基于响应相似性判定的 Web 越权漏洞测试方法

宋　虹, 马俊龙, 王伟平, 诸亿郎, 王建新

中南大学计算机学院, 长沙 中国 410012

摘要　Web 越权漏洞是一种允许攻击者以未授权的身份访问其他用户数据的高频 Web 应用漏洞。目前常用的越权漏洞人工测试方法主要依赖于安全专家对 Web 应用进行人工测试, 效率低下, 且对测试人员的专业要求较高; 而现有的自动化漏洞测试方法主要针对 Web 应用中常见的 XSS、SQL 注入等漏洞, 受网站业务逻辑异构性的影响, 不适用于 Web 越权漏洞的检测。针对上述问题, 本文提出了一种基于响应相似性判定的 Web 越权漏洞黑盒测试方法, 该方法能够依据不同身份用户对同一访问接口的返回结果之间的差异性, 推测接口需要的访问控制权限, 从而发现具有访问权限要求的越权待测接口, 降低了模糊测试所需的测试用例。然后通过替换访问请求中的身份标志, 生成原用户的正常请求和越权用户的越权请求作为测试用例对待测接口进行测试, 进而依据返回结果的相似性判定是否存在越权漏洞。在判定方法上采用 Web 响应结构相似性来判定属于同一接口的流量, 采用 Web 响应内容相似性来判定越权待测接口和越权漏洞的存在与否。我们对开源网站和实际网站数据集进行了测试, 结果表明, 该方法能检测出开源网站中所有已知的越权漏洞, 同时检测出了若干个之前未知的越权漏洞, 并通过人工方式得到了验证。

关键词　越权漏洞; 黑盒测试; Web 安全

中图法分类号　TP311　DOI 号 10.19363/J.cnki.cn10-1380/tn.2025.03.02

Black-box Testing Method for Web Authentication Bypass Vulnerability Based on Response Similarity Determination

SONG Hong, MA Junlong, WANG Weiping, ZHU Yilang, WANG Jianxing

School of Computer Science and Engineering, Central South University, Changsha 410012, China

Abstract　Web authentication bypass vulnerability is a prevalent web application vulnerability which allows attackers to access other users' data with unauthorized identity. Commonly, Web authentication bypass vulnerabilities are tested by manual methods, which heavily rely on security expert knowledge. Manual testing leads to low efficiency and high professional requirements for testers. And due to the impact of website business logic heterogeneity, the existing automated vulnerability testing methods are not suitable for the detection of Web unauthorized vulnerabilities, because they primarily target common vulnerabilities in web applications, such as XSS vulnerability and SQL injection. In view of the above problems, we propose a black-box testing method for web authentication bypass vulnerability based on response similarity determination. The proposed method can automatically identify unauthorized interface according to the required access control permissions inferred by the difference among the returned results of various users' accessing the same interface. These identified unauthorized interfaces can used to yield test cases of specific access permissions, reducing some purposeless test cases and improving the efficiency of fuzz testing. Then, the new method replaces identity marks in access requests, generates request test cases for authorized users and unauthorized users to attempt testing bypass authentication actions of the targeted interfaces. Finally, according to the similarity of the returned results, the proposed method determines that whether there is an unauthorized vulnerability. At the determination stage of new method, the HTTP response structure similarity is adopted to determine the traffic of same interface, as well as the HTTP response content similarity is adopted to determine the unauthorized interface to be tested and the existence of authentication bypass vulnerabilities. In order to verify the effectiveness and feasibility of the proposed method, we use the data sets of open-source websites and actual websites. Results show that our new proposed method can detect all known-authentication bypass vulnerabilities in the open-source websites and several previously unknown authentication bypass vulnerabilities which have been verified manually.

Key words　authentication bypass vulnerability; black-box testing; web security

通讯作者: 王伟平, 博士, 教授, Email: wpwang@csu.edu.cn。

本课题得到国家自然科学基金(No. 62272486)资助。

收稿日期: 2023-06-19; 修改日期: 2023-08-03; 定稿日期: 2025-01-14

1 引言

Web 越权漏洞属于 2021 年 OWASP Top 10 中排行第一的失效访问控制类别[1], 是 Web 应用中出现频率较高的漏洞之一, 表现为 Web 应用在核验用户访问权限时存在疏漏, 使得攻击者能够通过强制访问或者篡改请求等方式以未经允许的身份权限获取用户隐私信息和网站资产数据。相比于 XSS[2]、CSRF[3]、SQL 注入[4]等传统 Web 应用漏洞, 越权漏洞往往更常见, 且造成的威胁由于与业务息息相关而并不亚于前者。

如图 1 所示, Web 越权按越权角色的不同, 可以划分为未授权访问漏洞、水平越权漏洞、垂直越权漏洞三种:

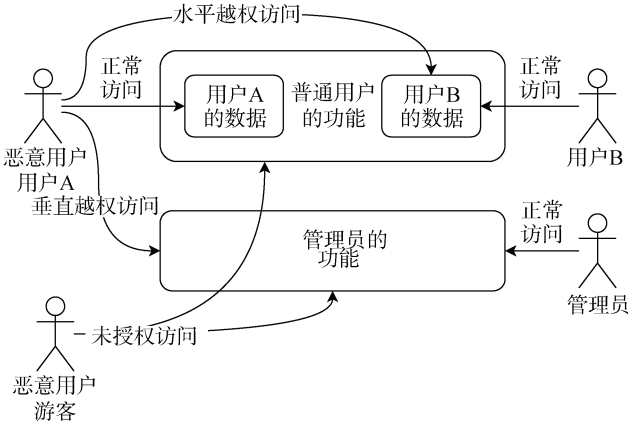


图 1 越权漏洞

Figure 1 Authentication bypass vulnerability

(1) 未授权访问漏洞是指没有登录的游客用户, 可以访问需要认证授权用户才能获取的网站数据。该漏洞包含游客未授权访问普通用户的数据和未授权访问管理员的数据两种情况;

(2) 水平越权漏洞是指登录的普通用户 A 可以访问只有普通用户 B 才能获取的网站数据, 反之亦成立。这种情况往往是由于 Web 应用只确认了普通用户有没有登录, 而没有具体区分访问数据的是哪个用户;

(3) 垂直越权漏洞是指登录的普通用户可以访问只有管理员才能获取的网站数据。这种情况往往是由于 Web 应用只确认了用户是否登录, 而没有具体区分访问数据的是普通用户还是管理员。

而针对 Web 应用的漏洞检测, 按源代码是否可见, 分为白盒测试[5-6]和黑盒测试[7]。白盒测试通过直接对应用源代码进行分析来发现其中的缺陷, 黑盒测试则是在不知道应用源代码的情况下, 构造相应的输入, 通过应用响应来判断漏洞的存在与否。

受网站业务逻辑设计异构性的影响, 人工黑盒测试是目前 Web 越权漏洞黑盒测试的主要方式, 测试人员需要依靠经验发现 Web 应用中潜在的越权漏洞待测接口, 并通过强制访问或者修改网站访问接口请求参数等方式来执行测试, 需要消耗较多的人力资源, 存在操作步骤繁琐、检测效率低下、测试人员门槛高等弊端, 难以实现对大量 Web 应用越权漏洞的批量检测。

针对越权漏洞的特点, 本文提出了一种不需要应用源代码, 通过测试账户遍历网站生成的访问流量, 自动识别越权待测接口并进行漏洞检测方法 BT_WABV(Black-box test method for web authentication bypass vulnerability), 主要创新研究如下:

(1) 针对现代网站多种接口形式, 设计了一种结合 URL 路径、参数和响应结构相似性的接口访问流量划分方法, 在不依靠服务器信息的前提下, 可以将网站的访问流量精准划分为访问不同接口的流量; 在此基础上, 提出了一种基于响应相似性的待测接口自动判定方法, 通过分析不同权限用户对同一接口的访问流量, 可以自动推断接口的访问权限要求, 并将具有特定访问权限要求的接口确定为越权待测接口;

