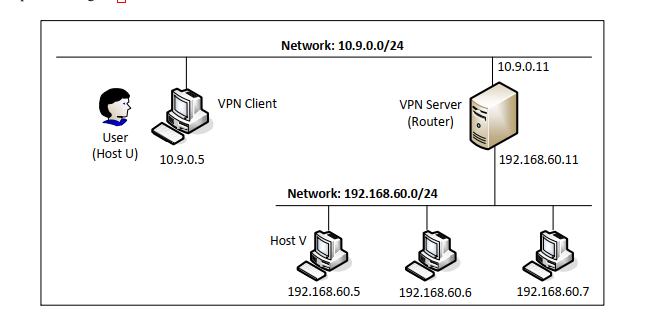
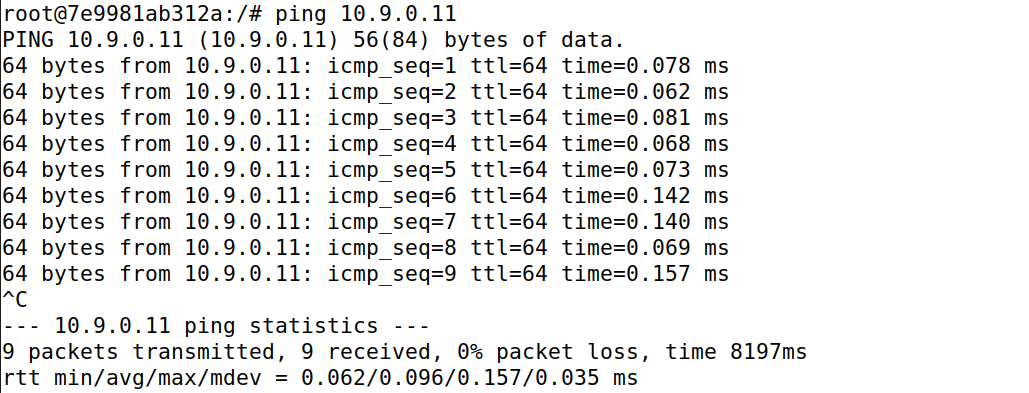
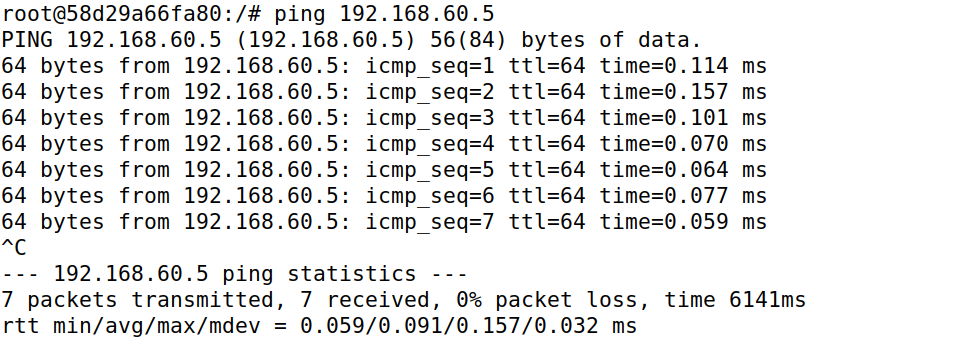
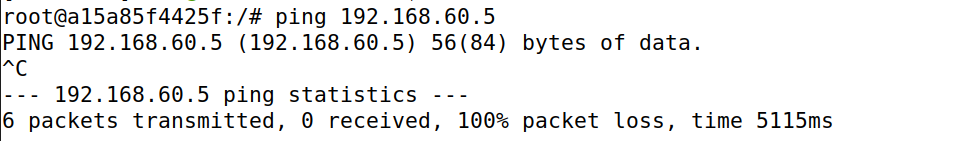
**Lab 7**

