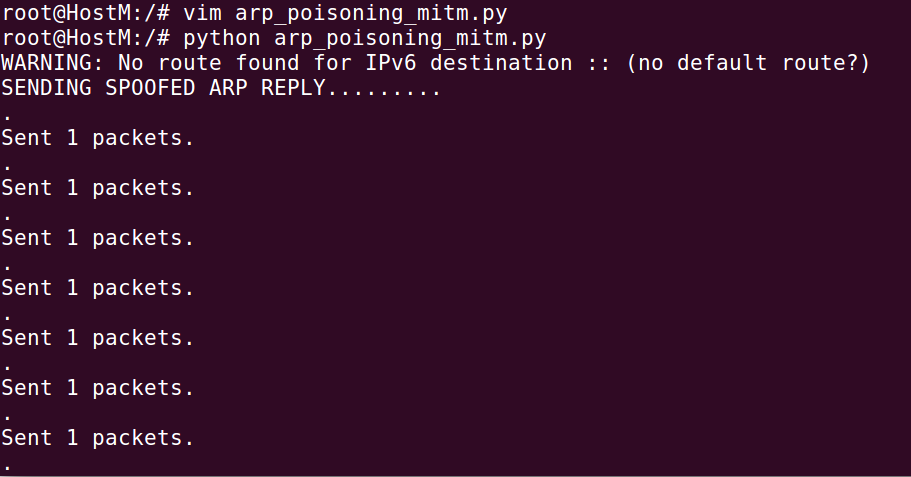
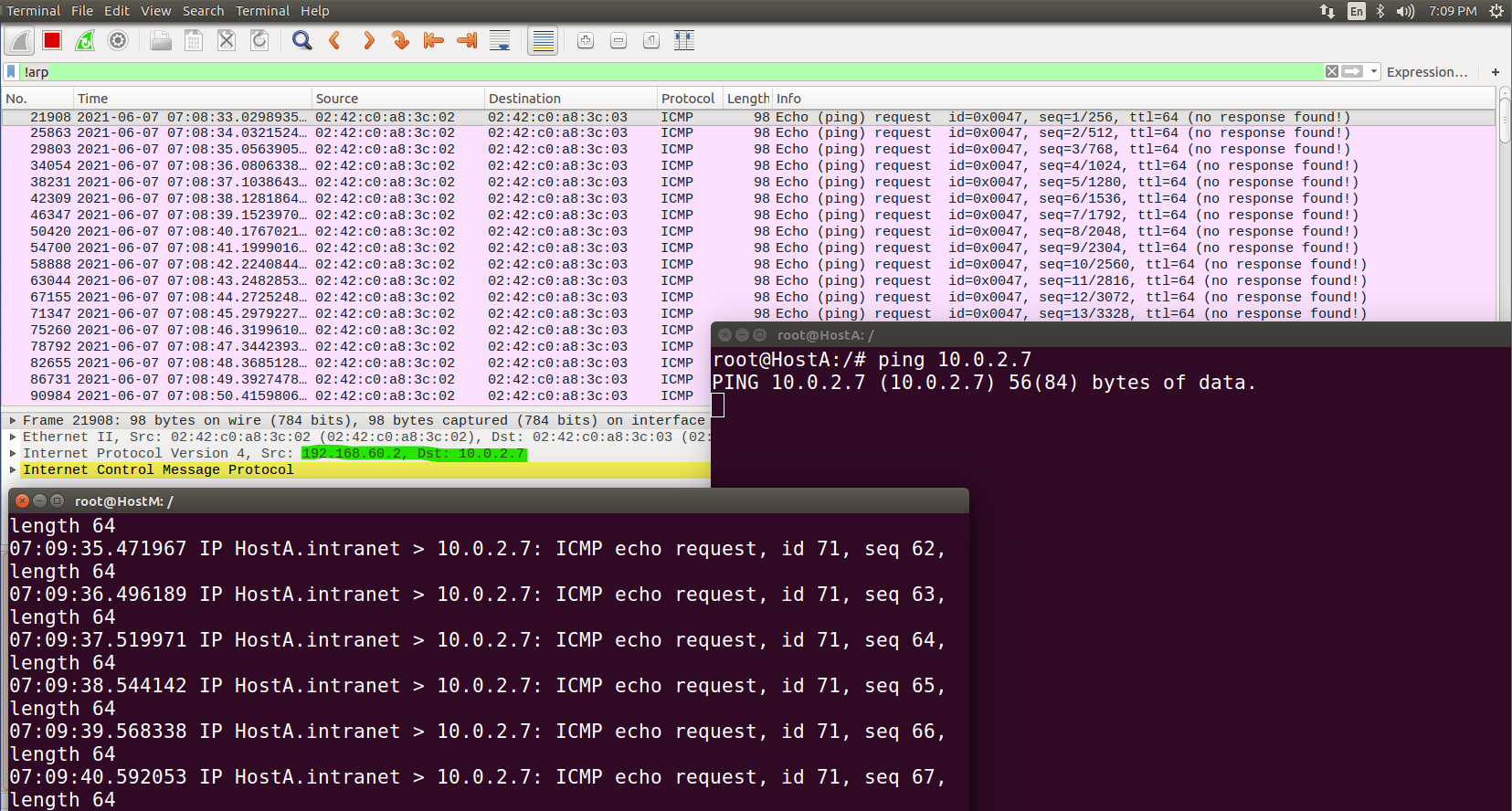
1. 完成脚本编写后，在HostM（攻击主机）上运行脚本



