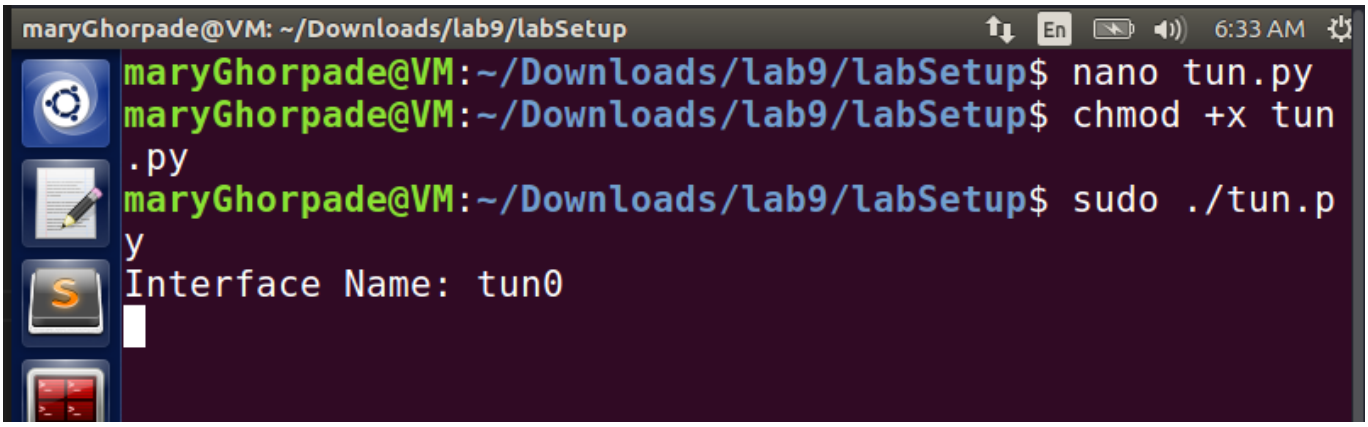
VPN Tunneling Lab

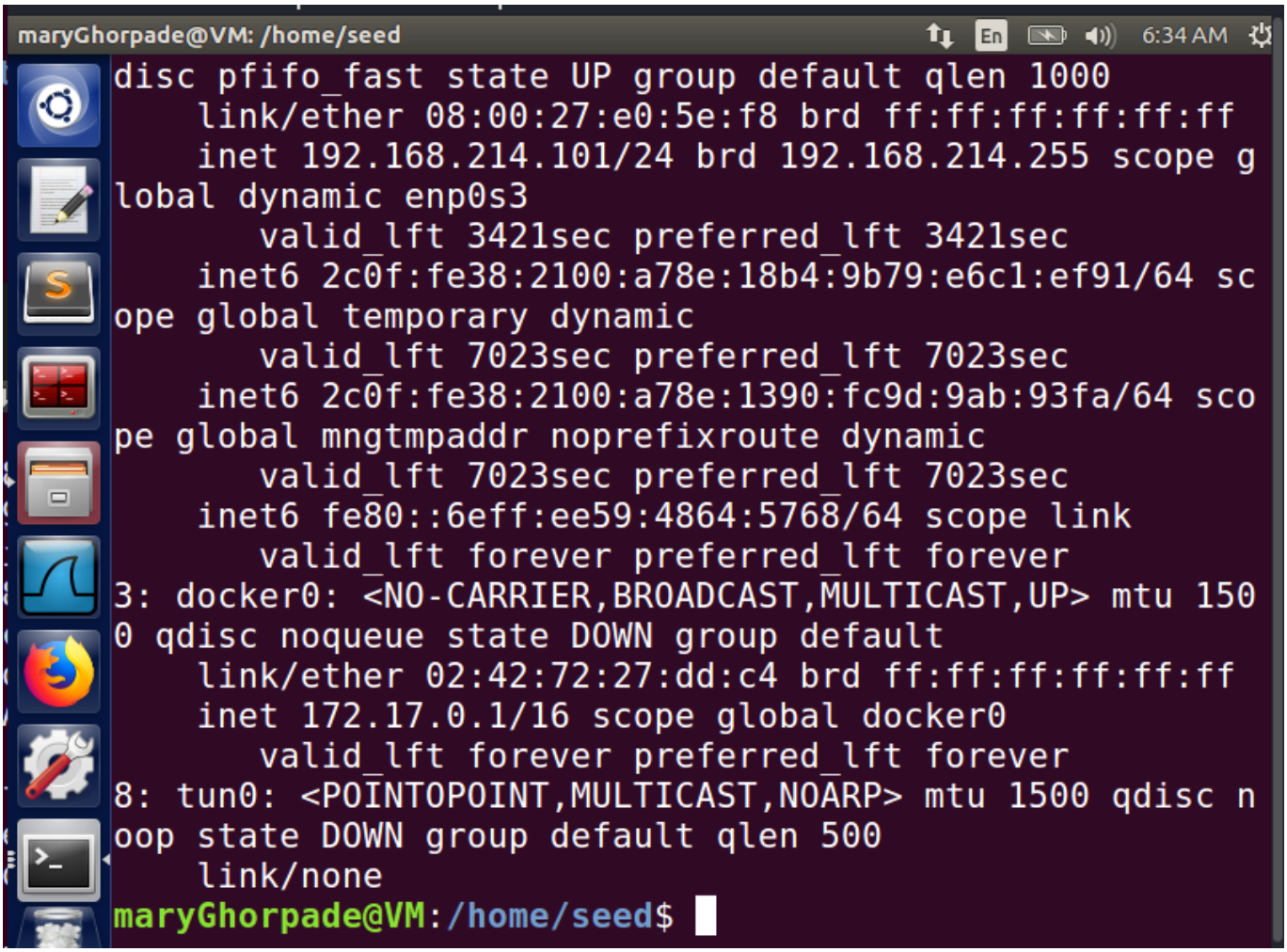
Objectives

- Virtual Private Network
- The TUN/TAP virtual interface
- IP tunneling
- Routing

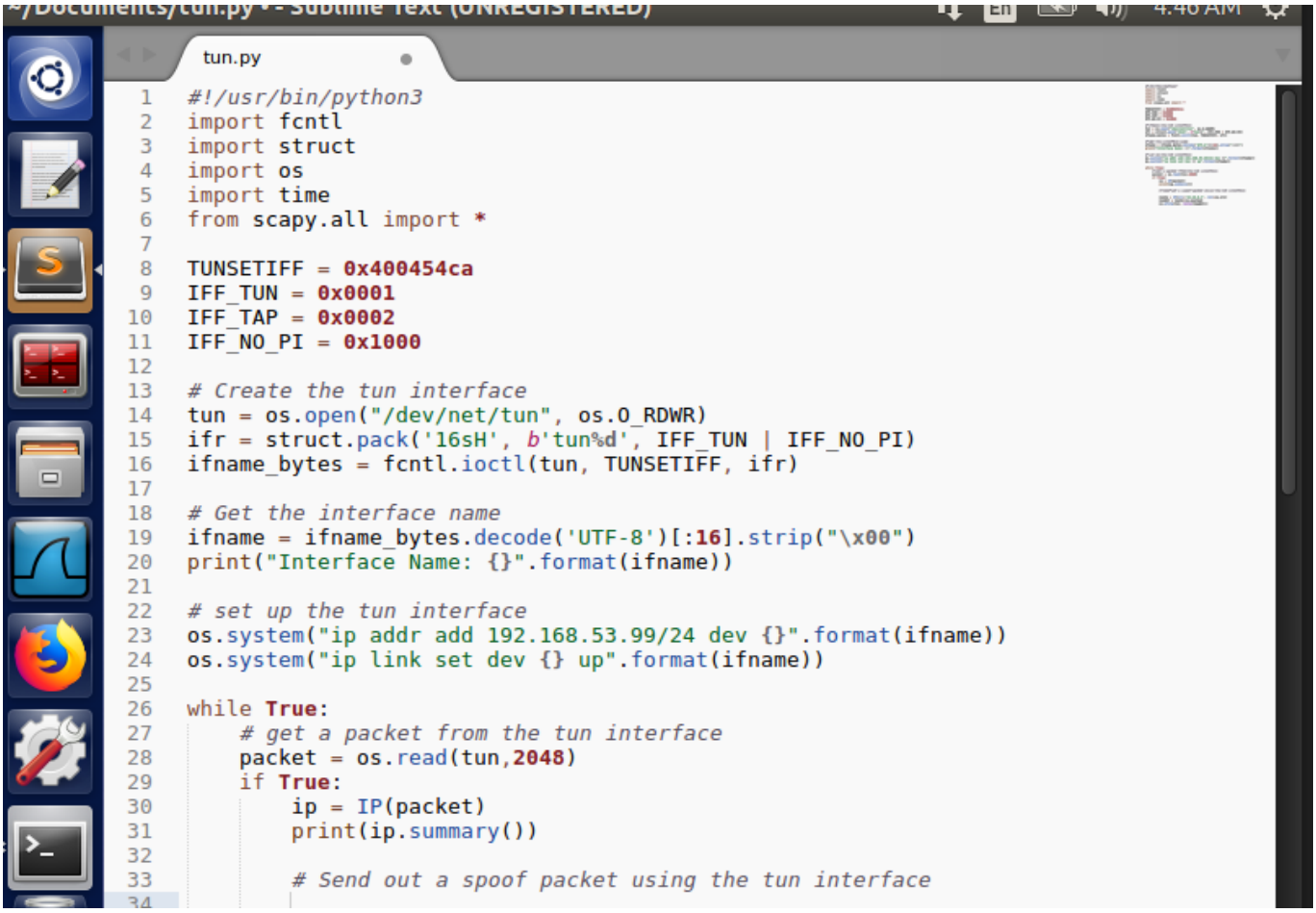
Creating and configuring Tun interface



A tun interface has been created after running the program



client_tun.py



```python
#!/usr/bin/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
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))

# set up 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:
    # get a packet from the tun interface
    packet = os.read(tun,2048)
    if True:
        ip = IP(packet)
        print(ip.summary())

        # Send out a spoof packet using the tun interface
```

