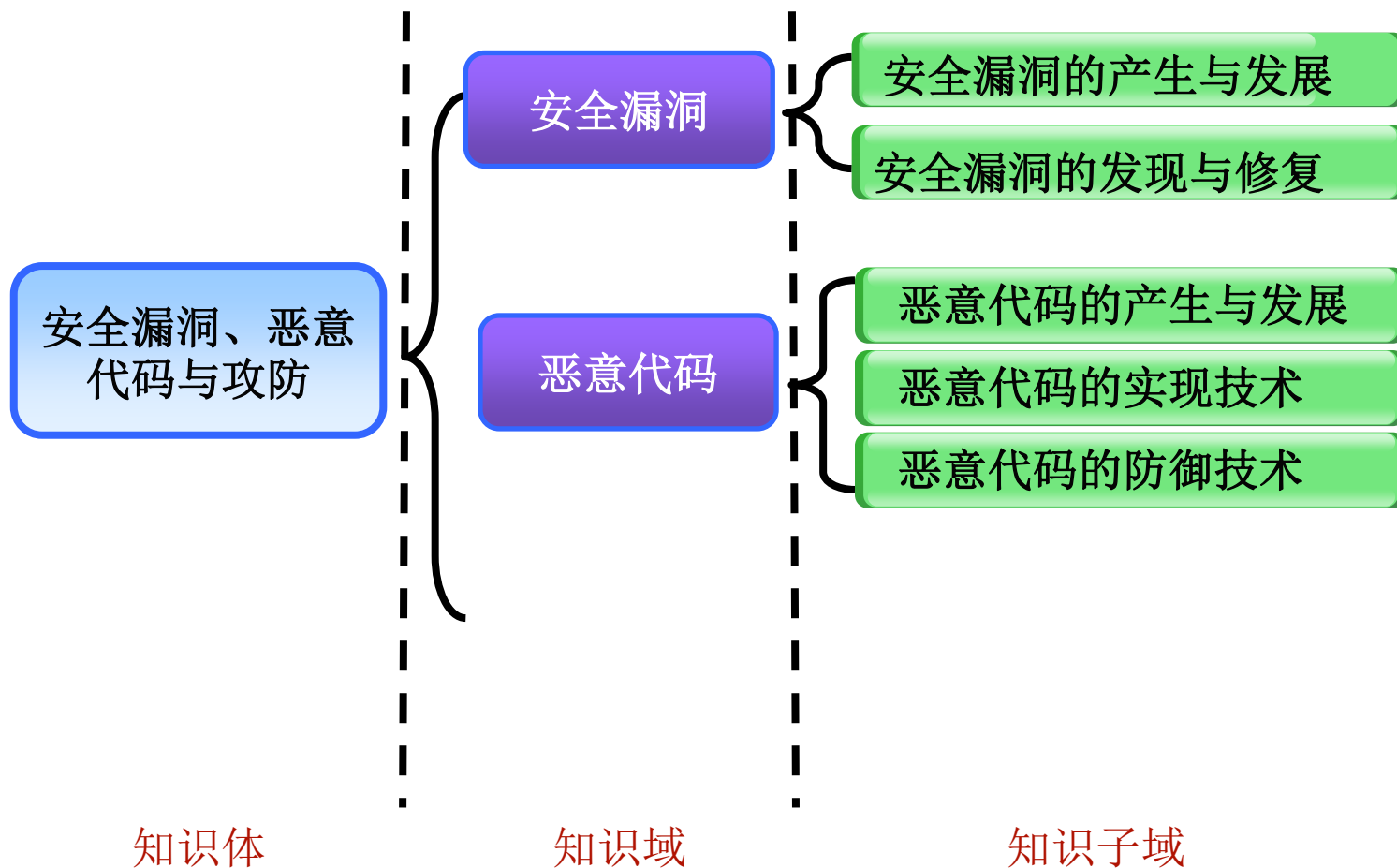


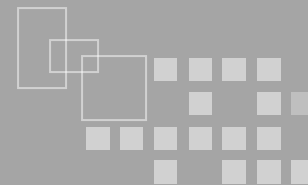


安全漏洞与恶意代码



课程内容



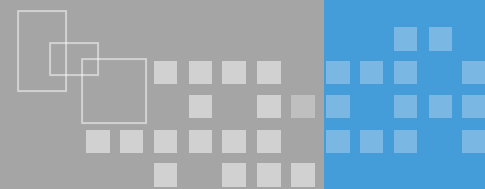


❖ 知识子域：安全漏洞的产生与发展

- 了解安全漏洞的含义
- 了解安全漏洞产生的原因
- 了解国内外常见安全漏洞分类
- 了解安全漏洞的发展趋势

❖ 知识子域：安全漏洞的发现与修复

- 了解安全漏洞的静态与动态挖掘方法的基本原理
- 了解补丁分类及修复时应注意的问题



❖ 漏洞概念的提出

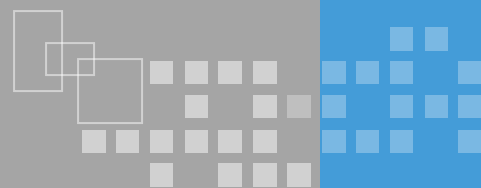
- 漏洞（Vulnerability）又叫脆弱性，这一概念早在1947年冯·诺依曼建立计算机系统结构理论时就有涉及，他认为计算机的发展和自然生命有相似性，一个计算机系统也有天生的类似基因的缺陷，也可能在使用和发展过程中产生意想不到的问题。

❖ 学者们对漏洞的定义

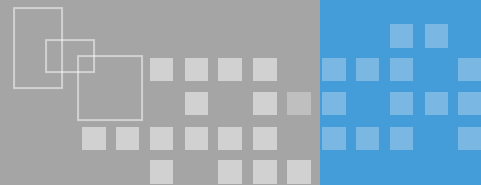
- 从访问控制角度定义（学者 Denning 做出的定义）
- 从风险管理角度的定义（学者Longstaff的定义）
- 使用状态空间描述的方法给出的定义（学者Bishop的定义）



标准机构的定义



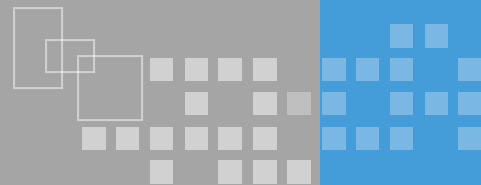
- ❖ 1999年，ISO/IEC15408（GB/T18336）定义：漏洞是存在于评估对象（TOE）中的，在一定的环境条件下可能违反安全功能要求的弱点；
- ❖ 2006年，美国NIST《信息安全关键术语词汇表》定义：漏洞是指存在于信息系统、系统安全过程、内部控制或实现中的，可被威胁源攻击或触发的弱点；
- ❖ 2006年，ISO/IEC SC 27《SD6：IT安全术语词汇表》定义：漏洞是一个或多个威胁可以利用的一个或一组资产的弱点；是违反某些环境中安全功能要求的TOE中的弱点；是在信息系统（包括其安全控制）或其环境的设计及实施中的缺陷、弱点或特性。



- ❖ 信息安全漏洞是信息技术、信息产品、信息系统在需求、设计、实现、配置、维护和使用等过程中，有意或无意产生的缺陷，这些缺陷一旦被恶意主体所利用，就会造成对信息产品或系统的安全损害，从而影响构建于信息产品或系统之上正常服务的运行，危害信息产品或系统及信息的安全属性。
- ❖ 错误、缺陷、弱点和故障并不等于漏洞。



漏洞产生的原因



❖ 技术原因

- 软件系统复杂性提高，质量难于控制，安全性降低
- 公用模块的使用引发了安全问题

❖ 经济原因

- “柠檬市场”效应

❖ 环境原因

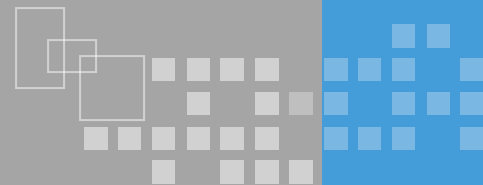
- 从传统的封闭、静态和可控变为开放、动态和难控
- 攻易守难

❖ 安全缺陷

- 安全性缺陷是信息系统或产品自身“与生俱来”的特征，是其固有成分



漏洞的分类



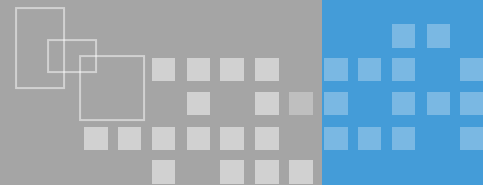
❖ 分类的目的

- 从各个方面来描述漏洞，如从漏洞的成因、利用漏洞的技术、漏洞的作用范围等；
- 用一个分类属性来表达漏洞的一个本质特征，而为漏洞的每个属性赋值的过程，就是给漏洞在该维属性上分类的过程；

❖ 分类原则：可接受性、易于理解性、完备性、确定性、互斥性、可重复性、可用性；



NVD漏洞分类

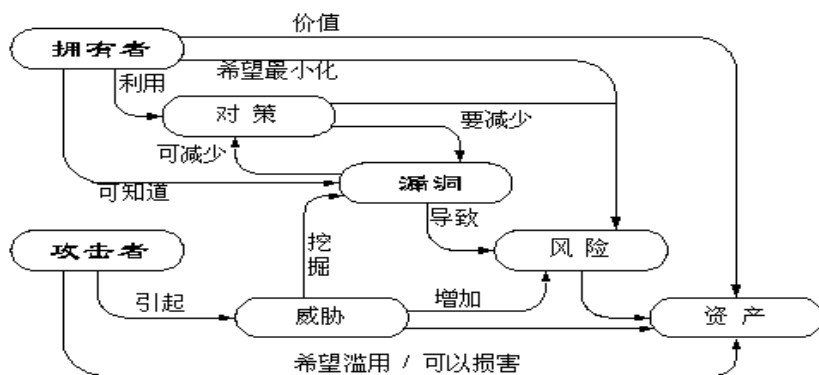


- ❖ 代码注入 (Code Injection)
- ❖ 缓冲错误 (Buffer Errors)
- ❖ 跨站脚本 (Cross-Site Scripting (XSS))
- ❖ 权限许可和访问控制 (Permissions, Privileges, and Access Control)
- ❖ 配置 (Configuration)
- ❖ 路径遍历 (Path Traversal)
- ❖ 数字错误 (Numeric Error)
- ❖ SQL注入 (SQL Injection)
- ❖ 输入验证 (Input validation)
- ❖ 授权问题 (Authentication Issues)
- ❖ 跨站请求伪造 (Cross-Site Request Forgery (CSRF))
- ❖ 资源管理错误 (Resource Management Errors)
- ❖ 信任管理 (Credentials Management)
- ❖ 加密问题 (Cryptographic Issues)
- ❖ 信息泄露 (Information Leak/Disclosure)
- ❖ 竞争条件 (Race Condition)
- ❖ 后置链接 (Link Following)
- ❖ 格式化字符串 (Format String Vulnerability)
- ❖ 操作系统OS命令注入 (OS Command Injections)
- ❖ 设计错误 (Design Error)
- ❖ 资料不足 (Insufficient Information) 。

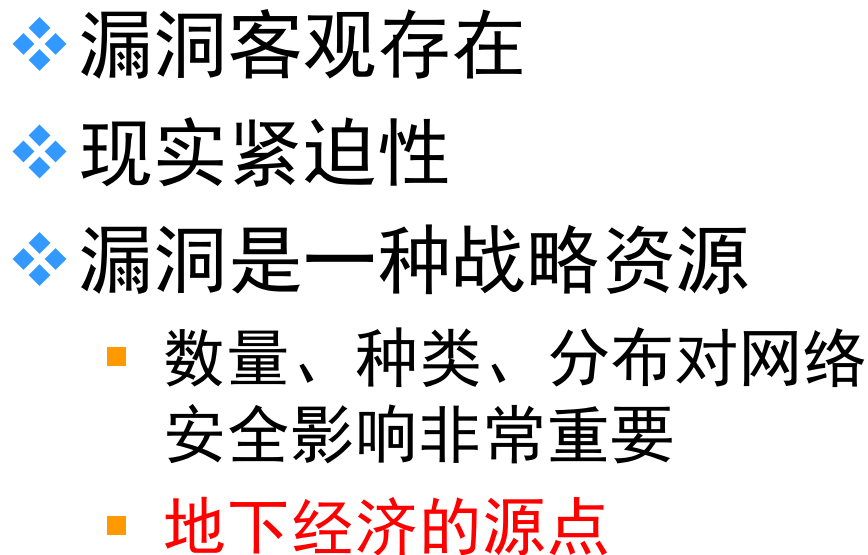


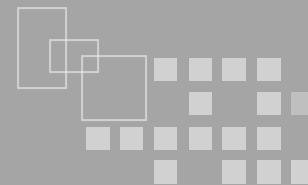
漏洞的危害

- ❖ 漏洞是信息安全的核心
- ❖ 漏洞无处不在
- ❖ 攻击者对漏洞的利用研究不断进步
- ❖ 漏洞的利用速度也越来越快



缓冲区溢出 SQL注入 路径遍历 跨站脚本 资源管理错误 权限许可和访问控制 跨站请求伪造
数字错误 授权问题 信任管理 配置错误 设计错误 输入验证 信息泄露 代码注入 加密问题
竞争条件 格式化字符串 操作系统命令注入 恶意链接 其他



