软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(1):37−49 [doi: 10.3724/SP.J.1001.2013.04334]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

字符串分析研究进展*

梅 宏[1,2], 王啸吟[1,2], 张 路[1,2]

[1](高可信软件技术教育部重点实验室(北京大学),北京 100871)

[2](北京大学 信息科学技术学院 软件工程研究所,北京 100871)

通讯作者: 张路, E-mail: zhanglu@sei.pku.edu.cn

摘 要: 随着软件应用范围的不断扩大,尤其是数据库软件和 Web 软件的广泛应用,字符串变量在软件程序中扮演的角色日益重要.与此同时,针对字符串变量的程序分析技术——字符串分析,也取得了长足的发展,并在软件工程中的很多领域中得到了成功的应用.字符串分析的基本应用模式是首先使用字符串值分析获得字符串变量的所有可能取值,然后使用字符串约束求解判断这些变量的取值是否满足一定约束,从而对程序进行正确性验证.为了使得字符串分析能够应用在安全分析和软件维护应用中,研究人员对字符串分析进行了扩展,进一步分析字符串变量的数据来源.综述了字符串分析技术的研究进展,提出了字符串分析的问题构型,介绍了这一领域现在的主要研究内容:字符串值分析、字符串约束求解、字符串数据来源分析以及字符串分析在软件工程中的应用.

关键词: 程序分析;字符串分析;约束求解

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 梅宏,王啸吟,张路.字符串分析研究进展.软件学报,2013,24(1):37−49. http://www.jos.org.cn/1000-9825/4334.htm

英文引用格式: Mei H, Wang XY, Zhang L. Progress in research on string analysis. Ruanjian Xuebao/Journal of Software, 2013, 24(1):37−49 (in Chinese). http://www.jos.org.cn/1000-9825/4334.htm

Progress in Research on String Analysis

MEI Hong[1,2], WANG Xiao-Yin[1,2], ZHANG Lu[1,2]

[1](Key Laboratory on High Confidence Software Technology of Ministry of Education (Peking University), Beijing 100871, China)

[2](Software Engineering Institute, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

Corresponding author: ZHANG Lu, E-mail: zhanglu@sei.pku.edu.cn

Abstract: With the ubiquitous software application, especially the wide usage of database applications and Web applications, strings have become a more important role in the software programs. At the same time, the program analysis techniques that consider the specialty of strings have been developed, and have been applied to various areas in software engineering. Usually, string value analysis is applied to acquire the possible values of a given string variable. Next, a constraint solver is applied to check whether the values satisfy predefined specifications, so that the correctness of the given string variable can be checked. To further apply string analysis to some security analysis and software maintenance problems, the string analysis is further improved to analyze the possible data origins of a given string variable. This paper presents a survey on string analysis, which mainly introduces the string value analysis, string constraint solving, string data origin analysis, and the applications of string analysis in software engineering.

Key words: program analysis; string analysis; constraint solving

随着计算机软件的应用重心从数值计算转向业务处理,字符串类型的变量和常量在软件程序中的使用越

* 基金项目: 国家重点基础研究发展计划(973)(2009CB320703); 国家自然科学基金(60931160444, 91118004, 61225007)
收稿时间: 2012-06-29; 定稿时间: 2012-10-16; jos 在线出版时间: 2012-11-23
CNKI 网络优先出版: 2012-11-23 12:12, http://www.cnki.net/kcms/detail/11.2560.TP.20121123.1212.004.html

来越频繁.例如,在基于数据库的软件中,SQL 语句动态生成主要通过字符串操作完成.图形界面软件中也包含大量的用于显示的字符串变量和常量,并通过字符串操作动态地生成需要显示的字符串.在作为 Internet 时代主要软件类型的 Web 应用软件中,字符串变量和常量更是界面的基本构成元素,Web 应用软件的整个用户界面以及界面的相关源代码都由字符串操作产生[1].在当前被广泛采用的多数主流语言(例如 Java,C#,PHP 等)中,字符串都是基本类型,字符串拼接操作也有相对应的运算符,而且这些语言的基本库函数中都包含一些字符串操作函数,例如字符串比较、字符替换、字符串切分等[2−4].

由于早期的软件主要用于数值计算,因此传统的程序分析主要针对源代码中的数值量以及数值操作.对于字符串变量和字符串操作,传统的程序分析仅将其作为无法确定取值的数值变量和未知库函数来处理[5].例如,对于涉及字符串变量的分支条件,传统程序可达分析只能假定字符串变量的值为任意,从而简单地判定任一分支都是可达的.对于字符替换这一字符串操作,传统的数据依赖分析也只能简单判定这一操作的所有操作数(原字符串、被替换字符、替换字符)都是操作输出的可能数据来源.实际上,被替换字符这一操作数不可能是操作输出的数据来源,而其他两个操作数是否可能是操作输出的数据来源取决于它们的取值.在抽象解释领域,有学者提出使用前缀/后缀抽象域(即前缀或后缀为特定常量的字符串集合)表示字符串的运行时取值,但是这种方法只能近似处理赋值操作,无法处理包括字符串拼接在内的各种字符串操作[6].2003 年,Christensen 等人提出了一种专门针对程序中的字符串变量的程序分析[7].这种分析方法能够通过分析程序中字符串变量和常量的赋值和操作,获得程序中给定字符串变量的所有可能取值,并用正则语言表示.此后,针对字符串变量的程序分析迅速发展,并被广泛应用于软件验证、软件维护演化和软件测试等领域的一些实际问题中[8−10].这些用于判定给定字符串变量某些性质(如取值)的程序分析,在本文中统称为字符串分析.

下面首先介绍字符串分析的问题构型.接下来介绍目前字符串分析领域的主要研究方向.然后分类介绍目前字符串分析的主要应用.最后对未来研究进行展望.

1 问题构型

字符串分析是用于判定软件程序中给定字符串变量(一般称为热点变量)的某些性质的程序分析.与针对数值变量的程序分析相比,字符串分析必须处理字符串变量的两个特性:

- 首先,字符串变量的值域与数值变量不同:数值变量的值域是一个数值集合,通常可以表示为数轴上若干个离散的点或连续区间;而字符串变量的值域是一个字符串集合,需要一个形式化的符号系统(例如正则语言)准确地描述这一字符串集合;
- 其次,字符串变量上定义的操作(拼接、字符替换等)与数值变量上的操作不同,因此需要准确地确定这些操作会对字符串变量上所关注的性质产生怎样的影响,即对于这些字符串操作,已知操作数的性质,如何确定操作结果的性质.

对字符串变量这两个特性的处理,即字符串变量值域的抽象表示和字符串操作的处理方式是字符串分析与传统的针对数值变量的程序分析的主要区别,也是字符串分析的主要研究内容和难点.

目前,根据实际应用的需要,字符串分析领域的研究主要针对热点变量的两类性质:字符串变量取值相关的性质以及字符串变量数据来源相关的性质.

