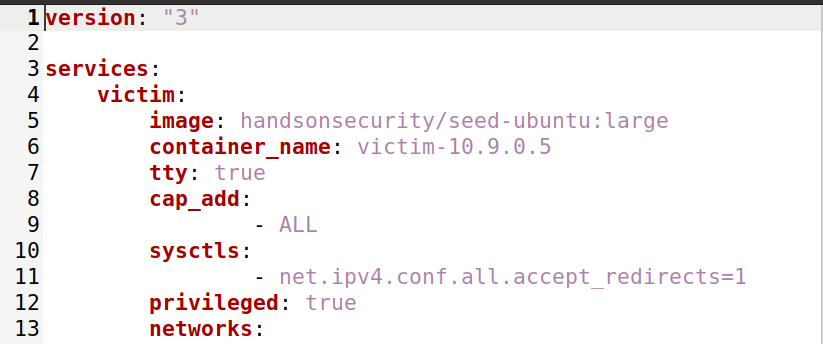
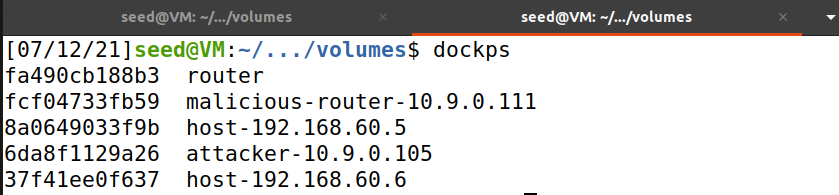
**ICMP Redirect Attack Lab**

**Task 1: Launching ICMP Redirect Attack**

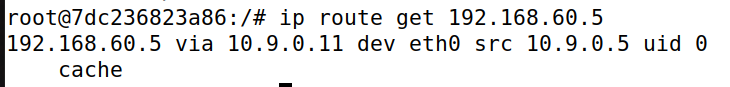
首先关闭countermeasure





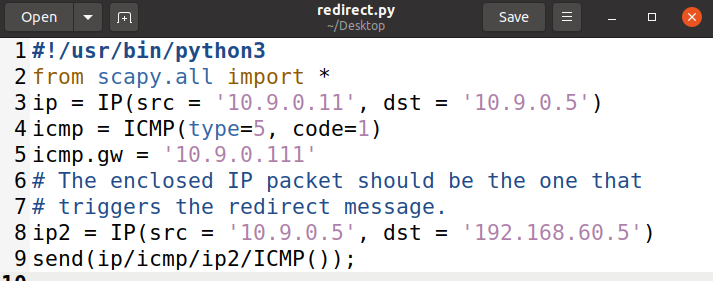
从攻击者容器攻击受害者容器，受害者容器将使用路由器192.168.60.11作为路由器来访问192.168.60.0/24网络

