南開大學

汇编语言与逆向技术课程实验报告

实验五： PEViewer

学　　院　网络空间安全学院
专　　业　信息安全、法学双学位
学　　号　2212000
姓　　名　宋奕纬
班　　级　1061

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、实验原理

（设计说明与控制流图放在最后一部分）

1、实验环境：MASM32 编译环境、Windows 命令行窗口

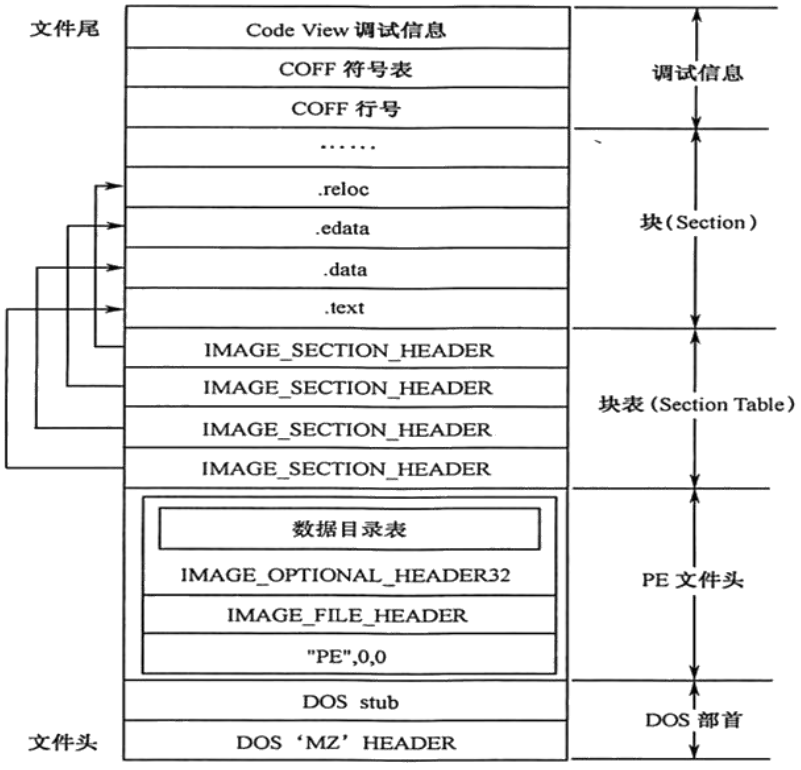
2、总体思路：

总体思路：读取——转化——输出——变址——读取

（1） 输出提示语
（2） 读取 PE 文件信息
（3） 转化为十六进制数字符串
（4） 输出 PE 文件信息
（5） 更改地址（下一个文件信息）

3、实验原理：

（1）PE 文件结构



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
```

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_HEADER3
IMAGE_NT_HEADERS ENDS
```

①**Signature** 的定义是 IMAGE_NT_HEADER，值为 00004550h

②**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    ?
```

③**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
```

④**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。

①CreateFile 函数的原型如下：

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

②SetFilePointe 函数原型如下：

```
DWORD SetFilePointer(
  HANDLE hFile,
  LONG lDistanceToMove,
  PLONG lpDistanceToMoveHigh,
  DWORD dwMoveMethod
```

);

③ReadFile 函数原型如下：

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

④CloseHandle 函数原型如下：

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

以上函数的 MASM 调用示例如所示。

```
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
```

三、实验过程

1、指定处理器、指令集，指定内存模式，引入头文件，链接静态库文件。

```
.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
```

2、定义数据段，进行相关数据的定义、声名和初始化。

```
.data
hi BYTE "Please input a PE file:",0;定义“输入pe文件”的提示语
filename BYTE 20 DUP(0);存储输入的PE文件名
hfile DWORD 0;存储文件句柄
buf5 DWORD 4000 dup(?);存储从文件中接收的数据（未初始化）
buf4 DWORD 4000 dup(0);存储数据（初始化为0）
buf3 word 4000 dup(0) ;存储数据（初始化为0）

;定义输出中显示各个头部元素的提示与标志
IDH BYTE "IMAGE_DOS_HEADER",0
em BYTE "   e_magic:  ",0
el BYTE "   e_lfanew:  ",0

INH BYTE "IMAGE_NT_HEADERS",0
sig BYTE  "   Signature:  ",0

IFH BYTE "IMAGE_FILE_HEADERS",0
nos BYTE "   NumberOfSections:  ",0
tds BYTE "   TimeDateStamp:  ",0
chc BYTE "   Characteics:  ",0

IOH BYTE "IMAGE_OPTIONAL_HEADER",0
aop BYTE "   AddressOfEntryPoint:  ",0
ib BYTE "   ImageBase:  ",0
sa BYTE "   SectionAligment:  ",0
fa BYTE "   FILEAligment:  ",0

endl BYTE 0Ah,0Dh,0;换行（0Ah表示换行，0Dh表示回车）
```

