软件安全

徐剑

信息安全系

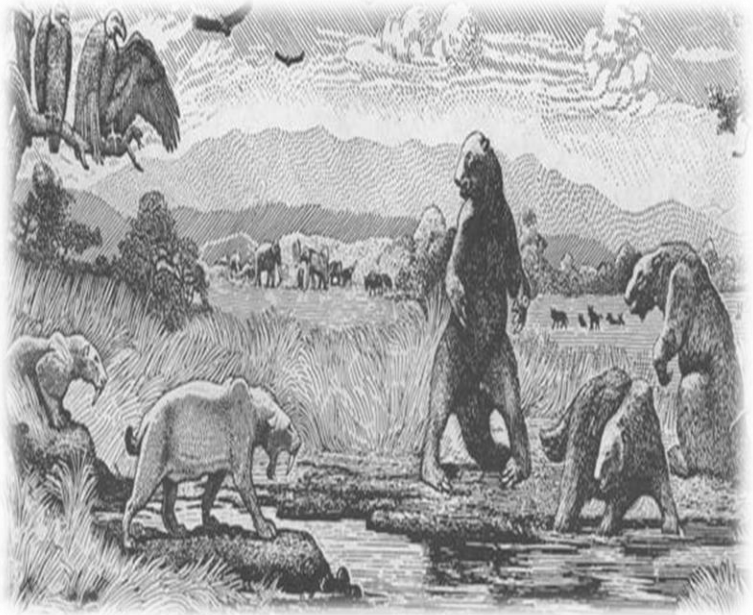
xuj@mail.neu.edu.com

第2章 软件安全开发模型

- 软件危机与软件工程
- 软件安全开发生命周期
- SDL模型
- BSI模型
- SAMM模型

软件危机与软件工程

软件危机



- 过去几十年的大型系统开发就犹如这样一个焦油坑（左图），很多大型和强壮的动物在其中剧烈地挣扎。他们中大多数开发出了可运行的系统——不过，其中只有非常少数的项目满足了目标、时间进度和预算的要求。
- 各种团队，大型的和小型的，庞杂的和精干的，一个接一个淹没在了焦油坑中。表面上看起来好像没有任何一个单独的问题会导致困难，每个都能被解决，但是当它们相互纠缠和累积在一起的时候，团队的行动就会变得越来越慢。对问题的麻烦程度，每个人似乎都会感到惊讶，并且很难看清问题的本质。不过，如果我们想解决问题，就必须试图先去理解它。

软件危机

- 软件危机是在软件产品的开发和维护中产生的一系列严重问题。爆发于上世纪60年代并延续至今。典型表现包括：
  - 软件延期交付（开发时间）
  - 开发成本远超预算（开发成本）
  - 产品存在质量缺陷，未达到用户要求或造成巨大损失（产品质量）
  - 产品适应性维护的难度和成本过高，服役期过短（产生寿命）
  - 生产效率远低于硬件生产水平……
- 典型案例——《软件开发的滑铁卢》电子工业出版社
  - 1996年数据转换溢出致阿丽亚娜5型火箭爆炸，50亿美元损失。
  - 1982年美利坚银行MasterNet CRM系统,预算2000千万，超期27个月，超预算3000万，软件故障导致客户抽走资金40亿。
  - Standish集团95年统计报告：全美8000个软件项目，84%超期、超预算且未完成预定要求，30%中途取消。

软件危机

- 软件危机根源——软件开发的固有困难（《人月神话》）
  - 复杂性：问题本身的复杂性；实现目标的大量细节造成的复杂性。
    - Win95 1100万行源码；Win98 1800万行；WinXp 3500万行；
    - Vista 5000万行，相关开发人员近9000人，返工代码近60%。
  - 一致性：软件必须遵守已经建立的约束、习惯、系统环境和结构。
    - Locus被IBM收购后，为使其网络OA系统兼容IBM网关，改写了系统中近60%的代码。
  - 可变性：商业环境、业务规则、技术及硬件更新等因素的变化将迫使软件随之调整。
    - Yahoo！个人搜索引擎为将其竞争对手所推出的新特性纳入其中，在发布前进行了三次实质性的返工。
    - 安信达的物流管理软件为适应货运市场的灵活性重新规划架构，开发工作量超过预期60%。
  - 不可见性：不具备外观特征，难以交流、审核和管理。
    - IBM OS/360系统，耗资过亿，4000多个模块，100万行代码，交付后发现的故障超过了2000个。

软件危机

- 应对软件危机
  - 目标：控制开发难度、提高产品质量
  - 途径：借鉴其他行业的先进经验
    - 类比——建造摩天大楼
      - 任务：计划、设计、施工（土建、电气、消防…）、销售…
      - 主体建设者：各类施工人员
      - 产品质量：基本令人满意
  - 措施：
    - 建立适合软件开发特点的系统的、可重复的工程化模式和方法

软件工程

- 产生背景
  - 为应对软件危机，1968年北大西洋公约组织（NATO）在德国Garmish召开的学术会议上，Feitz Bauer首先提出了“软件工程”概念。后逐步发展为以研究软件生产客观规律性的工程学科。
  - 软件工程的定义
    - Software Engineering is the establishment & use of sound engineering principles in order to obtain economically software that is reliable & work efficiently on real machines。——软件工程是为了经济地获得可靠的和能在实际机器上高效运行的软件而确立和使用的健全的工程原理（方法）。
  - 工程的含义
    - 以某组设想的目标为依据，应用有关的科学知识和技术手段，通过一群人的有组织活动将某个或某些现有实体转化为具有预期使用价值的人造产品过程 。

软件工程

- 研究内容
  - 软件生产的客观规律性
  - 有关工程化软件生产的概念、原则、方法、技术和工具
  - 管理和支持软件生产活动的方法和措施
- 知识结构
  - 综合学科：计算机科学、经济学、管理学、心理学……
  - 2005 ISO/IEC JTC1发布《SWEBOK指南v0.95》，将软件工程的知识结构划分为10个知识区域。
    - 软件需求、软件设计、软件构造、软件测试、软件维护；
    - 软件配置管理、软件工程管理、软件工程过程、软件工程工具与方法、软件质量

