Lab 5 Firewall

> 16307130212 管佳乐

Task 1: Using Firewall

> The firewall is working on 10.0.2.5

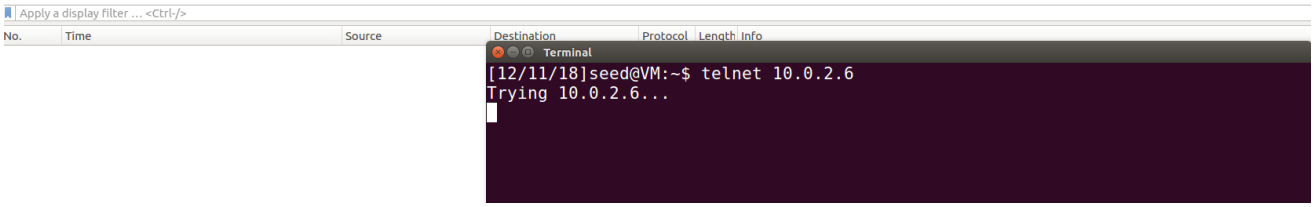
Set up and configure ufw on 10.0.2.5, which is denoted by A

```
$ sudo vi /etc/default/ufw
DEFAULT_INPUT_POLICY="ACCEPT"
```

Task 1.1 Prevent A from doing telnet to Machine B.

```
$ sudo ufw deny out from 10.0.2.5 to 10.0.2.6 port 23
```

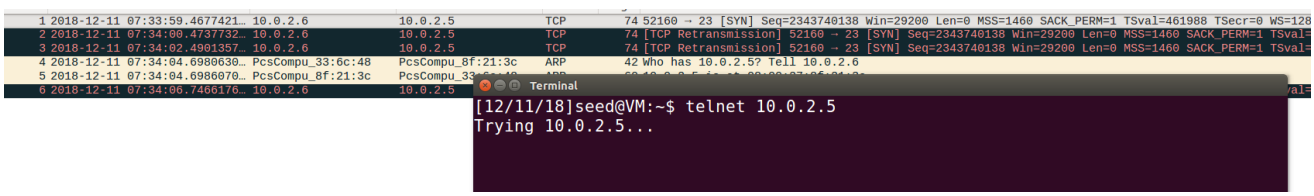
There was no packet record on Wireshark. And the terminal was keep trying. The filter succeeded.



Task 1.2 Prevent B from doing telnet to Machine A.

```
$ sudo ufw deny from 10.0.2.6 to 10.0.2.5 port 23
```

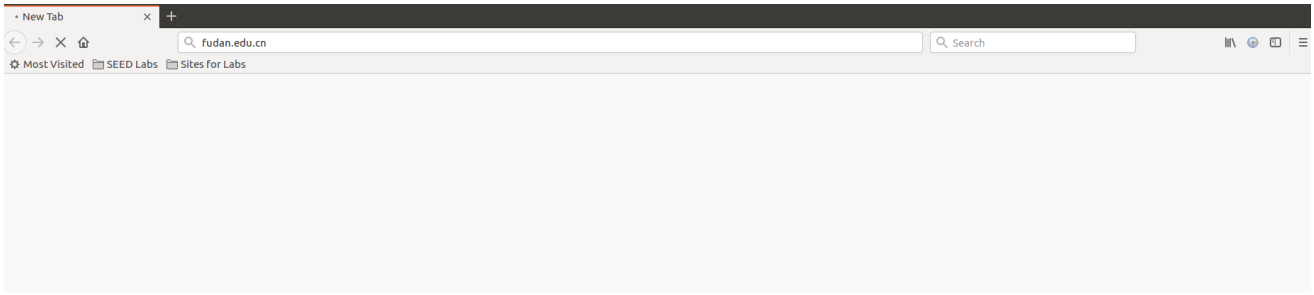
This is a screenshot from 10.0.2.6. B was keep trying. But A has not received it, so it did not reply. Therefore there was continuous retransmissions on wireshark.



Task 1.3 Prevent A from visiting an external web site

```
$ sudo ufw deny out from 10.0.2.5 to 202.120.224.115 port 80
```

I blocked A from visiting university website. So firefox cannot load the page from port 80.



