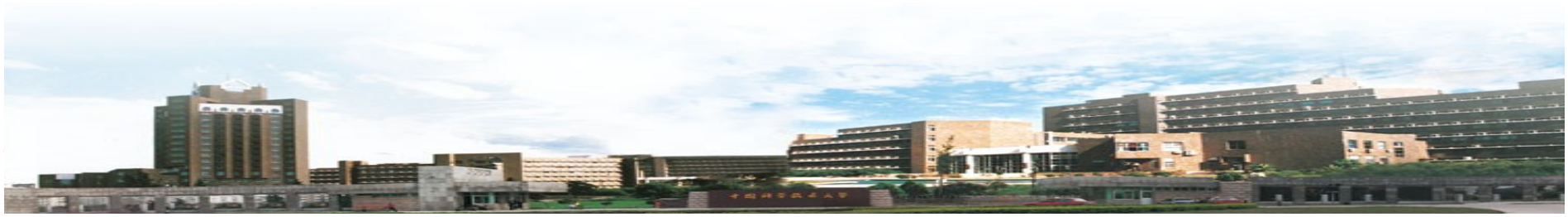
软件安全性测试

程绍银

sycheng@ustc.edu.cn



大纲

- 软件安全性测试
- 安全性测试主要方法
- 软件系统的安全性测试举例

大纲

- ✓ 软件安全性测试
- 安全性测试主要方法
- 软件系统的安全性测试举例

软件安全性测试

- 安全性测试的概念
- 安全性测试的特点
- 安全性测试的分类
- 安全性测试的目标
- 安全性测试的方法学和方法示例
- 软件最优方法 VS. 安全性测试方式示例

安全性测试的概念

- 软件安全性测试是确定软件的安全特性实现是否与预期设计一致的过程
- 软件安全性测试是在软件的生命周期内采取的一系列措施，用来防止出现有违反安全策略的异常情况和在软件的设计、开发、部署、升级以及维护过程中的潜在系统漏洞

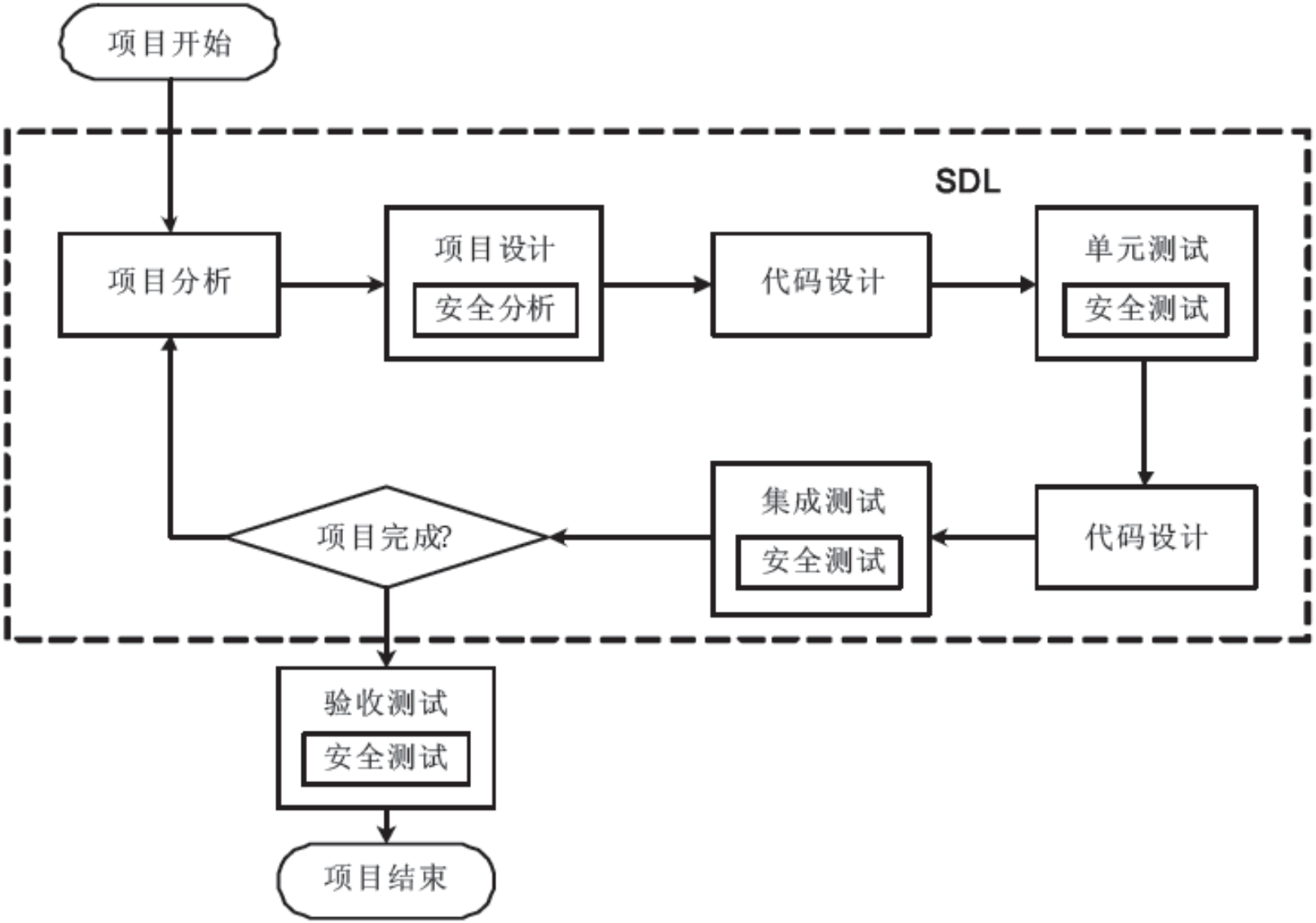
安全性测试的特点

- 安全性测试不同于传统测试类型最大的区别是，它强调软件不应当做什么，而不是软件要做什么
- 功能性测试：软件做它应该做的事
  - 应用输入，验证正确的输出
  - 不正确的输出/行为 → 缺陷(Bug)
  - 非安全性缺陷常常是违反规约，即软件应当做A，它却做了B
- 安全性测试：软件不做它不应该做的事
  - 应用输入，验证没有不安全的事情发生
  - 安全性缺陷（漏洞，脆弱性，Vulnerability）
  - 安全性缺陷常常由软件的副作用引起，即软件应当做A，它做了A的同时，又做了B

安全性测试的分类

- 安全功能测试
  - 基于软件的安全功能需求说明，测试软件的安全功能实现是否与安全需求一致，需求实现是否完备
  - 主要安全功能需求包括数据机密性、完整性、可用性、不可否认性、身份认证、授权、访问控制、审计跟踪、委托、隐私保护、安全管理等
- 安全漏洞测试
  - 从攻击者的角度，以发现软件的安全漏洞为目的
  - 安全漏洞是指系统在设计、实现、操作、管理上存在的可被利用的缺陷或弱点
  - 漏洞被利用可能造成软件受到攻击，使软件进入不安全的状态，安全漏洞测试就是识别软件的安全漏洞

将安全性测试整合到软件开发生命周期中



安全性测试的主要工作

- 全面检验软件在软件需求规格说明中规定的防止危险状态措施的有效性和在每一个危险状态下的反应
- 对软件设计中用于提高安全性的结构、算法、容错、冗余、中断处理等方案，进行针对性测试
- 在异常条件下测试软件，以表明不会因可能的单个或多个输入错误而导致不安全状态
- 用错误的安全性关键操作进行测试，以验证系统对这些操作错误的反应
- 对安全性关键的软件单元和软件部件，要单独进行加强的测试，以确认其满足安全性需求

安全性测试的目标

- 通常的目标
  - 内存溢出
  - SQL injection/XSS
  - 各种输入验证型问题
- 进一步的目标
  - 访问控制
  - 信息泄露
  - 不充分的随机数
  - 鉴别与加密
  - …

安全性测试的方法学

	Static	Dynamic
Black Box	Examine MRD/PRD and specs	Fuzzing Fault injection
White Box	audit source	Debug, unit test

安全性测试的方法学

- White Box
  - Use commercial statistic analysis tool to audit the source code
  - Coverity/Fortify/Etc..
- Black Box
  - Fault injection
  - Dumb Fuzzing
- Gray Box
  - Smart Fuzzing (Fuzzing with deeply knowledge of the product)

