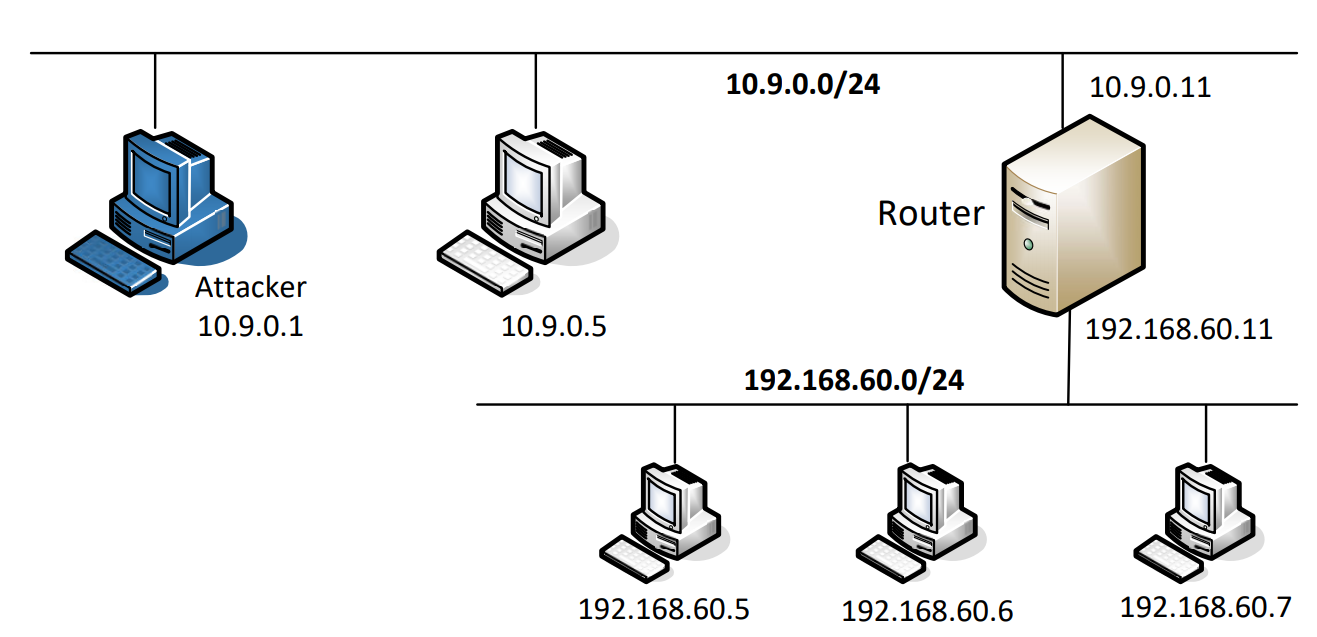
第六次作业

卞郡菁

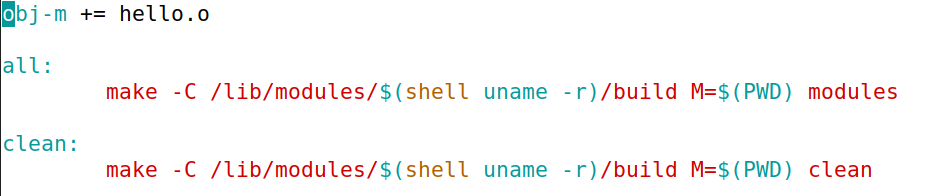


Desktop/Labs\_20.04/Network\ Security/Firewall\ Exploration\ Lab/Labsetup/

**Task1.A**

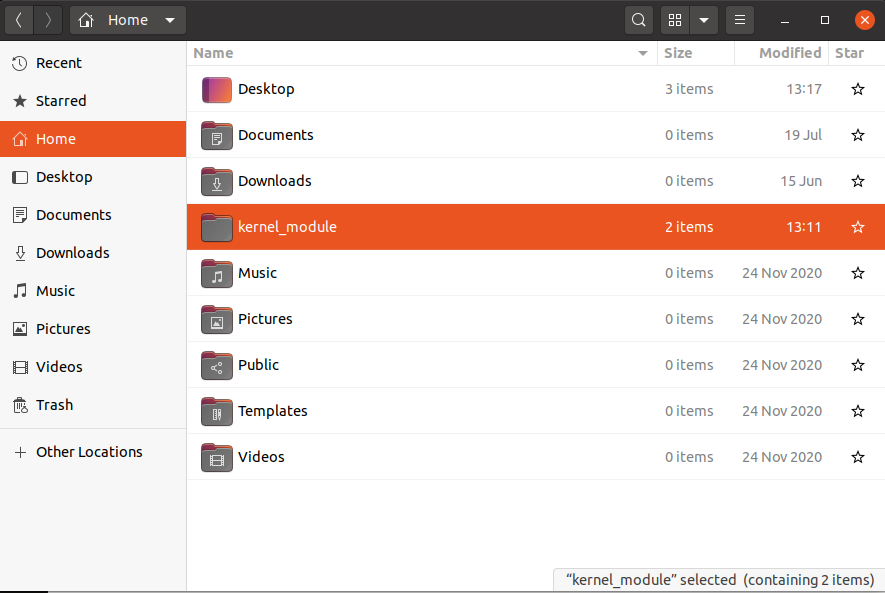


hello.c

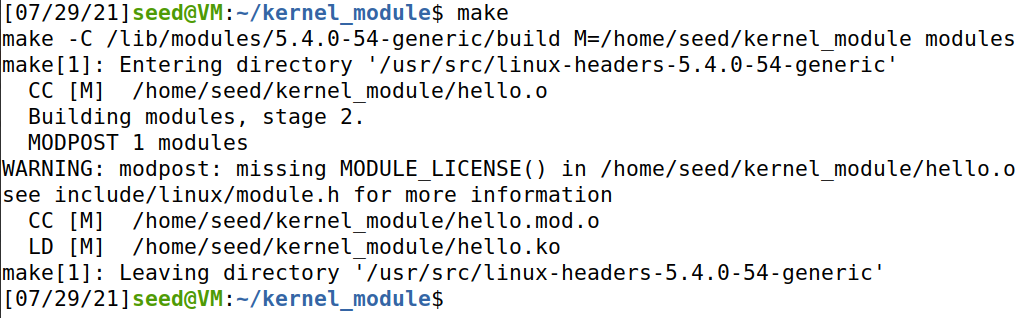


Makefile

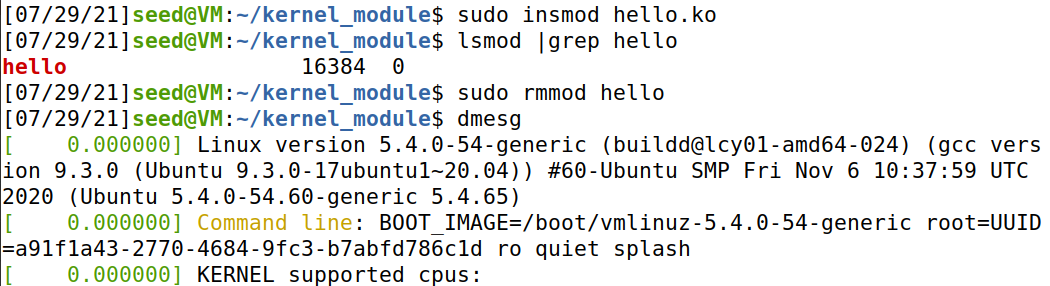
拷贝文件夹

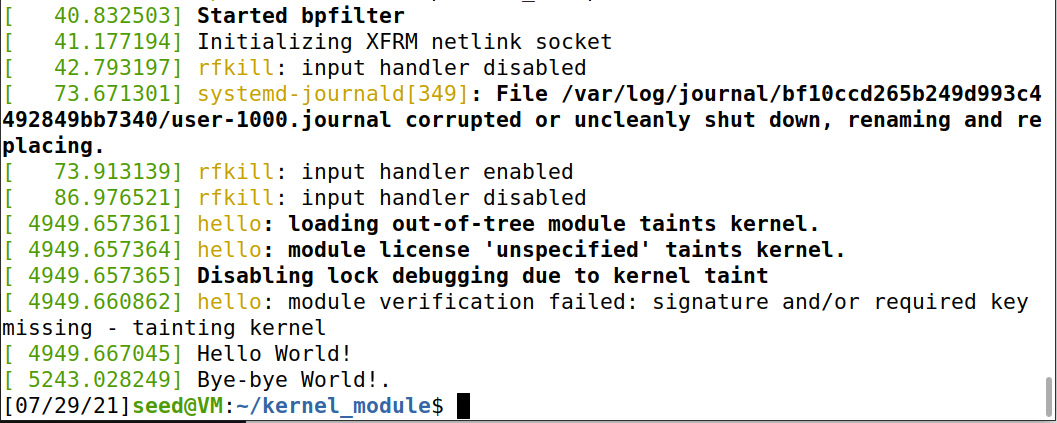


