# 第二十章 软件开发安全

## 20.1 系统开发控制概述

为了防范漏洞，在整个系统开发的声明周期引入安全性是至关重要的

### 20.1.1 软件开发

软件开发项目的初期，给系统构建安全性比现有系统中添加安全性容易得多

1. 编程语音：
    - 汇编语言
    - 高级语言
    - 解释性语语言
    - 语言优缺点：
        - 编码代码通常不易被第三方草种，也更容易在编译代码中嵌入后门和其他安全缺陷并逃避检查
        - 解释型语言不易插入代码，但是任何能够更改编程人员的指令，都可能在解释性语言中添加恶意代码
2. 面向对象编程
    - 消息：对象的通信或输入
    - 方法：定义执行响应消息操作的内部代码
    - 行为：通过方法处理消息的结果
    - 类：对象行为的一组对象的公共方法的集合
    - 实例：包含对象方法的类的实例或例子
    - 继承：某个类的方法被另一个子类集成时就出现继承性
    - 委托：某个对象请求转发给另一个对象或委托对象，如果对象没有处理特定消息的方法，就需要委托
    - 多态性：当外部条件变化时，允许以不同的行为响应相同的消息或方法
    - 内聚：相同类中方法目的之间关系的强度
    - 耦合：对象之间的交互级别
3. 保证
    - 保证过程只是据此在系统生命周期内构件信任的正规过程
4. 避免和缓解系统故障
    - 输入验证：核实用户提供的值是否匹配程序员的期待，通过代码确保数字落在一个可接受的范围，称为限制检测，输入验证应该存在于事物处理的服务器端
    - 故障防护和应急开发
        - 当系统故障时有两个基本选择

        - 将系统置入高级别安全性，直至管理员能够诊断问题并将系统还完至正常状态
        - 应急开放环境允许用户绕开失败的安全控制，此时用户获得特权过高
    - 一旦出现安全防护操作，编程人员就应该靠接接下来的活动

## 20.1.2 系统开发生命周期

系统或应用程序的整个生命周期内斗进行计划和管理，安全性是最有效的，安全开发生命周期的动作

1. 概念定义：为系统创建基本的概念声明即由所有利益相关方协商的简单生命，规定了项目用途以及系统答题需求 2. 功能需求确定：具体的系统功能被累出来，开发人员开始考虑系统的这部分应当如何相互协作以满足功能需求 3. 控制规范的开发：从许多安全角度对系统进行分析
2. 设计评审：设计人员正确的确定系统不同部分将如何相互操作以及如何布置模块化的系统结构
3. 代码审查走查：项目经理安排代码审查走差会议，以寻找逻辑流中的问题或其他设计/安全性缺陷
4. 用户验收测试：大多数组织由开发人员执行系统的初始测试，从而找出明显的错误
5. 维护和变更管理：任何代码的变更都要通过正式的变更流程来进行

## 20.1.3 生命周期模型

1. 瀑布模型：有7个开发阶段，每个阶段完成后，项目进入下一个阶段，瀑布模型的主要批评是，只准许开发人员后退一个阶段，也没对开发周期后期发现错误做出响应规定
2. 螺旋模型：允许瀑布模型处理过程多次反复
3. 敏捷软件开发：强调客户需求的和快速开发的新功能，并以迭代的方式满足这些需求
4. 软件能力成熟度模型：主张所有从事软件开发的组织都一次经历不同的阶段，SW-CMM的不同阶段
    - 初始级： 几乎没有或完全没有定义组织开发过程
    - 可重复级：出现生命周期管理过程，开始有组织的重用代码
    - 定义级：依照一系列正式的、文档化的软件开发过程进行操作
    - 管理级：定量衡量被用来获得对开发过程的详细了解
    - 优化及：采用继续改进的过程，成熟的软件开发过程已被确立
5. IDEAL模型：IDEAL模型的5个阶段：
    - 启动：概述更改的业务原因，为举措提供支持
    - 诊断：工程师分析组织的当前状态，并给出一般性建议
    - 建立：组织采用诊断阶级的一般建议并开发帮助实现这些更改的具体功动作计划
    - 行动：组织开发解决方法，随后测试、改进和实现解决方法
    - 学习：组织不断分析齐努力的结果，从而确定是否已实现期望的目标

### 20.1.4 甘特图与PERT

- 甘特图是一种显示不同时间项目和调度之间相互关系的条形图，提供了帮助计划、协调和跟踪项目特定任务的调度图表
- 计划评审技术（PERT）用于直接改进项目管理和软件编码

### 20.1.5 变更和配置管理

- 变更的基本组件
  - 请求变更： 提供了一个有组织的框架，在框架内，用户可以请求变更，管理者可以进行成本/效益分析，开发人员可以优化任务
  - 变更控制： 重新创建用户遭遇的特定情况并且分析能够进行弥补的适当变更
  - 发布控制： 通过发布控制过程来进行发布认可，符合并确定更改过程中作为编程辅助设计插入的任何代码
  - 配置标识：管理员记录整个组织范围内的软件产品的配置
  - 配置控制：确保对软件版本的更改要与更改控制和配置管理的策略一致
  - 配置状态统计：用于跟踪所有发生的授权更改的正规过程
  - 配置审计：进行定期的配置审计能够确保实际的生产环境与统计环境一致，确保没有发生未授权的配置更改

