

---

# 华中科技大学

## “计算机网络安全”实验报告

### 题目： VPN 实验

院 系 网络空间安全学院

专业班级 信安 1804

姓 名 余新宇

学 号 U201813742

日 期 2021 年 5 月

评分项	实验报告评分 (40%)	检查单分数 (60%)	综合得分	教师签名
得分				

## 实验报告评分标准

评分项目	分值	评分标准	得分
实验原理	20	18-20 系统流程清晰，报文处理过程描述清楚； 15-17 系统流程比较清晰，报文处理过程描述比较清楚； 12 以下 描述简单	
实验步骤	30	25-30 实验步骤描述详细、清楚、完整，前后关系清晰； 18-24 实验步骤描述比较清楚，关键步骤都进行了描述 18 分以下，实验步骤描述比较简单或不完整	
结果验证与分析	20	16-20，任务完成，针对任务点的测试，对结果有分析 10-15，针对任务点的测试截图，没分析 10 分以下，测试很简单，没有覆盖任务点	
心得体会	10	8-10 有自己的真实体会 4-7 真实体会套话 3 分以下，没有写什么体会	
格式规范	10	图、表的说明，行间距、缩进、目录等，一种不规范扣 1 分	
实验思考	10	思考题的回答，以及其它的简介	
总 分			

---

## 目 录

实验三 VPN 实验.....	1
1 实验目的.....	1
2 实验环境.....	1
3 实验内容.....	3
4 实验步骤及结果分析.....	3
5 实验思考.....	16
心得体会与建议.....	17
1 心得体会.....	17
2 建议.....	17

## 实验三 VPN 实验

### 1 实验目的

虚拟专用网络（VPN）用于创建计算机通信的专用的通信域，或为专用网络到不安全的网络（如 Internet）的安全扩展。VPN 是一种被广泛使用的安全技术。在 IPSec 或 TLS/SSL

（传输层安全性/安全套接字层）上构建 VPN 是两种根本不同的方法。本实验中，我们重

点关注基于 TLS/SSL 的 VPN。这种类型的 VPN 通常被称为 TLS/SSL VPN。

本实验的学习目标是让学生掌握 VPN 的网络和安全技术。为实现这一目标，要求学生实现简单的 TLS/SSL VPN。虽然这个 VPN 很简单，但它包含了 VPN 的所有基本元素。

TLS/SSL VPN 的设计和实现体现了许多安全原则，包括以下内容：

- 虚拟专用网络
- TUN/TAP 和 IP 隧道
- 路由
- 公钥加密，PKI 和 X.509 证书
- TLS/SSL 编程
- 身份认证

### 2 实验环境

#### 2.1 网络拓扑



图 2-1 网络拓扑图

## 2.2 操作系统

Ubuntu Seed 虚拟机

ubuntu 系统的用户密码: seed: dees

root 密码: seedubuntu

实验需要多台虚拟机, 可以采用虚拟机+docker 容器构建

## 2.3 网络配置

利用 docker 容器创建 HostU, HostV 及网络结构

在 VM 上创建 docker 网络 extranet

```
$ sudo docker network create --subnet=10.0.2.0/24 --gateway=10.0.2.8 --opt  
"com.docker.network.bridge.name"="docker1" extranet
```

在 VM 上创建 docker 网络 intranet

```
$ sudo docker network create --subnet=192.168.60.0/24 --gateway=192.168.60.1 --  
opt "com.docker.network.bridge.name"="docker2" intranet
```

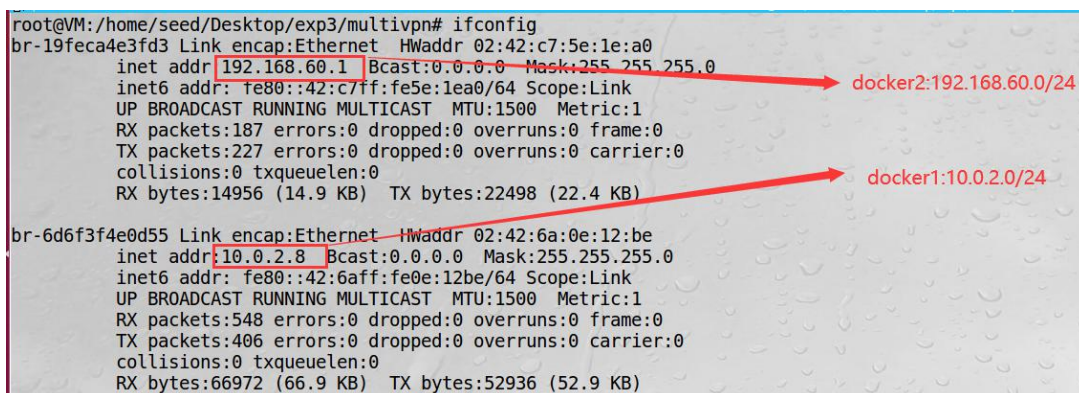
在 VM 上新开一个终端, 创建并运行容器 HostU

```
$ sudo docker run -it --name=HostU --hostname=HostU --net=extranet --ip=10.0.2.7  
--privileged "seedubuntu" /bin/bash
```

在 VM 上新开一个终端, 创建并运行容器 HostV

```
$ sudo docker run -it --name=HostV --hostname=HostV --net=intranet --  
ip=192.168.60.101 --privileged "seedubuntu" /bin/bash
```

图 2-2 docker 网络配置代码



```
root@VM:/home/seed/Desktop/exp3/multivpn# ifconfig  
br-19feca4e3fd3 Link encap:Ethernet HWaddr 02:42:c7:5e:1e:a0  
  inet addr: 192.168.60.1 Bcast:0.0.0.0 Mask:255.255.255.0  
  inet6 addr: fe80::42:c7:ff:fe5e:1ea0/64 Scope:Link  
  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
  RX packets:187 errors:0 dropped:0 overruns:0 frame:0  
  TX packets:227 errors:0 dropped:0 overruns:0 carrier:0  
  collisions:0 txqueuelen:0  
  RX bytes:14956 (14.9 KB) TX bytes:22498 (22.4 KB)  
  
br-6d6f3f4e0d55 Link encap:Ethernet HWaddr 02:42:6a:0e:12:be  
  inet addr: 10.0.2.8 Bcast:0.0.0.0 Mask:255.255.255.0  
  inet6 addr: fe80::42:6aff:fe0e:12be/64 Scope:Link  
  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
  RX packets:548 errors:0 dropped:0 overruns:0 frame:0  
  TX packets:406 errors:0 dropped:0 overruns:0 carrier:0  
  collisions:0 txqueuelen:0  
  RX bytes:66972 (66.9 KB) TX bytes:52936 (52.9 KB)
```

