1. PE头字段说明
2. DOS头：

WORD e\_magic \* “MZ标记”用于判断是否为可执行文件。 5A4D

DWORD Eilane \* PE头相对于文件的偏移，用于定位PE文件。 100

1. 标准PE头:

WORD Machine \* 程序运行的CPU型号：0x0 任何处理器/0x14C 386及后续处理器。 14C

WORD NumberOfSections \* 文件中存在的节的总数，如果要新增节或者合并节，就要修改这个值。 4

DWORD TimeDateStamp \* 时间戳，文件的创建时间（和操作系统的创建时间无关），编译器填写的。50154494

DWORD PointerToSymbolTable

DWORD NumberOfSymbols

WORD SizeOfOptionalHeader \* 可选PE头的大小，32位PE文件默认E0h 64位PE文件默认位F0h,大小可以自定义。 E0

WORD Characteristics \* 每个位有不同的含义，可执行文件值为10F，即0、1、2、3、8位置1。 010F

1. 可选PE头：

WORD Magic \* 说明文件类型：10B 32位下的PE文件，20B 64位下的PE文件 010B

BYTE MajorLinkerVersion

BYTE MinorLinkerVersion

DWORD SizeOfCode \* 所有代码节的和，必须是FileAlignment的整数倍，编译器填的，没用。 1C000

DWORD SizeOfInitializedData \* 已初始化数据大小的和，必须是FileAlignment的整数倍，编译器填的，没用。

DWORD SizeOfUninitializedData \* 未初始化数据大小的和，必须是FileAlignment的整数倍，编译器填的，没用。

DWORD AddressOfEntryPoint \* 程序入口 193BE

DWORD BaseOfCode \* 代码开始的基址，编译器填的，没用。

DWORD BaseOfData \* 数据开始的基址，编译器填的，没用。

DWORD ImageBase \* 内存镜像基址。400000

DWORD SectionAlignment \* 内存对齐 1000

DWORD FileAlignment \* 文件对齐 1000

WORD MajorOperatingSystemVersion

WORD MinorOperatingSystemVersion

WORD MajorImageVersion

WORD MinorImageVersion

WORD MajorSubsystemVersion

WORD MinorSubsystemVersion

DWORD Win32VersionValue

DWORD SizeOfImage \* 内存中整个PE文件的映射尺寸，可以比实际的值大，但必须是SectionAlignmen整数倍 033000

DWORD SizeOfHeaders \* 所有头+节表按照文件对齐后的大小，否则加载会出错。 1000

DWORD CheckSum \* 校验和，一些系统文件有要求，用来判断文件是否倍修改。

WORD Subsystem

WORD DllCharacteristics

DWORD SizeOfStackReserve \* 初始化时保留的堆栈大小

DWORD SizeOfStackCommit \* 初始化时实际提交的大小

DWORD SizeOfHeapReserve \* 初始化时保留的堆大小

DWORD SizeOfHeapCommit \* 初始化时实际提交的大小

DWORD LoaderFlages

DWORD NumberOfRvaAndSizes \* 目录项数目 10

二、PE节表

#define IMAGE\_SIZEOF\_SHORT\_NAME

Typedef struct IMAGE\_SECTION\_HEADER {

BYTE Name[IMAGE\_SIZEOF\_SHORT\_NAME];

Union {

DWORD PhysicalAddress;

DWORD VirtualSize;

} Misc;

DWORD VirtualAddress;

DWORD SizeOfRawData;

DWORD PointerToRawData;

DWORD PointerToRelocations;

DWORD PointerToLinenumbers;

WORD NumberOfRelocations;

WORD NumberOfLinenumbers;

DWORD Characteristics;

} IMAGE\_SECTION\_HEADER, \*PIMAGE\_SECTION\_HEADER;

1. Name 8个字节 一般情况下是以“\0”结尾的ASCII码字符串来标识的名称，内容可以自定义。

注意：该名称并不遵守必须以“\0”结尾的规律，如果不以“\0”结尾，系统会截取8个字节的长度进行处理。

1. Misc { 是以下两个值的联合结构，可以（只）使用其中的任何一个，一般是取后一个。

PhysicalAdress, 物理地址

VitrualSzie 真实长度。是该节在没有对齐前的真实尺寸，该值可以不准确。

}

1. VirtualAddress 节区在内存中的偏移地址。加上ImageBase才是在内存中的真正地址。
2. SizeOfRawData 节在文件中对齐后的尺寸。
3. PointerToRawData 节区在文件中的偏移。
4. PointerToRelocations 在obj文件中使用，对exe无意义。
5. PointerToLinenumbers 行号表的位置，调试时使用。
6. NumberOfRelocations 在obj文件中使用，对exe无意义。
7. NumberOfLinenumbers 符合表中行号的数量，调试的时候使用。
8. Characteristics节的属性。

三.任意节空白区添加代码

1、codeBegin = (PBYTE)((DWORD)pImageBuffer + pSectionHeader->VirtualAddress +pSectionHeader->Misc.VirtualSize);//将代码复制到空闲区。

2、DWORD callAddr = (MESSAGEBOXADDR –(pOptionHeader->ImageBase+((DWORD)(codeBegin+0xD) – (DWORD)pImageBuffer)));//修正E8。

3、DWORD jmpAddr = ((pOptionHeader->ImageBase + pOptionHeader->AddressOfEntryPoint) – (pOptionHeader->ImageBase+((DWORD)codeBegin+SHELLCODELEN)));//修改E9。

\*（PDWORD）(codeBegin+0xE) = jmpAddr;

4、pOptionHeader->AddressOfEntryPoint = (DWORD)codeBegin – (DWORD)pImageBuffer;//修改OEP。