XSS Vulnerability Scanning Algorithm Based on Anti-filtering Rules

Bo-wen LIU¹, Jun WANG¹, Jian-yi LIU¹, Ru ZHANG¹, Wen-xin SUN¹ and Yuan-gang YAO²

¹Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China
²China Information Technology Security Evaluation Center

Keywords: Web system, XSS vulnerability, Filtering mechanisms, Vulnerability scanning.

Abstract. In this paper, we investigate the main security vulnerabilities of web system and vulnerability scanning technology and the causes of different web system vulnerabilities are analyzed. We deeply analyze XSS vulnerability’s formation principle, vulnerability utilization. By researching web server filtering mechanism, we put forward XSS vulnerability scanning algorithm based on anti-filtering rules. Then we use this algorithm to expand and optimize the XSS vulnerability test case library. Thus, the coverage of XSS vulnerability scanning is improved and the security of web system is guaranteed.

Introduction

In recent years, the security of web systems has attracted wide attention. The number of vulnerabilities in web system determines its security.[1]

In this paper, we research the filtering mechanism of web server and the corresponding anti-filtering rule set. Based on anti-filtering rules, we propose XSS vulnerability scanning algorithm, thus improving web system security much better.

Relating Work

Jovanovic open source vulnerability scanning tool Pixy [2] for the static analysis in 2006. It completes the web system vulnerability mining work through the detection and analysis by the direction of the data flow. Penetration testing The technology of for dynamic analysis is established by SecuBat. But it only supports limited vulnerability scanning and cannot scan for new vulnerabilities. However, Nessus[3] could be a good solution to this problem. It is designed by Renaud Deriasion and based on C/S architecture. But Nessus system is very bloated due to the long time. The costs are high.

There are also very good vulnerability scanners at home and abroad, such as Web Inspect, WebSOC and so on. However, these scanners usually require high configuration. As a result, they are not applied to web system of small and medium-sized enterprises.

XSS Vulnerability Research

When an application receives untrusted data and returns it to the browser without proper validation and escaping, it will produce a cross-site vulnerability (XSS).[4]

The web system does not check the input data strictly so that the attacker can inject the script fragment by the user input domain, in which these scripts can be accessed by the victim to achieve the purpose of attack [5]. The above is what is XSS.

According to their different ways of triggering, XSS vulnerability can be divided into three categories.

1. Reflective XSS vulnerability. This type of XSS vulnerability is a non-persistent vulnerability. Vulnerability occurs when the web client uses the server client script to generate page to provide data for the users [6].

2. Storage XSS vulnerability. The attacker first stores the attacking data in the server, which is called storage property. Then the attacker induces users to access vulnerabilities page. Each time
these attacking data have been accessed, they can appear in the client without be processed. It will lead to the emergence of storage XSS vulnerabilities, which is called persistence.

3. XSS vulnerability based on DOM. The attacker modifies the DOM "environment" and executes it in the original client script so that the client's code runs in an unexpected way.

At present, the main XSS vulnerability detection methods are detection based on the source code, detection based on server-side and client-based detection. The static detection technique for the detection of source code is to audit the source code without running the target program. The purpose of the server detection is to confirm the untrusted data and use the filtering measures to handle the dangerous keywords. On the client side, it detects the script language in the browser, prevents malicious script to run, and prompts the user with a reasonable warning.

Web Vulnerability Scanning Algorithm Based on Anti-filtering Rules

Filtering Rules Analysis

Web system developers tend to regard all user data as untrusted data. They deal with it through different ways in order to reduce the security risk of web system. In addition to the firewall outside the web system, IDS, IPS and other ways, the most common usage is to filter and process untrusted data in different web system internal logic layer.

There are three main ways to filter in general: filtering based on keywords, filtering based on encoding, and filtering based on string processing.

Anti-filtering Rules Analysis

The anti-filtering rule set is a set of untrusted data conversion methods. Through the use of one or more methods of the anti-filtering rule set, untrusted data which is submitted by the user can be converted, making it possible to bypass the filtering rules of web system server and execute it normally.

There are five main methods of anti-filtering rule set as follows:

- Content recoding. In web system blacklist, if the key part of the test data is recoded without affecting its execution, it can bypass the type of keyword filtering.
- Code sensitive word processing. Web system will filter a variety of different keywords, just like alert, script and other keywords of XSS vulnerabilities. it must deal with related sensitive words to bypass filtering. Especially in dealing with sensitive words, it could be bypassed through replacement, annotation and other methods.
- Character writing style confusion. Without affecting the implementation, trying to do the mix of capital and lowercase or the mix of full-width and half-width character will achieve the purpose of bypassing the filtering mechanism.
- Code format reconstruction: HTML and JavaScript all support multi-line code format, thus you can try to insert multiple Tab and Enter key to reconstruct format in order to bypass the filtering mechanism.
- Encoding format conversion: There are a lot of character encoding formats, such as ANSI, Unicode, UTF-8, etc. Therefore, it is a good way to bypass the filtering rules by encoding format conversion.

