Lab6-report

57118239 张鹏

Task1:

Task1.A:

该次实验的目的是通过 LKM 添加一个模块，当模块加载时打印出“Hello World！”，当模块从内核中移除时，打印出“Bye-bye World！”，打印的内容在/var/log/syslog 文件中，可以通过 dmesg 命令查看。

下面是模块加载和移除时的函数，通过 printk 打印到日志文件中，并指定两个入口点的函数，代码如下:

hello.c

```c
#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);

```

以下是 makefile，用于编译内核模块。

Makefile

```
obj-m += hello.o

all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean

```

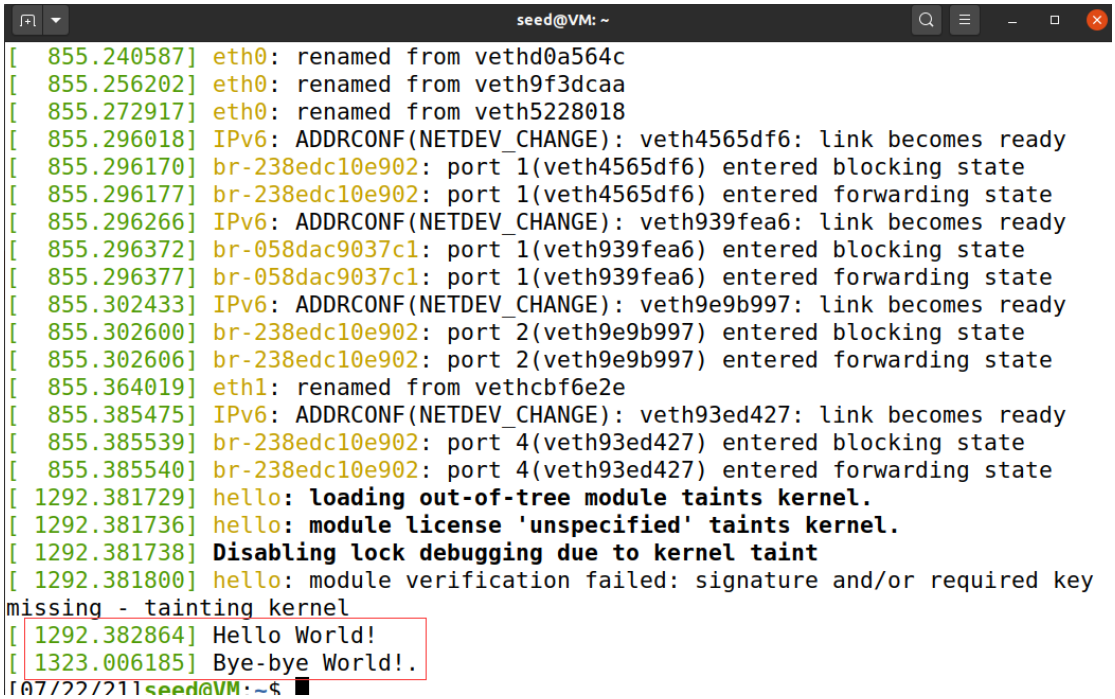
通过 make 命令对文件进行编译。

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

当内核模块生成后，通过 insmod 命令载入模块，通过 lsmod 命令查看，通过 rmmod 命令将其从内核中卸载。

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

通过 dmesg 命令查看日志文件中有相应的打印内容。



Task1.B.1：

在这个任务中，我们将以 LKM 的形式编写包过滤程序，然后将其插入到内核中的包处理路径中。

Netfilter 是为授权用户操作数据包而设计的。它通过在 Linux 内核中实现许多 hook 来实现这个目标。这些 hook 被插入到不同的位置，包括数据包的进出路径。如果我们想要操作传入的数据包，我们只需要将自己的程序(在 LKM 内)连接到相应的 hook。一旦传入数据包到达，我们的程序将被调用。我们的程序可以决定这个数据包是否应该被阻塞;此外，还可以对程序中的数据包进行修改。

根据提供的代码和 makefile 文件来编译该代码，并将其加载到内核中，查看防火墙是否有没有执行预定义的规则。给定的代码如下，将源码中的 DNS 服务器从 8.8.8.8 改为 114.114.114.114，代码如下：

```c
#include <linux/udp.h>
#include <linux/if_ether.h>
#include <linux/inet.h>


static struct nf_hook_ops hook1, hook2;


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

   u16  port   = 53;
   char ip[16] = "114.114.114.114";//"8.8.8.8";
   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_UDP) {
       udph = udp_hdr(skb);
       if (iph->daddr == ip_addr && ntohs(udph->dest) == port){
           printk(KERN_WARNING "*** Dropping %pI4 (UDP), port %d\n", &(iph->daddr), port);
           return NF_DROP;
        }
   }
```

makefile 如下：

```
obj-m += seedFilter.o
all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean

ins:
	sudo dmesg -C
	sudo insmod seedFilter.ko

rm:
	sudo rmmod seedFilter
```

编译代码。

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

将代码加载到内核。

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

通过该 DNS 服务器发起请求。

```
[07/22/21]seed@VM:~/Desktop$ dig @114.114.114.114 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @114.114.114.114 www.example.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

可以发现请求被阻塞了，达到预期的实验目的。

Task1.B.2:

NF_INET_LOCAL_OUT:

```
69                      &(ipn->saddr), &(ipn->daddr), protocol);
70
71    return NF_ACCEPT;
72 }
73
74
75 int registerFilter(void) {
76    printk(KERN_INFO "Registering filters.\n");
77
78    hook1.hook = printInfo;
79    hook1.hooknum = NF_INET_LOCAL_OUT;
80    hook1.pf = PF_INET;
81    hook1.priority = NF_IP_PRI_FIRST;
82    nf_register_net_hook(&init_net, &hook1);
83
84    hook2.hook = blockUDP;
85    hook2.hooknum = NF_INET_POST_ROUTING;
86    hook2.pf = PF_INET;
87    hook2.priority = NF_IP_PRI_FIRST;
88    nf_register_net_hook(&init_net, &hook2);
89
90    return 0;
91 }
92
93 void removeFilter(void) {
94    printk(KERN_INFO "The filters are being removed.\n");
95    nf_unregister_net_hook(&init_net, &hook1);
96    nf_unregister_net_hook(&init_net, &hook2);
97 }
98
```

```
[07/22/21]seed@VM:~$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/seedFilter.mod.o
  LD [M]  /home/seed/seedFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[07/22/21]seed@VM:~$ sudo insmod seedFilter.ko
[07/22/21]seed@VM:~$ lsmod | grep seedFilter
seedFilter             16384  0
[07/22/21]seed@VM:~/Desktop$ dig @114.114.114.114 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @114.114.114.114 www.example.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

```
[ 8458.855237]      192.168.43.81  --> 114.114.114.114 (UDP)
[ 8458.855241] *** Dropping 114.114.114.114 (UDP), port 53
[ 8463.855231] *** LOCAL_OUT
[ 8463.855236]      192.168.43.81  --> 114.114.114.114 (UDP)
[ 8463.855259] *** Dropping 114.114.114.114 (UDP), port 53
[ 8468.859152] *** LOCAL_OUT
[ 8468.859158]      192.168.43.81  --> 114.114.114.114 (UDP)
[ 8468.859274] *** Dropping 114.114.114.114 (UDP), port 53
[07/22/21]seed@VM:~$ sudo rmmod seedFilter
```

NF_INET_PRE_ROUTING:

