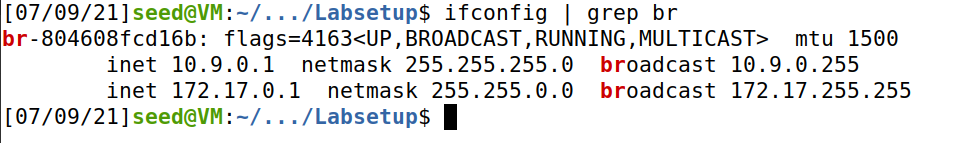
TASK 1.1

1.1A

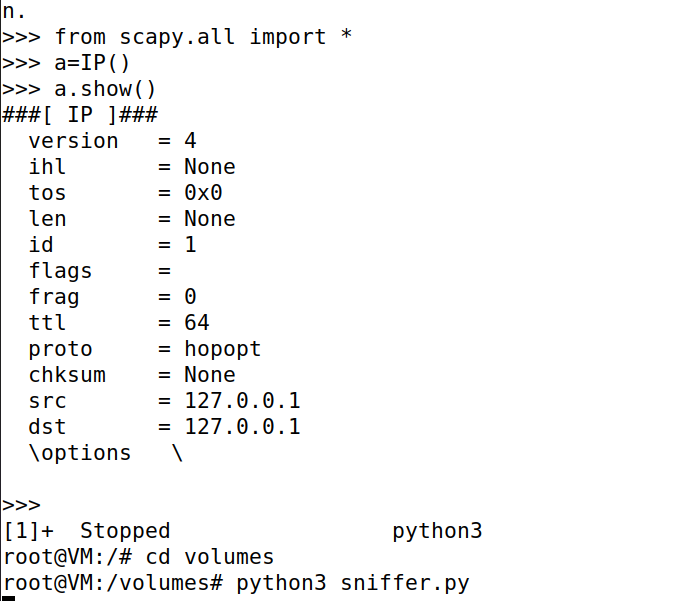
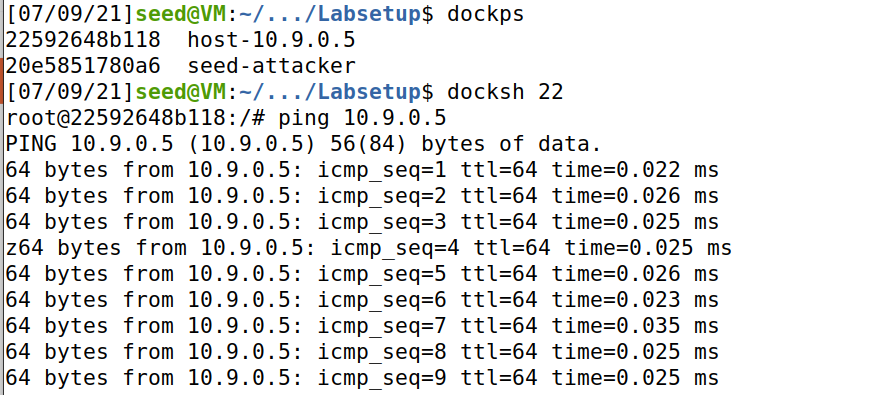
用ifconfig查询自己的iface， 为br-804608fcd16b



