Computer Science and Application 计算机科学与应用, 2019, 9(6), 1061-1071
Published Online June 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.96120

Hans汉斯

Research on Cross-Site Scripting Attack Defense Based on Proxy

Fen Yan, Caixia Guo, Tao Qiao

College of Information Engineering, Yangzhou University, Yangzhou Jiangsu
Email: 13040281105@163.com

Received: May 31st, 2019; accepted: Jun. 11th, 2019; published: Jun. 18th, 2019

Abstract

In the field of network security, cross site scripting is one of the most destructive security problems. This paper puts forward a kind of XSS defense framework based on agent after an in-depth study of XSS attack mode. The framework is deployed between the client and the server, and the framework analyzes and filters the data by intercepting the communication data between the client and server using the algorithm. The evaluation shows that it can not only effectively defense against XSS attacks, but also provide a good experience for the user through the release of legitimate tags.

Keywords

XSS, Defense, Web Security

基于代理的跨站脚本攻击防御研究

严　芬，郭彩霞，乔　涛

扬州大学信息工程学院，江苏　扬州
Email: 13040281105@163.com

收稿日期：2019年5月31日；录用日期：2019年6月11日；发布日期：2019年6月18日

摘　要

在网络安全领域中，跨站脚本攻击已经是最严重的安全问题之一。本文在深入研究跨站脚本攻击方式的基础上，提出了一种基于代理的跨站脚本防御框架。该框架部署在客户端和服务器端之间，利用算法对通信数据进行深度分析和过滤，从而对跨站脚本进行有效的防御。实验证明设计的框架不仅能对跨站脚本攻击进行有效的防御，而且可以对合法标签进行放行，为用户提供良好的访问体验。

文章引用: 严芬, 郭彩霞, 乔涛. 基于代理的跨站脚本攻击防御研究[J]. 计算机科学与应用, 2019, 9(6): 1061-1071
DOI: 10.12677/csa.2019.96120

关键词

跨站脚本攻击，防御，Web安全

1. 引言

跨站脚本攻击(Cross Site Scripting, XSS)是目前网络攻击中最流行的方法之一，它通过 Web 应用中的漏洞将恶意脚本注入到网页中[1]。针对跨站脚本攻击防御，国内外许多专家和学者都对其作了大量研究工作。

A. Guha 等人[2]提出了一种静态方法，对 JavaScript 脚本的 XMLHttpRequest 请求建立请求图，在服务器端监视用户发送的 XMLHttpRequest 请求，若不符合请求图则视为攻击。该方法可以防止普通跨站脚本攻击以及跨站请求伪造攻击(Cross-Site Request Forgery, CDRF)，但是局限于分析 XMLHttpRequest 请求，不能防御那些不向 Web 程序所在服务器发送的 XMLHttpRequest 请求的攻击，而且此方法需要修改服务器端 Web 程序源代码。

Noxes [3]通过制定一系列的规则，对用户的数据在客户端进行拦截或放行，从而对跨站脚本进行有效防御，但是随着近几年跨站脚本攻击技术的不断更新，基于编码的跨站脚本攻击已经能突破 Noxes 的拦截规则。

2011 年中国科学计算机网络入侵防范中心王夏莉提出基于行为的跨站脚本攻击检测，通过分析 JavaScript 命令代码的特征行为，对于危险的 HTTP 请求数据包放入隔离区，将其中可疑的请求操作转发给用户，由用户自身进行判断。一般用户无法判断是否存在危险，不适合大面推广[4]。

Likarish [5]使用网络爬虫工具去收集网络上的良性脚本和恶意脚本数据，提出了基于函数和代码的特征的机器学习算法。该算法通过深度的学习之后，可以对数据进行检测区分。但是，检测的准确度依靠机器学习的数据样本，对数据辨别的准确率还有待提高。