软件工程

- 软件工程模型

CASE工具
（Computer Aided（or Assisted）Software Engineering）

软件开发的支持环境

软件开发方法

软件开发的技术指导

软件过程

软件开发的分工组织

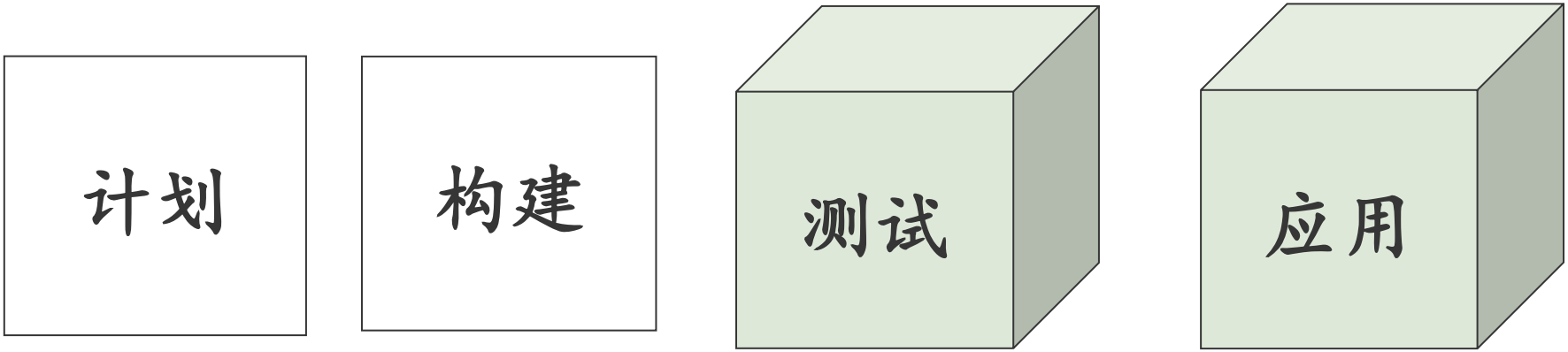
产品质量

关注焦点

软件安全开发生命周期

软件安全开发生命周期

- 对待软件安全问题的传统方式

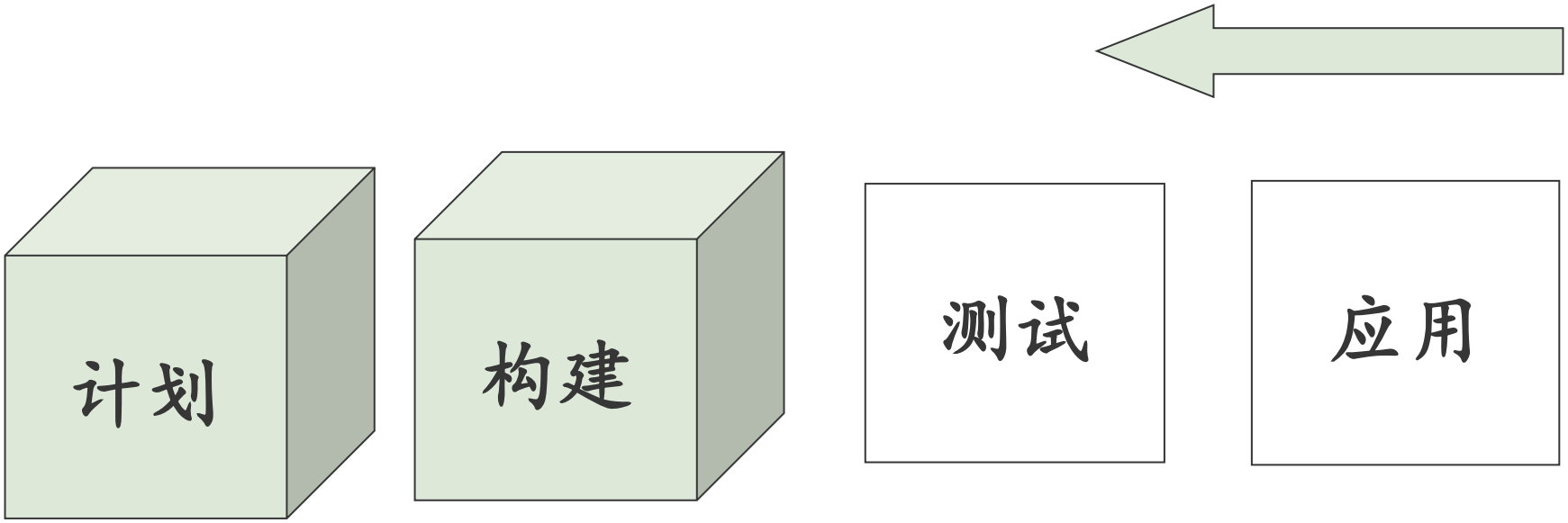


防火墙
入侵检测
渗透测试

这种“亡羊补牢”的安全问题处理方式，其代价是极其高昂的。众多的企业软件开发项目实践和相关研究成果表明，在软件发布后在进行安全漏洞修复所需的成本往往是在软件设计和编码阶段进行修复的几十倍。

