

“计算机网络安全”实验报告

——VPN实验

|  |  |
| --- | --- |
| **院 系：** | 网络空间安全学院 |
| **专业班级：** | 网安190x班 |
| **姓 名：** |  |
| **学 号：** |  |
| **日 期：** | 2022.06.12 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **评分项** | **实验报告评分**  **（50%）** | **检查单分数**  **（50%）** | **综合得分** | **教师签名** |
| **得分** |  |  |  |  |

实验报告评分表

|  |  |  |  |
| --- | --- | --- | --- |
| **评分项目** | **分值** | **评分标准** | **得分** |
| 实验原理 | 10 | 10-8：原理理解准确，说明清晰完整  7-5：原理理解基本准确，说明较为清楚  4-0：说明过于简单 |  |
| VPN系统设计 | 25 | 25-19：系统架构和模块划分合理，详细设计说明详实准确  18-11：系统架构和模块划分基本合理，详细设计说明较为详实准确  10-0：系统架构和模块划分不恰当，详细设计说明过于简单 |  |
| VPN实现细节 | 25 | 25-19：文字表达清晰流畅，实现方法技术优良，与设计实现及代码一致  18-11：文字表达较清晰流畅，实现方法一般，与设计实现及代码有偏差  10-0：文字表达混乱，实现方法过于简单 |  |
| 测试结果与分析 | 20 | 20-15：功能测试覆盖完备，测试结果理想，分析说明合理可信  14-9：功能测试覆盖基本完备，测试结果基本达标，分析说明较少  8-0：功能测试覆盖不够，测试未达到任务要求，缺乏分析说明 |  |
| 体会与建议 | 10 | 10-8：心得体会真情实感，意见中肯，建议合理可行，体现了个人的思考  7-5：心得体会较为真实，意见建议较为具体  4-0：过于简单敷衍 |  |
| 格式规范 | 10 | 图、表的说明，行间距、缩进、目录等不规范相应扣分 |  |
| **总 分** | | |  |

目 录

[1 实验原理 1](#_Toc105959647)

[1.1 综述 1](#_Toc105959648)

[1.2 实验环境 1](#_Toc105959649)

[1.2.1 网络拓扑 1](#_Toc105959650)

[1.2.2 实验平台 2](#_Toc105959651)

[1.3 通信机制 2](#_Toc105959652)

[1.4 加密原理 4](#_Toc105959653)

[1.5 认证机制 5](#_Toc105959654)

[2 VPN系统设计 7](#_Toc105959655)

[2.1 概要设计 7](#_Toc105959656)

[2.1.1 系统应用架构 7](#_Toc105959657)

[2.1.2 模块划分 8](#_Toc105959658)

[2.1.3 工作机制 8](#_Toc105959659)

[2.2 详细设计 10](#_Toc105959660)

[2.2.1 建立TLS隧道 10](#_Toc105959661)

[2.2.2 初始化TCP连接 11](#_Toc105959662)

[2.2.3 创建虚拟专用网络 12](#_Toc105959663)

[2.2.4 身份认证 13](#_Toc105959664)

[2.2.5 监听转发 14](#_Toc105959665)

[2.2.6 客户端线程管理 15](#_Toc105959666)

[3 VPN实现细节 15](#_Toc105959667)

[4 测试结果与分析 17](#_Toc105959668)

[4.1 认证VPN服务器 18](#_Toc105959669)

[4.2 认证VPN客户端 18](#_Toc105959670)

[4.3 加密隧道通信 19](#_Toc105959671)

[4.4 支持多客户端 20](#_Toc105959672)

[4.5 易用性和稳定性 21](#_Toc105959673)

[5 体会与建议 22](#_Toc105959674)

[5.1 心得体会 22](#_Toc105959675)

[5.2 意见建议 23](#_Toc105959676)

# 实验原理

综述

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TLS/SSL VPN。

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；

实验环境

网络拓扑

实验的网络拓扑如图1-1所示。

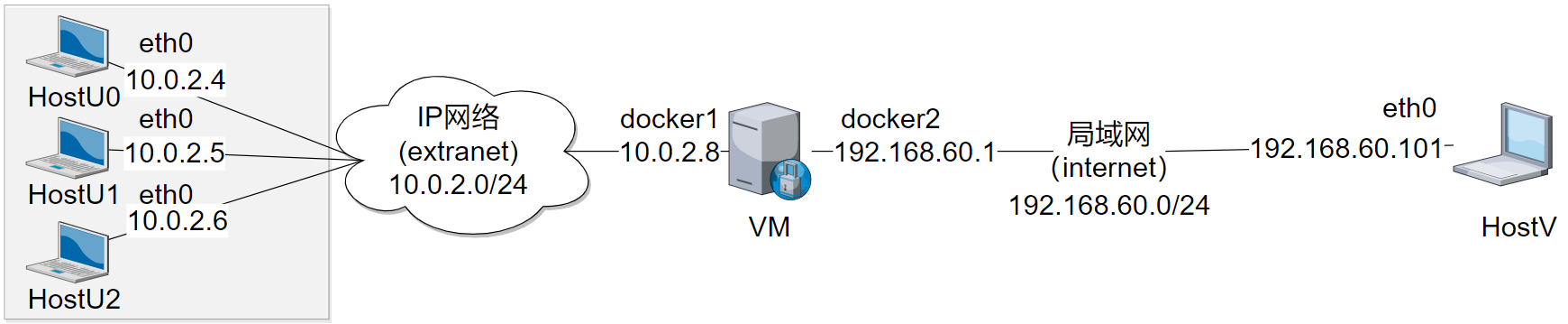


图1-1 实验的网络拓扑

HostU是外网主机，代表一系列运行VPN客户端的主机，在网段10.0.2.0/24。实验开启3个VPN客户端容器，分别为HostU0、HostU1和HostU2。

VPN网关上运行着VPN服务端，其在外网extranet上的IP地址为10.0.2.8，在内网intranet的地址为192.168.60.1。

HostV代表所有内网主机，在网段192.168.60.0/24。

实验平台

实验使用的虚拟机是SEEDUbuntu16.04，上图中VPN网关为VMware虚拟机，其余的主机均运行在docker中。我们需要创建两个网卡、创建4个docker，并将所有docker的默认路由删除，在VM上开启IP转发、关闭防火墙。

实验环境搭建使用代码如下：



通信机制

SSL VPN的网卡和IP分配：

由于本次实验VPN的网段设置在192.168.53.0/24，除了2个广播地址外，只有254个可用的IP地址。一个VPN客户端IP地址，需要对应一个VPN服务端的IP地址，因此该VPN程序最多支持127个VPN连接。

检查要求并未提出需要为客户端完全随机地分配IP，为了简化程序，不妨令192.168.53.1~127为分配给服务端的IP，192.168.53.128~253为分配给客户端的IP。网卡设备名、虚拟IP的分配遵循如下规律：

虚拟网卡设备名=tun(X)；

服务端虚拟IP=192.168.53.(X+1)；

客户端虚拟IP=192.168.53.(X+127)；

通信机制可用下图1-2概括，下图的SSL隧道上用的是TLS 1.2协议：

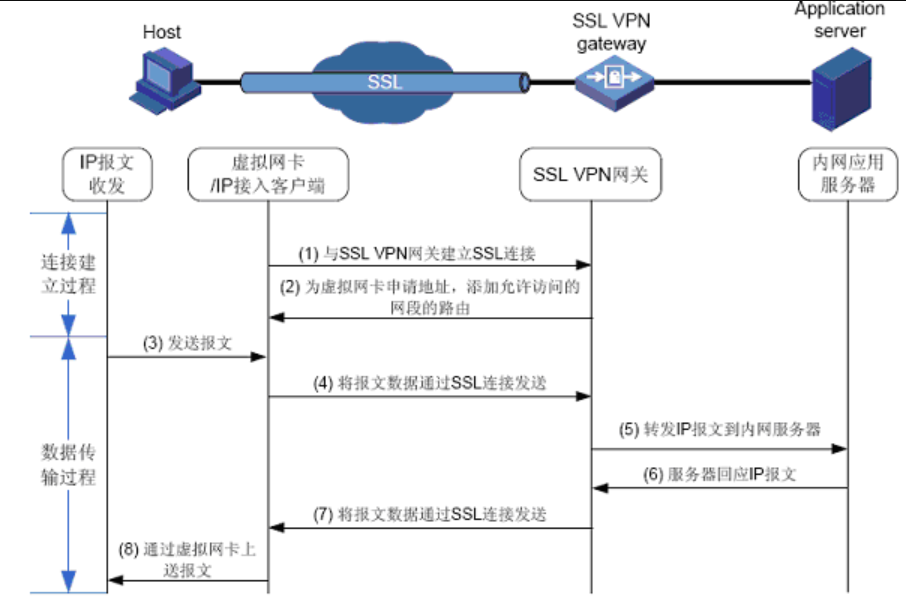


图1-2 通信机制

保证VPN Server始终在SSL VPN网关上运行，通信过程如下：

1）外网主机主动与SSL VPN网关建立SSL安全隧道：协商SSL协议的版本、加密套件、密钥参数，验证通讯双方的身份，建立SSL连接。

2）SSL VPN网关会自动建立新的虚拟网卡，然后向外网主机推送虚拟网卡的虚拟IP地址，虚拟IP地址将从客户端网段192.168.53.128~253中顺序自动分配。外网主机收到来自网关的虚拟IP地址后，客户端也将自动配置虚拟IP相关的路由。