图 2-3 VPNserver 网络配置

```

root@HostU:/multivpn# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:0a:00:02:07
          inet addr:10.0.2.7  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::42:aff:fe00:207/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:261 errors:0 dropped:0 overruns:0 frame:0
          TX packets:181 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:23597 (23.5 KB)  TX bytes:30679 (30.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:192.168.53.5  P-t-P:192.168.53.5  Mask:255.255.255.0
          inet6 addr: fe80::1da6:4229:952e:a228/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:161 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:13416 (13.4 KB)

```

图 2-4 HostU 网络配置

```

root@HostV:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:c0:a8:3c:65
          inet addr:192.168.60.101  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::42:c0ff:fea8:3c65/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:259 errors:0 dropped:0 overruns:0 frame:0
          TX packets:187 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:26150 (26.1 KB)  TX bytes:17574 (17.5 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

```

图 2-5 HostV 网络配置

### 3 实验内容

本次实验，需要为 Linux 操作系统实现一个简单的 VPN。我们将其称为 miniVPN。

#### 任务 1：配置虚拟机环境

我们将在计算机（客户端）和网关之间创建 VPN 隧道，允许计算机通过网关安全地访问专用网络。我们至少需要三个虚拟机：VPN 客户端（也称为服务器作为主机 U），VPN 服务器（网关）和专用网络中的主机（主机 V）。网络设置如图 3-1 所示。

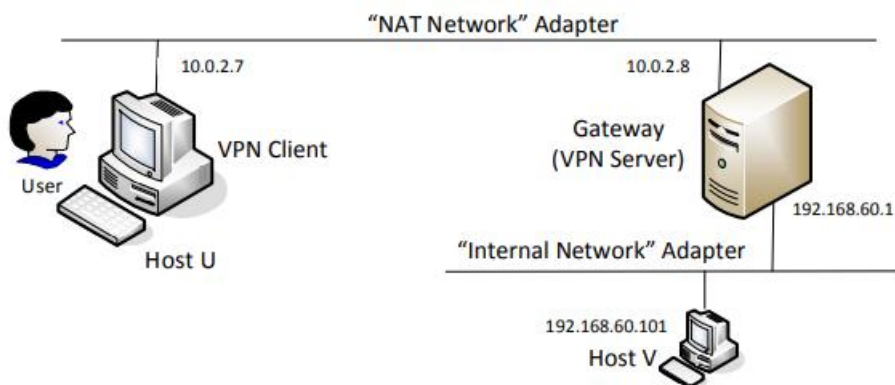


图 3-1 配置虚拟机实验环境

## 任务 2：使用 TUN/TAP 创建一个主机到主机的隧道

`vpnclient` 和 `vpnservice` 程序是 VPN 隧道的两端，它们使用 TCP 或 UDP 协议通过套接字相互通信。为简单起见，在我们的示例代码中，我们选择使用 UDP。客户端和服务端之间的虚线描述了 VPN 隧道的路径。VPN 客户端和服务端程序通过 TUN 接口连接到主机系统，做以下两件事：

- （1）从主机系统获取 IP 数据包，因此数据包可以通过隧道发送；
- （2）从隧道获取 IP 数据包，然后将其转发到主机系统，主机系统将数据包转发到其最终目的地。以下过程介绍了如何使用 `vpnclient` 和 `vpnservice` 程序创建 VPN 隧道。

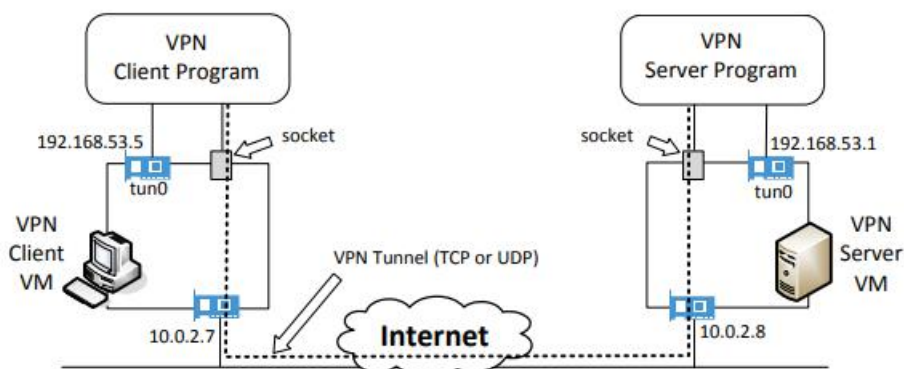


图 3-2 VPN 的客户端和服务端

## 任务 3：加密隧道

此时，我们已经创建了一个 IP 隧道，但是我们的隧道没有受到保护。只有在我们保障了隧道的安全之后，才能将其称为 VPN 隧道。这就是我们在这项任务中要实现的目标。为了保护这条隧道，我们需要实现两个目标，即机密性和完整性。使用加密来实现机密性，即，通过隧道的内容将被加密。完整性目标确保没有人可以篡改隧道中的流量或发起重放攻击。使用消息验证代码（MAC）可以实现完整性。可以使用传输层协议（TLS）实现这两个目标。

TLS 通常建立在 TCP 之上。任务 2 中的示例 VPN 客户端和服务端程序使用 DP，因此我们首先需要使用 TCP 通道替换示例代码中的 UDP 通道，然后在隧道的两端之间建立 TLS 会话。

## 任务 4：认证 VPN 服务器



在建立 VPN 之前，VPN 客户端必须对 VPN 服务器进行身份认证，确保服务器不是假冒的服务器。另一方面，VPN 服务器必须认证客户端（即用户），确保用户具有访问专用网络的权限。在这个任务中，我们实现服务器认证。客户端身份认证在下一个任务中。

认证服务器的典型方法是使用公钥证书。VPN 服务器需要首先从证书颁发机构（CA）获取公钥证书。当客户端连接到 VPN 服务器时，服务器将使用证书来证明它是客户端预期的服务器。Web 中的 HTTPS 协议使用这种方式来认证 Web 服务器，确保客户端正在与预期的 Web 服务器通信，而不是伪造的 Web 服务器。

