基于状态偏离分析的Web访问控制漏洞检测方法

马琪灿，武泽慧，王允超，王新蕾

引用本文

马琪灿，武泽慧，王允超，王新蕾.基于状态偏离分析的Web访问控制漏洞检测方法[J]．计算机科学, 2023, 50(2): 346-352.

MA Qican, WU Zehui, WANG Yunchao, WANG Xinlei. Approach of Web Application Access Control Vulnerability Detection Based on State Deviation Analysis [J]. Computer Science, 2023, 50(2): 346-352.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211100166

基于状态偏离分析的 Web 访问控制漏洞检测方法

马琪灿　武泽慧　王允超　王新蕾
信息工程大学数学工程与先进计算国家重点实验室　郑州 450001
(expolit@88.com)

摘　要　攻击者可利用 Web 应用程序中存在的漏洞实施破坏应用功能、木马植入等恶意行为。针对 Web 应用程序的访问控制漏洞的检测问题,现有方法由于代码特征难提取、行为刻画不准确等问题导致误报率和漏报率过高,且效率低下。文中提出了一种基于状态偏离分析的 Web 访问控制漏洞检测方法,结合白盒测试技术,提取代码中与访问控制有关的约束,以此生成 Web 应用程序预期访问策略,再通过动态分析生成 Web 应用程序实际访问策略,将对访问控制漏洞的检测转换为对状态偏离的检测。使用提出的方法开发原型工具 ACVD,可对访问控制漏洞中未授权访问、越权访问等类型的漏洞进行准确检测。在 5 个真实 Web 应用程序中进行测试,发现 16 个真实漏洞,查全率达到了 98%,检测效率较传统黑盒工具提升了约 300%。

关键词:Web 应用程序;访问控制漏洞;逻辑漏洞;有限状态机
中图法分类号　TP311

Approach of Web Application Access Control Vulnerability Detection Based on State Deviation Analysis

MA Qican,WU Zehui,WANG Yunchao and WANG Xinlei
State Key Laboratory of Mathematical Engineering and Advanced Computing,Information Engineering University,Zhengzhou 450001,China

Abstract　Attackers can exploit vulnerabilities in Web applications to implement malicious behaviors such as disrupting application functionality and Trojan implantation. For the detection of access control vulnerabilities in Web applications,existing methods have high false alarm,leakage rates and low efficiency due to the difficulty of extracting code features and inaccurate behavior portrayal. This paper proposes a method for detecting Web access control vulnerabilities based on state deviation analysis,which combines white-box testing techniques to extract access control-related constraints in code to generate Web application expected access policies,and then generates Web application actual access policies through dynamic analysis,converting the detection of access control vulnerabilities into the detection of state deviation. Using this technology to develop the prototype tool ACVD,it is possible to accurately detect the types of access control vulnerabilities such as unauthorized access and ultra vires access. Tested in 5 real Web applications,16 real vulnerabilities are found,and the recall rate reaches 98%,which is about 300% higher than traditional black box tools.

Keywords　Web application,Access control vulnerability,Logic vulnerability,Finite state machine

1　引言

随着互联网的发展,Web 技术对人们生活的影响日益增大。在为人们的日常生活带来便利的同时,针对 Web 应用的攻击数量每年都在急速上升。在对 Web 漏洞的研究中,研究者最开始主要将目光聚焦在注入漏洞上,因为注入类漏洞具有一个统一的逻辑,即用户的输入经过不充分的过滤到达了一个有风险的处理函数,所有的注入类漏洞都具有此类模式,因此,通过对这种数据流传递模式的识别便可以发现这类漏洞[1-4]。而 Web 应用逻辑漏洞因没有固定的发现模式,又与 Web 应用程序的功能有关,因此识别和处理这些漏洞极具挑战性。逻辑漏洞中的访问控制漏洞在 OWASP 2021 Top 10[5]中从第五上升到第一,在其测试的应用中有 94%都存在访问控制漏洞,且该类漏洞均具有较高危险性及易利用性,按照 CVSS 标准可达到高危害标准评分。本文将针对该类漏洞展开研究。

如果用户可以超出其预期权限进行操作,则说明目标系统存在访问控制漏洞,这会导致未经授权的访问、信息泄露以及数据的修改或破坏。访问控制漏洞是因预期的访问控制策略和在网络应用程序中实际执行的策略之间的差异性而形成的,考虑到应用程序的实施通常没有明确的访问控制策略,推导出预期的访问控制策略成为了识别访问控制漏洞的关键步骤。现有的工作提出了一些技术,用于自动提取访问控制策略,但现有方法受到了建立访问控制模型的粗粒度、处理数据实体之间的复杂关系、处理源代码语言与特定平台的要求等问题的限制,而且都不能对跨越多个程序文件的复杂数据

到稿日期:2021-11-15　返修日期:2022-06-21
基金项目:国家重点研发计划(2019QY0501)
This work was supported by the National Key Research and Development Program of China(2019QY0501).
通信作者:武泽慧(wuzehui2010@foxmail.com)