在makefile所在的文件夹中执行make，make成功，上述C程序被编译成 一个 loadable 内核模块



使用 sudo insmod hello.ko 命令加载模块并使用 lsmod | gre

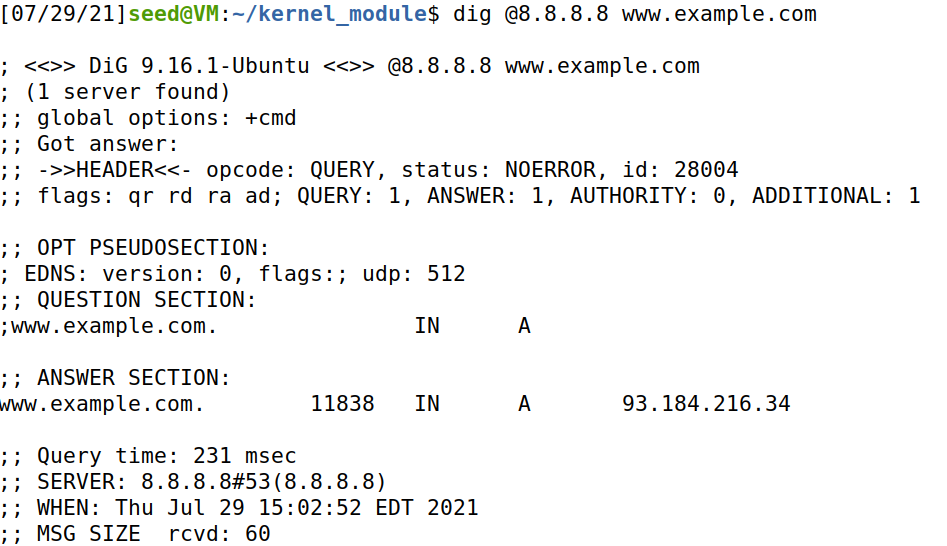




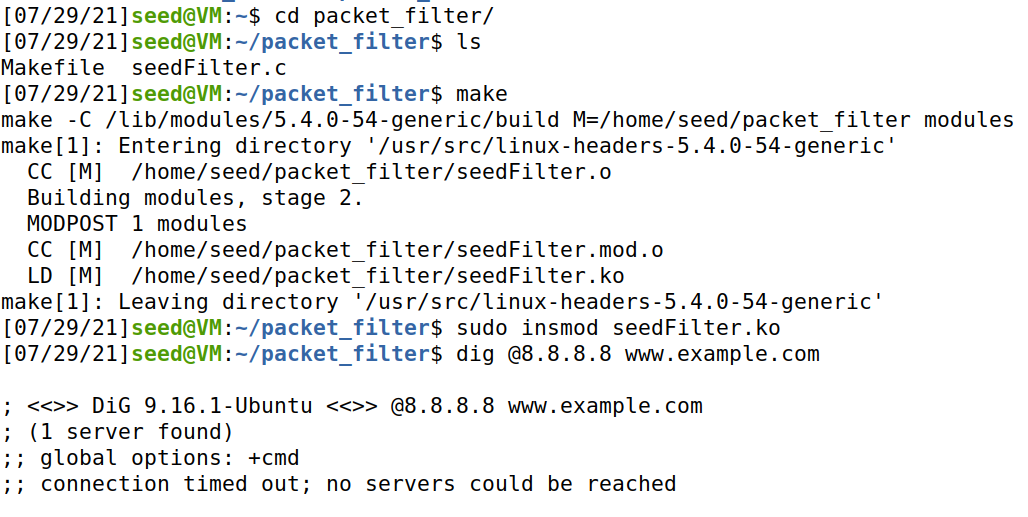
**Task1.B**

（1）

编译示例代码前使用命令 dig @8.8.8.8 www.example.com 生成到谷歌服务器 的 udp，成功查询到 DNS 信息，说明数据包能够成功到达



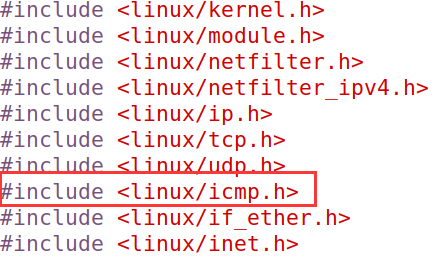
在 packet\_filter 文件夹编译加载packet\_Filter LKM

