计算机科学
COMPUTER SCIENCE

基于站点地图的Web访问控制漏洞检测方法

任家东, 李尚洋, 任蓉, 张炳, 王倩

引用本文

任家东, 李尚洋, 任蓉, 张炳, 王倩. 基于站点地图的Web访问控制漏洞检测方法[J]. 计算机科学, 2024, 51(9): 416-424.

REN Jiadong, LI Shangyang, REN Rong, ZHANG Bing, WANG Qian. Web Access Control Vulnerability Detection Approach Based on Site Maps [J]. Computer Science, 2024, 51(9): 416-424.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230900075

基于站点地图的 Web 访问控制漏洞检测方法

任家东[1,2]　李尚洋[1]　任　蓉[1]　张　炳[1,2]　王　倩[1]
1 燕山大学信息科学与工程学院　河北 秦皇岛 066004
2 河北省软件工程重点实验室　河北 秦皇岛 066004
(jdren@ysu.edu.cn)

摘　要　攻击者通常利用 Web 应用程序的访问控制漏洞实现对系统的非授权访问、信息窃取等恶意行为。针对 Web 应用程序的访问控制漏洞的检测问题,现有方法由于页面覆盖率低、检测过程开销大等问题,因此漏报率过高且效率低下。为此,基于动态分析,提出了一种基于站点地图的 Web 访问控制漏洞检测方法。该方法首先为不同角色下的用户分别建立各自的站点地图,并形成不同角色的完整站点地图,再通过对其分析生成 Web 应用程序预期访问控制策略,构建非法测试用例进行动态访问并分析执行结果实现对未授权访问、越权访问等类型访问控制漏洞的检测。最后,在 7 个真实开源 Web 应用程序中对所提方法进行验证,结果表明该方法能有效降低开销,其页面覆盖率达到 90%以上;发现了 10 个真实漏洞,准确率达到了 100%。

关键词:访问控制;站点地图;测试用例;漏洞检测;CVE 分析
中图分类号　TP311

Web Access Control Vulnerability Detection Approach Based on Site Maps

REN Jiadong[1,2],LI Shangyang[1],REN Rong[1],ZHANG Bing[1,2] and WANG Qian[1]
1 School of Information Science and Engineering,Yanshan University,Qinhuangdao,Hebei 066004,China
2 The Key Laboratory of Software Engineering of Hebei Province,Qinhuangdao,Hebei 066004,China

Abstract　Attackers usually exploit access control vulnerabilities in web applications to gain unauthorized access to systems or engage in malicious activities such as data theft. Existing methods for detecting access control vulnerabilities in web applications suffer from low page coverage and high detection overhead,resulting in high false negative rates and inefficient performance. To address this issue,a web access control vulnerability detection method based on sitemap is proposed using dynamic analysis. The method starts by establishing separate site maps for different user roles and combining them to create comprehensive site maps for each role. Then,by analyzing the site maps,the expected web application access control strategies are derived. Illegal test cases are constructed to dynamically access and analyze the execution results,enabling the detection of unauthorized access and privilege escalation vulnerabilities. Finally,the proposed method is validated on seven real-world open-source web applications. The results demonstrate that this approach significantly reduces overhead,achieves a page coverage rate of over 90%,and successfully detects 10 real vulnerabilities with a recall rate of 100%.

Keywords　Access control,Site maps,Test cases,Vulnerability detection,CVE analysis

1　引言

网络安全愈发受到软件漏洞的严重威胁,《瑞星 2020 年中国网络安全报告》指出[1],国家信息安全漏洞共享平台收录安全漏洞数量共计 20 704 个,较上年漏洞收录总数 16 190 个增长了 27.9%,其中 Web 应用漏洞排名第二(占 29.5%)。开放式 Web 程序安全项目组织(OWASP)于 2021 年发布的十大 Web 应用安全风险榜单[2]中失效的访问控制漏洞从 2017 年榜单的第五位升至第一位,其结果显示 94%的应用程序都遭受过某种破坏访问控制机制的攻击,因而访问控制漏洞被认为是当今 Web 应用程序管理敏感信息面临的最严重的安全威胁之一。因此,对其分析和检测具有重大意义。

理论上,目前的访问控制机制[3-4]能够防止未经授权的攻击者访问应用程序,然而实际上许多应用程序并没有遵循完备的权限验证步骤,或采用了逻辑不完备的访问控制机制进行安全防控,从而导致应用程序存在访问控制漏洞,使得攻击

到稿日期:2023-09-13　返修日期:2024-07-15
基金项目:国家自然科学基金面上项目(62376240);河北省省级科技计划资助(226Z0701G,236Z0702G,236Z0304G);河北省自然科学基金(F2022203026,F2022203089);河北省高等学校科学技术研究项目(BJK2022029);河北省创新能力提升计划项目(22567637H)
This work was supported by the National Natural Science Foundation of China(62376240),S&T Program of Hebei(226Z0701G,236Z0702G,236Z0304G),Natural Science Foundation of Hebei Province,China(F2022203026,F2022203089),Science and Technology Project of Hebei Education Department(BJK2022029) and Innovation Capability Improvement Plan Project of Hebei Province(22567637H).
通信作者:张炳(bingzhang@ysu.edu.cn)

者能够通过特定技术非法控制系统或窃取数据。

针对 Web 应用程序的访问控制漏洞检测，静态分析方法[5-8]主要通过对程序源代码进行直接分析来发现其中的缺陷，但其无法捕获程序实际参数变化，存在较高的误报率与人工参与度。此外，在工业实践过程中，很多程序源代码是闭源的，难以获取。相比之下，动态分析方法[9-15]则不需要程序的源代码，整个测试过程基于对 Web 应用的运行，并具有注重程序的功能等优点。但其在 Web 访问控制漏洞检测分析过程中，对动态执行过程的爬虫获取和监测也存在不足：1）网络爬虫访问应用时只对页面上的链接进行分析，未对页面所包含的表单进行处理；2）网络爬虫对应用进行爬取时未考虑页面重复爬取问题，因此很难找到隐藏页面和不完整的页面，导致获取的用户预期访问权限并不完整。在新兴的人工智能算法分析方面，一种基于机器学习的新的漏洞检测方法[16]利用现有的访问日志数据，从中提取域输入规范策略，并对其进行扩展以生成更多的访问请求，训练机器学习模型以进一步优化访问控制。然而，这种方法仍然需要进行大量的人工分析才能验证推断出的访问控制规则的真实可信性。

