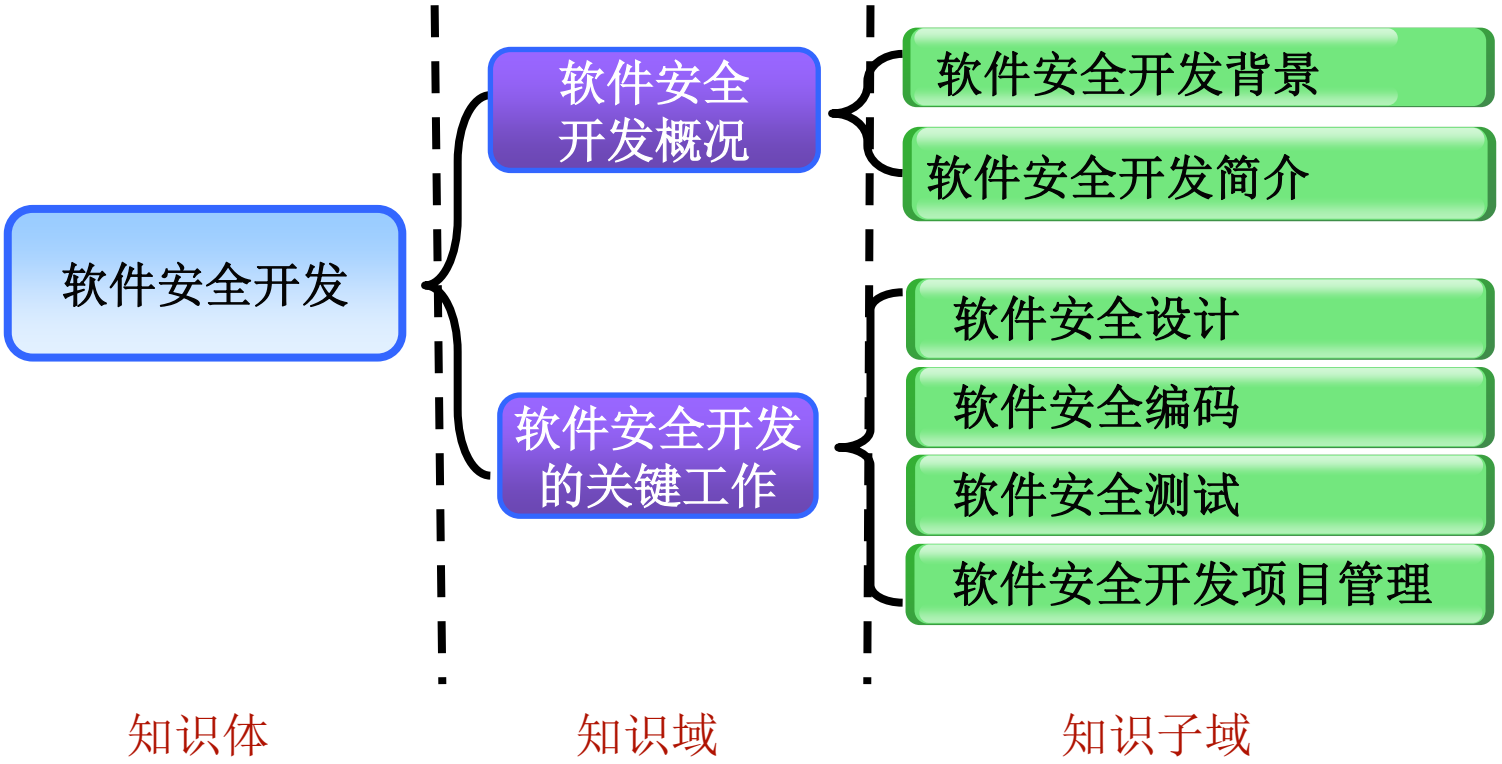
软件安全开发

培训机构名称

讲师名字

课程内容


软件安全开发
软件安全开发概况
软件安全开发的关键工作
软件安全开发背景
软件安全开发简介
软件安全设计
软件安全编码
软件安全测试
软件安全开发项目管理
知识体
知识域
知识子域

知识域：软件安全开发概述

- 知识子域：软件安全开发必要性
  - 了解软件安全问题及其原因
  - 了解传统软件开发的局限性和软件安全开发必要性

软件安全重要性 – 软件危机

- 第一次“软件危机”– 20世纪60年代
  - 根源：汇编语言不能处理日益庞大和复杂的程序
  - 解决：高级语言的诞生– FORTRAN和C
- 第二次“软件危机”– 20世纪80年代
  - 根源：大型程序：数百万行，数百人同时开发
  - 解决
    - 面向对象语言–C++/java/c#
    - 软件工程
- 第三次“软件危机”– 21世纪头十年
  - 根源：软件安全？

软件安全问题广泛存在

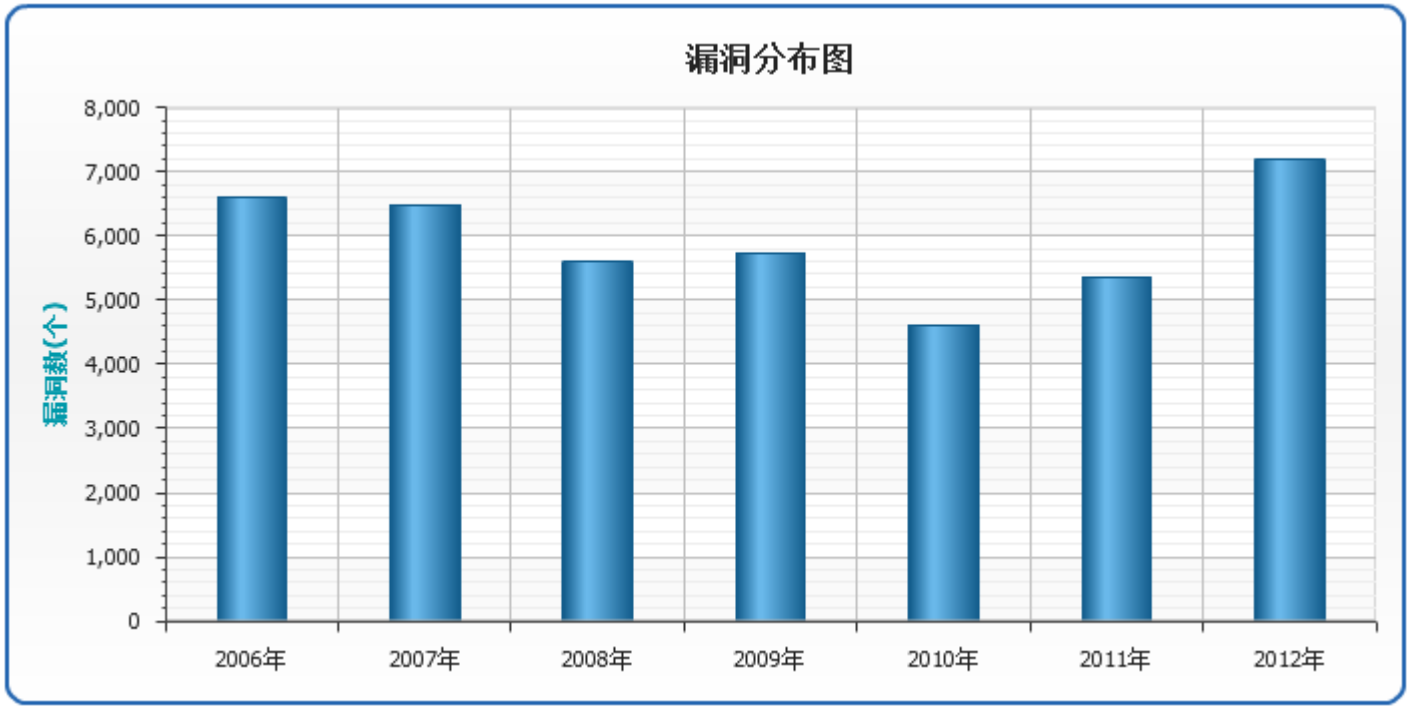
- 软件应用广泛
  - 电脑游戏、火车票售票系统、多媒体教学
  - 手机、航天飞机、人造卫星
  - …
- 软件安全问题广泛存在
  - 运行错误
  - 售票系统反应慢、连不上、崩溃
  - 多媒体教学系统死机
  - 黑客盗取泄漏的银行密码
  - …
- 安全问题日益增加…

软件安全问题产生后果

- 软件安全问题的后果
  - 造成产品运行不稳定，得不到正确结果甚至崩溃
    - 可靠性、可用性
  - 被恶意攻击，导致信息泄漏/数据破坏等后果
    - 保密性、完整性
- 一些因软件安全问题导致的严重后果
  - 售票系统瘫痪
  - 美国放射治疗仪超剂量辐射事件
  - 阿丽亚纳5号火箭首发失败事件
  - Stuxnet病毒攻击伊朗布什尔核电站

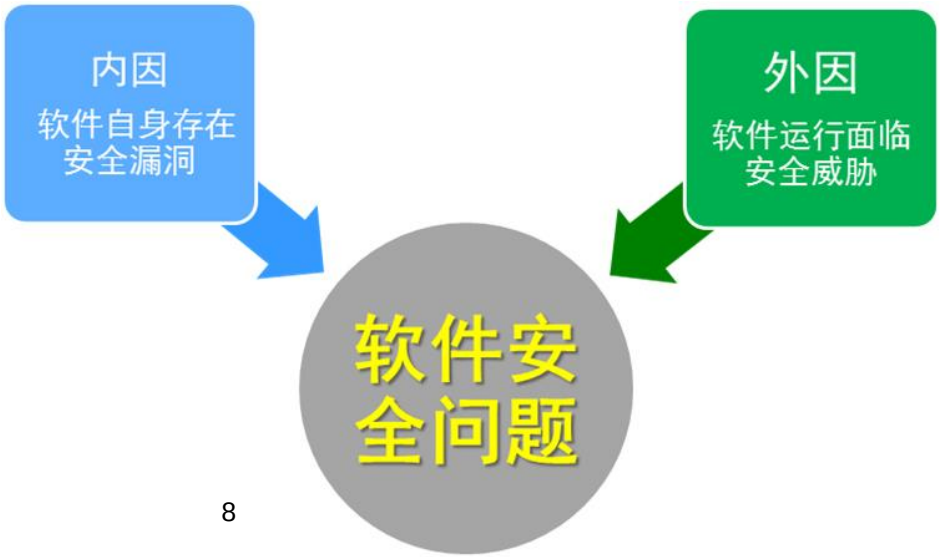
漏洞情况统计

- 中国国家漏洞库最近七年漏洞数量情况统计

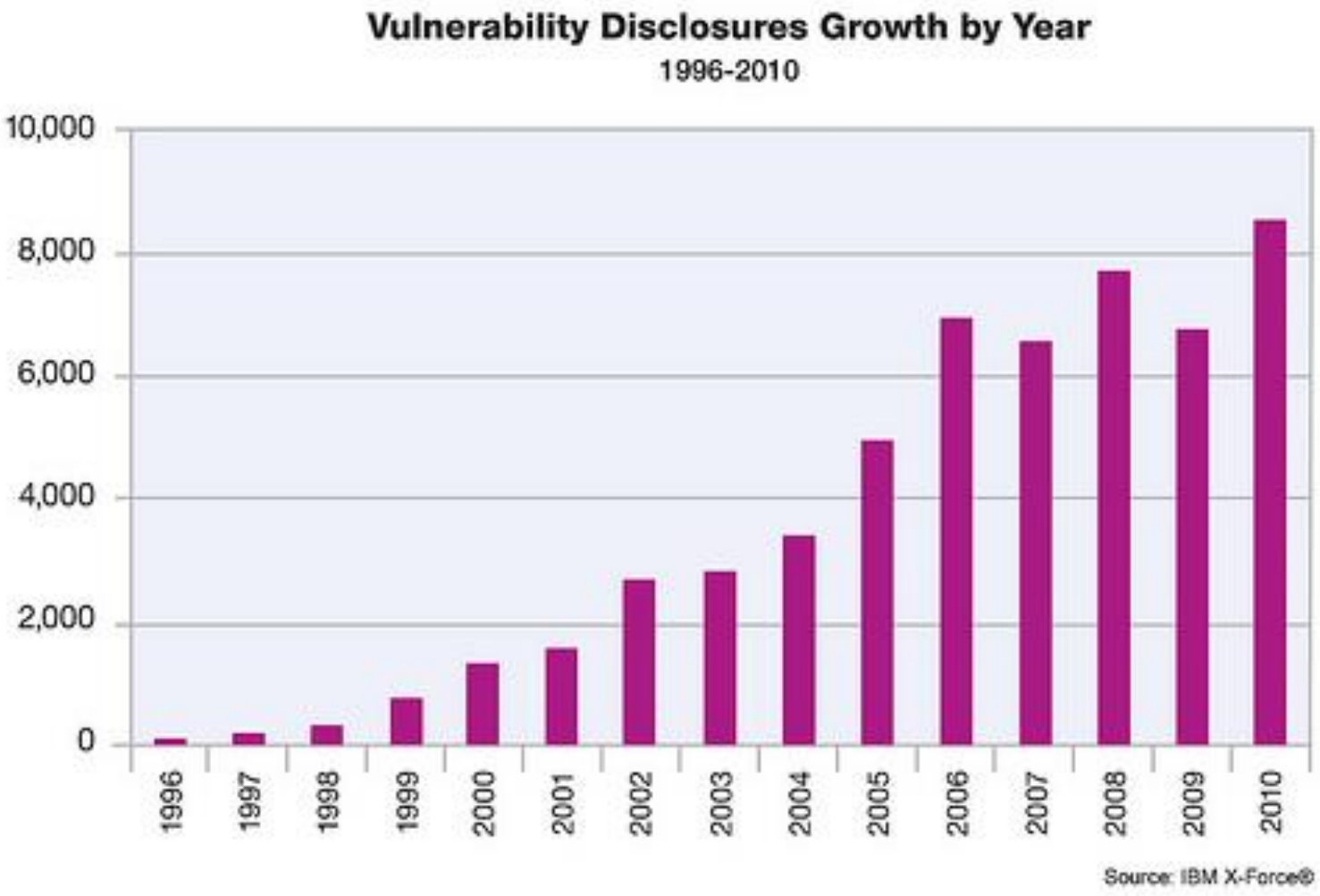


软件安全问题原因

- 存在诸多安全问题的原因
  - 软件开发周期短，工作量大，无暇顾及安全
  - 软件设计时缺乏安全设计
  - 软件开发人员缺乏安全编程经验
  - 功能越来越多，软件越来越复杂
  - 软件模块复用，可扩展性/灵活性要求高
  - 互联网环境下的安全挑战
- 根本原因
  - 存在漏洞
  - 存在威胁