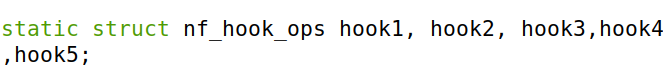


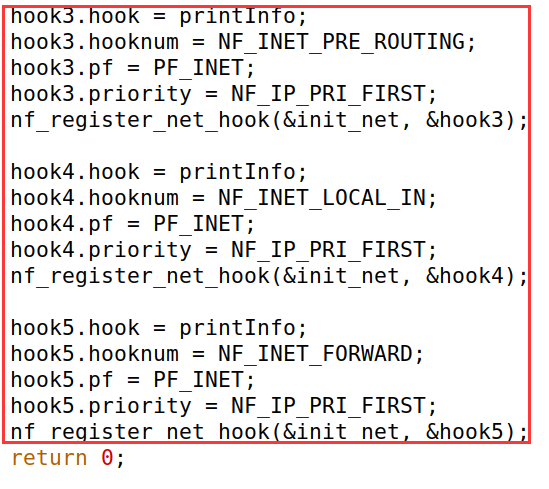
发现连接超时，无法达到服务器。说明防火墙成功工作

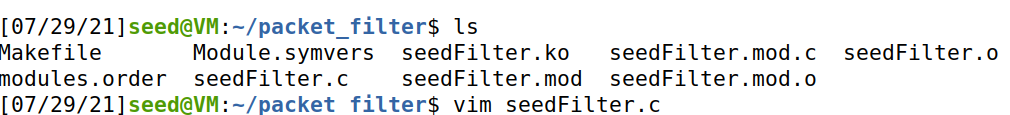
（2）每个hook函数在什么情况下会被调用？

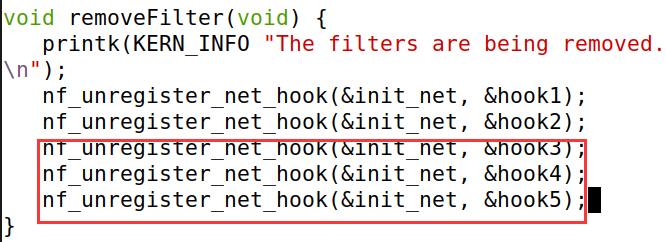
修改seed\_Filter.c文件，



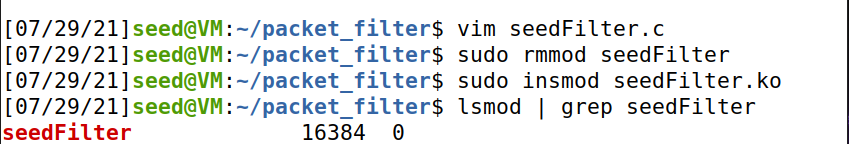




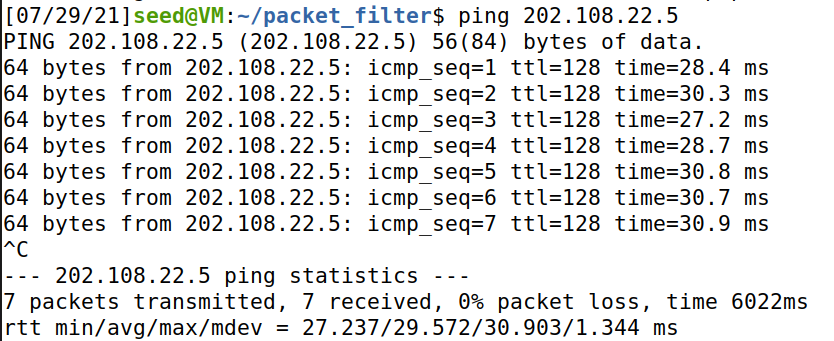




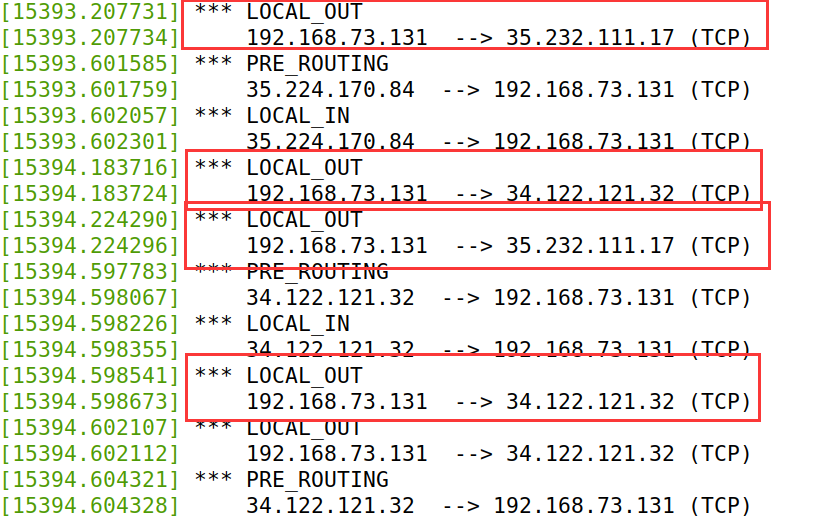
重新编译和加载（加载前删除以前的模块）



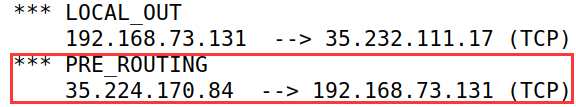
我们可以 ping 下百度地址 202.108.22.5，



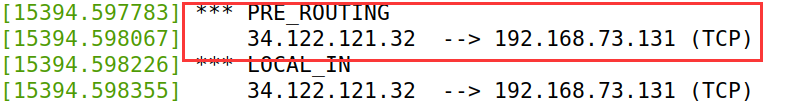
NF\_IP\_LOCAL\_OUT:在数据包以其方式离开主机之前。



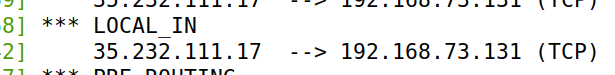
NF\_IP\_POST\_ROUTING:数据包离开主机并进入不同的网络之后。



