

基于 AST 分析与 Fuzzing 的反射型 XSS 漏洞识别模型

郭晓军^{a, b}, 闫宇辰^a, 吴志浩^a

(西藏民族大学 a. 信息工程学院; b. 西藏自治区光信息处理与可视化技术重点实验室, 陕西 咸阳 712082)

摘要: 针对 Web 应用中反射型 XSS 检测效率较低、误报率较高的问题, 提出一种基于 AST 分析和 Fuzzing 的反射型 XSS 漏洞识别模型。通过发送探针载荷请求目标 Web 页面, 根据 AST 语法树解析结果, 初步判定该 Web 页面存在反射型 XSS 漏洞的可能性。再根据探针载荷回显位置确认该页面中可疑 XSS 注入点, 选择对应的逃逸技术和逃逸行为生成初始攻击载荷。将初始攻击载荷与绕过规则库相结合, 生成攻击向量库, 对可疑注入点进行 Fuzzing, 以确认是否存在反射型 XSS 漏洞。实验结果表明, 相较于 Burp Suite 和 AWVS, 在相同漏洞环境中, 此模型检测时平均请求次数较少。在保证较低误报率的同时, 有较高的检测效率。

关键词: 漏洞检测; 反射型 XSS 漏洞; AST 分析; 模糊测试; 攻击载荷

中图分类号: TP 393.08

文献标志码: A

文章编号: 1006-7167(2022)10-0049-05



Reflective XSS Vulnerability Identification Model Based on AST Analysis and Fuzzing

GUO Xiaojun^{a, b}, YAN Yuchen^a, WU Zhihao^a

(a. School of Information Engineering; b. Key Laboratory of Optical Information Processing and Visualization Technology of Tibet Autonomous Region, Xizang Minzu University, Xianyang 712082, Shaanxi, China)

Abstract: In view of the low detection efficiency and high false alarm rate of reflected XSS in Web applications, a reflected XSS vulnerability identification model based on AST analysis and Fuzzing is proposed. By sending the probe payload to request the target Web page, and according to the analysis result of the AST syntax tree, it is preliminarily determined that the Web page has the possibility of reflected XSS vulnerabilities. Then we confirm the suspicious XSS injection point in the Web page according to the echo position of the probe payload, and select the corresponding escape technology and escape behavior to generate the initial attack payload. The initial attack load is combined with the bypass rule library to generate an attack vector library, and fuzz the suspicious injection points to confirm whether there is a reflected XSS vulnerability. Experimental results show that compared with the Burp Suite and AWVS, in the same vulnerability environment, the average number of requests during detection is less. It ensures a low false alarm rate, and has a high detection efficiency.

Key words: vulnerability detection; reflected XSS vulnerability; AST analysis; fuzzing; attack payload

0 引言

随着 Web 应用的普及, 出现一系列安全问题。其中跨站脚本(Cross Site Scripting, XSS)是 Web 中一个持续存在的问题, 在 2017 版 OWASP Top 10 中, XSS 攻击位列第 7^[1]。XSS 攻击主要由于攻击者往 Web 页面里插入恶意 HTML 代码, 当用户浏览该页时, 嵌

收稿日期: 2022-01-10

基金项目: 西藏自治区自然科学基金项目(XZ2019ZRG-36(Z)); 西藏民族大学项目(324011810216; 324042000709)

作者简介: 郭晓军(1983-), 男, 山西长治人, 博士, 副教授, 研究方向为网络安全、网络测量。

Tel.: 13468528810; E-mail: aikt@xamu.edu.cn

入该 Web 页面的 HTML 代码会被执行,达到恶意攻击用户的特殊目的^[2-3]。其主要危害有网络钓鱼,会话劫持后执行任意操作,强制弹窗,网页挂马等,给普通用户和企业带来严重危害^[4-5]。

常见的 XSS 检测方式主要有静态分析^[6]和动态分析^[7]。静态分析技术通过分析 Web 应用的源代码或对目标代码进行语法分析及对源码结构和数据处理过程分析,发现系统潜在漏洞^[8]。与动态分析相比,检测效率和准确度相对较高。Ibéria 等^[9]提出使用静态分析的方法,找出并删除 Web 应用程序中潜在的漏洞。邱子谨^[10]设计并实现基于静态分析的 PHP 源码缺陷自动检测工具,采用基于格的数据流分析框架并提出一种基于有效路径的污点分析方法,对污点型漏洞有着较高的检测率,但只能针对 PHP 语言进行分析,具有局限性。