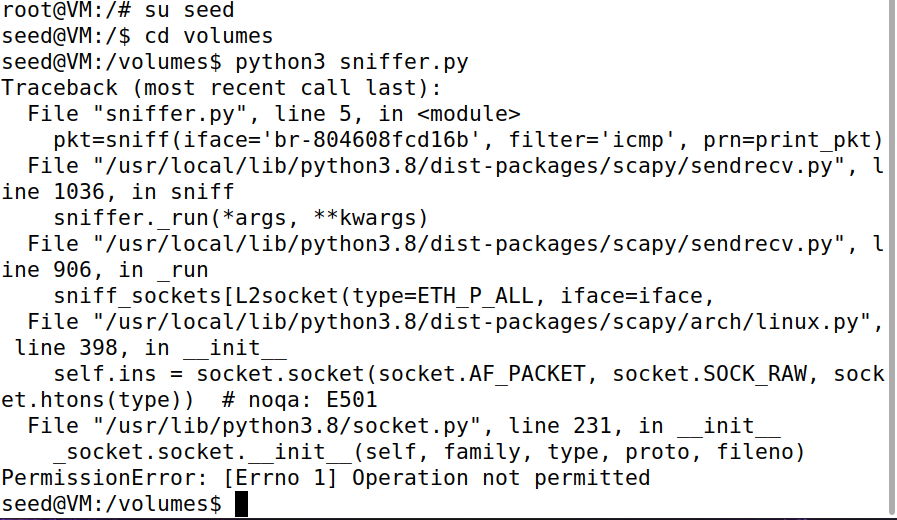
在 volumes 目录下新建 sniffer.py，写入下图：



以root权限运行sniffer.py，新开一个命令行对主机ip进行ping命令



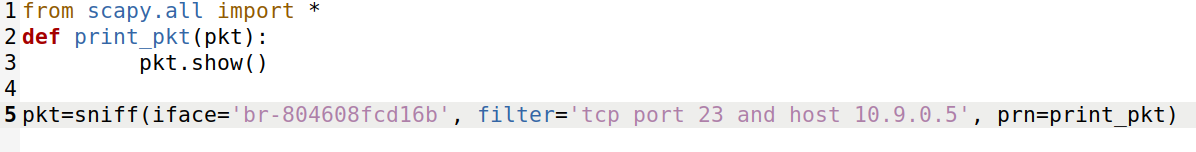
以seed用户运行sniffer.py时，系统会报错



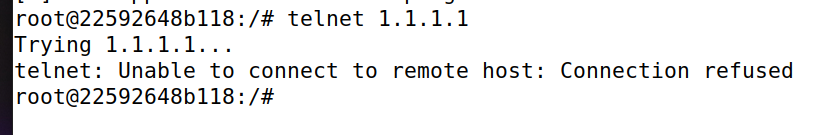
Task 1.1B

只抓取icmp报文，见task1.1a

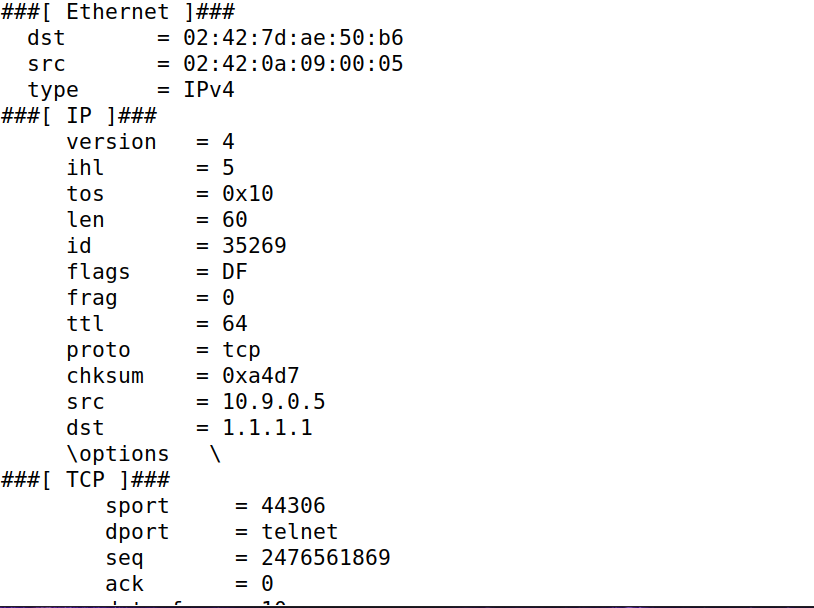
