LAB6: Firewall Exploration Lab

学号： 57118227　　**姓名：** 孙浩　　**日期：** 2021年7月22日

Container names & IDs

本实验报告在展示结果时用到了较多的命令行终端的窗口截图，为了区分是在哪个Dokcer容器上运行的结果，在这里对这些容器的名字和ID进行一个说明。

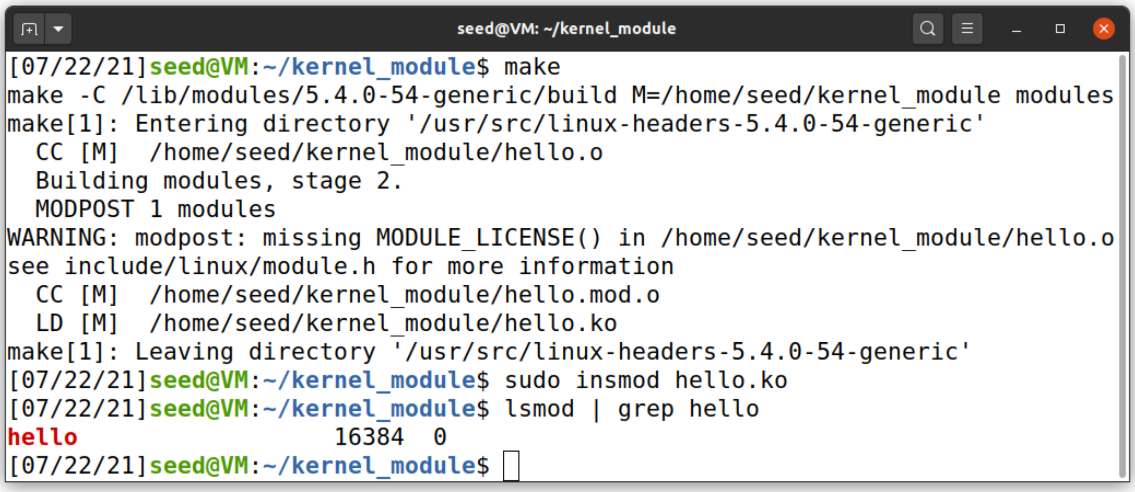
NAME	ID
router	7aa75ec3d416
10.9.0.5	738b7df2d917
192.168.60.5	3c3940eea097
192.168.60.6	fae13d34d8dd
192.168.60.7	967fef38d8d3

Task 1: Implementing a Simple Firewall

Task 1.A: Implement a Simple Kernel Module

> *Please compile this simple kernel module on your VM, and run it on the VM. For this task, we will not use containers. Please show your running results in the lab report.*

将*kernel_module*这个文件夹移到一个没有空格的目录下，然后直接***make***编译，可以看到编译成功，然后下一步通过***insmod hello.ko***将其插入，可以这个模块已经成功进入了内核。



```
[07/22/21]seed@VM:~/kernel_module$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/kernel_module modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/kernel_module/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/seed/kernel_module/hello.o
see include/linux/module.h for more information
  CC [M]  /home/seed/kernel_module/hello.mod.o
  LD [M]  /home/seed/kernel_module/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[07/22/21]seed@VM:~/kernel_module$ sudo insmod hello.ko
[07/22/21]seed@VM:~/kernel_module$ lsmod | grep hello
hello                  16384  0
[07/22/21]seed@VM:~/kernel_module$ 
```

再将其移除，恢复原始环境，通过***dmesg***可查看日志看见其输出。

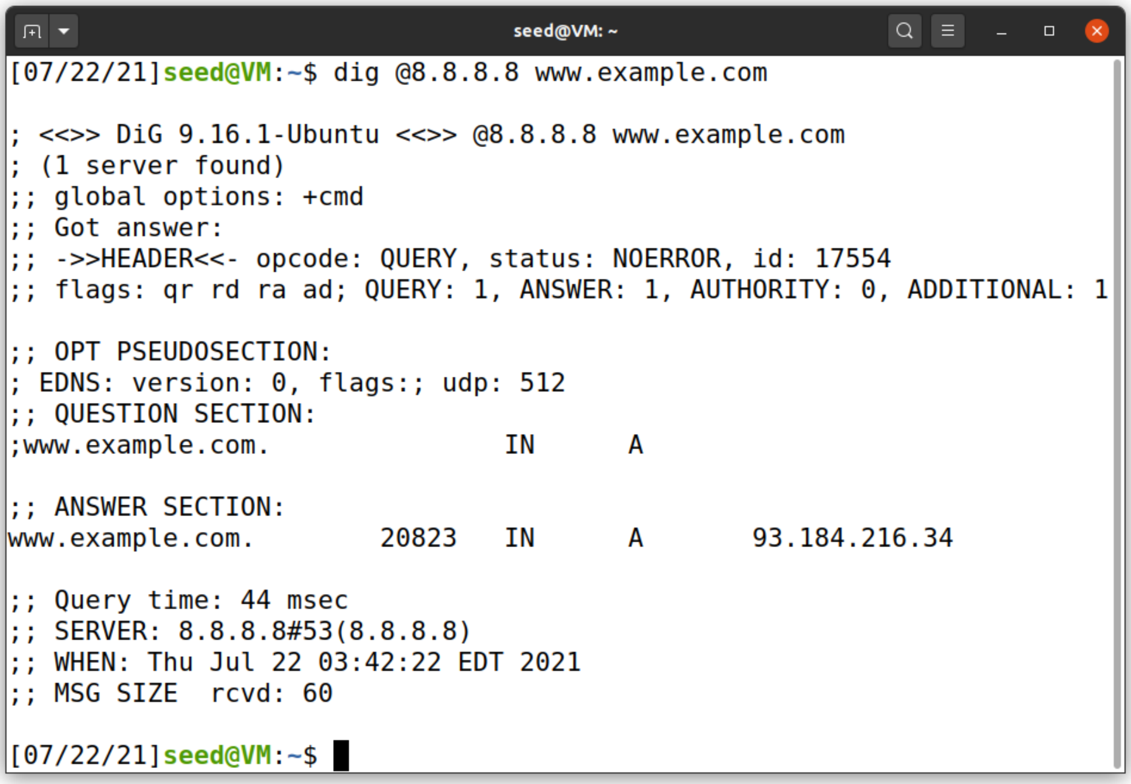
```
[ 3154.379783] Hello World!
[ 3568.236052] Bye-bye World!.
```

要切记重新编译插入 module 前要把旧的 rmmod 掉，否则原来钩子上的函数不会消失，会导致预期外的结果

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

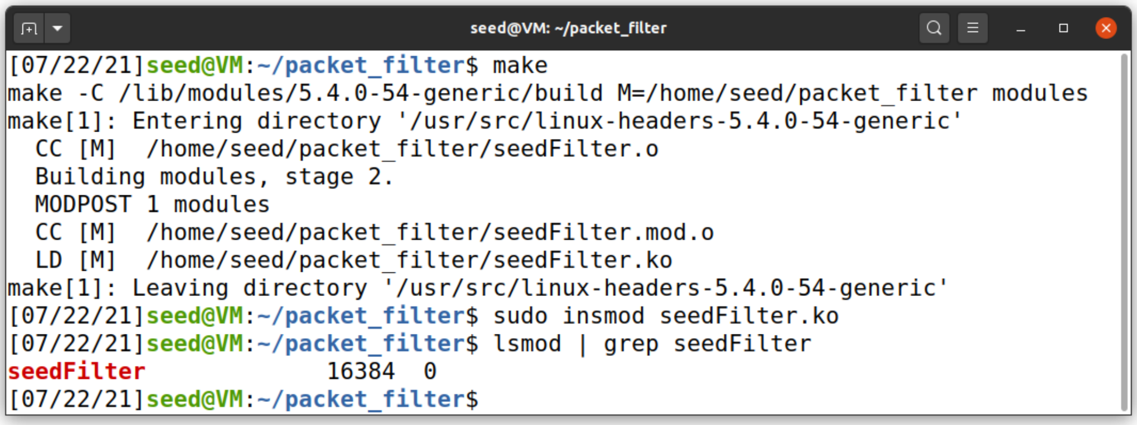
> *Compile the sample code using the provided Makefile. Load it into the kernel, and demonstrate that the firewall is working as expected. You can use the following command to generate UDP packets to 8.8.8.8, which is Google's DNS server. If your firewall works, your request will be blocked; otherwise, you will get a response.*
>
> ```
> dig @8.8.8.8 www.example.com
> ```

