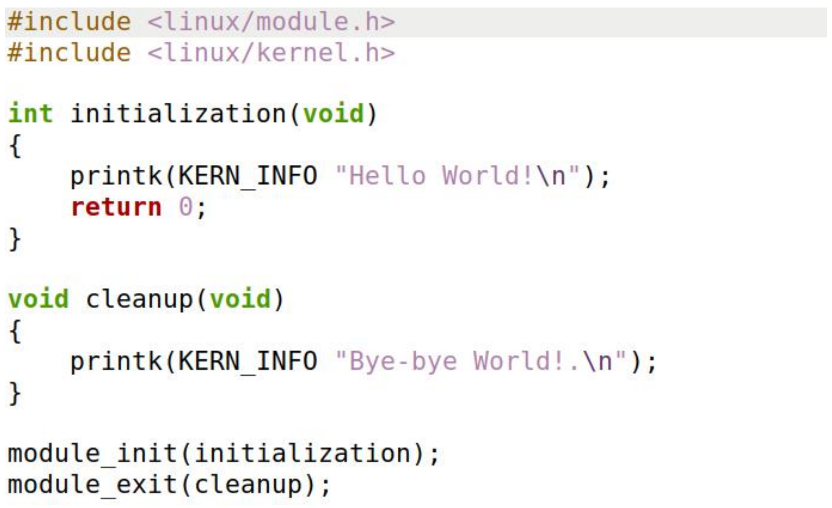
网络空间安全实训 第六次实验报告

57118221 梅昊

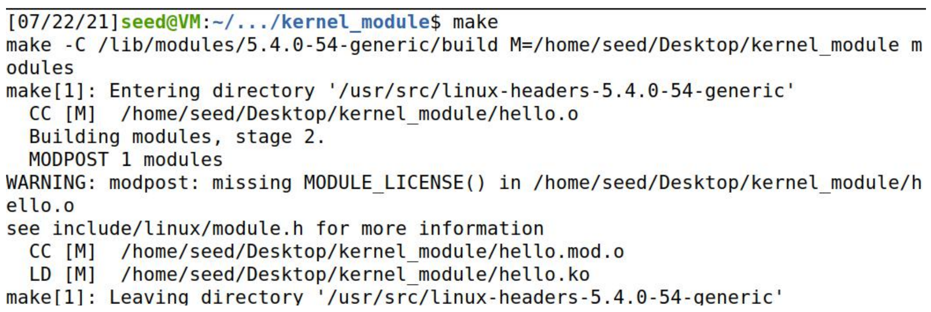
Task 1.A

1. 找到相关的 hello.c 文件。



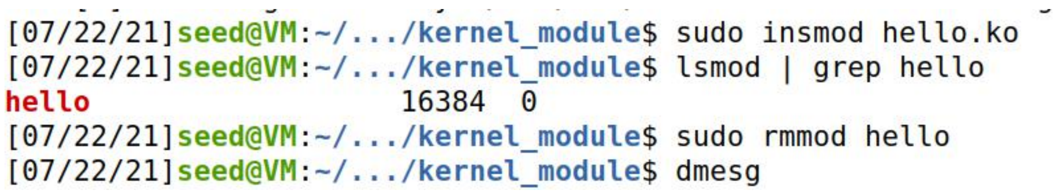
```
#include <linux/module.h>
#include <linux/kernel.h>

int initialization(void)
{
    printk(KERN_INFO "Hello World!\n");
    return 0;
}

void cleanup(void)
{
    printk(KERN_INFO "Bye-bye World!.\n");
}

module_init(initialization);
module_exit(cleanup);
```

2. 在 VM 上完成 hello.c 的编译。



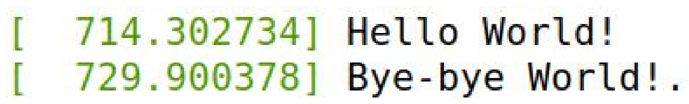
```
[07/22/21]seed@VM:~/.../kernel_module$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Desktop/kernel_module m
odules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Desktop/kernel_module/hello.o
  Building modules, stage 2.
  MODPOST 1 modules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

3. 添加内核模块，可以观察到内核模块列表里已经有了 hello 模块的信息。



```
[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$ sudo rmmod hello
[07/22/21]seed@VM:~/.../kernel_module$ dmesg
```

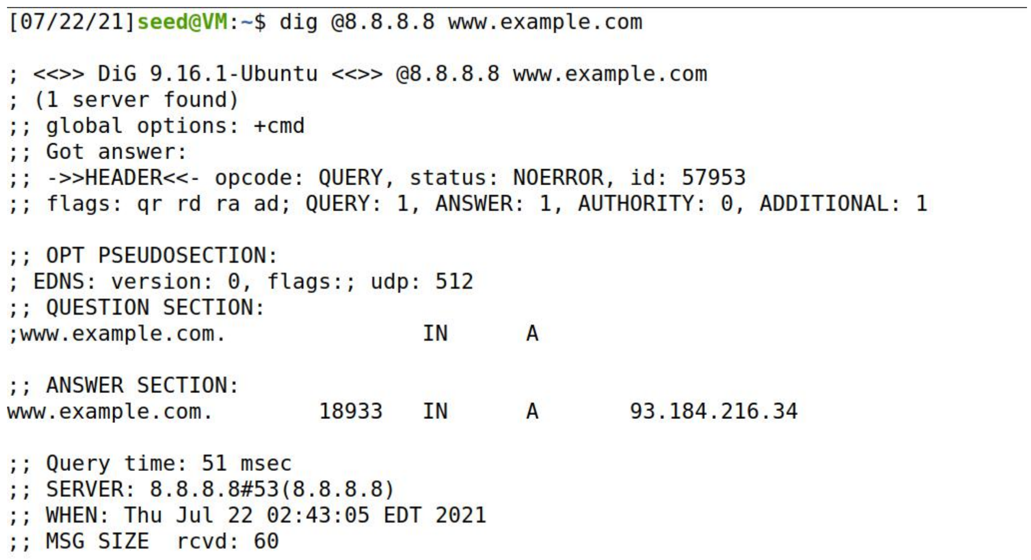
4. 在系统消息中可以看到加载和退出 hello 模块的消息。