结合上述问题，本文基于动态分析的优势，提出了一种基于站点地图的访问控制漏洞检测方法。该方法首先采用改进的网络爬虫技术模拟不同角色下的用户访问应用程序，分析爬虫过程中收集到的请求与响应，然后构建角色级别与用户级别的站点地图，推导出应用程序的访问控制策略，构建针对不同类型访问控制漏洞的非法测试用例进行攻击，检测应用程序中潜在的访问控制漏洞。本文的主要贡献如下：

1）针对现有网络爬虫存在的爬取效率与覆盖率低的问题，提出基于 URL 语义相似度的页面去重方法，设计基于 selenium-wire 工具的自动化表单填充方法，使网络爬虫的覆盖率与效率得到提升。

2）利用收集到的用户执行轨迹构建站点地图，推导出应用程序的访问控制策略，并利用基于 HTTP 协议的身份验证机制，提出基于站点地图的测试用例生成技术。通过对合法测试用例的定向修改构建非法测试用例，降低非法测试用例构建的随机性，实现对访问控制漏洞的检测。

3）在 7 个真实开源 Web 应用程序上的评估结果表明，所提方法能有效发现 Web 应用程序中潜在的访问控制漏洞，能准确地识别出 10 个真实的访问控制漏洞，准确率达到了 100%。

2 Web 访问控制及漏洞分类

2.1 Web 访问控制

在 Web 应用程序的核心安全机制中，访问控制的实现建立在身份验证和会话管理之上，当访问控制机制存在缺陷时，攻击者通常会危及整个应用程序，并访问属于其他用户的敏感数据。访问控制技术指通过开发者制定的访问控制策略来判定请求服务的用户是否具有访问服务以及资源信息的权限[17]，并规定不同权限用户所能执行的访问操作集合，以防止系统被非法操作访问，从而保护系统的数据资源。

基于角色的访问控制模型[18-20]（Role-Based Access Control，RBAC）指按照角色对用户进行归类，即一个用户所具有的操作权限由用户所属的角色决定。若某用户属于多个角色，则该用户享有这些角色的操作权限，即其操作权限集合为这些角色的并集。基于角色的访问控制模型有 3 个基本元素，分别是用户、角色与权限。三者之间的关系如图 1 所示。RBAC 为用户提供细粒度控制，并为访问管理提供简便和可管理的方式，相对于单独分配权限而言，此法不易出错，能确保用户只执行完成任务所需的操作。

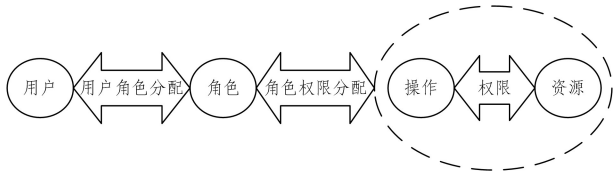


图 1 基于角色的访问控制模型

Fig. 1 Role-based access control model

2.2 Web 访问控制漏洞分类

访问控制依赖于应用程序本身，不同应用程序的访问控制实现逻辑有很大不同，而访问控制漏洞主要分为以下 3 种情况：

1）认证绕过：攻击者利用访客身份向权限页面发送访问请求，而不提供合法凭证，以获得非本用户所能访问的数据资源。

2）垂直越权：攻击者使用一个只有低权限的账户，采取参数篡改的方式向权限页面发送访问请求，获取到只有高权限用户才能够访问的数据资源。

3）水平越权：攻击者使用某一用户访问凭证向其他同级权限用户私有页面发送访问请求，获取到该用户私有的数据资源。

3 方法

3.1 研究框架

本文的结构框架包括执行轨迹收集、站点地图生成和越权漏洞检测 3 个部分，如图 2 所示。

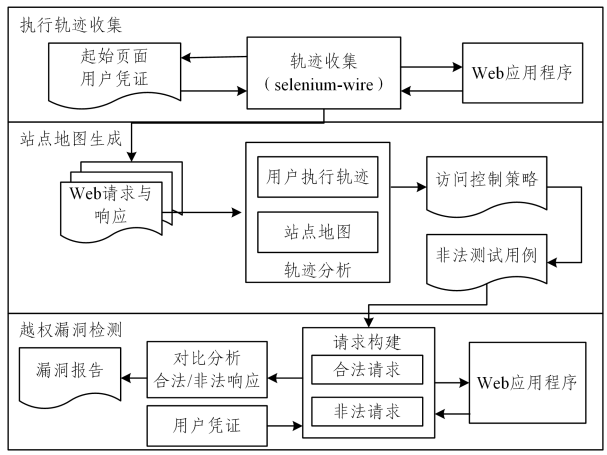


图 2 所提方法框架示意图

Fig. 2 Schematic diagram of the proposed method

3.2 执行轨迹收集

执行轨迹收集主要通过有针对性的网络爬虫技术来进行，旨在收集不同角色下用户访问 Web 应用程序的行为轨迹，以及访问过程中 Web 应用发送的访问请求与收到的页面响应，用于构建其对应的站点地图，从而进一步推导 Web

应用的访问控制策略。爬虫技术改进点具体为:爬虫过程中对访问的页面进行表单检测,对存在填充组件的页面进行自动填充与提交,模拟用户对应用程序进行功能操作,提高爬虫系统爬取页面的覆盖率;对爬虫过程中提取到的页面地址进行模糊处理,过滤掉语义相同的链接,避免由于应用中页面链接存在环路而重复爬取相同页面链接,从而提高爬虫系统的爬取效率。

3.3 站点地图构建

为了构建完整的站点地图,在爬虫期间需要收集不同身份权限用户与应用程序之间的交互轨迹。本文在网络爬虫模拟不同身份权限用户访问应用程序时,使用 selenium-wire 获取用户访问应用程序的 HTTP 请求与响应,并从中提取用户访问 Web 应用时的执行轨迹,分析执行轨迹生成站点地图。其中执行轨迹生成流程如算法 1 所示,执行轨迹如 *track* 所示。

算法 1 执行轨迹生成算法

输入:起始页面 N,等待队列 Q,角色集合 SetR,用户集合 SetU,用户身份凭证 I

输出:执行轨迹 track

```
1. FOR 角色集合 SetR 中每个角色 Ri
2.   FOR 用户集合 SetU 中具有角色 Ri身份的每个用户 Uj
3.     WHILE Ui is browsering
4.       response=Ui. request(N,I);
5.       urls=Extractor(response);
6.       Q. add(de-emphasis(urls));
7.       track. add(〈N,request. method,request. referer,request. data,request. cookie,Ri,Uj〉);
8.       IF(response. exist('form'))
9.         new-url=autofill(N);
10.    ENDWHILE
11.  ENDFOR
12. ENDFOR
```

$$track=\{\langle url,method,referer,data,cookie,R_i,U_i\rangle \mid R_i\in SetR,U_i\in SetU\}$$

