VPN Lab: The Container Version

Task 1: Network Setup

```
[11/16/24]seed@VM:~/Labsetup$ docksh client-10.9.0.5
root@8150e5153364:/# echo $PS1
\u@\h:\w\$
root@8150e5153364:/# export PS1="U-10.9.0.5:\w\n\$>"
U-10.9.0.5:/
$>
```

```
[11/16/24]seed@VM:~/Labsetup$ docksh server-router
root@2115fd7b6741:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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 forever preferred_lft forever
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP g
roup default
    link/ether 02:42:0a:09:00:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.11/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
14: eth1@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:c0:a8:3c:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.60.11/24 brd 192.168.60.255 scope global eth1
       valid_lft forever preferred_lft forever

root@2115fd7b6741:/# export PS1="router-10.9.0.11-192.168.60.11:\w\n\$>"
router-10.9.0.11-192.168.60.11:/
```

```
[11/16/24]seed@VM:~/Labsetup$ docksh host-192.168.60.5
root@14cdb158c9e0:/# export PS1="V-192.168.60.5:\w\n\$>"
V-192.168.60.5:/
```

Host U can communicate with VPN Server.

```
root@8150e5153364:/# export PS1="U-10.9.0.5:\w\n\$>"
U-10.9.0.5:/
$>ping 10.9.0.11 -c 2
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.532 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.097 ms

--- 10.9.0.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1009ms
rtt min/avg/max/mdev = 0.097/0.314/0.532/0.217 ms
U-10.9.0.5:/
```

VPN Server can communicate with Host V.

```
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=64 time=0.283 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.097 ms

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1007ms
rtt min/avg/max/mdev = 0.097/0.190/0.283/0.093 ms
router-10.9.0.11-192.168.60.11:/
```

Host U should not be able to communicate with Host V.

```
$>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 1007ms

U-10.9.0.5:/
```

Run tcpdump on the router, and sniff the traffic on each of the network. Show that you can capture packets.

VPN Client to VPN Server

```
$>ping 10.9.0.11 -c 2
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.116 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.116 ms

--- 10.9.0.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1020ms
rtt min/avg/max/mdev = 0.116/0.116/0.116/0.000 ms
U-10.9.0.5:/
```

```
$>tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
20:49:15.859605 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 1, length
 64
20:49:15.859632 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 1, length 6
4
20:49:16.879704 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 2, length
 64
20:49:16.879740 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 2, length 6
4
20:49:21.106388 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
20:49:21.106768 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, length 28
20:49:21.106776 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, length 28
20:49:21.106781 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
```

VPN Server to Host V

```
$>ping 192.168.60.11 -c 2
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.256 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.113 ms

--- 192.168.60.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1021ms
rtt min/avg/max/mdev = 0.113/0.184/0.256/0.071 ms
V-192.168.60.5:/
```

```
$>tcpdump -i eth1 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
20:52:51.730310 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 32, seq 1
, length 64
20:52:51.730343 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 32, seq 1,
length 64
20:52:52.751500 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 32, seq 2
, length 64
20:52:52.751535 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 32, seq 2,
length 64
20:52:56.783794 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
20:52:56.783908 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
20:52:56.783914 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
20:52:56.783920 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
```

Task 2.a: Name of the Interface

```
$>./tun.py &
[1] 22
U-10.9.0.5:/volumes
$>Interface Name: tun0

U-10.9.0.5:/volumes
$>jobs
[1]+  Running                 ./tun.py &
U-10.9.0.5:/volumes
$>ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 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 forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
12: eth0@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
U-10.9.0.5:/volumes
```

```python
#!/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
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Ahmad%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))

while True:
    time.sleep(10)
```

```
$>./tun.py &
[1] 27
U-10.9.0.5:/volumes
$>Interface Name: Ahmad0

U-10.9.0.5:/volumes
$>ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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 forever preferred_lft forever
4: Ahmad0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group de
fault qlen 500
    link/none
12: eth0@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
U-10.9.0.5:/volumes
```

Task 2.b: Set up the TUN 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
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Ahmad%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 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:
```

