doi:10.3772/j.issn.1002-0470.2025.03.001

Web 应用访问控制漏洞检测研究进展①

孟海宁② 陆 杰 李昊峰 黄永恒 曹立庆 李 炼③

(中国科学院计算技术研究所处理器芯片全国重点实验室 北京 100190)

(中国科学院大学 北京 100049)

摘 要 访问控制机制在维护 Web 应用安全方面扮演着至关重要的角色。然而,不安全的访问控制机制已经成为威胁 Web 应用安全的主要风险来源之一。为了保护 Web 应用安全,及早地检测并报告访问控制漏洞至关重要。为了帮助安全研究者和程序开发者更深入地了解 Web 应用的访问控制漏洞以及现有工作的研究进展,本文对该领域的相关工作进行了系统总结,概括了访问控制漏洞的基本概念与特征,归纳了检测访问控制漏洞的关键步骤和核心思想。此外,本文进一步针对功能级、对象级和对象属性级 3 类高危访问控制漏洞,详细分析和讨论了现有工作的漏洞检测能力。最后,对 Web 应用访问控制漏洞检测技术的发展进行了总结与展望。

关键词 Web 应用;访问控制;访问控制漏洞;漏洞检测;越权访问;软件缺陷检测

Web 应用存储了海量用户数据。然而,在保护敏感数据方面,现有应用面临着内部管理不善和外部黑客攻击的双重威胁[1]。因此,Web 应用引入了访问控制机制,以防止未授权的访问和数据泄露[2]。但是随着软件系统复杂度的日趋增加和开发人员的疏忽,现实中访问控制漏洞通常难以根除。例如,利用隐藏链接保护特权页面的措施无法抵御对服务器的越权攻击[3]。同时,开放式 Web 应用程序安全项目(open worldwide application security project, OWASP)的报告显示[4],访问控制漏洞一直位居 Web 应用安全风险榜单的前列,且近年来攀升至首位。一旦这些漏洞被恶意利用,可能导致海量敏感数据被篡改或攻击者获取站点管理权限,引发严重的安全问题。

漏洞检测是及时发现潜在漏洞的有效手段。检测 Web 应用访问控制漏洞最大的挑战在于该漏洞本质上属于典型的逻辑漏洞[5-7],表现为应用中访问控制机制的实际执行不符合预期授权策略。判断是否具有访问控制漏洞需要因具体应用而异,所以该类漏洞难以被现有的其他类型漏洞的检测模式识别出来[8-10]。同时,预期授权策略是访问控制漏洞检测的基础。它可以由软件开发者显式指定,而实际中更常见的情况是需要从应用中推断出来,但策略推断本身也充满了挑战[6]。目前已有许多工作针对上述问题提出了解决方案。

本文对近年来检测访问控制漏洞的技术进行了分析和分类,主要贡献如下。

(1) 梳理并分析了国内外在 2007 - 2023 年间针对访问控制漏洞检测所开展的研究工作,为该领域提供了一个全面的研究综述。

(2) 抽象并确定了访问控制漏洞检测的 2 项核心问题,即预期授权策略的获取和漏洞检测的执行,并详尽探讨了现有方法的关键技术。

(3) 针对功能级、对象级和对象属性级这 3 种高风险的访问控制漏洞,深入分析和讨论了目前技术的检测能力。

① 国家自然科学基金(62132020)资助项目。

② 女,1995 年生,博士生;研究方向:程序分析;E-mail:menghaining@ict.ac.cn。

③ 通信作者,E-mail:lianli@ict.ac.cn。

(收稿日期:2024-01-29)

(4) 总结了访问控制漏洞检测的研究现状,并给出了 4 个具有发展潜力的研究方向。

1 背 景

1.1 访问控制漏洞

访问控制是允许或限制主体对客体访问能力及范围的一种安全机制[11-12]。访问控制的核心是授权策略[13],该策略定义了用户对特定功能或资源的访问能力,是运行时的权限检查依据。良好的访问控制机制应当满足设计者的安全意图[14-16],否则,被视为存在访问控制漏洞。利用访问控制漏洞实现对安全敏感操作(security sensitive operation)越权访问,导致权限提升。因此,访问控制漏洞被视为最危险的 Web 应用漏洞之一[17]。

检测 Web 应用访问控制漏洞的本质是比较实际访问控制机制与预期授权策略之间的差异[6],因此,本文将检测该类漏洞抽象成以下 2 个关键问题。

(1)预期授权策略的获取。预期授权策略是用于检测漏洞的基准,其关键在于如何获取每个应用程序独特的预期授权策略。

(2)漏洞检测的执行。该问题的本质是一致性检查,涉及识别出程序实际的访问控制机制,进而判断实际授权机制是否违反预期授权策略。

1.2 三类访问控制漏洞

Web 应用访问控制漏洞涵盖内容繁多[3],本文根据最新榜单[18],选择威胁 Web 应用安全排名前 3 的访问控制漏洞子类型作为研究关注点,即功能级、对象级和对象属性级访问控制漏洞。图 1 展示了上述 3 个类型的漏洞实例的抽象代码片段。其中,每个代码片段代表一个外部可访问的功能,其中带“+”的代码行表示漏洞修复补丁。本小节围绕该例子介绍每个访问控制漏洞子类的特征、漏洞实例触发场景和修复措施。

1.2.1 功能级访问控制漏洞

功能级访问控制漏洞(broken function level authorization, BFLA)是 Web 应用程序未充分检查用户对某些功能的访问权限而引发的安全风险[19]。该类漏洞的本质是服务端的处理函数对用户访问的限制不足,导致攻击者能够越权访问本应受保护的特殊功能及其关联的敏感资源。BFLA 漏洞在最新的 OWASP 应用程序接口(application programming interface, API)安全风险榜单中排名第 5 位[18]。

图 1(a)所示功能的预期授权策略是只允许管理员更新成员信息。但由于这段代码在实现上缺少对操作者的权限检查,攻击者在获悉特定地址后,能以普通用户的身份强制访问该功能,完成越权操作。为了修复该漏洞,开发人员在第 2 行添加注解@RequiresPermissions 限制操作者权限。

1.2.2 对象级访问控制漏洞

对象级访问控制漏洞(broken object level authorization, BOLA),也被称作不安全的直接对象引用(insecure direct object reference, IDOR),指当使用外部可操纵的值去索引资源对象而产生的越权访问漏洞。在现代基于 Web API 的应用中,服务端访问的对象通常依赖于外部请求参数[20],BOLA 漏洞的本质是未充分验证操作者是否具有操作其指定数据对象的权限,导致攻击者可以通过操纵请求参数,直接访问、操作或篡改任意指定的对象数据。BOLA 漏洞居于 OWASP API 安全风险榜单首位[18]。