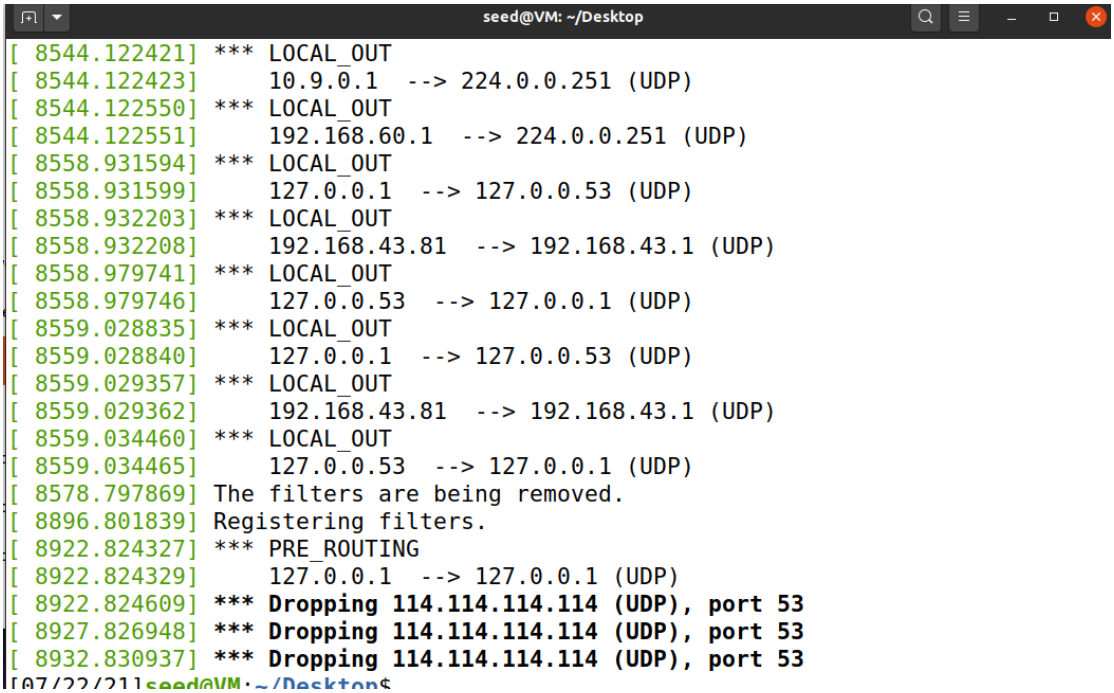
seedFilter.c

```
69                    &(iph->saddr), &(iph->daddr), protocol);
70
71    return NF_ACCEPT;
72}
73
74
75int registerFilter(void) {
76    printk(KERN_INFO "Registering filters.\n");
77
78    hook1.hook = printInfo;
79    hook1.hooknum = NF_INET_PRE_ROUTING;
80    hook1.pf = PF_INET;
81    hook1.priority = NF_IP_PRI_FIRST;
82    nf_register_net_hook(&init_net, &hook1);
83
84    hook2.hook = blockUDP;
85    hook2.hooknum = NF_INET_POST_ROUTING;
86    hook2.pf = PF_INET;
87    hook2.priority = NF_IP_PRI_FIRST;
88    nf_register_net_hook(&init_net, &hook2);
89
90    return 0;
91}
92
93void removeFilter(void) {
94    printk(KERN_INFO "The filters are being removed.\n");
95    nf_unregister_net_hook(&init_net, &hook1);
96    nf_unregister_net_hook(&init_net, &hook2);
97}
98
```

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

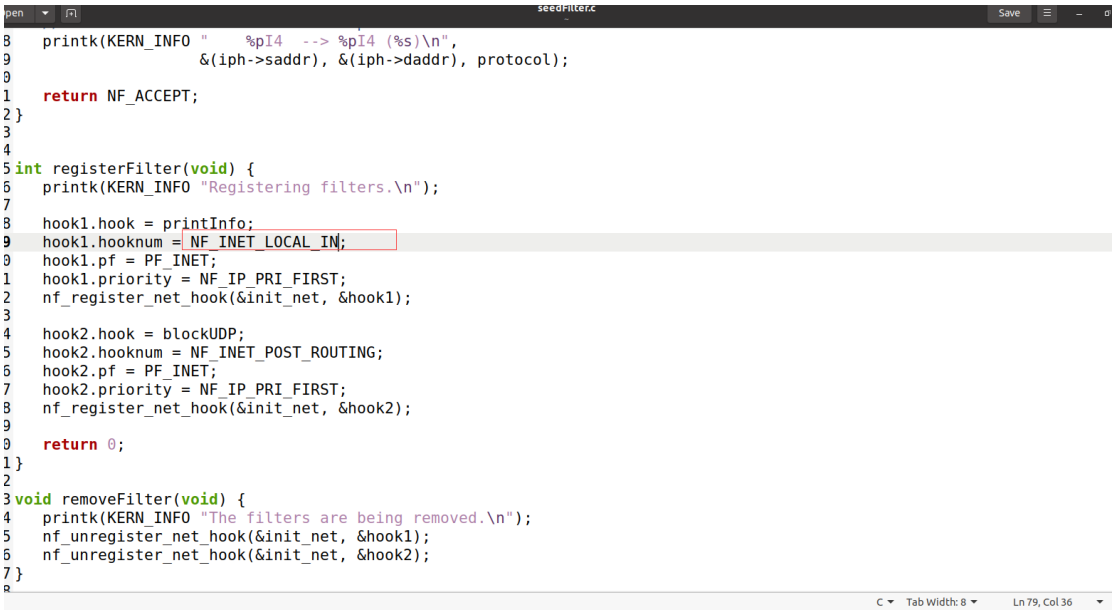
```
[ 8468.859274] *** Dropping 114.114.114.114 (UDP), port 53
[07/22/21]seed@VM:~/Desktop$ dig @114.114.114.114 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @114.114.114.114 www.example.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```




```
[07/22/21]seed@VM:~$ sudo rmmod seedFilter
[07/22/21]seed@VM:~$ lsmod | grep seedFilter
[07/22/21]seed@VM:~$
```

NF_INET_LOCAL_IN:




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

```
[07/22/21]seed@VM:~/Desktop$ dig @114.114.114.114 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @114.114.114.114 www.example.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

```
seed@VM: ~/Desktop
[ 8558.932208]     192.168.43.81  --> 192.168.43.1 (UDP)
[ 8558.979741] *** LOCAL_OUT
[ 8558.979746]     127.0.0.53  --> 127.0.0.1 (UDP)
[ 8559.028835] *** LOCAL_OUT
[ 8559.028840]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 8559.029357] *** LOCAL_OUT
[ 8559.029362]     192.168.43.81  --> 192.168.43.1 (UDP)
[ 8559.034460] *** LOCAL_OUT
[ 8559.034465]     127.0.0.53  --> 127.0.0.1 (UDP)
[ 8578.797869] The filters are being removed.
[ 8896.801839] Registering filters.
[ 8922.824327] *** PRE_ROUTING
[ 8922.824329]     127.0.0.1  --> 127.0.0.1 (UDP)
[ 8922.824609] *** Dropping 114.114.114.114 (UDP), port 53
[ 8927.826948] *** Dropping 114.114.114.114 (UDP), port 53
[ 8932.830937] *** Dropping 114.114.114.114 (UDP), port 53
[ 8995.873250] The filters are being removed.
[ 9103.429887] Registering filters.
[ 9133.103637] *** LOCAL_IN
[ 9133.103639]     127.0.0.1  --> 127.0.0.1 (UDP)
[ 9133.103932] *** Dropping 114.114.114.114 (UDP), port 53
[ 9138.110880] *** Dropping 114.114.114.114 (UDP), port 53
[ 9143.114880] *** Dropping 114.114.114.114 (UDP), port 53
```

```
[07/22/21]seed@VM:~$ sudo rmmod seedFilter
[07/22/21]seed@VM:~$ lsmod | grep seedFilter
[07/22/21]seed@VM:~$
```

NF_INET_FORWARD:

Open seedFilter.c Save

```
64      default:          protocol = "OTHER"; break;
65
66    }
67    // Print out the IP addresses and protocol
68    printk(KERN_INFO "    %pI4  --> %pI4 (%s)\n",
69                     &(iph->saddr), &(iph->daddr), protocol);
70
71    return NF_ACCEPT;
72 }
73
74
75 int registerFilter(void) {
76    printk(KERN_INFO "Registering filters.\n");
77
78    hook1.hook = printInfo;
79    hook1.hooknum = NF_INET_FORWARD;
80    hook1.pf = PF_INET;
81    hook1.priority = NF_IP_PRI_FIRST;
82    nf_register_net_hook(&init_net, &hook1);
83
84    hook2.hook = blockUDP;
85    hook2.hooknum = NF_INET_POST_ROUTING;
86    hook2.pf = PF_INET;
87    hook2.priority = NF_IP_PRI_FIRST;
88    nf_register_net_hook(&init_net, &hook2);
89
90    return 0;
91 }
92
93 void removeFilter(void) {
94    printk(KERN_INFO "The filters are being removed.\n");
```