NF\_IP\_PRE\_ROUTING:在做出任何路由决策之前



NF\_IP\_LOCAL\_IN:在发送到网络堆栈之前

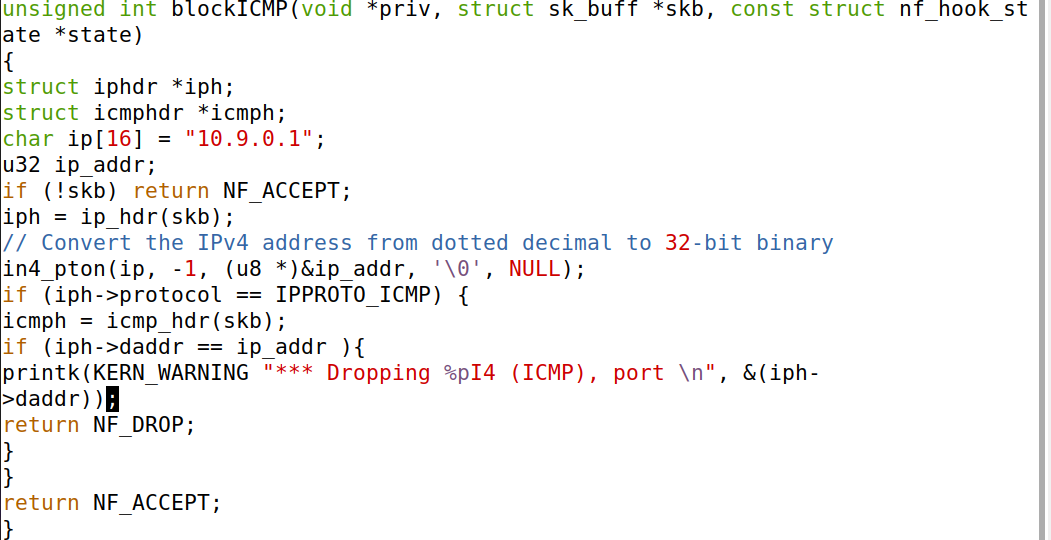


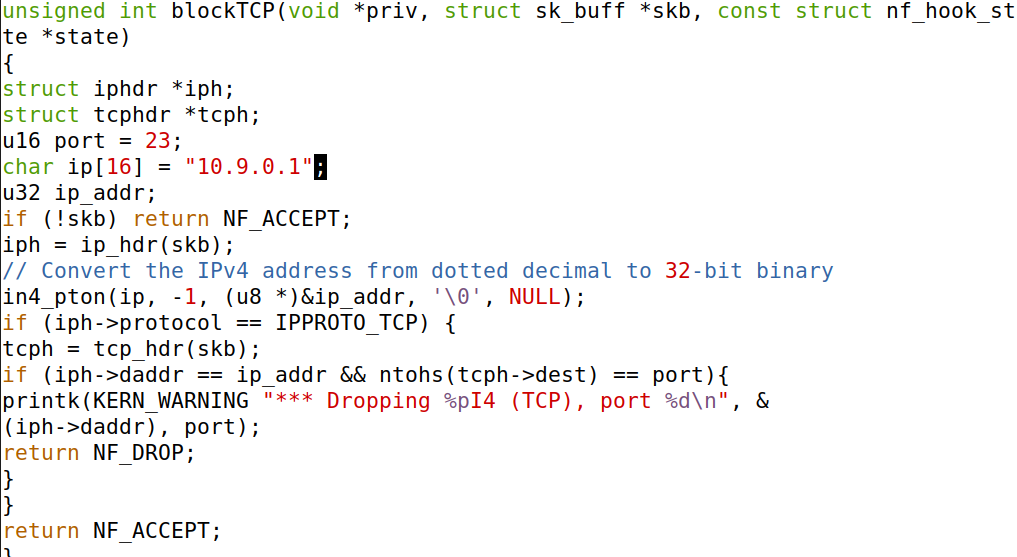
NF\_IP\_FORWARD:向其他主机转发报文。被转发的数据包才会通过FORWARD，因此没有捕获到相关报文。

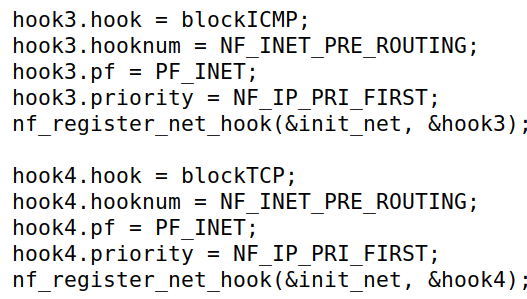
数据报从进入系统，进行IP校验以后，首先经过第一个HOOK函数NF\_INET\_PRE\_ROUTING进行处理；然后就进入路由代码，其决定该数据报是需要转发还是发给本机的；若该数据报是发被本机的，则该数据经过HOOK函数NF\_INET\_LOCAL\_IN处理以后然后传递给上层协议；若该数据报应该被转发则它被NF\_INET\_FORWARD处理；经过转发的数据报经过最后一个HOOK函数NF\_INET\_POST\_ROUTING处理以后，再传输到网络上。本地产生的数据经过HOOK函数NF\_INET\_LOCAL\_OUT 处理后，进行路由选择处理，然后经过NF\_INET\_POST\_ROUTING处理后发送出去。

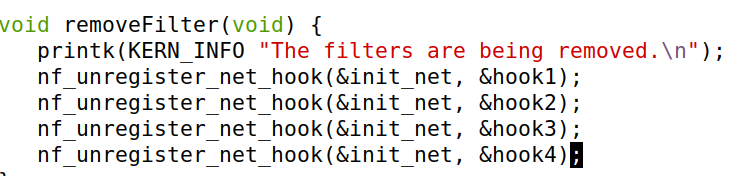
（3）再实现两个HOOK，实现以下目的:(1)防止其他计算机ping VM，(2)防止其他计算机telnet到VM。请实现两个不同的HOOK函数，但将它们注册到同一个netfilter

