



# 软件安全

徐剑

信息安全系

[xuj@mail.neu.edu.com](mailto:xuj@mail.neu.edu.com)

**Software Collge of NEU**

# 第3章 软件安全需求与设计



软件安全需求概述

软件安全需求内容

软件安全需求获取方法

软件安全策略

软件安全设计原则

软件安全设计方法-威胁建模



# 软件安全需求概述



# 软件安全需求分析的重要性

- 软件的安全性问题往往都是由于没有获得充分的安全需求而造成的（例如，没有考虑如何处理各种特殊情况或者考虑不够充分等）
- 安全需求分析阶段的小改进都会极大地降低未来安全维护的成本
- 软件安全需求的确定与准确描述是确保软件安全的关键性因素，并将直接影响软件安全生命周期后续的其他阶段。



# 软件安全需求过程中存在的问题

- 混淆安全功能和安全需求的关系，将口令保护、身份认证、访问控制、权限管理等安全功能认为是安全需求
- 在软件功能全部完成后，才考虑安全问题
- 将软件安全需求与软件的具体功能需求割裂，没有将软件安全需求与软件功能需求集成（例如数据表设计时为考虑相关字段需要加密）
- 软件安全需求通常被混同于一般功能性描述中，缺乏独立、规范和明确的描述，缺少对于功能、性能与安全需求之间相互关系的描述机制
- 没有形成从攻击者角度来考虑安全问题的理念，没有真正掌握安全需求分析方法和技术手段

# 软件安全需求定义

- 软件安全需求是软件需求的一个必要的组成部分，它描述了为了实现软件安全目标，软件应该做什么，才能有效地提高软件产品的安全质量，减少软件安全问题数量，降低软件安全问题严重性
- 软件安全需求通过提供一组清晰的**安全属性要求和预期行为规范**，来保护系统资产（数据、文件等）免受未经授权的访问以及来自外部攻击者的攻击，例如拒绝服务、身份窃取、病毒等攻击



# 软件安全需求定义

## ● 相关概念

- **安全目标**：从威胁的角度来说，也称为**安全需求**（threat target），是系统包含的**资源**和**信息**。
- **安全漏洞**：系统的弱点或缺陷，例如设计中存在的缺陷或编码中存在的Bug，如果安全漏洞被攻击者利用，则会违背资产的安全目标。
- **威胁**：对资产造成危害的潜在事件，如果这个事件变成攻击，将造成严重的后果。
- **资产**：应对威胁的意图说明，以满足系统的安全需要。
- **安全需求**：提供功能性需求的约束和预期行为的规范，指出系统应该做什么来满足安全目标，消减资产中存在的安全漏洞。在安全需求的语义描述中应避免“如何去做的出现”。



# 软件安全需求的特点

- 软件安全需求是由系统的客观属性所决定，而不是从用户的兴趣出发（这是与其他需求不同之处，安全需求是伴生需求并带有一定的强制性）
- 软件安全需求是一个**主动式防御**过程
  - 主动获取系统的安全需求，将安全需求作为功能需求过程的一部分
  - 首先进行整体安全需求分析，然后对每个明确的软件功能分别进行安全需求分析，使安全需求分析结果成为软件需求分析结果的子集
  - 将安全需求结果作为一项**安全契约**用以指导软件开发后期的设计、实现和测试，使得软件生命周期的各个阶段都有安全保障
  - 主动式防御避免了之前被动式防御（安全漏洞或安全问题出现后来修复的方法）所存在的缺点，达到了安全理念、安全技术和安全方法在软件中完全植入的目的





# 在获取软件安全需求时应考虑的问题

- 确定应用程序的使用者
  - 对使用者来说，安全意味着什么？
  - 对使用者中不同的成员来说，安全是否具有不同的含义？
  - 不同的使用者是否有不同的安全需求？
- 确定应用程序运行位置。
  - 是在Internet上？还是在防火墙之后运行？还是在移动网络中运行？
- 要保护的对象是什么？
- 如果受保护的对象受到损害，这对用户意味着什么？
- 应用程序的管理者是谁？
  - 普通用户还是公司的IT管理人员？
- 产品的通信要求是什么？
  - 是公司内部通信还是外部通信，还是二者兼而有之？
- 操作系统和运行环境可以提供那些可以利用的基础安全服务？

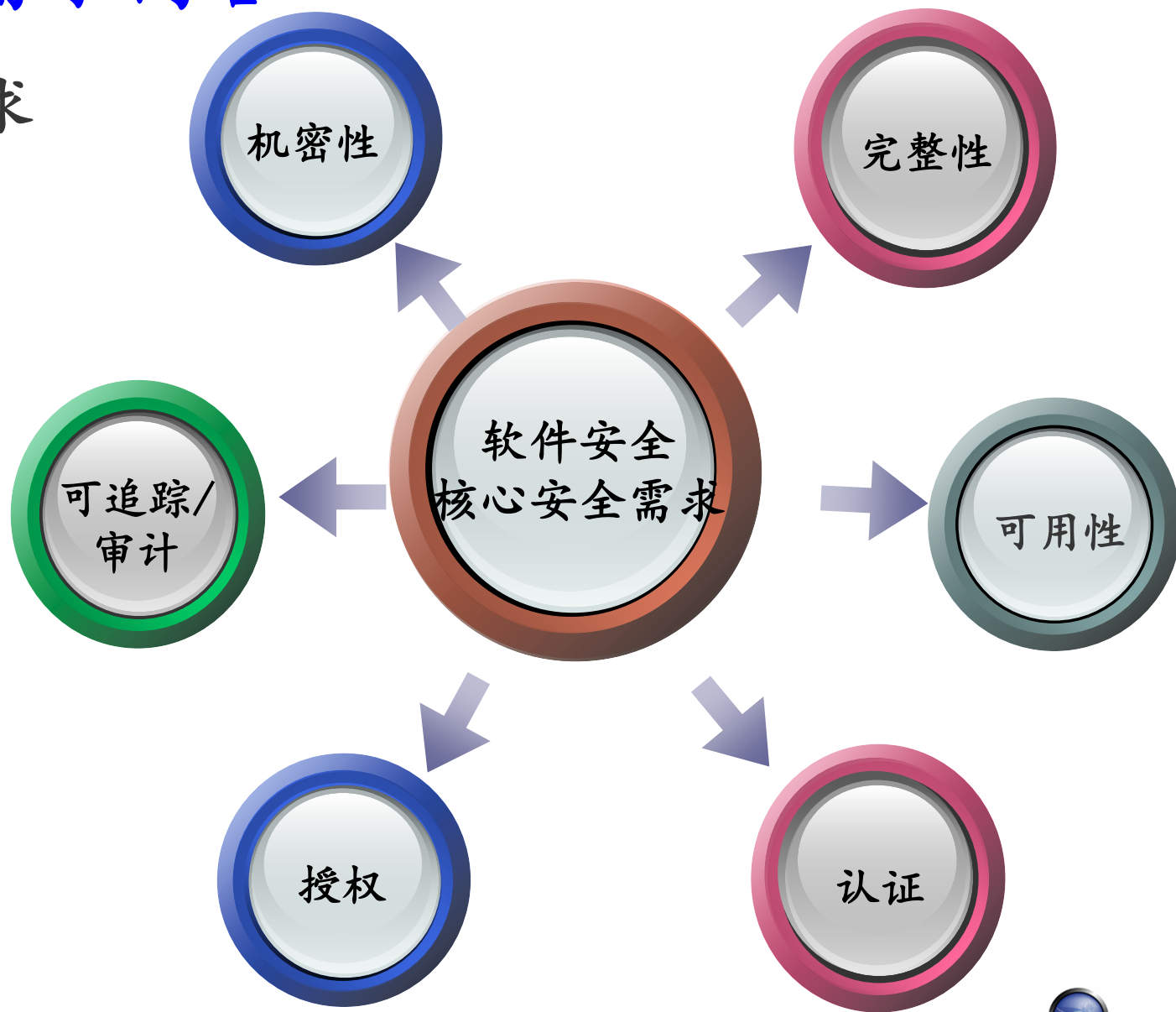


# 软件安全需求内容



# 软件安全需求内容

## 核心安全需求



# 软件安全需求内容

## ● 机密性

- 软件的机密性是指确保与软件相关的**资产**（包括所处理的数据和管理的资源）以及相关**属性特征**（包括运行环境和用户之间的联系），在整个软件生命周期中都不会被未经授权的  
个人、实体或处理过程所获取和访问
- 在软件需求过程中，需要从全生命周期角度，对敏感资产的机密性保护需求进行明确定义和描述
- 以数据为例，在全生命周期中都要考虑机密性保护
  - 数据传输过程
  - 数据处理过程
  - 数据存储过程
- 机密性需求具有**时效性**



# 软件安全需求内容

## ● 机密性需求内容

- 确定那些数据是属于必须得到机密性保护的数据
- 明确口令不能以明文的形式存储在系统的后台
- 将敏感数据，例如支付宝账号信息，在网络上传送时的机密性保护需求
- 日志文件中含有敏感信息，应如何处理
- 当开发或采购软件产品的时候，要充分考虑保护机制的时效性和保护的范围



# 软件安全需求内容

## ● 软件完整性

- 软件的完整性是指，软件要确保与软件相关的资产（主要包括数据及其管理的资源）**不会遭到非法篡改或在遭到篡改后可快速发现**，即，软件以及受其管理的资源必须能够抵御非法篡改并能从中恢复。
- 非法篡改一般由未授权的修改引起，这些修改针对软件源代码、受管理的资源或系统配置等。在软件开发过程和运行中都必须保证其完整性。



# 软件安全需求内容

- 软件完整性需求主要解决的安全问题
  - 系统的可靠性保证
    - 确保系统或软件按照设计和预期的功能那样工作
  - 防止未经授权的修改
    - **系统完整性**：对系统或软件进行修改的保护
    - **数据完整性**：对系统或软件所处理数据的保护，要保证数据在传输或者静止状态下实现完整性控制，防止故意或无意的未经授权的操作



# 软件安全需求内容

## ● 完整性需求内容

- 所有**输入的表单**和**参数**在被软件处理之前，都需要根据允许的输入数据集进行比较验证；
- 发布的软件都应该具备**校验和以及散列函数**的功能，以便用户可以验证其完整性；
- 所有的**非实体用户（具体的人）行为**，如系统和批处理程序都**需要被识别和监控**，以防止它们对运行的系统数据进行操作，除非它们有明确的授权；
- 在决定要**开发或购买的软件产品**的完整性需求时，必须考虑系统和数据的可靠性、准确性、完备性和一致性要求。





# 软件安全需求内容

## ● 软件可用性

- 软件的可用性是指，获得授权的实体可以根据需要访问和使用软件的属性，即软件必须按照服务水平协议（Service Level Agreement, SLA），对其授权用户（人或是进程）提供满足要求的相关服务。
- 可用性描述了软件在何种程度上能够帮助特定的用户在特定的环境下高效、可靠地实现所设定的预期目标。
- 重要指标
  - MTD（Maximum Tolerated Downtime）最大可容忍的宕机时间
    - 可用性的最低水平
  - RTO（Recovery Time Object）恢复时间目标
    - 快速响应能力

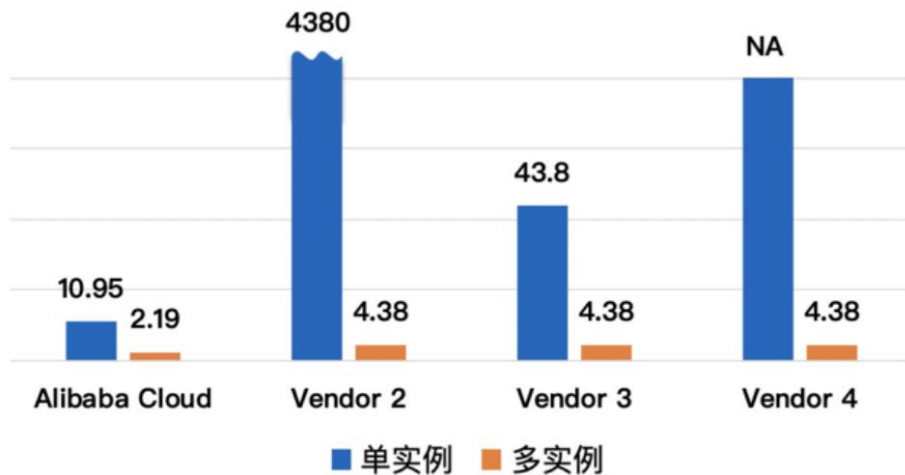


# 软件安全需求内容

## ● 可用性需求内容

Cloud Vendors	单实例可用性SLA	跨AZ多实例可用性SLA
Alibaba Cloud	99.975%	99.995%
Vendor 2	90%	99.99%
Vendor 3	99.9%	99.99%
Vendor 4	Not supported	99.99%

主流云厂商弹性计算实例每月不可用时长(分钟)



来源：2019年12月1日摘自其他相关云厂商的SLA声明文件



# 软件安全需求内容

## ● 认证

- 认证是指对一个实体的声明进行验证的过程，以确定这个实体是否具有其所声称的身份，其中实体可能是一个人、一个过程或一个硬件设备
- 常见的认证方法是实体提供身份声明或身份凭证，由这些凭证的信任源对这些凭证进行验证
- 认证需求就是确保那些提出认证申请的身份的合法性和有效性
- 身份认证凭证可以由不同的因素或因素的组合构成
  - 3W
  - 单因素&多因素身份认证
  - 考虑不同认证方法的优缺点和适用领域



# 软件安全需求内容

## ● 认证需求内容

- 软件只能在局域网环境中部署时，一旦经过身份认证的用户登录到网络，不应该在要求他们提供用户名和密码
- 软件需要为用户提供第三方零售商和供应商的单点登录支持，如果这些身份主体已经记录在供应商预定义的用户认证列表中
- 系统内网和互联网用户都应该能够访问软件
- 对所有财务处理软件，都需要双因素或多因素身份认证



# 软件安全需求内容

- 考虑软件的认证需求时，需要注意的问题
  - 是否需要认证
  - 认证所需的级别
    - 对于X行为来说，单因素认证是否已经足够；
    - 对于X行为来说，双因素认证是否足够；
    - 将控制/保护认证数据库；
    - 将允许外部机构来控制认证数据库；
    - 将允许外部机构来控制部分认证数据库，例如普通登录用户而非管理员；
    - 那些IP范围内的人需要实施认证；
    - 那些物理范围的人需要实施认证。



# 软件安全需求内容

## ● 考虑软件的认证需求时，需要注意的问题（续）

### ● 用户账户的管理

- 是否任何人都可以建立账户；
- 是否任何拥有电子邮箱的人都可以建立账户；
- 是否任何拥有信用卡的人都可以建立账户；
- 是否任何拥有微信/微博/QQ账号的人都可以建立账户；
- 是否只有授权的管理员才可以建立账户；
- 是否要有两个及两个以上的授权的管理员同时参与才可以建立账户；
- 是否任何人都可以随意关闭账户，并尽快删除数据；
- 是否任何人都可以随时关闭账户，并出于商业目的保留数据；
- 是否关闭账户需要管理员参加；



# 软件安全需求内容

## ● 授权

- 授权是在身份认证基础上，根据访问主体的身份和职能为其分配一定的权限，访问主体只能在权限范围内合法访问。
- 授权会为一个经过认证的用户确定所需要的访问权限、优先级以及可执行的动作，即确定一个用户可以做什么和不可以做什么。
- 为了确定授权需求，首先需要确定主体和对象
  - 主体是提出访问请求的实体，可以是用户或系统过程；
  - 对象是主体要实施动作的事务，也被称为客体。
- 在决定授权需求时，对于对象施加的动作也需要明确地获得，软件用户执行的动作是关于数据或信息的时候，通常被称为CRUD操作，即创建（Creat）、读取（Read）、更新（Update）、删除（Delete）。



# 软件安全需求内容

## ● 授权需求分析时需要考虑的问题

- 明确主体和访问对象之间的关系，建立**主体-访问对象控制关系模型**
- 在无法有效建立主体、访问对象映射关系的情况下，可以根据基于资源的访问控制方法，确定访问控制管理模型
- 最小权限原则
  - **主体最小权限原则**：主体仅被授予完成工作所必需的访问权限
  - **角色最小权限原则**：角色仅被授予对资源进行操作所需要的权限
- 基于角色的权限访问控制（Role-Based Access Control, RBAC）模型进行授权决策时，必须确保职责分离SoD（Separation of Duty）的原则
  - 不可以分配两个互斥的角色





# 软件安全需求内容

## ● 授权需求内容

- 高敏感的秘密文件必须严格限制访问，只有拥有最高许可证水平的用户才能访问；
- 一旦认证成功后，用户不应该被要求每次都提供他们的认证凭证；
- 所有未经身份认证的用户将仅有只读权限，这是访客用户角色的权限；而经过身份验证的用户将默认拥有“读”和“写”的权限，这是普通用户角色的权限；管理员角色成员将拥有普通用户角色的权限，另外拥有操作执行权限



