十、PE文件二

PE文件型病毒

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- Windows系统病毒之一，在安全模式下可以删除
- 将可执行文件的代码中程序入口地址改为病毒的程序入口，这样就会导致用户在运行的时候执行病毒文件
- 黑客比较常用的方式

PE文件型病毒

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- CIH病毒
  - 一种能够破坏计算机系统硬件的恶性病毒
  - CIH病毒是一位名叫陈盈豪的台湾大学生所编写的
  - 载体是一个名为“ICQ中文Chat模块”的工具，并以热门盗版光盘游戏如“古墓奇兵”或Windows95/98为媒介
  - 互联网各网站互相转载，使其迅速传播
  - 属文件型病毒，杀伤力极强
  - 主要表现在于病毒发作后，硬盘数据全部丢失，甚至主板上BIOS中的原内容也会被彻底破坏，主机无法启动

病毒感染PE文件的基本方法

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

1. 判断目标文件开始的两个字节是否为“MZ”
2. 判断PE文件标记“PE”
3. 判断感染标记，如果已被感染过则跳过这个文件，否则继续
4. 获得Directory(数据目录)的个数，每个数据目录信息占8个字节
5. 得到节表起始位置：Directory的地址+数据目录占用的字节数=节表起始位置
6. 得到目前最后节表的末尾偏移(紧接其后用于写入一个新的病毒节)： 节表起始位置+节的个数×(每个节表占用的字节数28H)=目前最后节表的末尾偏移

7. 开始写入新的节表项
   ① 写入节名(8字节)
   ② 写入节的实际字节数(4字节)
   ③ 写入新节在内存中的开始偏移地址(4字节)，同时可以计算出病毒入口位置： 上节在内存中的开始偏移地址+(上节大小/节对齐+1)×节对齐
   ④ 写入新节(即病毒节)在文件中对齐后的大小
   ⑤ 写入新节在文件中的开始位置： 上节在文件中的开始位置+上节对齐后的大小

病毒感染PE文件的基本方法

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

8. 修改映像文件头中的节表数目
9. 修改AddressOfEntryPoint(即程序入口点指向病毒入口位置)，同时保存旧的AddressOfEntryPoint，以便返回HOST继续执行。
10. 更新SizeOfImage(内存中整个PE映像尺寸=原SizeOfImage+病毒节经过内存节对齐后的大小)；
11. 写入感染标记(可以放在PE头中)
12. 写入病毒代码到新节指向的文件偏移中

PE病毒编写的关键技术

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 重定位
- 获取API函数
- 搜索目标文件
- 感染
- 破坏

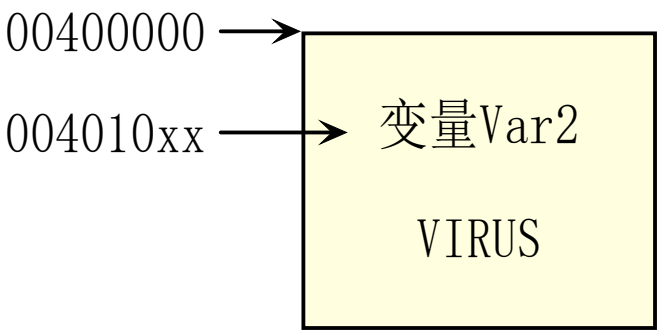
PE病毒编写的关键技术

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

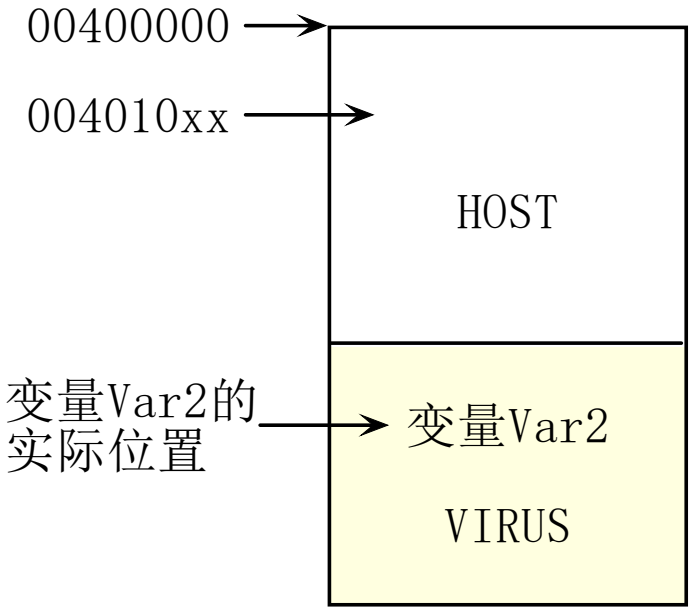
- 病毒是在宿主的运行环境下运行，所以无法像在自己本身的运行环境下一样访问自己的静态（全局）变量的数据和直接调用系统API

