南开大学

PE Viewer

（汇编语言与逆向技术实验 5）

姓名：申宗尚

学号：2213924

专业：信息安全

一．实验目的

1、熟悉 PE 文件结构；

2、使用 Windows API 函数读取文件内容

二．实验环境

Windows 记事本的汇编语言编写环境

MASM32 编译环境

Windows 命令行窗口

三．实验原理

（1）PE 文件结构

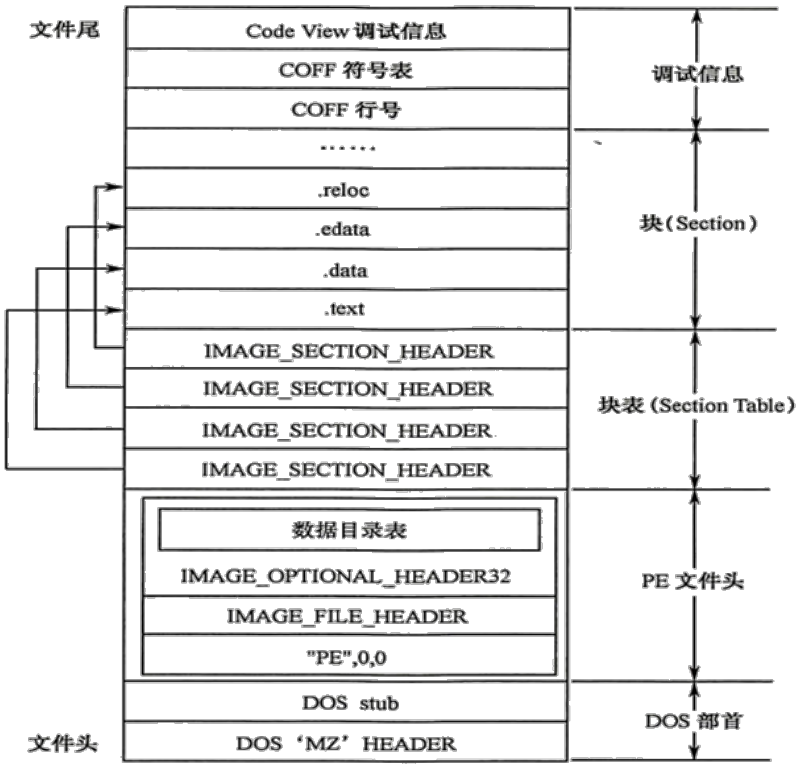


图 1 PE 文件结构

PE 文件结构如图 1 所示。二进制 PE 文件包括：DOS 部首、PE 文件头、块表（Section Table）、块（Section）、调试信息 5 个部分。

DOS 部首是 DOS 系统的残留内容，目的是防止 Windows 系统的可执行程序在 DOS 系统上执行时导致 DOS 系统崩溃。DOS 部首是一

段 DOS 程序，输出一段提示信息，说明程序只能运行在 Windows 系统上，不能运行在 DOS 系统上。

```
IMAGE_DOS_HEADER_STRUCT{
  +0h    e_magic      WORD  ?          ;DOS 可执行文件标记 "MZ"
  +2h    e_cblp       WORD  ?
  +4h    e_cp         WORD  ?
  +6h    e_crlc       WORD  ?
  +8h    e_cparhdr    WORD  ?
  +0ah   e_minalloc   WORD  ?
  +0ch   e_maxalloc   WORD  ?
  +0eh   e_ss         WORD  ?
  +10h   e_sp         WORD  ?
  +12h   e_csum       WORD  ?
  +14h   e_ip         WORD  ?          ;DOS 代码入口 IP
  +16h   e_cs         WORD  ?          ;DOS 代码入口 CS
  +18h   e_lfarlc     WORD  ?
  +1ah   e_ovno       WORD  ?
  +1ch   e_res        WORD  4 dup(?)
  +24h   e_oemid      WORD  ?
  +26h   e_oeminfo    WORD  ?
  +28h   e_res2       WORD 10 dup(?)
  +3ch   e_lfanew     DWORD ?          ;指向 PE 文件头"PE",0,0
} IMAGE_DOS_HEADER_ENDS
```

图 2 DOS 头的数据结构

PE 文件头记录了各种文件的装载信息，有映像的基地址（ImageBase）、程序的入口地址（EntryPoint）、数据块、编译时间、运行平台、数据目录表等信息。PE 文件头包括 Signature、FileHeader、OptionalHeader 三部分，数据结构如下所示：

IMAGE_NT_HEADERS STRUCT

+0h Signature DWORD ?

