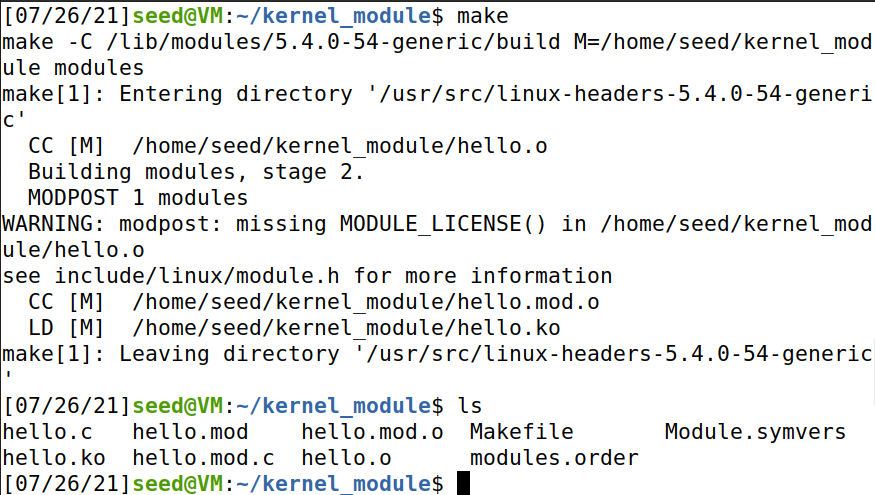
**Firewall Exploration Lab**

**57118113 蔡义涵**

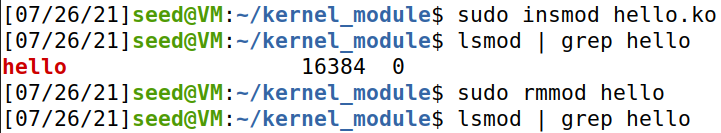
**Task 1: Implementing a Simple Firewall**

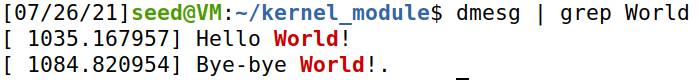
**Task 1.A: Implement a Simple Kernel Module**