对于字符串变量取值相关的性质,实际的应用通常是判定给定的字符串取值是否合法(如符合 SQL 语法)这类比较复杂的性质.对于这类性质,目前的字符串分析方法通常需要分两个步骤进行判定,即首先确定热点变量的值域,然后将判定是否合法的标准转化成针对字符串值的约束,并进一步确定字符串变量的值域是否满足给定的字符串值约束.在下文中,将前一个步骤称为字符串值分析,后一个步骤称为字符串约束求解.字符串值分析的难点即是前文指出的针对字符串特点的处理,而字符串约束求解的难点在于如何高效地求解用于解决实际问题所需要的各种约束.

对于字符串变量数据来源相关的性质,目前的主要研究方法是在字符串值分析的基础上,通过标记传播确定热点变量的数据来源,因此,在针对字符串变量特点的处理上通常采用与字符串值分析相同的方法,而其难点

在于如何定义标记和如何在字符串值分析的过程中传播标记.

下文将分别介绍确定热点变量值域的字符串值分析,判定热点变量值域是否满足给定字符串值约束的字符串约束求解,以及判定热点变量数据来源相关性质的字符串数据来源分析.

2 字符串值分析

字符串值分析是一种通过分析程序判断热点变量的值域的一种程序分析.这种分析的输入是软件项目的源代码以及源代码中的热点变量,输出即是描述热点变量的所有运行时可能取值的一个符号系统.一般要求字符串值分析是保守的,即热点变量的任何取值都必然能被输出的符号系统所描述,但该符号系统可能也可以描述热点变量所不可能取值的字符串.

根据使用描述热点变量取值的符号系统,现有字符串值分析可以分为基于正则文法的字符串值分析、基于上下文无关文法的字符串值分析和基于带一元谓词的二阶逻辑的字符串值分析.

2.1 基于正则文法的字符串值分析

由于基于正则文法的字符串值分析是最早提出的一种针对字符串的程序分析,在文献中也直接称为字符串分析(string analysis).它是 2003 年由 Christensen 等人提出来的[7].

Christensen 等人的方法是使用正则文法作为表示字符串变量值域的符号系统,并使用预定义的自动机映射模拟字符串操作,包括如下 4 个步骤:

(1) 将软件源代码转化为静态单赋值形式

首先,将待分析程序的源代码转化为静态单赋值形式(static single assignment,简称 SSA)[11].静态单赋值形式是 1991 年提出的一种源代码形式,在将源代码转化为静态单赋值形式的过程中,如果一个变量在源代码中被赋值多次,则会将这个变量通过换名拆分成多个变量,因此,在静态单赋值形式的源代码中,每个变量只被赋值 1 次.当源代码中的程序分支导致一个变量在一次赋值中可能被赋不同的值时,静态单赋值形式的源代码使用一个特殊的φ函数描述这种赋多个值的情况,例如,$x=\varphi(a,b)$表示 x 被赋值为 a 也可能被赋值为 b.

(2) 提取字符串操作文法

使用程序数据依赖分析对静态单赋值形式的代码进行分析,结果是包含程序中所有变量/常量/表达式之间数据依赖关系的一个程序数据依赖图.热点变量是这个字符串依赖图中的一个节点.字符串分析通过在程序数据依赖图上进行可达分析删去从热点变量不可达的节点,并删去那些不对应字符串类型的变量/常量/表达式的节点,从而得到一个字符串依赖图.该图的节点集合对应热点变量所可能依赖的所有字符串类型的变量/常量/表达式.

字符串操作文法是一个扩展的上下文无关文法.字符串操作文法定义为一个六元组($N,T,S,P,OP1,OP2$).其中,N 为非终结符的集合,T 为终结符的集合,S 为起始非终结符,P 为产生式集合,$OP1$ 为一元字符串操作(例如 Java 语言中的 trim 操作)集合,$OP2$ 为二元字符串操作的集合(例如 Java 语言中的对字符串进行 replace 的操作).在从字符串依赖图中提取文法时,将每个赋值关系(边)转化为一个产生式,将字符串依赖图中的常量/变量节点转化为终结符/非终结符,将字符串拼接表达式节点转化为文法中的符号序列,将其他字符串表达式节点转化为文法中的字符串操作.

(3) 对字符串操作文法进行线性近似,得到一个正则操作文法

正则近似的目标是求出一个正则操作文法 N,这个正则操作文法的语言 $L(N)$是字符串操作文法 G 的语言 $L(G)$的尽可能小的超集.由于字符串操作文法是扩展的上下文无关文法,因此可以使用对上下文无关文法进行线性近似的 Mohri-Nederhof 算法[12]对字符串操作文法进行近似.对上下文无关文法进行正则近似的基本思路是,通过使用Σ^*近似去掉上下文无关文法的左递归圈和右递归圈,从而得到一个包含字符串操作的正则操作文法.

(4) 消除字符串操作

字符串分析的最后一个步骤是消除正则操作文法中的字符串操作,从而得到一个真正的正则文法来估计

热点变量的所有可能取值.基于正则文法的字符串分析采用预定义的自动机映射来解决这个问题.预定义的一系列映射规则描述给定正则文法通过字符串操作之后所得到的文法.

基于正则文法的字符串值分析是最早被提出来的字符串值分析方法,其优点是分析速度较快;缺点是由于正则语言的表达能力有限,得出的正则文法通常不是很准确,包含较多热点变量所不可能取的字符串值.另外,由于这一方法未能使用统一的模型描述字符串操作,因此可扩展性较差.Yu 等人[13,14]对基于正则文法的字符串值分析加以改进,引入自动机变换模拟字符串操作,从而避免了对每个字符串操作定义相应的映射.

2.2 基于上下文无关文法的字符串值分析

2005 年,Minamide 在基于正则文法的字符串值分析的基础上,提出了基于上下文无关文法的字符串值分析,即使用上下文无关文法描述热点变量所有可能取值的分析方法[15].该方法使用上下文无关文法作为描述字符串变量值域的符号系统,使用有限状态转换机(finite state transducer)[16]模拟字符串操作.

基于上下文无关文法的字符串分析主要分为 3 个步骤,其中前两步与基于正则文法的字符串分析相同.在第 3 步中,Minamide 提出使用一个有限状态转换机模拟每一个字符串操作,并且对于每一个字符串操作 OP,使用有限状态转换机的转换算法(本质上是一个有限状态自动机与上下文无关文法求交集的算法的变种)[17]构造一个文法 G 通过有限状态转换机之后的文法 G',将 G'加入到原文法 OP 的位置即可消解 OP.通过不断地消解字符串操作,即可将包含字符串操作的字符串操作文法直接转化为一个普通的上下文无关文法.有限状态转换机是一个有输出的有限状态自动机,有限状态转换机的一个重要特性是上下文无关语言经过有限状态转换机的映射后依然是上下文无关语言,很多的字符串操作都可以由有限状态转换机来模拟.例如:对于字符串操作 str_replace(“00”,“0”,$x),可以通过如图 1 所示的有限状态转换机模拟.在图 1 中,a 表示除 0 以外的字符.转换机共有 0,1,2 这 3 个状态,0 状态是初始状态,1 和 2 是终结状态.转换机的状态转移上形如 X/Y 的规则表示当转换机接受输入 X 时会输出 Y.例如,0/0a 表示接受输入 0 并输出 0a.一个输入 00abc11 经过该有限状态转换机后,输出为 0abc11.