算法 1 遍历每个角色 R_i 下所有用户 U_i,通过模拟用户访问应用中每个角色被授予访问权限的页面,在访问应用程序的过程中始终携带同一用户身份凭证,模拟用户访问时会始终保持一种用户身份探索整个应用,同时抓取应用程序与服务器的交互数据,得到目标应用程序的完整执行轨迹集合。此时 track 存储了用户的全部执行轨迹,通过遍历并分析每个执行轨迹,依据其携带的身份标识,将执行轨迹的信息加入相应的站点地图中,生成目标应用程序中角色与用户的完整站点地图。其具体实现过程如算法 2 所示,其中站点地图如 *sitemap* 所示。

算法 2 站点地图生成算法

输入:爬虫轨迹 track,用户集合 SetU,角色集合 SetR

输出:站点地图 sitemap

```
1. FOR 用户集合 SetU 中具有角色 Ri身份的每个用户 Ui
2.   FOR 爬虫轨迹 track. get(Ui)中的每个 tracki
3.     IF(tracki. contains(Ri))
4.       sitemap. get(Ri). add(tracki);
5.     IF(tracki. contains(Ui))
6.       sitemap. get(Ui). add(tracki);
7.     ENDIF
8.   ENDFOR
9. ENDFOR
```

$$sitemap=\{Node,Edge,Params\}$$

$$Node=\{url_1,url_2,\cdots,url_n\}$$

$$Edge=\{\langle url_i,url_j\rangle \mid url_i,url_j\in Node\}$$

$$Params=\{\langle p_i\rangle \mid i\in[1,n]\}$$

其中,*Node* 表示用户权限页面集合;*Edge* 表示各个节点间的跳转关系,即为边;*Params* 表示页面访问过程中传递的参数信息集合,边上的 p_i($p_i\in Params$)由六元组〈*url*,*method*,*data*,R_i,U_{ij},*cookie*〉表示;R_i 为当前用户的角色;U_{ij} 表示当前授权用户。*cookie* 为用户 U_{ij} 在访问过程中的身份凭证;*url* 为页面来源,对于边〈url_i,url_j〉,页面来源即为 url_i;*method* 为页面请求方式;*data* 为页面请求参数信息。

3.4 访问控制漏洞检测

当 Web 应用在加载特权操作页面之前缺乏对用户身份权限的验证,攻击者能够依据该页面实现对应用数据的增删改查操作而无须费力获得一个合法账户,进而窃取并破坏 Web 应用系统中存储的信息,这就存在访问控制漏洞。本文从攻击者的角度出发,模拟其非法访问 Web 应用,根据非法测试用例构建非法请求并向服务器发送请求以访问系统数据资源,然后拦截服务器反馈的响应信息,通过对比非法请求响应与合法请求响应,进而识别 Web 应用中存在的访问控制漏洞。

3.4.1 访问控制策略推导

在 Web 应用中配置访问控制策略时,应该为不同角色授予不同页面的访问控制权限,高权限角色的用户具有访问某些页面的权限,而低权限角色中的用户不可以执行这些访问操作。针对这些页面,由于应用中低权限用户无权访问它们,因此在开发应用时,开发人员常常以各种方式向低权限用户隐藏这些页面的链接。故在低权限用户正常访问 Web 应用时,并不会向这些页面发送访问请求。Web 应用中不同角色下的用户在访问应用时会根据其被授予的权限向不同的页面发送请求,进而访问应用的不同页面。

本文提出了一种基于站点地图的访问控制策略生成方法,该方法利用站点地图中的节点操作信息,为每个角色与用户生成相应的访问控制策略。通过遍历角色下用户访问应用所构建的站点地图,根据节点对应的操作生成相应的访问控制策略。在此过程中,特别考虑了访客的特殊身份,通过去除访客所能进行的公共访问操作,确保了访问控制策略的安全性和灵活性。其实现流程如算法 3 所示,其中访问控制策略如 *SetPolicy* 所示。

算法 3 基于角色的访问控制策略生成算法

输入:角色集合 SetR,站点地图 sitemap

输出:访问控制策略 SetPolicy

```
1. FOR SetR 下每个角色 Ri
2.   Policy. addRole(Ri);
3.   WHILE 遍历 sitemap. get(Ri)
4.     IF(! Policy. contains(url))
5.       Policy. add(url,method,data,Uij,cookie);
6.       ENDIF
7.     SetPolicy. get(Ri). add(Policy);
```

8. ENDFOR
9. SetPolicy. get(R_i)＝SetPolicy. get(R_i)－SetPolicy. get(Guest)；
10. END

$$SetPolicy=\{\langle url,method,data,R_i,U_{ik},cookie\rangle \mid R_i \in SetR, U_{ik} \in SetU\}$$

用户约束的实现是通过操作参数来限制用户访问的页面视图。在 Web 应用中，部分页面是和用户的身份相关联的，因此可以通过保证不同用户访问的页面不同，从而实现用户级别的访问控制。当携带不同身份凭证的用户访问 Web 应用时，服务器收到用户的请求参数不尽相同，就会得到不同的页面视图。例如 *view. php?userid*＝12 就是运用参数 *user-id* 实现用户约束，将访问的页面限制在 *userid* 为 12 的用户上传图片的视图。为了保证上述约束正确实施访问控制策略，应在用户请求页面时，检查当前系统授权用户与发送请求中的用户约束是否一致。反之，若在请求页面时缺乏对用户身份和用户约束的检查，当用户随意修改页面请求中的参数值，且修改后的参数值恰与某一用户的信息相同，该用户就请求到了另一用户的资源，造成水平越权漏洞。本文通过对不同用户站点地图的遍历，得到用户级别的访问控制策略，其实现过程类似于角色级别的访问控制策略推导方法。

3.4.2 测试用例生成

在 Web 应用中，访问控制策略实施的关键在于对页面请求进行处理时是否对用户的访问权限进行检查。为了检测目标 Web 应用中可能存在的访问控制漏洞，需要构建测试用例以模拟用户对应用进行合法与非法访问。本文提出了一种基于站点地图的测试用例生成方法，利用推导的访问控制策略生成合法测试用例，分别违背角色与用户层次的策略约束，从而生成非法测试用例。最常用的访问控制漏洞检测方法为强制浏览，此时攻击者尝试直接访问特定的网页来获取数据信息。本文依据推导的访问控制策略构建针对不同漏洞的非法测试用例，避免通过直接在浏览器的地址栏输入请求地址进行访问控制漏洞检测。对于构造基于角色的非法测试用例，根据推导的访问控制策略构造合法测试用例，之后遍历高权限角色的合法测试用例，将测试用例中用户身份凭证定向修改为低权限用户身份凭证，并保持其他参数不变，从而构建违反角色级别策略约束的非法测试用例。

