

网络安全——

# 恶意代码

北京邮电大学

郑康锋

[zkfbupt@163.com](mailto:zkfbupt@163.com)

# 目录

---

- 恶意代码概念
- 病毒、木马、蠕虫
- 恶意代码技术（启动、隐藏、传播）
- 恶意代码分析

恶意代码——

# 恶意代码概念

# 恶意代码概念

---

- 恶意代码是指一切旨在破坏计算机或者网络资源的可用性、机密性和完整性的程序。
- 2010年CNCERT发布的《网络安全术语解释》：**恶意代码是指在未经授权的情况下，在信息系统中安装、执行并达到不正当目的的程序。**
- 现在还没有一个针对恶意代码的统一分类标准，软件公司一般分为病毒（Virus）、蠕虫（Worm）、木马（Trojan）、黑客工具（Hacktool）、风险软件（RiskWare）、灰色软件（GrayWare）、垃圾软件（JunkFile）、测试文件（TestFile）等八类。

恶意代码——

病毒

# 计算机病毒的定义

---

- 在生物学中，**病毒是指侵入动植物体等有机生命体中的具有感染性、潜伏性、破坏性的微生物**，而且不同的病毒具有不同的诱发因素。
- “计算机病毒” (Computer Virus) 一词是人们联系到破坏计算机系统的“病原体”具有与生物病毒相似的特征，借用生物学病毒而使用的计算机术语。
- 美国计算机安全专家Frederick Cohen博士是这样定义计算机病毒的：**“病毒程序通过修改其他程序的方法将自己的精确拷贝或可能演化的形式放入其他程序中，从而感染它们”**。

# 计算机病毒的定义

寄生

- 1994年《中华人民共和国计算机病毒安全保护条例》定义：

“计算机病毒是指编制或者在计算机程序中插入的破坏计算机功能或数据，影响计算机使用并能自我复制的一组计算机指令或者程序代码”。

传染

- 思考：按照上面的定义，网络蠕虫、特洛伊木马是否是计算机病毒？

# 计算机病毒的定义

---

- 广义定义一：计算机病毒是一种人为制造的、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或指令的集合。摘自《计算机病毒与反病毒技术》
- 广义定义二：能够引起计算机故障，破坏计算机数据的程序都统称为计算机病毒。
- 从上面定义可以看出，目前比较流行的网络蠕虫、特洛伊木马都具备广义定义的特征，属于广义定义下的计算机病毒。



# 计算机病毒的特征

---

- 计算机病毒的基本特征主要有以下几个方面：
  - **可执行性**：即程序性，计算机病毒是一段代码（可执行程序）。该特征决定了计算机病毒的可防治性、可清除性；
  - **传染性**：类似于生物病毒，计算机病毒也具有传染性，即病毒具有把自身复制到其它程序的能力；
  - **隐蔽性**：计算机病毒具有隐蔽能力，其传染过程也具有隐蔽能力；
  - **潜伏性**：计算机病毒往往在感染系统后，首先进入潜伏期，等待触发时机；
  - **可触发性**：计算机病毒在触发条件满足时会自动从潜伏期进入活动阶段；如“黑色星期五”会在某月13日的星期五发作。

# 计算机病毒的特征

---

- **破坏性**：计算机病毒大都具有破坏性，如：占用系统资源（内存、硬盘、CPU），破坏数据等。如CIH不仅破坏硬盘的引导区和分区表，而且破坏计算机系统flashBIOS芯片中的系统程序，导致主板损坏，是发现的首例直接破坏计算机系统硬件的病毒；
- **寄生性**：计算机病毒嵌入到宿主程序中，依赖于宿主程序而生存；
- **衍生性**：计算机病毒可以衍生出与原版本不同的变种；
- **诱骗性**：有些病毒通过一些特殊的方式，诱骗用户触发、激活病毒。如：VBS.LoveLetter（别名I LOVE YOU，我爱你，爱虫，情书）病毒是一个用VBS编写的蠕虫病毒，该病毒通过电子邮件及IRC（Internet Relay Chat）传播，通过诱惑用户点击进行激活；

# 计算机病毒的生命周期

---

- **1、开发期：**即病毒的编写调试期；
- **2、传染期：**即病毒的发布期，通过各种途径传播病毒；
- **3、潜伏期：**病毒休眠或隐藏；
- **4、发作期：**即表现期，即病毒的表现形式或破坏能力；
- **5、发现期：**病毒被检测、发现、隔离并被通报研究；
- **6、消化期：**反病毒技术人员修改软件来检测和消除病毒；
- **7、消亡期：**由于用户的防护，病毒难以生存和传播；

# 计算机病毒的发展简史

---

**1949年**，计算机之父冯·诺依曼在《复杂自动机组织论》中提出“一部事实上足够复杂的机器能够复制自身”。

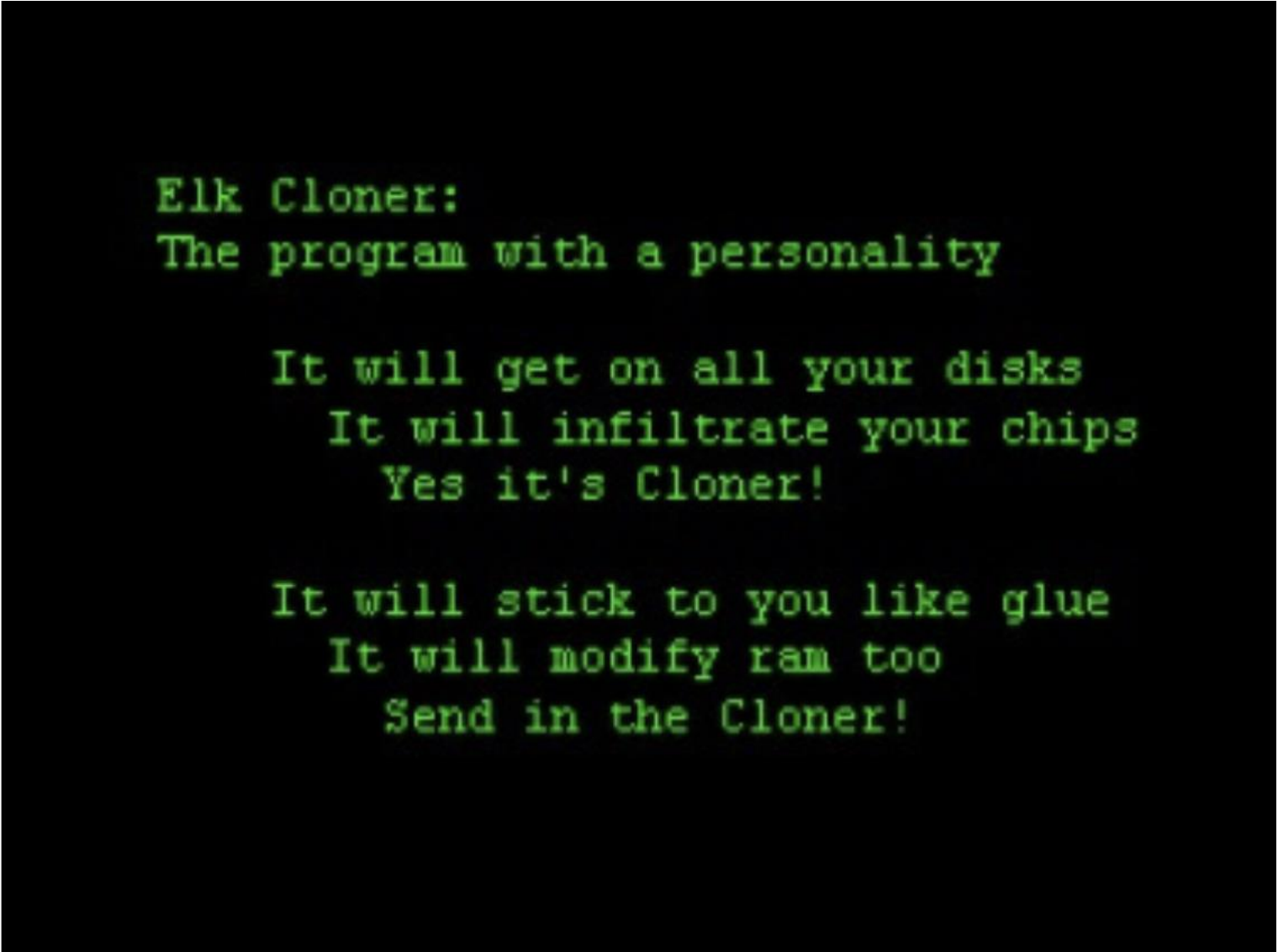
**20世纪60年代初**，美国贝尔实验室里“**磁芯大战**”(core war)的游戏。

**1975年**，《Shock Wave Rider》(John Bruner)出现了“Virus”一词。

# 计算机病毒的发展简史

---

1981年，世界上诞生了真正意义上的**计算机病毒**—**Elk Cloner**，15岁的**Richard Skrenta**，**Apple II**系统，这个病毒将自己附着在磁盘的引导扇区上，通过磁盘进行感染。



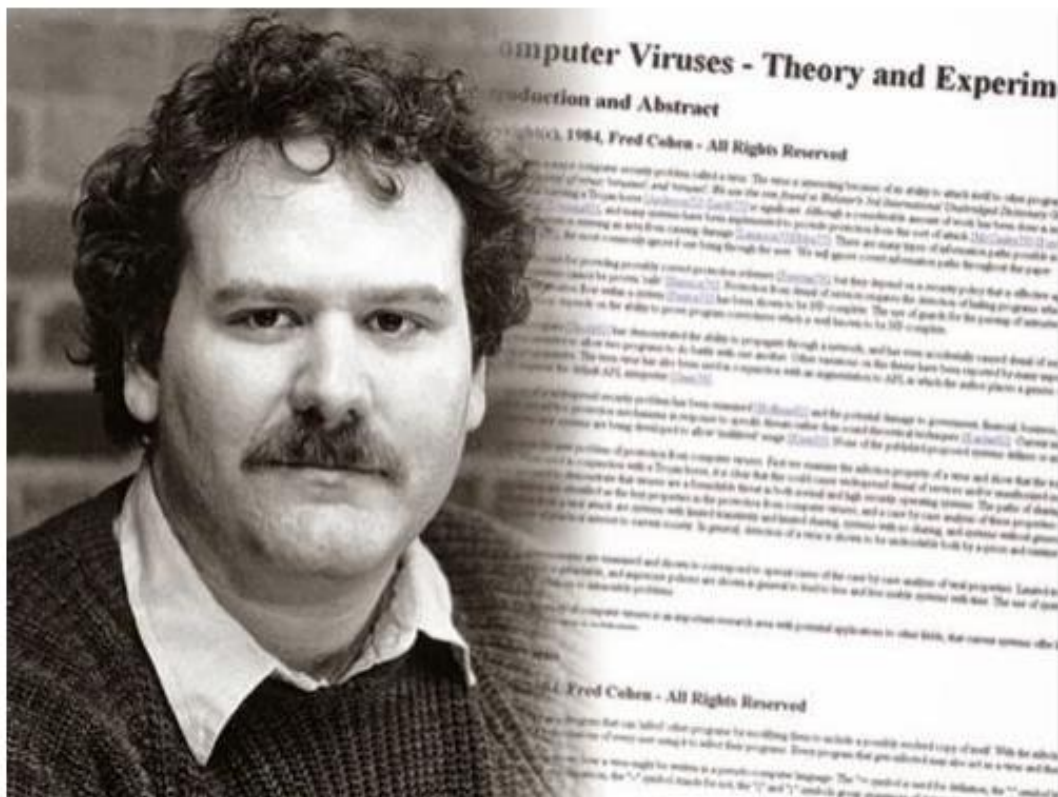
```
Elk Cloner:  
The program with a personality  
  
It will get on all your disks  
It will infiltrate your chips  
Yes it's Cloner!  
  
It will stick to you like glue  
It will modify ram too  
Send in the Cloner!
```

# 计算机病毒的发展简史

1983年11月3日，**Frederick Cohen**博士**首次提出计算机病毒的概念**。

同一天，专家们在**VAX11/750**计算机系统中验证了计算机病毒的存在。

在其后的一周内，在**5次病毒试验**中，平均**30分钟**病毒就可使计算机系统瘫痪。





# 计算机病毒的发展简史

---

1986年底，病毒**Brain**开始流行。Brain病毒首次使用了伪装的手段来迷惑计算机用户。1987年10月，美国新闻机构报道了这一事件。

在这一年，中国的公安部成立了计算机病毒研究小组，并派出专业技术人员到中科院计算所和美国、欧洲进修、学习计算机安全技术，标志着计算机病毒引起了中国政府的警惕。

1987年，DOS环境下的文件型病毒得到了很大的发展。出现了能自我加解密的病毒——Cascade，Stoned病毒和PingPong病毒等等。

# 计算机病毒的发展简史

1987年12月，第一个广泛破坏的计算机蠕虫**Christmas Tree EXEC**开始流行，瘫痪了数个国际计算机网络。

程序运行时，它将显示文本图形，然后将其自身转发到用户地址文件中包含的邮箱地址，传播到了欧洲学术研究网络（EARN），BITNET和IBM的全球VNET。

*	
*	
***	
*****	
*****	
*****	
*****	A
*****	
*****	VERY
*****	
*****	HAPPY
*****	
*****	CHRISTMAS
*****	
*****	AND MY
*****	
*****	BEST WISHES
*****	
*****	FOR THE NEXT
*****	
*****	YEAR
*****	



# 计算机病毒的发展简史

---

## Morris蠕虫

- 起因:

康奈尔大学研究生罗伯特·泰潘·莫里斯（Robert Tappan Morris），本意是作为一套试验程序，想测量互联网的规模。

- 机理:

莫里斯蠕虫利用了Unix系统中sendmail、Finger、rsh/rexec等程序的已知漏洞以及薄弱密码。

蠕虫代码使同一台计算机会被重复感染，每次感染都会造成计算机运行变慢直至无法使用，导致拒绝服务。蠕虫的主体只能感染DEC的VAX机上运行的BSD 4以及Sun 3系统，而程序中的一段C语言代码会调用程序主体，使其在其它的系统上也能运行。

# 计算机病毒的发展简史

- 后果:

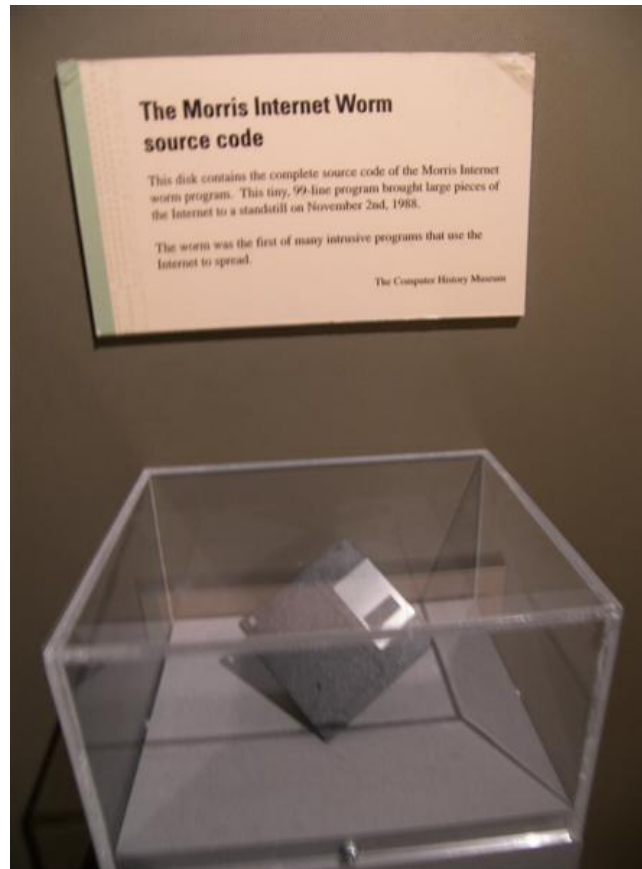
1988年11月2日，在MIT将自己编制的蠕虫程序传入互联网，12小时内超过6200台采用Unix操作系统的SUN工作站和VAX小型机瘫痪或半瘫痪，其中涉及NASA、各主要大学以及未被披露的美国军事基地。

普渡和伯克利大学的研究人员花了72个小时来阻止这种蠕虫。

1990年，莫里斯被判3年缓刑、400小时社区服务及1万美元罚金。



Robert Tappan Morris



波士顿科学博物馆  
保存的  
存有莫里斯蠕虫源  
代码的磁盘

# 计算机病毒的发展简史

---

## 反病毒技术发展

所罗门公司的反病毒工具——Doctors Solomon's Anti-Virus Toolkit——成为当时最强大的反病毒软件。

所罗门博士的防病毒工具包是一个防病毒套件，其中包含针对Microsoft Windows（最多98个），Novell，SCO Unix，Sun Solaris和OS / 2的预防，检测和修复功能，由S & S International的Alan Solomon撰写。

所罗门博士的反病毒工具最初是在1988年创建的，并于1991年正式投入商用，1998年6月9日，McAfee（当时的Network Associates）同意以6.42亿美元的股票收购Solomon博士的Group PLC。

1989年，病毒家族开始出现了，比如Yankee病毒，Eddie病毒，Frodo病毒（第一个全秘密寄生的文件病毒）。同年出现了名为AIDS的特洛伊木马型病毒。

# 计算机病毒的发展简史

---

1993年、1994年，采用密码技术、编写技巧高超的**隐蔽型病毒和多态性病毒**相继出现，也出现了感染源代码文件的SrcVir病毒和感染OBJ文件的Shifter病毒。

1995年8月9日，在美国首次发现专门攻击Word文件的**宏病毒**——Concept。

1997年2月，第一个**Linux环境下的病毒**Bliss出现。1997年4月，第一个使用FTP进行传播的Homer病毒出现。

1998年6月，**CIH病毒**被发现。这一年也出现了远程控制工具“Back Orifice”、“Netbus”等，第一个感染Java可执行文件的Strange Brew病毒，用实用VB脚本语言编写的Robbit病毒。

1999年，通过**邮件进行病毒传播**开始成为病毒传播的主要途径，而宏病毒仍然是最流行的病毒。这一年，比较有名的病毒有：Melissa，Happy99；FunLove等等。

# 计算机病毒的发展简史

---

2000年被称作**VBScript病毒/蠕虫之年**。大量使用脚本技术的病毒出现，脚本技术和蠕虫、传统的病毒、木马程序以及操作系统的安全漏洞相结合，给病毒技术带来了一个新的发展高峰。最著名的如VBS/KAK蠕虫，Loveletter病毒。2000年，中国的金山公司发布金山毒霸，金山公司开始进入杀毒软件市场。

2001年7月出现了**Code Red和 Code Red II**，9月出现的Nimda病毒突破了以往病毒的各种传播途径，它们会利用微软服务器漏洞植入后门程序的特洛伊木马，或是通E-mail大肆传播、衍生无数变种的计算机蠕虫，也有可能是通过浏览网页下载病毒，甚至三者兼具，造成了大范围的因特网上的服务器被阻断或访问速度下降，在世界范围内造成了巨大的损失。仅Code Red病毒所造成的经济损失，就远远超过过去6年来任何一年的年度损失。



# 计算机病毒的发展简史

---

## 灰鸽子(2001年)

灰鸽子是一款远程控制软件，有时也被视为一种集多种控制方法于一体的木马病毒。用户计算机不幸感染，一举一动就都在黑客的监控之下，窃取账号、密码、照片、重要文件都轻而易举。灰鸽子还可以连续捕获远程计算机屏幕，还能监控被控计算机上的[摄像头](#)，自动开机并利用摄像头进行录像。截至2006年底，“灰鸽子”木马已经出现了6万多个变种。

## SQL Slammer (2003年)

Slammer，也称蓝宝石病毒，是一款DDoS恶意程序，透过一种全新的传染途径，采取分布式阻断服务攻击感染服务器，它利用 SQL Server 弱点采取阻断服务攻击1434端口并在内存中感染 SQL Slammer，通过被感染的 SQL Server 再大量的散播阻断服务攻击与感染，造成 SQL Server 无法正常作业或宕机，使内部网络拥塞。在补丁和病毒专杀软件出现之前，这种病毒造成10亿美元以上的损失。蓝宝石病毒的传播过程十分迅速。和 Code Red 一样，它只是驻留在被攻击服务器的内存中。

# 计算机病毒的发展简史

---

## 冲击波蠕虫（Worm.Blaster）

2003年的8月11日出现，不断繁殖并感染，在8月13日达到高峰

症状：死机、重启、显示60秒倒计时关机；IE浏览器不能正常地打开链接；不能复制粘贴；有时出现应用程序异常，比如Word异常；网络变慢；任务管理器里出现msblast.exe进程

