VPN Tunneling Lab-Part-2

1) Task-7: Routing experiment on Host V

1. First we need to open host V (4th tab)- docksh 192.168.60.5

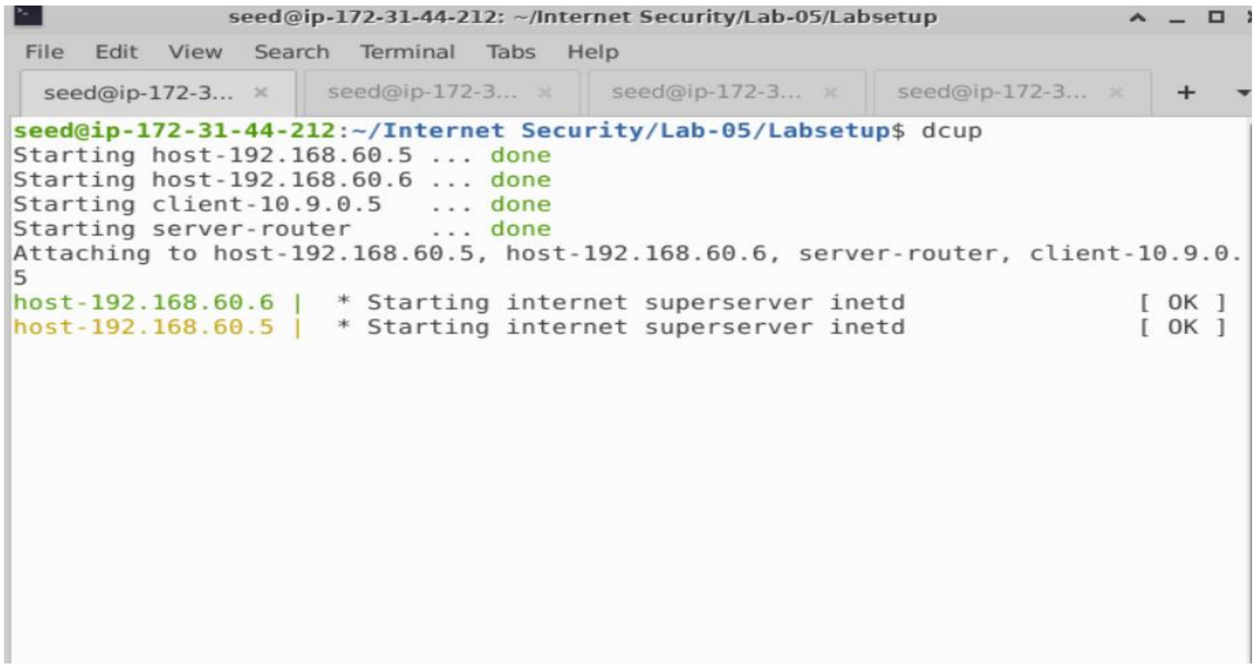
2. We need to check the iptable—ip route

3. We are asked to delete the default and add the new one- ip route del default.

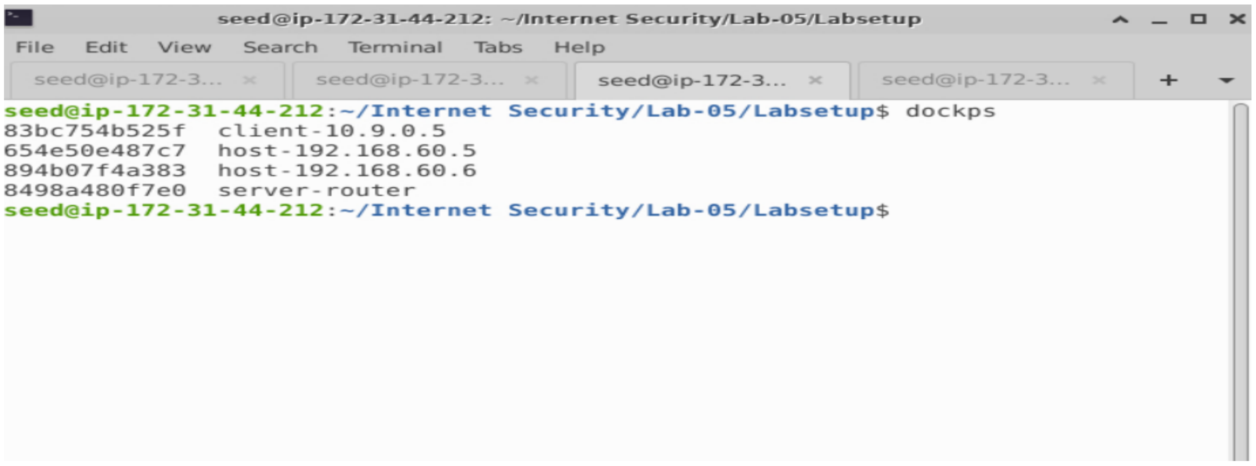
4. Again we need do the ip route

5.And we add an entry any packets going to the client side, return packets can be reloaded back to the vpn server, so we can add the client side.

6.The packets will be routed through this gateway—ip route add 192.168.53.0/24 via 192.168.60.11

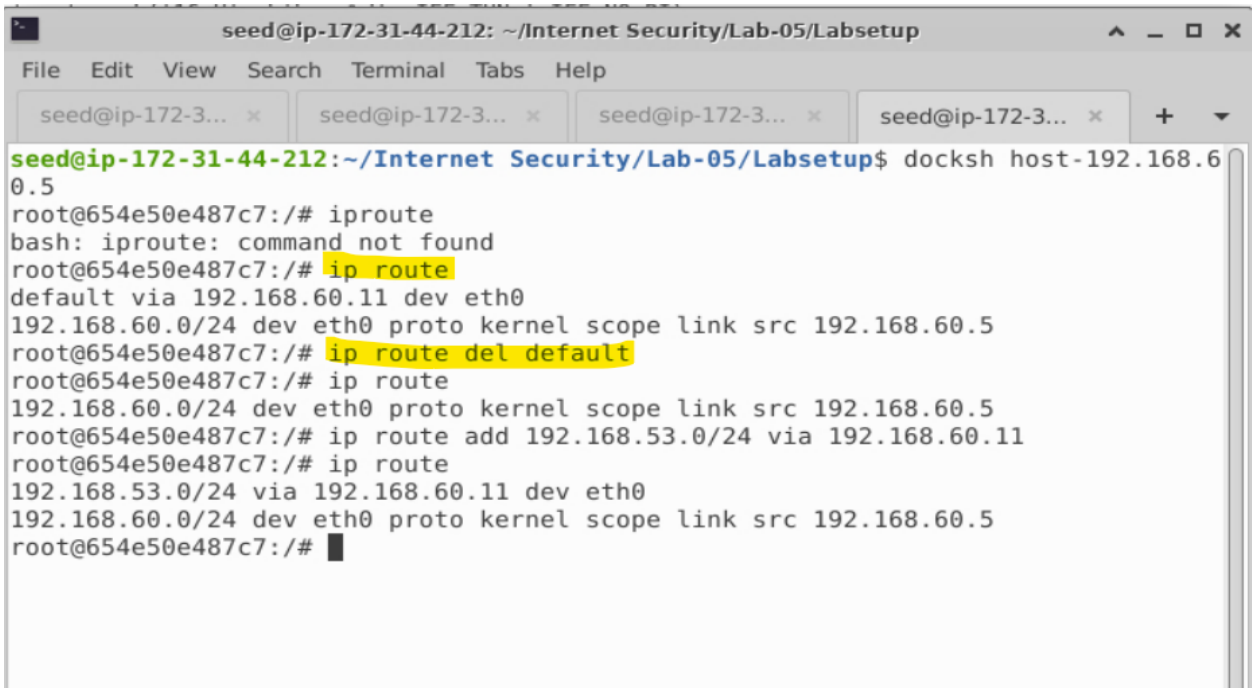

seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup
File Edit View Search Terminal Tabs Help
seed@ip-172-3...
seed@ip-172-3...
seed@ip-172-3...
seed@ip-172-3...

```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ dcup
Starting host-192.168.60.5 ... done
Starting host-192.168.60.6 ... done
Starting client-10.9.0.5   ... done
Starting server-router     ... done
Attaching to host-192.168.60.5, host-192.168.60.6, server-router, client-10.9.0.
5
host-192.168.60.6 |  * Starting internet superserver inetd              [ OK ]
host-192.168.60.5 |  * Starting internet superserver inetd              [ OK ]
```



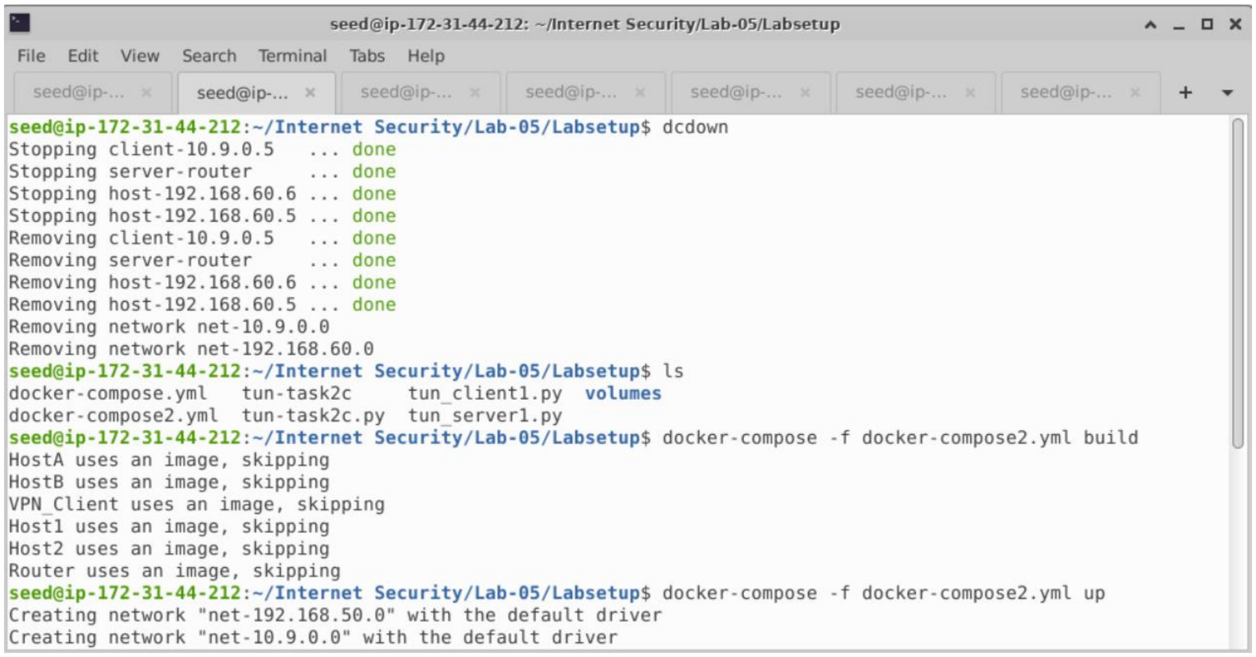
```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ dockps
83bc754b525f  client-10.9.0.5
654e50e487c7  host-192.168.60.5
894b07f4a383  host-192.168.60.6
8498a480f7e0  server-router
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$
```

To remove the default entry and add the new entry
//Delete the default entry
#ip route del default
//Add an entry
# ip rute add <network prefix> via <router ip>

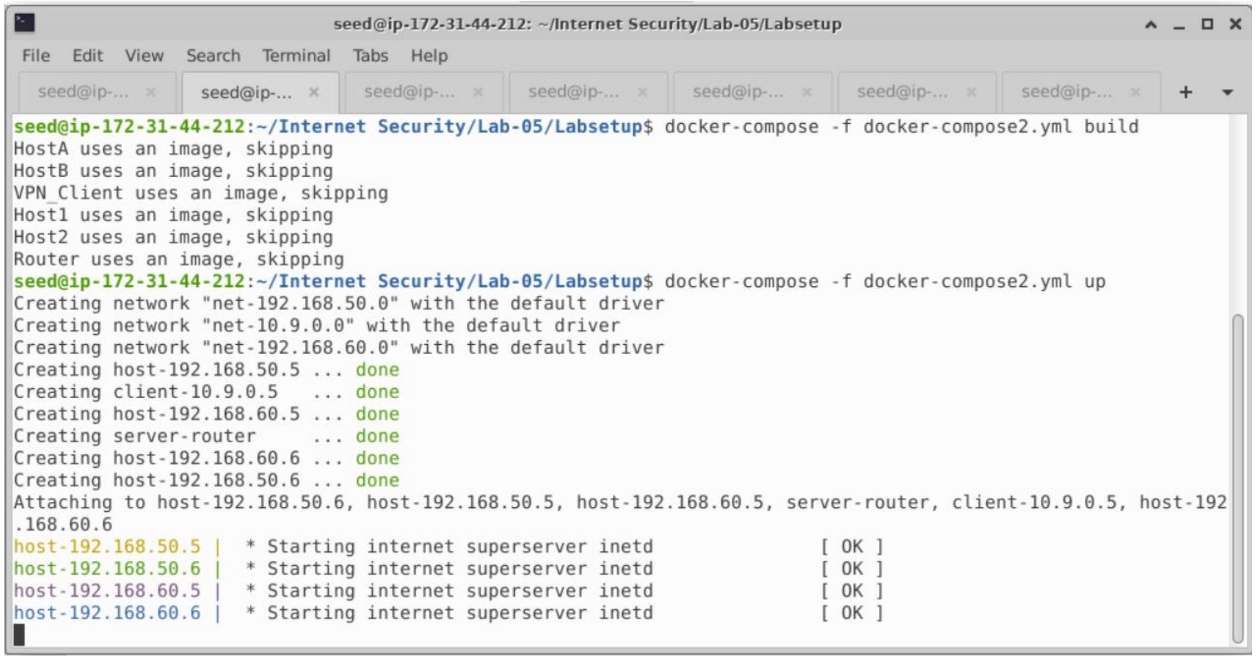


```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh host-192.168.6
0.5
root@654e50e487c7:/# iproute
bash: iproute: command not found
root@654e50e487c7:/# ip route
default via 192.168.60.11 dev eth0
192.168.60.0/24 dev eth0 proto kernel scope link src 192.168.60.5
root@654e50e487c7:/# ip route del default
root@654e50e487c7:/# ip route
192.168.60.0/24 dev eth0 proto kernel scope link src 192.168.60.5
root@654e50e487c7:/# ip route add 192.168.53.0/24 via 192.168.60.11
root@654e50e487c7:/# ip route
192.168.53.0/24 via 192.168.60.11 dev eth0
192.168.60.0/24 dev eth0 proto kernel scope link src 192.168.60.5
root@654e50e487c7:/# 
```