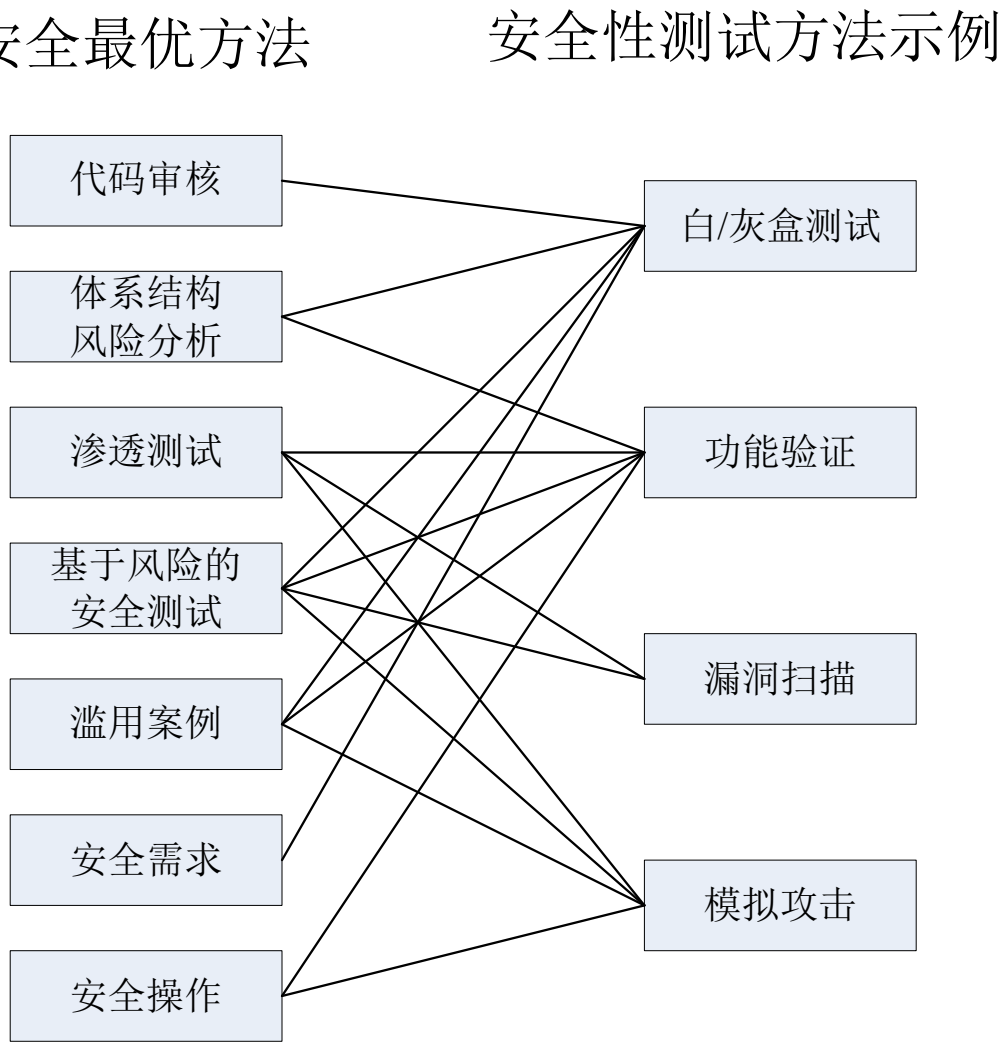
安全性测试的方法示例

- 白/灰盒测试
  - 对软件工程文档/源代码/二进制代码进行静态分析/审核
  - 对运行时系统进行动态监测
  - 例子：污点传播分析/符号执行
- 功能验证
  - 功能验证是采用软件测试当中的黑盒测试方法，对涉及安全的软件功能，如：用户管理模块、权限管理、加密系统、认证系统等进行测试，主要验证上述功能是否有效

安全性测试的方法示例

- 漏洞扫描
  - 安全漏洞扫描主要是借助于特定的漏洞扫描器完成的。通过使用漏洞扫描器，系统管理员能够发现系统存在的安全漏洞，从而在系统安全中及时修补漏洞的措施
  - 分为两种类型：主机漏洞扫描器是指在系统本地运行检测系统漏洞的程序；网络漏洞扫描器是指基于网络远程检测目标网络和主机系统漏洞的程序
- 模拟攻击
  - 对于安全测试来说，模拟攻击测试是一组特殊的极端的测试方法，以模拟攻击来验证软件系统的安全防护能力
  - 例子：Fuzzing，使用大量半有效的数据作为应用程序的输入，以程序是否出现异常为标志，来发现应用程序中可能存在的安全漏洞

安全最优方法 VS. 安全性测试



大纲

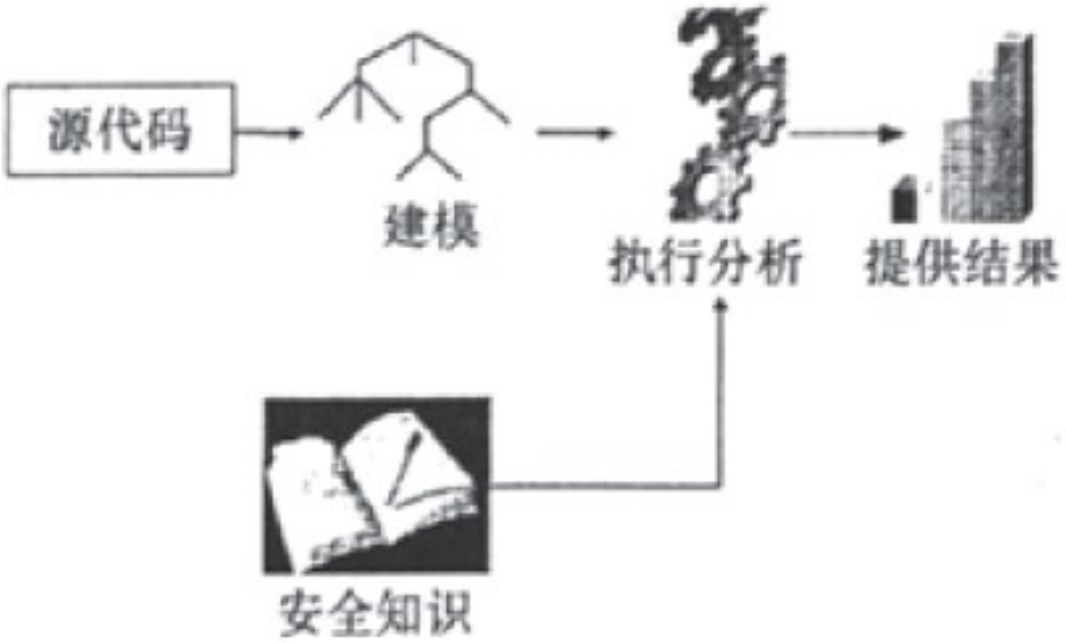
- 软件安全性测试
- ✓ 安全性测试主要方法
- 软件系统的安全性测试举例

安全性测试主要方法

- 静态分析测试
- 基于模型的安全测试
- 基于故障注入的安全性测试
- 基于语法的安全性测试
- 模糊测试
- 基于属性的安全性测试
- 形式化安全性测试
- 基于风险的安全性测试
- 基于故障树的安全性测试技术
- 基于渗透的安全性测试

静态分析测试

- 静态分析是指在不执行代码的情况下对其进行评估的过程
- 建模：代码→程序模型，即一组代表此代码的数据结构
  - 词法分析
  - 语法分析
  - 语义分析
  - 程序控制流
  - 数据流分析

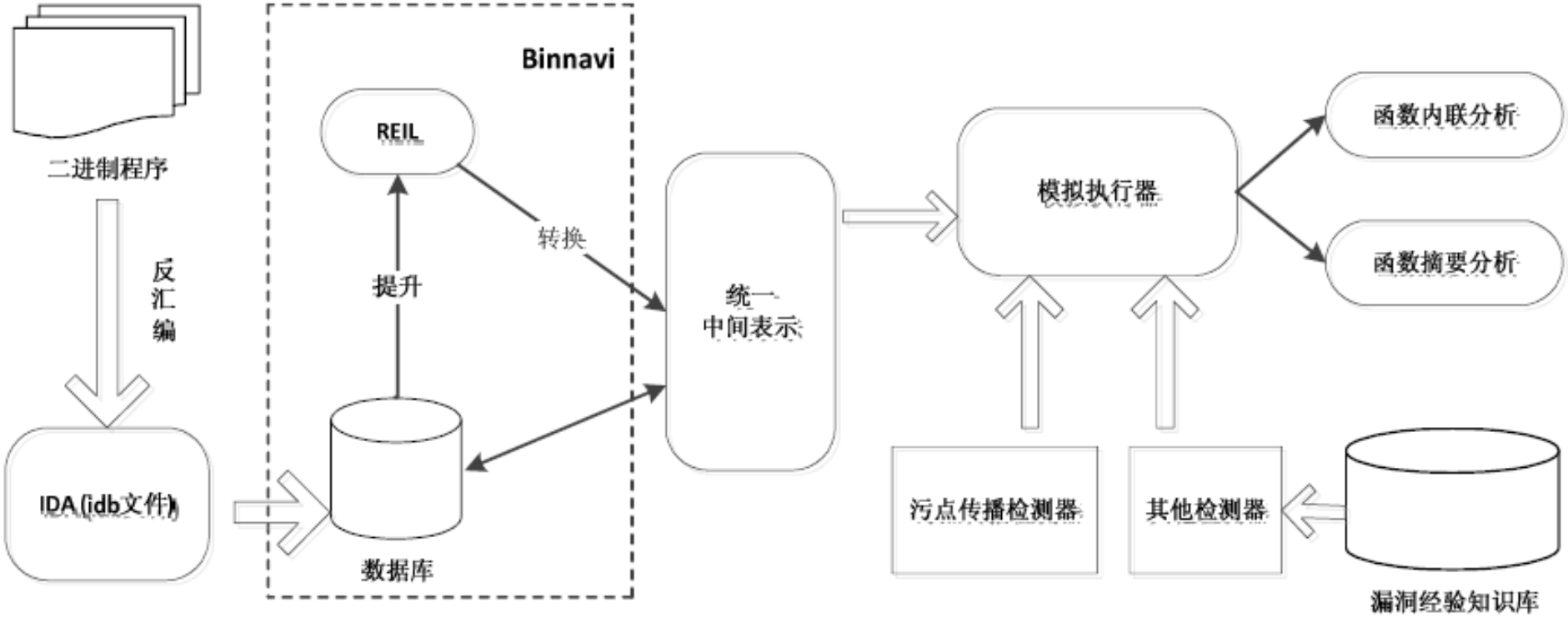


静态分析测试

- 优点
  - 通过对代码的检查，往往能指出安全问题的根源
  - 能够在开发早期发现错误
  - 当安全研究人员发现一种新的攻击时，静态分析工具可以容易地对大量代码进行重新检查
- 缺点
  - 普遍存在的问题是误报和漏报现象，因此要在误报和漏报之间进行平衡

