·安全技术· 文章编号：1000-3428(2018)08-0167-07 文献标志码：A 中图分类号：TP319

一种自动化的跨站脚本漏洞发现模型

马富天，钱雪忠，宋 威

（江南大学 物联网工程学院 物联网技术应用教育部工程研究中心，江苏 无锡 214122）

摘 要： 跨站脚本攻击给 Web 应用带来严重的威胁，在应用发布之前，对其进行检测能够有效地降低漏洞风险。针对现有跨站脚本在动态检测中存在漏报误报的问题，提出一种动态检测方法。基于攻击向量基本侯选元素库和初始攻击向量种子库，在检测过程中自动生成符合输出点类型的有效攻击向量，根据当前时刻的检测结果，自适应调整攻击向量优先级，待所有注入点攻击完毕，重新二次遍历整个站点检验待发现的漏洞。实验结果表明，与 APPScan、WVS 相比，该方法能发现更多漏洞。

关键词： 跨站脚本；动态检测；静态分析；攻击向量；合法向量

中文引用格式： 马富天，钱雪忠，宋 威. 一种自动化的跨站脚本漏洞发现模型[J]. 计算机工程，2018，44(8)：167-173.

英文引用格式： MA Futian，QIAN Xuezhong，SONG Wei. An automated cross-site scripting loopholes discovery model[J]. Computer Engineering，2018，44(8)：167-173.

An Automated Cross-site Scripting Loopholes Discovery Model

MA Futian，QIAN Xuezhong，SONG Wei

（Engineering Research Center of Internet of Things Technology Applications Ministry of Education，
School of Internet of Things Engineering，Jiangnan University，Wuxi，Jiangsu 214122，China）

【Abstract】 Cross-site Scripting（XSS） attacks pose serious threats to web applications. Before the application is released，detecting them can effectively reduce the risk of vulnerabilities. Aiming at the problems in the current detection of cross-site scripting，such as missed reports and false alarms，a dynamic detection method is proposed. Based on the basic candidate element library of attack vectors and the initial attack vector seed library，an effective attack vector conforming to the output point type is automatically generated during the detection process. According to the detection result at the current moment，the priority of the attack vector is adaptively adjusted，and all the injection point attacks are performed，after finishing，it traverses the entire site twice to check the vulnerabilities to be discovered. Experimental results show that compared with APPScan、WVS，this method can find more vulnerabilities.

【Key words】 Cross-site Scripting（XSS）；dynamic detection；static analysis；attack vector；legal vector

DOI： 10.19678/j.issn.1000-3428.0047650

0 概述

跨站脚本（Cross-site Scripting，XSS）是当前最为普遍的 Web 安全漏洞之一。根据 OWASP（Open Web Application Security Project）开放式 Web 应用程序安全项目在 2007 年—2017 年期间进行了四次十大 Web 应用安全漏洞统计，报告显示跨站脚本攻击一直高居前列，是黑客攻击 Web 应用的主要手段之一。根据 XSS 攻击方式及特征，可以将它分为反射型跨站脚本（reflected-XSS）、存储型跨站脚本（stored-XSS）和基于文档对象模型的跨站脚本（DOM-based XSS）[1]，其中，存储型跨站攻击的危害较大。产生跨站脚本的直接原因是由于 Web 应用程序缺少对用户输入的数据进行有效验证与过滤[2-3]。攻击者利用存在的漏洞把恶意脚本注入到页面中，当用户浏览这些页面时，会触发页面中恶意脚本的执行，对受害者发起信息盗取、会话劫持、网页木马挂载及破坏页面结构等各类攻击。

通常检测跨站脚本的方法主要有静态分析、动态分析和动静结合方式检测[4]。静态分析是通过审核源代码的方式来挖掘潜在的漏洞，如文献[5-6]，其检测准确率和效率比动态分析方法高一些。但在多数情况下，检测人员都无法直接获得 Web 应用的源代码，并且它对程序逻辑数据处理中出现的问题

基金项目： 国家自然科学基金（61673193）；中央高校基本科研业务费专项资金（JUSRP51510，JUSRP51635B）。

作者简介： 马富天（1992—），男，硕士研究生，主研方向为 Web 安全；钱雪忠，副教授、硕士；宋 威，副教授、博士。

收稿日期： 2017-06-20 **修回日期：** 2017-08-23 **E-mail：** mafutian@126.com

不能有效的检测,对编程语言的类型也有一定的要求,同时其检测能力也受限于源代码的框架,可扩展性能力较差。动态分析是指在无需源代码的前提下,采用模糊测试的方法来挖掘漏洞。在动态分析方法中,文献[7]通过分析 HTML5 的新标签、新属性,引入了 14 个 HTML5 元素相关的初始攻击向量,基于这些攻击向量构建了攻击向量库,实现一个侧重于 Webmail 系统的动态跨站脚本检测工具,该工具能够有效的利用基于 HTML5 的攻击向量检测出系统中的 XSS。文献[8]提出通过攻击向量规则库生成初始攻击向量,然后对初始攻击向量使用机器学习算法进行优化,大幅度的减少初始攻击向量库的大小,得到最后的攻击向量库。