通过ip route命令，我们可以看到路由的默认网关地址是10.9.0.11，下面我们需要修改重定向，使得下一跳地址变为恶意路由。

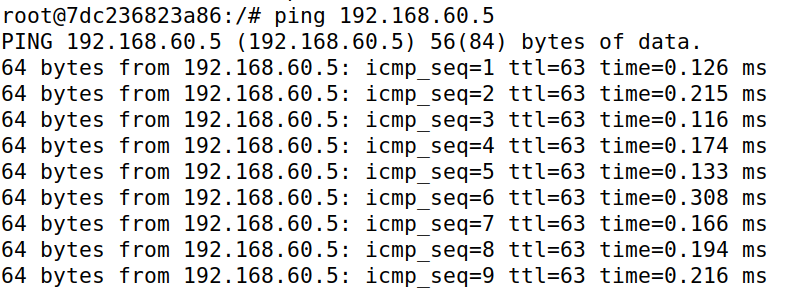


编写脚本，其中IP的src是网关（伪装成默认网关），dst是受害者主机10.9.0.5，icmp.gw是攻击主机，也就是一个恶意路由

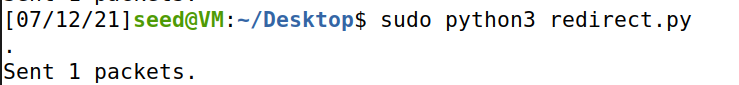
Ip2的src是受害主机10.9.0.5，dst是主机192.168.60.5



通过容器登录受害主机，ping 192.168.10.5，确保路线可达，在ping的同时在VM中运行脚本



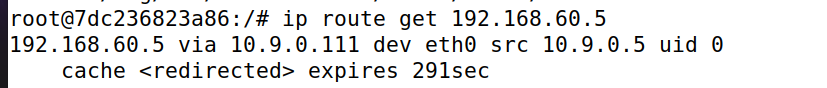
脚本发送成功



通过抓包可以看出是一个Redirect包

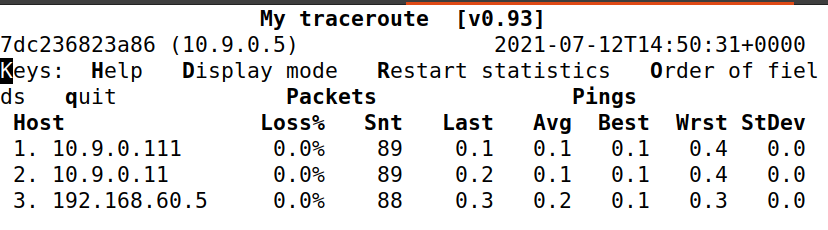


再次运行ip route命令查看路由路径，成功从10.9.0.11变成了恶意路由10.9.0.111

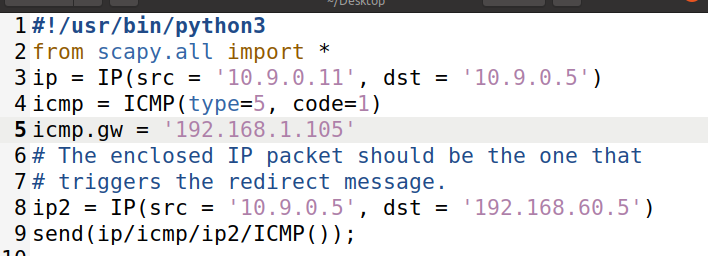


运行mtr命令，进行路径追踪，查看是否重定向成功

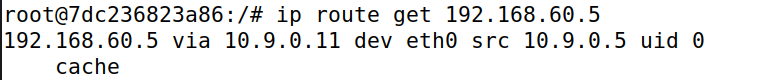
发现10.9.0.111成功加入路径，由此可知，重定向攻击成功



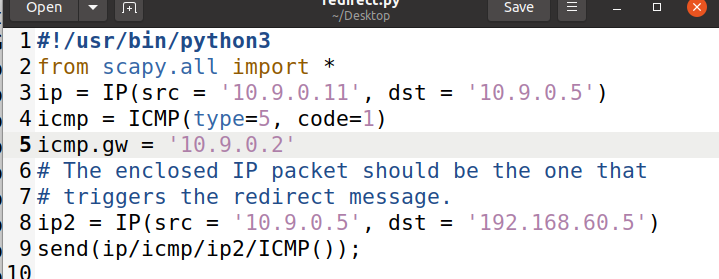
Q1：使用远程主机（非子网内主机）进行尝试



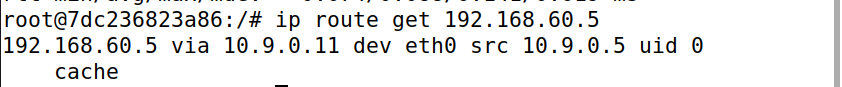
发现又变回了默认网关，说明对于重定向的包，系统会进行一个判断，鉴别是否在同一子网内，是可达的，进而在进行重定向