影响系统：Windows NT、Windows 2000、Windows XP、Windows Server 2003

利用漏洞：DCOM远程过程调用（RPC）出现的缓存溢出漏洞

病毒运行时会不停地利用IP扫描技术寻找网络上系统为Win2K或XP的计算机，找到后就利用DCOM RPC缓冲区漏洞攻击该系统。

# 计算机病毒的发展简史

## 震荡波 (Sasser, 2004年)

德国的17岁Sven Jaschan2004年制造了Sasser和NetSky。Sasser蠕虫利用微软WindowsNT内核平台上的LSASS漏洞，随机的扫描其它网络中计算机的IP端口，然后进行传播并直接引导这些计算机下载病毒文件并执行，因此整个传播和发作过程不需要人为干预。病毒会修改用户的操作系统，不强行关机的话便无法正常关机。

Netsky 病毒通过邮件和网络进行传播。它同样进行邮件地址欺骗，通过22016比特文件附件进行传播。在病毒传播的时候，会同时进行拒绝式服务攻击(DoS)，以此控制网络流量。Sophos的专家认为，Netsky和它的变种曾经感染了互联网上1/4的计算机。

## 熊猫烧香 (06-07年)

熊猫烧香是一种经过多次变种的蠕虫病毒，2006年10月16日由25岁的中国湖北人李俊编写，2007年1月初肆虐网络。在极短时间之内就可以感染几千台计算机，严重时会导致网络瘫痪。反病毒工程师们将它命名为“尼姆亚”。病毒变种使用户计算机中毒后可能会出现蓝屏、频繁重启以及系统硬盘中数据文件被破坏等现象。同时，该病毒的某些变种可以通过局域网进行传播，进而感染局域网内所有计算机系统，最终导致企业局域网瘫痪，无法正常使用，它能感染系统中exe, com, pif, src, html, asp等文件，它还能终止大量的反病毒软件进程并且删除扩展名为gho的备份文件。被感染的用户系统中所有.exe可执行文件全部被改成熊猫举着三根香的模样。



# 计算机病毒的发展简史

---

## 磁碟机病毒(2007年)

这是一个下载者病毒，会关闭一些安全工具和杀毒软件并阻止其运行;对于不能关闭的某些辅助工具会通过发送窗口信息洪水使得相关程序因为消息得不到处理处于假死状态;破坏安全模式，删除一些杀毒软件和实时监控的服务，远程注入到其它进程来启动被结束进程的病毒，病毒会在每个分区下释放 **AUTORUN.INF**来达到自运行。感染除**SYSTEM32**目录外其它目录下的所有可执行文件。并且会感染**RAR**压缩包内的文件。病毒造成的危害及损失10倍于“熊猫烧香”。

## 机器狗病毒(2007年)

机器狗病毒因最初的版本采用电子狗的照片做图标而被网民命名为“机器狗”，该病毒的主要危害是充当病毒木马下载器，与**AV**终结者病毒相似，病毒通过修改注册表，让大多数流行的安全软件失效，然后疯狂下载各种盗号工具或黑客工具，给用户计算机带来严重的威胁。机器狗病毒直接操作磁盘以绕过系统文件完整性的检验，通过感染系统文件(比如**explorer.exe**，**userinit.exe**，**winhlp32.exe**等)达到隐蔽启动;通过还原系统软件导致大量网吧用户感染病毒，无法通过还原来保证系统的安全。

# 计算机病毒的发展简史

---

## 震网(Stuxnet, 2009-2010)

震网是一种Windows平台上针对工业控制系统的计算机蠕虫，它是首个旨在破坏真实世界，而非虚拟世界的计算机病毒。

利用西门子公司控制系统(SIMATIC WinCC/Step7)存在的漏洞感染数据采集与监控系统(SCADA)，向可编程逻辑控制器(PLCs)写入代码并将代码隐藏，严重干扰位于Natanz的铀浓缩工厂中的离心机保护系统。

Stuxnet同时利用微软和西门子公司产品的7个最新漏洞进行攻击（MS08-067、MS10-046、MS10-061、MS10-073、MS10-092等5个针对Windows系统），2个针对西门子SIMATIC WinCC系统。

这是有史以来第一个包含PLC Rootkit的计算机蠕虫，也是已知的第一个以关键工业基础设施为目标的蠕虫。

据报道，该蠕虫病毒可能已感染并破坏了伊朗纳坦兹的核设施，并最终使伊朗的布什尔核电站推迟启动。

# 计算机病毒的发展简史

---

## 火焰病毒 (Flame, 2010-2014)

“火焰”病毒的全名为Worm.Win32.Flame，是一种后门程序和木马病毒，同时又具有蠕虫病毒的特点。以Lua和C++语言写成，代码量约是之前发现的震网病毒 (Stuxnet) 或毒区病毒 (Duqu) 的20倍，“火焰”设计极为复杂，2012年5月被发现。

火焰病毒伪装成微软开发的合法程序，侵入个人电脑、窃取私密数据，它就能在网络、移动设备中进行自我复制。一旦电脑系统被感染，病毒将开始一系列复杂的行动，包括监测网络流量、获取截屏画面、记录音频对话、截获键盘输入等。被感染系统中所有的数据都能通过链接传到病毒指定的服务器，让操控者一目了然。

据卡巴斯基实验室统计，迄今发现感染该病毒的案例已有500多起，其中主要发生在伊朗、以色列和巴勒斯坦。苏丹、叙利亚、黎巴嫩、沙特阿拉伯、埃及和中国（家庭电脑较多）等国也有个别案例。

# 计算机病毒在我国的发展简况

---

- 开始时传播速度和范围没达到西方的规模，时间上滞后。
- 随着计算机网络在中国的普及，计算机病毒在中国的出现逐步与世界“接轨”。
- 中国越来越多地出现了“国产病毒”（“新世纪”、“中国炸弹”、“冰河”、“灰鸽子”、“熊猫烧香”等）

# 计算机病毒的分类

---

## ● 1. 按计算机病毒攻击的机型分类

- 攻击微型机的病毒;
- 攻击小型计算机的病毒;
- 攻击工作站的病毒;
- 攻击中、大型计算机的病毒;

## ● 2. 按传播媒介分类

- 单机病毒：通常以磁盘、优盘、光盘等存储设备为载体;
- 网络病毒：通过网络协议或电子邮件进行传播，如：QQ、FTP、Web等;

# 计算机病毒的分类

---

## ● 3. 按计算机病毒攻击的操作系统分类

- 攻击DOS系统的病毒;1995年后发展缓慢。
- 攻击Windows系统的病毒；1995年以后已经成为病毒的主要攻击对象。
- 攻击攻击UNIX或OS/2系统的病毒；UNIX与Linux应用广泛，也称为病毒攻击的主要对象。
- 攻击Macintosh系统的病毒；
- 其他操作系统：如嵌入式操作系统（主要是智能手机及PDA使用的操作系统）；

**2001年4月，出现了首例Windows和Linux下都能传播的Win32.Winux病毒，主要感染Windows PE和Linux ELF文件。**

# 计算机病毒的分类

---

## ● 4. 按计算机病毒的链接方式分类

- 源码型病毒：攻击用高级语言编写的程序，在编译之前将病毒代码插入源程序中；
- 入侵型病毒：病毒将自身嵌入到已有程序中，把计算机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；
- 外壳型病毒：通常附着在宿主程序的首部或尾部，相当于给宿主程序加了个“外壳”；
- 译码型病毒：隐藏在如Office、HTML等文档中，如常见的宏病毒、脚本病毒；
- 操作系统型病毒：加入或取代部分操作系统功能进行工作，具有很强的破坏性；



# 计算机病毒的分类

---

## ● 5. 按病毒的表现（破坏）情况分类

- 良性病毒：不包含对计算机系统产生直接破坏作用的代码，主要是表现其存在，只是不停的传播并占用资源；包括无危害性病毒和无危险型病毒。
- 恶性病毒：代码中包含损伤和破坏计算机系统的操作、传染或发作时对系统产生直接破坏作用；包括危险型病毒和非常危险型病毒。

## ● 6. 按计算机病毒寄生方式分类

- 引导型病毒：病毒程序占据操作系统引导区；如小球病毒。
- 文件型病毒：主要感染一些可执行文件，如.COM、.EXE，是主流的病毒，如目录病毒、宏病毒。
- 混合型病毒：集引导型和文件型病毒特性于一体。



# 计算机病毒的通用命名规则

---

- **1.按病毒发作的时间命名：**这种命名取决于病毒表现或破坏系统的发作时间，这类病毒的表现或破坏部分一般为定时炸弹，如“黑色星期五”，是因为该病毒在某月的13日恰逢星期五时破坏执行的文件而得名；
- **2.按病毒发作症状命名：**以病毒发作时的表现现象来命名，如“小球”病毒，是因为该病毒病发时在屏幕上出现小球不停地运动而得名；又如“火炬”病毒，是因为该病毒病发时在屏幕上出现五支闪烁的火炬而得名；
- **3.按病毒的字节长度命名：**以病毒传染文件时文件的增加长度或病毒自身代码的长度来命名，如1575、2153、1701、1704、1514、4096等，“黑色星期五”命名为“1813病毒”；

# 计算机病毒的通用命名规则

---

- **4.按病毒自身包含的标志命名：**以病毒中出现的字符串、病毒标识、存放位置或病发表现时病毒自身宣布的名称来命名，如“大麻”病毒中含有Mar\_i junana及Stoned字样，所以人们将该病毒命名为Mari junana(译为“大麻”)和Stoned病毒；又如“Liberty”病毒，是因为该病毒中含有该标识；再如“DiskKiller”病毒，该病毒自称为DiskKiller(磁盘杀手)。CIH病毒命名源自病毒程序的首位是“CIH”。
- **5.按病毒发现地命名：**以病毒首先发现的地点来命名，如“黑色星期五”又称Jurusalem(耶路撒冷)病毒，是因为该病毒首先在Jurusalem发现；又如Vienna(维也纳)病毒是首先在维也纳发现的。

# 计算机病毒命名的国际惯例

---

- 每个反病毒公司的命名规则都不太一样，但大体都是采取一个统一的命名方式来命名的，一般为：

**〈病毒前缀〉. 〈病毒名〉. 〈病毒后缀〉**

- **病毒前缀：**是指一个病毒的种类，表示病毒的操作平台或病毒类型。比如常见的木马病毒的前缀 Trojan ，蠕虫病毒的前缀是 Worm 等等；
- **病毒名：**是指一个病毒的名称及家族特性，是用来差别和标识病毒家族的，如以前有名的CIH病毒的家族名都是统一的“ CIH ”，振荡波蠕虫病毒的家族名是“ Sasser ”。
- **病毒后缀：**是指一个病毒的变种特性，是用来差别具体某个家族病毒的某个变种的。一般都采取英文中的26个字母或数字来表现，如Worm. Sasser. B就是指 振荡波蠕虫病毒的变种B，因此一般称为“振荡波B变种”或者“振荡波变种B”。

# 常见的病毒前缀

---

- **1、系统病毒：**前缀为W2K、PE、WNT、W32、W95等。这些病毒的一般公有的特性是可以感染Windows操作系统的\*.exe 和 \*.dll 文件，并通过这些文件进行流传；
- **2、蠕虫病毒：**前缀为Worm。特性是通过网络或者系统漏洞进行流传，很大部分的蠕虫病毒都有向外发送带毒邮件，阻塞网络的特性。比如冲击波（阻塞网络）、小邮差（发带毒邮件） 等；
- **3、木马病毒、黑客病毒：**前缀为Trojan、Hack 。木马病毒的公有特性是通过网络或者系统漏洞进入用户的系统并隐蔽，然后向外界泄漏用户的信息，而黑客病毒则有一个可视的界面，能对用户的电脑进行远程掌握。如QQ新闻尾巴木马 Trojan.QQ3344；

# 常见的病毒前缀

---

- **4、脚本病毒：**前缀是Script。运用脚本语言编写，通过网页进行的流传的病毒，如红色代码（Script.Redlof）。脚本病毒还会有如下前缀：VBS、JS（表明是何种脚本编写的）；
- **5、宏病毒：**是脚本病毒的一种，由于它的特殊性，因此在这里单独算成一类。宏病毒的前缀是：Macro，第二前缀是：Word、Word97、Excel、Excel97等。凡是只沾染WORD97及以前版本WORD文档的病毒采取Word97做为第二前缀，款式是：Macro.Word97；凡是只沾染WORD97以后版本WORD文档的病毒采取Word做为第二前缀，款式是：Macro.Word；
- **6、后门病毒：**前缀是Backdoor。该类病毒的公有特性是通过网络流传，给系统开后门，给用户电脑带来安全隐患。如54很多朋友遇到过的IRC后门Backdoor.IRCBot。

# 常见的病毒前缀

---

- **7、病毒种植程序病毒：**特性是运行时会从体内释放出一个或几个新的病毒到系统目录下，由释放出来的新病毒产生损坏。如：冰河播种者（Dropper.BingHe2.2C）等。
- **8. 损坏性程序病毒：**前缀是Harm。特性是本身具有好看的图标来诱惑用户点击，当用户点击这类病毒时，病毒便会直接对用户计算机产生损坏。如：款式化C盘（Harm.formatC.f）、杀手命令（Harm.Command.Killer）等。
- **9. 玩笑病毒：**前缀是Joke，也称恶作剧病毒。这类病毒的公有特性是本身具有好看的图标来诱惑用户点击，当用户点击这类病毒时，病毒会做出各种损坏操作来恫吓用户，其实病毒并没有对用户电脑进行任何损坏。如：女鬼（Joke.Girlghost）病毒。



# 常见的病毒前缀

---

- **10. 捆绑机病毒:** Binder。特性是病毒作者会运用特定的捆绑程序将病毒与一些运用程序如QQ、IE捆绑起来，表面上看是一个正常的文件，当用户运行这些捆绑病毒时，会表面上运行这些运用程序，然后隐蔽运行捆绑在一起的病毒，从而给用户造成迫害。  
如：捆绑QQ（Binder.QQPass.QQBin）、系统杀手（Binder.killsys）等。
- 其它前缀：
  - DoS:** 会针对某台主机或者服务器进行DoS攻击；
  - Exploit:** 会主动通过溢出对方或者自己的系统漏洞来流传自身，或者他本身就是一个用于Hacking的溢出工具；
  - HackTool:** 黑客工具，也许本身并不损坏你的机子，但是会被别人加以运用来用你做替身去损坏别人。

# 计算机病毒的结构组成

---

- 计算机病毒之所以具有寄生能力和破坏能力，与病毒程序的结构有关。
- 从目前已出现的病毒来看，病毒都具有相同的逻辑程序结构，一般包含3大模块，即：**引导模块**、**感染模块**和**表现（破坏）模块**。
- 其中后两个模块都包括一段触发条件检查程序段，它们分别检查是否满足触发条件和是否满足表现（破坏）的条件。一旦相应的条件得到满足，病毒就会进行感染和表现（破坏）。



# 计算机病毒的结构组成



引导模块的功能是使病毒程序获得执行并使其后面的两个模块处于激活状态。

感染模块的功能是，在感染条件满足时把病毒感染到所攻击的对象上。

表现（破坏）模块的功能是，在病毒发作条件（表现、破坏条件）满足时，实施对系统的干扰和破坏活动。

**注：**并不是所有的计算机病毒都由这3大功能模块组成，有的病毒可能没有破坏模块，而有的病毒在3个模块之间可能没有明显的界线。

另外病毒一般还都有自己的**潜伏（或隐藏）**技巧。

# 计算机病毒的结构组成

一个用Pascal语言编写的非常简单的良性病毒的程序作为例子

```
Program Virus_command;           {1}
Uses Crt,dos;                     {2}
Var First3:string[3];      user_Command:string;
    {3}
Begin                             {4}
Repeat                             {5}
    Write( 'A>' );                {6}
Readln(User_Command);             {7}
Exec( 'A:\Comm.com' , '/C' +User_Command); {8}
First3:=copy(User_Command, 1,3);  {9}
if (First3= 'dir' ) or (First3= 'ver' ) then {10}
begin                             {11}
    User_Command:= 'Copy A:\Comm.com' + 'B:\Comm.com' ; {12}
    Exec( 'A:\Comm.com' , '/C' +User_Command);
    {13}
    User_Command:= 'Copy A:\Command.com' +B:\Command.com' ; {14}
    Exec( 'A:\Comm.com' , '/C' + User_Command);
    {15}
    Writeln( 'This is virus program' ); {16}
End;
Until False;
```

# 计算机病毒的结构组成

---

- 用已感染此病毒的A盘（系统盘）启动机器后，COMMAND.COM（**COMMAND.COM为病毒文件本身，原COMMAND.COM文件被改名为COMM.COM**）首先被执行，从而引导了寄生的病毒。所以**此病毒的引导机制是利用了DOS系统的引导机制，用一个假的COMMAND.COM来欺骗系统**。第6行将显示A>来欺骗用户，等候用户输入命令User\_Command，第8行正常执行用户命令，第9~10行等候时机，每当输入命令为“dir”或“ver”时，开始繁殖，这是病毒的**感染模块**（包括感染条件判断模块和实施感染模块）。最后显示出“This is virus program”，是病毒的**表现模块**。
- 程序能够给人以平安无事的假象主要是**第8行调用真正的命令解释器COMM.COM来解释执行用户命令**，对真正的COMMAND.COM进行了“截留盗用”。因此，此病毒可以看成是以外壳的形式寄生在原COMMAND.COM现COMM.COM外，是一个外壳式病毒。

# 计算机病毒的传播途径

---

- 通过不可移动的计算机硬件设备进行传播：如硬盘；
- 通过移动存储设备传播：如磁带、软盘、移动硬盘、优盘、光盘等；
- 通过有线网络系统传播：主要包括电子邮件、Web、BBS、Ftp、即时通信等；
- 通过无线通信系统传播：如攻击和利用WAP服务器、网关，蓝牙等；

# 计算机病毒的传播机制

---

- 计算机病毒的**传播机制**，也称为计算机病毒的**感染机制**或**传染机制**，目的是实现病毒自身的复制和隐藏。
- 病毒的传染性是判断一个程序是否为病毒的必要条件。
- 病毒的传播机制相关编码是病毒的重要敏感部分。可以通过修改此部分代码使病毒失效，或针对其传播机制及时切断其传播途径。

# 病毒的感染对象及感染过程

---

- **感染对象**主要有两种：一种是寄生在磁盘引导扇区；另一种是寄生在可执行文件中。另外，宏病毒、脚本病毒是比较特殊的病毒，在用户使用Office或浏览器的时候获取执行权；还有一些蠕虫只寄生在内存中，而不感染引导扇区和文件。
- **感染过程**一般分为三步：
  - 当宿主程序运行时，获取控制权
  - 寻找感染的突破口
  - 将病毒代码放入新的宿主程序

# 感染机理

---

- **引导型病毒传染机理**：利用在开机引导时窃取中断控制权，并在计算机运行过程中监视软盘读写时机，趁机完成对软盘的引导区感染，被感染的软盘又会传染给其它计算机
- **文件型病毒传染机理**：执行被感染的可执行文件后，病毒进驻内存，监视系统运行，并查找可能被感染的文件进行感染
- **混合感染**：既感染引导扇区又感染文件
- **交叉感染**：一个宿主程序上感染多种病毒，这类感染的消毒处理比较麻烦