将kernel\_module文件夹拷贝到/home目录下进行编译：



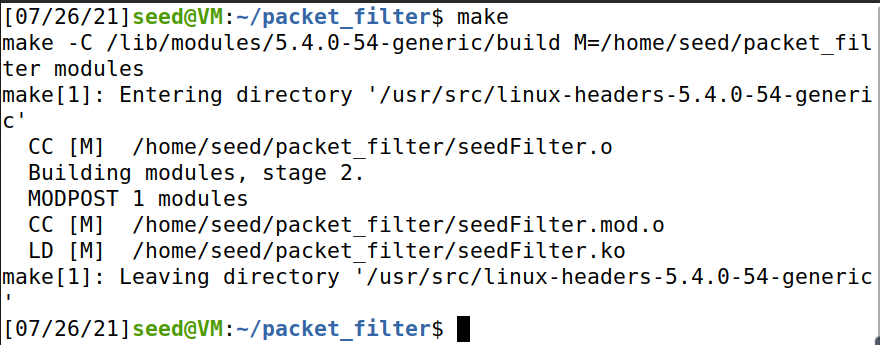
编译成功，接下来测试加载模块、删除模块、查看信息等命令：



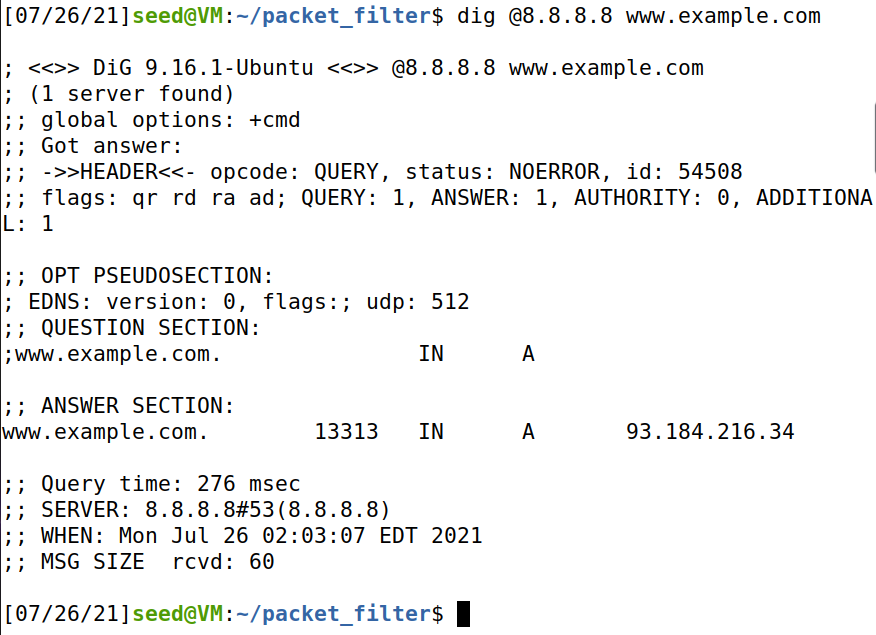


**Task 1.B: Implement a Simple Firewall Using Netfilter**

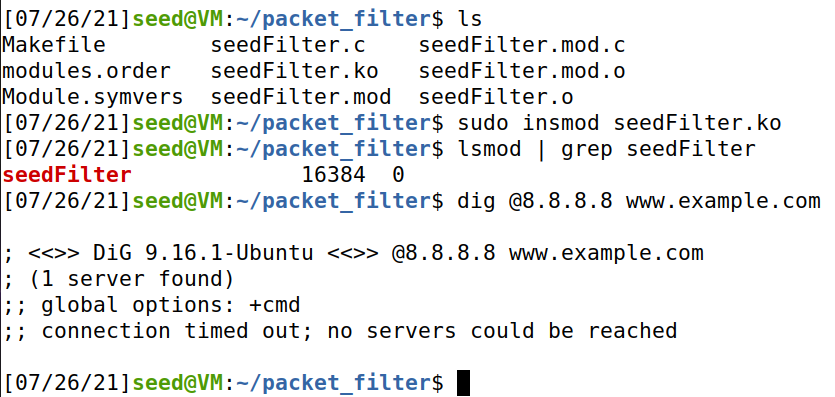
1. 将packet\_filter拷贝到/home目录下进行编译：



先测试dig @8.8.8.8 www.example.com命令：



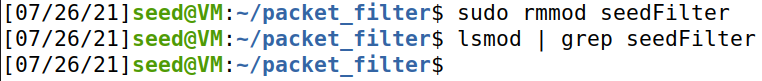
加载内核，再次执行dig @8.8.8.8 www.example.com命令：



可以看到，防火墙生效，udp报文被拦截。

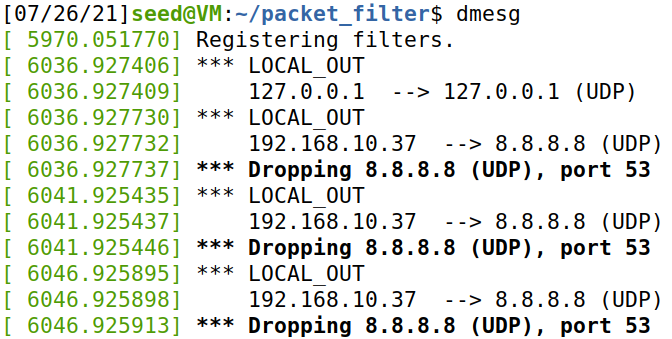
1. 在实验中，每次修改完代码都要重新编译，并使用sudo insmod seedFilter.ko命令加载到内核，在VM上执行dig命令，使用sudo dmesg -c来查看报文消息，实验完成后移除加载的内核。