+4h FileHeader IMAGE_FILE_HEADER <>

+18h OptionalHeader IMAGE_OPTIONAL_HEADER32 <>

IMAGE_NT_HEADERS ENDS

Signature 的定义是 IMAGE_NT_HEADER，值为 00004550h

FileHeader 的数据结构如下所示：

IMAGE_FILE_HEADER STRUCT

+04h Machine	WORD	?
+06h NumberOfSections	WORD	?
+08h TimeDateStamp	DWORD	?
+0Ch PointerToSymbolTable	DWORD	?
+10h NumberOfSymbols	DWORD	?
+14h SizeOfOptionalHeader	WORD	?
+16h Characteristics	WORD	?

OptionalHeader 的数据结构如下所示：

IMAGE_OPTIONAL_HEADER STRUCT

+18h	Magic	WORD	?
+1Ah	MajorLinkerVersion	BYTE	?
+1Bh	MinorLinkerVersion	BYTE	?
+1Ch	SizeOfCode	DWORD	?
+20h	SizeOfInitializedData	DWORD	?
+24h	SizeOfUninitializedData	DWORD	?
+28h	AddressOfEntryPoint	DWORD	?
+2Ch	BaseOfCode	DWORD	?
+30h	BaseOfData	DWORD	?
+34h	ImageBase	DWORD	?
+38h	SectionAlignment	DWORD	?
+3Ch	FileAlignment	DWORD	?

```
+40h    MajorOperatingSystemVersion    WORD ?
+42h    MinorOperatingSystemVersion    WORD ?
+44h    MajorImageVersion              WORD ?
+46h    MinorImageVersion              WORD ?
+48h    MajorSubsystemVersion          WORD ?
+4Ah    MinorSubsystemVersion          WORD ?
+4Ch    Win32VersionValue              DWORD ?
+50h    SizeOfImage                    DWORD ?
+54h    SizeOfHeaders                  DWORD ?
+58h    CheckSum                       DWORD ?
+5Ch    Subsystem                      WORD ?
+5Eh    DllCharacteristics             WORD ?
+60h    SizeOfStackReserve             DWORD ?
+64h    SizeOfStackCommit              DWORD ?
+68h    SizeOfHeapReserve              DWORD ?
+6Ch    SizeOfHeapCommit               DWORD ?
+70h    LoaderFlags                    DWORD ?
+74h    NumberOfRvaAndSizes            DWORD ?
+78h    DataDirectory
[IMAGE_NUMBEROF_DIRECTORY_ENTRIES] IMAGE_DATA_DIRECTORY <>
    IMAGE_OPTIONAL_HEADER ENDS
```

块表（Section Table）描述代码块、数据块、资源块等不同数据块

的文件和内存的映射，数据块的各种属性。

块（Section）分别存储了程序的代码、数据、资源等信息。

（2） Windows 文件读操作

读一个文件用到的 Windows API 函数有 CreateFile、SetFilePointer、ReadFile、CloseHandle。

CreateFile 的 MSDN 文档地址 https://docs.microsoft.com/en-us/previous-versions/aa914735(v=msdn.10)，函数的原型如下：

```
HANDLE CreateFile(
  LPCTSTR lpFileName,
  DWORD dwDesiredAccess,
  DWORD dwShareMode,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  DWORD dwCreationDisposition,
  DWORD dwFlagsAndAttributes,
  HANDLE hTemplateFile
    );
```

CreateFile 在 MASM 汇编语言中的应用实例如图 3 所示。

```
invoke CreateFile,ADDR buf2,\
                  GENERIC_READ,\
                  FILE_SHARE_READ,\
                  0,\
                  OPEN_EXISTING,\
                  FILE_ATTRIBUTE_ARCHIVE,\
                  0
MOV hfile, EAX
invoke SetFilePointer, hfile, 0, 0, FILE_BEGIN
invoke ReadFile, hfile, ADDR buf3, 4000, 0, 0
MOV EAX, DWORD PTR buf3
invoke dw2hex, EAX, ADDR buf4
invoke StdOut, ADDR buf4
invoke CloseHandle, hfile
```

