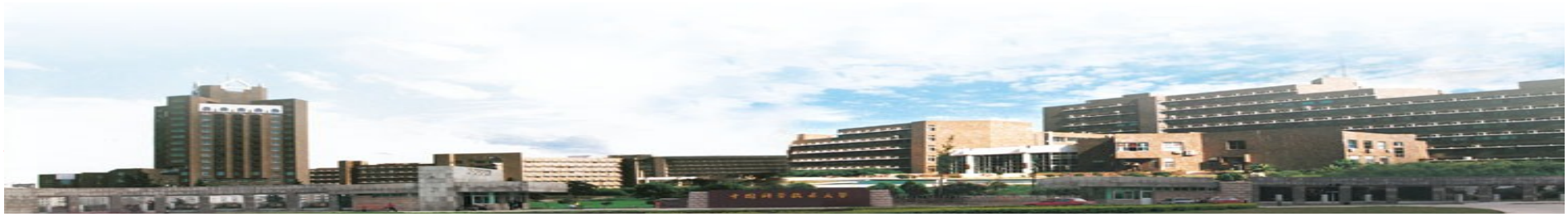
恶意代码分析

程绍银

sycheng@ustc.edu.cn



大纲

- 恶意软件的分类和区别
- 病毒的机理与防治
- 蠕虫的机理与防治
- 木马的机理与防治
- 其他恶意代码的机理
- 恶意代码分析技术

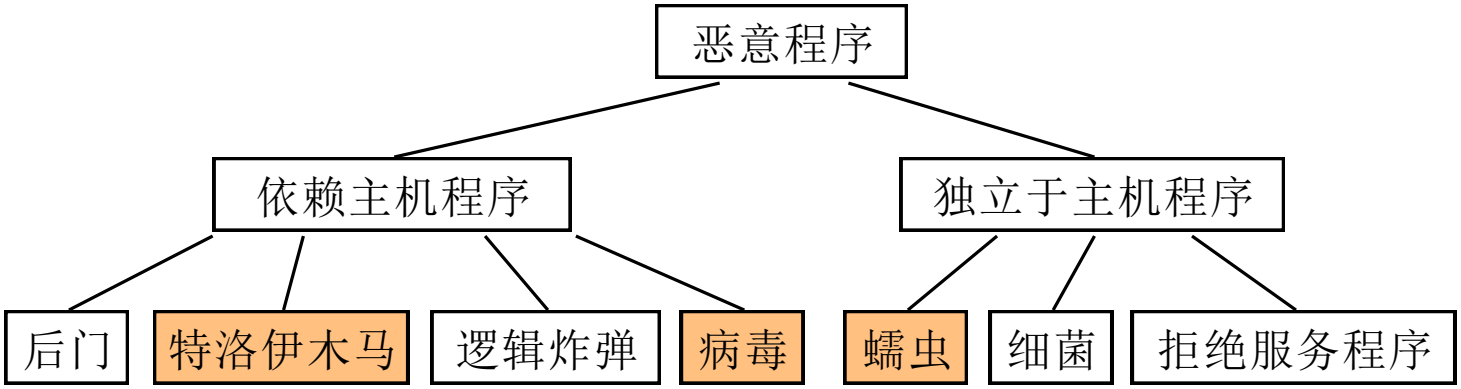
大纲

- ✓ 恶意软件的分类和区别
- 病毒的机理与防治
- 蠕虫的机理与防治
- 木马的机理与防治
- 其他恶意代码的机理
- 恶意代码分析技术

恶意软件的分类和区别

- 恶意软件的定义
  - 恶意软件，统称其行为损害系统用户和系统所有者利益的软件，是故意在计算机系统上执行恶意任务的恶意代码的集合
- 恶意软件的分类（从主机依赖的角度）
  - 依赖主机程序的恶意软件：不能独立于应用程序或系统程序，即存在宿主文件，必须依赖宿主的运行而启动
  - 独立主机的恶意软件：能在操作系统上运行的、独立的程序

恶意软件的分类



- 病毒
- 木马
- 蠕虫

- 后门
- 逻辑炸弹
- 细菌
- 拒绝服务攻击程序
- 垃圾邮件发送程序

病毒（Virus）

- 如果恶意代码将其自身的副本添加到文件、文档或磁盘驱动器的启动扇区来进行复制，则被认为是病毒
- 病毒代码的明显特征是自行复制
- 病毒通常会将其包含的负载（如木马）放置在一个本地计算机上，然后执行一个或多个恶意操作（如删除用户数据）
- 另外，仅进行复制而不具有负载的病毒仍然是恶意软件，因为该病毒自身在复制时可能会损坏数据、消耗系统资源并占用网络带宽

蠕虫（Worm）

- 如果代码在没有携带者（宿主文件）的情况下复制，则被是认为是蠕虫
- 蠕虫试图将自己复制到宿主计算机上，然后利用此计算机的通信信道进行复制
- 病毒寻找文件以进行感染，但蠕虫仅尝试复制其自身

- 重新开机？

木马

- 木马与病毒或者蠕虫的区别在于它不进行复制（传播）。但是病毒和蠕虫可用于将木马作为攻击负载的一部分复制到目标系统上
- 木马通常的意图是在系统中提供后门，使攻击者可以窃取数据
- 关于木马的术语
  - 特洛伊木马
  - 远程访问特洛伊
  - Rootkit

木马

- 远程访问特洛伊：某些木马程序使攻击者可以远程控制系统，此类程序称为“远程访问特洛伊”(Remote Access Trojan, RAT)。RAT的示例包括Back Orifice、Cafeene和SubSeven
- Rootkit：Rootkit是一组高级软件工具程序的集合，攻击者可用于获取对计算机的未经授权的远程访问权限并发动其它攻击。这些程序可能使用许多高级的技术，能够监控系统运行状态，包括监视击键、更改系统日志文件、在系统中创建后门以及对网络上的其它计算机发起攻击

rootkit

- 特殊恶意软件，功能：在安装目标上隐藏自身及指定的文件、进程和网络链接等信息
- 目的：隐藏踪迹和保留root访问权限
- 使用：一般和木马、后门等其他恶意程序结合使用
- 原理：通过加载特殊的驱动，修改系统内核，进而达到隐藏信息的目的
- 反rootkit工具
  - IceSword冰刃：中科大0010，PJF
  - DarkSpy：中科大00信安，CardMagic

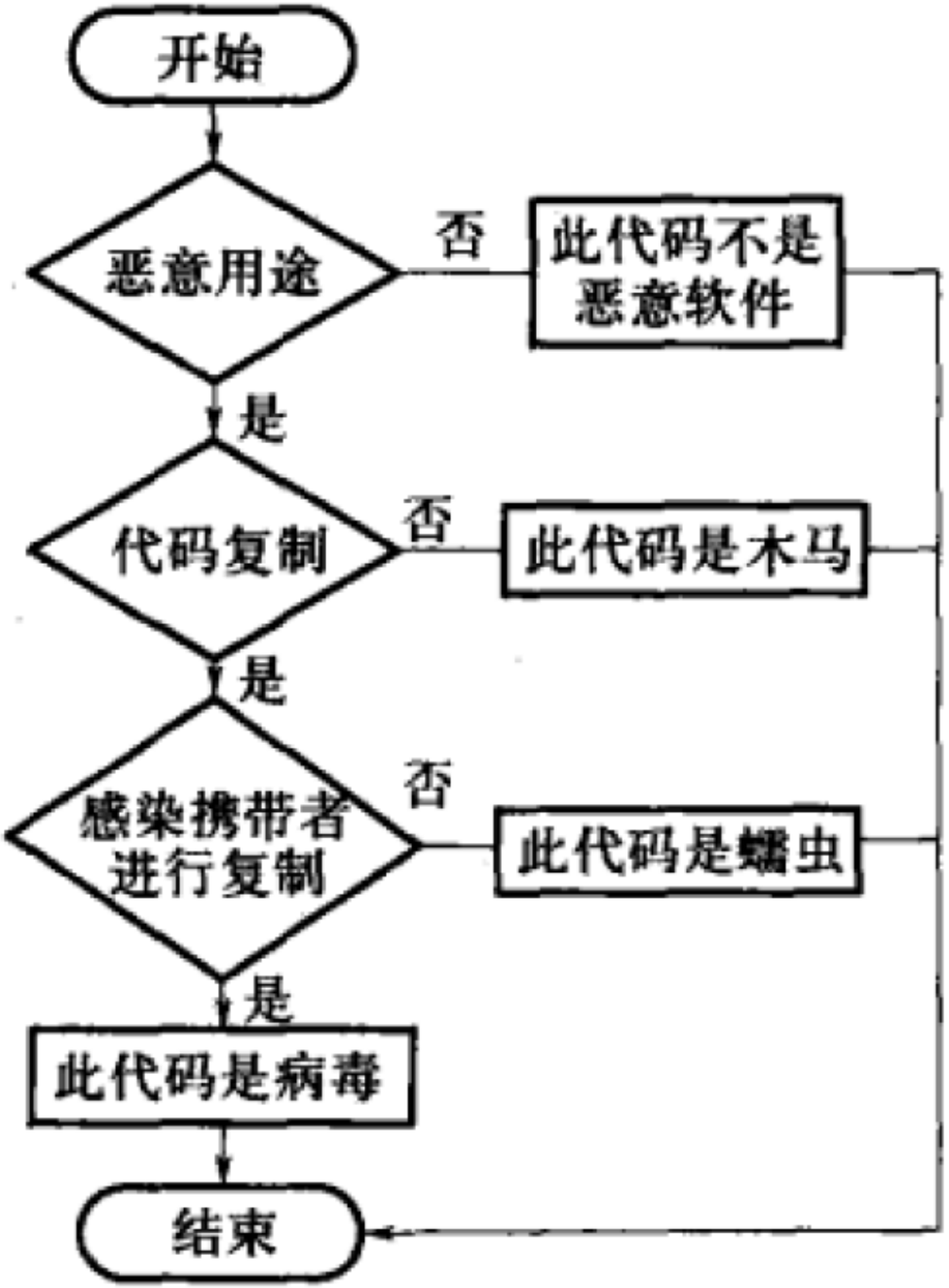
Bootkit

- 更高级的Rootkit，最早于2005年被eEye Digital公司的“BootRoot"项目提及，通过感染MBR(磁盘主引记录）的方式，实现绕过内核检查和启动隐身
  - 广义：所有在开机时比Windows内核更早加载，实现内核劫持的技术
  - 例如：BIOS Rootkit、Vbootkit、SMM Rootkit等
- BootKit具有以下几个特点：
  - 在Ring3下可完成Hook(改写NTLDR)
  - 注入内核的代码没有内存大小限制，也无需自己读入代码
  - BootDriver驱动初始化时加载(依情况而定，也可hook内核其它地方)
  - 理论上可以Hook各种版本ntldr
  - 理论上可以引导各个版本NT内核和内存相关boot.ini参数

Windows Resource Kit tools

- 一组为管理员、开发者和高级用户设计的软件工具
- 包括管理活动目录、组策略、TCP/IP网络、注册表、系统安全、监测等涉及Windows 操作系统的其它很多方面的非常规安装的工具组
- http://en.wikipedia.org/wiki/Resource_Kit
- http://technet.microsoft.com/library/bb968968.aspx

- 病毒、蠕虫、木马的区别


开始
恶意用途
否
此代码不是恶意软件
是
代码复制
否
此代码是木马
是
感染携带者进行复制
否
此代码是蠕虫
是
此代码是病毒
结束

其他恶意软件

- 逻辑炸弹（Logic Bomb）
  - 合法的应用程序，只是在编程时被故意写入某种“恶意功能（malfunction）”，在一定程度下（如时间，次数，或者某种逻辑组合）会出现
  - 例如，作为版权保护方案，某个应用程序有可能会在运行几次后就在硬盘中将其自身删除
  - 另外，程序中可能隐藏额外的功能，例如在大型项目中加入复活节彩蛋（Easter Eggs），其目的是为了隐藏一些额外的荣誉页面（credit page）来显示项目组的成员。微软原Office套装软件中的程序隐藏着许多复活节彩蛋。尽管这些内容不是恶意的，但会占用额外的硬盘空间

其他恶意软件

- 后门
  - 后面是恶意攻击者选择用来远程连接系统的工具。典型的后门会在运行它的主机上打开一个网络端口，然后侦听的后面程序会等待攻击者的远程连接
  - 瑞士军刀 netcat
- 细菌（germ）
  - 细菌是第一代病毒，是病毒还不能进行感染的一种形式。通常，当病毒第一次被编译时，是以一种特殊的形式存在地，一般是不会附着在宿主程序上的。细菌不会像第二代病毒那样标记感染的文件，以避免重复感染。加密病毒或者多态病毒（polymorphic virus）的细菌通常不会被加密，而是以明文的、可读的代码形式存在。细菌的检测方法与检测第二代或以后各代的病毒有所不同

其他恶意软件

- 垃圾邮件发送程序(spammer)
  - 用来向即时消息组、新闻组或者其它任何种类的移动设备发送未经请求的消息
  - 形式：使用电子邮件或者手机短信
  - 动机：通过增加web站点的点击率来赚钱
  - 每天1200亿封垃圾邮件
  - 每年1400亿美元的反垃圾邮件市场
- 另外，常被用来网络钓鱼（phishing）攻击

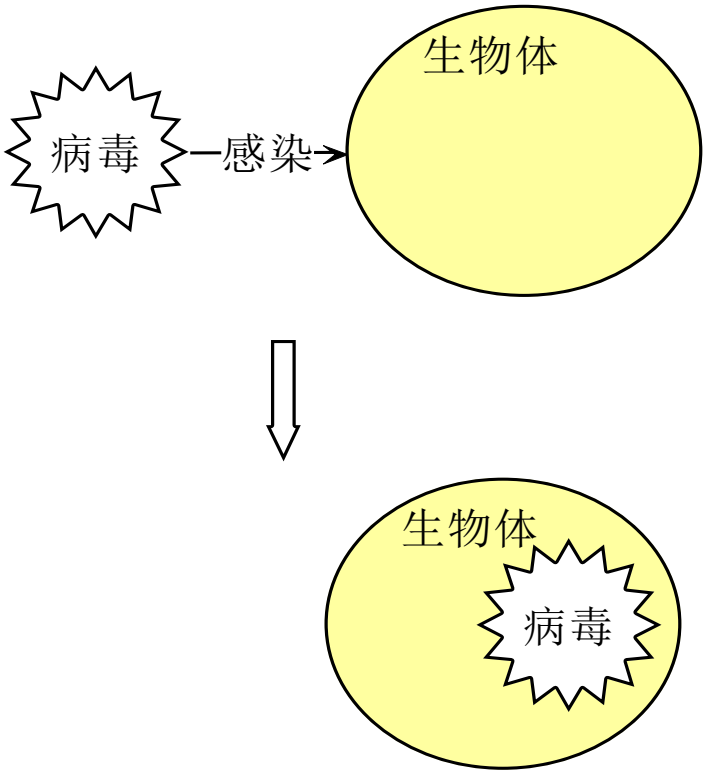
大纲

- 恶意软件的分类和区别
- ✓ 病毒的机理与防治
- 蠕虫的机理与防治
- 木马的机理与防治
- 其他恶意代码的机理
- 恶意代码分析技术

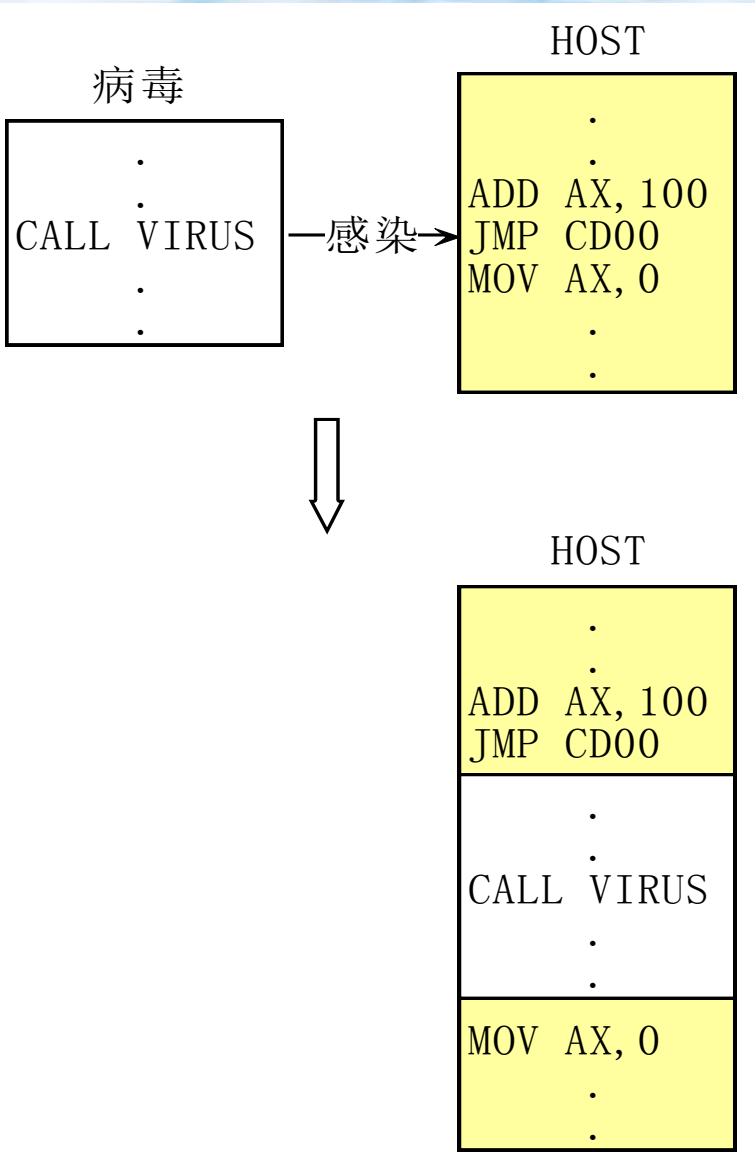
病毒的定义

- 病毒(virus)是一种人为制造的、能够进行自我复制的、具有对计算机资源的破坏作用的一组程序和指令的集合
- 病毒与生物病毒一样，有其自身的病毒体（病毒程序）和寄生体（宿主HOST，病毒载体，携带者）
- 所谓感染或寄生，是指病毒将自身嵌入到宿主指令序列中
- 寄生体为病毒提供一种生存环境，是一种合法程序。病毒程序寄生于合法程序后，成为了程序的一部分，并随着合法程序的执行而执行，也随着合法程序的消失而消失

病毒感染示意



生物病毒感染与寄生



计算机病毒感染与寄生

病毒的定义

- 常见的目标携带者有：
  - 可执行文件。通过其自身附加到宿主程序进行复制的“典型”病毒类型的目标对象。例如.exe, .com, .sys, .dll, .ocx和.prg文件
  - 脚本。例如.vbs, .js, .wsh, .prl文件
  - 宏。携带者是宏脚本语言的文件
  - 启动扇区。计算机磁盘上的特定区域（例如，主启动记录（MBR）也可以作为携带者，因为它可以执行恶意代码。当某个磁盘被感染时，如果使用该磁盘来启动其它计算机系统，将会复制病毒

病毒的分类

- 按照病毒攻击的操作系统分类
  - DOS病毒。1995年前，广泛使用DOS系统
  - Windows病毒。例如CIH病毒
  - UNIX病毒。1997年，出现首例病毒Bliss。2001年，出现首例Windows和Linux都能传播的病毒Win32.Winux
  - OS/2病毒。1996年，AEP病毒
  - Macintosh病毒。Mac.simpsons病毒
  - 其它操作系统病毒。手机。。。

病毒的分类

- 按照病毒的链接方式分类
  - 源码型病毒
  - 嵌入型病毒
  - 外壳病毒
  - 译码型病毒
  - 操作系统型病毒

（1）源码型病毒。该病毒攻击高级语言（如 C、FORTRAN、PASCAL 等）编写的程序。在编译用高级语言编写的程序之前，将病毒代码插入到源程序中，经编译成为合法程序的一部分。这类病毒一般存在于语言处理程序和链接程序中。首例感染 C 语言和 PASCAL 语言源代码的病毒是 Srevir 病毒。

