**ARP Cache Poisoning Attack Lab**

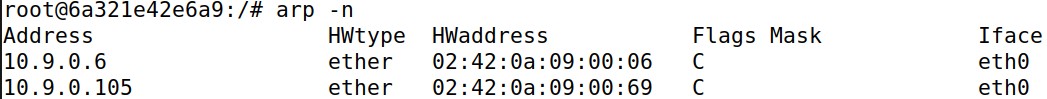
# Task1

1.A

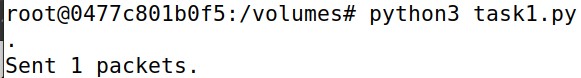
代码如下，op=1 代表是 ARP request。



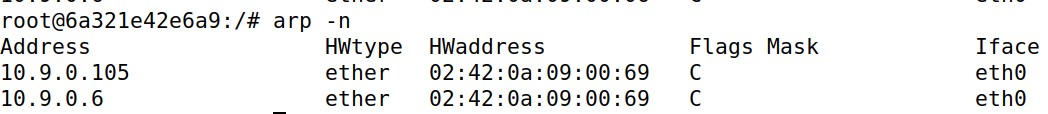
在没有运行前，先用 ping 命令将两者地址加入 cache，看一下两者的 MAC 地址，两者是不一样的，如下。