关系进行建模，无法确定由此产生的访问控制问题。对此，本文提出将对访问控制漏洞的分析问题转化为，通过源码分析出应用预期行为与真实访问的实际行为的差异性问题，使得模型粗粒度与数据关系复杂性带来的限制降低到最低。

在当前研究中主要采用的方法有黑盒分析与静态分析，采用静态分析[6-7]分析访问控制漏洞的优点是方便、速度快、资源消耗少，静态扫描不需要应用环境，可以直接开始对控制流的分析，缺点是在分析访问控制漏洞时，静态分析对数据库带来的鉴权差异是难以分析的，对逻辑上存在的差异也很难做出判断，同时也无法对分析结果进行验证。黑盒测试[8-11]的优点是其与编程语言无关，专注于分析应用功能本身的问题，可以跨语言实现对不同应用的分析，缺点是爬虫难以发现所有的功能点，且会做大量无用扫描。传统的自动化黑盒扫描器，如 AWVS[12]，AppScan[13]等通过大量的模糊测试输入，实现对漏洞的发掘，这种方法并不能对访问控制漏洞有很好的挖掘效果，仅能发现部分未授权访问漏洞，且受到黑盒工具本身的局限性，难以获得较高的路径覆盖率。白盒测试不只局限于静态分析，黑盒测试很难通过外部访问来获得应用程序的预期行为，因此通过代码获得应用程序预期行为后，将此作为判断标准可以大大降低动态分析的难度，且在动态分析过程中，采用白盒方法可以使得动态分析有指向性地侧重于存在功能点的部分，实现了对动态测试的路径剪枝，大大提高了分析效率。然而，在白盒静态分析中，不同语言、不同架构、不同开发者都会使得代码具有不一样的风格，实现一种通用方法以解析所有语言和结构的代码几乎不可能实现，在那些商业白盒扫描器（如 Fortify[14]，Coverity[15]）中采用的解决方案是针对不同语言、不同框架提出对应的特征配置文件，这需要极大的工作量与时间成本。在本文研究的方法中，为了使白盒分析技术不会因此局限性受极大约束，使用静态分析仅对代码中的参数及访问控制函数进行提取，以此对后续动态分析作辅助，此时仅需针对编程语言的访问控制函数进行适配及参数提取。也有一些相关研究[16]使用机器学习的方法来实现对访问控制漏洞的识别，逻辑漏洞虽然在 Web 应用中广泛存在，但很难收集足够的漏洞代码对模型进行训练。

本文结合白黑盒优点，提出了基于白盒测试的访问控制漏洞挖掘原型工具 ACVD。该原型工具通过有限状态机（Finite State Machine，FSM）对 Web 应用程序的逻辑进行建模，将访问控制漏洞形式化为预期 FSM 与实际 FSM 之间的差异，由此实现对访问控制漏洞的挖掘。在对逻辑漏洞的分析中也有相关研究[17]通过 FSM 构建模型，经实际测试发现，通过黑盒方法难以实现预期行为的模型提取。本文提出使用源码获取预期访问策略来建立预期行为 Web 模型，与实际行为模型进行状态偏移分析，实现对访问控制漏洞的漏洞挖掘。

本文第 2 节对访问控制漏洞进行分析，明确需要解决的问题；第 3 节提出 ACVD 原型工具的系统架构，以及对 ACVD 工具使用的状态偏移检测方法进行阐述；第 4 节对本文提出的原型工具进行评估，并对性能与相关研究进行对比；最后总结全文并展望未来。

2 问题分析

图 1 给出了包含两个访问控制漏洞的 Web 应用。在代码 1 index.php 与代码 2 functions.php 中存在权限验证函数（虚线框内），而代码 4 del_user.php 通过文件包含的方式引入代码 2，使得在执行后续功能前也需要权限验证。代码 1 index.php 的执行逻辑是检查用户权限，若为‘admin’则可以执行增加用户和删除用户的功能，若为‘user’则只能够执行用户自身资料进行修改的功能。而代码 3 add_user.php 在执行增加用户功能时，并没有检查用户角色是否为‘admin’，任何用户都可以调用此功能。代码 5 edit_user.php 在修改用户资料时，也没有检查传入的‘user_id’是否属于用户，使得用户可以编辑其他用户的资料。

正常的功能调用逻辑应是图 1 中箭头所指向的调用关系，经过 index.php 进行角色检验后执行相关操作，但代码中的漏洞使得攻击者可以直接访问功能函数所在文件进行功能调用的可能。

通过上述分析可知，当出现以下情况时，则存在访问控制漏洞。

（1）只允许角色 r 使用的功能，能被低权限角色执行。

（2）属于用户 a 的私有资源，能被同角色其他用户访问。

上述问题是因缺少访问检查或存在不完整的访问检查所致，因为 Web 应用程序使用无状态的 HTTP 协议，以独立的方式处理每个网络请求和响应，而 Web 应用采用会话（Session）来表示用户的登录状态、操作顺序、权限级别等状态。按照安全访问控制策略，Web 应用程序会在授权访问特权资源或功能前对会话变量进行检查，然而有些页面可能不执行这些检查，或检查存在被绕过的可能，使得恶意用户可以获得本不属于他的页面的访问权。

