

# 汇编语言与逆向技术实验报告

## Lab4-Peviewer

学号：2110957      姓名：蒋薇      专业： 计算机科学与技术

### 一、 实验内容

- (1) 窗口输出 “HelloWorld!” 字符串的汇编程序。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 out 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 out AddressOfEntryPoint、ImageBase、SectionAlignment、FileAlignment 的值，按照实验演示的方式输出到命令行窗口；

### 二、 实验步骤

- (1) peviewer 程序的设计说明和控制流图

利用 Windows API 函数 CreateFile、SetFilePointer、ReadFile、CloseHandle；

需要读取的内容是通过指针偏移量来定位的，确定好指针的起始位置，是 Dos 部首，还是 PE 文件头

文件头的信息    pe 文件节表的信息    pe 文件数据目录表的信息

从 RVA 到 Frva 的计算

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI ASCII
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ            YY
00000016	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	,            @
00000032	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000048	00	00	00	00	00	00	00	00	00	00	00	00	B0	00	00	00	e Ifanew.
00000064	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	2        'i!, Li!Th
00000080	69	73	20	70	72	6F	67	72	40	00	00	00	00	00	00	00	is, Program canno
00000096	74	20	62	65	20	72	75	6F	20	69	6E	20	44	4F	53	20	as be run in DOS
00000112	6D	6F	64	65	2E	0D	0D	0A	24	80	00	00	00	00	00	00	mode.    \$
00000128	E5	EA	7F	84	11	8B	11	D7	A1	8B	11	D7	A1	8B	11	D7	â ¨;< x;< x;< x
00000144	2F	94	02	07	AB	9B	11	D7	5D	AB	03	D7	AB	11	D7		/" x«< x]« x«< x
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Rich;< x
00000176	50	45	00	00	4C	01	03	00	CE	02	69	63	00	00	00	00	PE L    î ic
00000192	00	00	00	00	E0	00	0E	01	0B	01	05	0C	00	06	00	00	à
00000208	00	A0	00	00	00	00	00	00	00	10	00	00	00	10	00	00	
00000224	00	20	00	00	00	00	40	00	00	10	00	00	00	02	00	00	@
00000240	04	00	00	00	00	00	00	00	04	00	00	00	00	00	00	00	
00000256	00	D0	00	00	00	04	00	00	00	00	00	00	00	03	00	00	Ð
00000272	00	00	10	00	00	10	00	00	00	00	10	00	00	10	00	00	
00000288	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	
00000304	20	20	00	00	28	00	00	00	00	00	00	00	00	00	00	00	(
00000320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

(2) peviewer.asm 的源代码和注释

.386

.model flat,stdcall

option casemap :none

include \masm32\include\windows.inc

include \masm32\include\masm32.inc

include \masm32\macros\macros.asm

include \masm32\include\kernel32.inc

includelib \masm32\lib\masm32.lib

includelib \masm32\lib\kernel32.lib

.data

hi BYTE "Please input a PE file:",0;输入 pe 文件的语句

filename BYTE 20 DUP(0);将要输入的文件名输入进这里头

hfile DWORD 0;文件句柄

buf3 DWORD 4000 dup(?);从文件中接收数据的容器

buf4 DWORD 4000 dup(0);

buf5 word 4000 dup(0)

;以下是一些元素名称

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;用于换行

.code

main PROC

    invoke StdOut,addr hi

    invoke StdIn, addr filename,20

    ;调用 createfile 程序

    invoke CreateFile,     addr filename,\

        GENERIC\_READ,\

        FILE\_SHARE\_READ,\

        0,\

        OPEN\_EXISTING,\

        FILE\_ATTRIBUTE\_ARCHIVE,\

        0

    mov hfile, eax;将读取到的文件句柄传入 hfile

    ;调用 setfilepointer 程序

    invoke SetFilePointer, hfile,0 ,0,FILE\_BEGIN

    ;Indicates that the starting point is zero or the beginning of the  
file.

