

## 1. 认证 VPN 服务器

### 用 openssl 检查 VPN 服务器证书信息

在 VM 输入以下指令查看 VPN 服务器证书信息：

openssl x509 -in ca-dsy-crt.pem -text -noout。

展示信息如下，含有本人信息 dsy、邮箱地址、过期时间等内容：

```
root@VM:/home/seed/Desktop/miniVPN/cert_server# openssl
x509 -in ca-dsy-crt.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            f5:f3:56:a1:44:76:fb:46
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=CN, ST=HB, L=WH, O=HUST, OU=CSE, CN=dsy/emailAddress=Shawn_Dai@outlook.com
        Validity
            Not Before: Dec  9 17:11:19 2024 GMT
            Not After : Jan  8 17:11:19 2025 GMT
        Subject: C=CN, ST=HB, L=WH, O=HUST, OU=CSE, CN=dsy/emailAddress=Shawn_Dai@outlook.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
            Modulus:
                00:9b:a8:c5:eb:db:ff:14:19:66:b1:82
```

### 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

新建一台测试用客户端容器 timeout：

```
sudo docker run -it --name=timeout --hostname=timeout --net=extranet --ip=10.0.2.6 -
-privileged "seedubuntu" /bin/bash
```

在容器 timeout 中，输入以下指令调整系统时间到过期时间之后（更改/etc/localtime 文件）：

```
sudo date -s "2035-01-01 12:00:00"
```

```
root@timeout:/miniVPN# ./miniVPNclient dsy 4433
Enter PEM pass phrase:
subject= /C=CN/ST=HB/L=WH/O=HUST/OU=CSE/CN=dsy/emailAdd
ress=Shawn_Dai@outlook.com
Verification failed: certificate has expired.
```

测试成功，将系统时间调回正常同步时间：

## 2. 认证 VPN 客户端

### VPN 客户端以错误的用户名或口令登录 VPN 服务器

尝试输入错误的（未在服务端/etc/shadow 中设置的）用户名/密码，会有错误的提示。  
针对用户名不存在的情况，会有如下提示：

```
Malloc ip successfully.Your ip is 192.168.53.5
Server says:
Please input username:non-existence
Please input password:
login response: length is 3, content is -1
User non-existence not found or it is not a valid user.
```

针对密码错误不匹配的情况，会有如下提示：

```
Malloc ip successfully.Your ip is 192.168.53.5
Server says:
Please input username:dsyyy
Please input password:
login response: length is 3, content is -2
Invalid password.
```

### VPN 客户端以正确的用户名口令登录 VPN 服务器

输入预先在服务端的/etc/shadow 中设置的用户名+密码（dsyy+dsy），可以正确登录：

```
Please input username:dsyy
Please input password:
login response: length is 2, content is 1
Login successfully.
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
```

### 3. 加密隧道通信

#### 能通信

在 VPNclient 和 VPNserver 都启动的情况下，在 HostU 去 ping 192.168.60.101 (HostV)，可以 ping 通：

```
root@HostU:/miniVPN# ping 192.168.60.101
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data.
64 bytes from 192.168.60.101: icmp_seq=1 ttl=63 time=1.01 ms
64 bytes from 192.168.60.101: icmp_seq=2 ttl=63 time=0.808 ms
64 bytes from 192.168.60.101: icmp_seq=3 ttl=63 time=1.02 ms
64 bytes from 192.168.60.101: icmp_seq=4 ttl=63 time=0.669 ms
64 bytes from 192.168.60.101: icmp_seq=5 ttl=63 time=0.486 ms
```

并且可以看见在 VM 启动的服务端有对应的信息记录：

```
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
```

此时结束 HostU 的客户端进程，HostU 和 HostV 不再能通信：

```
SSL read 0 bytes
tun written 0 bytes
ip 192.168.53.3解除占用
管道文件 /home/seed/Desktop/miniVPN/pipe/192.168.53.3
对应的文件描述符弃用
Close sock and end subprocess pid:24069

root@HostU:/miniVPN# ping 192.168.60.101
connect: Network is unreachable
root@HostU:/miniVPN#
```

