江苏大学学报(自然科学版)
JOURNAL OF JIANGSU UNIVERSITY(Natural Science Edition)
2018 年 1 月 Jan. 2018
第 39 卷 第 1 期 Vol. 39 No. 1
doi: 10.3969/j.issn.1671-7775.2018.01.012

基于变异技术的第三方构件安全性测试系统

陈锦富[1], 葛宏河[1], 蔡赛华[1,2], 陈加梅[1], 詹永照[1]
(1. 江苏大学 计算机科学与通信工程学院, 江苏 镇江 212013; 2. 中国农业大学 信息与电气工程学院, 北京 100083)

摘要: 针对基于变异技术的第三方 COM(component object model)构件安全性异常的自动检测问题,设计实现了一个第三方构件安全性测试原型系统 TCSTS(third-party component security testing system). TCSTS 系统的主要功能模块有构件接口分析模块、参数变异测试模块、条件变异测试模块、状态变异测试模块和安全分析模块. 构件接口分析模块能分析得到被测试构件的接口方法和参数信息;在参数变异测试模块中,采用参数变异测试用例生成算法生成参数变异值集合进行变异测试;在条件变异测试模块中,生成违背前置条件的测试用例并结合后置条件检测条件语句中是否存在安全漏洞;在状态变异测试模块中,基于行为冲突算法和条件冲突算法变异可扩展有穷状态机以生成冲突序列,进而参照变异前序列进行安全性测试,最终通过安全分析模块生成测试报告. 试验结果表明:所设计实现的 TCSTS 原型系统具有自动化程度高、操作简单以及测试效果较好的特点;TCSTS 对异常序列的检测率达到了 20% 以上,表明 TCSTS 能够较好地对构件中状态相关的安全漏洞进行检测.

关键词: 安全性测试; 第三方构件; 参数变异测试; 条件变异测试; 状态变异测试
中图分类号: TP311.5 **文献标志码**: A **文章编号**: 1671-7775(2018)01-0071-07
引文格式: 陈锦富,葛宏河,蔡赛华,等. 基于变异技术的第三方构件安全性测试系统[J]. 江苏大学学报(自然科学版),2018,39(1):71-77.

Design and implementation of third-party component security testing system based on mutation technology

CHEN Jinfu[1], *GE Honghe*[1], *CAI Saihua*[1,2], *CHEN Jiamei*[1], *ZHAN Yongzhao*[1]
(1. School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, Jiangsu 212013, China; 2. College of Information and Electrical Engineering, China Agriculture University, Beijing 100083, China)

Abstract: To solve the automatic detection problem of component security exceptions for the third-party component, a prototype tool of third-party component security testing system(TCSTS) was designed and implemented. The TCSTS could employ three aspects of security testing for third-party component of parameter mutation testing, condition mutation testing and state mutation testing. In parameter mutation testing, the variance value was set as parameter constraint by the test case generation algorithm based on the parameter constraint (TCGPC), and the test cases were generated with different parameter constraint. In condition mutation testing, the test cases were generated to satisfy and violate the pre-condition, and the existence of security vulnerabilities was checked in the condition judgment statement with post-condition. In state mutation testing, executable method sequences of components were transformed into extended finite state machine (EFSM), and operations conflict sequences generated algorithm (OCGA) and conditions conflict sequences generated algorithm (CCGA) were designed to

收稿日期: 2016-09-28
基金项目: 国家自然科学基金资助项目(61202110, 61502205); 江苏省第十三批"六大人才高峰"项目(XYDXXJS-016)
作者简介: 陈锦富(1978—),男,江西信丰人,博士,副教授(jinfuchen@ujs.edu.cn),主要从事软件测试和可信软件研究.
葛宏河(1990—),男,江苏兴化人,硕士研究生(673534482@qq.com),主要从事软件测试研究.

mutate EFSM and generate conflict sequences. The security testing of conflict sequences after mutation was implemented. The results show that the TCSTS has good operational ability and testing ability. The detection rate of abnormal sequence by TCSTS is more than 20%, which indicates that TCSTS can detect the state-related security vulnerabilities in the component security testing.

