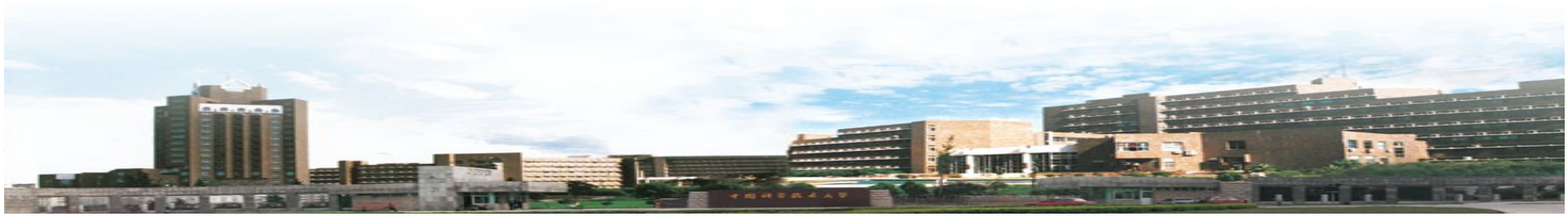
软件安全概述

程绍银

sycheng@ustc.edu.cn



大纲

- 软件与软件安全
- 软件安全的三个支柱
- 软件安全与其他相关领域的关系
- 软件安全工具简介

大纲

- ✓ 软件与软件安全
- 软件安全的三个支柱
- 软件安全与其他相关领域的关系
- 软件安全工具简介

软件的定义

- Software：软件（大陆、香港），软体（台湾）
- 1983年IEEE为软件下的定义是：计算机程序、方法、规则和相关的文档资料以及在计算机上运行时所需的数据
- 国标中对软件的定义为：与计算机系统操作有关的计算机程序、规程、规则，以及可能有的文件、文档及数据
- 通俗解释：软件 = 程序 + 数据 + 文档资料
  - 程序是完成特定功能和满足性能要求的指令序列
  - 数据是程序运行的基础和操作的对象
  - 文档与程序开发、维护和使用有关的图文资料

软件 VS. 硬件

- “硬件”是躯体，“软件”是灵魂
- “软件”安装的硬件平台
  - 主机(PC、笔记本、服务器等)
  - 网络设备(路由器、交换机、防火墙、UTM等)
  - 消费类电子（智能手机、iPad等）
- “软件”可以包括
  - 操作系统
  - 数据库系统（与系统中的数据有区分）
  - 业务相关的管理信息系统（MIS）
  - 应用程序商店（App Store）
  - **联网软件**：通信协议、通信加密、认证等

软件的特点

- 软件不同于硬件，他是计算机系统中的逻辑实体而不是物理实体，具有抽象性
- 软件的生产不同于硬件，它没有明显的制作过程，一旦开发成功，可以大量拷贝同一内容的副本
- 软件在运行过程中不会因为使用时间过长而出现磨损、老化以及用坏问题
- 由于软件漏洞不可避免，因此软件一般需要不断升级
- 软件的开发、运行在很大程度上依赖于计算机系统，受计算机系统的限制，在客观上出现了软件移植问题
- 软件开发复杂性高，开发周期长，成本较大
- 软件开发还涉及诸多的社会因素

软件的分类

- 按软件的功能分类
  - 系统软件：计算机运行的必不可少的组成部分，它与计算机硬件紧密配合，控制并协调计算机系统各个部件、相关的软件和数据高效地工作。例如操作系统、设备驱动程序、通信处理程序等
  - 支撑软件：协助用户开发软件的工具性软件，其中包括帮助程序人员开发软件产品的工具，也包括帮助管理人员控制开发进程的工具。例如支持需求/设计/编码/测试的软件、IDE、SDE等
  - 应用软件：在特定领域内开发，为特定目的服务的软件。如MIS、OA等

软件的分类

- 按软件规模分类

类 别	参加人员数	研制期限	产品规模(源程序行数)
微型	1	1 周~4 周	500 行
小型	1	1 月~6 月	1000 行~2000 行
中型	2~5	1 年~2 年	5 千行~5 万行
大型	5~20	2 年~3 年	5 万行~10 万行
甚大型	100~1000	4 年~5 年	100 万行以上
极大型	2000~5000	5 年~10 年	1000 万行以上

软件的分类

- 按软件工作方式分类
  - 实时处理软件
  - 分时软件
  - 交互式软件
  - 批处理软件
- 按软件服务对象的范围划分
  - 面向部分客户的项目软件（定制软件）
  - 面向市场的产品软件

软件的分类

- 按软件运行平台划分
  - 服务器端软件
  - PC端软件
  - 手机应用软件
  - 嵌入式软件

软件的分类

- 按授权方式划分
  - 专属软件：此类授权通常不允许用户随意的复制、研究、修改或散布该软件。违反此类授权通常会有严重的法律责任。传统的商业软件公司会采用此类授权，例如微软的Windows和办公软件。专属软件的源码通常被公司视为私有财产而予以严密的保护
  - 自由软件：此类授权正好与专属软件相反，赋予用户复制、研究、修改和散布该软件的权利，并提供源码供用户自由使用，仅给予些许的其它限制。以Linux、Firefox 和OpenOffice 可做为此类软件的代表

软件的分类

- 按授权方式划分
  - 共享软件：通常可免费的取得并使用其试用版，但在功能或使用期间上受到限制。开发者会鼓励用户付费以取得功能完整的商业版本。根据共享软件作者的授权，用户可以从各种渠道免费得到它的拷贝，也可以自由传播它。
  - 免费软件：可免费取得和转载，但并不提供源码，也无法修改
  - 公共软件：原作者已放弃权利，著作权过期，或作者已经不可考究的软件。使用上无任何限制

软件 VS. 服务

- 软件即服务 (Software as a Service，简称SaaS)
  - 是一种软件交付模式
  - 在这种交付模式中，云端集中式托管软件及其相关的数据，软件仅需透过互联网，而不须透过安装即可使用
  - 用户通常使用精简客户端经由一个网页浏览器来访问软件即服务
  - 通过将硬件和软件维护及支持外包给软件即服务的提供者，来降低信息技术成本