Task 2: Implementing a Simple Firewall

First, delete all the preconfiguration in task 1

```
$ sudo ufw delete 1
```

And then install the kernel mode.

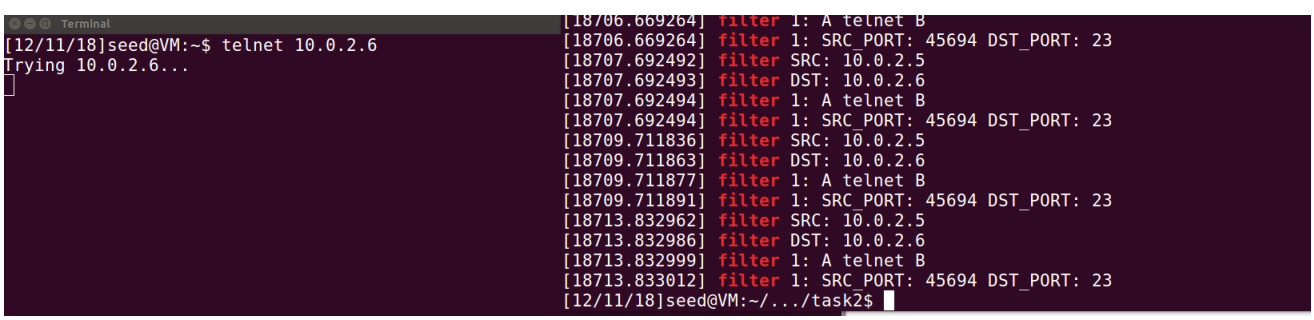
```
$ sudo insmod task2.ko
```

Filter 1 A telnet B

Note that `filter SRC` and `filter DST` display only to indicate the source and destination. While `filter 1` means the packet is dropped.

```
// in hook_func_out tcp segment
// filter 1: A telnet B
      if (dst_port == 23){
         print_address(ip_header); // print SRC and DST
         if (!check_address_src(ip_header, 10, 0, 2, 5){
            printk(KERN_INFO "filter 1: src not match\n");
            return NF_ACCEPT;
         }
         if (!check_address_dst(ip_header, 10, 0, 2, 6)){
            printk(KERN_INFO "filter 1: dst not match\n");
            return NF_ACCEPT;
         }
         printk(KERN_INFO "filter 1: A telnet B\n");
         printk(KERN_INFO "filter 1: SRC_PORT: %d DST_PORT: %d\n", src_port, dst_port);
         return NF_DROP;
      }
```

The left is the working terminal, and the right executes `dmesg | grep filter`



Filter 2 B telnet A

```
// in hook_func_in tcp segment
// filter 2: B telnet A
      if (dst_port == 23){
         print_address(ip_header);
         if (!check_address_src(ip_header, 10, 0, 2, 6)){
            printk(KERN_INFO "filter 2: src not match\n");
            return NF_ACCEPT;
         }
         if (!check_address_dst(ip_header, 10, 0, 2, 5)){
            printk(KERN_INFO "filter 2: dst not match\n");
            return NF_ACCEPT;
         }
         printk(KERN_INFO "filter 2: B telnet A\n");
         printk(KERN_INFO "filter 2: SRC_PORT: %d DST_PORT: %d\n", src_port, dst_port);
         return NF_DROP;
      }
```

The screenshot from 10.0.2.6

```
[12/11/18]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
```

