Firewall Exploration Lab

57118212 晏宇珂

Task 1: Implementing a Simple Firewall

Task 1.A: Implementing a Simple Kernel Module

编译内核模块`hello`，运行并查看消息：

```
[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
[07/22/21]seed@VM:~/kernel_module$ dmesg
[  358.393669] hello: loading out-of-tree module taints kernel.
[  358.393676] hello: module license 'unspecified' taints kernel.
[  358.393678] Disabling lock debugging due to kernel taint
[  358.393738] hello: module verification failed: signature and/or required key
missing - tainting kernel
[  358.394705] Hello World!
[  373.966090] Bye-bye World!.
```

Task 1.B: Implementing a Simple Firewall Using Netfilter

1. 编译`seedFilter`，代码改成百度的`DNS`服务器`180.76.76.76`，打开防火墙后进行查询，发现查询不到百度的地址，可以看到`drop`消息：

```
[07/24/21]seed@VM:~/packet_filter$ dig @180.76.76.76 www.baidu.com

; <<>> DiG 9.16.1-Ubuntu <<>> @180.76.76.76 www.baidu.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

```
[  243.933809] *** LOCAL_OUT
[  243.933813]     192.168.43.148  --> 180.76.76.76 (UDP)
[  243.933827] *** Dropping 180.76.76.76 (UDP), port 53
[  248.931985] *** LOCAL_OUT
[  248.931990]     192.168.43.148  --> 180.76.76.76 (UDP)
[  248.932015] *** Dropping 180.76.76.76 (UDP), port 53
[  253.939217] *** LOCAL_OUT
[  253.939223]     192.168.43.148  --> 180.76.76.76 (UDP)
[  253.939248] *** Dropping 180.76.76.76 (UDP), port 53
[  265.278222] The filters are being removed.
```

去除该内核模块可以正常查询：

```
[07/24/21]seed@VM:~/packet_filter$ dig @180.76.76.76 www.baidu.com

; <<>> DiG 9.16.1-Ubuntu <<>> @180.76.76.76 www.baidu.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7757
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 5, ADDITIONAL: 6

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.baidu.com.                 IN      A

;; ANSWER SECTION:
www.baidu.com.          114     IN      CNAME   www.a.shifen.com.
www.a.shifen.com.       240     IN      A       180.101.49.11
www.a.shifen.com.       240     IN      A       180.101.49.12

;; AUTHORITY SECTION:
a.shifen.com.           988     IN      NS      ns2.a.shifen.com.
```

2. 先把 `printInfo` 函数挂到所有 `netfilter` 的 `hook` 上：

```c
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 = printInfo;
   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 = printInfo;
   hook4.hooknum = NF_INET_FORWARD;
   hook4.pf = PF_INET;
   hook4.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook4);

   hook5.hook = printInfo;
   hook5.hooknum = NF_INET_PRE_ROUTING;
   hook5.pf = PF_INET;
   hook5.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook5);

   hook6.hook = printInfo;
   hook6.hooknum = NF_INET_POST_ROUTING;
   hook6.pf = PF_INET;
```

```
    hook6.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook6);

    return 0;
}
```

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)
```

`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，决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会调用

`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

3. 实现两个钩子函数 `prevent_ping` 和 `prevent_telnet`：

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

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

```
    return NF_ACCEPT;
}

unsigned int prevent_telnet(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;
}
```

把两个函数注册到同一个`hook`，我选的是`LOCAL_IN`(`ping`的时候不会调用`POST_ROUTING`):

```
    hook2.hook = prevent_ping;
    hook2.hooknum = NF_INET_LOCAL_IN;
    hook2.pf = PF_INET;
    hook2.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook2);

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

进入`10.9.0.5`发现无法`ping`或`telnet`到`10.9.0.1`:

```
root@7d9aaa6891bd:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
```

```
root@7d9aaa6891bd:/# telnet 10.9.0.1
Trying 10.9.0.1...
```

`dmesg`中也能看见`drop`信息：

```
[  157.297725] *** Dropping 10.9.0.1 (ICMP)
[  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)
[  162.414844] *** Dropping 10.9.0.1 (ICMP)
[  163.441212] *** Dropping 10.9.0.1 (ICMP)
[  211.237488] *** Dropping 10.9.0.1 (TCP), port 23
[  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
```

- **Important note**

  一定要注意！写1.B崩溃了之后才发现忘记取消注册钩子了。

Task 2: Experimenting with Stateless Firewall Rules

Task 2.A: Protecting the Router

在路由器上执行`iptables`命令：

```
#允许其他机器ping通防火墙
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
#设置filter表的OUTPUT链和INPUT链的默认策略为丢包，不接受任何连接
iptables -P OUTPUT DROP
iptables -P INPUT DROP
```

在`10.9.0.5`上：

(1) `ping`通路由器