# 文件型病毒的感染方式

---

- 寄生感染
- 无入口点感染
- 滋生感染
- 链式感染
- OBJ、LIB和源码感染

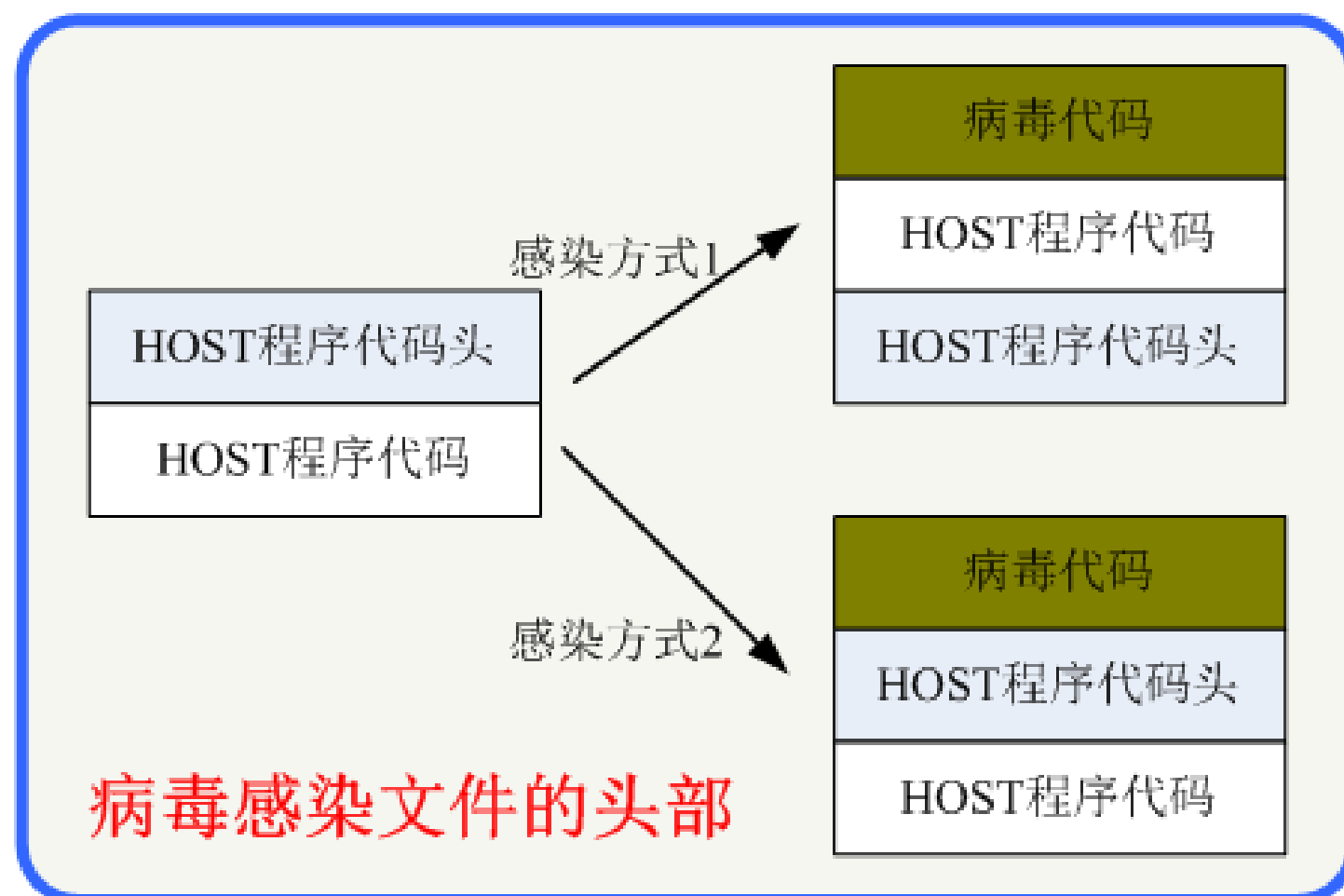
# 寄生感染

---

- **寄生感染是最常见的感染方式。**主要分为两类：
  - **替代法**：病毒程序用自己的部分或全部指令代码，替代磁盘引导扇区或文件中的全部或部分内容；
  - **链接法**：病毒程序将自身代码作为正常程序的一部分与宿主HOST链接在一起，病毒的位置可能在首部、尾部或中间。
  - **寄生在引导扇区的病毒一般采用替代法，寄生在可执行文件中的病毒一般采用链接法。**

# 寄生感染

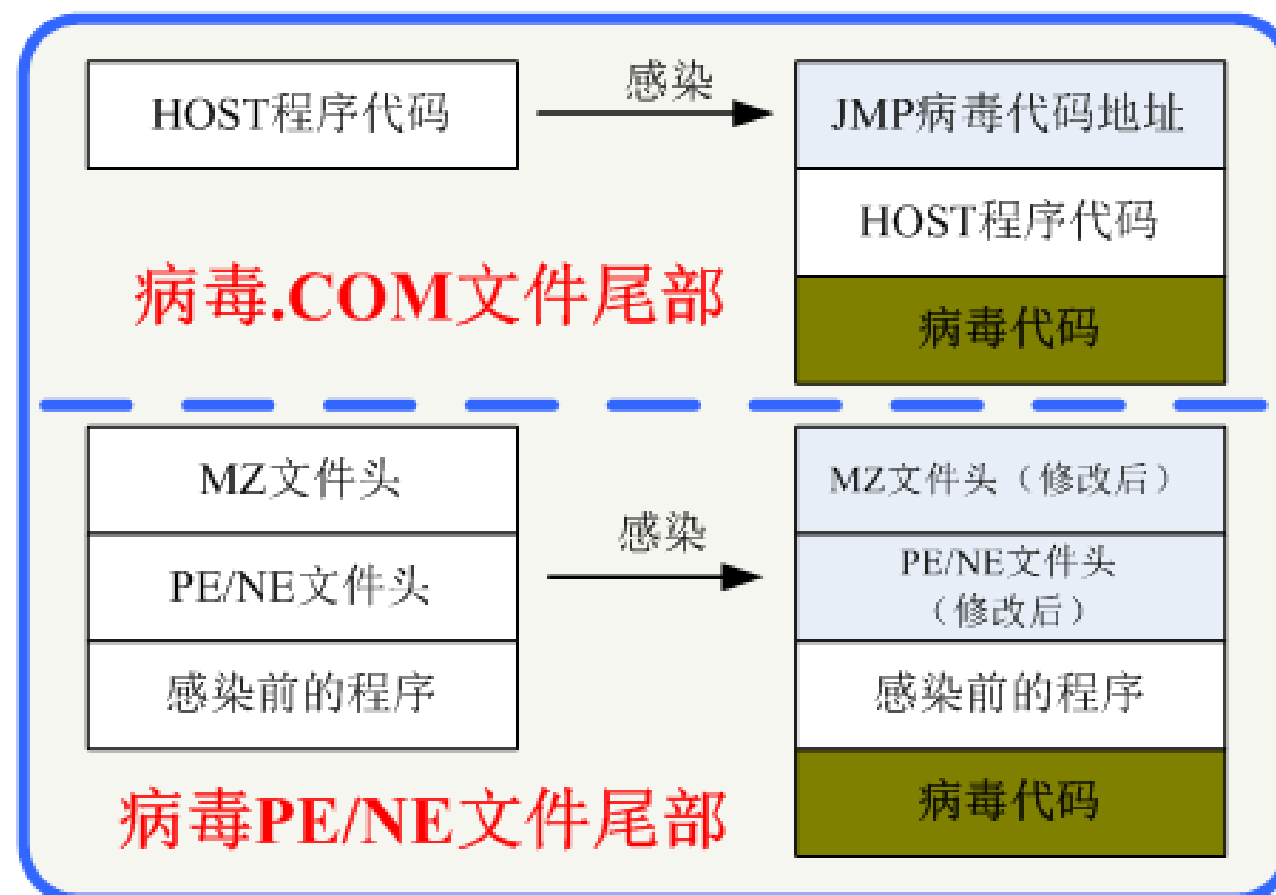
- **文件头部寄生**主要有两种方法：一种是将HOST的文件头拷贝到程序的最后，然后将文件头用病毒代码覆盖；另外一种生成一个新的文件，再用新的文件替换原来的文件从而完成感染。



# 寄生感染

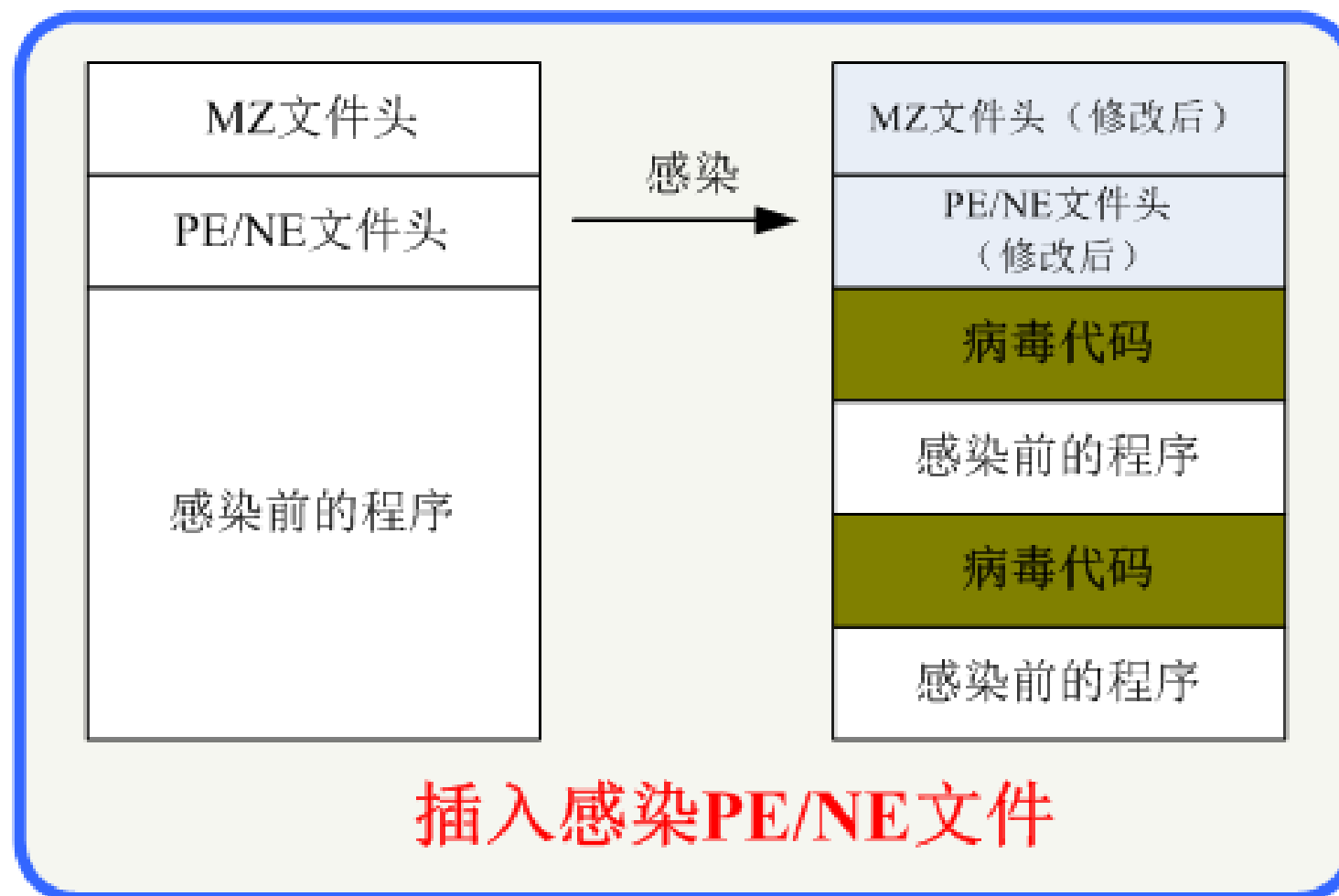
## ● 文件尾部寄生

- 在DOS下.COM可执行文件是简单的二进制代码，没有结构信息，可以将文件头部的3个字节改为“JMP 病毒代码开始地址”；.EXE文件可以转化为.COM感染或修改文件头（改变代码启动地址CS:IP、执行模块长度等）进行感染；
- 在Windows下的.EXE文件，病毒感染后要修改文件头，即PE文件头或NE文件头，需要修改程序入口地址、节的开始地址、节的属性等。



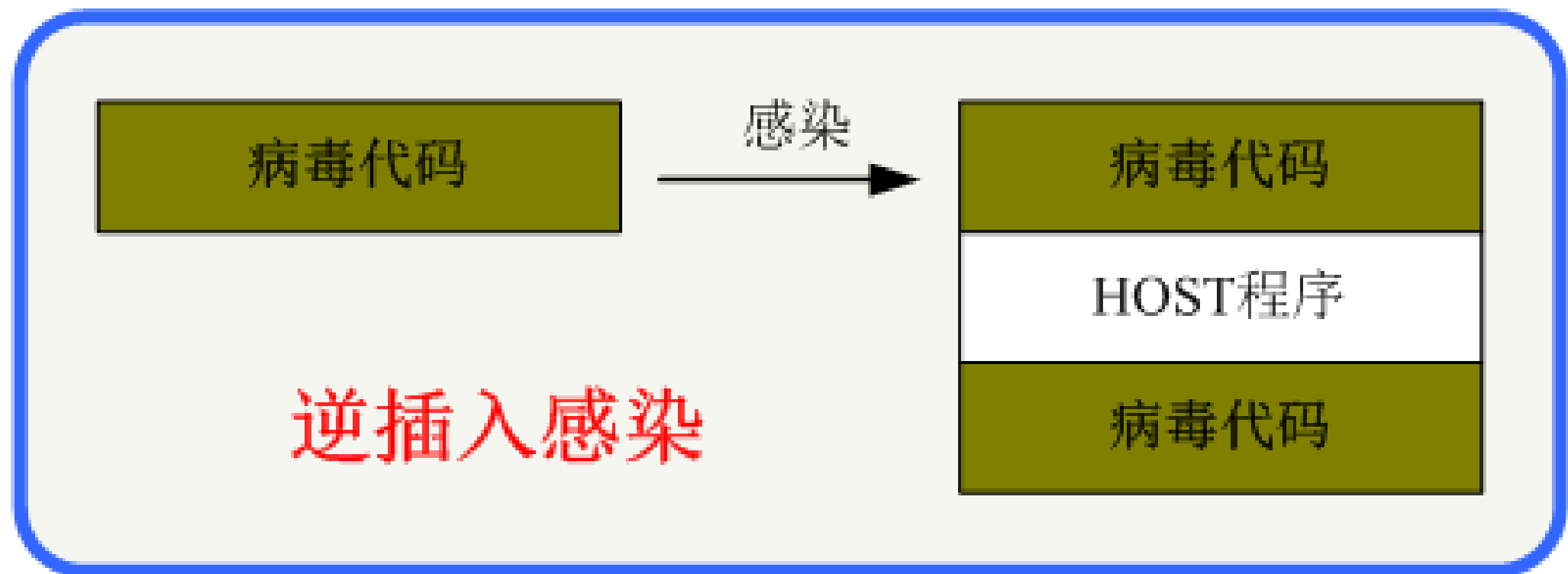
# 寄生感染

- **插入感染**：病毒将自己插入被感染的程序中，可以整段插入，也可以分段。有时需要通过压缩原来的代码方法保持被感染文件的大小不变。



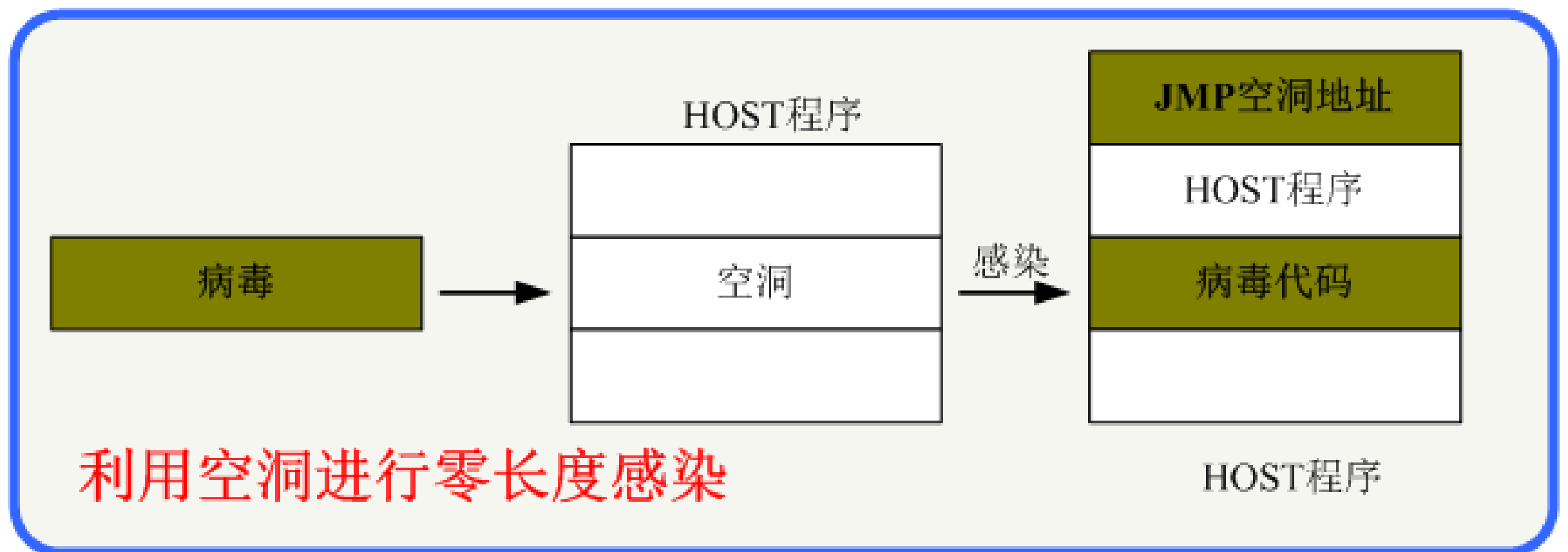
# 寄生感染

- **逆插入感染**：将病毒代码分割为头部和尾部两大块，感染宿主程序时，将宿主程序插入病毒代码中间。



# 寄生感染

- **利用空洞——零长度感染**：病毒查找宿主程序中的空洞（足够长度的全部为零的数据区或堆栈区），并把病毒代码放入空洞中，改变宿主程序开始处代码或入口地址以优先执行病毒代码。





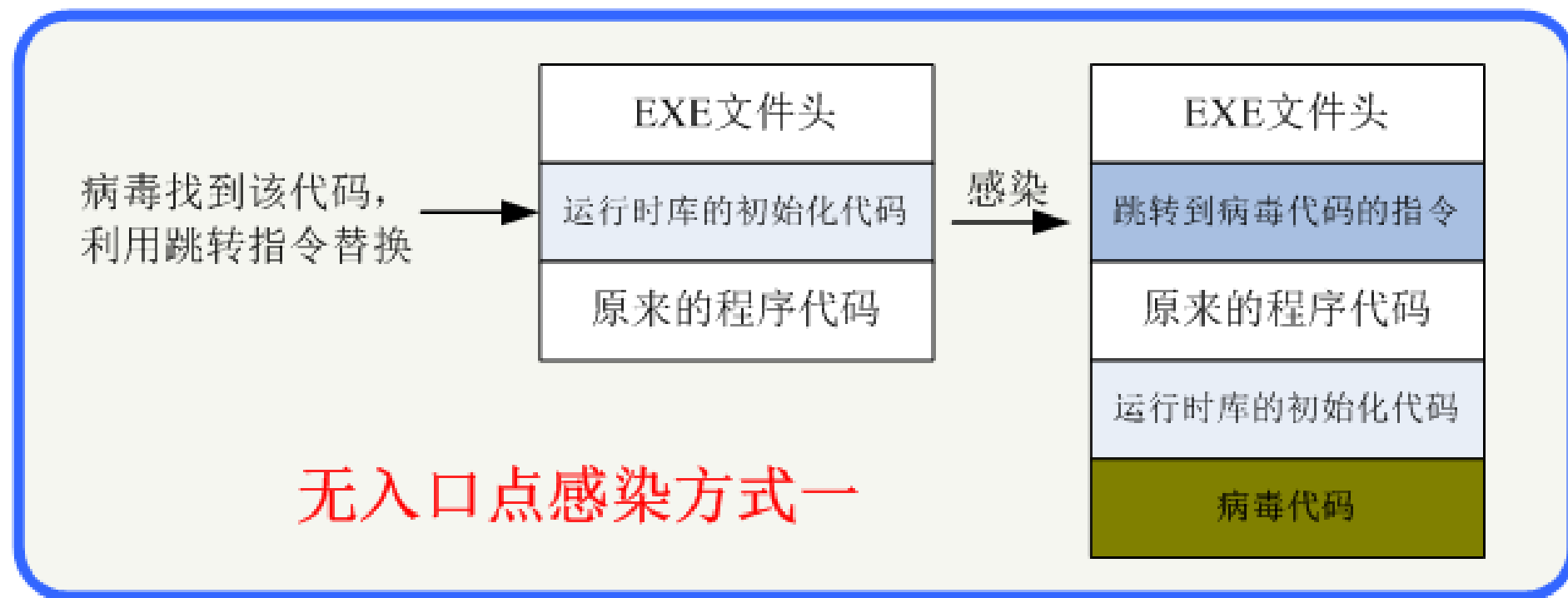
# 无入口点感染

---

- 无入口点感染即入口点模糊技术EPO (Entry Point Obscuring )
  - 在通常的病毒感染方法中，病毒往往会修改PE文件头部的入口 (Entry Point) 字段使其指向病毒代码以达到获得控制权的目的；
  - EPO技术在**不修改宿主原入口点**的前提下，通过在宿主代码体内某处插入跳转指令来使病毒获得控制权；
  - 例： 1) 在.text中寻找E8 xxxxxxxx，根据xxxxxxx获得JMP DWORD文件偏移； 2) 计算跳转到VStart的相对距离，覆盖xxxxxxx； 3) 执行完病毒后再通过跳转指令JMP DWORD PTR [YYYYYYYY]恢复程序的正常功能 。

