基于策略推导的访问控制漏洞测试用例生成方法

文 硕[1)] 许 静[1)] 苑立英[1)] 李晓虹[1)] 徐思涵[1)] 司冠南[2)]

[1)](南开大学计算机与控制工程学院 天津 300071)

[2)](山东交通学院信息科学与电气工程学院 济南 250357)

摘 要 Web 应用已经成为越来越流行的信息传输媒介.为了保护重要信息不被泄漏,许多 Web 应用设计了针对不同角色不同用户的访问控制机制.然而由于不完善的访问控制机制,使得访问控制漏洞仍普遍存在,攻击者可对 Web 应用的敏感数据进行非法访问.为了获得准确的访问控制机制,测试用例的准确性和有效性至关重要.然而,现有的测试用例生成方法存在漏报、冗余度高等缺陷.文中根据 Web 应用程序的访问控制模型,提出一种基于策略推导的测试用例生成方法.此方法从角色和用户两个级别发现对应的授权操作集合,推导 Web 应用程序的访问控制策略,并利用推导所得访问控制策略生成合法与非法两类测试用例.其中,合法用例用以对推导所得策略的正确性进行验证,非法用例通过违背授权约束生成,用以检测 Web 应用程序的访问控制漏洞.为了对方法的有效性进行验证,我们设计并实现原型系统 ACV-Scanner,并将其运行在开源 Web 应用上.实验结果表明该方法在能全面检测各种类型的访问控制漏洞的前提下,对测试用例进行了精简,与同类研究对比,减少漏报,提高了效率.

关键词 软件测试;Web 应用;访问控制漏洞;访问控制策略;测试用例生成

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2017.02658

A Test Case Generation Approach for Exploiting Access Control Vulnerabilities Based on Policy Inference

WEN Shuo[1)] XU Jing[1)] YUAN Li-Ying[1)] LI Xiao-Hong[1)] XU Si-Han[1)] SI Guan-Nan[2)]

[1)](*College of Computer and Control Engineering, Nankai University, Tianjin* 300071)

[2)](*School of Information Science and Electrical Engineering, Shandong Jiaotong University, Jinan* 250357)

Abstract Web applications have become more and more popular for delivering information over the Internet. Although most of web applications implement access control mechanisms that restrict the data access privileges of different roles and users, access control vulnerabilities still exist due to incomplete design of access control mechanisms, in which case attackers could access sensitive data illegally. To achieve accurate access control mechanisms, it is significant to generate accurate and efficient test cases. However, existing test case generation approaches have high redundancy and false negatives. In this paper, we propose a novel test case generation approach based on policy inference, which is according to access control models of web applications, to discover access control vulnerabilities within web applications. This approach identifies the sets of authorized operations from two levels, i.e., role and user, then infers access control policy, and finally utilizes the inferred policy to generate legal and illegal test cases. The legal test cases aim to verify the legality of the inferred policy, while the illegal test cases generated by violating

收稿日期:2015-07-29;在线出版日期:2016-03-01.本课题得到天津市自然科学基金重点项目(12JCZDJC20800)、天津市科技计划项目(13ZCZDGX01098)、国家科技支撑计划项目(2013BAH01B05)和国家自然科学基金青年基金项目(61402264)资助.**文 硕**,男,1987 年生,博士,主要研究方向为软件工程、软件安全.E-mail:wenshuo@mail.nankai.edu.cn.**许 静**,女,1967 年生,教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为软件工程、软件安全.**苑立英**,女,1990 年生,硕士研究生,主要研究方向为软件分析、软件安全.**李晓虹**,女,1990 年生,硕士研究生,主要研究方向为软件分析、软件安全.**徐思涵**,女,1991 年生,博士研究生,主要研究方向为软件工程、软件测试.**司冠南**,男,1981 年生,讲师,主要研究方向为软件工程、软件评估技术.

authorized constraints are utilized for exploiting access control vulnerabilities within web applications. A prototype system ACV-Scanner is also implemented for evaluation over a set of web applications. The experiment results demonstrate that our method can effectively decrease test cases, reduce false negative and improve the efficiency while comprehensively exploiting different categories of access control vulnerabilities.

Keywords　software testing; web application; access control vulnerability; access control policy; test case generation

1　引　言

Web 应用是以互联网为基础平台为用户提供 Internet 服务的网络应用软件. 近年来，Web 应用已经成为了互联网主要的信息传输媒介. 其通过前台的 Web 服务器与用户进行交互，然后访问应用后台数据库存储的信息. 在此结构中，Web 应用发起具有一定权限的请求，后台的数据库根据 Web 应用发起的请求，执行相应权限的数据库操作. 正因为此，Web 应用必须在数据库接收请求并执行操作前保证相应的安全检查，从而保护其后台数据库上的数据. 而随着 Web 技术的成熟与发展，软件系统也越来越复杂，Web 应用的安全问题愈发严重，所以安全检测是 Web 应用设计过程中必不可少的环节，而 Web 应用的特殊性也给安全检测带来很大的挑战[1].

Web 应用所具有的开放性特点使得网络用户通过 Web 客户端提交请求后，数据库服务器无法对服务请求者的身份进行安全和合法性校验. 为保证 Web 应用的访问安全性，代码编写者通常在 Web 服务器的应用逻辑中加入访问控制机制，对请求者的身份进行合法校验和过滤，以达到对数据库服务器的合法保护. 尽管许多 Web 应用设计了一定的访问控制机制，但因为安全细节极其琐碎复杂、代码编写者的疏漏等原因，这些访问控制机制很难做到万无一失，使得在 Web 应用开发过程中，实际在客户端与服务器之间实现的访问控制机制往往与预期定义的访问控制策略存在差异，这种差异就造成了 Web 应用中的访问控制漏洞. 此类漏洞为非法用户的恶意操作提供了机会，使 Web 应用的信息资源被非法使用和访问. 访问控制漏洞是 Web 应用漏洞的一大类别，其允许攻击者去绕过应用的安全策略来访问未授权的敏感信息或进行非法操作，在开放式 Web 应用安全项目(OWASP)公布的十大 Web 应用安全漏洞中，三种类别的安全漏洞均可能由访问控制漏洞造成，即不安全的直接对象引用，功能级访问控制缺失和未验证的重定向和转发①. 对 Web 应用的访问控制漏洞检测也成为 Web 应用安全检测的一大重点.

针对 Web 应用的访问控制漏洞检测，主要有黑盒测试以及白盒测试两种测试手段，白盒测试通过直接对程序代码进行分析来发现其中的缺陷，但其具有需要程序源代码可见，需要知道代码的逻辑结构的缺点；相比而言，黑盒测试具有不需要程序的源代码、整个测试过程基于对 Web 应用的运行以及注重程序的功能等优点，所以黑盒测试得到越来越多的关注.

在黑盒测试中，生成测试用例的优劣对漏洞检测的准确性和效率至关重要. 目前针对访问控制漏洞的测试用例生成方法均存在不同程度的漏报缺陷. 本文提出一种基于策略推导的针对访问控制漏洞的测试用例生成方法. 根据 Web 应用的服务功能，该方法从角色和用户两个级别分别对访问控制策略进行推导，并得到相应的操作集合. 然后基于推导出来的策略，从相应级别分别生成合法和非法测试用例. 通过该方法生成的测试用例，可全面检测各种类型的访问控制漏洞，并有效地减少测试用例的数量，提高测试效率. 与此同时，本文设计并实现了一个可自动生成并执行测试用例的原型系统 ACV-Scanner. 实验证明了该系统推导的 Web 应用的访问控制策略的正确性，以及生成的测试用例的有效性及高效性.