```
[  714.302734] Hello World!
[  729.900378] Bye-bye World!.
```

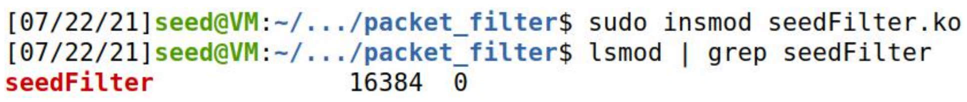
Task 1.B.1

1. 首先使用 dig 命令，观察在正常情况下得到的消息。



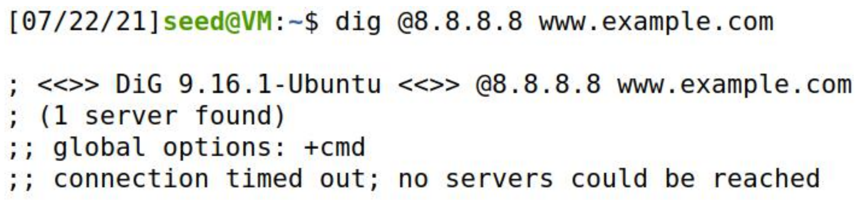
```
[07/22/21]seed@VM:~$ dig @8.8.8.8 www.example.com

; <<>> 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: 57953
;; 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.        18933   IN      A       93.184.216.34

;; Query time: 51 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Jul 22 02:43:05 EDT 2021
;; MSG SIZE  rcvd: 60
```

2. 编译运行 seedFilter.c 文件，并动态加载内核模块。



```
[07/22/21]seed@VM:~/.../packet_filter$ sudo insmod seedFilter.ko
[07/22/21]seed@VM:~/.../packet_filter$ lsmod | grep seedFilter
seedFilter             16384  0
```

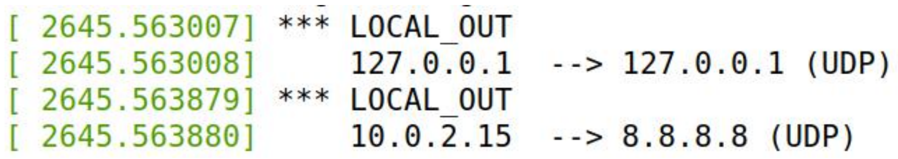
3. 此时无法连接 8.8.8.8。



```
[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
;; connection timed out; no servers could be reached
```

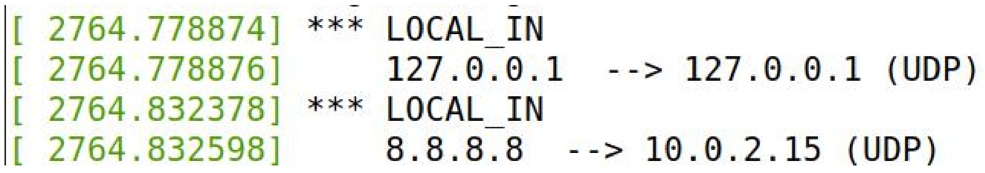
Task 1.B.2

1. 首先是 LOCAL_OUT，是本机产生的数据包到达的第一个点。可以看出只有本机产生的包会触发该 hook。



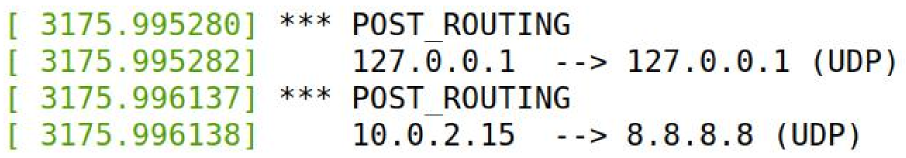
```
[ 2645.563007] *** LOCAL_OUT
[ 2645.563008]     127.0.0.1  --> 127.0.0.1 (UDP)
[ 2645.563879] *** LOCAL_OUT
[ 2645.563880]     10.0.2.15  --> 8.8.8.8 (UDP)
```

2. 然后是 LOCAL_IN，是发给本机不进行转发的包通过的点。可以看出目的地址是本机。



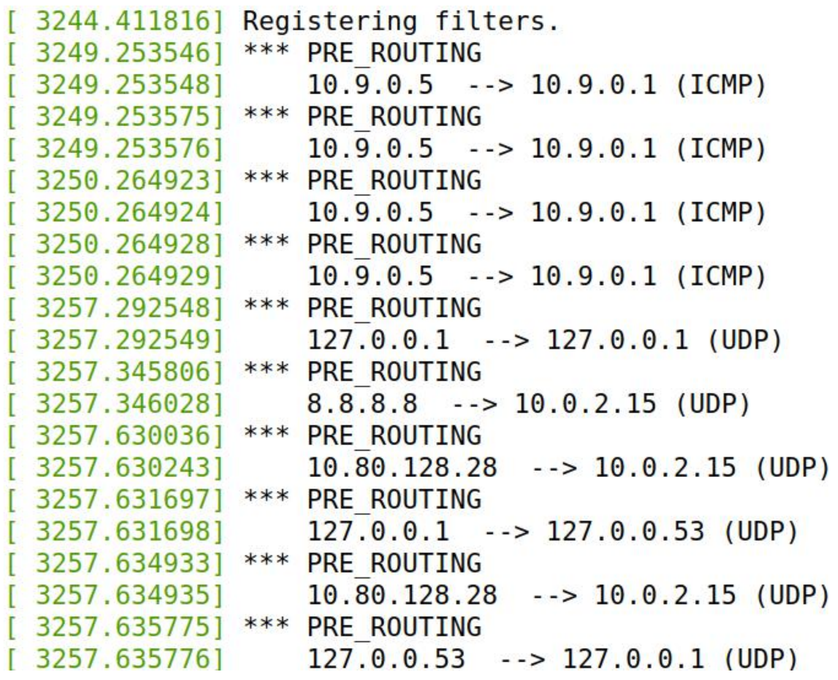
```
[ 2764.778874] *** LOCAL_IN
[ 2764.778876]     127.0.0.1  --> 127.0.0.1 (UDP)
[ 2764.832378] *** LOCAL_IN
[ 2764.832598]     8.8.8.8  --> 10.0.2.15 (UDP)
```

3. POST_ROUTING 是需要被转发或本机产生的数据包通过的点。源地址为本机，或这个包通过本机转发。



```
[ 3175.995280] *** POST_ROUTING
[ 3175.995282]     127.0.0.1  --> 127.0.0.1 (UDP)
[ 3175.996137] *** POST_ROUTING
[ 3175.996138]     10.0.2.15  --> 8.8.8.8 (UDP)
```

4. PRE_ROUTING 是所有数据包都经过的 hook。除了混杂模式下 sniff 到的包。可以看到基本所有的包都可以触发了这个 hook。