The screenshot from 10.0.2.5

```
dmesg | grep 'filter 2'
```

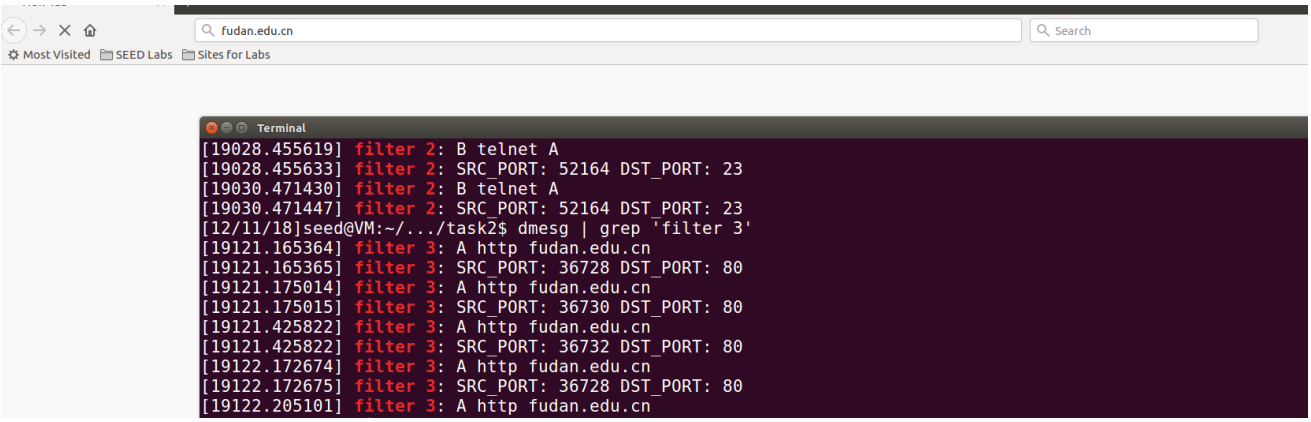
```
[18887.463508] filter 2: A telnet B
[18887.463522] filter 2: SRC_PORT: 52162 DST_PORT: 23
[19027.441384] filter 2: B telnet A
[19027.441402] filter 2: SRC_PORT: 52164 DST_PORT: 23
[19028.455619] filter 2: B telnet A
[19028.455633] filter 2: SRC_PORT: 52164 DST_PORT: 23
[19030.471430] filter 2: B telnet A
[19030.471447] filter 2: SRC_PORT: 52164 DST_PORT: 23
```

Filter 3 A http fudan.edu.cn

```
// in hook_func_out tcp segment
// filter 3: A http fudan.edu.cn(202.120.224.115)
      if (dst_port == 80){
         print_address(ip_header);
         if (!check_address_src(ip_header, 10, 0, 2, 5)){
            printk(KERN_INFO "filter 3: src not match\n");
            return NF_ACCEPT;
         }
         if (!check_address_dst(ip_header, 202, 120, 224, 115)){
            printk(KERN_INFO "filter 3: dst not match\n");
            return NF_ACCEPT;
         }
         printk(KERN_INFO "filter 3: A http fudan.edu.cn\n");
         printk(KERN_INFO "filter 3: SRC_PORT: %d DST_PORT: %d\n", src_port, dst_port);
         return NF_DROP;
```

```
        }
```

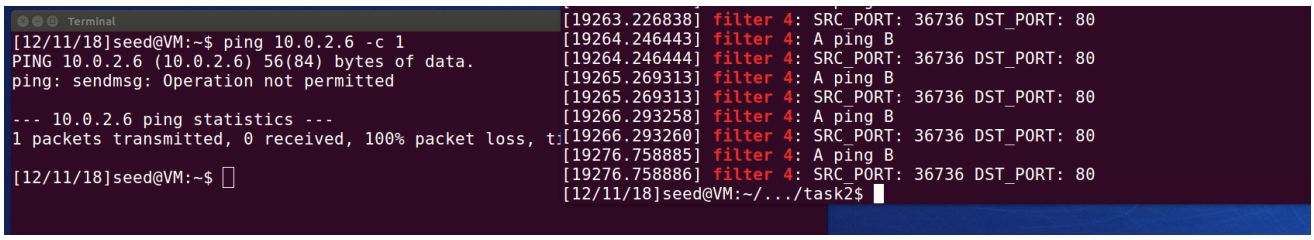
The page was successfully unloaded.



Filter 4 A ping B

```
// in hook_func_out icmp segment
// filter 4: A ping B
      if (icmp_header->type == 8){
         print_address(ip_header);
         if (!check_address_src(ip_header, 10, 0, 2, 5)){
            printk(KERN_INFO "filter 4: src not match\n");
            return NF_ACCEPT;
         }
         if (!check_address_dst(ip_header, 10, 0, 2, 6)){
            printk(KERN_INFO "filter 4: dst not match\n");
            return NF_ACCEPT;
         }
         printk(KERN_INFO "filter 4: A ping B\n");
         printk(KERN_INFO "filter 4: SRC_PORT: %d DST_PORT: %d\n", src_port, dst_port);
         return NF_DROP;
      }
```

Ping was deprived of the privilege to send query.



