

**汇编语言与逆向技术实验报告**

**实验六：读取PE文件的输入表和输出表**

****

学 院 网络空间安全学院

专 业 信息安全

学 号 2212046

姓 名 王昱

班 级 信息安全班

1. **实验目的**

熟悉PE文件的输入表和输出表结构；

1. **实验原理**
   * 1. **输入表**

在 PE文件头的 IMAGE\_OPTIONAL\_HEADER 结构中的 DataDirectory(数据目录表) 的第二个成员就是指向输入表。每个被链接进来的 DLL文件都分别对应一个 IMAGE\_IMPORT\_DESCRIPTOR (简称IID) 数组结构。

* + 1. **输出表**

在 PE文件头的 IMAGE\_OPTIONAL\_HEADER 结构中的DataDirectory(数据目录表) 的第一个成员就是指向输出表。输出表是用来描述模块中导出函数的数据结构。如果一个模块导出了函数，那么这个函数会被记录在输出表中。

1. **实验环境**

Windows操作系统，MASM32编译环境。

1. **实验过程**

**1.描述PE文件的输入表的作用和数据结构**

PE文件的输入表是用来保存程序调用的外部函数的信息的。这些函数在程序代码中没有定义，而是在外部的DLL文件中定义。输入表包含了被调用函数的名称、所在的DLL文件名、函数的入口地址等信息。输入表的数据结构是由IMAGE\_IMPORT\_DESCRIPTOR、IMAGE\_THUNK\_DATA32和IMAGE\_IMPORT\_BY\_NAME三个结构体组成的。其中，IMAGE\_IMPORT\_DESCRIPTOR结构体包含了指向IMAGE\_THUNK\_DATA32结构体的指针，IMAGE\_THUNK\_DATA32结构体包含了指向IMAGE\_IMPORT\_BY\_NAME结构体的指针，IMAGE\_IMPORT\_BY\_NAME结构体包含了被调用函数的名称。

**2.描述PE文件的输出表的作用和数据结构**

导出表的主要成分是一个表格，内含函数名称，输出序数等。序数是指定DLL中的某个函数的16位数字，在所指的DLL文件是独一无二的。输出表可以用来保存程序导出函数的信息，这些函数在程序代码中被定义，而是在外部的DLL文件中被调用。输出表包含了导出函数的名称、序号、所在的DLL文件名、函数的入口地址等信息。输出表的数据结构是由IMAGE\_EXPORT\_DIRECTORY、IMAGE\_EXPORT\_ADDRESS和IMAGE\_EXPORT\_NAME三个结构体组成的。其中，IMAGE\_EXPORT\_DIRECTORY结构体包含了指向IMAGE\_EXPORT\_ADDRESS结构体和IMAGE\_EXPORT\_NAME结构体的指针，IMAGE\_EXPORT\_ADDRESS结构体包含了导出函数的地址，IMAGE\_EXPORT\_NAME结构体包含了导出函数的名称。