图 1(b)的代码用于删除用户指定的数据对象。预期授权策略是用户只能删除属于自己的数据。而在实现中,代码并没有验证操作者是否具有删除其指定对象的权限。因此,攻击者可以通过修改请求参数的方式,任意指定“id”(identity)值,实现删除本属于其他任意用户的数据条目。为了修复该漏洞,开发人员在第 6 行添加检查操作,以确保用户操作数据对象的权限合法性。

1.2.3 对象属性级访问控制漏洞

现代软件框架允许将网络请求或响应中的参数和代码中的对象进行自动关联。然而,并非同一个对象的所有属性都具有相同的访问权限。对象属性级访问漏洞(broken object property level authorization, BOPLA)源自于未正确检查对象属性的权限而产生越权行为[21]。BOPLA 漏洞涉及 2 个方面:一是过多的数据暴露(excessive data exposure),例如当响应中暴露了过多的数据时,攻击者通过嗅探网络流量轻松获取敏感信息;二是批量赋值(mass as-

signment),例如攻击者通过请求参数修改了页面中的只读属性值。这类漏洞在 OWASP API 安全风险榜单中排名第 3 位[18]。

图 1(c)的功能是更新用户节点信息,其中节点的名字属性在用户界面上是只读的。然而,该约束并没有在功能代码上正确体现。攻击者利用批量赋值漏洞,通过修改请求中的参数,成功在服务端修改节点对象的名字属性。为了修复该漏洞,开发人员在第 10 行添加属性检查语句,以确保只读属性不会被外部修改。

```
1 @PostMapping("/member/update")
2 + @RequiresPermissions("PROJECT_USER:READ+EDIT")
3 public void updateMember(@RequestBody WorkspaceMemberDTO memberDTO) {
4     projectService.updateMember(memberDTO);}
```

(a) 功能级访问控制漏洞实例: CVE-2023-35937

```
1 @DeleteMapping("/consume/delete/{id}")
2 public Response<Boolean> delete(@PathVariable(name = "id") Integer id) {
3     return Response.success(consumeService.delete(id, LoginUserUtils.getLoginUser().getName()));}
4 public Boolean delete(Integer id, String currentUser) {
5     InlongConsumeEntity entity = consumeMapper.selectById(id);
6+    userService.checkUser(entity.getInCharges(), currentUser, "...");
7     deleteInDB(entity);
8     return true ;}
```

(b) 对象级访问控制漏洞实例: CVE-2023-31453

```
1  @PostMapping(value = "/node/update")
2  public Response<Boolean> update(@RequestBody DataNodeRequest request) {
3     return Response.success(dataNodeService.update(request, LoginUserUtils.getLoginUser().getName()));}
4  public Boolean update(DataNodeRequest request, String operator) {
5     DataNodeEntity curEntity = dataNodeMapper.selectById(request.getId());
6     userService.checkUser(curEntity.getInCharges(), operator, "... ");
7     // check whether modify unmodifiable parameters
8     Preconditions.expectEquals(curEntity.getType(), request.getType(), ...); //check type
9     Preconditions.expectEquals(curEntity.getVersion(), request.getVersion(), ...); //check version
10+   Preconditions.expectEquals(curEntity.getName(), request.getName(), ...); //check name
11    updateInDB(curEntity, request, operator);
12    return true;}
```

(c) 对象属性级访问控制漏洞实例: CVE-2023-31206

图 1　三种访问控制漏洞实例

1.3　文献获取与相关工作

本文选取 2014 – 2023 年为搜索时间,在国内外主流数据库中使用 Web 应用访问控制漏洞相关的关键词进行组合搜索。经人工筛选与引文补充,得到相关文献共计 43 篇作为本文的研究对象,涵盖软件工程、安全等领域的顶级期刊和会议,如 ASE(Automated Software Engineering)、ICSE(International Conference on Software Engineering)、OSDI(Operating Systems Design and Implementation)、OOPSLA(Object-Oriented Programming Systems, Languages, and Applications)、CCS(Computer and Communications Security)和 USENIX Security 等。

值得注意的是,在这些研究中包含了 7 篇与访问控制漏洞检测相关的文献综述。表 1 显示了这些综述工作的侧重点。可见,现有文献综述对访问控制漏洞检测的分类单一且讨论粗略,缺乏对方法检测能力进行更细粒度的讨论。因此,本文重点关注 3 个排名前列的访问控制漏洞类型,并将漏洞检测抽象为 2 个重要模块,对现有检测技术和能力进行归纳和总结。

2　预期授权策略的获取

2.1　获取方式

2.1.1　人工提供

获取目标应用预期授权策略最直接的方法是要求应用开发者提供[26-28]或人工理解每个应用的实现逻辑后从中提取预期授权策略[10,16]。然而在实践中,被测目标往往难以提供上述信息,因此研究人员尝试从开发阶段使用的其他文档记录中寻找预期授权策略。例如,文献[29-31]从应用的 OpenAPI

表 1　访问控制漏洞相关综述

文献序号	年份	侧重点
[5]	2023	关注访问控制漏洞;围绕软件编写、测试和运行 3 个阶段,综述现有的漏洞预防或检测工作;检测分为异常检测、自动测试和形式化方法 3 类
[22]	2022	包括但非针对访问控制漏洞;文献映射研究
[23]	2022	包含但非针对访问控制漏洞;综述漏洞检测方法的检测效率和效果
[24]	2020	关注访问控制漏洞的静态检测,检测方法分为基于图、基于上下文和基于模型 3 类
[7]	2016	关注逻辑漏洞(包括访问控制漏洞);围绕软件生命周期不同阶段综述访问控制漏洞的预防和检测;检测技术分为白盒与黑盒 2 类
[25]	2015	包括但非针对访问控制漏洞;文献映射研究
[6]	2014	关注逻辑漏洞(包括访问控制漏洞);围绕软件编写、测试和运行每个阶段,综述检测或缓解漏洞的方法;检测技术分为静态和动态 2 类

文件或描述业务工作流程的活动图中获取预期授权策略。此外,也有研究人员基于“开发人员通常希望将特定模式统一应用于给定资源类型的所有用途”的假设,探索从现实 Web 应用的开发经验中分析和提炼出常见的访问控制模式,将此作为通用的访问控制预期授权策略[32]。

人工获取预期授权策略具有直接且精确的优势,但检测目标数量庞大,人工为每个目标定制化是不现实的。此外,被测应用的开发文档并非能轻易获得,因此从文档中获取预期的方法无法广泛应用。尽管将通用访问控制机制作为预期授权策略的方法可以降低一定的人力成本,但是该方法受限于人工经验和所有应用共享相似安全机制的假设。

2.1.2　静态推断

通常在 Web 应用的代码中已经存在部分实现是满足设计者的预期授权策略的,即程序中至少有一条路径上的权限检查符合预期。因此,可以利用静态代码分析技术推断出预期授权策略,其流程抽象如图 2 所示。该类方法的关键在于找到到达安全敏感操作的预期权限约束。