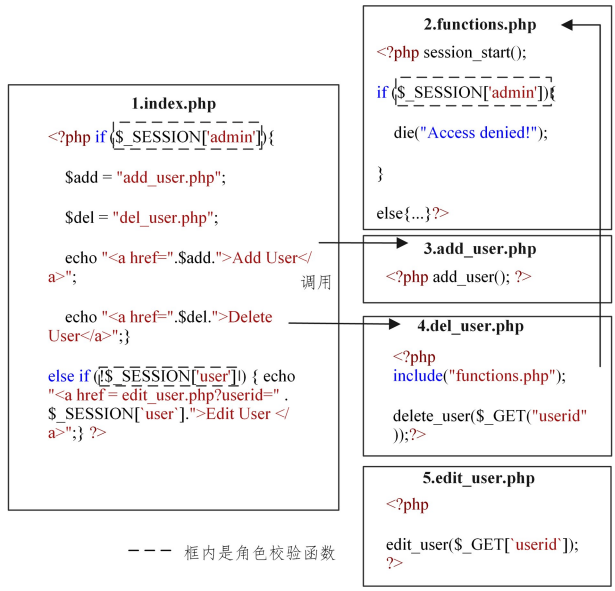


图 1 示例应用程序

Fig. 1 Demo Web application

3 系统设计

通过对访问控制漏洞的分析可知，ACVD 的原型系统拥有 3 个模块：静态分析模块、动态分析模块、决策模块。静态分析模块所做的主要工作是从代码中提取预期行为，其次为动态分析做辅助，动态分析中爬虫探索网站地图需做大量无意义尝试，静态分析通过对应用结构及路由的提取，使得动态

分析需探索的路径得到大规模的剪枝，并通过对参数的提取能够使得动态分析中模糊测试存在定向性，从而使得动态分析变得更加高效。最后的决策模块是应用程序模型分析HTTP请求响应后，构造出当前应用执行的访问控制策略，将其与预期访问策略结合进行分析，判断访问控制漏洞是否存在。系统架构与模块内的关系如图2所示。

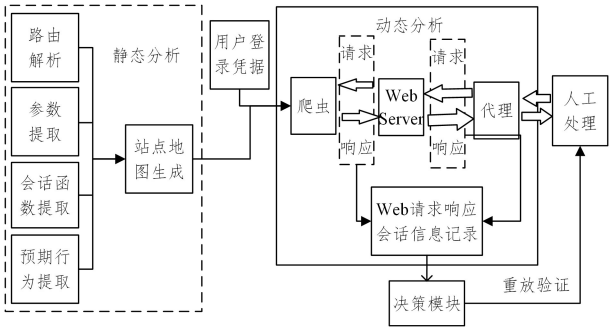


图2 系统架构及模块内关系介绍

Fig. 2 System architecture and relationship of modules

本节首先阐述提出的行为模型建立方法，然后对系统在静态分析阶段、动态分析阶段以及决策阶段采用的状态偏离分析技术进行介绍。各模块的执行流程如图3所示。

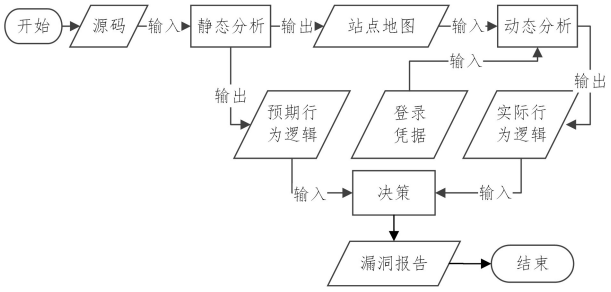


图3 系统执行流程

Fig. 3 System execution flow

3.1 基于有限状态机的Web应用行为建模

通过对第2节所展示的Web应用中访问控制漏洞的分析可知，若要提出一个系统方法用于检测Web应用程序中存在的访问控制漏洞，则需要对不依赖于应用程序功能的访问控制策略进行预期行为分析，并制定能够反映数据流与控制流行为的应用模型。在研究对访问控制漏洞的漏洞模型中，数据流指跨页面请求的参数与会话，控制流是Web应用程序在实现功能过程中，触发应用页面的操作顺序。基于此，分析策略提出了原型系统ACVD，它通过FSM构建Web应用程序行为模型来识别访问控制漏洞。

使用有限状态机(FSM)对网络应用的逻辑进行建模，将访问控制漏洞的发现形式化为预期的FSM和实际的FSM之间的差异性比对。采用静态分析对代码中展现的应用逻辑中的访问控制信息进行分析，建立理想状态下的预期FSM，再与动态分析时应用程序实际运行过程中通过HTTP请求与响应中蕴含的访问控制信息建立的实际FSM寻找差异点。

