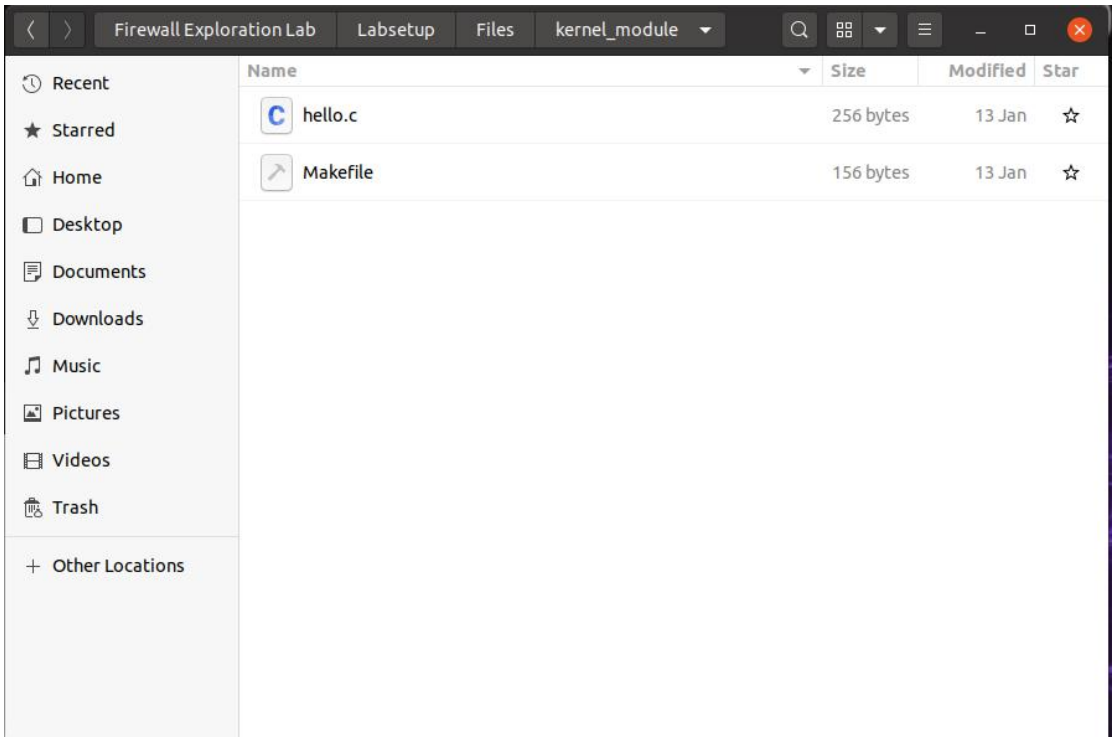
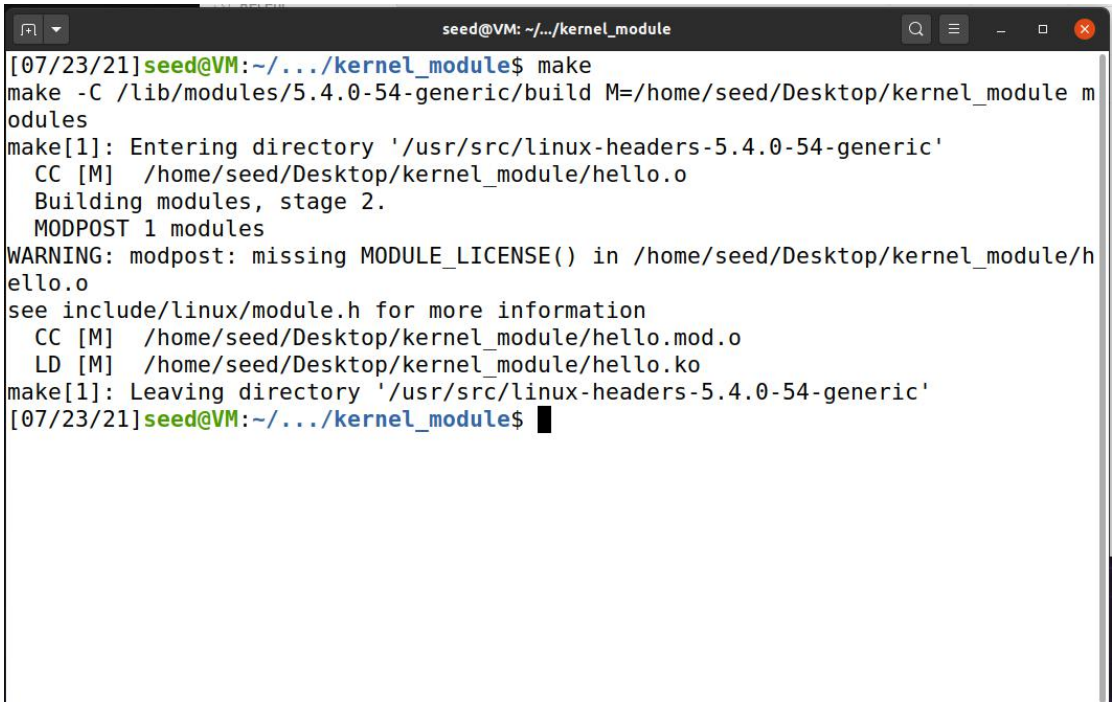
Task1.A