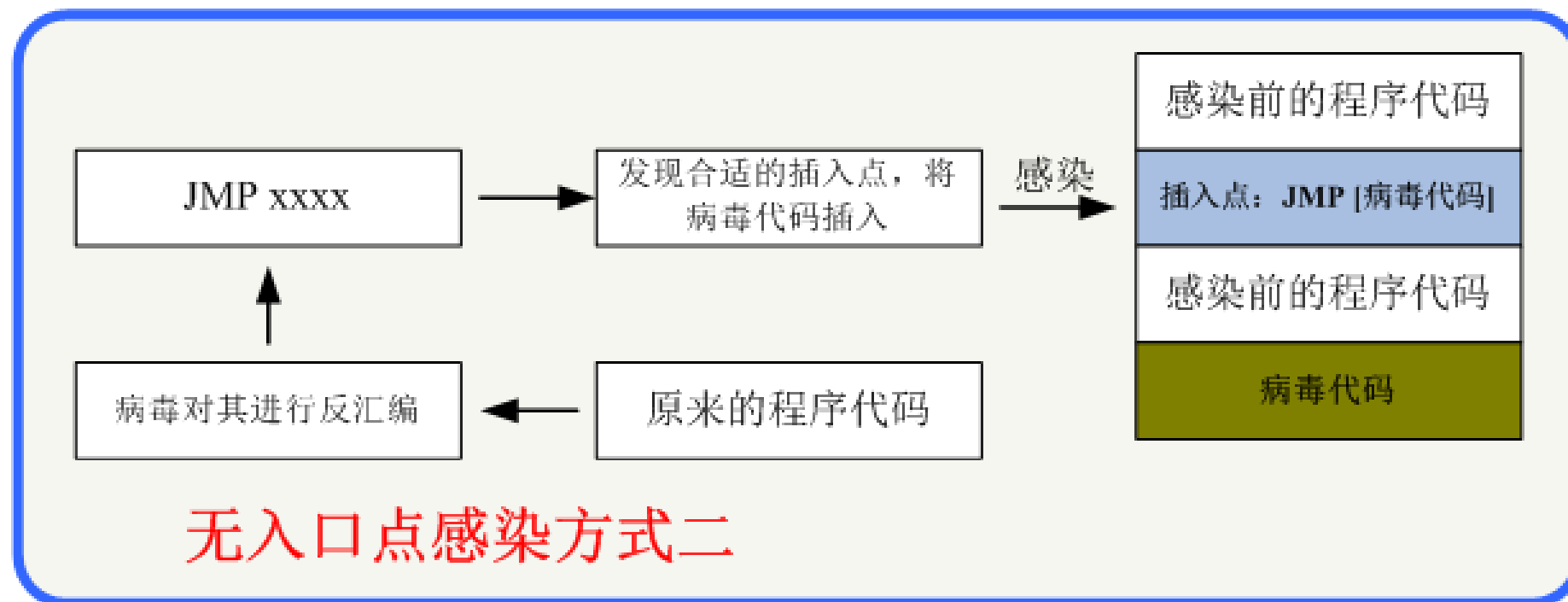
# 无入口点感染

- **方法一**：程序会使用一些基本库函数，编译器会增加一些代码对库进行初始化，病毒可以据此找到程序初始化代码，修改这段代码的起始部分以便跳转到病毒代码处。纽克瑞希尔病毒即采用此类方法。



# 无入口点感染

- **方法二**：病毒的感染部分包含一个小型的反汇编跟踪软件，可以在反汇编中查找符合条件的汇编指令，然后修改该指令使之跳转到病毒程序；“CNTV”和“中间感染”病毒。



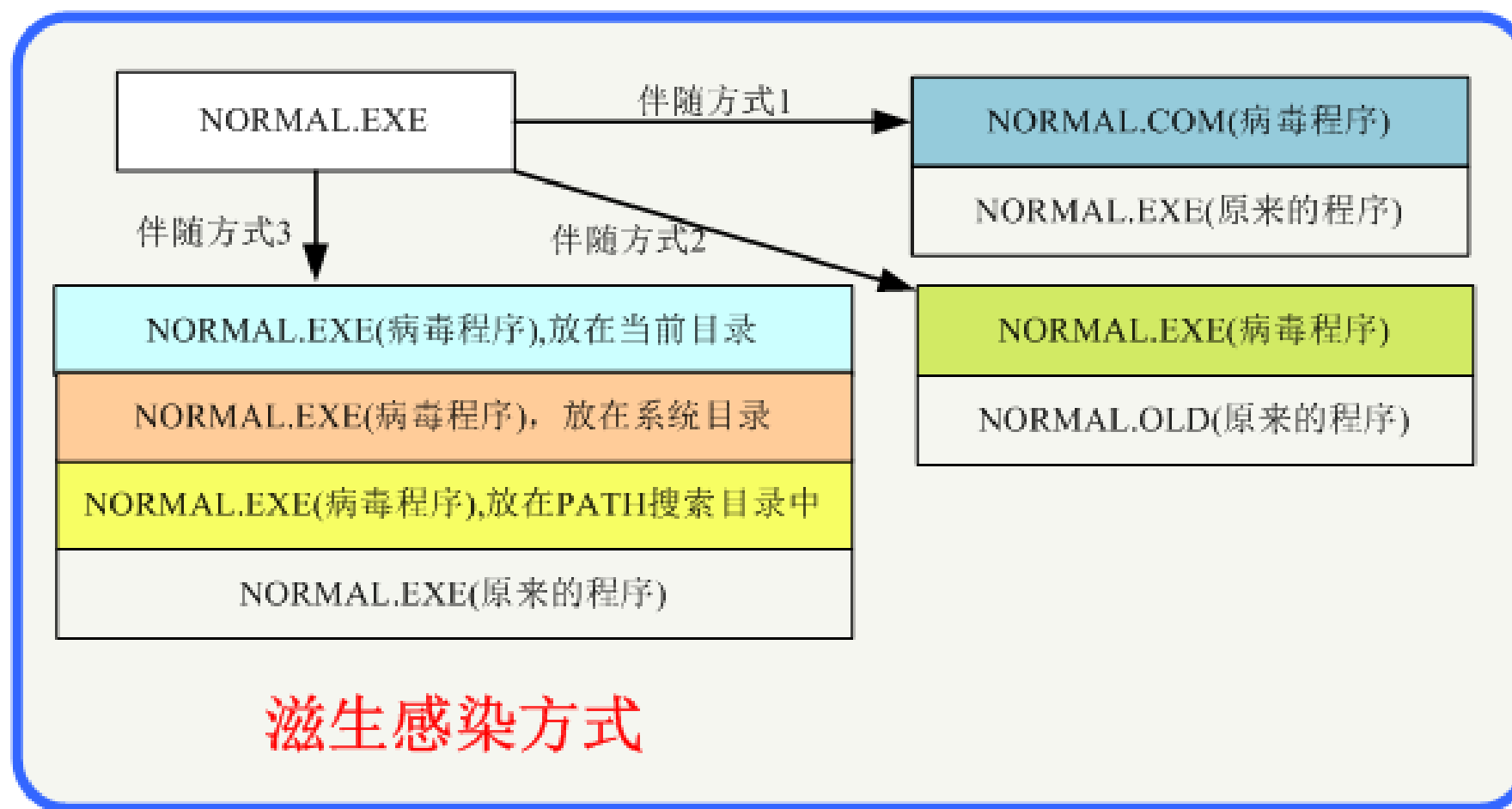
# 无入口点感染

---

- **方法三：**采用TSR病毒技术（Terminal Still Resident 中止仍然驻留），TSR是DOS中的一类重要程序，特点是程序执行完毕后仍然部分驻留在内存中，而且基本上是中断服务程序，病毒可以修改TSR的中断服务代码。如  
“Avata.Position”
- **方法四：**通过.EXE文件的重定位表（指当程序装载到内存时必须固定的地址）获得程序控制权。

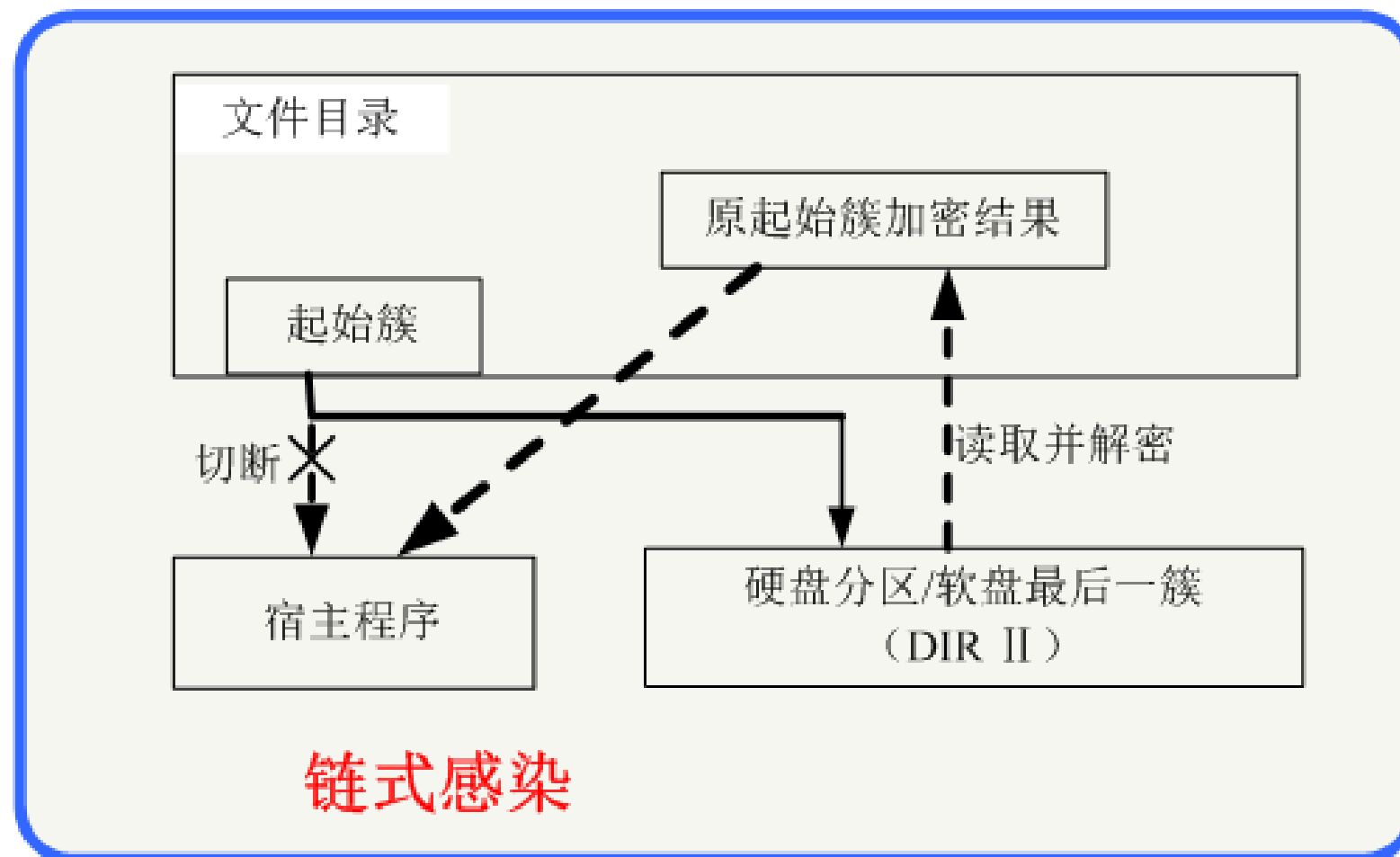
# 滋生感染

- **滋生感染**是一种特殊的感染方式，又称作**伴侣病毒**或**伴随型病毒**，此类病毒不改变被感染文件，而是为被感染文件创建一个伴随文件（病毒），当用户执行被感染文件时会激活病毒。



# 链式感染

- 病毒在感染时完全不改动宿主程序本体，而是改动或利用与宿主程序相关的信息，将病毒程序与宿主程序链成一体，当宿主程序欲运行时，利用相关关系使病毒部分首先运行。



# OBJ、LIB和源码感染

---

- 病毒感染编译器生成的**中间对象文件或库文件**，由于这些文件不是直接的可执行文件，所以病毒感染这些文件后并不能直接传染，必须使用被感染的.OBJ或.LIB链接生成.EXE(.COM)程序之后，才能完成实际的感染过程，所生成的可执行文件中包含了病毒。
- 源码感染则直接对**源代码**进行修改，在源代码文件中增加病毒的内容。

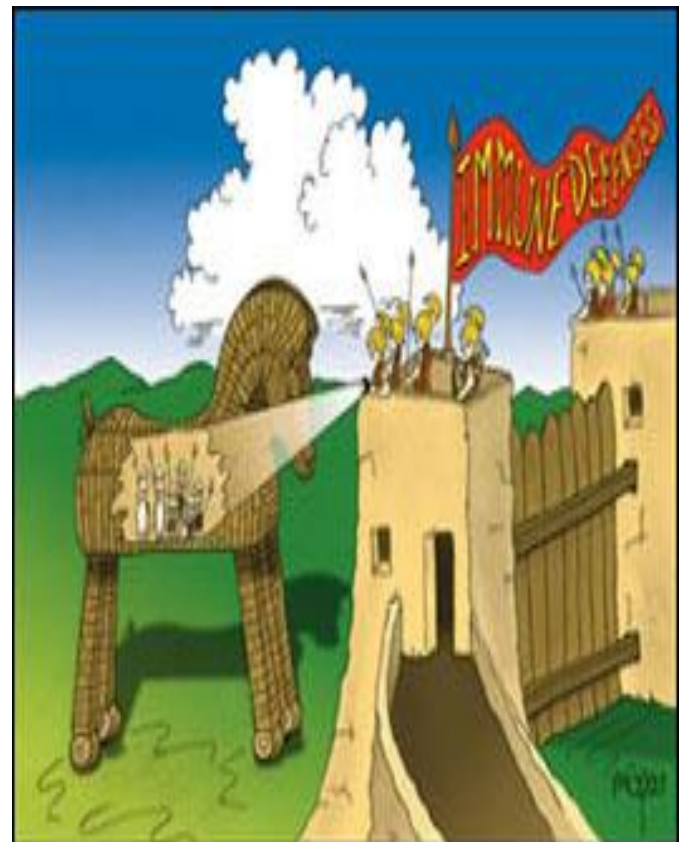
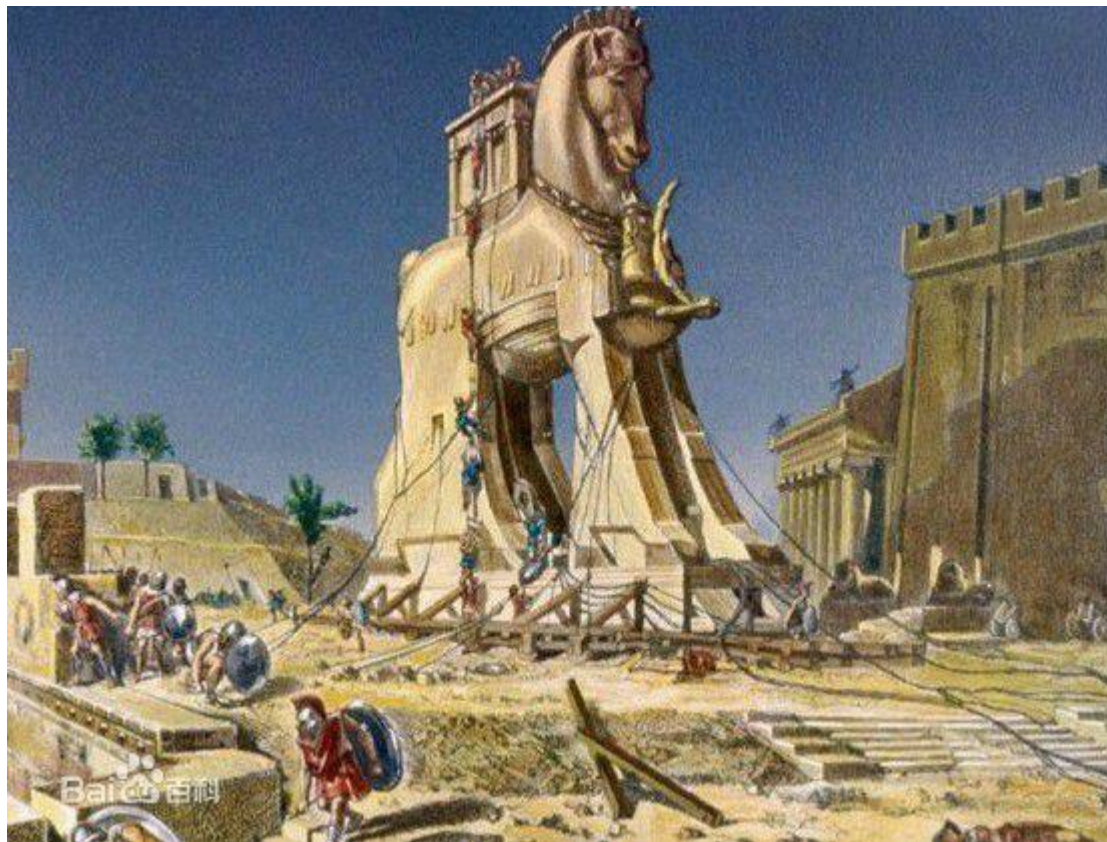


恶意代码——

木马

# 特洛伊木马的来历

希腊人攻打特洛伊城十年，始终未获成功，后来建造了一个大木马，并假装撤退，希腊将士却暗藏于马腹中。特洛伊人以为希腊人已走，就把木马当作是献给雅典娜的礼物搬入城中。晚上，木马中隐藏的希腊将士冲出来打开城门，希腊将士里应外合毁灭了特洛伊城。后来我们把进入敌人内部攻破防线的手段叫做木马计，木马计中使用的里应外合的工具叫做特洛伊木马。



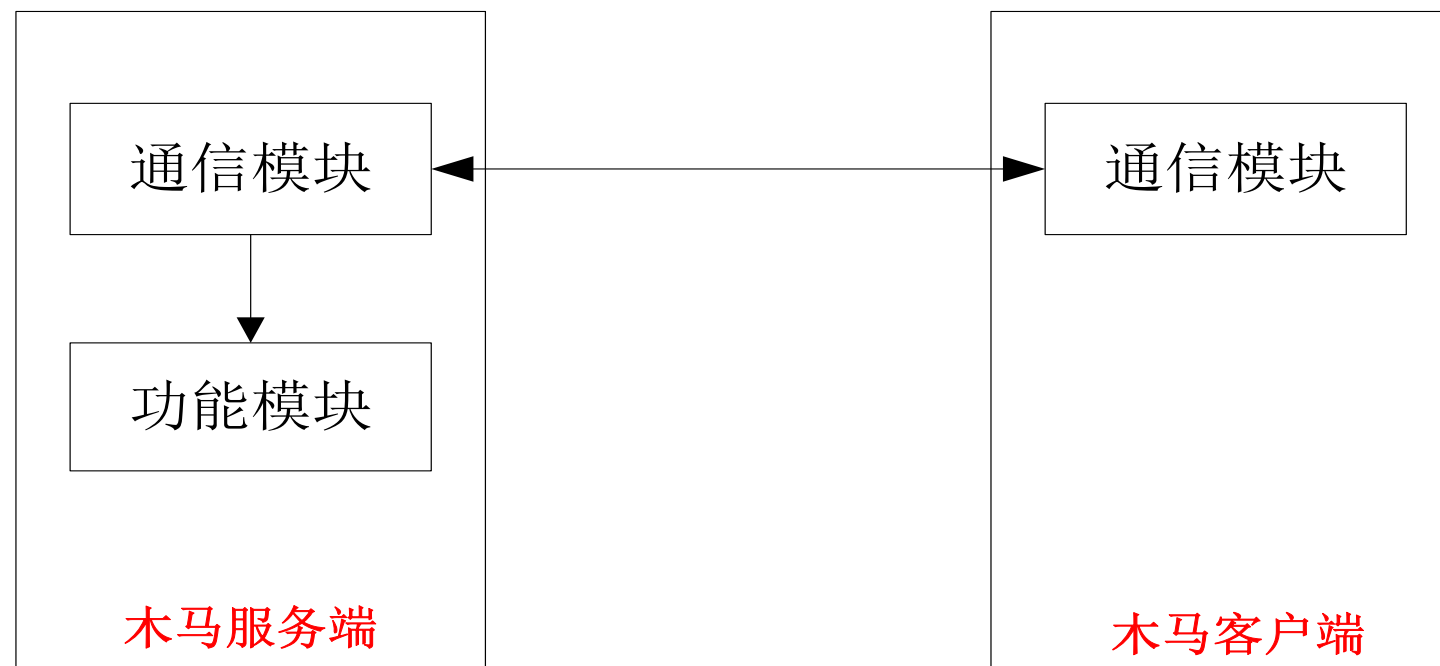
# 木马的定义

---

- 特洛伊木马被喻为“不怀好意的礼物”。我们在计算机领域所讲的木马，在早期的传播中也是伪装成免费赠予的礼物，如好看的屏保、好玩的游戏、有趣的视频等好东西，通过邮件散发或提供网络下载。当用户下载并运行这些木马程序时，木马就进入受害者的电脑中，默默进行着间谍性质的工作，直到用户发现某些文件丢失或帐号被人冒用才会意识到它的存在，与特洛伊战役中的木马使用了相同的战术，这也是木马程序被称为特洛伊木马的原因。
- RFC1244节点安全手册中给出“**特洛伊木马程序是这样一种程序，它提供了一些有用的，或仅仅是有意思的功能。但是通常要做一些用户不希望的事，诸如在你不了解的情况下拷贝文件或窃取你的密码**”，这种对木马程序定义的叙述被人们广泛接受。