首先在什么也不做的情况下运行该命令，查看结果。



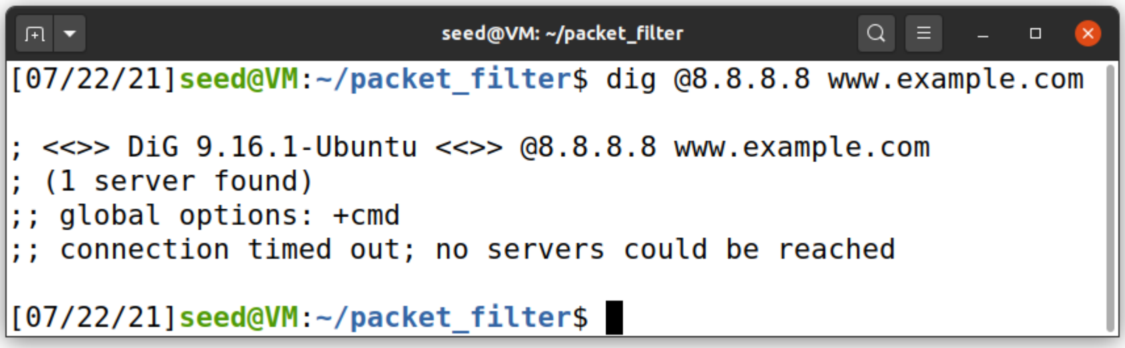
```
[07/22/21]seed@VM:~$ dig @8.8.8.8 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17554
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        20823   IN      A       93.184.216.34

;; Query time: 44 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Jul 22 03:42:22 EDT 2021
;; MSG SIZE  rcvd: 60

[07/22/21]seed@VM:~$ 
```

然后将示例代码运行编译进内核，运行结果如下。



```
[07/22/21]seed@VM:~/packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/packet_filter/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/packet_filter/seedFilter.mod.o
  LD [M]  /home/seed/packet_filter/seedFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[07/22/21]seed@VM:~/packet_filter$ sudo insmod seedFilter.ko
[07/22/21]seed@VM:~/packet_filter$ lsmod | grep seedFilter
seedFilter             16384  0
[07/22/21]seed@VM:~/packet_filter$
```

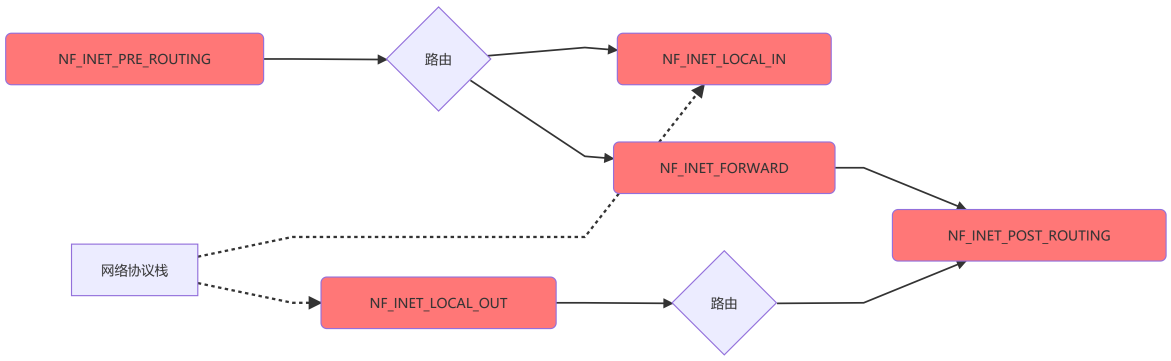
再次进行***dig***，发现代码生效。



```
[07/22/21]seed@VM:~/packet_filter$ dig @8.8.8.8 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached

[07/22/21]seed@VM:~/packet_filter$
```

> *Hook the printInfo function to all of the netfilter hooks. Here are the macros of the hook numbers. Using your experiment results to help explain at what condition will each of the hook function be invoked.*
>
> ```
> NF_INET_PRE_ROUTING
> NF_INET_LOCAL_IN
> NF_INET_FORWARD
> NF_INET_LOCAL_OUT
> NF_INET_POST_ROUTING
> ```

五种钩子函数的关系图如下



- LOCAL_OUT&&POST_ROUTING

LOCAL_OUT本机产生的数据包到达的第一个钩子点，而POST_ROUTING是需要被转发或者由本机产生的数据包都会经过的一个钩子。例如原示例代码所示，即为这两个钩子被调用了，DROP是在POST_ROUTING这个钩子点调用的。

```
[ 4771.880065] *** LOCAL_OUT
[ 4771.880065]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 4771.880068] *** Dropping 8.8.8.8 (UDP), port 53
[ 4776.886303] *** LOCAL_OUT
[ 4776.886305]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 4776.886313] *** Dropping 8.8.8.8 (UDP), port 53
[ 4781.999736] *** LOCAL_OUT
[ 4781.999737]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 4781.999747] *** Dropping 8.8.8.8 (UDP), port 53
```

- PRE_ROUTING

除了混杂模式，所有数据包都将经过这个钩子点。它上面注册的钩子函数在路由判决之前被调用。

```
[ 8345.421980]     112.80.248.75  --> 10.0.2.15 (ICMP)
[ 8345.422332] *** PRE_ROUTING
[ 8345.422335]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 8345.425224] *** PRE_ROUTING
[ 8345.425228]     10.80.128.28  --> 10.0.2.15 (UDP)
[ 8345.427106] *** PRE_ROUTING
[ 8345.427109]     10.80.128.28  --> 10.0.2.15 (UDP)
[ 8345.427488] *** PRE_ROUTING
```

- LOCAL_IN

数据包要进行路由判决，以决定需要被转发还是发往本机，前一种情况下，数据包将前往转发路径；而后一种情况下，数据包将通过这个钩子点，之后被发送到网络协议栈，并最终被主机接收。

```
[ 8751.385140] *** LOCAL_IN
[ 8751.385145]     35.232.111.17  --> 10.0.2.15 (TCP)
[ 8751.385696] *** LOCAL_IN
[ 8751.385698]     35.232.111.17  --> 10.0.2.15 (TCP)
[ 8751.649998] *** LOCAL_IN
[ 8751.650047]     35.232.111.17  --> 10.0.2.15 (TCP)
[ 8751.650137] *** LOCAL_IN
[ 8751.650152]     35.232.111.17  --> 10.0.2.15 (TCP)
[ 8751.651494] *** LOCAL_IN
[ 8751.651499]     35.232.111.17  --> 10.0.2.15 (TCP)
```

- FORWARD

需要被转发的数据包会到达这个钩子点，这个钩子点对于防火墙来说是十分重要的。

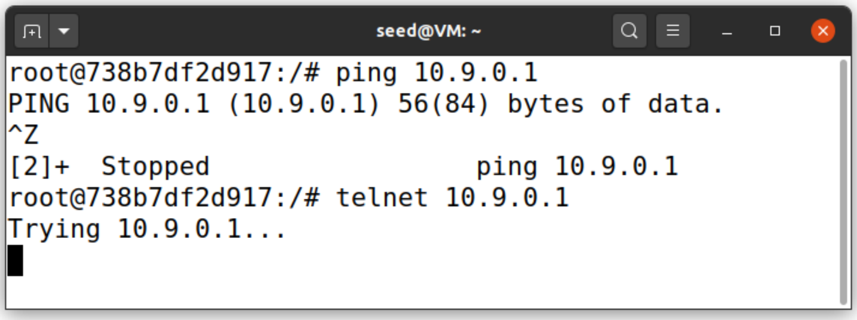
```
[ 9208.982523] *** FORWARD
[ 9208.982526]     192.168.60.5  --> 10.9.0.6 (ICMP)
[ 9211.043983] *** FORWARD
[ 9211.043987]     192.168.60.11  --> 192.168.60.5 (ICMP)
[ 9211.043998] *** FORWARD
[ 9211.044000]     192.168.60.11  --> 192.168.60.5 (ICMP)
[ 9211.116597] *** FORWARD
[ 9211.116600]     192.168.60.5  --> 10.9.0.1 (ICMP)
```