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

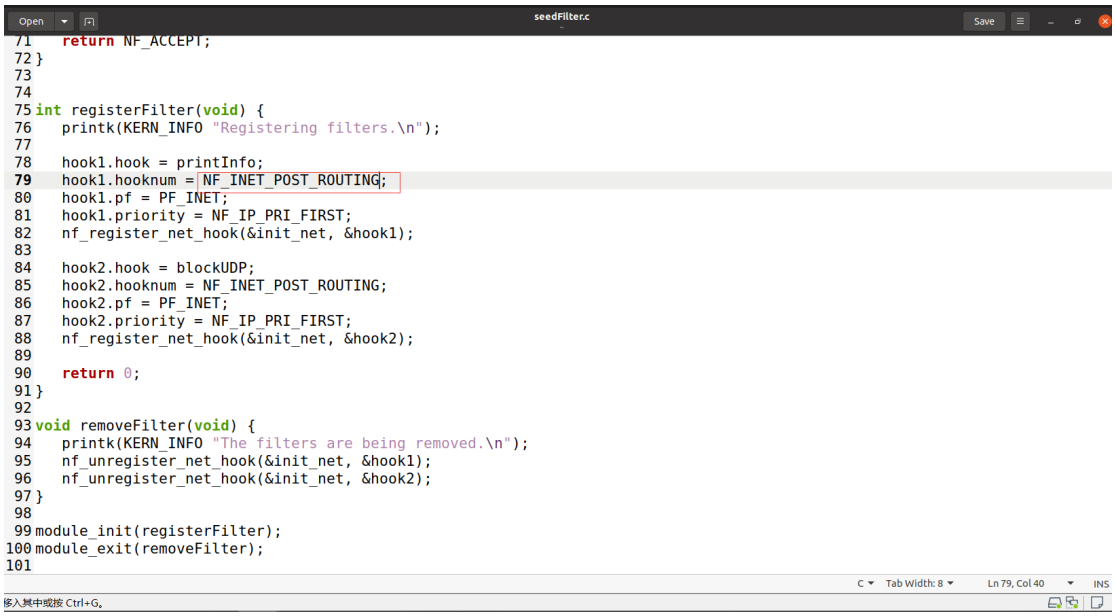
```
[07/22/21]seed@VM:~/Desktop$ dig @114.114.114.114 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @114.114.114.114 www.example.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

```
seed@VM: ~/Desktop
[ 9103.429887] Registering filters.
[ 9133.103637] *** LOCAL_IN
[ 9133.103639]     127.0.0.1  --> 127.0.0.1 (UDP)
[ 9133.103932] *** Dropping 114.114.114.114 (UDP), port 53
[ 9138.110880] *** Dropping 114.114.114.114 (UDP), port 53
[ 9143.114880] *** Dropping 114.114.114.114 (UDP), port 53
[ 9158.893419] *** LOCAL_IN
[ 9158.893421]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 9158.942115] *** LOCAL_IN
[ 9158.942142]     192.168.43.1  --> 192.168.43.81 (UDP)
[ 9158.942750] *** LOCAL_IN
[ 9158.942753]     127.0.0.53  --> 127.0.0.1 (UDP)
[ 9159.026563] *** LOCAL_IN
[ 9159.026566]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 9159.038940] *** LOCAL_IN
[ 9159.038945]     192.168.43.1  --> 192.168.43.81 (UDP)
[ 9159.039371] *** LOCAL_IN
[ 9159.039375]     127.0.0.53  --> 127.0.0.1 (UDP)
[ 9179.617574] The filters are being removed.
[ 9549.156504] Registering filters.
[ 9571.419106] *** Dropping 114.114.114.114 (UDP), port 53
[ 9576.419031] *** Dropping 114.114.114.114 (UDP), port 53
[ 9581.423107] *** Dropping 114.114.114.114 (UDP), port 53
[07/22/21]seed@VM:~/Desktop$
```

```
[07/22/21]seed@VM:~$ sudo rmmod seedFilter
[07/22/21]seed@VM:~$ lsmod | grep seedFilter
[07/22/21]seed@VM:~$
```

NF_INET_POST_ROUTING:

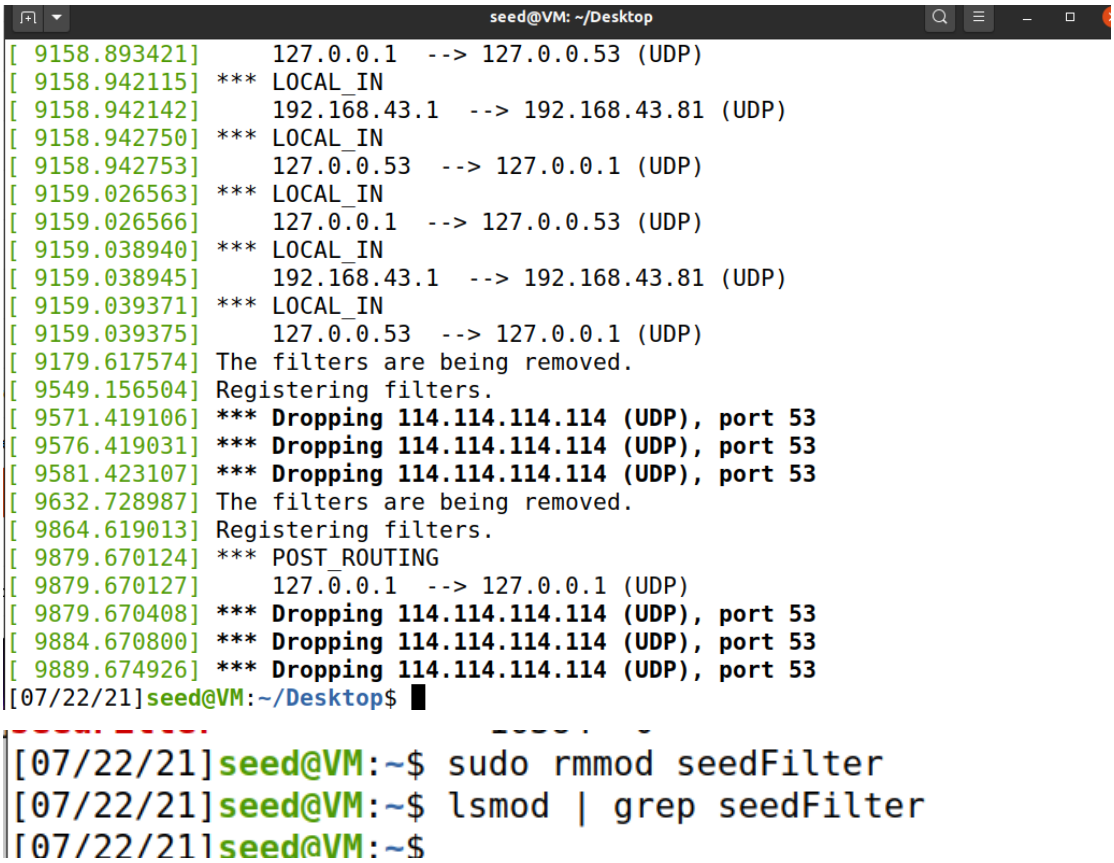



```
[07/22/21]seed@VM:~$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/seedFilter.mod.o
  LD [M]  /home/seed/seedFilter.ko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

```
[07/22/21]seed@VM:~/Desktop$ dig @114.114.114.114 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @114.114.114.114 www.example.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached

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



以上是将 printfo 函数挂在不同 hook 上的结果，根据实验结果，我们可以知：
NF_IP_LOCAL_OUT:在数据包以其方式离开主机之前。
NF_IP_POST_ROUTING:数据包离开主机并进入不同的网络之后。
NF_IP_PRE_ROUTING:在做出任何路由决策之前。
NF_IP_LOCAL_IN:在发送到网络堆栈之前。
NF_IP_FORWARD:向其他主机转发报文。

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

Task1.B.3：

该实验的目的是实现两个 hook 达到以下功能：（1）防止其他计算机 ping VM。（2）防止其他计算机 telnet 到 VM。telnet 是属于 tcp 的协议且端口号为 23，ping 是 ICMP 协议，通过其协议和端口则可进行过滤。代码如下：

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

   iph = ip_hdr(skb);
   tcph = (void *)iph+iph->ihl*4;

   if(iph->protocol == IPPROTO_TCP && tcph->dest == htons(23))
   {
        printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d\n",((unsigned char *)&iph->daddr)[0],((unsigned char
 *)&iph->daddr)[1],((unsigned char *)&iph->daddr)[2],((unsigned char *)&iph->daddr)[3]);
        return NF_DROP;
        }
    else
    {
    return NF_ACCEPT;
    }


}


unsigned int pingFilter(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_INFO "Dropping ping packet to %d.%d.%d.%d\n",((unsigned char *)&iph->daddr)[0],((unsigned char
 *)&iph->daddr)[1],((unsigned char *)&iph->daddr)[2],((unsigned char *)&iph->daddr)[3]);
        return NF_DROP;
        }
    else
    {
    return NF_ACCEPT;
    }
}


   hook3.hook = telnetFilter;
   hook3.hooknum = NF_INET_LOCAL_IN;
   hook3.pf = PF_INET;
   hook3.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook3);

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

   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);
   nf_unregister_net_hook(&init_net, &hook3);
   nf_unregister_net_hook(&init_net, &hook4);
}
```

编译并加载内核模块。

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

尝试 ping 和 telnet 虚拟机，发现都失败了。

```
seed@VM: ~/Desktop
root@61963641596a:/# telnet 10.9.0.1
Trying 10.9.0.1...
telnet: Unable to connect to remote host: Connection timed out
root@61963641596a:/# 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 ---
546 packets transmitted, 0 received, 100% packet loss, time 558089ms
```

实验做完后，要将模块移除。

```
[07/22/21]seed@VM:~$ sudo rmmod seedFilter
[07/22/21]seed@VM:~$ lsmod | grep seedFilter
[07/22/21]seed@VM:~$
```

移除模块后，telnet 和 ping 虚拟机就成功了，说明实现的 hook 奏效，起到了作用。

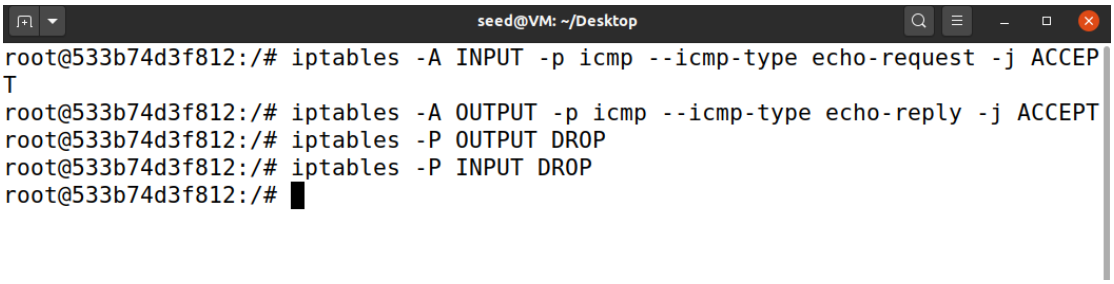
```
root@61963641596a:/# telnet 10.9.0.1
Trying 10.9.0.1...
Connected to 10.9.0.1.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
VM login:
```

```
root@61963641596a:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
64 bytes from 10.9.0.1: icmp_seq=1 ttl=64 time=0.095 ms
64 bytes from 10.9.0.1: icmp_seq=2 ttl=64 time=0.129 ms
64 bytes from 10.9.0.1: icmp_seq=3 ttl=64 time=0.118 ms
64 bytes from 10.9.0.1: icmp_seq=4 ttl=64 time=0.134 ms
```

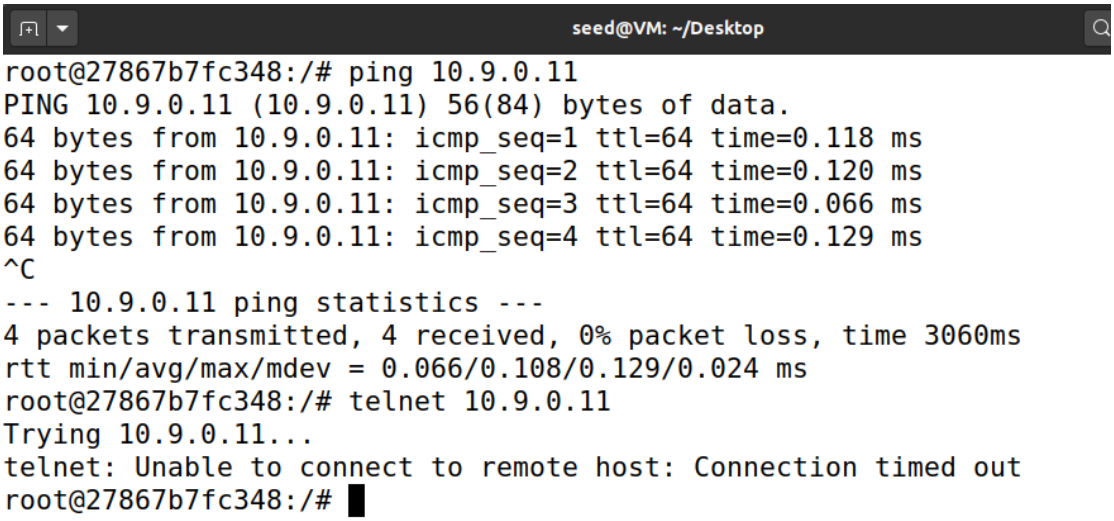
Task2

Task2.A:

这个任务的实验目的是将通过 iptables 设置规则防止外部计算机访问路由器（ping 除外）。根据手册给出，我们将在路由器容器中执行如下命令:



从 10.9.0.5ping 路由器可以 ping 通，但是尝试其他访问，如 telnet，发现被防火墙阻塞了。



实验完成之后，要清除掉写入的规则。

```
root@533b74d3f812:/# iptables -F
root@533b74d3f812:/# iptables -P OUTPUT ACCEPT
root@533b74d3f812:/# iptables -P INPUT ACCEPT
root@533b74d3f812:/#
```

Task2.B:

在这个任务中，我们将在路由器上设置防火墙规则来保护内部网络 192.168.60.0/24。为此，我们需要使用 FORWARD 链。信息包在输入和输出链中的方向很清楚:信息包要么进入(输入)，要么出(输出)。这对于 FORWARD 链是不正确的，因为它是双向的:进入内部网络或出去到外部网络的数据包都要经过这个链。为了指定方向，我们可以使用“-i xyz”(来自 xyz 接口)和/或“-o xyz”(来自 xyz 接口)添加接口选项。内部网络和外部网络的接口不一样。
设置规则如下:

```
root@533b74d3f812:/volumes# iptables -A FORWARD -i eth0 -o eth1 -p icmp --icmp-t
ype echo-request -j DROP
root@533b74d3f812:/volumes# iptables -A FORWARD -s 10.9.0.11 -p icmp --icmp-type
 echo-request -j ACCEPT
root@533b74d3f812:/volumes# iptables -A FORWARD -o eth0 -p icmp --icmp-type echo
-reply -j ACCEPT
root@533b74d3f812:/volumes# iptables -A FORWARD -i eth1 -o eth0 -p icmp --icmp-t
ype echo-request -j ACCEPT
root@533b74d3f812:/volumes# iptables -A FORWARD -i eth0 -o eth1 -p icmp --icmp-t
ype echo-reply -j ACCEPT
root@533b74d3f812:/volumes# iptables -P FORWARD DROP
root@533b74d3f812:/volumes#
```

外部主机不可以 ping 内网，但是可以 ping 路由器。

```
seed@VM: ~/Desktop
root@27867b7fc348:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
54 packets transmitted, 0 received, 100% packet loss, time 54267ms

root@27867b7fc348:/# 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=0.062 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.093 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.056 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.054 ms
^C
--- 10.9.0.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3075ms
rtt min/avg/max/mdev = 0.054/0.066/0.093/0.015 ms
root@27867b7fc348:/#
```

内网可以 ping 外网。

```
seed@VM: ~/Desktop
[07/22/21]seed@VM:~/Desktop$ dockps
d35ada53686e  host2-192.168.60.6
7e954ad31267  host1-192.168.60.5
a2799e117214  host3-192.168.60.7
27867b7fc348  hostA-10.9.0.5
533b74d3f812  seed-router
[07/22/21]seed@VM:~/Desktop$ docksh d3
root@d35ada53686e:/# 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=0.121 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.212 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.152 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.111 ms
```

其他的在内外网之间的包会被阻塞，尝试在内网 telnet 外网主机和在外网 telnet 内网主机。

```
root@27867b7fc348:/# telnet 192.168.60.5
Trying 192.168.60.5...
telnet: Unable to connect to remote host: Connection timed out
root@27867b7fc348:/#
```

```
root@d35ada53686e:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@d35ada53686e:/#
```

结果均被阻塞。实验达到了预期的结果。

```
root@533b74d3f812:/volumes# iptables -P FORWARD ACCEPT
root@533b74d3f812:/volumes# iptables -F
root@533b74d3f812:/volumes#
```

实验完成后需要清空规则。

Task2.C:

在这个实验中，我们需要保护内部网络的服务器，需要实现的目标，通过规则和运行结果展示，规则如下：

```
seed@VM: ~/Desktop
root@533b74d3f812:/volumes# iptables -A FORWARD -i eth0 -o eth1 -d 192.168.60.5
-p tcp --dport 23 -j ACCEPT
root@533b74d3f812:/volumes# iptables -A FORWARD -i eth1 -o eth0 -s 192.168.60.5
-p tcp --sport 23 -j ACCEPT
root@533b74d3f812:/volumes# iptables -A FORWARD -i eth0 -o eth1 -j DROP
root@533b74d3f812:/volumes# iptables -A FORWARD -i eth1 -o eth1 -j ACCEPT
root@533b74d3f812:/volumes# iptables -A FORWARD -i eth1 -o eth0 -j DROP
root@533b74d3f812:/volumes# iptables -P FORWARD DROP
root@533b74d3f812:/volumes#
```

外网的主机只能 telnet 到内网的 192.168.60.5 这台主机，其他主机不行。

```
root@27867b7fc348:/# telnet 192.168.60.6
Trying 192.168.60.6...
telnet: Unable to connect to remote host: Connection timed out
root@27867b7fc348:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.

Ubuntu 20.04.1 LTS

7e954ad31267 login: ^[^A
```

外部主机无法访问内部主机。

```
root@27867b7fc348:/# ping 192.168.60.6
PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data.
^C
--- 192.168.60.6 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6137ms

root@27867b7fc348:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

^C
--- 192.168.60.5 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6122ms
```

内部主机可以访问所有内部服务器。

```
seed@VM: ~/Desktop
root@d35ada53686e:/# ping 192.168.60.6
PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data.
64 bytes from 192.168.60.6: icmp_seq=1 ttl=64 time=0.036 ms
64 bytes from 192.168.60.6: icmp_seq=2 ttl=64 time=0.082 ms
64 bytes from 192.168.60.6: icmp_seq=3 ttl=64 time=0.051 ms
64 bytes from 192.168.60.6: icmp_seq=4 ttl=64 time=0.077 ms
^C
--- 192.168.60.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.036/0.061/0.082/0.018 ms
root@d35ada53686e:/# ping 192.168.60.7
PING 192.168.60.7 (192.168.60.7) 56(84) bytes of data.
64 bytes from 192.168.60.7: icmp_seq=1 ttl=64 time=0.121 ms
64 bytes from 192.168.60.7: icmp_seq=2 ttl=64 time=0.114 ms
64 bytes from 192.168.60.7: icmp_seq=3 ttl=64 time=0.125 ms
64 bytes from 192.168.60.7: icmp_seq=4 ttl=64 time=0.126 ms
^C
--- 192.168.60.7 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3075ms
rtt min/avg/max/mdev = 0.114/0.121/0.126/0.004 ms
root@d35ada53686e:/#
```

内部主机无法访问外部主机。

```
root@d35ada53686e:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
^C
--- 10.9.0.5 ping statistics ---
45 packets transmitted, 0 received, 100% packet loss, time 45047ms

root@d35ada53686e:/#
```

实验结果符合预期目标。

最后实验结束后需要清空规则。

```
root@533b74d3f812:/volumes# iptables -P FORWARD ACCEPT
root@533b74d3f812:/volumes# iptables -F
root@533b74d3f812:/volumes#
```

Task3

Task3.A:

执行 conntrack -L 命令可以查看路由器上的连接跟踪信息。

```
root@225e6e6e2ae8:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
root@225e6e6e2ae8:/#
```

ICMP 实验:在 10.9.0.5 上 ping 192.168.60.5 后查看路由器上的连接跟踪信息。

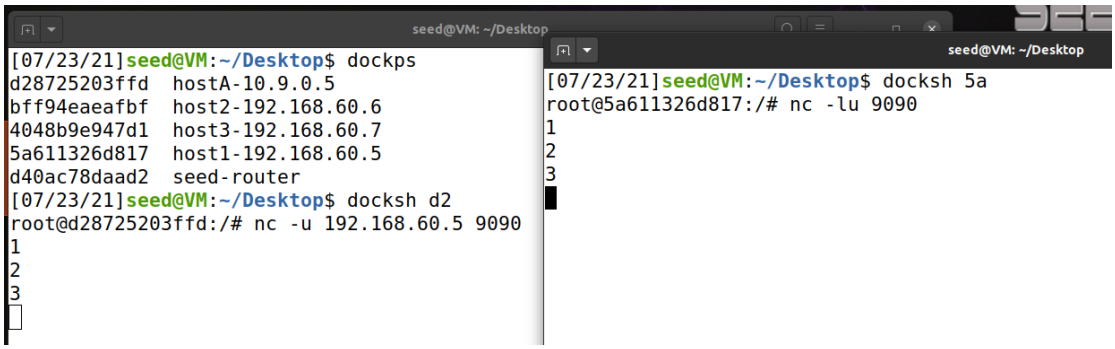
```
root@48f3bca9239d:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.079 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.110 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.152 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.162 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.150 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.115 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.071 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.146 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.097 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.150 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.072 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.115 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.154 ms
```

```
root@225e6e6e2ae8:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=31 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=31
 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@225e6e6e2ae8:/#
```

```
root@225e6e6e2ae8:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=31 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=31
 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@225e6e6e2ae8:/# conntrack -L
icmp     1 27 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=31 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=31
 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@225e6e6e2ae8:/# conntrack -L
icmp     1 25 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=31 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=31
 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@225e6e6e2ae8:/# conntrack -L
icmp     1 16 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=31 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=31
 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@225e6e6e2ae8:/# conntrack -L
icmp     1 7 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=31 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=31
mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@225e6e6e2ae8:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
```

可以观察到持续时间为 30s。

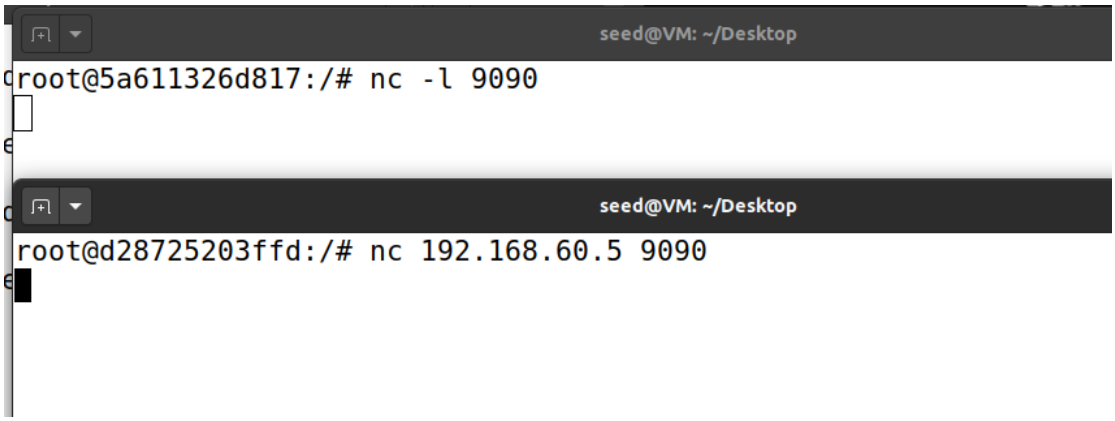
UDP 实验：在 192.168.60.5 上开一个 UDP 服务器，在 10.9.0.5 上发送 UDP 报文，然后查看路由器上的连接跟踪信息。




```
root@d40ac78daad2:/# conntrack -L
udp      17 29 src=10.9.0.5 dst=192.168.60.5 sport=40514 dport=9090 [UNREPLIED]
src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=40514 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

可以观察到持续时间是 30s。

TCP 实验：在 192.168.60.5 上开一个 TCP 服务器，在 10.9.0.5 上发送 TCP 报文，然后查看路由器上的连接跟踪信息。




