基于 PhantomJS 的跨站脚本检测方法研究

黄细标[1]
HUANG Xibiao

摘 要 针对现有动态检测跨站脚本方法中存在传统爬虫不能爬取由 JavaScript 异步加载的网页数据等问题，提出一种基于 PhantomJS 的网络爬虫的跨站脚本检测方法，更加全面地获取网站页面与漏洞注入点。通过综合考虑字符串长度、字符类型等因素对攻击向量进行了优化，以绕过服务端的过滤机制，并提出一种基于自动化攻击向量与合法向量生成算法，在攻击过程中动态生成攻击向量与合法向量，以提高挖掘漏洞的准确性。依据所提出的检测方法，实现了一款跨站脚本漏洞检测工具（PhantomJS-based XSS scanner，PXS），并通过实验结果验证了所提出的方法能够有效发掘漏洞。

关键词 跨站脚本攻击；PhantomJS；攻击向量；合法向量

doi：10.3969/j.issn.1672-9528.2024.06.042

0 引言

多年来，跨站脚本攻击（cross-site scripting，XSS）一直是 Web 安全漏洞中最具风险的漏洞之一 [1]。根据 OWASP 在 2021 年公布的十大安全漏洞，注入攻击（Injection）高居第三位，跨站脚本攻击发生率依旧较高，仍然给 Web 应用程序造成安全威胁。

跨站脚本攻击是指攻击者通过将不同形式的恶意脚本代码注入到 Web 应用程序中 [2]，代码在 Web 浏览器中执行，它会更改网页，使用户在访问受到感染的页面时被攻击。跨站脚本可用于解决窃取敏感信息、劫持用户会话、破坏服务器、攻击系统的完整性等问题。跨站脚本攻击的目标是窃取客户端 cookie 或任何其他用于向网站认证客户端的敏感信息，如用户名和密码等信息，用于绕过访问控制等操作。

跨站脚本攻击方式通常分为三类：基于 DOM 的 XSS、反射型 XSS、存储型 XSS。其中，存储型 XSS 的影响最为严重。Web 应用程序出现 XSS 原因在于未对用户输入的数据进行有效的过滤。跨站脚本攻击涉及三方：攻击者、客户端和网站。攻击者通过分析网页信息，找到应用中的漏洞注入点，并伪造不可信的数据对注入点进行攻击。因此，在正式运行 Web 应用前，使用漏洞检测工具对其进行扫描，可以帮助网站管理员发现可能存在的漏洞，提升安全等级。

目前国内外主流的 XSS 漏洞检测工具主要有 AWVS、AppScan、Paros Proxy 等 [3]，但是现有的漏洞检测工具在漏洞扫描的完整性、准确率方面依然存在问题，特别是针对网页中使用 Ajax/JavaScript 异步加载数据的情况，容易出现漏扫情况。

1. 福建开放大学 福建龙岩 364000

1 相关工作

针对传统的网络爬虫无法获取并分析 Web 应用程序页面中由 Ajax/JavaScript 异步加载的数据，容易导致页面数据及注入点的遗漏，并且在判断漏洞攻击是否成功的过程中会出现少报、漏报现象，本文提出采用基于浏览器内核 PhantomJS[4] 的爬虫来抓取页面数据，可以获取到由 Ajax/JavaScript 异步加载的数据。

1.1 PhantomJS 简介

PhantomJS 是一个无界面的 Webkit 浏览器。虽然 PhantomJS 没有界面，但 DOM 渲染、JavaScript 运行、网络访问、Canvas/SVG 绘制等功能都很完备，在页面抓取、页面输出、自动化测试等方面有广泛的应用。

使用基于 PhantomJS 无头浏览器的网络爬虫，模拟用户在浏览器中对页面的操作，改进了传统静态爬虫对跨站脚本漏洞注入点、页面内容、检测覆盖率小的不足，尽可能多地获取网页中的注入点，扩大漏洞注入点的覆盖率。