2) Task-8: VPN Between Private Networks

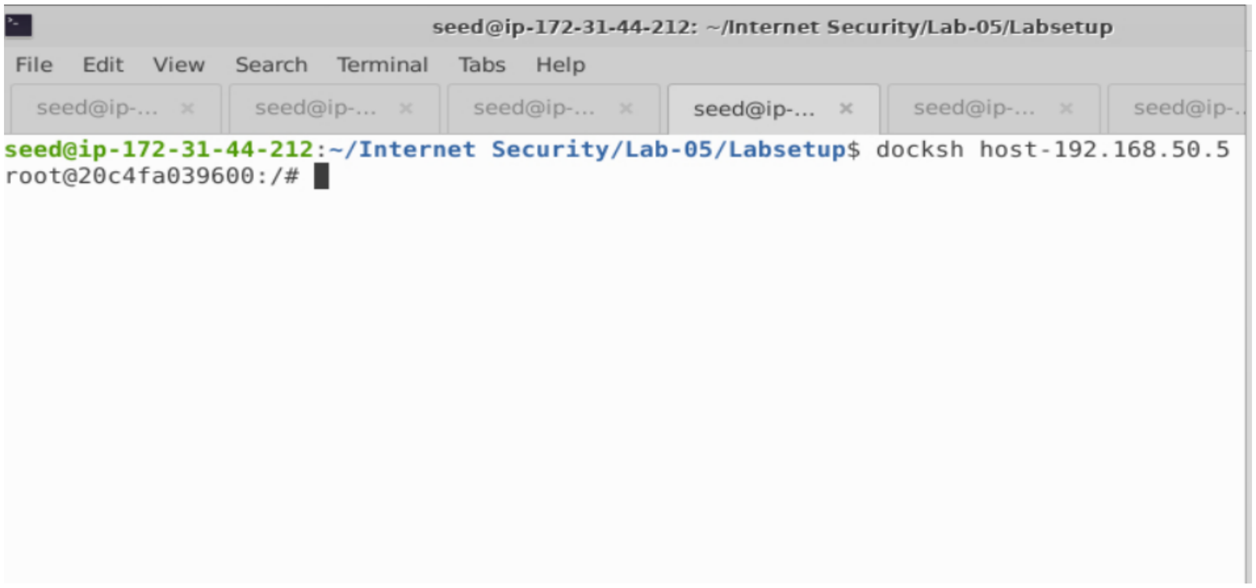

seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup
File Edit View Search Terminal Tabs Help

```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ dcdown
Stopping client-10.9.0.5   ... done
Stopping server-router     ... done
Stopping host-192.168.60.6 ... done
Stopping host-192.168.60.5 ... done
Removing client-10.9.0.5   ... done
Removing server-router     ... done
Removing host-192.168.60.6 ... done
Removing host-192.168.60.5 ... done
Removing network net-10.9.0.0
Removing network net-192.168.60.0
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ ls
docker-compose.yml   tun-task2c     tun_client1.py  volumes
docker-compose2.yml  tun-task2c.py  tun_server1.py
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docker-compose -f docker-compose2.yml build
HostA uses an image, skipping
HostB uses an image, skipping
VPN_Client uses an image, skipping
Host1 uses an image, skipping
Host2 uses an image, skipping
Router uses an image, skipping
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docker-compose -f docker-compose2.yml up
Creating network "net-192.168.50.0" with the default driver
Creating network "net-10.9.0.0" with the default driver
```

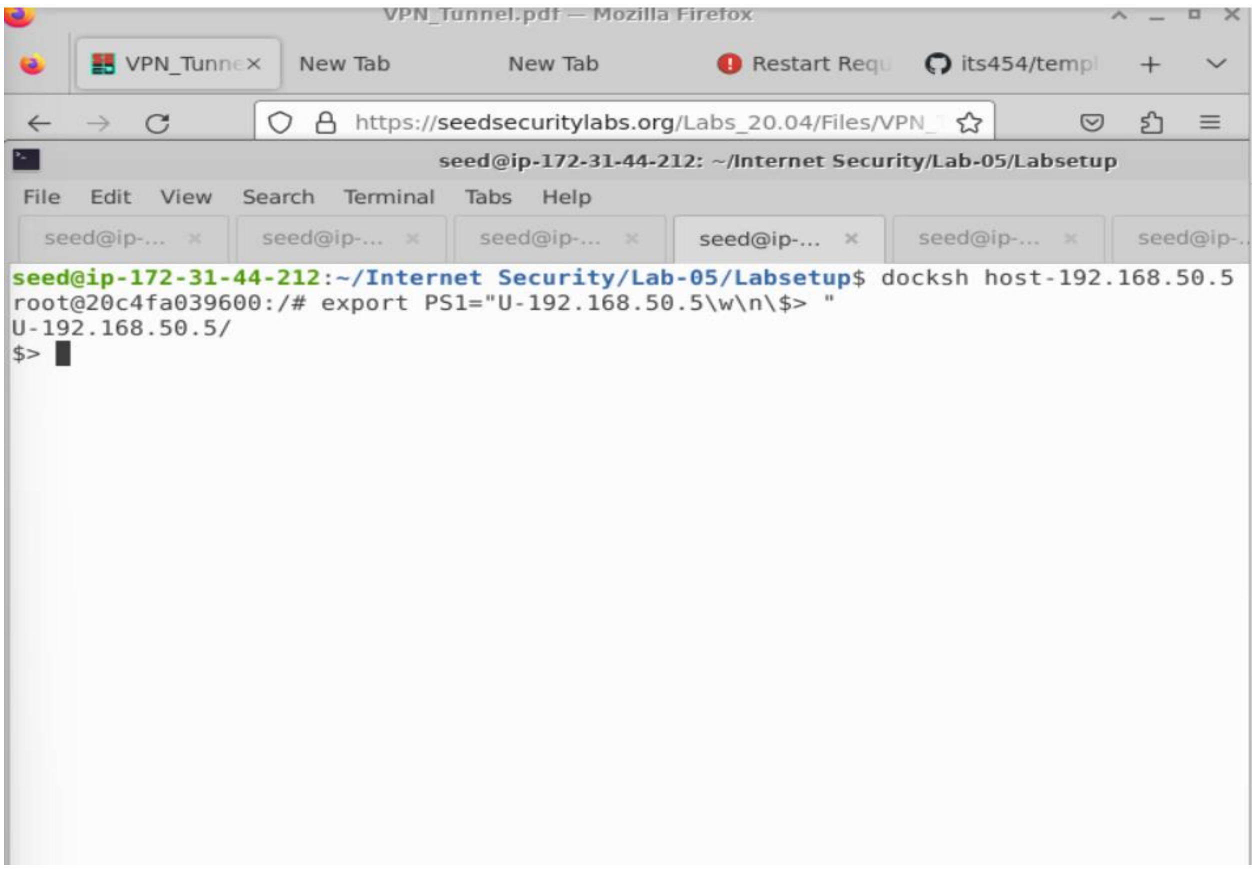

seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup
File Edit View Search Terminal Tabs Help

```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docker-compose -f docker-compose2.yml build
HostA uses an image, skipping
HostB uses an image, skipping
VPN_Client uses an image, skipping
Host1 uses an image, skipping
Host2 uses an image, skipping
Router uses an image, skipping
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docker-compose -f docker-compose2.yml up
Creating network "net-192.168.50.0" with the default driver
Creating network "net-10.9.0.0" with the default driver
Creating network "net-192.168.60.0" with the default driver
Creating host-192.168.50.5 ... done
Creating client-10.9.0.5   ... done
Creating host-192.168.60.5 ... done
Creating server-router     ... done
Creating host-192.168.60.6 ... done
Creating host-192.168.50.6 ... done
Attaching to host-192.168.50.6, host-192.168.50.5, host-192.168.60.5, server-router, client-10.9.0.5, host-192
.168.60.6
host-192.168.50.5 |  * Starting internet superserver inetd               [ OK ]
host-192.168.50.6 |  * Starting internet superserver inetd               [ OK ]
host-192.168.60.5 |  * Starting internet superserver inetd               [ OK ]
host-192.168.60.6 |  * Starting internet superserver inetd               [ OK ]
```

We want to put host U

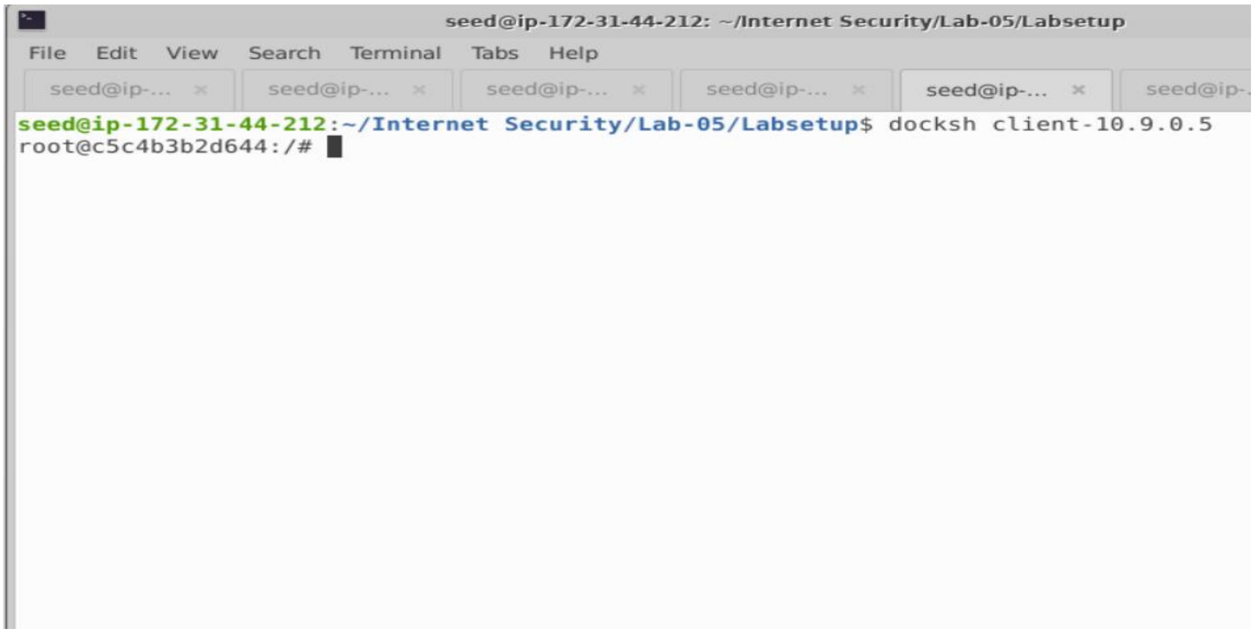

seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup
File Edit View Search Terminal Tabs Help
seed@ip-...

```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh host-192.168.50.5
root@20c4fa039600:/# 
```

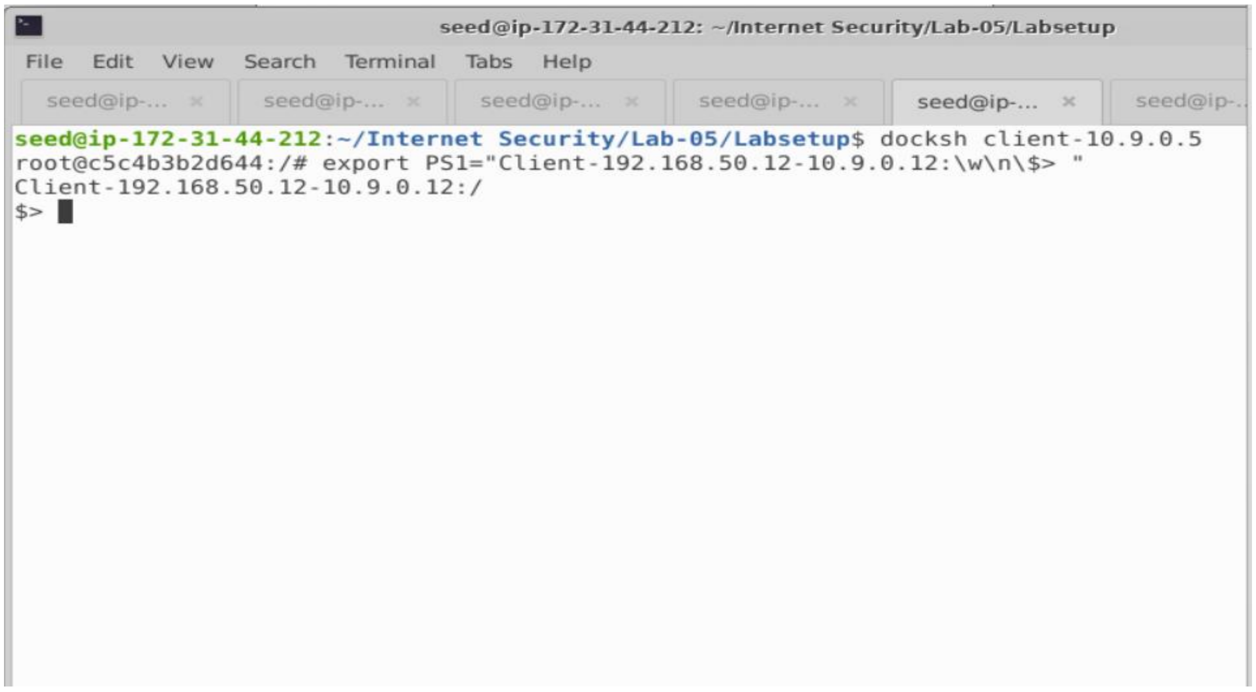

VPN_Tunnel.pdf — Mozilla Firefox
VPN_Tunne
New Tab
New Tab
Restart Requ
its454/templ
https://seedsecuritylabs.org/Labs_20.04/Files/VPN_
seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup
File Edit View Search Terminal Tabs Help
seed@ip-...

```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh host-192.168.50.5
root@20c4fa039600:/# export PS1="U-192.168.50.5\w\n\$> "
U-192.168.50.5/
$> 
```

Now client side-Two interfaces

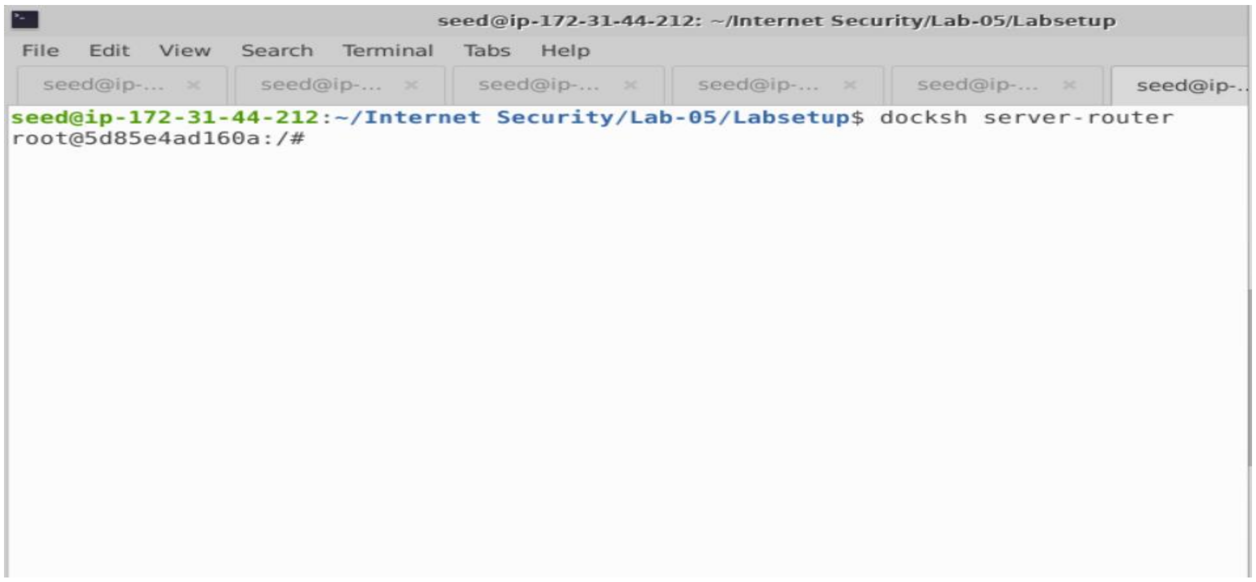


```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh client-10.9.0.5
root@c5c4b3b2d644:/#
```