### 任务 5：认证 VPN 客户端

访问专用网络内的计算机是一项权限，仅授予授权用户，而不是授予所有人。

因此，只允许授权用户与 VPN 服务器建立 VPN 隧道。在此任务中，授权用户是在 VPN 服务器上拥有有效帐户的用户。因此，我们将使用标准密码身份验证来验证用户身份。基本上，当用户尝试与 VPN 服务器建立 VPN 隧道时，将要求用户提供用户名和密码。服务器将检查其影子文件（/etc/shadow）；如果找到匹配的记录，则对用户进行认证，并建立 VPN 隧道。如果没有匹配，服务器将断开与用户的连接，因此不会建立隧道。

### 任务 6：支持多个客户端

在真实应用中，一个 VPN 服务器通常支持多个 VPN 隧道。也就是说，VPN 服务器允许多个客户端同时连接到它，每个客户端都有自己的 VPN 隧道（从而有自己的 TLS 会话）。MiniVPN 应该支持多个客户端。

一旦做出决定并选择了隧道，父进程就需要将数据包发送到附加了所选隧道的子进程。

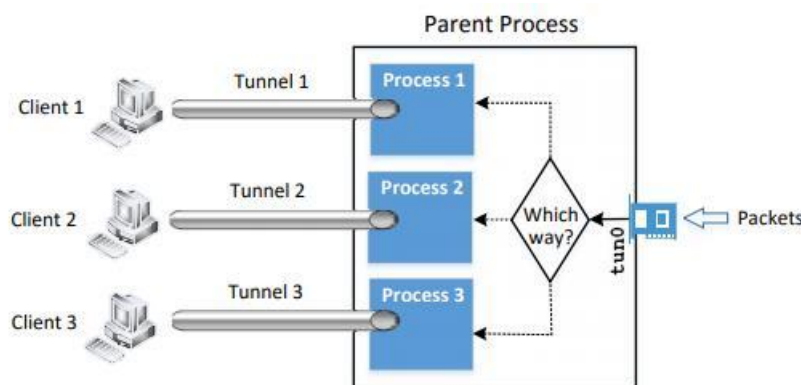


图 3-3 支持多个 VPN 客户端

## 4 实验步骤及结果分析

### 4.1 配置环境

在 VM 上创建 docker 网络 extranet、intranet



```

root@VM:/home/seed# docker network create --subnet=10.0.2.0/24 --gateway=10.0.2.8 --opt "com.docker.network.bridge.name0"="docker1" extranet
6d6f3f4e0d5524dc8b135279462d1f88a71d4894a361e0e9ed54a10603820058
root@VM:/home/seed# docker network create --subnet=192.168.60.0/24 --gateway=172.168.60.1 --opt "com.docker.network.bridge.name0"="docker2" intranet
no matching subnet for gateway 172.168.60.1
root@VM:/home/seed# docker network create --subnet=192.168.60.0/24 --gateway=192.168.60.1 --opt "com.docker.network.bridge.name0"="docker2" intranet
19feca4e3fd3af4c17982b6bb85b9262ef6453e4165bbe770e73b05cdd1bda5f

```

图 4-1 创建网络

新开一个终端，创建并运行容器 HostV，HostV 与 VM 在 extranet 外网（192.168.60.0/24）中，其中 HostV 为 192.168.60.101，VM 为 192.168.60.1

```

[05/07/21]seed@VM:~$ sudo docker run -it --name=HostV --hostname=HostV --net=intranet --ip=192.168.60.101 --privileged "seedubuntu" /bin/bash
h
[05/07/21]seed@VM:~$ sudo docker start HostV
HostV
[05/07/21]seed@VM:~$ sudo docker exec -it HostV /bin/bash
root@HostV:/#

```

图 4-2 docker 创建 HostV

新开一个终端，创建并运行容器 HostU，HostV 与 VM 在 internet 内网（10.0.2.0/24）中，其中 HostU 为 10.0.2.7，VM 为 10.0.2.8

```

root@788e61c38a72:/# exit[05/07/21]seed@VM:~$
[05/07/21]seed@VM:~$ sudo docker run -it --name=HostU --hostname=HostU --net=extranet --ip=10.0.2.7 --privileged "seedubuntu" /bin/bash
[sudo] password for seed:
root@HostU:/# sudo docker start HostU
Cannot connect to the Docker daemon. Is the docker daemon running on this host?
Error: failed to start containers: HostU
root@HostU:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:0a:00:02:07
          inet addr:10.0.2.7  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::42:aff:fe00:207/64 Scope:Link

```

图 4-3 docker 创建 HostU

在容器 HostU 和 HostV 内分别删除掉默认路由  
# route del default

#### 4.2.1 运行 VPN 服务器

```
tun0 Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00
POINTOPOINT NOARP MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:500
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

vethb4dedfb Link encap:Ethernet HWaddr 0e:3a:09:01:a5:ef
inet6 addr: fe80::c3a:9ff:fe01:a5ef/64 Scope:Link

root@VM: /home/seed/Desktop/exp3/vpn
root@VM: /home/seed/Desktop/exp3/vpn 80x7
Makefile README vpnclient.c vpnserv.c
root@VM: /home/seed/Desktop/exp3/vpn# make
gcc -o vpnserv vpnserv.c
gcc -o vpnclient vpnclient.c
root@VM: /home/seed/Desktop/exp3/vpn# sudo ./vpnserv
root@VM: /home/seed/Desktop/exp3/vpn# Setup TUN interface success!
```

当服务器运行以后，VM 就会出现名为 tun0 的网卡，所以在另一个终端 (HostU) 配置对应的 tun0 虚拟 IP 地址并激活接口

```
[05/07/21]seed@VM:~$ sudo ifconfig tun0 192.168.53.1/24 up
[sudo] password for seed:
[05/07/21]seed@VM:~$
```

在 VM 上启用 IP 转发、在 VM 上清除 iptables 规则

```
[05/07/21]seed@VM:~$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[05/07/21]seed@VM:~$
[05/07/21]seed@VM:~$ sudo iptables -F
```

#### 4.2.2 运行 VPN 客户端

```
[05/07/21]seed@vm:~/..../vpns sudo docker cp vpnclient host0:/vpnclient
[05/07/21]seed@VM:~/..../vpns [1]

root@VM: /home/seed/Desktop/exp3/vpn
root@VM: /home/seed/Desktop/exp3/vpn# sudo ./vpnserv
root@VM: /home/seed/Desktop/exp3/vpn# Setup TUN interface success!
Accept connect from client 10.0.2.7: Hello
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN

root@HostU: /# route del default
root@HostU: /# ..vpnclient 10.0.2.8
root@HostU: /# Setup TUN interface success!
Connect to server 10.0.2.8: Hello
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
```