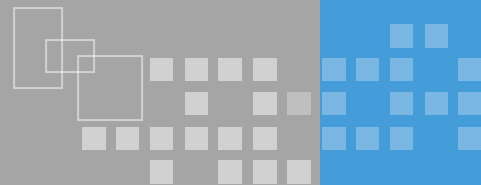


❖ 知识子域：安全漏洞的发现与修复

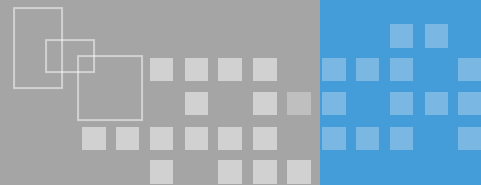
- 了解安全漏洞的静态与动态挖掘方法的基本原理
- 了解补丁分类及修复时应注意的问题



漏洞的发现



- ❖ 从人工发现阶段发展到了依靠自动分析工具辅助的半自动化阶段
- ❖ 漏洞发现方法
 - 静态漏洞检测
 - 动态漏洞检测



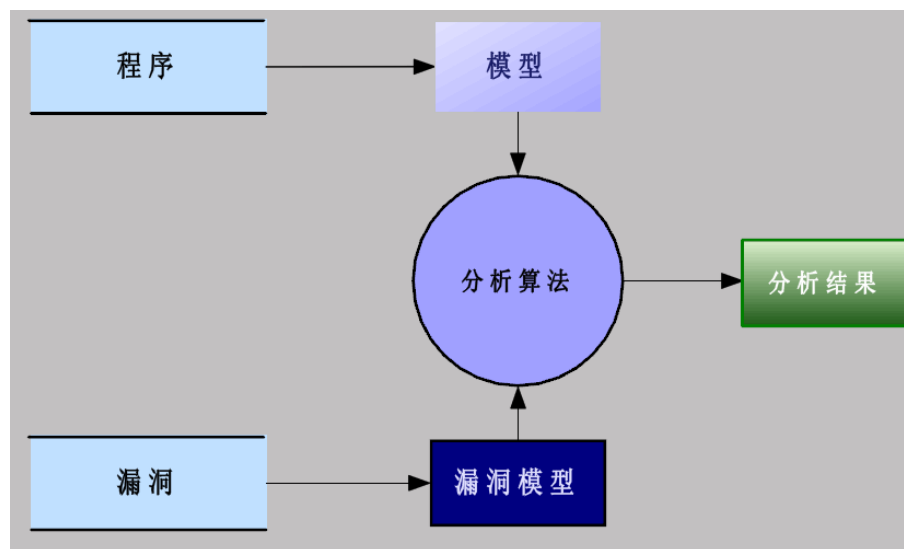
❖ 不运行代码而直接对代码进行漏洞挖掘的方法

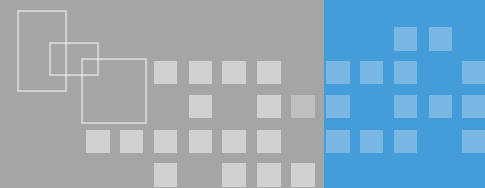
❖ 适用对象

- 完整的或不完整的源代码
- 二进制代码
- 中间代码片段

❖ 方法原理

- 流分析
- 符号执行
- 模型检测





❖ 控制流

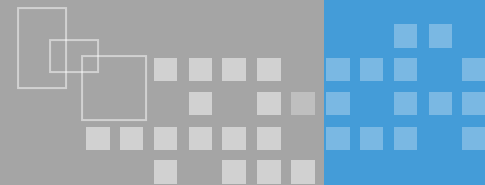
- 分析代码中控制流走向的信息，获得控制流图（CFG），即代码的控制结构信息。
- 控制流图是对代码执行时可能经过的所有路径的图形化表示，通过对代码中的分支、循环等关系的分析来获得代码的结构关系。

❖ 数据流

- 数据流分析是要得出程序中数据流动的信息，也就是程序中变量的相关信息，比如，可到达的变量定义，可用的表达式，别名信息，变量的使用及取值情况等



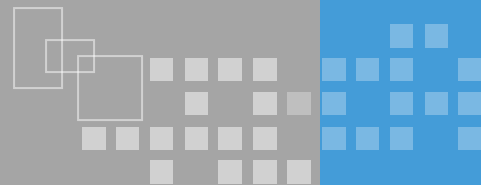
符号执行



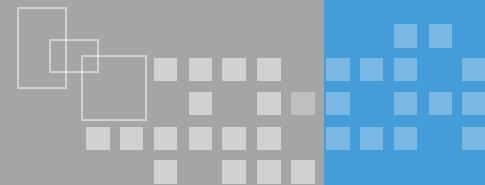
- ❖ 符号执行的目标是把程序转化成一组约束，同时检查程序模拟执行过程中的状态是否出错。这组约束中既包含程序中的路径条件，也包含要求程序满足的正确性条件或者程序员给出的断言。
- ❖ 符号执行的方法也是在程序的CFG上使用WorkList算法进行遍历。



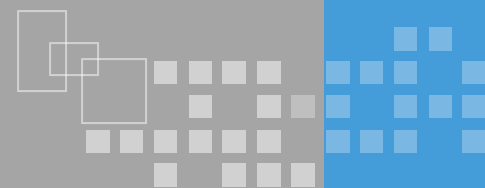
- ❖ 模型检测是给定被测系统的模型和目标属性的描述之后，可自动地对被测系统的状态空间进行穷尽搜索，以检测目标属性是否被满足。
- ❖ 度量指标：
 - 可靠性——被检测为真的任何属性，都确实为真，即无误报；
 - 完备性——所有确实为真的属性，必然可被检测出为真，即无漏报。



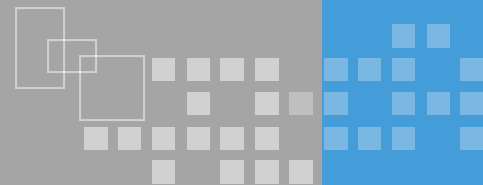
- ❖ 在代码运行的状态下，通过监测代码的运行状态或根据测试用例结果来挖掘漏洞的方法
- ❖ 特点
 - 与静态分析方法相比，动态分析方法的最大优势在于其分析结果的精确，即误报率较低
- ❖ 方法
 - 模糊测试
 - 渗透测试
 - 软件监测



- ❖ 在程序外部提供非预期输入，并监控程序对输入的反应，从而发现程序内部故障，广泛应用于软件安全测试
- ❖ 发展阶段：
 - 第一代主要用于健壮性和可靠性测试。
 - 第二代主要用于发现系统的漏洞。
 - 第三代智能模糊测试侧重于更合理的测试数据集的构造。



- ❖ 在程序运行时，标记某些信息，例如变量、存储单位、寄存器的值等，从而跟踪攻击路径，获取漏洞信息
- ❖ 步骤
 - 标识污点源，如不可信文件、不可信网络、各种输入
 - 分析污染源的传播
 - 根据触发机制，对具有污染标识的数据、内存等进行检查，从而发现可能的安全问题



❖ 渗透测试的概念

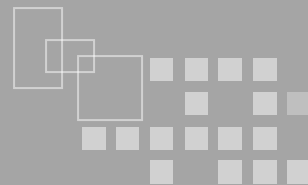
- 渗透测试是通过模拟攻击方法，来评估对象（系统或产品）安全的一种方法。

❖ 渗透测试的优势

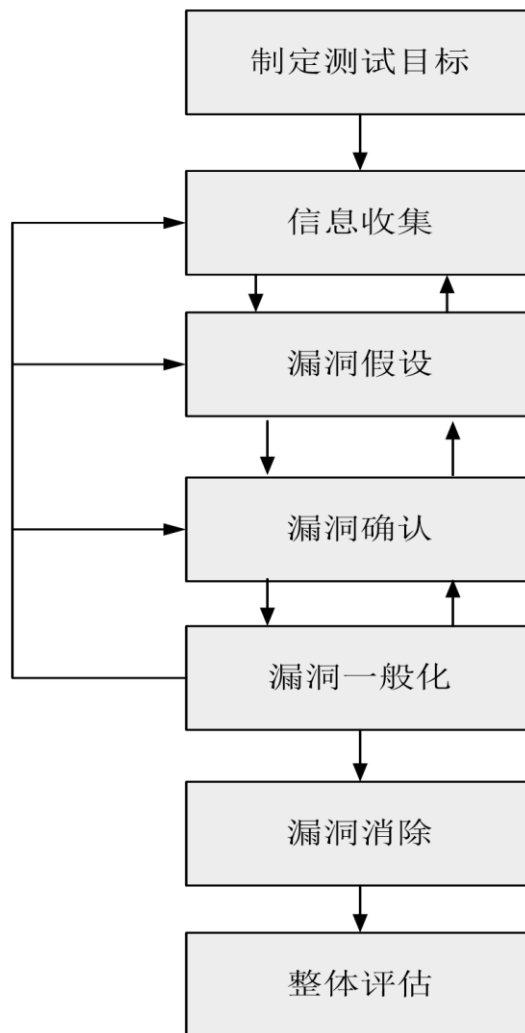
- 测试是基于软件运行的最后环境，因此，除了可以发现软件本身的安全问题外，更重要的是还可以发现一些关于环境和配置的安全问题。

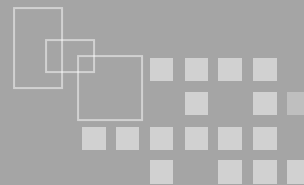


渗透测试的方法及步骤



- ❖ 授权与鉴别安全分析
- ❖ 非法操作分析
- ❖ 管理架构安全分析
- ❖ 规则有效性分析
- ❖ 性能隐患分析
- ❖ 核心安全功能强度分析
- ❖ 隐通道分析

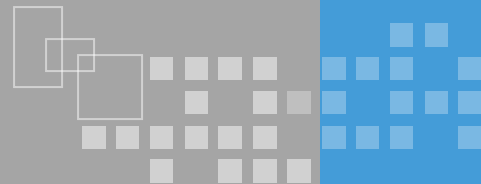




- ❖ 安装补丁是漏洞消减的技术手段之一。
 - 数据显示，及时安装有效补丁可避免约95%的信息安全损失
- ❖ 补丁修复中存在的两难问题：
 - 打什么样的补丁？——补丁质量问题
 - 如何打补丁？——操作方式问题
 - 什么时间打补丁？——修复时机问题



补丁分类



❖ 从文件类型

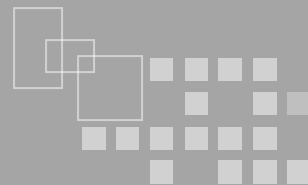
- 以源代码形式存在
- 以二进制形式存在

❖ 从内存角度

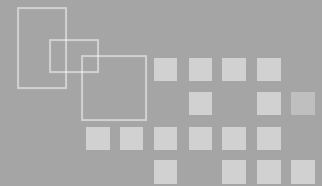
- 文件补丁（冷补丁）
- 内存补丁（热补丁）



补丁安装时应注意的问题



- ❖ 补丁安装部署之前需要经过必要的测试
- ❖ 需要从可靠来源不断获取最新补丁信息
- ❖ 安装补丁时需要做好备份和相应的应急措施



❖ 标准化的安全配置

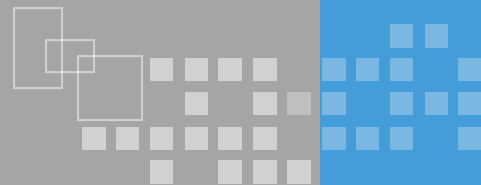
- 2002年，美国率先在军队推行标准化的安全配置与核查（IAVA）

❖ 根据漏洞分析和风险评估的安全加固

- 传统的安全加固手段越来越难以应付日益复杂的攻击行为，漏洞信息的及时披露和分发越来越重要。

❖ 加固核查与问责

- 通过安全审计核实漏洞消除情况和效果。



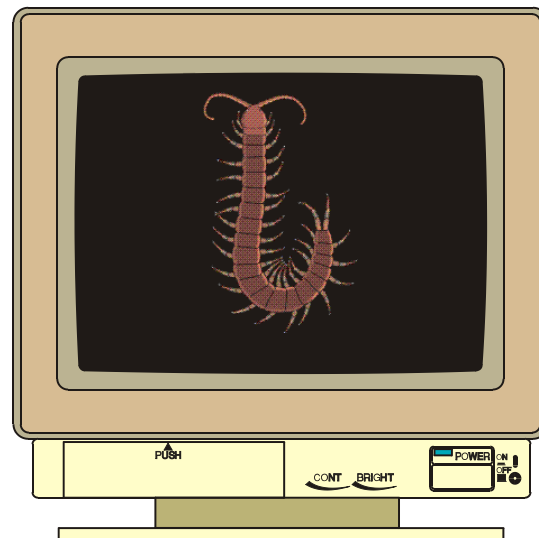
❖ 知识子域：恶意代码的产生与发展

- 了解恶意代码的发展历史及趋势
- 了解恶意代码的类型
- 理解恶意代码的传播方式



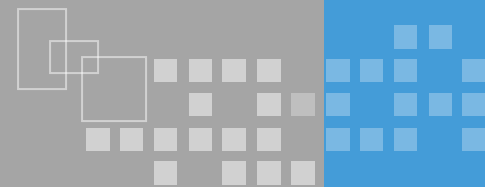
什么是恶意代码

- ❖ 没有有效作用、干扰或破坏计算机系统/网络功能的程序或代码（一组指令）
- ❖ 指令类型
 - 二进制代码
 - 脚本语言
 - 宏语言
 -
- ❖ 表现形式
 - 病毒、蠕虫、后门程序、木马、流氓软件、逻辑炸弹等