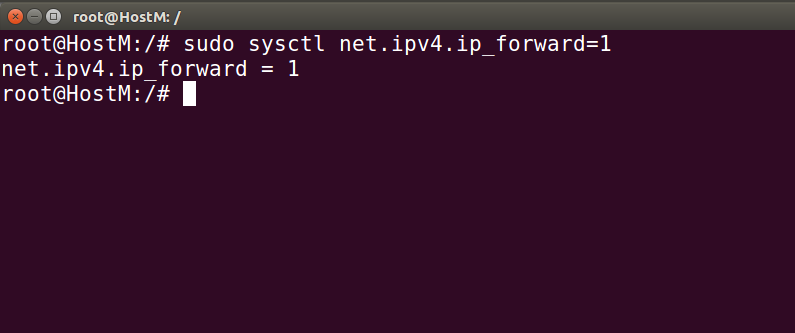
1. 执行脚本保持持续发送arp污染数据包（对象分别为HostA以及网关）

同时在HostA（被攻击主机）上执行对外网IP（10.0.2.7）的ping命令、在HostM（攻击主机）上打开tcpdump、在虚拟机（vm）上打开wireshark监听指定网卡

可以看到，tcpdump捕获了所有来自HostA的ICMP包，而wireshark中所有来自HostA的ICMP包信息部分全部为无回复，原因在于没有在HostM（攻击主机）上打开IP转发