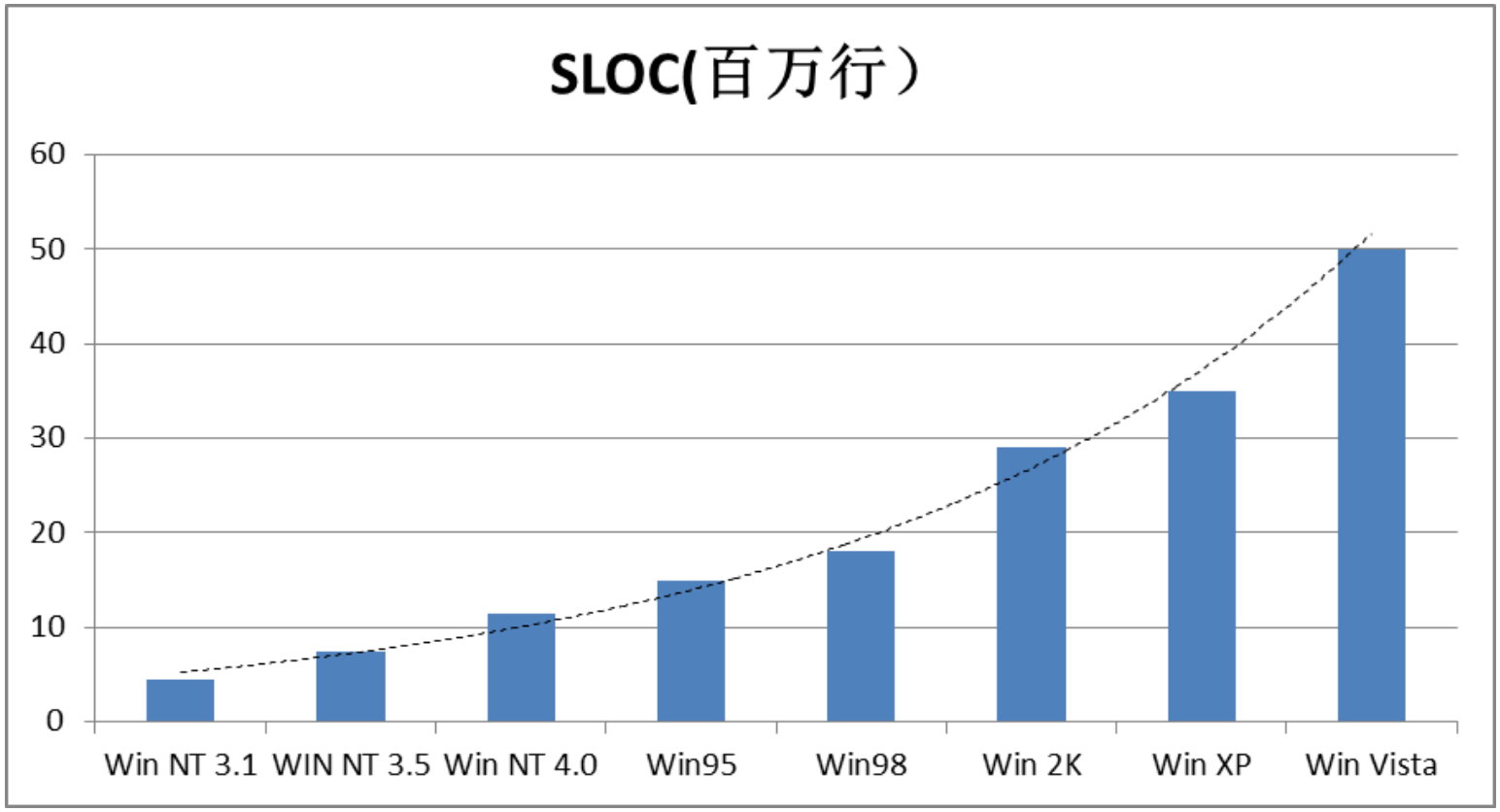


软件漏洞逐渐增加


Vulnerability Disclosures Growth by Year
1996-2010
10,000
8,000
6,000
4,000
2,000
0
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
Source: IBM X-Force®

软件越来越复杂

- **Windows 系列软件源代码行数**



漏洞与软件安全

- 漏洞已经成为危害软件安全的主要因素
  - 危及用户对软件的信任、业务运营，还会危及一系列关键基础设施和应用
- 漏洞普遍存在
  - 普通软件工程师，每千行代码（KLOC）
    - 存在20个缺陷
  - 由于采用严格的软件开发质量管理机制和多重测试环节，软件公司的缺陷率（每千行代码）
    - 普通软件开发公司的缺陷密度为4～40个缺陷
    - 高水平的软件公司的缺陷密度为2～4个缺陷
    - 美国NASA的软件缺陷密度可达到0.1个缺陷

美国重视安全开发和源代码安全

- 2007年美国空军成立了“Application Software Assurance Center of Excellence”开始对所用应用软件进行源代码缺陷检测
- 2008年加州大选软件因为没有通过源代码安全审查而被取消
- 2008年美国 FDA 器械和辐射健康中心开始使用源代码缺陷检测工具对发生问题和事故的医疗设备进行检测
- 2008年美国电力联盟也启动系统安全检测评估

需要安全的软件

- 安全的软件
  - 不存在安全漏洞
  - 能抵御各种攻击威胁
  - 按照预期的方式执行：do what is intended

- 保证程序可以正常执行任务
  - 通过在软件开发生命周期各阶段采取必要的、相适应的安全措施来避免绝大多数的安全漏洞。
  - 采取措施只能有效减少，但并不能完全杜绝所有的安全漏洞。

软件安全保障

软件安全保障的概念

- 软件安全保障是对“软件可以规避安全漏洞而按照预期的方式执行其功能”的信心 。这些安全漏洞或者故意设计在软件之中，或者在其生命周期被偶然插入到软件中。

软件安全保障目标

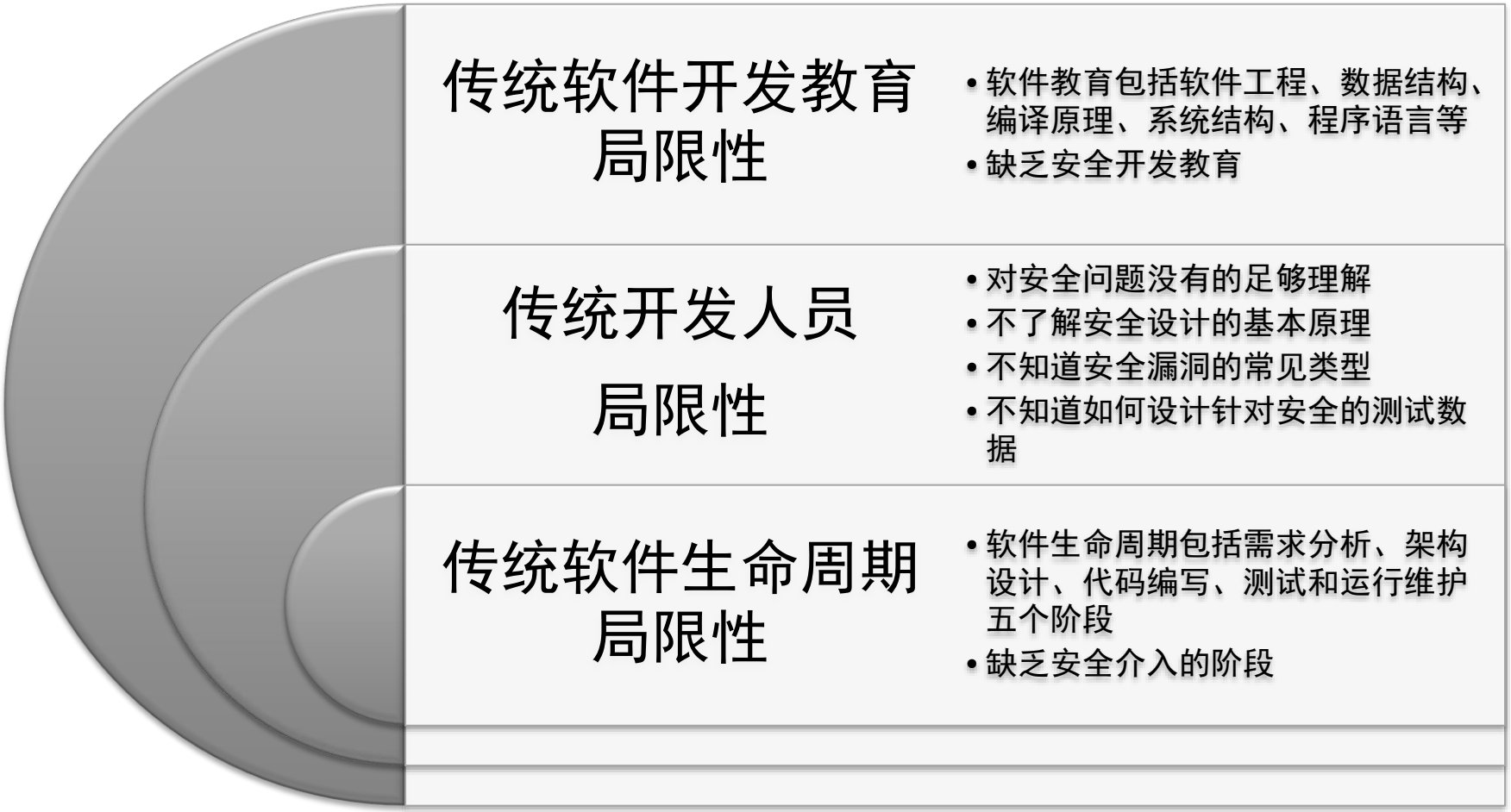
- 软件安全保障目标是在软件开发生命周期中提升软件的安全性，主要目的是
  - **可信赖性**：无论是恶意而为还是无意疏忽，软件都没有可利用的漏洞存在
  - **可预见性**：对软件执行时其功能符合开发者的意图的信心。
  - **遵循性**：将（软件开发）跨学科的活动计划并系统化，以确保软件过程和软件产品满足需求、遵循相关标准。

– DHS，2006

软件安全保障与风险管理

- 在软件安全保障中，需要贯彻风险管理的思想
  - “安全就是风险管理”
- 软件安全是以风险管理为基础
  - 安全不必是完美无缺的，但风险必须是能够管理的
- 最适宜的软件安全策略就是最优的风险管理对策
  - 这是一个在有限资源前提下的最优选择问题
  - 防范不足会造成直接的损失；防范过多又会造成间接的损失

传统的软件开发局限性



- 需要安全的软件开发！

知识域：软件安全开发概述

知识子域：软件安全开发简介

- 理解安全开发有关概念，包括软件安全、安全软件开发生命周期等
- 了解安全软件开发生命周期有关模型研究及应用情况，包括微软SDL、BSI系列模型、CLASP和SAMM等

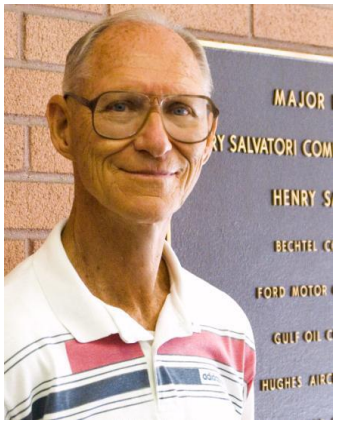
软件安全开发基本概念

- 软件安全开发
  - 采取措施防止由于设计、开发、提交、升级或维护中的缺陷而导致的系统脆弱性
  - 20世纪末/21世纪初开始展开研究
- 安全软件开发生命周期
  - 安全软件开发涵盖了软件开发整个生命周期
  - Secure Software Development Lifecycle
  - 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

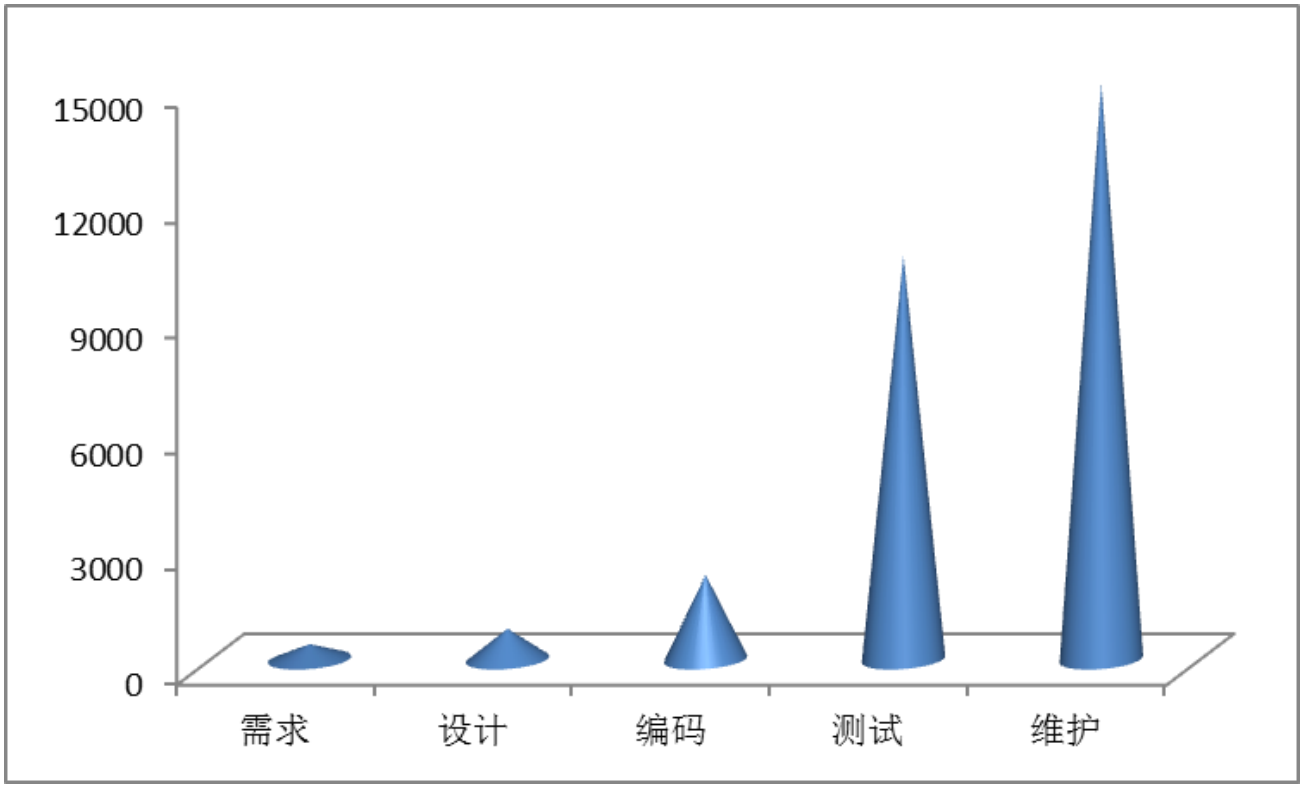
安全软件开发生命周期

- 将安全融入
  - 在设计/开发/测试等过程中融入安全
  - 在传统的过程中增加安全过程
- 安全提前介入
  - NIST：在软件发布以后进行修复的代价是在软件设计和编码阶段即进行修复所花代价的30倍
  - 软件系统发布以后才进行漏洞修复代价是最高的，且常常伴随着软件系统使用者的极大损失
- 实施软件安全开发
  - 规范指南
  - 最佳实践

