Lkm.c

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/netdevice.h>

#include <linux/netfilter.h>

#include <linux/netfilter\_ipv4.h>

#include <linux/udp.h>

#include <linux/ip.h>

#include <linux/tcp.h>

#include <linux/skbuff.h>

#include <linux/inet.h>

#include <linux/init.h>

#include <linux/types.h>

static struct nf\_hook\_ops nfho\_in;

static struct nf\_hook\_ops nfho\_out;

unsigned int hook\_func(void \*priv, struct sk\_buff \*skb, const struct nf\_hook\_state \*state)

{

struct iphdr \*ip;

struct tcphdr \*tcp;

\_\_u32 sou\_ip;

\_\_u32 des\_ip;

\_\_u16 sou\_port;

\_\_u16 des\_port;

ip = (struct iphdr\*) skb\_network\_header(skb);

sou\_ip = ip->saddr;

des\_ip = ip->daddr;

tcp= (struct tcphdr\*)((\_\_u32 \*)ip + ip->ihl);

sou\_port = tcp->source;

des\_port = tcp->dest;

if(sou\_ip == in\_aton("10.0.2.10") && des\_ip == in\_aton("10.0.2.7") && ntohs(des\_port) == 23){

printk(KERN\_INFO "blocking telnet:VM1 to VM2.\n");

return NF\_DROP;

}

if(sou\_ip == in\_aton("10.0.2.7") && des\_ip == in\_aton("10.0.2.10") && ntohs(des\_port) == 23){

printk(KERN\_INFO "blocking telnet:VM2 to VM1.\n");

return NF\_DROP;

}

if(sou\_ip == in\_aton("10.0.2.10") && ntohs(des\_port) == 80){

printk(KERN\_INFO "blocking external website access\n");

return NF\_DROP;

}

if(sou\_ip == in\_aton("10.0.2.10") && des\_ip == in\_aton("10.0.2.7") && ntohs(des\_port) == 22){

printk(KERN\_INFO "blocking ssh: VM1 to VM2.\n");

return NF\_DROP;

}

if(sou\_ip == in\_aton("10.0.2.7") && des\_ip == in\_aton("10.0.2.10") && ntohs(des\_port) == 22){

printk(KERN\_INFO "blocking ssh:VM2 to VM1.\n");

return NF\_DROP;

}

printk(KERN\_INFO "allow packet.\n");

return NF\_ACCEPT;

}

int init\_module()

{

printk(KERN\_INFO "hello netfiler!\n");

nfho\_in.hook = hook\_func;

nfho\_in.hooknum = NF\_INET\_PRE\_ROUTING;

nfho\_in.pf = PF\_INET;

nfho\_in.priority = NF\_IP\_PRI\_FIRST;

nf\_register\_hook(&nfho\_in);

nfho\_out.hook = hook\_func;

nfho\_out.hooknum = NF\_INET\_POST\_ROUTING;

nfho\_out.pf = PF\_INET;

nfho\_out.priority = NF\_IP\_PRI\_FIRST;

nf\_register\_hook(&nfho\_out);

return 0;

}

void cleanup\_module()

{

printk(KERN\_INFO "bye netfiler!\n");

nf\_unregister\_hook(&nfho\_in);

nf\_unregister\_hook(&nfho\_out);

}

Make:

obj-m += lkm.o

all:

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

clean:

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