文章编号：1000-8608(2017)03-0302-06

基于用户行为模拟的XSS漏洞检测

王 丹*，刘 源，赵文兵，付利华，杜晓林

（北京工业大学 计算机学院，北京 100124）

摘要： 为改进XSS漏洞检测系统中对复杂网页漏洞注入点发现不够充分、动态地分析目标站点的响应信息不足等问题，改善XSS漏洞检测系统的注入点提取、攻击测试向量生成和响应分析等，提出了基于用户行为模拟的XSS漏洞检测方法.通过分析网页结构找到多种非格式化注入点，并通过综合考虑字符串长度、字符种类等因素对攻击向量进行了优化，以绕过服务器的过滤函数，缩短漏洞测试所用的时间.测试结果表明所提方法提高了漏洞注入点的检测覆盖率，提升了XSS漏洞的检测效果.

关键词： XSS漏洞；检测；Headless浏览器；Ghost.py

中图分类号： TP308 **文献标识码：** A **doi：** 10.7511/dllgxb201703013

0 引 言

近年来，随着网络的广泛使用，网络安全问题也日益突出. OWASP（Open Web Application Security Project）公布的2013年十大网络应用安全风险中，跨站脚本漏洞XSS（cross site scripting）名列第3，已成为当前各类网站需共同面对的常见的安全风险之一. XSS漏洞攻击是一种注入型的恶意攻击方式，它通过某种手段向网页中嵌入恶意的脚本代码，用户正常浏览该网页时，这些嵌入到网页的恶意脚本代码就会被客户端浏览器自动解析执行，造成对正常用户的攻击，引发挂马、钓鱼、盗取用户数据、劫持用户网络等恶意行为[1-2]. XSS漏洞的产生源于未对来自用户输入的不可信数据进行验证，以及反射回浏览器而没有进行编码或转义的情况下被网络应用程序所应用.由于很多网络应用在开发过程中忽略了必要及有效的输入验证，就容易被跨站脚本攻击.如何在大量网页中找到隐藏的漏洞注入点并进行漏洞检测是防范XSS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会由Ajax和JavaScript生成，人工检测漏洞注入点显然是不现实的，需要尽可能地采用自动化的方法.目前，针对XSS漏洞检测工具的研究还不是很充分[3-6]，使用静态爬虫框架如果要获得由Ajax生成的内容，需要对页面的HTML代码进行解析，找到发送异步请求的方式以及目标URL来加载Ajax，流程非常复杂，对隐藏式的漏洞注入点的覆盖率很低.本文在设计网页爬虫时，采用了模拟浏览器行为的方法，像用户操作一样让浏览器内核来解析JavaScript和加载Ajax，使漏洞注入点的发现工作更有效和准确.

本文提出基于Headless浏览器模拟浏览器的网络爬虫框架，改善传统静态爬虫对XSS漏洞注入点检测覆盖率小的不足；采用基于Ghost.py库解析JavaScript和加载Ajax，并采用Beautiful Soup作为HTML解析器的解析方法，尽可能多地获取网页中的隐藏式漏洞注入点和非格式化漏洞注入点，扩大漏洞注入点获取的覆盖率.在漏洞攻击测试向量的设计方面，通过综合考虑字符串长度、字符种类等因素对XSS Filter Evasion Cheat Sheet[7]进行优化，并适当应用字符编码对XSS攻击向量进行变形，合理使用冗余的尖括号或标签，以绕过服务器的过滤函数，缩短漏洞检测时间.

1 检测系统设计

本文设计并实现的XSS漏洞检测系统主要

收稿日期： 2016-07-05； **修回日期：** 2017-01-17.
基金项目： 国家自然科学基金重大研究计划培育项目(91546111)；北京市自然科学基金资助项目(4173072).
作者简介： 王 丹*(1969-)，女，教授，E-mail：neuwd@sina.com.cn.

由爬虫模块和漏洞检测模块组成.

1.1　爬虫模块

爬虫模块负责将网页中的 URL 爬取出来供漏洞检测模块依次进行检测. 本文使用开源的 Ghost. py 作为引擎,采用迭代的深度优先爬虫策略,不断地循环抓取网页存入 URL 队列. 由于使用了浏览器引擎的 API,该模块可以动态加载页面,并触发网页中的事件以获取 JavaScript 或 Ajax 生成的新的 URL 存入 URL 队列. 下面描述爬虫算法的设计和事件触发过程.