# 软件安全需求内容

## ● 可追踪/审计

- 可追踪确保实体的访问动作可以唯一地被区别、跟踪和记录。
- 可追踪通过审计日志实现。
- 与安全相关的软件行为都必须可追踪和记录，并进行责任归因。
- 追踪过程在这些行为发生时和发生后都要进行，并且软件系统的安全策略中应该指出哪些行为是安全的。



# 软件安全需求内容

## ● 审计需求

- 建立一个用户操作的历史记录，为每一项活动提供一个完整的、可审计的轨迹
- 审计轨迹可以帮助检测未授权用户的修改或授权用户的未经授权的修改，上述两种操作都违反完整性保护需求
- 如果软件可以被适当地记录或跟踪，那么审计不仅可以作为一种检测控制手段帮助计算机调查取证，同时也可以用于错误或异常故障诊断



# 软件安全需求内容

## ● 审计需求内容

- 执行动作的主体（用户或进程）是谁？
- 所执行的动作是什么？
- 动作执行的对象是哪一个？
- 动作执行的时间是什么（时间戳）？
- 所有失败的登录尝试都需要连同时间戳一起被记录下来，并确认该请求的来源；
- 审计日志应总是以附加的方式进行保存，而永远不会被覆盖；
- 必须保证审计日志在一段内被安全保存，例如2年。



# 软件安全需求内容

## ● 审计需求中需要注意的问题

### ● 审计日志的设计与保护

- 严格保护审计日志的配置接口，防止攻击者攻击
- 审计日志是敏感的资产，要审慎地确定日志内容与格式
  - 不记录敏感信息，例如记录认证失败时，则不应记录口令值，如需记录，则应记录口令的哈希值

### ● 其他问题

- 性能影响（频繁记录对性能的影响）
- 负载均衡（安全与性能的平衡）
- 容量限制（占用大量存在空间）
- 安全与可靠性（可用性）的关系



# 软件安全需求内容

## ● 通用安全需求



# 软件安全需求内容

## ● 会话管理

- 会话管理用于状态维护，在身份认证成功的基础上，系统将给用户颁发一个会话标识符（ID），用于跟踪用户行为、维护用户的认证状态，直到会话被终止或者认证状态变为非认证状态
- 会话管理需求确保一旦会话被建立，可以保持在一种不损坏软件安全性的状态下进行有效会话，即，已经建立的会话不容易受到任何威胁的影响，因为它应用了机密性、完整性和可用性的安全策略。
- 如果没有会话管理，用户/过程的每一个访问请求都需要认证，增加用户负担
- 防止会话被攻击者劫持，需要制定会话管理计划
- 会话管理需求确保会话可以抵御强力攻击、可预测攻击以及中间人劫持等攻击



# 软件安全需求内容

## ● 会话管理需求内容

- 每个用户活动都需要唯一地被记录和追踪；
- 一旦经过应用程序的认证，不应该要求用户提供认证凭证；
- 当用户注销或关闭浏览器窗口的时候，会话必须明确停止；
- 用于识别用户会话的标识符不能以明文或容易猜测的方式进行存储和传递。





# 软件安全需求内容

## ● 错误和异常管理

- 不正确的错误和异常信息处理方式会成为破坏软件安全性的潜在因素
- 一些不恰当的错误提示信息和未处理的异常报告会导致内部应用程序体系结构、设计、配置信息的泄露，进而破坏软件的机密性
- 使用正确的错误信息提示和结构化的异常处理方法，可以减少因为错误或异常管理不当所引发的安全问题



# 软件安全需求内容

## ● 错误和异常管理需求内容

- 明确对所有的例外进行的处理、获取和阻断的方式
- 向终端用户显示的错误信息仅包含必要的信息，而不会泄露任何内部系统的错误细节以及用户的个人信息
- 需要对安全例外的细节进行实时监控和周期性的审计

**HTTP Status 404 - /api/**

**type** Status report

**message** /api/

**description** The requested resource is not available.

**Apache Tomcat (TomEE)/7.0.68 (1.7.4)**



# 软件安全需求内容

## ● 错误和异常管理



### 帐号或密码错误

提示:

1. 请检查帐号拼写，是否输入有误
2. 若帐号长期未登录，可能已被注销，请[重新注册](#)
3. 若您忘记密码，请[找回密码](#)
4. 若您需要锁定此帐号，请[点击这里](#)
5. 若手机号码邮箱的手机号已更换，可[找回原帐号](#)

邮箱帐号登录

动态密码登录

netwaera @163.com

•••••

☐ 十天内免登录

[忘记密码了?](#)

登 录

注 册

版本: 默认版本 ▾

正使用SSL登录

网易官方  
邮箱APP

下载邮箱大师

- 3亿春运火车票红包发放中!
- 网易邮箱安全访问公告



# 软件安全需求内容

## ● 配置管理

- 软件配置参数和组成软件的代码需要被防护以防止攻击者的攻击
- 软件配置参数和代码通常在软件运行之前需要进行初始化
- 在软件需求分析和设计阶段，识别并获取这些配置参数对于确定适当的安全保护水平是至关重要的。



# 软件安全需求内容

## ● 配置管理需求内容

- 必须对数据库连接设置、注册表设置和其他敏感的应用程序设置进行必要的安全保护
- 口令等重要信息不能硬编码在程序代码中
- 应用程序或会话的起始和终止事件必须包含对配置信息的安全保护



# 软件安全需求内容

## ● 运维安全需求



# 软件安全需求内容

## ● 运维安全需求

### ● 软件投入使用后，需要关注：

- 硬件环境
- 操作系统环境
- 网络环境
- 运行状态
- 参数设置



- 软件在实际应用中出现的问题，多数与缺少运维安全需求或运维安全需求不完整有关
- 运维安全需求是指，识别软件所需要的安全能力和依赖关系，为业务提供预期的服务功能



# 软件安全需求内容

## ● 运维需求内容

- 应该使用严格的访问控制机制对应用程序之间共享的加密密钥进行保护和维护；
- 对安全日志进行访问控制保护，安全日志的备份和复制操作也要得到保护；
- 及时为软件打补丁和进行漏洞修复，并且只有在获得所有必要的批准之后才能对生成环境进行变更；
- 安全事件的处理必须遵循事件管理流程，必须对事件产生的原因进行分析；
- 软件必须受到监控，以确保它不受到新的威胁影响。





# 软件安全需求内容

## ● 环境部署需求内容

- 软件部署的网络环境，是Internet还是Intranet？
- 部署在何种硬件和操作系统环境之下？
- 软件是否部署在非军事区（DMZ）？
- 采用的网络协议和端口是什么？
- 在实际环境中有哪些优先权？
- 软件是否负载平衡？采用何种集群架构？
- 软件是否需要单点登录（SSO）支持？
- 是否可以利用现有的操作系统事件日志进行安全审计？



# 软件安全需求内容

## ● 归档需求内容（保持业务连续性）

- 归档的数据将要存在哪里？
- 归档数据要保存在哪一类事物处理系统中？是远程、在线还是离线存储介质？
- 归档系统需要多大的存储空间？
- 归档数据的检索速度是多少？
- 如何保存归档文件？包括如何防止归档文件被覆盖、被篡改等
- 归档文件的需要存储多长时间？
- 是否存在对归档文件的监管要求？
- 归档文件的存储格式是什么样的？明文还是密文？
- 如果需要密文形式的存储，则如何实现？



# 软件安全需求内容

## ● 反盗版需求

- 软件需要进行数字签名以防止篡改和逆向工程；
- 需要对代码进行混淆，以防止代码被复制；
- 软件许可证的密钥不能静态地硬编码到二进制文件中，因为攻击者可以通过调试和反编译获得密钥；
- 许可证验证检查必须是动态的，不依赖终端用户可以改变的因素。



软件安全  
其他

12起  
GDPI



继上个月在美  
罚1700万欧元

近日，爱尔兰  
的12期数据泄  
全，违反了

## 因造成部分数据泄露，北京市网信办对三家企业作出行政处罚

IT之家 10月30日消息，据“网信北京”，根据国家网信办移交的问题线索，北京市网信办依据《中华人民共和国数据安全法》对属地三家企业涉嫌存在网络数据安全违法行为进行立案调查并作出行政处罚（IT之家注：未公布具体名称）。



据介绍，三家企业违反了《中华人民共和国数据安全法》第二十七条规定，未履行数据安全保护义务，部署的 ElasticSearch 数据库存在未授权访问漏洞，造成部分数据泄露。

北京市网信办依据《中华人民共和国数据安全法》第四十五条第一款规定，对三家企业分别作出责令改正，给予警告，并处5万元罚款的行政处罚，对直接主管人员和其他责任人员处以1万元罚款处罚。

时间/  
-of-

反

ta

兰被

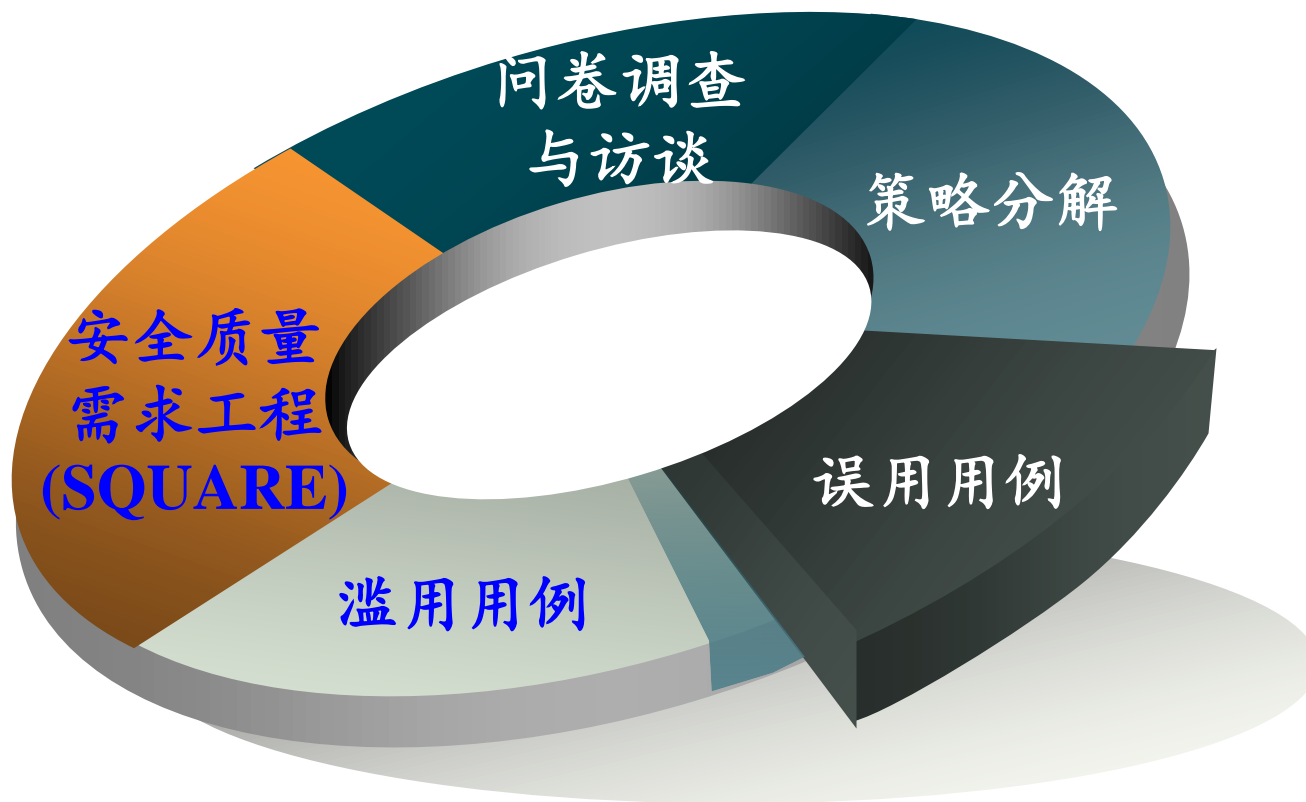
18年  
安

# 软件安全需求获取方法



# 软件安全需求获取方法

- 典型的软件安全需求获取方法



# 软件安全需求获取方法

## ● 问卷调查与访谈

- 问卷调查和访谈的有效性取决于设计面向调查和访谈对象的合适问题
  - 明确那些是具体的确定调查和访谈问题，那些是开放性调查和访谈问题
  - 开放性的调查和访谈问题有助于产生一些重要的软件安全相关信息，而这些信息未必可以在具体的调查和访谈问题中体现出来。
- 软件安全需求问卷调查或访谈应该考虑**业务风险**、**过程风险**、**技术风险**等内容，要覆盖软件安全设计原型、安全配置等内容，并且问题的答案或访谈的结果可以**直接**用于产生软件安全需求。
- 问卷调查可采用在线问卷或E-mail方式，也可以采用面谈方式
- 在开展面谈式的问卷调查过程中，调查人员需要独立、客观地与不同的被调查人员进行充分的沟通和交流，并完整准确地记录下调查结果





# 软件安全需求获取方法

## ● 问卷调查与访谈中的常见问题

- 软件需要产生、传输、存储和管理何种类型的数据？
- 数据的敏感度水平如何？
- 软件是否需要处理与个人身份或隐私相关的信息？
- 软件是否需要处理企业或个人的财务数据？
- 软件是否需要处理涉密信息？
- 软件是否需要对用户的行为进行审计和监控？
- 软件需要提供的认证方式有哪些？
- 用户角色都有哪些？需要何种级别的访问控制机制？
- 用户可以接受的系统宕机时间是多少？
- 当系统出现安全问题时，可以采取的措施有哪些？





# 软件安全需求获取方法

## ● 策略分解

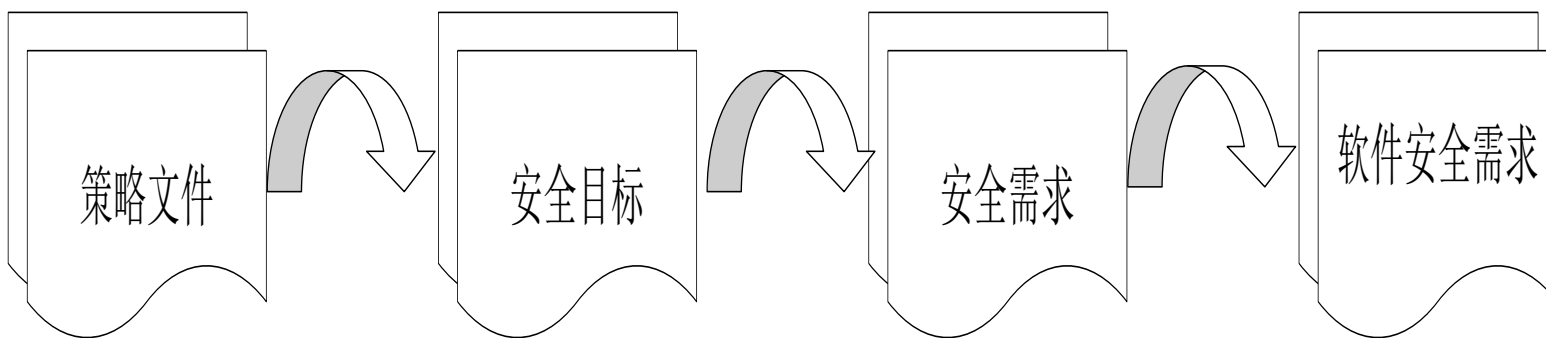
- 安全需求的来源之一是组织需要遵守的内部和外部政策
- 政策属于高层次文件，需要被分解为更加细化的安全需求
- 将高层次的政策要求分解成具体安全需求的过程并不仅限于组织内部的政策，外部法律与法规、隐私和遵从性命令也可以被分解为详细的安全需求
- 为避免混淆，将这些高层次的安全政策统一称为**策略文件**，不管它们是内部的还是外部的