```
[ 3244.411816] Registering filters.
[ 3249.253546] *** PRE_ROUTING
[ 3249.253548]     10.9.0.5  --> 10.9.0.1 (ICMP)
[ 3249.253575] *** PRE_ROUTING
[ 3249.253576]     10.9.0.5  --> 10.9.0.1 (ICMP)
[ 3250.264923] *** PRE_ROUTING
[ 3250.264924]     10.9.0.5  --> 10.9.0.1 (ICMP)
[ 3250.264928] *** PRE_ROUTING
[ 3250.264929]     10.9.0.5  --> 10.9.0.1 (ICMP)
[ 3257.292548] *** PRE_ROUTING
[ 3257.292549]     127.0.0.1  --> 127.0.0.1 (UDP)
[ 3257.345806] *** PRE_ROUTING
[ 3257.346028]     8.8.8.8  --> 10.0.2.15 (UDP)
[ 3257.630036] *** PRE_ROUTING
[ 3257.630243]     10.80.128.28  --> 10.0.2.15 (UDP)
[ 3257.631697] *** PRE_ROUTING
[ 3257.631698]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 3257.634933] *** PRE_ROUTING
[ 3257.634935]     10.80.128.28  --> 10.0.2.15 (UDP)
[ 3257.635775] *** PRE_ROUTING
[ 3257.635776]     127.0.0.53  --> 127.0.0.1 (UDP)
```

5. 对于 FORWARD，没有捕获的相关的数据。因为只有被转发的数据包才会通过这个 hook。相关的环境在之前的实验中已经有了涉及，在此不再赘述。

Task 1.B.3

1. 编写函数 blockICMP 与 blockTELNET。用来阻隔 ICMP 与 TELNET 的相关报文。

```
unsigned int blockICMP(void *priv, struct sk_buff *skb,
                       const struct nf_hook_state *state)
{
   struct iphdr *iph;
   iph = ip_hdr(skb);
   if (iph->protocol == IPPROTO_ICMP){
       printk(KERN_WARNING "*** Dropping %pI4 (ICMP)\n", &(iph-
>daddr));
       return NF_DROP;
   }
   return NF_ACCEPT;
}
```

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

   u16  port   = 23;

   if (!skb) return NF_ACCEPT;
   iph = ip_hdr(skb);

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

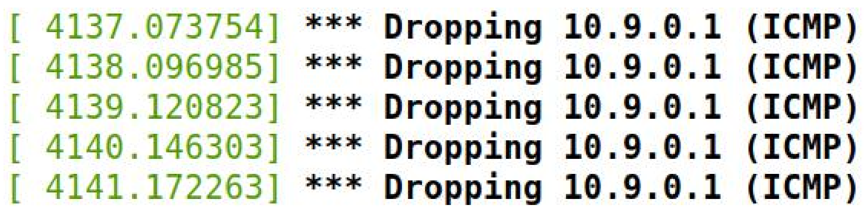
2. 在 register 函数中注册这两个 hook。

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

   hook1.hook = blockICMP;
   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 = blockTELNET;
   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;
}
```

3. 编译运行相关的文件，并将其加载入内核模块。可以看到无法 ping 通。

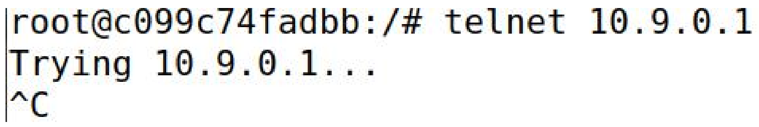
```
root@c099c74fadbb:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
^C
--- 10.9.0.1 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4080ms
```

4. 查看系统消息，有多个 drop 的 ICMP 报文。

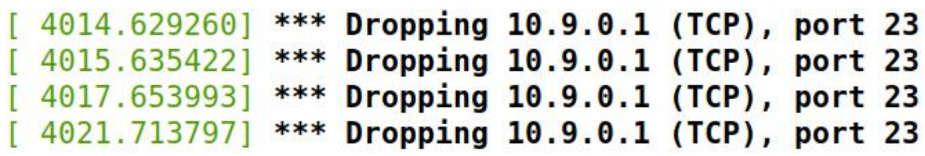


```
[ 4137.073754] *** Dropping 10.9.0.1 (ICMP)
[ 4138.096985] *** Dropping 10.9.0.1 (ICMP)
[ 4139.120823] *** Dropping 10.9.0.1 (ICMP)
[ 4140.146303] *** Dropping 10.9.0.1 (ICMP)
[ 4141.172263] *** Dropping 10.9.0.1 (ICMP)
```

5. 在 10.9.0.5 上 Telnet 10.9.0.1，可以看到无法建立连接。查看系统消息，发现有多个 drop 的 tcp 报文。防火墙起到了应有的作用。



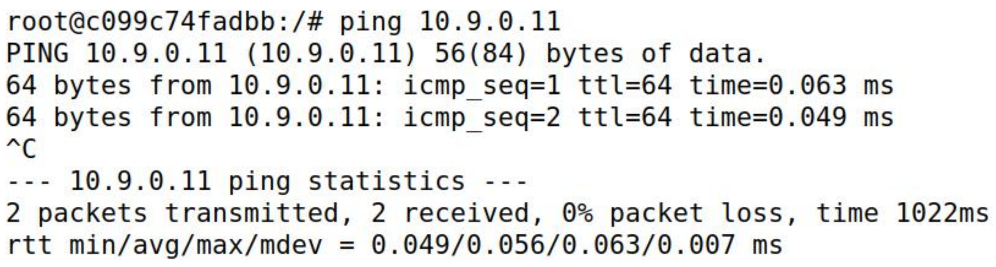
```
root@c099c74fadbb:/# telnet 10.9.0.1
Trying 10.9.0.1...
^C
```