3）外网主机向内网发送IP报文时，会路由到虚拟网卡，虚拟网卡再把报文转发到IP接入客户端。

4）外网主机再将报文使用TLS 1.2协议加密封装，通过SSL隧道发送到SSL VPN网关。

5）SSL VPN网关收到加密的报文后，对其解密得到源IP地址为虚拟IP的报文，SSL VPN网关将该IP报文转发到内网网络中，进而路由到内网主机上。

6）内网主机回报文时，路由到SSL VPN网关。

7）SSL VPN网关收到报文后对其加密，并通过SSL隧道，发送回IP接入客户端。

8）IP接入客户端解出SSL隧道收到的报文，得到目的IP地址为虚拟IP的IP报文，从而写回虚拟网卡。虚拟网卡再把数据报文回给用户应用程序。

当服务端或客户端收到长度为0的报文时视为连接关闭。

加密原理

建立会话时与加密相关的过程可由下图1-3表示。

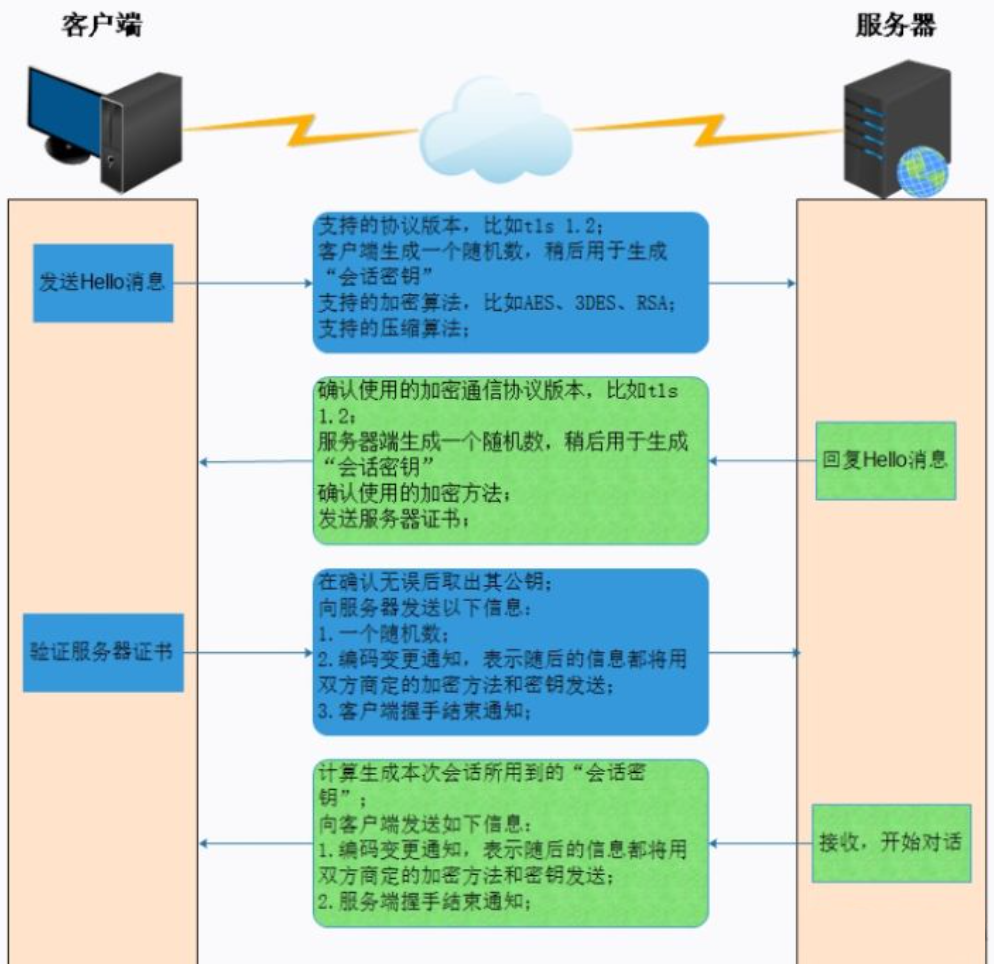


图 1-3 建立会话时与加密相关的过程

会话过程如下：

1）客户端向服务器端索要并验证证书；

2）双方协商生成“会话密钥”；

3）双方采用“会话密钥”进行加密通信。

SSL协议利用RSA作为公钥算法，使用DES、RC4等对称加密算法进行数据加密。SSL协议是采用SSL/TLS综合加密的方式来保障数据安全的。本实验中我们选择TLS 1.2协议。

通信时报文的加密封装如下图1-4所示：

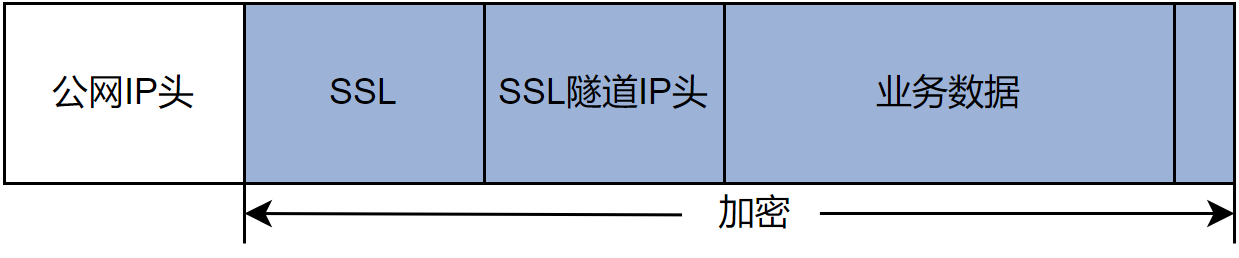


图1-4 报文加密封装

认证机制

SSL使用X.509来认证，本实验中只对服务端认证，客户端认证不采用SSL认证机制，而采用账号密码。

1）客户端对服务器的身份认证：通过公钥证书完成。服务器使用openssl从证书的颁发机构CA获取公钥证书。当客户端连接到VPN服务器时，服务器会将公钥证书发送给客户端，客户端会通过公钥证书验证服务器的身份。

2）服务器对于客户端的认证：通过用户名密码登录完成。在正常的VPN通信之前，服务器会要求客户端发送账号和密码，服务器接受到之后，会将其和/etc/shadow中的内容进行对比。若账号密码正确，就验证成功；否则验证失败，断开连接。

本实验中的CA使用openssl自行创建，并将CA直接放在服务器（域名为chengziServer.com）上：openssl req -new -x509 -keyout ca.key -out ca.crt -config /usr/lib/ssl/openssl.cnf，生成了ca.crt和ca.key，其中ca.key的密钥为00000000000000000，也就是我的学号中的数字部分。具体信息如下图1-5所示：

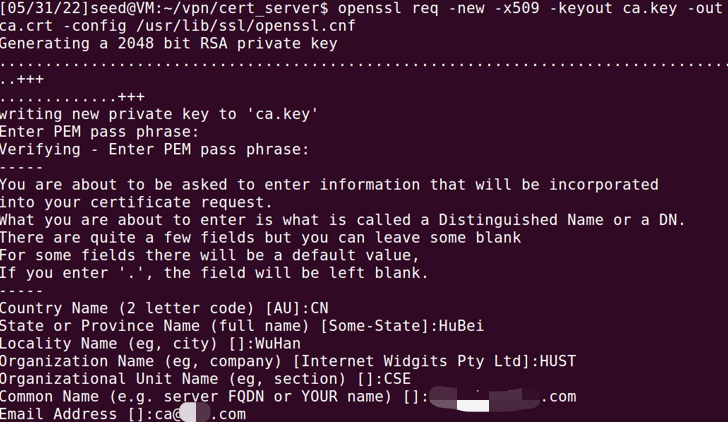


图1-5 创建证书颁发机构

为了方便，再使用同样的密码作为服务器私钥server.key，运行openssl命令创建服务器私钥：openssl genrsa -des3 -out server.key 1024。如下图1-6所示。

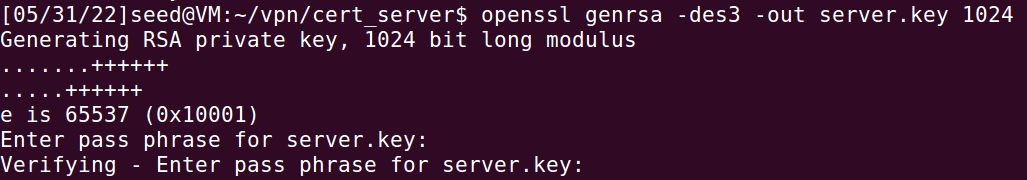


图1-6 创建服务器私钥

再利用server.key创建证书签名请求server.csr：openssl req -new -key server.key -out server.csr -config /usr/lib/ssl/openssl.cnf。

再把server.csr发送给CA，结合ca.crt生成服务端证书server.crt：openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config /usr/lib/ssl/openssl.cnf。