目前，对跨站脚本防御做的比较成功的是编码防御。其中，对编码技术总结的比较全面的是 OWASP 对跨站脚本攻击作出的一系列防御规则[6]。这种防御规则确保数据在前台展示的时候都经过了编码，有效预防了危险数据在 HTML 文件中被引用[7]。但是，这种规则在带来安全的同时，也限制了 Web 应用的部分功能，使得部分合法的标签缺少使用权限，从而降低客户端用户的体验。本着放开合法标签的目的，我们提出了一种基于代理的跨站防御方法。该方法通过对数据的深入拆分解析，准确识别出标签是否合法、数据中是否存在脚本以及数据安全。研究结果表明，该方法对于当前流行的跨站脚本攻击有良好的防御效果。

2. 基于代理的跨站脚本防御框架模型

我们构造的基于代理的跨站脚本防御框架模型如图 1 所示。防御模块由分析引擎和处理引擎两大部分组成。分析引擎的功能是对数据进行重编码、去除干扰和整理标签处理，为下一步的处理引擎提供预处理之后的数据以及数据基本特征。处理引擎的功能是对分析引擎传入的数据进行非法标签、非法事件和非法引用的处理，确认无风险后进行放行，并将此次处理记录日志。该框架模型能够对合法标签进行放行，为客户端提供多功能的访问体验通过正确识别 HTML 标签，对跨站脚本攻击进行有效的防御。

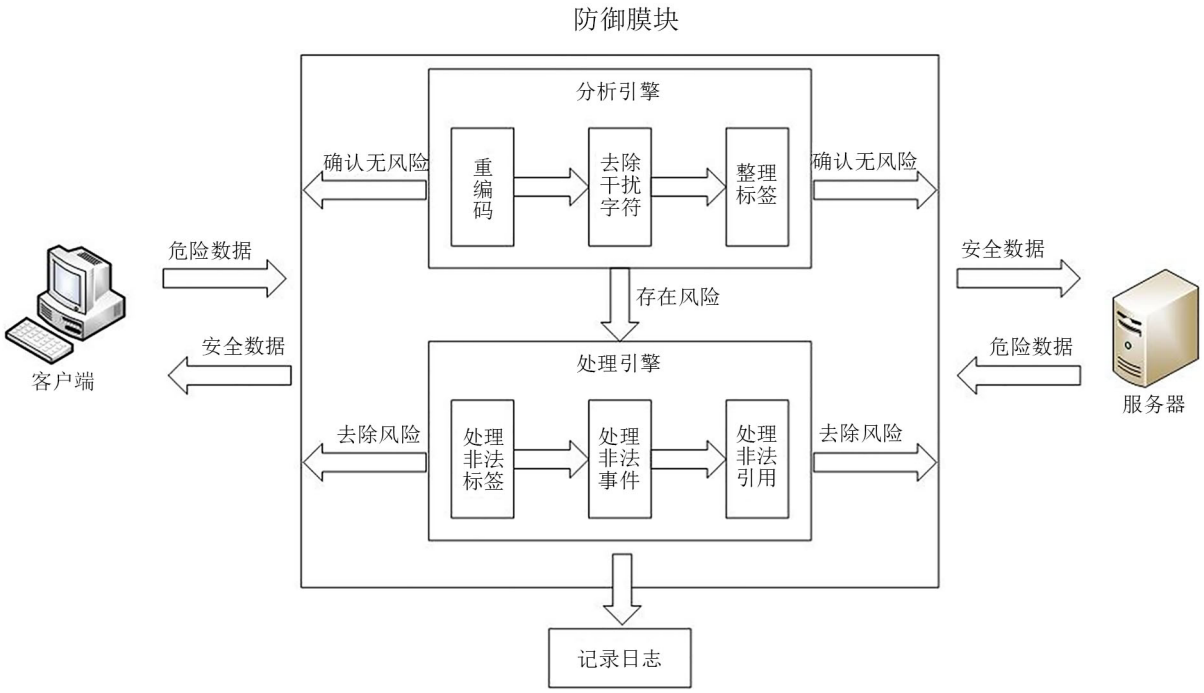

Figure 1. Agent-based cross-site scripting defense framework model
图 1. 基于代理的跨站脚本防御框架模型