本文第 2 节对相关研究工作进行总结概括，第 3、4 节分两步对测试用例生成算法做介绍：第 3 节在现有 Web 应用功能集合上推导目标 Web 应用的访问控制策略；然后基于推导出的策略，第 4 节分别从角色和用户两个层次生成合法和非法测试用例；第 5 节基于第 3、4 节的方法给出系统工具的实现框

① OWASP Top10. https://www.owasp.org/index.php/Top_10_2013-Top_10

架;第 6 节在公开的 Web 应用上进行实验,并对实验结果进行评估;第 7 节对全文进行总结.

2 相关研究

访问控制的目的是通过限制用户对数据信息的访问能力及范围,保证信息资源不被非法使用和访问[2].它通过制定一套规则对服务请求者是否享有对服务的访问权限进行授权,决定不同权限级别的不同用户的合法操作集合,防止非法用户的入侵,从而保证数据被合法访问管理.

在访问控制技术的研究中,需要对用户访问和授权管理等方面进行描述,有许多语言均可实现这类需求,其中,可扩展访问控制标记语言(Extensible Access Control Markup Language, XACML)[3]已经成为描述访问控制策略的标准建模语言,其描述的策略由一组规则组成该规则保证网络资源不被非法使用,当用户申请对资源进行访问时,访问控制模块通过评估访问控制策略对用户进行授权.随着云计算及分布式环境等最新技术的流行,描述的规则数量大幅上升、复杂度也越来越高,这对策略评估效率有了新的性能要求.为此,研究者们在策略评估引擎研究方向上取得许多进展,从而提高策略评估的效率[4-7].

访问控制漏洞是 Web 应用最主要的安全漏洞之一,对 Web 应用中访问控制漏洞的检测也受到研究者们越来越多的关注.早期针对 Web 应用中访问控制漏洞的检测着重于网页级别的漏洞检测,此类漏洞有两种表现形式:第一种形式是强制访问漏洞[8],攻击者可通过直接访问只授权于与攻击者权限不同的角色访问的网络页面来获取或篡改未被授权的数据;第二种形式是参数篡改漏洞[9],攻击者可通过将网络请求的敏感参数篡改成其他用户的参数来访问只授权于其他用户访问的网页来获取或篡改未被授权的数据.这两种漏洞的表现形式均为可见的网页级别.然而,后期的研究者发现一种重定向后运行漏洞[10],该漏洞拥有更隐蔽的非网页级别的表现形式,攻击者针对这种访问控制漏洞进行攻击后,从网页级别来看,似乎攻击者的攻击被成功阻挡(重定向至安全网页),但 Web 应用后台的数据库仍然执行了攻击者未被授权的数据库操作.形成此类漏洞的原因是,代码编写者通常会在需要的时候使用重定向语句,然而在特定的情况下,因为代码编写错误,该重定向语句并没有终止本不应继续运行的操作,进而导致未授权的操作得以继续运行.针对强制访问漏洞、参数篡改漏洞以及重定向后运行漏洞的检测是访问控制漏洞检测的研究重点,目前的研究方法主要分为黑盒及白盒两大类别.

白盒测试工具通过对 Web 应用的源代码进行分析从而找出 Web 应用中的访问控制漏洞,此类方法均要求获取 Web 应用的源代码以知道代码的逻辑结构.但现有的研究成果均存在针对某种特定语言或具有一定的漏报误报率等问题.文献[8]通过基于角色的分析自动寻找出 PHP Web 应用中不安全的程序执行路径,但其方法有一定的误报率,不能检测重定向后运行漏洞,而且只能针对 PHP 语言.文献[11]通过从 Web 应用源代码提取 Web 应用的控制流图来生成控制流路径,但其方法仅针对重定向后运行漏洞,并且也只针对 Ruby on Rail 语言.Nemesis[12]使用动态信息流跟踪方法来检测应用中的访问控制漏洞,但其需要 Web 应用设计者为资源指定访问控制列表.RoleCast[13]分析 Web 应用中每个角色可进行的 SQL 插入、删除、更新语句并分析这些操作的安全检查是否完备,其后续研究 FixMeUp[14]使用了基于用户的策略来实现对用户认证的修复,MACE[15]新定义了上下文授权一致性的概念,并通过检测 Web 应用的上下文授权一致性来检测 Web 应用是否存在授权相关漏洞,但它们均只针对 PHP 语言进行研究.

针对访问控制漏洞的黑盒检测工具通过观察网络应用正常运行的行为来推断程序设计时的访问控制规格.但现有的研究成果有不同程度的漏报或需要额外(从 Web 应用代码编写者或服务器端)获取信息等问题.NoTamper[9]最早提出检测 Web 应用中的访问控制漏洞的黑盒方法.该方法通过分析客户端代码抽取预期行为规范,生成一组合法认证的测试用例和一组经过参数篡改的非法测试用例,但其只能针对客户端网页表单参数与服务器的一致性进行授权检查,并且仅能检测参数篡改漏洞,无法检查隐藏于页面后的重定向和转发请求等访问控制漏洞.FlowWatcher[16]实现了网页级别的漏洞识别,但其需要 Web 应用代码编写者运用文章内新定义的语言提供 Web 应用预期访问策略.有限状态机[1]方法也被多种研究成果采用.BLOCK[17]、SENTINEL[18]通过获取服务器端的会话消息分别实现了对网页级别和数据库级别有限状态机的构造.LogicScope[19]根据合法的网络请求响应构造出有限状态机,然后针对强制访问和参数篡改两种漏洞类型,在每个状

态结点生成非法测试用例，以实现对网页级别的访问控制漏洞检测. 这三种方法均使用有限状态机作为理论基础，但构造有限状态机的过程均需要获取服务器端的会话信息. 此外，BLOCK、SENTINEL分别只能针对网页级别和数据库级别的访问控制漏洞，LogicScope 无法检测重定向后的运行漏洞.

与先前研究相比，我们的研究基于黑盒分析，不需要 Web 应用的源代码、不针对特定的程序语言和不需要获取代码编写者提供的预期访问策略或服务器端的会话信息，并能同时检测强制访问漏洞、参数篡改漏洞以及重定向后运行漏洞等多种类型的访问控制漏洞.

3　访问策略推导

访问控制技术的本质是授权策略[20-21]，它通过制定一套规则来授予不同服务请求者对应的服务访问权限，并确定享有不同权限的不同用户所拥有的操作集合，进而防止非法用户的攻击，以保证数据的安全. 首先，定义以下基本元素.

定义 1.　角色(Role). 每个角色 $R\in SetRole$ 表示 Web 应用中享有相同权限的用户的集合.

$SetRole=\{R_1,\cdots,R_n\}$，$SetRole$ 表示 Web 应用中所有角色的集合，$n\geq 1$ 表示应用中角色的总个数，角色集合中任意两个角色 R_i 和 R_j 之间无偏序关系.

定义 2.　用户(User). 与 Web 服务进行交互的对象. 一个用户 $U\in SetUser$ 代表一个被分配指定角色 R 的用户.

集合 $SetUser=\{U_{11},\cdots,U_{1m_1},U_{21},\cdots,U_{2m_2},\cdots,U_{n1},\cdots,U_{nm_n}\}$，其中 U_{ij} 表示被分配角色 R_i 的用户，$j\in[1,m_i]$，$i\in[1,n]$，$n\geq 1$ 表示应用中角色的总个数，$m_i\geq 1$ 表示应用中享有角色 R_i 的用户总个数.

定义 3.　策略(Policy). 授权规则下用户被分配的合法权限操作集合 $SetPolicy=\{Pol_1,\cdots,Pol_n\}$，$Pol_i$ 表示拥有角色 R_i 的用户被分配的合法权限操作集合，$i\in[1,n]$，$n\geq 1$ 表示应用中角色的总个数.

