计算机科学与探索
Journal of Frontiers of Computer Science and Technology
1673-9418/2019/13(11)-1839-13
doi: 10.3778/j.issn.1673-9418.1812004

Web应用程序搜索功能的组合测试*

吕成成[1],张　龙[2],邓　茜[2],曾凡平[1,3+],严　俊[2],张　健[2]
1. 中国科学技术大学 计算机科学与技术系,合肥 230026
2. 中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190
3. 安徽省计算与通讯软件重点实验室,合肥 230026
+ 通讯作者 E-mail: billzeng@ustc.edu.cn

摘　要:为了方便用户查询感兴趣的资源,许多Web应用程序会提供搜索功能。如果搜索功能存在故障,将会导致Web应用程序的功能异常,甚至会引发安全问题,因而需要对其进行充分的测试。可以使用组合测试的方法生成测试用例测试Web应用程序的搜索功能,其中每一个测试用例是由特殊字符组成的字符串。对于引起系统错误的测试用例,使用组合测试错误定位的方法找到系统错误是由哪些字符组合引起的。使用该方法对学校、政府和事业单位的96个网站进行了测试,发现其中23个网站在搜索某些特殊字符组合时,会引起服务器错误响应。错误定位结果表明,56%的服务器错误响应是由“%”“<”“’”“\”和其他字符的组合引起的。

关键词: Web测试;组合测试;错误定位
文献标志码:A　**中图分类号:**TP311

吕成成, 张龙, 邓茜, 等. Web应用程序搜索功能的组合测试[J]. 计算机科学与探索, 2019, 13(11): 1839-1851.
LV C C, ZHANG L, DENG X, et al. Combinatorial testing of search function in Web application[J]. Journal of Frontiers of Computer Science and Technology, 2019, 13(11): 1839-1851.

Combinatorial Testing of Search Function in Web Application*

LV Chengcheng[1], ZHANG Long[2], DENG Xi[2], ZENG Fanping[1,3+], YAN Jun[2], ZHANG Jian[2]
1. Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China
2. State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
3. Key Lab of Computing and Communication Software of Anhui Province, Hefei 230026, China

Abstract: In order to facilitate users to find resources of interest, many Web applications may provide search function. If there is a failure in the search function, the Web application will run abnormally and even security problems will be caused. So the search function needs to be fully tested. This paper uses the method of combinatorial

* The Frontier Science Key Project of Chinese Academy of Sciences under Grant No. QYZDJ-SSW-JSC036 (中国科学院前沿科学重点项目); the National Natural Science Foundation of China under Grant No. 61772487 (国家自然科学基金); the National Key Research and Development Program of China under Grant No. 2018YFB0803400 (国家重点研发计划).
Received 2018-12-03, Accepted 2019-03-11.
CNKI网络出版: 2019-04-02, http://kns.cnki.net/KCMS/detail/11.5602.TP.20190329.1805.011.html

testing to generate test cases to test the search function of Web application, where each test case is a string of special characters. For test cases that can cause server errors, this paper uses the method of fault location based on combinatorial testing to find the combinations of characters that can cause server errors. This paper tests 96 websites, including schools, governments and institutions, and it is found that 23 of these sites will meet a server error when searching for some special combinations of characters. The experimental result of fault location indicates that 56% of server errors are caused by the combinations of“ % ”“ < ”“ ’ ”“ \ ”and other characters.

Key words: Web testing; combinatorial testing; fault location

1 引言

随着互联网的快速发展,Web应用程序迅速进入普通用户的视野[1]。为了方便用户查询感兴趣的资源,许多Web应用程序会提供搜索功能。浏览器将用户输入的关键词提交给服务器,然后服务器处理该请求并将查询结果返回给用户。如果开发者在实现该功能的时候,对用户输入的合法性没有进行充分的验证,可能会带来安全隐患。如果用户输入的数据中含有某些特殊字符,使得服务器程序无法正常执行,就会产生错误响应,并将异常代码抛到前端浏览器中或者直接在前端显示服务器错误等信息。

例如,图1(a)是某市疾控中心的主页,当搜索关键词“占比%1”时,服务器不会返回正常的搜索结果,而是返回一个只包含错误信息的页面,如图1(b)所示,显示服务器遇到内部错误,服务端的java程序遇到了“java.lang.NullPointerException”的异常,无法完成此请求。Web应用程序中存在很多这样的错误,图2是4种不同的出错页面,都是因为搜索某些特殊字符导致的。这些应用程序错误会导致服务器无法返回正常的搜索结果,同时有些错误页面也会暴露敏感信息,如服务器物理路径、堆栈调用信息、数据库信息等。

此类故障的存在不但会影响用户体验,而且极有可能成为安全问题。比如,攻击者在寻找SQL(structured query language)注入点时,通常会输入某些特殊字符,如果服务器返回错误,很可能可以进行SQL注入,因为这表明Web应用程序中对用户提交的数据未进行合理的验证或过滤[2-3]。同时,错误页面暴露的敏感信息会为攻击者提供极大的便利[4]。因此,对搜索功能的充分测试是非常有必要的。

对于Web应用程序的搜索功能,覆盖全空间测试用例的完全测试几乎是不可能完成的任务,因为搜索空间巨大,无法在有限时间内完全覆盖。而只考虑搜索那些常见内容的测试场景是很难满足测试要求的,因为在实际操作中,用户很可能不小心输错内容,也有很多恶意用户会尝试搜索一些危险且不常用的字符。因此,测试用例要充分考虑各种常用和不常用字符及它们的组合。可以将组合测试的方法应用到Web程序搜索功能上。

在系统测试中,检查系统参数的所有取值组合

(a) Normal page (a) 正常页面　　(b) Fault page (b) 异常页面

Fig.1 Search function display

图1 搜索功能展示

(a) Common fault page1
(a) 常见异常页面1

(b) Common fault page2
(b) 常见异常页面2

(c) Common fault page3
(c) 常见异常页面3

(d) Common fault page4
(d) 常见异常页面4

Fig.2 Web page error message

图2 网页错误信息

来进行充分的测试需要花费很高的代价，因为随着参数数量的增加，测试用例的数量呈指数增长。组合测试[5]是一种有效的黑盒测试方法，其充分考虑到系统中各参数之间的交互作用，进而生成高质量的测试用例。很多应用程序错误是由少数几个参数的相互作用导致的[5]，例如Kuhn和Reilly[6]分析了Mozilla浏览器的错误报告记录，发现超过70%的错误是由某两个参数的相互作用触发的，超过90%的错误是由3个以内的参数互相作用触发的。组合测试（CT）[5]的方法就是选择部分测试用例，可以覆盖任意t（t是一个正整数，且t≥2）个参数可能取值的全部组合。