```
root@d40ac78daad2:/# conntrack -L
tcp      6 94 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=50758 dport=9090 src
=192.168.60.5 dst=10.9.0.5 sport=9090 dport=50758 [ASSURED] mark=0 use=1
tcp      6 431999 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=50762 dport=90
90 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=50762 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
root@d40ac78daad2:/#
```

可以观察到持续时间是 432000s 。

Task3.B:

该实验的要求是 Task 2.C 中的防火墙规则，但这次，我们将添加一个允许内部主机访问任何外部服务器的规则(这在 Task 2.C 中是不允许的)。

更改后的规则如下：

```
seed@VM: ~/Desktop
root@2638f975b089:/# iptables -A FORWARD -p tcp -m conntrack --ctstate ESTABLISH
ED,RELATED -j ACCEPT
root@2638f975b089:/# iptables -A FORWARD -i eth0 -o eth1 -d 192.168.60.5 -p tcp
--dport 23 --syn -m conntrack --ctstate NEW -j ACCEPT
root@2638f975b089:/# iptables -A FORWARD -i eth0 -o eth1 -p tcp --syn -m conntra
ck --ctstate NEW -j DROP
root@2638f975b089:/# iptables -A FORWARD -i eth1 -o eth1 -p tcp --syn -m conntra
ck --ctstate NEW -j ACCEPT
root@2638f975b089:/# iptables -A FORWARD -i eth1 -o eth0 -p tcp --syn -m conntra
ck --ctstate NEW -j ACCEPT
root@2638f975b089:/# iptables -P FORWARD DROP
root@2638f975b089:/#
```

可以看到外部主机可以且仅可以 telnet 到内网的 192.168.60.5 主机。

```
[07/23/21]seed@VM:~/Desktop$ docksh 01
root@015b108301f1:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
37416bc977e1 login: ^CConnection closed by foreign host.
root@015b108301f1:/# telnet 192.168.60.6
Trying 192.168.60.6...
telnet: Unable to connect to remote host: Connection timed out
```

内网的主机可以访问任意内网或者外网的主机。

```
seed@VM: ~/Desktop
root@37416bc977e1:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
015b108301f1 login: ^CConnection closed by foreign host.
root@37416bc977e1:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
d9550bdb5710 login: Connection closed by foreign host.
root@37416bc977e1:/#
```

实验完成后需要将定义的规则进行清除。

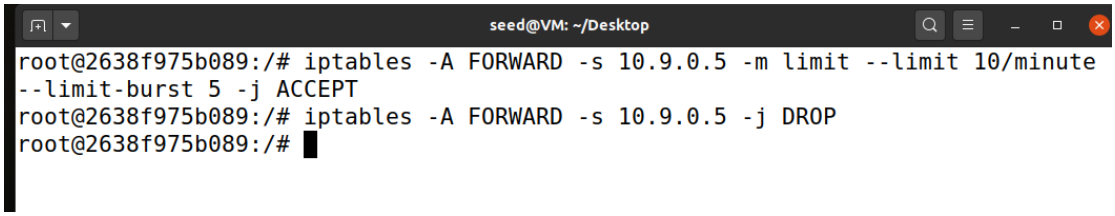
```
root@2638f975b089:/# iptables -P FORWARD ACCEPT
root@2638f975b089:/# iptables -F
root@2638f975b089:/#
```

上述两种机制中，如果不用连接跟踪机制，则需要设置防火墙可以转发内网访问外网 23

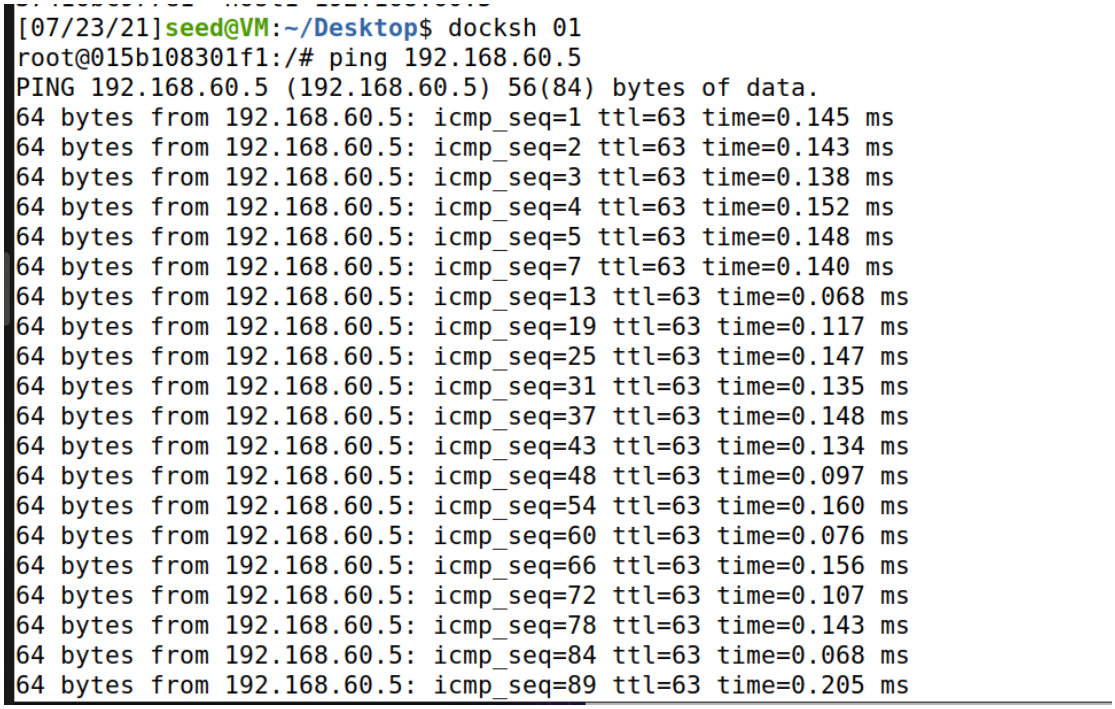
端口的规则，这样比设置有上下文关系的跟踪机制麻烦，并且即使打开了，也可能不能防止三次握手以外的恶意连接，但是设置了跟踪机制就可以只考虑到三次握手的情况，其余包均丢弃。

Task4：

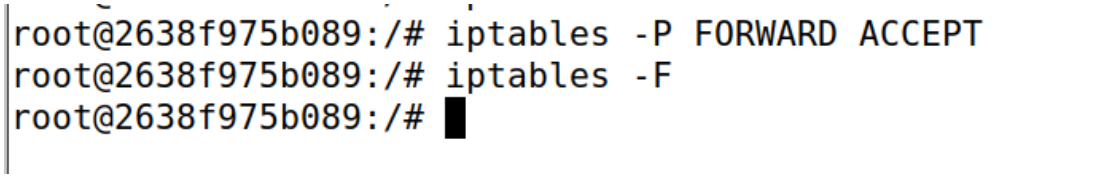
本次任务的目的是通过 limit 模块来限制通过防火墙的数据包数量，根据文档中给出的规则进行实验，规则如下：



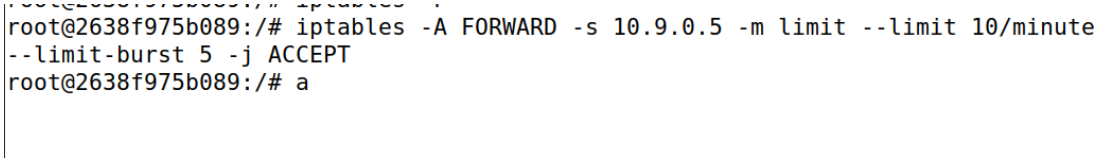
然后在 10.9.0.5ping192.168.60.5，发现有丢包现象，ICMP 的请求和响应存在时间间隔。



实验结束后需要清除规则。



然后尝试没有第二条规则的情况下，会发生什么。



可以发现没有第二个规则的时候，可以 ping 成功，并且不存在丢包的现象。