首先移除Task1.A中加载的内核：



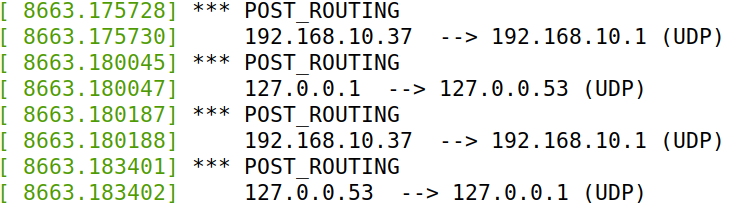
**NF\_INET\_LOCAL\_OUT**

NF\_INET\_LOCAL\_OUT表示数据包离开host之前的hook，原示例代码即调用了这个hook。



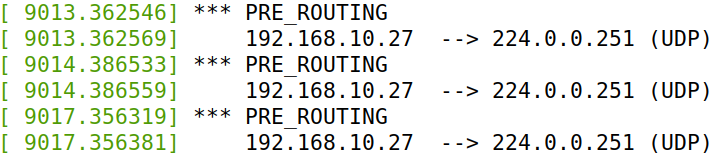
**NF\_INET\_POST\_ROUTING**

NF\_INET\_POST\_ROUTING表示数据包离开host进入其他网络之后的hook。



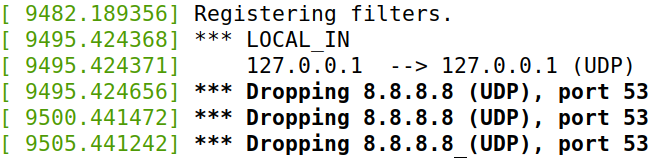
**NF\_IP\_PRE\_ROUTING**

NF\_IP\_PRE\_ROUTING表示做路由决策之前的hook。



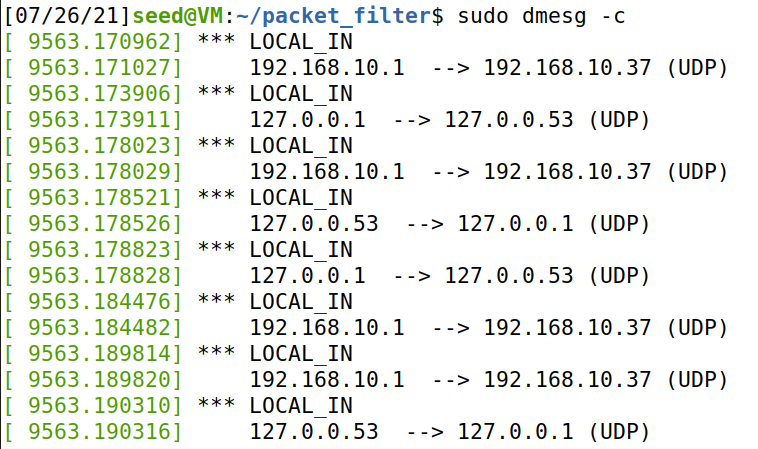
**NF\_IP\_LOCAL\_IN**

NF\_IP\_LOCAL\_IN表示数据包被发送到网络堆栈之前。



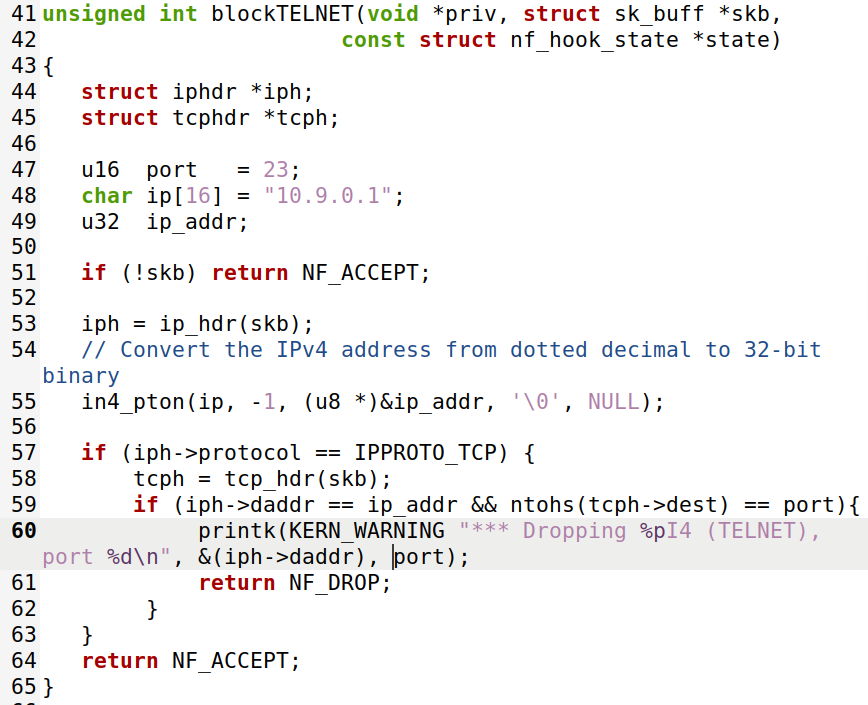
**NF\_IP\_FORWARD**

NF\_IP\_FORWARD表示转发报文到其他host的hook。

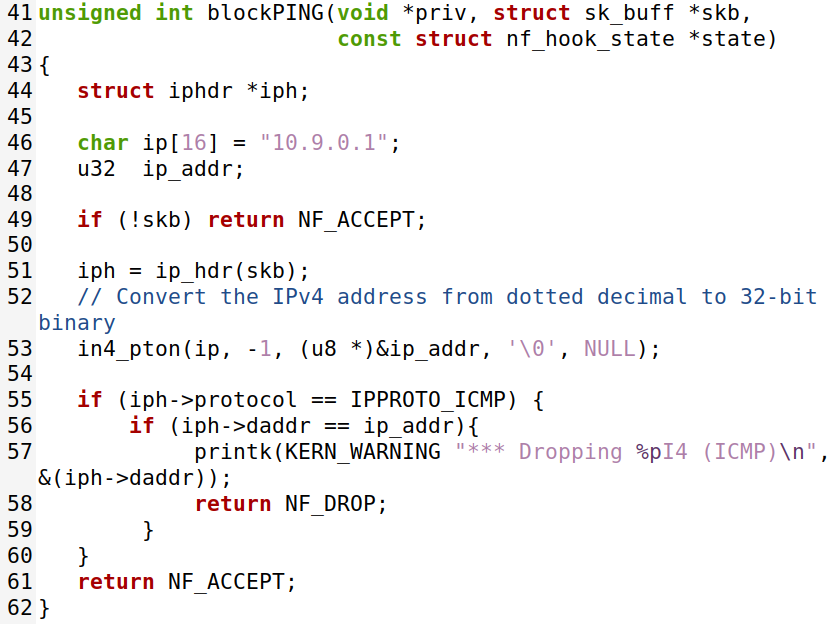


1. 首先开启docker。

编写函数blockTELNET：

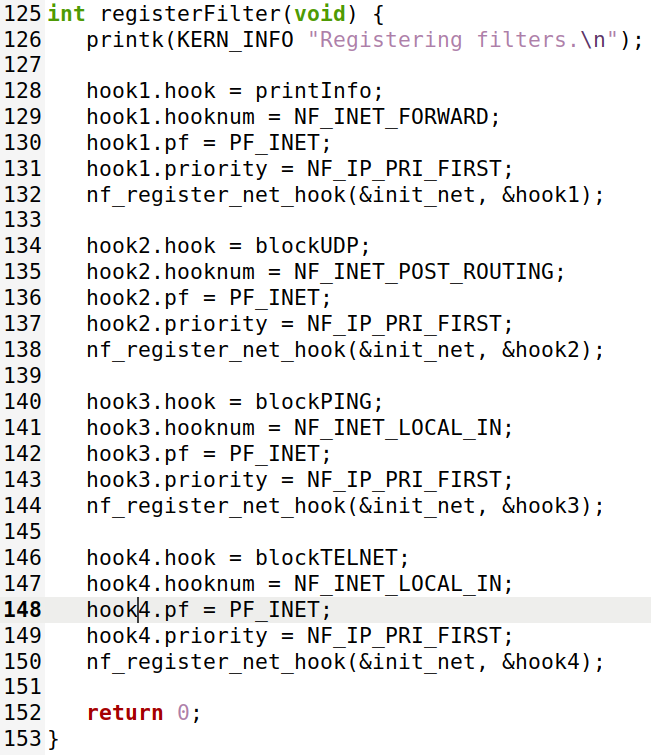