静态分析测试

二进制程序的静态分析



基于模型的安全测试

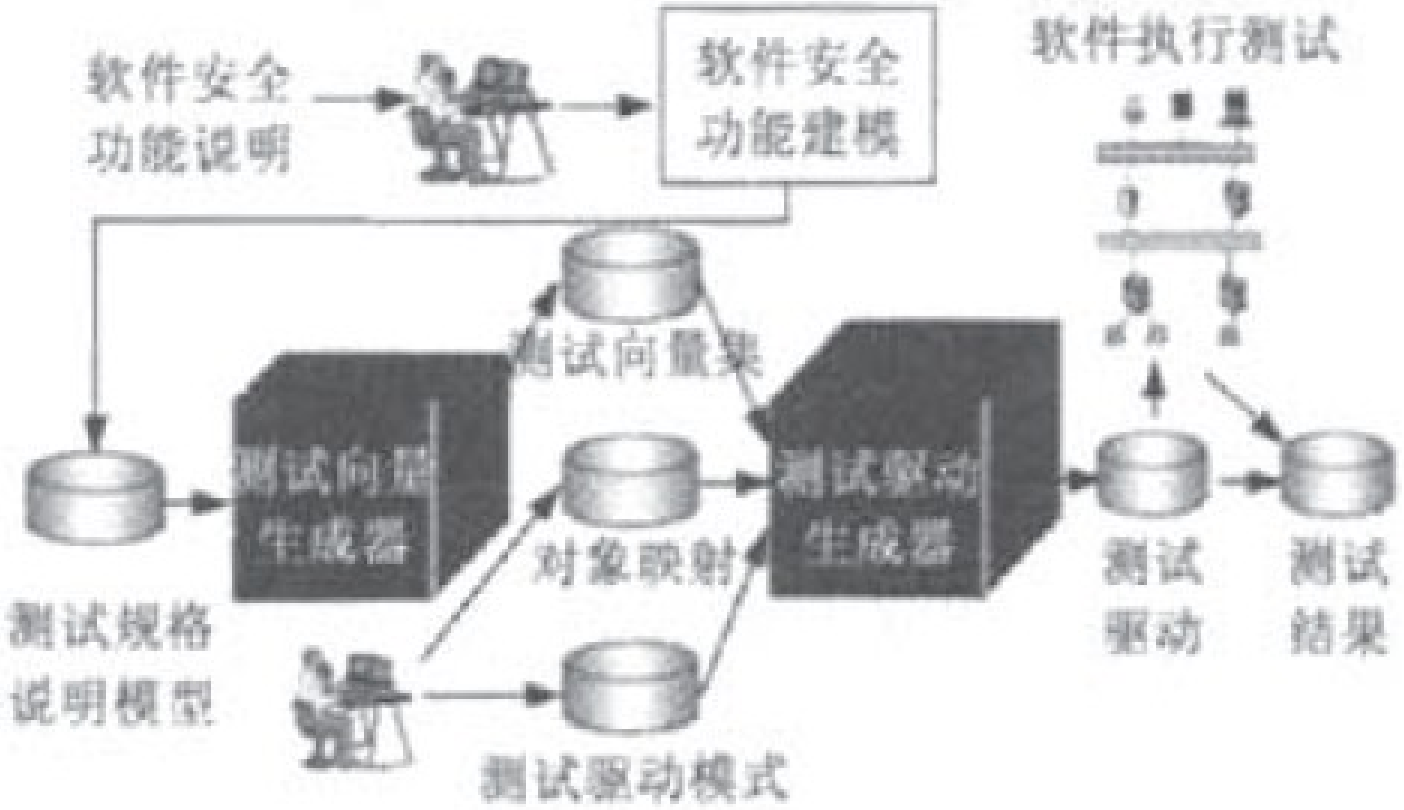
- 基于模型的安全性测试是对软件的结构和行为进行建模，生成相应的测试模型，再由测试模型自动生成测试用例，以驱动安全性测试
- 常用的软件测试模型有有限状态自动机、UML模型、马尔可夫链等
- 适用范围取决于安全功能的建模能力
  - 特别适用于建模用与或子句表达逻辑关系的安全需求，对授权、访问控制等安全功能测试比较适用

以UML模型为例的方法流程

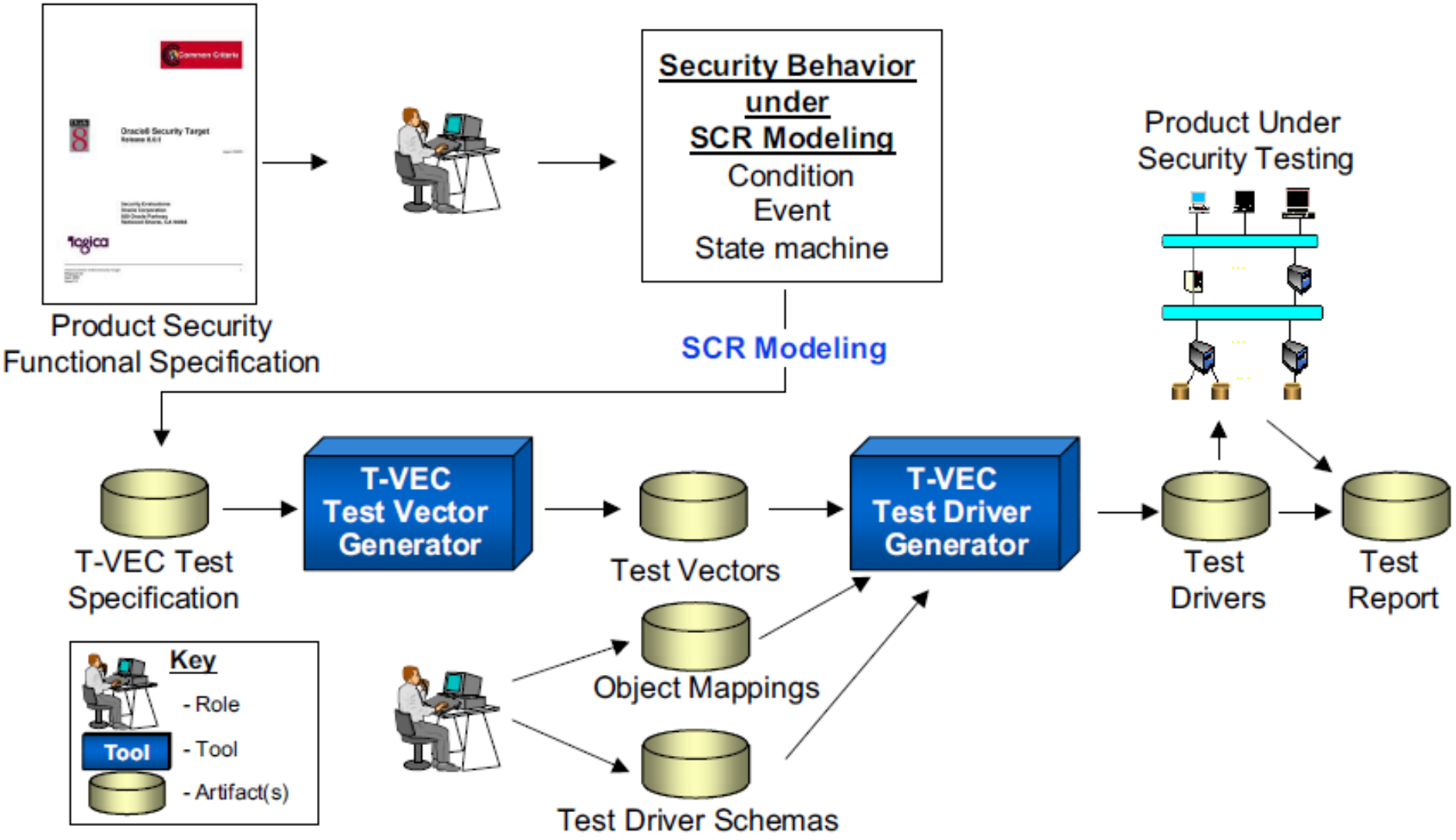
- 安全性威胁首先被以UML序列图的形式进行建模
- 以威胁建模为基础，提取威胁路径的集合
  - 在程序执行过程中这些威胁都是不应该发生的
- 代码用随机生成的测试用例来执行，执行的路径被收集起来，并与定义好的威胁路径进行比较
  - 如果一个执行路径符合其中某一条威胁路径，则其需要被上报并处理

自动化安全功能测试处理流程

- Ramaswamy Chandramouli, Mark Blackburn．Automated testing of security functions using a combined model and interface-driven approach．Big Island, HI, USA: Proc 37th Hawaii International Conference on System Sciences,2004: 5-8.