;调用 readfile 程序

;;;invoke ReadFile, hfile, addr buf3, 4000, 0, 0

;文件柄，盛接读取到的数据，读取的字节大小，指向  
读取的字节数的指针，NULL

;此时 hfile 在 e\_magic 的位置 MZ 的那个位置

invoke ReadFile, hfile, addr buf3, 400 , 0 ,0;buf3 是文件的入

口

mov esi, offset buf3

mov eax, dword PTR [esi]

invoke dw2hex, eax, addr buf4

mov ax,word PTR[buf4+4]

mov buf5,ax

invoke StdOut,addr IDH

invoke StdOut,addr endl

invoke StdOut,addr em

;invoke StdOut, addr buf4;读出 mz

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 el

add esi,3ch;指到 e\_lfanew

mov eax,dword PTR[esi]

invoke dw2hex,eax,addr buf4

invoke StdOut,addr buf4

invoke StdOut,addr endl

invoke StdOut, addr endl

invoke StdOut, addr INH

invoke StdOut, addr endl

invoke StdOut, addr sig

mov edx,dword PTR [esi]

mov esi,offset buf3

add esi, edx; e\_lfanew 内存的是 nt 头相对于文件的偏移地址

mov eax, dword PTR[esi] ;此时 esi 指在 signature

invoke dw2hex, eax,addr buf4

invoke StdOut, addr buf4

invoke StdOut, addr endl

invoke StdOut, addr endl

invoke StdOut, addr IFH

invoke StdOut, addr endl

invoke StdOut, addr nos

```
add esi,6h;移动到 FileHeader
mov eax,dword PTR[esi]
invoke dw2hex, eax,addr buf4
mov ax,word PTR [buf4+4]
mov buf5,ax;leorange13
invoke StdOut,addr buf5
mov ax,word PTR[buf4+6]
mov buf5,ax;leorange13
invoke StdOut,addr buf5
invoke StdOut,addr endl
invoke StdOut,addr tds
add esi,2h;移动到了 timedatestamp
mov eax,dword PTR[esi]
invoke dw2hex,eax, addr buf4
invoke StdOut,addr buf4
invoke StdOut, addr endl
add esi,0eh
invoke StdOut,addr chc
mov eax,dword ptr[esi]
invoke dw2hex, eax,addr buf4
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

invoke StdOut, addr endl

invoke StdOut,addr IOH

add esi ,12h;leorange13

invoke StdOut, addr endl

invoke StdOut, addr aop

mov eax, dword ptr[esi]

invoke dw2hex, eax, addr buf4

invoke StdOut,addr buf4

invoke StdOut, addr endl

invoke StdOut,addr ib

add esi,4h;!!!!

add esi,4h;

add esi,4h

mov eax,dword PTR[esi]



invoke dw2hex,eax,addr buf4

invoke StdOut, addr buf4

invoke StdOut,addr endl

invoke StdOut,addr sa

add esi,4h

mov eax,dword PTR[esi]

invoke dw2hex,eax, addr buf4

invoke StdOut,addr buf4

invoke StdOut,addr endl

add esi,4h

mov eax,dword ptr[esi]

invoke StdOut,addr fa

mov eax,dword PTR[esi]

invoke dw2hex,eax,addr buf4

invoke StdOut,addr buf4

invoke StdOut,addr endl

invoke CloseHandle , hfile

main ENDP

END main

peviewer.asm - 记事本

文件 编辑 查看

```
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\masm32.inc
include \masm32\macros\macros.asm
include \masm32\include\kernel32.inc
includelib \masm32\lib\masm32.lib
includelib \masm32\lib\kernel32.lib

.data
hi BYTE "Please input a PE file:",0;输入pe文件的语句
filename BYTE 20 DUP(0);将要输入的文件名输入进这里头
hfile DWORD 0;文件句柄
buf3 DWORD 4000 dup(?);从文件中接收数据的容器
buf4 DWORD 4000 dup(0);
buf5 word 4000 dup(0)