(1)爬虫算法

网络爬虫按照一定规则爬取特定网页的程序或者脚本,它从一个或者若干初始网页开始,爬取页面中符合规则的链接,然后将这些新链接放入等待爬取的 URL 队列中. 接着遍历这个等待队列,进入下一个循环,直到满足设定好的结束条件后跳出循环终止. 本文构建了轻量的爬虫框架,采用的爬虫算法方案为迭代的深度优先搜索. 这是一种平衡深度优先搜索和宽度优先搜索的一个折中策略,其思想是对深度优先搜索使用一个深度界限,一旦搜索的深度在某个层次以下,深度界限便强制停止对这条路径的搜索. 与深度优先搜索相同,它的空间需求小,但分支可以达到和宽度优先搜索一样的效果,用户可以设定该深度界限,当这个值为 0 时,系统只检测当前页面. 这样一来,方便了用户在单页面的漏洞检测和全站的漏洞检测功能间的切换.

(2)触发事件

为了得到页面中隐藏的 DOM 元素,本文借助 Ghost. py 触发网页中的事件,并通过 Headless 浏览器对 JavaScript 和 Ajax 进行解析. 在这之前,需要找到网页中可以触发的事件,即含有事件属性的标签. 本文选用了 Beautiful Soup 库,可以很方便地搜索带有事件属性的标签,之后再用 Ghost. py 提供的 API 模拟浏览器行为触发事件. 下面的一段代码是找到带有 onclick 事件的标签并模拟点击:

```
list = soup.find_all(onclick=True)
for tag in list:
    ghost.click('''*[onclick="%s"]''' % tag['onclick'], expect_loading=True)
```

这段代码可以将当前页面中所有含有点击事件的标签点击一遍,对事件进行点击后,可能引发浏览器解析 JavaScript 和加载 Ajax,产生 DOM 元素的改变或者 URL 的跳转,对此需要采取不同的方式应对. 如果跳转到新的 URL,存储当前 URL 并返回之前的页面即可;若产生 DOM 元素,则需再次寻找是否出现了新的事件,直至不再产生 DOM 元素为止. 相关过程如算法 1 所示.

算法 1　页面 DOM 元素展开算法

输入:第一次请求得到的页面 HTML 代码

输出:展开后的页面 HTML 代码

步骤 1　获取所有含有事件的标签存入 tag_list,去除重复的标签.

步骤 2　模拟点击 tag_list 中下一个未访问过的标签. 若所有标签都被访问过则结束.

步骤 3　将该标签存入 visit[],标记为访问过.

步骤 4　若页面跳转,执行步骤 5;否则,执行步骤 6.

步骤 5　将跳转后的页面 URL 存入 URL_list,执行步骤 2.

步骤 6　如果 DOM 元素改变,执行步骤 1.

1.2　漏洞检测模块

XSS 漏洞检测模块包含页面分析模块和动态判断模块,同样使用 Ghost. py 作为引擎,通过 Beautiful Soup 库进行页面内容分析,提取注入点. 页面分析模块负责对爬取到的注入点进行提取,动态判断模块对注入点填充攻击向量并提交攻击请求. 每次攻击测试向量被提交后,均由动态判断模块分析结果,判断该注入点是否存在 XSS 漏洞并存储结果. 具体来说,就是用 Ghost. py 库提供的 API 解析执行 JavaScript 进行攻击向量的提交过程,以及根据部分响应页面的响应信息判断 XSS 漏洞是否存在. 下面进行具体介绍.