1.2 攻击向量与合法向量

攻击向量 [5] 指的是能够触发 XSS 攻击的 JavaScript 代码，能够对应用程序造成安全威胁的脚本，也就是攻击者伪造不可信数据，而合法向量则是有效的、合法的应用程序数据。在检测漏洞中，攻击向量与合法向量的设计，对挖掘注入点有着至关重要的作用。

在实际应用中，大部分网站都会对客户端输入的内容进行一定程度的拦截、过滤与净化 [6]，因此在攻击向量的设计方面，可以根据 RSnake 总结分析的攻击向量库，并收集一些基于 HTML5 的攻击代码 [7]，通过综合考虑字符串长度、字符种类等因素，对 XSS Filter Evasion CheatSheet，添加适当的编码，对测试代码进行变换，合理使用模糊前缀，以绕

过服务器的过滤函数，缩短检测时间。

1.3 漏洞注入点

跨站脚本攻击的漏洞注入点是指可以注入攻击向量的位置。对于攻击者来说，找到页面中的输入框，就是找到漏洞注入点。如图 1 所示，从攻击者的角度来找注入点，查看网页的源代码。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>xss 漏洞演示—留言板</title>
</head>
<body>
    <form method="get" action="login.php">
        <textarea name="message"></textarea>
        <input type="submit" name="submit">
    </form>
    <p></p>
</body>
</html>
```

图 1 页面源代码

此时，源代码中的 <form> 就是一个潜在的注入点，攻击的目标点是 login.php，攻击的参数名称是 message，此时可以构造出攻击向量：<script>alert(‘xss’)</script>。如果没有对 XSS 进行有效防护，就可能会发生 XSS 攻击，如图 2 所示。

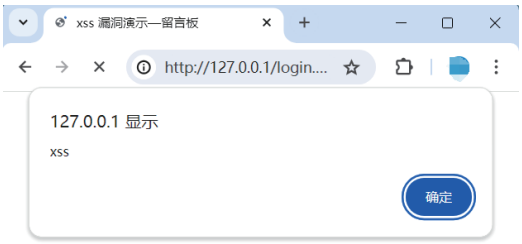

图 2 页面 xss 漏洞展示

注入点与 URL 有区别，每个 URL 对应着一个网页，而非每个网页中都含有注入点，但也有可能一个页面中包含着多个注入点。如何从众多复杂的页面中找到更多更全的注入点，是检测漏洞中的关键一步。

1.4 主要工作

本文主要完成了以下几项工作。（1）深入分析当前 Web 站点页面中异步加载数据的点，提出基于 PhantomJS 的爬虫。（2）设计并实现了一种自动生成攻击向量与合法向量的方法。（3）进行了对比实验，并对实验结果作出分析，总结了实验结果。

2 检测系统设计

检测系统分为三个模块：爬虫模块、注入模块、验证模块。其中，爬虫模块主要负责漏洞注入点的挖掘；在注入模块中，则会对所有注入点模拟攻击，并调整攻击向量的生成规则；最后在验证模块中，会验证发送的攻击向量是否达到预期的攻击效果，若是，则认定该注入点有漏洞。

2.1 爬虫模块

如何在大量的网页中找到隐藏的漏洞注入点是检测跨站脚本漏洞的关键之一。而且在网页内容日益丰富的今天，越来越多的网页内容是由 Ajax 和 JavaScript 生成的，人工检测漏洞注入点显然是不现实的，需要尽可能采用自动化的方法 [8]。

在 Web 页面中有许多超链接，页面之间可能会互相链接，在抓取过程中需要识别出相互链接的页面，并去除重复出现的 URL，避免陷入死循环。

捕获漏洞注入点的算法如下。

输入：待爬取 URL 队列

输出：已搜索的 URL 队列、漏洞注入点