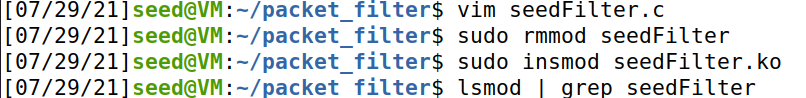


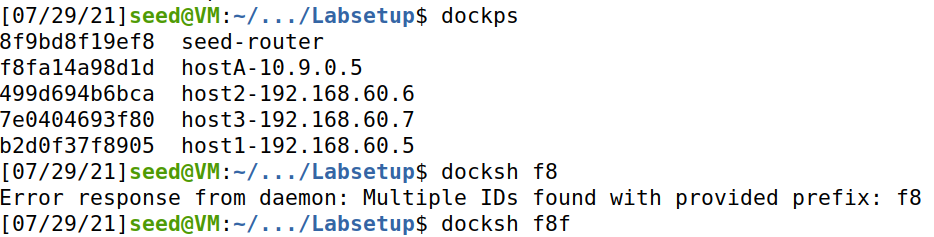




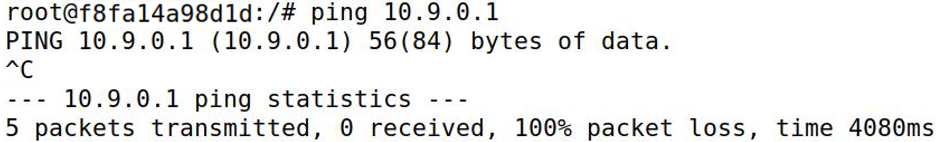


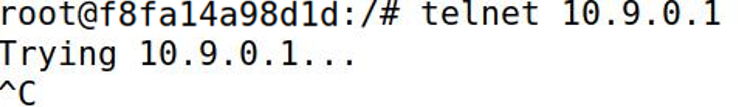






可以看到无法ping通或进行TELNET





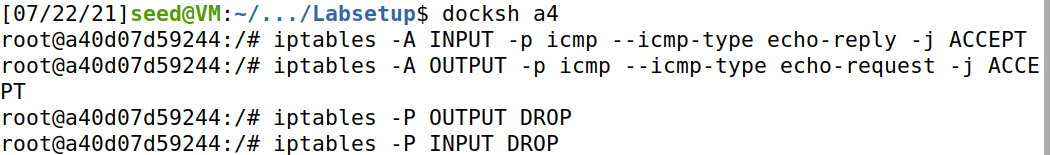
**Task 2: Experimenting with Stateless Firewall Rules**

**TASK2.A**

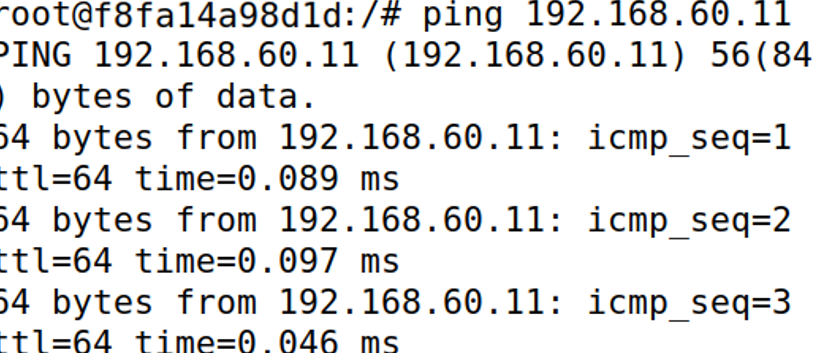
用户主机的IP地址为10.9.0.5，路由器的IP地址为10.9.0.11，内网网段的IP地址192.168.60.0/24。

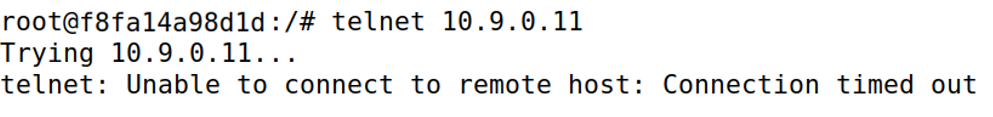
1. 你能ping通路由器吗；能telnet到路由器(一个telnet服务器运行在所有容器上;在它们上面创建了一个名为seed的帐户，密码为dees)。

首先在路由器上设置以下过滤规则：

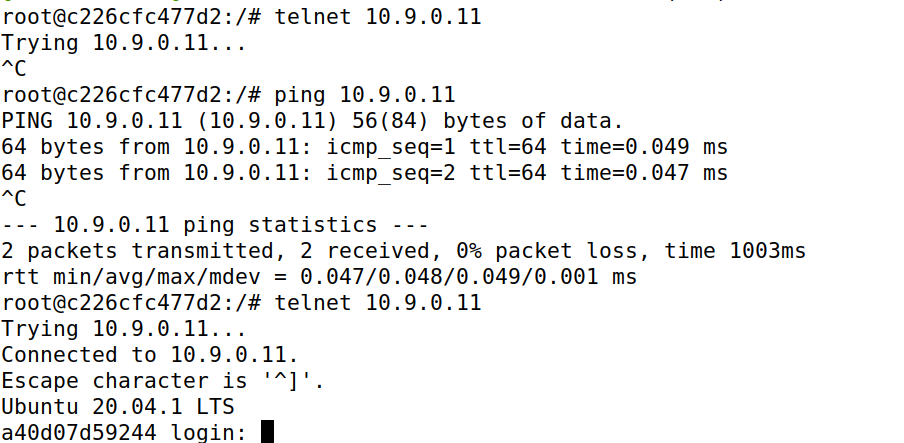


结果发现，从10.9.0.5上可以ping通路由器，但无法telnet到路由器





（2）将上述规则取消掉，发现可以ping和telnet



（3）请报告你的观察并解释每条规则的目的。

①

iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT （将手册中的顺序换了一下）

iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT

上述两条规则表示外部主机可以ping通防火墙，即其他主机可以ping通防火墙主机（即router），防火 墙接收icmp的request请求报文，也可以发出icmp相应报文。

②

iptables -P OUTPUT DROP丢弃所有外出的包

iptables -P INPUT DROP router所有进入的包都被丢弃了，但是外出的包不受限制。

**Task2.B: Protecting the Internal Network**

在这个任务中，我们将在路由器上设置防火墙规则来保护内部网络192.168.60.0/24。

为此，我们需要使用FORWARD链来指定方向。

iptables规则：

iptables -P OUTPUT DROP

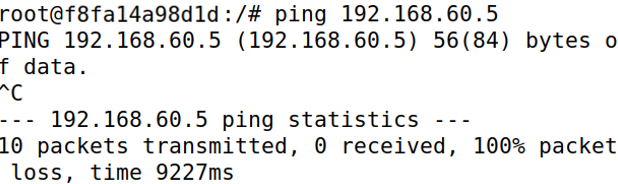
iptables -P INPUT DROP

iptables -A FORWARD -p icmp --icmp-type echo-request -o eth1 -j DROP

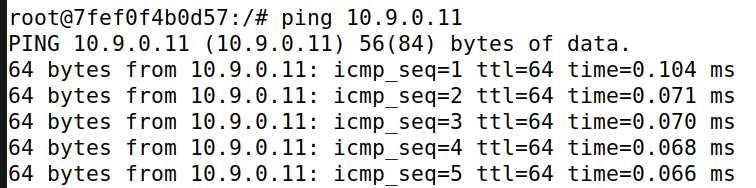
iptables -I OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT

iptables -I INPUT -p icmp --icmp-type echo-request -j ACCEPT

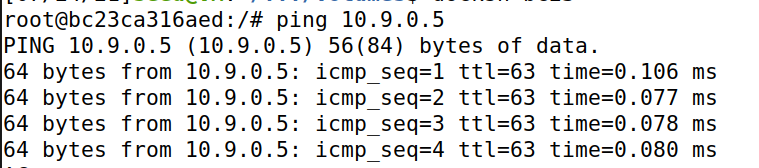
1）外部主机不可以ping通内部主机：发现从10.9.0.5 ping到内网全部丢失