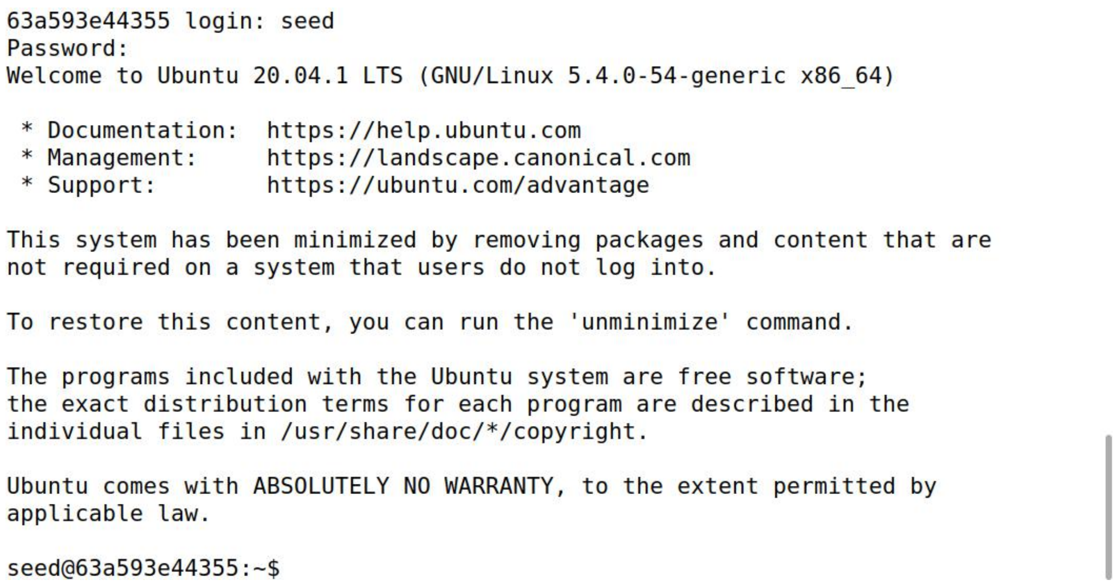
```
[ 4014.629260] *** Dropping 10.9.0.1 (TCP), port 23
[ 4015.635422] *** Dropping 10.9.0.1 (TCP), port 23
[ 4017.653993] *** Dropping 10.9.0.1 (TCP), port 23
[ 4021.713797] *** Dropping 10.9.0.1 (TCP), port 23
```

Task 2.A

1. 首先在 10.9.0.5 上 ping 路由器，并且对路由器进行 telnet。可以看到都是可以建立连接的。

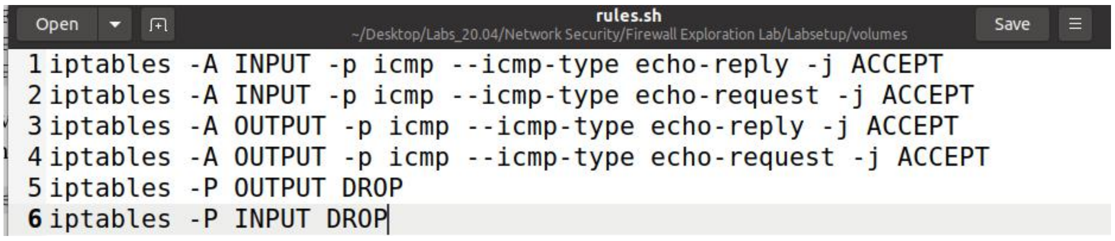


```
root@c099c74fadbb:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.063 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.049 ms
^C
--- 10.9.0.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1022ms
rtt min/avg/max/mdev = 0.049/0.056/0.063/0.007 ms
```



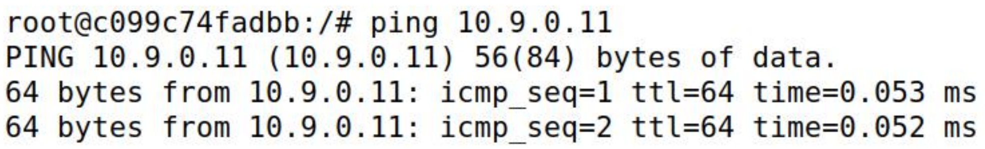
```
63a593e44355 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * 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 that 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@63a593e44355:~$
```

2. 编写 iptables 规则，并且在 router 上运行。

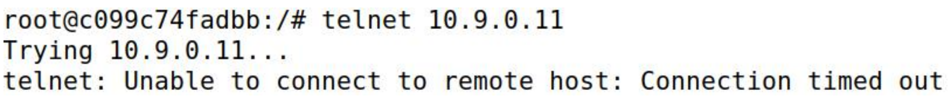

Open
rules.sh
~/Desktop/Labs_20.04/Network Security/Firewall Exploration Lab/Labsetup/volumes
Save

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

3. 再次在 10.9.0.5 上 ping 和 telnet 路由器。可以看到 ping 可以联通，但 telnet 无法建立连接。证明规则编写是正确的。



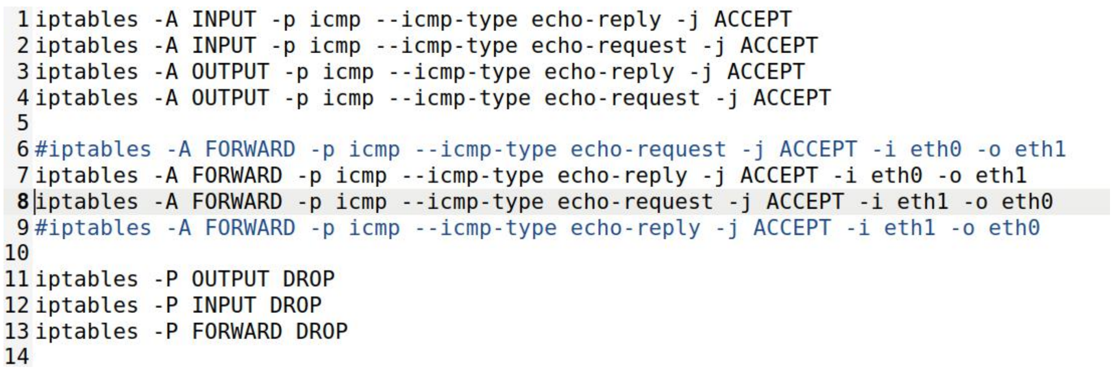
```
root@c099c74fadbb:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.053 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.052 ms
```



```
root@c099c74fadbb:/# telnet 10.9.0.11
Trying 10.9.0.11...
telnet: Unable to connect to remote host: Connection timed out
```

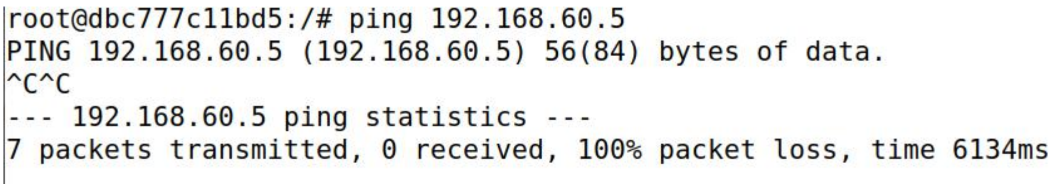
Task 2.B

1. 编写 iptables 规则，并且在 router 上运行。



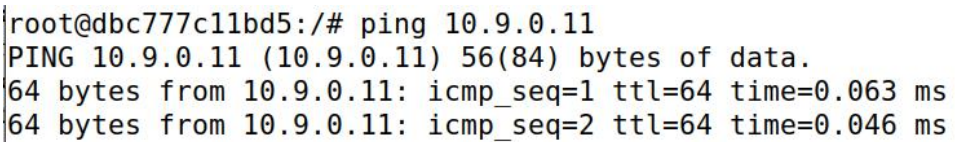
```
iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT

#iptables -A FORWARD -p icmp --icmp-type echo-request -j ACCEPT -i eth0 -o eth1
iptables -A FORWARD -p icmp --icmp-type echo-reply -j ACCEPT -i eth0 -o eth1
iptables -A FORWARD -p icmp --icmp-type echo-request -j ACCEPT -i eth1 -o eth0
#iptables -A FORWARD -p icmp --icmp-type echo-reply -j ACCEPT -i eth1 -o eth0

iptables -P OUTPUT DROP
iptables -P INPUT DROP
iptables -P FORWARD DROP

