Lab7-report

57117214 吴国铨

**Task1: Network Setup**

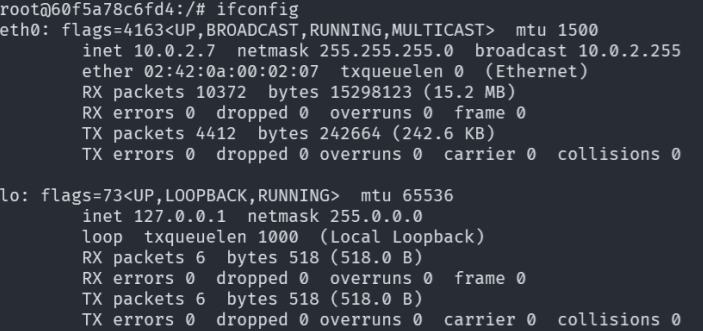
首先，创建两个Docker的bridge类型网络internet与intranet：

docker network create --subnet=10.0.2.0/24 internet

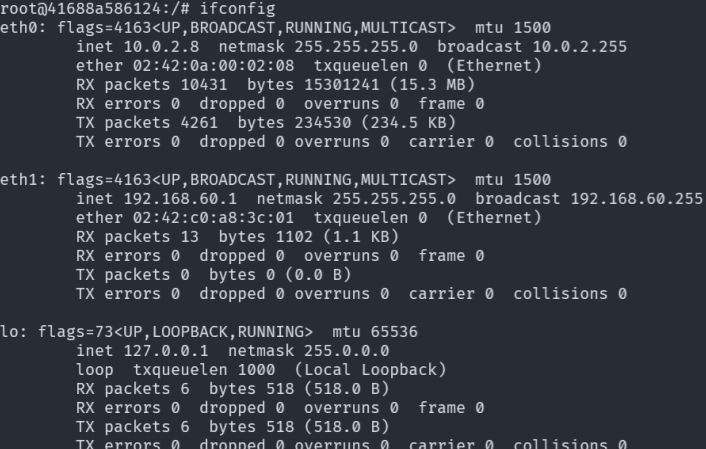
docker network create --subnet=192.168.60.0/24 intranet

然后创建三个容器，具体如下：

容器A，作为主机：

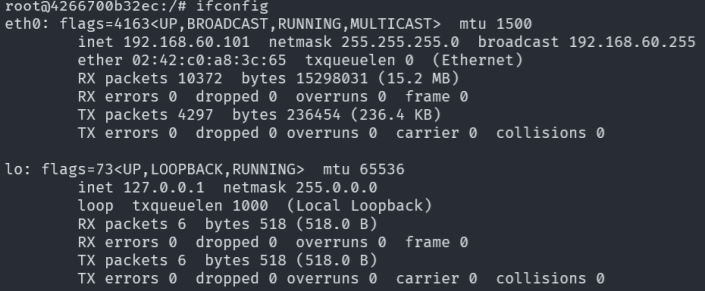


容器B，作为VPN服务器：



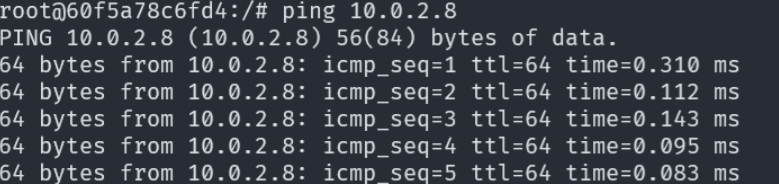
其中192.168.60.1连接intranet，10.0.2.8连接internet

