**2012年全国大学生电子设计竞赛**

--信息安全技术专题邀请赛设计报告附录

**网络隐形系统**

Cyber Stealth System



参赛学校: 西安电子科技大学

参赛队员: 朱利军 张子兼 高小青

指导老师: 张卫东

2012年8月

目录

[附录A: 虚拟化网络模块代码 3](#_Toc331968348)

[附录B: 反侦察中心代码 42](#_Toc331968349)

[附录C: 栈指纹混淆代码 50](#_Toc331968350)

[附录D: 应用程序主要代码 57](#_Toc331968351)

# 附录A: 虚拟化网络模块代码

function.c

#include <stdio.h>

#include <string.h>

#include <memory.h>

#include "main.h"

#include "function.h"

#include <libnet.h>

int initial()

{

int i;

bad\_n=0;

get\_interface();//网卡信息

mac\_ip\_diver();//伪造mac和ip

create\_thread();//创建线程，抓arp回复包

send\_local\_arp();//发送arp询问

wait\_thread();//等待线程结束

//printf("$$$$$$$the ip\_num is :%d\n",ip\_num);

//readfile();

//printf("here======================\n");

ip\_num=ip\_num+1;

get\_rand\_ip(ip\_num);////rand select the used ip 分配多网段地址

writefile\_host(ip\_for\_host,ip\_num);

writefile\_chg(ip\_for\_host);

cap\_packet();//抓包与发包

}

void writefile\_host(int ip\_chg,int ip\_sum)

{

FILE\*fp;

int i;

if((fp=fopen("host.txt","w"))==NULL)

{

printf("Can not open file!\n");

return 0;

}

for(i=ip\_chg+1;i<ip\_sum;i++)

{

char buf[16];

memcpy(buf,inet\_ntoa(ip\_host\_addr[i].src\_ip),16);

fputs(buf,fp);

fputs("\n",fp);

}

fclose(fp);

}

void writefile\_chg(int len)

{

FILE\*fp;

int i;

if((fp=fopen("chg.txt","w"))==NULL)

{

printf("Can not open file!\n");

return 0;

}

for(i=0;i<=len;i++)

{

char buf[16];

memcpy(buf,inet\_ntoa(ip\_host\_addr[i].src\_ip),16);

fputs(buf,fp);

fputs("\n",fp);

if(ip\_host\_addr[i].chg==0)

{

fputs("等待跳变...",fp);

fputs("\n",fp);

}

else if(ip\_host\_addr[i].chg==1)

{

fputs("正在使用中...",fp);

fputs("\n",fp);

}

//printf("the len is:%d---i is:%d\n",len,i);

}

fclose(fp);

}

void readfile()

{

FILE \*fp;

int i;

if((fp=fopen("test.txt","r"))==NULL)

{

printf("Can open file \n");

return 0;

}

for(i=0;i<255;i++)

{

fread(&ip\_mac\_addr[i],sizeof(struct ip\_mac),1,fp);

fseek(fp,i\*sizeof(struct ip\_mac),0);

/\*printf("the ip is:%s\n",inet\_ntoa(ip\_mac\_addr[i].src\_ip));

printf("the ip statue is:%d\n",ip\_mac\_addr[i].statue);

printf("the ip use\_num is:%d\n",ip\_mac\_addr[i].use\_num);\*/

}

close(fp);

}

void get\_use\_n(int buf[],int len)

{

int i;

srand((unsigned long)time(0));

for(i=0;i<len;i++)

{

//printf("the rand is:%d\n",rand()%10);

buf[i]=(rand()%10);

}

}

void get\_rand\_ip(int num)////select num ip number

{

int i,j,n,ip\_buf[255],k;

int ip\_n[num];

n=0;

unsigned long tmp1\_ip,tmp2\_ip;

memset(ip\_buf, 0, sizeof(ip\_buf));

ran(ip\_buf,255);

get\_use\_n(ip\_n,num);

ip\_host\_addr[0].src\_ip=local\_ip;

ip\_host\_addr[0].chg=1;

for(j=1;j<=num;j++)

{

while(1)

{

k=ip\_buf[n];

if(ip\_mac\_addr[k].statue==0)

{

tmp1\_ip=htonl(net\_ip)+k;

tmp2\_ip=ntohl(tmp1\_ip);////此处有问题

struct in\_addr ip\_temp=\*(struct in\_addr \*)&(tmp2\_ip);

ip\_host\_addr[j].src\_ip=ip\_temp;

int lastn=ip\_temp.s\_addr>>24;

memcpy(ip\_host\_addr[j].src\_mac,ip\_mac\_addr[lastn].src\_mac,6);

printf("the ip is:%s\n",inet\_ntoa(ip\_temp));

n++;

break;

}

else

{

n++;

continue;

}

}

ip\_host\_addr[j].use\_num=ip\_n[j];

}

}

void ran(int buffer[], int length)

{

int i,j,k;

int buf[length];

srand((unsigned long)time(0));

for(i = 0; i < length; i++)

{

int temp=rand()%254;

if(temp!=0)

buf[i] =temp;

}

for(j=0;j<length;j++)

{

int n=buf[j];

for(k=j+1;k<length;k++)

{

if(n==buf[k])

buf[k]=rand()%254;

else

continue;

}

buffer[j]=n;

}

return buffer;

}

void send\_local\_arp()

{

int i,num;

u\_char dest\_mac[6]={0xff,0xff,0xff,0xff,0xff,0xff};

num=local\_ip.s\_addr>>24;

for(i=1;i<255;i++)

send\_arp(ip\_mac\_addr[num].src\_mac,dest\_mac,ip\_mac\_addr[num].src\_ip,ip\_mac\_addr[i].src\_ip,1);

}

void send\_arp(u\_char\*src\_mac,u\_char\*dst\_mac,struct in\_addr src\_ip,struct in\_addr dst\_ip,int type)

{

int res;

libnet\_t \*l;/\*\*\*\*\*libnet handler\*/

libnet\_ptag\_t p\_tag;

char err\_buff[LIBNET\_ERRBUF\_SIZE];

/\*\*\*\*\*\*\*\*\*\*init libnet\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sleep(0.5);

l=libnet\_init(LIBNET\_LINK\_ADV,net\_dev,err\_buff);

if(l==NULL)

{

printf("libnet\_init err!\n");

fprintf(stderr,"%s",err\_buff);

exit(0);

}

/\*\*\*\*\*\*\*\*\*\*build arp packet\*\*\*\*\*\*\*\*\*\*\*\*/

p\_tag=libnet\_build\_arp(ARPHRD\_ETHER,ETHERTYPE\_IP,6,4,type,src\_mac,(u\_int8\_t\*) &src\_ip,dst\_mac,

(u\_int8\_t\*) &dst\_ip,

NULL,/\*payload\*/

0,/\*payload size\*/

l,/\*libnet handler\*/

0/\*'0' stands out building a new packet\*/

);

if(p\_tag==-1)

{

printf("libnet\_build\_arp err!\n");

exit(0);

}

/\*\*\*\*\*\*\*\*\*\*\*build ethernet packet header\*\*\*\*\*\*\*\*\*\*\*\*\*/

p\_tag=libnet\_build\_ethernet(dst\_mac,src\_mac,ETHERTYPE\_ARP,NULL,0,l,0);

if(p\_tag==-1)

{

printf("libnet\_build\_ethernet err!\n");

exit(0);

}

if((res=libnet\_write(l))==-1)

{

printf("libnet\_write err!\n");

exit(0);

}

/\*\*\*\*\*\*\*\*\*over and destroy\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

libnet\_destroy(l);

return 0;

}

void create\_thread()

{

int ret\_req\_arp,ret\_packet;

memset(&thread\_req\_arp,0,sizeof(thread\_req\_arp));

if((ret\_req\_arp=pthread\_create(&thread\_req\_arp,NULL,cap\_req\_arp,NULL))!=0)

printf("thread create failed!\n");

}

void wait\_thread()

{

if(thread\_req\_arp!=0){

pthread\_join(thread\_req\_arp,NULL);

printf("thread\_req\_arp over!\n");

}

}

void cap\_packet()

{

printf("this is capture net arp reply--------------\n");

while(1)

{

const u\_char \*packet;

struct pcap\_pkthdr header;

char filter\_packet[]="";

pcap\_t\* handle\_packet;

handle\_packet = pcap\_open\_live(net\_dev, BUFSIZ, 1, 0, err\_cnt);

struct bpf\_program bpf\_filter;///设置过滤条件\*

pcap\_compile(handle\_packet, &bpf\_filter, filter\_packet, 0, net\_ip);//编译过滤条件

pcap\_setfilter(handle\_packet, &bpf\_filter);//设置过滤器

if(pcap\_datalink(handle\_packet) != DLT\_EN10MB)///数据链路层

return 0;

pcap\_loop(handle\_packet,-1,callback\_packet,0);//循环抓包

return 0;

}

}

void callback\_packet(u\_char \*argument, const struct pcap\_pkthdr\* header, const u\_char\* packet)

{

struct ethhdr \*Ether;

Ether=(struct ethhdr\*)(packet+0);

unsigned short protype;

protype=ntohs(Ether->h\_proto);

switch(protype)

{

case ETH\_P\_ARP:

//printf("this arp\n");

arp\_packet(packet);//回复虚假主机的mac地址

break;

case ETH\_P\_IP:

//printf("this ip\n");

ip\_packet(packet);//回复虚假主机的扫描数据包

break;

default:

break;

}

}

void change\_ip()

{

time\_t now;

struct tm \*timenow;

time(&now);

sleep(1);

timenow=localtime(&now);

//printf("----------the last time is:%d\n",min);

//printf("the time is :%s\n",asctime(timenow));

if(min!=timenow->tm\_min){

int temp=timenow->tm\_min-min;

if(temp<0){

int m=timenow->tm\_min+60-min;

if(m%time\_change==0)

{

chg\_ipaddr();

min=timenow->tm\_min;

printf("the chg1 ip is:%d\n",ip\_chg\_n);

}

}

///为了避免时间溢出超过60而执行出错

else if((temp)%time\_change==0)

{

printf("this the best time to change ip!\n");

chg\_ipaddr();

min=timenow->tm\_min;

printf("the chg ip2 is:%d\n",ip\_chg\_n);

}

}

}

void chg\_ipaddr()

{

int n=ip\_for\_host;

char timebuf[100];

time\_t now;

int i;

struct tm \*timenow;

time(&now);

sleep(1);

timenow=localtime(&now);

printf("the time is :%s\n",asctime(timenow));

sprintf(timebuf,"%s",asctime(timenow));

printf("the ip\_chg\_n is:%d\n",ip\_chg\_n);

if(ip\_chg\_n<ip\_for\_host){

char strip[16];

sprintf(strip,"%s",inet\_ntoa(ip\_host\_addr[ip\_chg\_n+1].src\_ip));

char buf[100]="ifconfig ";

strcat(buf,net\_dev);

strcat(buf," ");

strcat(buf,strip);

strcat(buf," up" );

printf("the str is:%s\n",buf);

ip\_host\_addr[ip\_chg\_n].chg=0;

ip\_host\_addr[ip\_chg\_n+1].chg=1;

writefile\_chg(ip\_for\_host);

system(buf);

strcat(timebuf,"---------此时IP地址是:");

strcat(timebuf,strip);

fileadd(timebuf);

ip\_chg\_n++;

}

else if(ip\_chg\_n==ip\_for\_host)

{

char strip[16];

ip\_host\_addr[ip\_chg\_n].chg=0;

ip\_chg\_n=0;

ip\_host\_addr[ip\_chg\_n].chg=1;

sprintf(strip,"%s",inet\_ntoa(ip\_host\_addr[ip\_chg\_n].src\_ip));

printf("the chang ip is:%s\n",inet\_ntoa(ip\_host\_addr[ip\_chg\_n].src\_ip));

char buf[100]="ifconfig ";

strcat(buf,net\_dev);

strcat(buf," ");

strcat(buf,strip);

strcat(buf," up" );

printf("the str is:%s\n",buf);

writefile\_chg(ip\_for\_host);

system(buf);

strcat(timebuf,"---------此时IP地址是:");

strcat(timebuf,strip);

fileadd(timebuf);

}

}

void fileadd(char str[])

{

FILE \*fp;

char ch;

fp=fopen("chglog.txt","at+");

fputs(str,fp);

fputs("\n",fp);

rewind(fp);

printf("\n");

fclose(fp);

}

void num(int buf[],int len)

{

int i;

srand((unsigned long)time(0));

for(i=0;i<len;i++)

buf[i]=rand()%len;

}

void send\_packet()

{

int i,j,k,buf\_n[ip\_num-ip\_for\_host];

get\_use\_n(buf\_n,ip\_num-ip\_for\_host);

/\*for(i=0;i<10;i++)

{

printf("the dst\_addr is:%s\n",inet\_ntoa(dst\_addr[i].src\_ip));

printf("the dst\_mac is:%02x-%02x-%02x-%02x-%02x-%02x\n",\*dst\_addr[i].src\_mac,

\*(dst\_addr[i].src\_mac+1),

\*(dst\_addr[i].src\_mac+2),

\*(dst\_addr[i].src\_mac+3),

\*(dst\_addr[i].src\_mac+4),

\*(dst\_addr[i].src\_mac+5));

}\*/

u\_int16\_t s\_port1,d\_port1;

u\_int32\_t seq1,seq\_r1,ack1;

u\_int16\_t s\_port2,d\_port2;

u\_int32\_t seq2,seq\_r2,ack2;

u\_int16\_t s\_port3,d\_port3;

u\_int32\_t seq3,seq\_r3,ack3;

int buf\_port[2];

int buf[10]={80,86,63,445,43,115,765,21,180,15};//1,9,86,63,445,43,115,765,180,15};

//printf("the ip\_for host is:%d\n",ip\_for\_host);

//printf("the ip\_num is:%d\n",ip\_num);

int temp=(ip\_num-ip\_for\_host)/3;

for(j=ip\_for\_host;j<ip\_for\_host+temp;j++){

int i;

get\_use\_n(buf\_port,2);

int n1=buf\_port[1];

get\_use\_n(buf\_port,2);

int n2=buf\_port[1];

get\_use\_n(buf\_port,2);

int n3=buf\_port[1];

d\_port1=buf[n1];

d\_port2=buf[n2];

d\_port3=buf[n3];

int num1=buf\_n[j-ip\_for\_host];

int num2=buf\_n[j-ip\_for\_host+temp];

int num3=buf\_n[j-ip\_for\_host+temp\*2];

s\_port1 = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

s\_port2 = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

s\_port3 = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

p\_send\_udp(ip\_host\_addr[j+temp].src\_mac,ip\_host\_addr[j+temp].src\_ip,dst\_addr[num2].src\_mac,dst\_addr[num2].src\_ip);

///三次握手主动请求

p\_send\_syn(ip\_host\_addr[j+temp\*2].src\_mac,ip\_host\_addr[j+temp\*2].src\_ip,dst\_addr[num3].src\_mac,dst\_addr[num3].src\_ip,0,0,s\_port3,d\_port3);///0为seq 0为ack

///回复请求信息

p\_send\_syn\_ack(dst\_addr[num1].src\_mac,dst\_addr[num1].src\_ip,ip\_host\_addr[j].src\_mac,ip\_host\_addr[j].src\_ip,0,1,s\_port1,d\_port1);

p\_send\_syn(ip\_host\_addr[j].src\_mac,ip\_host\_addr[j].src\_ip,dst\_addr[num1].src\_mac,dst\_addr[num2].src\_ip,0,0,s\_port1,d\_port1);

p\_send\_ack(ip\_host\_addr[j].src\_mac,ip\_host\_addr[j].src\_ip,dst\_addr[num1].src\_mac,dst\_addr[num1].src\_ip,1,1,s\_port1,d\_port1);

p\_send\_syn(ip\_host\_addr[j+temp].src\_mac,ip\_host\_addr[j+temp].src\_ip,dst\_addr[num2].src\_mac,dst\_addr[num2].src\_ip,0,0,s\_port2,d\_port2);

p\_send\_syn\_ack(dst\_addr[num3].src\_mac,dst\_addr[num3].src\_ip,ip\_host\_addr[j+temp\*2].src\_mac,ip\_host\_addr[j+temp\*2].src\_ip,0,1,s\_port3,d\_port3);

p\_send\_ack(ip\_host\_addr[j+temp\*2].src\_mac,ip\_host\_addr[j+temp\*2].src\_ip,dst\_addr[num3].src\_mac,dst\_addr[num3].src\_ip,1,1,s\_port3,d\_port3);

p\_send\_udp(ip\_host\_addr[j+temp\*2].src\_mac,ip\_host\_addr[j+temp\*2].src\_ip,dst\_addr[num3].src\_mac,dst\_addr[num3].src\_ip);

p\_send\_syn\_ack(dst\_addr[num2].src\_mac,dst\_addr[num2].src\_ip,ip\_host\_addr[j+temp].src\_mac,ip\_host\_addr[j+temp].src\_ip,0,1,s\_port2,d\_port2);

///确认收建立连接

p\_send\_ack(ip\_host\_addr[j+temp].src\_mac,ip\_host\_addr[j+temp].src\_ip,dst\_addr[num2].src\_mac,dst\_addr[num2].src\_ip,1,1,s\_port2,d\_port2);

///以上是建立三次握手的过程

///以下是UDP数据包的构造

p\_send\_udp(ip\_host\_addr[j].src\_mac,ip\_host\_addr[j].src\_ip,dst\_addr[num1].src\_mac,dst\_addr[num1].src\_ip);

/////////////////////////////////

//printf("the ip\_use\_num is:%d\n",ip\_host\_addr[j].use\_num);

for(k=1;k<ip\_host\_addr[j].use\_num;k++)