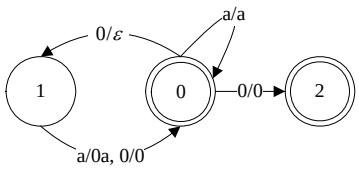

Fig.1 A finite state automaton

图 1 一个有限状态转换机

由于 Minamide 提出的方法使用一个上下文无关文法来近似表示热点字符串变量或表达式的取值,而上下文无关文法较正则文法表达能力更强,因此能够更准确地刻画热点变量的取值.

2.3 基于带一元谓词的二阶逻辑的字符串值分析

2011 年,Tateishi 等人[18]提出了一种基于带一元谓词的二阶逻辑(monadic second-order logic,简称 MSOL)[19]的字符串分析方法.该方法的核心思想是,使用在字符串上定义的带一元谓词的二阶逻辑表达式 M2L(str)作为符号系统描述热点变量的值域,并使用逻辑推导公式模拟字符串操作.M2L(str)的基本组成部分是逻辑表达式‘a’(t),当热点变量的位置 t 上的字符是‘a’时,这一表达式的值为真.根据程序中生成热点变量值的过程对这一基本逻辑表达式进行组合,可以生成热点变量对应的逻辑表达式 $Expr$,使得 $Expr$ 为真的所有字符串即为热点变量的所有可能取值.例如,当 $Expr$=‘a’(1)∧[‘b’(2)∨‘c’(2)]时,即表示该分析方法求出的热点变量可能取值为 ab 或 ac;又如,当 $Expr$=‘a’(x)∧[y>x→‘b’(y)]时,即表示热点变量的取值需以 ab*结尾.该方法主要分为 3 个步骤:首先过滤掉与热点变量之间不存在依赖关系的程序语句,得到精简的程序中间代码;第 2 步,对于常量字符串,直接生成其对应的逻辑表达式;对于从外界输入的字符串变量,由于其取值可以为任意,所以其对应的逻辑表达式为永真;第 3 步,在中间代码中,根据赋值语句右边变量的逻辑表达式求出左边变量对应的逻辑表达式,直到热点变量对

应的逻辑表达式被求出为止.对于字符串操作,该方法提供了每一种操作对应的推导公式,这些公式给出了根据字符串操作数对应的逻辑表达式来计算字符串操作结果对应的逻辑表达式的方法.

由于这一字符串值分析方法能够处理与字符串中的字符下标相关的逻辑表达式,因此这种分析方法能够处理与字符下标相关的字符串操作,例如 IndexOf 操作等.目前,这一方法也是唯一能够处理与字符下标相关的字符串操作的字符串值分析方法.

2.4 字符串值分析方法比较

表 1 给出了现有的字符串值分析方法之间的比较.在表 1 中,分别比较了现有字符串值分析方法的技术特点,包括表示字符串变量值域的符号系统和对字符串操作的处理方式;然后比较了字符串值分析方法的分析准确度,包括是否路径敏感以及对字符下标的处理能力;最后比较了现有方法的主要优缺点.从表 1 中可以看出,在现有的 4 种字符串值分析方法中,大部分方法使用形式语言(正则语言或上下文无关语言)描述字符串变量的取值,Tateishi 等人的方法使用带一元谓词的二阶逻辑描述字符串变量的取值.4 种方法分别使用不同的字符串操作处理方式,但只有 Tateishi 等人的方法是路径敏感的,且可以处理字符下标.基于正则语言的两种字符串值分析方法的主要优点是计算效率高,缺点是正则语言的表达能力较差,结果不够准确.基于上下文无关文法的分析方法计算效率有所降低,但是表达能力有所增强.Tateishi 等人的方法准确度最高,主要的问题是缺乏相应的字符串约束求解方法[20].

Table 1 Comparison of string value analyis approaches

表 1 字符串值分析方法比较

字符串值分析方法		Christensen 等人(2003)	Yu 等人(2008)	Minamide (2005)	Tateishi 等人(2011)
技术特点	表示字符串变量值域的符号系统	正则语言	正则语言	上下文无关语言	MSOL
	字符串操作处理方式	映射规则	自动机变换	有限状态转换机	逻辑推导公式
分析准确度	是否路径敏感 能否处理字符下标	否 否	否 否	否 否	是 是
主要优点		效率高	效率高 易扩展	表达能力强 易扩展	表达能力强 准确度高 易扩展
主要缺点		不易扩展 准确度低 表达能力较差	准确度低 表达能力较差	准确度较低 效率较低	效率低 缺乏复杂约束求解的算法

3 字符串约束求解

字符串约束求解的目的是,对于给定的字符串符号约束,判断是否存在字符串值示例满足这一约束.在使用字符串分析求出字符串变量的可能取值后,经常需要判断这些值是否满足给定的约束,从而达到正确性验证、安全性检测等目的.总体而言,字符串约束求解方法可以分为以下两类.

3.1 基于文法交集求解的字符串约束求解

正则文法求交集的算法以及正则文法与上下文无关文法之间求交集的算法是计算机语言理论方面的基本算法[17].在实际应用中,存在很多与文法包含相关的约束.例如,要判断一个表示 SQL 语句的字符串变量是否可能取值为非法的 SQL 语句,可以约束该变量的取值包含于 SQL 文法推出的语言.由于文法包含关系可以通过求交集判断($A\cap\overline{B}=\varnothing\rightarrow A\subseteq B$)(对于 SQL 文法等无法求补集的上下文无关文法,可以通过限制嵌套层数等得到它们的近似正则文法,并求补集),因此,文法交集求解算法自然就成为了一种字符串约束求解算法.这种算法可以求解文法包含方面的约束.Klarlund 等人根据字符串约束求解的特点实现了这一算法,并开发了学术和产业界广泛使用的工具 MONA[21].

3.2 基于布尔向量约束求解的字符串约束求解

基于文法交集的字符串约束求解的最大缺点就是无法求解与值相关的判断约束,例如字符串相等、字符串长度比较等.针对这一问题,很多学者开始研究基于布尔向量约束求解[22]的字符串约束求解.这一方法的基本思路就是将字符串解码为布尔向量,然后使用布尔向量约束求解算法求解.