基于角色的非法测试用例生成时，需要对其他角色的访问控制策略的操作集进行遍历。对比分析当前角色与其他角色的操作集，获取当前角色所独有的操作权限集，之后选取两个角色下用户通过违反当前角色的访问控制策略构建非法测试用例。其具体实现如算法 4 所示，非法测试用例如 *illegalTC* 所示。

$$illegalTC=\{\langle url,method,data,R_i,U_i,R_j,U_j\rangle \mid R_i,R_j \in SetR, U_i,U_j \in SetU\}$$

算法 4 基于角色的非法测试用例生成算法

输入：角色集合 SetR，用户集合 SetU，访问控制策略 SetPolicy
输出：非法测试用例 illegalTC

1. FOR SetR 下每个角色 R_i
2. 从 SetPolicy. get(R_i)中获取页面操作集合 P_i；
3. FOR SetR-R_i中的每个角色 R_j
4. 从 SetPolicy. get(Rj)中获取页面操作集合 P_j；
5. SetDiff＝P_i－P_j；
6. FOR SetDiff 下每个操作 op
7. 分别选取 R_i、R_j下的用户 U_i、U_j；
8. TC＝generate(op，R_i，U_i，R_j，U_j)；
9. illegalTC. get(R_i). add(TC)；
10. ENDFOR
11. ENDFOR
12. ENDFOR

其中，R_i 和 R_j 分别表示不同权限等级的角色，U_i 和 U_j 分别表示角色 R_i 和 R_j 下的用户，*url* 表示角色 R_i 的特权页面，*method* 和 *data* 分别表示用户 U_i 访问特权页面 *url* 的请求方式与请求参数。

基于用户的非法测试用例生成时，需要遍历当前角色下其他用户的访问控制策略，其余操作类似于角色级别的非法测试用例生成方法。图 3 展示了构建非法测试用例的详细过程，首先获取管理员对该页面的访问操作信息与普通用户对公共页面的访问操作信息，之后将管理员对此特权页面的访问操作中的操作信息以及普通用户请求页面的用户身份凭证进行组合，就构建了一个针对垂直越权访问控制漏洞的非法测试用例。其中 *admin. php?mode＝adduser* 页面只允许管理员访问，如果存在完善的访问控制机制，则普通用户不能通过身份权限验证，即该页面会拒绝普通用户的访问操作。

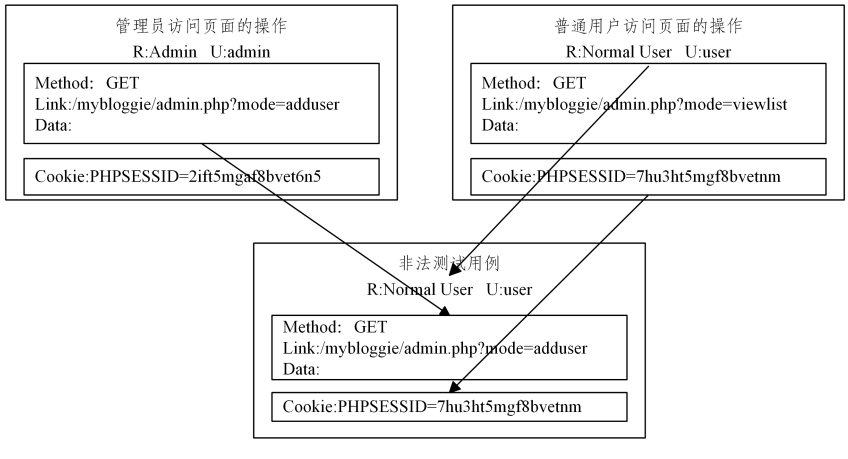


图 3 非法测试用例构建示例

Fig. 3 Example of illegal test case construction

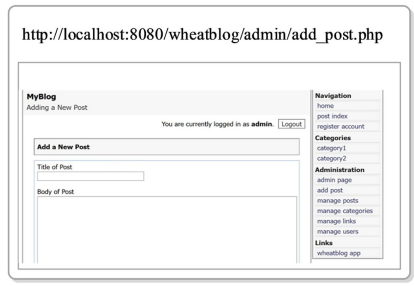
3.4.3 漏洞检测

本文从攻击者的角度出发，模拟其非法访问 Web 应用，根据测试用例构建访问系统数据资源的非法请求，然后拦截服务器反馈的响应信息，通过对比分析非法请求响应与合法请求响应，进而识别 Web 应用中存在的访问控制漏洞。为了验证两次获得的响应是否相同，本文使用一系列检验方法对请求的响应进行分析。

1）响应状态分析：对页面响应进行对比分析时，应先对响应状态进行分析，检查 HTTP 响应的状态码是否代表请求成功以及响应内容是否为拒绝访问页面。首先使用状态码验证器检查请求是否成功。大多数 Web 应用程序使用以下状态代码来响应浏览器请求：允许访问将返回状态码 200；拒绝访问将返回 301，305，403，404，500，503 等状态码。其中，以 3 开头的状态码表示重定向，以 4 开头的状态码表示客户端错误，以 5 开头的状态码表示服务器端错误。如果响应的状态码不是 200，则认为页面访问请求被拒绝，表示该页面不存在访问控制漏洞。若 Web 应用严格按照这些状态码约定返回响应，则可以只按照状态码来确定是否允许用户访问。然而在实际情况下，许多应用程序并不遵循这些惯例。Web 应用程序可以始终使用状态码 200 回复页面请求，这意味着它在拒绝授予访问权限时会向用户显示自定义错误页面。但如果是其他原因导致访问被拒绝，它仍然会按照约定响应相应的状态码，包括服务器拒绝执行请求、服务器无法完成请求、无法定位资源等。故除了状态码 200 之外的其他状态码都被视为拒绝访问页面，但状态码为 200 的页面响应仍需进一步分析。

下一步，所有状态码为 200 的响应都将由访问拒绝检验器进行分析。这个检验器在响应正文中查找是否存在自定义访问拒绝消息，以决定允许和拒绝访问。在一些应用程序中受到非法访问时并不会回复 404 以拒绝访问，而是会正常响应用户请求，但响应的内容是自定义的访问被拒绝的错误信息，此时通过匹配响应内容即可判断其是否为正常响应。若匹配，则认为是拒绝访问的决定，不会标记为漏洞。如图 4 所示，非法请求获得的响应内容为自定义的拒绝访问消息，而合法请求获得的响应内容为添加博客、用户管理等信息。

