Name: Devin Sunil Lilaramani
CU ID: C15309802
CPSC 8570

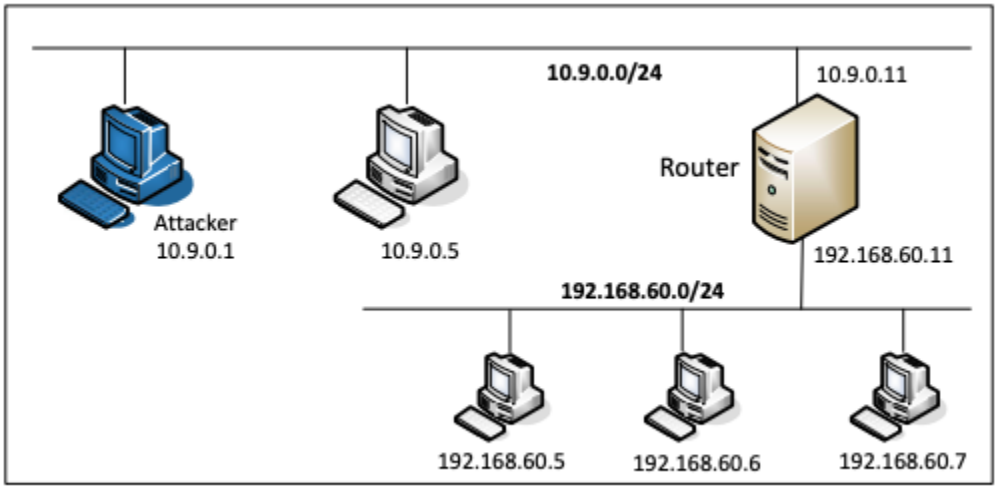
LAB 5 Report - Firewall Exploration Lab

Introduction: The learning objective of this lab is two-fold: learning how firewalls work, and setting up a simple firewall for a network.We will first implement a simple stateless packet-filtering firewall, which inspects packets, and decides whether to drop or forward a packet based on firewall rules. Through this implementation task, we can get the basic ideas on how a firewall works. Actually, Linux already has a built-in firewall, also based on Netfilter. This firewall is called iptables.

This Lab covers the following topics:

- Firewall
- Netfilter

Labsetup:



Task 1: Implementing a Simple Firewall: In this work, we'll build a simple packet filtering firewall that inspects all incoming and outgoing packets and enforces the administrator's firewall settings. Because packet processing takes place within the kernel, filtering must likewise take place there. As a result, it appears that creating such a firewall will necessitate changes to the Linux kernel. Previously, this required altering and rebuilding the kernel. Modern Linux operating systems have a number of additional ways for manipulating packets without having to rebuild the kernel image. The Loadable Kernel Module (LKM) and Netfilter are the two techniques.

Task 1.A: Implement a Simple Kernel Module: We need to first run Make. The purpose of the make utility is to determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them.

We can view the output of the make command by executing an ls command followed by the make.

LKM allows us to add a new module to the kernel at the runtime. This new module enables us to extend the functionalities of the kernel, without rebuilding the kernel or even rebooting the computer. The packet filtering part of a firewall can be implemented as an LKM. In this task, we will get familiar with LKM

