-service attack](https://en.wikipedia.org/wiki/Denial-of-service_attack) by causing some or all of the packets on the network to be dropped, or NetCut attack uses ARP spoofing to monopolizing the bandwidth by cutting the communication of all other devices.

ARP is a [stateless protocol](https://en.wikipedia.org/wiki/Stateless_protocol). Network hosts will automatically [cache](https://en.wikipedia.org/wiki/Cache_(computing)) any ARP replies they receive, regardless of whether network hosts requested them. Even ARP entries that have not yet expired will be overwritten when a new ARP reply packet is received. There is no method in the ARP protocol by which a host can [authenticate](https://en.wikipedia.org/wiki/Authenticate) the peer from which the packet originated. There are two basic spoofing techniques utilizing the vulnerability of the ARP protocol. The first technique is spoofing ARP request packet and the second technique is spoofing ARP reply packet.

## 2.1 ARP request spoofing

This spoofing technique can be best understood by example of man in the middle attack. In this the attacker

simultaneously tricks the victim and the gateway**. Attacker sends APR request packet to the victim**. This ARP request packet contains spoofed data to trick the victim. The victim believes that the sender of that ARP request packet was gateway, therefore the victim caches the information from the ARP request packet into its own ARP cache table similarly he tricks the gateway. This poisoning re-routes all traffic between the gateway and the victim to the attacker. The attacker then acts as a relay between them. The victim usually does not aware of this attack because there is no disruption in the network connection.

## 2.2 ARP reply spoofing

Spoofing using ARP reply packet has a similar effect with

spoofing using ARP request. The only difference is the type

of the ARP packet. **The attacker directly sends ARP reply to the victim** even though the victim never requests it. However, sometimes this type of attack is easily noticeable by the Intrusion Detection System (IDS) because it is very unusual for a host received an ARP reply without sending an ARP request.

# 3 EXPLANATION AND IMPLEMENTATION

Some operating systems allow their users to manually add static ARP entries in their ARP table**. Using this feature the code creates static ARP tables in accordance with users' needs.**

Static ARP entries prevent the attack , as it already has the confirmed address .

Many routers do not ship with built in **static ARP cache support, but this can be done manually with ease.**

We will be using Python 3.8 for this demonstration, but the idea can be applied to any language.

To approach this, we assume that the the time of creation of the table all devices are in safe state. Every device on the WAN will be pinged which will cause the **arp table to be updated with the MAC addresses for all the devices**.

def pingall(address):

counter=1

value=[i for i in address.split('.')]

string=""

for i in range(len(value)-1):

     string+=value[i]+"."

while(counter<=255):

     if(counter%15==0)

         getarp.gettable(getarp.map)

         os.system("ping "+string+str(counter)+" -c 1")

   counter+=1

These are stored in table and made static with the feature mentioned above .

def gettable(map):

with os.popen('arp -a') as f:

     data = f.read()

data=data.split()

for i in range(len(data)):

     if(data[i]=="at" and data[i+1]!="<incomplete>" ):

         map[data[i-1]]=data[i+1]

This process will reduce the task of manually adding static IP and corresponding MAC manually for every device which can be a lengthy process. These functions can also be reduced to just work with the existing arp table in case cache for all devices is already present.

**However the user might want to add some addresses which are not present at the time of execution, they can be manually added .**

def addvalues(map):

n=int(input("How many values do you want to add"))

for i in range(n):

     ip=input("Enter the value of ip")

     ip='('+ip+')'

     mac=input("Enter the corresponding MAC address")

     map[ip]=mac

**The second part of our implementation includes scanning for ARP Poisoning .**

For this we will consider two scenarios :

1. **ARP SPOOF WITHOUT MAC SPOOF (Weak attack)**
2. **ARP SPOOF WITH MAC SPOOF (Strong attack )**

The implementation for the first part is very straight forward , **we are checking for multiple entries for the same local ip address .**

def scan():

arr=set()

with os.popen('arp -a') as f:

     data = f.read()

data=data.split()

for i in range(len(data)):

     if(data[i]=="at" and data[i+1]!="<incomplete>" ):

         if data[i-1] in arr:

             return False

         else:

             arr.add(data[i-1])

return True

In the second case we will use the SCAPY library to detect MAC spoofing in case the first scan returns True .

**This will compare the “real” MAC address to the “BROADCASTED” MAC address .**

This can be run indefinitely to keep on comparing the real and spoofed mac address of every device .

def mypern(packet):

if packet.haslayer(ARP):

     if packet[ARP].op == 2:

         real = mac(packet[ARP].psrc)

            showvalue = packet[ARP].hwsrc

            if real != show:

                print(“Attack has occurred”)

Real Time implementation will take a lot more functions for other tasks, but the major functions given above will remain the same .

For full working code reference please refer  : <https://github.com/digishek/ARP-SPOOF-SCANNING-PREVENTING->

**The above implementation takes a preventive measure (static table) and a detection method which considers all the concerned scenarios.**

**4 RESULT**

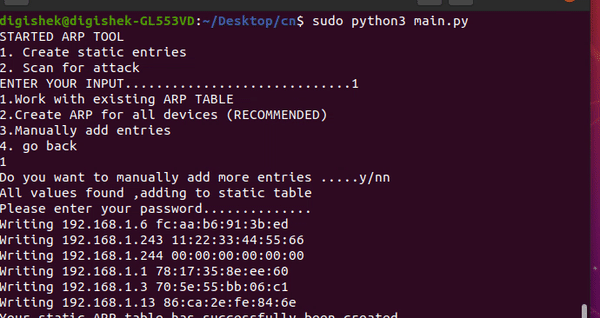
In the experiment, we tested all the fuctionalities of the module one by one. We started by testing the function to generate static tables, in this we generated static tables by broadcasting the request for mac addresses of all the devices in the LAN. The result can be seen in gif 1.

Then we tested the next functionality that is if we want to use pre-existing table and add some entries manually. The result can be seen in gif 2.

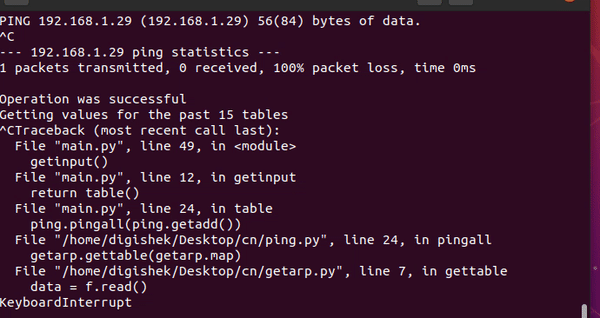
Thirdly, we scan for possible attacks if arp poisoning is occurring, it will be detected. This can be seen in gif 3.

And lastly if there is no attack found, then **we recommend the user to keep the program running in background so that any further attack can be scanned.** This can be seen in gif 4.

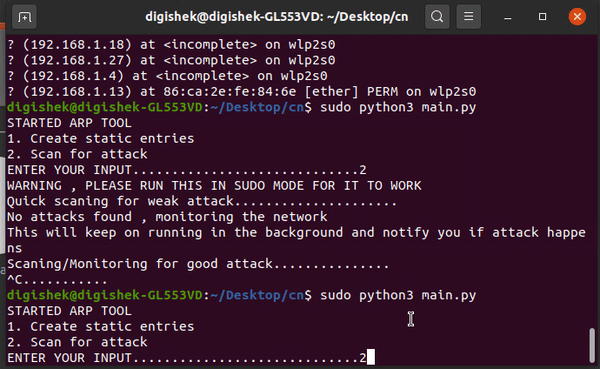
**\*\*open in preview mode for gif animations to work\*\***



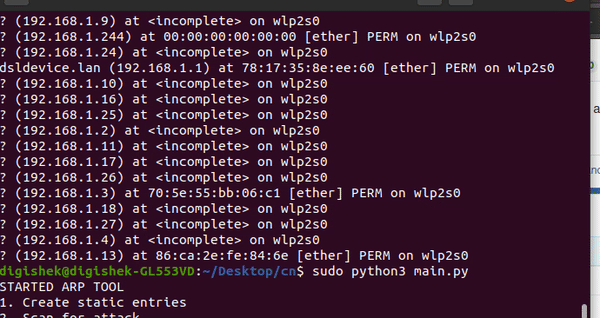
Gif 1 – generating static table



Gif 2- already existing static table



Gif 3- attack found



Gif 4- attack not found

# Conclusion

**We successfully proved the effectiveness of our proposed**

**method. Our proposed method can protect the hosts against all type of ARP spoofing attacks, including the MiTM attack. It does not need any modification of existing ARP protocol standard and devices**. It also easy to implement in every host. One minor weakness of this proposed method is it cannot prevent attack for hosts without our entries, but it can detect it. This ARP protocol will soon be legacy due to its limitation but till then there are good workarounds for it.

**Acknowledgment**

This research was supported **by Bennett University**, Greater Noida. We thank our Faculty of Computer Science who provided insight and expertise that greatly assisted the research.

**References**

[1] 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.

[2] J. Singh and V. Grewal, “A Survey of Different Strategies to Pacify

ARP Poisoning Attacks in Wireless Networks,” Int. J. Comput. Appl.,

vol. 116, no. 11, pp. 25–28, 2015.

## [3] Mahendra Data “ [The Defense Against ARP Spoofing Attack Using Semi-Static ARP Cache Table](https://ieeexplore.ieee.org/document/8693155/)”, in [2018 International Conference on Sustainable Information Engineering and Technology (SIET)](https://ieeexplore.ieee.org/xpl/conhome/8685076/proceeding)

## [4] Various blogs and educational sites on internet l