2）外部主机可以ping通router



在IP地址为192.168.60.5的内网主机上ping用户主机，得到结果如下，可知可以连接



在用户主机上telnet远程连接内网主机192.168.60.5，连接失败

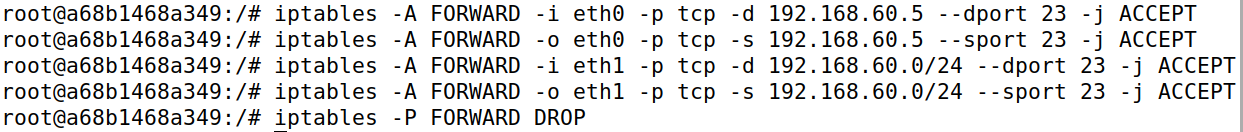


在IP地址为192.168.60.5的内网主机telnet远程用户主机，连接失败

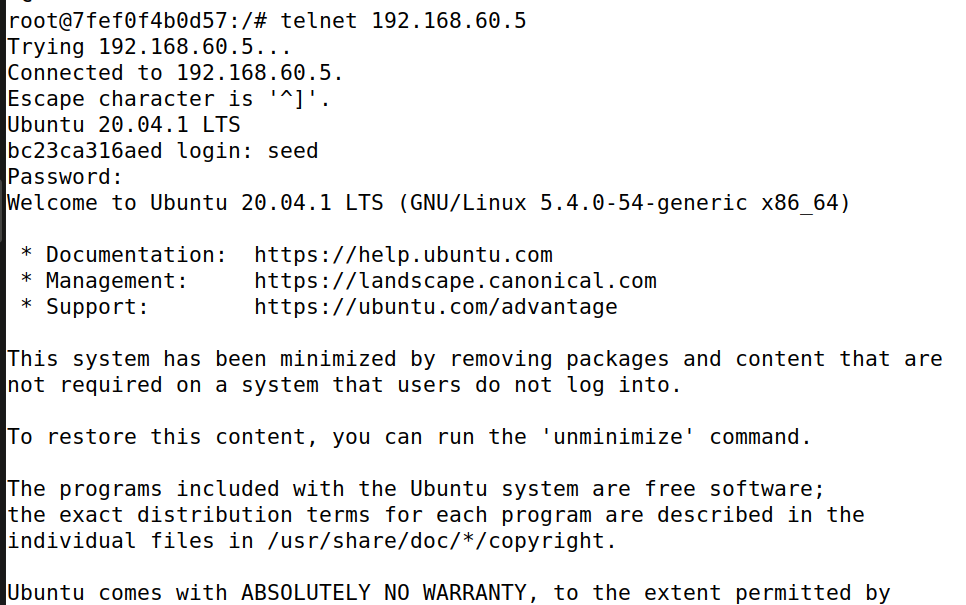


**Task 2.C: Protecting Internal Servers**

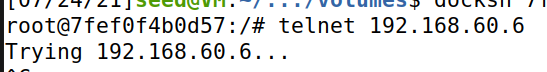
在路由器上利用ptables命令，创建过滤规则如下。



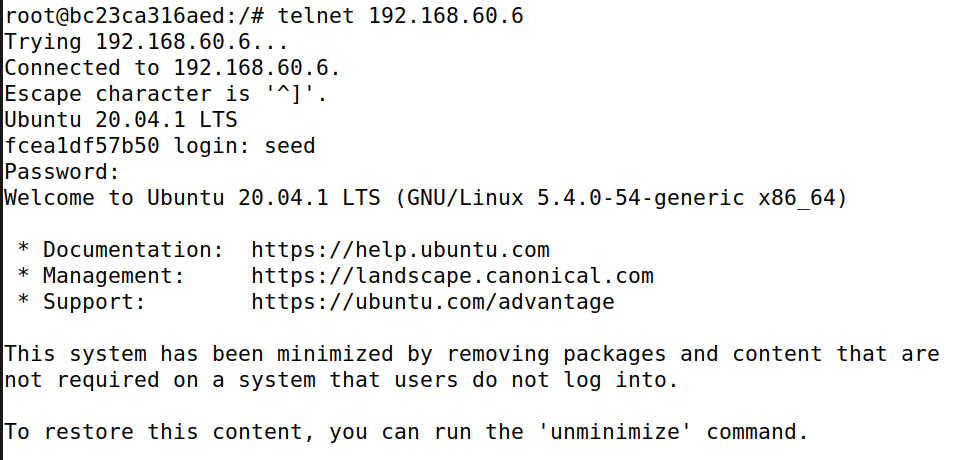
在用户主机上telnet远程内网主机192.168.60.5



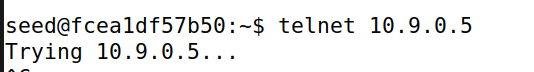
在用户主机上telnet远程连接内网主机192.168.60.6连接失败



在IP地址为192.168.60.5的内网主机上telnet远程连接内网主机192.168.60.6，得到结果如下，连接成功



在IP地址为192.168.60.5的内网主机上telnet远程连接用户主机，得到结果如下，可知连接失败

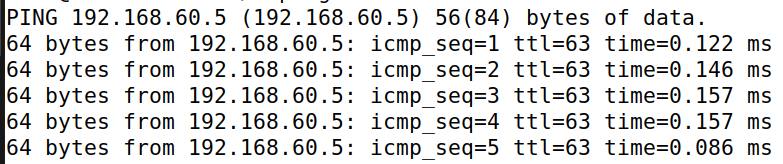


**Task 3: Connection Tracking and Stateful Firewall**

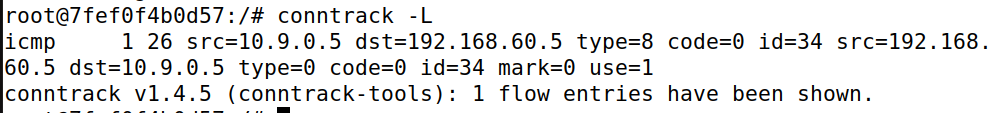
**Task 3.A: Experiment with the Connection Tracking**