容器 HostU 新开一个终端，tun0 虚拟 IP 地址并激活接口，这个过程 VM 上的 vpnserver 和 HostU 的 vpnclient 都是可以接收到包的

```
[05/07/21]seed@VM:~$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[05/07/21]seed@VM:~$
[05/07/21]seed@VM:~$ sudo iptables -F
[05/07/21]seed@VM:~$ sudo docker cp vpnclient HostU:/vpnclient
lsstat /home/seed/vpnclient: no such file or directory
[05/07/21]seed@VM:~$ cd Desktop/
exp2/ exp3/ TCP/
[05/07/21]seed@VM:~$ cd Desktop/exp3/vpn/
[05/07/21]seed@VM:~/exp3/vpn$ ls
Makefile README vpnclient vpnclient.c vpnserver vpnserver.c
[05/07/21]seed@VM:~/exp3/vpn$ sudo docker cp vpnclient HostU:/vpnclient
[05/07/21]seed@VM:~/exp3/vpn$
root@HostU:/# Setup TUN interface success!
Connect to server 10.0.2.8: Hello
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
[05/07/21]seed@VM:~$ sudo docker exec -it HostU /bin/bash
[sudo] password for seed:
root@HostU:/# sudo ifconfig tun0 192.168.53.5/24 up
root@HostU:/#
```

图 4-8 客户端服务器正常通讯

4.2.3 在 HostU 和 VPN 服务器上设置路由

HostU 中将 192.168.60.0/24 数据包路由到接口 tun0

```
root@VM:/home/seed/Desktop/exp3/vpn# route add -net 192.168.60.0/24 tun0
root@VM:/home/seed/Desktop/exp3/vpn# route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
0.0.0.0          192.168.142.2   0.0.0.0        UG    100    0      0 ens33
10.0.2.0         0.0.0.0         255.255.255.0  U     0      0      0 br-6d6f3f4e6d55
169.254.0.0      0.0.0.0         255.255.0.0    U     1000   0      0 docker0
172.17.0.0       0.0.0.0         255.255.0.0    U     0      0      0 docker0
192.168.53.0     0.0.0.0         255.255.255.0  U     0      0      0 tun0
192.168.53.0     0.0.0.0         255.255.255.0  U     0      0      0 tun0
192.168.60.0     0.0.0.0         255.255.255.0  U     0      0      0 tun0
```

图 4-9 设置路由

将 HostU 和 VM 中进入 192.168.53.0/24 网络的所有流量定向到 tun0 接口（非必须，已含有）

4.2.4 在 HostV 上设置路由

将 HostV 想发给 10.0.2.0/24 的全部数据包定向到 tun0

route add -net 192.168.53.0/24 gw 192.168.60.1

4.2.5 测试 VPN 隧道

分别 ping 和 telnet

Ping:

```
64 bytes from 192.168.60.101: icmp_seq=3 ttl=63 time=0.170 ms
64 bytes from 192.168.60.101: icmp_seq=4 ttl=63 time=0.188 ms
^Z
[6]+ Stopped ping 192.168.60.101
```

图 4-10 ping 测试 VPN 隧道图

Telnet:

HostV 重启 telnet

sudo /etc/init.d/openbsd-inetd restart



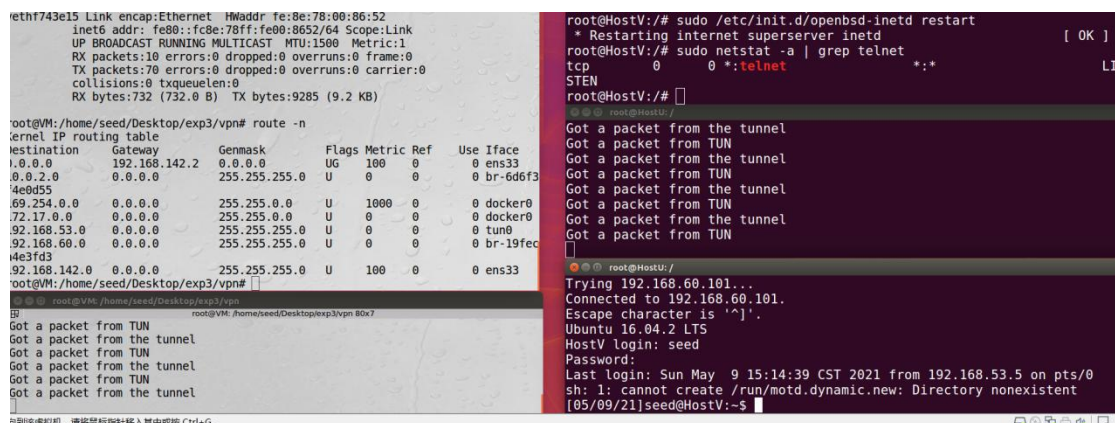


图 4-11 telnet 测试 VPN 隧道图

### Wireshark 截图

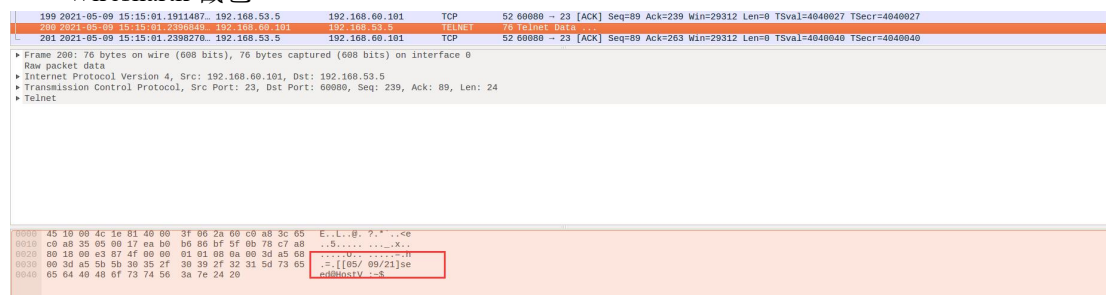


图 4-12 wireshark 截图

## 4.2.6 Tunnel 断开测试

在保持 telnet 连接存活的同时，断开 VPN 隧道。

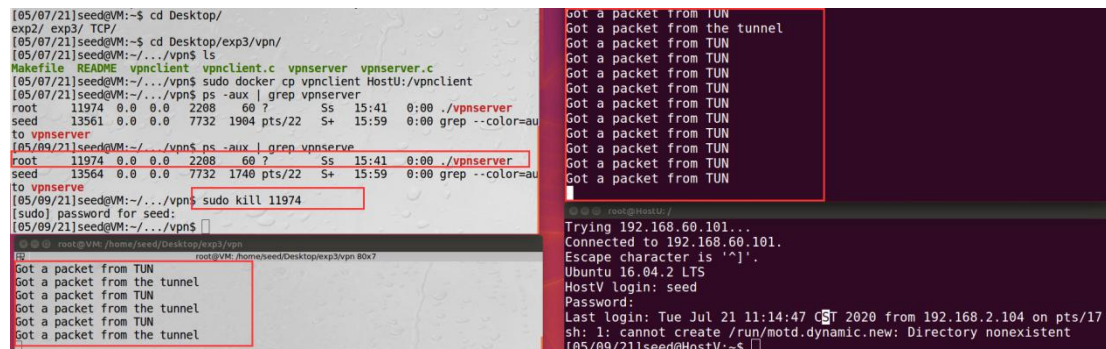


图 4-13 Tunnel 断开测试图

如图，kill 进程 vpnserver 以后，当再次在 telnet 中写入时，HostU 中的 vpnclient 通过隧道连发几个包均无响应，于是 telnet 连接断开，且再也无法写入或其他操作。

当 vpnserver 与 vpnclient 重连以后，telnet 仍然无响应，这是因为对应进程号一定不相同了，连接又不是同一个连接，所以仍然无反应

### 4.3 任务 3：加密隧道

同样打开 `vpnsrv` 和 `vpnc`，建立 `tun0` 隧道  
之后打开 `tlssrv`，然后按照要求配置，在 `/etc/hosts` 中添加 `10.0.2.8 vpnlabsrvr.com`，  
运行 `tlsc` `vpnlabsrvr.com 4433`