```
$>./tun.py &
[1] 32
U-10.9.0.5:/volumes
$>Interface Name: Ahmad0

U-10.9.0.5:/volumes
$>ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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 forever preferred_lft forever
5: Ahmad0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel sta
te UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global Ahmad0
       valid_lft forever preferred_lft forever
12: eth0@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
U-10.9.0.5:/volumes
```

There is an IP address assigned to it now.

Task 2.c: Read from the TUN 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
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Ahmad%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 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 packet:
    ip = IP(packet)
    print("{}:".format(ifname), ip.summary())
```

```
$>./tun.py &
[1] 50
U-10.9.0.5:/volumes
$>Interface Name: Ahmad0

U-10.9.0.5:/volumes
$>ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
Ahmad0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
Ahmad0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw

--- 192.168.53.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1021ms

U-10.9.0.5:/volumes
$>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 1023ms

U-10.9.0.5:/volumes
```

Why Packets Are Seen for `192.168.53.0/24`:

- The `192.168.53.0/24` network is explicitly routed to the TUN interface. This causes `tun.py` to capture and print the packets.

Why No Packets for `192.168.60.0/24`:

- The `192.168.60.0/24` network is not routed to the TUN interface. As a result, these packets bypass the TUN interface entirely.

Why 100% Packet Loss in Both Cases:

- For `192.168.53.5`, the TUN interface captures the packets but does not forward them correctly, leading to no ICMP reply.
- For `192.168.60.5`, the packets do not reach the intended host due to incorrect routing.

Task 2.d: Write to the TUN 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
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Ahmad%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 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 packet:
    pkt = IP(packet)
    print("{}:".format(ifname), pkt.summary())

    # Send out a spoof packet using the tun interface
    if ICMP in pkt and pkt[ICMP].type == 8:
        print("Original Packet.........")
        print("Source IP : ", pkt[IP].src)
        print("Destination IP :", pkt[IP].dst)

        # spoof an icmp echo reply packet
        # swap srcip and dstip
        ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
        icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
        data = pkt[Raw].load
        newpkt = ip/icmp/data

        print("Spoofed Packet.........")
        print("Source IP : ", newpkt[IP].src)
```

```
        print("Destination IP :", newpkt[IP].dst)
        os.write(tun, bytes(newpkt))
```

```
$>Interface Name: Ahmad0

U-10.9.0.5:/volumes
$>ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
Ahmad0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
Original Packet.........
Source IP :  192.168.53.99
Destination IP : 192.168.53.5
Spoofed Packet.........
Source IP :  192.168.53.5
Destination IP : 192.168.53.99
64 bytes from 192.168.53.5: icmp_seq=1 ttl=64 time=2.40 ms
Ahmad0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
Original Packet.........
Source IP :  192.168.53.99
Destination IP : 192.168.53.5
Spoofed Packet.........
Source IP :  192.168.53.5
Destination IP : 192.168.53.99
64 bytes from 192.168.53.5: icmp_seq=2 ttl=64 time=6.41 ms

--- 192.168.53.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
```

Received 2 spoofed packets

Why the Arbitrary Data Is Ignored:

- The TUN interface operates at the IP layer. Any data written to it is expected to conform to the IP protocol.
- Non-IP data, such as `b"Hello, TUN Interface!"`, does not have the correct structure (IP header) and will typically be dropped.

Network Stack Behavior:

- The kernel does not process malformed or non-standard packets at the IP layer.
- However, if the data had a valid IP header but incorrect payload, it might forward it based on routing rules.

```
$>ping 192.168.53.5 -c 1
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
Ahmad0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
Original Packet.........
Source IP :  192.168.53.99
Destination IP : 192.168.53.5
Spoofed Packet.........
Source IP :  192.168.53.5
Destination IP : 192.168.53.99
22:57:53.806417 IP 192.168.53.99 > 192.168.53.5: ICMP echo request, id 81, seq 1
, length 64
22:57:53.808762 IP truncated-ip - 27735 bytes missing! 110.116.101.114 > 102.97.
99.101: ip-proto-78