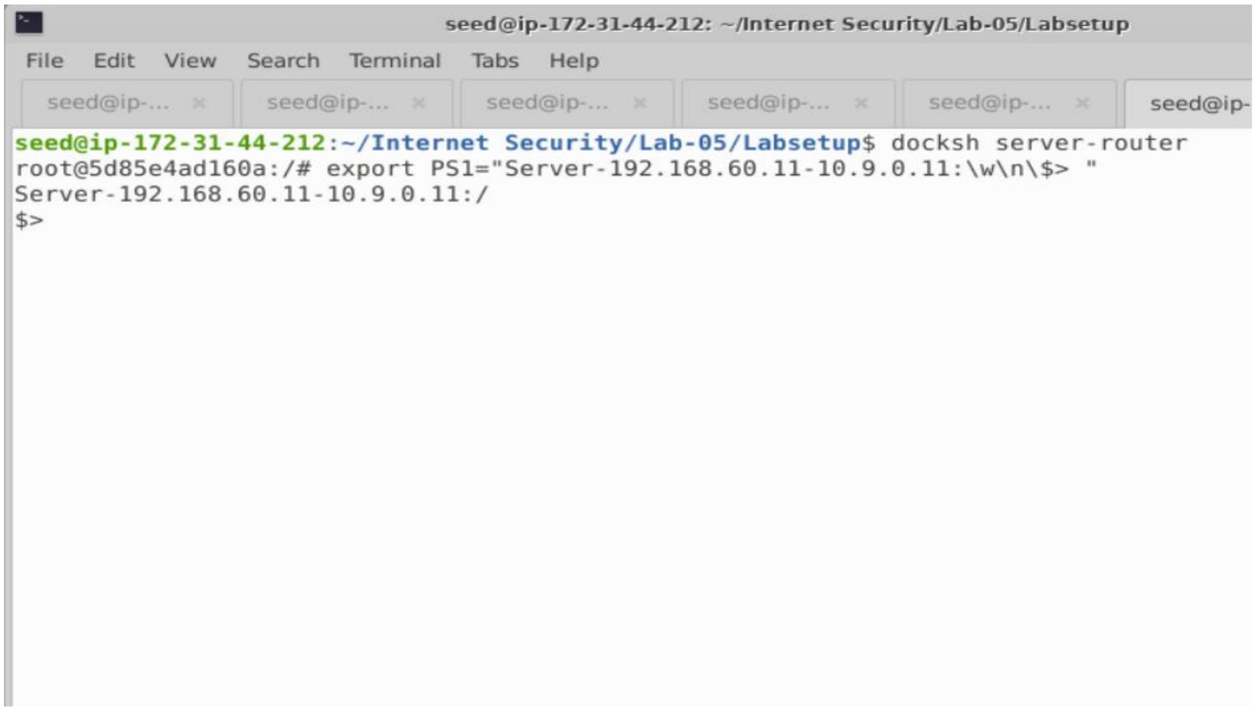


```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh client-10.9.0.5
root@c5c4b3b2d644:/# export PS1="Client-192.168.50.12-10.9.0.12:\w\n\$> "
Client-192.168.50.12-10.9.0.12:/
$>
```

Then the sever will also have two interfaces

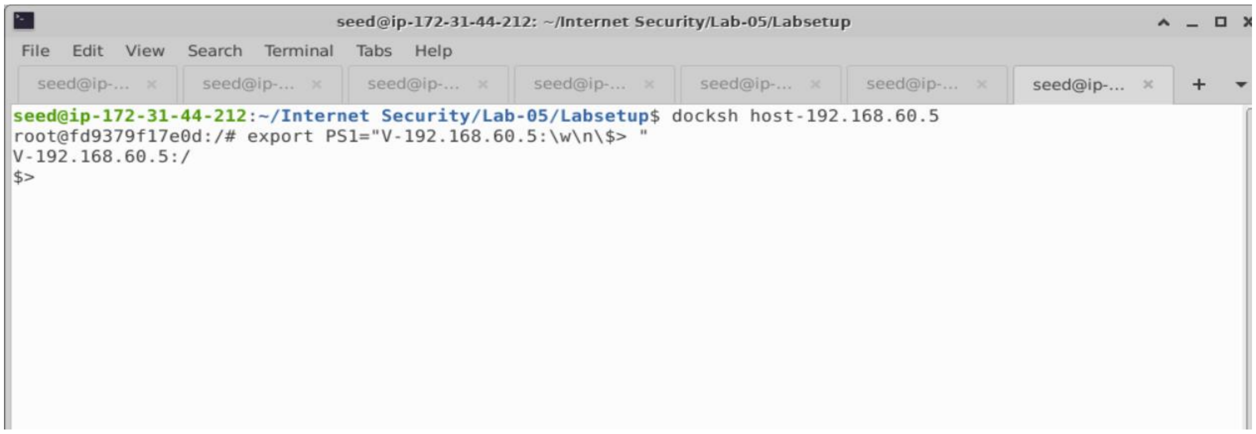

seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup
File Edit View Search Terminal Tabs Help

```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh server-router
root@5d85e4ad160a:/#
```


seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup
File Edit View Search Terminal Tabs Help

```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh server-router
root@5d85e4ad160a:/# export PS1="Server-192.168.60.11-10.9.0.11:\w\n\$> "
Server-192.168.60.11-10.9.0.11:/
$>
```

Host V



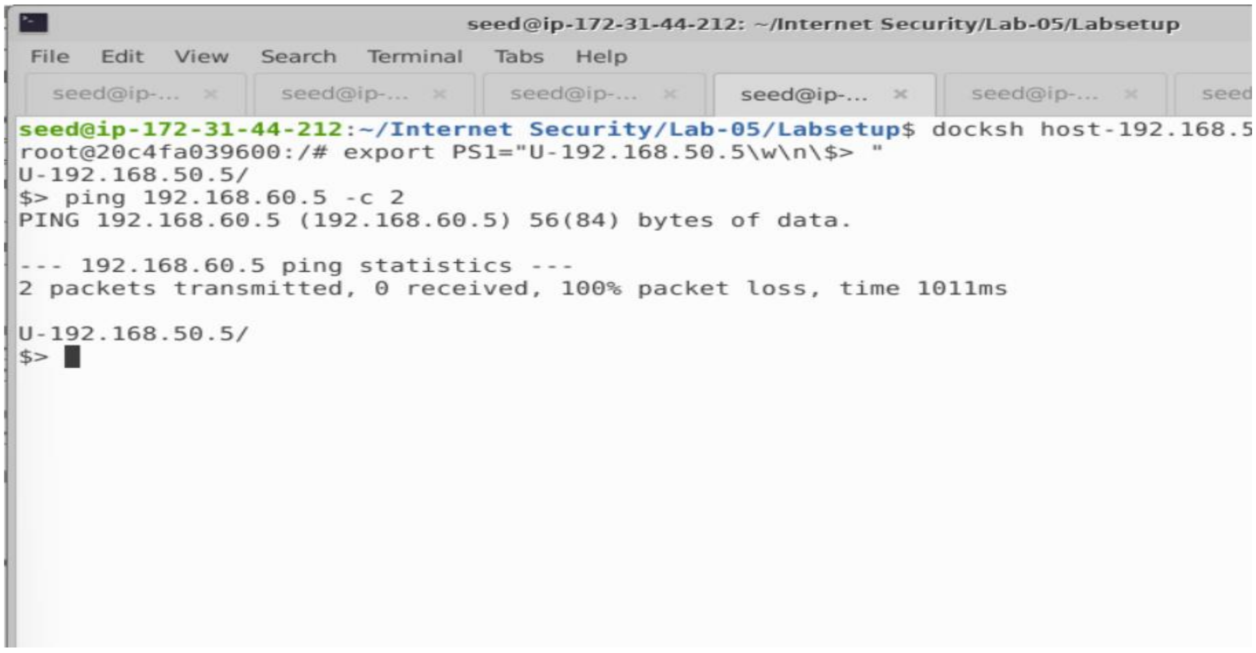
seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup

```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh host-192.168.60.5
root@fd9379f17e0d:/# export PS1="V-192.168.60.5:\w\n\$> "
V-192.168.60.5:/
$>
```

Now we need to set up the correct routing so the packet between these two private networks can get routed into the VPN tunnel.

Description- So we can set up a tunnel from host V to host U we should see a reply coming back, if everything is set up correctly, then we break the tunnel we will see, there's no ping reply or we can use telnet from host U to host V.

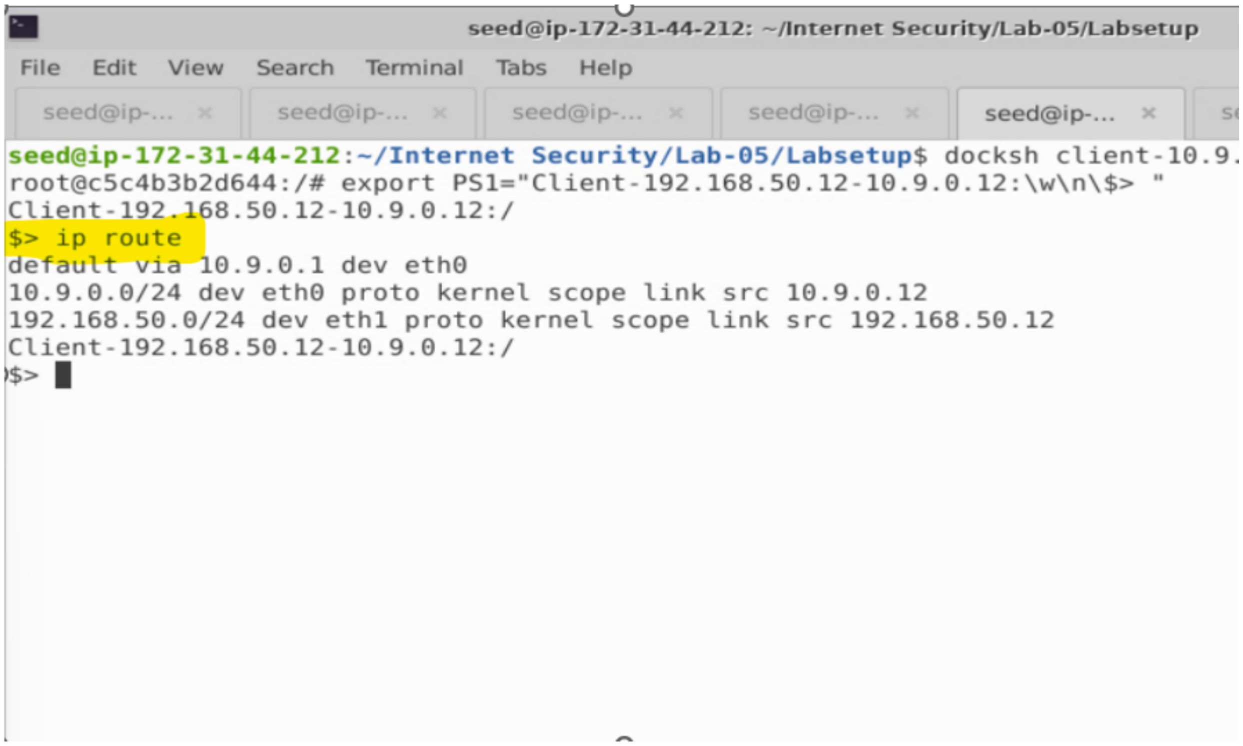
We need to ping from host U to host V.



seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup

```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh host-192.168.5
root@20c4fa039600:/# export PS1="U-192.168.50.5\w\n\$> "
U-192.168.50.5/
$> ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1011ms

U-192.168.50.5/
$>
```

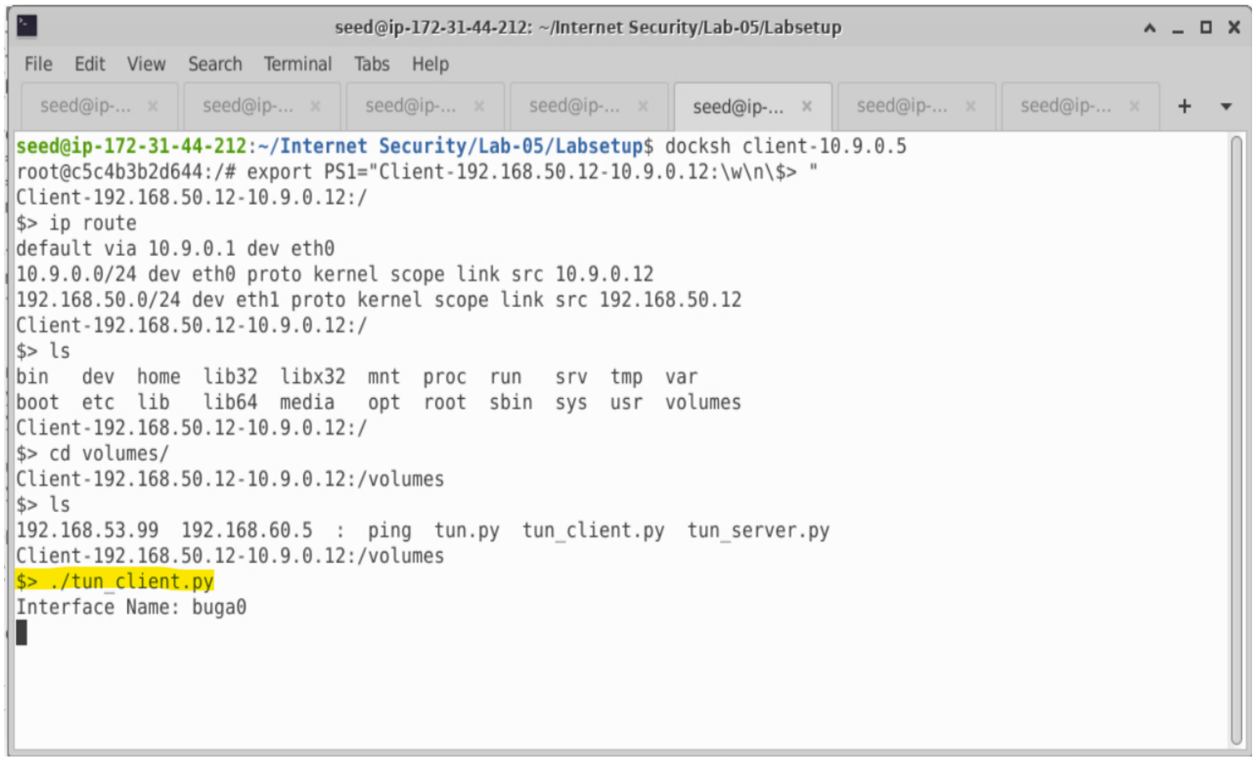
We can check the default routing of server and host



seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup

File Edit View Search Terminal Tabs Help

```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh client-10.9.
root@c5c4b3b2d644:/# export PS1="Client-192.168.50.12-10.9.0.12:\w\n\$> "
Client-192.168.50.12-10.9.0.12:/
$> ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.12
192.168.50.0/24 dev eth1 proto kernel scope link src 192.168.50.12
Client-192.168.50.12-10.9.0.12:/
$>
```