> *Implement two more hooks to achieve the following: (1) preventing other computers to ping the VM, and (2) preventing other computers to telnet into the VM. Please implement two different hook functions, but register them to the same netfilter hook. You should decide what hook to use. Telnet's default port is TCP port 23. To test it, you can start the containers, go to 10.9.0.5, run the following commands (10.9.0.1 is the IP address assigned to the VM; for the sake of simplicity, you can hardcode this IP address in your firewall rules):*
>
> ```
> ping 10.9.0.1
> telnet 10.9.0.1
> ```

根据题目要求，分别设计两个钩子函数，一个为***blockTELNET***，判断是否为TCP协议，并且目的端口为23，另一个为***blockPING***，判断是否为ICMP协议，两个函数都挂在NF_INET_LOCAL_IN这个钩子下即可。代码如下。

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>
#include <linux/udp.h>
#include <linux/if_ether.h>
#include <linux/inet.h>

static struct nf_hook_ops hook1, hook2;

unsigned int blockTELNET(void *priv, struct sk_buff *skb,
                       const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct tcphdr *tcph;

   u16  port   = 23;
   char ip[16] = "10.9.0.1";
   u32  ip_addr;

   if (!skb) return NF_ACCEPT;

   iph = ip_hdr(skb);
   // Convert the IPv4 address from dotted decimal to 32-bit binary
   in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);

   if (iph->protocol == IPPROTO_TCP) {
       tcph = tcp_hdr(skb);
       if (iph->daddr == ip_addr && ntohs(tcph->dest) == port){
            printk(KERN_WARNING "*** Dropping %pI4 (TELNET)\n", &(iph->saddr));
            return NF_DROP;
        }
   }
   return NF_ACCEPT;
}

unsigned int blockPING(void *priv, struct sk_buff *skb,
                 const struct nf_hook_state *state)
{
   struct iphdr *iph;

   char ip[16] = "10.9.0.1";
   u32  ip_addr;

   if (!skb) return NF_ACCEPT;

   iph = ip_hdr(skb);
   // Convert the IPv4 address from dotted decimal to 32-bit binary
   in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);

   if (iph->protocol == IPPROTO_ICMP) {
       if (iph->daddr == ip_addr){
            printk(KERN_WARNING "*** Dropping %pI4 (PING)\n", &(iph->saddr));
            return NF_DROP;
        }
   }
   return NF_ACCEPT;
}

int registerFilter(void) {
   printk(KERN_INFO "Registering filters.\n");

   hook1.hook = blockTELNET;
   hook1.hooknum = NF_INET_LOCAL_IN;
   hook1.pf = PF_INET;
   hook1.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook1);

   hook2.hook = blockPING;
   hook2.hooknum = NF_INET_LOCAL_IN;
   hook2.pf = PF_INET;
   hook2.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook2);
   return 0;
}

void removeFilter(void) {
   printk(KERN_INFO "The filters are being removed.\n");
   nf_unregister_net_hook(&init_net, &hook1);
   nf_unregister_net_hook(&init_net, &hook2);
}

module_init(registerFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

在10.9.0.5上进行telnet和ping测试，都不能成功进行。



```
root@738b7df2d917:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
^Z
[2]+  Stopped                 ping 10.9.0.1
root@738b7df2d917:/# telnet 10.9.0.1
Trying 10.9.0.1...
```

接下来查看日志，可以看到相关的DROP信息。

```
[13881.245081] *** Dropping 10.9.0.5 (PING)
[13882.247234] *** Dropping 10.9.0.5 (PING)
[13883.271319] *** Dropping 10.9.0.5 (PING)
[13890.592060] *** Dropping 10.9.0.5 (TELNET)
[13891.623427] *** Dropping 10.9.0.5 (TELNET)
[13893.638557] *** Dropping 10.9.0.5 (TELNET)
```

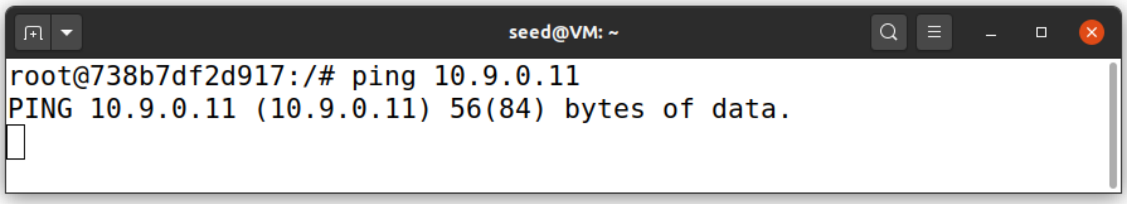
Task 2: Experimenting with Stateless Firewall Rules

Task 2.A: Protecting the Router

在10.9.0.11上执行下列命令，值得说明的是，这里没有指明Table，默认是在***filter***上。

```
iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -P OUTPUT DROP
iptables -P INPUT DROP
```

在10.9.0.5上进行ping 10.9.0.11，发现不成功，为什么呢?

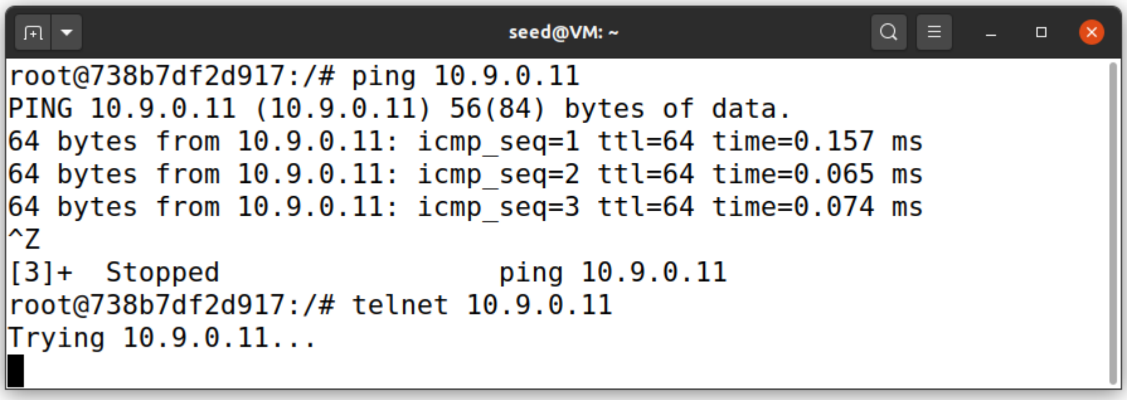


```
root@738b7df2d917:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
```

原因很简单，因为文档提供的命令是错的，应该将前两条命令所在的Chain进行一个对换，如下面所示，就可以成功了。

```
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -P OUTPUT DROP
iptables -P INPUT DROP
```

可以看到，使用正确命令后，可以PING成功，但是不能TELNET。



```
root@738b7df2d917:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.157 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.065 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.074 ms
^Z
[3]+  Stopped                 ping 10.9.0.11
root@738b7df2d917:/# telnet 10.9.0.11
Trying 10.9.0.11...
```

Task 2.B: Protecting the Internal Network

> 1. Outside hosts cannot ping internal hosts.
> 2. Outside hosts can ping the router.
> 3. Internel hosts can ping outside hosts.
> 4. All other packets between the internal and external networks should be blocked