（a）自定义拒绝访问页面



（b）正常访问页面

图 4 非法访问与正常访问页面响应

Fig. 4 Page response to illegal and normal accesses

2）响应内容分析：除了分析页面响应状态之外，还应该进一步对页面响应内容进行分析，即对页面结构与内容进行对比分析，检查非法请求的响应内容是否与合法请求的响应内容相似，避免将正确行为报告为漏洞。通常可以使用多种方法来比较页面响应的相似性。从图 5 可以看出，页面响应内容是包含 HTML 标签的纯文本文件。最简单快捷的方法是直接比较两个响应的内容是否相等，但这样的对比方式存在一个严重的缺点，即哪怕两个页面响应之间存在极小的差异，它也会认为两个页面响应是不同的。实际上，当用户向应用发送相同的请求时，因为存在动态元素通常不能保证响应完全相同。因此，需要一种更好的方法来处理这种差异。

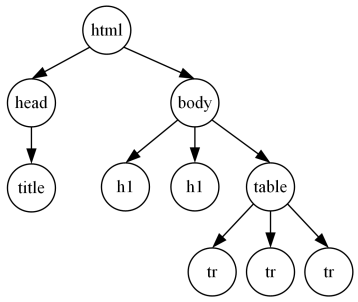
```
<html xmlns = ‘/xhtml’ >
        <head>
                <title>Event Calendar </title>
        </head>
        <body>
                <h1>Event Calendar </h1>
                <div class = “main_area” >
                        <a name = “28” ></a>
                        <span class = “event_name” >Test Event </span>
                        <a href = “http://www.mwvin.com/” > mevin </a>
                </div>
        </body>
</html>
```

图 5 页面响应示例

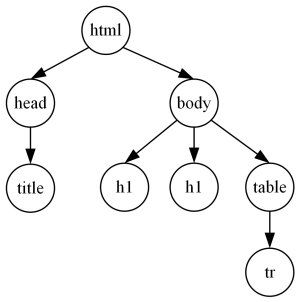
Fig. 5 Example of page response

不同权限级别的用户执行的代码逻辑不同，页面响应中标签的组成决定了网页的 DOM 结构存在差异。一般而言，页面中相似的内容分布在相似的树结构中，对于同一个页面，若两个 DOM 树是相似的，它们的内容也是相似的。由于合法与非法访问向应用传递的参数信息除了用户身份凭证外都是一致的，若页面的代码中缺乏相应的用户身份权限验证，那么服务器处理两个请求的逻辑与反馈的响应结构应该是一致的，则可以通过分析对比响应的 DOM 结构的相似度，来判断页面是否存在访问控制漏洞。例如 U_i 和 U_j 是网上商城中提供商品的商家与商城管理员，U_i 作为商城管理员可以对商家

与顾客的账户信息进行管理，U_j 作为商家仅可以管理自己的账户信息，在以用户 U_j 的用户身份凭证访问此账户管理页面时，服务器只返回具有一个账户信息的列表，若仅使用响应状态验证器就会将其报告为漏洞页面，但从访问控制的角度来看这属于正确行为，通过对比页面响应 DOM 结构，如图 6 所示，可以识别出页面响应内容是不同的，不会报告为访问控制漏洞。



(a)用户 U_i 请求账户管理页面响应 DOM 结构



(b)用户 U_j 请求账户管理页面响应 DOM 结构

图 6 页面响应 DOM 结构

Fig. 6 Page response DOM structure

基于页面响应的访问控制漏洞检测依据测试用例构建合法访问请求与非法访问请求，当其向应用程序发送合法请求时会获得正常响应，该响应用于判断非法请求能否成功访问。当应用程序收到非法请求后会拒绝访问请求或响应访问请求，通过对非法响应的响应状态进行分析判断应用是否拒绝该请求，若拒绝则说明应用程序正确实施了访问控制策略；而当应用程序响应了请求，则要对此响应进一步分析。若非法访问用户与合法访问用户为不同角色时，则通过内容相似度验证器拆解其 DOM 结构，判断其是否与合法响应相似，若相似，则该权限页面可能存在访问控制漏洞；而两个请求来自同一角色下的不同用户时，由于它们的访问控制策略一致，但不同用户间的相互访问会致使应用存在水平越权漏洞，故采用内容相似度验证器对响应内容进行模糊哈希对比，若相似则该权限页面可能为漏洞页面。其实现过程如算法 5 所示。

算法 5 基于页面响应的访问控制漏洞检测算法

输入：非法测试用例 illegalTC 自定义访问错误信息 ErrorCode

输出：漏洞报告 report

```
1. FOR illegalTC 下每一个 TCi
2.    legalRequest,illegalRequest=generate(TCi);
3.    legalResponse=SendRequest(legalRequest);
4.    illegalResponse=SendRequest(illegalRequest);
5.    IF check(illegalResponse.status_code)
6.       IF ErrorCode not in illegalResponse
7.          IF legalRequest.role ! = illegalRequest.role
8.             similary(legalResponse.dom,illegalResponse.dom);
9.             report.add(TCi);
10.            ELSE
11.         similary(tlsh(legalResponse),tlsh(illegalResponse));
12.         report.add(TCi);
14. ENDFOR
```

4 实验与结果

4.1 实验环境及数据集

本文描述的所有实验都是在配备 i5-10210U CPU @ 1.60GHz 的 CPU 处理器和 16GB RAM 的 PC 上运行的，部署 PHP(版本 5.3.10)和 WampServer(版本 2.2)，并在 7 个开源应用程序上进行实验。这些应用程序的基本数据信息如表 1 所列，包括文件数量、角色数量、可执行代码行(LoC)以及漏洞详情。