--- 192.168.53.5 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

U-10.9.0.5:/volumes
```

Task 3: Send the IP Packet to VPN Server Through a Tunnel

tun_server.py

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

from scapy.all import *

IP_A = '0.0.0.0'
PORT = 9090

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

while True:
  data, (ip,port) = sock.recvfrom(2048)
  print("{}:{} --> {}:{}".format(ip,port, IP_A, PORT))
  pkt = IP(data)
  print("  Inside: {} --> {}".format(pkt.src, pkt.dst))
```

tun_client.py

```
#!/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 UDP socket
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)

SERVER_IP, SERVER_PORT = “10.9.0.11”, 9090

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Ahmad%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 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 packet:
    # Send the packet via the tunnel
    sock.sendto(packet, (SERVER_IP, SERVER_PORT))
```

```
U-10.9.0.5:/volumes
$>ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.

--- 192.168.53.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1025ms
```

```
$>chmod a+x tun_server.py
 router-10.9.0.11-192.168.60.11:/volumes
$>./tun_server.py
10.9.0.5:34416 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:34416 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.53.5
```

Encapsulation by Client:

- The client intercepts packets destined for the TUN interface and encapsulates them in a transport protocol (e.g., UDP).
- These encapsulated packets are sent to the VPN Server.

Decapsulation by Server:

- The server receives the encapsulated packets, extracts the original packet (e.g., ICMP echo request), and processes it.
- It forwards the decapsulated packet into the `192.168.53.0/24` network.

Forwarding and Reply:

- The local network host in `192.168.53.0/24` replies to the ICMP echo request.
- The server intercepts the reply, encapsulates it, and sends it back to the client.

End-to-End Communication:

- The client decapsulates the ICMP echo reply and delivers it to the original application (ping) on Host U.

Access the hosts inside the private network 192.168.60.0/24 using the tunnel.

```
$>ip route add 192.168.60.0/24 dev Ahmad0
U-10.9.0.5:/volumes
$>ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.53.0/24 dev Ahmad0 proto kernel scope link src 192.168.53.99
192.168.60.0/24 dev Ahmad0 scope link
```

```
$>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 1024ms
```

```
10.9.0.5:34416 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:34416 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
```

Task 4: Set Up the VPN Server

tun_server.py

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

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

#tun Interface
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'Ahmad%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 interface
os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

#UDP Server
IP_A = '0.0.0.0'
PORT = 9090

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

while True:
  data, (ip,port) = sock.recvfrom(2048)
  print("{}:{} --> {}:{}".format(ip,port, IP_A, PORT))
  pkt = IP(data)
  print("  Inside: {} --> {}".format(pkt.src, pkt.dst))
  os.write(tun, bytes(pkt))
```

tun_client.py

```
#!/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 UDP socket
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)

SERVER_IP, SERVER_PORT = "10.9.0.11", 9090

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Ahmad%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 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:
  # Get a packet from the tun interface
  packet = os.read(tun, 2048)
  if packet:
    # Send the packet via the tunnel
    sock.sendto(packet, (SERVER_IP, SERVER_PORT))
```

```
U-10.9.0.5:/volumes
$>Interface Name: Ahmad0
$>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 1012ms
```

```
$>jobs
 router-10.9.0.11-192.168.60.11:/volumes
$>./tun_server.py
Interface Name: Ahmad0
10.9.0.5:56193 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56193 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
```

```
$>tcpdump -i eth0 -n 2>/dev/null
22:47:07.783008 IP6 fe80::42:ccff:fece:691b > ff02::2: ICMP6, router solicitatio
n, length 16
22:54:57.589059 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
22:54:57.589096 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
22:54:57.589116 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 51, seq 1
, length 64
22:54:57.589129 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 51, seq 1,
length 64
22:54:58.601449 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 51, seq 2
, length 64
22:54:58.601495 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 51, seq 2,
length 64
22:55:02.666595 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
22:55:02.666745 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
```

Task 5: Handling Traffic in Both Directions

tun_server.py

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

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

#tun Interface
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'Ahmad%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 interface
os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))
os.system("ip link set dev {} up".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))

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)
      print("From UDP {}:{} --> {}:{}".format(ip,port,IP_A,PORT))
      pkt = IP(data)
      print("From socket(IP) ==>: {} --> {}".format(pkt.src, pkt.dst))
      os.write(tun, bytes(pkt))
    if fd is tun:
      packet = os.read(tun,2048)
      pkt = IP(packet)
      print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
      sock.sendto(packet, (ip,port))
```