{

get\_use\_n(buf\_port,2);

int m1=buf\_port[1];

get\_use\_n(buf\_port,2);

int m2=buf\_port[1];

get\_use\_n(buf\_port,2);

int m3=buf\_port[1];

d\_port1=buf[m1];

d\_port2=buf[m2];

d\_port3=buf[m3];

s\_port1 = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

s\_port2 = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

s\_port3 = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

if(d\_port3==80|d\_port1==80|d\_port2==80)

{

p\_send\_http(dst\_addr[num3].src\_mac,dst\_addr[num3].src\_ip,ip\_host\_addr[j+temp\*2].src\_mac,ip\_host\_addr[j+temp\*2].src\_ip,1+(k-1)\*1460,1+(k-1)\*1460,80,s\_port3);

p\_send\_http(dst\_addr[num1].src\_mac,dst\_addr[num1].src\_ip,ip\_host\_addr[j].src\_mac,ip\_host\_addr[j].src\_ip,1+(k-1)\*1460,1+(k-1)\*1460,80,s\_port1);

p\_send\_http(dst\_addr[num2].src\_mac,dst\_addr[num2].src\_ip,ip\_host\_addr[j+temp].src\_mac,ip\_host\_addr[j+temp].src\_ip,1+(k-1)\*1460,1+(k-1)\*1460,80,s\_port2);

///确认收到数据

}

else

{

p\_send\_tcp(dst\_addr[num2].src\_mac,dst\_addr[num2].src\_ip,ip\_host\_addr[j].src\_mac,ip\_host\_addr[j+temp].src\_ip,1+(k-1)\*1460,1+(k-1)\*1460,d\_port2,s\_port2);

p\_send\_tcp(dst\_addr[num3].src\_mac,dst\_addr[num3].src\_ip,ip\_host\_addr[j+temp\*2].src\_mac,ip\_host\_addr[j+temp\*2].src\_ip,1+(k-1)\*1460,1+(k-1)\*1460,d\_port3,s\_port3);

p\_send\_tcp(dst\_addr[num1].src\_mac,dst\_addr[num1].src\_ip,ip\_host\_addr[j].src\_mac,ip\_host\_addr[j].src\_ip,1+(k-1)\*1460,1+(k-1)\*1460,d\_port1,s\_port1);

////////其他类型的数据包回复确认

}

}

///由本地主机回复确认

for(i=0;i<ip\_host\_addr[j].use\_num/3;i++)

{

p\_send\_ack(ip\_host\_addr[j].src\_mac,ip\_host\_addr[j].src\_ip,dst\_addr[num1].src\_mac,dst\_addr[num1].src\_ip,1+(k-1)\*1460,1+k\*1460,s\_port1,d\_port1);

p\_send\_ack(ip\_host\_addr[j+temp\*2].src\_mac,ip\_host\_addr[j+temp\*2].src\_ip,dst\_addr[num3].src\_mac,dst\_addr[num3].src\_ip,1+(k-1)\*1460,1+k\*1460,s\_port3,d\_port3);

p\_send\_ack(ip\_host\_addr[j+temp].src\_mac,ip\_host\_addr[j+temp].src\_ip,dst\_addr[num2].src\_mac,dst\_addr[num2].src\_ip,1+(k-1)\*1460,1+k\*1460,s\_port2,d\_port2);

}

///关闭连接

p\_send\_fin\_ack(ip\_host\_addr[j+temp\*2].src\_mac,ip\_host\_addr[j+temp\*2].src\_ip,dst\_addr[num3].src\_mac,dst\_addr[num3].src\_ip,1+k\*1460,2+(k-1)\*1460,s\_port3,d\_port3);

p\_send\_fin\_ack(ip\_host\_addr[j+temp].src\_mac,ip\_host\_addr[j+temp].src\_ip,dst\_addr[num2].src\_mac,dst\_addr[num2].src\_ip,1+k\*1460,2+(k-1)\*1460,s\_port2,d\_port2);

p\_send\_ack(dst\_addr[num3].src\_mac,dst\_addr[num3].src\_ip,ip\_host\_addr[j+temp\*2].src\_mac,ip\_host\_addr[j+temp\*2].src\_ip,2+(k-1)\*1460,2+k\*1460,d\_port3,s\_port3);

p\_send\_ack(dst\_addr[num2].src\_mac,dst\_addr[num2].src\_ip,ip\_host\_addr[j+temp].src\_mac,ip\_host\_addr[j+temp].src\_ip,2+(k-1)\*1460,2+k\*1460,d\_port2,s\_port2);

p\_send\_fin\_ack(dst\_addr[num3].src\_mac,dst\_addr[num3].src\_ip,ip\_host\_addr[j+temp\*2].src\_mac,ip\_host\_addr[j+temp\*2].src\_ip,2+k\*1460,3+(k-1)\*1460,d\_port3,s\_port3);

p\_send\_ack(ip\_host\_addr[j+temp\*2].src\_mac,ip\_host\_addr[j+temp\*2].src\_ip,dst\_addr[num3].src\_mac,dst\_addr[num3].src\_ip,3+(k-1)\*1460,3+k\*1460,s\_port3,d\_port3);

p\_send\_fin\_ack(ip\_host\_addr[j].src\_mac,ip\_host\_addr[j].src\_ip,dst\_addr[num1].src\_mac,dst\_addr[num1].src\_ip,1+k\*1460,2+(k-1)\*1460,s\_port1,d\_port1);

p\_send\_fin\_ack(dst\_addr[num2].src\_mac,dst\_addr[num2].src\_ip,ip\_host\_addr[j+temp].src\_mac,ip\_host\_addr[j+temp].src\_ip,2+k\*1460,3+(k-1)\*1460,d\_port2,s\_port2);

p\_send\_ack(dst\_addr[num1].src\_mac,dst\_addr[num1].src\_ip,ip\_host\_addr[j].src\_mac,ip\_host\_addr[j].src\_ip,2+(k-1)\*1460,2+k\*1460,d\_port1,s\_port1);

p\_send\_fin\_ack(dst\_addr[num1].src\_mac,dst\_addr[num1].src\_ip,ip\_host\_addr[j].src\_mac,ip\_host\_addr[j].src\_ip,2+k\*1460,3+(k-1)\*1460,d\_port1,s\_port1);

p\_send\_ack(ip\_host\_addr[j].src\_mac,ip\_host\_addr[j].src\_ip,dst\_addr[num1].src\_mac,dst\_addr[num1].src\_ip,3+(k-1)\*1460,3+k\*1460,s\_port1,d\_port1);

p\_send\_ack(ip\_host\_addr[j+temp].src\_mac,ip\_host\_addr[j+temp].src\_ip,dst\_addr[num1].src\_mac,dst\_addr[num1].src\_ip,3+(k-1)\*1460,3+k\*1460,s\_port2,d\_port2);

}

}

void p\_send\_syn(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip,u\_int32\_t seq,u\_int32\_t ack,u\_int16\_t s\_port,u\_int16\_t d\_port)

{

libnet\_t \*plibnet\_ack;

libnet\_ptag\_t ptag;

u\_int16\_t ip\_id;

char errbuf[100];

u\_char \*payload="";

ip\_id = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

plibnet\_ack=libnet\_init(LIBNET\_LINK,net\_dev,errbuf);

if(ptag==-1)

printf("the error\n");

libnet\_build\_tcp(d\_port,s\_port,seq,ack,

TH\_SYN,1460,0,10,LIBNET\_TCP\_H,NULL,0,plibnet\_ack,0);

libnet\_build\_tcp\_options(NULL,0,plibnet\_ack,0);

libnet\_build\_ipv4(LIBNET\_IPV4\_H+LIBNET\_TCP\_H,//sizeof(payload),

IPTOS\_LOWDELAY,ip\_id,0,189,IPPROTO\_TCP,0,s\_ip.s\_addr,d\_ip.s\_addr,NULL,0,plibnet\_ack,0);

ptag=libnet\_build\_ethernet(d\_mac,//dst

s\_mac,//src

ETHERTYPE\_IP,

NULL,//(u\_int8\_t\*)payload,

0,//sizeof(payload),

plibnet\_ack,

0);

libnet\_write(plibnet\_ack);

libnet\_destroy(plibnet\_ack);

}

void p\_send\_syn\_ack(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip,u\_int32\_t seq,u\_int32\_t ack,u\_int16\_t s\_port,u\_int16\_t d\_port)

{

libnet\_t \*plibnet\_ack;

libnet\_ptag\_t ptag;

u\_int16\_t ip\_id;

char errbuf[100];

u\_char payload[]={};///表示携带数据为，空注意“”是为1，字符指针是为4

//printf("the sizeof is:%d\n",sizeof(payload));

ip\_id = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

plibnet\_ack=libnet\_init(LIBNET\_LINK,net\_dev,errbuf);

if(ptag==-1)

printf("the error\n");

libnet\_build\_tcp(s\_port,

d\_port,

seq,ack,

TH\_SYN|TH\_ACK,

1460,

0,

10,

LIBNET\_TCP\_H+20,

payload,

sizeof(payload),

plibnet\_ack,

0);

libnet\_build\_tcp\_options(NULL,0,plibnet\_ack,0);

libnet\_build\_ipv4(LIBNET\_IPV4\_H+LIBNET\_TCP\_H+sizeof(payload),

IPTOS\_LOWDELAY,

ip\_id,

0,

199,

IPPROTO\_TCP,

0,

s\_ip.s\_addr,

d\_ip.s\_addr,

NULL,

0,

plibnet\_ack,

0);

ptag=libnet\_build\_ethernet(d\_mac,//dst

s\_mac,//src

ETHERTYPE\_IP,

NULL,//(u\_int8\_t\*)payload,

0,//sizeof(payload),

plibnet\_ack,

0);

libnet\_write(plibnet\_ack);

libnet\_destroy(plibnet\_ack);

}

void p\_send\_ack(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip,u\_int32\_t seq,u\_int32\_t ack,u\_int16\_t s\_port,u\_int16\_t d\_port)

{

libnet\_t \*plibnet\_ack;

libnet\_ptag\_t ptag;

u\_int16\_t ip\_id;

char errbuf[100];

char payload[]={};

ip\_id = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

plibnet\_ack=libnet\_init(LIBNET\_LINK,net\_dev,errbuf);

if(ptag==-1)

printf("the error\n");

libnet\_build\_tcp(s\_port,d\_port,seq,ack,

TH\_ACK,1460,0,10,LIBNET\_TCP\_H+20+sizeof(payload),NULL,0,plibnet\_ack,0);

libnet\_build\_tcp\_options(NULL,0,plibnet\_ack,0);

libnet\_build\_ipv4(LIBNET\_IPV4\_H+LIBNET\_TCP\_H+sizeof(payload),

IPTOS\_LOWDELAY,ip\_id,0,99,IPPROTO\_TCP,0,s\_ip.s\_addr,d\_ip.s\_addr,NULL,0,plibnet\_ack,0);

ptag=libnet\_build\_ethernet(d\_mac,//dst

s\_mac,//src

ETHERTYPE\_IP,

NULL,//(u\_int8\_t\*)payload,

0,//sizeof(payload),

plibnet\_ack,

0);

libnet\_write(plibnet\_ack);

libnet\_destroy(plibnet\_ack);

}

void p\_send\_udp(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip)

{

libnet\_t \*plibnet\_udp;

libnet\_ptag\_t ptag;

u\_int16\_t ip\_id;

u\_int16\_t u\_s\_port,u\_d\_port;

char errbuf[100];

u\_char payload[760];

rand\_mem(payload,760);

//printf("the s\_port is:%d\n",u\_s\_port);

//printf("the d\_port is:%d\n",u\_d\_port);

int len=sizeof(payload);

ip\_id = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

while(u\_s\_port==0||u\_d\_port==0)

{

srand((unsigned long)time(0));

u\_s\_port= (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

u\_d\_port= (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

//printf("the u\_s\_port is:%d\n",u\_s\_port);

//printf("the u\_d\_port is:%d\n",u\_d\_port);

}

plibnet\_udp=libnet\_init(LIBNET\_LINK,net\_dev,errbuf);

if(plibnet\_udp==NULL)

printf("hhhhhhhhhhhhhhhhhhhhhhhhhhhhhh\n");

ptag=libnet\_build\_udp(u\_s\_port,u\_d\_port,LIBNET\_UDP\_H+len,0,payload,len,plibnet\_udp,NULL);

if(ptag==-1)

printf("here is ===========================problem\_udp\n");

libnet\_build\_ipv4(LIBNET\_IPV4\_H+LIBNET\_UDP\_H+len,

IPTOS\_LOWDELAY,ip\_id,0,99,IPPROTO\_UDP,0,s\_ip.s\_addr,d\_ip.s\_addr,NULL,0,plibnet\_udp,0);

ptag=libnet\_build\_ethernet(d\_mac,//dst

s\_mac,//src

ETHERTYPE\_IP,

NULL,//(u\_int8\_t\*)payload,

0,//sizeof(payload),

plibnet\_udp,

0);

ptag=libnet\_write(plibnet\_udp);

if(ptag==-1)

printf("here is problem\n");

//printf("发送UDP Packet\n");

libnet\_destroy(plibnet\_udp);

}

void p\_send\_tcp(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip,u\_int32\_t seq,u\_int32\_t ack,u\_int16\_t s\_port,u\_int16\_t d\_port)

{

libnet\_t \*plibnet\_ack;

libnet\_ptag\_t ptag;

u\_int16\_t ip\_id;

char errbuf[100];

u\_char payload[1460];

rand\_mem(payload,1460);

int len=sizeof(payload);

ip\_id = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

plibnet\_ack=libnet\_init(LIBNET\_LINK,net\_dev,errbuf);

if(plibnet\_ack==NULL)

printf("hhhhhhhhhhhhhhhhhhhhhhhhhhhhhh\n");

//(80,10,0x10,0x01,TH\_SYN|TH\_ACK,1460,0,10,LIBNET\_TCP\_H+20,NULL,0,l,0

ptag=libnet\_build\_tcp(d\_port,s\_port,seq,ack,TH\_ACK,1460,0,10,LIBNET\_TCP\_H+len,payload,len,plibnet\_ack,0);

if(ptag==-1)

printf("here is ===========================problem\_tcp\n");

libnet\_build\_tcp\_options(NULL,0,plibnet\_ack,0);

libnet\_build\_ipv4(LIBNET\_IPV4\_H+LIBNET\_TCP\_H+len,

IPTOS\_LOWDELAY,ip\_id,0,99,IPPROTO\_TCP,0,s\_ip.s\_addr,d\_ip.s\_addr,NULL,0,plibnet\_ack,0);

ptag=libnet\_build\_ethernet(d\_mac,//dst

s\_mac,//src

ETHERTYPE\_IP,

NULL,//(u\_int8\_t\*)payload,

0,//sizeof(payload),

plibnet\_ack,

0);

ptag=libnet\_write(plibnet\_ack);

if(ptag==-1)

printf("here is problem\n");

libnet\_destroy(plibnet\_ack);

}

void p\_send\_http(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip,u\_int32\_t seq,u\_int32\_t ack,u\_int16\_t s\_port,u\_int16\_t d\_port)

{

libnet\_t \*plibnet\_ack;

libnet\_ptag\_t ptag;

u\_int16\_t ip\_id;

char errbuf[100];

