

Lab 6.1 Firewall VPN

16307130212 管佳乐

Task 1: VM Setup

VM1: 10.0.2.5 as the client

VM2: 10.0.2.6 as the server

Task 2: Set up Firewall

```
sudo ufw deny out on enp0s3 from 10.0.2.5 to 202.120.224.115
```

No.	Time	Source	Destination	Protocol	Length	Info
1	2018-12-12 21:04:08.723548611	10.0.2.5	202.120.224.26	DNS	72	Standard query 0x5cea A fudan.edu.cn
2	2018-12-12 21:04:08.723602087	10.0.2.5	61.129.42.6	DNS	72	Standard query 0x5cea A fudan.edu.cn
3	2018-12-12 21:04:08.723709885	10.0.2.5	202.120.224.6	DNS	72	Standard query 0x5cea A fudan.edu.cn
4	2018-12-12 21:04:08.727807390	202.120.224.26	10.0.2.5	DNS	149	Standard query response 0x5cea A fudan.edu.cn A 202.120.224.115 NS ns.fudan.edu.cn A 202.12
5	2018-12-12 21:04:08.728011819	202.120.224.6	10.0.2.5	DNS	149	Standard query response 0x5cea A fudan.edu.cn A 202.120.224.115 NS ns.fudan.edu.cn A 202.12
6	2018-12-12 21:04:08.728584345	61.129.42.6	10.0.2.5	DNS	149	Standard query response 0x5cea A fudan.edu.cn A 202.120.224.115 NS ns.fudan.edu.cn A 202.12
7	2018-12-12 21:04:13.962475201	PcsCompu_8f:21:3c	RealtekU_12:35:00	ARP	42	Who has 10.0.2.1? Tell 10.0.2.5
8	2018-12-12 21:04:13.962842226	RealtekU_12:35:00	PcsCompu_8f:21:3c	ARP	60	10.0.2.1 is at 52:54:00:12:35:00

```
Terminal
[12/12/18]seed@VM:~$ ping fudan.edu.cn -c 1
PING fudan.edu.cn (202.120.224.115) 56(84) bytes of data.
ping: sendmsg: Operation not permitted

--- fudan.edu.cn ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

[12/12/18]seed@VM:~$
```

Task 3: Bypassing Firewall using VPN

Step 1: Run VPN Server

```
$ make
$ sudo ./vpnsrver
```

In another terminal

```
$ sudo ifconfig tun0 192.168.53.1/24 up
$ sudo sysctl net.ipv4.ip_forward=1
```

Step 2: Run VPN Client

I changed the server ip, and then start the client program.

```
$ sudo ./vpnclient
```

Configure the interface

```
sudo ifconfig tun0 192.168.53.5/24 up
```

Step 3: Set Up Routing on Client and Server VMs.

Client

```
$ sudo route add -net 202.120.224.0/24 tun0
# fudan.edu.cn
$ sudo route add -net 192.168.53.0/24 tun0
# tunnel subnet
```

Server

```
$ sudo route add -net 192.168.53.0/24 tun0
# tunnel subnet
```

Step 4: Set Up NAT on Server VM

```
$ sudo iptables -F
# Flush the selected chain (all the chains in the table if none is given). This
is equivalent to deleting all the rules one by one.
$ sudo iptables -t nat -F
# Flush nat chain
$ sudo iptables -t nat -A POSTROUTING -j MASQUERADE -o enp0s3
# Append postrouting chain and jump to masquerade
```

Then try to ping fudan.edu.cn. There was a query from fudan.edu.cn. The firewall was bypassed.

```
[12/13/18]seed@VM:~$ ping fudan.edu.cn -c 1
PING fudan.edu.cn (202.120.224.115) 56(84) bytes of data:
64 bytes from 224.fudan.edu.cn (202.120.224.115): icmp_seq=1 ttl=58 time=15.4 ms

--- fudan.edu.cn ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 15.499/15.499/15.499/0.000 ms
[12/13/18]seed@VM:~$
```