```
root@7d9aaa6891bd:/# 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=0.324 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.127 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.126 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.370 ms
```

(2) telnet不到路由器

```
root@7d9aaa6891bd:/# telnet 10.9.0.11
Trying 10.9.0.11...
```

Task 2.B: Protecting the Internal Network

在路由器上执行以下命令：

```
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@7d9aaa6891bd:/# 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 `ping` 通路由器：

```
root@7d9aaa6891bd:/# 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=0.060 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.066 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.129 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.132 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.104 ms
64 bytes from 10.9.0.11: icmp_seq=6 ttl=64 time=0.131 ms
64 bytes from 10.9.0.11: icmp_seq=7 ttl=64 time=0.122 ms
64 bytes from 10.9.0.11: icmp_seq=8 ttl=64 time=0.137 ms
64 bytes from 10.9.0.11: icmp_seq=9 ttl=64 time=0.121 ms
64 bytes from 10.9.0.11: icmp_seq=10 ttl=64 time=0.121 ms
64 bytes from 10.9.0.11: icmp_seq=11 ttl=64 time=0.049 ms
```

内部主机可以 `ping` 通外部主机：

```
root@f67a5610d8e0:/# 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=0.108 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.150 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=0.148 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.151 ms
```

内网和外网之间的其他所有数据包都被阻止：

```
root@7d9aaa6891bd:/# telnet 192.168.60.5
Trying 192.168.60.5...
```

```
root@f67a5610d8e0:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

Task 2.C: Protecting Internal Servers

在路由器上执行以下命令：

```
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
```

外部主机只能`telnet`到`192.168.60.5`上的服务器：

```
root@7d9aaa6891bd:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
f67a5610d8e0 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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sat Jul 24 16:12:28 UTC 2021 on pts/2
seed@f67a5610d8e0:~$
```

不能访问其他内网服务器：

```
root@7d9aaa6891bd:/# telnet 192.168.60.6
Trying 192.168.60.6...
```

```
root@7d9aaa6891bd:/# 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@f67a5610d8e0:/# telnet 192.168.60.7
Trying 192.168.60.7...
Connected to 192.168.60.7.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
58745f4944b8 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5
```

```
root@f67a5610d8e0:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
0da6814ec334 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5
```

内部主机无法访问外部服务器：

```
root@f67a5610d8e0:/# 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`

  大概测试了一下，连接状态持续30秒。

  `conntrack -L` 能看到 `icmp` 后面第二个数字从29开始变小直到0：

```
root@52b59d815de5:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=35 src=192.168.60.5
 dst=10.9.0.5 type=0 code=0 id=35 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@52b59d815de5:/# conntrack -L
icmp     1 27 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=35 src=192.168.60.5
 dst=10.9.0.5 type=0 code=0 id=35 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@52b59d815de5:/# conntrack -L
icmp     1 18 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=35 src=192.168.60.5
 dst=10.9.0.5 type=0 code=0 id=35 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@52b59d815de5:/# conntrack -L
icmp     1 7 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=35 src=192.168.60.5
dst=10.9.0.5 type=0 code=0 id=35 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@52b59d815de5:/# conntrack -L
icmp     1 0 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=35 src=192.168.60.5
dst=10.9.0.5 type=0 code=0 id=35 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@52b59d815de5:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
```

- `UDP`

  大约持续：

```
root@52b59d815de5:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
root@52b59d815de5:/# conntrack -L
udp      17 29 src=10.9.0.5 dst=192.168.60.5 sport=35521 dport=9090 [UNREPLIED]
src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=35521 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

- `TCP`

  大约持续432000秒：

```
root@52b59d815de5:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
root@52b59d815de5:/# conntrack -L
tcp      6 431999 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=44482 dport=90
90 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=44482 [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

在路由器中执行如下命令：

```
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 -p tcp -i eth0 -o eth1 --dport 23 -d 192.168.60.5 --syn -
m conntrack --ctstate NEW -j ACCEPT
iptables -A FORWARD -p tcp -i eth1 -o eth0 -m conntrack --ctstate
NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -p tcp -i eth0 -o eth1 -m conntrack --ctstate
ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -j DROP
```

其他情况都和2.C一样，但内网能`telnet`上外网了：

```
root@f67a5610d8e0:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
7d9aaa6891bd login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Lir
```

Task 4: Limiting Network Traffic

在路由器执行以下命令：

```
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@7d9aaa6891bd:/# 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=0.380 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.154 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.157 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.158 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.161 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.151 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.196 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.136 ms
```

如果只执行第一条命令：

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

从`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。

`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
```

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。

（其实我两个实验里都是发给`192.168.60.5`的报文偏多，我觉得是因为一次性发送的报文偏少，但是我写脚本发送之后又会阻塞，懒得写惹，先这样吧）