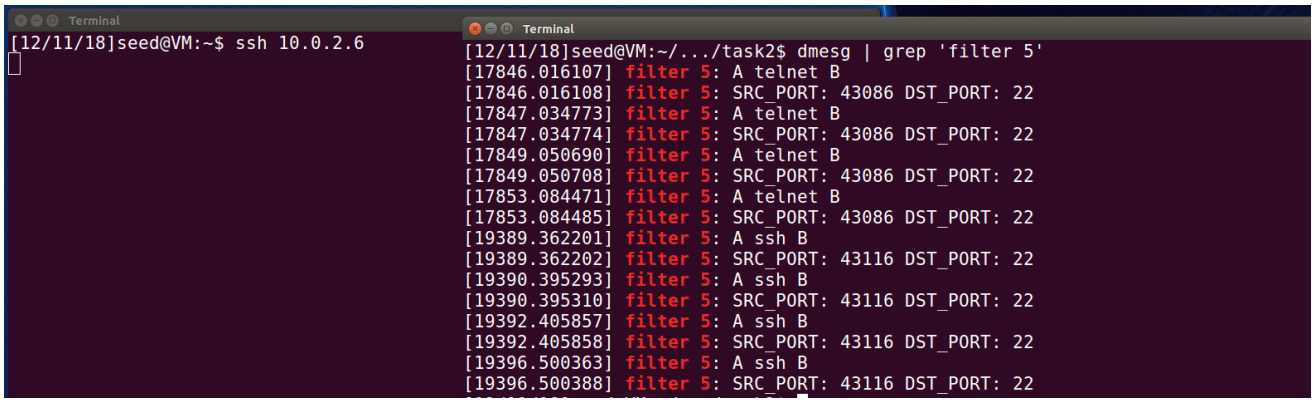
filter 5: A ssh B

```
// in hook_func_out tcp segment
// filter 5: A ssh B
      if (dst_port == 22){
         print_address(ip_header);
         if (!check_address_src(ip_header, 10, 0, 2, 5)){
            printk(KERN_INFO "filter 5: src not match\n");
```

```
            return NF_ACCEPT;
        }
        if (!check_address_dst(ip_header, 10, 0, 2, 6)){
            printk(KERN_INFO "filter 5: dst not match\n");
            return NF_ACCEPT;
        }
        printk(KERN_INFO "filter 5: A ssh B\n");
        printk(KERN_INFO "filter 5: SRC_PORT: %d DST_PORT: %d\n", src_port, dst_port);
        return NF_DROP;
    }
```

ssh was again successfully unsuccessful.

The first few items was the effect of my previous tries.



Task 3: Evading Egress Filtering

Setting the filters and eliminate the effect of previous tasks.

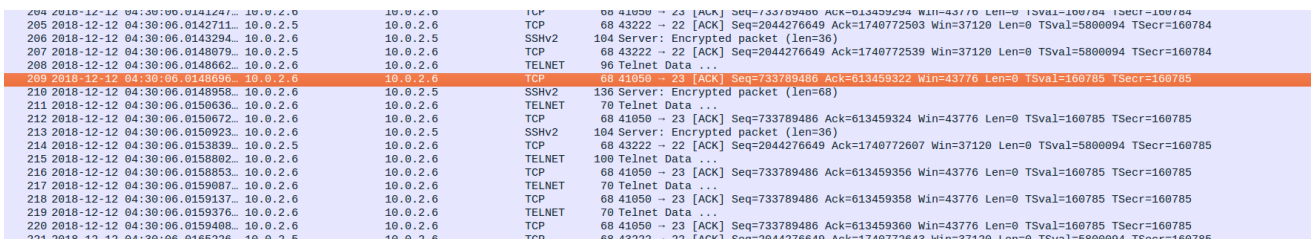
```
$ sudo rmmod task2.ko
$ sudo ufw deny out from 10.0.2.5 to any port 23
# Block all the outgoing traffic to external telnet servers
$ sudo ufw deny out from 10.0.2.5 to 202.120.224.115
# Block all the outgoing fudan.edu.cn
```

Task 3.a: Telnet to Machine B through the firewall

```
$ sh -L 8000:10.0.2.6:23 seed@10.0.2.6
```

In another terminal, I tryed to connect port 8000

```
$ telnet localhost 8000
```



From the screenshot from wireshark, we can tell that ssh acts as a bridge for the real telnet connection.