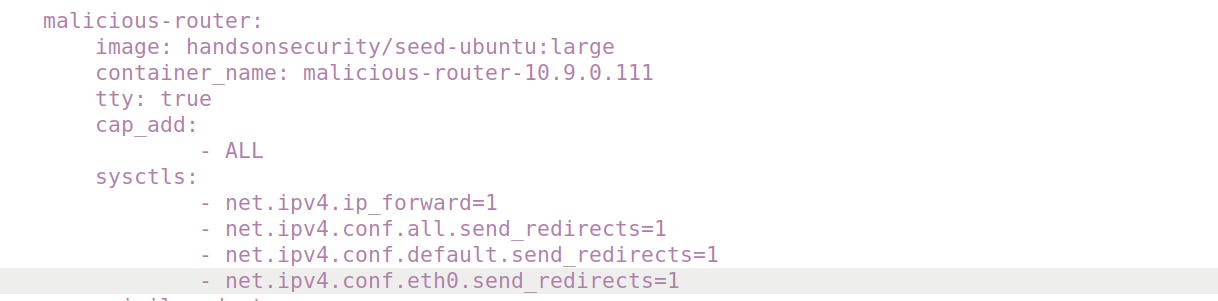
Q2：使用一个在子网内不存在的机器进行重定向尝试



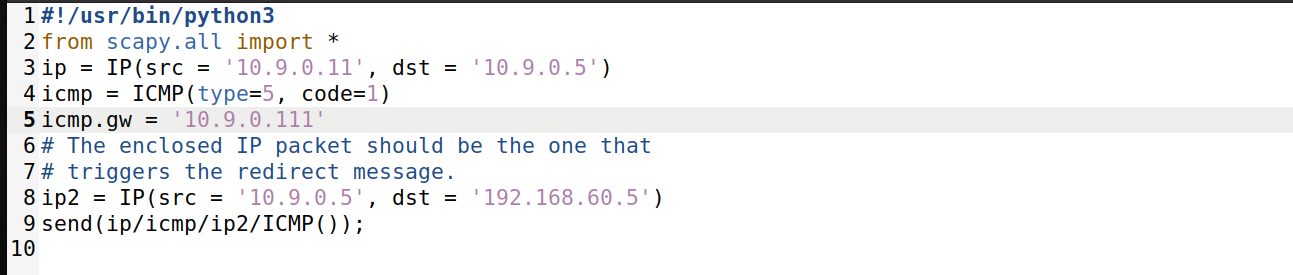
再次运行，发现还是使用的默认网关



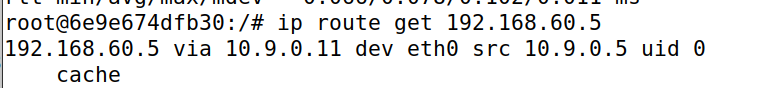
Q3：修改文件中有关重定向的句柄的值



执行脚本也进行相应的修改，重新启动docker，并再次发动攻击



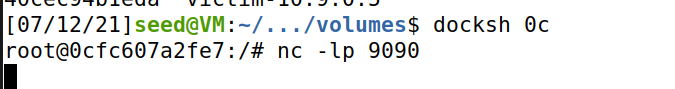
运行脚本，发现重定向攻击并没有成功修改，由此可知，相关的保护措施阻止了重定向攻击的实现

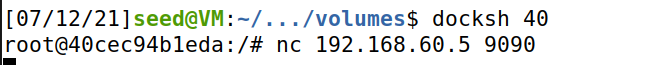


由此推断这些表项是由于确定重定向功能的开关以及闭合，为1时，系统将拒绝对路由表的修改。

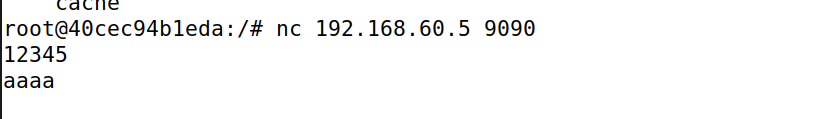
**Task 2: Launching the MITM Attack**

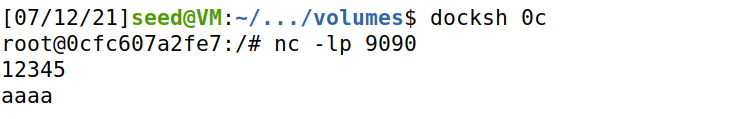
通过重定向攻击，我们可以让受害者使用我们的恶意路由作为达到目的主机的一个中间路由，因此，所有从受害主机发到目的主机的包都会经过恶意路由，因此我们可以修改受害者的报文。



