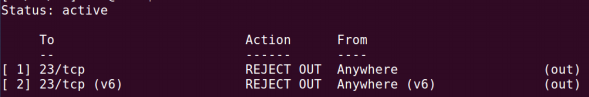
Lab6-report 57117219尚林灏

Task1:Using Firewall

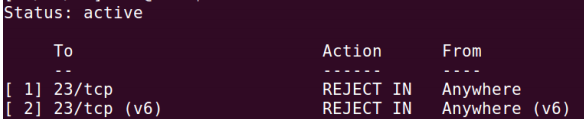
设置两台虚拟机A,B。防止A telnet到B；防止B向A telnet。

Ifconfig命令得出ip地址A：192.168.1.106 B：192.168.1.103

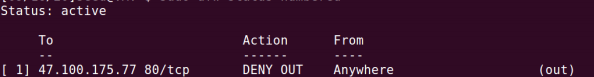
阻⽌A对B发起telnet



阻⽌B对A发起telnet



阻⽌A访问特定的外部⽹⻚



Task2：Implementing a Simple Firewall

代码：

#include <linux/ module.h>

#include <linux/ kernel.h>

#include <linux / skbuff.h>

#include <linux/ip.h>

#include <linux/netfilter.h>

#include <linux/netfilter\_ipv4.h>

static struct nf\_hook\_ops nfho;

/\*IP address we want to drop from\*/

static unsigned char \*drop\_ip = "\x0a\x00\x00\x04";

unsigned int hook\_func(unsigned int hooknum;

struct sk\_buff \*\*skb;

const struct net\_device \*in;

const struct net\_device \*out;

int (\*okfn) (struct sk\_buff \*))

{

struct sk\_buff \*sb = \*skb;

if(ip\_hdr(sb)->saddr == \*(unsigned int \*)drop\_ip)

{

return NF\_DROP;

}else{

return NF\_ACCEPT;

}

}

int init\_module()

{

nfho.hook = hook\_func;

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

nfho. pf = PF\_INET;

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

nf\_register\_hook(&nfho);

return 0 ;

}

#include <linux/ module.h>

#include <linux/ kernel.h>

#include <linux / skbuff.h>

#include <linux/ip.h>

#include <linux/netfilter.h>

#include <linux/netfilter\_ipv4.h>

static struct nf\_hook\_ops nfho;

/\*IP address we want to drop from\*/

static unsigned char \*drop\_ip = "\x0a\x00\x00\x04";

unsigned int hook\_func(unsigned int hooknum;

struct sk\_buff \*\*skb;

const struct net\_device \*in;

const struct net\_device \*out;

int (\*okfn) (struct sk\_buff \*))

{

struct sk\_buff \*sb = \*skb;

if(ip\_hdr(sb)->saddr == \*(unsigned int \*)drop\_ip)

{

return NF\_DROP;

}else{

return NF\_ACCEPT;

}

}

int init\_module()

{

nfho.hook = hook\_func;

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

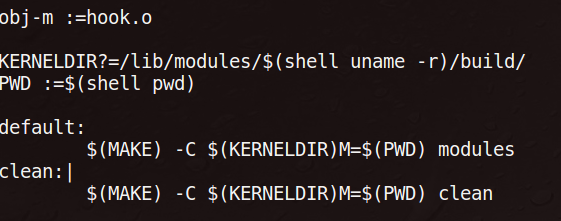
nfho. pf = PF\_INET;

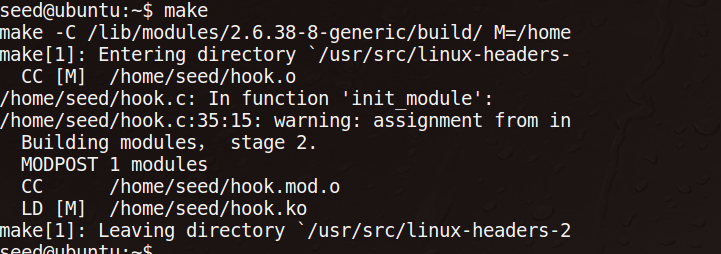
nfho.priority =NF\_IP\_PRI\_FIRST;

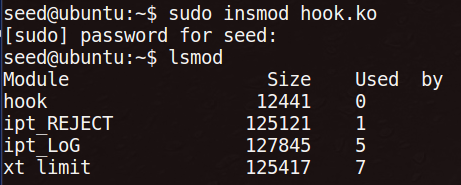
nf\_register\_hook(&nfho);

return 0 ;

}







PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.

From 10.0.0.4: icmp\_seq=2 Destination Host Unreachable

From 10.0.0.4: icmp\_seq=3 Destination Host Unreachable

From 10.0.0.4: icmp\_seq=4 Destination Host Unreachable

From 10.0.0.4: icmp\_seq=5 Destination Host Unreachable

From 10.0.0.4: icmp\_seq=6 Destination Host Unreachable

From 10.0.0.4: icmp\_seq=7 Destination Host Unreachable

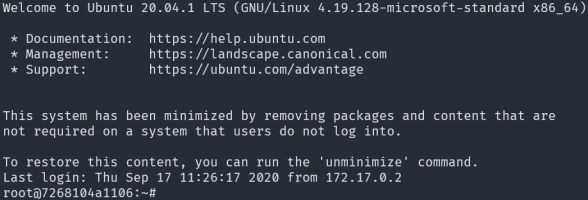
From 10.0.0.4: icmp\_seq=8 Destination Host Unreachable

From 10.0.0.4: icmp\_seq=10 Destination Host Unreachable

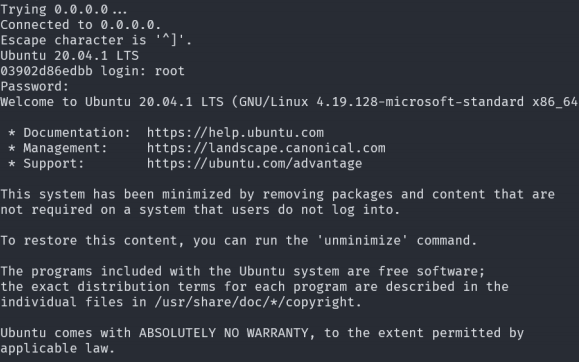
Task3：Evading Egress Filtering

设置三台虚拟机ABC。

在A中向B发起SSH请求，以B为跳板访问C

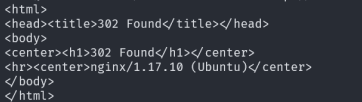


再在A中开⼀个shell，对⾃身的 8000 端⼝发起 telnet 连接：



在A中使⽤-C root@172.17.0.3

然后再在A的另⼀个Shell连接中，使⽤ 9000 端⼝作为Socksv5的代理。



A通过SSH向B容器传输请求，B访问 47.100.175.77 的80端⼝，并把数据返

回给A。将SSH关闭后，再次使⽤ curl 请求数据发⽣错误

Task4

在A中阻断外部向内部发起 80 端⼝和 22 端⼝的连接，⽤ nc 在80端⼝监听。在A中使⽤ssh -fCNR 172.17.0.3:2333:172.17.0.2:2334 [root@172.17.0.3](mailto:root@172.17.0.3)

对B的 2333 端⼝发起反向SSH隧道。在B中，向⾃身的 2333 端⼝发出TCP连接请求，在A中可观察到