生成的服务器证书如下图1-7所示：

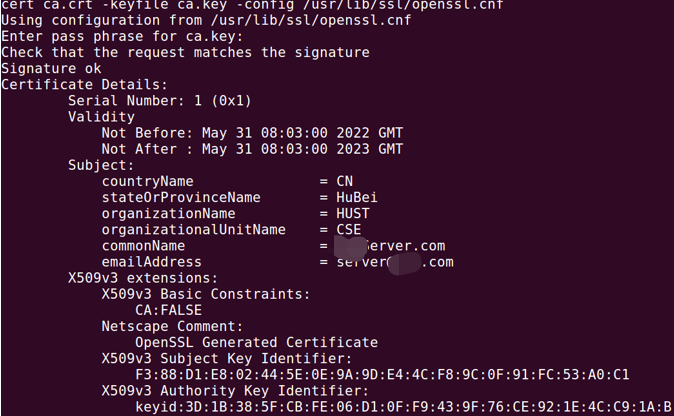


图1-7 生成服务端证书

客户端只需要有CA的证书ca.crt，并配置/etc/hosts，将chengziServer.com域名解析成CA所在的主机的IP地址10.0.2.8，即可向CA验证服务器发来的证书server.crt。

# VPN系统设计

概要设计

系统应用架构

图2-1中VPN网关服务器和其他服务器位于内部网络中不同的主机上。从功能上来说, VPN网关服务器相当于内部网络中的安全代理。对于客户端和VPN网关服务器，由于位于不同的网络之间，因此二者之间要形成一个安全通道，必须使用SSL进行数据加密通信。

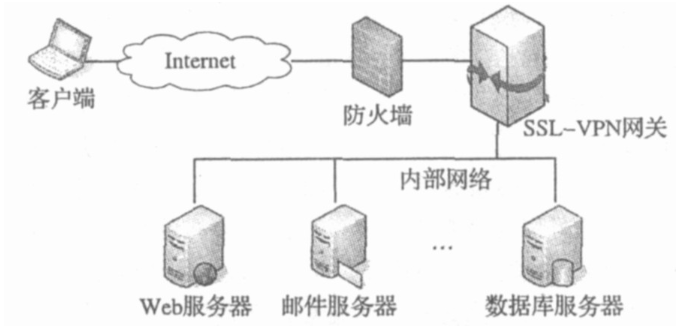


图2-1 VPN网关的应用架构

模块划分

VPN网关可分为五个模块：TLS模块、TCP连接模块、身份认证模块、监听转发模块、客户端管理配置模块。VPN客户端可分为四个模块：TLS模块、TCP连接模块、身份认证模块、监听转发模块。

（1）TLS模块：包括TLS相关参数的初始化、TLS会话协商。对于客户端，还包括接收网关分配的虚拟IP、创建虚拟网卡设备、自动创建路由。

（2）TCP连接模块：对于网关，包括初始化TCP服务端；对于客户端，包括初始化TCP客户端、使虚拟网卡与TCP套接字绑定。

（3）身份认证模块：对于网关，包括客户端验证、用户登录函数；对于客户端，包括验证服务端证书、向服务器认证自己的身份。

（4）监听转发模块：包括向TLS隧道发送数据、从TLS隧道接收数据。

（5）客户端管理配置模块：包括客户端线程处理、创建虚拟网卡设备、分配虚拟IP、自动创建路由。只有VPN网关中有，用于多客户端管理。

工作机制

首先，VPN网关使用“sudo ./vpnserver”运行VPN服务端程序，输入CA的公钥验证自己的身份（我的PEM是00000000000000000），当输入正确时，会显示“Private key match the certificate public key”，然后进入等待客户端连接的状态；输入错误时会退出程序。如下图2-2所示。

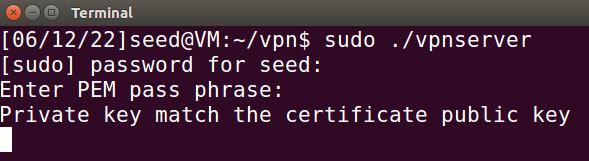


图2-2 VPN网关运行VPN服务端程序

客户端使用“sudo ./vpnclient”运行VPN客户端程序，并输入用户名和密码建立连接。当用户名密码输入正确时，会成功建立连接如下图2-3所示。

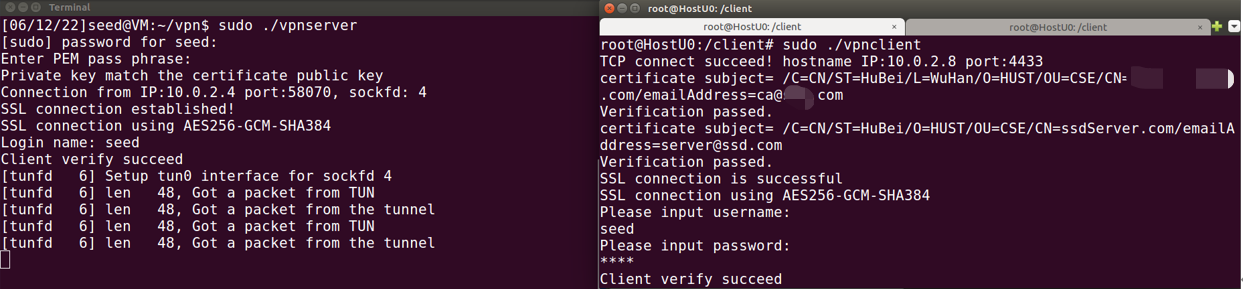


图2-3 客户端成功与VPN网关建立连接

在这一过程中，服务端创建一个客户端线程，与客户端自动协商TLS会话密钥，自动完成虚拟网卡的分配和设置，以及路由的设置。调用到上述所有的模块。

在建立连接后，客户端程序便可以正常访问VPN内网主机192.168.60.101，以telnet连接为例，如下图2-4所示。（内网主机需要使用“service openbsd-inetd start”开启telnet服务）