还有一些需要记住的关键信息，比如router上的接口信息。

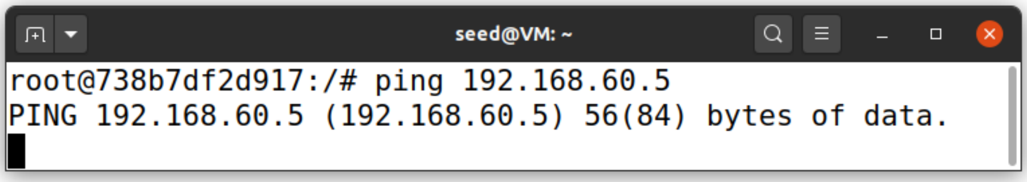
接口	IP	掩码
eth0	10.9.0.11	255.255.255.0
eth1	192.168.60.11	255.255.255.0

根据要求，在router上执行如下3条命令即可。

```
iptables -A FORWARD  -i eth1 -p icmp --icmp-type echo-request -j ACCEPT
iptables -A FORWARD -o eth1 -p icmp --icmp-type echo-reply -j ACCEPT
iptables -P FORWARD DROP
```

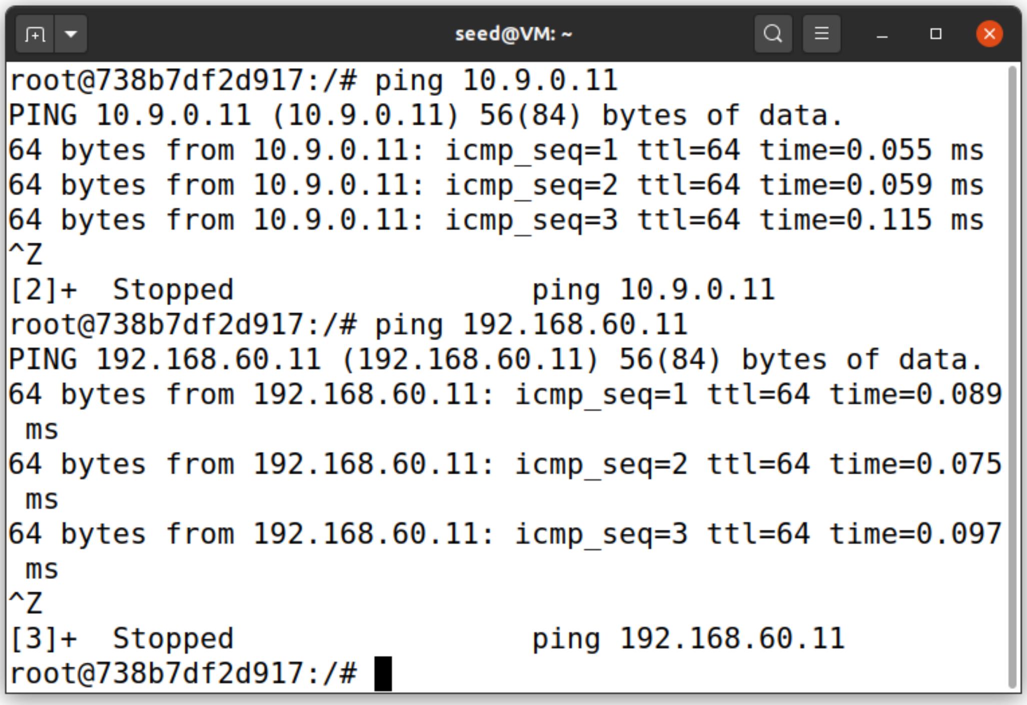
接下来进行测试验证。

- Outside hosts cannot ping internal hosts.



```
root@738b7df2d917:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
```

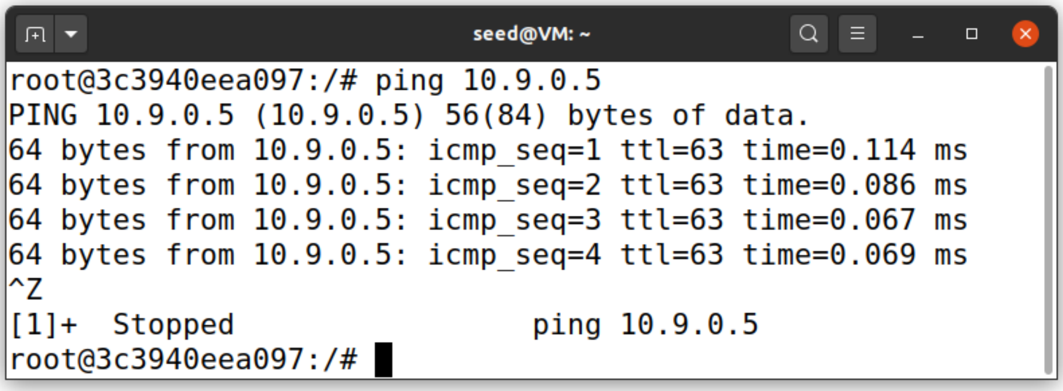
- Outside hosts can ping the router.



```
root@738b7df2d917:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.055 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.115 ms
^Z
[2]+  Stopped                 ping 10.9.0.11
root@738b7df2d917:/# ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.089
 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.075
 ms
64 bytes from 192.168.60.11: icmp_seq=3 ttl=64 time=0.097
 ms
^Z
[3]+  Stopped                 ping 192.168.60.11
root@738b7df2d917:/# 
```

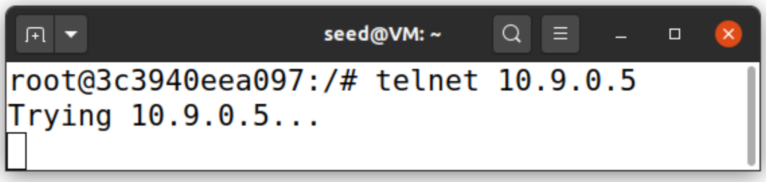
可以看到router的两个接口上的IP都可以PING通。

- Internel hosts can ping outside hosts.

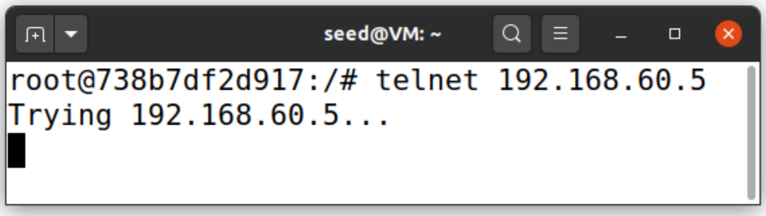


```
root@3c3940eea097:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.114 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.086 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.067 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.069 ms
^Z
[1]+  Stopped                 ping 10.9.0.5
root@3c3940eea097:/# 
```

- All other packets between the internal and external networks should be blocked



```
root@3c3940eea097:/# telnet 10.9.0.5
Trying 10.9.0.5...
```



```
root@738b7df2d917:/# telnet 192.168.60.5
Trying 192.168.60.5...
```

