

# 第9章 软件安全与恶意代码

- **本章学习要点：**

- 了解软件安全的概念以及当前软件安全威胁的主要来源；
- 熟悉软件安全风险分析的过程，特别是微软STRIDE威胁建模方法；
- 了解安全软件开发生命周期的构成方式以及主要步骤；
- 理解并掌握恶意软件的定义，特别是病毒，蠕虫和木马三者的机理与防治技术。

# 9.1 软件安全概述

## ● 9.1 软件安全概述

- 软件安全是指：采取工程的方法使得软件在敌对攻击的情况下仍继续正常工作。
- 方法：采用系统化、规范化、数量化的方法来指导构建安全的软件。
- 软件安全是计算机安全问题中的一个关键问题。软件的缺陷，包括实现中的错误，以及设计中的错误，已经出现很多年了。同时随着软件系统的不断增加和越来越复杂，使得安全隐患也不断增多。

## 9.2 软件体系安全分析

- 安全软件开发的体系安全需要考虑**安全风险分析、威胁建模、安全风险**管理3个方面。
- 软件体系安全分析方法主要有3个关键步骤，即**抗攻击分析、不确定性分析、弱点分析**。
- **抗攻击分析**主要分析对已知的攻击、攻击模式和弱点的抗攻击能力，通常采用清单的方式。
- **不确定性分析**主要针对发现新的风险，创造性要求较高，需要有经验的分析人员参与。
- **弱点分析**是指分析软件所依赖的外部软件的弱点。

## 9.2.1 基于标准的风险分析

### ● 1. NIST的ASSET

- 自动安全自评估工具ASSET由NIST提出。ASSET能自动完成信息系统安全自我评估指南中包含的调查表，调查表的结果提供了一种评价特定系统安全的方法。通过对调查表的解释，用户可以评估组织内的信息系统安全，以及组织安全项目规划的安全性。
- **ASSET包含两种工具：ASSET系统和ASSET管理者。**
- ASSET系统提供4种报告的能力，这4种报告是：**根据有效性进行主题领域的总结、非可应用问题列表、基于风险的决策列表、系统总结报告。**
- ASSET管理者也提供了4种报告：**所有系统的总结、根据类型进行系统列表、系统敏感程度列表、组织总结。**

## 9.2.1 基于标准的风险分析

### ● 2. CMU SEI (卡耐基梅隆大学软件工程研究院) 的OCTAVE

- CMU SEI提出的操作型关键威胁，评级以及漏洞评估系统 (OCTAVE)。它是一个标识和管理信息安全风险的框架，由一套基于风险的信息安全策略评价和规划工具、技术和方法组成。同时它定义了一种允许组织标识信息资产的综合评估方法从而设计和实现保护策略来减少信息资产的整体风险。
- OCTAVE方法使用三阶段方法来检查组织和技术问题。这种方法主要关注：**发现关键的评估标准以及对这些评估的威胁；发现人员组织上和技术上的弱点，面临的威胁、风险。开发基于实践的保护策略，抵御风险的规划，优先级等。**

## 9.2.2 STRIDE威胁建模

### ● 1. STRIDE威胁模型

- STRIDE建模方法由微软提出，该方法通过审查系统设计或架构来发现或纠正设计级 (design-level)的安全问题。
- 软件开发人员认为安全主要是指代码质量，网络管理员认为安全主要是防火墙、以及系统管理。学术界认为安全是指Saltzer和Schroeder原则。
- 而安全的含义包括机密性、完整性、可用性、对用户正确进行身份验证和授权、以及事务处理不可否认等，下表介绍了每个属性。

# 9.2.2 STRIDE威胁建模

常见安全属性

属性	说明
机密性(confidentiality)	数据只限应具有权限的人员访问。
完整性(integrity)	数据和系统资源只限适当的人员以适当的方式进行更改。
可用性(availability)	系统在需要时一切就绪，可以正常执行操作。
身份验证(authentication)	建立用户身份（或者接受匿名用户）。
授权(authorization)	明确允许或拒绝用户访问资源。
不可否认(nonrepudiation)	用户无法在执行某操作后否认执行了此操作。

## 9.2.2 STRIDE威胁建模

- 针对这些安全属性，给出STRIDE威胁模型，STRIDE是6种威胁类型的英文首字母缩写，这6种威胁如下：
  - (1) **欺骗证识(Spoofing identity)**，典型的例子是使用其他用户的认证信息进行非法访问。例如利用用户名和口令等认证信息进行非法访问。
  - (2) **篡改数据(Tempering with data)**，在未授权的情况下恶意地修改数据。这种修改可能是在数据库中保存的数据，也可能是在网络中传输的数据。
  - (3) **可抵赖(Repudiation)**，用户从事一项非法操作，但该用户拒绝承认，且没有方法可以证明他是在抵赖。例如某用户从事一项非法操作，但系统又缺乏跟踪非法操作的功能。
  - (4) **信息泄露(Information disclosure)**，信息暴露给不允许对它访问的人。例如用户读到没有给他赋予访问权限的文件的内容，信息在网络中传递时内容被泄密。



## 9.2.2 STRIDE威胁建模

- **(5) 拒绝服务(Denial of Service)**，拒绝对正当用户的服务。例如Web服务器短时间不可以访问，可能是遭到拒绝服务攻击，因此需要提高系统的可用性和可靠性。
- **(6) 权限提升(Elevation of privilege)**，一个没有特权的用户获得访问特权，从而有足够的权限做出摧毁整个系统的事情。

将威胁映射为防护它们的安全属性

威胁	安全性属性
假冒(Spoofing)	身份验证
篡改(Tempering)	完整性
可抵赖(Repudiation)	不可否认
信息泄漏(Information disclosure)	机密性
拒绝服务(Denial of Service)	可用性
提升权限(Elevation of privilege)	授权

## 9.2.2 STRIDE威胁建模

### ● 2. 威胁建模的过程

- ✓ 威胁建模的过程主要有4个方面。
- 发现已知的对系统的威胁。
- 将威胁以风险从高到低排序。
- 确定减少威胁的技术。
- 选择合适的技术。
- ✓ 以上过程可能反复进行多次，因为一次标识所有可能的威胁是很困难的。