开源软件的授权方式

- 开源软件共同特点：源代码开放、免费修改、免费重新发布
- GPL：自由软件。允许免费修改、免费重发布，但要求修改代码必须也遵守GPL。在GPL下面还有LGPL，允许商业代码链接LGPL代码，这样商业软件在利用LGPL软件的同时能够很大程度上保留商业利益。如Linux/gcc/KDE/gnome等
- BSD：公共域软件，其中apache的授权叫APL，是比较典型的授权声明。虽然保留版权，但不但免费修改、免费重新发布，而且允许商业使用，允许商业修改后不公布修改的软件代码。如xWindows/freeBSD/apache/perl/python/ruby等
- MPL：商业公司的开源策略。允许免费重发布、免费修改，但要求修改后的代码版权归软件的发起者，这样发起者和组织者具有更优越的地位。MPL一般也是同时遵守LGPL的。如mozilla/openoffice/vim等

软件安全的概念

- 软件安全就是使软件在受到恶意攻击的情形下依然能够继续正确运行的工程化软件思想
  - Gary McGraw, Cigital公司首席技术官和董事会成员
  - 《Software Security: Building Security In》/《软件安全：使安全成为软件开发必需的部分》
  - http://buildsecurityin.us-cert.gov/portal/
- 采用系统化、规范化、和数量化的方法来指导构建安全的软件

软件安全

- 软件安全 vs. 安全的软件，不是一回事
- 软件安全是一种系统级的问题，需要考虑安全机制（比如访问控制）和基于安全的设计（比如使攻击难以实施的健壮设计）
- 软件安全必须成为完整的软件开发生命周期方法的一部分

软件安全的特点

- 软件安全是计算机安全的一个分支
- 软件安全是计算机安全的关键
- 软件安全问题的根源是软件存在弱点
- 用工程化的方法来实施软件安全

软件安全是计算机安全的一个分支

- 现代社会中，我们生活所需的一切似乎都离不开计算机系统，我们的电力、供水、交通、通讯、金融等等，都依赖于计算机系统的安全运行。但是，计算机系统并不安全，它潜伏着严重的不安全性、脆弱性和危险性
- 如何保障计算机系统的安全就成为我们这个时代的一个根本问题，计算机安全这门学科也因此应运而生，并成为近二十年来最热门的学科之一
- 计算机安全的研究范畴包括硬件安全、软件安全、数据安全、运行安全和网络安全

软件安全是计算机安全的关键

- 由于病毒主要是通过网络传播，而黑客主要是通过网络来进行攻击，因此，多年来人们一直认为网络安全是计算机安全的主要问题。但是在网络安全上的巨大投入却没有从根本上解决计算机安全问题
- 软件是计算机安全的大问题，危害计算机安全的绝大部分因素都与软件相关
  - 软件的不稳定导致系统崩溃和数据丢失
  - 病毒攻击的是软件的缺陷
  - 黑客利用的是软件的弱点
  - 机密和隐私的泄漏是因为软件存在漏洞
- 计算机安全中的首要和关键问题是软件问题

软件安全问题的根源是软件存在弱点

- 现有方法并不能解决软件安全问题
  - 反病毒程序和防火墙之类的保护程序
  - 密码学之类的信息加密技术
- 假设有一个程序，其密码验证是用加密算法来实现的，它安装在一个被防火墙保护的系统中，系统中安装了反病毒程序
  - 这个程序是否健壮，可能存在溢出类弱点
  - 没有谁能保证防火墙和反病毒软件是完全可靠的，它们也可能存在缓冲区溢出之类的弱点

软件安全问题的根源是软件存在弱点

- 软件安全问题的根源就是我们所依赖的软件存在太多的安全弱点，而不断增加的软件复杂性和可扩展性更是火上浇油般地助长了这种情形
- 解决软件安全问题的根本方法就是改善我们建造软件的方式，以建造健壮的软件，使其在遭受恶意攻击时依然能够安全可靠和正确运行

任何软件都是不安全的

- 测试只能减少软件安全问题的发生，但是不能完全解决安全问题
- 业界公认一个事实：几乎所有的软件都是带着安全隐患投入运行
- 从技术方面讲，软件的安全问题是普遍存在的。黑客总可以采取种种手段入侵，让用户防不胜防

软件不安全性的表现

- 软件在运行过程中不稳定，出现异常现象、得不到正常结果或者在特殊情况下由于一些原因造成系统崩溃
  - 由于异常处理不当，软件运行期间遇到突发问题，处理异常之后无法释放资源，导致这些资源被锁定无法使用
  - 由于线程处理不当，软件运行中得不到正常结果
  - 由于网络连接处理不当，网络软件运行过程中，内存消耗越来越大，系统越来越慢，最后崩溃
  - 由于编程没有进行优化，程序运行消耗资源过大

软件不安全性的表现

- 黑客利用各种方式达到窃取信息、破坏系统的目的
  - 黑客通过一些手段获取数据库中的明文密码
  - 黑客利用软件的缓冲区溢出，运行敏感的函数
  - 黑客利用软件对数据的校验不全面，给用户发送虚假信息
  - 黑客对用户进行拒绝服务攻击

软件不安全的原因

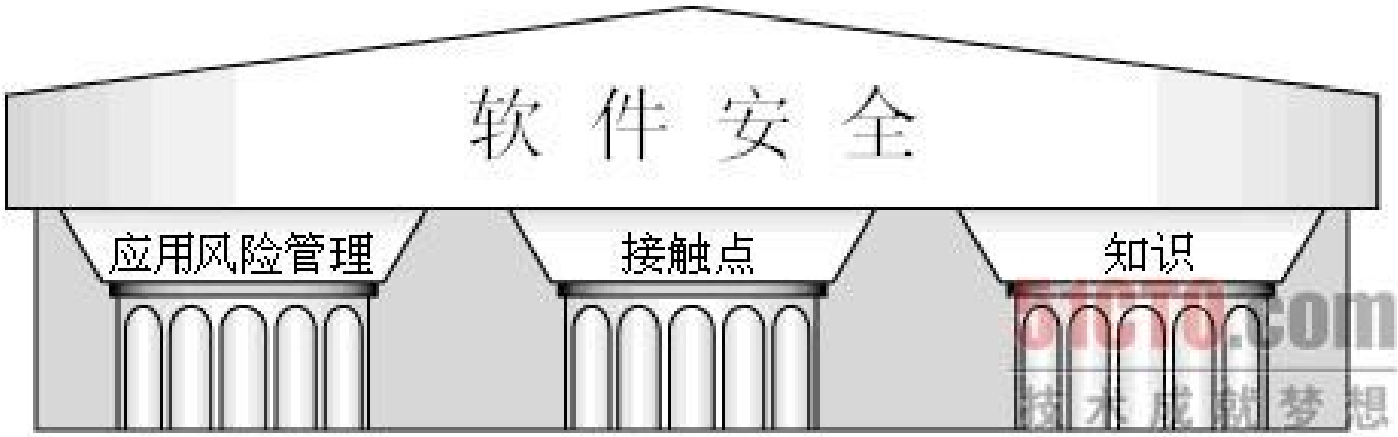
- 从软件开发者的角度
  - 软件生产没有严格遵守软件工程流程
  - 编程人员没有采用科学的编程方法
  - 测试不到位（不过有时是无法到位）
- 从软件工程客观角度
  - 软件复杂性和工程进度的平衡
  - 安全问题的不可预见性
  - 由于软件需求的变动
  - 软件组件之间的交互的不可预见性