u\_char payload[560]={

0x48,0x54,0x54,0x50,0x2f,0x31,0x2e,0x31,0x20,0x32,0x30,0x30,0x20,0x4f,0x4b,0x0d,

0x0a,0x44,0x61,0x74,0x65,0x3a,0x20,0x4d,0x6f,0x6e,0x2c,0x20,0x32,0x33,0x20,0x4a,

0x75,0x6c,0x20,0x32,0x30,0x31,0x32,0x20,0x30,0x38,0x3a,0x34,0x34,0x3a,0x31,0x36,

0x20,0x47,0x4d,0x54,0x0d,0x0a,0x53,0x65,0x72,0x76,0x65,0x72,0x3a,0x20,0x41,0x70,

0x61,0x63,0x68,0x65,0x2f,0x32,0x2e,0x32,0x2e,0x31,0x32,0x20,0x28,0x55,0x62,0x75,

0x6e,0x74,0x75,0x29,0x20,0x6d,0x6f,0x64,0x5f,0x6a,0x6b,0x2f,0x31,0x2e,0x32,0x2e,

0x32,0x36,0x20,0x50,0x48,0x50,0x2f,0x35,0x2e,0x32,0x2e,0x31,0x30,0x2d,0x32,0x75,

0x62,0x75,0x6e,0x74,0x75,0x36,0x2e,0x34,0x20,0x77,0x69,0x74,0x68,0x20,0x53,0x75,

0x68,0x6f,0x73,0x69,0x6e,0x2d,0x50,0x61,0x74,0x63,0x68,0x0d,0x0a,0x53,0x65,0x74,

0x2d,0x43,0x6f,0x6f,0x6b,0x69,0x65,0x3a,0x20,0x73,0x75,0x70,0x70,0x6c,0x79,0x66,

0x72,0x61,0x6d,0x65,0x55,0x73,0x65,0x72,0x49,0x64,0x3d,0x32,0x31,0x30,0x2e,0x32,

0x37,0x2e,0x31,0x30,0x2e,0x32,0x32,0x2e,0x31,0x33,0x34,0x33,0x30,0x33,0x33,0x30,

0x35,0x36,0x32,0x34,0x38,0x39,0x38,0x31,0x3b,0x20,0x70,0x61,0x74,0x68,0x3d,0x2f,

0x3b,0x20,0x65,0x78,0x70,0x69,0x72,0x65,0x73,0x3d,0x53,0x61,0x74,0x2c,0x20,0x32,

0x32,0x2d,0x4a,0x75,0x6c,0x2d,0x31,0x37,0x20,0x30,0x38,0x3a,0x34,0x34,0x3a,0x31,

0x36,0x20,0x47,0x4d,0x54,0x3b,0x20,0x64,0x6f,0x6d,0x61,0x69,0x6e,0x3d,0x2e,0x73,

0x75,0x70,0x70,0x6c,0x79,0x66,0x72,0x61,0x6d,0x65,0x2e,0x63,0x6f,0x6d,0x0d,0x0a,

0x58,0x2d,0x50,0x6f,0x77,0x65,0x72,0x65,0x64,0x2d,0x42,0x79,0x3a,0x20,0x53,0x65,

0x72,0x76,0x6c,0x65,0x74,0x20,0x32,0x2e,0x34,0x3b,0x20,0x4a,0x42,0x6f,0x73,0x73,

0x2d,0x34,0x2e,0x30,0x2e,0x35,0x2e,0x47,0x41,0x20,0x28,0x62,0x75,0x69,0x6c,0x64,

0x3a,0x20,0x43,0x56,0x53,0x54,0x61,0x67,0x3d,0x42,0x72,0x61,0x6e,0x63,0x68,0x5f,

0x34,0x5f,0x30,0x20,0x64,0x61,0x74,0x65,0x3d,0x32,0x30,0x30,0x36,0x31,0x30,0x31,

0x36,0x32,0x33,0x33,0x30,0x29,0x2f,0x54,0x6f,0x6d,0x63,0x61,0x74,0x2d,0x35,0x2e,

0x35,0x0d,0x0a,0x53,0x65,0x74,0x2d,0x43,0x6f,0x6f,0x6b,0x69,0x65,0x3a,0x20,0x4a,

0x53,0x45,0x53,0x53,0x49,0x4f,0x4e,0x49,0x44,0x3d,0x38,0x39,0x46,0x46,0x44,0x37,

0x31,0x43,0x41,0x38,0x37,0x38,0x38,0x38,0x38,0x36,0x30,0x35,0x46,0x33,0x37,0x35,

0x46,0x38,0x41,0x36,0x37,0x39,0x41,0x36,0x41,0x32,0x2e,0x6a,0x62,0x6f,0x73,0x73,

0x5f,0x70,0x61,0x72,0x74,0x73,0x65,0x61,0x72,0x63,0x68,0x33,0x3b,0x20,0x50,0x61,

0x74,0x68,0x3d,0x2f,0x0d,0x0a,0x43,0x6f,0x6e,0x74,0x65,0x6e,0x74,0x2d,0x4c,0x65,

0x6e,0x67,0x74,0x68,0x3a,0x20,0x34,0x33,0x0d,0x0a,0x4b,0x65,0x65,0x70,0x2d,0x41,

0x6c,0x69,0x76,0x65,0x3a,0x20,0x74,0x69,0x6d,0x65,0x6f,0x75,0x74,0x3d,0x32,0x2c,

0x20,0x6d,0x61,0x78,0x3d,0x31,0x30,0x30,0x0d,0x0a,0x43,0x6f,0x6e,0x6e,0x65,0x63,

0x74,0x69,0x6f,0x6e,0x3a,0x20,0x4b,0x65,0x65,0x70,0x2d,0x41,0x67,0x69,0x76,0x65,

0x0d,0x0a,0x43,0x6f,0x6e,0x74,0x65,0x6e,0x74,0x2d,0x54,0x79,0x70,0x65,0x3a,0x20,

0x69,0x6d,0x61,0x67,0x65,0x2f,0x67,0x69,0x66,0x0d,0x0a,0x0d,0x0a};

int len=sizeof(payload);

ip\_id = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

plibnet\_ack=libnet\_init(LIBNET\_LINK,net\_dev,errbuf);

if(ptag==-1)

printf("the error\n");

libnet\_build\_tcp(s\_port,d\_port,seq,ack,

TH\_ACK,1460,0,10,LIBNET\_TCP\_H+len,payload,len,plibnet\_ack,0);

libnet\_build\_tcp\_options(NULL,0,plibnet\_ack,0);

libnet\_build\_ipv4(LIBNET\_IPV4\_H+LIBNET\_TCP\_H+len,

IPTOS\_LOWDELAY,ip\_id,0,199,IPPROTO\_TCP,0,s\_ip.s\_addr,d\_ip.s\_addr,NULL,0,plibnet\_ack,0);

ptag=libnet\_build\_ethernet(d\_mac,//dst

s\_mac,//src

ETHERTYPE\_IP,

NULL,//(u\_int8\_t\*)payload,

0,//sizeof(payload),

plibnet\_ack,

0);

libnet\_write(plibnet\_ack);

libnet\_destroy(plibnet\_ack);

}

void p\_send\_fin\_ack(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip,u\_int32\_t seq,u\_int32\_t ack,u\_int16\_t s\_port,u\_int16\_t d\_port)

{

libnet\_t \*plibnet\_ack;

libnet\_ptag\_t ptag;

u\_int16\_t ip\_id;

char errbuf[100];

u\_char payload[]={};

ip\_id = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

plibnet\_ack=libnet\_init(LIBNET\_LINK,net\_dev,errbuf);

if(ptag==-1)

printf("the error\n");

libnet\_build\_tcp(s\_port,d\_port,seq,ack,

TH\_FIN|TH\_ACK,1460,0,10,LIBNET\_TCP\_H+20,NULL,0,plibnet\_ack,0);

libnet\_build\_tcp\_options(NULL,0,plibnet\_ack,0);

libnet\_build\_ipv4(LIBNET\_IPV4\_H+LIBNET\_TCP\_H+sizeof(payload),

IPTOS\_LOWDELAY,ip\_id,0,99,IPPROTO\_TCP,0,s\_ip.s\_addr,d\_ip.s\_addr,NULL,0,plibnet\_ack,0);

ptag=libnet\_build\_ethernet(d\_mac,//dst

s\_mac,//src

ETHERTYPE\_IP,

NULL,//(u\_int8\_t\*)payload,

0,//sizeof(payload),

plibnet\_ack,

0);

libnet\_write(plibnet\_ack);

libnet\_destroy(plibnet\_ack);

}

void ip\_packet(const u\_char\* packet)

{

struct iphdr\* ip\_hdr;

struct icmphdr\*icmp\_hdr;

ip\_hdr=(struct iphdr\*)(packet+14);

u\_char ip\_type=ip\_hdr->protocol;

switch(ip\_type)

{

case IPPROTO\_ICMP:

//printf("this is icmp\n");

send\_icmp(packet);

break;

case IPPROTO\_TCP:

if(sendpacketflag==0)

tcp\_packet(packet);

break;

default :

//printf("hhhhhhhhhhhhhhhhhhhh\n");

change\_ip();

if(sendpacketflag==1)

send\_packet();

break;//printf("this is ip other packet!\n");

}

}

void send\_icmp(const u\_char\*packet)

{

struct iphdr \*ip\_hdr=NULL;

struct ethhdr \*Ether;

struct icmphdr \*icmp\_hdr=NULL;

u\_char \*s\_mac,\*d\_mac;

s\_mac=(u\_char\*)malloc(6);

d\_mac=(u\_char\*)malloc(6);

u\_int type;

Ether= (struct Eth\_header \*) (packet+0);//获取ARP头起始地址

ip\_hdr=(struct iphdr\*)(packet+14);

icmp\_hdr=(struct icmphdr\*)(packet+14+20);

type=icmp\_hdr->type;

if(type==8)

{

int i,tag\_num;

struct in\_addr d\_ip;

memcpy((void \*) &d\_ip, (void \*) &ip\_hdr->daddr, sizeof (struct in\_addr));

tag\_num=d\_ip.s\_addr>>24;

for(i=0;i<ip\_num;i++)

{

int host\_num;

host\_num=ip\_host\_addr[i].src\_ip.s\_addr>>24;

if(host\_num==tag\_num){

libnet\_t \*plibnet\_icmp;

libnet\_ptag\_t ptag;

u\_int16\_t ip\_id;

char errbuf[100];

u\_char payload[32]="abcdefghijklmnopqrstuvwabcdefghi";

ip\_id = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

s\_mac=(u\_char \*)&Ether->h\_source;

d\_mac=(u\_char \*)&Ether->h\_dest;

plibnet\_icmp=libnet\_init(LIBNET\_LINK,net\_dev,errbuf);

u\_int16\_t id=icmp\_hdr->un.echo.id;

u\_int16\_t seq=icmp\_hdr->un.echo.sequence;

ptag=libnet\_build\_icmpv4\_echo(

ICMP\_ECHOREPLY,

0,//code

0,//sum

id,//id,//id

seq,//seq,//seq

payload,

sizeof(payload),//LIBNET\_ICMPV4\_ECHO\_H,

plibnet\_icmp,

0);

ptag=libnet\_build\_ipv4(

LIBNET\_IPV4\_H+LIBNET\_ICMPV4\_ECHO\_H+sizeof(payload),

0,

ip\_id,

0,

10,

IPPROTO\_ICMP,

0,

ip\_hdr->daddr,

ip\_hdr->saddr,

NULL,//payload,

0,//sizeof(payload),

plibnet\_icmp,

0 );

ptag=libnet\_build\_ethernet(

s\_mac,

d\_mac,

ETHERTYPE\_IP,

NULL,//(u\_int8\_t\*)payload,

0,//sizeof(payload),

plibnet\_icmp,

0);

libnet\_write(plibnet\_icmp);

libnet\_destroy(plibnet\_icmp);

// }

}

}

}

}

void tcp\_packet(const u\_char \*packet)

{

int s;

struct iphdr \*ip\_hdr=NULL;

struct ethhdr \*Ether;

struct tcphdr \*tcp\_hdr=NULL;

u\_char \*s\_mac,\*d\_mac;

s\_mac=(u\_char\*)malloc(6);

d\_mac=(u\_char\*)malloc(6);

ip\_hdr=(struct iphdr\*)(packet+14);

tcp\_hdr=(struct tcphdr\*)(packet+14+20);

u\_int type;

u\_int16\_t d\_prt=ntohs(tcp\_hdr->dest);

u\_int16\_t flag\_syn=tcp\_hdr->syn;

//printf("the flag\_syn is:%d\n",flag\_syn);

u\_int16\_t flag\_ack=tcp\_hdr->ack;

//printf("the flag\_ack is:%d\n",flag\_ack);

switch(d\_prt)

{

case 139:

if(flag\_syn==1)

send\_ack(packet,3);

//printf("++++++++++++++139+++++++++++++\n");

break;

case 445:

if(flag\_syn==1)

send\_ack(packet,5);

break;

case 3389:

if(flag\_syn==1)

send\_ack(packet,4);

break;

case 1025:

if(flag\_syn==1)

send\_ack(packet,7);

break;

case 1029:

if(flag\_syn==1)

send\_ack(packet,3);

break;

case 1028:

if(flag\_syn==1)

send\_ack(packet,2);

break;

case 19:

if(flag\_syn==1)

send\_ack(packet,4);

break;

case 42:

if(flag\_syn==1)

send\_ack(packet,5);

break;

case 161:

if(flag\_syn==1)

send\_ack(packet,5);

break;

case 102:

if(flag\_syn==1)

send\_ack(packet,1);

break;

case 1713:

if(flag\_syn==1)

send\_ack(packet,6);

break;

case 800:

if(flag\_syn==1)

send\_ack(packet,4);

break;

case 123:

if(flag\_syn==1)

send\_ack(packet,7);

break;

default :if((flag\_syn==1)|(flag\_ack==1))

send\_rst(packet);

break;

}

}

void send\_rst(const u\_char \*packet)

{

struct ethhdr\* eth\_hdr;

struct iphdr\* ip\_hdr;

struct tcphdr\* tcp\_hdr;

eth\_hdr=(struct ethhdr\*)(packet+0);

ip\_hdr=(struct iphdr\*)(packet+14);

tcp\_hdr=(struct tcphdr\*)(packet+14+20);

u\_char \*s\_mac,\*d\_mac;

s\_mac=(u\_char\*)malloc(6);

d\_mac=(u\_char\*)malloc(6);

libnet\_t \*plibnet\_ack;

libnet\_ptag\_t ptag;

u\_int16\_t ip\_id;

char errbuf[100];

u\_char \*payload="";

ip\_id = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

s\_mac=eth\_hdr->h\_source;

//printf("the s\_mac is:%02x-%02x-%02x-%02x-%02x-%02x\n",\*s\_mac,\*(s\_mac+1),\*(s\_mac+2),\*(s\_mac+3),\*(s\_mac+4),\*(s\_mac+5));

d\_mac=eth\_hdr->h\_dest;

//printf("the d\_mac is:%02x-%02x-%02x-%02x-%02x-%02x\n",\*d\_mac,\*(d\_mac+1),\*(d\_mac+2),\*(d\_mac+3),\*(d\_mac+4),\*(d\_mac+5));

struct in\_addr s\_ip,d\_ip;

s\_ip.s\_addr=ip\_hdr->saddr;

d\_ip.s\_addr=ip\_hdr->daddr;

int i,tag\_num;

tag\_num=d\_ip.s\_addr>>24;

//printf("the ip\_last is:%d\n",tag\_num);

for(i=0;i<ip\_num;i++)

{

int host\_num,send\_n;

send\_n=ip\_hdr->saddr>>24;

host\_num=ip\_host\_addr[i].src\_ip.s\_addr>>24;

//printf("the host num is:%d\n",host\_num);

if(host\_num==tag\_num){

if(ip\_mac\_addr[send\_n].statue==1){

plibnet\_ack=libnet\_init(LIBNET\_LINK,net\_dev,errbuf);

if(ptag==-1)

printf("the error\n");

libnet\_build\_tcp(htons(tcp\_hdr->dest),

htons(tcp\_hdr->source),

htonl(tcp\_hdr->ack),

htonl(tcp\_hdr->ack\_seq),

TH\_RST|TH\_ACK,1460,0,10,

LIBNET\_TCP\_H+20,NULL,0,plibnet\_ack,0);

libnet\_build\_tcp\_options(NULL,0,plibnet\_ack,0);

libnet\_build\_ipv4(LIBNET\_IPV4\_H+LIBNET\_TCP\_H+sizeof(payload),

IPTOS\_LOWDELAY,32,0,99,IPPROTO\_TCP,0,

ip\_hdr->daddr,ip\_hdr->saddr,NULL,0,plibnet\_ack,0);

ptag=libnet\_build\_ethernet(

s\_mac,//dst

d\_mac,//src

ETHERTYPE\_IP,

NULL,//(u\_int8\_t\*)payload,

0,//sizeof(payload),

plibnet\_ack,

0);

libnet\_write(plibnet\_ack);

libnet\_destroy(plibnet\_ack);

}

}

}

}

void send\_ack(const u\_char\* packet,int use\_n)

{

struct ethhdr\* eth\_hdr;

struct iphdr\* ip\_hdr;

struct tcphdr\* tcp\_hdr;

eth\_hdr=(struct ethhdr\*)(packet+0);

ip\_hdr=(struct iphdr\*)(packet+14);

tcp\_hdr=(struct tcphdr\*)(packet+14+20);

u\_char \*s\_mac,\*d\_mac;

s\_mac=(u\_char\*)malloc(6);

d\_mac=(u\_char\*)malloc(6);

libnet\_t \*plibnet\_ack;

libnet\_ptag\_t ptag;

u\_int16\_t ip\_id;

char errbuf[100];

u\_char \*payload="";

ip\_id = (u\_int16\_t)libnet\_get\_prand(LIBNET\_PR16);

s\_mac=eth\_hdr->h\_source;

//printf("the s\_mac is:%02x-%02x-%02x-%02x-%02x-%02x\n",\*s\_mac,\*(s\_mac+1),\*(s\_mac+2),\*(s\_mac+3),\*(s\_mac+4),\*(s\_mac+5));

d\_mac=eth\_hdr->h\_dest;

//printf("the d\_mac is:%02x-%02x-%02x-%02x-%02x-%02x\n",\*d\_mac,\*(d\_mac+1),\*(d\_mac+2),\*(d\_mac+3),\*(d\_mac+4),\*(d\_mac+5));

struct in\_addr s\_ip,d\_ip;

s\_ip.s\_addr=ip\_hdr->saddr;

d\_ip.s\_addr=ip\_hdr->daddr;

int i,tag\_num;

tag\_num=d\_ip.s\_addr>>24;

//printf("the ip\_last is:%d\n",tag\_num);

for(i=0;i<ip\_num;i++)

{

int host\_num;

host\_num=ip\_host\_addr[i].src\_ip.s\_addr>>24;

//printf("the host num is:%d\n",host\_num);

if(ip\_host\_addr[i].use\_num>use\_n){

if(host\_num==tag\_num){

plibnet\_ack=libnet\_init(LIBNET\_LINK,net\_dev,errbuf);

if(ptag==-1)

printf("the error\n");//

libnet\_build\_tcp(htons(tcp\_hdr->dest),

htons(tcp\_hdr->source),

htonl(tcp\_hdr->ack),

htonl(tcp\_hdr->ack\_seq),

TH\_ACK|TH\_SYN,1460,0,10,

LIBNET\_TCP\_H+20,NULL,0,plibnet\_ack,0);

libnet\_build\_tcp\_options(NULL,0,plibnet\_ack,0);

libnet\_build\_ipv4(LIBNET\_IPV4\_H+LIBNET\_TCP\_H+sizeof(payload),

IPTOS\_LOWDELAY,32,0,199,IPPROTO\_TCP,0,

ip\_hdr->daddr,ip\_hdr->saddr,NULL,0,plibnet\_ack,0);

ptag=libnet\_build\_ethernet(

s\_mac,//dst

d\_mac,//src

ETHERTYPE\_IP,

NULL,//(u\_int8\_t\*)payload,

0,//sizeof(payload),

plibnet\_ack,

0);

libnet\_write(plibnet\_ack);

libnet\_destroy(plibnet\_ack);

}

}

}

}

void chgnew\_ip(int num,int ip\_num)

{

int temp;

int i;

for(i=0;i<ip\_num;i++)

{

temp=ip\_host\_addr[i].src\_ip.s\_addr>>24;

if(temp==num)

