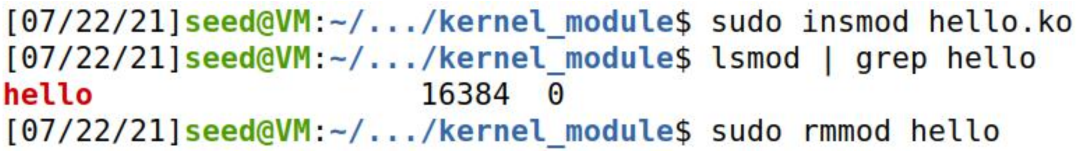
网络空间安全实训实验报告

纪盛谦 57118218 2021-7-22

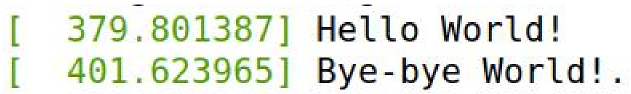
Task1 : Implementing a Simple Firewall

Task1.A : Implement a Simple Kernel Module

将其 make 后，加载、移除 hello 后，可以看到有记录。



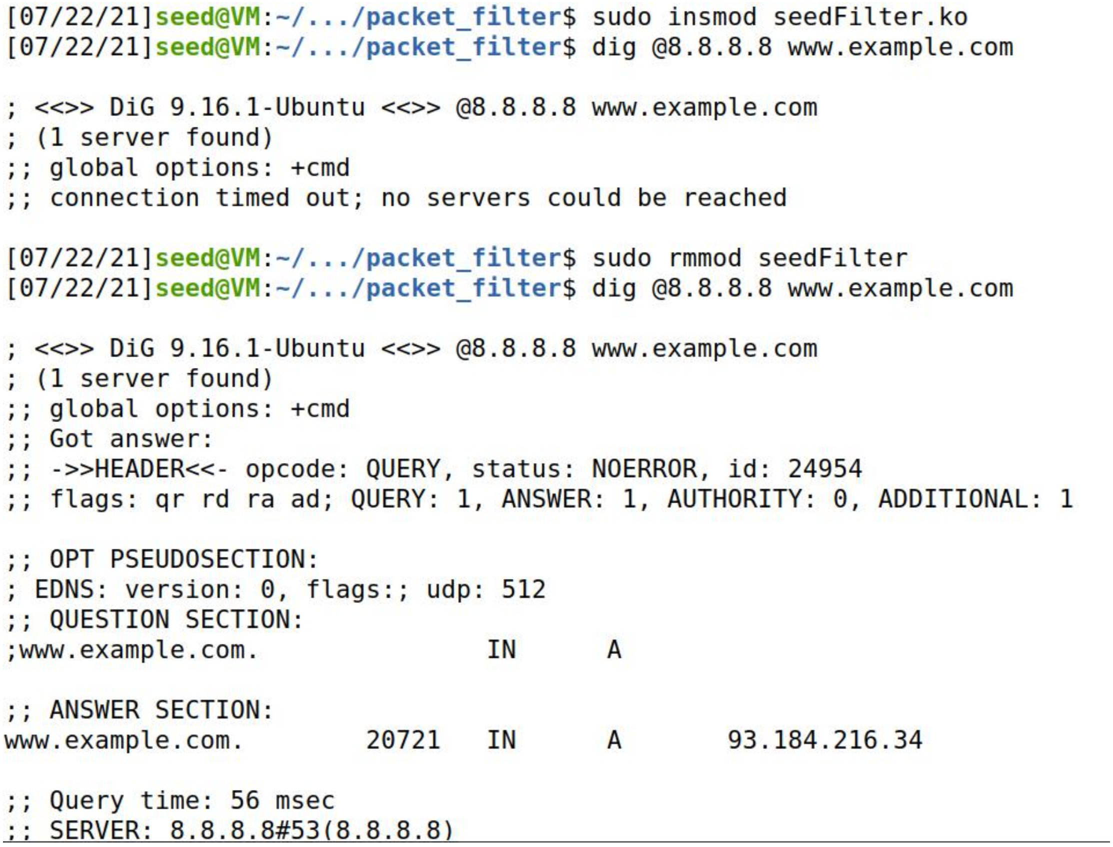
```
[07/22/21]seed@VM:~/.../kernel_module$ sudo insmod hello.ko
[07/22/21]seed@VM:~/.../kernel_module$ lsmod | grep hello
hello                  16384  0
[07/22/21]seed@VM:~/.../kernel_module$ sudo rmmod hello
```



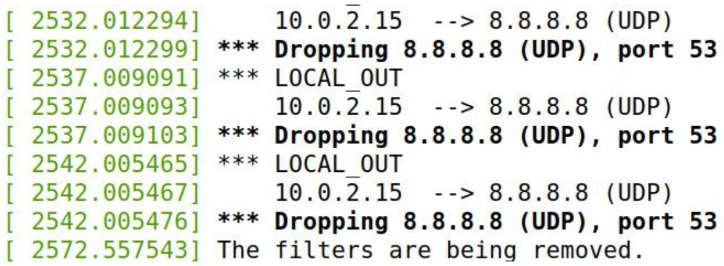
```
[  379.801387] Hello World!
[  401.623965] Bye-bye World!.
```