The following is a simple loadable kernel module. It prints out "Hello World!" when the module is loaded; when the module is removed from the kernel, it prints out "Bye-bye World!". The messages are not printed out on the screen; they are actually printed into the /var/log/syslog file. You can use "dmesg" to view the messages.

```
seed@VM: ~/.../kernel_module
hello.c  Makefile
[04/22/22]seed@VM:~/.../kernel_module$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Downloads/Labsetup/File
s/kernel_module modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Downloads/Labsetup/Files/kernel_module/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Downloads/Labsetup/Files/kernel_module/hello.mod.o
  LD [M]  /home/seed/Downloads/Labsetup/Files/kernel_module/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[04/22/22]seed@VM:~/.../kernel_module$ ls
hello.c   hello.mod    hello.mod.o  Makefile       Module.symvers
hello.ko  hello.mod.c  hello.o      modules.order
[04/22/22]seed@VM:~/.../kernel_module$ lsmod
Module                  Size  Used by
xt_conntrack           16384  4
xt_MASQUERADE          20480  4
nf_conntrack_netlink    45056  0
nfnetlink              16384  2 nf_conntrack_netlink
xfrm_user              36864  1
xfrm_algo              16384  1 xfrm_user
xt_addrtype            16384  2
iptable_filter         16384  1
```

```
seed@VM: ~/.../kernel_module
drm                    491520  6 vmwgfx,drm_kms_helper,ttm
ip_tables               32768  2 iptable_filter,iptable_nat
x_tables                40960  5 xt_conntrack,iptable_filter,xt_addrtype,ip_table
s,xt_MASQUERADE
autofs4                 45056  2
hid_generic             16384  0
usbhid                  57344  0
hid                    131072  2 usbhid,hid_generic
crc32_pclmul            16384  0
ahci                    40960  2
psmouse                155648  0
i2c_piix4               28672  0
libahci                 32768  1 ahci
e1000                  147456  0
pata_acpi               16384  0
video                   49152  0
[04/22/22]seed@VM:~/.../kernel_module$ lsmod | grep hello
[04/22/22]seed@VM:~/.../kernel_module$ ls
hello.c   hello.mod    hello.mod.o  Makefile       Module.symvers
hello.ko  hello.mod.c  hello.o      modules.order
[04/22/22]seed@VM:~/.../kernel_module$ sudo insmod hello.ko
[04/22/22]seed@VM:~/.../kernel_module$ lsmod | grep hello
hello                   16384  0
[04/22/22]seed@VM:~/.../kernel_module$
```

In the below screenshot we can see the message "***Hello World!***" on running the command "***dmesg***".

```
seed@VM: ~/.../kernel_module
[   13.978210] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[   14.004101] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   16.258349] aufs 5.4.3-20200302
[   20.163391] kauditd_printk_skb: 28 callbacks suppressed
[   20.163393] audit: type=1400 audit(1650667326.623:40): apparmor="STATUS" oper
ation="profile_load" profile="unconfined" name="docker-default" pid=1167 comm="a
pparmor_parser"
[   21.437961] bridge: filtering via arp/ip/ip6tables is no longer available by
default. Update your scripts to load br_netfilter if you need this.
[   21.448739] Bridge firewalling registered
[   21.517947] Started bpfilter
[   21.518259] bpfilter: Loaded bpfilter_umh pid 1205
[   22.387398] Initializing XFRM netlink socket
[   24.458634] rfkill: input handler disabled
[   95.983171] systemd-journald[218]: File /var/log/journal/bf10ccd265b249d993c4
492849bb7340/user-1000.journal corrupted or uncleanly shut down, renaming and re
placing.
[   96.462460] rfkill: input handler enabled
[  102.828911] rfkill: input handler disabled
[  893.778985] hello: module verification failed: signature and/or required key
missing - tainting kernel
[  893.782289] Hello World!
[04/22/22]seed@VM:~/.../kernel_module$
```

```
seed@VM: ~/.../kernel_module
[   14.004101] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   16.258349] aufs 5.4.3-20200302
[   20.163391] kauditd_printk_skb: 28 callbacks suppressed
[   20.163393] audit: type=1400 audit(1650667326.623:40): apparmor="STATUS" oper
ation="profile_load" profile="unconfined" name="docker-default" pid=1167 comm="a
pparmor_parser"
[   21.437961] bridge: filtering via arp/ip/ip6tables is no longer available by
default. Update your scripts to load br_netfilter if you need this.
[   21.448739] Bridge firewalling registered
[   21.517947] Started bpfilter
[   21.518259] bpfilter: Loaded bpfilter_umh pid 1205
[   22.387398] Initializing XFRM netlink socket
[   24.458634] rfkill: input handler disabled
[   95.983171] systemd-journald[218]: File /var/log/journal/bf10ccd265b249d993c4
492849bb7340/user-1000.journal corrupted or uncleanly shut down, renaming and re
placing.
[   96.462460] rfkill: input handler enabled
[  102.828911] rfkill: input handler disabled
[  893.778985] hello: module verification failed: signature and/or required key
missing - tainting kernel
[  893.782289] Hello World!
[04/22/22]seed@VM:~/.../kernel_module$ sudo rmmod hello
[04/22/22]seed@VM:~/.../kernel_module$ lsmod | grep hello
[04/22/22]seed@VM:~/.../kernel_module$
```