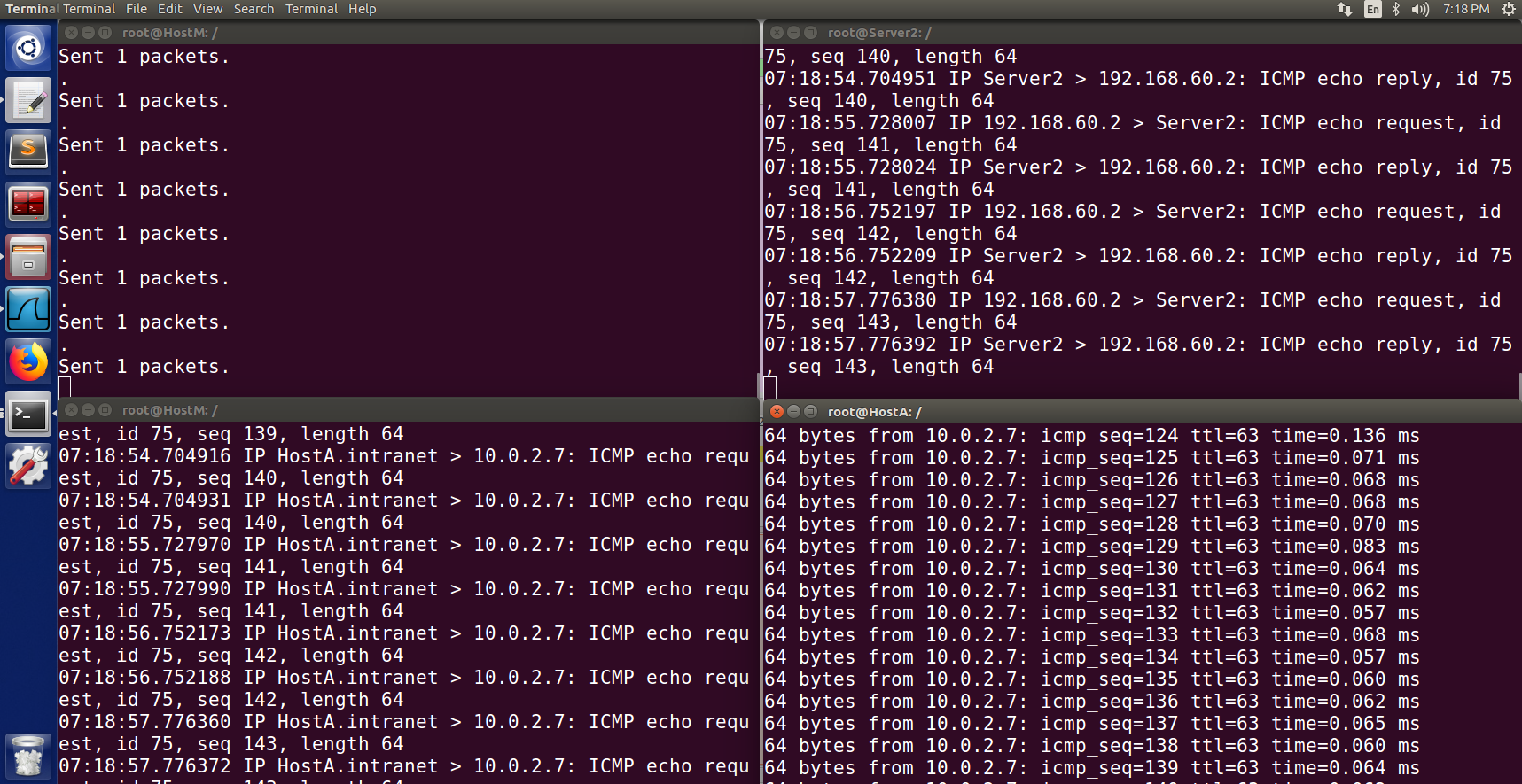


1. 在HostM（攻击主机）上打开IP转发



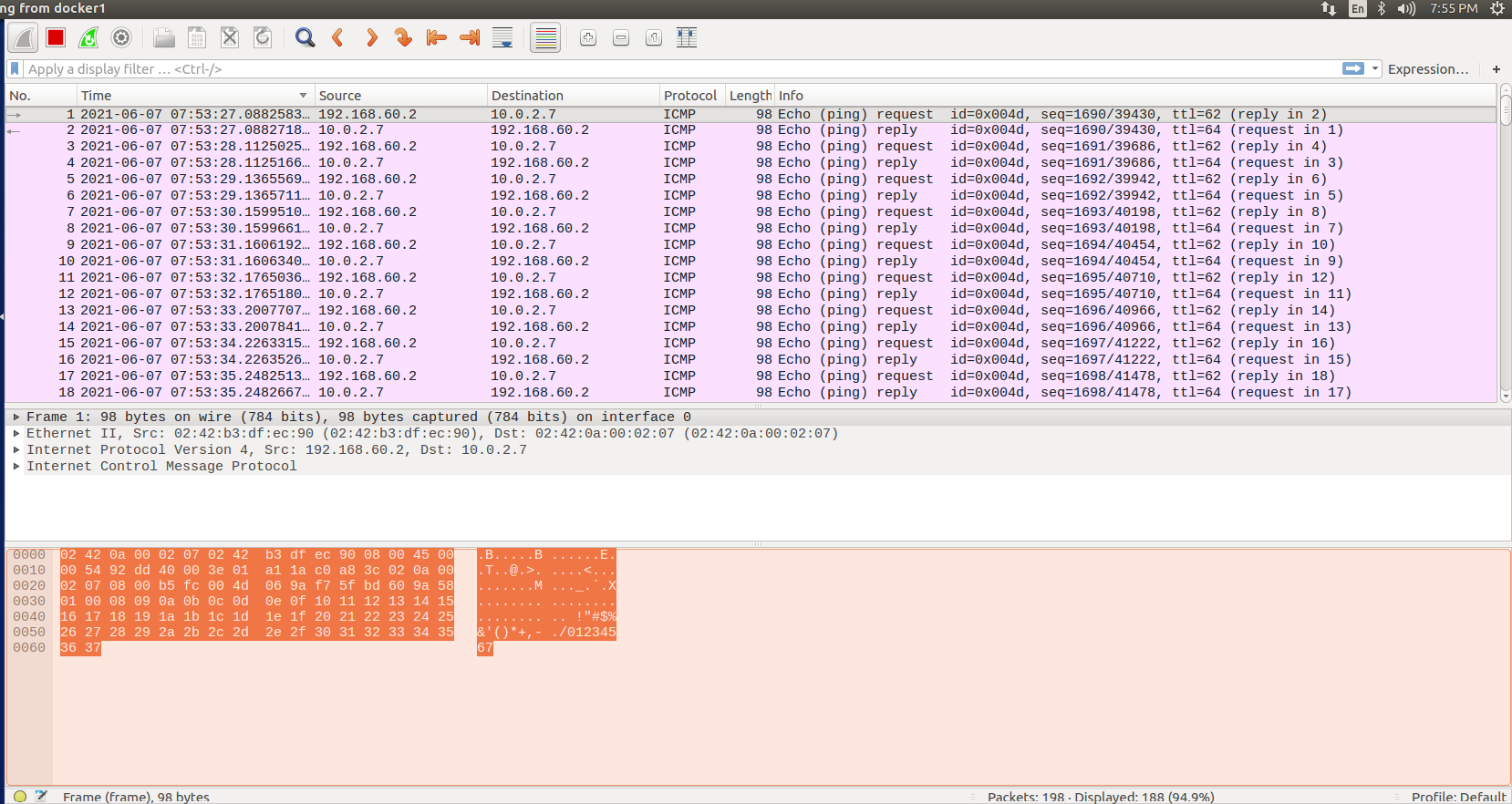
1. 在HostA（被攻击主机）上执行对外网IP（10.0.2.7）的ping命令、在HostM（攻击主机）上打开tcpdump、在Server2（外网IP）上打开tcpdump

可以看到在保持arp攻击的情况下，在HostM（攻击主机）上打开IP转发后，HostA（被攻击主机）开始收到来自Server2的应答包，而Server2（外网IP）的tcpdump中也成功收到了来自HostA的ICMP包。此时HostM（攻击主机）上的tcpdump捕获了所有HostA和Server2之间交互的数据包，完成了中间人的监听效果



1. 在虚拟机vm上打开wireshark，分别指定网卡docker1、docker2——分别对应10.0.2.0/24网段和192.168.60.0/24网段进行监听

docker1——10.0.2.0/24网段，可以看到捕获到的ICMP包中存在相对应的request和reply包，表明通信过程是完整的



docker2——192.168.60.0/24网段，可以看到捕获到的ICMP包中存在相对应的request和reply包，表明在HostM（攻击主机）打开IP转发情况下，HostA（被攻击主机）是能够正常进行数据包的发送和接收的