## 9.2.2 STRIDE威胁建模

- **对基于Web的工资应用程序的一些威胁，包括如下几方面。**

- ✓ 威胁1：一个恶意用户，在从Web服务器到客户端的途中，或在从客户端到Web服务器的途中，查看或者篡改个人工资数据。
- ✓ 威胁2：一个恶意用户，在从Web服务器到COM组件的途中，或在从COM组件到Web服务器的途中，查看或篡改个人工资数据。
- ✓ 威胁3：一个恶意用户，直接在数据库中访问或篡改工资数据。
- ✓ 威胁4：一个恶意用户，查看LDAP认证包，并学习如何恢复它们，以便他能够冒充别的用户。
- ✓ 威胁5：一个恶意用户，通过改变一个或多个Web页，来丑化Web服务器。
- ✓ 威胁6：一名攻击者通过发送大量的TCP/IP包，使工资数据库服务器计算机拒绝访问。
- ✓ 威胁7：一名攻击者删除或者修改审核日志。
- ✓ 威胁8：一名攻击者使用分布式DoS攻击，杀死真正的工资服务器后，将他自己的工资Web服务器放在网络上。

。

## 9.2.2 STRIDE威胁建模

- **(1) 发现已知的对系统的威胁。**观察应用程序的每一部分，为组件或进程判断是否有任何S、T、R、I、D或者E威胁存在。大部分都会存在许多威胁，将它们都记录下来是很重要的。
- **(2) 将风险从高到低排序。**对每个服务器中的资产，通过以下方式决定优先级别：攻击发生的概率，即需要多少努力、代价、时间来发起攻击，1 = 高概率，10 = 低概率  
一旦攻击发生，将会带来什么破坏和损失？1 = 小损失，10 = 大损失  
**风险 = 攻击发生后的损失/攻击的概率** 1 = 小风险，10 = 大风险（即损失大，攻击概率高）  
为了减少风险，通常首先处理高风险的项目，下表中给出的比例可以作为参考。

## 9.2.2 STRIDE威胁建模

主要威胁导致的弱点在攻击中占的比例

弱 点	占攻击的比例
可旁路的限制(restrictions that can be bypassed)	20%
参数检查(argument checking)	19%
没有检查的缓冲区(unchecked buffer)	18%
不正确的控制标记(incorrect control marking)	10%
不正确的许可(incorrect permissions)	9%
架构错误(architectural error)	6%
实现错误(other implementation error)	18%

## 9.2.2 STRIDE威胁建模

- (3) 确定防御威胁的相关技术，下表给出了与威胁相关的防御技术。

防御各个威胁的技术

威胁类型	防御技术
身份假冒(spoofing identity)	认证(authentication) 保护秘密(protect secrets) 不保存秘密(do not store secrets)
篡改数据(tampering with data)	授权(authorization) 哈希函数(hashes) 消息认证码(message authentication codes) 数字签名(digital signatures) 防篡改协议(tamper-resistant protocols)
否认(repudiation)	签名(digital signatures) 时间戳(timestamps) 审计跟踪(audit trails)

## 9.2.2 STRIDE威胁建模

- (3) 确定防御威胁的相关技术，下表给出了与威胁相关的防御技术。

信息泄漏(information disclosure)	授权(authorization) 隐私保护协议(privacy-enhanced protocols) 加密(encryption) 保护秘密(protect secrets) 不保存秘密(do not store secrets)
拒绝服务攻击(denial of service)	认证(authentication) 授权(authorization) 过滤(filtering) 流量控制(throttling) 服务质量(quality of service)
权限提升(elevation of privilege)	最小权限运行(run with least privilege)

## 9.3 安全软件开发生命周期

### 软件危机

- 从20世纪50年代末开始，计算机越来越普及，并广泛应用。可到了70年代初，出现了“软件危机”。什么是软件危机？
  - ✓ 危机主要表现为：软件成本超出预算，开发进度一再拖延，软件质量难以保证。
  - ✓ 原因在于：系统规模越来越大，复杂度也越来越高，用户需求不明确，缺乏正确的理论指导。
- “软件危机”使人们意识到信息系统的开发需要一套科学的、工程化的方法来指导，这就是常说的“系统分析与设计方法”。

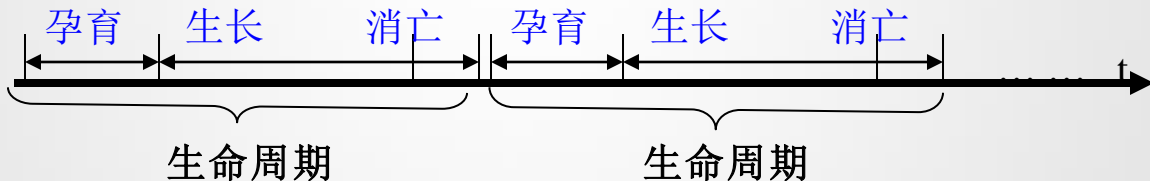


# 系统工程思想及应用

- ✓ 系统工程是一门用于大规模复杂系统设计的学问，是组织管理系统的规划、设计、制造、试验和使用的科学方法。
- ✓ 它的思想是以系统概念为基础的思想，表现为由粗到细、由表及里、由上到下、由整体到局部，逐步求精的分析。
- ✓ 系统工程方法一般步骤：调研→确定目标→确定功能→考虑方案（多个）→选择一个方案→实施→维护和评价。