Key words: security testing; third-party component; parameter mutation test; condition mutation test; state mutation test

构件技术的快速发展使得更多的软件商购买并使用第三方构件产品[1-2]，包括一些对安全性有着极强要求的关键软件（如军事、医疗、银行、铁路及金融等行业软件）. 基于构件的软件工程（component-based software engineering, CBSE）目前已经成为整个软件工程领域的研究热点[3-4]，CBSE 的出现使得软件开发效率有了很大程度上的提高，软件运维的成本也得到了一定程度的降低，但安全性问题还未能被有效解决，故而一直困扰着构件开发者以及使用者. 第三方构件是指除了构件销售商和使用者之外的第三方组织或者个人开发的构件，而第三方构件的设计说明文档和源码不对构件使用者公开，导致构件使用者无法知晓构件的设计原理和架构，因此开发第三方构件存在很大的隐忧[5]. 为了尽可能地降低隐忧程度，需要利用相关技术对构件或者构件系统进行必要测试[6]，然而现如今仍然缺少针对构件安全性的较为有效的测试方法，已存在的一些构件测试方法主要适用于存在详尽需求说明和源代码的构件，对高度独立且说明文档及源码不被使用者知晓的第三方构件则显得无能为力，因此，设计并实现一种有效的针对第三方构件的安全性测试原型系统具有深远的研究意义[7-8].

对软件测试的研究主要从理论、技术、辅助工具和管理这 4 个方面进行，其中，设计和实现一个便捷、高效的软件测试原型系统具有很强的现实和应用意义. COM 构件是由微软公司开发、运行在 Windows 平台下、使用最广泛的一种构件，文中拟针对第三方 COM 构件，基于前面研究的条件和参数变异测试用例生成方法和状态变异[9]测试用例生成方法，讨论如何设计并实现一个第三方构件安全性测试原型系统（third-party component security testing system, TCSTS）. TCSTS 主要从参数变异模块、条件变异模块、状态变异模块 3 个方面对注入多个漏洞的构件实施安全性测试，并进行分析.

1 系统总体模块框架

TCSTS 是在 Windows 7 平台上使用 Visual Studio 2008 开发工具和 C#语言开发的一个针对 COM 构件的安全性测试系统，其主要有以下功能：① 对待测的 COM 构件进行接口分析，得到必要的构件方法以及参数等信息；② 对参数进行变异，并对参数变异后的构件进行变异测试；③ 分析方法的前置及后置条件，生成符合前置条件要求的测试用例，然后对这些测试用例进行条件变异并对测试用例进行变异测试；④ 对待测构件的方法序列进行转换，使其转换成 UML 中的可扩展状态图，然后针对 EFSM 进行变异，从而生成行为冲突序列及条件冲突序列，最后对上述不可达的冲突序列进行执行，并对其进行状态变异测试；⑤ 对经过参数变异、条件变异和状态变异后得到的测试用例进行安全性分析，生成构件安全检测报告. 图 1 为 TCSTS 系统的总体结构图.

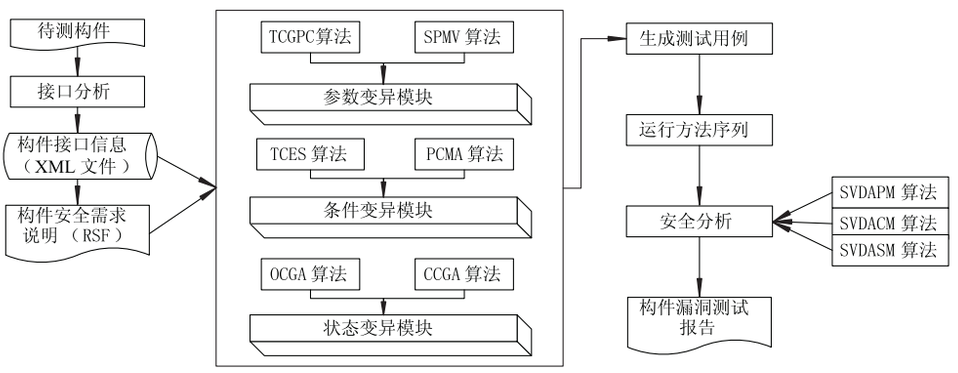

图 1 TCSTS 系统总体结构图

对图 1 中的主要功能模块做如下简单介绍，其中的核心模块在第 2 –4 节中进行详细讨论.