算法：

（1）从待爬取 URL 队列中，取出一个 URL，使用基于 PhantomJS 的爬虫抓取该页面的所有数据。在抓取之前，爬虫会等待页面中所有数据加载完毕后，再将页面中的数据抓取存放到本地文件中，并将此 URL 加入已搜索队列中。

（2）分析页面中存在的 URL，存储获取到的 URL，采用 Bloom Filter 算法高效率地将页面中重复的 URL 去重。若不在已搜索队列和待搜索队列中，则加入到待爬取 URL 队列中。

（3）注入点捕获：提取页面中所有的 Form 表单元素，对 Ajax 类的注入点进行爬取，对爬取到的注入点，进行去重操作。

（4）JavaScript 文件解析：对页面代码中的 JavaScript 代码及 JavaScript 所添加的事件进行提取，并将提取到的 JavaScript 代码传递给 PhantomJS 的 JavaScript 引擎模块编译执行，解析输出的数据，并解析数据中的 URL 及注入点。

（5）循环以上步骤，直到待爬取 URL 队列为空。

在爬虫模块中，采用广度优先搜索策略，以此能够尽可能地覆盖每一个网页，一层一层地遍历，直到出现的 URL 都是重复的，在维护已经追踪访问过的节点时会比深度优先策略更简单。

另外，本系统启用多线程异步处理待爬取 URL 队列，开启多个线程并行处理爬取任务，有效提升注入点爬取的速度。

2.2 注入模块

在注入模块中，主要是对注入点进行攻击，攻击过程中根据参数相关信息构造攻击向量与合法向量，攻击完成后，记录响应信息，用于后续验证模块的验证流程。

首先，遍历每一个注入点，解析请求参数包括参数的名称、类型、长度等任何有关的信息，通过攻击向量与合法向量结合的方式来构造请求向量。例如在一个注入点中有多个请求参数，在构造向量的时候，并非将所有的向量都构造为攻击向量，因为 Web 服务端可能会对请求参数做过滤，若全部使用攻击向量，则很可能由于一个向量被服务端过滤导致

本次攻击失败，所以在每次发送攻击的时候，只取其中一个参数作为攻击向量，其它剩下参数则根据参数名称，构造出最符合要求的合法向量。例如参数名称为 age，检测系统则会判断该向量是年龄类型，生成的合法向量会是 0 ～ 100 的整数，这样可以提高注入成功概率。

在注入过程中，会给每一个注入点设置一个唯一的 ID，并且在给当前的注入点生成攻击向量时，攻击向量中会带着对应的注入点 ID，用于标识此攻击向量针对的注入点，避免攻击成功后，无法识别攻击的注入点。如图 3 所示，为对单个注入点攻击的流程图。

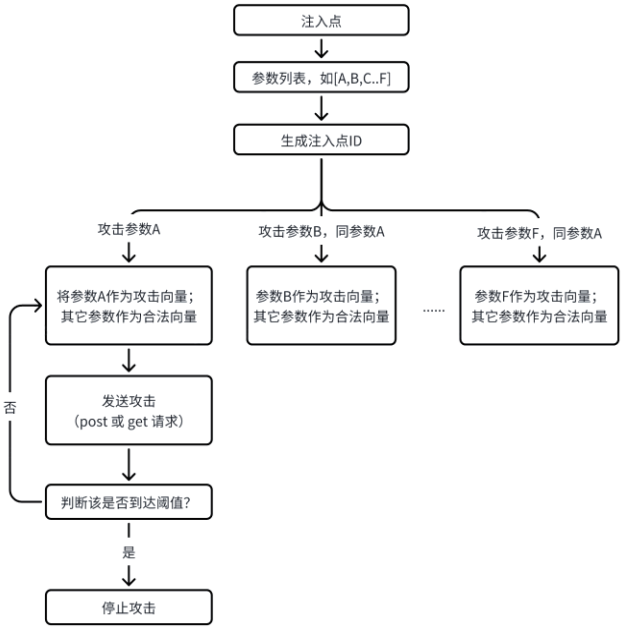

图 3 对单个注入点的攻击流程

2.3 验证模块

在验证模块中，主要是检测对发出去的攻击向量在页面中是否有响应的反馈，以及根据注入模块中记录的响应信息，来识别攻击是否成功，判断对应的注入点。

对于同一个页面，攻击前和攻击后没有发生变化的页面，进行二次遍历，则会浪费时间，降低检测效率。本文系统使用 md5() 函数对攻击前和攻击后的页面进行加密，然后匹配。若相同，则表明没有发现任何变化，无需进行二次遍历检测。在做攻击前后对比时，采用这种方式效率会更高。如图 4 所示。然而，传统的检测方法是将攻击向量注入到漏洞注入点中，然后分析响应后的页面进行漏洞检测的判断。但这种思路并不完善，原因是注入攻击向量后，攻击向量不一定只出现在服务器响应后的跳转页面中，也有可能出现在 Web 应用中的任何其他页面中。本文将分析页面范围扩大至注入点请求页面、响应后的页面及跳转页面，这样可以在检测中提高漏洞检测效率。在获取响应后的页面，跳转页面数据的时候，也会使用基于 PhantomJS 的爬虫来抓取响应信息，这一点同样用来避免遗漏那些异步加载的信息，使得捕获的响应信息更全，这样在验证漏洞是否攻击成功时也同样起着重要作用。