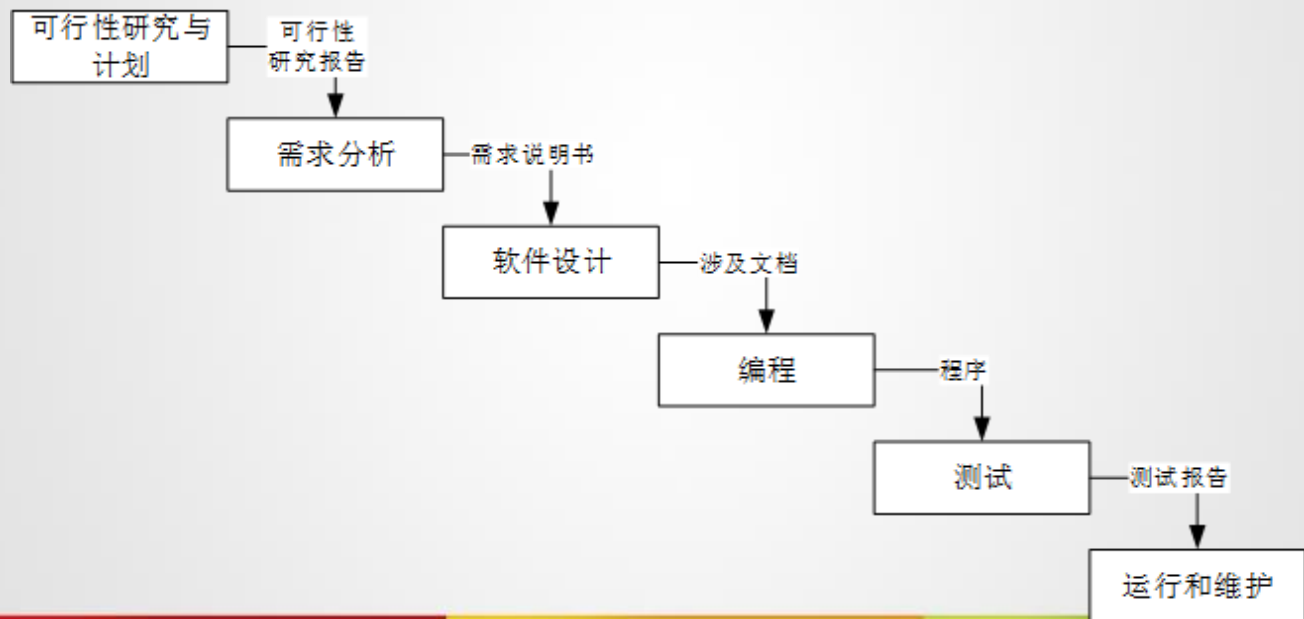
# 软件开发的生命周期

- ✓ 软件开发的生命周期（life cycle）：一个软件系统从它的提出、开发应用到系统的更新，经历一个孕育、生长到消亡的过程。这个过程周而复始，循环不息，每一次循环称为它的一个生命周期。生命周期的概念始于20世纪70年代。



## 9.3.1 传统软件开发生命周期

- 传统的瀑布模型将软件开发过程划分成若干个互相区别而又彼此联系的阶段，**这几个阶段分别为：可行性研究与计划、需求分析、软件设计、编程、测试、运行和维护**，每个阶段的工作都是以上一个阶段工作的结果为依据，同时又为下一个阶段的工作提供前提，瀑布模型如下图所示。



## 9.3.1 传统软件开发生命周期

### ● 要求：

- 1.瀑布模型的顺序活动的特点，必须按照阶段顺序地安排工作。
- 2. 要求每个阶段的工作都要有完整、准确的文档资料，并且每个阶段结束前都要对文档进行审查，尽早发现问题，尽早解决。

### ● 缺点：

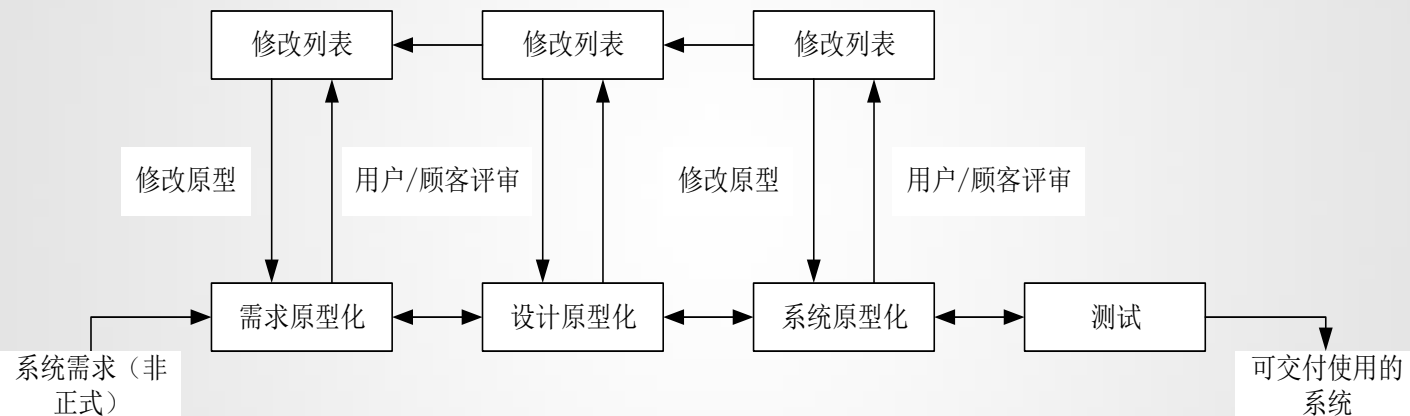
- 1.在项目开始阶段，用户对需求的描述常常是不全面的。所以理解上难免会出现遗漏或者偏差，就会影响到后面的工作。
- 2.瀑布模型是由文档驱动的。当用户在使用软件时往往会产生一些新的想法，或许会对软件的使用方面提出一些建议，而此时想对系统修改难度会很大。

## 9.3.1 传统软件开发生命周期

### ● 2. 原型模型（快速原型模型）

- 基本思想是：
- 1. 开发人员在与用户进行需求分析时，先建立一个能够反映用户主要需求的原型模型，让用户在计算机上进行操作，然后提出改进意见。
- 2. 开发人员根据用户的建议，对原型进行补充和完善，然后再由用户试用、评价、提出意见，重复这一过程，直至用户满意为止。

## 9.3.1 传统软件开发生命周期



原型模型

## 9.3.1 传统软件开发生命周期

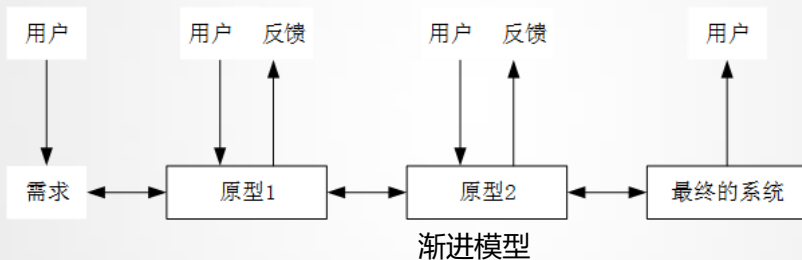
### ● 原型模型优点：

- 原型模型让用户有机会实践系统的基本功能，因而可以对不尽合理的内容提出修改意见和建议。
- 使开发者和用户充分交流，对一些模糊需求也能够处理。
- 开发人员通过建立原型模型对系统有了更深层次的理解，有助于软件的开发工作顺利进行。
- 用户在使用原型模型时已经对系统有了初步了解，建立模型的过程也相当是用户的一个学习软件的过程。
- 适合人机界面的，用户通过交互界面的内容，能够提出有关操作、功能上的建议。