# 软件安全需求获取方法

## ● 策略分解过程

- 策略分解过程是一个连续的和结构化的过程
- 首先，需要将策略文件转化为高水平的安全目标，然后将这些安全目标再分解成安全需求，这是软件安全需求产生的前提。

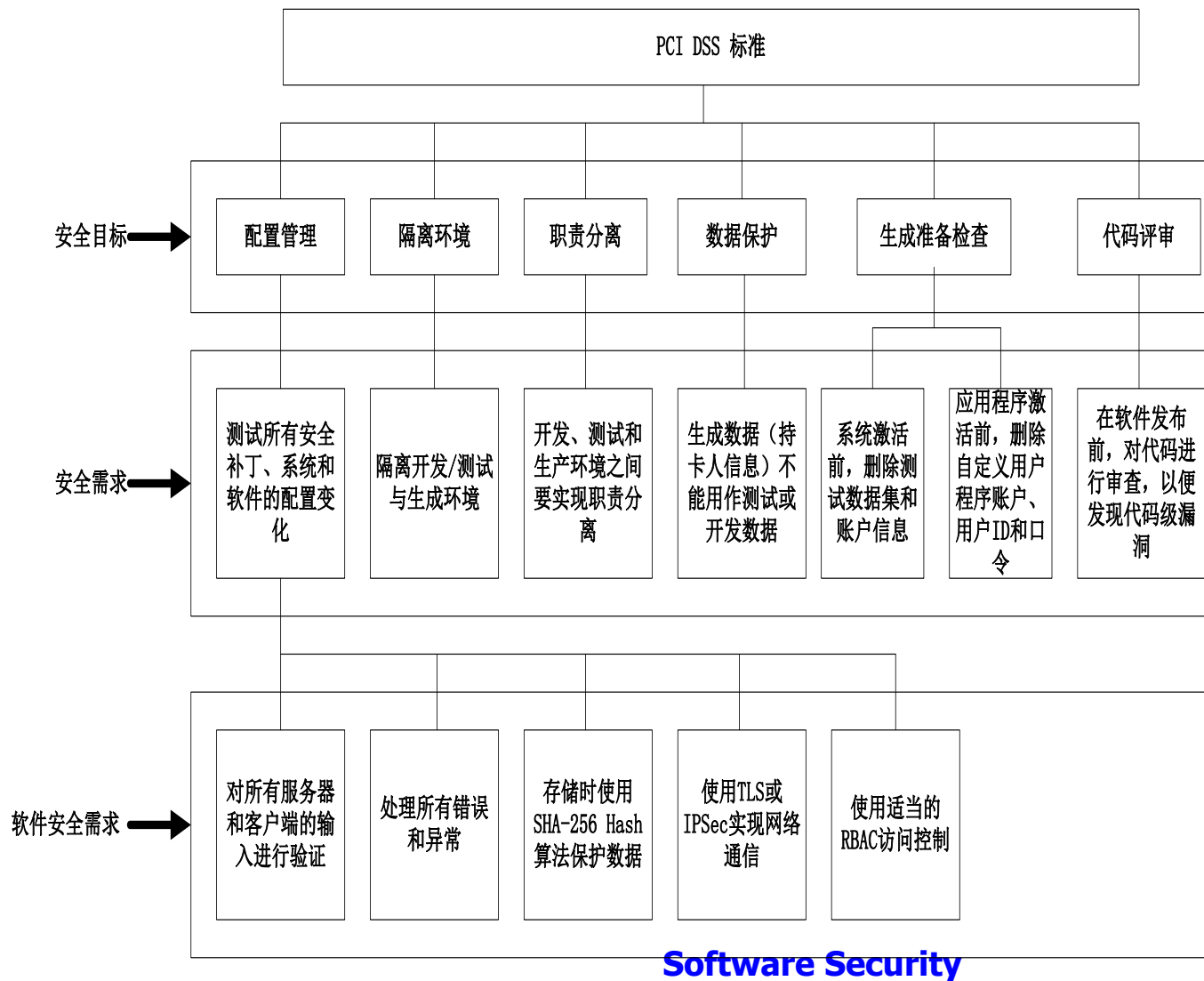


软件安全策略分解过程



# 软件安全需求获取方法

- 策略分解实例-第三方支付行业数据安全标准（Payment Card Industry (PCI) Data Security Standard, PCI DSS）

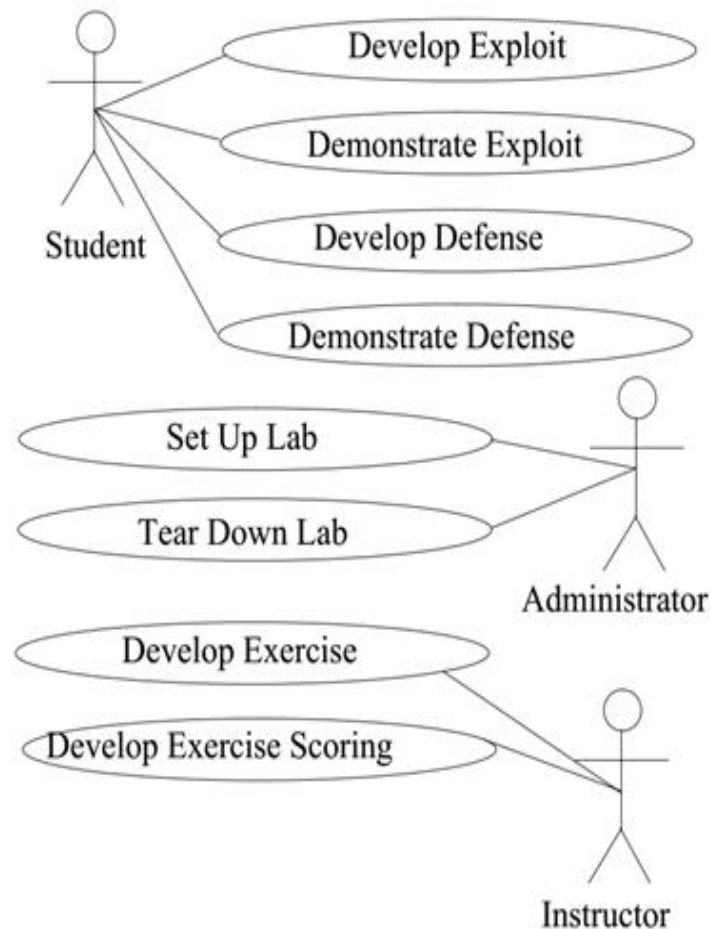


# 软件安全需求获取方法

## ● 误用/滥用用例

### ● 用例

- 用例（Use Case）是软件工程中対系统如何反应外界请求的描述，是一种通过用户的使用场景来获取需求的技术。
- 用例包括确定**动作者**（访问主体）、预期的**系统行为**（使用用例）、执行序列，以及动作者和使用用例之间的关系。
- 动作者可能是一个个体、一个角色或非自然人，例如一个人、管理员或后台批处理过程都可能是一个动作者。
- 用例可以帮助软件开发人员规范地描述软件或系统的预期行为，即，通过用例描述用户的预期行为，而预期行为则描述了完成业务功能所需要的行为和事件的顺序。



基于Internet的信息安全实验室用例图

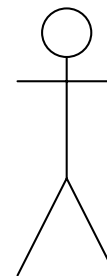


# 软件安全需求获取方法

- 用例中的重要概念和定义

- 用例**是软件产品使用者（参与者）和软件产品本身之间的交互。

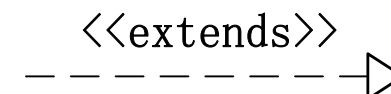
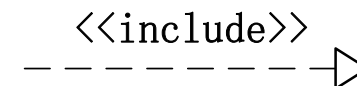
- 参与者（Actor）**：与软件产品发生关联的人，包括软件的使用者，某项业务的发起者和用例中起到关键作用的人。



参与者（Actor）



用例

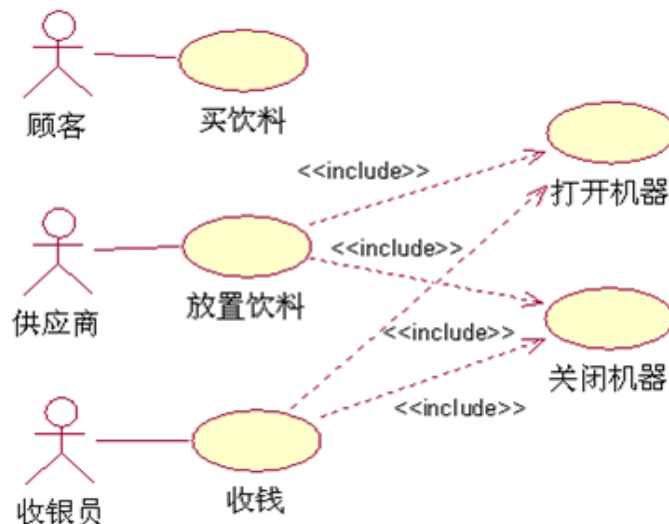
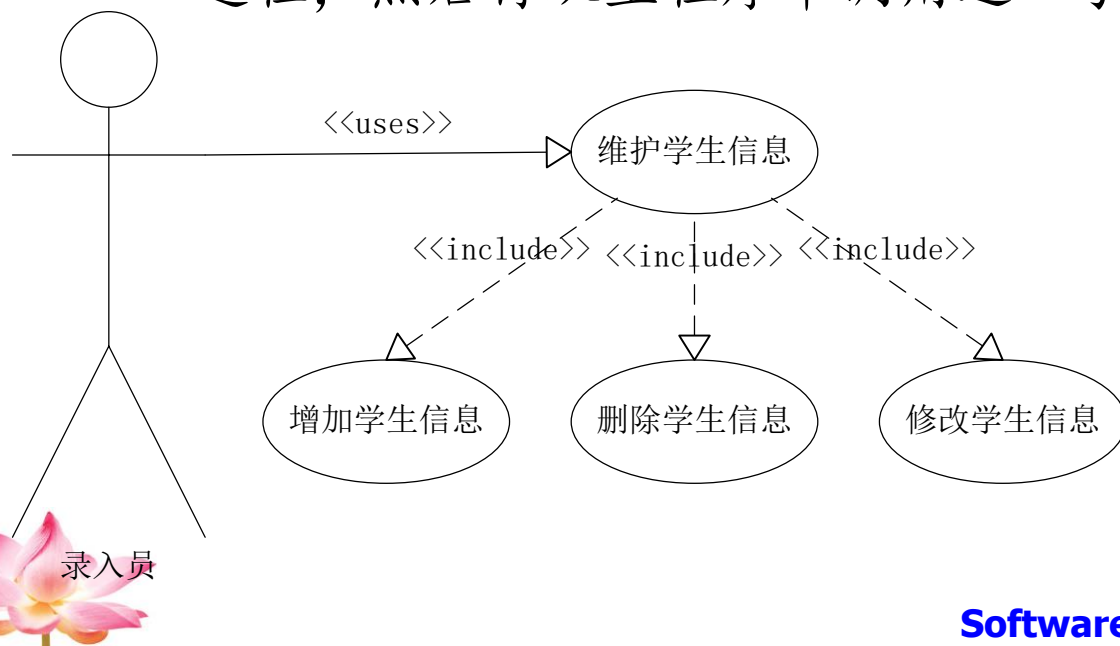


# 软件安全需求获取方法

## UML用例图中包含(include)、扩展(extend)和泛化(generalization) 三种关系

### 包含关系

- 使用**包含 (Inclusion) 用例**来封装一组跨越多个用例的相似动作（行为片断），以便多个基（Base）用例复用
- 类似于在过程设计语言中，将程序的某一段算法封装成一个子过程，然后再从主程序中调用这一子过程

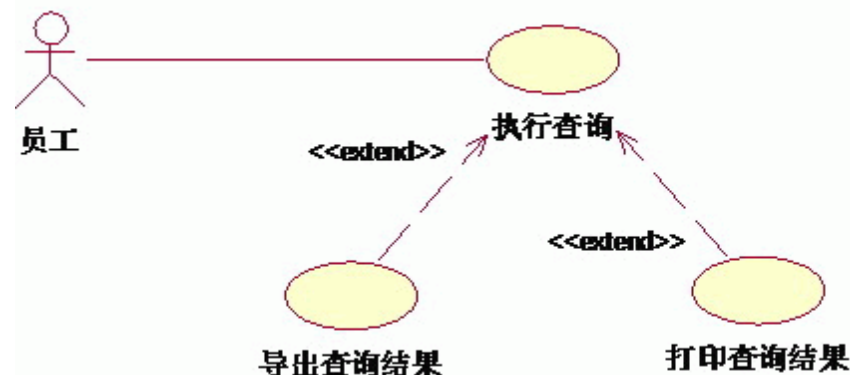
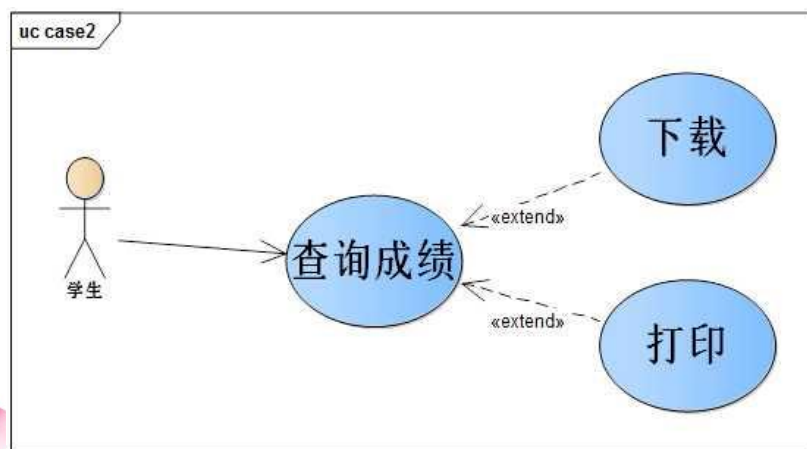


# 软件安全需求获取方法

- UML用例图中包含(include)、扩展(extend)和泛化(generalization)三种关系

- 扩展关系

- 将基用例中一段相对独立并且可选的动作，用扩展（Extension）用例加以封装，再让它从基用例中声明的扩展点（Extension Point）上进行扩展，从而使基用例行为更简练和目标更集中
- 扩展用例为基用例添加新的行为
- 扩展用例可以访问基用例的属性，因此它可以根据基用例中扩展点的当前状态来判断是否执行自己

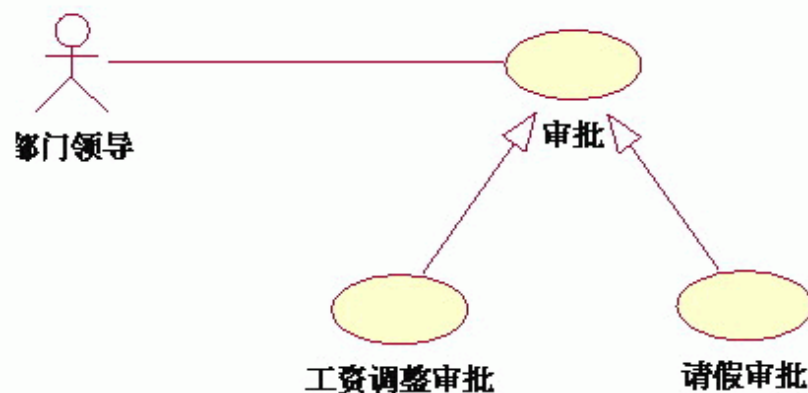


# 软件安全需求获取方法

- UML用例图中包含(include)、扩展(extend)和泛化(generalization)三种关系

- 泛化关系

- 子用例和父用例相似，但表现出更特别的行为；子用例将继承父用例的所有结构、行为和关系。
- 子用例可以使用父用例的一段行为，也可以重载它。
- 父用例通常是抽象的





# 软件安全需求获取方法

## ● 误用/滥用用例概述

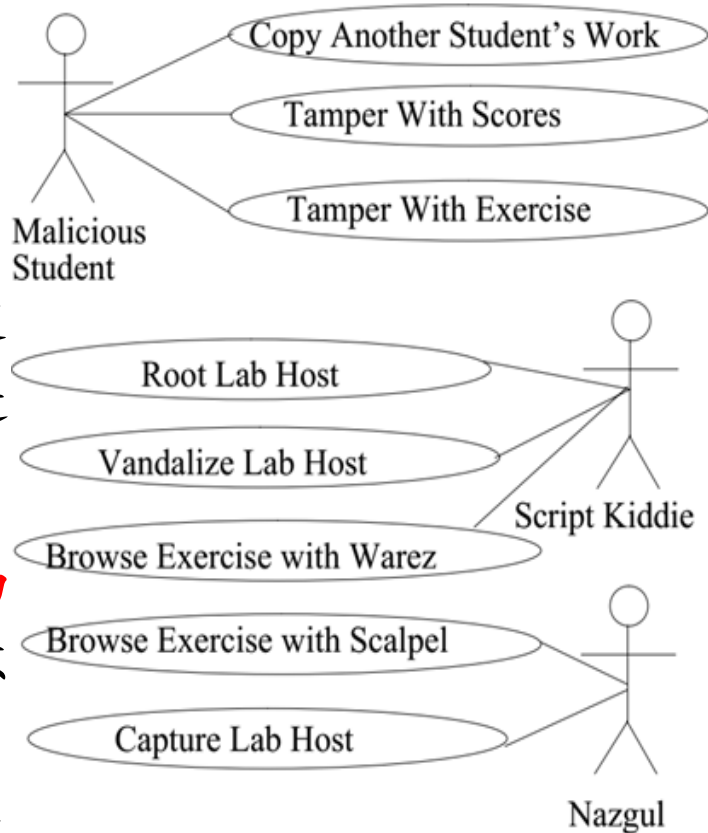
- 传统用例分析方法只能用于分析获取软件系统的功能性需求，无法获取软件的安全需求
- 开发安全的软件，软件开发人员要更多考虑意外或反常的行为，这样才能更好地理解如何创建安全的软件。
- **滥用用例（Abuse Case）和误用用例（Misuse Case）**
  - 对传统用例分析方法的扩展，弥补了传统用例分析方法在获取软件安全需求方面的缺陷
  - 主要思想：帮助软件开发人员将软件置于敌手环境（受攻击状态）中，从攻击者的角度考虑系统面临的威胁，对软件在运行过程中可能面临的非期望的、非标准的情况进行描述性陈述，分析系统存在的安全漏洞，建立威胁用例，针对威胁用例建立安全需求用例，进而达到减少攻击者可利用漏洞的目的。
  - 通过分析软件可能面临的常见攻击，可以有效地捕获误用和滥用用例。



