基于形式化单子的第三方构件安全性测试模型及其应用

陈锦富 赵小磊 刘一松 黄如兵 蔡赛华 郭昱池

(江苏大学计算机科学与通信工程学院 江苏 镇江 212013)

摘 要 因第三方构件通常由不同的组织开发完成,相应的需求开发文档及源代码无法获取,传统的软件安全测试方法无法应用.通过分析构件接口信息,借鉴软件数据挖掘技术可以获取构件相应的接口方法运行序列及需求规约等信息,进而也能对测试日志信息进行分析和处理.针对第三方构件安全性难以测试的问题,本文提出了一种采用数据挖掘技术的测试模型及其测试框架.提出的方法首先形式化定义了测试模型的基本元素,然后基于此模型设计了一个测试框架,并采用单子技术对测试流程和数据挖掘算法进行了形式化描述.为了实现该测试框架,论文实现了数据挖掘相关算法并研发了一个测试第三方构件安全性的测试系统(Component Security Testing System based on Data Mining,CSTS-DM),最后对测试模型及 CSTS-DM 进行了案例分析及实验验证,并通过对商业构件和模拟构件的分析测试,验证了 CSTS-DM 原型测试系统的效果和效率,也分析了提出的测试框架及测试模型的有效性及可行性.

关键词 构件软件;安全性测试;测试模型;接口方法;数据挖掘算法

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2019.01626

Security Testing Model of Third-Party Component Based on Formal Monad and Its Application

CHEN Jin-Fu ZHAO Xiao-Lei LIU Yi-Song HUANG Ru-Bing CAI Sai-Hua GUO Yu-Chi

(*School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, Jiangsu* 212013)

Abstract The third-party components have been widely used in software system since the third-party components usually are developed by different organizers. Component-based software engineering (CBSE) has a rapid development with software engineering technology, which enhances the development efficiency for different software. As a result, the time of software development is reduced, and the cost of software maintenance is also cut down to some extent. At current, CBSE is an important development approach of software system in real industry. With the popularization of the third-party components, the research on security testing for third-party components is very important. Component security vulnerabilities mean the flaws in the aspects of component security including all the factors that threatening and destroying component security. Component security vulnerabilities usually include explicit and implicit vulnerabilities. Explicit vulnerabilities are commonly caused by memory leak or buffer overflow; while implicit vulnerabilities are usually caused by violating security requirement specification. Both explicit vulnerabilities and implicit vulnerabilities are very difficult to be detected by traditional approaches, which lack effective detection model and detection means. Due to some components come from third-party providers and their source codes and detailed development documentation cannot be obtained, it is

收稿日期:2017-10-30;在线出版日期:2018-05-14.本课题得到国家自然科学基金(61202110,61502205)和中国博士后科学基金(2015M571687,2015M581739)资助.**陈锦富**,博士,教授,中国计算机学会(CCF)会员,主要研究方向为软件测试、可信软件.E-mail:jinfuchen@ujs.edu.cn.**赵小磊**,硕士研究生,主要研究方向为软件测试.**刘一松**,博士,教授,主要研究领域为计算机图形学、人机交互及软件工程.**黄如兵**,博士,副教授,主要研究方向为软件测试.**蔡赛华**,硕士研究生,主要研究方向为软件测试.**郭昱池**,硕士研究生,主要研究方向为软件测试.

difficult to ensure their security through traditional testing methods. With the technology of data mining, the component methods' test sequence and component requirements may be obtained through processing large-scale test data. Based on data mining technology, we presented the testing framework and testing model for the component security. (1) We proposed a testing model for component security on the basis of data mining technique. Firstly, the major model elements are created, the model elements include component specification, component testing log, component method sequence, data mining algorithm, security testing algorithm, security detecting result and vulnerability rules. Then the monad technique is proposed to formally describe the testing process. (2) In order to guide component security testing, we proposed a security testing framework based on data mining algorithms and security testing model. The testing framework analyzes component interface information and method parameters information. After further mining the component specification, the available component methods are obtained. On the basis of collecting available method name, pre-conditions and post-conditions, the method testing sequences can be generated. In order to test component, the framework can produce the executable test suites, and the monitoring log can be generated by monitoring the running process. Furthermore, the component security exceptions may be detected in the running process. (3) Based on the proposed testing framework, we designed and implemented a component security testing system based on data mining technology (CSTS-DM). In order to validate and test the effectiveness of the proposed testing model and CSTS-DM, we conducted the case analysis and empirical study based on some real-life components including commercial components and self-development components. The case analysis results and experimental results show the proposed model and framework are operable and effective.

Keywords　component software; security testing; testing model; interface method; data mining algorithm

1　引　言

近年来随着物联网技术的进一步发展，软件越来越多被普及和应用. 为了加快软件的开发效率，构件技术被广泛使用. 各种构件开发技术和开发环境也得到了迅速发展，极大地提高了构件软件的开发效果和效率，且开发和维护的成本也有所降低[1-3]. 构件是现今软件系统中的基础软件模块，其质量好坏将影响整个软件系统的质量优劣，低劣的构件将使整个软件系统无法正常运行，导致整个系统受到严重影响. 为保障构件安全性与稳定性，对构件进行充分的验证和测试是非常重要的. 目前构件功能性测试方法相对较多，但相应的构件可靠性和安全性测试方法较少. 特别是针对第三方构件，因无法得到构件的设计开发文档和源代码，传统的软件测试方法不能有效用于测试构件的安全性[4-5]. 而第三方构件的安全性决定了以第三方构件为基础的构件软件系统的安全性，因此研究第三方构件的安全性测试并以此建立一种安全性测试模型具有非常重要的意义.

当前，国内外有一些学者研究构件安全性测试. Ma 等人[6]形式化描述了构件耦合测试准则，并构造了定义使用表，通过把这些信息封装成元数据提供给测试者进行构件系统的组装测试. Li 等人[7]则提出以接口自动机作为模型来描述构件行为，并提出接口方法序列产生算法，进而对接口方法进行测试和分析，最终校验构件行为功能是否符合需求. 此外，一种基于元数据和扩展的构件接口规约模型也被 Naseer 等人提出[8]，进一步通过测试用例产生算法对构件进行测试. 上述提出的方法需要构件提供详细的规格说明和接口描述，无法应用于无源代码的第三方构件安全性测试.

Winter 等人[9]基于软件故障植入技术对构件进行健壮性评估，论述了几个用来评价构件故障模型健壮性漏洞检测能力的指标. 本课题组也提出过

基于错误注入的构件安全性测试方法及测试用例生成技术[10-11].错误注入测试方法对检测构件显式安全漏洞有一些效果,却对于大规模的构件及构件系统测试效率不甚理想.通过研究状态机模型,Lei 等人[12]研究了一种适用于构件可靠性的测试技术,但这种技术只适用于非第三方构件安全性测试.

Khan 等人[13]提出了一个针对用户数据的安全保护模型,并给出了相应的安全性描述框架.伍建焜[14]基于抽象状态机理论,提出了一种网构软件系统的形式化模型.还有学者研究制定有效的构件安全模型如访问控制模型[15-16]、基于评估的可信实体模型[17-18].现有的这些方法由于理论性强且难以实现自动化,故没有很好的应用于工业实践.

当前,软件工程研究者们已较好的使用数据挖掘技术进行领域研究和实践.一般过程是先对相关数据进行预处理,进而选用一些相关的数据挖掘算法对处理后的数据进行分析,生成相应的序列模式、频繁数据及模式规则等对软件工程活动有用的信息[19-20].基于改进的一些挖掘方法[21-24],对构件中系统缺省方法和冗余无效数据进行筛选,进而获取构件中的有效接口序列方法和相应的需求说明等测试数据,以此对大量的复杂测试数据做详细的分析,来检验被测试的构件是否达到安全性和可靠性的要求.