不同阶段修复漏洞的代价



Barry Boehm



相关模型和研究

- 安全软件开发生命周期
  - 安全设计原则
  - 安全开发方法
  - 最佳实践
  - 安全专家经验
- 多种模型被提出和研究
  - 可信计算安全开发生命周期（微软）
  - BSI系列模型（Gary McGraw等）
  - SAMM（OWASP）
  - CLASP（OWASP）

可信计算安全开发生命周期

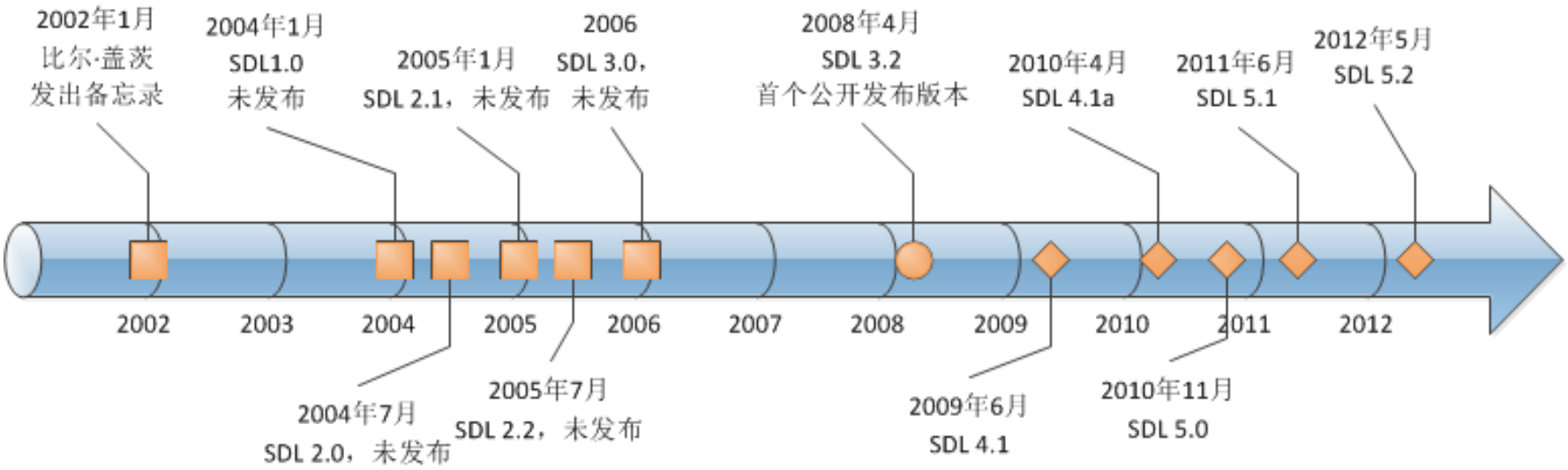
- 微软，2002.1，盖茨
- 安全开发生命周期
  - SDL（The Trustworthy Computing Security Development Lifecycle）
  - 强调开发安全的软件对于微软未来的重要性，并为此计划花费了3亿美元和2000多个工作日
  - 自 2004 起，SDL 作为全公司的计划和强制政策，在将安全和隐私植入软件和企业文化方面发挥了重要作用。
  - 通过将整体和实践方法相结合，SDL 致力于减少软件中漏洞的数量和严重性。SDL 在开发过程的所有阶段中均引入了安全和隐私。

SDL使用

- SDL 是一个安全保证过程，其在开发过程的所有阶段中引入了安全和隐私原则。
- Microsoft 将 SDL 与软件行业和客户开发组织自由共享，供他们用来开发更为安全的软件。
- 如何使用SDL？
  - 为了实现所需安全和隐私目标，项目团队或安全顾问可以自行决定添加可选的安全活动
  - 开发团队应以SDL指南为指导，实施SDL的时候结合考虑组织的时间、资源和业务运营方式
  - Cisco、EMC、Symantec等安全公司均借鉴微软SDL中的做法——CSDL

SDL发展历史

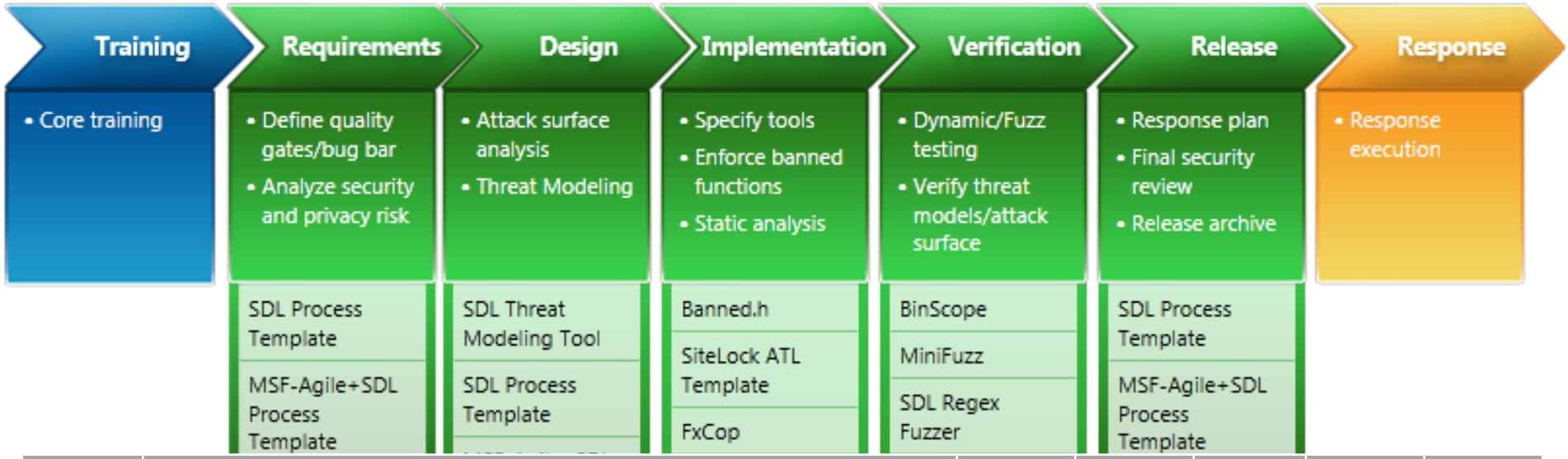
- 2002. 1～今
  - 20 世纪 90 年代中期到后期（Melissa）和 21 世纪初期（Code Red、Nimda、UPnP 等）出现了一系列影响重大的恶意软件事件，促使微软重新考虑开发人员安全过程和策略



SDL的阶段和安全活动

- 软件安全开发生命阶段
  - 5+2个阶段
  - 16项必需的安全活动

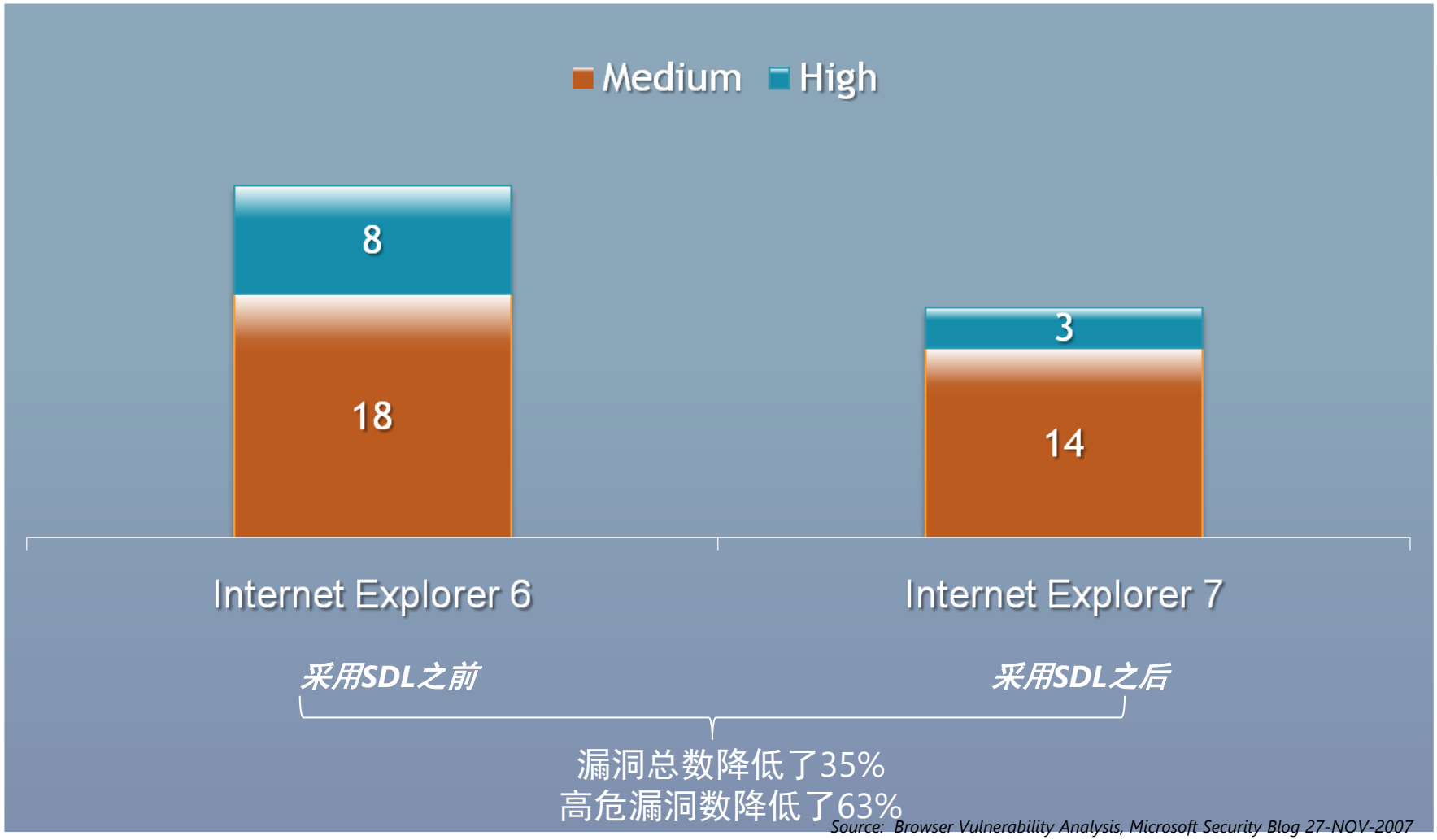
SDL每个阶段用到的工具



序号	工具	需求	设计	实现	验证	发布
1	SDL过程模板和MSF-Aglie+SDL过程模板	√	√	√	√	√
2	SDL威胁建模工具		√			
3	Banned.h、SiteLock ATL模板、FxCop、C/C++源代码分析工具、Anti-XSS库、32位的CAT.NET、64位的CAT.NET			√		
4	BinScope、MiniFuzz、SDL Regex Fuzzer、AppVerifier				√	

IE采用SDL后的效果

正式发布后12个月内修复的漏洞总数



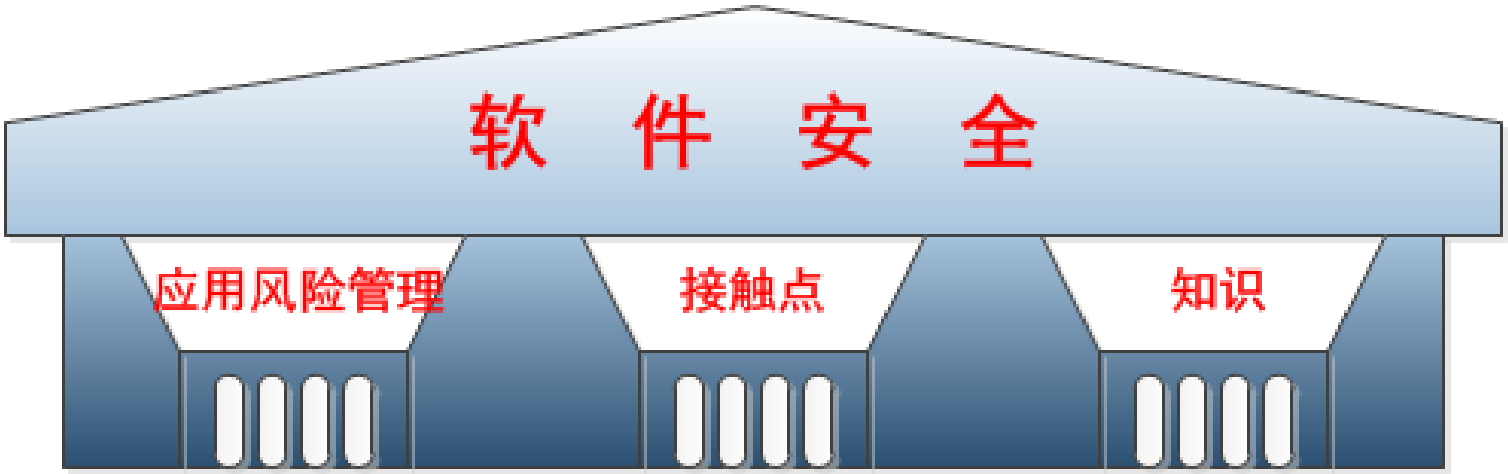
Source: Browser Vulnerability Analysis, Microsoft Security Blog 27-NOV-2007

BSI系列模型

- BSI——使安全成为软件开发必须的部分
  - Building Security IN, BSI
  - Gray McGraw, Cigital公司
- 在整个软件开发生命周期中要确保将安全作为软件的一个有机组成部分。
  - 无须改变你的软件开发方法
  - 适用各种软件开发生命周期
- 合作
  - NIST
  - 美国国土安全部
  - 大学（加州大学戴维斯分校、普林斯顿、莱斯…）