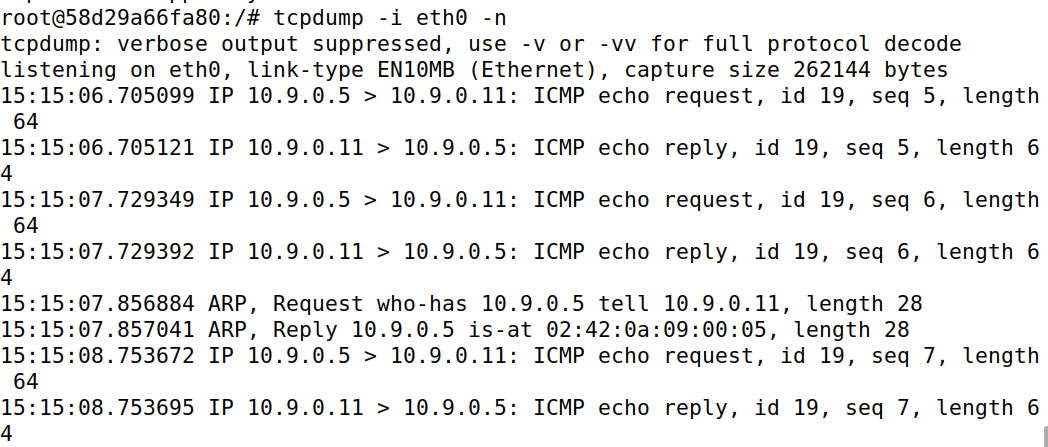
57118201邓彤

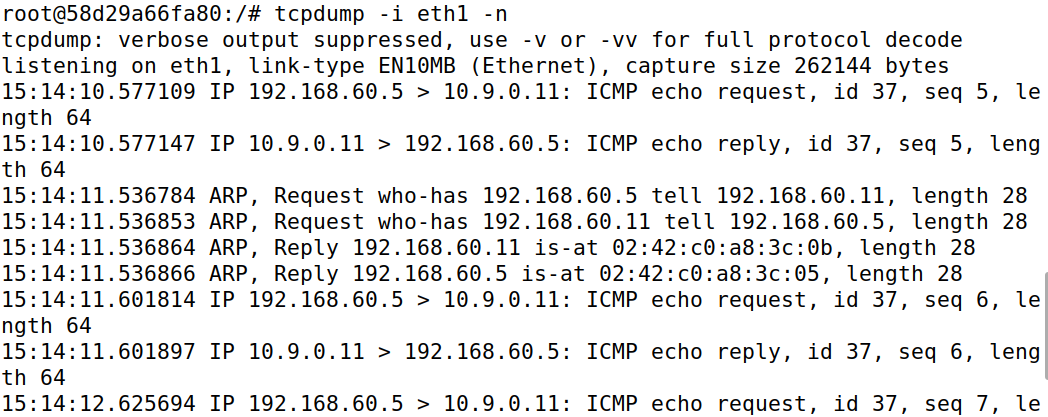
  
**Task 1: Network Setup**  
**1.在主机 V 上 ping VPN 服务器：**

 **正常通信  
2.在 VPN 服务器上 ping 主机 V:**

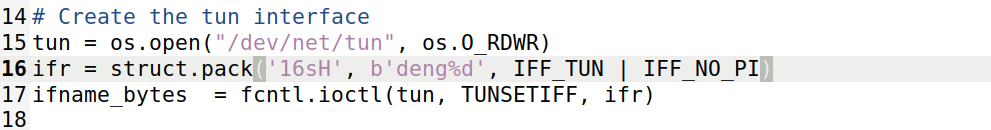
 **正常通信  
3.主机 U 上 ping 主机 V:**

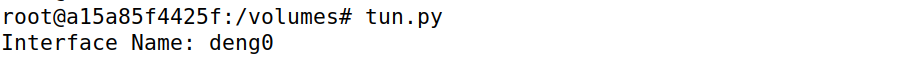
 **不能进行通信  
4. 路由器上运行 tcpdump：  
嗅探接口 eth0:**

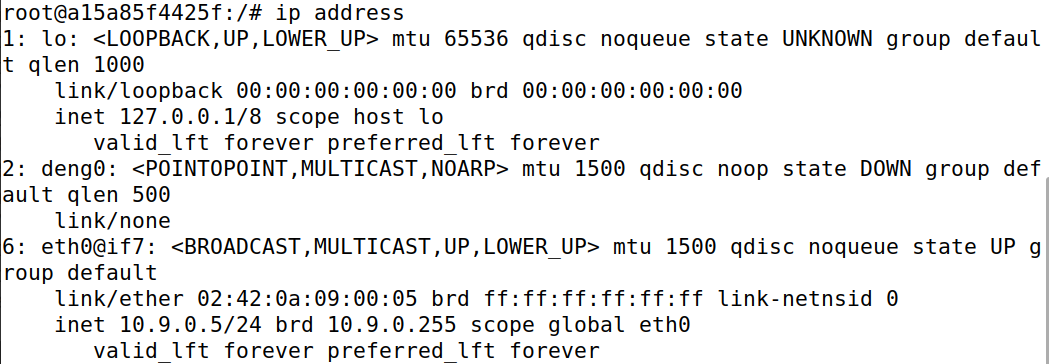
 **嗅探接口 eth1:**

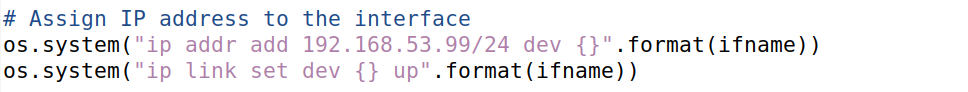
 **网络流量都可以正常嗅探  
配置正常。**

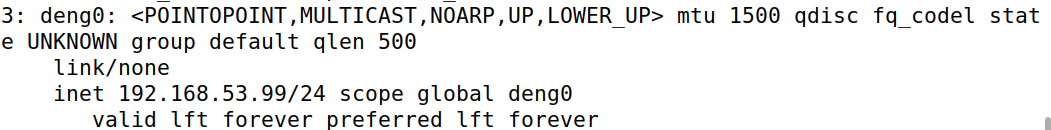
**Task 2: Create and Configure TUN Interface  
Task 2.a: Name of the Interface  
在代码中修改端口名为“deng” :**