1.1　构件接口分析模块

在 TCSTS 原型系统中，通过分析构件类型信息(TypeInfo)获得与构件接口方法相关的详细信息. TCSTS 原型系统提取 COM 构件或者动态链接库中的类型库文件，类型库文件既可以是单独的二进制文件(. tlb)、动态链接库文件(. dll)，也可以是可执行文件(. olb，. exe)中的资源. 对类型库进行分析得到以下 4 个信息：

1）类型库的名称以及类型库版本信息.

2）类型信息的数目.

3）每个类型信息的接口类型(如 Interface，Dispatch 等)、函数的数目及变量的数目.

4）构件方法名、方法的参数名、参数的数量、参数的类型、返回值的类型以及调用的类型等.

TCSTS 使用 XML 技术将从安全需求说明 RSF 中提取获得的相关信息，如前置及后置条件、参数值约束等，存储到 XML 文档中，为后续变异过程提供方便.

1.2　参数变异模块

参数变异模块的主要功能是依据参数类型和与之有联系的全部变异算子产生变异值集合以生成易触发异常的数据，然后通过组合分析减少测试用例集的数量，根据参数值约束和参数间关系约束将符合需求的所有测试用例挑选出来，接着执行方法序列以及漏洞检测算法，从而检测构件是否含有安全漏洞.

1.3　条件变异模块

条件变异模块实现的主要功能是：一方面根据前置条件生成满足要求的测试用例，然后在方法中代入测试用例并执行，如果在执行过程中方法出现异常或者方法与后置条件相违背，那么说明存在安全漏洞；另一方面，将所有违反前置条件的测试用例代入到方法中加以运行，若方法在运行过程中出现异常或者方法满足后置条件约束，那么此方法中有安全异常的存在.

1.4　状态变异模块

状态变异模块首先考虑可扩展的有穷状态机 EFSM 与构件方法序列之间的联系，通过对构件执行序列和安全需求说明将方法序列进行转换，得到符合要求的 EFSM；然后根据 EFSM 在有穷状态机 FSM 上添加警戒条件以及操作等相关信息时会生成不可达序列这一特征，利用预先定义的条件冲突和行为冲突这 2 个概念设计变异算子和测试用例生成算法，从 EFSM 上生成冲突的不可达序列；最后，通过运行这些冲突序列并采用状态变异漏洞检测算法，检测出第三方构件是否含有安全漏洞.

1.5　安全分析模块

TCSTS 原型系统安全分析模块的主要工作是实现 3 个安全漏洞检测算法，分别是与参数变异相对应的 SVDAPM、与条件变异相对应的 SVDACM 以及与状态变异相对应的 SVDASM 安全漏洞检测算法，然后使用上述 3 个算法检测第三方构件是否含有安全漏洞，同时整理获取构件的安全检测报告.

2　参数变异测试

参数变异测试首先需要生成变异值集合，该集合是根据参数类型并利用与之相关联的全部变异算子生成所得，如果是数值型的参数则选择所有与值约束相符合的值，如果是非数值型参数则选择所有与值约束相违背的值；然后采用组合分析的方法对生成的参数值集合进行处理生成测试用例集，对测试用例集进行遍历操作并挑选出所有与参数约束表达式不相符合的测试用例. 如果在执行过程中有异常方法的出现或者方法与后置条件相违背，那么此方法中有安全漏洞的存在，继而保存异常方法和测试用例等信息. 参数变异测试的框架图如图 2 所示.

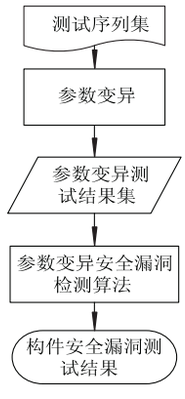

图 2　参数变异测试框架图

对图 2 中的核心模块描述如下：

1）测试用例序列集生成. 测试用例序列集通过基于参数约束的测试用例生成算法(TCGPC)生成，该算法的主要思想是依据方法参数的类型去调用单参数变异值函数(SPMV)以得到基于算子的数据集，然而数据集过于庞大，因此需要借助组合思想实

现生成测试用例，接着根据参数间约束（relCS）将所有无法满足约束的测试用例排除在外.