本文针对反射型和存储型跨站脚本漏洞,提出一种动态检测 XSS 的方法。该方法首先通过自动爬虫爬取并分析 Web 应用的所有页面及注入点,然后根据输入点、输出点类型自动生成合法向量与攻击向量,在攻击过程中自适应调整攻击向量的优先级,优先选取当前时刻优先级较高的攻击向量类型,向服务器提交 HTTP 请求,根据反馈信息判断漏洞是否存在,待整个攻击过程结束,对站点进行二次遍历,重新检测每个页面中尚未被验证的攻击向量。

1 测试向量

Web 应用程序为了避免受到跨站脚本攻击,所有的不受信数据都必须要经过过滤模块才能被写入数据库或返回至客户端浏览器。过滤模块是指 Web 应用程序中用来过滤或净化用户所输入的数据以防止引起跨站脚本攻击的函数或模块。过滤模块比较常用的方法[9]包括黑名单过滤,一般使用正则表达式进行合法性校验;另一种做法是只接收已知的合法数据(白名单过滤)或对输入长度与编码进行限制;最后一种方法是将特殊字符或标签转成无害的。

因此,为了能够充分测试过滤模块是否有效,需要精心地构造出尽可能全面且有效的测试向量来绕过各种验证函数,从而挖掘出由于过滤模块过滤不够全面而造成的跨站脚本。测试向量包括攻击向量、合法向量。传统的检测方法在攻击过程中,对每个注入点的所有注入变量同时发送攻击向量,但这会极大降低攻击的成功率。本文模型提出在发送攻击的过程中,每次攻击仅对注入点的某个注入变量发送攻击向量,而其他的注入变量使用合法向量。

1.1 攻击向量

针对 Web 应用程序过滤模块可能采取的过滤方式[10],表 1 列举部分常见的过滤方式,可依此制定出能够绕过过滤模块的攻击向量。

表 1 Web 应用程序过滤方式

方式	描述
F_1	不过滤输入内容,未作出任何过滤处理
F_2	将输入的敏感词屏蔽或删除
F_3	将输入的敏感词替换成无害字符
F_4	将输入的敏感词添加注释
F_5	将特殊符号进行转义或实体化,如" <"" >"等
F_6	将输入的 HEX/DEC 字符重新编码成 ASCII 字符
F_7	只对小写字符进行过滤或只对大写字符进行过滤
F_8	只对半角字符进行过滤或只对全角字符进行过滤

攻击向量即指能够执行 JavaScript 代码的恶意字符串(例如:< script > alert(1) </script >),攻击向量的类型多种多样。通过分析许多著名安全工程师共同总结出的 XSS Cheat Sheet(XSS 过滤绕过备忘单)[11],本文在 HTML4 元素的基础上,也引入了 HTML5 新标签、新属性构建了基本候选元素库(见表 2),并构造出基于 HTML5 的攻击向量(见表 3),最后总结出初始攻击向量种子库(见表 4)。初始攻击向量种子库是指所有尚未经过格式变换的已知的可以用于跨站攻击的 XSS 代码集合,库中的每一个 XSS 代码都代表了一种 XSS 攻击类型。

表 2 基本候选元素库

符号	候选元素
HTML_Tag	script,body,div,span,article,aside,figure,…
HTML_Attr	value,autofocus,placeholder,…
HTML_Attr2Tag	src:[img],action:[form],href[a,…]
HTML_CSSAttr	style,…
HTML_URLAttr	src,href,action,dynsrc,lowsrc,longdesc,…
HTML_Event	onclick,ondblclick,onload,onhashchange,…
HTML_Event2Tag	onerror:[img,…],onclick:[div,p,…],…
HTML_InsideTagText	Text inside HTML tag
JS_Payload	alert(1),confirm(1),prompt(1),…
URL_Payload	javascript:JS_Payload,…
CSS_Payload	height:expression(JS_Payload),background-image:url(URL_Payload)
FuzzBigin	',",>,\>,-->,</title>,</textarea>,1',*/;,…
FuzzJS	/* abc */,\t,\r,\n,…
FuzzCSS	/* abc */,\\,\0
FuzzHtmlSpace	space,\n,\t,\r,…
FuzzText	Fuzzy text
Quote	“,’,none