To remove the module from the kernel, we implement the following command “sudo rmmod hello”

```
seed@VM: ~/.../kernel_module
 RX
[   14.004101] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   16.258349] aufs 5.4.3-20200302
[   20.163391] kauditd_printk_skb: 28 callbacks suppressed
[   20.163393] audit: type=1400 audit(1650667326.623:40): apparmor="STATUS" oper
ation="profile_load" profile="unconfined" name="docker-default" pid=1167 comm="a
pparmor_parser"
[   21.437961] bridge: filtering via arp/ip/ip6tables is no longer available by
default. Update your scripts to load br_netfilter if you need this.
[   21.448739] Bridge firewalling registered
[   21.517947] Started bpfilter
[   21.518259] bpfilter: Loaded bpfilter_umh pid 1205
[   22.387398] Initializing XFRM netlink socket
[   24.458634] rfkill: input handler disabled
[   95.983171] systemd-journald[218]: File /var/log/journal/bf10ccd265b249d993c4
492849bb7340/user-1000.journal corrupted or uncleanly shut down, renaming and re
placing.
[   96.462460] rfkill: input handler enabled
[  102.828911] rfkill: input handler disabled
[  893.778985] hello: module verification failed: signature and/or required key
missing - tainting kernel
[  893.782289] Hello World!
[ 1044.144768] Bye-bye World!.
[04/22/22]seed@VM:~/.../kernel_module$
```

On execution of sudo rmmod hello, we observe the message “***Bye-bye World!***” as expected.

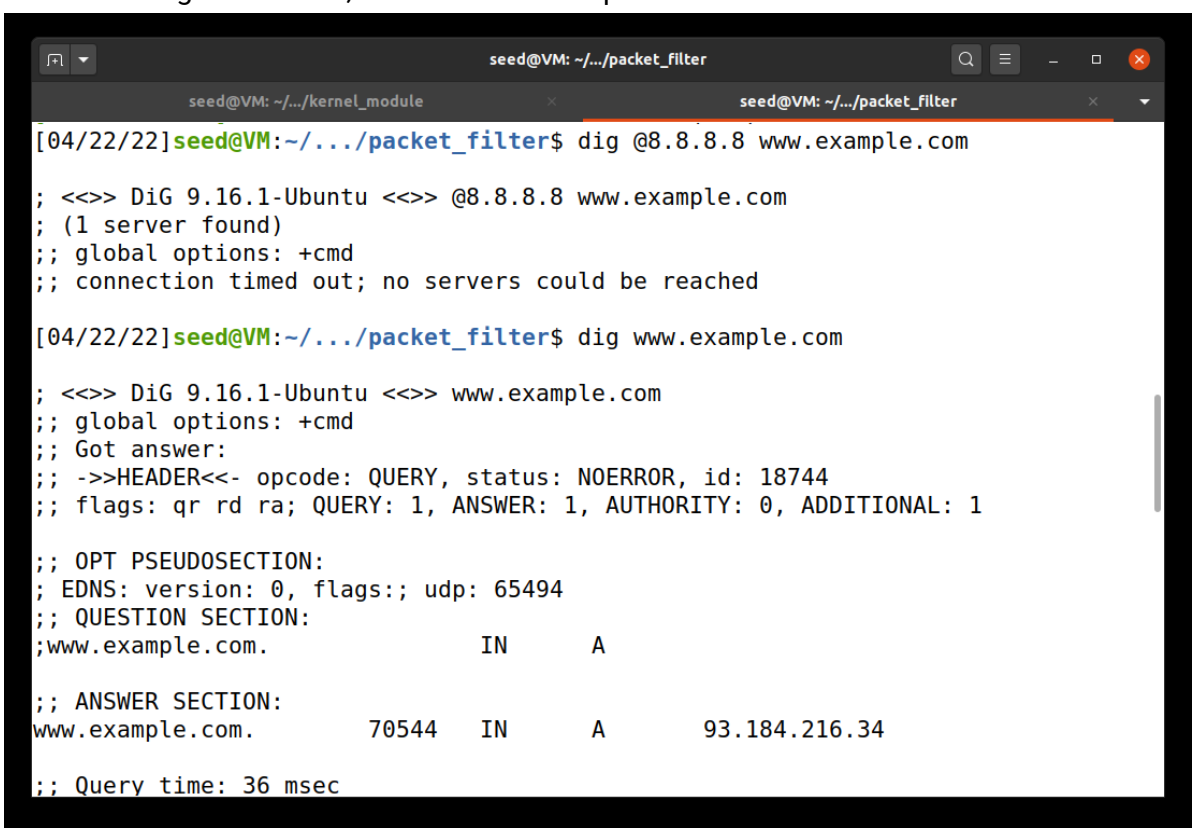
Task 1.B: Implement a Simple Firewall Using Netfilter: Netfilter is designed to facilitate the manipulation of packets by authorized users. It achieves this goal by implementing a number of hooks in the Linux kernel. These hooks are inserted into various places, including the packet incoming and outgoing paths. If we want to manipulate the incoming packets, we simply need to connect our own programs (within LKM) to the corresponding hooks. Once an incoming packet arrives, our program will be invoked. Our program can decide whether this packet should be blocked or not; moreover, we can also modify the packets in the program. In this task, you need to use LKM and Netfilter to implement a packet filtering module. This module will fetch the firewall policies from a data structure, and use the policies to decide whether packets should be blocked or not. We would like students to focus on the filtering part, the core of firewalls, so students are allowed to hardcode firewall policies in the program.

