哈尔滨工业大学(深圳)

《网络与系统安全》
实验报告

实验六

防火墙 实验

学　院: 计算机科学与技术学院

姓　名: 王志铭

学　号: 200110611

专　业: 计算机科学与技术

日　期: 2023 年 4 月

1. Task1：加载 seedFilter 模块，执行 dig dig @8.8.8.8 www.example.com，卸载 seedFilter 后再执行 dmesg 命令查看内核日志，把日志信息中加载、卸载 seefFilter 模块以及阻止 UDP 数据包的信息截图，并进行分析说明。

```
[  652.933717] Bye-bye World!.
[  787.119276] Registering filters.
[  808.173220] *** LOCAL_OUT
[  808.173224]     127.0.0.1  --> 127.0.0.1 (UDP)
[  808.173885] *** LOCAL_OUT
[  808.173887]     10.0.2.15  --> 8.8.8.8 (UDP)
[  808.173901] *** Dropping 8.8.8.8 (UDP), port 53
[  813.172512] *** LOCAL_OUT
[  813.172516]     10.0.2.15  --> 8.8.8.8 (UDP)
[  813.172529] *** Dropping 8.8.8.8 (UDP), port 53
[  813.240862] *** LOCAL_OUT
[  813.240865]     10.0.2.15  --> 91.189.94.4 (UDP)
[  818.132704] *** LOCAL_OUT
[  818.132706]     10.0.2.15  --> 8.8.8.8 (UDP)
[  818.132718] *** Dropping 8.8.8.8 (UDP), port 53
[  820.440581] *** LOCAL_OUT
[  820.440584]     10.0.2.15  --> 172.20.10.1 (UDP)
[  820.837559] *** LOCAL_OUT
[  820.837561]     127.0.0.1  --> 127.0.0.53 (UDP)
[  823.983735] *** LOCAL_OUT
[  823.983737]     127.0.0.53  --> 127.0.0.1 (UDP)
[  828.106626] *** LOCAL_OUT
[  828.106629]     127.0.0.1  --> 127.0.0.53 (UDP)
[  828.106813] *** LOCAL_OUT
[  828.106815]     127.0.0.53  --> 127.0.0.1 (UDP)
[  835.794249] The filters are being removed.
```

如上图所示，可以看到，在加载了 seedFilter 模块之后，所有的 ip 地址为 8.8.8.8，端口为 53 的 udp 包都被防火墙拦截、丢弃了。

2. Task2：阻止 TCP 端口和 PING，把增加和修改的代码截图，并在卸载模块后将 dmesg 的日志信息的截图，并分析说明原因。

阻挡 ICMP 和 TCP 的函数与 BlockUDP 类似，如下图所示。

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

   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){
          printk(KERN_WARNING "*** Dropping %pI4 (TCP), port %d\n", &(iph->daddr), ntohs(tcph->dest));
          return NF_DROP;
       }
   }
   return NF_ACCEPT;
}
```

```c
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;
}
```

需要注意的是，如果希望获得 icmp 的报头，需要引入<linux/icmp.h>头文件。

注册和移除钩子也是与 udp 类似。

```
    hook2.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook2);

    hook3.hook = blockICMP;
    hook3.hooknum = NF_INET_POST_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_POST_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);
}
```

修改完之后再 ping / telnet，然后查看日志，可以看到以下结果。

```
[  828.106815]     127.0.0.53  --> 127.0.0.1 (UDP)
[  835.794249] The filters are being removed.
[ 2280.484521] Registering filters.
[ 2292.770866] *** LOCAL_OUT
[ 2292.770868]     10.9.0.1  --> 10.9.0.1 (ICMP)
[ 2292.771035] *** Dropping 10.9.0.1 (ICMP)
[ 2293.786914] *** LOCAL_OUT
[ 2293.786917]     10.9.0.1  --> 10.9.0.1 (ICMP)
[ 2293.786931] *** Dropping 10.9.0.1 (ICMP)
[ 2294.809285] *** LOCAL_OUT
[ 2294.809288]     10.9.0.1  --> 10.9.0.1 (ICMP)
[ 2294.809302] *** Dropping 10.9.0.1 (ICMP)
[ 2295.830991] *** LOCAL_OUT
[ 2295.830993]     10.9.0.1  --> 10.9.0.1 (ICMP)
[ 2295.831004] *** Dropping 10.9.0.1 (ICMP)
[ 2296.854794] *** LOCAL_OUT
[ 2296.854798]     10.9.0.1  --> 10.9.0.1 (ICMP)
[ 2296.854818] *** Dropping 10.9.0.1 (ICMP)
[ 2297.876709] *** LOCAL_OUT
```