但大多数情况,无法直接获取程序源码,且无法分析用户和 Web 应用交互所造成的一些与环境相关的漏洞,无法保证漏洞检测的全面性^[11]。王丹等^[12]充分提取 XSS 漏洞注入点,模拟用户行为,触发 Web 页面交互点,制定规则并选择合适的攻击载荷验证漏洞存在,但检测效率较低。赵跃华等^[13]设计了一种结合黑盒测试与动态污点分析技术的 XSS 漏洞检测方案,依据 XSS 攻击向量模版,对不同注入点选择合适攻击载荷,并根据过滤规则集测试结果进行反过滤变换,但 XSS 源代码种子库覆盖面不全,导致检测准确度降低。吴子敬等^[14]定义一套可绕过服务器对 XSS 漏洞代码进行过滤的反过滤规则集,通过反过滤规则集对 XSS 代码进行变换,使其绕过过滤成功执行。但攻击向量模板和变换规则的不全面导致检测准确度下降。马富天等^[15]提出一种动态分析方法,候选元素与初始攻击向量组合,在特定输出点自动生成对应攻击向量,并自动调整向量优先级。倪萍等^[16]为指定规则构造的模糊测试用例设置初始权重,依据输出点类型,选择对应有潜力的攻击载荷并对其进行变异操作形成变异攻击载荷作为请求参数,对网站响应进行分析,调整元素的权重以便生成更加高效的攻击载荷,提高了检测效率。

用动态分析方法进行漏洞检测时,无需程序源码,但为降低误报率而增加攻击向量数量,则会导致服务器负荷增加,也降低了检测的效率。本文根据初始载荷的位置选择适当的绕过方式,形成攻击向量库,再对注入点进行模糊测试(Fuzzing),提高了检测效率。

为了提高检测效率,同时保证较高的准确度,本文基于抽象语法树(Abstract Syntax Tree,AST)解析和 Fuzzing,提出了 F-Xscan 漏洞模型。通过识别合法向量在返回页面中的位置,选择适当的攻击载荷,并与绕过规则进行适当组合,形成攻击向量库。

1 F-Xscan 漏洞模型

1.1 术语介绍

(1) AST 语法树。AST 是一种分层程序表示,以树状的形式表现编程语言的语法结构,树上的每个 AST 节点对应于源代码的一项^[17]。

(2) 语法分析器。语法分析器(Parser)通常作为编译器或解释器的组件出现,其作用是进行语法检查,并构建由输入单词组成的数据结构。本文采用了 AST 语法树对 HTML 和 JavaScript 进行解析,可准确判断回显的位置,便于下文攻击载荷的生成。

(3) 动态分析。动态分析主要是模拟攻击者攻击的过程,向注入点写入包含特定代码的脚本,分析服务器返回的页面响应内容,如果响应内容中存在特定的数据,则表示存在 XSS 漏洞。

(4) 模糊测试。模糊测试^[18]是一种基于缺陷注入的自动化软件漏洞挖掘技术,其基本思想与黑盒测试类似。模糊测试通过向待测目标软件输入一些半随机的数据并执行程序,监控程序的运行状况,同时记录并进一步分析目标程序发生的异常来发现潜在的漏洞。本文采取模糊测试的方式,对最终攻击向量进行确定。

1.2 F-Xscan 漏洞扫描器

本文提出的 F-Xscan 漏洞检测模型共分为 4 个部分,分别为探针载荷测试、初始攻击载荷生成、绕过规则选择和最终攻击向量生成等模块。总体模块设计如图 1 所示。

1.3 探针载荷

反射型 XSS 漏洞产生的原因是服务器端对用户输入恶意数据未过滤,直接“反射”给浏览器,即用户输入的参数直接输出到页面上。所以,首先要验证良性输入向量是否能在页面中回显。此处良性输入向量即为探针载荷。若确认页面回显,则当前页面可能存在反射型 XSS 漏洞。