软件安全开发生命周期

- 从软件开发生全命周期角度来考虑安全问题

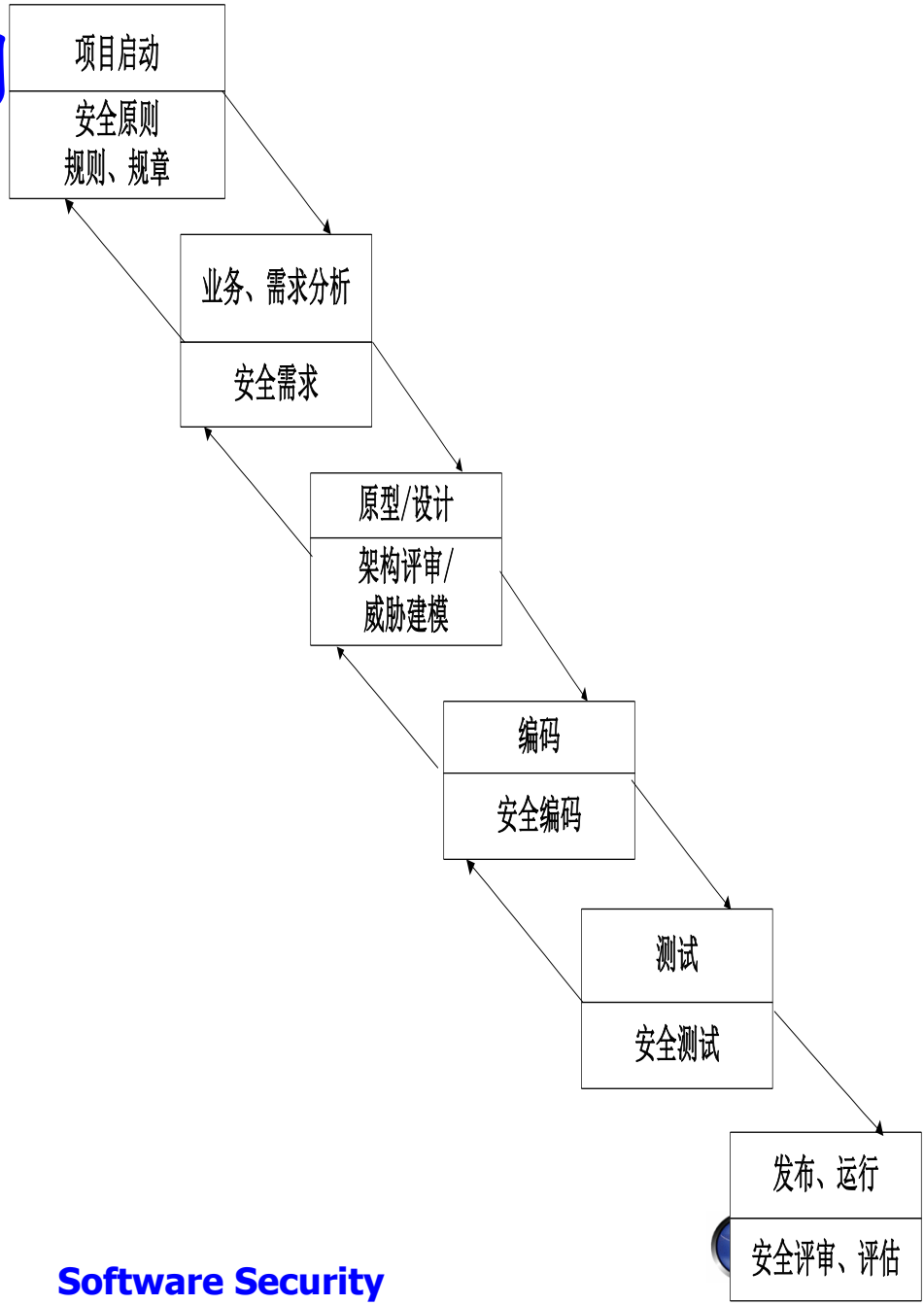


静态源代码分析
架构风险分析
安全需求

未雨绸缪的方式

软件安全开发生命周期

- 软件安全开发生命周期（SDL，Secure Development Lifecycle）
  - 将安全内置到全软件生命周期，即通过软件开发的各个步骤来最大限度地确保软件安全
  - 核心思想：在软件开发生命周期的每个阶段融合安全要素，用以解决软件开发各阶段可能出现的安全问题
  - 最终目标：使软件能够按照开发者的设定目标执行，并且在受到恶意攻击的情况下依然能够继续正确运行



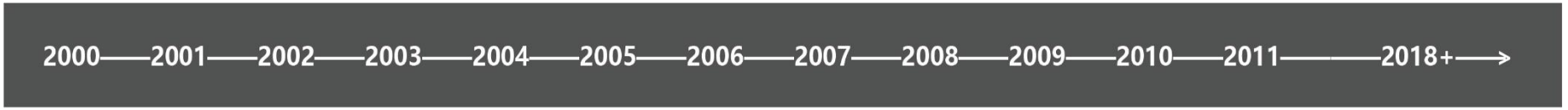
SDL模型

SDL模型

- SDL(Secure Development Lifecycle)模型
  - 2002年1月，微软公司推出了“可信赖计算计划”（Trustworthy Computing），SDL是其中的重要成果；
  - 2004年，SDL 2.0
    - 微软公司推出的第一个官方版本的SDL规范
    - 该版本给出了在.NET和Windows安全改进期间使用过的技术和改进方法，包括威胁建模、静态分析和安全审查等。
  - Vista是微软公司第一个采用SDL过程开发的操作系统
  - 2007年，SDL 3.2
    - 微软第一次发布的SDL的公开版本
    - 向软件行业公开了用于保护Microsoft软件的过程和技术
  - SDL的目标
    - 减少安全漏洞和隐私问题的数量（数量）
    - 降低仍然存在的漏洞的严重性（危害程度）

SDL模型

- SDL发展历程



- Growth of home PC's
- Rise of malicious software
- Incrasing privacy concerns
- Internet use expansion

- Bill Gates' TwC memo
- Microsoft security push
- Microsoft SDL released
- SDL becomes mandatory policy at Microsoft
- Windows XP SP2 and Windows Server 2003 launched with security emphasis

- Windows Vista and Office 2007 fully integrate the SDL
- SDL released to public
- Data Execution Prevention (DEP)&, Address Space Layout Randomization (ASLR) introduced as features
- Threat Modeling Tool

- Microsoft joins SAFECode
- Microsoft Establish SDL Pro Network
- Defense information Systems Agency (DISA) & National Institution Standards and Technology (NIST) specify featured in the SDL
- Microsoft collaborates with Adobe and Cisco on SDL practices
- SDL revised under the Creative Commons License

- Additional resources dedicated to address prjected growth in Mobile app downloads
- Indsty-wide accptance of practices aligned with SDL
- Adaption of SDL to new technologies and changes in the threat landscape
- Increased industry resources to enable global secure development adoption

