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

Search ASM Code for Overflow exploit

提交作品： lion
提交日期： 2004-02-29
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6次/总3951次

```
/*
**************************************************************************
*
* sac.cpp - Search ASM Code for Overflow exploit.
*
* Copyright (C) 2004 HUC All Rights Reserved.
*
* 1. 在同一种系统的各个语言版本(如win2000 sp4), 在 msvcrt.dll 中找到的地址可以通用.
* 2. 在同一种语言的各个系统版本(如简体中文win2000, winxp, win2003), 在 0x7FFA000
*
* Code by lion (lion@cnhonker.net)
* Welcome to HUC Website: http://www.cnhonker.com
* 2004-01-28
*
* ASM Code:
*
*       call eax        FF D0
*       call ebx        FF D3
*       call ecx        FF D1
*       call edx        FF D2
*       call edi        FF D7
*       call esi        FF D6
*       call esp        FF D4
*       call ebp        FF D5
*
*       call [eax]     FF 10
*       call [ebx]     FF 13
*       call [ecx]     FF 11
*       call [edx]     FF 12
*       call [edi]     FF 17
*       call [esi]     FF 16
*       call [esp]     FF 14 24
*       call [ebp]     FF 55 00
*
*       jmp eax         FF E0
*       jmp ebx         FF E3
*       jmp ecx         FF E1
```

```
*      jmp edx       FF E2
*      jmp edi       FF E7
*      jmp esi       FF E6
*      jmp esp       FF E4
*      jmp ebp       FF E5
*
*      jmp [eax]    FF 20
*      jmp [ebx]    FF 23
*      jmp [ecx]    FF 21
*      jmp [edx]    FF 22
*      jmp [edi]    FF 27
*      jmp [esi]    FF 26
*      jmp [esp]    FF 24 24
*      jmp [ebp]    FF 65 00
*
*      push eax       50
*      push ebx       53
*      push ecx       51
*      push edx       52
*      push edi       57
*      push esi       56
*      push esp       54
*      push ebp       55
*
*      push [eax]    FF 30
*      push [ebx]    FF 33
*      push [ecx]    FF 31
*      push [edx]    FF 32
*      push [edi]    FF 37
*      push [esi]    FF 36
*      push [esp]    FF 34 24
*      push [ebp]    FF 75 00
*
*      pop eax       58
*      pop ebx       5B
*      pop ecx       59
*      pop edx       5A
*      pop edi       5F
*      pop esi       5E
*      pop esp       5C
*      pop ebp       5D
*
*      ret          C3
*
**********************************************************************
*/

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

#pragma comment(lib, "Version.lib")

#define VERSION  "0.20"

void usage(char *p)
{
   printf("Usage:\t%s\t-r <register> [Options]\n"
      "\t\t-v Show System Version\n\n"
      "[Options:]\n"
      "\t-r\tregister \tEAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP, XPRET\n"
      "\t-d\tdllname \tNODLL (0x7FFA0000), MSVCRT, USER32 or Other\n"
//       "\t-l\tlogfile \tlog.txt\n"
      "\n", p);
}

BOOL DisplaySystemVersion()
```

```
{
  OSVERSIONINFOEX osvi;
  BOOL bOsVersionInfoEx;

  // Try calling GetVersionEx using the OSVERSIONINFOEX structure.
  //
  // If that fails, try using the OSVERSIONINFO structure.

  ZeroMemory(&osvi, sizeof(OSVERSIONINFOEX));
  osvi.dwOSVersionInfoSize = sizeof(OSVERSIONINFOEX);

  if( !(bOsVersionInfoEx = GetVersionEx ((OSVERSIONINFO *) &osvi)) )
  {
    // If OSVERSIONINFOEX doesn't work, try OSVERSIONINFO.

    osvi.dwOSVersionInfoSize = sizeof (OSVERSIONINFO);
    if (! GetVersionEx ( (OSVERSIONINFO *) &osvi) )
      return FALSE;
  }

  switch (osvi.dwPlatformId)
  {
    case VER_PLATFORM_WIN32_NT:

    // Test for the product.

      if ( osvi.dwMajorVersion <= 4 )
        printf("Microsoft Windows NT ");

      if ( osvi.dwMajorVersion == 5 && osvi.dwMinorVersion == 0 )
        printf ("Microsoft Windows 2000 ");

      if ( osvi.dwMajorVersion == 5 && osvi.dwMinorVersion == 1 )
        printf ("Microsoft Windows XP ");

      if ( osvi.dwMajorVersion == 5 && osvi.dwMinorVersion == 2 )
        printf ("Microsoft Windows 2003 ");

    // Test for product type.