The query from 10.0.2.6 was indicated by tunnel

```
[12/13/18]seed@VM:~/.../Atask2$ sudo ./vpnclient 10.0.2.6
[sudo] password for seed:
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
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 the tunnel
```

client, any

1	2018-12-13	20:56:49.944992539	127.0.0.1	127.0.1.1	DNS	74 Standard query 0x6fd2 A fudan.edu.cn
2	2018-12-13	20:56:49.945038500	10.0.2.5	202.120.224.26	DNS	74 Standard query 0x1008 A fudan.edu.cn
3	2018-12-13	20:56:49.945089926	10.0.2.5	61.129.42.6	DNS	74 Standard query 0x1008 A fudan.edu.cn
4	2018-12-13	20:56:49.945112333	10.0.2.5	202.120.224.6	DNS	74 Standard query 0x1008 A fudan.edu.cn
5	2018-12-13	20:56:49.956938304	202.120.224.26	10.0.2.5	DNS	151 Standard query response 0x1008 A fudan.edu.cn A 202.120.224.115 NS ns.fudan.edu.cn A 202.
6	2018-12-13	20:56:49.956952290	202.120.224.6	10.0.2.5	DNS	151 Standard query response 0x1008 A fudan.edu.cn A 202.120.224.115 NS ns.fudan.edu.cn A 202.
7	2018-12-13	20:56:49.956954699	61.129.42.6	10.0.2.5	DNS	151 Standard query response 0x1008 A fudan.edu.cn A 202.120.224.115 NS ns.fudan.edu.cn A 202.
8	2018-12-13	20:56:49.957056065	127.0.1.1	127.0.0.1	DNS	151 Standard query response 0x6fd2 A fudan.edu.cn A 202.120.224.115 NS ns.fudan.edu.cn A 202.
9	2018-12-13	20:56:49.957234947	192.168.53.5	202.120.224.115	ICMP	100 Echo (ping) request id=0x1b06, seq=1/256, ttl=64 (reply in 12)
10	2018-12-13	20:56:49.957367061	10.0.2.5	10.0.2.6	UDP	128 37892 -- 55555 Len=84
11	2018-12-13	20:56:49.962346939	10.0.2.6	10.0.2.5	UDP	128 55555 -- 37892 Len=84
12	2018-12-13	20:56:49.962637539	202.120.224.115	192.168.53.5	ICMP	100 Echo (ping) reply id=0x1b06, seq=1/256, ttl=58 (request in 9)
13	2018-12-13	20:56:49.962819600	127.0.0.1	127.0.1.1	DNS	90 Standard query 0x2b35 PTR 115.224.120.202.in-addr.arpa
14	2018-12-13	20:56:49.962858545	10.0.2.5	61.129.42.6	DNS	90 Standard query 0x80e6 PTR 115.224.120.202.in-addr.arpa
15	2018-12-13	20:56:49.969852362	61.129.42.6	10.0.2.5	DNS	181 Standard query response 0x80e6 PTR 115.224.120.202.in-addr.arpa PTR 224.fudan.edu.cn NS n
16	2018-12-13	20:56:49.970007941	127.0.1.1	127.0.0.1	DNS	181 Standard query response 0x2b35 PTR 115.224.120.202.in-addr.arpa PTR 224.fudan.edu.cn NS n
17	2018-12-13	20:56:50.788678228	10.0.2.5	10.0.2.3	DHCP	344 DHCP Request - Transaction ID 0xf5b29a71
18	2018-12-13	20:56:50.795451642	10.0.2.3	255.255.255.255	DHCP	592 DHCP ACK - Transaction ID 0xf5b29a71
19	2018-12-13	20:56:55.124448008	PcsCompu_33:8c:40		ARP	62 Who has 10.0.2.5? Tell 10.0.2.6
20	2018-12-13	20:56:55.124464916	PcsCompu_8f:21:3c		ARP	44 10.0.2.5 is at 08:00:27:8f:21:3c
21	2018-12-13	20:56:56.037059420	PcsCompu_8f:21:3c		ARP	44 Who has 10.0.2.3? Tell 10.0.2.5

client, tun0

From user's perspective, tun0 was doing the whole thing

1	2018-12-13	21:03:47.506292545	192.168.53.5	202.120.224.115	ICMP	84 Echo (ping) request	id=0x1b33, seq=1/256, ttl=64 (reply in 2)
2	2018-12-13	21:03:47.531981517	202.120.224.115	192.168.53.5	ICMP	84 Echo (ping) reply	id=0x1b33, seq=1/256, ttl=58 (request in 1)

client, enp0s3

In fact, 10.0.2.5 works as a middle man and it is 10.0.2.6 that interacts with fudan.edu.cn

