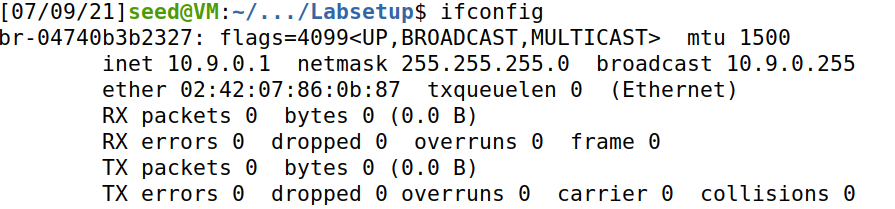
**57118110 杨紫瑄**

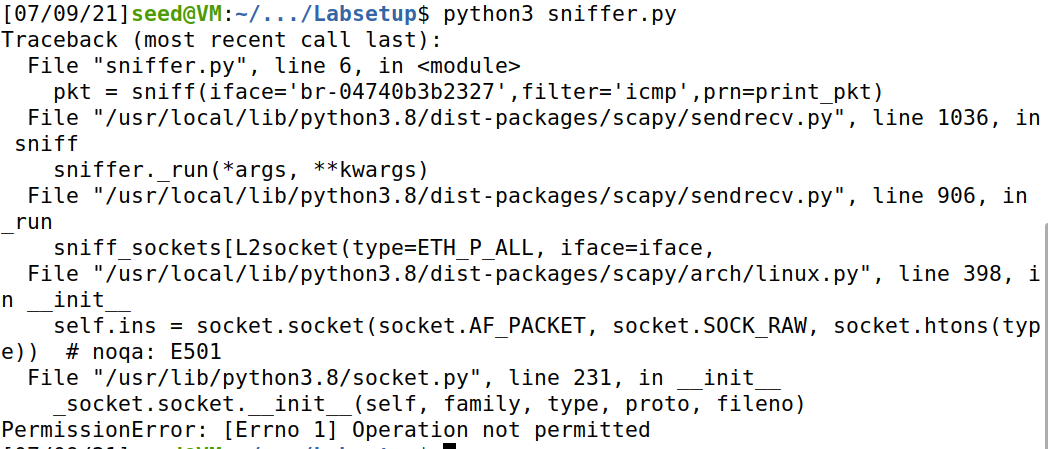
**Task1.1A**



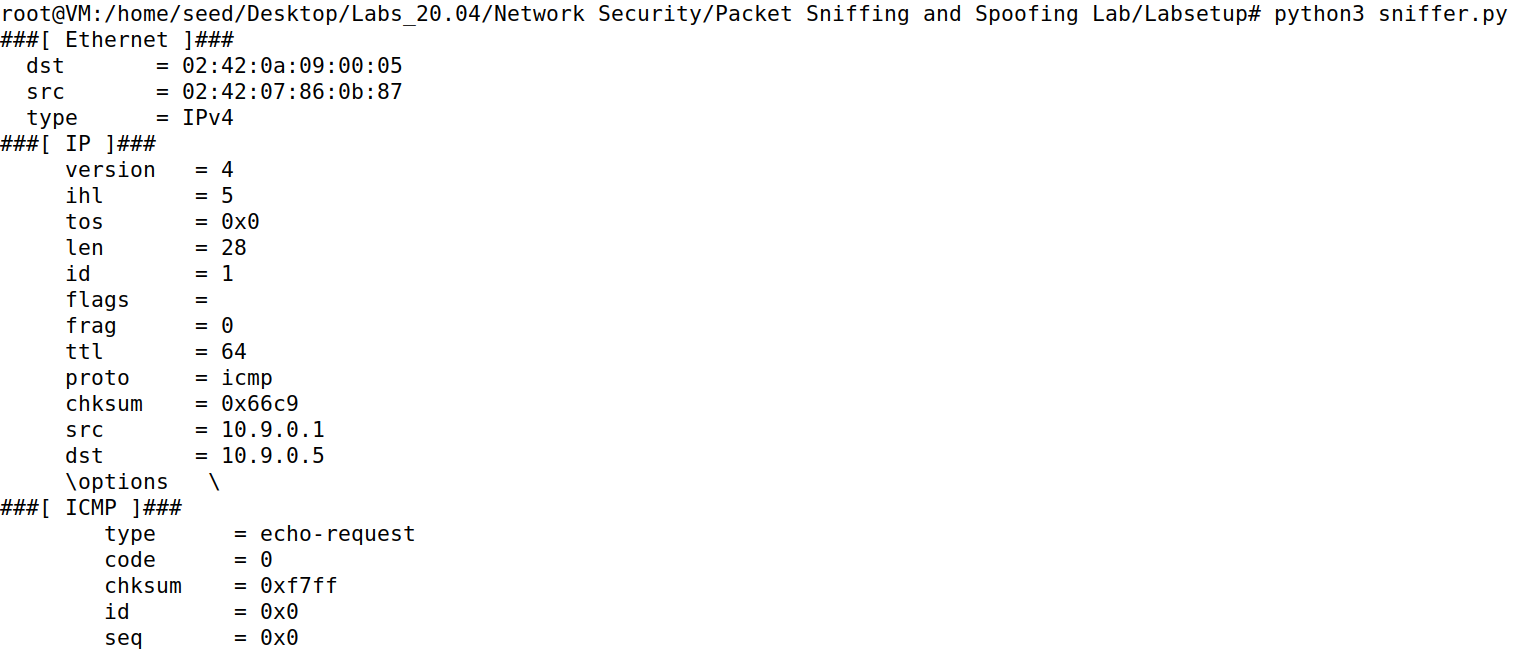
查询自己的iface,为br-04740b3b2327



新建sniffer.py，内容如上图