(2) 设计了一种从用户的访问流量中构造正常的 HTTP 请求作为正常请求, 并通过替换请求中的身份标志来生成越权请求作为测试用例的方法, 在保证越权测试覆盖率的同时降低了测试用例的数量; 在此基础上, 通过正常响应与越权响应之间的内容对比来判断是否存在越权漏洞;

(3) 对多个开源网站和实际校园网站进行了测试, 成功检出了开源网站所有的已知越权漏洞, 同时发现并验证了被测网站中存在的未知越权漏洞。

本文第 2 节介绍越权漏洞检测的相关工具和研究现状。第 3 节介绍本文设计的基于响应相似度的 Web 越权漏洞黑盒测试方法的整体流程。第 4、5 节分别阐述了越权待测接口判定流程和越权模糊测试流程。第 6 节通过在网站测试和分析结果验证了所提方法的有效性。第 7 节总结全文的研究工作。

2 相关工作

目前针对越权漏洞的研究方法主要分为黑盒和白盒两大类:

白盒测试通过对 Web 应用的源代码进行分析从而找出越权漏洞。Nemesis[12]使用动态信息流跟踪的方法来检测 Web 应用中的越权漏洞, 但其需要应用

设计者提供资源访问控制列表。针对这个问题, Sun 等人[13]通过对源代码的分析构建不同权限用户的访问地图, 以此判断垂直越权漏洞。而 AuthCheck[14] 则是构建了 Spring Web 应用的有限状态机进行分析以发现越权漏洞。MACE[15]新定义了上下文授权一致性的概念, 通过检测 Web 应用的上下文授权一致性, 来发现 Web 应用中的授权相关漏洞。而 Ma 等人[16]则是在上下文授权一致性的基础上提出了一种可用于检测越权漏洞的五层访问控制模型, 结合授权上下文和五层模型来判断系统是否存在越权漏洞。但白盒测试均要求获取 Web 应用的源代码以解析应用的逻辑结构, 仅适用于相同框架和语言的 Web 应用。

相比于白盒测试, 黑盒测试通常通过构造相应的输入, 依据待测目标的响应来判断漏洞是否存在。

FlowWatcher[17]通过监控 HTTP 流量中的数据在不同的用户之间传播, 检测未经授权的数据泄露。但 FlowWatcher 需要代码编写者提供 Web 应用预期访问策略。

BLOCK[18]针对越权攻击的检测, 从正常 HTTP 请求响应序列及其相关会话信息中构造正常工作的有限状态机, 并从中提取参数的约束, 进而将违反约束的 HTTP 请求判定为越权攻击。LogicScope[19] 和 DetLogic[20]则是依据正常访问 Web 应用流量信息检测越权漏洞, 将 Web 应用的正常行为构建成一个有限状态机, 结合服务器上用于判断用户权限的会话信息(如用户角色、允许权限等)来获取预期的访问控制策略, 然后生成违反访问控制策略的攻击请求来检测越权漏洞。但这些方法均需要在 Web 服务器上设置插件以获取对应的会话信息, 并且需要针对不同类型的 Web 服务器进行适配。

现有方法需要 Web 应用的预期访问策略, 或是用户无法获取的服务器信息才能进行越权测试。与现有方法不同, 本文提出了一种 Web 越权漏洞黑盒测试方法BT_WABV, 该方法仅需要不同身份的客户端流量数据, 不依赖于服务器的专有信息, 就可以进行三种类型的越权漏洞检测。BT_WABV 独立于应用语言、源代码和服务器实现, 适用不同语言和框架的 Web 应用。

3 自动测试越权漏洞的方法 BT_WABV

本文依据人工黑盒测试越权漏洞的思想, 设计实现了一种自动测试越权漏洞的方法BT_WABV, 总体流程如图 2 所示, 分为越权待测接口判定和越权测试两个阶段:

(1) 越权待测接口判定

由于只有具有访问权限的接口才可能具有越权漏洞, 因此越权漏洞待测接口判定的目的是从访问流量识别出网站中需要用户权限的访问接口, 定位到需要模糊测试的接口。

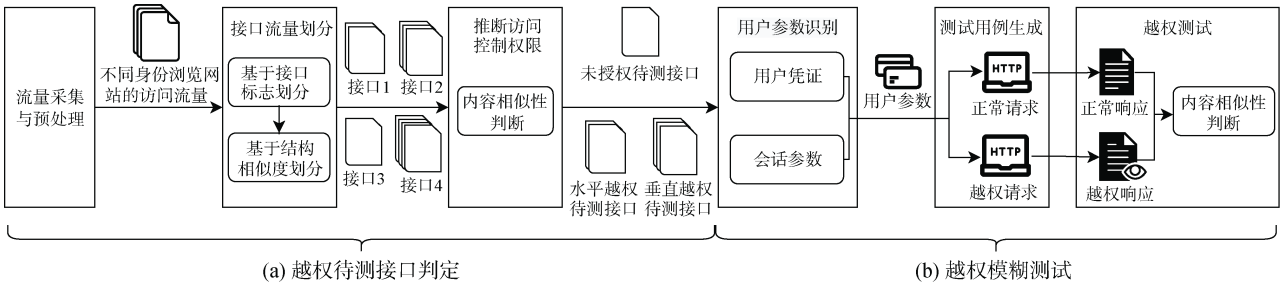


图 2 BT_WABV 的整体流程

Figure 2 The architecture of BT_WABV

基本方法如图 2 所示, 首先进行流量采集, 获取以不同等级用户身份访问遍历网站的流量; 再将流量按照访问接口进行划分; 按照不同身份用户对同一访问接口的返回结果之间的差异性, 推测接口需要的访问控制权限, 以此推断接口是否为待测接口, 需要进行哪种类型的越权测试。

(2) 越权模糊测试

根据(1)确定的待测接口和待测越权类型, 以待测接口的 HTTP 请求为模板, 首先从流量中识别身份标志, 包括用户凭证和会话参数识别, 构建正常请求, 并采用替换参数的方法构建越权请求作为测试用例。发送测试请求, 获取对应的 HTTP 响应, 计算正常响应和越权响应之间的相似性, 当越权用户获得与正常用户相同的响应时, 判断存在对应类型的越权漏洞。

在整个流程中, (1) 中需要重点解决的问题是如何判定访问接口, 由于功能参数的存在, 直接按照 URL、参数名和参数值无法准确划分接口流量。因此我们提出了一种结合 HTTP 请求中的 URL 路径、参数和 HTTP 响应结构相似性的方法, 划分接口访

问流量。而另外一个难题则是如何判断两个返回结果是否一致, 由于动态网站返回的响应内容不固定, 其中包含推荐内容、用户信息和广告等随机内容, 因此我们提出了一个计算响应内容相似度的算法来判断返回结果是否一致。(2) 中的难题则是如何准确地测试越权漏洞是否真实存在, 本文提出了一个基于身份标志替换的越权测试方法, 能够生成符合 Web 应用要求的 HTTP 请求, 并进行越权测试, 具体详见第 4、5 节。

4 越权待测接口判定

为了更好地描述越权待测接口判定步骤, 本文引入下列定义:

用户: 通常, 网站的访问权限是以信息内容无权限、只读和可读写访问来区分用户权限的, 所以定义所对应的访问用户为游客、普通用户和管理员。本文选取一个管理员账户 A、两个普通用户账户 U1、U2 和一个游客账户 G 作为测试用户。

$$user \in \{A, U1, U2, G\}$$

接口: 网站实现单一功能的应用程序接口。一个网站包含了众多的访问接口, 例如一个论坛网站包含观看帖子、发起帖子、修改帖子和给帖子点赞等接口, 用 *API* 表示。

越权待测接口判定的思想如下: 若不同用户对同一接口的访问结果不一致, 或者部分用户无法通过网页正常交互访问到该接口, 则推断该接口要求访问控制权限, 即为待测接口。

