Lab6

57118232 谢隆文

Task 1：Implementing a Simple Firewall

Task1.A Implement a Simple Kernel Module

原始目录存在空格，目录的空格被 make 识别为编译的 target ，所以我们需要把 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_module modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/seed/kernel_module/hello.o
see include/linux/module.h for more information
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[07/26/21]seed@VM:~/kernel_module$ sudo insmod hello.ko
[07/26/21]seed@VM:~/kernel_module$ lsmod | grep hello
hello                  16384  0
[  402.783661] Hello World!
[  568.469739] Bye-bye World!.
```

Task 1.B：Implement a Simple Firewall Using Netfilter

1、和前面一样，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_filter 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'
[07/26/21]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: 46429
;; 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.        20930   IN      A       93.184.216.34

;; Query time: 259 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Jul 26 02:58:02 EDT 2021
;; MSG SIZE  rcvd: 60
```

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$ 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
```

2. 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

```
[07/26/21]seed@VM:~/packet_filter$ sudo dmesg -c
[13751.417489] *** PRE_ROUTING
[13751.417491]     10.80.128.28  --> 10.0.2.15 (UDP)
[13751.418269] *** PRE_ROUTING
[13751.418271]     127.0.0.1  --> 127.0.0.53 (UDP)
[13751.421439] *** PRE_ROUTING
[13751.421441]     10.80.128.28  --> 10.0.2.15 (UDP)
[13751.421536] *** PRE_ROUTING
[13751.421537]     127.0.0.53  --> 127.0.0.1 (UDP)
[13784.320530] *** PRE_ROUTING
[13784.320531]     127.0.0.1  --> 127.0.0.1 (UDP)
[13784.320786] *** Dropping 8.8.8.8 (UDP), port 53
[13789.320678] *** Dropping 8.8.8.8 (UDP), port 53
[13794.321599] *** Dropping 8.8.8.8 (UDP), port 53
```

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。

```
[14207.723137] The filters are being removed.
[14232.637124] Registering filters.
[14235.174264] *** Dropping 8.8.8.8 (UDP), port 53
[14240.175847] *** Dropping 8.8.8.8 (UDP), port 53
[14245.179881] *** Dropping 8.8.8.8 (UDP), port 53
```

若该数据报应该被转发则它被 NF_INET_FORWARD 处理

```
[11011.229107] Registering filters.
[11038.881281] *** LOCAL_OUT
[11038.881282]     127.0.0.1  --> 127.0.0.1 (UDP)
[11038.881702] *** LOCAL_OUT
[11038.881703]     10.0.2.15  --> 8.8.8.8 (UDP)
[11038.881708] *** Dropping 8.8.8.8 (UDP), port 53
[11043.879873] *** LOCAL_OUT
[11043.879875]     10.0.2.15  --> 8.8.8.8 (UDP)
[11043.879884] *** Dropping 8.8.8.8 (UDP), port 53
[11048.879945] *** LOCAL_OUT
[11048.879947]     10.0.2.15  --> 8.8.8.8 (UDP)
[11048.879956] *** Dropping 8.8.8.8 (UDP), port 53
[11051.520558] *** LOCAL_OUT
[11051.520560]     10.0.2.15  --> 10.80.128.28 (UDP)
[11051.525987] *** LOCAL_OUT
[11051.525988]     127.0.0.1  --> 127.0.0.53 (UDP)
[11051.526083] *** LOCAL_OUT
[11051.526083]     10.0.2.15  --> 10.80.128.28 (UDP)
[11051.529714] *** LOCAL_OUT
[11051.529715]     127.0.0.53  --> 127.0.0.1 (UDP)
```

挂载 NF_INET_LOCAL_OUT 时，本机产生的数据包将会第一个到达此 HOOK ，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。