tun_client.py

```
#!/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 UDP socket
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)

SERVER_IP, SERVER_PORT = "10.9.0.11", 9090

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Ahmad%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 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)
      print("From UDP {}:{} --> {}".format(ip,port,"10.9.0.5"))
      pkt = IP(data)
      print("From socket(IP) ==>: {} --> {}".format(pkt.src, pkt.dst))
      os.write(tun, bytes(pkt))
    if fd is tun:
      packet = os.read(tun,2048)
      pkt = IP(packet)
      print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
      # Send the packet via the tunnel
      sock.sendto(packet, (SERVER_IP, SERVER_PORT))
```

```
$>ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=6.39 ms
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=8.79 ms

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 6.391/7.592/8.794/1.201 ms
U-10.9.0.5:/volumes
```

```
 router-10.9.0.11-192.168.60.11:/volumes
$>./tun_server.py
Interface Name: Ahmad0
From UDP 10.9.0.5:60498 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:60498 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
```

```
23:35:11.642208 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 77, seq 1
, length 64
23:35:11.642250 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 77, seq 1,
length 64
23:35:12.648141 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 77, seq 2
, length 64
23:35:12.648176 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 77, seq 2,
length 64
23:35:16.743529 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
23:35:16.743651 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
23:35:16.743656 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
23:35:16.743673 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
```

12	2024-11-17 18:3…	172.217.165.3	10.0.2.15	TCP	60 443 → 51794 [ACK] Seq-
13	2024-11-17 18:3…	10.0.2.15	192.168.48.1	DNS	86 Standard query 0x628b
14	2024-11-17 18:3…	192.168.48.1	10.0.2.15	DNS	102 Standard query respons
15	2024-11-17 18:3…	172.217.165.3	10.0.2.15	TLSv1.2	271 Application Data, Appl

Telnet

```
U-10.9.0.5:/volumes
$>telnet 192.168.60.5
Trying 192.168.60.5...
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
Connected to 192.168.60.5.
Escape character is '^]'.
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
```

```
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
                                            From UDP 10.9.0.11:9090 --> 10.9.0.5
                                                                                   From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
                                                  Ubuntu 20.04.1 LTS
From tun ==>: 192.168.53.99 --> 192.168.60.5
                                            From UDP 10.9.0.11:9090 --> 10.9.0.5
```

```
rom socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
                                                    Ubuntu 20.04.1 LTS
From tun ==>: 192.168.53.99 --> 192.168.60.5
                                                From UDP 10.9.0.11:9090 --> 10.9.0.5
                                                                                    F
rom socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
                                                    5c3701942b61 login: From tun =
=>: 192.168.53.99 --> 192.168.60.5
                                      From UDP 10.9.0.11:9090 --> 10.9.0.5
                                                                          From socke
t(IP) ==>: 192.168.60.5 --> 192.168.53.99

Login timed out after 60 seconds.
From tun ==>: 192.168.53.99 --> 192.168.60.5
                                                From UDP 10.9.0.11:9090 --> 10.9.0.5
                                                                                    F
rom socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
                                                    Connection closed by foreign h
ost.
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket(IP) ==>: 192.168.60.5 --> 192.168.53.99
l-10.9.0.5:/volumes
```

	Time	Source	Destination	Protocol	Length	Info
28	2024-11-18 15:0…	91.189.91.49	10.0.2.15	TCP	60	80 → 38060 [SYN, ACK] S
29	2024-11-18 15:0…	10.0.2.15	91.189.91.49	TCP	54	38060 → 80 [ACK] Seq=39
30	2024-11-18 15:0…	10.0.2.15	91.189.91.49	HTTP	141	GET / HTTP/1.1
31	2024-11-18 15:0…	91.189.91.49	10.0.2.15	TCP	60	80 → 38060 [ACK] Seq=14
32	2024-11-18 15:0…	91.189.91.49	10.0.2.15	HTTP	243	HTTP/1.1 204 No Content
33	2024-11-18 15:0…	10.0.2.15	91.189.91.49	TCP	54	38060 → 80 [ACK] Seq=39
34	2024-11-18 15:0…	91.189.91.49	10.0.2.15	TCP	60	80 → 38060 [FIN, ACK] S
35	2024-11-18 15:0…	10.0.2.15	91.189.91.49	TCP	54	38060 → 80 [FIN, ACK] S
36	2024-11-18 15:0…	91.189.91.49	10.0.2.15	TCP	60	80 → 38060 [ACK] Seq=14