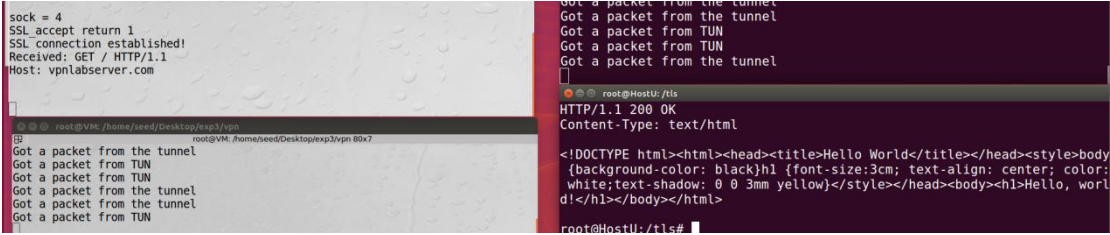


图 4-14 tls 测试图

成功建立 ssl 连接  
在 `Wireshark` 中也可以检测到 `tun0` 中有信息传递，具体如下，确实完成了加密

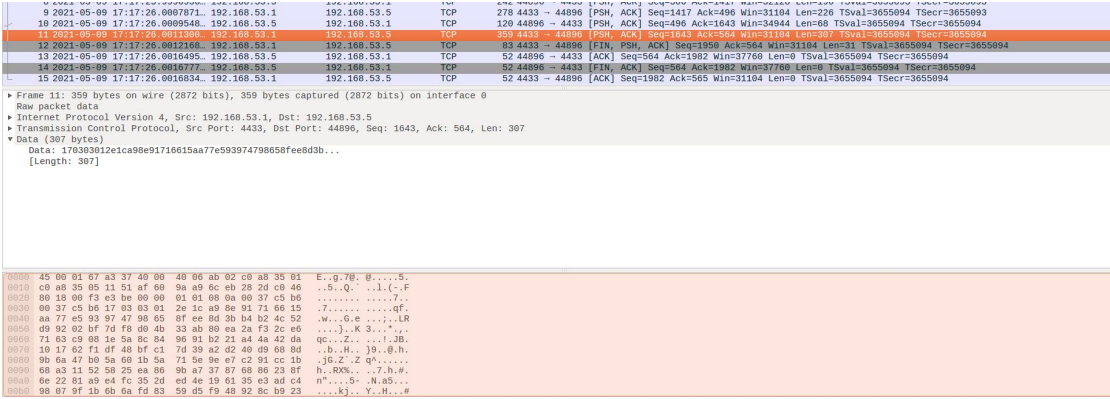


图 4-15 Wireshark 截包图

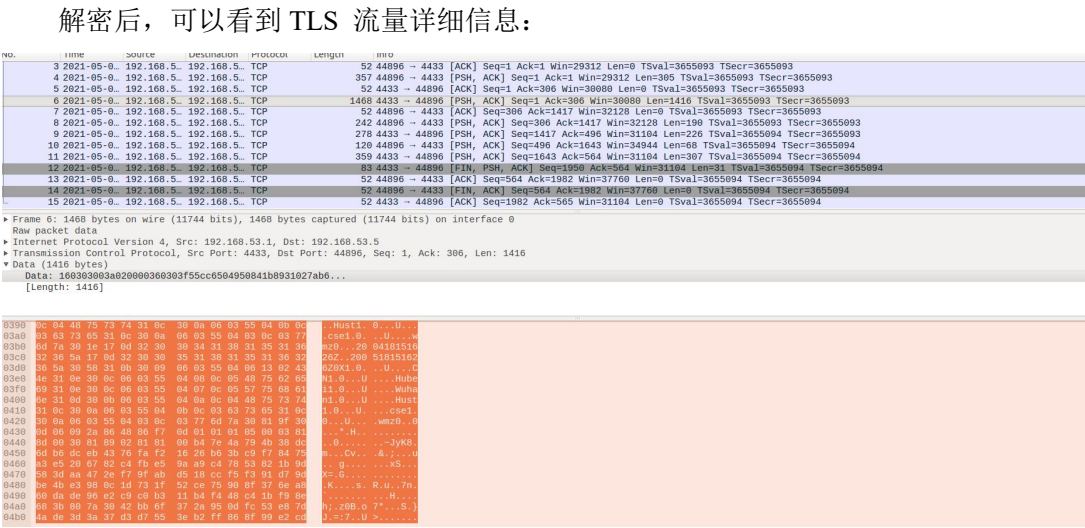


图 4-16 解密后的 wireshark 截包图



#### 4.4 任务 4：认证 VPN 服务器

使用 OpenSSL 来创建证书必须有一个配置文件, 将该文件直接复制到当前文件夹后, 需要根据配置文件中的说明创建多个子目录

```
root@VM: /home/seed/Desktop/exp3/MyCA/demoCA 80x17
root@VM:/home/seed/Desktop/exp3/MyCA# ls
root@VM:/home/seed/Desktop/exp3/MyCA# cp /usr/lib/ssl/openssl.cnf openssl.cnf
root@VM:/home/seed/Desktop/exp3/MyCA# ls
openssl.cnf
root@VM:/home/seed/Desktop/exp3/MyCA# gedit openssl.cnf

** (gedit:14342): WARNING **: Set document metadata failed: Setting attribute me
tadata::gedit-position not supported
root@VM:/home/seed/Desktop/exp3/MyCA# mkdir demoCA
root@VM:/home/seed/Desktop/exp3/MyCA# cd demoCA/
root@VM:/home/seed/Desktop/exp3/MyCA/demoCA# ls
root@VM:/home/seed/Desktop/exp3/MyCA/demoCA# mkdir certs
root@VM:/home/seed/Desktop/exp3/MyCA/demoCA# mkdir crl
root@VM:/home/seed/Desktop/exp3/MyCA/demoCA# touch index.txt
root@VM:/home/seed/Desktop/exp3/MyCA/demoCA# mkdir newcerts
root@VM:/home/seed/Desktop/exp3/MyCA/demoCA# gedit serial
```

图 4-17 生成 CA 的目录配置图

运行以下命令为 CA 生成自签名证书:

\$ openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf

```
root@VM:/home/seed/Desktop/exp3/MyCA# openssl req -new -x509 -keyout ca.key -out
ca.crt -config openssl.cnf
Generating a 2048 bit RSA private key
..+++
.....
.....+++
writing new private key to 'ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CN
State or Province Name (full name) [Some-State]:Hubei
Locality Name (eg, city) []:WUHAN
Organization Name (eg, company) [Internet Widgits Pty Ltd]:HUST
Organizational Unit Name (eg, section) []:Cyber Science And Engineering
Common Name (e.g. server FQDN or YOUR name) []: shexinyu
Email Address []:304827134@qq.com
root@VM:/home/seed/Desktop/exp3/MyCA# ls
ca.crt ca.key demoCA openssl.cnf
```

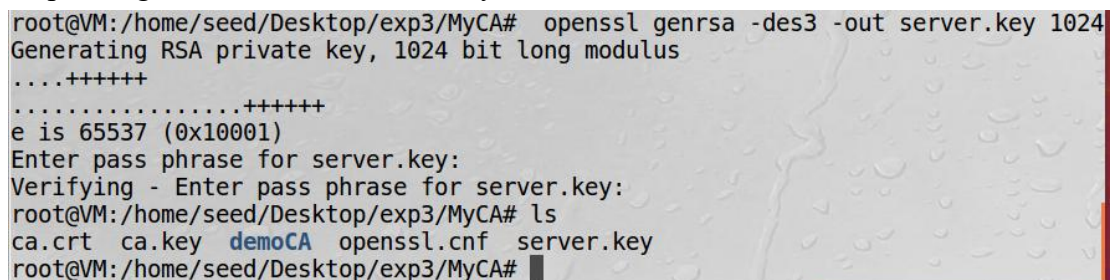
图 4-18 生成 CA 图

Pass phrase : shexinyu

服务器端

创建一对公钥和私钥，在服务器端执行以下命令获得 RSA 密钥对，还需要我们自己提供一个密码来保护密钥，密钥将会存储在 server.key 文件里面。

```
$ openssl genrsa -des3 -out server.key 1024
```



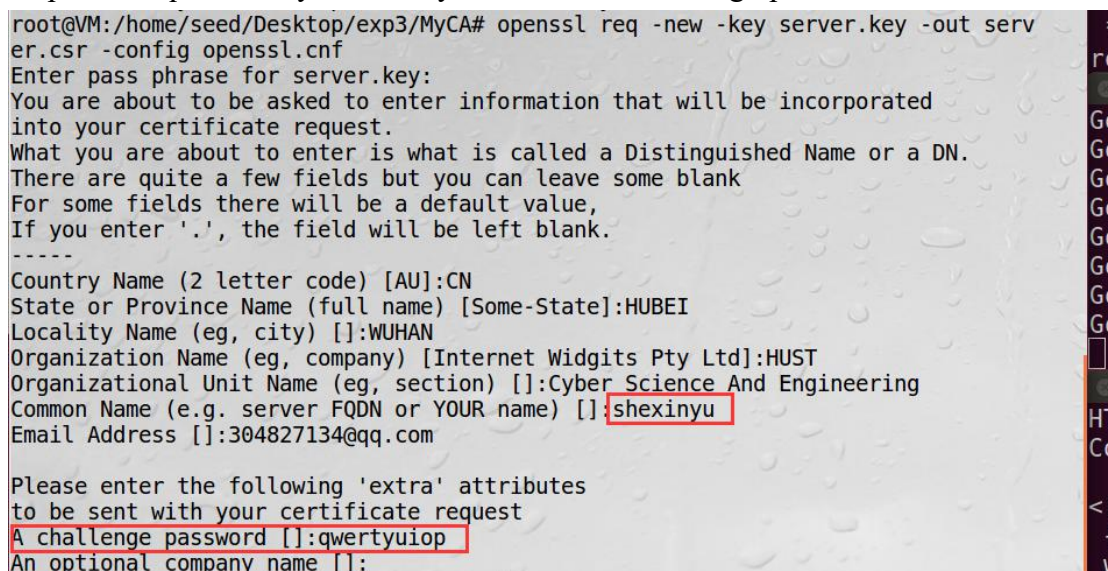
```
root@VM:/home/seed/Desktop/exp3/MyCA# openssl genrsa -des3 -out server.key 1024
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
root@VM:/home/seed/Desktop/exp3/MyCA# ls
ca.crt ca.key demoCA openssl.cnf server.key
root@VM:/home/seed/Desktop/exp3/MyCA#
```

图 4-19 创建 RSA 密钥对

Pass phrase :9635741

生成证书签名请求（CSR）。CSR 将发送给 CA，CA 将为密钥生成证书（通常在确保 CSR 中的身份信息与服务器的真实身份匹配之后）。

```
$ openssl req -new -key server.key -out server.csr -config openssl.cnf
```



```
root@VM:/home/seed/Desktop/exp3/MyCA# openssl req -new -key server.key -out server.csr -config openssl.cnf
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CN
State or Province Name (full name) [Some-State]:HUBEI
Locality Name (eg, city) []:WUHAN
Organization Name (eg, company) [Internet Widgits Pty Ltd]:HUST
Organizational Unit Name (eg, section) []:Cyber Science And Engineering
Common Name (e.g. server FQDN or YOUR name) []:shexinyu
Email Address []:304827134@qq.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:qwertyuiop
An optional company name []:
```