```
[07/26/21]seed@VM:~/packet_filter$ sudo dmesg -c
[14494.411309] The filters are being removed.
[14725.712695] Registering filters.
[14728.213308] *** POST_ROUTING
[14728.213310]     127.0.0.1  --> 127.0.0.1 (UDP)
[14728.214146] *** Dropping 8.8.8.8 (UDP), port 53
[14733.216328] *** Dropping 8.8.8.8 (UDP), port 53
[14738.229683] *** Dropping 8.8.8.8 (UDP), port 53
[14788.992506] *** POST_ROUTING
[14788.992508]     10.0.2.15  --> 35.224.170.84 (TCP)
[14789.995610] *** POST_ROUTING
[14789.995623]     10.0.2.15  --> 35.224.170.84 (TCP)
[14792.011197] *** POST_ROUTING
[14792.011199]     10.0.2.15  --> 35.224.170.84 (TCP)
[14796.139081] *** POST_ROUTING
[14796.139101]     10.0.2.15  --> 35.224.170.84 (TCP)
[14804.331815] *** POST_ROUTING
[14804.331834]     10.0.2.15  --> 35.224.170.84 (TCP)
```

3、修改后的代码如下：

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>
#include <linux/udp.h>
#include <linux/icmp.h>
#include <linux/if_ether.h>
#include <linux/inet.h>
static struct nf_hook_ops hook1, hook2, hook3, hook4;
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] = "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;
}
}
return NF_ACCEPT;
}
unsigned int blockTCP(void *priv, struct sk_buff *skb,
const struct nf_hook_state *state)
{
struct iphdr *iph;
struct tcphdr *tcph;
u16 port = 23;
char ip[16] = "10.9.0.1";
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_TCP) {
tcph = tcp_hdr(skb);
if (iph->daddr == ip_addr && ntohs(tcph->dest) == port){
printk(KERN_WARNING "*** Dropping %pI4 (TCP), port %d\n", &(iph-
>daddr), port);
return NF_DROP;
}
}
return NF_ACCEPT;
}
unsigned int blockICMP(void *priv, struct sk_buff *skb,
const struct nf_hook_state *state)
{
struct iphdr *iph;
struct icmphdr *icmph;
char ip[16] = "10.9.0.1";
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_ICMP) {
icmph = icmp_hdr(skb);
if (iph->daddr == ip_addr){
printk(KERN_WARNING "*** Dropping %pI4 (ICMP)\n", &(iph->daddr));
return NF_DROP;
}
}
```

```
return NF_ACCEPT;
}
unsigned int printInfo(void *priv, struct sk_buff *skb,
const struct nf_hook_state *state)
{
struct iphdr *iph;
char *hook;
char *protocol;
switch (state->hook){
case NF_INET_LOCAL_IN: hook = "LOCAL_IN"; break;
case NF_INET_LOCAL_OUT: hook = "LOCAL_OUT"; break;
case NF_INET_PRE_ROUTING: hook = "PRE_ROUTING"; break;
case NF_INET_POST_ROUTING: hook = "POST_ROUTING"; break;
case NF_INET_FORWARD: hook = "FORWARD"; break;
default: hook = "IMPOSSIBLE"; break;
}
printk(KERN_INFO "*** %s\n", hook); // Print out the hook info
iph = ip_hdr(skb);
switch (iph->protocol){
case IPPROTO_UDP: protocol = "UDP"; break;
case IPPROTO_TCP: protocol = "TCP"; break;
case IPPROTO_ICMP: protocol = "ICMP"; break;
default: protocol = "OTHER"; break;
}
// Print out the IP addresses and protocol
printk(KERN_INFO " %pI4 --> %pI4 (%s)\n",
&(iph->saddr), &(iph->daddr), protocol);
return NF_ACCEPT;
}
int registerFilter(void) {
printk(KERN_INFO "Registering filters.\n");
hook1.hook = printInfo;
hook1.hooknum = NF_INET_LOCAL_OUT;
hook1.pf = PF_INET;
hook1.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook1);
hook2.hook = blockUDP;
hook2.hooknum = NF_INET_POST_ROUTING;
hook2.pf = PF_INET;
hook2.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook2);
hook3.hook = blockICMP;
hook3.hooknum = NF_INET_PRE_ROUTING;
hook3.pf = PF_INET;
hook3.priority = NF_IP_PRI_FIRST;
```

```
nf_register_net_hook(&init_net, &hook3);
hook4.hook = blockTCP;
hook4.hooknum = NF_INET_PRE_ROUTING;
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);
}
module_init(registerFilter);
module_exit(removeFilter);
MODULE_LICENSE("GPL");
```

加载内核

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

开启容器，在 10.9.0.5 容器上分别进行 ping 10.9.0.1 和 telnet 10.9.0.1 发现都不通过

dmesg 查看：

```
root@89c2a49a4fe2:/# 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 ---
6 packets transmitted, 0 received, 100% packet loss, time 5101ms

root@89c2a49a4fe2:/# telnet 10.9.0.1
Trying 10.9.0.1...
^C
[ 2094.390684] *** Dropping 10.9.0.1 (ICMP), port
[ 2095.396858] *** Dropping 10.9.0.1 (ICMP), port
[ 2096.419797] *** Dropping 10.9.0.1 (ICMP), port
[ 2097.443347] *** Dropping 10.9.0.1 (ICMP), port
[ 2098.466919] *** Dropping 10.9.0.1 (ICMP), port
[ 2099.491227] *** Dropping 10.9.0.1 (ICMP), port
[ 2102.334535] *** Dropping 10.9.0.1 (TCP), port 23
[ 2103.362271] *** Dropping 10.9.0.1 (TCP), port 23
[ 2105.379589] *** Dropping 10.9.0.1 (TCP), port 23
```