4.1 数据采集与预处理

数据采集与预处理的目标是获取四个账户遍历待测网站的访问流量, 并处理流量中的冗余数据和格式不规范数据。

数据采集: 分别使用四个测试用户, 手动点击触发 Web 应用的所有接口, 通过浏览器代理导出所有的访问流量, 可以直接获取 HTTP 和 HTTPS 的请求和响应, 无需额外解密流量;

数据筛选: 通过 HTTP 响应中的 content-type 字段来筛选会产生越权漏洞的 Doc 和 XHR 类型的访问流量;

数据去重: 过滤重复访问接口的冗余访问流量。因为在采集 Web 访问流量时用户可能会多次访问同一个 Web 接口, 导致采集的数据中有许多重复的接口访问流量, 通过去重可以减少数据量, 提高后续数据分析的效率;

数据格式化: 统一 HTTP 响应的格式, 同样的响应返回数据会因为格式和顺序表现为不同的文本, 不利于后继算法的计算, 需要处理成统一的格式。

4.2 接口流量划分

由于需要提取不同用户对同一接口访问的流量进行比对, 因此需要识别出同一接口的流量。

4.2.1 接口的 URL 表示

在以往的研究中有按照网站开发的常用思路, 使用 HTTP 请求中的 URL 路径和参数来划分接口的方法, 但是随着网站开发技术的发展, 出现了请求拥有相同 URL 路径和参数名, 却依靠具体的参数值来区分不同的接口的情况(本文称这类参数为功能参数, 例如图 3 中加粗的参数)。如图 3 中第 3 和第 4 条的情况, 尽管两者 URL 拥有相同的路径和参数名 t.cn/index.php?action&id, 但 action 参数作为功能参数, 提供了 payment 和 delivery_edit 两种不同的功能, 这种情况下是属于两个不同的接口。

由于无法直接从 URL 表示中判定参数是否为功能参数, 因此直接按照 URL、参数名和参数值无法准确划分接口流量。

4.2.2 接口访问流量划分方法

通过实验观察发现, 大部分情况下, 尽管不同用户访问同一接口的响应内容可能有所不同, 但 HTTP 响应的结构是相同的。因此, 本文提出了一种结合 HTTP 请求中的 URL 路径、参数和 HTTP 响应结构相似性的方法, 划分接口访问流量。如图 3 所示, 分为按接口标志划分和按响应结构划分两个步骤。

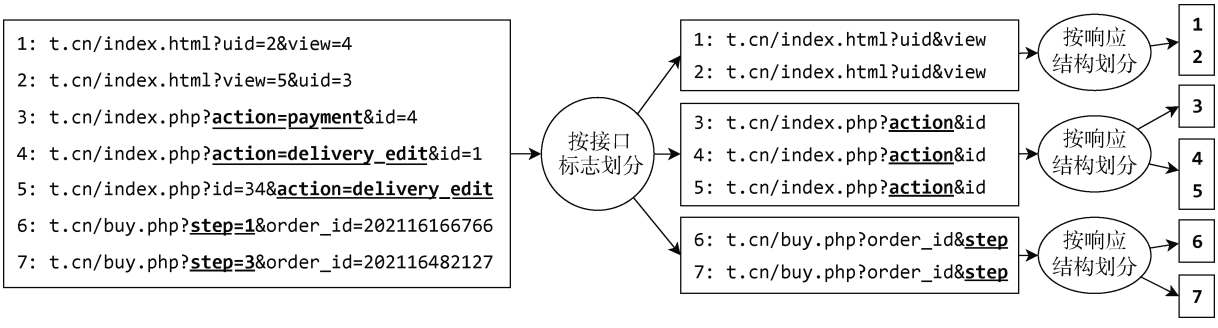


图 3 接口划分示例

Figure 3 An example of web interface division

前一步骤将 URL、参数名相同的流量划分为一组; 后一步骤借助于响应结构相似性进一步将功能参数不同的接口流量分开。

(1) 按接口标志划分

首先提取访问流量中的接口标志, 将具有相同接口标志的访问流量划分为一组;

接口标志具体由 HTTP 请求中的域名、路径、GET 参数名、POST 参数名组成, 表 1 给出了几个例子。由于 GET 参数以及 POST 参数在请求中的排列顺序并不固定, 因此将参数名排序后再作为接口标志, 如表中 1, 从 HTTP 请求 http://t.cn/abc.html?view=5&uid=3 中抽取得到的接口标志为 t.cn/abc.html?uid&view。

表 1 Web 接口标志举例

Table 1 Examples of Web interface flag

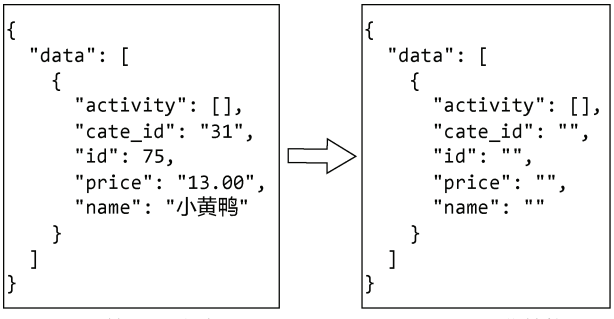
URL	POST 参数	接口标志
t.cn/abc.html?uid=2&view=4	无	t.cn/abc.html?uid&view
t.cn/abc.html?view=5&uid=3	无	t.cn/abc.html?uid&view
a.cn:88/index.php	profile=4, uid=1	a.cn:88/index.php?profile&uid
q.com/c.html?uid=3	fid=5	q.com/c.html?fid&uid

(2) 响应结构相似性

在每一组内按 HTTP 响应的结构是否相似划分接口, 若两个流量的响应结构相似, 则认为是同一接口的流量, 反之则划分为两个不同的接口。

为了定义其相似性, 分别针对 JSON 文本和 HTML 文本定义了其结构相似度, 当相似度大于阈值时, 则认为它们的结构相似。

针对 JSON 格式的响应, 构建了一种名为 JSON 泛化结构的特征文本, 如图 4 所示。



(a) 原始JSON文本 (b) JSON泛化结构

图 4 泛化前后的 JSON 结构示例

Figure 4 An example JSON structure before and after generalization

通过递归将 JSON 文本的最底层值(即 JSON 树的叶子节点值)转换为空值, 以此来泛化具体的 JSON 数据并保留 JSON 结构。对于两个 JSON 文本, 将它们的 JSON 泛化结构的莱文斯坦比[21]作为 JSON 结构相似度 *SJ*, 计算公式如下:

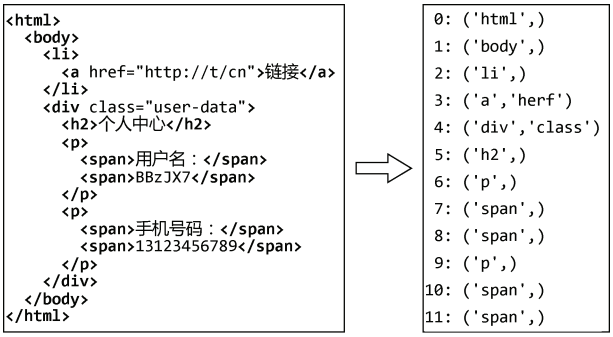
$$SJ=(N-D)/N \tag{1}$$

其中 N 为两个 JSON 泛化结构的总长度, D 为泛化结构的类编辑距离数, $SJ\in[0,1]$, SJ 越大表示两个 JSON 文本的结构越相似, 相对应的 JSON 结构相似度阈值为 $\Phi 1$。选择莱文斯坦比来计算 JSON 结构相似度的原因是: JSON 数据中存在大量的构造字符(括号、逗号、冒号等), 这些构造字符稍有不同, JSON 结构可能就大不相同, 而莱文斯坦比计算了类编辑距离, 可以更好地体现构造字符的结构变化。

