DOI:10.3969/j.issn.1671-0673.2022.05.011

基于 FP-Growth 算法的 XSS 漏洞动态检测方法

肖斯文,张 斌

(信息工程大学,河南 郑州 450001)

摘要:为提高跨站脚本(XSS)漏洞动态检测方法的检测能力,提出一种基于 FP-Growth 算法构造攻击向量库并利用 web 测试工具进行检测的 XSS 漏洞动态检测方法。首先,建立基于 FP-Growth 算法的关联规则挖掘模型,利用模型挖掘大量 XSS 攻击实例间的关联规则,并以此为基础构造一个内含更多种类攻击向量的攻击向量库;其次,设计注入点匹配规则和攻击向量验证规则并实现基于 Selenium 的 XSS 漏洞检测模块 SmartXSS,以适应加载攻击向量库实施检测。实验结果表明,基于实验环境所提方法通过发送正常请求实施 XSS 漏洞检测的检测率可达 88.89%,相比 Wapiti 等同类工具提高了 22.22 个百分点。

关键词:FP-Growth 算法;XSS 漏洞;动态检测;关联规则挖掘

中图分类号:TP393 **文献标识码**:A **文章编号**:1671-0673(2022)05-0586-08

XSS Vulnerability Dynamic Detection Method Based on FP-Growth Algorithm

XIAO Siwen, ZHANG Bin

(Information Engineering University, Zhengzhou 450001, China)

Abstract: To improve the detection ability of cross site scripting (XSS) vulnerability dynamic detection method, a dynamic XSS vulnerability detection method based on FP-Growth algorithm is proposed to construct attack vector library and web testing tools to detect XSS vulnerability. Firstly, we establish an association rule mining model based on FP-Growth algorithm to discover association rules among attack instances and construct an attack vector library containing more kinds of attack vectors. Secondly, the injection matching rules and the attack vector verification rules are used to match the attack vector library for detection. Finally, an XSS vulnerability detection module named SmartXSS based on Selenium is proposed. According to the performance of the test under laboratory conditions, the detection rate of XSS vulnerability detection is 88.89% by sending the normal requests, which is 22.22 percentage point higher than Wapiti and other similar tools.

Key words: FP-Growth algorithm; XSS vulnerability; dynamic detection; association rules mining

0 引言

跨站脚本(Cross Site Scripting,XSS)攻击是网络中常见的一种攻击,XSS 漏洞在安全监测平台扫描的漏洞中占比高达 37.3%[1]。通过 XSS 可以劫持用户会话,盗取用户敏感信息,实施网页挂马等恶意操作,极大危害网络安全,因此对 Web 应用进行 XSS 漏洞检测十分重要。

XSS 漏洞检测方法按分析机制主要可分为静态分析方法和动态分析方法。静态分析方法一般采用污染传播模型,通过对程序的源代码进行分

收稿日期:2021-11-24;修回日期:2021-12-21

作者简介:肖斯文(1993-),男,硕士生,主要研究方向为 XSS 漏洞检测。

析,查看输入的数据是否经过安全处理。静态分析效率较高,但依赖 Web 应用源码和内部信息,检测效果受安全处理规则的影响较大,存在一定的误报和漏报现象[1-2];而动态分析则不依赖 Web 应用源码,采用模拟真实攻击的方式进行检测,通过向 Web 应用注入攻击向量,依据站点的响应信息分析相应检测结果。这类检测方法避免了误报,但其漏洞检测率依赖于所使用的攻击向量库[3]。目前,对构造和优化 XSS 攻击向量库的研究主要分为两类:一类是基于已有知识和经验的构造方法[4-6]。根据 XSS 攻击原理以及已知的 XSS 攻击向量组成规律,定义攻击向量生成式和变异式并根据其生成攻击向量。这类方法能够产生出较为有效的通用攻击向量库,但依赖由人工定义的攻击向量规则,检测应用较新防御策略的 Web 应用时适应性不强[7];另一类是基于机器学习的构造和优化方法。根据 XSS 攻击向量的实际攻击结果进行评估,利用机器学习算法进行学习并反作用于攻击向量的生成和优化[8-9]。这类方法对于变异和筛选攻击向量表现较好,但易造成攻击向量向已检测页面聚集,进而导致攻击向量库对其他类型页面的适配性不足[10]。