探针载荷生成过程:

(1) 输入待检测的 URL,确认请求方法及请求参数。

(2) 生成良性载荷。选取随机 6 位数字、字母组合作为参数,并发送请求包。例如“kc4d9”。

(3) 根据 HTML、JavaScript、CSS 语法树解析,找到良性载荷的位置。根据载荷所处位置的上下文语法,形成对应初始攻击向量。

(4) 若良性载荷在页面中无回显,则不存在反射型 XSS。

良性攻击载荷回显流程如图 2 所示。

1.4 构造初始攻击载荷

有害数据的输入可能会触发浏览器的防御机制,

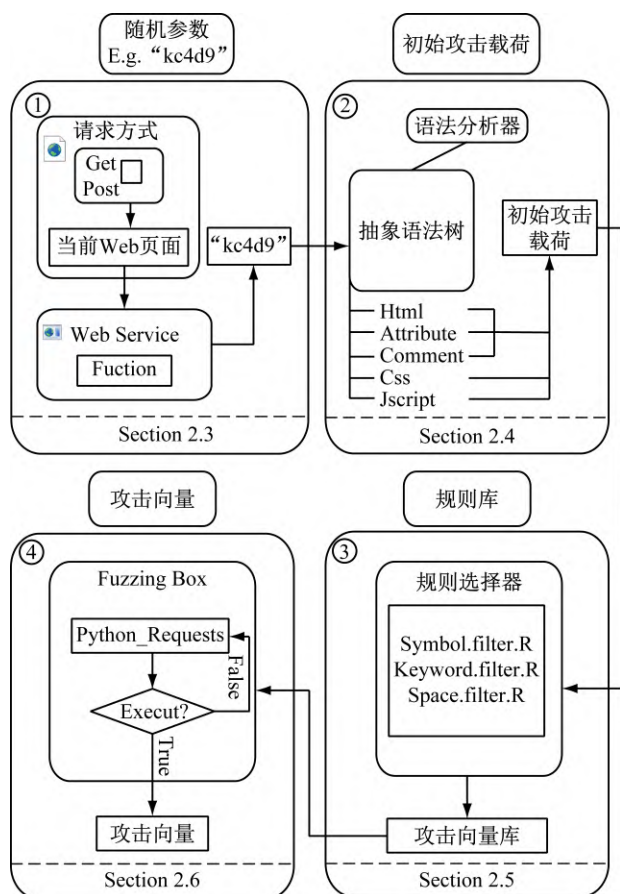


图 1 F-Xscan 漏洞模型

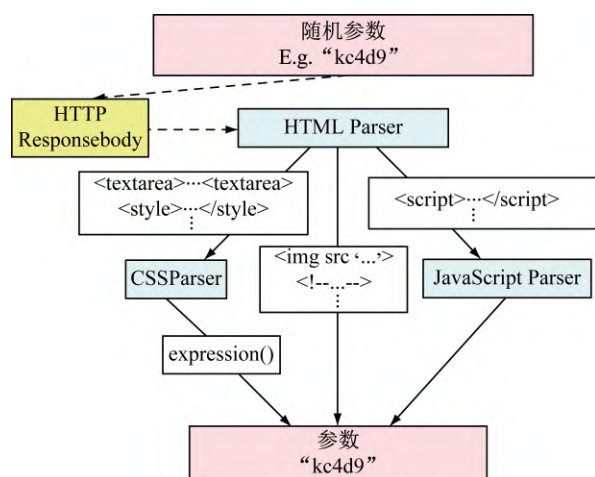


图 2 良性攻击载荷回显

而无法准确判断 XSS 漏洞是否存在^[19]。本文构造一种初始攻击载荷,以良性载荷为基础,构造载荷的逃逸来判定漏洞是否存在,避免过滤机制的触发使得结果更为准确。

根据图 2—探针载荷在页面中输出的位置,运用 AST 语法树解析、分析输出点类型,选择对应的标签、符号闭合。以探针载荷为初始值对应生成初始攻击载荷。该初始攻击载荷由回显页面对应上下文得出。预处理得到的初始攻击载荷更符合页面的攻击语法规则,使得攻击方案更优,减少后续不必要的模糊测试开销。在保证准确率的同时,提高了效率。