图 4-20 生成服务器 CSR

Challenge password : qwertyuiop

## 客户端

客户端可以按照以下相似的命令来生成 RSA 密钥对和 CSR，与服务器同理

```
openssl genrsa -des3 -out client.key 1024
```

Pass phrase : 1475369

```
$ openssl req -new -key client.key -out client.csr -config openssl.cnf
```



```

root@VM:/home/seed/Desktop/exp3/VPN/client# openssl req -new -key client.key -out client.csr -c
onfig ../openssl.cnf
Enter pass phrase for client.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CN
State or Province Name (full name) [Some-State]:HUBEI
Locality Name (eg, city) []:WUHAN
Organization Name (eg, company) [Internet Widgits Pty Ltd]:HUST
Organizational Unit Name (eg, section) []:CLIENT
Common Name (e.g. server FQDN or YOUR name) []:SXY
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:poiuytrewq
An optional company name []:

```

图 4-21 生成客户端 CSR

Challenge password : poiuytrewq

## 生成证书

使用可信 CA 来分别为 client 和 server 的 CSR 文件签名，生成证书：  
服务器 server.crt :

```
openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config
openssl.cnf
```

```

root@VM:/home/seed/Desktop/exp3/VPN# openssl ca -in server.csr -out server.crt -cert ca.crt -ke
yfile ca.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 4096 (0x1000)
  Validity
    Not Before: May 20 12:58:10 2021 GMT
    Not After : May 20 12:58:10 2022 GMT
  Subject:
    countryName           = CN
    stateOrProvinceName   = HUBEI
    organizationName      = HUST
    commonName            = SHEXINYU
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:FALSE
    Netscape Comment:
      OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
      97:F5:42:E5:1F:BE:F0:62:45:9C:53:2D:CF:98:35:8F:14:E2:68
    X509v3 Authority Key Identifier:
      keyid:A9:8F:46:BC:3E:2A:D6:03:FF:4D:D0:B4:48:E2:C2:FD:B1:42:C3:23

Certificate is to be certified until May 20 12:58:10 2022 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

```

图 4-22 生成服务器证书

客户端 client.crt :

```
openssl ca -in client.csr -out client.crt -cert ca.crt -keyfile ca.key -config
openssl.cnf
```

```

root@VM:/home/seed/Desktop/exp3/VPN# openssl ca -in client.csr -out client.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 4097 (0x1001)
  Validity
    Not Before: May 20 13:14:36 2021 GMT
    Not After : May 20 13:14:36 2022 GMT
  Subject:
    countryName           = CN
    stateOrProvinceName   = HUBEI
    organizationName       = HUST
    organizationalUnitName = CLIENT
    commonName             = SXY
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:FALSE
    Netscape Comment:
      OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
      E5:62:00:2D:45:A3:DF:F0:38:F9:32:25:F2:BA:A4:76:66:AA:46:09
    X509v3 Authority Key Identifier:
      keyid:A9:8F:46:BC:3E:2A:D6:03:FF:4D:D0:B4:48:E2:C2:FD:B1:42:C3:23

```

图 4-23 生成客户端证书

之后配置文件，如图：

```

root@VM:/home/seed/Desktop/exp3/multivpn# tree
.
├── ca.crt
├── cli
├── client.cpp
├── client.crt
├── client.key
├── Makefile
├── README
├── serv
├── server.cpp
├── server.crt
└── server.key

```

图 4-24 配置文件目录图

#### 4.5 任务 5：认证 VPN 客户端

认证成功，服务器端的认证也在这里，在输入密码以后，即判断服务器是否合法，已经完成认证

```

root@HostU2:/multivpn
1
SSL connection is successful
SSL connection using AES256-GCM-SHA384
Got 278 chars: 'HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE html><html><head><title>Hello World</title></head><style>body
{background-color: black}h1 {font-size:3cm; text-align: center; color:
white;text-shadow: 0 0 3mm yellow}</style></head><body><h1>Hello, worl
d!</h1></body></html>'
SIOCADDRTR: File exists
Got 48 byte from the socket client `
Got 48 byte from the socket_client `
Got 48 byte from the socket_client `

```

```

LOOPBACK RUNNING MTU:65536 Metric:1
packets:0 errors:0 dropped:0 overruns:0 frame:0
root@VM:/home/seed/Desktop/exp3/multivpn
root@VM:/home/seed/Desktop/exp3/multivpn# ./server
Enter PEM pass phrase:
Connection from 602000a, port b6aa
#Login name: seed
Passwd: dees
客户端验证成功！
SSL connection using AES256-GCM-SHA384
SSL connection established!
服务器验证成功！
Received: Hello World!
net.ipv4.ip forward = 1
Got 48 byte from the tun_server: bfe03efc
Got 48 byte from the tun_server: bfe03efc
Got 48 byte from the tun_server: bfe03efc

```

图 4-25 客户端、服务器认证成功图

该认证通过 `getspnam()` 从 `shadow` 文件中获取给定用户的帐户信息，包括散列密码。然后，它使用 `crypt()` 来散列给定的密码，并查看结果是否与从 `shadow` 文件中获取的值匹配。如果是，则用户名和密码匹配，并且验证成功。

实现部分代码如下图：

```
int login(char *user, char *passwd)
{
    printf("Login name: %s\n", user); // 用户登录名
    printf("Passwd: %s\n", passwd); // 加密口令
    struct spwd *pw; // shadow文件的结构体
    char *epasswd;
    pw = getspnam(user); // 通过用户名获取单个用户的spwd信息
    if (pw == NULL) {
        return -1;
    }
    epasswd = crypt(passwd, pw->sp_pwdp); // 对passwd进行加密
    if (strcmp(epasswd, pw->sp_pwdp)) {
        return -1;
    }
    return 1;
}
```

图 4-26 认证代码图

HostU telnet 192.168.60.101(HostV)，可以看到接收到包，而且 wireshark 在 internet (10.0.2.0/24) 可以截到包