图 3 MASM 汇编中调用 CreateFile、SetFilePointer、ReadFile、CloseHandle 的示例

SetFilePointer 函数的 MSDN 文档地址 https://docs.microsoft.com/en-us/previous-versions/aa911934(v%3dmsdn.10)，函数原型如下：

```
DWORD SetFilePointer(
```

```
    HANDLE hFile,
    LONG lDistanceToMove,
    PLONG lpDistanceToMoveHigh,
    DWORD dwMoveMethod
      );
```

SetFilePointer 函数的 MASM 调用示例如图 3 所示。

ReadFile 函数的 MSDN 文档地址 https://docs.microsoft.com/en-us/previous-versions/aa914377(v%3dmsdn.10)，函数原型如下：

```
BOOL ReadFile(
    HANDLE hFile,
    LPVOID lpBuffer,
    DWORD nNumberOfBytesToRead,
    LPDWORD lpNumberOfBytesRead,
    LPOVERLAPPED lpOverlapped
);
```

ReadFile 函数的 MASM 调用示例如图 3 所示。

CloseHandle 函数的 MSDN 文档地址 https://docs.microsoft.com/en-us/previous-versions/aa914720(v%3dmsdn.10)，函数原型如下：

```
BOOL CloseHandle(
    HANDLE hObject
);
```

CloseHandle 函数的 MASM 调用示例如图 3 所示。

四．实验内容

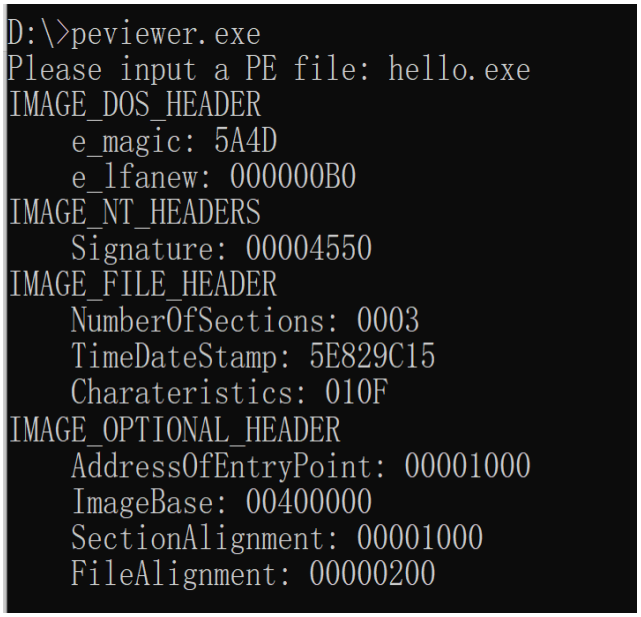


图 4 peviewer 实验演示

（1）输入 PE 文件的文件名，peviewer 程序调用 Windows API 函数，打开指定的 PE 文件；

（2）从文件的头部开始，读取 IMAGE_DOS_HEADER 结构中的 e_magic 和 e_lfanew 字段的值，按照实验演示的方式输出到命令行窗口；

（3）继续读取 PE 文件的 IMAGE_NT_HEADER 结构中的 Signature 字段的值，按照实验演示的方式输出到命令行窗口；

（4）继续读取 IMAGE_NT_HEADER 结构中的 IMAGE_FILE_HEADER 结构，从中读取出字段 NumberOfSections、TimeDateStamp、Characteristics 的值，按照实验演示的方式输出到命令行窗口；

（5）继续读取 IMAGE_NT_HEADER 结构中的 IMAGE_OPTIONAL_HEADER 结构，从中读取字段

AddressOfEntryPoint、ImageBase、SectionAlignment、FileAlignment 的值，按照实验演示的方式输出到命令行窗口；

五、实验程序调试及过程说明

1. 编辑：编写汇编程序 peviewer.asm。

- 流程：

1.输入 PE 文件的文件名，peviewer 程序调用 Windows API 函数，打开指定的 PE 文件；

2.从文件的头部开始，读取 IMAGE_DOS_HEADER 结构中的 e_magic 和 e_lfanew 字段的值，按照实验演示的方式输出到命令行窗口；

3.继续读取 PE 文件的 IMAGE_NT_HEADER 结构中的 Signature 字段的值，按照实验演示的方式输出到命令行窗口；

4.继续读取 IMAGE_NT_HEADER 结构中的 IMAGE_FILE_HEADER 结构，从中读取出字段 NumberOfSections、TimeDateStamp、Characteristics 的值，按照实验演示的方式输出到命令行窗口；