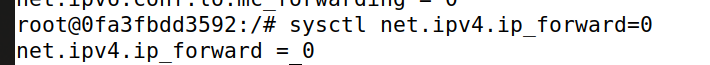


在未进行重定向攻击时，在受害者主机上输入，会在目的主机上接收到



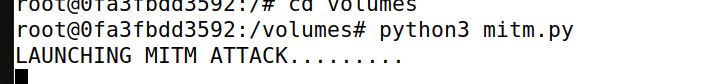


将恶意路由上的IP转发功能关闭

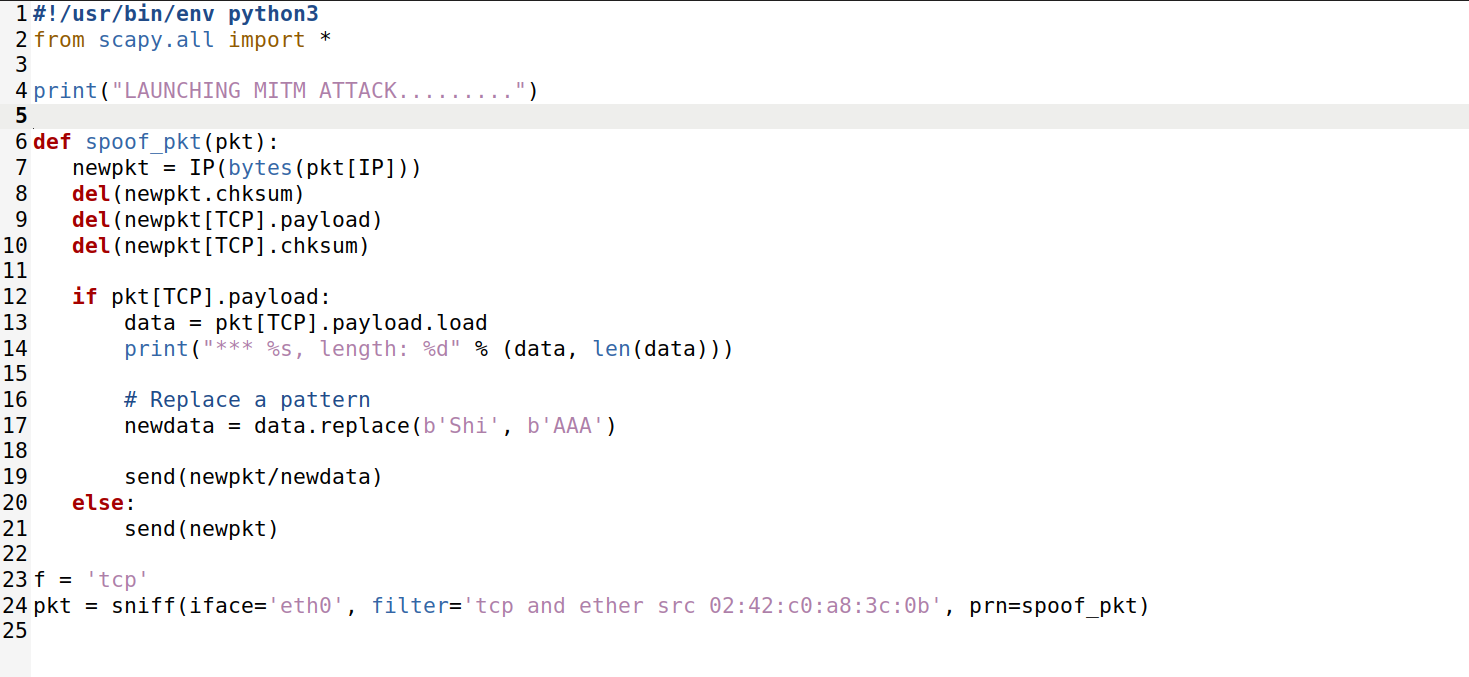


下面我们对受害主机进行重定向攻击，将报文流量转接到恶意主机10.9.0.111

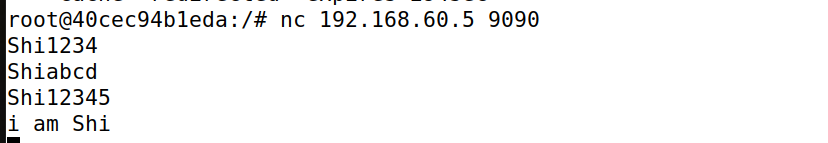
开启端口监听后，运行脚本

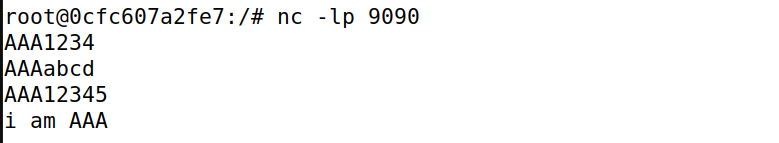


脚本代码如下，内容为将字符“Shi”替换成“AAA”