（2）嵌入型病毒，也称为入侵型病毒。该类病毒将自身嵌入到已有程序中，把病毒的主体程序与其攻击对象以插入方式链接，并代替其中部分不常用的功能模块或堆栈区。这种病毒较难发现。

（3）外壳病毒。通常附在宿主程序的首部或者尾部，对原来的程序不做修改（若寄生在尾部，则修改程序的第一条可执行指令，使病毒能先于宿主程序执行，控制主动权以便传播蔓延），相当于给宿主程序加了个外壳。这种病毒最为常见，易于发现和清除。

（4）译码型病毒。隐藏在微软 Office、AmiPro 等文档中，如宏病毒、脚本（VBS/JS）病毒等，此类病毒一般是解释执行。

（5）操作系统型病毒。这种病毒用自己的程序试图加入或取代部分操作系统功能进行工作，具有很强的破坏力，可导致整个系统的瘫痪，如圆点病毒和大麻病毒。这种病毒在运行时，用自己的逻辑部分取代操作系统的合法程序模块，根据病毒自身的特点和被替代的操作系统中合法程序模块运行的作用以及病毒取代操作系统的方式等，对操作系统进行破坏。

病毒的分类

- 病毒的寄生存储的位置分类
  - 引导型病毒，也称为开机病毒，引导区病毒
  - 文件型病毒，文件型病毒主要感染可执行文件，如扩展名为.EXE、.COM、.OVL的文件，是一种较为常见的病毒
  - 目录型病毒是文件型病毒的一种特例，其感染方式非常独特，仅修改目录区，便可达到感染的目的，如DIR2病毒。宏病毒则是一种数据文件型病毒
  - 混合型病毒，也称为多型病毒，是综合了引导型和文件型病毒特征的病毒，可感染文件和引导扇区两种目标

计算机病毒的基本特征

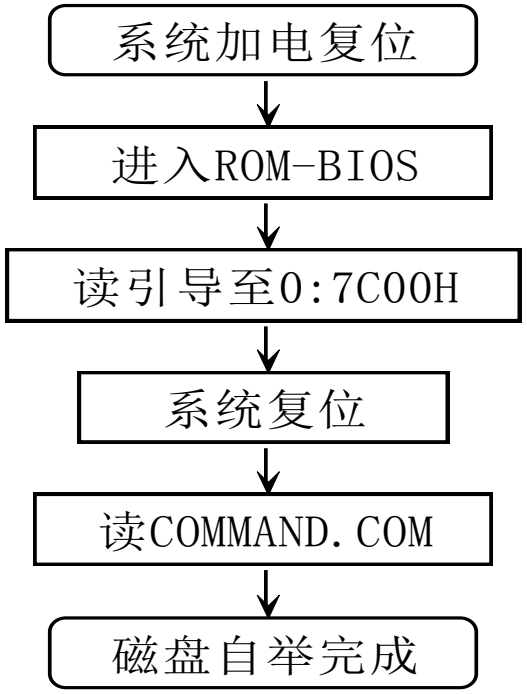
- 计算机病毒攻击的主动性
  - 计算机病毒对系统的攻击是主动的，是不以人的意志为转移的
- 计算机病毒的针对性
  - 要使计算机病毒得以运行，就必须具有适合该病毒发生作用的特定软硬件环境
- 计算机病毒的衍生性
  - 衍生性为一些好事者提供了一种创造新病毒的捷径。衍生出来的变种病毒造成的后果可能比原版病毒严重
  - 衍生性，是导致产生变体病毒的必然原因

计算机病毒的基本特征

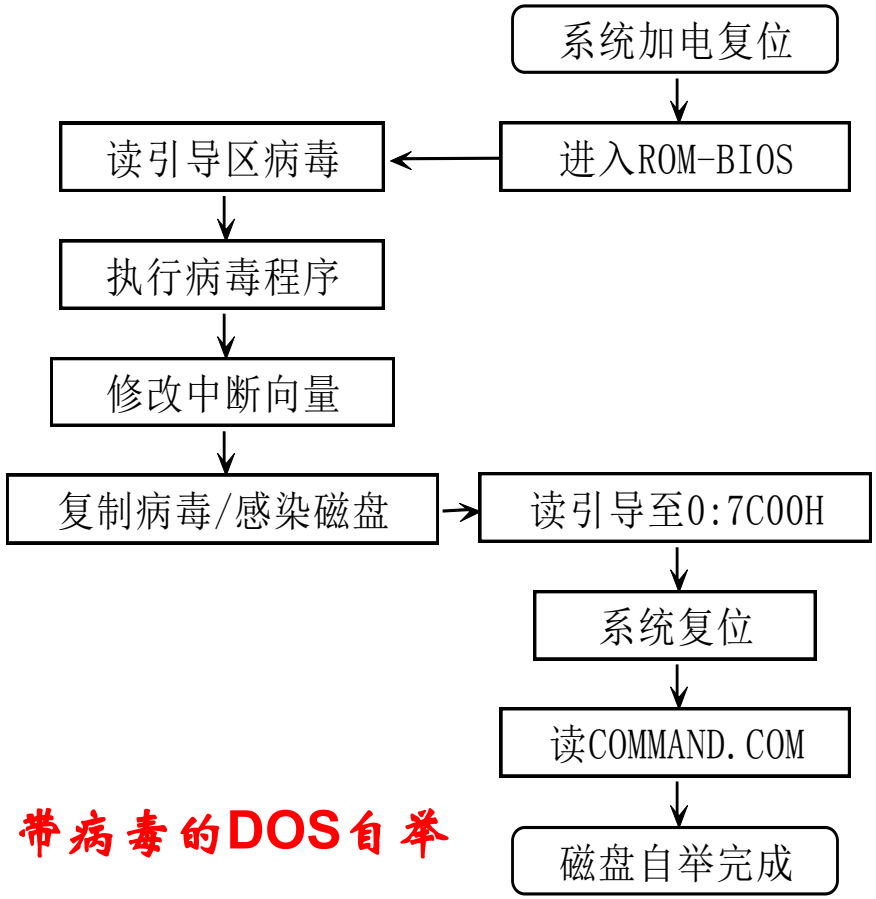
- 计算机病毒的寄生性(依附性)
  - 病毒程序嵌入到宿主程序中，依赖于宿主程序的执行而生存，这就是计算机病毒的寄生性
- 计算机病毒的不可预见性
  - 反病毒软件预防措施和技术手段往往滞后于病毒的产生速度
- 计算机病毒的诱惑欺骗性
  - 某些病毒常以某种特殊的表现方式，引诱、欺骗用户不自觉地触发、激活病毒，从而实施其感染、破坏功能
- 计算机病毒的持久性
  - 即使在病毒程序被发现以后，数据和程序以至操作系统的恢复都非常困难

计算机病毒的本质

- 计算机病毒利用计算机系统使用过程中出现的频繁中断或操作，通过修改地址，使病毒指令程序插入中断程序与正常程序之间



正常DOS自举



带病毒的DOS自举

病毒的通用命名规则

- 按病毒发作的时间命名
  - 如“黑色星期五”
- 按病毒发作症状命名
  - 如“小球”病毒
- 按病毒的传染方式命名
  - 如黑色星期五病毒，又命名为疯狂拷贝病毒
- 按病毒自身宣布的名称或包含的标志命名
  - CIH病毒的命名源于其含有“CIH”字符
- 按病毒发现地命名
  - 如“黑色星期五”又称Jerusalem(耶路撒冷)病毒
- 按病毒的字节长度命名
  - 如黑色星期五病毒又称作1813病毒
- 思考：这种命名方式，存在什么不足？

国际上对病毒命名的惯例

- 计算机病毒英文命名规则也就是国际上对病毒命名的一般惯例为“前缀+病毒名+后缀”，即三元组命名规则
  - 前缀表示该病毒发作的操作平台或者病毒的类型，而DOS下的病毒一般是没有前缀
  - 病毒名为该病毒的名称及其家族
  - 后缀一般可以不要，只是以此区别在该病毒家族中各病毒的不同，可以为字母，或者为数字以说明此病毒的大小
- 三元组中“病毒名”的命名优先级为：
  - 病毒的发现者(或制造者)→病毒的发作症状→病毒的发源地→病毒代码中的特征字符串
- 例如：WM.Cap.A
  - A表示在Cap病毒家族中的一个变种，WM表示该病毒是一个Word宏(Macro)病毒
- 病毒名中若有PSW或者PWD之类的，一般都表示该病毒有盗取口令的功能

病毒命名亟待进一步规范、统一

- 由于存在“灵活”的命名规则和惯例，再加上杀毒软件开发商各自的命名体系存在差异、计算机病毒研究学者/反病毒人员在为病毒命名时的个人观点、所依据的方法也各不相同，最终造成同种病毒出现不同名称的混乱现象
  - 如“新欢乐时光”病毒
    - HTML.Redlof.A [Symantec]
    - VBS.KJ [金山]
    - Script.RedLof [瑞星]
    - VBS/KJ [江民]
- 病毒命名可以做出更细致的规定，如：
  - [病毒前缀]+[主要变量]+[次要变量]+[病毒名]+[病毒后缀]
  - 比如一个病毒名为：DosVirus.com.BOOT.kot.B，那么可以解释为：这是一个DOS病毒，仅仅感染.COM，感染引导区，病毒名为kot，版本号为B。

病毒的逻辑结构与基本机制

- 病毒的状态
- 病毒的基本环节
- 病毒的逻辑结构
- 病毒的基本机制

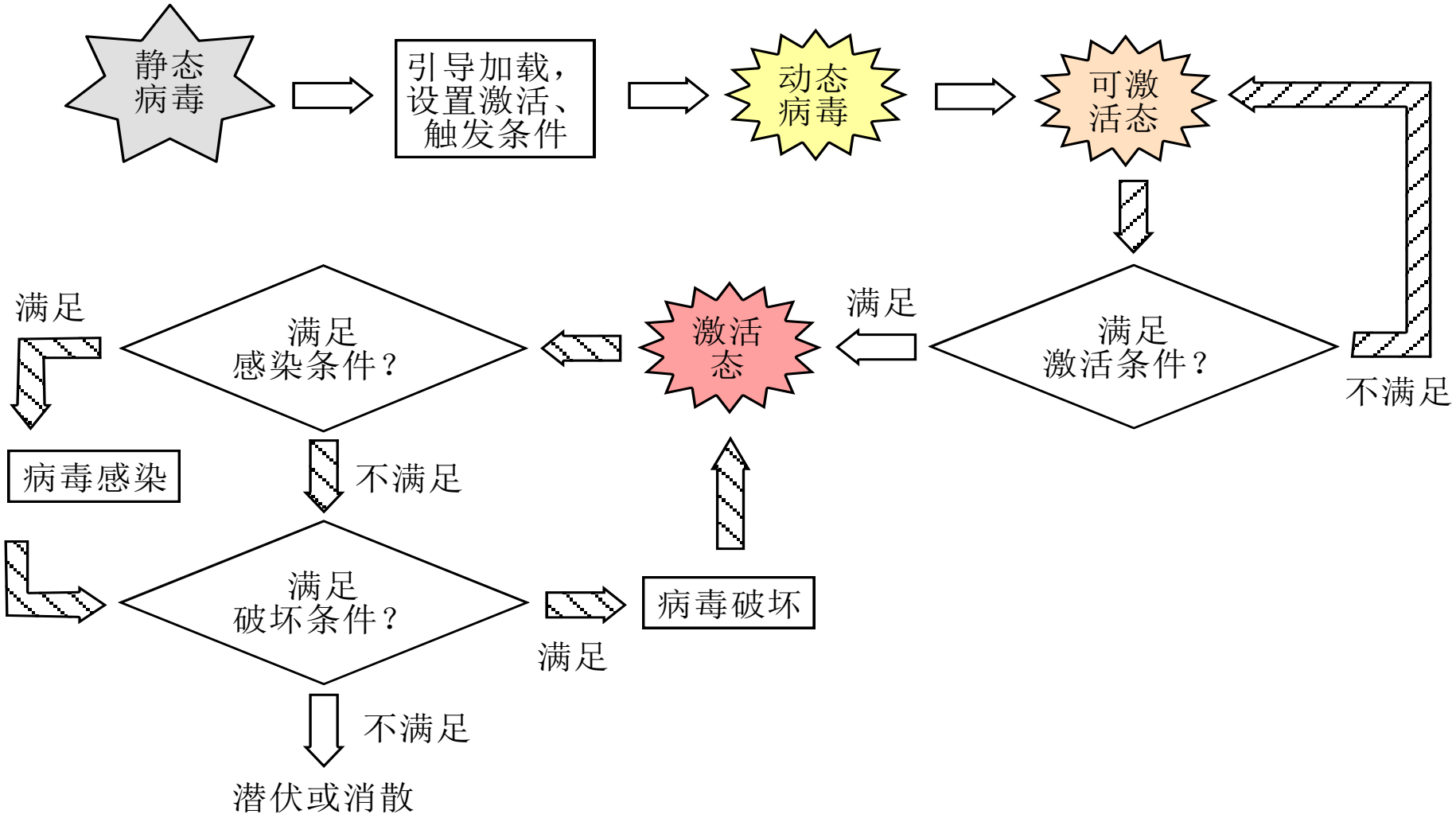
计算机病毒的状态

- 计算机病毒在传播过程中存在两种状态，即静态和动态
- 静态病毒，是指存在于辅助存储介质中的计算机病毒，一般不能执行病毒的破坏或表现功能，其传播只能通过文件下载(拷贝)实现
  - 静态病毒尚未被加载和进入内存，不可能获取系统执行权限
  - 病毒之所以处于静态，有两种可能
    - 没有用户启动该病毒或运行感染了该病毒的文件
    - 该病毒存在于不可执行它的系统中
- 当病毒完成初始引导，进入内存后，便处于动态
- 动态病毒本身处于运行状态，通过截流盗用系统中断等方式监视系统运行状态或窃取系统控制权。病毒的主动传染和破坏作用，都是动态病毒的“杰作”

计算机病毒的状态

- 计算机病毒的基本流程与状态转换
  - 病毒由静态转变为动态的过程，称为病毒的启动。实际上，病毒的启动过程就是病毒的首次激活过程
  - 内存中的动态病毒又有两种状态：可激活态和激活态
    - 当内存中的病毒代码能够被系统的正常运行机制所执行时，动态病毒就处于可激活态
    - 一般而言，动态病毒都是可激活的
    - 系统正在执行病毒代码时，动态病毒就处于激活态
    - 病毒处于激活态时，不一定进行传染和破坏；但进行传染和破坏时，必然处于激活态

计算机病毒的状态


静态
病毒
引导加载，
设置激活、
触发条件
动态
病毒
可激
活态
满足
激活
态
满足
激活条件？
不满足
满足
感染条件？
满足
病毒感染
不满足
满足
破坏条件？
满足
病毒破坏
不满足
潜伏或消散

计算机病毒的状态

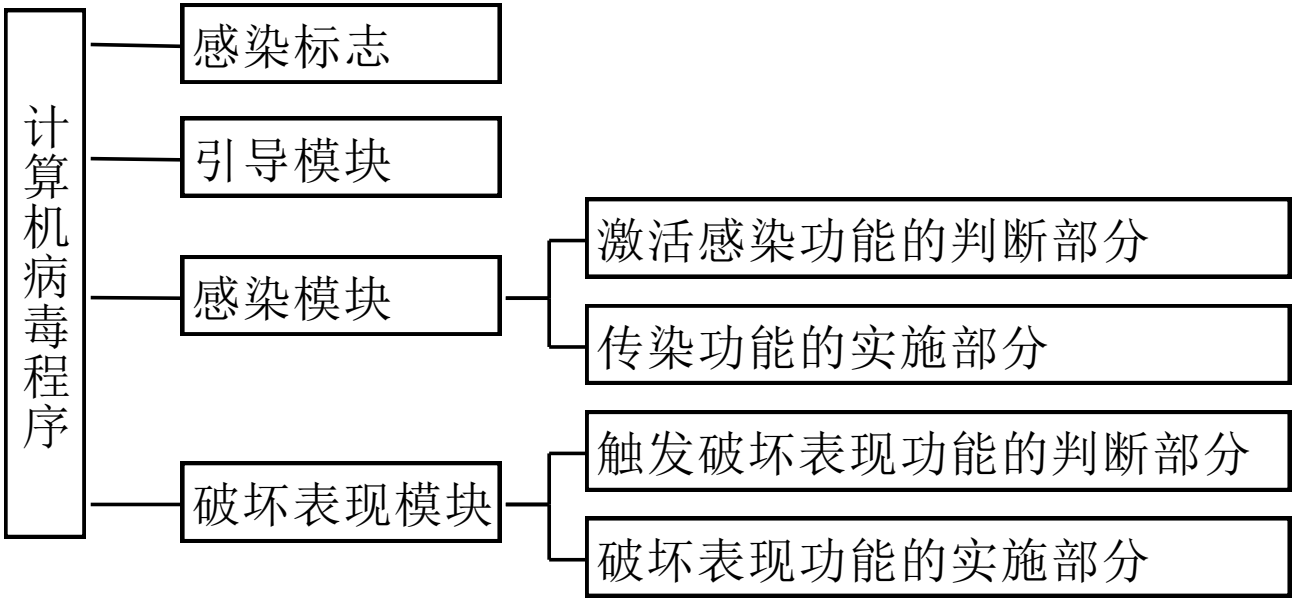
- 内存中的病毒还有一种较为特殊的状态——失活态
  - 一般情况下不会出现这种状态，它的出现一般是由于用户对病毒的干预(用杀毒软件或手工方法)
  - 处于失活态的病毒不可能进行传染或破坏，它与静态病毒的不同仅在于病毒代码在内存中，但得不到执行
  - 如果用户把中断向量表恢复成正确值，修改中断向量表的动态病毒就失活了
  - 病毒能由激活态转变为失活态，也就是可激活态病毒的可触发性被破坏，处于激活态的病毒一般不会自己转变为失活态，失活态的出现必定是有外在干预
- 对于处于不同状态的病毒，应采用不同的分析、清除手段

计算机病毒的基本环节

- 计算机病毒要完成一次完整的传播破坏过程，必须经过以下几个环节：
  - 分发拷贝阶段
  - 潜伏繁殖阶段
  - 破坏表现阶段
- 在任何一个环节(阶段)都可以抑制病毒的传播、蔓延，或者清除病毒
- 我们应当尽可能地在病毒进行破坏性攻击之前切断病毒传染源、抑制病毒的传播蔓延

计算机病毒的逻辑结构

- 计算机病毒是以现代计算机网络系统为环境而存在并发展的，即计算机系统的软、硬件环境决定了计算机病毒的结构，而这种结构是能够充分利用系统资源进行活动的最合理体现



- 有时也把破坏表现模块中触发条件判断部分作为一个单独的模块，称作触发模块

计算机病毒的逻辑结构

- 感染标志
  - 有的病毒有一个感染标志，又称病毒签名，但不是所有的病毒都有感染标志
  - 感染标志是一些数字或字符串，它们以ASCII码方式存放在宿主程序程序里
  - 病毒在感染程序之前，一般要查看其是否带有感染标志
  - 感染标志不仅被病毒用来决定是否实施感染，还被病毒用来实施欺骗
  - 不同病毒的感染标志的位置、内容都不同
  - 杀毒软件可以将感染标志作为病毒的特征码之一
  - 也可以利用病毒根据感染标志是否进行感染这一特性，人为地、主动在文件中添加感染标志，从而在某种程度上达到病毒免疫的目的

计算机病毒的逻辑结构

- 引导模块
  - 染毒程序运行时，首先运行的是病毒的引导模块
  - 引导模块的基本动作是：
    - 检查运行的环境，如确定操作系统类型、内存容量、现行区段、磁盘设置、显示器类型等参数
    - 将病毒引入内存，使病毒处于动态，并保护内存中的病毒代码不被覆盖
    - 设置病毒的激活条件和触发条件，使病毒处于可激活态，以便病毒被激活后根据满足的条件调用感染模块或破坏表现模块