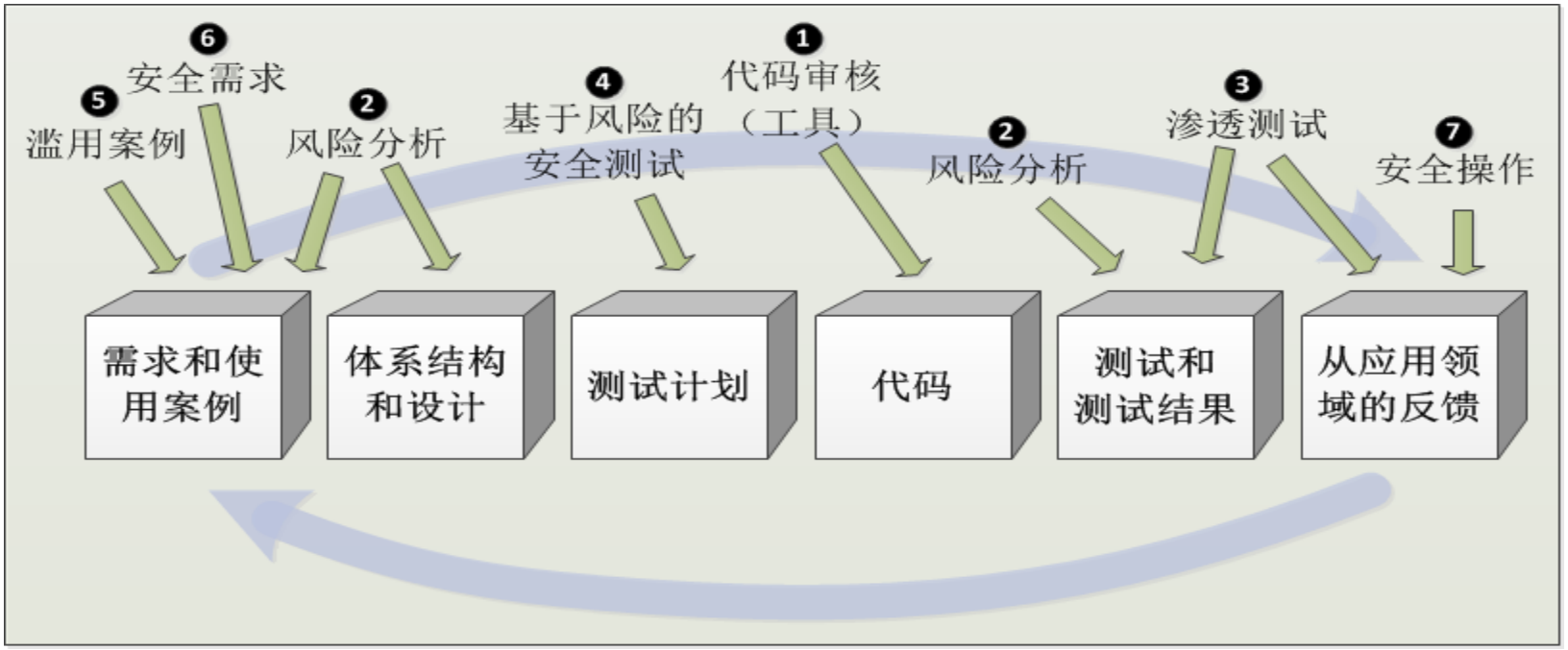
软件安全的三根支柱

- 三根支柱
  - 应用风险管理
  - 软件安全的接触点
  - 知识



接触点模型

- 接触点，即在软件开发生命周期中保障软件安全
  - 一套最优方法、一种战术性方法
  - 在每一个开发阶段上尽可能地避免和消除漏洞
  - “黑帽子”和“白帽子”



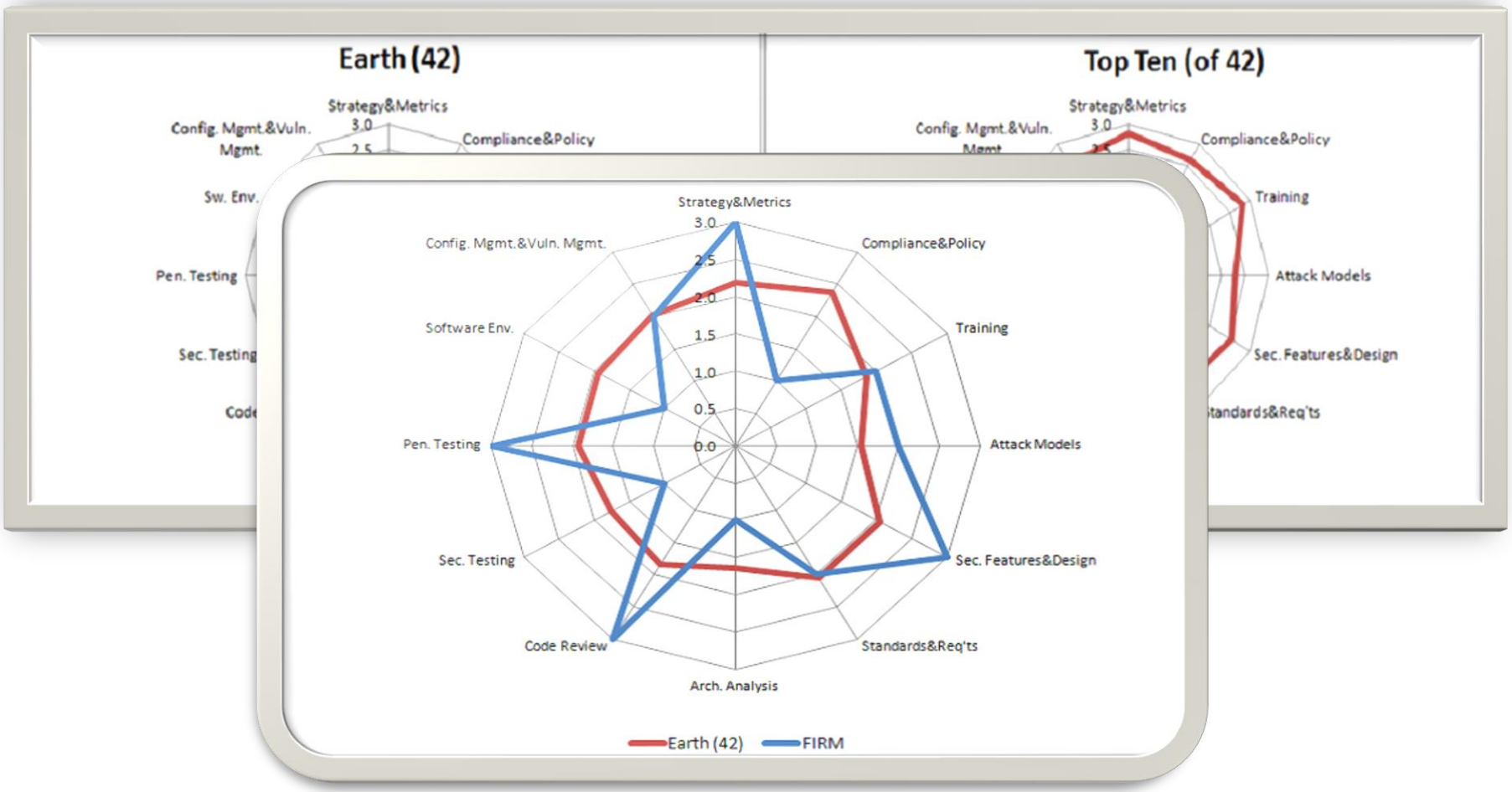
- SSF（Software Security Framework）
  - 软件安全框架

监管	信息/情报	SSDL接触点	部署
策略和度量	攻击模式	架构分析	渗透测试
履约和策略	安全特征和设计	代码审计	软件环境
培训	标准和需求	安全测试	配置管理和漏洞管理

BSIMM

- BSI成熟度模型
  - Building Security In Maturity Mode
  - Gary McGraw、Brian Chess和Sammy Migues
  - 使用SSF对所有项目进行描述
  - 了解别人的安全项目过程，指导自己的安全项目
- 目标
  - 是对真实的软件安全项目所开展的活动进行量化
  - 构建和不断发展软件安全行动的指南
- BSIMM 3.0
  - 2011年
  - 42个公司（Microsoft、Intel、Google、…）

BSIMM结果图

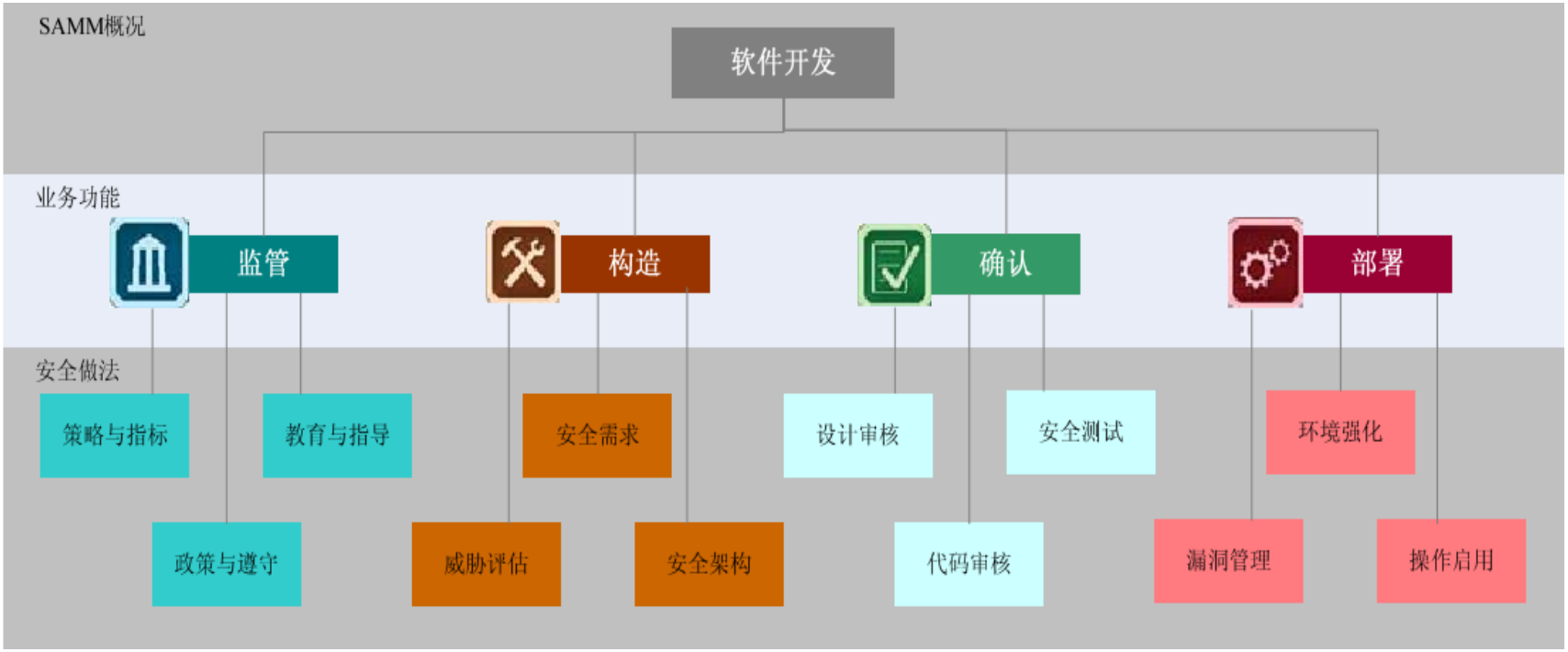

Earth (42)
Top Ten (of 42)
Strategy&Metrics
Compliance&Policy
Training
Attack Models
Sec. Features&Design
Standards&Req'ts
Arch. Analysis
Code Review
Sec. Testing
Pen. Testing
Software Env.
Config. Mgmt.&Vuln. Mgmt.
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Earth (42)
FIRM

SAMM

- OWASP SAMM
  - Software Assurance Maturity Mode
  - 软件保证成熟度模型
  - OWASP（开放Web应用安全项目）
- 一个开放的框架，用以帮助制定并实施针对软件安全特定风险的策略
  - 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。

- 规定了四个软件开发过程中的核心业务功能
  - 治理：组织管理其软件开发的过程和活动
  - 构造：组织在开发项目中确定目标并开发软件的过程与活动
  - 验证：组织测试和验证软件的过程与活动
  - 部署：组织软件发布的相关管理过程与活动
- 4个成熟度级别
  - 0-3级

OWASP SAMM


SAMM概况
软件开发
业务功能
监管
构造
确认
部署
安全做法
策略与指标
教育与指导
安全需求
设计审核
安全测试
环境强化
政策与遵守
威胁评估
安全架构
代码审核
漏洞管理
操作启用

CLASP

- 综合的轻量应用安全过程（Comprehensive, Lightweight Application Security Process (CLASP) ）
- 选取了30个特定的基于角色的活动(activities)，用于提升整个开发团队的安全意识，并针对这些活动给出了相应的指南、导则和检查列表

CLASP介绍

- 基于角色
  - 项目经理 、需求分析师 、软件架构师 、设计者 、实施人员 、集成和编译人员 、测试者和测试分析师 、安全审计员
- 对于每个活动，CLASP描述了以下内容
  - 安全活动应该在什么时间、应该如何实施
  - 如果不进行这项安全活动，将会带来的多大的风险
  - 如果实施这项安全活动，估计需要多少成本

各模型比较

SDL	文档丰富，维护更新及时	较多工具支持	适合大型企业
BSI接触点	强调开发安全重点	注重实用方法	上手容易
BSIMM	最佳实践参考	他山之玉	不强制实践
CLASP	轻量级过程；	以角色及其职责为核心	适合小型企业
SAMM	开放框架	安全知识要求较低	和BSIMM的安全活动能对应

知识域：软件安全开发的关键阶段

- 知识子域：软件安全设计
  - 了解软件安全设计的重要性
  - 理解软件安全设计基本原则
  - 理解受攻击面概念和常用减少受攻击面的保护措施
  - 了解威胁建模的概念和目的
  - 理解威胁建模的关键因素及作用

安全设计的重要性

- 安全编码？安全测试？
  - 传统方法：软件发布后测试、等待修复Bug
  - Gary McGraw ：50%的安全问题由设计瑕疵引起
  - 安全提前介入，效益高，成本低

- 设计缺陷——举例
  - Microsoft Bob
  - 明文存储口令，甚至将口令拿到客户端对比验证

安全设计目标

- 安全设计工作目标：
  - 制定项目计划来定义安全行为
  - 制定安全检查点来保证安全控制措施的质量
  - 识别配置过程和变更控制过程
- 主要设计内容
  - 确定访问控制机制，定义主体角色和权限
  - 选择加密方法和算法
  - 解决敏感数据处理问题
  - 评估内部通信机制
  - 确定完整性机制
  - …