捕获任何来自特定ip的tcp数据包，目的端口为23



利用docksh获取host的shell，Telne任意一个ip地址建立连接



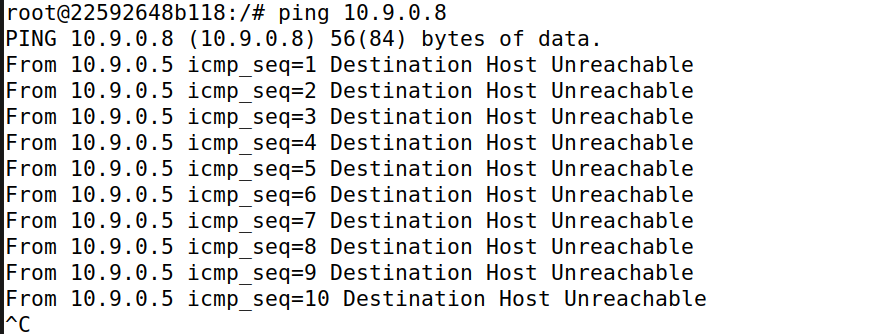
在另一处看见的tcp数据包

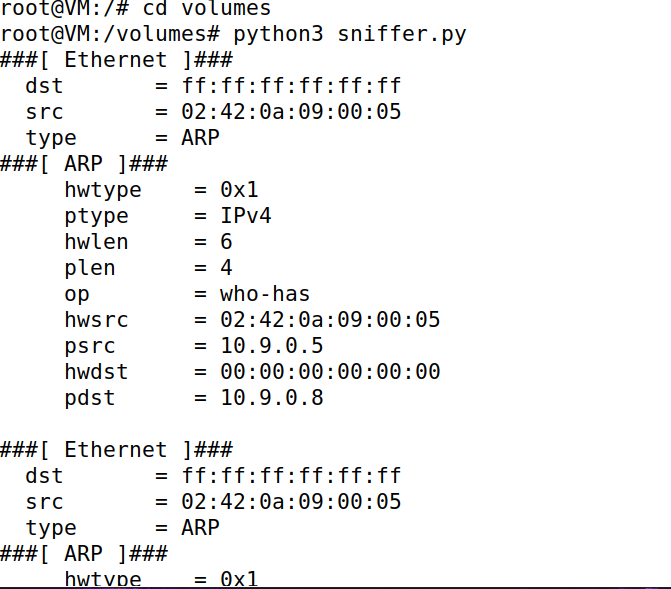


捕获来自或去特定子网的数据包。 可以选择任何子网，如128.230.0.0/16；不应该选择VM所绑定的子网。

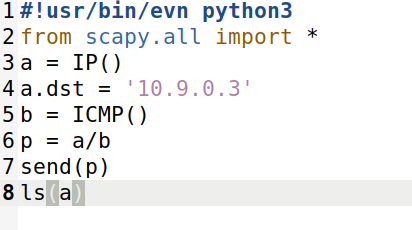


直接ping 10.9.0.8， 可获得捕获的数据包。

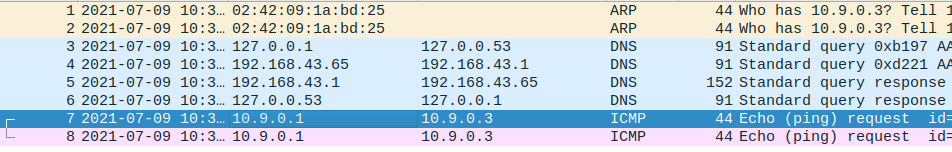