# 木马的定义

- 木马程序一般由**客户端和服务端**组成：
  - **服务端**是黑客植入到目标机器的独立的木马模块，等待接受客户端的各种命令操作。
  - **客户端**是控制端，被使用木马的黑客操控，享有服务端各种操作的最大权限。



# 木马的定义

---

- 木马、病毒的区别

人们习惯上把所有的恶意软件统称为病毒，其实木马和病毒还是存在一定区别的。最典型的区别就是：

- 病毒依附于其它文件或程序并自我复制，在运行指定的程序时被激活运行
- 木马通常把自己伪装成一个有用的程序，一般来说自我复制并不是木马功能的一部份。

举个例子，当你打开一个被病毒感染的Word文档时，该病毒就会被激活运行，而木马通常只有运行它本身的程序时才会被激活。木马还有一种运行方式，与其它程序关联，当关联的程序运行时被触发，例如，冰河木马就当你试图打开txt文档时运行，这是由于该木马通过修改注册表将txt文档的打开程序设置为木马程序，txt文档本身并没有被植入木马代码，与病毒的运行方式有明显区别。



# 木马的定义

---

- 木马、病毒的区别

- 计算机病毒具有破坏性和传染性，对受害者的机器进行软硬件破坏、删除文件、长期占用CPU、不能重启机子等各种破坏行为。计算机病毒能够自动寻找其宿主对象，感染机器中的程序文件，依附于其他程序，并可通过网络传染给网络上的其他计算机，是一种具备传染、隐藏、破坏、繁殖等能力的可执行程序，可以说**传染性是病毒最本质的性质之一**。
- 计算机病毒相比，木马主要以偷盗为主，偷取机密文件，记录各种密码，为黑客提供控制受害者机器的后门等。为了能长期隐藏在主机中进行这些操作，木马一般具有很好的隐蔽性，默默地潜伏在受害者机器中进行各种活动。**木马不具有自我复制性**。

# 木马的发展史

---

- 第一代木马功能较为单一，主要以破坏文件和窃取密码为主

特洛伊木马出现于20世纪80年代早期。那时发达国家中的电脑已经比较普及，出现了最早的独立程序员，他们写了很多游戏和小程序，放在BBS上供人们下载。与此同时，一些活跃的程序员出于兴趣编写出大量盗取账户和密码的小程序，来彰显自己的技术水平。他们可以设计出和真实登录界面相同的木马程序来迷惑不知情的用户，从而盗取用户密码。

Pc-Write木马是第一个在计算机界出名的木马，出现在1986年。该木马伪装成Quicksoft公司的用于字处理的共享软件Pc-Write的2.72版本（实际上该公司并未发行2.72版）。当用户信以为真下载并运行该程序时，这个木马会干两件事：一是清除FAT（文件分配表，电脑上用于组织硬件存储内容的系统）；二是格式化硬盘，删除所有存储的数据。该木马具有病毒的特征现象，被称为伪病毒型木马。



# 木马的发展史

---

- 第二代木马功能较为完备，可以实现文件控制、信息窃取、转向攻击后门等多种功能，以B02000、Sbuseven为代表。

冰河是国内著名的木马，由国内安全程序员黄鑫在1999年编写完成，凭借着国产化和暂时无杀毒软件能防范的优势迅速成为当时黑客使用最广泛的木马，是当时为数不多的具有窗口界面的国产木马。冰河木马属于文本关联型程序，其服务器会随文本文件的打开而运行，具有远程修改注册表，点对点及时通信功能，可以对受害者机器上的文件进行复制、修改、删除等非法操作。

# 木马的发展史

---

- 第三代木马

第三代木马在信息传递方面有较大突破，反弹端口、ICMP协议传输等。国内的“灰鸽子”采用反弹端口技术，主动向客户端发起连接，逃避防火墙对外部连接请求的阻碍。“灰鸽子”为入侵者提供后门，最近两年都位于十大流行病毒行列。它具有后门功能，感染后的机器可以被黑客远程控制，如：记录键盘、结束指定的进程、强制重启电脑、执行系统命令、获取系统信息，从网上下载指定的文件等。具有相似功能的还有十大流行病毒的“德芙”木马病毒。

# 木马的发展史

---

- 第四代木马

第四代木马在进程隐藏方面有重大突破。DLL注入型木马经常使用我们常见的一些系统服务进程作为寄主进程，如lsass.exe、svhost.exe、和IEXPLORE.EXE等系统服务进程，这些进程本来就会存在，不会引起怀疑，从而增强其隐蔽性。

# 木马的发展史

---

- 第五代木马

与蠕虫、病毒相结合的第五代木马具有更大的危害性。

2006年的“熊猫烧香”病毒，就是融合了木马、蠕虫、病毒三种入侵手段，利用蠕虫的传播能力和多种渠道传播，在短短几个月内感染了数以万计的电脑，疯狂下载、运行各种木马程序。类似的还有仇英、艾妮等病毒。

恶意代码——

蠕虫

# 网络蠕虫定义

---

- 蠕虫原本是生物学的术语。1982年，Xerox PARC的John F. Shoch等人，为了分布式计算模型的测试，将蠕虫引入到计算机领域，并给出了计算机蠕虫的两个最基本特征：  
“可以从一台计算机移动到另一台计算机”和“可以自我复制”。
- 到1988年Morris蠕虫的爆发，Spaford为了区分蠕虫和病毒，对蠕虫重新进行了定义，他认为“蠕虫可以独立运行，并且自动复制自身并传播到另一台主机。”

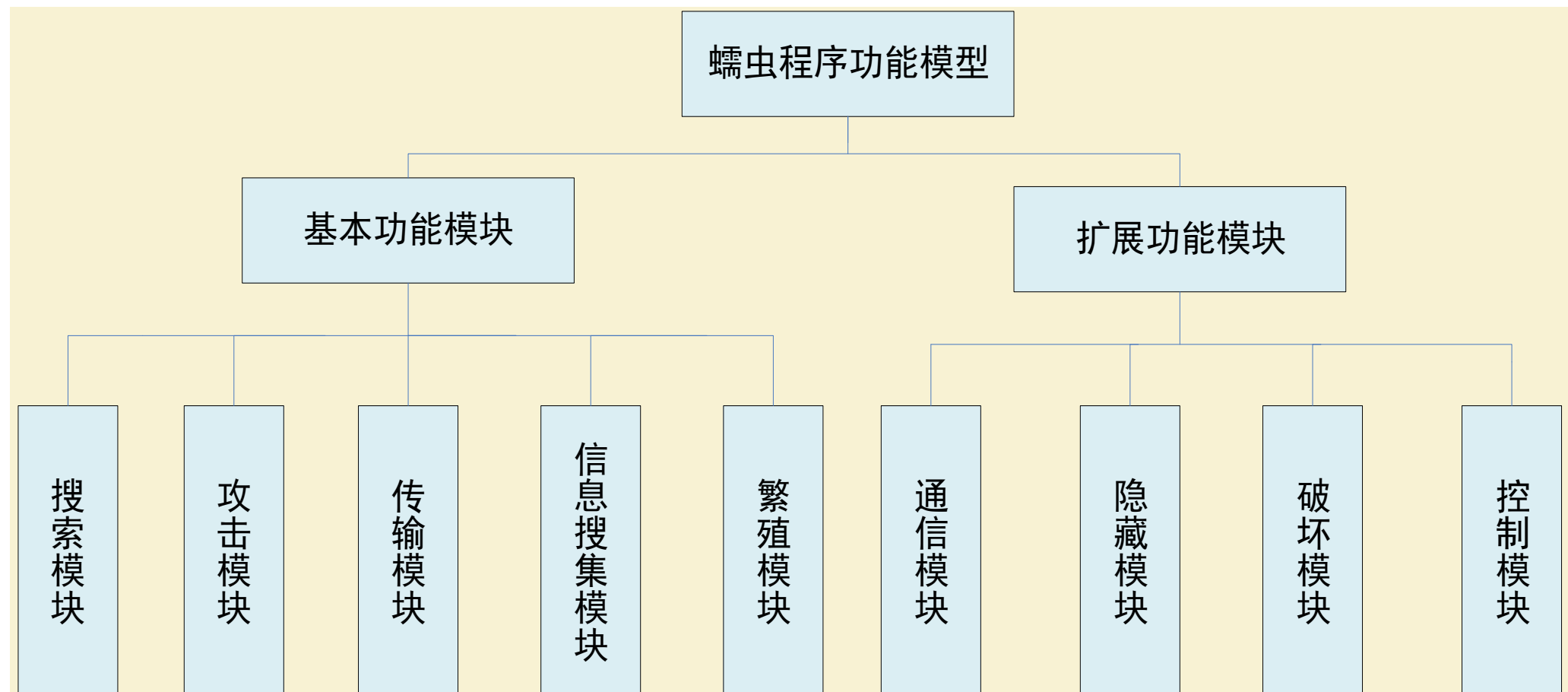
# 病毒、蠕虫区别

---

	普通病毒	蠕虫病毒
存在形式	寄存文件	独立程序
复制形式	插入到宿主程序中	自身的拷贝
传染机制	宿主程序运行	系统存在的漏洞
攻击目标	本地文件	网络上的其他计算机
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	从宿主文件中摘除	为系统打补丁
对抗主体	计算机使用者、反病毒厂商	系统软件和服务软件提供商、网络管理人员

# 网络蠕虫功能结构

- 通过对曾经爆发蠕虫的程序进行分析，得到其功能模型，包括**基本功能模块**和**扩展功能模块**。基本功能模块包含搜索模块、攻击模块、传输模块、信息搜集模块和繁殖模块；扩展模块包含隐藏、通信、破坏等模块。





# 网络蠕虫功能结构

---

- 主体功能模块由**5个模块**构成，主要完成攻击传播的过程。
  - **探索模块**：功能是寻找下一台要感染的机器，可以采用一系列的扫描算法。大多数蠕虫采用随机扫描的方式，不对攻击目标是否存在漏洞进行判断，从而产生大量的无效攻击。
  - **攻击模块**：功能是利用攻击目标上存在的漏洞，从被感染的机器到攻击目标上建立传输通道。
  - **传输模块**：很多蠕虫在获得系统权限后利用tftp系统程序. 完成蠕虫副本在不同机器之间传递的功能，从而有效的减少了自身的大小。

# 网络蠕虫功能结构

---

- **信息搜集模块**: 充分利用本机上, 搜集到的信息来确定新的攻击目标, 搜集的信息可以单独使用或被传送到集中的地点, 信息包括网络信息、本机信息、系统信息、邮件列表等。
- **繁殖模块**: 功能是建立自身的多个副本, 包括实体副本的建立和进程副本的建立, 在同一台机器上主要是提高传染效率, 增加判断避免重复传染。

# 网络蠕虫功能结构

---

- 功能结构模型中的辅助功能既包含对已知蠕虫功能的总结，也包含未来将要出现蠕虫功能的概括，有些功能模块在现有的蠕虫中还未出现，是对蠕虫技术发展的一种预测。
  - **通信模块**: 使蠕虫间、蠕虫同黑客间进行交流，这是未来蠕虫发展的重点，利用通信模块，蠕虫间可以共享某些信息。
  - **隐藏模块**: 包括文件形态的各个实体组成部分的隐藏、变形、加密，以及进程空间的隐藏，包括内核一级的修改工作，该模块可以大大提高蠕虫的生存能力。

# 网络蠕虫功能结构

---

- **破坏模块**: 摧毁或破坏被感染的计算机，破坏网络正常运行，对指定目标做某种方式的攻击，或在被感染计算机上留下后门。
- **控制模块**: 功能是调整蠕虫行为，更新其他模块，控制被感染机器，可能是未来蠕虫发展的重点，执行蠕虫编写者下达的命令。

# 网络蠕虫扫描策略

---

- **网络蠕虫扫描策略**是指网络蠕虫在扩散的过程中，选择下一步潜在感染对象时所采用的方法，扫描策略的好坏与否，直接影响着网络蠕虫传播速度。
- 研究人员对目前出现的蠕虫进行分析得知网络蠕虫的扫描策略主要有：**随机性扫描、顺序扫描、目标列表扫描、路由信息扫描、DNS扫描、分治扫描等**。介绍完扫描策略后，会对各种扫描策略进行评价。

# 网络蠕虫扫描策略

---

- 选择性随机扫描(selective random scan)
  - 随机扫描会对整个地址空间的IP**随机抽取进行扫描**，即IP地址没有任何的规律；选择性随机扫描是将最有可能存在漏洞主机的地址集合作为扫描的地址空间，这也是随机扫描策略的一种。
  - **所选的目标地址按照一定的算法随机生成**，互联网地址空间中未分配的或者保留的地址块不在扫描之列。例如，Bogon蠕虫列表中包含近32个地址块，这些地址块对公网中不可能出现的一些地址进行了标识。
  - 选择性随机扫描具有**算法简单、实现容易**的特点，若与本地优先原则结合，则能达到更好的传播效果。但选择性随机扫描**容易引起网络阻塞**，使得网络蠕虫在爆发之前易被发现，**隐蔽性差**。CodeRed，Slapper和Slammer的传播采用了选择性随机扫描策略。

# 网络蠕虫扫描策略

---

- 顺序扫描(sequential scan)
  - 顺序扫描是指被感染主机上网络蠕虫会随机选择一个C类网络地址进行传播。
  - 根据本地优先原则，网络蠕虫一般会选择它所在网络内的IP地址。若蠕虫扫描的目标地址IP为A, 则扫描的下一地址IP为A+1或者A-1。一旦扫描到具有很多漏洞主机的网络时就会达到很好的传播效果。该策略的不足是对同一台主机可能重复扫描，引起网络拥塞。W32.Blaster是典型的顺序扫描蠕虫。

# 网络蠕虫扫描策略

---

- 基于目标列表的扫描(hit-list scan)
  - 基于目标列表的扫描，是指网络蠕虫在寻找受感染的目标之前预先生成一份可能**易传染的目标列表**，然后对该列表进行攻击尝试和传播。目标列表生成方法有两种：
    - 通过小规模扫描或者互联网的共享信息产生目标列表；
    - 通过分布式扫描可以生成全面的列表数据库。理想化蠕虫Flash就是一种基于IPv4地址空间列表的快速扫描网络蠕虫。



# 网络蠕虫扫描策略

---

- 基于路由的扫描 (routable scan)
  - 基于路由的扫描是指网络蠕虫根据网络中的路由信息，对IP地址空间进行选择性的扫描的一种方法。
  - 采用随机扫描的网络蠕虫会对未分配的地址空间进行探测。而这些地址大部分在互联网上是无法路由的，因此会影响到蠕虫的传播速度。如果网络蠕虫能够知道哪些IP是可路由的，它就能够更快、更有效地进行传播，并能躲避对抗工具的检测。
  - 网络蠕虫的设计者通常**利用BGP路由表公开的信息获取互连网路由的IP地址前缀**，然后来验证BGP数据库的可用性。基于路由的扫描极大地提高了蠕虫的传播速度，以CodeRed为例，路由扫描蠕虫的感染率是采用随机扫描蠕虫感染率的3.5倍。基于路由扫描的不足之处就是网络蠕虫传播时必须携带一个路由IP地址库，蠕虫代码量大。

# 网络蠕虫扫描策略

---

- 基于DNS扫描 (DNS scan)
  - 基于DNS扫描是指网络蠕虫从**DNS服务器**获取IP地址来建立目标地址库。
  - 该扫描策略的优点在于，所获得的IP地址具有针对性和可用性强的特点。
  - 基于DNS扫描的不足是：
    - 难以得到有DNS记录的地址完整列表；
    - 编虫代码需要携带非常大的地址库，传播速度慢；
    - 目标地址列表中地址数受公共域名主机的限制。例如CodeRed I所感染的主机中几乎一半没有DNS记录。

# 网络蠕虫扫描策略

---

## ● 网络蠕虫各种扫描策略评价

- 网络蠕虫传播速度的关键影响因素有：**目标地址空间选择、是否采用多线程搜索易感染主机、是否有易感染主机列表以及是否采用多种扫描策略结合**。各扫描策略的差异主要在于目标地址空间的选择。
- 网络蠕虫感染一台主机的时间取决于**蠕虫搜索到易感染主机所需要的时间**。因此，网络蠕虫快速传播的关键在于设计良好的扫描策略。
- 一般情况下，采用DNS扫描传播的蠕虫速度最慢，选择性扫描和路由扫描比随机扫描的速度要快：对于Hit-list扫描，当列表超过1M字节时，蠕虫传播的速度就会比路由扫描蠕虫慢；当列表大于6K时，蠕虫传播速度比随机扫描还慢。目前，网络蠕虫首先采用路由扫描，

# 网络蠕虫传播途径

---

网络蠕虫一般通过以下几种方式进行传播：

- Email：电子信箱传播方式。这是一种比较普遍的方式。黑客通过把网络蠕虫以**附件的形式发给邮件用户。通常这样的附件题目都是人们关心的话题**，用户处于好奇点击附件打开时，网络蠕虫激发运行，并搜索该用户通过邮件的联系人，并把带有此附件的邮件发给这些联系人。如此蠕虫在网络快速传播了。这类网络蠕虫有求职信病毒(W32. Wantjob. worm)等。
- FTP：通过利用某些FTP服务程序(如Wu\_FTP等)的漏洞进行传播，此类网络蠕虫比较少见。且危害往往也不是很大。例子有Linux. Ramen. worm 。

# 网络蠕虫传播途径

---

- HTTP: 此类网络蠕虫很常见. 利用Microsoft IIS或Apache服务器的**漏洞**来进行传播, 如红色代码(W32. CodeRed. Worm)。
- NetBios: 网络蠕虫可以通过NetBios. 利用**打开的局域网共享**资源进行传播, 比如W32. Hai. Worm等:
- RPC: 此类蠕虫病毒通过Microsoft Windows操作系统中的**远程过程调用**RPC (Remote Procedure Call), 服务中的漏洞进行传播。明显的例子是“**冲击波**”蠕虫病毒(W32. Blaster. Worm)。此病毒造成了巨大的破坏. 而且此RPC漏洞也被认为是有史以来危害最严重、传播最广泛的漏洞之一。
- 数据库: 网络蠕虫可以利用**网络上的数据库**进行传播, 比如利用Microsoft SQL Server数据库的漏洞进行传播。

# 病毒、蠕虫和木马

- 病毒、蠕虫和木马的比较

特性	病毒	蠕虫	木马
传染性	强	很少	很少
传播能力	强	极强	一般
感染对象	文件	进程	进程
主要传播方式	文件	网络	网络
破坏性	强	强	很少
隐蔽性	强	强	极强
顽固性	较强	较强	极强
欺骗性	一般	一般	强
主要攻击目的	破坏数据和信息	耗尽计算机资源	窃取信息、提供后门