安全设计原则

- 保护最薄弱的环节
- 纵深防御
- 最小特权
- 最小共享
- 权限分离

安全设计原则

- 经济性原则
- 保护隐私
- 正确理解“秘密”
- 安全的错误处理
- 心理接受能力

受攻击面

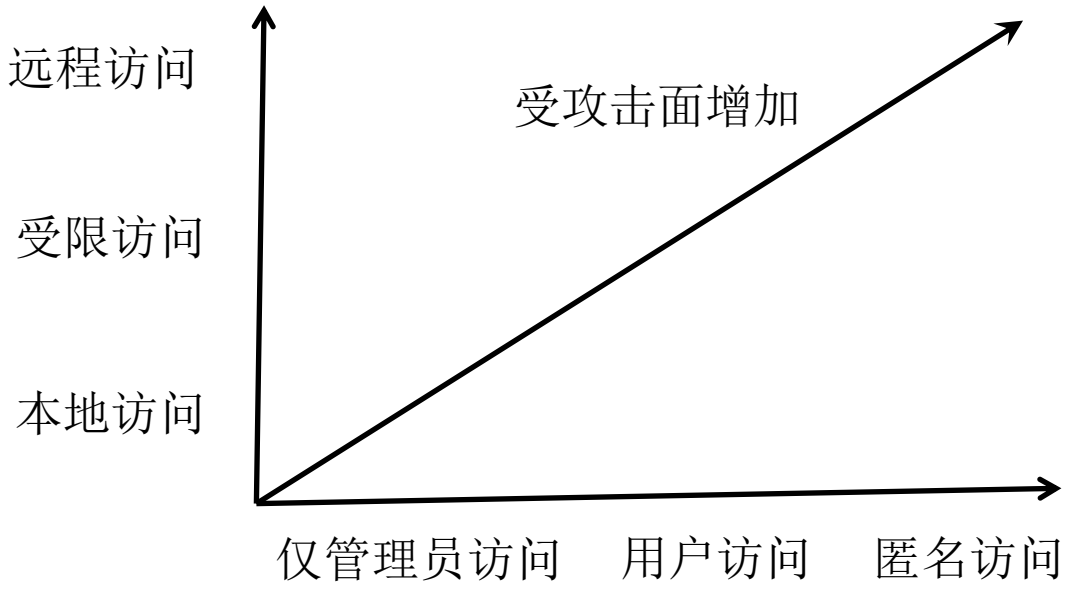
- 受攻击面
  - 对一个软件系统可以采取的攻击方法集合
  - Attack Surface Reduction，ASR
  - 功能、API、接口、资源、数据存储等
- 一个软件的攻击面越大安全风险就越大

降低受攻击面

对于提高软件安全性 至关重要！

降低受攻击面的方法

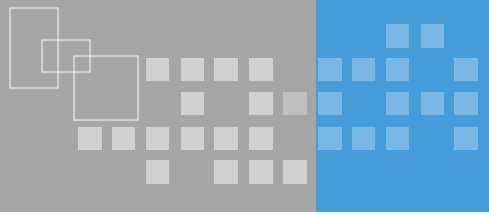
- 第一步：分析产品功能的重要性
  - 是否必须
- 第二步：分析从哪里访问这些功能
- 第三步：采取合理措施（降低特权）



较少软件受攻击面

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

微软产品ASR举例



Windows XP SP2

- 所有RPC都必须认证
- 防火墙默认开启

IIS6

- 默认关闭
- 仅以Network service权限运行
- 默认仅支持静态文件

SQL Server 2005

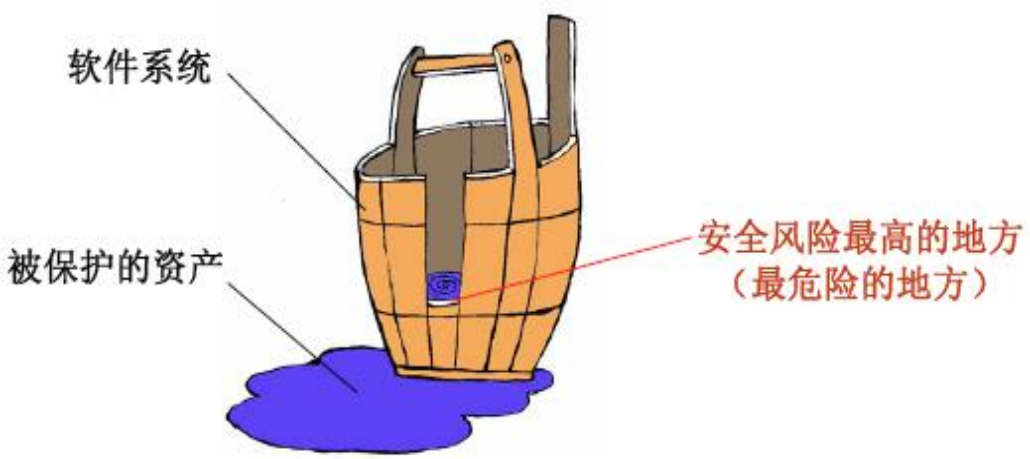
- 默认关闭xp_cmdshell
- 默认关闭CLR和COM
- 网络服务

Visual Studio® 2005

- Web服务器默认仅运行本地连接
- SQL Server Express 默认仅允许本地连接

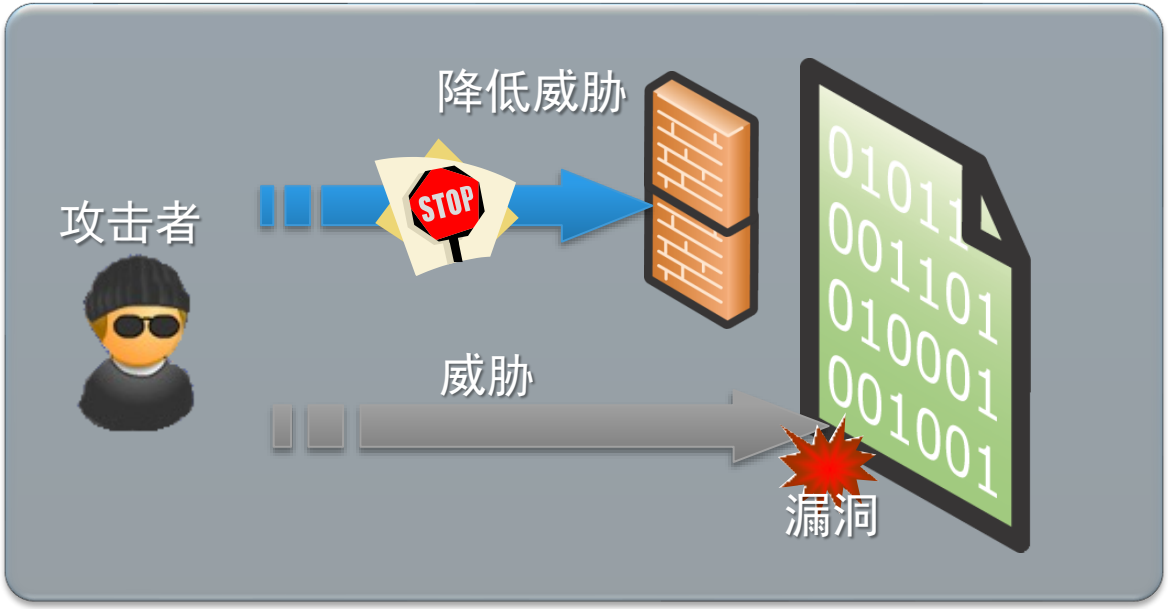
威胁建模

- 威胁建模是以结构化的方式，识别、评估应用系统面临的威胁
- 目的
  - 帮助在设计阶段充分了解各种安全威胁，并指导选择适当的应对措施
  - 对可能的风险进行管理
  - 可以重新验证其务
  - 有助于软件的受



威胁建模流程

- 确定建模对象
- 识别威胁
- 评估威胁
- 消减威胁



威胁建模流程

确定建模对象

- 应用的可信任边界之内的所有功能组件
- 边界之外的应用最实际部分

识别威胁

- 发现组件或进程存在的威胁
- 威胁不等于漏洞

评估威胁

- 判断攻击发生概率
- 攻击后果
- 计算风险

消减威胁

- 重新设计并排除这个威胁
- 使用标准的威胁消减技术
- 发明新的消减方法
- 根据安全Bug标准来确定是否可接受风险
- 把威胁作为漏洞记录下来，以后再想办法消减

STRIDE建模方法

- STRIDE建模
  - 微软提出
  - 发现或纠正设计级（design-level)的安全问题
  - 是微软SDL的一部分

S	Spoolfing Identity	假冒身份/欺骗标识
T	Tampering with data	篡改数据
R	Repudiation	抵赖
I	Information Disclosure	信息泄漏
D	Denial of Service	拒绝服务
E	Elevation of Privilege	权限提升

理解STRIDE威胁

威胁	安全属性	定义	举例
Spoofing（哄骗）	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。

消减威胁举例

消减威胁

威胁类型	消减机制举例	消减技术举例
假冒	认证	认证方式： Cookie认证、Kerberos认证、PKI
篡改	完整性	哈希函数、消息认证码、数字签名、防篡改协议
抵赖	非抵赖性服务	强认证、安全审计、数字签名、时间戳
信息泄露	保密性	加密、保护秘密、访问控制、不保存秘密、隐私保护协议
拒绝服务	可用性	认证、访问控制、过滤、流量控制、授权
特权提升	授权	访问控制列表、最小权限运行

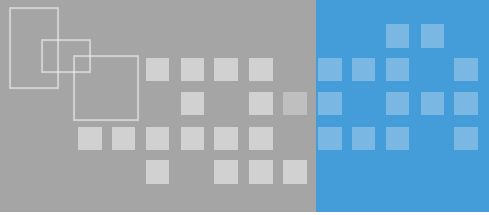
知识域：软件安全开发的关键阶段

- 知识子域：软件安全编码
  - 了解通用安全编程准则，包括验证输入、避免缓存溢出、程序内部安全、安全调用组件、程序编写编译等
  - 理解编码时禁止使用的函数
  - 了解相关的安全编码标准和建议
  - 常见代码安全问题及处置办法举例
  - 理解代码审查的目的
  - 了解常见源代码静态分析工具

通用安全编程准则

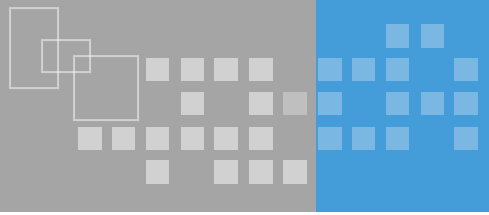
- 验证输入
- 避免缓存溢出
- 程序内部安全
- 安全调用组件
- 程序编写和编译

验证输入



- 安全程序第一道防线是检查每一个不可信的输入
  - DirectXMIDI 库，难以检查所有的输入，可能导致IE被恶意利用
- 检查、验证或者过滤输入
  - 不让恶意数据进入程序后续处理
  - 类似网络中部署防火墙
- 何处检查
  - 最初接收数据时
  - （第一次）使用数据时

验证输入——常见输入源



- 命令行
  - 参数数量、数据格式、内容
- 环境变量
  - 环境变量可能超出期望
  - 有的环境变量存储格式存在危险
- 文件
  - 不信任可以被不可信用户控制的文件内容
  - 不信任临时文件
- 网络
  - 来自网络的数据是“高度不可信的”
- 其他来源

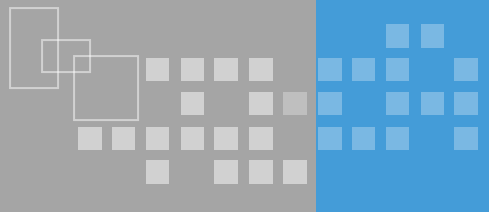
验证输入——常见数据类型

- 字符串
  - 确定合法范围，拒绝非法字符（串）
  - 0（NIL）
  - 行结束编码
    - 0x0a（unix）
    - 0x0d 0x0a（dos, windows）
    - 0x0d（APPLE MacOS）
    - 0x85（IBM OS/390）
  - 特定字符
    - 系统字符
    - 分割字符

验证输入——常见数据类型

- 数字
  - 数字检查（正则表达式/ASCII值）
  - 负数检查（大数溢出为负数）
    - Sendamil 攻击
  - 合法范围检查
- 文件名
  - 最好不让用户设置文件名
  - 避免特殊字符
    - . / - -rf ../ com1

验证输入——常见数据类型



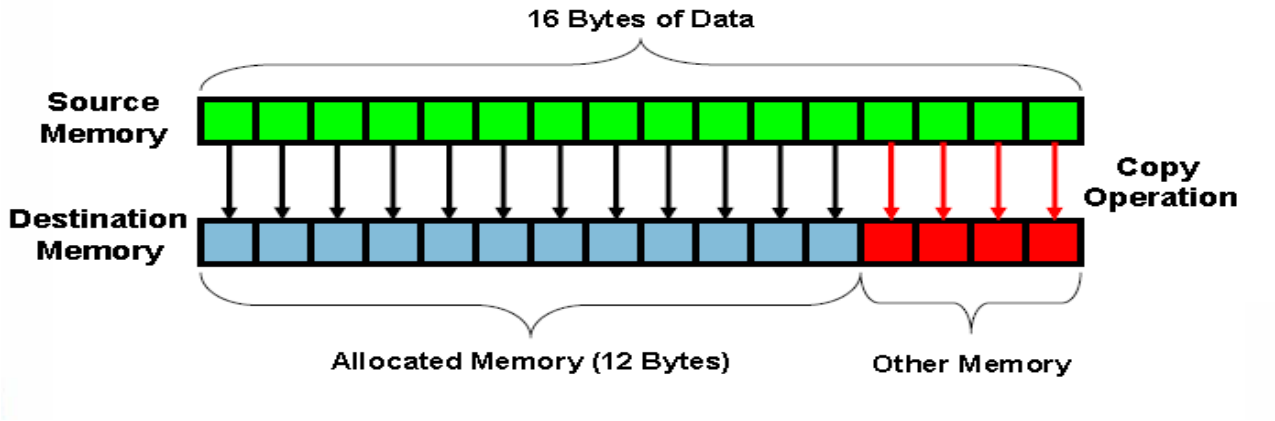
- 电子邮件地址
  - 限制合法的电子邮件地址
- UTF-8
  - 变长编码
- URI/URL
  - 非法地址
  - 在合法地址后面增加恶意内容

通用安全编程准则

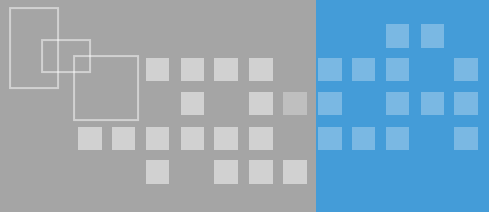
- 验证输入
- 避免缓存溢出
- 程序内部安全
- 安全调用组件
- 程序编写和编译

避免缓冲区溢出

- 缓冲区溢出
  - 缓冲区：包含相同数据类型的实例的一个连续计算机内存块
  - 溢出：数据被添加到分配给该缓冲区的内存块之外
- 外部数据比目标空间大
- 是一个非常普遍而且严重的问题