可以看见，里面连不上外面，外面也连不上里面。

Task 2.C: Protecting Internal Servers

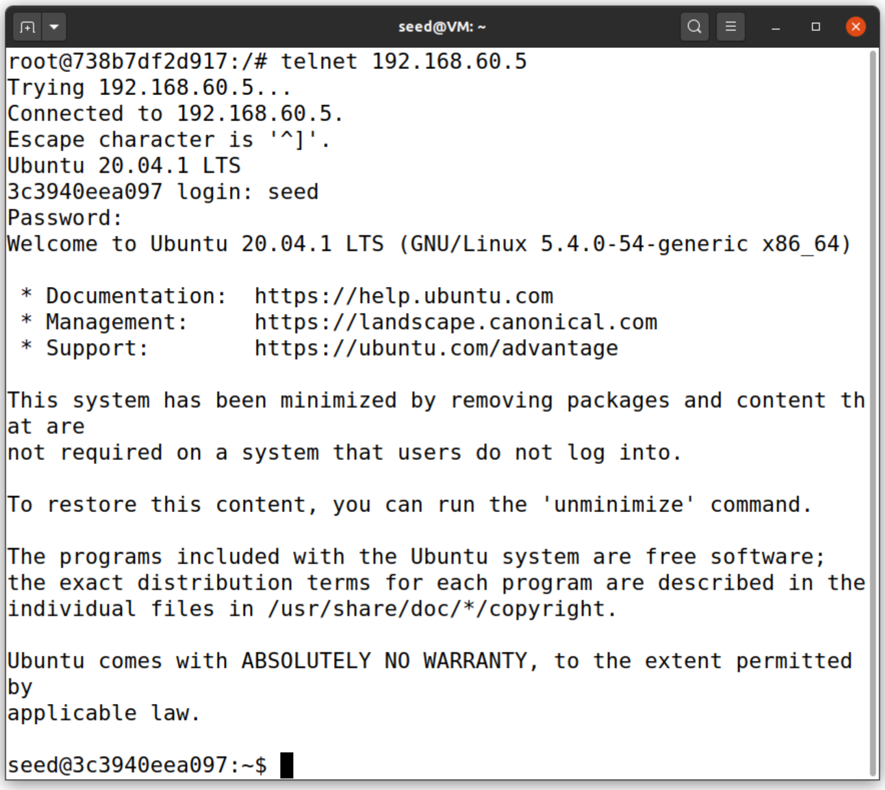
> 1. All the internal hosts run a telnet server(listening to port 23). Outside hosts can only access the telnet server 192.168.60.5, not the other internal hosts.
> 2. Outside hosts cannot access other internal hosts.
> 3. Internal hosts can access all the internal servers.
> 4. Internal hosts cannot access external servers.
> 5. In this task, the connection tracking mechanism is not allowed. It will be used in a later task.

根据要求，在router上运行如下命令即可。

```
iptables -A FORWARD -i eth0 -d 192.168.60.5 -p tcp --dport 23 -j ACCEPT
iptables -A FORWARD -o eth0 -s 192.168.60.5 -p tcp --sport 23 -j ACCEPT
iptables -P FORWARD DROP
```

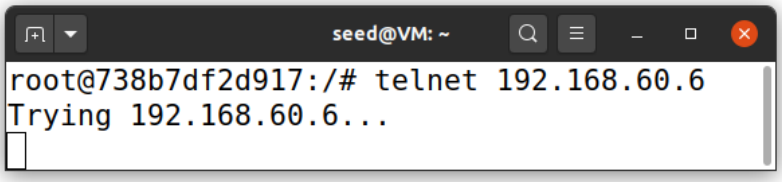
结果如下所示。

- Outside hosts can only access the telnet server 192.168.60.5



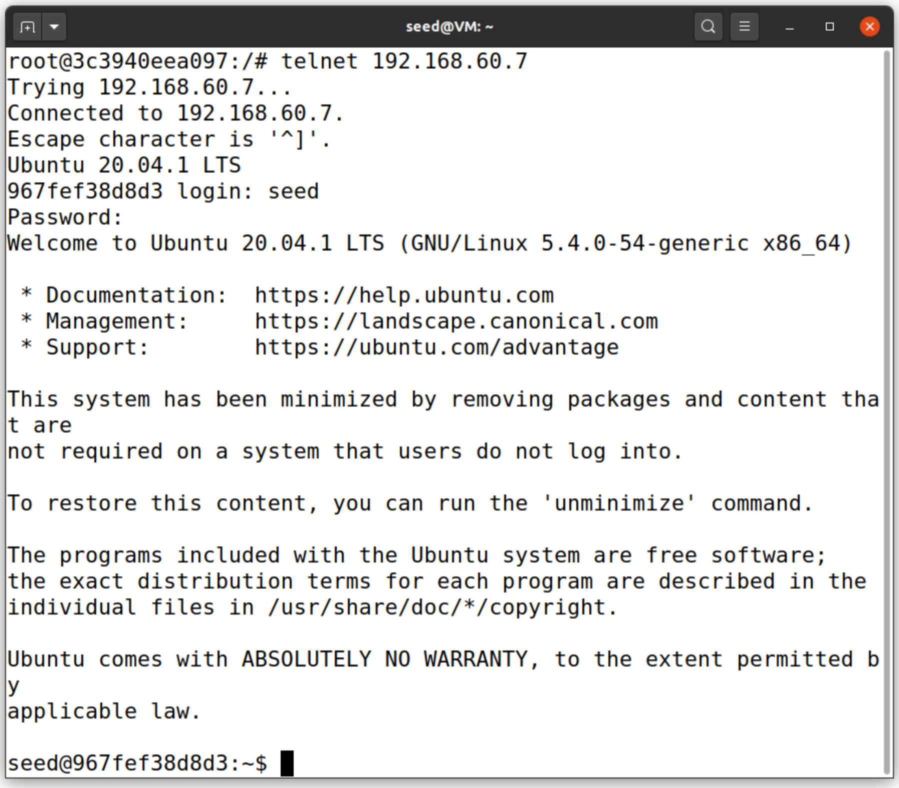
```
root@738b7df2d917:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
3c3940eea097 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content th
at are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted
by
applicable law.

seed@3c3940eea097:~$ 
```

- Outside hosts cannot access other internal hosts.



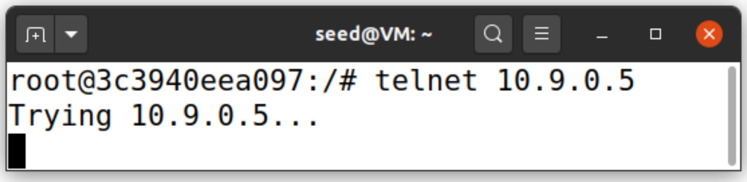
```
root@738b7df2d917:/# telnet 192.168.60.6
Trying 192.168.60.6...
```

- Internal hosts can access all the internal servers.



```
root@3c3940eea097:/# telnet 192.168.60.7
Trying 192.168.60.7...
Connected to 192.168.60.7.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
967fef38d8d3 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content tha
t are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted b
y
applicable law.

seed@967fef38d8d3:~$ 
```

- Internal hosts cannot access external servers.



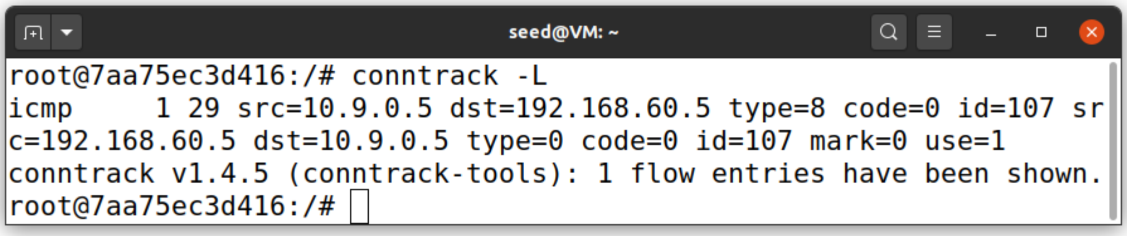
```
root@3c3940eea097:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

Task 3: Connection Tracking and Stateful Firewall

Task 3.A: Experiment with the Connection Tracking

- ICMP experiment: Run the following command and check the connection tracking information on the router. Describe your obserbation. How long is the ICMP connection state be kept?

在10.9.0.5上来ping 192.168.60.5，在router上进行conntrack -L，发现第三个字段的值基本以秒为单位递减，递减至0时，连接消失，猜测是连接状态的计时器。也就是说ICMP连接持续时间为30s。

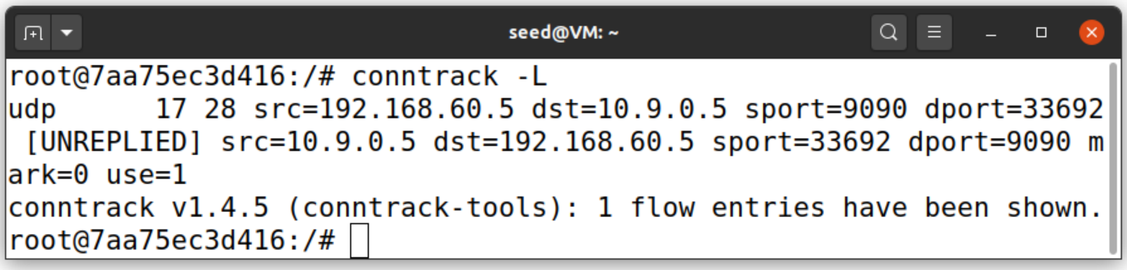


```
root@7aa75ec3d416:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=107 sr
c=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=107 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@7aa75ec3d416:/# 
```

- UDP experiment: Run the following command and check the connection tracking information on the router. Describe your observation. How long is the UDP connection state be kept.