3. 模型实现

3.1. 相关定义

定义 3.1 潜在风险数据：所有用户输入的数据只要满足以下特征中的任意一条，就被认为是潜在风险数据：1) 数据中存在 html 标签；2) 数据中存在一种特殊字符，如<、>、%、&、/、”、’、#等。

潜在风险数据需要送分析引擎以及处理引擎作进一步的处理。

定义 3.2 干扰字符：在用户输入的数据中，连续存在的空白符或被浏览器解释引擎自动忽略的字符被定义为干扰字符，比如注释符。

定义 3.3 标签特征(Tag Object)：对于数据之中存在的一个或多个完整标签，每个标签都有其固定的特征存在，包括标签名和标签属性。

定义 3.4 非法标签：非法标签包括用户自定义的标签、存在脚本的标签、内容结构不完整的标签和标签内容与标签名不符合的标签。

3.2. 分析引擎

分析引擎是数据传输至防御框架中经过的第一个模块。所有的数据经过分析引擎重编码、去除干扰字符和整理标签处理之后，被分为安全数据和潜在威胁数据(即存在 html 标签和特殊字符的数据)。对于安全数据直接放行，而潜在威胁数据将被传输至处理引擎做进一步处理。

1) 重编码子模块

重编码子模块主要依靠前缀特征如%、%u、\u、&等来判断字符串的编码类型，然后再对其进行统一的解码处理。

2) 去除干扰字符子模块

去除干扰字符子模块主要依靠字符特征来判断解码后的数据中是否存在干扰字符，然后对干扰字符

进行删除处理。

重编码子模块和去除干扰字符子模块的实现算法如算法 1 所示。

算法 1：重编码和去除干扰字符算法

Input: S

Output: PretreatmentS

1: CodingPattern←%,% u,\u,&

2: ##重编码

3: if S satisfy CodingPattern, then

4: if S is URLEncoding, then

5: S←URLDecode(S,'gb2312'),end if

6: if S is UnicodeEncoding, then

7: S←UnicodeDecode(S,'gb2312'),end if

8: if S is HexEncoding, then

9: S←HexDecode(S,'gb2312'),end if

10: if S is HTMLEncoding, then

11: S←HTMLDecode(S,'gb2312'),end if

12: ...continue decoding util there is no encoding

13: end if

14: ##去除干扰字符

15: NotePattern←/*…*/,<!--…-->

16: if S satisfy NotePattern, then

17: S←S.delete(Note)

18: if S contains Continuous blank character, then

19: S←S.replace(Continuous blank character,[Space])

20: PretreatmentS←S

21: return PretreatmentS

3) 整理标签子模块

整理标签子模块是分析引擎中最重要的工作，其主要工作包括：标签重排序、删除未知标签、标签属性解析。

标签重排序部分，根据标签的完整性将标签提取出来。标签分为完整标签和半完整标签，对于一个完整的标签，它的标志性结尾是“/”。检测完整的标签，记为 list{tag_1, tag_2, …, tag_n}；接着检测每个完整标签中是否存在完整的或半完整的标签，另外解析；直至标签中不夹杂标签，将这些后加入的标签加入 list 集中。

删除未知标签部分，准确定位标签的标签名，如果该标签不是标签库中的正常标签，则该标签直接定义为非法标签，进行丢弃处理。

标签属性解析部分，list 中的每个 tag 都有其属性，记为 tag = TagName + (Tag ArrributrName$_1$: TagAttributeValue$_1$) +….+(TagArrributrName$_n$:TagAttributeValue$_n$)。