计算机病毒的逻辑结构

- 感染模块
  - 是病毒实施感染动作的部分，负责实现病毒的感染机制
  - 感染模块的主要功能如下：
    - 寻找感染目标
    - 检查目标中是否存在感染标志或设定的感染条件是否满足
    - 如果没有感染标志或条件满足，进行感染，将病毒代码放入宿主程序
  - 无论是文件型病毒还是引导型病毒，其感染过程总的来说是相似的，分为三步：进驻内存、判断感染条件、实施感染
    - 感染条件控制病毒的感染动作、控制病毒感染的频率：频繁感染，容易让用户发觉；苛刻的感染条件，又让病毒放弃了众多传播机会

计算机病毒的逻辑结构

- 破坏模块
  - 负责实施病毒的破坏动作，其内部是实现病毒编写者预定破坏动作的代码
  - 病毒的破坏力取决于破坏模块
  - 破坏模块导致各种异常现象，因此，该模块又被称为病毒的表现模块
  - 计算机病毒的破坏现象和表现症状因具体病毒而异。计算机病毒的破坏行为和破坏程度，取决于病毒编写者的主观愿望和技术能力
  - 触发条件控制病毒的破坏动作，控制病毒破坏的频率，使病毒在隐蔽的状态下实施感染
  - 病毒的触发条件多种多样，例如，特定日期触发、特定键盘按键输入，等等，都可以作为触发条件

病毒实施感染的前提条件

- 病毒感染的必要条件是病毒代码在本次启动计算机后，至少被执行过一次
- 病毒感染还有一个必要条件，即必须要有传染目标——病毒宿主
- 病毒在以下情况下有可能被首次执行
  - 染有引导型病毒的磁盘在启动计算机时
  - 文件型病毒的病毒代码在执行染毒文件时被执行
  - 初始化批处理启动病毒，或利用系统初始化配置文件的启动项启动病毒
  - Win32病毒借助Windows注册表的特殊键值，在启动Windows时随之启动

病毒的感染对象和感染过程

- 感染对象
  - 寄生在磁盘引导扇区
  - 寄生在可执行文件
  - 宏病毒和脚本病毒是比较特殊的病毒，是通过打开Office文档或浏览网页等用户行为而获取执行权
  - 某些广义下的病毒，如蠕虫，主要寄生在内存中，并不感染引导扇区和文件

病毒的感染对象和感染过程

- 传染方式
  - 所谓传染，是指计算机病毒由一个载体传播到另一个载体，由一个系统进入另一个系统的过程
  - 只有载体还不足以使病毒得到传播。促成病毒的传染还有一个先决条件，可分为两种情况，或者称作两种方式
    - 被动传染：用户在进行拷贝磁盘或文件时，把一个病毒由一个载体复制到另一个载体上；或者是通过网络上的信息传递，把一个病毒程序从一方传递到另一方
    - 主动传染：计算机病毒是以计算机系统的运行以及病毒程序处于激活态为先决条件。当病毒处于激活态时，只要传染条件满足，病毒程序就能主动地把病毒自身传染给另一个载体或另一个系统

病毒的感染对象和感染过程

- 传染过程
  - 对于病毒的被动传染而言，其传染过程是随着拷贝磁盘或文件工作的进行而进行的
  - 对于计算机病毒的主动传染而言，其传染过程一般是：
    - 在系统运行时，病毒通过病毒载体即系统的外存储器进入系统的内存储器，常驻内存，并在系统内存中监视系统的运行
    - 在病毒引导模块将病毒传染模块驻留内存的过程中，通常还要修改系统中断向量入口地址，使该中断向量指向病毒程序传染模块
    - 当系统执行磁盘读写操作或系统功能调用，病毒传染模块就被激活，传染模块在判断传染条件满足的条件下，把病毒自身传染给被读写的磁盘或被加载的程序

病毒的感染对象和感染过程

- 传染过程
  - 计算机病毒实施感染的过程基本可分为两大类
    - 立即传染：病毒在被执行到的瞬间，抢在宿主程序执行前，立即感染磁盘上的其他程序，然后再执行宿主程序
    - 驻留内存并伺机传染：内存中的病毒检查当前系统环境，在执行一个程序或DIR等操作时感染磁盘上的程序，驻留在系统内存中的病毒程序在宿主程序运行结束后，仍可活动，直至关闭计算机
  - 总之，计算机病毒感染的过程一般有三步：
    - (1)当宿主程序运行时，截取控制权
    - (2)寻找感染的突破口
    - (3)将病毒代码放入宿主程序
  - 病毒攻击的宿主程序是病毒的栖身地，宿主程序既是病毒传播的目的地，又是下一次感染的出发点

引导型病毒传染机理

- 引导型病毒针对软硬盘的不同特点采用了不同的传染方式
  - 引导型病毒利用在开机引导时窃取的INT 13H控制权，在整个计算机运行过程中随时监视软盘操作情况，趁读写软盘的时机读出软盘引导区，判断软盘是否染毒，如未感染就按病毒的寄生方式把原引导区写到软盘另一位置，把病毒写入软盘第一个扇区，从而完成对软盘的传染
  - 染毒的软盘在软件交流中又会传染其他计算机
  - 引导型病毒对硬盘的传染，往往是在计算机上第一次使用带毒软盘、优盘、移动硬盘时进行的，具体步骤与软盘传染相似，也是读出引导区判断后写入病毒

件型病毒传染机理

- 当执行被传染的.COM或.EXE可执行文件时，病毒进驻内存，始监视系统的运行，当发现被传染的目标时，进行如下操作：
  - 首先对运行的可执行文件特定地址的标识位信息进行判断，判断是否已感染了病毒
  - 当条件满足，利用INT 13H将病毒链接到可执行文件的首部、尾部或中间，并存盘
  - 完成传染后，继续监视系统的运行，试图寻找新的攻击目标

文件型病毒传染机理

- 文件型病毒通过与磁盘文件有关的操作进行传染，主要传染途径有：
  - 加载执行文件
  - 列目录过程
  - 创建文件过程

混合感染和交叉感染

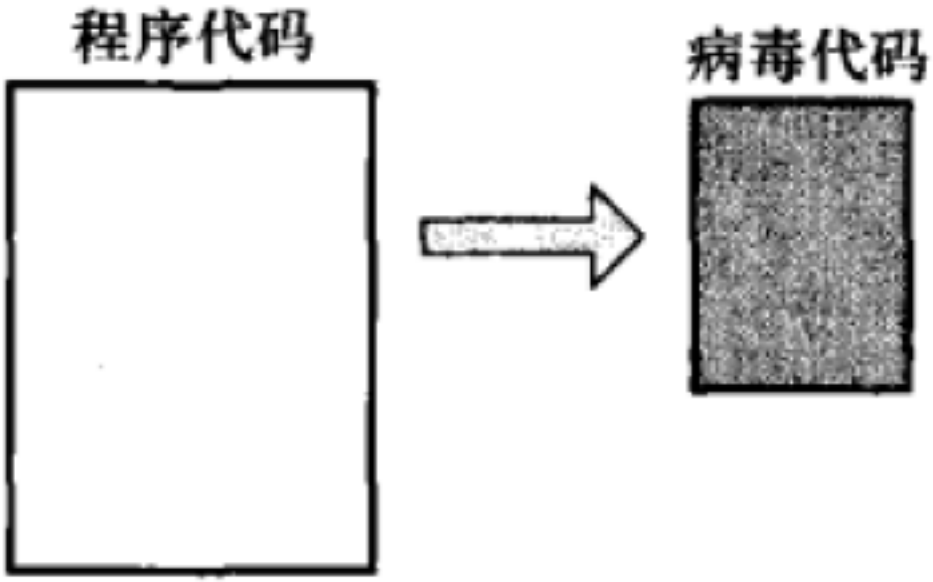
- 混合感染
  - 既感染引导扇区又感染文件
- 交叉感染
  - 一台计算机中常常会同时染上多种病毒。当一个无毒的宿主程序在此计算机上运行时，便会在一个宿主程序上感染多种病毒，称为交叉感染。
  - 当文件感染数种病毒，并且病毒代码在宿主程序头部和尾部都有的情况下，必须分清各病毒的感染先后顺序，否则消毒后程序不能正常运行

文件型病毒的感染技术

- 重写病毒
- 追加病毒
- 前置病毒
- 蛀穴病毒
- 分割型蛀穴病毒
- 压缩型病毒
- 变形虫感染技术
- 嵌入式解密程序技术
- 嵌入式解密程序和病毒体技术
- 迷惑性欺骗跳转技术
- 入口点隐蔽病毒

重写病毒

- 这种病毒从磁盘上找到一个文件，简单地用自己的拷贝改写该文件，是一种较初级的技术
- 重写病毒是不能从系统中彻底删掉的，只能把被感染的文件删掉，然后再从备份介质中恢复
- 重写病毒攻击时改变宿主文件的大小

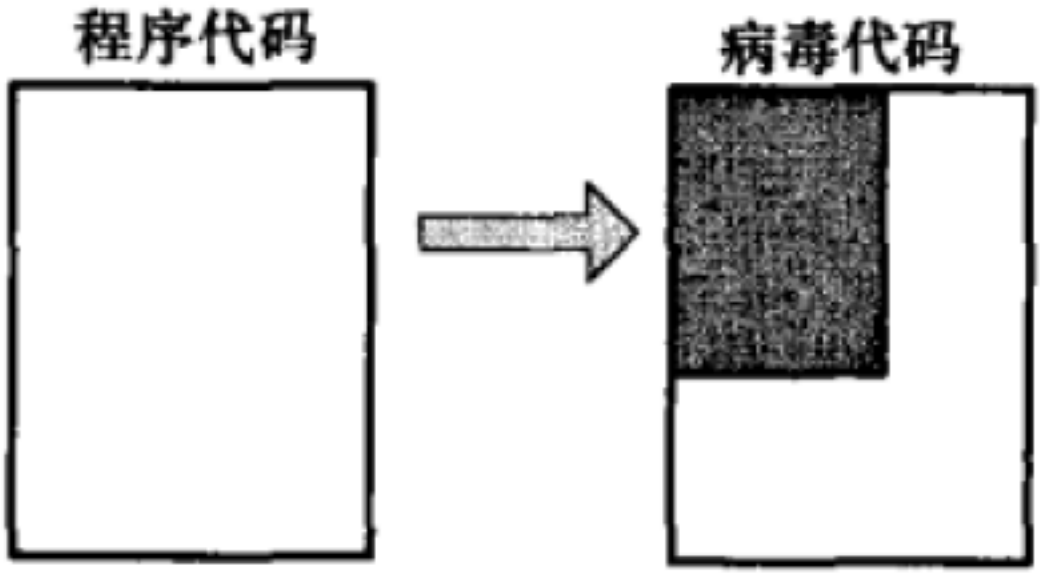


重写病毒

- 另一种重写病毒传染方式用于非常短小的tiny病毒。90年代初，许多病毒作者试图写出最短的病毒。如有些病毒仅22个字节。这种病毒的算法非常简单：
  - 在当前目录下寻找任何新的宿主文件
  - 已写的方式打开文件
  - 把病毒代码写入宿主文件的顶端

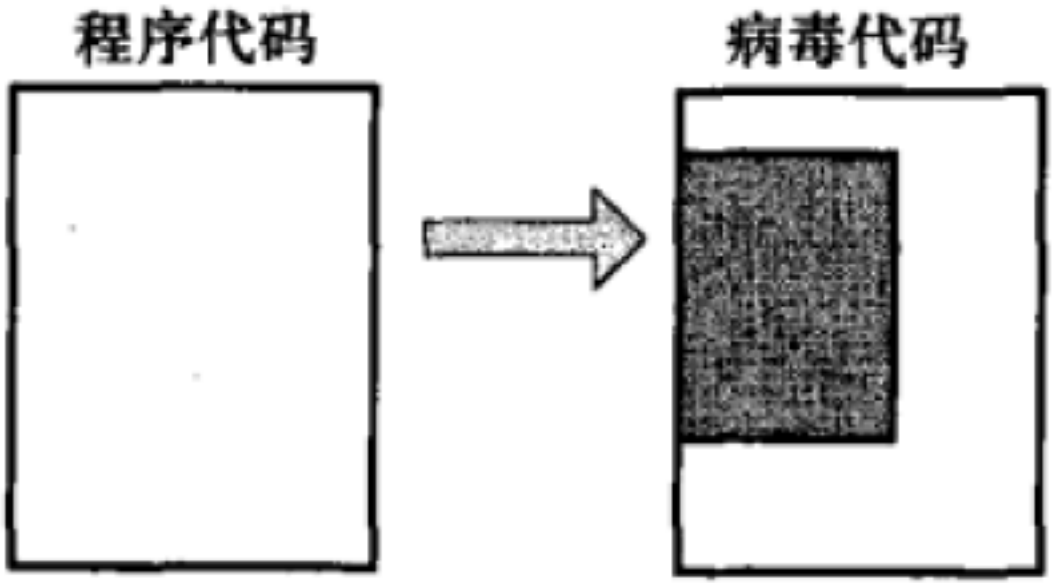
重写病毒

- 重写病毒简单地重写了宿主文件的顶部，而没有改变文件的大小



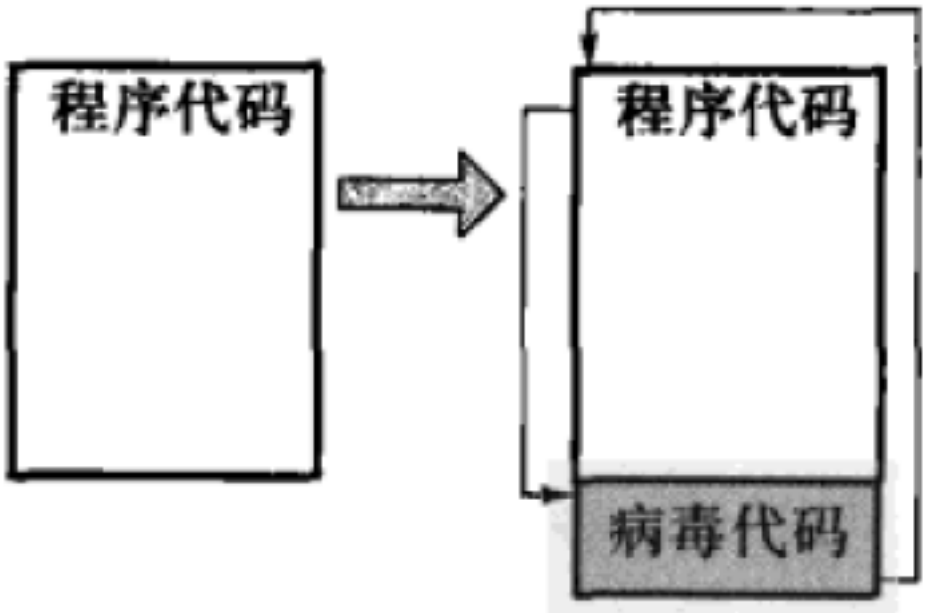
重写病毒

- 随机重写：不改变文件顶部的代码，而是在宿主文件中随机找一个位置把自己写进去。显然，这种病毒不太可能获得控制权，它通常会导致宿主在执行到病毒代码之前就崩溃了，例如Omud病毒



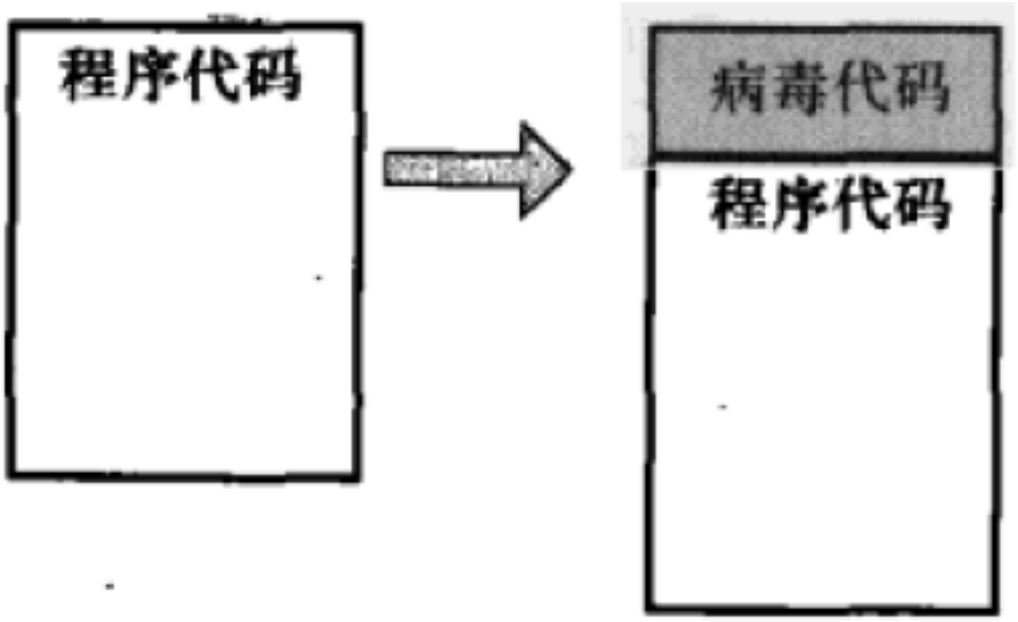
追加病毒

- 一种典型的DOS环境下的COM文件感染技术是在宿主文件的头部插入一条JMP指令，指向初始文件的尾部
- 追加技术可使用在任何类型的可执行文件中
- 病毒将主程序的入口点替换成追加到文件末尾的病毒代码的起始地址



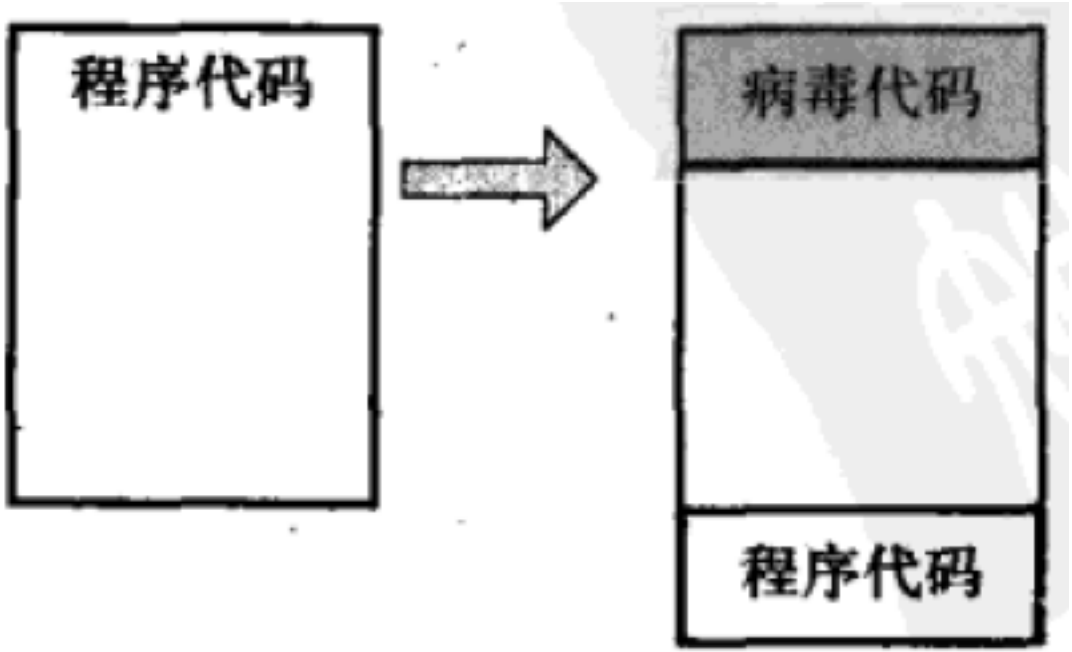
前置病毒

- 把病毒的代码插入到宿主程序的前面，这些代码通常采用高级语言如C、Pascal等实现，通常在磁盘上创建一个包含原文件内容的临时文件，然后用system这样的函数执行临时文件中原来的程序