大纲

- 软件与软件安全
- ✓ 软件安全的三个支柱
- 软件安全与其他相关领域的关系
- 软件安全工具简介

用工程化的方法来实施软件安全

- 在整个软件开发生命周期中都要确保将安全作为软件的一个有机组成部分
- 软件安全的三根支柱：应用风险管理、软件安全的接触点（切入点）和知识



应用风险管理

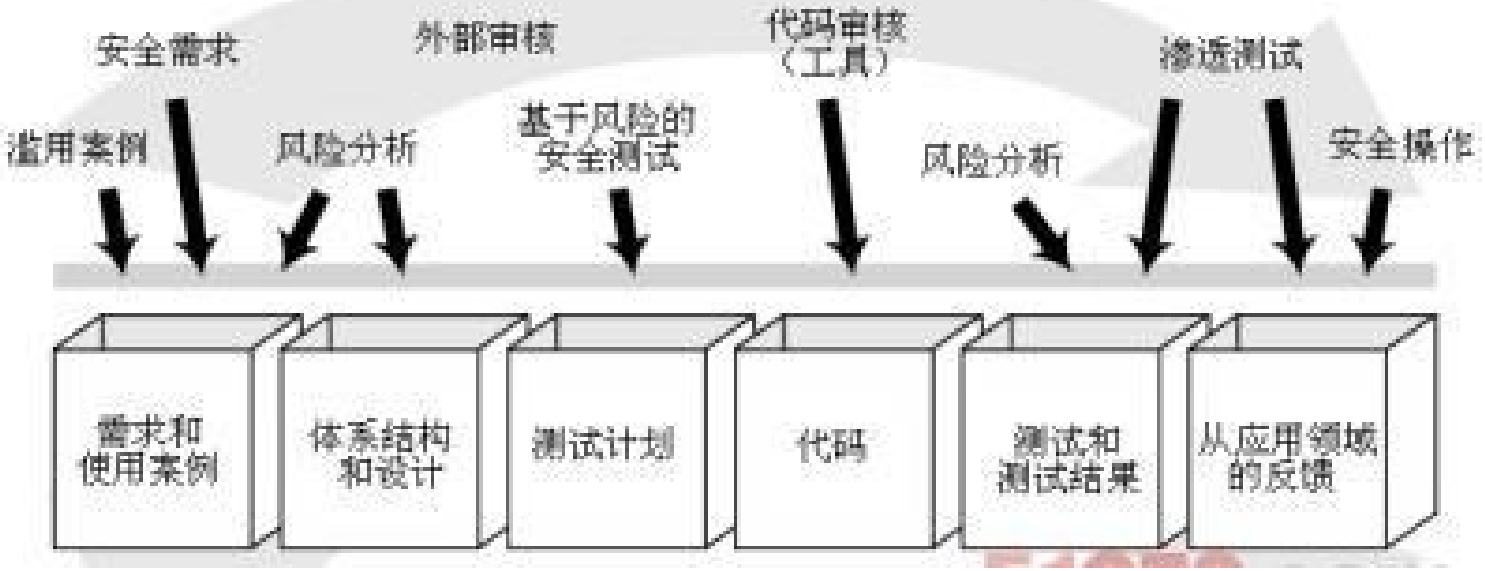
- 风险管理是一种战略性方法，即将追踪和减轻风险作为一种贯穿整个生命周期的指导方针
- 成功的风险管理其实就是一种业务级中的决策支持工具；一种收集必需的数据并基于弱点、威胁、影响和概率的知识作出正确判断的方法

软件安全的接触点

- 接触点，即在软件开发生命周期中保障软件安全的一套最优方法，是一种战术性方法
- 七个接触点：代码审核、体系结构风险分析、渗透测试、基于风险的安全测试、滥用案例、安全需求和安全操作
- “安全的”开发生命周期能够在每一个开发阶段上尽可能地避免和消除漏洞

软件安全的接触点

- 应用于不同的软件工件的软件安全最优方法（根据传统的瀑布模型来布局）



软件安全的知识

- 知识，包括收集、压缩和共享能用于为软件安全方法提供坚实基础的安全知识
- 由于软件安全是一门新的学科，及时总结知识，并用知识来教育所有相关的人员，对确保软件安全是至关重要的
- 在整个开发生命周期中综合应用这些方法，就能从设计、编码和测试等各个层面上消除软件中的安全弱点，从制度上、方法上最大限度地保障软件安全

软件安全的知识

- 知识是特定领域中相互关联的信息 --- 能够通过一定的方法和过程得以运用的信息
- 例如：C和C++中的一组潜在的安全缺陷是信息；在静态分析工具中的这样的信息就是知识
- 软件安全知识可以归成为七种（原则、方针、规则、弱点、攻击程序、攻击模式和历史风险），并划分为三个知识类（说明性知识、诊断性知识和历史知识）

软件安全的知识

- 描述性知识：包括原则、方针和规则
  - 原则和方针是从方法论的高度进行定义和描述
  - 规则是从代码级角度进行有针对性地抽象和统一
- 描述性知识类提供了一些建议，旨在说明该做什么和在构建安全的软件时应该避免什么
- 历史知识包括历史风险，在有些情形下也包括弱点的历史数据库
  - 这类知识还包括对在实际的软件开发中所发现的特定问题的详细描述，以及该问题产生的影响

