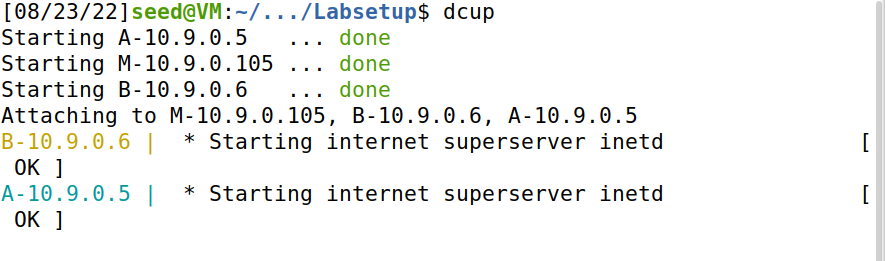
**Task 5 – ARP、ICMP**

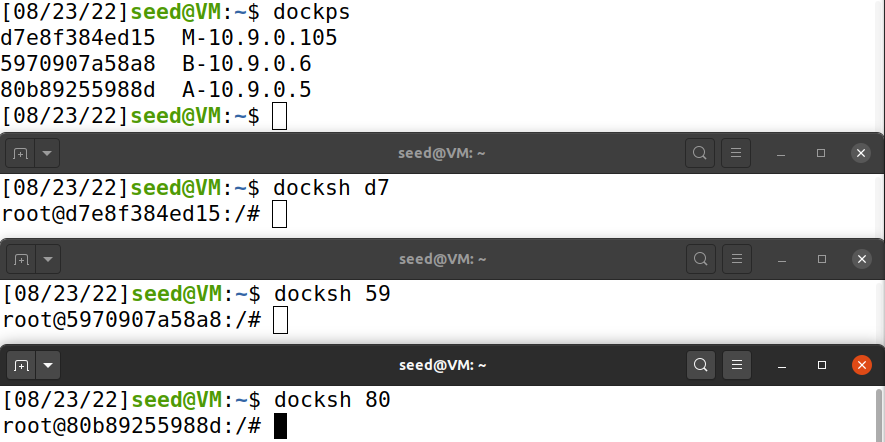
**ARP部分**

**环境配置**

进入实验目录/home/seed/Desktop/Labs\_20.04/Network Security/ARP Cache Poisoning Attack Lab/Labsetup，输入dcup开启容器：



