Name : Linson Peter Rodrigues

Lab 13 : Firewall Exploration Lab

Container setup

```
[12/01/23]seed@VM:~/.../Labsetup$ docker-compose build
HostA uses an image, skipping
Host1 uses an image, skipping
Host2 uses an image, skipping
Host3 uses an image, skipping
Building Router
Step 1/2 : FROM handsonsecurity/seed-ubuntu:large
 ---> cecb04fbf1dd
Step 2/2 : RUN apt-get update     && apt-get install -y kmod      && apt-get clean
 ---> Using cache
 ---> f66666ef3fd3

Successfully built f66666ef3fd3
Successfully tagged seed-router-image:latest
[12/01/23]seed@VM:~/.../Labsetup$ dcup
Creating network "net-192.168.60.0" with the default driver
WARNING: Found orphan containers (attacker-ns-10.9.0.153, user-10.9.0.5, user2-10.9.0.7, user1-10.9.0.6, victim-10.9.0.5, seed-attacker, loca
l-dns-server-10.9.0.53) for this project. If you removed or renamed this service in your compose file, you can run this command with the --re
move-orphans flag to clean it up.
Recreating seed-router      ... done
Creating host3-192.168.60.7 ... done
Creating host1-192.168.60.5 ... done
Creating host2-192.168.60.6 ... done
Creating hostA-10.9.0.5     ... done
Attaching to host1-192.168.60.5, hostA-10.9.0.5, host3-192.168.60.7, host2-192.168.60.6, seed-router
host1-192.168.60.5 |  * Starting internet superserver inetd              [ OK ]
host2-192.168.60.6 |  * Starting internet superserver inetd              [ OK ]
hostA-10.9.0.5 |  * Starting internet superserver inetd                 [ OK ]
host3-192.168.60.7 |  * Starting internet superserver inetd              [ OK ]
seed-router |  * Starting internet superserver inetd                    [ OK ]
```

```
[12/01/23]seed@VM:~/.../Labsetup$ dockps
aa2bcc3305a0  hostA-10.9.0.5
3c97701457f8  host2-192.168.60.6
148af05eb5dc  seed-router
176aa5fea341  host1-192.168.60.5
1311f7344a29  host3-192.168.60.7
[12/01/23]seed@VM:~/.../Labsetup$
```

3.2 Task1.B: Implement a simple Firewall Using Netfilter

Make File:

Makefile

```
# obj-m += seedFilter.o
# obj-m += seedPrint.o task 2b
# obj-m += seedBlock.o task 2c
all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean

ins:
	sudo dmesg -C
	sudo insmod seedFilter.ko

rm:
	sudo rmmod seedFilter

```

1. How Firewall works

We need to ping from **hostA-10.9.0.5** to **10.9.0.1** as well as **telnet 10.9.0.1** from **hostA-10.9.0.5**

```
[12/06/23]seed@VM:~/.../Labsetup$ docksh hostA-10.9.0.5
root@aa2bcc3305a0:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
64 bytes from 10.9.0.1: icmp_seq=1 ttl=64 time=0.166 ms
64 bytes from 10.9.0.1: icmp_seq=2 ttl=64 time=0.068 ms
64 bytes from 10.9.0.1: icmp_seq=3 ttl=64 time=0.060 ms
^C
--- 10.9.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2048ms
rtt min/avg/max/mdev = 0.060/0.098/0.166/0.048 ms
root@aa2bcc3305a0:/# telnet 10.9.0.1
Trying 10.9.0.1...
Connected to 10.9.0.1.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
VM 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

515 updates can be installed immediately.
515 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sat Nov 18 17:53:31 EST 2023 from VM on pts/7
```

```
[12/05/23]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Downloads/Labsetup/Files/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedPrint.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedPrint.mod.o
  LD [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedPrint.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[12/05/23]seed@VM:~/.../packet_filter$ ls
Makefile       Module.symvers  seedFilter.ko   seedFilter.mod.c  seedFilter.o  seedPrint.ko   seedPrint.mod.c  seedPrint.o
modules.order  seedFilter.c    seedFilter.mod  seedFilter.mod.o  seedPrint.c   seedPrint.mod  seedPrint.mod.o
[12/05/23]seed@VM:~/.../packet_filter$ sudo insmod seedPrint.ko
```