```
[ 2502.875779]     10.9.0.1  --> 10.9.0.1 (TCP)
[ 2502.875787] *** Dropping 10.9.0.1 (TCP), port 23
[ 2509.640521] *** LOCAL_OUT
[ 2509.640523]     10.9.0.1  --> 10.9.0.1 (TCP)
[ 2509.640529] *** Dropping 10.9.0.1 (TCP), port 23
[ 2509.811628] *** LOCAL_OUT
[ 2509.811630]     10.9.0.1  --> 10.9.0.1 (TCP)
[ 2509.811637] *** Dropping 10.9.0.1 (TCP), port 23
[ 2510.757208] *** LOCAL_OUT
[ 2510.757307]     10.9.0.1  --> 10.9.0.1 (TCP)
[ 2510.757410] *** Dropping 10.9.0.1 (TCP), port 23
[ 2530.625623] The filters are being removed.
[06/08/23]seed@VM:~/.../packet_filter$
```

可以看到，防火墙成功阻挡了 tcp 和 icmp 协议的数据包。

3. Task3：保护 Router，将配置 iptables 规则前后 ping 和 telnet 的连通性测试结果截图，并分析说明原因。

配置规则前：

```
root@dad68bcc7448:/# 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=0.644 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.065 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.083 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.055 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.066 ms
^C
--- 10.9.0.11 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4063ms
rtt min/avg/max/mdev = 0.055/0.182/0.644/0.230 ms
root@dad68bcc7448:/# telnet 10.9.0.11
Trying 10.9.0.11...
Connected to 10.9.0.11.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
1a53968741ac login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

配置规则后：

```
root@dad68bcc7448:/# 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=0.052 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.074 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.071 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.054 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.061 ms
^C
--- 10.9.0.11 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4091ms
rtt min/avg/max/mdev = 0.052/0.062/0.074/0.008 ms
root@dad68bcc7448:/# telnet 10.9.0.11
Trying 10.9.0.11...
telnet: Unable to connect to remote host: Connection timed out
root@dad68bcc7448:/#
```

配置规则前，由于防火墙没有限制，所以 ping 和 telnet 都能通。

配置规则后，限制了 icmp 协议能出入，而其他协议的包会被丢弃。因此可以 ping，但是不能 telnet（tcp）。

4、Task4：保护内网，将配置 iptables 规则前后 ping 的连通性测试结果截图，并分析说明原因。

HostA ping / telnet 内网：

```
root@dad68bcc7448:/# 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 ---
40 packets transmitted, 0 received, 100% packet loss, time 39933ms

root@dad68bcc7448:/# telnet 192.168.60.5
Trying 192.168.60.5...
telnet: Unable to connect to remote host: Connection timed out
```

Host1 ping 内网和外网：

```
[06/08/23]seed@VM:~$ docksh 01
root@0178eb44298f:/# ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.101 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.078 ms
64 bytes from 192.168.60.11: icmp_seq=3 ttl=64 time=0.070 ms
64 bytes from 192.168.60.11: icmp_seq=4 ttl=64 time=0.053 ms
64 bytes from 192.168.60.11: icmp_seq=5 ttl=64 time=0.069 ms
^C
--- 192.168.60.11 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4086ms
rtt min/avg/max/mdev = 0.053/0.074/0.101/0.015 ms
root@0178eb44298f:/# 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=1.12 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.083 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.115 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.068 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.073 ms
^C
--- 10.9.0.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4075ms
rtt min/avg/max/mdev = 0.068/0.290/1.115/0.412 ms
```

可以看到，配置规则后外网不能 ping 通内网，内网可能 ping 通外网、内网。这是因为我们设置了防火墙规则为：

iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-request -j DROP

iptables -A FORWARD -i eth1 -p icmp --icmp-type echo-request -j ACCEPT

iptables -A FORWARD -p icmp --icmp-type echo-reply -j ACCEPT

iptables -P FORWARD DROP

即

1、外网不能 ping 通内网

2、外网可以 ping 通 Router

3、内网可以 ping 通外网

4、所有其他的内网和外网交互的数据包被阻止掉。