PE 文件内部结构探密*
Exploration of the Interior Structure on PE File

王建军**
（乐山师范学院　乐山　614004）

摘要：通过对 Windows 环境下 PE 文件结构的探索，揭示了 PE 文件的内部结构，为程序员更深层次编写程序提供了实质性的理论基础，尤其在病毒安全、黑客攻防等方面更需要借鉴。

关键词：PE；地址；段；RAR
中图分类号：TP311.1　　**文献标识码**：B　　**文章编号**：(1672-4550(2005)03-0037-04

1　引言

正如操作系统这个概念，使用电脑的人都离不开它，但还是有许多人却不明白其含义。Windows 环境下编写程序对大家来说，早已不是什么新鲜事了，但却依然有大量的人，包括不少程序员也不明白自己所写的程序是怎样的结构，且程序是如何在 Windows 平台上运行起来的。

2　PE 的由来

PE 是 Portable Executeable（可移植的执行体）的简称，可移植是指采用这种技术的文件在各种硬件平台（如 X86 ，Mips，Alpha）上都能够执行。从 Windows 95 到 Windows XP，它在 Win 32 平台上得到了广泛的应用。另外包括 EXE，dll，SYS，VDM 等都使用了这一技术。与之对应的，在 DOS 环境下使用的可执行文件格式技术被称为 MZ，在 Win3. x 等 Win16 上使用的可执行文件格式技术被称为 NE，以及在 W16 上的虚拟设备驱动程序 VXD 使用的 32 位的文件格式 LE。

3　PE 结构

PE 文件整个格式的组成规划如下：一个 MS-DOS 的 MZ 头部之后是一个实模式的残留程序、PE 文件标志、PE 映像头部、PE 可选映像头部、所有的段头部，最后是所有的段实体，如图 1 所示：

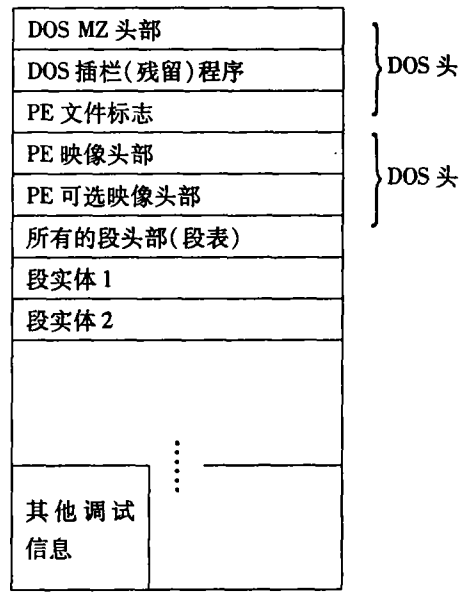

图　1

3.1　DOS 头部

为了避免用户有意或无意在 DOS 环境中运行 PE 文件而造成死机的现象，PE 保留了 DOS 信息。我们经常看到的“This program can not be run DOS mode.”等类似提示信息就是由 PE 文件的 DOS 头部完成的。这是非常重要而程序员一般又不感兴趣的东西，所以编译自动生成了。DOS 头部常常被称为 MZ，就是其标志字节为“MZ”的 ASCII 码。用二进制编辑工具打开任何一个 Win32 可执行程序，都会发现前两个字节（即相对偏移地址 RAR0

* ［收稿日期］2004-11-30；　［修订日期］2004-12-29
** ［作者简介］王建军（1972—），男，讲师，主要从事操作系统，数据库方面教学、实验和科研工作。

处)为4D5A，如图2所示。

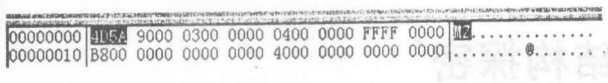

图 2

紧接后面的就是完成 DOS 提示的简单程序，而有少数既能够在 Windows 下工作，又能在 DOS 下工作的程序，这一部分就不是简单的残留程序了。

更为重要的是还有指向 PE 文件头的指针 E_lfanew，它位于 PE 文件 RAR 的 3CH 处，连续使用了 4 个字节(可表示 4G 的寻址范围)，其值随不同的程序有所差异，如图 3 所示。