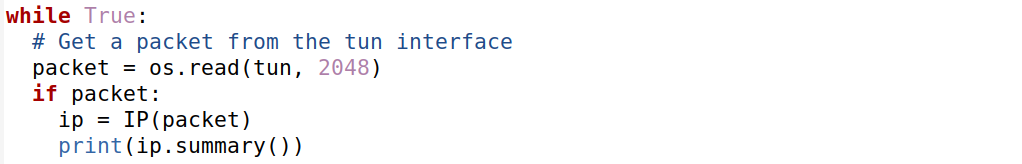
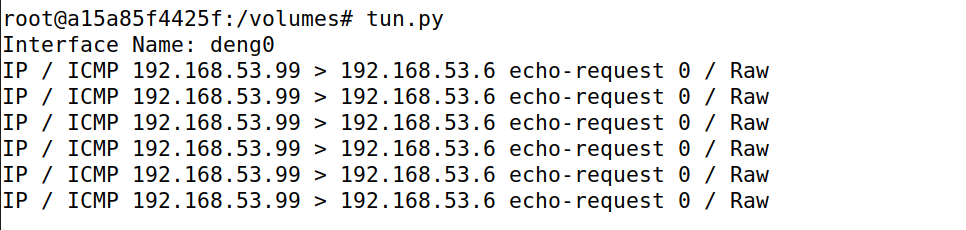
 **在主机 U 上运行程序：**

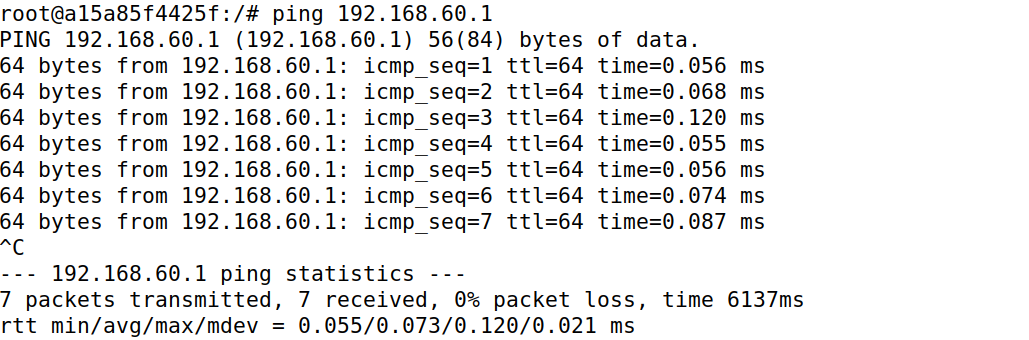
 **打开另一个终端查看：**

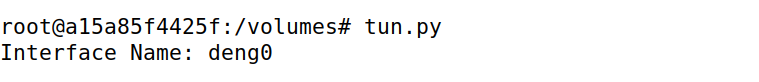
 **端口名成功修改为 deng0  
Task 2.b: Set up the TUN Interface  
程序中添加两行代码给端口 qiu0 自动分配 ip 地址：**

 **再次运行程序， 并执行 ip address 命令：**

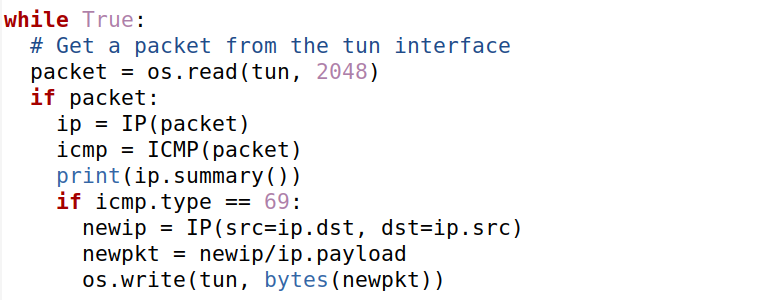
 **此时端口已经被成功分配了 ip 地址  
Task 2.c: Read from the TUN Interface  
修改程序中的 while 循环：**

 **再次执行程序，并 ping 192.168.53.0/24 网段中任一主机，ping 192.168.53.6  
此时 ping 不通，从 tun.py 程序的输出可以知道 ICMP 请求报文都被端口捕获了，** **因为发送给 192.168.53.0/24 的数据包是从 deng0 端口发出  
Ping 192.168.60.1 时：**

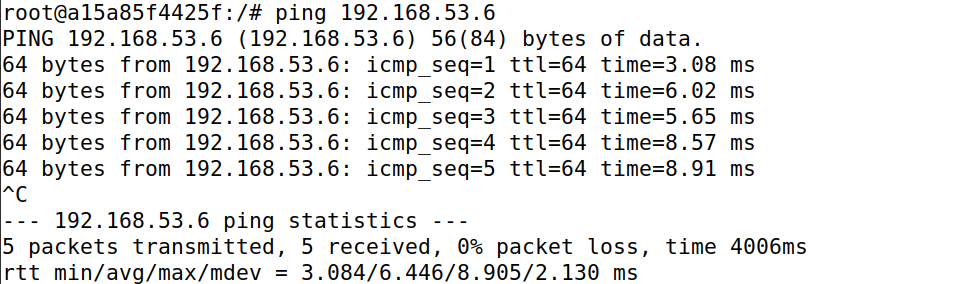
 **此时能 ping 通， 且此时程序没有输出**



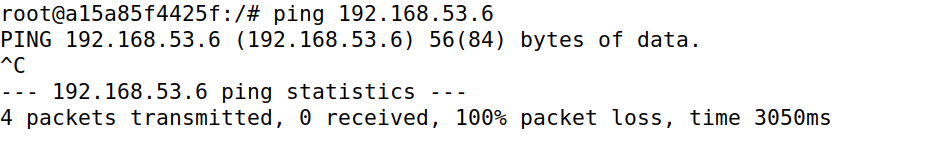
**这是因为发送给 192.168.60.1 的报文不经过 deng0 端口， 所以没有捕获报文。  
Task 2.d: Write to the TUN Interface  
修改 while 循环如下：**