页面分析模块主要负责提取注入点并将其存入注入点列表中. 如前所述,执行页面中的事件、解析 JavaScript 和 Ajax 可以获取页面中的隐藏注入点. 而对于非格式化注入点,则需要根据一定的匹配规则,将允许用户输入数据的元素和与之对应的负责提交请求的元素从页面中提取出来. 本文对现有的网络环境中可能的注入点和交互点格式进行了分析归纳,并按照这些可能的格式对注入点和交互点进行查找. 注入点与交互点的特征分类如表 1 所示. “input”包含了 HTML5 环境下,input 标签 type 属性全部可能的值,其中一些 type 属性的值可以使 input 允许用户输入数据,它们位于表中的注入点一列;而另一些 type 属性的值可以使 input 成为按钮,允许用户点击,它们在表中位于交互点一列. 此外,注入点还可能以

表1 注入点与交互点的特征分类

Tab.1 Categories of injection point and interaction point

标签	注入点	交互点
input	‘’, ‘text’, ‘hidden’, ‘password’, ‘checkbox’, ‘file’, ‘image’, ‘radio’, ‘number’, ‘color’, ‘date’, ‘datetime’, ‘datetime-local’, ‘month’, ‘week’, ‘time’, ‘email’, ‘range’, ‘search’, ‘tel’, ‘url’	‘button’, ‘submit’, ‘reset’
textarea		
button		‘button’, ‘submit’, ‘reset’
others		‘a’, ‘span’, ‘i’…

textarea标签存在.当注入点是input标签时,它的值存在value属性中,而当注入点是textarea标签时,它的值可以通过innerHTML获得.相比于注入点的格式,交互点的格式更加繁多,因此还需要通过分析页面的DOM结构来进行提取.通常,注入点和其对应的交互点,以如下几种结构存在于HTML页面中.

```
〈input type="text" id="header_search_input" /〉
〈input type="button" onclick="search($('#header_search_input').val())" /〉
```

或者:

```
〈input type="text" id="header_search_input" /〉
〈div〉〈input type="button" onclick="search($('#header_search_input').val())" /〉〈/div〉
```

又或者是更复杂的:

```
〈div〉〈input type="text" id="header_search_input" /〉〈/div〉
〈div〉
〈div〉〈input type="button" onclick="search($('#header_search_input').val())" /〉〈/div〉
〈div〉〈span class="SomeClass"〉Clickable〈/span〉〈/div〉
〈/div〉
```

对于第1种情况,获取注入点时应该将第1个input元素匹配为注入点,并根据表中的信息,从第1个input的兄弟节点找到第2个input,匹配为其对应的交互点.对于第2种情况,如果第1个input的兄弟节点中没有能够匹配为交互点的标签,则从它的子节点中寻找.而对于较为复杂的第3种情况,第1个input的兄弟节点和子节点中均没有能够匹配为交互点的标签,则从第1个input的父节点开始,按照之前的规则再寻找一遍.此时,第2个input标签和span标签都有可能是注入点对应的交互点,因此,将它们都存入input的数据结构中,之后的动态判断模块会确定哪一个可以执行提交请求的功能.网页中的XSS漏洞注入点的解析过程如算法2所示.

算法2 网页中的XSS漏洞注入点的解析

输入:网页源码

输出:注入点列表

步骤1 按照表1总结的注入点特征找到所有的标签,存入Input列表.

步骤2 找到所有表单,存入Form列表.

步骤3 对于Form列表中每个元素,同样按照注入点特征找到该表单中的标签,并保存在该表单的辅助标签属性中.

步骤4 从Input列表中去掉和表单元素中相同的注入点标签.

步骤5 对Input列表中每一个元素

①设置当前位置cur为该Input元素在DOM树中的位置.

②按照表1总结的交互点特征,从cur的所有兄弟节点及其子节点中找到交互点,并保存在该注入点的对应交互点列表中.如果不存在交互点,执行③.

③将cur赋值为cur的父节点,执行②.

步骤6 结束.

2 XSS漏洞攻击向量

目前,大部分网站都会对用户提交的数据进行一定程度的拦截、过滤和净化[8-9],这种处理方式虽然在一定程度上缓解了攻击的发生,但是仍不能从根本上消除漏洞[10].为了提高XSS漏洞动态检测的效果,本文针对网络应用中常见的数据过滤机制进行了研究,采取了一些过滤机制的逃逸技术[11],并应用这些技术对攻击向量进行优化,将其应用于提交攻击向量阶段中.过滤机制逃逸技术包括测试脚本编码、混淆字符、JavaScript函数、层叠样式表、畸形标签等方面的处理.最终,攻击向量成为经过优化后的XSS Filter Evasion Cheat Sheet,相对于传统的XSS Filter Evasion Cheat Sheet,测试样例数量少,但效率更高,可以绕过很多XSS过滤器.