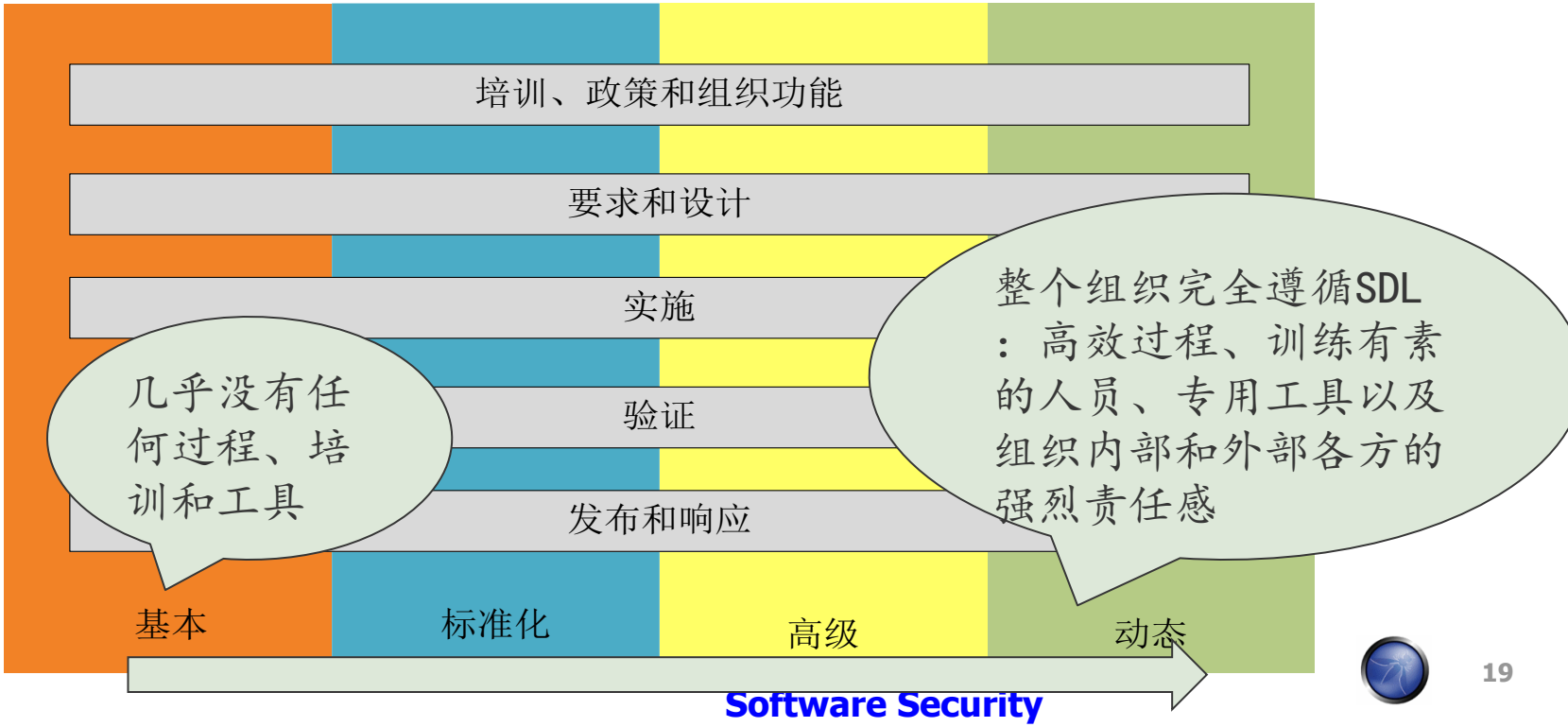
SDL模型

- SDL的核心概念

SDL模型

- SDL优化模型（具有功能和成熟度水平）
  - 软件安全是一个高度复杂的问题，将安全整合到软件开发过程，如果方式不当，将会影响软件开发进度、降低软件质量、增加软件开发成本。
  - 根据开发团队的成熟度水平（基本、标准化、高级、动态）确定软件安全实施优先级，可以控制软件安全开发实践过程中的一些因素的影响。

SDL优化模型



SDL模型

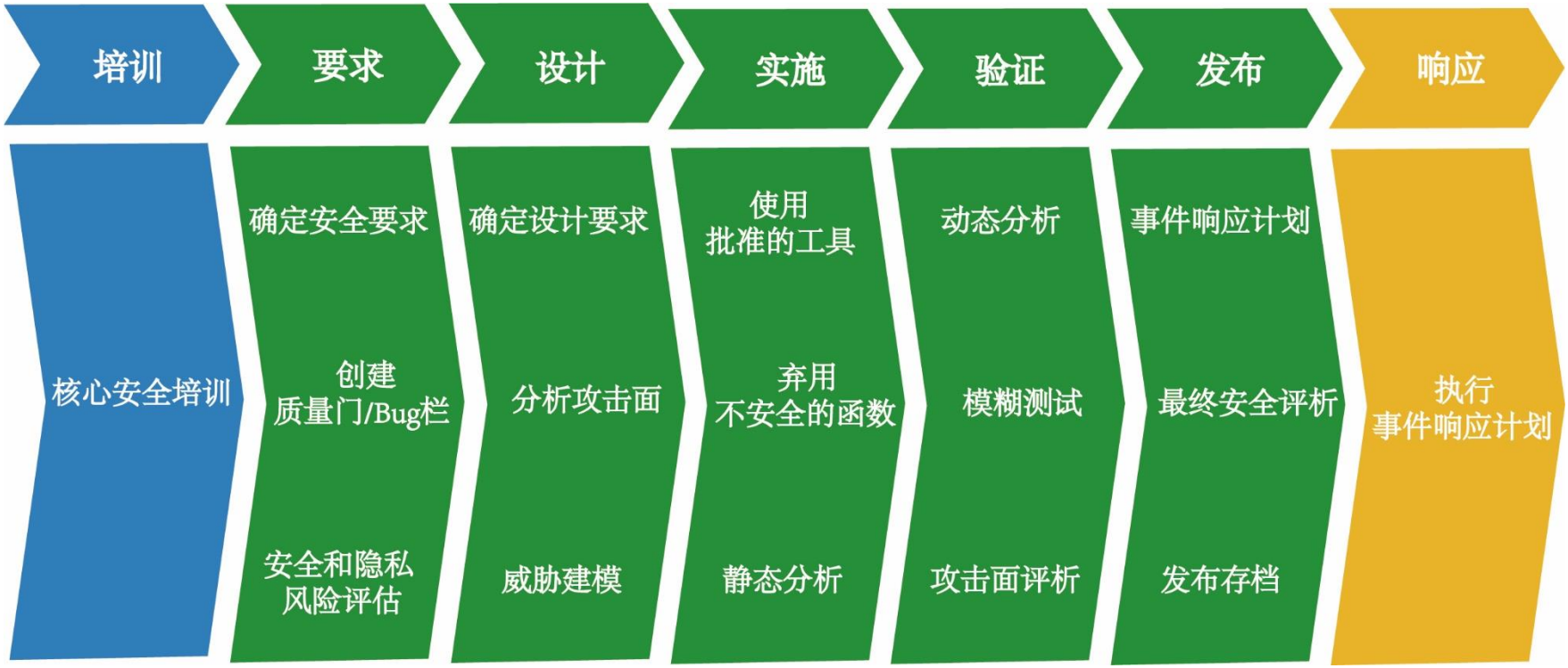
- SDL的适应性(具备以下一个或多个特征)

SDL模型

- SDL中安全人员的角色与职责
  - 评析者/顾问
    - 对项目安全和隐私进行监督，有权接受或拒绝项目团队的安全和隐私计划
    - 安全顾问/隐私顾问（组织内部/外部人员均可）
      - **审计官**：不受项目团队干扰的情况下，监控软件开发过程的每个阶段，保障每个安全需求都成功实现；并自主证明软件开发过程是否符合安全和隐私方面要求。
      - **专家**：拥有软件安全开发专业知识的顾问
    - 顾问角色组合（安全顾问角色和隐私顾问角色组合）
  - 团队负责人
    - 负责协商、接受和跟踪最低安全和隐私需求
    - 安全负责人/隐私负责人：负责确保软件解决了所有安全问题，负责协调和跟踪项目安全问题。
    - 角色组合