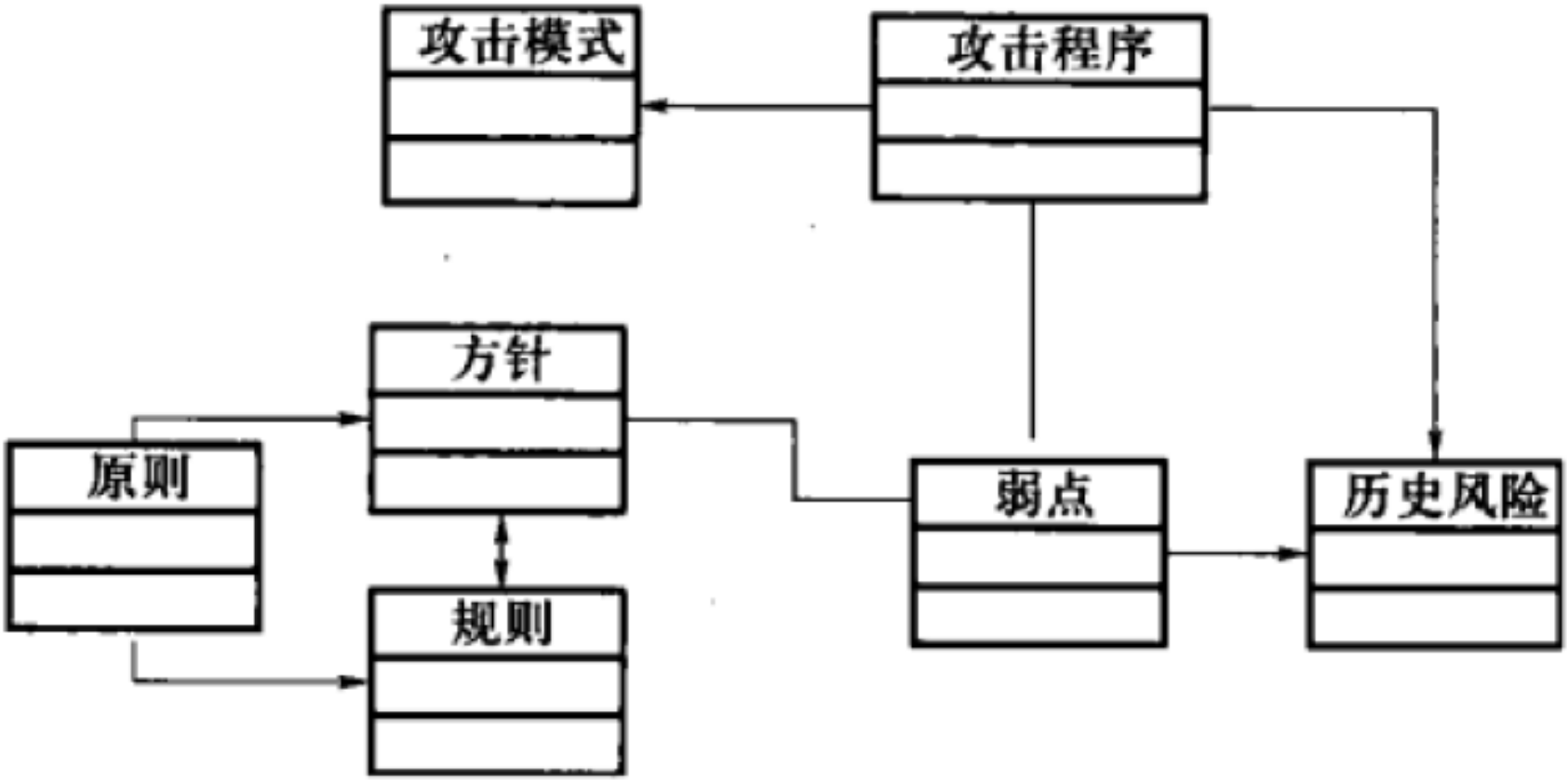
软件安全的知识

- 诊断性知识：包括攻击模式、攻击程序和弱点
  - 攻击模式采用较抽象的形式来描述常见的攻击程序，这种形式能够应用于跨越多个系统的情形，即在多个系统中均存在的攻击模式，该知识可被安全分析人员所利用，如基于滥用案例的可靠性检测等。
  - 攻击程序描述了弱点实例如何被用来对特定系统造成特别的安全危害
  - 弱点知识是对真实系统中出现过并报告的软件弱点的描述
- 诊断性知识不仅包括关于实践的描述性陈述，其更重要的目标是帮助操作人员识别和处理导致安全攻击的常见问题

软件安全的知识

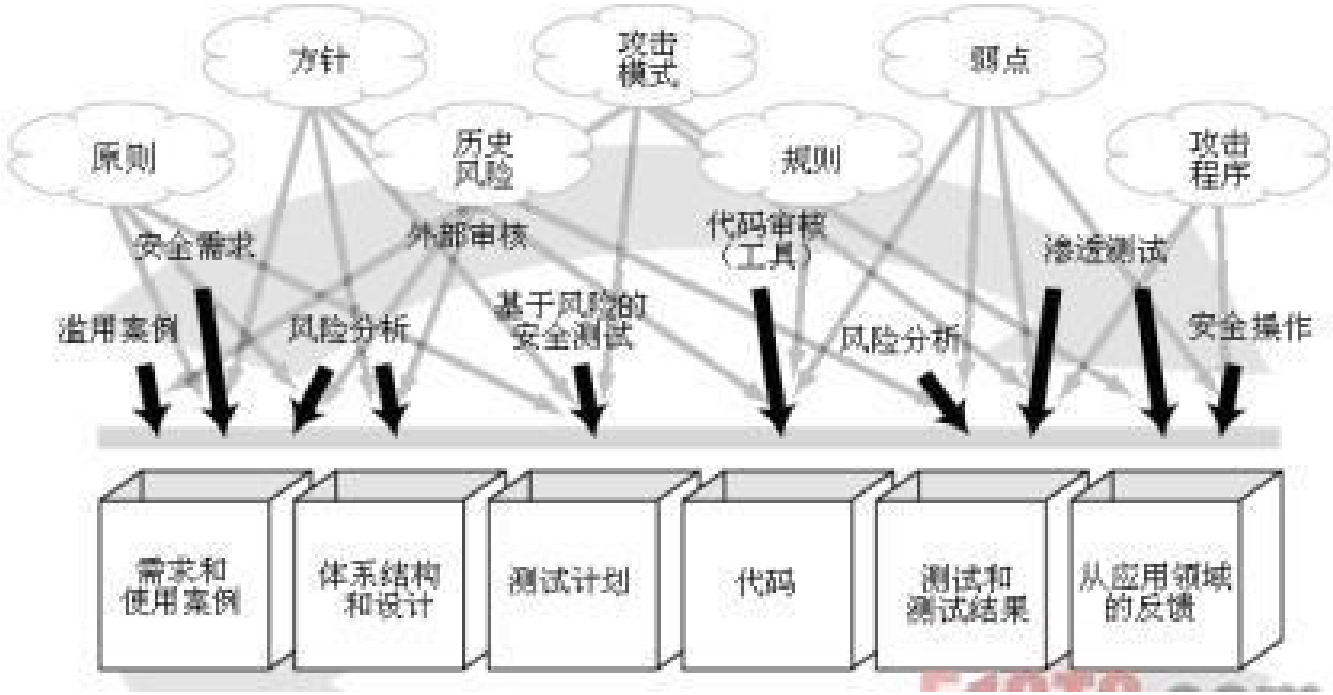
- 描述性知识从战略的角度进行描述，主要包含一些长期积累和提炼出来相对抽象的元知识
- 诊断性知识从战术的角度进行描述，可能与具体系统相关，攻击模式和程序从攻击的角度描述，弱点从防御的角度描述
- 历史知识库是知识的历史积累和前后关联的总结