{

//ip\_host\_addr[i].src\_ip.s\_addr=0;

break;

}

}

}

void arp\_packet(const u\_char\* packet)//应答arp询问的mac

{

struct Arp\_header \*arp\_header\_prl;

u\_short opt\_code;

u\_char buf[6];

struct in\_addr source\_ip\_address,destion\_ip\_address;

arp\_header\_prl = (struct Arp\_header \*) (packet+ 14);//获取ARP头起始地址

opt\_code = ntohs(arp\_header\_prl->arp\_operation\_code);

memcpy((void \*) & source\_ip\_address, (void \*) & arp\_header\_prl->arp\_source\_ip\_address, sizeof (struct in\_addr));

memcpy((void \*) & destion\_ip\_address, (void \*) & arp\_header\_prl->arp\_destination\_ip\_address, sizeof (struct in\_addr));

memcpy(buf,(void \*) & arp\_header\_prl->arp\_source\_ethernet\_address,6);

if(opt\_code == 2)///判断是否有回复若是2者有回复，若为1者无应答

{

printf("jjjfslfljsdj\n");

int i;

for(i=0;i<ip\_num;i++)

{

int host\_num,send\_num,tag\_num;

host\_num=ip\_host\_addr[i].src\_ip.s\_addr>>24;

send\_num=source\_ip\_address.s\_addr>>24;

tag\_num=destion\_ip\_address.s\_addr>>24;

if(host\_num==send\_num&&ip\_mac\_addr[send\_num].statue==0){

ip\_mac\_addr[send\_num].statue=1;

//chgnew\_ip(send\_num,ip\_num);

writefile\_chg(ip\_for\_host);

writefile\_host(ip\_for\_host,ip\_num);

}

}

int k,j;

for(k=0;k<ip\_num;k++)

{

int temp=ip\_host\_addr[k].src\_ip.s\_addr>>24;

if(temp==0)

for(j=k;j<ip\_num-1;j++)

{

ip\_host\_addr[k].src\_ip.s\_addr=ip\_host\_addr[k+1].src\_ip.s\_addr;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*主机上线了\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

ip\_num=ip\_num-1;

}

}

}

else if(opt\_code==1)

{

printf("--------------------------------the arp is reply\n");

int i,tag\_num;

tag\_num=destion\_ip\_address.s\_addr>>24;

int flag;

for(i=0;i<ip\_num;i++)

{

int host\_num;

host\_num=ip\_host\_addr[i].src\_ip.s\_addr>>24;

if(host\_num==tag\_num){

if(ip\_mac\_addr[host\_num].statue==0){

flag=1;

send\_arp(ip\_mac\_addr[host\_num].src\_mac

,buf

,ip\_mac\_addr[host\_num].src\_ip

,source\_ip\_address,2);////send the untrue host mac

printf("--------------------------------the arp is reply\n");

}

}

}

/\* if(flag==1)

{

FILE \*fp;

char buf[16];

sprintf(buf,"%s",source\_ip\_address);

fp=fopen("bad.txt","w")

fseek(fp,16,2);

fputs(fp,buf);

bad\_ip[bad\_n++].src\_ip=source\_ip\_address;

}\*/

}

}

void \*cap\_req\_arp()//抓本机arp

{

//printf("the success!\n");

char filter\_arp[]="arp";

//printf("this is capture local arp request--------------\n");

handle\_req\_arp = pcap\_open\_live(net\_dev, BUFSIZ, 1, 0, err\_cnt);

struct bpf\_program bpf\_filter;///设置过滤条件\*/

pcap\_compile(handle\_req\_arp, &bpf\_filter, filter\_arp, 0, net\_ip);//编译过滤条件

pcap\_setfilter(handle\_req\_arp, &bpf\_filter);//设置过滤器

if(pcap\_datalink(handle\_req\_arp) != DLT\_EN10MB)///数据链路层

return 0;

pcap\_loop(handle\_req\_arp,-1,callback\_inita\_statue,0);

return 0;

}

void callback\_inita\_statue(u\_char \*argument, const struct pcap\_pkthdr\* header, const u\_char\* packet)

{

struct Arp\_header \*arp\_header\_prl;

u\_short opt\_code;

int local\_num;

int num,tag\_num;

struct in\_addr source\_ip\_address,destion\_ip\_address;

arp\_header\_prl = (struct Arp\_header \*) (packet+ 14);//获取ARP头起始地址

opt\_code = ntohs(arp\_header\_prl->arp\_operation\_code);

memcpy((void \*) & source\_ip\_address, (void \*) & arp\_header\_prl->arp\_source\_ip\_address, sizeof (struct in\_addr));

memcpy((void \*) & destion\_ip\_address, (void \*) & arp\_header\_prl->arp\_destination\_ip\_address, sizeof (struct in\_addr));

local\_num=local\_ip.s\_addr>>24;

tag\_num=destion\_ip\_address.s\_addr>>24;

num=source\_ip\_address.s\_addr>>24;

//printf("the num is:%d\n",num);

if(opt\_code == 2)///判断是否有回复若是2者有回复，若为1者无应答

{

if(local\_num==tag\_num){

ip\_mac\_addr[num].statue=1;

//printf("##########################\n");

}

}

if(tag\_num==254)

{

pcap\_breakloop(handle\_req\_arp);

printf("termin the pcap\_loop()\n");

}

}

void mac\_ip\_diver()

{

int n,j;

int num[254];

unsigned char mac\_part1[10][3]={

0x00,0x0f,0xe2,

0x00,0x26,0x82,

0x08,0xa0,0x07,

0x00,0x16,0xb8,

0x00,0x26,0x54,

0x00,0x1e,0x4f,

0x00,0x1c,0x29,

0x00,0x1c,0x58,

0x00,0x06,0x4b,

0x00,0x1a,0x6b};

unsigned long tmp1\_ip,tmp2\_ip;

pcap\_lookupnet(net\_dev, &net\_ip, &net\_mask, err\_cnt);//寻找网络接

for(n=1;n<255;n++){

tmp1\_ip=htonl(net\_ip)+n;

tmp2\_ip=ntohl(tmp1\_ip);////此处有问题

struct in\_addr ip\_temp=\*(struct in\_addr \*)&(tmp2\_ip);

ip\_mac\_addr[n].src\_ip=ip\_temp;

}

int local\_n=local\_ip.s\_addr>>24;

for(j=1;j<255;j++){

if(j!=local\_n){

int k;

u\_char mac\_part2[3];

rand\_mem(mac\_part2,3);///随机产生MAC地址序列

num[j]=rand()%10;

k=num[j];

ip\_mac\_addr[j].src\_mac[0]=mac\_part1[k][0];

ip\_mac\_addr[j].src\_mac[1]=mac\_part1[k][1];

ip\_mac\_addr[j].src\_mac[2]=mac\_part1[k][2];

ip\_mac\_addr[j].src\_mac[3]=mac\_part2[0];

ip\_mac\_addr[j].src\_mac[4]=mac\_part2[1];

ip\_mac\_addr[j].src\_mac[5]=mac\_part2[2];

}

else

continue;

}

///////为虚假连接分配常见IP地址和MAC地址

dst\_addr[0].src\_ip.s\_addr=inet\_addr("222.10.1.39");

rand\_mem(dst\_addr[0].src\_mac,6);;

dst\_addr[1].src\_ip.s\_addr=inet\_addr("118.67.212.80");

rand\_mem(dst\_addr[1].src\_mac,6);

dst\_addr[2].src\_ip.s\_addr=inet\_addr("222.89.16.13");

rand\_mem(dst\_addr[2].src\_mac,6);

dst\_addr[3].src\_ip.s\_addr=inet\_addr("222.202.76.132");

rand\_mem(dst\_addr[3].src\_mac,6);

dst\_addr[4].src\_ip.s\_addr=inet\_addr("112.90.34.77");

rand\_mem(dst\_addr[4].src\_mac,6);

dst\_addr[5].src\_ip.s\_addr=inet\_addr("202.93.76.131");

rand\_mem(dst\_addr[5].src\_mac,6);

dst\_addr[6].src\_ip.s\_addr=inet\_addr("222.27.14.13");

rand\_mem(dst\_addr[6].src\_mac,6);

dst\_addr[7].src\_ip.s\_addr=inet\_addr("61.152.89.69");

rand\_mem(dst\_addr[7].src\_mac,6);

dst\_addr[8].src\_ip.s\_addr=inet\_addr("183.60.4.206");

rand\_mem(dst\_addr[8].src\_mac,6);

dst\_addr[9].src\_ip.s\_addr=inet\_addr("58.215.57.171");

rand\_mem(dst\_addr[9].src\_mac,6);

}

int get\_interface()

{

int n;

register int reqfd;

struct ifreq buf[MAXINTERFACES];

struct ifconf ifc;

//struct arpreq arp;

int interface\_num;

reqfd = socket(AF\_INET, SOCK\_DGRAM, 0);

ifc.ifc\_len = sizeof(buf);

ifc.ifc\_buf=(caddr\_t)buf;

net\_dev=pcap\_lookupdev(err\_cnt);

printf("the dev is :%s\n",net\_dev);

///用SIOCGIFCONF 获取所有接口的清单

if(!ioctl(reqfd,SIOCGIFCONF,(char\*)&ifc))

{

interface\_num = ifc.ifc\_len / sizeof(struct ifreq);

while(interface\_num-->0)///解决多网卡问题

{

if(strcmp(buf[interface\_num].ifr\_name,net\_dev)==0)

{

if(!(ioctl(reqfd,SIOCGIFADDR,(char\*)&buf[interface\_num])))

{

local\_ip=((struct sockaddr\_in\*)(&buf[interface\_num].ifr\_addr))->sin\_addr;

n=local\_ip.s\_addr>>24;

ip\_mac\_addr[n].src\_ip=local\_ip;

}

if(!(ioctl(reqfd, SIOCGIFHWADDR,(char \*)&buf[interface\_num])))// SIOCGIFHWADDR

{

mac\_addr=(u\_char\*)&buf[interface\_num].ifr\_hwaddr.sa\_data[0];

memcpy(ip\_mac\_addr[n].src\_mac,mac\_addr,6);

}

}

}

}

return 0;

}

void rand\_mem(u\_char \*ptr,int length)

{

int i;

for(i=0;i<length;i++,ptr++)

\*ptr=libnet\_get\_prand(LIBNET\_PR8);///获取0——255的随机数

}

function.h

#ifndef FUNCTION\_H\_INCLUDED

#define FUNCTION\_H\_INCLUDED

#include <pcap/pcap.h>

#endif // FUNCTION\_H\_INCLUDED

int initial();

int get\_interface();

void create\_thread();

void wait\_thread();

void\*cap\_req\_arp();

void send\_arp(u\_char src\_mac[],u\_char dst\_mac[],struct in\_addr src\_ip,struct in\_addr dst\_ip,int type);

void send\_local\_arp();

void callback\_inita\_statue(u\_char \*argument, const struct pcap\_pkthdr\* packet\_header\_arp, const u\_char\* packet\_content\_arp);

void cap\_packet();

void get\_rand\_ip(int num);

void ran(int buffer[], int length);

void callback\_packet(u\_char \*argument, const struct pcap\_pkthdr\* header, const u\_char\* packet);

void arp\_packet();

void ip\_packet(const u\_char\* packet);

void icmp\_packet(const u\_char\*packet);

void tcp\_packet(const u\_char \*packet);

//void \*cap\_rpl\_icmp();

void send\_icmp(const u\_char\*packet);

void send\_ack(const u\_char\* packet,int n);

void send\_rst(const u\_char\* packet);

void get\_use\_n(int buf[],int len);

void send\_packet();

void chg\_ipaddr();

void num(int buf[],int len);

void writefile\_host(int ip\_chg,int ip\_sum);

void writefile\_chg(int len);

void readfile();

void chgnew\_ip(int num,int ip\_num);

void p\_send\_syn(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip,u\_int32\_t seq,u\_int32\_t ack,u\_int16\_t s\_port,u\_int16\_t d\_port);

void p\_send\_syn\_ack(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip,u\_int32\_t seq,u\_int32\_t ack,u\_int16\_t s\_port,u\_int16\_t d\_port);

void p\_send\_ack(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip,u\_int32\_t seq,u\_int32\_t ack,u\_int16\_t s\_port,u\_int16\_t d\_port);

void p\_send\_tcp(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip,u\_int32\_t seq,u\_int32\_t ack,u\_int16\_t s\_port,u\_int16\_t d\_port);

void p\_send\_http(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip,u\_int32\_t seq,u\_int32\_t ack,u\_int16\_t s\_port,u\_int16\_t d\_port);

void p\_send\_fin\_ack(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip,u\_int32\_t seq,u\_int32\_t ack,u\_int16\_t s\_port,u\_int16\_t d\_port);

void p\_send\_udp(u\_char \*s\_mac,struct in\_addr s\_ip,u\_char \*d\_mac,struct in\_addr d\_ip);

void change\_ip();

void fileadd(char str[]);

void ini\_arp();

void net\_ip\_div(int net\_n,int ip\_n);

void listen\_reply();

void listen\_arp();

void listen\_icmp();

void \*prt();

//void icmp\_reply\_callback(u\_char \*argument, const struct pcap\_pkthdr\* packet\_header\_icmp, const u\_char\* packet\_content\_icmp);

void \*capture\_icmp( );

void mac\_ip\_diver();

void rand\_mem(u\_char \*ptr,int length);

void arp\_protocol\_packet\_callback(u\_char \*argument, const struct pcap\_pkthdr\* packet\_header\_arp, const u\_char\* packet\_content\_arp);

void \*capture();//抓取ARP包结构

void \*capture\_reply\_arp( );

void arp\_reply\_callback(u\_char \*argument, const struct pcap\_pkthdr\* packet\_header, const u\_char\* packet\_content);

//void Send\_arp(u\_char \*s\_mac,u\_char \*d\_mac,struct in\_addr s\_ip,struct in\_addr d\_ip);

//void send\_icmp( const struct pcap\_pkthdr\* packet\_header\_icmp, const u\_char\* packet\_content\_icmp);

main.c

#include <stdio.h>

#include <stdlib.h>

#include "main.h"

#include "function.h"

char filter\_arp[]="arp";

int use\_num=1;

//ip\_num=50;

//ip\_for\_host=3;

//time\_change=5;

//sendpacketflag=0;

int main(int argc, char \*argv[])

{

int temp;

temp=atoi(argv[1]);

ip\_for\_host=atoi(argv[2]);

ip\_num=temp+ip\_for\_host;

time\_change=atoi(argv[3]);

sendpacketflag=atoi(argv[4]);

printf("the ip\_num:%d\n",ip\_num);

printf("the ip\_for\_host:%d\n",ip\_for\_host);

printf("the tiem\_change:%d\n",time\_change);

printf("the senflag:%d\n",sendpacketflag);

initial();

printf("Hello world!\n");

return 0;

}

main.h

#ifndef MAIN\_H\_INCLUDED

#define MAIN\_H\_INCLUDED

#endif // MAIN\_H\_INCLUDED

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <netdb.h>

#include <netinet/in.h>

#include <net/ethernet.h>

#include <linux/types.h>

#include <sys/socket.h>

#include <sys/ioctl.h>

#include <pthread.h>///linux 线程函数

#include <arpa/inet.h>

#include <pcap/pcap.h>

#include <libnet/libnet-macros.h>

#include <linux/if\_ether.h>

//#include <linux/if\_arp.h>

#include "function.h"

#include <time.h>

#include <libnet.h>

#include <linux/icmp.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include "shm\_com.h"

#define MAC\_BCAST\_ADDR (uint8\_t \*) "\xff\xff\xff\xff\xff\xff"////定义广播类型的MAC

#define ETHER\_LEN sizeof(struct Eth\_header) ///以太帧头大小

#define ARP\_LEN sizeof(struct Arp\_header) //ARP头大小

#define MAXINTERFACES 16

pthread\_mutex\_t mut;

char err\_cnt[PCAP\_ERRBUF\_SIZE]; ///定义pcap错误类型

int time\_change;

int min;

int ip\_chg\_n;

int sendpacketflag; //定义接口变量

u\_char \*mac\_temp;

struct in\_addr ip\_temp[4];

pcap\_t\* pcap\_handle;

char filter\_arp[];

///直接收ARP包

bpf\_u\_int32 net\_mask;

//bpf\_u\_int32 net\_ip;

char \*net\_dev; //the local dev

uint32\_t net\_ip;//the local netip addr

//u\_char MAC[32];

u\_char \*mac\_addr;///the local mac

struct in\_addr local\_ip;

////全局变量记录本季网络地址，物理地址，IP地址

struct ip\_mac

{

struct in\_addr src\_ip;

u\_char src\_mac[6];

int statue;

int use\_num;///为了避免端口扫描都是一样的结果///xujiazhuji yu Ip change qubie

int chg;

}ip\_mac\_addr[255],ip\_host\_addr[255],dst\_addr[10],net\_ip\_addr[255][255],bad\_ip[255];

int use\_num;/// 为了循环从1开始

int num\_ip;//

int bad\_n;

struct Eth\_header

{ //以太帧头部结构

u\_int8\_t ether\_dhost[6];

u\_int8\_t ether\_shost[6];

u\_int16\_t ether\_type;

}eth\_header;

typedef u\_int32\_t in\_addr\_t;

struct Arp\_header

{ //arp头部结构

u\_int16\_t arp\_hardware\_type;

u\_int16\_t arp\_protocol\_type;

u\_int8\_t arp\_hardware\_length;

u\_int8\_t arp\_protocol\_length;

u\_int16\_t arp\_operation\_code;

u\_int8\_t arp\_source\_ethernet\_address[6];

u\_int8\_t arp\_source\_ip\_address[4];

u\_int8\_t arp\_destination\_ethernet\_address[6];

u\_int8\_t arp\_destination\_ip\_address[4];

}arp\_header;

//////thread

pthread\_mutex\_t mut;

pthread\_t thread\_req\_arp;

pthread\_t thread\_packet;

pcap\_t\* handle\_req\_arp;

int ip\_num;///虚假的IP个数

int ip\_for\_host;///主机IP跳变个数

shm\_com.h

#ifndef SHM\_COM\_H

#define SHM\_COM\_H

#define TEXT\_SZ 2048

struct \_shm\_data

{

int is\_written;

char data[TEXT\_SZ];

};

typedef struct \_shm\_data shm\_data ;

#endif

# 附录B: 反侦察中心代码

* stkernel

/\*

\* author:Zhang Zijian

\*

\* This program is free software; you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation; either version 2 of the License, or

\* (at your option) any later version.