;以下是一些元素名称
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
```

行 17, 列 22

```
IOH BYTE "IMAGE_OPTIONAL_HEADER",0
aop BYTE " AddressOfEntryPoint:",0
ib BYTE " ImageBase:",0
sa BYTE " SectionAlignment:",0
fa BYTE " FILEAlignment:",0

endl BYTE 0Ah,0Dh,0;用于换行
.code
main PROC
    invoke StdOut,addr hi
    invoke StdIn, addr filename,20
    ;调用createfile程序
    invoke CreateFile, addr filename,\
        GENERIC_READ,\
        FILE_SHARE_READ,\
        0,\
        OPEN_EXISTING,\
        FILE_ATTRIBUTE_ARCHIVE,\
        0

    mov hfile, eax;将读取到的文件句柄传入hfile
    ;调用setfilepointer程序
    invoke SetFilePointer, hfile,0,0,FILE_BEGIN
    ;Indicates that the starting point is zero or the beginning of the file.
    ;调用readfile程序
    ;:::invoke ReadFile, hfile, addr buf3, 4000, 0, 0
    ;文件柄, 盛接读取到的数据, 读取的字节大小, 指向读取的字节数的指针, NULL
    ;此时 hfile在e_magic的位置 MZ的那个位置
    invoke ReadFile, hfile, addr buf3,400, 0,0;buf3是文件的入口
    mov esi, offset buf3
    mov eax, dword PTR [esi]
    invoke dw2hex, eax, addr buf4
```

```

invoke dw2hex, eax, addr buf4
invoke StdOut, addr buf4
invoke StdOut, addr endl
invoke StdOut, addr endl

invoke StdOut, addr IFH
invoke StdOut, addr endl
invoke StdOut, addr nos
add esi, 6h;移动到FileHeader
mov eax, dword PTR[esi]
invoke dw2hex, eax, addr buf4
mov ax, word PTR [buf4+4]
mov buf5, ax;leeorange13
invoke StdOut, addr buf5
mov ax, word PTR[buf4+6]
mov buf5, ax;leeorange13
invoke StdOut, addr buf5
invoke StdOut, addr endl
invoke StdOut, addr tds
add esi, 2h;移动到了timedatestamp
mov eax, dword PTR[esi]
invoke dw2hex, eax, addr buf4
invoke StdOut, addr buf4
invoke StdOut, addr endl
add esi, 0eh
invoke StdOut, addr chc
mov eax, dword ptr[esi]
invoke dw2hex, eax, addr buf4
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 dw2hex, eax, addr buf4
mov ax, word PTR[buf4+4]
mov buf5, ax
invoke StdOut, addr IDH
invoke StdOut, addr endl
invoke StdOut, addr em
;invoke StdOut, addr buf4;读出mz
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 el
add esi, 3ch;指到e_lfanew
mov eax, dword PTR[esi]
invoke dw2hex, eax, addr buf4
invoke StdOut, addr buf4
invoke StdOut, addr endl
invoke StdOut, addr endl

invoke StdOut, addr INH
invoke StdOut, addr endl
invoke StdOut, addr sig
mov edx, dword PTR [esi]
mov esi, offset buf3
add esi, edx; e_lfanew内存的是nt头相对于文件的偏移地址
mov eax, dword PTR[esi] ;此时esi指在signature
invoke dw2hex, eax, addr buf4
invoke StdOut, addr buf4
invoke StdOut, addr endl
invoke StdOut, addr endl

```