Task 2：Experimenting with Stateless Firewall Rules

Task 2.A： Protecting the Router

输入以下命令后， ping 10.9.0.11 和 telnet 10.9.0.11 都不通。

```
root@be47739dff58:/# iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
root@be47739dff58:/# iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCE
PT
root@be47739dff58:/# iptables -P OUTPUT DROP
root@be47739dff58:/# iptables -P INPUT DROP
root@4362d2195d99:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
^C
--- 10.9.0.11 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3075ms

root@4362d2195d99:/# telnet 10.9.0.11
Trying 10.9.0.11...
^C
```

修改成这样，可以 ping 通，但是 telnet 不通。

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

iptables -P OUTPUT DROP
iptables -P INPUT DROP
```

设置了 iptables -P OUTPUT DROP 后，二者无法 ping 通，表示丢弃所有外出的包 在单独设置了 iptables -P INPUT DROP，可以发现，router 可以 ping 通其他主机，但是其他主机不可以 通 router，表示所有进入的包都被丢弃了，但是外出的包不受限制

Task 2.B：Protecting the Internal Network

```
iptables -A FORWARD -p icmp --icmp-type echo-request -d 10.9.0.5/24 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type echo-reply -d 192.168.60.0/24 -j ACCEPT
iptables -A FORWARD -p icmp --icmp-type echo-request -d 192.168.60.0/24 -j ACCEPT
iptables -A INPUT -p icmp -j ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT
iptables -P FORWARD DROP
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  anywhere             anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  anywhere             10.9.0.0/24          icmp echo-request
ACCEPT     icmp --  anywhere             192.168.60.0/24      icmp echo-reply
ACCEPT     icmp --  anywhere             192.168.60.0/24      icmp echo-request

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

从外部主机 ping 路由器，可以 ping 通； ping 内部主机不通； telnet 内部主机不通。 内部主机 ping 外部主机，可以 ping 通； telnet 外部主机不通。

Task 2.C：Protecting Internal Servers

```
iptables -A FORWARD -p tcp --dport 23 -d 192.168.60.5 -j ACCEPT
iptables -A FORWARD -p tcp --sport 23 -s 192.168.60.5 -j ACCEPT
iptables -A FORWARD -d 10.9.0.0/24 -j DROP
iptables -A FORWARD -d 192.168.60.0/24 -j DROP
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  anywhere             anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  anywhere             10.9.0.0/24          icmp echo-request
ACCEPT     icmp --  anywhere             192.168.60.0/24      icmp echo-reply
ACCEPT     icmp --  anywhere             192.168.60.0/24      icmp echo-request
ACCEPT     tcp  --  anywhere             host1-192.168.60.5.net-192.168.60.0  tcp dpt:telnet
ACCEPT     tcp  --  host1-192.168.60.5.net-192.168.60.0  anywhere             tcp spt:telnet
DROP       all  --  anywhere             10.9.0.0/24
DROP       all  --  anywhere             192.168.60.0/24

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

从外部主机(10.9.0.5)telnet 192.168.60.5 ，可以连接成功。 从外部主机(10.9.0.5)telnet 192.168.60.6 ，无法连接。 外部主机不能访问内部服务器，内部主机可以访问所有内部服务器，

内部主机不可以访问外部服务器 所有内部主机都运行 telnet 服务器(侦听端口 23)。外部主机只能访问 192.168.60.5 上的 telnet 服务器，不能访问其他内部主机。

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@85267f447274:/# conntrack -L
icmp     1 8 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 i
d=31 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@85267f447274:/# conntrack -L
icmp     1 6 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 i
d=31 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@85267f447274:/# conntrack -L
icmp     1 4 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 i
d=31 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@85267f447274:/# conntrack -L
icmp     1 1 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 i
d=31 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@85267f447274:/# 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只有 20~30 秒之间。

```
root@85267f447274:/# conntrack -L
tcp      6 53 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=51068 dport=9090 src=192.168.60.5 dst=10.9.0.5 s
port=9090 dport=51068 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@85267f447274:/# conntrack -L
tcp      6 52 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=51068 dport=9090 src=192.168.60.5 dst=10.9.0.5 s
port=9090 dport=51068 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@85267f447274:/# conntrack -L
tcp      6 48 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=51068 dport=9090 src=192.168.60.5 dst=10.9.0.5 s
port=9090 dport=51068 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@85267f447274:/# conntrack -L
tcp      6 44 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=51068 dport=9090 src=192.168.60.5 dst=10.9.0.5 s
port=9090 dport=51068 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@85267f447274:/# conntrack -L
tcp      6 38 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=51068 dport=9090 src=192.168.60.5 dst=10.9.0.5 s
port=9090 dport=51068 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