ICMP experiment

在用户主机10.9.0.5上ping 192.169.60.5

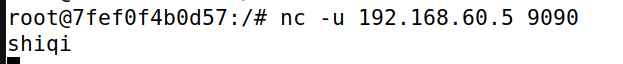


之后使用conntrack -L检查路由器中的追踪信息，结果如下，ICMP 的连接时间约为27s



UDP experiment

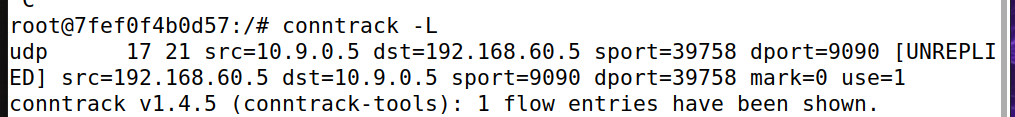
在用户主机10.9.0.5上发送UDP报文



在192.168.60.5上开启netcat服务，监听结果如下



在路由器上利用connntracl -L实现连接跟踪，结果如下UDP的连接时间大约为21s



TCP experiment

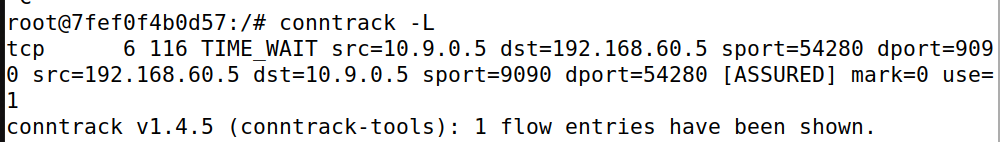
在用户主机上利用TCP远程连接192.168.60.5，并发送如下消息



在192.168.60.5上，开启TCP服务监听结果如下

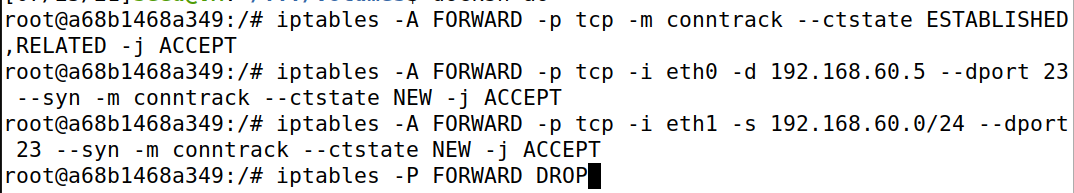


在路由器上利用connntracl -L实现连接跟踪，结果如下，TCP的连接时间大约为116s

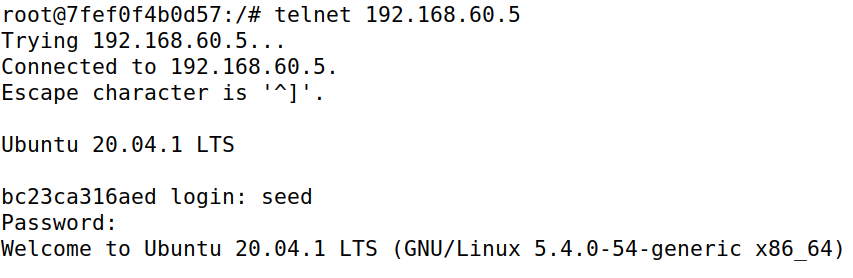


**Task 3.B: Setting Up a Sateful Firewall**

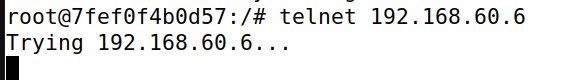
在路由器上利用iptables命令追踪连接并设置过滤规则



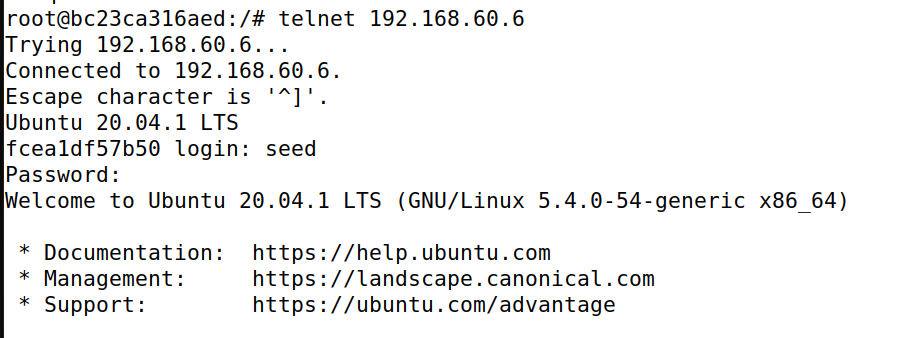
在用户主机上telnet远程连接内网192.168.60.5，得到结果如下，连接成功



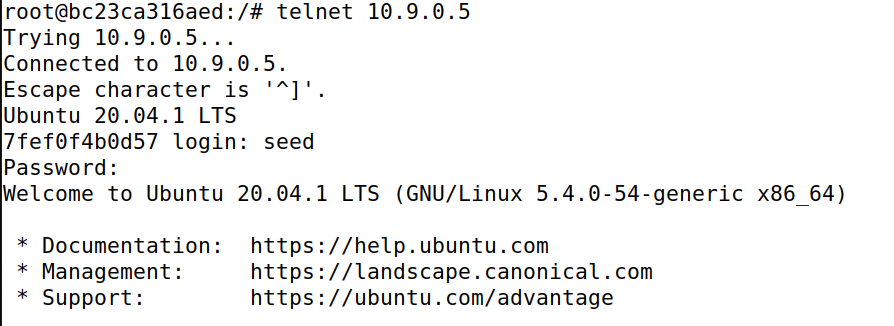
在用户主机上telnet远程连接内网主机192.168.60.6，得到结果如下，可知连接失败



在IP地址为192.168.60.5的内网主机上telnet远程连接内网主机192.168.60.6，得到结果如下，可知连接成功



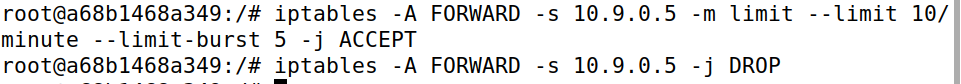
在IP地址为192.168.60.5的内网主机上telent远程连接用户主机，得到结果如下，可知连接成功



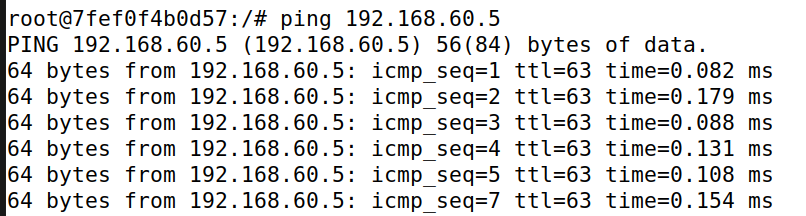
将本次实验与task 2.C进行对比可以发现，不利于连接跟踪机制的过滤规则仅对数据包的首部进行检查，其优点是处理速度快，缺点是无法定义精细的规则，不适合复杂的访问控制；而利用连接跟踪机制的过滤规则对数据包的状态也进行检查，其优点是能够定义更加严格的规则、应用范围更广、安全性更高，缺点是无法对数据包内容进行识别。