Through the study of the filtering rules, we know how the web system handles the untrusted data. We can carry out the corresponding anti-filtering process according to these processing methods. Hence, the pass rate of the test cases is greatly improved in vulnerability scanning.

Application of the Algorithm in XSS Scanning

XSS Cross-site Injection. The composition of XSS vulnerability case sample library of web system is described as follows:

- XSS test cases of direct injection. We need to list its statements directly.
- Test cases of URL pseudo protocol in JavaScript. We need to list the various attributes that support the URL pseudo protocol, such as src, herf, etc.
For the test cases in HTML, we need to list the commonly used HTML labels and events according to the HTML syntax rules to compose test cases. According to the different scopes, HTML labels are divided into 4 categories [7]. They are window label, form label, image label and other label. At the same time, events are also divided into 5 categories [7] and the specific categories are shown as follows (Table 1).

Table 1. Labels and Events Classification [7].

<table>
<thead>
<tr>
<th>labels and events</th>
<th>list</th>
</tr>
</thead>
<tbody>
<tr>
<td>window label</td>
<td>&lt;body&gt;, &lt;frame&gt;, &lt;iframe&gt;, &lt;frameset&gt;, etc</td>
</tr>
<tr>
<td>form label</td>
<td>&lt;input&gt;, &lt;form&gt;, &lt;textarea&gt;, &lt;button&gt;, &lt;select&gt;, &lt;option&gt;,</td>
</tr>
<tr>
<td></td>
<td>&lt;isindex&gt;, &lt;label&gt;, &lt;fieldset&gt;, &lt;legend&gt;, &lt;optgroup&gt;, &lt;meter&gt;, etc</td>
</tr>
<tr>
<td>Audio and video label</td>
<td>&lt;img&gt;, &lt;audio&gt;, &lt;video&gt;, etc</td>
</tr>
<tr>
<td>other labels</td>
<td>&lt;table&gt;, &lt;meta&gt;, &lt;base&gt;, &lt;object&gt;, &lt;div&gt;, &lt;source&gt;, etc</td>
</tr>
<tr>
<td>window event</td>
<td>&lt;onload&gt;, &lt;onunload&gt;, etc</td>
</tr>
<tr>
<td>form event</td>
<td>&lt;onchange&gt;, &lt;onsubmit&gt;, &lt;onreset&gt;, &lt;onselect&gt;, &lt;onfocus&gt;,</td>
</tr>
<tr>
<td></td>
<td>&lt;onblur&gt;, &lt;onscroll&gt;, etc</td>
</tr>
<tr>
<td>image event</td>
<td>&lt;onabort&gt;, &lt;onresize&gt;, etc</td>
</tr>
<tr>
<td>keyboard and mouse event</td>
<td>&lt;onkeydown&gt;, &lt;onkeypress&gt;, &lt;onclick&gt;, &lt;ondblclick&gt;,</td>
</tr>
<tr>
<td></td>
<td>&lt;onmousemove&gt;, &lt;onmousedown&gt;, &lt;onmouseout&gt;, &lt;onmouseover&gt;,</td>
</tr>
<tr>
<td></td>
<td>&lt;onmouseup&gt;, &lt;onerror&gt;, etc</td>
</tr>
<tr>
<td>other events</td>
<td>&lt;onerror&gt;, etc</td>
</tr>
</tbody>
</table>

The Way of Attack Data Optimization. Through the analysis of the current filtering rules and the way of anti-filtering, we could implement a targeted change to test cases to improve the test case library. The anti-filtering rules represent the conversion of the XSS code, not only can be used alone, but also mixed with several different ways to use.

Mode 1: Code sensitive word processing. For sensitive word filtering methods, such as masking, forbidding execution, we can bypass the filtering by decomposing the sensitive words or adding comments to code.

For example, The original code and after inserting comments, the code is shown below.

```html
< div style="width:expression( alert( 'xss') ) ; ">
< div style="width:expression( alert( 'xss') ) ; ">
```

Mode 2: Character writing style confusion. The case requirements of HTML language are not strict while Javascript is case-sensitive. So mixed writing in the code can break some of the filtering behavior on XSS. For example:

```html
< img src ="JavaSCript: alert( /XSS/ )"> 
< img src ="JavaSCript: alert( /XSS/ )"> 
```

Mode 3: Content re-encoding or transcoding. So we can bypass the filtering rules by re-encoding or converting the encoding format of the corresponding code or labels. For example:

```
```

Mode 4: Code format reconstruction. Html code and Javascript code all support multiple lines of code format, so we can bypass the server filtering by code format reconstruction. For example, regarding `<script>alert('test')</script>`, we could convert it as follows.

```html
< script>alert('test')
```

Towards the web system common security filtering, we can use the above four methods optimize the test samples which need to send. They expand the test sample library and enhance the XSS vulnerability detection results.
Experiment

Experimental Method Design

In order to verify the application effect of the anti-filtering rules in the XSS vulnerability scanning, the anti-filtering rules conversion module is added to XSS vulnerability scanning process. If the original test cases can not trigger the vulnerability, it transforms according to the rules of the anti-filtering and does retest until all transformation modes are tried. It is mainly divided into three parts: the web crawler, the test preparation and the vulnerability detection.