## 9.3.1 传统软件开发生命周期

### 3. 渐进模型

**目的：**和客户一起工作，从最初的大概的需求说明演化出最终的系统。抛出原型的目的是**逐渐理解需求**，没必要一次性完全理解需求，如下图所示。



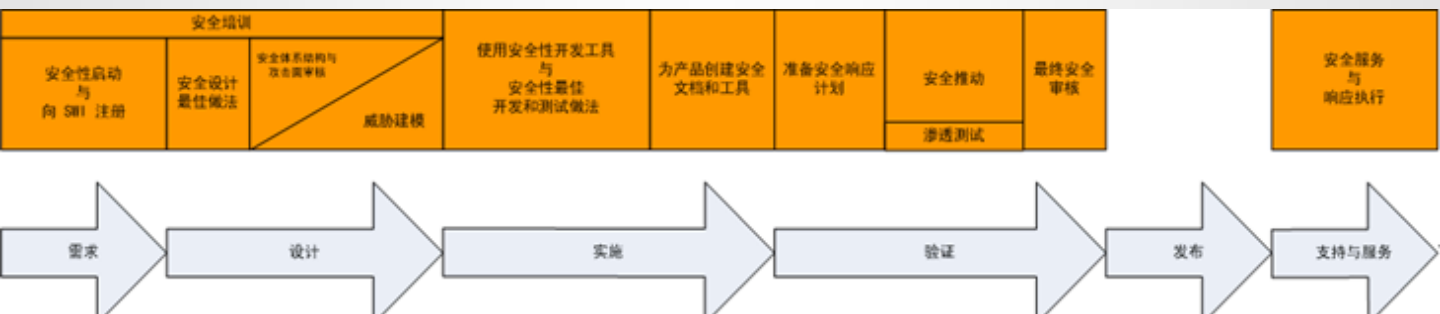
**存在的问题：**缺乏过程的可见性，系统通常不能够很好地结构化。渐进模型的一般应用在中小规模规模的交互式系统或大型系统的一部分或生命周期较短的系统。



## 9.3.2安全软件开发生命周期

- **安全软件的开发生命周期(SSDL)**通过软件开发的各个步骤来确保软件的安全性，其目标是确保安全的软件得以成功实现。通常由5个主要部分组成。
- **(1) 安全原则、规则及规章。**通常被视为保护性需求。该阶段应创建一份系统范围内的规范，在其中定义将应用到本系统的安全需求。如果系统需要遵循特定行业的安全标准，在需求分析时考虑到这些要求。
- **(2) 安全需求工程。**以攻击者的角度看待系统应注意的地方，定义了应该避免的缺点和错误。同样，这种安全需求需要通过文档在项目起始的时候就定义下来。
- **(3) 架构和设计评审、威胁建模。**软件的架构和设计应该被安全分析人员尽早评审，避免形成有安全缺陷的体系结构和设计。
- **(4) 软件安全编码。**需要代码的实现者对软件漏洞和软件安全编码原则有所了解。
- **(5) 软件安全测试。**测试系统的安全漏洞或软件漏洞，判定这些漏洞是否可被攻击者利用，从而构成威胁。

## 9.3.3其他安全软件开发生命周期



Microsoft可信技术安全开发生命周期

## 9.4.1 恶意软件的分类与区别

### ● 9.4.1 恶意软件的分类与区别

- 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中执行恶意任务的恶意代码的集合。
- 恶意软件大致分为两类，是从**主机依赖的角度**进行的分类。



## 9.4.1 恶意软件的分类与区别

### ● 1. 病毒

- 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放置在一个本地计算机上，然后执行一个或多个恶意操作。

### ● 2. 蠕虫

- 如果代码在没有携带者（宿主文件）的情况下复制，则认为是蠕虫。**病毒寻找文件以进行感染，但蠕虫仅尝试复制其自身。**

### ● 3. 木马

- 木马与病毒、蠕虫的区别在于它不进行复制。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。

## 9.4.1 恶意软件的分类与区别

### ● 4 . 其他恶意软件

#### ● (1) 后门

- 后门是在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，然后侦听的后门程序会等待攻击者的远程连接（执行恶意操作）。

#### ● (2) 逻辑炸弹

- 逻辑炸弹是合法的应用程序，只是在编程时被故意写入的某种恶意功能，在一定程度下合（如时间、次数或者某种逻辑组合）会被执行。

## 9.4.2 病毒的机理与防治

### ● 1. 病毒的定义

- **病毒**是一种人为制造的、能够进行自我复制的、对计算机资源具有破坏作用的一组计算机指令或者程序代码。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。
- **(1) 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。除了使用.exe扩展名的典型可执行文件之外，具有扩展名.com、.sys、.dll等文件。
- **(2) 脚本**。将脚本作为携带者目标文件。此类文件的扩展名包括.vbs、.js和.prl。
- **(3) 宏**。携带者是宏脚本语言的文件。
- **(4) 启动扇区**。计算机磁盘上的特定区域也可以作为携带者，因为它可以执行恶意代码。

## ● 2. 病毒的分类

### ● 按照病毒的链接方式分类

- (1) **源码型病毒**。该病毒攻击高级语言编写的程序。这类病毒一般存在于语言处理程序和链接程序中。
- (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。
- (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，相当于给宿主程序加了个外壳。  
。
- (4) **译码型病毒**。隐藏在微软Office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，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，对操作系统进行破坏。

