**网络安全综合课程设计实验7**

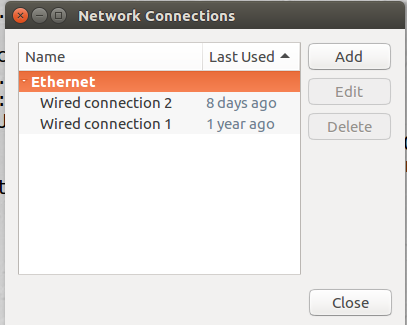
57117204 姚晓

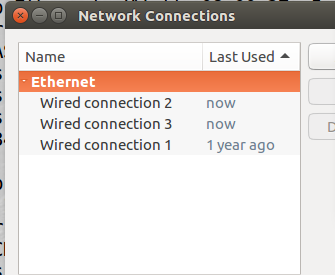
2020/9/18

**VPN Tunneling Lab**

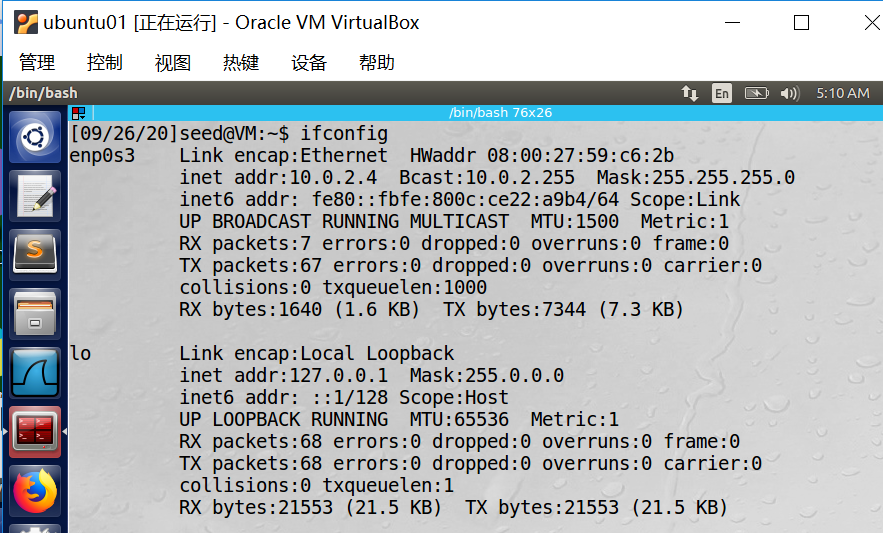
**Task1: Network Setup**

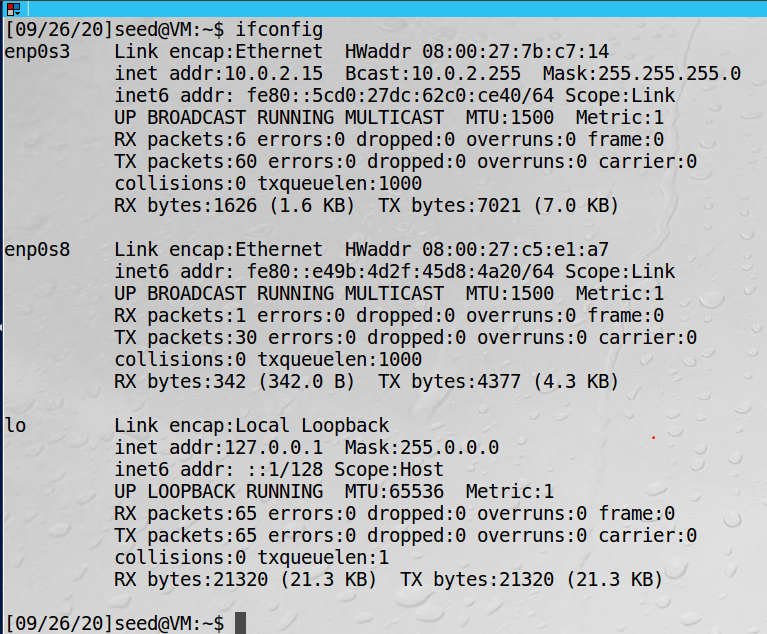
首先按照实验手册上的说明配置实验环境，将Ubuntu01作为Host U，Ubuntu02作为VPN server，Ubuntu03作为Host V，VPNserver和Host V的网卡设置为internal，并设置IP地址：