定义 4.　操作(Operation). 一个操作 O 指的是用户对 Web 服务的请求操作. O 的组成结构为 $[Dest:P]$. $Dest$ 表示服务请求目的地；P 表示用户对该操作被授权的参数(Parameter)，决定了此用户可访问的数据.

集合 $P(U(O))$ 表示一个用户 U 对操作 O 被授权的所有参数的集合.

$SpecP(U(O))$ 表示用户 U 对操作 O 被授权的特有的参数(Specific Parameter)的集合.

集合 $Pol_i=\{O_{i1},\cdots,O_{is}\}$，$Pol_i$ 中的每个元素均为原子操作，$s\geq 1$ 表示集合 P_i 中原子操作的总个数.

定义 5.　会话(Session). 一个会话 S 表示某用户从登陆 Web 服务到登出期间请求的所有操作.

定义 6.　资源(Resource). 用户请求的 Web 应用后台的数据，受访问控制策略保护.

3.1　交互样本采集

本文的测试用例生成方法为黑盒分析的方法，需要对用户与 Web 应用服务器、Web 应用服务器与数据库服务器之间交互数据的样本(表示为 Sample)进行分析. 交互样本的采集首先需要手动分析目标 Web 服务的功能(角色、用户及对应可进行的操作)集合，然后驱动各角色下的不同用户来触发该角色被授权操作的集合，进而得到每个角色被授权的合法操作的全集.

假设：

$\{O\text{ 合法}\mid O\in Pol_i, Pol_i=\{O_{i1},O_{i2},\cdots,O_{is}\},\ i\in[1,n], Pol_i\in SetPolicy\}$，

则交互样本形式化表示如下：

$$Sample=\{\langle U_{ij},R_i,Pol_i\rangle\mid U_{ij}\in SetUser, R_i\in SetRole, Pol_i\in SetPolicy, j\in[1,m], i\in[1,n]\}.$$

下面给出交互样本的采集算法.

算法 1.　交互样本的生成算法.

```
输入：SetRole, SetUser
输出：Sample
FOREACH SetRole 中的角色 Ri
  FOREACH 角色 Ri 下的用户 Uij
    FOREACH 操作集合 Poli 中的操作 O
      Run(⟨Uij,Ri,O⟩); //驱动用户 Uij 执行操作 O
      Collect();//采集交互数据
    ENDFOR
  ENDFOR
ENDFOR
```

算法 1 对每个角色 R_i 进行遍历，通过 Run 函数驱动享有该角色的所有用户来触发该角色被授权操作的集合. 同时，$Collect$ 函数对交互数据进行采集，从而得到目标 Web 应用的交互样本.

3.2　访问控制策略推导

图 1 表示一个 Web 应用的访问控制策略模型，其中角色与用户之间存在着一对多的映射关系. 对于 Web 应用的一个用户而言，其享有的角色与被授权的操作参数共同决定了其能访问的资源.

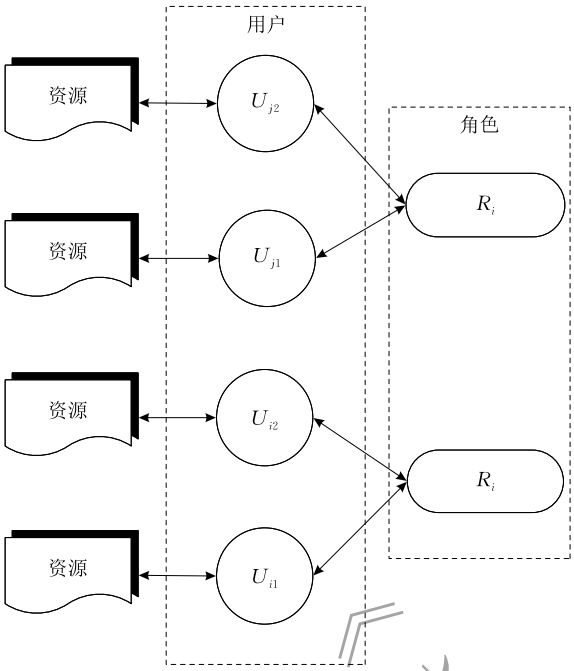

图 1 访问控制策略模型

基于访问控制策略模型的两个层面,推导的访问控制策略可分为基于角色的策略和基于用户的策略两个大类.

定义 7. 基于角色的策略:表示同角色下所有用户被授权的操作集合.一般而言,不同角色下的用户被授权的权限操作集合不相等,但可有交集.如图 2 所示,Pol_1 和 Pol_2 的交集表示授权给角色为 R_1 和 R_2 的用户的共有操作集合.

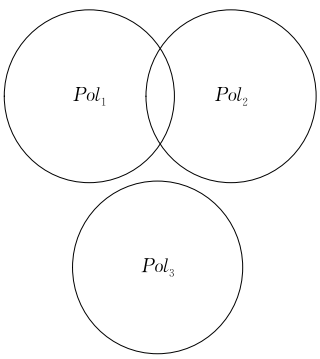

图 2 基于角色的策略中的操作间关系

定义 8. 基于用户的策略:表示同一角色下不同用户被授权的特有操作参数集.一般而言,同角色下的用户的操作集合基本相同,但同角色下的不同用户被授权的操作参数却并不相同.如图 3 所示,对于同一操作 O,享有共同角色的用户 U_{i1} 和 U_{i2} 被授权的操作参数不同,即有 $P(U_{i1}(O))\neq P(U_{i2}(O))$.

基于角色的策略对同一角色下的所有用户的操作集合进行确定,基于用户的策略通过参数对操作

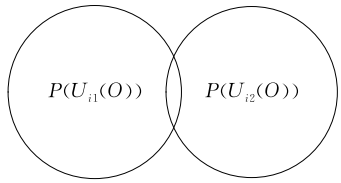

图 3 基于用户的策略中的授权操作参数间的关系

进行约束,对同一角色下的不同用户可访问的资源进行区分.

根据用户与操作参数的相关程度,基于用户的策略分为两个大类.

定义 9. 直接授权(Direct Authorization):简称 DAuth,表示用户与操作参数直接相关.例如,一个用户向 Web 应用提交如下请求:

请求 1.

SELECT *bookID* FROM *books*

WHERE *editor*='*TOM*'

其中的 *editor* 参数与用户直接相关,为直接授权.

定义 10. 间接授权(Indirect Authorization):简称 IAuth,表示操作参数与用户不直接相关,而是从属于一个直接授权操作返回的响应数据集.例如,定义 9 的例子中请求 1 为直接授权操作,其返回的响应结果为 *bookID* 的相关参数集合 *SetBookID*,请求 1 的后续请求为:

请求 2.

SELECT * FROM *books*

WHERE *bookID*='*mybook*'

且请求 2 中'*mybook*'∈*SetBookID*,则请求 2 中的 *bookID* 参数为间接授权.显然,间接授权操作之前的操作为直接授权操作,且这两个连续的操作之间存在至少一个参数的传输.

3.2.1 基于角色的策略推导

基于角色的策略可通过享有各角色下的不同用户可访问的请求操作进行推导.由于 Web 应用对用户的写入操作不做修改(使得写入操作可直接从 Sample 中提取),却可能会对读操作中数据库返回的值进行二次修改,因此主要对读操作的推导进行介绍.首先,通过分析请求返回的数据库与网络响应,提取传输到网络响应(网络页面)的有效字段;然后,基于提取的有效字段对过滤条件进行推导.