**3.程序的汇编语言源代码和注释**

**.386 ;**

**.model flat, stdcall ;**

**option casemap :none ;**

**include \masm32\include\windows.inc ;**

**include \masm32\include\kernel32.inc ;**

**include \masm32\include\masm32.inc ;**

**includelib \masm32\lib\kernel32.lib ;**

**includelib \masm32\lib\masm32.lib ;**

**.data ;数据段，用来定义变量和常量**

**input BYTE "Please input a PE file:",0 ;提示输入**

**file DWORD 0,0 ;文件句柄**

**buffer DWORD 1000000 DUP(0) ;文件内容**

**name BYTE 100 DUP(0) ;文件名**

**import BYTE "Import table:",0 ;输出导入表**

**export BYTE "Export table:",0 ;输出导出表**

**imhint BYTE "NO IMPORT TABLE",0 ;提示没有导入表**

**exhint BYTE "NO EXPORT TABLE",0 ;提示没有导出表**

**retract1 BYTE " ",0 ;输出缩进**

**retract2 BYTE " ",0 ;输出缩进**

**va DWORD 0 ;虚拟地址**

**rva DWORD 0 ;相对虚拟地址**

**sec DWORD 0 ;节表地址**

**nt DWORD 0 ;NT头地址**

**exp DWORD 0 ;导出表地址**

**imp DWORD 0 ;导入表地址**

**temp DWORD 0**

**temp1 DWORD 0**

**temp2 DWORD 0**

**temp3 DWORD 0**

**endl BYTE 0Ah,0Dh,0;换行符**

**.code**

**Import PROC ;导入表过程**

**mov esi,OFFSET buffer**

**add esi,sec**

**add esi,0ch**

**mov eax,DWORD PTR[esi]**

**cmp imp,0**

**je L7 ;如果没有导入表，跳转到L7**

**L8:**

**add esi,028h**

**mov ecx,DWORD PTR[esi]**

**cmp ecx,0**

**je L9 ;如果遍历完所有节，跳转到L9**

**cmp ecx,imp**

**ja L9 ;如果当前节的虚拟地址大于导入表地址，跳转到L9**

**jbe L8 ;如果当前节的虚拟地址小于等于导入表地址，继续遍历**

**L9:**

**sub esi,028h ;回退到包含导入表的节**

**mov ecx,DWORD PTR[esi]**

**mov va,ecx ;保存当前节的虚拟地址**

**add esi,08h**

**mov edx,DWORD PTR[esi]**

**mov rva,edx ;保存当前节的相对虚拟地址**

**mov esi,OFFSET buffer**

**mov edx,imp**

**sub edx,va**

**add edx,rva ;计算导入表在文件中的偏移量**

**add esi,edx ;定位到导入表**

**L13:**

**add esi,0ch**

**mov edi,DWORD PTR[esi]**

**sub edi,va**

**add edi,rva ;计算导入模块名称在文件中的偏移量**

**add edi,OFFSET buffer ;定位到导入模块名称**

**mov temp1,edi**

**invoke StdOut,addr retract1 ;输出缩进**

**mov edi,temp1**

**invoke StdOut,edi ;输出导入模块名称**

**invoke StdOut,addr endl ;输出换行**

**sub esi,0ch**

**mov edx,DWORD PTR[esi]**

**sub edx,va**

**add edx,rva ;计算导入函数表在文件中的偏移量**

**add edx,OFFSET buffer ;定位到导入函数表**

**mov ecx,DWORD PTR[edx]**

**mov temp,edx**

**L12:**

**sub ecx,va**

**add ecx,rva ;计算导入函数名称在文件中的偏移量**

**add ecx,OFFSET buffer**

**add ecx,02h ;定位到导入函数名称**

**mov temp2,ecx**

**invoke StdOut,addr retract2 ;输出缩进**

**mov ecx,temp2**

**invoke StdOut,ecx ;输出导入函数名称**

**invoke StdOut,addr endl ;输出换行**

**mov edx,temp**

**add edx,04h ;移动到下一个导入函数**

**mov temp,edx**

**mov ecx,DWORD PTR[edx]**

**cmp ecx,0**

**jne L12 ;如果不为0，继续遍历**

**add esi,014h ;移动到下一个导入模块**

**mov edi,DWORD PTR[esi]**

**cmp edi,0**

**jne L13 ;如果不为0，继续遍历**

**L7:**

**ret**

**Import ENDP**

**Export PROC ;导出表过程**

**mov esi,OFFSET buffer**

**add esi,sec**

**add esi,0ch**

**mov eax,DWORD PTR[esi]**

**cmp exp,0**

**je L3 ;如果没有导出表，跳转到L3**

**L1:**

**add esi,028h**

**mov ecx,DWORD PTR[esi]**

**cmp ecx,0**

**je L5 ;如果遍历完所有节，跳转到L5**

**cmp ecx,exp**

**ja L5 ;如果当前节的虚拟地址大于导出表地址，跳转到L5**

**jbe L1 ;如果当前节的虚拟地址小于等于导出表地址，继续遍历**

**L5:**

**sub esi,028h ;回退到包含导出表的节**

**mov ecx,DWORD PTR[esi]**

**mov