```
root@015b108301f1:/# 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=0.071 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.159 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.169 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.106 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.156 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.153 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.161 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=0.156 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.068 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.149 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.157 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.094 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.151 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.067 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.212 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.130 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.128 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.152 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.105 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.072 ms
64 bytes from 192.168.60.5: icmp_seq=22 ttl=63 time=0.182 ms
```

实验完成后需要清空规则。

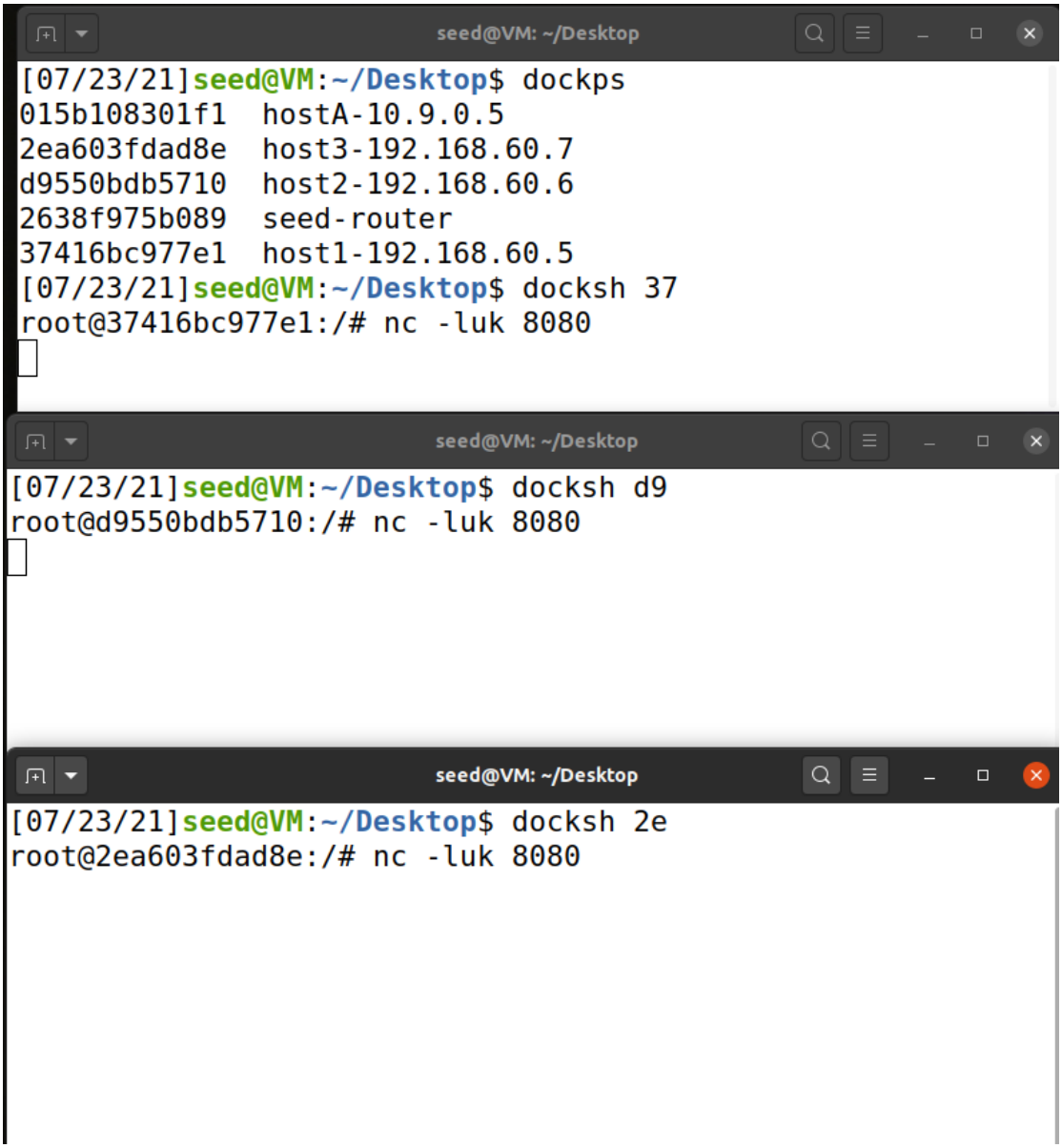
```
root@2638f975b089:/# iptables -P FORWARD ACCEPT
root@2638f975b089:/# iptables -F
root@2638f975b089:/#
```

分析第二条规则有与无所造成的实验结果不同的原因，是因为第一条设置了最大匹配率，但没有第二条默认情况下将 192.168.60.5 的回复都丢掉的话，就不会达到我们预想的结果，即会被默认接收。

Task5：

采用 nth mode：

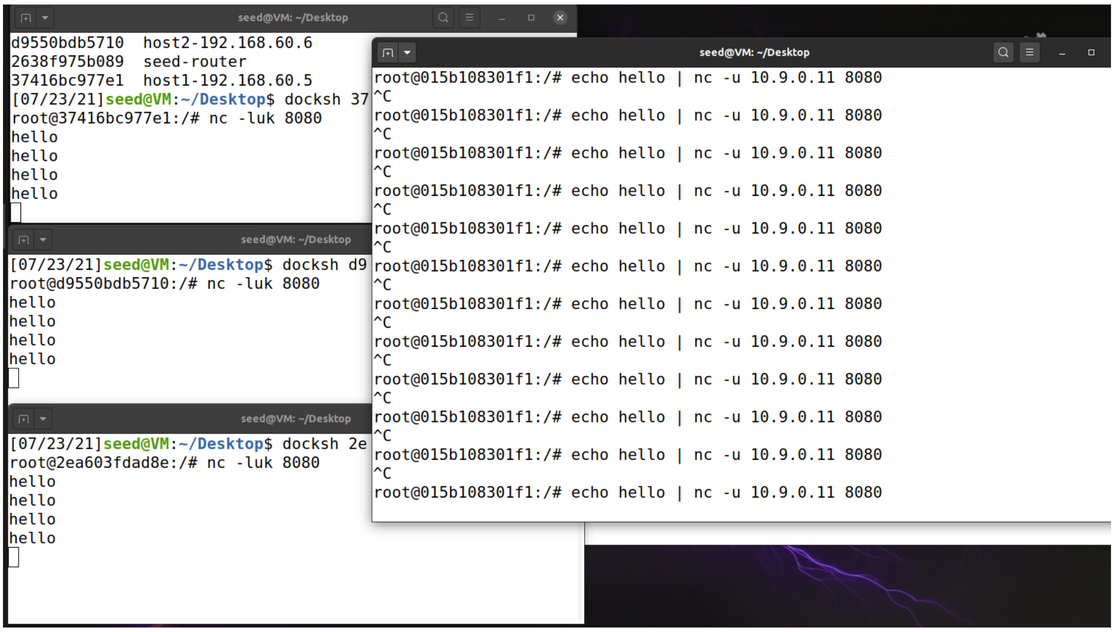
首先在 192.168.60.5,192.168.60.6,192.168.60.7 三个主机上的 8080 端口开一个 UDP 服务器。



通过文档提供的 nth mode 规则进行负载均衡，这里需要注意的是，每三个报文中一个交给 192.168.60.5:8080，则剩下的每两个中有一个交给 192.168.60.6:8080，其余剩余的交给 192.168.60.7:8080，则匹配规则如下：