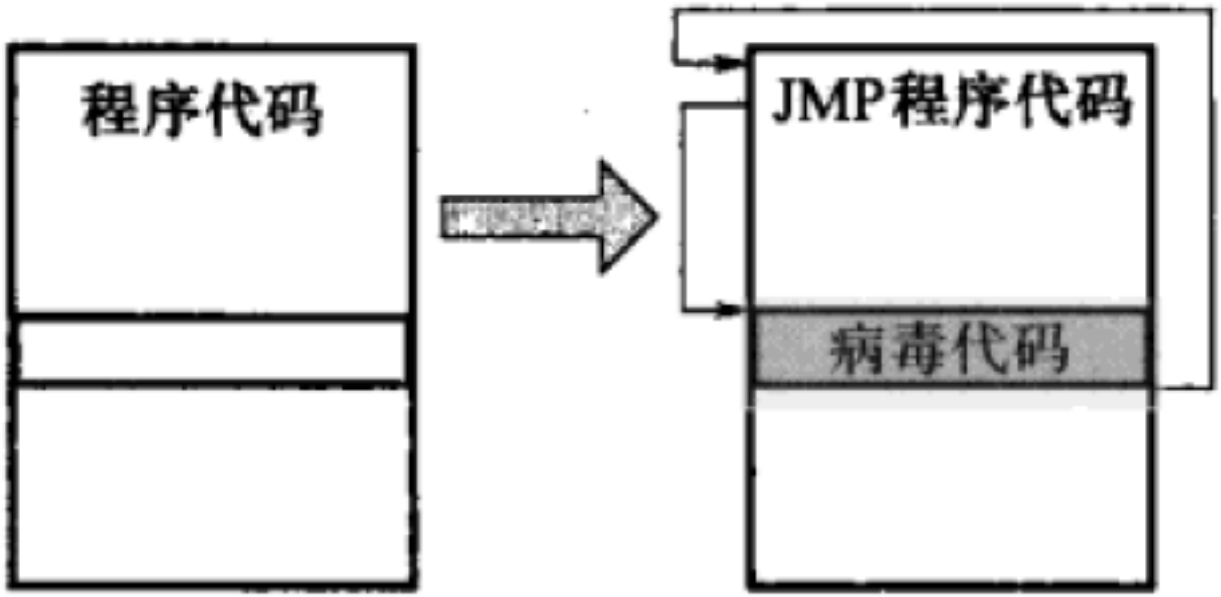
寄生病毒

- 前置病毒的变种为典型寄生病毒
- 这种病毒用自身的代码重写宿主顶部的数据，并把宿主顶部的这些数据存放在宿主程序的最后，长度通常等于病毒体的长度



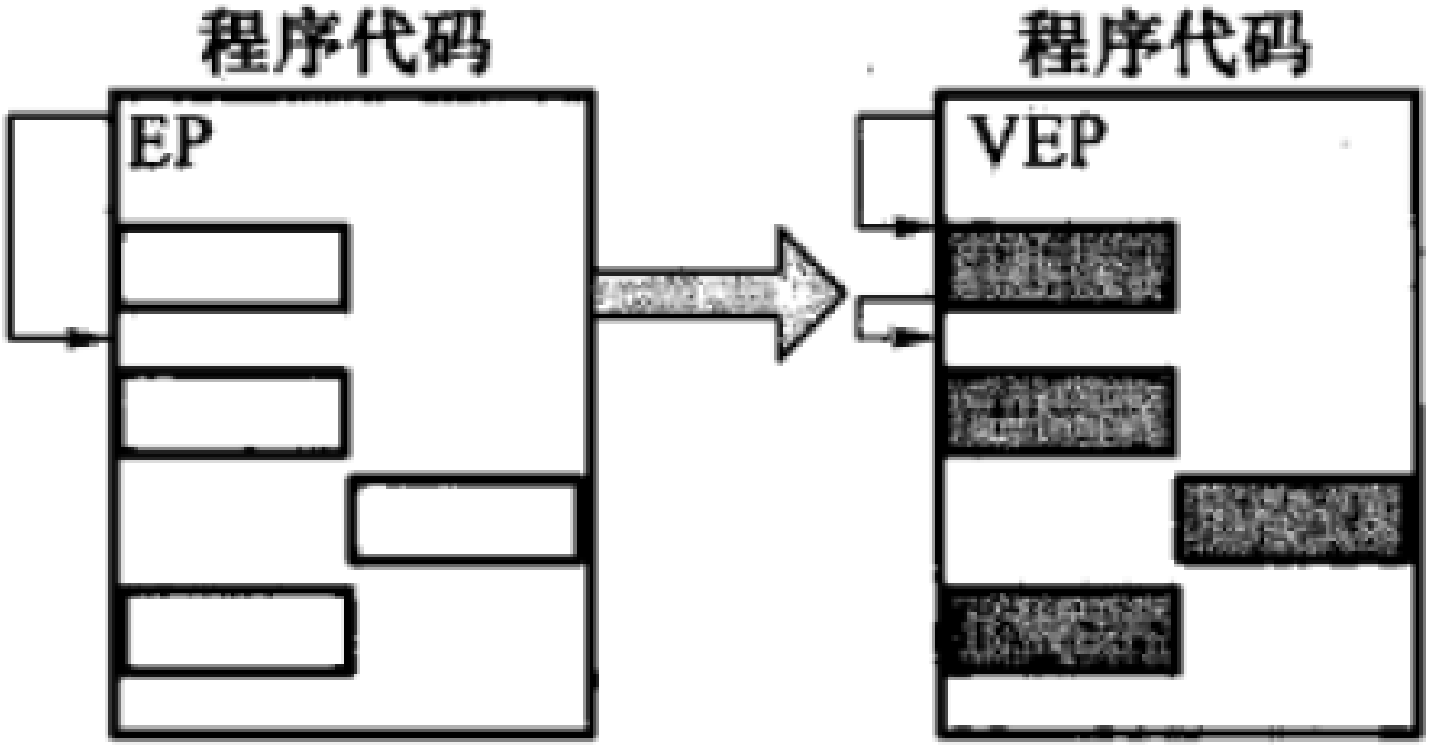
蛀穴病毒（cavity virus）

- 通常不增加被感染对象的大小，而是重写宿主文件中可用来安全存放病毒代码的区域，如重写二进制宿主文件中的零值区域， 或包含空格的区域
- 蛀穴病毒将自身代码注入到宿主文件的一个洞穴中



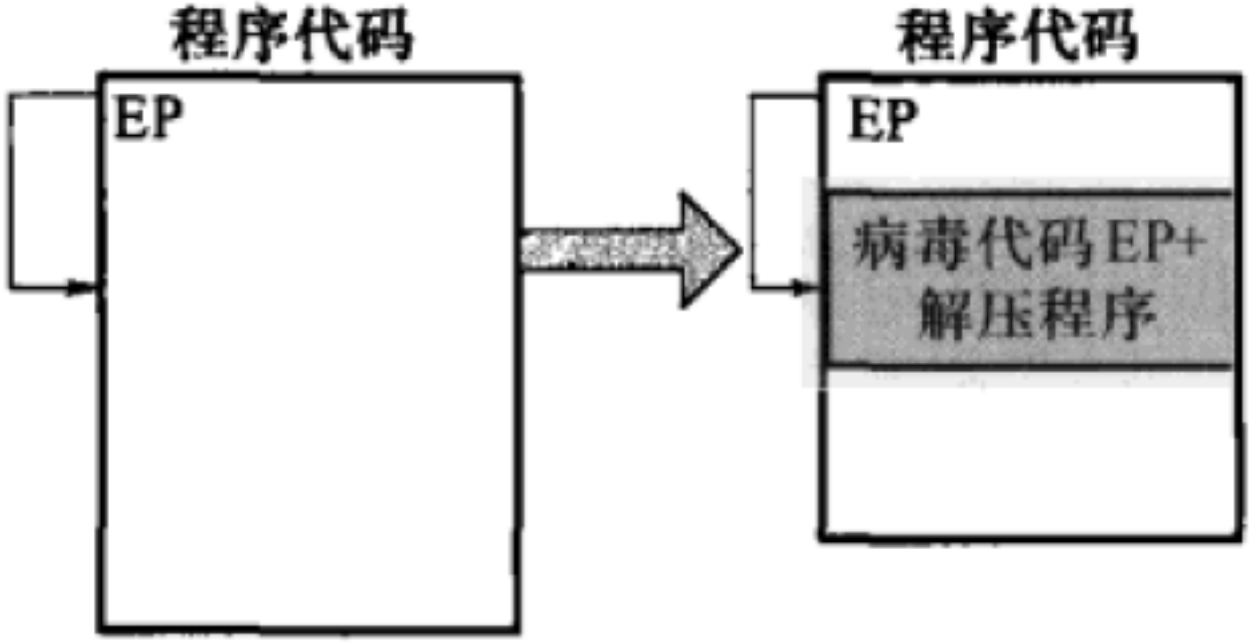
分割型蛀穴病毒

- 分割型蛀穴病毒：蛀穴病毒的一个变形，即病毒代码被分割成一个装入例程和N个片断，这些片断位于包含闲散空间的节中



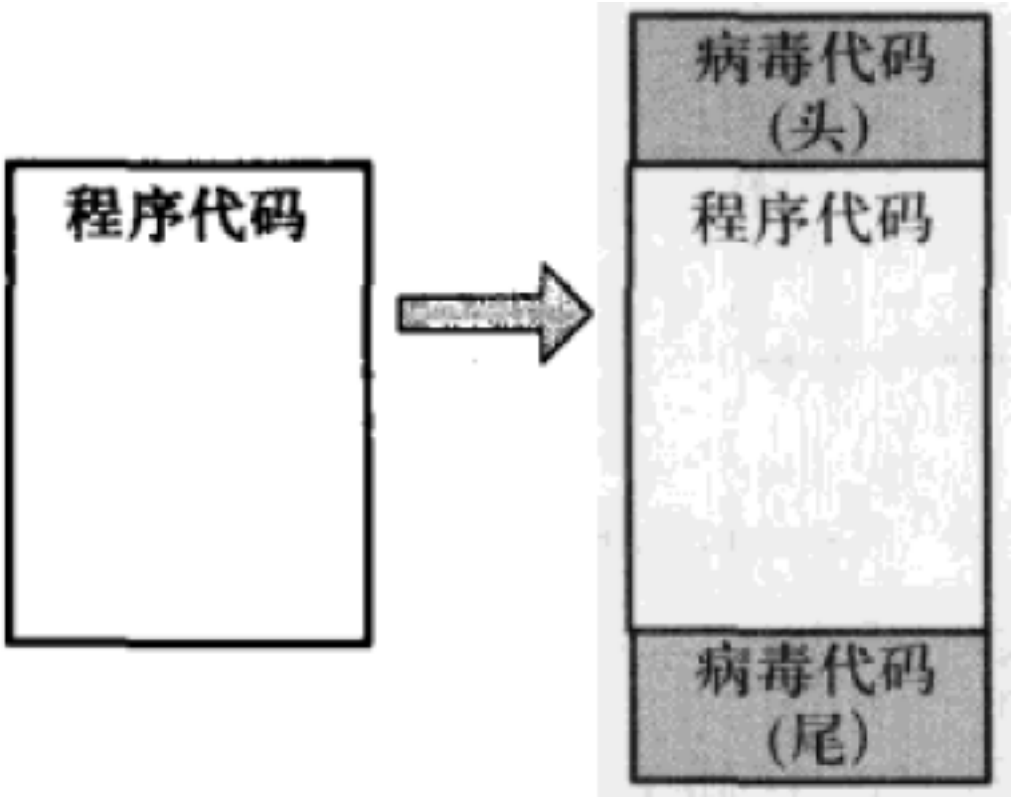
压缩型病毒

- 有时用来隐瞒宿主程序长度的增长：采用一个二进制压缩算法，对宿主程序进行充分压缩，从而节省空间
- 二进制压缩工具，如PKLITE，是非常流行的程序。很多也被攻击者用来压缩木马、病毒或蠕虫，以增加迷惑性，同时减少长度



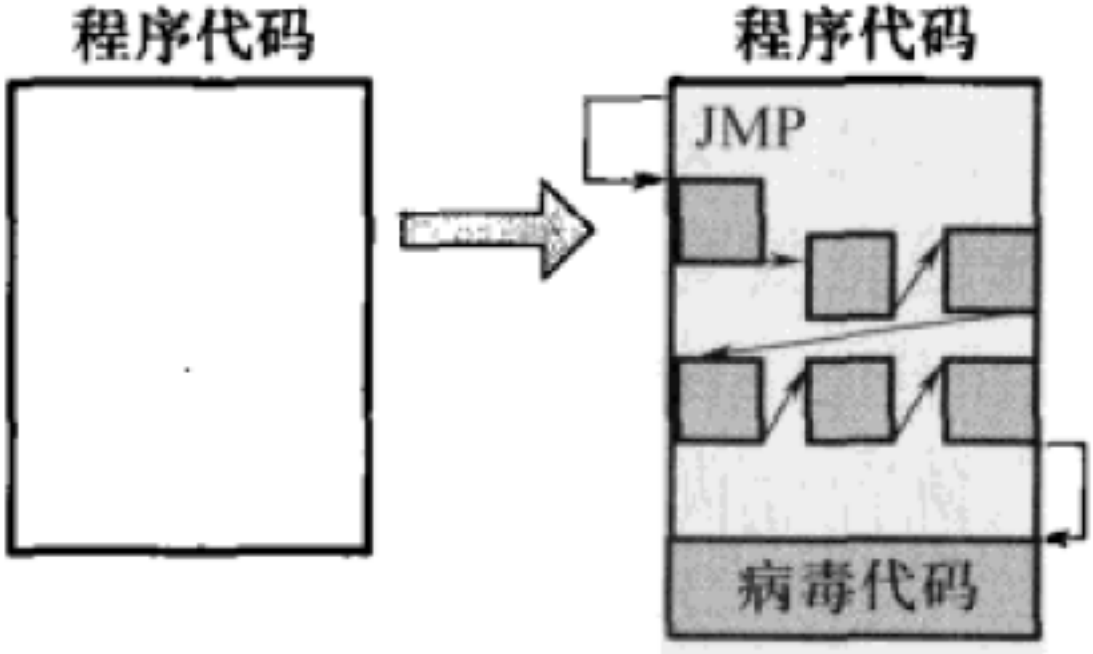
变形虫（amoeba）感染技术

- 这种技术较罕见，它把宿主程序嵌入到病毒体中，即把病毒头部放到文件之前，病毒尾部追加到宿主之后



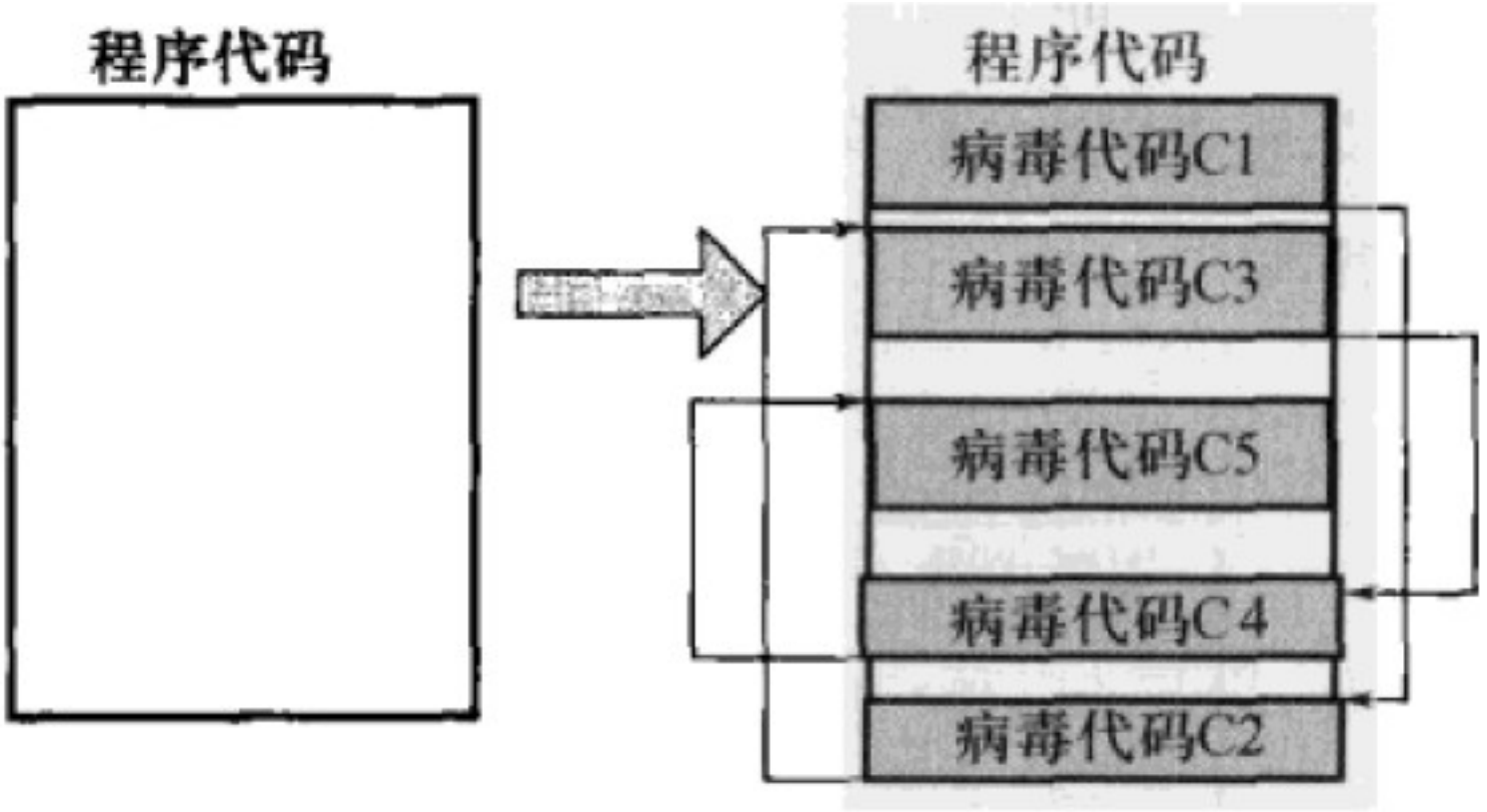
嵌入式解密程序技术

- 一些高级的病毒会把其解密程序注入可以执行的宿主文件中，并将宿主入口点修改为指向解密程序代码
- 解密程序的注入位置是随机选择的，解密程序被分割成多个部分。病毒会把被重写的区域存储在病毒代码中，以便感染之后宿主程序可以正确执行