## ● 按照病毒的寄生存储的位置分类

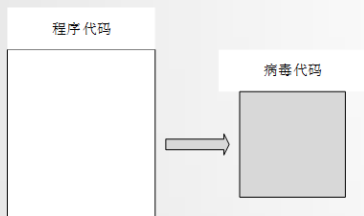
- **(1) 引导型病毒。**也称为引导区病毒。操作系统的引导模块存放在磁盘引导区，并且控制权的转接方式是以物理地址为依据，因此病毒占据该物理位置即可获得控制权。
- **(2) 文件型病毒。**文件型病毒主要感染可执行文件，如扩展名为.EXE、.COM等文件，是一种较为常见的病毒。目录病毒是文件型病毒的一种特例，其感染方式非常独特，仅修改目录区，便可达感染的目的。宏病毒则是一种数据文件型病毒。
- **(3) 混合型病毒。**也称为多型病毒，是综合了引导型和文件型病毒特征的病毒，可感染文件和引导扇区两种目标。



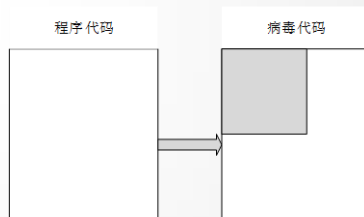
### ● 3. 文件型病毒的感染技术

#### ● (1) 重写病毒

- 这种病毒从磁盘上找到一个文件，简单地用自己的副本改写该文件，是一种较初级的技术。下图表示了重写病毒攻击时宿主文件内容的变化。



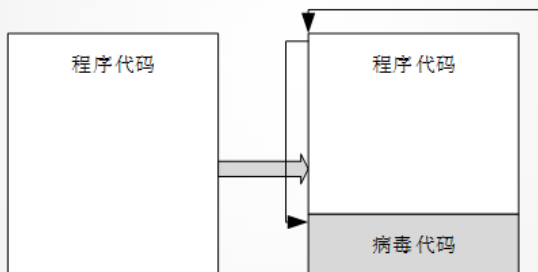
图a 重写病毒攻击时改变宿主文件大小



图b 重写病毒攻击时未改变宿主文件大小（只重写了宿主文件的顶部）

## ● (2) 追加病毒

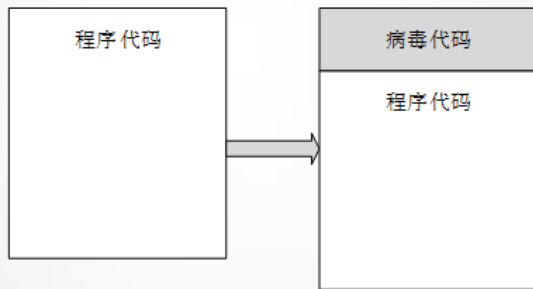
- 典型的DOS环境下的COM文件感染技术是在宿主文件的头部插入一条JMP指令，指向初始文件的尾部（即追加的病毒代码）。下图显示典型的DOS COM追加病毒。



追加病毒

### ● (3) 前置病毒

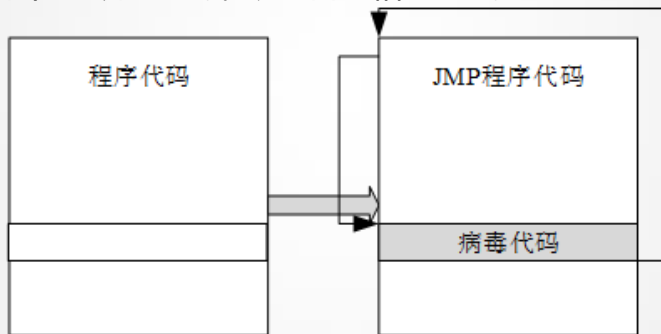
- 把病毒的代码插入到宿主程序的前面，这些代码通常采用高级语言如C、PASCAL等实现，通常在磁盘上创建一个包含原文件内容的临时文件，然后用system这样的函数执行临时文件中原来的程序。



- 前置病毒

## ● (4) 蛀穴病毒

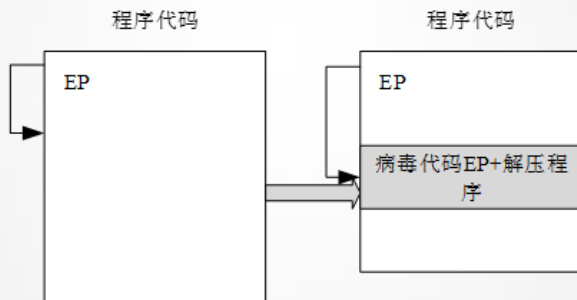
- 蛀穴病毒通常不增加被感染对象的大小，而是重写宿主文件中可用来安全存放病毒代码的区域，如重写二进制宿主文件中的零值区域，或包含空格的区域。



蛀穴病毒将自身代码注入到宿主文件的一个洞穴中

## ● ( 5 ) 压缩型病毒

- 压缩宿主程序是一种特殊的感染技术。这种技术有时用来隐瞒宿主程序长度的增长：采用一个二进制的压缩算法，对宿主程序进行充分的压缩，从而节省了空间。



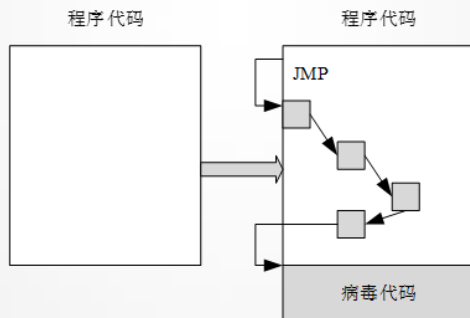
- 压缩性病毒

## ● (6) 变形虫感染技术

- 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。病毒头部可以访问尾部，然后被载入。病毒在硬盘上生成一个包含原始宿主内容的新文件，以便于它将来可以正确运行。

## ● (7) 嵌入式解密程序技术

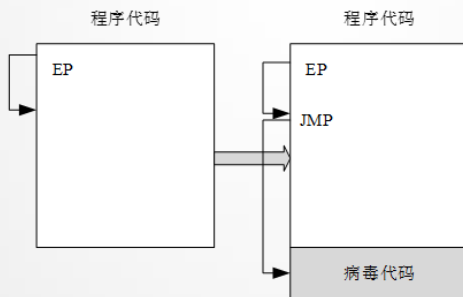
- ✓ 高级的病毒会把其解密程序注入可以执行的宿主文件中，并将宿主入口点修改为指向解密程序代码。解密程序的注入位置是随机选择的，解密程序被分割成多个部分。
- ✓ 当被感染程序执行时，解密代码被执行，解密病毒代码，并执行该代码
- ✓ 更难进行相应的病毒检测