# 软件安全需求获取方法

## ● 滥用用例

- 滥用用例是对系统与一个或多个参与者之间的**具有破坏性**的交互行为的描述，即，交互行为的结果对系统、某个参与者或系统利益相关者是有害的。
- 不能仅根据参与者与系统之间的交互来定义滥用行为，而是必须根据交互过程所造成的**实际损害**来对滥用进行定义。
- 滥用用例是在用例基础上，通过对**负面场景**（即不是系统预期的行为，而是一个不希望在正常使用用例情境中发生的动作）进行建模，来捕获攻击者与系统之间的交互所产生的威胁，进而确定安全威胁和安全需求内容。



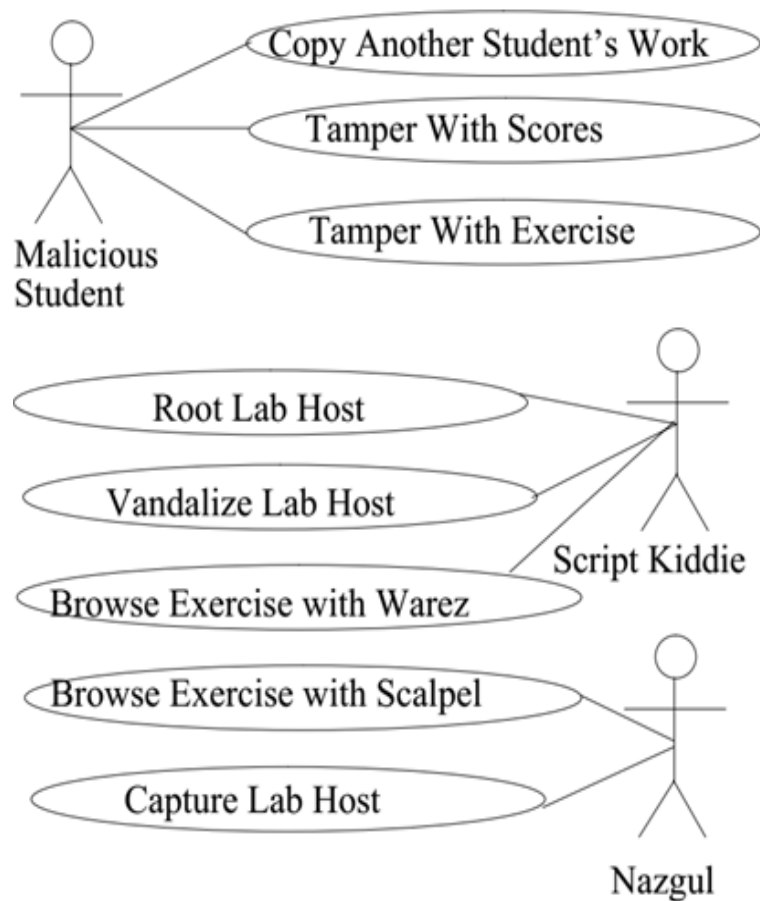
基于Internet的信息安全  
实验室的滥用用例



# 软件安全需求获取方法

## ● 滥用安全例构建过程

- Step 1. 描述参与者和用例。
  - Step 2. 引入主要的滥用者和滥用用例。
  - Step 3. 研究滥用用例和用例之间潜在的**include**（包含）关系。
  - Step 4. 引入新的用例来发现或阻止滥用用例。
  - Step 5. 形成更加详细的需求记录。
- 在滥用用例中没有引入新的术语或特殊符号，所采用的符号与用例中一致
  - 滥用用例图和用例图通常是分开的，即滥用案例不会出现在用例图中，反之亦然



基于Internet的信息安全实验室的滥用用例



# 软件安全需求获取方法

## ● 滥用用例-示例说明

### ● 3个参与者和8个滥用用例

#### ● 恶意学生

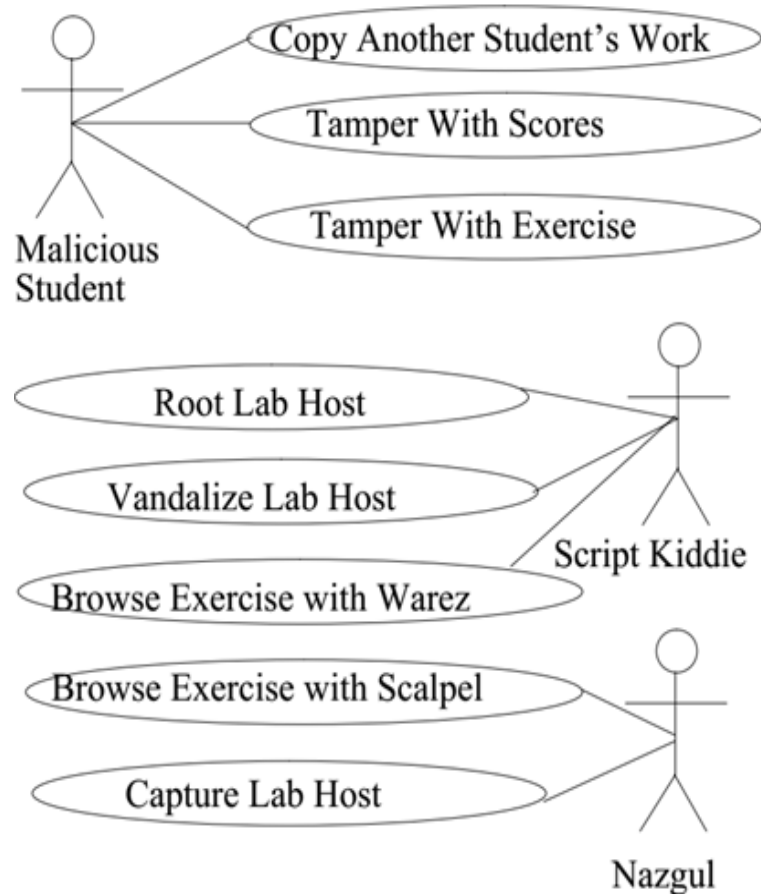
- 拷贝其他同学作业
- 篡改成绩
- 篡改所完成的练习

#### ● 脚本小子

- 获取实验室主机根权限
- 破坏实验室主机
- 利用恶意软件浏览练习内容

#### ● 骇客组织

- 为专门渗透系统而精心设计的攻击
- 控制实验室主机



# 软件安全需求获取方法

## ● 误用用例

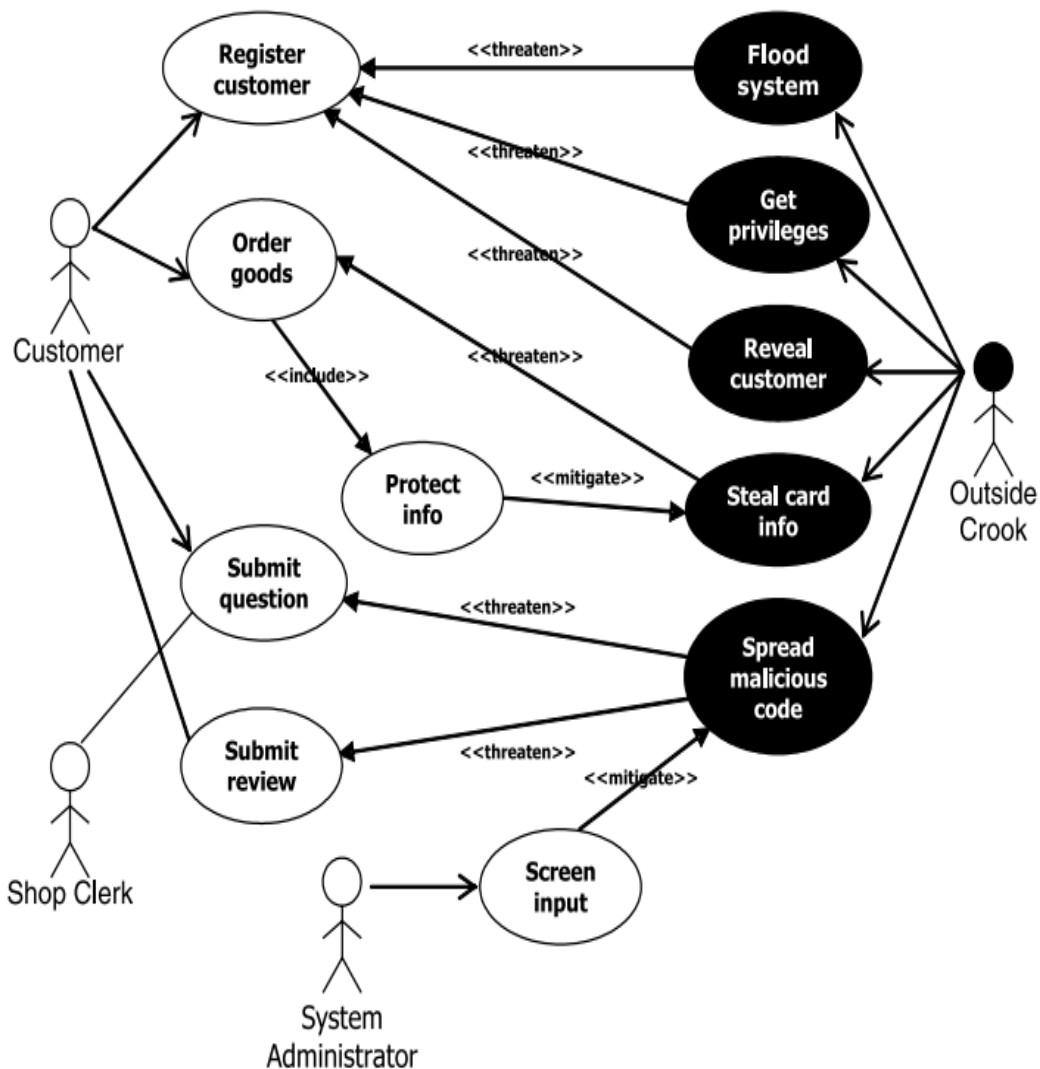
- 误用用例用于描述系统或用户所不希望发生的行为
- 在误用用例中，描述了系统或其他实体可以与误用用户（Misuser）交互执行的一系列动作序列，如果这些动作序列被允许，并执行完成，则会对系统的相关用户造成损害。
- 误用用户是指，有意或无意地引发误用用例的行为人。



# 软件安全需求获取方法

## 误用用例-示例

- 除了标准的“include”和“extends”关系之外，误用用例中引入了一些新的关系，右图所示的误用用例图中的“威胁”和“消减”关系。
- 误用用例和用例既可以是“威胁”关系也可以是“消减”关系。用例可以减轻误用情况，这意味着用例减少了误用用例成功的机会。
- 误用案例也可以威胁用例，即用例被误用用例利用或阻碍



# 软件安全需求获取方法

## ● 误用用例的构建过程

- Step 1. 关注正常用户和他们引起的用例，例如，不考虑任何安全问题的用户需求的服务。利用UML方法建议的正常方式描述角色（用户）和用例。
- Step 2. 引入主要的误用用户和误用案例，例如可能发生的威胁。
- Step 3. 分析误用用例和用例的潜在关系。这一步非常重要，因为系统的许多威胁很大程度上可以通过使用系统的正常功能来实现。
- Step 4. 引入新的用例来“消减”误用用例。
- Step 5. 继续编写更详细的需求文档。





# 软件安全需求获取方法

## ● 误用用例和滥用用例的区别

### 误用用例& 滥用用例

```
graph TD; A([误用用例&滥用用例]) --> B[滥用用例]; A --> C[误用用例];
```

#### 滥用用例

- 侧重威胁产生的危害后果
- 捕获攻击者与系统之间的交互所产生的威胁及威胁带来的后果易于重新复查错误
- 针对识别出的威胁，单独建立威胁用例，并且建立好的威胁用例并不与功能性用例产生交互，威胁用例仅说明系统面临的安全威胁

#### 误用用例

- 侧重威胁本身无法解决全部的软件安全问题
- 从功能性用例的文本描述中分析出可能存在的安全漏洞并识别出对应的威胁
- 误用用例需要与正常用例交互



# 软件安全需求获取方法

- **安全质量需求工程（Secure Quality Requirements Engineering, SQUARE）**
  - 卡耐基梅隆大学（Carnegie Mellon University）提出的一种在软件开发项目中提取和确定安全需求优先级的过程模型
  - SQUARE为信息系统及应用的安全需求提供了一种启发、分类和排序的方法。
  - SQUARE在软件开发生命周期的早期阶段就植入安全概念，并且可以与软件需求工程实践紧密结合。



# 软件安全需求获取方法

## ● SQUARE中的处理过程

序号	步骤	输入	技术	参加者	输出
1	统一定义	IEEE和其他标准的候选定义	结构化访谈、专题小组讨论	投资人、需求小组	统一的定义
2	确定安全目标	定义、候选目标、业务驱动程序、策略和过程、示例	工作会议、调查、面试	投资人、需求工程师	安全目标
3	开发支持安全需求定义的工件	潜在工件（脚本、误用案例、模板、表格）	工作会议	需求工程师	所需的工件：场景、误用案例、模型、模板、示例
4	进行安全风险评估	误用案例、场景、安全目标	风险评估方法、对组织风险承受能力的预期风险分析，包括威胁分析	需求工程师、风险专家、投资者	风险评估结果
5	选择性启发方法	目标、定义、备选技术、投资人专业知识、组织风格、文化、安全需求等级、成本效益分析	工作会议	需求工程师	选择性启发技术



# 软件安全需求获取方法

## ● SQUARE中的处理过程

序号	步骤	输入	技术	参加者	输出
6	获得安全需求	工件、风险评估结果、可选择的技术	联合应用程序开发(JAD)、访谈、调查、基于模型的分析、检查列表、可重用需求类型列表、文档评审	在需求工程师协助下的投资者	削减后的安全需求
7	将需求按级（系统、软件等）别分类，并判断是否是需求或是其他类型的约束	初始需求、架构	依据一套标准的分类召开的工作会议	需求工程师、其他需要的专家	分类后的需求
8	需求排序	分类后的需求和风险评估结果	优先级排序方法，如Triage, Win-Win	在需求工程师协助下的投资者	排序后的需求
9	需求审查	排序后的需求，候选的形式化检查方法	检查方法，如Fagan，同行评议	检查小组	初始的选择需求，决策过程和依据文档