另开多个终端，输入doksh xx开启相应容器：



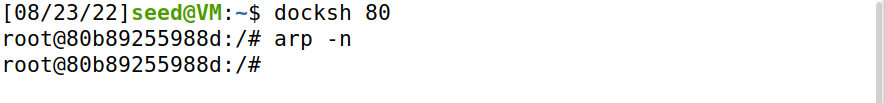
1. **ARP Cache Poisoning**

**准备工作：**

本实验的任务是要攻击A机器的ARP cache，在A的ARP缓存中，将B的IP地址对应到M的MAC地址上。需要进行以下实验前的准备：

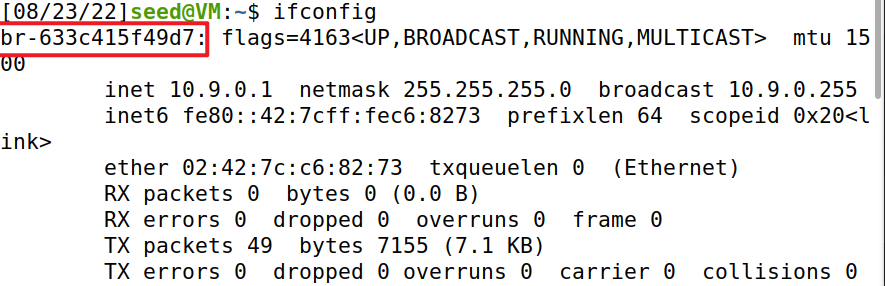
**查看主机A的ARP缓存：**

在主机A中输入arp -n查看ARP缓存，当前为空：



**查看虚拟机为容器创建的网卡信息：**

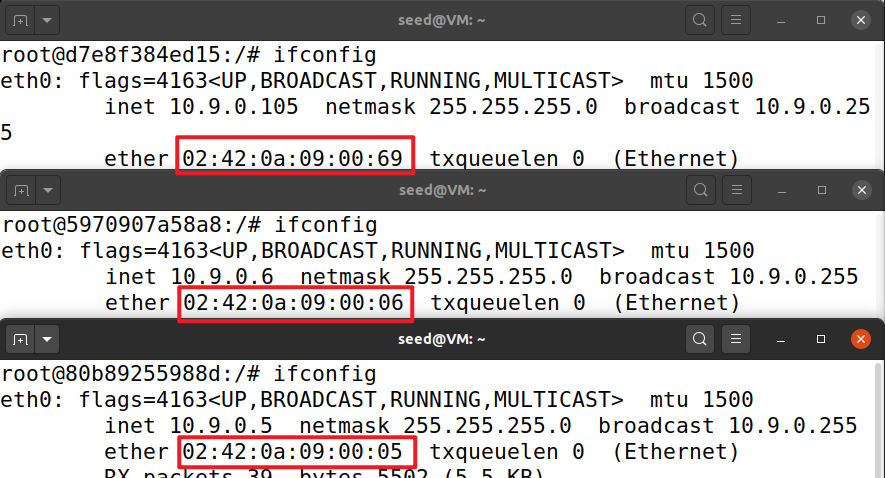
在虚拟机上执行ifconfig命令查看网卡信息：



可以看到对应子网10.9.0.0/24的网卡是br-633c415f49d7。

**查看容器MAC地址：**

在各容器中分别执行ifconfig命令查看MAC地址：

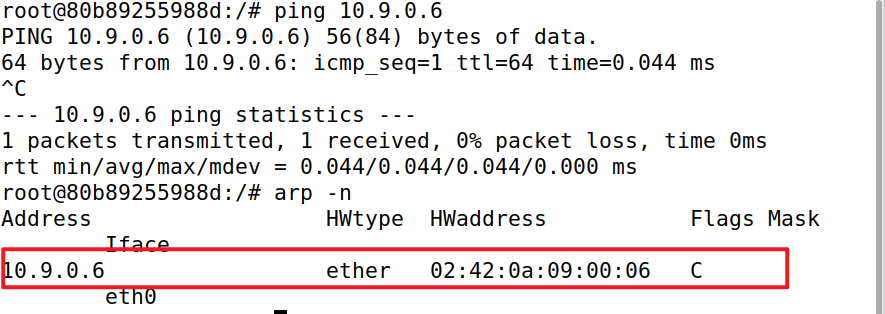


从上到下依次为主机M、B、A的MAC地址信息。

**攻击过程：**

* **1A 使用ARP Request**

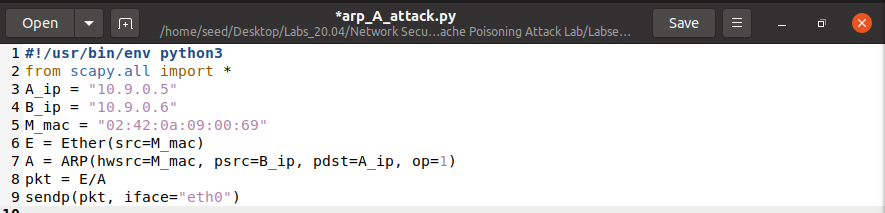
主机A执行ping 10.9.0.6，连接主机B，再次查看A的ARP缓存：



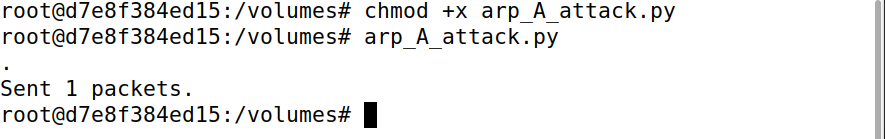
新增了一条缓存记录，与B的各项信息相匹配。

在主机M的/volumes路径下新建文件arp\_A\_attack.py，内容如下：

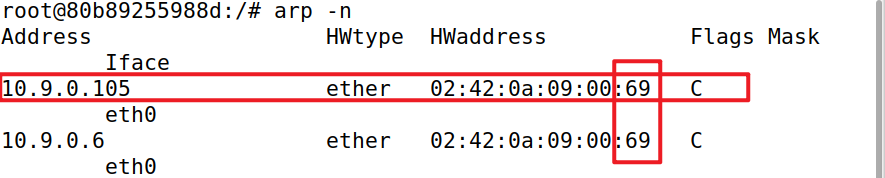




执行chmod +x arp\_A\_attack.py为文件添加执行权限后，运行程序：



在A中再次查看ARP缓存：

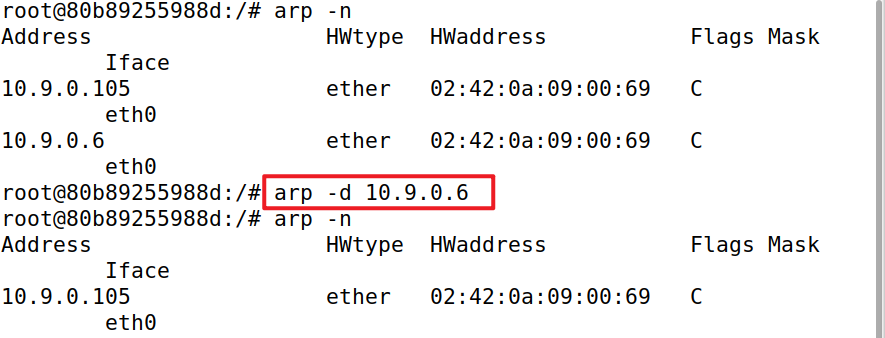


可以看到与前面的结果相比，新增了一条M的记录，同时发现B的MAC地址的也从…06变为了…69，是M的MAC地址。

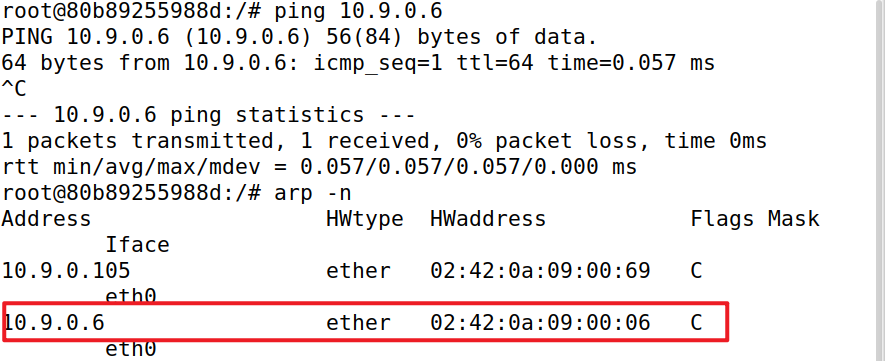
写入成功。

* **1B 使用ARP Reply**

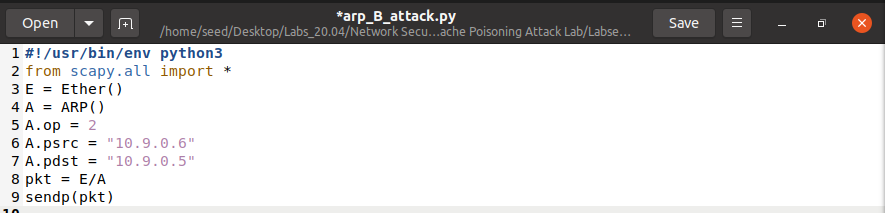
B的记录已经存在在A的ARP缓存中，执行arp -d 10.9.0.6命令清除该记录：



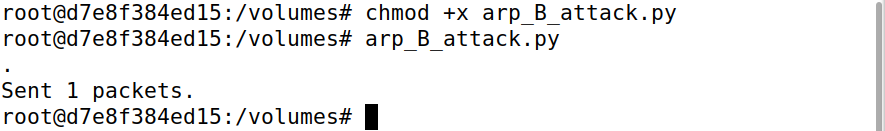
主机A ping 主机B后，A的ARP缓存中再次出现信息正确的关于B的缓存：



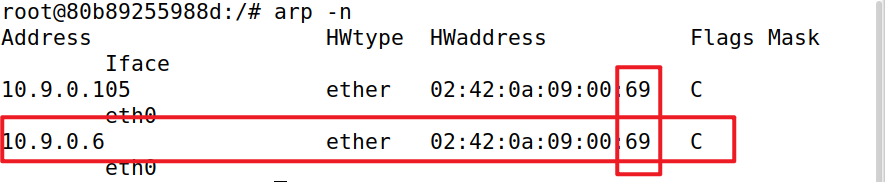
在主机M上创建arp\_B\_attack.py文件，内容如下：



添加执行权限后，运行该文件：

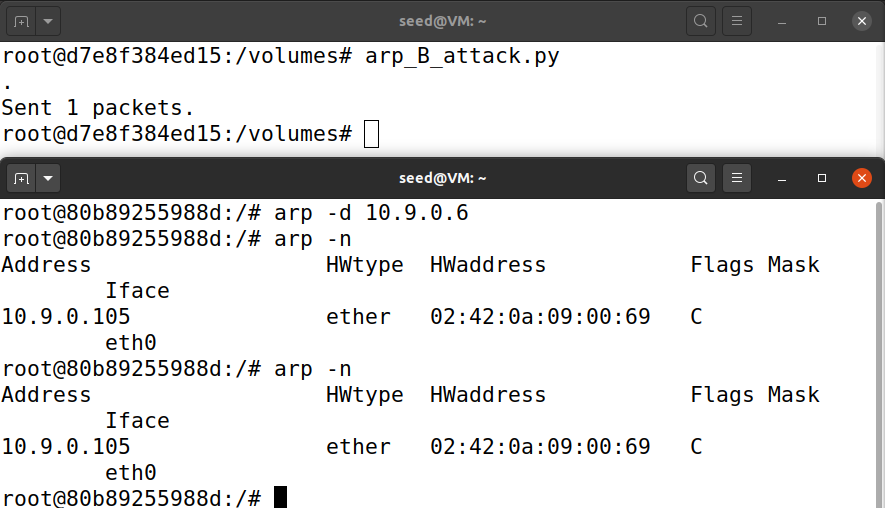


在主机A上再次查看ARP缓存：



写入成功。

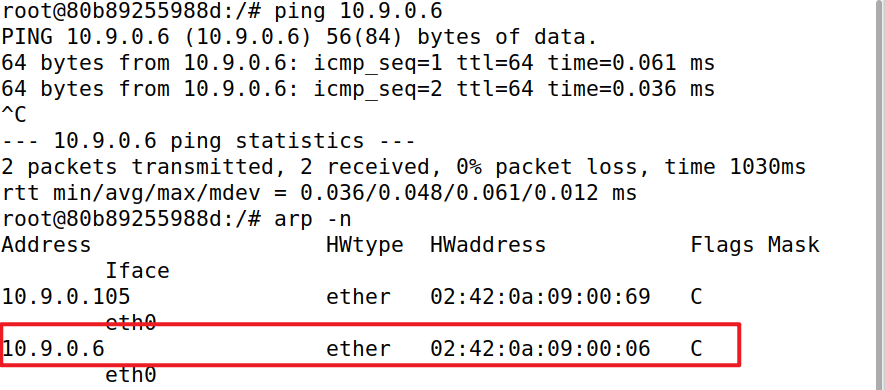
将A中关于B的ARP缓存清除后，在主机M上运行程序，发现ARP缓存没有增加新的条目：