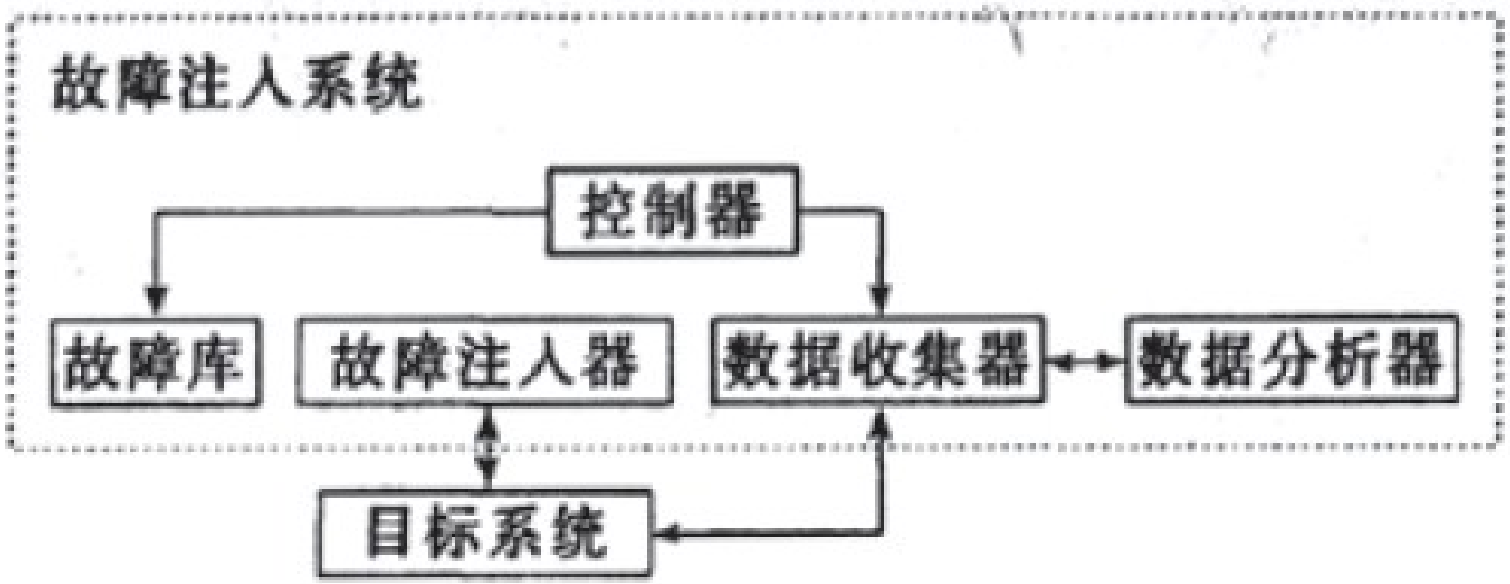
Test Automation Framework for Security Functional Testing



基于故障注入的安全性测试

- 故障注入是评测容错机制的一种有效方法。通过人为方式将故障引人到系统当中，加速系统发生故障的失效的过程
- 针对应用与环境的交互点，主要包括用户输入、文件系统、网络接口、环境变量等引起的故障
- 故障注入可以有效地模拟各种异常程序行为，通过故障注人函数能强制使程序进入某些特定状态，而这些状态在采用常规的标准测试技术时是无法达到的

故障注入系统的一般结构图



当前故障注入的主要方法

- 仿真故障注入
  - 需要以较好的目标系统仿真模型为基础
- 硬件故障注入
  - 需要专业的硬件设备(也可通过软件方法模拟实现)
- 软件故障注入
  - 对目标系统硬件环境没有任选损坏，能方便地跟踪目标程序的执行并回收数据，系统开稍小，且有较好的可移植性
- 基于环境混乱的故障注入
  - 把应用程序和其运行环境都纳入系统的范畴，通过改变正常的环境因素(如文件，网络等)来测试系统对环境故障的客错能力

当前比较成熟的软件故障注入工具

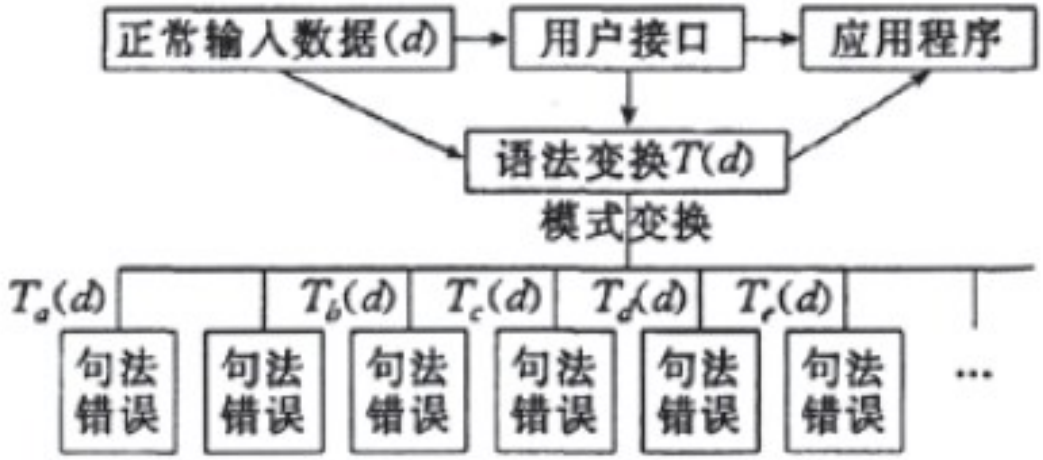
- FIAT
  - 可能破坏任务的内存映像
- FERRARI
  - 利用UNIX的ptrace函数在运行期间破坏程序的内存映像，在故障被激活的地方插入陷阱指令
- Xception
  - 在目标处理器内部直接编程调试硬件，并带有许多故障触发器来激活故障

基于语法的安全性测试

- 语法测试是根据被测软件的功能接口的语法生成测试输入，检测被测软件对各类输入的响应
  - 接口可以有多种类型，命令行、文件、环境变量、套接字等
- 语法测试的思想：软件的接口或明确或隐含规定了输入的语法
  - 语法定义了软件接受的输入数据的类型、格式
  - 语法定义可采用BNF或正则表达式
- 语法测试适用于被测软件有较明确的接口语法，易于表达语法并生成测试输入的情况
- 语法测试结合故障注入技术可得到更好的测试效果

基于语法的安全性测试

- 语法测试的步骤
  - 识别被测软件接口的语言，定义语言的语法
  - 根据语法生成测试用例并执行测试。生成的测试输入应当包含各类语法错误，符合语法的正确输入，不符合语法的畸形输入等
  - 通过察看被测软件对各类输入的处理情况，确定被测软件是否存在安全缺陷



模糊测试

- Fuzzing：基于黑盒的随机性测试
  - 通过随机的变异正常的程序输入来检测程序的响应，以发现程序中隐藏的安全漏洞
- 模糊测试可以使用语法规则来产生正常的输入，也可以使用基于特定程序的输入并用试探法来指导输入变量的生成
- 模糊测试非常有效，不会误报，但漏报较多，代码覆盖率较低
  - 设x是一个随机的32位输入
  - 语句“z = 1/(x-7)”的除零缺陷，只有2^{32}分之一的概率被发现
  - 语句“if (x == 7) then”的then分支只有2^{32}分之一的概率被测试到

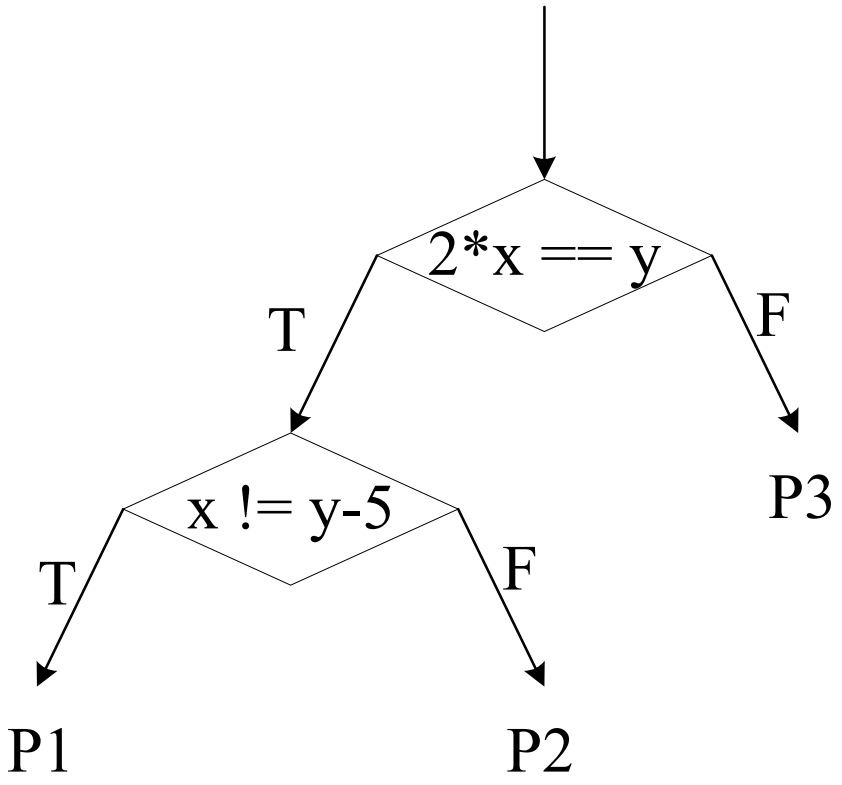
模糊测试

- 模糊测试的主要分类
  - 文件格式类Fuzzing
  - 网络协议类Fuzzing
- 当前比较成熟的Fuzzing工具
  - Peach
  - Spike
  - Filefuzz
- 一般根据Fuzzing思想，自行编写Fuzzing工具