为提升检测中 XSS 攻击向量库的适应性,提出一种基于 FP-Growth 算法的 XSS 攻击向量库构造方法。首先,基于 FP-Growth 算法建立 XSS 攻击实例关联规则挖掘模型,然后应用模型分析 XSS 攻击实例事务数据库进而挖掘攻击实例各元素之间的关联规则,获取生成攻击向量的依据;其次,结合现有知识经验生成初始攻击向量并制定用于绕过防御措施的变异规则,并应用变异规则生成种类更多的 XSS 攻击向量。为适应攻击向量库进行注入检测,制定注入点匹配规则和攻击向量验证规则,增强针对 XSS 漏洞的检测能力;最后,根据规则要求设计并实现基于 Selenium 的 XSS 检测模块 SmartXSS。为验证所提方法的有效性,使用 SmartXSS 以及 Wapiti 等同类工具分别对安全测试平台 PentesterLab 进行检测,实验结果表明,基于实验环境 SmartXSS 较 Wapiti 等工具在漏洞检测率方面有较大提升。

1　基于 FP-Growth 算法的 XSS 攻击向量库构造方法

FP-Growth 算法[11]是一种关联分析算法,利用 FP-Growth 算法能够发现 XSS 攻击实例各元素之间的联系,进而为攻击向量生成和变异提供依据。首先基于 FP-Growth 建立 XSS 攻击实例关联规则挖掘模型并通过模型对 XSS 攻击实例事务集进行分析,进而得出包含构造 XSS 攻击向量元素的关联关系;其次,由于 XSS 攻击向量具有一定的结构性,因此需要对关联关系进行分析并按照 HTML 语法将互相关联的元素置于相应的位置组成攻击向量,然后结合 XSS 备忘单[12]定义相关变异规则,生成可以绕过一些防御措施的攻击向量并组成攻击向量库。

1.1　XSS 攻击实例关联规则挖掘模型

XSS 攻击实例关联规则挖掘的目的是获取 XSS 攻击实例中各元素的频繁项集。通过分析频繁项集发现构造 XSS 攻击向量组成元素之间蕴含的关联关系。关联规则挖掘模型如图 1 所示。

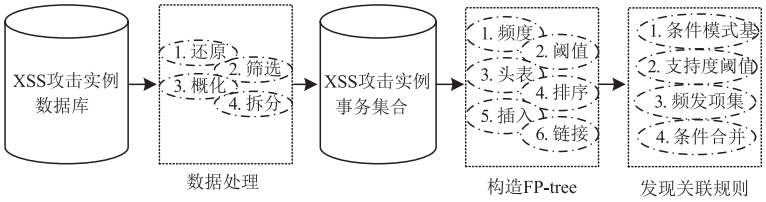


图 1　XSS 攻击实例关联规则挖掘模型

XSS 攻击实例数据库。通过获取 XSSed 等记录 XSS 真实攻击的网站数据以及收集一些 XSS 攻击日志,建立一个 XSS 攻击实例数据库,其中每条记录表现为一条包含攻击载荷的 URL 链接。

数据处理。通过解码还原、筛选载荷、概化处理、拆分元素以及统计频度等措施将 XSS 攻击实例库中的记录转换为一组元素的集合。

XSS 攻击实例事务集合。通过将数据处理后的元素集合进行合并统计后构成事务集合,每一项事务表现为一个元素集合以及其出现的频度。

构造 FP-tree。首先,通过遍历事务集合,统计元素频度并以此为依据确定最小频度阈值;其次,按元素频度从高至低建立项头表;再次,遍历数据集合,按项头表顺序将事务中的元素排序,从根节

点下将元素逐项插入、合并计频并从项头表出发将所有路径上的相同节点链接起来。

挖掘关联规则。根据FP-tree生成每一项元素的条件模式基并确定支持度阈值,通过条件模式基以及支持度阈值逐步查找频繁1项集至频繁N项集,查找并去除所有的闭频繁项集即为模型输出的关联规则。

1.1.1 数据处理

XSS攻击实例数据库中的记录是连续的字符串,而FP-Growth算法支持的输入是一组项的集合,因此要进行解码还原、筛选载荷、概化处理和拆分处理将原始记录处理为FP-Growth算法支持的形式。处理过程实例如表1所示。

解码还原。记录中通常包含一些编码,如URL编码、HTML编码、进制编码等,可读性较差不利于数据挖掘,因此需首先通过解码操作将记录还原为可读的字符串。

筛选载荷。为避免无关信息影响结果,采用正则表达式匹配并删除网站域名和参数等与攻击载荷无关的信息,使用的正则表达式如下:

$$http([s]?)://((\w+-?\w+\.)|(\w+\.))+\w+\/?(\w+|((\w+-)*\w+)|((\w+\.)*\w+)\/)*?(\w+|(\w+-)*\w+)*(\.\w+)*\??((\w+|(\w+-)*\w+)=)?((\w+|(\w+-)*\w+)?&(\w+|(\w+-)*\w+)=)* \quad (1)$$