TCGPC 算法首先对方法中的任意参数都调用 SMPV 方法，并结合所有与参数类型有联系的算子生成值集合. 接着将参数的值约束作为判断的标准对值集合进行挑选，如果参数的类型是数值型的，则挑选所有符合值约束的值，否则选择所有与值约束不相符合的值. 如果方法中仅仅只有 1 个参数，那么返回相应的值集；如果有 2 个参数，则进行组合覆盖；如果参数个数等于或者大于 3，那么进行 3 因素组合覆盖以缩小测试用例集；最后挑选出所有违反参数约束的测试用例.

2）参数变异. 参数变异需要借助 SPMV 函数，该函数的主要思想是借助参数的类型和与此参数有关联的所有算子，生成比较容易产生安全漏洞的测试数据. SPMV 函数将参数分为整型、字符型、字符串型、布尔型、指针型、浮点型、数组型和结构型等类型，不同类型的参数和与之有联系的所有变异算子可以生成容易产生变异的值集合.

3）参数变异安全漏洞检测算法. 遍历所有方法序列中的方法是参数变异安全漏洞检测算法（SVDAPM）的主要思想，当方法序列中的方法存在参数时，则需要采用 TCGPC 算法以生成测试集；然后依次将测试用例代入方法中并执行 SVDAPM 算法进行检测操作运行，如果执行检测操作后得到的结果与预期值存在差异，那么认为此测试用例是有效的；最后将参数变异后的测试用例以及注入参数变异后的方法信息记录到检测报告中.

3 条件变异测试

条件变异测试的目标是测试前置条件中的关系表达式，从而生成 2 类测试用例：一类是满足前置条件的测试用例，另一类是与前置条件相违背的测试用例. 然后根据后置条件，检验条件判断语句中是否含有安全漏洞的隐患. 条件变异测试的框架及框架图见图 3.

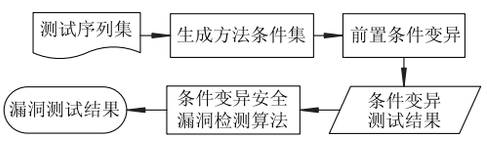

图 3 条件变异测试框架图

对图 3 中的核心模块描述如下：

1）前置条件变异. 前置条件变异过程通过前置条件变异算法（PCMA）实现，PCMA 算法的目标是产生所有导致前置条件为假的表达式. 不妨假设前置条件中子项数量是 m，算法步骤如下：① 依据函数 $RRF(S_0)$ 获得了所有属于第 1 个子项的变异式；② 遍历这些变异式 t，然后通过函数 *Exclusive* 对其进行判断，可知道 2 个变异式之间不存在互斥关系，那么将这 2 个变异式并入到变异后的前置条件 T 中；③ 重复①，②步直到执行到 S_m 为止.

2）条件变异安全漏洞检测算法. 条件变异安全漏洞检测算法（SVDACM）依次对各个方法序列中的方法进行检验，如果方法中存在前置条件，则采用基于约束方程组的测试用例生成算法（TCES）生成合法的测试数据，如果这些生成的测试数据中有异常出现或者与后置条件不相符合的情况，那么则表明构件中含有安全漏洞. 此外，调用 PCMA 算法对前置条件进行变异操作，并生成所有与前置条件相违背的测试用例，如果运行这些生成的测试用例，产生的结果没有异常且运行的结果与预期有差异，那么方法中有漏洞的存在；如果方法中没有前置条件的存在，则通过边界值方法和随机值方法获取测试用例，然后通过结合后置条件判断方法判断是否含有安全漏洞.

4 状态变异测试

状态变异测试是基于变异技术的第三方构件安全性测试原型系统中一个极其重要的模块，状态变异测试的框架图见图 4.

