**汇编语言与逆向技术实验报告**

**Lab4-peviewer**

**学号：2113662 姓名：张丛 专业：信息安全**

1. **实验内容**
   1. 熟悉PE文件结构；
   2. 使用Windows API函数读取文件内容
2. **实验步骤**

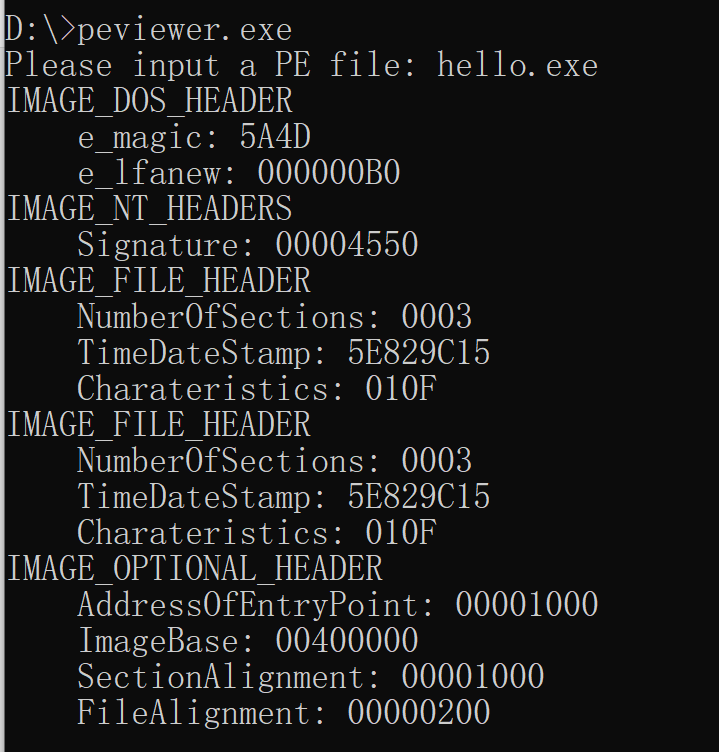


图 1 peviewer实验演示

* + 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；

1. **实验代码**

.386

.model flat, stdcall；平坦模式，调用约定

option casemap:none；大小写敏感

include D:\masm32\include\windows.inc

include D:\masm32\include\kernel32.inc

include D:\masm32\include\masm32.inc

includelib D:\masm32\lib\kernel32.lib

includelib D:\masm32\lib\masm32.lib

include D:\masm32\include\user32.inc

includelib D:\masm32\lib\user32.lib

ExitProcess PROTO, dwEXITCODE: DWORD

.data

;待处理文件绝对路径

path BYTE "D:\MASM projects\hello\_window.exe",0；处理hello\_window.exe文件

;存放待处理文件的句柄

filehandle DWORD ?

;以下数据全为显示字符，没有实际含义

str0 BYTE " ",0

str1 BYTE "IMAGE\_DOS\_HEADER:",0

str2 BYTE "IMAGE\_NT\_HEADER:",0

str3 BYTE "IMAGE\_FILE\_HEADE:",0

str4 BYTE "IMAGE\_OPTIONAL\_HEADER:",0

str5 BYTE "e\_magic:",0

str6 BYTE "e\_lfanew:",0

str7 BYTE "Signature:",0

str8 BYTE "NumberOfSections:",0

str9 BYTE "TimeDateStamp:",0

stra BYTE "Characteristic:",0

strb BYTE "AddressOfEntryPoint:",0

strc BYTE "ImageBase:",0

strd BYTE "SectionAligment:",0

stre BYTE "FileAligment:",0

strn BYTE 0ah,00h

filebuf BYTE 4000 DUP(0)

filebase DWORD ?

var DWORD ?

;下面声明的是把十六进制值转为字符串输出的函数

Print PROTO stdcall:DWORD,:DWORD,:DWORD

Res BYTE "00000000",0

table BYTE "0123456789ABCDEF",0

.code

Print PROC res:DWORD,hexdw:DWORD,t:DWORD

;参数res待写缓冲区，hexdw待转换十六进制值，t数据种类，控制最后输出的字符个数

MOV edx,0

MOV edi,hexdw

MOV ecx,8h

MOV eax,res

;edi指向待转换的值，ecx为循环变量，eax指向待写的缓冲区字符串

;循环将最高位保留做与运算，拿到最低位作为索引在table里面拿值

L2:

MOV ebx,edi

AND ebx,0f0000000h

SHR ebx,28

MOV dl,BYTE PTR [table+ebx]

MOV BYTE PTR[eax],dl

INC eax

SHL edi,4

LOOP L2

;将res的前t位变成空格

MOV ecx,t

MOV eax,res

L1:

CMP ecx,0h

JE L999

MOV BYTE PTR[eax],20h

INC eax

DEC ecx

JMP L1

L999:

