

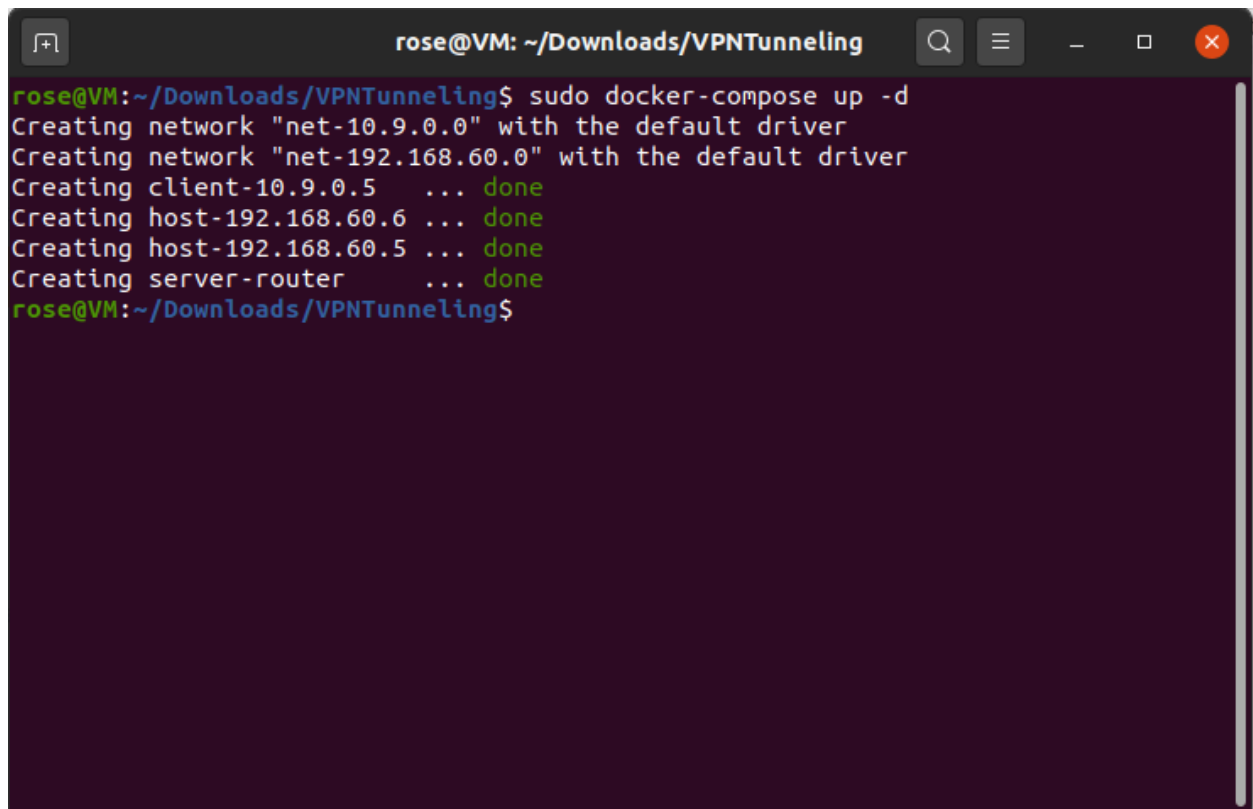
Samantha Jackson  
CSCI4321  
04-14-2025

## Lab 9 - VPN Tunneling

### Composing a Docker

I downloaded the files for the lab into my downloads folder in Ubuntu under the name 'VPNTunneling' and composed a docker using 'sudo docker-compose up -d'.