本文研究并提出了采用数据挖掘技术的适用于第三方构件的一种安全性测试模型,用来解决构件的安全性测试方面的问题.该模型的基本元素采用形式化方法进行描述.并且,基于此模型设计了一个测试框架,并用单子技术对测试流程进行了描述.单子是一种软件分析与验证的形式化手段,论文中采用单子技术对构件测试流程进行了形式化描述.每个单子转换器被设计成有意义的一种计算.论文把构件测试作为一类计算,由相应的单子转换器来统一描述,能够隐藏构件测试时所需要的基本信息,加强形式化描述的模块性和严谨性.

在提出的模型中,首先分析第三方构件的接口信息,获取构件方法等参数;接着基于构件方法参数挖掘相关的需求规约,生成可行的接口方法和方法用例测试集;从而产生构件测试执行脚本,并且对产生的测试用例集和测试脚本进行执行,产生测试监测日志集;然后,对监测日志集进行分析,得出被测构件的安全检测报告.最后,我们基于实验及案例分析,进一步验证了构件安全测试模型的有效性及效果.

2 研究背景

本文主要研究构件安全测试模型和框架,该研究内容是前期通过数据挖掘方法检测第三方构件安全漏洞项目的部分研究内容.该课题有四块主要研究内容:(1)研究采用数据挖掘技术对构件进行安全测试的相关理论模型;(2)研究第三方构件中方法序列测试数据生成算法;(3)研究安全漏洞检测中的变异测试框架及相应算法;(4)研究基于数据挖掘的构件监测日志异常行为检测方法.下面对项目主要的研究内容和方法进行阐述.

项目首先要研究出用于指导构件安全测试的理论测试模型,基于数据挖掘技术形成相关的安全检测规则、测试用例生成算法及分析识别规则.同时基于构件接口信息研究安全需求挖掘技术,并且利用挖掘方法来产生可行的接口方法规约.进一步研究适用于安全性测试方面的挖掘算法,用于生成有效的测试数据集.此外,项目将基于变异技术研究构件方法条件及参数等变异方法,帮助生成非常规的变异测试方法序列,进一步基于漏洞测试算法检测构件接口安全漏洞.最后,研究采用数据挖掘中相关数据特征提取及分类技术,基于监测数据集对构件中异常关联规则和方法序列进行分析,生成构件安全性测试数据.进而利用相关的测试用例生成算法及漏洞检测算法,对构件进行安全性测试.

3 单子和单子转换器

单子(monad)在软件形式化语言中表示特别的数据计算结构类型,主要用于表示抽象计算,而非数据本身,也能用于表示计算过程中有序运算操作.单子主要特征是能在计算时隐藏相关基本数据信息,但保持让外部访问这些数据信息的接口.以下是单子的基本定义[25-28].

定义 1. 单子(Monad)表示为三元组(p,*revert*,*combine*),含构建类型元素 p 及接口函数 *revert* 和 *combine*:

$revert :: m \to p\ m$

$combine :: p\ m \to (m \to p\ n) \to p\ n$

接口函数 *revert* 和 *combine*:满足以下规律:

左幺元:$(revert\ m)$ '*combine*' $k = k\ m$,

右幺元:p '*combine*' $revert = p$,

结合律:p '*combine*' $\lambda m.(k\ m$ '*combine*' $l) =$

$(p$ ‘combine’ $k)$ ‘combine’ l.

其中，$f::M\rightarrow N$ 代表在输入为 M 类型数据时操作函数 f 的输出是 N 类型的数据. 若给出数据类型 m，则类型 $p\ m$ 表示输出类型为 m 的计算操作，单子 p 预先描述的副作用也可能在该计算过程中出现. *revert* 函数是将数据类型 m 提升为计算 $p\ m$ 函数操作，*combine* 函数则提升到 p 之上，且把 $f::m\rightarrow p\ n$ 函数运用到计算 $p\ m$ 中. x ‘combine’ f 表示计算 x 先执行，若计算值类型为 m，即将它传递于 f 函数，产生最后的值类型 $p\ n$.

较早提出单子构造器观点的是 Moggi，主要思想是把额外的计算信息给每个单子，即为单子转换器. 不同的计算能被每类单子转换器表示，已有学者提出了不同类型的单子转换器[25]. 如表 1 给出了常见的单子转换器.

表 1　常用的 4 个单子转换器

具体类型	定义及说明
Circumstance:	**type** CirT $r\ p\ m$ $=r\rightarrow p\ m$ *revert* m $=\lambda r.revert\ m$ p ‘combine’ k $=\lambda r.(p\ r$ ‘combine’ $\lambda m.k\ m\ r)$ *raise* p $=\lambda r.p$ ‘combine’ *revert* *rdCir* $=\lambda r.revert\ r$ *inCir* $r\ p$ $=\lambda\_.p\ r$
Condition:	**type** ConditionT $s\ p\ m=s\rightarrow p(s,m)$ *revert* m $=\lambda s.revert\ (s,m)$ p ‘combine’ $k=\lambda s.(p\ s$ ‘combine’ $\lambda(s',m).k\ m\ s')$ *raise* p $=\lambda s.(p$ ‘combine’ $\lambda m.revert\ (s,m))$ *update* f $=\lambda s.revert\ (f\ s,s)$
Fault:	**data** Fau m $=$Ok m | Fault String **type** FaultT $p\ m=p$(Fault m) *revert* m $=revert$ (Ok m) p ‘combine’ k $=p$ ‘combine’ $\lambda m.$(**case** m **of** Ok $y\rightarrow k\ y$ Fault $s\rightarrow revert$ (Fault s)) *raise* p $=p$ ‘combine’ $\lambda m.revert$(Ok m) *Fau* s $=revert$ (Fault s)
InOut:	**type** InOutT $p\ m$ $=$String$\rightarrow p(m,$String$)$ *revert* m $=\lambda s.revert(m,s)$ p ‘combine’ k $=\lambda s.(p\ s$ ‘combine’ $\lambda(m,s').k\ m\ s')$ *raise* p $=\lambda s.(p$ ‘combine’ $\lambda m.revert(m,s))$ *getval* $=\lambda s.revert(s,s)$ *putval* s $=\lambda\_.revert(s,())$

单子转换器包含两个特点：第一，单子转换器为现有单子增加了新的操作和特性；第二，单子转换器可以进行互相结合. 一些主要概念如下[25-27].

定义 2.　单子转换器可表示为：$\{t,raise\}$，其中 t 是类型构造器，*raise* 是提升函数. t 可将存在的单子 $(p,revert_p,combine_p)$ 转换至新单子 $(t\ p,revert_{tp},combine_{tp})$，提升函数 *raise*：

$raise::p\ m\rightarrow t\ p\ m$

并且符合：

$raise\ revert_p=revert_{tp}$

$raise(p$ ‘$combine_p$’ $k)=raise\ p$ ‘$combine_{tp}$’ $(raise\ k)$.

4　测试模型定义及测试框架

4.1　测试模型的定义

本文提出一种如图 1 所示的面向数据挖掘技术的第三方构件安全测试模型 DMTM. 该模型将构件的测试需求和由构件产生的监测日志信息转化为可以被数据挖掘技术处理的数据集，同时设计一种面向数据集的频繁模式挖掘算法，最终生成由测试方法、测试规则、测试算法和测试结果组成的集合项，从而分析出构件脆弱性情况.