容器C作为主机：



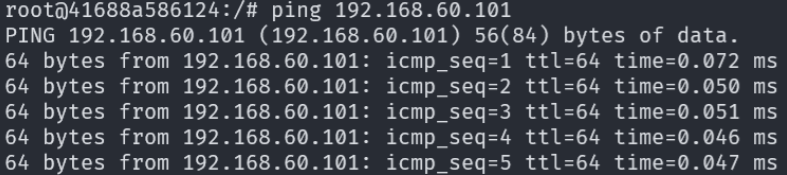
下面进行测试。

使用主机A连接VPN服务器B：



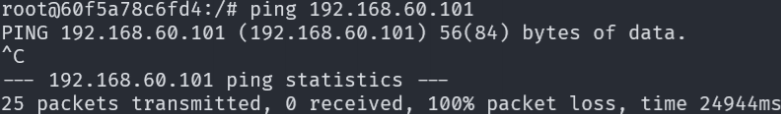
成功连接。

使用VPN服务器B连接主机C：



成功连接

使用主机A连接主机C：

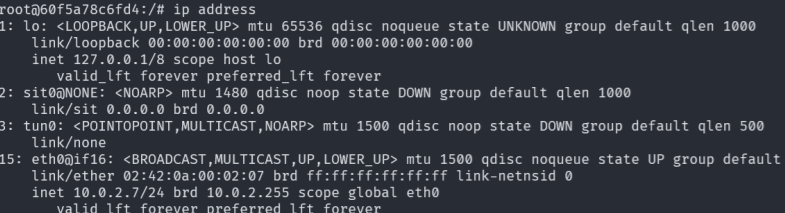


连接失败。

**Task 2: Cerate and Configure TUN Interface**

**Task 2.a: Name of the Interface**

首先运行题目所给代码，然后使用ip address查看所有网口信息：



成功注册tun0网口。

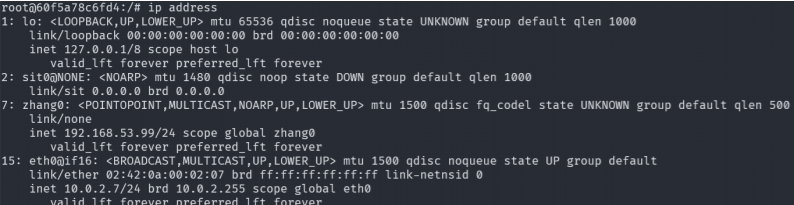
为了将tun0网口改为\*（任意想要的名称）0，我们只需要将代码中初始化ifr的代码进行更改，如下：

ifr = struct.pack('16sH', b'\*%d', IFF\_TUN | IFF\_NO\_PI)

运行后即可得到结果。

**Task 2.b: Set up the TUN Interface**

按照题目运行代码，结果如下：

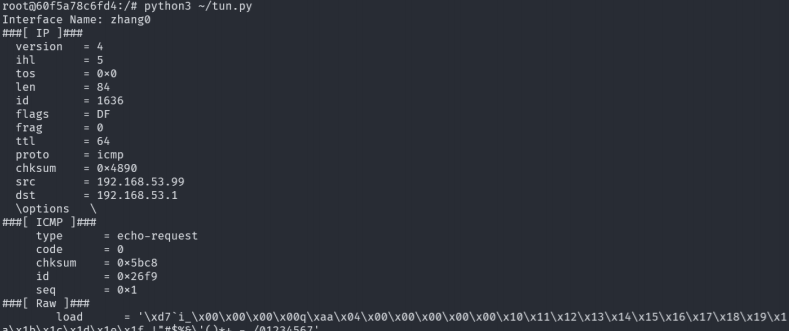


**Task 2.c: Read from the TUN interface**

按照题目要求修改代码并运行，在主机A的另一个shell中：

ping 192.168.53.1

输出为：



确实能看到ICMP报文。

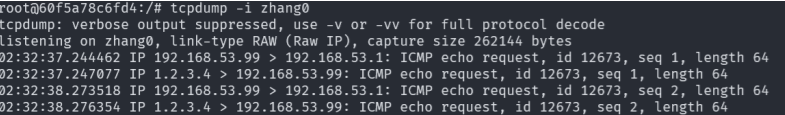
但是如果ping 192.168.60.0，却看不到输出，因为ping如果找不到就在该子网的网口，就会找第一个网口发送命令。

**Task 2.d: Write to the TUN Interface**

按照题目要求修改代码，然后：

Ping 192.168.53.1

使用tcpdump查看：



确实看到了伪造的报文写入了tun网口。

但是如果不写IP报文，将代码修改成：

os.write(tun,b’one’)

运行同样的命令，在tcpdump中只能看到：

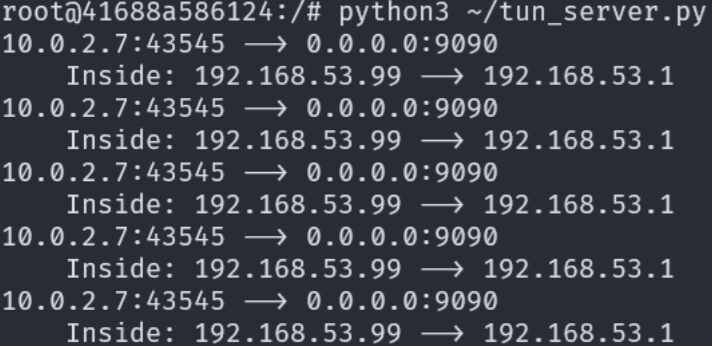


被识别错误。

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

根据题目要求编写代码并运行，在主机A中ping 192.168.53.1

在VPN服务器B中查看：



如果直接在主机A中ping192.168.60.101，VPN服务器B是不会收到报文的，为了解决这个问题，需要在主机A中添加静态路由：  
route add -net 192.168.60.0/24 one