```
// On 192.168.60.5, start a netcat UDP server
# nc -lu 9090

// On 10.9.0.5, send out UDP packets
# nc -u 192.168.60.5 9090
<type something, then hit return>
```

实验步骤基本同上，UDP的连接时间也为30s.

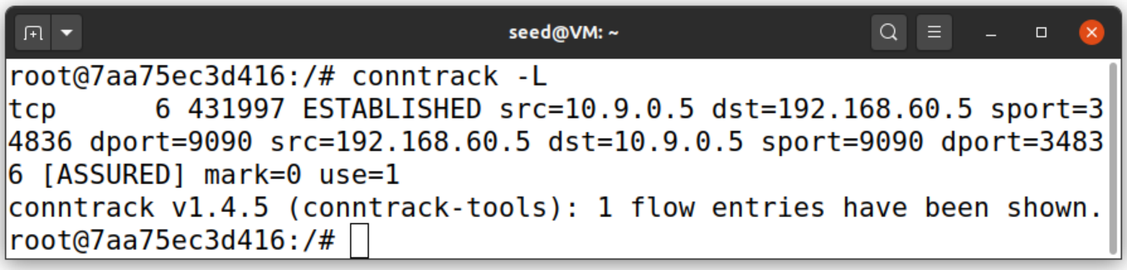


```
root@7aa75ec3d416:/# conntrack -L
udp      17 28 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=33692
 [UNREPLIED] src=10.9.0.5 dst=192.168.60.5 sport=33692 dport=9090 m
ark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@7aa75ec3d416:/# 
```

- TCP experiment: Run the following command and check the connection tracking information on the router. Describe your observation. How long is the TCP connecion state be kept?

```
// On 192.168.60.5, start a netcat TCP server
# nc -l 9090

// On 10.9.0.5, send out TCP packets
# nc 192.168.60.5 9090
<type something, then hit return>
```

可以看到处于TCP ESTABLISHED连接状态时，router上保持连接的时间为432000s = 7200min = 120h = 5day



```
root@7aa75ec3d416:/# conntrack -L
tcp      6 431997 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=34836 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=34836 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@7aa75ec3d416:/#
```

当TCP连接断开时，处于TIME_WAIT状态（可以在任意一方 `Ctrl` + `C` 来实现，而使用 `Ctrl` + `Z` 是不会中断TCP连接的），这时连接时间会变为120s。

Task 3.B: Setting Up a Stateful Firewall

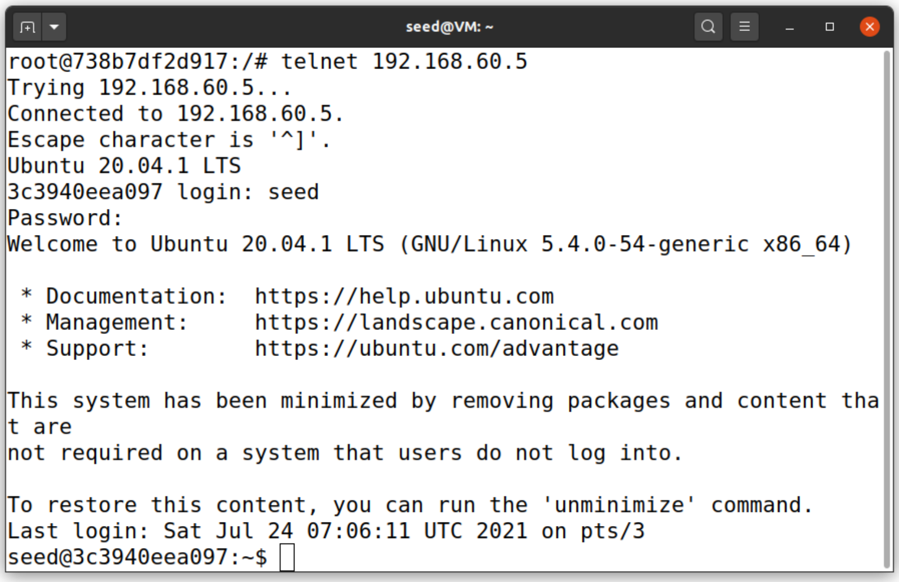
具体要求与Task2.C相比有所变化，如下。

> 1. All the internal hosts run a telnet server(listening to port 23). Outside hosts can only access the telnet server 192.168.60.5, not the other internal hosts.
> 2. Outside hosts cannot access other internal hosts.
> 3. Internal hosts can access all the internal servers.
> 4. **Internal hosts ~~cannot~~ can access external servers.**
> 5. ~~In this task, the connection tracking mechanism is not allowed. It will be used in a later task.~~

在router上运行以下命令即可。

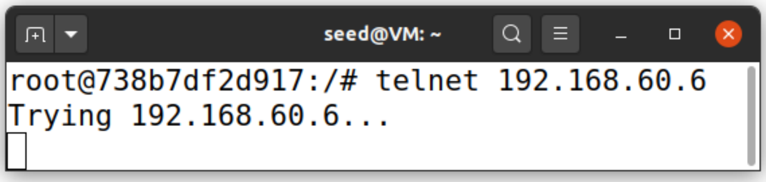
```
iptables -A FORWARD -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -p tcp -d 192.168.60.5 -i eth0 --dport 23 --syn -m conntrack
--ctstate NEW -j ACCEPT
iptables -A FORWARD -p tcp -i eth1 --dport 23 --syn -m conntrack --ctstate NEW -
j ACCEPT
iptables -P FORWARD DROP
```

- Outside hosts can only access the telnet server 192.168.60.5



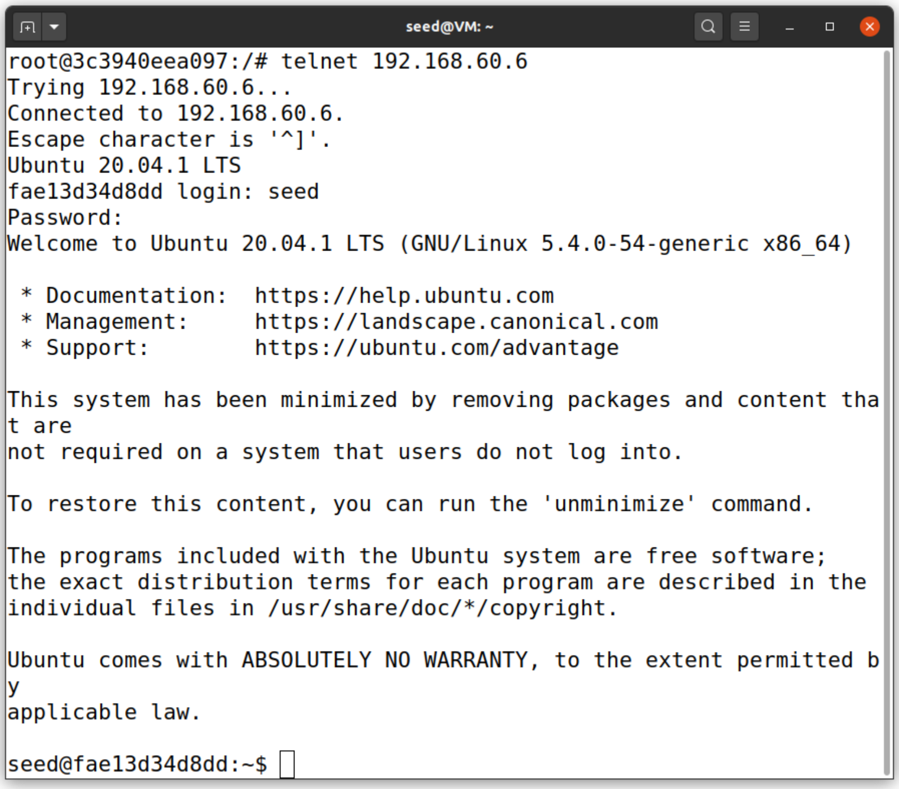
```
root@738b7df2d917:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
3c3940eea097 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content tha
t are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sat Jul 24 07:06:11 UTC 2021 on pts/3
seed@3c3940eea097:~$ 
```

- Outside hosts cannot access other internal hosts.



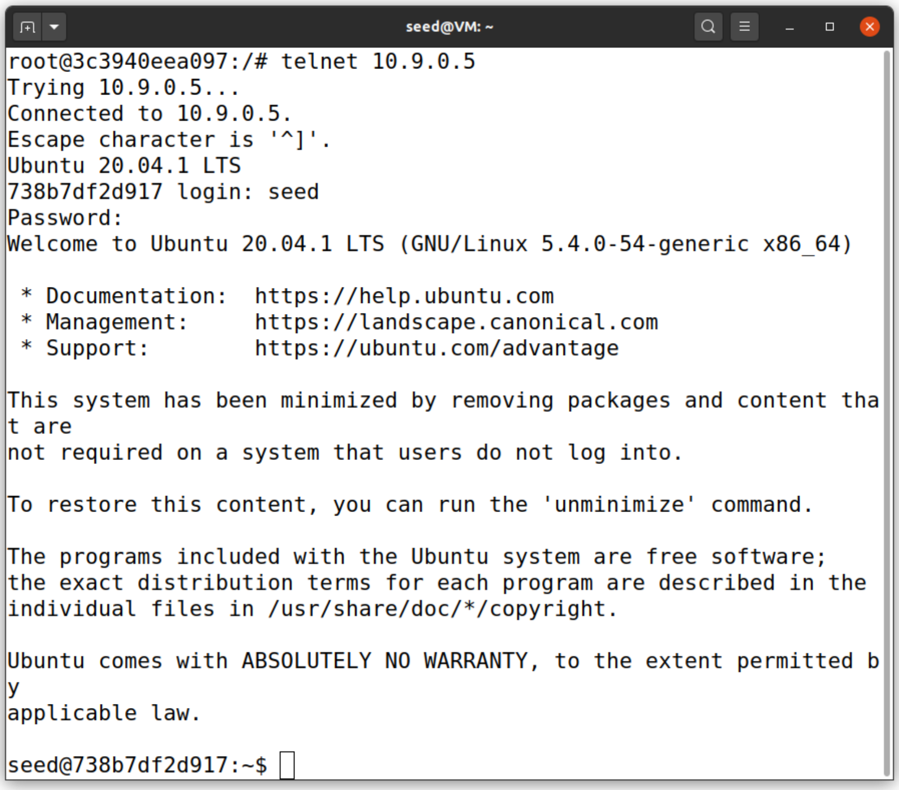
```
root@738b7df2d917:/# telnet 192.168.60.6
Trying 192.168.60.6...
```

- Internal hosts can access all the internal servers.