整理标签子模块的实现算法如算法 2 所示。

算法 2：整理标签算法

Input: PretreatmentS

Output: FixedS,list{Tag}

1: ##标签排序
2: CompleteTagPattern←<…></…>, <…/>
3: CommonTagPattern←<…>
4: while PretreatmentS satisfy CompleteTagPattern, then
5: search the nearest CompleteTag,
6: list←CompleteTag,
7: find the next CompleteTag, until false, end while
8: for each CompleteTag in list
9: while CompleteTag satisfy CompleteTagPattern or CommonTagPattern,then
10: serch each nearest Tag
11: list←tag,
12: find the next Tag, until false, end while
13: end for
14: FixedS←newRules(list){taga+tagb+…+tagm}(下标均小于 list.length)
15: ##删除未知标签
16: for each tagi in list
17: find TagName of Tagi by Defination 3-3
18: TagName←Tagi
19: if TagName is not the correct HTMLTagName,then
20: delete the tag in the list,
21: delete the tag in the FixedS, end if
22: end for
23: ##标签属性解析
24: for each tagi in list
25: divide each TagAttribute of Tagi by Defination 3-3
26: for each TagAttributei in tagi.TagAttribute
27: find TagAttributeName of TagAttributei by Defination 3-3
28: find TagAttributeValue of TagAttributei by Defination 3-3
29: list{TagAttribute}←TagAttributeName: TagAttributeValue
30: end for
31: tagi←TagName+list{ TagAttribute }
32: end for
33: return FixedS,list{Tag}
34: find TagAttributeValue of TagAttributei by Defination 3-3
35: list{TagAttribute}←TagAttributeName: TagAttributeValue
36: end for
37: tagi←TagName+list{ TagAttribute }
38: end for
39: return FixedS,list{Tag}

3.3. 处理引擎

处理引擎通过标签特征识别数据是否存在威胁，包括三个子模块，每个子模块都采用了不同规则集分别对非法标签、非法事件和非法引用进行处理。单纯使用黑名单或者白名单算法，检测效果会不佳，因此，在以下三个子模块中，结合了黑名单和白名单共同对进行数据处理。

1) 处理非法标签子模块

数据在经过算法 2 处理之后传入处理引擎，同时数据的标签特性也传入了处理引擎。

处理非法标签子模块主要采用黑名单算对标签名进行处理。设黑名单中的有限元素集合为 List{BlackTag} = {BT_1,BT_2,…,BT_n}，此时传入处理引擎的数据为整理标签算法处理后得到的 FixedS 和 List{Tag}。根据 List{Tag}得到数据的有限标签名集合 List{TagName} = {TN_1,TN_2,…,TN_m}。对于 list{TagName}中的每一个元素 TN_i，如果 TN_i∈List{BlackTag}，那么删除元素 TN_i 对应的 Tag，同时对 FixedS 中对应的标签做删除处理。

表 1 列举了部分自定义的黑名单标签来说明黑名单中的规则。

Table 1. Custom blacklist labels (partial)
表 1. 自定义的黑名单标签(部分)

HTML 黑名单标签	描述
<a>	该标签定义超链接，由于外部链接也可以作为脚本，因此属于极危险标签，应禁止
<script>	该标签用于定义客户端脚本，可直接在文档中进行脚本插入，因此属于极危险标签，应禁止
<iframe>	该标签会建立一个 html 页中的框架，由于该标签的 src 属性可以插入脚本，可以引起跨站脚本漏洞，因此属于极危险标签，应禁止
<input>	该标签为网页中的输入框，由于其可以添加脚本的属性比较多，且网页中随便插入该标签会引起网页排版混乱，因此属于极危险标签，应禁止

