第5章
基于缺陷模式的软件测试

张圣林
zhangsl@nankai.edu.cn
http://nkcs.iops.ai/courses/softwaretesting/

第五章 基于缺陷模式的软件测试

- 5.1 基于缺陷模式的软件测试概述
- 5.2 缺陷模式及实例

5.1 基于缺陷模式的软件测试概述

- 缺陷模式
  - 程序中经常发生的错误或缺陷所呈现的语法及语义特征
  - 通常由具有领域程序设计经验的程序员或者测试人员总结出来
  - 不同的编程语言，往往对应于不同的缺陷模式集

5.1 基于缺陷模式的软件测试概述

- 基于缺陷模式的软件测试技术特点
  - 针对性强
    - 如果说某种模式的缺陷是经常发生的，并且在被测软件中是存在的，则面向缺陷的测试可以检测出此类缺陷。
  - 基于缺陷模式的软件测试技术往往能发现其他测试技术难以发现的故障，如小概率、不明显、误操作。
  - 工具自动化程度高以及测试效率高。
  - 缺陷定位准确
    - 对测试所发现的缺陷能够准确定位。

5.2 缺陷模式及实例

- 以缺陷产生后果的严重性高低为评判标准，从程序的源代码形式着眼，对缺陷模式进行分类
  - 故障模式
  - 安全漏洞模式
  - 疑问代码模式
  - 规则模式

5.2.1 故障模式

- 一经产生，会导致系统异常或出错
  - 存储泄露的故障模式
  - 数组越界故障模式
  - 使用未初始化变量故障模式
  - 空指针使用故障模式
  - 非法计算故障模式
  - 死循环结构模式
  - 资源泄漏故障模式
  - 并发故障模式

5.2.1 故障模式

- 1．存储泄漏的故障模式（Memory Leak Fault, MLF）
  - 设在程序的某处申请了大小为M的空间，凡在程序结束时M或者M的一部分没被释放、或者多次释放M或M的一部分都是内存泄漏故障
  - MLF有三种形式
    - 遗漏故障：是指申请的内存没有被释放
    - 不匹配故障：是指申请函数和释放函数不匹配
    - 不相等的释放错误：是指释放的空间和申请的空间大小不一样

5.2.1 故障模式

- 2. 数组越界故障模式（Out of Bounds Array Access Fault, OBAF）
  - 设某数组定义为Array[min, max]，若引用Array[i]且i<min或i>max都是数组越界故障。在C++中，若i<0或i>max是数组越界故障
  - 对程序中任何出现Array[i]的地方，都要判断i的范围
    - 若i是在数组定义的范围内，则是正确的
    - 若i是在数组定义的范围外，则是OBAF
  - 字符串拷贝过程中存在的数组越界故障

5.2.1 故障模式

- 3．使用未初始化变量故障模式（Uninitialized Variable Fault, UVF）
  - 存在一个路径，在该路径上使用前面没有被赋初值的变量是使用未初始化变量故障
- 4．空指针使用故障（NULL Pointer Dereference Fault, NPDF）
  - 引用空指针或给空指针赋值的都是空指针使用故障

5.2.1 故障模式

- 5．非法计算类故障（Illegal Computing Fault, ILCF）
  - 是指计算机不允许的计算
  - 一旦非法计算类故障产生，系统将强行退出
    - 除数为0故障
    - 对数自变量为0或负数故障
    - 根号内为负数的故障

5.2.1 故障模式

- 6． 死循环结构模式 (Dead Loop Fault, DLF)
  - 在控制流图中，对任何一个循环结构
    - for语句中的死循环结构
    - while语句中的死循环结构
    - do-while语句中的死循环结构
    - goto语句中的死循环结构
    - 函数循环调用造成的死循环结构

5.2.1 故障模式

- 7． 资源泄漏故障（Resource Leak Fault, RLF）
  - 当一个资源被打开后，如果并不是在所有的可执行路径上都对其进行了显式的释放操作，则是一个资源泄漏故障

5.2.1 故障模式

- 8. 并发故障模式
  - 程序员对多线程的编码机制、各种同步方法、Java存储器模式和Java虚拟机的工作机制不清楚
  - 由于线程启动的任意性和不确定性使用户无法确定所编写的代码具体何时执行而导致对公共区域错误使用
  - 这类模式主要包括不正确的同步、死锁、多线程应用中方法调用时机或方式不正确、同一变量的双重验证、相互初始化的类和临界区内调用阻塞函数等

5.2.2 安全漏洞模式

- 此类缺陷会给系统留下安全隐患，为攻击该系统开了绿灯
  - 缓冲区溢出
  - 未验证输入
  - 竞争条件
  - 风险操作
  - …

5.2.2 安全漏洞模式

- 缓冲区溢出（buffer overflow）漏洞模式
  - 当程序要在一个缓冲区内存储比该缓冲区的大小还要多的数据时，即会产生缓冲区溢出漏洞
  - 主要类型
    - 数据拷贝造成的缓冲区溢出
    - 格式化字符串造成的缓冲区溢出

5.2.2 安全漏洞模式

- 未验证输入(Tainted Data)
  - 程序从外部获取数据时，这些数据可能含有具有欺骗性或者是不想要的垃圾数据，如果在使用这些数据前不进行合法性检查则将威胁到程序的安全
  - Tainted data可能会导致程序不按原计划执行，也有可能直接或间接地导致缓冲区溢出缺陷
  - 主要类型
    - 使用的数据来自外部的全局变量
    - 使用的数据来自输入函数

5.2.2 安全漏洞模式

- 竞争条件(Race Condition)
  - 如果程序中有两种不同的I/O调用同一文件进行操作，而且这两种调用是通过绝对路径或相对路径引用文件的，那么就易出现Race Condition问题。在两种操作进行的间隙，黑客可能改变文件系统，那么将会导致对两个不同的文件操作而不是同一文件进行操作
  - 这种典型的问题发生在用户拥有不同的权限运行的程序中(例如：setuid程序、数据库和服务器程序等)。

5.2.2 安全漏洞模式

- 风险操作（Risky Operation）
  - 如果不恰当地使用了某些标准库函数，可能会带来安全隐患。甚至在某些情况下，某些函数一经被使用，就可能会带来安全隐患
  - 例如：像rand()和random()这样的随机数生成函数，它们在生成伪随机值的时候表现出来的性能是非常差的，如果用它们来生成默认的口令，这些口令将很容易被攻击者猜测到。

5.2.3 疑问代码模式

- 此类问题未必会造成系统的错误，可能是误操作造成的，或者是由工程师不熟悉开发程序造成的，起到提示作用
  - 性能缺陷：此类缺陷会降低系统的性能
  - 争议代码：让人费解的代码

5.2.3 疑问代码模式

- 低性能模式
  - 该模式导致软件运行效率低下，因此建议采用更高效的代码来完成同样的功能
  - 主要包括使用低效函数/代码、使用多余函数、Java中显式垃圾回收、冗余代码、头文件中定义的静态变量、不必要的文件包含、字符串低效操作和有更简单的运算可以替代等

5.2.4 规则模式

- 软件开发总要遵循一定的规则，某个团队也有一些开发规则，违反这些规则也是不允许的
  - 声明定义
  - 版面书写
  - 分支控制
  - 指针使用
  - 运算处理
  - …

谢谢！