```

2. 在外部 ping 内网，无法连通。



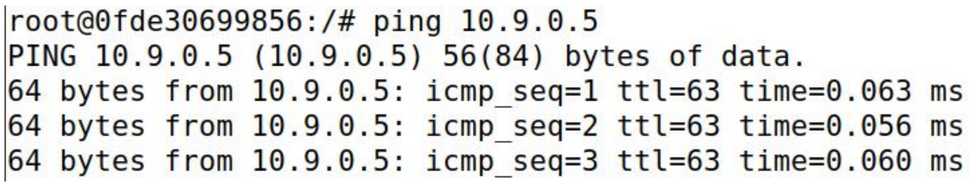
```
root@dbc777c11bd5:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C^C
--- 192.168.60.5 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6134ms
```

3. 在外部 ping 路由器，可以连通。



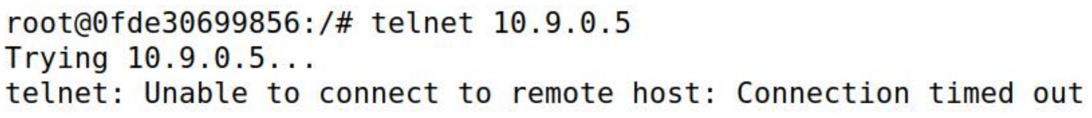
```
root@dbc777c11bd5:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.063 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.046 ms
```

4. 在内部 ping 外部，可以连通。

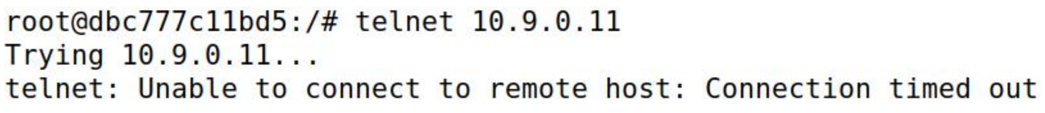


```
root@0fde30699856:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.063 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.056 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.060 ms
```

5. 除了 ping 以外的通信，例如 telnet，均无法联通。



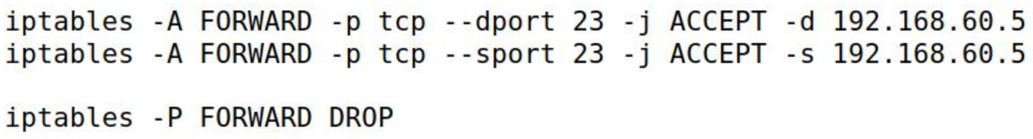
```
root@0fde30699856:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```



```
root@dbc777c11bd5:/# telnet 10.9.0.11
Trying 10.9.0.11...
telnet: Unable to connect to remote host: Connection timed out
```

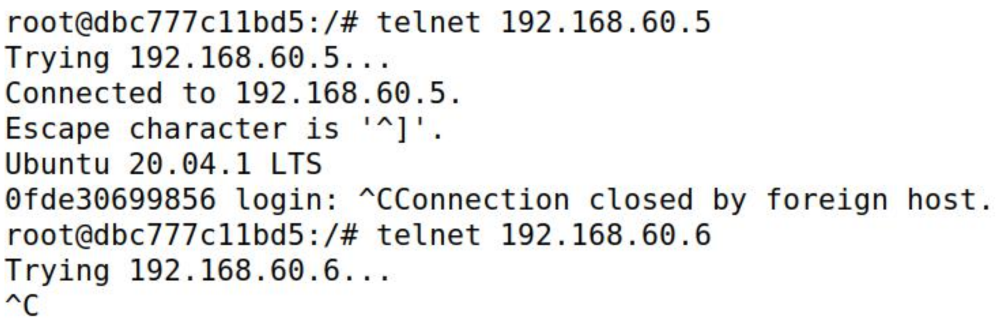
Task 2.C

1. 编写相关的 iptables 规则，并在 router 上运行。



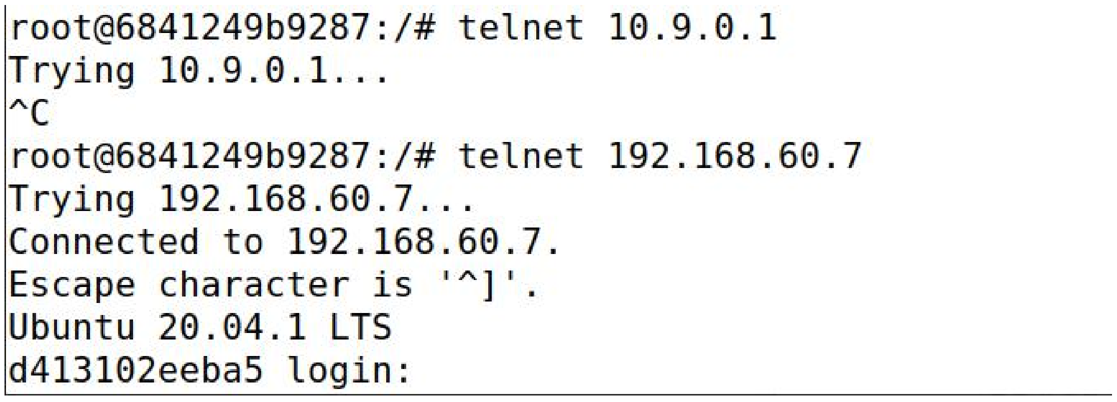
```
iptables -A FORWARD -p tcp --dport 23 -j ACCEPT -d 192.168.60.5
iptables -A FORWARD -p tcp --sport 23 -j ACCEPT -s 192.168.60.5

iptables -P FORWARD DROP
```

2. 在外部的 host，如 10.9.0.5 上 telnet 192.168.60.5 可以连通，但是其他的服务器无法连通。