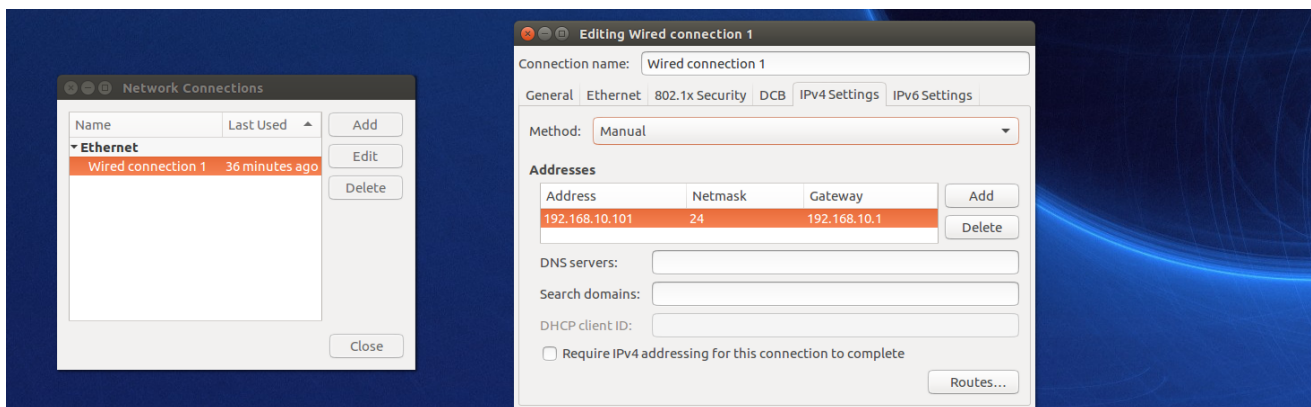
1	2018-12-13	21:04:44.693494997	10.0.2.5	202.120.224.26	DNS	72 Standard query 0x4e4a A fudan.edu.cn	
2	2018-12-13	21:04:44.693547207	10.0.2.5	61.129.42.6	DNS	72 Standard query 0x4e4a A fudan.edu.cn	
3	2018-12-13	21:04:44.693566069	10.0.2.5	202.120.224.6	DNS	72 Standard query 0x4e4a A fudan.edu.cn	
4	2018-12-13	21:04:44.698121157	202.120.224.26	10.0.2.5	DNS	149 Standard query response 0x4e4a A fudan.edu.cn A 202.120.224.115 NS ns.fudan.edu.cn A 202.120.224.115	
5	2018-12-13	21:04:44.698973319	61.129.42.6	10.0.2.5	DNS	149 Standard query response 0x4e4a A fudan.edu.cn A 202.120.224.115 NS ns.fudan.edu.cn A 202.120.224.115	
6	2018-12-13	21:04:44.699379917	10.0.2.5	10.0.2.6	UDP	126 37892 → 55555 Len=84	
7	2018-12-13	21:04:44.699702653	202.120.224.6	10.0.2.5	DNS	149 Standard query response 0x4e4a A fudan.edu.cn A 202.120.224.115 NS ns.fudan.edu.cn A 202.120.224.115	
8	2018-12-13	21:04:44.700043155	10.0.2.6	202.120.224.115	ICMP	98 Echo (ping) request id=0x1b3b, seq=1/256, ttl=63 (reply in 9)	
9	2018-12-13	21:04:44.704535479	202.120.224.115	10.0.2.6	ICMP	98 Echo (ping) reply id=0x1b3b, seq=1/256, ttl=59 (request in 8)	
10	2018-12-13	21:04:44.704933908	10.0.2.6	10.0.2.5	UDP	126 55555 → 37892 Len=84	
11	2018-12-13	21:04:44.705799819	10.0.2.5	61.129.42.6	DNS	88 Standard query 0x9ae5 PTR 115.224.120.202.in-addr.arpa	
12	2018-12-13	21:04:44.710756449	61.129.42.6	10.0.2.5	DNS	179 Standard query response 0x9ae5 PTR 115.224.120.202.in-addr.arpa PTR 224.fudan.edu.cn NS ns.fudan.edu.cn	
13	2018-12-13	21:04:49.749067684	PcsCompu_33:6c:48	PcsCompu_8f:21:3c	ARP	66 Who has 10.0.2.5? Tell 10.0.2.6	