找到 kernel_module 文件夹，打开终端进行编译：





编译成功后，测试插入模块、罗列模块、移除模块、查看信息四条指令：

```
[07/24/21]seed@VM:~/.../kernel_module$ sudo insmod hello.ko
[07/24/21]seed@VM:~/.../kernel_module$  lsmod | grep hello
hello                  16384  0
[07/24/21]seed@VM:~/.../kernel_module$ sudo rmmod hello
[07/24/21]seed@VM:~/.../kernel_module$ dmesg
```

```
missing - tainting kernel
[ 4961.508998] Hello World!
[ 5005.207357] Bye-bye World!.
```

四条指令运行成功，信息中可以看到输出信息，证明安装成功。

Task1.B.1

在开始攻击前使用 dig 指令进行查询：

```
[07/24/21]seed@VM:~/.../kernel_module$ 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: 58715
;; 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.        19115   IN      A       93.184.216.34

;; Query time: 56 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sat Jul 24 16:35:47 EDT 2021
;; MSG SIZE  rcvd: 60
```

编译 packet_filter 内核：

```
seed@VM: ~/.../packet_filter
[07/24/21]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Desktop/packet_filter m
odules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Desktop/packet_filter/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Desktop/packet_filter/seedFilter.mod.o
  LD [M]  /home/seed/Desktop/packet_filter/seedFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[07/24/21]seed@VM:~/.../packet_filter$
```

加载模块：

```
[07/24/21]seed@VM:~/.../packet_filter$  sudo insmod seedFilter.ko
[07/24/21]seed@VM:~/.../packet_filter$ lsmod | grep seedFilter
seedFilter             16384  0
```

再测试 dig 指令:

```
[07/24/21]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
```

可以观察到 dig 指令显示连接超时，证明模块加载成功。

Task1.B.2:

修改 seedFilter.c 的程序，增加 hook3，hook4，hook5，对应 FORWARD、LOCAL OUT、POST ROUTING，如下所示：

```
11
12 static struct nf_hook_ops hook1, hook2, hook3, hook4, hook5;
```

```
hook3.hook = printfo;
hook3.hooknum = NF_INET_FORWARD;
hook3.pf = PF_INET;
hook3.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook3);

hook4.hook = printfo;
hook4.hooknum = NF_INET_LOCAL_OUT;
hook4.pf = PF_INET;
hook4.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook4);

hook5.hook = printfo;
hook5.hooknum = NF_INET_POST_ROUTING;
hook5.pf = PF_INET;
hook5.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook5);
```

```
110 void removeFilter(void) {
111     printk(KERN_INFO "The filters are being removed.\n");
112     nf_unregister_net_hook(&init_net, &hook1);
113     nf_unregister_net_hook(&init_net, &hook2);
114     nf_unregister_net_hook(&init_net, &hook3);
115     nf_unregister_net_hook(&init_net, &hook4);
116     nf_unregister_net_hook(&init_net, &hook5);
117 }
```

重新编译：

```
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Desktop/packet_filter m
odules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Desktop/packet_filter/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Desktop/packet_filter/seedFilter.mod.o
  LD [M]  /home/seed/Desktop/packet_filter/seedFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
```