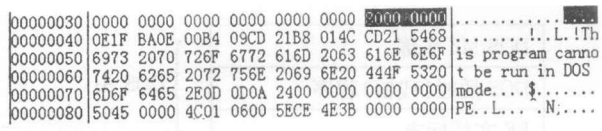

图 3

在偏移为 3CH 处找到 PE 头部所在相对偏移地址为 00000080H(注意按高低字节存放原理，而不是 8000000H)，在左下角发现对应的内容为 5045000，每个字节为一个 ASCII 符号，即 PE\ 0\ 0。

3.2 PE 头

紧接着 DOS stub 的是 PE header。PE 头由三部分构成：PE 标识符、PE 映像头、PE 可选映像头。

(1) PE 标识符就是“PE\ 0\ 0”，从图 3 中也可以反映出来，就不赘述了。

(2) 紧随其后的是 PE 头的第二部分。PE 映像头包括了该 PE 文件的运行平台，所包含的段(块)的总数，文件所创建的日期时间等信息，可用如下的结构来描述这一部分内容：

```
typedef struct _ IMAGE_ FILE_ HEADER
{
    WORD    Machine;        //0x04
    WORD    NumberOfSections;       //0x06
    DWORD   TimeDateStamp;      //0x08
    DWORD   PointerToSymbolTable;       //0x0c
    DWORD   NumberOfSymbols;        //0x10
    WORD    SizeOfOptionalHeader;       //0x14
    WORD    Characteristics; //0x16
} IMAGE_ FILE_ HEADER, * PIMAGE_ FILE_ HEADER;
```

仍查看图 3，PE 标识(占用 4 字节)后面的两个字节内容为 014CH，表示该程序运行于 Intel I386 系列 CPU。

(3) PE 可选映像头是一可选部分，它包含很多关于 PE 文件定位的信息。如下所示：

```
typedef struct _ IMAGE_ OPTIONAL_ HEADER {
    WORD    Magic;    //0x18
    BYTE    MajorLinkerVersion;    //0x1a
    BYTE    MinorLinkerVersion;    //0x1b
    DWORD   SizeOfCode;    //0x1c
    DWORD   SizeOfInitializedData;    //0x20
    DWORD   SizeOfUninitializedData;    //0x24
    DWORD   AddressOfEntryPoint;    //0x28
    DWORD   BaseOfCode;    //0x2c
    DWORD   BaseOfData;    //0x30
    DWORD   ImageBase;    //0x34
    DWORD   SectionAlignment;    //0x38
    DWORD   FileAlignment;    //0x3c
    WORD    MajorOperatingSystemVersion;    //0x3e
    WORD    MinorOperatingSystemVersion;    //0x40
    WORD    MajorImageVersion;    //0x42
    WORD    MinorImageVersion;    //0x44
    WORD    MajorSubsystemVersion;    //0x46
    WORD    MinorSubsystemVersion;    //0x48
    DWORD   Win32VersionValue;    //0x4c
    DWORD   SizeOfImage;    //0x50
    DWORD   SizeOfHeaders;    //0x54
    DWORD   CheckSum;    //0x58
    WORD    Subsystem;    //0x5c
    WORD    DllCharacteristics;    //0x5e
    DWORD   SizeOfStackReserve;    //0x60
    DWORD   SizeOfStackCommit;    //0x64
    DWORD   SizeOfHeapReserve;    //0x68
    DWORD   SizeOfHeapCommit;    //0x6c
    DWORD   LoaderFlags;    //0x70
    DWORD   NumberOfRvaAndSizes;    //0x74
    IMAGE_ DATA_ DIRECTORY DataDirectory [IMAGE_ NUMBEROF_ DIRECTORY_ ENTRIES];
} IMAGE_ OPTIONAL_ HEADER, * PIMAGE_ OPTIONAL_ HEADER;
```

其中的内容非常多，这里就不一一介绍了。举一个例，第一个标志 Magic(距离 PE 标识偏移 18H 处)表示文件是 ROM 中的映像或是普通可执行的映像，一般都是普通 PE 文件，其值为 010BH，如图 4 所示。