Observations/Issues: None

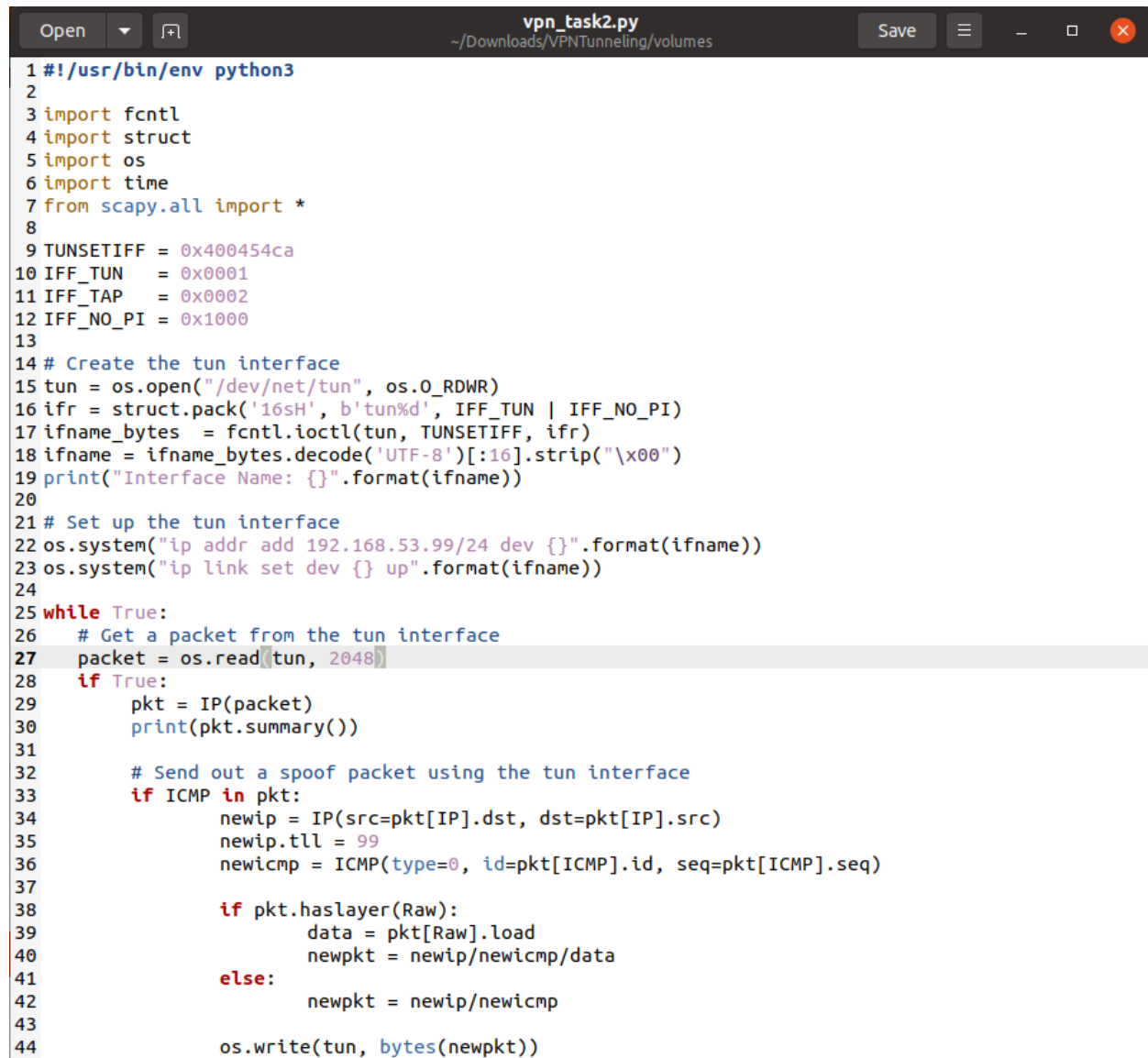
A terminal window titled 'rose@VM: ~/Downloads/VPNTunneling' with standard window controls. The terminal shows the execution of 'sudo docker-compose up -d'. The output indicates the creation of two networks ('net-10.9.0.0' and 'net-192.168.60.0') and four services ('client-10.9.0.5', 'host-192.168.60.6', 'host-192.168.60.5', and 'server-router'), all of which were successfully created and are now running in detached mode. The prompt returns to the shell.

```
rose@VM:~/Downloads/VPNTunneling$ sudo docker-compose up -d
Creating network "net-10.9.0.0" with the default driver
Creating network "net-192.168.60.0" with the default driver
Creating client-10.9.0.5    ... done
Creating host-192.168.60.6 ... done
Creating host-192.168.60.5 ... done
Creating server-router     ... done
rose@VM:~/Downloads/VPNTunneling$
```