3、代码段

```
.code
main PROC
    invoke StdOut,addr hi; 输出提示信息
    invoke StdIn, addr filename,20 ;读取用户输入的PE文件名，并将其保存
                                    在filename变量中
    ; 调用CreateFile函数打开指定文件，返回文件句柄
    invoke CreateFile,   addr filename,\
            GENERIC_READ,\
            FILE_SHARE_READ,\
            0,\
            OPEN_EXISTING,\
            FILE_ATTRIBUTE_ARCHIVE,\
```

```
        0
mov hfile, eax ; 将文件句柄保存在hfile变量中
invoke SetFilePointer, hfile,0 ,0,FILE_BEGIN; 调用SetFilePointer函
                                              数将文件指针移动到文
                                              件开头
invoke ReadFile, hfile, addr buf3,400 , 0 ,0; 读取PE文件的头部信息
                                              到buf3数组中
mov esi, offset buf3    ; 将buf3数组的首地址保存在esi寄存器中

; -----------------IMAGE_DOS_HEADER---------------------------------
invoke StdOut,addr IDH  ; 输出MS-DOS头的名称
invoke StdOut,addr endl ; 换行
invoke StdOut,addr em   ; 输出e_magic元素的名称

mov eax, dword PTR [esi]
invoke dw2hex, eax, addr buf4; 读取e_magic元素的值，并将其转换为16
                               进制字符串，保存在buf4中
mov ax,word PTR[buf4+4]
mov buf5,ax
invoke StdOut,addr buf5      ; 将buf5中的值输出
mov ax,word PTR[buf4+6]
mov buf5,ax
invoke StdOut,addr buf5      ; 将buf5中的值输出
invoke StdOut,addr endl      ; 换行

invoke StdOut,addr el        ;输出e_lfanew元素的名称
add esi,3ch                  ;指到e_lfanew
mov eax,dword PTR[esi]
invoke dw2hex,eax,addr buf4  ;读取e_lfanew元素的值，并将其转换为16
                              进制字符串，保存在buf4中

invoke StdOut,addr buf4      ;将buf4中的值输出
invoke StdOut,addr endl      ; 换行

;--------------------- IMAGE_NT_HEADERS-----------------------------
invoke StdOut, addr INH      ; 输出NT头的名称
invoke StdOut, addr endl     ; 换行
invoke StdOut, addr sig      ; 输出Signature元素的名称
mov edx,dword PTR [esi]      ; 将相对于文件开头的NT头的偏移量保存在
                               edx寄存器
mov esi,offset buf3          ; esi重新指向buf3数组的首地址
add esi, edx                 ; 将esi指向Signature元素的值的位置
mov eax, dword PTR[esi]
invoke dw2hex, eax,addr buf4 ; 读取Signature元素的值，并将其转换为
```

```
                                    16进制字符串，保存在buf4中
    invoke StdOut, addr buf4      ; 将buf4中的值输出到屏幕上
    invoke StdOut, addr endl      ; 换行
    invoke StdOut, addr endl      ; 换行

;-------------------------------- IMAGE_FILE_HEADERS----------------
    invoke StdOut, addr IFH       ; 输出文件头名称
    invoke StdOut, addr endl      ; 换行
    invoke StdOut, addr nos       ; 输出NumberOfSections元素的名称
    add esi,6h                    ; 将esi指向FileHeader元素的位置
    mov eax,dword PTR[esi]
    invoke dw2hex, eax,addr buf4  ; 读取NumberOfSections元素的值，并将其
                                    转换为16进制字符串，保存在buf4中
    ; 将buf4中的值输出到屏幕上
    mov ax,word PTR [buf4+4]
    mov buf5,ax
    invoke StdOut,addr buf5
    mov ax,word PTR[buf4+6]
    mov buf5,ax
    invoke StdOut,addr buf5
    invoke StdOut,addr endl         ; 换行
    invoke StdOut,addr tds          ; 输出TimeDateStamp元素的名称
    add esi,2h                      ;将esi指向TimeDateStamp元素值的位置
    invoke dw2hex,eax, addr buf4    ; 读取TimeDateStamp元素的值，并将其
                                      转换为16进制字符串，保存在buf4中
    mov eax,dword PTR[esi]
    invoke StdOut,addr buf4         ; 将buf4中的值输
    invoke StdOut, addr endl        ; 换行
    add esi,0eh                     ; 将esi指向Characteristics元素的值的
                                      位置
    invoke StdOut,addr chc; 输出Characteristics元素的名称

    mov eax,dword ptr[esi]
    invoke dw2hex, eax,addr buf4    ; 读取Characteristics元素的值，并将
                                    其转换为16进制字符串，保存在buf4中
    mov ax,word ptr[buf4+4]
    mov buf5,ax
    invoke StdOut, addr buf5
    mov ax,word ptr[buf4+6]
    mov buf5,ax
    invoke StdOut,addr buf5
    invoke StdOut,addr endl         ; 换行
    invoke StdOut, addr endl        ; 换行
```

```
;------------------------ IMAGE_OPTIONAL_HEADER-------------------
invoke StdOut,addr IOH
add esi ,12h                ; 将esi指向OptionalHeader元素的值的位置
invoke StdOut, addr endl    ; 换行
invoke StdOut, addr aop     ; 输出AddressOfEntryPoint元素的名称
mov eax, dword ptr[esi]
invoke dw2hex, eax, addr buf4
invoke StdOut,addr  buf4    ; 将buf4中的值输出到屏幕上
invoke StdOut, addr endl    ; 换行
; 输出ImageBase元素的名称
invoke StdOut,addr ib
; 将esi指向ImageBase元素的值的位置
add esi,4h
add esi,4h
add esi,4h
mov eax,dword PTR[esi]
invoke dw2hex,eax,addr buf4 ; 读取ImageBase元素的值，并将其转换为16
                              进制字符串，保存在buf4中
invoke StdOut, addr buf4    ; 将buf4中的值输出
invoke StdOut,addr endl     ; 换行
invoke StdOut,addr sa       ; 输出SectionAlignment元素的名称
add esi,4h                  ; 将esi指向SectionAlignment元素的值的位
                              置

mov eax,dword PTR[esi]
invoke dw2hex,eax, addr buf4 ; 读取SectionAlignment元素的值，并将其
                               转换为16进制字符串，保存在buf4中
invoke StdOut,addr buf4      ; 将buf4中的值输出
invoke StdOut,addr endl      ; 换行

add esi,4h                  ; 将esi指向FileAlignment元素的值的位置
invoke StdOut,addr fa       ; 输出FileAlignment元素的名称
mov eax,dword ptr[esi]
invoke dw2hex,eax,addr buf4  ;读取FileAlignment元素的值，并将其转
                              换为16进制字符串，保存在buf4中
invoke StdOut,addr buf4      ; 将buf4中的值输出
invoke StdOut,addr endl      ; 换行
invoke CloseHandle , hfile   ; 关闭文件句柄
```