Lab 6.2 VPN

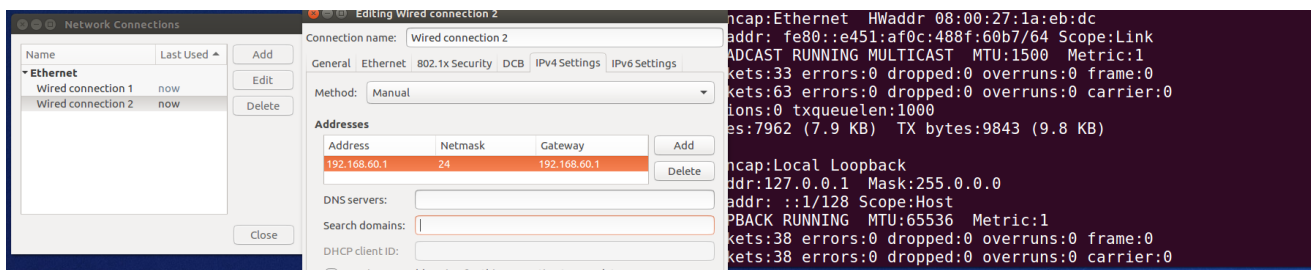
Task 1: VM Setup

	Host U	Server	Host V
NAT	10.0.2.5	10.0.2.6	
Internal		192.169.60.1	192.169.60.101

On Host V



On Server



Task 2: Creating a VPN Tunnel using TUN/TAP

Step 1: Run VPN Server

```
$ make
$ sudo ./vpnservice
```

In another terminal

```
$ sudo ifconfig tun0 192.168.53.1/24 up
sudo sysctl net.ipv4.ip_forward=1
```

Step 2: Run VPN Client

I modified the source code to make it read the arguments, and then start the client program.

```
$ sudo ./vpncclient 10.0.2.6
```

Configure the interface

```
sudo ifconfig tun0 192.168.53.5/24 up
```

Step 3: Set Up Routing on Client and Server VMs.

Client

```
$ sudo route add -net 192.168.60.0/24 tun0
# to the internal
```

Server

```
$ sudo route add -net 192.168.60.0/24 enp0s8
# internal
```

Step 4: Set Up Routing on Host V

The packet will be sent to default gateway in default NIC since it has only one. So no special configuration is needed.

Step 5: Test the VPN Tunnel

Ping

```
ping 192.168.60.101 -c 1
```

```
[12/22/18]seed@VM:~$ ping 192.168.60.101 -c 1
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=1.05 ms

--- 192.168.60.101 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.059/1.059/1.059/0.000 ms
[12/22/18]seed@VM:~$
```

Telnet

```
telnet 192.168.60.101
```

```
[12/22/18]seed@VM:~$ telnet 192.168.60.101
Trying 192.168.60.101...
Connected to 192.168.60.101.
Escape character is '^]'.
Ubuntu 16.04.5 LTS
VM login: seed
Password:
Last login: Mon Nov 12 13:10:20 EST 2018 from 10.0.2.5 on pts/11
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-38-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

146 packages can be updated.
114 updates are security updates.

[12/22/18]seed@VM:~$
```

Step 6: Tunnel-Breaking Test

Keep the connection alive and break the tunnel

I break the tunnel by redirect the route table

```
$ sudo route del -net 192.168.60.0/24 tun0
```