HAMPI 是由 Kiezun 等人[23]开发的一个字符串约束求解工具,它允许使用者用给定长度的上下文无关文法和正则文法描述变量和语言,并判断给定的字符串表达式是否包含于约束中的语言.虽然 HAMPI 本身并不能判断字符串相等关系,但是 HAMPI 中可以约束字符串变量和语言的长度,而且 HAMPI 工具使用的基于布尔向量约束求解方法最终解决了字符串相等约束求解等问题.

Bjorner 等人提出了一种名为 WordEqn 的可以求解字符串相等约束的约束求解方法[24].这一方法首先从字符串约束中提取出字符串长度约束.由于长度约束是整数数值约束,可以通过传统的整数数值约束求解方法进行求解.如果长度约束求解得出无解,则对应的字符串约束自然无解;如果长度约束求解有解,则对于长度约束求解的每一个解,可以确定约束中每个字符串变量的长度.然后,约束求解器将每个字符串转换化成为已知长度的布尔向量,并使用布尔向量约束求解的方法求解是否存在满足约束的字符串取值.这一方法有效地解决了字符串相等约束问题,但是这一方法所支持的约束中不能存在限定字符串包含于特定文法的约束项.

Saxena 等人开发了一种名为 kaluza 的字符串约束求解器[25].这一求解器结合字符串长度约束求解和 HAMPI,允许用户对包含字符串长度关系、字符串包含于文法关系和字符串相等等多种关系组成的综合约束进行求解.Kaluza 所基于的方法与 Bjorner 等人 2009 年提出的方法类似,但是加入了对正则文法的支持以及对字符串包含于文法的约束项的支持.

3.3 字符串约束求解方法的比较

表 2 给出了现有的字符串约束求解方法的比较.

Table 2 Comparison of string constraint solvers

表 2 字符串约束求解方法比较

约束求解方法		正则文法求交集	正则文法与上下文无关文法求交集	HAMPI	WordEqn	Kaluza
技术特点		基于文法求交集算法	基于文法求交集算法	基于布尔向量约束求解	基于布尔向量约束求解	基于布尔向量约束求解
支持约束能力	约束中允许的变量数	多个	多个	单个	多个	多个
	约束中允许的判定条件	包含	包含	包含	相等	包含/相等
	约束中允许出现的文法	正则文法	正则文法,上下文无关文法(2)	正则文法,上下文无关文法	字符串集合	正则文法,上下文无关文法
	约束需要给出字符串解的长度上界	否	否	是	是	是
求解时间复杂度		$O(n)$(1)	$O(pn^3)$	NP 难(3)	NP 难(3)	NP 难(3)
主要优点		效率高	效率较高,能处理上下文无关文法	能处理多个文法组成的约束	能处理字符串相等关系	能同时处理文法包含和字符串相等关系
主要缺点		只能处理文法包含关系	约束中只能包含单个上下文无关文法,只能处理文法包含关系	效率较低,需要指定字符串解上界,只能处理文法包含关系	效率较低,需要指定字符串解上界,处理有限字符串集合	效率较低,需要指定字符串解上界

注:(1) n 为正则文法的状态个数,p 为上下文无关文法转换为乔姆斯基范式后的产生式个数,当约束中不涉及正则文法时 n=1;
(2) 只能出现一个上下文无关语言;
(3) 计算复杂度的相关输入均为字符串长度上界.

在表 2 中,首先比较了现有字符串约束求解方法的技术特点;然后比较了现有字符串约束求解方法支持字符串约束的能力,包括支持约束中允许定义的变量数、允许定义的判定条件、允许定义的语言、是否需要给出字符串解的长度上界;最后比较了各方法求解约束的时间复杂度及其优缺点.从表 2 中可以看出,基于文法求交集的两种约束求解方法的计算复杂度为多项式时间,但是最多只允许约束中出现一个上下文无关文法,而且只能求解文法包含关系组成的约束.HAMPI,WordEqn,Kaluza 这 3 种基于布尔向量约束求解的字符串约束求解方法中,Kaluza 的约束求解能力是最强的,能够求解所有其他求解方法所能求解的约束.另外,3 种基于布尔向量约束求解的字符串约束求解方法均需要提供约束解的长度上界,且计算复杂度均为 NP 难,因此当约束的解的长度较长时,无法准确地求解约束.

现有字符串约束求解方法存在的主要问题有:

(1) 现有的字符串约束求解技术都是针对形式语言符号系统进行约束求解,缺乏针对 MSOL 表达式的有效约束求解技术;

(2) 目前,在基于文法求交集的字符串约束求解技术中,约束表达能力有限,最多只允许约束中出现一个上下文无关文法,而其他技术虽然表达能力有所增强,但是效率较低,且需要给出字符串解的长度上界;

(3) 目前的字符串约束求解方法都只针对字符串变量的约束进行求解,无法求解字符串约束和数值约束组成的复合约束.

4 字符串数据来源分析

传统的字符串分析可以确定源代码中热点变量的可能取值,然而对于一些涉及安全的任务(例如源代码安全漏洞检测)中,仅仅获得热点变量的所有可能取值是不够的,还需要了解热点变量的值是否可能来自不安全的来源[26].为了获取热点变量的这一性质,Wassermann 和 Su 在基于上下文无关文法的字符串分析的基础上提出了字符串标记分析[10],用于分析字符串变量是否包含来自不安全的来源(例如用户输入)的子串.字符串标记分析的基本思路是,在字符串操作文法中的终结符和非终结符加入标记,并在文法中传播这些标记,最终根据热点变量对应的非终结符是否被加上不安全的标记来判断其是否包含来自不安全来源的子串.

字符串标记分析首先采用基于上下文无关文法的字符串分析对源代码进行分析,得到对应于某一热点变量的字符串操作文法;然后,对于字符串操作文法中的每个非终结符,如果它来自于用户可以影响的源,则将其标记为危险;接着,字符串标记分析在字符串操作文法中传播这些标记.传播的基本方法是将标记从文法中产生式的右端传播到左端.对于字符串操作的情况,字符串标记分析采用转换机标记传播算法进行处理.转换机标记传播算法就是在计算原文法 G 通过转换机之后的新文法 G'时,将 G 中给定非终结符 NT 上的标记传播到所有新文法中与 NT 对应的非终结符上.

在字符串标记分析的基础上,Wang 等人[27,28]提出了字符串位置标记分析.字符串位置标记分析将字符串标记分析中的标记概念进行了扩展,定义了一种包含字符串数据来源位置信息的标记,并给出了这种标记的传播规则.通过字符串位置标记分析,可以获得热点变量的全部数据来源在源代码中的位置信息,从而定位这些数据来源.

目前的字符串数据来源分析普遍由使用形式语言作为符号系统的字符串值分析扩展而来,因此也存在与这类字符串值分析类似的问题.