HTML 文本由一系列的 Dom 节点组成, 因此本文提取了 HTML 文本中 Dom 节点的标签及属性名构成 HTML 的结构序列, 如图 5 所示。对于两个 HTML 文本, 将它们的结构序列的序列相似度[22]*SH* 作为结构相似度, 计算公式如下:

$$SH=2\times M/T \tag{2}$$

其中 T 为两个 HTML 结构序列中行的总数, M 为两个序列相同行的数量, $SH\in[0,1]$, *SH* 越大表示 HTML 文本的结构越相似。相对应的 HTML 结构相似度阈值为 $\Phi 2$。



(a) 原始HTML文本 (b) HTML结构序列

图 5 泛化前后的 HTML 结构示例

Figure 5 An example of HTML structure before and after generalization

4.3 接口待测越权类型判定

4.3.1 待测接口判定策略

流量按同一接口划分之后, 需要依据属于同一接口的不同用户流量来判定该接口是否验证用户权限。判定方法是直接依据不同类型用户对同一接口的访问结果是否有差异, 存在差异即表示该接口会验证不同用户身份而返回不同的内容, 则直接判定该接口为越权待测接口。

按照访问接口潜在的越权漏洞分类, 将待测接

口分为 3 类: 未授权访问、水平越权以及垂直越权待测接口。

未授权访问是指用户 G 可以获取其他授权用户的访问结果; 在本文定义的四个用户中, 即 G 获取 U1、U2 或 A 的访问结果。

水平越权访问是授权用户可以获取同类型的其他用户的访问结果。即 U1 获取 U2 或 U2 获取 U1 的访问结果。

垂直越权访问是指低权限授权用户可以获取高权限用户的访问结果。即 U1、U2 获得本该属于 A 的访问结果。

越权待测接口的判定算法如算法 1 所示。

算法 1	越权待测接口判定.
输入:	接口的用户访问流量
输出:	接口的越权待测类型以及相应的越权和被越权用户
1:	get α,β from {*A*, U1, U2, *G*} **where** $\alpha\neq\beta$ **and** $L_\alpha\geqslant L_\beta$
2:	**if (** $f(\alpha, API_i)$ exist **and** $f(\beta, API_i)$ not exist **)**
3:	**or (** $f(\alpha, API_i)$ exist **and** $f(\beta, API_i)$ exist **and** $R(\alpha, API_i)$ not similar $R(\beta, API_i)$ **)**:
4:	**if** β is *G*:
5:	API_i is 未授权访问待测接口
6:	**else if** $L_\alpha > L_\beta$:
7:	API_i is 垂直越权待测接口
8:	**else if** $L_\alpha = L_\beta$:
9:	API_i is 水平越权待测接口
10:	**end if**
11:	**If** API_i is 待测接口:
12:	α is 被越权用户
13:	β is 越权用户
14:	**end if**
15:	**end if**

为了更好地描述判定策略, 本文引入以下定义:

(1) L_{user}: 表示用户权限的级别, 满足 $L_A > L_{U1} = L_{U2} > L_G$。

(2) $f(user, API_i)$: 表示用户 *user* 对接口 API_i 的访问流量。包含对该访问接口的请求和响应流量。

(3) $R(user, API_i)$: 表示用户 *user* 访问接口 API_i 的 HTTP 响应流量。

(4) $\alpha\uparrow\beta$: 表示以用户 α 替换用户 β 进行垂直越权测试, 其中 $L_\alpha < L_\beta$。

(5) $\alpha\rightarrow\beta$: 表示以用户 α 替换用户 β 进行水平越权测试, 其中 $L_\alpha = L_\beta$。

(6) $\alpha\hat{}\beta$: 表示以用户 α 替换用户 β 进行未授权访问测试, 其中 α 为未授权用户。

具体的越权待测接口判定策略如表 2 所示, 即对用户集合{A, U1, U2, G}中任意两个组合(*user1*, *user2*), 在某接口的流量进行观察(其中 *user1* 级别高于或等于 *user2*)。若接口的流量满足以下条件之一, 则该接口为待测接口:

(1) 只存在 *user1* 的流量;

(2) *user1* 和 *user2* 的流量都存在, 但返回响应结果不同(具体判定方法见 4.3.2);

表 2 越权待测接口判定策略

Table 2 Decision strategy of unauthorized interface to be tested

用户 2 \ 用户 1	管理员 A	普通用户 U1	普通用户 U2	游客 G
管理员 A				
普通用户 U1	U1↑A		U1→U2	
普通用户 U2	U2↑A	U2→U1		
游客 G	G^A	G^U1	G^U2	

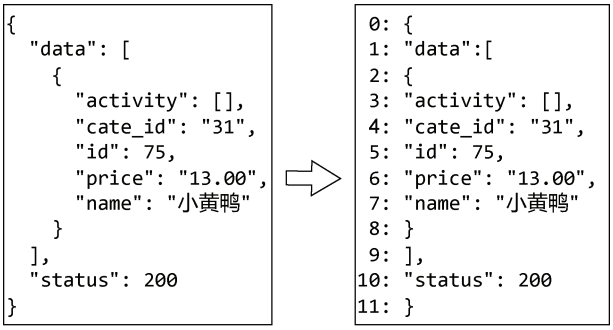
例如某接口中只包含管理员 A 和普通用户 U1 的访问流量, 且两者的响应内容不同, 则在该接口需测试 U1↑A、U2↑A、U2→U1、G^A 和 G^U1 的情况。

4.3.2 响应内容相似性判定

响应内容相似性判定是在判定响应格式相同的基础上, 计算内容相似度, 当内容相似度超过一定的阈值Ψ时, 则认为它们的内容相似。

(1) JSON 响应内容相似性

由于JSON中的键和值都可以表示数据内容, 因此本文将格式化后的JSON文本按行分割成JSON文本行序列, 如图 6 所示。



(a) 原始JSON文本 (b) JSON文本序列

图 6 JSON 文本序列示例

Figure 6 An example of JSON text sequence

采用公式(3)计算 JSON 内容相似度 *CJ*, 其中 *S*

为两个 JSON 文本行的总行数, L 为相同的 JSON 文本行数。相对应的 JSON 内容相似度阈值为 $\Psi 1$。

$$CJ=2\times L/S \tag{3}$$

(2) HTML 响应内容相似性

相较于 JSON 文本, HTML 文本内容较为冗余, 可能包含大量的页面元素(如网页上下导航菜单的文本、较长的公开信息等), 导致两个不同的 HTML 文本的全文字符相似度很高。同时由于网站包含推荐内容、用户信息和广告等随机内容, 可能导致两个相同的 HTML 文本存在少部分的差异。

因此本文设计了一个兼顾全文相似性和差异部分相似性的加权相似度算法。定义行相似度 w、字符相似度 c、差异序列行的字符相似度 d, 分别用于衡量两个页面相同行的比例、全页面字符相同比例、不同行中相同字符比例。

w=2×内容文本的相同行数÷两个内容文本的总行数

c=2×内容文本的相同字符数÷两个内容文本的总字符数

d=2×差异文本的相同字符数÷两个差异文本的总字符数

基于这三个相似度计算 HTML 响应内容相似度 CH, 相对应的 HTML 响应内容相似度阈值为 $\Psi 2$。

$$CH=wc+(1-w)d,\ CH\in[0,1] \tag{4}$$

具体步骤如图 7 所示, 首先提取 HTML 文本中所有文本内容, 构建成一个内容文本序列, 如图 8 所示。

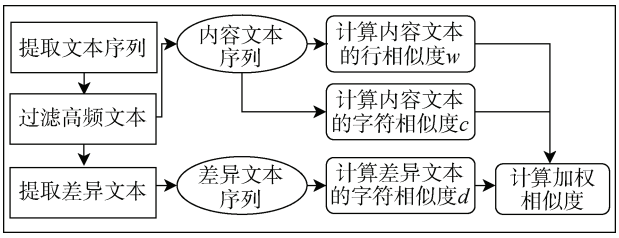
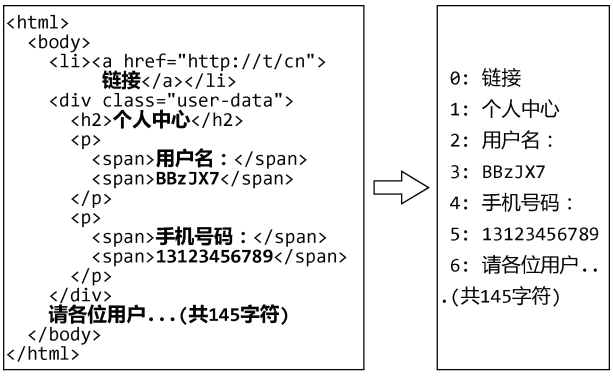