va,ecx ;保存当前节的虚拟地址**

**add esi,08h**

**mov edx,DWORD PTR[esi]**

**mov rva,edx ;保存当前节的相对虚拟地址**

**mov esi,OFFSET buffer**

**mov edx,exp**

**sub edx,va**

**add edx,rva ;计算导出表在文件中的偏移量**

**add esi,edx ;定位到导出表**

**add esi,020h**

**mov edi,DWORD PTR[esi]**

**mov temp,edi**

**mov eax,temp**

**add eax,rva**

**sub eax,va ;计算导出模块名称在文件中的偏移量**

**mov temp,eax**

**mov esi,OFFSET buffer**

**add esi,temp ;定位到导出模块名称**

**mov ecx,DWORD PTR[esi]**

**sub ecx,va**

**add ecx,rva ;计算导出函数表在文件中的偏移量**

**mov temp,ecx**

**mov esi,OFFSET buffer**

**add esi,temp ;定位到导出函数表**

**L2:**

**mov temp3,esi**

**invoke StdOut,addr retract1 ;输出缩进**

**mov esi,temp3**

**invoke StdOut,esi ;输出导出函数名称**

**invoke StdOut,addr endl ;输出换行**

**L4:**

**inc esi ;移动到下一个字节**

**mov eax,0**

**mov al,BYTE PTR[esi] ;读取当前字节**

**cmp al,0**

**jne L4 ;如果不为0，继续移动**

**inc esi ;移动到下一个字节**

**mov al,BYTE PTR[esi] ;读取当前字节**

**cmp al,0**

**jne L2 ;如果不为0，继续遍历**

**L3:**

**ret**

**Export ENDP**

**start: ;开始过程**

**invoke StdOut,addr input;输出提示**

**invoke StdIn,addr name,100;输入文件名**

**invoke CreateFile,addr name,GENERIC\_READ,FILE\_SHARE\_READ,0,OPEN\_EXISTING,FILE\_ATTRIBUTE\_ARCHIVE,0 ;打开文件**

**mov file,eax;保存文件句柄**

**invoke SetFilePointer,file,0,0,FILE\_BEGIN;设置文件指针**

**invoke ReadFile,file,addr buffer,1000000,0,0;读取文件内容**

**mov esi,OFFSET buffer**

**mov eax,DWORD PTR[esi+3ch] ;获取NT头地址**

**mov nt,eax**

**add esi,nt**

**mov ebx,0**

**mov bx,WORD PTR[esi+14h] ;获取节表数量**

**add ebx,nt**

**add ebx,18h ;获取节表地址**

**mov sec,ebx**

**mov esi,OFFSET buffer**

**mov edx,sec**

**sub edx,080h ;获取数据目录地址**

**add esi,edx**

**mov eax,DWORD PTR[esi] ;获取导出表地址**

**mov exp,eax**

**add esi,08h**

**mov edx,DWORD PTR[esi] ;获取导入表地址**

**mov imp,edx**

**invoke StdOut,addr import ;输出导入表**

**invoke StdOut,addr endl**

**call Import ;调用导入表过程**

**invoke StdOut,addr endl**

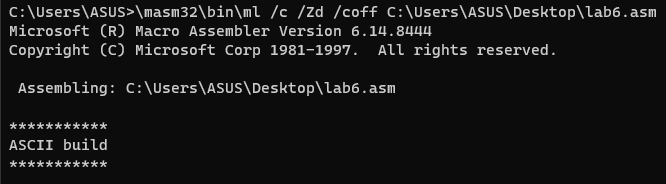
**call Export ;调用导出表过程**

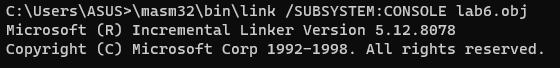
**invoke CloseHandle,hfile ;关闭文件句柄，释放资源**

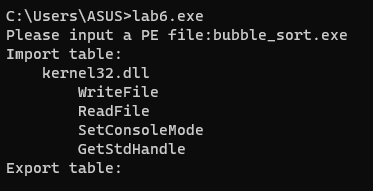
**invoke ExitProcess,0 ;退出进程，返回0**

**End start ;**

**4.程序测试输出结果的截图**

****

****

****

实验结果正确

**5.讨论输出表的安全问题**

输出表的安全问题主要这些：

(1)输出表可能被恶意修改，导致原本的导出函数被替换为恶意代码，从而影响其他依赖该文件的程序的正常运行或安全性。

(2)输出表可能被恶意隐藏，导致分析工具无法正确识别该文件的导出函数，从而难以进行逆向工程或恶意代码检测。

(3)输出表可能被恶意增加，导致该文件导出一些不必要或危险的函数，从而增加该文件的攻击面或被其他恶意程序利用。

(4)输出表可能被恶意删除，导致该文件无法提供任何导出函数，从而破坏该文件的功能或与其他程序的兼容性。

可以根据这些方法来对输出表进行完整性校验：检查输出表是否被修改或损坏，如果发现异常，及时报警或恢复。对输出表进行加密或混淆，使得输出表的内容不容易被分析或修改，增加恶意代码的难度和成本。

**五、实验结论及心得体会**

这个实验是一个关于PE文件的导入表和导出表的分析实验。PE文件是一种可执行文件格式，包含了一些重要的部分，如NT头、节表、数据目录、导入表、导出表等，它们分别记录了该文件的基本信息、节的信息、数据的信息、依赖的外部函数、提供的内部函数等。通过使用MASM32汇编语言，可以编写一个可以读取和输出PE文件的导入表和导出表的信息的程序，该程序的主要步骤是：打开文件，读取文件内容，获取NT头地址，节表地址，导入表地址和导出表地址，遍历导入表，输出导入模块名称和导入函数名称，遍历导出表，输出导出函数名称，关闭文件，退出程序。

通过这个实验，我遇到了很多问题并且不断解决问题，比如32位/64位文件结构差异等问题，在解决问题的过程中收获了许多知识以及技巧。在本次汇编实验中，我学习了PE文件的结构和功能，MASM32汇编语言的基本语法和指令，我对汇编语言的编程有了更深的理解和掌握。同时，我也锻炼了我的分析能力和编程能力，还感受到了汇编语言的魅力和挑战，我发现汇编语言虽然简洁和直接，但是也非常复杂和细致，它让我对硬件有了更大的控制，也让我对软件有了更高的要求。