## Task-02: VPN Tunneling

I created a program 'vpn\_task2.py' that creates a virtual interface 'tun0' used to intercept traffic. I logged into the VPN Client terminal on two separate instances of the terminal using the command 'sudo docker exec -it client-10.9.0.5 bash'. On the right I'm executing the vpn\_task2.py script in /volumes. On the left I'm pinging 192.168.53.99 and I'm getting continuous replies for this IP that shouldn't exist demonstrating that the 'tun0' interface is working.

Observations/Issues: None

A screenshot of a code editor window titled 'vpn\_task2.py' with the path '~/.Downloads/VPNTunneling/volumes'. The editor shows a Python script that creates a virtual interface 'tun0' and intercepts traffic. The script uses the 'fcntl' module for ioctl operations, 'struct' for packing/unpacking, and 'scapy' for packet manipulation. It sets up the interface with 'IFF\_TUN' and 'IFF\_NO\_PI' flags, adds an IP address '192.168.53.99', and enters a loop to receive and spoof packets. The spoofing logic checks for ICMP packets and creates new IP and ICMP headers to forward the traffic.

```
1#!/usr/bin/env python3
2
3import fcntl
4import struct
5import os
6import time
7from scapy.all import *
8
9TUNSETIFF = 0x400454ca
10 IFF_TUN   = 0x0001
11 IFF_TAP   = 0x0002
12 IFF_NO_PI = 0x1000
13
14# Create the tun interface
15tun = os.open("/dev/net/tun", os.O_RDWR)
16ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
17ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
18ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
19print("Interface Name: {}".format(ifname))
20
21# Set up the tun interface
22os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
23os.system("ip link set dev {} up".format(ifname))
24
25while True:
26    # Get a packet from the tun interface
27    packet = os.read(tun, 2048)
28    if True:
29        pkt = IP(packet)
30        print(pkt.summary())
31
32    # Send out a spoof packet using the tun interface
33    if ICMP in pkt:
34        newip = IP(src=pkt[IP].dst, dst=pkt[IP].src)
35        newip.ttl = 99
36        newicmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
37
38        if pkt.haslayer(Raw):
39            data = pkt[Raw].load
40            newpkt = newip/newicmp/data
41        else:
42            newpkt = newip/newicmp
43
44    os.write(tun, bytes(newpkt))
```

```
rose@VM: ~  
root@6f6270cce0eb: /# ping 192.168.53.99  
PING 192.168.53.99 (192.168.53.99) 56(84) bytes of data.  
64 bytes from 192.168.53.99: icmp_seq=29 ttl=64 time=0.043 ms  
64 bytes from 192.168.53.99: icmp_seq=30 ttl=64 time=0.041 ms  
64 bytes from 192.168.53.99: icmp_seq=31 ttl=64 time=0.560 ms  
64 bytes from 192.168.53.99: icmp_seq=32 ttl=64 time=0.038 ms  
64 bytes from 192.168.53.99: icmp_seq=33 ttl=64 time=0.038 ms  
64 bytes from 192.168.53.99: icmp_seq=34 ttl=64 time=0.042 ms  
64 bytes from 192.168.53.99: icmp_seq=35 ttl=64 time=0.042 ms  
64 bytes from 192.168.53.99: icmp_seq=36 ttl=64 time=0.046 ms  
64 bytes from 192.168.53.99: icmp_seq=37 ttl=64 time=0.042 ms  
64 bytes from 192.168.53.99: icmp_seq=38 ttl=64 time=0.040 ms  
64 bytes from 192.168.53.99: icmp_seq=39 ttl=64 time=0.040 ms  
64 bytes from 192.168.53.99: icmp_seq=40 ttl=64 time=0.042 ms  
64 bytes from 192.168.53.99: icmp_seq=41 ttl=64 time=0.041 ms  