式(1)表示匹配以http或https协议开头以“=”结束的多种类型的路径和参数。

概化处理。载荷内对攻击功能无影响的元素以及大小写混淆字符的存在,一是会带来无意义的计算消耗,二是会对频繁项集的支持度产生影响,如“scRipt”如果单列为一项,会降低“script”的支持度。因此将所有字符统一为小写并使用正则表达式匹配载荷内与攻击功能无关的元素并统一替换字符为字符“XSS”。此处使用的正则表达式为

$$alert(.*?)|prompt(.*?)|confirm(.*?) \quad (2)$$

式(2)表示以非贪婪的模式匹配“alert()”、“prompt()”或“confirm()”中的字符。

拆分处理。采用正则表达式匹配非单词字符并利用re库的split()函数进行拆分,将字符串分离为单个元素后去重构成元素集合。

表1 处理过程实例

操作类别	数据
原始记录	http://sv.888.com/new888/home.htm?page=gettingstared&lang=%22%3E%3CscRipt%3Ealert(document.cookie)%3C/script%3E
解码还原	http://sv.888.com/new888/home.htm?page=gettingstared&lang="><scRipt>alert(document.cookie)</script>
筛选载荷	"><scRipt>alert(document.cookie)</script>
概化处理	"><script>alert(“XSS”)</script>
拆分处理	(script,alert)

1.1.2 构造FP-tree

FP-tree是一种紧密的数据结构,由事务集合的各项事务中满足最小支持度的项映射而成,包含全部频繁项集的信息。构造FP-tree需要遍历两次事务集合,先后完成建立项头表和写入事务。

建立项头表。首先,遍历XSS攻击实例事务集合统计元素项和频度,得到一个键为元素、值为频度的字典。通过观察与分析字典,发现频度较低且与XSS攻击向量构造不相关元素的频度分布范围,如一些数据处理后遗留的随机数字、字母组合等,据此设定最小支持度ζ并只保留频度不低于ζ的元素;其次,对其余的元素进行逐个分析,去除与XSS攻击向量构造关联不大或是后续可以补全的元素以降低查找空间,包括某些标签和属性值,如“h1”、“position”等,以及某些并无实际意义但频度很高的元素,如空格、数字以及一些单词等;最后将其余元素按照频度降序排列创建项头表,项头表中的每一行包含元素名称以及其头部链接。

写入事务。再次遍历事务集合,首先去除不在项头表中的元素,其次将事务集合中每项事务的元素按照项头表的顺序进行排序以便于子树的查找,最后建立根节点“Null”并逐条插入事务。

图2以写入3条事务为例,展示FP-tree的构造过程。写入第1条事务时FP-tree中仅含有“Null”节点,因此新增节点并依次写入元素并在节点中添加频度计数器。当写入第2条事务时,与“Null”节点直接链接的节点未含有“src”元素,因此新建该节点并在该节点后继续插入第3条事务的其余元素。当写入第3条事务时,由于在树中已存在此序列,则不再生成新节点,而是与旧节点进行合并,并将计数器上的值进行更新。在创建树的同时,项头表元素后的指针开始移动,依次链接到树中每个相同的元素,图2中表示了在写入第3项事务后FP-tree中链接的情况,写入所有事务后同