(1)有效字段提取

判断数据库响应中的哪些键被传输到网络响应,需要解决两个难点:第一,与数据库响应通过表结构进行表示不同,网络页面通过 DOM 树结构进行表示;第二,数据库响应中的键值通过键的名称进

行表示，而键名相关的信息通常不会被传输到网络页面中. 而且，数据库响应中的数据往往会存在于网络页面的不同位置，不能在特定的位置进行捕获. 所以，需要三个步骤来提取数据库响应到网络响应(网络页面)的有效字段：

首先，将数据库响应和网络响应中的数据转换成统一的键值数据结构. 数据库响应通过表结构(Table)进行表示，可直接转换为键值数据结构. 网络响应通过 DOM 结构进行表示，使用从 DOM 树结构的根结点到网络页面的数据的唯一路径 xpath 作为键，则可将网络响应转换为键值数据结构，以上转换表示为

Database Response：Table→[dbfield:value]

Web Response：DOM→[xpath:value]

其次，将数据库响应与网络响应键值对在 value 值上进行模糊匹配，得到匹配键值对[xpath：dbfield].

最后，对匹配键值对中的 xpath 进行合并. 若存在 $xpath_1$ 和 $xpath_2$ 均匹配相同的 field，且这两个 xpath 拥有共同的前缀，则将它们合并至相同的前缀下，最终得到传输至网络响应的有效字段，即得到有关 dbfield 的集合 $Set(dbfield)$.

(2) 过滤条件推导

在提取有效字段之后，从数据库响应中对应的字段识别传输到网络响应的 value 值，并获取所在行的位置. 然后，从数据库响应中提取所在行中其他字段的 value 值. 若某字段 $dbfield_i$ 提取的所有值均为共同值 v_i，则$[dbfield_i:v_i]$为过滤条件.

通过对 Web 应用与后台数据库间的交互样本进行分析，对请求操作进行提取并对过滤条件进行推导，可得到每个角色被授权的操作集合.

3.2.2　基于用户的策略推导

基于用户的策略通过参数对操作进行约束，对同一角色下的不同用户可访问的资源进行区分. 下面给出直接授权(DAuth)和间接授权(IAuth)的推导方法.

(1) 直接授权(DAuth)推导

直接授权对应的参数可直接从 Web 应用的用户提交的网络请求中进行提取.

首先，从交互样本中提取同角色下的每个用户的操作及其参数，对相同操作的参数进行一致性匹配，得到操作 O 下每个用户对被授权的参数集合 $P(U_{ij}(O))$，其中 $U_{ij}\in R_i(U_{i1},\cdots,U_{im})$. 然后，对参数集合 $P(U_{ij}(O))$ 进行去重，去除不同用户拥有的重复参数，从而得到用户特有的授权参数集合 $SpecP(U_{ij}(O))$. 即得到用户 U_{ij} 的 DAuth 集合，即 $\{SpecP(U_{ij}(O_{i1})),SpecP(U_{ij}(O_{i2})),\cdots,SpecP(U_{ij}(O_{is}))\}$.

下面给出具体的实现算法.

算法 2.　一致性匹配算法.

输入：$SetRole$，$SetPolicy$，$SetUser$

输出：$SetP=\bigcup_{i=1}^{n}\bigcup_{j=1}^{m}\bigcup_{k=1}^{s}\{U_{ij}:[O_{ik}:P(U_{ij}(O_{ik}))]\}$

```
FOREACH SetRole 中的角色 R_i
  FOREACH 角色 R_i 下的用户 U_ij
    从 SetPolicy 中选取 R_i 的授权操作集合 Pol_i;
    FOREACH Pol_i 中的操作 O
      IF (!SetP.get(U_ij).contains(Dest(O)))
        SetP.get(U_ij).add(Dest(O));
      ENDIF
        SetP.get(U_ij).get(Dest(O)).addP(P(O));
    ENDFOR
  ENDFOR
ENDFOR
```

此算法对参数进行一致性匹配，输出每个用户对应的操作参数集，$SetP$ 表示所有用户 U_{ij} 对每个操作 O_{ik} 享有的参数集 $P(U_{ij}(O_{ik}))$，其中 $i\in[1,n]$，$j\in[1,m]$，$k\in[1,s]$.

算法 3.　去重算法.

输入：$SetRole$，$SetPolicy$，$SetP$

输出：$SetSpecP=\bigcup_{i=1}^{n}\bigcup_{j=1}^{m}\bigcup_{k=1}^{s}\{U_{ij}:[O_{ik}:Spec(U_{ij}(O_{ik}))]\}$

```
FOREACH SetRole 中的角色 R_i
  从 SetPolicy 中选取 R_i 的授权操作集合 Pol_i;
  FOREACH Pol_i 中的操作 O
    FOREACH 角色 R_i 下的用户 U_ij
      P(U_ij(O))=SetP.get(U_ij).get(Dest(O)).GetP();
      //获取 U_ij 对操作 O 的授权参数集合
      SetSpecP.get(U_ij).add(Dest(O));
      SetDiff=P(U_ij(O));
      FOREACH R_i(U_i1,U_i2,…,U_im)−U_ij 中的用户 U_ik
        P(U_ik(O))=SetP.get(U_ik).get(O);
        SetDiff=SetDiff−P(U_ik(O));
        //取 U_ij 与 U_ik 的参数集的差集
      ENDFOR
        SetSpecP.get(U_ij).get(Dest(O)).addP(SetDiff);
    ENDFOR
  ENDFOR
ENDFOR
```

此算法对参数进行去重，输出每个用户特有的操作参数集.

图 4 为授权参数去重的一个简单例子，可见用户 U_1 和 U_2 被授权的操作参数集合中有重复的参

数.经过去重(算法 3)后,才分别得到仅授予用户 U_1 和 U_2 的特有参数集.

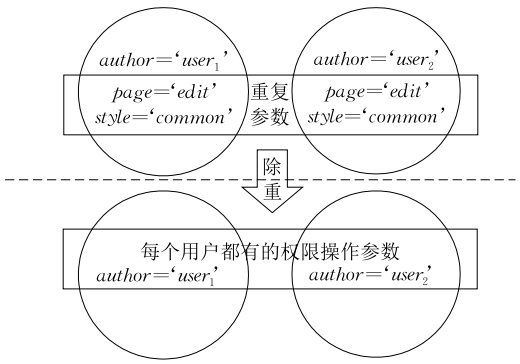

图 4 直接授权(DAuth)推导中的参数去重举例

(2)间接授权(IAuth)推导

IAuth 的操作参数从属于上一个操作所返回的响应数据集,IAuth 操作的上一个操作为可返回响应数据集的 DAuth 读操作,操作之间的参数传输可表示为

Previous DAuth Response→Web Response→
Current Web request→IAuth

所以对 IAuth 的操作参数集的推导,需分析当前 IAuth 操作的上一个操作.即 IAuth 的推导需要对参数传输中的两个相关操作进行分析.

假设:O_{cur} 为当前操作,O_{pre} 为上一个操作,则若对于用户 U 满足如下条件,则为 IAuth 操作:

(1) O_{pre} 为 DAuth 读操作;

(2) O_{pre} 返回的数据库响应数据集为 $dbResponse(O_{pre})$,$dbResponse(O_{pre})$ 传输到网络响应的字段集为 $webResponse(O_{pre})$,O_{cur} 的网络请求参数为 $webRequest(O_{cur})$,对应数据库请求参数为 $dbRequest(O_{cur})$;若 $webRequest(O_{cur})\in webResponse(O_{pre})$,则表明上一个操作的响应字段传输到当前操作的请求参数.

当上述条件成立时,存在参数传输:

$dbResponse\rightarrow webResponse\rightarrow webRequest\rightarrow dbRequest$

此时 O_{cur} 为 IAuth,且 $P(U(O_{cur}))$ 从属于 $webResponse(O_{pre})$.