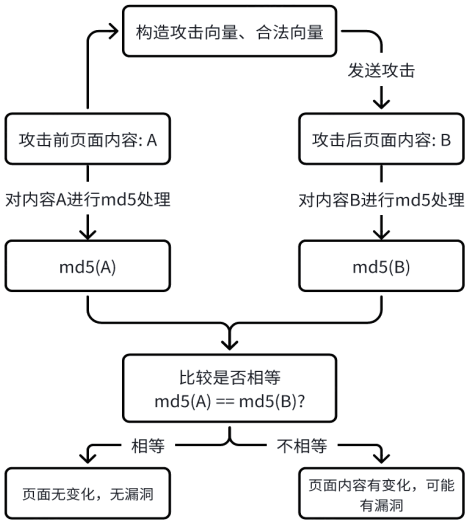

图 4 攻击前后页面数据对比方式

另外，在做验证时，本文提出了扩大验证范围的方法，不仅要验证同一个页面攻击前后的变化情况，而且会通过注入模块中记录下来的响应信息去找发送出去的攻击向量，再根据攻击向量去找对应的注入点，这样就能确定具体是哪个注入点存在 XSS 漏洞。流程如图 5 所示。此方法解决了存储型 XSS 将攻击向量保存在服务器端，但是展示在其它页面中的情况，有效扩大了验证范围，能够找到更多的漏洞。

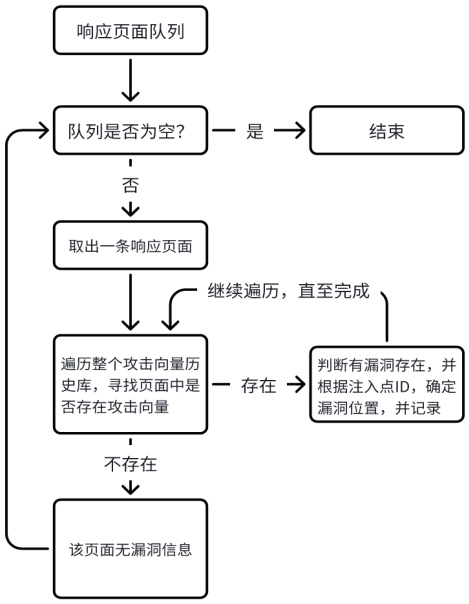

图 5 扩大漏洞验证范围流程

3 实验与结果分析

为了验证本文方法的有效性，采用 PHP 语言编程实现 PXS 检测系统，同时使用两款漏洞扫描工具 AWVS 和 WXS 对三个测试站点进行检测。AWVS 是一款功能全面的漏洞扫

描工具，可以检测包括但不限于 XSS 漏洞的工具；WXS 则为另外一款漏洞检测工具，是专门针对于跨站脚本漏洞的检测工具。

3.1 实验环境

由于本文使用的检测系统是分布式的，所以需要有一台控制机和两台工作机。本实验使用两台 PC 机：服务器和客户端机器。本文实验对 PC 机的配置没有特殊的要求，采用普通配置的计算机即可。

站点一为自建的 PHP 留言板网站，提供留言、文章评论功能；站点二为 OWASP 所提供的开源渗透测试漏洞网站，用于给安全研究人员做安全测试，包含了跨站脚本漏洞等常见 Web 应用漏洞；站点三为自建博客网站，由 JSP 语言开发。

实验结果如表 1 至表 4 所示，表 1 为站点一的检测结果，对比 PXS、AWVS、WXS 三个工具，结果的指标有抓取的 URL 数、注入点数，及发现的 XSS 漏洞个数。表 2 为站点 2 的检测结果，表 3 为站点 3 的检测结果。表 4 为 3 个检测工具检测站点 1、2、3 分别的耗时时间。

表 1 站点一检测结果

工具	URL/ 个	注入点 / 个	XSS/ 个
PXS	32	8	6
AWVS	30	5	3
WXS	28	4	3

表 2 站点二检测结果

工具	URL/ 个	注入点 / 个	XSS/ 个
PXS	266	22	12
AWVS	230	16	7
WXS	262	18	8

表 3 站点三检测结果

工具	URL/ 个	注入点 / 个	XSS/ 个
PXS	862	31	18
AWVS	855	29	15
WXS	822	27	11

表 4 耗时对比

工具	站点一	站点二	站点三
PXS	18 分 35 秒	52 分 47 秒	1 小时 47 分
AWVS	12 分 17 秒	43 分 52 秒	1 小时 23 分
WXS	14 分 06 秒	46 分 26 秒	1 小时 32 分

3.2 实验结果

实验结果可以看出，PXS 抓到的注入点是最多的，挖掘出来的 XSS 漏洞数量也是最多的。由此可知，基于 PhantomJS 模拟浏览器的爬虫，可以抓取到更多的 URL 页面信息，并且在解析执行 JavaScript 代码中也能够解析出更多的数据，可以挖掘出更多信息。

另外，从耗时时间对比上看，PXS 的耗时较长，原因在于基于 PhantomJS 去爬取页面数据，需要等待所有异步数据加载完毕，因此需要更多时间。

4 结语