Whitebox Fuzzing

- Smart Fuzzing / Whitebox Fuzzing
- 结合符合执行+具体执行，以及约束求解技术

- 符号值：x, y
- 执行步骤P1->P2->P3
  - seed: x = 0, y = 0
  - 具体执行路径P1
    - 符号执行收集路径条件
      2*x == y && x != y-5
  - 对路径子条件取反
    - 2*x == y && x == y-5
    - 求解得 x =5, y = 10



Whitebox Fuzzing经典论文

- G. Patrice, K. Nils, and S. Koushik, "DART: directed automated random testing," in *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation* Chicago, IL, USA: ACM, 2005.
- C. Cristian, G. Vijay, M. P. Peter, L. D. David, and R. E. Dawson, "EXE: automatically generating inputs of death," in *Proceedings of the 13th ACM conference on Computer and communications security* Alexandria, Virginia, USA: ACM, 2006.
- S. Koushik, M. Darko, and A. Gul, "CUTE: a concolic unit testing engine for C," in *Proceedings of the 10th European software engineering conference held jointly with 13th* ACM *SIGSOFT international symposium on Foundations of software engineering* Lisbon, Portugal: ACM, 2005.
- P. Godefroid, M. Levin, and D. A. Molnar, "Automated Whitebox Fuzz Testing," Microsoft ResearchMay 2007.
- G. Patrice, K. Adam, and Y. L. Michael, "Grammar-based whitebox fuzzing," in *Proceedings of the 2008 ACM SIGPLAN conference on Programming language design and implementation* Tucson, AZ, USA: ACM, 2008.
- L. Andrea, M. Lorenzo, M. Mattia, and P. Roberto, "A Smart Fuzzer for x86 Executables," in *Proceedings of the Third International Workshop on Software Engineering for Secure Systems*: IEEE Computer Society, 2007.

基于属性的安全性测试

- 首先确定安全编程规则，然后把这些规则编码以作为安全性属性，之后就可以此来验证程序代码是否遵守了这些规则
- 由于人工验证的成本太高，需要一个程序分析工具来自动化完成这一过程。一个可行的方法是：
  - 把安全属性形式化描述为一个有限状态自动机
  - 把待分析的代码模型化为一个下推自动机
  - 利用模型验证技术来确定模型中的任何一个违反安全属性的状态在程序中是否是可达的
- 基于属性的测试有针对性的测试目标软件的特定安全属性，可满足安全属性的分类和优先级排序要求，且部分与具体软件无关的属性规格说明是可重用的

形式化安全性测试

- 形式化安全测试的基本方法是建立软件的数学模型，在形式规格说明语言的辅助下，提供形式化的规格说明
- 形式规格说明语言主要有
  - 基于模型的Z、VDM和B语言
  - 基于有限状态语言，如有限状态自动机、SDL和状态图
  - 基于行为的CSP、CCS、LOTOS和Petri Nets等语言
  - 代数语言，如OBJ
  - 混合语言，如离散和连续数学的规格说明语言等

形式化安全性测试的分类

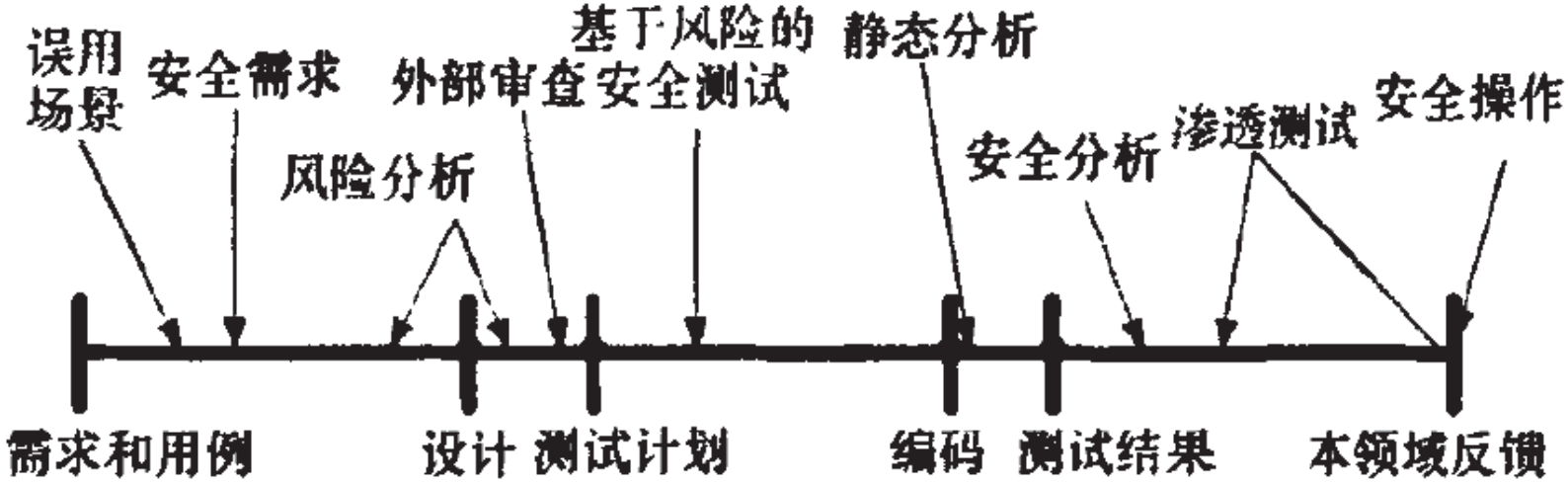
- 模型检测
  - 一种针对有限状态迁移系统的高效自动验证技术，它主要通过显式状态搜索或隐式不动点计算来验证有穷状态并发系统的模态命题性质，已被广泛应用于计算机硬件、通信协议、控制系统、安全认证协议等方面的分析与验证
  - 用状态迁移系统S描述软件的行为，用时序逻辑、计算树逻辑或μ演算公式F表示软件执行必须满足的性质，通过自动搜索S中不满足公式F的状态来发现软件中的漏洞
  - 2007图灵奖，Edmund M. Clarke、E Allen Emerson和Joseph Sifaki，主要贡献：Model Checking。其中，Edmund M. Clarke曾于2009年10月来我校访问
- 定理证明
  - 将程序转换为逻辑公式，然后使用公理和规则证明程序是一个合法的定理

形式化安全性测试的优缺点

- 优点
  - 形式化方法具有完备的数学基础，因而形式规格说明是精确的
  - 通过揭示不一致性、二义性和不完全性，能够有效增加对系统的深人理解
  - 能根据形式规格说明进行正确性证明
- 缺点
  - 开发成本较高、维护困难
  - 状态爆炸问题
- 当前主要成熟工具：Spin，Verisoft，Java Path Finder

基于风险的安全性测试

- 风险是指错误发生的可能性和造成的危害程度的结合
- 基于风险的安全性测试：以软件安全风险作为测试的出发点和测试活动的主要参考依据
- 把风险分析与管理、安全测试以及软件开发过程统一起来，在软件开发的各个阶段中就把有风险的安全漏洞考虑在内，将安全测试与软件开发同步进行



基于风险的安全性测试

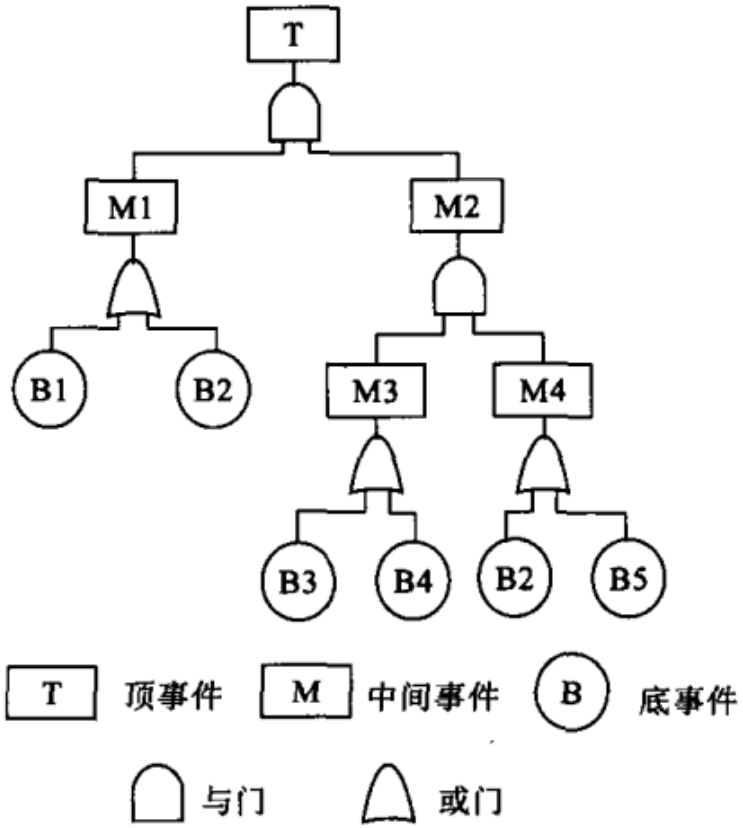
- 通过误用模式、异常场景、风险分析以及渗透测试等技术来处理具有风险的安全问题
- 基于风险的测试以软件模块的质量风险为主要参考依据，来进行测试力量的分配
- 基于风险的安全测试技术可以实现尽早地发现尽可能多的潜在安全问题，以最少的资源、最短的时间有效地达成用户需求，并确保合适的软件品质以避免大量的后期维护工作