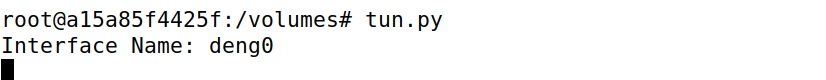


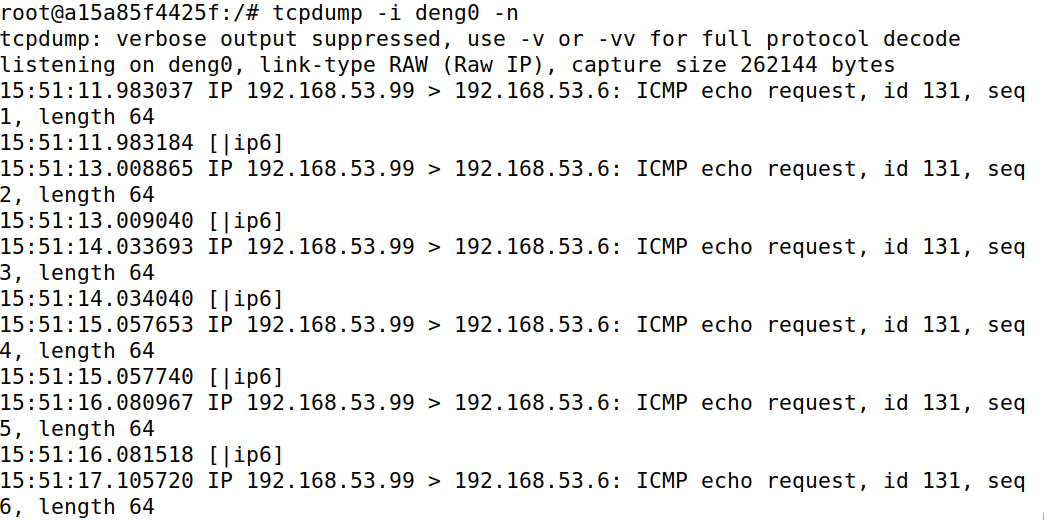
**运行程序，然后再次 ping 192.168.53.6：**

 **此时能够 ping 通，说明我们伪造响应包成功  
修改 while 循环，不写入 ip 数据包而是任意数据：**

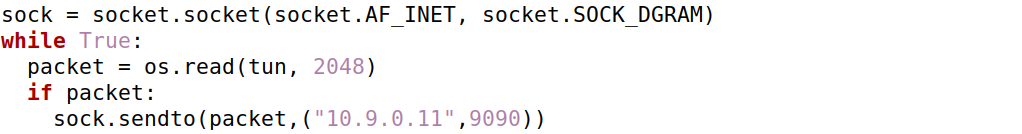
 **运行程序，然后再次 ping 192.168.53.6：**

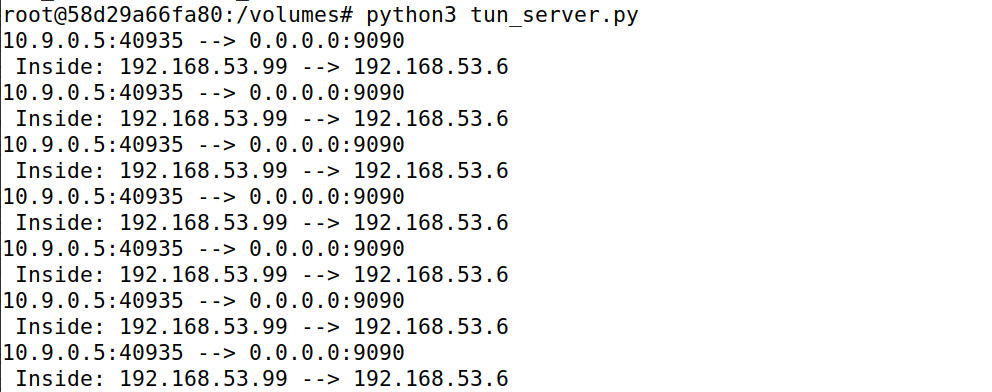
 **Ping 不通，且程序无输出，**

**执行 tcpdump -i deng0 -n 命令：**

 **可以看到发送的任意数据确实发送出去了，但是其不符合报文格式**

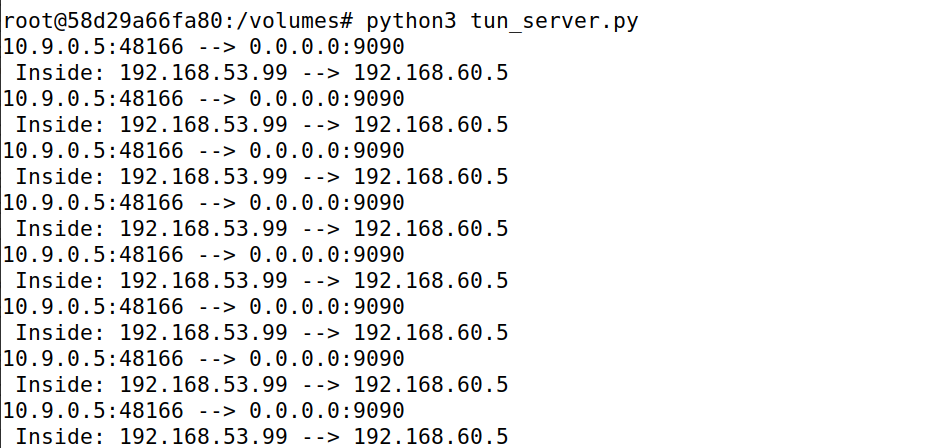
**Task 3: Send the IP Packet to VPN Server Through a Tunnel  
将 tun.py 程序的 while 循环修改为如下代码即为 tun\_client.py：**