一些研究工作将访问特权页面视为安全敏感操作,静态推断不同页面对应的保护预期类型。现有方法需要人工在程序中标记特殊变量或角色含义[9,15],这些特殊变量和角色通过分支判断的方式保护特权页面。在推断预期保护的页面时,Sun 等人[9]结合程序控制流图、显式链接和文件包含关系,计算出每个角色的特权页面。Gauthier 和 Merlo[15]则利用启发式规则和模型检验(modelchecking)计

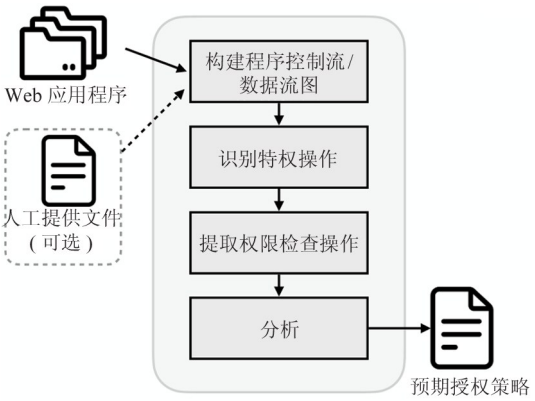

图 2　现有研究中静态推断预期授权策略流程

算预期受保护的页面集合。

文献[33-36]关注代码层面的安全敏感操作语句(如数据库执行语句),认为访问控制约束体现在程序的条件检查语句中,因此可以从条件检查语句和敏感操作语句的关系中推断出预期授权策略。推断流程总结为先收集程序中所有敏感操作语句,然后通过数据流、控制流等静态关系分析,最后找到约束敏感操作语句的条件检查语句,从中提取权限检查操作。然而,由于访问控制高度依赖应用具体逻辑,上述工作仍然需要人工标注程序中权限相关的变量。其中,集成在开发环境中的工具 ASIDE[33-34](Application Security plug-in for the Integrated Development Environment for PHP,PHP 集成开发环境的应用程序安全插件)在程序开发时期若检测到程序的敏感操作语句,则交互式地要求开发者标注对应的权限检查语句。此外,Son 等人[37]实现的 RoleCast 基于 Web 应用程序的文件组织结构与用户角色存

在联系这一发现,探索预期授权策略的自动化推断方法。该方法首先对程序文件按照组织结构进行分类,每类对应不同的角色。然后在同一角色内,选取条件语句中出现频率高于阈值的变量作为关键变量,对这些变量的条件检查语句被认为是权限检查操作。

静态推断预期授权策略依赖于程序中已实现的逻辑片段。然而,如果应用的实现逻辑存在错误,那么推断出的授权策略的可信度将大幅降低。另外,虽然许多现有工作[9,15,33-36]仍需要人工参与,但与需要理解整个应用逻辑相比,这些工作仅涉及对代码片段的理解和标注,降低了人工参与的门槛和工作量。其中,RoleCast 是自动推断具体意图的新尝试,虽然实验显示它面对现代大规模应用时受限于时间成本,不适用于时间敏感的检测需求,但其思路仍值得借鉴。

2.1.3 动态推断

Web 应用界面展示的功能通常经过了充分测试,因而可视为用户被允许的预期行为[38]。据此,研究人员尝试从这些正常交互中推断出预期的授权策略。动态推断流程如图 3 所示。

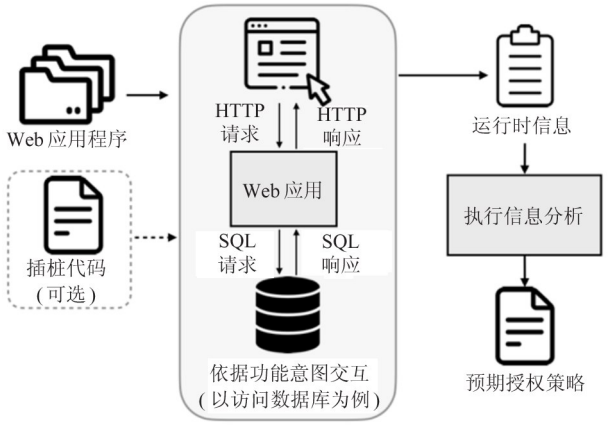

图 3 现有研究中动态推断预期授权策略流程

为了获取运行时信息,文献[38-40]对程序源码代码插桩,捕获运行时特定程序点的关键变量值(如存储在会话中的变量)。这些变量面对不同请求展现出的固定值或与请求参数的不变关系,被用于刻画程序在运行时的预期行为。

当无法获取检测目标程序源码时,研究者探索利用黑盒测试来获取程序执行的状态信息。文献[41-48]通过人工或爬虫的方式与目标应用进行正常交互获取特定资源被允许访问的预期。还有一些工作在正常交互时拦截与 Web 服务器交互的网络请求和响应[8,14,49-53],以及应用和数据库交互的请求和响应[8,14,51-52],从中分析正常交互下的预期执行逻辑。其中,Li 等人[49]是最早使用黑盒测试来推断预期授权策略的工作。他们基于“会话变量维护无状态请求之间用户请求的状态”这一特征,将应用程序建模为无状态系统,再根据交互请求和响应涉及的会话变量值,分析出程序预期执行状态下的不变关系。文献[8,50,52,53]利用正常访问时拦截的会话变量、参数等信息构建表示预期授权策略的有限状态机(finite state machine,FSM)。文献[14,51]从角色和角色内部不同用户进行细粒度划分授权策略,通过分析数据库响应视图和 Web 响应视图在属性值上的关联性,以及对请求内和请求间的数据依赖分析,分别推断出预期的角色级策略和预期的用户级策略,二者共同表示了应用程序的预期授权策略。

动态交互能准确捕捉程序的运行时信息,理论上可以从中推断出理想的预期授权策略。但由于现代应用逻辑和使用场景的复杂性,成功部署 Web 应用本身并非易事。同时,预期授权策略的完整度依赖于动态获取信息的丰富程度,而其又取决于交互的路径覆盖程度。

2.2 表示方式

2.2.1 安全约束

安全约束描述了在执行安全敏感操作时,操作者需要满足的权限条件。其中,权限条件可以基于角色、属性、程序结构等多种因素定义。文献[10,14,15,27,29,32,51]以用户或角色为粒度,将预期授权策略表示为安全敏感操作(如数据库访问操作)与被允许的角色或用户属性(如管理员角色或用户所属变量)的对应关系。关注程序分支结构的一些工作[17,28,33,36,54]将预期授权策略表示为安全敏感操作与其依赖的条件检查语句。在结合上述 2 种表示工作[9,16,35,37,40]的预期授权策略中,安全敏感操作的约束从被允许的角色以及相同角色内部的条件检查语句 2 个维度表示。