表 3 基于 HTML5 的攻击向量示例

序号	代码示例
1	< input onblur = alert(1) autofocus / >
2	< video > < source onerror = alert(1) >
3	< body onhashchange = alert(1) > < a href = # > click
4	< svg onload = alert(1) / >
5	< header ondblclick = alert(1) > HEADER < /header >
6	< audio src onerror = alert(1) >
7	< svg xmlns:svg = "http://www.w3.org/2000/svg" xmlns = "http://www.w3.org/2000/svg" xmlns:xlink = "http://www.w3.org/1999/xlink" version = "1.0" x = "10" y = "10" width = "60" height = "60" > < script type = "text/javascript" > confirm(1); </script > </svg >

表 4　初始攻击向量种子库

种子	模式
S_1	< HTML_Tag >　JS_Payload　</HTML_Tag >
S_2	< HTML _ Tag　HTML _ URLAttr = URL _ Payload > HTML_InsideTagText </HTML_Tag >
S_3	< HTML_Tag　HTML _ Event = JS _ Payload > HTML _ InsideTagText </HTML_Tag >
S_4	< HTML_Tag > CSS_Payload　</HTML_Tag >
S_5	< HTML _ Tag　HTML _ CSSAttr = CSS _ Payload > HTML_InsideTagText　</HTML_Tag >
S_6	JS_Payload
S_7	URL_Payload
S_8	CSS_Payload
S_9	HTML_Event = JS_Payload
S_{10}	HTML_Tag | CSS_Payload |

如果在 Web 应用的过滤模块中,对某种编码格式不支持的话,则可能通过编码格式转换的方式来绕过过滤[12-13]。在实际中,攻击者会对攻击向量做各种形式的变换来试图规避过滤模块对用户输入进行的过滤和净化。其中编码变换有十六进制编码、八进制编码或 UTF-8 编码等。在初始攻击向量经过变换规则(见表 5)变形后,添加模糊前缀,得到最终的攻击向量。针对不同的输出点类型应该选取相应的攻击向量种子(见表 6),才能够在该输出点位置上触发 JavaScript 代码的执行。最后,由于仅通过攻击向量自动生成算法构造的攻击向量并不能完全涵盖所有的类型,因此本文模型另外构建了攻击向量静态库作为补充,使其覆盖范围更为全面。

表 5　攻击向量变换规则库

规则	描述	变换对象
R_1	随机大小写编码(仅大写、仅小写、混合编写)	[HTML_Tag,HTML_Attr,…]
R_2	八进制编码	[JS_Payload,URL_Payload,…]
R_3	十六进制编码	[JS_Payload,URL_Payload,…]
R_4	Unicode 编码	[JS_Payload,…]
R_5	base64 编码	[S_1,S_2,…]
R_6	敏感词分解(添加注释、回车、tab 键等)	JS_Payload,URL_Payload,…]
R_7	添加模糊前缀	[S_1,S_2,…]
R_8	全角字符、半角字符的混合输入	[HTML_Tag,HTML_Attr,…]
R_9	混合使用以上方法	[HTML_Tag,HTML_Attr,…]

表 6　攻击向量输出点类型

序号	类型	描述	种子
P_1	在 HTML 普通标签之间	这是最常见的情况,如 div、p、td、th 等,无需设置前缀	S_1、S_2、S_3、S_4、S_5
P_2	在 HTML 特殊标签中	如 title、iframe、textarea、noscript 等无法执行脚本代码的标签	S_1、S_2、S_3、S_4、S_5
P_3	在 JavaScript 节点中	可以使用 </script > 或分号来结束	S_1、S_2、S_3、S_4、S_5 或 S_6
P_4	在注释之间	使用前缀-- > 来关闭注释	S_1、S_2、S_3、S_4
P_5	在 style 标签中	使用 </style > 或分号结束	S_1、S_2、S_3、S_4、S_5 或 S_{10}
P_6	在支持 URL 伪协议的属性中	如 href、src 等	S_7
P_7	标签的 style 属性中	使用;来结束	S_8
P_8	标签普通属性中	如 input[value]中,使用单引号或双引号结束	S_9

1.2　合法向量

传统检测方法没有针对合法向量的选取与生成进行详细分析,但合法向量的选取也是至为重要的。若仅使用数字与字母组合随机生成的字符串作为合法向量,极为可能无法绕过过滤模块,导致注入失败。因此,本文提出根据表单相关信息(即输入点类型),构造出相应的合法向量。