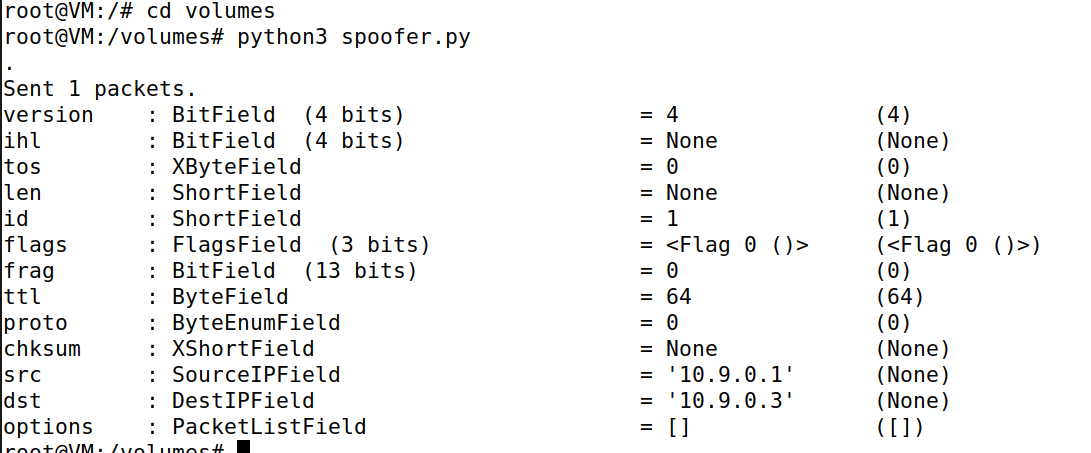




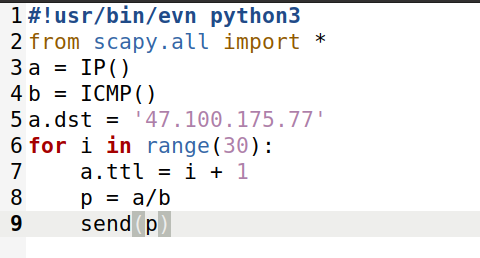
Task1.2



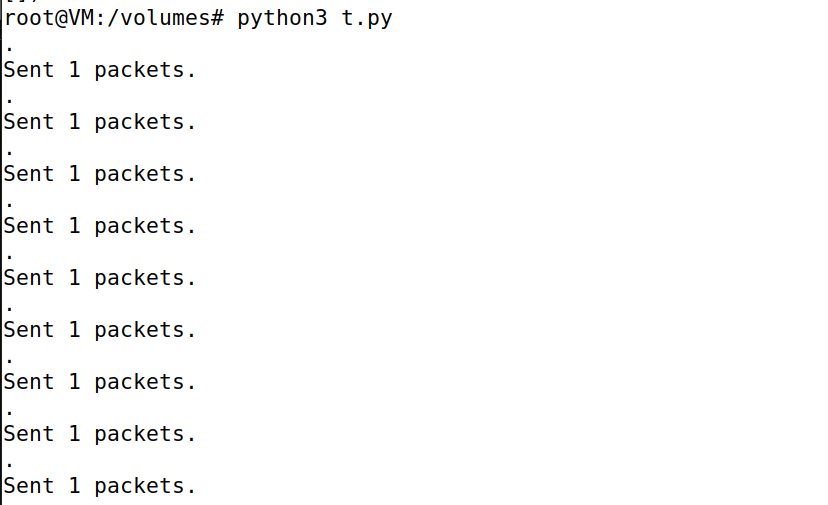
第一行创建了一个ICMP对象，默认类型为echo request。在第六行中，我们将a和b堆叠在一起形成了一个新对象，“/”操作符被重载，不在表示除法，而是将b添加为a的有效负载字段，并相应地修改a的字段。最终我们得到一个表示ICMP数据包的新对象，报文重组后，向子网内的一个IP发送数据包，打开wireshark可观测发送数据包和响应数据