TCP 的连接状态保持时间非常长，大约 430000 秒。

```
root@85267f447274:/# conntrack -L
tcp      6 431997 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=51068 dport=9090 src=192.168.60.5 dst=10.9
.0.5 sport=9090 dport=51068 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@85267f447274:/# conntrack -L
tcp      6 431994 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=51068 dport=9090 src=192.168.60.5 dst=10.9
.0.5 sport=9090 dport=51068 [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

在路由器上利用 iptables 命令和连接跟踪机制，创建过滤规则如下：

```
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
```

从外部主机(10.9.0.5)telnet 192.168.60.5 可以连接成功。

```
root@597b5efa5bf3:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
4644a85d9f8c login: seed
Password:

Login incorrect
4644a85d9f8c login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
```

从外部主机(10.9.0.5)telnet 192.168.0.6 不成功

```
root@597b5efa5bf3:/# telnet 192.168.60.6
Trying 192.168.60.6...
^C
```

从内部主机(192.168.60.5)telnet 10.9.0.5 和 192.168.60.6 ， 连接成功。

```
root@4644a85d9f8c:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
597b5efa5bf3 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
15a777504b7d login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
```

Task 4：Limiting Network Traffic

```
root@137e7980c0ce:/# 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=0.166 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.100 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.104 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=0.106 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=0.091 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.140 ms
^C
--- 192.168.60.5 ping statistics ---
24 packets transmitted, 8 received, 66.6667% packet loss, time 235
61ms
rtt min/avg/max/mdev = 0.091/0.114/0.166/0.023 ms
```

一开始会以正常速度发送，后面每隔 6s 发一个包。 如果只执行第一条命令，会议正常速度发送。

Task 5：Load Balancing

```
iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 3 --packet 0 -j DNAT --to-destination

iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 3 --packet 1 -j DNAT --to-destination

iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 3 --packet 2 -j DNAT --to-destination
```

按顺序 hello_1 被发送到 192.168.60.5 8080 ， hello_2 被发送到 192.168.60.6 8080 ， hello_3 被发送到 192.168.60.7 8080 。

```
root@137e7980c0ce:/# echo hello|nc -u 10.9.0.11 8080
^C
root@137e7980c0ce:/# echo hello1|nc -u 10.9.0.11 8080
root@137e7980c0ce:/# echo hello_1|nc -u 10.9.0.11 8080
^C
root@137e7980c0ce:/# echo hello_2|nc -u 10.9.0.11 8080
^C
root@137e7980c0ce:/# echo hello_3|nc -u 10.9.0.11 8080
^C
root@137e7980c0ce:/# echo hello_4|nc -u 10.9.0.11 8080
root@137e7980c0ce:/# echo hello_4|nc -u 10.9.0.11 8080
^C
root@137e7980c0ce:/# echo hello_5|nc -u 10.9.0.11 8080
^C
root@137e7980c0ce:/# echo hello_6|nc -u 10.9.0.11 8080
root@137e7980c0ce:/# echo hello_6|nc -u 10.9.0.11 8080
^[[A^C
root@137e7980c0ce:/# echo hello_7|nc -u 10.9.0.11 8080
^C
root@137e7980c0ce:/# echo hello_8|nc -u 10.9.0.11 8080
^C
root@137e7980c0ce:/# echo hello_9|nc -u 10.9.0.11 8080
```

```
root@552e72b0412e:/#
hello_1
```

```
root@d998244af73c:/#
hello
hello_2
```

```
root@49e64dea0623:/# nc -luk 8080
hello_3
```

```
root@e94e2533f8f6:/# iptables -F
root@e94e2533f8f6:/# 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
root@e94e2533f8f6:/# 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
root@e94e2533f8f6:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m
statistic --mode random --probability 0.33 -j DNAT --to-destination
192.168.60.7:8080
```

虽然是等概率发送数据，但每个主机收到的数量各不相同，甚至有的差异较大，当样本数量足够多时， 应该是趋于平均的

```
root@d998244af73c:/# nc -luk 8080
hello_1
hello_4
hello_5
hello_7
hello_9
```

```
root@552e72b0412e:/# nc -luk 8080
hello_2
hello_3
hello_8
```

```
root@49e64dea0623:/# nc -luk 8080
hello_6
```