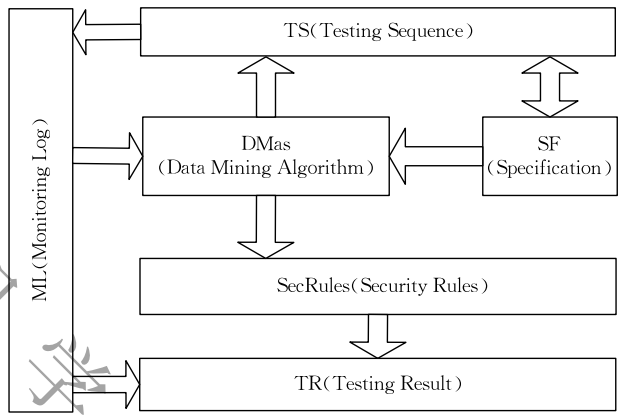

图 1　DMTM 模型结构图

定义 3.　第三方构件安全性测试模型(DMTM)表示为：DMTM$=\langle SF,ML,DMas,TS,TR,SecRules\rangle$，其中 SF 是构件的需求规约集合，ML 是构件生成的监测日志数据集合；$DMas$ 是在对第三方构件进行安全性测试时所运用的数据挖掘算法集；TS 是预先定义的第三方构件运行产生的安全序列集；TR 是面向第三方构件进行安全性测试得到的结果集；$SecRules$ 是第三方构件安全性测试中采用的安全漏洞检测算法以及安全规则集合.

基于构件需求说明、构件的接口方法及接口定义语言，利用相应的数据挖掘算法有效地挖掘需求规约集合 SF(定义 4).

定义 4.　$SF=\langle Meth,Para,PreA,PostA\rangle$，其中 $Meth$ 是包含在第三方构件内所有方法的集；$Para$ 是各方法中的参数集合；$PreA$ 是某个方法的前置条件集合；$PostA$ 是某个方法的后置条件集.

需求规约集合 SF 中包含的每个子元素如定义 5 所示.

定义 5. $Meth=\{m_1,m_2,\cdots,m_i,\cdots,m_n\}$，其中 m_i 代表接口方法中一个实际的方法.

定义 6. $Para=\{pa_1,pa_2,\cdots,pa_i,\cdots,pa_n\}$，$pa_i$ 代表 m_i 方法中具体参数，$pa_i=(DataTy_1,DataTy_2,\cdots,DataTy_i,\cdots,DataTy_k)$，$DataTy_i$ 是指某个具体参数的数据类型.

定义 7. 对于第三方构件方法集合中的任意方法 m_i，它通过接受输入的相关参数 pa_i 的方式获取该方法执行后产生的返回值信息($m_i(pa_i)$).

定义 8. $PreA=\{preA_1,preA_2,\cdots,preA_i,\cdots,preA_n\}$，其中 $preA_i$ 代表的含义为在执行 m_i 前所需遵守的约束条件，则组成约束条件的主要因素为逻辑运算符和关系表达式.

定义 9. $PostA=\{postA_1,postA_2,\cdots,postA_i,\cdots,postA_n\}$，其中 $postA_i$ 是一个条件约束集，它对在执行完方法 m_i 后可执行(调用)的下一个方法进行约束，方法 m_i 可以调用的后续方法记为 $m_i.postA_i$. 同样地，构成这些约束条件的因素主要是关系表达式和逻辑运算符.

ML 主要对第三方构件中各方法的执行情况进行详细的记录，间接地对构件各方法间的调用情况监测，*ML* 可以形式化地用定义 10 所示的七元组进行表示.

定义 10. $ML=\langle TI,Meth,CC,FN,Para,RV,EI\rangle$，其中 *TI* 是方法的执行时间；*Meth* 是方法名；*CC* 是方法运行过程中的约束条件；*FN* 是方法下一步调用的函数名；*Para* 是方法中函数的参数集；*RV* 是构件运行后的返回值信息；*EI* 是异常信息. $ML=\{ML_1,ML_2,ML_3,\cdots,ML_n\}$，$ML_i$ 是监测日志集合中的一条日志信息，$ML_i=(testTime,m_j,preA_j,m_j.postA_j,p_j,m_j(p_j),m_j.fau)$，其中 $m_j(m_j\in Meth)$ 是构件中当前执行的方法名.

监测日志对第三方构件中各个方法的运行情况进行详细的记录，然后利用设计实现的构件安全漏洞检测算法从监测日志中分析构件可能出现的安全问题，所以，监测日志的获取是第三方构件安全检测的基础.

利用数据挖掘等技术排除构件中缺省的系统方法以及冗余、无效的数据内容，获取与构件相关的需求规约和接口方法执行序列等信息；采用数据挖掘算法挖掘由第三方构件产生的大规模监测日志集，获得相关的构件状态信息与安全测试信息. *DMas* 由两个设计实现的面向第三方构件的需求说明、接口定义语言、构件需求规约以及测试日志的频繁模式挖掘和序列模式挖掘算法组成，如定义 11 所示.

定义 11. $DMas=\langle FrequPattMine,SequPattMine\rangle$，其中 *FrequPattMine* 代表面向构件监测日志的频繁模式挖掘算法，主要面向接口方法，最终挖掘得到有效的方法 $Meth'$；*SequPattMine* 代表面向构件监测日志的序列模式挖掘算法，主要面向方法的执行序列，最终挖掘得到序列模式 *LS*.

为了更全面、更充分地对定义 11 进行解释，下面将 *FrequPattMine* 算法和 *SequPattMine* 算法进行具体的展示，如算法 1 和算法 2 所示.

算法 1. *FrequPattMine*.

输入：*Meth*，*SF*

输出：$Meth'$

```
1.  Data=⟨Meth,SF⟩
2.  L1←(one-item frequent sets∈Data)
3.  for (k=2;Lk-1!=null;k++) do
4.     Ck=gen_can(Lk-1)
5.     for each set s∈Data do
6.        Cs=subset(Ck,s)
7.        for each candidate c∈Cs do
8.           c.count++
9.        end for
10.    end for
11.    Lk={c∈Ck|c.count≥min_sup}
12.    if (Lk.count==1) then
13.       Meth'=Lk
14.    end if
15. end for
16. return Meth'
```

设 n 为集合的数量，k 为最大集合的长度，频繁项集挖掘算法 *FrequPattMine* 对集合进行第一次扫描需要的时间是 $O(n\times k)$. 在每次扫描中连接步骤需要 $O(|L_k|\times|L_k|)$，此外，对所有候选频繁项集进行统计需要 $O(n\times|C_k|)$. 因此在最坏情况下，*FrequPattMine* 算法时间复杂度为 $O\left(n\times k+\sum_{k\geq 2}(|L_k|\times|L_k|+n\times|C_k|+|C_k|)\right)$.

算法 2. *SequPattMine*.

输入：*Sequence*，*ML*

输出：*LS*

```
1.  MS←(sequence∈ML)
2.  Data=⟨Meth,SF⟩
3.  L1←(one-item frequent sets∈Data)
4.  int n=0
5.  while (Lk-1!=null)
6.     Ck=gen_can(Lk-1)
```

```
7.    for each method sequence ms∈Data do
8.      C_ms=subMethodSequence(C_k,ms)
9.      for each candidates tc∈C_ms do
10.       tc.count++
11.     end for
12.   end for
13.   L_k={tc∈C_k|tc.count≥min_sup}
14.   if (L_k.count==0) then
15.     n=k-1
16.   end if
17. end while
18. LS=L_n
19. return LS
```

面向构件监测日志的序列模式挖掘算法 *SequPattMine* 利用哈希树存储候选序列，扫描所有方法序列的时间复杂度为 $O(n\times k)$，其中 n 表示序列数量，k 为序列长度，t 表示哈希树的叶子结点最大容量. 每次扫描数据集连接操作需要 $O(|L_k|\times|L_k|)$，扫描哈希树候选序列需要 $O(n\times t)$. 因此 *SequPattMine* 时间复杂度为 $O\left(n\times k+\sum_{k\geq 2}(|L_k|\times|L_k|+n\times t+|C_k|)\right)$.