SDL模型

- 简化的SDL安全活动



SDL模型

- 简化的SDL安全活动-安全培训（所有成员必须参加）

安全设计	威胁建模	安全编码	安全测试	隐私保护	高级概念
• 减小攻击面 • 深度防御 • 最小权限原则 • 安全默认设置	• 威胁建模概述 • 威胁模型的设计意义 • 基于威胁模型的编码约束	• 缓冲区溢出 • 整数算法错误 • 跨站点脚本 • SQL 注入	• 安全测试与功能测试之间的区别 • 风险评估 • 安全测试方法	• 隐私敏感数据的类型 • 隐私设计最佳实践 • 风险评估 • 隐私开发最佳实践 • 隐私测试最佳实践	• 高级安全设计和体系结构 • 可信用户界面设计 • 安全漏洞细节 • 实施自定义威胁缓解

SDL模型

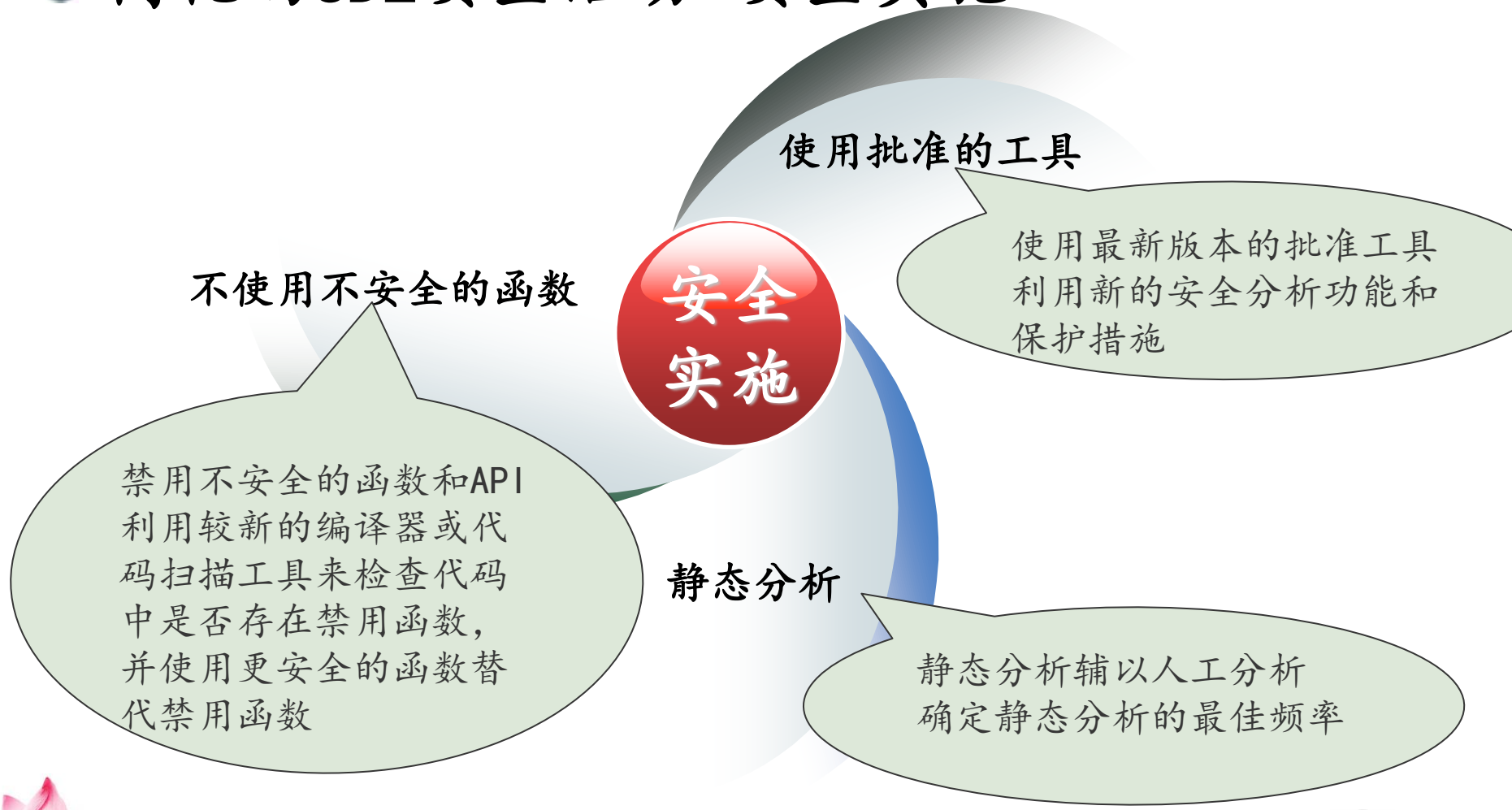
- 简化的SDL安全活动-安全需求
  - 安全需求阶段需要关注的问题
    - 质量/缺陷（Bug）门槛
      - 确立安全和隐私的最低可接受级别
      - 项目团队必须协商确定每个开发阶段的Bug门槛（数量&严重性）
      - Bug门槛用于定义安全漏洞的严重性阈值
      - Bug门槛一经设定，则必须坚持
    - 安全和隐私风险评估
      - 项目的哪些部分在发布前需要威胁模型
      - 项目的哪些部分在发布前需要进行安全设计分析
      - 项目的哪些部分（如果有）需要由不属于项目团队且双方认可的小组进行渗透测试
      - 是否存在安全顾问认为有必要增加的测试或分析要求，以缓解安全风险
      - 安全模糊测试要求的具体范围是什么？
      - 基于以下准则回答隐私对评级的影响
        - P1 高隐私风险/ P2 中等隐私风险/ P3 低隐私风险