Server side recieving

```
$>./tun_server.py
Interface Name: Ahmad0
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
```

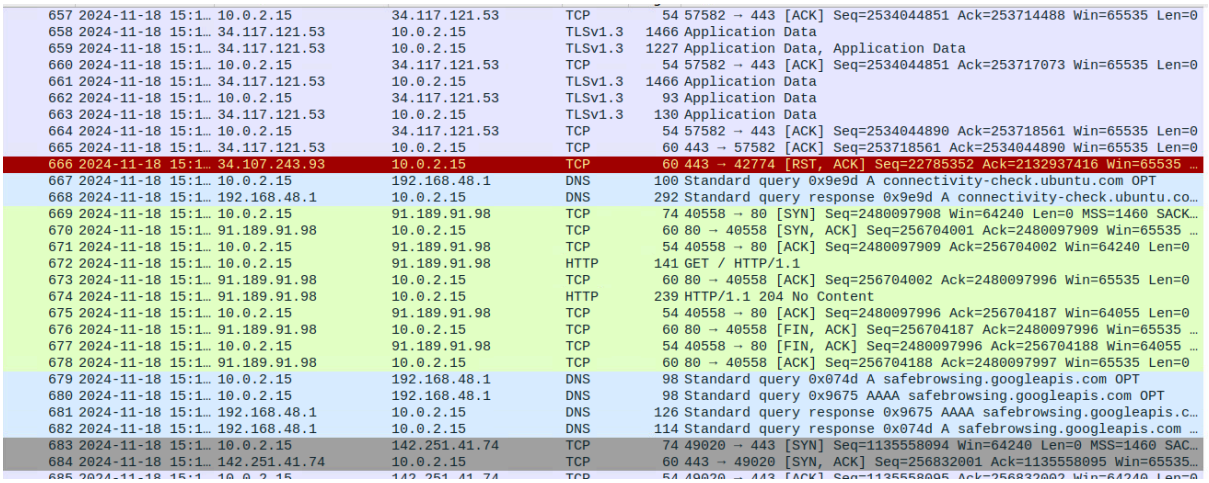
```
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:59440 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
```

```
$>./tun_client.py &>/dev/null &
[1] 36
U-10.9.0.5:/volumes
$>telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
5c3701942b61 login: seed
Password:
Welcome to Ubuntu 20.04.1 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.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

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

Task 6: Tunnel-Breaking Experiment

```
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
5c3701942b61 login: seed
Password:
Welcome to Ubuntu 20.04.1 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.

To restore this content, you can run the 'unminimize' command.
Last login: Mon Nov 18 20:07:34 UTC 2024 on pts/2
seed@5c3701942b61:~$ date
Mon Nov 18 20:14:07 UTC 2024
seed@5c3701942b61:~$ date
Mon Nov 18 20:15:02 UTC 2024
seed@5c3701942b61:~$ date
Mon Nov 18 20:15:08 UTC 2024
seed@5c3701942b61:~$
```

```
From UDP 10.9.0.5:40038 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:40038 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:40038 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
^CTraceback (most recent call last):
  File "./tun_server.py", line 39, in <module>
    ready,_,_ = select.select([sock,tun],[],[])
KeyboardInterrupt

