



W32Dasm 使用介绍

本节主要介绍反汇编。在进行反汇编前，建议用 FileInfo，PEiD 等侦测工具分析一下文件是否加壳。如果加壳，就需利用后面章节的脱壳技术脱壳，脱壳后再反汇编。

1.1 准备工作

在此以一个 Microsoft Visual C++ 6.0 编译程序 ReverseMe 进行反汇编，以讲述如何从反汇编代码清单中理解程序的功能。ReverseMe 是一个基本的 Win32 程序，该程序创建一个普通应用程序窗口，有关程序运行机理请参考 Win32 编程的相关资料。源码如下：

```
#include <windows.h>
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    static TCHAR szAppName[] = TEXT ("chap231");
    HWND          hwnd ;
    MSG           msg ;
    WNDCLASS      wndclass ;
    // 填充窗口类结构
    wndclass.style = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc = WndProc ;
    wndclass.cbClsExtra = 0 ;
    wndclass.cbWndExtra = 0 ;
    wndclass.hInstance = hInstance ;
    wndclass.hIcon = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName = NULL ;
    wndclass.lpszClassName = szAppName ;
    if (!RegisterClass (&wndclass))
    {
        return 0 ;
    }
    //创建窗口
    hwnd = CreateWindow (szAppName,
                        TEXT ("静态分析技术实例"),
                        // 指定窗口类名
                        // 指向窗口标题
```

```

        WS_OVERLAPPEDWINDOW,    // 窗口样式
        CW_USEDEFAULT,          // 窗口左上角坐标: x
        CW_USEDEFAULT,          // 窗口左上角坐标: y
        CW_USEDEFAULT,          // 窗口的宽度
        CW_USEDEFAULT,          // 窗口的高度
        NULL,                   // 父窗口句柄, 无则为 NULL
        NULL,                   // 指定窗口主菜单句柄
        hInstance,              // 应用程序当前的句柄
        NULL);                  // 指向传递给窗口的参数值指针
ShowWindow (hwnd, iCmdShow);    // 显示窗口
UpdateWindow (hwnd);           // 更新窗口的客户区
//建立消息循环, 从该应用程序的消息队列中检取消息, 送到消息处理过程
while (GetMessage (&msg, NULL, 0, 0))
{
    TranslateMessage (&msg);
    DispatchMessage (&msg);
}
return msg.wParam ;
}

//窗口函数
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    int input;
    switch (message)
    {
    case WM_DESTROY:                // 退出消息
        PostQuitMessage (0);       // 调用退出函数
        return 0 ;
    }
    case WM_CLOSE:
        input=MessageBox(hwnd,TEXT ("你真的决定退出吗?"),TEXT ("退出"),
                        MB_YESNO|MB_ICONQUESTION);
        switch(input)
        {
            case IDYES:
                break;
            case IDNO:
                return 0;
        }
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}


```

```
}
```

1.2 操作步骤


W32Dasm 是一款功能强大、使用简单并且免费的反汇编工具，其终结版是 8.93。原本不支持中文字串，故爱好者需要将其改进，使其支持中文字串。

1. 选项设置

- ☐ 运行 W32Dasm，从“反汇编 (Disassembler)”菜单中，选择“反汇编程序选项 (Disassembler Options)”，将出现如图 3.6 中所示的对话框，将 3 项交叉参考功能全选上。
- ☐ 从“反汇编 (Disassembler)”菜单中，选择“打开文件 (Open File)”选项或按工具栏按钮。然后选择光盘提供的 ReverseMe 文件。
- ☐ 如果反汇编代码显示不全或者有字体太小的现象，可以在“反汇编 (Disassembler) / 字体 (Font)”菜单中进行适当调整。默认的字体系是 8 号的 Courier New。

2. 保存反汇编文本文件

在这里，可把反汇编代码保存为 ASCII 或 alf 项目文件。这样，再次分析该软件时，只要打开该项目文件即可。

- ☐ 从“反汇编 (Disassembler)”菜单中选择“保存反汇编文件或创建工程文件 (Save Disassembly Text File and Create Project File)”或按工具栏按钮。
- ☐ 在另存为对话框中，默认文件是以 .alf 为扩展名。当 alf 文件被创建时，wpj 文件自动创建，.alf 是一个文本格式文件。