通过对 Ajax 异步加载数据导致检测技术中出现漏报等问题，本文提出了基于 PhantomJS 内核的无头浏览器对页面中异步加载的数据进行捕获，并使用了自动化生成攻击向量与合法向量方式来构造请求参数，使模拟攻击变得更加灵活。最后与其它漏洞扫描工具做对比实验时，PXS 检测工具发现的漏洞注入点较多，并且检测到的漏洞数量较多。PXS 工具的缺点在于检测时间比较长，需要耗费更多的时间在抓取数据模块上，尽管使用了多线程来处理爬虫模块，但单机整体的资源有限。下一阶段的研究主要是考虑使用分布式任务调度方式来处理任务，进一步处理耗时过长的问题。

参考文献：

[1]ZHAO W, WANG D, DING Z.Review of XSS attack and detection on web applications[C]//International Conference on Computer Networks and Communication Technology. Paris: Atlantis Press,2017:798-804.

[2] 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.

[3] 赵鹏 , 冯增辉 , 林义勇 . 基于网络爬虫的 XSS 漏洞扫描 [C]// 中国高科技产业化研究会智能信息处理产业化分会 , 中国高科技产业化研究会信号处理专家委员会 . 第十一届全国信号和智能信息处理与应用学术会议专刊 . 凉山 : 西昌卫星发射中心 , 2017:5.

[4]BELTRAN A.Getting started with phantomJS[EB/OL].(2018-01-04)[2024-03-06].https://www.3pillarglobal.com/insights/beginning-with-phantomjs/.

[5]DONG G, ZHANG Y, WANG X, et al.Detecting cross site scripting vulnerabilities introduced by HTML5[C]//2014 11th International joint conference on computer science and software engineering. Piscataway: IEEE,2019:319-323.

[6]HANSEN R.XSS filter evasion cheat sheet[EB/OL].(2021-10-14)[2024-02-26].https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet.

[7]DUCHENE F, GROZ R, RAWAT S, et al.XSS vulnerability detection using model inference assisted evolutionary fuzzing[C]//2012 International Fifth Conference on Software Testing, Verification and Validation.Piscataway: IEEE, 2012: 815-817.

[8] 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.

【作者简介】

黄细标（1979—），男，福建上杭人，本科，讲师，研究方向：网络安全。

（收稿日期：2024-04-29）