

“计算机网络安全”实验报告

**题目： VPN实验**

院 系

专业班级

姓 名

学 号

日 期

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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](#_Toc70453539)

[**1 实验目的** 1](#_Toc70453540)

[**2 实验环境** 1](#_Toc70453541)

[**3实验内容** 2](#_Toc70453542)

[**4 实验步骤及结果分析** 5](#_Toc70453543)

[**5 实验思考** 14](#_Toc70453544)

[心得体会与建议 16](#_Toc70453545)

[**1 心得体会** 16](#_Toc70453546)

[**2 建议** 17](#_Toc70453547)

# 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：

（1）虚拟专用网络

（2）TUN/TAP 和 IP 隧道

（3）路由

（4）公钥加密，PKI 和 X.509 证书

（5）TLS/SSL 编程

（6）身份认证

## **2 实验环境**

VMware Workstation 虚拟机。

Ubuntu 16.04 操作系统（SEEDUbuntu16.04）。

本次实验需要使用 openssl 软件，该软件包含头文件，库函数和命令。该软件包已经安装在我们上述 VM 镜像中。

实验中需要模拟VPN客户端、VPN网关和内网主机，实验的网络拓补图如图2-1所示。

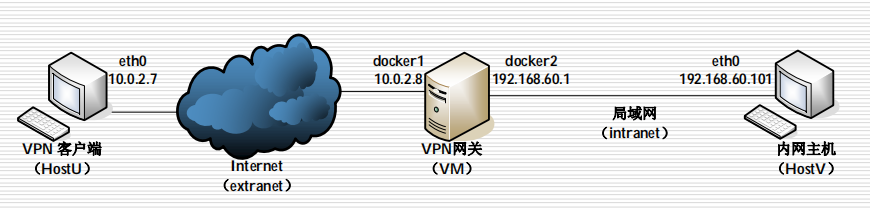


图2-1 网络拓补图

网络配置：VPN服务器（网关）——双网卡，用VM自身实现，docker1为10.0.2.8，docker2为192.168.60.1，

HostU、HostV分别用2个docker容器实现，IP设置为10.0.2.7和192.168.60.101

需要建立2个docker网络extranet（模拟Internet，10.0.2.0/24网段）、intranet（模拟局域网，192.168.60.0/24网段）

## **3实验内容**

本次实验，我们主要在linux操作系统上完成一个简单的VPN，把其称为miniVPN，需要实现的工作有：环境配置、隧道创建、加密隧道、VPN服务器端验证、VPN客户端验证和多客户端的支持。

### **3.1环境配置**

实验中要求我们在计算机（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。我们首先在SEEDUbuntu中创建两个容器，分别用作客户机HostU和VPN内网的主机HostV，使用docker网络“extranet”模拟Internet，连接客户机和VPN局域网的网关，然后使用docker网络“intranet”模拟局域网，将HostV 与 VPN 服务器网关的内网口连接，模拟内部局域网，使得HostV不能被直接通过Internet访问。

在创建完网络配置以及终端后，在容器 HostU 和 HostV 内分别删除掉默认路由，重定向其数据包的默认发送地址，使得隧道未建立时，外部局域网不能和内部网络通信。

### **3.2创建隧道**

TLS/SSL VPN 中使用了 TUN/TAP 技术，即虚拟网络内核驱动程序，它们可以实现完全由软件支持的网络设备。TAP 模拟以太网设备，处理的是以太网帧等二层数据包；TUN 模拟网络层设备，处理的是 IP 等三层数据包。我们可以用 TAP/TUN 创建虚拟网络接口。用户空间程序通常访问 TUN/TAP 虚拟网络接口。操作系统通过 TUN/TAP网络接口将数据包传送到用户空间程序。另一方面，程序通过 TUN/TAP 网络接口发送的数据包被注入操作系统网络栈；在操作系统看来，数据包是通过虚拟网络接口的外部源进来的。

实验中数据包的流动过程如图3-1所示。vpnclient 和 vpnserver 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。客户端和服务器之间的虚线描述了 VPN 隧道的路径。VPN 客户端和服务器程序通过 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。

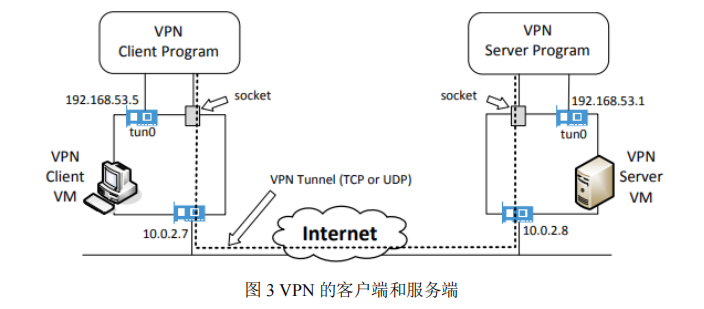


图3-1 VPN的客户端和服务器

### **3.3加密隧道**

建立好IP隧道之后，需要为隧道添加数据传输保护才能成为合格的VPN隧道。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。