恶意代码的危害



❖ 网络拥塞

- 蠕虫传播或爆发占用大量网络资源，导致网络瘫痪

❖ 系统控制

- 形成危害严重的僵尸网络，被作者用来发动任何攻击

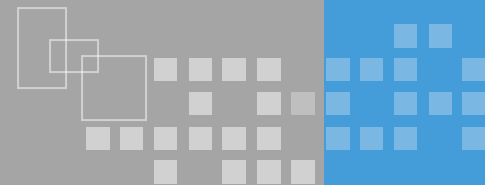
❖ 信息泄露

- 监视用户操作，窃取个人隐私

❖ 破坏系统及数据

❖

❖ 它已经成为网络犯罪的主要工具，也是国家、组织之间网络战的主要武器

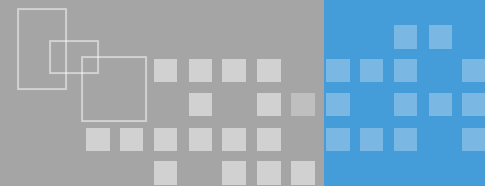


❖ 孕育和诞生

- 1949：冯·诺依曼在《复杂自动机组织论》提出概念
- 1960：生命游戏（约翰·康维）
磁芯大战（贝尔实验室三名程序员）
- 1977年科幻小说《P-1的青春》使得恶意程序有了计算机病毒的名称
- 1983：真正的恶意代码在实验室产生



恶意代码发展



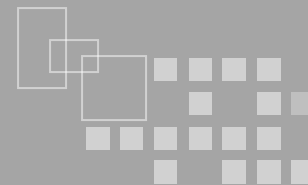
- ❖ 1986年—第一个PC病毒：Brain virus
- ❖ 1988年—Morris Internet worm—6000多台
- ❖ 1990年—第一个多态病毒
- ❖ 1991年—virus construction set—病毒生产机
- ❖ 1991年— DIR2病毒
- ❖ 1994年—Good Times (joys)
- ❖ 1995年—首次发现macro virus



罗伯特.莫里斯



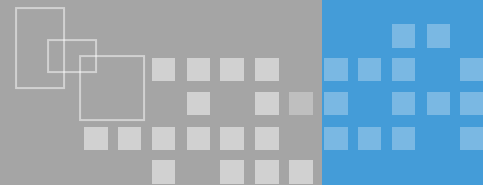
恶意代码发展



- ❖ 1996年—netcat的UNIX版发布 (nc)
- ❖ 1998年—第一个Java virus (StrangeBrew)
- ❖ 1998年—netcat的Windows版发布 (nc)
- ❖ 1998年—back orifice (BO) / CIH
- ❖ 1999年—melissa/worm (macrovirus by email)
- ❖ 1999年—back orifice (BO) for WIN2k
- ❖ 1999年—DOS/DDOS—Denial of Service TFT/trin00
- ❖ 1999年—knark内核级rootkit (linux)



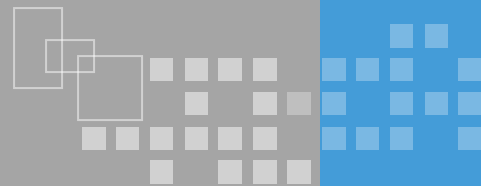
恶意代码发展



- ❖ 2000年—love Bug (VBScript)
- ❖ 2001年—Code Red - worm (overflow for IIS)
- ❖ 2001年—Nimda-worm (IIS/ web browser/outlook/file share etc.)
- ❖ 2002年—SQL slammer (sqlserver)
- ❖ 2003年—MSBlaster/ Nachi
- ❖ 2003年—中文上网
- ❖ 2004年—MyDoom/ Sasser
- ❖ 2006年—熊猫烧香

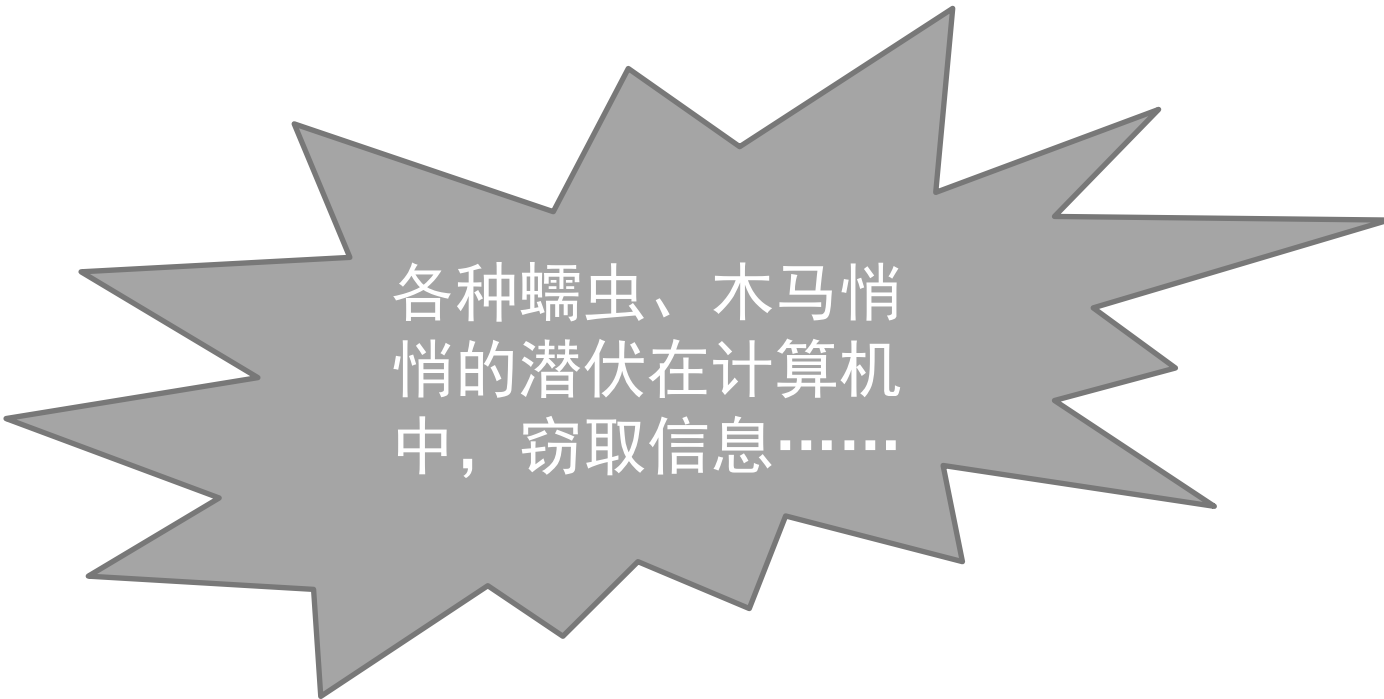


恶意代码发展



❖ 2010年—Stuxnet (工业蠕虫)

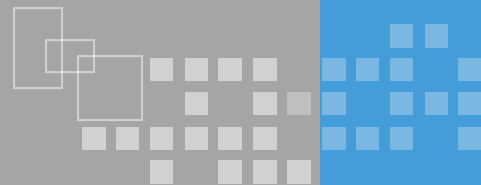
❖ 2012年—火焰病毒

A large, grey, multi-pointed starburst graphic with a black outline, containing text.

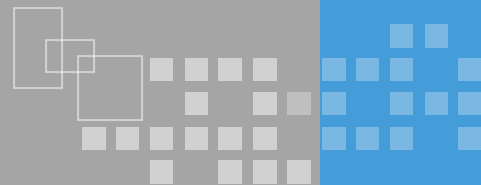
各种蠕虫、木马悄悄的潜伏在计算机中，窃取信息……



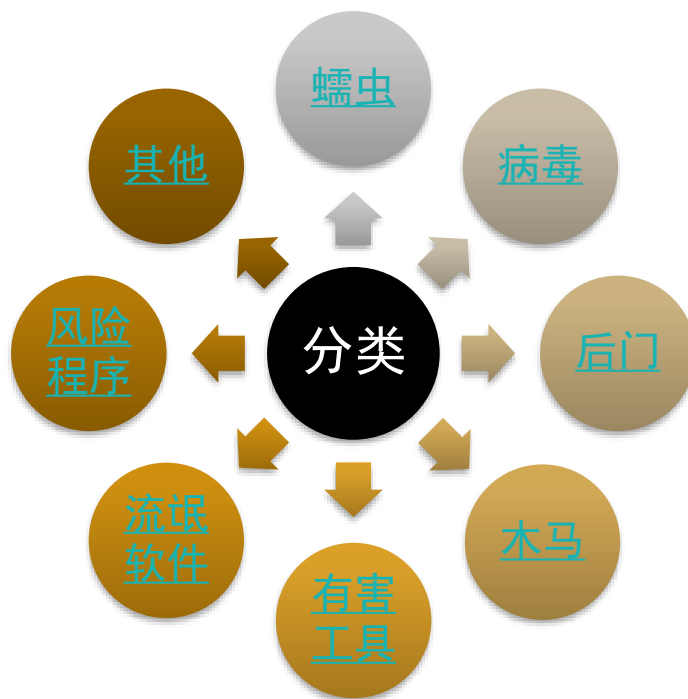
恶意代码的发展趋势



- ❖ 从传播速度上来看，恶意代码爆发和传播速度越来越快
- ❖ 从攻击意图来看，恶意代码的开发者越来越专业化，其意图也从游戏、炫耀专向为恶意牟利
- ❖ 从功能上来看，恶意代码的分工越来越细
- ❖ 从实现技术来看，恶意代码实现的关键技术不断变化
- ❖ 从传播范围来看，恶意代码呈现多平台传播的特征



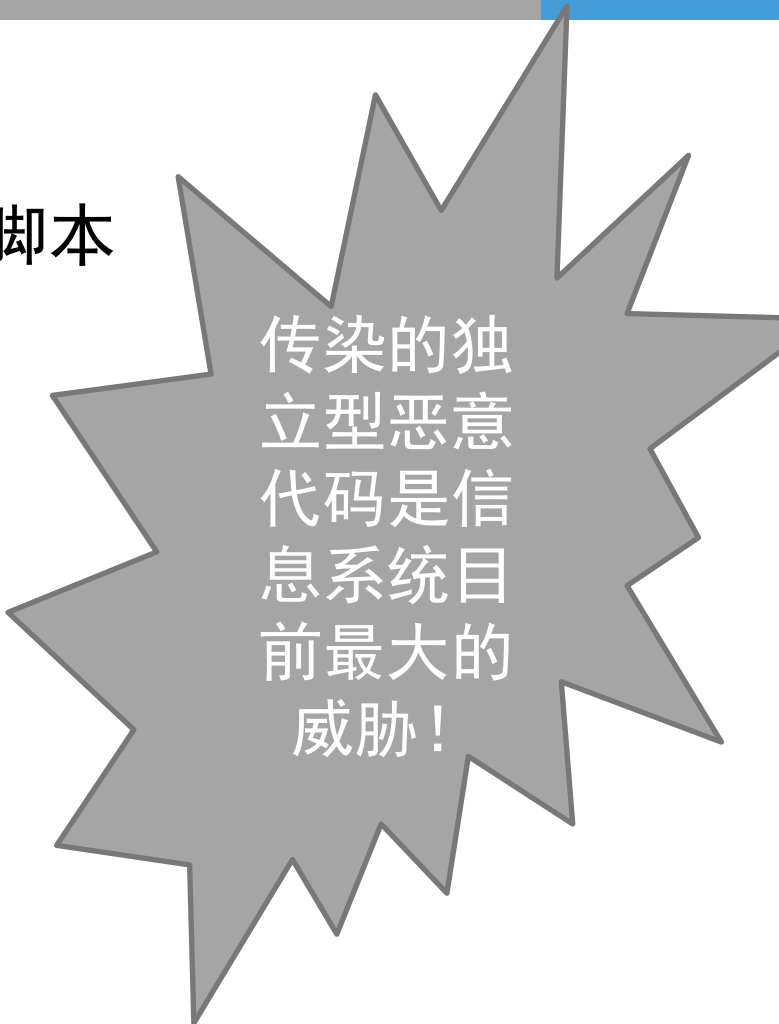
- ❖ 照恶意代码运行平台
- ❖ 按照恶意代码传播方式
- ❖ 按照恶意代码的工作机制
- ❖ 按照恶意代码危害





恶意代码分类

- ❖ 不传染的依附型恶意代码
 - 流氓软件、逻辑炸弹、恶意脚本
- ❖ 不传染的独立型恶意代码
 - 木马、rootkit、风险程序
- ❖ 传染的依附型恶意代码
 - 传统的病毒（CIH等）
- ❖ 传染的独立型恶意代码
 - 蠕虫病毒