 router-10.9.0.11-192.168.60.11:/volumes
$>./tun_server.py
Interface Name: Ahmad0
From UDP 10.9.0.5:40038 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
```



657	2024-11-18 15:1…	10.0.2.15	34.117.121.53	TCP	54	57582 → 443 [ACK] Seq=2534044851 Ack=253714488 Win=65535 Len=0
658	2024-11-18 15:1…	34.117.121.53	10.0.2.15	TLSv1.3	1466	Application Data
659	2024-11-18 15:1…	34.117.121.53	10.0.2.15	TLSv1.3	1227	Application Data, Application Data
660	2024-11-18 15:1…	10.0.2.15	34.117.121.53	TCP	54	57582 → 443 [ACK] Seq=2534044851 Ack=253717073 Win=65535 Len=0
661	2024-11-18 15:1…	34.117.121.53	10.0.2.15	TLSv1.3	1466	Application Data
662	2024-11-18 15:1…	10.0.2.15	34.117.121.53	TLSv1.3	93	Application Data
663	2024-11-18 15:1…	34.117.121.53	10.0.2.15	TLSv1.3	130	Application Data
664	2024-11-18 15:1…	10.0.2.15	34.117.121.53	TCP	54	57582 → 443 [ACK] Seq=2534044890 Ack=253718561 Win=65535 Len=0
665	2024-11-18 15:1…	34.117.121.53	10.0.2.15	TCP	60	443 → 57582 [ACK] Seq=253718561 Ack=2534044890 Win=65535 Len=0
666	2024-11-18 15:1…	34.107.243.93	10.0.2.15	TCP	60	443 → 42774 [RST, ACK] Seq=22785352 Ack=2132937416 Win=65535 …
667	2024-11-18 15:1…	10.0.2.15	192.168.48.1	DNS	100	Standard query 0x9e9d A connectivity-check.ubuntu.com OPT
668	2024-11-18 15:1…	192.168.48.1	10.0.2.15	DNS	292	Standard query response 0x9e9d A connectivity-check.ubuntu.co…
669	2024-11-18 15:1…	10.0.2.15	91.189.91.98	TCP	74	40558 → 80 [SYN] Seq=2480097908 Win=64240 Len=0 MSS=1460 SACK…
670	2024-11-18 15:1…	91.189.91.98	10.0.2.15	TCP	60	80 → 40558 [SYN, ACK] Seq=256704001 Ack=2480097909 Win=65535 …
671	2024-11-18 15:1…	10.0.2.15	91.189.91.98	TCP	54	40558 → 80 [ACK] Seq=2480097909 Ack=256704002 Win=64240 Len=0
672	2024-11-18 15:1…	10.0.2.15	91.189.91.98	HTTP	141	GET / HTTP/1.1
673	2024-11-18 15:1…	91.189.91.98	10.0.2.15	TCP	60	80 → 40558 [ACK] Seq=256704002 Ack=2480097996 Win=65535 Len=0
674	2024-11-18 15:1…	91.189.91.98	10.0.2.15	HTTP	239	HTTP/1.1 204 No Content
675	2024-11-18 15:1…	10.0.2.15	91.189.91.98	TCP	54	40558 → 80 [ACK] Seq=2480097996 Ack=256704187 Win=64055 Len=0
676	2024-11-18 15:1…	91.189.91.98	10.0.2.15	TCP	60	80 → 40558 [FIN, ACK] Seq=256704187 Ack=2480097996 Win=65535 …
677	2024-11-18 15:1…	10.0.2.15	91.189.91.98	TCP	54	40558 → 80 [FIN, ACK] Seq=2480097996 Ack=256704188 Win=64055 …
678	2024-11-18 15:1…	91.189.91.98	10.0.2.15	TCP	60	80 → 40558 [ACK] Seq=256704188 Ack=2480097997 Win=65535 Len=0
679	2024-11-18 15:1…	10.0.2.15	192.168.48.1	DNS	98	Standard query 0x074d A safebrowsing.googleapis.com OPT
680	2024-11-18 15:1…	10.0.2.15	192.168.48.1	DNS	98	Standard query 0x9675 AAAA safebrowsing.googleapis.com OPT
681	2024-11-18 15:1…	192.168.48.1	10.0.2.15	DNS	126	Standard query response 0x9675 AAAA safebrowsing.googleapis.c…
682	2024-11-18 15:1…	192.168.48.1	10.0.2.15	DNS	114	Standard query response 0x074d A safebrowsing.googleapis.com …
683	2024-11-18 15:1…	10.0.2.15	142.251.41.74	TCP	74	49020 → 443 [SYN] Seq=1135558094 Win=64240 Len=0 MSS=1460 SAC…
684	2024-11-18 15:1…	142.251.41.74	10.0.2.15	TCP	60	443 → 49020 [SYN, ACK] Seq=256832001 Ack=1135558095 Win=65535…

Action	Observation
Disconnecting the VPN tunnel	Telnet session hangs. Typing produces no visible output.
Typing during tunnel downtime	Data is not sent or acknowledged. Nothing appears in the Telnet window.
Reconnecting the VPN before timeout	Telnet session resumes. Any typed data during downtime is eventually sent.
Reconnecting after timeout	Telnet session is terminated. A new connection must be established.

Task 7: Routing Experiment on Host V