#### 经隧道封装

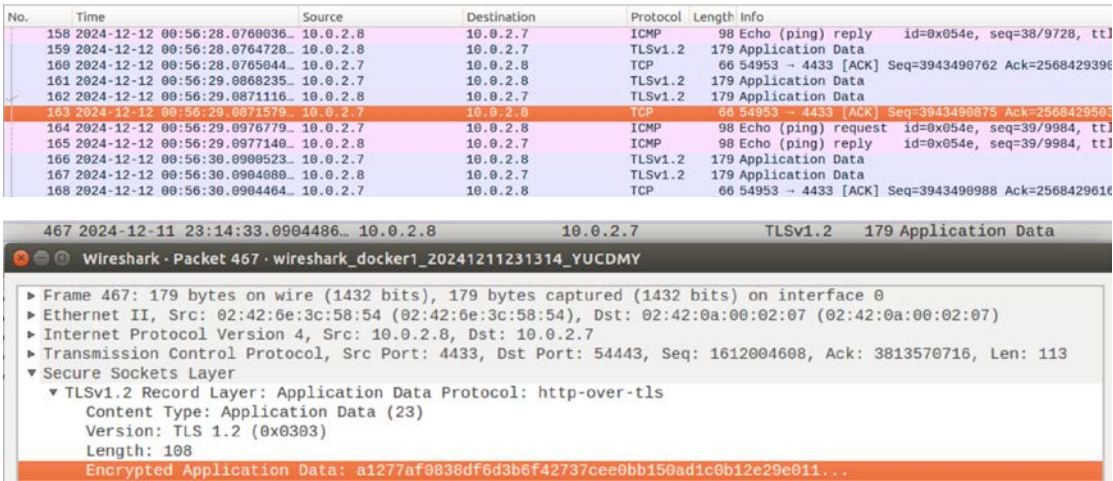
Wireshark 根据端口号识别 TLS/SSL 流量。它知道 443 是 HTTPS 的默认端口号，但我们的 VPN 服务器侦听不同的非标准端口号。我们需要让 Wireshark 知道这一点；否则，Wireshark 不会将我们的流量标记为 SSL/TLS 流量。

转到 Wireshark 的 Edit 菜单，然后单击 Preferences，Protocols，HTTP，然后找到“SSL /



TLS Ports"条目。添加 SSL 服务器端口。例如，我们可以将条目的内容更改为 443, 4433, 其中 4433 是我们的 SSL 服务器使用的端口。

使用 HostU 去 ping HostV, 同时在 wireshark 中查看流量，可以看到在客户端和服务端之间的通信报文，且数据使用了 TLS 进行了加密：



## 隧道为 TLS

上面显示的方法只能让 Wireshark 将流量识别为 TLS/SSL 流量，Wireshark 无法解密加的流量。出于调试目的，我们希望看到无法解密加的流量。Wireshark 提供了这样的功能。我们需要做就是 Wireshark 提供服务器的私钥，Wireshark 将自动从 TLS/SSL 握手协议派生会话密钥，并使用这些解密流量。

进行以下操作，进行必要的修改，如服务器私钥路径和密码即可。

Click Edit -> Preferences -> Protocols -> SSL

Find the "RSA key list", and click the Edit button

Provide the required information about the server, see this example:

IP Address: 10.0.2.65

Port: 4433

Protocol: ssl

Key File: /home/seed/vpn/server-key.pem (privat key file)

Password: deesdees

出现的问题：wireshark 需要完整的 p12 文件，而不知 pem 文件。所以需要先用 openssl 把之前生成的私钥转换成 PKCS#12 文件才能使用：

```
openssl pkcs12 -export -out server-cert.p12 -inkey server-dsy-key.pem -in server-dsy-crt.pem
```