TS 依据第三方构件中可能存在的安全漏洞的触发原因、构件的具体表现形式以及构件核心代码的特征生成相应的测试用例集合，它可以由如下的形式化定义所表示.

定义 12.　$TS=\{S_1,S_2,\cdots,S_i,\cdots,S_n\}$，$S_i$ 为一组测试执行用例序列集，它由若干个方法组成并可形式化为如下的表示：$S_i=(m_1,m_2,\cdots,m_i,\cdots,m_n)$，其中 $m_i\in Meth$ 且测试用例的执行的顺序依次为：$m_1,m_2,m_3,\cdots,m_n$.

TR 是第三方构件安全性测试的结果集合，它获取的主要方式是分析由构件产生的监测日志，它可以被形式化地定义为：

定义 13.　$TR=\{IV,EV\}$，IV 代表第三方构件的隐式安全漏洞，EV 表示显式安全漏洞，它们都通过一个或多个字符串连接组成.

显式安全漏洞的表现方式为：方法 m 存在显式安全异常，异常的类型是存取越界；隐式安全漏洞的表现方式为：方法执行序列 (m_1,m_2,m_3,m_5,m_4) 是不安全的执行序列.

SecRules 是指在第三方构件安全性测试中拟采用的构件安全性漏洞测试算法以及相应的安全规则集合，构件脆弱性包括显式及隐式两种. 构件的显式脆弱性指让构件在运行过程中出现明显安全异常的漏洞，如访问异常、内存溢出、被 0 整除、指针异常等；隐式脆弱性代表构件中方法的执行顺序与构件预先设定的安全需求说明相违背. *SecRules* 的形式化定义如下所示.

定义 14.　$SecRules=\langle Cr,Alg\rangle$，$Cr$ 代表安全异常模式集，Alg 代表检测构件安全漏洞算法集合. $Cr=\langle N,E\rangle$，其中 N 是异常规则名称，$N=\{n_1,n_2,n_3,\cdots,n_m\}$；$E$ 是异常规则的具体描述，$E=\{e_1,e_2,e_3,\cdots,e_m\}$.

为了更好地对定义 14 进行比较完整的解释，我们对算法 Alg 进行具体的描述. 算法 Alg 包含了两个子算法：*ExpDet* 和 *ImpDet*，其中 *ExpDet* 是指构件显式安全漏洞检测算法，*ImpDet* 是指构件隐式安全漏洞检测算法，两个子算法在这里就不再进行详细的描述. 算法 Alg 具体的过程算法 3 所示.

算法 3.　Alg.

```
输入：Meth′,ML,Cr,TS
输出：TR
1.  for each ml∈ML do
2.    if ml.m∉Meth′ then
3.      delete ml
4.    end if
5.  end for
6.  ExpTR=ExpDet(Meth′,ML,Cr)
7.  ImpTR=ImpDet(TS,ML)
8.  TR←ExpTR   //AddG ExpTR to TR
9.  TR←ImpTR
10. return TR
```

通过形式化地描述构件模型中的各元素对其具体的表现形式进行研究，得到由测试方法、测试规则、测试算法和测试结果构成的集合，从而为进一步提升构件安全性测试的准确率提供坚实的理论基础，并为面向第三方构件的安全性测试方法提供模型基础和测试准则.

4.2　测试框架

基于通过形式化方法构建的第三方构件安全性测试模型建立如图 2 所示的构件安全测试框架，用于对第三方构件安全性测试进行指导操作. 基于建立的测试框架，我们用形式化方法详细地阐述了面向第三方构件的安全性测试流程. 该流程具体如下所示：

(1) 对第三方构件的接口进行详细的分析，获取其包含的所有接口方法集合 *Meth*，但 *Meth* 中可能包含在第三方构件开发过程中自动加载的系统方法，如 AddRef、Release，因此，在测试时需要增加

额外的测试成本.然而,由于这些系统方法对构件的功能没有任何影响,因而可以通过将这些方法移除的方式得到构件的有效方法,为构件安全性测试提供保障.通过 *FrequPattMine* 算法挖掘由 *Meth*:*List*〈*string*〉和 *SF*:*List*〈*string*〉组成的〈*Meth*,*SF*〉中支持度大于等于 2 的频繁项集合,生成第三方构件中包含的有效接口方法集合 *Meth*′.

$$FrequPattMine:(SF,Meth)\rightarrow Meth'.$$

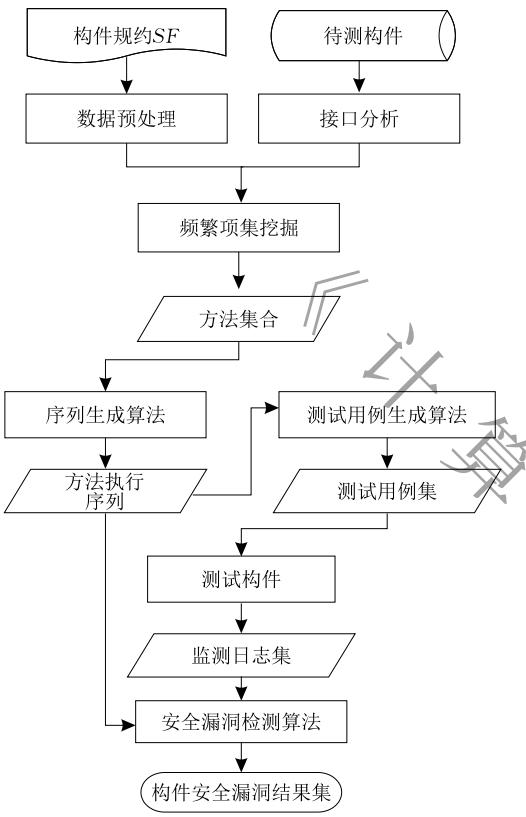

图 2 测试框架

(2)依据 *SF* 中 *Meth*′的 *PreA* 以及 *PostA*,建立第三方构件的方法执行序列集 *TS*,整个建立过程可以通过序列生成单子 GSMonad 进行表示.

type GSMonad *SF TS p m*=(*SF*,*TS*)→*p*(*m*,*TS*)

revert x=λ(*SF*,*TS*).*revert*(*x*,*TS*)

p '*combine*' *f*=

λ(*SF*,*TS*).(*p*(*SF*,*TS*) '*combine*' λ(*p*,*TS*′).*f m* (*SF*,*TS*′))

raise p=λ(*SF*,*TS*).(*p* '*combine*' λ*m*.*revert*(*m*,*TS*))

GSMonad *SF TS e* 将 *SF* 和 *TS* 作为参数进行输入,其中 *TS* 的初始值是空且它的结果由一系列的计算构成,而最终的计算结果由计算值和 *TS* 构成.*revert* 函数返回当初给定的值,在此过程中不需要作任何变动.*combine* 函数的工作过程如下所示:①将 *SF* 和 *TS* 传送给单子 *p* 执行计算操作,得到计算值 *m* 和临时的方法执行序列集合 *TS*′;②将函数 *f* 和 *m* 结合,生成单子 *fm*;③ *fm* 对 *SF* 和 *TS*′进行计算操作得到最终的 *m* 和 *TS*.提升函数 *raise* 的操作步骤是:①保留初始的 *SF* 和 *TS*;②计算 *p*;③返回计算得到的最终结果 *m* 和 *TS*.