```
root@3c3940eea097:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
fae13d34d8dd login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content tha
t are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted b
y
applicable law.

seed@fae13d34d8dd:~$ 
```

- Internal hosts can access external servers.



```
root@3c3940eea097:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
738b7df2d917 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content tha
t are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted b
y
applicable law.

seed@738b7df2d917:~$ 
```

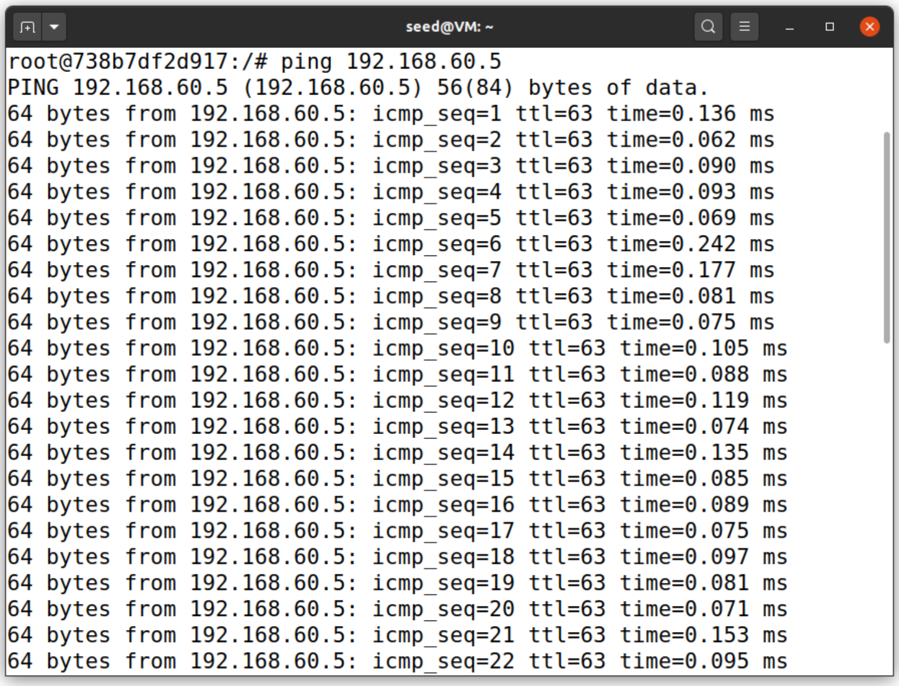
有状态的防火墙肯定会更加精确，书写规则也一定更方便，但牺牲的是一定的存储空间和包过滤的性能。

Task 4: Limiting Network Traffic

在router上运行下面这条命令。

```
iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute --limit-burst 5 -j
ACCEPT
```

然后在10.9.0.5上ping192.168.60.5.此时ICMP报文并没有收到限制。

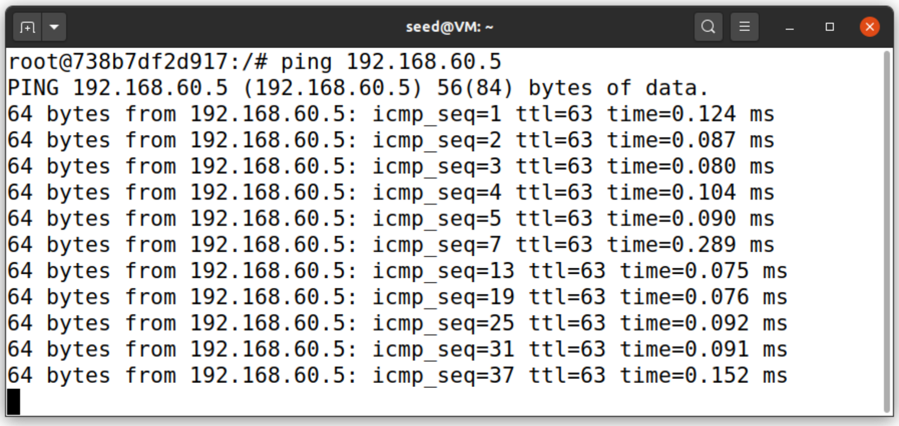