```

mov buf5, ax
invoke StdOut, addr buf5
invoke StdOut, addr endl
invoke StdOut, addr endl

invoke StdOut, addr IOH
add esi, 12h; leorange13
invoke StdOut, addr endl
invoke StdOut, addr aop
mov eax, dword ptr[esi]
invoke dw2hex, eax, addr buf4
invoke StdOut, addr buf4
invoke StdOut, addr endl
invoke StdOut, addr ib
add esi, 4h; !!!!
add esi, 4h;
add esi, 4h

mov eax, dword PTR[esi]
invoke dw2hex, eax, addr buf4
invoke StdOut, addr buf4

invoke StdOut, addr endl
invoke StdOut, addr sa
add esi, 4h
mov eax, dword PTR[esi]
invoke dw2hex, eax, addr buf4
invoke StdOut, addr buf4

invoke StdOut, addr endl

```

```

invoke StdOut, addr endl
add esi, 4h
mov eax, dword ptr[esi]
invoke StdOut, addr fa
mov eax, dword PTR[esi]
invoke dw2hex, eax, addr buf4
invoke StdOut, addr buf4

invoke StdOut, addr endl
invoke CloseHandle, hfile

```

```

ain ENDP
ND main

```

### (3) peviewer.exe 运行截图



```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.22000.1098]
(c) Microsoft Corporation。保留所有权利。

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

    Assembling: peviewer.asm

*****
ASCII build
*****

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

```
C:\Windows\System32\cmd.exe

D:\masm32>peviewer.exe
Please input a PE file:helloworld_cmd.exe
IMAGE_DOS_HEADER
    e_magic:  5A4D
    e_lfanew: 000000B0

IMAGE_NT_HEADERS
    Signature: 00004550

IMAGE_FILE_HEADERS
    NumberOfSections: 0003
    TimeDateStamp: 63493E1B
    Characterics: 010F

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

D:\masm32>
```

```
C:\Windows\System32\cmd.exe
D:\masm32>peviewer.exe
Please input a PE file:peviewer.exe
IMAGE_DOS_HEADER
  e_magic: 5A4D
  e_lfanew: 000000B0

IMAGE_NT_HEADERS
  Signature: 00004550

IMAGE_FILE_HEADERS
  NumberOfSections: 0003
  TimeDateStamp: 636D0008
  Characteristics: 010F

IMAGE_OPTIONAL_HEADER
  AddressOfEntryPoint: 00001000
  ImageBase: 00400000
  SectionAlignment: 00001000
  FileAlignment: 00000200

D:\masm32>
```

实验过程中的详细步骤，包括关键代码与截图等。

### 三、 实验心得

PE 文件：文件头(PE 文件头，DOS 文件头)，节表（主要每个节的 RVA 地址，RVA 地址是文件被装载到内存后数据相对于文件起始位置的偏移量），节区（属性相同的文件数据放在一起，可执行的，只读的放在一起）

- **DOS 文件头**

DOS 头部分由 MZ 格式的文件头和可执行代码部分组成。而 MZ 格式的文件头是由一个 IMAGE\_DOS\_HEADER 结构定义的：

```

IMAE_DOS_HEADER STRUCT{                                //DOS .EXE heade 位置

    e_magic    WORD    ?    //DOS 可执行文件标记, 为"MZ"    0x00
    e_cblp     WORD    ?    //Bytes on last page of file    0x02
    e_cp       WORD    ?    //Pages in file    0x04
    e_crlc     WORD    ?    //Relocations    0x06
    e_cparhdr  WORD    ?    //Size of header in paragraphs    0x08
    e_minalloc WORD    ?    //Minimum extra paragraphs needed    0x0A
    e_maxalloc WORD    ?    //Maximum extra paragraphs needed    0x0C
    e_ss       WORD    ?    //DOS 代码初始化堆栈段    0x0E
    e_sp       WORD    ?    //DOS 代码的初始化堆栈指针    0x10
    e_csum     WORD    ?    //Checksum    0x12
    e_ip       WORD    ?    //DOS 代码入口 IP    0x14
    e_cs       WORD    ?    //DOS 代码入口 CS    0x16
    e_lfarlc   WORD    ?    //File address of relocation table    0x18
    e_ovno     WORD    ?    //Overlay number    0x1A
    e_res      WORD    4 dup(?) //Reserved words    0x1C
    e_oemid    WORD    ?    //OEM identifier (for e_oeminfo)    0x24
    e_oeminfo  WORD    ?    //OEM information; e_oemid specific    0x26
    e_res2     WORD    10 dup(?) //Reserved words    0x28
    e_lfanew   DWORD    ?    //指向 PE 文件头    0x3C

IMAGE_DOS-HEADER ENDS