(3)通过赋值操作为 *TS* 中方法的每个参数进行赋值并构建第三方构件方法测试用例集合 *Testcases*(以 XML 格式进行表示),然后对构件执行安全测试.其中,构件安全性测试操作可以转化为单子 TestMonad 的形式加以表现.

type *v*={*Testcases*::XML String,

CC::String,

FN::String}

type *s*={*TI*::Float,

RV::String}

type TestMonad *v s p m*=(*v*,*s*)→*p*(*m*,*s*)

revert x=λ(*v*,*s*).*revert*(*x*,*s*)

p '*combine*' *f*=

λ(*v*,*s*).(*p*(*v*,*s*) '*combine*' λ(*m*,*s*′).*f m* (*v*,*s*′))

raise p=λ(*v*,*s*).(*p* '*combine*' λ*m*.*revert*(*m*,*s*))

其中,TestMonad 中的参数 *v* 指的是测试环境,*s* 指的是测试状态.*v* 可以由测试用例、约束条件等组成;*s* 可以由运行时间和返回值等组成.由于在第三方构件的安全性测试过程中存在生成额外异常的情况,而 FaultT 单子有捕获异常的能力,所以可以考虑将 TestMonad 单子和 FaultT 单子通过组合的方式形成一个构件安全性测试单子 TestT,然后利用 InOutT 单子进行输出操作:

TestT=(TestMonad · FaultT) InOutT.

(4)通过 TestT 获取构件运行生成的 *ML*,并对 *ML* 进行相应的分析,如果 *ML* 中存在更多的安全漏洞信息,那么这些安全漏洞会被更加容易地检测出来.然后以 *Meth*′、*ML*、*TS*、*Cr* 为参数,利用算法 *Alg* 对第三方构件进行安全漏洞检测操作并生成安全漏洞信息 *TR*.

$$Alg:(Meth',ML,TS,Cr)\rightarrow TR.$$

其中,算法 *Alg* 中包含了序列模式挖掘算法 *SequPattMine*.算法将方法的执行序列 *S* 和监测日志集 *ML* 作为输入内容,首先从 *ML* 中得到方法序列 *MS*,然后在由 *S* 和 *MS* 构建的数据集〈*S*,*MS*〉中挖掘出支持度大于等于 2 的序列模式.通过对比 *S* 和挖掘得到的最长模式 *LS* 来对 *MS* 是否是不安全序列进行判断.*TR* 是最终的第三方构件安全漏洞检测结果,它将构件中包含的安全漏洞进行较为详细的呈现.

5　案例研究

为了更清晰地对第三方构件安全性测试模型的工作流程进行阐述，本节以 1 个在线购物构件 OnlineShoppingCom 作为例子进行具体的案例分析. 该构件有一个接口 IShopingInter，接口中有 7 个方法. 经过对该第三方构件进行接口信息分析得到具体的方法分别是：AddG（添加商品）、RemoveG（移除商品）、AccountG（结算）、PayG（付款）、DeliverG（发货）以及 AddRef、Release（这两个是系统缺省方法）. 根据该构建的说明文档和接口定义语言可以得到如表 2 所示的构建需求规约 *SF*.

表 2　OnlineShoppingCom 需求规约 *SF*

Meth	*P*	*PreA*	*PostA*
AddG	*a*	*flag*=0	/
RemoveG	*b*	*flag*=0&&*number*>0	/
AccountG	*c*	*number*>0	*flag*=1
PayG	*d*	*flag*=1	*flag*=2
DeliverG	*e*	*flag*=2	/

表 2 中的 *a* 代表需要被增加物品，*b* 代表需要被删除物品，*c* 代表已经存在于购物车中的物品，*d* 代表等将付款订单，*e* 代表已经完成的订单. *number* 代表存在于购物车中商品的数目，*flag* 代表当前订单的状态，其中 *flag*=0 指该当前订单没有完成，*flag*=1 指已经产生当前订单，*flag*=2 指已经支付当前订单.

第一步，采用 *FrequPattMine* 算法挖掘在 *SF* 中接口方法集合{AddG，RemoveG，AccountG，PayG，DeliverG，AddRef，Release}中挖掘构件自定义的有效方法集合，排除系统的缺省方法，从而得到有效的构件方法 *Meth′*={AddG，RemoveG，AccountG，PayG，DeliverG}.

第二步，根据 *Meth′* 的 *PreA* 和 *PostA* 生成 *TS*，利用单子 GSMonad 得到以下的构件测试执行序列：*TS*={(AddG，AccountG，PayG，DeliverG)，(AddG，RemoveG，AddG，AccountG，PayG，DeliverG)，(AddG，AddG，AccountG，PayG，DeliverG)}

第三步，通过依次读取方法的执行序列操作对该第三方构件进行安全性检测. 如：任意选取一条构件的方法执行序列（AddG，RemoveG，AddG，AccountG，PayG，DeliverG），将该序列生成测试用例并加以测试，通过单子 TestT 对在测试过程中构件的返回信息进行详细的记录，得到一条完整的构件安全日志 *ML*，*ML* 的具体内容如表 3 所示.

表 3　监测日志集 *ML*

TI	*Meth*	*CC*	*FN*	*P*	*RV*	*EI*
09:12:13	AddG	*flag*=0	RemoveG	键盘	添加成功	/
09:12:14	RemoveG	*flag*=0&&*number*>0	AddG	键盘	移除失败	存取越界
09:12:16	AddG	*flag*=0	PayG	鼠标	添加成功	/
09:12:17	PayG	*number*>0	DeliverG	未支付订单	支付成功	/
09:12:19	DeliverG	*flag*=2	/	已支付订单	发货成功	/

第四步，采用算法 *Alg* 对 *ML* 做详细的检测与分析操作，在对方法 RemoveG 进行检测时出现存取越界显式安全异常. 在隐式安全漏洞异常检测阶段，*ML* 中方法的执行序列依次是：(AddG，RemoveG，AddG，AccountG，PayG，DeliverG)，而 *TS* 中的正确方法执行序列依次是：(AddG，RemoveG，AddG，AccountG，PayG，DeliverG)，因此它与正确的方法执行序列相违背.

因此，最终的构件漏洞安全检测结果是：(1) 显式安全漏洞. 方法 RemoveG 存在存取越界的显式异常；(2) 隐式安全漏洞. 构件执行的方法执行序列(AddG，RemoveG，AddG，AccountG，PayG，DeliverG)与正确的顺序相违背.

6　模型实现及实验分析

为验证提出的测试模型及其框架的可行性及效果，6.1 节论述了在测试原型系统 CSTS-DM 中模型元素和相关的算法如何在系统中实现，6.2 节“典型案例分析”和 6.3 节“实验对比分析”中通过对商业构件和模拟构件进行分析测试，进一步验证本方法的有效性.

6.1　模型实现

基于上述的测试模型和框架，我们已经在 Windows 平台上基于 Dot Net 框架下的 C++与 C# 实现了论文提出的第三方构件安全性测试原型系统 CSTS-DM. 系统界面如图 3 所示.