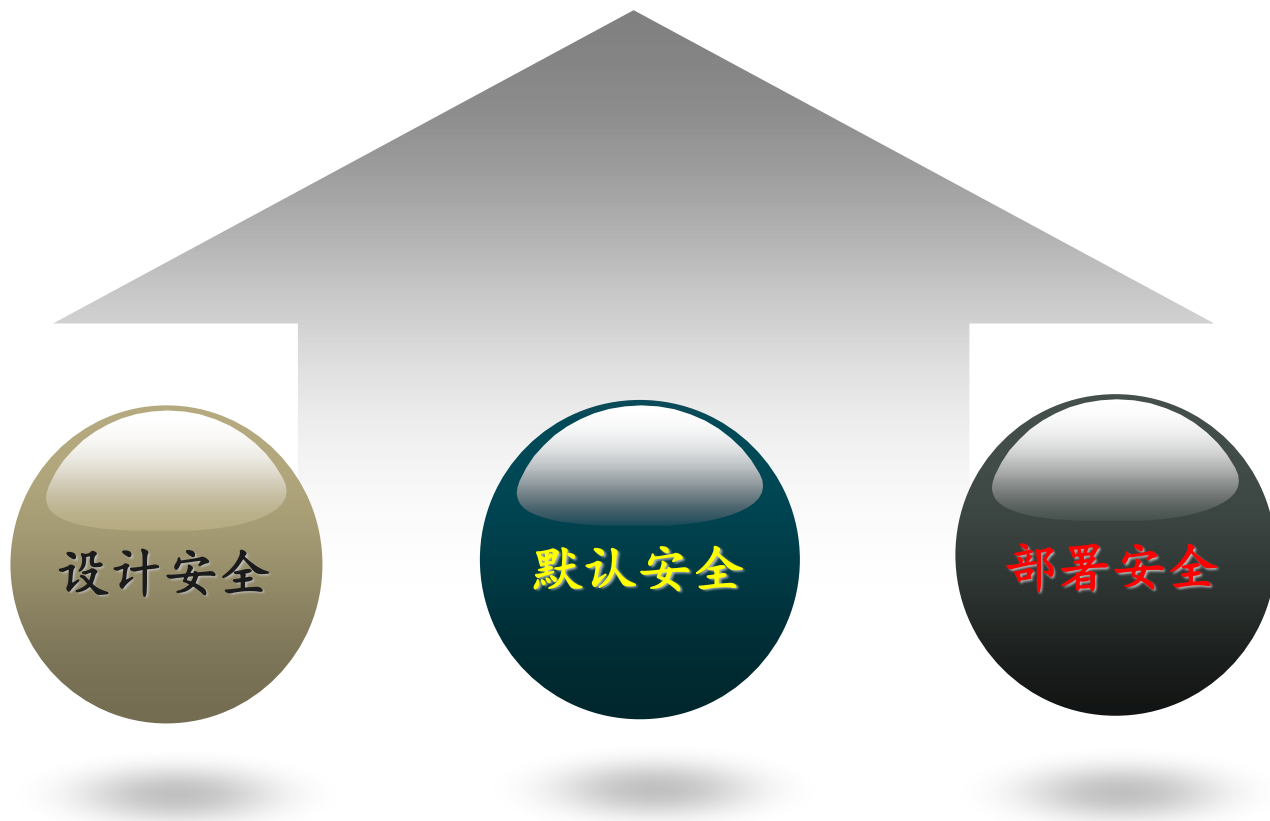


# 软件安全策略



# 软件安全策略

## 软件安全策略 (SD<sup>3</sup>)



# 软件安全策略

## ● 设计安全 (Secure by Design)

- 软件产品从设计之初就应该考虑安全性，即，从软件设计开始就考虑安全问题以抵御外部攻击
- 软件设计安全的实现步骤
  - Step 1. 指定一个软件产品的安全问题负责人
  - Step 2. 对项目组所有的相关人员进行安全培训
    - 软件中基本的安全弱点
    - 常见的软件安全漏洞类型、
    - 基本的软件安全设计方法
    - 常用的软件安全测试方法
  - Step 3. 确保在软件设计完成时，威胁模型已经构建完成
  - Step 4. 坚持设计与编码的基本准则
  - Step 5. 尽快修复偏离准则的所有缺陷和Bug
  - Step 6. 确保软件安全准则的及时更新
  - Step 7. 对所有已经修复的漏洞进行回归测试
  - Step 8. 简化代码和安全模型
    - 代码退化 (Code Degeneration) 或“代码腐败” (Code Rot)
  - Step 9. 在软件产品发布之前进行渗透测试 (Penetration Test)



# 软件安全策略

## ● 默认安全(Secure by Default)

- 当用户打开软件安装包时，所获得的软件就是安全的。即，软件开发商应该向用户提供打开包装就是足够安全的软件产品
- 实现默认安全的方法
  - 不要默认安全装软件的全部特性和功能。只安装那些对多数用户都会使用的特性和功能，并向用户提供可以选择其他特性和功能的简单机制。
  - 在软件应用过程中使用最小特权。即，当软件不需要本地管理员或者域管理员这样高的权限时，则不赋予软件用户这样的权限。
  - 对资源采取适当的保护措施，避免敏感数据和重要资源遭到攻击。



# 软件安全策略

## ● 部署安全 (Secure by Deployment)

- 当用户安装完软件产品后，需要向用户提供相应的文档和工具，指导用户安全地使用和维护软件
- 实现部署安全应该考虑的问题
  - 确保软件提供了配置与管理安全功能的方法。如果软件不提供安全选项和安全配置，则软件管理员是无法确定软件是否安全的。
  - 尽快地创建良好的安全补丁。
    - 如果发现软件中存在安全漏洞，必须及时地的创建能够修复该安全漏洞的补丁
    - 在考虑修复速度的同时，也要考虑补丁的质量，要避免因修复旧的漏洞而引入新的漏洞。
  - 向用户提供必要的帮助信息，通过在线帮助、文档或屏幕提示等方式，帮助用户了解以何种方式安全地使用软件。

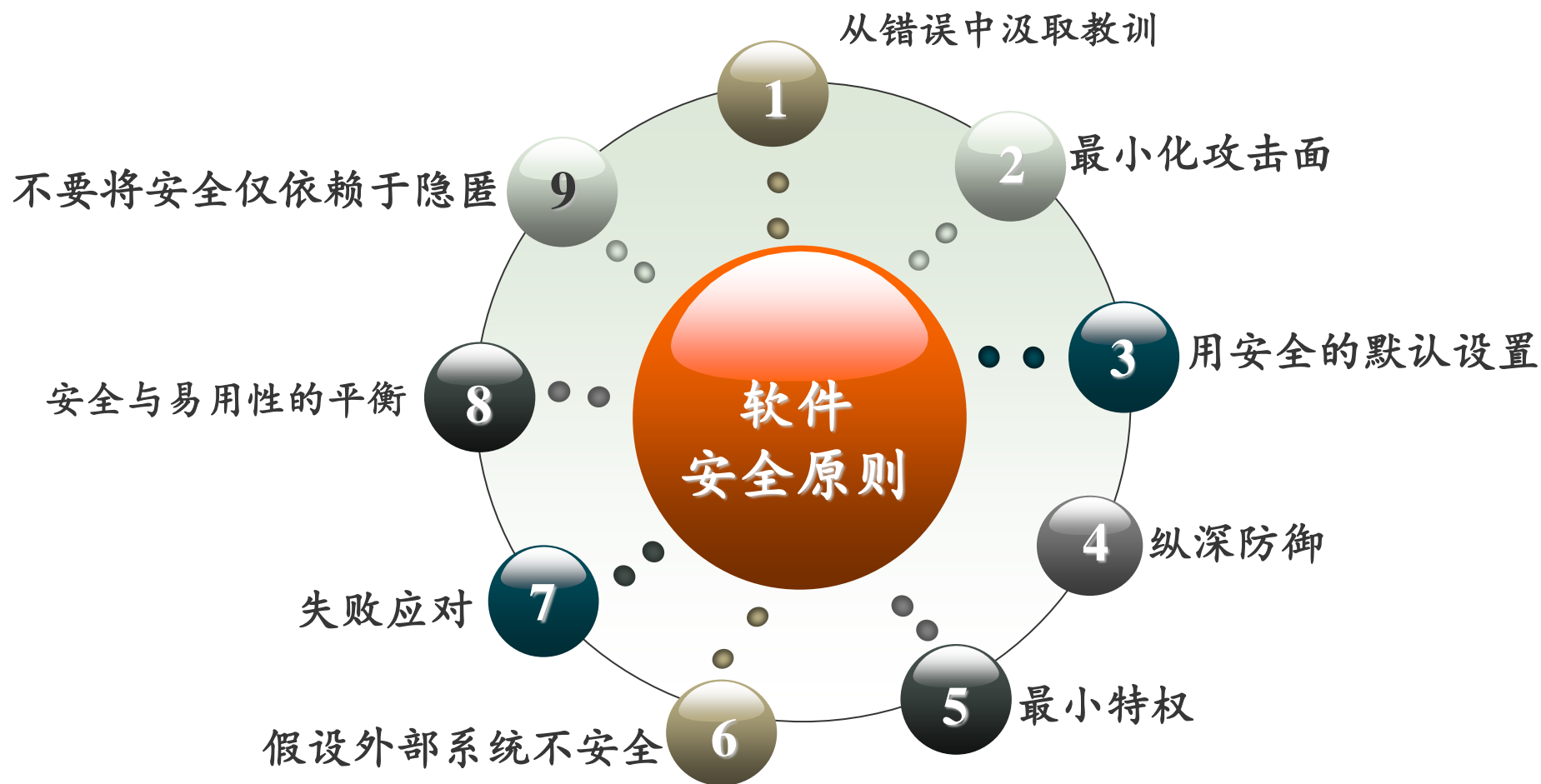




# 软件安全设计原则



# 软件安全原则



# 软件安全原则

## ● 从错误中汲取教训

- 从错误中汲取经验教训，是学习和掌握软件安全知识的最佳途径，也是避免软件安全问题的有效手段
- 当面临一个软件安全问题时，需要从如下方面进行归纳总结
  - 这个安全错误是如何出现的？
  - 同样的安全错误在软件的其他部分是否存在？
  - 如何防止这个安全错误发生？
  - 如何确保这类安全错误将来不会再次发生？
  - 是否需要更新软件安全培训内容以及软件安全分析工具？



# 软件安全原则

## ● 从错误中汲取教训

### ● 软件安全问题修复记录

项目名称	内容
软件产品名称	
软件产品版本号	
联系人	
Bug编号	
漏洞描述	
漏洞含义	
问题是否存在于软件产品的默认安全装中	
为防止这种安全缺陷，设计、开发人员以及测试人员都开展了那些工作	
修复细节描述	



# 软件安全原则

## ● 最小化攻击面

- 软件的受攻击面是一个混合体，包括用户、攻击者以及其他应用程序可以访问该软件的所有**入口点**，涵盖了用户以及潜在的攻击者都能够访问到的软件的**所有功能和代码的总和**。
- 软件的受攻击面不仅包括代码、接口、服务，也包括对所有用户提供服务的协议，尤其是那些未被验证的或远程用户都可以访问到的协议。
- 软件的每个入口点都存在被恶意用户利用的可能。一个软件的**受攻击面越大，则面临的安全风险也越大**。
- 最小化软件攻击面就是去除、禁止一切软件中不必要的功能模块、协议和服务，其目的是将攻击者可以访问软件的入口点保持最少，即最大限度地减少攻击者可以利用的潜在漏洞。



# 软件安全原则

## ● 最小化攻击面

### ● 攻击面示例

较高受攻击面	较低受攻击面
默认执行	默认关闭
打开网络连接	关闭网络连接
同时侦听UDP和TCP流量	仅侦听TCP流量
匿名访问	鉴别用户访问
弱ACLs	强ACLs
管理员访问	普通用户访问
因特网访问	本地子网访问
代码以管理员或root权限运行	代码以Network Services、Local Services或自定义的低权限账户运行
统一缺省配置	用户可选的配置
ActiveX控件	.NET代码
标记有脚本安全的ActiveX控件	未标记有脚本安全的ActiveX控件
非SiteLocked ActiveX控件	SiteLocked ActiveX控件



# 软件安全原则

## ● 最小化攻击面

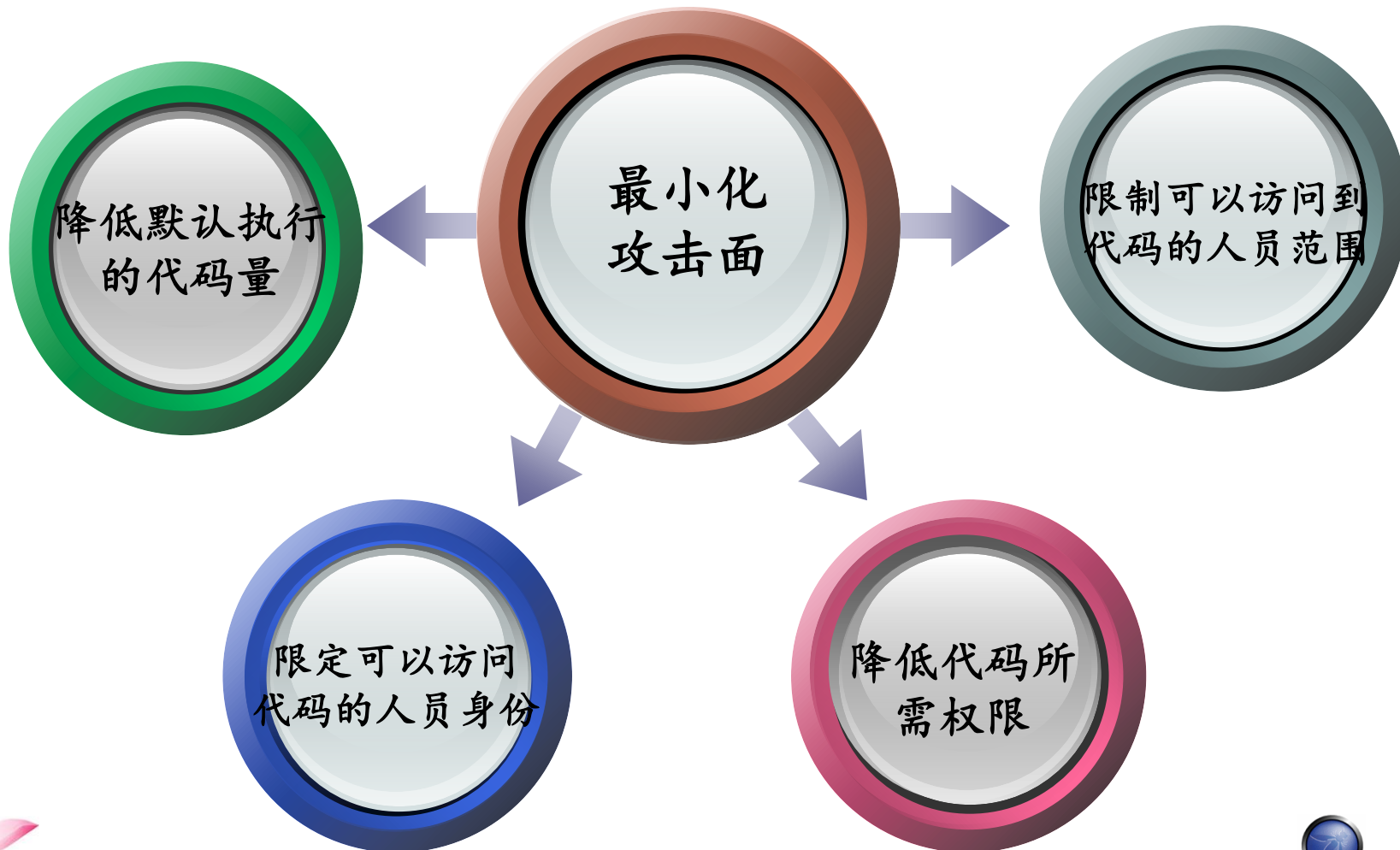
### ● 软件攻击面记录表

攻击面	数量
打开的SOCKET (TCP和UDP)	
打开的命名管道	
打开的远程过程调用 (Remote Procedure Call, RPC)	
终端	
全部服务	
在默认情况下运行的服务	
提高特权后运行的服务	
动态Web页面	
向管理员组添加的账号	
具有弱访问控制列表 (Access Control List, ACL) 的文件、目录以及注册表项	



# 软件安全原则

## ● 最小化攻击面应关注的内容





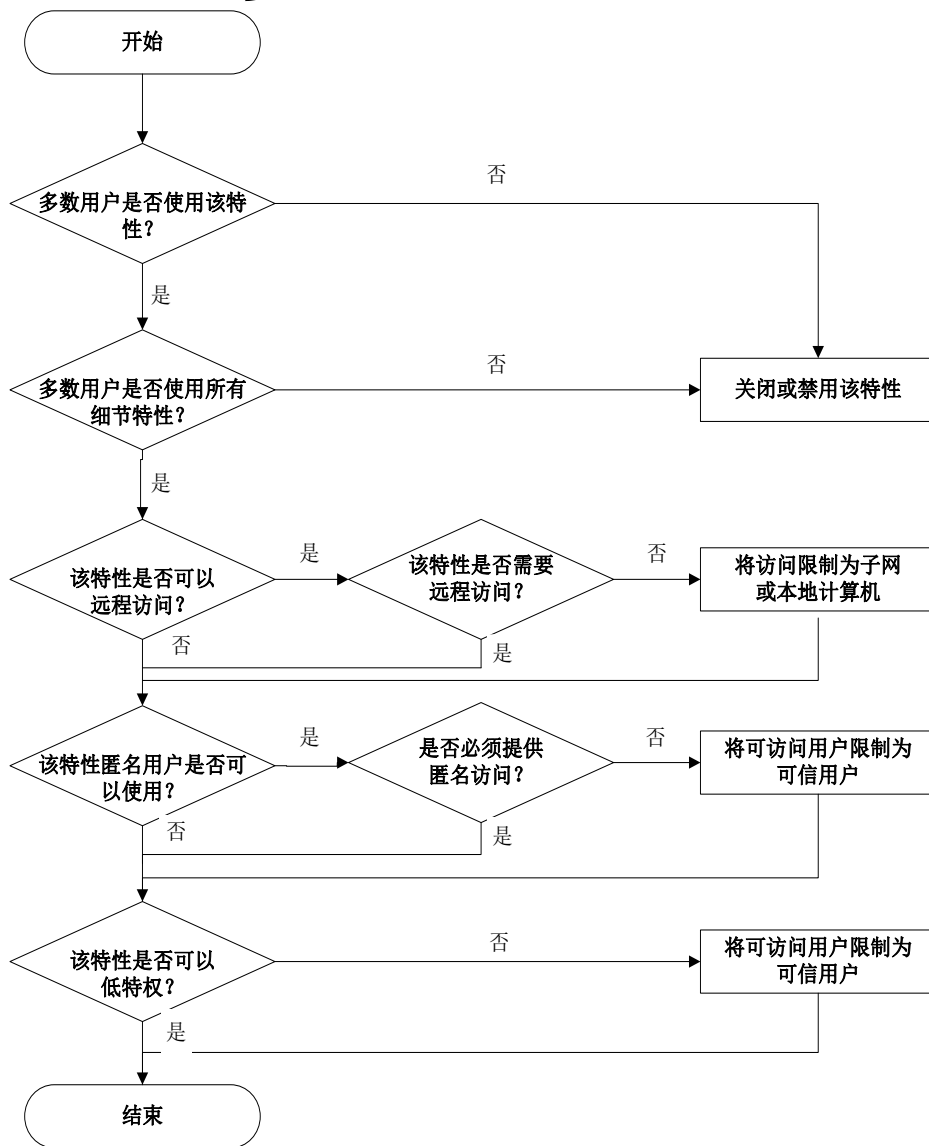
# 软件安全原则

- 降低软件攻击面的过程中需要考虑的问题
  - 分析软件产品功能的重要性，即某些模块、协议等是否有必要保留
  - 分析从哪里可以访问这些功能，从而了解到需要对什么进行控制
  - 采取合理的策略来降低受攻击面，通常采用的策略：
    - 重要等级为低，攻击面大的功能，则考虑取消该功能；
    - 重要等级为中，攻击面大的功能，将该功能设置为非默认开启，需要用户配置后才予以开启；
    - 重要等级为高，攻击面大的功能，关闭或限制一些接口方式，并采用有针对性安全的保证措施或技术。