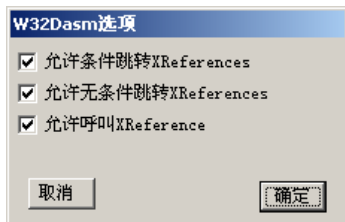




图 3.6 W32Dasm 选项对话框

3. 查找功能

利用查找功能可以查找反汇编清单中的相关代码和字段。在工具栏上按按钮或从“查找 (Search)”菜单中打开查找对话框，只要是清单上出现的字符都可被列入查找范围。

4. 转到程序入口点 (Goto Program Entry Point)

在工具栏中按按钮或从“转移 (Goto)”菜单中选择“转移到程序入口点 (Goto Program Entry Point)”或按 F10 键，这样光标将转到程序入口点 (Entry Point)，这里就是程序执行的起点，此时可通过双击鼠标或用“Shift+上下光标键”改变光条的位置。反汇编清单主要是由虚拟地址、机器码、汇编代码三部分组成，如图 3.7 所示。

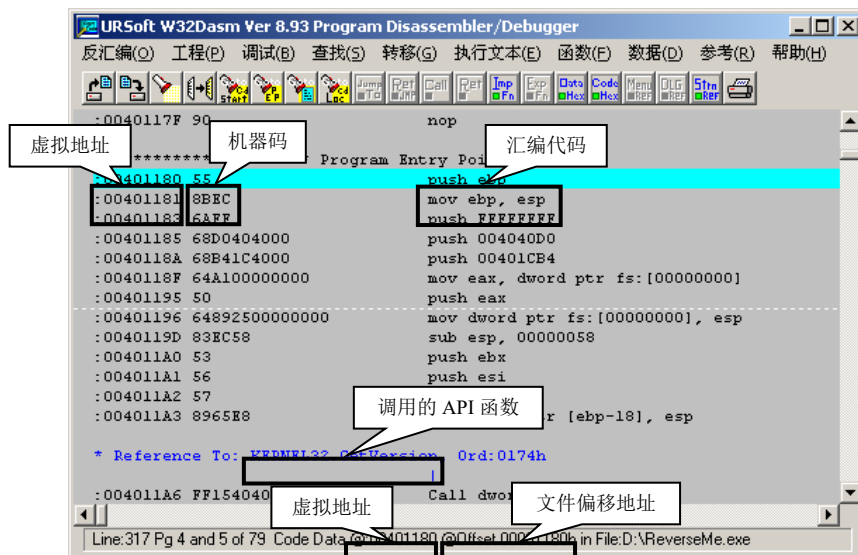



图 3.7 W32Dasm 主窗口

5. VA 与 File Offset 转换


可用 W32Dasm 来实现内存虚拟地址（Virtual Address，简称 VA）和文件偏移地址（File Offset）的转换。在 W32Dasm 主窗口中将光条移到某一行代码上，窗口底部有一行文字提示其文件偏移地址。本例中，光条在 ReverseMe 程序入口点那行，W32Dasm 状态栏显示入口点虚拟地址为 CodeData @:00401180h，对应的文件偏移地址为 @Offset 00001180h。

Visual C++ 编译器默认编译时，程序的区块在磁盘中的偏移值与内存中是相同的。此时文件偏移地址就等于相对虚拟地址（RVA）。转换相当方便，将虚拟地址减去基地址就得到文件偏移地址，即 $0x401180 - 0x400000 = 0x1180$ 。


6. 转到代码开始处（Goto Code Start）

在工具栏中按  按钮或从“转移（Goto）”菜单中选择“转到代码头（Goto Code Start）”或按 Ctrl+S 键。不要将代码起始点和程序入口点搞混淆，代码的起始点是反汇编代码清单中汇编指令的开始点。