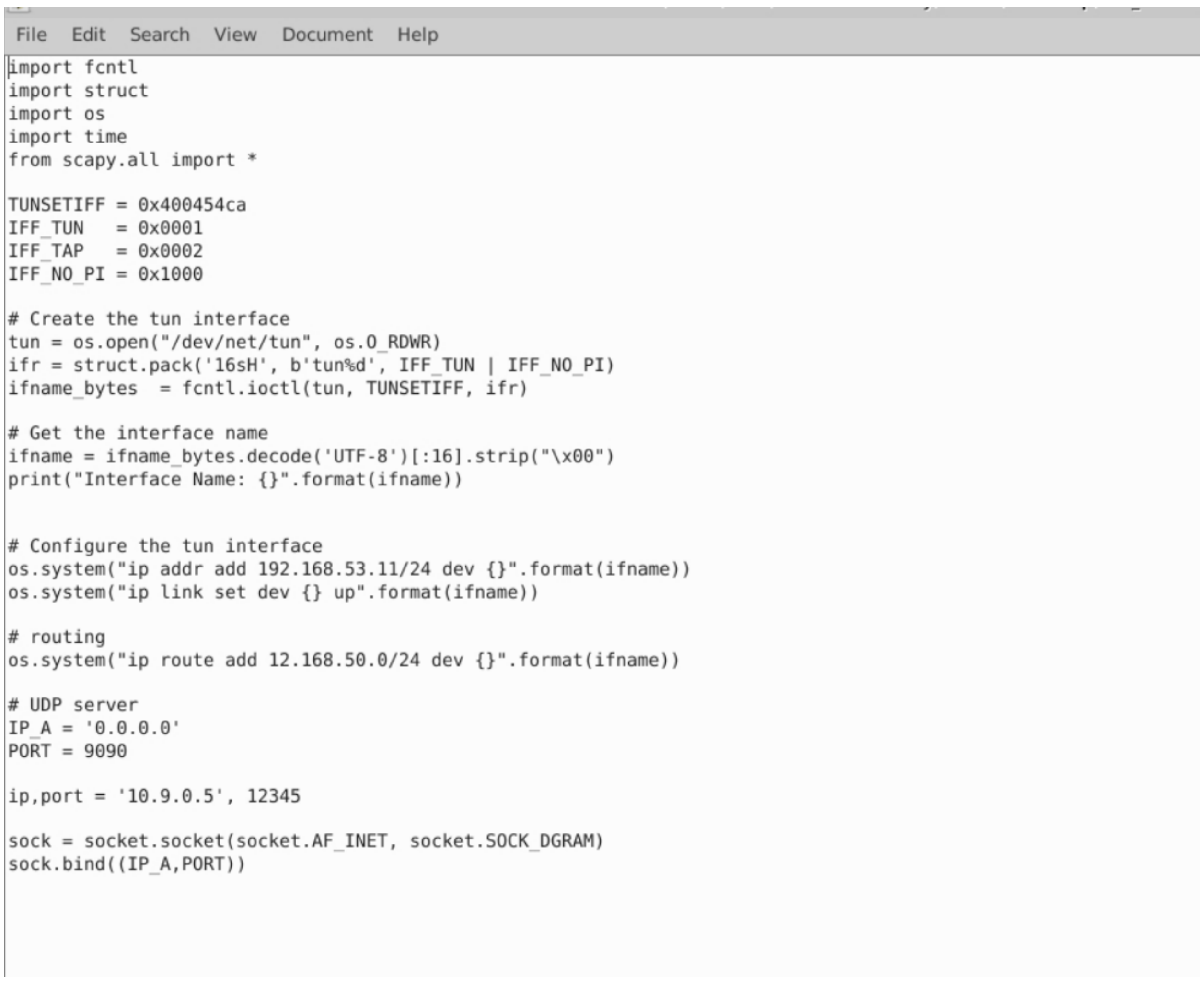


seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup

File Edit View Search Terminal Tabs Help

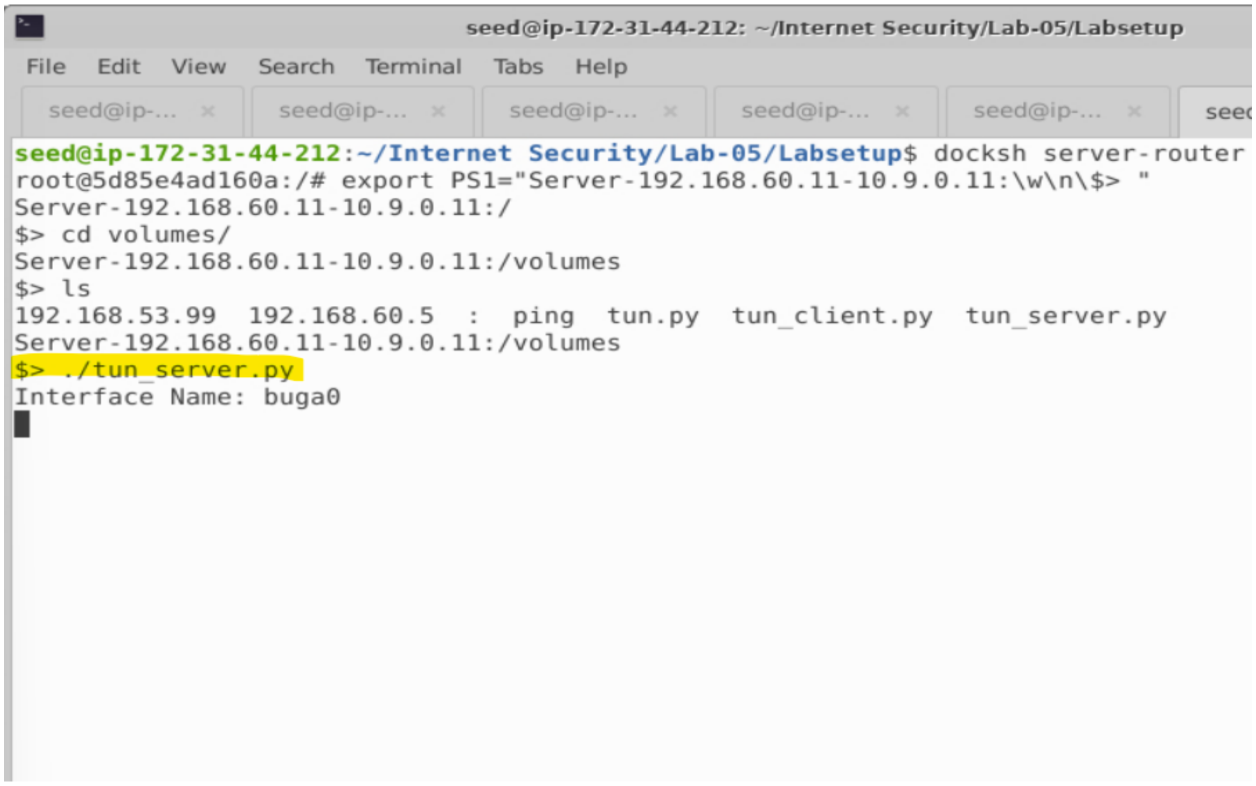
```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh client-10.9.0.5
root@c5c4b3b2d644:/# export PS1="Client-192.168.50.12-10.9.0.12:\w\n\$> "
Client-192.168.50.12-10.9.0.12:/
$> ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.12
192.168.50.0/24 dev eth1 proto kernel scope link src 192.168.50.12
Client-192.168.50.12-10.9.0.12:/
$> ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
Client-192.168.50.12-10.9.0.12:/
$> cd volumes/
Client-192.168.50.12-10.9.0.12:/volumes
$> ls
192.168.53.99  192.168.60.5  :  ping  tun.py  tun_client.py  tun_server.py
Client-192.168.50.12-10.9.0.12:/volumes
$> ./tun_client.py
Interface Name: buga0
```

Source code:-tun_server.py



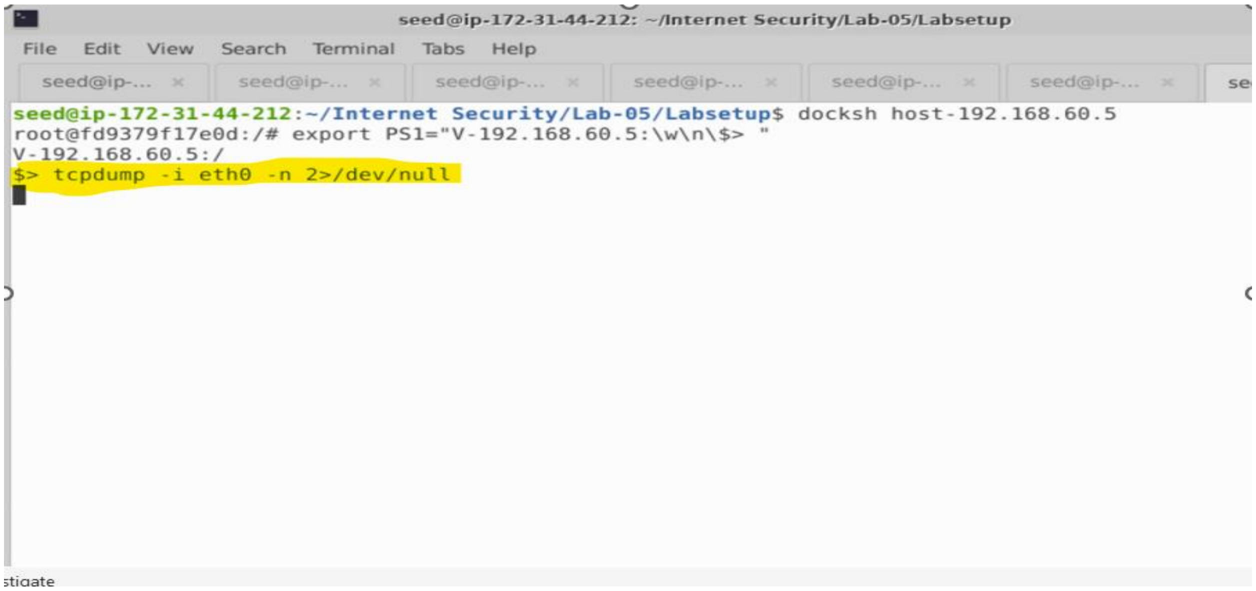
```
import fcntl
import struct
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))


# Configure the tun interface
os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# routing
os.system("ip route add 12.168.50.0/24 dev {}".format(ifname))

# UDP server
IP_A = '0.0.0.0'
PORT = 9090

ip,port = '10.9.0.5', 12345

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A,PORT))
```

On the server side we don’t need to run on the background

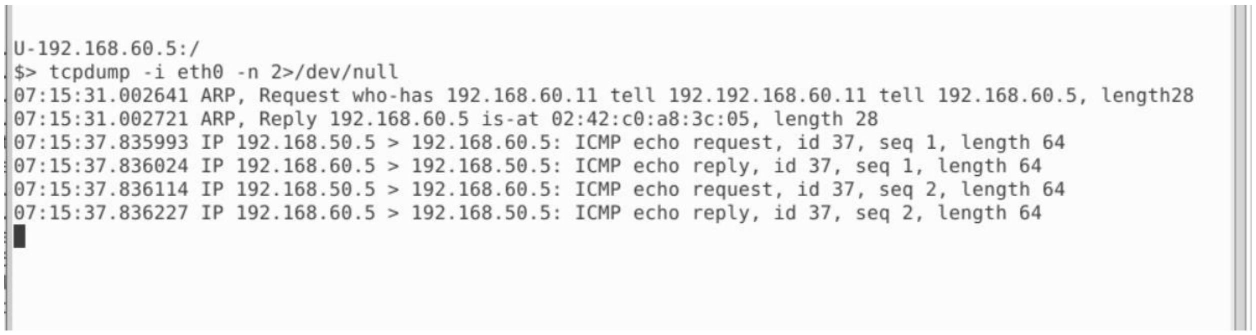

seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup
File Edit View Search Terminal Tabs Help

```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh server-router
root@5d85e4ad160a:/# export PS1="Server-192.168.60.11-10.9.0.11:\w\n\$> "
Server-192.168.60.11-10.9.0.11:/
$> cd volumes/
Server-192.168.60.11-10.9.0.11:/volumes
$> ls
192.168.53.99  192.168.60.5  :  ping  tun.py  tun_client.py  tun_server.py
Server-192.168.60.11-10.9.0.11:/volumes
$> ./tun_server.py
Interface Name: buga0
```

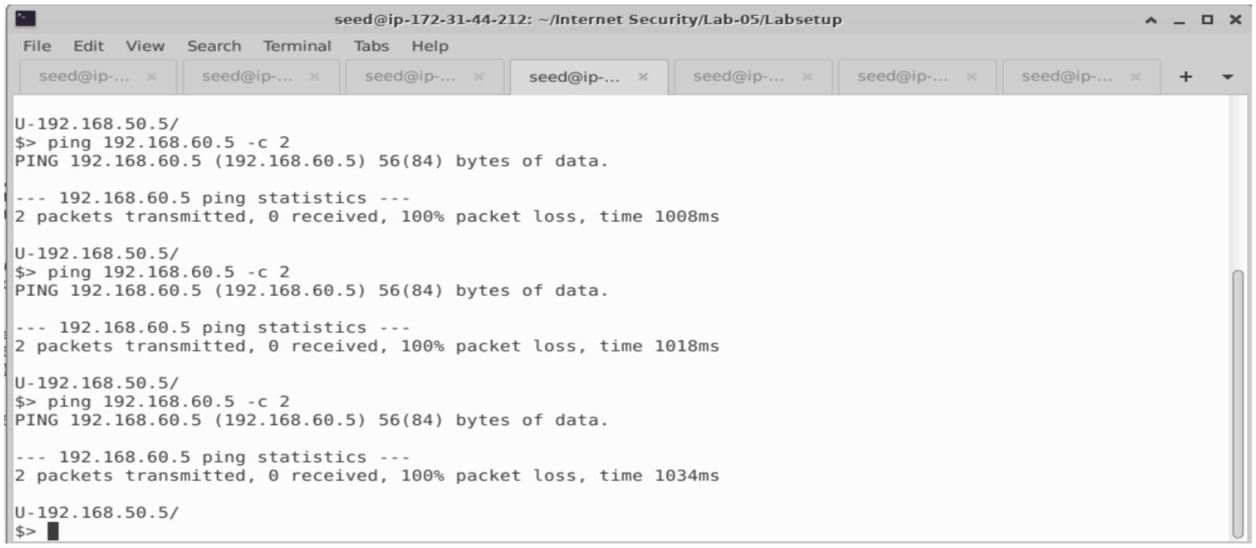
We can also add a host V to run tcpdump


seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup
File Edit View Search Terminal Tabs Help

```
seed@ip-172-31-44-212:~/Internet Security/Lab-05/Labsetup$ docksh host-192.168.60.5
root@fd9379f17e0d:/# export PS1="V-192.168.60.5:\w\n\$> "
V-192.168.60.5:/
$> tcpdump -i eth0 -n 2>/dev/null
```