由以上实验可以说明，ARP Reply只能修改现有的ARP缓存条目，并不能写入新的内容。

* **1C ARP Gratuitous mess**

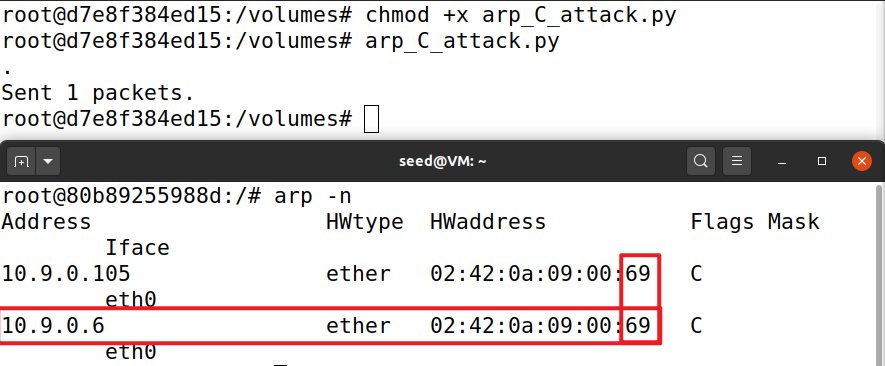
清除A中的关于B的ARP缓存，通过ping命令写入正确的ARP缓存：



在主机M中新建arp\_C\_attack.py文件，内容如下：

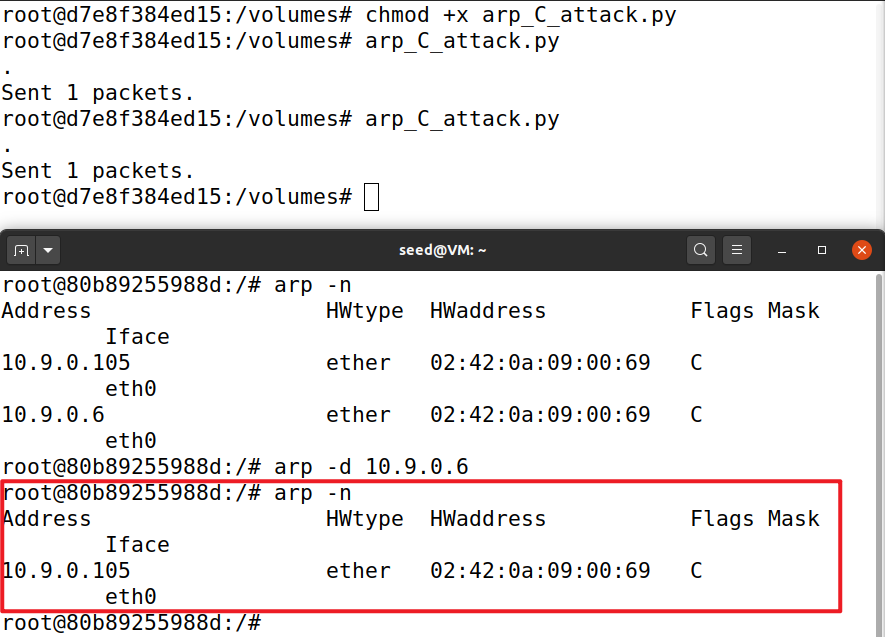


添加执行权限后运行，查看A的ARP缓存：



写入成功。

清除该条记录后，再次运行arp\_C\_atack.py，查看A的ARP缓存，无新增项：



所以这一方法也是只能够修改而不能写入新的项目。

1. **MITM Attack on Telnet using ARP Cache Poisoning**

* **2A ARP Poisoning**

在主机M上创建文件mitm\_attack.py，内容如下：

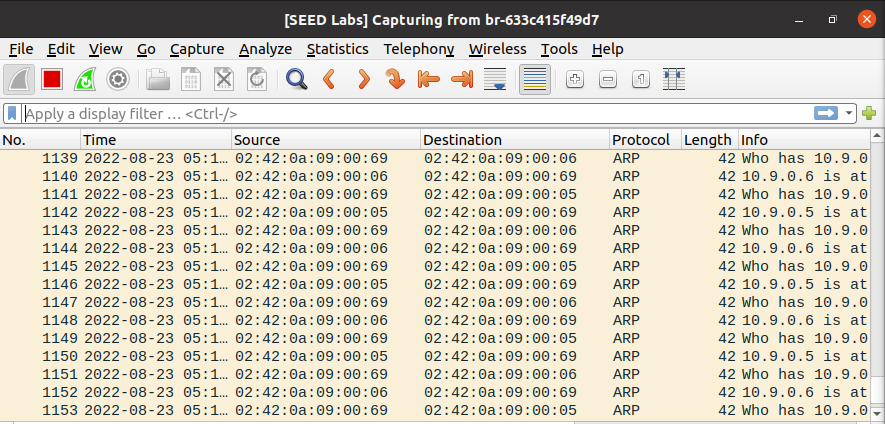


添加执行权限后运行程序，即可实现ARP投毒攻击：

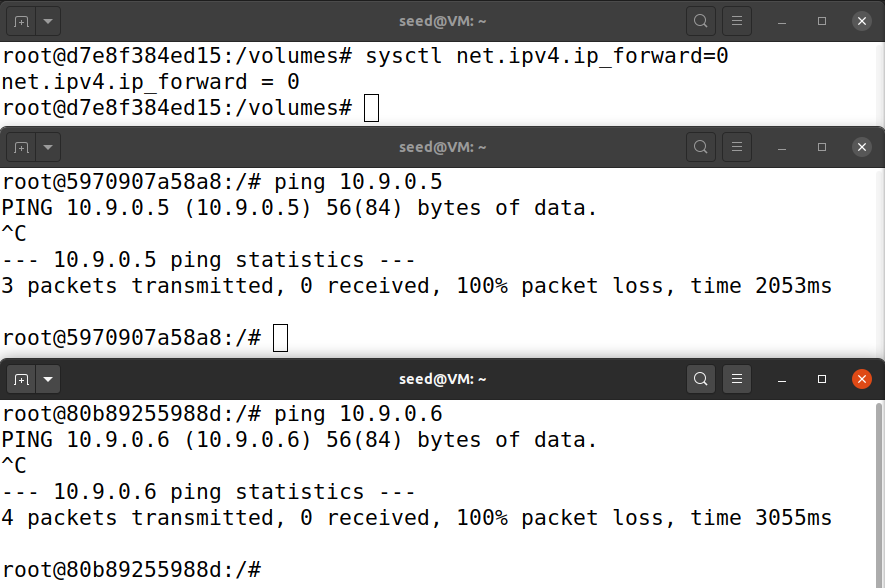


* **2B Testing**

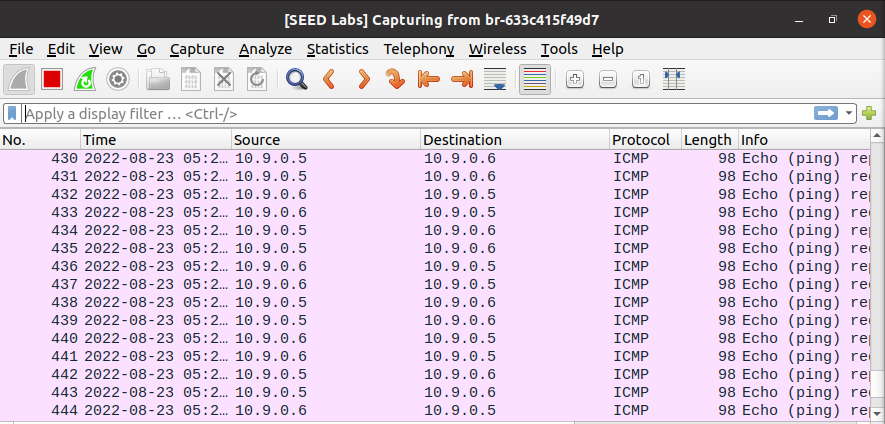
开启wireshark，监听网卡br-633c415f49d7：

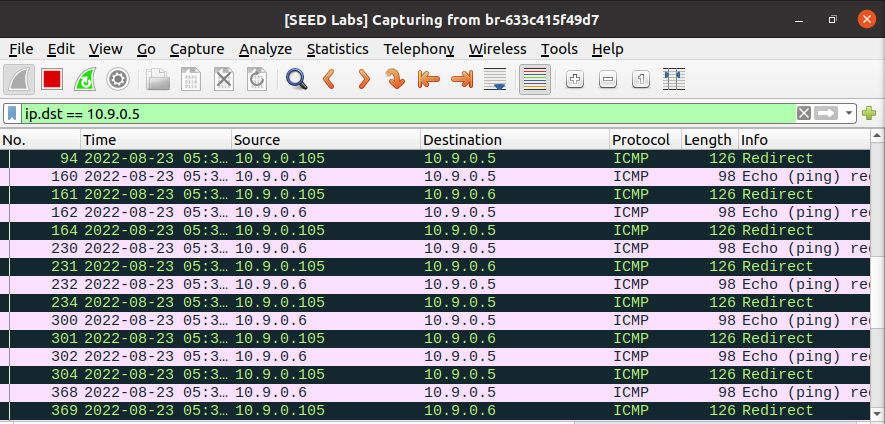