Every time A send a telnet message, shh will do the communiation things.

```
A         Virtual B       C
43222 -- (22   41050) -- 23
```

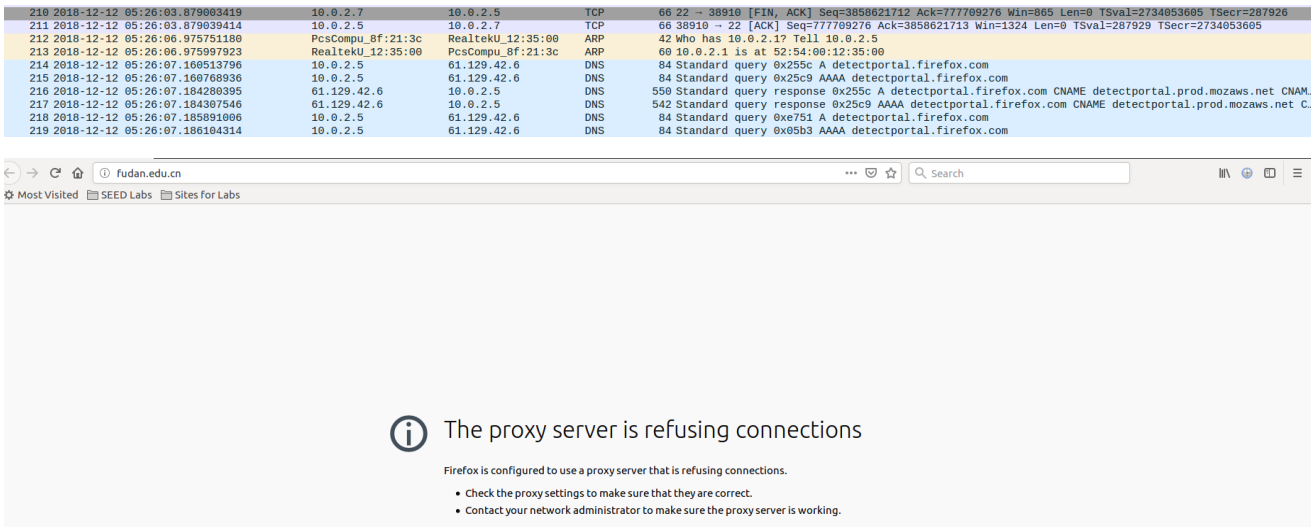
Task 3.b: Connect to Facebook using SSH Tunnel.

designate 9000 as the proxy port

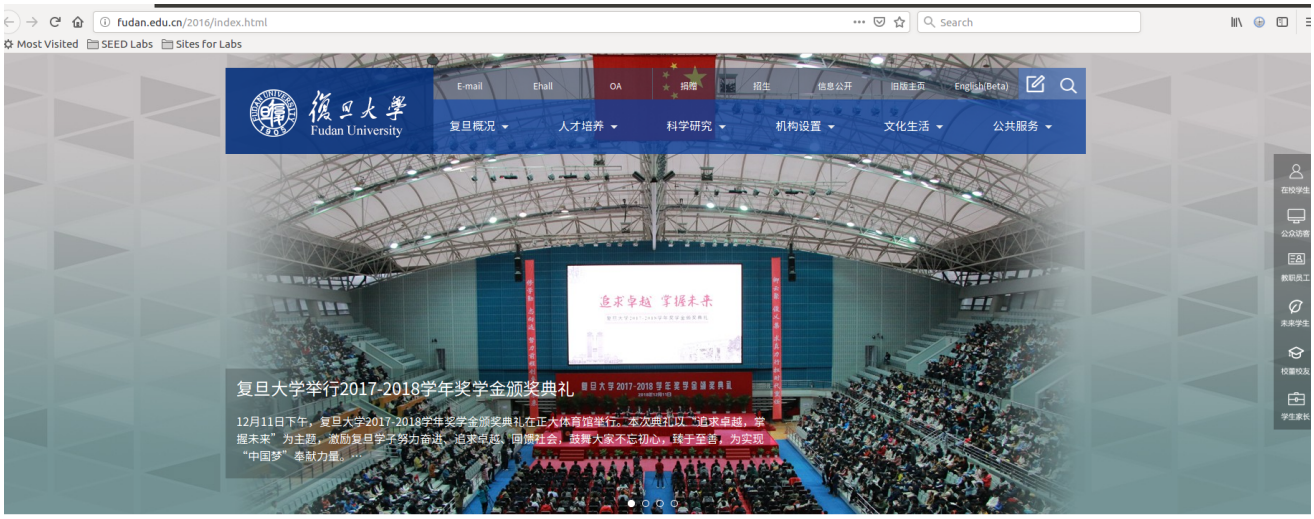
```
ssh -D 9000 -C seed@10.0.2.7
# D means dynamic port forwarding
```