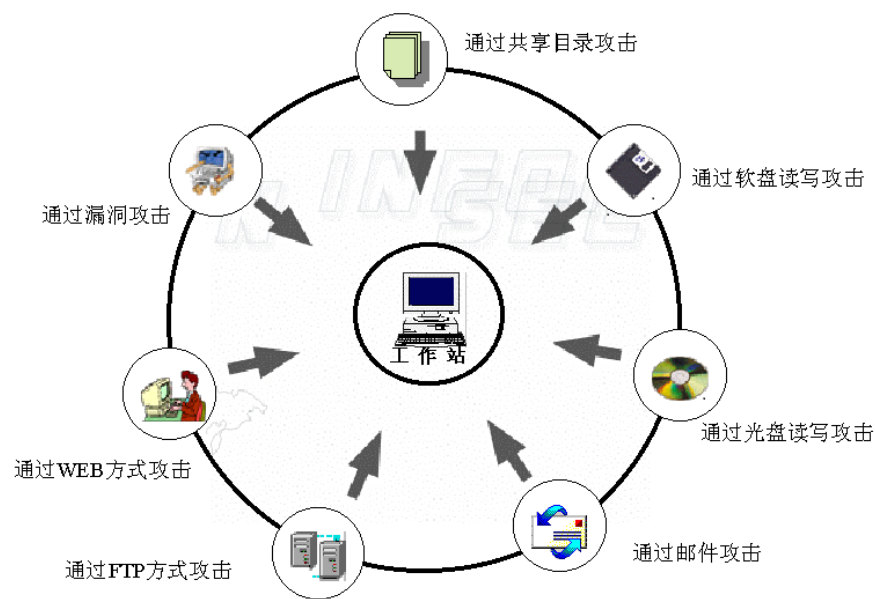


传染的独立型恶意代码是信息系统目前最大的威胁！



恶意代码的传播方式

- ❖ 移动存储
- ❖ 文件传播
 - 软件捆绑
- ❖ 网络传播
 - 网页
 - 电子邮件
 - 即时通讯
 - 共享
- ❖ 主动放置





恶意代码传播方式-移动存储

❖ 自动播放功能

- Windows默认
- 自动执行 autorun. inf 指定的文件

❖ 设置

- 组策略编辑器



[AutoRun]
OPEN=Autorun.exe
ICON=icon.ico



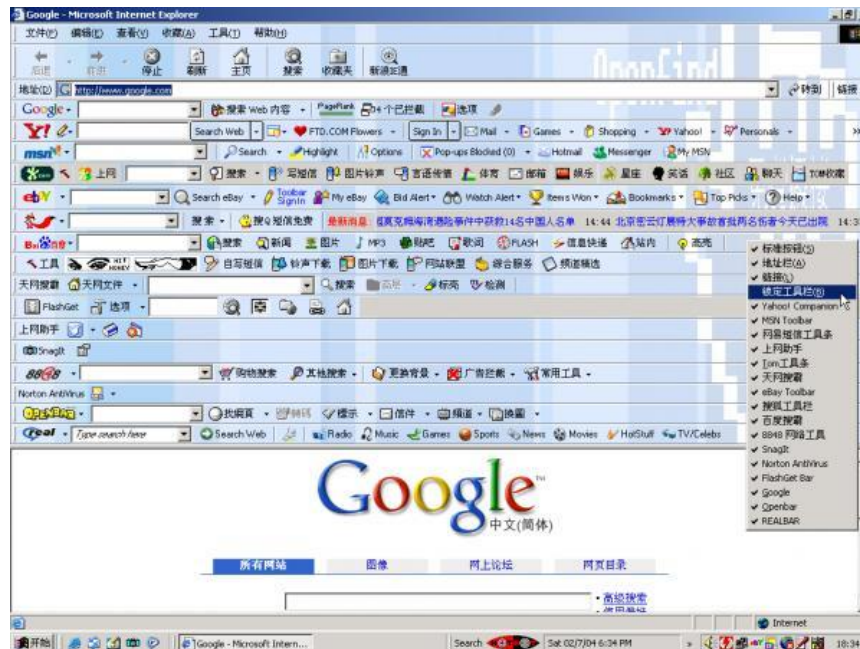
恶意代码传播方式-软件捆绑

❖ 强制安装

- 在安装其他软件时被强制安装上

❖ 默认安装

- 在安装其他软件是被默认安装上





恶意代码传播方式-网页

- ❖ 将木马伪装为页面元素
- ❖ 利用脚本运行的漏洞
- ❖ 伪装为缺失的组件
- ❖ 通过脚本运行调用某些com组件
- ❖ 在渲染页面内容的过程中利用格式溢出释放或下载木马





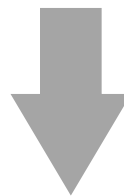
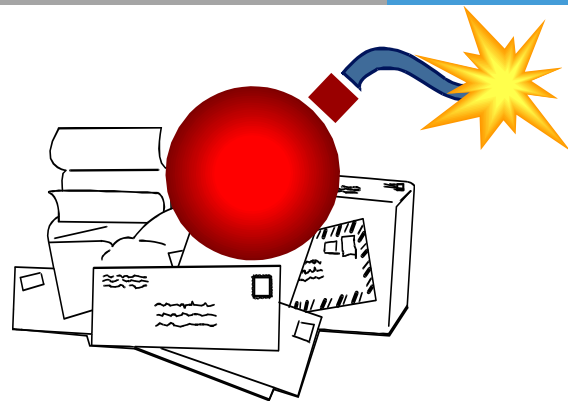
恶意代码传播方式-邮件