 **在主机 U 上运行 tun\_client.py，在 VPN 服务器上运行 tun\_server.py。  
然后在主机 U 中 ping 192.168.53.6，VPN 服务器输出如下：**

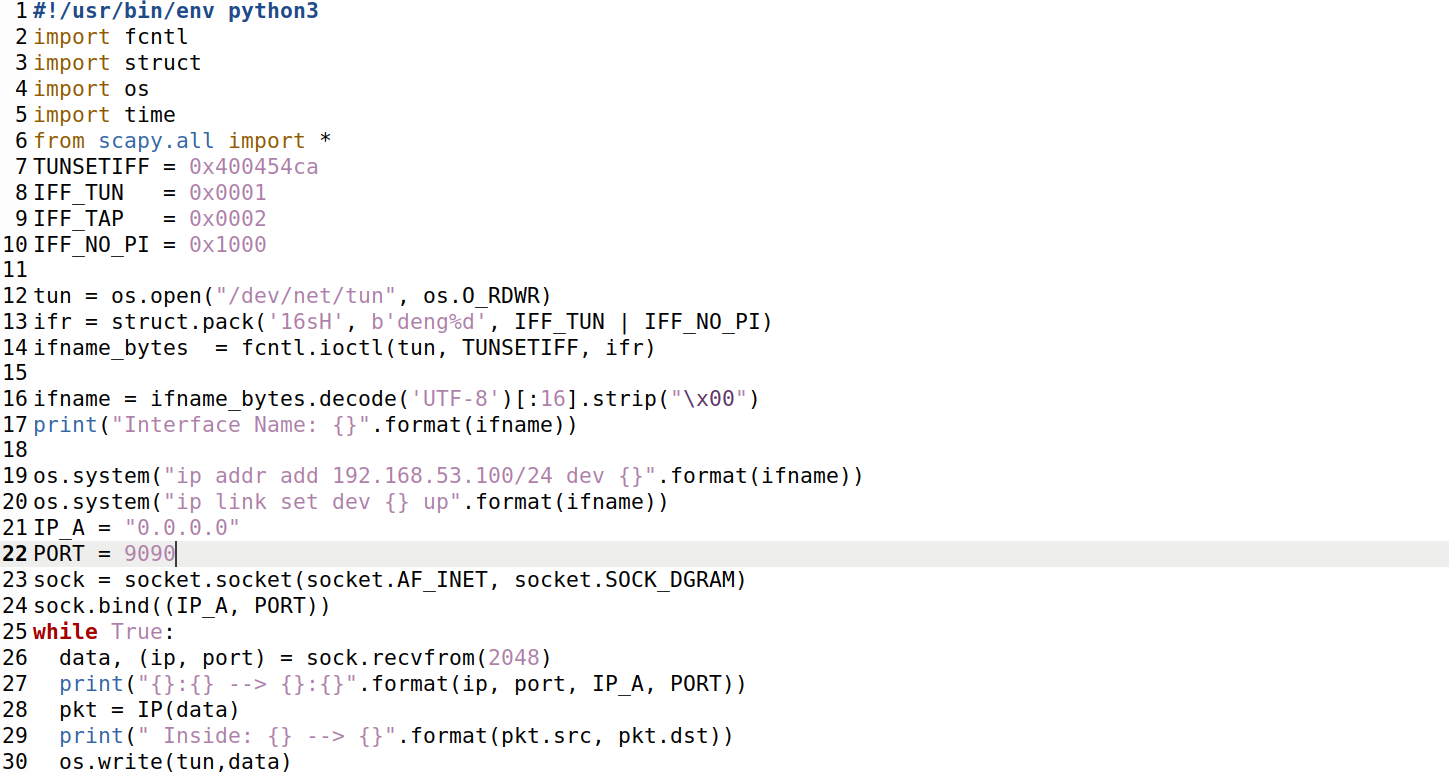
 **此时 VPN 服务器成功捕获到了报文，这是因为 tun\_client.py 程序将捕获的报文发给了 VPN 服务器的 9090 端口。  
在主机 U 上 Ping 主机 V:**

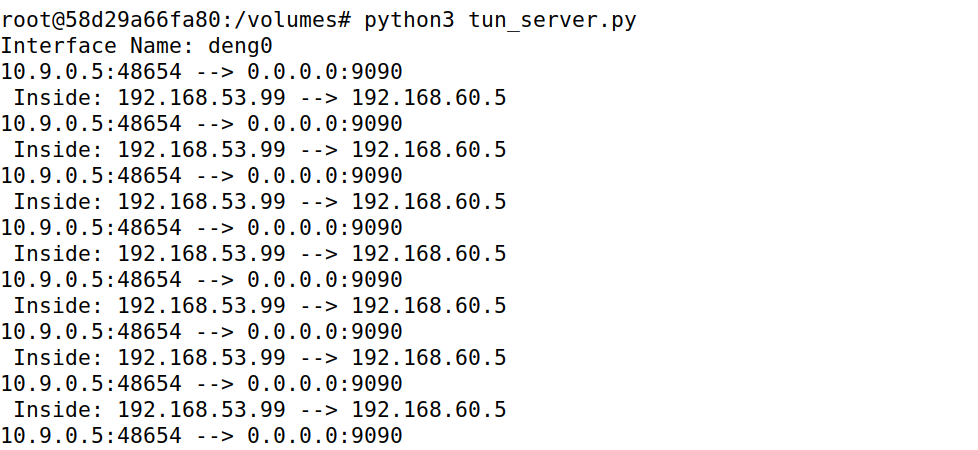
 **此时 VPN 服务器没有输出，这是因为此时主机 U 上没有去往 192.168.60.0/24的路由，报文不会从 tun 端口发出。  
手动配置路由：**