      if( bOsVersionInfoEx )
      {
        if ( osvi.wProductType == VER_NT_WORKSTATION )
        {
          if( osvi.wSuiteMask & VER_SUITE_PERSONAL )
            printf ( "Personal " );
          else
            printf ( "Professional " );
        }

        else if ( osvi.wProductType == VER_NT_SERVER )
        {
          if( osvi.wSuiteMask & VER_SUITE_DATACENTER )
            printf ( "DataCenter Server " );
          else if( osvi.wSuiteMask & VER_SUITE_ENTERPRISE )
            printf ( "Advanced Server " );
          else
            printf ( "Server " );
        }
      }


    // Display version, service pack (if any), and build number.

      if ( osvi.dwMajorVersion <= 4 )
      {
        printf ("version %d.%d %s (Build %d)\n",
```

```
            osvi.dwMajorVersion,
            osvi.dwMinorVersion,
            osvi.szCSDVersion,
            osvi.dwBuildNumber & 0xFFFF);
        }
        else
        {
          printf ("%s (Build %d)\n",
            osvi.szCSDVersion,
            osvi.dwBuildNumber & 0xFFFF);
        }
        break;
  }
  return TRUE;
}


int main(int argc, char *argv[])
{
    bool bLoaded = false;
    HINSTANCE h;
    char *pDllName = "msvcrt.dll";
    char *pRegister = "ebx";
    bool bEAX = false, bEBX = false, bECX = false, bEDX = false, bEDI = false, bESI
bEBP = false, bRET = false;
    int count = 0;

    printf("Search ASM Code Tool for Overflow exploit V%s\n"
        "Code by lion (lion@cnhonker.net)\n"
        "Welcome to HUC website http://www.cnhonker.com\n\n"
        , VERSION);

    if(argc < 2)
    {
        usage(argv[0]);
        return -1;
    }
    else
    if(argc == 2 && argv[1][1] == 'v')
    {
        DisplaySystemVersion();
        return 1;
    }

    for(int 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 'r':
                pRegister = argv[i+1];

                if(stricmp(pRegister, "eax") == 0)
                    bEAX = true;
                else
                if(stricmp(pRegister, "ebx") == 0)
                    bEBX = true;
```

```
            else
            if(stricmp(pRegister, "ecx") == 0)
               bECX = true;
            else
            if(stricmp(pRegister, "edx") == 0)
               bEDX = true;
            else
            if(stricmp(pRegister, "edi") == 0)
               bEDI = true;
            else
            if(stricmp(pRegister, "esi") == 0)
               bESI = true;
            else
            if(stricmp(pRegister, "esp") == 0)
               bESP = true;
            else
            if(stricmp(pRegister, "ebp") == 0)
               bEBP = true;
            else
            if(stricmp(pRegister, "xpret") == 0)
               bRET = true;
            else
            {
               usage(argv[0]);
               return -1;
            }
            break;
         case 'd':
            pDllName = argv[i+1];
            break;
      }
   }

   //DisplaySystemVersion();

   printf("Search Mode or Register: %s\n", pRegister);

   // no dll for search.
   if(stricmp(pDllName, "nodll") == 0)
   {
      h = (HINSTANCE__ *)0x7FFa0000;

      printf("Start search ASM code in: 0x7FFA0000\n\n", pRegister);
   }
   else
   {
      // Load DLL.
      h = GetModuleHandle(pDllName);
      if(h == NULL)
      {
         h = LoadLibrary(pDllName);
         if(h == NULL)
         {
            printf("Loading DLL error: %s\n", pDllName);
            return -1;
         }

         printf("Start Search ASM Code in: %s\n\n", pDllName);

         bLoaded = true;
      }
   }

// Search TOP SEH
/*
   unsigned int *un;
   unsigned int sehaddr;
```

```
   HMODULE hk = LoadLibrary("kernel32");

   un = (unsigned int *)GetProcAddress(hk, "SetUnhandledExceptionFilter");

//    un = (int *)UnhandledExceptionFilter;
   _asm{
      mov eax,un
      add eax,5
      mov ebx,[eax]
      mov sehaddr, ebx
   }

   printf("0x%X\tTOP SEH\n\n", sehaddr);
*/

   BYTE* ptr = (BYTE*)h;
   bool done = false;

   // Start Search ASM Code
   for(ULONG y = 0; !done; y++)
   {
      try
      {
         if(ptr[y] == 0xFF )
         {
            ULONG pos = (ULONG)ptr + y;

            if(bEAX)
            {
               if(ptr[y+1] == 0xD0)
               {
                  printf("0x%X\tcall eax\r\n", pos);
                  count++;
               }
               else
               if(ptr[y+1] == 0xE0)
               {
                  printf("0x%X\tjmp eax\r\n", pos);
                  count++;
               }
            }
            else
            if(bEBX)
            {
               if(ptr[y+1] == 0xD3)