为了实现对访问控制类漏洞的识别，使用一个FSM模型$(S,s_0,\sigma,\Lambda,A,\delta,F)$对应用进行建模，有限状态机常被用来对应用程序的行为进行建模，因为FSM很适合对任何系统的行为进行建模。在设计的方法中，因对预期访问逻辑的提取是通过代码实现的，为了生成能够进行差异性分析的预期行为模型，将代码中的权限验证函数与其作用域内的应用功能建立约束关系作为状态，将约束关系的变化作为状态的变化，使用FSM建立模型。

在状态机$(S,s_0,\sigma,\Lambda,A,\delta,F)$中，$S$表示有限的状态集，这些状态被表示为Web应用程序具有唯一URL的页面；s_0表示初始状态，初始状态即是Web应用的主页，从这里开始导航；σ表示有限的输入集合；Λ表示输出集合；A是一个三元组$\langle R,P,Se\rangle$，R代表访问该页面的用户的访问权限级别，P代表HTTP请求中的参数的集合及其对应的值；Se代表会话变量的集合及其对应的值；δ表示过渡函数，其定义为从$S\times\sigma\times A$到S的映射；F代表最终状态的集合，指应用程序内最后结束的页面。

通过算法1、算法2实现从源码中对预期行为逻辑的提取，建立Web应用预期行为模型。构建实际行为FSM模型所需的信息是在动态分析中代理服务收集的爬虫、人工访问、服务端的执行痕迹中提取的。通过算法3实现对实际行为模型的建立。

算法1 Web预期行为模型建立算法

输入：Web程序源码

输出：Web应用模型

```
1. S,P,Se,δ,σ=[],s0=null,ssource=null
2. /*初始化变量*/
3. While INPUT do
4.     if has routerfile then
5.      /*如果有路由文件获得URL到文件的映射*/
6.       Read routerfile
7.       /*读取路由文件*/
8.       Construct Relational Mapping
9.      /*构建关系映射*/
10.    end if
11.    if para in page then
12.      P=ExtractParas(page)
13.      /*提取参数*/
14.    end if
15.    if GetSessionFunc(page) not null then
16.      /*查找页面内是否有与访问控制有关的函数*/
17.      F=GetSessionFunc(page)
18.      P=ExtractRoles(page)
19.      Traversal(S,P,Se,σ,δ,F)
20.      /*实现逻辑见算法2*/
21.    end
22. end while
```

算法2 遍历函数算法

输入：页面信息

输出：状态集合

```
1. Foreach f in F do
2.     Range=GetfuncRange(f)
3.     /*确定访问控制函数约束的范围*/
4.     Foreach r in Range do
5.          Se=Constraint Function with role
6.          /*建立应用功能与角色的约束关系*/
7.            S0= CreateNewState()
8.            Add s0 to S
```

```
9.          δs=CreateTransition(ssource,s0,Se,P,page)
10.       /*生成页面应用功能与约束的映射关系*/
11.       Add δs to δ
12.       /*添加到集合*/
13.       if δs already in δ but Se not equal
14.       then
15.       /*如果映射关系已经存在但约束不同*/
16.       Mark(δs)/*就做标记*/
17.       AddSe toδs[Se] /*添加约束到列表*/
18.       ssource=s0
19.     end
20. end
```

算法 3 Web 实际行为模型建立算法

```
输入:HTTP 请求与响应
输出:Web 应用模型
1. S,P,Se,δ,σ=[],s0=null,ssource=null
2. While INPUT do
3.    Read data
4.    if request then /*处理 HTTP 请求*/
5.       Add request to σ
6.         If para in request then
7.           P=ExtractParas(request)
8.         end if
9.    else
10.       /*处理 HTTP 响应*/
11.       Se= GetSession()
12.       /*获得 Session 状态*/
13.       if page has not been visited before
14.       then
15.         s0= CreatenewState()
16.         /*如果页面之前没有被访问过*/
17.         Add s0 to S
18.         /*就创建一个新状态*/
19.         List=[ssource,s0,Se,P,request]
20.         δs=CreateTransition(List)
21.         /*生成请求响应的映射*/
22.         Add δs to σ
23.       else
24.         s0=GetState(S)
25.         /*如果之前访问过,就直接获得存储过的状态*/
26.           δs=CreateTransition()
27.           /*建立映射*/
28.           if δs already in δ but Se not
29.           equal then
30.           /*如果该映射已经存在但 Session 不同*/
31.             Add Se to δs[Se]
32.             /*则将该 Se 添加到当前映射的 Se 集合中*/
33.           end if
34.         end if
35.    end if
36.    ssource=s0/*更新 Ssource*/
37. end while
```

对逻辑类漏洞的识别都可以通过建立两个 FSM 来实现，然后对它们进行差异性分析来判断其是否存在漏洞。这两个 FSM 分别是理想 FSM(记为 F_{ideal})与实际 FSM(记为 F_{impl})。F_{ideal} 是网络应用程序的预期(或期望)行为，假设没有任何安全漏洞，F_{impl} 是网络应用程序的实际行为，如果 $F_{ideal}=F_{impl}$，则说明该 Web 应用是安全的；如果 F_{ideal} 与 F_{impl} 存在差异，并且该差异涉及访问控制，则说明该 Web 应用存在访问控制漏洞。