```
root@2638f975b089:/# iptables -t nat -A PR
EROUTING -p udp --dport 8080 -m statistic
--mode nth --every 3 --packet 0 -j DNAT --
to-destination 192.168.60.5:8080
root@2638f975b089:/# iptables -t nat -A PR
EROUTING -p udp --dport 8080 -m statisroot
@2638f975b089:/# iptables -t nat -A PREROU
root@2638f975b089:/# iptables -t nat -A PR
EROUTING -p udp --dport 8080 -j DNAT --to-
destination 192.168.60.7:8080-
root@2638f975b089:/#
```

在 10.9.0.5 上我们键入足够多的 echo hello | nc -u 10.9.0.11 8080，可以发现三个服务器上的 hello 数量较为均衡，即各为三分之一。

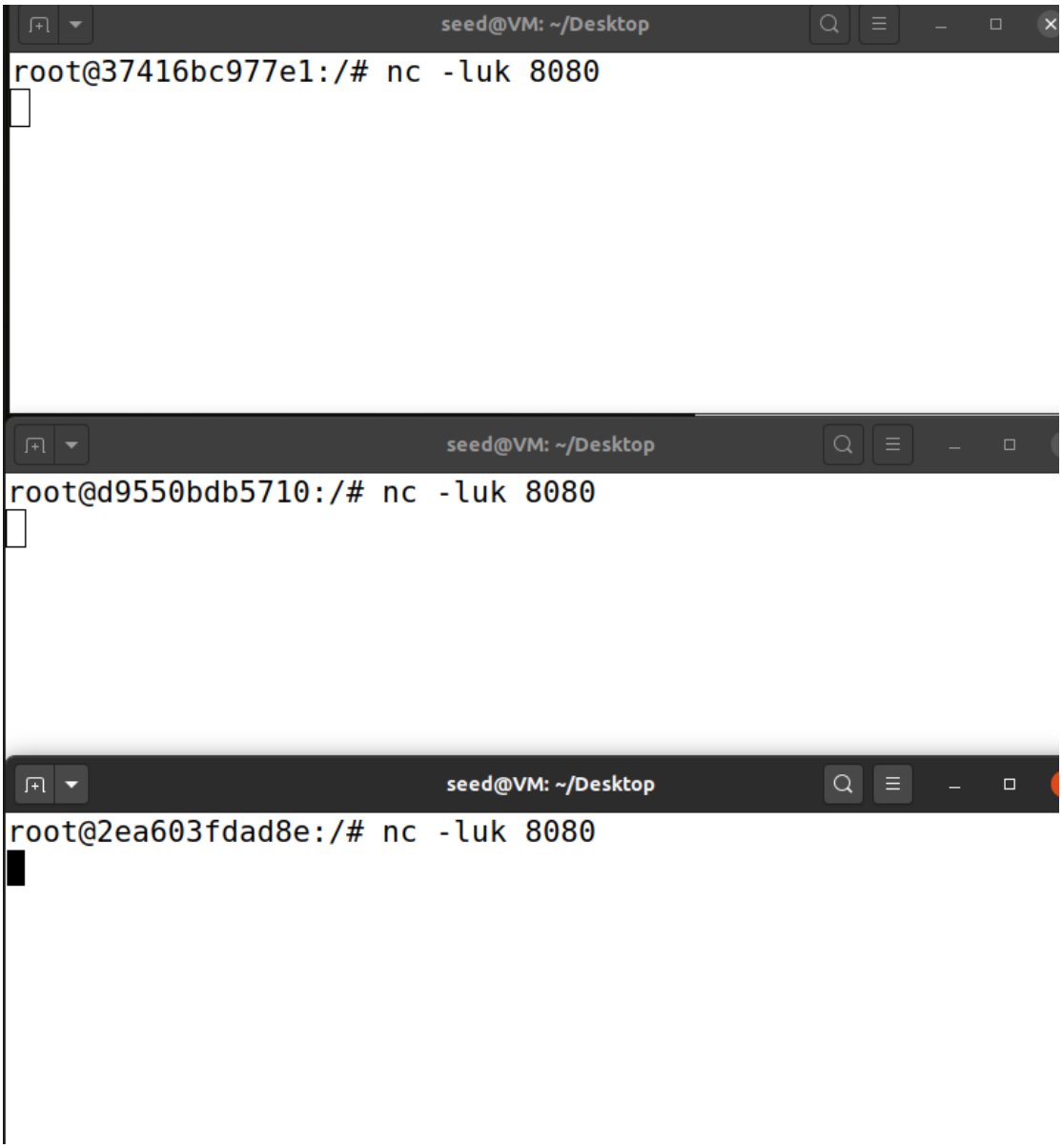


实验完成后需要清除规则。

```
root@2638f975b089:/# iptables -t nat -P PREROUTING ACCEPT
root@2638f975b089:/# iptables -F
root@2638f975b089:/#
```

采用 random mode：

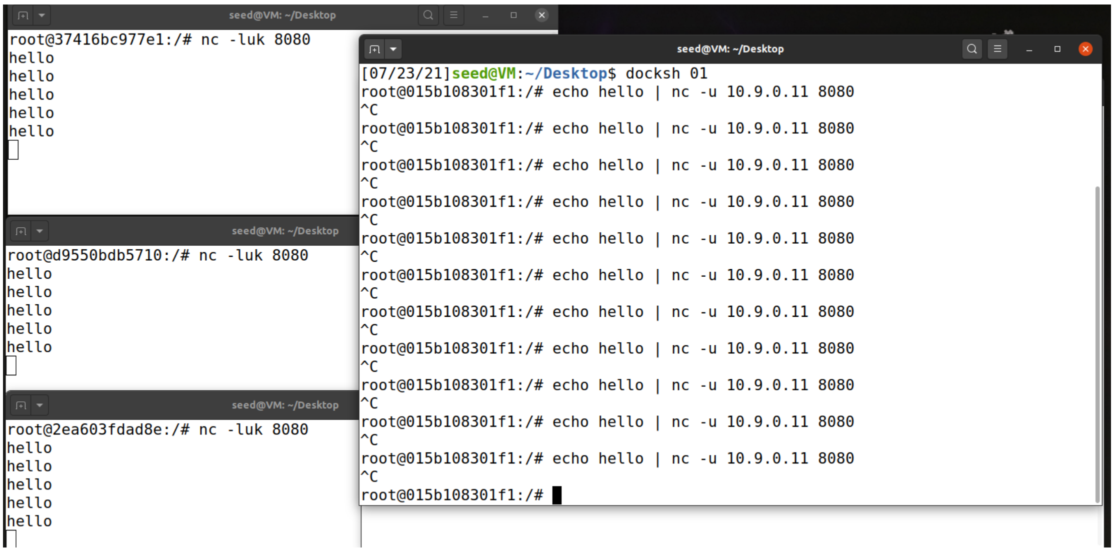
首先在 192.168.60.5,192.168.60.6,192.168.60.7 三个主机上的 8080 端口开一个 UDP 服务器。



通过文档提供的 random mode 规则进行负载均衡，这里需要注意的是，第一个交给 192.168.60.5:8080 的概率是 0.33，则剩下的 0.67 交给 192.168.60.6:8080 应该为 0.5 才能保持概率也是 0.33，其余剩余的交给 192.168.60.7:8080，则匹配规则如下：

```
root@2638f975b089:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode random --probability 0.33 -j DNAT --to-destination 192.168.60.5:8080
root@2638f975b089:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode random --probability 0.5 -j DNAT --to-destination 192.168.60.6:8080
root@2638f975b089:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -j DNAT -
-to-destination 192.168.60.7:8080
root@2638f975b089:/#
```

在 10.9.0.5 上我们键入足够多的 echo hello | nc -u 10.9.0.11 8080，可以发现三个服务器上的 hello 数量较为均衡，即各为三分之一。



实验完成之后需要清除规则。

```
root@2638f975b089:/# iptables -t nat -P PREROUTING ACCEPT
root@2638f975b089:/# iptables -F
```

总结:

本次的实验中，我学会了利用 LKM 和 netfilter 实现防火墙，并对报文进行一些基本的过滤，同时学会了使用 iptables 实现好的功能完成一些更复杂的过滤，通过连接跟踪还可以搭建状态防火墙。这次的实验中让我对防火墙的基本原理和应用有了更深的认识，在实验中规则的实现的一些语法需要注意，容易出现错误。