SDL模型

- 简化的SDL安全活动-安

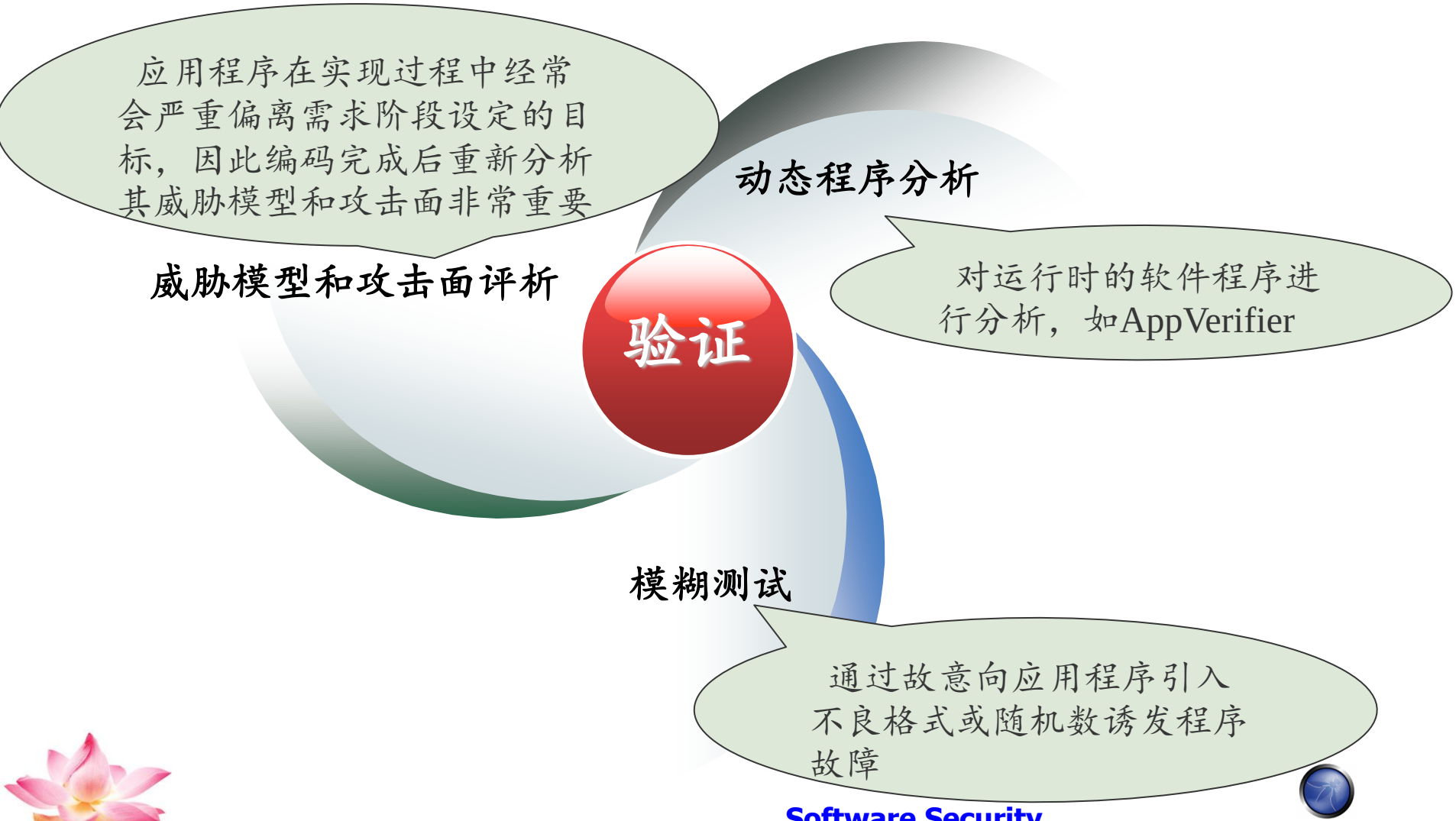
SDL模型

- 简化的SDL安全活动-安全实施



SDL模型

- 简化的SDL安全活动-验证



SDL模型

- 简化的SDL安全活动-发布及响应

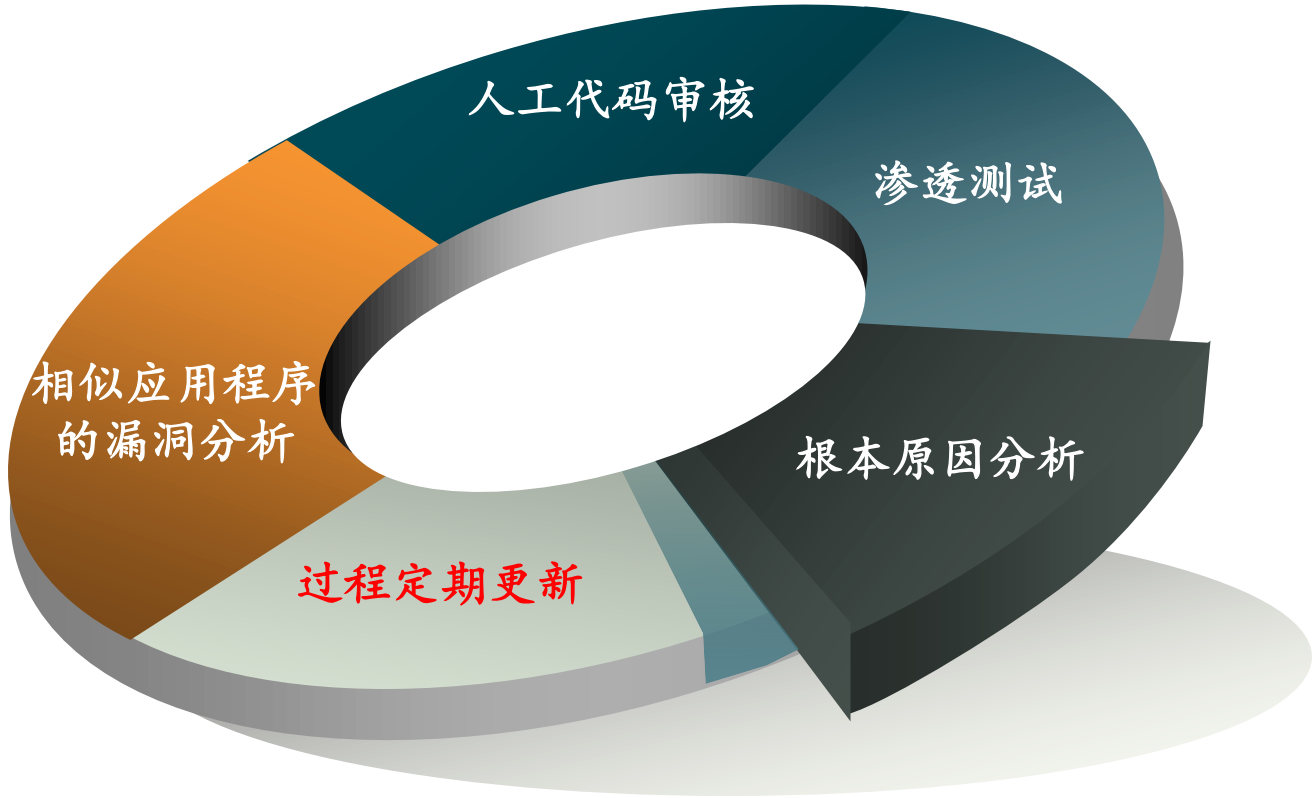
- 单独指定的可持续工程（Sustainable Engineering，SE）团队
- 与决策机构的电话联系（7*24）
- 针对从组织中其他小组继承的代码的安全维护计划
- 针对获得许可的第三方代码的安全维护计划，包括文件名、版本、源代码、第三方联系信息以及要更改的合同许可