结合第 2 节中提出的示例应用，通过提出的 FSM 模型进行分析，该示例应用程序有 3 种状态，即用户未登录(S_0)、普通用户登录(S_1)和管理员用户登录(S_2)，每个输入 $I\in\sigma$ 都是 Web 请求的抽象表达，将访问应用页面时发送的 HTTP 请求简化为[请求方式，请求页面，请求参数，Cookie，Session]结构的数组，输出 $O\in\Lambda$ 是应用程序返回给用户的 Web 响应的抽象表达，简化为请求结果展示。

通过算法 3 对第 2 节中的示例程序进行建模，得到的是应用程序的实际 FSM，如图 4 所示。

而该应用的理想 FSM(F_{ideal})应是如下情况。

首先对状态示例的 3 种状态(用户未登录(S_0)、普通用户登录(S_1)和管理员用户登录(S_2))添加约束，在 S_0 与 S_1 状态下，只能对自身状态产生影响，而 S_3 因具备高级权限，可以向下管理。在此基础上，通过 index.php 可知，add_user 与 delete_user 都是在验证角色后才能执行的函数，而在 add_user.php 中却没有经过验证便可以执行 add_user，此时两种不同的授权使得在处理理想 FSM 时，按照所需权限的最高要求进行访问控制，此时 I_2 应获得 O_3 作为响应而不是正常返回。而在 edit_user.php 中，受角色约束，普通用户不应对除其自身之外的其他用户有影响，因此 I_4 也应被访问控制拦截，得到返回 O_3。此时，通过观察两个 FSM 的差异性，即可发现两个访问控制漏洞。

一个 Web 应用程序通常使用会话变量来维护用户的会话状态。由于一个会话变量可以取无限多的值，因此导致了无限多的状态，首先需要确定每个会话变量的值域。对每个会话变量所收集的数值集进行检验，并将每个变量分为两种类型。1) 有界的，这意味着该变量的值是有界的，是一个有限的集合。例如，$ SESSION['privilege']只假设两个值，即 user 和 admin，表示两种类型的用户。有界的会话变量的值域由观察到的值的集合组成。2)无界的，这意味着变量可能采取的值的数量与样本的数量呈线性增长。例如，$ SESSION['userid']可以取与用户数量一样多的值。非边界会话变量的值域包括 null 和 notnull，其中 null 表示该变量不存在。

然后，通过使用会话变量的值域的笛卡尔积来构建应用状态，将其称为状态签名，并从跟踪中识别出应用状态的集合。一个状态签名的例子是[Privilege=user][userid=notnull]，它代表一个普通用户登录时的应用状态。在理论上，应用程序的状态空间可能是非常巨大的，这取决于会话变量的数量和它们的值域。但在现实中可以观察到，实际的状态数量要小得多，因为某些会话变量是相互关联的。例如，当一个用户登录时，不可能出现两个会话变量都被更新的状态。[privilege=user][userid=null]，这使得状态空间中的状态依能够进行枚举测试。

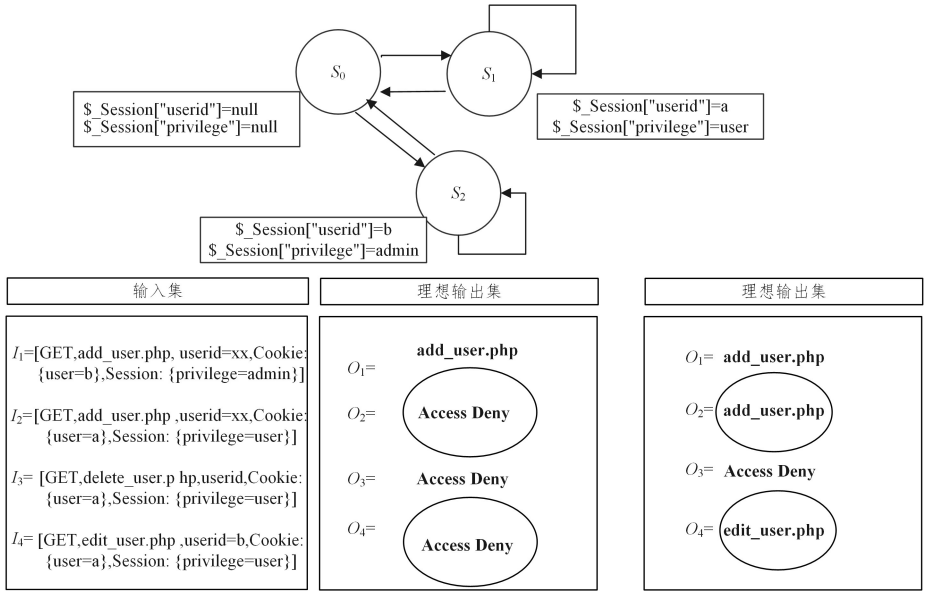


图 4 示例应用对应的实际 FSM 状态转换

Fig. 4 Implement FSM state transition of demo application

3.2 静态分析阶段

(1)路由解析

目前的 Web 应用程序多采用 MVC 架构,这意味着仅从文件结构分析,通过构造文件所在位置 URL 访问相应功能是不再可行的,此模块通过对路由文件及代码中映射函数的解析,来建立 URL 到文件的映射关系,为后续生成网站地图作前置工作。