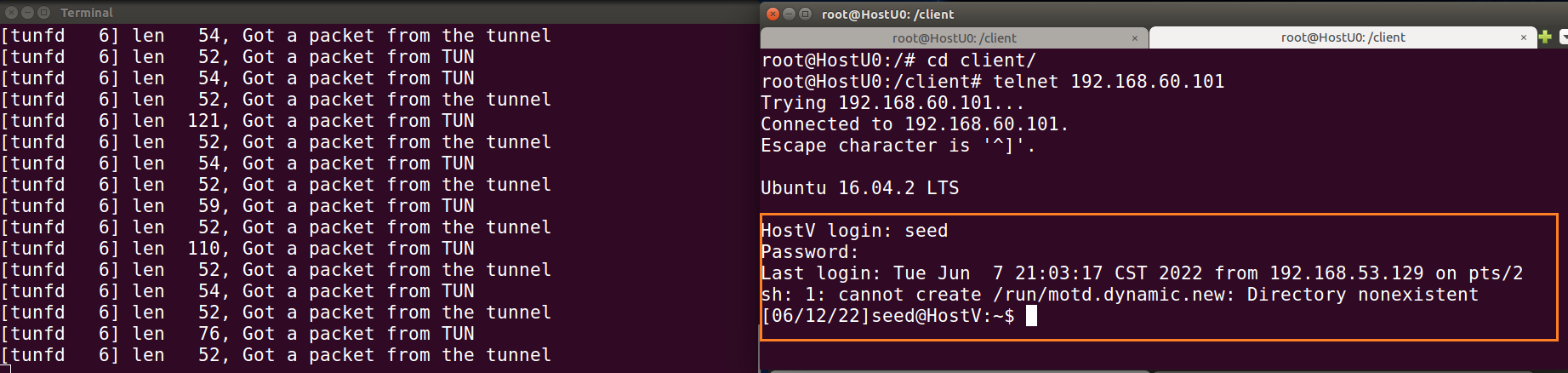


图2-4 客户端telnet连接内网主机

多个客户端向服务器建立连接，其结果与上述过程相仿，只是服务端会另外开几个客户端线程。

当一个客户端停止连接建立时，将会给服务端发送长度为0的报文，服务端将会中止该客户端的线程，但服务段程序并不会随之中止。如下图2-5所示。

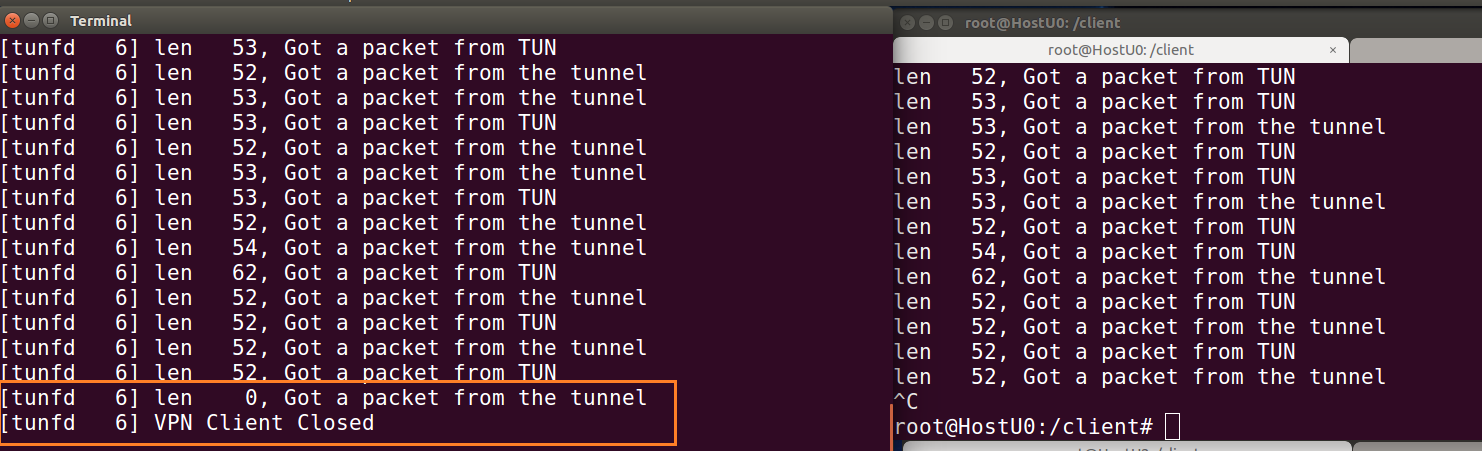


图2-5 客户端程序主动停止时服务端线程主动关闭

同理，当服务端的VPN程序停止运行时，客户端进程会自动关闭。如下图2-6所示。

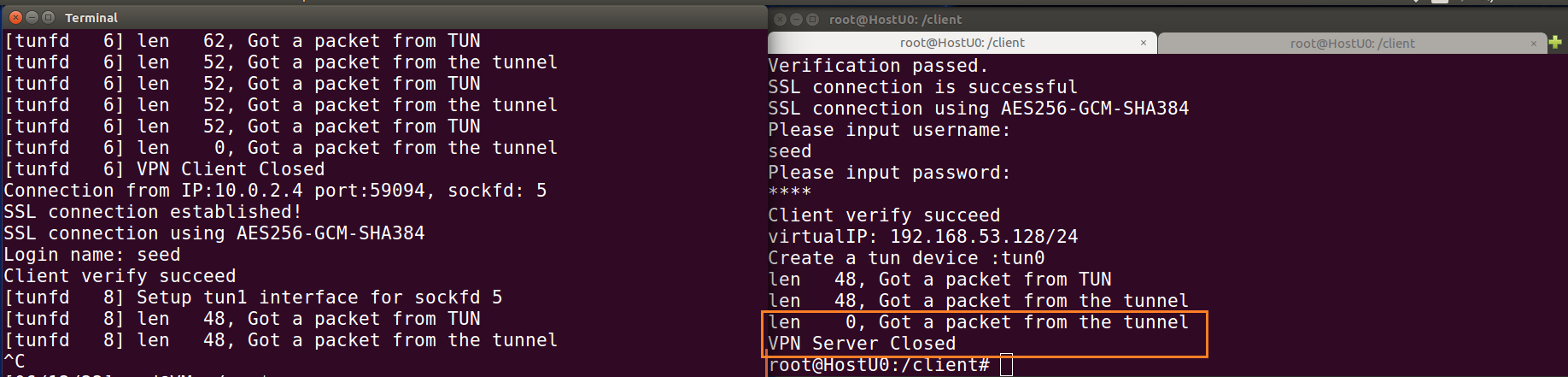


图2-6 服务端程序停止运行时客户端进程主动关闭

详细设计

需要说明的主要功能有建立TLS隧道（包括TLS初始化、会话协商）、创建TCP连接、创建虚拟专用网络（包括创建虚拟网卡设备、分配虚拟IP、自动创建路由）、身份认证（包括证书认证和用户名密码认证）、监听转发、客户端线程管理。

建立TLS隧道

（1）TLS初始化

TLS初始化函数的流程图如下图2-7所示。服务端会返回SSL\_CTX结构的指针，客户端会返回SSL结构的指针。

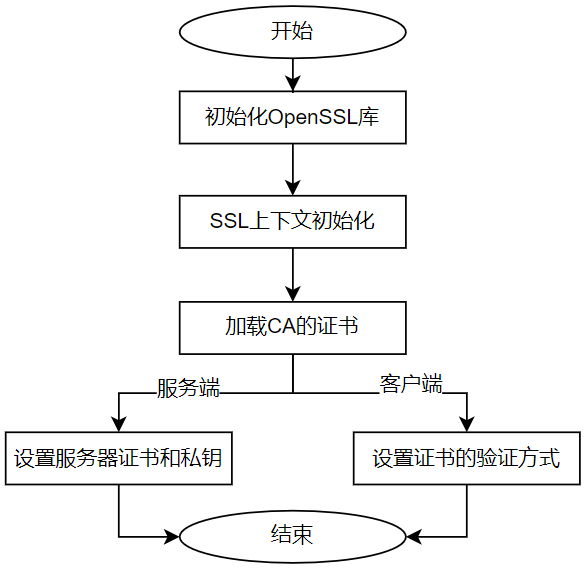


图2-7 TLS初始化函数

（2）TLS会话协商

服务端协商过程是新建套接字，然后绑定TCP套接字，再使用SSL\_accept接受客户端的连接握手。如下图2-8所示：

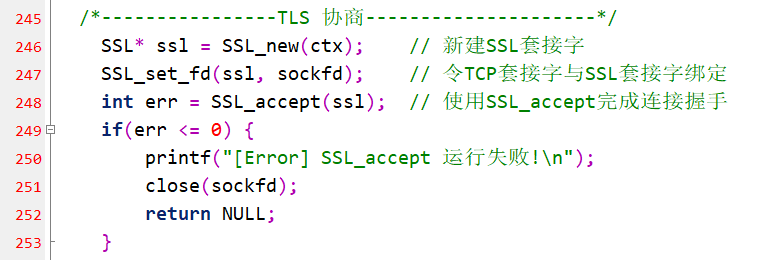


图2-8 服务端的TLS协商

客户端协商过程是将套接字与TCP套接字绑定，再使用SSL\_connect向服务端发起SSL连接。如下图2-9所示：

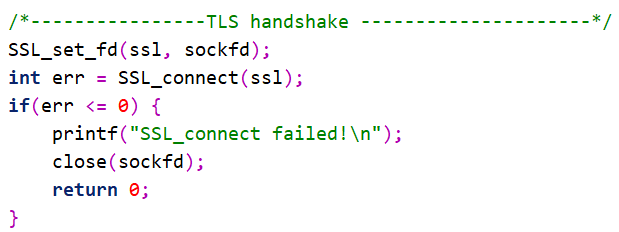


图2-9 客户端的TLS协商

初始化TCP连接

服务端，如下图2-10所示：

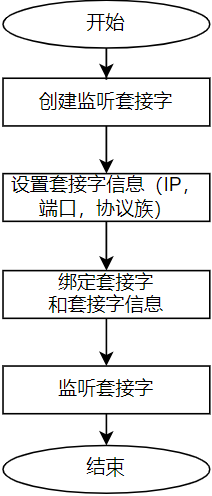


图2-10 服务端设置TCP连接监听

客户端，如下图2-11所示：

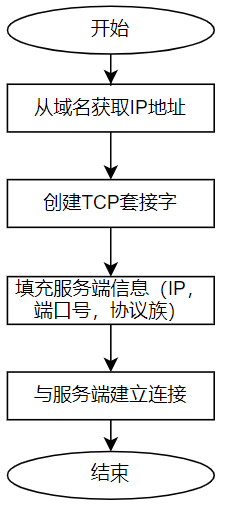


图2-11 客户端发起TCP连接

创建虚拟专用网络

（1）创建虚拟网卡设备

关键代码如下图2-12所示（删去了错误处理）：



图2-12 创建虚拟网卡设备的关键代码

注意在创建tun虚拟设备的时候，需要先加互斥锁，以免客户端打开同一个设备。打开设备后使用ioctl设置设备的结构，然后获得当前虚拟设备的编号。不过客户端程序不需要加锁。

（2）分配虚拟IP

由服务器程序来分配，分配的原则如下：

若服务端的虚拟网卡设备名=tun(X)，则①服务端虚拟IP=192.168.53.(X+1)；②客户端虚拟IP=192.168.53.(X+127)。

1. 分配服务器的虚拟IP：服务执行sudo ifconfig tun**(X)** 192.168.53.**(X+1)**/24 up；
2. 分配客户端的虚拟IP：将虚拟IP发回给客户端，然后客户端执行sudo ifconfig tun**(X)** 192.168.53.**(X+127)**/24 up；