在 M 里运行代码，发送了一个包，如下。



此时再看两者的 MAC 地址，发现 B 的 MAC 地址已经被改成 M 的了，攻击成功。

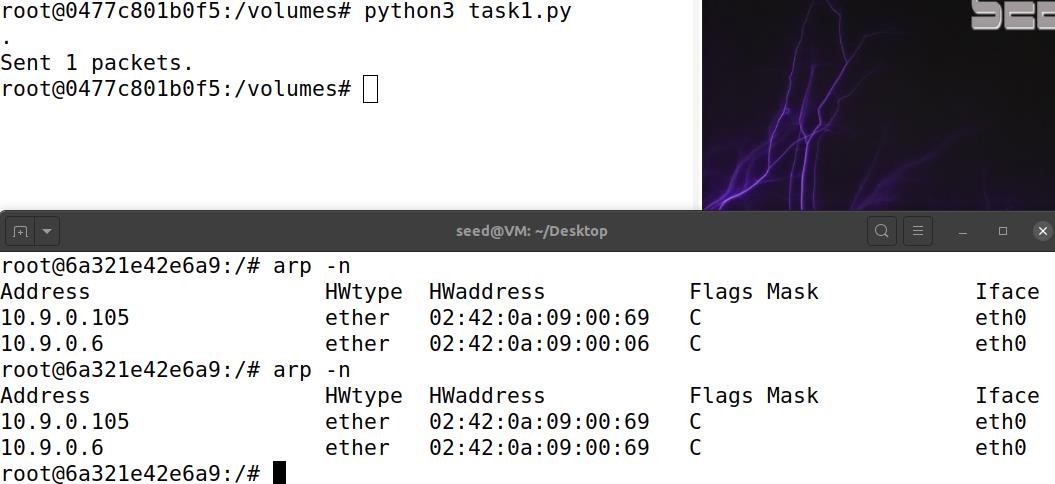


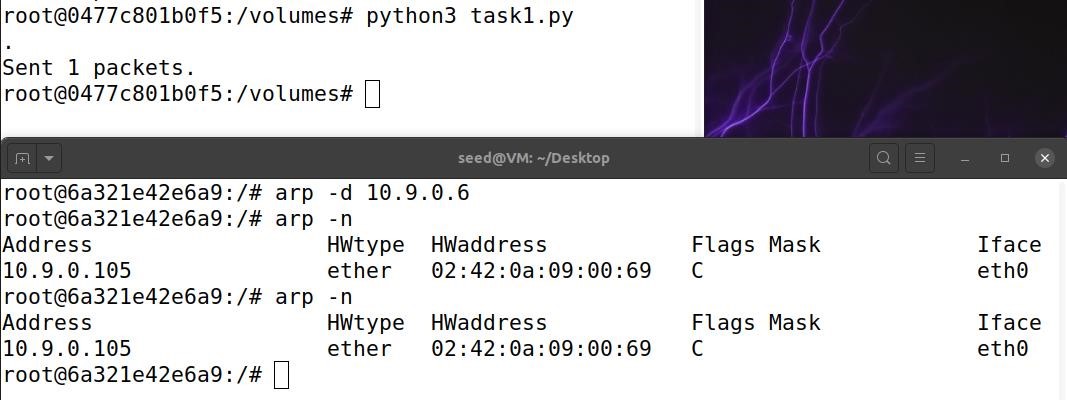
1.B

修改代码如下，op=2 表示是 ARP reply。



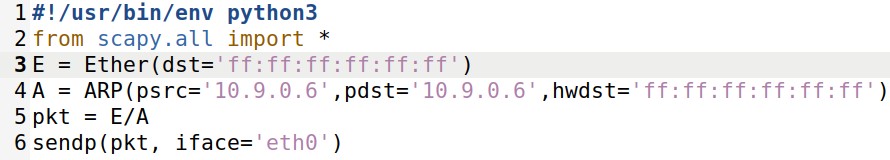
首先是 B 的地址在 A 的里面的情况，先看一下 cache，发现 B 和 M 此时的地址是不一样的，然后在 M 里运行代码，再看一下 cache，发现 B 的地址已经被改成 M 的地址，攻击成功，如下。

 接着是 B 的地址不在 A 里面的情况，先用 arp -d 命令删除 B 的地址，此时查看 cache，只有 M 的地址存在，然后在 M 里运行代码，再次查看 cache，发现依然只有 M 的地址，攻击失败。