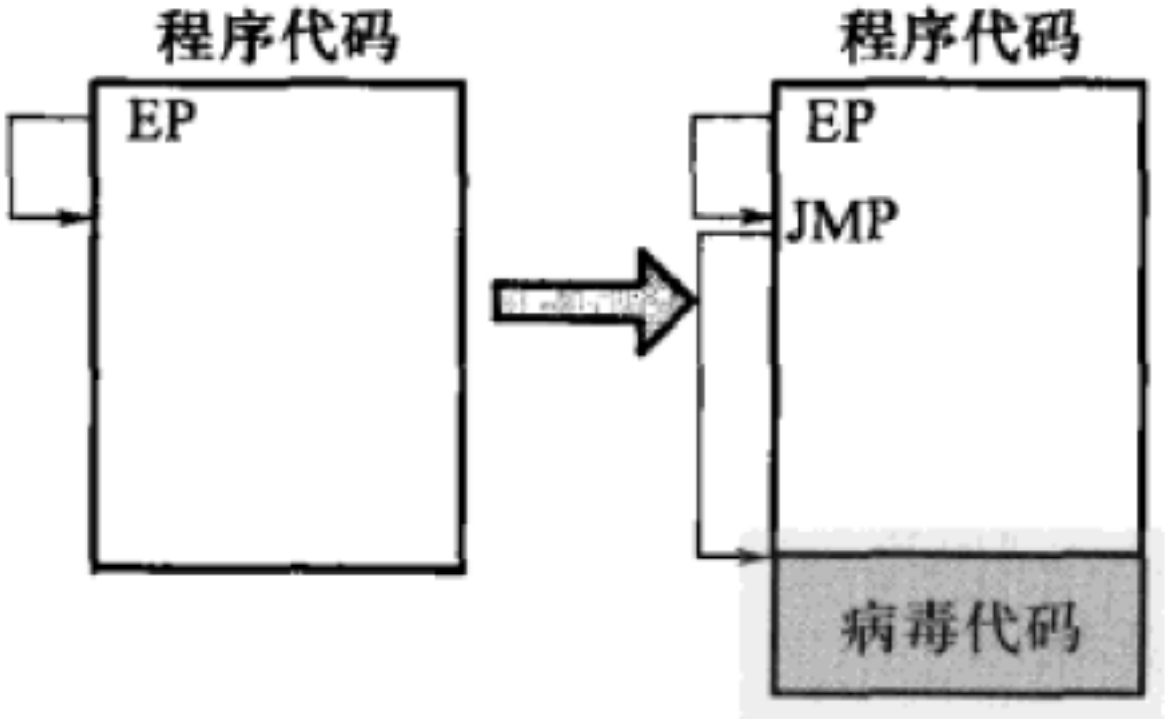
嵌入式解密程序和病毒体技术

- 病毒体被分成几个部分，将它们注入到宿主程序中的随机位置，重写那些位置原来的内容
- 病毒代码的头部位于宿主文件的最开始，它执行时将控制权交给病毒代码的第二个片段，如此一直到最后一个片段



迷惑性欺骗跳转技术

- 迷惑性欺骗跳转技术是一种避免修改宿主文件原始入口点的常见技术。该技术可以对抗启发式检测

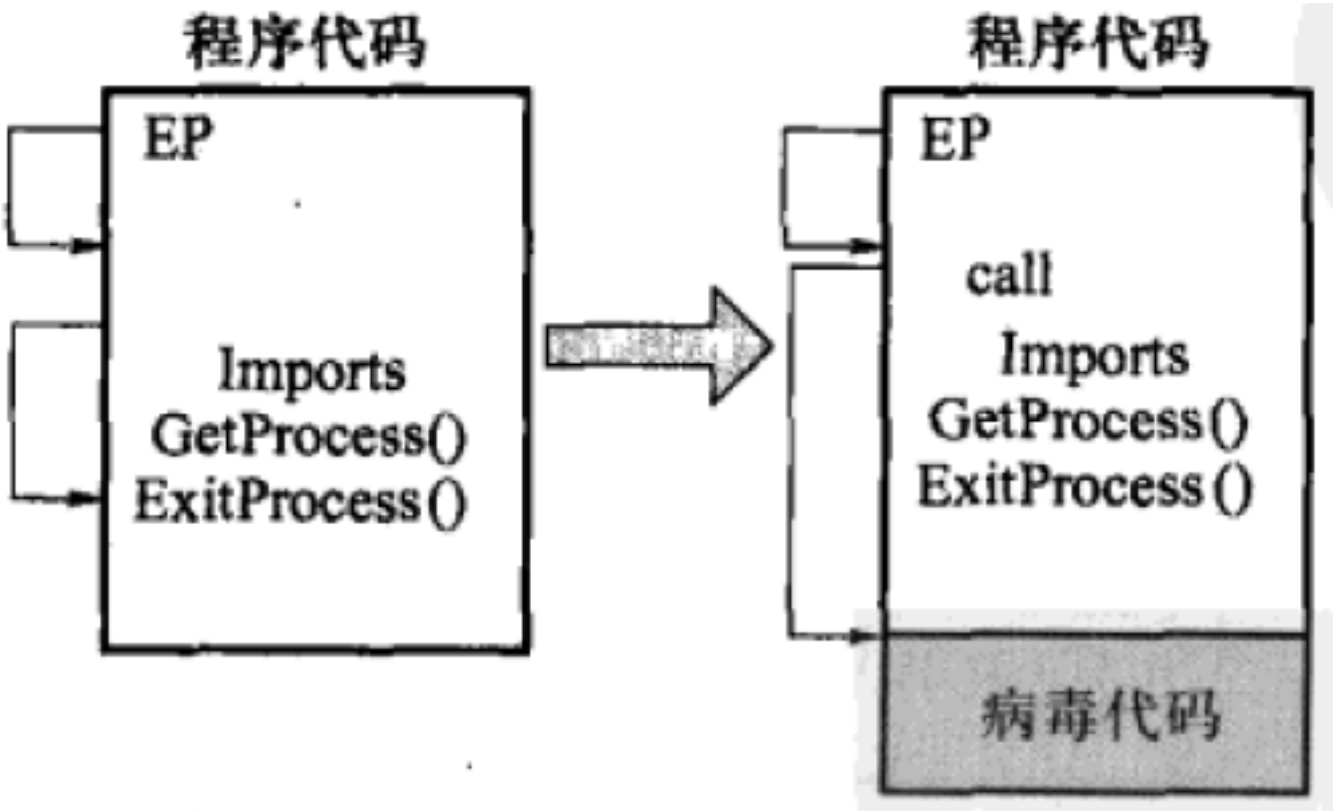


迷惑性欺骗跳转技术

W32/Donut 是最早感染 . NET 可执行文件的病毒，它并不依赖及时(JIT)编译技术。当执行已被感染的 . NET PE 文件时，Donut 病毒立即获得了控制权。该病毒使用最简单可行的技术来感染 . NET 文件。它把位于 . NET 文件入口点的 6 字节长的指向_CorExeMain()导入表的跳转指令替换为一个指向病毒入口点的跳转指令。_CorExeMain()函数是用来启动微软中间语言(MSIL)代码的公共语言运行库(CLR)的运行。头部的入口点不会被病毒改变。这个技术称为迷惑性欺骗跳转。入口点的实际跳转会被替换为一个0Xe9(JMP)操作码，后面跟着一个偏移地址，指向位于重定位节第一个物理字节的病毒体。

入口点隐蔽病毒

- 入口点隐蔽病毒不会改变宿主程序的入口点位置，也不会改变入口点的代码，而是通过改变宿主代码的某个位置而令病毒随机地获得控制权



(1) Win32 上的 API 挂钩技术。以 Win32 系统为例,最常见的 EPO 技术是基于对程序代码节中某个指令模式进行挂钩。EPO 病毒对 API 调用的位置进行修改,将其指针改为指向病毒自身代码的起点。

例如 W32/CTX 病毒就是在宿主程序的代码节中寻找指向导入表目录的 CALL 指令。这样病毒就可以识别出属于某个函数调用的字节模式。然后,病毒会修改 CALL 指令,使其指向位于其他地方的病毒代码。此类病毒通常会寻找如下的 API 微软的 API 调用实现。

CALL DWORD PTR[ ]

此类病毒还可能完全随机地选择一个 API 挂钩位置;这样可能会出现不是每次宿主执行时病毒代码都可以获得控制权的情况。

病毒可以挂钩到应用程序每次退出返回时都会调用的 API 上。例如,大多数程序都会调用 ExitProcess( )API。如果把对 ExitProcess( )的调用替换为对病毒体的调用,则在程序退出时就可以更可靠地触发感染例程。

图 4.16 显示一个 EPO 病毒用指向病毒代码的 CALL 指令替换了指向 ExitProcess( )的 CALL 指令。当病毒取得控制权后,它修复内存中的宿主代码,然后把控制权传递给修复后的代码区块,即最终还是要执行原始的宿主代码。

(2) Win32 上的函数调用挂钩。EPO 病毒常用的另一种技术是在应用程序代码节中可靠地找出一个指向子程序的函数调用。由于代表 CALL 指令的字节模式可能正好是另一条指令的数据部分,因此如果仅寻找 CALL 指令,可能病毒不能正确识别指令的边界。

为解决这个问题,病毒常常检查是否 CALL 指令指向的字节模式看起来类似于典型的子程序调用的开始位置,例如:

```
CALL FOOBAR
FOOBAR:
PUSH EBP                    ; opcode ox55
MOV EBP,ESP                 ; opcode 0x89e5
```

图 4.17 显示了指向病毒起点的 CALL 指令替换指向 FOOBAR( )的 Call 指令。Foobar( )函数以 0x55 0x89 0xe5 序列开始,病毒很容易识别出这是一个函数的入口点。还有一个类似的操作码(opcode)序列 0x55 0x8B 0xEC,也对应于同样的汇编代码。W32/RainSong 病毒的变种使用了这种感染技术。

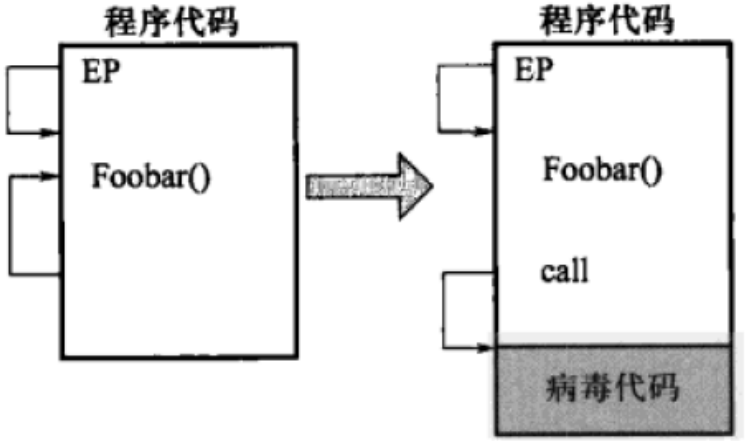
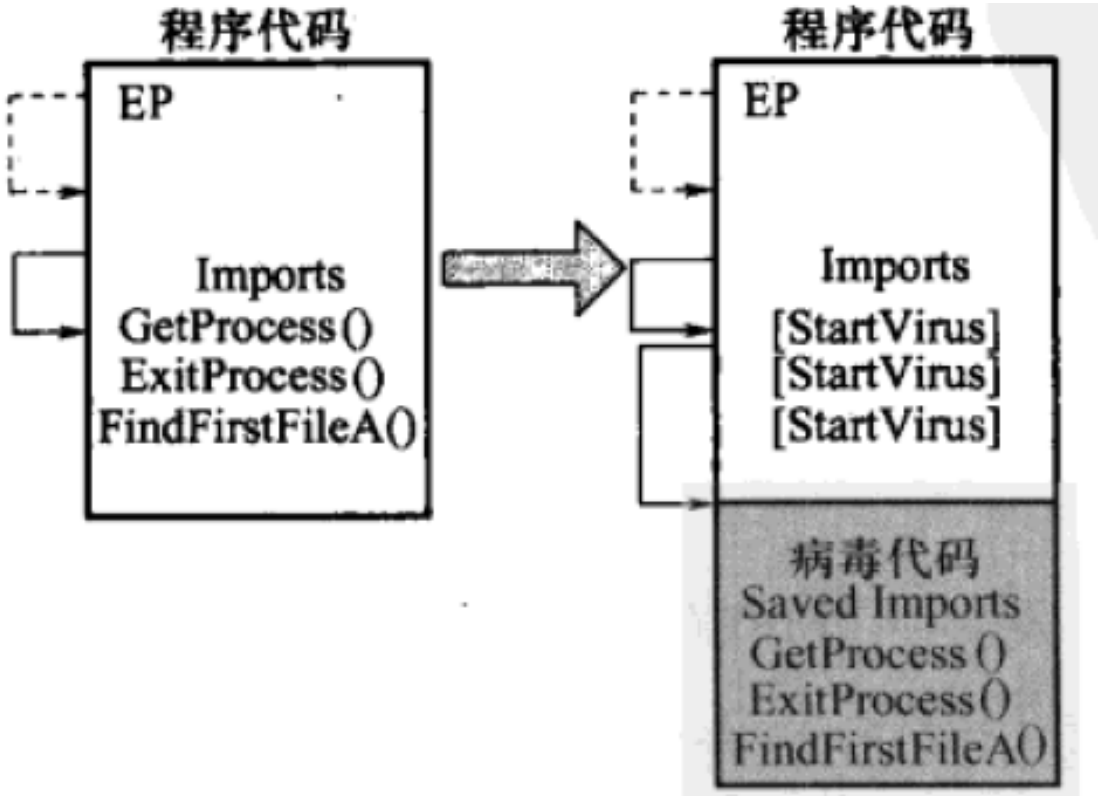


图 4.17 函数调用挂钩型 EPO 病毒

(3) Win32 上的导入表替换(import table replacing)。较新的 Win32 病毒感染 Win32 PE 文件时,不需要修改宿主程序的原始代码来获得控制权,只需修改 PE 宿主文件的导入地址表(import address table)条目,把此应用程序通过导入地址目录所做的每个 API 调用都替换为执行病毒代码。而激活的病毒代码则在该程序内存映像中提供了一个新的导入表。结果 API 调用就通过修复后的导入表(即新的导入表)执行原来真正的入口点代码。

W32/Idele 病毒使用了该技术,如图 4.18 所示,W32/Idele 用一段短小的例程修改了程序代码节的闲散空间,这段例程用于分配内存,以存储解密后的病毒代码,然后执行该代码。Idele 病毒避免了创建所包含地址病毒不指向代码节的导入表条目。



病毒的检测

- 比较法：比较法是用原始的或者正常的内容与被检测的进行比较。比较法包括长度比较法、内容比较法、内存比较法、中断比较法等
- 校验和法：计算正常文件的内容的校验和，并将该校验和写入文件中保存。可识别未知病毒
- 扫描法：扫描法是根据每一种病毒含有的特征字符串，对被检测的对象进行扫描。如果在被检测对象内部发现了某一种特定字符串，表明发现了该字符串所代表的病毒。扫描法包括特征代码扫描法和特征字扫描。扫描程序由病毒特征库和扫描程序组成

病毒的检测

- 行为监测法：利用病毒的特有行为特性监测病毒
- 感染实验法：简单实用，可以检测出病毒检测工具不认识的新病毒，摆脱对病毒检测工具的依赖，自主的检测可疑的新病毒
- 软件模拟法：多态型病毒每次感染都改变其病毒密码，对付这种病毒时特征代码法会失效。用软件方法来模拟和分析程序的运行：模拟CPU运行，在其设计的虚拟机下假执行病毒的变体引擎解码程序，安全地将多态病毒解开，使其显露出真实面目
- 蜜罐、FireEye

病毒的检测

- 分析法：搞清楚病毒体的大致结构，提取特征识别用的字符串或者特征字，用于增添到病毒代码库供病毒扫描和识别程序使用
  - 反病毒技术人员：专业病毒分析师
  - 静态分析：反汇编器。查看病毒分为哪些模块，使用了哪些系统调用，采用了哪些技巧，如何将病毒感染过程变成清除病毒、修复文件的过程，哪些代码可以作为特征码
  - 动态分析：调试。对病毒进行动态跟踪，观察病毒的具体工作过程，以进一步理解病毒工作机理

病毒的检测

- 启发式代码扫描技术：主要源于人工智能技术，是基于给定的判断规则和定义的扫描技术，若发现被扫描程序中存在可疑的程序功能指令，则作出存在病毒的预警。它是对传统的特征代码扫描法的改进
- 虚拟机查毒技术：动态特征码扫描技术的关键，是分析多态病毒的关键技术。通过虚拟执行，对付加密、变形、异型、加壳、压缩、病毒生成机及大部分未知病毒及破坏性病毒
- 实时内存扫描和内存监控技术：对当前系统中运行的进程和线程的内存进行扫描，通过提取病毒运行时的内存特征进行匹配来清除已经运行的病毒，从而实现带毒杀毒

大纲

- 恶意软件的分类和区别
- 病毒的机理与防治
- ✓ 蠕虫的机理与防治
- 木马的机理与防治
- 其他恶意代码的机理
- 恶意代码分析技术

蠕虫的起源

- 1980年，Xerox PARC的研究人员John Shoch和Jon Hupp在研究分布式计算、监测网络上的其他计算机是否活跃时，编写了一种特殊程序，Xerox蠕虫
- 1988年11月2日，世界上第一个破坏性计算机蠕虫正式诞生
  - Morris为了求证计算机程序能否在不同的计算机之间进行自我复制传播，编写了一段试验程序
  - 为了让程序能顺利进入另一台计算机，他还写了一段破解用户口令的代码
  - 11月2日早上5点，这段被称为“Worm”(蠕虫)的程序开始了它的旅行。它果然没有辜负Morris的期望，爬进了几千台计算机，让它们死机
  - Morris蠕虫利用sendmail的漏洞、fingerD的缓冲区溢出及REXE的漏洞进行传播
- Morris在证明其结论的同时，也开启了蠕虫新纪元

蠕虫和病毒的区别及联系

- 蠕虫的最大特点是利用各种安全漏洞进行自动传播
- 蠕虫和病毒都具有传染性和复制功能，但是两者还是有区别
- 认识这种区别有利于采取有针对性的措施进行防治

蠕虫和病毒的区别

	病　毒	蠕　虫
存在形式	寄生	独立个体
复制机制	插入到宿主程序(文件)中	自身的拷贝
传染机制	宿主程序运行	系统存在漏洞(Vulnerability)
搜索机制(传染目标)	主要是针对本地文件	主要针对网络上的其它计算机
触发传染	计算机使用者	程序自身
影响重点	文件系统	网络性能、系统性能
计算机使用者角色	病毒传播中的关键环节	无关
防治措施	从宿主程序中摘除	为系统打补丁(Patch)
对抗主体	计算机使用者，反病毒厂商	系统提供商，网络管理员

蠕虫的分类

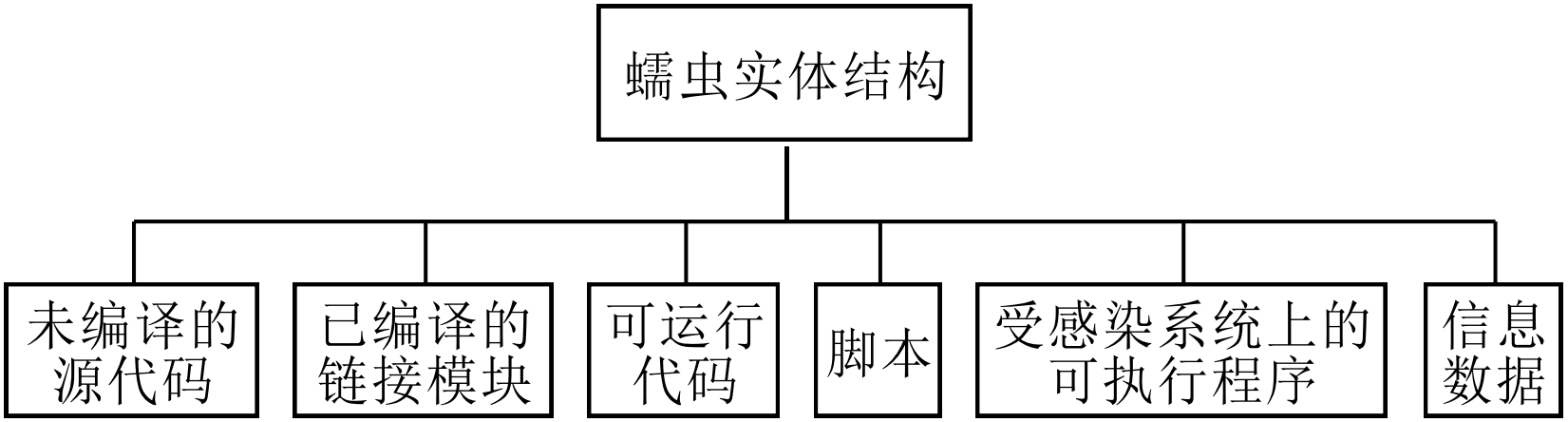
- 根据蠕虫的传播、运作方式，可将蠕虫分为
  - 主机蠕虫：主机蠕虫的所有部分均包含在其所运行的计算机中，利用网络连接仅仅是为了将其自身拷贝到其它计算机中。任意时刻只有一个蠕虫的复制在运行，又称为“兔子”
  - 网络蠕虫：网络蠕虫有许多部分（称为段，segment）组成，而且每一个部分运行在不同的计算机中，并且使用网络的目的，是为了进行各个部分之间的通信以及传播。具有一个主段，用于协调其他段的运行，又称为“章鱼”