图 7 HTML 内容相似性算法流程

Figure 7 The architecture of HTML content similarity

然后从每个用户访问流量中, 分别统计每个用户的 HTML 内容文本块出现的频数, 并汇总各个用户访问频数前 10%高的内容文本块, 作为整个网站的高频文本集, 其中包含较常出现的页面元素和用户个人信息; 接着从内容文本序列中过滤掉这些高频文本, 得到 HTML 内容文本序列, 并提取内容文本序列的差异文本序列。然后分别计算三个相似度, 并以此为基础计算整个 HTML 响应的内容相似度。



(a) 原始HTML文本 (b) 内容文本序列

图 8 HTML 文本序列示例

Figure 8 An example of HTML text sequence

5 越权模糊测试

识别越权待测接口后, 使用基于身份替换的越权测试方法进行越权模糊测试。其主要思想是, 以不同用户身份登录待测网站, 获取请求中的不同用户身份凭证(可能为 Cookie 字段值和 CSRF-token 值)。在保持登录状态前提下, 按照已经确定的越权待测接口类型, 利用越权者的身份凭证替换请求中被越权者的身份凭证, 发送替换后的请求, 获取响应, 判定其与正常响应的相似性, 当相似性大于阈值时, 则认为替换身份越权后获取了与原身份相同的响应, 则越权成功。

如图 9 所示, 该方法分为三个步骤:

(1) 用户凭证参数识别: 从访问流量中识别并提取请求中表征身份的参数信息, 包括 Cookie 字段值、CSRF-token 值等。

(2) 测试用例生成: 从访问流量中构造能被 Web 服务器接受的 HTTP 请求, 并替换请求中的身份凭证参数, 生成相应的正常请求和越权请求作为测试用例;

(3) 越权测试: 向 Web 服务器发送作为测试用例的 HTTP 请求, 根据响应的内容是否相似, 判定接口是否存在相应类型的越权漏洞。

5.1 用户凭证参数识别

因为 HTTP 协议的局限, 服务器必须使用 HTTP 请求中的用户凭证来辨别发起请求的用户身份。而用户凭证包含 HTTP 请求中标识身份的头字段, 如图 10 中的 Cookie 字段; 以及为了防止跨站请求伪造的 GET、POST 参数 CSRF-token, 如图 10 中的参数 CSRF。而这两种用户凭证参数都是与用户相关且高频出现的较长无序字符串。两者具体表现为: 在同一用户的大部分 HTTP 请求中均有出现的参数或字段, 并且同一用户的值在会话过程中保持不变, 而不同

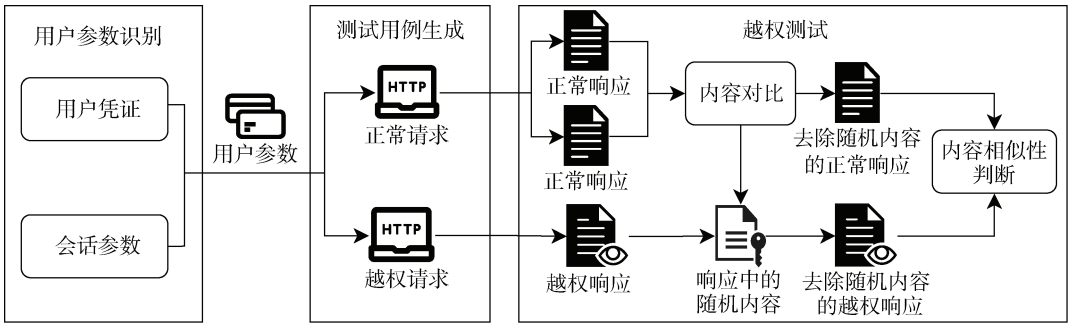


图 9　越权模糊测试过程

Figure 9　Process of bypass fuzzy test

用户的值均不一致。依据两者的表现, 本文识别身份凭证参数的方法为:

(1) 去除 HTTP 请求中常见的与用户凭证无关字段, 包括 Host、User-Agent、Content-Length、Referer、Connection 等 18 个常用字段;

(2) 依次对比同一用户的全部 HTTP 请求, 从请求中筛选出值保持不变且出现频率较高, 同时值类型为较长无序字符串的字段和参数;

(3) 在筛选出的字段和参数中, 再次对比筛选在不同用户请求均有出现且不为相同值的字段和参数, 判定这些字段和参数为身份凭证参数。

识别出 HTTP 请求中的身份凭证参数后, 可以通过替换身份凭证参数的方法, 重构用户的 HTTP 请求, 生成测试用例。

5.2　测试用例生成

根据 4.3.1 的待测接口判定结果, 针对每个待测接口生成相应的测试用例: 从访问流量中构造被越权用户的正常请求和越权用户的越权请求作为一组测试用例, 具体流程如图 10 所示。

(1) 构造正常请求

从待测接口的访问流量中提取被越权用户的 HTTP 请求中的 URL、POST 参数和 HTTP 请求头, 构造被越权用户正常的 HTTP 请求;

(2) 构造越权请求

使用对应越权用户的身份参数来替换正常请求中的身份参数, 构造相应的越权请求。

其中部分网站存在相应的防御措施, 如将 HTTP 请求中参数进行加密, 或是添加签名作为 HTTP 请求的安全验证以防止用户篡改 HTTP 请求。虽然这些防御措施均是在客户端执行的, 理论上是可以从客户端获取相关的算法和公钥进行解密, 但本文更关注的是越权漏洞的测试, 如何自动破解这些防御措施已不是本文的研究内容。面对这种防止篡改 HTTP 请求的防御措施, 本文这种从访问流量中重构 HTTP 请求并替换身份参数的方法将无法构造相应的请求进行测试。

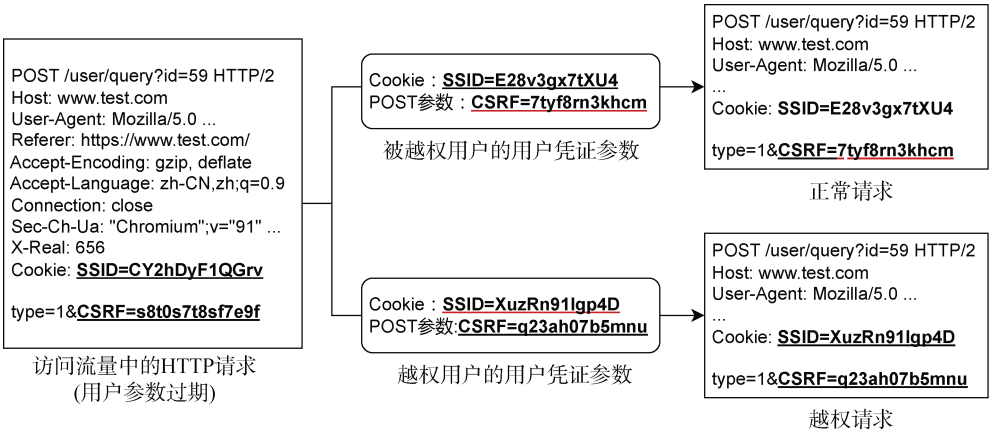


图 10　测试用例生成示例

Figure 10　An example of test case generation

5.3　越权测试

如图 11 所示, 按以下流程使用测试用例对每个待测接口进行越权测试:

(1) 以某一用户身份发送两次正常请求, 获取两个正常响应, 比对得到响应中的动态内容部分(如时间、广告等不固定内容)并去除, 得到去除动态内容

后的正常响应 R1;

(2) 发送越权请求, 获取越权响应。参考(1)中响应中的动态内容位置去除越权响应中的相应部分, 得到越权响应 R2;

(3) 计算 R1 和 R2 的内容相似度, 通过两者相似度是否超过阈值来判断是否存在相应的越权漏洞。

由于响应 R1 和 R2 的数据依旧为 HTML 文本和 JSON 文本, 在判定其相似性时, 采用 3.3 的内容相似性判定算法和阈值。

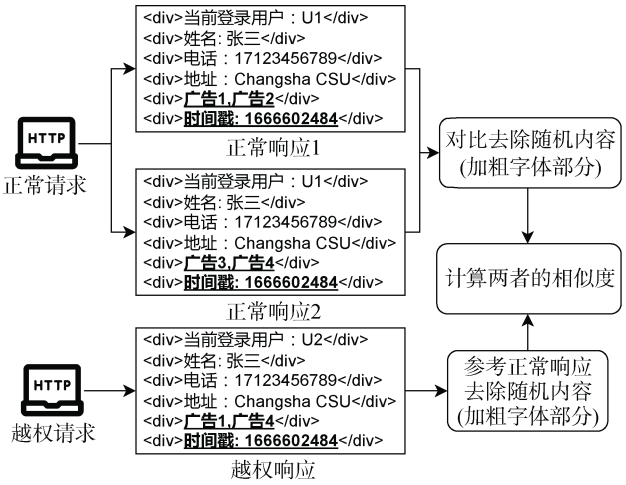


图 11 越权测试示例

Figure 11 An example of bypass test

6 实验评估

6.1 测试数据说明

当前针对越权漏洞的研究缺乏公共的实验数据集, 为了便于验证, 本文选取一些曾被发现越权漏洞或通过修改源码制造越权漏洞的开源 CMS 网站应用和未知漏洞情况的校园网站作为待测 Web 应用, 其中部分测试网站信息如表 3 所示, 通过人工采集数据的方式获取了不同身份用户遍历 Web 应用的访问流量作为本方法的测试数据。

表 3 测试网站

Table 3 The test websites

序号	Web 应用名字	版本	类型
1	74CMS	4.2.3	招聘
2	CRMEB	1.0	电商
3	iWebShop	5.8	电商
4	PHPSHE	1.5	电商
5	StartBBS	1.2.3	论坛
6	极致 CMS	1.7.1	论坛
7	校园网站 1	/	信息管理
8	校园网站 2	/	信息管理
9	校园网站 3	/	信息管理

对于 6 个开源 CMS 网站, 测试用户为: A、U1、U2、G。而对于校园网站, 由于没有获取到管理员用户, 所以测试用户为: U1、U2、G, 即只对校园网站进行水平越权访问和未授权访问漏洞的检测。

6.2 相似度阈值评估

为了获取最优的相似度阈值, 本文分别设计实验评估了结构相似度的阈值 $\Phi 1$ 和 $\Phi 2$, 以及内容相似度的阈值 $\Psi 1$ 和 $\Psi 2$。

$\Phi 1$ 和 $\Phi 2$ 的测试结果如图 12 所示, 本文从测试网站中, 随机抽取了 300 个接口, 采集了它们的访问流量, 设置不同的阈值来进行接口流量划分测试, 经人工验证接口划分的准确率, 本文选取了实验中测试数据划分成功率最高的 0.85 和 0.90 分别作为 $\Phi 1$ 和 $\Phi 2$ 的值。

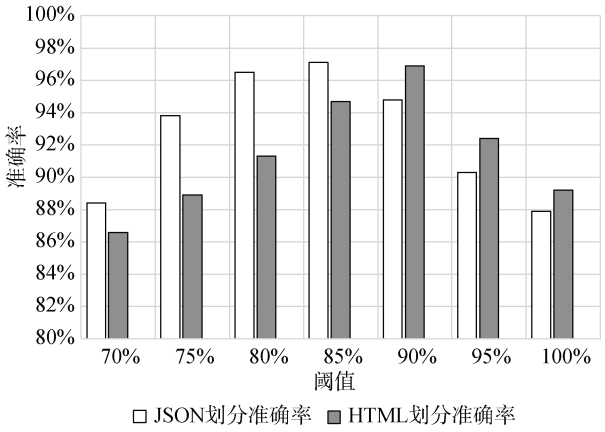


图 12 不同 $\Phi 1$ 和 $\Phi 2$ 下的接口划分准确率

Figure 12 Accuracy of interface division with different $\Phi 1$ and $\Phi 2$

$\Psi 1$ 和 $\Psi 2$ 测试结果如图 13 和图 14 所示, 随机选取 500 个接口响应内容, 测试在不同相似度阈值下的接口响应内容相似性判定效果, 本文选取 $\Psi 1$ 为 0.98, $\Psi 2$ 为 0.95。

与结构相似度阈值相比, 内容相似度阈值显著更高, 这是因为相比于结构差异, 内容差异可能很小, 需要更高的相似度阈值才能判定是否真的内容不同。

6.3 越权待测接口判定准确率

本文通过人工识别统计的方法验证了越权待测接口判定方法的效果, 各网站的越权待测接口判定结果如表 4 所示。表中各项统计数据含义如下:

①实际待测数 C: 实际的越权待测接口数, 人工统计得到;

②判定待测数 P: 判定的越权待测接口数, 本方法判定得到;

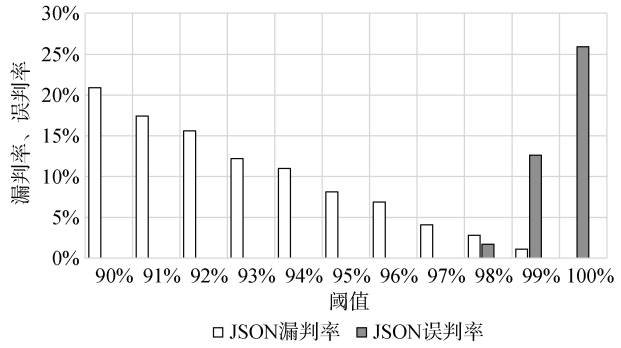


图 13　不同 $\Psi 1$ 下的判定结果

Figure 13　The judgment result of different $\Psi 1$

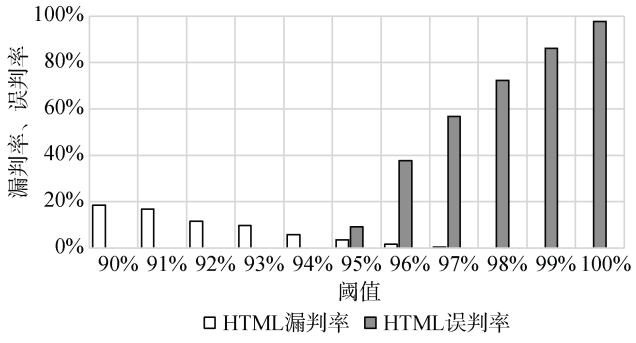


图 14　不同 $\Psi 2$ 下的判定结果

Figure 14　The judgment results of different $\Psi 2$

表 4　越权待测接口判定方法实验结果

Table 4　The result of decision strategy of unauthorized interface to be tested

Web 应用	实际待测数 *C*	判定待测数 *P*	判定正确数 *T*	查全率 *Recall*	查准率 *Precision*	平均判定时长(s)
74CMS	421	440	421	100.0%	95.7%	7.9
CRMEB	310	310	310	100.0%	100.0%	0.5
iWebShop	150	149	149	99.3%	100.0%	6.7
PHPSHE	105	111	105	100.0%	94.6%	0.6
StartBBS	76	76	76	100.0%	100.0%	0.5
极致 CMS	112	114	112	100.0%	98.2%	0.5
校园网站 1	7	7	7	100.0%	100.0%	0.3
校园网站 2	20	20	20	100.0%	100.0%	0.5
校园网站 3	8	8	8	100.0%	100.0%	0.3
平均	134.3	137.2	134.2	99.9%	98.7%	1.98