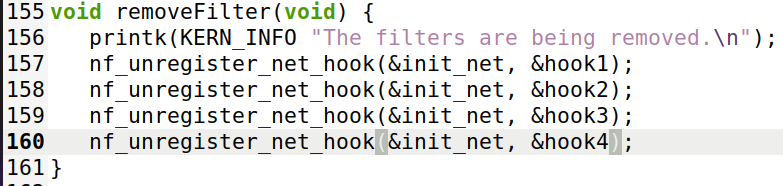
编写函数blockPING：



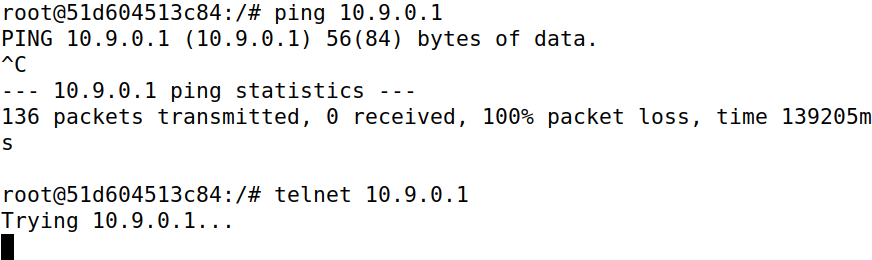
修改hook相关代码：



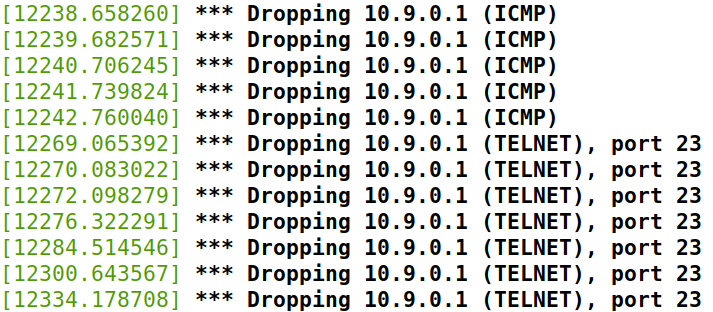




编译代码后加载内核，在10.9.0.5上分别执行ping和telnet命令。



都不能成功，查看日志。

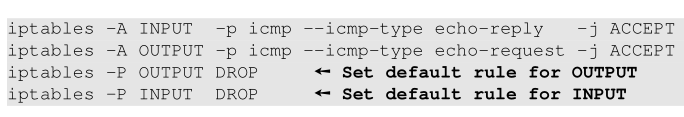


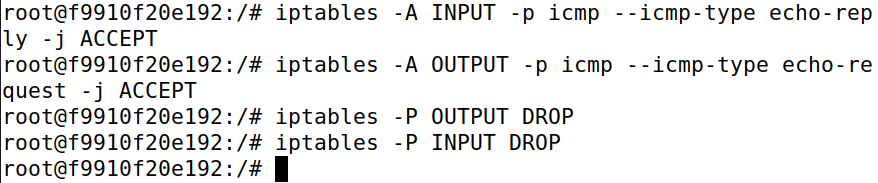
然后移除加载的内核。

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

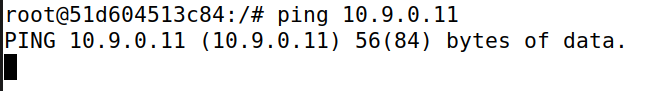
**Task 2.A: Protecting the Router**

在10.9.0.11执行以下命令：

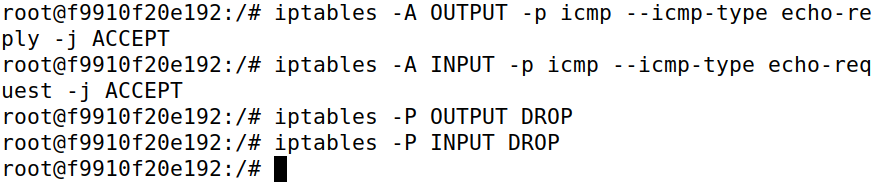


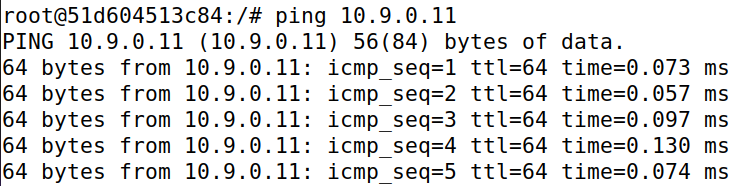


在10.9.0.5上ping 10.9.0.11：

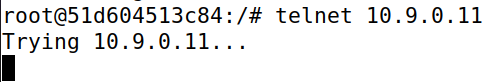


发现不成功，是因为提供的命令中前两条所在的chain反了。四条命令的含义是：仅允许通过input的echo-reply报文或output的echo-request报文。执行正确的命令后再次尝试：