在M上执行sysctl net.ipv4.ip\_forward=0的命令，再运行mitm\_attack.py，此时主机A、B之间都是无法ping通的：



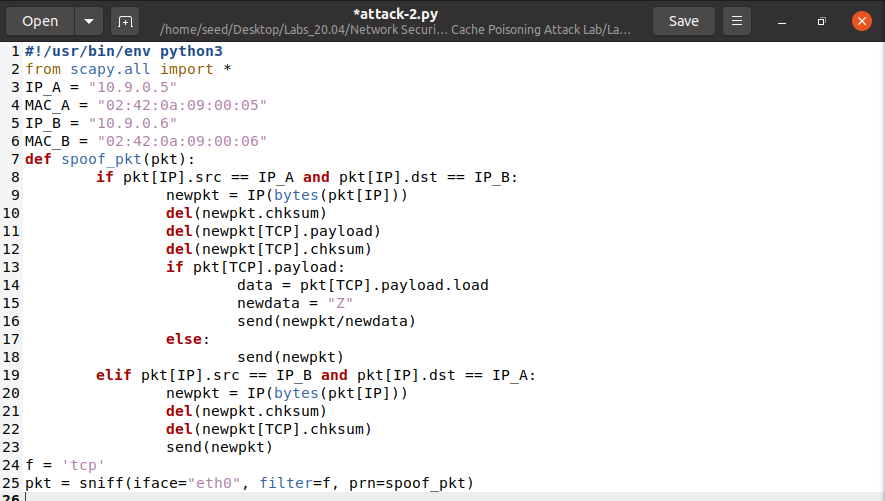
在wireshark中看到，主机A和B的ping只有请求而没有应答：



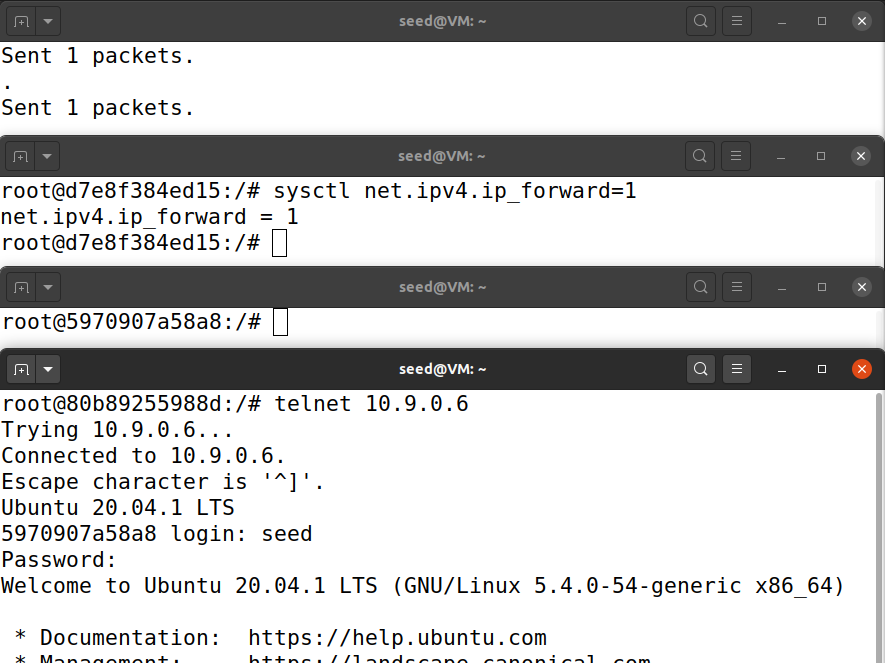
在主机M上执行sysctl net.ipv4.ip\_forwarding = 1将net.ipv4.ip\_forwarding设置为1，运行mitm\_attack.py，主机B ping 主机A时，可以再wireshark中看到有了应答：

* **2C Sniff and spoof**

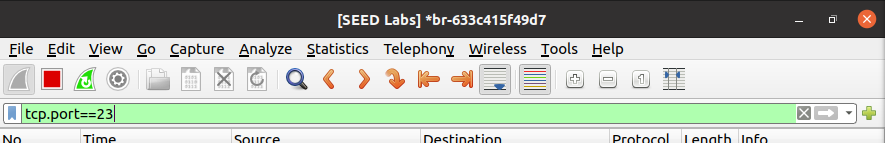
在主机M上创建attack-2.py文件，内容如下：



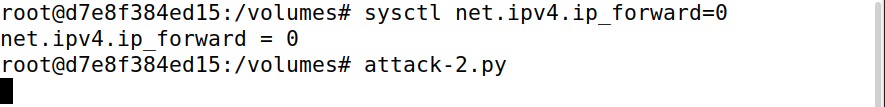
在主机M中执行mitm­\_attack.py程序，另外开启一个页面用于后续攻击，执行sysctl net.ipv4.ip\_forward=1命令开启转发，让主机A、B可建立正常telnet连接：



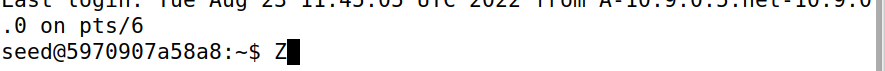
开启wireshark，监听网卡br-633c415f49d7，设定过滤规则tcp.port==23进行筛选：