图 3 CSTS-DM 原型系统快照

菜单栏中显示了文件、挖掘、接口分析、生成测试用例、生成方法序列、测试执行及结果分析等模块. 系统文件模块用来加载指定路径的待测构件,为该构件创建独立的测试项目,并包含保存和打开项目的功能;挖掘模块主要是挖掘分析需求规约和初始方法集合,消除构件开发过程中系统自动生成的接口方法,找出有效的方法集合;接口分析模块主要是分析构件暴露出来的接口信息并将这些信息写入 XML 结构文件;在生成方法序列时,将基于接口方法及方法前后置等条件,得到有效的测试执行序列;测试用例生成模块主要是结合接口方法参数的取值,生成可执行的测试用例以及脚本测试程序;构件测试模块主要是执行测试脚本进行构件测试,实时地监控构件运行,并记录日志;安全漏洞检测模块主要就是分析处理监测日志以检测构件中存在的安全漏洞.

原型系统 CSTS-DM 的主要测试流程:(1) 新建项目. 通过菜单"项目文件"下"新建项目"加载待测构件,创建独立的测试项目;(2) 接口分析. 通过菜单"接口分析",系统就开始分析构件的基本信息,主要有:对象名、接口名、所有接口方法名以及各方法的返回值类型及参数信息等. 构件基本信息被保存到 XML 结构文件中,最终在系统左侧以树状结构展示;(3) 频繁项集挖掘. 通过菜单"频繁项集挖掘",开始挖掘由构件初始方法和需求说明 *SF* 组成的数据集,生成有效的接口方法数据集;(4) 方法序列生成:通过菜单"方法序列生成",根据每个有效的构件接口方法和方法的前后置条件,生成符合条件的方法序列集;(5) 测试用例生成. 通过生成测试用例功能,测试系统通过测试用例生成算法产生可执行的测试用例执行脚本,并且该脚本程序可以手动进行更改,以此来获得更高质量的测试用例;(6) 构件测试. 通过菜单"构件测试",执行测试脚本构件测试,并将构件运行时产生的监测日志信息实时地在界面下方的消息框中显示;(7) 安全漏洞检测. 通过菜单"安全漏洞检测",开始分析监测日志,并在界面下方的消息框中显示构件的显式和隐式安全漏洞等信息.

6.2 典型案例分析

通过以下 4 个构件案例我们对 CSTS-DM 原型系统进行了测试和分析,分析结果如下:

(1) 第三方商业构件:ThunderAgent_005.dll

构件功能:迅雷的 ActiveX 控件

版本信息:1.0.0.11

发现漏洞:接口函数 GetTaskStructFunc() 未考虑到输入参数异常的情况,当输入参数值超过阈值时,导致整数溢出. 函数原型为:void GetTaskStructFunc (int pTaskStruct).

(2) 第三方商业构件:GLItemCom.dll

构件功能:联众世界游戏大厅 ActiveX 控件

版本信息:2.7.0.8

发现漏洞:接口函数 SetUserInfoFunc() 未考

虑到输入参数异常的情况，内部没有判断输入的字符串长度，当输入字符串过长时，函数会抛出异常. 函数原型为：void SetUserInfoFunc(BSTR itemStr, BSTR gameStr).

(3) 模拟实验构件：Calculator.dll

构件功能：实现了计算器功能

版本信息：1.0.0.1

发现漏洞：接口函数 DivAlgorithm()未考虑参数异常输入，如果第二个输入值为 0，函数会产生异常. 函数原型为：void DivAlgorithm (int dividend, int divisor, int&reval).

(4) 模拟实验构件：OnlineShoppingCom.dll

构件功能：实现了模拟在线购物的功能

版本信息：1.0.0.1

发现漏洞：当特殊情况用例输入测试，内部函数调用关系不正确，构件安全异常将触发.

每一个构件在 CSTS-DM 测试系统中都有大量的测试脚本用例生成，以下仅给出了典型的使构件产生安全异常的用例脚本.

(1) ThunderAgent_005.dll 的典型用例：

```
[Test]
public void GetTaskStructTest (){
  myAgentObj. GetTaskStructFunc(0x7fffffff);
}
```

(2) GLItemCom.dll 的典型用例：

```
[Test]
public void SetUserInfoTest (){
  string itemStr=new string('C',325);
  string gameStr=new string('B',438);
  myGLItemComObj.SetUserInfoFunc(itemStr,
  gameStr);
}
```

(3) Calculator.dll 的典型用例：

```
[Test]
public void DivAlgorithmTest(){
  myCalculatorObj. DivAlgorithm (9,0);
}
```

(4) OnlineShoppingCom.dll 的典型用例：

```
[Test]
public bool IShoppingAddGTest (){
  myShoppingObj.AddG ("键盘");
  return true;
}
[Test]
public bool IShoppingRmTest(){
  myShoppingObj.RemoveG ("键盘");
  return true;
}
[Test]
public bool IShoppingAddGTest(){
  myShoppingObj.AddG ("鼠标");
  return true;
}
[Test]
public bool IShoppingAccountGTest() {
  myShoppingObj.AccountG ("购物车商品");
  return true;
}
[Test]
public bool IShoppingPayGTest() {
  myShoppingObj.PayG ("待支付订单");
  return true;
}
[Test]
public bool IShoppingDeliverGTest() {
  myShoppingObj.DeliverG ("已支付订单");
  return true;
}
```

基于以上分析测试，系统产生了四个构件案例的安全性检测结果，具体信息如表 4 所示. 分析结果与业界公布的漏洞及植入的安全异常信息一致，进一步验证了本方法的有效性.

表 4　被测构件结果分析

构件名称	结果分析
ThunderAgent_005.dll	接口函数 GetTaskStructFunc 产生异常，存取越界，属于显式安全漏洞
GLItemCom.dll	接口函数 SetUserInfoFunc 产生异常，存取越界，属于显式安全漏洞
Calculator.dll	接口函数 DivAlgorithm 产生异常，除数为 0，属于显式安全漏洞
OnlineShoppingCom.dll	发现不安全执行序列(AddG, RemoveG, AddG, PayG, DeliverG)，属于隐式安全漏洞

为了能够得到有效的方法集合，数据挖掘算法 FrequPattMine 预先对数据集$\langle SF, M\rangle$进行分析，消除了构件开发过程中缺省的接口方法，为构件安全性测试剔除了相关无关的被测试方法，减少了后续的测试开销. 此外，通过 SequenceMining 算法得到可行的方法序列和测试中的实际执行方法序列，进一步通过序列模式算法比对方法序列，可以检测构件中非显式安全异常，所以在构件安全性分析测试中算法 SequenceMining 发挥了关键作用.

6.3　实验对比分析

为了进一步分析基于 DMTM 模型的原型系统

CSTS-DM 的效果，我们自己模拟实现 5 个 COM 构件：BankTransaction.dll、Examine.dll、Calculator.dll、OnlineShoppingCom.dll 和 beveragevending.dll. 并对这 5 个构件注入一些安全异常错误，5 个构件的信息如表 5 所示. 我们分别和基于错误注入的测试工具 CSTS[29] 及接口覆盖测试方法[10,30] 进行了对比实验，实验结果如表 6 所示.