（3）自动创建路由

服务端执行：sudo route add -host 192.168.53.**(X+127)** tun**(X)**；

客户端执行：sudo route add -net 192.168.60.0/24 dev tun%d，这里的tun是客户端的编号，不一定是X。

身份认证

（1）证书认证

客户端得到服务端发来的证书时，将利用CA机构验证服务器的证书，这个功能在客户端由verifyCallback()函数完成，如下图2-13所示。



图2-13 认证服务端证书

（2）用户名密码认证

在进行身份认证时，客户端需要输入账号密码，这个功能在服务端由verifyClient()函数完成，如下图2-14所示。

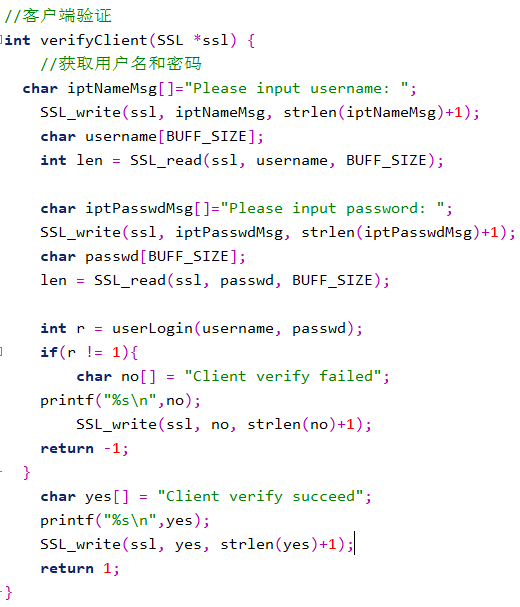


图2-14 认证客户端用户密码

监听转发

主要用到SSL\_write、SSL\_read这两个函数。

（1）向TLS隧道发送数据（如下图2-15所示）

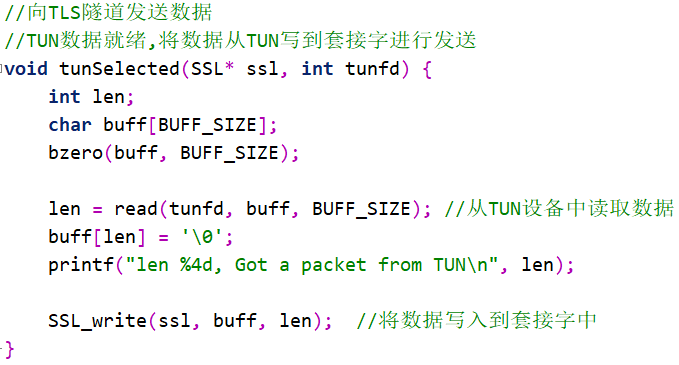


图2-15 向TLS隧道发送数据

（2）从TLS隧道接收数据（如下图2-16所示）



图2-16 从TLS隧道接收数据

客户端线程管理

服务端一旦与客户端TLS协商成功，就会创建一个客户端线程。客户端线程中的具体处理在服务端程序的threadClient函数里。如下图2-17所示。

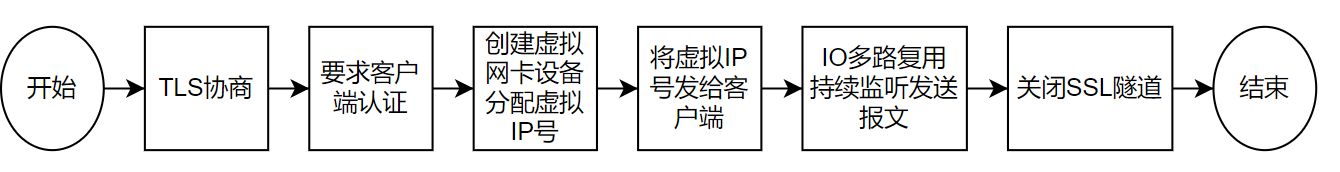


图2-17 客户端线程函数的流程图

# VPN实现细节

1. **你的VPN客户端虚IP分别是如何获得的？是手动还是自动配置到tun虚网卡上的？自动配置如何实现？**

答：我的VPN客户端虚IP是通过VPN服务端分配的，是自动配置到tun虚网卡上的。

自动配置的过程：

1. 运行VPN客户端程序，与VPN服务端建立由TLS协议保护的TCP连接；
2. 建立正确的TCP连接后，服务器将创建新的线程；
3. 服务器线程打开未使用的Tun虚拟网卡，将(网卡编号+127)通过TCP连接发送给VPN客户端。
4. VPN客户端将“192.168.53.(网卡编号+127)”作为虚IP，并自动打开一个Tun虚网卡，自动运行“sudo ifconfig tun<虚网卡号> <虚IP>/24 up”将虚IP绑定到Tun虚网卡上。
5. **你的VPN客户端到内部子网的路由，是手动还是自动配置的？自动配置如何实现？**

答：我的VPN客户端到内部子网的路由是自动配置的。如（1）所答，VPN客户端接收到VPN服务端发回的虚IP号之后，用“sudo route add -net 192.168.60.0/24 dev tun<虚网卡号>”自动设置VPN客户端到内部子网的路由。

1. **你的VPN客户端用户名口令认证是在隧道协商的什么阶段实现的？认证失败时双方是如何处理的？**
   1. 是在TCP连接建立完成之后。
   2. 认证失败时，客户端报错并退出，服务端报错但不中止程序。如下图3-1所示。

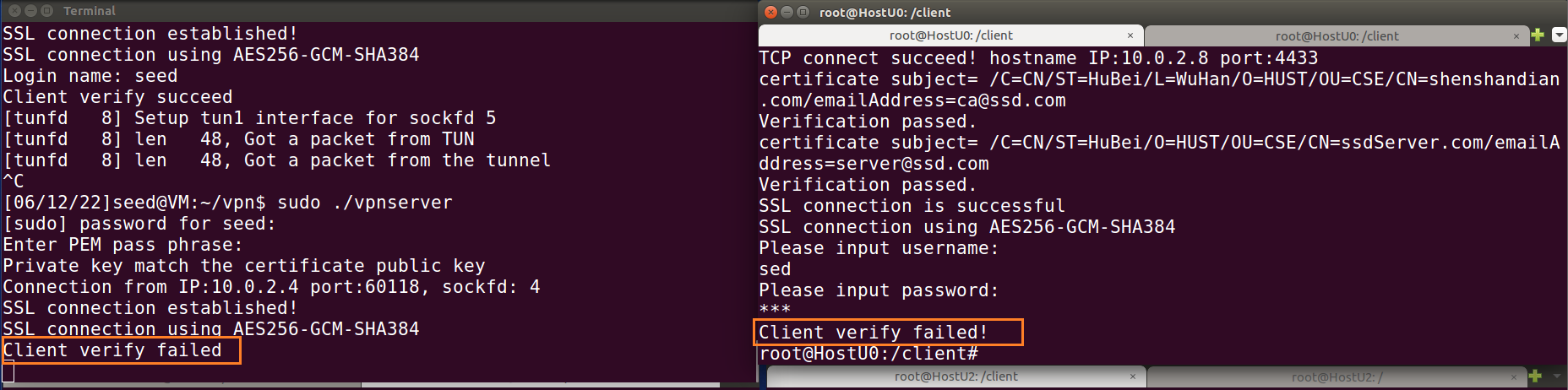


图3-1 认证失败时双方的处理方式

1. **你的VPN服务器支持多客户端采用的多进程还是多线程架构？进程间通信采用的何种技术？VPN服务器收到内网主机的应答报文时，如何判断应发送给哪个VPN客户端的隧道？**
   1. 多线程。
   2. 使用的是多线程，同步和互斥使用的是mutex锁。不涉及进程间通信。
   3. 根据内网的报文发往哪个TUN接口，来判断数据应该发往哪个客户端的隧道。
2. **你的VPN客户端退出时，VPN服务器如何发现？如何处理？**
   1. VPN客户端退出时，VPN服务器会收到长度为0的报文，VPN服务器就会发现；
   2. 服务器的该线程会停止继续监听，然后直接返回，结束这个线程；
3. **你的VPN服务器退出时，VPN客户端如何发现？如何处理？**
   1. VPN服务器退出时，VPN客户端会收到长度为0的报文，VPN客户端就会发现；
   2. VPN客户端进程会立即结束；
4. **请介绍你的设计实现中个人最满意的特色部分。**