加载模块：

```
[07/24/21]seed@VM:~/.../packet_filter$ sudo insmod seedFilter.ko
[07/24/21]seed@VM:~/.../packet_filter$ lsmod | greep seedFilter

Command 'greep' not found, did you mean:

  command 'greed' from deb greed (4.2-1)
  command 'grep' from deb grep (3.4-1)

Try: sudo apt install <deb name>

[07/24/21]seed@VM:~/.../packet_filter$ lsmod | grep seedFilter
seedFilter             16384  0
```

再次测试 dig，并在之后查看 dmesg：

```
[10022.431144] *** LOCAL_OUT
[10022.431145]     192.168.228.130  --> 8.8.8.8 (UDP)
[10022.431150] *** POST_ROUTING
[10022.431151]     192.168.228.130  --> 8.8.8.8 (UDP)
[10022.482571] *** PRE_ROUTING
[10022.482574]     8.8.8.8  --> 192.168.228.130 (UDP)
[10022.482586] *** LOCAL_IN
[10022.482587]     8.8.8.8  --> 192.168.228.130 (UDP)
```

可以观察到，对于本地产生向外部网络发出的数据包，本地产生包时触发 LOCAL_OUT 的处理，向外发送时触发 POST_ROUTING 的处理。对于发往本地的数据包，在判断发往外部还是本地接收时触发 PRE_ROUTING 的处理，在发往本地时触发 LOCAL_IN 的处理。dmesg 中没有 FORWARD 的相关处理，可能是在需要向外转发时触发。

Task1.B.3:

增加 blockICMP 函数用于拦截 ICMP 报文，增加 blockTelnet 函数用于拦截 telnet 报文：

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

    iph = ip_hdr(skb);
    // Convert the IPv4 address from dotted decimal to 32-bit binary

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

    }
    return NF_ACCEPT;
}
```

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

   u16  port   = 23;

   iph = ip_hdr(skb);
   // Convert the IPv4 address from dotted decimal to 32-bit binary


   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;
}
```

所有触发 NF_INET_LOCAL_IN 的都是发往本地的包，因此无需再比较 ip，只需要将函数注册为 NF_INET_LOCAL_IN 点：

```c

    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);


```

再次编译：

```
[07/24/21]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Desktop/packet_filter m
odules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Desktop/packet_filter/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Desktop/packet_filter/seedFilter.mod.o
  LD [M]  /home/seed/Desktop/packet_filter/seedFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
```

加载模块

```
[07/24/21]seed@VM:~/.../packet_filter$ sudo insmod seedFilter.ko
[07/24/21]seed@VM:~/.../packet_filter$ lsmod | grep seedFilter
seedFilter             16384  0
[07/24/21]seed@VM:~/.../packet_filter$
```

在 hostA 主机上 ping 10.9.0.1 进行测试，可以看到 ping 不通：

```
root@f2afe487c033:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
```

查看 dmesg 可以看到 ICMP 包的丢弃，证明 ICMP 拦截成功：

```
[11816.693984] *** Dropping 192.168.228.130 (ICMP)
[11817.731324] *** Dropping 192.168.228.130 (ICMP)
[11818.755262] *** Dropping 192.168.228.130 (ICMP)
[11819.776571] *** Dropping 192.168.228.130 (ICMP)
[11820.804908] *** Dropping 192.168.228.130 (ICMP)
[11821.848844] *** Dropping 192.168.228.130 (ICMP)
[11822.844964] *** Dropping 192.168.228.130 (ICMP)
[11823.870904] *** Dropping 192.168.228.130 (ICMP)
[11824.895509] *** Dropping 192.168.228.130 (ICMP)
[11825.916981] *** Dropping 192.168.228.130 (ICMP)
[11826.940233] *** Dropping 192.168.228.130 (ICMP)
[11827.966058] *** Dropping 192.168.228.130 (ICMP)
[11828.991103] *** Dropping 192.168.228.130 (ICMP)
[11830.011711] *** Dropping 192.168.228.130 (ICMP)
[11831.043806] *** Dropping 192.168.228.130 (ICMP)
[11832.065302] *** Dropping 192.168.228.130 (ICMP)
```

再在 hostA 主机上测试 telnet 登录 10.9.0.1，也连接不上无法登陆：

```
root@f2afe487c033:/# telnet 10.9.0.1
Trying 10.9.0.1...
^C
```

查看 dmesg 可以看到 TCP 丢包，证明 Telnet 拦截成功：

```
[12300.965253] *** Dropping 10.9.0.1 (TCP), port 23
[12301.976818] *** Dropping 10.9.0.1 (TCP), port 23
[12303.993364] *** Dropping 10.9.0.1 (TCP), port 23
```