7. 转到页（Goto Page）

在工具栏中按  按钮或从“转移（Goto）”菜单中选择“转到页（Goto Page）”或按 F11 键，这时弹出一个对话框，输入页数可跳转到相关页面。

8. 转到代码位置（Goto Code Location）

在工具栏中按  按钮或从 Goto 菜单中选择“转到代码位置（Goto Code Location）”或按 Shift + F12 键，将出现一个对话框，允许用户输入代码虚拟地址，以跳转到此地址处。如果输入的虚拟地址值小于或者大于当前反汇编代码中的有效虚拟地址值，程序将会自动取最接近的有效地址；如果输入的虚拟地址值在有效范围内但是没有精确值与之匹配，则最接近的有效虚拟地址值将自动被选取。在此填入 00401078，以跳到 00401078 地址处。

9. 文本跳转

该功能是在“执行文本 (Execute Text)”菜单里,“执行跳跃 (Execute Jump)”功能的激活条件是光标在代码的跳转指令这行上(这时光条是高亮度的绿颜色)。此时工具条的跳转按钮也被激活,具体见图 3.8。按按钮或选择菜单选项“执行跳跃 (Execute Jump)”或按右光标键,光条将跳转到指令所指的位置。在本例中,将转到:00401083 这一行代码处。此时执行菜单中“返回上一次跳跃 (Return From Last Jump)”、按按钮或在“:00401078(C)”处双击右键,光条将返回到上次跳跃位置处。

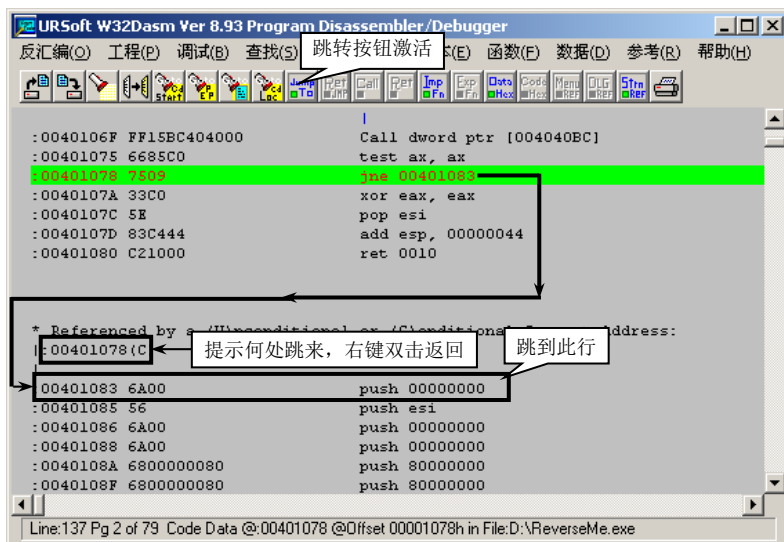


图 3.8 文本跳转

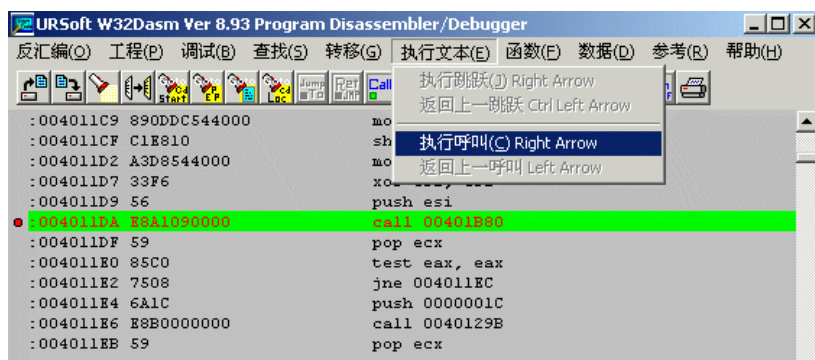






图 3.9 执行呼叫

10. 文本呼叫

这个功能在“执行文本 (Execute Text)”菜单中,此功能激活的条件是光条在 CALL 指令行上。在这一行上时光条将发绿,按钮将被激活。执行时光条将转到 CALL 所指向的地址处。如图 3.9 所示,光条在 4011DA 代码行上,此时按按钮或从菜单中选择“执行呼叫 (Execute Text Call)”或按右光标键,光条将转到 CALL 所指的地址 4011DA 这一行。执行