输出信息设计得不错。

由于是多客户端程序，服务端对客户端的输出信息理应有一定的区分。我以tunfd作为区分，并且在建立连接时提示tunfd对应的是哪个客户端。如下图3-2所示。



图3-2 服务端程序的输出信息

# 测试结果与分析

我运行一个服务端程序、三个客户端程序测试，服务端运行在seed@VM中，每个客户端都是docker环境，并分别启用两个终端。测试初始时界面如下：

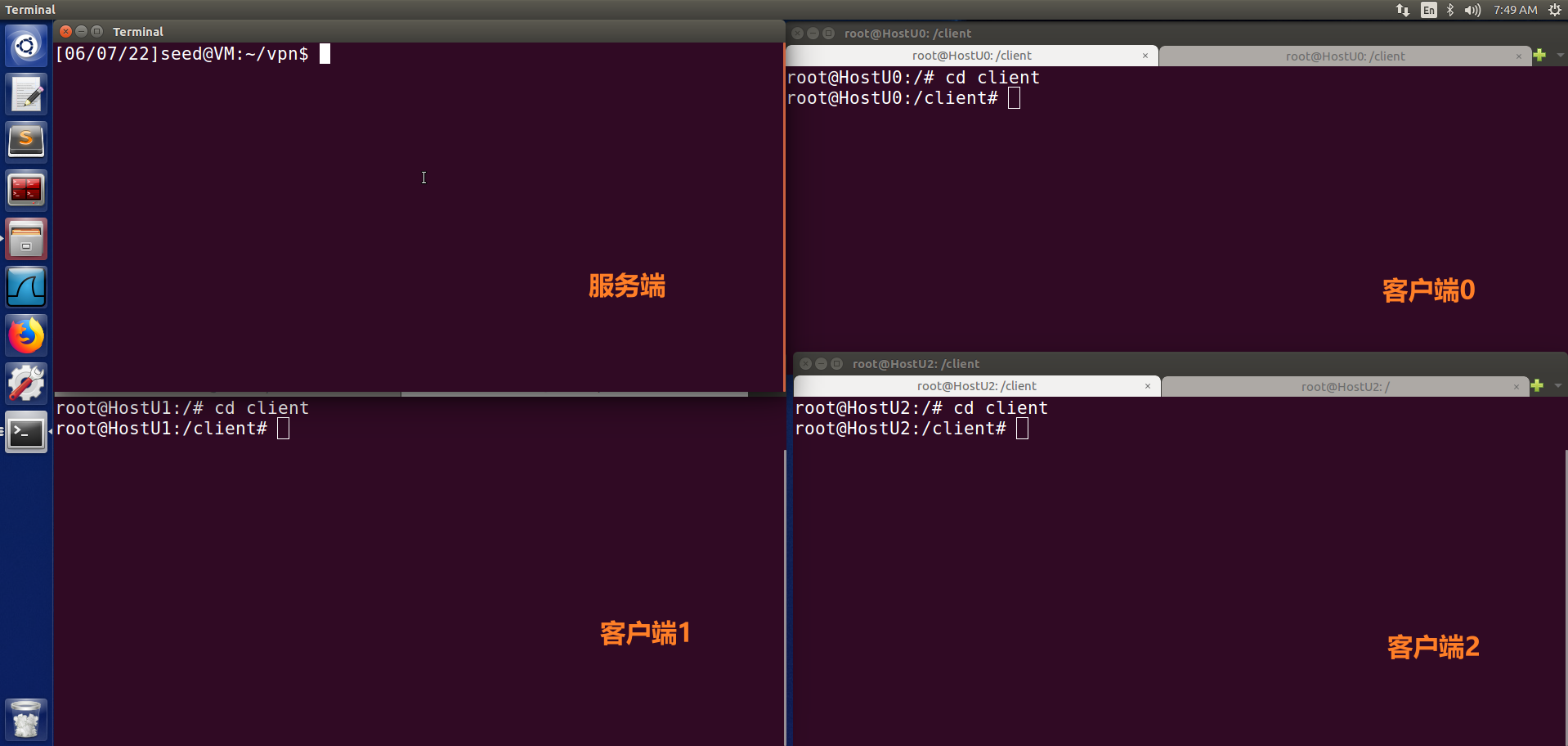


图4-1 测试初始时的界面

认证VPN服务器

（1）用opensssl检查VPN服务器证书信息。

在seed@VM中运行指令：openssl x509 -noout -text -in cert\_server/server.crt。如下图4-2所示，证书主题包含个人信息，符合要求。

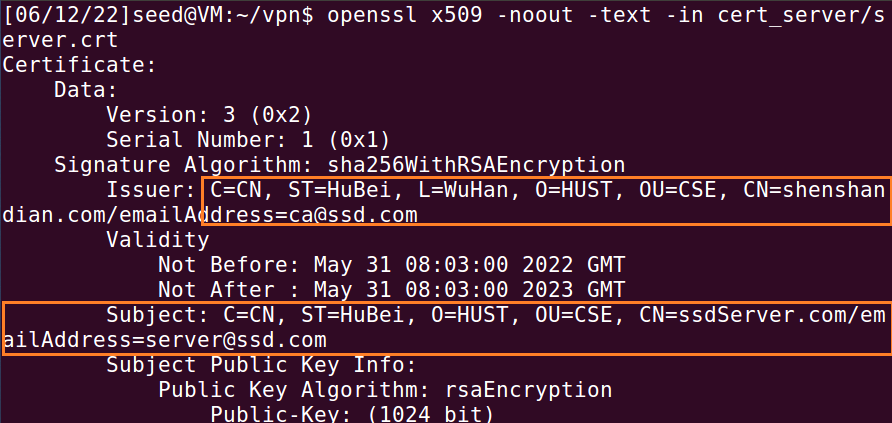


图4-2 用opensssl检查VPN服务器证书信息

（2）修改VPN客户端主机时间到VPN服务器证书有效期之后再登录VPN服务器。

我的证书有效期是2022-06-30。在HostU0中运行指令：date -s 2023-06-30，修改docker中的日期，此时再尝试登录VPN服务器。如下图4-3所示，VPN客户端提示证书过期，符合要求。