软件安全的统一知识体系结构



软件安全的知识

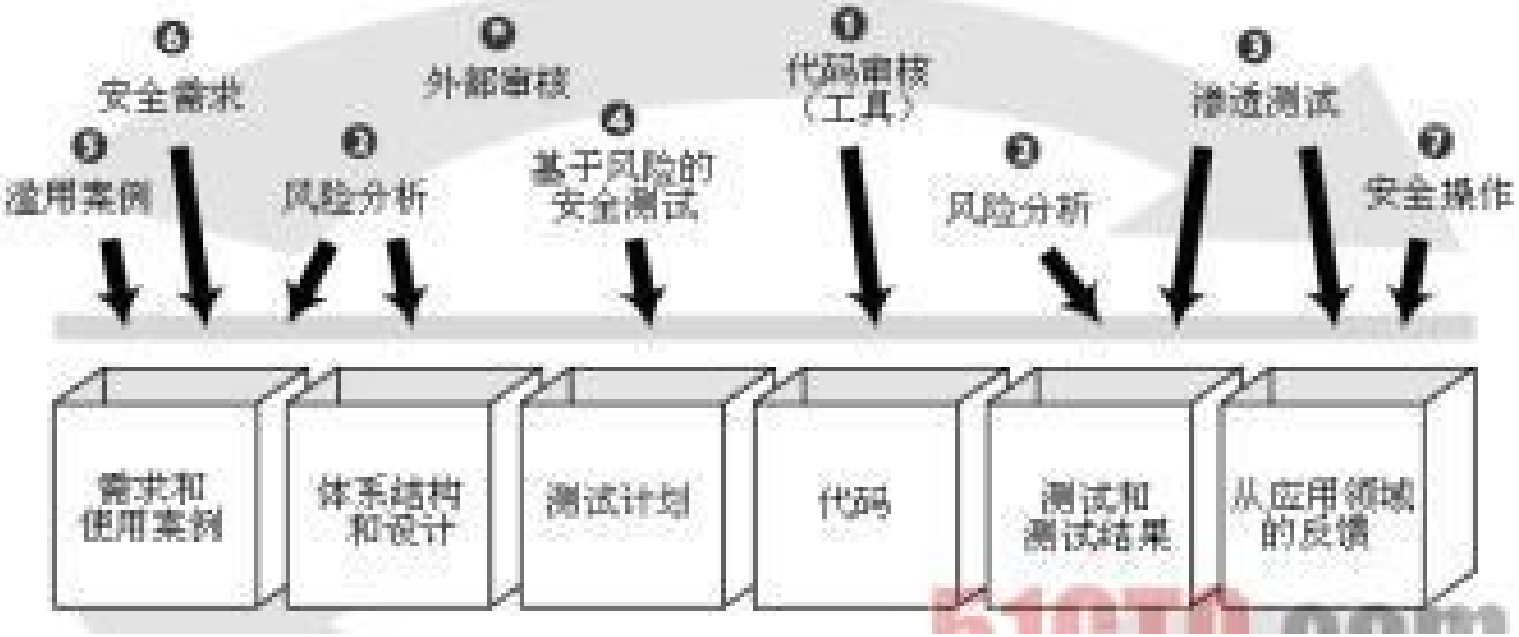
- 软件安全知识类映射为不同的软件工件和软件安全最优方法



软件安全的最优方法

- 实施软件安全要求对常规建造软件的方式做一些修改
- 这些修改并不是根本性的、翻天覆地的或者费用高得难以承受
- 采用一组简单明了的最优工程方法，这些方法可以结合到现有的开发过程中
- 在软件开发生命周期中集成软件安全最优方法（接触点），是软件安全的三根支柱的核心

软件安全的最优方法


❺
安全需求
❶
滥用案例
❷
外部审核
❸
风险分析
❹
基于风险的
安全测试
❶
代码审核
（工具）
❸
风险分析
❺
渗透测试
❼
安全操作
需求和
使用案例
体系结构
和设计
测试计划
代码
测试和
测试结果
从应用领域
的反馈

七个接触点

- 1. 代码审核
- 2. 体系结构风险分析
- 3. 渗透测试
- 4. 基于风险的安全测试
- 5. 滥用案例
- 6. 安全需求
- 7. 安全操作

1、代码审核

- 工件：代码
- 发现的风险的例子：在代码的第42行中发现缓冲区溢出
- 所有的软件都会至少产生一种工件——代码。在代码级中，关注的焦点是实现缺陷，特别是那些静态分析工具就更是如此，它们通过扫描源代码能发现一般的弱点
- 代码审核是一种实现安全的软件的必要而不充分的方法。安全缺陷（特别是在C和C++中的安全缺陷）是显而易见的，而体系结构瑕疵则是真正棘手的问题
- 单独进行代码审核是一种特别有用的方法，但是，由于这种类型的审核只能确定缺陷，因此，即使是最好的代码审核也只能发现大约50%的安全问题。仅仅盯着代码是很难（并且几乎是不可能）发现体系结构问题的。现代的系统都由数百万行代码构成，这种方法就更不能奏效了。实现软件安全的完整方法是包括代码审核和体系结构分析的有机组合