               {
                  printf("0x%X\tcall ebx\r\n", pos);
                  count++;
               }
               else
               if(ptr[y+1] == 0xE3)
               {
                  printf("0x%X\tjmp ebx\r\n", pos);
                  count++;
               }
            }
            else
            if(bECX)
            {
               if(ptr[y+1] == 0xD1)
               {
                  printf("0x%X\tcall ecx\r\n", pos);
                  count++;
               }
               else
               if(ptr[y+1] == 0xE1)
```

```
    {
        printf("0x%X\tjmp ecx\r\n", pos);
        count++;
    }
}
else
if(bEDX)
{
    if(ptr[y+1] == 0xD2)
    {
        printf("0x%X\tcall edx\r\n", pos);
        count++;
    }
    else
    if(ptr[y+1] == 0xE2)
    {
        printf("0x%X\tjmp edx\r\n", pos);
        count++;
    }
}
else
if(bEDI)
{
    if(ptr[y+1] == 0xD7)
    {
        printf("0x%X\tcall edi\r\n", pos);
        count++;
    }
    else
    if(ptr[y+1] == 0xE7)
    {
        printf("0x%X\tjmp edi\r\n", pos);
        count++;
    }
}
else
if(bESI)
{
    if(ptr[y+1] == 0xD6)
    {
        printf("0x%X\tcall esi\r\n", pos);
        count++;
    }
    else
    if(ptr[y+1] == 0xE6)
    {
        printf("0x%X\tjmp esi\r\n", pos);
        count++;
    }
}
else
if(bESP)
{
    if(ptr[y+1] == 0xD4)
    {
        printf("0x%X\tcall esp\r\n", pos);
        count++;
    }
    else
    if(ptr[y+1] == 0xE4)
    {
        printf("0x%X\tjmp esp\r\n", pos);
        count++;
    }
}
else
if(bEBP)
```

```
            {
               if(ptr[y+1] == 0xD5)
               {
                  printf("0x%X\tcall ebp\r\n", pos);
                  count++;
               }
               else
               if(ptr[y+1] == 0xE5)
               {
                  printf("0x%X\tjmp ebp\r\n", pos);
                  count++;
               }
            }
         }
         else if(ptr[y] == 0xC3)
         {

            // Start search pop reg, pop reg, ret addr.
//             if(bRET || bEBX)
            if(bRET)
            {
               if(ptr[y-1] >= 0x58 && ptr[y-1] <= 0x5F)
               {
                  if(ptr[y-2] >= 0x58 && ptr[y-2] <= 0x5F)
                  {

                     ULONG pos = (ULONG)ptr + y - 2;
                     printf("0x%X\tpop reg, pop reg, ret\r\n", pos);
                     count++;
                  }
/*
                  else if((ptr[y-6] == 0xBF) && (ptr[y-7] >= 0x58 && ptr[y-7] <= 0
                  {
                     ULONG pos = (ULONG)ptr + y - 7;
                     printf("0x%X\tpop reg, pop reg, ret\n", pos);
                     count++;
                  }
*/
               }
            }

            ULONG pos = (ULONG)ptr + y - 1;

            if(bEAX)
            {
               if(ptr[y-1] == 0x50)
               {
                  printf("0x%X\tpush eax, ret\n", pos);
                  count++;
               }
            }
            else
            if(bEBX)
            {
               if(ptr[y-1] == 0x53)
               {
                  printf("0x%X\tpush ebx, ret\n", pos);
                  count++;
               }
            }
            else
            if(bECX)
            {
               if(ptr[y-1] == 0x51)
               {
                  printf("0x%X\tpush ecx, ret\n", pos);
                  count++;
```

```
            }
         }
         else
         if(bEDX)
         {
            if(ptr[y-1] == 0x52)
            {
               printf("0x%X\tpush edx, ret\n", pos);
               count++;
            }
         }
         else
         if(bEDI)
         {
            if(ptr[y-1] == 0x57)
            {
               printf("0x%X\tpush edi, ret\n", pos);
               count++;
            }
         }
         else
         if(bESI)
         {
            if(ptr[y-1] == 0x56)
            {
               printf("0x%X\tpush esi, ret\n", pos);
               count++;
            }
         }
         else
         if(bESP)
         {
            if(ptr[y-1] == 0x54)
            {
               printf("0x%X\tpush esp, ret\n", pos);
               count++;
            }
         }
         else
         if(bEBP)
         {
            if(ptr[y-1] == 0x55)
            {
               printf("0x%X\tpush ebp, ret\n", pos);
               count++;
            }
         }
      }
   }

   catch(...)
   {
      printf("\nSearch end.\r\n", pDllName);
      printf("Found %d addr.\r\n", count);

      done = true;
   }
 }

 if(bLoaded)
   FreeLibrary(h);

 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