\*

\* This program is writed to authorize data group from internet to PC by

\* banning all of actibe connection requests from outer cyber

\*

\* Attention: This is a kernel program. If you want to run this, you have

\* to be a root or a superuser

\*

\*/

#include<linux/module.h>

#include<linux/init.h>

#include<linux/skbuff.h>

#include<linux/netfilter.h>

#include<linux/netfilter\_ipv4.h>

#include<linux/string.h>

#include<linux/ip.h>

#include<linux/in.h>

#include<linux/fs.h>

#include<linux/string.h>

#include<linux/mm.h>

#include<linux/syscalls.h>

#include<linux/unistd.h>

#include<linux/proc\_fs.h>

#include<asm/unistd.h>

#include<asm/uaccess.h>

#include<net/ip.h>

#include<linux/if\_ether.h>

#include<linux/icmp.h>

MODULE\_LICENSE("Dual BSD/GPL");

#define BL\_MAX 128

#define LL\_MAX 256

static struct linklist{

unsigned int addr, port, protocol;

} \*ll\_head=0;

static struct blacklist{

unsigned int addr, poss;

} \*bl\_head=0;

static struct nf\_hook\_ops nf\_hook\_in;

static struct nf\_hook\_ops nf\_hook\_out;

int blnum=0, lastdel=0;

static int proc\_read(char \*buff,

char \*\*buff\_start,

off\_t offset,

int count,

int \*eof,

void \*data)

{

char temp[4096];

unsigned int m,t,ipt,j,i=0,p=0;

if (offset>0) return 0;

memset(temp,0,sizeof(temp));

while (bl\_head[i].addr){

ipt=bl\_head[i].addr;

if (ipt==0) continue;

for(j=0;j<4;j++){

t=ipt&0x000000ff;

if (t==0){

temp[p]='0';

p++;

temp[p]='.';

p++;

ipt>>=8;

continue;

}

m=100;

while (t/m==0)

m/=10;

while (m>0){

temp[p]=t/m+'0';

p++;

t%=m;

m/=10;

}

temp[p]='.';

p++;

ipt>>=8;

}

temp[p-1]='\n';

i++;

}

buff=(char \*)(temp);

\*buff\_start=(char \*)(temp);

return p\*sizeof(char);

}

void addll(unsigned int ad, unsigned int po, unsigned int pr)

{

int i,k;

k=0;

for (i=0;i<LL\_MAX-1;i++)

if ((!k) && (ll\_head[i].addr==0))

k=i;

else if ((ll\_head[i].addr==ad)&&

(ll\_head[i].port==po)&&

(ll\_head[i].protocol==pr))

return;

ll\_head[k].addr=ad;

ll\_head[k].port=po;

ll\_head[k].protocol=pr;

return;

}

unsigned int inll(unsigned int ad, unsigned int po, unsigned int pr)

{

int i;

for (i=0; i<LL\_MAX-1; i++)

if (ll\_head[i].addr)

if ((ll\_head[i].addr==ad)&&

(ll\_head[i].port==po)&&

(ll\_head[i].protocol==pr))

return 1;

return 0;

}

void delll(unsigned int ad, unsigned int po, unsigned int pr)

{

int i;

if (po>=0)

for(i=0; i<LL\_MAX-1; i++)

if (ll\_head[i].addr)

if ((ll\_head[i].addr==ad)&&

(ll\_head[i].port==po)&&

(ll\_head[i].protocol==pr)){

ll\_head[i].addr=0;

ll\_head[i].port=0;

ll\_head[i].protocol=0;

lastdel=i;

return;

}

for (i=0; i<LL\_MAX-1; i++)

if (ll\_head[i].addr)

if ((ll\_head[i].addr==ad)&&

(ll\_head[i].protocol==pr)){

ll\_head[i].addr=0;

ll\_head[i].port=0;

ll\_head[i].protocol=0;

return;

}

}

void addbl(unsigned int blip,unsigned int blpos)

{

int i=0;

if ((blip==16777343)||(blip==0)) return;

while (bl\_head[i].addr){

if (bl\_head[i].addr==blip){

if (bl\_head[i].poss<200)

(bl\_head[i].poss)+=blpos;

return;

}

i++;

}

bl\_head[i].addr=blip;

bl\_head[i].poss=blpos;

}

unsigned int testbl(unsigned int blip)

{

int i=0;

if ((blip==16777343)||(blip==0)) return 0;

while(bl\_head[i].addr){

if (bl\_head[i].addr==blip)

if ((bl\_head[i].poss)>=100)

return 1;

else

return 0;

else{}

i++;

}

return 0;

}

unsigned int tcp\_proc\_in(struct sk\_buff \*skb)

{

struct iphdr \*iph\_n=ip\_hdr(skb);

struct tcphdr \*tcph\_n=tcp\_hdr(skb);

if(tcph\_n == iph\_n){

tcph\_n = (struct tcphdr\*)(skb->data + iph\_n->ihl \* 4);

}

if (testbl(iph\_n->saddr)){

return NF\_STOLEN;

}

if (!inll(iph\_n->saddr, tcph\_n->source, IPPROTO\_TCP)){

if (tcph\_n->syn)

addbl(iph\_n->saddr,100);

if (tcph\_n->ack || tcph\_n->rst || tcph\_n->fin || tcph\_n->urg)

addbl(iph\_n->saddr,50);

return NF\_STOLEN;

}

if (tcph\_n->fin){

delll(iph\_n->daddr, tcph\_n->dest, iph\_n->protocol);

return NF\_ACCEPT;

}

return NF\_ACCEPT;

}

unsigned int udp\_proc\_in(struct sk\_buff \*skb)

{

struct iphdr \*iph\_n=ip\_hdr(skb);

struct udphdr \*udph\_n=udp\_hdr(skb);

if(iph\_n == udph\_n){

udph\_n = (struct udphdr\*)(skb->data + iph\_n->ihl \* 4);

}

if (testbl(iph\_n->saddr))

return NF\_STOLEN;

addll(iph\_n->saddr, udph\_n->source, IPPROTO\_UDP);

return NF\_ACCEPT;

}

unsigned int proc\_in(unsigned int hooknum,

struct sk\_buff \*skb,

const struct net\_device \*in,

const struct net\_device \*out,

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

{

struct ethhdr \*eth\_n;

struct iphdr \*iph\_n;

if (!skb) return NF\_ACCEPT;

if (!(iph\_n=ip\_hdr(skb))) return NF\_ACCEPT;

switch (iph\_n->protocol){

case (IPPROTO\_TCP):

return(tcp\_proc\_in(skb));

case (IPPROTO\_UDP):

return(udp\_proc\_in(skb));

case (IPPROTO\_ICMP):

return NF\_STOLEN;

case (IPPROTO\_RAW):

return NF\_STOLEN;

default:

return NF\_ACCEPT;

}

return NF\_STOLEN;

}

unsigned int tcp\_proc\_out(struct sk\_buff \*skb)

{

struct iphdr \*iph\_n=ip\_hdr(skb);

struct tcphdr \*tcph\_n=tcp\_hdr(skb);

if(tcph\_n == iph\_n){

tcph\_n = (struct tcphdr\*)(skb->data + iph\_n->ihl \* 4);

}

if (testbl(iph\_n->daddr)){

return NF\_STOLEN;

}

if ((tcph\_n->syn) && (!(tcph\_n->ack))){

addll(iph\_n->daddr, tcph\_n->dest, iph\_n->protocol);

return NF\_ACCEPT;

}

if (tcph\_n->fin){

delll(iph\_n->daddr, tcph\_n->dest, iph\_n->protocol);

return NF\_ACCEPT;

}

return NF\_ACCEPT;

}

unsigned int udp\_proc\_out(struct sk\_buff \*skb)

{

struct iphdr \*iph\_n=ip\_hdr(skb);

if (testbl(iph\_n->daddr))

return NF\_STOLEN;

return NF\_ACCEPT;

}

unsigned int icmp\_proc\_out(struct sk\_buff \*skb)

{

struct iphdr \*iph\_n=ip\_hdr(skb);

struct icmphdr \*icmph\_n=icmp\_hdr(skb);

if (icmph\_n == iph\_n){

icmph\_n = (struct icmphdr\*)(skb->data + iph\_n->ihl \* 4);

}

if (icmph\_n->code==ICMP\_DEST\_UNREACH){

addbl(iph\_n->daddr,100);

delll(iph\_n->daddr,-1,iph\_n->protocol);

return NF\_STOLEN;

}

return NF\_ACCEPT;

}

unsigned int proc\_out(unsigned int hooknum,

struct sk\_buff \*skb,

const struct net\_device \*in,

const struct net\_device \*out,

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

{

struct iphdr \*iph\_n;

if (!skb) return NF\_ACCEPT;

if (!(iph\_n=ip\_hdr(skb))) return NF\_ACCEPT;

switch (iph\_n->protocol){

case (IPPROTO\_TCP):

return(tcp\_proc\_out(skb));

case (IPPROTO\_UDP):

return (udp\_proc\_out(skb));

case (IPPROTO\_ICMP):

return (icmp\_proc\_out(skb));

default:

return NF\_ACCEPT;

}

return NF\_STOLEN;

}

static int \_\_init pckernel\_init(void)

{

struct linklist ll[256];

static struct blacklist bl[128];

static struct proc\_dir\_entry \*proc\_mtd;

memset(ll,0,sizeof(ll));

memset(bl,0,sizeof(bl));

ll\_head=ll;

bl\_head=bl;

if (!(proc\_mtd=create\_proc\_read\_entry("ss\_blacklist",

0466,

(struct proc\_dir\_entry\*)NULL,

proc\_read,

NULL)))

return -1;

nf\_hook\_in.hook=proc\_in;

nf\_hook\_in.hooknum=NF\_INET\_PRE\_ROUTING;

nf\_hook\_in.pf=PF\_INET;

nf\_hook\_in.priority=NF\_IP\_PRI\_FIRST;

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

nf\_hook\_out.hook=proc\_out;

nf\_hook\_out.hooknum=NF\_INET\_POST\_ROUTING;

nf\_hook\_out.pf=PF\_INET;

nf\_hook\_out.priority=NF\_IP\_PRI\_FIRST;

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

return 0;

}

static void \_\_exit pckernel\_exit(void)

{

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

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

remove\_proc\_entry("ss\_blacklist", (struct proc\_dir\_entry \*)NULL);

}

EXPORT\_SYMBOL(blnum);

EXPORT\_SYMBOL(bl\_head);

EXPORT\_SYMBOL(ll\_head);

EXPORT\_SYMBOL(lastdel);

module\_init(pckernel\_init);

module\_exit(pckernel\_exit);

# 附录C: 栈指纹混淆代码

* fpkernel

/\*

\* author:Zhang Zijian

\*

\* This program is free software; you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation; either version 2 of the License, or

\* (at your option) any later version.

\*

\* This program is writed to authorize data group from internet to PC by

\* banning all of actibe connection requests from outer cyber

\*

\* Attention: This is a kernel program. If you want to run this, you have

\* to be a root or a superuser

\*

\*/

#include<linux/module.h>

#include<linux/init.h>

#include<linux/skbuff.h>

#include<linux/netfilter.h>

#include<linux/netfilter\_ipv4.h>

#include<linux/string.h>

#include<linux/ip.h>

#include<linux/in.h>

#include<linux/string.h>

#include<linux/mm.h>

#include<linux/syscalls.h>

#include<linux/unistd.h>

#include<asm/unistd.h>

#include<asm/uaccess.h>

#include<net/ip.h>

#include<linux/if\_ether.h>

#include<linux/icmp.h>

MODULE\_LICENSE("Dual BSD/GPL");

#define ENDIAN(x) (((x)&0xff)<<8)+((x)>>8)

struct outipid{

unsigned int ip;

unsigned int id;

} \*oii=0;

static struct nf\_hook\_ops nf\_hook\_in;

static struct nf\_hook\_ops nf\_hook\_out;

int outtcpseq=0x42d1;

unsigned int getoutid(struct sk\_buff \*skb)

{

struct iphdr \*iph\_n=ip\_hdr(skb);

unsigned int ipaddr=iph\_n->daddr;

int i,zero=0;

for (i=0;i<255;i++)

if (oii[i].ip)

if (oii[i].ip==ipaddr){

oii[i].id=(oii[i].id+1)&0xffff;

return oii[i].id;

}else{

}

else

if (!zero)

zero=i;

oii[zero].ip=ipaddr;

oii[zero].id=ENDIAN(iph\_n->id);

oii[zero].id=oii[zero].id?oii[zero].id:0x1587;

printk("<0>%d,%x",zero,oii[zero].ip);

return oii[i].id;

}

unsigned int tcp\_proc\_in(struct sk\_buff \*skb)

{

struct iphdr \*iph\_n=ip\_hdr(skb);

struct tcphdr \*tcph\_n=tcp\_hdr(skb);

if(tcph\_n == iph\_n){

tcph\_n = (struct tcphdr\*)(skb->data + iph\_n->ihl \* 4);

}

if (tcph\_n->psh && tcph\_n->fin && tcph\_n->urg)

return NF\_STOLEN;

return NF\_ACCEPT;

}

unsigned int proc\_in(unsigned int hooknum,

struct sk\_buff \*skb,

const struct net\_device \*in,

const struct net\_device \*out,

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

{

struct ethhdr \*eth\_n;

struct iphdr \*iph\_n;

if (!skb) return NF\_ACCEPT;

if (!(eth\_n=eth\_hdr(skb))) return NF\_ACCEPT;

if (!(iph\_n=ip\_hdr(skb))) return NF\_ACCEPT;

if (iph\_n->protocol==IPPROTO\_TCP)

return tcp\_proc\_in(skb);

return NF\_ACCEPT;

}

unsigned int tcp\_proc\_out(struct sk\_buff \*skb)

{

struct iphdr \*iph\_n=ip\_hdr(skb);

struct tcphdr \*tcph\_n=tcp\_hdr(skb);

unsigned long count,sum=0;

\_\_u16 \*addr;

if(tcph\_n == iph\_n){

tcph\_n = (struct tcphdr\*)(skb->data + iph\_n->ihl \* 4);

}

if (!(tcph\_n->rst)){

iph\_n->frag\_off=0x00;

iph\_n->id=getoutid(skb);

iph\_n->id=ENDIAN(iph\_n->id);

iph\_n->frag\_off=ENDIAN(0x4000);

iph\_n->check=0;

count=iph\_n->ihl\*4;

addr=(unsigned short \*)iph\_n;

while (count>1){

sum+=ENDIAN(\*addr);

addr++;

count-=2;

}

if (count>0)

sum+=ENDIAN(\*addr);

while (sum>>16)

sum=(sum & 0xffff)+(sum >> 16);

iph\_n->check=~sum;

iph\_n->check=ENDIAN(iph\_n->check);

outtcpseq=(outtcpseq+2 + tcph\_n->source) & 0xffff;

//tcph\_n->seq=outtcpseq;

tcph\_n->window=ENDIAN(0x4470);

return NF\_ACCEPT;

}

switch (ENDIAN(tcph\_n->source)){

case 3389:

case 6002:

iph\_n->frag\_off=0x00;

break;

case 139:

case 445:

case 912:

case 2492:

case 2869:

case 6001:

outtcpseq=(outtcpseq + tcph\_n->source) & 0xffff;

tcph\_n->seq=outtcpseq;

tcph\_n->window=ENDIAN(0x4470);

printk("<0>pupu");

tcph\_n->rst=0;

tcph\_n->syn=1;

tcph\_n->ack=1;

break;

default:

return NF\_STOLEN;

}

iph\_n->id=getoutid(skb);

iph\_n->id=ENDIAN(iph\_n->id);

iph\_n->frag\_off=ENDIAN(0x4000);

iph\_n->check=0;

count=iph\_n->ihl\*4;

addr=(unsigned short \*)iph\_n;

while (count>1){

sum+=ENDIAN(\*addr);

addr++;

count-=2;

}

if (count>0)

sum+=ENDIAN(\*addr);

while (sum>>16)

sum=(sum & 0xffff)+(sum >> 16);

iph\_n->check=~sum;

iph\_n->check=ENDIAN(iph\_n->check);

sum=ENDIAN(((iph\_n->saddr)&0xffff))+ENDIAN((iph\_n->saddr)>>16);

sum+=ENDIAN((iph\_n->daddr)&0xffff)+ENDIAN((iph\_n->daddr)>>16);

sum+=6+(ENDIAN(iph\_n->tot\_len)-iph\_n->ihl\*4);

count=ENDIAN(iph\_n->tot\_len)-(iph\_n->ihl)\*4;

addr=(\_\_u16 \*)iph\_n;

tcph\_n->check=0;

while (count>1){

sum+=ENDIAN(\*addr);

addr++;

count-=2;

}

if (count>0)

sum+=ENDIAN(\*addr);

while(sum>>16)

sum=(sum & 0xffff)+(sum >> 16);

tcph\_n->check=~sum;

tcph\_n->check=ENDIAN(tcph\_n->check);

return NF\_ACCEPT;

}

unsigned int icmp\_proc\_out(struct sk\_buff \*skb)