```
root@dbc777c11bd5:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
0fde30699856 login: ^CConnection closed by foreign host.
root@dbc777c11bd5:/# telnet 192.168.60.6
Trying 192.168.60.6...
^C
```

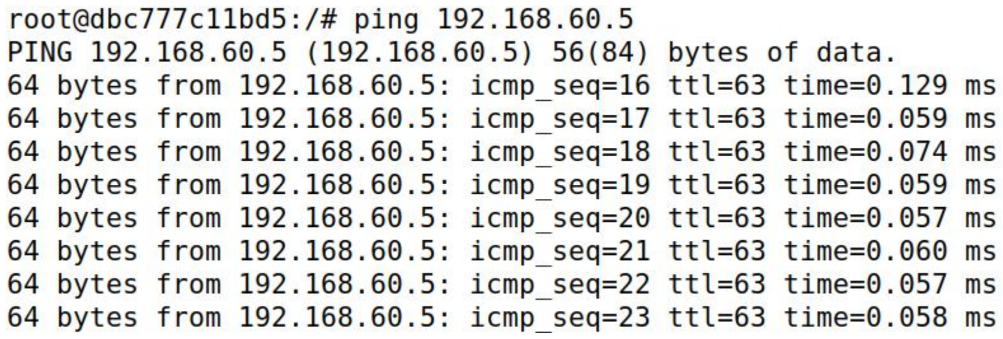
3. 在内部的 host 尝试 telnet 外部，无法连通。但可以连通其他在内部的服务。



```
root@6841249b9287:/# telnet 10.9.0.1
Trying 10.9.0.1...
^C
root@6841249b9287:/# telnet 192.168.60.7
Trying 192.168.60.7...
Connected to 192.168.60.7.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
d413102eeba5 login:
```

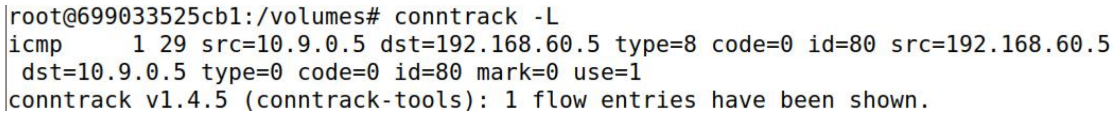
Task 3.A

1. 在 10.6.0.5 上使用 ping。



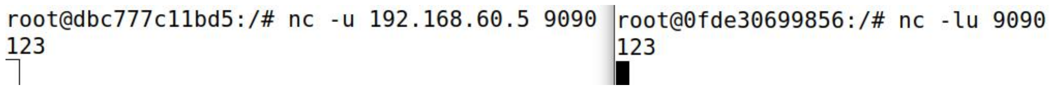
```
root@dbc777c11bd5:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.129 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.074 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.060 ms
64 bytes from 192.168.60.5: icmp_seq=22 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=23 ttl=63 time=0.058 ms
```

2. 可以观测到相应的记录项。存活时间为 30s。



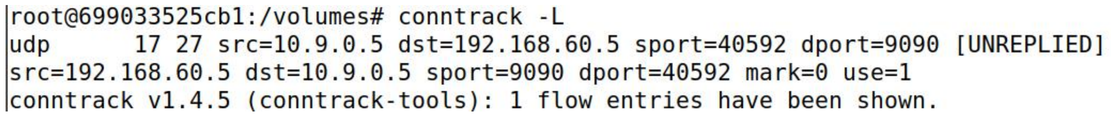
```
root@699033525cb1:/volumes# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=80 src=192.168.60.5
 dst=10.9.0.5 type=0 code=0 id=80 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

3. 使用 udp 连接，可以看到倒计时。存活时间同样为 30s。

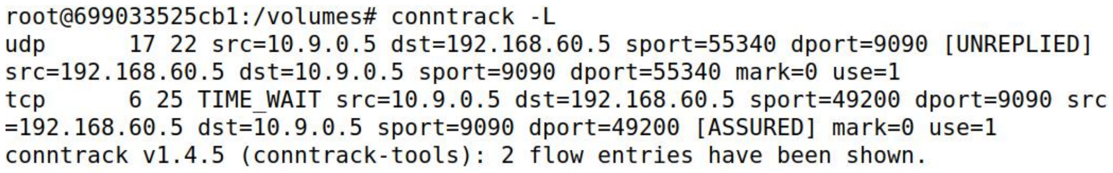


```
root@dbc777c11bd5:/# nc -u 192.168.60.5 9090
123
```

```
root@0fde30699856:/# nc -lu 9090
123
```

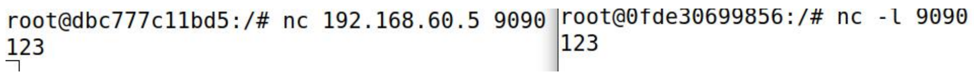


```
root@699033525cb1:/volumes# conntrack -L
udp      17 27 src=10.9.0.5 dst=192.168.60.5 sport=40592 dport=9090 [UNREPLIED]
src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=40592 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```



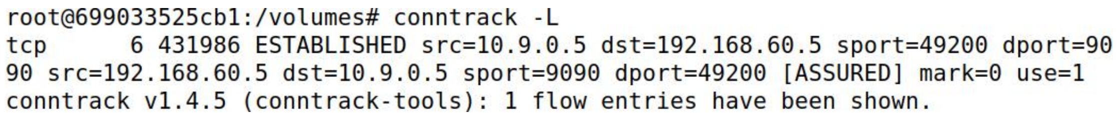
```
root@699033525cb1:/volumes# conntrack -L
udp      17 22 src=10.9.0.5 dst=192.168.60.5 sport=55340 dport=9090 [UNREPLIED]
src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=55340 mark=0 use=1
tcp      6 25 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=49200 dport=9090 src
=192.168.60.5 dst=10.9.0.5 sport=9090 dport=49200 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
```

4. 使用 tcp 连接。可以 track 到相应的表项。在连接建立状态下，存活时间为 43 万多秒。当连接切断后，进入等待状态，等待时间为 120s。

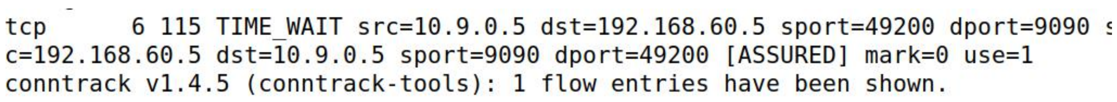


```
root@dbc777c11bd5:/# nc 192.168.60.5 9090
123
```

```
root@0fde30699856:/# nc -l 9090
123
```



```
root@699033525cb1:/volumes# conntrack -L
tcp      6 431986 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=49200 dport=90
90 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=49200 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```