如果执行完测试用例后，有部分测试用例会引起服务器错误响应，可以使用组合测试错误定位的方法来找到这些字符组合。传统的错误定位是想找到错误的根本原因以及错误在源码中的位置，而本文这里的定位是想找到程序输入中哪些参数及其取值组合会引起系统错误，本文提到的错误定位和错误组合定位都指后者。本文在已有的研究基础上，结合搜索功能测试的特点，提出一种组合测试错误定位方法，可以有效地找到会引起错误组合。

对搜索功能的充分测试可以暴露出服务端程序在处理特殊字符组合时潜在的问题。而错误定位可以帮助开发者修复这些故障，提升网站的形象，提高用户体验，避免恶意用户的攻击。因此，对搜索功能的充分测试是非常有意义的。

总的来说，这篇文章主要有以下三点贡献：

（1）在Raunak等人[7]的基础上，实现了一个使用组合测试方法测试Web应用程序的搜索功能原型工具，并提出一种组合测试错误定位方法，可以有效地找到会引起服务器错误响应的字符组合。

（2）测试了96个学校、政府和事业单位类网站，发现其中23个网站的搜索功能存在问题，当用户搜索某些特殊字符组合时，会引起服务器的错误响应。

（3）对存在问题的网站的错误定位结果进行分析，发现56%的服务器错误响应是由“%”“<”“’”“\”和其他字符的组合引起的。

本文的组织结构如下：第2章介绍组合测试的背景知识；第3章介绍本文提出的方法；第4章介绍实验结果和评估；第5章介绍相关的研究工作；第6章是本文的总结与展望。

2 背景介绍

关于组合测试和错误定位，相关定义[8]如下：

定义1（SUT（software under testing）模型） 一个SUT(k,s)模型有 k 个参数 $p_1,p_2,\cdots,p_k$。 s 是一个长度为 k 的向量，可表示为 $\langle s_1,s_2,\cdots,s_k\rangle$，其中 s_j 表示参数 p_i 可能取值的个数，p_i 的定义域为 $D_i=\{d_{i1},d_{i2},\cdots,d_{is_j}\}$。

定义2（测试用例） 一个测试用例 t 是一个长度

为 k 的向量，可表示为 $\langle v_1,v_2,\cdots,v_k\rangle$，其中 $v_1\in D_1,v_2\in D_2,\cdots,v_k\in D_k$，表示为SUT中每个参数赋予一个确定的值，$p_1=v_1,p_2=v_2,\cdots,p_k=v_k$。

定义3（测试用例集）一个测试用例集 T 是一组测试用例集合 $\{t_1,t_2,\cdots,t_m\}$。

定义4（combinatorial interaction，CI）一个交互组合CI是一个长度为 k 的向量，其中 t 个参数赋予特定的值，剩下的 $k-t$ 个参数未确定(未确定的值用-表示)。这里 t 表示CI的大小。例如，长度为 k 的向量 $\langle v_1,-,\cdots,-\rangle$ 可表示一个大小为1的CI，其中 $v_1\in D_1$，它表示参数 $p_1=v_1$，剩下的 $k-1$ 个参数未确定。

定义5（CI包含关系）一个CI P_1 被另一CI P_2 包含（contain），当且仅当 P_1 的所有赋值的参数在 P_2 中也被赋值，并且这些参数在 P_1 和 P_2 中被赋予了相同的值。一个CI P 被测试用例 T 包含，当且仅当 P 中的所有赋值的参数具有与 T 中相同的值。

定义6（faulty combinatorial interaction，FCI）一个错误交互组合FCI是一种特殊的CI，所有包含它的测试用例都会失败。

如果 P_i 是FCI，其他任何包含 P_i 的CI也是FCI。比如 P_1 是一个FCI，且 P_2 包含 P_1，所有包含 P_2 的测试用例也都将包含 P_1，因此这些测试用例都会失败，即 P_2 也是FCI。组合测试的错误定位是想找到那些最小FCI（minimal FCI），最小FCI不包含任何比它小的FCI。

组合测试的错误定位主要可以分为两种，非自适应方法和自适应方法。非自适应的方法是指所有的测试用例可以并行执行，不依赖于先前测试用例的执行结果。Colbourn和McClary[9]提出了一种错误定位方法，他们通过构造Locating Array的方法来找到系统中的最小出错组合。Zhang等[8]提出了一种基于约束求解和优化技术的方法，无需生成额外的测试用例就可以找到最小出错组合。自适应的方法是指部分测试用例的选择是基于先前测试用例的执行结果。Zeller和Hildebrandt[10]提出了一种典型的自适应方法。这种方法的主要思想是通过修改输入参数来找到最小出错组合。对于失败的测试用例，修改其中部分参数的值；如果修改后的测试用例仍然是失败的，则修改的参数与系统错误无关；否则，修改的参数与系统错误有关。

3 研究方法

图3是整体方法的流程图。主要分成测试用例生成、测试用例执行和错误组合定位三部分。下面详细介绍各部分内容。

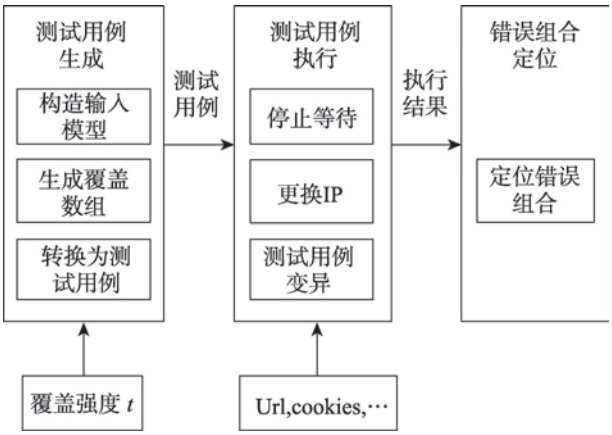

Fig.3 Workflow of method in this paper

图3 方法流程图

3.1 测试用例生成

本文使用组合测试的方法来生成测试用例。组合测试是一种有效的测试方法，可以应用于网站搜索功能的测试。可以根据组合覆盖率生成测试用例，测试强度为 t（t 为正整数，且 $t\geqslant 2$）的组合测试可以保证任何参数值的 t 组合至少被一个测试用例覆盖。组合测试用例的构造方法，多数是基于覆盖数组的。组合测试设计过程可简要描述如下：