5.继续读取 IMAGE_NT_HEADER 结构中的 IMAGE_OPTIONAL_HEADER 结构，从中读取字段 AddressOfEntryPoint、ImageBase、SectionAlignment、FileAlignment 的值，按照实验演示的方式输出到命令行窗口；

- 设计说明：

1.调用 CreateFile 函数，利用用户输入的文件名打开相应文件；

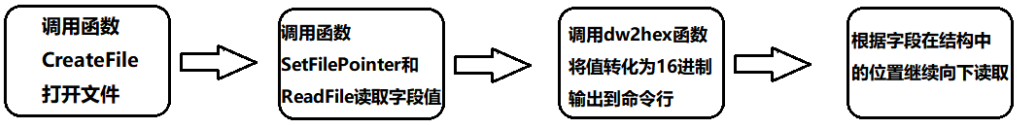
2.从文件的头部开始，调用 SetFilePointer 和 ReadFile 函数读取相应字段的值；

3.将读取出来的值通过 dw2hex 函数转化为 16 进制输出到命令行；

4.根据文件中其他字段与 e_magic 和 e_lfanew 的结构关系依次读取 Signature、NumberOfSections、TimeDateStamp、Characteristics、AddressOfEntryPoint、ImageBase、SectionAlignment、FileAlignment 的值并转化为 16 进制输出到命令行；

5.关闭。

- 控制流图：



2. 编译：使用 ml 将 peviewer.asm 文件汇编到 peviewer.obj 目标文件。

编译命令：“\masm32\bin\ml /c /Zd /coff peviewer.asm”

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19045.3448]
(c) Microsoft Corporation。保留所有权利。

C:\Users\KKkai>\masm32\bin\ml /c /Zd /coff C:\Users\KKkai\Desktop\pe.asm
Microsoft (R) Macro Assembler Version 6.14.8444
Copyright (C) Microsoft Corp 1981-1997.  All rights reserved.

 Assembling: C:\Users\KKkai\Desktop\pe.asm

************
ASCII build
************
```

3. 链接：使用 link 将目标文件 peviewer.obj 链接成 peviewer.exe 可执行文件。

链接命令：“\masm32\bin\Link /SUBSYSTEM:CONSOLE peviewer.obj”

```
C:\Users\KKkai>\masm32\bin\Link /SUBSYSTEM:CONSOLE pe.obj
Microsoft (R) Incremental Linker Version 5.12.8078
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.
```

4. 测试：直接执行peviewer.exe可执行文件。(以bubble.exe与dec2hex.exe为例)

```
C:\Users\KKkai>pe.exe
Please input a PE file :C:\Users\KKkai\Desktop\bubble.exe
C:\Users\KKkai\Desktop\bubble.exe
IMAGE_DOS_HEADER
   e_magic: 5A4D
   e_lfanew: 000000C0
IMAGE_NT_HEADERS
   Signature: 00004550
IMAGE_FILE_HEADER
   NumberOfSections: 0003
   TimeDateStamp: 654CD932
   Charateristics: 010F
IMAGE_OPTIONAL_HEADER
   AddressOfEntryPoint: 00001000
   ImageBase: 00400000
   SectionAlignment: 00001000
   FileAlignment: 00000200
C:\Users\KKkai>_
```

```
C:\Users\KKkai>pe.exe
Please input a PE file :C:\Users\KKkai\dec2hex.exe
C:\Users\KKkai\dec2hex.exe
IMAGE_DOS_HEADER
   e_magic: 5A4D
   e_lfanew: 000000C0
IMAGE_NT_HEADERS
   Signature: 00004550
IMAGE_FILE_HEADER
   NumberOfSections: 0003
   TimeDateStamp: 653A2818
   Charateristics: 010F
IMAGE_OPTIONAL_HEADER
   AddressOfEntryPoint: 00001000
   ImageBase: 00400000
   SectionAlignment: 00001000
   FileAlignment: 00000200
C:\Users\KKkai>
```

六、 实验源代码及注释解释

```
.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\masm32.lib
includelib \masm32\lib\kernel32.lib