2.2.2 不变量

不变量是指在程序某个片段在所有执行过程中需要保持为真的逻辑断言[55]。在 Web 应用场景中,如果遵循应用预期导航的执行顺序,每当执行到程序点 p 处,变量 $ *SESSION*['role'] 的值总为'admin',那么,$ *SESSION*['role'] = 'admin'能被用来刻画执行到程序点 p 时需要满足的必要条件。许多工作在特定程序点抽象出不变量来表示应用在运行时需要满足的预期授权策略。文献[38,39]将运行时特定程序点变量之间的不变关系作为预期授权策略。而在将 Web 应用视作黑盒的工作中,不变量则与特定的请求或页面关联,它可以表示为过滤掉时间戳等无关内容后的交互的响应[41-47,56],或是会话变量、网络或数据库的请求或响应(包括参数)等之间存在的恒等或先后续等关系[8,49,53]。

2.2.3 程序流程

程序流程是指程序预期的执行流程,包括了访问控制的预期执行流程。Ghorbanzadeh 和 Shahriari[31]利用活动图与控制流图构建活动流图(activity flow graph,AFG),其中虚线边表示包括预期的访问控制流程在内的预期执行流程。Li 和 Xue[50]以及 Wen 等人[52]将预期的执行逻辑建模成预期有限状态机(intended FSM),用其表达程序在特定状态下接受给定输入发生的预期状态迁移以及预期输出,其中包括了对预期的访问控制机制的描述。

2.2.4 小 结

预期授权策略的表示是贯穿整个漏洞检测过程的重要数据结构,与具体的检测方法息息相关。安全约束直观地界定了敏感操作点以及其所需的权限条件,然而其表达能力并非自身所决定,而是取决于敏感操作点的表示细节。不变量和程序流程都具体地刻画了程序整体运行时需要满足的权限条件,但其现有的获取途径或基于已知文档描述,或基于动态推断,受限于提供文档的完整度、程序覆盖率与人力成本。

3 Web 访问控制漏洞检测方法

本节将围绕第 1.2 节介绍的 3 类访问控制漏洞,介绍现有研究工作对它们的检测能力。然而大多数检测方法并非针对这 3 类漏洞而设计,故本文将检测方法分为统一方案和针对性方案 2 类。图 4 是对现有工作的分类图。本节将详细阐述每类方法的漏洞检测技术类型、核心思想以及能够检测的漏洞种类。

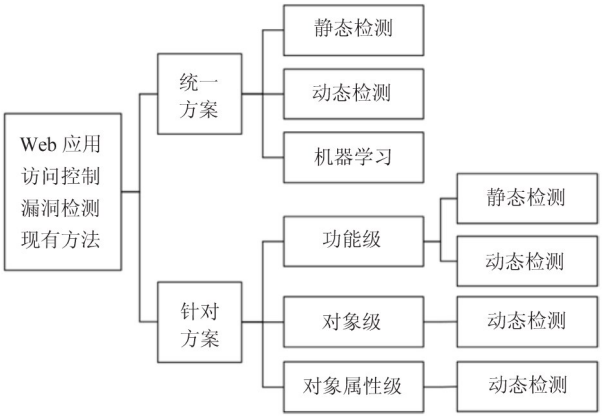

图 4 Web 应用访问控制漏洞现有检测方法分类

3.1 统一方案

统一方案未对 3 类访问控制类型的特征进行特殊处理,其能够检测的访问控制漏洞类型由方案本身决定。

3.1.1 静态分析

静态分析是指对程序代码进行自动化的扫描、分析,而不必运行程序[57]。直接静态地比较实际访问控制执行和预期授权策略之间的差异,是最直观的检测方法。对于关注条件语句保护敏感操作点的授权策略而言,文献[16,33,35-37,54]从代码分支结构中提取出实际权限约束,通过检查到达同一敏感操作的不同路径上施加的权限检查的一致性,交叉验证(cross check)出潜在的访问控制漏洞。其中,Son 等人的 FixMeUp[16]和 RoleCast[37]将一致性检查限定在具有相同角色权限范围下展开。对于关注敏感操作点用户角色范围的授权策略而言,则需要计算实际敏感操作点的操作权限。Ghorbanzadeh 和 Reza[10]提出了结合人工标记对程序进行数据流和控制流分析方法,计算读写特定资源操作的访问控制权限,然后比较对资源的实际操作权限与人工标记的该资源所需权限的差异即可检测漏洞。此外,他们还利用活动图、控制流图和人工注解构建 AFG 图(设计和实际实现分别用虚线和实线表示),

然后通过遍历 AFG 图比较二者之间的差异,以检测由设计和实现层面的预期工作流程差异引起的漏洞,包括访问控制实现漏洞[31]。Luong 等人[27]通过资源之间的关系构建实际访问控制矩阵,与预期授权策略比较以检测差异。

还有一些工作通过静态验证安全敏感操作点的预期授权策略的可满足性来判断是否存在漏洞。Felmetsger 等人[38]提出的 Waler 利用模型验证技术来识别预期授权策略的违反。Bocic 等人[40]提出的 CanCheck 将预期授权策略和程序中的实现逻辑转换为逻辑表达式,利用定理证明器验证其可满足性。Near 和 Jackson[32]提出的 SPACE 使用约束求解器验证应用实现逻辑是否符合白名单中表达的常见访问控制模式。而 Chlipala[26]提出的 UrFlow 则利用符号求值和自动定理证明的方法在数据库更新点和响应点进行静态检测,以验证人工提供的类结构化查询语言(structured query language, SQL)预期授权策略是否得到满足。

上述方法均判断了特定功能的执行权限是否符合预期,因此都能够用来检测功能级访问控制漏洞。部分工作[10,26,31-32,35,38,40]还具备检测对象级访问控制漏洞的能力。Waler 在获取预期授权策略时收集的不变量中包含了针对对象访问的约束,实验结果也显示它能够发现对象级访问控制漏洞。SPACE 的预定义访问控制模式目录中包含了对资源所有权的约束;一些工作或是依赖人工标记权限检查代码片段[10,31],或是在求解可满足性时使用的预期授权策略中包括了访问资源时资源所属对象的约束[26,40],这些方法对对象级访问控制漏洞的检测依赖于开发者标记、预定义白名单范围的准确性或运行时的路径覆盖。与这些方法不同,MACE 在计算调用数据库操作语句的授权上下文时,还额外利用数据依赖关系计算了 SQL 语句中表达的资源所属关系,这使得 MACE[35]能够检测出对象级访问控制漏洞。然而计算该约束依赖于人工为每个检测目标应用标注全局变量,同时在计算授权上下文时忽略选择语句也会导致推断时使用的语义不完备。