（1）建立输入模型（input model），模型包括有哪些参数以及每个参数的可能取值。

（2）根据输入模型，生成覆盖数组。

（3）将生成的覆盖数组转化为测试用例，覆盖数组的每一行表示一个测试用例。

步骤（1）的输入模型将在下面详细介绍。步骤（2）使用组合测试用例生成工具ACTS[11]（advanced combinatorial testing system）来生成覆盖数组。ACTS由NIST（National Institute of Standards and Technology）和UTA（University of Texas at Arlington）联合开发，

实验表明，与类似工具相比，在覆盖度 t 相同的情况下，ACTS生成的测试集较小，并且比其他类似工具更快，比如AETG（http://aetgweb.argreenhouse.com/）。步骤（3）将覆盖数组转化为测试用例并执行，覆盖数组中每一行是一个测试用例，每一个测试用例是一个由特殊字符组成的字符串。最后，通过组合测试错误定位的方法来找到是哪些特殊字符的组合引起的服务器错误响应。各部分方法的详细介绍如下。

从用户提交输入到得到返回页面服务端程序对用户输入可能的处理操作有：

（1）数据的编码和解码操作，如果用户的输入不满足特定的参数格式，可能会引起错误（如URL（uniform resoure locator）解码，可以考虑%与其他字符的组合）。

（2）过滤转义操作：考虑到XSS（cascading style sheets）、SQL注入等安全问题，很多网站会对用户输入进行过滤和转义，因此需要在输入模型中加入部分XSS、SQL元素。

（3）查询操作：服务端程序可能会将用户的输入拼接成数据库查询语句，当存在某些特殊字符时，可能会改变查询语句的语义从而引起错误。

可以构造输入模型如表1所示，共有7个参数。参数的选择和取值并不是唯一的，开发者可以根据实际的需求来构造模型。这里借鉴Bozic等[12-13]构造XSS攻击向量的方法，取7个参数，各参数分别表示为：

参数1 模拟输入标签的结束符号。比如SQL查询中，待比较的字符串应该填充在两个单引号之间，参数1选择单引号可以模拟SQL语句中单引号作用区间的结束，使后面的字符组合可以当作代码被执行。

参数2 表示打开一个作用区间，比如“{”等，可以与后面的关闭标记“}”组成一个作用区间。

参数3、4、5 表示在参数2和参数6构成的作用区间中特殊字符组合，比如英文字母“aBcD”、数字“22”、标点符号“?”、运算符号“+”、其他特殊字符“&”等。

参数6 关闭由参数2打开的作用区间，比如“}”，可以与前面的打开标记“{”组成一个作用区间。

参数7 表示在输入末尾有特殊作用的关键词，比如在SQL语句中有注释作用的“#”。

该模型可表示为一个SUT (7, <7,9,9,8,6,6,5>)。从每个参数的取值范围中选择一个值组成一个用于搜索的字符串，每个参数分别有7、9、9、8、6、6、5个可能取值，这样一共有816 480种不同的字符串，而使用ACTS构造的覆盖强度为2的测试集只需要81个测试用例。

Table 1 Input model

表1 输入模型

参数	可能取值
1	)、}、“、‘、>、space、NULL
2	(、[、{、<、$、%、11、space、NULL
3	script、aBcD、22、!、?、~、/、space、NULL
4	|、&、-、/、\、,、.、NULL
5	^、+、=、*、space、NULL
6	>、)、]、}、space、NULL
7	#、“ \\、space、NULL

在这个输入模型中，space代表空格，NULL表示不取一个空字符。其中，数字和字母组合有aBcD、script、11等；特殊字符有(、)、[、]、{、}、<、>、”、’、$、%、!、?、~、/、|、&、-、\、,、.、^、+、=、*、#，包括运算符号、标点符号等。

3.2 测试用例执行

通过组合测试工具生成测试用例，每一个测试用例是一个由特殊字符组成的字符串。对于每一个测试用例，通过构造URL来模拟查询字符串，如果返回正常的页面，则认为此测试用例的执行结果为通过，如果服务端直接将异常代码抛到前端浏览器，或者在前端浏览器显示服务器错误，就认为此测试用例的执行结果为失败。

在实际测试中，可能遇到以下两种情况。（1）IP限制，即如果检测到同一IP在短时间内发送大量请求，则限制该IP的访问；（2）存在部分无效测试用例，可能的原因之一是Web应用程序内部的防御机制。如果用户的输入满足防御机制内置的安全规则，则重置链接或拒绝执行此请求[14]。

对于第一种情况，采用停止等待或者更换IP的方法，同时这也要求本文的测试用例规模不能太大。

第二种情况中，部分测试用例可以得到执行结果，而部分测试用例无法得到执行结果，是无效的。无效测试用例中所包含的字符组合会因为测试用例

无效而无法被覆盖到,这种行为在组合测试中称为masking effect[15]。造成部分测试用例无效的原因是由于包含了部分字符及其组合,可以生成额外的测试用例来解决这个问题。

发现一个测试用例 $t=\langle v_1,v_2,\cdots,v_k\rangle$ 是无效的,使用Simplified-One-Factor-One-Time[16]的方法对测试用例进行变异,构造 k 个额外的测试用例,$\langle *,v_2,\cdots,v_k\rangle$,$\langle v_1,*,\cdots,v_k\rangle$,$\cdots$,$\langle v_1,v_2,\cdots,*\rangle$,其中*表示一个随机的并且和原测试用例不相同的值。这样因为原测试用例无效而无法被覆盖到的组合可以重新被覆盖到。

比如,对于一个目标网站,有测试用例"'{script|^># ",在执行该测试用例时,因为"'"的出现该字符串满足此网站的安全规则,服务器拒绝执行该请求,则该测试用例无效。此时,无法确定其他字符及组合是否会引起服务器内部错误,实际上,"{script"是会引起服务器错误的。此时,可以使用一阶变异的方法额外生成7个测试用例,"{script|^>#""'script|^>#""' {|^>#""' {script^>#""' {script|>#""' {script|^#""'{script|^>"。执行这7个测试用例,其中"{script|^>#"这个字符串会引起服务器内部错误。这样,就不会因为该测试用例无效而使错误组合"{script"无法被发现。