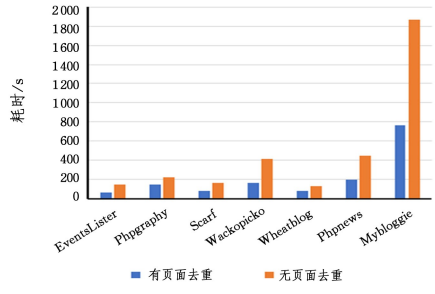
表 1 Web 应用程序信息

Table 1 Web application information

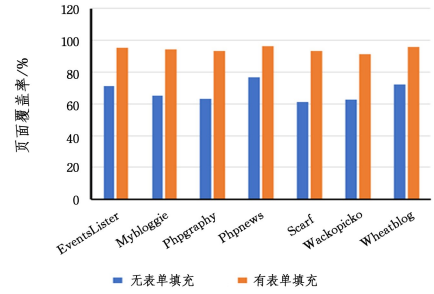
应用程序	版本号	文件数量	可执行代码行	漏洞详情
EventsLister	2.0.3	27	837	CVE2009-3168
Mybloggie	2.1.3	59	6261	CVE2006-4042 CVE2006-4043
OpenInovice	0.8	27	2571	CVE-2008-6524
Phpnews	1.3.0	21	6037	
Scarf	Beta	19	797	CVE2006-5909
WackoPicko	1.0	52	952	
Wheatblog	1.1	42	4801	CVE-2006-5922

4.2 实验结果

本文使用网络爬虫来收集用户访问应用程序的行为轨迹，并针对每个角色使用不同的用户身份凭证来探索应用程序，构建用户级别和角色级别的站点地图。为了更高效地模拟用户对 Web 应用的访问操作，本文在网络爬虫过程中增加了表单自动化填充和页面去重功能。其中网络爬虫具有了表单自动化填充功能，可以模拟用户访问应用程序中的表单处理部分，包括向应用程序发送 GET 和 POST 请求，执行添加和删除数据等操作。这种网络爬虫可以有效地收集应用程序的行为轨迹，若无表单处理操作收集到的执行轨迹，则可能缺乏对某些页面的操作。如图 7 所示，分别对网络爬虫改进前后对目标 Web 应用的页面覆盖率与网络爬虫耗时进行对比，可以看出表单自动填充功能提高了网络爬虫的页面覆盖率，页面去重功能使网络爬虫的爬取效率得到有效提高。原有网络爬虫不具备表单自动化填充功能，导致页面覆盖率一般在 80%以下，如图 7(a)所示；而改进后的网络爬虫大幅提高了页面覆盖率，使得网络爬虫模拟器几乎可以访问 Web 应用的全部页面。由于 Web 应用会在多个页面中存在指向同一页面的链接，使应用中的页面链接存在环路，若在网络爬虫过程中未实现页面去重功能，就会导致重复抓取某一页面；而实现了页面去重功能的网络爬虫，提高了网络爬虫的效率，明显减少了爬取 Web 应用的耗时，如图 7(b)所示。



(a)网络爬虫改进前后耗时



(b)网络爬虫改进前后页面覆盖率

图 7 网络爬虫改进前后性能对比

Fig. 7 Performance comparison before and after web crawle improvement

对于不同的 Web 应用,本文模拟每个角色下的多个用户访问应用,并利用 selenium-wire 收集客户端与服务器之间的请求与响应,以推导 Web 应用的预期访问控制策略。访问控制策略规定了不同用户对应用中页面的访问权限。基于角色的访问控制策略通过验证用户的身份决定了用户可访问的特权页面集,基于用户的访问控制策略通过验证用户的操作参数决定用户可访问的特权页面集。其中特权页面标识访问控制策略将访问该页面视为特权操作,表现为低权限用户无法获取此页面链接且无法访问此页面。表 2 列出了测试程序配置的角色、不同角色下用户与应用交互时产生的 HTTP 请求与响应,其中权限页面数量是相对于低一级权限角色的,例如管理员相对于普通用户,普通用户相对于访客用户。

表 2 Web 应用角色权限信息

Table 2 Role privilege information of Web applications

应用名称	角色	请求响应	权限页面
Mybloggie	visitor	58	0
	user	182	9
	admin	439	20
Scarf	visitor	52	0
	user	admin	63
	203	4	13
EventsLister	visitor	55	0
	admin	145	7
Phpnews	user	37	7
	admin	240	27
OpenInvoice	user	158	19
	admin	200	29
Wackopicko	visitor	107	0
	user	288	14
	admin	292	16
Wheatblog	visitor	34	0
	user	40	2
	admin	193	15

本文在对应用程序访问控制漏洞进行检测时,基于推导出的访问控制策略,构建了基于角色与基于用户的访问控制漏洞测试用例,生成权限页面非法访问请求并收集页面响应,使用页面响应检验方法判定越权访问是否成功,从而判断漏洞是否存在。表 3 列出了生成的测试用例数量、检测到的访问控制漏洞数量与实际存在的访问控制漏洞数量。实验结果表明,该方法能够有效地检测出应用程序中的访问控制漏洞,从而更好地保护应用中隐私数据。

表 3 Web 应用中访问控制漏洞

Table 3 Access control vulnerabilities in Web applications

应用程序	测试用例数量	检测到的漏洞	应用中已知的漏洞
EventsLister	9	2	**2**
Mybloggie	49	2	**2**
OpenInvoice	39	1	**1**
Phpnews	11	0	**0**
Scarf	20	2	**2**
Wackopicko	21	2	**2**
Wheatblog	15	1	**1**

表 4 列出了本文与现有文献的检测结果比较,由于文献使用不同的基准程序,因此有些结果是空白的,应用程序 phpnews 不存在访问控制漏洞,所以没有在表中展示。结果表明,本文方法能够有效地检测出应用程序中的访问控制漏洞,适应更多的 Web 应用程序。

表 4 对比实验结果

Table 4 Comparative experimental results

应用程序	本文	[10]	[5]	[21]	[12]	[22]	[23]	[6]
EventsLister	2		2	2				2
Mybloggie	2					5		
OpenInovice	1			4				
Scarf	2	1	1	1	2	1	1	1
WackoPicko	2	2		2	2			
Wheatblog	1							
Sum	**10**	**3**	**3**	**9**	**4**	**6**	**1**	**3**

以表 4 中的应用程序 Eventslister 为例,页面 eventslister/admin/user_add.php 中的漏洞如图 8 所示,其缺少对用户身份的验证。如果使用 *checkUser*() 函数对访问用户的身份进行检查,当用户身份不是管理员,此访问请求就会被拒绝,重定向到应用程序的登录页面,就不会存在越权访问控制漏洞的问题。