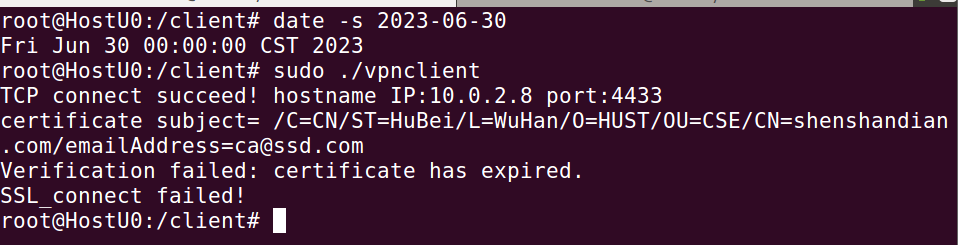


图4-3 证书有效期之后再登录VPN服务器

认证VPN客户端

（1）VPN客户端以错误的用户名或口令登录VPN服务器

如下图4-4所示，提示错误无法登录，符合要求。

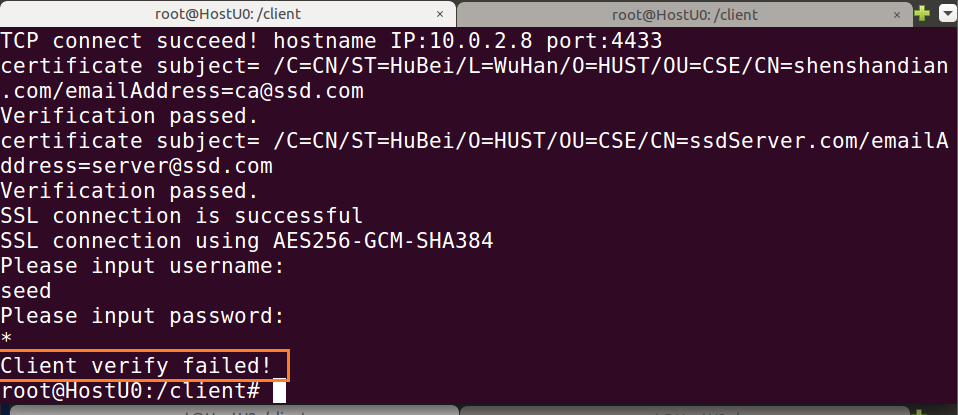


图4-4 以错误的口令登录VPN服务器

（2）VPN客户端以正确的用户名口令登录VPN服务器

如下图4-5所示，能正确登录，符合要求。

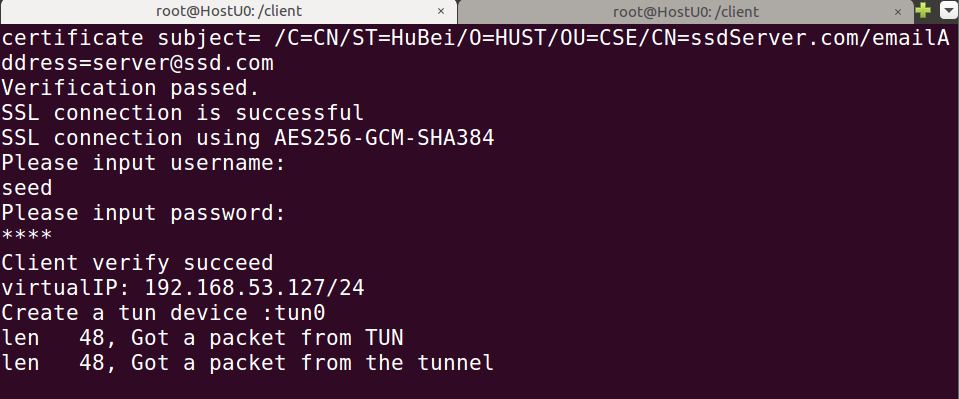


图4-5 以正确的用户名口令登录VPN服务器

加密隧道通信

（1）VPN客户端ping内网主机，wireshark在VPN服务器外口截包检查

如下图4-6所示，选择网卡docker1，能通信、经隧道封装、隧道为TLS，符合要求。

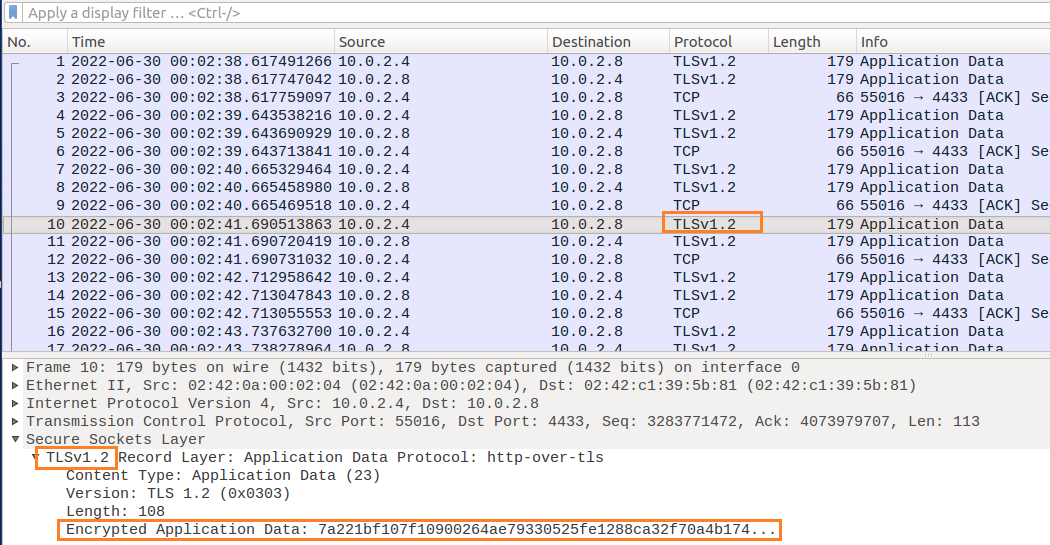


图4-6 Wireshark在在VPN服务器外口截包检查

支持多客户端

（1）开启2个以上VPN客户端容器，同时登录VPN服务器，分别测试telnet通信

如下图4-7所示，各自正常登录通信，符合要求。

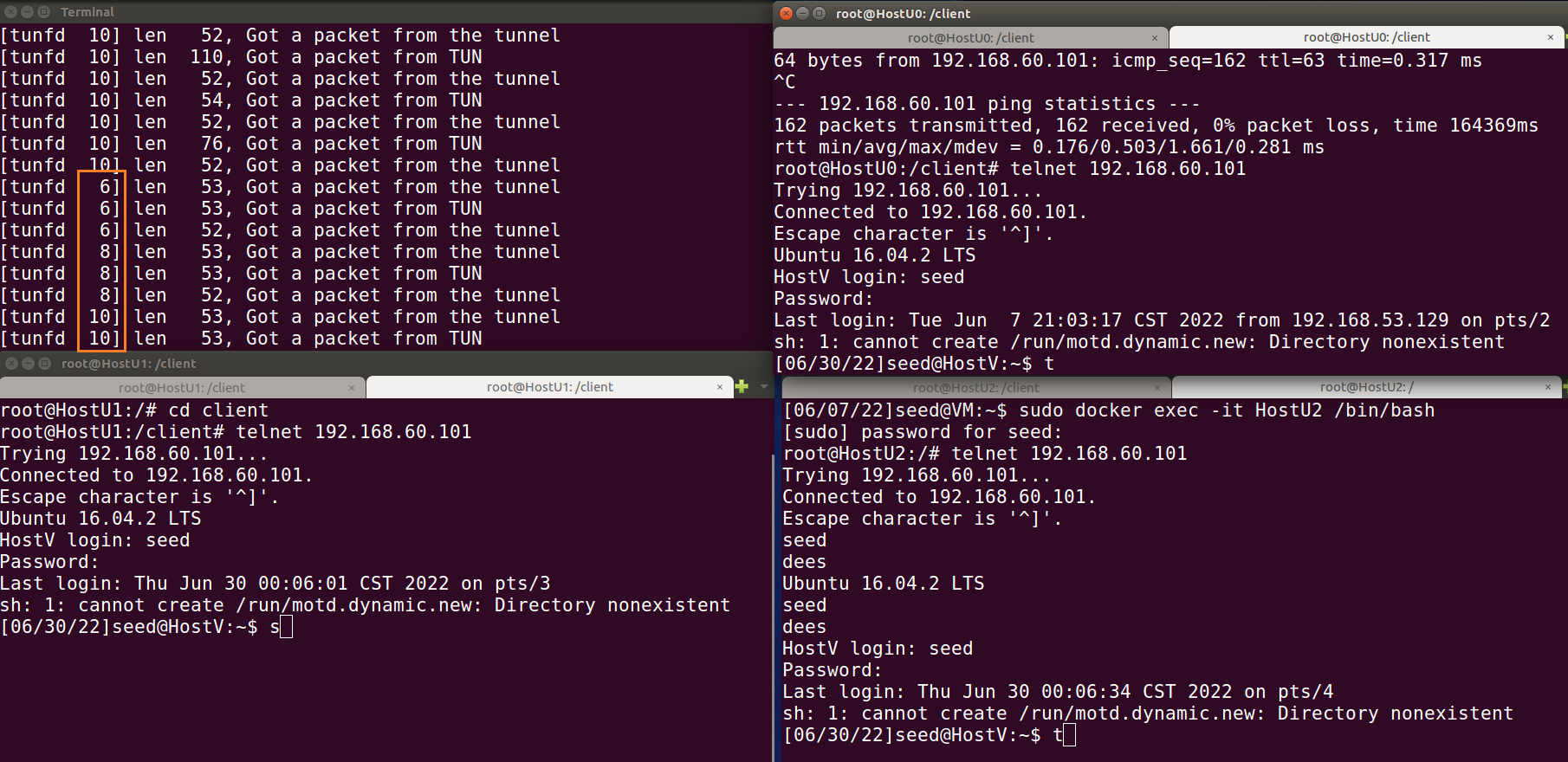


图4-7 三个VPN客户端同时登录VPN并telnet连接内网主机

（2）断开其中一个VPN客户端，测试另外一个的隧道通信

如下图4-8所示，断开HostU0的，HostU1的隧道保持，通信不受影响，符合要求。

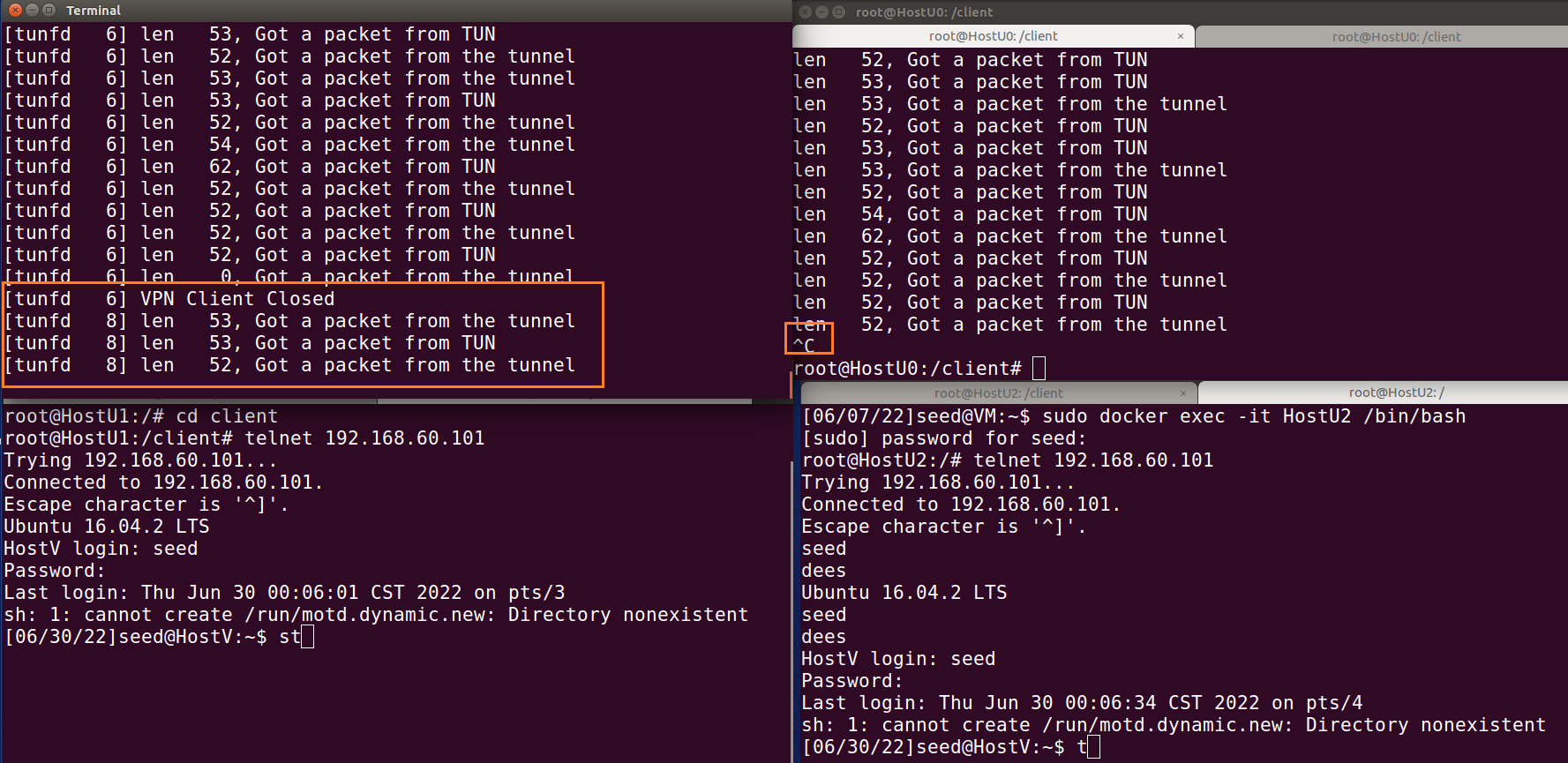


图4-8 断开HostU0测试HostU1的隧道通信

易用性和稳定性

（1）VPN客户端虚拟IP获取、VPN客户端虚拟IP配置

如下图4-9所示，运行客户端程序，可见由VPN服务器自动分配并获取，符合要求。

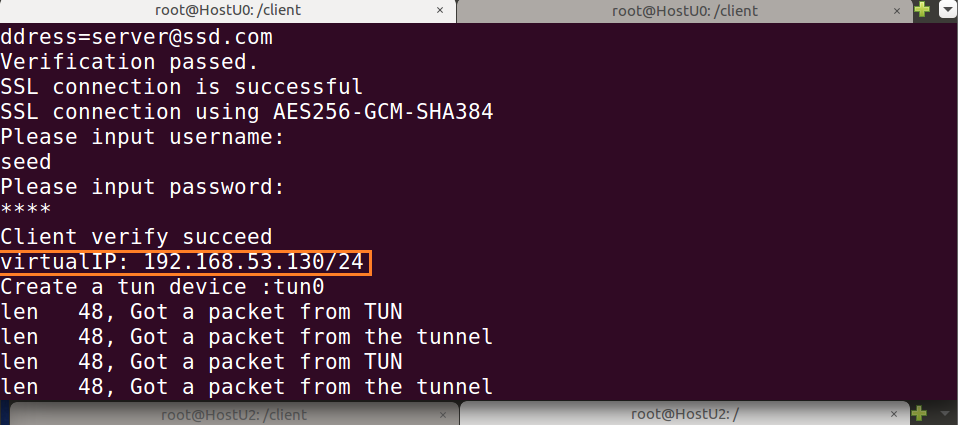


图4-9 VPN客户端虚拟IP获取及配置

（2）VPN客户端内网路由配置

如下图4-10所示，客户端上运行route -n查看当前路由，可见由程序自动添加，符合要求。