XSS 初始攻击载荷生成规则如下:

- (1) 确认探针载荷回显后,分析回显页面结构。
- (2) 根据回显位置,选择适当的逃逸技术和逃逸行为组成初始攻击载荷。

初始攻击载荷生成规则见表 1,逃逸技术及行为见表 2。

表 1 初始攻击载荷生成规则

载荷回显位置	逃逸方法	逃逸行为
HTMLContext	E1	B1
HTMLAttribute	E2	B2
HTMLComment	E3	B1
JavaScript	E4, E5	B2
CSS	E6	B2

表 2 逃逸技术及行为

序号	逃逸技术及行为	示例
E1	HTML 标签逃逸	</tag> <script> alert(1) </script>
E2	HTML 属性逃逸	" autofocus onfocus = alert(1)
E3	HTML 注释逃逸	--> <script> alert(1) </script>
E4	JavaScript 注释逃逸	/* /* */prompt(1) //
E5	JavaScript 表达式链接逃逸	"-alert(1) or ; prompt(1)
E6	CSS 样式表逃逸	expression(alert(1))
B1	标签行为	Body, img
B2	事件行为	onclick, onkeyup

例如可控点位于 HTMLAttribute:

```
<textarea>
  <div>
    <imgsrc = "1">
  <div>
</textarea>
```

首先发送良性探针载荷“kc4d9”,确定值“1”可控。通过 AST 语法树解析知,可控值“1”位于 <textarea> </textarea> 标签之间和 标签的 src 属性中。由表 1、2 可知需先闭合 src 属性,再闭合 <textarea> 标签。所以初始攻击载荷为 </textarea> "> <svgonload = alert(1) >。

1.5 绕过规则选择

越来越多的网站都制定了拦截规则,对用户提交的数据进行拦截和过滤。在一定程度上减少了用户的恶意输入,减少了漏洞的产生。但仍有不完全的过滤方式使得网站存在漏洞。针对 Web 应用中常见的过滤方式进行归纳总结,制定了自己的规则即过滤机制的逃逸技术。对 1.4 节初始攻击载荷进行变异,用于攻击向量库生成阶段。总体绕过对象分 2 种:过滤符号和过滤关键字(见表 3)。

制定绕过规则库(见表 4)。

1.6 攻击向量库生成

在 1.4、1.5 节中,制定了逃逸规则和绕过规则。本小节对初始攻击载荷进行上述 2 种类型规则的组合绕过,生成攻击向量库。攻击向量样本见表 5。

表 4 绕过规则库

序号	规则	示例	变换对象
R1	大小写混合	< scRipT > alert(1) < /scRipT >	E1-E6 ,B1-B2
R2	编码绕过(八进制编码、Unicode 编码等)	< imgsrc = 1 onerror = " \u0061\u006c\u0065\u0072\u0074(1) " >	E1-E6 ,B1-B2
R3	字符串干扰(畸形标签)	< img"'" > < script > alert(1) < /script > " >	E1 ,E2 ,E3 ,B1
R4	双写绕过	< < script > alert(1) < < /script >	E1-E6 ,B1-B2
R5	反引号绕过"~"	alert1`	E1-E6 ,B1-B2
R6	变量赋值	a = alert; a(1)	E1-E6 ,B1-B2
R7	全角、半角字符混合	< img style = "a; expResSion(aler('1')) " >	E6
R8	敏感关键字拆分(Tab、回车、空格、注释符)	< script > a%0Ale/* xxx* /rt(1) < /script >	E1-E6 ,B1-B2
R9	"/"代替空格	< svg/onload = alert(1) >	E2 ,E5

表 5 最终攻击向量样本

序号	最终攻击向量样本
1	< /textarea > < oBje%0act/dATa = ' data: text/html; ; ; ; base64 ,PHNjcmlwdD5 hbGVydCgxKTWvc2 NyaXB0Pg = = ' > < /ObjECt >
2	`> < < ImG/sR%0DC = javascript:alert('XSS') >
3	--> < < ScRIPt > a%0Ale/* Sw* /rt1`< < /scRiPT >
4	/* /* * / < fOrm > < texTARea/ /onkeyup = "\u0061\u006C\u0065\u0072\u0074(1) "`> //
5	a = alert; < imG/STy%0Ale = "z; expResSion(4 '1')) " >