我们使用传输层协议TLS实现数据传输的机密性和完整性，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，就可以实现隧道的加密。

### **3.4认证VPN服务器**

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，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）（例如 Verisign）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，我们的工作是生成自己的CA，并使用openssl中的函数以完成证书认证工作。

### **3.5认证客户端**

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。最后一步就是在server程序端对客户机进行认证，只允许授权用户与 VPN 服务器建立 VPN 隧道我们将使用标准密码身份验证来验证用户身份。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。

### **3.6多客户端**

为了模拟现实中的多个客户端同时使用VPN服务器的情况，需要我们实现多个客户端同时访问服务器的处理。VPN服务器应当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。

为实现多客户端通信，可以选择进程间通信或者划分虚拟子网的方法，服务器每次创建一个TCP连接，就使用fork函数创建一个子进程，使该子进程维持该连接。而父进程继续监听准备进行下一个TCP连接。

当数据包来自隧道时，其相应的子进程将获取数据包，并将其转发到 TUN 接口。当数据包到达 TUN 接口（来自专用网络）时，父进程将获取数据包，再确定该数据包应该到达哪个隧道。在实验中我们使用虚拟子网的方式，每个客户端跟服务器建立隧道以后，服务器给每个客户端分配一个/30的虚拟子网（一个虚地址给客户端tun接口，一个地址给服务器端的虚接口）,服务器为每个隧道启动1个tun接口，这样服务器就会有多个tun接口，tun0、tun1....，每个tun接口上会有对应的虚拟子网路由，内网主机返回的报文，直接根据目的地址(虚IP地址）查找路由就可以找到对应的tun接口。

## **4 实验步骤及结果分析**

**4.1 配置环境**

本实验要求Internet上的主机 HostU通过 VPN 隧道与内部局域网的主机 HostV 通信实验，因此需要将外网的VPN 客户端（HostU）和VPN 服务器连接到同一docker网络“extranet”，模拟 Interne，同时使用 docker 网络“intranet”将 HostV 与 VPN 服务器网关的内网口连接，模拟内部局域网。使用docker network create命令分别创建docker网络 “extranet”和“intranet”。配置的命令如下：

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

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

接下来创建两个容器来模拟HostU和HostV，为HostU指定网络为extranet，指定IP为10.0.2.7；为HostV指定网络为intranet，指定IP为192.168.60.101；如图4-1所示，可见容器创建并配置成功。创建完成后使用route del default命令在HostU和HostV上删除默认路由，阻止外部网和局域网的直接通信。