3.3 错误组合定位

对于存在错误的Web应用,通过组合测试错误定位的方法来找到是哪些特殊字符的组合引起的服务器错误响应。使用组合测试生成字符串进行测试时不需要全部参数,只使用部分参数,也能构造一个测试用例,可以将这个特点应用到错误定位中。

Raunak等[7]使用了Colbourn和McClary[9]提出的一种非自适应的组合测试错误定位方法。在该方法的基础上,提出一种新的错误定位方法。该方法的算法过程如算法1所示。

算法的第一步是得到一组可疑的 *CI* 集合。算法的输入是一组测试用例的执行结果,*Fail_T* 表示失败测试用例集,*Pass_T* 表示通过测试用例集,输出是 *FCI* 集合(出错组合)。算法的第一行将 *Minimal_FCI* 设置为空集,它将用来保存找到的 *FCI* 。算法的第2行,是找出可疑 *CI* 集合。如果一个 *CI* 出现在失败测试用例中,它可能是一个 *FCI* ,如果一个 *CI* 出现在通过的测试用例中,它一定不是 *FCI* 。这样,那些出现在失败测试用例中,但没出现在通过测试用例集中的 *CI* 就是可疑 *CI* 。用 *Fail_T* 包含的全部 *CI* 减去 *Pass_T* 中包含的全部 *CI* ,则会删除一定不可能是 *FCI* 的 *CI* ,但剩下的 *CI* 不一定就是 *FCI* ,*Susp_CI_Set* 表示这些可疑的 *CI* 集合。

算法的第二步是从可疑 *CI* 集合中得到最小出错组合(minimal FCI)。本文和Raunak等[7]的方法有所不同。Raunak的方法假设 $P_i,P_j\in$ *Susp_CI_Set*,且 P_i 包含 P_j。P_i 和 P_j 都能引起错误,但 P_i 不是最小出错组合(minimal FCI),需要将 P_i 删除。因此Raunak将 *Susp_CI_Set* 包含其他 *CI* 的 *CI* 删除,得到的就是最终结果。这种方法的有效性不高,因为会存在部分 *CI* 虽然只出现在了失败测试用例中,但并不能导致服务器错误,不能将其当作 *FCI* ,因此也就不能将包含此 *CI* 的其他 *CI* 直接删除。

为提高定位的有效性,本文从算法的第3行到第11行,确认 *Susp_CI_Set* 中可疑的 *CI* 是否为 *FCI* 。因为希望找出最小出错组合,所以从大小为1的 *CI* 开始确认,如果这个 *CI* 构成的测试用例执行结果为失败,它就是 *FCI* ,将其加入到 *Minimal_FCI* ,如算法的第7行所示。因为包含此 *CI* 的其他 *CI* 也是 *FCI* ,但不是最小出错组合,在算法的第8行,将 *Susp_CI_Set* 中包含此 *FCI* 的其他 *CI* 删除。确认完大小为 k 的 *FCI* 后,返回 *Minimal_FCI* 。此时,*Minimal_FCI* 就是错误定位到的最终结果。

算法1 组合测试错误定位

输入:失败测试用例集 *Fail_T*,通过测试用例集 *Pass_T*。

输出:*FCI* 集合。

1. *Minimal_FCI* = ∅
2. *Susp_CI_Set* = *Fail_T* 包含的所有CI-*Pass_T* 包含的所有 *CI*
3. for each $s_i\in\{1,2,\cdots,k\}$ do
4. *Tmp_Susp_CI_Set*=*Susp_CI_Set* 中大小等于 s_i 的 *CI*
5. for each $CI_j\in$ *Tmp_Susp_CI_Set* do
6. If CI_j 的执行结果为失败 then
7. *Minimal_FCI* = *Minimal_FCI* ∪ {CI_j}

8. 删除*Susp_CI_Set*中包含CI_j的其他*CI*
9. end if
10. end for
11. end for
12. return *Minimal_FCI*

4 实验结果与分析

在python 3.5.4环境下实现了一个原型工具，可以实现本文提到的全部方法。使用该工具测试实际中的Web应用程序。

360发表的《2017中国网站安全形势分析报告》(http://zt.360.cn/1101061855.php?dtid=1101062368&did=490995546)显示，教育培训、政府机构和事业单位这3个行业的网站是检测出网站漏洞最多的行业。搜索功能的测试和网站的安全关系密切，而且政府、学校等网站是面向大众的，具有特定权威、严肃等属性，这些网站出现故障将极大影响公共安全和其公信力。从安徽、北京、陕西等地的学校，政府机构和事业单位网站中收集了96个提供搜索功能的网站，其中有：48个学校类网站，包括12个研究生院网站，3个大学网站和5个中学网站；40个政府机构网站，包括13个省政府网站和27个市政府网站；8个事业单位类网站。在这96个网站中，有76个使用get方法发送http请求，有20个网站使用post方法发送http请求。

4.1 实验结果

使用组合测试的方法测试了这96个网站，实验结果发现，96个网站中共有23个网站的搜索功能存在问题，实验结果如表2所示。其中在8个事业单位类的网站中，有3个网站存在错误，占比最高为37.5%；48个学校类的网站中存在错误的网站有15个，占为31.25%，有50%的研究生院网站存在错误，相比于大学和中学网站，研究生院的网站更容易检测到错误；40个政府机构网站中存在错误网站有5个，占比最小，为12.5%，市政府网站比省政府网站更容易检测出错误。

在这96个网站中，使用get方法的76个网站中，有14个网站存在错误，占比为18.4%。使用post方法的20个网站中，有9个网站存在错误，占比为45.0%。在本次实验中，使用post方法发送http请求的网站更容易检测出错误。

Table 2 Results of Website testing

表2 网站测试结果

行业	属性	网站总数	存在错误的网站总数	存在错误的网站占比/%
学校	研究生院	12	6	50.0
	大学	31	8	25.8
	中学	5	1	20.0
	合计	48	15	31.3
政府机构	省政府	13	1	7.7
	市政府	27	4	14.8
	合计	40	5	12.5
事业单位		8	3	37.5
合计		96	23	24.0

对搜索功能存在错误的22个网站进行错误定位(剩下的一个网站，虽然检测到了错误，但因为它对ip的限制严格而无法错误定位)，来确定是哪些字符组合引起的服务器错误。共发现112种不同的FCI，对应着112种不同的会引起网站服务器错误的字符组合。这112种FCI的大小的分布图如图4，可见大小为2的FCI共有65种，占比最大，93.75%的FCI值小于3。