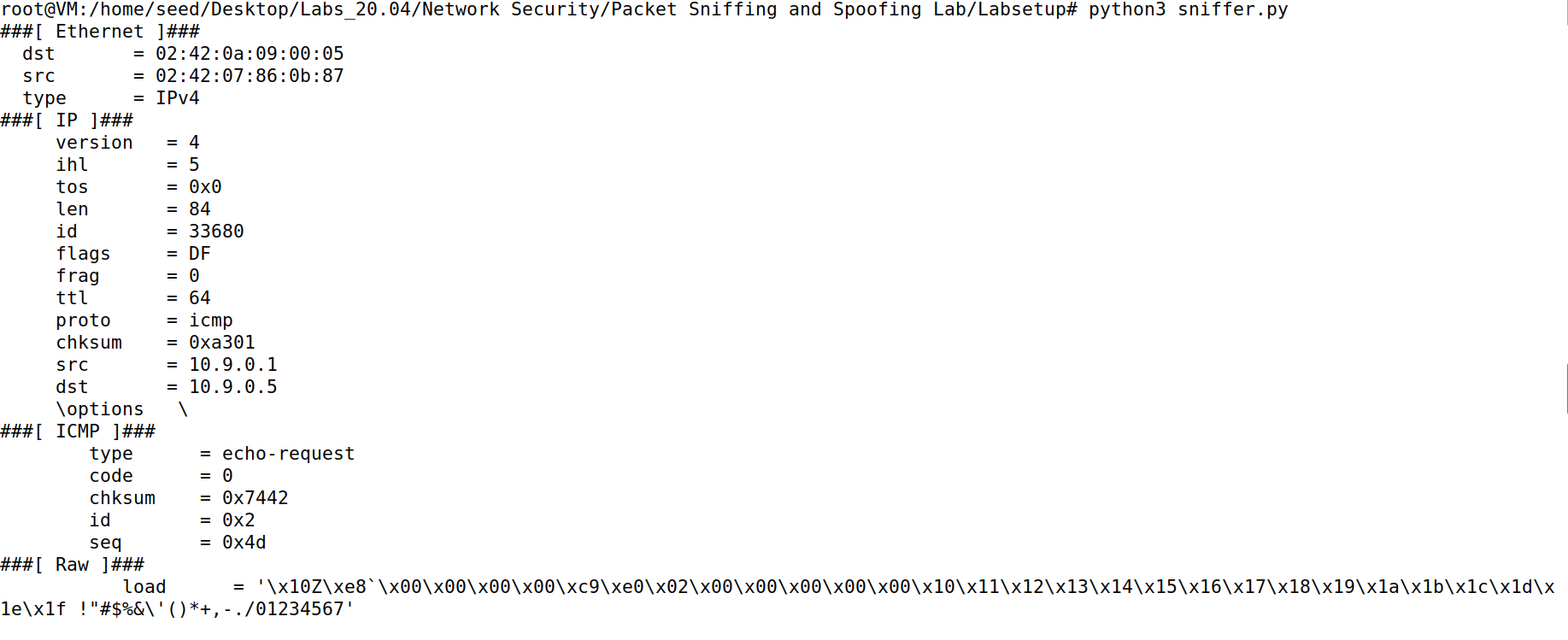
在用户态下运行sniffer.py发现运行失败，权限不够



进入root后再次运行sniffer.py，成功捕获

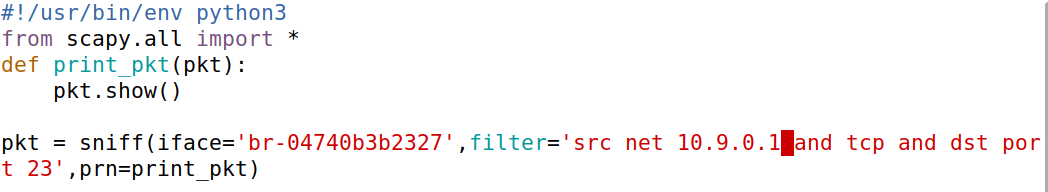
**Task1.1B**

**只捕获ICMP报文**

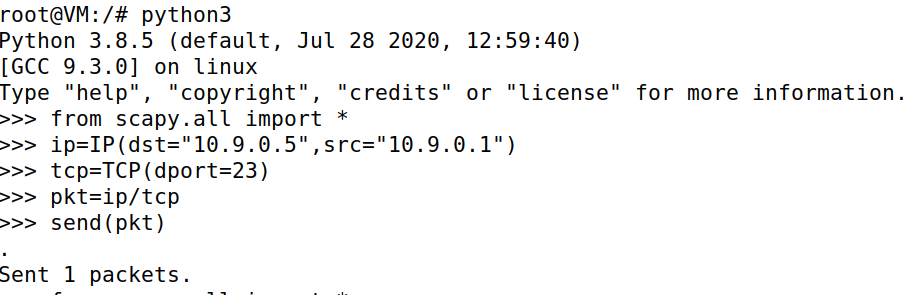