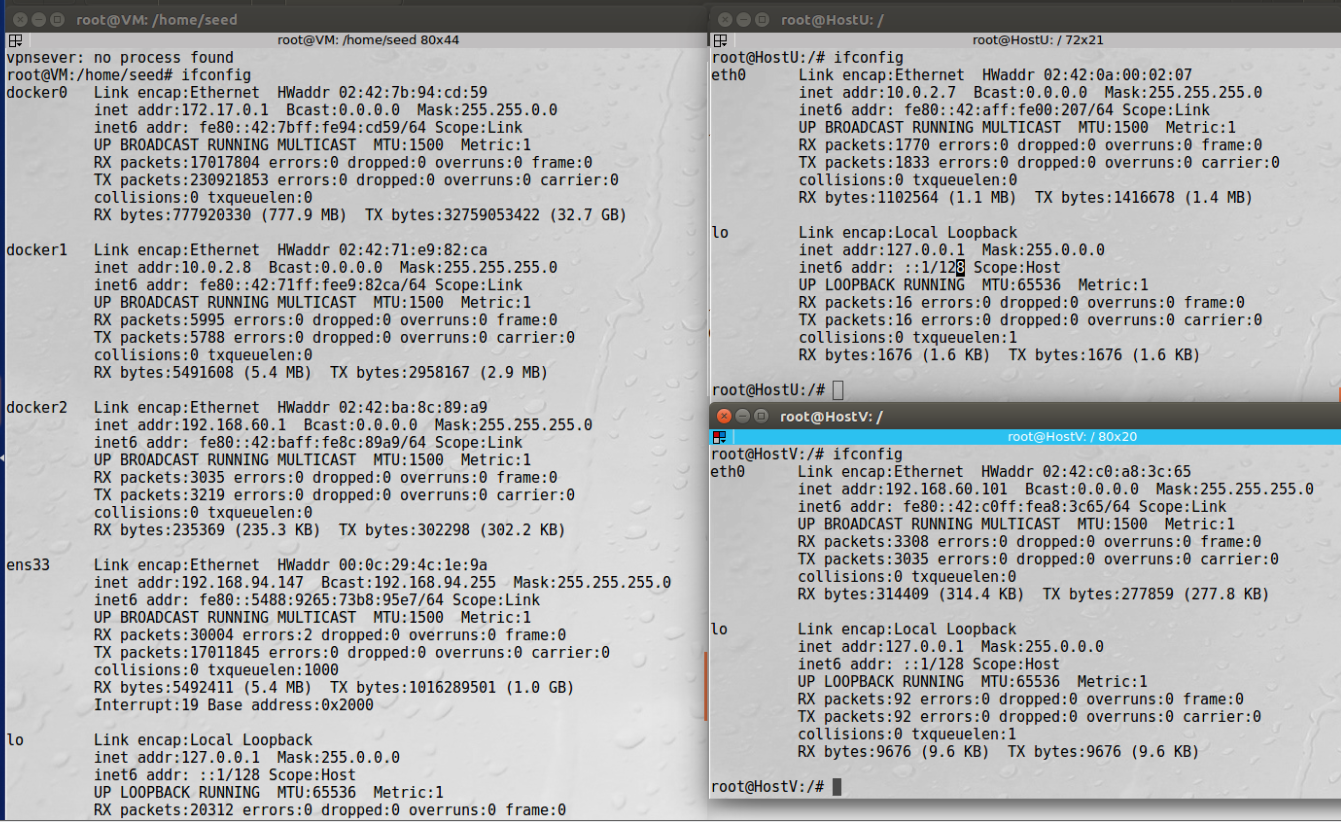


图4-1 环境配置

**4.2主机到主机的隧道**

（1）编写VPN客户端和服务器程序

首先编写客户端程序，客户端程序的流程图如图4-2所示。是使用函数open("/dev/net/tun", O\_RDWR) 创建TUN接口，返回一个文件描述符tunfd，然后使用ioctl(tunfd, TUNSETIFF, &ifr)调用系统调用创建TUN接口，其中ifr结构中包含接口名等信息。

当程序从 TUN接口读取数据时，计算机发送到此接口的 IP 数据包将被传送给程序；另一方面，程序发送到此接口的 IP 数据包将被传送到计算机中，当做是通过这个虚拟网络接口从外部来的。程序中使用标准的read()和 write()系统调用来接收或发送数据包到虚拟接口。

为了完成数据的发送，要实现对不同接口的监听。VPN服务器程序的流程图和客户端程序大致相同，不同的是在创建连接时要使用accept函数监听，如果监听到一个连接，就创建一个新的sock用于会话。



图4-2 创建简单IP隧道的客户端程序流程图

（2）运行VPN客户端和服务器端程序并配置路由

首先，在服务器VM上运行VPN服务器程序vpnserver，程序运行后，系统中将出现一个虚拟 TUN 网络接口tun0。新开一个终端，进行ip地址的配置。命令为sudo ifconfig tun0 192.168.53.1/24 up。

VPN Server 需要在内网和隧道之间转发报文，因此需要作为网关。我们需要为计算机启用 IP 转发，使其行为类似于网关。由于 VM 上的 iptables 规则可能阻断转发报文，我们还需要清除 iptables 规则。VPN服务器上的配置如图4-3所示。

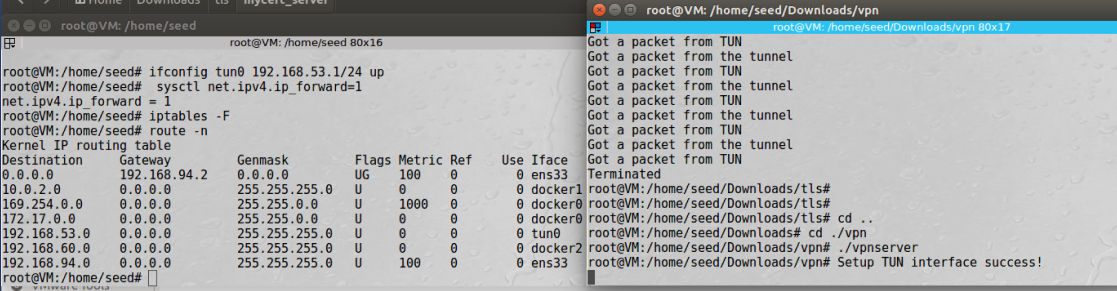


图4-3 VPN服务器配置

将VPN客户端程序vpnclient复制到容器HostV里，启动VPN客户端，连接到10.0.2.8上运行的服务器程序，在容器 HostU 的另一个终端 tun0 虚拟 IP 地址并激活接口。在 HostU 上创建隧道路由，将所有进入专用网络（192.168.60.0/24）的数据包定向到 tun0 接口。同样在VPN服务器和内网主机HostV上也进行相应的路由配置，发送应答的数据包。HostV收到的IP包的源IP是192.168.53.5，因此回复时的目的IP也是192.168.53.5，由于不在同一网段下，需要配置路由，让目的地址是192.168.53.0/24网段的数据包路由到网关。HosTV上配置的命令为：