- 最终安全分析（FSR）是在软件发布之前仔细检查对软件执行的所有安全活动
- FSR的三种结果
  - ✓ 通过 FSR
  - ✓ 通过 FSR 但有异常
  - ✓ 需上报问题的 FSR

- 满足安全和隐私保护要求，才能发布产品
- 必须对所有相关信息和数据进行存档
  - ✓ 规范
  - ✓ 源代码
  - ✓ 二进制文件
  - ✓ 威胁模型
  - ✓ 文档
  - ✓ 紧急响应计划
  - ✓ 任何第三方软件的许可证和服务条款

SDL模型

- SDL中的可选安全活动



BSI模型

BSI模型

- **BSI（Build Security In）模型**
  - 内建安全模型或软件安全接触点模型
  - Gary McGraw博士及其合作者提出的以**风险管理**、**软件安全接触点**和**安全知识**为基础的软件安全开发模型
  - 与SDL模型一样，BSI模型强调使用**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，为软件开发人员、架构师和安全从业者提供软件安全开发实践工具、指导方法和其他资源，以便其在开发的每一阶段使用BSI方法来构建安全软件，确保将安全作为软件的一个有机组成部分
  - BSI已成为美国国土安全部软件保障计划中的一个项目
    - https://www.us-cert.gov/bsi

An official website of the United States government Here's how you know

CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY

Search

CISA.gov | Services | Report

Alerts and Tips | Resources | Industrial Control Systems

BUILD SECURITY IN

Setting a higher standard for software assurance

This document is part of the US-CERT website archive. These documents are no longer updated and may contain outdated information. Links may also no longer function. Please contact info@us-cert.gov if you have any questions about the US-CERT website archive.

Topics

- Best Practices Articles
- Knowledge Articles
- Tools Articles

Build Security In

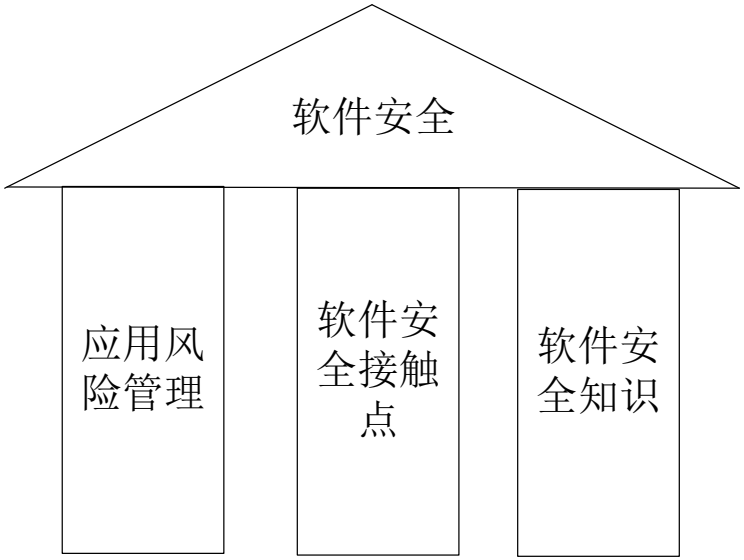
Build Security In / Software & Supply Chain Assurance content is no longer updated.
The articles are provided here for historical reference.
Suggested resource: https://www.cert.org/cybersecurity-engineering/

Build Security In was a collaborative effort that provided practices, tools, guidelines, rules, principles, and other resources that software developers, architects, and security practitioners can use to build security into software in every phase of its development. Software assurance (SwA) is the level of confidence that soft ware is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life cycle, and that the software functions in the intended manner.

BSI模型

BSI模型-软件安全三根支柱

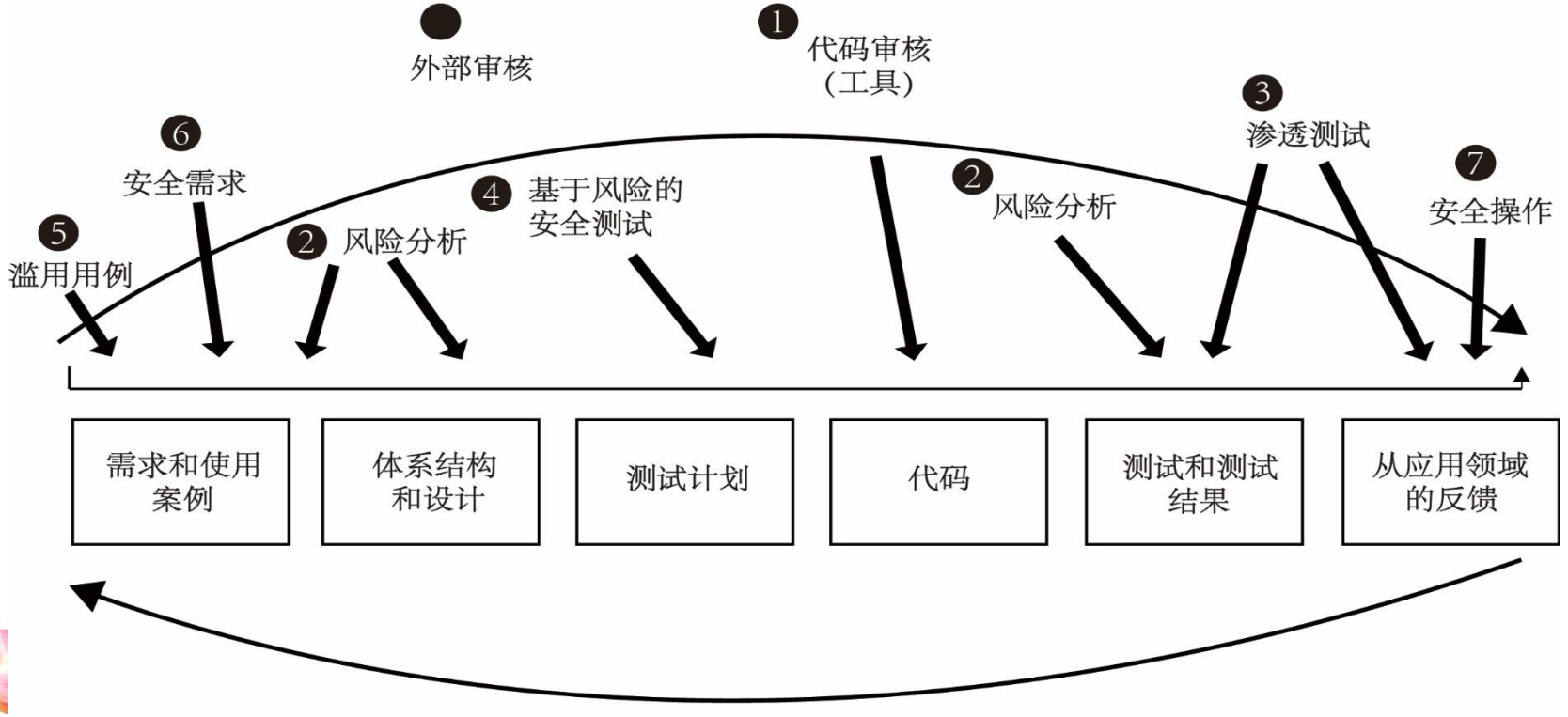
- **应用风险管理**：将**减轻风险**作为一种贯穿整个软件开发生命周期的指导方针，BSI模型强调软件生命周期中风险管理的重要性，并要求**风险管理**框架贯穿整个开发过程
- **软件安全接触点**：以优秀的软件工程方法为基础，在整个软件开发生命周期中都认真、明确地考虑安全问题，并综合考虑软件安全的两个方面—攻击与防御，从不同角度提供保障安全的行为方式
- **软件安全知识**：强调对软件安全经验和与软件安全相关的专业技术进行收集汇总，对软件开发人员进行培训，并将软件安全接触点实际运用到项目过程中