步生成从项目头表开始的节点链表。

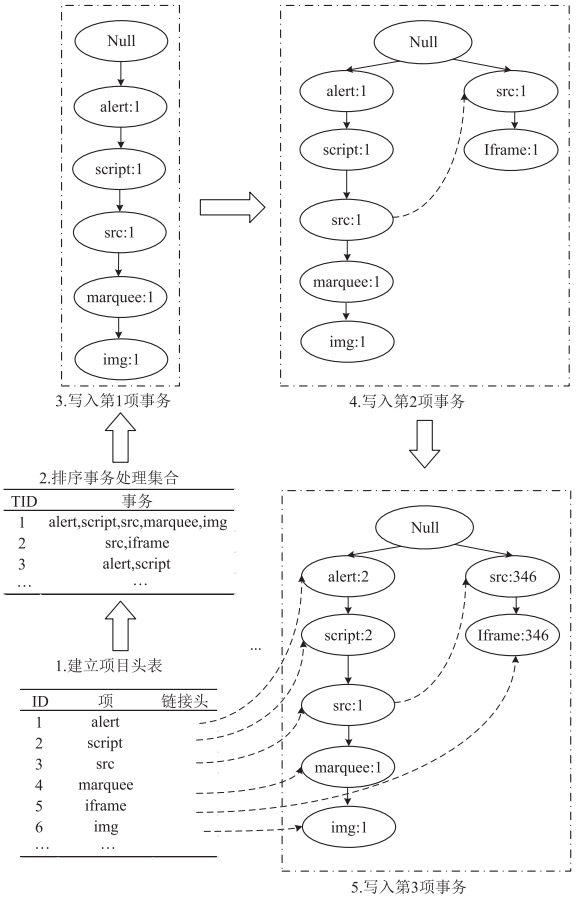


图 2　FP-tree 构造过程

1.1.3　发现关联规则

关联规则体现为符合一定条件的频繁项集,挖掘关联规则首先应找到所有的频繁项集。在 FP-Growth 算法中,挖掘频繁项集主要通过搜索 FP-tree 实现。

FP-tree 性质:对任意的频繁项 ai,所有包含 ai 的频繁项集都可以从 FP-tree 项目头表开始的节点链表获得。

根据这一性质从项头表底部开始依次查找以 ai 为后缀的全部路径并从中获取频繁项集,其中以 ai 为后缀的全部路径称为 ai 的条件模式基。下面以挖掘含有“window”元素的频繁项集为例阐述其过程。

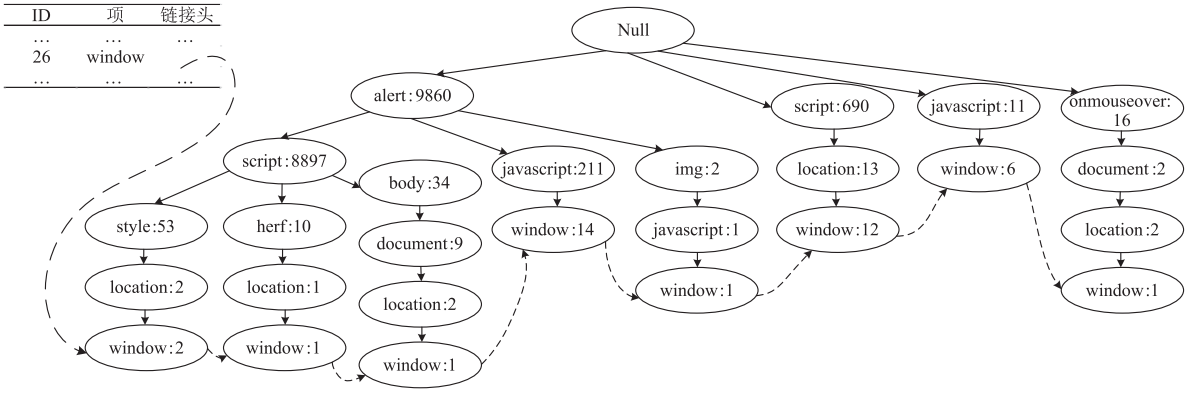
如图 3 所示,首先从项目头表“window”项的链接头出发,找出 FP-tree 中以“window”项为后缀的所有前缀元素,得到图 3(a)中的结果;然后根据图 3(a)可以发现以“window”为后缀的所有路径,构成如图 3(b)所示的“window”的条件模式基,并根据其生成“window”的条件模式子树,如图 3(c)所示。此处根据频繁项集的总体分布情况设定最小支持度 ζ’ 为 4,因此在生成条件模式子树时去除小于 ζ’ 的“herf”、“body”等元素。通过图 3(c)可以得出包含“window”的频繁 2 项集为(window,javascript)、(window,alert)、(window,location)、(window,script)。

随后对图 3(c)进行递归查询直至到达“Null”节点,依次可以得到“window,script”、“window,location”以及“window,location,script”的条件模式基以及对应的条件模式子树,如图 3(d)~图 3(f)所示,从而获得“window”的频繁 3 项集(window,location,alert)、(window,document,location)、(window,script,location)、(window,script,alert)、(window,alert,javascript)以及“window”的频繁 4 项集(window,script,location,alert)。此时,注意到频繁项集中存在闭频繁项集(window,script,alert),可以被(window,script,location,alert)完全覆盖,因此去除闭频项集。与此类似,可以获得头部字典所有元素的频繁项集,去除闭频繁项集后即为发现的关联规则。

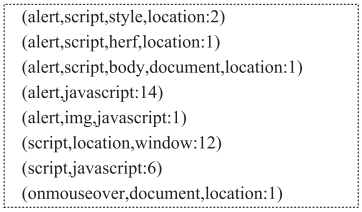
1.2　攻击向量生成

由于 XSS 攻击向量遵从一定的结构和顺序,因此首先需要对关联规则进行逐个分析,将各项置于符合 HTML 语法和脚本语言语法的位置并补全缺少的元素,构成基本攻击向量生成规则;其次,为绕过目前 Web 应用中常用的防御策略以有效查找 XSS 漏洞,需要制定相应的变异规则以生成变异攻击向量;同时,为避免出现错误应用变异规则导致攻击向量失效的情况,制定应用变异规则的约束规则以确定变异规则的应用范围;最后,通过组合使用 3 种规则生成最终的攻击向量。