基于故障树的安全性测试技术

- 基于故障树的安全测试技术是利用故障分析树和故障树的最小割集来生成安全性测试用例的方法
- 故障树分析法(Fault Tree Analysis，FTA)是一种将系统故障形成原因由总体到部分按树状细分的分析方法
- 基于故障树的安全性测试可以显著提高测试的自动化程度，对大型复杂软件系统的测试具体侧重性，能够有效提高测试效率，而且还是非常充分的安全性测试

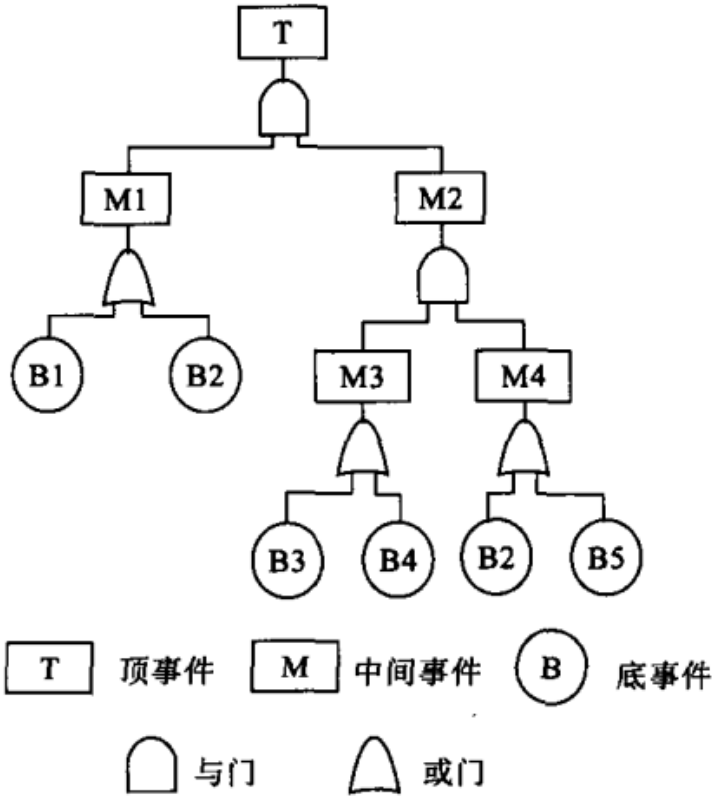
故障树

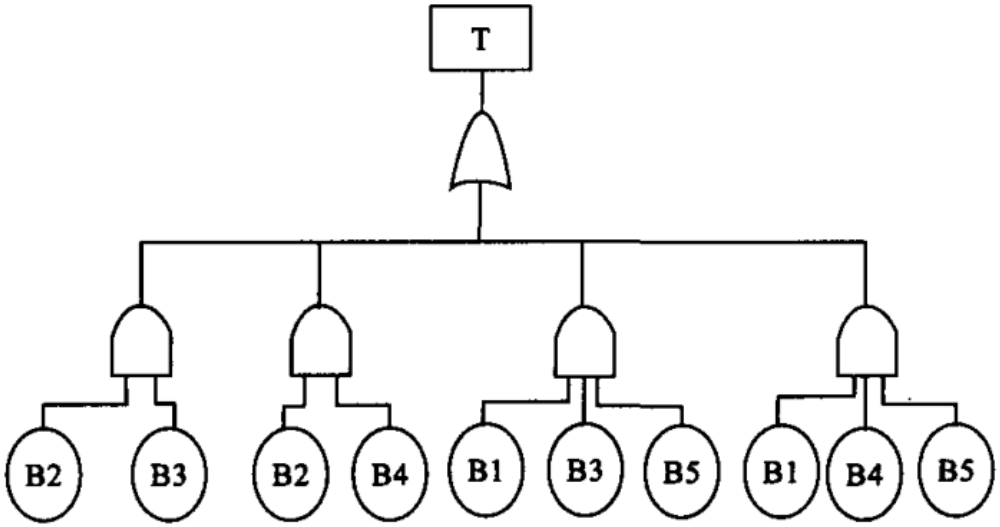
- 故障树(Fault Tree，FT)
  - 以系统最不希望发生的事件作为故障分析树的根(顶事件)
  - 寻找导致这一故障发生的全部可能因素(中间事件和底事件)
  - 用倒立树状图形表示出顶事件与底事件之间的逻辑关系
  - 用相应的事件符号代表这些事件，再用适当的逻辑门符号把顶事件、中间事件和底事件连接起来



最小割集

- 在由故障树的某几个底事件组成的集合中，如果该集合中的底事件同时发生能引起顶事件的发生；并且当去掉此集合中任何一个底事件将不再能引起顶事件的发生，则这个集合被称为最小割集





基于渗透的安全性测试

- 渗透测试(Penetration Testing)是一个评估主机系统和网络的安全性时模仿黑客特定攻击行为的过程
- 安全测试工程师尽可能真实地模拟黑客使用的漏洞发现技术和攻击手段，对目标的安全性作深入的探测，发现系统最薄弱环节的过程
- 渗透测试一般可分为两类，被动攻击和主动攻击
  - 被动攻击不采用直接进入目标系统的方式去收集信息
  - 主动攻击则直接侵入目标系统或网络内部收集所需要的信息

基于渗透的安全性测试

- 一次完整的渗透测试步骤为：收集信息，抽取出对网络渗透有用的信息，制定渗透策略并进行漏洞测试，模拟攻击的真实过程，最后整理收集到的信息并提交测试报告
- 渗透测试比较真实有效，发现的漏洞一般都是真实存在的，而且较为严重
- 但是由于模拟的测试只能达到有限的测试点，覆盖率较低。根据美国权威机构统计，渗透测试的覆盖率只有达到20%"-30%，而且存在漏报率较高的情况

安全性测试方法特点比较

测试方法	特点
静态分析测试	无需运行代码，易于自动化实现；可以找到安全问题的根源而非症状；存在漏报和误报现象；具有不可判定性[3]。
基于模型的安全测试	根据被测软件来设计、生成测试模型和测试用例；较大地提高测试自动化水平和测试效率；部分解决了测试失效辨识问题[19]；利于测试用例的重用；往往能发现其他测试方法不能发现的故障。
基于故障注入的安全测试	人为将错误引入系统，加速系统失效，测试系统容错性。
模糊测试	使用语法规则随机地生成程序输入，或对输入进行变异后用于测试程序响应，以发现程序安全漏洞；代码覆盖率不高。
基于属性的安全测试	创建安全规则，把规则编码为安全属性，再使用人工或自动化的方法来验证程序是否违反了这些安全规则。
形式化安全测试	具有严格的数学基础；可根据形式规则生成测试用例并进行正确性证明。
基于风险的安全测试	以软件模块的质量风险为依据，在软件开发生命周期中进行测试力量的合理分配。
基于故障树的安全测试	使用故障树来分析发生某一故障需具备的因素或因素组合；故障树安全测试具有充分性。
基于渗透的安全测试	模仿黑客行为对主机或网络进行攻击；具有真实性、有效性；覆盖率不高。