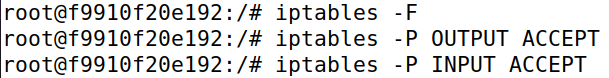


若尝试telnet命令：



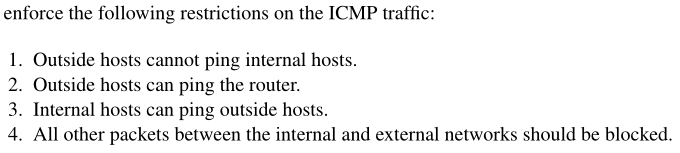
发现失败，是因为telnet并不是icmp报文。

执行以下命令将filter表恢复：

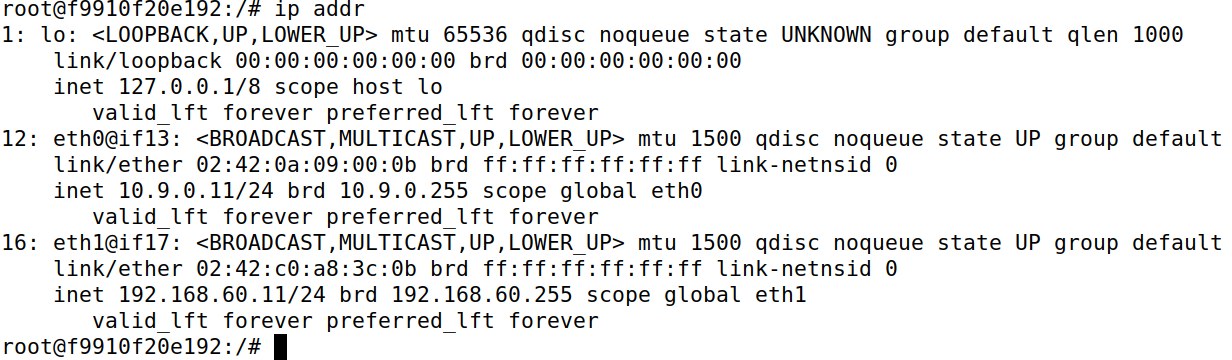


**Task 2.B: Protecting the Internal Network**

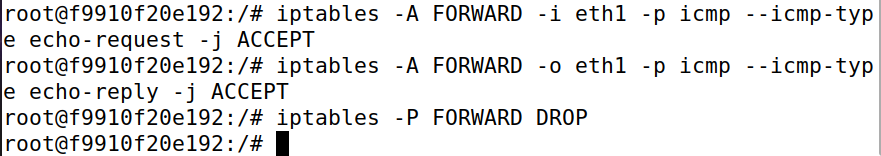
要实现的要求是：



查看router的接口信息：



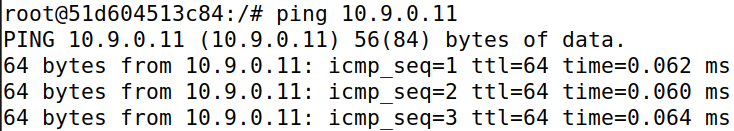
在router上执行以下命令：

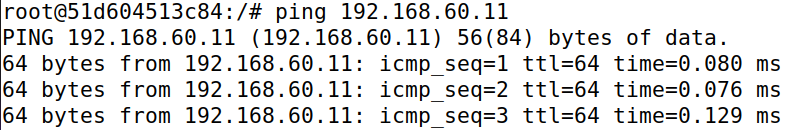


**使用10.9.0.5（外） ping 192.168.60.5（内）失败**

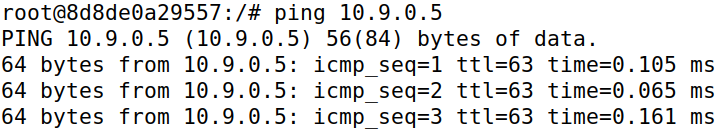


**使用10.9.0.5（外） ping router成功**





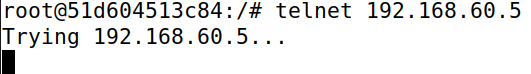
**使用192.168.60.5 ping 10.9.0.5成功**



**使用192.168.60.5 telnet 10.9.0.5失败**



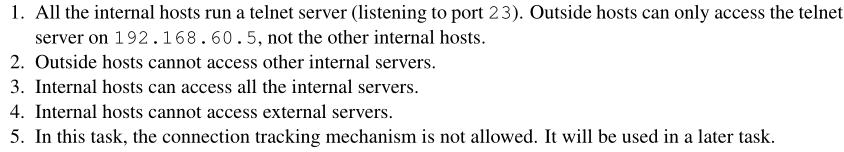
**使用10.9.0.5 telnet 192.168.60.5失败**



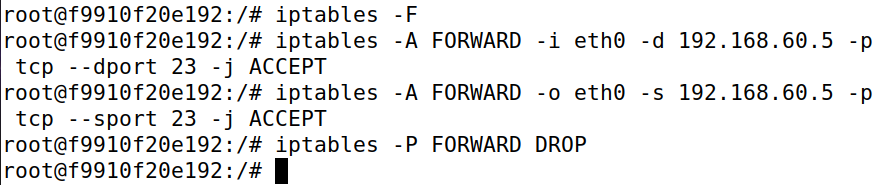
**总结**：实验结果满足规定的四个要求。

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