3.1.2 动态分析

异常检测能够在运行时识别出状态违规情况,辨别攻击请求。Cova 等人[39]监视程序的运行信息,将其与预期状态的模型进行对比以报告异常情况。另一些工作[8,49]将预期授权策略转化为评价函数,然后手动构造攻击,判断运行状态是否满足预期评价函数,若不满足则判定其存在漏洞。

自动化动态检测访问控制漏洞可以从测试用例生成与执行、评估执行结果 2 个方面考虑。在自动测试用例生成与执行方面,现有工作的主要思想是自动地构建违反预期授权策略的请求序列并发送至测试目标。Wen 等人[52]提出的 Lom 关注非关系型数据库操作,根据用户的正常执行构建了预期状态机,结合预期状态机的状态和强制访问的攻击原理进行输入变异。文献[14,43-45,51,53]根据预期授权策略变异正常请求中的参数,以生成新的测试请求序列。例如,用低权限参数替换请求的高权限参数,或指定不属于用户的资源参数。自动化结果评估阶段的主要目的是自动判断攻击结果与预期结果的差异,可以将实际执行映射到自动机的输出中判断是否满足预期[52],或者通过分析实际执行与原请求的响应差异进行漏洞检测[14,43-45,51,53]。

在访问控制漏洞检测能力方面,上述方法都能够检测功能级访问控制漏洞。其中,一些方法还可以检测出对象级访问控制漏洞。例如,通过从不同用户对同一目标的访问执行中差分出请求中的特殊参数,而这些参数直接指向资源对象[43-45]。文献[8,14,51,53]在定义预期授权策略时考虑了操作者与被参数索引的数据库对象之间的关系,并在漏洞检测时基于这些关系进行违背检测。然而,上述方法的漏洞检测能力同样依赖于正常交互阶段获取到的执行语义。

3.1.3 机器学习

Khalid 等人[58]尝试利用包含多个决策树的随机森林作为 Web 应用漏洞分类器。他们使用 73/233个与授权相关的用例作为训练集,利用随机森林算法训练出一个能够区分包括访问控制漏洞在内的多种常见 Web 应用漏洞的分类器。该分类器用于应用的漏洞检测。这类方法能够检测的访问控制漏洞类型取决于训练数据集的内容。

3.2 针对性方案

随着研究的深入和系统安全性要求的提高,研

究者对访问控制漏洞的关注也更有针对性。本节将围绕 3 个特定的访问控制漏洞类型,介绍针对性解决各类漏洞的现有研究方法。

3.2.1 检测功能级访问控制漏洞

功能级访问控制漏洞的检测旨在验证是否存在对某些特权操作背后的处理函数或特定页面的访问权限。Kong 等人[17]从响应用户请求的功能函数出发,通过后向程序切片与启发式方法相结合,获取调用每个功能函数的实际控制流,然后对比预期授权涉及的控制流,找到权限检查不足的路径,即存在功能级访问控制漏洞的路径。Gauthier 和 Merlo[15]根据人工标记的权限将功能页面进行分类,计算实际被保护页面与预期被保护页面的差异从而找出未被安全保护的功能页面。Piskachev 等人[28]提出的 AuthCheck 采用黑名单策略,预定义了描述访问控制漏洞的状态机,状态包括是否授权和出现漏洞。该方法从处理用户请求的入口函数开始,遍历所有访问敏感资源或敏感操作的路径,判断这些路径是否会转移到漏洞状态。

此外,模拟强制访问(force browsing)也是检测应用中是否存在功能级访问控制漏洞的一种有效手段。Sun 等人[9]使用的预期授权策略描述了每个角色与其能访问的页面,他们通过静态模拟强制访问攻击,针对特定页面分别计算有权限和没有权限角色直接访问所产生的结果,比较二者的差异以检测功能级访问控制漏洞。Li 和 Xue[50]提出的 LogicScope 根据正常执行构建预期有限状态机,其中状态代表不同权限的会话变量值,输入是请求抽象,输出是可访问的页面。在预期状态机的不同状态下,对页面访问所使用的输入进行变异,随后携带变异值攻击目标应用,对比动态执行变异前后的响应以检测漏洞。

上述检测方法专注于功能级访问控制漏洞,将前端的页面或后端处理不同功能请求的函数作为保护对象,以检测越权行为。然而,这些方法只关注是否具有访问特权操作或特权页面的权限,并未考虑是否具有操作由外部指向的、特权操作背后的资源对象或特权页面的具体元素的权限。因此,上述方法均不适用于更细粒度的对象级访问控制漏洞和对象属性级访问控制漏洞的检测。

3.2.2 检测对象级访问控制漏洞

对象级访问控制漏洞的检测关键点包括:(1)识别潜在的直接引用对象的发生点;(2)评估是否存在对象级访问控制漏洞[59]。针对这类漏洞的现有检测方法均采用动态检测技术。

常见的检测对象级访问控制漏洞的现有方法基于人工渗透测试技术[41-42,47]。具体为根据不同角色拥有的不同私有对象,变异不同角色的请求所携带的参数。手动测试涉及正常交互、识别可疑请求和参数、判断漏洞发生 3 个步骤。即使存在渗透测试工具如 Burp Suite 和 OWASP ZAP,手动测试仍然面临效率低和依赖人工经验的问题。

Hadavi 等人[46]提出了一种半自动化的检测对象级访问控制漏洞方法。该方法只需要在交互阶段进行半自动化操作,其余阶段均为全自动。其实现的工具——不安全的直接对象引用检测工具(Insecure Direct Object reference detection Tool,IDOT)对请求地址的相似性分析并提取可疑请求,然后根据不同身份用户的请求识别并更改可疑的参数,最后通过分析响应内容与受影响的页面来判断是否触发漏洞。

上述方法是基于对象级访问控制漏洞的检测流程设计的,因此对于该类漏洞的检测效果显著。然而,该检测流程的关注点主要在于外部输入和其索引对象之间的关系,并且只关心是否能触发漏洞,没有进一步探索与被索引对象的属性之间的关系,因而无法检测出另外 2 类访问控制漏洞。

3.2.3 检测对象属性级访问控制漏洞

对象属性级访问控制漏洞的检测关注 2 类漏洞:过度数据泄露和批量赋值,且现有检测方法均采用动态检测技术。

Pan 等人[48]提出的过度数据暴露的模糊测试(excessive data exposures fuzz,EDEFuzz),是针对过度数据泄露漏洞自动检测的首个工作。该方法通过聚类操作和启发式规则,首先识别出 OpenAPI 文件中描述的只读属性;然后,构建尝试覆盖只读属性的请求序列进行动态测试,识别出潜在的对象属性级访问控制漏洞。

Corradini 等人[29]是第一个公开的自动化检测批量赋值工作的研究团队。该方法基于“记录-重放(record-replay)”模式,对正常运行过程中记录的原始请求参数进行突变,最后比较突变前后的响应内容以判断应用中是否存在批量赋值漏洞。