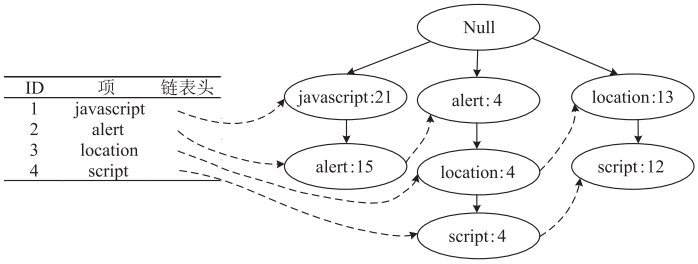
在制定基本攻击向量规则时,为避免生成失效和重复的攻击向量,去除目前各类型浏览器支持较少的 XSS 利用方式,如“style”标签中的“background”属性、“img”标签中的“src”属性;合并一些类似属性,如可以执行 js 代码的“on”事件,归纳出如表 2 所示的涵盖基础攻击向量(Basic_Payload)、HTML 标签中的攻击向量(HTML_Payload)以及闭合前项标签的攻击向量(Tag_close_Payload) 3 种类型基本攻击向量的生成规则。根据关联规则以及数据处理时掌握的变形方式结合 XSS 备忘单记录的变异样式,制定的变异规则如表 3 所示。根据 HTML 语言与脚本语言的差异,制定应用变异规则的约束规则如表 4 所示。组合使用 3 类规则生成 XSS 攻击向量的方式与 XSS Validator 插件中 XSS 攻击向量的构造方式相比,主要增加了链接、输入、按钮以及远程调用的恶意脚本等形式的攻击向量。



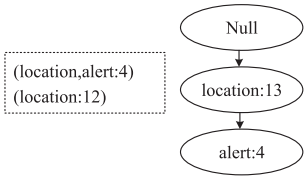
(a) 以“window”为后缀的部分FP-tree



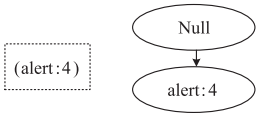
（b） “window”的条件模式基



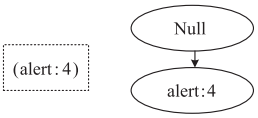
(c) “window”的条件模式子树



(d) “window,script”的条件模式基及条件模式子树



(e) “window,location”的条件模式基及条件模式子树



(f) “window,script,location”的条件模式基及条件模式子树

图 3 以“window”为后缀的频繁项集挖掘过程

表 2 基本攻击向量生成规则

类别	攻击向量生成规则
Basic_Payload	<script>JS_Payload</script><Tag> Basic_Payload</Tag><Tag Event = JS_Payload ></Tag><Tag Src = JS_Payload></Tag><Tag Src = URL_Payload></
HTML_Payload	Tag><Tag Src = Protocol_Script JS_Payload></Tag><Tag Src = Protocol_Script URL_Payload</Tag><Tag Src = Protocol_Fake Protocol_Fake_Field JS_Payload</Tag>
Tag_close_Payload	End_Tag Basic_Payload End_Tag HTML_Payload

表 2 中各组成部分的含义如下。

JS_Payload：脚本载荷，如“alert(“XSS”)”，“prompt(“XSS”)”，“confirm(“XSS”)”等。

Tag：HTML 标签，如“body”，“input”，“form”，“ iframe”等。

Event：HTML 事件，如“onload”，“onclick”，“onerror”等。

Src：链接属性，如“src”，“href”等。

Protocol_Script：脚本协议，如“javascript:”。

Protocol_Fake：伪协议，如“data:”等。

Protocol_Fake_Field：伪协议字段，如“text/java-script,”。

URL_Payload：远程脚本，如“http://xx/xss.js”。

End_Tag：闭合标签，用于闭合前向标签或属性，如“/>”、“;”等。

表 3 变异规则

变异规则	变异示例
R1：大小写变换	<script>alert(‘XSS’)</script> → <sCrIPt>alert(‘XSS’)</scrIpt>
R2：利用回车、空格、Tab	<script src="data:text/javascript, alert(1)"></script> → <script src="data:text/java script,alert(1)"> </script>