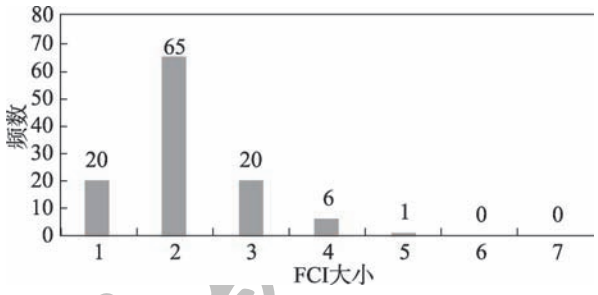

Fig.4 FCI size distribution

图4 FCI大小分布图

根据这些会引起服务器错误的字符组合的频数进行了统计，统计结果如表3所示。频数最多字符组合共出现了11次，是“%!”“%?”“%/”“%~”，并且56%的服务器错误都是由“%”“<”“’”“\”和其他字符的组合引起的。根据部分网站的应用程序错误信息，分析这些字符组合引起服务器错误的可能原因如下：

“%”：“%”通常会用来做URL编码，Web程序将用户提交的输入直接通过get或post方法提交到服务器，服务器程序会先对输入做URL解码，此时包含“%”并且不符合URL编码规则的输入就会引起程序

Table 3 Result of fault location

表3 错误定位结果

频数	字符组合								
9,10,11	%/	%!	%?	%~	%sp				
6,7,8	%	</	<aBcD	%script	<!	<?			
4,5	sp%	%\	<script	}%	'%	script\			
3	‘	\\	sp\	aBcD\					
2	!	sp	"	(	-	\	^	?	>
	[	]	)	{	}	~	?|	11!	!&
	=}	!-	/^	{!	>{	}sp	sp{	-^	{/
	>"	^>	|^	>%	?,	22\	}\\	"$	sp!
	"sp	^}	{?	'{	sp-	?/	?-	?\	.^
	="	"<	?&	$!	sp?	{aBcD	-sp	{sp	"11
	{22	!\	.sp}	^sp\\	$aBcD|	/_}	11sp\	sp,^	=sp"
	?._								
1	&	<	)%	&^	]\\	sp"	sp#	spsp	sp&
	&=	}{	^sp#	{script	"<?	sp{!	"!,	'sp?	<?>
	spsp"	<aBcD\	){sp/	/-sp)	</-sp	<!/sp	<scriptsp	)/sp)	}(sp|}

崩溃,造成服务器错误响应。

“<”:“<”是一个常见的html标签,和XSS注入关系密切。在asp.net程序中,如果检测到用户提交的输入中含有“<!”等危险输入,就会停止运行,从而造成服务器的错误响应。

“ ”:SQL查询中,待比较的字符串应该填充在两个单引号之间,如果用户提交的输入中含有“’”,且服务端程序直接将用户输入拼接成SQL语句进行查询,就会导致SQL查询的异常,从而造成服务器的错误响应。一般来说,如果含有“’”的字符组合可以引起服务器的错误响应,则该网站有很大的可能存在SQL注入的漏洞[2-3]。

“\”:在正则表达式中以及SQL语句中表示转义字符,表示下一个字符为特殊字符。服务端程序在处理含有“\”的输入(比如java程序中的replaceAll()方法)或者进行SQL查询时,没有考虑到“\”的特殊含义,可能会引起服务器的错误响应。

4.2 错误网站案例

同时,也将本文的测试方法应用到部分商业网站的测试中。在一个招聘网站的测试中检测到了问题,下面详细介绍本文的测试过程。分别执行覆盖强度为2、3、4的测试用例,执行结果如表4所示。使用错误定位的方法来找到引起系统错误的字符组合,定位结果如表5所示。

Table 4 Results of test case execution

表4 测试用例执行结果

覆盖强度	测试用例数量	失败测试用例数量	失败测试用例占比/%
2	81	6	7.4
3	725	57	7.9
4	5 459	412	7.5

Table 5 Results of fault location of recruitment Website

表5 招聘网站的错误定位结果

覆盖强度	定位结果									
2	}%	‘%	sp%	%!	%?	%NULL	%/	%sp	%~	
3	%!	%?	%~	%/	%sp	%NULL				
4	%!	%?	%~	%/	%sp	%NULL				

可以发现,覆盖强度2、3、4生成的测试用例中失败测试用例的比例基本相似,覆盖强度3、4的错误定位结果是相同的,这些错误组合主要是由%和其他字符的组合引起的。可能的一个原因是包含特殊字符组合(比如“%!”“%?”)的输入会因为不符合URL解码函数的输入而引起了“Internal Server Error”的响应。

4.3 测试用例变异方法评估

在实际情况中,部分Web应用程序在测试时存在masking effect情况。此时,采用Simplified-One-

Factor-One-Time的测试用例变异的方法生成额外的测试用例来解决这个问题。在96个测试网站中，检测到有26个网站存在这样的情况，其中有3个网站存在错误。无效测试用例的存在使得某些错误组合无法被覆盖到。比较3个存在无效测试用例且搜索功能存在错误的网站的错误定位结果，发现有两个网站在执行完变异测试用例后定位到的错误组合数比不执行变异测试用例定位到的错误组合数多1。由此可见，使用测试用例变异的方法可以使错误定位得到的结果更全面。

4.4 错误定位方法评估

Raunak等[7]使用Colbourn和McClary[9]提出的组合测试错误定位方法，将这种方法记作Method_A。本文的方法记作Method_B。主要考虑两方面：(1)错误定位方法是否可以找到所有的错误组合；(2)错误定位方法找到的结果是否准确。从“查全率”和“查准率”两方面来分析两种组合测试错误定位方法在Web应用程序搜索功能测试上的有效性。在15个存在错误的学校类网站中，有6个网站存在IP限制和无效测试用例，以剩下的9个学校类网站为测试对象。

“查全率”是指错误定位方法能不能找到所有的错误组合。可以将定位到的FCI转换为约束加入到输入模型中，重新生成测试用例，使得生成的测试用例不包含任何定位到的FCI。例如，有SUT $(3,\langle 2,2,2\rangle)$，它的3个参数为 $p_1\{0,1\}$、$p_2\{0,1\}$、$p_3\{0,1\}$，定位得到的FCI为$\langle 0,1,-\rangle$、$\langle 0,-,1\rangle$、$\langle -,0,0\rangle$。此时可以生成约束(约束定义方式使用ACTS[10]工具的定义标准)如下：

[Constraint]