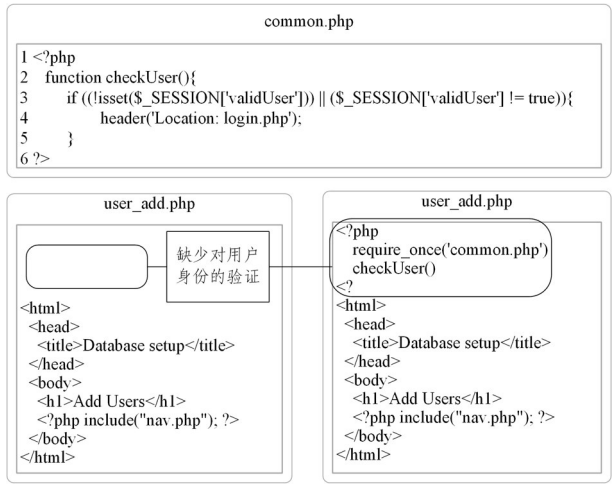


图 8 EventsLister 漏洞案例分析

Fig. 8 Vulnerability case analysis of EventsLister

4.3 访问控制漏洞(CVE)详情分析

根据表 1 中各应用程序访问控制漏洞，本文检测到的漏洞详细分析如下：

Scarf 应用是斯坦福的会议与研究论坛，用于协助用户审阅、储存稿件。该应用程序有 3 个角色：管理员用户、普通用户、访客用户。该应用中使用 *require_loggedin*()函数来验证用户是否为登录状态，用 *require_admin*()函数来验证用户是否具有管理员权限。在该应用程序中，只允许访客用户访问用户注册页面、忘记密码页面、用户登录页面以及查看会议列表、文章列表与文章评论页面；允许普通用户访问个人信息管理页面与发布文章评论页面；允许管理员用户访问文章管理页面、会议管理页面、用户管理页面与评论查看页面。本文在 generaloptions. php 页面中检测到了一个已知的身份验证绕过漏洞(CVE-2006-5909)，该漏洞页面代码中遗漏了用于验证管理员权限的 *require_admin*()函数，且此漏洞允许攻击者作为访客用户或普通用户篡改会议和用户信息。此外，在 comments. php 页面中亦检测到了身份验证绕过漏洞，该漏洞页面代码中同样缺少对用户的身份验证函数，此漏洞允许攻击者作为访客用户或普通用户查看评论列表。

EventsLister 应用是一个事件列表器，用于协助用户管理事件。该应用程序有两个角色：管理员用户和访客用户。该应用中使用 *checkUser*()函数来验证用户是否为管理员用户。在该应用程序中，只允许访客用户访问用户登录页面以及用户密码重置页面；允许管理员用户访问添加事件页面、更新事件页面、删除事件页面、管理用户页面。本文在 user_add. php 页面中检测到了一个已知的身份验证绕过漏洞(CVE-2009-3168)，该漏洞页面代码中遗漏了用于验证用户身份的 *checkUser*()函数，此漏洞允许攻击者作为访客用户添加新的管理员账户。此外，在 recover. php 页面中亦缺乏访问控制检查，并且允许攻击者作为访客用户通过设置请求中参数值的方式重置其密码。

MyBloggie 应用是一个开发简单的用户友好的博客程序。该应用程序有 3 个角色：管理员用户、普通用户、访客用户。该应用中使用 *verifyuser*()函数来验证用户是否为登录状态，使用 $ *userid*['*level*']特权参数来验证用户的身份权限。在该应用程序中，只允许访客用户访问查看博客详情页面与评论发布页面；允许普通用户访问博客发布页面、博客管理页面、评论管理页面以及文件上传页面；允许管理员用户访问博客类别管理页面、添加用户页面、管理用户页面、删除用户页面。本文在 admin. php? mode= edituserlist 与 admin. php? mode=deluserlist 页面中检测到存在垂直越权访问控制漏洞，允许攻击者以普通用户的身份查看用户列表信息。

Wackopicko 应用是一个图片管理系统，用于协助用户上传、购买图片。该应用程序有 3 个角色：管理员用户、普通用户与访客用户。其使用 *require_login*()函数来验证用户身份权限。在该应用程序中，只允许访客用户访问用户注册页面、用户登录页面、留言簿留言发布页面，以及查看上传的图片；允许普通用户访问图片上传页面、图片的评论发布页面、已购买图片查看页面与购物车管理页面；允许管理员访问系统管理页面。本文在应用程序中确定了两个访问控制缺陷：第一个存在于 users/view. php 页面中，它无法检查参数 *userid* 上预期的约束，这允许攻击者通过篡改这个参数来访问任何用户的信息；第二个存在于 highquality. php 页面中，该页面用于显示用户购买的高质量图片，该页面中的漏洞无法检查参数 *pic_id* 的约束，这允许攻击者查看任何高质量的图片。

OpenInvoice 应用是一个小型发票工具，用于跟踪客户、发票与物品。该应用程序有 3 个角色：管理员用户、普通用户、访客用户。该程序使用 *checkLogin*()函数验证用户的登录状态，使用权限参数 $ *_SESSION*['*level*']验证用户身份权限。在该程序中，只允许访客用户访问用户登录页面；允许普通用户访问客户信息页面、发票管理页面、个人信息管理页面、消息管理员页面；允许超级用户与管理员用户查看用户列表并管理用户信息。本文在应用程序中确定了一个访问控制漏洞，在 resetpass. php 页面中检测到一个已知漏洞(CVE-2008-6524)，攻击者可以利用该漏洞以普通用户身份修改任意用户的密码。

Phpnews 应用是一个新闻发布软件，用于管理、评论新闻。其使用 *is_admin*()函数验证用户的身份权限。在该应用程序中，只允许普通用户访问新闻发布页面、新闻列表页面、新闻修改页面与个人信息管理页面；允许管理员用户访问用户添加页面、用户管理页面以及系统配置页面。经过检测，本文未在系统中检测出基于角色与基于用户的访问控制漏洞。

Wheatblog 应用是一个博客软件，用于维护在线博客、期刊与新闻等其他内容。该应用程序有 3 个角色：管理员用户、普通用户、访客用户。其使用 *isAdmin*()函数验证用户身份权限。在该应用程序中，只允许访客用户访问用户注册页面、博客查看页面与评论发布页面；允许管理员用户访问博客发布页面、博客管理页面、类别管理页面、用户管理页面、链接管理页面。本文在应用程序中确定了一个访问控制漏洞，在 index. php 页面存在垂直越权访问控制漏洞，在该漏洞页面代码中遗漏了用于验证用户身份权限的 *isAdmin*()函数，且此漏洞允许攻击者作为访客用户查看管理员主页。

结束语 本文基于测试用例生成技术、网络爬虫技术以及基于 HTTP 协议的身份验证机制，针对现有访问控制漏洞检测技术存在的缺陷与不足，提出了基于站点地图的访问控制漏洞检测方法。该方法提高了网络爬虫的效率与覆盖率，使得对应用程序中的访问控制漏洞的检测具有更高的准确度。