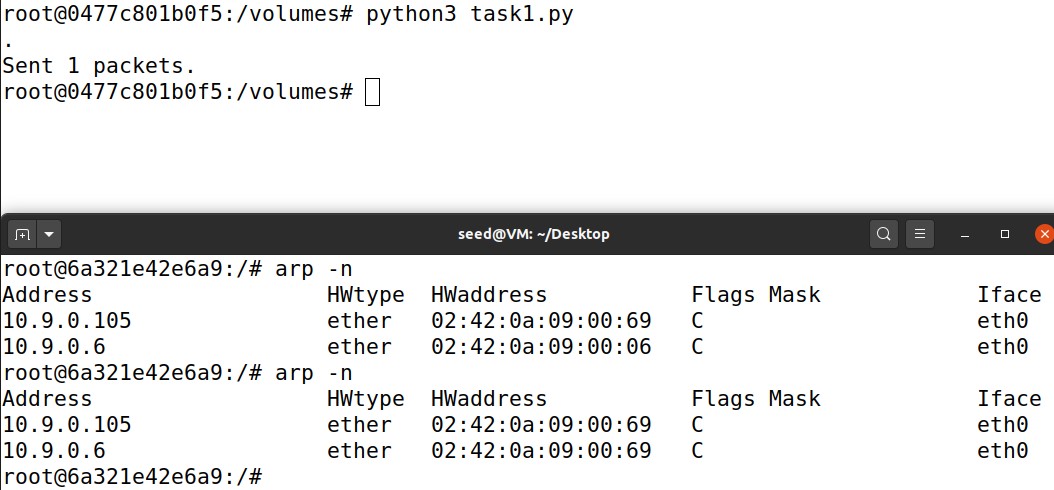


1.C

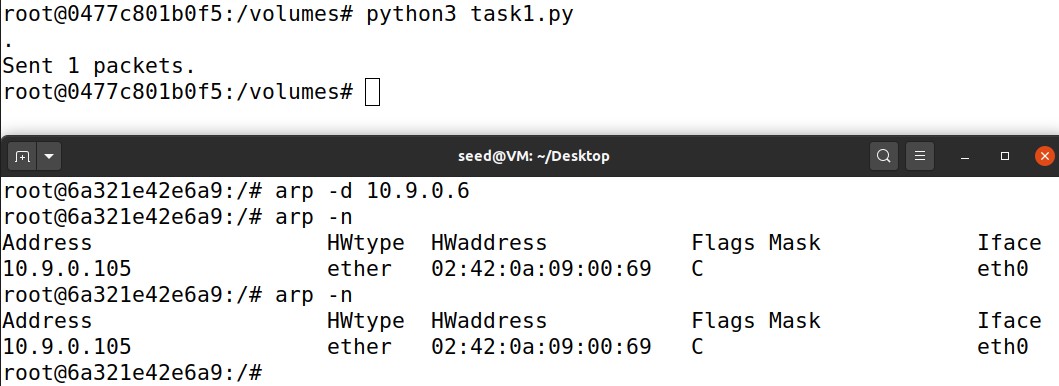
根据题目提示，修改代码如下。



同样分两种情况，首先是 B 的地址在 A 的里面的情况，先看一下 cache，发现 B 和 M 此时的地址是不一样的，然后在 M 里运行代码，再看一下 cache，发现 B 的地址已经被改成 M 的地址，攻击成功，如下。



接着是 B 的地址不在 A 里面的情况，先用 arp -d 命令删除 B 的地址，此时查看 cache，只有 M 的地址存在，然后在 M 里运行代码，再次查看 cache，发现依然只有 M 的地址，攻击失败。