③判定正确数 *T*: 判定正确的越权待测接口数, 人工统计得到;

④查全率 *Recall*: 表示实际的待测接口中有多少比例被判对, 计算公式如下:

$$Recall=T/C\times 100\% \tag{4}$$

⑤查准率 *Precision*: 表示判定的待测接口中有多少比例被判对, 计算公式如下:

$$Precision=T/P\times 100\% \tag{5}$$

⑥平均判定时长: 多次判定测试 Web 应用越权待测接口的平均耗时。

(1) 查全率分析

经人工分析, iWebShop 漏报了 1 个水平越权待测接口。产生漏报的原因是接口数据划分不充分, 该应用某接口中有 2 个不同功能接口的响应结构非常相似。

(2) 查准率分析

经人工分析, 总计误报了 13 个未授权待测接口、12 个水平越权待测接口、2 个垂直越权待测接口。与 iWebShop 产生漏报的情况刚好相反, 误报是因为某一接口的一些访问流量响应结构差异较大, 导致在划分访问流量时, 将这接口的流量错误拆分成了多个接口。

(3) 平均判定时长分析

本文在一台 CPU 为 Intel Core i7-7700、内存 16G 的 PC 上测试了本方法对 9 个待测 Web 应用判定越权待测接口的平均判定时长。

如表 4 所示, CRMEB、PHPSHE、极致 CMS 等接口数量较少的测试 Web 应用的平均耗时都在 1s 以内, 而采集的访问流量总量较高的 74CMS 和 iWebShop 的平均耗时也没有超过 10s。

6.4　越权测试用例有效率

本文检查了 9 个测试网站中的越权测试用例的执行情况, 一共生成了 1262 个测试用例, 发现除了 CRMEB 网站有 3 个请求超时的水平和垂直越权测试用例失败以外, 其余测试用例均为有效的测试用例, 可以获取到测试网站的正常响应, 图 15 展示了各网站三类越权待测接口的测试用例及其测试成功率。

分析 CRMEB 中的 3 个超时的测试用例, 发现其请求的接口中含有一次性参数, 在该参数重复访问的情况下, Web 应用会不予响应, 导致这 3 个测试用例响应超时。

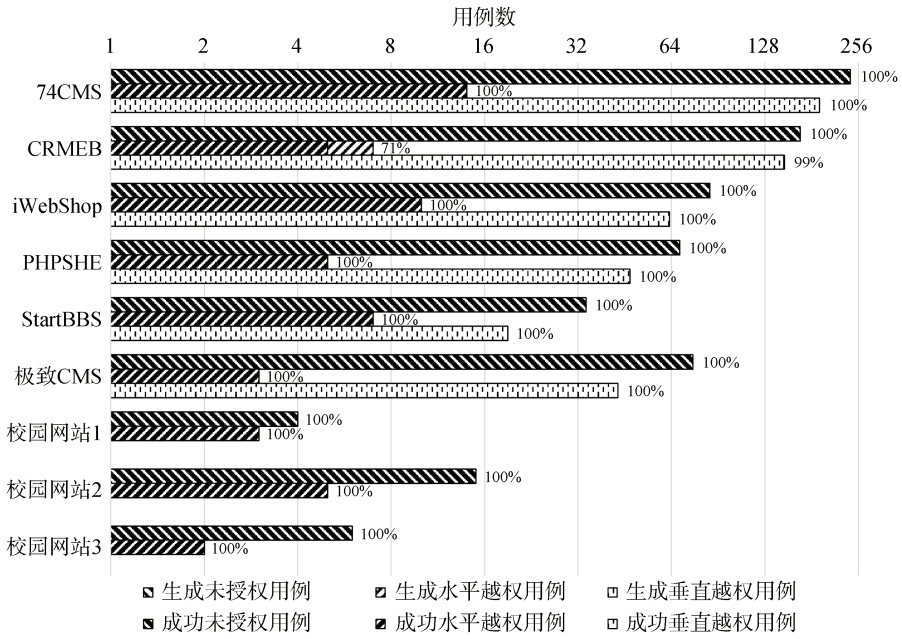


图 15　各网站三类待测接口的测试用例有效率

Figure 15　Efficiency of test cases for three types of interfaces to be tested on each website

6.5　越权测试测试结果

表 5 是本文越权漏洞黑盒测试方法对 6 个开源 CMS 网站和 3 个校园网站的测试结果。

(1) 已知漏洞接口的检出数

根据表 5, 本文的越权漏洞黑盒测试方法, 成功检测出了所有测试 Web 应用的 9 个已知漏洞, 具有较高的越权漏洞查全率。

(2) 新发现漏洞接口数

在测试结果中, 发现了 10 个之前未知的越权漏洞接口。其中 CRMEB、iWebShop 和极致 CMS 发现的新漏洞的都是未授权访问漏洞。经人工分析, 这些越权漏洞获取都是网站的一些公告或通知数据, 而游客正常情况下是无法访问的。而校园网站 1 存在的未授权漏洞是游客身份可以获取任意同学课程考试成绩; 2 个水平越权漏洞是以任何学生身份可以查询任意同学的身份信息和考试成绩。校园网站 2 发现的水平越权漏洞是以任何学生身份可以查询任意同学的身份信息、户籍信息等敏感数据。目前已经将这些漏洞上报到有关部门。

(3) 漏洞接口的误报数

在测试结果中, 只发现了 74CMS 存在一个误报。经分析, 是越权待测接口判定时将接口误判为未授权待测接口, 即游客本来是可以访问该接口的, 然而由于接口访问流量划分过度, 导致游客的访问流量从该接口中被划出, 从而误判该接口游客不能访问, 最终导致误报。

表 5　BT_WABV 测试结果

Table 5　The detection result of BT_WABV

Web 应用	检出漏洞数	已知漏洞数	新发现漏洞数	误报漏洞数	平均测试时长(s)
74CMS	1/1/0	0/1/0	0/0/0	1/0/0	135.9
CRMEB	2/1/0	0/1/0	2/0/0	0/0/0	108.1
iWebShop	2/1/0	0/1/0	2/0/0	0/0/0	23.2
PHPSHE	1/1/0	1/1/0	0/0/0	0/0/0	5.5
StartBBS	0/1/1	0/1/1	0/0/0	0/0/0	2.2
极致 CMS	3/1/0	1/1/0	2/0/0	0/0/0	7.9
校园网站 1	1/2/0	0/0/0	1/2/0	0/0/0	0.3
校园网站 2	0/1/0	0/0/0	0/1/0	0/0/0	0.9
校园网站 3	0/0/0	0/0/0	0/0/0	0/0/0	0.5

表格中的 x/y/z 格式, x、y、z 分别对应未授权访问、水平越权、垂直越权的漏洞数。

(4) 平均测试时长

如表 5 所示, Web 应用的平均测试时间受测试用例数量的多寡而不同, 但即使是测试用例最多的 74CMS 和 CRMEB, 其平均测试时间也不超过 3 分钟, 甚至对于一些功能少接口不多的网站如校园网 3, 可以在 1s 内检测完毕, 相比较于人工测试动辄几个小时的测试时间, 效率获得了显著提升。

表 6 给出了几款常见的漏洞扫描器面向 Web 漏洞检测的能力, 其中典型漏洞是指 SQL 注入、跨站脚本攻击和目录遍历等常见漏洞, 组件漏洞是指使用存在漏洞的组件、框架所导致的漏洞。