续表 3　变异规则

变异规则	变异示例
R3:添加注释	<script src="data:text/javascript, alert(1)"></script> → <script src="data:text/ja/**/vascript, alert(1)"></script>
R4:进制编码	<script>alert('XSS')</script> → <script>\u0061lert(1)</script>
R5:Hex 编码	XSS → XSS
R6:Base64 编码	<script src=data:text/javascript;alert(1)></script> → <script src=data:text/javascript;base64,YWxlcnQoMSk=></script>
R7:HTML 编码	XSS → XSS
R8:混合编码	<script src="data:text/javascript, alert(1)"></script> →<script src=data:text/javascript;base64, %59%57%78%6c%63%6e%51%6f%4d%53%6b%3d></script>
R9:eval 函数	<script>alert('XSS')</script> →<script>a=eval;b=alert;a(b(/xss/.source));</script>

表 4　约束规则

组成部分	变异规则
JS_Payload	R4,R6,R8,R9
Tag,Event	R1
Protocol_Script,Protocol_ Fake	R1,R2,R3,R4,R7
Protocol_Fake_Field	R2,R3
URL_Payload	R4,R5,R6,R8

2　基于 Selenium 的 XSS 动态检测模块设计

XSS 动态检测模块的功能是通过给定的待检页面,自动识别可注入点并向其加载 XSS 攻击向量,同时还能监测攻击向量的执行情况并返回最终的检测结果。

根据功能需求和 selenium 应用要求,所设计的 SmartXSS 由页面分析组件、攻击向量加载组件、漏洞检测组件以及日志记录组件构成。攻击向量组件负责将外部攻击向量传入工具中,包括加载本地的攻击向量文件以及搭建的服务器上的远程攻击向量文件;页面分析组件主要负责对待测页面的解析以及检索可注入点,检索范围包括利用 get 方式的参数、锚链接以及利用 post 方式的表单等;漏洞检测组件主要负责对注入点注入攻击向量并检测攻击向量的执行情况,其中包含操纵浏览器以及攻击向量验证等功能;日志记录组件主要负责对检测过程中关键结果的记录,包括经验证存在 XSS 漏洞的注入点以及发现该漏洞所使用的攻击向量。下面对页面分析组件中的注入点匹配功能以及漏洞检测组件中攻击向量验证功能加以分析。

注入点按出现位置可分为地址参数、表单项以及锚链接 3 类,但 XSS Validator 等工具仅对地址参数和表单项进行识别。SmartXSS 通过 urllib 库对 URL 地址解析,能够获取包含地址参数和锚链接在内的一组参数,随后通过识别名称的方式区分地址参数和锚链接。而对于表单项这一类注入点,SmartXSS 采用 mechanize 库进行识别。首先查找页面中的表单,随后对表单中各项的属性进行分析,找出存在可输入属性“TextControl”的表单项。

SmartXSS 主要通过两种方式实施攻击向量验证:一是对于注入后可自动执行载荷内脚本的攻击向量,由于所有的攻击向量实现了弹窗功能,可通过监测弹窗的方式进行验证;二是以链接、按钮以及表单等形式需要二次操作触发的攻击向量,由于均统一了名称,通过 XPath 定位并操控进而再由第一种方式验证,这也是 SmartXSS 较 XSS validator 等工具新增的验证方式。

Smarts 检测流程如图 4 所示,首先调用浏览器驱动创建浏览器实例,输入待测的 URL 地址,此时会查询该网址是否已被检测并返回结果,若未被检测则交由页面解析,查看网页中是否存在注入点,若存在则写入到待测内容。随后待测内容将注入点传入浏览器并对其进行定位,然后操控浏览器从已加载的攻击向量中选择一条注入,若待测内容为地址参数或锚链接,则构造出新的 Web 请求并提交;若待测内容为表单项,则调用浏览器驱动向表单项内填入攻击向量并提交。通过监测弹窗的方式监控页面响应,若载荷内的脚本执行发生弹窗,则记录 URL 和注入点,若未检测到弹窗,则重新选择攻击向量注入直至攻击向量库中的所有攻击向量都已注入,最后输出检测结果。

