Create your Proxy DLLs automatically

Michael Chourdakis, 14 May 2007

Here is a small program that will create the CPP and DEF for a proxy DLL, based on the exports of another DLL. You can use it to generate a template and then you edit this template to satisfy your needs.

Download source - 2.09 Kb

Introduction

A lot of us have tried to create a proxy DLL to replace an existing one and spy other programs' calls. Here is a small program that will create the CPP and DEF for a proxy DLL based on the exports of another DLL. You can use it to generate a template and then edit this template to satisfy your needs.

Background

When creating a proxy DLL, you have to export precisely the same names as exported by the original DLL. This can be painful, for two reasons:

1. There are too many exports.
2. There are functions that you don't know what they do; you'd just want to spy on one specific function call.

The second problem is solved with assembly and with the aid of the `__declspec(naked)` attribute. The program creates function stubs that do nothing but `JUMP` (not call) to the exported address, so the stack is left as it should be. This allows you to create code only for functions that you actually know what they do.

Using the program

Hide Copy Code

```
WRAPPIT <dll> <txt> <convention> <point dll name> <cpp> <def>
```

- `<dll>` is the new DLL name you want to create. The program can compile the DLL using VC++ or BC++, depending on how you comment or edit lines 243-248:

Hide Copy Code

```
//
// _stprintf(ay,_T("BCC32 -o%s.obj -c %s\r\n"),argv[5],argv[5]);
_stprintf(ay,_T("CL.EXE /O2 /GL /I \".\" /D \"WIN32\" /D \"NDEBUG\" /D"
          "\"_WINDOWS\" /D \"_WINDLL\" /FD /EHsc /MT /Fo\".\\%s.obj\" "
          "/Fd\".\\vc80.pdb\" /W3 /nologo /c /Wp64 /TP "
          "/errorReport:prompt %s\r\n"),argv[5],argv[5]);
system(ay);
// _stprintf(ay,_T("ILINK32 -c -Tpd %s.obj,
//           %s,,,%s\r\n"),argv[5],argv[1],argv[6]);
_stprintf(ay,_T("LINK.EXE /OUT:\"%s\" /INCREMENTAL:NO /NOLOGO /DLL"
          " /MANIFEST /DEF:\"%s\" /SUBSYSTEM:WINDOWS /OPT:REF "
          "/OPT:ICF /LTCG /MACHINE:X86 /ERRORREPORT:PROMPT "
          "%s.obj kernel32.lib user32.lib gdi32.lib winspool.lib "
          "comdlg32.lib advapi32.lib shell32.lib ole32.lib "
          "oleaut32.lib uuid.lib odbc32.lib odbccp32.lib\r\n"),
          argv[1],argv[6],argv[5]);
```

```
system(ay);
//
```

- `<txt>` is a text file containing the exports from the original DLL. You can create this file with either `dumpbin`:

  Hide Copy Code

  ```
  dumpbin /exports original.dll > exports.txt
  ```

  or with `tdump`:

  Hide Copy Code

  ```
  tdump original.dll -ee > exports.txt
  ```

- `<convention>` is the convention call you want your functions to have. You will usually want to use `__stdcall`, but it hardly matters what you use because the stub functions immediately jump to the existing code and therefore, they should work with any calling convention.
- `<point dll name>` is the DLL name that your proxy DLL will try to load. Make sure you use C++ escape characters like `\\`.
- `<cpp>` is the generated CPP file.
- `<def>` is the generated DEF file.

Example:

You have *WSOCK32.DLL* and you want to create a proxy for it, replacing the original DLL as *WSOCK32_.DLL*. What would you do?

- `move wsock32.dll wsock32_.dll`
- `dumpbin /exports wsock32_.dll > exports.txt`
- `wrappit wsock32.dll exports.txt __stdcall .\\wsock32_.dll wsock32.cpp wsock32.def`

This will:

- Parse the text file for exports and create the DEF. Exported functions by ordinal only are supported.
- Create the sample CPP code. In the DLL's code `DllMain`, the original `wsock32_dll` will be loaded with `LoadLibrary()`. Then all the original exported functions' addresses will be returned by `GetProcAddress` and stored in an internal pointer. Then stubs for each function will be created.

A single CPP will look like this:

Hide Shrink Copy Code

```
//
#include <windows.h>
#pragma pack(1)
HINSTANCE hLThis = 0;
HINSTANCE hL = 0;
FARPROC p[75] = {0};
// -----------
BOOL WINAPI DllMain(HINSTANCE hInst,DWORD reason,LPVOID)
{
    if (reason == DLL_PROCESS_ATTACH)
    {
        hLThis = hInst;
        hL = LoadLibrary(".\\wsock32_.dll");
        if (!hL) return false;

        p[0] = GetProcAddress(hL,"AcceptEx");
        p[1] = GetProcAddress(hL,"EnumProtocolsA");
        p[2] = GetProcAddress(hL,"EnumProtocolsW");
      ...
    }
    if (reason == DLL_PROCESS_DETACH)
    {
        FreeLibrary(hL);
    }
    return 1;
}
```

```
// AcceptEx
extern "C" __declspec(naked) void __stdcall __E__0__()
{
    __asm
    {
        jmp p[0*4];
    }
}

// EnumProtocolsA
extern "C" __declspec(naked) void __stdcall __E__1__()
{
    __asm
    {
        jmp p[1*4];
    }
}

// EnumProtocolsW
extern "C" __declspec(naked) void __stdcall __E__2__()
{
    __asm
    {
        jmp p[2*4];
    }
}
...
//
```

A single DEF will look like this:

Hide Copy Code

```
EXPORTS
AcceptEx=__E__0__ @1141
EnumProtocolsA=__E__1__ @1111
EnumProtocolsW=__E__2__ @1112
...
```

You may now edit CPP/DEF files and reuse them to create your own proxy DLL!

Important!

Once the cpp is ready, you should replace functions that you know how to use. For example, If you want to spy on `Wsock32.send()`:

Hide Copy Code

```
// send, created by wrappit
extern "C" __declspec(naked) void __stdcall __E__69__()
   {
   __asm
    {
    jmp p[69*4];
    }
 }

// If you want to manipulate it, change to:
extern "C" int __stdcall __E__69__(SOCKET x,char* b,int l,int pr)
  {
  // manipulate here parameters

.....
  // call original send
     typedef int (__stdcall *pS)(SOCKET,char*,int,int);
     pS pps = (pS)p[63*4];
     int rv = pps(x,b,l,pr);
```

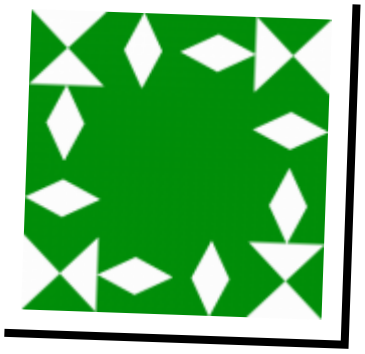
```
    return rv;
}
```

History

- 14 May, 2007 - Fixed problem occuring when *dumpbin.exe* generates RVA information as well

License

About the Author



Michael Chourdakis

Engineer

Greece

I'm working in C++, PHP , Java, Windows, iOS and Android.

I 've a PhD in Digital Signal Processing and I specialize in Pro Audio applications.

My home page: http://www.michaelchourdakis.com