2. 体系结构风险分析

- 工件：设计和说明书
- 发现的风险的例子：对关键数据的区分和保护很糟糕；Web服务未能验证调用代码及其用户，并且没有基于正确的上下文来作出访问控制决定
- 在设计和体系结构级中，系统必须是连贯一致的，并提供统一的安全防线。设计人员、架构人员和分析人员应该用文档清晰地记录各种前提假设，并确定可能的攻击。在基于说明书的体系结构阶段和类层次的设计阶段，体系结构风险分析都是必需的。此时，安全分析人员揭示体系结构瑕疵，对它们评级，并开始进行降低风险的活动。忽视这个级别的风险分析会在日后引起严重的问题
- 注意，在软件生命周期的所有阶段中都可能出现风险，因此，强烈地建议采用持续的风险管理方法，并不断地进行追踪和监视风险的活动

3、渗透测试

- 工件：处于环境中的系统
- 发现的风险的例子：在Web接口中处理程序状态的糟糕方法
- 渗透测试非常有用，如果根据体系结构风险分析来设计测试，效果就更好。渗透测试的优点是，它给出了对处于真实运行环境中的实际部署的软件的很好的理解。但是，没有考虑软件体系结构的任何这类测试，或许都不能揭示关于软件风险的任何有用的信息。不能通过预构的应用程序安全测试工具所实施的封闭黑箱测试的软件肯定是非常糟糕的。因此，通过低层次的渗透测试，只能揭示出一点点软件的真实安全状况的信息，但是未能通过封闭的渗透测试，则说明你确实处于很糟糕的状况中
- 渗透测试与进行测试的人员有关，这是它的一个缺陷。应该特别小心那些“改过自新的黑客”，他们改过自新的唯一证明只有一些自我描述。还要小心，网络渗透测试与应用程序或者软件所面临的渗透测试并不相同

4. 基于风险的安全测试

- 工件：单元和系统
- 发现的风险的例子：由于处理数据保护风险而导致可能的大量数据泄漏
- 安全测试必须包含两种策略：（1）用标准功能测试技术来进行的安全功能性测试；（2）以攻击模式、风险分析结果和滥用案例为基础的基于风险的安全测试。一份好的安全测试计划包含这两种策略
- 即便你直接探测一个系统，安全问题也并不总是显而易见的，因此，标准的质量保障方法可能不能揭示所有严重的安全问题。QA是为了保证所有好的事情的发生
- 安全测试是为了保证坏的事情不会发生。像攻击者一样地考虑问题很重要。因此，用关于软件体系结构、一般性攻击和攻击者的心态的知识来指导安全测试是极为重要的

5、滥用案例

- 工件：需求和使用案例
- 发现的风险的例子：易受广为人知的攻击——篡改攻击的影响
- 建造滥用案例是深入攻击者的心理的好办法。类似于使用案例，滥用案例描述了系统在受到攻击时的行为表现；建造滥用案例要求明确地说明应该保护什么、免受谁的攻击，以及保护多长时间

6、安全需求

- 工件：需求
- 发现的风险的例子：没有明确描述数据保护要求
- 必须明确地在需求级中加入安全考量。好的安全需求包括明显的功能安全（比如说，使用实用的加密方法）和突然出现的特性（滥用案例和攻击模式可以很好地捕获它们）。确定和维护安全需求的方法是非常复杂的，应该灵活地处理

7、安全操作

- 工件：实际部署的软件
- 发现的风险的例子：没有足够的日志记录以追踪某个已知的攻击者
- 软件安全可以从网络安全中借鉴很多方法。经过有效组合的安全操作允许和鼓励网络安全专业人员积极应用接触点，提供开发团队可能缺乏的经验和安全智慧。在增强系统的安全状况的过程中，身经百战的操作人员认真地设置和监视实际部署的系统。不论设计和实现的力度如何，都会出现攻击，因此，理解导致攻击成功的软件的行为就是一种重要的防御技术。通过理解攻击和攻击程序而获得的知识应该再应用到软件开发中

大纲

- 软件与软件安全
- 软件安全的三个支柱
- ✓ 软件安全与其他相关领域的关系
- 软件安全工具简介

软件安全与其他相关领域的关系

- 软件工程
- 软件保证
- 软件质量
- 软件可靠性
- 软件容错
- 应用安全

软件工程

- 软件安全不是孤立的学科，与软件工程关系最密切
- 软件工程是一门研究如何使用系统化、规范化、数量化等工程原则和方法去进行软件的开发、运行和维护的学科
- 软件工程采用工程的概念、原理、技术和方法来开发维护软件，把经过时间考验而证明正确的管理方法和最先进的软件开发技术结合起来，应用到软件开发和维护过程中，来解决软件危机问题，生产出无故障的、及时交付的、在预算之内的和满足用户需求的软件