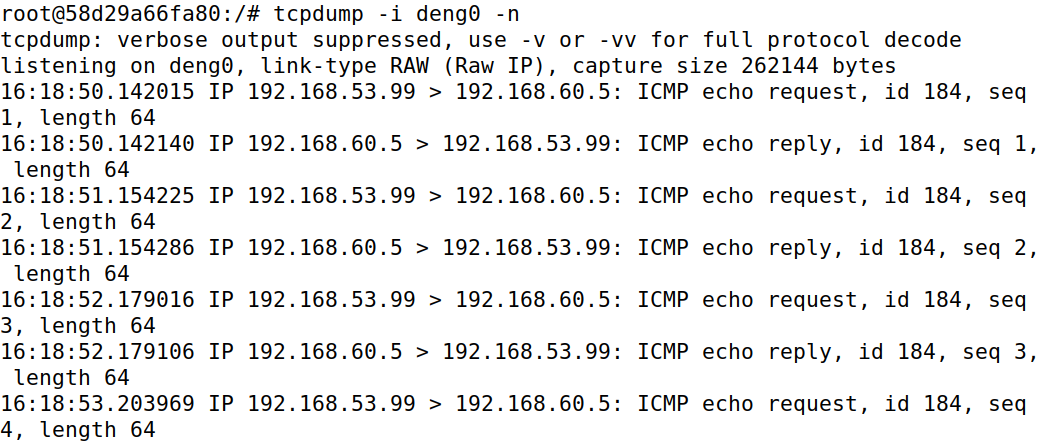
 **重复操作：**

 **此时 VPN 服务器有输出，说明 tun\_server.py 通过隧道接收到报文，实验成功**

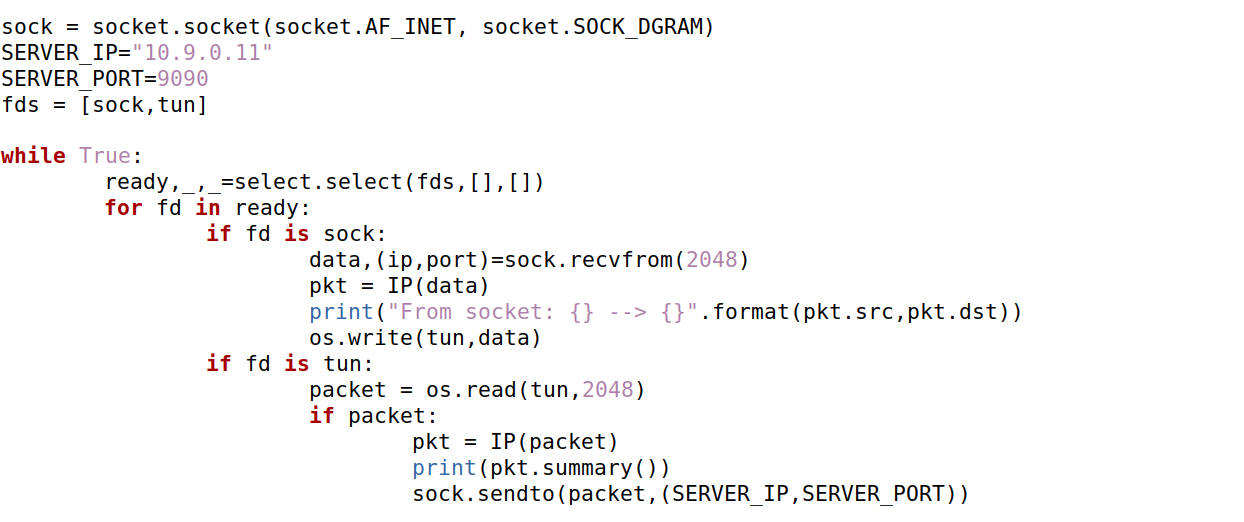
**Task 4: Set Up the VPN Server  
修改 tun\_server.py 代码使得建立一个 tun 接口将数据包路由到最终目的地，增加的代码类似于 task2：**

