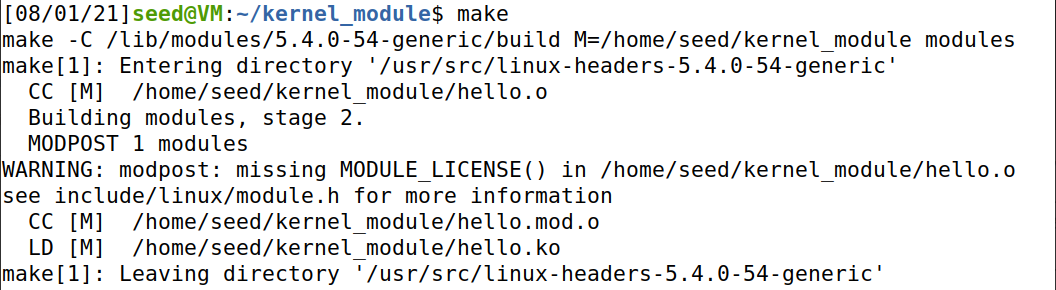
**第六次实验**

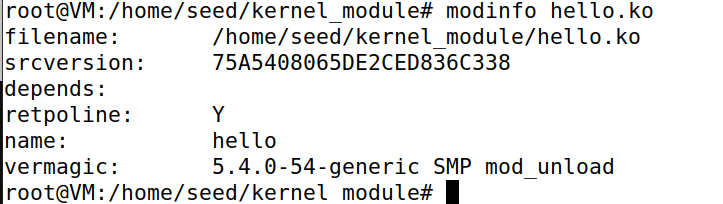
Task1

1. A

进行创建



模块信息



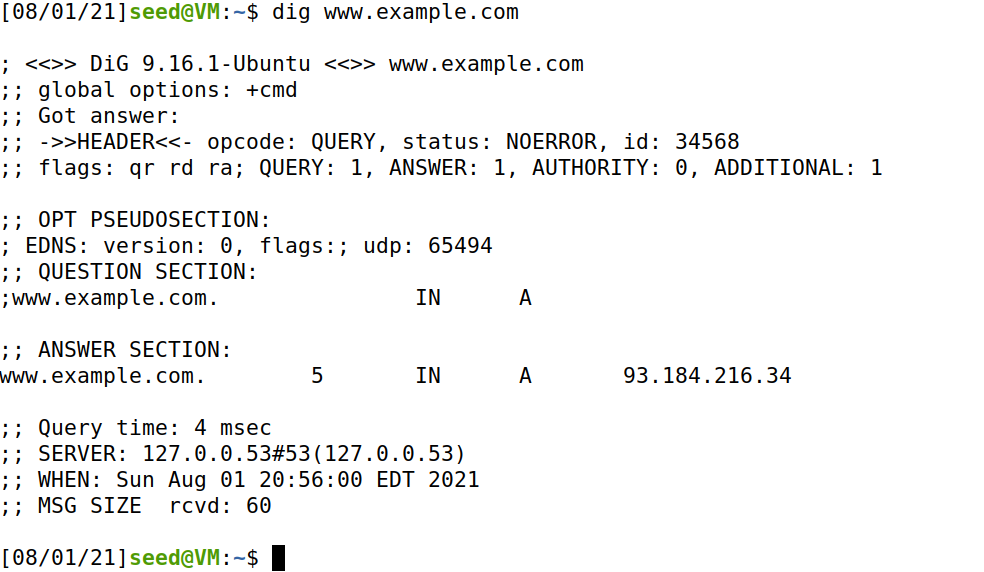
删除后输出Bye-bye World!



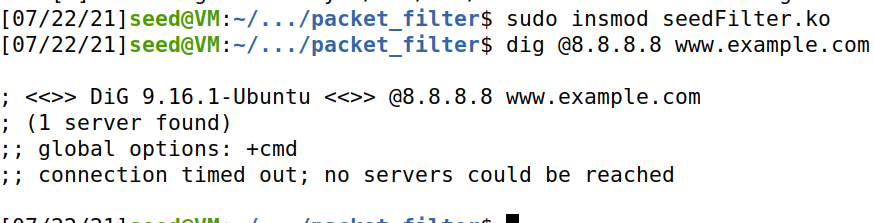
1. B

1

未开启防御前，正常连接



开启防火墙后，连接失败，无法连接到DNS服务器



2

将函数挂到五个钩子上

#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/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);

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;

}

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;

}

int registerFilter(void) {

printk(KERN\_INFO "Registering filters.\n");

// Hook 1

hook1.hook = printInfo;

hook1.hooknum = NF\_INET\_LOCAL\_IN;

hook1.pf = PF\_INET;

hook1.priority = NF\_IP\_PRI\_FIRST;

nf\_register\_net\_hook(&init\_net, &hook1);