- 嵌入式解密程序技术

## ● (8) 迷惑性欺骗跳转技术

- ✓ 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。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。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。下图显示了这种跳转技术。





## 9.4.2病毒的机理与防治

### ● 4. 病毒的检测

- 传统的病毒检测方法有：比较法、搜索法、分析法、感染实验法、软件模拟法等。目前广泛采用的病毒检测技术有病毒行为监测技术、启发式代码扫描分析技术、虚拟机查毒技术等。

#### ● (1) 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。包括长度比较法、内容比较法、内存比较法、中断比较法等。
- 长度比较法和内容比较法检测长度和内容的变化，并以此作为判定的依据。
- 内存比较法是一种对内存驻留病毒进行检测的方法。由于病毒驻留内存，因此必须在内存中申请一定的空间，并占用该空间，通过对内存的检测，观察其空间变化，判定是否有病毒驻留空间。
- 比较法的好处是简单方便，缺点是无法确认病毒的种类。

## ● ( 2 ) 校验和法 ( checksum )

- 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。在文件过程中，定期地 *检测文件的校验和，判别是否和原来保存的校验和一致*，从而判断文件是否感染病毒，这种叫校验和法。它既可以发现已知病毒，也可以发现未知病毒。该方法无法识别病毒种类。

## ● ( 3 ) 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法。


## ● (4) 行为监测法

- 利用病毒的特有行为特性监测病毒的方法称为行为监测法。行为检测法的优点在于不仅可以发现已知病毒，而且可以预报多数未知病毒。缺点在于可能有误报和不能识别病毒的名称，实现难度较大。

## ● (5) 感染实验法

- 感染实验是一种简单实用的检测方法，该方法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，自主检测可疑的新病毒。
  - 先运行可疑程序，再运行其他原本正常的程序，观察这些程序是否被感染（大小，校验和等发生变化）

## ● (6) 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。
  - 在虚拟机下运行病毒程序，观察其演化过程，比对特征行为和特征代码。
- 

# 9.4.3蠕虫的机理与防治

## 1. 蠕虫与病毒的区别及联系

- 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进，但是两者还是有区别，如下表所示。了解这些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

属性	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	自身的复制
传染机制	宿主程序运行	系统存在漏洞
搜索机制	主要针对本地文件	主要针对网络上的计算机
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	为系统补丁
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

## 9.4.3 蠕虫的机理与防治

### ● 2. 蠕虫的分类

- 蠕虫的传播、运作方式，可将蠕虫分为两类：**主机蠕虫**和**网络蠕虫**。
- **主机蠕虫**。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复制到其他计算机中。
- **网络蠕虫**。由许多部分（称为段）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。

## 9.4.3 蠕虫的机理与防治

### ● 3. 蠕虫与软件漏洞的关系

- 根据蠕虫利用漏洞的不同，可将其细分为**邮件蠕虫**、**网页蠕虫**和**系统漏洞蠕虫**。

#### ● (1) 邮件蠕虫

- ✓ 邮件蠕虫要利用多用途网际邮件扩充协议 ( MIME ) 漏洞。MIME 允许邮件附带多媒体文件，蠕虫程序伪装成音乐或图片等形式附在邮件里，当用户收到邮件，里面的多媒体文件被运行时，蠕虫程序则被执行。

#### ● (2) 网页蠕虫

- 网页蠕虫主要利用IFrame漏洞和MIME漏洞。
- 网页蠕虫可以分为两种：一种是用一个IFrame漏洞插入一个邮件框架，同时利用MIME漏洞执行蠕虫，这是直接沿用邮件蠕虫的方法；另一种是用IFrame漏洞和浏览器下载文件的漏洞来运作的，首先由一个包含特殊代码的页面去下载放在另一个网站的恶意文件，然后运行它，完成蠕虫传播。

## 9.4.3 蠕虫的机理与防治

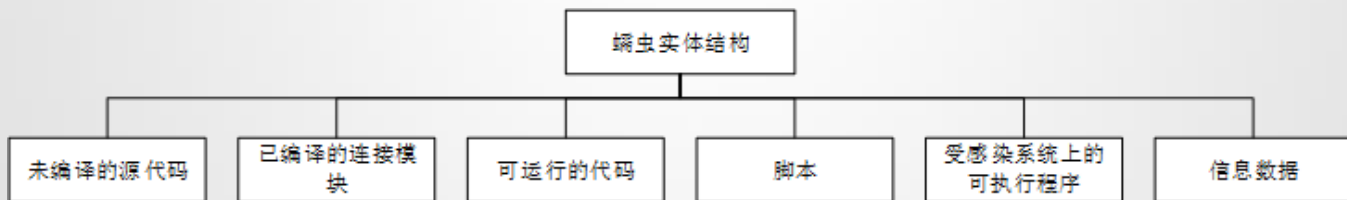
### （3）系统漏洞蠕虫

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。

## 4. 蠕虫的基本结构

### （1）蠕虫的实体结构

蠕虫程序相对于一般的应用程序，在实体结构方面体现了更大的复杂性。通过对蠕虫程序的分析，可以把实体结构分为如下6个部分，具体的某个蠕虫可能仅包含其中几个部分，如图所示。