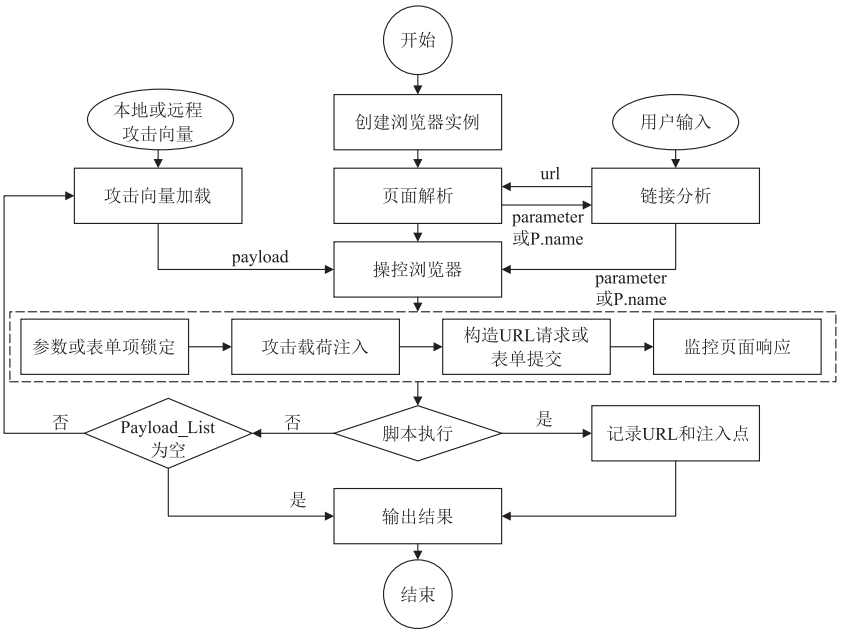


图4 SmartXSS检测流程

3 实验与分析

实验通过SmartXSS、XSS Validator与Wapiti等XSS漏洞检测工具分别对PentesterLab进行检测并用对比的方式来验证所提方法的优越性。

实验环境：硬件环境为Intel Core i7-9750H@2.60 GHz 6核处理器，32 GB内存；软件环境为Windows 10操作系统，搭载2台虚拟机，其中靶场虚拟机1台：Debian 6环境，分配2 GB内存，其上搭建1款渗透测试平台PentesterLab，内含9个XSS漏洞；检测虚拟机1台：kali环境，分配3 G内存，配备含有XSS validator 1.3.2的Burp Suite社区版v2021.8.3以及Wapiti v3.0.4。SmartXSS配置在本地PC机，使用Firefox驱动，浏览器版本为93.0。

实验数据：XSS攻击实例数据库来源于爬取XSSed网站上公布的攻击记录，包含在13 142条URL链接中。通过所提方法进行处理，构造了包含62项元素的FP-tree，发现了450条关联规则，通过分析关联规则制定了3类攻击向量生成规则并根据规则生成了228条攻击向量。

实验采用以下指标进行评价。

Detection_rate：检测率，检出漏洞数占所有漏洞数的比例，是评价XSS动态检测方法主要指标。

$$Detection\_rate=\frac{Detected\_number}{Total\_number}\times 100\% \quad (3)$$

式中，$Detected_num$为检测工具实际检测到的漏洞数，$Total_num$为Web应用存在的XSS漏洞总数。

Requests_mean：平均请求数，表示平均检出1个漏洞需发送的请求数，一定程度上反映检测方法的检测效率。

$$Requests\_mean=\frac{\sum_{i=1}^{Detected\_num} Requests\_num_i}{Detected\_num} \quad (4)$$

式中，$Requests_num_i$表示检测第i个漏洞时发送的所有请求数目。

漏洞测试结果如表5所示（“√”代表检出漏洞，“×”代表未检出漏洞，“O”代表无法检测）。同时为较为公平地分析平均请求数，仅对3种工具共同检出的5个漏洞的请求数进行分析，其漏洞检测请求数如表6所示。从两表中可以看出，SmartXSS的漏洞检测率最高，较其他两款工具高出22.22个百分点，但平均请求数较高。这主要是由于SmartXSS内置更多种类和数量的攻击向量，实现了适应这些攻击向量的检测和验证方式，同时相比其他工具包含了更多注入点匹配方式，能够应对更多类型的XSS漏洞和更多样的防御措施。

表5 PentesterLab测试结果

检测工具	漏洞1	漏洞2	漏洞3	漏洞4	漏洞5	漏洞6	漏洞7	漏洞8	漏洞9	Detection_rate
XSS Validator	√	√	√	×	√	√	√	×	O	66.67%
Wapiti	√	√	√	√	√	√	×	×	×	66.67%
Smart XSS	√	√	√	√	√	√	√	×	√	88.89%

具体分析如下：漏洞1是由未经安全处理的参数返回显示造成的，使用基础攻击向量即可检出该

表 6　3 种工具的检测请求数

检测工具	漏洞 1	漏洞 2	漏洞 3	漏洞 5	漏洞 6	Requests_mean
XSS Validator	2	25	29	4	41	20.2
Wapiti	4	4	7	10	4	5.8
SmartXSS	2	2	17	78	7	21.2