Task2.A

在设置防火墙之前，在主机上测试 ping 路由器地址可以看到能 ping 通：

```
root@f2afe487c033:/# 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=0.186 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.093 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.068 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.075 ms
```

在路由器上上设置防火墙规则如下：

```
root@d81d8992dbf0:/# iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
root@d81d8992dbf0:/# iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
root@d81d8992dbf0:/# iptables -t filter -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 8

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 0
root@d81d8992dbf0:/# iptables -P OUTPUT DROP
root@d81d8992dbf0:/# iptables -P INPUT DROP
```

再次测试 ping，发现此时 ping 指令依然成功：

```
root@f2afe487c033:/# 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=0.087 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.129 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.073 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.071 ms
64 bytes from 10.9.0.11: icmp_seq=6 ttl=64 time=0.069 ms
64 bytes from 10.9.0.11: icmp_seq=7 ttl=64 time=0.069 ms
^C
--- 10.9.0.11 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6123ms
rtt min/avg/max/mdev = 0.069/0.081/0.129/0.020 ms
```

测试 telnet 指令，发现 telnet 指令失败，证明防火墙规则设置成功：

```
root@f2afe487c033:/# telnet 10.9.0.11
Trying 10.9.0.11...
```

Task2.B：

在路由器上查看端口对应的 ip 地址，其中 eth0 面向外网，eth1 面向内网：

```
root@d81d8992dbf0:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.11  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:0b  txqueuelen 0  (Ethernet)
        RX packets 90  bytes 8724 (8.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 21  bytes 1778 (1.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.60.11  netmask 255.255.255.0  broadcast 192.168.60.255
        ether 02:42:c0:a8:3c:0b  txqueuelen 0  (Ethernet)
        RX packets 46  bytes 4868 (4.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

设置路由器规则如下：

```
root@d81d8992dbf0:/# iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
root@d81d8992dbf0:/# iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
root@d81d8992dbf0:/# iptables -A FORWARD -p icmp --icmp-type echo-request -o eth0 -j ACCEPT
root@d81d8992dbf0:/# iptables -A FORWARD -p icmp --icmp-type echo-request -i eth1 -j ACCEPT
root@d81d8992dbf0:/# iptables -A FORWARD -p icmp --icmp-type echo-reply -i eth0 -j ACCEPT
root@d81d8992dbf0:/# iptables -A FORWARD -p icmp --icmp-type echo-reply -o eth1 -j ACCEPT
root@d81d8992dbf0:/# iptables -P OUTPUT DROP
root@d81d8992dbf0:/# iptables -P INPUT DROP
root@d81d8992dbf0:/# iptables -P FORWARD DROP
root@d81d8992dbf0:/# iptables -t filter -L -n
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 8

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 8
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 8
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 0
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 0

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 0
```

测试外网 ping 内网，ping 指令失败：

```
root@f2afe487c033:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
```

测试内网 ping 外网，ping 指令成功：

```
root@62d08b8cc2e3:/# 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=0.128 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.081 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.092 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.100 ms
^C
--- 10.9.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3070ms
rtt min/avg/max/mdev = 0.081/0.100/0.128/0.017 ms
root@62d08b8cc2e3:/#
```

测试外网 ping 路由器，ping 指令成功：

```
root@62d08b8cc2e3:/# 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=0.206 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.068 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.070 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.071 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.076 ms
64 bytes from 10.9.0.11: icmp_seq=6 ttl=64 time=0.069 ms
64 bytes from 10.9.0.11: icmp_seq=7 ttl=64 time=0.098 ms
^C
--- 10.9.0.11 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6132ms
rtt min/avg/max/mdev = 0.068/0.094/0.206/0.046 ms
```

测试外网 telnet 登录内网，telnet 登录失败：

```
root@f2afe487c033:/# telnet 192.168.60.5
Trying 192.168.60.5...
```

测试内网 telnet 登录外网，telnet 登陆失败：