软件工程的内容

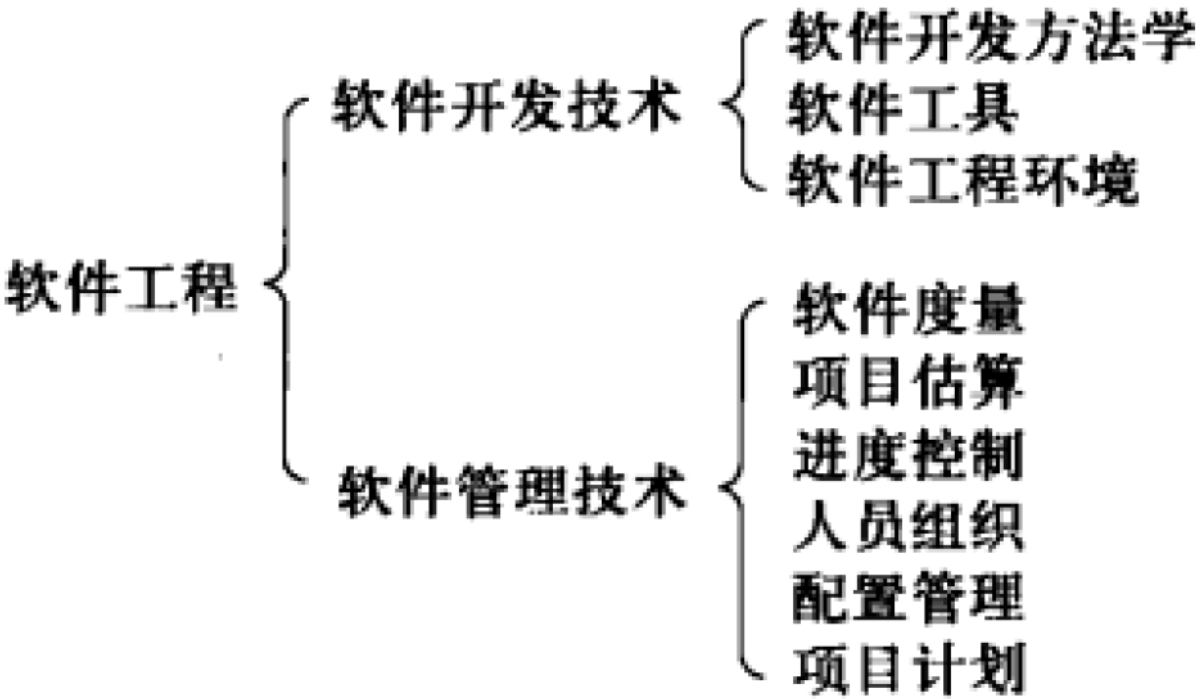
- 和任何工程方法一样，软件工程以质量为关注焦点，全面质量管理及相关的现代管理理念为软件工程奠定强有力的根基
- 软件工程的三要素
  - 过程
  - 方法
  - 工具

软件工程的内容

- 软件工程的基础是过程层。软件工程过程是为获得软件产品，在软件工具支持下由软件人员完成的一系列软件工程活动
  - 过程层将方法和工具结合在一起
- 方法层提供了软件开发的各种方法
  - 项目计划与估算方法、需求分析和设计方法、编程、测试方法及运维方法
- 工具层为软件工程方法提供了自动的或半自动的软件支撑环境
  - 计算机辅助软件工程（CASE）

软件工程的内容

- 软件工程是一门涉及内容广泛的学科(一级学科)，所依据的基础理论极为丰富，包括数学、计算机科学、经济学、工程学、管理学和心理学等其他学科，其研究的内容包括软件开发技术和软件管理技术



软件保证(Software Assurance)

- 软件保证是计划和系统行动集合来确保软件过程和产品符合需求，标准和规程
- 软件保证要达到如下两个目标
  - 可信性(trustworthiness)，即没有可利用的弱点存在，无论是来自敌意的还是无意的企图
  - 执行可以预见(predictable execution)，可证明的信任软件在执行的时候是依照所希望的那样进行工作。这些行为包括：需求分析、测试、验证，以及报告
- 软件保证关系到两个软件属性
  - 质量(quality)，即软件保证是软件质量保障的简短形式
  - 可靠性(reliability)，特别是其中最重要的内容--安全(safety)

软件质量

- 软件质量，是贯穿软件生存期的一个极为重要的问题，是软件开发过程中所使用的各种开发技术和验证方法的最终体现
- 软件质量的定义：满足软件的各项精确定义的功能和性能需求，遵守文档化的开发标准，需要相应地给出或设计一些质量特性及其组合。如果这些质量特性及其组合都能在产品中得到满足，则这个软件产品质量就是高的

软件质量

- 软件质量反映了以下三方面的问题：
  - 软件需求是度量软件质量的基础，不符合需求的软件就不具备质量
  - 在各种标准中定义了一些开发准则，用来指导软件人员用工程化的方法来开发软件。如果不遵守这些开发准则，软件质量就得不到保证
  - 往往会有一些隐含的需求没有明确地提出来，例如，软件应具备良好的可维护性，如果软件只满足那些精确定义了的需求而没有满足这些隐含的需求，软件质量也不能保证

软件质量保证

- 软件质量保证(Software Quality Assurance)
  - 通过对软件产品有计划地进行评审和审计来验证软件是否合乎标准的系统工程
- SQA的活动原则
  - 确保SQA要自始至终有计划地进行
  - 审查软件产品是否遵守适用的标准、规程和要求，并得到客观验证
  - SQA和结果要保证全员参与，沟通顺畅
  - 逐级解决不符合问题

软件质量的属性

- 可靠性(reliability)：产品在规定的条件下和规定的时间内，完成规定功能的能力
- 软件可靠性(software reliability)：在规定环节下及时间内，软件不引起系统失效的概率
- 可维护性(maintainability)：产品在规定的条件下和时间内，按规定的程序和方法进行维修、保持或恢复到规定的状态的能力
- 可用性(available)：产品在任一随机时刻需要和开始执行任务时，处于可工作或可使用状态的程度
- 安全性(safety)：将伤害或损坏的风险限制在可接受水平内的能力
- 机密性(confidentiality)：避免未经许可泄露信息的能力
- 完整性(integrity)：避免不适当地更改的能力