## 9.4.3 蠕虫的机理与防治

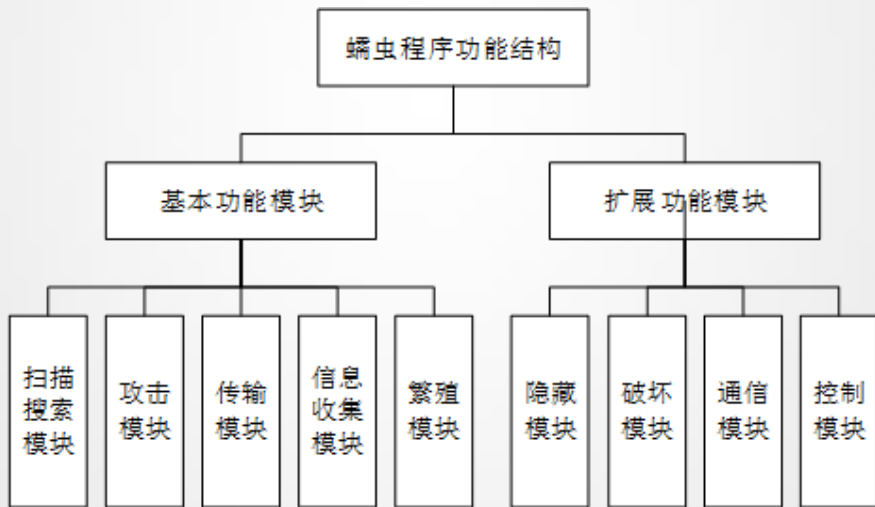
- **(a) 未编译的源代码**：由于某些程序参数必须在编译时确定，所以蠕虫程序可能包含一部分未编译的程序源代码。
- **(b) 已编译的链接模块**：不同的系统，可能需要不同的运行模块，例如，不同的硬件厂商和不同的系统厂商可能采用不同的运行库。
- **(c) 可运行代码**：整个蠕虫可能由多个编译好的程序组成。
- **(d) 脚本**：利用脚本可以节省大量的程序代码，充分利用系统shell的功能。
- **(e) 受感染系统上的可执行程序**：受感染系统上的可执行程序，如文件传输等，可以被蠕虫作为为自己的组成部分。
- **(f) 信息数据**：包括已经破解的口令、要攻击的地址列表、蠕虫自身的压缩包等。



## 9.4.3 蠕虫的机理与防治

### （2）蠕虫的功能结构

蠕虫在功能上可以分为**基本功能模块**和**扩展功能模块**。实现了基本功能模块的蠕虫程序就能完成复制传播流程，包含扩展功能模块的蠕虫程序则具有更强的生存能力和破坏力。蠕虫程序的功能结构如图所示。



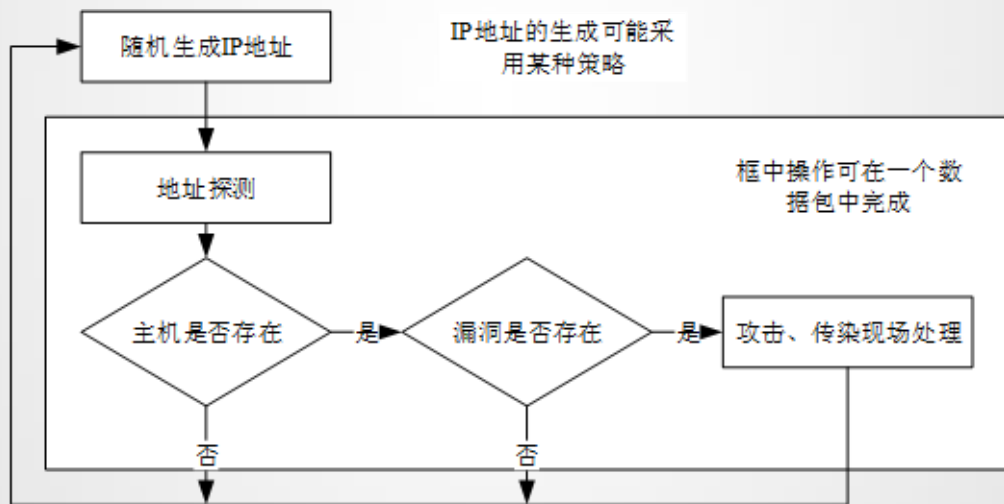
## 9.4.3 蠕虫的机理与防治

- **基本功能由5个功能模块组成：**
  - **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。
- **扩展功能模块由4个部分组成：**
  - **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门程序等。
  - **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重点。
  - **控制模块：**调整蠕虫行为，更新其他功能模块，控制被感染计算机。

## 9.4.3 蠕虫的机理与防治

### ● 5. 蠕虫的工作方式

- 蠕虫的工作方式一般是：扫描→攻击→复制，如图所示。



## 9.4.3 蠕虫的机理与防治

- (1) 搜索扫描

- ✓ 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。

- (2) 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。

- (3) 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。复制过程也有很多种方法，可以利用系统本身的程序实现，也可以用蠕虫自带的程序实现。复制过程实际上就是一个网络文件传输的过程。

## 9.4.3 蠕虫的机理与防治

### ● 6. 蠕虫的防治与检测

#### ● (1) 蠕虫防治的方案

- 蠕虫防治方案可以从两个方面来考虑：

- 第一，从它的实体结构来考虑，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；
- 第二，从它的功能组成来考虑，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。

- (1) 修补系统漏洞。主要是由系统服务提供商负责，及时提供系统漏洞补丁程序，用户及时安装补丁。

## 9.4.3 蠕虫的机理与防治

- **(2) 分析蠕虫行为。**通过分析特定蠕虫的行为，给出有针对性的预防措施。
- **(3) 重命名或者删除命令解释器。**如UNIX系统下的shell、Windows系统下的WScript.exe。重命名和删除命令解释器，可以避免执行蠕虫实体中的脚本。
- **(4) 防火墙。**禁止除服务端口外的其他端口，这将切断蠕虫的传播通道和通信通道。
- **(5) 公告。**通过邮件列表等公告措施，加快、协调技术人员之间的信息交流和对蠕虫攻击的对抗工作。

## 9.4.3 蠕虫的机理与防治

### ● (2) 蠕虫的防治周期

- 蠕虫的防治周期可分为4个阶段。
- 1.预防阶段。**在利用某个漏洞进行攻击的蠕虫产生之前，积极主动地升级系统、安装防火墙、安装入侵检测系统等。
- 2.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。
- 3.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。
- 4.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。

### ● (3) 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。

## 9.4.4木马的机理与防治

- 9.4.4木马的机理与防治

- 1. 木马的定义

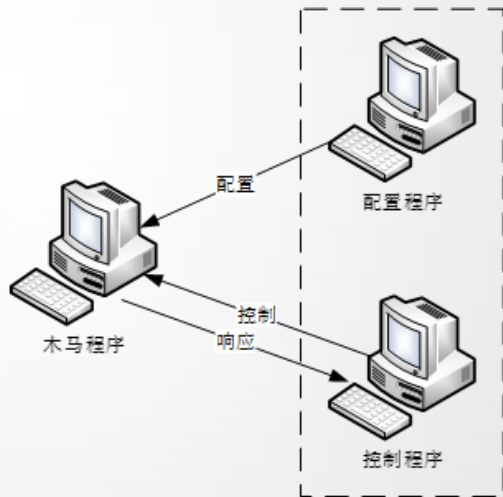
- 木马是一种恶意程序，是一种基于远程控制的攻击工具，它一旦入侵用户的计算机，就悄悄地在宿主计算机上运行，在用户毫无觉察的情况下，让攻击者获得远程访问和控制系统的权限，进而在用户的计算机中修改文件、注册表、控制鼠标、监视/控制键盘，或窃取用户信息，乃至实施远程控制。它是攻击者的主要攻击手段之一，具有隐蔽性和非授权性等特点。