恶意代码——

恶意代码技术——启动

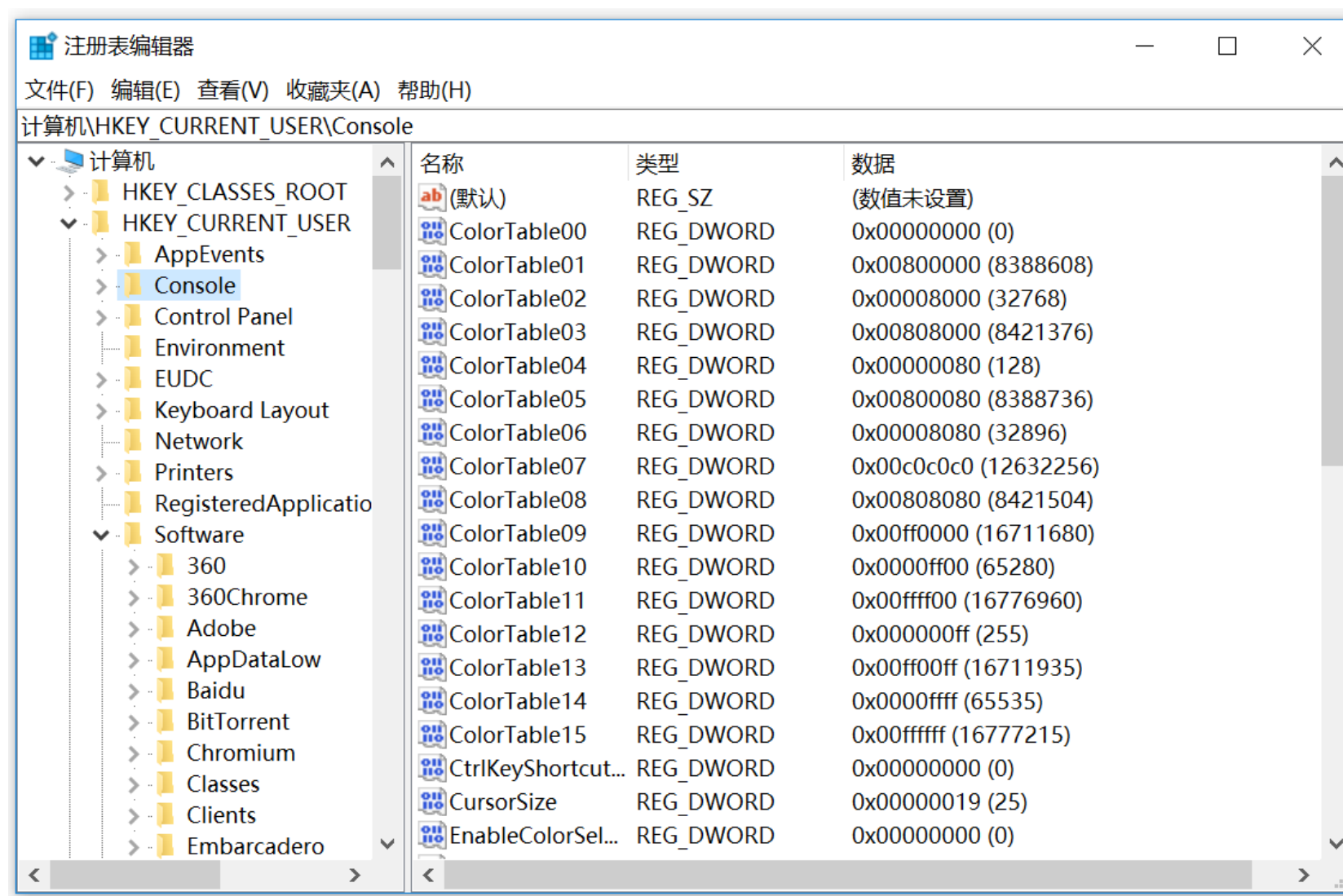


# Windows系统木马自启动方式

## 方式一：通过注册表启动

Windows的注册表

(Registry) 是一个庞大的数据库，存储软硬件的有关配置和状态等信息，应用程序和资源管理器外壳的初始条件、首选项和卸载数据，系统的设置和各种许可，文件扩展名与应用程序的关联，硬件的描述、状态和属性，计算机性能记录和底层的系统状态信息等。

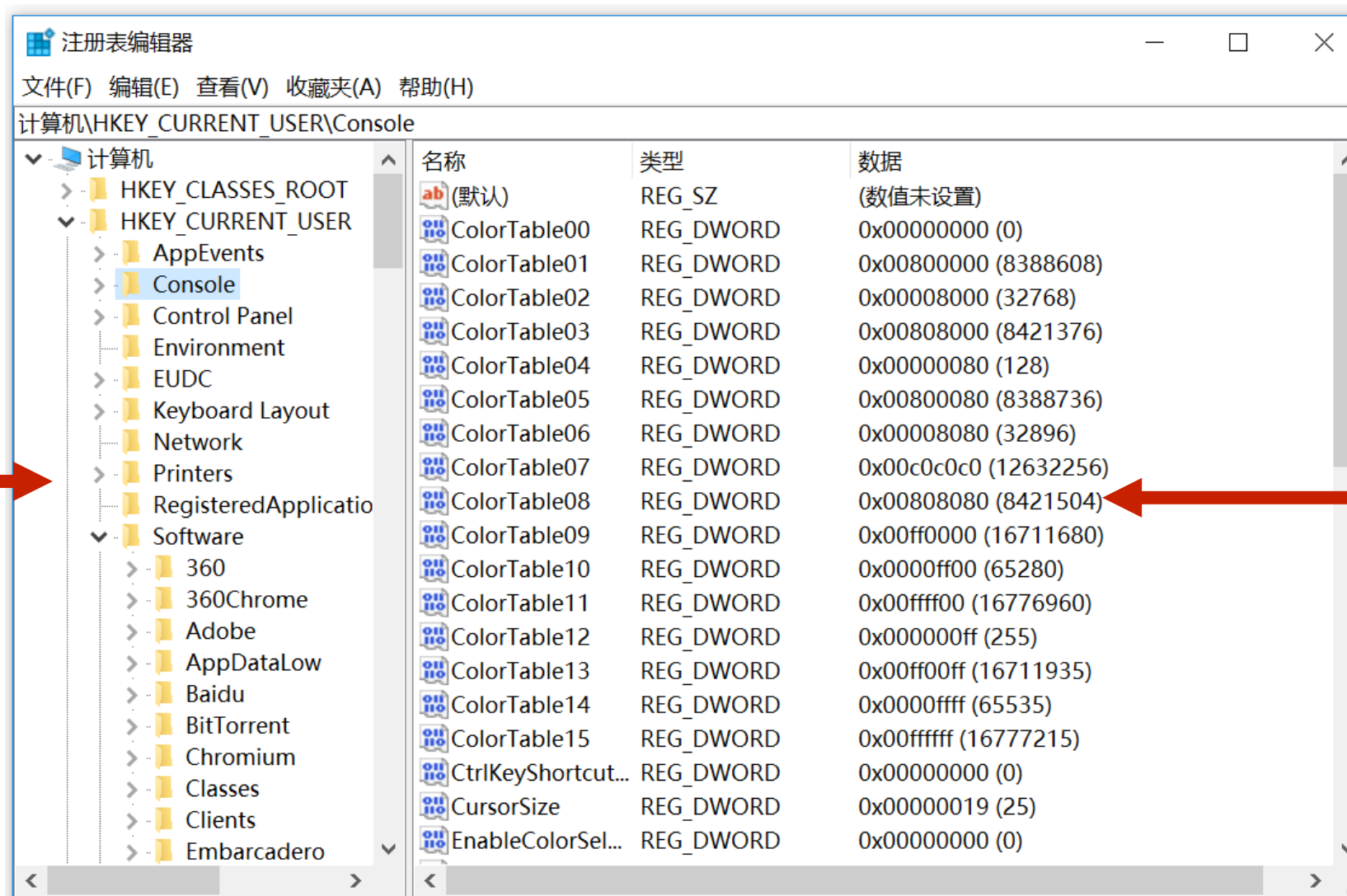




# 通过注册表启动

注册表编辑器（Regedit.exe）是Windows提供给用户的一个查看和维护注册表的工具，其界面与资源管理器的界面类似。注册表编辑器的界面如下图所示，界面左边窗格为注册表主键及子键，右边窗格为键值的名称、类型和数据。

主键和子键



键值



# 通过注册表启动

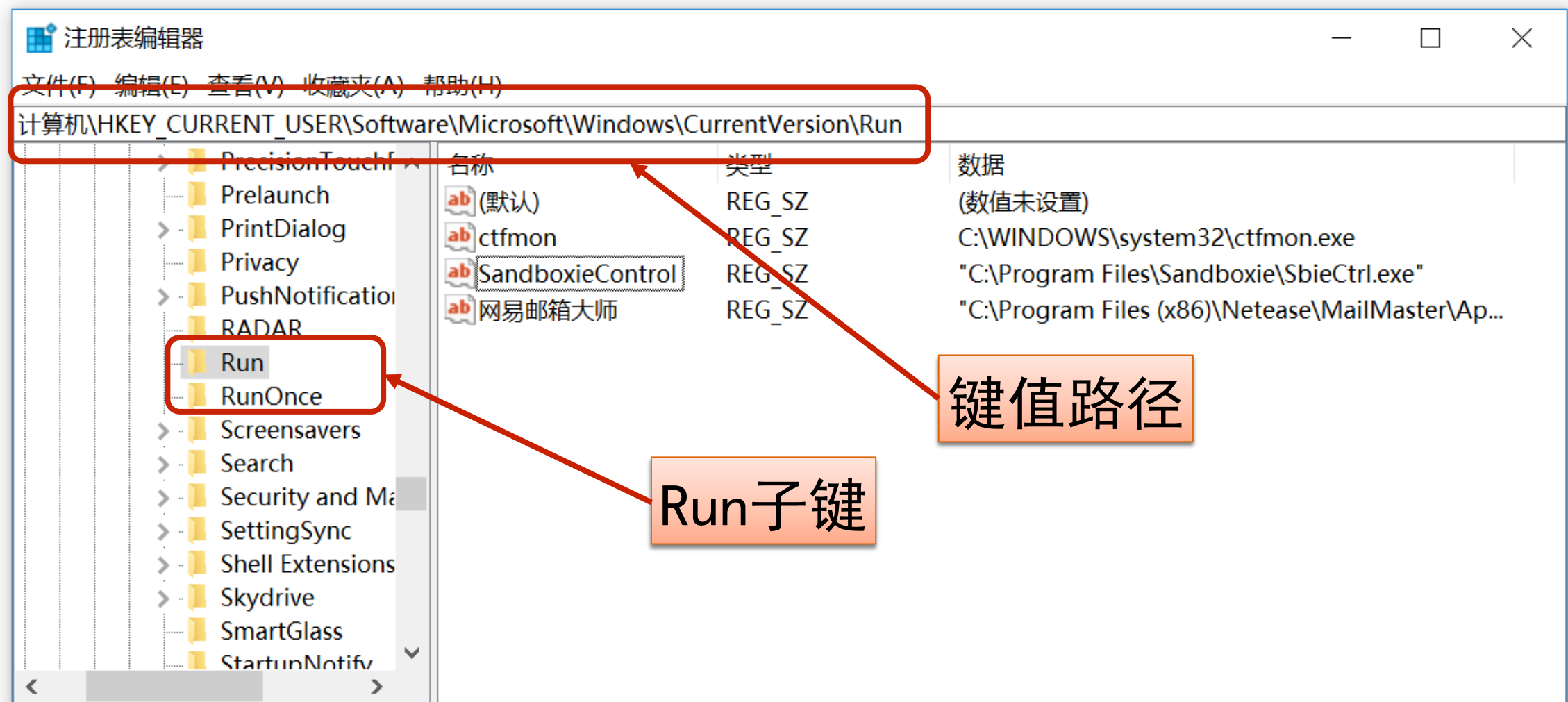
---

## ● 通过注册表中的Run来启动

在注册表中有一系列以“Run”开头的子键，该键下的键值在系统启动后会被系统加载运行。具体位置有以下几种：

- ❑ [HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run]：在系统启动时执行；
- ❑ [HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce]：在系统下次重新启动时执行的程序；
- ❑ [HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnceEx]：在启动过程中要运行的脚本和程序，作为其它各种进程中独立的线程调用；
- ❑ [HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices]：在系统启动时提供的服务列表；

# 通过注册表启动



注：“RunOnce”子键下的键值在Windows启动时执行完后会被清空（只启动一次），有些木马利用这一特点，在系统关闭时写入该子键，下次启动时可以隐蔽地动木马且。

# 通过注册表启动



[HKEY\_CURRENT\_USER\  
CurrentVersion\ Policies\Explorer\Run] 中。



# 通过注册表启动

---

## ● 通过其他位置启动

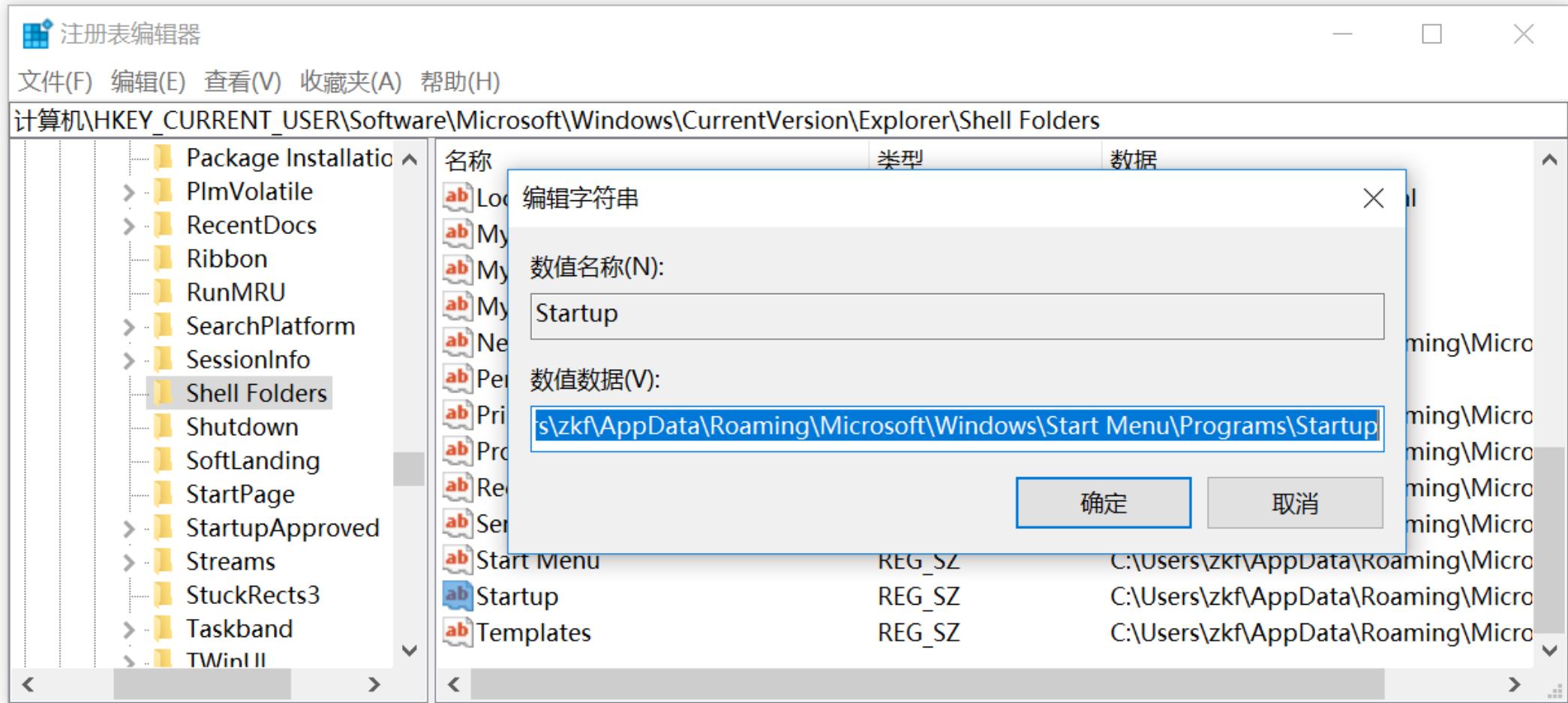
注册表中还有很多位置记录系统启动或用户登录时系统所启动执行的程序或脚本，如下：

- [HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon、Userinit]：用户登录时系统启动的程序，例如用户GUI；
- [HKEY\_LOCAL\_MACHINE\Policies\Microsoft\Windows\System\Scripts]：系统启动时要执行的脚本；  
[HKEY\_CURRENT\_USER\Software\Microsoft\WindowsNT\CurrentVersion\Windows\Load]：用户登录时装载的程序；

工具AutoRuns可以列出所有Windows系统中自启动的所有程序任务。

# 通过启动项启动

## 方式二：通过启动项启动



加在“开始”菜单的“程序\启动”中。从注册表可以找到这个文件夹，键值在[HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellFolders]下，双击Startup，就可以查看文件夹的位置。

# 通过任务计划启动

---

## 方式三：通过任务计划启动

任务计划服务让用户指定系统在一个特定的时间、特定的日期或者特定事件发生时（计算机启动时、登录时）执行特定的程序。用户可以通过“添加任务计划”向导或者代码来添加自己的任务计划。

Windows提供了六个有关任务计划的API函数：

- 增加任务计划NetScheduleJobAdd()
- 删除任务计划NetScheduleJobDel()
- 枚举特定主机上的所有任务计划NetScheduleJobEnum()
- 获取特定主机上的任务计划信息NetScheduleJobGetInfo()
- 获得任务计划服务的帐户名称GetNetScheduleAccountInformation()
- 设置任务计划服务的帐户和密码SetNetScheduleAccountInformation()

# 通过任务计划启动

下面程序段显示了在任务计划服务开启的情况下如何在本地添加一个程序myschl.exe的定时启动任务计划，展示了如何使用函数NetScheduleJobAdd()。

```
char JobPath[MAX_PATH] = "\\C:\\myschl.exe\\";
WCHAR wJobPath[MAX_PATH] = {0};
MultiByteToWideChar(CP_ACP,0,JobPath,-1,wJobPath,MAX_PATH);
//获得系统当前时间
time_t ltime;
time(&ltime);
//初始化AT_INFO结构
AT_INFO ai;
memset(&ai,0,sizeof(ai));
ai.Command = wJobPath;
ai.DaysOfMonth = 0;
ai.DaysOfWeek = 0;
ai.Flags = 0;
ai.JobTime = ltime+10*60*1000;

//在10分钟后执行
NET_API_STATUS nStatus;
DWORD JobId = 0;
nStatus = NetScheduleJobAdd(NULL,
LPBYTE(&ai), &JobId);
if(nStatus != NERR_Success)
{
    //错误信息输出及错误处理
    return FALSE;
}
```



# 通过配置文件启动

---

## 方式四：通过配置文件启动

- 批处理文件

在启动**Windows**时，系统会自动加载运行一些批处理文件，这些地方也往往是木马的实现自动加载的地方。

**Winstart.bat**用于在早期**Windows**系统中启动**MS-DOS**程序，多数情况下为应用程序及**Windows**自动生成，在执行了**Win.com**并加载了多数驱动程序之后开始执行。

**Autoexec.BAT** 和 **config.sys** 是从 **MS-DOS** 时代就流传下来的系统配置文件，对于 **MS-DOS** 系统来说它是很重要的，许多系统配置的关键参数及设备驱动程序都是由此文件定义。从 **Win95** 开始，**Autoexec.BAT** 和 **Config.SYS** 的大部分功能都已由 **Windows** 接管，这两个文件从 **Win95** 开始实际都已经没有了作用，**Win95** 以后的系统保留它们只是为了兼容旧版程序。对于大部分 **Win95** 以上版本的系统，这两个文件都是空的。

攻击者通过简单地加入与程序相关的命令行解释器实现木马的启动。

# 通过配置文件启动

---

## ● System.ini和Win.ini文件

- ❑ WIN. INI控制Windows用户窗口环境的概貌(如窗口边界宽度、系统字体等)，而system. INI包含整个系统的信息(如显示卡驱动程序等)，是存放Windows启动时所需要的重要配置信息的文件，相当于DOS中的CONFIG. SYS。
- ❑ 在Win. ini的[windows]字段中有启动命令“load=”和“run=”，这里放的是启动windows后自动执行的程序，在一般情况下“=”后面是空白的，如果有程序，比方说是：run=c:/windows/file.exe或load=c:/windows/file.exe，这个file.exe很可能就是木马。
- ❑ System. ini的[boot]字段的shell=Explorer.exe是木马喜欢的隐蔽加载之所，木马通常的做法是将该句变为这样：shell=Explorer.exe window.exe，注意这里的window.exe就是木马程序。

# 通过配置文件启动

---

另外，**System.ini**中的**[386Enh]**字段，一般是放置系统本身和外加的驱动程序。在此段内的“**driver=路径/程序名**”，这里也有可能被木马所利用。再有，在**System.ini**中的**[mic]**、**[drivers]**、**[drivers32]**这三个字段，这些段也是起到加载驱动程序的作用，但也是增添木马程序的好场所。

## ● 通过应用程序的配置文件启动

\*.**ini**文件，即应用程序的启动配置文件，攻击者可以利用这些文件能启动程序的特点，制作带有木马启动命令的同名文件，这样就可以达到启动木马的目的了。

# 通过文件关联启动

---

## 方式五：通过文件关联启动

所谓文件关联，通俗地讲就是指规定某种格式的文件用哪个程序打开，或用某个程序来打开哪些文件。例如我们双击**.txt**文件，一般情况下会默认**notepad.exe**程序打开，但如果我们愿意，可以用**word**等其它文字编辑工具打开。