sudo route add -net 192.168.53.0/24 gw 192.168.60.1

配置完成后结果如图4-4所示。

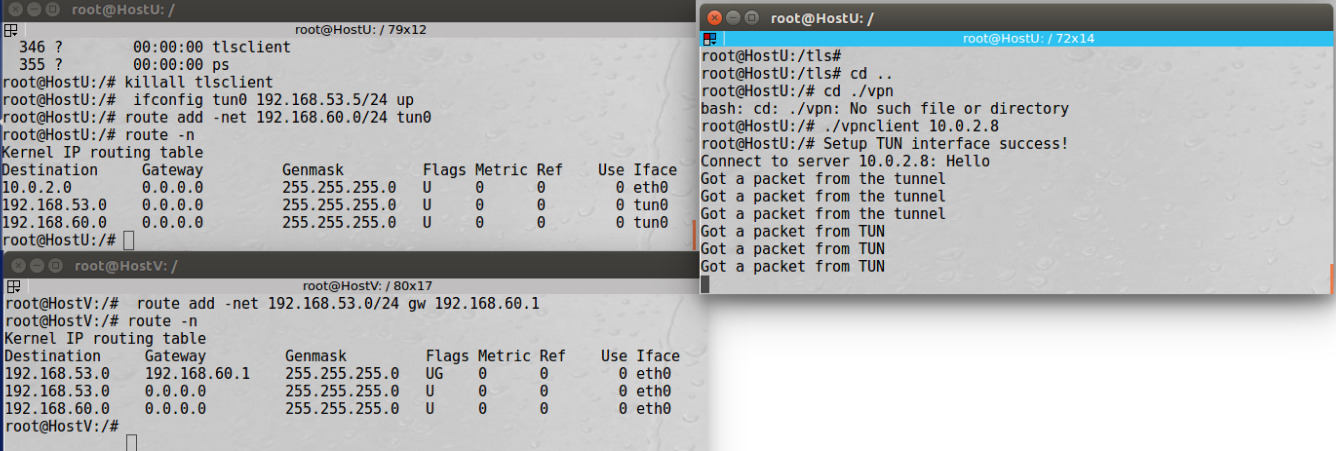


图4-4 HostU和HostV配置

（3）隧道测试

此时我们已经完成了隧道的配置，为了完成隧道的测试，我们需要借助ping和telnet来观察是否有HostU到HostV的连接。

实验的镜像中没有开启telnet服务，因此需要我们自行安装。安装telnet服务的命令如下：

apt-get install openbsd-inetd

sudo apt-get install telnetd

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

sudo netstat -a | grep telnet

在内网主机上开启telnet服务后，测试外网主机HostU是否能与内网主机HostV建立连接。分别使用ping命令和telnet命令进行测试，结果如图4-5所示。