BSI模型

BSI模型-软件安全接触点

- 在BSI模型中，**探索和描述一组软件安全的最优方法称为软件安全接触点**
- 软件安全接触点强调了使安全成为软件开发必需的部分而必须进行的保障活动
- 要保证软件安全，软件项目就必须在整个软件生命周期中都应用接触点对系统进行分析与审核，随着项目的进展不断地实施安全保证。
- **接触点与具体设计过程无关**，可以在不同阶段不止一次地被循环使用。



BSI模型

- **BSI模型-软件安全接触点**
  - **代码审核**
    - 代码审核是一种有效的发现代码缺陷的手段，它关注的重点是**编码过程中引入的缺陷**
    - 代码审核只是一种实现安全软件的必要而不充分的方法，即使是最好的代码审核也只能发现大约50%的安全问题
    - **代码审核是很难（并且几乎是不可能）发现体系结构问题的**
  - **体系结构风险分析**
    - 用文档清晰地记录下各种前提假设，并确定可能的攻击
    - 分析可能的攻击，并提供一致的安全防护措施
    - 对所发现的体系结构安全风险进行评级，并开展降低风险的相关活动
  - **渗透测试**
    - 在开展渗透测试之时，一定要认真考虑软件架构
    - 考虑软件渗透测试与网络渗透测试的差异性

BSI模型

BSI模型-软件安全接触点

- 基于风险的安全测试
  - 安全测试包括的内容
    - 用标准功能测试技术进行的安全功能测试（**安全功能测试**）
    - 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（**安全的功能测试-测试功能的安全性**）
  - 安全测试确保非预期的功能或事件不会发生，需要像攻击者一样思考问题
- 滥用案例（详见第3章 软件安全需求与设计）
  - 滥用用例非常地有利于深入理解攻击者的心理
  - 滥用用例描述了系统在受到攻击时的行为表现
  - 建造滥用案例要明确地说明应该保护什么、免受谁的攻击以及保护的时间长度

BSI模型

- **BSI模型-软件安全接触点**
  - **安全需求**
    - 显性的功能安全（例如，基于应用密码学的安全功能）
    - 应对突发情况的特性（这些特性可以通过滥用案例和攻击模式来获得）
  - **安全操作**
    - 要求软件公司不同部门、不同职位人员在实践中应用安全接触点时密切合作、协同一致地工作
    - 在增强系统安全的过程中，操作人员应认真地设置和监视实际部署的系统

SAMM模型

SAMM模型

- 模型概述
  - 软件保证成熟度模型（Software Assurance Maturity Model，SAMM），由独立软件安全顾问Pravir Chandra创建，并由Fortify软件公司资助。
  - 目前，SAMM作为一个开放项目，由OWASP维护。
  - SAMM是一个开放的框架，用以帮助组织制定并实施针对组织所面临的来自软件安全的特定风险策略
  - SAMM侧重于建立一种迭代的安全保证计划，根据组织的行为随着时间的推移而逐渐改变，软件的安全保证也应该持续改进，以保证信息系统对于业务功能支持的不断进化

SAMM模型

- 模型概述
  - 由SAMM提供的资源的作用
    - 评估一个组织已有的软件安全实践
    - 建立一个迭代的、权衡的软件安全保证计划
    - 证明安全保证计划带来的实质性改善
    - 定义并衡量组织中与安全相关的措施
  - SAMM以灵活的方式定义，以使它可以被大、中、小型组织使用于任何类型的软件开发过程中
  - SAMM模型可适用于全组织范围内，既可以是整个组织，也可以是一个单一的项目

SAMM模型

- 模型原则
  - 一个成功的软件安全计划，应当详细说明每一个小的迭代步骤，以使其能够提供有形的保证收益，并向项目的长期目标前进
  - 一个软件安全框架必须是灵活的，并允许组织可以根据他们的风险承受能力和他们开发和使用软件的方式，去改进他们选择的安全框架
  - 与安全措施相关的指导必须是规范的，即所有建立和评估保证计划的步骤应是简单的、明确的和可测量的

SAMM模型

- 模型功能
  - SAMM模型的基础是软件开发过程中面向安全保证的四个核心业务功能，即监管、构造、确认和部署
  - 每个业务功能都有相应的安全实践，每个安全实践都是一个与安全相关的措施，保证相关业务功能的实现
  - SAMM的安全实践都是改进软件开发业务功能的独立部分。
  - 对于每一个实践，SAMM都设置了三个成熟度等级和一个隐含的为零的起点。在每个成熟度等级上，组织可以进行的各种实践，以降低软件的安全风险并加强软件开发过程的安全质量保证。
    - 0级：隐含起点，代表在实践中的措施尚未实现；
    - 1级：对安全实践有了初步了解，并有专门的设计和使用；
    - 2级：提高了安全实践的效率和（或）有效性；
    - 3级：在一定规模上综合、有效地掌握了安全实践。

SAMM模型

- SAMM模型功能