```
[11/19/24]seed@VM:~/Labsetup$ docksh host-192.168.60.5
root@8bbac9c00d95:/# 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@8bbac9c00d95:/# ip route del default
root@8bbac9c00d95:/# ip route
192.168.60.0/24 dev eth0 proto kernel scope link src 192.168.60.5
root@8bbac9c00d95:/# ip route add 192.168.53.0/24 via 192.168.60.11
root@8bbac9c00d95:/# 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
```

Task 8: VPN Between Private Networks

```
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.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=62 time=5.24 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=62 time=3.65 ms

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1017ms
rtt min/avg/max/mdev = 3.647/4.444/5.242/0.797 ms
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.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=62 time=6.85 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=62 time=13.1 ms

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 6.851/9.966/13.082/3.115 ms
U-192.168.50.5/
```

```
Client-192.168.50.12-10.9.0.12:/volumes
$> ./tun_client.py
Interface Name: Ahmad0
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
```

```
Server-192.168.60.11-10.9.0.11:/volumes
$> ./tun_server.py
Interface Name: Ahmad0
From UDP 10.9.0.12:58530 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.50.5 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.50.5
From UDP 10.9.0.12:58530 --> 0.0.0.0:9090
From socket(IP) ==>: 192.168.50.5 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.50.5
```

```
$> tcpdump -i eth0 -n 2>/dev/null
22:07:44.470576 IP 192.168.50.5 > 192.168.60.5: ICMP echo request, id 34, seq 1,
 length 64
22:07:44.470644 IP 192.168.60.5 > 192.168.50.5: ICMP echo reply, id 34, seq 1, l
ength 64
22:07:45.472229 IP 192.168.50.5 > 192.168.60.5: ICMP echo request, id 34, seq 2,
 length 64
22:07:45.472280 IP 192.168.60.5 > 192.168.50.5: ICMP echo reply, id 34, seq 2, l
ength 64
22:07:49.579539 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
22:07:49.579659 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
22:07:49.579665 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
22:07:49.579685 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
```

```
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
614cd24572c1 login: seed
Password:
Welcome to Ubuntu 20.04.1 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.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

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

seed@614cd24572c1:~$
```

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

seed@614cd24572c1:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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 forever preferred_lft forever
21: eth0@if22: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:c0:a8:3c: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
```

Task 9: Experiment with the TAP Interface

tap.py

```
#!/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'Ahmad%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 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())
```

```
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 1036ms
pipe 2
```

ARP Requests:

- The output shows multiple ARPs `who have` messages. This indicates that your machine is broadcasting ARP requests to resolve the IP address `192.168.53.5` into its corresponding MAC address.
- However, there is no response from `192.168.53.5`, meaning that the target host is not reachable at the network layer.

ICMP Responses:

- The response to the ICMP `ping` request from the intermediate router (`192.168.53.99`) indicates "Destination Host Unreachable".
- This shows that the packet reached a router or gateway, but the intended target (`192.168.53.5`) could not be reached.

```
Client-192.168.50.12-10.9.0.12:/volumes
$> arping -I Ahmad0 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
$> arping -I Ahmad0 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

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

Client-192.168.50.12-10.9.0.12:/volumes
```

tap.py

```
#!/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'Ahmad%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 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:
    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(tap, bytes(newpkt))
```

```
Client-192.168.50.12-10.9.0.12:/volumes
$> Interface Name: Ahmad0

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

--- 192.168.53.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1004ms
```

```
Client-192.168.50.12-10.9.0.12:/volumes
$> arping -I Ahmad0 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=2.971 msec
------------------------------
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=1 time=7.876 msec

--- 192.168.53.33 statistics ---
2 packets transmitted, 2 packets received,   0% unanswered (0 extra)
rtt min/avg/max/std-dev = 2.971/5.423/7.876/2.452 ms
Client-192.168.50.12-10.9.0.12:/volumes
```

```
Client-192.168.50.12-10.9.0.12:/volumes
$> arping -I Ahmad0 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=7.127 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=1 time=4.286 msec

--- 1.2.3.4 statistics ---
2 packets transmitted, 2 packets received,   0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.007/2.147/4.286/2.139 ms
Client-192.168.50.12-10.9.0.12:/volumes
```