- 通过一些技术可以克服上述的难点，但编写起来会比较繁琐

病毒编译后



病毒模块进入宿主体

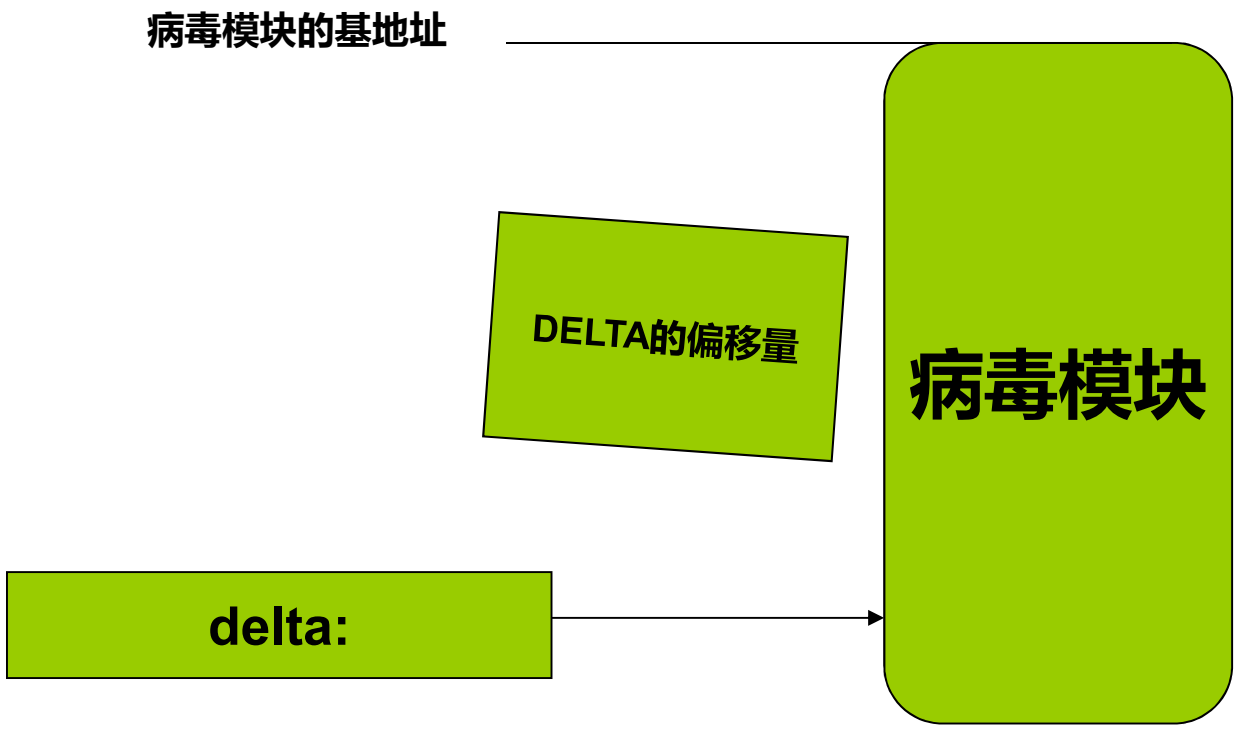


重定位

- **为什么程序加载到一个不同的位置之后会出错?**

- **病毒是否能够事先预料到自己的病毒代码将添加到HOST什么位置? 加载之后又在什么位置?**
  - 不能
  - 怎么办?
    - 代码重定位

重定位



病毒代码基地址= Delta的地址 - Delta的偏移量

重定位

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

```
call delta
delta:   pop eax
         sub eax,offset delta
```

运行后，eax中存放的是病毒代码基地址

则

V2的地址 = eax + offset v2

重定位

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

1. call指令跳转到下一条指令,使下一条指令感染后在内存中的实际地址进栈

2. 用pop EXX,[ESP]指令取出栈顶内容,得到感染后下一条指令内存中的实际地址Base

3. varstart为感染前call指令的下一条指令地址,varlable为感染前变量地址,则感染后var实际地址为(Base-Offset varstart)(基地址)+Offset varlable