蠕虫与软件漏洞的关系

- 根据邮件利用漏洞的不同，可以分为
  - 邮件蠕虫：邮件蠕虫主要利用漏洞。蠕虫病毒的主要来源
  - 网页蠕虫：网页蠕虫主要利用IFrame漏洞和MIME漏洞
  - 系统漏洞蠕虫：系统漏洞蠕虫一般具备一个小型的漏洞利用系统，它随机产生IP地址并尝试漏洞利用，然后将自身复制过去。杀伤力最大的蠕虫，例如利用RPC溢出漏洞的冲击波、利用LSASS漏洞的震荡波等

蠕虫的基本结构

- 蠕虫的实体结构
  - 未编译的源代码
  - 已编译的链接模块
  - 可运行代码
  - 脚本
  - 受感染系统上的可执行程序
  - 信息数据
- 具体的某个蠕虫可能仅包含其中几个部分



蠕虫的实体结构

（1）未编译的源代码：由于某些程序参数必须在编译时确定，所以蠕虫程序可能包含一部分未编译的程序源代码。

（2）已编译的链接模块：不同的系统，可能需要不同的运行模块，例如，不同的硬件厂商和不同的系统厂商可能采用不同的运行库。

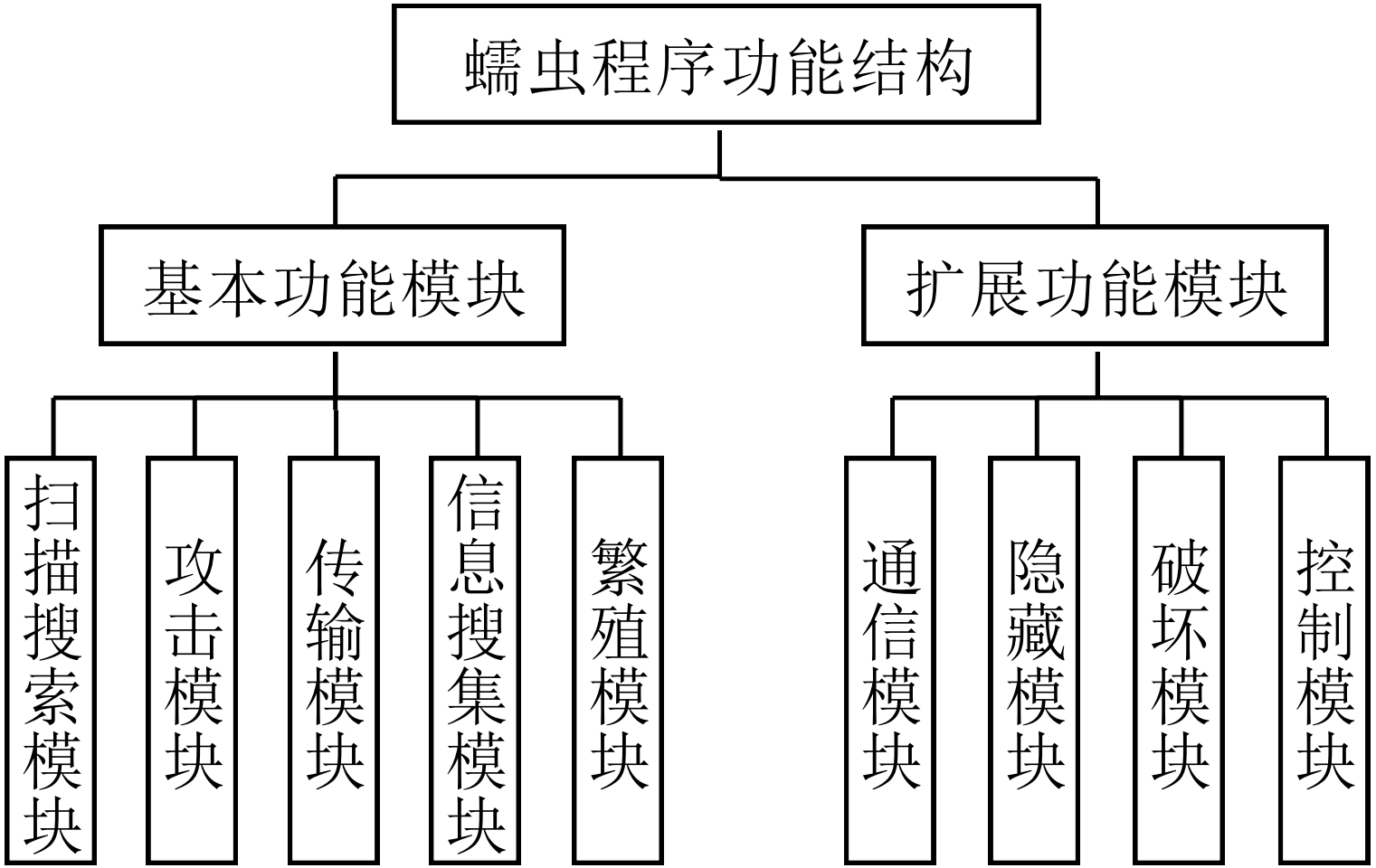
（3）可运行代码：整个蠕虫可能由多个编译好的程序组成。

（4）脚本：利用脚本可以节省大量的程序代码，充分利用系统 shell 的功能。

（5）受感染系统上的可执行程序：受感染系统上的可执行程序，如文件传输等，可以被蠕虫作为自己的组成部分。

（6）信息数据：包括已经破解的口令、要攻击的地址列表、蠕虫自身的压缩包等。

蠕虫的功能结构


蠕虫程序功能结构
基本功能模块
扩展功能模块
扫描搜索模块
攻击模块
传输模块
信息搜集模块
繁殖模块
通信模块
隐藏模块
破坏模块
控制模块

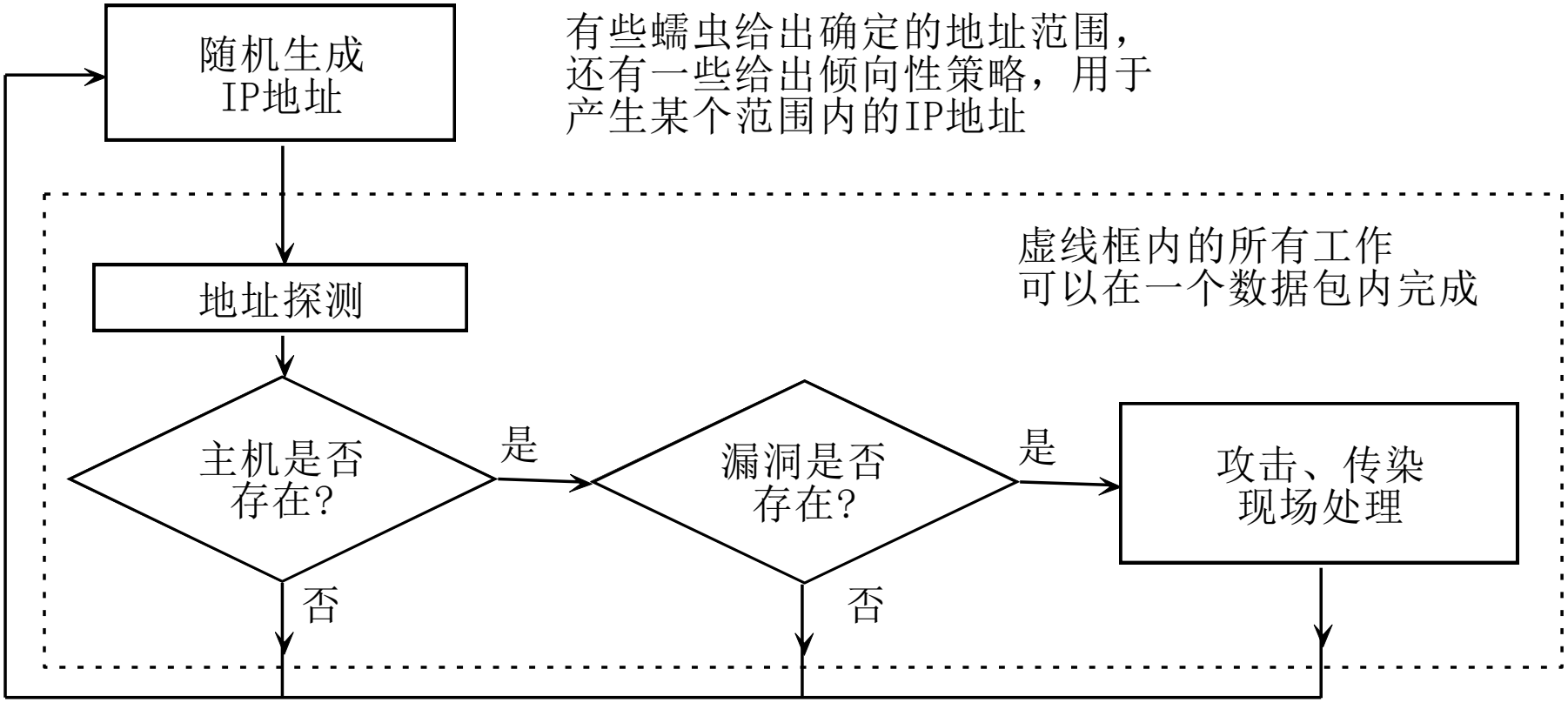
蠕虫的功能结构

- 基本功能：完成复制传播流程
  - 扫描搜索模块：寻找下一台要传染的计算机；为提高搜索效率，可以采用搜索算法
  - 攻击模块：在被感染的计算机上建立传输通道；为减少传染数据传输量，可以采用引导式结构
  - 传输模块：计算机之间的蠕虫程序复制
  - 信息收集模块：搜寻和建立被传染计算机的信息
  - 繁殖模块：建立自身的多个副本；在同一台计算机上提高传输效率、避免重复传输

蠕虫的功能结构

- 扩展功能：更强的生存能力和破坏力
  - 隐藏模块：隐藏蠕虫程序，使简单的检测不能发现蠕虫
  - 破坏模块：摧毁或破坏被感染计算机，或在被感染计算机上留下后面程序等
  - 通信模块：蠕虫之间、蠕虫同黑客之间进行交流，这可能是未来蠕虫发展的侧重点。Botnet
  - 控制模块：调整蠕虫行为、更新其它功能模块、控制被感染计算机

蠕虫的工作方式


随机生成
IP地址
地址探测
主机是否
存在?
是
漏洞是否
存在?
是
攻击、传染
现场处理
否
否
有些蠕虫给出确定的地址范围，
还有一些给出倾向性策略，用于
产生某个范围内的IP地址
虚线框内的所有工作
可以在一个数据包内完成

蠕虫的工作方式

- 扫描：由蠕虫的搜索扫描功能模块复杂探测存在漏洞的主机。当程序向某个主机发送探测漏洞的信息并收到成功的反馈信息后，就得到一个可攻击的对象
- 攻击：攻击模块按漏洞攻击步骤自动攻击上一步骤中找到的对象，取得该主机的权限，获得一个shell
- 复制：繁殖模块通过原主机和新主机之间的交互，将蠕虫程序复制到新主机并启动

蠕虫的扫描策略

- 蠕虫的扫描策略
  - 现在流行的蠕虫采用的传播技术目标，一般是尽快地传播到尽量多的计算机中
  - 扫描模块采用的扫描策略是：随机选取某一段IP地址，然后对这一地址段上的主机进行扫描
  - 没有优化的扫描程序可能会不断重复上面这一过程，大量蠕虫程序的扫描引起严重的网络拥塞
- 对扫描策略的改进
  - 在IP地址段的选择上，可以主要针对当前主机所在的网段进行扫描，对外网段则随机选择几个小的IP地址段进行扫描
  - 对扫描次数进行限制，只进行几次扫描
  - 把扫描分散在不同的时间段进行

蠕虫的扫描策略

- 扫描策略设计的原则
  - 尽量减少重复的扫描，使扫描发送的数据包总量减少到最小
  - 保证扫描覆盖到尽量大的范围
  - 处理好扫描的时间分布，使得扫描不要集中在某一时间内发生
  - 怎样找到一个合适的策略需要在考虑以上原则的前提下进行分析，甚至需要试验验证

蠕虫的扫描策略

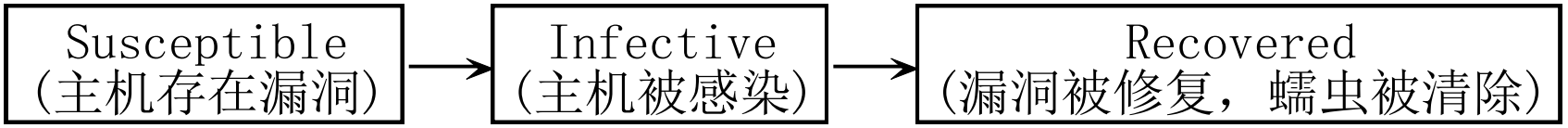
- 蠕虫常用的扫描策略
  - 选择性随机扫描(包括本地优先扫描)
  - 可路由地址扫描(Routable Scan)
  - 地址分组扫描(Divide-Conquer Scan)
  - 组合扫描(Hybrid Scan)
  - 极端扫描(Extreme Scan)

蠕虫的传播模型

- 蠕虫在计算机网络上的传播，可看作是服从某种规律的网络传播行为
- 一个精确的蠕虫传播模型可以使人们对蠕虫有更清楚的认识，能确定其在传播过程中的弱点，而且能更精确的预测蠕虫所造成的损失
- 目前已有很多学者对蠕虫进行了深入研究，提出了一些模型，这些蠕虫传播模型都是针对随机网络的，不能很好的模拟实际网络环境。如何精确地描述蠕虫传播行为，揭示它的特性，找出对该行为进行有效控制的方法，一直是学者们共同关注的焦点
- 最经典网络传播模型的SIS模型和SIR模型
  - 在SIS模型中，每一个网络节点只能处于两种状态中的一种，一是易感的，二是已被感染从而具有传染能力的
  - 在SIR模型中，节点还可以处于一种叫做免疫的状态，在这种状态下，节点既不会被感染，也不会感染其它节点

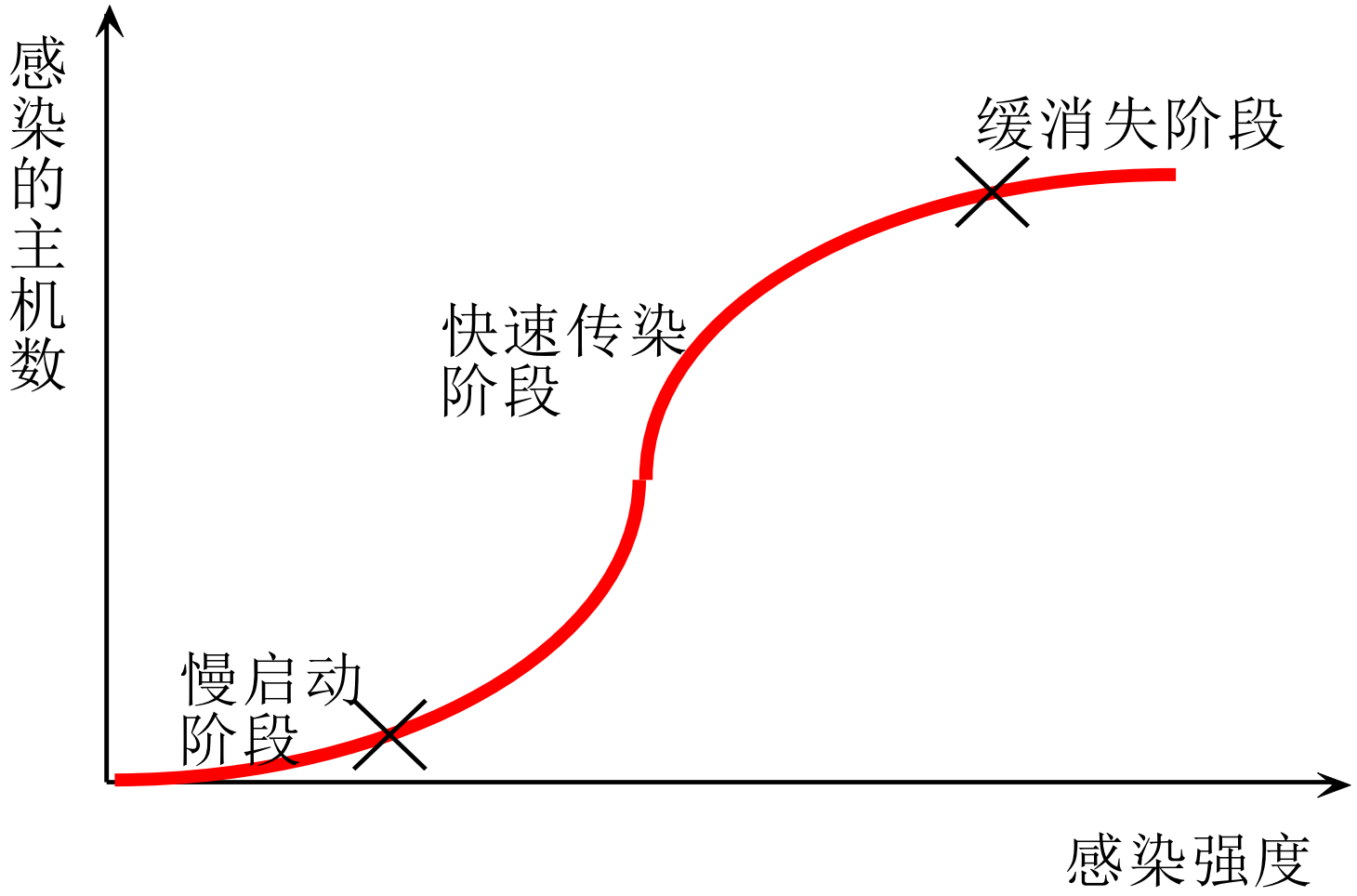
蠕虫的传播模型

- Kermack-Mckendrick模型是SIR模型中的经典
  - 在Kermack-Mckendrick模型中考虑了感染主机的恢复，它假定在蠕虫传播期间，一些受感染的主机可以恢复为正常状态或死亡，且对此蠕虫具有免疫功能，因此每个主机具有三种状态：易感、感染或恢复，其状态转移可表示为易感→感染→恢复，或永远保持易感状态



蠕虫的传播模型

- 感染的主机数与感染强度示意图



蠕虫的传播模型

- Kermack-Mckendrick模型用数学公式可以表示为
  - I(t)表示在时刻t被感染的主机数
  - R(t)表示在时刻t被恢复的主机数
  - S(t)表示在t时刻尚未感染的主机数
  - N表示主机总数
  - β - 感染系数　　γ - 恢复系数

$$\begin{cases} \dfrac{dI(t)}{dt} = \beta I(t)S(t) - \gamma I(t) \\ \dfrac{dR(t)}{dt} = \gamma I(t) \\ N = I(t) + R(t) + S(t) \end{cases}$$

- 从Kermack-Mckendrick模型可以得出一个重要的理论——蠕虫爆发定理：一个大规模蠕虫爆发的充分必要条件是初始易感主机的数目S(0)>ρ
- 由于存在蠕虫爆发的阈值，因此，采取各种防治手段，如安装杀毒软件、打补丁、断开网络连接等降低蠕虫感染率，通过先进的治疗手段提高恢复率，使S(0)≤ρ，从而有效地遏制蠕虫的传播。能否在蠕虫的缓慢传播阶段实现对蠕虫的检测和防治成为有效防治蠕虫的关键

蠕虫的行为特征

- 通过对蠕虫的整个工作流程进行分析，可以归纳得到它的行为特征
  - 主动攻击
  - 行踪隐蔽
  - 利用系统、网络应用服务器漏洞
  - 造成网络拥塞
  - 消耗系统资源，降低系统性能
  - 产生安全隐患
  - 反复性
  - 破坏性