Then the firefox can load fudan.edu.cn

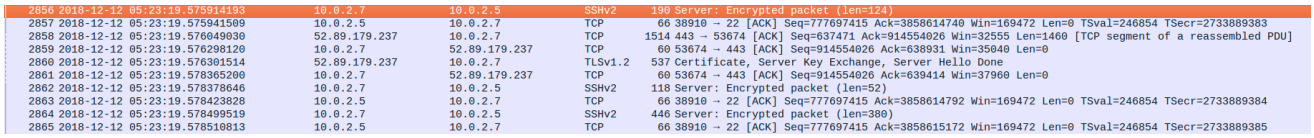
After I turned off the ssh session, there was no new packets since A was denied from browsing fudan.edu.cn



After I restarted the session, the website can be loaded again.

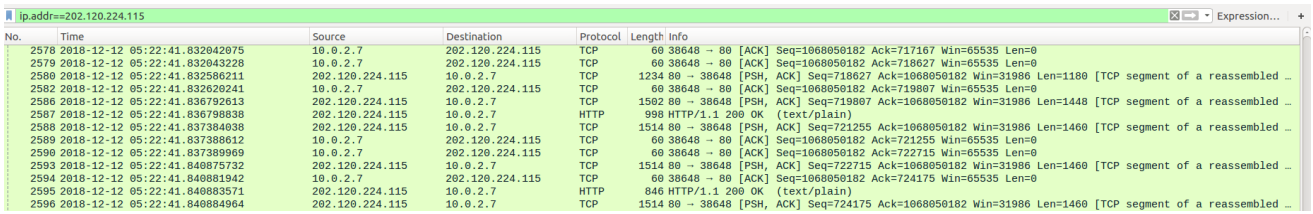


During the session, 10.0.2.7 acts as the proxy.



2856	2018-12-12 05:23:19.575914193	10.0.2.7	10.0.2.5	SSHv2	190 Server: Encrypted packet (len=124)
2857	2018-12-12 05:23:19.575941509	10.0.2.5	10.0.2.7	TCP	66 38910 → 22 [ACK] Seq=777697415 Ack=3858614740 Win=169472 Len=0 TSval=246854 TSecr=2733889383
2858	2018-12-12 05:23:19.576049030	52.89.179.237	10.0.2.7	TCP	1514 443 → 53674 [ACK] Seq=637471 Ack=914554026 Win=32555 Len=1460 [TCP segment of a reassembled PDU]
2859	2018-12-12 05:23:19.576298120	10.0.2.7	52.89.179.237	TCP	60 53674 → 443 [ACK] Seq=914554026 Ack=638931 Win=35040 Len=0
2860	2018-12-12 05:23:19.576301514	52.89.179.237	10.0.2.7	TLSv1.2	537 Certificate, Server Key Exchange, Server Hello Done
2861	2018-12-12 05:23:19.578365200	10.0.2.7	52.89.179.237	TCP	60 53674 → 443 [ACK] Seq=914554026 Ack=639414 Win=37960 Len=0
2862	2018-12-12 05:23:19.578378646	10.0.2.7	10.0.2.5	SSHv2	118 Server: Encrypted packet (len=52)
2863	2018-12-12 05:23:19.578423828	10.0.2.5	10.0.2.7	TCP	66 38910 → 22 [ACK] Seq=777697415 Ack=3858614792 Win=169472 Len=0 TSval=246854 TSecr=2733889384
2864	2018-12-12 05:23:19.578499519	10.0.2.7	10.0.2.5	SSHv2	446 Server: Encrypted packet (len=380)
2865	2018-12-12 05:23:19.578510813	10.0.2.5	10.0.2.7	TCP	66 38910 → 22 [ACK] Seq=777697415 Ack=3858615172 Win=169472 Len=0 TSval=246854 TSecr=2733889385