获取API函数地址

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

为什么要获取API函数地址

- Win32下的系统功能调用一般通过调用动态连接库中的API函数实现

- API函数调用的实质是找到函数地址，然后call

```
call MessageBox(0,”123”,0,0);
Call 0x7c91001c;   (WIN XP SP3)
```

获取API函数地址

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 输入表是用来描述可执行文件需要调用的外部函数(API)

ExitProcess	7081CDDA
MessageBoxA	77D504EA
wsprintfA	77D1A8AD

获取API函数地址

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 病毒和普通程序一样需要调用API函数实现某些功能，但病毒运行在宿主环境下，在编写上不能直接写函数名去调用API（引入表提供把函数名转换为函数地址），必须病毒自身去获取API函数地址（动态调用API）
- 静态方式：调用时，根据函数名查引入表，就可以获取该函数的地址
- 动态方式：使用函数LoadLibrary装载需要调用的函数所在的dll文件，获取模块句柄。然后调用GetProcAddress获取需要调用的函数地址。这种方式是在需要调用函数时才将函数所在的模块调入到内存中，同时也不需要编译器为函数在引入表中建立相应的项

获取API函数地址

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

1. LoadLibrary加载一个DLL，返回DLL地址
2. GetProcAddress通过DLL地址和API函数名获得API函数的地址

C语言实例：

DLL地址 = LoadLibrary(“DLL名”);

API函数地址 = GetProcAddress(DLL地址,“函数名”);

获取API函数地址

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 获得LoadLibrary和GetProcAddress的地址
- 这两函数是系统模块kernel32.dll提供的，所以他们必定在kernel32的引出表中被导出
- 只要我们能得到kernel32的地址，我们就可以通过搜索kernel32的引出表，搜索得到它们的地址

获取模块kernel32地址 方法1

- 由于程序入口点是被kernel32某个函数调用的，所以这个调用函数肯定在kernel32的地址空间上
- 那么我们只要取得这个返回地址，就得到了一个kernel32空间中的一个地址
- 通过这个地址，我们可以从高地址向低地址方向进行搜索，通过PE标志的判断，搜索到kernel32模块的基地址

获取模块kernel32地址 方法1

- 大致流程如下：
  取栈顶值到寄存器A（KERNEL32中的一个地址）
  A = A 与 0FFFFF000h（分配粒度是1000h，基地址必然在xxxx000h处）
  循环：
  如果[A] == IMAGE_DOS_SIGNATURE（判断DOS头标志）
  {B = A；B =B+e_lfanew； 指向PE标志
  如果[B] ==IMAGE_NT_SIGNATURE （判断“PE\0\0”标志）
  { 跳出循环；（找到，退出！）}
  A = A - 01000h;
  循环结束

获取模块kernel32地址 方法2

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 通过宿主进程的PEB：进程环境块获得

- Fs寄存器->TEB
- TEB + 0x30->PEB
- PEB + 0xc->PEB_LDR_DATA
- PEB_LDR_DATA+0x1c处存放了一些DLL的地址，第一个是nt.dll地址，第二个就是kernel32.dll的地址

获取模块kernel32地址 方法3

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- SEH(结构化异常处理)
- SEH链表中最顶层的异常处理函数是Kernel32.dll中的一个函数
- 可以遍历这个链表去搜索这个函数地址，通过这个函数地址向低地址方向以64KB为对齐单位查找PE文件的DOS头标志”MZ”，从而找到Kernel32.dll的地址

获取模块kernel32地址 方法4

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 宿主进程中的TEB：线程环境块，这个块存放了线程的栈顶地址，这个地址+0x1c肯定位于kernel32.dll中（NT系统）

获取API函数地址

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 在得到了Kernel32的模块地址以后，就可以搜索他的导出表得到GetProcAddress和LoadLibrary两个API函数的地址

- 对这两个API函数的联合调用就可以得到WIN32 应用层上任何所需要的API函数地址了

通过函数名称查找函数地址

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

1. 定位到PE文件头
2. 从PE文件头中的可选文件头中取出数据目录表的第一个数据目录，得到导出表的地址
3. 从导出表的NumberOfNames字段得到以命名函数的总数，并以这个数字做微循环的次数来构造一个循环
4. 从AddressOfNames字段指向的函数名称地址表的第一项开始，在循环中将每一项定义的函数名与要查找的函数名比较，如果没有任何一个函数名符合，说明文件中没有指定名称的函数
5. 如果某一项定义的函数名与要查找的函数名符合，那么记住这个函数名在字符串地址表中的索引值（如x），然后在AddressOfNameOrdinals指向的数组中以同样的索引值x去找数组项中的值，假如该值为y
6. 以y值作为索引值，在AddressOfFunctions字段指向的函数入口地址表中获取的RVA就是函数的入口地址，当函数被装入内存后，这个RVA值加上模块实际装入的基址(ImageBase)，就得到了函数真正的入口地址