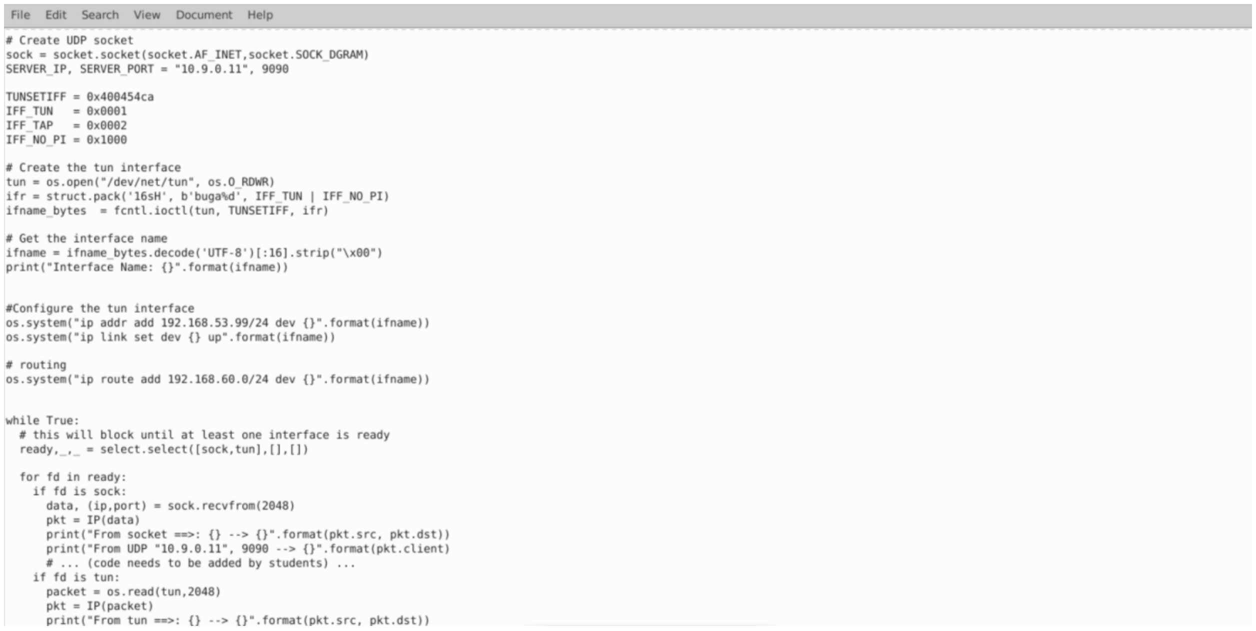


```
U-192.168.60.5:/
$> tcpdump -i eth0 -n 2>/dev/null
07:15:31.002641 ARP, Request who-has 192.168.60.11 tell 192.192.168.60.11 tell 192.168.60.5, length28
07:15:31.002721 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
07:15:37.835993 IP 192.168.50.5 > 192.168.60.5: ICMP echo request, id 37, seq 1, length 64
07:15:37.836024 IP 192.168.60.5 > 192.168.50.5: ICMP echo reply, id 37, seq 1, length 64
07:15:37.836114 IP 192.168.50.5 > 192.168.60.5: ICMP echo request, id 37, seq 2, length 64
07:15:37.836227 IP 192.168.60.5 > 192.168.50.5: ICMP echo reply, id 37, seq 2, length 64
```

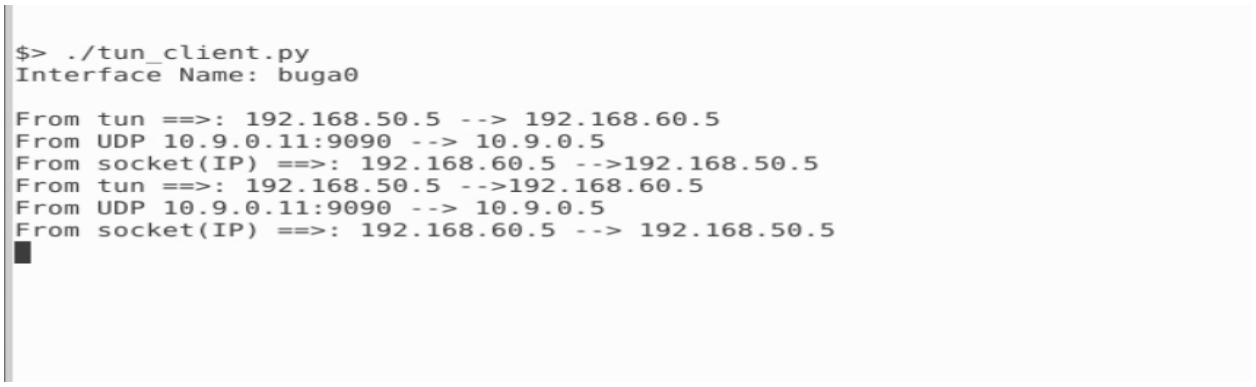

seed@ip-172-31-44-212: ~/Internet Security/Lab-05/Labsetup
File Edit View Search Terminal Tabs Help
seed@ip-...

```
U-192.168.50.5/
$> ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1008ms

U-192.168.50.5/
$> ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1018ms

U-192.168.50.5/
$> ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1034ms

U-192.168.50.5/
$> 
```

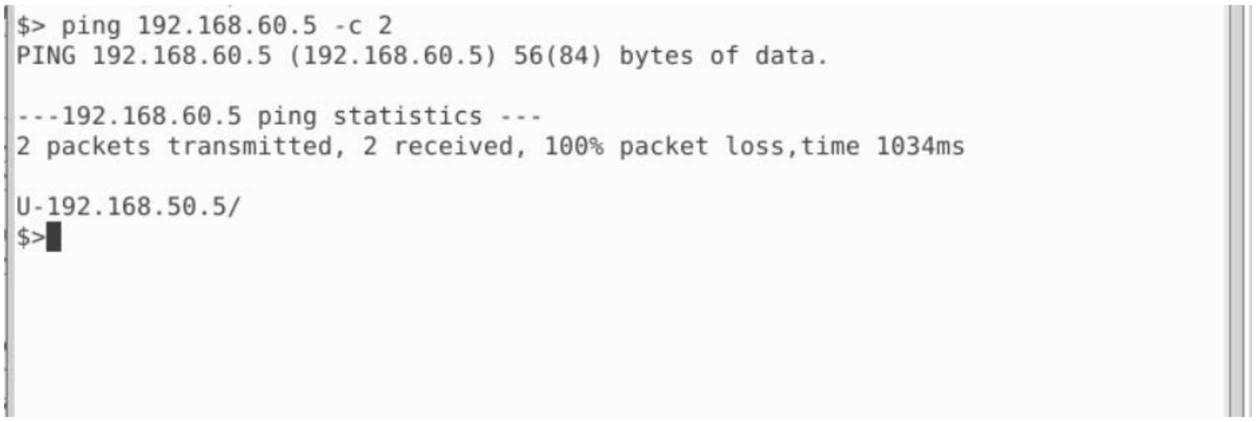
Source Code: -tun_client.py



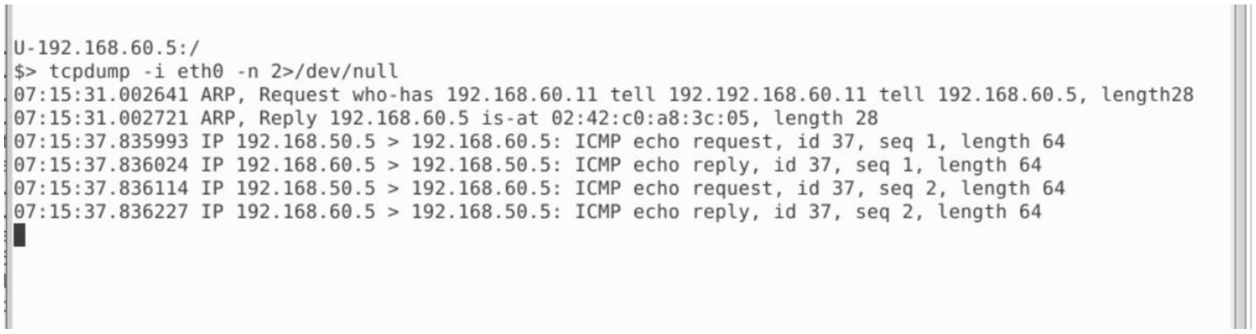
```
# Create UDP socket
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
SERVER_IP, SERVER_PORT = "10.9.0.11", 9090

TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'buga%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))


#Configure the tun interface
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# routing
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))


while True:
  # this will block until at least one interface is ready
  ready,_,_ = select.select([sock,tun],[],[])

  for fd in ready:
    if fd is sock:
      data, (ip,port) = sock.recvfrom(2048)
      pkt = IP(data)
      print("From socket ==>: {} --> {}".format(pkt.src, pkt.dst))
      print("From UDP "10.9.0.11", 9090 --> {}".format(pkt.client)
      # ... (code needs to be added by students) ...
    if fd is tun:
      packet = os.read(tun,2048)
      pkt = IP(packet)
      print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
```



```
$> ./tun_client.py
Interface Name: buga0

From tun ==>: 192.168.50.5 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 -->192.168.50.5
From tun ==>: 192.168.50.5 -->192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.50.5
```



```
$> ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

---192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 100% packet loss,time 1034ms

U-192.168.50.5/
$>
```



```
U-192.168.60.5:/
$> tcpdump -i eth0 -n 2>/dev/null
07:15:31.002641 ARP, Request who-has 192.168.60.11 tell 192.192.168.60.11 tell 192.168.60.5, length28
07:15:31.002721 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
07:15:37.835993 IP 192.168.50.5 > 192.168.60.5: ICMP echo request, id 37, seq 1, length 64
07:15:37.836024 IP 192.168.60.5 > 192.168.50.5: ICMP echo reply, id 37, seq 1, length 64
07:15:37.836114 IP 192.168.50.5 > 192.168.60.5: ICMP echo request, id 37, seq 2, length 64
07:15:37.836227 IP 192.168.60.5 > 192.168.50.5: ICMP echo reply, id 37, seq 2, length 64
```

When we ping the host U we see 2 packets transmitted and 2 received, so there’s a host U

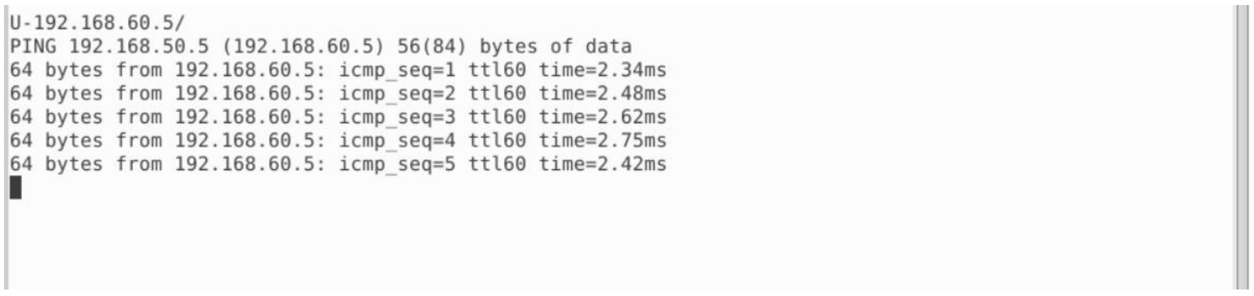
Now we check the vpn client we see how the packets are transmitted. The packet is send into the tunnel. We have the payload

There's a reply from 50.5 to 60.5 then goes through the tunnel, then it extracts the payload for the VPN client

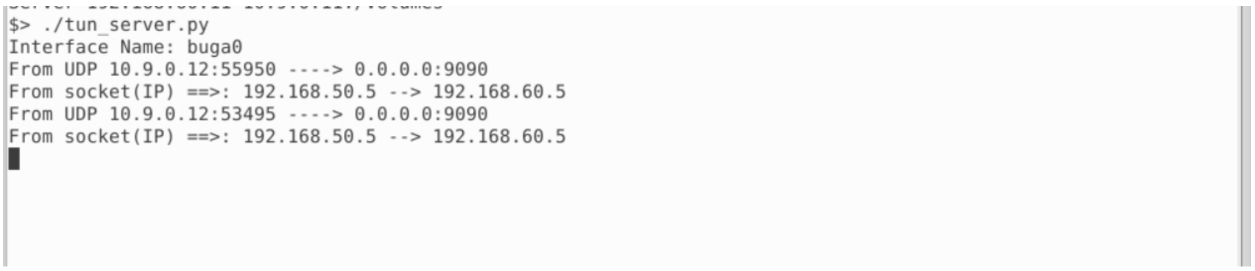
In VPN server we see the similar output.

On host V we see the packets echo request and echo reply, so the tunnel is set up.

Now again we can ping on host U to send out the ping request continuously.



```
U-192.168.60.5/
PING 192.168.50.5 (192.168.60.5) 56(84) bytes of data
64 bytes from 192.168.60.5: icmp_seq=1 ttl60 time=2.34ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl60 time=2.48ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl60 time=2.62ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl60 time=2.75ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl60 time=2.42ms
```

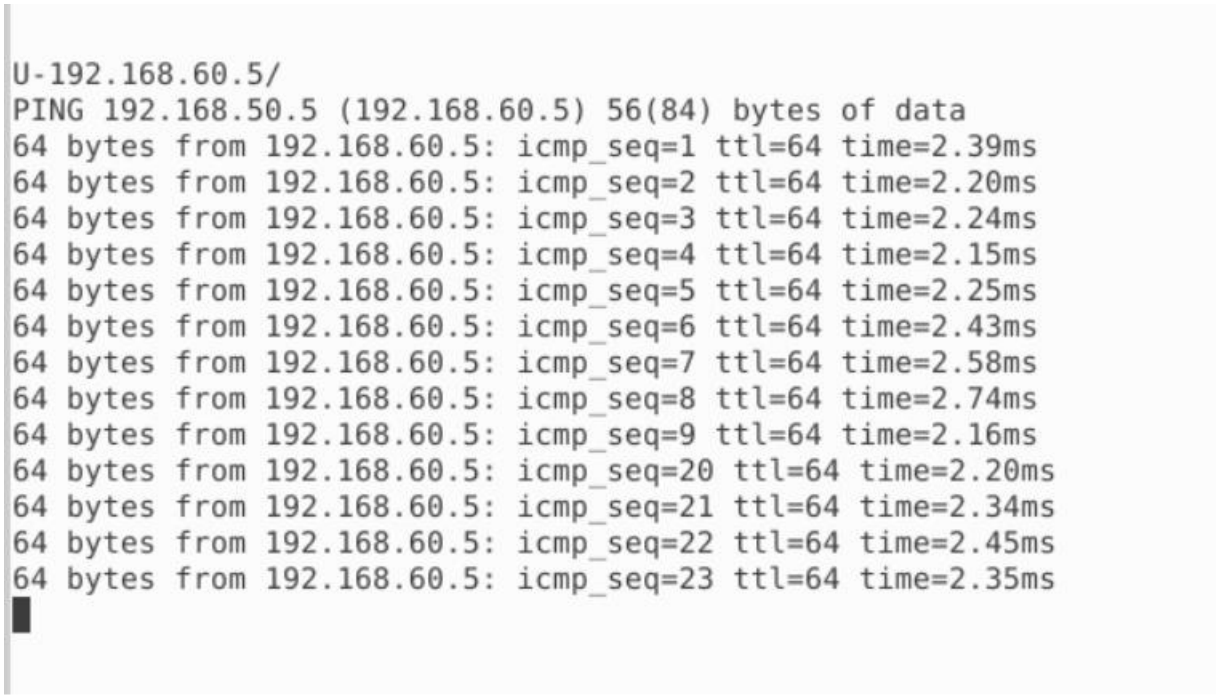


```
$> ./tun_server.py
Interface Name: buga0
From UDP 10.9.0.12:55950 ----> 0.0.0.0:9090
From socket(IP) ==>: 192.168.50.5 --> 192.168.60.5
From UDP 10.9.0.12:53495 ----> 0.0.0.0:9090
From socket(IP) ==>: 192.168.50.5 --> 192.168.60.5
█
```