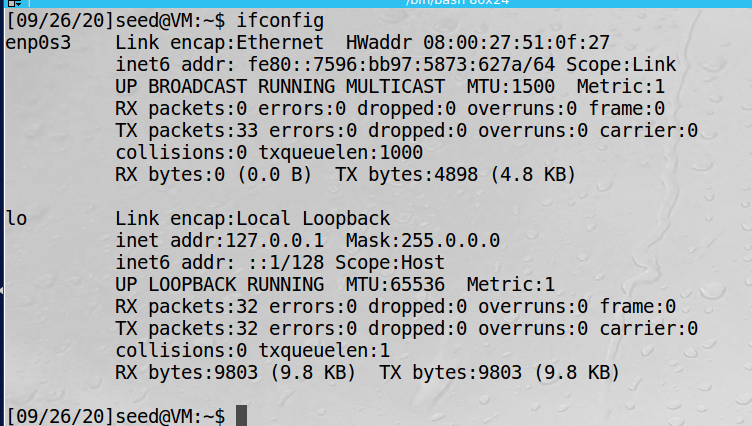




各个主机的IP地址MAC地址信息：



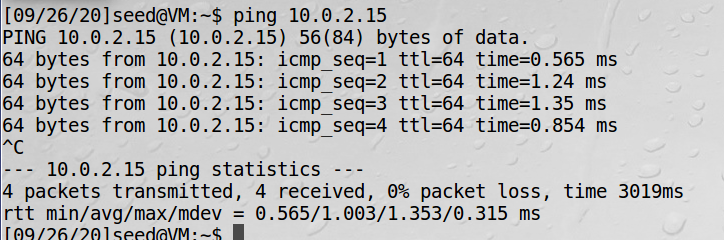




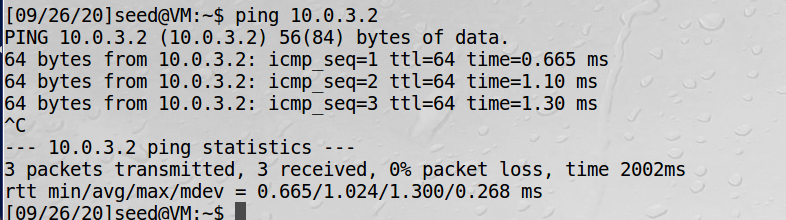
为了方便用表格形式展现：

|  |  |  |
| --- | --- | --- |
| **Host U** | **IP地址** | **MAC地址** |
| 10.0.2.4 | 08:00:27:59:c6:2b |
| **VPN　server** | **IP地址** | **MAC地址** |
| 10.0.2.15 | 08:00:27:7b:c7:14 |
| 10.0.3.3 | 08:00:27:C5:E1:A7 |
| **Host　V** | **IP地址** | **MAC地址** |
| 10.0.3.2 | 08:00:27:51:0F:27 |

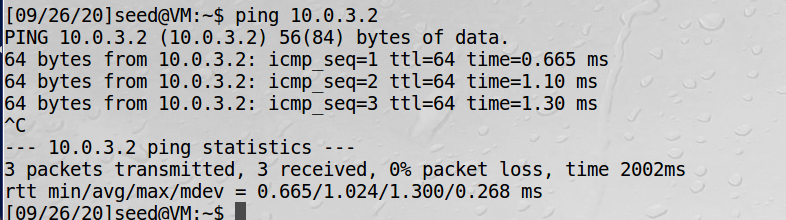
配置好环境之后，用Host U　ping　VPN　server，可以ping通：