上述 2 种方法分别针对对象属性级访问控制漏洞中的 2 类典型问题,即过度数据暴露和批量赋值漏洞,展开研究并探索出动态的检测方案。然而,由于缺乏对功能触发权限和对象引用权限的验证机制,这些方法无法有效检测与之相关的其他漏洞类型。同时,这些近期研究成果表明,更细粒度的访问控制漏洞逐渐引起研究者的关注。

3.3 检测方法总结

早期的工作倾向于使用一种笼统的方式对访问控制漏洞进行检测,技术涉及传统的动态、静态分析以及机器学习方法。然而,统一方案的实际检测能力取决于研究者对漏洞的理解。以数据库操作为例,它是应用功能背后的实际操作敏感资源的常见方式。能否执行数据库操作则代表是否通过功能级访问控制检查。此时,若进一步考虑敏感操作使用的外部指向对象与操作者的权限关系,则能够检测对象级访问控制漏洞。同时,若在直接进行如非公开资源的重写或公开操作前考虑资源对象中属性的权限问题,则能够检测更细粒度的对象属性级访问控制漏洞。

随着对访问控制漏洞关注的细粒度化,针对性地检测访问控制子类漏洞成为关注点。然而,目前除了功能级访问控制漏洞检测包括静态方法外,其他漏洞类型的检测主要采用动态检测技术。功能级访问控制漏洞直观地与代码中处理函数的授权逻辑相关,因此成为大部分研究工作的关注点。而对象级和对象属性级访问控制漏洞约束粒度更细,交织在业务逻辑代码中更不易静态察觉,因此通常需要通过监视程序运行时行为,包括追踪变量的访问和修改情况或对象之间的交互等方式,以检查操作是否符合预期的授权策略。同时,从研究的发生时间与关注度来看,对细粒度访问控制漏洞的检测也成为趋势。

4 趋势展望

本文深入分析现有研究成果,并据此展望访问控制漏洞检测领域的潜在研究趋势。

(1) Web 应用访问控制漏洞的根本原因分析。实证研究的结果对更精细的访问控制漏洞检测的研究与探索具有重要意义。实证研究的根本原因分析结果可以极大地提高漏洞检测效率。同时,大量开源项目中的暴露漏洞为实证研究提供了充足的数据来源。在访问控制漏洞的根本原因研究中,首先大规模地收集不同类型应用已报告的漏洞,然后深入分析案例,识别访问控制漏洞的特征、发生场景及其成因。最后,总结出漏洞产生的常见模式和规律。研究结果可以指导未来在设计漏洞检测方案时更加精准地定位和解决问题,有助于未来更加有针对性地改进检测算法。

(2) 基于日志的预期授权策略获取。现有的研究通过源码分析或 SQL 查询分析的方式提取不变量[8],并使用这些不变量来表达权限授予的预期策略。然而这些方法不适用于分析闭源系统。同时,基于日志分析的不变量推导被广泛地应用于系统故障检测方面[60]。但并非所有 Web 应用都具备日志记录功能。取而代之的是,大多数现有的 Web 系统都依赖数据库来存储信息。鉴于此,本文提出一个新的研究方向:利用数据库日志来推导不变量,并以此来描述 Web 应用预期授权策略。不过,这一方法面临着一个重要挑战——数据库日志并不直接反映 Web 系统的用户操作行为。在数据库日志与 Web 系统的操作模式之间缺乏直接对应关系,这给不变量和 Web 预期授权策略的映射带来难题。

(3) 细粒度访问控制漏洞的静态检测。针对功能级访问控制漏洞,已有多种静态分析技术研究成果。然而,对于更细粒度的访问控制漏洞类型,如对象级和对象属性级访问控制漏洞的检测,大多依赖动态分析。特别是在对象属性级访问控制漏洞检测方面,静态分析技术的研究工作较少,存在显著的研究空白。动态方法虽能获取真实执行情况,但存在运行时开销,并可能由于测试用例覆盖不全而遗漏潜在漏洞。另一方面,静态分析存在高误报率,可通

过机器学习等技术进行误报消除。

(4) 探索现代 Web 应用中的漏洞检测。安全框架在现代 Web 应用中得到广泛应用,例如 Apache Shiro 等。这些框架将授权策略的实现封装在内部,在提高开发效率的同时也使访问控制漏洞检测变得更为复杂。在预期授权策略的获取方面,随着自签名的令牌如 JWT(JSON Web Token)逐渐取代会话变量用于状态维护,现有的推断方法大多无法直接使用。在检测技术方面,传统的静态代码分析和动态执行分析可能无法直接获取到框架内部的访问控制逻辑,难以全面检测潜在的漏洞。

5 结 论

Web 应用访问控制漏洞具有高危害性,而漏洞检测技术有助于及时揭露潜在的风险。本文选取 Web 应用访问控制漏洞检测为研究点,根据该类漏洞特征,从预期授权策略的获取和有效漏洞检测的执行 2 个方面,分析现有研究工作的核心思想和存在问题。本文围绕 3 类高风险的访问控制漏洞类型,讨论了现有工作的漏洞检测能力。涉及研究工作的文献详细汇总见附录 A。此外,本文融合访问控制漏洞研究热点与 Web 应用安全架构特征,系统提出了检测技术的未来演进方向。本文旨在为访问控制漏洞检测领域提供一个全面的研究综述,以供后续研究者基于此探索更为有效、精度更高的漏洞检测方法。

参考文献

[1] 刘雅辉, 张铁赢, 靳小龙, 等. 大数据时代的个人隐私保护[J]. 计算机研究与发展, 2015,52(1):229-247.

[2] 王于丁, 杨家海, 徐聪, 等. 云计算访问控制技术研究综述[J]. 软件学报, 2015,26(5):22.

[3] OWASP FOUNDATION. 2021-broken access control[EB/OL]. [2023-07-18]. https://owasp.org/Top10/A01_2021-Broken_Access_Control.

[4] OWASP FOUNDATION. OWASP top 10[EB/OL]. [2023-07-18]. https://owasp.org/www-project-top-ten.

[5] ZHONG L. A survey of prevent and detect access control vulnerabilities[EB/OL]. (2023-04-20)[2023-05-04]. https://arxiv.org/pdf/2304.10600.

[6] LI X, XUE Y. A survey on server-side approaches to securing web applications[J]. ACM Computing Surveys, 2014,46(4):1-29.

[7] DEEPA G, THILAGAM P S. Securing Web applications from injection and logic vulnerabilities: approaches and challenges[J]. Information and Software Technology, 2016,74:160-180.

[8] LI X, YAN W, XUE Y. SENTINEL: securing database from logic flaws in Web applications[C]// Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy. San Antonio, USA: ACM, 2012:25-36.

[9] SUN F, XU L, SU Z. Static detection of access control vulnerabilities in Web applications[C]// Proceedings of the 20th USENIX Conference on Security. San Francisco, USA: USENIX Association, 2011:11.