sniffer.py文件内容同1.1.A，捕获的报文如上图

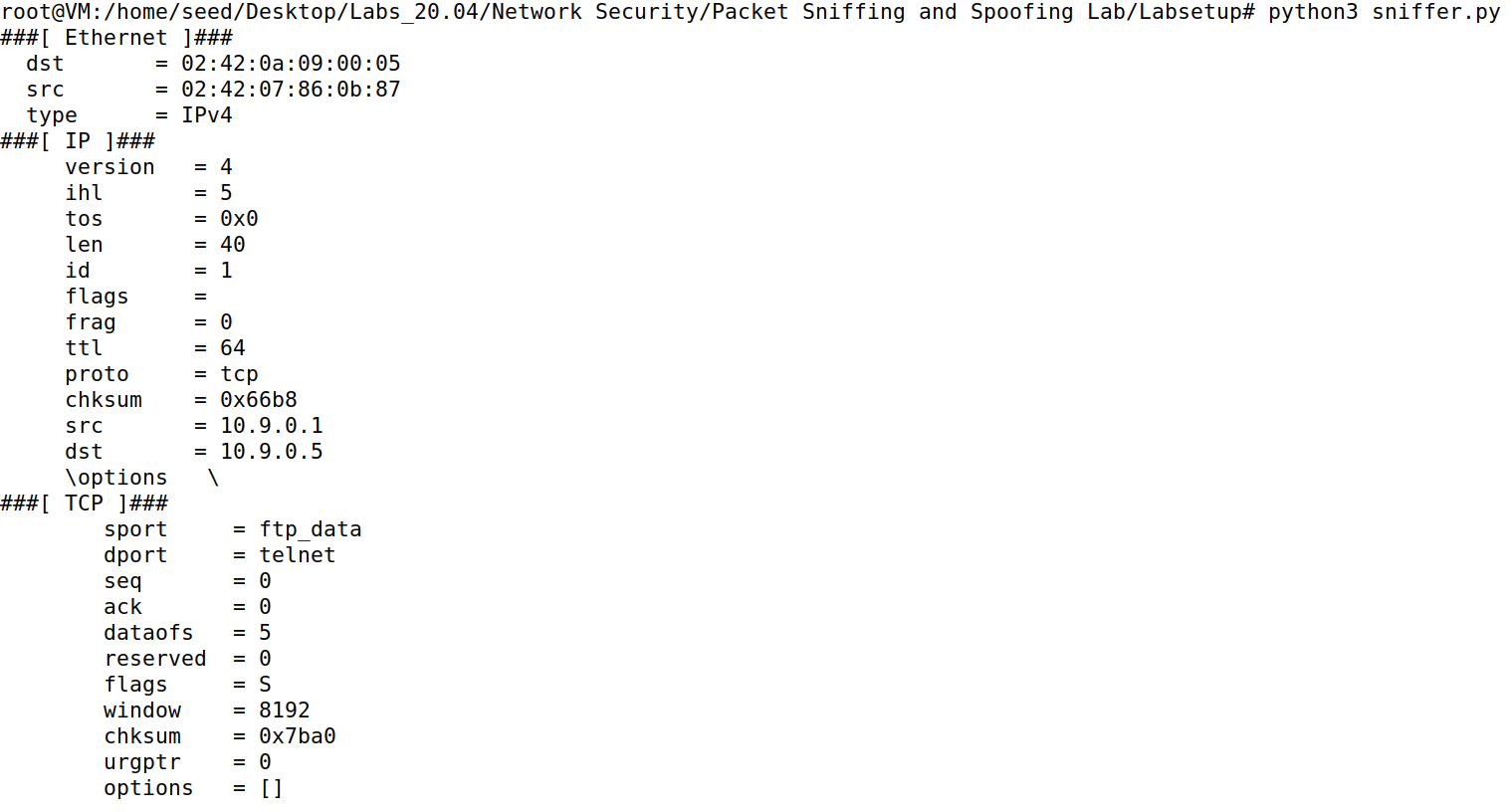
**捕获来自特定IP，目标端口为23的任何tcp数据包**



将sniffer.py文件中filter进行修改如上图

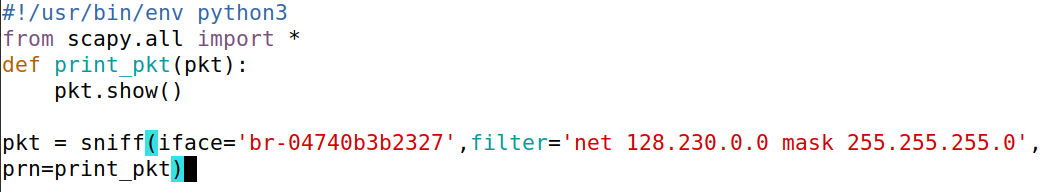