Then my input would get no response. That is because Host V will not receive any packet from Host U, so neither will Host V send any echo to Host U

```
[12/22/18]seed@VM:~$ telnet 192.168.60.101
Trying 192.168.60.101...
Connected to 192.168.60.101.
Escape character is '^]'.
Ubuntu 16.04.5 LTS
VM login: seed
Password:
Last login: Sat Dec 22 01:29:37 EST 2018 from 192.168.53.5 on pts/4
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-38-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

146 packages can be updated.
114 updates are security updates.

[12/22/18]seed@VM:~$ ls
bin          Desktop    examples.desktop  Pictures  Templates
cipher.txt   Documents  NOISE          Public    Videos
Customization Downloads  Music           source

[12/22/18]seed@VM:~$
```

53	2018-12-22 20:44:37.089957479	127.0.0.1	127.0.0.1	DNS	260 Standard query response 0xcd1f A daisy.ubuntu.com A 162.213.33.132 A 162.213.33.108 NS ns3.p2...
54	2018-12-22 20:44:37.969438644	192.168.53.5	192.168.60.101	TELNET	69 Telnet Data ...
55	2018-12-22 20:44:38.173502333	192.168.53.5	192.168.60.101	TELNET	69 Telnet Data ...
56	2018-12-22 20:44:38.382195994	192.168.53.5	192.168.60.101	TCP	70 [TCP Retransmission] 48778 - 23 [PSH, ACK] Seq=3858719655 Ack=4121914625 Win=245 Len=2 TSval=...
57	2018-12-22 20:44:38.892160880	192.168.53.5	192.168.60.101	TCP	70 [TCP Retransmission] 48778 - 23 [PSH, ACK] Seq=3858719655 Ack=4121914625 Win=245 Len=2 TSval=...
58	2018-12-22 20:44:39.633842244	192.168.53.5	192.168.60.101	TCP	70 [TCP Retransmission] 48778 - 23 [PSH, ACK] Seq=3858719655 Ack=4121914625 Win=245 Len=2 TSval=...
59	2018-12-22 20:44:41.119731537	PcsCompu.8f:21:3c		ARP	44 Who has 10.0.2.1? Tell 10.0.2.5
60	2018-12-22 20:44:41.110871412	RealtekU.12:35:00		ARP	62 10.0.2.1 is at 52:54:00:12:35:00
61	2018-12-22 20:44:41.298231763	192.168.53.5	192.168.60.101	TCP	70 [TCP Retransmission] 48778 - 23 [PSH, ACK] Seq=3858719655 Ack=4121914625 Win=245 Len=2 TSval=...
62	2018-12-22 20:44:44.691401068	192.168.53.5	192.168.60.101	TCP	70 [TCP Retransmission] 48778 - 23 [PSH, ACK] Seq=3858719655 Ack=4121914625 Win=245 Len=2 TSval=...
63	2018-12-22 20:44:51.946326449	192.168.53.5	192.168.60.101	TCP	70 [TCP Retransmission] 48778 - 23 [PSH, ACK] Seq=3858719655 Ack=4121914625 Win=245 Len=2 TSval=...
64	2018-12-22 20:45:02.941378771	127.0.0.1	127.0.0.1	DNS	81 Standard query 0xc84c A mirrors.ustc.edu.cn
65	2018-12-22 20:45:02.941407223	127.0.0.1	127.0.0.1	DNS	81 Standard query 0x4375 AAAA mirrors.ustc.edu.cn

Then resume the tunnel by direct the route table back.

```
$ sudo route add -net 192.168.60.0/24 tun0
```