漏洞;漏洞 2 位置采用过滤小写样式<script>标签的安全措施,因此可通过大小写混淆绕过过滤,3 种工具均包含此类攻击向量;漏洞 3 位置采用了不区分大小写过滤<script>标签的安全措施,Wapiti 使用触发事件型的攻击向量绕过,攻击向量为<SvG/oNloAd = alert(/wfu2xf7ffy/)>,XSS validator 与 SmartXSS 使用拼接标签的方法绕过,攻击向量为<scri<script>pt>alert(1);</scr</script>ipt>;漏洞 4 位置采用不区分大小写匹配<script>标签匹配成功则直接终止程序运行的安全措施,Wapiti 与 SmartXSS 使用了事件触发型攻击向量,而 XSS Validator 由于自动添加一些特殊字符导致该型攻击向量失效;漏洞 5 位置采用对"alert"进行了不区分大小写过滤的安全措施,Wapiti 使用外部链接脚本进行绕过,XSS Validator 与 SmartXSS 采用了其他 JavaScript 方法,如 confirm;漏洞 6 位置将变量封装<script>标签的双引号内,Wapiti 采用闭合前向标签的方法,使用的攻击向量为</script><ScRiPt>alert('w5c4f6jkj9')</sCrIpT>,XSS Validator 与 SmartXSS 采用闭合前面的双引号并插入 JavaScript 代码并闭合后面双引号的方法,使用的攻击向量为";alert(299792458);";漏洞 7 与漏洞 6 类似,只是将变量置于单引号中,XSS Validator 与 SmartXSS 使用';alert(1);'检出该漏洞,Wapiti 未检出;漏洞 8 的变量采用 form 表单传入并使用实体标记编码输出位置,变量经过实体编码后无法执行 JS 代码,但其存储表单输入的中间变量存在漏洞,可先闭合 form 标签,再插入攻击向量,但 3 种工具均无法查询到中间变量,因此无法向中间位置变量注入攻击向量,故未检出该漏洞;漏洞 9 通过 location.hash 传递参数,因此直接使用攻击向量覆盖到锚标记位置即可,由于 Wapiti 与 XSS Validator 未检测锚标记的设计,因此只有 SmartXSS 检出该漏洞。

4　结束语

本文从提升 XSS 漏洞动态检测方法的检测能力出发,提出了一种基于 FP-Growth 算法的 XSS 动态检测方法。通过建立 XSS 攻击实例关联规则模型大量真实 XSS 攻击实例的关联关系,并以此为基础构造包含更多种类攻击向量的攻击向量库;基于 Selenium 设计并实现了 XSS 漏洞检测模块,相较于 XSS validator 等工具添加了锚链接注入点匹配以及二次操作的验证方式。实验结果表明,所提方法在实验环境下较其他同类方法具有更高的 XSS 漏洞检测率。下一步通过降低 XSS 漏洞动态检测方法的平均请求数进一步提升检测效率。

参考文献:

[1] 奇安信行业安全研究中心.2019 年中国政企机构网络安全形势分析报告[EB/OL].(2020-03-13)[2021-11-01].https://shs3.b.qianxin.com/qax/54d4179815390b5bbe8185408d5ccb7f.pdf.

[2] 孙伟,张凯寓,薛临风,等.XSS 漏洞研究综述[J].信息安全研究,2016,2(12):1068-1079.

[3] 刘奇旭,温涛,闻观行.Flash 跨站脚本漏洞挖掘技术研究[J].计算机研究与发展,2014,51(7):1624-1632.

[4] 李威,李晓红.Web 应用存储型 XSS 漏洞检测方法及实现[J].计算机应用与软件,2016,33(1):24-27,37.

[5] HOU X Y,ZHAO X L,WU M J,et al.A dynamic detection technique for XSS vulnerabilities[C]//2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC).Wuhan,China:IEEE,2018:34-43.

[6] 黄文锋,李晓伟,霍占强.基于 EBNF 和二次爬取策略的 XSS 漏洞检测技术[J].计算机应用研究,2019,36(8):2458-2463.

[7] 潘瑾琨.跨站脚本漏洞检测技术研究[D].长沙:国防科技大学,2017.

[8] GUO X B,JIN S Y,ZHANG Y X.XSS vulnerability detection using optimized attack vector repertory[C]//2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery.Xi'an,China:IEEE,2015:29-36.

[9] GARN B,RADAVELLI M,GARGANTINI A,et al.A fault-driven combinatorial process for model evolution in XSS vulnerability detection[C]//Advances and Trends in Artificial Intelligence.From Theory to Practice.Cham:Springer International Publishing,2019:207-215.

[10] 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.

[11] HAN J W,PEI J,YIN Y W,et al.Mining frequent patterns without candidate generation:a frequent-pattern tree approach[J].Data Mining and Knowledge Discovery,2004,8(1):53-87.

[12] PortSwigger.Cross-site scripting (XSS) cheat sheet[EB/OL].[2021-07-07].https://portswigger.net/web-security/cross-site-scripting/cheat-sheet.

(编辑:高明霞)