在攻击者docker中发送报文

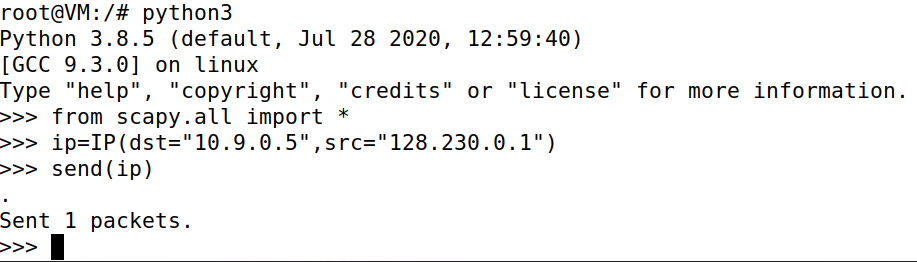


运行sniffer.py，捕获上述报文，其中dport端口为固定值23

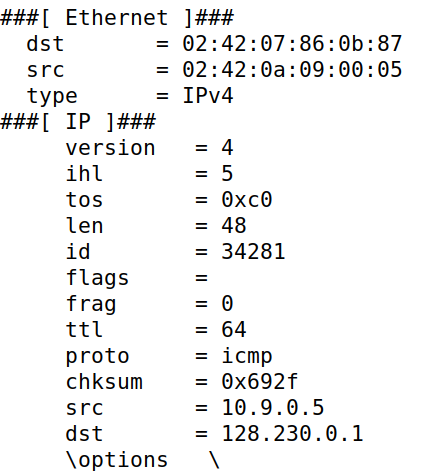
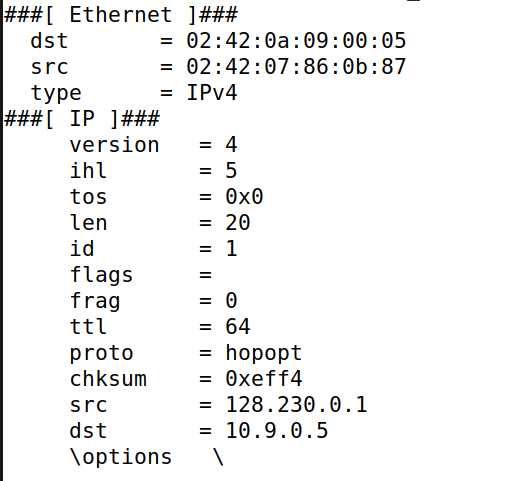
**捕获来自或去往特定子网的报文**