2) 处理非法事件子模块

处理非法事件子模块主要采用白名单算法，针对标签属性中的标签属性名进行处理。根据每个标签不同的特性，开放部分安全属性供用户使用，只有白名单允许的标签属性才能存在于标签中。设此白名单中的有限元素集合为 List{WhiteAttribute} = {WA_1, WA_2, …,WAn}，根据 List{Tag}得到标签 Tag_i 的有限标签属性名集合 List{AttributeName}={AN_1,AN_2,…,ANm}。对于 AN_i，只有当 AN_i∈List{WhiteAttribute}时，该 AN_i 对应的标签属性予以保留，其他标签属性进行删除。同时，对 FixedS 中对应的标签做删除相关标签处理。

表 2 列举了一些自定义的白名单标签来说明白名单中的规则。

Table 2. Custom whitelist labels (partial)
表 2. 自定义的白名单标签(部分)

HTML 标签	白名单属性	说明
<b>	class, id, title, dir	该标签定义粗体文件，允许的属性都不允许加载 script 脚本
<img>	src, id, height, width, title, alt	该标签向网页中嵌入图像，允许的属性除 src 外，都不存在加载的 script 脚本风险。而对 src 属性在下一个模块做进一步的检测
<video>	src, id, width, title, height, loop	该标签定义视频，在网页中嵌入视频流，允许的标签除 src 外，都不允许加载 script 脚本的风险。而对 src 属性在下一模块做进一步的检测说明
<bgsound>	src, id, width, title, height, loop	该标签定义视频，在网页中嵌入视频流，允许的标签除 src 外，都不允许加载 script 脚本的风险。而对 src 属性在下一模块做进一步的检测说明

3) 处理非法引用子模块

处理非法引用子模块主要针对标签中的标签属性值，本子模块主要对恶意链接以及是否存在精心构造的脚本进行检测。关于链接的检测，主要分为两类：第一，如果一个链接为网站应用本服务器上的链接，检测到的地址与服务器域名相同，则认为该链接安全，不对其进行处理；第二，如果该链接为跨域链接，则检测该链接是否符合标签允许值，如标签<img>属性中的 src 值允许为第三方网站的 URL。但是，对于 src 属性，则必须包含后缀为.jpg，.gif，.bmp 等图像文件，否则认为其危险。同样地，对于<video>标签和<bgsound>标签，都必须检测 src 的值是否与标签名相符合。另外，属性值中不能包含常见的脚本恶意字符，比如 script，javascript，expression 等恶意字符。

处理引擎模块的具体实现如算法 3 所示。

算法 3：处理引擎算法

Input: FixedS, list{Tag}

Output: SecurityS

```
##处理非法标签
    define List{BlackTag}
    for each tagi in list{Tag}
      for each BTj in List{BlackTag}
          if tagi.TagName == BTj, then
              delete tagi in list{Tag}
              delete tagi in FixedS, end if
      end for
    end for
##处理非法事件
    define List{WhiteAttribute}
    for each tagi in list{Tag}
        for each TagAttributej in tagi.TagAttribute
          if TagAttributej.TagAttributeName is not in List{WhiteAttribute},
then
                delete tagi.TagAttributej in tagi
                modify tagi in list{Tag}
                modify tagi in FixedS, end if
          end for
      end for
##处理非法引用
    DangerPattern←script, javascript, vbscript…
      for each tagi in list{Tag}
      for each TagAttributej in tagi.TagAttribute
        if TagAttributej.TagAttributeValue is URL
          Url←TagAttributej.TagAttributeValue
              if Url!=System.domain
                  checkout Url
```

```
28:                    if Url is illegal
29:                    delete tagi in list{Tag}
30:                    delete tagi in FixedS, end if
31:                end if
32:            end if
33:        end for
34:    end for
35:  SecurityS←FixedS
36: return SecurityS
```

4. 实验结果与分析

为了测试和验证本文提出的基于代理的跨站脚本防御方法，我们将自主研发的 Web 应用系统(“基于 Struts 框架的学生报到管理系统”，以下简称“报到系统”)作为检测对象，并对单纯使用 OWASP 规则和使用本文方案中所设计的规则进行了比较。实验主要完成以下目标：