本文在目前爬虫模拟器的基础上，设计了基于 URL 语义相似度的页面去重方法，可以避免网络爬虫反复向功能相似页面发送访问请求，提高网络爬虫效率；提出了基于 selenium-wire 工具的表单自动化填充方法，能更好地模拟用户对应用程序的功能操作，提高网络爬虫覆盖率。在现有测试用例生成方法的基础上，结合应用程序身份验证机制，改进了生成非法测试用例的方法，依据站点地图对合法测试用例进行定向修改，生成更具针对性的访问控制漏洞的非法测试用例。依据测试用例收集分析页面响应，通过一系列验证方法检测出应用程序中潜在的访问控制漏洞。

虽然本文对访问控制漏洞检测方法的研究取得了一定的成果，但仍存在不足之处。本文的网络爬虫方法尚无法真正

实现对用户访问应用程序全部行为的模拟，例如在响应中存在JS加载的动态内容时无法获取此部分内容，且对复杂大型应用程序进行站点地图建模时，开销较大。在以后的研究中，可以完善网络爬虫功能，利用动态技术实现页面的快速处理，并基于访问控制漏洞检测结果，研发访问控制漏洞修复工具以实现对漏洞的修复。

参 考 文 献

[1] Beijing Rising Information Technology Co., Ltd. 2020 China Cyber Security Report[J]. Journal of Information Security Research, 2021, 7(2): 102-109.

[2] OWASP. Open Web Application Security Project 2021[DB/OL]. http://www.owasp.org.cn/OWASP-CHINA/owasp-project/2021-owasp-top-10/.

[3] DEEPA G, SANTHI P. Securing Web Applications from Injection and Logic Vulnerabilities: Approaches and Challenges[J]. Information and Software Technology, 2016, 74(4): 160-180.

[4] ZHANG B, LI J, REN J, et al. Efficiency and Effectiveness of Web Application Vulnerability Detection Approaches: A Review [J]. ACM Computing Surveys(CSUR), 2022, 54(9): 1-35.

[5] SUN F, XU L, SU Z. Static Detection of Access Control Vulnerabilities in Web Applications[C]// Proceedings of the 20th USENIX Conference on Security. Berkeley, CA, USA: USENIX Association, 2011: 45-78.

[6] GAUTHIER F, MERLO E. Fast Detection of Access Control Vulnerabilities in PHP Applications[C]// 19th Working Conference on Reverse Engineering. Kingston: IEEE, 2012: 281-290.

[7] PAN K, WANG Q. Static Detection of Access Control Vulnerabilities in Vue Applications[J]. Journal of Physics: Conference Series, 2020, 1646(1): 12-21.

[8] MONSHIZADEH M, NALDURG P, VENKATAKRISHNAN V N. MACE: Detecting Privilege Escalation Vulnerabilities in Web Applications[J]. Bone, 2014, 47(Suppl 1): 690-701.

[9] LE H T, NGUYEN C D, BRIAND L, et al. Automated Inference of Access Control Policies for Web Applications [J]. ACM Transactions on Software Engineering and Methodology, 2015, 24(3): 27-37.

[10] LI X, YUAN X. LogicScope: Automatic Discovery of Logic Vulnerabilities within Web Applications[C]// Acm Sigsac Symposium on Information. Hangzhou, China, 2013, 2013(5): 481-486.

[11] LI X, SI X, YUAN X. Automated Black-box Detection of Access Control Vulnerabilities in Web Applications[C]// ACM Conference on Data & Application Security & Privacy. ACM, San Antonio, Texas, USA, 2014, 2014(3): 49-60.

[12] DEEPA G, THILAGAM P S, PRASEED A, et al. DetLogic: A Black-box Approach for Detecting Logic Vulnerabilities in Web applications[J]. Journal of Network & Computer Applications, 2018, 109(5): 89-109.

[13] LI X, YUAN X. BLOCK: A Black-box Approach for Detection of State Violation Attacks Towards Web Applications[C]// Computer Security Applications Conference. ACM, Orlando, Florida, USA, 2011: 247-256.

[14] REN J, WU M, ZHANG B, et al. DetAC: Approach to Detect Access Control Vulnerability in Web application Based on Sitemap Model with Global Information Representation[J]. International Journal of Software Engineering and Knowledge Engineering, 2023, 33(9): 1327-1354.

[15] KUSHNIR M, FAVRE O, RENNHARD M, et al. Automated black box detection of HTTP GET request-based access control vulnerabilities in web applications[C]// ICISSP 2021. SciTePress, 2021: 204-216.

[16] LE H T, SHAR L K, BIANCULLI D, et al. Automated reverse engineering of role-based access control policies of web applications[J]. Journal of Systems and Software, 2022, 184: 111109.

[17] LIU X, JIANG W, ZHANG Y. A survey of access control models[J]. IEEE Communications Surveys & Tutorials, 2016, 18(1): 829-856.

[18] ZHANG Y, XIE T, LIU Y, et al. A Survey on Role-based Access Control Models[J]. Journal of Computer Science and Technology, 2021, 36(3): 439-466.

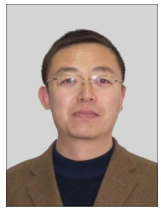
[19] LIU J X, MA S M, QI H L. Research and implementation of access-rights control in web systems[J]. Computer Engineering and Design, 2008, 10: 2550-2553.

[20] YANG J, SHEN X, CHEN W, et al. A Model Study on Collaborative Learning and Exploration of RBAC Roles[J]. Wireless Communications and Mobile Computing, 2021, 2021(5): 1-9.

[21] LI X, SI X, YUAN X. Automated Black-box Detection of Access Control Vulnerabilities in Web Applications[C]// ACM Conference on Data & Application Security & Privacy. ACM, San Antonio, Texas, USA, 2014, 2014(3): 49-60.

[22] MONSHIZADEH M, NALDURG P, VENKATAKRISHNAN V N. MACE: Detecting Privilege Escalation Vulnerabilities in Web Applications[J]. Bone, 2014, 47(Suppl 1): 690-701.

[23] XIA Z J, PENG G J, HU H F. Detection of ac cess control vulnerabilities in Web applications based on privilege verification graph [J]. Computer Engineering and Applications, 2018, 54(12): 63-68.



REN Jiadong, born in 1967, Ph.D, professor, Ph. D supervisor, is a senier member of CCF(No. 13382S). His main research interests include data mining and software security.

ZHANG Bing, born in 1989, Ph.D, associate professor, Ph. D supervisor, is a member of CCF (No. H3272M). His main research interests include data mining and software security.

（责任编辑：何杨）