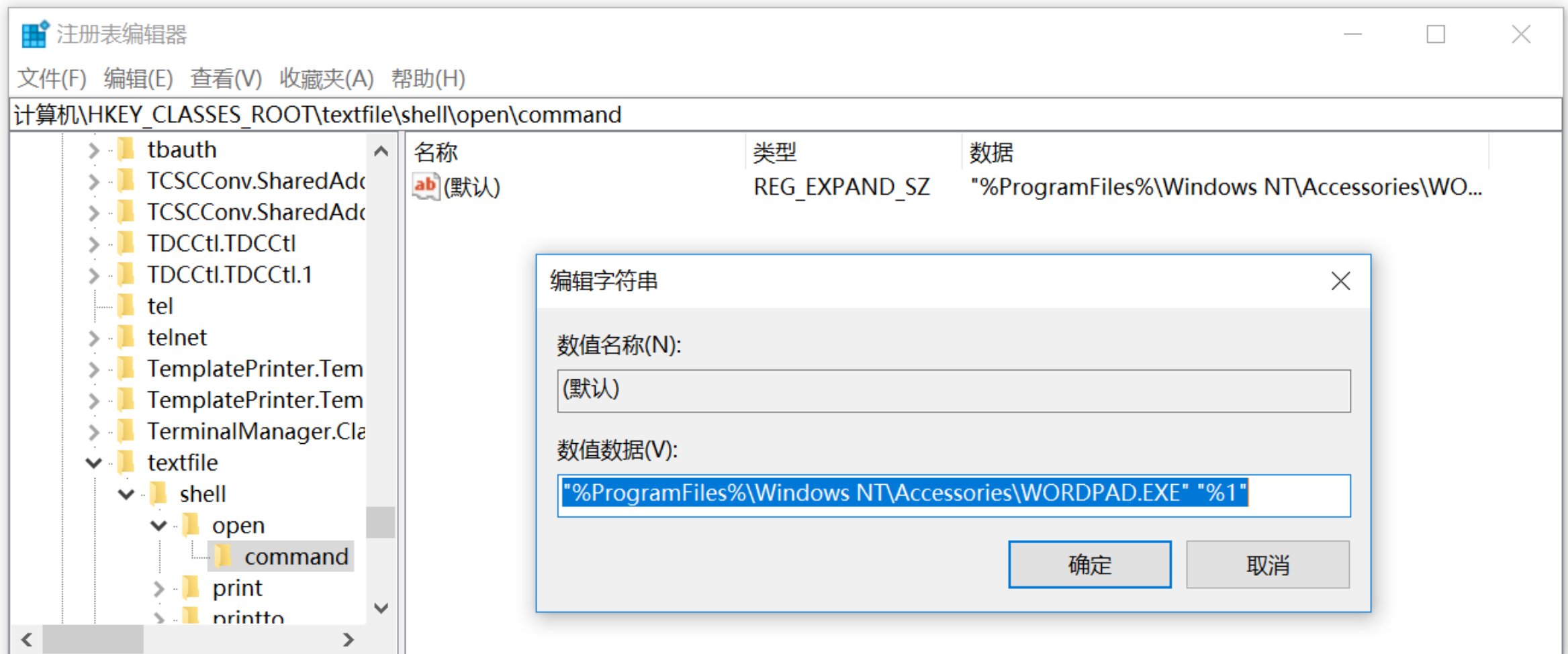
文件关联可以通过“我的电脑”——“工具”——“文件夹选项”——“文件关联”来设置，但对于一个木马来说，比较方便的还是修改注册表。在注册表的**HKEY\_CLASSES\_ROOT**主键下，可以看到许多扩展名——所有的文件类型都必须在这里注册，才是系统识别的文件类型。

# 通过文件关联启动

---

- 在注册表HKEY\_CLASSES\_ROOT主键下，找到.txt子键，可以看到其默认键值的数据项为txtfile。
- 仍然在HKEY\_CLASSES\_ROOT主键下，我们还可以找到名为txtfile的主键。在这里，我们将.txt称为扩展名主键，将txtfile称为标识主键。
- 展开txtfile，如下图，txtfile的默认值“文本文档”就是该类文档的类型描述，还可以看到DefaultIcon等子键。
- 其中，DefaultIcon定义该类文件的默认图标，Shell子键是文件关联的主要部分，其子键open、print、printto表示对文档的操作类型，操作子键下的command子键则是相应操作的执行函数。

# 通过文件关联启动



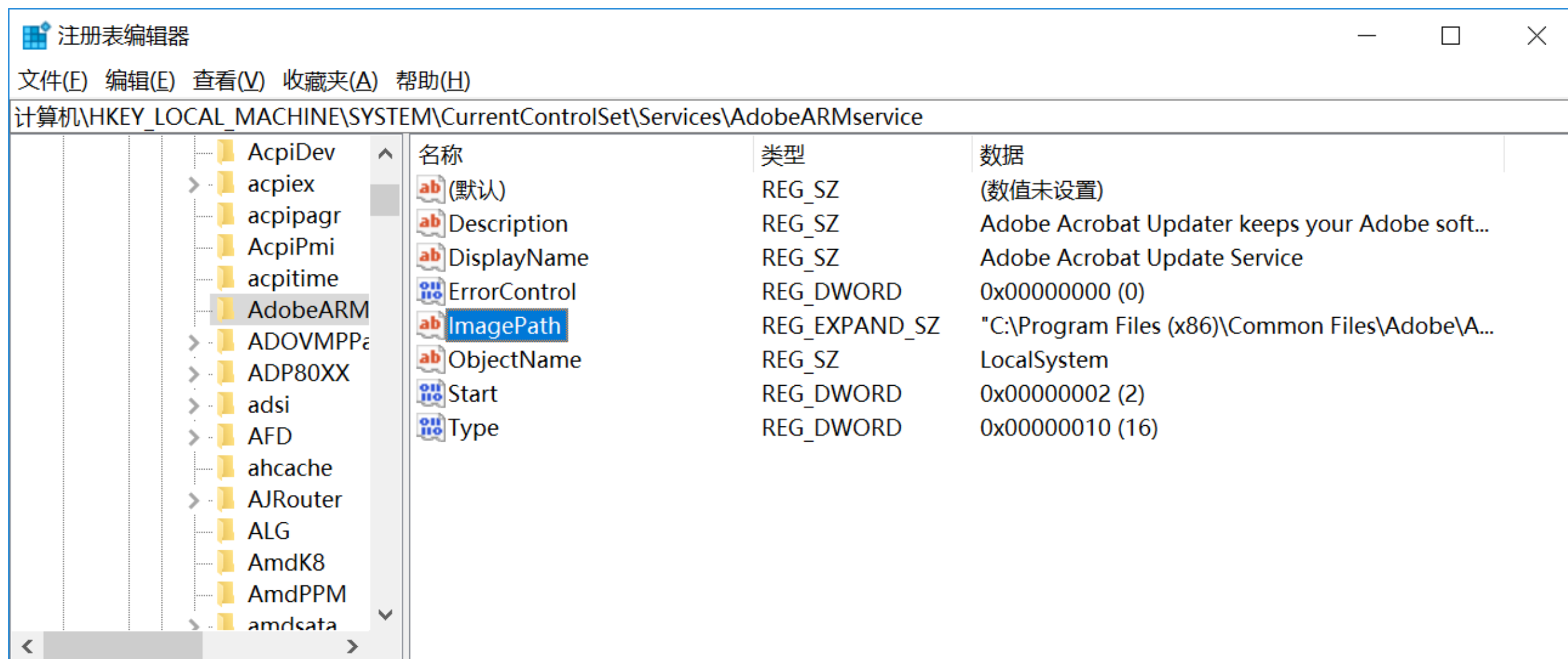
我们比较关心的是open操作子键。右键点击一个文本文档，菜单中出现的“打开”操作，就是在这里定义的。现在，我们将open键的command子键该为用xxx程序打开，然后打开一个记事本文档，出现的就不是一个文档了，改为xxx程序。



# 通过伪装成服务启动

## 方式六：通过伪装成服务来启动

注册表的[HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services]键记录了系统内的所有服务。点击子键，可以看到服务名称、执行路径、启动类型等属性，点击**Adobe ARM Service**服务子键，右边窗口显示的为该服务的属性。



# 通过伪装成服务启动

注意属性主要有四个：**DisplayName**显示名称、**ImagePath**执行文件路径、**Start**启动类型和**Type**服务类型。

数值	名称	意义
0x00	SERVICE_BOOT_START	由系统装入程序启动。系统在这个时候还没有启动，大部分系统尚不可用，所以一般的驱动程序不会使用该值。
0x01	SERVICE_SYSTEM_START	在系统初始化时启动。
0x02	SERVICE_AUTO_START	系统启动并运行后由服务控制管理器装入。
0x03	SERVICE_DEMAND_START	手工启动。
0x04	SERVICE_DISABLED	驱动程序被禁止。

Start启动类型



# 通过伪装成服务启动

## 服务类型

数值	名称	意义
0x01	SERVICE_KERNEL_DRIVER	驱动服务
0x02	SERVICE_FILE_SYSTEM_DRIVER	文件系统驱动服务
0x10	SERVICE_WIN32_OWN_PROCESS	只有单个服务程序
0x20	SERVICE_WIN32_SHARE_PROCESS	可以有多个服务程序

明白了服务的启动类型和服务类型之后，可以理解服务型木马的运作原理了：将自己注册为一个服务程序，设置对应的服务启动类型为0x00、0x01或0x02，就可以实现自启动了。

# 通过伪装成服务启动

---

**Windows**系统给我们提供了很多对服务进行操作的接口：

- ❑ 打开某台主机的服务控制管理器 (Service Control Manager)
- ❑ `OpenSCManager()`
- ❑ 创建服务 `CreateService()`
- ❑ 启动服务 `StartService()`
- ❑ 枚举服务 `EnumServicesStatusEx()`
- ❑ 更改服务配置属性 `ChangeServiceConfig()`

详细内容请参考**MSDN**。

# 通过其它特定程序启动

---

## 方式七：通过其它特定程序启动

### ● 寄生在特定程序中

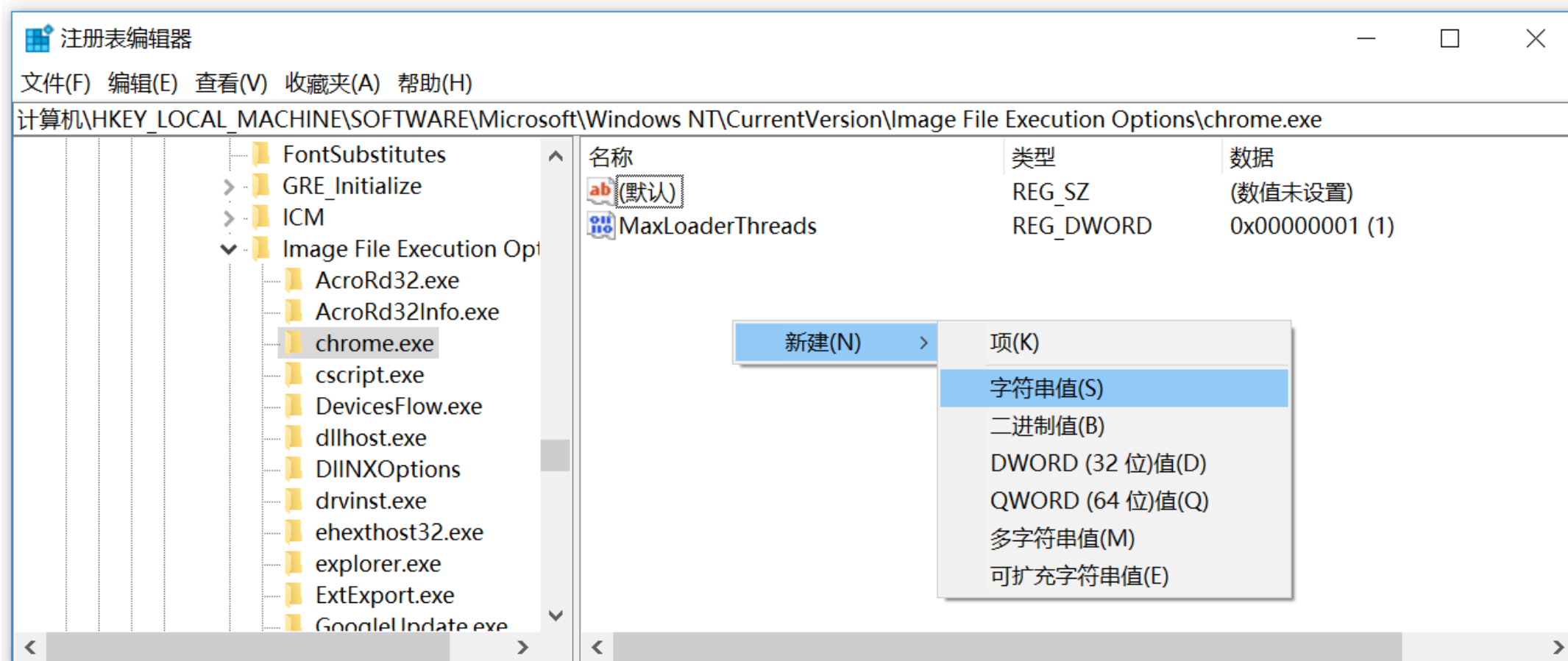
即木马和正常程序捆绑，在正常程序文件后面加节，或者加在正常程序文件节与节的空隙中实现捆绑，用户运行植入木马的“正常程序”时，木马程序先获得控制权启动运行。

### ● 代替其它程序启动

在系统运行过程中，有许多程序是自动运行的，例如磁盘清理程序（`cleanmgr.exe`）、输入法程序（`internat.exe`）等。通过覆盖系统自动运行的文件，而不必修改任何设置，系统就可以自动运行它们。或者还可以利用其它小技巧，如利用`system32`目录比`Windows`目录优先的特点，以相同的文件名，将程序放到`system32`目录中，例如可以将木马程序改名为`notepad.exe`，放入`system32`目录下，这样在执行记事本程序时就会首先执行木马程序。

# 通过其它特定程序启动

**映射劫持方法。** 将原本**A**的映射关系**A**→**A**改为**A**→**B**，这样当用户输入命令**A**时，实际上运行的是命令**B**。修改映射关系的注册表键值为**[HKEY\_CURRENT\_USER\Software\Microsoft\WindowsNT\CurrentVersion\Image File Execution Options]**，在该主键下添加子键，子键名称为要修改的程序名称（包括后缀名），子键的键值数据为替代程序的完整路径。



# 通过其它特定程序启动

---

## ● 通过API HOOK技术启动

这种方法主要针对DLL木马，技术性高，通过替换系统的DLL文件，让系统启动指定的程序。当用户的应用程序调用某个API函数时，木马程序就会首先启动，然后利用函数转发器将正常的调用转发给原DLL。而当特定消息（事先约定的特定请求）到达时，木马视为对主机进行特定攻击的指令进行相应处理。

但是在system32目录下的dllcache目录（需要去掉“文件夹选项—>查看”中的“隐藏受保护的操作系统文件（推荐）”选项）对系统的DLL文件做了备份，一旦操作系统通过数字签名技术发现被保护的DLL文件被篡改，就会自动从dllcache中恢复该文件。可以通过在

[HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\Session Manager\KnownDlls]键值下修改DLL的默认启动路径来绕过DLL保护。

# 通过驱动启动

---

## 方式八：通过驱动启动

有的木马将自己伪装成驱动程序来实现自启动。

系统扫描注册表的

[HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services]键下的键值来构造驱动程序列表。这个键下的每一个子键都有一个名称为“**Start**”的REG\_DWORD型键值，表示该驱动程序加载的时间。

在安装驱动程序时，将**Start**值设为0、1、2，该驱动程序就可以在系统重新启动时启动了。有的木马就是采用了这个原理进行自启动，但难度较大。

# 通过自动运行功能启动

---

## 方式九：通过自动运行功能启动

Windows系统在默认状态下都开启了光盘和硬盘的自动运行功能，如果能借助硬盘的自动运行，木马也是可以启动的。**AutoRun.inf**文件是一个可以让硬盘在运行时自动运行程序的文件，可以仿照下面的格式书写该文件：

```
[AutoRun]                                //表示AutoRun部分开始，必须输
Open=myprograme.exe                      //指定要运行程序的路径和名称， 如果该程序不在
该盘符的根目录下，需要全路径。
```

这样，在双击该盘符时，就会自动运行程序**myprograme.exe**。但是这种方法会导致双击打不开硬盘盘符，仍然需要在木马程序中做相应处理。



# 通过浏览器启动

---

## 方式十：通过浏览器启动

通过浏览器启动的方法有很多，因为Explorer和IE浏览器在启动时会加载一些组件。这里我们介绍其中的一种——BHO启动。

### ● 通过BHO启动

BHO全称Browser Helper Object，即浏览器辅助对象，是微软推出的作为浏览器对第三程序员开放交互接口。通过这个接口，程序员可以编写代码来获取浏览器的行为，还可以编写代码来控制浏览行为。

开发好的BHO插件以COM组件的形式存在。在注册表特定的位置注册好后，每当微软浏览器启动，BHO实例就会被创建，并且和浏览器运行在相同的内存上下文里，因此能在可用的窗口和模块里完成任何操作。那么，如果将一个木马注册为BHO插件，就可以随着浏览器的启动而启动。但这种直接利用BHO挂接浏览器进程的方法实现起来比较复杂，不仅要熟悉COM组件编程，而且木马程序仍然要注册为COM SERVER后才能使用。



# 通过浏览器启动

---

## ● 通过Java Applet启动

随着IE浏览器和脚本语言的发展，很多基于IE浏览器的攻击方法也应运而生，用户往往只是点击了一个链接就遭受了病毒、木马等攻击。Java Applet就是其中的一种。

Java Applet是用Java语言编写的一些小应用程序，这些程序直接嵌入到网页中，当用户访问这样的网页时，Applet被下载到用户的计算机上，由支持Java的浏览器解释执行。而如果Applet是一段恶意代码，一旦下载到用户的缓存中，就可能修改注册表，启动木马程序。

恶意代码——

恶意代码技术——隐藏

# 隐藏方式

---

- **任务栏隐藏：** 使程序不出现在任务栏；
- **进程隐藏：** 躲避任务管理等进程管理工具；
- **端口隐藏：** 躲避嗅探工具，使用户无法发现隐蔽的网络通信；
- **通信隐藏：** 进行通信时，将目标端口设定为常用的80、21等端口，可以躲过防火墙的过滤；
- **隐藏加载方式：** 通过各类脚本允许木马；修改设备驱动程序或修改动态链接库（DLL）来加载木马；

# 进程隐藏技术

---

- 进程隐藏，就是通过某种手段，使用户不能发现当前运行着的病毒（此处主要是指木马、蠕虫）进程，或者当前病毒程序不以进程或服务的形式存在。
- 病毒的进程隐藏包括两方面：**伪隐藏和真隐藏**。
  - 伪隐藏，就是指木马程序的进程仍然存在，只不过是消失在进程列表里；
  - 真隐藏，则是让木马程序彻底的消失，不以一个进程或者服务的方式工作。

# 伪隐藏

---

- 在 Windows 9x系统下，常通过将木马程序注册为服务的方式实现隐藏。在windows NT/2000/xp下，可以运用API HOOK技术，拦截PSAPI的EnumProcess-Modules或PDH, ToolHelp API等相关函数，控制检测工具对进程或服务的遍历调用，实现隐藏。

# 伪隐藏

---

- API HOOK技术指应用程序调用真正的系统API前先被截获，进行一些处理再调用真正的API函数。
- 根据被替换API函数的位置API HOOK技术可分为：用户级API HOOK和内核级API HOOK。典型的API HOOK方法有：改写API执行代码、修改PE文件导入表/导出表、修改系统服务分配表(SSDT)、修改中断描述符表(IDT)等。

# 真隐藏

---

- 真隐藏的基本原理是将木马核心代码以线程，DLL等的方式插入到远程进程中，由于远程进程是合法的用户程序，用户又很难发现被插入的部分，从而达到木马隐藏的目的。

# 真隐藏

---

- 在 Windows系统中常见的真隐藏实现方式有：
  - **使用注册表插入**：通过修改注册表，实现插入部件的自运行，这也是最常见的木马自启动方式；
  - **通过调试程序插入**：指利用调试程序强制将某些代码插入被调试进程的地址空间，然后使被调试进程的主线程执行该代码。这种方法要求对被调试程序的CONTEXT结构进行操作；
  - **通过CreateProcess插入、远程线程插入**：远程线程插入指通过在另一个运行的进程中创建远程线程的方法进入它的内存地址空间，该方法灵活性比较大，但不能用于Windows9x系统。



# 进程隐藏方法

---

- **远程线程注入技术**：将木马插入到其它进程
- 利用**Windows挂钩来插入DLL**：使用Hook插入DLL
- 基于**svchost服务**的进程隐藏：使用svchost启动木马
- 基于**Hook SSDT**的进程隐藏：修改SSDT，截获系统服务调用
- 基于**修改活动进程链表**的进程隐藏技术：修改全局内核变量PsActiveProcessHead链表中EPROCESS（进程执行块）的ActiveProcessLinks
- 基于**DKOM**的进程隐藏技术：通过内核模块直接（Direct Kernel Object Modify）修改达到隐藏
- 基于**SPI**的进程隐藏技术：通过服务提供者接口（Service Provider Interface）加载