1) 将攻击数据和普通数据混合，在 Web 应用系统中进行输入，测试防御框架的检测和防御能力。

2) 将防御框架部署在实际的 Web 应用系统，验证本文防御框架的有效性和防御效果。

4.1. 本文防御算法的检测防御能力验证

为了证明本章提出的防御算法的有效性，我们采用了 XSS Cheat Sheet 中的数据和普通数据混合进行测试攻击，选取了 XSS Cheat Sheet 中部分数据作为实验展示。其中，表 3 是原始测试数据表，表 4 为原始数据经过本文防御框架中分析引擎处理后的数据，表 5 为数据经过本文防御框架中处理引擎处理后的数据，表 6 为所有数据测试结果集。

Table 3. Partial test data in xss_payload
表 3. xss_payload 中部分测试数据

序号	原始测试数据
1	%3Cscript%3Ealert(‘XSS’)C/script%3E 你好
2	<img src=”javas<!—xss-->cript:alert(‘XSS’)”>hi!
3	 <xxxonerron=”alert(‘xss’)”>
4	Hello!<BGSOUND src=”javascript:alert(‘XSS’);”onlOAd=”alert(‘xss’)”>
5	<style><img src=”</style><img src=’a.pig’onerror=’alert(xss)’”>

Table 4. Data processed by the analysis engine
表 4. 经过分析引擎处理后的数据

序号	经过分析引擎处理后的数据	标签名	标签属性
1	<script>alert(‘XSS’)</script>你好	script	{null}
2	<img src=”javascript:alert(‘XSS’)”>hi!	img	{ “src”:”javascript:alert(‘XSS’)}
3	<img src=”javascript:alert(‘XSS’)”> 	img img	{ “src”:”javascript:alert(‘XSS’)} {“src”:”a.gif”}
4	Hello!<BGSOUND src=”javascript:alert(‘XSS’);” onlOAd=”alert(‘xss’)”>	bgsound	{ “src”:”javascript:alert(‘XSS’)”,” onload”:”alert(‘xss’)”}
5	<style><img src=”</style><img src=’a.pig’onerror=’alert(xss)’”>	style img	{null} {“src”:”a.pig”,”onerror”:”alert(‘xss’)”}

如表 4 所示，原始数据经过分析引擎处理之后传入处理引擎的数据主要包含两部分：一种是经过重编码、去除干扰字符并且重新排序的数据，另一部分是数据的特性，包括标签名和标签属性。从实验结果可以看出，分析引擎对于数据的标签分析是准确的，这对于处理引擎是否能准确处理具有决定性的作用。

Table 5. Data processed by the processing engine
表 5. 经过处理引擎处理后的数据

序号	经过处理引擎处理之后的数据	处理原因	
1	你好	Script	标签为黑名单标签
2	hi	Img	标签中src值非法
3		Img 标签中 src 值非法	
4	Hello!	Bgsound	标签为黑名单标签
5		Style	标签为黑名单标签

如表 5 所示，我们通过判断标签名，放开了部分合法标签，合法的标签在检查其标签属性是否合法之后，留下了安全的标签。

实验共选取 200 条攻击数据、200 条普通数据，以及 200 条攻击数据与普通数据混合的数据，总共 600 条数据进行测试。测试结果如表 6 所示。

Table 6. The detection and defense ability test of this algorithm
表 6. 本文算法检测防御能力测试

输入类型	数量	处理数量	符合预期结果数量	检测率	是否存在跨站脚本
跨站脚本	200	200	198	99%	否
不含特殊字符的普通数据	50	0	50	100%	否
含特殊字符的普通数据	100	100	100	100%	否
含标签的普通数据	50	50	50	100%	否
混合标签	200	200	197	98.5%	否