避免缓冲区溢出

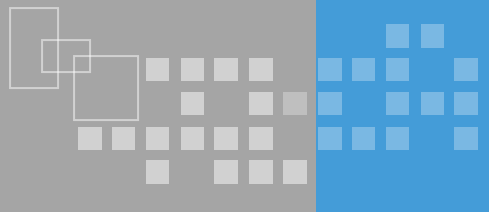


- 溢出后果
  - 攻击者可以使远程服务程序或者本地程序崩溃
  - 攻击者可以设计溢出后执行的代码
- C/C++语言
  - 语言特性决定
  - 大量的库函数存在溢出
    - strcpy、strcat、gets等
- 其他语言
  - 调用C语言库
  - C#允许设置“不安全”例程

避免缓冲区溢出——解决办法

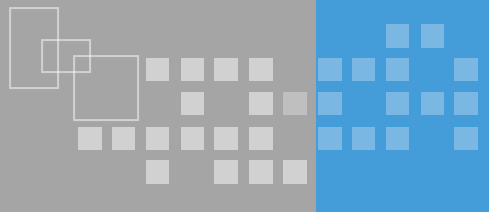
- 解决办法
  - 填充数据时计算边界
    - 动态分配内存
    - 控制输入
  - 使用没有缓冲区溢出问题的函数
    - strncpy、strncat、C++中std:string
  - 使用替代库
    - Libmib、libsafe
  - 基于探测方法的防御
    - StackGuard、ProPolice、/GS
    - 将一个“探测”值插入到返回地址的前面
  - 非执行的堆栈防御
    - 不可在堆栈上执行代码

通用安全编程准则



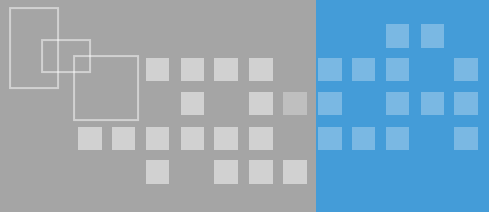
- 验证输入
- 避免缓存溢出
- 程序内部安全
- 安全调用组件
- 程序编写和编译

程序内部安全



- 程序内部接口安全
  - 程序内部接口数据的检查
  - assert
- 安全地失败
  - 检测异常，安全处理各种可能运行路径
  - 检测到某些错误行为/数据，必须以合适的方式处理，保证程序运行安全
  - 必要时立即拒绝服务，甚至不回送详细的错误代码

程序内部安全



- 最小化反馈
  - 避免给予不可靠用户过多的信息
    - 成功或失败
    - 作为跟踪检查的日志可以记录较为详细的信息
  - 认证程序在认证前尽量少给信息（版本）
  - 如果程序接受了密码，不要返回它
- 避免拒绝服务攻击
  - 输入错误尽快返回
  - 设置超时
  - 延时服务

程序内部安全

- 避免竞争条件
  - 访问共享资源时（文件/变量）没有被适当地控制
  - 使用原子操作
  - 使用锁操作——避免死锁
- 安全使用临时文件
  - 很多安全漏洞发生在访问已知文件名或可猜测的临时文件时

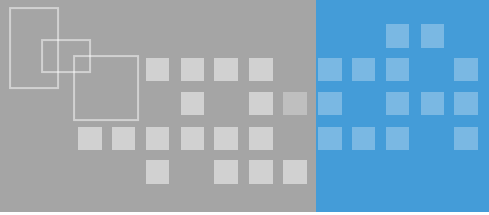
通用安全编程准则

- 验证输入
- 避免缓存溢出
- 程序内部安全
- 安全调用组件
- 程序编写和编译

安全调用其他组件

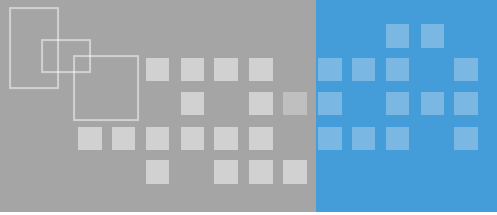
- 应用程序实际上几乎都不会是自包含的，它们通常都会调用其他组件
  - 底层的操作系统
  - 数据库
  - 可重用的库
  - 网络服务（WEB、DNS）

安全调用其他组件



- 使用安全组件，并且只采用安全的方式
  - 检查组件文档，搜索相关说明
    - gets
    - 随机数
  - 使用经过认可的组件
  - 尽可能不调用外部命令，如果不得已要调用，必须严格检查参数
    - system、open、exec、

安全调用其他组件

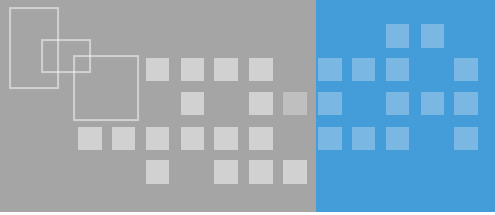


- 正确处理返回值
  - 一定要检查返回值，调用是否成功
  - 成功时检查
    - 返回值，是否按照期望值处理
    - 数据中可能含有 NUL 字符、无效字符或其他可能产生问题的东西
  - 错误时检查
    - 错误码
- 保护应用程序和组件之间传递的数据
  - 视安全需求和安全环境
    - 考虑传输加密，包括密码算法和安全协议

通用安全编程准则

- 验证输入
- 避免缓存溢出
- 程序内部安全
- 安全调用组件
- 程序编写和编译

程序编写和编译



- 使用最新版本编译器与支持工具
- 使用编译器内置防御特性
  - `gcc -Wall -Wpointer-arith -Wstrict-prototypes -O2`
- 减少潜在可被利用的编码结构或设计
- 保护秘密，及时清除密码和密钥等敏感数据
- 程序只实现你指定的功能
- 永远不要信任用户输入
- 必须考虑意外情况并进行处理
- 使用安全编码检查清单

安全编码检查清单

- 编码中禁止使用的危险函数举例

禁止使用
strcpy, wcscpy, trcpy, strcpy, _tcscpy, _ftcscpy,_mbscpy
strcat, wcscat, trcat, strcat, _tcscat, _ftcscat,_mbscat
vsprintf, vswprintf, wvsprintf, wvnsprintf, _vstprintf
sprintf, swprintf, wsprintf, wnsprintf, _stprintf
gets, _getws, _getts

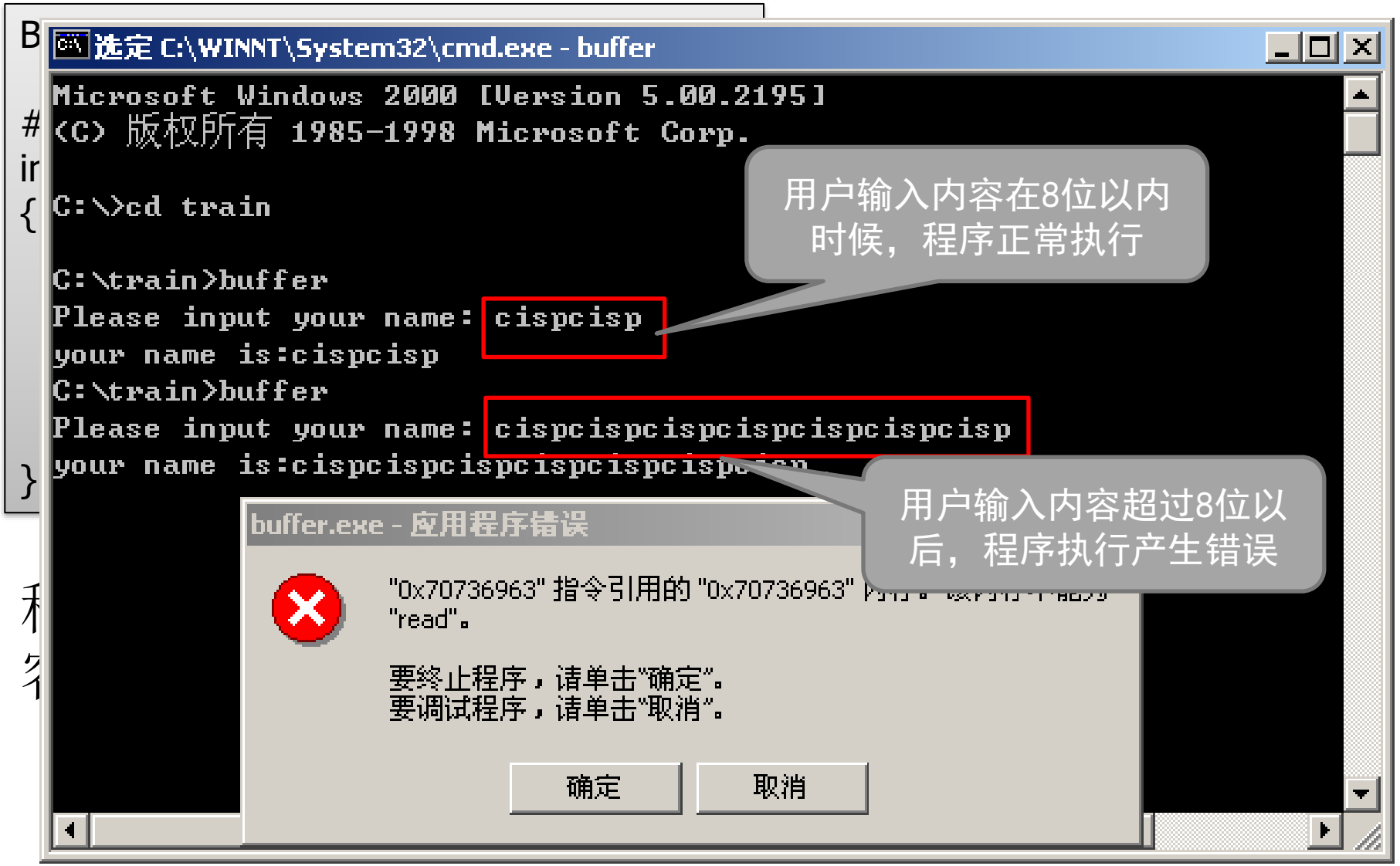
参考安全编程规范

※CERT安全编码规范　※ OWASP安全编码规范　……

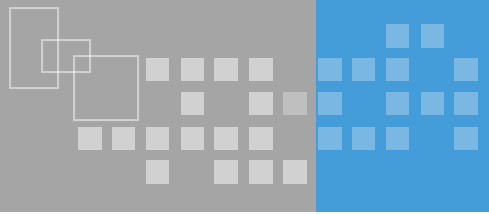
常见代码安全问题及处置办法

- 常见C/C++安全问题及处置办法举例
- 常见WEB安全问题及处置办法举例

C/C++程序——缓冲区溢出


选定 C:\WINNT\System32\cmd.exe - buffer
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-1998 Microsoft Corp.
C:\>cd train
C:\train>buffer
Please input your name: cispcisp
your name is:cispcisp
C:\train>buffer
Please input your name: cispcispcispcispcispcispcisp
用户输入内容在8位以内时候，程序正常执行
用户输入内容超过8位以后，程序执行产生错误
buffer.exe - 应用程序错误
"0x70736963" 指令引用的 "0x70736963" 内存。该内存不能为 "read"。
要终止程序，请单击"确定"。
要调试程序，请单击"取消"。
确定
取消

C/C++程序——检查输入



```
string command = "md5sum < " + filename + " > ./results";
system(command.c_str());
```

文件名修改成 x ; ;rm -fr ~

```
md5sum < x ; rm -fr ~ > ./my-output
```

WEB安全风险

- WEB应用广泛
- The Open Web Application Security Project (OWASP)发布WEB十大漏洞/风险

OWASP Top 10 – 2007 (Previous)	OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	A1 – Injection
A1 – Cross Site Scripting (XSS)	A2 – Cross-Site Scripting (XSS)
A7 – Broken Authentication and Session Management	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	A5 – Cross-Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	A6 – Security Misconfiguration (NEW)
A8 – Insecure Cryptographic Storage	A7 – Insecure Cryptographic Storage
A10 – Failure to Restrict URL Access	A8 – Failure to Restrict URL Access
A9 – Insecure Communications	A9 – Insufficient Transport Layer Protection
<not in T10 2007>	A10 – Unvalidated Redirects and Forwards (NEW)
A3 – Malicious File Execution	<dropped from T10 2010>
A6 – Information Leakage and Improper Error Handling	<dropped from T10 2010>

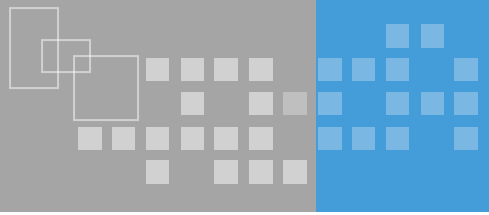
SQL注入简单示例

Select * from table where user='admin' and pwd='SdfG#345!';

Select * from table where user='admin' and pwd='123' or '1=1'

由于密码的输入方式，使得查询语句返回值永远为True，因此通过验证

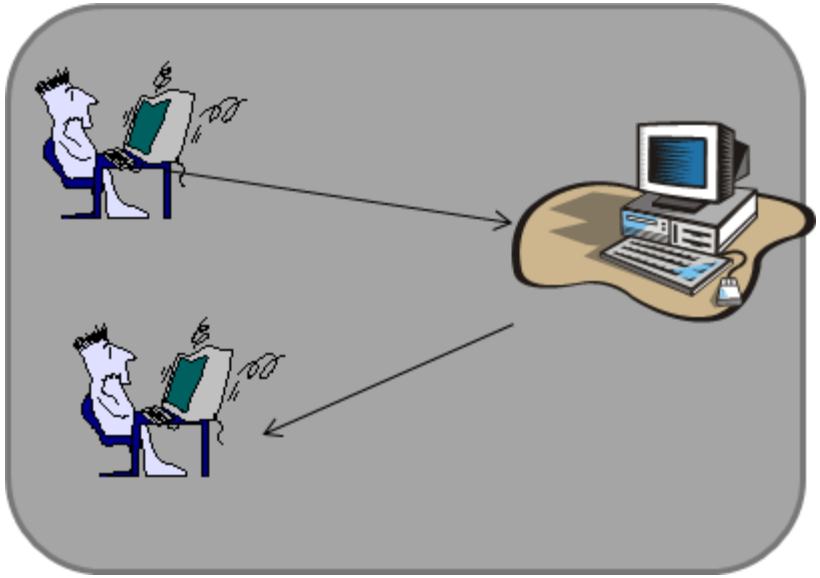
解决注入问题



- 不要相信用户的输入，一切的输入都是危险的
- 解决之道
  - 验证输入，过滤非法字符
  - 对所有用户输入进行转义
  - 参数化查询
  - 使用存储过程
  - 使用视图
  - 最小权限原则

跨站攻击

- 跨站攻击（XSS）：恶意攻击者往Web页面里插入恶意html代码，当用户浏览该页之时，嵌入其中Web里面的html代码会被执行，从而达到恶意攻击的目的。
- 攻击危害性
  - 敏感信息泄露
  - 钓鱼攻击
  - 屏蔽/伪造页面特定信息
  - Cookie欺骗
  - 拒绝服务攻击
  - …