.data
    output db 100 DUP(0)     ;最后输出的字符串
    fileName db 100 DUP(0)   ;文件名
    hFile HANDLE 0
    content db 4000 DUP(0)
    e_lfnew dd 0

    str1 db "Please input a PE file :",0
    str2 db 0Ah,"IMAGE_DOS_HEADER",0Ah,"   e_magic: ",0
    str3 db 0Ah,"   e_lfanew: ",0
    str4 db 0Ah,"IMAGE_NT_HEADERS",0Ah,"   Signature: ",0
    str5 db 0Ah,"IMAGE_FILE_HEADER",0Ah,"   Number0fSections: ",0
    str6 db 0Ah,"   TimeDateStamp: ",0
    str7 db 0Ah,"   Charateristics: ",0
    str8 db 0Ah,"IMAGE_OPTIONAL_HEADER",0Ah,"   Address0fEntryPoint: ",0
    str9 db 0Ah,"   ImageBase: ",0
    str10 db 0Ah,"   SectionAlignment: ",0
    str11 db 0Ah,"   FileAlignment: ",0
```

```
.code
start:
    invoke StdOut, ADDR str1
    invoke StdIn, ADDR fileName, 100
    invoke StdOut, ADDR fileName

    ;CreateFile
    invoke CreateFile, ADDR fileName,\
                       GENERIC_READ,\
                       FILE_SHARE_READ,\
                       0,\
                       OPEN_EXISTING,\
                       FILE_ATTRIBUTE_ARCHIVE,\
                       0

    ;SetFilePointer 和 ReadFile
    mov hFile, eax
    invoke SetFilePointer, hFile,\
                           0,\
                           0,\
                           FILE_BEGIN
    invoke ReadFile, hFile,\
                     ADDR content,\
                     4000,\
                     0,\
                     0


`
    mov eax, 0
    mov ax, WORD PTR content
    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str2
    invoke StdOut, ADDR output+4

    ;输出 e_lfanew
    mov eax, 0
    mov eax, DWORD PTR [content+3ch]
    mov e_lfnew, eax
    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str3
    invoke StdOut, ADDR output
```

```
;输出 Signature
lea ebx, content
add ebx, e_lfnew
mov eax, 0
mov eax, DWORD PTR [ebx]
invoke dw2hex, eax, ADDR output

invoke StdOut, ADDR str4
invoke StdOut, ADDR output

;输出 Number0fSections
lea ebx, content
add ebx, e_lfnew
mov eax, 0
mov ax, WORD PTR [ebx+6h]
invoke dw2hex, eax, ADDR output

invoke StdOut, ADDR str5
invoke StdOut, ADDR output+4

;输出 TimeDateStamp
lea ebx, content
add ebx, e_lfnew
mov eax, 0
mov eax, DWORD PTR [ebx+8h]
invoke dw2hex, eax, ADDR output

invoke StdOut, ADDR str6
invoke StdOut, ADDR output

;输出 Charateristics
lea ebx, content
add ebx, e_lfnew
mov eax, 0
mov ax, WORD PTR [ebx+16h]
invoke dw2hex, eax, ADDR output

invoke StdOut, ADDR str7
invoke StdOut, ADDR output+4

;输出 AddressOfEntryPoint
lea ebx, content
add ebx, e_lfnew
```

```
    mov eax, 0
    mov eax, DWORD PTR [ebx+28h]
    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str8
    invoke StdOut, ADDR output

    ;输出 ImageBase
    lea ebx, content
    add ebx, e_lfnew
    mov eax, 0
    mov eax, DWORD PTR [ebx+34h]
    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str9
    invoke StdOut, ADDR output

    ;输出 SectionAlignment
    lea ebx, content
    add ebx, e_lfnew
    mov eax, 0
    mov eax, DWORD PTR [ebx+38h]
    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str10
    invoke StdOut, ADDR output

    ;输出 FileAlignment
    lea ebx, content
    add ebx, e_lfnew
    mov eax, 0
    mov eax, DWORD PTR [ebx+3Ch]
    invoke dw2hex, eax, ADDR output

    invoke StdOut, ADDR str11
    invoke StdOut, ADDR output

    ;调用函数 CloseHandle 关闭句柄
    invoke CloseHandle, hFile
    invoke ExitProcess, 0
END start
```