Now we stop the client by pressing Ctrl+C to stop it and you see here the ping is also stopped

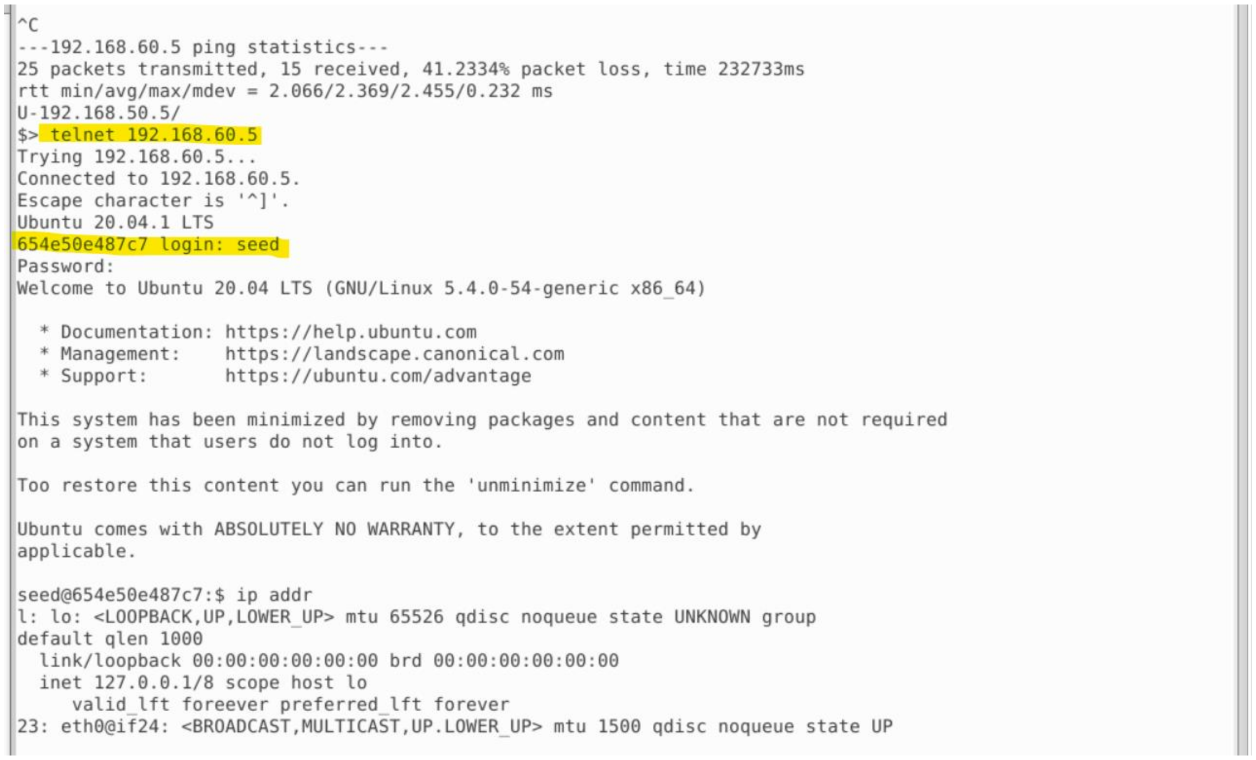
Here again we set up the tunnel and by running the ./tun_client.py and you will see the ping continues.

Now stop the ping, we will see the sequence number from 9 to 20. 14 received and 24 packets transmitted and 24 – 14 is 10 here 20-9 is 11 we still need to minus 1 is the time vector x missed

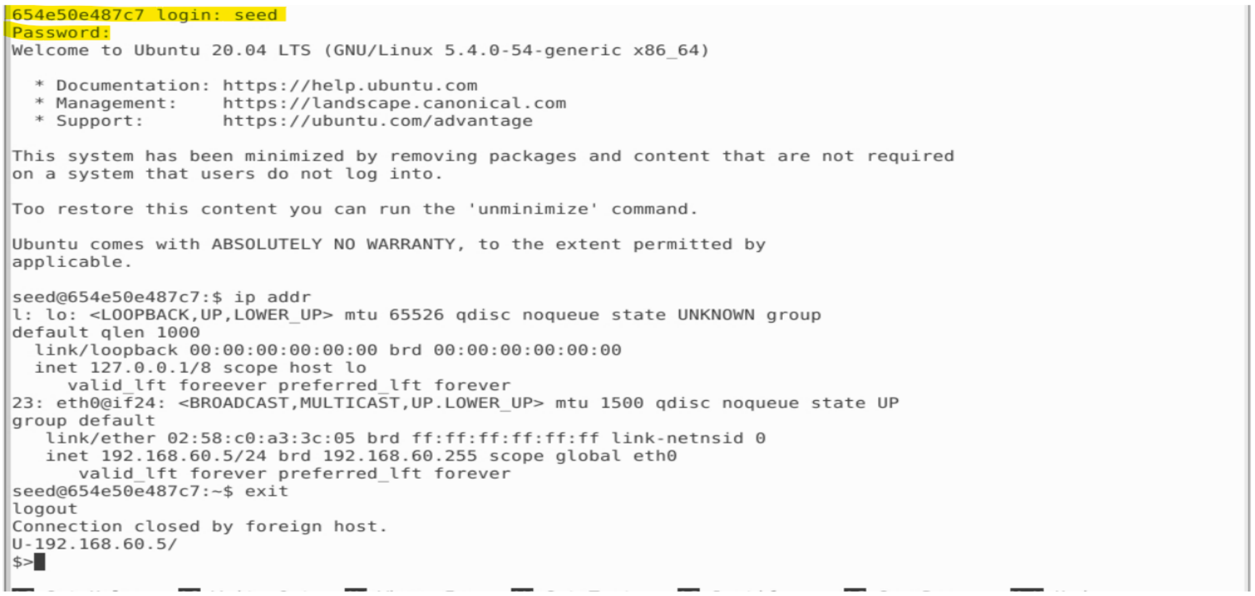


```
U-192.168.60.5/
PING 192.168.50.5 (192.168.60.5) 56(84) bytes of data
64 bytes from 192.168.60.5: icmp_seq=1 ttl=64 time=2.39ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=2.20ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=2.24ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=2.15ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=64 time=2.25ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=64 time=2.43ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=64 time=2.58ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=64 time=2.74ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=64 time=2.16ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=64 time=2.20ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=64 time=2.34ms
64 bytes from 192.168.60.5: icmp_seq=22 ttl=64 time=2.45ms
64 bytes from 192.168.60.5: icmp_seq=23 ttl=64 time=2.35ms
█
```

So we can see these are ping packets went through the tunnel. We can also use telnet, tunnel to this host V and we can see it has worked.



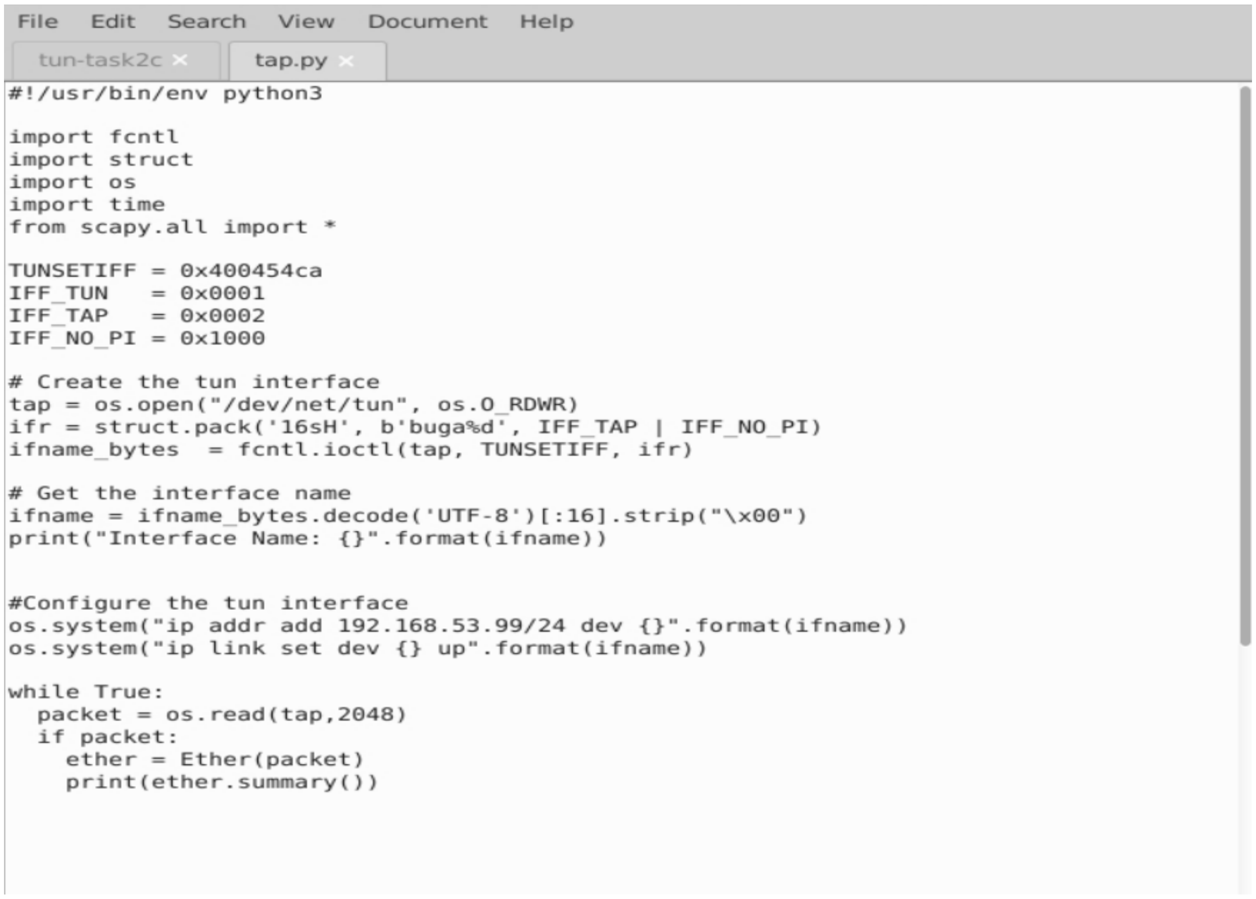
```
^C
---192.168.60.5 ping statistics---
25 packets transmitted, 15 received, 41.2334% packet loss, time 232733ms
rtt min/avg/max/mdev = 2.066/2.369/2.455/0.232 ms
U-192.168.50.5/
$> telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
654e50e487c7 login: seed
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-54-generic x86_64)

  * Documentation: https://help.ubuntu.com
  * Management:    https://landscape.canonical.com
  * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are not required
on a system that users do not log into.

Too restore this content you can run the 'unminimize' command.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable.

seed@654e50e487c7:$ ip addr
l: lo: <LOOPBACK,UP,LOWER_UP> mtu 65526 qdisc noqueue state UNKNOWN group
default qlen 1000
  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 scope host lo
     valid_lft foreever preferred_lft forever
23: eth0@if24: <BROADCAST,MULTICAST,UP.LOWER_UP> mtu 1500 qdisc noqueue state UP
```



```
654e50e487c7 login: seed
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-54-generic x86_64)

  * Documentation: https://help.ubuntu.com
  * Management:    https://landscape.canonical.com
  * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are not required
on a system that users do not log into.

Too restore this content you can run the 'unminimize' command.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable.

seed@654e50e487c7:$ ip addr
l: lo: <LOOPBACK,UP,LOWER_UP> mtu 65526 qdisc noqueue state UNKNOWN group
default qlen 1000
  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 scope host lo
     valid_lft foreever preferred_lft forever
23: eth0@if24: <BROADCAST,MULTICAST,UP.LOWER_UP> mtu 1500 qdisc noqueue state UP
group default
   link/ether 02:58:c0:a3:3c:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
   inet 192.168.60.5/24 brd 192.168.60.255 scope global eth0
      valid_lft forever preferred_lft forever
seed@654e50e487c7:~$ exit
logout
Connection closed by foreign host.
U-192.168.60.5/
$>
```

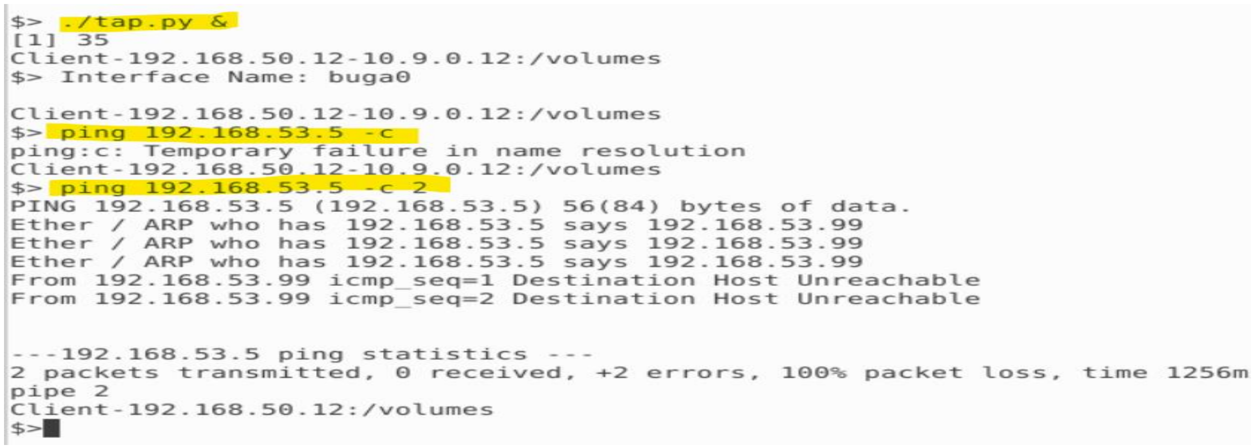
We set up the tunnel with VPN tunnel an we can route the packets from host U to host V and back.We test it with ping and telnet those are the proofs

Task 9: Experiment with the TAP Interface



```
#!/usr/bin/env python3

import fcntl
import struct
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tap = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'buga%d', IFF_TAP | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tap, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))


#Configure the tun interface
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

while True:
  packet = os.read(tap,2048)
  if packet:
    ether = Ether(packet)
    print(ether.summary())
```

We run tap.py in the VPN client



```
$> ./tap.py &
[1] 35
Client-192.168.50.12-10.9.0.12:/volumes
$> Interface Name: buga0

Client-192.168.50.12-10.9.0.12:/volumes
$> ping 192.168.53.5 -c
ping:c: Temporary failure in name resolution
Client-192.168.50.12-10.9.0.12:/volumes
$> ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
Ether / ARP who has 192.168.53.5 says 192.168.53.99
Ether / ARP who has 192.168.53.5 says 192.168.53.99
Ether / ARP who has 192.168.53.5 says 192.168.53.99
From 192.168.53.99 icmp_seq=1 Destination Host Unreachable
From 192.168.53.99 icmp_seq=2 Destination Host Unreachable


---192.168.53.5 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1256m
pipe 2
Client-192.168.50.12:/volumes
$>
```

We ping twice first in what ping the machine need to find the physical address

1) Here the tap interface is 192.168.53.5 which means it asks for physical address, it's sent on three times

and here is the print out by the ping command to send out to a ping packet request destination host unreachable

The three lines are output by the print summary

and this one is the print out by the print command.