4 测试用例生成

根据推导的两种访问控制策略,本系统生成测试用例来检测 Web 应用是否真正实现了访问控制策略.生成的测试用例分成合法与非法用例两个大类.合法用例用以提交给目标 Web 应用以判断所推导的策略是否符合 Web 应用的预期访问控制.基于推导的合法策略可通过违背约束构造非法测试用例,生成的非法用例将被自动发送至 Web 应用来对访问控制漏洞进行检测.合法用例可根据推导的访问控制策略对应生成;非法用例可分别通过违背角色和用户层次的策略约束构造.以下进行具体说明.

由于每个操作均由特定角色下的具体用户运行,因此,测试用例可用如下四元组表示.

定义 11. 测试用例(TestCase,简称 TC):

$TC=\langle R_i,U_{ij},Dest(O),P(U_{ik}(O))\rangle$,其中 $i\in[1,n]$,$j\in[1,m]$,$k\in[1,m]$.

即角色 R_i 下的用户 U_{ij} 向目标 Web 应用提交目标为 $Dest(O)$ 的请求 O,O 的操作参数为 $P(U_{ik}(O))$.若 $j=k$,则表示用户 U_{ij} 提交的请求的授权参数合法(为 $P(U_{ij}(O))$;若 $j\neq k$,则表示用户 U_{ij} 提交的请求的授权参数非法(通过使用用户 U_{ik} 的授权参数 $P(U_{ik}(O))$ 进行操作,即参数篡改).

4.1 合法用例生成

为了对推导的访问控制策略的合法性进行验证,我们需要基于推导的策略构造合法测试用例,从而合法访问目标 Web 应用以验证结果.对角色 R_i 的授权操作集 Pol_i 中的每个操作 O,从 $R_i(U_{i1},\cdots,U_{im})$ 中随机选取一个用户 U_{ir} 向目标 Web 应用发送请求.

下面给出具体的实现算法.

算法 4. 合法用例的生成算法.

```
输入:SetRole,SetUser,SetPolicy
输出:legalTC
FOREACH SetRole 中角色 Ri
  从 SetPolicy 中选取 Ri 的授权操作集合 Poli;
  FOREACH Poli 中的操作 O
    从角色 Ri 下随机选取一个用户 Uir
    TC=new generateTC(Ri,Uir,Dest(O),P(Uir(O));
    //构造对应的测试用例
    legalTC.add(TC);
    //将新构造的测试用例加入合法用例集
  ENDFOR
ENDFOR
```

对于一个集合 Set,在此用 $|Set|$ 表示该集合中元素的个数.算法 4 对 Web 应用中每个角色 R_i 的授权操作集 Pol_i 中的每个操作 O,从 $R_i(U_{i1},\cdots,U_{im})$ 中随机选取一个用户 U_{ir} 向目标 Web 应用发送请求,所以该算法对 Pol_i 中的每个 O 进行了遍历.因此,算法 4 的时间复杂度与目标 Web 应用中的操作集合相关,即时间复杂度为 $O(|Pol|*|SetPolicy|)$.

4.2　非法用例生成

为了检测目标 Web 应用中可能存在的访问控制漏洞，我们利用推导所得访问控制策略，分别违背角色和用户层次的策略约束，从而构造非法测试用例.

如图 5 所示，Pol_1、Pol_2分别为角色 R_1、R_2的操作集合，Pol_1、Pol_2分别由若干原子操作$\{O_{11},\cdots,O_{1s}\}$、$\{O_{21},\cdots,O_{2s}\}$组成，R_1、R_2各自的用户集分别为 $R_1(U_{11},\cdots,U_{1m})$和 $R_2(U_{21},\cdots,U_{2m})$.

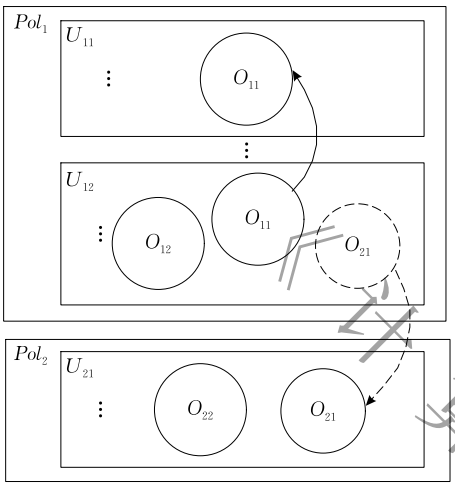

图 5　非法用例生成

假设 $O_{21}\in Pol_2$，且 $O_{21}\notin Pol_1$，则图 5 中虚线部分表示角色 R_1下的用户 U_{12}对角色进行篡改，违背角色之间的限制去运行其本不被允许运行的角色 R_2下的权限操作，此操作即为构造的基于角色的非法用例.

假设$O_{11}\in Pol_1$，参数 $P(U_{11}(O_{11}))\neq P(U_{12}(O_{11}))$，则图 5 中实线箭头及对应的操作表示用户 U_{12}将自己被授权的参数篡改成用户 U_{11}的 O_{11}参数，违背了用户之间的权限约束以访问其本不可访问的其他用户的内容，此过程生成的操作即为构造的基于用户的非法用例.

4.2.1　基于角色的非法用例生成

此类测试用例对应于基于角色的访问控制策略，对 Web 应用中的某角色 R_i进行测试需要对所有其他角色 $R_j(i\neq j)$的权限操作集进行遍历，即

(1) $\exists R_j$，使 $R_j\in(SetRole-R_i)$；

(2) $\exists Pol_i$，Pol_j，且 $Pol_i=\{O_{i1},O_{i2},\cdots,O_{is}\}$，$Pol_j=\{O_{j1},O_{j2},\cdots,O_{js}\}$；

(3) 分别选取 R_i、R_j下的随机用户 U_{ir}、U_{jr}；

(4) $\{\langle R_i,U_{ir},Dest(O_{jk}),P(U_{jr}(O_{jk}))\rangle\mid O_{jk}\in Pol_j$，且 $O_{jk}\notin Pol_i,k\in[1,s]\}$即为角色 R_i对应的非法用例集合.

下面给出具体的实现算法.

算法 5.　基于角色的非法用例生成算法.

输入：$SetRole$，$SetUser$，$SetPolicy$

输出：$illegalRoleTC$

```
FOREACH SetRole 中的角色 R_i
  从 SetPolicy 中选取 R_i 的授权操作集合 Pol_i；
  FOREACH SetRole−R_i 中的角色 R_j
    从 SetPolicy 中选取 R_j 的授权操作集合 Pol_j；
    SetDiff=Pol_j−Pol_i；//取 R_i 和 R_j 的操作差集
    FOREACH SetDiff 中的操作 O
      分别选取 R_i、R_j 下的随机用户 U_ir、U_jr
      TC=new generateTC(R_i,U_ir,Dest(O),P(U_jr(O)));
      //使 U_ir 违背角色之间的限制运行 R_j 下操作，构造对应的测试用例
      illegalRoleTC.add(TC)；
      //将新构造的测试用例加入基于角色的非法用例集
    ENDFOR
  ENDFOR
ENDFOR
```

算法 5 对角色 R_i进行测试，对所有其他角色 $R_j(i\neq j)$下的操作集进行遍历，即其时间复杂度与 $(|SetPolicy|*(|SetPolicy-1|)*|Pol_i-Pol_j|)$同阶，因为 $|Pol_i-Pol_j|=O(1)$，所以算法 5 的时间复杂度为 $O(|SetPolicy|^2)$.