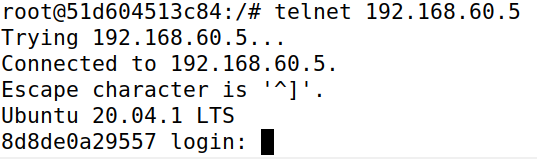
要实现的要求是：



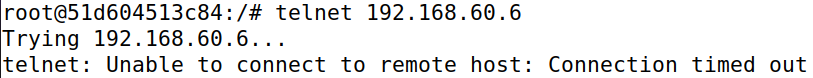
在router上执行以下命令：



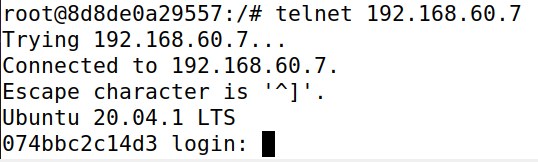
**使用10.9.0.5 telnet 192.168.60.5成功**



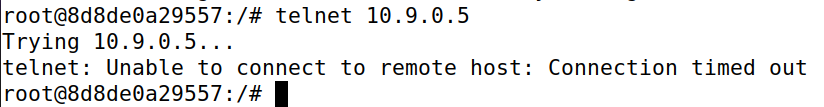
**使用10.9.0.5 telnet 192.168.60.6失败**



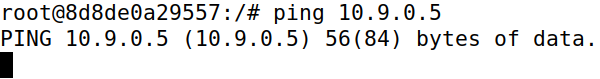
**使用192.168.60.5 telnet 192.68.60.7成功**



**使用192.168.60.5 telnet 10.9.0.5失败**



**使用192.168.60.5 ping 10.9.0.5失败**



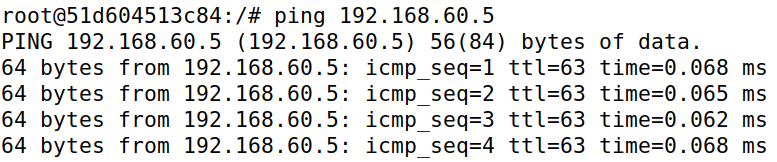
**总结**：实验结果满足规定的五个要求。

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

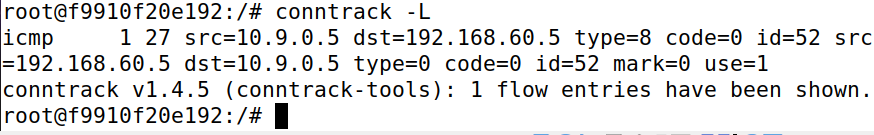
**Task 3.A: Experiment with the Connection Tracking**

**ICMP experiment**

在10.9.0.5上，ping 192.168.60.5



查看router上的连接跟踪信息：



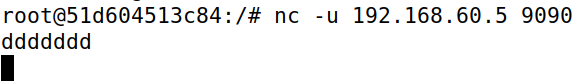
Ping结束时，发现第三个字段的值基本以秒为单位，递减至0时连接消失，猜测ICMP连接持续大约30s。