```
root@738b7df2d917:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.136 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.062 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.090 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.093 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.069 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.242 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.177 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.081 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.075 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.105 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.088 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.119 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.074 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.135 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.089 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.075 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.097 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.081 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.071 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.153 ms
64 bytes from 192.168.60.5: icmp_seq=22 ttl=63 time=0.095 ms
```

增加下面这条命令，然后再次进行PING命令。

```
iptables -A FORWARD -s 10.9.0.5 -j DROP
```

可以通过序列号明显的发现，从第5个报文后，开始出现丢包的现象，也就是说受到了限制。



```
root@738b7df2d917:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.124 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.087 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.080 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.104 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.090 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.289 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.075 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.076 ms
64 bytes from 192.168.60.5: icmp_seq=25 ttl=63 time=0.092 ms
64 bytes from 192.168.60.5: icmp_seq=31 ttl=63 time=0.091 ms
64 bytes from 192.168.60.5: icmp_seq=37 ttl=63 time=0.152 ms
```

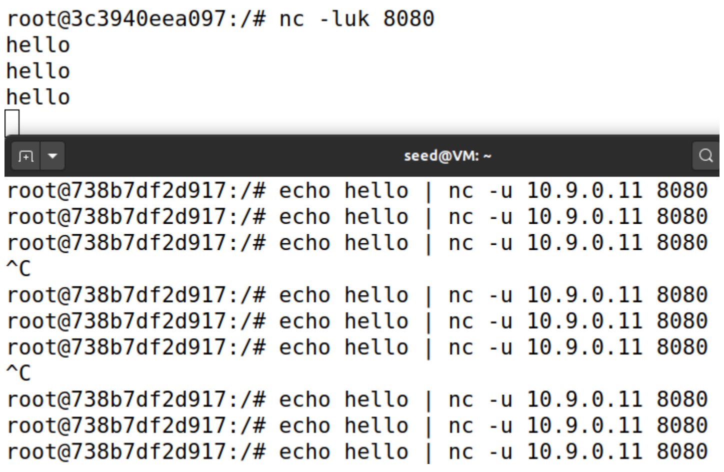
Task 5: Load Balancing

- Using the nth mode (round-robin)

在router上运行下面这条命令。

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --
every 3 --packet 0 -j DNAT --to-destination 192.168.60.5:8080
```

可以看到每3个报文就有一个发送给192.168.60.5.



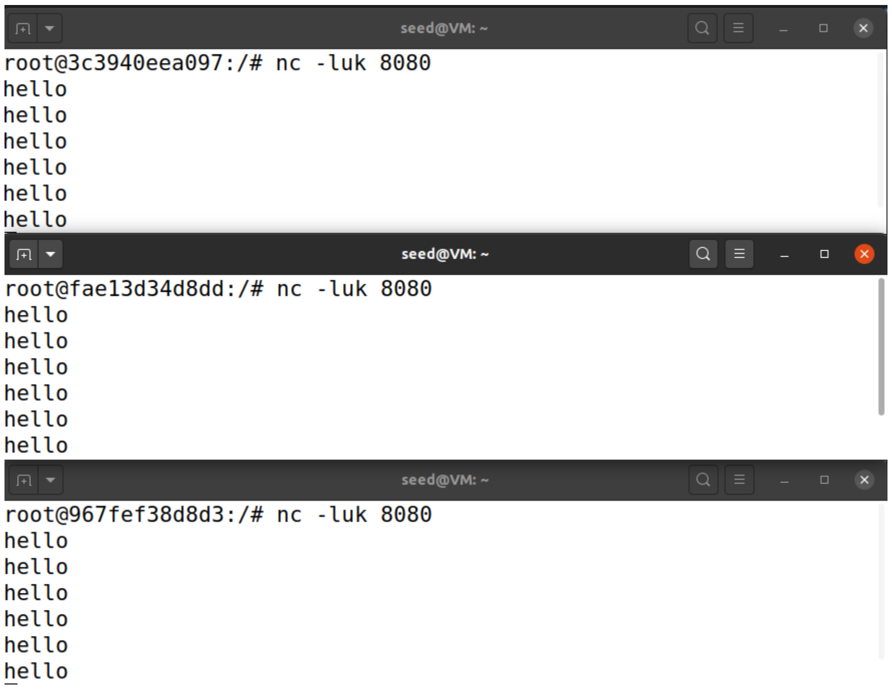
```
root@3c3940eea097:/# nc -luk 8080
hello
hello
hello
```

```
root@738b7df2d917:/# echo hello | nc -u 10.9.0.11 8080
root@738b7df2d917:/# echo hello | nc -u 10.9.0.11 8080
root@738b7df2d917:/# echo hello | nc -u 10.9.0.11 8080
^C
root@738b7df2d917:/# echo hello | nc -u 10.9.0.11 8080
root@738b7df2d917:/# echo hello | nc -u 10.9.0.11 8080
root@738b7df2d917:/# echo hello | nc -u 10.9.0.11 8080
^C
root@738b7df2d917:/# echo hello | nc -u 10.9.0.11 8080
root@738b7df2d917:/# echo hello | nc -u 10.9.0.11 8080
root@738b7df2d917:/# echo hello | nc -u 10.9.0.11 8080
```

再增加如下两条命令，使报文做到轮询式负载均衡。

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --
every 3 --packet 1 -j DNAT --to-destination 192.168.60.6:8080
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --
every 3 --packet 2 -j DNAT --to-destination 192.168.60.7:8080
```

可以看到三个server上依次收到报文。



```
root@3c3940eea097:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
```

```
root@fae13d34d8dd:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
```

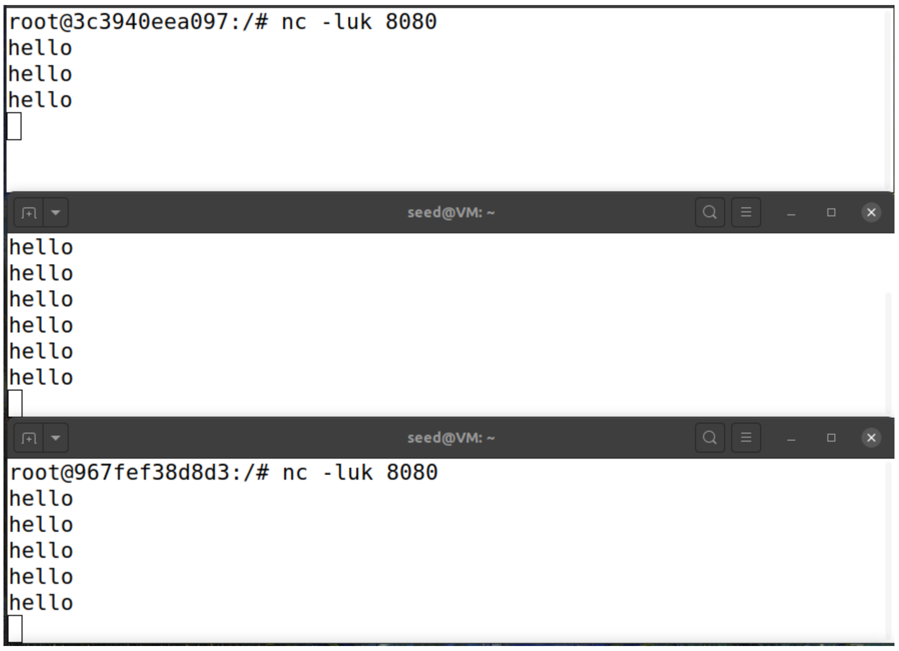
```
root@967fef38d8d3:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
```

- Using the random mode

在router上运行的命令如下，三个server概率均等为0.33

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.33 -j DNAT --to-destination 192.168.60.5:8080
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.33 -j DNAT --to-destination 192.168.60.6:8080
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.33 -j DNAT --to-destination 192.168.60.7:8080
```

可以看到基本上处于一个负载均衡的状态。



```
root@3c3940eea097:/# nc -luk 8080
hello
hello
hello
```

```
hello
hello
hello
hello
hello
hello
```

```
root@967fef38d8d3:/# nc -luk 8080
hello
hello
hello
hello
hello
```