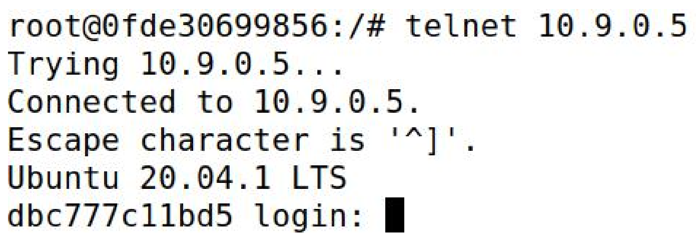
```
tcp      6 115 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=49200 dport=9090 s
c=192.168.60.5 dst=10.9.0.5 sport=9090 dport=49200 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

Task 3.B

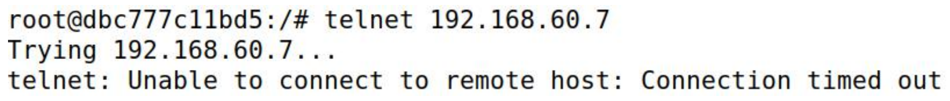
1. 利用 conntrack，编写 iptables 的相关规则，并在 router 上运行。

```
iptables -A FORWARD -p tcp --dport 23 -j ACCEPT -d 192.168.60.5
iptables -A FORWARD -p tcp --sport 23 -j ACCEPT -s 192.168.60.5

iptables -A FORWARD -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -p tcp -m conntrack --ctstate NEW -i eth1 -j ACCEPT


iptables -P FORWARD DROP
```

2. 在内部 host，如 192.168.60.7 上尝试访问外网服务，可以建立连接。但是在外网尝试访问内网服务，如 192.168.60.7，则无法建立相关连接。



```
root@0fde30699856:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
dbc777c11bd5 login: █
```



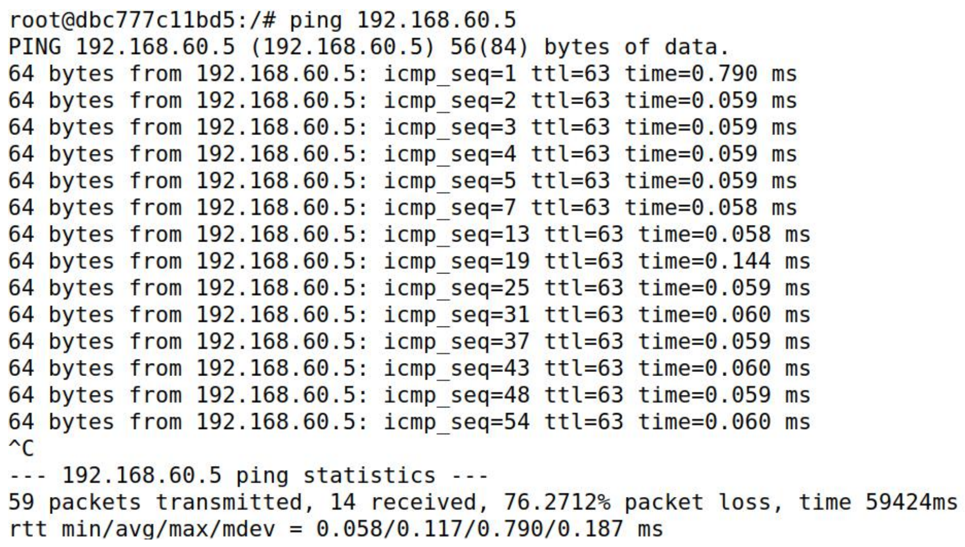
```
root@dbc777c11bd5:/# telnet 192.168.60.7
Trying 192.168.60.7...
telnet: Unable to connect to remote host: Connection timed out
```

Task 4

1. 首先在 router 上运行相关的规则。

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

2. 开始 ping。可以观察到，前五个报文接受速度很快，后面相对较慢。最后的统计数据可以看到，除了前五个 burst 限制的报文，其余报文的接受速度基本上是每分钟 10 个，符合预期。



```
root@dbc777c11bd5:/# 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=0.790 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.058 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.058 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.144 ms
64 bytes from 192.168.60.5: icmp_seq=25 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=31 ttl=63 time=0.060 ms
64 bytes from 192.168.60.5: icmp_seq=37 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=43 ttl=63 time=0.060 ms
64 bytes from 192.168.60.5: icmp_seq=48 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=54 ttl=63 time=0.060 ms
^C
--- 192.168.60.5 ping statistics ---
59 packets transmitted, 14 received, 76.2712% packet loss, time 59424ms
rtt min/avg/max/mdev = 0.058/0.117/0.790/0.187 ms
```

3. 如果去掉第二条规则，再次运行 ping，可以看到，报文并没有受到规则的限制。因为没有最后的 DROP 规则，没有在第一条规则下通过的报文会直接通过最底层的 ACCEPT 规则通过。因此第一条规则也没有起到作用。

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

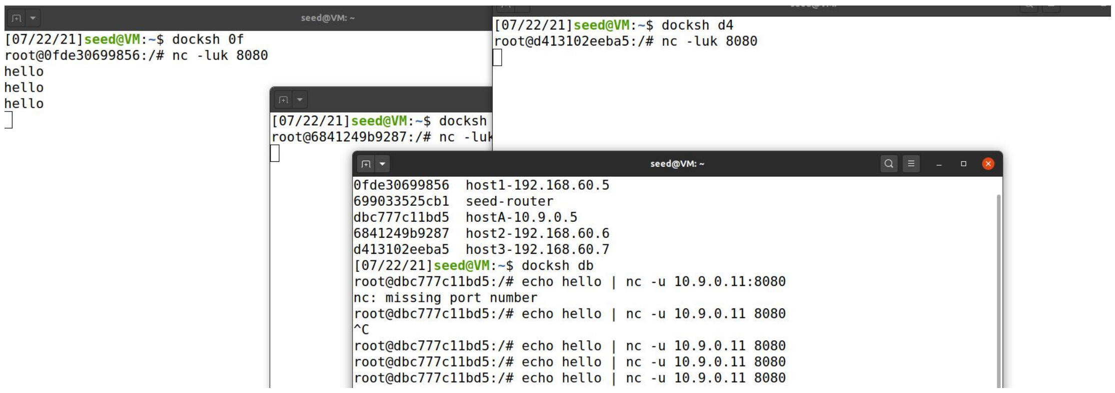
```
root@dbc777c11bd5:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.064 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.060 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.067 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.077 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.058 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.060 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.063 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.067 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.056 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.059 ms
^C
--- 192.168.60.5 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14332ms
rtt min/avg/max/mdev = 0.056/0.061/0.077/0.005 ms
```

Task 5

1. 首先在 router 上运行相关的规则。