(2)参数提取

当执行黑盒测试时,由于不知道存在什么参数,因此需做大量模糊测试工作,故而对参数进行猜测,但在白盒测试中,可以通过指定参数来缩减测试流程,且访问页面中的功能通常都是需要传入参数来执行指定方法,通过爬虫在前端并不能获得所有功能方法,通过提取参数与调用关键字可以遍历应用功能,从而对爬虫所不能发现的功能函数进行访问控制测试。

(3)预期访问提取

通过对代码进行静态分析,分析代码中所包含的权限检查及会话管理方法,结合交叉引用信息,提取出各个页面与方法所应该遵循的访问控制策略,且分析是否存在方法在不同引用位置存在不同的访问控制策略,这种差异性需在动态分析中进行优先检查。

(4)站点地图生成

通过由路由解析获得的 URL 到文件的映射关系、参数提取中获得的参数到方法的映射关系、预期访问提取中获得的文件与方法的预期访问控制策略,可以生成站点地图,为 FSM 模型填充细节,并为爬虫在爬取时省去了大量模糊测试过程。

3.3 动态分析阶段

(1)爬虫

爬虫作为动态分析中最重要的模块,通过给定的 URL 及多个用户登录凭证,分别在不同登录状态下获取访问各个页面的响应,并将其记录到数据库中,以供决策模块寻找漏洞。采用的爬虫为 Crawlergo[18],该爬虫为主动式爬虫,能够对整个网页的关键位置与 DOM 渲染阶段进行 HOOK,自动进行表单填充并提交,配合智能的 JavaScript 事件触发,尽可能地收集网站暴露出的入口,捕获的 URL 被存储在一个列表中,爬虫开始以深度搜索的方式探索应用程序的网页,直到所有的网页被访问。Crawlergo 将网页特征向量抽离出来,索引存储,用于判断大量网页的相似度,以实现对页面的去重。再结合静态扫描得到的站点地图,有针对性地对功能较复杂的地方进行指向性测试。将其与服务端交互的 HTTP 请求与响应记录下来,用于后续提取会话信息。

(2)代理

即便通过静态分析生成了站点地图,但爬虫还是不能对所有的页面及其 API 进行遍历,还是会存在遗漏,此时可以通过中间代理采用人工的方式执行正常交互,中间代理获取 HTTP 的请求与响应,用于后续会话信息提取。这种形式也常被用在各种 Web 渗透测试工具中,用于实现对网络请求的重放与参数测试。在经决策模块验证后,将可能存在问题的 HTTP 请求通过代理进行重放,对漏洞进行验证。

(3)会话提取

会话可以通过两种方式来维护:纯粹在客户端通过 cookie 的方式以及服务器和客户端结合的方式。在后一种方法中,一个包含唯一的会话标识符(即会话 ID)的 cookie 被维护在客户端。对于每个请求,这个 cookie 被发送到服务器,然后服务器使用会话 ID 与存储在服务器上的会话信息相匹配。会话 ID 被用来提取实际的会话变量名称和它的值,然后与 URL 进行映射。PHP 应用程序的会话信息提取过程描述如下:会话信息被存储在服务器端,并以会话 ID 为索引;会话 ID 通常与 HTTP 响应头一起传递;在 Linux 服务器中,会话文件通常存储在/tmp/目录下,会话信息被序列化后并以变量名值对的形式存储在会话文件中。

3.4 决策阶段-状态偏离检测

在爬虫进行爬取时,对其输入多种登录状态的用户登录

凭证，这样可以获得多个状态访问同一页面的不同响应，以识别出以下类型的网页：1）只有登录后才能访问的页面；2）只能由某些角色访问的网页；3）用户定制的页面，其他相同角色用户也不能访问。

在通过 HTTP 请求与响应构建 Web 应用的 FSM 模型后，通过分析差异性，识别出以下访问控制漏洞类型。

（1）认证绕过。这种攻击的重点是识别需要由特权用户访问的网页，但未能在源代码中施加访问检查。这样的网页被作为第三方/非特权用户访问，提交网页的 HTTP 请求，而不提交有效的凭证。换句话说，一个应用程序中的网页被任何用户访问而无需登录，这就是所谓的认证绕过攻击。这种漏洞在两种 FSM 中具备完全相同的行为，但它又是存在漏洞的，因为在代码中需要约束条件才能识别出存在访问控制的页面，但可以通过在那些不存在会话检测及访问控制的页面中查询调用关系来建立起与登录状态的关联，从而分析出是否存在认证绕过。