主流安全性测试工具分类

工具类型	商用	开源或免费
源代码分析器	Fortify Software 公司的 Source Code Analysis Suite；Ounce Labs 实验室的 Prexis；Secure Software 公司的 CodeAssure；Klocwork 公司的 K7；Coverity 公司的 Prevent；Compuware 公司的 DevPartner SecurityChecker；Parasoft 公司的 C++Test、JTest、TEST、WebKing；CenterLine Systems 公司的 CodeCenter；CodeScan Labs 实验室的 CodeScan；GrammaTech 公司的 CodeSonar；SPI Dynamics 公司的 DevInspect；SofCheck 公司的 SoftCheck Inspector；PolySpace Technologies 公司的 PolySpace	Rough Auditing Tool for Security (RATS)；lawFinder；BOON；ASTREE；C Code nalyzer(CCA)；Csur；CQual；Jlint；ITS4；Splint；UNO；LAPSE；PHP-Sat；PMD
字节码扫描器	LogicLab 公司的 BugScan	FindBugs
二进制代码扫描器	Aspect Security 公司的 AspectCheck；Security Innovation 公司的 BEAST；LogicLab 公司的 BugScan	FxCop；BugScam
数据库脆弱性扫描器	Application Security 公司的 AppDetective；Internet Security Systems 公司的 Database Scanner	MetaCortex
网络漏洞扫描器	Internet Security Systems 公司的 Internet Security Scanner；NT Objectives 公司的 NTOSpider；GFI 公司的 GFI LANguard；eEye 公司的 Retina；Advanced Research Corporation 公司的 Security Auditor's Research Assistant (SARA)；Saintcorporation 公司的 SAINT	NMAP；Nessus；SuperScan；Metasploit Framework
Web 应用漏洞扫描器	SPI Dynamics 公司的 WebInspect；Watchfire 公司的 AppScan；N-Stalker 公司的 N-Stalker Web Application Security Scanner；Acunetix 公司的 Acunetix Web Vulnerability Scanner；portswigger.net 公司的 Burp suite	OWASP WebScarab；OWASP Berretta；Nikto；Wikto；Paros ProxySpike Proxy；EOR；Pantera
Web 服务扫描器	Parasoft 公司的 SOATest；Vordel 公司的 SOAPbox；Optimyz 公司的 WebServiceTester；CrossCheck 公司的 SOAPSonar；Forum Systems 公司的 XRay Diagnosis；MindReef 公司的 SOAPScope；Empirix 公司的 e-TEST suite for Web Services Testing；AdventNet 公司的 QEngine；ITKO 公司的 LISA Complete SOA Test Suite	Foundstone WSDigger；Pushtotest TestMaker
动态分析工具	Compuware BoundsChecker	Foundstone .NETMon；CLR Profiler；NProf
配置分析工具	Desaware CAS/Tester	Foundstone SSLDigger；PermCalc
设计验证工具	SDMetrics 公司的 SDMetrics	

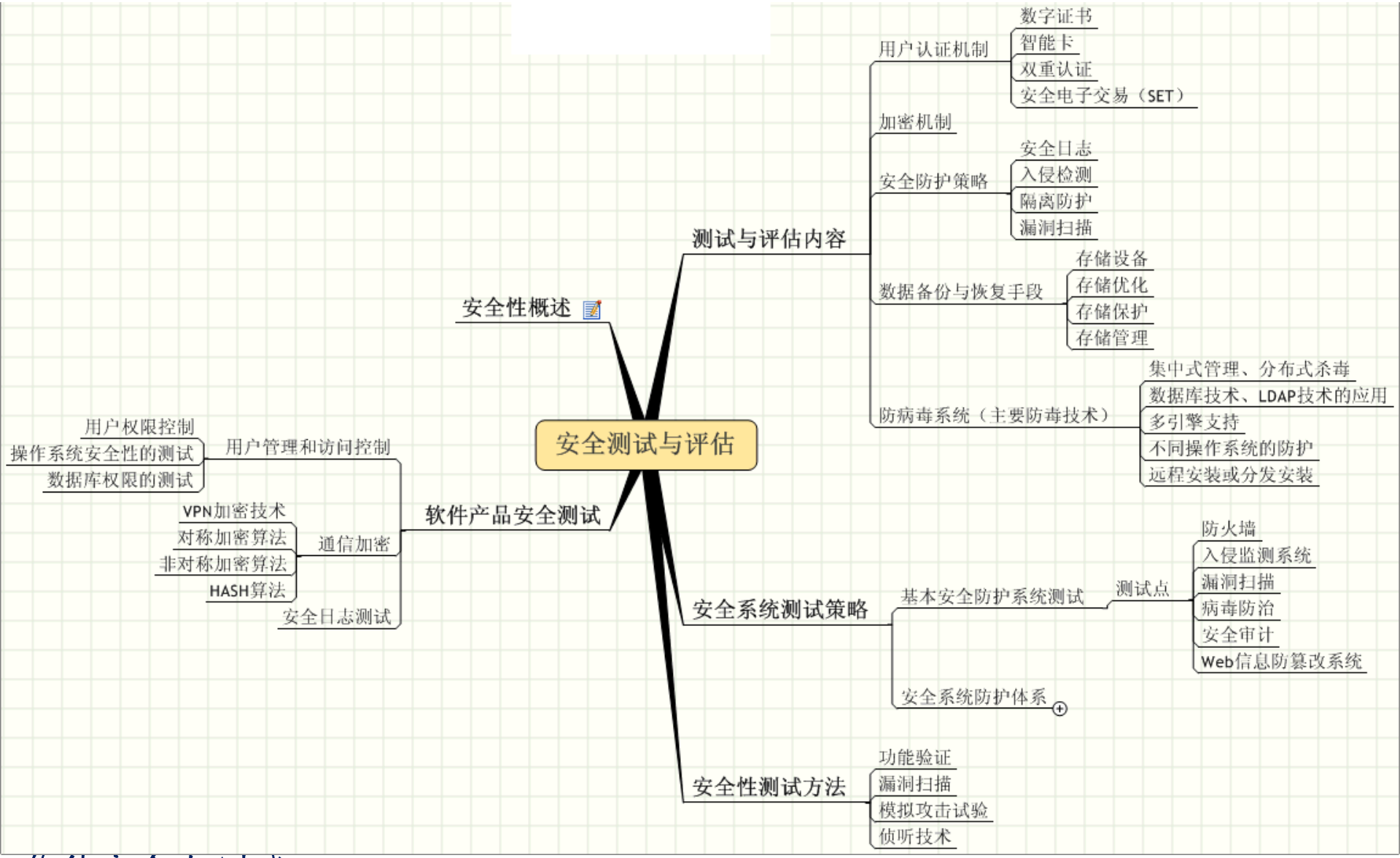
大纲

- 软件安全性测试
- 安全性测试主要方法
- ✓ 软件系统的安全性测试举例

软件系统的安全性测试举例

- 根据软件系统安全指标不同，测试策略也不同
- 安全测试与评估
- 软件系统安全性测试列表(Checklist)
- 手机应用软件安全性测试

安全测试与评估


安全测试与评估
安全性概述
测试与评估内容
用户认证机制
数字证书
智能卡
双重认证
安全电子交易（SET）
加密机制
安全防护策略
安全日志
入侵检测
隔离防护
漏洞扫描
数据备份与恢复手段
存储设备
存储优化
存储保护
存储管理
防病毒系统（主要防毒技术）
集中式管理、分布式杀毒
数据库技术、LDAP技术的应用
多引擎支持
不同操作系统的防护
远程安装或分发安装
软件产品安全测试
用户管理和访问控制
用户权限控制
操作系统安全性的测试
数据库权限的测试
通信加密
VPN加密技术
对称加密算法
非对称加密算法
HASH算法
安全日志测试
安全系统测试策略
基本安全防护系统测试
测试点
防火墙
入侵监测系统
漏洞扫描
病毒防治
安全审计
Web信息防篡改系统
安全系统防护体系
安全性测试方法
功能验证
漏洞扫描
模拟攻击试验
侦听技术

软件系统安全性测试列表(Checklist)

- 系统安全性及其测试方法
- 代码安全性检验
- Web安全性测试
- 系统功能安全性测试
- 数据安全性验证
- 网络和通信安全性验证

系统安全性及其测试方法

- 软件系统的安全性
- 系统安全规范与标准
- 源代码评审方法
- 基于风险的安全测试
- 渗透性测试方法
- 模糊测试方法

代码安全性检验

- 程序代码安全性
- C++/Java安全性列表（Checklist）
- JavaScript安全性列表
- 代码安全性扫描工具

Web安全性测试

- 动态跟踪元素属性
- 检测JavaScript事件
- 跨站脚本攻击（XSS）
- 跨站请求伪造攻击（CSRF）
- 拒绝服务攻击（DoS）
- Cookie劫持
- 输入验证
- 浏览器安全问题

Web安全性测试

- 文件上传风险
- Web服务器端安全性
- MS IIS 漏洞检验
- Apache/Tomcat/...漏洞检验
- 内容安全性
- 会话管理
- 截获和修改post请求
- SQL注入及其实例

Web安全性测试

- AJAX安全性测试
- 多系统单点登录机制
- 渗透性Web安全测试
- 使用工具扫描SQL注入漏洞
- 使用Firebug观察实时的请求头
- 使用Webscarab观察实时的post数据
- 使用Tamperdata观察实时的响应头
- 使用curl检验URL重定向攻击
- 使用nikto扫描网站

系统功能安全性测试

- 口令安全性
- 身份验证
- 用户权限
- 非授权攻击
- 访问控制策略
- 操作日志检查
- 配置管理
- 功能失效、异常带来的安全风险

数据安全性验证

- 数据编码验证
- 数据加密和解密
- 系统数据完整性
- 数据管理性
- 数据的独立性
- 数据备份和灾难恢复

网络和通信安全性验证

- 协议一致性验证
- 防火墙
- 入侵检测技术
- 网络拦截
- IPSec/SSL VPN
- PKI/CA
- 网络漏洞检查工具

Checklist还需要细化

- 用户身份认证的安全性测试
  - 明确区分系统中不同用户权限
  - 系统中会不会出现用户冲突
  - 系统会不会因用户的权限的改变造成混乱
  - 用户登陆密码是否是可见、可复制
  - 是否可以通过绝对途径登陆系统（拷贝用户登陆后的链接直接进入系统）
  - 用户退出系统后是否删除了所有鉴权标记，是否可以使用后退键而不通过输入口令进入系统