```
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
```

2. 在 10.9.0.5 上发送 hello，可以看到，每发送三次 hello 才会有一次被 192.168.60.5 接受。



```
[07/22/21]seed@VM:~$ docksh 0f
root@0fde30699856:/# nc -luk 8080
hello
hello
hello
```

```
[07/22/21]seed@VM:~$ docksh d4
root@d413102eeba5:/# nc -luk 8080
```

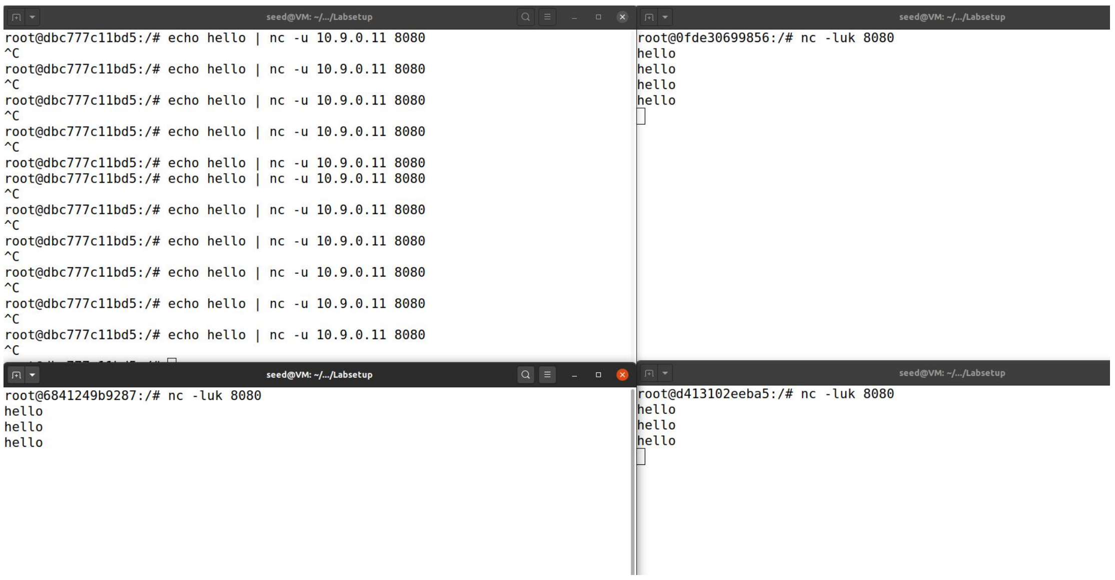
```
[07/22/21]seed@VM:~$ docksh
root@6841249b9287:/# nc -luk
```

```
0fde30699856  host1-192.168.60.5
699033525cb1  seed-router
dbc777c11bd5  hostA-10.9.0.5
6841249b9287  host2-192.168.60.6
d413102eeba5  host3-192.168.60.7
[07/22/21]seed@VM:~$ docksh db
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11:8080
nc: missing port number
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
```

3. 针对负载均衡的要求更改规则，并在 router 上运行。

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --ever
y 3 --packet 0 -j DNAT --to-destination 192.168.60.5:8080
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --ever
y 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 --ever
y 3 --packet 2 -j DNAT --to-destination 192.168.60.7:8080
```

4. 可以看到，在不停的发送 hello 时，三个服务器接受的报文是被负载均衡过的。



```
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
```

```
root@0fde30699856:/# nc -luk 8080
hello
hello
hello
hello
```

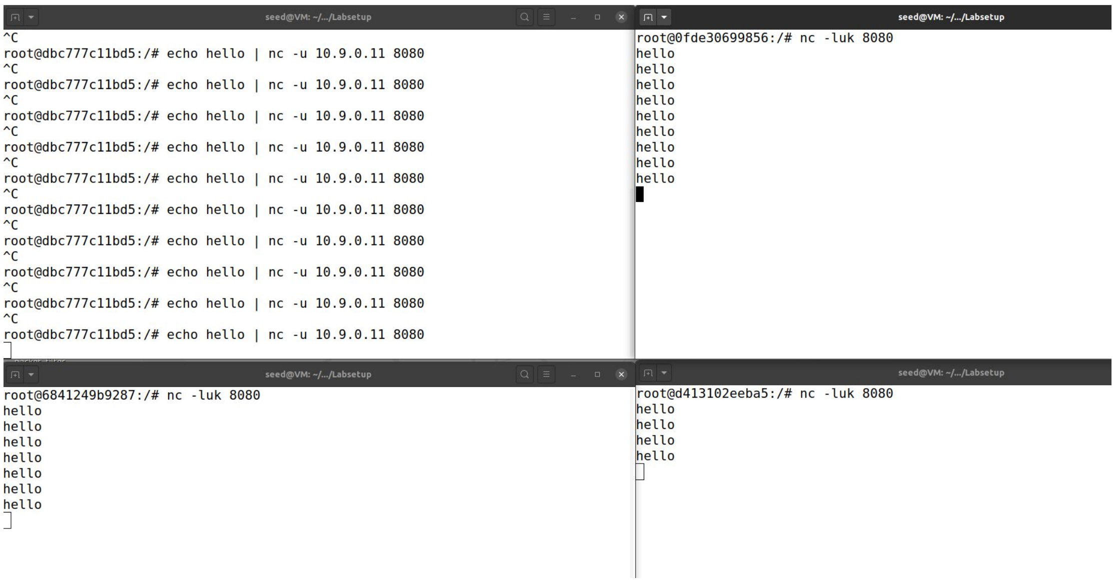
```
root@6841249b9287:/# nc -luk 8080
hello
hello
hello
```

```
root@d413102eeba5:/# nc -luk 8080
hello
hello
hello
```

5. 使用随机模式，设定三个服务器接受的概率均为 0.333。在 router 上运行规则。

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --p
robability 0.333 -j DNAT --to-destination 192.168.60.5:8080
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --p
robability 0.333 -j DNAT --to-destination 192.168.60.6:8080
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --p
robability 0.333 -j DNAT --to-destination 192.168.60.7:8080
```

6. 可以看到，在不停发送 hello 时，三个服务器接受的报文是被负载均衡过的。但由于概率原因，并不是完全均衡，仍然有概率因素存在。



```
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
^C
root@dbc777c11bd5:/# echo hello | nc -u 10.9.0.11 8080
```

```
root@0fde30699856:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

```
root@6841249b9287:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
hello
```

```
root@d413102eeba5:/# nc -luk 8080
hello
hello
hello
hello
```