2.1 XSS漏洞攻击向量的生成

首先,需要填写XSS攻击向量.本文使用

Ghost.py 提供的填写表单的函数在表单栏填写 XSS 攻击向量，如：

```
ghost.set_field_value("input[name=%s]" % name, xss)
```

考虑到注入点可能会存在前端验证，如限制输入长度、过滤某些字符等，这会导致一些 XSS 攻击向量无法提交. 本文设计了如下的使用 JavaScript 移除相关验证函数的方式绕过这些验证，以开展攻击注入：

```
input.removeAttribute("onfocus");
input.removeAttribute("placeholder");
```

对于格式化注入点，还可以使用如下的 Ghost.py 提供的模拟 JavaScript 语句来提交表单：

```
ghost.evaluate("document.querySelector('form')['submit']();", expect_loading=True)
```

本文设计的自动填充攻击向量并提交的算法可以归纳为算法 3 中的描述.

算法 3　自动填充攻击向量并提交

输入：格式化和非格式化注入点的列表

输出：漏洞检测结果

步骤 1　遍历格式化和非格式化注入点的列表.

步骤 2　如果为格式化注入点，执行步骤 4.

步骤 3　如果为非格式化注入点，遍历其链接的所有交互点.

①点击当前交互点，模拟提交行为.

②如果提交成功，记录该交互点，执行步骤 4.

步骤 4　提交探子请求，如果在响应页面中检测不到探子，执行步骤 1.

步骤 5　遍历保存全部 XSS 攻击向量的列表.

步骤 6　用当前攻击向量填充注入点并提交.

步骤 7　得到响应页面后，根据动态检测模块提供的方法判断是否存在 XSS 漏洞，如果存在执行步骤 8；否则执行步骤 1.

步骤 8　存储漏洞在 DOM 中的位置、当前页面 URL 及其信息.

步骤 9　结束.

2.2　漏洞的动态判别

漏洞的动态判别就是在动态提交攻击向量后，通过对响应结果进行分析和判断，得出漏洞是否存在的结果. 一般的方法是先进行探子请求. 对于探子请求，如果在响应页面中检测不到探子，检测过程就会收到返回值，并跳过对全部预设的 XSS 攻击向量的提交. 如果被测试的某注入点存在 XSS 漏洞攻击，在攻击向量提交给服务器并返回响应页面后，提交的攻击向量将会在浏览器中被解析执行. 其中，可分为如下几种情况：

(1)如果提交的是脚本型的攻击向量，响应页面会执行“alert('XSS');”代码. 此时，本文动态判断模块将调用 Ghost.py 的函数 wait_for_alert()，该函数可检测响应页面是否执行了“alert('XSS');”代码，如果执行了，则可认为该注入点存在 XSS 漏洞.

(2)如果提交的是资源型攻击向量，如攻击系统资源、攻击网络带宽等，响应页面会向指定的地址即 http://wedge.sinaapp.com 发送一个请求，攻击向量不同，请求的路径也不同. 不同的请求会调用响应逻辑将第三方服务器的数据库中某一行数据修改为不同的值. 之后，本文的漏洞检测模块会发出一个对第三方服务器的请求以获得这一行数据，如果这一行数据被修改成了请求中预设的值，那么则可认为该注入点存在 XSS 漏洞.

3　测试与结果分析

本文将 DVWA[12] 作为测试程序，DVWA 是用 PHP 和 MySQL 编写的一套用于常规漏洞教学和检测的脆弱性测试程序，包含了 SQL 注入、XSS 等常见的一些安全漏洞. 同时，本文在本地搭建了 PHP 和 MySQL 环境并使用 XAMPP，它是一个集成了 Apache、MySQL、PHP 以及 PERL 的功能强大的建站集成软件包，支持 Windows、Linux 和 OS X 操作系统，并提供 phpMyAdmin、Webalizer 等搭建网络环境的常用工具及组件. 在本地安装完 XAMPP 和 DVWA 后，将 DVWA 应用移动到 XAMPP 安装目录下的 htdocs 文件夹中，搭建 DVWA 的工作就完成了. 启动 XAMPP 应用，输入网址 http://127.0.0.1/dvwa 即可查看首页.