5 字符串分析的应用

字符串分析有很多重要的应用,按照应用所涉及的软件工程任务,可以把字符串分析的应用分为 3 类:软件验证方面的应用、软件维护演化方面的应用以及软件测试方面的应用.

5.1 字符串分析在软件验证方面的应用

5.1.1 正确性验证

Gould 等人提出了一种检验动态生成的 SQL 语句正确性的方法[8].这一方法能够检查数据库软件中表示 SQL 语句的变量是否会取不符合 SQL 语法的值,从而避免程序运行时出现数据库 SQL 语句错误.这一方法将数据库软件中表示 SQL 语句的字符串变量作为热点变量,使用基于正则文法的字符串值分析判断通过热点变量的所有可能取值,并判断这些可能取值是否包含不符合 SQL 语法的 SQL 语句.这一判断过程通过使用基于文法交集的字符串约束求解来完成.如果对于一个热点变量,字符串值分析得出对应其所有可能取值的正则文法与经过正则近似的 SQL 文法的补集相交不为空,则该热点变量可能取值为非法 SQL 语句.

Halfond 和 Orso 提出使用字符串值分析在 Web 应用中验证是否存在不匹配的模块接口调用[29].在传统应用中,一个模块的接口是非常容易确定的,在开发环境对程序进行连接时就会自动检查接口是否匹配,并将不匹配的接口报告给开发人员.然而对于 Web 应用软件,确定一个模块的接口本身就是一个比较困难的问题.这是因为 Web 应用软件中的模块是一段服务器端源代码(一般地称其为 Servlet).Servlet 有两种调用方式:第 1 种是直接调用,例如函数调用以及访问该模块的 URL;第 2 种是间接调用,一个模块可以在它所输出的界面中包含一个允许用户访问被调模块的链接,并将部分参数放入这个界面中.当用户点击链接访问被调模块时,则就完成了一次间接调用.间接调用的接口通常是一个用于获得用户输入的表单.对于静态 Web 应用软件,确定用户输入的表单只需要对 HTML 网页作简单的语法分析就可以了;然而对于动态 Web 应用软件,用户输入的表单则比较难以确定.由于模块接口难以确定,接口是否匹配的问题也就成为了一个很难解决的问题.Halfond 等人的方法主要包括如下 4 个步骤:第 1 步使用字符串值分析获取一个模块生成的 HTML 网页的所有可能取值;第 2 步根据 HTML 语法从这些取值中获得可能的表单结构以及表单项名称与类型的集合;第 3 步分析 Servlet 中使用 request 语句读取的表单项集合,并将这些表单项与这个模块的直接调用接口(即函数参数)共同作为这个模块的接口;第 4 步进行接口匹配检查,对程序中所有对该模块的调用进行分析,找出不匹配的接口调用.

Minamide[15]应用基于上下文无关文法的字符串分析检验 Web 应用是否可能产生不正确的 Web 页面.在动态 Web 应用软件中,通常根据用户的输入、外界环境等信息,在服务器端使用服务器端语言(PHP,Java 等)动态地生成浏览器端的 Web 页面.由于动态生成 Web 页面的可能性是无限的,判断一个服务器端程序是否会输出不正确的 Web 页面是一个很重要又不容易解决的问题.Web 应用软件中,复杂的字符串操作使得基于正则语言的字符串分析准确度不足.因此,Minmade 提出使用基于上下文无关文法的字符串值分析对 Web 应用软件的服务器端源代码进行分析,并判断其是否可能产生不正确的 Web 页面.这一方法首先使用基于上下文无关文法的字符串分析获取一个 Web 页面的所有可能取值,这些取值用一个上下文无关文法 G 表示.同时,这一方法设计了一个能够描述非法的 Web 页面的文法模板 F',F'的实质是通过限制嵌套深度进行了正则近似的 HTML 文法的补集.最后求上下文无关文法 G 与这个对应非法 Web 页面的模板 F'的交集,如果交集不为空,则说明服务器端代码可能产生不正确的 Web 页面.

Hirzel 等人使用字符串值分析对 Java 语言中的动态加载对象进行精确的类型分析[30].在 Java 等面向对象语言中,一般允许定义抽象级别最高类型(Java 中为 Object)的变量,并在运行时通过动态加载的方式改变这个变量的类型.然而在很多演化支持中,都需要静态地判断一个变量的类型,因此就需要对这些动态加载对象的可能类型进行静态分析.由于动态加载函数的参数一般为字符串变量,因此这种静态分析需要字符串分析的支持.Hirzel 等人的研究就是将动态加载函数的参数作为热点变量进行字符串值分析,得出这些参数的可能取值,形成一个文法;然后使用文法的成员判定算法[17]判断哪些源代码类型树中的类型名称包含于这个文法,从而判断这个动态加载函数可能加载的类型;最后,这一方法使用指针分析将对于同一变量的多次动态加载操作整合起来,得到变量的可能类型集合.

5.1.2 安全性验证

Wassermann 等人使用字符串标记分析,判断数据库软件中是否存在 SQL 注入漏洞(SQL-injection vulnerability)[10].SQL 注入攻击是最常见的针对数据库应用的攻击方式[18].SQL 注入攻击的基本原理是,输入能

够改变 SQL 语句结构的内容,从而改变 SQL 语句的语义,达到攻击者的目的.例如,当应用程序要求用户输入用户名和密码时,用户如果输入类似“OR ‘a’=‘a’”这种使得 SQL 命令中的条件永真的内容,就可以获得所有用户的信息,达到攻击的目的.通常情况下,开发人员在代码中应当对用户输入的字符串进行检查,过滤掉其中的危险字符(例如上面例子中的 SQL 语言关键字“OR”和“‘”).然而在很多情况下,开发人员所编写的输入检查代码未能成功地过滤掉所有的危险字符,就形成了一个 SQL 注入漏洞.Wassermann 等人的方法将数据库软件中表示 SQL 语句的字符串变量作为热点变量进行字符串标记分析,得到一个带标记的上下文无关文法 G-tag.在带标记的上下文无关文法中,每个非终结符都带有一个标记标明是否来源于安全内容(默认情况下,用户输入的内容为不安全,其他内容为安全);然后,这种方法通过定义一系列模板判断 G-tag 是否可能产生类似“OR ‘a’=‘a’”的不合法的字符串,并判断 G-tag 中是否包含带有来源于不安全内容标记的非终结符,从而可以判定热点变量对应的 SQL 语句是否存在 SQL 注入漏洞.这一方法的最终输出是可能存在 SQL 注入漏洞的热点变量集合.