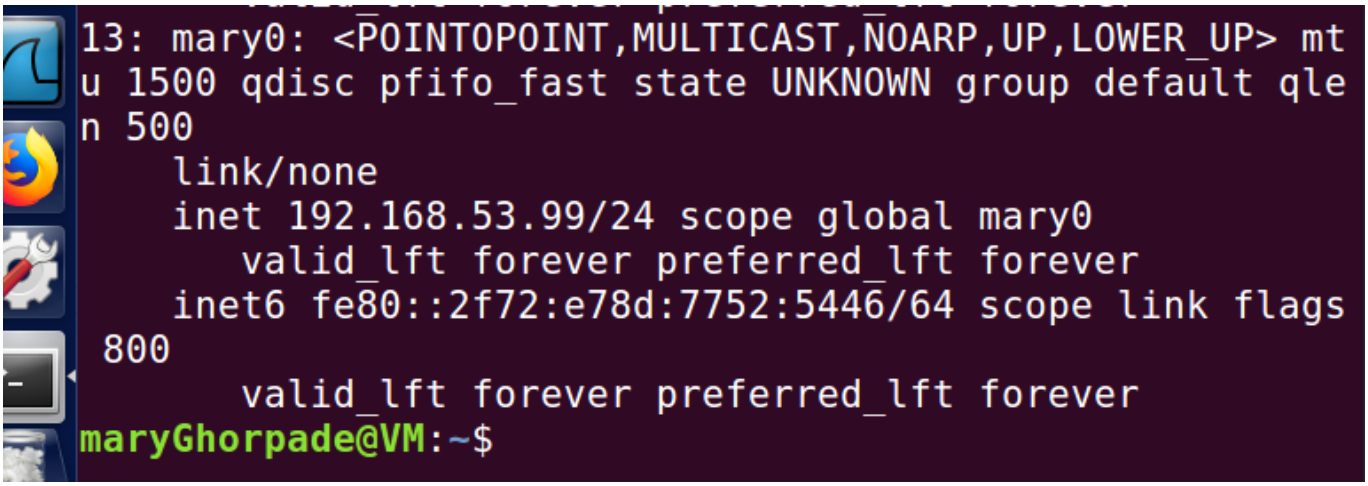
Setting up TUN interface

Assign it an IP address to it and bring the interface up

> sudo ip addr add 192.168.53.99/24 dev tun0

> sudo ip link set dev tun0 up

The interface is assigned an ip as seen below



```
13: mary0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mt
u 1500 qdisc pfifo_fast state UNKNOWN group default qle
n 500
    link/none
    inet 192.168.53.99/24 scope global mary0
       valid_lft forever preferred_lft forever
    inet6 fe80::2f72:e78d:7752:5446/64 scope link flags
 800
       valid_lft forever preferred_lft forever
maryGhorpade@VM:~$
```

Reading TUN interface

```
while True:
    # Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if True:
        ip = IP(packet)
        print(ip.summary())
```

It indicates that the VPN Server received the ICMP packets from the tun interface (192.168.53.99) which are intended for the chosen host in 192.168.53.0/24 (192.168.53.20).

Write to the TUN Interface

```
# Send out a spoof packet using the tun interface
newip = IP(src='1.2.3.4', dst=ip.src)
newpkt = newip/ip.payload
os.write(tun, bytes(newpkt))
```

Set up VPN Server

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

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000
TUN_IP = "192.168.53.98"

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

IP forwarding can be enabled using the following command

> sudo sysctl net.ipv4.ip_forward=1

It should be noted that although Host V will respond to the ICMP packets, the reply will not get back to Host U, because we have not set up everything yet. Therefore, for this task, it is sufficient to show the ICMP packets have arrived at Host V.

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
02:17:11.772281 IP 192.168.60.1.5353 > 224.0.0.251.5353: 0 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
02:17:11.773027 IP6 fe80::42:32ff:feae:f7a7.5353 > ff02::fb.5353: 0 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
02:17:11.773615 IP6 fe80::fcd1:4cff:feb2:2529.5353 > ff02::fb.5353: 0 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
```

```
9.0.!
Insi
9.0.!
Insi
9.0.5:36661 -->
Inside: 192.168.
9.0.5:36661 -->
Inside: 192.168.
9.0.5:36661 -->
Inside: 192.168.
9.0.5:36661 -->
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
03:46:21.707911 IP 192.168.53.99 > ced9612d0882: ICMP echo request, id 52, seq 196
, length 64
03:46:21.708011 IP ced9612d0882 > 192.168.53.99: ICMP echo reply, id 52, seq 196,
length 64
03:46:21.715608 IP ced9612d0882.35106 > 192.168.1.1.domain: 30836+ PTR? 99.53.168.
192.in-addr.arpa. (44)
03:46:22.724393 IP 192.168.53.99 > ced9612d0882: ICMP echo request, id 52, seq 197
, length 64
03:46:22.724428 IP ced9612d0882 > 192.168.53.99: ICMP echo reply, id 52, seq 197,
length 64
```