例如:对于表单元素 < input type = " email" name = " email" required = " required" / >,通过有效信息可以判断需要输入的是邮箱地址,因此,自动生成向量如:xSs _ sCanner123 @ test. com 等,对于 < input type = " text" name = " url" size = " 255" placeholder = " 请输入网址" / >,通过分析相关有效信息可以判断需要输入的是网址,可生成向量如:http://www. scanner1. com 等。表 7 列举部分表单元素相关信息,利用这些信息有助于生成更加准确有效的合法向量。

表 7　表单元素相关信息

表单元素	属性(有效信息)
form	method,action,autocomplete
input	name, value, size, required, placeholder, type:[text,password,hidden,url,email,number,…]
textarea	name,innerHTML ,placeholder
button	name,type,value
select[option]	name,value

2 总体结构与检测流程

本文模型总体结构(如图 1 所示)主要分为 3 个模块,分别是注入点分析模块、测试向量挖掘模块、攻击与分析模块。各个模块之间通过查询与操作 session 对象或数据库来完成相互协作。

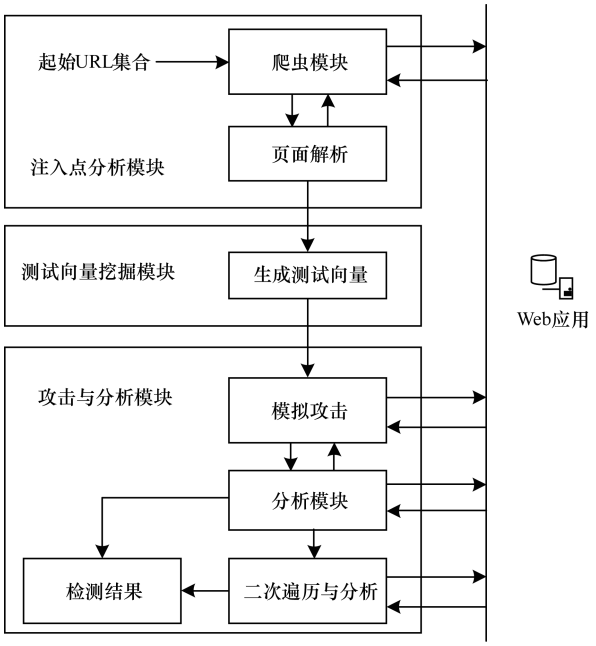

图 1 本文模型总体结构

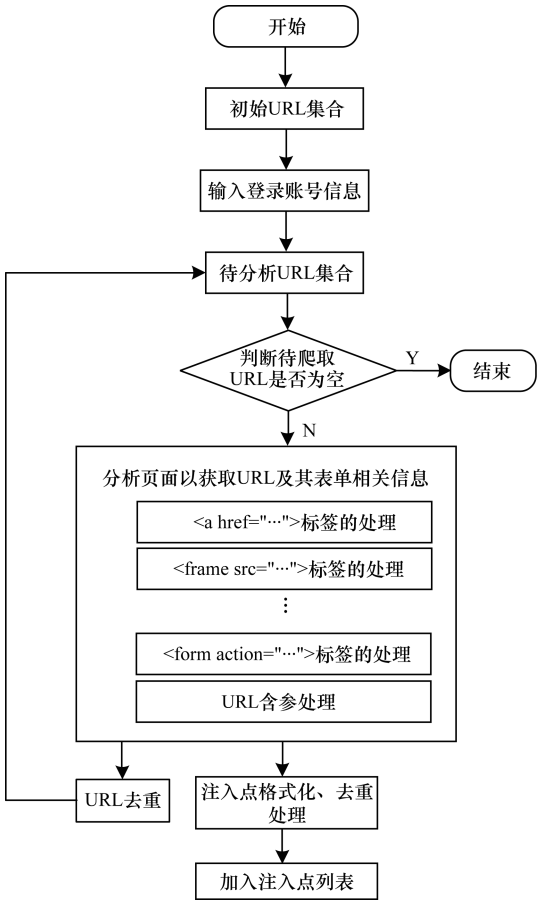

图 2 注入点分析模块结构

2.1 注入点分析模块

完整、准确、高效地挖掘出所有页面及漏洞注入点是增强检测能力、减少漏报误报的关键问题之一,注入点分析模块(如图 2 所示)在检测中起着重要的作用。当前,网页提供给用户输入数据的方式各式各样,而传统检测中爬虫的功能较为单一(如文献[14]),仅能够提取出表单内容,比较难解析基于 JavaScript 或 Ajax 的提交方式,对于需要登录后才能够访问的页面无法获取到。为了获取更为全面的注入点,将跨站脚本攻击的注入点可大致分为 3 类:1)表单输入类;2)含参数的 url 类;3)基于 JavaScript 或 Ajax 的数据提交方式。注入点分析模块通过爬虫爬取目标站点的所有页面,并分析页面以获取注入点及相关信息。本文采用多线程网络爬虫,在爬取过程中采取广度优先搜索策略,使用正则表达式匹配页面中的 url,过滤不属于目标站点的 url,并将相对路径的 url 转成绝对路径加入到待爬取队列中,同时使用 Bloom Filter 算法[15]对 url 进行去重。