❖ 社会工程学

- 欺骗性标题
- 吸引人的标题
 - I love you病毒
 - 库娃等
-

❖ 利用系统及邮件客户端漏洞

- 尼姆达（畸形邮件MIME头漏洞）
-





恶意代码传播方式-通讯与数据传播

❖ 即时通讯

- 伪装即时通讯中的用户向其联系人发送消息。使用欺骗性或诱惑性的字眼

❖ P2P下载

- 伪造有效资源进行传播





恶意代码传播方式-共享

❖ 管理共享

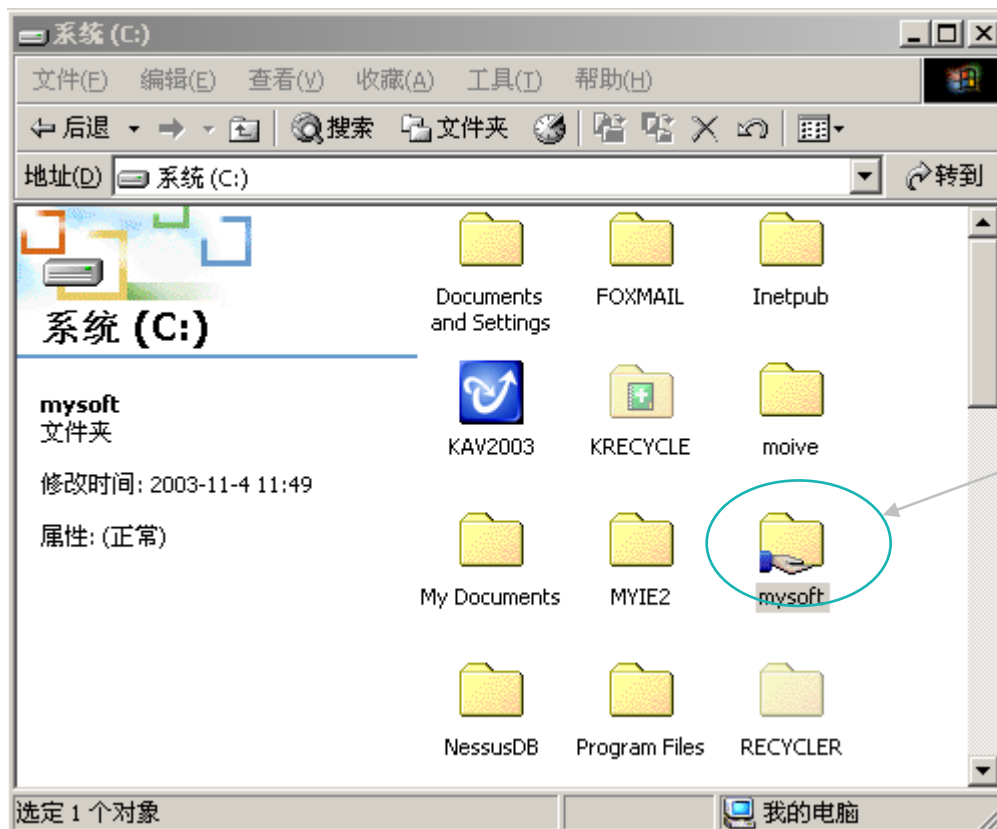
- C盘、D盘.....
- Windows安装目录

❖ 用户共享

- 用户设置的共享

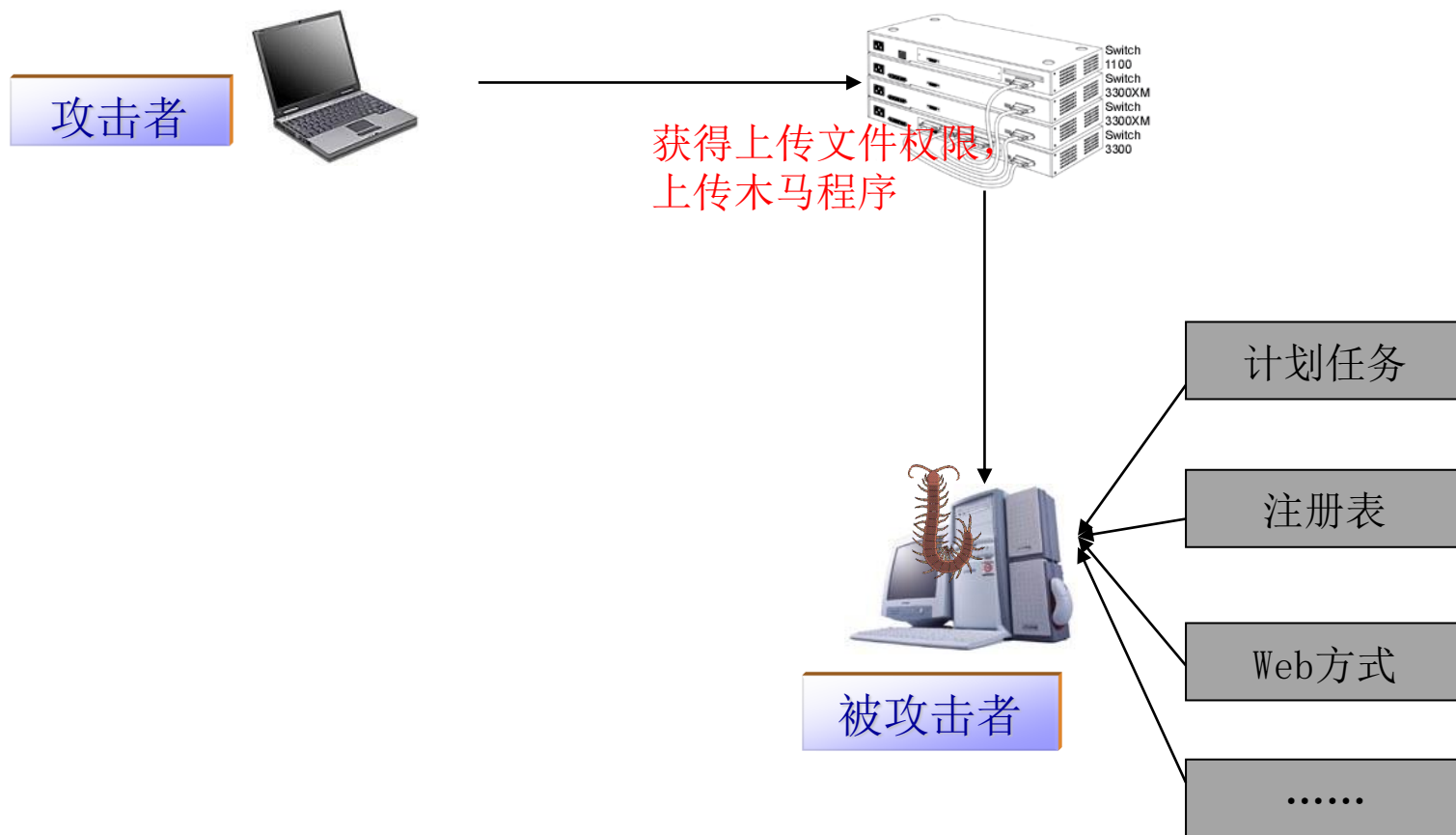
❖ 典型病毒

- Lovegate
- Spybot
- Sdbot
-



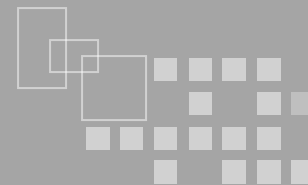


恶意代码传播方式-主动放置





恶意代码传播方式-漏洞

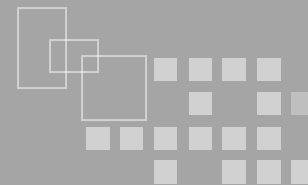


❖ 利用各种系统漏洞

- 缓冲区溢出：冲击波、振荡波

❖ 利用服务漏洞

- IIS的unicode解码漏洞：红色代码
-

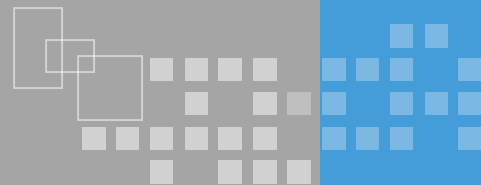


❖ 知识子域：恶意代码的实现技术

- 理解恶意代码修改配置文件、修改注册表、设置系统服务等加载方式
- 理解恶意代码进程、网络及系统隐藏技术
- 理解恶意代码进程保护和检测对抗自我保护技术



恶意代码加载方式



❖ 随系统启动而加载

- 开始菜单中的启动项
- 启动配置文件
- 注册表启动项
- 系统服务
- 组策略
-

❖ 随文件执行加载

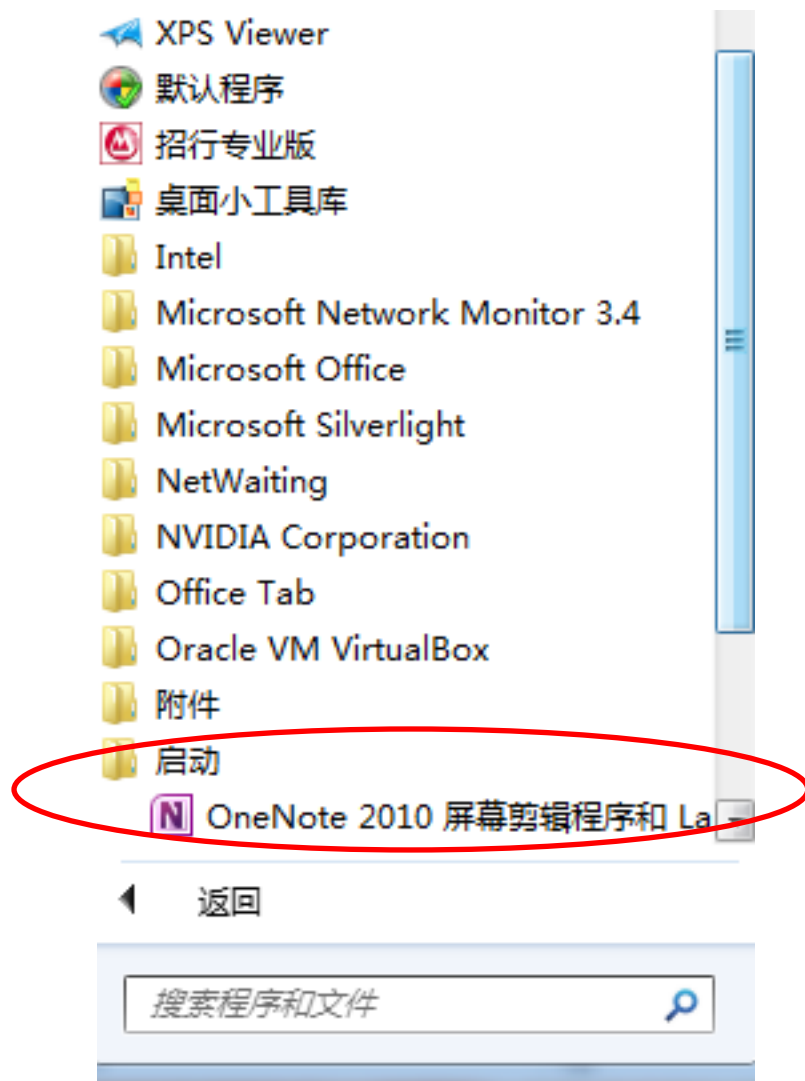
- 感染/文件捆绑
- 浏览器插件
- 修改文件关联

❖ 其他



随系统启动加载方式-启动配置

- ❖ 开始菜单启动项
- ❖ 启动配置文件
 - Autorun. bat
 - Win. ini
 - System. ini
- ❖ 已经很少有病毒采用
 - 过于明显
 - 用户登录后才能启动





随系统启动加载方式-注册表

❖ 优势

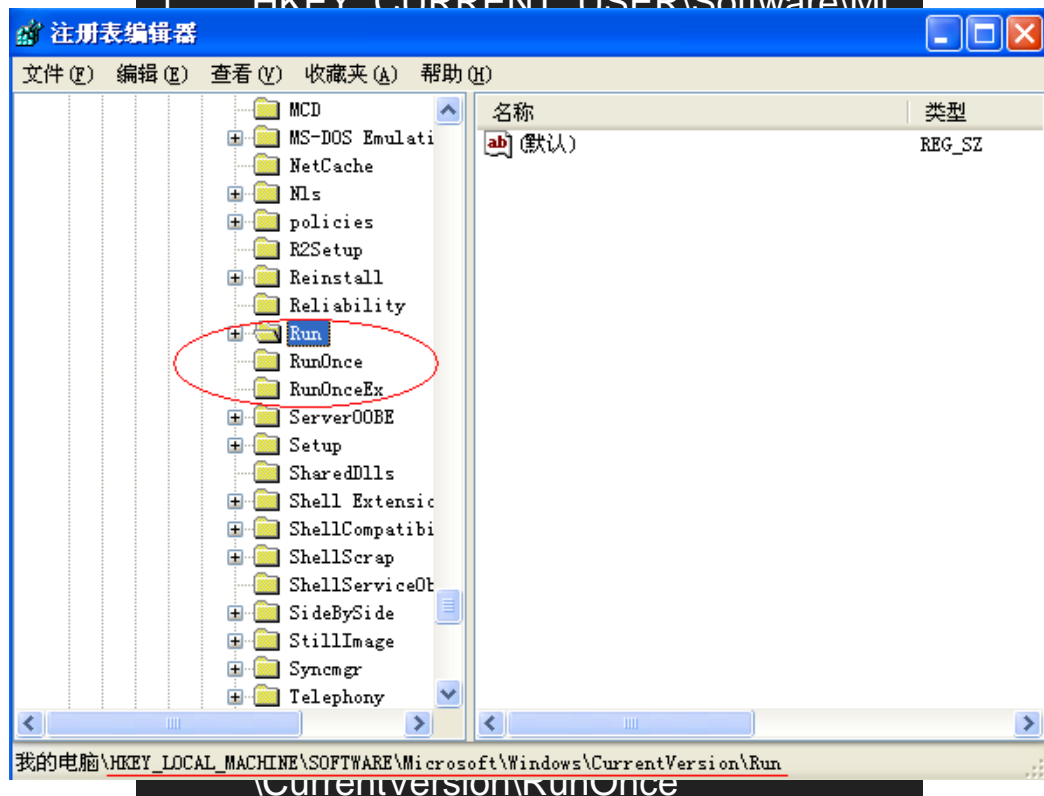
- 隐蔽性强
- 方式多样

❖ 加载位置

- Load键值
- User init键值
- Run
- RunServicesOnce
- RunServices
- RunOnce
-

注册表启动项：

1. HKEY_CURRENT_USER\Software\Mi



6.



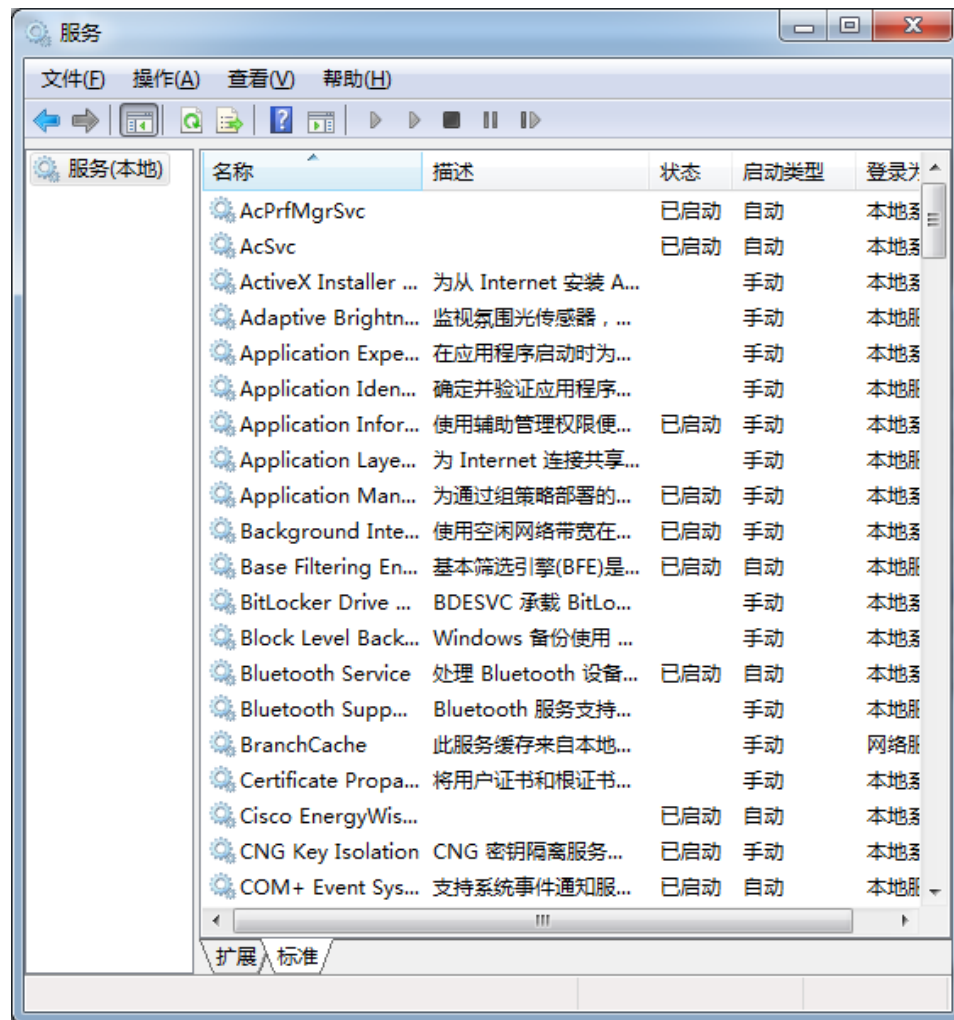
随系统启动加载方式-服务

❖ 优势

- 隐蔽性强
- 无需用户登录
- 权限较高

❖ 加载方式

- 单独服务
- 替换系统服务程序





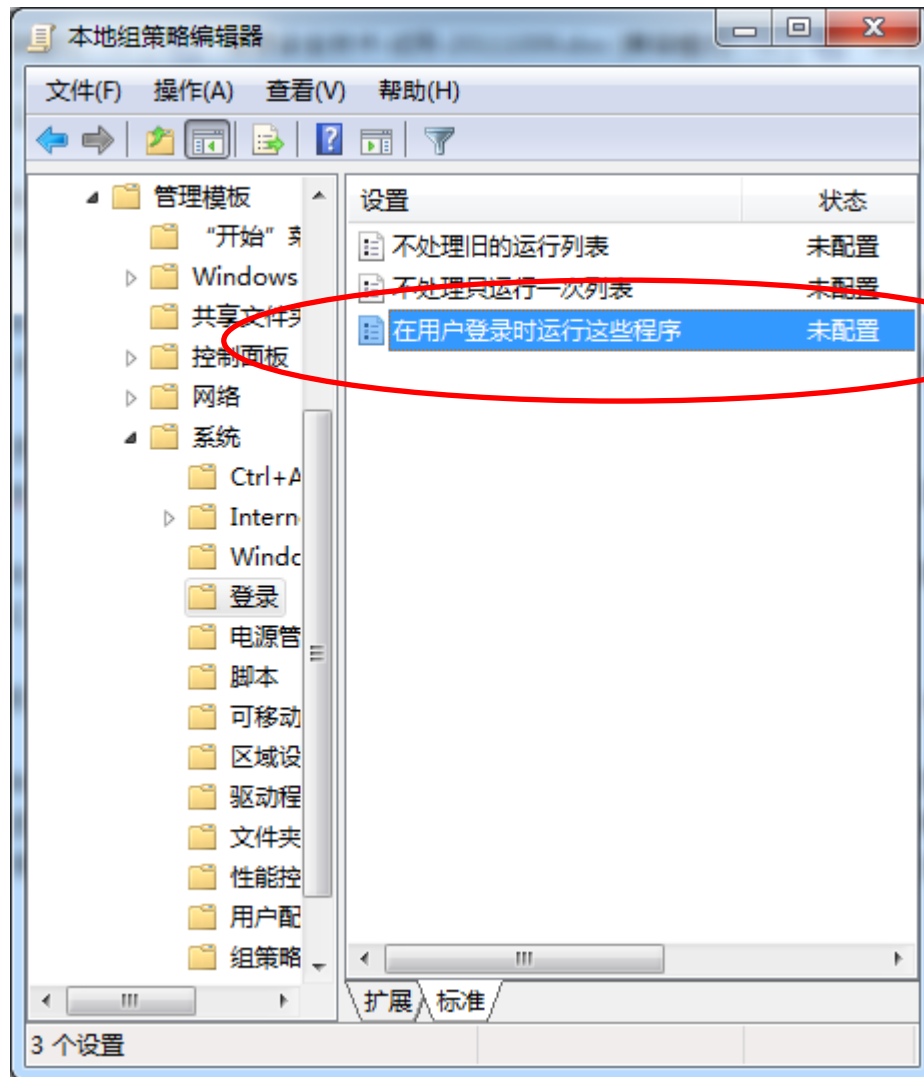
随系统启动加载方式-组策略

❖ 优势

- 类似启动项，但隐蔽性更高

❖ 不足

- 需要用户登录





随文件执行加载方式-感染/文件合并

- ❖ 传统病毒
- ❖ 宏病毒
- ❖ 程序合并

软件大小: 1.05 M	软件性质: 共享软件
更新时间: 2009-05-12	联系人:
下载次数: 160994	开发商:
软件语言: 简体中文	软件类别: 国产软件 / 文件处理
应用平台: Win2003/WinXP/Win2000/WinNT/Win9x	

 下载地址

 网友评论

 下载帮助

 收藏该页

 这软件不错
77.8%(399)

顶

 这软件很差
22.2%(114)

踩

软件介绍:

这个软件能够将两个EXE文件合并在一起, 生成一个新的EXE文件。运行这个新的EXE文件, 就相当于运行原来的两个EXE文件。适合于程序补丁制作, 本软件还可以彻底粉碎文件功能, 使其不可恢复。

2009 最新增加EXE文件加密功能, 它可以给任何EXE文件加口令 (包括已经用其它任何工具压缩或加密过的EXE文件), 您可以用来加密您的程序, 只有您能使用, 加密后完全独立运行, 支持运行带有参数的EXE文件 (如notepad.exe、QQ.exe等)、支持限制解锁口令次数、