// Hook 2

hook2.hook = printInfo;

hook2.hooknum = NF\_INET\_PRE\_ROUTING;

hook2.pf = PF\_INET;

hook2.priority = NF\_IP\_PRI\_FIRST;

nf\_register\_net\_hook(&init\_net, &hook2);

// Hook 3

hook3.hook = printInfo;

hook3.hooknum = NF\_INET\_FORWARD;

hook3.pf = PF\_INET;

hook3.priority = NF\_IP\_PRI\_FIRST;

nf\_register\_net\_hook(&init\_net, &hook3);

// Hook 4

hook4.hook = printInfo;

hook4.hooknum = NF\_INET\_LOCAL\_OUT;

hook4.pf = PF\_INET;

hook4.priority = NF\_IP\_PRI\_FIRST;

nf\_register\_net\_hook(&init\_net, &hook4);

// Hook 5

hook5.hook = printInfo;

hook5.hooknum = NF\_INET\_POST\_ROUTING;

hook5.pf = PF\_INET;

hook5.priority = NF\_IP\_PRI\_FIRST;

nf\_register\_net\_hook(&init\_net, &hook5);

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);

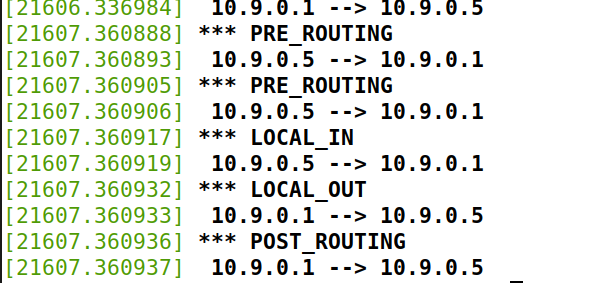
}

module\_init(registerFilter);

module\_exit(removeFilter);

MODULE\_LICENSE("GPL");

Ping10.9.0.1后查询信息，得到函数的输出



3

#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>

static struct nf\_hook\_ops telnetFilterHook;

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

struct iphdr \*iph;

struct tcphdr \*tcph;

iph = ip\_hdr(skb);

tcph = (void \*)iph+iph->ihl\*4;

if((iph->protocol == IPPROTO\_TCP && (tcph->dest == htons(23)

|| tcph->dest== htons(22)

|| tcph->dest== htons(21)))

|| (iph->protocol == IPPROTO\_ICMP &&((((unsigned char \*)&iph->daddr)[0]==10 &&

((unsigned char \*)&iph->daddr)[1]==9

&& ((unsigned char \*)&iph->daddr)[2]==0 && ((unsigned char \*)&iph->daddr)[3]==1)

|| (((unsigned char \*)&iph->daddr)[0]==10 && ((unsigned char \*)&iph->daddr)[1]==9

&& ((unsigned char \*)&iph->daddr)[2]==0 && ((unsigned char \*)&iph->daddr)[3]==1)))){

printk(KERN\_INFO "Dropping telent packdt to %d.%d.%d.%d\n",

((unsigned char \*)&iph->daddr)[0],

((unsigned char \*)&iph->daddr)[1],

((unsigned char \*)&iph->daddr)[2],

((unsigned char \*)&iph->daddr)[3]);

return NF\_DROP;

}else{

return NF\_ACCEPT;

}

}

void removeFilter(void){

printk(KERN\_INFO "Telnet filter has been removed.\n");

nf\_unregister\_net\_hook(&init\_net,&telnetFilterHook);

}

int setUpFilter(void){

telnetFilterHook.hook = telnetFilter;

telnetFilterHook.hooknum = NF\_INET\_PRE\_ROUTING;

telnetFilterHook.pf = PF\_INET;

telnetFilterHook.priority = NF\_IP\_PRI\_FILTER;

if(nf\_register\_net\_hook(&init\_net,&telnetFilterHook)!=0){

printk(KERN\_WARNING "register Telnet filter hook error!\n");

goto err;

}

printk(KERN\_INFO "Registering a Telnet filter");

return 0;

err:

removeFilter();

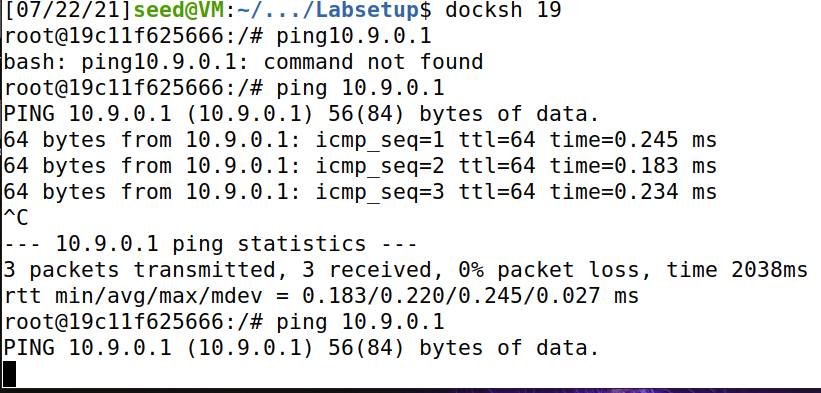
return -1;