3.1　功能测试

本文首先对 DVWA 进行漏洞检测. 通过 Ghost.py 提供的 API 模拟登录，并将脆弱性测试程序的安全等级设置为中等：

```
ghost.open('http://127.0.0.1/dvwa/')
ghost.fill("form", {"username": 'admin', "password": 'password'})
ghost.click('input[type=submit]', expect_loading=True)
ghost.open('http://127.0.0.1/dvwa/security.php')
ghost.evaluate("document.getElementsByName('security')[0].value = 'medium';")
ghost.click('input[type=submit]', expect_loading
```

=True)

安全等级为中等时,XSS 表单服务器代码为

```
〈? php
if(! array_key_exists("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == '')
{
$isempty = true;
} else {
echo '〈pre〉';
echo 'Hello ' . str_replace('〈script〉', '', $_GET['name']);
echo '〈/pre〉';
}
?〉
```

从后台逻辑可以看出,符合不带有〈script〉标签且可以执行的脚本都能够绕过该过滤函数. 这一点可从下面程序运行时显示的结果中看出:

```
〈IMG SRC=&#104;&#116;&#116;&#112;&#58;&#47;&#47;&#119;&#101;&#100;&#103;&#101;&#46;&#115;&#105;&#110;&#97;&#97;&#112;&#112;&#46;&#99;&#111;&#109;&#47;&#52;〉
〈BODY ONLOAD=alert('XSS')〉
〈META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');"〉
```

这个结果符合之前对后台代码逻辑的分析,说明使用本文的方法可以成功检测出表单是否具有 XSS 漏洞. 由于篇幅有限,这里不列出检测结果的界面.

之后,将本文设计的系统用于检测整个 DVWA 包含的网页,共计访问了 19 个页面,总共涉及 8 个注入点,测试了 8 个注入点,最终检测出 http://127.0.0.1/dvwa/vulnerabilities/xss_r/ 页面上的表单存在反射型 XSS 漏洞,http://127.0.0.1/dvwa/vulnerabilities/xss_s/ 页面上的表单存在存储型 XSS 漏洞.

3.2 对比测试

本文的设计方案在对比中命名为 XSS scanner based on Headless browser,与 XSS-Me、WebInspect、Acunetix Web Vulnerability Scanner 及 Paros Proxy 进行对比测试. 测试目标网站为南方周末网和大街网.

XSS-Me 是一个用于测试反射型 XSS 漏洞的漏洞检测工具,目前还不支持存储型 XSS 漏洞的测试. 它的原理是在提交表单时,用预定义的 XSS 攻击向量替换表单中的参数值,以进行测试,它可以自动寻找注入点并提交请求.

WebInspect 是一款自动化的网络应用漏洞评估系统. 它支持对目前应用 Web 2.0 技术的网站进行手动辅助检测,具有扫描速度快、安全评估范围广和安全扫描结果准确的优点. 并且许多传统扫描工具无法识别的安全漏洞也可以被 WebInspect 检测到.

Acunetix Web Vulnerability Scanner 简称 AWVS,是一个自动化的网络应用程序安全测试工具,它通过网络爬虫检测网站中含有的常见的安全漏洞,包括 XSS 漏洞、SQL 注入和弱口令等. 优点是扫描速度快、漏洞发现率高.

Paros Proxy 是一个基于 Java 的 HTTP/HTTPS 代理,可以用于检测网络应用程序的漏洞. 它通过网络爬虫爬取整个网站并检测其中的安全漏洞. 它带有内置的实用工具,可以代理 HTTP 业务,支持动态地编辑、查看 HTTP 消息等.

从表 2 中可以看出,对于页面格式较为简单的南方周末网,现有工具基本都能检测出存在 XSS 漏洞,而对于页面格式较为复杂,包含隐藏注入点和非格式化注入点较多的大街网,不含有解析 JavaScript 和分析页面 DOM 格式功能的漏洞检测工具就无法检测出其中的漏洞.