3.3 段表头

段表头也称为块表头。这一部分是 PE 头与实际数据之间的部分，它实际上就是该 PE 文件共使用的段的目录列表，从这里可以快速检索到每一个

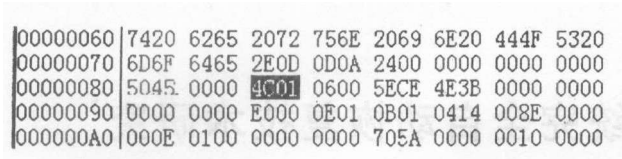

图 4

块的名字，段总长度，在内存中的相对虚拟地址 RARdeng 等详细信息，也是 PE 文件正确执行和提取数据的依据。它的作用就相当于磁盘上的文件目录，通过目录，我们就可以快速地找到相应文件数据，可见其重要性。段表头的结构定义如下：

```
    typedef struct _ IMAGE_ SECTION_ HEADER {
UCHAR Name [8];       //块名
 union{          //段长度
  ULONG PhysicalAddress;
  ULONG VirtualSize;
   } Misc;
 ULONG VirtualAddress;   //该段（块）的 RAR
 ULONG SizeOfRawData;    //对齐后的大小
 ULONG PointerToRawData;    //在文件中偏移
 ULONG PointerToRelocations; 在 obj 中，重定位偏移
 ULONG PointerToLinenumbers;   //行号偏移
 USHORT NumberOfRelocations;   // 在 obj 中，重定位项数
 USHORT NumberOfLinenumbers;    //行号数目
 ULONG Characteristics; 段(块)属性
  } IMAGE_ SECTION_ HEADER, * PIMAGE_ SECTION _ HEADER;
```

通过图 5 的示例可见其段的名字 . text(2e74 6578 74)，还可从偏移地址 0180 中看见其值的大小为 00001000H 即 4096，刚好 4KB。

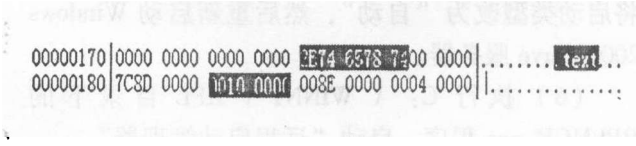

图 5

3.4 段数据

`，接着就是各段的具体数据，一般都包括 . text，. data，. idata，. rsrd 等段的具体数据，这里的段和 DOS 下汇编程序的段是不同的，其他文中有详细的讨论，这里就不详细描述了。

4 PE 与计算机安全

对 PE 文件内部结构不了解就不可能在计算机安全方面进行更深入的研究，更谈不上对计算机病毒的编写和黑客程序的编写等技术的研究。下面就这两个方面简单谈谈与 PE 结构相关的技术。

4.1 病毒安全

Win32 下的针对 PE 文件的计算机病毒习惯被称为 Win32 病毒或 PE 病毒，主要目标是感染 Win32 的 EXE 和 DLL 文件。如劳拉(Win32. Xorala)病，Nimda，QQ 病毒，CIH 等，其传播途径多种多样。

Win32 病毒通过各自的传播方式侵入电脑被激活后往往要收索磁盘上的 PE 文件，确定病毒的客体。要把病毒自身化整为零隐藏在 PE 文件中，这样一来，一般用户就很难通过文件大小来发现是否感染病毒了。PE 文件的头部和各段中都有少量的自由空间，而计算机病毒往往都写得非常精炼，而且再分成若干个更小的块就很容易隐藏于正常的 PE 文件中了。

Win32 病毒往往把自己的头部隐藏于 PE 文件头部，还要建立自己的病毒链表，以便在将来要运行的时候迅速地化零为整，组装起来。而病毒的其他部分分别置入到本文所提到的 PE 文件的其它各段的自由空间中。

另外，Win32 病毒还必须修改文件入口地址(本文已经提到)，让其指向病毒本身代码，并且把原来的入口地址也保存下来，以便能够回到正常的执行地址。以达到在 PE 文件执行之前首先获得执行，才能处于活动状态以便继续感染其他文件或者发作。

反病毒技术是病毒技术的逆过程，这里不进行讨论了。

4.2 黑客攻防

随着 IT 技术的发展，黑客技术与病毒技术的边缘越来越模糊，结合得越来越紧密。而反病毒技术往往与防火墙技术也结合起来了。很多带有恶意性质的网站就利用黑客技术把不怀好意的 PE 文件传播到你的磁盘，这些 PE 文件本身就可能是病毒或窃取你的密码资料的木马。黑客把这些 PE 文件编码成 javascript 脚本文件隐藏于网页中（这里不讨论怎样编码了），当不知情的用户访问这些网页时，这些脚本又会把原来的 PE 文件解码到用户的启动盘的启动目录中。在用户不知不觉的情况下，下次启动电脑时，这些 PE 文件就开始按照黑客的设计意愿开始卖力了。