用VPN　server　ping　Host　V，可以ping通：



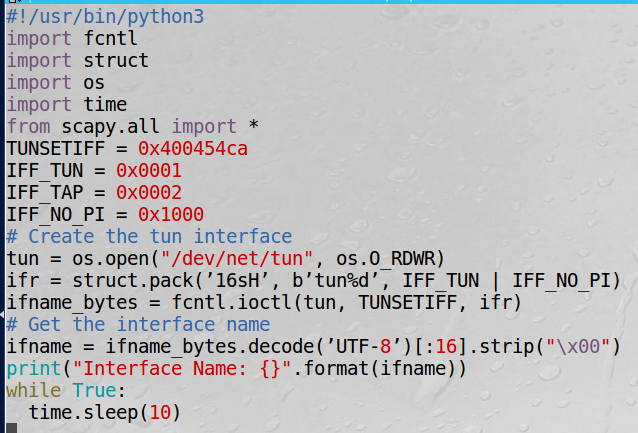
用Host　U　ping　Host　V，不能ping通：

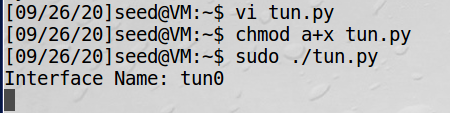


**Task2: Create and Configure TUN Interface**

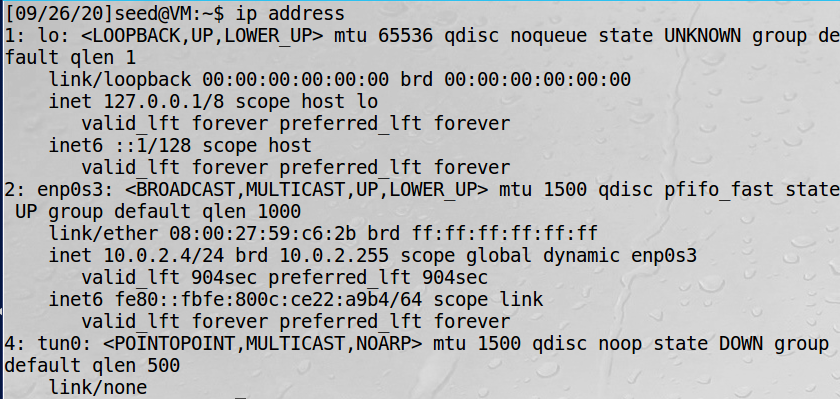
Task2.a: Name of the Interface

　　编译并运行实验手册的代码：





然后用Ip address命令查看，可以看到有虚拟端口ｔｕｎ０：

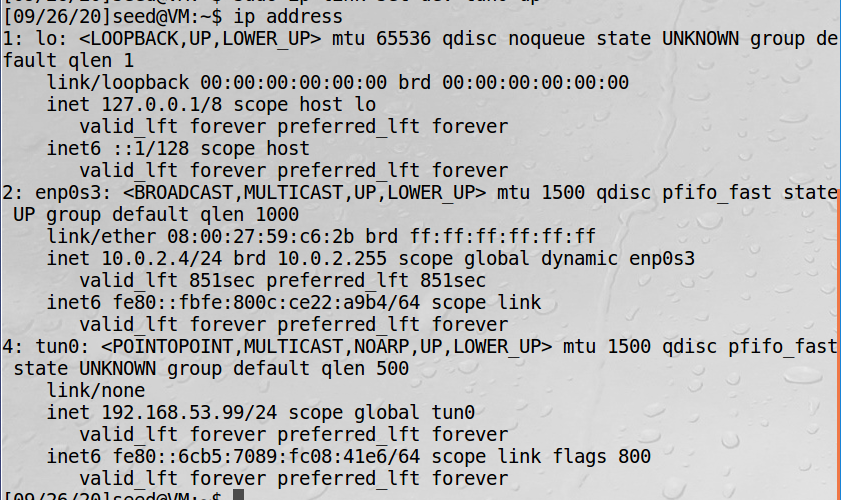