**UDP experiment**

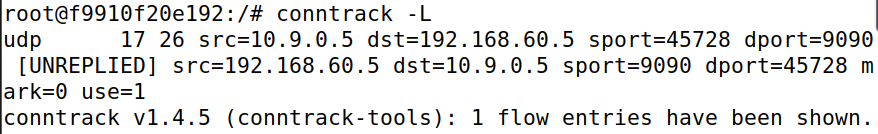
在192.168.60.5上，开启netcat UDP服务：



在10.9.0.5上，发送UDP报文：



查看router上的连接跟踪信息：



跟踪信息第三个字段的值基本以秒为单位，递减至0时消失，猜测连接时间也为大概30s。

**TCP experiment**

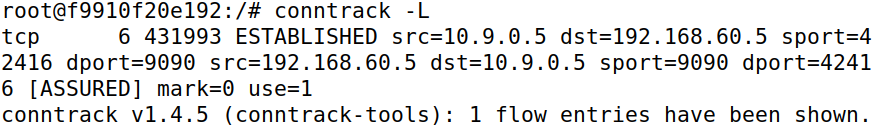
在192.168.60.5上，开启netcat TCP服务：



在10.9.0.5上，发送TCP报文：

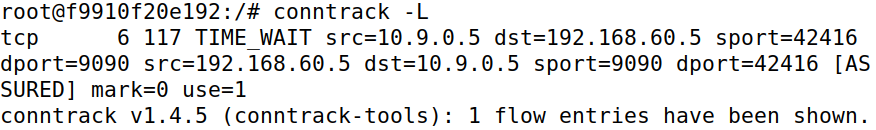


查看router上的连接跟踪信息：



Router上保持连接的时间大概是432000s=5day。

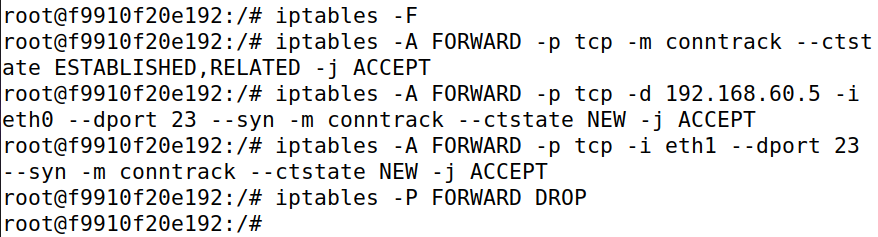
当TCP连接使用Ctrl+C断开连接时，查看router上的连接跟踪信息：



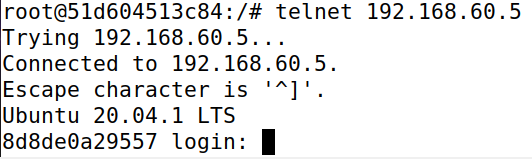
发现连接状态发生改变，变成time\_wait，大约持续120s。

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

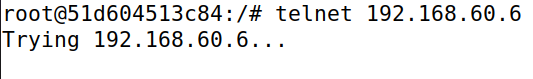
在router上执行以下命令：



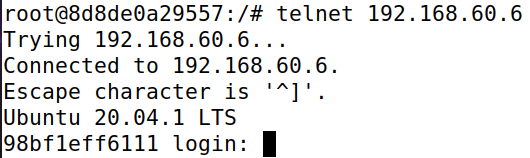
**使用10.9.0.5 telnet 192.168.60.5成功**



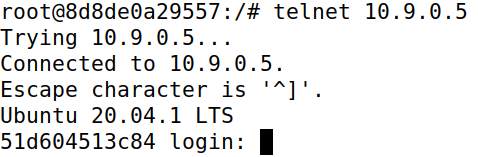
**使用10.9.0.5 telnet 192.168.60.6失败**



**使用192.168.60.5 telnet 192.168.60.6成功**



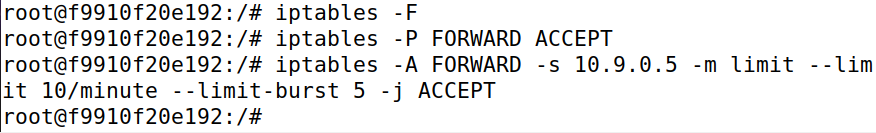
**使用192.168.60.5 telnet 10.9.0.5成功**



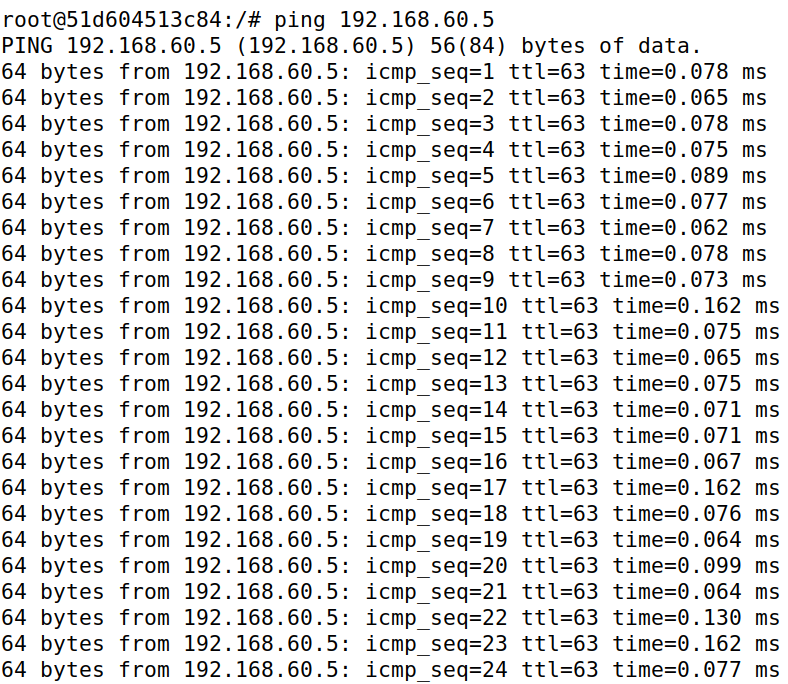
**总结**：以上测试满足了提出的要求。有状态的防火墙肯定会更加精准，书写规则也会更加方便，但牺牲了一定的存储空间和包过滤的性能。