[10] GHORBANZADEH M, REZA SHAHRIARI H. ANOVUL: detection of logic vulnerabilities in annotated programs via data and control flow analysis[J]. IET Information Security, 2020,14(3):352-364.

[11] 刘奇旭, 靳泽, 陈灿华, 等. 物联网访问控制安全性综述[J]. 计算机研究与发展, 2022,59(10):2190-2211.

[12] 沈海波, 洪帆. 访问控制模型研究综述[J]. 计算机应用研究, 2005,22(6):3.

[13] 韩道军, 高洁, 翟浩良, 等. 访问控制模型研究进展[J]. 计算机科学, 2010,37(11):29-33,43.

[14] LI X, SI X, XUE Y. Automated black-box detection of access control vulnerabilities in Web applications[C]// Proceedings of the 4th ACM Conference on Data and Application Security and Privacy. San Antonio, USA: ACM, 2014:49-60.

[15] GAUTHIER F, MERLO E. Fast detection of access control vulnerabilities in PHP applications[C]// 2012 19th Working Conference on Reverse Engineering. Kingston, UK: IEEE, 2012:247-256.

[16] SON S, MCKINLEY K S, SHMATIKOV V. Fix me up: repairing access-control bugs in Web applications[C]// The 20th Annual Network and Distributed System Security Symposium. San Diego, USA: The Internet Society, 2013:1-16.

[17] KONG Y, ZHANG Y, FANG Z, et al. Static detection of logic vulnerabilities in Java Web applications[C]//2012

IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications. Liverpool, UK: IEEE, 2012:1083-1088.

[18] OWASP FOUNDATION. OWASP API security top 10[EB/OL]. [2023-07-18]. https://owasp.org/API-Security.

[19] OWASP FOUNDATION. API5:2023 broken function level authorization[EB/OL]. [2023-07-18]. https://owasp.org/API-Security/editions/2023/en/0xa5-broken-function-level-authorization.

[20] OWASP FOUNDATION. API1:2023 broken object level authorization[EB/OL]. [2023-07-18]. https://owasp.org/API-Security/editions/2023/en/0xa1-broken-object-level-authorization.

[21] OWASP FOUNDATION. API3:2023 broken object property level authorization [EB/OL]. [2023-07-18]. https://owasp.org/API-Security/editions/2023/en/0xa3-broken-object-property-level-authorization.

[22] RAHMAN K, IZURIETA C. A mapping study of security vulnerability detection approaches for web applications[C]// 2022 48th Euromicro Conference on Software Engineering and Advanced Applications. Gran Canaria, Spain: IEEE, 2022:491-494.

[23] ZHANG B, LI J, REN J, et al. Efficiency and effectiveness of Web application vulnerability detection approaches: a review[J]. ACM Computing Surveys, 2022, 54(9):1-35.

[24] WANG Q, SUN J, WANG C, et al. Access control vulnerabilities detection for Web application components[C]//2020 IEEE 6th International Conference on Big Data Security on Cloud, IEEE International Conference on High Performance and Smart Computing, and IEEE International Conference on Intelligent Data and Security. Baltimore, USA: IEEE, 2020:24-28.

[25] RAFIQUE S, HUMAYUN M, HAMID B, et al. Web application security vulnerabilities detection approaches: a systematic mapping study[C]// 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. Takamatsu, Japan: IEEE, 2015:1-6.

[26] CHLIPALA A. Static checking of dynamically-varying security policies in database-backed applications[C]// Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. Vancouver, Canada: USENIX Association, 2010:105-118.

[27] LUONG T N, VO D H, TO V K, et al. On the compliance of access control policies in Web applications[C]// International Conference on Context-Aware Systems and Applications. Viet Tri City, Vietnam: Springer, 2018: 58-69.

[28] PISKACHEV G, PETRASCH T, SPÄTH J, et al. AuthCheck: program-state analysis for access-control vulnerabilities[C]// Formal Methods 2019 International Workshops. Porto, Portugal: Springer, 2020:557-572.

[29] CORRADINI D, PASQUA M, CECCATO M. Automated black-box testing of mass assignment vulnerabilities in RESTful APIs[C]//2023 IEEE/ACM 45th International Conference on Software Engineering. Melbourne, Australia: IEEE, 2023:2553-2564.

[30] THE LINUX FOUNDATION. OpenAPI[EB/OL]. [2023-07-18]. https://www.openapis.org.

[31] GHORBANZADEH M, SHAHRIARI H R. Detecting application logic vulnerabilities via finding incompatibility between application design and implementation[J]. IET Software, 2020,14(4):377-388.

[32] NEAR J P, JACKSON D. Finding security bugs in Web applications using a catalog of access control patterns[C]// Proceedings of the 38th International Conference on Software Engineering. Austin, USA: ACM, 2016:947-958.

[33] ZHU J, CHU B, LIPFORD H, et al. Mitigating access control vulnerabilities through interactive static analysis[C]// Proceedings of the 20th ACM Symposium on Access Control Models and Technologies. Vienna, Austria: ACM, 2015:199-209.

[34] ZHU J, XIE J, RICHTER L H, et al. Supporting secure programming in Web applications through interactive static analysis[J]. Journal of Advanced Research, 2013, 5(4):449-462.

[35] MONSHIZADEH M, NALDURG P, VENKATAKRISHNAN V N. MACE: detecting privilege escalation vulnerabilities in Web applications[C]//Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. Scottsdale, USA: ACM, 2014:690-701.

[36] 夏志坚, 彭国军, 胡鸿富. 基于权限验证图的 Web 应用访问控制漏洞检测[J]. 计算机工程与应用, 2018, 54(12):63.

[37] SON S, MCKINLEY K S, SHMATIKOV V. RoleCast:

finding missing security checks when you do not know what checks are[C]//Proceedings of the 2011 ACM International Conference on Object Oriented Programming Systems Languages and Applications. Portland, USA: ACM, 2011:1069-1084.

[38] FELMETSGER V, CAVEDON L, KRUEGEL C, et al. Toward automated detection of logic vulnerabilities in Web applications[C]//Proceedings of the 19th USENIX Conference on Security. Washington, USA: USENIX Association, 2010:10.

[39] COVA M, BALZAROTTI D, FELMETSGER V, et al. Swaddler: an approach for the anomaly-based detection of state violations in Web applications[C]//Recent Advances in Intrusion Detection. Berlin, Germany: Springer, 2007:63-86.

[40] BOCIC I, BULTAN T. Finding access control bugs in Web applications with CanCheck[C]//Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. Singapore: ACM, 2016:155-166.

[41] KUMARSHRESTHA A, SINGH MAHARJAN P, PAUDEL S. Identification and illustration of insecure direct object references and their countermeasures[J]. International Journal of Computer Applications, 2015,114(18): 39-44.