在整个爬取过程中需要与 Web 服务器进行多次交互,针对一些需要登录后才能访问的页面,需要保持会话状态,否则在访问页面时 Web 应用会拒绝访问或者页面重定向。本文方法运用 cookie 技术来解决这一问题,预先输入用户登录的相关信息(如账号、密码等)以获取返回的 cookie 信息,并需要将退出登录的链接从待爬取 url 集合中删去,当再次访问页面时,则将相应的 cookie 信息添加到爬虫的 HTTP 请求协议头中,使其能够在爬取过程中保持会话状态,以获取更多页面。

在爬取 url 的过程中,分析注入点,将含参的 url 类与基于 Ajax 的数据提交类注入点使用表单类形式保存,统一注入点形式,同样也需对注入点进行去重处理。

2.2 测试向量生成模块

由第 1 节提供的攻击向量基本候选元素库,以及攻击向量变换规则库,并添加模糊数据,例如注释、空白符、tab 键等,且动态生成模糊前缀,生成最终的攻击向量(见表 8)。

表 8　最终的攻击向量

序号	攻击向量
1	3′ > < b oNclick = \u0061 \u006c \u0065 \u0072 \u0074 (3) > xss < /B >
2	– > < object daTa = " data: text/html; base64, MTAiIDxhIGhSZWY9IkpBdmFTQ1JpcFQ6ZG9jdW1lbn Qud3JpdGUoNDcyNykiPjBpRVY8L2E + " > < /object >
3	< /tiTLe > < a hREf = jav A Sc R IP T : prom pt(4506) > 1luC < /A >
4	< a hReF = javaScriPt: consol e . log(9551) > 9gqr < /A >
5	x > < H4 STYLe = FONT-SIZE: expression (\u0063 \u006f\u006e\u0073\u006f\u006c\u0065. \u006c\u006f\u0067(2836)) ; > GRSZ < /h4 >
6	3′ < ScR < scRIPt > IPT > prompt(5103) ; < /SCRIPT >

2.3　检测与分析模块

并非所有用户能够输入数据的地方都能够导致跨站脚本的产生。发生反射型和存储型跨站脚本攻击的必要条件是用户输入的数据能够在某一个时刻被 Web 应用程序调用且加载到 Web 页面中。

传统检测方法将攻击向量提交至漏洞注入点,且对该注入点的所有注入变量同时使用攻击向量进行攻击,在攻击过程中判断漏洞是否存在只需检测响应页面中是否含有该攻击向量,就能大致判断漏洞是否存在。但这种检测方法是不完善的,易造成漏报误报,因为攻击向量不仅可能出现在服务器响应页面中,还可能某一个时刻出现在其他任何页面中,并且在同一页面中也可能有若干个不同的输出点类型(如图 3 所示)。

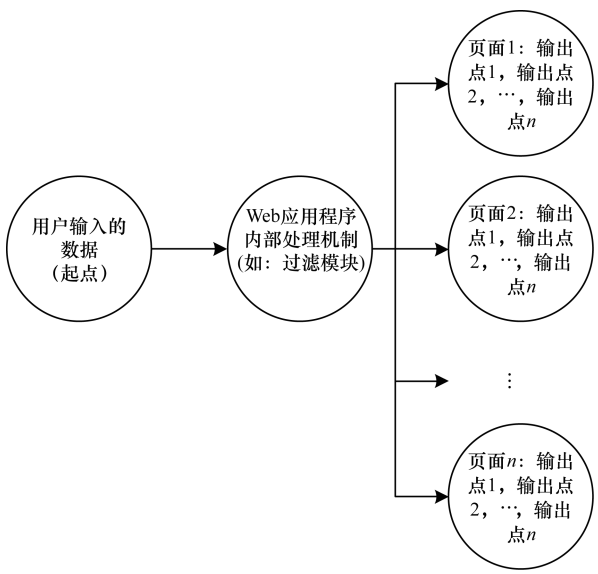

图 3　输入数据输出终点的处理过程示意图

在检测效率方面,传统检测方法针对每个注入点,将攻击向量库中的攻击向量全部提交完成后才判断漏洞是否存在,这样会大大降低检测效率。在现有动态生成攻击向量的方法中,对于攻击向量的选取大多数采用随机选取方法,其优点在于简单快速,易于理解与实现,但由于随机选取方法中并没有利用其他任何相关的信息,其检测能力往往不是很理想。因此,如何使用较少的攻击向量来尽快发现漏洞是尤为重要的。