The connection will still work. Since Host V can receive packets now. The interrupt before is transparent to Host V.

```

address.c: In function 'main':
address.c:19:12: warning: format '%d' expects enp0s3
but argument 2 has type 'struct in_addr' [-l
printf("%d",ip->sin_addr);

[12/22/18]seed@VM:~/../test$ gcc address.c
address.c: In function 'main':
address.c:19:5: error: aggregate value used
ted
printf("%d",(int)ip->sin_addr);
lo
[12/22/18]seed@VM:~/../test$ gcc address.c
address.c: In function 'main':
address.c:19:5: error: aggregate value used
ted
printf("%d",(int)(ip->sin_addr));

[12/22/18]seed@VM:~/../test$ sudo route de
[sudo] password for seed:
[12/22/18]seed@VM:~/../test$ sudo route ad[12/22/18]seed@VM:~$

[12/22/18]seed@VM:~/../test$ sudo route de
[12/22/18]seed@VM:~/../test$ sudo route ad
[12/22/18]seed@VM:~/../test$

[12/22/18]seed@VM:~$ ifconfig
Link encap:Ethernet HWaddr 08:00:27:58:3d:bc
inet addr:192.168.60.101 Bcast:192.168.60.255 Mask:255.255.255.0
inet6 addr: fe80::97bf:7188:c5ad:6610/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:759 errors:0 dropped:0 overruns:0 frame:0
TX packets:843 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:56773 (56.7 KB) TX bytes:73439 (73.4 KB)

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:211 errors:0 dropped:0 overruns:0 frame:0
TX packets:211 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:17260 (17.2 KB) TX bytes:17260 (17.2 KB)

```

Task 3: Encrypting the Tunnel

run the server

```

$ make 3
$ sudo ./server3
$ sudo ifconfig tun0 192.168.53.1/24 up
$ sudo sysctl net.ipv4.ip_forward=1

```

```

[12/29/18]seed@VM:~/../lab6$ sudo ./server3
[sudo] password for seed:
TUN setup
TCP Setup
13771: Start
13771: Handshake
13771: Working

```

run the client

```

$ sudo ./client3
# default argument is "serverguan.com", 4433
$ sudo ifconfig tun0 192.168.53.5/24 up
$ sudo route add -net 192.168.60.0/24 tun0
$ ping 192.168.60.101 -c 1

```

```

[12/29/18]seed@VM:~/../lab6$ sudo ./client3
[sudo] password for seed:
TLS Initialized
TCP Connected
Verification passed.
Verification passed.
SSL connection is successful
SSL connection using AES256-GCM-SHA384
^[^A

```

```

[12/29/18]seed@VM:~$ ping 192.168.60.101 -c 1
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=0.772 ms

--- 192.168.60.101 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.772/0.772/0.772/0.000 ms

```

Check the wireshark

121	2018-12-29 20:00:52.421814665	10.0.2.5	10.0.2.6	TLSv1.2	181 Application Data	
122	2018-12-29 20:00:52.422058261	10.0.2.6	10.0.2.5	TCP	68 4433 → 46528 [ACK] Seq=2714389795 Ack=2360432783 Win=243 Len=0 TSval=5554631 TSecr=4294957760	
123	2018-12-29 20:00:52.422366276	10.0.2.6	10.0.2.5	TLSv1.2	181 Application Data	
124	2018-12-29 20:00:52.422377442	10.0.2.5	10.0.2.6	TCP	68 46528 → 4433 [ACK] Seq=2360432783 Ack=2714389908 Win=264 Len=0 TSval=4294957761 TSecr=5554631	
125	2018-12-29 20:00:52.422456243	192.168.60.101	192.168.53.5	ICMP	100 Echo (ping) reply id=0x6a74, seq=1/256, ttl=63 (request in 120)	
126	2018-12-29 20:00:57.517494877	PcsCommv 33.16c:48		ARP	62 Who has 10.0.2.5? Tell 10.0.2.6	

▶ Frame 121: 181 bytes on wire (1448 bits), 181 bytes captured (1448 bits) on interface 0
 ▶ Linux cooked capture
 ▶ Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.6
 ▶ Transmission Control Protocol, Src Port: 46528, Dst Port: 4433, Seq: 2360432670, Ack: 2714389795, Len: 113
 ▶ Secure Sockets Layer
 ▼ TLSv1.2 Record Layer: Application Data Protocol: ssl
 Content Type: Application Data (23)
 Version: TLS 1.2 (0x0303)
 Length: 108
 Encrypted Application Data: d95dcc99dafc51088a9e5e066e1be59a36ae3cc9bf5b23b1...

It is indeed encrypted.