在主机M上执行sysctl net.ipv4.ip\_forward=0禁止转发，运行attack-2.py：



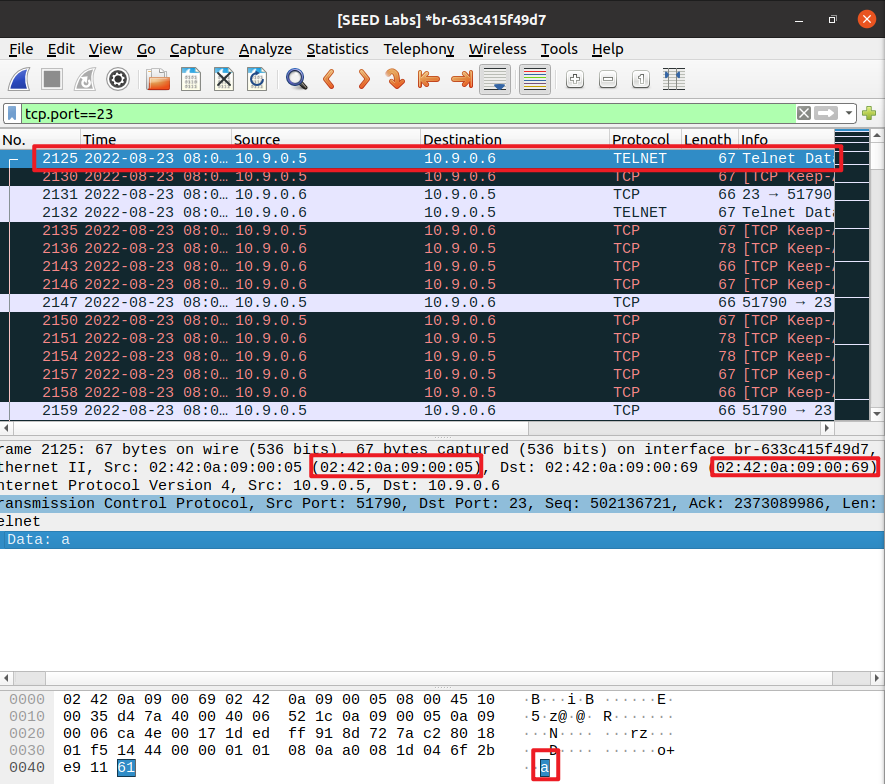
在A主机上任意输入一个字符到建立好的telnet连接中：



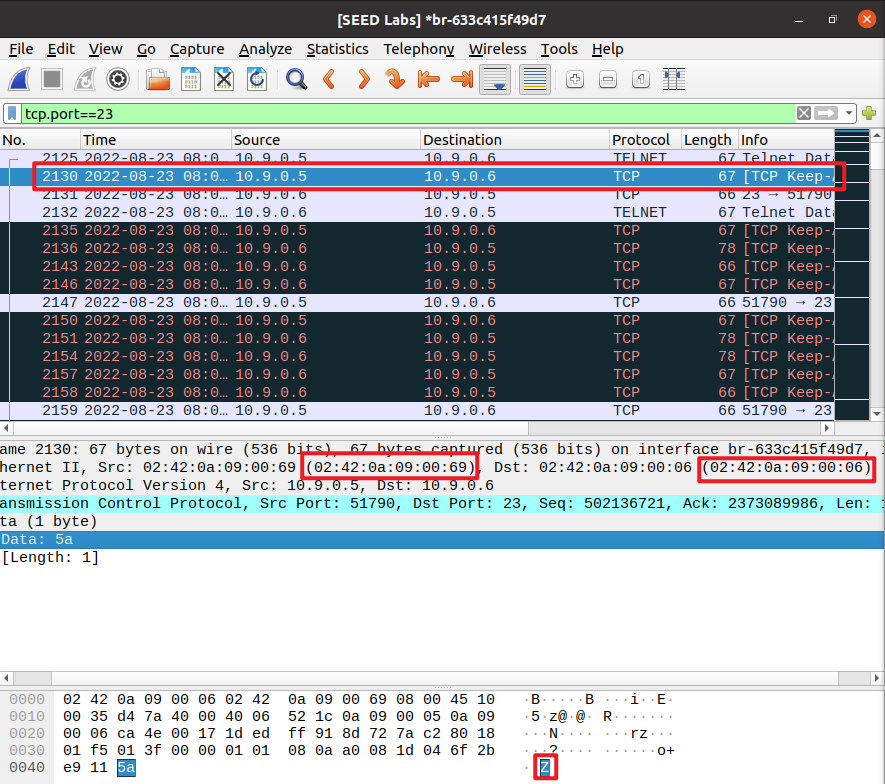
发现无论输入什么都会变成‘Z’，

在wireshark中分析抓到的包：

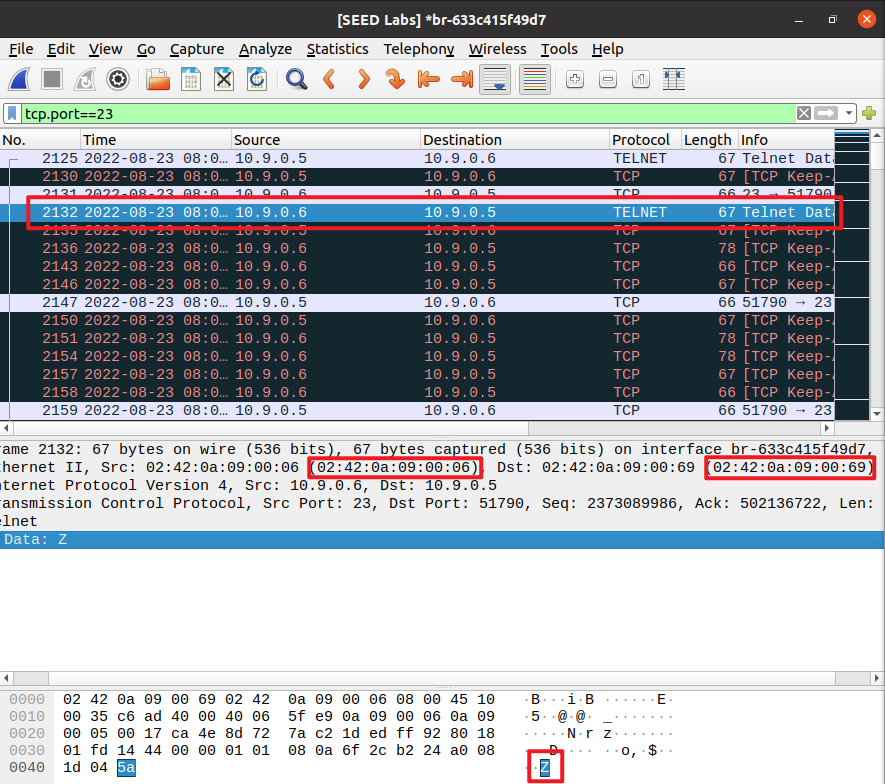
第一个包是A->M（10.9.0.5->10.9.0.6），可以看到最后面的字符是原本输入的‘a’：



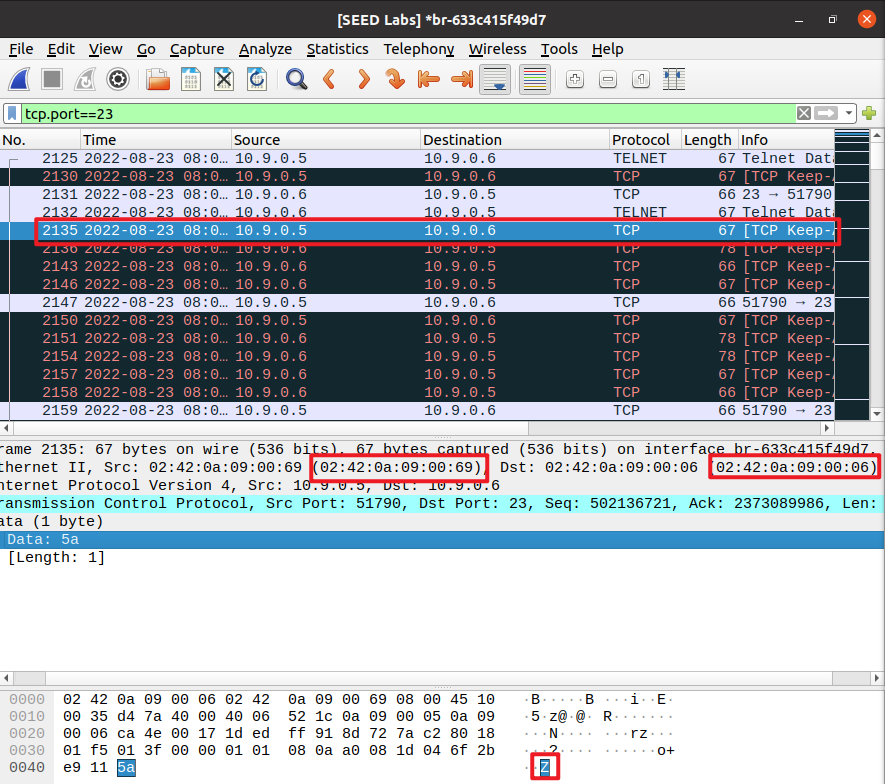
第二个包是M->B（10.9.0.5->10.9.0.6），可以看到最后面的字符变成了修改过后的‘Z’：