64 bytes from 192.168.53.99: icmp_seq=42 ttl=64 time=0.048 ms  
64 bytes from 192.168.53.99: icmp_seq=43 ttl=64 time=0.044 ms  
64 bytes from 192.168.53.99: icmp_seq=44 ttl=64 time=0.039 ms  
64 bytes from 192.168.53.99: icmp_seq=45 ttl=64 time=0.039 ms  
64 bytes from 192.168.53.99: icmp_seq=46 ttl=64 time=0.040 ms  
64 bytes from 192.168.53.99: icmp_seq=47 ttl=64 time=0.043 ms  
64 bytes from 192.168.53.99: icmp_seq=48 ttl=64 time=0.035 ms  
64 bytes from 192.168.53.99: icmp_seq=49 ttl=64 time=0.058 ms  
64 bytes from 192.168.53.99: icmp_seq=50 ttl=64 time=0.050 ms  
64 bytes from 192.168.53.99: icmp_seq=51 ttl=64 time=0.042 ms  
64 bytes from 192.168.53.99: icmp_seq=52 ttl=64 time=0.044 ms  
64 bytes from 192.168.53.99: icmp_seq=53 ttl=64 time=0.044 ms  
64 bytes from 192.168.53.99: icmp_seq=54 ttl=64 time=0.041 ms  
64 bytes from 192.168.53.99: icmp_seq=55 ttl=64 time=0.042 ms  
64 bytes from 192.168.53.99: icmp_seq=56 ttl=64 time=0.042 ms  
64 bytes from 192.168.53.99: icmp_seq=57 ttl=64 time=0.050 ms  
64 bytes from 192.168.53.99: icmp_seq=58 ttl=64 time=0.045 ms  
64 bytes from 192.168.53.99: icmp_seq=59 ttl=64 time=0.045 ms  
64 bytes from 192.168.53.99: icmp_seq=60 ttl=64 time=0.040 ms  
64 bytes from 192.168.53.99: icmp_seq=61 ttl=64 time=0.041 ms  
64 bytes from 192.168.53.99: icmp_seq=62 ttl=64 time=0.040 ms  
64 bytes from 192.168.53.99: icmp_seq=63 ttl=64 time=0.043 ms  
64 bytes from 192.168.53.99: icmp_seq=64 ttl=64 time=0.050 ms  
64 bytes from 192.168.53.99: icmp_seq=65 ttl=64 time=0.042 ms  
64 bytes from 192.168.53.99: icmp_seq=66 ttl=64 time=0.046 ms  
64 bytes from 192.168.53.99: icmp_seq=67 ttl=64 time=0.043 ms  
64 bytes from 192.168.53.99: icmp_seq=68 ttl=64 time=0.043 ms  
64 bytes from 192.168.53.99: icmp_seq=69 ttl=64 time=0.044 ms  
64 bytes from 192.168.53.99: icmp_seq=70 ttl=64 time=0.044 ms  
64 bytes from 192.168.53.99: icmp_seq=71 ttl=64 time=0.043 ms  
64 bytes from 192.168.53.99: icmp_seq=72 ttl=64 time=0.042 ms  
64 bytes from 192.168.53.99: icmp_seq=73 ttl=64 time=0.041 ms  
64 bytes from 192.168.53.99: icmp_seq=74 ttl=64 time=0.046 ms  
64 bytes from 192.168.53.99: icmp_seq=75 ttl=64 time=0.185 ms  
64 bytes from 192.168.53.99: icmp_seq=76 ttl=64 time=0.042 ms  
64 bytes from 192.168.53.99: icmp_seq=77 ttl=64 time=0.042 ms  
64 bytes from 192.168.53.99: icmp_seq=78 ttl=64 time=0.043 ms  
64 bytes from 192.168.53.99: icmp_seq=79 ttl=64 time=0.044 ms  
64 bytes from 192.168.53.99: icmp_seq=80 ttl=64 time=0.043 ms  
64 bytes from 192.168.53.99: icmp_seq=81 ttl=64 time=0.041 ms  
64 bytes from 192.168.53.99: icmp_seq=82 ttl=64 time=0.039 ms  
64 bytes from 192.168.53.99: icmp_seq=83 ttl=64 time=0.041 ms  
64 bytes from 192.168.53.99: icmp_seq=84 ttl=64 time=0.043 ms  
64 bytes from 192.168.53.99: icmp_seq=85 ttl=64 time=0.041 ms  
^C  
--- 192.168.53.99 ping statistics ---  
85 packets transmitted, 57 received, 32.9412% packet loss, time 86006ms  
rtt min/avg/max/ndev = 0.035/0.054/0.560/0.070 ms  
root@6f6270cce0eb: /#  
  
root@6f6270cce0eb: /volumes# ls  
tun.py vpn_task2.py  
root@6f6270cce0eb: /volumes# python3 vpn_task2.py  
Interface Name: tun0  
^C  
Traceback (most recent call last):  
  File "vpn_task2.py", line 27, in <module>  
    packet = os.read(tun, 2048)  
KeyboardInterrupt  
  
root@6f6270cce0eb: /volumes#
```