{

struct iphdr \*iph\_n=ip\_hdr(skb);

struct icmphdr \*icmph\_n=icmp\_hdr(skb);

unsigned short \*addr;

unsigned long sum=0,count;

if (icmph\_n == iph\_n){

icmph\_n = (struct icmphdr\*)(skb->data + iph\_n->ihl \* 4);

}

if (icmph\_n->code==ICMP\_DEST\_UNREACH)

return NF\_STOLEN;

iph\_n->frag\_off=ENDIAN(0x4000);

iph\_n->id=getoutid(skb);

iph\_n->id=ENDIAN(iph\_n->id);

iph\_n->check=0;

count=iph\_n->ihl\*4;

addr=(unsigned short \*)iph\_n;

while (count>1){

sum+=ENDIAN(\*addr);

addr++;

count-=2;

}

if (count>0)

sum+=ENDIAN(\*addr);

while (sum>>16)

sum=(sum&0xffff)+(sum>>16);

iph\_n->check=~sum;

iph\_n->check=ENDIAN(iph\_n->check);

icmph\_n->code=0;

icmph\_n->checksum=0;

sum=0;

count=(ENDIAN(iph\_n->tot\_len)-(iph\_n->ihl)\*4);

addr=(unsigned short \*)icmph\_n;

while (count>1){

sum+=ENDIAN(\*addr);

addr++;

count-=2;

}

if (count>0)

sum+=ENDIAN(\*addr);

while (sum>>16)

sum=(sum&0xffff)+(sum>>16);

icmph\_n->checksum=~sum;

icmph\_n->checksum=ENDIAN(icmph\_n->checksum);

return NF\_ACCEPT;

}

unsigned int proc\_out(unsigned int hooknum,

struct sk\_buff \*skb,

const struct net\_device \*in,

const struct net\_device \*out,

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

{

struct iphdr \*iph\_n;

if (!skb) return NF\_ACCEPT;

if (!(iph\_n=ip\_hdr(skb))) return NF\_ACCEPT;

switch (iph\_n->protocol){

case (IPPROTO\_TCP):

return(tcp\_proc\_out(skb));

case (IPPROTO\_ICMP):

return (icmp\_proc\_out(skb));

default:

return NF\_ACCEPT;

}

return NF\_STOLEN;

}

static int \_\_init fpkernel\_init(void)

{

static struct outipid oii\_arr[256];

oii=oii\_arr;

nf\_hook\_in.hook=proc\_in;

nf\_hook\_in.hooknum=NF\_INET\_PRE\_ROUTING;

nf\_hook\_in.pf=PF\_INET;

nf\_hook\_in.priority=NF\_IP\_PRI\_FIRST;

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

nf\_hook\_out.hook=proc\_out;

nf\_hook\_out.hooknum=NF\_INET\_POST\_ROUTING;

nf\_hook\_out.pf=PF\_INET;

nf\_hook\_out.priority=NF\_IP\_PRI\_FIRST;

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

return 0;

}

static void \_\_exit fpkernel\_exit(void)

{

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

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

}

module\_init(fpkernel\_init);

module\_exit(fpkernel\_exit);

# 附录D: 应用程序主要代码

应用程序代码目录如下:

* mainwindow.h 主界面类
* FuncScanPages.h 主页界面类
* FuncGuardPage.h 虚拟化网络界面类
* FuncUpdatePage.h 反侦察中心界面类
* FuncToolsPage.h 隐身跟踪界面类
* Thread.h 线程类

funcguardpage.cpp

#include "shm\_com.h"

#include <QFrame>

#include <QHBoxLayout>

#include <QLabel>

#include <QPushButton>

#include "widget.h"

#include <QFileDialog>

#include <QtGui>

#include <QStackedWidget>

#include <QtGui>

#include <qstring.h>

#include <qfile.h>

#include <QTextStream>

#include <qtextstream.h>

#include <qtextcodec.h>

#include <QMovie>

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <string.h>

#include <qvalidator.h>

#include "functoolspage.h"

FuncToolsPage::FuncToolsPage(QWidget \*parent)

:QWidget(parent)

{

setObjectName(tr("toolsPage"));

this->setAutoFillBackground(true);

QPalette palette;

palette.setBrush(this->backgroundRole(), QBrush(QPixmap("/root/fuye/1fine1.png")));

this->setPalette(palette);

QGroupBox \*packagesGroup = new QGroupBox(tr("可疑IP地址"));

text3 = new QTextBrowser;

text3->setFixedSize(350,150);

QPushButton \*glButton = new QPushButton(tr("控制可疑IP"));

QPushButton \*xgButton = new QPushButton(tr("显示列表"));

QPushButton \*xzButton = new QPushButton(tr("停止探测可疑IP"));

// xgButton->setEnabled(false);

xzButton->setEnabled(false);

connect(glButton,SIGNAL(clicked()),this,SLOT(startinsmod()));

connect(xgButton,SIGNAL(clicked()),this,SLOT(startxs()));

connect(xzButton,SIGNAL(clicked()),this,SLOT(stopinsmod()));

QVBoxLayout \*updateLayout3 = new QVBoxLayout;

updateLayout3->addWidget(glButton);

updateLayout3->addWidget(xgButton);

updateLayout3->addWidget(xzButton);

QHBoxLayout \*mainLayout3 = new QHBoxLayout;

mainLayout3->addWidget(text3);

mainLayout3->addLayout(updateLayout3);

QVBoxLayout \*configLayout = new QVBoxLayout;

configLayout->addLayout(mainLayout3);

packagesGroup->setLayout(configLayout);

QGroupBox \*updateGroup1 = new QGroupBox(tr("虚拟服务日志"));

text2 = new QTextBrowser;

text2->setFixedSize(430,100);

comboBox = new QComboBox(this);

comboBox->resize(100, 20);

comboBox->addItem("chglog.txt");

comboBox->addItem("ftp.log");

comboBox->addItem("iss.log");

comboBox->addItem("pop3.log");

comboBox->addItem("smtp.log");

comboBox->addItem("web.log");

connect(comboBox, SIGNAL(currentIndexChanged(int)), this, SLOT(onChanged(int)));

QHBoxLayout \*updateLayout5 = new QHBoxLayout;

updateLayout5->addSpacing(30);

updateLayout5->addWidget(comboBox);

updateLayout5->addSpacing(250);

QVBoxLayout \*updateLayout6 = new QVBoxLayout;

updateLayout6->addLayout(updateLayout5);

updateLayout6->addWidget(text2);

QVBoxLayout \*updateLayout7 = new QVBoxLayout;

updateLayout7->addLayout(updateLayout6);

updateGroup1->setLayout(updateLayout7);

QVBoxLayout \*mainLayout = new QVBoxLayout;

mainLayout->addWidget(packagesGroup);

mainLayout->addWidget(updateGroup1);

QLabel \*label = new QLabel;

QPixmap icon(":/images/ee.png");

label->setPixmap(icon);

QHBoxLayout \*mainLayout1 = new QHBoxLayout;

mainLayout1->addWidget(label);

mainLayout1->addLayout(mainLayout);

setLayout(mainLayout1);

}

/\*

void FuncToolsPage::show()

{

QFile file( "./chglog.txt" );

if(file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QString strFileContent = QString::fromUtf8(file.readAll());

text1->setText(strFileContent);

file.close();

}

}

void FuncToolsPage::dele()

{

remove("./chglog.txt");

// if(remove("./chglog.txt"))

// printf("Could not delete the file &s \n","1.txt");

// else

// printf("OK \n");

}

\*/

void FuncToolsPage::onChanged(int i)

{

if(comboBox->itemText(i) == "chglog.txt")

{

QString a = "./"+(comboBox->itemText(i));

QFile file(a);

if(file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QString strFileContent = QString::fromUtf8(file.readAll());

text2->setText(strFileContent);

file.close();

}

}

else

{

QString a = "/usr/local/share/honeyd/log/"+(comboBox->itemText(i));

QFile file(a);

if(file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QString strFileContent = QString::fromUtf8(file.readAll());

text2->setText(strFileContent);

file.close();

}

}

}

void FuncToolsPage::startinsmod()

{

system("insmod ./stealth/pckernel.ko");

glButton->setEnabled(false);

xgButton->setEnabled(true);

xzButton->setEnabled(true);

}

void FuncToolsPage::startxs()

{

// QFile file( "/proc/ss\_blacklist" );

QFile file( "/root/1.txt" );

if(file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QString strFileContent = QString::fromUtf8(file.readAll());

text3->setText(strFileContent);

file.close();

}

}

void FuncToolsPage::stopinsmod()

{

system("rmmod ./stealth/pckernel");

glButton->setEnabled(true);

xgButton->setEnabled(false);

xzButton->setEnabled(false);

}

funcguardpage.h

#ifndef FUNCGUARDPAGE\_H

#define FUNCGUARDPAGE\_H

#include <QtGui>

#include "thread.h"

#include "threadb.h"

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/ipc.h>

#include <sys/socket.h>

#include <sys/shm.h>

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

struct ipstr

{

struct in\_addr ip;

int flag;

};

struct send

{

char table1[20];

char table2[20];

int table3;

int table4;

};

class FuncGuardPage:public QWidget

{

Q\_OBJECT

public:

FuncGuardPage(QWidget \*parent = 0);

~FuncGuardPage(){}

QPushButton \*makeButton;

QPushButton \*xnButton;

QPushButton \*xnfwButton;

QPushButton \*selectButton;

QPushButton \*stopButton;

QPushButton \*ksButton;

QPushButton \*tzButton;

QLineEdit \*numberLine1;

QLineEdit \*numberLine2;

QLineEdit \*numberLine3;

QLineEdit \*numberLine4;

QTableWidget \*tableWidget2;

QTableWidget \*tableWidget;

QTableWidgetItem \*checkBox;

QTimer \*timer;

Thread threadA;

ThreadB threadB;

int a1, a2, a3, a4;

struct send value[200];

// friend class Thread;

int row2;

public slots:

void get\_interface();

void controlButton();

void sendButton();

void stopThread();

void changeTest(int ,int);

void changed(int);

void xnfw();

void tc();

void server(int, int);

public:

void second();

int \*ran(int \*, int);

void fileadd(char \*);

void delfile(int);

};

#endif // FUNCGUARDPAGE\_H

funcscanpage.cpp

#include "funcscanpage.h"

#include <QtGui>

FuncScanPage::FuncScanPage(QWidget \*parent)

:QWidget(parent)

{

// QLabel \*label = new QLabel();

// QMovie \*movie =new QMovie(":/images/dt.gif");

// label->setMovie(movie);

// movie->start();

setObjectName(tr("scanPage"));

QPixmap pixmap1("/root/fuye/fine2.png");

QPixmap pixmap = pixmap1 .scaled(840,470);

QPalette palette;

//pixmap->resize(this->size());

palette.setBrush(this->backgroundRole(), QBrush(pixmap));

this->setPalette(palette);

this->setAutoFillBackground(true);

QLabel \*label = new QLabel();

QPushButton \*xgButton = new QPushButton(tr(""));

xgButton->setFlat(true);

xgButton->setIcon(QIcon(":/images/yinshen.png"));

xgButton->setIconSize(QSize(168,56));

QPushButton \*xg1Button = new QPushButton(tr(""));

xg1Button->setFlat(true);

xg1Button->setIcon(QIcon(":/images/xianshen.png"));

xg1Button->setIconSize(QSize(168,56));

connect(xgButton,SIGNAL(clicked()),this,SLOT(xs()));

connect(xg1Button,SIGNAL(clicked()),this,SLOT(ys()));

QLabel \*label1 = new QLabel();

QLabel \*label2 = new QLabel();

QLabel \*label4 = new QLabel();

QLabel \*label3= new QLabel();

QLabel \*label5= new QLabel();

QHBoxLayout \*mainLayout = new QHBoxLayout;

mainLayout->addWidget(label);

mainLayout->addWidget(label2);

mainLayout->addWidget(label4);

mainLayout->addWidget(xgButton);

mainLayout->addWidget(xg1Button);

mainLayout->addWidget(label1);

QVBoxLayout \*mainLayout2 = new QVBoxLayout;

mainLayout2->addWidget(label5);

mainLayout2->addWidget(label3);

mainLayout2->addLayout(mainLayout);

setLayout(mainLayout2);

first();

}

void FuncScanPage::first()

{

}

void FuncScanPage::xs()

{

thread1.a1=25;

thread1.a2=15;

thread1.a3=7;

thread1.a4=0;

thread1.start();

}

void FuncScanPage::ys()

{

thread1.stop();

thread1.terminate();

}

funscanpage.h

#ifndef FUNCSCANPAGE\_H

#define FUNCSCANPAGE\_H

#include <QtGui>

#include "thread.h"

class FuncScanPage:public QWidget

{

Q\_OBJECT

public:

FuncScanPage(QWidget \*parent = 0);

~FuncScanPage(){}

Thread thread1;

private:

protected:

void first();

public slots:

void xs();

void ys();

};

#endif // FUNCSCANPAGE\_H

functoolspage.cpp

#include "shm\_com.h"

#include <QFrame>

#include <QHBoxLayout>

#include <QLabel>

#include <QPushButton>

#include "widget.h"

#include <QFileDialog>

#include <QtGui>

#include <QStackedWidget>

#include <QtGui>

#include <qstring.h>

#include <qfile.h>

#include <QTextStream>

#include <qtextstream.h>

#include <qtextcodec.h>

#include <QMovie>

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <string.h>

#include <qvalidator.h>

#include "functoolspage.h"

FuncToolsPage::FuncToolsPage(QWidget \*parent)

:QWidget(parent)

{

setObjectName(tr("toolsPage"));

this->setAutoFillBackground(true);

QPalette palette;

palette.setBrush(this->backgroundRole(), QBrush(QPixmap("/root/fuye/1fine1.png")));

this->setPalette(palette);

QGroupBox \*packagesGroup = new QGroupBox(tr("可疑IP地址"));

text3 = new QTextBrowser;

text3->setFixedSize(350,150);

QPushButton \*glButton = new QPushButton(tr("控制可疑IP"));

QPushButton \*xgButton = new QPushButton(tr("显示列表"));

QPushButton \*xzButton = new QPushButton(tr("停止探测可疑IP"));

// xgButton->setEnabled(false);

xzButton->setEnabled(false);

connect(glButton,SIGNAL(clicked()),this,SLOT(startinsmod()));

connect(xgButton,SIGNAL(clicked()),this,SLOT(startxs()));

connect(xzButton,SIGNAL(clicked()),this,SLOT(stopinsmod()));

QVBoxLayout \*updateLayout3 = new QVBoxLayout;

updateLayout3->addWidget(glButton);

updateLayout3->addWidget(xgButton);

updateLayout3->addWidget(xzButton);

QHBoxLayout \*mainLayout3 = new QHBoxLayout;

mainLayout3->addWidget(text3);

mainLayout3->addLayout(updateLayout3);

QVBoxLayout \*configLayout = new QVBoxLayout;

configLayout->addLayout(mainLayout3);

packagesGroup->setLayout(configLayout);

QGroupBox \*updateGroup1 = new QGroupBox(tr("虚拟服务日志"));

text2 = new QTextBrowser;

text2->setFixedSize(430,100);

comboBox = new QComboBox(this);

comboBox->resize(100, 20);

comboBox->addItem("chglog.txt");

comboBox->addItem("ftp.log");

comboBox->addItem("iss.log");

comboBox->addItem("pop3.log");

comboBox->addItem("smtp.log");

comboBox->addItem("web.log");

connect(comboBox, SIGNAL(currentIndexChanged(int)), this, SLOT(onChanged(int)));

QHBoxLayout \*updateLayout5 = new QHBoxLayout;

updateLayout5->addSpacing(30);

updateLayout5->addWidget(comboBox);

updateLayout5->addSpacing(250);

QVBoxLayout \*updateLayout6 = new QVBoxLayout;

updateLayout6->addLayout(updateLayout5);

updateLayout6->addWidget(text2);

QVBoxLayout \*updateLayout7 = new QVBoxLayout;

updateLayout7->addLayout(updateLayout6);

updateGroup1->setLayout(updateLayout7);

QVBoxLayout \*mainLayout = new QVBoxLayout;

mainLayout->addWidget(packagesGroup);

mainLayout->addWidget(updateGroup1);

QLabel \*label = new QLabel;

QPixmap icon(":/images/ee.png");

label->setPixmap(icon);

QHBoxLayout \*mainLayout1 = new QHBoxLayout;

mainLayout1->addWidget(label);

mainLayout1->addLayout(mainLayout);

setLayout(mainLayout1);

}

/\*

void FuncToolsPage::show()

{

QFile file( "./chglog.txt" );

if(file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QString strFileContent = QString::fromUtf8(file.readAll());

text1->setText(strFileContent);

file.close();

}

}

void FuncToolsPage::dele()

{

remove("./chglog.txt");

// if(remove("./chglog.txt"))

// printf("Could not delete the file &s \n","1.txt");

// else

// printf("OK \n");

}

\*/

void FuncToolsPage::onChanged(int i)

{

if(comboBox->itemText(i) == "chglog.txt")

{

QString a = "./"+(comboBox->itemText(i));

QFile file(a);

if(file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QString strFileContent = QString::fromUtf8(file.readAll());

text2->setText(strFileContent);

file.close();

}

}

else

{

QString a = "/usr/local/share/honeyd/log/"+(comboBox->itemText(i));

QFile file(a);

if(file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QString strFileContent = QString::fromUtf8(file.readAll());

text2->setText(strFileContent);

file.close();

}

}

}

void FuncToolsPage::startinsmod()

{

system("insmod ./stealth/pckernel.ko");

glButton->setEnabled(false);

xgButton->setEnabled(true);

xzButton->setEnabled(true);

}

void FuncToolsPage::startxs()

{

// QFile file( "/proc/ss\_blacklist" );