Task2.b: Setup the TUN Interface

　　用下面的命令为该端口分配IP地址，然后启动端口：



　　可以看到接口配置和开启成功：

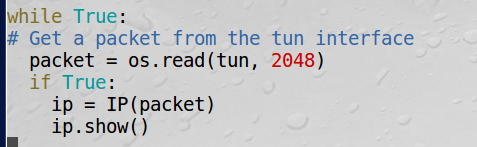


　　把修改代码，把如下语句加入代码中，每次运行代码的时候就可以自动配置了：

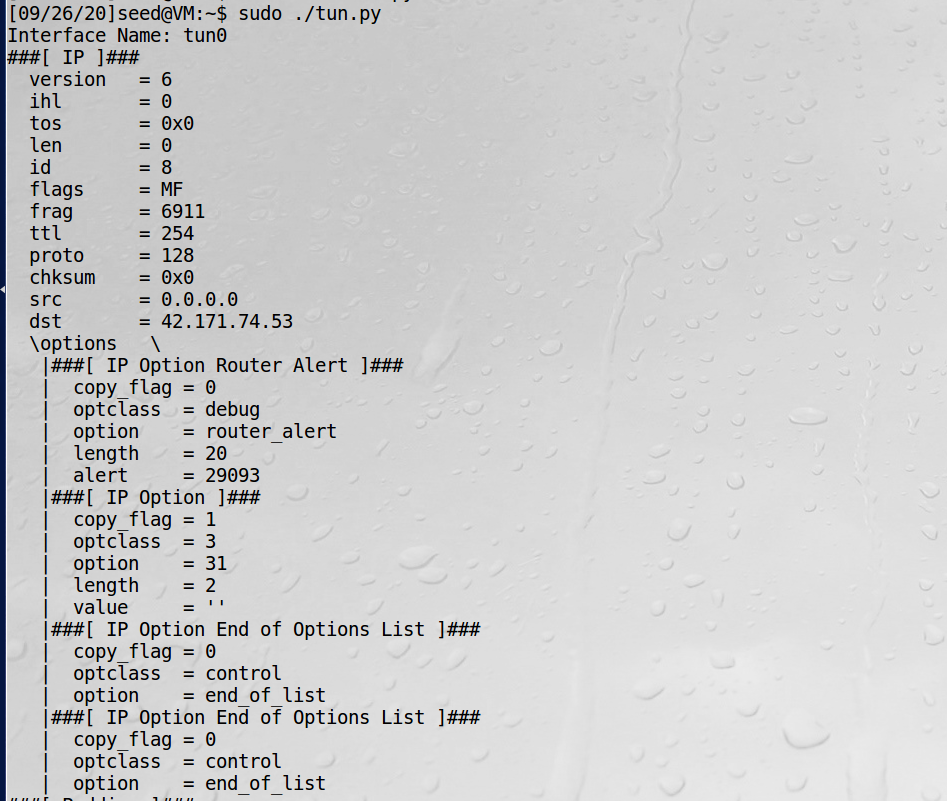


Task2.c: Read from the TUN Interface

修改代码：



　　运行代码：



Ping　１９２.１６８.５３.１２可以看到打印出发往192.168.53.１2的数据包发送给虚拟接口，源地址被修改为ｔｕｎ０端口的IP地址，而非１０.０.２.４：