Task 1.B: Implement a Simple Firewall Using Netfilter

（1）挂载程序后，到 8.8.8.8/53 的报文被拦截，移除检测程序后则可以发出该报文且有回复。



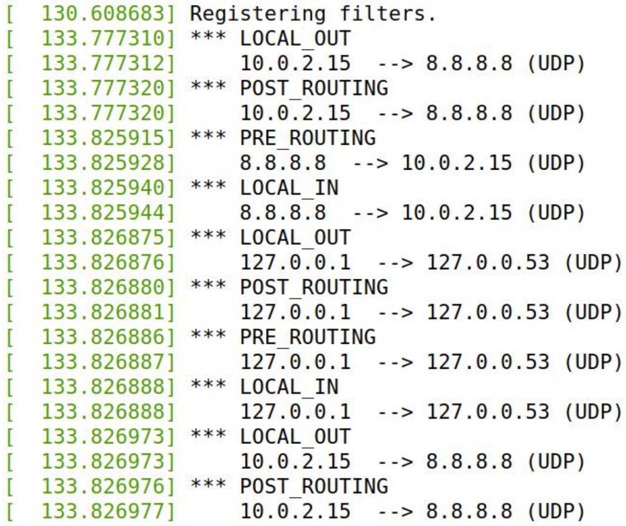
```
[07/22/21]seed@VM:~/.../packet_filter$ sudo insmod seedFilter.ko
[07/22/21]seed@VM:~/.../packet_filter$ dig @8.8.8.8 www.example.com

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

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

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24954
;; 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.        20721   IN      A       93.184.216.34

;; Query time: 56 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
```



```
[ 2532.012294]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 2532.012299] *** Dropping 8.8.8.8 (UDP), port 53
[ 2537.009091] *** LOCAL_OUT
[ 2537.009093]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 2537.009103] *** Dropping 8.8.8.8 (UDP), port 53
[ 2542.005465] *** LOCAL_OUT
[ 2542.005467]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 2542.005476] *** Dropping 8.8.8.8 (UDP), port 53
[ 2572.557543] The filters are being removed.
```

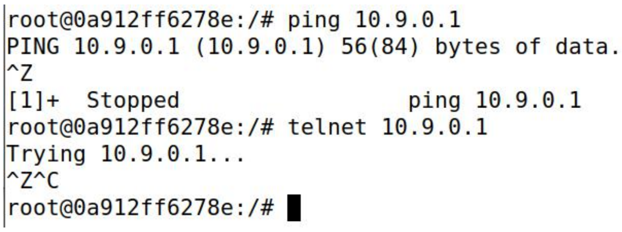
（2）通过实验可以发现数据报从进入系统，进行 IP 校验以后，首先经过第一个 HOOK 函数 NF_IP_PRE_ROUTING 进行处理；然后就进入路由代码，其决定该数据报是需要转发还是发给本机的；若该数据报是发被本机的，则该数据经过 HOOK 函数 NF_IP_LOCAL_IN 处理以后然后传递给上层协议；若该数据报应该被转发则它被 NF_IP_FORWARD 处理；经过转发的数据报经过最后一个 HOOK 函数 NF_IP_POST_ROUTING 处理以后，再传输到网络上。

本地产生的数据经过 HOOK 函数 NF_IP_LOCAL_OUT 处理后，进行路由选择处理，然后经过 NF_IP_POST_ROUTING 处理后发送出去。



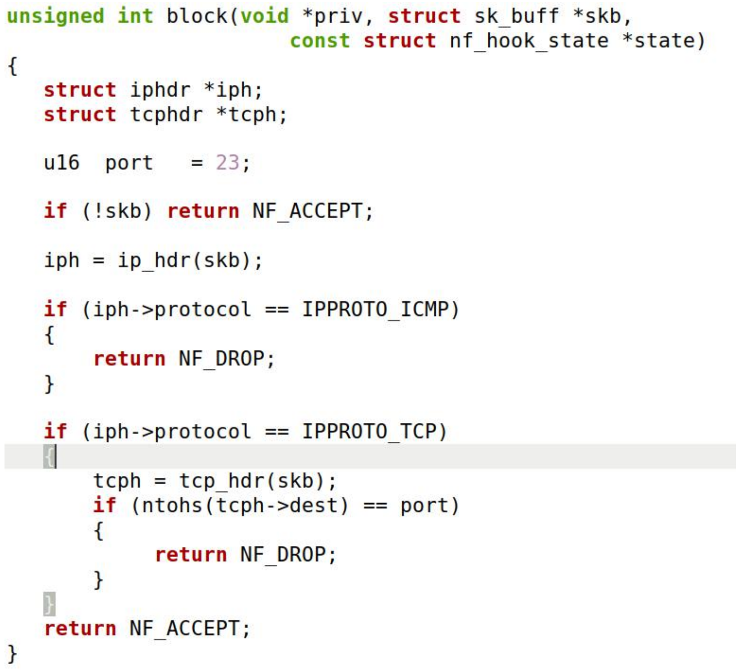
```
[  130.608683] Registering filters.
[  133.777310] *** LOCAL_OUT
[  133.777312]     10.0.2.15  --> 8.8.8.8 (UDP)
[  133.777320] *** POST_ROUTING
[  133.777320]     10.0.2.15  --> 8.8.8.8 (UDP)
[  133.825915] *** PRE_ROUTING
[  133.825928]     8.8.8.8  --> 10.0.2.15 (UDP)
[  133.825940] *** LOCAL_IN
[  133.825944]     8.8.8.8  --> 10.0.2.15 (UDP)
[  133.826875] *** LOCAL_OUT
[  133.826876]     127.0.0.1  --> 127.0.0.53 (UDP)
[  133.826880] *** POST_ROUTING
[  133.826881]     127.0.0.1  --> 127.0.0.53 (UDP)
[  133.826886] *** PRE_ROUTING
[  133.826887]     127.0.0.1  --> 127.0.0.53 (UDP)
[  133.826888] *** LOCAL_IN
[  133.826888]     127.0.0.1  --> 127.0.0.53 (UDP)
[  133.826973] *** LOCAL_OUT
[  133.826973]     10.0.2.15  --> 8.8.8.8 (UDP)
[  133.826976] *** POST_ROUTING
[  133.826977]     10.0.2.15  --> 8.8.8.8 (UDP)
```

（3）选用 NF_IP_LOCAL_IN 这个 hook，最终实验结果如下，从 10.9.0.5 向 10.9.0.1ping 和 telnet 都不能通。



```
root@0a912ff6278e:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
^Z
[1]+  Stopped                 ping 10.9.0.1
root@0a912ff6278e:/# telnet 10.9.0.1
Trying 10.9.0.1...
^Z^C
root@0a912ff6278e:/# 
```

其中处理函数代码如下

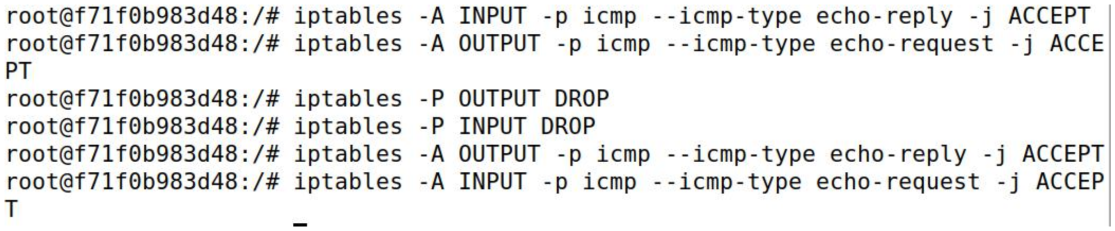


```
unsigned int block(void *priv, struct sk_buff *skb,
                       const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct tcphdr *tcph;

   u16  port   = 23;

   if (!skb) return NF_ACCEPT;

   iph = ip_hdr(skb);

   if (iph->protocol == IPPROTO_ICMP)
   {
       return NF_DROP;
   }

   if (iph->protocol == IPPROTO_TCP)
   {
       tcph = tcp_hdr(skb);
       if (ntohs(tcph->dest) == port)
       {
            return NF_DROP;
       }
   }
   return NF_ACCEPT;
}
```

Task2: Experimenting with Stateless Firewall Rules

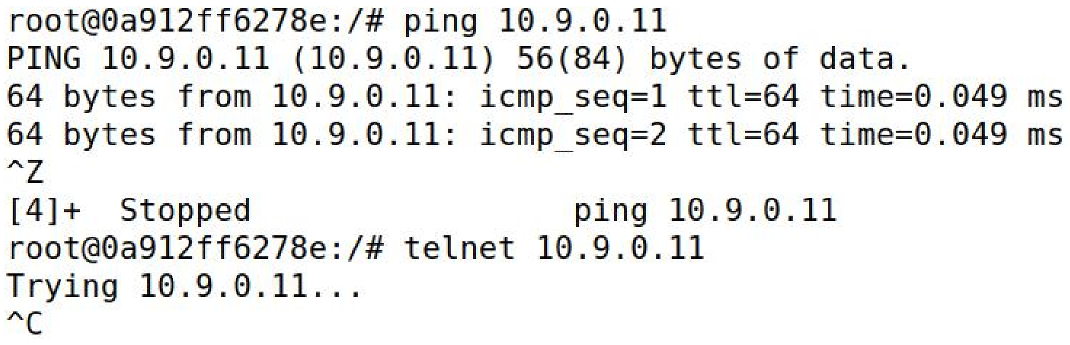
Task2.A：Protecting the Router

在 router 上设置 iptable，如果只安装手册的指令是不能达到效果的，只有在 INPUT 和 OUTPUT 都设置通过 echo-request 和 echo-reply 才可以。



```
root@f71f0b983d48:/# iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
root@f71f0b983d48:/# iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCE
PT
root@f71f0b983d48:/# iptables -P OUTPUT DROP
root@f71f0b983d48:/# iptables -P INPUT DROP
root@f71f0b983d48:/# iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
root@f71f0b983d48:/# iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEP
T
```

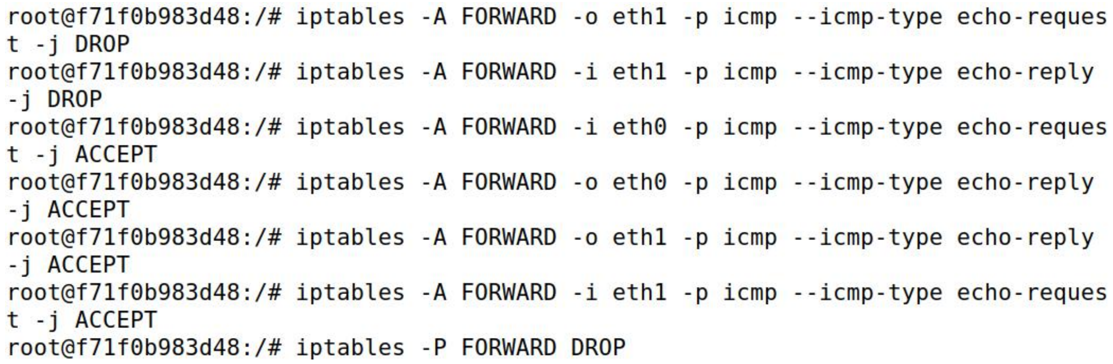
最终从 10.9.0.5 向路由 ping 和 telnet 效果如下。



```
root@0a912ff6278e:/# 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=0.049 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.049 ms
^Z
[4]+  Stopped                 ping 10.9.0.11
root@0a912ff6278e:/# telnet 10.9.0.11
Trying 10.9.0.11...
^C
```

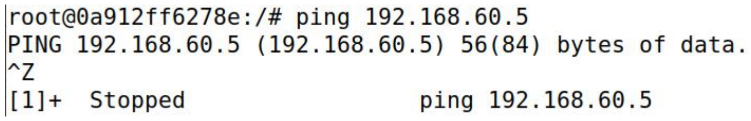
Task2.B：Protecting the Internal Network

在路由上配置 iptable



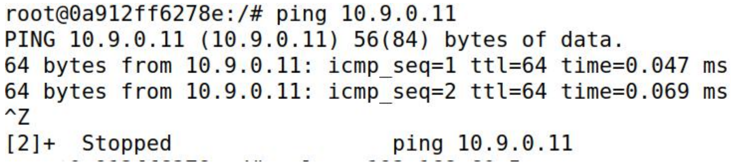
```
root@f71f0b983d48:/# iptables -A FORWARD -o eth1 -p icmp --icmp-type echo-reques
t -j DROP
root@f71f0b983d48:/# iptables -A FORWARD -i eth1 -p icmp --icmp-type echo-reply
-j DROP
root@f71f0b983d48:/# iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-reques
t -j ACCEPT
root@f71f0b983d48:/# iptables -A FORWARD -o eth0 -p icmp --icmp-type echo-reply
-j ACCEPT
root@f71f0b983d48:/# iptables -A FORWARD -o eth1 -p icmp --icmp-type echo-reply
-j ACCEPT
root@f71f0b983d48:/# iptables -A FORWARD -i eth1 -p icmp --icmp-type echo-reques
t -j ACCEPT
root@f71f0b983d48:/# iptables -P FORWARD DROP
```

外部主机无法 ping 内部主机



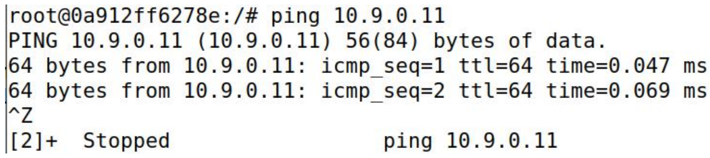
```
root@0a912ff6278e:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^Z
[1]+  Stopped                 ping 192.168.60.5
```

外部主机可以 ping 路由



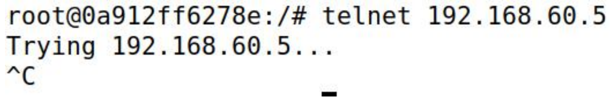
```
root@0a912ff6278e:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.047 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.069 ms
^Z
[2]+  Stopped                 ping 10.9.0.11
```

内部主机可以 ping 外部主机

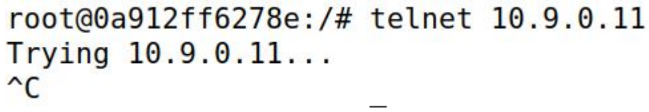


```
root@0a912ff6278e:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.047 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.069 ms
^Z
[2]+  Stopped                 ping 10.9.0.11
```

其他数据包内外不通（内外无法互相 telnet）



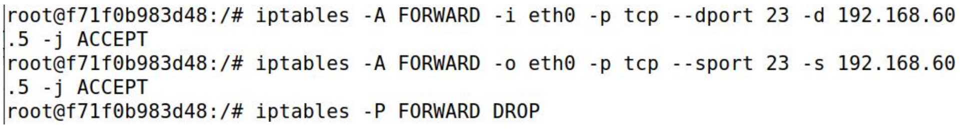
```
root@0a912ff6278e:/# telnet 192.168.60.5
Trying 192.168.60.5...
^C
```



```
root@0a912ff6278e:/# telnet 10.9.0.11
Trying 10.9.0.11...
^C
```