Output:

```
[ 3431.751218] Registering filters.
[ 3545.684271] *** LOCAL_OUT
[ 3545.684276]     10.0.2.15  --> 192.168.0.1 (UDP)
[ 3564.867308] *** LOCAL_OUT
[ 3564.867310]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 3564.867444] *** LOCAL_OUT
```

```
[12/05/23]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

[12/05/23]seed@VM:~/.../packet_filter$ sudo rmmod seedFilter
[12/05/23]seed@VM:~/.../packet_filter$
```

```
[ 5132.965729] *** Dropping 8.8.8.8 (UDP), port 53
[ 5135.635901] *** LOCAL_OUT
[ 5135.635903]     10.0.2.15  --> 192.168.0.1 (UDP)
[ 5135.663900] *** LOCAL_OUT
[ 5135.663902]     10.0.2.15  --> 185.125.190.17 (TCP)
[ 5135.790117] *** LOCAL_OUT
[ 5135.790143]     10.0.2.15  --> 185.125.190.17 (TCP)
[ 5135.790211] *** LOCAL_OUT
[ 5135.790213]     10.0.2.15  --> 185.125.190.17 (TCP)
[ 5135.886575] *** LOCAL_OUT
[ 5135.886578]     10.0.2.15  --> 185.125.190.17 (TCP)
[ 5135.886704] *** LOCAL_OUT
[ 5135.886705]     10.0.2.15  --> 185.125.190.17 (TCP)
[ 5135.889602] *** LOCAL_OUT
[ 5135.889604]     10.0.2.15  --> 185.125.190.17 (TCP)
[ 5137.968107] *** LOCAL_OUT
[ 5137.968109]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 5137.968118] *** Dropping 8.8.8.8 (UDP), port 53
[ 5198.022240] The filters are being removed.
```

The output above demonstrates that you may **ping 10.9.0.1** prior to introducing the module; however, UDP packet drops occur when the module is introduced since the firewall is blocking the request.

2. The code: Invoking Hook function

```c
#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, hook5;


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);
```

```
34        if (iph->daddr == ip_addr && ntohs(udph->dest) == port){
35             printk(KERN_WARNING "*** Dropping %pI4 (UDP), port %d\n", &(iph->daddr), port);
36             return NF_DROP;
37         }
38    }
39    return NF_ACCEPT;
40 }
41
42 unsigned int printInfo(void *priv, struct sk_buff *skb,
43                  const struct nf_hook_state *state)
44 {
45    struct iphdr *iph;
46    char *hook;
47    char *protocol;
48
49    switch (state->hook){
50      case NF_INET_LOCAL_IN:     hook = "LOCAL_IN";     break;
51      case NF_INET_LOCAL_OUT:    hook = "LOCAL_OUT";    break;
52      case NF_INET_PRE_ROUTING:  hook = "PRE_ROUTING";  break;
53      case NF_INET_POST_ROUTING: hook = "POST_ROUTING"; break;
54      case NF_INET_FORWARD:      hook = "FORWARD";      break;
55      default:                   hook = "IMPOSSIBLE";   break;
56    }
57    printk(KERN_INFO "*** %s\n", hook); // Print out the hook info
58
59    iph = ip_hdr(skb);
60    switch (iph->protocol){
61      case IPPROTO_UDP:  protocol = "UDP";   break;
62      case IPPROTO_TCP:  protocol = "TCP";   break;
63      case IPPROTO_ICMP: protocol = "ICMP";  break;
64      default:           protocol = "OTHER"; break;
65
66    }
```

```
67    // Print out the IP addresses and protocol
68    printk(KERN_INFO "     %pI4  --> %pI4 (%s)\n",
69                     &(iph->saddr), &(iph->daddr), protocol);
70
71    return NF_ACCEPT;
72 }
73
74
75 int registerFilter(void) {
76    printk(KERN_INFO "seedPrint: Registering filters.\n");
77
78          // NF_INET_PRE_ROUTING
79    hook1.hook = printInfo;
80    hook1.hooknum = NF_INET_PRE_ROUTING;
81    hook1.pf = PF_INET;
82    hook1.priority = NF_IP_PRI_FIRST;
83    nf_register_net_hook(&init_net, &hook1);
84          // NF_INET_LOCAL_IN
85    hook2.hook = printInfo;
86    hook2.hooknum = NF_INET_LOCAL_IN;
87    hook2.pf = PF_INET;
88    hook2.priority = NF_IP_PRI_FIRST;
89    nf_register_net_hook(&init_net, &hook2);
90          // NF_INET_FORWARD
```

```
 91    hook3.hook = printInfo;
 92    hook3.hooknum = NF_INET_FORWARD;
 93    hook3.pf = PF_INET;
 94    hook3.priority = NF_IP_PRI_FIRST;
 95    nf_register_net_hook(&init_net, &hook3);
 96          // NF_INET_LOCAL_OUT
 97    hook4.hook = printInfo;
 98    hook4.hooknum = NF_INET_LOCAL_OUT;
 99    hook4.pf = PF_INET;
100    hook4.priority = NF_IP_PRI_FIRST;
101    nf_register_net_hook(&init_net, &hook4);
102          // NF_INET_POST_ROUTING
103    hook5.hook = printInfo;
104    hook5.hooknum = NF_INET_POST_ROUTING;
105    hook5.pf = PF_INET;
106    hook5.priority = NF_IP_PRI_FIRST;
107    nf_register_net_hook(&init_net, &hook5);
108
109    return 0;
110 }
111
112 void removeFilter(void) {
113    printk(KERN_INFO "seedPrint: The filters are being removed.\n");
114    nf_unregister_net_hook(&init_net, &hook1);
115    nf_unregister_net_hook(&init_net, &hook2);
116    nf_unregister_net_hook(&init_net, &hook3);
117    nf_unregister_net_hook(&init_net, &hook4);
118    nf_unregister_net_hook(&init_net, &hook5);
119 }
120
121 module_init(registerFilter);
122 module_exit(removeFilter);
```

```
[12/05/23]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Downloads/Labsetup/Files/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedPrint.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedPrint.mod.o
  LD [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedPrint.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[12/05/23]seed@VM:~/.../packet_filter$ ls
Makefile       Module.symvers  seedFilter.ko   seedFilter.mod.c  seedFilter.o  seedPrint.ko   seedPrint.mod.c  seedPrint.o
modules.order  seedFilter.c    seedFilter.mod  seedFilter.mod.o  seedPrint.c   seedPrint.mod  seedPrint.mod.o
[12/05/23]seed@VM:~/.../packet_filter$ sudo insmod seedPrint.ko
```

```
[12/05/23]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: 60937
;; 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.        3615    IN      A       93.184.216.34

;; Query time: 16 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue Dec 05 22:00:08 EST 2023
;; MSG SIZE  rcvd: 60

[12/05/23]seed@VM:~/.../packet_filter$
```