# 软件安全原则

## 降低软件攻击面的步骤



# 软件安全原则

## ● 采用安全的默认设置

- 最小化攻击面原则意味着软件开发人员需要为软件产品定义安全的默认安装设置
- 软件开发人员需要为用户选择适当的功能，并且确保这些功能的安全
- 对于那些不常用的功能，在默认状态下应该设置为关闭，进而最小化软件的攻击面，减少潜在的安全风险

系统中的典型默认安全原则

应用组件/设置	默认安全原则
防火墙	防火墙默认打开
SSL	默认使用最版本的SSL
匿名访问	默认需要经过认证才能访问应用
口令设置	默认检查口令的强壮性
口令存储	默认存储用户口令的Hash值



# 软件安全原则

## ● 采用安全的默认设置-帕累托原则

- “帕累托原则”（Pareto Principle），又称为“80-20”原则，即软件中20%的功能是为80%的用户所经常使用的。
- 通常情况下，默认20%的功能是打开的，而另外80%的功能是关闭的；
- 向用户提供简单的操作方式以方便用户打开/关闭软件中的功能。
- 减少默认状态下的启动功能还可以节约系统资源，提升系统性能。



# 软件安全原则

## ● 纵深防御

- 即使所有保护软件安全的外部防御机制都遭到破坏，软件自身仍具备抵御攻击和破坏的能力。
- 本质：软件要提供自我保护机制，而不是仅仅依赖外部的安全保障机制，例如已知有防火墙等安全措施来保护软件，但是，仍要考虑防火墙等安全机制均被破坏的情况，并建立应对机制。
- 纵深防御不依赖于单一的安全解决方案，使用多重防御策略、多种互补的安全技术/产品来抵御风险，当一层防御不够时，另一层防御将会阻止进一步的破坏，即使所有外部防御措施均被攻破时，软件自身也应具备防御攻击的能力，不会导致整个系统被彻底攻陷。



# 软件安全原则

## ● 最小特权

- 所有应用程序都应该以能够完成工作的最小特权来执行，而不能使用更多的特权。
  - 当授予系统某些资源的访问权时，就存在滥用该访问权的风险。
  - 最小特权是解决此类问题的有效方法，即，不给予系统必要的访问权限以外的任何其他权限。
- 最小特权原则的实质是任何实体（用户、管理员、进程、应用和系统等）仅拥有其完成规定任务所必需的权限，即：
  - 仅将所需权限的最小集合授权给需要访问资源的主体，并且该权限的持续时间也应该尽可能短。
  - 尽量避免将系统资源暴露在攻击之下，减少因遭受攻击而造成的损失。
- 在软件设计过程中，最小特权原则是指：
  - 所有软件都应该以能够完成工作的最小权限来执行，而不能使用更多的权限确保软件发生故障、错误或遭受攻击等造成的损失最小。



# 软件安全原则

## ● 最小特权

- 用户在最短的时间内以最低的访问权限完成操作。
- 为实现软件的最小特权管理，通常可以采用“因需而知”原则、许可级别和信息敏感度分类以及使用非管理员账户，具体如下：
  - “因需而知”确保敏感信息只能披露给那些被授权可以对消息进行访问的人，可以帮助实现敏感信息的机密性。
  - 许可级别意味着可以根据主体所拥有的安全许可级别（如绝密、机密、秘密、内部等）以及信息的敏感度分类来决定敏感信息的访问权限。建议即使对管理员身份分配敏感资源访问权限时，也应设置为受限访问，而不是超级用户权限。
  - 非管理用账号的使用也可以帮助实现最小特权，例如使用“datareader”或“datawriter”账号替代“admin”或“sysadmin”账户去访问或执行数据库命令，也是实施最小特权的例子。



# 软件安全原则

## ● 假设外部系统不安全

- 假设外部系统不安全原则与纵深防御原则密切相关，这个原则也是确保软件安全的重要防御手段之一
- 主要关注的内容
  - 如果从一个不能完全控制的系统接收数据，那么接收到的任何数据都应该假设为是不安全的，并且有可能是一个攻击源
  - 外部服务器也有可能成为潜在的攻击点
  - 要假设与应用程序进行通信的其他应用程序会对用户可以执行的命令做限制，用户是从用户界面或应用程序中基于Web的客户部分执行这些命令的





# 软件安全原则

## ● 失败应对

- 任何软件都存在失败的可能
- 失败对软件是不可避免的，所以需要有应对失败的计划，以减少或避免与失败相关的安全问题。
- 软件失败带来的安全问题在于，软件失败后会暴露出一些不安全的行为。
- 攻击者只需要触发某些条件导致软件出现失败，或者等待软件失败发生，就可以趁机攻破系统。



# 软件安全原则

## ● 失败应对

### ● 失败应对方案，通常需要考虑的问题如下

- 如果防火墙被攻破会发生何种情况？
- 如果Web站点被破坏会发生何种情况？
- 如果应用程序被破坏会发生何种情况？
- 如果后台数据库被入侵会发生何种情况？
- 一定要针对可能出现的失败情况，有充分的准备，而不能认为失败情况从来不会发生！

### ● 软件失败后出现的情况

- 不安全的失败，即应用程序会泄露包括敏感数据在内的更多数据，数据也有可能已经遭到攻击者的篡改
- 安全的失败，即失败后进入安全模式。在安全模式下，敏感数据泄露、数据遭受篡改等情况均未发生
  - 当应用程序失败，进入安全模式后，不要提供大量的关于失败的解释信息



# 软件安全原则

## ● 失败应对

### ● 软件失败后应考虑的安全问题

- 当达到最大登录尝试次数后，默认状态下用户被拒绝访问，账号被锁定；
- 软件应该设计为不能忽略错误就继续执行下一步操作；
- 错误和异常必须被明确地处理。



# 软件安全原则

## ● 不要将安全仅依赖于隐匿

- 软件设计时，始终要假设攻击者掌握软件设计、开发人员所掌握的所有信息，包括所有的源代码和所有的设计文档资料

## ● 密码学-柯克霍夫原则

- 加密系统除了密钥之外其他信息都应该是公开的，加密机制的安全性依赖于密钥的保密
- 开放设计，安全防御机制的实现不应该依赖设计本身

## ● 软件安全设计也要遵循柯克霍夫原则

- 软件安全不应该依赖于设计的保密；
- 通过模糊或隐匿的方式来实现软件安全是不可取的；
- 软件安全保障机制的设计应该对团队成员的审查工作开放，让每一个团队成员有机会去发现软件中的脆弱点和漏洞远比让攻击者发现好。



# 软件安全原则

## ● 安全与易用性的平衡

- 软件的安全保障机制不应该影响用户对相关资源的访问，不应该影响软件的易用性，不应该影响软件的用户使用体验
- 充分考虑安全保障机制与易用性之间的平衡关系
  - 软件安全保障机制对用户应该是透明的，并且对用户的易用性影响最小，为用户所接受
  - 软件安全机制应该对用户“友好”，为用户提供便利的配置接口，方便用户对软件安全保障机制的理解和使用
  - 不要因考虑用户的易用性，而破坏了软件的安全性



# 软件安全设计方法-威胁建模



# 威胁建模

## ● 概述

- 威胁建模是通过结构化的方法，系统地识别产品面临的安全威胁，确定威胁的风险等级，并通过适当的缓解措施来消减安全威胁，提高产品安全性的工程技术
- 核心：“**像攻击者一样思考**”，从攻击者的角度去评估产品的安全性，分析产品中每个组件是否可能被篡改、仿冒，是否可能会造成信息泄露、拒绝服务攻击等威胁
- 通过规范化的方式来考虑软件的安全性，帮助软件开发人员在设计阶段充分了解各种潜在的安全威胁以及所引起攻击的表现形式，对可能的风险进行管理，并指导其选择适当的应对措施来缓和威胁，降低软件的受攻击面，使软件变得更加安全
- 作用：更倾向于确保产品架构、功能设计的安全，无法保证编码的安全，但可以指导软件开发人员编写出安全的代码，同时也可以辅助渗透测试人员开展安全测试



# 威胁建模

## ● 威胁建模过程

- 威胁建模通过建立分层应用程序数据流图，标识应用程序入口点和信任边界来刻画出整个软件架构
- 建立威胁模型分析欺骗（Spoofing）、篡改（Tampering）、否认（Repudiation）、信息泄露（Information Disclosure）、拒绝服务（Denial of Service）和特权提升（Elevation of Privilege）六类威胁的风险等级（**STRIDE**威胁分类法）
- 指导软件架构的安全设计，并指导后续开发工作

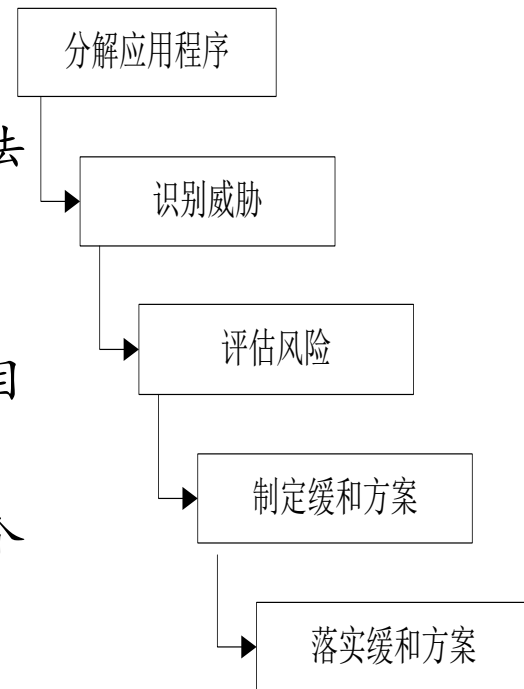




# 威胁建模

## 威胁模型建立过程

- Step 1. 建立威胁建模小组。
- Step 2. 分解应用程序。利用数据流图（Data Flow Diagram, DFD）将应用程序分解为威胁目标的方法。在使用DFD的情况下，威胁目标是每个数据源、过程、数据以及交互者或参与者。
- Step 3. 使用STRIDE威胁分类方法，识别每个威胁目标所面临的威胁，并为每个威胁构建威胁树。
- Step 4. 使用DREAD或其他威胁评估方法，计算每个威胁的风险，以风险递减的方式为威胁排序。
- Step 5. 选择应对威胁的方法。
- Step 6. 选择缓和威胁的技术。
- Step 7. 从确定下来的技术中选择适当的方法来消减威胁。



威胁建模过程



# 成立威胁建模小组


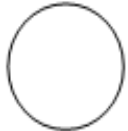
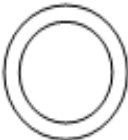



- 成立威胁建模小组是威胁建模过程中的第一步
- 从软件开发小组中选拔相关人员构成威胁建模小组。
- 在该小组中要选择一个负责人，他应该是所有成员中对安全最熟悉的人，能够在任何给定的应用程序或设计中发现威胁，并明确指出攻击者会如何利用该威胁。
- 确保每一个软件开发部门都至少有一个成员参加威胁建模讨论，包括设计、编码、测试和文档部门。
- 在进行威胁建模之前，要向参加的会议的成员明确如下问题：
  - 会议的目的不是解决问题，而是标识出应用程序的各个组成部分，它们之间是如何交互的，最后找到尽可能多的安全威胁
  - 修改设计和代码及为此进行的讨论将在会议结束后进行
  - 不要试图在威胁建模会议上改正问题或者提出解决方案
  - 会议的目的是寻找威胁，而不是如何缓解或解决威胁。但是，讨论过程中往往不可避免地涉及到一些威胁缓和技术，这是可以接受的，但是，一定不要讨论过多的技术细节



# 分解应用

## ● 数据流图 (Data Flow Diagram, DFD)

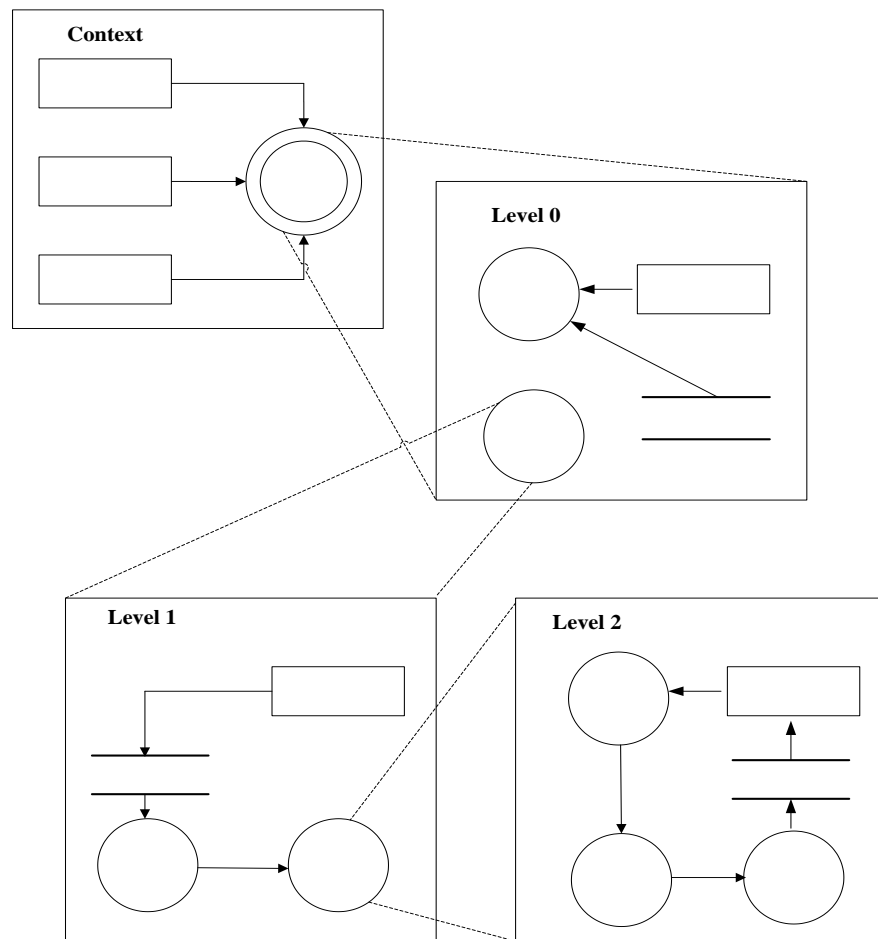
- DFD是建立威胁模型的基础
- DFD重点关注软件所处理的数据是如何流动的以及流动过程中所发生的情况，通过提供软件如何处理数据的可视化表示，使开发人员更好地理解软件是如何工作的以及底层架构

元素类型↵	类型描述↵	形状↵
外部实体 (External Elements) ↵	非受控的外部实体 (可以是用户、软件系统或设备)，可以通过软件提供的入口点与软件进行交互↵	 ↵
单个处理过程 (Process) ↵	转化或操作数据的单个任务或处理过程↵	 ↵
多个处理过程 (Multiple Process) ↵	转化或操作数据的多个处理过程↵	 ↵
数据存储 (Data Store) ↵	存储数据的内部实体，即可以存储永久数据，也可以存储临时数据↵	 ↵
数据流 (Data Flow) ↵	外部实体与进程、进程与进程或者进程与数据存储之间的交互，表示数据的流转，箭头表示数据流动的方向↵	 ↵
信任边界 (Trust Boundary) ↵	信任边界↵	 ↵