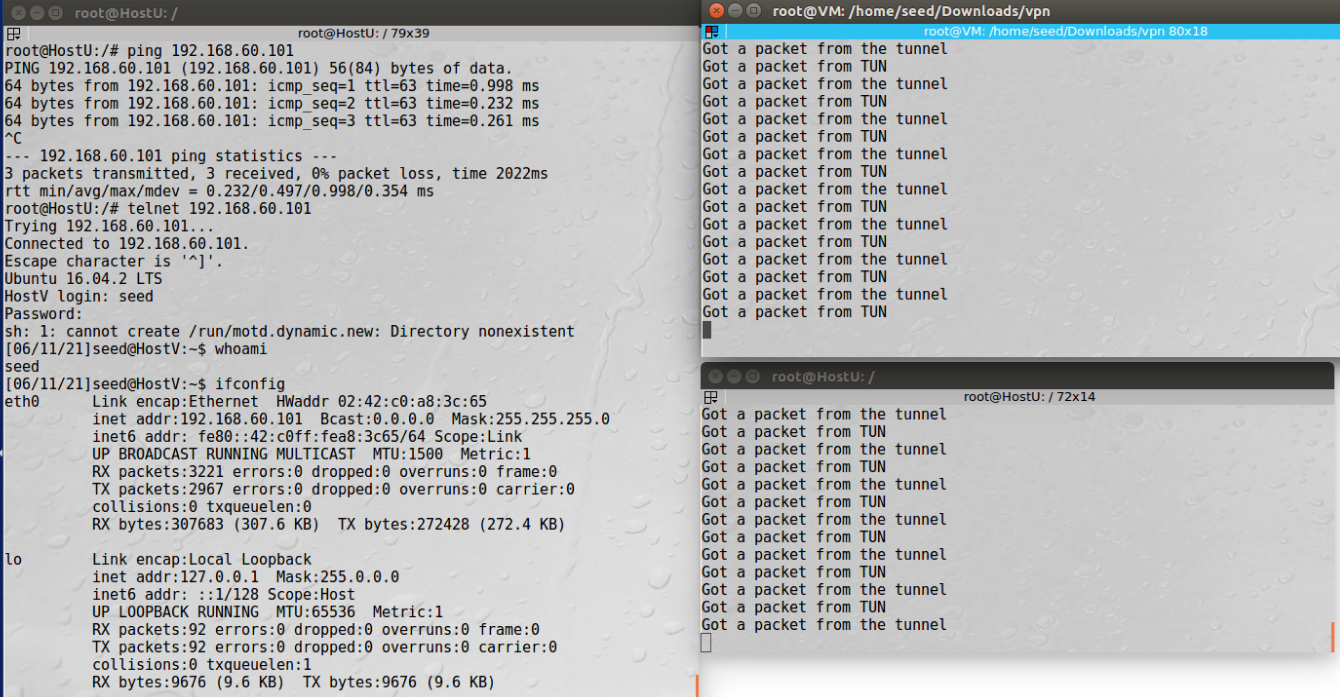


图4-5 隧道测试

如图所示，HostU与HostV成功进行了ping测试和telnet连接，表明隧道建立成功。

**4.2 隧道加密**

为了保护我们建立的隧道的机密性和完整性，实验中需要使用传输层协议（TLS）。TLS 通常建立在 TCP 之上。主要通过以下步骤完成：

（1）将样例vpn程序里的UDP连接替换成TCP连接；

（2）在之前代码的基础上需要增加一个setupTLSClient函数，使用openssl的一些API函数进行TLS连接的初始化操作，最终返回一个SSL类型的指针ssl。

（3）使用ssl\_read替换原来代码中的recv函数，用SSL\_write替换send函数，这两个函数内部封装了recv和send函数，在完成数据发送和接收的同时，起到加解密的作用，实现机密性目标。

（4）在创建TCP连接之后使用SSL\_set\_fd(ssl, sockfd)将ssl和TCP连接绑定，然后使用SSL\_connect(ssl)进行TLS握手。

在完成上述代码的修改后，分别在服务器端和客户端运行vpn程序。此时，数据的流动就发生改变。我们使用的SSL\_write函数会让数据加密后再通过eth1发出，而SSL\_read接收到数据之后，先进行解密，再将解密后的内容通过tun0发出。在完成隧道加密后，启用wireshark监听，在内部网卡上可以监听到明文，在外部网卡上监听隧道，得到的TLS数据报就是加密后的内容，如图4-6所示。

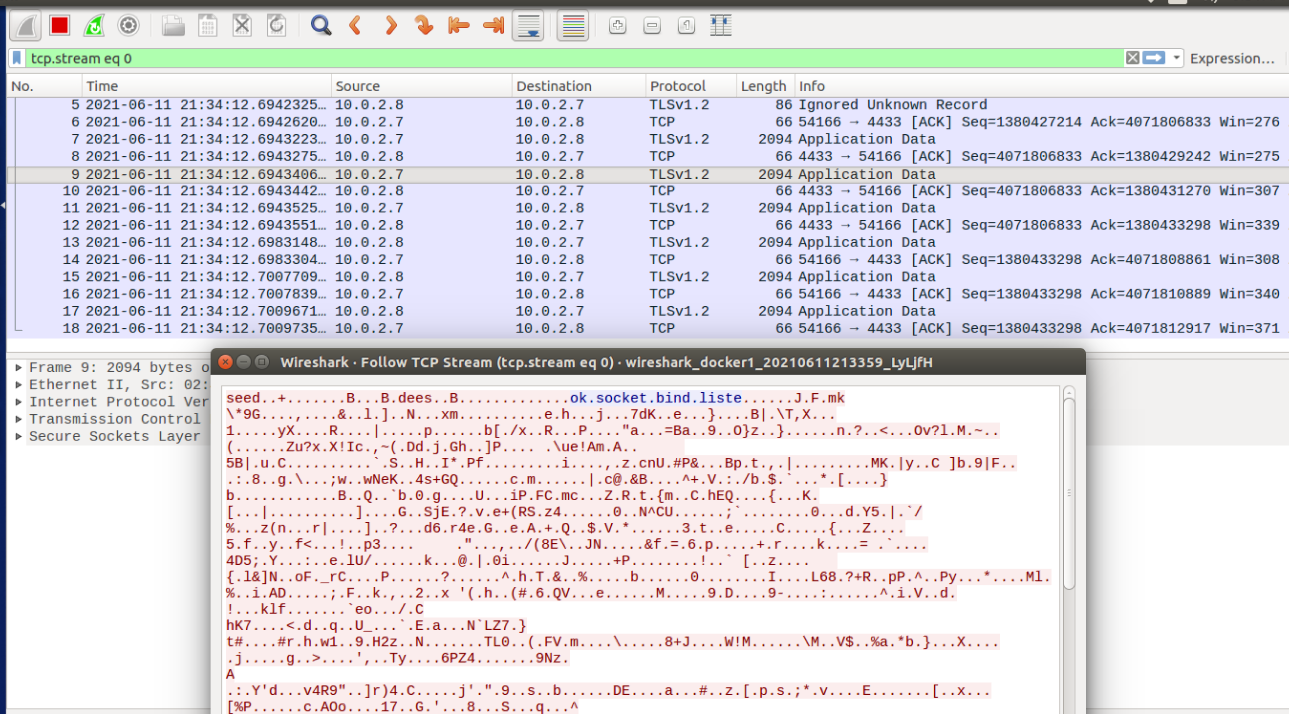


图4-6 使用wireshark抓包

**4.3 服务器端认证**

（1）证书生成

VPN 客户端必须对 VPN 服务器进行身份认证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。生成服务器证书的主要命令有：