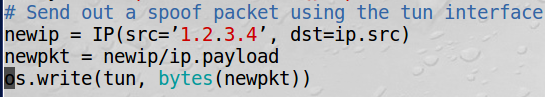


Ping 10.0.3.2没有信息产生



Task2.d: Write to the TUN Interface

　　修改代码：

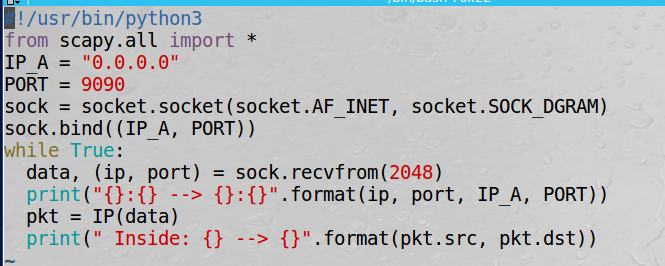


　　发现有不同方向的报文出现，TUN运行成功，随意写入数据而非IP数据报发现会报错

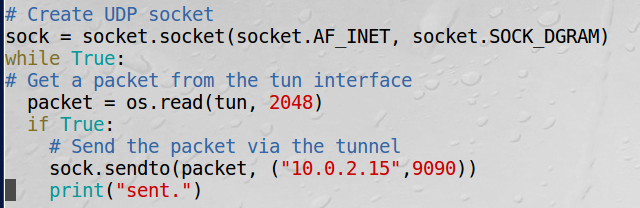
。

**Task3: Send the IP Packet to VPN Server Through a Tunnel**

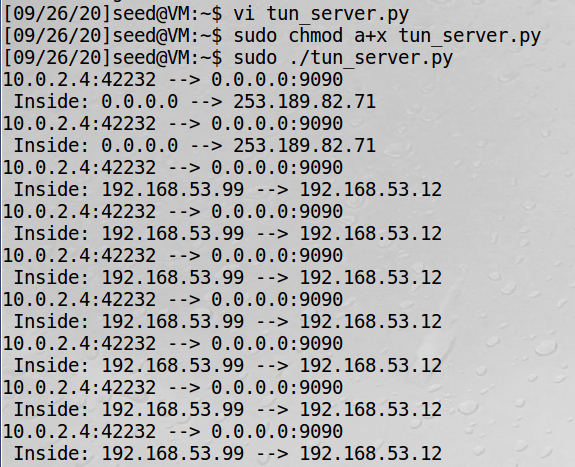
　　在VPN server里编译并运行tun\_server.py:



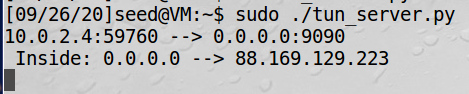
　　在Host U里编译并运行tun\_client.py：



　　双方同时运行程序，并ping 192.168.53.12，得到外层由10.0.2.4:42232发往自己的IP数据报，内部为192.168.53.99发往192.168.53.12的数据报。由此通过隧道在Host U和VPN Server间实现了通信：