表 5 罗列了部分生成的攻击向量。以往对攻击向量的选取,普遍采用随机选择的方法。虽然其单个选择时间短,但随机性较大,易造成无目的的选取。使其效率及准确率降低,造成检测效果的不理想。

本文在初始攻击向量的基础上,添加绕过规则,随机生成最终攻击向量。根据网站的过滤情况自动调整绕过规则的选取,使其更快速准确地检测出漏洞。对传统的随机产生向量的方法进行了补足。

2 实验结果与分析

为客观、有效地评估本文提出的反射型 XSS 漏洞检测模型,搭建了 2 个 XSS 漏洞检测平台,并与其他 Web 漏洞检测模型对比分析,得出最终结论。

2.1 实验环境及工具

环境 1 Damn Vulnerable Web Application (DVWA) 是基于 PHP/MySQL 环境编写的,用来进行安全脆弱性鉴定的 Web 应用,帮助网络安全专业人员在合法的环境中测试其专业技能。实验选取了其安全等级为 Low 和 Medium 时做对比测试。

环境 2 该站点是以 PHP 语言开发,搭载 MySQL 数据库的自行搭建的博客站点。

表 3 绕过对象

序号	绕过对象	对象示例	绕过规则
1	过滤符号	< > , " , ' , space 等	R2 ,R4 ,R5 ,R9
2	过滤关键字	img ,onclick 等	R1 ,R2 ,R3 ,R4 ,R6 ,R7 ,R8

Acunetix Web Vulnerability Scanner(AWVS) 是一款 Web 应用程序漏洞扫描工具。在其扫描并发现漏洞后,将模拟黑客攻击过程,在可输入数据的地方自定义脚本,尝试不同输入组合。实验选取的版本为 AWVS 11。

Burp Suite 是用于攻击 Web 应用程序的集成平台,开启代理模式后,将作为一个在浏览器和目标应用程序之间的中间人,允许拦截、查看和修改在 2 个方向上的原始数据流。实验选取的版本为 Burp Suite V1. 7. 31。本机环境及配置见表 6。

表 6 本机环境及配置

本机环境	配置
操作系统	Windows10
脚本解释器	Python3. 8. 0
内存	16G
处理器	Intel Core i7-7820HK
硬盘空间	1T

2.2 实验结果与分析

设置对比试验,将模型与 2 款著名 Web 漏洞扫描软件做对比,验证了本文模型的检测效率和准确率。

检测准确率和误报率,见表 7。

表 7 博客环境检测结果

工具	总漏洞数	检测出漏洞数	准确率/%	误报数	误报率/%
Burp Suite		24	80.0	2	6.7
AWVS	30	25	86.6	1	3.3
F-Xscan		27	90.0	1	3.3

为排除爬虫对检测时 HTTP 请求次数的干扰,本文只统计反射型 XSS 检测时各模型选取载荷测试时的请求,总共对 30 个注入点进行了测试。对博客环境 HTTP 平均请求次数测试如图 3 所示。

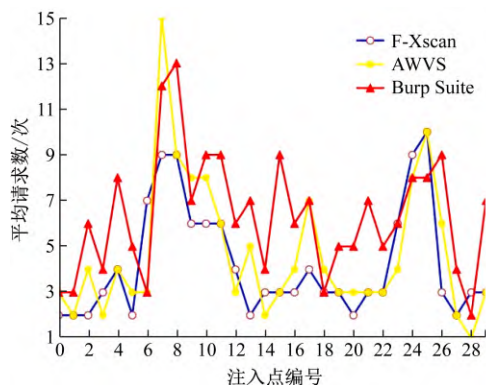


图 3 博客环境 HTTP 平均请求次数

对 DVWA 环境 HTTP 平均请求次数测试见表 8。

表 8 DVWA 环境 HTTP 平均请求次数表

方法	请求次数	
	Low	Medium
Burp Suite	3	5
AWVS	2	2
F-Xscan	2	2

由表 7、8 及图 3 所示,与传统的 Web 漏洞检测模型 Burp Suite 和 AWVS 相比,针对探针载荷在输出页面上上下文点的类型,选取适当绕过规则形成攻击向量,该方法在一定程度上减少了 HTTP 请求,降低了页面负载。同时根据上下文类型可更好地确定漏洞是否存在,具有更高的准确率,同时保持了较低的误报率。