为解决上述问题,本文提出单次攻击只对某一注入点的某一个注入变量在某一输出点类型上进行攻击。首先,在检测过程中根据注入变量的输出点类型自动生成攻击向量,并随着当前时刻的检测结果自适应调整攻击向量优先级,优先选取攻击成功概率较高的攻击向量类型,以尽可能早地发现漏洞,弥补随机选取方法的不足。

为了对不同漏洞注入点注入的攻击向量进行区分,需要对每个注入点的某一个注入变量的攻击向量分配一个唯一的标识(即攻击向量 ID),同时其他的注入变量根据输入点类型自动选取相应合法向量,向 Web 应用服务器发送 Get 或 Post 请求。当检测到特殊字符如 < , > 等标签已经被后台实体化或过滤后,导致无法执行脚本代码,则停止对此注入点变量的攻击。另外,将获取到的输出点类型攻击完毕之后,若依旧没有发现漏洞,则对每个注入变量都需要提交其他尚未选取的输出点类型自动选取对应的攻击向量进行攻击,避免在其他页面中有不同输出点类型中出现,造成漏报。在判断攻击是否成功的过程中,将分析页面范围扩大至注入点展示页面、请求页面以及服务器响应页面,判断漏洞存在当且仅当攻击向量类型与输出点类型相符合,并且攻击向量 ID 与注入点变量相匹配,能够触发脚本代码,才判定漏洞存在,这样可以降低误报率。

当所有注入点被攻击完成后,对整个网站进行二次遍历,检测尚未被验证的注入变量。如图 3 所示,用户输入的数据可能在任何页面的任何输出点位置上,且每个输出点位置可能是不同的输出点类型,因此,仅仅在攻击过程中同时根据反馈信息来判断漏洞是否存在会造成漏报问题,二次遍历对 Web 应用的原有页面与新页面进行重新遍历搜索,查找所有尚未得到验证的攻击向量,一旦确认查找到的攻击向量与相应的输出点类型匹配,则判断该攻击向量对应的注入点变量是存在跨站脚本漏洞的,并将该注入点变量对应的所有攻击向量移除,然后继

续匹配直到待查找的攻击向量为空或者所有页面已遍历完毕,则整个检测过程结束。

假设 I 指注入点集合,其中,I_i 指第 i 个注入点,I_{ij} 指 I_i 的第 j 个注入变量,$i \in \{1,2,\cdots,n\}$(n 是注入点总数),$j = \{1,2,\cdots,m\}$(m 是 I_i 的注入变量总数),P 指输出点类型集合,P_k 指 I_{ij}的第 k 个输出点位置,$k \in \{1,2,\cdots,s\}$(s 是 I_{ij}的输出点类型总数),q 指对 I_{ij}在 P_k 上的攻击次数,H_t 指对 I_{ij}在位置 P_k 发送的第 t 次 http 请求,$t \in \{1,2,\cdots,q\}$,U 指 url 集合,$\boldsymbol{V}$ 指具有 I_{ij}标识的攻击向量,V_{list}指待检测的攻击向量列表。

输入 注入点集合 I,url 集合 U

输出 发现的跨站脚本漏洞

具体检测算法描述如下:

1)若 $i < n$ 成立,则设 $i++$,从 I 中取出 I_i,设 $j = 1$;否则,转到第 9 个步骤。

2)若 $j < m$ 成立,则设 $j++$,从 I_i 中取出 I_{ij},设 $k = 1$;否则,重复第 1 个步骤。

3)若 $k < s$ 成立,则设 $k++$,$t = 1$,对 I_{ij}的下一个输出点类型 P_k 进行攻击;否则,重复第 2 个步骤。

4)若 $t < q$ 成立,则设 $t++$,对 I_{ij}在 P_k 上选取优先级较高的攻击向量 $\boldsymbol{V}$,并将 $\boldsymbol{V}$ 加入到 V_{list}中,对 $I_{ij'}$($j' \neq j$)选取合法向量,对 I_i 发送 H_t;否则,重复第 3 个步骤。

5)检测相关反馈页面中出现的 $\boldsymbol{V}$ 所在输出点类型是否与攻击向量 $\boldsymbol{V}$ 相符合,若是,则判定 I_{ij}存在漏洞,更新 $\boldsymbol{V}$ 的优先级,并从 V_{list}中移除与 I_{ij}相关的 $\boldsymbol{V}$,停止攻击 I_{ij},重复第 4 个步骤。