表 5 被测试构件信息

构件名	构件描述及功能	方法数	异常数
BankTransaction.dll	具有验证、查询、事务处理及转账等功能的构件	6	3
Examine.dll	统计学习考试成绩，并根据总成绩给出评语的构件	6	2
Calculator.dll	实现加减乘除功能计算构件	9	4
OnlineShoppingCom.dll	具有付款、发货、清结算等功能的电商构件	8	3
BeverageVending.dll	投币、购买饮料、找零的商品自助售卖机构件	7	3

表 6 实验对比结果

构件名	测试方法	检测错误数
BankTransaction.dll	CSTS-DM	6
	CSTS	4
	接口覆盖法	2
Examine.dll	CSTS-DM	3
	CSTS	2
	接口覆盖法	1
Calculator.dll	CSTS-DM	8
	CSTS	5
	接口覆盖法	3
OnlineShoppingCom.dll	CSTS-DM	6
	CSTS	4
	接口覆盖法	2
BeverageVending.dll	CSTS-DM	5
	CSTS	3
	接口覆盖法	1

表 6 中检测到的安全异常错误数包括重复检测到的错误. 从表 6 中可以看出 CSTS-DM 方法要优于 CSTS 和接口覆盖法，基于错误注入的 CSTS 方法则要好于接口覆盖法. 图 4 显示了三种方法在 5 个构件上的平均对比结果，总体来看，随着测试用例的增加 CSTS-DM 发现的异常数要比 CSTS 和接口覆盖法多，主要原因是 CSTS-DM 方法采用了数据挖掘技术去动态分析监测日志从而能发现更多的错误数.

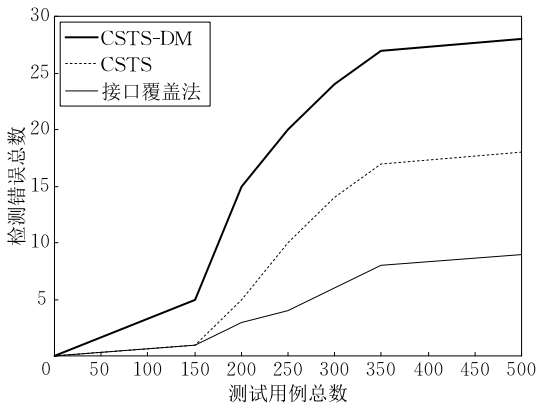

图 4 五个构件总体效果实验对比

本文还将提出的 DMTM 模型与当前相关的其它技术进行了比较. 比较的主要相关技术和方法有：基于需求规约的构件元数据与接口规约黑盒测试模型、基于错误注入技术的构件脆弱性检测方法、基于形式化状态机模型的健壮性测试技术及一种理论安全评估模型和框架. 详细的比较信息如表 7 所示，结果显示提出的 DMTM 模型可以对第三方构件进行安全漏洞检测，且提供了具体的实现方法.

表 7 相关的构件测试方法比较

测试方法	是否有支持工具	是否适用于第三方构件	效果分析
Naseer 等人[8]	否	否	对较好地验证元数据和接口规约模型，却需要构件需求规约支撑，无法应用于第三方构件.
Chen 等人[10-11]	是	是	对构件显式脆弱性具有较好的检测效果，但需要软件错误注入技术支持.
Lei 等人[12]	是	否	对通用的软件构件有的检测能力，无法应用于第三方构件脆弱性测试.
Khan 等人[13]	否	否	对普通构件具有一定的验证效果，提出了安全模型及评估框架，但相关理论和方法未能实现，无有效测试工具.
本文测试模型 DMTM	是	是	对第三方构件脆弱性具有一定的检测效果，通过数据挖掘算法分析构件规约和测试日志.

7 结 论

第三方构件通常由不同的组织开发设计，相关的开发文档和源代码无法获取，其以二进制形式提供给集成商，此特性给测试第三方构件安全性带来挑战. 借助数据挖掘现有的一些算法，能够筛选构件的一些系统内置的方法及无效的接口方法，生成被

测试构件的方法测试序列及构件自定义需求等数据，同时分析处理大量的监测测试日志.论文通过研究数据挖掘技术提出并验证了适用于第三方构件安全性测试的测试模型及相关算法.对模型的各元素进行了形式化定义，随后我们基于该模型设计了一个安全性测试框架，并采用了单子技术对测试的流程进行了详细地描述.在该模型中，首先通过数据挖掘算法 FrequPattMine 挖掘需求规约得到有效的接口方法，然后利用序列生成单子转换器 GSMonad 产生构件方法安全性测试用例，进一步借助构件单子 TestT 生成测试日志，并利用脆弱性测试 Alg 算法对测试日志进行数据分析得到脆弱性检测结果.最后，我们通过案例研究和实验分析，验证了提出的测试模型及其测试框架的正确性和有效性.

未来的工作主要有：(1)为了提供更准确的测试准则，将完善测试模型各元素的接口规范和运算规范；(2)针对目前挖掘及匹配不够准确的问题，将研究基于机器学习的数据挖掘算法，通过对现有构件漏洞的学习建立检测规则，提高检测的准确性；(3)对现有的构件安全性测试系统 CSTS-DM 进行自动化改造，优化相关的挖掘算法，减少各部分人工干预的操作，确实提高 CSTS-DM 的自动化程度.

致　谢　感谢课题组的陈加梅、苏晨飞、葛宏河及于敏杰等同学参与了论文原型系统的设计和实现.同时也要感谢审稿专家们，您们的宝贵意见促使论文质量得到了进一步提升！

参　考　文　献

[1] Chen Jin-Fu, Lu Yan-Sheng, Wang Huan-Huan. Component security testing approach based on extended chemical abstract machine. International Journal of Software Engineering and Knowledge Engineering, 2012, 22(1): 59-83

[2] Lionel C B, Yvan L, Michal M S, et al. Automated, contract-based user testing of commercial-off-the-shelf components//Proceedings of the 28th International Conference on Software Engineering (ICSE). Shanghai, China, 2006: 92-101

[3] Scandariato R, Walden J, Hovsepyan A, et al. Predicting vulnerable software components via text mining. IEEE Transactions on Software Engineering, 2014, 40(10): 993-1006

[4] Mao Cheng-Ying, Lu Yan-Sheng. Research progress in testing techniques of component-based software. Journal of Computer Research and Development, 2006, 43(8): 1375-1382(in Chinese)
(毛澄映，卢炎生. 构件软件测试技术研究进展. 计算机研究与发展，2006，43(8)：1375-1382)

[5] Nagappan M, Mirakhorli M. Big (ger) data in software engineering//Proceedings of the 37th International Conference on Software Engineering (ICSE). Florence, USA, 2015: 957-958

[6] Ma Liang-Li, Guo Fu-Liang, Li Yong-Jie. Component integration test based on a metadata model of coupling testing information. Chinese Journal of Computers, 2007, 30(10): 1705-1712(in Chinese)
(马良荔，郭福亮，李永杰. 基于耦合测试信息元数据模型的构件集成测试. 计算机学报，2007，30(10)：1705-1712)

[7] Li Liang-Ming, Wang Zhi-Jian, Tang Long-Ye. Research on the testing of component functional behavior. Journal of Chinese Computer System, 2010, 31(4): 686-690(in Chinese)
(李良明，王志坚，唐龙业. 构件功能行为测试的研究. 小型微型计算机系统，2010，31(4)：686-690)

[8] Naseer F, Hussain K. Using meta-data technique for component based black box testing//Proceedings of the 6th International Conference on Emerging Technologies (ICET). Islamabad, Pakistan, 2010: 276-281

[9] Winter S, Sarbu C, Murphy B, et al. The impact of fault models on software robustness evaluations//Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011). Hawaii, USA, 2011: 51-60

[10] Chen Jin-Fu, Lu Yan-Sheng, Xie Xiao-Dong. Component security testing approach by using interface fault injection. Journal of Chinese Computer System, 2010, 31(6): 1090-1096(in Chinese)
(陈锦富，卢炎生，谢晓东. 一种采用接口错误注入的构件安全性测试方法. 小型微型计算机系统，2010，31(6)：1090-1096)