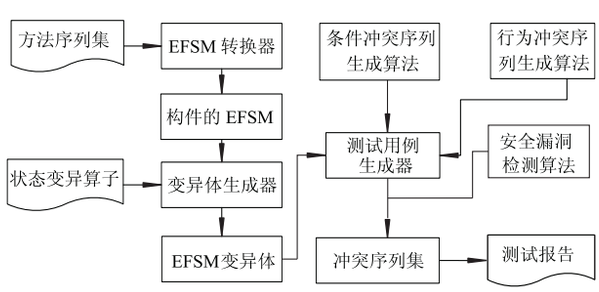

图 4 状态变异测试框架图

状态变异测试模块中核心模块功能说明如下：

1）EFSM 转换器. 方法的集合可以被看作构件中的方法执行后生成的方法序列集，而变异测试中的 EFSM 可以被看作是状态的集合，EFSM 中状态间的相互转换与方法的执行次序相类似，关系表达式可以对方法中的前置条件、后置条件以及在 EFSM 中的警戒条件全部进行表示. 综上所述，对 EF-

SM 转换器的功能可做如下概括：通过对构件中的方法序列以及 EFSM 进行关联性分析，将构件中的方法执行序列转化成与之对应的 EFSM.

2）变异体生成器. 当 EFSM 转换器将构件的方法执行序列转换为 EFSM 之后，变异体生成器开始执行，它的作用是设计一些将执行序列变为不可达的冲突序列的变异算子，通过这些变异算子对 EFSM 进行变异. 这些变异算子主要包括：添加新的迁移状态和改变现有的迁移状态.

3）行为冲突序列生成算法. 行为冲突序列生成算法（OCGA）的主要功能是对变异的 EFSM 进行遍历操作，从中挑选出符合行为冲突特征的不可达序列. OCGA 算法步骤如下：① 先依据 EFSM 拓扑序列图的开始状态，从而依次得到状态图的拓扑序列；② 需要把拓扑序列转化为迁移序列 T，T 满足所有最后得到的冲突序列从起始的方法开始. 如果 EFSM 中不存在环结构，那么直接得到其拓扑结构；如果存在环结构，那么依据拓扑序列的求解步骤对其进行求解.

4）条件冲突序列生成算法. 条件冲突序列集 W 是由条件冲突序列生成算法（CCGA）借助变异的 EFSM 图生成，这也是 CCGA 算法的主要功能. CCGA 算法的处理过程与 OCGA 算法的处理过程相类似，主要区别是 CCGA 算法需要判断迁移条件 t_i 和后续迁移 t_k 中的警戒条件 y 值域的交集是否为空集，如果交集是空集，那么此交集是条件冲突序列.

5）测试用例生成器. 测试用例生成器的作用是采用 CCGA 和 OCGA 对变异后的 EFSM 进行遍历，为了保证获得的构件执行路径能从起始方法为头节点，需要得到 EFSM 的迁移拓扑序列，并据此判断当前迁移和后续迁移中是否包含冲突迁移，然后在当前运行的迁移路径中将 EFSM 拓扑序列的起始方法保存.

6）安全漏洞检测算法. 状态变异安全漏洞检测算法（SVDASM）的执行顺序如下：首先执行 OCGA 算法和 CCGA 算法，获得行为冲突和条件冲突 2 类序列集；然后合并行为冲突和条件冲突这 2 类序列集，并遍历集合中的所有集合；接着构造序列中满足条件的约束方程组，并根据此方程组求得方法输入参数的取值；最后依次在方法中代入所取得的参数值，并对冲突序列进行执行操作，若冲突序列中的每个方法都被执行，那么证明构件中包含有安全漏洞.

5　系统实现及测试环境

5.1　系统实现

为了实现 TCSTS 原型系统的功能，系统界面使用了窗口浮动技术，并将主界面切分成多个不同的版块，每个版块不断增加相应的控制条以符合各个需求. 窗口浮动技术能够将一个窗体内的控件任意拖动到需求的位置，该技术的实现主要包括了 DockPanel 类、DockWindow 类、DockPane 类和 DockContent 类 4 个类，Dockpanel 类从 System. Windows. Forms. Panel 类继承而来，DockPanel 类能自动管理附加的窗体.

TCSTS 界面设计的主要步骤如下：① 创建一个多文档窗体；② 设置 DockPanel 的 DocumentStyle：DockPanel. DocumentStyle = DocumentStyle. DockingMdi；③ 实现窗口停靠技术，使得界面能够在任意位置进行停靠.