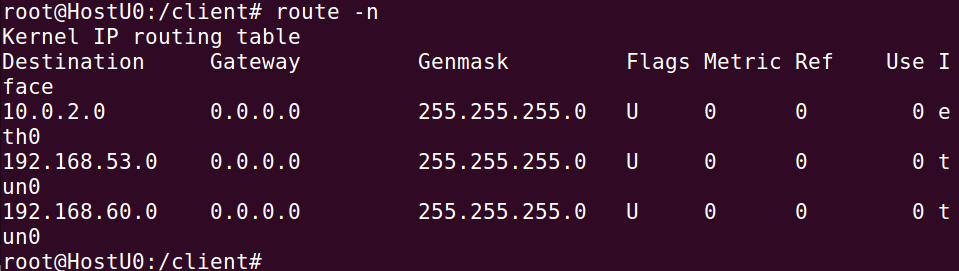


图4-10 VPN客户端内网路由配置

（3）正常使用时的稳定性

如下图4-11所示，同时开启所有客户端程序，并同时ping内网主机，可见运行稳定，符合要求。

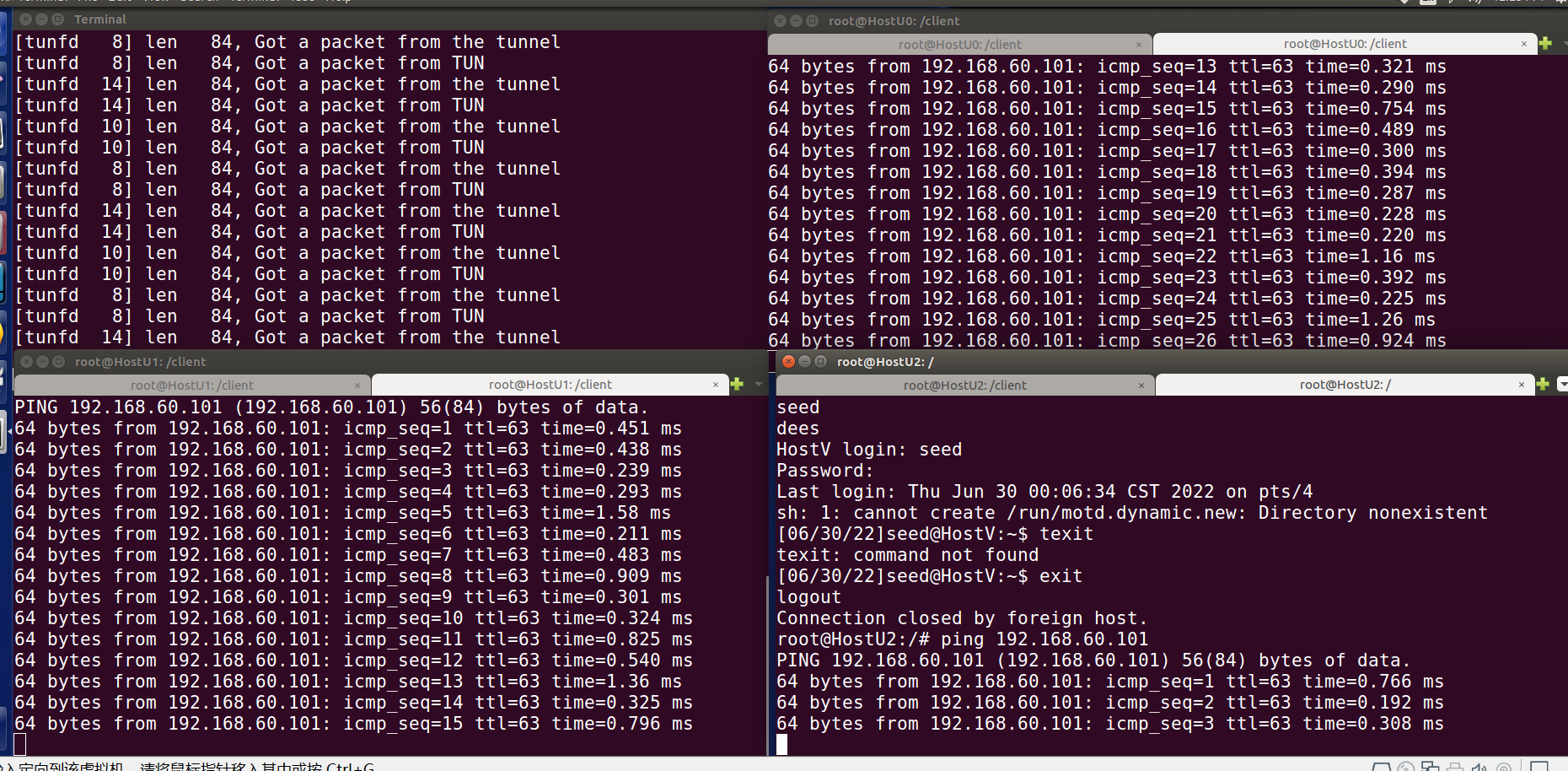


图4-11 开启所有客户端程序同时ping内网主机

# 体会与建议

心得体会

本次实验，在老师已经分别提供了VPN和TLS的代码的基础上，整体而言比较简单。

通过本次实验，我进一步巩固了理论课上的VPN通信相关的知识，对通信的原理和流程的理解更为透彻。并且充分理解了使用openssl创建证书，以及理解证书之间的关系。

最让我感到意外的是，VPN程序的实现原理竟然如此简单，有了虚拟网卡技术对相关操作的封装，完全不需要对数据包逐个逐个截获、拆解分析、转发，也不需要在VPN程序中去建立并维护路由表，只需要适当添加路由，并启用虚拟网卡。

在本次实验之前，VPN一直是流传在民间的神秘程序；而在本次实验之后，我却有了一定的把握在自己的服务器上写一个VPN程序，并且编织一个属于我自己的内网。揭开了内外网和VPN的神秘面纱，这让我颇有成就感。

感谢老师们精心编写的实验课程！

意见建议

这个实验报告的截止是在期末期间，实验报告的要求有些过于繁琐。我认为实验报告应该是体现我们经过探究得到的结果，需要弱化实验课件、材料中已经明确给出的内容。

例如，对于详细设计部分，需要介绍数据结构、流程图和关键代码。实话说，只要读一遍给出来的vpn和tls程序，便可以理解其中的函数的大致作用，我们在做实验的过程中，只需将现成的功能代码复用即可。在某种意义上讲，我们相当于面向对象在编程，对数据结构基本不关心。而我们所做的有效探索，即如何组织功能模块的部分，已经全部写进概要设计中了。

我清楚老师给出vpn和tls程序是为了减轻我们的负担，而不是让我们忽略细节。可是一方面，这些要求真的蛮累人的；另一方面，我们已经达到了实验目的，对程序本身的探索欲会大大降低。

我认为今后可以删去**2.2详细设计**，只保留**3 VPN实现细节**即可。

再次感谢老师提供的程序和讲解！