;输出

INVOKE StdOut, res

MOV eax,hexdw

ret

Print ENDP

main PROC

INVOKE CreateFile,addr path,\

GENERIC\_READ,\

FILE\_SHARE\_READ,\

0,\

OPEN\_EXISTING,\

FILE\_ATTRIBUTE\_ARCHIVE,\

0

MOV filehandle,eax

INVOKE SetFilePointer, filehandle,\

0,\

0,\

FILE\_BEGIN

INVOKE ReadFile, filehandle,\

addr filebuf,\

3900,\

0,\

0

MOV esi,OFFSET filebuf

;以上操作将文件的内容写到了filebuf，之后读取filebuf就可以拿到PE文件的内容

;以下所有，esi为目标量在内存中的地址，eax为该地址对应值，print函数把eax传入，输出对应8-t个字符串

INVOKE StdOut,addr str1

INVOKE StdOut,addr strn

MOV eax,0h

INVOKE StdOut,addr str0

INVOKE StdOut,addr str5

;读第一个量

MOV ax,WORD PTR [esi]

MOV filebase,esi

INVOKE Print,addr Res,\

eax,\

4

INVOKE StdOut,addr strn

ADD esi,3ch

INVOKE StdOut,addr str0

INVOKE StdOut,addr str6

;读第二个量

MOV eax,DWORD PTR [esi]

INVOKE Print,addr Res,\

eax,\

0

;此处要注意，该数据为NT头的偏移量，所以将esi调整至NT头->内存基址加上NT头偏移

ADD eax,filebase

MOV esi,eax

INVOKE StdOut,addr strn

INVOKE StdOut,addr str2

INVOKE StdOut,addr strn

INVOKE StdOut,addr str0

INVOKE StdOut,addr str7

;读第三个量

MOV eax,DWORD PTR [esi]

INVOKE Print,addr Res,\

eax,\

0

INVOKE StdOut,addr strn

INVOKE StdOut,addr str3

INVOKE StdOut,addr strn

ADD esi,6h

INVOKE StdOut,addr str0

INVOKE StdOut,addr str8

;读第四个量

MOV eax,0

MOV ax,WORD PTR [esi]

INVOKE Print,addr Res,\

eax,\

4

INVOKE StdOut,addr strn

ADD esi,2h

INVOKE StdOut,addr str0

INVOKE StdOut,addr str9

;读第五个量

MOV eax,DWORD PTR [esi]

INVOKE Print,addr Res,\

eax,\

0

INVOKE StdOut,addr strn

ADD esi,0Eh

INVOKE StdOut,addr str0

INVOKE StdOut,addr stra

MOV eax,0

;读第六个量

MOV ax,WORD PTR [esi]

INVOKE Print,addr Res,\

eax,\

4

INVOKE StdOut,addr strn

INVOKE StdOut,addr str4

INVOKE StdOut,addr strn

ADD esi,18

INVOKE StdOut,addr str0

INVOKE StdOut,addr strb

;读第七个量

MOV eax,DWORD PTR [esi]

INVOKE Print,addr Res,\

eax,\

0

INVOKE StdOut,addr strn

ADD esi,0ch

INVOKE StdOut,addr str0

INVOKE StdOut,addr strc

;读第八个量

MOV eax,DWORD PTR [esi]

INVOKE Print,addr Res,\

eax,\

0

INVOKE StdOut,addr strn

ADD esi,4h

INVOKE StdOut,addr str0

INVOKE StdOut,addr strd

;读第九个量

MOV eax,DWORD PTR [esi]

INVOKE Print,addr Res,\

eax,\

0

INVOKE StdOut,addr strn

ADD esi,4h

INVOKE StdOut,addr str0

INVOKE StdOut,addr stre

;读第十个量

MOV eax,DWORD PTR [esi]

INVOKE Print,addr Res,\

eax,\

0

INVOKE StdOut,addr strn

INVOKE CloseHandle,filehandle

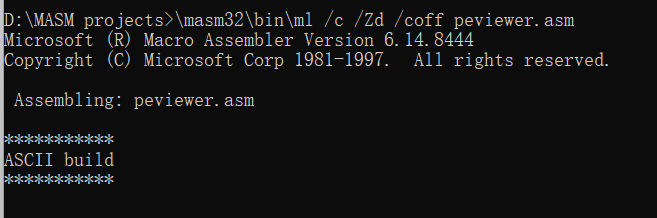
INVOKE ExitProcess,0

main ENDP

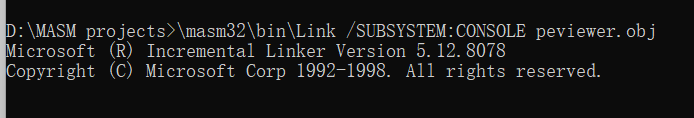
END main

1. **实验截图**

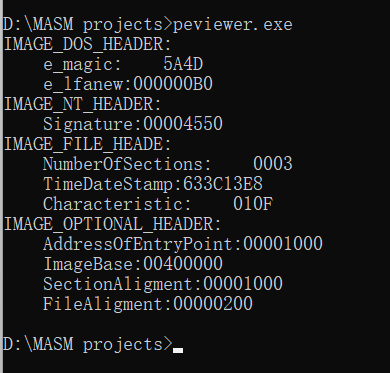
汇编：



链接：



实验结果：



1. **实验心得**

进一步熟悉了PE文件结构，与课上的知识点相结合；

能够使用Windows API函数读取文件内容