5.2　测试环境

TCSTS 原型系统是在 Visual Studio .NET 2008 平台上，采用 C#程序设计语言开发而成. TCSTS 可以对构件进行参数、条件和状态 3 个方面的变异操作，然后分别调用前面研究的 3 个安全漏洞检测算法检测构件是否有安全漏洞的存在. 在 TCSTS 原型系统中，生成的测试用例被自动保存在本地文件中，变异后的测试用例同样被保存在本地文件中，最后依据前面研究的检测算法对构件进行安全检测，完整的检测过程如图 5 所示.

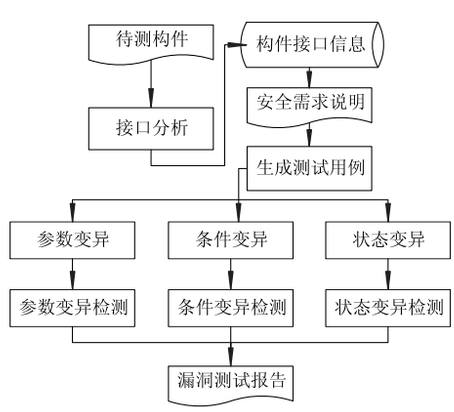

图 5　TCSTS 测试过程

TCSTS 原型系统首先对待测构件进行接口分析获得构件的接口信息；然后从构件的描述和 IDL 等信息中获得构件的安全需求规约；接着对测试用例执行变异操作，并调用对应的检测算法对构件进行变异检

测;最后将结果整理成构件安全漏洞检测报告. 图6是TCSTS原型系统在正常操作下的系统环境快照.

图6 TCSTS系统快照

6 系统测试结果分析

为了验证TCSTS原型系统的效果,对其进行多次试验验证,测试试验分别从参数变异测试、条件变异测试和状态变异测试3个方面展开.

6.1 参数变异测试

为了验证参数变异测试的有效性,对TestParam. dll构件进行试验验证,构件信息如下:构件名,TestParam. dll;方法个数,7;代码行数,85;植入错误个数,7. 该构件为笔者根据试验环境开发设计的构件. 图7比较了采用参数变异方法、随机方法FUZZ和边界值方法生成测试用例有效性.

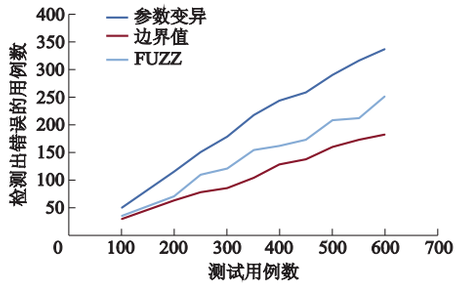

图7 参数变异检测结果

由图7可见,有效的测试用例数量随着生成的测试用例数量的增加而不断增加;3种方法中,采用边界值方法对构件进行检测得到的检测率最低,采用参数变异方法的检测率最高;另外,参数变异方法的优势也随着测试用例个数不断增加而越来越明显.

6.2 条件变异测试

为了验证条件变异测试的有效性,对TestCondiDll. dll构件进行了试验验证,构件信息如下:构件名,TestCondiDll. dll;方法个数,6;代码行数,63;植入错误个数,6. 该构件为笔者根据试验环境开发设计的构件.

为了得到不同的方法所产生的用例个数以及其覆盖的错误数2类数据,需要对构件TestCondiDll中的GetLargest方法采用不同的方法进行检测,表1分别是由条件变异法、判定覆盖法、条件覆盖法以及条件组合覆盖法4种方法检测而得的数据.

表1 构件TestCondiDll. dll的检测结果

测试方法	测试用例数	覆盖的错误数
条件变异	27	1
判定覆盖	2	0~1
条件覆盖	2	0~1
条件组合覆盖	8	0~1

由表1可见,判定覆盖法、条件覆盖法以及条件组合覆盖法无法检测到确切的错误个数,并且这3种测试方法产生的测试用例均被条件变异法产生的测试用例所包含. 根据条件变异法所产生的测试用例数目,可知该方法能产生全部可能的测试用例,通过检测可知,这些错误均是由*RRF*算子所引起.

6.3 状态变异测试

为了验证状态变异测试的有效性,对5个构件分别进行试验验证,构件信息如表2所示. 这些构件部分来源于开源网站,部分为笔者根据试验环境开发设计的构件.