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

openssl genrsa -des3 -out server.key 1024

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

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

客户端证书的生成过程也类似，服务器证书生成后的结果如图4-7所示。

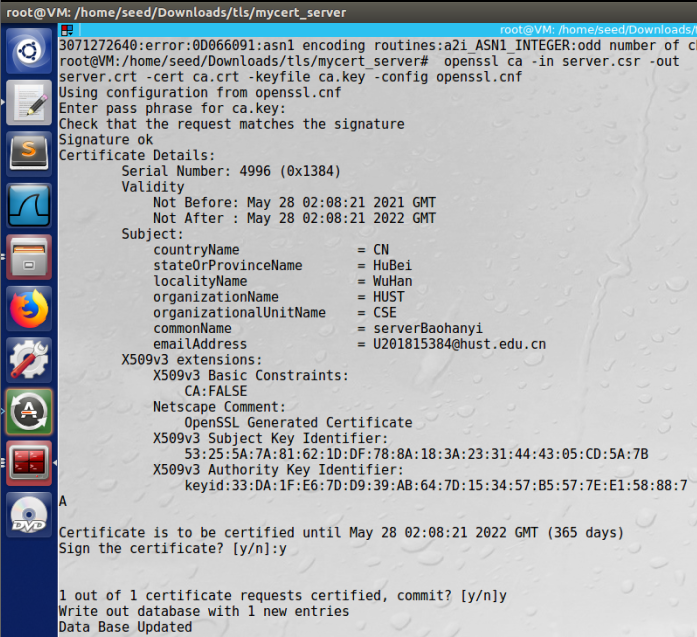


图4-7 服务器证书生成

（2）服务器身份认证

要实现服务器身份认证有三个重要步骤：1）验证服务器证书是否有效2）验证服务器是证书的所有者3）验证服务器是否是目标服务器。

对于步骤1，在服务端程序的TLS初始化函数中增加对本地证书的验证，即加入：SSL\_CTX\_use\_certificate\_file(ctx, CERTF, SSL\_FILETYPE\_PEM)

该函数会根据给定的路径CERTF读取证书文件，如果该证书不是有效的证书文件，则会返回小于0的数，程序中可以设置如果返回值为1则报错。

步骤2，就是验证服务器提供的证书和私钥是否匹配。首先使用SSL\_CTX\_use\_PrivateKey\_file(ctx, KEYF, SSL\_FILETYPE\_PEM)函数加载服务器的私钥，然后使用SSL\_CTX\_check\_private\_key(ctx)函数验证证书和私钥是否匹配即可。

对于步骤3，为了验证服务器是否是目标服务器，需要在客户端程序中对证书加以验证。加入函数SSL\_CTX\_set\_verify(ctx, SSL\_VERIFY\_PEER, verify\_callback)，其中verify\_callback是自定义的回调函数，会显示验证的结果。然后在TLS初始化函数中使用SSL\_CTX\_load\_verify\_locations加载本地的ca证书，以验证服务器证书的有效性。使用X509\_VERIFY\_PARAM\_set1\_host将主机名加入验证参数中。这样就可以在握手阶段完成主机名的验证和服务器证书的验证。

如图4-8所示，是服务器验证成功的结果。对服务器的证书进行替换，用无效证书伪造，则在客户端上的验证不能通过，如图4-9是服务器证书验证失败返回的结果。

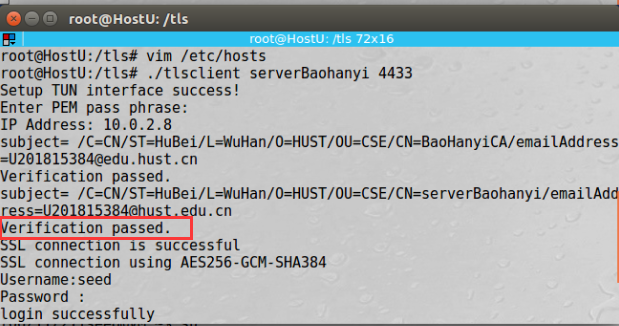


图4-8 服务器验证成功

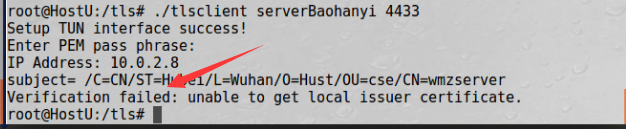


图4-9 服务器证书验证失败

**4.4 客户端认证**

VPN应该只允许授权用户与 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。客户端认证的主要思路如下：

1）当客户端和服务器进行tls握手完成之后，通过了服务器端验证之后，客户端应当提示用户输入账户和密码，然后将用户输入的账户和密码通过SSL\_write发送给服务器。