### 20.1.6 DevOps方法

- DevOps与敏捷开发方法紧密配合，旨在显著缩短开发、测试和部署软件更改所需的时间

### 20.1.7 应用编程接口

- API允许应用程序开发人员绕过传统的网页，并通过函数调用直接与底层服务进行交互
- API必须进行彻底测试安全缺陷

### 20.1.8 软件测试

- 组织内部分发任何软件之前都应当对其进行彻底测试，测试的最佳实践是设计模块之时
- 软件测试时，应该测试软件产品如何处理正常和有效的输入数据，不正确的类型、越界值以及其他界限和条件
- 测试软件时，应该应用于组织其他方面使用的相同的责任分离规则
- 软件测试方法
  - 白盒测试：检查程序的内部逻辑程序，并逐行执行代码，检测是否有存在的错误
  - 黑盒测试：提供广泛的输入场景和查看输出，从用户角度检测程序
  - 灰盒测试：从用户角度处理软件，分析输入输出，但是也会看源代码

- 测试模式：
  - 动态测试： 在运行时环境中评估软件的安全性
  - 静态测试： 分析源代码或变异的应用程序来评估软件的安全性，不需要运行软件

### 20.1.9 代码仓库

- 代码仓库是促进软件开发的出色协作工具，但他们也有自己的安全风险

### 20.1.10 软件等级协议

- 软件等级协议，是被服务提供商和服务供应商都任何的确保组织内部或外部客户提供服务并保持适当服务水平的一种方法

### 20.1.11 软件采购

## 20.2 创建数据库和数据仓储

### 20.2.1 数据库管理修通的体系结构

1. 层次数据库和分布式数据库
   - 层次数据库模型将关联的记录和字段组合成一个逻辑树结构，迎接是一对多
   - 分布式数据库模型将数据存储在多个数据库中，分布式数据库的映射是多对多
2. 关系数据库
   - 关系数据库是由航和列组成的平面二维表，行列结构提供一对一数据映射关系
   - 行的数量被称为技术，列的数量被称为度，关系的域是一组属性可以采用的允许值
   - 数据库的三种键
     - 候选键：用于唯一标识表中记录的属性子集
     - 主键： 唯一标识表中记录的键被称为主键
     - 外键：用于强制在两个表之间建立关系
   - 所有的关系数据库使用一种标准语言，即结构化查询语言（SQL）从而为用户存储、检测和更改数据
   - SQL为管理员、开发人员和终端用户为数据库交互提供了必须的完整功能

### 20.2.2 数据事物

- 关系数据库支持事物的显性和隐形使用，从而确保数据的完整性
- 所有数据库事务都具有5个必需的特征：
  - 原子性：数据库事务必须是源自，也就是说必须是要么全有，要么全无的事物
  - 一致性：所有事物都必须在于数据库所有规则一致的环境中开始操作
  - 隔离性： 要求事物彼此之间独立操作
  - 持久性：数据库事务必须是持久的，也就是说一旦提交给数据库，就会保留下来

### 20.2.3 多级数据库的安全性

多级安全性数据库包含大量不同分类几倍的信息，必须分配给用户的标签进行验证，并且根据用户的请求只提供适当的信息

管理员会通过部署可信前端为旧式或不安全的DBMS添加多级安全性

1. 并发性： 是一种预防性的安全机制，该机制试图使数据库中存储的数据时钟是正确的，或至少使其完整性和可 用性收到保护
2. 其他安全机制：
    - 语义完整性：确保用户的动作不会违反任何结构上的规则
    - 时间和日期来标记和维护数据的完整性和可用性
    - 在数据库内能够细粒度的控制对象，如内容相关的访问控制
    - 管理员可以使用数据库分区技术来防止聚合、推理和污染漏洞
    - 多实例针对某些推理攻击类型的方法措施
    - 利用噪音和干扰在BDMS中插入错误的或欺骗的数据，从而重定向或阻扰信息机密性攻击

### 20.2.4 ODBC

- 开放数据库互连（ODBC）：不必针对交互的每种数据库类型直接进行编码的情况下，允许应用程序与不同的数据库类型通信

## 20.3 存储数据和信息

### 20.3.1 存储器的类型

- 主存储器： 由系统CPU可以直接访问的主要存储资源组成，RAM，一般是系统可以使用的性能最高的存储资源
- 辅助存储器：由许多廉价的，非易失性的、可供系统长期使用的存储资源组成，如磁带、磁盘、硬盘、CD/DVD存储器
- 虚拟内存：系统利用辅助存储器模拟额外的主存储器的资源，这位多种应用程序提供了极快的文件系统，但没有提供恢复能力
- 随机访问存储器： 准许操作系统请求介质上的任意位置的内容，RAM和硬盘都是随机访问存储器
- 顺序访问存储器： 从头到指定地址对整个介质进行扫描
- 易失性存储器：资源断电时会丢失上面的存储内容
- 非易失性存储器：不依赖电源的供电来维持存储内容

### 20.3.2 存储器威胁

- 两种针对数据存储器的威胁

    - 无论正在使用哪种类型的存储器，都存在对存储器资源的非法访问
    - 隐蔽存储通道准许通过直接或间接的操作共享存储介质

# 20.4 理解基于知识的系统

## 20.4.1 专家系统

- 专家系统试图具体化人类在某个特殊学科积累的知识，并且以一致的方式将他们应用于将来的决定
    - 知识库： 包含专家系统已知的规则
    - 推理引擎：对知识库中的信息进行分析，从而得到正确的决策
- 专家系统的优点： 决策不涉及情绪影响

## 20.4.2 神经网络

- 建立互相插入和最终合计生成预期输出结果的计算决策长链

## 20.4.3 决策支持系统

- 决策支持系统（DSS）：一种知识型应用，分析业务数据并且以更容易做出业务决策的形式提供给用户

## 20.4.4 安全性应用

- 专家系统和神经系统的主要优点是：快速做出一致决策的能力