Output:

```
[ 2461.454628] seedPrint: Registering filters.
[ 2635.875665] *** LOCAL_OUT
[ 2635.875669]     10.0.2.15  --> 192.168.0.1 (UDP)
[ 2635.875682] *** POST_ROUTING
[ 2635.875684]     10.0.2.15  --> 192.168.0.1 (UDP)
[ 2635.889582] *** PRE_ROUTING
[ 2635.889586]     192.168.0.1  --> 10.0.2.15 (UDP)
[ 2635.889599] *** LOCAL_IN
[ 2635.889600]     192.168.0.1  --> 10.0.2.15 (UDP)
[ 2653.269728] *** LOCAL_OUT
```

Here, we can observe that the code that is printed above is called during PRE_Routing while the firewall policy is active.

Netfilter offers a number of supported hooks. They are defined in the header file **(#include <linux/netfilter_ipv4.h> or #include <linux/netfilter_ipv6.h>)** that is unique to a given protocol. For IPv4, there are five defined hooks.

1. NF_INET_PRE_ROUTING

Before inputting the routing code, incoming packets are examined. After that, use this hook function. This hook allows for the detection of several assaults, such as denial-of-service attempts, before the packets are handled.

2. NF_INET_LOCAL_IN

The packets intended for this machine pass the hook function after passing via the routing code.

3. NF_INET_FORWARD

The packets that are not meant for this computer flow through the hook function after passing through the routing code.

4. NP_INET_LOCAL_OUT

The local computer's generated packets proceed to the hook function. To filter outgoing packets, these packets are compared to the set of rules.

5. NF_INET_POST_ROUTING

The hook function checks the packets before they leave the local computer's NIC card.

3. Code: Implement two more hooks

```c
#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;

// block udp to 8.8.8.8:53

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;
}
```

```c
// block ping to vm:10.9.0.1

unsigned int blockICMP(void *priv, struct sk_buff *skb,
                       const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct icmphdr *icmph;

   //u16  port   = 53; //DNS
   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 && icmph->type == ICMP_ECHO){
            printk(KERN_WARNING "*** Dropping %pI4 (ICMP)\n", &(iph->daddr));
            return NF_DROP;
        }
   }
   return NF_ACCEPT;
}


// block telnet to vm:10.9.0.1:23

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

   u16  port   = 23; //Telnet
   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;
}
```

```c
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;
```

```c
int registerFilter(void) {
   printk(KERN_INFO "seedBlock: 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 = blockTelnet;
   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;
}
```

```c
void removeFilter(void) {
   printk(KERN_INFO "seedBlock: 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");
```

```
[12/06/23]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Downloads/Labsetup/Files/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedBlock.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedBlock.mod.o
  LD [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedBlock.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[12/06/23]seed@VM:~/.../packet_filter$ sudo insmod seedBlock.ko
[12/06/23]seed@VM:~/.../packet_filter$
```

```
root@aa2bcc3305a0:/# 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 ---
4 packets transmitted, 0 received, 100% packet loss, time 3081ms

root@aa2bcc3305a0:/# telnet 10.9.0.1
Trying 10.9.0.1...
^C
root@aa2bcc3305a0:/#
```

Output:

```
[ 8573.244592] seedBlock: Registering filters.
[ 8597.017428] *** Dropping 10.9.0.1 (ICMP)
[ 8598.043001] *** Dropping 10.9.0.1 (ICMP)
[ 8599.063198] *** Dropping 10.9.0.1 (ICMP)
[ 8600.098201] *** Dropping 10.9.0.1 (ICMP)
[ 8610.531660] *** Dropping 10.9.0.1 (TCP), port 23
[ 8611.542855] *** Dropping 10.9.0.1 (TCP), port 23
[ 8613.558796] *** Dropping 10.9.0.1 (TCP), port 23
```

We can see that ICMP, TCP, and UDP packets are lost when the block module is introduced since the firewall policy prevents all of the services listed in the preceding code, allowing us to connect to **10.9.0.1** from **host A-10.9.0.5**.