表 6 可以看出，虽然少量数据与预期结果存在误差，但是所有的数据在经过处理之后均已不存在跨站脚本，说明本文方法可以成功对跨站脚本进行防御。而产生这种误差的主要原因是在重新对标签进行排序分析的时候产生的。由于存在采用关键字混淆编码技术的跨站脚本数据，在还原数据的过程中，可能会出现少量数据中位置顺序错误的问题。而这部分处理对于跨站脚本攻击的检测是必不可少的。可以认为，本文提出的防御框架可以达到比较理想的检测和防御效果。

4.2. 本文防御算法的有效性和防御效果验证

如图 2 所示，原有的报到系统允许评论中引用第三方网站的图片。但是，为了防止跨站脚本攻击，使用了 OWASP 提供的防御规则[8]，客户端用户访问报到系统时时出现了一段编码后的标签代码，大大违背了网站应用的本意。而本文方法考虑了在尽可能地挖掘数据潜在危险数据的基础上，尽可能地放开合法标签，按照算法 1 和算法 2 提供的规则进行处理，确保数据的最大完整性和降低数据危险的前提下，将数据返回给客户端。

Figure 2. Reporting system using the OWASP coding framework
图 2. 使用 OWASP 编码框架的报到系统

图 2 所使用到的防御框架是 OWASP 提供的编码框架，实验测试表明，该防御框架对防御跨站脚本攻击有良好的效果，但是，该框架却限制了 Web 应用系统的部分功能。如图 3 中打标记处所示，网站的本意是允许用户在备注自由添加安全的标签，但由于编码的规则，标签全部以编码字符的形式存在，给系统的正常使用带来一定的影响。而图 3 所示，该系统使用了本文防御算法后，标签功能被保留了，并且，存在脚本的标签也被成功处理。

Figure 3. Reporting system using the defense algorithm
图 3. 使用本文防御算法的报到系统

5. 结束语

在保证数据完整性与安全性的前提下，本文提出了一种基于代理的跨站脚本漏洞的防御方法，并进行了实验分析和验证。研究表明，本文算法不仅能有效防御跨站脚本攻击，而且还可以放开传统的 Web 应用对于标签的使用限制，丰富了客户端用户的体验，同时，一定程度上减少了误报率，能够很好的对跨站脚本进行防御。

基金项目

“江苏省博士后科研资助计划”项目“信息安全公共服务平台及网络攻击检测技术的研究与应用”(NO: 1501106C)。

参考文献

[1] 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.

[2] Guha, A., Krishnamurthi, S. and Jim, T. (2009) Using Static Analysis for Ajax Intrusion Detection. *Proceedings of the 18th International Conference on World Wide Web*, Madrid, 20-24 April 2009, 561-570. https://doi.org/10.1145/1526709.1526785

[3] 王夏莉, 张玉清. 一种基于行为的 XSS 客户端防范方法[J]. 中国科学院大学学报, 2011, 28(5): 668-675.

[4] Likarish, P., Jung, E. and Jo, I. (2009) Obfuscated Malicious Javascript Detection Using Classification Techniques. *International Conference on Malicious and Unwanted Software*, Montreal, 13-14 October 2009, 47-54. https://doi.org/10.1109/malware.2009.5403020

[5] 鲍泽民, 王根英, 李娟. 跨站脚本攻击客户端防御技术研究[J]. 铁路计算机应用, 2015(7): 17-20.

[6] Arzt, S., Kussmaul, T. and Bodden, E. (2016) Towards Cross-Platform Cross-Language Analysis with Soot. *The 5th ACM SIGPLAN International Workshop*, Santa Barbara, 14 June 2016, 1-6. https://doi.org/10.1145/2931021.2931022

[7] 刘达. 通过 HTML 编码防御 XSS 跨站脚本攻击的研究[J]. 信息安全与技术, 2016, 7(6): 23-24.

[8] OWASP (2017) XSS (Cross Site Scripting) Prevention Cheat Sheet. http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet