Firewall Exploration Lab

57118211 谢瑞

Task 1: Implementing a Simple Firewall

Task 1.A: Implementing a Simple Kernel Module

把 kernel_module 拷贝到 /home/seed/ 目录下进行编译：

```
[07/26/21]seed@VM:~/kernel_module$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/kernel_modu
le modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic
'
  CC [M]  /home/seed/kernel_module/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/seed/kernel_modu
le/hello.o
see include/linux/module.h for more information
  CC [M]  /home/seed/kernel_module/hello.mod.o
  LD [M]  /home/seed/kernel_module/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
```

编译成功后测试以下命令：

```
[07/26/21]seed@VM:~/kernel_module$ sudo insmod hello.ko
[07/26/21]seed@VM:~/kernel_module$ lsmod | grep hello
hello                  16384  0
[07/26/21]seed@VM:~/kernel_module$ sudo rmmod hello
[07/26/21]seed@VM:~/kernel_module$ dmesg
[ 3514.959736] hello: module license 'unspecified' taints kernel.
[ 3514.959738] Disabling lock debugging due to kernel taint
[ 3514.959821] hello: module verification failed: signature and/or r
equired key missing - tainting kernel
[ 3514.961319] Hello World!
[ 3543.836630] Bye-bye World!.
```

Task 1. B: Implementing a Simple Firewall Using Netfilter

将文件拷贝到 /home/seed/ 下面进行编译：

```
[07/26/21]seed@VM:~/packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/packet_filt
er modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic
'
  CC [M]  /home/seed/packet_filter/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/packet_filter/seedFilter.mod.o
  LD [M]  /home/seed/packet_filter/seedFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
```

加载内核前，可以看到 dig @8.8.8.8 www.example 命令可以得到响应：

```
[07/26/21]seed@VM:~/packet_filter$ dig @8.8.8.8 www.example

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 www.example
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 30089
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL
: 1

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

;; AUTHORITY SECTION:
.                       86390   IN      SOA     a.root-servers.net.
nstld.verisign-grs.com. 2021072600 1800 900 604800 86400

;; Query time: 204 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Jul 26 07:52:32 EDT 2021
;; MSG SIZE  rcvd: 115
```

加载到内核后，可以看到防火墙生效：

```
[07/26/21]seed@VM:~/packet_filter$ sudo insmod seedFilter.ko
[07/26/21]seed@VM:~/packet_filter$ lsmod |grep seedFilter
seedFilter             16384  0
[07/26/21]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
```

最后从内核中移除：

```
[07/26/21]seed@VM:~/packet_filter$ sudo rmmod seedFilter
[07/26/21]seed@VM:~/packet_filter$ lsmod |grep seedFilter
```

把 printInfo 函数挂到 nerfilter 的 hook 上：

```
1. int registerFilter(void) {
2.  printk(KERN_INFO "Registering filters.\n";
3.
4.  hook1.hook = printInfo;
5.  hook1.hooknum = NF_INET_LOCAL_OUT;
6.  hook1.pf = PF_INET;
7.  hook1.priority = NF_IP_PRI_FIRST;
8.  nf_register_net_hook(&init_net, &hook1);
9.
10. hook2.hook = blockUDP;
11. hook2.hooknum = NF_INET_POST_ROUTING;
12. hook2.pf = PF_INET;
13. hook2.priority = NF_IP_PRI_FIRST;
14. nf_register_net_hook(&init_net, &hook2);
15.
16. hook3.hook = printInfo;
17. hook3.hooknum = NF_INET_LOCAL_IN;
18. hook3.pf = PF_INET;
19. hook3.priority = NF_IP_PRI_FIRST;
20. nf_register_net_hook(&init_net, &hook3);
21.
22. hook4.hook = printInfo;
23. hook4.hooknum = NF_INET_FORWARD;
24. hook4.pf = PF_INET;
25. hook4.priority = NF_IP_PRI_FIRST;
26. nf_register_net_hook(&init_net, &hook4);
27.
28. hook5.hook = printInfo;
29. hook5.hooknum = NF_INET_PRE_ROUTING;
30. hook5.pf = PF_INET;
31. hook5.priority = NF_IP_PRI_FIRST;
32. nf_register_net_hook(&init_net, &hook5);
33.
34. hook6.hook = printInfo;
35. hook6.hooknum = NF_INET_POST_ROUTING;
36. hook6.pf = PF_INET;
37. hook6.priority = NF_IP_PRI_FIRST;
38. nf_register_net_hook(&init_net, &hook6);
39.
40. return 0;
41.}
```

对 hook 函数进行分析：
PRE_ROUTING ： 当二层收包结束后，会根据注册的协议和回调函数分发数据包，其中 ipv4 的数据包会分发到 ip_rcv 函数进行三层协议栈处理，该函数对数据包的合法性进行检查，并且设置一些必要字段之后会调用。
LOCAL_IN ： ip_rcv 函数在经过了 PRE_ROUTING 钩子点之后，会调用 ip_rcv_finish 函数，该函数的主要功能是查路由确定数据包是输入到本地还是转发，并调用 dst_input 函数。当数据包输入本地时， dst_input 函数实际调用了 ip_local_deliver 函数，函数首先对分片进行检查，如果是分片则需要进行重组，然后调用该 hook。
FORWARD ： 主机作报文转发时会调用(本实验用 10.9.0.5 ping 192.168.60.7 观察)。
LOCAL_OUT ： 从本机发出的数据包在查询路由成功之后调用。
POST_ROUTING ： 转发的数据包或者是本地输出的数据包都会在函数设置设备和协议之后调用。

测试结果中出现所有 hook：

```
[ 7460.600524] *** LOCAL_IN
[ 7460.600525]     10.9.0.5   --> 10.9.0.1 (ICMP)
[ 7460.600538] *** LOCAL_OUT
[ 7460.600539]     10.9.0.1   --> 10.9.0.5 (ICMP)
[ 6752.617954] *** PRE_ROUTING
[ 6752.617954]     192.168.60.7   --> 10.9.0.5 (ICMP)
[ 6752.617955] *** FORWARD
[ 6752.617955]     192.168.60.7   --> 10.9.0.5 (ICMP)
[ 6752.617956] *** POST_ROUTING
[ 6752.617956]     192.168.60.7   --> 10.9.0.5 (ICMP)
```

设置两个 hook 函数 prevent_ping 和 prevent_telnet ：

```
1. unsigned int prevent_ping(void *priv, struct sk_buff *skb, const
   struct
2. nf_hook_state *state)
3. {
4.  struct iphdr *iph;
5.  char ip[16] = "10.9.0.1";
6.  u32  ip_addr;
7.  if (!skb) return NF_ACCEPT;
8.  iph = ip_hdr(skb);
9.  // Convert the IPv4 address from dotted decimal to 32-bit binary
10. in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);
11. if (iph->protocol == IPPROTO_ICMP) {
12.   if (iph->daddr == ip_addr){
13.      printk(KERN_WARNING "*** Dropping %pI4 (ICMP)\n", &(iph-
14.>daddr));
15.      return NF_DROP;
```

```
16.    }
17.  }
18.  return NF_ACCEPT;
19.}
20.
21.unsigned int prevent_telnet(void *priv, struct sk_buff *skb, cons
   t
22.struct nf_hook_state *state)
23.{
24.  struct iphdr *iph;
25.  struct tcphdr *tcph;
26.  u16  port  = 23;
27.  char ip[16] = "10.9.0.1";
28.  u32  ip_addr;
29.  if (!skb) return NF_ACCEPT;
30.  iph = ip_hdr(skb);
31.  // Convert the IPv4 address from dotted decimal to 32-bit binary
32.  in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);
33.  if (iph->protocol == IPPROTO_TCP)
34.  {
35.    tcph = tcp_hdr(skb);
36.    if (iph->daddr == ip_addr && ntohs(tcph->dest) == port)
37.   {
38.     printk(KERN_WARNING "*** Dropping %pI4 (TCP), port %d\n", &
39.(iph->daddr), port);
40.     return NF_DROP;
41.   }
42.  }
43.  return NF_ACCEPT;
44.}
```

把两个函数注册到同一个 hook LOCAL_IN 中：

```
1.  hook2.hook = prevent_ping;
2.  hook2.hooknum = NF_INET_LOCAL_IN;
3.  hook2.pf = PF_INET;
4.  hook2.priority = NF_IP_PRI_FIRST;
5.  nf_register_net_hook(&init_net, &hook2);
6.
7.  hook3.hook = prevent_telnet;
8.  hook3.hooknum = NF_INET_LOCAL_IN;
9.  hook3.pf = PF_INET;
10. hook3.priority = NF_IP_PRI_FIRST;
11. nf_register_net_hook(&init_net, &hook3);
```

开启容器，登录到 docker1(10.9.0.5) ，在容器上分别进行 ping 10.9.0.1 和 telnet 10.9.0.1 ：

```
root@76d628dfc61f:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
```

```
root@76d628dfc61f:/# telnet 10.9.0.1
Trying 10.9.0.1...
```

通过内核缓存可以看见：

```
[  158.321593] *** Dropping 10.9.0.1 (ICMP)
[  159.344744] *** Dropping 10.9.0.1 (ICMP)
[  160.368194] *** Dropping 10.9.0.1 (ICMP)
[  161.391290] *** Dropping 10.9.0.1 (ICMP)
[  212.239004] *** Dropping 10.9.0.1 (TCP), port 23
[  214.254512] *** Dropping 10.9.0.1 (TCP), port 23
[  218.382655] *** Dropping 10.9.0.1 (TCP), port 23
```

Task 2: Experimenting with Stateless Firewall Rules

Task 2.A: Protecting the Router

在路由器上输入以下命令：

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

在 10.5.0.1 上 ping 通路由器：

```
root@76d628dfc61f:/# 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=0.129 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.148 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.245 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.175 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.327 ms
64 bytes from 10.9.0.11: icmp_seq=6 ttl=64 time=0.146 ms
```

telnet 不到路由器：

```
root@76d628dfc61f:/# telnet 10.9.0.11
Trying 10.9.0.11...
telnet: Unable to connect to remote host: Connection timed out
```

Task 2. B: Protecting the Internal Network

在路由器上输入以下命令：

```
iptables -A INPUT -p icmp -j ACCEPT
iptables -A FORWARD -p icmp -i eth1 -o eth0 -j ACCEPT
iptables -A FORWARD -p icmp -i eth0 -o eth1 --icmp-type echo-reply -j
ACCEPT
iptables -A FORWARD -p icmp -i eth0 -o eth1 -j DROP
iptables -A FORWARD -j DROP
```

此时，外部主机无法 ping 通内部主机：

```
root@76d628dfc61f:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
```

外部主机可以 telnet 到内部主机

```
root@76d628dfc61f:/# 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=0.119 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.137 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.152 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.143 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.174 ms
64 bytes from 10.9.0.11: icmp_seq=6 ttl=64 time=0.152 ms
64 bytes from 10.9.0.11: icmp_seq=7 ttl=64 time=0.138 ms
```

内部主机可以 ping 通外部主机：

```
root@1154b40ae293:/# 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=0.233 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.266 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.179 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.187 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.160 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=63 time=0.371 ms
```

内网和外网之间的其他所有数据包被阻止：

```
root@76d628dfc61f:/# telnet 192.168.60.5
Trying 192.168.60.5...

root@1154b40ae293:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

Task 2. C: Protecting Internal Servers

在路由器上输入以下命令：

```
iptables -A INPUT -p tcp -j ACCEPT
iptables -A FORWARD -p tcp -i eth0 -o eth1 --dport 23 -d 192.168.60.5
-j ACCEPT
iptables -A FORWARD -p tcp -i eth1 -o eth0 --sport 23 -s 192.168.60.5
-j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -j DROP
iptables -A FORWARD -i eth1 -o eth0 -j DROP
```

外部主机能 telnet 到 192.168.60.5 上的服务器：

```
root@826d76c9fa49:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
d7a1ad3e2e0d login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

不能访问其他内网服务器：

```
root@826d76c9fa49:/# telnet 192.168.60.6
Trying 192.168.60.6...
```

```
root@826d76c9fa49:/# telnet 192.168.60.7
Trying 192.168.60.7...
```

内部主机可以访问所有内部服务器(以 192.168.60.5 为例)：

```
root@d7a1ad3e2e0d:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
e748c89a0396 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

```
root@d7a1ad3e2e0d:/# telnet 192.168.60.7
Trying 192.168.60.7...
Connected to 192.168.60.7.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
4986876faade login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

内部主机无法访问外部服务器：

```
root@d7a1ad3e2e0d:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

Task 3: Connection Tracking and Stateful Firewall

Task 3.A: Experiment with the Connection Tracking

ICMP 的连接状态保持时间只有 30 秒左右：

```
root@cf06900b167d:/# 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@cf06900b167d:/# conntrack -L
icmp     1 23 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@cf06900b167d:/# conntrack -L
icmp     1 14 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@cf06900b167d:/# conntrack -L
icmp     1 9 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@cf06900b167d:/# conntrack -L
icmp     1 2 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@cf06900b167d:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
```

UDP 的连接状态保持时间和也约 30 秒：