$!(p_1=0\ \&\&\ p_2=1)$

$!(p_1=0\ \&\&\ p_3=1)$

$!(p_2=0\ \&\&\ p_3=0)$

则加入此约束生成的所有测试用例都不包含FCI $\langle 0,1,-\rangle$和$\langle 0,-,1\rangle$。使用错误率 E 来表示失败测试用例数量占全部测试用例的比例。其中：

$$\text{错误率}=\frac{\text{失败测试用例数量}}{\text{全部测试用例数量}}\times 100\%$$

两种错误定位方法的“查全率”执行结果如表6所示。可以看到，在9个被测Web应用中，有8个Web应用在将Method_A定位得到的FCI以约束的形式加入到模型后生成的测试用例的执行结果为全部通过，7个Web应用在将Method_B定位得到的FCI以约束的形式加入到模型后生成的测试用例的执行结果为全部通过。将覆盖强度增加到4时，其余两个Web应用的执行结果也全部为正确。这说明两种定位方法可以全面地找到被测系统中的错误组合。

“查准率”是指定位得到的字符组合中确定会引起服务器错误的字符组合所占的比例。每次从定位结果中取出一个FCI，在输入模型中加入约束，使得重新生成测试用例全都包含此FCI而不包含任何其他的FCI。例如，有SUT $(3,\langle 2,2,2\rangle)$，它的3个参数为 $p_1\{0,1\}$、$p_2\{0,1\}$、$p_3\{0,1\}$，定位得到的FCI为$\langle 0,1,-\rangle$、$\langle 0,-,1\rangle$、$\langle -,0,0\rangle$。对于FCI $\langle 0,1,-\rangle$，可以生成约束如下：

[Constraint]

$(p_1=0\ \&\&\ p_2=1)$

$!(p_1=0\ \&\&\ p_3=1)$

$!(p_2=0\ \&\&\ p_3=0)$

则加入此约束后生成的测试用例全都包含FCI $\langle 0,1,-\rangle$，而不包含其他任何FCI。对于FCI $\langle 0,-,1\rangle$，可以生成约束如下：

[Constraint]

$(p_1=0\ \&\&\ p_3=1)$

$!(p_1=0\ \&\&\ p_2=1)$

$!(p_2=0\ \&\&\ p_3=0)$

则加入此约束后生成的测试用例全都包含FCI $\langle 0,-,1\rangle$，而不包含其他任何FCI。

对于选中的FCI，使用这种方法重新生成的所有测试用例都将包含此FCI且不包含任何定位到的其他FCI。如果所有的测试用例执行结果都引起服务器的错误响应，就认为此FCI为有效的FCI，否则是无效的。将定位得到的所有FCI中有效的FCI所占的比例记作“查准率”。

两种错误定位方法的“查准率”执行结果如表7所示。使用Method_A定位得到的错误组合的数量比较多，但平均只有7.4%的错误组合是有效的。使用Method_B定位得到的错误组合中，平均有77.2%的错误组合是有效的，而且有5个网站的定位结果是100%有效的。这说明相比于Method_A，Method_B能够更为准确地找到被测网站中的错误组合。

Table 6 “recall” of fault location method

表6 错误定位方法“查全率”

Web	覆盖强度 t	未加入FCI约束结果		加入FCI约束Method_A		加入FCI约束Method_B	
		测试用例数量	错误率/%	测试用例数量	错误率/%	测试用例数量	错误率/%
uir	2	81	4.9	87	0	82	0
cupl	2	81	6.2	87	0	84	0
csu	2	81	6.2	87	0	84	0
gscass	2	81	6.2	87	0	84	0
wenda	2	81	7.4	84	0	80	0
ahau	2	81	8.6	86	0	80	0
nwsuaf	2	81	11.1	86	0	77	1.3
uibe	2	81	8.6	87	2.3	80	0
	3	725	9.1	727	1.0		
	4	5 459	8.8	5 332	0		
bnu	2	81	8.6	87	2.3	80	0
	3	725	9.1	727	1.0		
	4	5 459	8.8	5 332	0		

Table 7 “precision” of fault location method

表7 错误定位方法“查准率”

Web	Method_A FCI数量	Method_A “查准率”/%	Method_B FCI数量	Method_B “查准率”/%
uir	58	6.9	4	100.0
cupl	72	6.9	5	100.0
csu	72	6.9	5	100.0
gscass	72	6.9	5	100.0
wenda	61	6.6	5	20.0
ahau	95	7.4	12	58.3
nwsuaf	126	5.6	15	46.7
uibe	95	7.4	7	100.0
bnu	126	5.6	10	70.0
平均	86.3	7.4	7.6	77.2

同时注意到，有一些网站，它们执行相同覆盖强度生成的测试用例的结果相同，比如csu和cupl，uibe和buct，虽然它们的前端显示页面不同，但后端处理程序很可能是相同的。

实验结果表明，本文的方法实现的工具可以有效地测试Web应用程序的搜索功能，本文的错误定位方法可以准确地找到出错字符组合。但测试的方法仍有不足，有些网站可能并不会让错误信息在前端浏览器上显示，因此无法判断本文的测试用例是否会引起服务器内部错误。开发者可以通过读取系统日志来解决此问题。

5 相关工作

对于许多复杂的软件系统，通常有不同的组件、选项或参数相互作用。对于这样的系统，组合测试是一种非常有效的黑盒测试技术，它可以应用于不同的测试级别，如单元测试、集成测试和系统测试。最近，有不少研究者将这种测试方法应用到Web测试和安全测试。

使用组合测试来检测Web应用程序的搜索功能是否存在错误，而Bozic和Simos等[12-13,17]则将组合测试以及组合测试错误定位的方法应用到XSS攻击检测上。Bozic使用不同组合强度生成的字符串测试用例对3个Web程序进行测试，结果显示组合测试的方法十分有效。在接下来的研究中，他们将约束条件加入到组合测试模型中，能显著提高测试用例的质量。Simos等[17]在Bozic的基础上，继续研究组合测试在XSS攻击上的应用。他们使用组合测试错误定位的方法来识别XSS诱导组合，XSS诱导组合是输入参数值的组合，任何包含此诱导组合的测试用例一定会在运行时成功触发XSS漏洞。XSS诱导组合的识别有助于更好地理解XSS漏洞的根本原因，帮助安全人员设计有效的过滤函数来避免这些漏洞。

Raunak等[7]将组合测试的方法应用到Web应用程序的搜索功能测试上。在国家漏洞库（National Vulnerability Database，NVD）网站开发实现新功能后，开发人员发现某些特殊字符会导致“服务器错误”