[42] VITHANAGE N M, JEYAMOHAN N. WebGuardia: an integrated penetration testing system to detect web application vulnerabilities[C]//2016 International Conference on Wireless Communications, Signal Processing and Networking. Chennai, India: IEEE, 2016:221-227.

[43] ZUO C, ZHAO Q, LIN Z. AUTHSCOPE: towards automatic discovery of vulnerable authorizations in online services[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. Dallas, USA: ACM, 2017:799-813.

[44] KUSHNIR M, FAVRE O, RENNHARD M, et al. Automated black box detection of http get request-based access control vulnerabilities in Web applications[C]//Proceedings of the 7th International Conference on Information Systems Security and Privacy. Online: Science and Technology Publications, 2021:204-216.

[45] RENNHARD M, KUSHNIR M, FAVRE O, et al. Automating the detection of access control vulnerabilities in Web applications[J]. SN Computer Science, 2022, 3(5):376.

[46] HADAVI M A, BAGHERDAEI A, GHASEMI S. IDOT: black-box detection of access control violations in Web applications[J]. The ISC International Journal of Information Security, 2021,13(2):117-129.

[47] PRATAMA I P A E, RHUSULI A M. Penetration testing on Web application using insecure direct object references(IDOR) method[C]//2022 International Conference on ICT for Smart Society. Bandung, Indonesia: IEEE, 2022:01-07.

[48] PAN L, COHNEY S, MURRAY T, et al. EDEFuzz: a Web API fuzzer for excessive data exposures[C]//Proceedings of the 46th IEEE/ACM International Conference on Software Engineering. Lisbon, Portugal: ACM, 2024: 530-541.

[49] LI X, XUE Y. BLOCK: a black-box approach for detection of state violation attacks towards Web applications[C]//Proceedings of the 27th Annual Computer Security Applications Conference. Orlando, USA: ACM, 2011: 247-256.

[50] LI X, XUE Y. LogicScope: automatic discovery of logic vulnerabilities within Web applications[C]//Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. Hangzhou, China: ACM, 2013:481-486.

[51] 文硕, 许静, 苑立英, 等. 基于策略推导的访问控制漏洞测试用例生成方法[J]. 计算机学报, 2017, 40(12):2658-2670.

[52] WEN S, XUE Y, XU J, et al. Lom: discovering logic flaws within MongoDB-based Web applications[J]. International Journal of Automation and Computing, 2017, 14(1):106-118.

[53] 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 and Computer Applications, 2018,109:89-109.

[54] SON S, SHMATIKOV V. SAFERPHP: finding semantic vulnerabilities in PHP applications[C]//Proceedings of the ACM SIGPLAN 6th Workshop on Programming Languages and Analysis for Security. San Jose, USA: ACM, 2011:1-13.

[55] 王博, 卢思睿, 姜佳君, 等. 基于动态分析的软件不变量综合技术[J]. 软件学报. 2020,31(6):1681-

1702.

[56] PAN L, MURRAY T, COHNEY S, et al. Detecting excessive data exposures in Web server responses with metamorphic fuzzing[EB/OL]. (2023-01-23)[2023-07-18]. https://arxiv.org/pdf/2301.09258.

[57] 张健, 张超, 玄跻峰, 等. 程序分析研究进展[J]. 软件学报, 2018,30(1):80-109.

[58] KHALID M N, FAROOQ H, IQBAL M, et al. Predicting web vulnerabilities in Web applications based on machine learning[C]// International Conference on Intelligent Technologies and Applications. Singapore: Springer, 2019: 473-484.

[59] OWASP FOUNDATION. Testing for insecure direct object references [EB/OL]. [2023-07-18]. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/04-Testing_for_Insecure_Direct_Object_References.

[60] LOU J, FU Q, YANG S, et al. Mining invariants from console logs for system problem detection[C]// Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference. Boston, USA: USENIX Association, 2010:1-14.

A survey of Web application access control vulnerability detection

MENG Haining, LU Jie, LI Haofeng, HUANG Yongheng, CAO Liqing, LI Lian

(State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

(University of Chinese Academy of Sciences, Beijing 100049)

Abstract

Access control plays an important role in maintaining the security of Web applications. However, insecure access control has become one of the biggest threats to Web application security. To protect the security of Web applications, it is crucial to detect and report access control vulnerabilities at an early stage. To provide security researchers and program developers with a deeper understanding of Web application access control vulnerabilities and existing detection technologies, this article systematically summarizes the relevant work in this field, explains the basic concepts and characteristics of access control vulnerabilities, and explains the main steps and core ideas for detecting access control vulnerabilities. In addition, this paper analyzes and discusses in detail the vulnerability detection capabilities in existing work for three types of high-risk access control vulnerabilities at the function, object, and object attribute levels. Finally, the development of Web application access control vulnerability detection technology is summarized and the future research directions are given.

Key words: Web application, access control, access control vulnerabilities, vulnerability detection, unauthorized access, software flaw detection

附录 A

表 A-1 Web 应用访问控制漏洞检测研究汇总

文献序号	年份	漏洞类型			检测技术		检测自动化		预期策略获取			预期策略表示		
		BFLA	BOLA	BOPLA	静态	动态	人工参与	自动	人工	静态	动态	不变量	安全约束	执行流程
[39]	2007	√				√	√				√	√		
[38]	2010	√	√		√			√			√	√		
[26]	2010	√	√		√			√		√			√	
[9]	2011	√			√			√		√			√	
[49]	2011	√				√		√			√	√		
[37]	2011	√			√			√		√			√	
[54]	2011	√			√			√		√			√	
[8]	2012	√	√			√		√			√	√		
[15]	2012	√			√			√	√				√	
[17]	2012	√			√			√		√			√	
[16]	2013	√			√			√	√				√	
[50]	2013	√				√	√		√					√
[14]	2014	√	√			√		√			√		√	
[35]	2014	√	√		√			√		√			√	
[33,34]	2015	√			√			√	√				√	
[41]	2015		√			√	√				√	√		
[32]	2016	√	√		√			√	√				√	
[40]	2016	√	√		√			√			√		√	
[42]	2016		√			√	√				√	√		
[43]	2017	√	√			√		√			√	√		
[51]	2017	√	√			√		√			√		√	
[52]	2017	√				√		√				√		√
[27]	2018	√			√			√	√				√	
[36]	2018	√			√			√		√			√	
[53]	2018	√	√			√		√			√	√		
[58]	2019	训练数据集			机器学习			√	训练数据集			-	-	-
[28]	2019	√			√				√	√			√	
[31]	2020	√	√		√			√	√					√
[10]	2020	√	√		√			√	√				√	
[44,45]	2021	√	√			√		√			√	√		
[46]	2021		√			√		√			√	√		
[47]	2022		√			√					√	√		
[29]	2023			√		√		√	√				√	
[48]	2023			√		√		√			√	√		