也就是把所有发向192.168.60.0/24的报文由yun网口one0发出。

修改后成功接收到ICMP请求。

**Task 4: Set Up the VPN Server**

修改tun\_serber.py如下：

import fcntl

import struct

import os

import time

from scapy.all import \*

TUNSETIFF = 0x400454ca

IFF\_TUN = 0x0001

IFF\_TAP = 0x0002

IFF\_NO\_PI = 0x1000

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)

ifname = ifname\_bytes.decode('UTF-8')[:16].strip("\x00")

print("Interface Name: {}".format(ifname))

os.system("ip addr add 192.168.53.98/24 dev {}".format(ifname))

os.system("ip link set dev {} up".format(ifname))

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)

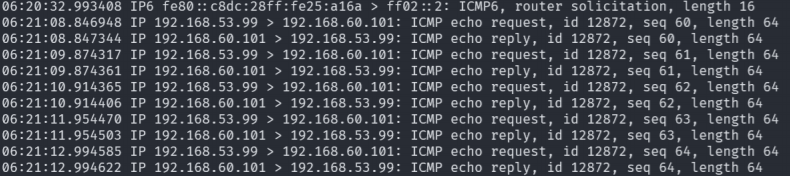
print("{}:{} --> {}:{}".format(ip, port, IP\_A, PORT))

pkt = IP(data)

print(" Inside: {} --> {}".format(pkt.src, pkt.dst))

print("Sending raw: {}".format(data))

os.write(tun, data)

运行后，在主机A中ping主机B，如下：  


成功接收。

**Task 5: Handling Traffic in Both Directions**

修改tun\_client.py:

import fcntl

import struct

import os

import time

import select

from scapy.all import \*

TUNSETIFF = 0x400454ca

IFF\_TUN = 0x0001

IFF\_TAP = 0x0002

IFF\_NO\_PI = 0x1000

SERVER\_IP = "10.0.2.8"

SERVER\_PORT = 9090

tun = os.open("/dev/net/tun", os.O\_RDWR)

ifr = struct.pack('16sH', b'zhang%d', IFF\_TUN | IFF\_NO\_PI)

ifname\_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

ifname = ifname\_bytes.decode('UTF-8')[:16].strip("\x00")

print("Interface Name: {}".format(ifname))

os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))

os.system("ip link set dev {} up".format(ifname))

os.system("route add -net 192.168.60.0/24 {}".format(ifname))

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

while True:

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))

os.write(tun, data)

if fd is tun:

packet = os.read(tun, 2048)

pkt = IP(packet)

print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))

sock.sendto(packet, (SERVER\_IP, SERVER\_PORT))

修改tum\_server.py:

import fcntl

import struct

import os

import time

import select

from scapy.all import \*

TUNSETIFF = 0x400454ca

IFF\_TUN = 0x0001

IFF\_TAP = 0x0002

IFF\_NO\_PI = 0x1000

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)

ifname = ifname\_bytes.decode('UTF-8')[:16].strip("\x00")

print("Interface Name: {}".format(ifname))

os.system("ip addr add 192.168.53.98/24 dev {}".format(ifname))

os.system("ip link set dev {} up".format(ifname))

IP\_A = "0.0.0.0"

PORT = 9090

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

sock.bind((IP\_A, PORT))

port

while True:

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))

os.write(tun, data)

if fd is tun:

packet = os.read(tun, 2048)

pkt = IP(packet)

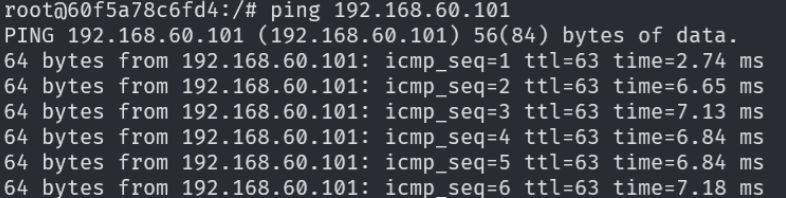
print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))

sock.sendto(packet, ("10.0.2.7", port))