响应。然而，不清楚哪些特殊字符的特定组合触发了这种响应，以及还存在多少这样的问题。他们使用组合测试的方法生成测试用例，并用错误定位的方法找到出错组合，最后发现了49个输入会引起“服务器错误”响应。但是，使用该错误定位方法找到的字符组合中有很多是无效的。本文的错误定位方法将每个可疑组合转化成测试用例执行，这样可以更有效地找到出错字符组合。

Qi等[18]提出了一种带约束的成对测试（覆盖强度为2的组合测试）算法，称为PTC（pairwise testing with constraints），以充分测试网站的表单交互过程。PTC算法使用组合测试生成需要提交的表单数据，并能够通过增加约束的方式处理语义约束和非法值的问题。根据算法实现了一个原型工具ComjaxTest，它能够系统地探索Web应用程序的状态空间。实验结果表明，采用PTC算法配置的ComjaxTest实现了动态网页的高覆盖率，并在一定时间内能检测出更多的错误。他们的测试用例是基于表单数据生成的，目的是达到高的网页覆盖率，而本文的测试用例是基于搜索的关键字生成的，目的是为了引起更多的服务器错误响应。

Wang等[19]基于组合测试提出了一种新的用于检测缓冲区溢出漏洞的黑盒测试方法。他们根据程序的外部参数生成测试用例，模拟攻击者利用缓冲区溢出漏洞的过程来检测漏洞。他们将这种方法应用到5个开源程序上，可以有效地检测这些程序中的缓冲区溢出漏洞。他们利用组合测试通常能达到高代码覆盖率的事实，可以让程序能够到达攻击点，而本文使用组合测试是因为组合测试生成的测试用例可以覆盖更多的字符组合。

相比于组合测试，模糊测试在Web程序安全测试方面有着更多的应用。Tripp等[20]提出一种Web应用程序安全测试工具XSS Analyzer。XSS Analyzer根据语法规则生成XSS攻击向量，并从未成功注入的攻击向量中学习语法规则中的约束，即攻击向量中不能包含哪些单词，从而绕过Web应用程序的检测。XSS Analyzer定位攻击向量中的无效元素的方法与本文方法不同，XSS Analyzer首先将攻击向量分解成一组单词，并将每个单词发送到目标Web程序，确认哪些单词是无效的。这种做法的有效性不高，无法解决多个单词组合出现时才会无效的情况，而且将每个单词都发送到目标Web程序会造成大量的http请求。

Appelt等[14]将机器学习和进化算法相结合，提出一种算法ML-Driven，可以绕过防火墙进而注入SQL语句。该方法可以自动生成测试用例并发送到防火墙，检测它们是否可以成功绕过防火墙。根据测试用例的执行结果，使用机器学习的算法来选择最有可能绕过的测试用例进行测试和变异。在实际测试中也会遇到防火墙的情况，但与ML-Driven方法不同，本文的目的是为了能覆盖到更多的字符组合，因此选择生成额外的测试用例来解决这个问题。

通过构造URL并动态执行的方法来判断该URL的执行结果，同时也可以通过分析这些URL的特征来做出判断。Sheykhkanloo[21]则使用深度学习的方法来判断。他们将SQL标签加入到正常的URL中，使之成为一个恶意的SQL注入的URL，并根据这些正常的URL和恶意的URL训练深度学习模型，该模型可以有效地检测URL为正常的或恶意的，并可以识别URL的注入类型。Rathore等[22]则提出一种基于机器学习的检测SNS（social networking services）XSS漏洞的方法。他们在选择特征时不只考虑到了URL的特点，同时也考虑到了生成的网页中html元素以及SNS特征的影响。

6 总结与展望

将组合测试的方法应用到Web应用程序的搜索功能，并结合搜索功能测试的特点，提出一种组合测试错误定位方法来找到具体是哪些特殊字符的组合会引起系统错误。实现了一个原型工具，并对学校、政府类和事业单位的96个网站进行了测试，其中23个网站在搜索某些特殊字符组合时，会引起服务器错误响应。错误定位的结果表明，56%的服务器错误响应是由“%”“<”“’”“\”和其他字符的组合引起的。实验结果表明，本文的方法可以有效地测试Web应用程序的搜索功能，该错误定位方法可以准确地找到出错字符组合。

未来工作中，主要考虑两方面的工作：一是完成一个Web应用程序输入参数的自动化的测试工具，

并将其应用到更多的网站测试中;二是把组合测试应用在Web应用程序安全测试上。搜索功能存在错误的网站有很大可能存在可注入代码的漏洞。组合测试的方法可以生成高质量的测试用例来测试这些Web应用程序的安全问题。

References:

[1] Mesbah A. Advances in testing JavaScript-based Web applications[J]. Advances in Computers, 2015, 97: 201-235.

[2] Chen X B, Zhang H Y, Luo L M, et al. Research on technique of SQL injection attacks and detection[J]. Computer Engineering and Applications, 2007, 43(11): 150-152.

[3] Lian K M, Xu J, Tian W, et al. Research on SQL injection vulnerability multi-level detection method[J]. Journal of Frontiers of Computer Science and Technology, 2011, 5(5): 474-480.

[4] Wu H Q. White hat speaking of Web security[M]. Beijing: Publishing House of Electronics Industry, 2012.

[5] Yan J, Zhang J. Combinatorial testing: principles and methods[J]. Journal of Software, 2009, 20(6): 1393-1405.

[6] Kuhn D R, Reilly M J. An investigation of the applicability of design of experiments to software testing[C]//Proceedings of the 27th Annual NASA Goddard/IEEE Software Engineering Workshop, Greenbelt, Dec 5-6, 2002. Washington: IEEE Computer Society, 2002: 91-95.

[7] Raunak M S, Kuhn D R, Kacker R. Combinatorial testing of full text search in Web applications[C]//Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security Companion, Prague, Jul 25-29, 2017. Piscataway: IEEE, 2017: 100-107.

[8] Zhang J, Ma F F, Zhang Z Q. Faulty interaction identification via constraint solving and optimization[C]//LNCS 7317: Proceedings of the 15th International Conference on Theory and Applications of Satisfiability Testing, Trento, Jun 17-20, 2012. Berlin, Heidelberg: Springer, 2012: 186-199.

[9] Colbourn C J, McClary D W. Locating and detecting arrays for interaction faults[J]. Journal of Combinatorial Optimization, 2008, 15(1): 17-48.