**Task 4: Limiting Network Traffic**

利用iptables命令限制经过防火墙的包的数量，限制规则如下



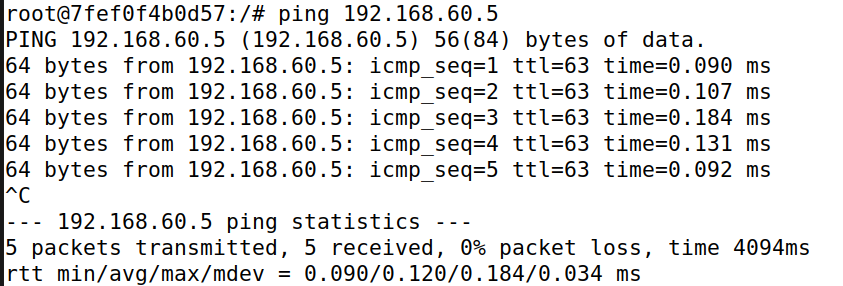
在用户主机上ping内网主机192.168.60.5，得到结果如下。可知能够连接，但部分报文因流量限制而丢失



在路由器上利用iptables命令修改 流量限制规则如下



在用户主机上ping内网主机192.168.60.5，得到结果如下，可知能够连接，且无报文丢失

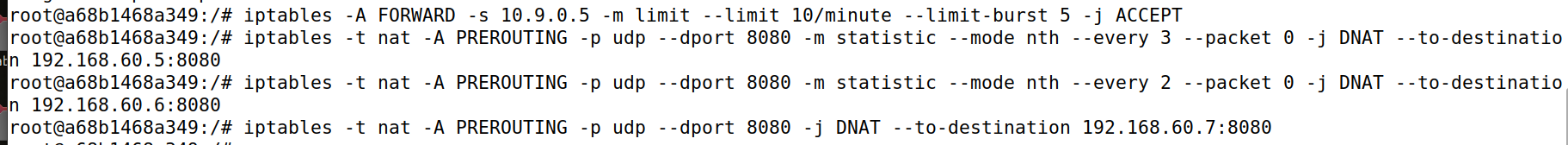


通过对比发现两者之间的差距，区别在于第一次过滤规则中多了第二条规则，出现该现象的原因是路由器的转发链的默认规则为ACCEPT，即使超过流量限制，报文根据默认规则也可以进行传输，会使过滤失败。因此第二条规则是必要的。

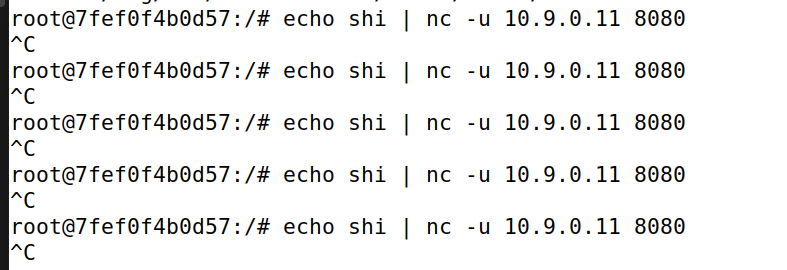
**Task 5: Load Balancing**

Using the nth mode(round-robin)

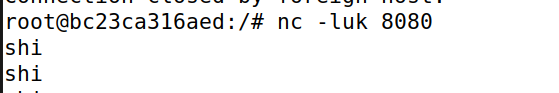
在路由器上利用iptables命令，采用nth模式创建负载均衡规则如下



在用户主机上向路由器发送UDP数据包如下：



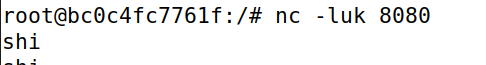
在服务器192.168.60.5上监听8080端口，结果如下



在服务器192.168.60.6上监听808端口，结果如下：



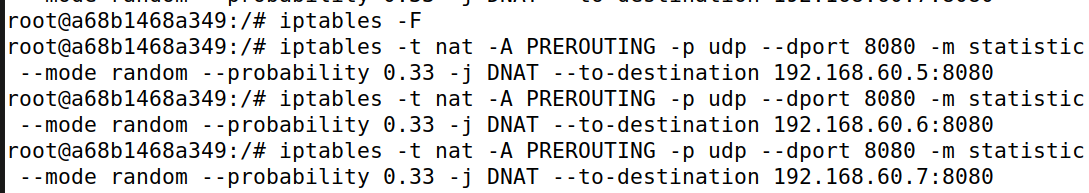
在服务器192.168.60.7上监听808端口，结果如下：



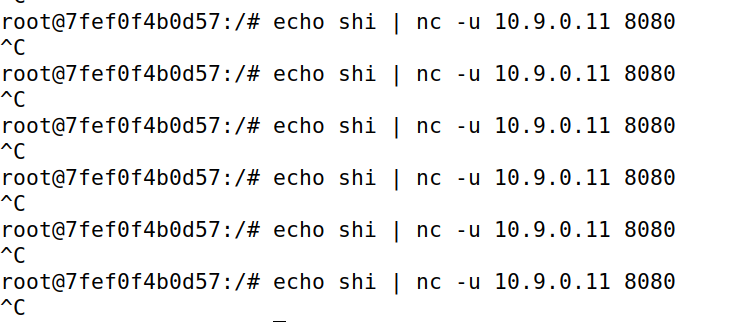
通过wireshark抓包可以发现，UDP报文发送成功，根据设置的规则按顺序发送至对应的服务器

Using the random mode

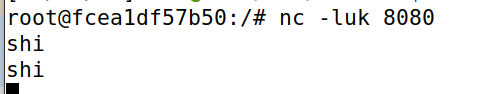
修改过滤规则如下：



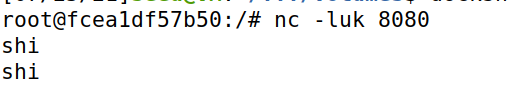
同上，再次发送数据包



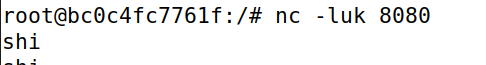
在服务器192.168.60.5上监听8080端口，结果如下



在服务器192.168.60.6上监听808端口，结果如下：



在服务器192.168.60.7上监听808端口，结果如下：



通过wireshark抓包可以发现，UDP报文发送成功，根据设置的规则随机发送至对应的服务器