查看运行结果，发现在受害主机上的输出会经由恶意路由修改后转发，说明中间人攻击成功



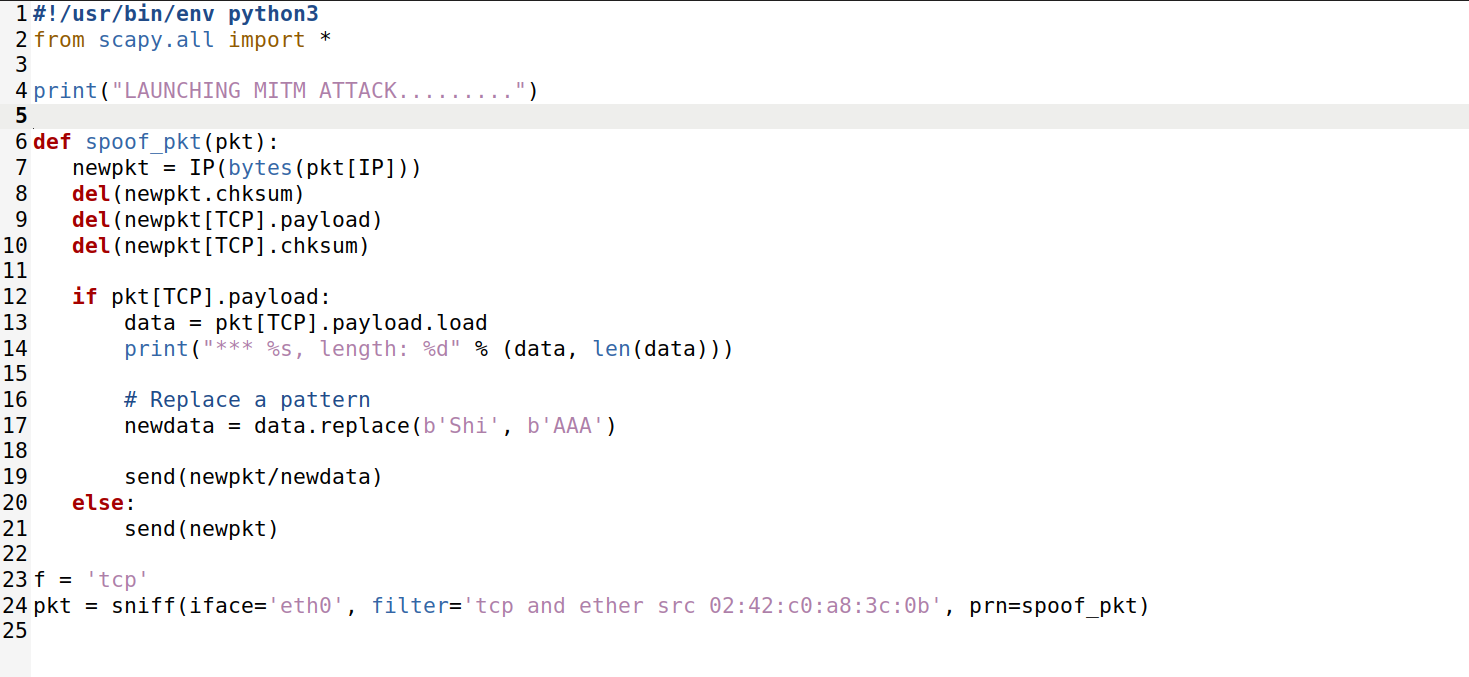


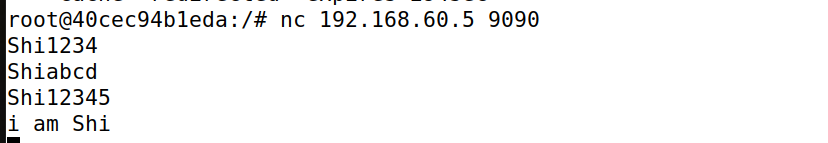
**Q4**

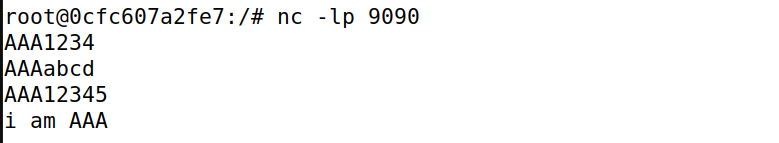
在此次攻击中，我只截取了从受害主机10.9.0.5发起，到目的端口192.168.60.5的流量因为重定向攻击的路由表是从受害主机发起，经由恶意路由的。