The first part is the web crawler. It is mainly responsible for crawling data in the URL of the beginning site. It also analyzes the content of the page to extract suspected injection point. The second part is the test case library preparation. This module will generate test cases according to the returned results of the probe data. It will carry out the vulnerability scanning confirmation. As a comparison, we extend the anti-filtering rule set according to the generated test cases. Then we also scan site vulnerabilities and finally complete the results comparison. The third part is vulnerability scanning. The injection point which is extracted by the crawler is connected with the corresponding test case. We submit data to the server through GET and POST and compare and analyze its return results. If the test cases have been successfully returned to the HTML page, we know that it exists XSS cross-site vulnerabilities. Then its corresponding website URL and the test case are stored and recorded.

We implement 12 times scans for 6 web sites, including 6 times for the test case optimization without using anti-filtering rules. As a contrast, the other 6 times optimize test cases using anti-filtering rules.

Experimental Result

During the experiment, we mainly aim at the efficiency and the time of XSS vulnerability scanning. Then we make a comprehensive comparison between scan coverage and scan efficiency. Finally, we prove the advantages of anti-filtering rules that are applied to the cross-site vulnerability scanning.

Specific experimental data are shown as follows (Taken website security into consideration, we do not post the website address.). Specific vulnerability scanning results and scan time are shown in Table 2.

<table>
<thead>
<tr>
<th>site</th>
<th>Vulnerability numbers without anti-filtering rules</th>
<th>Vulnerability numbers with anti-filtering rules</th>
<th>Scan time without anti-filtering rules(second)</th>
<th>Scan time without anti-filtering rules(second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>website 1</td>
<td>3</td>
<td>4</td>
<td>387</td>
<td>436</td>
</tr>
<tr>
<td>website 2</td>
<td>1</td>
<td>1</td>
<td>296</td>
<td>398</td>
</tr>
<tr>
<td>website 3</td>
<td>5</td>
<td>7</td>
<td>524</td>
<td>713</td>
</tr>
<tr>
<td>website 4</td>
<td>3</td>
<td>5</td>
<td>368</td>
<td>446</td>
</tr>
<tr>
<td>website 5</td>
<td>2</td>
<td>2</td>
<td>304</td>
<td>453</td>
</tr>
<tr>
<td>website 6</td>
<td>4</td>
<td>5</td>
<td>375</td>
<td>502</td>
</tr>
</tbody>
</table>

Experimental Data Analysis

During the experiment, we mainly record the effect of XSS vulnerability scanning and the time required for scanning. From a comprehensive comparison of scan coverage and scan efficiency, we validate the advantages of anti-filtering rules for cross-site vulnerability scanning and its applicability in other vulnerability scans.

The corresponding vulnerability scanning as shown in Figure 1(a), and the scan time is shown in Figure 1(b).
By observing the scan time consumption, the following conclusions can be drawn:
1. The coverage of XSS vulnerabilities has been improved by the anti-filtering rules.
2. The time of vulnerability scanning using the anti-filtering rules has an increase. The reason is using test cases after using anti-filtering rules leading the test data packages have an increase.
3. Through the comparison experiment, it has been found that the scanning efficiency has been greatly improved. The method has strong feasibility for small and medium-sized enterprises.

Summary

This paper summarizes the causes, classification and harm of web system vulnerability. It also analyzes the principle, characteristics, harm and defense of XSS vulnerability. The web vulnerability scanning algorithm which is based on anti-filtering rules is proposed. The core of the algorithm is establishing the anti-filtering rules set through the analysis of the filtering rules of the web system. The anti-filtering rule set is established and it is composed of untrusted data conversion method. The data is processed with content recoding and other methods in rule set. But it still maintain its basic function after filtering. We apply this algorithm to optimize the test case libraries of XSS vulnerability scanning needing. Thus coverage of vulnerability scanning is enhanced and we get a more secure web system.

Acknowledgement

The authors thank the editor and reviewers for their suggestions to improve the quality of paper. This work was supported by The National Key Research and Development Program of China under Grant 2016YFB0800903, the NSF of China (U1636112, U1636212) and NSF of China (U1536118).

References