将sniffer.py的filter修改如上图

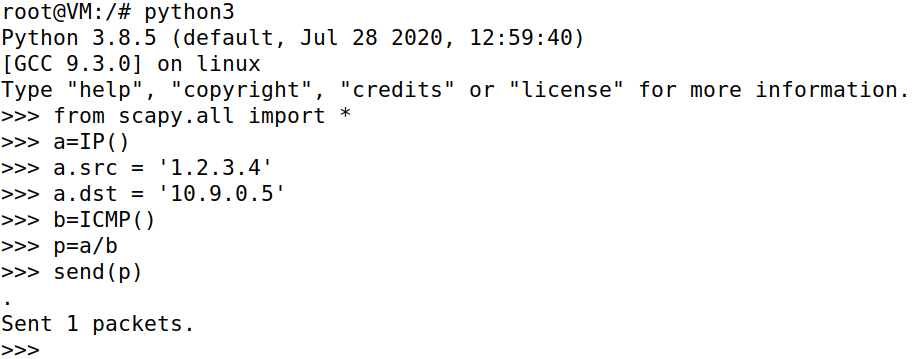


在docker中发送特定子网地址的报文



发现源地址或目的地址为128.230.0.1的报文都可以成功捕获

**Task1.2**

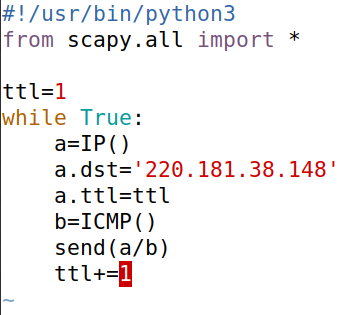


在docker中发送伪造源地址为1.2.3.4的报文

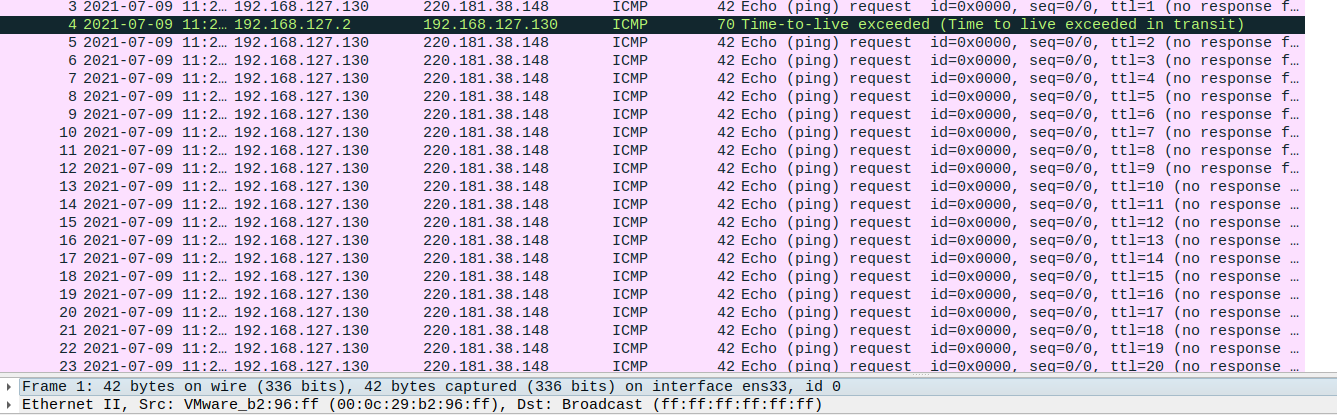


使用wireshark发现能够捕获该报文