```

### PE 文件头 (NT 文件头)

NT 文件头包括三个部分, 分别是 PE 文件标识: Signature, FileHeader, OptionalHeader。这三个结构都是在一个 IMAGE\_NT\_HEADERS 结构里面

```

IMAGE_NT_HEADERS STRUCT                                ;NT 文件头
    Signature    DWORD    ?    ;PE 文件标识
    FileHeader   IMAGE_FILE_HEADER    <>
    OptionalHeader IMAGE_OPTIONAL_HEADER32    <>

```



第一个字段就是用来标识 PE 文件的，第二个字段的结构 IMAGE\_FILE\_HEADER 包含了很多信息：

```

IMAGE_FILE_HEADER  STRUCT
    Machine          WORD    ?    ;0004h - 运行平台
    NumberOfSections WORD    ?    ;0006h - 文件的节区数目
    TimeDateStamp     DWORD   ?    ;0008h - 文件的创建日期和时间
    PointerToSymbolTable  DWORD  ?    ;000ch - 指向符号表（用于调试）
    NumberOfSymbols    DWORD   ?    ;0010h - 符号表中的符号数量（用于调试）
    SizeOfOptionalHeader WORD ? ;0014h - IMAGE_OPTIONAL_HEADER32 的结构长度
    Characteristics   WORD    ?    ;0016h - 文件属性
IMAGE_FILE_HEADER  ENDS

```

IMAE_DOS_HEADER STRUCT{			//DOS .EXE header	位置
e_magic	WORD	?	//DOS 可执行文件标记, 为"MZ"	0x00
e_cblp	WORD	?	//Bytes on last page of file	0x02
e_cp	WORD	?	//Pages in file	0x04
e_crlc	WORD	?	//Relocations	0x06
e_cparhdr	WORD	?	//Size of header in paragraphs	0x08
e_minalloc	WORD	?	//Minimum extra paragraphs needed	0x0A
e_maxalloc	WORD	?	//Maximum extra paragraphs needed	0x0C
e_ss	WORD	?	//DOS 代码初始化堆栈段	0x0E
e_sp	WORD	?	//DOS 代码的初始化堆栈指针	0x10
e_csum	WORD	?	//Checksum	0x12
e_ip	WORD	?	//DOS 代码入口 IP	0x14
e_cs	WORD	?	//DOS 代码入口 CS	0x16
e_lfarlc	WORD	?	//File address of relocation table	0x18
e_ovno	WORD	?	//Overlay number	0x1A

e_res	WORD	4 dup(?)	//Reserved words	0x1C
e_oemid	WORD	?	//OEM identifier (for e_oeminfo)	0x24
e_oeminfo	WORD	?	//OEM information; e_oemid specific	0x26
e_res2	WORD	10 dup(?)	//Reserved words	0x28
e_lfanew	DWORD	?	//指向 PE 文件头	0x3C

IMAGE\_DOS-HEADER ENDS

- 节表

IMAGE\_SECTION\_HEADER STRUCT

Namel db IMAGE\_SIZEOF\_SHORT\_NAME dup(?) ;8 个字节的节区名称

union Misc

PhysicalAddress dd ?

VirtualSize dd ? ;节区的尺寸

ends

VirtualAddress dd ? ;节区的 RVA 地址

SizeOfRawData dd ? ;在文件中对齐后的尺寸

PointerToRawData dd ? ;在文件中的偏移

PointerToRelocations dd ? ;在 OBJ 文件中使用

PointerToLinenumbers dd ? ;行号表的位置（调试用的）

NumberOfRelocations dw ? ;在 OBJ 文件中使用

Characteristics dd ? ;节的属性

IMAGE\_SECTION\_HEADER ENDS