# 文件隐藏技术

---

- 取与系统文件类似的名字，迷惑；
- 合并文件，有个别木马程序能把它自身的.exe文件和服务器端的图片文件绑定；
- 采用NTFS文件系统一个文件中包含多个数据流；
- API Hook技术：文件的枚举最终调用的是NtQueryDirectoryFile函数；
- 系统服务描述符表（System Service Descriptor Tables）：通过修改SSDT，截获系统服务调用；
- 通过文件系统过滤驱动来实现：主要是通过拦截系统的文件操作，其实就是拦截I/O管理器发向文件系统驱动程序的IRP。

# 通信隐藏技术

---

- 通信隐藏是在用户毫无感知的情况下完成木马客户端与服务器端的数据交互。
- 早期的黑客隐蔽通信技术很低级，例如，著名冰河木马等采用的通信技术都是简单的TCP连接方式，这样用户很容易就可以发现非法的连接，从而发现恶意代理的存在。
- 后来，黑客的隐蔽通信技术不断发展，端口反弹技术、ICMP型木马技术等隐蔽通信技术纷纷出现。

# 通信隐藏技术

---

- **隐秘通道**，利用IP，ICMP协议头中optional域和传输数据时很少利用的域)
- **在用户态下进行网络数据包拦截**
  - Winsock Layered Service Provider
  - Windows 2000 包过滤接口
  - 替换系统自带的WINSOCK动态连接库
- **内核态拦截网络数据包**
  - TDI过滤驱动程序 (TDI Filter Driver)
  - NDIS中间层驱动程序 (NDIS Intermediate Driver)
  - Win2k Filter-Hook Driver
  - NDIS Hook Driver

# 利用CreateRemoteThread() 函数注入

---

- 代码注入的方式很多，最著名的当属CreateRemoteThread()这个API函数，它可以在其它进程的地址空间中开启远程线程。它的原型如下：

```
HANDLE CreateRemoteThread(  
    HANDLE hProcess,  
    LPSECURITY_ATTRIBUTES lpThreadAttributes,  
    SIZE_T dwStackSize,  
    LPTHREAD_START_ROUTINE lpStartAddress,  
    LPVOID lpParameter,  
    DWORD dwCreationFlags,  
    LPDWORD lpThreadId  
);
```

- 它与CreateThread()函数很相似，唯一的区别是多了一个参数hProcess，即目标进程的句柄。

# 利用CreateRemoteThread() 函数注入

---

- 如果要注入的代码是一个**DLL**文件，即将一个**DLL**注入到其他进程中，需要考虑的一个问题就是怎样运行**DLL**中的代码。
- 只开启了远程线程还不够，还需要将**DLL**加载到目标进程中，这时需要在已开启的远程线程中调用**LoadLibrary()**函数。
- 一旦**LoadLibrary()**函数调用成功，**DLL**中的入口函数**DllMain()**就会获得执行机会，剩下的所有工作都可以交给**DLL**实现，因此远程线程中需要执行的代码只是一个简单的**LoadLibrary ()**调用。

# 利用CreateRemoteThread() 函数注入

---

- CreateRemoteThread()函数的第四个参数lpStartAddress，即线程函数起始地址直接设置为LoadLibraryA()（也可以使用LoadLibraryW函数）的地址即可。
- 获取LoadLibrary()的地址很简单，此函数由Kernel32.dll导出，Windows总是将Kernel32.dll映射到相同的地址，也就是说，不同的进程中Kernel32.dll的加载地址总是相同的，所以只要在自身的进程中获取到LoadLibrary()的地址就可以了。
- 具体方法是首先调用GetModuleHandle()函数获取Kernel32.dll的句柄，再调用GetProcAddress()函数获得LoadLibrary()的地址。



# 利用CreateRemoteThread() 函数注入

---

- CreateRemoteThread()函数还有一个参数比较重要，即lpParameter，它是线程函数的参数，也就是LoadLibrary()的参数，即DLL的文件名。一般情况下，我们只要先定义一个常量字符串，然后将字符串指针作为参数即可。但是在远程线程中却不可以这么做，原因很简单，我们调用的LoadLibrary()函数是在目标进程的地址空间中的，而我们自己定义的字符串却在自身进程的地址空间中，由于两个不同的地址空间互相隔离，所以LoadLibrary()函数将不能获得我们提供的参数。
- 解决的办法也比较简单，**那就是将字符串参数也放入目标进程的地址空间中**。完成这个功能，需要用到两个函数：**VirtualAllocEx()和WriteProcessMemory()**，它们的作用分别是在指定进程空间中分配内存和在指定进程空间中写入数据。

# 利用CreateRemoteThread() 函数注入

---

- 注入DLL的关键步骤已基本介绍完，但是还有一个值得注意的问题，我们要注入一个进程，前提条件是要操作该进程的权限。
- 通过简单实验就可以发现，很多系统进程的是无法用函数OpenProcess()获得ID的，即使Administrator用户也没有权限操作这些被操作系统所保护的进程。因此**必须将自身的权限提升，使其具有SeDebugPrivilege权限。**

# 利用CreateRemoteThread() 函数注入

---

- 提升进程权限的方法是获取进程的访问令牌，然后修改该令牌。获取进程访问令牌的**API**函数是 **OpenProcessToken()**，其原型为

```
BOOL OpenProcessToken(  
    HANDLE ProcessHandle,  
    DWORD DesiredAccess,  
    PHANDLE TokenHandle  
);
```

- 其中**TokenHandle**为获得的令牌句柄。

# 利用CreateRemoteThread() 函数注入

---

- 要修改访问令牌需要另一个API:  
**AdjustTokenPrivileges()**,原型如下:

```
BOOL AdjustTokenPrivileges(  
    HANDLE TokenHandle,  
    BOOL DisableAllPrivileges,  
    PTOKEN_PRIVILEGES NewState,  
    DWORD BufferLength,  
    PTOKEN_PRIVILEGES PreviousState,  
    PDWORD ReturnLength  
);
```

# 利用CreateRemoteThread() 函数注入

---

- 其中第三个参数**NewState**是一个指向**TOKEN\_PRIVILEGES**结构体的指针，这个结构体定义了令牌权限的信息，其定义如下

```
Typedef struct _TOKEN_PRIVILEGES {  
    DWORD PrivilegeCount;  
    LUID_AND_ATTRIBUTES Privileges[ANYSIZE_ARRAY];  
  
} TOKEN_PRIVILEGES, *PTOKEN_PRIVILEGES;
```

- **Privileges**是一个数组，**PrivilegeCount**是数组元素的个数。

# 利用CreateRemoteThread() 函数注入

---

- **LUID\_AND\_ATTRIBUTES**的定义如下

```
typedef struct _LUID_AND_ATTRIBUTES {  
    LUID Luid;  
    DWORD Attributes;  
}  
LUID_AND_ATTRIBUTES, *PLUID_AND_ATTRIBUTES;
```

- 其中**Attributes**为操作的类型，**Luid**为权限类型。由于**Luid**是一个64bit数值，所以我们需要知道**SeDebugPrivilege**权限对应的**Luid**值。另一个API提供了这个功能，就是**LookupPrivilegeValue()**。

# 利用CreateRemoteThread() 函数注入

---

- LookupPrivilegeValue()的原型为:

```
BOOL LookupPrivilegeValue(  
    LPCTSTR lpSystemName,  
    LPCTSTR lpName,  
    PLUID lpLuid  
);
```

- 我们通过指定参数lpName来确定权限类型，Winnt.h中已经预定义了一些权限的宏定义，SeDebugPrivilege对应的宏为SE\_DEBUG\_NAME。



# 利用CreateRemoteThread() 函数注入

---

- 远线程注入DLL的流程：
  - 提升自身进程的权限为**SeDebugPrivilege**，（如果不注入系统进程，此步骤可省略）。
  - **OpenProcess()**获取目标进程的句柄。
  - **VirtualAllocEx()**在目标进程中分配一块内存。
  - **WriteProcessMemory()**将要注入的**DLL**路径写入步骤3分配的内存。
  - **GetProcAddress()**获得**LoadLibraryA ()**函数的地址。
  - **CreateRemoteThread()**创建远程线程，线程起始地址设为**LoadLibraryA ()**函数的地址，线程的参数为步骤4中**DLL**路径在目标进程中的地址。
  - 等待远程线程结束后退出。

恶意代码——

恶意代码技术——传播

# 网络蠕虫传播模型

---

- 蠕虫传播数学建模是分析网络蠕虫高速传播并阻断的重要方法。通过对模型的分析，可以更好地理解网络蠕虫的传播特性，预测其发展趋势，采取防御网络蠕虫措施。
- 一个完整的网络蠕虫传播模型包括传播规则的制定和网络拓扑结构的选择。
- 因此在初期研究中，传播模型如基本的传染病模型及对SIR模型的改进模型等都是基于规则网络的。
- 近年来大量的研究表明，真实世界的复杂网络既具有小世界效应，又具有无尺度特性。因此2000年以后，网络蠕虫传播模型的研究重点已经转移到基于复杂网络的网络模型上。

# 网络蠕虫传播模型

---

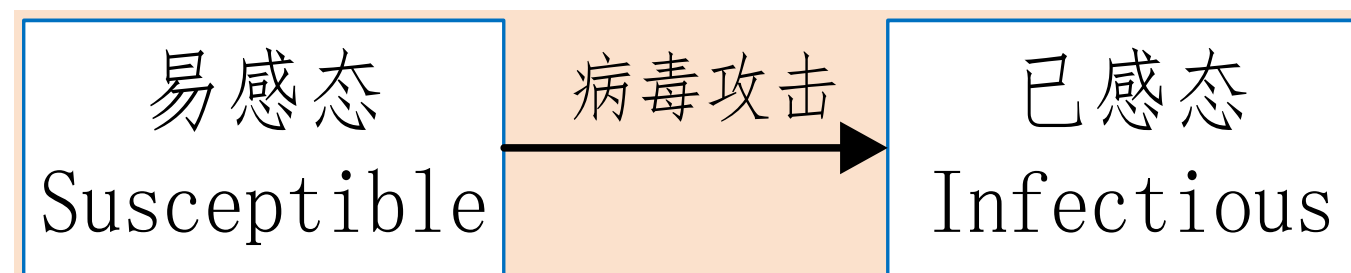
## ● 传染病模型

- 在传染病学模型中个体被分为了三种状态，即“**易感态 (Susceptible)**”、“**已感态 (Infected)**”、“**免疫态 (Recovered)**”。一个模型的类型通常决定于模型中个体状态的变化，例如，SI模型就表示个体的状态由易感态到已感态。
- 将传染病模型应用于网络蠕虫的传播研究是目前比较普遍的建模方法。第一个完整的网络蠕虫传播模型是由Kephart J. O.、White S. R. 建立的，他们对建模过程进行了如下两个基本假设：
  - 忽略传播细节，包括传播途径、传播方法、传播所用的时间等。
  - 模型网络为一个均匀网络，即网络中的个体总数固定不变，其中的每一个个体对外的连接数基本相等，在网络中具有相等的重要性。
- 这两个假设条件都存在于以下我们首先要介绍的基本模型中。

# 网络蠕虫传播模型

## ● SI模型

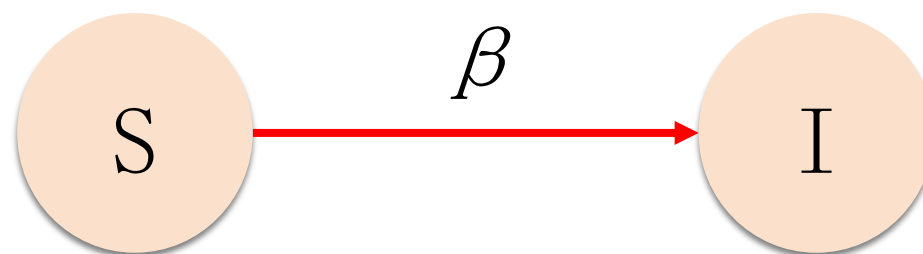
- **SI模型是传染病模型的经典模型，又称简单传播模型。**处于SI模型中的主机只有两个状态，**易感染状态和已感染状态**，而且一旦被感染，将保持被感染的状态不变。SI模型的状态转移图如图所示。
- 考虑足够多的主机数时，SI模型可以近似于连续状态连续时间的确定性模型，例如SEM (Simple Epidemic Model) 模型。SEM模型中，蠕虫传播采用随机扫描，主机的状态不影响主机间的通信过程。
- SI模型仅仅考虑了主机的易感和已感两种状态，但SI模型却是其它众多模型的研究基础，因而要研究网络蠕虫的传播模型。



SI模型的状态转移图

# 经典传播模型——SI模型

Susceptible-Infected模型，简称SI模型



$\beta$  为感染率

*Susceptible* 易感状态

*Infected* 感染状态

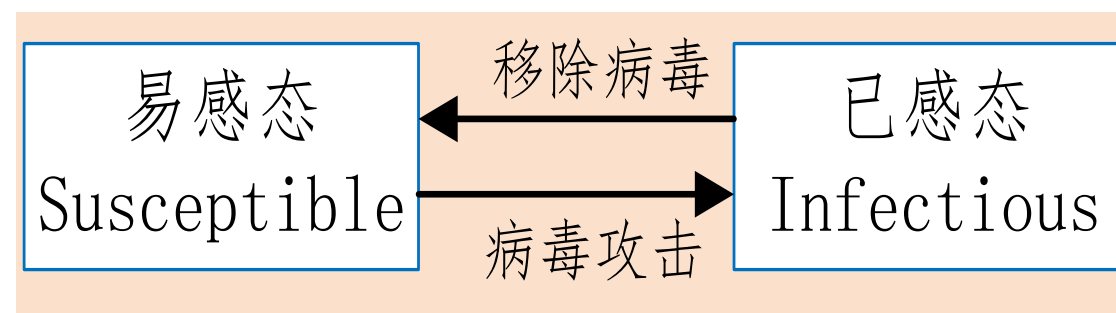
$$\begin{cases} \frac{dS}{dt} = -\beta \frac{SI}{N} \\ \frac{dI}{dt} = \beta \frac{SI}{N} \end{cases} \longrightarrow \frac{di}{dt} = \beta i(1-i) \longrightarrow i(t) = \frac{i_0 e^{\beta t}}{1 - i_0 + i_0 e^{\beta t}}, i_0 = i(0)$$

- $\beta SI$  表示易感染者单位时间内向感染者转化的增量， $N$  为总节点数

# 网络蠕虫传播模型

- SIS模型

- SIS模型与SIR模型类似，只是**已感主机被治愈后恢复为易感状态**，其状态转移图如图所示。
- 显然，该模型与SIR模型有相同的弊端。另外，假设一台主机被修复后状态又变回到易感染状态，也是与实际大多数情况所不同的。

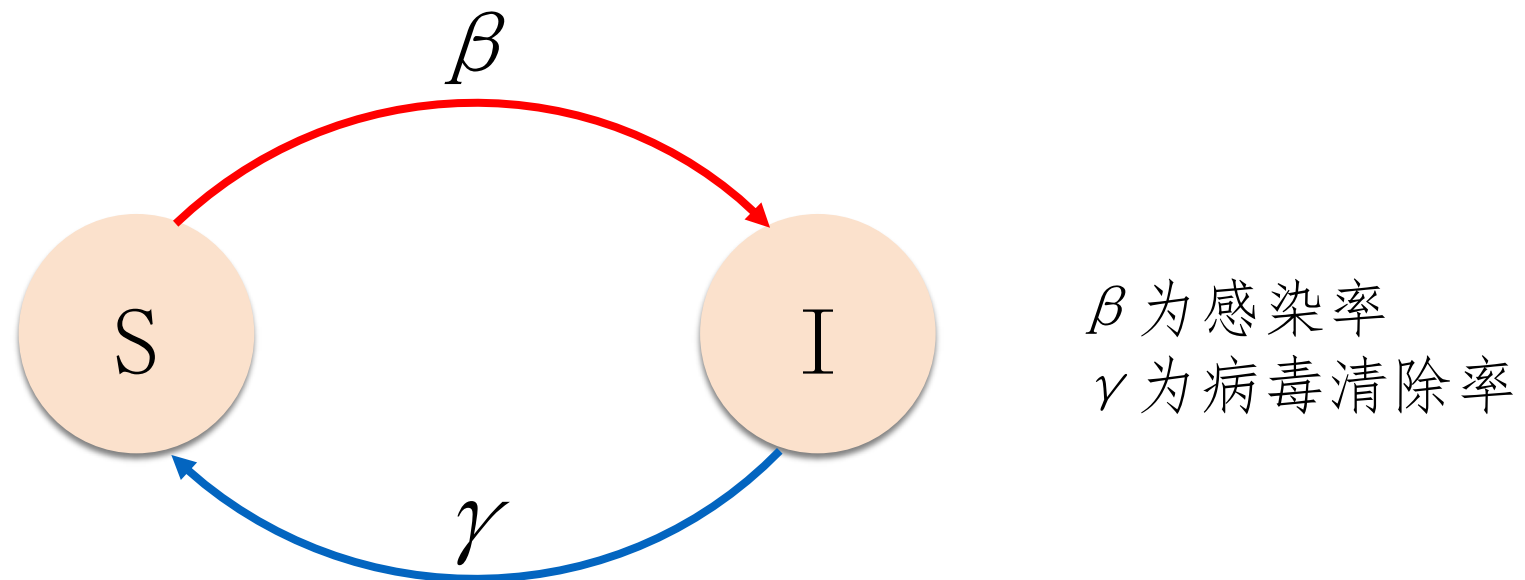


SIS模型的状态转移图



# 经典传播模型——SIS模型

Susceptible-Infected-Susceptible模型，简称SIS模型



$$\begin{cases} \frac{dS}{dt} = -\beta SI + \gamma I \\ \frac{dI}{dt} = \beta SI - \gamma I \\ S(0) = N - I_0 \\ I(0) = I_0 \end{cases}$$

- $\beta SI$  表示易感染者单位时间内向感染者转化的增量
- $\gamma I$  表示单位时间内被治愈的感染者数量
- $N$  为总节点数,  $I_0$  为病毒传播的初始时刻被感染的主机的数目。

# 网络蠕虫传播模型

- SIR模型

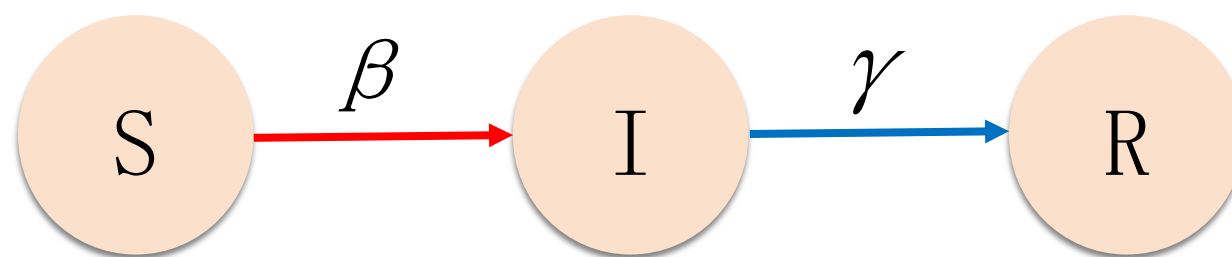
- SIR模型考虑了已感染主机，经过查杀蠕虫后成为免疫主机的过程。但此处的“免疫”其实有两种含义：**一为清除蠕虫并且不再具有感染此蠕虫的脆弱性；一为主机在模型考察的时间范围内因感染蠕虫而失去存活能力，相当于从网络中移除。**
- 模型的状态图如图所示：



SIR模型的状态转移图

# 经典传播模型——SIR模型

Susceptible-Infected-Removed模型，简称SIR模型



$\beta$  为感染率  
 $\gamma$  为病毒清除率

$$\begin{cases} \frac{dS}{dt} = -\beta SI \\ \frac{dI}{dt} = \beta SI - \gamma I \\ \frac{dR}{dt} = \gamma I \\ S(0) = N - I_0, I(0) = I_0, R(0) = 0 \end{cases}$$

$\beta SI$  表示易感染者单位时间内向感染者转化的增量

$\gamma I$  表示单位时间内被治愈的感染者数量

$N$  为总节点数， $I_0$  为病毒传播的初始时刻被感染的主机的数目。

# 无尺度网络模型

---

- 相比较上述基于随机网络的计算机病毒传播模型，基于无尺度网络的传播模型更符合实际情形，因而更具有研究价值。
- 无尺度网络具有增长（Growth）和偏好连接（Preferential attachment）两个特性，第一个特性表明无尺度网络可以不断地扩张，第二个特性则意味着两个节点连接能力的差异可以随着网络的扩张而增大，即“富者愈富”。
- 无尺度网络比随机网络更适合计算机病毒的传播，一旦出现一种新的计算机病毒，在同样的传染强度下，无尺度网络中感染的主机数要比其在随机网络中的数目大的多。
- 无尺度网络中计算机传播模型的建立要考虑很多因素，比如集散节点的选择、传播策略等，如果选择算法过于复杂，病毒主体难以保证足够小，会影响传播速度，缩短病毒寿命。

恶意代码——

# 恶意代码分析

见“恶意代码分析”课件

# 问题和讨论