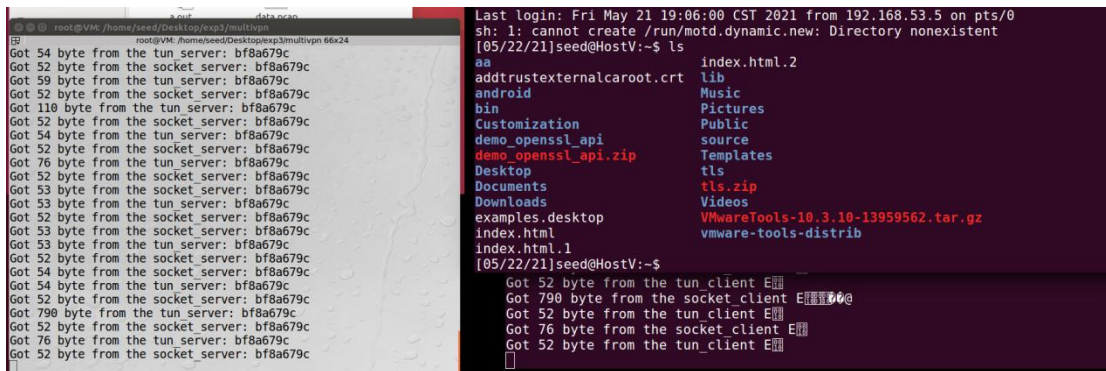


图 4-27 通过 VPN 成功通讯图



图 4-28 wireshark 截包中对应 TCP 流图

#### 4.6 任务 6：支持多个客户端

在 internet 中新建容器 HostU2，同样操作，也可以在已经与 HostU 连接的服务器进行通信，wireshark 中也可以截到包，可以看到 HostU 和 HostU2 可以同时 ping HostV，且 wireshark 中可以截到对应加密报文



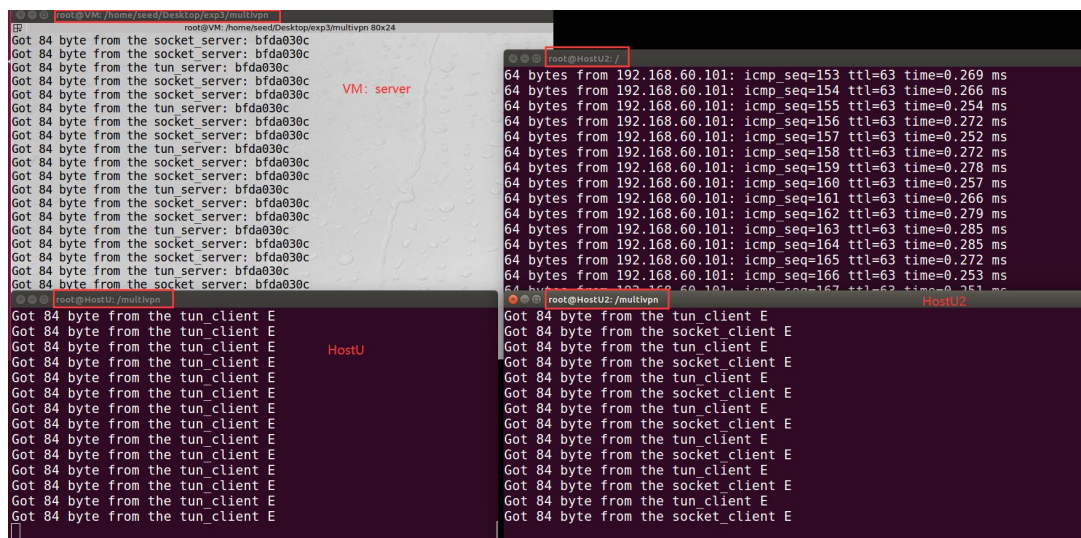


图 4-29 多客户端

可以在 10.0.2.0/24 的 internet 中截到对应包，其中：

HostU: 10.0.2.7

HostU2: 10.0.2.6

VM: 10.0.2.8

No.	Time	Source	Destination	Protocol	Length	Info
1	2021-05-25 01:04:25.5765450	10.0.2.7	10.0.2.8	TLSv1.2	181	Application Data
2	2021-05-25 01:04:25.5765450	10.0.2.7	10.0.2.8	TCP	181	[TCP Retransmission] 36872 → 1111 [PSH, ACK] Seq=1 Ack=114 Win=301 Len=113 TSval=43760 TSecr=43504
3	2021-05-25 01:04:25.5765501	10.0.2.8	10.0.2.7	TCP	68	1111 → 36872 [ACK] Seq=1 Ack=114 Win=243 Len=0 TSval=43760 TSecr=43760
4	2021-05-25 01:04:25.5765503	10.0.2.8	10.0.2.7	TCP	68	[TCP Dup ACK 3#2] 1111 → 36872 [ACK] Seq=1 Ack=114 Win=243 Len=0 TSval=43760 TSecr=43760
5	2021-05-25 01:04:25.6482987	10.0.2.6	10.0.2.8	TLSv1.2	181	Application Data
6	2021-05-25 01:04:25.6482987	10.0.2.6	10.0.2.8	TCP	181	[TCP Retransmission] 43610 → 1111 [PSH, ACK] Seq=1 Ack=1 Win=301 Len=113 TSval=43776 TSecr=43520
13	2021-05-25 01:04:25.6484388	10.0.2.8	10.0.2.6	TLSv1.2	181	Application Data
14	2021-05-25 01:04:25.6484317	10.0.2.8	10.0.2.6	TCP	181	[TCP Retransmission] 1111 → 43610 [PSH, ACK] Seq=1 Ack=114 Win=243 Len=113 TSval=43776 TSecr=43776
15	2021-05-25 01:04:25.6484384	10.0.2.6	10.0.2.8	TCP	68	43610 → 1111 [ACK] Seq=114 Ack=114 Win=301 Len=0 TSval=43776 TSecr=43776
16	2021-05-25 01:04:25.6484384	10.0.2.6	10.0.2.8	TCP	68	[TCP Dup ACK 1#1] 43610 → 1111 [ACK] Seq=114 Ack=114 Win=301 Len=0 TSval=43776 TSecr=43776
17	2021-05-25 01:04:26.8088843	10.0.2.7	10.0.2.8	TLSv1.2	181	Application Data
18	2021-05-25 01:04:26.8088843	10.0.2.7	10.0.2.8	TCP	181	[TCP Retransmission] 36872 → 1111 [PSH, ACK] Seq=1 Ack=1 Win=301 Len=113 TSval=44010 TSecr=43760
19	2021-05-25 01:04:26.8089150	10.0.2.8	10.0.2.7	TCP	68	1111 → 36872 [ACK] Seq=1 Ack=227 Win=243 Len=0 TSval=44010 TSecr=44010

图 4-30 多客户端 internet 截包

## 5 实验思考

(实验指导手册中的思考题)

---

## 心得体会与建议

### 1 心得体会

### 2 建议