Wassermann 和 Su 提出了一种基于字符串标记分析的检测 Web 应用中是否存在跨站脚本漏洞(cross-site scripting vulnerability)的方法[31].跨站脚本攻击是指攻击者通过网站允许的输入改变网站本身的语义,使得网站包含攻击者输入的源代码[32].当有其他人访问被修改的网站时,攻击者输入的源代码即可自动地达到攻击者窃取信息等目的.例如,很多类型的网站(BBS 和博客等)都允许一个用户登陆后输入一个留言,之后,留言会被显示在网页中.这时,一个攻击者可以不输入正常的留言内容,而是输入一段可执行的源代码(例如“⟨script⟩email(badboy@bad.com,user.getInfo)⟨/script⟩”),这样,这段源代码就被嵌入到网页之中.如果有其他的用户打开这个网页,他的用户信息就会被自动地发到攻击者的邮箱里.在 Wassermann 和 Su 提出的方法中,首先使用字符串标记分析获取一个 Web 页面所有可能取值的上下文无关文法 G-tag.G-tag 中每个非终结符都带有安全标记(标记其取值是否可能来源于用户);同时,这一方法定义了一个危险标签模板集合 C,C 中的危险标签模板可能导致服务器执行用户输入代码,例如*⟨script⟩*;最后,这一方法对 G-tag 与 C 中的危险标签模板求交集.在求交集的过程中,这一方法附加判断得到的交集上下文无关文法中是否至少有一个非终结符包含的安全标记为不安全(来自用户).如果最终的交集不为空且包含至少一个不安全的非终结符,则认为发现了跨站脚本漏洞,否则认为未发现漏洞.通过区别安全与不安全的非终结符,可以避免将开发人员编写的代码中包含的危险标签(例如⟨script⟩)误认为是跨站脚本漏洞.

5.2 字符串分析在软件维护演化方面的应用

Maule 等人[9]基于字符串值分析提出了一种方法,用于分析数据库应用中的数据库结构变化对源代码中的 SQL 语句产生的影响.这一方法所解决的问题是:判断当一个数据库软件系统对应的数据库的结构发生变化(例如增加表、删除表、增加 1 列、删除 1 列、修改列名等等)时,数据库软件源代码中的哪些 SQL 语句可能会受到影响.在这一方法中,首先以所有作为 SQL 语句传输到数据库的字符串作为热点变量/表达式进行基于正则语言的字符串分析;然后将得到的正则语言与包含数据库中表明和列名的正则模板求交,得出每个 SQL 语句可能涉及的表名和列名;最后,当这些被涉及的表名和列名发生变化时,与其相关的 SQL 语句会被报告给开发人员.

为了定位软件国际化中的待翻译字符串,Wang 等人[27]提出了一种基于字符串位置标记分析方法.一般情况下,待翻译字符串都会出现在软件的 GUI 界面中.基于这一前提,这一方法的主要步骤如下:首先使用 API 搜索引擎在软件源代码中搜索所有向 GUI 界面输出字符串的 API 调用;然后将这些向 GUI 界面输出的字符串实参作为热点变量进行字符串位置标记分析,从而定位这些实参的数据来源,并将所有常量字符串类型的、且满足一定值约束的(例如值中包含字母等)数据来源作为待翻译字符串提交给开发人员.2010 年,针对 Web 软件国际化,Wang 等人[28]又改进了上述方法,进一步分析了用户界面输出的数据来源是否会出现在 Web 软件输出的 HTML 网页的给定结构中,从而解决了 Web 软件国际化中待翻译字符串的定位问题.

5.3 字符串分析在软件测试方面的应用

Halfond 和 Orso[33]提出了一种 Web 应用中的单元测试[34]方法.在传统应用中,单元测试是比较容易自动化实现的一种测试活动.一般情况下,可以根据一个模块(函数、类等)的接口自动化地生成对这个接口的一个调用

或调用序列,然后自动地执行这些调用或调用序列以达到测试的目的.如前所述,对于 Web 应用软件,一个模块的接口是一个较难确定的问题.在 Halfond 和 Orso 提出的单元测试方法中,他们采用了之前提出的基于字符串值分析确定 Web 应用中模块接口的方法[29].在获得 Web 应用源代码块的接口之后,就可以使用与传统单元测试相同的方法对 Web 应用进行单元测试.

Wassermann 等人[35]提出了一种针对字符串类型输入的测试用例生成方法.测试用例生成一般基于动态符号执行[36],动态符号执行首先随机生成一条测试用例并执行,然后求出测试用例执行路径中每一条分支断言的与输入相关的表达式,并通过约束求解求出使该断言不成立的程序输入,从而获得一个新的与上一条测试用例执行路径不同的测试用例.通过不停地迭代上述过程,即可自动生成大量的执行路径互不相同的测试用例.由于传统的测试用例生成方法仅考虑数值表达式与数值约束求解,因此无法针对字符串类型的程序输入生成测试用例,也不能处理判断字符串的程序断言.Wasserman 等人提出的方法使用字符串值分析获得判断字符串的程序断言与字符串类型的程序输入相关的表达式,并使用字符串约束求解求出使得程序断言不成立的程序输入,从而能够生成字符串类型输入的测试用例.

Geay 等人[37]基于字符串值分析对构件进行许可条件分析.许可条件分析是一种求解一个构件在什么条件下能够使测试用例无法满足许可条件.许可条件分析的一般方法是首先分析构件中接口函数的源代码来确定使得接口函数运行拒绝访问分支的条件约束,然后使用符号执行方法[38,39]求出约束条件相关于接口参数的表达式,最后使用约束求解求出使得构件可以访问时参数应当满足的条件.由于传统的符号执行和约束求解不支持字符串类型,而构件的接口参数又往往包含大量字符串类型的参数,因此传统的许可条件分析只能对构件的访问条件作部分分析.Geay 等人首次引入了字符串值分析和约束求解,较好地解决了接口中包含字符串参数的构件的许可条件分析问题.

6 未来研究展望

字符串分析是针对程序中的给定字符串变量,通过分析程序获得该变量的一系列性质的一种程序分析.字符串分析可以有效地支持软件工程领域中的很多任务.本文对现有的字符串值分析方法和字符串约束求解方法进行了比较,简要介绍了最近出现的字符串数据来源分析,并对字符串分析的应用进行了分类介绍.

从本文的介绍中可以看出,字符串分析是一个有意义的研究热点,有丰富的理论价值和应用前景,值得深入研究.现有的研究取得了较好的进展,能够有效地解决一系列软件工程中的实际问题,但在字符串分析本身还存在一些不足之处,而字符串分析在应用方面还有很宽广的研究空间.在字符串分析这一领域主要存在如下一些值得进一步研究的问题:

1. 字符串值分析方面

(1) 在字符串变量值域的表示方面

目前,字符串变量值域的表示主要是基于形式语言或 MSOL 表达式.与形式语言相比,MSOL 表达式的优点是能够较容易地表示字符下标和下标之间的关系,但是得出的值域表达式相对不易求解(逻辑表达式求解至少为 NP 难问题).而比上下文无关语言表达能力更强的形式语言,其性质一般也很难判定.因此,在字符串变量值域的表示方面,可能需要探索新的符号系统,或在现有解决方案上作较大改进,才能获得准确又较易判定的字符串值域表示.