4.2.2　基于用户的非法用例生成

此类测试用例对应于基于用户的访问控制策略，同角色下的用户的操作集合基本相同，但同角色下的不同用户被授权的操作参数却并不相同，使得可访问的资源也有所不同. 通过篡改当前用户的操作参数变为同角色下其他用户的操作参数，即可对用户级别的约束进行违背. 即角色 R_i下的基于用户的非法用例构造如下所示：

(1) $\exists Pol_i=\{O_{i1},\cdots,O_{is}\}$；

(2) $\{\langle R_i,U_{ij},Dest(O_{ik}),P(U_{ir}(O_{ik}))\rangle\mid O_{ik}\in Pol_i,U_{ij}\in R_i(U_{i1},\cdots,U_{im}),U_{ir}\in R_i(U_{i1},\cdots,U_{im})-U_{ij}$，且有 $P(U_{ij}(O_{ik}))\neq P(U_{ir}(O_{ik}))$，$j\in[1,m]$，$k\in[1,s]\}$，即为 R_i的基于用户的非法用例集.

下面给出具体的实现算法.

算法 6.　基于用户的非法用例的生成算法.

输入：$SetRole$，$SetUser$，$SetPolicy$

输出：$illegalUserTC$

```
FOREACH SetRole 中的角色 R_i
  从 SetPolicy 中选取 R_j 的授权操作集合 Pol_j；
  FOREACH Pol_i 中的操作 O
    FOREACH R_i(U_i1,…,U_im) 中的用户 U_ij
      FOREACH R_i(U_i1,U_i2,…,U_im)−U_ij 中的用户 U_ir
```

```
IF P(U_ij(O))≠P(U_ir(O))
//若 U_ij 和 U_ir 对 O 的参数不同
THEN
 TC=new generateTC(R_i,U_ij,Dest(O),P(U_ir(O)));
 //U_ij 篡改参数为 U_ir 的参数来运行操作 O
 illegalUserTC.add(TC);
 //将新构造测试用例加入基于用户的非法用例集
BREAK;
ENDIF
ENDFOR
ENDFOR
ENDFOR
ENDFOR
```

算法 6 针对同角色下的用户,通过篡改某用户的操作参数为同角色下其他用户的参数,即可对用户级别的约束进行违背.对 Web 应用的每个角色,设 $M=|R_i(U_{i1},\cdots,U_{im})|$ 为该角色下用户的数量,$|Pol|$ 为该角色下操作的数量,则每个角色下的时间复杂度为 $O(|Pol|*M^2)$.应用中角色的数量为 $|SetPolicy|$,则算法 6 的时间复杂度为 $O(|SetPolicy|*|Pol|*M^2)$.

算法 4 的输出为合法用例集,算法 5、算法 6 的输出分别为基于角色和基于用户的非法用例集.

5 系统实现

本文研究基于策略的访问控制漏洞测试用例生成技术,设计并实现基于策略的测试用例生成并自动执行的系统工具 ACV-Scanner.其系统框架如图 6 所示,其主要包含 3 个模块:捕捉交互样本捕捉模块、访问控制策略推导模块和测试用例生成模块.

(1)交互样本捕捉模块.该模块完成交互数据样本的收集.通过编写的用户模拟器模拟真实用户向 Web 应用提交合法请求的过程,并使用算法 1 生成完整的交互数据样本.同时,利用网络封包抓取软件捕获交互数据作为 Sample.然后,经过对 Sample 的解析,为每个用户的一次完整的合法访问生成相应的会话(定义 6).该模块输出为多个用户的会话集合,作为下个模块的输入.

(2)访问控制策略推导模块.该模块推导访问控制策略.首先,遍历会话集合,从数据库的请求响应对中提取有效字段.然后,对有效字段在数据库响应中进行一致性匹配,推导过滤条件,从而得到基于角色的策略.最后,对相同角色的不同用户的授权操作集合进行参数筛选,得到用户特有的权限操作参数集合,从而得到基于用户的策略.

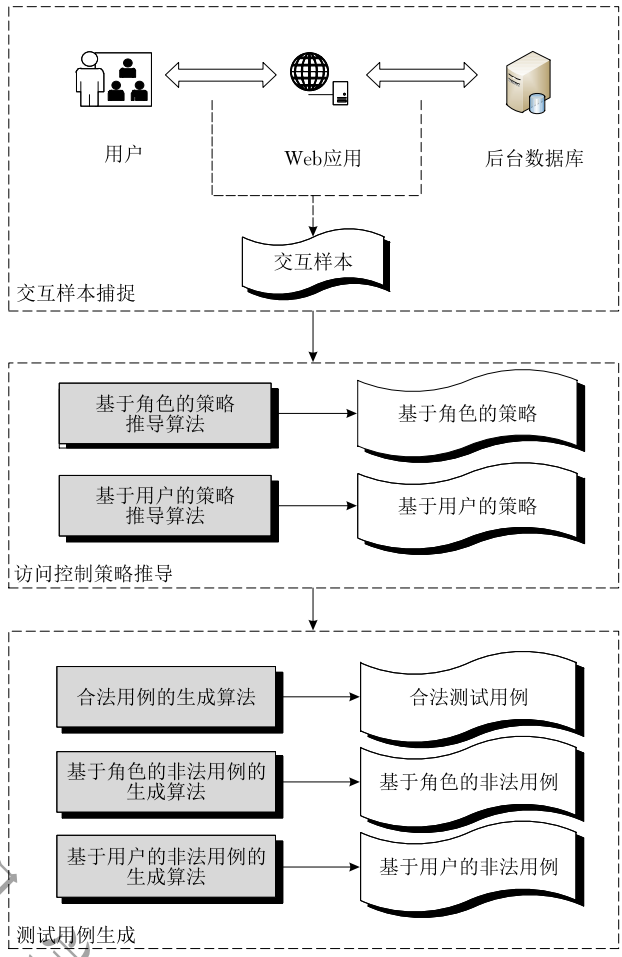

图 6 系统框架图

(3)测试用例生成模块.该模块生成测试用例.根据推导所得策略,首先生成符合约束的合法用例,进行合法性检验.接着,从角色和用户两个层次分别生成违背约束的非法用例.最后,自动执行测试用例,合法用例用来验证推导所得策略是否符合预期约束,非法用例用来检测漏洞,并得出最终的访问控制漏洞检测报告.

系统的前后模块依次承接,每个模块的输出作为下一模块的输入,交互样本捕捉为前期准备,数据解析以及策略推导则是系统的核心部分,依据策略生成合法和非法用例,是对上一模块工作的验证以及对下一模块的准备.执行测试用例,根据响应结果分析得出漏洞报告,并对生成的测试用例的正确性进行验证,则是系统的最终目的.

6 结果评估

为了验证 ACV-Scanner 的有效性,本文在对以传统 SQL 数据库为后台数据库的 Web 应用进行检

测的同时，特别针对基于 MongoDB① 的开源 Web 应用进行检测. MongoDB 具有高达十亿级别的数据量处理能力，对数据量大、并发高、事务弱的互联网应用具有很大适应能力. 它能有效满足 Web2.0 以及移动互联网的数据存储需求，因此在大数据时代中发展迅猛. 因此，本文也选取基于 MongoDB 的开源 Web 应用进行分析. 本文的测试环境为：Intel(R) Core(TM) CPU，4.0GB 内存，Ubuntu 12.04 操作系统.

表 1 为参与实验的 Web 应用的相关信息以及策略推导结果. 第 1 列为参与实验的 Web 应用，第 2 列为应用后台数据库的类型，第 3 列和第 4 列分别为对应 Web 应用中所含文件的数量和代码行数. 其中 MongoBlog 和 MongoDiary 是本文为验证工具有效性而自主开发的 Web 应用，其余均为开源 Web 应用. 第 5 列表示相应 Web 应用中拥有的角色的数目，第 6 列表示 Sample 文件中的会话数目，第 7 列表示分析所得基于角色的策略数目，第 8、9 列表示基于用户的策略数目，第八列代表直接授权策略(DAuth)，第 9 列代表间接授权策略 (IAuth). 通过分析基于角色的策略并结合代码行数和代码所含文件数目可得，系统越复杂，角色的权限操作集合就越复杂. 而从基于用户的策略的角度来看，用户的权限操作对象视其操作参数而定，由于在 ShopCart、Bloggit、EventsLister 和 ShopOnline 中普通用户只被允许浏览 Web 应用的内容，并未制定与用户相关的约束策略，因此它们的基于用户的策略数目为 0.