跨站攻击防御办法

- 存在跨站漏洞的页面或模块通常未对传入的url参数进行处理，或者确定传入来源是可靠的
- 过滤特殊字符
  - http://www.XXXX.com.cn/news/hotspot.jsp?column=123
  - 该页面是专用于显示一些简短信息的页面，它接受url中的一个参数column并把它显示出来
  - hotspot页面中，应对输入的colum进行过滤

源代码审核

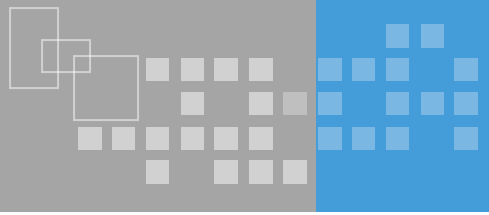
源代码审核关注编码中的实现缺陷，通常通过静态分析工具进行，它们扫描源代码，能够发现大约50%的安全问题。

- 源代码审核就是检查源代码，检测并报告源代码中的可能导致安全弱点的薄弱之处。
- 人工审核
  - 费时费力
  - 容易遗漏
- 工具审核
  - 速度快，自动
  - 可升级知识库

代码审核工具

- 商业工具
  - Coverity
  - Fortify
  - Ounce Labs
  - SecureSoftware
- 免费/开源工具
  - BOON
  - Cqual
  - Xg++
  - FindBugs

“好”的源代码分析工具

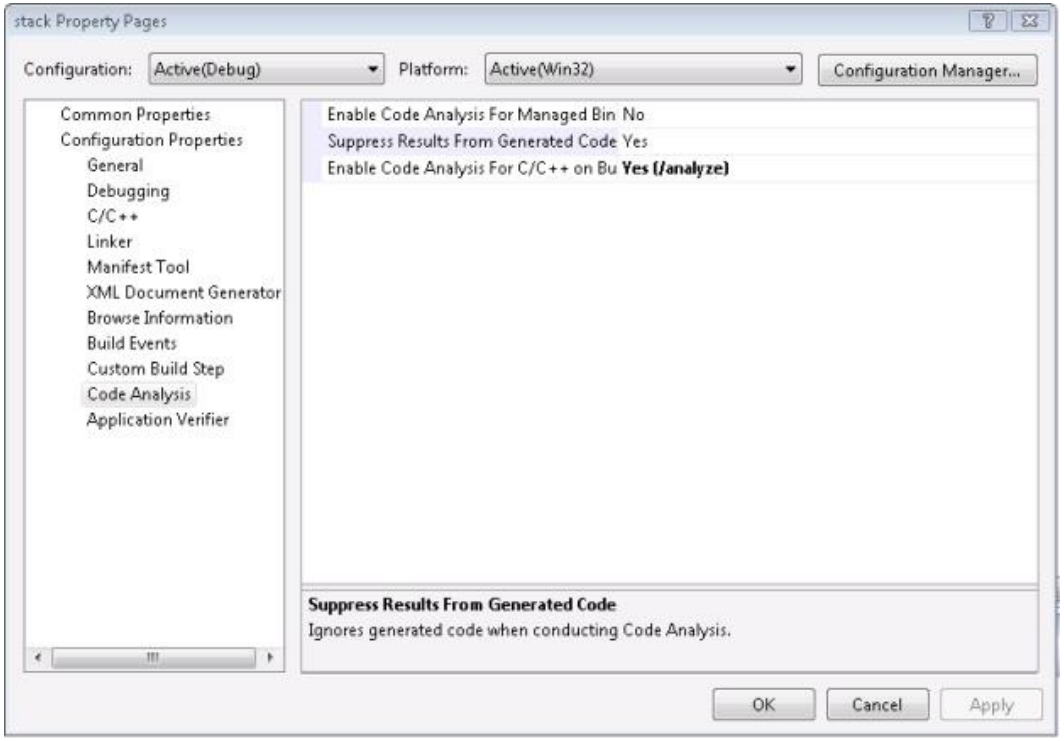


- 安全性
  - 安全审核，不要以功能为主
- 多层性
  - 软件的多层架构、多层平台、多种语言
- 可扩展性
  - 扩展规则、扩展技术
- 知识性
  - 主用于分析，开发者也能“学到” 安全编程知识
- 集成性
  - 支持与IDE集成，支持make、ant等工具

Prefast

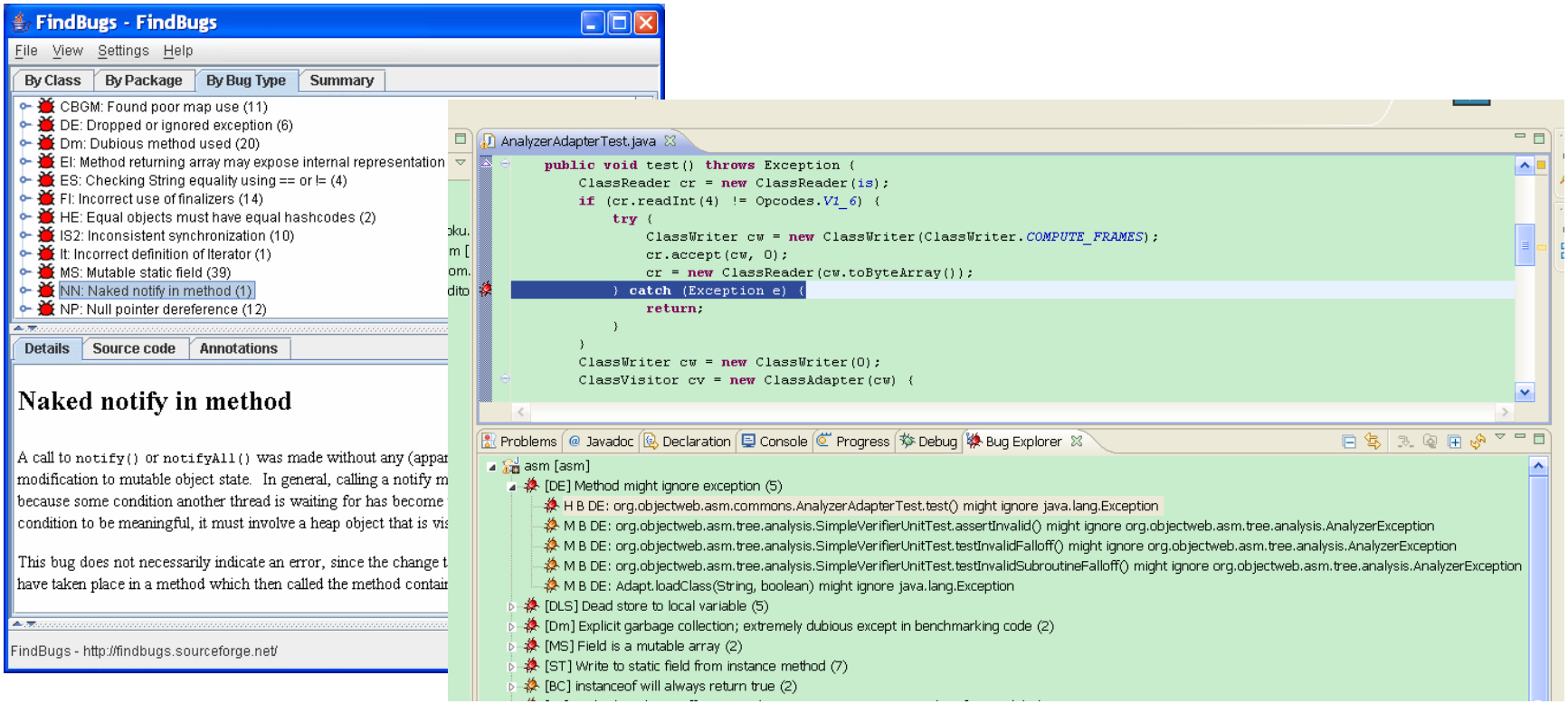
Prefast

- 微软研究院开发的静态代码分析工具
- 通过分析代码的数据和控制信息来检测程序中的缺陷



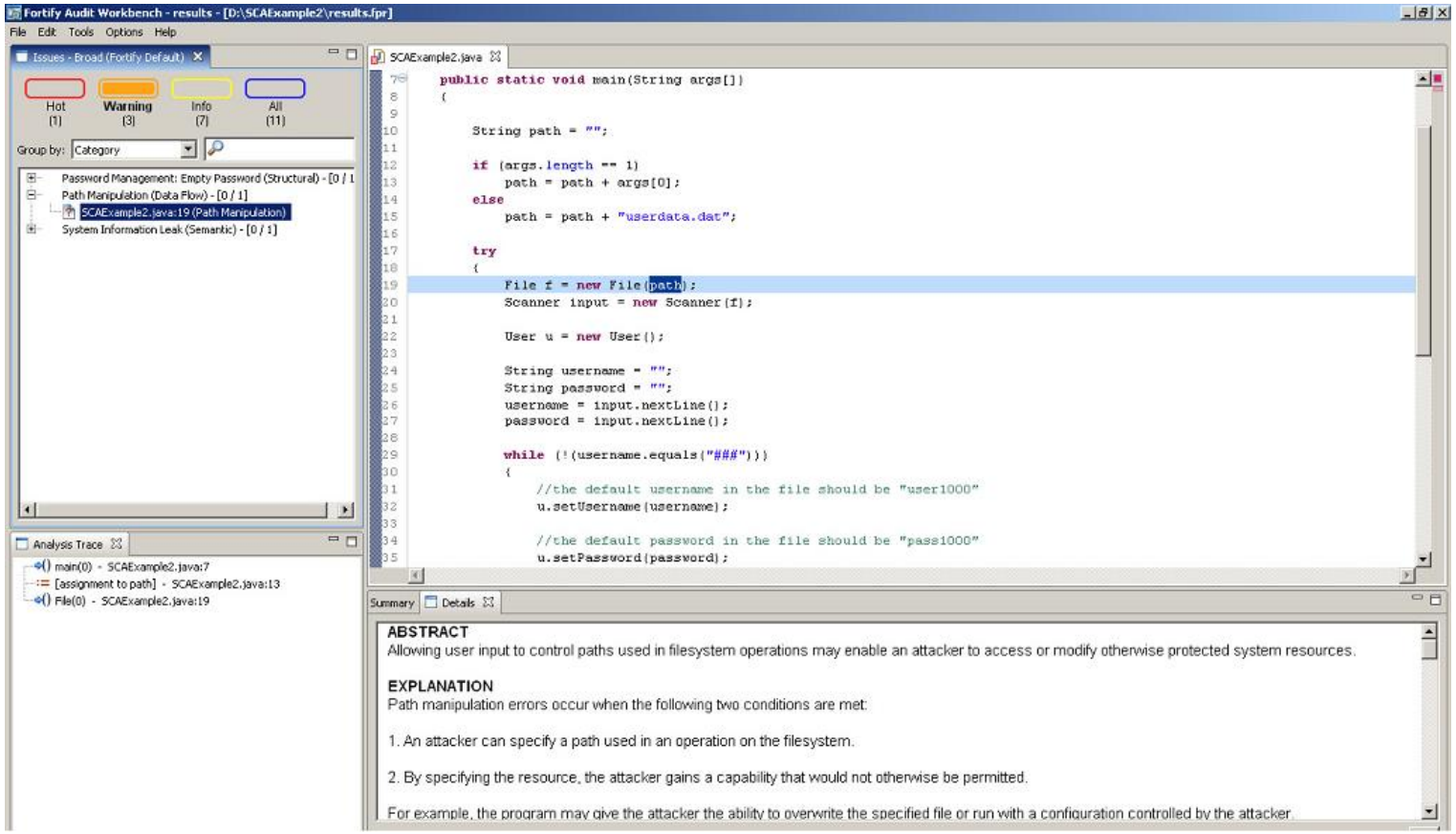
FindBugs

- 扫描+匹配(类似病毒检查)
  - 缺陷模式、字节码



FindBugs UI **Eclipse插件**

❖ 商业软件


Fortify Audit Workbench - results - [D:\SCAExample2\results.fpr]
File Edit Tools Options Help
Issues - Broad (Fortify Default)
Hot (1)
Warning (3)
Info (7)
All (11)
Group by: Category
Password Management: Empty Password (Structural) - [0 / 1
Path Manipulation (Data Flow) - [0 / 1]
SCAExample2.java:19 (Path Manipulation)
System Information Leak (Semantic) - [0 / 1]
SCAExample2.java
public static void main(String args[])
{
String path = "";
if (args.length == 1)
path = path + args[0];
else
path = path + "userdata.dat";
try
{
File f = new File(path);
Scanner input = new Scanner(f);
User u = new User();
String username = "";
String password = "";
username = input.nextLine();
password = input.nextLine();
while (!(username.equals("###")))
{
//the default username in the file should be "user1000"
u.setUsername(username);
//the default password in the file should be "pass1000"
u.setPassword(password);
Analysis Trace
main(0) - SCAExample2.java:7
[assignment to path] - SCAExample2.java:13
File(0) - SCAExample2.java:19
Summary
Details
ABSTRACT
Allowing user input to control paths used in filesystem operations may enable an attacker to access or modify otherwise protected system resources.
EXPLANATION
Path manipulation errors occur when the following two conditions are met:
1. An attacker can specify a path used in an operation on the filesystem.
2. By specifying the resource, the attacker gains a capability that would not otherwise be permitted.
For example, the program may give the attacker the ability to overwrite the specified file or run with a configuration controlled by the attacker.