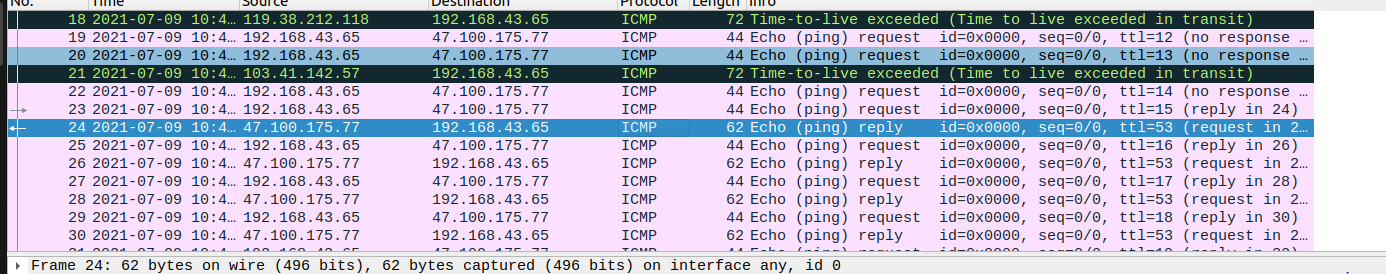


Task1.3

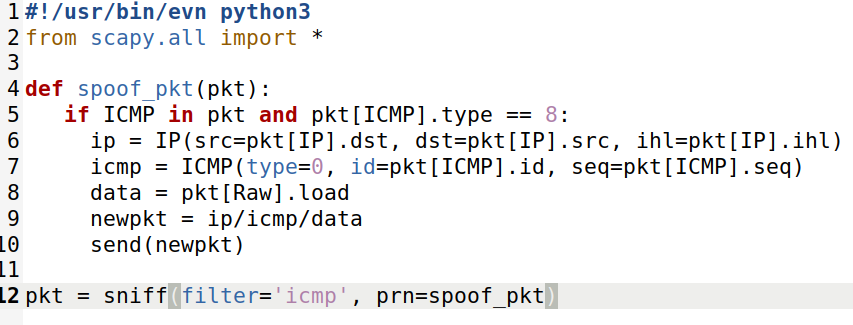


创建一个文件t.py，向目标ip发送 ICMP 数据包，一开始设置 TTL值 为 1, 那么发出的ICMP数据包在经历一个路由结点后, 就会失活被抛弃,我们利用循环, 不断增加TTL的值，最终使得数据包到达目的地。