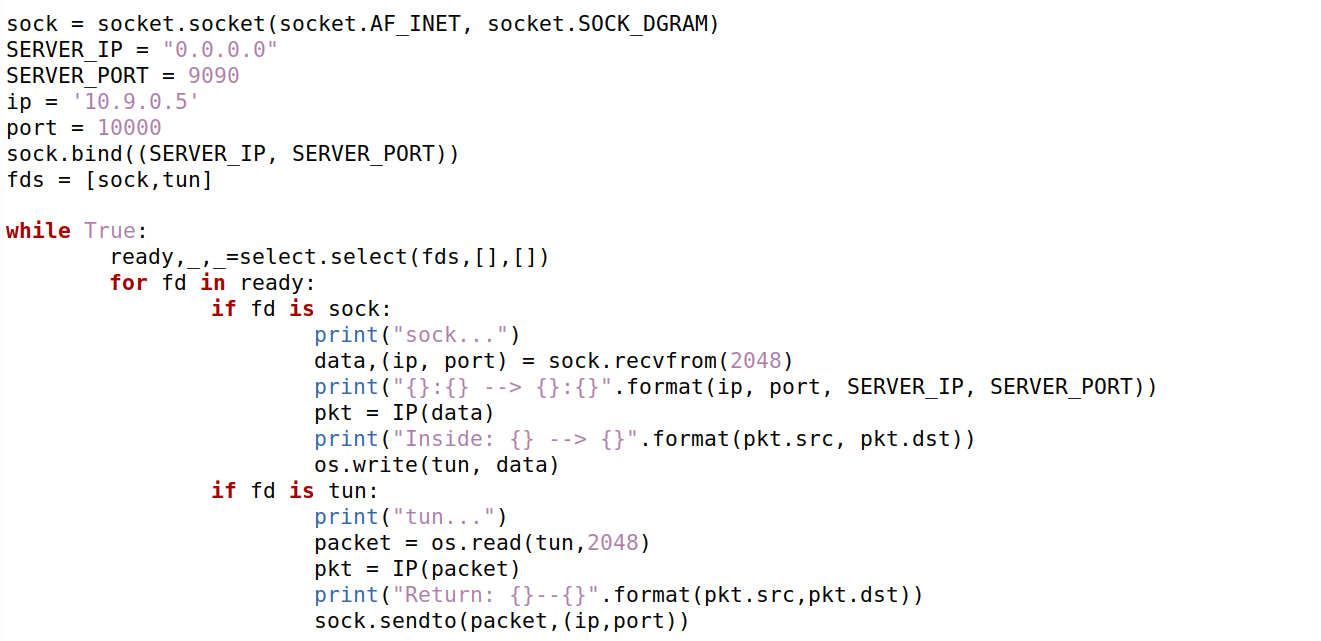
 **重复 task3 的操作，ping 192.168.60.5，tun\_server.py 输出如下：**

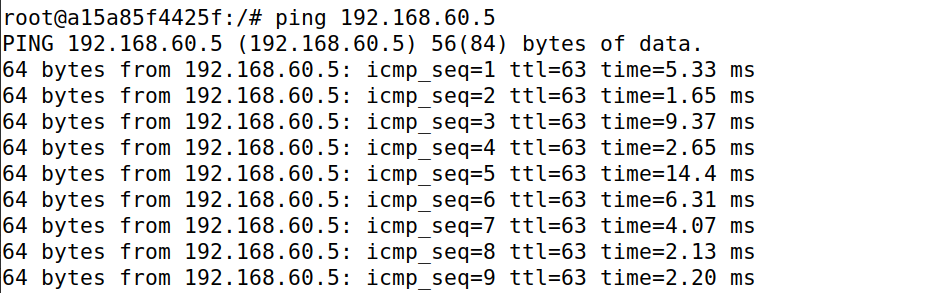
 **我们可以在 VPN 服务器上嗅探 deng0 端口：**

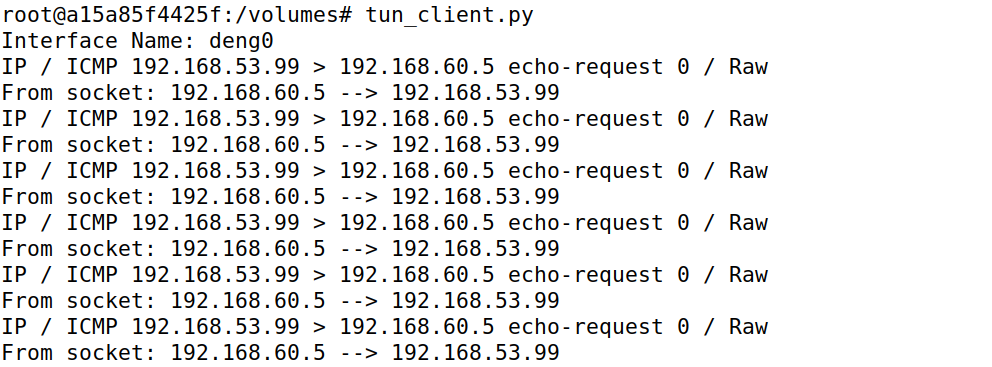
 **发现 ICMP 请求包成功通过隧道到达主机 V，且受到了主机 V 的 ICMP 响应包。但是此时还没有设置完成，此时隧道只有一个方向，故响应包无法到达主机 U。**