（2）垂直越权。高权限的网页是通过对网页提出 HTTP 请求，在参数中加入有效的用户名，但没有为权限设置适当的会话变量来进行访问，后期需首先确定特权页面和可以访问相应页面的角色。为了实现这一目标，需要识别 FSM 中通往同一目标状态的转换以及能够访问该状态的角色集。可以从会话变量中提取进入该状态的角色，并将其存储在可访问的角色列表中。将列表从用于应用程序的全部角色集合中排除，就得到了该状态的不应访问角色集合。列表被更新为目标状态的 URL 和不能访问该状态的角色集，不断重复这个过程，直到应用中的所有特权页面都被识别出来。识别出特权网页后，对那些有特权的网页和非特权角色提交攻击请求，将这些攻击请求得到的响应与正常执行时得到的响应进行比较，从而分析出是否存在垂直越权漏洞。

（3）水平越权。属于用户定制的页面，因只进行了角色校验而使得其能够被相同角色的其他用户访问。第一步是确定任何参数和会话变量具有相同值的过渡函数，第二步是对这些过渡函数的目标状态提出请求，参数的值与会话变量的值不同。将这些攻击请求得到的响应与正常执行期间得到的响应进行比较，从而确定是否存在水平越权漏洞。

将以上分析逻辑作为主要决策策略，识别出逻辑漏洞并生成漏洞报告，同时将疑似存在漏洞的 HTTP 请求记录到动态分析模块的代理中，以备复现分析，确定漏洞的真实性。

4 测试与评估

本节采用真实软件进行测试，以评估方法的可行性，同时选择同类研究成果 DetLogic[8] 和 SDACV[6] 所采用的方法进行性能对比测试。

4.1 实验设置

为了测试方法的可行性及原型工具 ACVD 对访问控制漏洞发掘的性能，因相关研究未开源代码，为做对照实验，本文在环境配置为 Intel（R） Core（TM） i7-10710U CPU @ 1.10 GHz 1.61 GHz，16 GB memory 的个人电脑中，部署 5 套开源 Web 应用程序进行实验，5 套开源 Web 应用均是相关研究中进行分析的 Web 应用程序。选取的应用程序如表 1 所列。

表 1 实验所采用的开源项目

Table 1 Open-source projects for experiment

id	Project name	Files	Lines of Code
1	SCARF	25	1913
2	Wackopicko	52	4044
3	Bloggit	24	3132
4	OpenIT	25	26150
5	Events Lister	37	2076

4.2 有效性

使用原型工具 ACVD 对选取的 5 个应用程序进行访问控制漏洞检测，结果如表 2 所列。本文综合对比了静态分析、纯黑盒测试与本文中的原型工具的性能差距，而两个原型工具在选取 Web 应用中存在差异，又因代码未开源，无法获得原型工具在论文未涉及项目中的表现，因此暂留空以供参考。

实验结果如表 2 所列，在对选取的应用程序进行性能测试时，发现原型工具 ACVD 在对访问控制漏洞的检测中，对选取应用程序中的能够发现的访问控制漏洞基本实现了全覆盖，能够对访问控制漏洞实现高查全，经人工复现审计后发现存在部分误报。与 DetLogic 实验数据进行对比可知，ACVD 对访问控制漏洞的检测有效率高于 DetLogic，且明显优于静态分析的检测结果。

表 2 漏洞检测结果及对比

Table 2 Vulnerability detection results and their comparison

Project name	Number of request	Number of response	ACVD found Vulns	DetLogic found Vulns	SDACV Found Vulns	Number of Real Vulns
SCARF	1348	1333	8	9	1	7
Wackopicko	2401	1539	2	3	Null	1
Bloggit	2710	2678	3	Null	Null	0
OpenIT	1433	1401	3	2	Null	2
Events Lister	1321	1082	0	Null	2	0

4.3 性能

本文提出的原型工具 ACVD 对访问控制漏洞的检测效率应是明显高于黑盒测试工具 DetLogic 的，但由于 DetLogic 没有开源，无法做到真实对比，但 ACVD 中的动态分析工具在不借助静态分析的结果时，也是一个独立模块，可做覆盖率性能测试，其性能应趋于 DetLogic，在此使用纯黑盒动态分析，与 ACVD 的使用进行对比。

因 ACVD 首先使用静态分析方法获得路径信息，所以其可知总的路径数，能够使覆盖率接近 100%，而使用纯黑盒测试时，难以发现所有路径，仅会在设定时间内不再发现新路径时才判断检测完毕。通过图 4 可知，ACVD 在效率上远超使用纯黑盒方法的 DetLogic，效率提升了约 3 倍。本实验出于作对照实验设计的目的，所选 Web 应用程序均为 PHP 语言编写的 Web 应用程序。在原型工具设计时，已考虑将 ACVD 做