后，按  按钮或从菜单里选择“返回呼叫（Return From Last Call）”或按左光标键或双击右键，光条将返回到上一次呼叫位置处。

11. 输入函数（Imports）

此命令是查看 PE 文件的输入函数（Imports），输入函数是指调用外部 DLL 文件的函数。按  按钮或从“函数（Functions）”菜单中选择“输入（Imports）”命令，执行后将列出当前文件的输入函数（见图 3.10）。可以双击表中的函数项目，主窗口的光条将转到调用这些函数的代码处。如果代码多处引用这些函数，双击这个项目函数时，光条将在调用它的几个位置代码处循环。

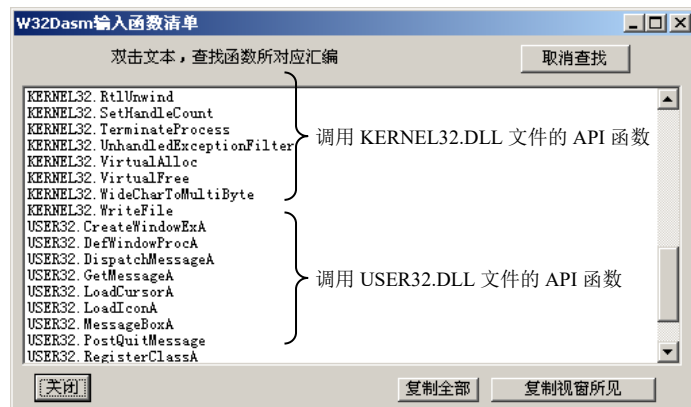



图 3.10 输入表

12. 输出函数（Exports）

此命令选项是查看 PE 文件的输出函数（Exports）。输出函数一般在 DLL 文件中，它的作用是提供内部函数接口供外部程序调用。例如，USER32.DLL 链接库提供各种 API 函数接口给应用程序调用。EXE 文件一般只有输入表，没有输出表，DLL 文件两者都有。

用 W32Dasm 打开 Windows 98 目录里的 USER32.DLL 文件，然后按  按钮或在“函数（Functions）”菜单中选择“输出（Exports）”命令，执行后将列出当前文件的输出函数（见图 3.11）。双击 MessageBoxW 函数，来到 MessageBoxW 函数代码处，在此将会看到 MessageBoxW 函数是如何将 Unicode 字符串转换成 ANSI 字符串的，最后再调用 ANSI 版的 MessageBoxExA 函数来显示窗口。

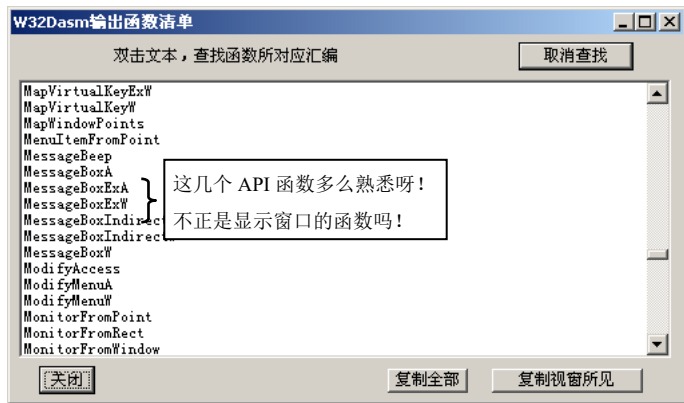




图 3.11 输出表

13. 串式数据参考 (String Data References)

串式数据参考功能列出了程序中相关的字符串、对话等信息，在静态分析中，利用这些信息可快速定位到相关代码上，该功能使用十分频繁。重新用 W32Dasm 打开 ReverseMe 程序，按  按钮或在“参考 (References)”菜单中选择“串式数据参考 (String Data References)”命令，打开串式数据参考窗口 (见图 3.12)。可以双击这些字符串跳到相关代码上。如多处调用，双击这个字符串时，光条将在调用它的几个位置代码处循环。

14. 对话框参考 (Dialog References)

对话框参考功能列出了程序调用对话框的信息。由于 ReverseMe 程序没采用对话框和菜单，因此用 W32Dasm 打开 Windows 的记事本程序 (notepad.exe) 来讲解。按  按钮或在

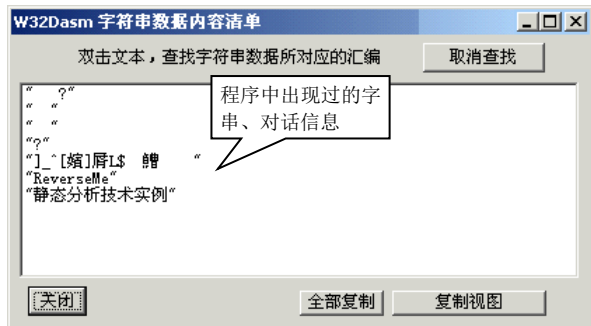



图 3.12 串式数据参考

“参考 (References)”菜单中选择“对话框参考 (Dialog References)”命令，打开对话框参考窗口 (见图 3.13)。

对话框参考形式是 Dialog: DialogID_000E，其中数据“0000E”是该对话框窗口的 ID 号 (十六进制表示)，该 ID 号的十进制为 14，用资源工具查看，可知其是记事本的“页面设置”对话框。双击“Dialog: DialogID_000E”光条将跳到相关代码上。如多处调用，双击这个文本时，光条将在调用它的几个位置代码处循环。

 注意：在软件解密过程中，将资源查看工具与 W32Dasm 对话框参考配合，能快速定位到

调用对话框的代码处。

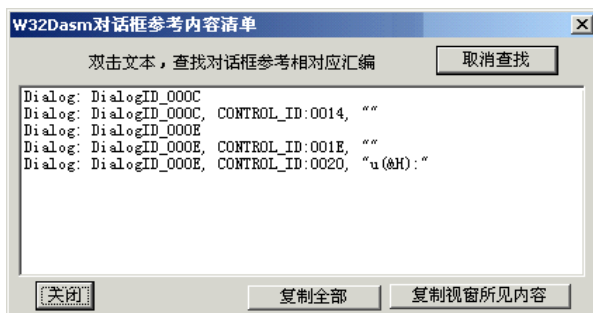



图 3.13 对话框参考窗口

15. 菜单参考 (Menu References)

菜单参考列出程序菜单的相关信息。按  按钮或在“参考 (References)”菜单中选择“菜单参考 (Menu References)”命令，打开菜单参考窗口。操作与对话框参考类似。

16. 复制汇编代码文本

W32Dasm 允许打印或复制指定行的汇编代码。首先将鼠标移到 W32Dasm 的最左边并单击，将会出现一个小红点，然后按住 Shift 键，移到需要的下一行，再单击鼠标，将选中一段。按 Ctrl+C 键或在“反汇编 (Disassembler)”菜单里选择“复制指定的行 (Copy Lines of Text)”功能或按  按钮，将把数据复制到剪贴板里，如图 3.14 所示。

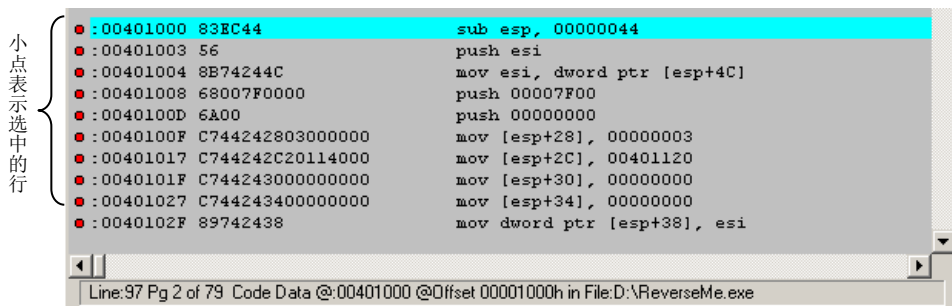


图 3.14 选定需要复制的行

17. 动态调试

W32Dasm 不仅是一款出色的静态分析工具，同时也具有动态分析功能，但在关于动态分析技术的一章中将有更优秀的工具，所以此处不再介绍 W32Dasm 动态调试功能。