就可以自动解密报文的内容。

## 4. 支持多客户端

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

开启两个容器 HostU 和 U2，清空默认路由，然后分别登录 VPN 服务器：

```
root@HostU:/miniVPN# ./miniVPNclient dsy 4433
Enter PEM pass phrase:

subject= /C=CN/ST=HB/L=WH/O=HUST/OU=CSE/CN=dsy/email=dsy@hust.edu.cn
Verification passed.
subject= /C=CN/ST=HB/L=WH/O=HUST/OU=CSE/CN=dsy/email=dsy@hust.edu.cn
Verification passed.
SSL connection is successful
SSL connection using AES256-GCM-SHA384
net.ipv4.ip forward = 1
Setup TUN interface success!
Malloc ip successfully.Your ip is 192.168.53.3
Server says:
Please input username:dsyyy
Please input password:
login response: length is 2, content is 1
Login successfully.
Got a packet from TUN

root@U2:/miniVPN# ./miniVPNclient dsy 4433
Enter PEM pass phrase:

subject= /C=CN/ST=HB/L=WH/O=HUST/OU=CSE/CN=dsy/email=dsy@hust.edu.cn
Verification passed.
subject= /C=CN/ST=HB/L=WH/O=HUST/OU=CSE/CN=dsy/email=dsy@hust.edu.cn
Verification passed.
SSL connection is successful
SSL connection using AES256-GCM-SHA384
net.ipv4.ip forward = 1
Setup TUN interface success!
Malloc ip successfully.Your ip is 192.168.53.4
Server says:
Please input username:wzyyy
Please input password:
login response: length is 2, content is 1
Login successfully.
Got a packet from TUN
```

同时在 HostU 和 U2 上 ping 192.168.60.101，都能 ping 通：

```
root@HostU:/# ping 192.168.60.101
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data.
64 bytes from 192.168.60.101: icmp_seq=1 ttl=63 time=2.49 ms
64 bytes from 192.168.60.101: icmp_seq=2 ttl=63 time=1.16 ms
64 bytes from 192.168.60.101: icmp_seq=3 ttl=63 time=0.913 ms

root@U2:/# ping 192.168.60.101
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data.
64 bytes from 192.168.60.101: icmp_seq=1 ttl=63 time=1.62 ms
64 bytes from 192.168.60.101: icmp_seq=2 ttl=63 time=1.66 ms
64 bytes from 192.168.60.101: icmp_seq=3 ttl=63 time=2.67 ms
```

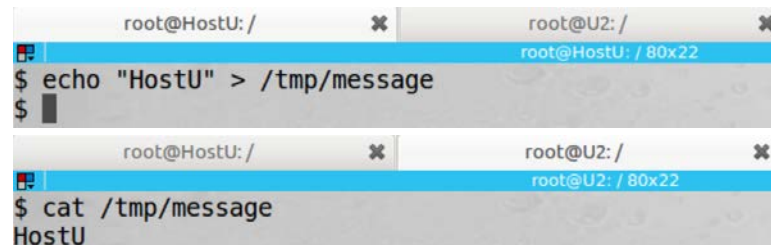
并且可以在服务器端看到分别对应的信息记录：

```
Receive from TUN: packet length:84,packet dst ip: 192.168.53.4
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.4 with pipefd 10
pipe listen running
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.4
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.4
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.4 with pipefd 10
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.4
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.4
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.4 with pipefd 10
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.4
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
```

telnet 的测试，先在 HostV 启动 telnet 服务：

```
service openbsd-inetd restart
```

然后分别在 HostU 和 U2 用 telnet 连上 192.168.60.101，使用 dsy 用户登录后，可以看到每输入一个字符，在 VPN 服务器端都有对应的数据，使用一个写文件，一个读文件做简单测试，可以看见多客户端的通信是正常的：



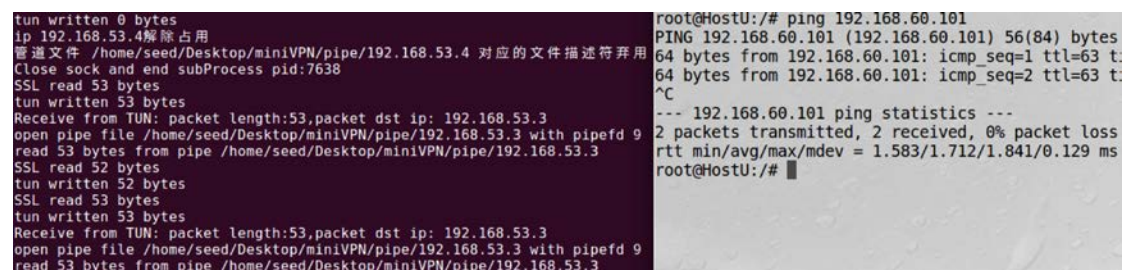
```
root@HostU: / root@U2: /
$ echo "HostU" > /tmp/message
$
root@HostU: / root@U2: /
$ cat /tmp/message
HostU
```