## 9.4.4木马的机理与防治

### ● 2. 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3个部分组成，如图所示。



## 9.4.4 木马的机理与防治

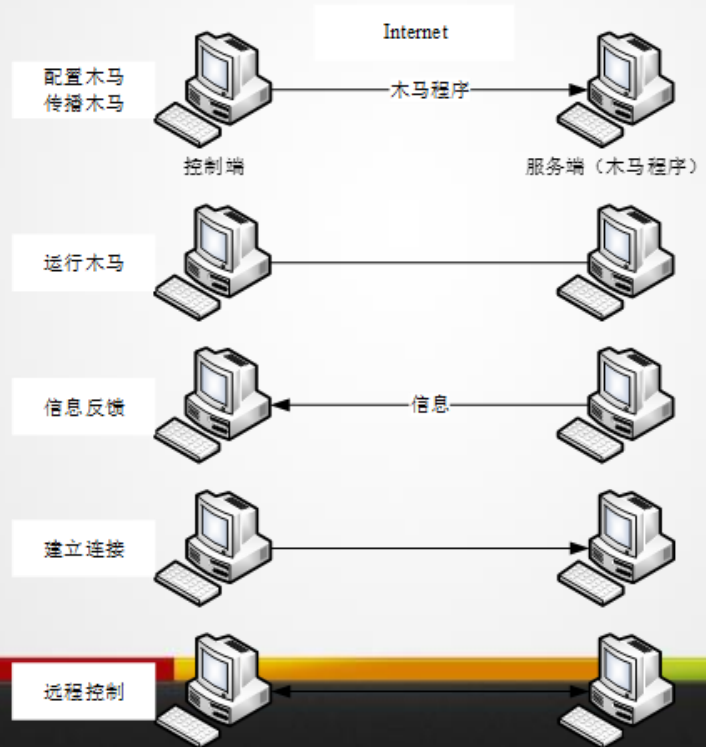
- **(1) 木马程序。**也称为服务器程序，驻留在受害者的系统中，非法获取其操作权限，负责接收控制指令，并根据指令或配置发送数据给控制端。
- **(2) 木马配置程序。**木马配置程序设置木马程序的端口号、触发条件、木马名称等，使其在服务器端隐藏得更隐蔽。
- **(3) 木马控制程序。**控制程序控制远程木马服务器统称为控制端（客户端）程序，负责配置服务器、给服务器发送指令，同时接收服务器传过来的数据。

## 9.4.4木马的机理与防治

### ● 3. 木马的基本原理

- 多数木马包括客户端和服务端两个部分，即采用服务器/客户端结构。
- 攻击者要利用客户端控制服务器，通常是需要先建立一个通信连接。建立木马连接，需要知道木马的计算机的IP地址，这个可以通过端口扫描来搜寻，因为木马服务器端会开放一些特殊的端口。
- 一旦客户端的扫描功能发现有些IP地址的特定端口处于开放状态，说明这些IP地址的机器可能中了木马。
- 获取木马服务器信息后，建立服务端和客户端的连接，控制端便可以进行一系列远程控制了。

- 4. 木马实施网络入侵的基本步骤
- 用木马入侵网络，通常包括6个步骤，如图所示。



## ● 1.配置木马

实现木马隐藏，设置信息反馈方式等

## ● 2.传播木马

## ● 3.运行木马

## ● 4.信息反馈

收集对方机器的软硬件信息

## ● 5.建立连接

（木马连接的建立必须具备两个条件：第一，服务端已经安装了木马程序；第二，控制端、服务端都在线。由于木马端口是事先设定的，所以如何获得服务端的IP地址就更加重要。这主要有两种方法：IP扫描和信息反馈。）

## ● 6.远程控制

木马连接建立后，控制端和木马端之间将会出现一条通信通道。控制端程序可通过这条通道以及木马程序对服务器端进行远程控制。

## 9.4.4木马的机理与防治

### ● 5. 木马的传播方式

- 传统木马的传播方式包括以下几个方面：
  - （1）以邮件的附件形式传播。
  - （2）通过聊天工具传播。
  - （3）通过软件下载网站传播。
  - （4）通过一般的病毒和蠕虫传播。
  - （5）通过带木马U盘和光盘传播。

# 第9章 软件安全与恶意代码

## 参考文献

- ✓ G. McGraw. 软件安全:使安全成为软件开发的必需部分[M]. 北京: 电子工业出版社, 2008.
- ✓ The MITRE Corporation. Common Vulnerabilities and Exposures[DB/OL]. <http://cve.mitre.org>, 2015.
- ✓ G. McGraw, Bruce Potter. Software Security Testing[J]. IEEE Security & Privacy, 2004, 2(5): 81 -85.
- ✓ McGraw G. Building Secure Software: Better than Protecting Bad Software[J]. IEEE Software, 2002, 19(6): 57 -59.
- ✓ CERT. Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) Framework[EB/OL]. <http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=13473>, 1999.
- ✓ F. Swiderski, W. Snyder. Threat Modeling[M]. Redmond, WA: Microsoft Press, 2004.
- ✓ G. Hoglund, G. McGraw. Exploiting Software: How to Break Code[M]. Boston, MA: Addison-Wesley, 2004.
- ✓ 王清, 张东辉, 周浩, 王继刚, 赵双. 0day安全:软件漏洞分析技术(第2版)[M]. 北京: 电子工业出版社, 2011.
- ✓ Michael Howard, Steve Lipner. 软件安全开发生命周期[M]. 北京: 电子工业出版社, 2008.
- ✓ Ian Sommerville. 软件工程(第9版)[M]. 北京: 机械工业出版社, 2011.
- ✓ G. McGraw, B. Chess, S. Migue. Software [In] security: The Building Security In Maturity Model (BSIMM)[EB/OL]. <http://www.informit.com/articles/article.aspx?p=1332285>, 2009.