表 2 对比测试

Tab. 2 Comparison test

方法	南方周末网		大街网	
	注入点	漏洞	注入点	漏洞
XSS scanner based on Headless browser	3	1(反射型)	14	1
XSS-Me	2	1	8	0
WebInspect	3	1	12	0
Acunetix Web Vulnerability Scanner	3	1	10	0
Paros Proxy	3	0	8	0

4 结 语

本文的 XSS 漏洞检测方法可以很好地完成包括寻找隐藏式注入点、触发攻击的工作,提高了注入点发现的覆盖率,有效判定其引发的 XSS 漏洞,并且更加方便,易于二次开发. 本文的研究仍需进一步的完善,包括:尝试引入多线程的架构以提高爬虫执行效率,将本文的方法应用于其他注入式漏洞,如 SQL 注入等,可扩大应用范围,并根据应用结果加以完善.

参考文献：

[1] ANTUNES N, VIEIRA M. Enhancing penetration testing with attack signatures and interface monitoring for the detection of injection vulnerabilities in web services [C] // **Proceedings — 2011 IEEE International Conference on Services Computing, SCC 2011**. Piscataway: IEEE Computer Society, 2011:104-111.

[2] 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.
WU Zijing, ZHANG Xianzhong, GUAN Lei, *et al*. Technique for deep discovering XSS vulnerability based on anti-filter rules set and automatic crawler program [J]. **Transactions of Beijing Institute of Technology**, 2012, **32**(4):395-401. (in Chinese)

[3] 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.
SHEN Shouzhong, ZHANG Yuqing. Design and implementation of XSS vulnerability detection tool based on crawler [J]. **Computer Engineering**, 2009, **35**(21):151-154. (in Chinese)

[4] CHEN Janmin, WU Chialun. An automated vulnerability scanner for injection attack based on injection point [C] // **ICS 2010 — International Computer Symposium**. Piscataway: IEEE Computer Society, 2010:113-118.

[5] VAN DEURSEN A, MESBAH A, NEDERLOF A. Crawl-based analysis of web applications: Prospects and challenges [J]. **Science of Computer Programming**, 2015, **97**(P1):173-180.

[6] WAN Fangfang, XIE Xusheng. Mining techniques of XSS vulnerabilities based on web crawler [J]. **Applied Mechanics and Materials**, 2014, **556-562**: 6290-6293.

[7] OWASP. XSS filter evasion cheat sheet [EB/OL]. (2016-10-04). https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet.

[8] SUNDARESWARAN S, SQUICCIARINI A C. XSS-Dec: A hybrid solution to mitigate cross-site scripting attacks [C] // **Data and Applications Security and Privacy XXVI — 26th Annual IFIP WG 11.3 Conference, DBSec 2012, Proceedings**. Heidelberg: Springer Verlag, 2012:223-238.

[9] ANTUNES J, NEVES N, CORREIA M, *et al*. Vulnerability discovery with attack injection [J]. **IEEE Transactions on Software Engineering**, 2010, **36**(3):357-370.

[10] THUMMALAPENTA S, LAKSHMI K V, SINHA S, *et al*. Guided test generation for web applications [C] // **2013 35th International Conference on Software Engineering, ICSE 2013 — Proceedings**. Washington D C: IEEE Computer Society, 2013:162-171.

[11] 王　欣. WEB 应用系统安全检测关键技术研究[D]. 北京:北京邮电大学, 2011.
WANG Xin. Research on key technologies of web application security detection [D]. Beijing: Beijing University of Posts and Telecommunications, 2011. (in Chinese)

[12] Dewhurst Security. Damn Vulnerable Web App (DVWA) [CP/OL]. [2016-01-10]. http://www.dvwa.co.uk/.

XSS vulnerability detection based on user′s behavior simulation

WANG Dan*, LIU Yuan, ZHAO Wenbing, FU Lihua, DU Xiaolin

(College of Computer Science, Beijing University of Technology, Beijing 100124, China)

Abstract: To deal with the problems, such as low coverage of found vulnerability injection points in complex web page, lacking of dynamical analysis for response message from target website faced by the detection system of XSS vulnerability, a method to detect XSS vulnerability based on user′s behavior simulation is proposed to make improvement for the detection system of XSS vulnerability on extracting injection points, generating attack test vector and analyzing response results. By searching for a variety of the unformatted injection points through analyzing web page structure as well as taking into consideration the length of the string and the type of the character, the attack test vector is optimized and it can bypass the server filter function and shorten the vulnerability detection time. Test results show that the proposed method can improve the detection coverage rate of the injection point and the detection effect of XSS vulnerability.

Key words: XSS vulnerability; detection; Headless browser; Ghost.py