Task 4: Authenticating the VPN Server

Subtask 1: Generate my own certificate

```
# generate ca
$ openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf
# generate server certificate
$ openssl genrsa -out serverguan.key 1024
$ openssl req -new -key serverguan.key -out serverguan.csr -config openssl.cnf
# sign
$ openssl ca -in serverguan.csr -out serverguan.crt -cert ca.crt -keyfile ca.key -
config openssl.cnf
```

Distribution

```
$ openssl x509 -in serverguan.pem -noout -subject_hash
40b51be0
# I'm working on shared folder, soft linking would not work
$ mv serverguan.pem 40b51be0.0
# ca
$ openssl x509 -in ca.pem -noout -subject_hash
$ mv ca.pem bdf70d7e.0
```

Subtask 2: Code demonstration

- Verifying that the server certificate is valid

Here, it stipulates that both client and server's certificates should be checked

```
// sets the verification flags for ctx to be mode and specifies the verify_callback
function to be used
// SSL_VERIFY_PEER means to examine the certificate of both sides.
SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, NULL);
```

Then check the certificate of the server

```
// get and set verification parameters
X509_VERIFY_PARAM *vpm = SSL_get0_param ssl);
X509_VERIFY_PARAM_set1_host(vpm, hostname, 0);
```

- Verifying that the server is the owner of the certificate

```
// Whenever a certificate is verified during a SSL/TLS handshake
// a verification function is called
int verify_callback(int preverify_ok, X509_STORE_CTX *x509_ctx)
{
    char buf[300];

    // returns the certificate in ctx which caused the error or NULL
    // if no certificate is relevant
    X509 *cert = X509_STORE_CTX_get_current_cert(x509_ctx);
    X509_NAME_oneline(X509_get_subject_name(cert), buf, 300);
    printf("subject= %s\n", buf);
    if (preverify_ok == 1)
    {
        printf("Verification passed.\n");
    }
    else
    {
        int err = X509_STORE_CTX_get_error(x509_ctx);
        printf("Verification failed: %s.\n",
            X509_verify_cert_error_string(err));
    }
}
}
```

- Verifying that the server is the intended server

```
#define CHK_SSL(err) \
    if ((err) < 1) \
    { \
        ERR_print_errors_fp(stderr); \
        exit(2); \
    } \
    int err = SSL_connect(ssl); \
    // During the connection \
    CHK_SSL(err);
```

Subtask 3: Failed authentication

I edit the host file to change the intend server

```
$ sudo vi /etc/hosts
10.0.2.6    fudan.edu.cn
```

run the client again

```
$ sudo ./client3 fudan.edu.cn
```

```
Legend: code, data, rodata, value
167      int err = SSL_connect(ssl);
gdb-peda$ n
Verification failed: Hostname mismatch.
[Inferior 1 (process 3386) exited normally]
Warning: not running or target is remote
gdb-peda$
```


During the handshake, it would report the hostname do not match since the server don't hold the key for fudan.edu.cn

Task 5: Authenticating the VPN Client

The login funtion of client

```
void login(SSL *ssl) {
    char username[NAME_LENGTH];
    char password[PASSWORD_LENGTH];
    char request[BUFF_SIZE];
    char reply[BUFF_SIZE];
    int len;

    printf("Your username:\n");
    scanf("%s", username);
    getchar();
    printf("Your password:\n");
    scanf("%s", password);

#ifdef DEBUG
    printf("Username: %s\nPassword: %s\n", username, password);
#endif

    // request
    bzero(request, BUFF_SIZE);
    strcpy(request, username);
    strcat(request, " ");
    strcat(request, password);
    len = strlen(username) + strlen(password) + 1;
    request[len] = '\0';

#ifdef DEBUG
    printf("Request: %s\tLen: %d\n", request, len);
#endif

    SSL_write(ssl, request, len);

    // check reply
    bzero(reply, BUFF_SIZE);
    len = SSL_read(ssl, reply, BUFF_SIZE - 1);
    reply[len] = '\0';

    // fail
    if (strcmp(reply, "success")) {
        printf("Login Failed\n");
        endRequest();
    }
    // success
}
```