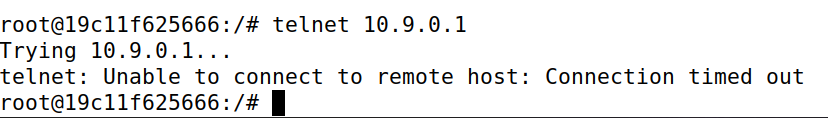
}

module\_init(setUpFilter);

module\_exit(removeFilter);

MODULE\_LICENSE("GPL");

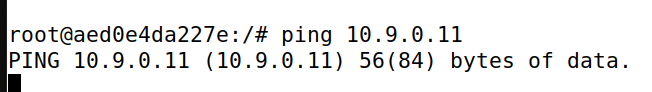




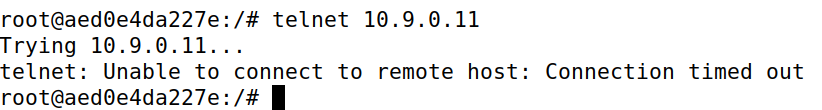
在10.9.0.5上ping和telnet10.9.0.1都不通，防火墙过滤成功

Task2

2.A



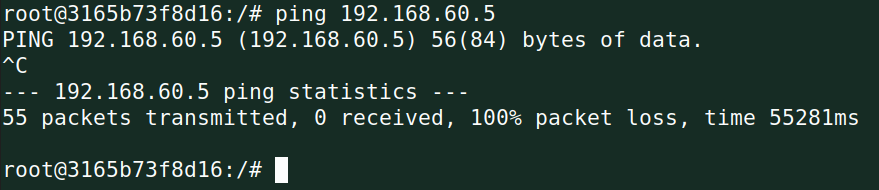
ping不通

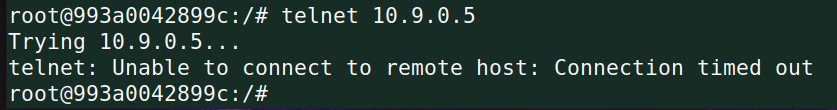
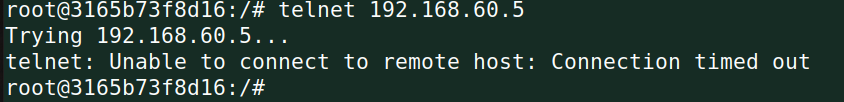
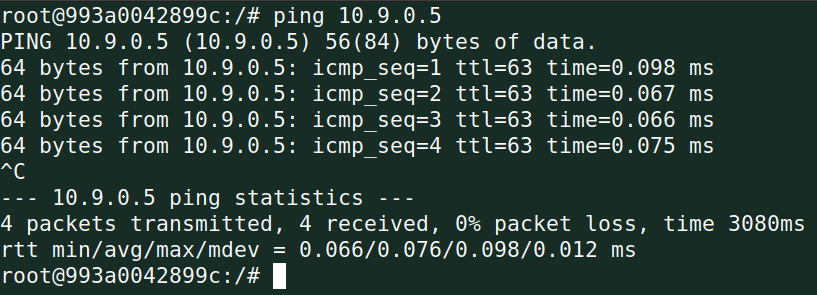
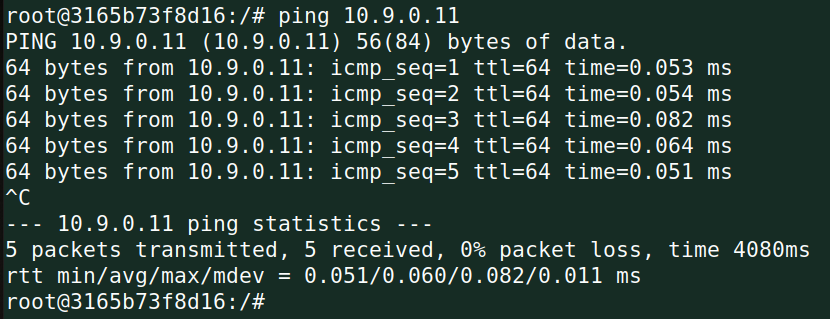


