汇编语言与逆向技术课程实验报告

实验五：Peviewer

学院：密码与网络空间安全学院

专业：信息安全

学号：2412950

姓名：路浩斌

班级：信安一班

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

程序设计说明

该程序的主要流程是利用 Windows API 提供的文件读写函数，首先根据用户输入的文件路径打开指定的 PE 文件，并将其前 4000 字节读入缓冲区。随后程序依照 PE 文件格式中各字段的固定偏移量，通过寄存器 EDX 控制当前访问的位置。每读取一个字段时，只需要向 EDX 写入相应的偏移量，程序即可自动从缓冲区中获取该字段的数据，实现对 PE 文件头部信息的逐项解析。

在具体的数据输出部分，程序通过自定义的 Output 过程统一处理所有字段的读取与显示。Output 过程内部会根据 EDX 所指示的偏移与基址计算目标字段的实际地址，然后取出对应的 4 字节内容。为了保证输出格式一致，该过程利用事先编写的 dw2hex 函数，将读取到的数值转换为十六进制字符串，从而以规范的格式显示在命令行中。此外，程序通过检查 ECX 的值来判断当前字段应以 4 位还是 8 位十六进制形式输出，从而适配不同字段的长度。

控制流图

程序控制流图为：

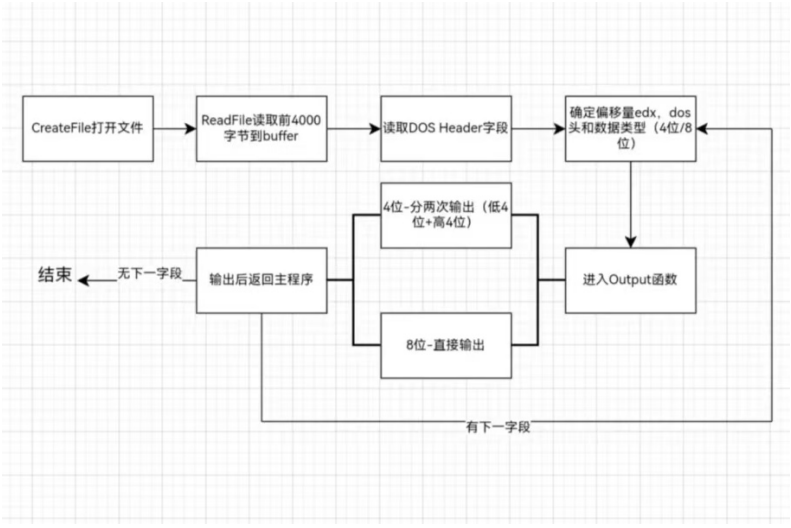

图 1: 运行截图

三、实验过程

源程序

```
.386
.model flat,stdcall
option casemap:none

include C:\masm32\include\windows.inc
include C:\masm32\include\masm32.inc
include C:\masm32\macros\macros.asm
include C:\masm32\include\kernel32.inc

includelib C:\masm32\lib\masm32.lib
includelib C:\masm32\lib\kernel32.lib

.data
; 输出文本
msg1     BYTE "Please␣input␣a␣PE␣file␣path:␣",0
msg2     BYTE "IMAGE_DOS_HEADER",0Ah,0Dh,0
msg3     BYTE "e_magic:",0
msg4     BYTE "e_lfanew:",0

msg5     BYTE "IMAGE_NT_HEADERS",0Ah,0Dh,0
msg6     BYTE "Signature:",0

msg7     BYTE "IMAGE_FILE_HEADER",0Ah,0Dh,0
msg8     BYTE "NumberOfSections:",0
msg9     BYTE "TimeDateStamp:",0
msg10    BYTE "Characteristics:",0

msg11    BYTE "IMAGE_OPTIONAL_HEADER",0Ah,0Dh,0
msg12    BYTE "AddressOfEntryPoint:",0
msg13    BYTE "ImageBase:",0
msg14    BYTE "SectionAlignment:",0
msg15    BYTE "FileAlignment:",0

msg16 BYTE 0Ah,0Dh,0    ; newline