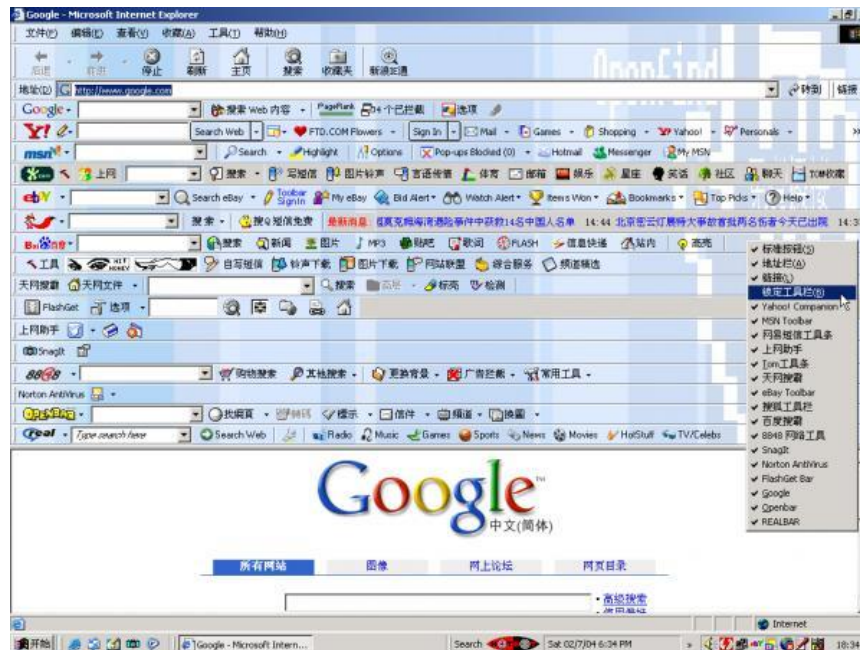
[更多介绍](#)



随文件执行加载方式-浏览器插件

❖ 优势

- 隐蔽性强
- 清理困难





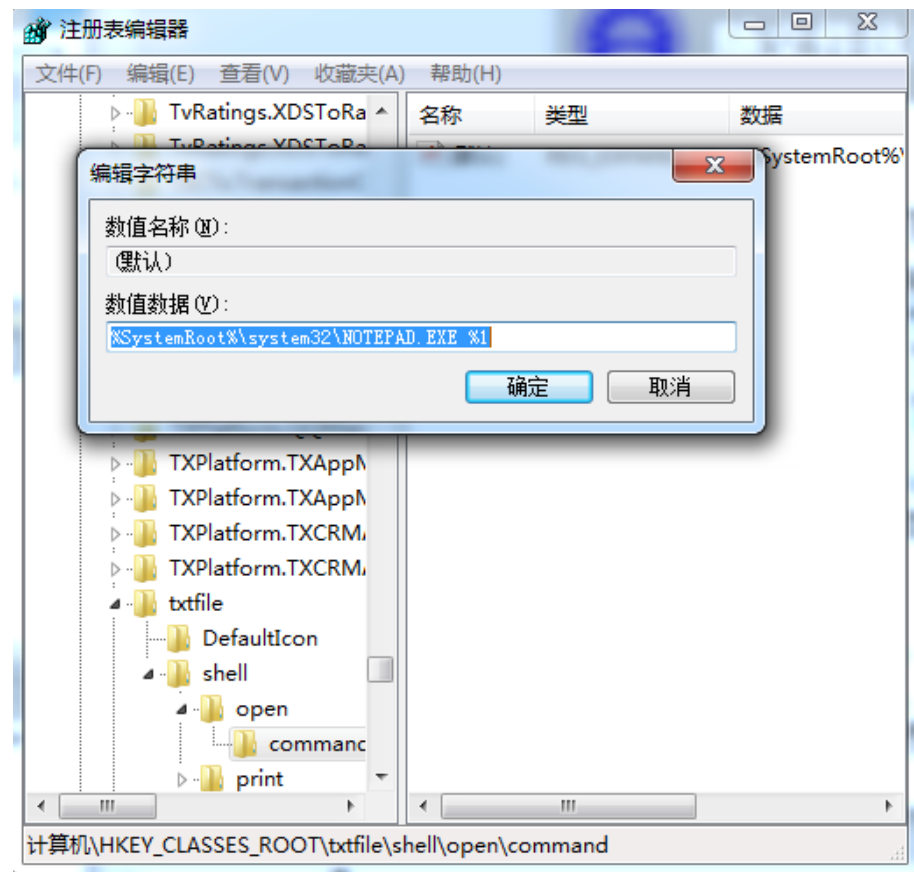
随文件执行加载方式-修改文件关联

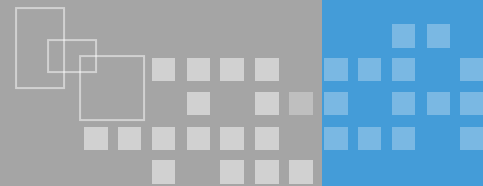
❖ 原理

- 正常情况下
 - 文本文件 (.txt) 关联到记事本
notepad.exe 打开
- 病毒修改
 - 文本文件 (.txt) 关联到病毒文件打开

❖ 优势

- 隐蔽性强，可关联任意类型文件，甚至可以关联目录操作





❖ 进程隐藏

- 进程迷惑
- DLL注入

❖ 网络隐藏

- 端口复用
- 无端口
- 反向端口

❖ 系统隐藏

- 隐藏、系统文件
- 流文件隐藏
- Hook技术



❖ 随机进程名

- 每次启动生成不一样的进程名，退出后无法查找

❖ 系统进程类命名

- Windows.exe
- System1.exe
- Kernel.exe

❖ 相似进程名

- 同名不同路径的进程

c:\windows\system32\iexplore.exe(trojan)

c:\Program Files\Internet Explorer\iexplore.exe(right)

- 名称相近的程序

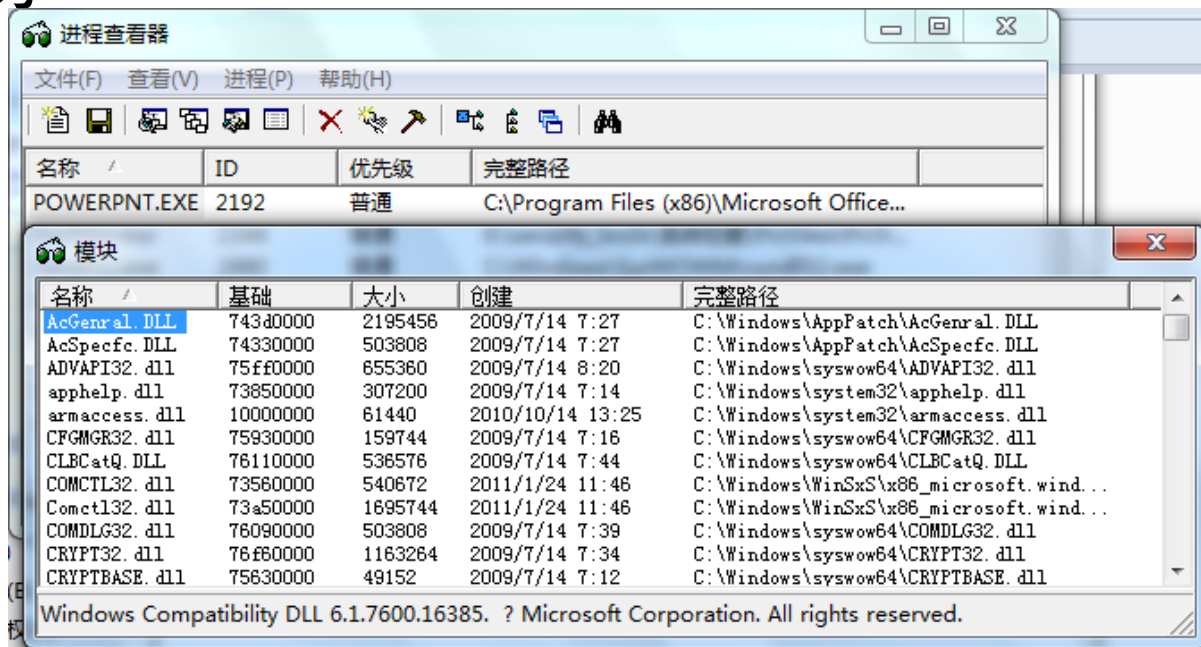
svchost.exe(right)

svch0st.exe(trojan)



恶意代码进程隐藏技术-DLL注入

- ❖ 动态链接库文件（DLL）概念
- ❖ 什么是DLL注入
 - DLL注入技术是恶意代码将DLL文件放进某个进程的地址空间里，让它成为那个进程的一部分
- ❖ DLL注入的优势
 - 无进程
 - 隐蔽性强
 - 清除难度大





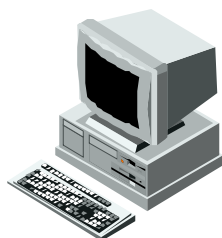
恶意代码网络隐藏技术-端口复用/无端口

❖ 端口复用技术

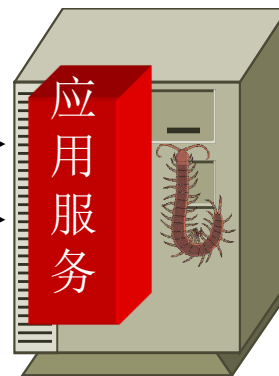
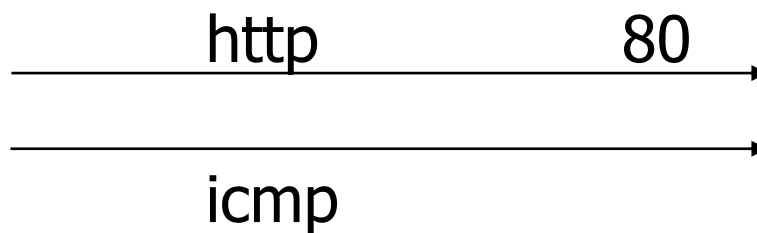
- 重复利用系统网络打开的端口（如25、80、135和139等常用端口）传送数据，这样既可以欺骗防火墙，又可以少开新端口
- 端口复用是在保证端口默认服务正常工作的条件下，具有很强的欺骗性

❖ 无端口

- 使用无端口的协议



客户机

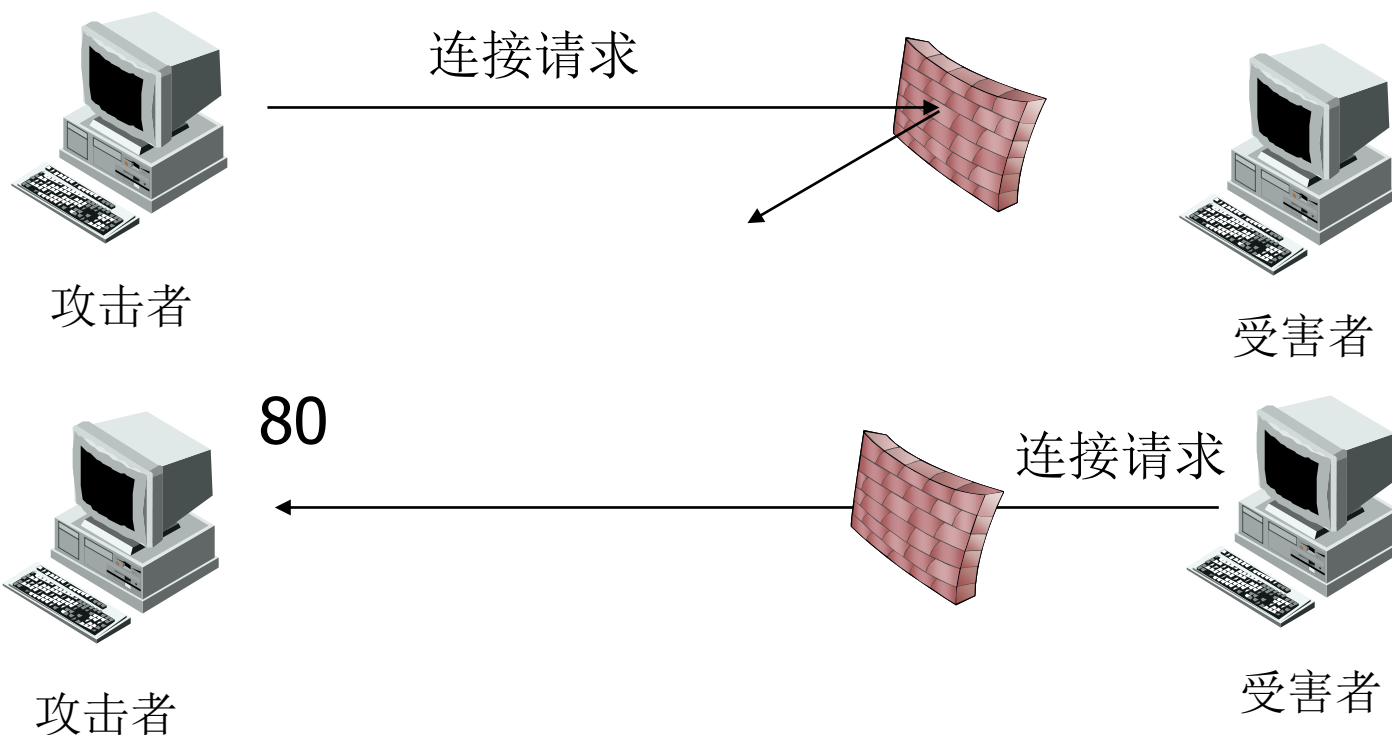


服务器



恶意代码网络隐藏技术-反弹端口

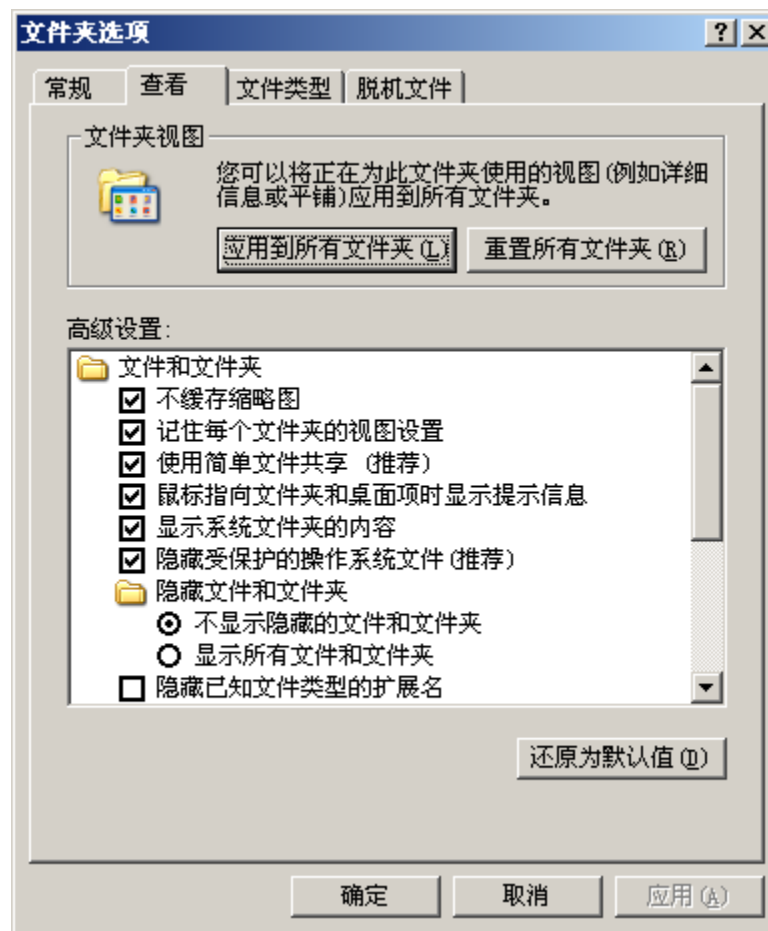
❖ 端口反向连接技术，系指恶意代码攻击的服务端（被控制端）主动连接客户端（控制端）。