## 断开其中一个 VPN 客户端，测试另外一个的隧道通信

断开 U2 客户端：

```
SSL read 0 bytes
tun written 0 bytes
ip 192.168.53.4解除占用
管道文件 /home/seed/Desktop/miniVPN/pipe/192.168.53.4 对应的文件描述符弃用
Close sock and end subprocess pid:7638
```

再次尝试用 HostU 通信 HostV，可以看见通信仍然正常：



```
tun written 0 bytes
ip 192.168.53.4解除占用
管道文件 /home/seed/Desktop/miniVPN/pipe/192.168.53.4 对应的文件描述符弃用
Close sock and end subprocess pid:7638
SSL read 53 bytes
tun written 53 bytes
Receive from TUN: packet length:53,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 53 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
SSL read 52 bytes
tun written 52 bytes
SSL read 53 bytes
tun written 53 bytes
Receive from TUN: packet length:53,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 53 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3

root@HostU:/# ping 192.168.60.101
PING 192.168.60.101 (192.168.60.101) 56(84) bytes
64 bytes from 192.168.60.101: icmp_seq=1 ttl=63 t
64 bytes from 192.168.60.101: icmp_seq=2 ttl=63 t
^C
--- 192.168.60.101 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss
rtt min/avg/max/mdev = 1.583/1.712/1.841/0.129 ms
root@HostU:/#
```



## 5. 易用性和稳定性

### VPN 客户端虚拟 IP 获取

虚拟 IP 的分配和获取都是自动的，首先在服务端，每次会检查可用的未分配 ip 并分配：

```
int getAvailableIP(char* clientip, ipFlag* flags)
{
    pthread_mutex_lock(&ipFlagMutexWorker->mutex);
    for (int i = 3; i < 255; ++i) {
        if ((flags->area)[i] == 0) {
            sprintf(clientip, "192.168.53.%d", i);
            (flags->area)[i] = 1;
            pthread_mutex_unlock(&ipFlagMutexWorker->mutex);
            return i;
        }
    }
    pthread_mutex_unlock(&ipFlagMutexWorker->mutex);
    return 0;
}
```

```
char clientip[256];
SSL_read(ssl, clientip, sizeof(clientip));
int malloc_ip;
if (0 != (malloc_ip = getAvailableIP(clientip, ipflags))) {
    printf("ip %s is allocated successfully\n", clientip);
    SSL_write(ssl, clientip, sizeof(clientip));
} else {
    printf("All ip is occupied\n");
    strcpy(clientip, "0.0.0.0");
    SSL_write(ssl, clientip, sizeof(clientip));
    SSL_shutdown(ssl);
    SSL_free(ssl);
    close(tcpClientConnectionSock);
    printf("Close sock and end subProcess pid:%d\n", pid);
    return 0; //退出子进程
}
```

在客户端，会接收服务端分配的 IP：

```
char clientip[200];
strcpy(clientip, "192.168.53.2");
SSL_write(ssl, clientip, sizeof(clientip));
int len = SSL_read(ssl, clientip, sizeof(clientip) - 1);
clientip[len] = '\0';
if (0 == strcmp(clientip, "0.0.0.0")) {
    printf("All ip is occupied. There is no ip for this client.\n");
    printf("miniVPN is going to exit.\n");
    exit(1);
}
printf("Malloc ip successfully. Your ip is %s\n", clientip);
```

### VPN 客户端虚拟 IP 配置

程序自动完成。首先 VPN 客户端的虚拟 IP 是由 VPN 服务器动态分配的。客户端在初始时使用默认 IP 与服务器通信，接收服务器分配的实际可用 IP 后，重新配置虚拟网卡 tun0 以使用新的 IP 地址。