And it is 10.0.2.7 that make the query to fudan.edu.cn



ip.addr==202.120.224.115 Expression…

No.	Time	Source	Destination	Protocol	Length	Info
2578	2018-12-12 05:22:41.832042075	10.0.2.7	202.120.224.115	TCP	60	38648 → 80 [ACK] Seq=1068050182 Ack=717167 Win=65535 Len=0
2579	2018-12-12 05:22:41.832043228	10.0.2.7	202.120.224.115	TCP	60	38648 → 80 [ACK] Seq=1068050182 Ack=718627 Win=65535 Len=0
2580	2018-12-12 05:22:41.832586211	202.120.224.115	10.0.2.7	TCP	1234	80 → 38648 [PSH, ACK] Seq=718627 Ack=1068050182 Win=31986 Len=1180 [TCP segment of a reassembled …
2582	2018-12-12 05:22:41.832620241	10.0.2.7	202.120.224.115	TCP	60	38648 → 80 [ACK] Seq=1068050182 Ack=719807 Win=65535 Len=0
2586	2018-12-12 05:22:41.836792613	202.120.224.115	10.0.2.7	TCP	1502	80 → 38648 [PSH, ACK] Seq=719807 Ack=1068050182 Win=31986 Len=1448 [TCP segment of a reassembled …
2587	2018-12-12 05:22:41.836798838	202.120.224.115	10.0.2.7	HTTP	998	HTTP/1.1 200 OK (text/plain)
2588	2018-12-12 05:22:41.837384038	202.120.224.115	10.0.2.7	TCP	1514	80 → 38648 [PSH, ACK] Seq=721255 Ack=1068050182 Win=31986 Len=1460 [TCP segment of a reassembled …
2589	2018-12-12 05:22:41.837388612	10.0.2.7	202.120.224.115	TCP	60	38648 → 80 [ACK] Seq=1068050182 Ack=721255 Win=65535 Len=0
2590	2018-12-12 05:22:41.837389969	10.0.2.7	202.120.224.115	TCP	60	38648 → 80 [ACK] Seq=1068050182 Ack=722715 Win=65535 Len=0
2593	2018-12-12 05:22:41.840875732	202.120.224.115	10.0.2.7	TCP	1514	80 → 38648 [PSH, ACK] Seq=722715 Ack=1068050182 Win=31986 Len=1460 [TCP segment of a reassembled …
2594	2018-12-12 05:22:41.840881942	10.0.2.7	202.120.224.115	TCP	60	38648 → 80 [ACK] Seq=1068050182 Ack=724175 Win=65535 Len=0
2595	2018-12-12 05:22:41.840883571	202.120.224.115	10.0.2.7	HTTP	846	HTTP/1.1 200 OK (text/plain)
2596	2018-12-12 05:22:41.840884964	202.120.224.115	10.0.2.7	TCP	1514	80 → 38648 [PSH, ACK] Seq=724175 Ack=1068050182 Win=31986 Len=1460 [TCP segment of a reassembled …

Task 4: Evading Ingress Filtering

The working model shows as below.

```
A       |                   B
        |<---HTTP----
        |<---SSH-----
       -------SSH---->
```

Then configure ufw.

```
$ sudo ufw deny in http
# Block all the incoming traffic of http
$ sudo ufw deny in ssh
# Block all the incoming traffic of ssh
```

> I referred to this site https://blog.ansheng.me/article/ssh-tunnel/ to learn the basic implementation of reverse tunnel

Setup a reverse tunnel on A.

```
$ ssh -p 22 -qngfNTR 7000:localhost:22 seed@10.0.2.7
```

B could connect with A then using the reservse tunnel.

```
$ ssh -p 7000 seed@localhost
```