```
root@62d08b8cc2e3:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

以上测试结果均符合要求，证明防火墙规则设置成功，正常生效。

Task2.C：

设计防火墙规则如下：

```
root@d81d8992dbf0:/# iptables -A FORWARD -p tcp -d 192.168.60.5 --dport 23 -j ACCEPT
root@d81d8992dbf0:/# iptables -A FORWARD -p tcp -s 192.168.60.5 --sport 23 -j ACCEPT
root@d81d8992dbf0:/# iptables -P FORWARD DROP
root@d81d8992dbf0:/# iptables -t filter -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            192.168.60.5         tcp dpt:23
ACCEPT     tcp  --  192.168.60.5         0.0.0.0/0            tcp spt:23

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

测试外网 telnet 登录主机 192.168.60.5，连接成功：

```
root@f2afe487c033:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
^CUbuntu 20.04.1 LTS
62d08b8cc2e3 login:
telnet> quit
Connection closed.
```

测试外网 telnet 登录内网其他主机，均失败：

```
root@f2afe487c033:/# telnet 192.168.60.6
Trying 192.168.60.6...
^C
root@f2afe487c033:/# telnet 192.168.60.7
Trying 192.168.60.7...
^C
```

测试内网 telnet 登录外网，telnet 登陆失败：

```
root@62d08b8cc2e3:/# telnet 10.9.0.5
Trying 10.9.0.5...
^C
```

测试内网 telnet 登录内网，telnet 登陆成功：

```
root@62d08b8cc2e3:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
1f9b06d743f7 login:
```

以上测试结果均符合要求，证明防火墙规则设置成功，正常生效。

Task3.A:

ICMP

在主机上 pnig 主机 192.168.60.5：

```
root@f2afe487c033:/# 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=0.173 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.086 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.089 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.089 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.084 ms
^C
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5108ms
rtt min/avg/max/mdev = 0.084/0.101/0.173/0.032 ms
```

Ping 指令执行的同时在路由器上查看追踪信息：

```
root@d81d8992dbf0:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=43 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=43 mark=0 use=
1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@d81d8992dbf0:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=43 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=43 mark=0 use=
1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@d81d8992dbf0:/# conntrack -L
icmp     1 27 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=43 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=43 mark=0 use=
1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@d81d8992dbf0:/# conntrack -L
icmp     1 26 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=43 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=43 mark=0 use=
1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@d81d8992dbf0:/# conntrack -L
icmp     1 13 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=43 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=43 mark=0 use=
1
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=44 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=44 mark=0 use=
1
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
root@d81d8992dbf0:/#
```

可以看到一个 icmp 持续时间为 30s，停止 ping 操作，icmp 会逐渐减少到 0，如果 30s 内再次 ping，则会显示两个 flow。

UDP

在 10.9.0.5 主机上用 nc 连接 192.168.60.5，并发送消息进行测试：

```
root@f2afe487c033:/# nc -u 192.168.60.5 9090
Liu
LZK
^C
root@f2afe487c033:/# nc -u 192.168.60.5 9090
LIU
```

```
root@62d08b8cc2e3:/# nc -lu 9090
Liu
LZK
^C
root@62d08b8cc2e3:/# nc -lu 9090
LIU
```

同时在路由器上查看追踪信息：

```
root@d81d8992dbf0:/# conntrack -L
udp      17 26 src=10.9.0.5 dst=192.168.60.5 sport=50299 dport=9090 [UNREPLIED] src=192.168.60.5 dst=10.9.0.5 sport=90
t=50299 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@d81d8992dbf0:/# conntrack -L
udp      17 12 src=10.9.0.5 dst=192.168.60.5 sport=50299 dport=9090 [UNREPLIED] src=192.168.60.5 dst=10.9.0.5 sport=90
t=50299 mark=0 use=1
udp      17 27 src=10.9.0.5 dst=192.168.60.5 sport=50719 dport=9090 [UNREPLIED] src=192.168.60.5 dst=10.9.0.5 sport=90
t=50719 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
```

可以观察到现象与 ICMP 一致，持续时间也为 30s，并在连接结束后逐渐减少至 0，持续时间内再次连接则出现两个 flow。

Tcp

在 10.9.0.5 主机上用 nc 连接 192.168.60.5，并发送消息进行测试：

```
root@f2afe487c033:/# nc 192.168.60.5 9090
liu
^C
```

```
root@62d08b8cc2e3:/# nc -l 9090
liu
```

同时在路由器上查看追踪信息：

```
root@d81d8992dbf0:/# conntrack -L
tcp      6 431944 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=45776 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 d
port=45776 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

```
root@d81d8992dbf0:/# conntrack -L
tcp      6 116 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=45774 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=
45774 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

可以观察到 TCP 的存在时间较长，大于 43000s，在连接结束后变为 120s 的存活时间，并逐渐减少到 0。

Task3.B

在路由器中设置防火墙规则如下：