 得出结论，只有所要攻击的地址在 cache 中时，攻击才能成功。

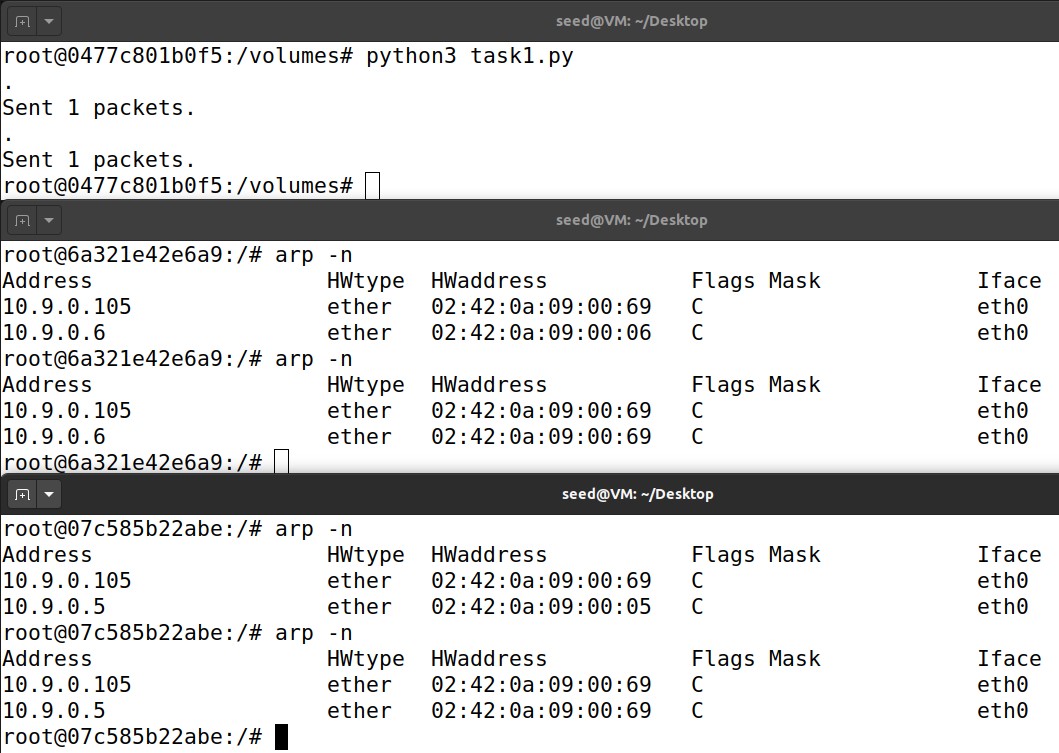
# Task2

Step1

修改代码如下。



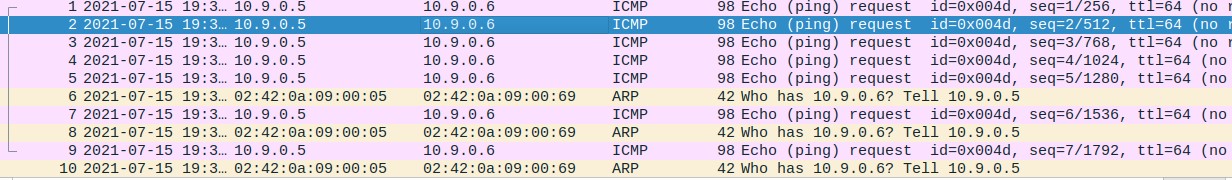
首先看一下 A 和 B 两者的 cache，发现彼此和 M 的 MAC 地址都是不同的，然后在 M 里运行代码，向 A 和 B 各发了一个包，在查看 cache，此时两者 cache 中对方的地址都变成了 M 的地址，如下。

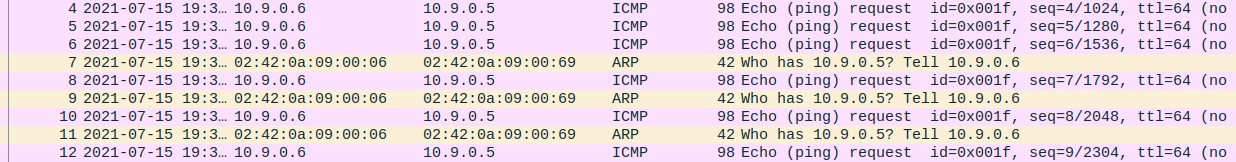


Step2

首先将 IP 转发关掉，如下。

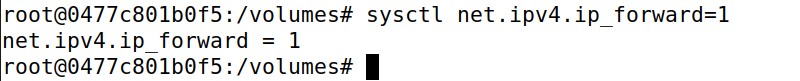
 此时在 A 上 ping B 时，wireshark 抓包如下，没有回应。

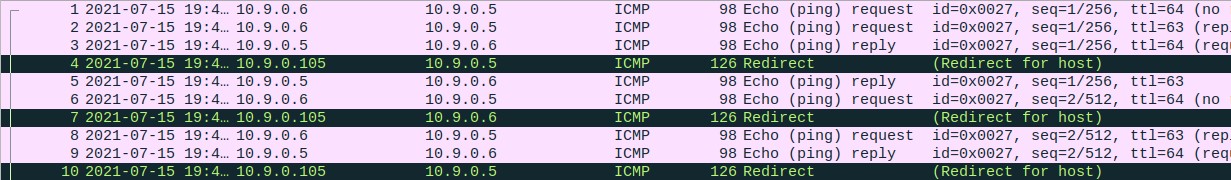
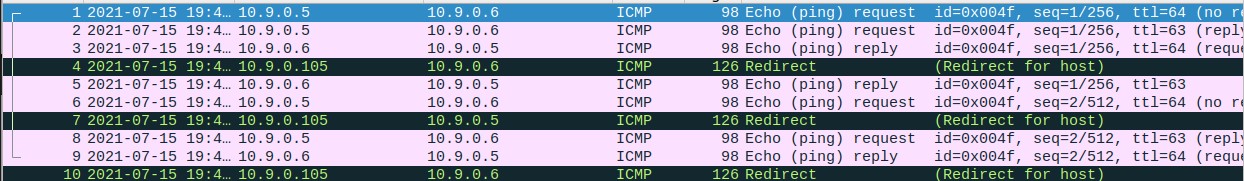
 同样的，在 B 上 ping A，也没有回应，如下。