[10] Zeller A, Hildebrandt R. Simplifying and isolating failure-inducing input[J]. IEEE Transactions on Software Engineering, 2002, 28(2): 183-200.

[11] Yu L B, Lei Y, Kacker R N, et al. ACTS: a combinatorial test generation tool[C]//Proceedings of the 2013 IEEE 6th International Conference on Software Testing, Verification and Validation, Luxembourg, Mar 18-22, 2013. Washington: IEEE Computer Society, 2013: 370-375.

[12] Simos D E, Kleine K, Ghandehari L S G, et al. A combinatorial approach to analyzing cross-site scripting (XSS) vulnerabilities in Web application security testing[C]//LNCS 9976: Proceedings of the IFIP International Conference on Testing Software and Systems, Graz, Oct 17-19, 2016. Berlin, Heidelberg: Springer, 2016: 70-85.

[13] Bozic J, Garn B, Kapsalis I, et al. Attack pattern-based combinatorial testing with constraints for web security testing[C]//Proceedings of the 2015 IEEE International Conference on Software Quality, Reliability and Security, Vancouver, Aug 3-5, 2015. Piscataway: IEEE, 2015: 207-212.

[14] Appelt D, Nguyen C D, Panichella A, et al. A machine-learning-driven evolutionary approach for testing web application firewalls[J]. IEEE Transactions on Reliability, 2018, 67(3): 733-757.

[15] Yilmaz C, Dumlu E, Cohen M B, et al. Reducing masking effects in combinatorial interaction testing: a feedback driven adaptive approach[J]. IEEE Transactions on Software Engineering, 2014, 40(1): 43-66.

[16] Nie C, Leung H. The minimal failure-causing schema of combinatorial testing[J]. ACM Transactions on Software Engineering and Methodology, 2011, 20(4): 15.

[17] Bozic J, Simos D E, Wotawa F. Attack pattern-based combinatorial testing[C]//Proceedings of the 9th International Workshop on Automation of Software Test, Hyderabad, May 31-Jun 1, 2014. New York: ACM, 2014: 1-7.

[18] Qi X F, Wang Z Y, Mao J Q, et al. Automated testing of Web applications using combinatorial strategies[J]. Journal of Computer Science and Technology, 2017, 32(1): 199-210.

[19] Wang W H, Lei Y, Liu D G, et al. A combinatorial approach to detecting buffer overflow vulnerabilities[C]//Proceedings of the 2011 IEEE/IFIP International Conference on Dependable Systems and Networks, Hong Kong, China, Jun 27-30, 2011. Washington: IEEE Computer Society, 2011: 269-278.

[20] Tripp O, Weisman O, Guy L. Finding your way in the testing jungle: a learning approach to Web security testing [C]//Proceedings of the 2013 International Symposium on Software Testing and Analysis, Lugano, Jul 15-20, 2013. New York: ACM, 2013: 347-357.

[21] Sheykhkanloo N M. A learning-based neural network model for the detection and classification of SQL injection attacks[J]. International Journal of Cyber Warfare and Terrorism, 2017, 7(2): 16-41.

[22] Rathore S, Sharma P K, Park J H. XSSClassifier: an efficient XSS attack detection approach based on machine learning classifier on SNSs[J]. Journal of Information Processing Systems, 2017, 13(4): 1014-1028.

附中文参考文献:

[2] 陈小兵, 张汉煜, 骆力明, 等. SQL注入攻击及其防范检测技术研究[J]. 计算机工程与应用, 2007, 43(11): 150-152.
[3] 练坤梅, 许静, 田伟, 等. SQL注入漏洞多等级检测方法研究[J]. 计算机科学与探索, 2011, 5(5): 474-480.
[4] 吴翰清. 白帽子讲Web安全[M]. 北京: 电子工业出版社, 2012.
[5] 严俊, 张健. 组合测试: 原理与方法[J]. 软件学报, 2009, 20(6): 1393-1405.

LV Chengcheng was born in 1994. He is an M.S. candidate at University of Science and Technology of China. His research interest is software testing.
吕成成 (1994—), 男, 陕西延安人, 中国科学技术大学计算机科学与技术系硕士研究生, 主要研究领域为软件测试。

ZHANG Long was born in 1989. He is a Ph.D. candidate at Institute of Software, Chinese Academy of Sciences, and the member of CCF. His research interests include program analysis, software testing, software debugging, etc.
张龙 (1989—), 男, 安徽怀远人, 中国科学院软件研究所博士研究生, CCF会员, 主要研究领域为程序分析, 软件测试, 软件调试等。

DENG Xi was born in 1993. She is a Ph.D. candidate at Institute of Software, Chinese Academy of Sciences, and the student member of CCF. Her research interests include software testing, static analysis, etc.
邓茜 (1993—), 女, 湖北武汉人, 中国科学院软件研究所博士研究生, CCF学生会员, 主要研究领域为软件测试, 静态分析等。

ZENG Fanping was born in 1967. He is an associate professor at University of Science and Technology of China, and the senior member of CCF. His research interests include software analysis and testing, information security, etc.
曾凡平 (1967—), 男, 江西南康人, 中国科学技术大学计算机学院副教授, CCF高级会员, 主要研究领域为软件分析与测试, 信息安全等。

YAN Jun was born in 1980. He received the Ph.D. degree in computer software and theory from Institute of Software, Chinese Academy of Sciences in 2007. Now he is a researcher at Institute of Software, Chinese Academy of Sciences, and the senior member of CCF. His research interests include program analysis, software testing, etc.
严俊 (1980—), 男, 湖北黄石人, 2007年于中国科学院软件研究所计算机软件与理论专业获得博士学位, 现为中国科学院软件研究所研究员, CCF高级会员, 主要研究领域为程序分析, 软件测试等。

ZHANG Jian was born in 1969. He received the Ph.D. degree from Institute of Software, Chinese Academy of Sciences in 1994. Now he is a professor and Ph.D. supervisor at Institute of Software, Chinese Academy of Sciences, and the fellow of CCF, the senior member of ACM and IEEE. His research interests include automated reasoning, constraint solving, semantic Web, program static analysis and error detection, software testing, data generation, etc.
张健 (1969—), 男, 安徽庐江人, 1994年于中国科学院软件研究所获得博士学位, 现为中国科学院软件研究所研究员、博士生导师, CCF会士, ACM和IEEE高级会员, 主要研究领域为自动推理, 约束求解, 语义Web, 程序静态分析与检错, 软件测试, 数据生成等。