2）服务器使用SSL\_read函数获得解密后的账户名和密码，然后使用getspnam(username)函数根据用户名获得/etc/shadow中对应用户的口令hash。如果返回值为空，表明没有这个账户，向客户端发送错误信息；如果返回值不为空，则使用计算客户端发送的密码的hash，与函数返回值做比较，相同则验证通过，不一致则同样返回错误信息。

3）客户端程序接收服务器返回的信息，如果验证通过,则开始收发数据。。如果验证不通过则显示对应提示信息。

如图4-10所示，是客户端登录的几种错误结果，表明客户端验证实现成功。

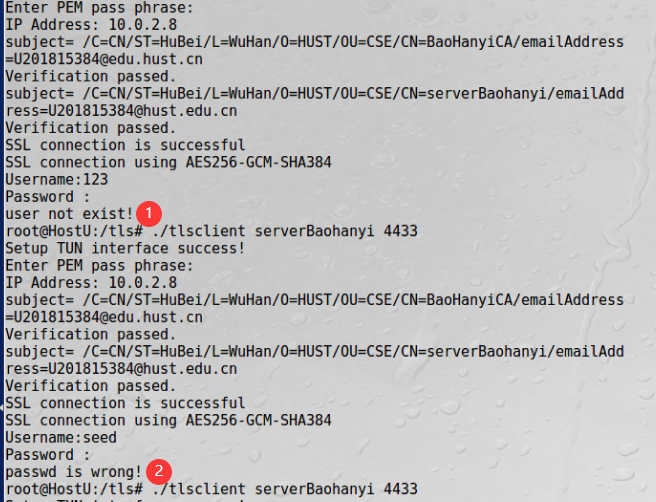


图4-10 客户端登陆结果

**4.5 多客户端支持**

实验中采用子网模式，即每个客户端跟服务器建立隧道以后，服务器给每个客户端分配一个的虚拟子网（一个虚地址给客户端tun接口，一个地址给服务器端的虚接口）,服务器为每个隧道启动1个tun接口，这样服务器就会有多个tun接口，tun0、tun1....，每个tun接口上会有对应的虚拟子网路由，内网主机返回的报文，直接根据目的地址查找路由就可以找到对应的tun接口。在这种模式下，服务器最多能够同时和255个客户端通信。

需要注意的是，对每个外网主机，实验过程中都需要在上面配置对应的路由，将发往HostV的包定向到对应的网卡；同时，每增加一个外网主机，HostV上也要增加一条返回的路由。以外网主机192.168.53.5（HostU）为例，就需要增加对应路由条目：

sudo route add -net 192.168.53.0/24 gw 192.168.60.1

服务器端每监听到一个连接请求，就使用fork函数创建一个子进程用来处理新建的tun和客户端之间的通信。

为了测试多客户端的功能，另外再创建一个docker容器，为其分配地址为10.0.2.6，名称为HostX。设置HostX的tun0的虚拟IP地址为192,.168.54.6，并配置好对应路由，然后在HostX启动VPN客户端程序，连接VPN服务器后，在VPN服务器上打开tun1，设置192.168.54.0/24，在HostV上增加返回的路由。完成后，让HostX与HostU同时ping HostV，如图4-11所示，可以看到两个客户端可以并发进行ping操作，多客户端实现成功。