4、**进行汇编操作：**使用 ml 将 PEviewer.asm 文件汇编到 PEviewer.obj 目标文件。

```
C:\Users\宋奕纬>D:

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

 Assembling: C:\Users\宋奕纬\Desktop\PEviewer.asm

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

5、**进行链接操作：**使用 link 将目标文件 PEviewer.obj 链接成 PEviewer.exe 可执行文件。

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

6、**测试运行**

```
D:\>PEviewer.exe
Please input a PE file:D:\dec2hex.exe
IMAGE_DOS_HEADER
   e_magic:  5A4D
   e_lfanew:  000000C0

IMAGE_NT_HEADERS
   Signature:  00004550

IMAGE_FILE_HEADERS
   NumberOfSections:  0003
   TimeDateStamp:  653B256F
   Characteics:  010F

IMAGE_OPTIONAL_HEADER
   AddressOfEntryPoint:  00001093
   ImageBase:  00400000
   SectionAligment:  00001000
   FILEAligment:  00000200
```

输入了前几次实验中的 dec2hex.exe 进行测试，成功输出其 PE 文件结构。

四、实验结论及心得体会

1、 **设计说明：**

总体思路：读取——转化——输出——变址——读取

（1）输出提示语

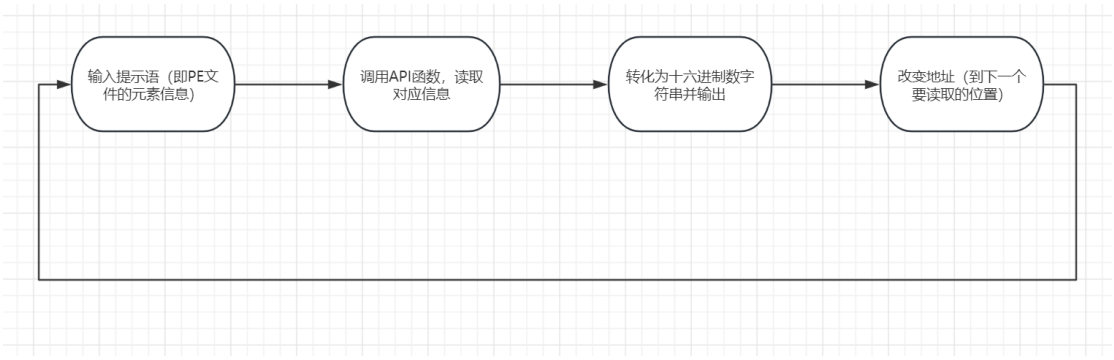
（2）读取 PE 文件信息

（3）转化为十六进制数字符串

（4）输出 PE 文件信息

（5）更改地址（下一个文件信息）

2、 控制流图：



3、 感受与收获：

更好的掌握了汇编语言的寻址与变址，对 PE 文件结构有了更加深入的了解，学会了部分 API 函数的用法。