The checkLogin function of server

```

int checkLogin(SSL *ssl, int conn) {
    char *pch;
    char username[100];
    char password[1000];
    char request[BUFF_SIZE];

    // if not clear, the string would be weird
    memset(&username, 0, sizeof(username));
    memset(&password, 0, sizeof(password));
    memset(&request, 0, BUFF_SIZE);

    int len = SSL_read(ssl, request, BUFF_SIZE - 1);
    request[len] = '\0';

#ifdef DEBUG
    printf("Logging in:%s\n", request);
#endif

    // username
    pch = strtok(request, " ");
    if (!pch) {
        printf("Invalid username\n");
        return -1;
    }
    strcpy(username, pch);
#ifdef DEBUG
    printf("Username:%s\n", username);
#endif

    // password
    pch = strtok(NULL, " ");
    if (!pch) {
        printf("Invalid password\n");
        return -1;
    }
    strcpy(password, pch);
#ifdef DEBUG
    printf("Password:%s\n", password);
#endif

    // check the shadow
    struct spwd *pw;
    char *epasswd;
    pw = getsppam(username);
    if (pw == NULL) {
        printf("Invalid account\n");
        return -1;
    }

    // return the result
    epasswd = crypt(password, pw->sp_pwdp);
    char *fail = "fail";
    char *success = "success";

```

```

    if (strcmp(epasswd, pw->sp_pwdp)) {
        printf("Username and password do not match\n");
        SSL_write(ssl, fail, strlen(fail));
        return -1;
    }
    SSL_write(ssl, success, strlen(success));
    return 1;
}

```

Run the server

```

$ make 5
$ sudo ./server5
$ sudo ifconfig tun0 192.168.53.1/24 up
$ sudo sysctl net.ipv4.ip_forward=1

```

```

[12/29/18]seed@VM:~/.../lab6$ sudo ./server5
TUN setup
TCP Setup
14931: Start
14931: Handshake
14931: Working

```

Run the client

```

$ sudo ./client5
$ sudo ifconfig tun0 192.168.53.5/24 up
$ sudo route add -net 192.168.60.0/24 tun0

```

```

[12/29/18]seed@VM:~/.../lab6$ sudo ./client5
TLS Initialized
TCP Connected
Verification passed.
Verification passed.
SSL connection is successful
SSL connection using AES256-GCM-SHA384
Your username:
seed
Your password:
dees

```

A failed login demonstration

```

[12/29/18]seed@VM:~/.../lab6$ sudo ./client5
TLS Initialized
TCP Connected
Verification passed.
Verification passed.
SSL connection is successful
SSL connection using AES256-GCM-SHA384
Your username:
seed
Your password:
notdees
Login Failed
[12/29/18]seed@VM:~/.../lab6$

```

```

14934: Start
14934: Handshake
Username and password do not match
14934: Login Fail

```

Task 6: Supporting Multiple Clients

```

+-----+
+-socket1<-----pipe 1-----+          |
|                               |          v
+-socket2<-----pipe 2-----+<--main proc <---tun0
|                               |
+-socket3<-----pipe 3-----+

```

At first, I think there would be 2 pipes for every connection, one for upstream and one for downstream.
But in the upstream, the datagram from socket could be transported to tun0 directly.

```

// 父进程监听 tun0, 一旦听到就写进 pipe
void tun2pipe(int tunfd, int pipefd) {
    int len;
    char buff[BUFF_SIZE];

    bzero(buff, BUFF_SIZE);
    len = read(tunfd, buff, BUFF_SIZE);
    buff[len] = '\0';
    // normal write, no ssl
    write(pipefd, buff, len);
}

// 子进程监听 pipe, 一旦听到就写到 socket
void pipe2socket(int pipefd, int sockfd, SSL *ssl) {
    int len;
    char buff[BUFF_SIZE];

    bzero(buff, BUFF_SIZE);
    len = read(pipefd, buff, BUFF_SIZE);
    buff[len] = '\0';
    write(ssl, buff, len);
}

// 子进程监听 socket, 一旦听到就写到 tun0
void socket2tun(int tunfd, int sockfd, SSL *ssl) {
    int len;
    char buff[BUFF_SIZE];

    bzero(buff, BUFF_SIZE);
    len = SSL_read(ssl, buff, BUFF_SIZE);
    buff[len] = '\0';
    SSL_write(tunfd, buff, len);
}

```