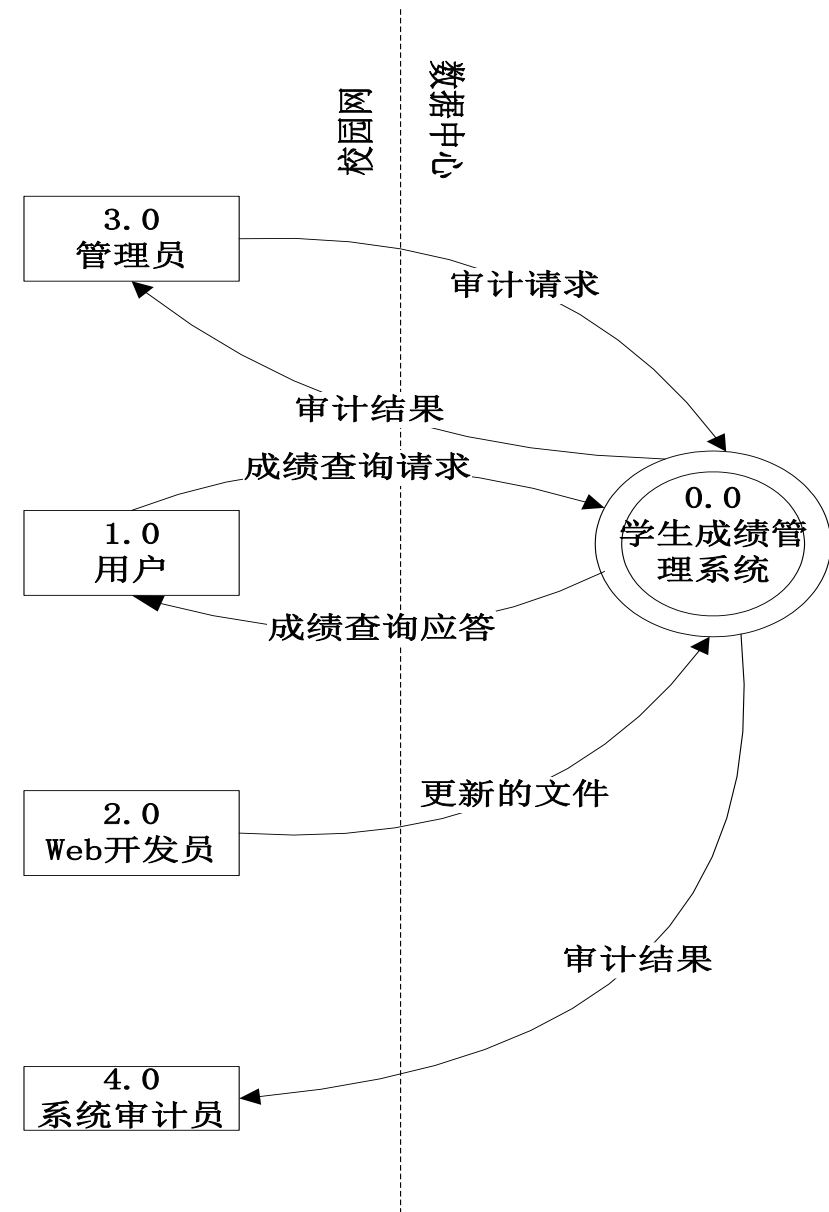
# 分解应用

- 分解应用程序的第一步是确定要分析的应用层的边界或作用范围，并了解可信任部分和不可信任部分之间的边界。
- DFD使用一个高级上下文图表定义应用程序的可达范围，在高级上下文图表中只有一个过程，通常没有存储。在完成第一步之后，需要向下进入较低的层次，使用0级图表、1级图表、2级图表等



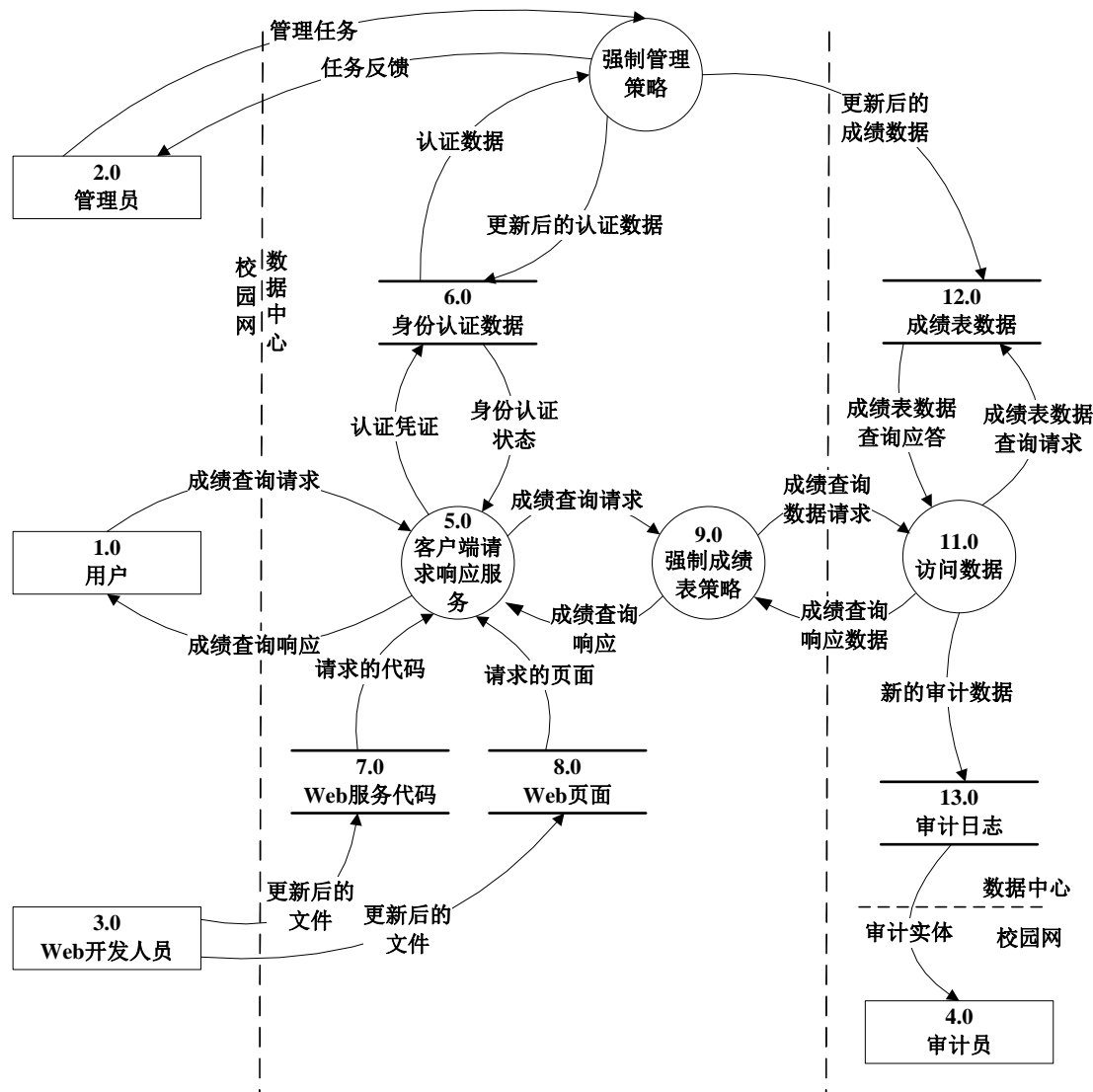
# 分解应用

- 在定义DFD作用范围的时候，需要考虑的问题
  - 忽略应用程序的内部工作方式
  - 那些事件和请求是系统必须响应的？
  - 程序将产生什么样的响应？
  - 标识出每次请求和响应相关的数据源
  - 确定每次响应的接收者



# 分解应用

## ● 学生成绩管理系统的1级数据流图



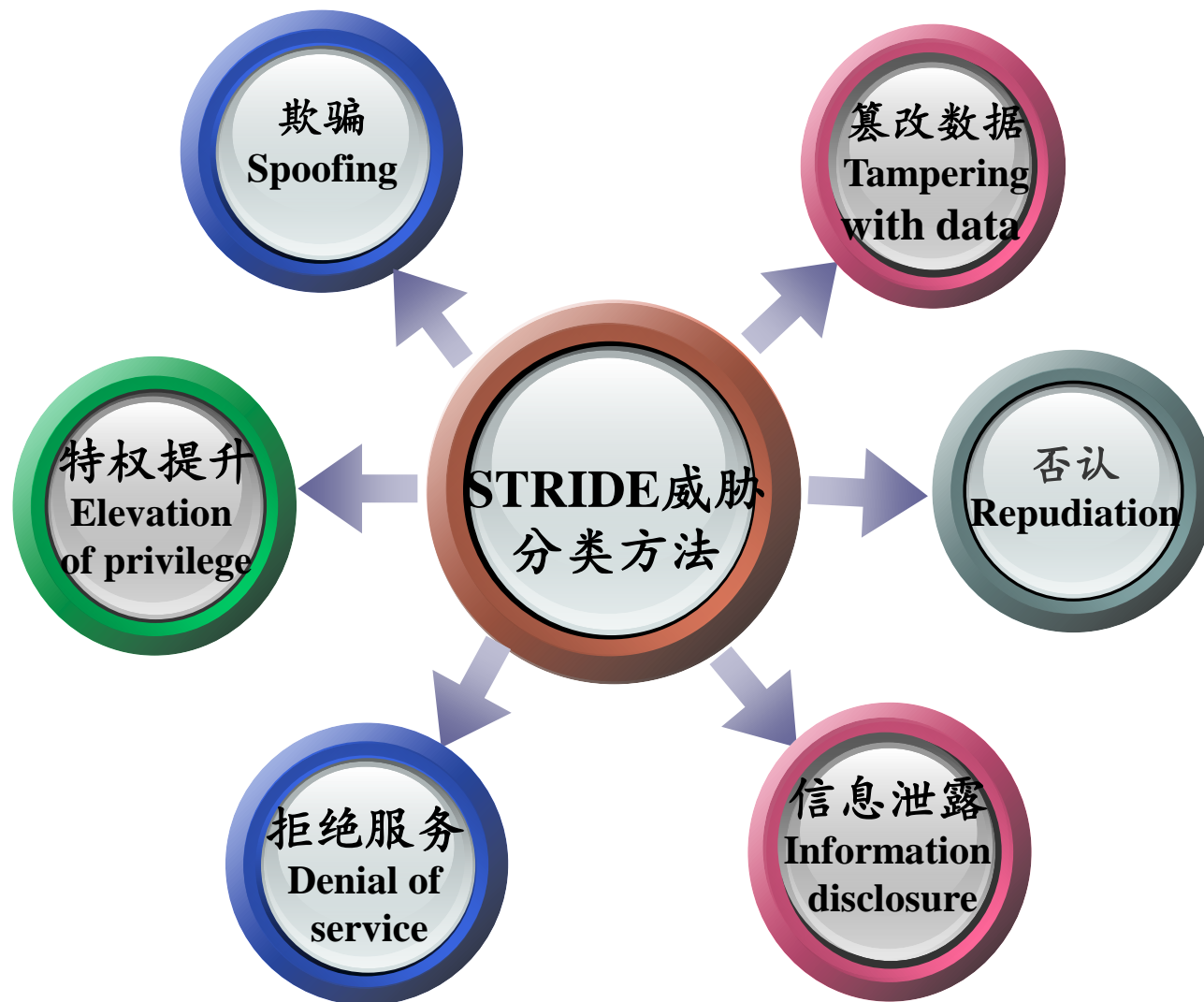
# 分解应用

- 在DFD中创建和命名实体时，应该遵循的规则
  - 一个过程必须至少有一个数据流流入和一个数据流流出
  - 所有数据流都从某个过程开始，到某个过程结束
  - 数据存储通过数据流与过程相连
  - 数据存储之间不能连接，它们必须通过过程连接
  - 过程的名称应该是动词、名词或动词短语（例如录入成绩、查询成绩即创建管理员等）。
  - 数据流的名称应该是名词或名词短语（例如学生成绩、课程名称等）。
  - 外部实体或交互者的名称应是名词（例如学生、教师、系统管理员等）。
  - 数据存储的名称应该是名词（例如学生成绩、审计日志等）。



# 识别威胁

## ● STRIDE威胁分类方法





# 识别威胁

## ● STRIDE威胁分类方法

### ● 欺骗 (Spoofing)

- 攻击者冒充一个用户，或者恶意服务器冒充合法服务器的威胁行为

### ● 篡改数据 (Tampering with data)

- 在未授权的情况下，对数据篡改的行为，包括篡改文件、篡改内存和篡改网络中传输的内容等

### ● 否认 (Repudiation)

- 用户拒绝承认从事过的某项活动，并且无法证明他是在抵赖

### ● 信息泄露 (Information disclosure)

- 信息被暴露给不允许对它访问的人

### ● 拒绝服务 (Denial of service)

- 拒绝为合法用户提供请求的服务

### ● 特权提升 (Elevation of privilege)

- 没有特权的用户获得访问特权，从而具备足够的能力对系统实施破坏性操作甚至摧毁系统



# 识别威胁

## ● STRIDE威胁描述

威胁	安全属性	定义	举例
Spoofing（欺骗）	可鉴别性	模仿其他人或实体	伪装成microsoft.com或ntdll.dll。
Tampering（篡改）	完整性	修改数据或代码	修改硬盘、DVD或网络数据包中的DLL
Repudiation（抵赖）	不可抵赖性	声称没有执行某个动作	“我没有发送过那封电子邮件”，“我没有修改过那个文件”，“亲爱的，我确实没有访问过那个网站！”
Information Disclosure（信息泄露）	机密性	把信息披露给那些无权知道的人	允许某人阅读Windows源代码；公布某个Web网站的用户清单。
Denial of Service（拒绝服务）	可用性	拒绝为用户提供服务	使得Windows或Web网站崩溃，发送数据包并耗尽CPU时间，将数据包路由到某黑洞中。
Elevation of Privilege（权限提升）	授权	获得非授权访问权	允许远程因特网用户执行命令，让受限用户获得管理员权限。



# 识别威胁

## ● STRIDE威胁与DFD中部分元素的对应关系

	外部实体	过程	数据流	存储
身份欺骗 (S)	✓	✓		
篡改 (T)		✓	✓	✓
否认 (R)	✓	✓		✓
信息泄露 (I)		✓	✓	✓
拒绝服务 (D)		✓	✓	✓
特权提升 (E)		✓		



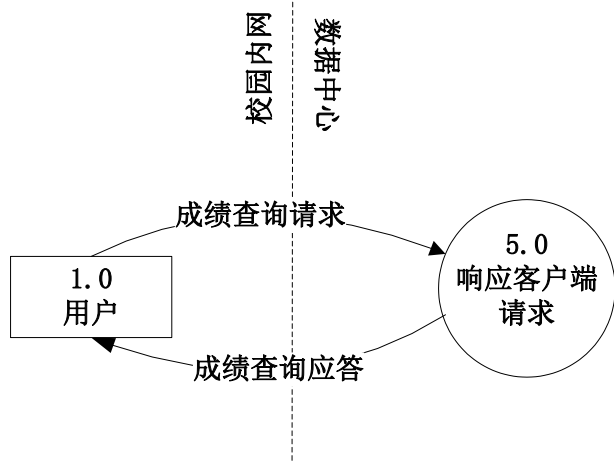
# 识别威胁

## ● 威胁树

- 威胁树（Threat Tree）起源于硬件故障识别领域常用的故障树（Fault Tree），采用树形结构描述系统面临的威胁。
- 威胁树的根节点表示系统所面临威胁的抽象描述，逐层细化威胁的细节信息，直到用叶节点表示具体攻击方式。
- 威胁树描述了攻击者破坏各组件所经历的决策过程，
- 威胁树主要思想
  - 应用程序是由威胁目标构成的
  - 每个目标都有漏洞
  - 任何漏洞被攻击者成功利用后，都会使整个系统遭到破坏



# 识别威胁



1.0 在线浏览学生成绩表数据 (I)