恶意代码隐藏技术-系统隐藏

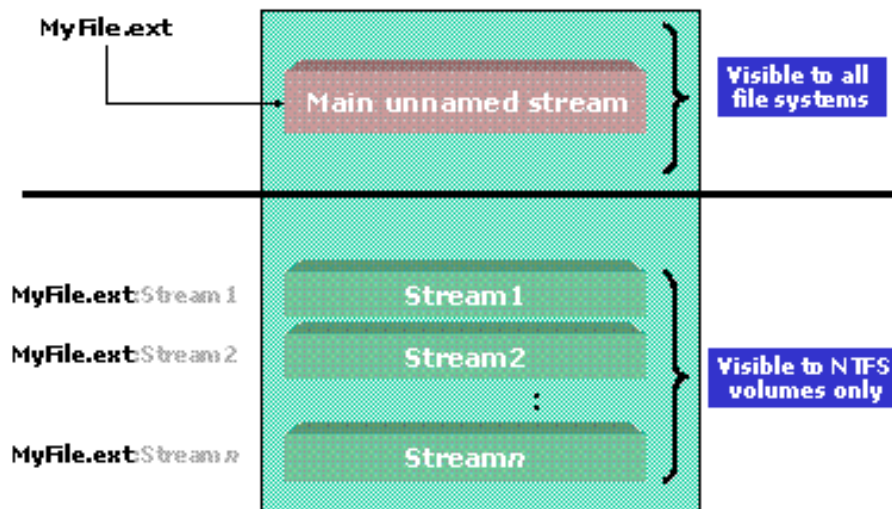
- ❖ 默认情况下，windows不显示隐藏文件和系统文件，恶意代码将自身属性设置为隐藏和系统文件以实现隐藏





❖ ADS (Alternate Data Streams) 交换数据流

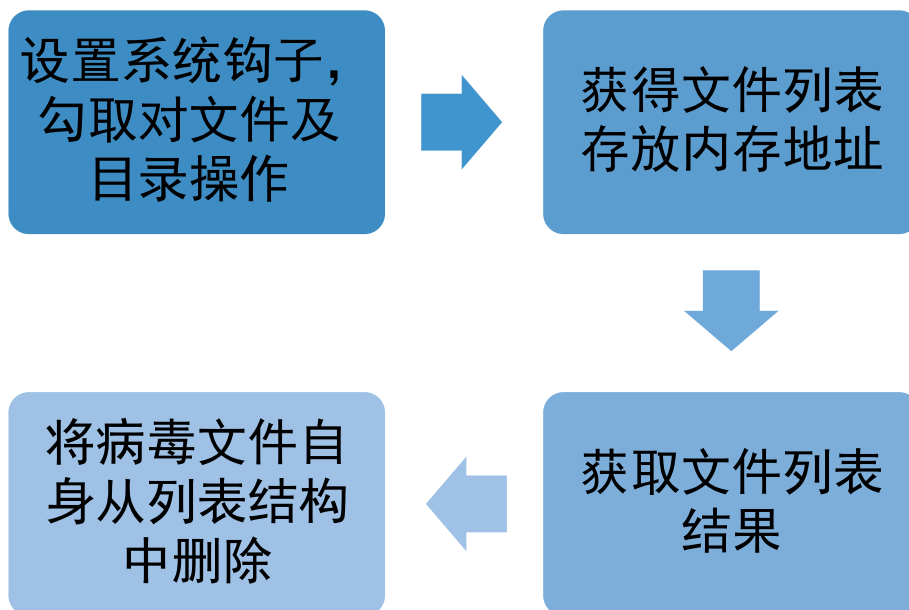
- NTFS 文件系统下，每个文件都可以有多个数据流
- 一个文件以流的形式附加到另一个文件（载体）中，流文件对explorer.exe等文件管理软件不可见，病毒可以利用此方式进行隐藏

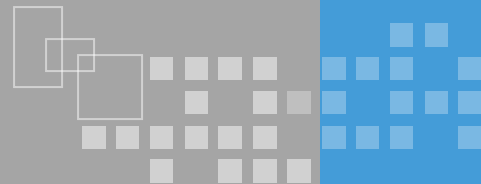




❖ Hook（系统钩子）

- 钩子机制允许应用程序截获处理window消息或特定事件。
- 在目标窗口处理消息前处理它





❖ 进程保护

- 进程守护
- 超级权限

❖ 检测对抗

- 反动态调试
- 反静态调试

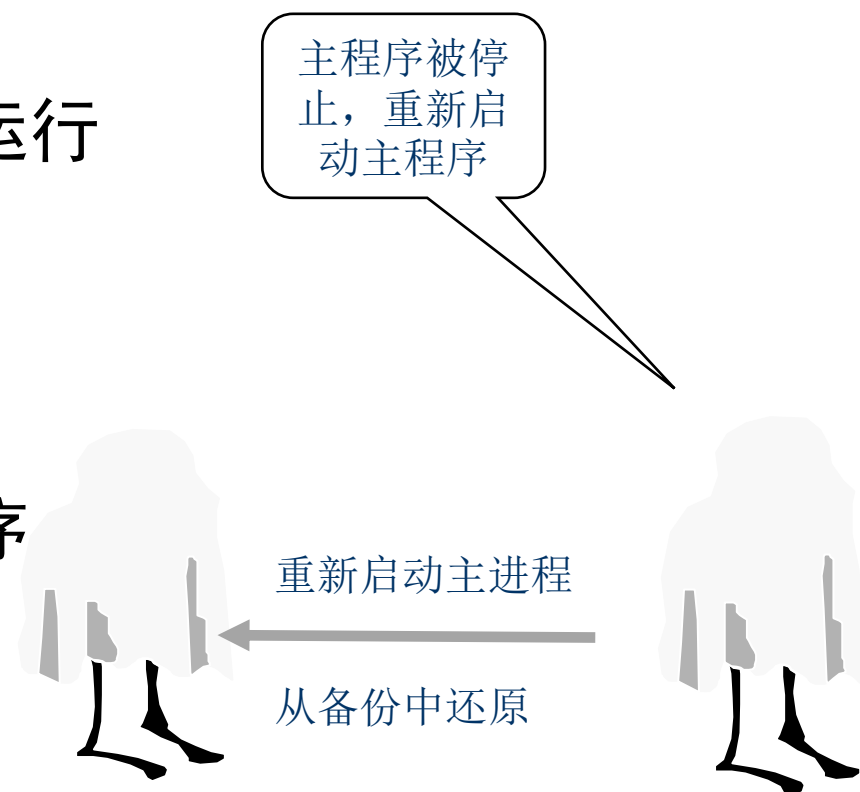


❖ 恶意代码主程序

- 实现恶意代码主功能

❖ 守护程序

- 监视并保护主进程正常运行
- 阻止主程序的退出
- 重启主程序
- 从备份中还原主程序
- 从网络中重新下载主程序



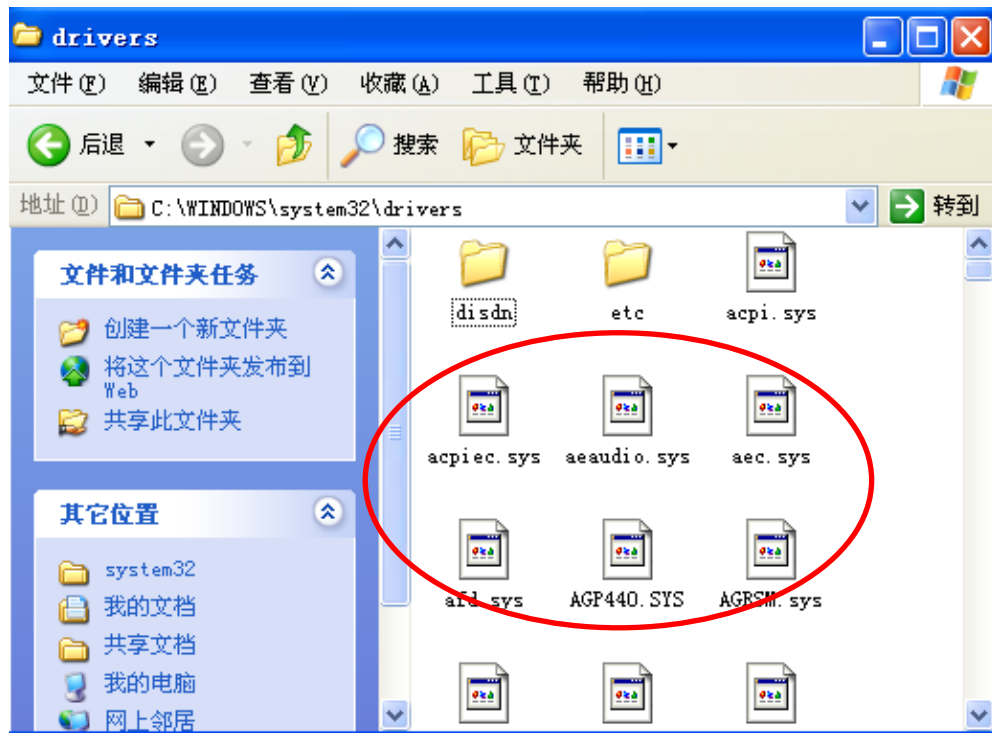


❖ 什么是超级权限技术

- 恶意代码通过将自身注册成为设备驱动，从而获得较高权限，阻止反病毒软件对它的查杀并干扰反恶意代码软件的正常运行

❖ 超级权限技术特点

- 非常高的权限
- 安全模式下可工作
- 无法直接查杀
-





❖ 为什么需要反跟踪

- 反跟踪技术可以提高自身的伪装能力和防破译能力，增加检测与清除恶意代码的难度。

❖ 常用的反跟踪技术有两类

- 反动态跟踪技术
- 反静态分析技术。

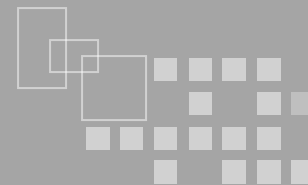


恶意代码反调试技术-反动态调试

- ❖ 通过禁止跟踪中断、封锁键盘输入和屏幕显示等方法，防止调试工具分析恶意代码
- ❖ 主要包括：
 - 禁止跟踪中断
 - 封锁键盘输入和屏幕显示
 - 检查运行环境，破坏调试工具运行

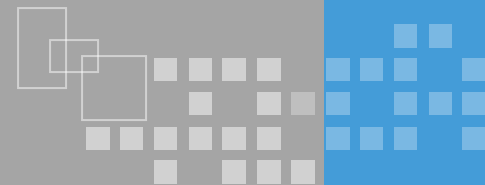


- ❖ 通过加壳、加密、变形以及代码混淆等技术，加大恶意代码自身的复杂性，增加调试分析的难度
- ❖ 主要方法
 - 加壳：对恶意代码的可执行二进制程序进行压缩，使其执行流程发生变化
 - 加密：随着加密密钥的变化，恶意代码会产生不同的表现形式，进一步提高了其抗静态分析的能力
 - 代码混淆：通过插入伪指令、混淆程序数据和控制流等方法，防止静态分析和检测



❖ 知识子域：恶意代码的防御技术

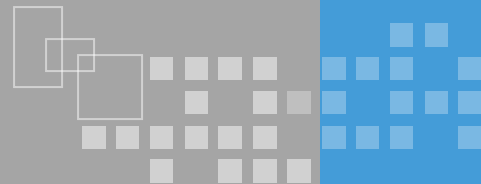
- 理解恶意代码预防方法：减少漏洞、减轻威胁等
- 理解恶意代码特征码扫描、行为检测等检测方法
- 理解恶意代码静态与动态分析方法
- 理解不同类型恶意代码的清除方法



- ❖ 增强安全策略与意识
- ❖ 减少漏洞
 - 补丁管理
 - 主机加固
- ❖ 减轻威胁
 - 防病毒软件
 - 间谍软件检测和删除工具
 - 入侵检测/入侵防御系统
 - 防火墙
 - 路由器、应用安全设置等