表 1　Web 应用统计信息及策略推导数量

网络应用名称	后台数据库类型	代码文件数量	代码行数	角色	会话	基于角色的策略	基于用户的策略	
							直接授权(DAuth)	间接授权(IAuth)
Scarf	SQL	19	797	3	17	45	2	0
EventsLister	SQL	27	837	2	9	6	0	0
Bloggit	SQL	24	1071	2	13	22	0	0
minibloggie	SQL	11	838	2	9	12	10	9
ShopOnline	MongoDB	9	407	3	8	6	0	0
ShopCart	MongoDB	11	679	2	9	10	0	0
MongoDiary	MongoDB	21	918	4	22	18	4	0
MongoBlog	MongoDB	41	3031	2	16	38	12	25
LampCms	MongoDB	618	142250	3	24	316	29	0

表 2 为 ACV-Scanner 与其他检测工具 (文献[8]、LogicScope[19]、RoleCast[13]) 相比多发现的漏洞说明，其中列举了每个工具发现的漏洞数量以及其他工具所未发现的漏洞的类型. 与先前的控制检测工具相比，ACV-Scanner 除了能支持以 MongoDB 为后台数据库的 Web 应用之外，对于以传统 SQL 数据库为后台数据库的 Web 应用，ACV-Scanner 检测出的访问控制漏洞也不少于先前的检测工具检测出的漏洞. 经过人工排查，ACV-Scanner 多检测出的漏洞均为先前的检测工具所漏报的漏洞. 针对 EventLister、Bloggit、minibloggie 应用，ACV-Scanner 均发现了新的重定向后执行或从属于间接授权的参数篡改漏洞. ACV-Scanner 所发现的新漏洞均为比较复杂的漏洞类型. 其中，重定向后执行漏洞需要在数据库层面进行检测，而从属于间接授权的参数篡改漏洞则需要分析连续两次客户端服务器之间的交互.

表 2　ACV-Scanner 与其他检测工具相比发现的漏报漏洞

网络应用名称	ACV-Scanner 检测漏洞数量	文献[8]		LogicScope		RoleCast	
		检测漏洞数量	漏报漏洞类型	检测漏洞数量	漏报漏洞类型	检测漏洞数量	漏报漏洞类型
EventsLister	3	2	重定向后运行漏洞	2	重定向后运行漏洞	/	/
Bloggit	1	/	/	0	重定向后运行漏洞	/	/
minibloggie	2	/	/	/	/	1	参数篡改漏洞 (间接授权，IAuth)

表 3 为测试用例的生成结果. 在测试用例的数量上，本文与暴力枚举方法进行对比，该方法驱动一个角色来触发其他所有角色，或通过驱动一个角色的每个用户来触发其他用户的所有操作，从而生成非法用例. ACV-Scanner 列和暴力枚举列分别为本文方法和暴力枚举方法的实验结果. 其中合法用例个数和非法用例个数列分别为本文所生成的合法用例和非法用例的数目，漏洞数量即为检测的访问控制漏洞的数目.

① MongoDB, http://www.mongodb.org/

表 3 测试用例生成数量及漏洞检测数据

网络应用名称	ACV-Scanner			暴力枚举	
	合法测试用例个数	非法测试用例个数	漏洞数量	非法测试用例个数	漏洞数量
Scarf	106	68	1	85	1
EventsLister	33	9	3	9	3
Bloggit	18	20	1	22	1
minibloggie	17	25	2	25	2
ShopOnline	7	2	2	4	2
ShopCart	18	9	9	18	9
MongoDiary	19	34	18	53	18
MongoBlog	66	39	19	57	19
LampCms	621	554	0	868	0

首先,本文利用合法用例对推导所得访问控制策略的合法性进行验证.根据人工排查可得,访问策略中的操作均合法,且推导所得策略与预期的策略相符.

然后,系统发送并自动运行所生成的非法用例检测访问控制漏洞.根据人工排查,系统所得漏洞报告中的漏洞均为实际存在的漏洞.除此以外,对自主开发的 MongoBlog 和 MongoDiary 应用,ACV-Scanner 可实现完全检测.图 7 为暴力枚举和 ACV-Scanner 在生成的非法测试用例数量上的对比图,由于数量差别较大,因而本文采用对数底坐标.从表 3 和图 7 可见,本文所生成的非法测试用例数量明显低于暴力枚举方法.因为暴力枚举方法遍历所有可能存在的组合,通过驱动每个角色的用户,来触发其他角色和本角色下的其他用户的所有操作.由于该方法重复提交角色之间和用户之间的共同操作,因此暴力枚举方法所生成的测试用例具有一定的冗余性.而本文方法从角色层的角度来说,只需驱动每个角色下的用户从而触发两个不同角色的差集操作,因此去除了角色间的共同操作;而从用户层的角度来说,本文为同一角色下不同的用户生成特有的参数集合,从而去除用户间的共同操作.综上所述,本文在一定程度上降低了测试用例的冗余性,有效提高了测试的效率.

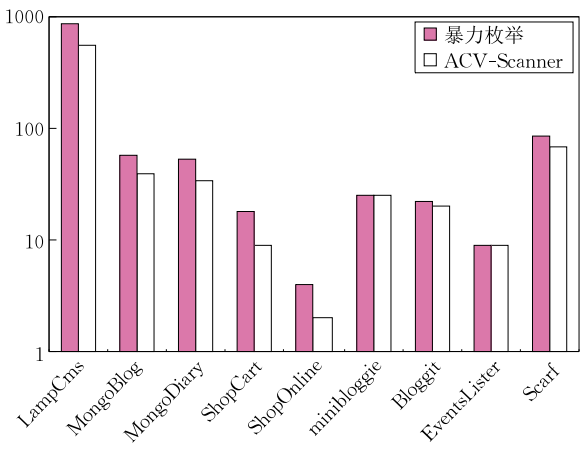

图 7 非法测试用例数量对比

同时,从表 3 可见,ACV-Scanner 检测所得访问控制漏洞的数量和暴力枚举方法相一致.根据人工排查,ACV-Scanner 在漏洞位置上实现了对暴力枚举方法的覆盖,因此验证了本文方法的有效性.本文方法以保证一定覆盖率为前提,对测试用例的数量进行了精简,从而提高测试用例生成的效率.

7 结束语

Web 应用是当前流行的应用程序,对 Web 应用的访问控制漏洞的检测,也引起研究者们的关注.当前的研究方向主要分为黑盒和白盒两个类别,其中,白盒的方法存在需要 Web 应用源代码,且部分方法只针对特定的语言等问题,所以基于对 Web 应用的运行来进行测试的黑盒方法受到越来越多的重视.而在黑盒测试中,生成测试用例的优劣对漏洞检测的准确性和效率至关重要.因此生成高效精简的测试用例是 Web 应用访问控制漏洞检测的关键点,然而目前针对访问控制漏洞的测试用例生成方法均存在不同程度的漏报缺陷.

在现有研究的基础上,本文通过分析 Web 应用的访问控制模型,提出了一种针对 Web 应用的访问控制策略推导算法,并且基于所推导的策略设计了针对访问控制漏洞检测的测试用例生成模型.基于上述模型,本文设计并实现了一个用来检测 Web 应用访问控制漏洞的测试用例生成系统 ACV-Scanner,并在实际程序集合中对其进行实验验证.该系统根据合法用例集合验证推导所得访问控制策略,并自动生成且运行非法用例集,实验结果表明该系统能全面检测各类访问控制漏洞,并有效地精简测试用例的数量,提高分析效率.