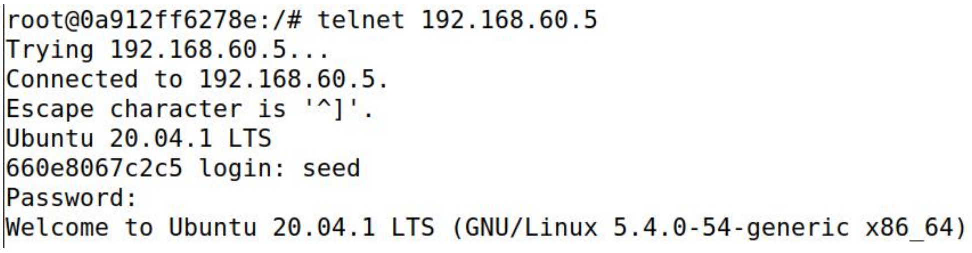
Task 2.C: Protecting Internal Servers

在路由上配置 iptable



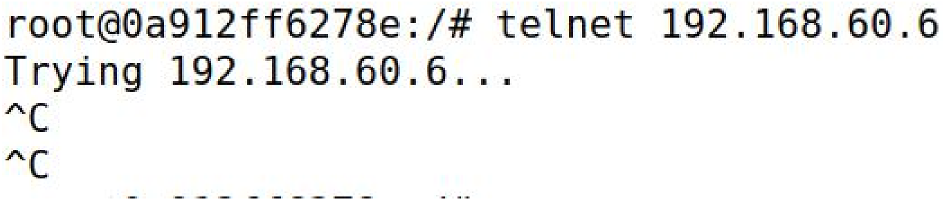
```
root@f71f0b983d48:/# iptables -A FORWARD -i eth0 -p tcp --dport 23 -d 192.168.60
.5 -j ACCEPT
root@f71f0b983d48:/# iptables -A FORWARD -o eth0 -p tcp --sport 23 -s 192.168.60
.5 -j ACCEPT
root@f71f0b983d48:/# iptables -P FORWARD DROP
```

外部主机可以 telnet 到 192.168.60.5



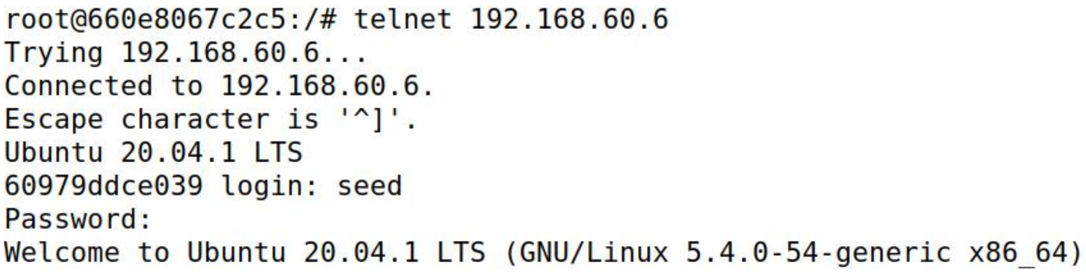
```
root@0a912ff6278e:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
660e8067c2c5 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

外部主机无法 telnet 到其他内部主机



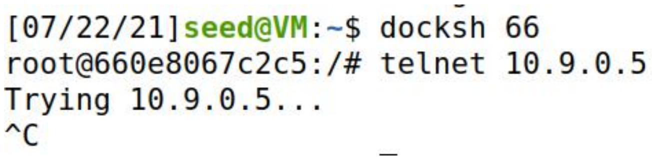
```
root@0a912ff6278e:/# telnet 192.168.60.6
Trying 192.168.60.6...
^C
^C
```

内部主机可以 telnet 到其他内部主机



```
root@660e8067c2c5:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
60979ddce039 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

内部主机无法 telnet 到外部主机

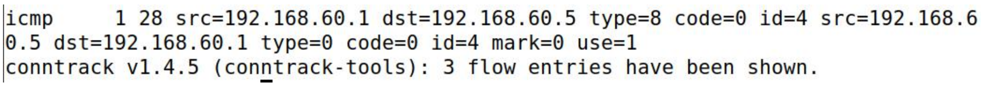


```
[07/22/21]seed@VM:~$ docksh 66
root@660e8067c2c5:/# telnet 10.9.0.5
Trying 10.9.0.5...
^C
```

Task3: Connection Tracking and Stateful Firewall

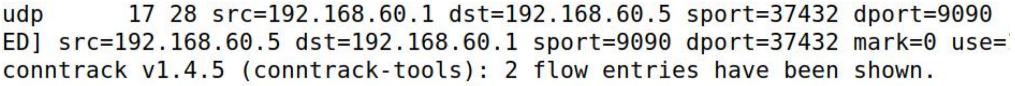
Task 3.A: Experiment with the Connection Tracking

ICMP 连接大概持续 30s



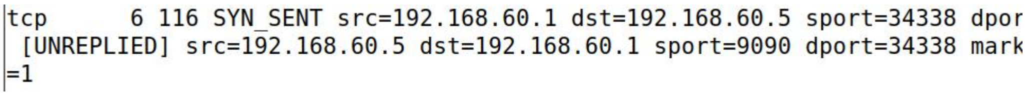
```
icmp     1 28 src=192.168.60.1 dst=192.168.60.5 type=8 code=0 id=4 src=192.168.6
0.5 dst=192.168.60.1 type=0 code=0 id=4 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 3 flow entries have been shown.
```

UDP 连接也大概持续 30s



```
udp      17 28 src=192.168.60.1 dst=192.168.60.5 sport=37432 dport=9090
ED] src=192.168.60.5 dst=192.168.60.1 sport=9090 dport=37432 mark=0 use=
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
```