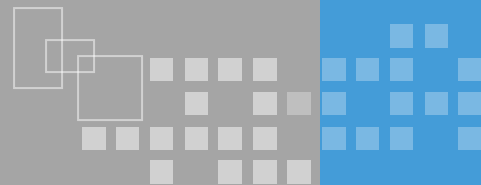
恶意代码检测技术



- ❖ 特征码扫描
- ❖ 校验和
- ❖ 行为监测



特征码扫描



❖ 工作机制：特征匹配

- 病毒库（恶意代码特征库）
- 扫描（特征匹配过程）

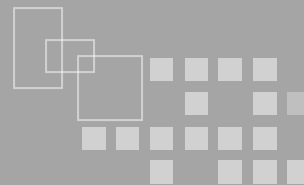
❖ 优势

- 准确（误报率低）
- 易于管理

❖ 不足

- 效率问题（特征库不断庞大、依赖厂商）
- 滞后（先有病毒后有特征库，需要更新特征库）
-



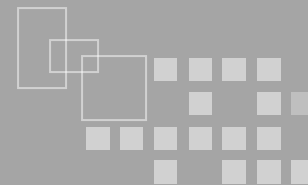


❖ 工作机制

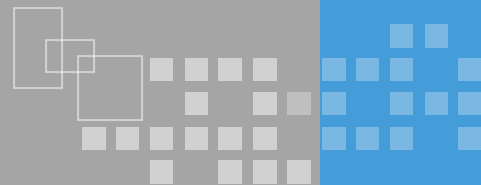
- 将恶意代码放入虚拟机中执行，其执行的所有操作都被虚拟化重定向，不改变实际操作系统优势

❖ 优点

- 能较好的解决变形代码的检测



- ❖ 工作机制：基于统计数据
 - 恶意代码行为有哪些
 - 行为符合度
- ❖ 优势
 - 能检测到未知病毒
- ❖ 不足
 - 误报率高
 - 难点：病毒不可判定原则



❖ 静态分析

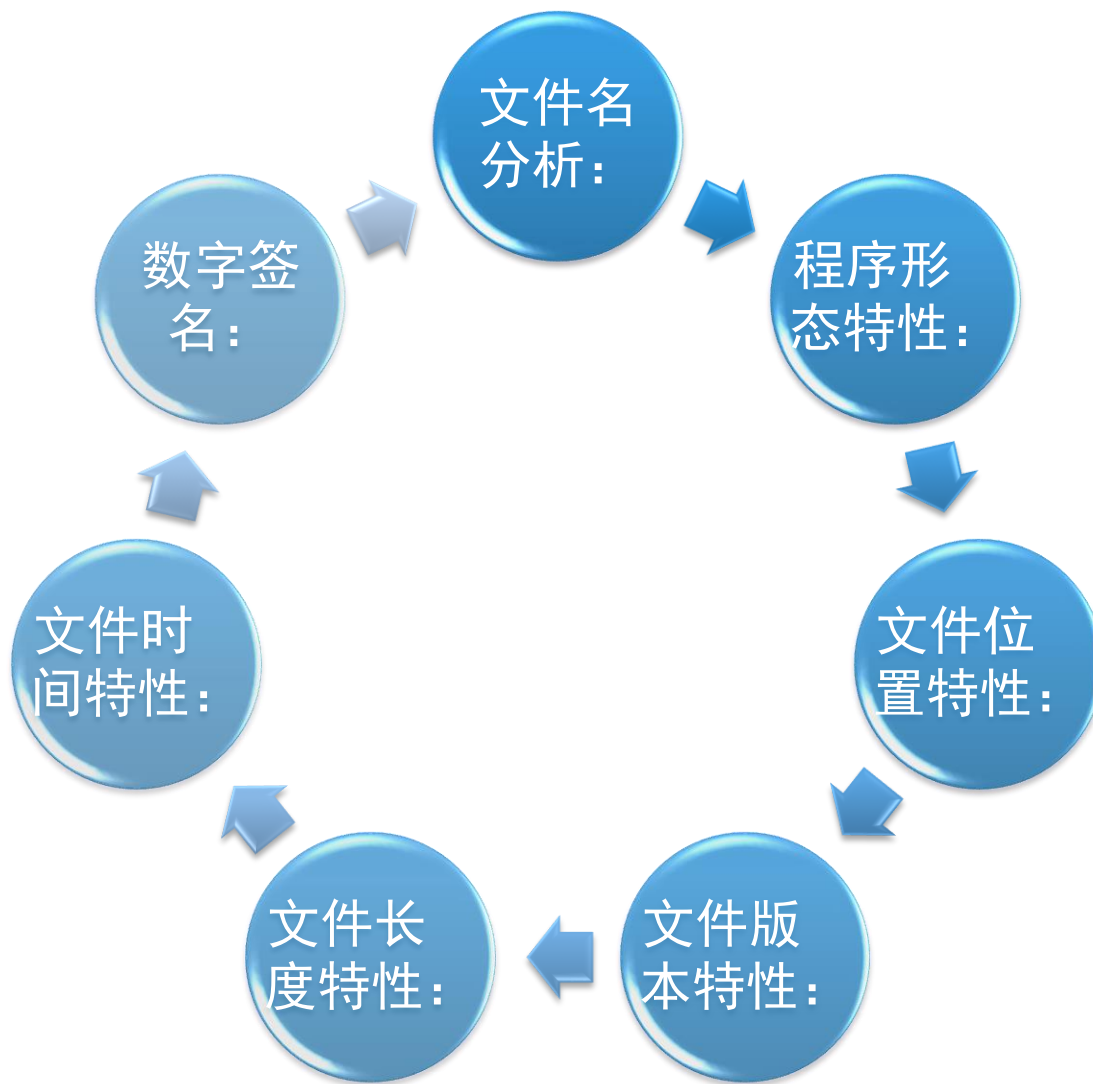
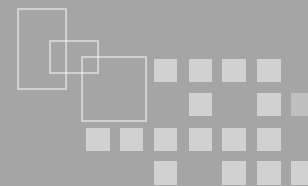
- 需要实际执行恶意代码，它通过对其二进制文件的分析，获得恶意代码的基本结构和特征，了解其工作方式和机制

❖ 动态分析

- 在虚拟运行环境中，使用测试及监控软件，检测恶意代码行为，分析其执行流程及处理数据的状态，从而判断恶意代码的性质，并掌握其行为特点。



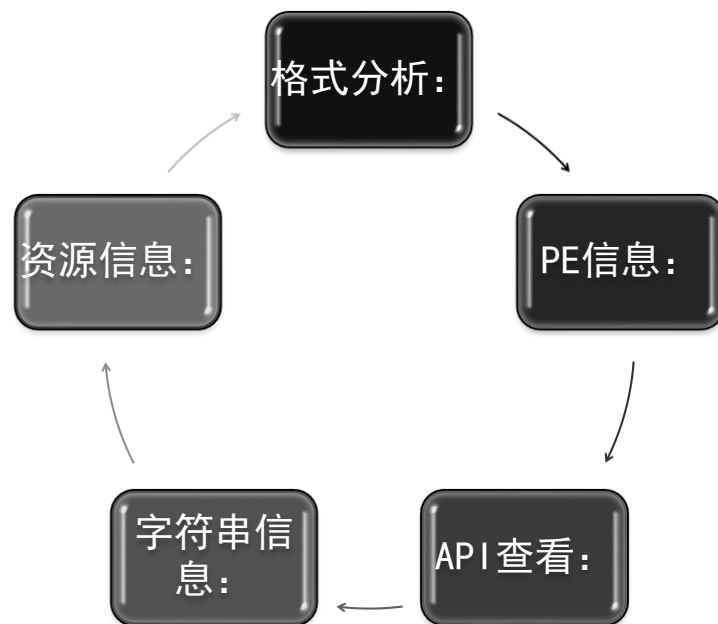
静态分析-常规分析





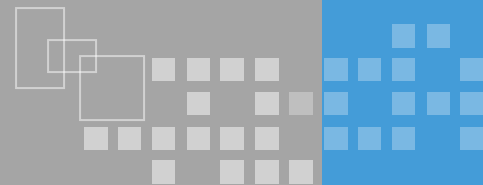
静态分析-代码分析

- ❖ 样本文件格式：PE文件，脚本文件等。
- ❖ PE信息分析（是否加壳、加壳类型等）
- ❖ API调用
- ❖ 附加数据分析（字符串、资源）





静态分析的特点



❖ 优点

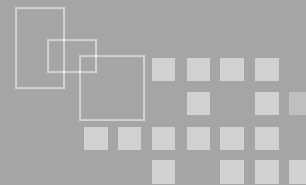
- 不需要运行恶意代码，不会影响运行环境的安全
- 可以分析恶意代码的所有执行路径

❖ 不足

- 随着复杂度的提高，执行路径数量庞大，冗余路径增多，分析效率较低



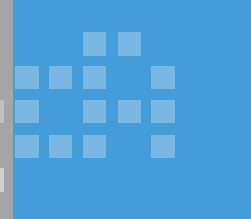
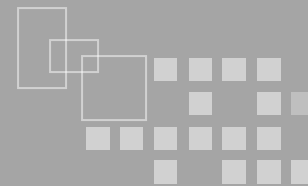
恶意代码动态分析-程序功能



- ❖ API 使用
- ❖ 文件读写
 - 新增?
 - 删除?
 - 改动?
- ❖ 注册表读写
 - 新增?
 - 删除?
 - 改动?
- ❖ 内核调用




恶意代码动态分析-行为分析



行为分析

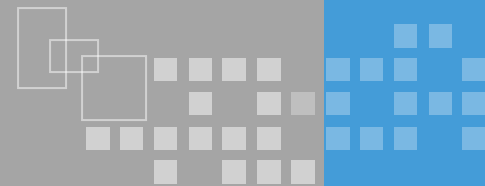
- 本地行为
- 网络行为
- 传播方式
- 运行位置
- 感染方式
- 其它结果等

A large, grey, multi-pointed starburst graphic with a black outline, containing text.

为恶意代
码查杀防
御提供支
撑！



动态分析的特点

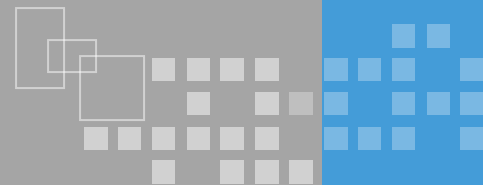


❖ 优点

- 针对性强
- 具有较高的准确性

❖ 不足

- 由于分析过程中覆盖的执行路径有限，分析的完整性难以保证



❖ 感染引导区型

- 修复/重建引导区

❖ 文件感染型

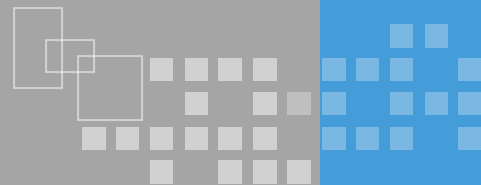
- 附着型：病毒行为逆向还原
- 替换型：备份还原

❖ 独立型

- 独立可执行程序：终止进程、删除
- 独立依附型：内存退出、删除

❖ 嵌入型

- 更新软件或系统
- 重置系统

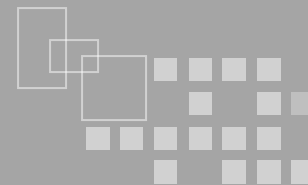


❖ 恶意代码检测与预警体系

- 蜜罐、蜜网

❖ 恶意代码云查杀

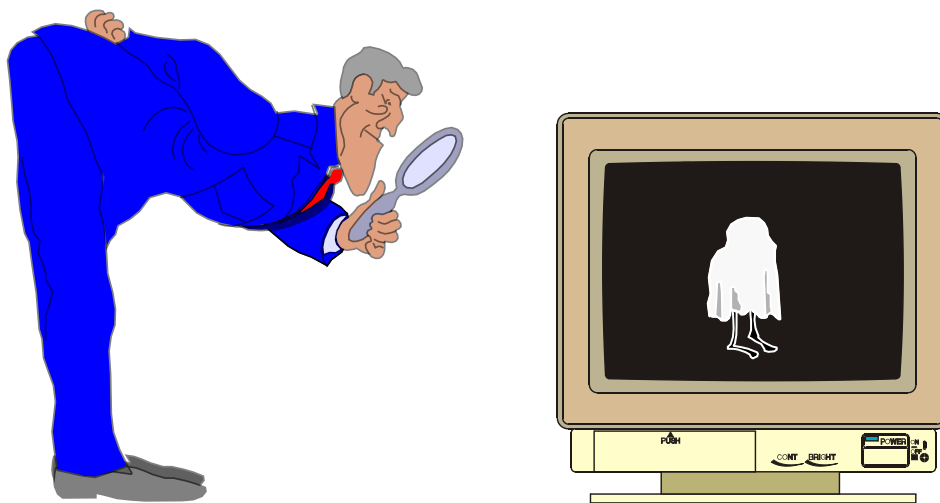
- 分布式计算



- ❖ 恶意代码具有相当的复杂性和行为不确定性，基于恶意代码的防范需要多种技术综合应用，包括：
 - 恶意代码监测与预警
 - 恶意代码传播抑制
 - 恶意代码处置（清除、阻断）



- ❖ 什么是Honeypot（蜜罐）技术
- ❖ Honeypot的作用
 - 网络上的“捕鼠器”，用于研究网络黑客攻击
 - 广泛被用于捕获蠕虫病毒或木马程序样本
- ❖ Honeypot与honeynet





基于互联网的恶意代码防御-云查杀

- ❖ 可疑软件行为检测
- ❖ 云端自动分析处理
- ❖ 快速分发解决措施



谢谢，请提问题！