6)检测相关反馈页面中出现的 $\boldsymbol{V}$ 所在输出点类型,更新 I_{ij}的输出点类型集合 P。

7)检测相关反馈页面中出现的 url,更新集合 U。

8)检测相关反馈页面中是否存在被实体化后的攻击向量,若是,且在 P_k 不符合触发脚本执行条件,判定此输出点类型不存在漏洞,并结束对 I_{ij}在 P_k 的攻击,重复第 3 个步骤;若否,重复第 4 个步骤。

9)遍历 U 中所有的 url,匹配 V_{list}中的 $\boldsymbol{V}$,若输出点类型与 $\boldsymbol{V}$ 的类型相符合,则判定存在漏洞,并从 V_{list}中删除与 I_{ij}相关的 $\boldsymbol{V}$。当 U 遍历完毕或 V_{list}为空时,则检测结束并输出结果。

3 实验结果与分析

依据本文提出的检测方法,设计并实现了一款自动化跨站脚本发现模型 WXS,模型使用 PHP 作为服务器端脚本语言,具有易于进行维护和二次开发的特点。检测流程简单、实用性强、自动化程度高。为了验证模型的有效性,同时使用著名商业工具 WVS(Web Vulnerability Scanner)与 APPScan 对2 个测试站点进行扫描测试。其中,站点 1 是自行搭建的个人博客站点,该网站使用 PHP 语言开发,使用 MySQL 数据库,站点 2 是某论坛网站,采用. NET 框架,提供用户留言评论等功能。检测结果如表 9 和表 10 所示,表 11 为检测耗时比较。

表 9 站点 1 检测结果

模型	URL 数	注入点数	XSS 数	误报数	http 请求次数
WXS	623	22	22	0	1 784
WVS	610	20	17	1	3 206
APPScan	769	20	13	0	4 237

表 10 站点 2 检测结果

模型	URL 数	注入点数	XSS 数	误报数	http 请求次数
WXS	85	2	4	0	467
WVS	53	1	1	0	1 342
APPScan	103	1	1	0	2 218

表 11 检测耗时比较

站点	WXS	WVS	APPScan
站点 1	17 min 43s	11 min 40 s	13 min 5 s
站点 2	3 min 33s	7 min 50 s	7 min 53 s

为了更好地体现出模型 WXS 的有效性,需要在真实环境下做测试。据 XSSed 官方(XSS 漏洞公布平台)截止到目前不完全数据显示,现收集有 45 884 个XSS 漏洞,已修复 3 026 个漏洞。本文实验从 XSSed 平台中分别收集已修复和尚未修复的漏洞各 50 例作为测试样本,分别使用上述 3 种模型对其进行检测,检测结果如表 12 所示。

表 12 XSSed 平台收集的测试样本检测结果

模型	XSS 个数	误报数
WXS	41	0
WVS	38	1
APPScan	23	0

实验结果表明,本文设计的模型 WXS 与其他工具相比较,能够检测出更多页面和注入点,挖掘出漏洞数量较多,并且误报数少。APPScan 没有针对 Ajax 提交数据的注入点类进行分析与研究,不能检测出此类注入点。WVS 与 APPScan 没有进行二次遍历扫描,并且缺乏对合法向量进行挖掘,容易被后台过滤模块过滤,存在漏报问题。在耗时方面,WXS 发送的 http 请求次数(攻击次数)少于 WVS 与

APPScan,说明在攻击向量与合法向量的选取上有较强的能力。

4　结束语

本文在分析跨站脚本攻击原理的基础上,提出一种动态检测反射型与存储型跨站脚本方法,设计并实现了一款自动化检测模型。该模型与同类检测工具相比,能够降低误报率和漏报率,从而更加有效地发现 Web 应用中的跨站脚本漏洞。下一步将针对 Dom-based XSS 进行分析与研究,使所提模型能够检测出该类型的跨站脚本漏洞。

参考文献

[1] 邱永华. XSS 跨站脚本攻击剖析与防御[M]. 北京:人民邮电出版社,2013.

[2] PIETRASZEK T, BERGHE C V. Defending against injection attacks through context-sensitive string evaluation[C]//Proceedings of International Conference on Recent Advances in Intrusion Detection. Berlin, Germany:Springer-Verlag,2005:124-145.

[3] ISMAIL O, ETOH M, KADOBAYASHI Y, et al. A proposal and implementation of automatic detection/collection system for cross-site scripting vulnerability[C]//Proceedings of International Conference on Advanced Information Networking and Applications. Washington D. C. ,USA:IEEE Press,2004:145-151.