蠕虫技术的发展

- 超级蠕虫
  - 超级蠕虫包含多操作系统的运行版本，包含丰富的漏洞库，从而具有更强的传染能力
  - 超级蠕虫的一次攻击就能针对多个漏洞，只要有一个漏洞没有打补丁，即可攻击成功
- 分布式蠕虫
  - 数据部分同运行代码分布在不同的计算机中，运行代码在攻击时，从数据存放地获取供给信息
  - 同时，攻击代码用一定的算法在多台计算机上寻找、复制数据的存放地
  - 不同功能模块分布在不同的计算机之间协调工作，产生更强的隐蔽性和攻击能力

蠕虫技术的发展

- 通信技术
  - 蠕虫具有蠕虫之间、编写者与蠕虫之间传递信息和指令的能力
- 与网络攻防技术的结合
  - 网络攻防技术将会被纳入到蠕虫的功能中来隐藏蠕虫的踪迹
    - 通过动态地改变攻击代码，逃避IDS的特征检测
    - 在一些常用的通信协议端口上建立隐蔽通道
    - 通过系统内核级后面控制一个系统
- 与病毒技术的结合
  - 攻击计算机系统后，继续攻击文件系统

蠕虫的防治与检测

- 蠕虫防治的方案
  - 从实体结构来考虑，如果破坏了它的实体组成的一个部分，则破坏了其完整性，使其不能正常工作，从而达到阻止其传播的目的
  - 从功能组成来考虑，如果使其某个功能组成部分不能正常工作，也同样能达到阻止其传播的目的
  - 具体可以分为如下一些措施：
    - 修补系统漏洞
    - 分析蠕虫行为
    - 重命名或者删除命令解释器
    - 防火墙
    - 公告

蠕虫的防治周期

- 预防阶段
  - 在利用某个漏洞进行攻击的蠕虫产生之间，积极主动地升级系统、安装防火墙、安装入侵检测系统等，防患于未然
- 检测阶段
  - 密切注意网络流量异常、TCP连接异常等异常现象，尽量在蠕虫的缓慢启动期发现蠕虫
- 遏制阶段
  - 在蠕虫的快速传播期，通过各种手段遏制蠕虫的快速传播
- 清除阶段
  - 清除已感染主机中的蠕虫；通过打补丁等手段，杜绝易感染主机的存在，最终清除蠕虫

蠕虫的防治与检测

- 对未知蠕虫的检测
  - 比较通用的方式是对流量异常的统计分析、对TCP连接异常的分析、对ICMP数据异常的分析等
  - 以ICMP流量异常的分析为例，在蠕虫的扫描阶段，会随机生成大量的IP地址进行扫描，探测漏洞主机。这些被扫描的IP中，存在许多空的或不可达的IP地址，从而在一段时间内，蠕虫主机会接收到大量的来自不同路由器的ICMP不可达数据包，通过对这些数据包进行检测和统计，即可在蠕虫的扫描阶段将其发现，然后将蠕虫主机进行隔离、分析

大纲

- 恶意软件的分类和区别
- 病毒的机理与防治
- 蠕虫的机理与防治
- ✓ 木马的机理与防治
- 其他恶意代码的机理
- 恶意代码分析技术

木马的定义

- 木马的名称源于古希腊神话特洛伊木马
- 木马是一种恶意程序，是一种基于远程控制的攻击工具，它一旦入侵用户的计算机，就悄悄地在宿主计算机上运行，在用户毫无觉察的情况下，让攻击者获得远程访问和控制系统的权限，进而在用户的计算机中修改文件、修改注册表、控制鼠标、监视/控制键盘，或窃取用户信息，乃至实施远程控制
- 它是攻击者的主要攻击手段之一，具有隐蔽性和非授权性等特点

木马的结构

- 木马系统通常采用服务器，客户端结构，即分为服务端和客户端。通常功能上由木马配置程序、控制程序和木马程序三个部分组成
  - 木马程序：木马程序也称为服务器程序，驻留在受害者的系统中，非法获取其操作权限，负责接收控制指令，并根据指令或配置发送数据给控制端
  - 木马配置程序：木马配置程序设置木马程序的端口号、触发条件、木马名称等，使其在服务器端隐藏得更隐蔽
  - 控制程序：控制程序控制远程木马服务器，统称为控制端程序，负责配置服务器、给服务器发送指令

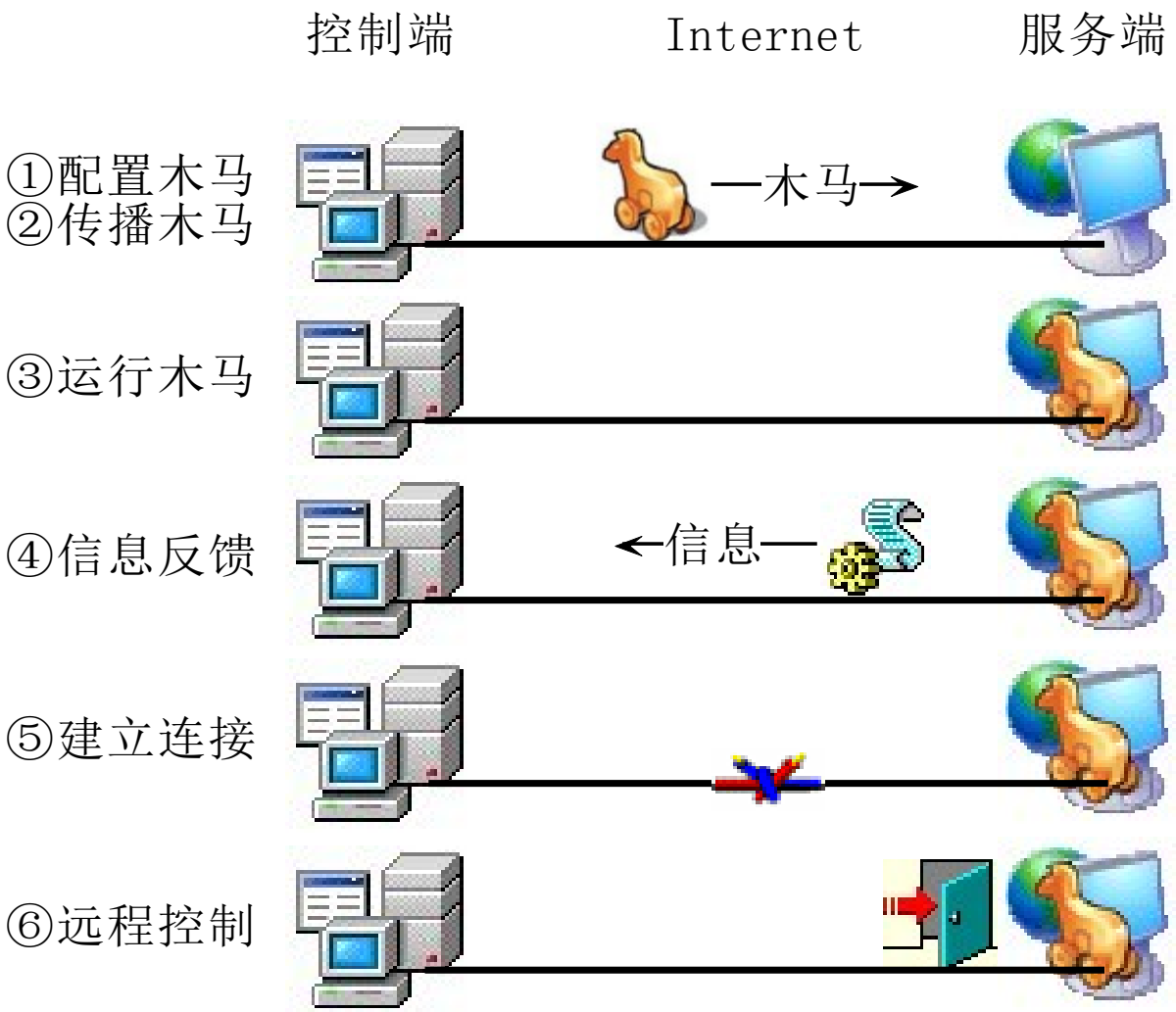
木马的结构

木马的基本原理

- 多数木马包括客户端和服务器端两个部分，即采用服务器/客户端结构
- 攻击者通常利用“绑定程序”将木马服务器绑定到某个合法软件上。安装即中招
- 攻击者利用客户端控制服务器，之间会建立一个通信连接
- 控制端可以进行远程控制。如果攻击者控制了大量计算机（僵尸、肉鸡或僵尸网络等），则可以发起DDoS攻击

木马实施网络入侵的基本步骤

- 配置木马
- 传播木马
- 运行木马
- 信息反馈
- 建立连接
- 远程控制



木马实施网络入侵的基本步骤

- 配置木马
  - 木马伪装：让木马在服务器尽可能隐藏得更加隐蔽
  - 信息反馈：设置信息反馈的方式或地址，如设置邮件地址、QQ号等
  - 在释放木马之前可以配置木马，释放木马之后也可以通过远程配置木马
- 传播木马
  - 采用各种传播方式，将配置好的木马传播出去

木马实施网络入侵的基本步骤

运行木马

服务端用户运行木马或捆绑木马的程序后，木马就会自动进行安装，木马首先将自身复制到 Windows 的系统文件夹中（C:\Windows，C:\Windows\system，或 C:\Windows\temp），然后在注册表、启动组、非启动组等位置设置木马的触发启动条件，完成木马服务器的安装。安装后就可以启动木马了。木马被激活后，进入内存，开启并监听预先定义的木马端口，准备与控制端建立连接，如图 4.24 所示。

图 4.24　木马运行的基本过程

此时，服务器端用户可以用 netstat 查看端口状态，在脱机状态下一般是不会有端口开放的，如果有端口开放，就要注意是否感染了木马。

木马实施网络入侵的基本步骤

- 信息反馈
  - 木马成功安装后，会手机一些服务端的软硬件信息，并通过E-mail、QQ等手段告知控制端的攻击者
- 建立连接
  - 具备两个条件
    - 服务端已经安装了木马
    - 控制端、服务端都在线
  - 服务端的木马端口和IP地址
    - 木马端口是事先设定的
    - 如何获得服务端的IP地址？

木马实施网络入侵的基本步骤

- 建立连接
  - 获得服务端的IP地址
    - IP扫描
    - 信息反馈
  - 反弹木马：服务端主动与控制端连接，穿透防火墙
- 远程控制
  - 控制端与木马端出现一条通信通道，可以通过该通道以及木马程序对服务器端进行远程控制

木马的传播方式

- 以邮件的附件形式传播
- 通过聊天工具（MSN，QQ等）传播
- 通过软件下载网站传播。有些下载网站提供下载的软件捆绑了木马文件，用户执行下载文件的同时，也运行了木马
- 通过一般的病毒和蠕虫传播
- 通过带木马U盘和光盘传播
- 新途径：网页挂马

木马的危害

- 木马的危害即木马能实现的功能，包括：
  - 窃取数据
  - 接受非授权操作者的指令
  - 远程管理服务端进程
  - 篡改文件和数据
  - 删除文件和数据
  - 操纵注册表
  - 监视服务器的一切动作
  - 释放病毒
  - 使系统自毁

特洛伊木马的特性

- 特洛伊木马具有如下特性：
  - 隐蔽性
  - 非授权性
  - 欺骗性
  - 自动运行性
  - 自动恢复性
  - 主动性
  - 功能的特殊性

特洛伊木马的分类

- 按照对计算机的破坏方式分类
  - 破坏型
  - 密码发送型
  - 远程访问型
  - 键盘记录型
  - DoS攻击型
  - 代理型
  - FTP型
  - 程序杀手型
- 按传输方式分类
  - 主动型
  - 反弹端口型
  - 嵌入式木马

木马的伪装方式

- 木马通过伪装达到降低用户警觉、欺骗用户的目的
  - 修改图标
  - 捆绑文件
  - 出错提示
  - 自我销毁
  - 木马更名

木马的隐藏方式

- 木马的隐藏方式
  - 在任务栏里隐藏
  - 在任务管理器里隐藏
  - 定制端口
  - 隐藏通讯
  - 新型隐身技术

木马的启动方式

- 经常用的方法主要有以下几种
  - 集成(捆绑)到应用程序中
  - 隐藏在Autoexec.bat和Config.sys中
  - 潜伏在Win.ini中
  - 在System.ini中藏身
  - 隐蔽在Winstart.bat中
  - 隐藏在应用程序的启动配置文件中
  - 伪装在普通文件中
  - 内置到注册表中
  - 隐形于启动组中
  - 修改文件关联
  - 修改运行可执行文件的方式
  - 设置在超级链接中

特洛伊木马技术的发展

- 木马的发展及成熟，大致也经历了两个阶段
  - Unix阶段
  - Windows阶段
- 木马技术发展至今，已经经历了4代
  - 第一代木马
    - 只是进行简单的密码窃取、发送等，没有什么特别之处
  - 第二代木马
    - 在密码窃取、发送等技术上有了很大的进步，冰河可以说是国内木马的典型代表之一
  - 第三代木马
    - 在数据传输技术上，又做了不小的改进，出现了ICMP等类型的木马，利用畸形报文传递数据，增加了查杀的难度
  - 第四代木马
    - 在进程隐藏方面，做了很大的改动，采用了内核插入式的嵌入方式，利用远程插入线程技术，嵌入DLL线程；或者挂接PSAPI(Process Status API)，实现木马程序的隐藏

木马程序的自动启动技术

- 木马程序的第一次运行，需要用户主动执行，这一次主要是利用伪装方式诱骗用户运行安装木马程序。此后，一般会在用户启动系统的同时自动加载木马程序
- 让程序自动运行的方法比较多
  - 加载程序到启动组，写程序启动路径到注册表的自动启动键值
  - 修改Boot.ini
  - 通过注册表里的输入法键值直接挂接启动
  - 通过修改Explorer.exe启动参数等方法

木马程序的自动启动技术

```
CRegKey hKey;
if(ERROR_SUCCESS==hKey.Create(HKEY_LOCAL_MACHINE,
                    "SOFTWARE\\MICROSOFT\\WINDOWS\\CURRENTVERSION\\RUN"))
{
    BOOL bWrite = TRUE;
    TCHAR lpBuf[_MAX_PATH];
    // 获取木马程序(当前正在运行的本程序)所在路径:
    GetModuleFileName(GetModuleHandle(NULL),lpBuf,sizeof(lpBuf));
    TCHAR szTemp[_MAX_PATH];
    DWORD dwCount = sizeof(szTemp);
    // 判断木马启动键值Tsys是否已设置:
    if(hKey.QueryValue(szTemp,"Tsys",&dwCount) == ERROR_SUCCESS)
    {
        if(_tcscmp(szTemp,lpBuf) ==0)
        {
            bWrite = !bWrite;
        }
    }
    // 未设置，需要在注册表中写之:
    if(bWrite)
        hKey.SetValue(lpBuf,"Tsys");
    hKey.Close();
}
```

木马的伪隐藏技术

- 进程、线程和服务
  - 进程：一个正常的Windows应用程序，在运行之后，都会在系统之中产生一个进程，同时，每个进程，分别对应了一个不同的PID(Progress ID，进程标识符)。这个进程会被系统分配一个虚拟的内存地址空间，一切相关的程序操作，都会在这个虚拟的空间中进行
  - 线程：一个进程，可以存在一个或多个线程，线程之间同步执行多种操作。通常，线程之间是相互独立的，当一个线程发生错误的时候，并不一定会导致整个进程的崩溃
  - 服务：一个进程当以服务的方式工作的时候，它将在后台工作，Windows 9x下不会出现在进程列表中，但是，在Windows NT/2000下，仍然会显示在进程列表中，并且可以通过服务管理器检查任何的服务程序是否被启动运行

木马的伪隐藏技术

- 伪隐藏与真隐藏
  - 隐藏木马的服务器端，可以伪隐藏，也可以是真隐藏
    - 伪隐藏：指程序的进程仍然存在，只不过是让他消失在进程列表中
    - 真隐藏：程序彻底地消失，不是以一个进程或者服务的方式工作
  - 伪隐藏的方法
    - 把木马服务器端的程序注册为一个服务就可以从进程列表中消失，因为系统不认为它是一个进程。但是，这种方法只适用于Windows 9x的系统，对于Windows NT、Windows 2000等，通过服务管理器，照样会发现在系统中注册过的服务

木马的伪隐藏技术

- Windows NT/2000下的伪隐藏
  - API的拦截技术，也称作进程欺骗技术。在Windows中有多种方法能够看到进程的存在：PSAPI（Process Status API）、PDH（Performance Data Helper）、ToolHelp API。如果能够欺骗用户或入侵检测软件用来查看进程的函数(例如截获相应的API调用，替换返回的数据)，就完全能实现进程隐藏
  - 例如，通过建立一个后台的系统钩子，拦截PSAPI的EnumProcessModules等相关的函数来实现对进程和服务的遍历调用的控制，当检测到进程ID(PID)为木马程序的服务器端进程的时候直接跳过，这样就实现了进程的隐藏

木马的伪隐藏技术

- 通过注册服务程序，实现进程伪隐藏

```
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {   DWORD dwVersion = GetVersion(); //取得Windows的版本号
        if (dwVersion >= 0x80000000)     // Windows 9x隐藏进程列表
        {
            int (CALLBACK *rsp)(DWORD,DWORD);
            HINSTANCE dll=LoadLibrary("KERNEL32.DLL");//装入KERNEL32.DLL
            //找到RegisterServiceProcess的入口:
            rsp=(int(CALLBACK *)(DWORD,DWORD))GetProcAddress(dll,
                                              "RegisterServiceProcess");
            rsp(NULL,1);                    //注册服务
            FreeLibrary(dll);               //释放DLL模块
        }
    }
    catch (Exception &exception)
    {    //处理异常事件
    }
     return 0;
}
```

木马的真隐藏技术

- 真隐藏则是让程序彻底地消失，不是以一个进程或者服务的方式工作
- 当进程为真隐藏的时候，那么这个木马服务器运行之后，就不应该具备一般进程的表现，也不应该具备服务的表现，也就是说，完全溶进了系统的内核
  - 不把木马作为一个应用程序，而把它作为一个线程、一个其他应用程序的线程，将其注入到其他应用程序的地址空间，而这个应用程序对于系统来说，是一个必不可少、绝对安全的程序，这样，就达到了彻底隐藏的效果，这样的结果，增加了查杀木马的难度

木马的真隐藏技术

- 利用DLL实现简单隐藏
  - 假设编写了一个木马DLL，并且通过别的进程来运行它，那么无论是入侵检测软件还是进程列表中，都只会出现该进程而并不会出现该木马DLL
  - 如果该进程是可信进程(如资源管理器Explorer.exe)，那么木马DLL作为该进程的一部分，也将成为被信赖的一员而为所欲为
  - 运行DLL文件最简单的方法是利用RunDLL32.exe
    - Rundll32.exe MyDll MyFunc
  - 如果在MyDll.DLL的MyFunc函数中实现了木马的功能，在系统管理员看来，进程列表中增加的是Rundll32.exe而并不是木马文件，这样也是木马的一种简易欺骗和自我保护方法

木马的真隐藏技术

- DLL木马
  - 使用RunDLL32的方法进行进程隐藏是简易的，但非常容易被识破
  - 比较高级的方法是使用DLL木马
    - 工作原理是替换常用的DLL文件，截获并处理特定的消息，将正常的调用转发给原DLL
      - 例如，Windows的Socket 1.x的函数都存放在wsock32.dll中，可以写一个wsock32.dll文件，替换原先的wsock32.dll(将其重命名为wsockold.dll，完全实现原DLL的功能是比较麻烦的事情，一般也没这个必要)
      - DLL木马wsock32.dll只做两件事：一是如果遇到不认识的调用，就直接转发给wsockold.dll(使用函数转发器forward)；二是遇到特殊的请求(事先约定的)就解码并处理。这样，理论上只要木马编写者通过Socket远程输入一定的暗号，就可以控制wsock32.dll(木马DLL)做任何操作