Step3

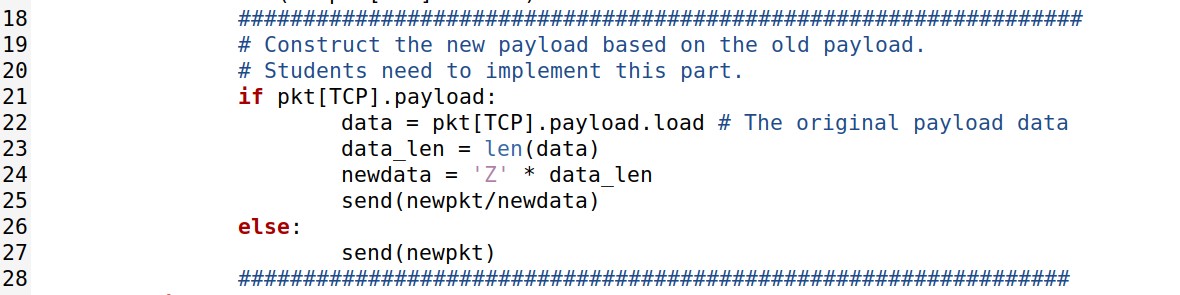
打开 IP 转发，如下。

 如下图所示，无论是在 A ping B，还是在 B ping A，由于有了中间人的转发，两者能够相互 ping 通。

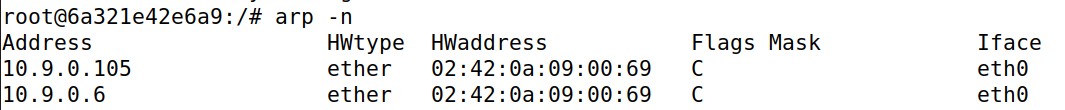
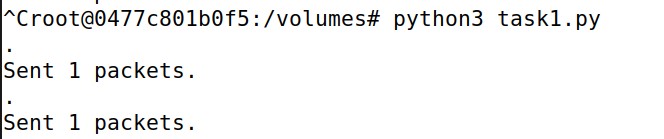