We need to enter into the seed VM and compile the seedFilter kernel module using the make command. Once we do that, the seedFilter.ko file is generated.

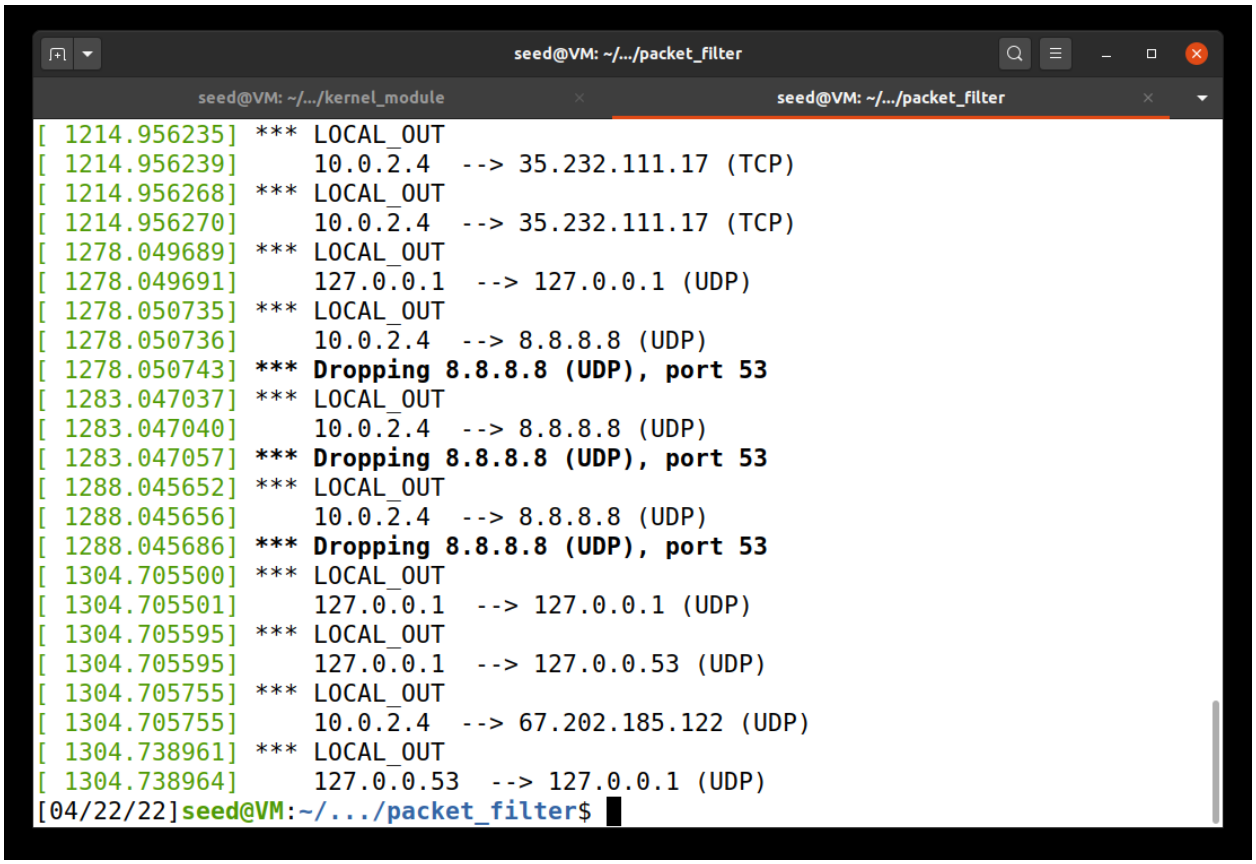
Now, we need to test whether our kernel module is blocking '8.8.8.8'. We can check that by running "dig @8.8.8.8 www.example.com" command.