## Task-03: Send the IP Packets to VPN Server Through a Tunnel

I created two programs 'task3\_tun\_server.py' and 'task3\_tun\_client.py' one that handles the server side script and the other handling the client side script. On the right I'm executing the task3\_tun\_server.py and task3\_tun\_client.py script in /volumes. On the left I'm ping 192.168.60.5. You can see that this is entirely unidirectional and doesn't go to the server side with 100% packet loss on the ping side.

Observations/Issues: Stuff to fix in Task-04.

```
vpn_task2.py x docker-compose.yml x task3_tun_server.py x task3_tun_client.py x
1 #!/usr/bin/env python3
2
3 import fcntl
4 import struct
5 import os
6 import socket
7 from scapy.all import *
8
9 IP_A = "0.0.0.0"
10 PORT = 9090
11 SERVER_IP = "10.9.0.11" # IP of the VPN server (Router container)
12
13 TUNSETIFF = 0x400454ca
14 IFF_TUN = 0x0001
15 IFF_NO_PI = 0x1000
16
17 # Create TUN interface
18 tun = os.open("/dev/net/tun", os.O_RDWR)
19 ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
20 ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
21 ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
22 print("Interface Name: {}".format(ifname))
23
24 # Set up interface and routing
25 os.system(f"ip addr add 192.168.53.99/24 dev {ifname}")
26 os.system(f"ip link set dev {ifname} up")
27 os.system(f"ip route add 192.168.60.0/24 dev {ifname}")
28
29 # Create socket
30 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
31
32 # Read packets from TUN and send over socket
33 while True:
34     packet = os.read(tun, 2048)
35     pkt = IP(packet)
36     print(f"[TUN] {pkt.src} -> {pkt.dst}")
37     sock.sendto(packet, (SERVER_IP, PORT))
38
```

```
vpn_task2.py × docker-compose.yml × task3_tun_server.py × task3_tun_client.py ×
1 #!/usr/bin/env python3
2
3 import fcntl
4 import struct
5 import os
6 import socket
7 from scapy.all import *
8
9 IP_A = "0.0.0.0"
10 PORT = 9090
11
12 TUNSETIFF = 0x400454ca
13 IFF_TUN = 0x0001
14 IFF_NO_PI = 0x1000
15
16 # Create TUN interface
17 tun = os.open("/dev/net/tun", os.O_RDWR)
18 ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
19 ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
20 ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
21 print("Interface Name: {}".format(ifname))
22
23 # Set up interface
24 os.system(f"ip addr add 192.168.53.1/24 dev {ifname}")
25 os.system(f"ip link set dev {ifname} up")
26
27 # Create socket and bind
28 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
29 sock.bind((IP_A, PORT))
30 print(f"[UDP] Listening on {IP_A}:{PORT}")
31
32 # Receive and forward to TUN
33 while True:
34     data, (ip, port) = sock.recvfrom(2048)
35     pkt = IP(data)
36     print(f"[UDP] From {ip}:{port} | {pkt.src} -> {pkt.dst}")
37     os.write(tun, data)
38
```

Python3 ▾ Tab Width: 8 ▾ Ln 12, Col 23 ▾ INS

[illegible]