手机应用软件安全性测试

- 手机应用相对普通的PC应用，有其自身的特点，在测试策略的选择上不能完全照搬PC应用软件的测试策略、方法和内容，而需要根据手机应用软件面临的风险特点采取相应的测试策略
- 手机应用软件分类
- 测试策略
- 测试内容

手机应用软件分类

应用类别	主要应用特点
手机游戏或离线游戏	其特点是游戏或游戏程序下载到手机后，在操作时或运行程序时不需要网路的支持，就可以得到所需的服务。
在线游戏	需要下载一个客户端程序到手机上，但在玩游戏或操作时，需要网络的支持，也就是需要连接到在线服务器上。
娱乐类	如卡拉 OK、网上聊天、寻找朋友、棋牌类、智力类、电子图书等等。
新闻类	如综合新闻和各种专题新闻、股票行情、交通业务信息、音乐影视、影迷/歌迷空间、天气预报等等。
基于定位或位置的服务	包括位置及地图、寻找他人、交通指南、周边服务等，这一类服务有其特殊的定位要求，如娱乐、信息、商务等都有一定的关系。
金融/交易/博彩类	主要包括移动银行、股票交易、彩票、电子商务等。这类服务最大的特点是需要端到端的加密-解密安全机制。也就是手机与服务器间交流的信息都过加密处理，移动运行商只提供透明的通信通道。
企业/商务应用	企业类的应用将会越来越多，重要性越来越大。其主要包括：一是企业的人员是流动或移动的，二是这些流动的人员有与“总部”或别人信息交流的需求，或者需要及时从数据库查找并得到信息，或者需要把信息或数据及时传送到数据库，如公安、税务、监察、环保、海关、保险、运输、维修等行业。还有就是针对特定会员或特定顾客的应用，如汽车俱乐部、健身俱乐部、美容俱乐部、保险公司等。

手机应用软件面临的安全风险

安全风险分类	安全风险内容
故意行为	(1) 非受权存取：即使设备不丢失，局外人会使用盗取的密码进入设备，导致数据被修改或数据丢失。 (2) 电子窃听和修改数据：局外人可能会窃取网络上传输或转换的数据。此外，信息泄漏会使局外人进行数据更改。 (3) 后门和陷门：主要是指用于调试用的入口，如直接存取硬编码的口令。 (4) 逻辑炸弹、木马、病毒。 (5) 病毒和蠕虫。
管理疏忽	管理疏忽，如设备丢失，导致存储在设备内、SIM 卡和存储卡内信息被别人存取，以及通过设备接收的信息如 e-mails 等相关信息。
用户错误	比如删除关键数据或输入错误。
技术故障	导致数据中断、删除和不可存取。
其他	其他不可预期和预防的失效和事件

测试策略

- 验证被测试收集应用软件是否满足预定的安全准则和要求，重点检查软件的防止灾难故障能力、防成败性故障能力和容错能力，对数据的非法访问的保护能力
- 硬件和软件在各种故障模式下的测试，对硬件和软件在降级配置时的处理和保护能力测试
- 各种保护能力测试，包括容错操作能力测试、操作数据安全性保护测试、通讯数据安全性保护测试，对重要数据抗非法访问能力测试，权限管理保护测试
- 多系统、多平台上运行的软件人机接口的安全性测试
- 测试过程中严格执行JAVA安全域的划分，并进行相应的签名或测试
- 注重安装测试的重要性，严格按照安装及卸载的安全性要求，设计测试用例

测试内容

- 安全及卸载的安全性要求
- 手机应用软件权限
- 数据安全性
- 通讯安全性
- 人机接口安全性

安装与卸载测试内容

测试项	测试内容
安装性测试	(1) 应用程序应能安装到安装设备驱动程序上； (2) 能够在安装设备驱动程序上找到应用程序的相应图标； (3) 是否包含数字签名信息； (4) J2ME 应用程序描述至少应包括以下特性：MIDlet-Name、MIDlet-Version、MIDlet-Vendor、MIDlet-Jar-URL、MIDlet-Jar-Size； (5) JAD 文件和 JAR 包中包含的所有托管属性及其值必需是正确的； (6) 应用程序描述规格应包含 MIDP 1.0 和 MIDP 2.0 规格规定的特征和内容； (7) JAD 文件显示的资料内容与应用程序显示的资料内容应一致； (8) 安装路径应能指定； (9) 没有用户的允许，应用程序不能预先设定自动启动；
卸载测试	(1) 卸载是否安全，其安装进去的文件是否全部卸载； (2) 卸载用户使用过程中产生的文件是否有提示； (3) 其修改的配置信息是否复原； (4) 卸载是否影响其他软件的功能； (5) 卸载应该移除所有的文件；

权限测试内容

序号	测试内容
1	互连网访问权限–限制/允许使用手机功能接入互联网
2	信息功能–限制/允许使用手机发送接受信息功能
3	自动启动权限–限制/允许应用程序来注册自动启动应用程序
4	本地连接–限制或使用本地连接
5	媒体录制权限–限制/允许使用手机拍照或录音
6	读取用户数据–限制/允许使用手机读取用户数据
7	写入用户数据–限制/允许使用手机写入用户数据

数据安全性要求

测试项	测试内容
密码显示、存储及长度要求	(1) 当将密码或其他的敏感数据输入到应用程序时，其不会被储存在设备中，同时密码也不会被解码。 (2) 输入的密码将不以明文形式进行显示； (3) 密码，信用卡明细，或其他的敏感数据将不被储存在它们预输入的位置上。 (4) 不同的应用程序的个人身份证或密码长度必需至少在 4-8 个数字长度之间。
敏感数据处理的预期行为	(1) 当应用程序处理信用卡明细，或其他的敏感数据时，不以明文形式将数据写到其它单独的文件或者临时文件中。以防止应用程序异常终止而又没有删除它的临时文件，文件可能遭受入侵者的袭击，然后读取这些数据信息。 (2) 当将敏感数据输入到应用程序时，其不会被储存在设备中。
备份与恢复要求	备份应该加密，恢复数据应考虑恢复过程的异常（通讯中断等），数据恢复后再使用前应该经过校验。
安全提示要求	(1) 应用程序应考虑系统或者虚拟机器产生的用户提示信息或安全警告。 (2) 应用程序不能忽略系统或者虚拟机器产生的用户提示信息或安全警告，更不能在安全警告显示前，利用显示误导信息欺骗用户。应用程序不应该模拟进行安全警告误导用户。
数据删除要求	(1) 应用程序必需明确，数据是否将会被永久性的删除或者能够简单恢复。 (2) 在数据删除之前，应用程序应当通知用户或者应用程序提供一个" "取消" 命令 的操作。 (3) "取消" 命令操作能够按照设计要求实现其功能。
PIN 信息存取要求	(1) 应用程序应当能够处理当不允许应用软件连接到个人信息管理的情况。 (2) 当进行读或写用户信息操作时，应用程序将会向用户发送一个操作错误的提示信息； (3) 错误信息必需声明，读或写操作是不可用的。

数据安全性要求

测试项	测试内容
个人信息的使用要求	(1) 应用程序必需能够正确连接到个人信息管理应用程序，同时保证在没有用户明确许可的前提下不损坏和删除个人信息管理应用程序中的任何内容。 (2) 在没有用户明确许可的前提下不损坏删除个人信息管理应用程序中的任何内容； (3) 应用程序读和写数据正确。 (4) 设备的闪存不被不必要的数据填满。 (5) 如果数据库中重要的数据正要被重写，应及时告知用户。 (6) 应用程序应当有异常保护。
应用环境的改变要求	(1) MIDlets 即使应用环境发生改变也能正常运行； (2) 能合理地处理出现的错误； (3) 在处理异常的同时 MIDlets 也能正常运行； (4) 意外情况下应提示用户。

通讯安全性要求

测试项	测试内容
运行中断要求	在运行其软件过程中，如果有来电、SMS、MMS、EMS、蓝牙、红外等通讯或充电时，是否能暂停程序，优先处理通信，并在处理完毕后能正常恢复软件，继续其原来的功能。
通讯中断要求	(1) 当创立连接时，应用程序能够处理因为网络连接中断，进而告诉用户连接中断的情况； (2) 应能处理通讯延时或中断； (3) 应用程序将保持工作到通讯超时，进而发送给用户一个错误信息指示有连接错误。
网络要求	(1) 应能处理网络异常和及时将异常情况通报用户； (2) 应用程序关闭或网络连接不再使用时应及时关闭/断开。
无线消息要求	(1) MIDlet 能正确发送文字短信。 (2) 如果短信发送出现意外应能准确地提示用户。 (3) 短信格式应符合规定，以便其能正确地被接收和处理。
蓝牙连接要求	(1) 应能有效地发现设备和服务。 (2) 连接在长时间不在使用（/闲置）时应中断。 (3) 多个设备互连式，即使其中一个蓝牙设备消失，其他设备之间也应保持连接。 (4) 如发生中断，应提示用户。

人机接口安全性要求

测试项	测试内容
用户接口菜单要求	返回菜单总保持可用；
用户接口命令要求	命令有优先权顺序；
声音要求	声音的设置不影响应用程序的功能；
GUI要求	应用程序必需利用目标设备适用的全屏尺寸来显示上述内容；
用户异常响应要求	应用程序必需能够处理不可预知的用户操作，例如错误的操作和同时按下多个键。

小结

- 软件安全性测试
- 安全性测试主要方法
- 软件系统的安全性测试举例