```

```
; 缓冲区，用于读取PE文件
buffer      DWORD 4000 DUP(0)    ; 文件读取缓冲区
hexbuf      DWORD 4000 DUP(0)    ; 转换后的十六进制字符串
shortbuf    WORD  4000 DUP(0)    ; 存储 4 位短输出

filepath    BYTE  20 DUP(0),0    ; 输入文件路径
hFile       DWORD ?              ; 文件句柄

tmpSize     DWORD ?              ; 保存输出字节数
tmpBase     DWORD ?              ; 头部基地址

.code

; 输出偏移内容
Output PROC
    ; EDX：字段偏移
    ; tmpBase：用于 NT header 偏移基址

    mov esi, OFFSET buffer
    add esi, edx           ; EDX 偏移
    add esi, tmpBase       ; 若 tmpBase !=0，则为 NT 头偏移

    mov eax, DWORD PTR [esi]  ; 读取 4 字节字段
    mov ebx, eax              ; 保存到 EBX

    invoke dw2hex, eax, addr hexbuf ; 转成 hex 字符串

    mov ecx, tmpSize          ; 决定输出 4 位还是 8 位

    .if ecx == 8              ; 输出 8 个 hex 字符
        invoke StdOut, addr hexbuf
    .else                     ; 输出两次 short
        mov ax, WORD PTR[hexbuf+4]
        mov shortbuf, ax
        invoke StdOut, addr shortbuf

        mov ax, WORD PTR[hexbuf+6]
        mov shortbuf, ax
        invoke StdOut, addr shortbuf
    .endif
```

```
77
78      invoke StdOut, addr msg16
79      ret
80  Output ENDP
81
82  start:
83
84      ; 输入文件路径
85      invoke StdOut, addr msg1
86      invoke StdIn, addr filepath, 20
87
88      ; 打开文件
89      invoke CreateFile, addr filepath, GENERIC_READ, FILE_SHARE_READ,
            \NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL
90
91      mov hFile, eax
92
93      ; 将文件指针设为开头
94      invoke SetFilePointer, hFile, 0, 0, FILE_BEGIN
95
96      ; 读取前 4000 字节进入 buffer
97      invoke ReadFile, hFile, addr buffer, 4000, NULL, NULL
98
99      invoke StdOut, addr msg2
100
101     ; 输出 e_magic
102     invoke StdOut, addr msg3
103     mov edx, 0
104     mov tmpBase, edx      ; DOS header 基址=0
105     mov tmpSize, 4
106     invoke Output
107
108     ; 输出 e_lfanew
109     invoke StdOut, addr msg4
110     mov edx, 3Ch          ; e_lfanew 在 0x3C
111     mov tmpSize, 8
112     invoke Output
113
114
115     invoke StdOut, addr msg5
```

```
116
117     invoke StdOut, addr msg6
118     mov tmpBase, ebx      ; EBX = 刚刚读取的 e_lfanew
119     mov edx, 0            ; Signature 在 e_lfanew + 0
120     invoke Output
121
122
123     invoke StdOut, addr msg7
124     invoke StdOut, addr msg8
125     ;输出NumberOfSections
126     mov edx, 6h
127     mov tmpSize, 4
128     invoke Output
129
130     ;输出TimeDateStamp
131     invoke StdOut, addr msg9
132     mov edx, 8h
133     mov tmpSize, 8
134     invoke Output
135
136     ;输出Characteristics
137     invoke StdOut, addr msg10
138     mov edx, 16h
139     mov tmpSize, 4
140     invoke Output
141
142     ;输出AddressOfEntryPoint
143     invoke StdOut, addr msg11
144     invoke StdOut, addr msg12
145     mov edx, 28h
146     mov tmpSize, 8
147     invoke Output
148
149     ;输出ImageBase
150     invoke StdOut, addr msg13
151     mov edx, 34h
152     invoke Output
153
154     ;输出SectionAlignment
155     invoke StdOut, addr msg14
```

```
156     mov edx, 38h
157     invoke Output
158
159     ;输出FileAlignment
160     invoke StdOut, addr msg15
161     mov edx, 3Ch
162     invoke Output
163
164
165     invoke CloseHandle, hFile
166     invoke ExitProcess, 0
167
168 end start
```

运行截图

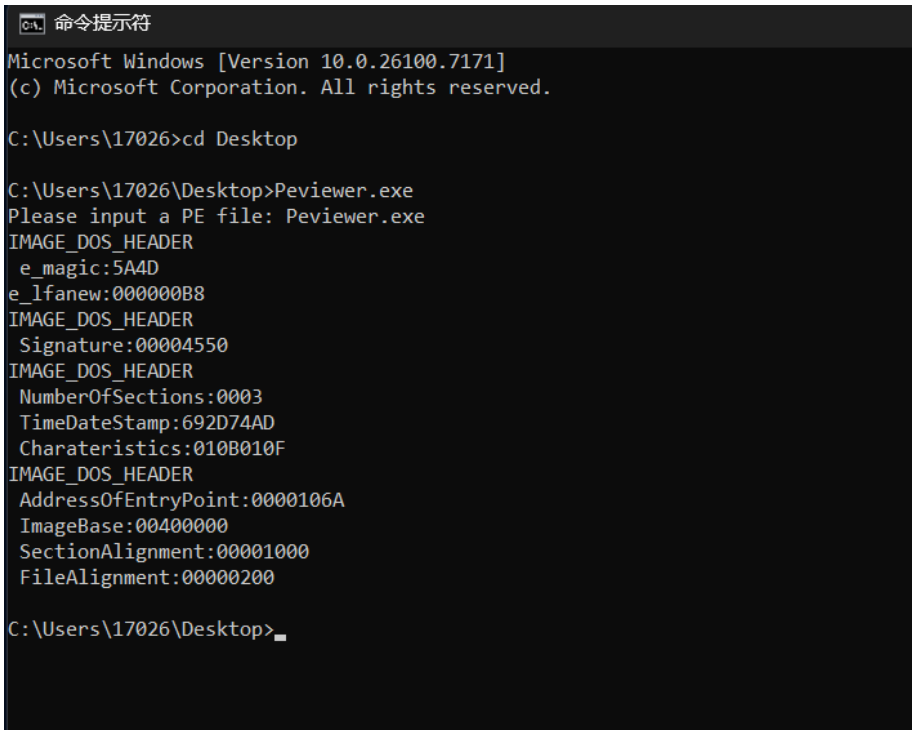

图 2: 运行截图

四、实验结论及心得体会

通过本次实验,我进一步加深了对 PE 文件结构的理解,并掌握了如何利用 Windows API 与汇编语言对文件头部进行解析。