软件质量的特性

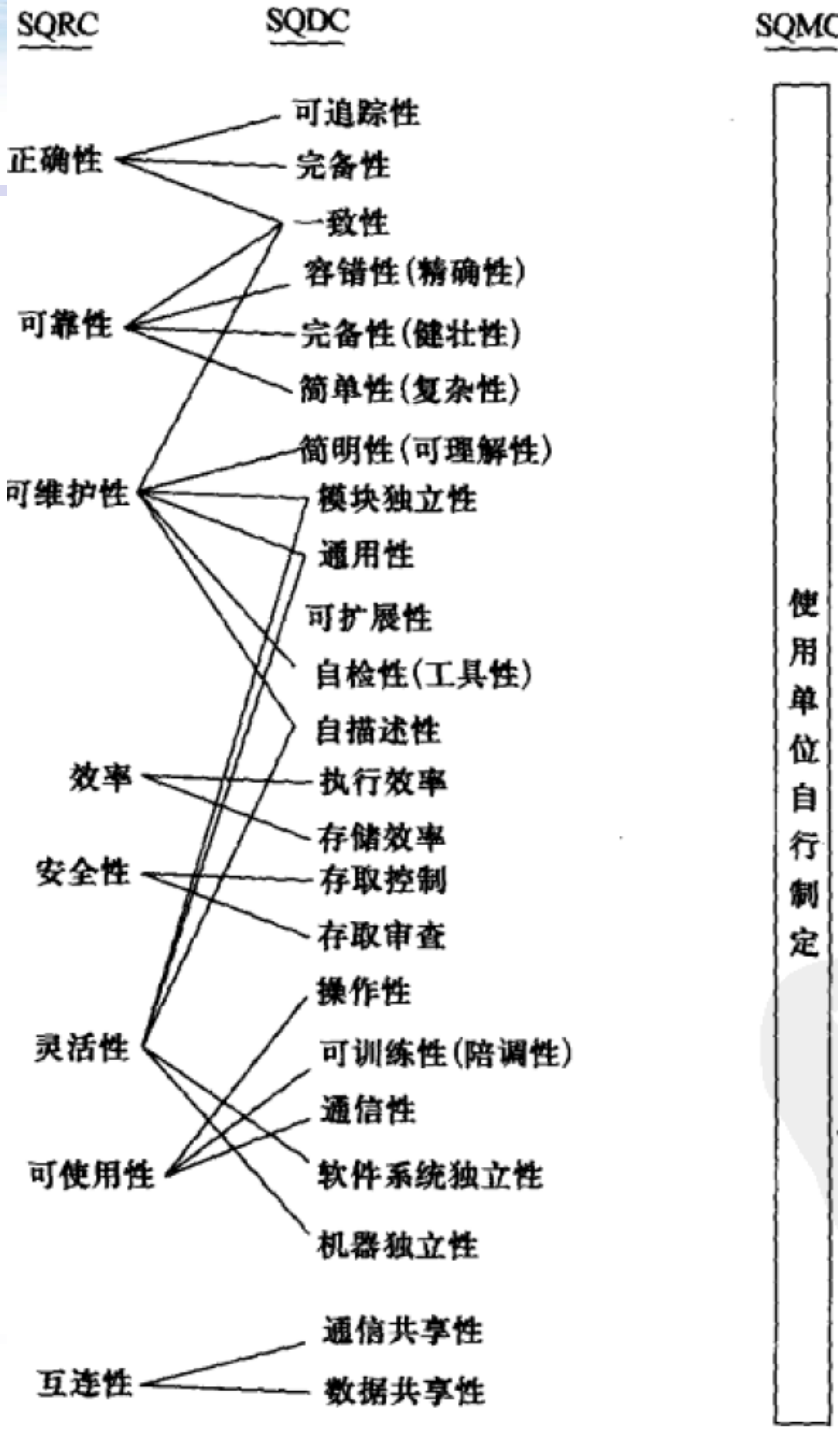
- 软件质量特性反映了软件的本质。讨论一个软件的质量，问题最终要归结到定义软件的质量特性（用软件质量模型来描述）
- 已有软件质量模型的共同特点是把软件质量特性定义成分层模型
- 在这种分层模型中，最基本的叫做基本质量特性，它可以由一些子质量特性定义和度量。二次特性在必要时又可由它的一些子质量特性定义和度量
- 按照ISO/TC97/SC7/WG3/1985-1-30/N382，软件质量度量模型由三层组成

软件质量度量模型

- 高层：软件质量需求评价准则(SQRC)
- 中层：软件质量设计评价准则(SQDC)
- 底层：软件质量设计度量准则(SQMC)

- 对高层和中层建立国际标准，低层由使用单位自行制定

- ISO9126质量特性国际标准
  - 质量特性(6个)：功能性、可取性、可维护性、效率、可使用性、可移植性
  - 质量子特性(21个)
  - 度量



软件可靠性

- 软件可靠性工程(Software Reliability Engineering)是预计、测量和管理以软件为基础的系统的可靠性，最大限度地满足用户要求的应用科学
- 从软件特性和用户需求分析开始，软件可靠性工程贯穿软件生命周期的全过程，主要包括下面三个方面的内容
  - 为满足用户对软件可靠性的要求，必须给出关于软件可靠性的规格说明，在做软件可靠性设计时，甚至要在其它软件质量指标间做出某些折中，如可适当增加开发成本，延长测试时间等以求得更高的软件可靠性
  - SR的量测与分析技术，这是必要条件
  - 软件工程中一整套保证软件开发自的方法和技术

软件可靠性

- 软件可靠性工程的研究范围
  - 软件可靠性定量评测
  - 软件可靠性的设计与管理
  - 软件可靠性保证技术
- 软件可靠性模型是指为预计或估算软件的可靠性所建立的结构和数学模型。建立可靠性模型是为了将复杂系统的可靠性逐级分解为简单系统的可靠性，以便定量预计、分配、估算和评价复杂系统的可靠性

软件可靠性模型

- 一般软件可靠性模型分两大类，即软件可靠性结构模型和软件可靠性预计模型
  - 可靠性结构模型：依据系统结构逻辑关系，对系统的可靠性特征及其发展变化规律做出可靠性评价。此模型既可用软件可靠性综合评价，又可用于软件可靠性分解。
  - 可靠性预计模型：用来描述软件失效与软件缺陷的关系，借助这类模型，可以对软件的可靠性特征作出定量的预计和评

软件容错

- 软件容错性是指软件运行时，能对非正常因素引起的运行错误给出适当处理或信息提示，使软件运行正常结束
- 容错技术包括以下几个主要方面
  - 故障检测技术
  - 故障恢复技术
  - 破坏估计
  - 故障隔离技术
  - 继续服务

应用安全

- 应用安全关注应用系统的安全性，或者网络应用层的安全性。包括如何保护软件以及软件所运行的系统
- 应用安全主要针对发现和修改已知的安全问题，而软件安全是为了安全目标而设计、构建、和测试软件的过程。软件安全实践者试图构建一个先念的能抵御各种攻击的安全软件

大纲

- 软件与软件安全
- 软件安全的三个支柱
- 软件安全与其他相关领域的关系
- ✓ 软件安全工具简介

软件安全工具介绍

- 反汇编器
  - IDA Pro、W32Dasm
- 反编译器
  - Hex Ray、baksmali
- 调试器
  - GDB、OllyDBG、WinDBG、SoftICE
- 系统监控
  - IceSword (PJF)、 DarkSpy (CardMagic)
  - Process Explorer、Process Monitor(Filemon & Regmon)
  - TCPView、Portmon
- 修补和转储
  - UltraEdit、 Hex Workshop、Dumpbin、PEview

小结

- 软件安全的基本概念
- 确保软件安全的主要方法
- 软件与其他领域的关系
- 软件安全工具