**Task 4: Limiting Network Traffic**

在router上执行下面的命令：



在10.9.0.5上ping192.168.60.5，查看结果：

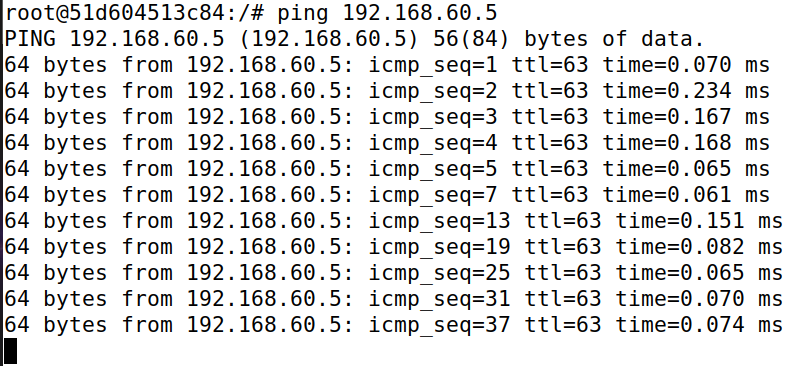


ICMP报文并没有被限制。

增加以下命令：



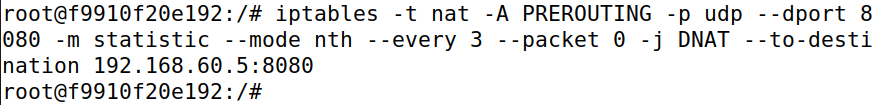
再次尝试ping命令：



发现几次ping之后，开始出现丢包的情况，受到了限制。

**Task 5: Load Balancing**

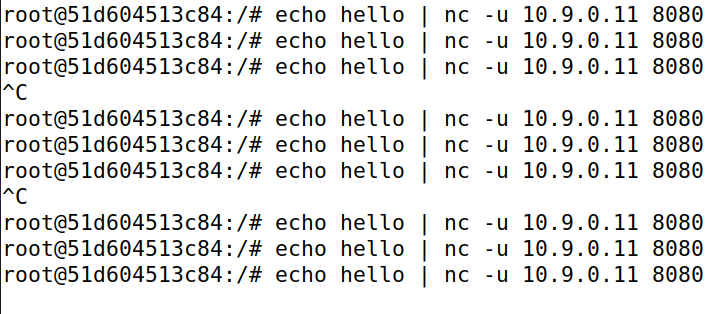
首先清除上个实验的设置，然后在router上运行以下命令：



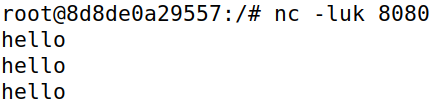
在192.168.60.5上执行以下命令：



在10.9.0.5上执行以下命令：

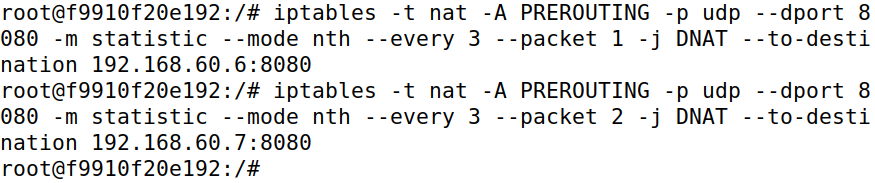


一共执行了九次，查看192.168.60.5上的结果：

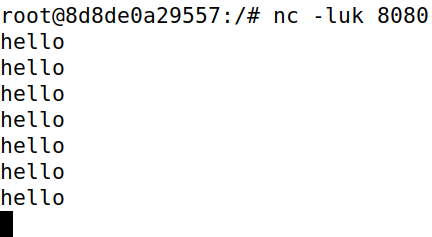


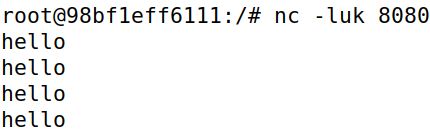
可以看到每3个报文就有一个发送给192.168.60.5。

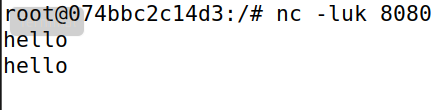
再在router上增加以下命令，使得报文能够负载均衡：



在192.168.60.5/6/7上开启服务，重复执行发送报文过程：

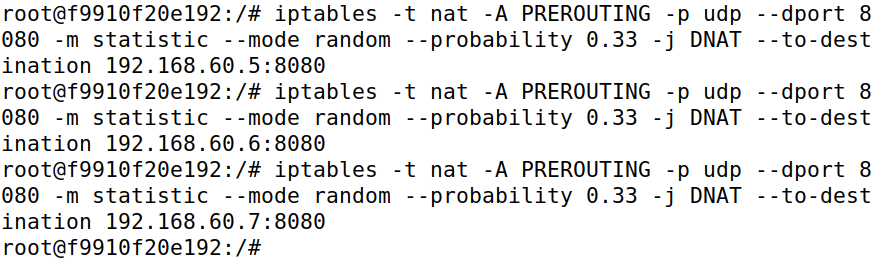




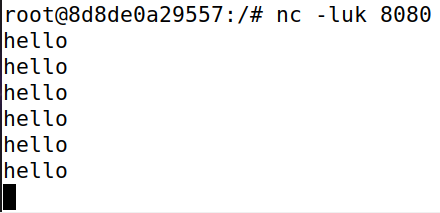


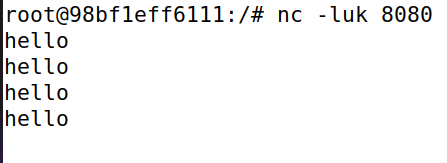
可以看到三个server上均收到报文。

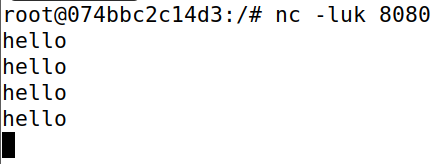
在router上运行以下命令：



再次实验，得到结果：







可以看出来基本处于负载均衡的状态。