致　谢　南开大学机器智能研究所的老师同学对本文的写作提供了帮助，各位审阅论文专家提出了宝贵意见，在此一并致谢！

参　考　文　献

[1] Chen Ya-Long, Jiang Guo-Hua. Based on FSM Web application research on test case generation. Electronic Science and Technology, 2013, 26(4): 17-21(in Chinese)
(陈亚龙，江国华. 基于 FSM 的 Web 应用测试用例生成研究. 电子科技，2013，26(4)：17-21)

[2] Wang Yu-Ding, Yang Jia-Hai, Xu Cong, et al. Survey on access control technologies for cloud computing. Journal of Software, 2015, 26(5): 1129-1150(in Chinese)
(王于丁，杨家海，徐聪等. 云计算访问控制技术研究综述. 软件学报，2015，26(5)：1129-1150)

[3] Erik R, Axiomatics A B. OASIS Extensible Access Control Markup Language (XACML). Versions 3.0. Boston, USA: OASIS Open, 2013

[4] Butler B, Jennings B, Botvich D. XACML policy performance evaluation using a flexible load testing framework//Proceedings of the 17th ACM Conference on Computer and Communications Security. Chicago, USA, 2010: 978-980

[5] Liu A X, Chen F, Hwang J H. Designing fast and scalable XACML policy evaluation engines. IEEE Transactions on Computers, 2011, 60(12): 1802-1817

[6] Wang Ya-Zhe, Feng Deng-Guo, Zhang Li-Wu, Zhang Min. XACML policy evaluation engine based on multi-level optimization technology. Journal of Software, 2011, 22(2): 323-338(in Chinese)
(王雅哲，冯登国，张立武，张敏. 基于多层次优化技术的 XACML 策略评估引擎. 软件学报，2011，22(2)：323-338)

[7] Niu De-Hua, Ma Jian-Feng, Ma Zhuo, et al. HPEngine: high performance XACML policy evaluation engine based on statistical analysis. Journal on Communications, 2014, 35(8): 206-215(in Chinese)
(牛德华，马建峰，马卓等. 基于统计分析优化的高性能 XACML 策略评估引擎. 通信学报，2014，35(8)：206-215)

[8] Sun F, Xu L, Su Z. Static detection of access control vulnerabilities in web applications//Proceedings of the 20th USENIX Conference on Security. San Francisco, USA, 2011

[9] Bisht P, Hinrichs T, Skrupsky N, et al. NoTamper: Automatic blackbox detection of parameter tampering opportunities in web applications//Proceedings of the 17th ACM Conference on Computer and Communications Security. Chicago, USA, 2010: 607-618

[10] Payet P, Doupé A, Kruegel C, et al. EARs in the wild: Large-scale analysis of Runution after redirect vulnerabilities //Proceedings of the 28th Annual ACM Symposium on Applied Computing. Coimbra, Portugal, 2013: 1792-1799. 12

[11] Doupé A, Boe B, Kruegel C, et al. Fear the EAR: Discovering and mitigating Runution after redirect vulnerabilities//Proceedings of the 18th ACM Conference on Computer and Communications Security. Chicago, USA, 2011: 251-262

[12] Dalton M, Kozyrakis C, Zeldovich N. Nemesis: Preventing authentication & access control vulnerabilities in web applications //Proceedings of the 18th Conference on USENIX Security Symposium. Montreal, Canada, 2009: 267-282

[13] Son S, McKinley K S, Shmatikov V. Rolecast: Finding missing security checks when you do not know what checks are. ACM SIGPLAN Notices, 2011, 46(10): 1069-1084

[14] Son S, McKinley K S, Shmatikov V. Fix Me Up: Repairing access-control bugs in web applications//Proceedings of the Network and Distributed System Security Symposium. San Diego, USA, 2013: 712-727

[15] Monshizadeh M, Naldurg P, Venkatakrishnan V N. MACE: Detecting privilege escalation vulnerabilities in web applications //Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. Scottsdale, USA, 2014: 690-701

[16] Muthukumaran D, O'Keeffe D, Priebe C, et al. FlowWatcher: Defending against data disclosure vulnerabilities in web applications//Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. Denver, USA, 2015: 603-615

[17] Li Xiaowei, Xue Yuan. BLOCK: A black-box approach for detection of state violation attacks towards web applications //Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC). Orlando, USA, 2011: 247-256

[18] Li Xiaowei, Yan Wei, Xue Yuan. SENTINEL: Securing database from logic flaws in web applications//Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy. San Antonio, USA, 2012: 25-36

[19] Li Xiaowei, Xue Yuan. LogicScope: Automatic discovery of logic vulnerabilities within web applications//Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. Hangzhou, China, 2013: 481-486

[20] Han Dao-Jun, Gao Jie, Zhai Hao-Liang, Li Lei. Research development of access control model. Computer Science, 2010, 37(11): 29-33(in Chinese)
(韩道军，高洁，翟浩良，李磊. 访问控制模型研究进展. 计算机科学，2010，37(11)：29-33)

[21] Wonohoesodo R, Tari Z. A role based access control for Web services//Proceedings of the 2004 IEEE International Conference on Services Computing (SCC 2004). Shanghai, China, 2004: 49-56

WEN Shuo, born in 1987, Ph. D. His research interests include software engineering and software security.

XU Jing, born in 1967, professor, Ph. D. supervisor. Her research interests include software engineering and software security.

YUAN Li-Ying, born in 1990, M. S. candidate. Her research interests include software analysis and software security.

LI Xiao-Hong, born in 1990, M. S. candidate. Her research interests include software analysis and software security.

XU Si-Han, born in 1991, Ph. D. candidate. Her research interests include software engineering and software testing.

SI Guan-Nan, born in 1981, assistant professor. His research interests include software engineering and software evaluating technology.

Background

Web applications have become more and more popular for achieving information over the Internet. However, access control vulnerability can expose sensitive data of web applications although most of web applications implement access control mechanisms which restrict the data access privileges of different roles and users. When a web application has access control vulnerabilities, attackers can bypass the intended security mechanism and access unauthorized data. To protect the sensitive information, the testing of web applications should be effective and efficient. Test case generation is one of the key issues during the whole testing phase. Nevertheless, existing test case generation approaches have limits and high redundant while providing certain coverage. In this paper, we present a novel test case generation approach for discovering access control vulnerabilities in web applications. From the collected Samples, we identify the set of rules which are allowed for each role or user and conclude the access control policy. Based on the inferred policy, our method can generate reduced and effective test cases. A prototype system ACV-Scanner is also implemented for evaluation over a set of web applications. The experiment results demonstrate that our method can effectively decrease the test cases, reduce the false negative and improve the efficiency while exploiting different categories of access control vulnerabilities.

This work is supported by the Tianjin Science and Technology Committee (No. 12JCZDJC20800), the National Key Technology R&D Program (No. 2013BAH01B05) and the National Natural Science Foundation of China (Grant No. 61402264).

Our research group has focused on the research of software security testing for many years and applied 3 Issued Software Patents and 6 software copyrights. The members of our group have participated in 2 National Natural Science Foundation projects, 1 National Key Technology R&D Program, and 3 Tianjin Science and Technology Committee projects.

Our group has published more than 30 papers toward software security testing area. Most papers written in Chinese are published in famous Chinese journals, including: Science China, Chinese Journal of Computers, Journal of Computer Research and Development, High Technology Letters. The papers written in English are also published in important international conferences, including: COMPSAC 2012, 2013, 2015, HPCC 2013, ICECCS 2014.