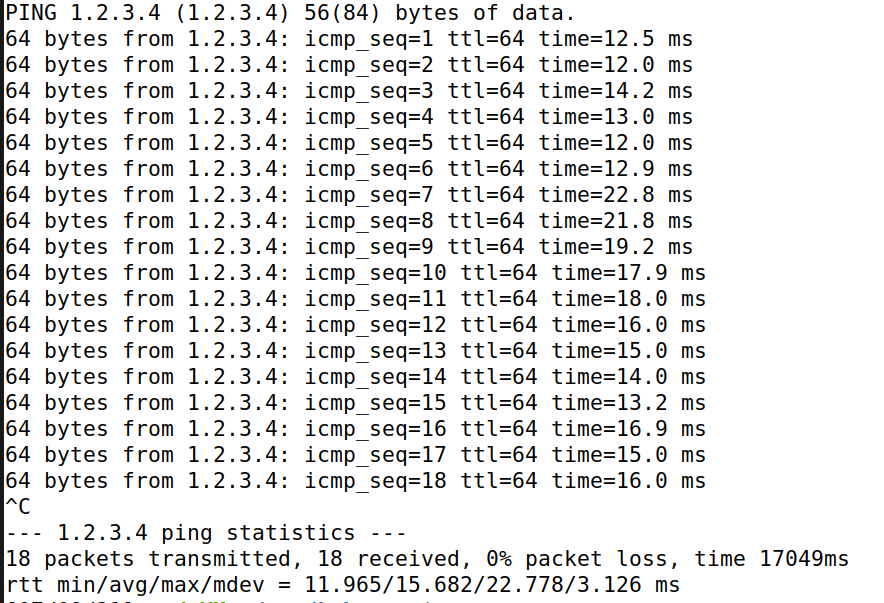




Task 1.4

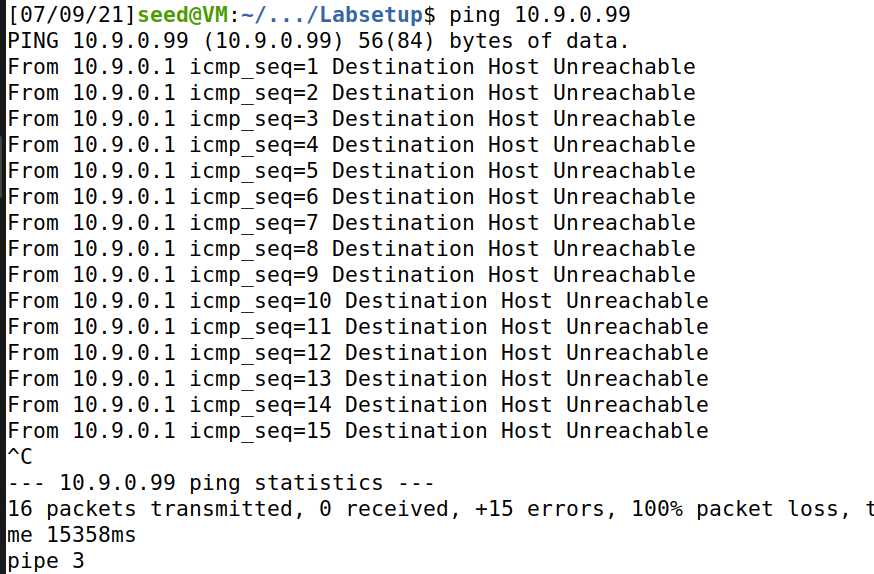


Ping 1.2.3.4 # a non-existing host on the internet



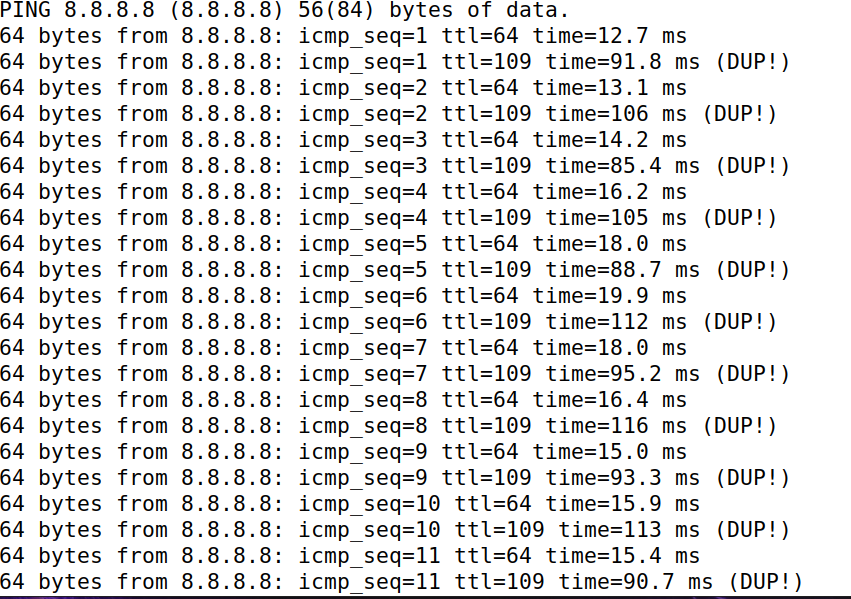
可以看到当我们ping一个网络上不存在的ip时，由于伪造报文，我们仍可以接收到响应。

Ping 10.9.0.99 # a non-existing host on the LAN



对于局域网内不存在的主机，先利用ARP进行MAC地址询问，由于一直得不到结果，所以没有ICMP报文，也就不存在报文欺骗。

Ping 8.8.8.8 #an existing host on the internet



对于网络上存在的主机，我们可以看到每个序列号的报文都存在一个重复报文，我们可以知道TTL=64，且时间较短的那个报文是伪造的报文。