telnet不通

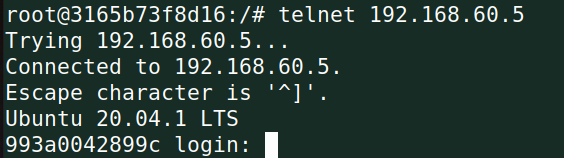
2.B

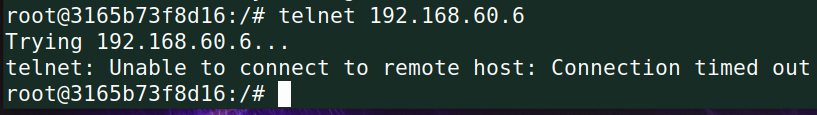
路由器中设置规则

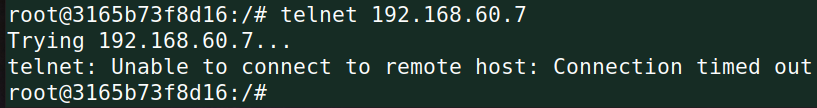


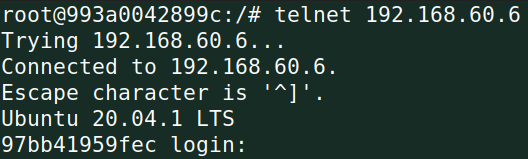


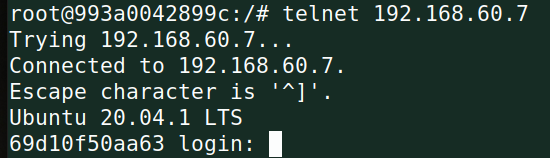
2.C

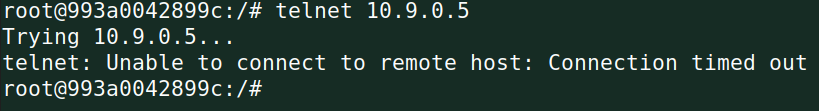


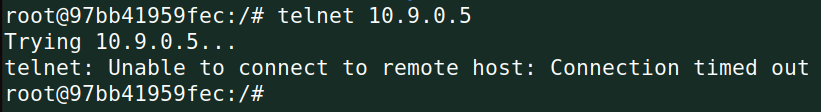


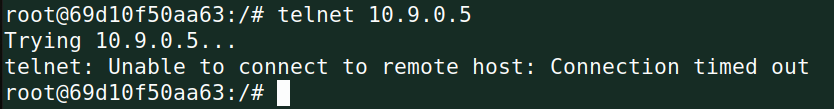








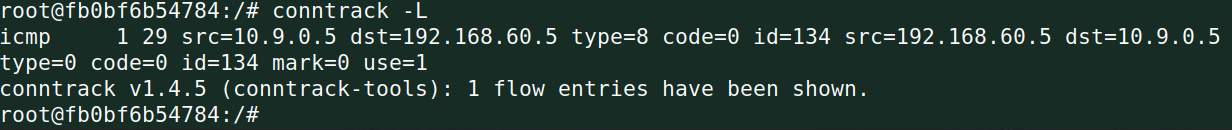




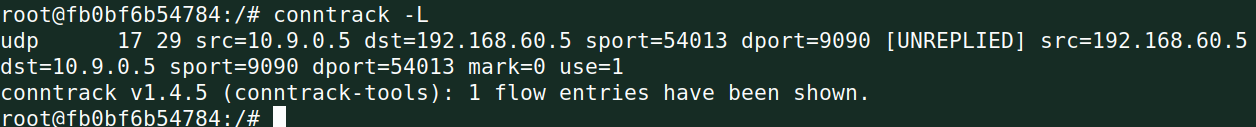
Task 3

3.A

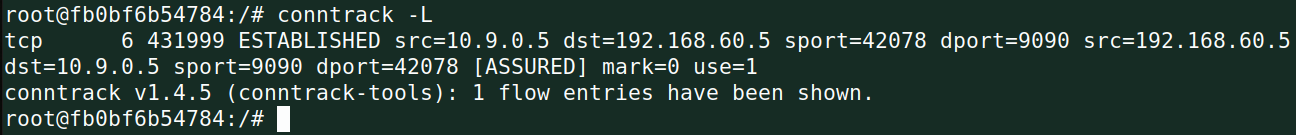
ICMP



UDP

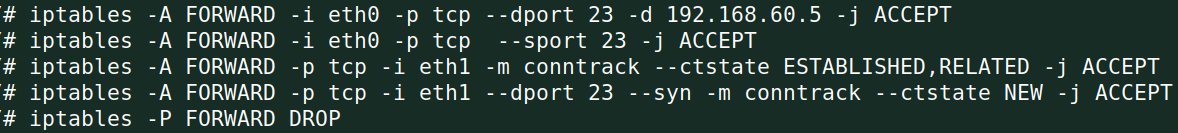


TCP



3.B:

使用跟踪连接修改规则如下：



Task 4

运行命令，在主机10.9.0.5里ping主机192.168.60.5，在连续接收到了多个回复后，过一段时间才能接收到下一个回复报文