表2 5个构件的信息

构件名	方法数	植入错误个数/个
BankTransaction. dll	6	3
Examine. dll	6	2
Calculator. dll	9	4
Order. dll	8	3
BeverageVending. dll	7	3

表3给出了对BankTransaction. dll, Examine. dll, Calculator. dll, Order. dll以及BeverageVending. dll这5个被测试构件进行状态变异测试后的结果,测试试验主要从表3中列出的4个方面入手,其中序列检测率是指有效的序列数目占冲突序列总数的百分比.

表3 5个构件的检测结果信息

构件名	行为冲突序列数	条件冲突序列数	有效的序列数	序列检测率/%
BankTransaction. dll	14	6	4	20.0
Examine. dll	9	6	4	26.7
Calculator. dll	23	20	11	25.6
Order. dll	20	12	8	25.0
BeverageVending. dll	19	6	7	28.0

由表 3 数据可知，产生冲突序列的数目与构件中方法的数目成正比；由于序列检测率维持在 20% 及以上，因此状态变异测试方法能比较好地对构件中的安全漏洞进行检测.

7　结　论

实现的 TCSTS 原型系统主要有以下几个优点：

1）强大的变异功能. TCSTS 原型系统能分别从构件的参数、条件和状态这 3 个方面进行变异操作，可以有效地触发第三方构件的安全漏洞.

2）强大的变异测试功能. TCSTS 原型系统能够分别对构件进行参数变异测试、条件变异测试和状态变异测试.

3）测试原型系统的自动化程度比较高，大部分操作由系统自动完成，仅有少许操作需要人工参与.

4）测试工程通过对测试项目的有效管理，实现了对部分资源的重新利用，通过重现测试活动的方式更好地对测试结果进行了对比分析.

到目前为止，文中在第三方构件安全性测试方面做了一些探索工作，实现的 TCSTS 原型系统仍然存在以下几点不足：

1）TCSTS 原型系统仍然没有实现全自动化，不能自动存取某些相关信息，如参数值约束、条件约束等信息.

2）TCSTS 原型系统中测试的试验对象主要是 COM 构件，在今后需要对系统进行改进以取得更强的适应性.

3）TCSTS 原型系统没有能够提供完整的构件安全等级评估机制.

参考文献(References)

[1] CHEN J F, LU Y S, WANG H H. Component security testing approach based on extended chemical abstract machine[J]. International Journal of Software Engineering and Knowledge Engineering, 2012, 22(1): 59 – 83.

[2] SCANDARIATO R, WALDEN J, HOVSEPYAN A, et al. Predicting vulnerable software components via text mining[J]. IEEE Transactions on Software Engineering, 2014, 40(10):993 – 1006.

[3] CHEN J F, CAI S H, ZHU L L, et al. An improved string-searching algorithm and its application in component security testing[J]. Tsinghua Science & Technology, 2016, 21(3):281 – 294.

[4] NAGAPPAN M, MIRAKHORLI M. Big(ger) data in software engineering[C] // Proceedings of the 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. Piscataway: IEEE Computer Society, 2015: 957 – 958.

[5] GHO S, NA J C, PARK K, et al. A fast algorithm for order-preserving pattern matching[J]. Information Processing Letters, 2015, 115(2):397 – 402.

[6] FAN Y J, YE Y F, CHEN L F. Malicious sequential pattern mining for automatic malware detection[J]. Expert Systems with Applications, 2016, 52:16 – 25.

[7] TSENG V S, WU C W, FOURNIER-VIGER P, et al. Efficient algorithms for mining the concise and lossless representation of high utility itemsets[J]. IEEE Transactions on Knowledge & Data Engineering, 2015, 27(3):726 – 739.

[8] 覃志东，侯颖，肖芳雄. 基于蚁群优化算法的同构多核任务分配与调度[J]. 江苏大学学报(自然科学版), 2014, 35(6):679 – 684.
QIN Z D, HOU Y, XIAO F X. Task allocation and scheduling for homogeneous multi-core processors based on ant colony optimization[J]. Journal of Jiangsu University (Natural Science Edition), 2014, 35(6):679 – 684. (in Chinese)

[9] CHEN J F, CHEN J M, HUANG R B, et al. An approach of security testing for third-party component based on state mutation[J]. Security & Communication Networks, 2016, 9(15): 2827 – 2842.

(责任编辑　梁家峰)