表 6 常见漏洞扫描器的功能

Table 6 Common vulnerability scanner features

漏洞扫描器	典型漏洞	中间件漏洞	越权漏洞		
			未授权访问	水平越权	垂直越权
AppScan[8]	✓	✓	✓	×	✓
Nessus[9]	✓	✓	×	×	×
Nikto[10]	✓	✓	×	×	×
Acunetix Web Vulnerability Scanner[11]	✓	✓	×	×	×
椰树 WEB 安全扫描器	✓	✓	×	×	×

可以看出, 在五种扫描器中, 只有 AppScan 支持垂直越权漏洞和未授权访问漏洞的检测, 但不支持水平越权漏洞的检测。

本文选择了相同类型的黑盒测试方法 AppScan 扫描器进行对比, 测试结果如表 7 所示。结果表明本文提出方法的效果优于 AppScan, AppScan 仅支持未授权访问漏洞和垂直越权漏洞的扫描, 并且未能发现所有的已知漏洞, 同时存在 4 个误报。

表 7 AppScan 和 BT_WABV 扫描结果对比

Table 7 The result comparison between AppScan and BT_WABV

Web 应用	总漏洞数	AppScan 检出漏洞数	BT_WABV 检出漏洞数	AppScan 误报漏洞数	BT_WABV 误报漏洞数
74CMS	0/1/0	1/0/0	1/1/0	1/0/0	1/0/0
CRMEB	2/1/0	2/0/0	2/1/0	0/0/0	0/0/0
iWebShop	2/1/0	3/0/0	2/1/0	1/0/0	0/0/0
PHPSHE	1/1/0	3/0/0	1/1/0	2/0/0	0/0/0
StartBBS	0/1/1	0/0/1	0/1/1	0/0/0	0/0/0
极致 CMS	3/1/0	1/0/0	3/1/0	0/0/0	0/0/0

表格中的 x/y/z 格式, x、y、z 分别对应未授权访问、水平越权、垂直越权的漏洞数。

7 结论

本文提出并实现了一种面向 Web 越权漏洞的黑盒测试方法, 包含越权待测接口判定和越权模糊测试两个步骤。解决了现有越权检测方法需要服务器相关会话信息的问题, 只需要客户端捕获的用户流量信息就可以检测三种越权漏洞, 并通过一系列的测试验证了方法的有效性, 但仍存在一些需要完善的地方。首先, 本方法仅考虑单一请求的越权操作, 多操作才能触发的越权问题由于存在操作的随意性, 可能带来路径爆炸问题; 其次, 本文提出的方法对网站访问流量的完整性由较高要求, 访问流量的缺失会导致误报和漏报, 后续需要研究如何自动化获取网站完整的访问流量; 再次, 当前测试的 Web 应用数量偏小, 后续将对互联网上流行的 Web 应用进行补充测试, 进一步验证方法对当前互联网上实际 Web 应用的适用性。

致 谢 本课题得到国家自然科学基金项目(No. 62272486)的资助, 在此表示感谢。

参考文献

[1] OWASP Top 10:2021. The Open Web Application Security Project. https://owasp.org/Top10. Sept. 2021.

[2] Rodríguez G E, Torres J G, Flores P, et al. Cross-Site Scripting (XSS) Attacks and Mitigation: A Survey[J]. *Computer Networks*, 2020, 166: 106960.

[3] Khodayari S, Pellegrino G. {JAW}: Studying Client-side {CSRF} with Hybrid Property Graphs and Declarative Traversals[C]. *30th USENIX Security Symposium,* 2021: 2525-2542.

[4] Tang P, Qiu W D, Huang Z, et al. Detection of SQL Injection Based on Artificial Neural Network[J]. *Knowledge-Based Systems*, 2020, 190: 105528.

[5] Gao F J, Wang Y, Chen T J, et al. Static Checking of Array Index out of Bounds Defects in C Programs Based on Taint Analysis[J]. *Journal of Software*, 2020, 31(10): 2983-3003.
(高凤娟, 王豫, 陈天骄, 等. 基于污点分析的数组越界缺陷的静态检测方法[J]. *软件学报*, 2020, 31(10): 2983-3003.)

[6] Pham V T, Böhme M, Roychoudhury A. Model-Based Whitebox Fuzzing for Program Binaries[C]. *The 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016: 543-553.

[7] Feng X T, Sun R X, Zhu X G, et al. Snipuzz: Black-Box Fuzzing of IoT Firmware *via* Message Snippet Inference[C]. *The 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021: 337-350.

[8] AppScan. HCL Technologies (Hindustan Computers Limited). https://www.hcltechsw.com/appscan. Oct. 2022.

[9] Nessus. tenable. https://www.tenable.com/products/nessus. Oct. 2022.

[10] Nikto. sullo, from Github. https://github.com/sullo/nikto. Sept. 2022.

[11] Acunetix Web Vulnerability Scanner. Acunetix. https://www. acu-

netix.com. Oct. 2022.

[12] Dalton M, Kozyrakis C, Zeldovich N. Nemesis: Preventing Authentication & Access Control Vulnerabilities in Web Applications[C]. *USENIX Security Symposium*, 2022

[13] Sun F Q, Xu L, Su Z D, et al. Static Detection of Access Control Vulnerabilities in Web Applications[C]. *The 20th USENIX conference on Security*, 2011: 11.

[14] Piskachev G, Petrasch T, Späth J, et al. AuthCheck: Program-State Analysis for Access-Control Vulnerabilities[M]. Formal Methods. FM 2019 International Workshops. Cham: Springer International Publishing, 2020: 557-572.

[15] Monshizadeh M, Naldurg P, Venkatakrishnan V N. MACE: Detecting Privilege Escalation Vulnerabilities in Web Applications[C]. *The 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014: 690-701.

[16] Ma L, Yan Y J, Xie H. A New Approach for Detecting Access Control Vulnerabilities[C]. *2019 7th International Conference on Information*, *Communication and Networks*, 2019: 109-113.

[17] Muthukumaran D, O'Keeffe D, Priebe C, et al. FlowWatcher: Defending Against Data Disclosure Vulnerabilities in Web Applications[C]. *The 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015: 603-615.

[18] Li X W, Xue Y, Li X W, et al. Block[C]. *The 27th Annual Computer Security Applications Conference*, 2011: 247-256.

[19] Li X W, Xue Y. LogicScope: Automatic Discovery of Logic Vulnerabilities within Web Applications[C]. *The 8th ACM SIGSAC Symposium on Information*, *Computer and Communications Security*, 2013: 481-486.

[20] 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.

[21] Sarkar S, Das D, Pakray P, et al. JUNITMZ at SemEval-2016 Task 1: Identifying Semantic Similarity Using Levenshtein Ratio[C]. *The 10th International Workshop on Semantic Evaluation*, 2016: 702-705.

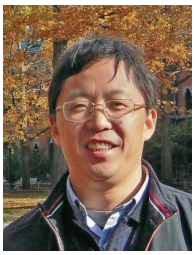
[22] Wołk K, Marasek K. A Sentence Meaning Based Alignment Method for Parallel Text Corpora Preparation[M]. New Perspectives in Information Systems and Technologies, Volume 1. Cham: Springer International Publishing, 2014: 229-237.

宋虹 于 2010 年在中南大学计算机应用技术专业获得博士学位。现任中南大学计算机学院副教授。研究领域为系统安全、网络安全。Email: songhong@csu.edu.cn

马俊龙 于 2020 年在中南大学信息安全专业获得学士学位。现在中南大学计算机技术专业攻读硕士学位。研究领域为网络安全。Email: mjl16@csu.edu.cn

王伟平 于 2004 年在中南大学计算机应用技术专业获得博士学位。现任中南大学计算机学院教授。研究领域为网络安全、软件安全、应用安全。Email: wpwang@csu.edu.cn

诸亿郎 于 2021 年在中南大学计算机科学与技术专业获得硕士学位。研究领域为网络安全。Email: orinoml@csu.edu.cn



王建新 于 2001 年在中南大学计算机应用技术专业获得博士学位。现任中南大学计算机学院教授。研究领域为网络优化与信息安全，计算机算法，生物信息学。Email: jxwang@mail.csu.edu.cn