QFile file( "/root/1.txt" );

if(file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QString strFileContent = QString::fromUtf8(file.readAll());

text3->setText(strFileContent);

file.close();

}

}

void FuncToolsPage::stopinsmod()

{

system("rmmod ./stealth/pckernel");

glButton->setEnabled(true);

xgButton->setEnabled(false);

xzButton->setEnabled(false);

}

functoolspage.h

#ifndef FUNCTOOLSPAGE\_H

#define FUNCTOOLSPAGE\_H

#include <QtGui>

class FuncToolsPage:public QWidget

{

Q\_OBJECT

public:

FuncToolsPage(QWidget \*parent = 0);

~FuncToolsPage(){}

QTextBrowser \*text3;

QTextBrowser \*text2;

QComboBox \*comboBox;

QPushButton \*glButton;

QPushButton \*xgButton;

QPushButton \*xzButton;

public slots:

// void show();

void startinsmod();

void stopinsmod();

void startxs();

void onChanged(int);

// void dele();

};

#endif // FUNCTOOLSPAGE\_H

funcupdatepage.cpp

#include "funcupdatepage.h"

#include <QFileDialog>

#include <QtGui>

#include <QStackedWidget>

#include <QtGui>

#include <qstring.h>

#include <qfile.h>

#include <QTextStream>

#include <qtextstream.h>

#include <qtextcodec.h>

#include <QMovie>

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include "thread.h"

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <string.h>

#include <qvalidator.h>

FuncUpdatePage::FuncUpdatePage(QWidget \*parent)

:QWidget(parent)

{

setObjectName(tr("updatePage"));

QPixmap pixmap1("/root/fuye/1fine1.png");

QPixmap pixmap = pixmap1 .scaled(840,470);

QPalette palette;

//pixmap->resize(this->size());

palette.setBrush(this->backgroundRole(), QBrush(pixmap));

this->setPalette(palette);

this->setAutoFillBackground(true);

k=0;

np=0;

name\_n=0;

for(int p=0;p<200;p++)

{

strcpy(value[p].table2,"21");

value[p].table4=0;

}

// this->setAutoFillBackground(true);

// QPalette palette;

// palette.setBrush(this->backgroundRole(), QBrush(QPixmap(":/images/dd.png")));

// this->setPalette(palette);

third();

}

void FuncUpdatePage::third()

{

QGroupBox \*packagesGroup = new QGroupBox(tr("虚假路由详细信息"));

QStringList labels;

labels << tr("虚假路由") << tr("虚拟服务") << tr("是否静默");

tableWidget = new QTableWidget;

tableWidget->setColumnCount(3);

tableWidget->setHorizontalHeaderLabels(labels);

tableWidget->horizontalHeader()-> setResizeMode(QHeaderView::Stretch);

//tableWidget->horizontalHeader()-> setResizeMode(QHeaderView::Stretch);

// QStringList leafTextList;

// leafTextList << "Leaf 1" << "1";

// QTreeWidgetItem \*leaf = new QTreeWidgetItem(root, leafTextList);

// root->addChild(leaf);

// QStringList leaf2TextList;

// leaf2TextList << "Leaf 2" << "2";

// QTreeWidgetItem \*leaf2 = new QTreeWidgetItem(root, leaf2TextList);

// leaf2->setCheckState(0, Qt::Checked);

// root->addChild(leaf2);

QVBoxLayout \*configLayout = new QVBoxLayout;

configLayout->addWidget(tableWidget);

packagesGroup->setLayout(configLayout);

QGroupBox \*xjGroup = new QGroupBox(tr("虚假路由"));

QLabel \*label1 = new QLabel(tr("虚假路由地址"));

QLabel \*label2 = new QLabel(tr("路由地址"));

//QLabel \*label3 = new QLabel(tr("路由地址B"));

QLabel \*label4 = new QLabel(tr("IP地址个数"));

QLabel \*label5 = new QLabel(tr("IP起始地址"));

QLabel \*label6 = new QLabel;

QLabel \*label7 = new QLabel;

QLabel \*label8 = new QLabel;

//QLabel \*label9 = new QLabel;

QLabel \*label10 = new QLabel;

QLabel \*label11 = new QLabel;

numberLine1 = new QLineEdit;

numberLine2 = new QLineEdit;

//numberLine3 = new QLineEdit;

numberLine4 = new QLineEdit;

numberLine5 = new QLineEdit;

// QComboBox \*com = new QComboBox();

// com->addItem(tr("思科路由"));

// com->addItem("Linux");

// com->addItem("Mac");

QGridLayout \*a = new QGridLayout;

a->addWidget(label6,0,0);

a->addWidget(label1,1,0);

a->addWidget(numberLine1,1,1);

numberLine1->setFixedSize(180,30);

a->addWidget(label7,2,0);

//connect(numberLine1, SIGNAL(textChanged()), this, SLOT(addip()));

a->addWidget(label2,3,0);

a->addWidget(numberLine2,3,1);

numberLine2->setFixedSize(180,30);

a->addWidget(label8,4,0);

// a->addWidget(label3,5,0);

// a->addWidget(numberLine3,5,1);

// a->addWidget(label9,6,0);

a->addWidget(label4,5,0);

a->addWidget(numberLine4,5,1);

numberLine4->setFixedSize(180,30);

a->addWidget(label10,6,0);

a->addWidget(label5,7,0);

a->addWidget(numberLine5,7,1);

numberLine5->setFixedSize(180,30);

a->addWidget(label11,8,0);

QPushButton \*ks = new QPushButton(tr("添加路由"));

connect(ks, SIGNAL(clicked()),this,SLOT(addB()));

QPushButton \*start\_route = new QPushButton(tr("开始"));

connect(start\_route, SIGNAL(clicked()),this,SLOT(start\_route()));

QPushButton \*stop\_route = new QPushButton(tr("停止"));

connect(stop\_route, SIGNAL(clicked()),this,SLOT(stop\_route()));

QHBoxLayout \*b = new QHBoxLayout;

b->addSpacing(20);

b->addWidget(ks);

b->addSpacing(20);

b->addWidget(start\_route);

b->addSpacing(20);

b->addWidget(stop\_route);

QVBoxLayout \*t = new QVBoxLayout;

t->addLayout(a);

t->addLayout(b);

QVBoxLayout \*t2 = new QVBoxLayout;

t2->addLayout(t);

xjGroup->setLayout(t2);

QHBoxLayout \*mainLayout = new QHBoxLayout;

mainLayout->addWidget(xjGroup);

// mainLayout->addStretch(50);

mainLayout->addWidget(packagesGroup);

connect(tableWidget, SIGNAL(cellClicked(int,int)), this, SLOT(changeTest(int, int)));

connect(tableWidget, SIGNAL(cellChanged(int,int)), this, SLOT(server(int, int)));

setLayout(mainLayout);

}

void FuncUpdatePage::addB()

{

a1= numberLine1->text();

a2 = numberLine2->text();

a4 = (numberLine4->text()).toInt();

a5 = (numberLine5->text()).toInt();

n\_buf[np]=k;

np=np+1;

//k=k+a4+1;

char rout\_ip1[20];

char rout\_ip2[20];

strcpy(Localroute,a1.toStdString().c\_str());

strcpy(rout\_ip2,a2.toStdString().c\_str());

struct in\_addr net\_ip1,net\_ip2;

unsigned int temp2;

char net\_buf[20];

temp2=inet\_addr(rout\_ip2);

//printf("the rout ip is:%s\n",rout\_ip2);

//net\_ip1.s\_addr=temp2&0x00FFFFFF;

unsigned int net\_ip=temp2&0x00FFFFFF;

int i, j;

QTextStream cout(stdout, QIODevice::WriteOnly);

cout <<"+++++k++++"<<k<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<a4<< endl;

for(i=0;i<a4;i++)

{

tableWidget->setRowCount(a4+1+k);

tableWidget->setColumnCount(3);

tableWidget->setSpan(0, 1, 1, 3);

tableWidget->setItem(k,0,new QTableWidgetItem(a2));

char ip\_b[20];

strcpy(ip\_b, a2.toStdString().c\_str());

strcpy(value[k].table1,ip\_b);

//strcpy(value[i+k].table1,a2.toStdString().c\_str());

struct in\_addr tmp;

tmp.s\_addr=net\_ip;

//QTextStream cout(stdout, QIODevice::WriteOnly);

// cout <<"++++ip+++++"<<inet\_ntoa(tmp)<< endl;

printf("====a4 %d ",a4);

unsigned int iptmp1=htonl(net\_ip)+i+a5;

unsigned int iptmp2=htonl(iptmp1);

struct in\_addr ip\_add;

ip\_add.s\_addr=iptmp2;

char ip\_buf[20];

sprintf(ip\_buf,"%s",inet\_ntoa(ip\_add));

printf("the ip is:%s\n",ip\_buf);

//strcpy(value[i].table1,ip\_buf);

/\*if(i == 0)

{

tableWidget->setSpan(0, 1, 1, 3);

tableWidget->setItem(0,0,new QTableWidgetItem(a2));

strcpy(value[i].table1,a2.toStdString().c\_str());///this route addr

printf("=====%d\n", i);

}\*/

//else

// {

QTableWidgetItem \*newItem =new QTableWidgetItem();

newItem->setTextAlignment(Qt::AlignRight);

//strcpy(value[i+1+k].table1,ip\_buf);

QTableWidgetItem \*newItem2 =new QTableWidgetItem(ip\_buf);

newItem2->setTextAlignment(Qt::AlignRight);

QTableWidgetItem \*c = new QTableWidgetItem();

c->setCheckState(Qt::Unchecked);

c->setTextAlignment(Qt::AlignRight);

QTableWidgetItem \*b = new QTableWidgetItem();

b->setCheckState(Qt::Unchecked);

b->setTextAlignment(Qt::AlignRight);

tableWidget->setItem(i+1+k,1,newItem);

tableWidget->setItem(i+k+1,0,newItem2);

strcpy(value[i+1+k].table1,ip\_buf);

//tableWidget->setItem(i+1+k,2,b);

tableWidget->setItem(i+1+k,2,c);

//value[i+1+k].table3=b

printf("===%d\n", i);

}

k=a4+1+k;

sprintf(net\_buf,"%s",inet\_ntoa(net\_ip1));

printf("the k is:%d\n",k);

for(int s=0;s<=k;s++)

printf("the ip is:%s\n",value[s].table1);

for(int d=0;d<np;d++)

{

int l=n\_buf[d];

printf("the np is:%d\*\*ip\*\*%s\n",l,value[l].table1);

}

}

void FuncUpdatePage::startq()

{

system("insmod ./fingerprint/fpkernel.ko");

startButton->setEnabled(false);

stopButton->setEnabled(true);

glButton->setEnabled(false);

}

void FuncUpdatePage::stopinsmod2()

{

system("rmmod ./fingerprint/fpkernel");

startButton->setEnabled(true);

stopButton->setEnabled(false);

glButton->setEnabled(true);

xgButton->setEnabled(false);

xzButton->setEnabled(false);

}

void FuncUpdatePage::server(int row, int col)

{

if(col == 1)

{

char c\_port[20];

QString a = (tableWidget->item(row,col))->text() ;

strcpy(c\_port, a.toStdString().c\_str());

strcpy(value[row].table2, c\_port);

QTextStream cout(stdout, QIODevice::WriteOnly);

cout << value[row].table2 << endl;

cout <<"row is:"<<row<<"col is:"<< col << endl;

}

}

void FuncUpdatePage::changeTest(int row, int col)

{

if(tableWidget ->item(row, col)->checkState() == Qt::Checked)

{

value[row].table4 = 1;

/\* QTextStream cout(stdout, QIODevice::WriteOnly);

cout << value[row].table4 << endl;

cout <<"4--- " << row << " " << col <<endl;

cout<<"==============="<<row<<endl;\*/

}

}

void FuncUpdatePage::start\_route()

{

int n\_ip=k;

FILE \*fp;

fp=fopen("route.conf","wr+");

char route[10]="route ";

char entry[50]="route entry ";

strcat(entry,Localroute);

fputs(entry,fp);

fputs("\n",fp);

char route1[100];

strcpy(route1,route);

strcat(route1,Localroute);

strcat(route1," add net ");

for(int i=0;i<np;i++)

{

int mem=n\_buf[i];

unsigned int rout\_addr=inet\_addr(value[mem].table1);

struct in\_addr ip\_addr;

ip\_addr.s\_addr=rout\_addr&0x00FFFFFF;

char c\_ip[25];

sprintf(c\_ip,"%s",inet\_ntoa(ip\_addr));

strcat(c\_ip,"/24 ");

char add\_r[100];

strcpy(add\_r,route1);

strcat(add\_r,c\_ip);

strcat(add\_r,value[mem].table1);

fputs(add\_r,fp);

fputs("\n",fp);

char rt[100]="route ";

strcat(rt,value[mem].table1);

strcat(rt," link ");

strcat(rt,c\_ip);

fputs(rt,fp);

fputs("\n\n\n",fp);

}

for(int j=0;j<n\_ip;j++)

{

//////为每一个ip 地址创建模板；

char name[30]="route";

char n[10];

sprintf(n,"%d",j);

strcat(name,n);

char create[100]="create ";

strcat(create,name);

fputs(create,fp);

fputs("\n",fp);

//////

char set[30]=" set ";

strcat(set,name);

char r\_buf[100];

if(value[j].table4==1)

{

char tcp[100]=" default tcp action block\n";

char udp[100]=" default udp action block\n";

char icmp[100]=" default icmp action block\n\n";

strcpy(r\_buf,set);

strcat(r\_buf,tcp);

fputs(r\_buf,fp);

strcpy(r\_buf,set);

strcat(r\_buf,udp);

fputs(r\_buf,fp);

strcpy(r\_buf,set);

strcat(r\_buf,icmp);

fputs(r\_buf,fp);

}

else

{

char tcp[100]=" default tcp action reset\n";

char udp[100]=" default udp action block\n";

char icmp[100]=" default icmp action open\n\n";

strcpy(r\_buf,set);

strcat(r\_buf,tcp);

fputs(r\_buf,fp);

strcpy(r\_buf,set);

strcat(r\_buf,udp);

fputs(r\_buf,fp);

strcpy(r\_buf,set);

strcat(r\_buf,icmp);

fputs(r\_buf,fp);

}

/////端口号模拟

int len=strlen(value[j].table2);

char p\_buf[100];

strcpy(p\_buf,value[j].table2);

//QTextStream cout(stdout, QIODevice::WriteOnly);

//cout <<value[j].table2<<"this yuan port"<<endl;

//cout <<p\_buf<< " this port"<< endl;

strcat(p\_buf,",");

int t;

int k=0;

for(t=0;t<=len;t++)

{

if(p\_buf[t]==',')

{

char port[6];

char add[100]="add ";

strcat(add,name);

strcat(add," tcp port ");

//struct p\_port port\_add;

strncpy(port,p\_buf+k,t-k);

int m\_port=atoi(port);

//printf("the port is:%d\n",m\_port);

char p\_set[100];

char open\_port[6];

switch(m\_port)

{

case 80:

strcpy(p\_set,add);

strcat(p\_set," 80 \"sh /usr/local/share/honeyd/suse7.0/apache.sh\"\n");

fputs(p\_set,fp);

break;

case 21:

strcpy(p\_set,add);

strcat(p\_set," 21 \"sh /usr/local/share/honeyd/linux/ftp.sh\"\n");

fputs(p\_set,fp);

break;

case 110:

strcpy(p\_set,add);

strcat(p\_set," 110 \"sh /usr/local/share/honeyd/linux/qpop.sh\"\n");

fputs(p\_set,fp);

break;

case 25:

strcpy(p\_set,add);

strcat(p\_set," 25 \"sh /usr/local/share/honeyd/linux/smtp.sh\"\n");

fputs(p\_set,fp);

break;

case 23:

strcpy(p\_set,add);

strcat(p\_set," 23 \"sh /usr/local/share/honeyd/suse7.0/telnetd.sh\"\n");

fputs(p\_set,fp);

break;

case 22:

strcpy(p\_set,add);

strcat(p\_set," 22 \"sh /usr/local/share/honeyd/suse7.0/ssh.sh\"\n");

fputs(p\_set,fp);

break;

case 135:

strcpy(p\_set,add);

strcat(p\_set," 135 \"sh /usr/local/share/honeyd/suse7.0/rpc.sh\"\n");

fputs(p\_set,fp);

break;

default:

sprintf(open\_port,"%d ",m\_port);

strcpy(p\_set,add);

strcat(p\_set,open\_port);

strcat(p\_set,"open\n");

fputs(p\_set,fp);

}

k=t+1;

}

}

/////bind ip 地址；

char bind[100]="bind ";

strcat(bind,value[j].table1);

strcat(bind," ");

strcat(bind,name);

fputs(bind,fp);

fputs("\n\n\n",fp);

}

fclose(fp);

threadC.start();

}

//////

/\* strcpy(netrouteA,value[0].table1);

unsigned int net\_tmpA=inet\_addr(netrouteA);

unsigned int net\_pA=net\_tmpA&0x00FFFFFF;

struct in\_addr ip\_addrA;

ip\_addrA.s\_addr=net\_pA;

char ip\_netA[20];

char ip\_m[100];

sprintf(ip\_netA,"%s",inet\_ntoa(ip\_addrA));

strcat(ip\_netA,"/24 ");

strcpy(ip\_m,ip\_netA);

strcat(ip\_m,netrouteA);

strcat(entry,Localroute);

fputs(entry,fp);

fputs("\n",fp);

int i;

char rout\_buf[100];

strcpy(rout\_buf,route);

strcpy(rout\_buf,Localroute);

strcat(rout\_buf," add net ");

strcat(rout\_buf,ip\_m);

fputs(rout\_buf,fp);

fputs("\n",fp);

char link[100]="route ";

strcat(link,netrouteA);

strcat(link, " link ");

strcat(link ,ip\_netA);

for(i=0;i<=a4;i++)