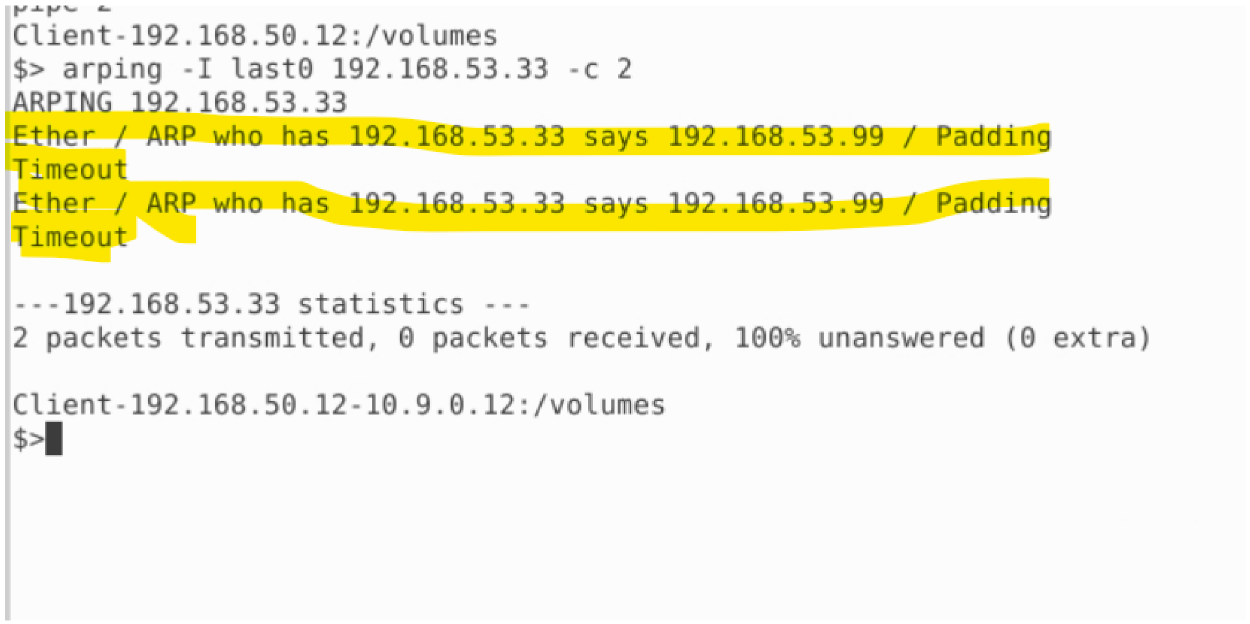
Even though 192.168.53.5 does not exist,the arp packets, the generator, which means its layer 2 packet or generator

2) Now we want to fake arp reply. Before we flick IP reply we can also run the below commands

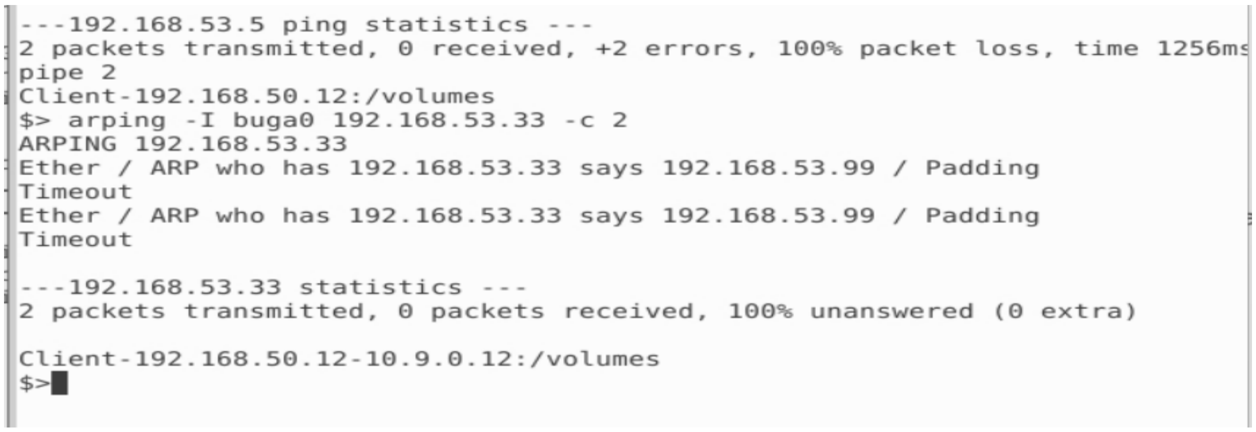
1. arping -I tap0 192.168.53.33

2.arping -I tap0 1.2.3.4

These commands try to find the physical address mapped to these IP address. We need to change the tap 0 to thelast name



```
Client-192.168.50.12:/volumes
$> arping -I last0 192.168.53.33 -c 2
ARPING 192.168.53.33
Ether / ARP who has 192.168.53.33 says 192.168.53.99 / Padding
Timeout
Ether / ARP who has 192.168.53.33 says 192.168.53.99 / Padding
Timeout

---192.168.53.33 statistics ---
2 packets transmitted, 0 packets received, 100% unanswered (0 extra)

Client-192.168.50.12-10.9.0.12:/volumes
$>
```

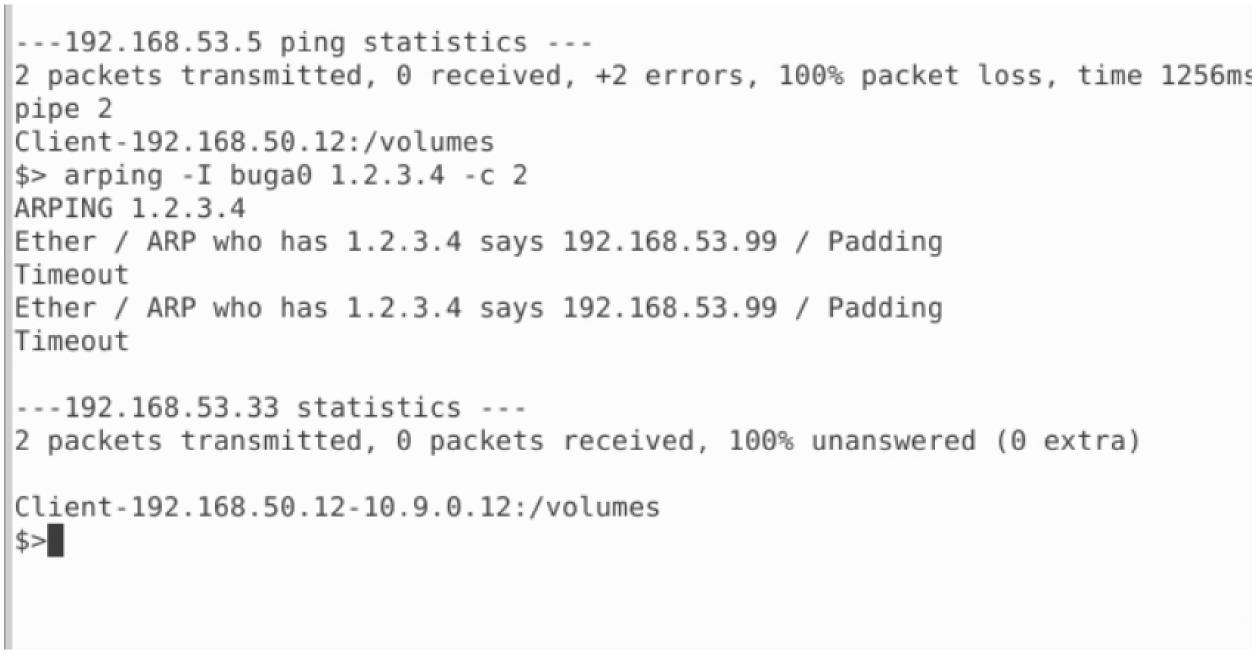

```
---192.168.53.5 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1256ms
pipe 2
Client-192.168.50.12:/volumes
$> arping -I buga0 192.168.53.33 -c 2
ARPING 192.168.53.33
Ether / ARP who has 192.168.53.33 says 192.168.53.99 / Padding
Timeout
Ether / ARP who has 192.168.53.33 says 192.168.53.99 / Padding
Timeout

---192.168.53.33 statistics ---
2 packets transmitted, 0 packets received, 100% unanswered (0 extra)

Client-192.168.50.12-10.9.0.12:/volumes
$>
```

These two lines are printed out by the physical address asked for the physical address.

Now this 33 host from our last 0. The IP address for our last 0 is .99 it’s configured in the code

The Ip address 192.168.53.33 is in the same local area network as type 0.

The other one-1.2.3.4 is outside IP address


```
---192.168.53.5 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1256ms
pipe 2
Client-192.168.50.12:/volumes
$> arping -I buga0 1.2.3.4 -c 2
ARPING 1.2.3.4
Ether / ARP who has 1.2.3.4 says 192.168.53.99 / Padding
Timeout
Ether / ARP who has 1.2.3.4 says 192.168.53.99 / Padding
Timeout

---192.168.53.33 statistics ---
2 packets transmitted, 0 packets received, 100% unanswered (0 extra)

Client-192.168.50.12-10.9.0.12:/volumes
$>
```

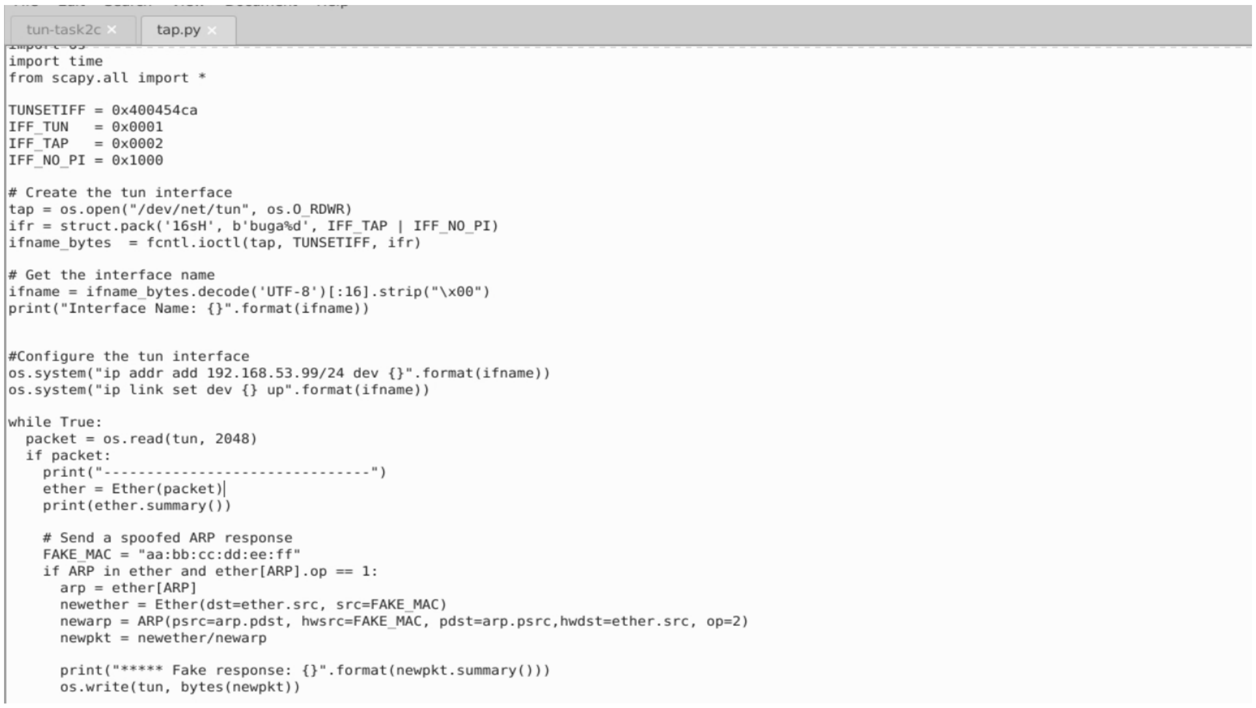
We are still getting it the print out by our program similar to the previous one. Two packets of IP transmission received.

On the second part we are going to fake or spoof and IP response using the given code

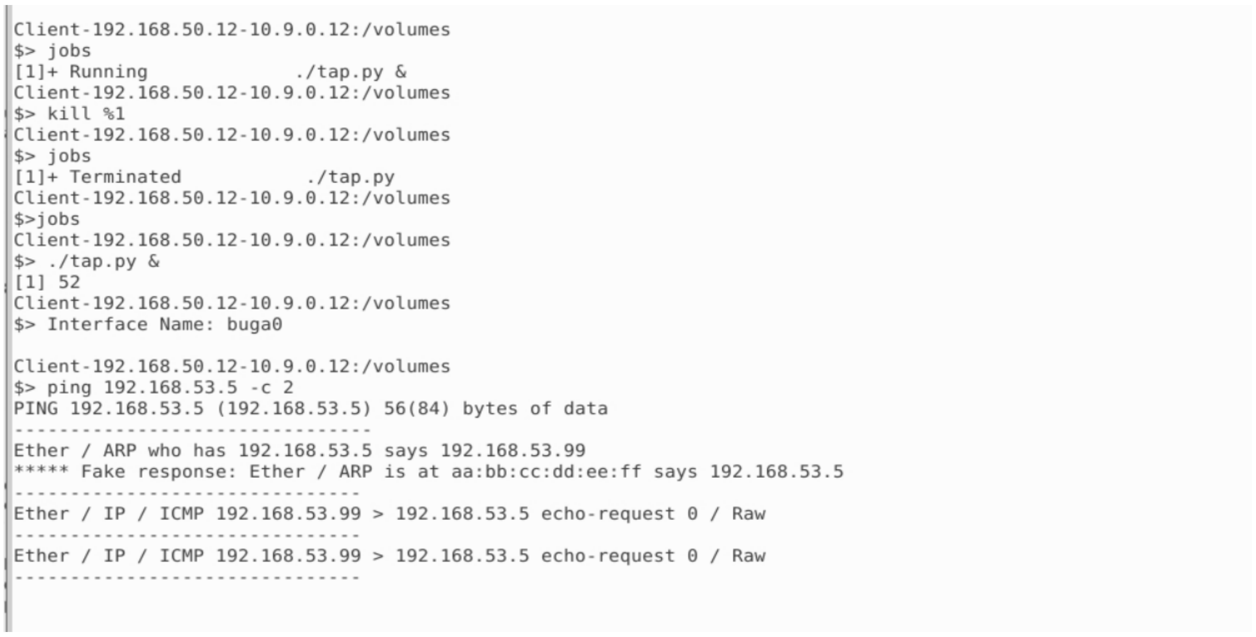
Source Code: (replacement part)

```
while True:
packet = os.read(tun, 2048)
if packet:
print("--------------------------------")
ether = Ether(packet)
print(ether.summary())
# Send a spoofed ARP response
FAKE_MAC = "aa:bb:cc:dd:ee:ff"
if ARP in ether and ether[ARP].op == 1 :
arp = ether[ARP]
newether = Ether(dst=ether.src, src=FAKE_MAC)
newarp = ARP(psrc=arp.pdst, hwsrc=FAKE_MAC,
pdst=arp.psrc, hwdst=ether.src, op=2)
newpkt = newether/newarp
print("***** Fake response: {}".format(newpkt.summary()))
os.write(tun, bytes(newpkt))
```

