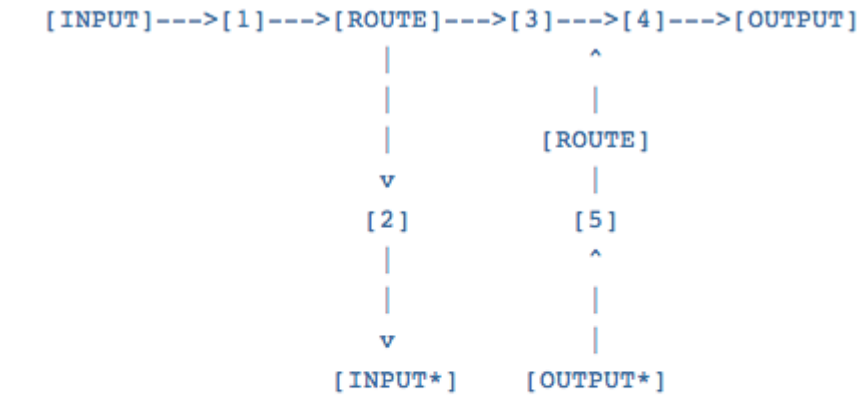
Networking

Objective: In this lab you will develop an LKM called "netdead" that will kill all incoming and outgoing network traffic in the kernel.

File(s) for this lab:

Netfilter is a packet filtering subsystem in the Linux kernel stack and has been there since kernel 2.4.x. Netfilter's core consists of five hook functions declared in linux/netfilter_ipv4.h. Although these functions are for IPv4, they aren't much different from those used in the IPv6 counterpart. The hooks are used to analyze packets in various locations on the network stack. This situation is depicted below:



[1] NF_IP_PRE_ROUTING
[2] NF_IP_LOCAL_IN
[3] NF_IP_FORWARD
[4] NF_IP_POST_ROUTING
[5] NF_IP_LOCAL_OUT
[*] Network Stack

NF_IP_PRE_ROUTING is called right after the packet has been received. This is the hook we are most interested in for our micro-firewall. NF_IP_LOCAL_IN is used for packets that are destined for the network stack and thus has not

been forwarded. NF_IP_FORWARD is for packets not addressed to us but that should be forwarded. NF_IP_POST_ROUTING is for packets that have been routed and are ready to leave, and NF_IP_LOCAL_OUT is for packets sent out from our own network stack. Each function has a chance to mangle or do what it wishes with the packets, but it eventually has to return a Netfilter code. Here are the codes that can be returned and what they mean:

NF_ACCEPT: accept the packet (continue network stack trip)

NF_DROP: drop the packet (don't continue trip)

NF_REPEAT: repeat the hook function

NF_STOLEN: hook steals the packet (don't continue trip)

NF_QUEUE: queue the packet to userspace

1. Complete the code below for “netdead” to kill network traffic on the host:

```
[user@rhki Lab19]$ sudo /sbin/insmod netdead.ko
[sudo] password for user:

64 bytes from ir2.fp.vip.bf1.yahoo.com (98.139.183.24): icmp_seq=30 ttl=128 time=131 ms
64 bytes from ir2.fp.vip.bf1.yahoo.com (98.139.183.24): icmp_seq=31 ttl=128 time=62.0 ms
64 bytes from ir2.fp.vip.bf1.yahoo.com (98.139.183.24): icmp_seq=32 ttl=128 time=55.1 ms
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
```

Hints:

- Examine netfilter.h for info on the struct nf_hook_ops

```
struct nf_hook_ops
{
        struct list_head list;

        /* User fills in from here down. */
        nf_hookfn *hook;
        struct module *owner;
        int pf;
        int hooknum;
        /* Hooks are ordered in ascending priority. */
        int priority;
};
```

- See the code comments on the following page for more help.

```c
MODULE_LICENSE("GPL");           // Get rid of taint message by declaring code as GPL.

/*  Or with defines, like this: */
MODULE_AUTHOR(DRIVER_AUTHOR);    // Who wrote this module?
MODULE_DESCRIPTION(DRIVER_DESC); // What does this module do?

/* IP Hooks */
/* After promisc drops, checksum checks. */
#define NF_IP_PRE_ROUTING   0
/* If the packet is destined for this box. */
#define NF_IP_LOCAL_IN      1
/* If the packet is destined for another interface. */
#define NF_IP_FORWARD       2
/* Packets coming from a local process. */
#define NF_IP_LOCAL_OUT     3
/* Packets about to hit the wire. */
#define NF_IP_POST_ROUTING  4
#define NF_IP_NUMHOOKS      5

static struct nf_hook_ops netfilter_ops_in; /* NF_IP_PRE_ROUTING */
static struct nf_hook_ops netfilter_ops_out; /* NF_IP_POST_ROUTING */

/* Function prototype in <linux/netfilter> */
unsigned int main_hook(unsigned int hooknum,
                       struct sk_buff **skb,
                       const struct net_device *in,
                       const struct net_device *out,
                       int (*okfn)(struct sk_buff*))
{

    /* Drop ALL Packets */
}

int init(void)
{
    printk(KERN_INFO "init_module() called\n");


    /* build netfilter_ops_in struct */
    /* build netfilter_ops_out struct */

    /* register hooks */

    return 0;
}

void cleanup(void)
{

    /* unregister hooks */

    printk(KERN_ALERT "Unloading netdead ...\n");
}

module_init(init);
module_exit(cleanup);
```