(2) 在字符串操作的处理方面

现有的基于 FST 的字符串操作处理方法计算复杂度较高,可考虑通过一些规则简化某些简单字符串操作处理,从而提高效率.例如,对于单字符的字符串替换操作,可以直接基于规则处理该操作,避免调用字符串操作处理算法.

现有的字符串分析方法都是静态分析,因此对于一些复杂的字符串操作,例如所有参数均为变量的字符串替换操作等,无法提供准确的分析结果.考虑到在实际应用中存在一些针对给定程序路径进行分析的应用场景,例如软件排错,有必要进一步研究能够根据程序运行时信息更为精确地处理字符串操作的技术.

2. 在字符串约束求解方面

目前的字符串约束求解方面的主要问题是:效率较高的基于文法求交集的方法只能支持较简单的文法包含类约束;而基于布尔向量约束求解的方法虽然支持多个文法组成的包含字符串相等关系的较复杂的约束,但却尚无多项式时间算法.因此,需要探索效率较高的支持复杂约束的字符串约束求解方法.

另外,现有的基于 MSOL 表达式表示字符串变量值域的字符串分析方法缺乏相应的字符串约束表示与求解方法,因此无法判定较复杂的字符串取值相关的性质.为了使得该类方法能够处理更为复杂的字符串取值相关的性质,需要进一步研究针对 MSOL 表达式的字符串约束求解方法.

3. 在字符串数据来源分析方面

由于字符串数据来源分析方法在技术上一般依赖于字符串值分析,因此字符串值分析方面的技术进展经过一定的变换也能增强相应的字符串数据来源分析方法.目前,基于 MSOL 表达式的字符串分析方法不支持字符串数据来源分析.如何在 MSOL 表达式中加入标记或其他能够表示数据来源的内容,是一个重要的研究点.

4. 在字符串分析的应用方面

首先,可以使用字符串分析提高通用程序分析的精度.由于通用程序分析无法准确分析字符串操作和字符串变量的取值,因此在处理包含大量字符串变量和操作的程序时会损失很多精度.通过在通用程序分析中使用字符串分析处理字符串变量和操作,可以有效提高通用程序分析的精度.例如,在进行通用程序分支可达性分析时,如果某一分支的断言涉及字符串比较等字符串操作,则可以通过字符串值分析和字符串约束求解来判断该断言是否有可能成立,从而确定对应的分支是否可达.

另外,随着反射、动态类加载等软件新技术的发展以及脚本语言的流行,开发人员越来越普遍地在软件中使用动态代码.所谓动态代码是指在程序中通过字符串的拼接和操作在运行时生成并执行的程序代码.相比于静态代码,动态代码的处理难度大为增加.由于在非运行时根本无法精确地确定某一个字符串变量所表示的所有可能的动态代码,因此现有的程序分析无法对动态代码进行甚至是简单的语法分析,而针对动态代码的大量软件工程任务(动态代码的修改、测试、验证等)都缺乏自动化的支持.

在未来的字符串分析应用研究中,需探索和发展使用字符串分析确定动态代码语法结构的方法,进而对动态代码的修改、测试、验证等提供支持.例如:可以通过定位那些可能出现在动态代码的特定语法结构中的字符串,为动态代码的修改提供支持;可以通过分析动态代码中的变量与程序静态代码中变量的对应关系确定动态代码的接口,从而对动态代码的测试提供支持等.

References:

[1] Ratschiller T, Gerken T. Web Application Development with PHP 4.0. Vancouver: Sams Publishing, 2000. 1−416.

[2] Stiefel M, Oberg R. Application Development Using C# and .NET. Prentice Hall Professional, 2001. 1−623.

[3] Tymann P, Schneider GM. Modern Software Development Using Java. 2nd ed., Stamford: Course Technology, 2007. 1−960.

[4] Welling L, Thomson L. PHP and MySQL Web Development. 4th ed., Boston: Addison-Wesley, 2008. 1−1008.

[5] Moser A, Kruegel C, Kirda E. Limits of static analysis for malware detection. In: Samarati P, Payne C, eds. Proc. of the 23rd Annual Computing Security Applications Conf. Washington: IEEE Computer Society Press, 2007. 421−430. [doi: 10.1109/ACSAC.2007.21]

[6] Cousot P. Abstract interpretation. ACM Computing Surveys, 1996,28(2):324−328. [doi: 10.1145/234528.234740]

[7] Christensen A, Moller A, Schwartzbach M. Precise analysis of string expressions. In: Cousot R, ed. Proc. of the Static Analysis Symp. Heidelberg: Springer-Verlag, 2003. 1−18.

[8] Gould C, Su Z, Devanbu P. Static checking of dynamically generated queries in database applications. In: Estublier J, Rosenblum D, eds. Proc. of the Int'l Conf. on Software Engineering. Washington: IEEE Computer Society Press, 2004. 645−654. [doi: 10.1145/1276933.1276935]

[9] Maule A, Emmerich W, Rosenblum DS. Impact analysis of database schema changes. In: Schafer W, Dwyer M, Gruhn V, eds. Proc. of the 30th Int'l Conf. on Software Engineering. New York: ACM Press, 2008. 451−460. [doi: 10.1145/1368088.1368150]

[10] Wassermann G, Su Z. Sound and precise analysis of Web applications for injection vulnerabilities. In: Ferrante J, McKinley K, eds. Proc. of the ACM SIGPLAN Conf. on Programming Languages Design and Implementation. New York: ACM Press, 2007. 32−41. [doi: 10.1145/1273442.1250739]

[11] Cytron R, Ferrante J, Rosen B, Wegman M, Zadek K. Efficiently computing static single assignment form and the control dependence graph. ACM Trans. on Programming Languages and Systems, 1991,13(4):451−490. [doi: 10.1145/115372.115320]

[12] Mohri M, Nederhof M. Robustness in Language and Speech Technology. Dordrecht: Kluwen Academic Publishers, 2001. 1−268.

[13] Yu F, Bultan T, Cova M, Ibarra O. Symbolic string verification: An automata-based approach. In: Havelund K, Majumdar R, Palsberg J, eds. Proc. of the 15th Int'l Workshop on Model Checking Software. Heidelberg: Springer-Verlag, 2008. 306−324. [doi: 10.1007/978-3-540-85114-1_21]

[14] Yu F, Alkhalaf M, Bultan T. Stranger: An automata-based string analysis tool for PHP. In: Esparza J, Majumdar R, eds. Proc. of the 16th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Heidelberg: Springer-Verlag, 2010. 154−157. [doi: 10.1007/978-3-642-12002-2_13]

[15] Minamide Y. Static approximation of dynamically generated Web pages. In: Ellis A, Hagino T, eds. Proc. of the 14th Int'l Conf. on World Wide Web. New York: ACM Press, 2005. 432−441. [doi: 10.1145/1060745.1060809]

[16] Berstel J. Transductions and Context-Free Languages. Stuttgart: Teubner Studienbucher, 1979. 1−278.