反黑技术在这些方面自然就要对进入计算机的数据流进行过滤，判断代码的工作意图，从而进行

(下转第 36 页)

模块提供时标信息（即当前计数值）。

（4）BHP 生成、验证以及复接模块：用于环回测试的模块，以某种方式 BHP 帧格式，模拟实际核心系统中的核心调度模块向 BHP 发送模块发送 BHP 帧，将由 BHP 接收回的数据与发送的 BHP 进行比较验证。此模块主要用于测试，生成 BHP 可采用多种方式，用生成的伪随机序列封装为 BHP 帧格式；用板载空闲跳线插槽通过 FPGA 输入固定信号作为 BHP 信息，可通过跳线更改输入数据；FPGA 自行产生周期信号。验证可通过板载 LED 指示，也可通过空闲针脚输出周期的验证信息，通过示波器检验输出波形。

（5）伪随机序列生成模块：可根据需要生成指定长度的伪随机序列。

多个环回模块并存即可实现多路 BHP 的复接测试。

4 结论

在目前的测试系统中，将由跳线插槽输入的信号嵌入 BHP 作为修正前的 offset time 字段值，接收到的 offset time 字段值在校验 LED 可以直接显示出，让时标发生器输出固定时标，跳线改变时，对应 LED 改变为预期值；每一输入/输出端口有专用模块处理，在校验复接模块复接放入缓存，校验；当 BHP 校验通过，由保留引脚引出校验信号，通过示波器可观测到稳定的校验信号；发送端 Serdes 的 Rclk 引脚能恢复出与发送端 Sclk 仅有相位差异的稳定周期波。可见测试系统能实现 BHP 的收发、同步、解析、部分修改、复接等基本功能。采用比较常用的 Serdes 器件，性能较为稳定，较易于实现，能为 OBS 核心控制提供稳定的高速接口。

当前的接口测试系统大部分功能均依靠核心 FPGA 实现，实现的功能也较为简单，而当统计、优先级排队、核心调度等 OBS 核心节点控制系统的其他模块加板测试时，可能对造成 FPGA 资源使用的紧张甚至缺乏。以后的系统中在不更换核心 FPGA 的情况下可考虑采用 FPSC 芯片，将接收模块独立成单板，通过标准接口与核心 FPGA 板连接，将高速接口的控制统计、编码解码、同步等工作交给 FPSC 分担，标准的接口也便于以后的扩展，通过更换接口板，使核心控制器可以实现支持千兆以太网等功能。

参 考 文 献

[1] C. Qiao and M. Yoo, " Optical burst switching (OBS) - A new paradigm for an optical internet," J. High Speed Networks, 1999, 8 (1): 68 - 84

[2] HFCT - 5205 SC Duplex Single Mode Transceiver Data Sheet January 28, 2001

[3] DS92LV1021A 16 - 40 MHz 10 Bit Bus LVDS Serializer January, 2003

[4] FPSC SERDES CML Buffer Interface July, 2003

[5] Virtex - II Complete Data Sheet (All four modules). 2004

[6] David G Cunningham ; William G Lane o Publisher . " Gigabit Ethernet networking", Macmillan Technical Pub, 1999

（上接第 39 页）

过滤，当然还有对内存的监控技术等。

5 结束语

本文对 PE 文件结构的探索、对病毒安全及黑客技术的初步讨论旨在抛砖引玉，以管窥豹。要对信息安全技术更深入地了解需要做更多的研究，参考更多文献。

参 考 文 献

[1] 段钢编著. 加密与解密. 北京:电子工业出版社,2003

[2] 杨季文等编著. 汇编语言程序设计. 北京：清华大学出版社，1998

[3] 汤子瀛等编著. 计算机操作系统. 西安：西安电子科技大学出版社，2001

[4] 徐小刚等编著. Visual C + + 入门与提高. 北京：清华大学出版社，1998

[5] Microsoft 公司 http：//msdn. microsoft. com/网络资源

耐心和恒心总会得到报酬的。

——爱因斯坦