搜索目标文件

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- PE病毒通常以PE文件格式的文件（如EXE、SCR、DLL等）作为感染目标
- 在对目标进行搜索时一般采用两个关键的API函数：
- FindFirstFile
- FindNextFile
- 其一般搜索“*.exe”、“*.scr”等文件进行感染。
- 在算法上可以采用递归或者非递归算法对所有盘符进行搜索

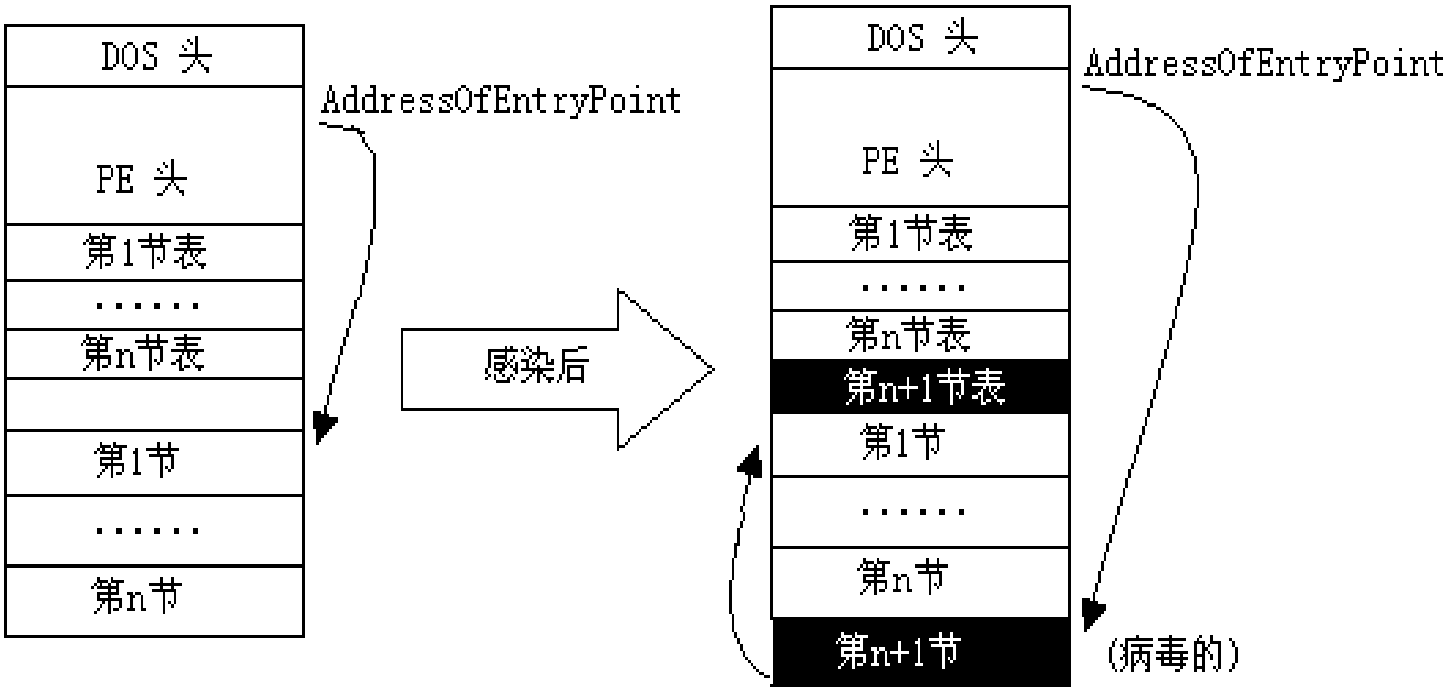
文件感染

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 添加节
- 扩展节
- 插入节

添加节方式修改PE

- 在文件的最后建立一个新节，在节表结构的后面建立一个节表，用以表述该节。入口地址修改为病毒所在节