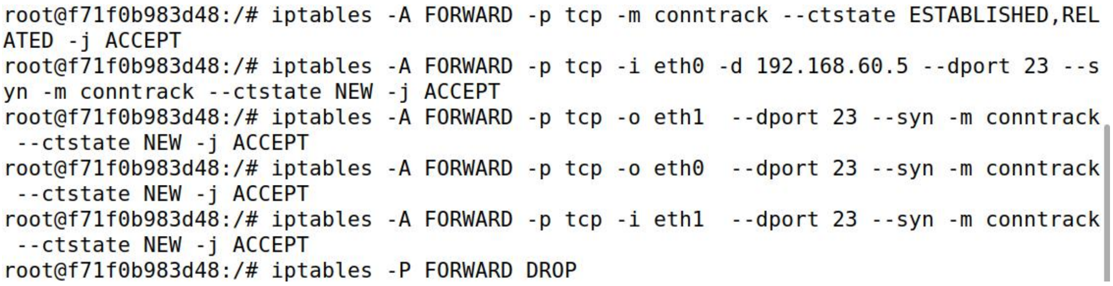
TCP 连接大概持续 120s



```
tcp      6 116 SYN_SENT src=192.168.60.1 dst=192.168.60.5 sport=34338 dpor
 [UNREPLIED] src=192.168.60.5 dst=192.168.60.1 sport=9090 dport=34338 mark
=1
```

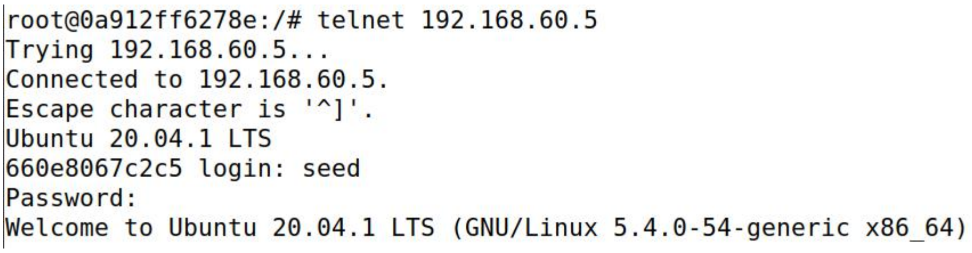
Task 3.B: Setting Up a Stateful Firewall

在路由上配置 iptable



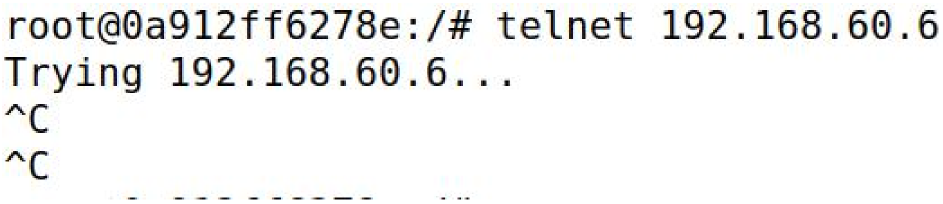
```
root@f71f0b983d48:/# iptables -A FORWARD -p tcp -m conntrack --ctstate ESTABLISHED,REL
ATED -j ACCEPT
root@f71f0b983d48:/# iptables -A FORWARD -p tcp -i eth0 -d 192.168.60.5 --dport 23 --s
yn -m conntrack --ctstate NEW -j ACCEPT
root@f71f0b983d48:/# iptables -A FORWARD -p tcp -o eth1  --dport 23 --syn -m conntrack
 --ctstate NEW -j ACCEPT
root@f71f0b983d48:/# iptables -A FORWARD -p tcp -o eth0  --dport 23 --syn -m conntrack
 --ctstate NEW -j ACCEPT
root@f71f0b983d48:/# iptables -A FORWARD -p tcp -i eth1  --dport 23 --syn -m conntrack
 --ctstate NEW -j ACCEPT
root@f71f0b983d48:/# iptables -P FORWARD DROP
```

外部主机可以 telnet 到 192.168.60.5



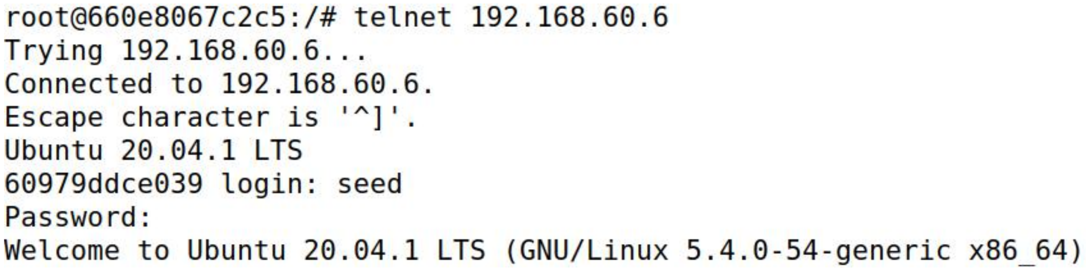
```
root@0a912ff6278e:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
660e8067c2c5 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

外部主机无法 telnet 到其他内部主机



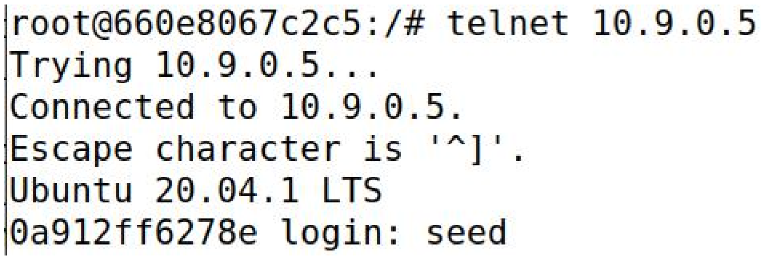
```
root@0a912ff6278e:/# telnet 192.168.60.6
Trying 192.168.60.6...
^C
^C
```

内部主机可以 telnet 到其他内部主机



```
root@660e8067c2c5:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
60979ddce039 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

内部主机可以 telnet 到外部主机

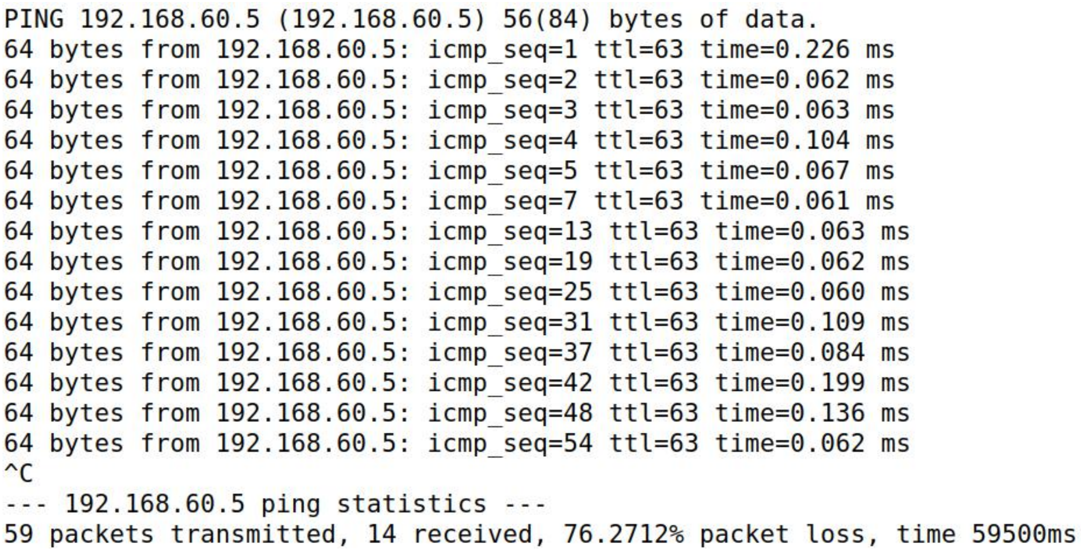


```
root@660e8067c2c5:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
0a912ff6278e login: seed
```

二者优缺点：基于连接的防火墙只需要在建立连接的时候判定是否合法，之后的报文只需要判定是否建立连接即可，但它需要调用 conntrack；而没有基于连接的防火墙则需要对所有报文进行判定是否合法，但不需要借助 conntrack。

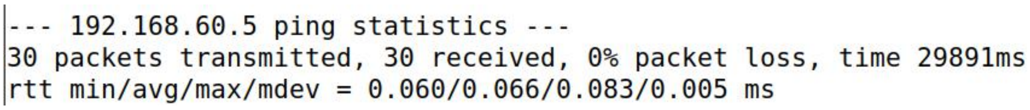
Task4: Limiting Network Traffific

两条都在的情况下，一开始会有 5 个比较快的 ping，之后平均每分钟 ping10 个报文。



```
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=0.226 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.062 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.063 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.104 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.067 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.061 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.063 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.062 ms
64 bytes from 192.168.60.5: icmp_seq=25 ttl=63 time=0.060 ms
64 bytes from 192.168.60.5: icmp_seq=31 ttl=63 time=0.109 ms
64 bytes from 192.168.60.5: icmp_seq=37 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=42 ttl=63 time=0.199 ms
64 bytes from 192.168.60.5: icmp_seq=48 ttl=63 time=0.136 ms
64 bytes from 192.168.60.5: icmp_seq=54 ttl=63 time=0.062 ms
^C
--- 192.168.60.5 ping statistics ---
59 packets transmitted, 14 received, 76.2712% packet loss, time 59500ms
```

去掉第二条规则后则报文数量和原来一样，平均每秒一个，没有受限。因此第二条规则是不可缺的，因为第二条规则决定了如何处理超出第一条限制的报文，没有第二条规则那么路由在处理超出限制的报文时采用接收，从而达不到限制报文数量的作用。

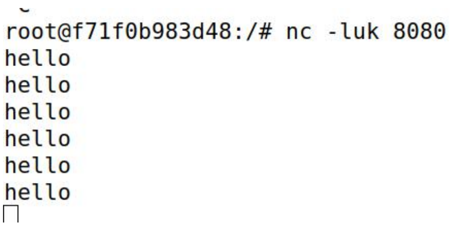


```
--- 192.168.60.5 ping statistics ---
30 packets transmitted, 30 received, 0% packet loss, time 29891ms
rtt min/avg/max/mdev = 0.060/0.066/0.083/0.005 ms
```

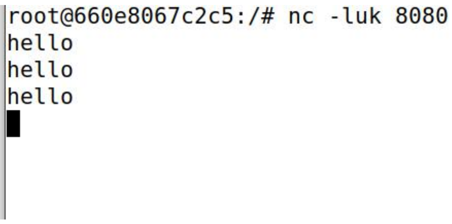
Task5:Load Balancing

Using the nth mode (round-robin)