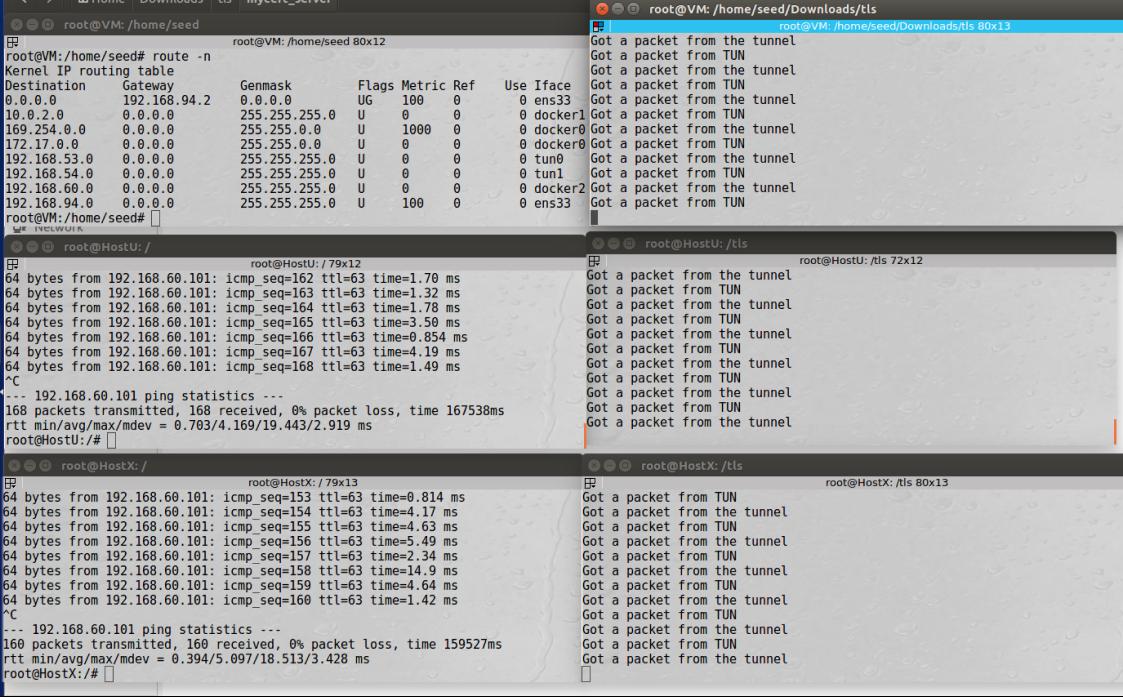


图4-11 多客户端连接

## **5 实验思考**

（1）保持 telnet 连接存活的同时，断开 VPN 隧道。接着在 telnet 窗口中输入内容，报告观察到的内容

在保持 telnet 连接存活的同时，断开 VPN 隧道。然后我们在 telnet 窗口中输入内容，发现并没有什么反应，输入的字符也不会显示。这是因为隧道断开后HostU和HostV 之间无法直接进行通信，因此HostU发出的telnet数据达到不了HostV，也就收不到HostV的回复，因此失去了回显功能，所以就表现为输入没有任何反应。

（2）重新连接 VPN 隧道，需要将 vpnserver 和 vpnclient 都退出后再重复操作的原因

在重新连接 VPN 隧道时，需要将 vpnserver 和 vpnclient 都退出后再重复操作，这是因为vpnserver 和 vpnclient在断开时都处于处理数据或等待接收数据的状态。如果只是断开一个，另一个仍在运行的话，并不能成功建立连接，也就没办法重新搭建隧道。

（3）正确重建隧道后telnet连接的状态

正确重连后，telnet 连接可能会继续，也可能会断开。

如果在断开后没有做任何的数据输入，或是两个主机间的数据传输，那么在重新连接后，telnet连接可以继续。反之，则仍然处于卡死的状态。这是因为，当隧道断开之后，TUN接口也随之消失，相应的路由也消失掉了。此时，如果主机通过Telnet发出数据，由于没有相应的路由，该包会被丢弃掉，而telnet程序仍然在等待回复，否则无法回显。即使此时重新建立隧道，被丢弃的包也不会再被发送目标主机，telnet服务接收不到回复的信息，就一直处于等待状态，也无法回显或输入。反之，如果在隧道断开后没有输入字符或没有进行telnet的数据传输，telnet程序就不会进入等待状态。在隧道重新建立后，系统重新加入路由，通信恢复正常，telnet连接也就可以继续。

# 心得体会与建议

## **1 心得体会**

本次实验完成了一个较基本的VPN程序，实现了外网主机和内网主机之间的数据通信，并解决了隧道加密、客户端认证、服务器认证、多客户端等问题，对VPN相关的知识有了进一步的理解。实验涉及到的东西主要是有计算机网络课程上学到的相关知识，也有证书、TUN、TLS、进程间通信这些更加深入，之前没有实际应用过的东西。总的来说，实验的任务不算太难，但要学习的东西比较多，略显复杂。

为解决外网主机和内网主机的连接问题，我们建立了ssl隧道，设置接口tun用于数据传输，并接借助TLS协议对数据进行加密传输。客户端通过SSL函数进行服务器认证，而服务器对客户端的认证就通过shadow文件进行口令匹配。多客户端的处理通过进程间通信或是虚拟子网来实现。实验的脉络还是比较清晰，实验的过程中，我也更加深入的理解了VPN的功能实现和需求。

实验中同样存在一些难点。首先是对报文处理过程的分析。实验中报文的流动过程，在哪一层进行加密，运输层报文如何构造和路由，这些都在一开始深深的困扰着我。在花费很多时间弄清楚这里的原理后，实验的大致框架就掌握了。其它的问题主要是对工具和相关技术不太熟悉导致的，如证书的生成和相关SSL函数如何进行验证等等，稍显繁琐但通过查找资料和结合老师提供的样例程序最终都成功解决了。比较遗憾的地方是最后多客户端的实现，开始尝试用线程和进程间通信进行实现，但出现了一些端口冲突的问题。后面改用虚拟子网实现，增加一个客户端就增加一个tun接口，过程很是简洁明了，但是存在一些效率方面的问题，实际应用中应该避免。

## **2 建议**

1. 实验中关于openssl相关函数的使用和证书的验证，网上的资料比较杂乱，不太好找，各种参数的意义查阅起来有些麻烦，如何验证也是摸索了半天。如果有些额外的资料可能更有助于我们理解。

2．实验检查起来太过繁琐，有时候一个配置不对就要弄好久，老师们和助教检查起来效率也不高，希望可以设计更合理的检查方案。