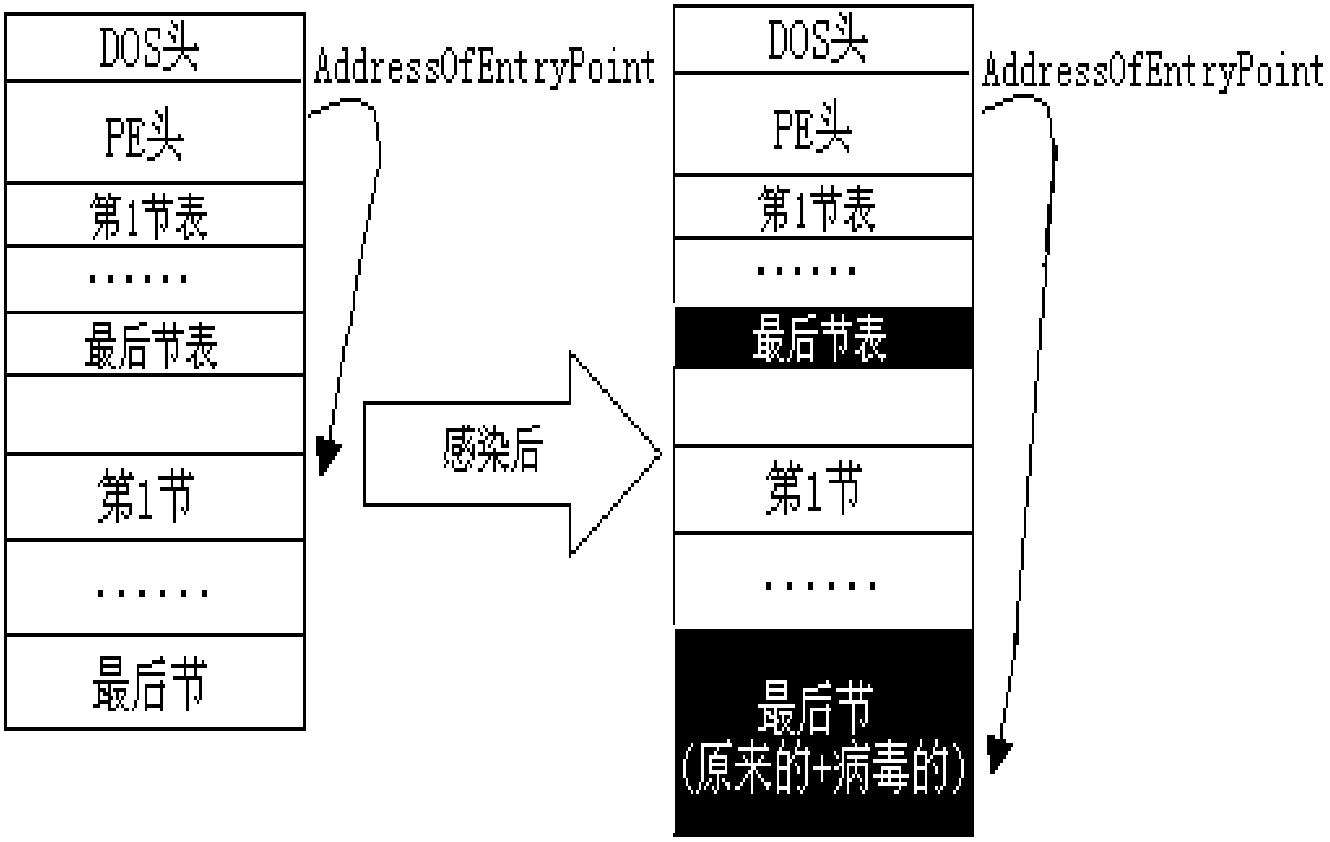


添加节方式修改PE

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 先把病毒代码追加到文件尾部
- 在节表中增加一个section header各项数据填写正确（VirtualSize,VirtualAddress,PointerToRawData…..）。
- 在FILEHEADER中修改节表项数目： +1
- 重新计算SizeofHeaders，并替换原值
- 重新计算SizeofImage，并替换原值
- 记录未感染时的AOEP（入口地址），因为在病毒代码结束时要让宿主程序正常执行。所以要先记录AOEP，在病毒程序结束后JMP跳到宿主程序的AOEP
- 修改OptionalHEADER中的AddressOfEntryPoint，让它指向新加节的入口代码