Step 4

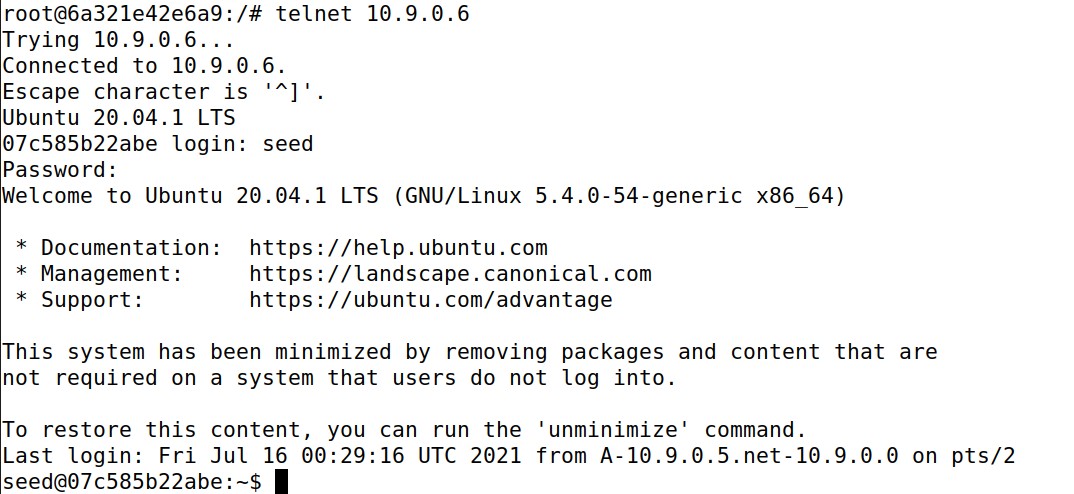
修改所给代码如下部分。

 先用 Task1 的代码，修改 A 的 cache，如下。

打开 IP 转发，如下。



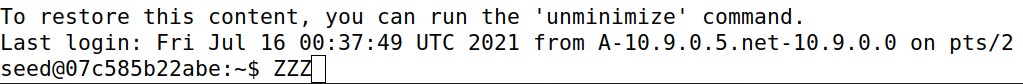
 在 A 上 telnet B，如下。

 关闭 IP 转发，如下。

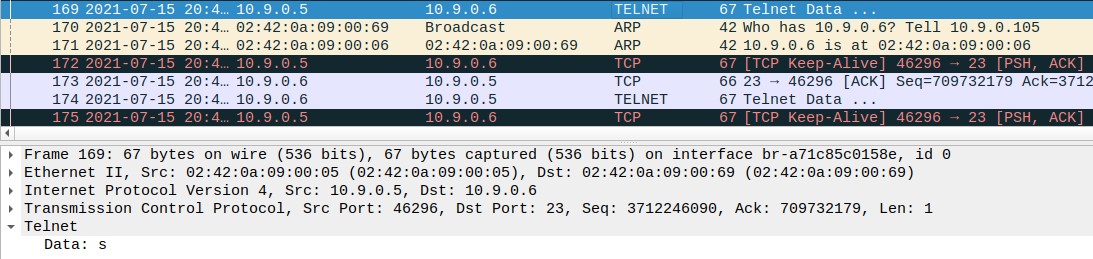
 在 M 里运行代码，此时代码没有任何响应，如下。

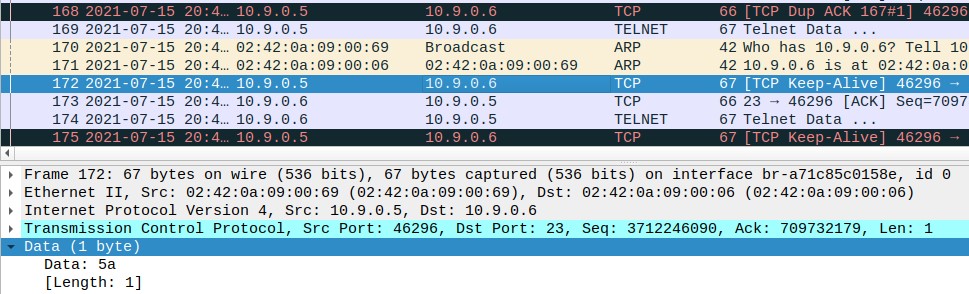


然后到 telnet 里面随意输入字符，发现显示出来的都是 Z 且伴随有卡顿现象，如下。

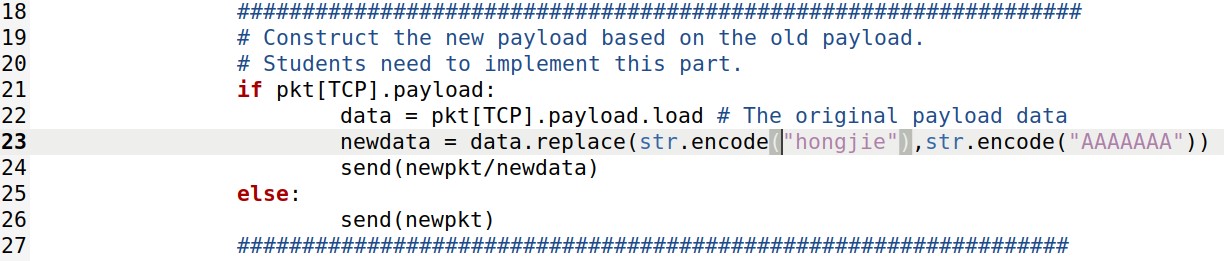
 回到 M 里面，发现代码有了响应，发送了很多包，如下。