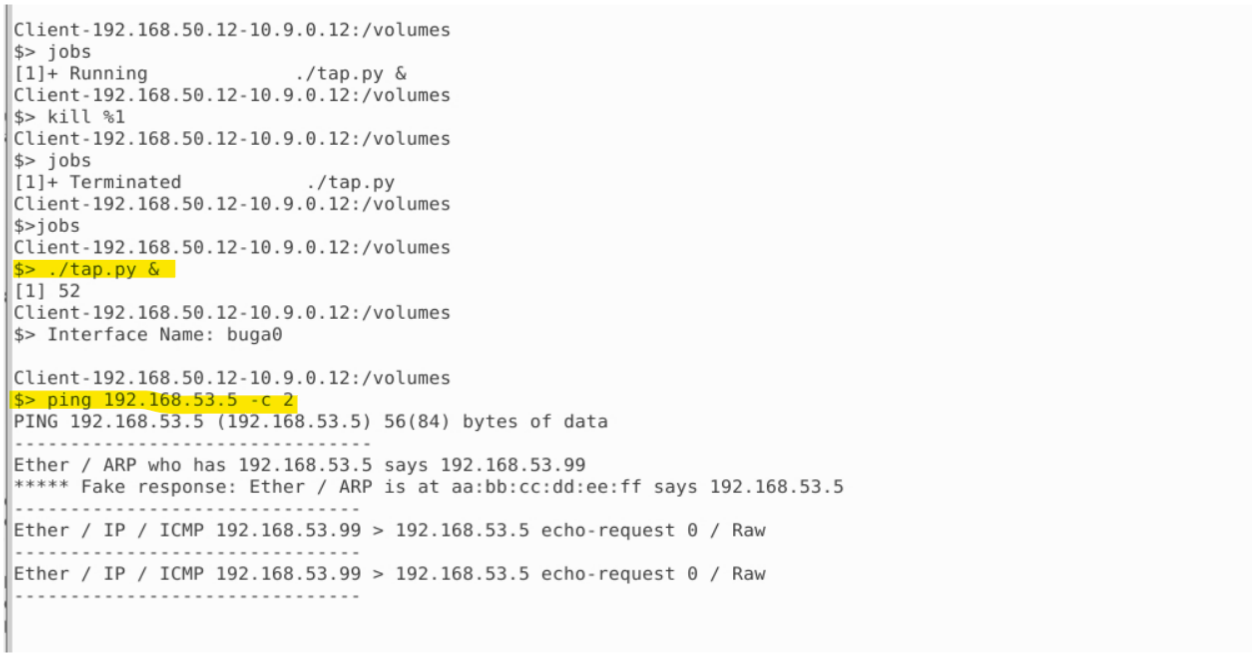
Replace the while loop



```
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tap = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'buga%d', IFF_TAP | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tap, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))


#Configure the tun interface
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

while True:
  packet = os.read(tun, 2048)
  if packet:
    print("------------------------------")
    ether = Ether(packet)
    print(ether.summary())

    # Send a spoofed ARP response
    FAKE_MAC = "aa:bb:cc:dd:ee:ff"
    if ARP in ether and ether[ARP].op == 1:
      arp = ether[ARP]
      newether = Ether(dst=ether.src, src=FAKE_MAC)
      newarp = ARP(psrc=arp.pdst, hwsrc=FAKE_MAC, pdst=arp.psrc,hwdst=ether.src, op=2)
      newpkt = newether/newarp

      print("***** Fake response: {}".format(newpkt.summary()))
      os.write(tun, bytes(newpkt))
```

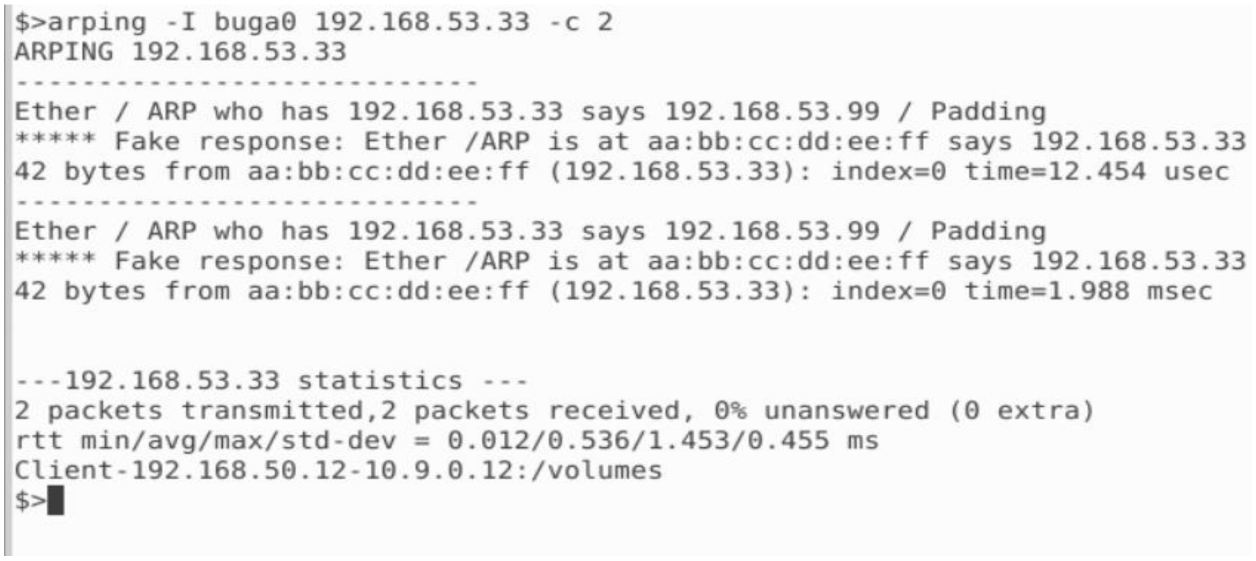


```
Client-192.168.50.12-10.9.0.12:/volumes
$> jobs
[1]+ Running              ./tap.py &
Client-192.168.50.12-10.9.0.12:/volumes
$> kill %1
Client-192.168.50.12-10.9.0.12:/volumes
$> jobs
[1]+ Terminated           ./tap.py
Client-192.168.50.12-10.9.0.12:/volumes
$>jobs
Client-192.168.50.12-10.9.0.12:/volumes
$> ./tap.py &
[1] 52
Client-192.168.50.12-10.9.0.12:/volumes
$> Interface Name: buga0

Client-192.168.50.12-10.9.0.12:/volumes
$> ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data
--------------------------------
Ether / ARP who has 192.168.53.5 says 192.168.53.99
***** Fake response: Ether / ARP is at aa:bb:cc:dd:ee:ff says 192.168.53.5
-------------------------------
Ether / IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
-------------------------------
Ether / IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
-------------------------------
```



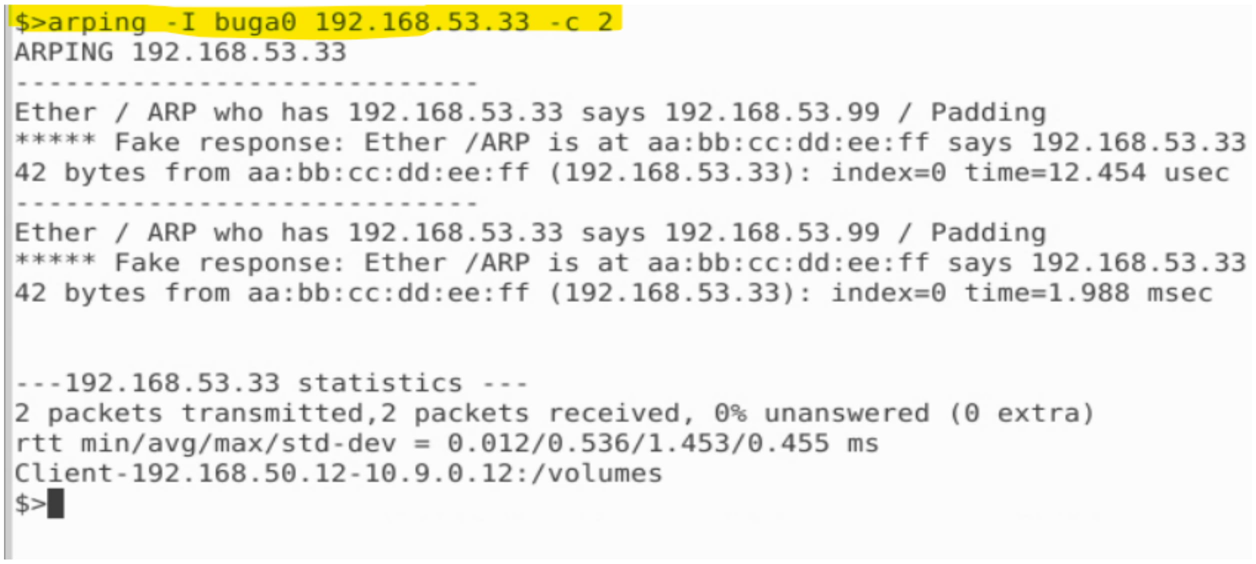
```
Client-192.168.50.12-10.9.0.12:/volumes
$> jobs
[1]+ Running              ./tap.py &
Client-192.168.50.12-10.9.0.12:/volumes
$> kill %1
Client-192.168.50.12-10.9.0.12:/volumes
$> jobs
[1]+ Terminated           ./tap.py
Client-192.168.50.12-10.9.0.12:/volumes
$>jobs
Client-192.168.50.12-10.9.0.12:/volumes
$> ./tap.py &
[1] 52
Client-192.168.50.12-10.9.0.12:/volumes
$> Interface Name: buga0

Client-192.168.50.12-10.9.0.12:/volumes
$> ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data
--------------------------------
Ether / ARP who has 192.168.53.5 says 192.168.53.99
***** Fake response: Ether / ARP is at aa:bb:cc:dd:ee:ff says 192.168.53.5
-------------------------------
Ether / IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
-------------------------------
Ether / IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
-------------------------------
```

We get a fake response that .file is at this place- aa:bb:cc:dd:ee:ff. The ARP request successfully getting a fake response.



```
$>arping -I buga0 192.168.53.33 -c 2
ARPING 192.168.53.33
----------------------------
Ether / ARP who has 192.168.53.33 says 192.168.53.99 / Padding
***** Fake response: Ether /ARP is at aa:bb:cc:dd:ee:ff says 192.168.53.33
42 bytes from aa:bb:cc:dd:ee:ff (192.168.53.33): index=0 time=12.454 usec
----------------------------
Ether / ARP who has 192.168.53.33 says 192.168.53.99 / Padding
***** Fake response: Ether /ARP is at aa:bb:cc:dd:ee:ff says 192.168.53.33
42 bytes from aa:bb:cc:dd:ee:ff (192.168.53.33): index=0 time=1.988 msec


---192.168.53.33 statistics ---
2 packets transmitted,2 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.012/0.536/1.453/0.455 ms
Client-192.168.50.12-10.9.0.12:/volumes
$>
```

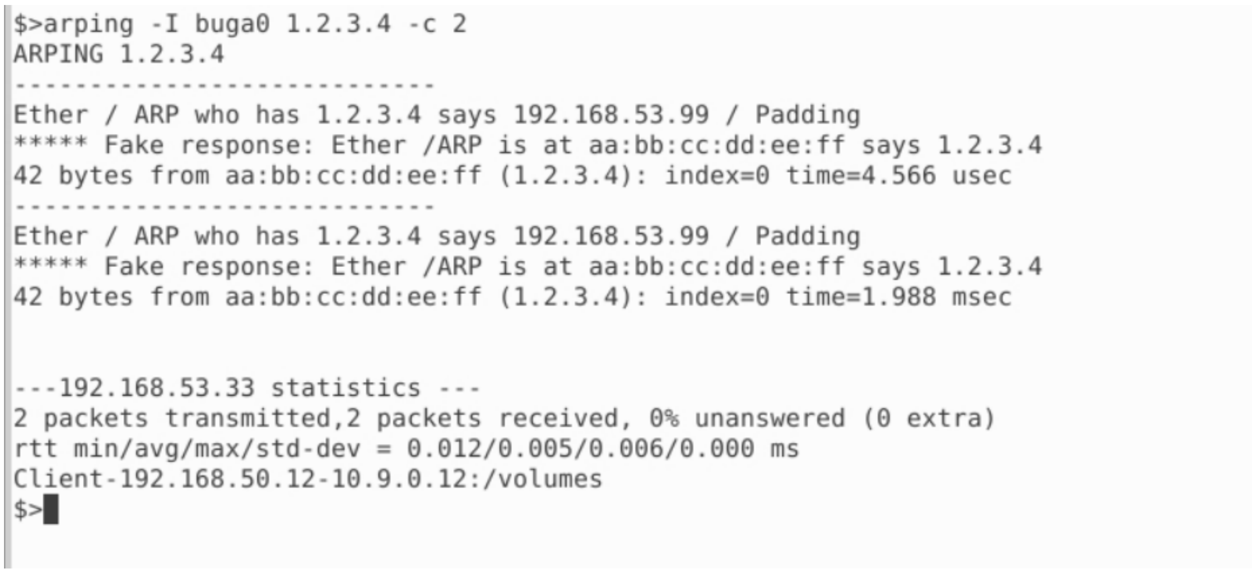
Here we get two packets transmitted and 2 packets received



```
$>arping -I buga0 192.168.53.33 -c 2
ARPING 192.168.53.33
----------------------------
Ether / ARP who has 192.168.53.33 says 192.168.53.99 / Padding
***** Fake response: Ether /ARP is at aa:bb:cc:dd:ee:ff says 192.168.53.33
42 bytes from aa:bb:cc:dd:ee:ff (192.168.53.33): index=0 time=12.454 usec
----------------------------
Ether / ARP who has 192.168.53.33 says 192.168.53.99 / Padding
***** Fake response: Ether /ARP is at aa:bb:cc:dd:ee:ff says 192.168.53.33
42 bytes from aa:bb:cc:dd:ee:ff (192.168.53.33): index=0 time=1.988 msec


---192.168.53.33 statistics ---
2 packets transmitted,2 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.012/0.536/1.453/0.455 ms
Client-192.168.50.12-10.9.0.12:/volumes
$>
```

We get a fake response that .file is at this place- aa:bb:cc:dd:ee:ff. The ARP request successfully getting a fake response, however the location aa:bb:cc:dd:ee:ff does not exist. The ping packets are also passed to our last zero and now we can see when we try to send this ping request packet to this 192.168.53.5 now exist the host at this physical address. We just used ping packets and zero received because it does not exist.
Now host with this IP address exist because we give it a fake arp request.
We tell this ping command saying this IP address 192.168.53.5 is mapped to this physical address thats why our ping failed and IP succeeded and for request will get a fake response.
Now with the arp ping we should be able to see to fake response

We should get the similar result for the arping -I buga0 1.2.3.4 -c 2


```
$>arping -I buga0 1.2.3.4 -c 2
ARPING 1.2.3.4
----------------------------
Ether / ARP who has 1.2.3.4 says 192.168.53.99 / Padding
***** Fake response: Ether /ARP is at aa:bb:cc:dd:ee:ff says 1.2.3.4
42 bytes from aa:bb:cc:dd:ee:ff (1.2.3.4): index=0 time=4.566 usec
----------------------------
Ether / ARP who has 1.2.3.4 says 192.168.53.99 / Padding
***** Fake response: Ether /ARP is at aa:bb:cc:dd:ee:ff says 1.2.3.4
42 bytes from aa:bb:cc:dd:ee:ff (1.2.3.4): index=0 time=1.988 msec


---192.168.53.33 statistics ---
2 packets transmitted,2 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.012/0.005/0.006/0.000 ms
Client-192.168.50.12-10.9.0.12:/volumes
$>
```