[17] Hopcroft J, Motwani R, Ullman J. Introduction to Automata Theory, Languages, and Computation. 3rd ed., Prentice Hall, 2006. 1−750.

[18] Tateishi T, Pistoia M, Tripp O. Path- and index-sensitive string analysis based on monadic second-order logic. In: Dwyer M, Tip F, eds. Proc. of the Int'l Symp. on Software Testing and Analysis. New York: ACM Press, 2011. 166−176. [doi: 10.1145/2001420.2001441]

[19] Muller D, Schupp P. The theory of ends, pushdown automata, and second-order logic. Theoretical Computer Science, 1985,37(1): 51−75. [doi: 10.1016/0304-3975(85)90087-8]

[20] Bolliq B. Formal Models of Communicating Systems: Languages, Automata, and Monadic Second-Order Logic. Heidelberg: Springer-Verlag, 2010. 1−182.

[21] Klarlund N. Mona & Fido: The logic-automaton connection in practice. In: Nielsen M, Thomas W, eds. Proc. of the Int'l Workshop on Computer Science Logic. Heidelberg: Springer-Verlag, 1998. 311−326. [doi: 10.1007/BFb0028022]

[22] Moura L, Bjorner N. Z3: An efficient SMT solver. In: Ramakrishnan CR, Rehof J, eds. Proc. of the 14th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Heidelberg: Springer-Verlag, 2008. 337−340.

[23] Kiezun A, Ganesh V, Guo P, Hooimeijer P, Ernst M. HAMPI: A solver for string constraints. In: Rothermel G, Dillon L, eds. Proc. of the 18th Int'l Symp. on Software Testing and Analysis. New York: ACM Press, 2009. 105−116. [doi: 10.1145/1572272.1572286]

[24] Bjorner N, Tillmann N, Voronkov A. Path feasibility analysis for string-manipulating programs. In: Kowalewski S, Phillippou A, eds. Proc. of the Tools and Algorithms for the Construction and Analysis of Systems. Heidelberg: Springer-Verlag, 2009. 307−321. [doi: 10.1007/978-3-642-00768-2_27]

[25] Saxena P, Akhawe D, Hanna S, Mao F, McCamant S, Song D. A symbolic execution framework for JavaScript. In: Evans D, Vigna G, eds. Proc. of the IEEE Symp. on Security and Privacy. Washington: IEEE Computer Society Press, 2010. 513−528. [doi: 10.1109/SP.2010.38]

[26] Tripp O, Pistoia M, Stephen F, Sridharan M, Weisman O. TAJ: Effective taint analysis of Web applications. ACM SIGPLAN Notices, 2009,44(6):87−97. [doi: 10.1145/1543135]

[27] Wang X, Zhang L, Xie T, Mei H, Sun J. Locating need-to-translate constant strings for software internationalization. In: Atlee J, Inverardi P, eds. Proc. of the Int'l Conf. on Software Engineering. Washington: IEEE Computer Society Press, 2009. 353−363.

[28] Wang X, Zhang L, Xie T, Mei H, Sun J. Locating need-to-translate constant strings in Web applications. In: Roman GC, Sullivan KJ, eds. Proc. of the ACM SIGSOFT Int'l Symp. on the Foundations of Software Engineering. New York: ACM Press, 2010. 87−96. [doi: 10.1145/1882291.1882306]

[29] Halfond W, Orso A. Automated identification of parameter mismatches in Web applications. In: Harrold MJ, Murphy G, eds. Proc. of the 16th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. New York: ACM Press, 2008. 181−191. [doi: 10.1145/1453101.1453126]

[30] Hirzel M, Diwan A, Hind M. Pointer analysis in the presence of dynamic class loading. In: Odersky M, ed. Proc. of the European Conf. on Object-Oriented Programming. Heidelberg: Springer-Verlag, 2004. 96−122. [doi: 10.1007/978-3-540-24851-4_5]

[31] Wassermann G, Su Z. Static detection of cross-site scripting vulnerabilities. In: Schafer W, Dwyer M, Gruhn V, eds. Proc. of the Int'l Conf. on Software Engineering. New York: ACM Press, 2008. 171−180. [doi: 10.1145/1368088.1368112]

[32] Kirda E, Kruegel C, Vigna G, Jovanovic N. Noxes: A client-side solution for mitigating cross-site scripting attacks. In: Haddad H, ed. Proc. of the ACM Symp. on Applied Computing. New York: ACM Press, 2006. 330−337. [doi: 10.1145/1141277.1141357]

[33] Halfond W, Orso A. Improving test case generation for Web applications using automated interface discovery. In: Crnkovic I, Bertolino A, eds. Proc. of the 6th Joint Meeting of the European Software Engineering Conf. and the 13th ACM SIGSOFT Symp. on the Foundations of Software Engineering. New York: ACM Press, 2007. 145−154. [doi: 10.1145/1287624.1287646]

[34] Hunt A, Thomas D. Pragmatic Unit Testing in Java with JUnit. Raleigh: The Pragmatic Programmers, 2003. 1−163.

[35] Wassermann G, Su Z. Static detection of cross-site scripting vulnerabilities. In: Schafer W, Dwyer M, Gruhn V, eds. Proc. of the Int'l Conf. on Software Engineering. New York: ACM Press, 2008. 171−180. [doi: 10.1145/1368088.1368112]

[36] Godefroid P, Klarlund N, Sen K. DART: Directed automated random testing. In: Sarkar V, Hall MW, eds. Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation. New York: ACM Press, 2005. 213−223. [doi: 10.1145/1064978.1065036]

[37] Geay E, Pistoia M, Tateishi T, Ryder B, Dolby J. Modular string-sensitive permission analysis with demand-driven precision. In: Atlee J, Inverardi P, eds. Proc. of the 31st Int'l Conf. on Software Engineering. Washington: IEEE Computer Society Press, 2009. 177−187. [doi: 10.1109/ICSE.2009.5070519]

[38] Clarke L. A program testing system. In: Gosden J, Johnson O, eds. Proc. of the ACM Annual Conf. New York: ACM Press, 1976. 488−491. [doi: 10.1145/800191.805647]

[39] King JC. Symbolic execution and program testing. Communications of the ACM, 1976,19(7):385−394. [doi: 10.1145/360248.360252]

梅宏(1963－),男,重庆人,博士,教授,博士生导师,中国科学院院士,CCF 会士,主要研究领域为软件工程,系统软件.
E-mail: meih@pku.edu.cn

张路(1973－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件分析与测试.
E-mail: zhanglu@sei.pku.edu.cn

王啸吟(1984－),男,博士,CCF 学生会员,主要研究领域为软件分析,程序理解.
E-mail: wangxy06@sei.pku.edu.cn