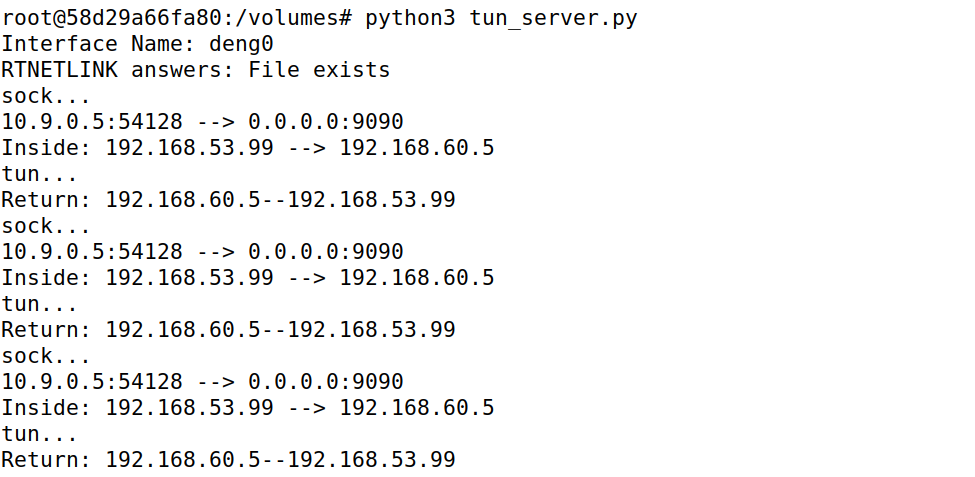
**Task 5: Handling Traffic in Both Directions  
为建立另一个方向的隧道， 我们修改代码中的 while 部分，tun\_client.py:**

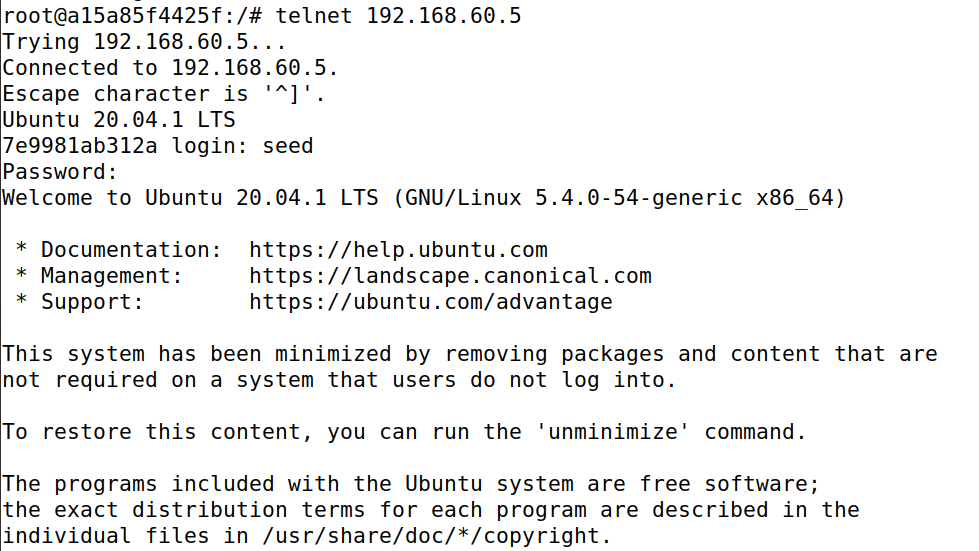
 **如果数据包来自 tun 接口， 则发给主机 U， 如果数据包来自 socket 接口， 则发给隧道。  
tun\_server.py:**

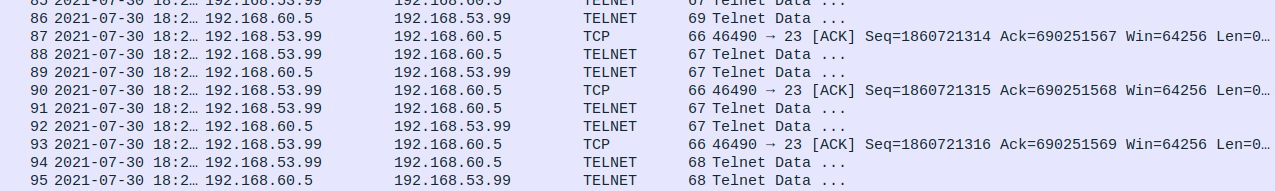
 **如果数据包来自 tun 接口，则发给主机 V，如果数据包来自 socket 接口，则发给隧道。  
重复之前的操作，ping 192.168.60.5:**

 **成功 ping 通，tun\_client.py 和 tun\_server.py 输出如下：**



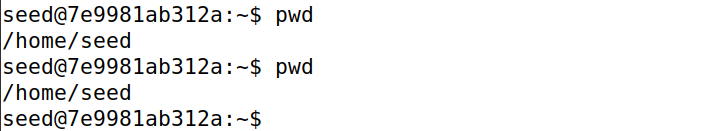
 **建立 Telnet 连接：**

 **Telnet 连接也成功建立。  
捕获 ping 过程中的数据包，查看 wireshark：**

 **数据报文从主机 U 发向主机 V， 报文先通过 tun 到达 VPN 服务器， 然后 VPN服务器通过 tun 发往主机 V 报文， 然后主机 V 返回响应报文通过 tun 达到 VPN服务器， VPN 服务器又通过 tun 将响应报文发给主机 U， 从而完成主机 U 和主机V 之间的通信。**

**Task 6: Tunnel-Breaking Experiment  
主机 U 向主机 V 建立 Telnet 连接，然后终止程序，发现无法输入任何字符：**

 **这是因为停止程序后隧道中断，数据包无法到达。  
短时间内再次执行程序：**

 **如果此时很快地执行程序恢复隧道，会发现前面中断程序时没能显示的输入会再次显示，Telnet 连接恢复。因为断开程序时的输入会在缓存区中一直发送报文， 如果恢复连接比较快速，前面的输入仍然会显示。但是较长时间还没再次执行程序就不能显示之前的输入了。**