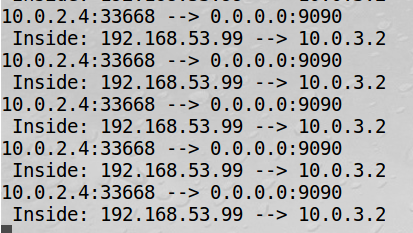
　　而ping 10.0.3.2没有响应：



　　配置Host U的路由信息，将192.168.22.0/24的出口设置为tun，让其走tun接口：

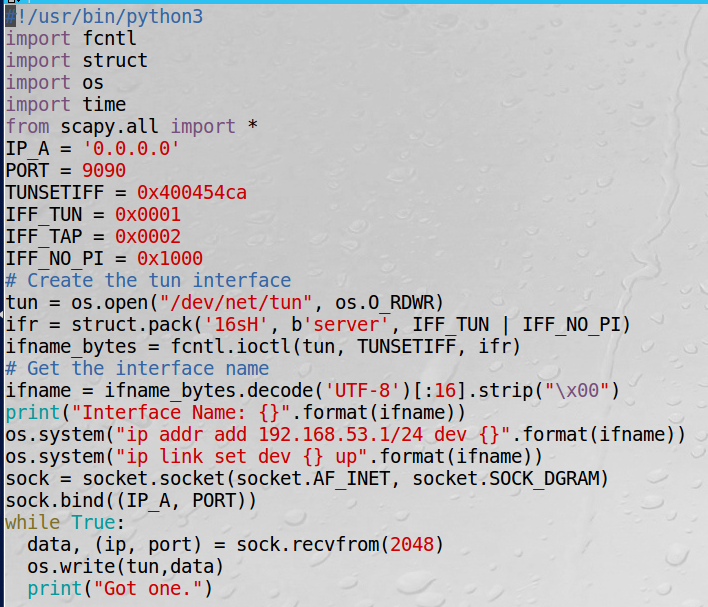


　　在Host U端ping 10.0.3.2，在VPN Server端正常接收到了Host U发往10.0.3.2的报文：



**Task4: Set Up the VPN Server**

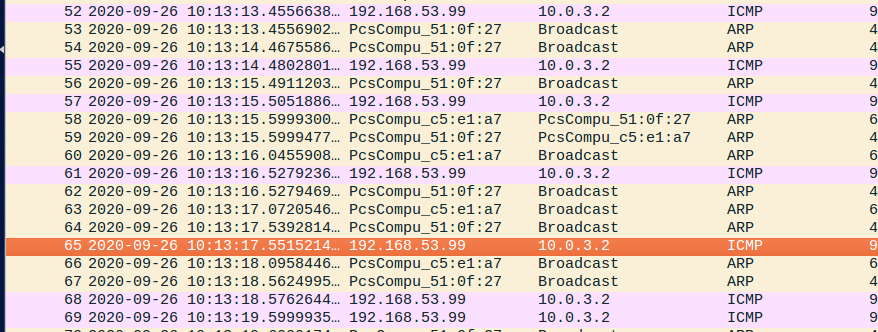
修改tun\_server.py：



　　打开VPN Server的路由选项：

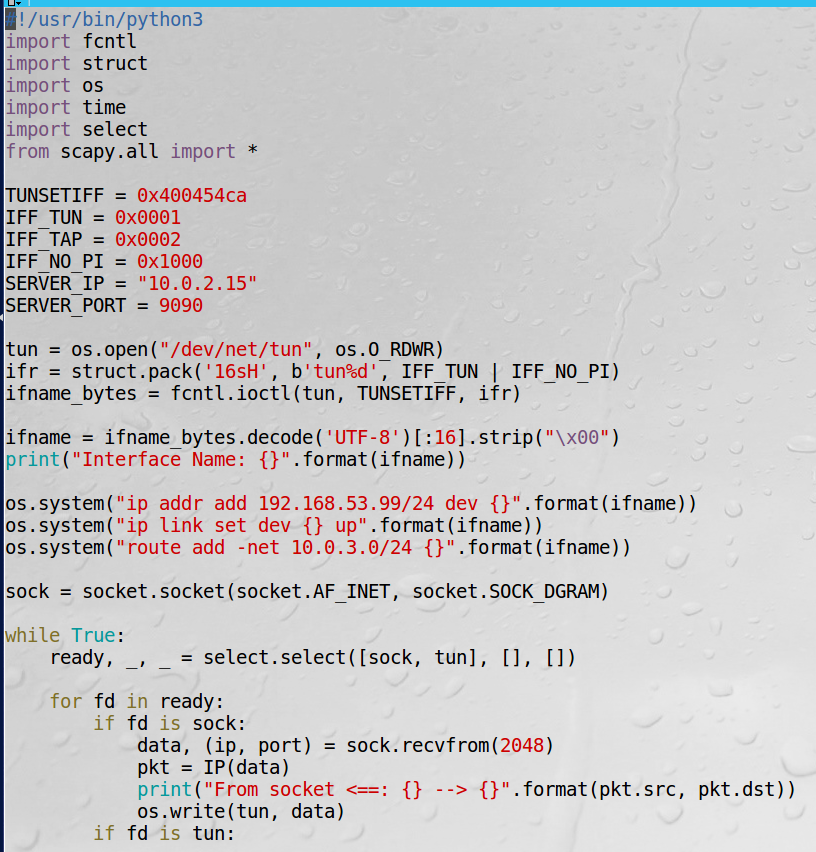


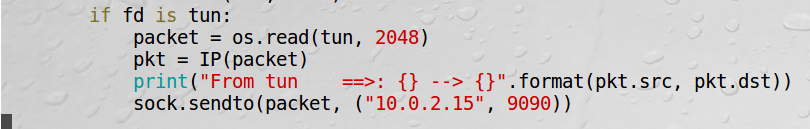
　　在Host V端用wireshark观察到已经收到了来自Host U的ICMP请求报文，但是只有单向流量：