第三个包是B->M（10.9.0.6->10.9.0.5），可以看到B返回的是‘Z’：

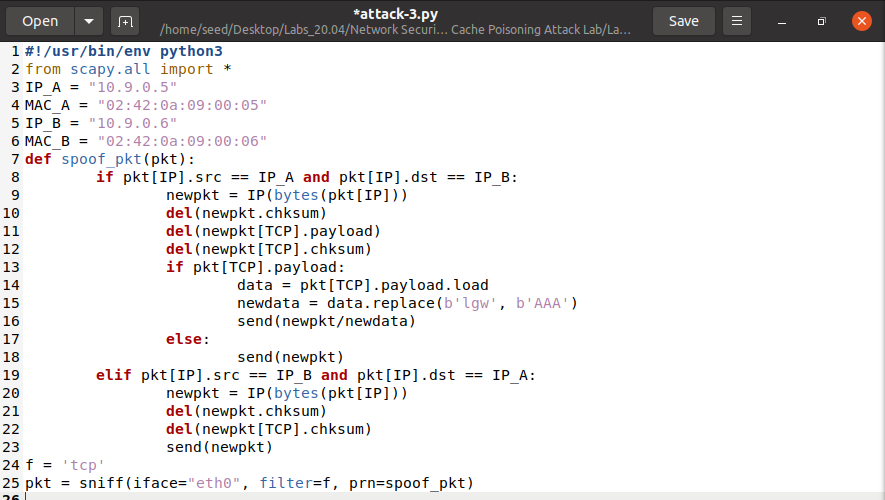


第四个包是M->A（10.9.0.6->10.9.0.5）：



1. **MITM Attack on Netcat using ARP Cache Poisoning**

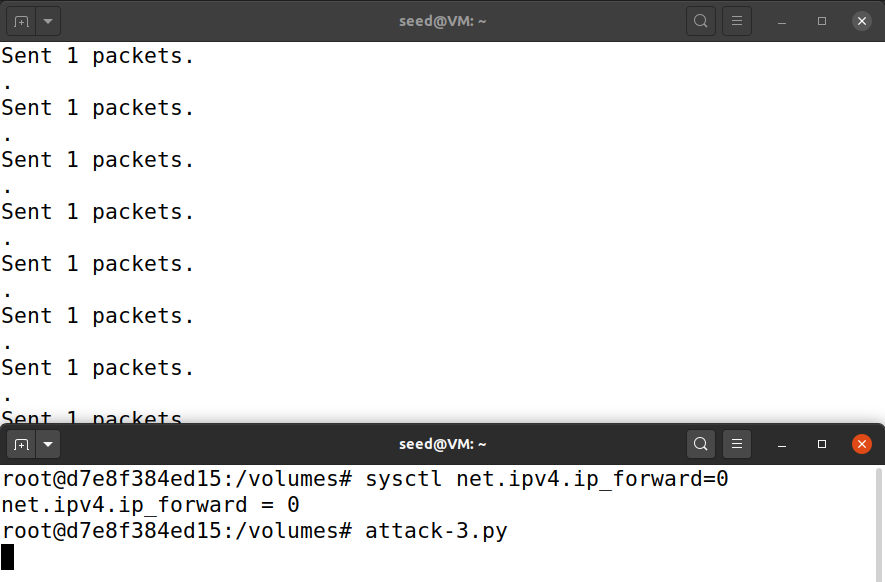
在M上创建attack-3.py，内容如下：



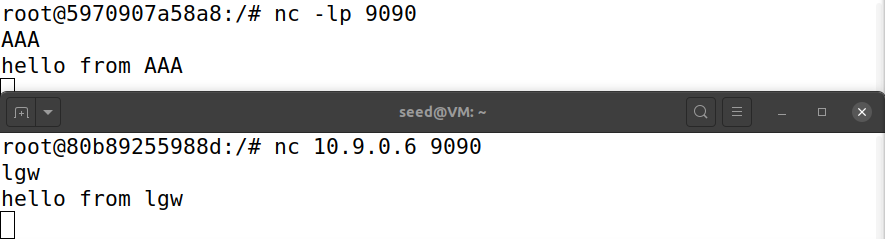
执行sysctl net.ipv4.ip\_forward=0关闭转发：



执行mitm\_attack.py进行投毒攻击，另一个页面上执行attack-3.py程序，等待来自主机A、B的数据：



主机B上执行nc -lp 9090开启netcat监听，在主机A上输入nc 10.9.0.6 9090，连接到主机B，输入含有“lgw”的字符串，在B中可以收到相对应的信息，但字符串中的“lgw”被替换为了“AAA”，与attack-3.py中的内容相符：

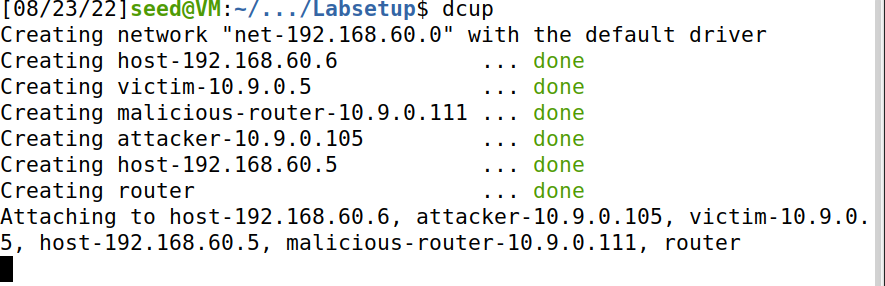


攻击成功。

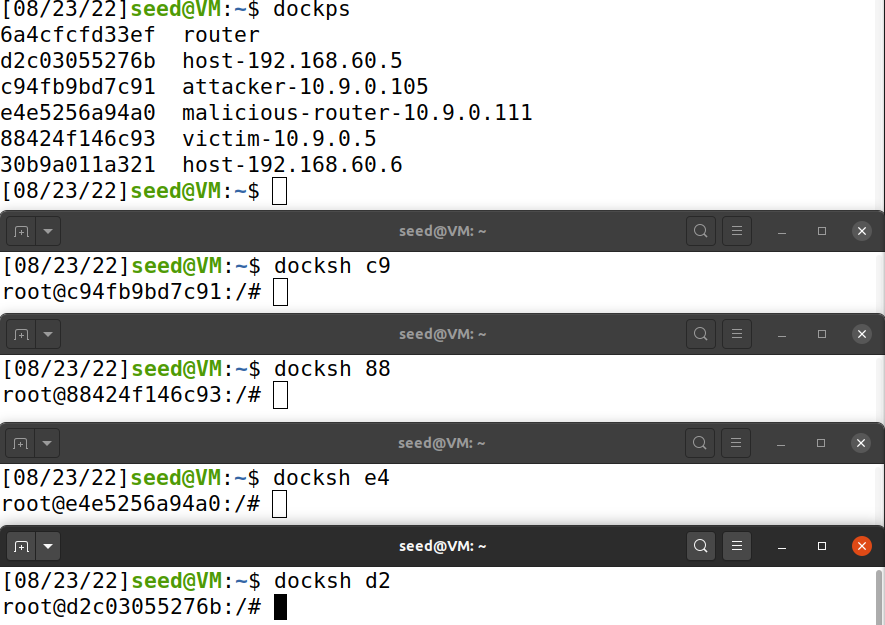
**ICMP部分**

**环境配置**

进入实验目录/home/seed/Desktop/Labs\_20.04/Network Security/ICMP Redirect Attack Lab/Labsetup，输入dcup开启容器：



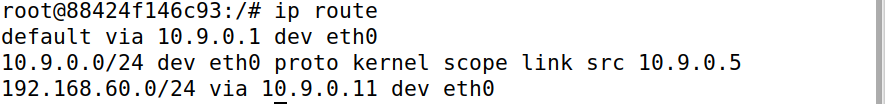
另开多个终端，输入doksh xx开启相应容器：



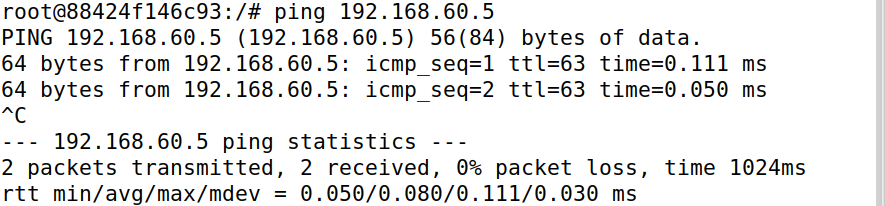
四个容器从上到下分别是attacker、victim、router、host-192.168.60.5。

1. **ICMP attack**

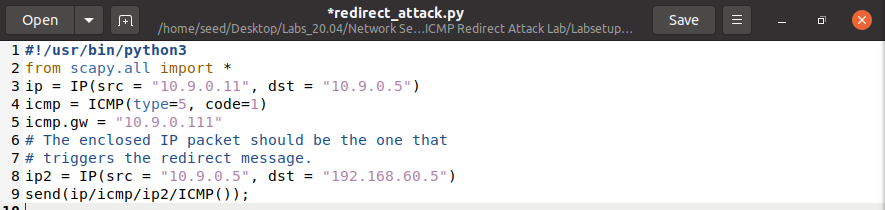
在victim主机上输入ip route查看路由表，如下：



Ping 192.168.60.5：



在attacker主机上创建redirect\_attack.py文件，内容如下：



（修改实例代码中的第一个ip项，ip地址是字符串，需要加引号）

第一个IP报文为重定向报文，attacker（10.9.0.105）伪装成原本正确的路由10.9.0.11发送给Victim（10.9.0.5）；

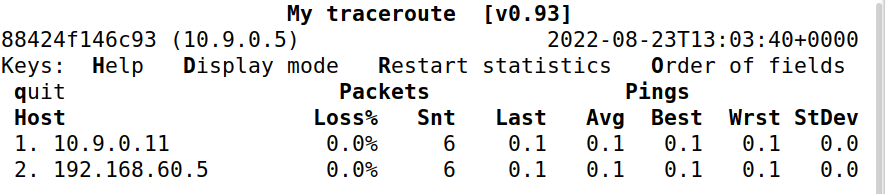
ICMP的报文内容表示将报文要重定向到10.9.0.111这个Router上；

ip2报文是捕获的IP报文，是Victim（10.9.0.5）发送ICMP报文给192.168.60.0/24子网的报文。

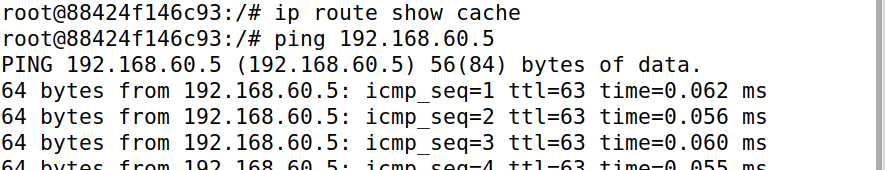
所以，注意攻击要和Victim主机ping 192.168.0.5同时进行，这样才能发送回复的重定向报文。

查看traceroute，这里的结果要和攻击后的结果进行对比，所以先记录下来。

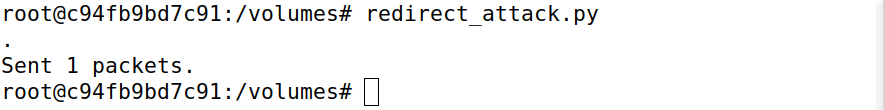
在Victim主机执行mtr -n 192.168.60.5命令，这是攻击之前的结果：