```
root@cf06900b167d:/# conntrack -L
udp      17 27 src=10.9.0.5 dst=192.168.60.5 sport=52159 dport=9090 [UNREPLIED] src=
192.168.60.5 dst=10.9.0.5 sport=9090 dport=52159 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@cf06900b167d:/# conntrack -L
udp      17 22 src=10.9.0.5 dst=192.168.60.5 sport=52159 dport=9090 [UNREPLIED] src=
192.168.60.5 dst=10.9.0.5 sport=9090 dport=52159 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@cf06900b167d:/# conntrack -L
udp      17 13 src=10.9.0.5 dst=192.168.60.5 sport=52159 dport=9090 [UNREPLIED] src=
192.168.60.5 dst=10.9.0.5 sport=9090 dport=52159 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@cf06900b167d:/# conntrack -L
udp      17 0 src=10.9.0.5 dst=192.168.60.5 sport=52159 dport=9090 [UNREPLIED] src=1
92.168.60.5 dst=10.9.0.5 sport=9090 dport=52159 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@cf06900b167d:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
```

TCP 的连接状态保持时间非常长，大约 432000 秒：

```
root@cf06900b167d:/# conntrack -L
tcp      6 431997 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=56530 dport=9090 s
rc=192.168.60.5 dst=10.9.0.5 sport=9090 dport=56530 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@cf06900b167d:/# conntrack -L
tcp      6 431990 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=56530 dport=9090 s
rc=192.168.60.5 dst=10.9.0.5 sport=9090 dport=56530 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

Task 3. B: Setting Up a Stateful Firewall

在路由器上输入以下命令：

```
iptables -F
iptables -A FORWARD -p tcp -m conntrack  -ctstate ESTABLISHED,RELATED
-j ACCEPT
iptables  -A  FORWARD  -p  tcp  --dport  23  -d  192.168.60.5  -syn  -m
conntrack --ctstate NEW -j ACCEPT
iptables -A FORWARD -p tcp --dport 23 -d 10.9.0.0/24  -syn -m conntrack
--ctstate NEW -j ACCEPT
iptables -P FORWARD DROP
```

其他情况都和 2.C 一样，但内网能 telnet 上外网了：

```
root@d7a1ad3e2e0d:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
826d76c9fa49 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x8
6_64)
```

Task 4: Limiting Network Traffic

在路由器上输入以下命令：

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

从 10.9.0.5 ping 192.168.60.5 ，可以观察到开始五个包发的很快，之后每 6 秒发一个包：

```
root@826d76c9fa49:/# 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=0.221 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.101 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.358 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.138 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.114 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.101 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.152 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.098 ms
64 bytes from 192.168.60.5: icmp_seq=25 ttl=63 time=0.124 ms
64 bytes from 192.168.60.5: icmp_seq=31 ttl=63 time=0.194 ms
```

如果只执行第一条命令，从外部 (10.9.0.5)ping 192.168.60.5 ，可以观察到和平时的发包速度一样，因为 iptables 默认的 FORWARD 表是接受所有包，所以如果不写第二条命令，发包会正常进行。

Task 5：Load Balancing

nth 模式：
在路由器中执行以下命令：

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode
nth --every 3 --packet 0 -j DNAT --to-destination 192.168.60.5:8080
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode
nth --every 3 --packet 1 -j DNAT --to-destination 192.168.60.6:8080
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode
nth --every 3 --packet 2 -j DNAT --to-destination 192.168.60.7:8080
```

在 10.9.0.5 上给路由器发报文，可以发现由于负载均衡，各个主机监听到的报文数量平均。

```
root@d7a1ad3e2e0d:/# nc -luk 8080
hello_1
```

```
root@e748c89a0396:/# nc -luk 8080
hello 2
```

```
root@4986876faade:/# nc -luk 8080
hello 3
```

random 模式：
在路由器中执行以下命令

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode
random --probability 0.33 -j DNAT --to-destination 192.168.60.5:8080
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode
random --probability 0.33 -j DNAT --to-destination 192.168.60.6:8080
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode
random --probability 0.34 -j DNAT --to-destination 192.168.60.7:8080
```

```
root@d7a1ad3e2e0d:/# nc -luk 8080
hello_1
hello_2
hello_4
hello_7
hello_13
hello_14
root@e748c89a0396:/# nc -luk 8080
hello_5
hello_6
hello_10
hello_11
hello_12
root@4986876faade:/# nc -luk 8080
hello_3
hello_8
hello_9
hello_15
```

在 10.9.0.5 上给路由器发报文，虽然是等概率发送数据，但可能由于样本数量较少，每个主机收到的数量各不相同，当样本数量足够多时，应该是趋于平均的。