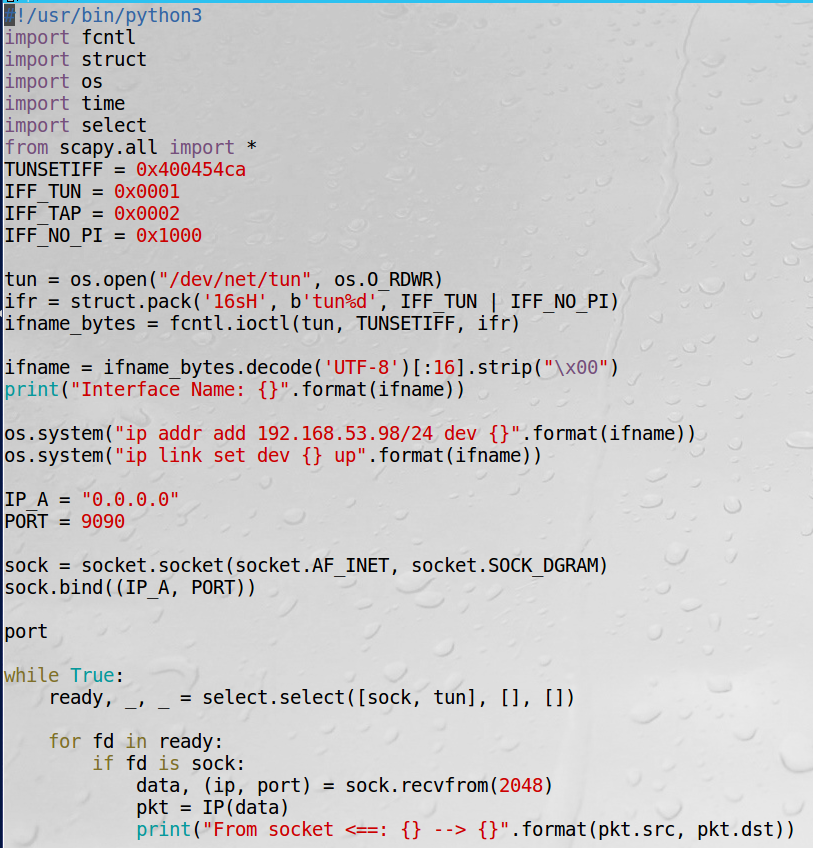
**Task5: Handling Traffic in Both Directions**

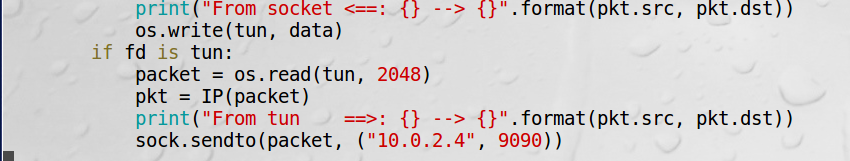
修改tun\_server.py:





修改tun\_client.py：

****

****

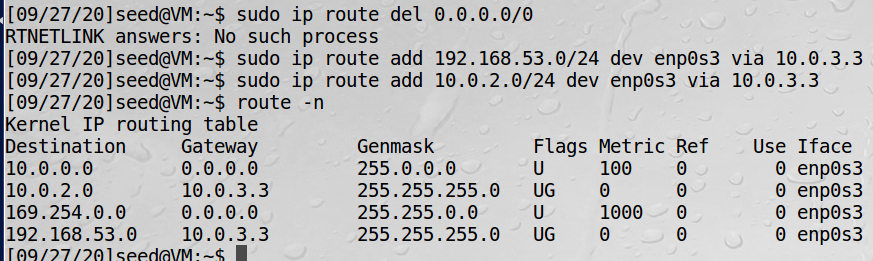
两端运行后在Host U执行ping 10.0.3.2，这一次能够看到双向流量。

**Task6: Tunnel-Breaking Experiment**

　　首先同时运行tun\_server.py和tun\_client.py，在Host U和Host V建立telnet连接之后可以正常输入命令，关闭tun\_server.py，发现没有反应但是连接并没有断，再次运行tun\_client.py，恢复正常。因为关闭tun\_client.py之后，之前建立的TCP连接将会将内容存进缓冲区进入重连状态，如果短时间内及时恢复tun\_client.py，那么缓冲区的字符又会重新通过TCP连接重新发送出去。

**Task7: Routing　Experiment　on　Host V**

对Host V进行路由配置：



然后由Host U对Host V发起telnet连接，双方经VPN server通信成功。