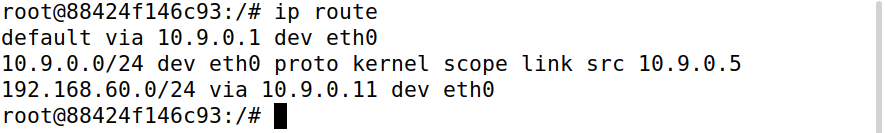
在Victim主机上输入ip route show cache，查看ip route cache，确保没有记录，然后ping 192.168.60/24子网：



然后attacker开始执行程序进行攻击：

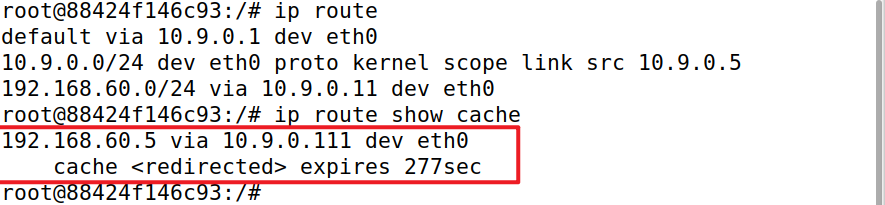


等待Attacker主机发送成功1秒后，Victim主机已经接收到重定向的数据包后，停止ping 192.168.60.5。然后输入查看ip route，结果如下：

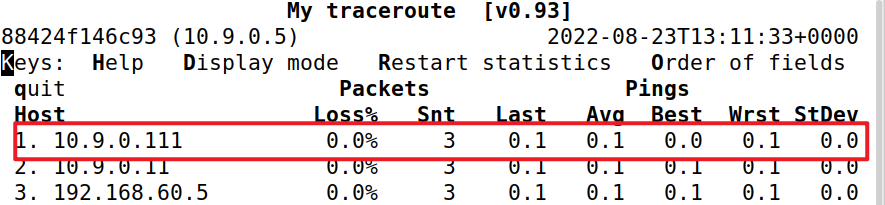


发现路由表没有变化，接入192.168.60.0/24子网还是通过10.9.0.11，而不是10.9.0.111。

再查看ip route cache：



在Victim主机执行mtr -n 192.168.60.5 命令，这是攻击之后的结果：



Traceroute中多了一项内容，10.9.0.111，说明攻击成功。

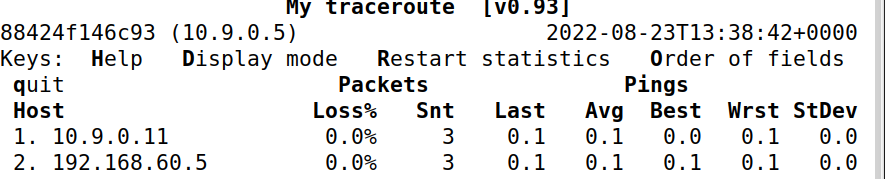
**问题1：**可否将重定向的IP地址也就是icmp.gw修改为外网地址？(修改后重新攻击查看traceroute结果)

开启host-192.168.60.6容器，修改程序中的icmp.gw为”192.168.60.6”：



在victim主机中输入ip route flush cache清除缓存，重新执行攻击，结束后查看ip route cache为空，且traceroute中也没有相应记录：

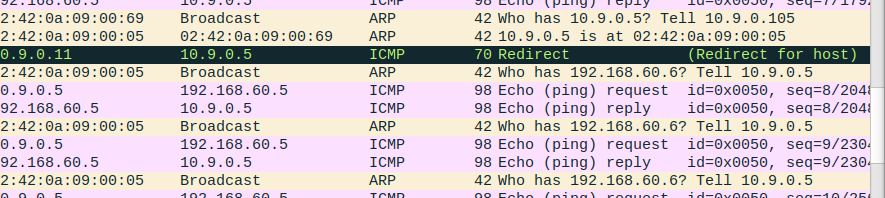




所以是不可以重定向到外网地址的。

**原因分析：**分析原因是该外网地址在路由表中不存在，并不知道应该交付到哪里，ARP广播请求也没有其他主机能够回应，重定向失败，按照原路径发送。

在wireshark中抓包查看这一过程，与上述分析相同：



即便在victim主机中事先ping通192.168.60.6也不可以。

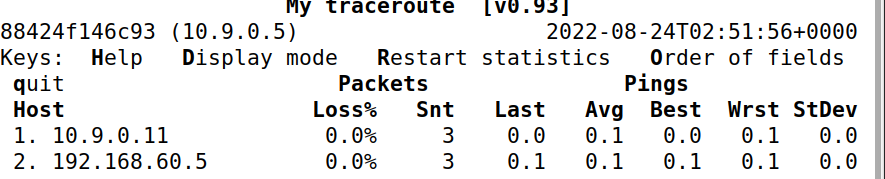
**问题2：**可否将重定向的IP地址也就是icmp.gw修改为不存在的地址？(修改后重新攻击查看traceroute结果)

修改icmp.gw为”10.9.0.123”（不存在的地址），再次进行操作：



Victim主机中ip route cache为空，traceroute中也没有新增：

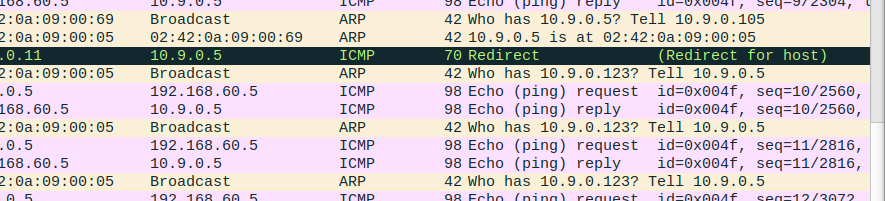




所以是不可以重定向到不存在的地址的。

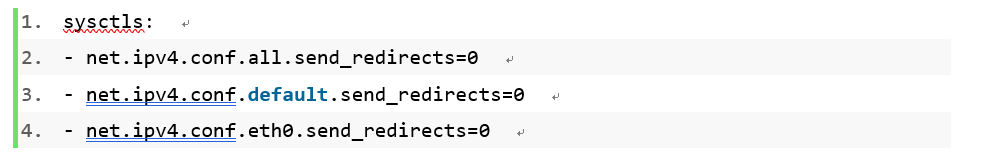
**原因分析：**victim主机收到了重定向报文，是相同网段的地址，但ARP缓存中并没有10.9.0.5，就需要发送ARP广播请求10.9.0.5的MAC地址。因为这个地址确实不存在，所以没有回应，其他主机也没有它的记录，那么victim重定向失败，只能按照原来的路径发送。