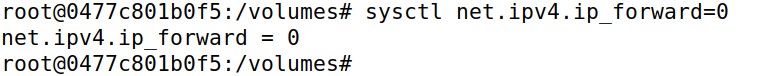
 用 wireshark 抓一下包，如下第 169 个，我们实际输入的内容是字母 s（图中 Data）。

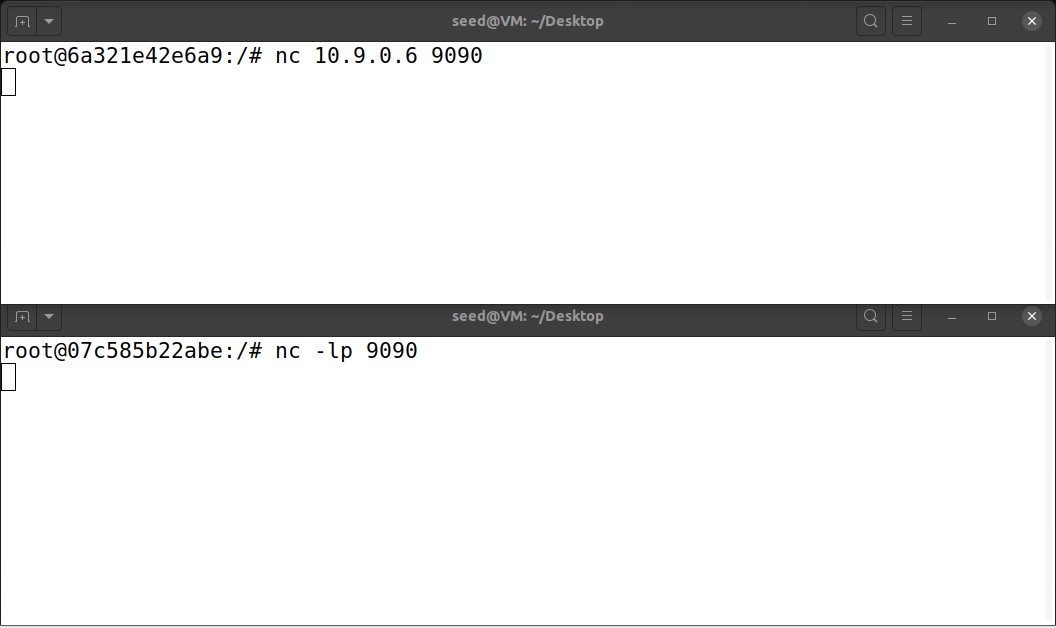
 但是在 172 个，得到的响应却返回的是 ASCII 码 5a，即 z，如下，攻击成功。

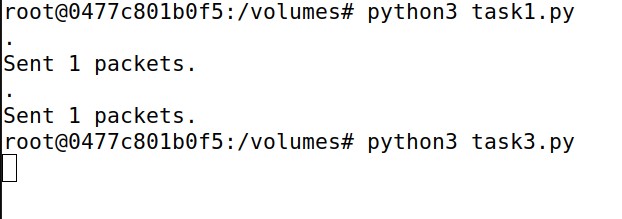


**Task3** 修改如图部分代码。

 将 IP 转发关掉，如下。

 分别到 B 和 A 里面输入如下命令，上面是 A，下面是 B。

 然后到 M 里面，先同 Task1 进行 ARP 欺骗，然后运行代码，此时代码没有响应，如下。



接着，到 A 里面输入携带自己名字的字符串并发送，在 B 里面接收到时，名字被替换成了 A’字符串，如下。

 回到 M 里面，此时代码显示发送了很多包，攻击成功，如下。