**Q5**

使用mac地址进行追踪





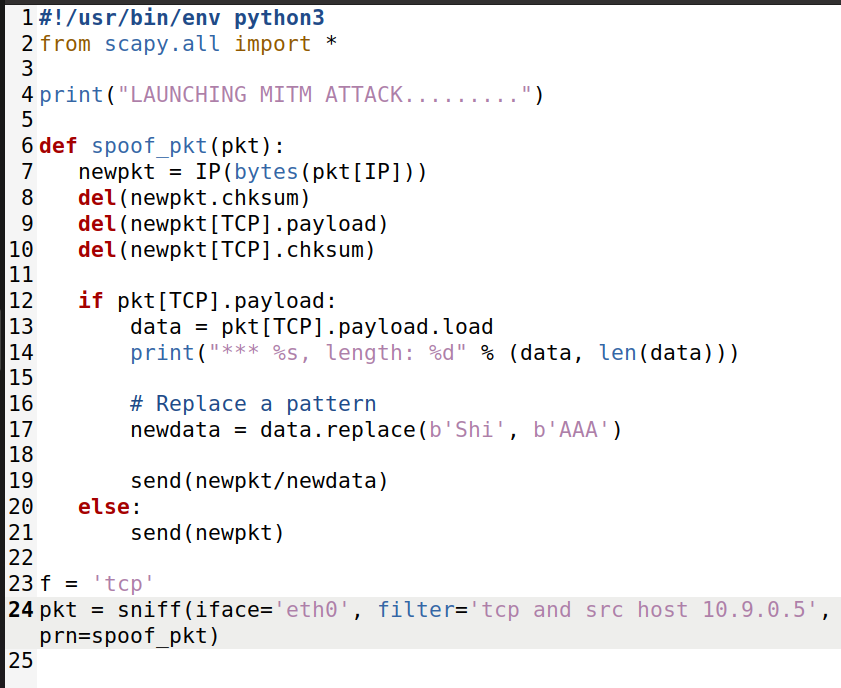


可以成功进行中间人攻击，但运行时间较慢，响应时间差不多有十几秒

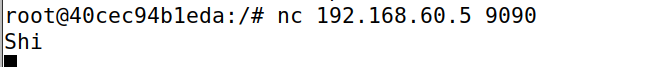


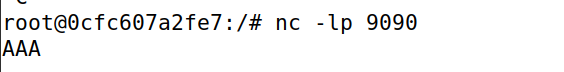
使用ip地址作为过滤标准时

tcp and ether src 02:42:c0:a8:3c:0b



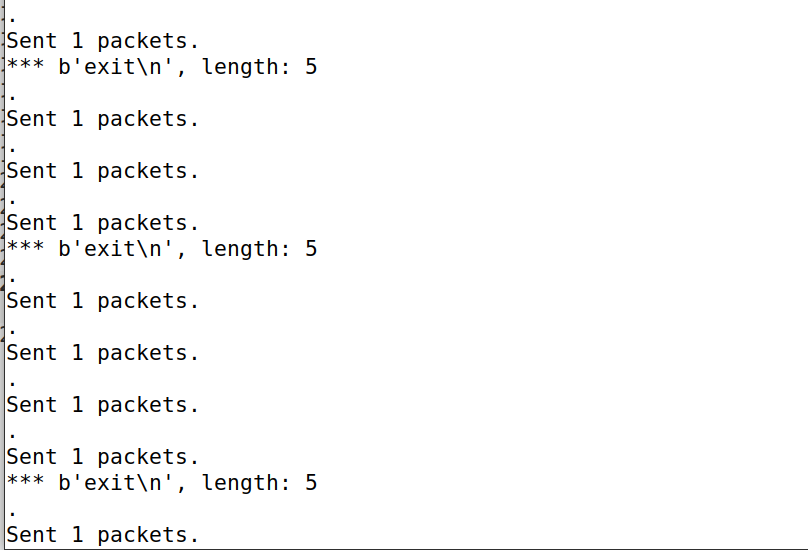
运行结果，发现可以成功实现中间人攻击





观看运行代码结果，发现出现多次发包的现象





由此，我们可以分析结果。使用ip地址发包较多，mac地址发包较少，因为mac地址和设备绑定，而同一个设备可以对应多个ip地址。