3 结 语

针对当前反射型 XSS 漏洞检测效率较低,误报率较高的问题,提出了一种基于 AST 分析和 Fuzzing 的反射型 XSS 漏洞识别模型。利用 AST 语法树分析在请求后的 Web 页面中,探针载荷的回显位置,选取合适的初始攻击载荷。再根据该 Web 页面过滤情况,结合绕过规则,生成攻击向量库,对可疑 Web 注入点进行 Fuzzing。仿真结果表明,本文方法在检测效率较高的同时,有着较低的误报率。

参考文献(References):

[1] Cui Y, Cui J, Hu J. A survey on XSS attack detection and

prevention in web applications [C] // ICMLC 2020: 2020 12 th International Conference on Machine Learning and Computing. New York, NY, USA: ACM, 2020: 443-449.

[2] Sivanesan A P, Mathur A, Javadi A Y. A google chromium browser extension for detecting XSS attack in HTML5 based websites [C] // Proceedings of the 2018 IEEE International Conference on Electro/Information Technology. Piscataway: IEEE, 2018: 302-304.

[3] Wassermann S G, Su Z. Sound and precise analysis of web applications for injection vulnerabilities [C] // Conference on Programming Language Design and Implementation. San Diego: ACM, 2007: 2-41.

[4] 郭锡泉, 陈香锡. 强化网络安全和安全情报意识, 共筑网络安全防线——基于 OWASP 和 CNCERT 相关项目的分析 [J]. 网络安全空间安全, 2020, 11(2): 66-74.

[5] Budianto E, Jia Y, Dong X, et al. You can't be me: enabling trusted paths and user sub-origins in web browsers [C] // International Workshop on Recent Advances in Intrusion Detection. Cham: Springer International Publishing, 2014: 150-171.

[6] Fanning M C, Guerrero N. Iterative static and dynamic software analysis: US. Patent 7,975,257 [P]. 2011-07-05.

[7] 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): 62-72.

[8] 陆 申, 左志强, 王林章. 静态程序分析并行化研究进展 [J]. 软件学报, 2020, 31(5): 1243-1254.

[9] Ibérica M, Neves N, Correia M. Detecting and removing web application vulnerabilities with static analysis and data mining [J]. IEEE Transactions on Reliability, 2016, 65(1): 54-69.

[10] 邱子谨. 基于静态分析技术的 PHP 代码自动化缺陷检测工具的研究与设计 [D]. 北京: 北京邮电大学, 2018.

[11] 谷家腾, 辛 阳. 基于动态分析的 XSS 漏洞检测模型 [J]. 计算机工程, 2018, 44(10): 34-41.

[12] 王 丹, 刘 源, 赵文兵, 等. 基于用户行为模拟的 XSS 漏洞检测 [J]. 大连理工大学学报, 2017, 57(3): 302-307.

[13] 赵跃华, 吴东耀. 一种 XSS 漏洞检测系统分析与设计 [J]. 软件导刊, 2019, 18(3): 162-167.

[14] 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.

[15] 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.

[16] 倪 萍, 陈 伟. 基于模糊测试的反射型跨站脚本漏洞检测 [J]. 计算机应用, 2021, 41(9): 2594-2601.

[17] 黄晓伟, 范贵生, 虞慧群, 等. 基于重子节点抽象语法树的软件缺陷预测 [J]. 计算机工程, 2021, 47(12): 230-235, 248.

[18] Duchene F, Groz R, Rawat S, et al. XSS vulnerability detection using model inference assisted evolutionary fuzzing [C] // Proceedings of 5 th IEEE International Conference on Software Testing, Verification and Validation. Montreal, Washington: IEEE Computer Society, 2012: 815-817.

[19] Antunes J, Neves N, Correia M, et al. Vulnerability discovery with attack injection [J]. IEEE Transactions on Software Engineering, 2010, 36(3): 357-370.