在 router 上配置 eth 规则后，在 router 和 192.168.60.5 上都开启 nc -luk 8080，不断从 10.9.0.5 向 10.9.0.11 发送 hello，可以看到每三个里有一个发给了 192.168.60.5。

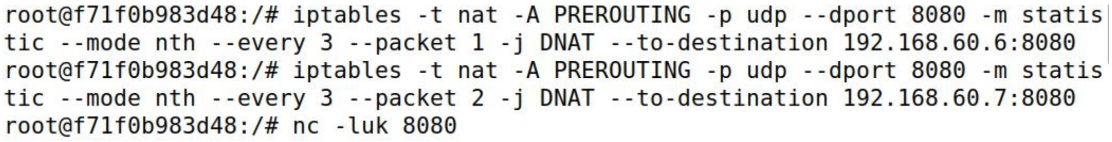


```
root@f71f0b983d48:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
```



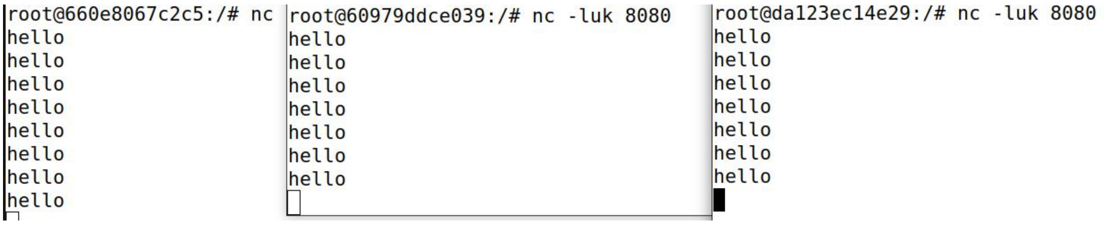
```
root@660e8067c2c5:/# nc -luk 8080
hello
hello
hello
```

添加新规则



```
root@f71f0b983d48:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode nth --every 3 --packet 1 -j DNAT --to-destination 192.168.60.6:8080
root@f71f0b983d48:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode nth --every 3 --packet 2 -j DNAT --to-destination 192.168.60.7:8080
root@f71f0b983d48:/# nc -luk 8080
```

在这种规则下，数据包按照顺序一人一个 hello



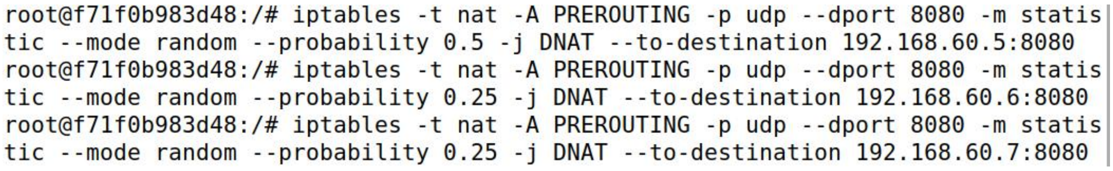
```
root@660e8067c2c5:/# nc
hello
hello
hello
hello
hello
hello
hello
hello
```

```
root@60979ddce039:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
hello
```

```
root@da123ec14e29:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
hello
```

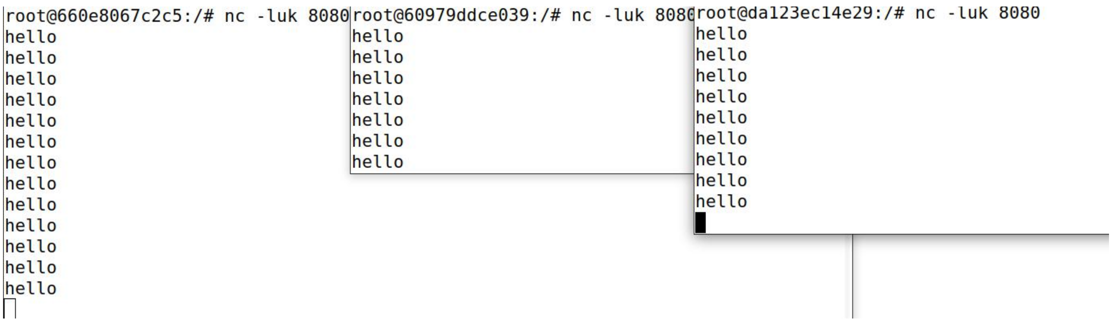
Using the random mode.

使用 random 规则，其中向 192.168.60.5 发送报文的概率为 0.5，向 192.168.60.6 发送报文的概率为 0.25，向 192.168.60.7 发送报文的概率为 0.25



```
root@f71f0b983d48:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode random --probability 0.5 -j DNAT --to-destination 192.168.60.5:8080
root@f71f0b983d48:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode random --probability 0.25 -j DNAT --to-destination 192.168.60.6:8080
root@f71f0b983d48:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode random --probability 0.25 -j DNAT --to-destination 192.168.60.7:8080
```

在这种规则下，192.168.60.5、192.168.60.6、192.168.60.7 收到的 hello 数量差不多为 2:1:1。



```
root@660e8067c2c5:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

```
root@60979ddce039:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
hello
```

```
root@da123ec14e29:/# nc -luk 8080
hello
hello
hello
hello
hello
hello
hello
hello
hello
```