[4] 王　丹,赵文兵,丁治明. Web 应用常见注入式安全漏洞检测关键技术综述[J]. 北京工业大学学报,2016,42(12):1822-1832.

[5] JOVANOVIC N,KRUEGEL C,KIRDA E. Pixy:a static analysis tool for detecting web application vulnerabilities[C]//Proceedings of IEEE Symposium on Security and Privacy. Washington D. C. , USA: IEEE Press,2006:136-263.

[6] CEPONIS J, CEPONIENE L, VENCKAUSKAS A, et al. Evaluation of open source server-side XSS protection solutions[J]. Communications in Computer & Information Science,2013,403:345-356.

[7] DONG Guowei, ZHANG Yan, WANG Xin, et al. Detecting cross site scripting vulnerabilities introduced by HTML5 [C]//Proceedings of International Joint Conference on Computer Science and Software Engineering. Washington D. C. , USA: IEEE Press, 2014:319-323.

[8] GUO Xiaobing, JIN Shuyuan, ZHANG Yaxing. XSS vulnerability detection using optimized attack vector repertory[C]//Proceedings of International Conference on Cyber-enabled Distributed Computing and Knowledge Discovery. Washington D. C. , USA: IEEE Press, 2015: 29-36.

[9] 钟晨鸣,徐少培. Web 前端黑客技术揭秘[J]. 计算机安全,2013(1):47.

[10] 吴子敬,张宪忠,管　磊,等. 基于反过滤规则集和自动爬虫的 XSS 漏洞深度挖掘技术[J]. 北京理工大学学报,2012,32(4):395-401.

[11] ROBERT H. XSS filter evasion chea-tsheet[EB/OL]. [2016-12-21]. https://www. owasp. org/index. php/XSS_Filter_Evasion_Cheat_Sheet.

[12] 潘古兵,周彦晖. 基于静态分析和动态检测的 XSS 漏洞发现[J]. 计算机科学,2012,39(6):51-53.

[13] SHAHRIAR H, ZULKERNINE M. MUTEC: mutation-based testing of cross site scripting[C]//Proceedings of ICSE Workshop on Software Engineering for Secure Systems. Washington D. C. , USA: IEEE Computer Society,2009:47-53.

[14] 沈寿忠,张玉清. 基于爬虫的 XSS 漏洞检测工具设计与实现[J]. 计算机工程,2009,35(21):151-154.

[15] Bloom filter [EB/OL]. [2016-11-21]. https://en. wikipedia. org/wiki/Bloom_filter.

编辑　刘　冰

(上接第 166 页)

[5] NARAYANAN A,CHEN Yongsheng, PANG Shuping. The effects of different representations on static structure analysis of computer malware signatures[J]. Scientific World Journal,2013,9(10):1155-1160.

[6] 郭　晨,梁家荣,梁美莲. 基于 BP 神经网络的病毒检测方法[J]. 计算机工程,2005,31(2):152-156.

[7] 吴俊利,张步涵,王　魁. 基于 Adaboost 的 BP 神经网络改进算法在短期风速预测中的应用[J]. 电网技术,2012,36(9):221-225.

[8] XU Lei, LI Yanda. Machine learning and intelligence science [J]. Frontiers of Electrical and Electronic Engineering in China,2012,7(11):1-4.

[9] 杨新武,马　壮,袁　顺. 基于弱分类器调整的多分类 Adaboost 算法[J]. 电子与信息学报,2016,38(2):373-380.

[10] 李　闯,丁晓青,吴佑寿. 一种改进的 AdaBoost 算法——AD AdaBoost[J]. 计算机学报,2007,30(1):103-109.

[11] 邱仁博,娄　震. 一种改进的带参数 AdaBoost 算法[J]. 计算机工程,2016,42(7):199-202,208.

[12] 刘明亮,甄建聚,孙来军,等. 基于 DS 证据理论的 SVM 分类模糊域数据修正[J]. 电力自动化设备,2012,32(3):71-75.

[13] 费　翔,周　健. 一种处理冲突证据的 D-S 证据权重计算方法[J]. 计算机工程,2016,42(2):142-145.

[14] 廖　平,郑友娟,覃才珑. 基于改进证据理论的全信息故障诊断[J]. 计算机工程,2016,42(3):308-311,316.

[15] BAYOGLU B, SOGUKPINAR I. Polymorphic worm detection using strong token-pair signatures[J]. Turkish Journal of Electrical Engineering and Computer Sciences,2012,17(2):163-182.

[16] YERIMA S Y, SEZER S, MCWILLIAMS G. Analysis of Bayesian classification based approaches for android malware detection[J]. IET Information Security,2013,8(1):25-36.

编辑　刘　冰