到了对代码的最小利用，可以通过扩展参数匹配规则及路由转换规则来增加对其他语言架构的 Web 应用程序的适用性。

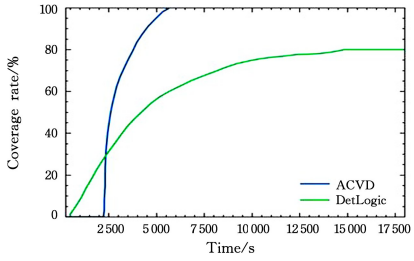


图 5 ACVD 与 DetLogic 的效率对比

Fig. 5 Efficiency comparison of ACVD and DetLogic

结束语 本文提出了一种针对访问控制漏洞的状态偏离检测算法，设计并实现了原型系统 ACVD。该原型工具采用白盒静态分析加动态分析结合的方法，通过静态分析提取出页面中的参数及访问控制函数存在情况，并生成站点地图对动态分析路径进行剪枝，从而提高动态分析效率，动态分析 Web 应用的请求与响应。然后，通过有限状态机(FSM)对 Web 应用程序的逻辑进行建模，将访问控制漏洞形式化为预期 FSM 与实施 FSM 之间的差异，进而实现对漏洞的发现。最后设计实验，对原型工具 ACVD 自动化漏洞检测的有效性进行了验证，并与同类工具 DetLogic 进行了效率比对，进一步验证了所提方法的先进性。

即便 ACVD 为了摆脱跨语言跨框架给白盒方法[19]带来的约束性尽量做到对代码的最小利用，但其分析仍要对不同语言进行适配。逻辑漏洞不是存在于某个函数，而是存在于代码功能。因此，在对逻辑漏洞的通用识别方法研究中，仍需进一步采用白盒方法，基于不局限于语言、架构的中间表达形式，提高对应用逻辑的检测，实现对逻辑漏洞的挖掘。在未来的工作中，应进一步提高中间语言对应用代码逻辑表达的准确程度，以此实现对逻辑漏洞的更全面的挖掘工作。

参 考 文 献

[1] KULENOVIC M, DONKO D. A survey of static code analysis methods for security vulnerabilities detection[C]// International Convention on Information and Communication Technology, Electronics and Microelectronics. 2014: 1381-1386.

[2] YAMAGUCHI F, GOLDE N, ARP D, et al. Modeling and discovering vulnerabilities with code property graphs[C]// 2014 IEEE Symposium on Security and Privacy. IEEE, 2014: 590-604.

[3] 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.

[4] GAO R, ZHOU C L, ZHU R. Research on vulnerability mining technology of network application program [J]. Modern Electronics Technique, 2018, 41(3): 115-119.

[5] The OWASP Top 10 2021. [OL]. https://owasp.org/Top10/.

[6] SUN F, XU L, SU Z. Static Detection of Access Control Vulnerabilities in Web Applications[C]// USENIX Security Symposium. 2011.

[7] MA L, YAN Y, XIE H. A new approach for detecting access control vulnerabilities[C]// 2019 7th International Conference on Information, Communication and Networks (ICICN). IEEE, 2019: 109-113.

[8] 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.

[9] LI X, SI X, XUE Y. Automated black-bo x detection of access control vulnerabilities in web applications[C]// Proceedings of the 4th ACM Conference on Data and Application Security and Privacy. 2014: 49-60.

[10] LI X, XUE Y. LogicScope: Automatic discovery of logic vulnerabilities within webapplications[C]// Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. 2013: 481-486.

[11] FELMETSGER V, CAVEDON L, KRUEGEL C, et al. Toward automated detection of logic vulnerabilities in web applications [C]// USENIX Security Symposium. 2010.

[12] Acunetix Vulnerability Scanner 2021[OL]. https://www.acunetix.com/vulnerability-scanner/.

[13] HCLAppScan[OL]. https://www.hcltechsw.com/appscan.

[14] Fotify2021[OL]. https://www.microfocus.com/enus/cyberres/application-security.

[15] Coverity. 2021[OL]. https://scan.coverity.com/.

[16] LI S H, SUN Q H, ZHAO M Y. A machine learning-based approach to detecting overrun vulnerabilities[J]. China Security Protection Technology and Application, 2021(2): 67-72.

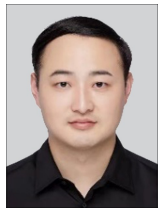
[17] JIANG H T, GUO Y J, CHEN H, et al. State-machine based vulnerability detection method for mobile application overridden access[J]. Journal of Nanjing University of Science and Technology, 2017, 41(4): 434-441.

[18] Qianlitp. 2019. Crawlergo. A powerful browser crawler for web vulnerability scanners [OL]. https://github.com/Qianlitp/crawlergo.

[19] LI M L, LU Y L, HUANG H, et al. Guided Grey-Box Fuzzing Test Method Combining Distance and Weight[J]. Computer Engineering, 2021, 47(3): 147-154.

[20] ZHANG J, JING W, CHEN F. Vulnerability detection of instant messaging network protocol based on passive clustering algorithm[J]. Journal of Jilin University (Engineering and Technology Edition), 2021, 51(6): 2253-2258.

MA Qican, born in 1997, postgraduate. His main research interests include automated code audit and IOT device security.



WU Zehui, born in 1988, Ph. D. His main research interests include software security analysis and vulnerability analysis of cloud platform.

(责任编辑：喻藜)