```
root@d81d8992dbf0:/# iptables -A FORWARD -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
root@d81d8992dbf0:/# iptables -A FORWARD -p tcp -m conntrack --ctstate NEW -i eth1 -j ACCEPT
root@d81d8992dbf0:/# iptables -P FORWARD DROP
root@d81d8992dbf0:/# iptables -t filter -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            ctstate RELATED,ESTABLISHED
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            ctstate NEW

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

测试内网 telent 登录外网，telnet 登录成功：

```
root@62d08b8cc2e3:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
f2afe487c033 login:
```

测试外网 telnet 登录内网，telnet 登录失败：

```
root@f2afe487c033:/# telnet 192.168.60.5
Trying 192.168.60.5...
```

以上现象符合要求，证明防火墙规则设置成功，成功生效。

Task4：

在防火墙设置规则如下：

```
root@d81d8992dbf0:/# iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute --limit-burst 5 -j ACCEPT
root@d81d8992dbf0:/# iptables -A FORWARD -s 10.9.0.5 -j DROP
root@d81d8992dbf0:/# iptables -t filter -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  10.9.0.5             0.0.0.0/0            limit: avg 10/min burst 5
DROP       all  --  10.9.0.5             0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

测试 10.9.0.5　ping　192.168.60.5，实际现象为前几个报文速度较快，后面报文之间的间隔为 6s 左右，符合防火墙规则的设置：

```
root@f2afe487c033:/# 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=0.098 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.089 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.086 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.089 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.094 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.134 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.093 ms
64 bytes from 192.168.60.5: icmp_seq=25 ttl=63 time=0.086 ms
^C
--- 192.168.60.5 ping statistics ---
27 packets transmitted, 9 received, 66.6667% packet loss, time 26621ms
rtt min/avg/max/mdev = 0.085/0.094/0.134/0.014 ms
```

去掉第二条规则再次进行测试，可以观察到报文速度又恢复到较快的水平，防火墙规则未生效：

```
root@d81d8992dbf0:/# iptables -F
root@d81d8992dbf0:/# iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute --limit-burst 5 -j ACCEPT
root@d81d8992dbf0:/#
```

```
root@f2afe487c033:/# 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=0.150 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.129 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.089 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.087 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.087 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.092 ms
```

上述现象是因为去除第二条规后，没有将报文设置为 DROP，所有报文都会从 ACCEPT 规则通过，从而导致第一条规则也失效。

Task5

轮询：

在路由器中设置规则如下：

```
root@d81d8992dbf0:/# 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
root@d81d8992dbf0:/#
```

在 192.168.60.5 上开启 nc -luk 8080 监听，在 10.9.0.5 处进行 nc 连接并发送三次 hello：

```
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
```

可以看到发送三次 hello 后 192.168.60.5 端才接收到一个 hello，符合防火墙规则。

```
root@62d08b8cc2e3:/# nc -luk 8080
hello
```

随机：

在路由器中设置规则如下，实现对三个主机等概率的随机分发：

```
root@d81d8992dbf0:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.333 -j DNAT
 --to-destination 192.168.60.5:8080
root@d81d8992dbf0:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.5 -j DNAT -
-to-destination 192.168.60.6:8080
root@d81d8992dbf0:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random -j DNAT --to-destination 19
2.168.60.7:8080
iptables v1.8.4 (legacy): --probability must be specified when using random mode
Try `iptables -h' or 'iptables --help' for more information.
root@d81d8992dbf0:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -j DNAT --to-destination 192.168.60.7:8080
```

在三台主机上都打开 nc -luk 8080 监听，在 10.9.0.5 主机上建立 nc 连接并不断发送 hello：

```
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/#
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
^C
root@f2afe487c033:/# echo hello | nc -u 10.9.0.11 8080
```

得到三台主机上的输出结果如下：

```
root@62d08b8cc2e3:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

```
root@f6c3e67c985e:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

```
root@1f9b06d743f7:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

可以观察到三个主机中接收到的 hello 的数量大致相同，发送更多的消息样本进行测试可能会得到更好的结果，但以上现象足以证明防火墙规则正常生效，实现了负载均衡。