{

char r\_name[20]="routname";

char m[4];

sprintf(m,"%s",i);

strcat(r\_name,m);

char create[20]="create ";

strcat(create,r\_name);

char set[10]="set ";

strcat(set,r\_name);

char r\_buf[100];

if(value[i].table4==1)

{

char tcp[100]=" default tcp action block\n";

char udp[100]=" default udp action block\n";

char icmp[100]=" default icmp action block\n";

strcpy(r\_buf,set);

strcat(r\_buf,tcp);

fputs(r\_buf,fp);

strcpy(r\_buf,set);

strcat(r\_buf,udp);

fputs(r\_buf,fp);

strcpy(r\_buf,set);

strcat(r\_buf,icmp);

fputs(r\_buf,fp);

}

else

{

char tcp[100]=" default tcp action reset\n";

char udp[100]=" default udp action block\n";

char icmp[100]=" default icmp action open\n";

strcpy(r\_buf,set);

strcat(r\_buf,tcp);

fputs(r\_buf,fp);

strcpy(r\_buf,set);

strcat(r\_buf,udp);

fputs(r\_buf,fp);

strcpy(r\_buf,set);

strcat(r\_buf,icmp);

fputs(r\_buf,fp);

}

int len=strlen(value[i].table2);

char p\_buf[100];

strcpy(p\_buf,value[i].table2);

QTextStream cout(stdout, QIODevice::WriteOnly);

cout <<value[i].table2<<"this yuan port"<<endl;

cout <<p\_buf<< " this port"<< endl;

strcat(p\_buf,",");

int j;

int k=0;

for(j=0;j<=len;j++)

{

//printf("the str is:%c",p\_buf[i]);

//printf("the zifu si:%c\n",'\0');

if(p\_buf[j]==',')

{

char port[6];

char add[100]="add ";

strcat(add,r\_name);

strcat(add,"tcp port ");

//struct p\_port port\_add;

strncpy(port,p\_buf+k,i-k);

//printf("the i is:%d,the port is:%s\n",i,port);

int m\_port=atoi(port);

//printf("the port is:%d\n",m\_port);

char p\_set[100];

char open\_port[6];

switch(m\_port)

{

case 80:

strcpy(p\_set,add);

strcat(p\_set," 80 \"sh /usr/local/share/honeyd/suse7.0/apache.sh\"\n");

fputs(p\_set,fp);

break;

case 21:

strcpy(p\_set,add);

strcat(p\_set," 21 \"sh /usr/local/share/honeyd/linux/ftp.sh\"\n");

fputs(p\_set,fp);

break;

case 110:

strcpy(p\_set,add);

strcat(p\_set," 110 \"sh /usr/local/share/honeyd/linux/qpop.sh\"\n");

fputs(p\_set,fp);

break;

case 25:

strcpy(p\_set,add);

strcat(p\_set," 25 \"sh /usr/local/share/honeyd/linux/smtp.sh\"\n");

fputs(p\_set,fp);

break;

case 23:

strcpy(p\_set,add);

strcat(p\_set," 23 \"sh /usr/local/share/honeyd/suse7.0/telnetd.sh\"\n");

fputs(p\_set,fp);

break;

case 22:

strcpy(p\_set,add);

strcat(p\_set," 22 \"sh /usr/local/share/honeyd/suse7.0/ssh.sh\"\n");

fputs(p\_set,fp);

break;

case 135:

strcpy(p\_set,add);

strcat(p\_set," 135 \"sh /usr/local/share/honeyd/suse7.0/rpc.sh\"\n");

fputs(p\_set,fp);

break;

default:

sprintf(open\_port,"%d ",m\_port);

strcpy(p\_set,add);

strcat(p\_set,open\_port);

strcat(p\_set,"open\n");

fputs(p\_set,fp);

}

k=j+1;

}

}

///////bind ip

char bind[100]="bind ";

strcat(bind,value[i].table1);

strcat(bind,r\_name);

fputs(bind,fp);

fputs("\n\n\n",fp);

}\*/

//////////////////route already over

//fclose(fp);

//}

void FuncUpdatePage::stop\_route()

{

threadC.stop();

}

funcupdatepage.h

#ifndef FUNCUPDATEPAGE\_H

#define FUNCUPDATEPAGE\_H

#include <QtGui>

#include <sys/types.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include "threadc.h"

struct send1

{

char table1[20];

char table2[20];

int table3;

int table4;

};

class FuncUpdatePage:public QWidget

{

Q\_OBJECT

public:

FuncUpdatePage(QWidget \*parent = 0);

~FuncUpdatePage(){}

QTableWidget \*tableWidget;

QTextBrowser \*text3;

QPushButton \*glButton;

QPushButton \*qButton;

QPushButton \*xgButton;

QPushButton \*xzButton;

QPushButton \*xzButton2;

QPushButton \*startButton;

QPushButton \*stopButton;

QLineEdit \*numberLine1;

QLineEdit \*numberLine2;

//QLineEdit \*numberLine3;

QLineEdit \*numberLine4;

QLineEdit \*numberLine5;

QString a1, a2;

int a4, a5;

int k;////总IP地址总数

int np;//单机按钮次数

int n\_buf[255];//存放虚假主机坐标

char Localroute[20];

int name\_n;

ThreadC threadC;

struct send1 value[200];

public slots:

void addB();

void stopinsmod2();

void startq();

void changeTest(int ,int);

void server(int, int);

void start\_route();

void stop\_route();

protected:

void third ();

};

#endif // FUNCUPDATEPAGE\_H

mainwindow.cpp

#include <QDebug>

#include <QFile>

#include <QDragMoveEvent>

#include <QMenu>

#include <QMenuBar>

#include "mainwindow.h"

#include "widget.h"

#include <QBitmap>

#include <QPainter>

MainWindow::MainWindow(QWidget \*parent)

:QFrame(parent)

{

setWindowFlags(Qt::FramelessWindowHint);

// Don't let this widget inherit the parent's backround color

setAutoFillBackground(true);

// Use a brush with a Highlight color role to render the background

// setBackgroundRole(QPalette::Highlight);

//setWindowOpacity(0.9); //

resize(850,600);

setMinimumWidth(850);

setMinimumHeight(600);

setObjectName(tr("MainWindow"));

setWindowModality(Qt::WindowModal);

QPalette palette;

palette.setBrush(this->backgroundRole(), QBrush(QPixmap(":/images/cc.png")));

this->setPalette(palette);

this->setAutoFillBackground(true);

appMinButton = new QPushButton();

appMinButton->setToolTip(tr("最小化"));

appMinButton->setCursor(Qt::PointingHandCursor);

QFile file("/root/final/qss/styleTest.qss");

file.open(QFile::ReadOnly);

QString styleSheet = QLatin1String(file.readAll());

setStyleSheet(styleSheet);

file.close();

appTitle = new QLabel();

appTitle->setText(tr("网络隐形系统"));

appTitle->setObjectName(tr("appTitle"));

appTitle->setContentsMargins(7, 0, 0, 0);

appMinButton = new QPushButton();

appMinButton->setObjectName(tr("appMinButton"));

connect(appMinButton, SIGNAL(clicked()), this, SLOT(showMinimized()));

appCloseButton = new QPushButton();

appCloseButton->setObjectName(tr("appCloseButton"));

appCloseButton->setToolTip(tr("关闭"));

appCloseButton->setCursor(Qt::PointingHandCursor);

connect(appCloseButton, SIGNAL(clicked()), this, SLOT(close()));

hBoxLayout1 = new QHBoxLayout;

hBoxLayout1->setAlignment(Qt::AlignTop|Qt::AlignRight); //°?Å¥×éLayoutÓÒ¶ÔÆë+ÖÃ¶¥

hBoxLayout1->setSpacing(0); //Èý?ö°?Å¥Ö®?äµÄ·ìÏ¶Îª0px

hBoxLayout1->setContentsMargins(0, 0, 10, 0); //ÉèÖÃ°?Å¥×éÀëÓÖ±ß?çµÄ?àÀëÎª10px

//hBoxLayout1->addWidget(styleChangeButton);

//hBoxLayout1->addWidget(appMenuButton);

hBoxLayout1->addWidget(appMinButton);

hBoxLayout1->addWidget(appCloseButton);

hTopLayout = new QHBoxLayout;

hTopLayout->addWidget(appTitle);

hTopLayout->addLayout(hBoxLayout1);

hTopLayout->setAlignment(Qt::AlignTop);

hTopLayout->setMargin(0);

/\*×?Ì¬Ìõ\*/

QHBoxLayout \*statusBar = new QHBoxLayout;

programVersion = new QLabel(tr("程序版本 1"));

dateOfLibs = new QLabel(tr("日期 2012-08-06\t"));

checkUpdate = new QLabel(tr("<a style=\"text-decoration:underline;\">检查更新</a>"));

statusBar->setAlignment(Qt::AlignBottom);

statusBar->setMargin(3);

statusBar->addSpacing(10);

statusBar->addWidget(programVersion);

statusBar->addSpacing(200);

statusBar->addWidget(dateOfLibs);

statusBar->addSpacing(250);

statusBar->addWidget(checkUpdate);

//statusBar->addStretch(1);

//statusBar->addWidget(connectStatus);

FunctionWidget \*functions = new FunctionWidget;

vBoxLayout = new QVBoxLayout;

vBoxLayout->setMargin(0);

vBoxLayout->addLayout(hTopLayout);

// vBoxLayout->addSpacing(4);

vBoxLayout->addWidget(functions);

vBoxLayout->setSpacing(0);

//vBoxLayout->addStretch(1);

//vBoxLayout->addSpacing(1);

vBoxLayout->addLayout(statusBar);

/\*ÉèÖÃ¶¥²¿?°Ìå²??Ö\*/

setLayout(vBoxLayout);

}

//µã»÷Êó±êÊ±?ÇÂ?Êó±êµÄÎ»ÖÃ£¬ÓÃÓÚÊó±êµÄÒÆ¶¯

void MainWindow::mousePressEvent(QMouseEvent \*event)

{

oldMousePos = event->globalPos() - this->pos();

event->ignore();

}

//ÊµÏÖ?çÃæµÄÍÏ¶¯

void MainWindow::mouseMoveEvent(QMouseEvent \*event)

{

if(event->y()<50||event->x()<50||rect().width()-event->x()<5)

{

move(event->globalPos()-oldMousePos);

}

event->ignore();

}

mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QtGui/QMainWindow>

#include <QtGui/QDialog>

#include <QtGui/QButtonGroup>

#include <QtGui/QPushButton>

#include <QToolButton>

#include <QtGui/QStatusBar>

#include <QtGui/QLabel>

#include <QHBoxLayout>

class MainWindow:public QFrame

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = 0);

~MainWindow(){}

private:

QLabel \*appTitle;

QPushButton \*styleChangeButton;

QToolButton \*appMenuButton;

QPushButton \*appMinButton;

QPushButton \*appCloseButton;

QHBoxLayout \*hBoxLayout1;

QVBoxLayout \*vBoxLayout;

QHBoxLayout \*hTopLayout;

QHBoxLayout \*testLayout1;

QHBoxLayout \*testLayout2;

QStatusBar \*statusBar;

QLabel \*programVersion;

QLabel \*dateOfLibs;

QLabel \*connectStatus;

QLabel \*checkUpdate;

protected:

QPoint oldMousePos;

void mouseMoveEvent(QMouseEvent \*event);

void mousePressEvent(QMouseEvent \*event);

};

#endif // MAINWINDOW\_H

shm\_com.h

#ifndef SHM\_COM\_H

#define SHM\_COM\_H

#define TEXT\_SZ 2048

struct \_shm\_data

{

int is\_written;

char data[TEXT\_SZ];

};

typedef struct \_shm\_data shm\_data ;

#endif // SHM\_COM\_H

thread.cpp

#include <QtCore>

#include <iostream>

#include "thread.h"

#include "widget.h"

#include <QMessageBox>

Thread::Thread()

{

}

void Thread::run()

{

char buf2[30];

printf("%d %d %d %d\n", a1, a2, a3, a4);

sprintf(buf2, "./IPMAC %d %d %d %d", a1,a2,a3,a4);

system(buf2);

}

void Thread::stop()

{

system("pkill IPMAC");

}

thread.h

#ifndef THREAD\_H

#define THREAD\_H

#include <QThread>

class Thread : public QThread

{

Q\_OBJECT

public:

int a1;

int a2;

int a3;

int a4;

public:

Thread();

void stop();

protected:

void run();

};

#endif // THREAD\_H

threadb.cpp

#include "threadb.h"

ThreadB::ThreadB()

{

}

void ThreadB::run()

{

char buf[]="honeyd -d -f -i -f syshost.conf ";

system(buf);

}

threadb.h

#ifndef THREADB\_H

#define THREADB\_H

#include <QThread>

class ThreadB : public QThread

{

Q\_OBJECT

public:

public:

ThreadB();

void stop();

protected:

void run();

};

#endif // THREADB\_H

threadc.cpp

#include "threadc.h"

ThreadC::ThreadC()

{

}

void ThreadC::run()

{

system("honeyd -d -i eth0 -f route.conf");

}

void ThreadC::stop()

{

system("pkill honeyd");

}

threadc.h

#ifndef THREADC\_H

#define THREADC\_H

#include <QThread>

class ThreadC : public QThread

{

Q\_OBJECT

public:

char a1[20], a2[20], a3[20];

int a4,a5;

public:

ThreadC();

void stop();

protected:

void run();

};

#endif // THREADC\_H

widget.cpp

#include <QHBoxLayout>

#include <QVBoxLayout>

#include "widget.h"

#include <QtGui>

FunctionWidget::FunctionWidget(QWidget \*parent)

:QDialog(parent)

{

setObjectName(tr("functionWidget"));

a = new FuncScanPage;

b = new FuncGuardPage;

c = new FuncUpdatePage;

d = new FuncToolsPage;

funcPages = new QStackedWidget;

funcPages->addWidget(a);

funcPages->addWidget(b);

funcPages->addWidget(c);

funcPages->addWidget(d);

scanButton = new QPushButton;

scanButton->setObjectName(tr("scanButton"));

//scanButton->setText(tr("主页"));

scanButton->setFlat(true);

scanButton->setIcon(QIcon(":/images/zhuye.png"));

scanButton->setIconSize(QSize(120,90));

connect(scanButton,SIGNAL(clicked()),this,SLOT(setScanPage()));

guardButton = new QPushButton;

guardButton->setObjectName(tr("guardButton"));

//guardButton->setText(tr("虚拟化网络"));

guardButton->setFlat(true);

guardButton->setIcon(QIcon(":/images/zhencha.png"));

guardButton->setIconSize(QSize(120,90));

connect(guardButton,SIGNAL(clicked()),this,SLOT(setGuardPage()));

updateButton = new QPushButton;

updateButton->setObjectName(tr("updateButton"));

//updateButton->setText(tr("内核隐身"));

updateButton->setFlat(true);

updateButton->setIcon(QIcon(":/images/xuni.png"));

updateButton->setIconSize(QSize(120,90));

connect(updateButton,SIGNAL(clicked()),this,SLOT(setUpdatePage()));

toolsButton = new QPushButton;

toolsButton->setObjectName(tr("toolsButton"));

//toolsButton->setText(tr("监控日志"));

toolsButton->setFlat(true);

toolsButton->setIcon(QIcon(":/images/rizhi.png"));

toolsButton->setIconSize(QSize(120,90));

connect(toolsButton,SIGNAL(clicked()),this,SLOT(setToolsPage()));

QHBoxLayout \*funcBtnLayout = new QHBoxLayout;

funcBtnLayout->setMargin(0);

funcBtnLayout->setAlignment(Qt::AlignTop);

funcBtnLayout->addWidget(scanButton);

funcBtnLayout->addWidget(guardButton);

funcBtnLayout->addWidget(updateButton);

funcBtnLayout->addWidget(toolsButton);

funcBtnLayout->addStretch(1);

QVBoxLayout \*mainLayout = new QVBoxLayout;

mainLayout->addLayout(funcBtnLayout);

mainLayout->setSpacing(0); //

mainLayout->addWidget(funcPages);

//mainLayout->addStretch(1);

setLayout(mainLayout);

}

void FunctionWidget::setScanPage()

{

funcPages->setCurrentIndex(0);

}

void FunctionWidget::setGuardPage()

{

funcPages->setCurrentIndex(1);

}

void FunctionWidget::setUpdatePage()

{

funcPages->setCurrentIndex(2);

}

void FunctionWidget::setToolsPage()

{

funcPages->setCurrentIndex(3);

}

widget.h

#ifndef WIDGET\_H

#define WIDGET\_H

#include <QDialog>

#include <QStackedWidget>

#include <QPushButton>

#include "funcguardpage.h"

#include "funcscanpage.h"

#include "functoolspage.h"

#include "funcupdatepage.h"

class FunctionWidget:public QDialog

{

Q\_OBJECT

public:

FunctionWidget(QWidget \*parent = 0);

~FunctionWidget(){}

public:

QPushButton \*scanButton;

QPushButton \*guardButton;

QPushButton \*updateButton;

QPushButton \*toolsButton;

QStackedWidget \*funcPages;

FuncScanPage \*a;

FuncGuardPage \*b;

FuncUpdatePage \*c;

FuncToolsPage \*d;

public slots:

void setScanPage();

void setGuardPage();

void setUpdatePage();

void setToolsPage();

};

#endif // WIDGET\_H

main.cpp

#include <QtGui/QApplication>

#include <QTextCodec>

#include "mainwindow.h"

#include <QSplashScreen>

int main(int argc, char \*argv[])

{

QApplication a(argc,argv);

QTextCodec::setCodecForTr(QTextCodec::codecForName("utf8"));

MainWindow w;

w.setWindowIcon(QIcon(":/images/www.png"));

w.show();

return a.exec();

}