知识域：软件安全开发的关键阶段

- 知识子域：软件安全测试
  - 了解软件安全测试重要性和基本概念
  - 理解模糊测试的目的和方法
  - 了解渗透测试的目的、步骤和方法

为什么要软件安全测试？

- **软件测试**
  - 按照特定规程，发现软件错误的过程。
  - 检查软件是否满足规定的要求，或是清楚地了解预期结果与实际结果之间的差异
  - 其目的在于发现软件中的错误。
- **软件安全测试**
  - 有关验证软件安全等级和识别潜在安全缺陷的过程
  - 查找软件自身程序设计中存在的安全隐患，并检查应用程序对非法侵入的防范能力
  - **传统测试仅考虑软件出错时的处理，没有考虑对软件的故意攻击**

安全测试

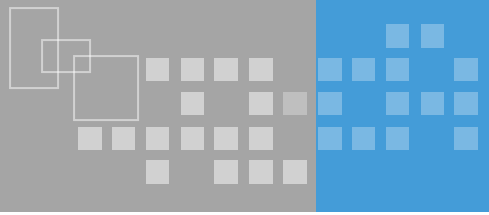
- 在应用投产前，应由独立的安全团队对应用的安全性进行综合评估
  - 功能性安全测试
  - 对抗性安全测试
- 传统测试方法
  - 白盒测试
  - 黑盒测试
  - 灰盒测试
- 特定的安全测试手段
  - 模糊测试
  - 渗透测试

模糊测试

- 模糊测试（Fuzz测试）
  - Baron Miller、Lars Fredriksen、Bryan So首次提出
  - 是一种通过提供非预期的输入并监视异常结果来发现软件故障的方法
- 属于黑盒测试
  - 不关心被测试目标的内部实现
  - 设计输入，检测结果，发现安全漏洞
- 微软SDL包含了Fuzz测试

非常有效的漏洞挖掘技术，已知漏洞大部分都是通过这种技术发现的。

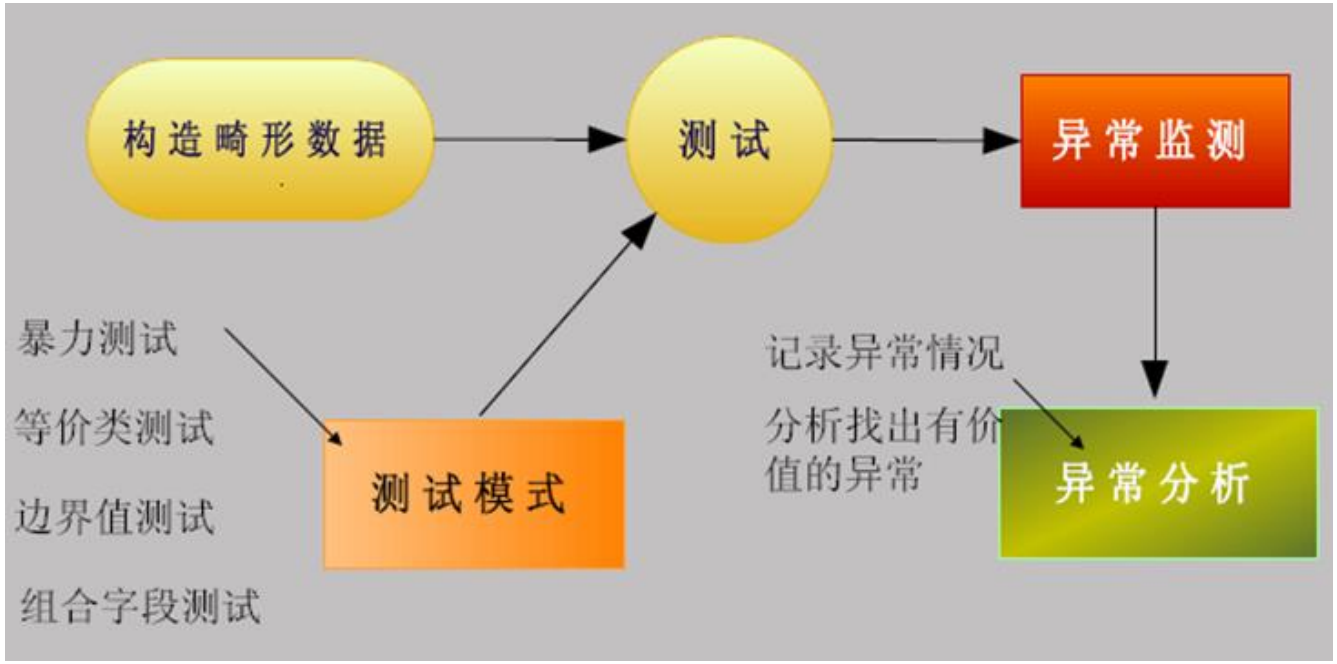
Fuzz测试



- 强制软件程序使用恶意/破坏性的数据并进行观察结果的一种测试方法
  - 不够强壮的程序会崩溃
  - 编码良好的程序正常运行
- 特性
  - 方法学
  - 随机值
  - 大量测试用例
  - 查找漏洞或可靠性错误

Fuzz测试步骤

- 生成大量的畸形数据作为测试用例
- 将这些测试用例作为输入应用于被测对象
- 监测和记录由输入导致的任何崩溃或异常现象
- 查看测试日志，深入分析产生崩溃或异常的原因



一份Fuzz测试结果（1995）

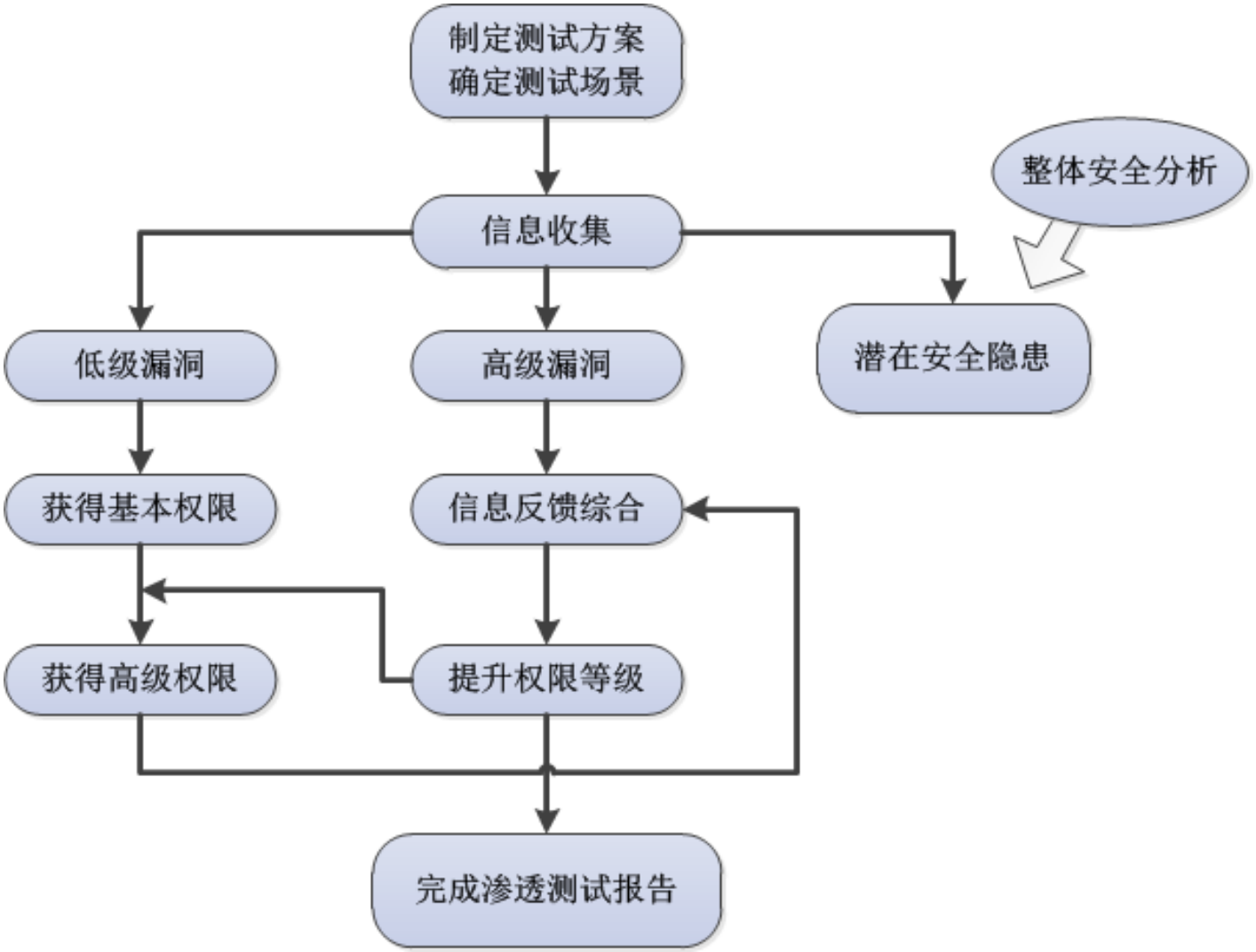
	SunOS	HP-UX	AIX	Solaris	Linux
被测程序个数	80	74	74	70	55
崩溃程序个数	18	13	15	16	5
百分比	23%	18%	20%	23%	9%

来源：1995 Barton P. Miller, University of Wisconsin Madison

渗透测试

- 渗透测试
  - 通过模拟恶意黑客的攻击方法，来评估系统安全的一种评估方法
  - 从攻击的角度测试软件系统是否安全
  - 使用自动化工具或者人工的方法模拟黑客的输入，找出运行时刻目标系统所存在的安全漏洞
- 优点
  - 找出来的问题都是真实的，也是较为严重的
- 缺点
  - 只能到达有限的测试点，覆盖率较低

渗透测试流程


制定测试方案
确定测试场景
整体安全分析
信息收集
低级漏洞
高级漏洞
潜在安全隐患
获得基本权限
信息反馈综合
获得高级权限
提升权限等级
完成渗透测试报告

渗透测试要点

- 测试目的
  - 是进行安全性的评估，不是摧毁或破坏
- 测试人员
  - 技术、知识和经验很重要
  - 像“坏人”一样思考问题
- 安全问题
  - 系统备份和恢复措施
  - 风险规避

＊ 如果测试参数由哪些不想发现安全问题的人所确定，那么，渗透测试就很可能变成一种毫无用处的自我满足的练习！

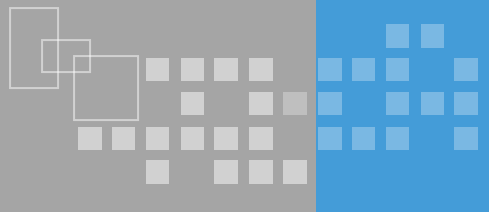
知识域：软件安全开发的关键阶段

- 知识子域：软件安全开发项目管理
  - 了解项目安全需求分析和安全设计重要性
  - 了解软件安全开发角色
  - 了解安全培训重要性和内容
  - 了解如何实施自己的软件安全计划

项目安全需求分析和安全设计

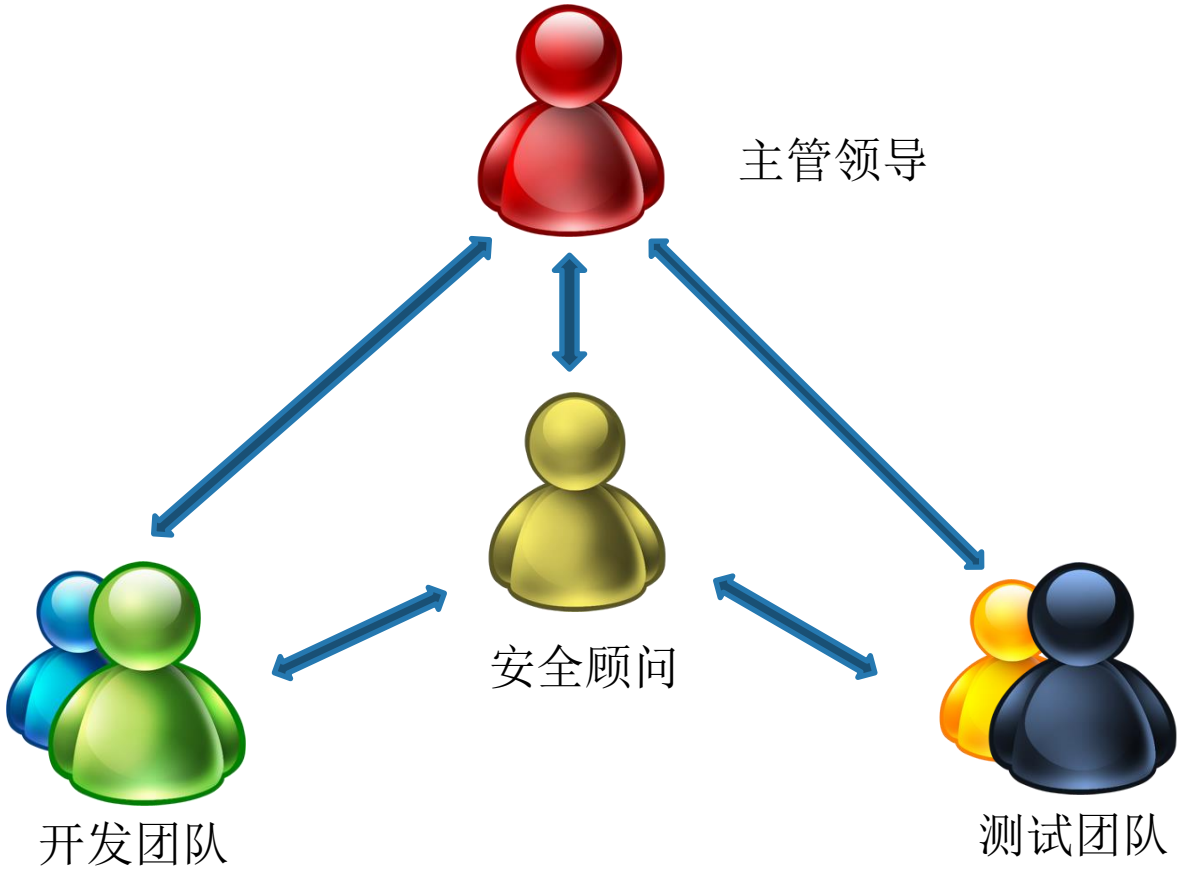
- 编写安全的软件的需要从一开始就关注安全性
  - 很多系统是后来才添加了安全性功能部件
  - Windows9x
- 开发和划分安全性需求
  - 充分理解和完全文档化的需求
  - 分析安全场景和安全风险
- 注重安全设计
  - 应用软件结构选择
  - 系统平台确定
  - 编程/开发软件选择

软件安全开发角色

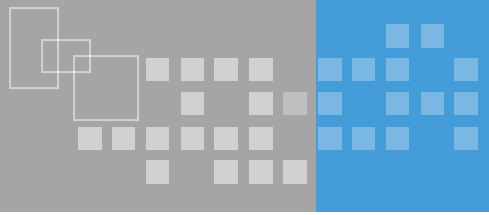


- 项目高管
  - 关心安全问题
- 安全群组/安全顾问（SSG）
  - BSIMM3：42个项目中，都有一个相当规模的SSG
- 开发团队
  - 项目经理、架构师、开发人员
- 测试团队
  - 独立测试团队

软件安全开发角色



加强安全培训



- 软件开发团队需要接受专门的安全培训
- 定期/非定期培训
  - 新员工培训
  - 新知识培训
  - 安全意识培训
  - …
- 培训内容可包括：
  - 信息安全基础知识、密码技术、网络安全
  - 威胁建模、架构设计
  - 安全编码训练、编程习惯
  - …

实施软件安全计划

- 1、建立适合自己的计划。
  - 适应项目的商业和技术环境，确定最佳路径，逐步调整
- 2、认真实施每一个最优方法计划
  - 确定负责人，局部实验->推广
- 3、培训人员
  - 必须的
- 4、评估进展
  - 风险评估、监测项目进展
- 5、持续改进

灵活安排自己的“组合”

- 代码审核 + 体系结构风险评估
- 基于风险的安全测试 + 渗透测试
- 安全需求分析 + 滥用案例开发
- 代码审核 + 渗透测试
- 体系结构风险分析 + 基于风险的测试
- …

谢谢，请提问！