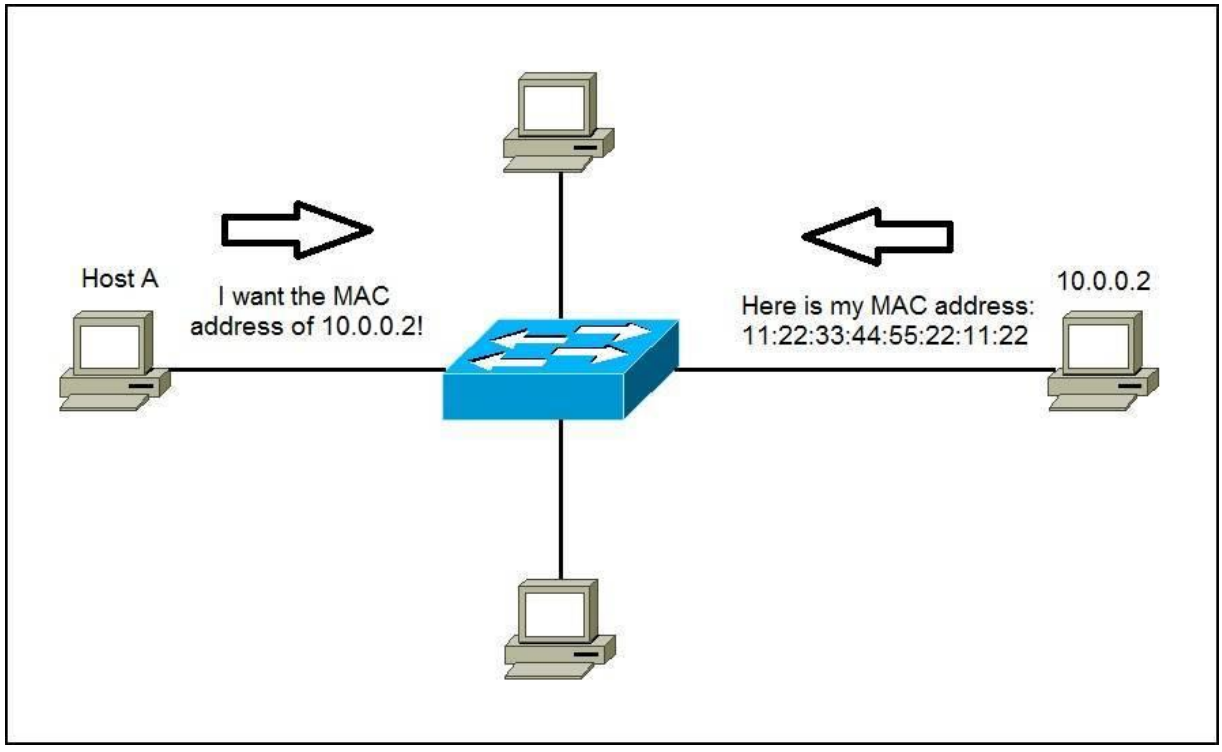
ARP SPOOF PROTECTION SYSTEM

INTRODUCTION

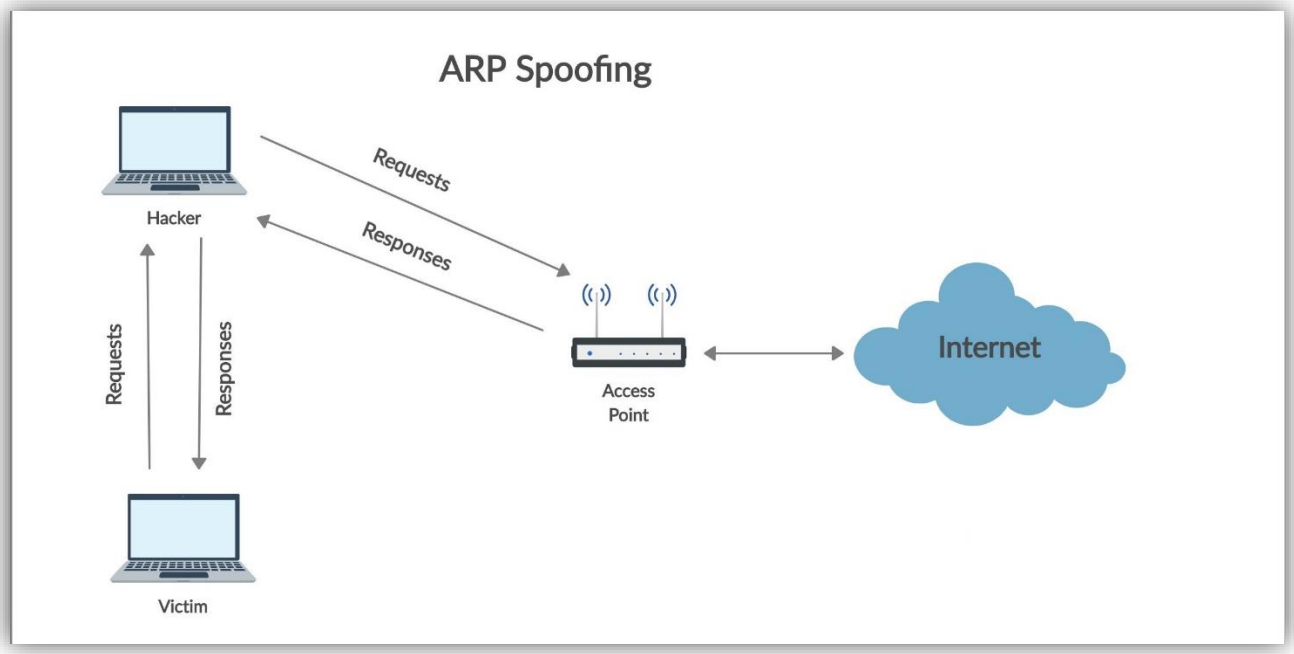
ARP

The **Address Resolution Protocol** (**ARP**) is a communication protocol used for discovering the link layer address, such as a MAC address, associated with a given internet layer address, typically an IPv4 address. This mapping is a critical function in the Internet protocol suite. ARP was defined in 1982 by RFC 826, which is Internet Standard STD 37.



ARP SPOOF

ARP spoofing is a type of attack in which a malicious actor sends falsified ARP (Address Resolution Protocol) messages over a local area network. This results in the linking of an attacker's MAC address with the IP address of a legitimate computer or server on the network. Once the attacker's MAC address is connected to an authentic IP address, the attacker will begin receiving any data that is intended for that IP address. ARP spoofing can enable malicious parties to intercept, modify or even stop data in-transit. ARP spoofing attacks can only occur on local area networks that utilize the Address Resolution Protocol. It is a type of man in the middle attack.



GOAL OF THE PROJECT

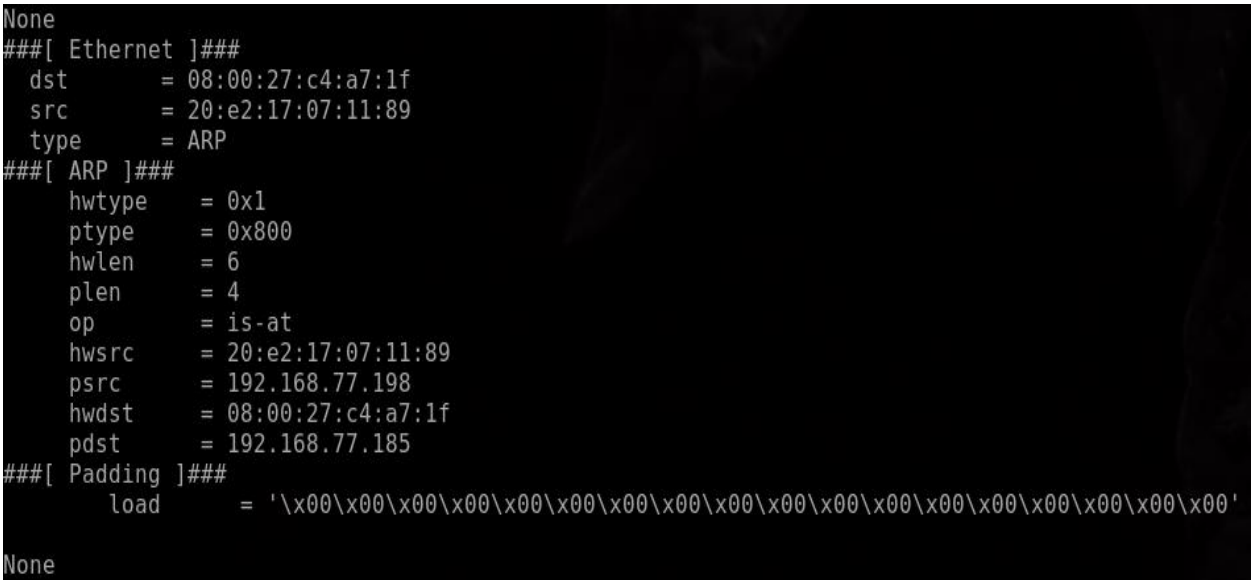
- *To detect ARP spoof attack in a system.*
- *To protect the system from the attack.*

TOOLS USED

- ***Operating System:***
  - Kali Linux
  - Windows
- ***Programming language:***
  - Python 3.9.1
- ***Text Editor:***
  - Sublime Text

WORKING OF THE SYSTEM

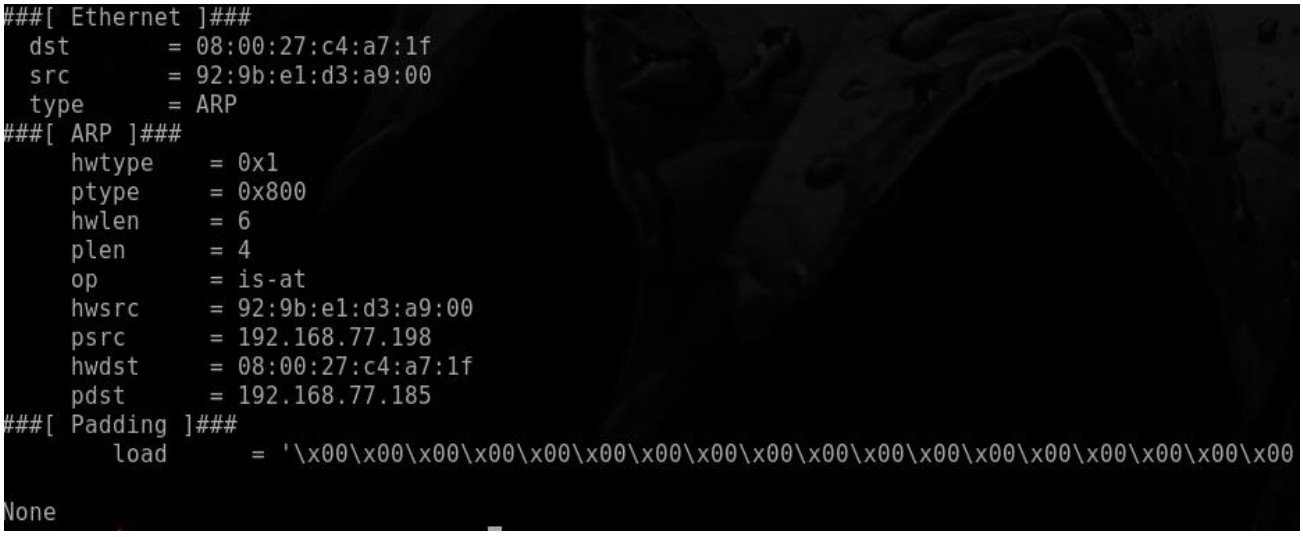
ARP Spoof protection system basically checks for some parameters in an ARP response. We are capturing the packets which is coming as only as an ARP response to us.



Here, we can see that we are getting an ARP response from IP (Internet Protocol) address of the router which is the “psrc” field with a MAC (Media Access Control) Address of the router which is the “hwsrc” field. Now, we will check for the real MAC address of the router by scanninng our Network with the IP address which we got from the “psrc” field.

```
def scanNetwork(ip):
    arp_request = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    broadcast_address = broadcast/arp_request
    answered = scapy.srp(broadcast_address,timeout=2, verbose=False)[0]
    return answered[0][1].hwsrc
```

Now, this function sends a broadcast MAC for which the IP address has been provided and returns the real MAC address of the IP.



So, here we can see that the MAC address from the packet that we got and the MAC address from scanning the network which is at “hwsrc” field **differs**. This means, our system has been spoofed as because the real MAC and the response MAC of the router are different.

```
def detectSpoof(packet):
    try:
        if(packet.haslayer(scapy.ARP) and packet['ARP'].op == 2):
            real_mac = scanNetwork(packet['ARP'].psrc)
            response_mac = packet['ARP'].hwsrc
            if(real_mac != response_mac):
                print("\n[+] Your System is under attack!!")
                abolishConnection()
                restoreConnection()
                catchPacket(interface)
                sys.exit()
    except IndexError:
        pass
```

So, to **protect** our system from spoofing, We are disconnecting our system's internet for 30 seconds so that no information can be transferred from victim to hacker computer. After 30 seconds, we are restoring the internet connection and If again the system is spoofed, the network will get disconnected again and this will continue in a loop until and unless ARP spoofing is stopped.

```
def abolishConnection():
    print("[-] Disconnecting your network.......please wait for 30 seconds")
    if(os_platform == "Linux"):
        sub.call("ifconfig "+interface+" down", shell=True)
    if(os_platform == "Windows"):
        sub.call("netsh interface set interface "+interface+" disable")
    time.sleep(30)


def restoreConnection():
    print("[+] Restoring connection.......please wait")
    if(os_platform == "Linux"):
        sub.call("ifconfig "+interface+" up", shell=True)
    if(os_platform == "Windows"):
        sub.call("netsh interface set interface "+interface+" enable")
    time.sleep(2)
    print("[+] Connection Restored")
```

Successful protection from ARP spoof attack

```
----------------------------------------
        Interfaces Available
----------------------------------------

DEVICE  TYPE      STATE      CONNECTION
eth0    ethernet  connected  Wired connection 1
lo      loopback  unmanaged  --

Enter the interface name from the following(state=connected): eth0

[+] Your System is under attack!!
[-] Disconnecting your network.......please wait for 30 seconds
[+] Restoring connection.......please wait
[+] Connection Restored

[+] Your System is under attack!!
[-] Disconnecting your network.......please wait for 30 seconds
[+] Restoring connection.......please wait
[+] Connection Restored
```

```
-----------Interfaces Available----------


Admin State    State          Type             Interface Name
-------------------------------------------------------------------------
Enabled        Connected      Dedicated        Wi-Fi
Enabled        Disconnected   Dedicated        Ethernet


Enter the interface name from the following(state=connected): Wi-Fi

[+] Your System is under attack!!
[-] Disconnecting your network.......please wait for 30 seconds

[+] Restoring connection.......please wait

[+] Connection Restored

[+] Your System is under attack!!
[-] Disconnecting your network.......please wait for 30 seconds

[+] Restoring connection.......please wait

[+] Connection Restored
```

SOURCE CODE

```
import scapy.all as scapy
import subprocess as sub
import time
import sys
import platform


def scanNetwork(ip):
    arp_request = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    broadcast_address = broadcast/arp_request
    answered = scapy.srp(broadcast_address,timeout=2, verbose=False)[0]
    return answered[0][1].hwsrc

def abolishConnection():
    print("[-] Disconnecting your network.......please wait for 30 seconds")
    if(os_platform == "Linux"):
        sub.call("ifconfig "+interface+" down", shell=True)
    if(os_platform == "Windows"):
        sub.call("netsh interface set interface "+interface+" disable")
    time.sleep(30)


def restoreConnection():
    print("[+] Restoring connection.......please wait")
    if(os_platform == "Linux"):
        sub.call("ifconfig "+interface+" up", shell=True)
    if(os_platform == "Windows"):
        sub.call("netsh interface set interface "+interface+" enable")
    time.sleep(2)
    print("[+] Connection Restored")
```

```
def catchPacket(interface):
    scapy.sniff(iface=interface, store=False, prn=detectSpoof)

def detectSpoof(packet):
    try:
        if(packet.haslayer(scapy.ARP) and packet['ARP'].op == 2):
            real_mac = scanNetwork(packet['ARP'].psrc)
            response_mac = packet['ARP'].hwsrc
            if(real_mac != response_mac):
                print("\n[+] Your System is under attack!!")
                abolishConnection()
                restoreConnection()
                catchPacket(interface)
                sys.exit()
    except IndexError:
        pass


os_platform = platform.system()

print("-------------------------------------\n\tInterfaces Available\n-------------------------------------\n")

if(os_platform == "Linux"):
    sub.call("nmcli device status",shell=True)
elif(os_platform == "Windows"):
    sub.call("netsh interface show interface",shell=True)


try:
    interface = input("\nEnter the interface name from the following(state=connected): ")
    catchPacket(interface)
except ValueError:
    sys.exit()
```

ADVANTAGES:

- Hackers won't be able to steal or access victim's information due to continuous disconnection of network.
- User doesn't need to worry about anything after starting the system as it keeps running on background until its forcefully stopped.
- This system works on both Windows and Linux providing securities to many devices.

LIMITATIONS:

- This system doesn't support protection in mac-OS
- If the user is connected in Windows with a Wi-Fi connection, then he needs to manually connect to the network after the connection is restored.
- If the hacker is patient enough to keep the attack active, then user cannot connect to the Internet until and unless its stopped.

CONCLUSION

In the world of networking and IT, nothing is perfectly secure. Every system has vulnerabilities from which the hackers try to gain access to the system and try to exploit. So, we have to secure our system as much as we can through these type of protections so that the necessary and confidential informations are never get leaked.