1.1 HTTP通信不受保护 (与)

1.2 攻击者浏览网络传输数据

1.2.1 使用协议分析器嗅探网络通信

1.2.2 监听路由器上的数据

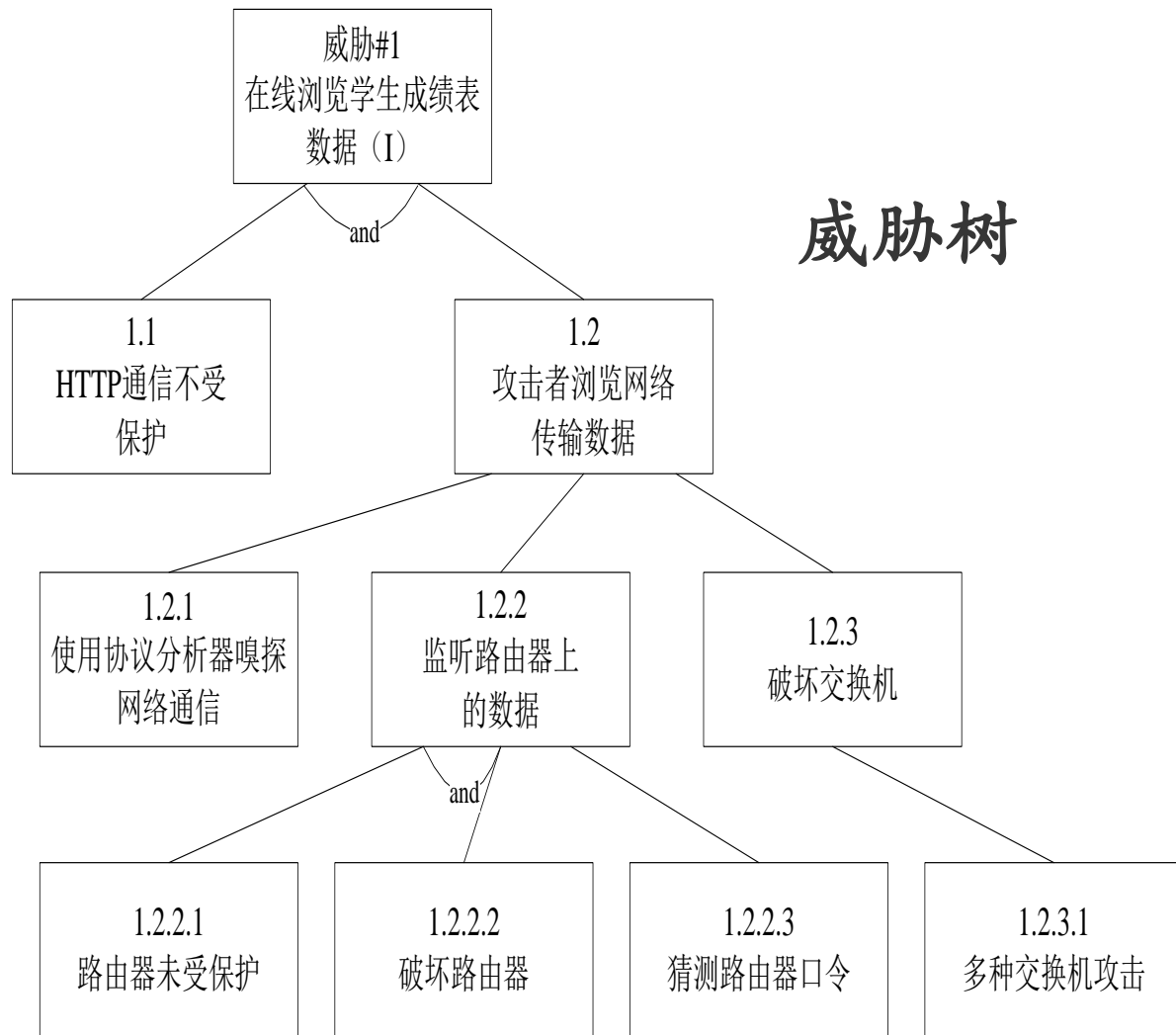
1.2.2.1 路由器未受保护 (与)

1.2.2.2 破坏路由器

1.2.2.3 猜测路由器口令

1.2.3 破坏交换机

1.2.3.1 多种交换机攻击



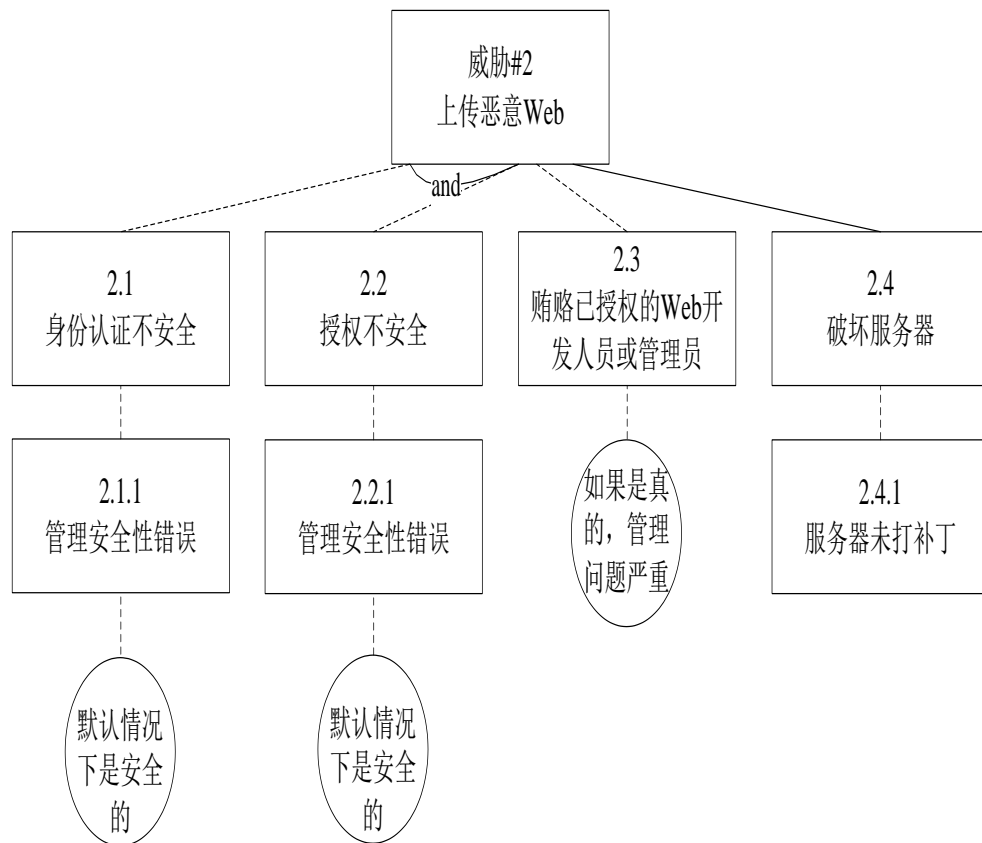
## 威胁树



# 识别威胁

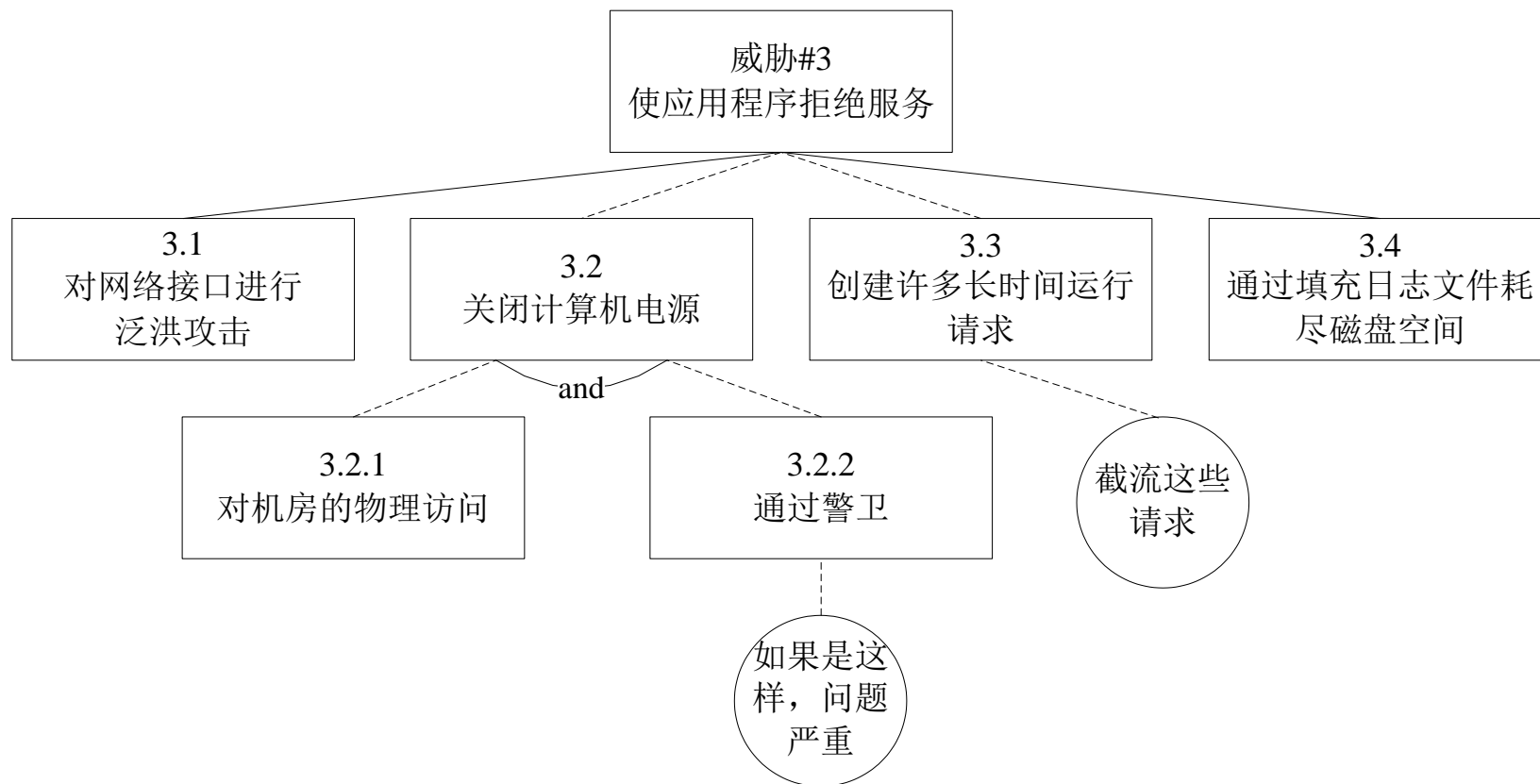
## 对威胁树的改进

- 向威胁树增加一些额外的描述，来显示最有可能的攻击。
- 可以用虚线表示最不可能发生的攻击点，而实线表示最可能发生的攻击点。
- 可以在最不可能的节点下放置圆圈，说明为什么威胁会减弱
- 尽量不在威胁建模过程添加缓和和威胁的圆圈
- 向威胁树添加“表示最不可能的攻击点的虚线”可以达到威胁树的“剪枝”效果



# 识别威胁

## 威胁树示例



# 识别威胁

## ● 威胁表

项目	注释
标题	语言简洁明确，威胁应该从标题上就可以体现出来，例如： 攻击者获取用户口令
威胁目标	应用程序中易受攻击的部分。例如学生成绩管理系统示例中， 威胁目标包括：成绩请求数据流（1.0-5.0）即执行管理员策略的过程（14.0）
威胁类型	基于STRIDE威胁分类方法记录威胁类型
风险	选择风险计算方法，要保证计算结果的一致性
威胁树	攻击者如何表示威胁？保持威胁树的简单性，不要过于复杂
缓和技术（可选）	如何缓和威胁？记录下缓和威胁的技术。
缓和状态	威胁是否缓和？
Bug数量（可选）	记录Bug数量





# 评估威胁

## ● 通用威胁评估方法

- 一个按照风险排序的威胁列表将有助于确定威胁的等级以及消减威胁次序
- 工业界所接受的通用的风险计算方法

风险(Risk) = 潜在破坏性(Damage Potential) × 发生概率(Probability)

- 威胁的潜在破坏程度 (Damage Potential) 与其所影响的应用程序的组件数量以及可能的破坏程度有关
  - 攻击者是否可以完全接管整个系统?
  - 攻击者是否可以获得系统的管理权限?
  - 攻击者是否可以破坏系统?
  - 攻击者是否可以获取系统中的敏感信息?
  - 威胁所影响的数据源或系统数量有多少?
- 威胁的发生概率 (Probability) 与威胁被攻击者利用的难易程度相关, 主要取决于威胁的类型和系统的特征
  - 攻击者可以远程地利用这个漏洞吗?
  - 攻击者是否需要身份认证?
  - 这个威胁可以被自动利用吗?
- 计算出的风险 (Risk) 通常由数字1-10来描述, 1表示最低的危急程度和发生的可能性, 10表示最高的危急程度或发生的可能性。



# 评估威胁

## ● DREAD威胁评估方法 (Microsoft)

- **D**amage potential (破坏性)
  - 衡量威胁可能造成的实际破坏程度.
- **R**eproducibility (再现性)
  - 衡量威胁变成实际攻击的难易程度
- **E**xploitability(可利用性)
  - 衡量进行一次攻击需要多少努力和专业知识
- **A**ffected users(影响用户)
  - 攻击可能影响的用户数.
- **D**iscoverability(可发现性)
  - 衡量是否易于发现的程度 (最难衡量的标准)
- 上述每个方面危害程度的取值范围是1-10, 数值越高造成的威胁越大, 10表示威胁造成的危害程度最大, 1表示威胁造成的危害程度最小



# 评估威胁

## ● DREAD威胁评估方法取值示例

	低风险 (1)	中风险 (5)	高风险 (10)
潜在破坏性 (D)	普通的信息泄露	敏感信息泄露	攻击者可以破坏安全系统；获得完全信任授权；以管理员身份运行；可以上传任意内容
再现性 (R)	即使存在安全漏洞，攻击也很难发生	在特定的情况和要求下，攻击可以发生	攻击在任何时刻都可以很容易地发生
可利用性 (E)	攻击需要精深的专业知识或高级的专业工具	攻击需要普通的专业知识或一般专业工具	几乎不需要专业知识或仅需要常见的工具
受影响用户 (A)	比例极低的用户	部分用户	全部用户
可发现性 (D)	特征模糊，极难发现	存在于少部分的功能中，部分用户可以发现	特征明显，极易发现



# 评估威胁

- DREAD威胁计算公式

风险(Risk) = 潜在破坏性(Damage Potential) × 发生概率(Probability) (1)

- 依据式(1)，可以计算出五个风险值，然后对五个风险值求平均数，如式(2)所示，平均数越大，则威胁对系统造成的风险就越大

- $$\text{Risk\_DREAD} = (\text{DAMAGE} + \text{REPRODUCIBILITY} + \text{EXPLOITABILITY} + \text{FFECTED USERS} + \text{DISCOVERABILITY})/5 \quad (2)$$



# 评估威胁

## ● DREAD威胁计算示例

威胁1：在线浏览学生成绩表数据	
潜在破坏性 (D)	9
再现性 (R)	10
可利用性 (E)	8
受影响用户 (A)	10
可发现性 (D)	10
风险： $(9+10+8+10+10) / 5 = 9$	



# 消减威胁

## ● 针对威胁的解决方案

1

保持现状，不做任何工作

- ✓ 低风险的安全威胁
- ✓ 威胁将始终潜伏在应用程序中
- ✓ 应该向用户告知可能存在的安全威胁

2

将威胁告知用户

- ✓ 告知用户，由用户决定
- ✓ 过于专业，用户无法决定
- ✓ 频繁的威胁警告，会导致用户忽略警告

3

将存在威胁的功能删除或关闭

- ✓ 将存在该威胁的功能从产品中删除或关闭
- ✓ 移除最彻底，关闭只是降低风险

4

消减威胁

- ✓ 寻找针对威胁的解决方案，并选择合适的技术来消减产品中的威胁
- ✓ 最明显的解决方案
- ✓ 最难的解决方案



# 威胁缓解方法与技术

## ● 针对STRIDE威胁的缓解方法

威胁类型	缓解方法
身份欺骗 (S)	认证 (Authentication)
篡改数据 (T)	完整性验证 (Integrity Verification)
否认 (R)	抗抵赖服务 (Non-repudiation services)
信息泄露 (I)	保密性技术 (Confidentiality)
拒绝服务 (D)	可用性技术 (Availability)
特权提升 (E)	授权管理 (Authorization)



# 威胁缓解方法与技术

## ● 部分威胁缓解技术

缓解方法	缓解技术	缓解方法	缓解技术
认证	Basic认证 Digest认证 Cookie认证 Windows认证 Kerberos认证 SSL/TLS PKI IPSec 数字签名 MAC Hash函数	抗抵赖服务	强认证 安全审计与日志记录 数字签名 安全时间戳 可信第三方
		保密性技术	访问控制列表 加密
		可用性技术	访问控制列表 过滤 (Filtering) 配额 (Quota) 授权
		授权管理	访问控制列表 组或角色成员 特权属主 权限
完整性验证	强完整性控制 访问控制列表 数字签名 MAC		