In the below figure we can see that on running the dig command, no output was generated.

After removing the module, we can see the output for the command



Here, We can see the dropped packets which were blocked by the firewall.



To execute the Subtask 2; we need to open the SeedFilter.c file and make the following changes to the C program, we need to comment out Hook 2 and declare:

- Hook 3(PRE_ROUTING)
- Hook 4(LOCAL_IN)
- Hook 5(INET_FORWARD)
- Hook 6(POST_ROUTING)

I have attached screenshots of the changes that were made to the SeedFilter.c file below.

```
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_PRE_ROUTING;
   hook3.pf = PF_INET;
   hook3.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook3);

   hook4.hook = printInfo;
   hook4.hooknum = NF_INET_LOCAL_IN;
   hook4.pf = PF_INET;
   hook4.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook4);

   hook5.hook = printInfo;
   hook5.hooknum = NF_INET_FORWARD;
   hook5.pf = PF_INET;
                                                                    98,36          72%
```

```
   hook4.pf = PF_INET;
   hook4.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook4);

   hook5.hook = printInfo;
   hook5.hooknum = NF_INET_FORWARD;
   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, &hook1);

   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);
   nf_unregister_net_hook(&init_net, &hook5);
   nf_unregister_net_hook(&init_net, &hook6);

}

module_init(registerFilter);
                                                                    114,1          96%
```

Now we need to compile the new code and install it as we did before.

One important thing to note is that we need to remove the old module that is still in the kernel otherwise we will encounter an error.

The hooks that are registered with netfilter are as follows (with the functions in each hook in the order that they are actually called):

```
--->PRE------>[ROUTE]--->FWD---------->POST------>
    Conntrack    |       Mangle   ^    Mangle
    Mangle       |       Filter   |    NAT (Src)
    NAT (Dst)    |                |    Conntrack
    (QDisc)      |             [ROUTE]
                 v                |
                 IN Filter       OUT Conntrack
                 |  Conntrack     ^  Mangle
                 |  Mangle        |  NAT (Dst)
                 v                |  Filter
```

```
seed@VM: ~/.....   seed@VM: ~/.....   seed@VM: ~/.....   seed@VM: ~/.....   seed@VM: ~
[ 3855.434682] *** LOCAL_OUT
[ 3855.434683]     10.20.30.13  --> 192.168.129.129 (UDP)
[ 3855.434687] *** POST_ROUTING
[ 3855.434688]     10.20.30.13  --> 192.168.129.129 (UDP)
[ 3855.435733] *** PRE_ROUTING
[ 3855.435734]     192.168.129.129  --> 10.20.30.13 (UDP)
[ 3855.435739] *** LOCAL_IN
[ 3855.435740]     192.168.129.129  --> 10.20.30.13 (UDP)
[ 3855.435806] *** LOCAL_OUT
[ 3855.435807]     127.0.0.53  --> 127.0.0.1 (UDP)
[ 3855.435809] *** POST_ROUTING
[ 3855.435809]     127.0.0.53  --> 127.0.0.1 (UDP)
[ 3855.435813] *** PRE_ROUTING
[ 3855.435813]     127.0.0.53  --> 127.0.0.1 (UDP)
[ 3855.435814] *** LOCAL_IN
[ 3855.435815]     127.0.0.53  --> 127.0.0.1 (UDP)
```

These extra filters will block the PING or Telnet signals, so if we try to implement a PING/Telnet we will observe dropped packets as seen above.

If we connect to the router and implement a PING command we can see the forwarded packets as seen in the screenshot below.

References:

[1] https://seedsecuritylabs.org/Labs_20.04/Files/Firewall/Firewall.pdf