然后再主机A中添加路由表项：

ip route add 192.168.53.0/24 gw 192.168.60.1 eth0

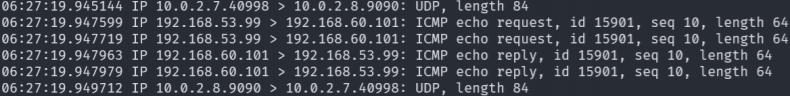
设置完成后再次由主机Aping主机C：



此时主机的Atcpdump：



VPN服务器B上的tcpdump：



**Task 6: Tunnel-Breaking Experiment**

在主机A与主机C建立Telnet连接之后，关闭tun\_client.py，发现无论输入什么，主机A的telnet界面都没有显示，但是此时的TCP连接并没有终止。再次云心tun\_client.py，随意输入几个字符，一段时间之后发现输入的字符重新出现在了telnet界面中。

**Task 7: Routing Experiment on Host V**

因为前面实验直接是在主机B上连接，所以没有使用默认路由，相当于已经完成。

**Task 8: Experiment with TUN IP Address**

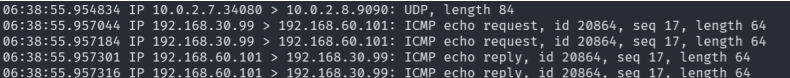
将主机A上的tun网口IP地址改成192.168.30.99后，无法ping通。

首先查看主机A上的tcpdump：

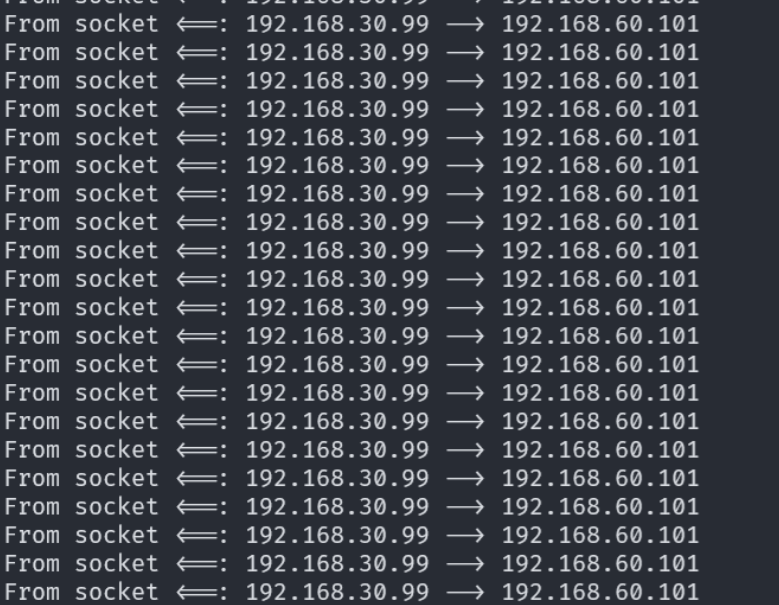


发现只将其封装的UDP发送出去了，并没有返回的UDP，

然后再VPN服务器B上查看tcpdump：

收到了UDP报文，并发送给了主机C，也受到了主机C的ICMP响应报文，但是并没有发送返回UDP

查看tun\_server.py



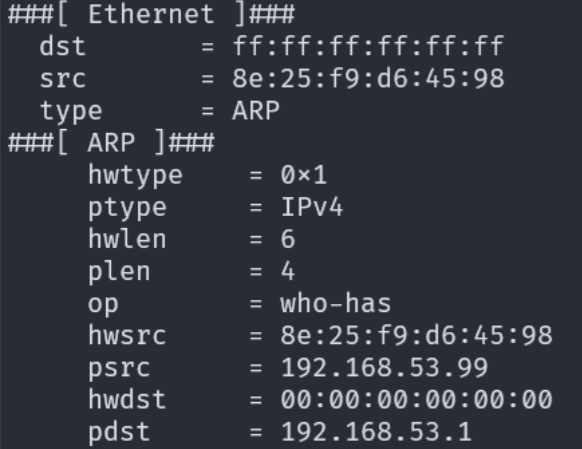
确实没有从tun口收到包

说明返回UDP阶段丢包了

**Task 9: Experiment with the TAP Interface**

按照题目要求编写代码并运行，并再主机A上ping 192.168.53.1

然后查看输出：



发现TAP口有一个ARP请求。

这是因为ping命令会首先根据192.168.53.1发送一个ARP请求，查看是否再局域网内有这个IP设备，而我们的TAP口的IP与其再同一个子网内，所以命令就转发到了TAP口中，从而我们可以查看到这事一个ARP的who-has请求，查看哪个设备拥有192.168.53.1