木马的真隐藏技术

- DLL木马技术是比较古老的技术，微软也对此做了相当的防范
  - 在Windows 2000的system32目录下有一个dllcache目录，一旦操作系统发现被保护的DLL文件被篡改(利用数字签名技术)，就会自动从dllcache中恢复该文件
  - 这个方法也不能算是DLL木马的最佳选择
- 采用动态嵌入技术的DLL木马
  - DLL木马的最高境界是动态嵌入技术
  - 动态嵌入技术是指将自己的代码嵌入正在运行的进程中的技术。Windows系统中的每个进程都有自己的私有内存空间，一般不允许别的进程对这个私有空间进行操作，但是实际上，仍然可以利用种种方法进入并操作进程的私有内存。在多种动态嵌入技术(窗口钩子即Hook函数、挂接API、远程线程)中，常用的是远程线程技术

木马的真隐藏技术

- 在进程中，可以通过CreateThread函数创建线程
- 通过CreateRemoteThread也同样可以在另一个进程内创建新线程，被创建的远程线程同样可以共享远程进程的地址空间
- 创建一个远程线程，进入远程进程的内存地址空间，就拥有了该远程进程的权限，可以启动一个DLL木马，甚至随意篡改该进程的数据

木马的真隐藏技术

- 动态嵌入的实现步骤
  - ①通过OpenProcess 函数打开试图嵌入的进程
    - 因为需要写入远程进程的内存地址空间，所以必须申请足够的权限，包括远程创建线程、远程VM操作、远程VM写权限
  - ②为LoadLibraryW函数线程启动DLL木马准备参数
    - LoadLibraryW函数是在kernel32.dll中定义的一个功能函数，用于加载DLL文件，它只有一个参数，就是DLL文件的绝对路径名(也就是木马DLL文件的全路径文件名)。由于木马DLL是在远程进程内调用的，所以还需要将这个文件名复制到远程地址空间
  - ③计算LoadLibraryW的入口地址，启动远程线程LoadLibraryW，通过远程线程调用木马DLL

木马的真隐藏技术

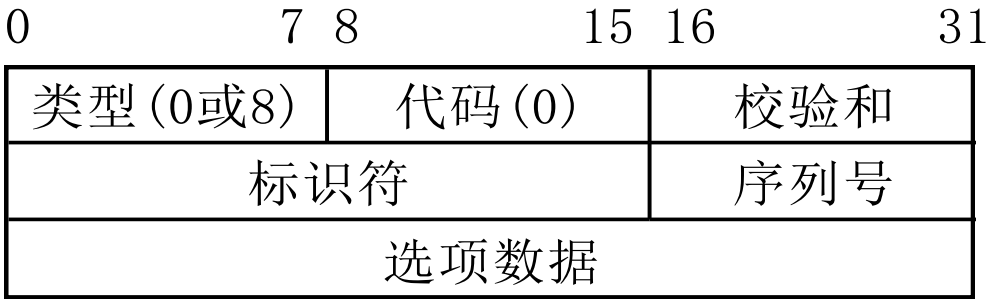
- 可以用远程线程技术启动木马DLL，也可以事先将一段代码复制到远程进程的内存空间，然后通过远程线程起动这段代码
- 无论是采取哪种方式，都是让木马的核心代码运行于别的进程的内存空间，这样不仅能很好地隐藏自己，也能更好地保护自己
- 此时的木马，不仅欺骗、进入用户的计算机，甚至进入了用户进程的内部。从某种意义上说，这种木马已经具备了普通病毒的很多特性，如寄生性(与宿主同生共死)

马的秘密信道技术

- 木马程序的数据传递方法
  - 利用TCP、UDP传输数据
    - 利用WinSock与目标机的指定端口建立连接，使用send和recv等API进行数据的传递
    - 这种方法的隐蔽性比较差，在命令行状态下使用netstat命令，就可以查看到当前的活动TCP、UDP连接
  - 攻击者为了不让用户察觉其与木马程序之间的通信，经常使用各种协议来建立控制服务端的秘密通信隧道
    - 利用ICMP协议建立秘密通道
    - 利用HTTP协议建立秘密通道

木马的秘密信道技术

- 利用ICMP协议建立秘密通道
  - ICMP回显请求(type=0)和回显应答(type=8)报文



  - 规范约定ICMP报文中的标识符和序列号字段由发送端任意选择，因此在ICMP包中标识符、序列号和选项数据等部分都可用来秘密携带信息
  - 由于防火墙、入侵检测系统等网络设备通常只检查ICMP报文的首部，因此使用ICMP建立秘密通道时往往直接把数据放到选项数据中
  - 这类秘密信道可以实现直接的客户端和服务端通信，具有准实时的特点

木马的秘密信道技术

- 利用HTTP协议建立秘密通道
  - 利用反弹端口型木马的“逆向连接”技术建立木马连接
  - 是合并端口法
    - 使用特殊的手段，在一个端口上同时绑定两个TCP或者UDP连接，通过把木马端口绑定于特定的服务端口之上，比如HTTP的80端口，使得其秘密通信对防火墙更具迷惑性，从而降低秘密通信流量被发现的风险
  - 使用报文伪装技术建立秘密信道
    - 将数据插入到HTTP协议报文的一些无用的段内，然后利用建立TCP连接的三次握手规则进行秘密通信

木马的远程监控技术

- 木马的远程监控技术，包括
  - 远程监视技术
    - 对服务端计算机的监视，包括对鼠标、键盘以及屏幕显示、甚至网络流量流向的监视，也包括对服务端计算机系统信息(如磁盘信息、操作系统信息、硬件信息)的搜集
  - 远程控制技术
    - 远程控制则是攻击者控制服务端计算机按照自己的意愿，运行某程序或关闭某服务，包括控制服务端计算机的鼠标、键盘、操作系统、文件系统，或者让其启动/停止某种服务程序，甚至关闭服务端计算机
  - 远程监控功能是木马最主要的功能，也是木马的最终目的

中木马后常出现的状况

- 发现以下异常，应当检查计算机是否感染了木马
  - 当浏览一个网站时，弹出来一些广告窗口是很正常的事情，可是如果用户根本没有打开浏览器，而浏览器突然自己打开，并且进入某个网站，那么，就要怀疑是否中了木马
  - 正在操作计算机，突然一个警告框或者是询问框弹出来，问一些用户从来没有在计算机上接触过的问题
  - Windows系统配置经常被莫名其妙地自动更改
  - 总是无缘无故地读硬盘，软驱灯经常自己亮起，网络连接及鼠标屏幕出现异常现象
  - 计算机意外地打开了某个端口，用嗅探器发现存在异常的网络数据传输
  - 拨号上网用户离线操作计算机时，突然弹出拨号对话框

检测和清除木马的方法

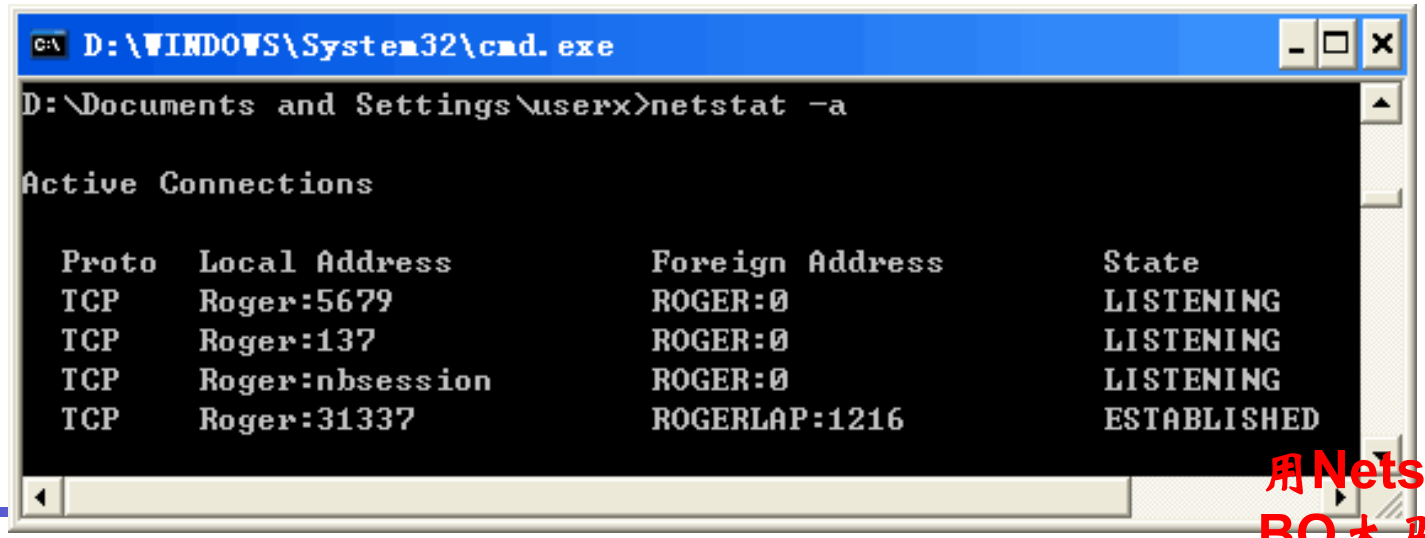
- 反击恶意代码，最佳的武器是最新的、成熟的病毒扫描工具
- 扫描工具能够检测出大多数特洛伊木马，并尽可能地使清理过程自动化
- 许多管理员过分依赖某些专门针对特洛伊木马的工具来检测和清除木马，但某些工具的效果令人怀疑，至少不值得完全信任，更何况任何工具软件在防治新木马时都存在一定的滞后性

检测和清除木马的方法

- 检测和清除木马的一般流程(对于动态嵌入式DLL木马，一般不是查看端口，而是查看内存模块)

查看端口 → 断开网络连接 → 查找并停止木马进程、线程 → 清除木马启动项及木马文件

  - 特洛伊木马入侵的一个明显证据是受害计算机上意外地打开了某个端口



用Netstat检测BO木马

检测和清除木马的方法

Windows XP的Netstat工具提供了一个新的-o选项，能够显示出正在使用端口的程序或服务的进程标识符(PID)。有了PID，用任务管理器就可以方便地根据PID找到对应的程序，以便终止之

木马“广外女生”的分析和清除

- 木马“广外女生”简介
  - 是广东外语外贸大学“广外女生”网络小组的作品，它可以运行于Windows 98、Windows 98SE、Windows ME、Windows NT、Windows 2000或已经安装Winsock2.0的Windows 95/97上
  - 该木马把启动项设在HKLM\ SOFTWARE \Classes\exefile\shell\open\command\下，这个注册表项的作用是定义运行可执行文件的格式
  - 木马将HKLM\ SOFTWARE \Classes\exefile\shell\open\command\下的键值由原来的“"%1" %*”修改为“C:\WINDOWS\System32\DIAGCFG.EXE "%1" %*”，包含了木马程序DIAGCFG.EXE，此后每次再运行任何可执行文件时都要先运行DIAGCFG.EXE，即启动木马

木马“广外女生”的分析和清除

- 木马端口
  - Fport是FoundStone出品的一个用来列出系统中所有打开的TCP/IP和UDP端口，以及它们对应应用程序的完整路径、PID标识、进程名称等信息的软件。在命令行中运行fport.exe，可以看到：

```
E:\Tools\Fport-2.0>fport

Pid  Process       Port    Proto  Path
584  tcpsvcs    -> 7       TCP    C:\WINDOWS\System32\tcpsvcs.exe
...
464  msdtc      -> 3372    TCP    C:\WINDOWS\System32\msdtc.exe
1176 DIAGCFG    -> 6267    TCP    C:\WINDOWS\System32\DIAGCFG.EXE
836  inetinfo   -> 7075    TCP    C:\WINDOWS\System32\inetsrv\inetinfo.exe
...
836  inetinfo   -> 3456    UDP    C:\WINDOWS\System32\inetsrv\inetinfo.exe
```

木马“广外女生”的分析和清除

- 清除木马
  - 由于“广外女生”木马自启动项的特殊性，如果先删除C:\WINDOWS\System32\目录下的DIAGCFG.EXE，将无法在系统中运行任何可执行文件。因此，清除该木马应按如下步骤，不能颠倒：
    - (1)按“开始”菜单，选择“运行”，输入regedit，按确定，打开下面键值：HKEY_LOCAL_MACHINE\SOFTWARE\Classes\exefile\shell\open\command\，但是不要修改，因为如果这时就修改注册表，DIAGCFG.EXE进程仍然会立刻把它改回来 (2)打开“任务管理器”，找到DIAGCFG.EXE这个进程，选中它，按“结束进程”来关掉这个进程。注意，一定也不要先关进程再打开注册表管理器，否则执行regedit.exe时又会启动DIAGCFG.EXE

木马“户外女生”的分析和清除

- (3)把HKEY_LOCAL_MACHINE\SOFTWARE\Classes\exefile\shell\open\command\的键值由原来的“C:\WINDOWS\System32\DIAGCFG.EXE "%1"%*”改为“"%1"%*”。这样，即使不删除DIAGCFG.EXE文件，只要用户不再双击运行之，木马就会因无激活机会而不能继续实施破坏
- (4)删除C:\WINDOWS\System32\目录下的DIAGCFG.EXE

- 如果要深入分析这个木马，可以在第(4)步中不删除它，而是把它拷贝到其他的目录以便研究

大纲

- 恶意软件的分类和区别
- 病毒的机理与防治
- 蠕虫的机理与防治
- 木马的机理与防治
- ✓ 其他恶意代码的机理
- 恶意代码分析技术

其他恶意代码的机理

- 移动代码
  - 移动代码的特殊性是可移动的并且可以在很多系统上运行而不需要终端用户显式地在系统上安装它们
  - 如网页交互页面，Java Applets、Active控件等
- 广告软件和间谍软件
  - 广告软件之自发强制地向终端用户做广告的程序
  - 基本思想：收集有关终端用户的浏览习惯和购物习惯的统计数据，然后利用这些信息选择有针对性的广告显示
  - 广告软件与免费软件捆绑发布

其他恶意代码的机理

- 粘人软件（**sticky software**）
  - 一些间谍、广告软件一旦被安装，便很难删除。这类软件在防止用户卸载方面想了很多办法
  - 如：3721，浏览器劫持等
- 网页恶意脚本程序
  - 这类恶意代码指利用.asp、.html、.vbs、.js类型的文件进行传播的基于VB Script和Java Script脚本语言并由Windows Scripting Host解释执行的一类恶意程序

其他恶意代码的机理

- 即时通信病毒
  - 第一类为以QQ、MSN等即时通讯软件为传播途径的病毒
  - 第二类为专门即时通讯软件本身的盗窃用户帐号、密码的病毒
  - 第三类似不断给用户发消息的骚扰型恶意代码
- 手机病毒
  - 以手机为感染对象，以手机网络和计算机网络为平台，通过发送病毒短信等形式对手机进行攻击，从而造成手机状态异常

其他恶意代码的机理

- 手机病毒
  - 手机病毒的主要有以下几个传播途径：
    - 捆绑在下载的程序文件中，利用电信运营商的无线下载通道传播
    - 通过手机本身的红外和蓝牙功能传播
    - 向手机发送垃圾信息，攻击和控制网关，导致网络运行瘫痪
    - 病毒短信或乱码电话方式
    - 移动互联网

其他恶意代码的机理

- 宏病毒
  - 宏病毒代码以“宏”的形式潜伏在Microsoft Office文档中
  - 所谓宏，就是软件设计者为了让人们在使用软件工作时，避免重复相同的动作而设计出的一种工具。它利用简单的语法，把常用的动作写成类似批处理命令的多行代码的集合，即宏。在工作时，直接自动运行编好的宏，去完成某项特定的任务，而不必再重复相同的动作

大纲

- 恶意软件的分类和区别
- 病毒的机理与防治
- 蠕虫的机理与防治
- 木马的机理与防治
- 其他恶意代码的机理
- ✓ 恶意代码分析技术

分析前的准备

- 恶意代码分析系统是一个与外界隔离的独立网络环境。其分析系统有两种基本类型：
  - 第一种是基于真实系统来建立，例如能够运行多个操作系统的常规PC，能从干净的系统迅速启动，能够迅速从备份中恢复到一个干净的状态
  - 第二种使用虚拟机软件。虚拟机软件可以运行多种操作系统的多个映像，可以快速重启各种干净的操作系统。或者是在专用系统上运行基于代码仿真的虚拟机
  - 另外，蜜罐系统，FireEye

分析过程的具体步骤

- 第一步是对可疑对象进行快速检查
- 第二步工作是用一个或一组反病毒扫描器进行过滤 → 已知病毒
- 第三步实对尚待分析的文件进行残留物清除
- 第四步是对可疑对象中常见的病毒代码的位置进行快速检查
- 第五步是分析对象中的字符串，使用“strings”这种工具，将字符串转储出来，使用理解Unicode字符串的工具

分析过程的具体步骤

- 第六步，使用反汇编工具在应用程序代码的入口点附近进行快速检查，观察该处是否存在异常的恶意代码
- 第七步，在有恶意代码的迹象下，可以借鉴一些软件安全测试的方法
- 第八步，由于大量的32位计算机病毒都是使用运行时工具如UPX或ASPACK加壳，且大部分运行时压缩工具不支持把压缩文件脱壳，即只能在加壳的文件被执行时，在内存中对其进行脱壳操作

FireEye

- http://www.fireeye.com
- FireEye 的安全防护方式是在客户的系统之上加载虚拟机器，任何进出客户系统的数据都要经过这些虚拟机器
- 因此 FireEye 可以观测所有的网络行为，如果这些数据包被认为是恶意的（无论是已知还是未知行为），虚拟机器就会阻止它们进入客户的网络
- 这种方式与传统靠病毒签名数据库匹配查找有很大的不同

加壳与脱壳

- 加壳：通过一系列的数学运算，将可执行程序或动态链接文件的编码进行改变，已达到缩小程序体积或加密程序编码的目的
- 运行加壳程序，首先执行外壳程序，外壳程序负责把用户原来的程序在内存中解压，并把控制权交还给真正程序
- 脱壳：把在内存中真正还原的程序抓取下来，修正后变成可执行文件

常用的加壳压缩工具

名 称	作 者	介 绍
ASPack	A. Solodovnikov	非常强大的 Win32 加壳工具，其压缩率、速度和兼容性都很好，是目前流行的一种加壳压缩工具
UPX	M. Oberhumer & L. Molnar	全能的 EXE 压缩工具，可用 UPX－D 命令脱壳
Petite	I. Luck	能够压缩 PE 文件的 code、data 等资源
PE－PACK	ANAKiN	一个自身体积小巧的压缩工具
PKLITE32	PKWARE，Inc.	32 位压缩工具(DLL/EXE)
WWPack32	P. Warezak & R. Wierzbicki	32 位压缩工具(DLL/EXE)
Shrinker	Blink Inc.	32 位压缩工具(DLL/EXE)

常用的脱壳工具

名 称	作 者	介 绍
ASPack unpacker	Bane	给 ASPack 的压缩 PE 文件脱壳
UnPEPack	M. o. D.	脱 PEPack 的壳
ProcDump32		“万能”脱壳工具，但已不再升级，只能脱去老版本压缩工具的壳。除了能脱去已知壳外，还可以脱去未知的壳，甚至可以手动的方法增强其脱壳能力

- 检测壳类型的工具

名 称	介 绍
FileInfo	脱壳前用来判断是否加壳，加了哪种壳
GetType	脱壳前用来判断是否加壳，加了哪种壳
TYP	脱壳前用来判断是否加壳，加了哪种壳
PEID	可以检查 600 种不同的包装器

分析过程的具体步骤

- 第九步，使用动态分析，并同时监控恶意代码的行为
  - 文件监控
  - 注册表变动监控
  - 进程和线程监控
  - 网络端口监控
  - 系统调用监控

小结

- 恶意软件的分类和区别
- 病毒的机理与防治
- 蠕虫的机理与防治
- 木马的机理与防治
- 其他恶意代码的机理
- 恶意代码分析技术

实验

- 编写一个文件型病毒，实现对特定目录（比如U盘）中的可执行文件进行扫描，并实现感染