加长最后一节修改PE

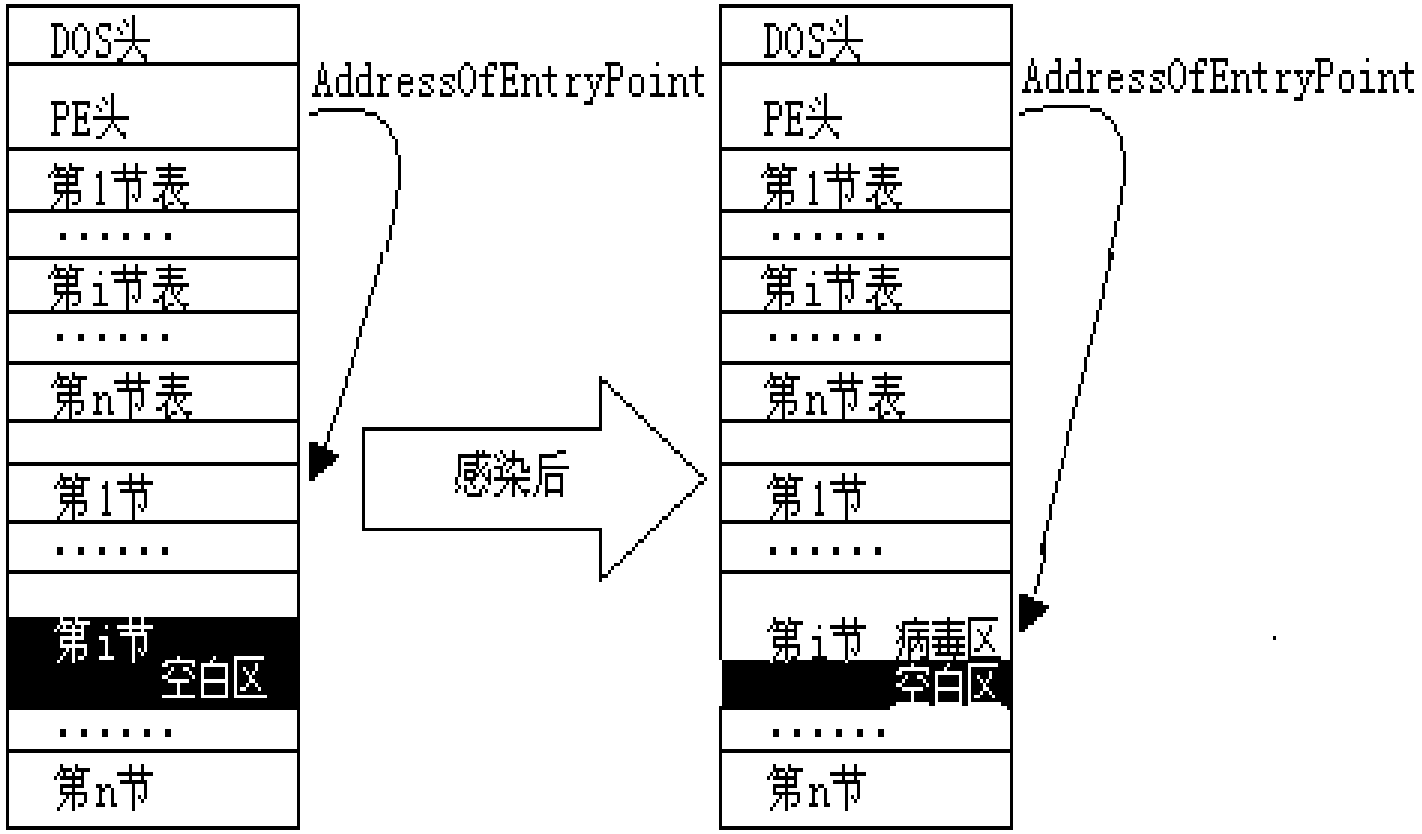

DOS头
PE头
第1节表
.....
最后节表
第1节
.....
最后节
AddressOfEntryPoint
感染后
DOS头
PE头
第1节表
.....
最后节表
第1节
.....
最后节
(原来的+病毒的)
AddressOfEntryPoint

加长最后一节修改PE

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 先把病毒代码追加到最后一个节的尾部
- 修改节表中最后一项section header并增加SizeOfRawData 的大小和内存布局大小

插入节方式修改PE



这种方式不增加节的个数和文件长度，病毒搜寻到一个可执行文件后，分析每个节，查询节的空白空间是否可以容纳病毒代码，若可以，则感染之。CIH病毒就是采用这种方法感染可执行文件的