得到 server 返回的可用 ip 后，程序会使用新的 ip 重启虚拟网卡 tun0。

```
// 关闭默认IP配置
strcpy(tunWorkString, ipstring);
strcat(tunWorkString, " down");
system(tunWorkString);

// 根据服务器分配的IP配置TUN设备
strcpy(tunWorkString, "ifconfig tun0 ");
strcat(tunWorkString, clientip);
strcat(tunWorkString, "/24 up");
system(tunWorkString);

// 添加内网路由
system("route add -net 192.168.60.0/24 tun0");
```

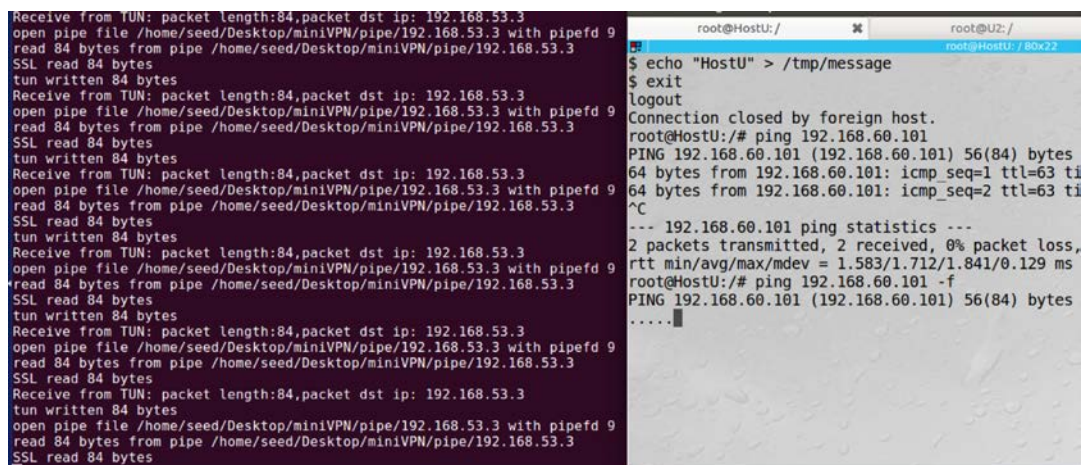
## VPN 客户端内网路由配置

程序自动完成。代码将内网 192.168.60.0/24 的路由通过 tun0 接口添加到路由表中，确保所有指向该网络的流量通过虚拟网络适配器 tun0 转发。

```
// 添加内网路由
system("route add -net 192.168.60.0/24 tun0");
```

## 正常使用时的稳定性

在压力测试下，在 HostU 上执行命令"ping 192.168.60.101 -f"，其中参数"-f"使得程序每 1 秒发送 100 个 ICMP echo 数据包，而不是正常情况下的 1 秒 1 个。服务端没有崩溃，VPN 一切正常。



The image shows two terminal windows. The left window displays a series of network events: receiving packets from a TUN interface, opening pipe files, reading data from pipes, and writing data to the TUN interface. The right window shows a terminal session where a message is sent to 'HostU', the user exits the session, and then performs a continuous ping test to 192.168.60.101. The ping statistics show 2 packets transmitted, 2 received, and 0% packet loss, with a round-trip time (rtt) of approximately 1.583ms.

```
Receive from TUN: packet length:84,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
SSL read 84 bytes
tun written 84 bytes
Receive from TUN: packet length:84,packet dst ip: 192.168.53.3
open pipe file /home/seed/Desktop/miniVPN/pipe/192.168.53.3 with pipefd 9
read 84 bytes from pipe /home/seed/Desktop/miniVPN/pipe/192.168.53.3
SSL read 84 bytes
tun written 84 bytes
```

```
root@HostU: /
root@U2: /
root@HostU: / 80x22
$ echo "HostU" > /tmp/message
$ exit
logout
Connection closed by foreign host.
root@HostU:/# ping 192.168.60.101
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data:
64 bytes from 192.168.60.101: icmp_seq=1 ttl=63 time=1.583 ms
64 bytes from 192.168.60.101: icmp_seq=2 ttl=63 time=1.712 ms
^C
--- 192.168.60.101 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time=1.583 ms
rtt min/avg/max/mdev = 1.583/1.712/1.841/0.129 ms
root@HostU:/# ping 192.168.60.101 -f
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data:
.....
```