在wireshark中抓包查看这一过程，与上述分析相同：



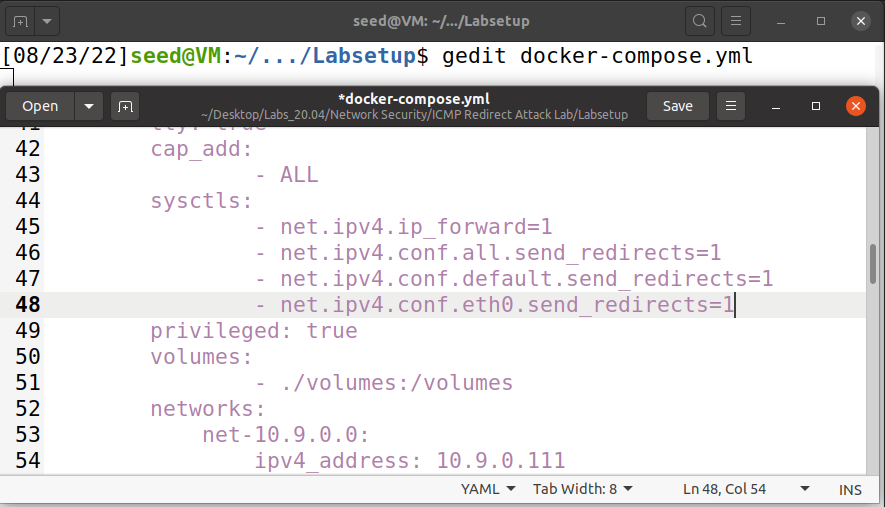
在收到重定向报文后就开始寻找10.9.0.123但是始终没有回应。

**问题3：**解释下面参数的含义，然后把它们修改为1，再进行攻击(需要重启容器)，结果如何？



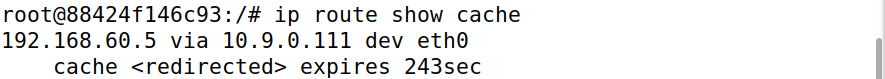
这几个参数含义是是否允许重定向，修改为1是可以进行重定向，能够充当路由器。

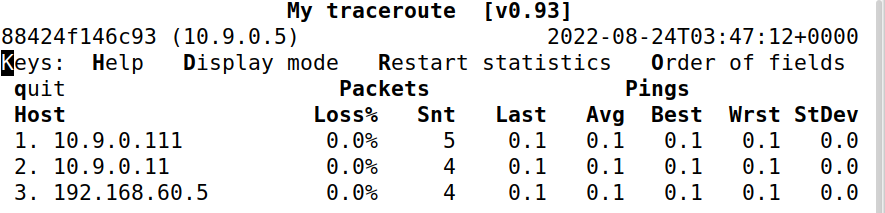
修改docker-compose.yml中对应的值为1；



重启这几个容器后再次进行攻击：

再次查看victim主机的ip route cache以及traceroute，发现重定向的10.9.0.111已经写入了ip route cache：



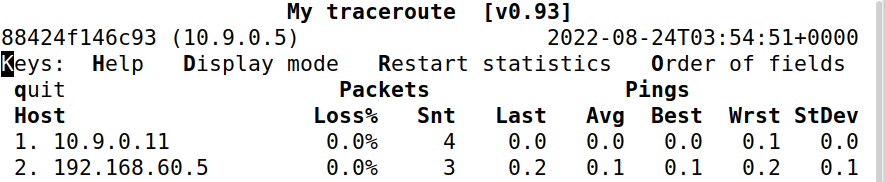


1. **Launching the MITM Attack**

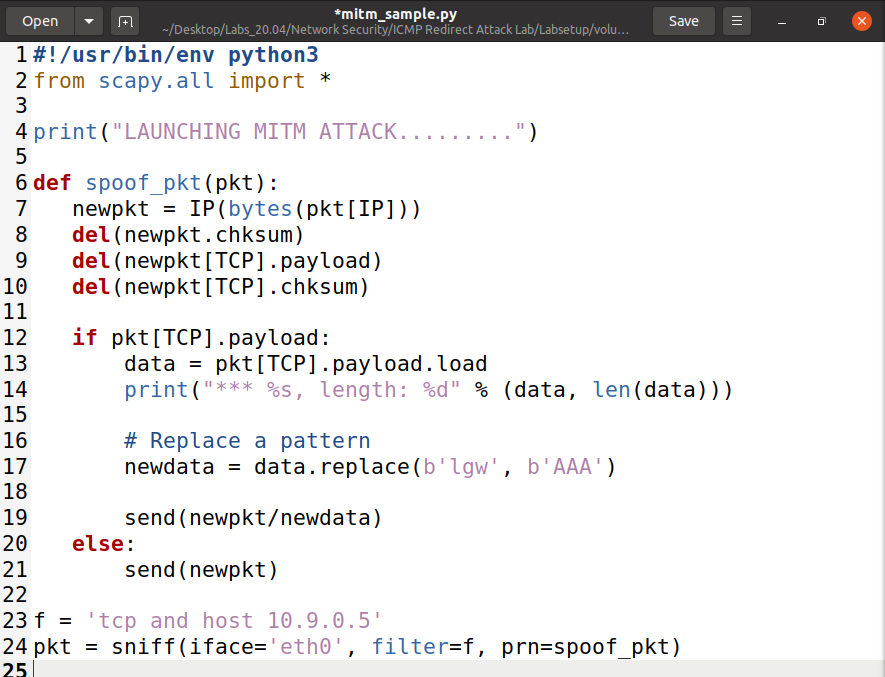
在192.168.60.5主机上输入nc -lp 9090开启netcat监听服务



在victim主机上查看traceroute：



在恶意路由器(10.9.0.111)上编写数据包篡改程序mitm\_sample.py，内容如下，是将‘lgw’替换为‘AAA’：

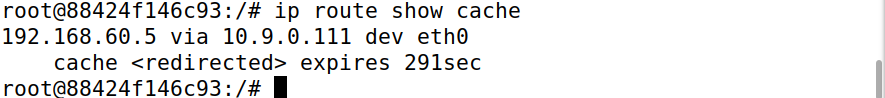


执行chmod +x mitm\_sample.py添加执行权限。

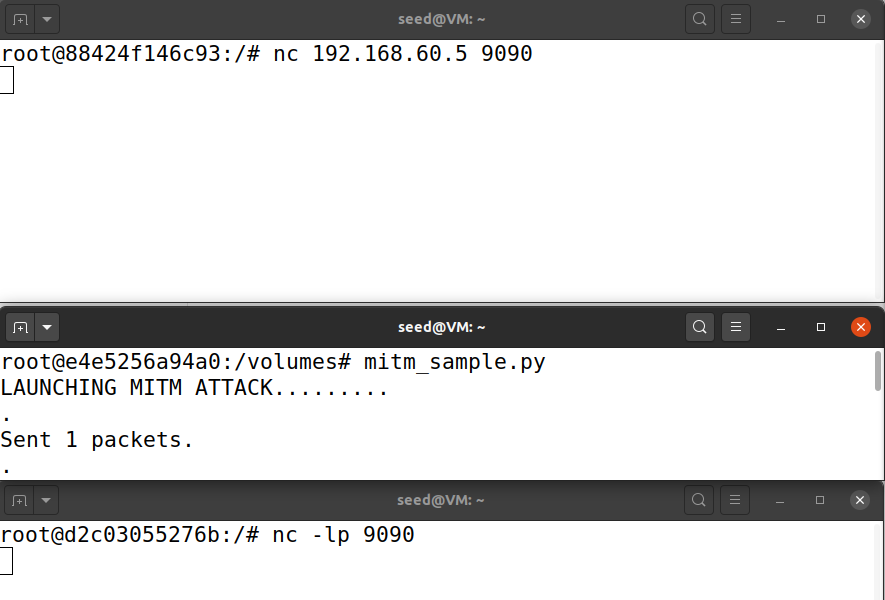
在恶意路由器上执行sysctl net.ipv4.ip\_forward=0关闭转发：



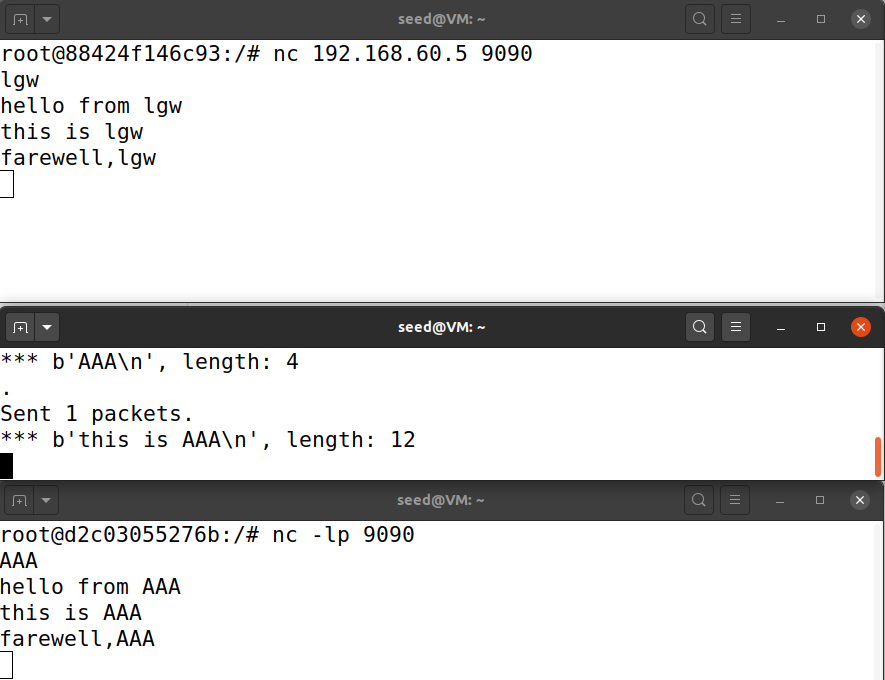
在attacker和victim中重复上个实验的攻击步骤，将10.9.0.111写入victim的ip route cache：



在恶意路由器上执行mitm\_sample.py程序发起攻击，在victim主机上执行nc 192.168.60.5 9090与192.168.60.5建立连接，可以看到恶意路由器已经开始发送数据包来进行欺骗了：



在victim主机上输入发送含有“lgw的”字符串，可以在恶意路由器中看到不断有数据包发出，而在192.168.60.5主机上收到的内容则显示为原本的“lgw”被替换为了“AAA”，如下：



攻击成功。