**Task1.3**



新建traceroute.py文件设置目标地址为baidu.com的220.181.38.148发送多个报文

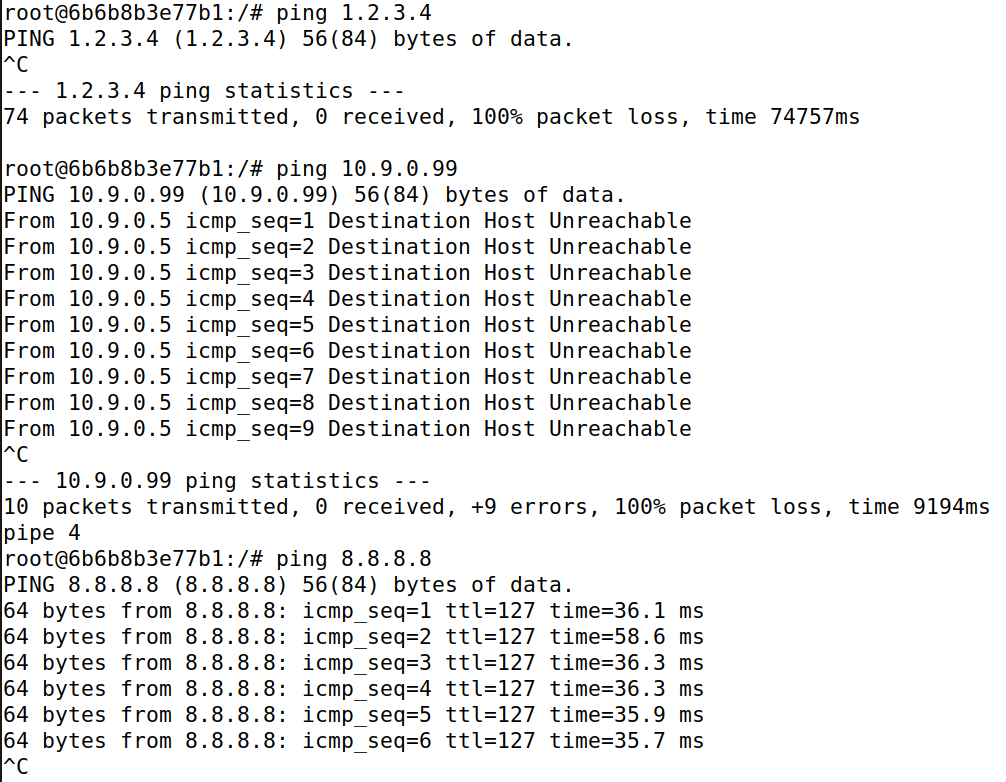


使用wireshark捕获报文，但由于未联网，所以尽管ttl数据较大也无响应

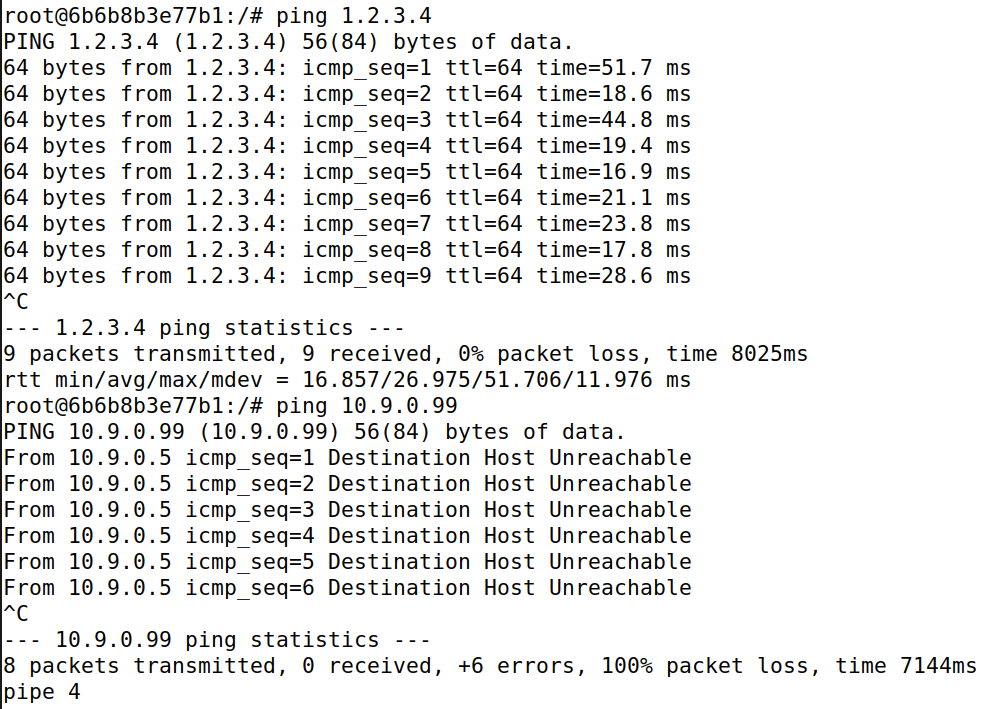
**Task1.4**

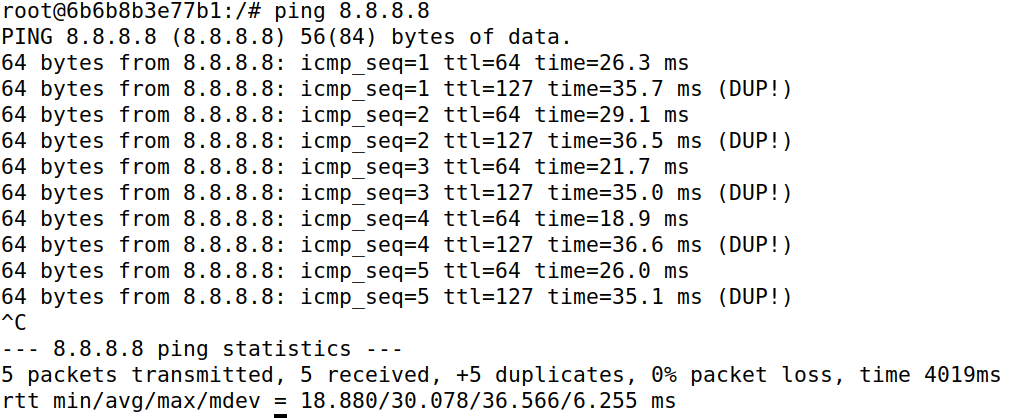


构建sniff\_spoof文件，通过捕获ICMP报⽂，并将其源地址与目的地址交换，并设置ICMP类型为Reply，再发出后，就可以伪造ICMP的 Reply。



运行程序前的ping发现1.2.3.4和10.9.0.99是不可达的，而8.8.8.8可达





运行程序后，10.9.0.99仍旧不可达，而1.2.3.4以及8.8.8.8可达

网关的地址为10.9.0.1，也是attacker的地址，运行程序后外网的两个地址1.2.3.4以及8.8.8.8能ping通主要是因为他们的报文要经过attacker出去，所以被 attacker检测到，并伪造了返回报文，让本机误以为可以ping通，但是ping内网的地址时，不需要经过网关attacker，所以attacker没有返回伪造报文，而 10.9.0.99这个内网地址又是不存在的，所以仍旧和未运行程序时一样ping不通。