[11] Chen Jin-Fu, Lu Yan-Sheng, Xie Xiao-Dong. A fault injection model of component security testing. Journal of Computer Research and Development, 2009, 46(7): 1127-1135(in Chinese)
(陈锦富，卢炎生，谢晓东. 一种构件安全测试错误注入模型. 计算机研究与发展，2009，46(7)：1127-1135)

[12] Lei Bin, Wang Lin-Zhang, Bu Lei, Li Xuan-Dong. Robustness testing for components based on state machine model. Journal of Software, 2010, 21(5): 930-941(in Chinese)
(雷斌，王林章，卜磊，李宣东. 基于状态机模型的构件健壮性测试. 软件学报，2010，21(5)：930-941)

[13] Khan K M, Han J. Assessing security properties of software components: A software engineer's perspective//Proceedings of the 17th Australian Software Engineering Conference (ASWEC 2006). Sydney, Australia, 2006: 199-210

[14] Wu Jian-Kun. Formal Analysis and Construction for Internetware-Based Software System [Ph. D. dissertation]. Shanghai Jiao Tong University, Shanghai, 2009(in Chinese)
(伍建焜. 网构软件系统构建的形式化分析研究[博士学位论文]. 上海交通大学，上海，2009)

[15] Wile D, Balzer R, Goldman N, et al. Adapting COTS products //Proceedings of the 26th IEEE International Conference on Software Maintenance (ICSM 2010). Timişoara, Romania, 2010: 1-9

[16] Tao Chuanqi, Li Bixin, Gao Jerry. Regression testing of component-based software: A systematic practise based on state testing//Proceedings of the 13th IEEE International Symposium on High-Assurance Systems Engineering (HASE). Florida, USA, 2011: 29-32

[17] Xu Jing, Si Guan-Nan, Yang Ju-Feng, et al. An Internetware dependable entity model and trust measurement based on evaluation. Scientia Sinica Informationis, 2013, 43(1): 108-125(in Chinese)
(许静, 司冠南, 杨巨峰等. 一个网构软件可信实体模型及基于评估的信任度量. 中国科学:信息科学, 2013, 43(1): 108-125)

[18] Si Guan-Nan, Ren Yu-Han, Xu Jing, et al. A dependability evaluation model for Internetware based on Bayesian network. Journal of Computer Research and Development, 2012, 49(5): 1028-1038(in Chinese)
(司冠南, 任宇涵, 徐静等. 基于贝叶斯网络的网构软件可信性评估模型. 计算机研究与发展, 2012, 49(5): 1028-1038)

[19] Xie T, Thummalapenta S, Lo D, et al. Data mining for software engineering. Computer, 2009, 42(8): 55-62

[20] Mao Cheng-Ying, Lu Yan-Sheng, Hu Xiao-Hua. Data mining applications on the software engineering tasks: A state of the art. Computer Science, 2009, 36(5): 1-6 (in Chinese)
(毛澄映, 卢炎生, 胡小华. 数据挖掘技术在软件工程中的应用综述. 计算机科学, 2009, 36(5): 1-6)

[21] Khan Z, Haseen F, Rizvi S T A. Enhanced BitApriori algorithm: An intelligent approach for mining frequent itemset. Advances in Intelligent Systems & Computing, 2015, 32(5): 23-34

[22] Mining S. Efficient and accurate discovery of patterns in sequence datasets. IEEE Transactions on Knowledge & Data Engineering, 2010, 23(8): 461-472

[23] Wu C W, Fournier-Viger P, Yu P S. Efficient algorithms for mining the concise and lossless representation of closed+high utility itemsets. IEEE Transactions on Knowledge & Data Engineering, 2015, 27(3): 726-739

[24] Hryniow K. Parallel pattern mining-application of GSP algorithm for graphics processing units//Proceedings of the International Carpathian Control Conference (ICCC). Kosice, Slovakia, 2012: 233-236

[25] Liang S, Hudak P, Jones M. Monad transformers and modular interpreters//Proceedings of the 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. New York, USA, 1995: 333-343

[26] Zhang Ying-Zhou, Fu Wei. An approach of monadic slicing for interprocedural programs. Acta Electronica Sinica, 2013, 41(8): 1457-1461(in Chinese)
(张迎周, 符炜. 一种过程间单子切片方法. 电子学报, 2013, 41(8): 1457-1461)

[27] Zhang Ying-Zhou, Xu Bao-Wen. An approach to dynamic program slicing based on modular monadic semantics. Chinese Journal of Computers, 2006, 29(4): 526-534(in Chinese)
(张迎周, 徐宝文. 一种基于模块单子语义的动态程序切片方法. 计算机学报, 2006, 29(4): 526-534)

[28] Yuan Qi . Research on Monad Theory and Its Application [Ph. D. dissertation]. Jilin University, Changchun, 2000(in Chinese)
(袁琦. Monad 理论及其应用研究[博士学位论文]. 吉林大学, 长春, 2000)

[29] Chen Jin-Fu, Lu Yan-Sheng, Xie Xiao-Dong. CSTS: A prototype tool for testing COM component security//Proceedings of the 9th International Conference on Hybrid Intelligent Systems (HIS-2009), Shenyang, China, 2009: 83-88

[30] Yang Jian-Jun, Chen Wei-Dong, Ye Cheng-Qing, et al. Interface mutation for component. Journal of Zhejiang University (Engineering Science), 2003, 37(2): 129-133(in Chinese)
(杨建军, 陈卫东, 叶澄清等. 面向组件的接口变异测试方法. 浙江大学学报(工学版), 2003, 37(2): 129-133)

CHEN Jin-Fu, Ph. D., professor. His major research interests include software testing and trusted software.

ZHAO Xiao-Lei, M. S. candidate. His research interests focus on software testing.

LIU Yi-Song, Ph. D., professor. His major research interests include computer graphics, human-computer interaction techniques and software engineering.

HUANG Ru-Bing, Ph. D., associate professor. His major research interests include software testing and trusted software.

CAI Sai-Hua, M. S. candidate. His research interests focus on software testing.

GUO Yu-Chi, M. S. candidate. Her research interests focus on software testing.

Background

With the popularization of the third-party components, the research on security testing for third-party components is very important. Component security vulnerabilities mean the flaws in the aspects of component security including all the factors that threatening and destroying component security. Component security vulnerabilities usually include explicit and implicit vulnerabilities. Explicit vulnerabilities are commonly caused by memory leak or buffer overflow; while implicit vulnerabilities are usually caused by violating security requirement specification. Both explicit vulnerabilities and implicit vulnerabilities are very difficult to be detected by traditional approaches, which lack effective detection model and detection means. With technology of data mining, the effective requirements specification and testing sequence of interface methods can be mined and got by analyzing large and complex testing log. This paper proposes a security testing model of third-party component based on data mining, which can effectively detect component security vulnerabilities. In the testing model, the elements of the model are defined and described through formal definition. Then a testing framework is presented based on the model and the testing process is also described based on monad technology.

The main content studied in this paper is a part of the previous project. The main modules of this project include specification mining, method execution sequence mining and component security testing. The specification mining and method execution sequence mining module have been addressed in previous work. In addition, the principle of component security testing is to use data mining techniques such as data classification, frequent item and sequence pattern mining algorithm to mine security association rule, abnormal methods and sequences based on monitor log. Component security testing approaches are the current research focus. In future, we will improve the proposed testing model, and based on the testing model address some effective component security testing approaches.

These research contents are part of the project of National Natural Science Foundation of China (Project name: Research on the Testing Approach of the Third-Party Component Security Based on Data Mining, No. 61202110). This work is also supported partly by the Natural Science Foundation of Jiangsu Province under Grant No. BK2012284, and the Research Fund for the Doctoral Program of Higher Education of China under Grant No. 20103227120005.